

Building infrastructure with AWS

no code
**Wix Laravel
API**

System Development Course
English version

by Shunta Sako

Recommended for **Laravel9 support**
NoCode development!

Detailed explanation from
infrastructure construction



**No HTML and CSS
coding required!**
**API integration with
Django in Wix**

**Wix Laravel API
System Development Course
English Version**

Shunta Sako

preface

Before viewing this book, please note that if you are viewing this book on a Kindle for PC, the images may not display properly because it takes too long for the kindle for PC to download the images of this book. We recommend using a tablet, smartphone, or Kindle device to view this book.

This book is for those who learn system development with Wix, a no-code tool that does not use HTML and CSS for screen design. This book describes know-how for intuitive and efficient system development without designing by coding by using Wix instead of screen design using HTML and CSS as usual.

We will develop the system by building the infrastructure using AWS, building the Laravel framework on the infrastructure, and using Laravel's API to interface the API with Wix. We will assume that you have a minimum understanding of PHP, the language for development, due to its nature.

The target audience for this book is the following people

- Those who want to build infrastructure for their services on AWS.
- Laravel learning beginner
- Beginner in system development
- Those who want to develop systems with Wix
- Those who want to develop systems using LaravelAPI
- Beginner in learning PHP

Learn this book in video

https://www.udemy.com/course/wix-laravel-api/?couponCode=COUPON_20220421



Table of Contents

[preface](#)

[Learn this book in video](#)

[Table of Contents](#)

[Chapter 1 Introduction](#)

[Overview of this course and what you will learn](#)

[Chapter 2 Getting Started with AWS](#)

[Create an AWS account](#)

[Let's set up fee alerts in CloudWatch](#)

[Let's create a working user with IAM](#)

[Chapter 3: Build Your Network](#)

[Let's create a VPC](#)

[Let's create a subnet](#)

[Let's set up routing](#)

[Chapter 4: Let's Build a Web Server](#)

[Let's set up an EC2 instance](#)

[Let's fix your IP address with ElasticIP address](#)

[Install Visual Studio Code](#)

[Connect to an EC2 instance with Visual Studio Code](#)

[Let's install Laravel on an EC2 instance](#)

[Let's see if Laravel is displayed on the screen](#)

[Chapter 5: Convert your website to HTTPS with AWS](#)

[Get your domain name for free](#)

[Let's create a hosted zone](#)

[Get a free SSL certificate with AWS Certificate Manager](#)

[Let's set up a load balancer with AWS ELB](#)

[Let's connect the subnet for the load balancer to the Internet gateway](#)

[Let's set up an ELB with Route 53](#)

[Chapter 6: Let's build a DB server](#)

[Let's create a private subnet for RDS](#)

[Let's create a security group for RDS](#)

[Let's create a subnet group for RDS](#)

[Let's create a DB parameter group for RDS](#)

[Let's install RDS](#)

[Chapter 7 Creating Screens with Wix](#)

[Let's create a Wix account](#)

[Basic editing screen functions](#)

[Basic operation and explanation](#)

[The concept of the strip and its basic usage](#)

[Let's check the screen of the application to be created](#)

[Let's create a strip for the menu](#)

[Let's create a strip for the tweet search screen](#)

[Let's create a strip for saving tweets](#)

[Let's create a strip for searching saved tweet data](#)

[Let's create a strip for the data table](#)

[Let's create a strip for displaying selected data](#)

[Let's create a strip for detailed analysis](#)

[Let's create a strip for storing notes](#)

[Let's create a confirmation screen for deletion](#)

[Let's create a preloader](#)

[Let's adjust the page design](#)

[Let's create an API call function](#)

[getBazzReachApi.js](#)

[Let's create an event function](#)

[home.js](#)

[Use the log function to check the log](#)

[Chapter 8: Create a Wix Application with Laravel API](#)

[Let's set up a connection from Laravel to RDS](#)

[About the MVC Model](#)

[Create Controllers and Models](#)

[BazzReachController.php](#)

[BazzReach.php](#)

[create bazz reaches table.php](#)

[Let's configure Laravel routing](#)

[Cors.php](#)

[Kernel.php](#)

[api.php](#)

[Let's perform migration](#)

[Connect to RDS with PGAdmin](#)

[Let's create a tweet search function with TweeterAPI](#)

[TwitterApi.php](#)

[Let's show search tweets](#)

[BazzReachController.php](#)

[getBazzReachApi.js](#)

[home.js](#)

Let's register your search tweets in the DB

BazzReachController.php

getBazzReachApi.jsw

home.js

Let's display the registered tweets in a table

BazzReachController.php

getBazzReachApi.jsw

home.js

Let's display registered tweets with partial search

home.js

Let's create a memo function for registered tweets

BazzReachController.php

getBazzReachApi.jsw

home.js

Let's create an analysis function for registered tweets

BazzReach.php

Sentiment.php

KeyPhrase.php

add sentiment to bazz reaches table.php

create sentiments table.php

create key phrases table.php

ComprehendApi.php

BazzReachController.php

getBazzReachApi.jsw

home.js

Let's view the analysis results of the registered tweets

BazzReachController.php

[getBazzReachApi.jsw](#)

[home.js](#)

[Let's create a function to delete registered tweets](#)

[BazzReachController.php](#)

[getBazzReachApi.jsw](#)

[home.js](#)

[Let's check and adjust the entire application](#)

[home.js](#)

[postscript](#)

[Author Profile](#)

[Learn about this book in the video](#)

Chapter 1 Introduction

Overview of this course and what you will learn

Thank you for purchasing the Wix Laravel API System Development Course. I am Shunta SAKO, the developer of this course. Thank you very much for your cooperation. In this course, I would like to share with you how to build infrastructure using AWS, create Laravel API on AWS, and develop a system using Wix and Laravel. First, let me give you an overview of this course and what you will learn.

Overview of this course and what you will learn

- Can build his own AWS infrastructure for services
- You can learn to develop systems in Laravel
- Create applications while designing intuitively with Wix
- Ability to create services in Laravel on built infrastructure

The first one is that you will be able to build your own AWS infrastructure for your services. Many people these days want to be able to release their services and earn an income. Many people are learning to program for this purpose, but in reality, the service you create needs to be released to a

server. Therefore, you need to rent a server, but the rented server has many restrictions and is not flexible enough. With AWS, however, you can build your infrastructure environment as you like.

Second, you will be able to learn how to develop systems in Laravel, which is the world's number one PHP framework and the most widely used system development framework. Laravel includes basic functions, a certain amount of programming is automatically generated, and the generated program code can be freely written, so if you want to extend it, you can do so as you wish. The system can be used as desired. In this case, we will create an API to work with Wix. API stands for application program interface and refers to an interface between software, programs, and web services. In this article, we will implement the API functionality available in Laravel.

The third is to be able to design and create applications intuitively in Wix. I often ask my students to create some kind of system work, and there were always a great many students who had trouble with the design at such times. They want to create a system like this, but they can't get the design right, and they ask, "Why do I have to code when I can create the design in the first place?" Many students said, "I'm sorry, I don't know what to do.

Wix is the solution to such problems: intuitive design and system development instead of HTML-like coding design. Wix is an open development platform for building web applications with React.js front-end and node.js backend. This platform will allow for smooth system development.

Fourth, you can create an application by creating an API in Laravel and calling that API in Wix on the infrastructure you have built. If you can create an API in Laravel and call it in Wix on the AWS infrastructure you have built, you are already running as a service. Once the service is up and running, you can add your own better services and brush upon them.

These are the four things you will outline and learn in this course. We will now begin the actual work in the next chapter.

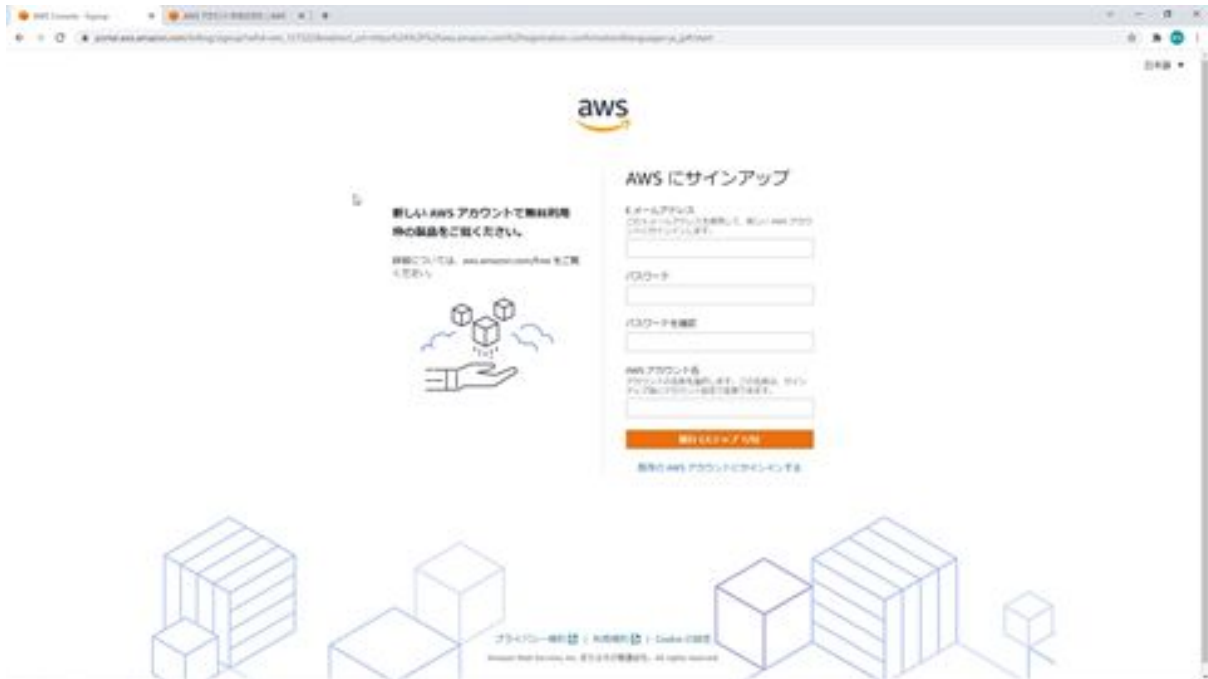
Chapter 2 Getting Started with AWS

Create an AWS account

The first step is to register for an AWS account. First, search for AWS on google. The AWS website will appear at the top of the list.



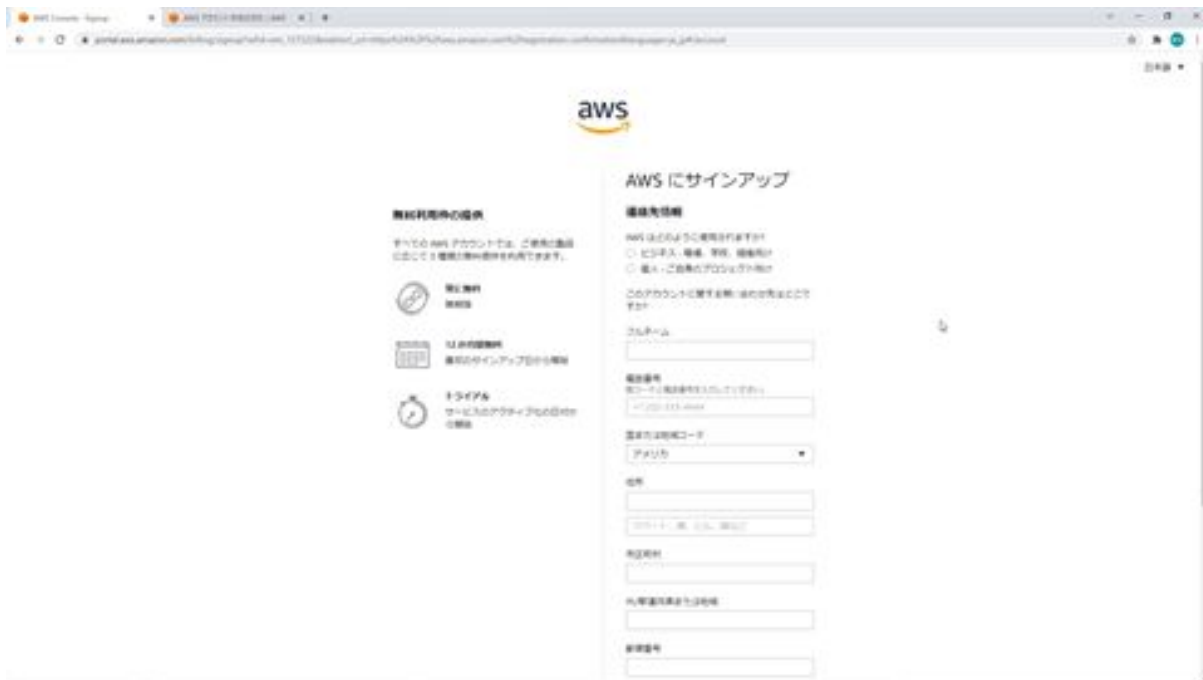
Next, click on the "Create Free Account" button.



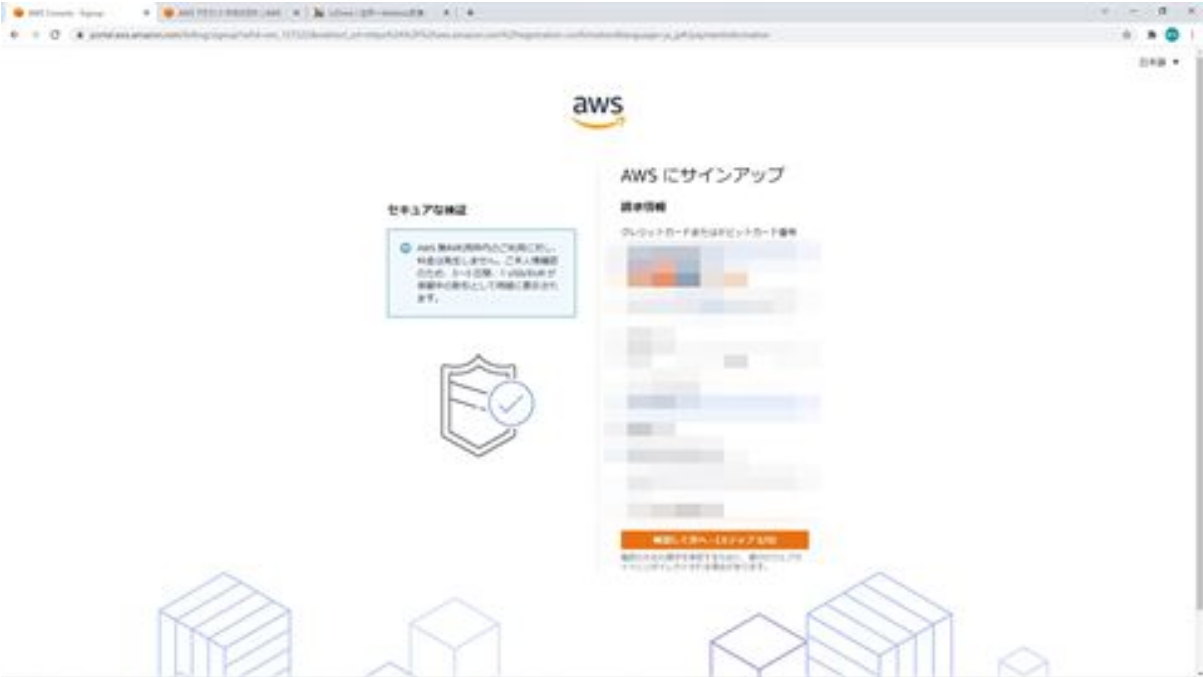
Once you have done so, you will enter the information required to sign up here. For the account creation process, there is a site in AWS that describes how to create an account, so please refer to this site to enter your information. First, on the very first screen, enter your email address and password, followed by your AWS account name. Enter the account name in half-width alphabetical characters.



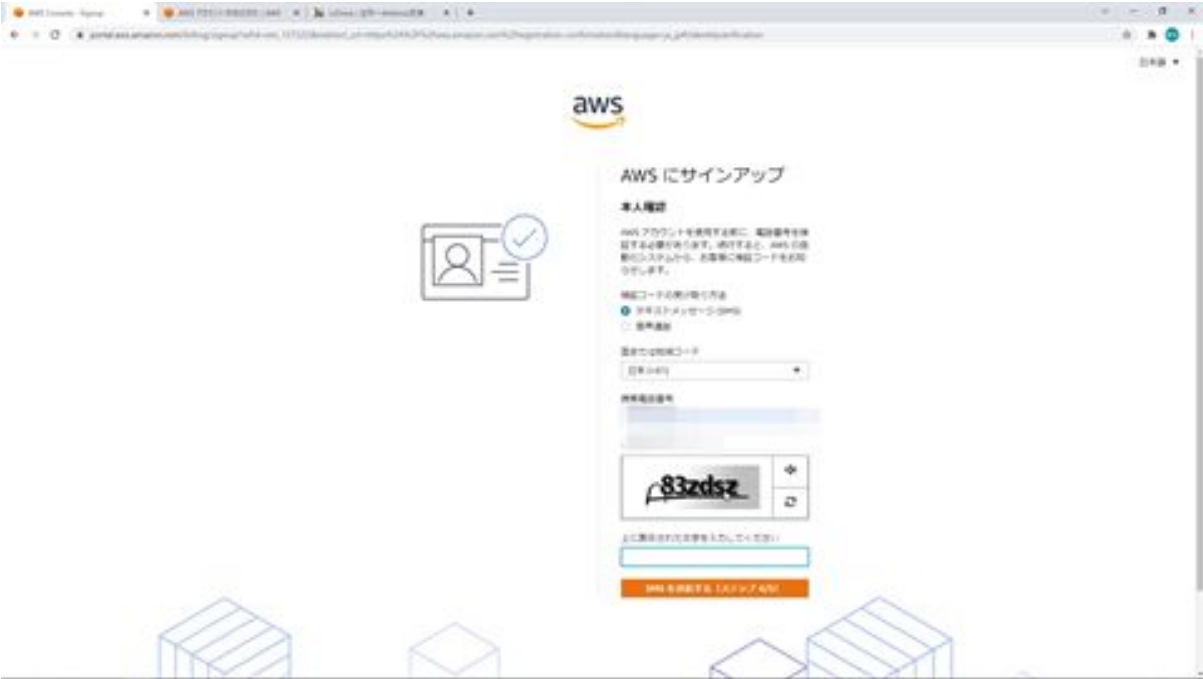
Next, enter your contact information. First, select whether you are a corporate or a sole proprietorship and enter your name, address, phone number, and other information.



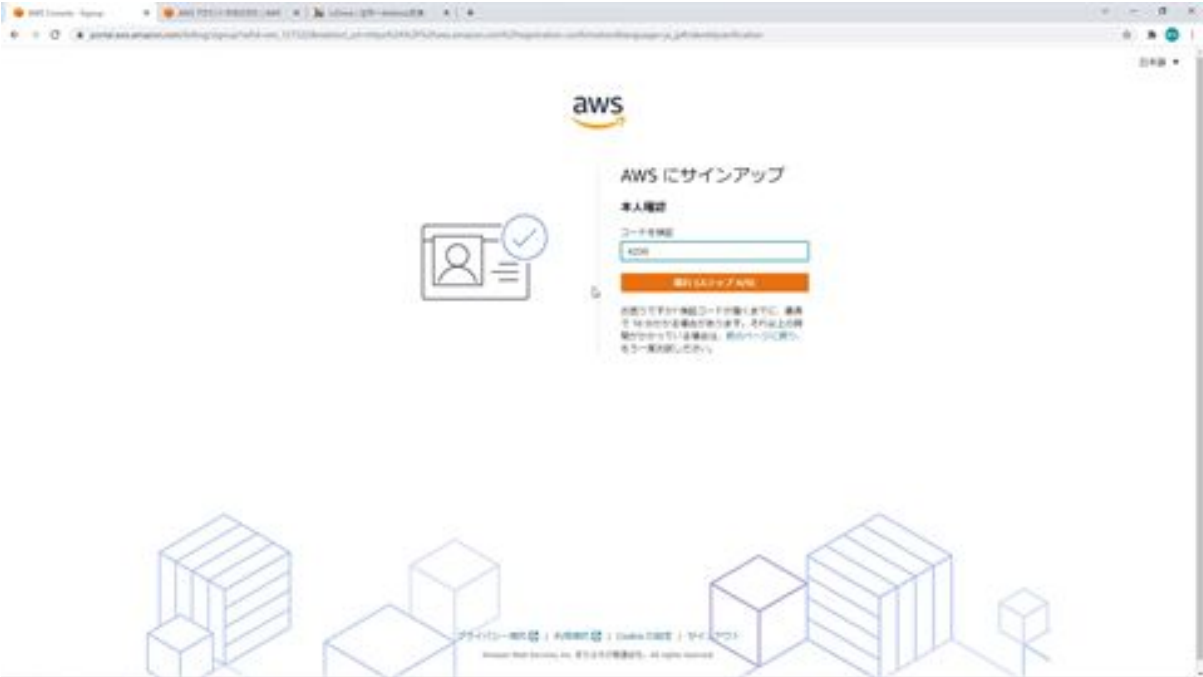
Next, enter billing information. This is the same as entering regular credit card information, so register your credit or debit card information. Then confirm and click the Next button.



Next, enter your identification information. Here you will be asked to confirm your identity by SMS or voice call, so choose either SMS or voice call, select Japan here, enter your phone number, and then enter the letters shown here, which means security check.



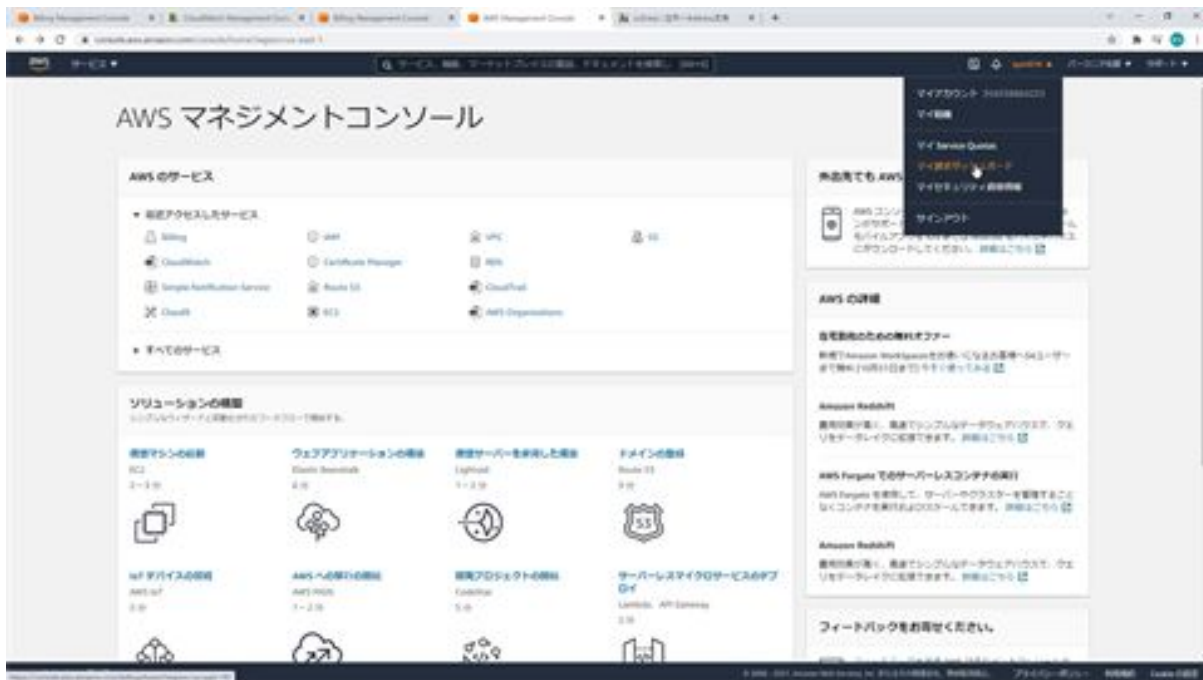
You will then receive an authentication code by SMS to the phone number you entered. After entering the code, click the Continue button.



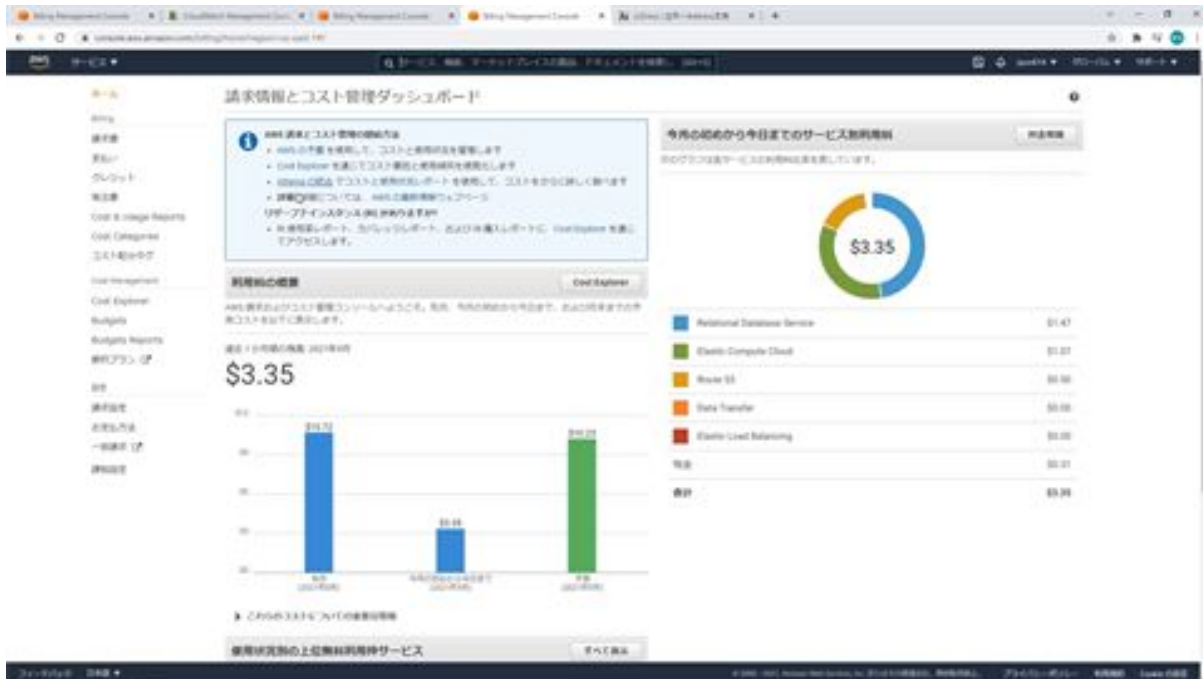
Then, finally, you will select a support plan. In this case, we will select the Basic Support free plan for personal use.

Let's set up fee alerts in CloudWatch

Yes, then next, I would like to set up a fee alert in AWS, which is essentially a pay-as-you-go system, so if you find yourself using too much, you may end up with an unexpected bill. To prevent this from happening, I would like to set up a fee alert. First, click on My Billing Dashboard from the menu above.



Once you have done so, click on Billing Setup on the left.



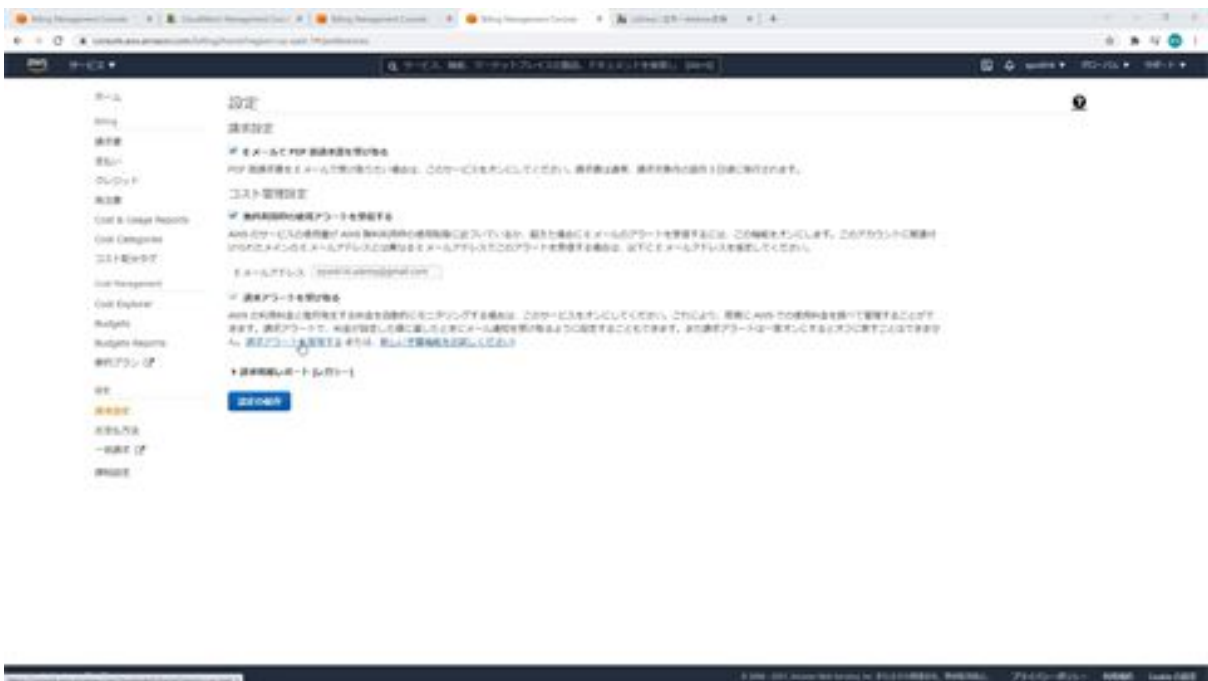
First, check the box at the top to receive PDF invoices via email. Check here to receive a PDF version of your invoice. Next, check the box to receive alerts about the use of your free usage allowance.

When you create a new AWS account, you are entitled to one year of free use. We would like to proceed with this lecture while using this free usage allowance. The free usage limit is the amount of money you can spend each month for one year. If you are about to exceed that quota, the check will alert you, which will be very useful, so we will check this box.

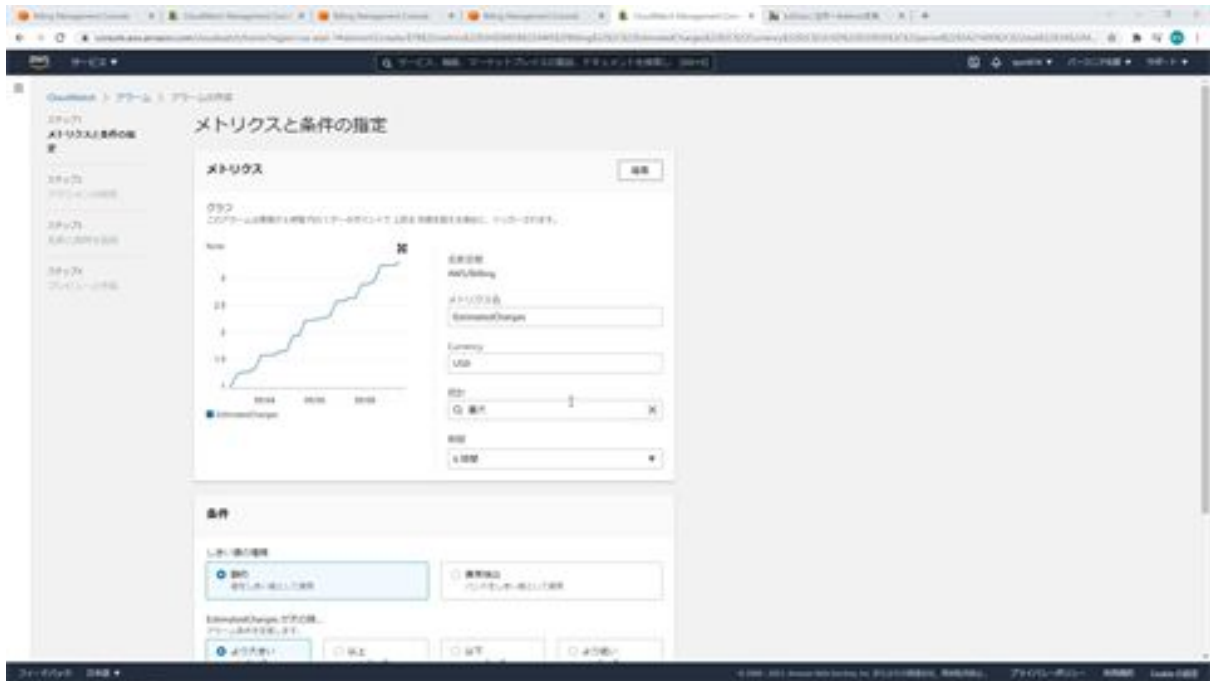
Next, after checking the box, enter your e-mail address in the e-mail address section. Next, you will also check the "Receive billing alerts" box. If you check the "Receive billing alerts" box, you will be able to set up billing alerts, so check this box as well. The billing alert will alert you when you exceed a certain amount of charges. Now, after checking these three checkboxes, click Save Settings.



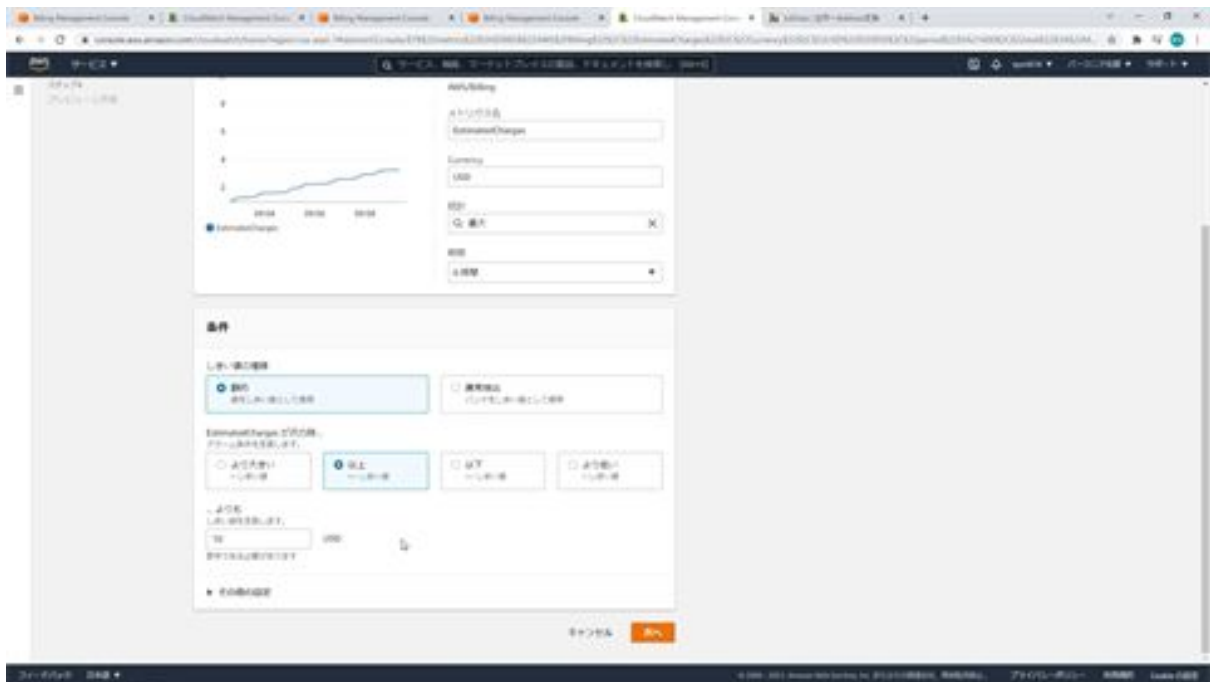
We will now go on to configure CloudWatch, which has a feature called Fee Alerts. Click on Manage Billing Alerts here.



The CloudWatch screen is now displayed, now click on Billing in the menu on the left.



On this screen, we will set up an alert if the fee exceeds \$10.



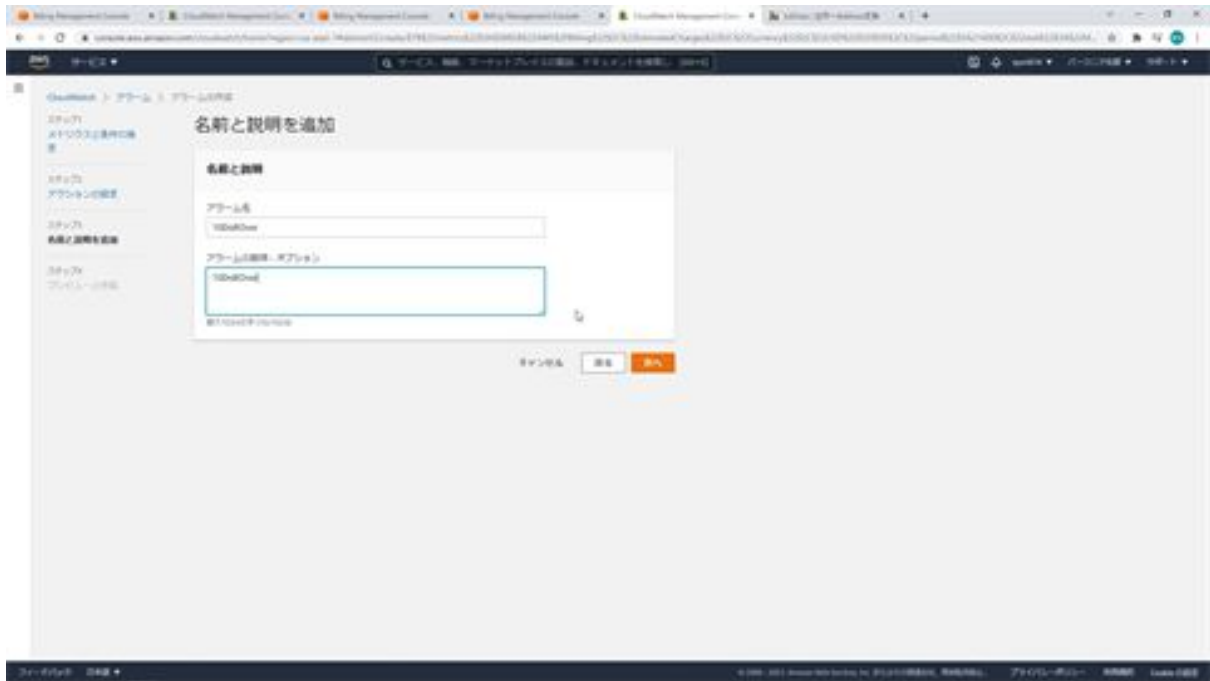
First, the threshold is in dollars, so enter 10. Then, check the above to be alerted if the alarm condition is exceeded, and click Next.



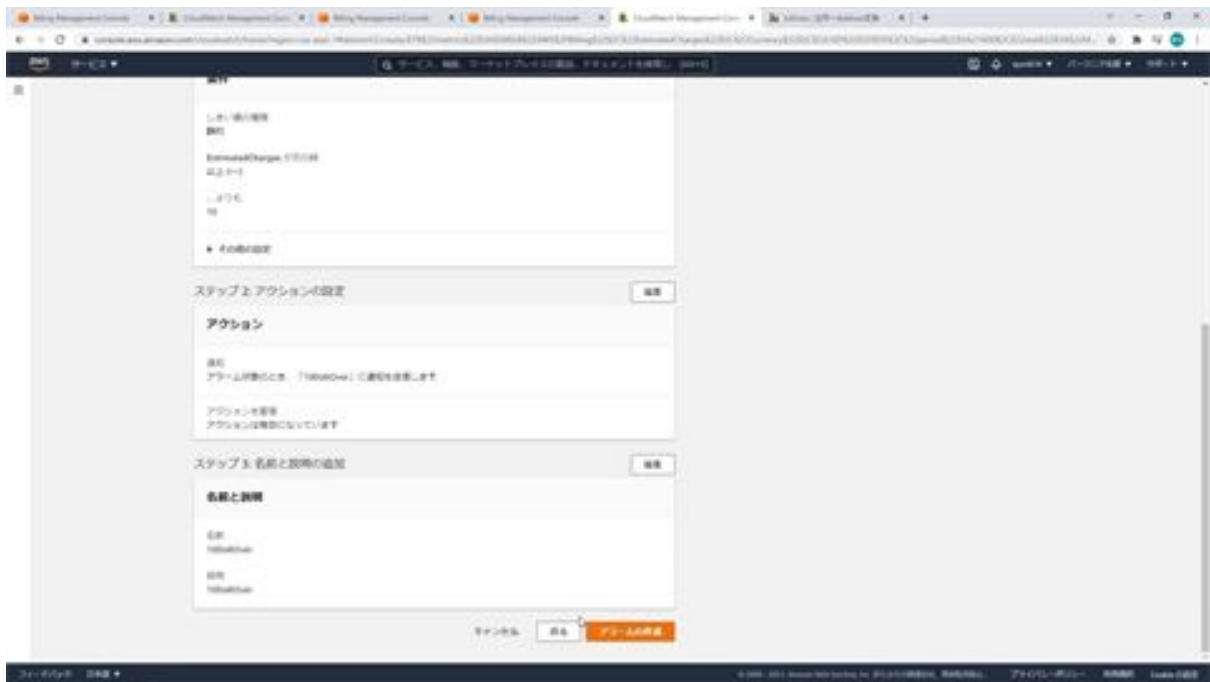
Next is the alarm state trigger, where we define the alarm state that will trigger this action. Here, the threshold of \$10 or more for an alarm is exceeded, so we select the alarm state.

Next, you are asked to select a topic for the SNS, but no topic has been created yet, so select Create New Topic and enter a topic name and email address. After entering the topic name and e-mail address, click the Create Topic button.

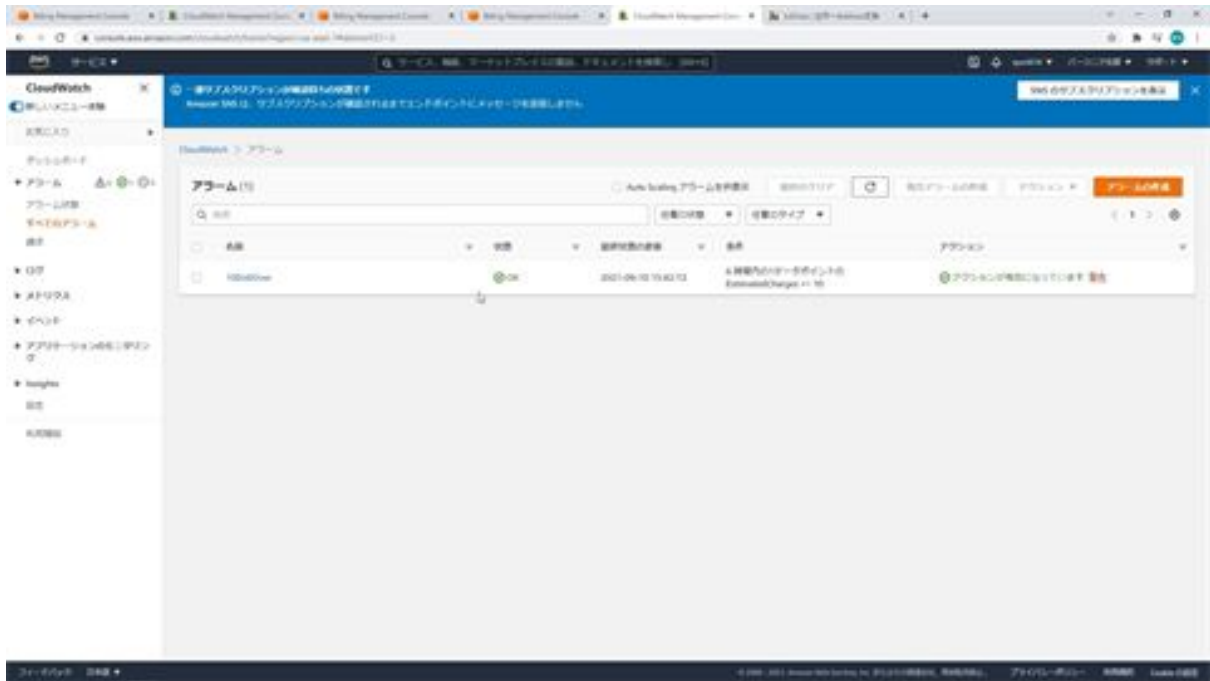
A new topic has now been created. Then click the Next button at the bottom. Next, set the alarm name. Once entered, click the Next button.



Click the Create Alarm button.



Wait a few minutes and the alarm will be ready for use.

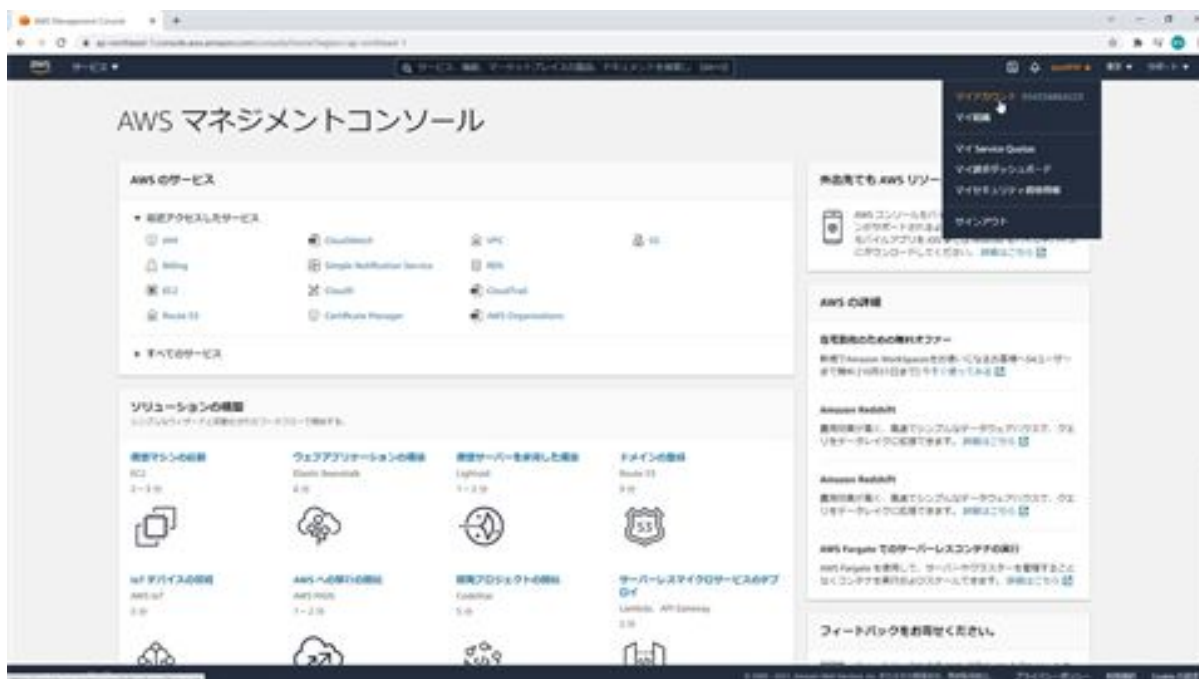


The status is now OK and a new alarm has been created. The alarm is now created and an email is sent to the email address set in the alarm.

Let's create a working user with IAM

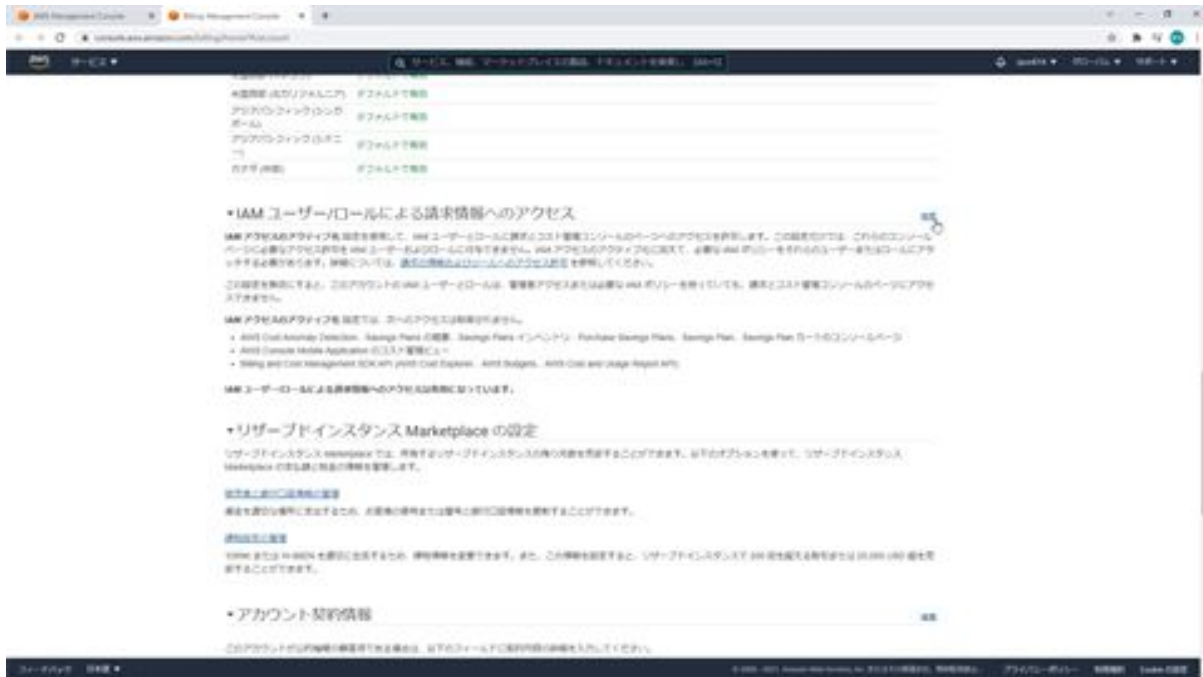
Now we will go on to create a working user for AWS. The user who is currently logged in right now is the root user, a privileged user, which is an account that can do almost anything that can be done with the services related to this account.

Since almost everything can be done, it means that unexpected operations can also be performed. Before we go on to create a working user, we would like to change the billing information so that it can be viewed by the working user. First, click on My Account from the menu above.

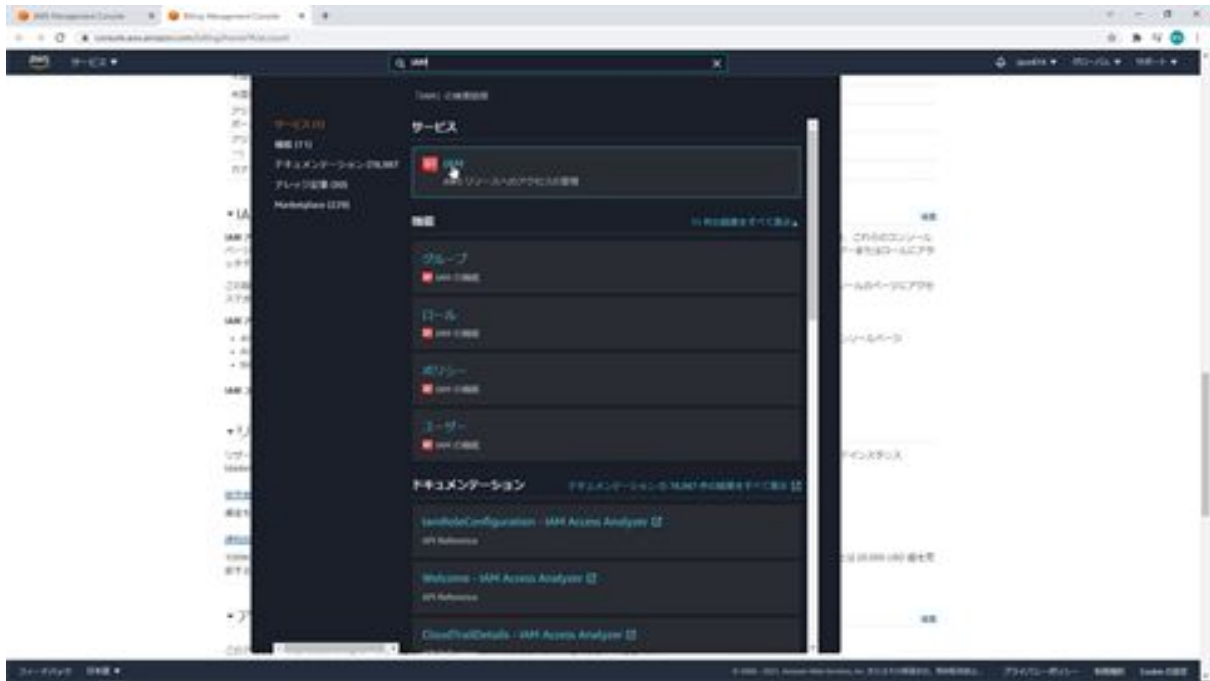


You will then see a section in the center that says Access to billing information by IAM user/role. Click on this edit and check the "Activate

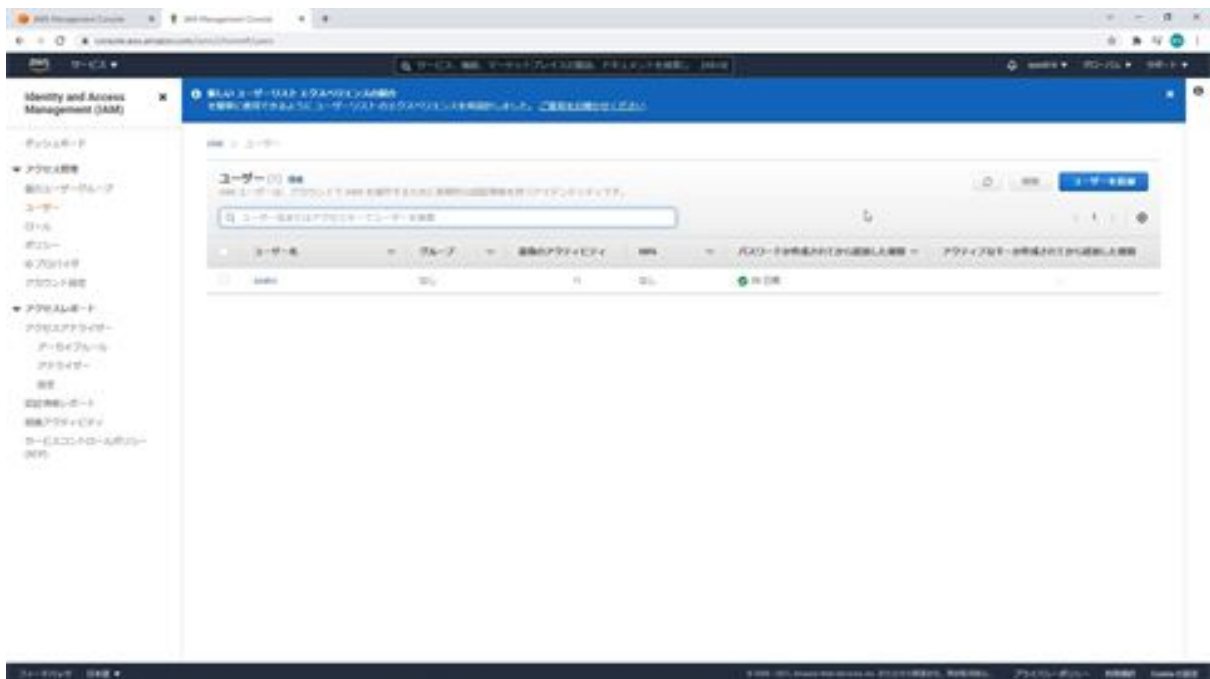
IAM access" checkbox and click on the "Change" button.



Now the working user has access to the billing information. Next, we will create a work user. In Services, type IAM and search. The IAM service will appear at the top of the list.



Next, click on Users in the menu on the left, then click on the Add User button. From this screen, you can add users.



First, enter a user name. In this case, we will enter dev_user. Next, select the type of AWS access. Here, check the Access to AWS Management

Console checkbox.

Next, enter the password for the console. Here you will select a custom password and enter your password. Uncheck the "Password needs to be reset" box. In this case, you are the user who will use the console, so you do not need to reset the password, but if someone else will use the console, it is better to have someone else set the password again, so check the "Password reset is required" box.

By checking this box, you will be prompted to set your password again when someone else logs in, and you will set your password there. Now click the Next Step button.



Next, configure access permissions. Here, click on Attach an existing policy directly and enter administrator in the policy filter. Then you will see "AdministratorAccess" at the top of the list, check this box. This is AdministratorAccess, which is the access permission that allows you to use most services. We are using this one in this case. Click the Next Step button.

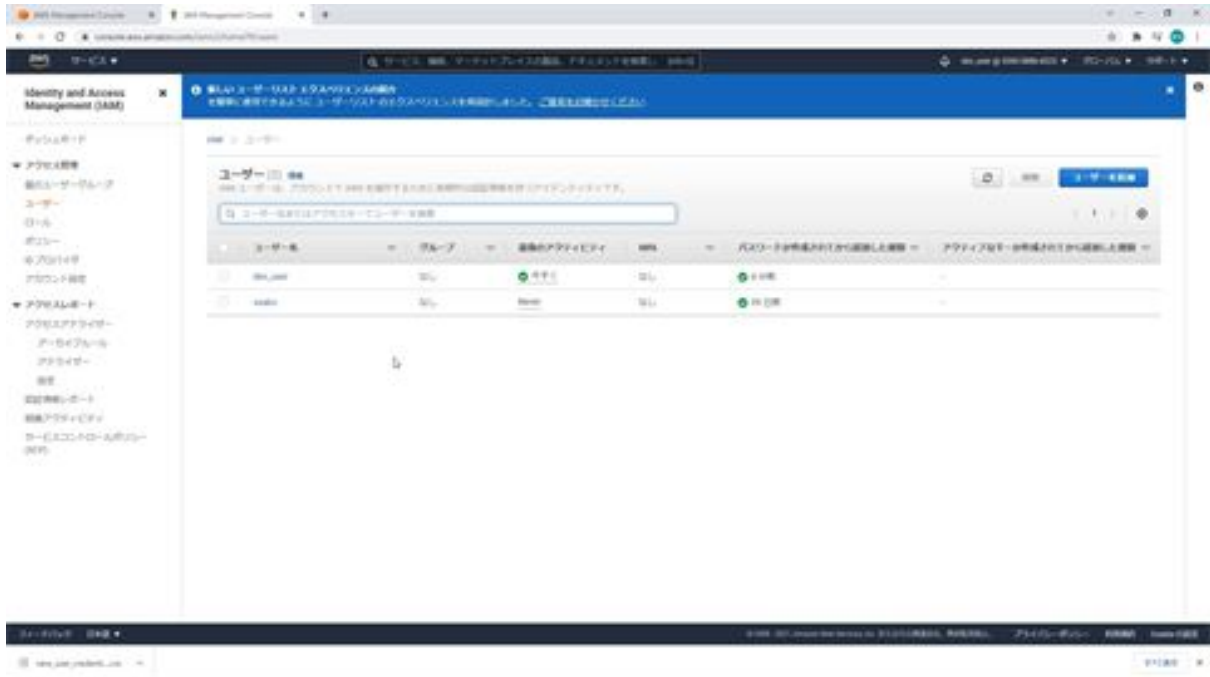
A confirmation screen will then appear. Confirm what is displayed and click the Create User button.



The user has now been added. Once the user has been added, you will see the URL here and the CSV download button, be sure to make a note of this URL, as it is the URL required for login. And this CSV download is a file with the user and URL. Make sure to download this file as well, just in case.



Now we will log in with a working user. The login is completed and the user has been added. This completes the addition of the working user.



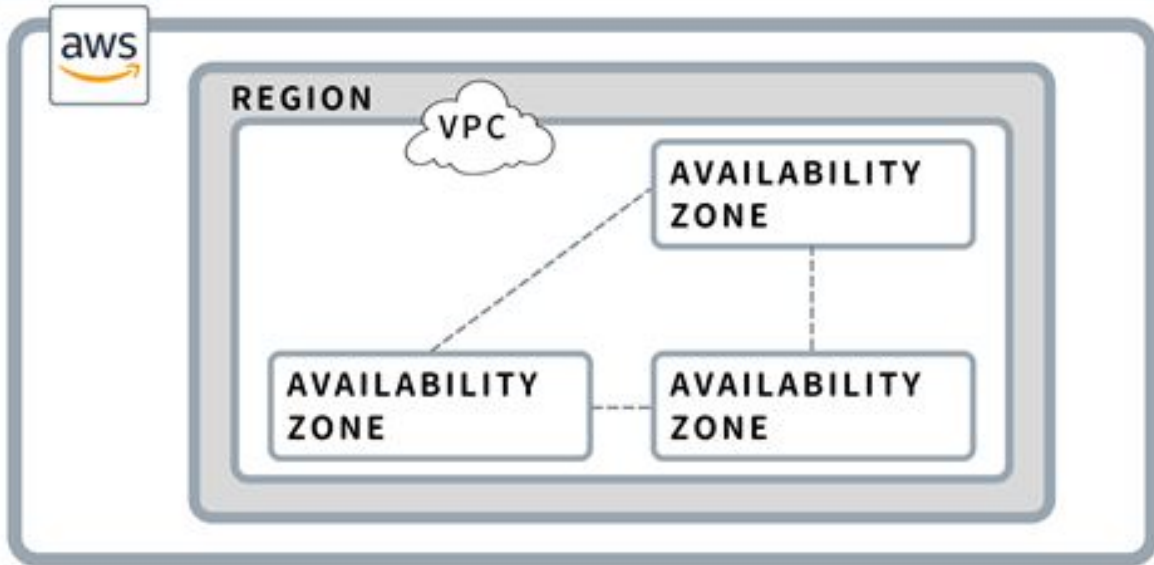
Chapter 3: Build Your Network

Let's create a VPC



Yes, so next we would like to create a virtual network within AWS. Before creating a virtual network, we first need to decide which region we will create our virtual network in. The following is a brief overview of the process. The services available in each region differ, so it is necessary to check in advance whether the service you want to use is available in the region you have selected. If you plan to deploy services in Japan, the Tokyo region is recommended, as the response time will be shorter if the AWS servers are located in Japan rather than outside of Japan.

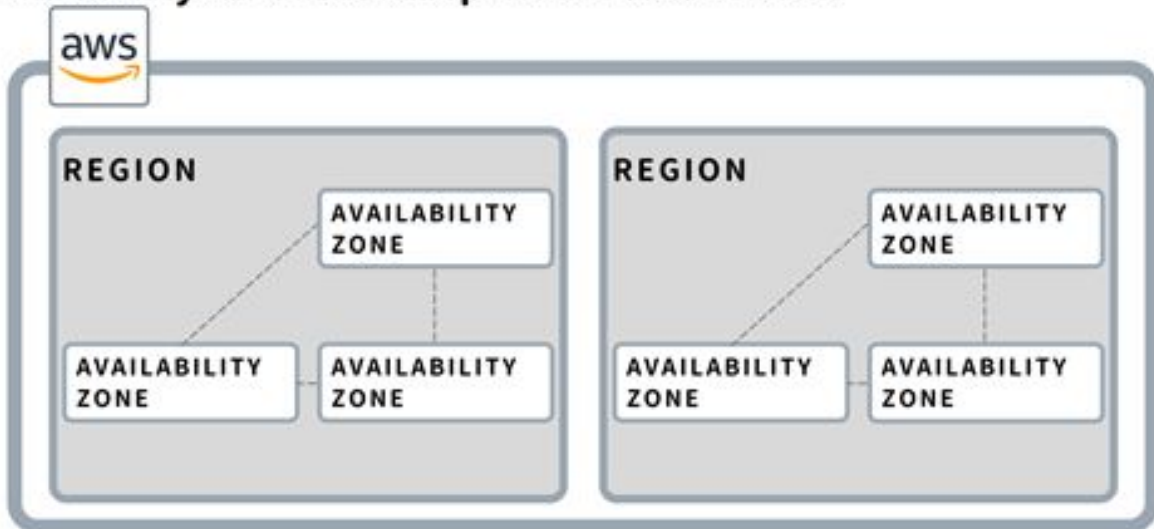
VPC (Virtual Private Cloud)



VPC stands for Virtual Private Cloud and is a service that allows you to create a virtual network on AWS. When you create a VPC, you select a region to create it.

Availability Zone

Availability zones are independent data centers



VPCs will also be created across availability zones. Availability zones are independent data centers, and there are multiple of them in a region.

Creating a VPC

TOKYO REGION

VPC

192.168.0.0/16

Because VPC creates a virtual private network space, the Use of private IP addresses is recommended

private IP address

- 10.0.0.0~10.255.255.255
- 172.16.0.0~172.31.255.255
- 192.168.0.0~192.168.255.255 ← Use this range this time

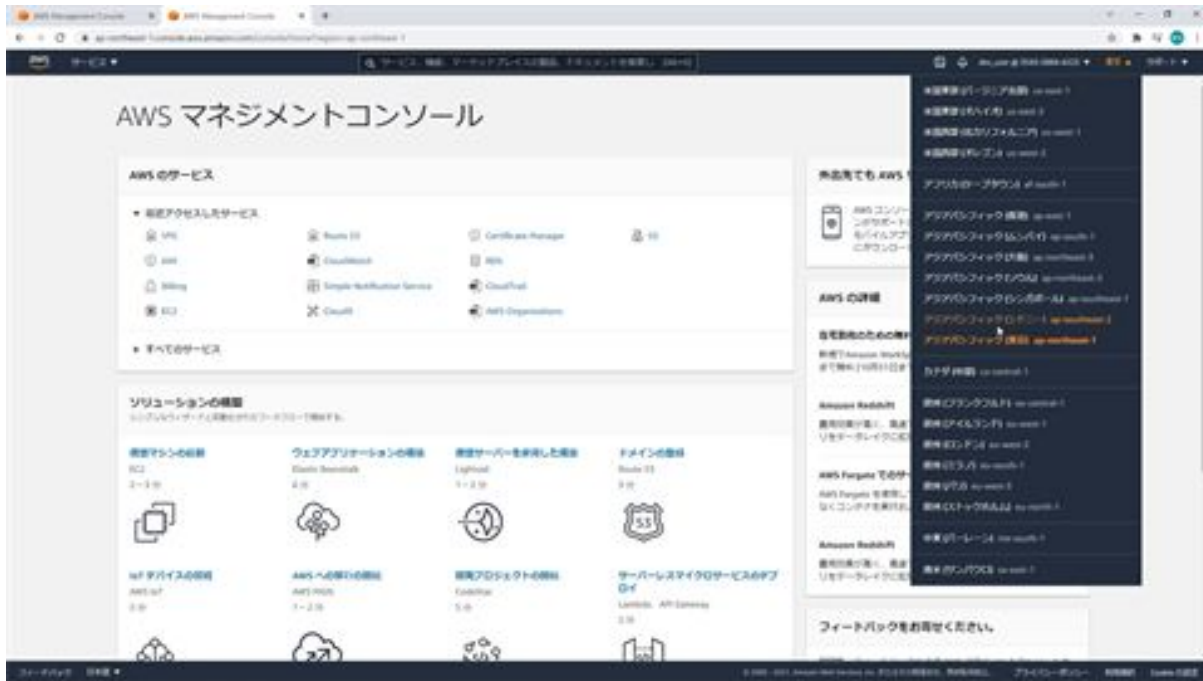
Range of IP addresses that can be used

- Size can be set from /28 to /16 (e.g., /28 is 2^4 , /16 allows 2^{16})
- Set a larger value because it cannot be changed after creation.
- /16 is recommended ← Use /16 this time

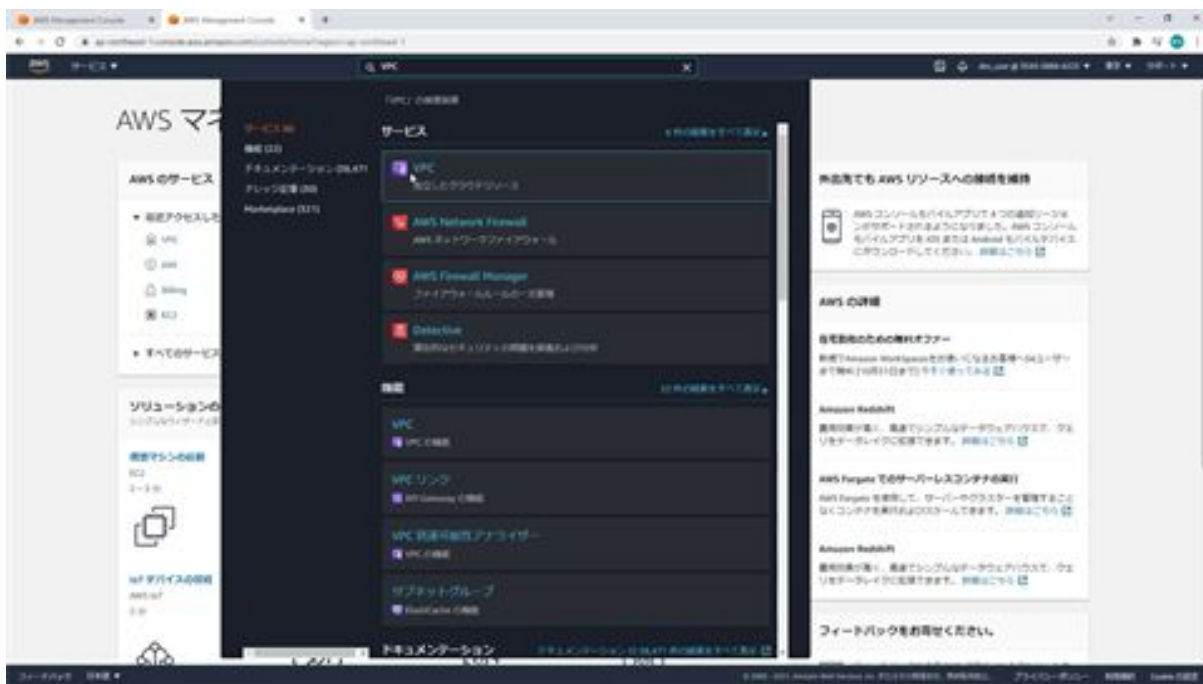
To create a VPC, you will need to enter the IPV4CIDR block. This is a CIDR notation that determines the IP address and range of IP addresses be used.

The recommended IP address to be used in a VPC is a private IP address. Then, enter a slash and a number after the IP address to determine the range of IP addresses to be used. The number entered after the slash is subtracted from 32 and then multiplied by two to determine the number of IP addresses that can be used. For /28, the number is 16 to the 4th power of 2. For /16, the number is 65536 to the 16th power of 2. This time, we will set up a VPC with 192.168.0.0./16. Now let's create the VPC.

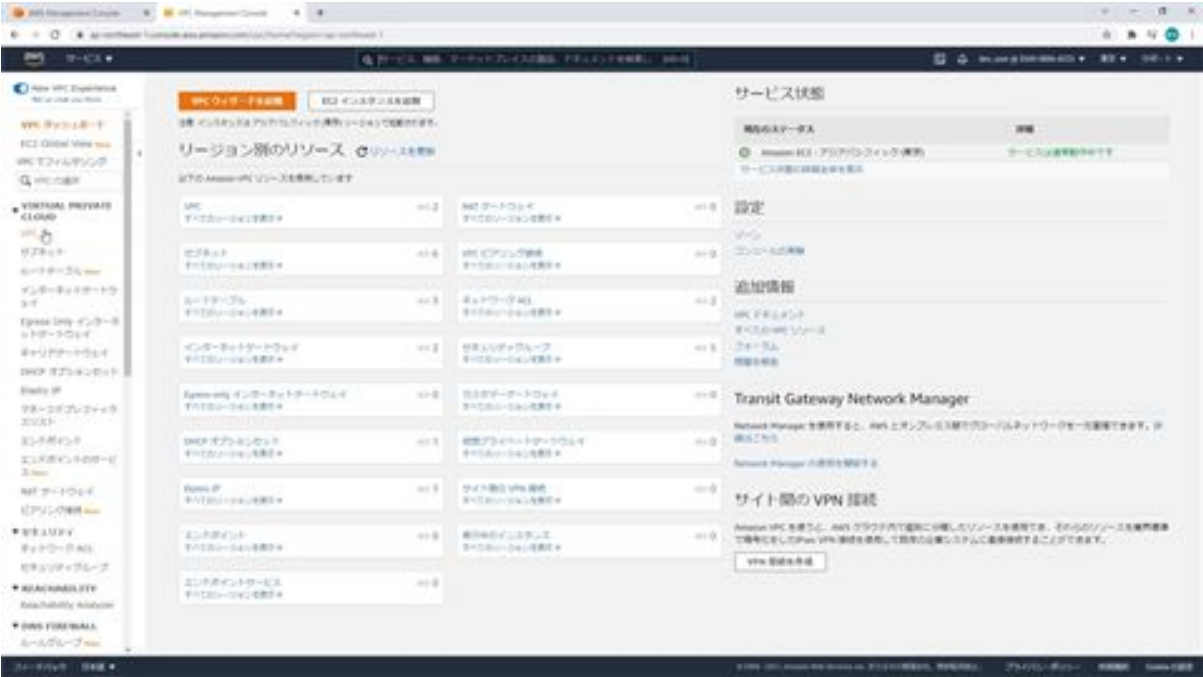
In the upper right corner, there is a menu to select a region, here select Tokyo.



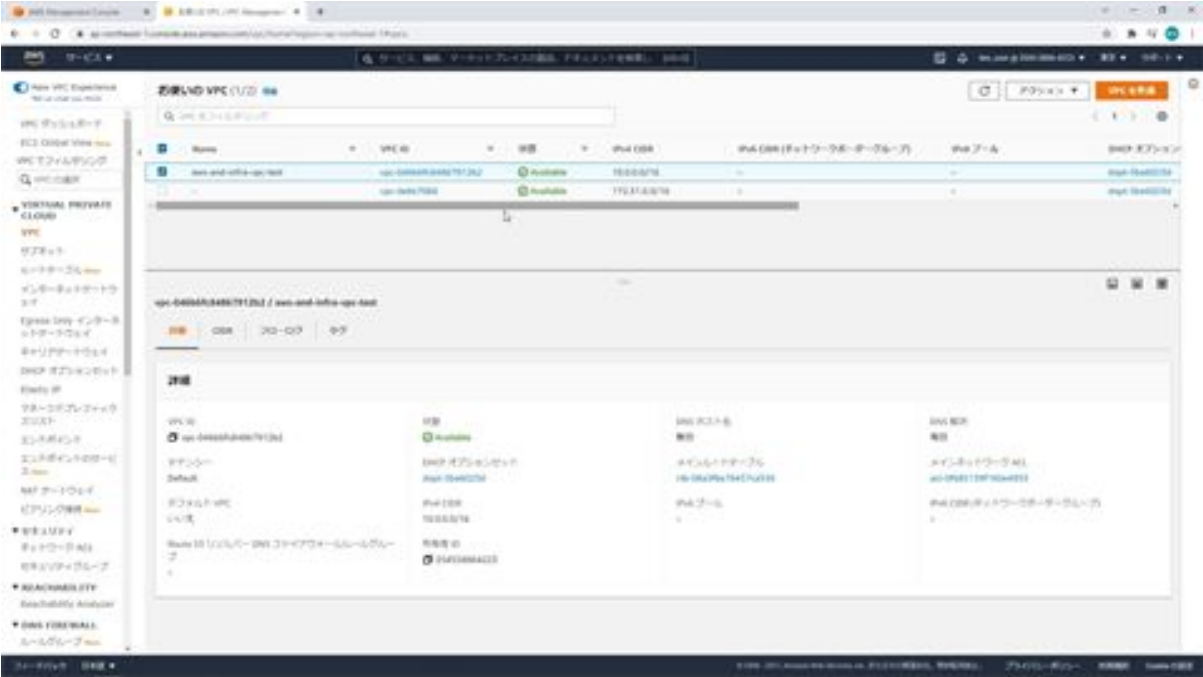
Then, the next step is to use a service called VPC to create a virtual network within AWS. Search for VPC in the service search and VPC will appear at the top of the list.



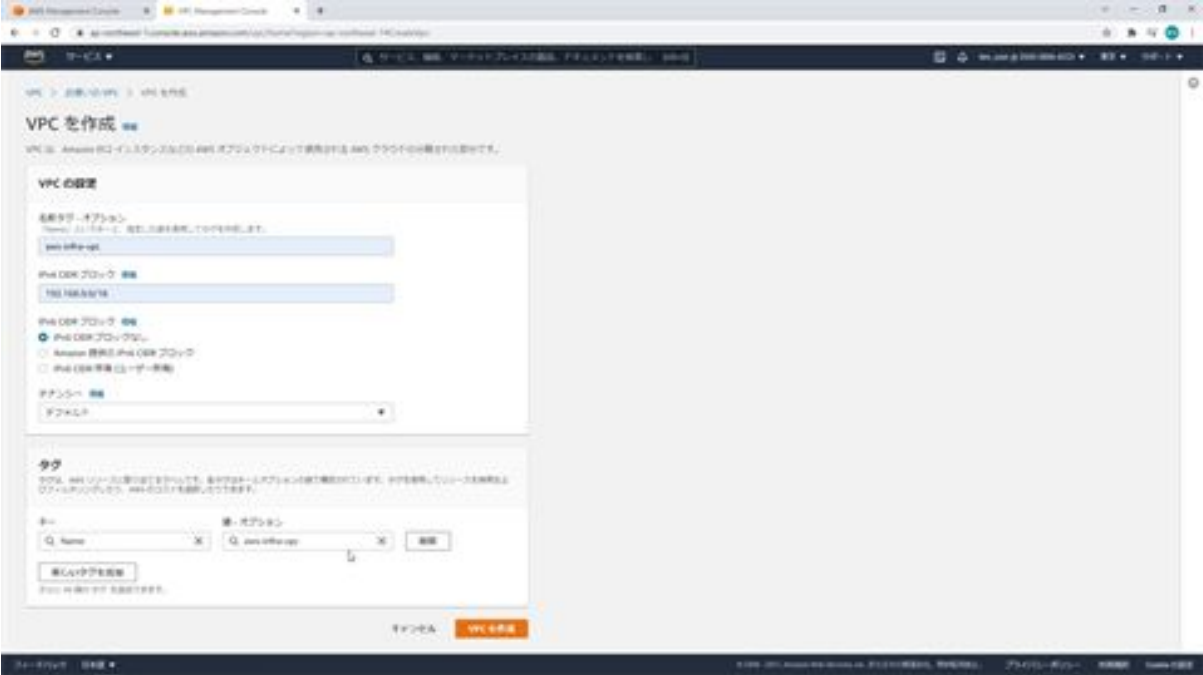
Next, click on VPC from the menu on the left and then click on the Create VPC button in the upper right corner.



Once you have done so, you will create a VPC on this screen.



First, enter the name of the VPC. Then enter 192.168.0.0/16 for the IPV4 input, and we will select the default of no block for the IPV6 CIDR block. This is a setting that determines whether or not to occupy physical hardware. Then click the Create VPC button at the bottom.

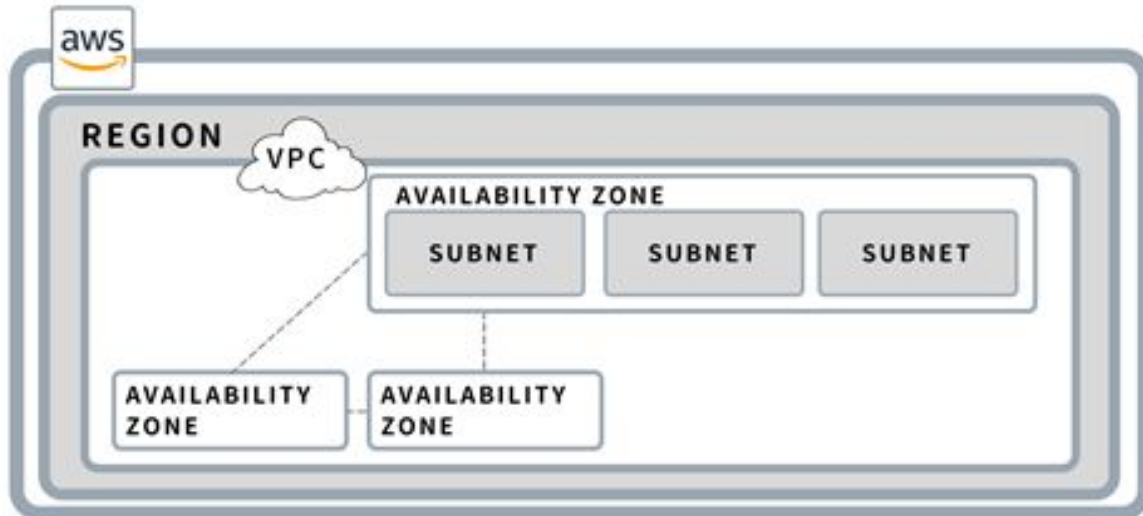


Click again on VPC in the menu on the left. Here we see that a new virtual network has been created. This completes the creation of the VPC.

Let's create a subnet

Subnet

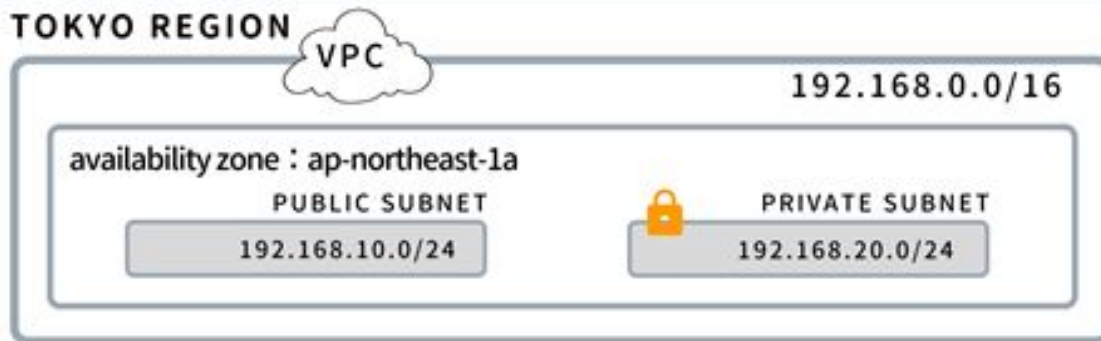
A subnet is a network of finely divided VPCs



Next, we will configure subnetting. Subnets are used to divide a VPC's IP address range into smaller IP address ranges for ease of use. The first is that it allows you to specify multiple availability zones, which are independent data centers physically separated from each other, so that in the event of a disaster if one availability zone becomes unavailable, you can still use the data centers in the other zone. The advantage is that the system can operate in other availability zones.

Subnet Creation

TOKYO REGION



Available network addresses

- Carve out an IP address range from a range of VPC CIDR blocks
- The first four and last IP addresses in each subnet CIDR block are reserved and cannot be used. For this public subnet, 192.168.10.0 to 192.168.10.3 and 192.168.10.255 cannot be used.

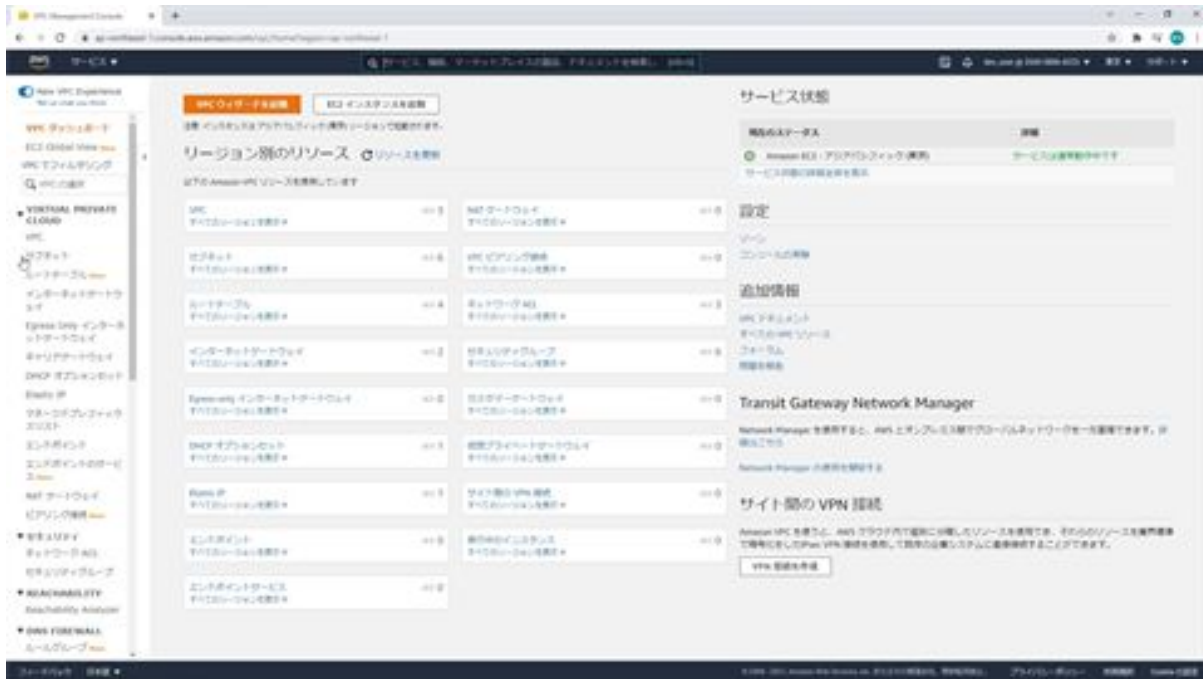
Estimate the number of IP addresses needed

- /24 is typical

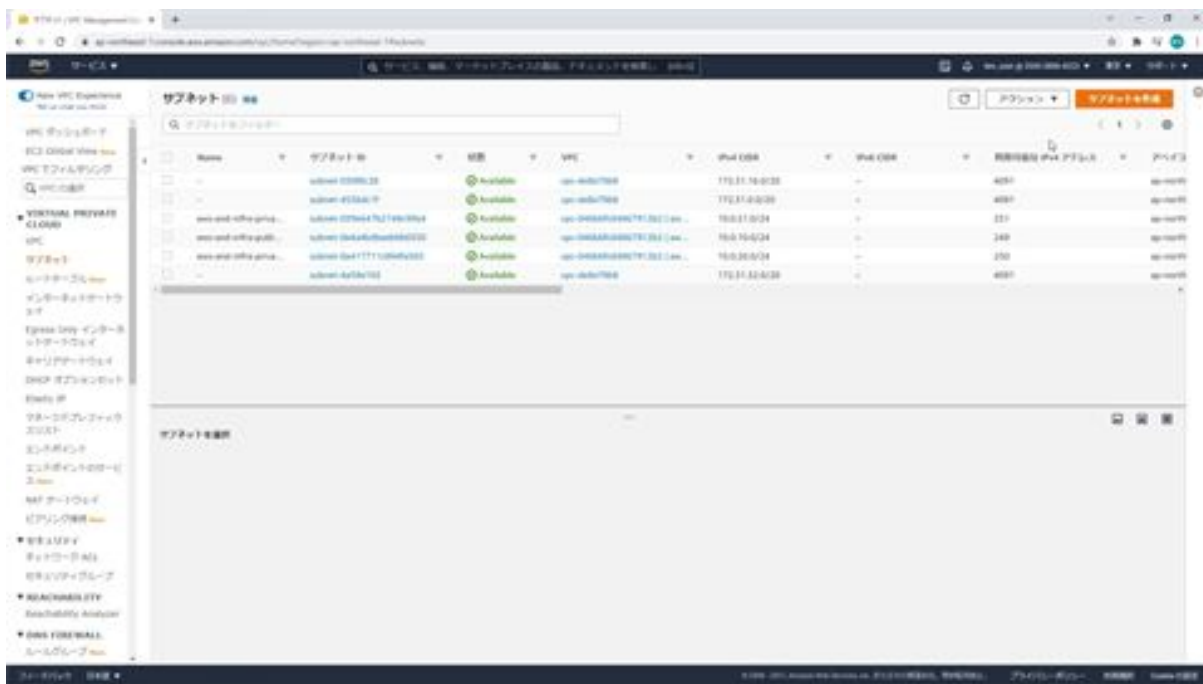
The other is a security issue. If you want the webserver to be connected to the Internet, but you want the database server to be hidden from the Internet, you need to divide it with a subnet. In this case, we want to create two subnets, one public subnet where the webserver will be located and one private subnet where the database server will be located.

The web server is placed on a public subnet to be connected to the Internet, and the database server is placed on a private subnet to be disconnected from the Internet. Now we will create the subnets.

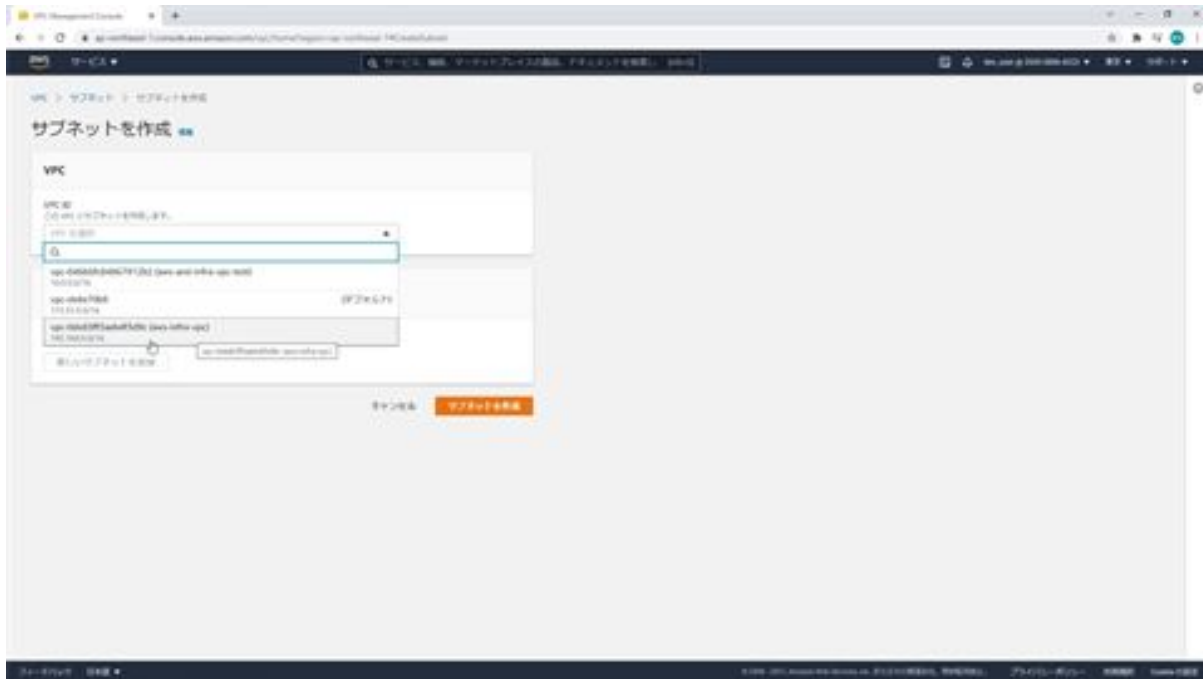
First, from the AWS Management Console, search for VPC and click on the VPC service at the top. When the screen appears, click Subnet in the menu on the left.



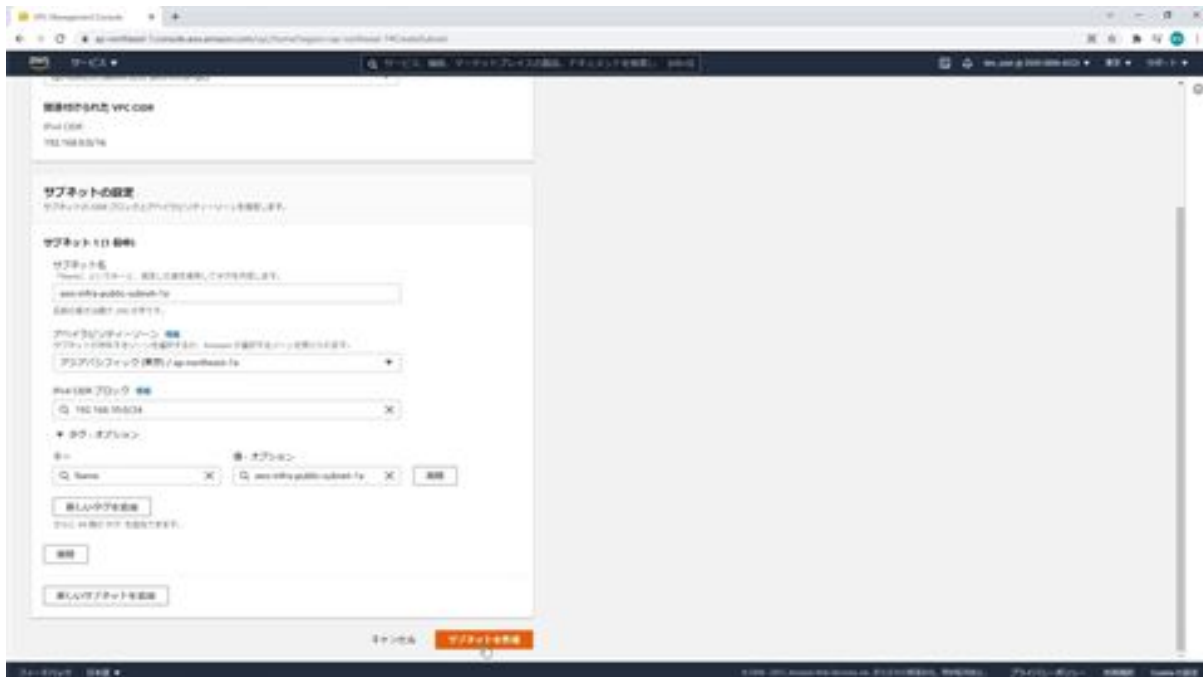
Next click on the Create Subnet button on the right.



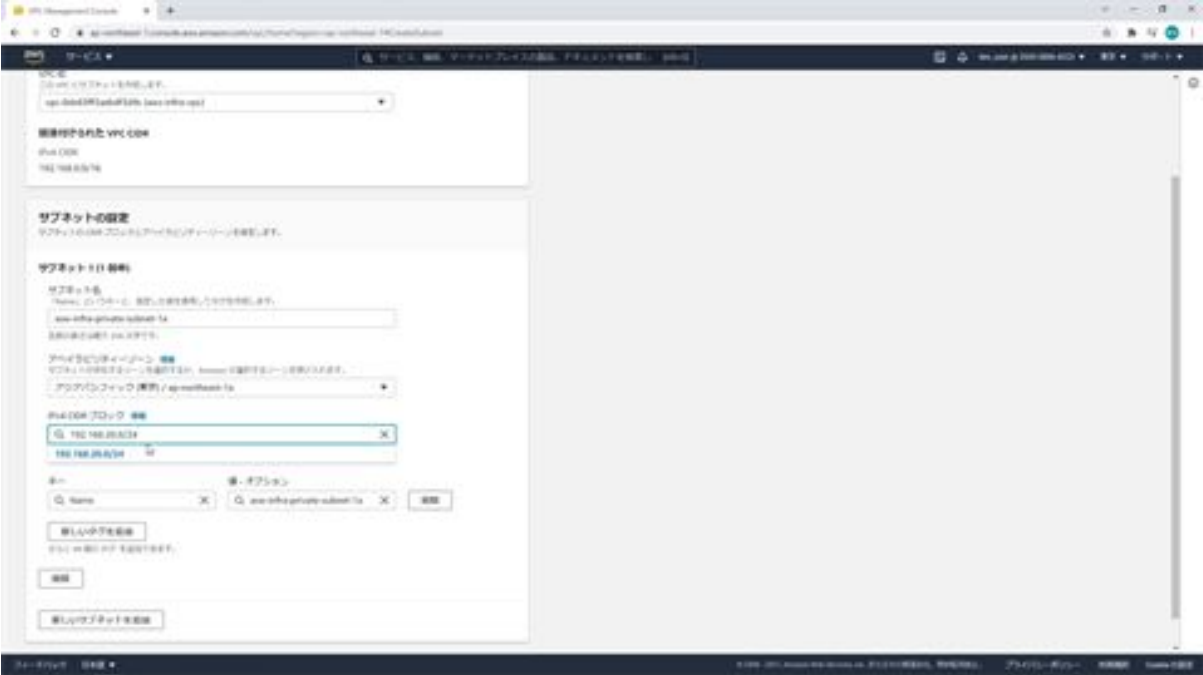
This screen allows you to create a subnet. First, select a VPC. Select the previously created VPC, aws-infra-vpc.



Next, enter the subnet name, availability zone, and then the CIDR block. Enter aws-infra-public-subnet-1a for the subnet name. Next, for the availability zone, enter 1a. for the CIDR block, enter 192.168.10.0/24. Now click the Create Subnet button. The public subnet is now created.

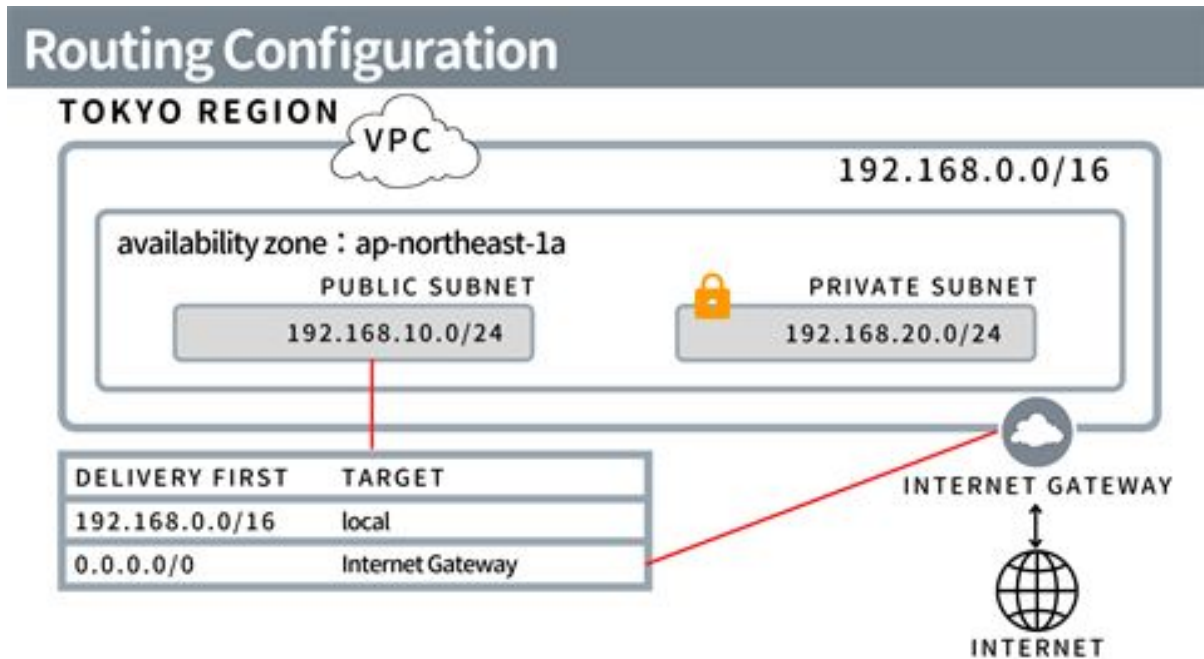


Next, we will create a private subnet. Click the Create Subnet button in the same way. Again, select the same VPC and enter aws-infra-private-subnet-1a here as well. Select 1a for the availability zone as well and enter 192.168.20.0/24 for the CIDR block.

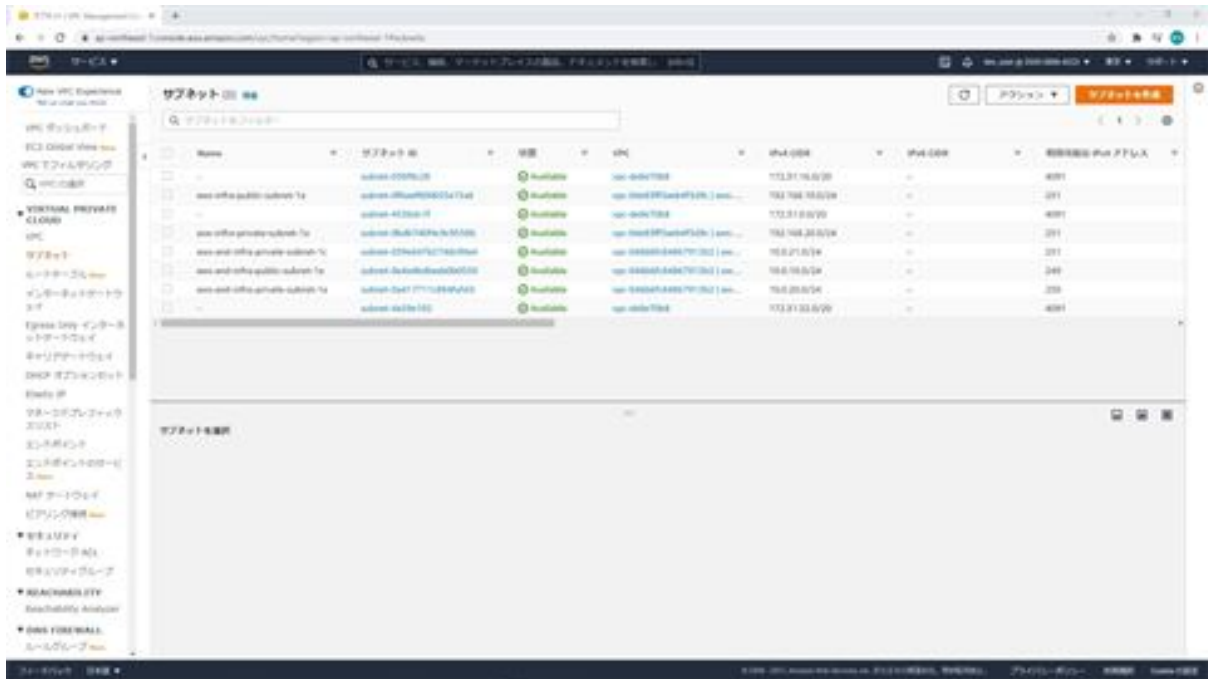


Now click the Create Subnet button. Two subnets have now been created, a public subnet and a private subnet.

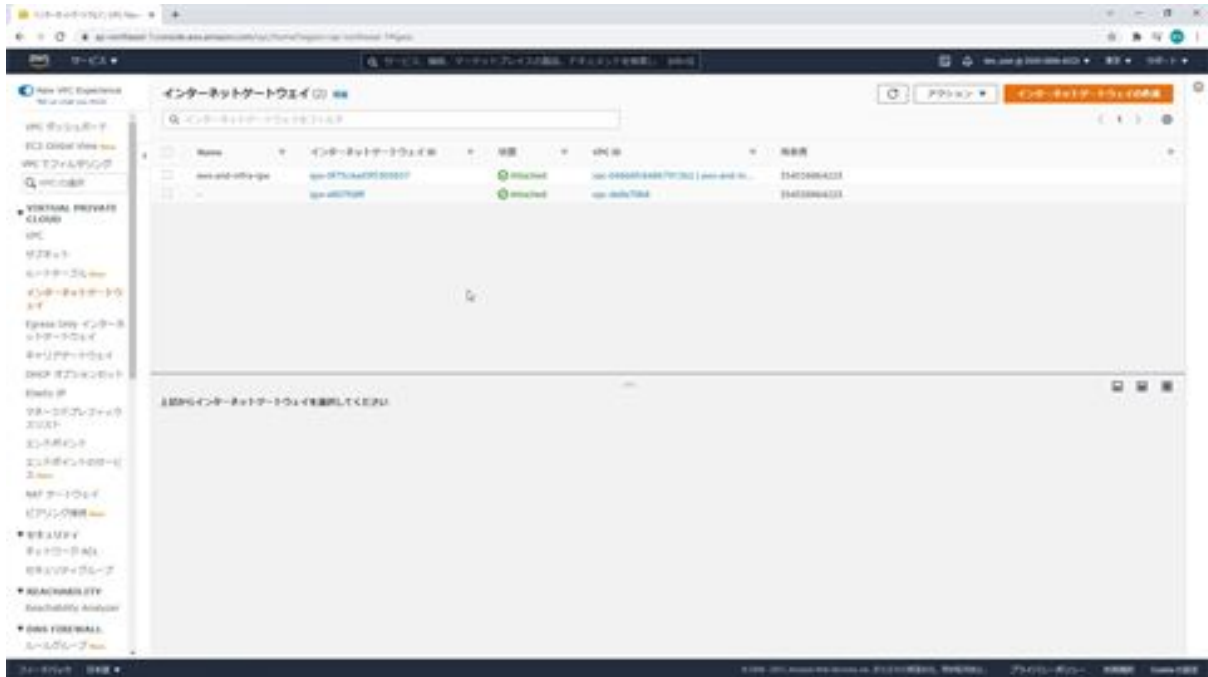
Let's set up routing



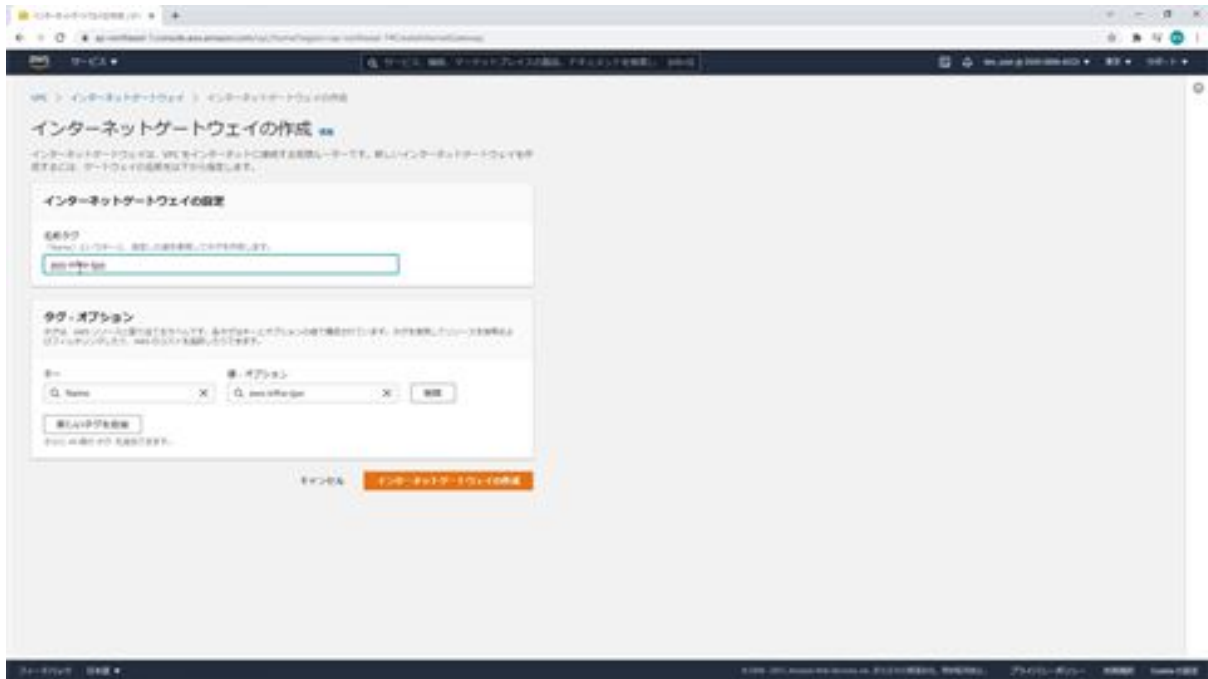
Now we will move on to the routing configuration. This configuration will allow us to connect to the Internet from a public subnet. We will place our web server on the public subnet, so it must be connected to the Internet. We will now proceed with the configuration. First, click on Internet Gateway in the left menu.



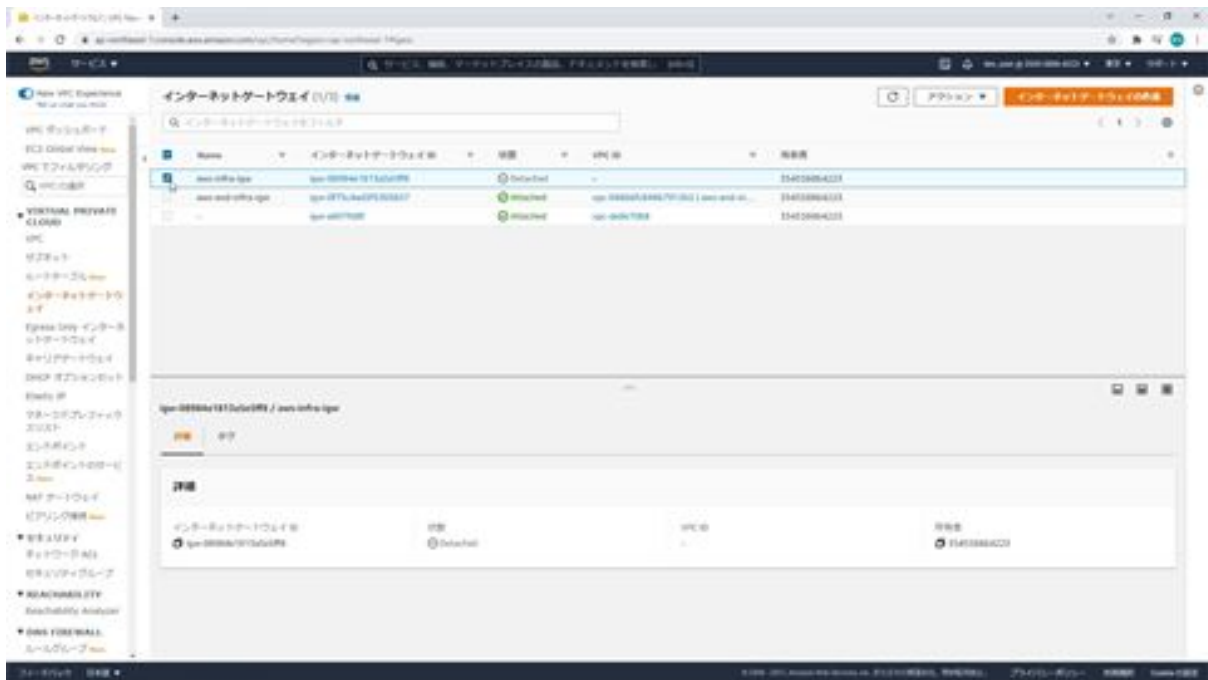
Next, click on the Create Internet Gateway button and go to this screen to create an Internet gateway.



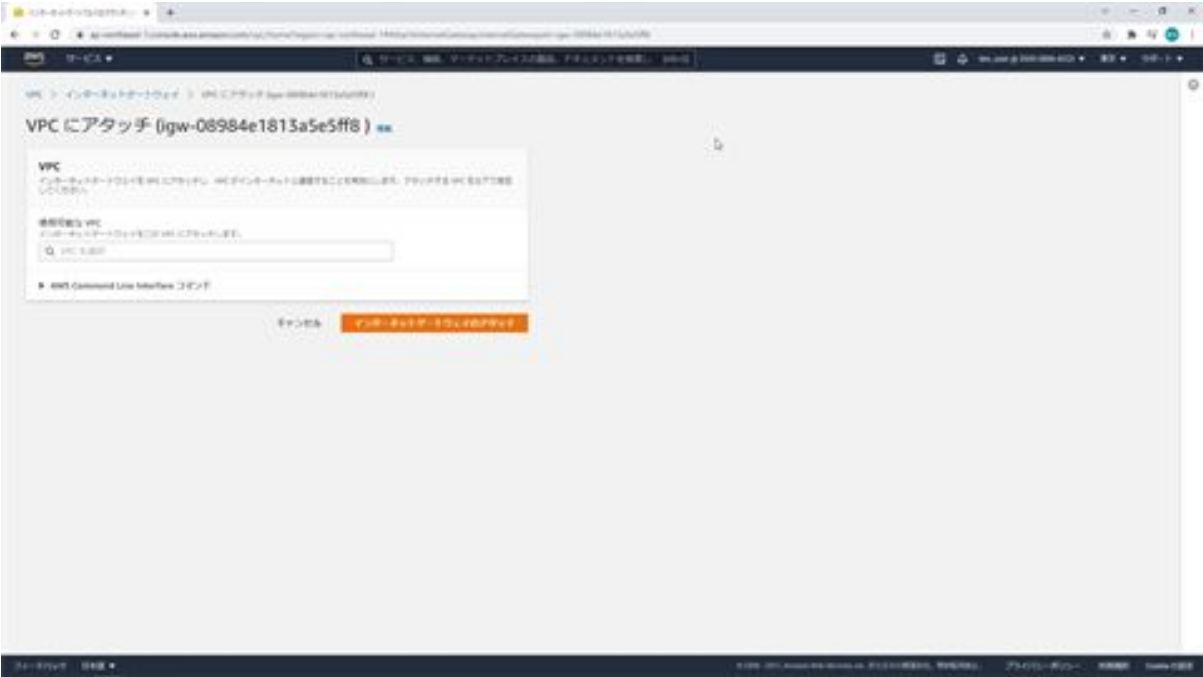
First, enter aws-infra-igw as the name and click the Create Internet Gateway button. The Internet Gateway has now been created.



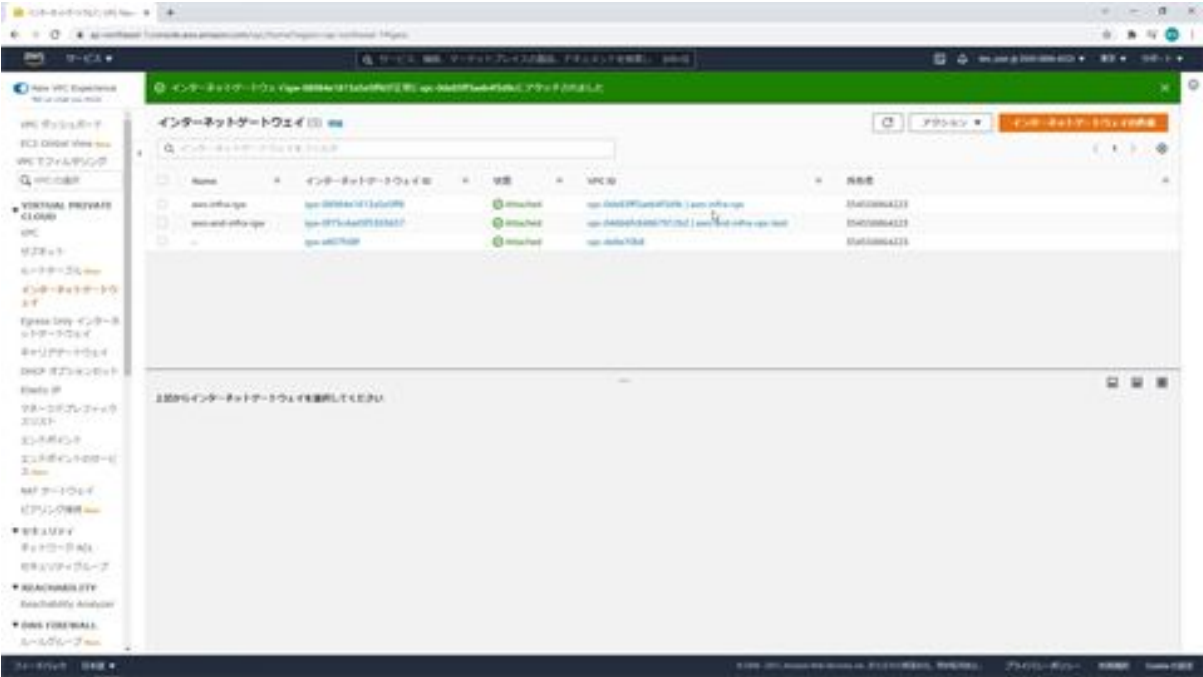
The next step is to tie the created Internet gateway to the VPC. Once back at the Internet Gateway screen, check aws-infra-igw and select Attach to VPC from Actions.



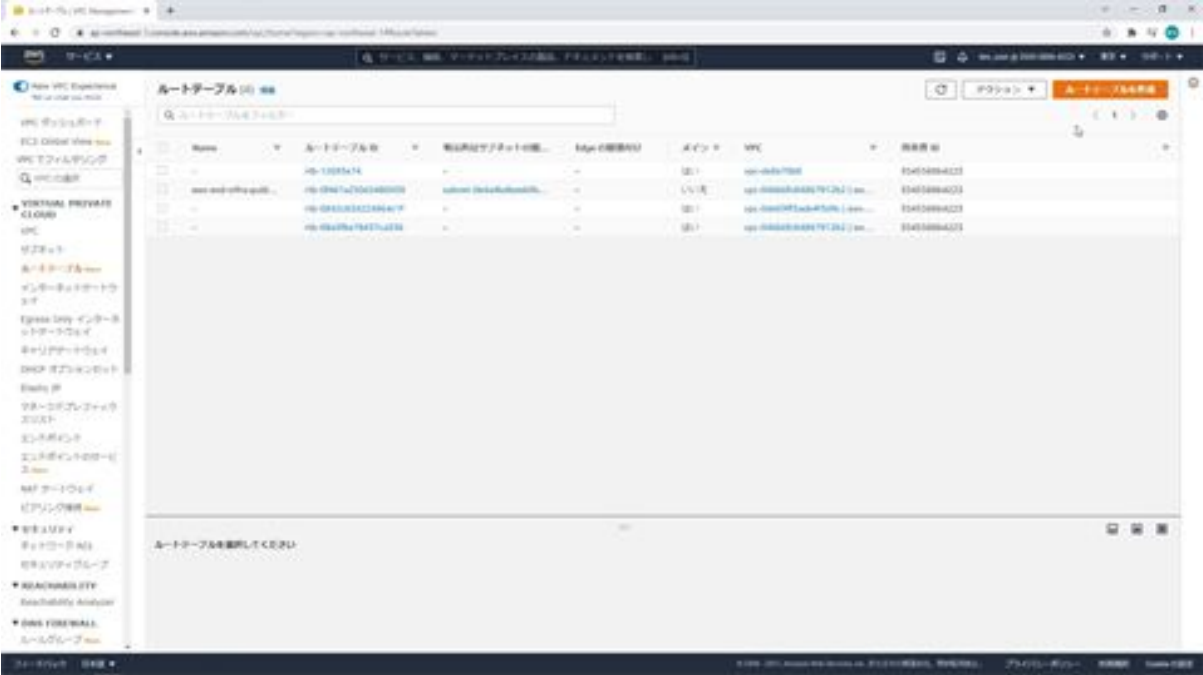
Next, the VPC selection screen will appear, so select the VPC you created here and click the attach button. This completes the attaching process.



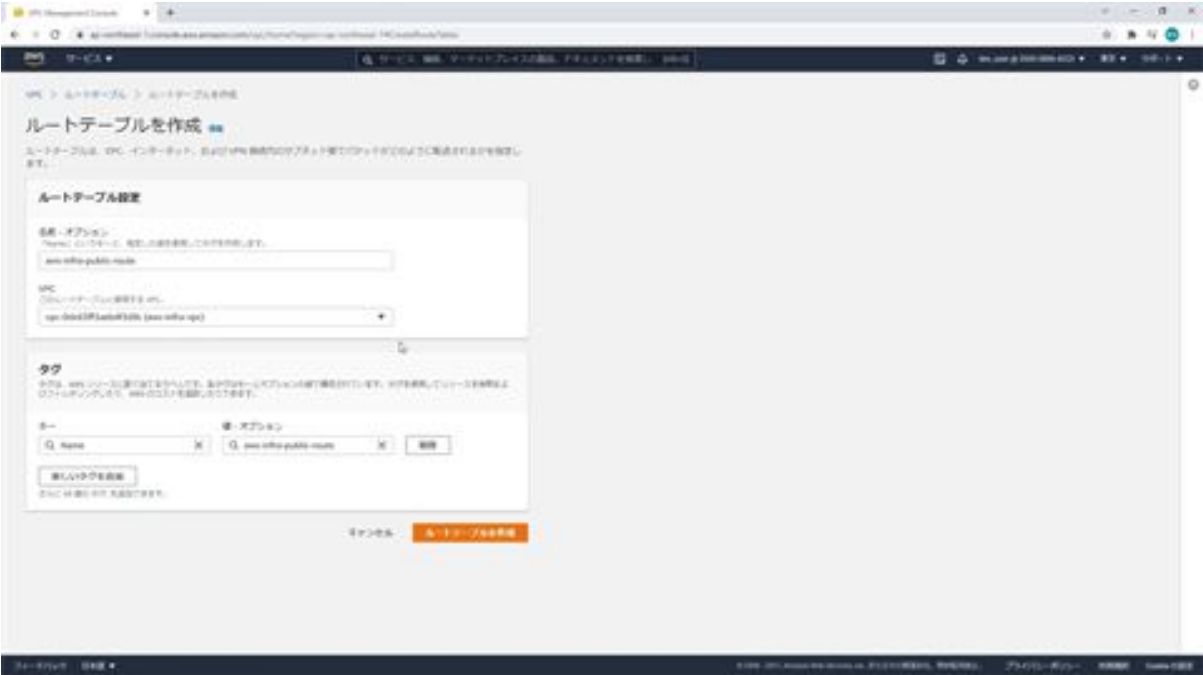
If you look at the VPC ID section, the ID of the attached VPC is shown as aws-infra-vpc.



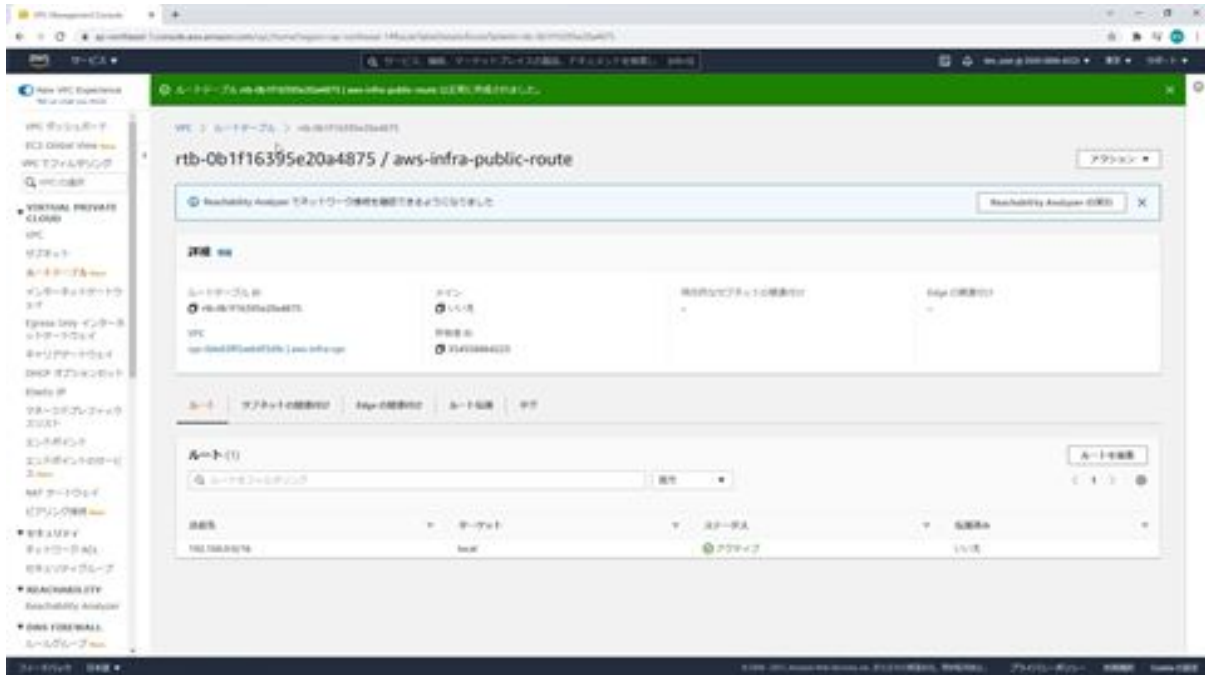
The next step is to create a routing table and tie it to the public subnet. Click on Route Table from the menu on the left, then click on the Create Route Table



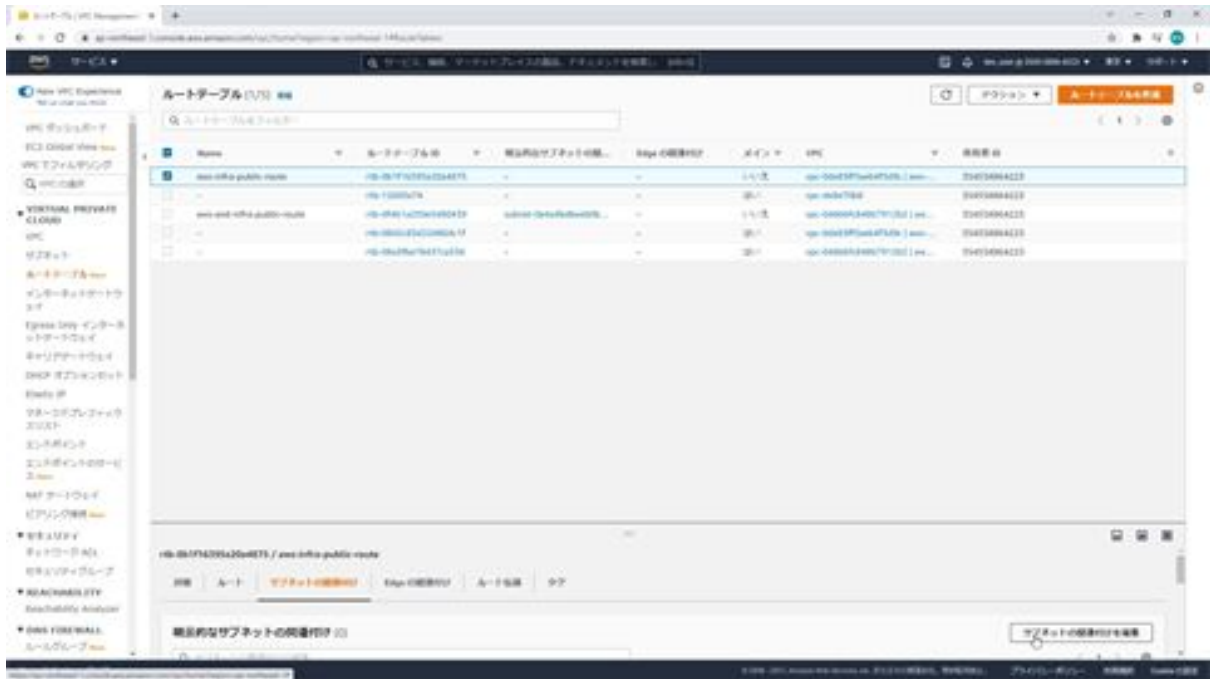
The root table will be created on this screen.



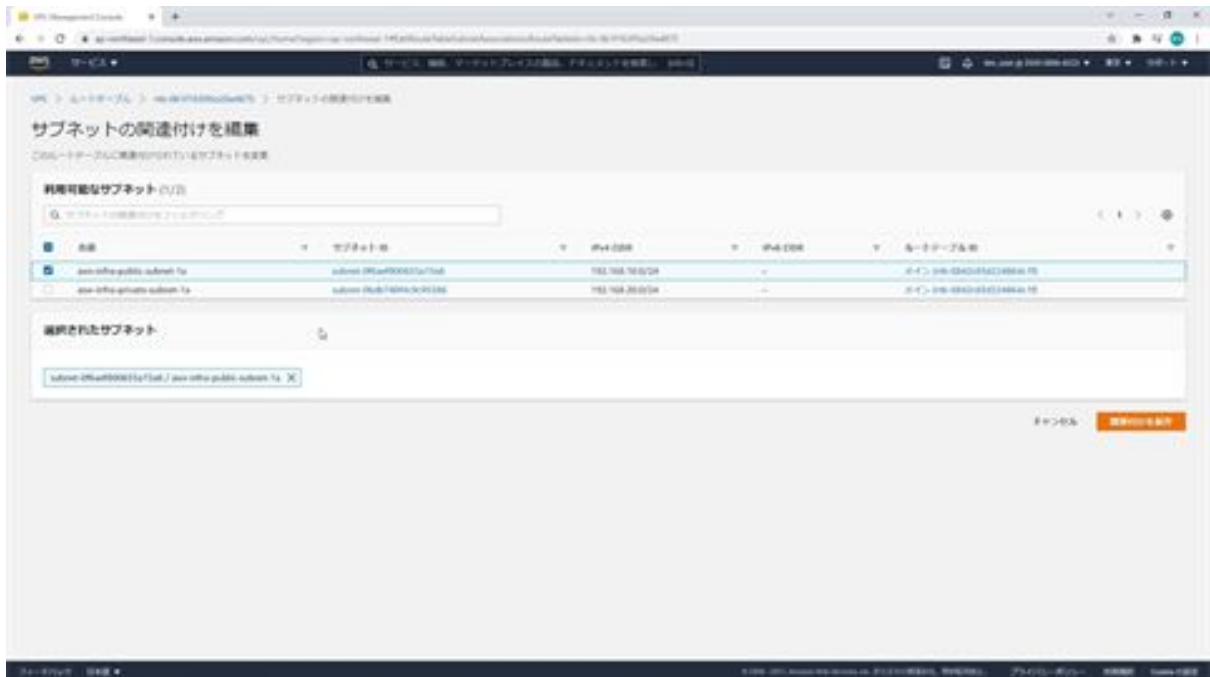
For the name, enter aws-infra-public-route, and for the VPC, select aws-infra-vpc. Then click on the Create Route Table button. You have now created the routing table. Now that this route table has been assigned to the VPC, we will associate it with the public subnet as well.



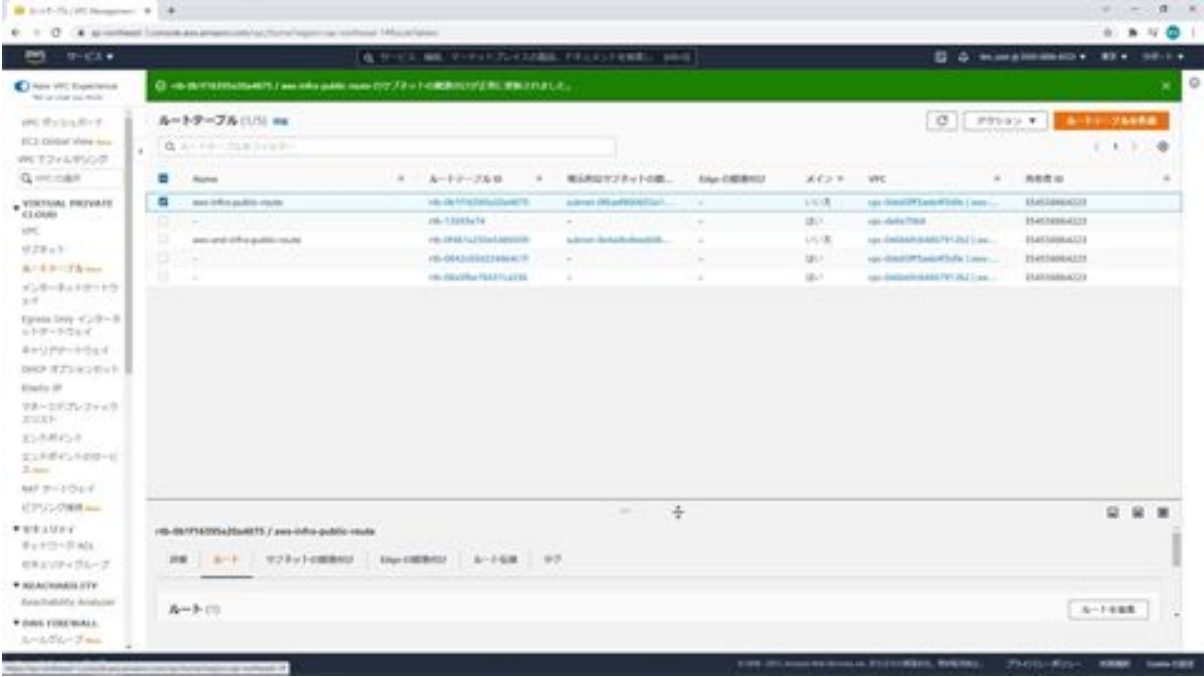
Return to the route table screen, select aws-infra-public-route and click on the subnet association tab at the bottom. Then click the Edit Subnet Associations button.



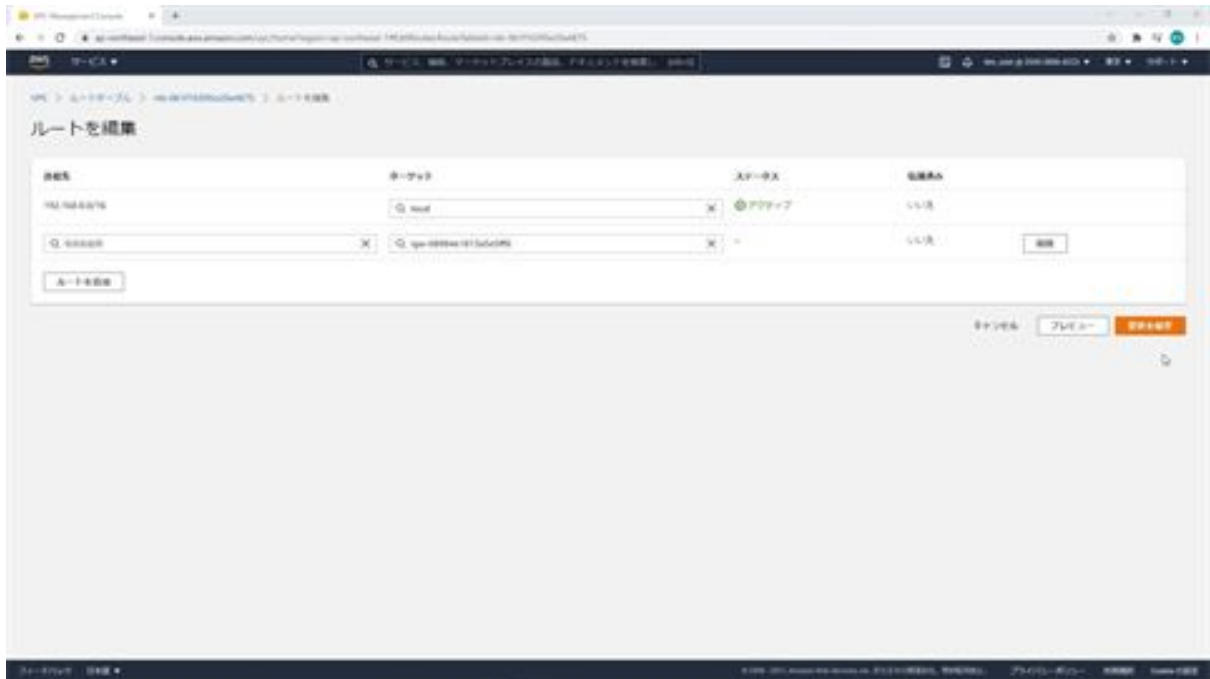
The available subnets will be displayed, check the public subnet of aws-infra-public-subnet-1a, and click the Save Associations button. The public subnet will now appear in the subnet associations and you can now assign this route table to the public subnet.



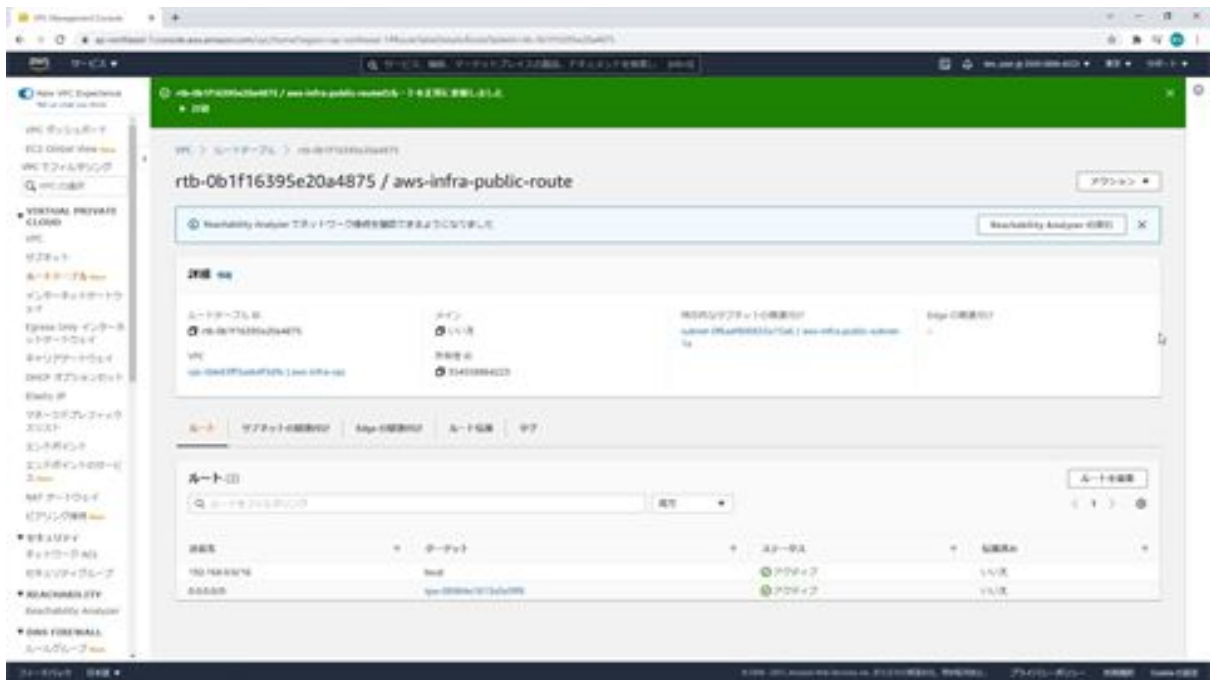
We will then set the default route to Internet Gateway: select aws-infra-public-route, click on the route tab below, and click on the Edit Route button.



You can edit the route on this screen, so click the Add Route button. Then enter 0.0.0.0/0 for Destination and select Internet Gateway as Target. You will then see aws-infra-igw and select it. Now click the Save Changes button.



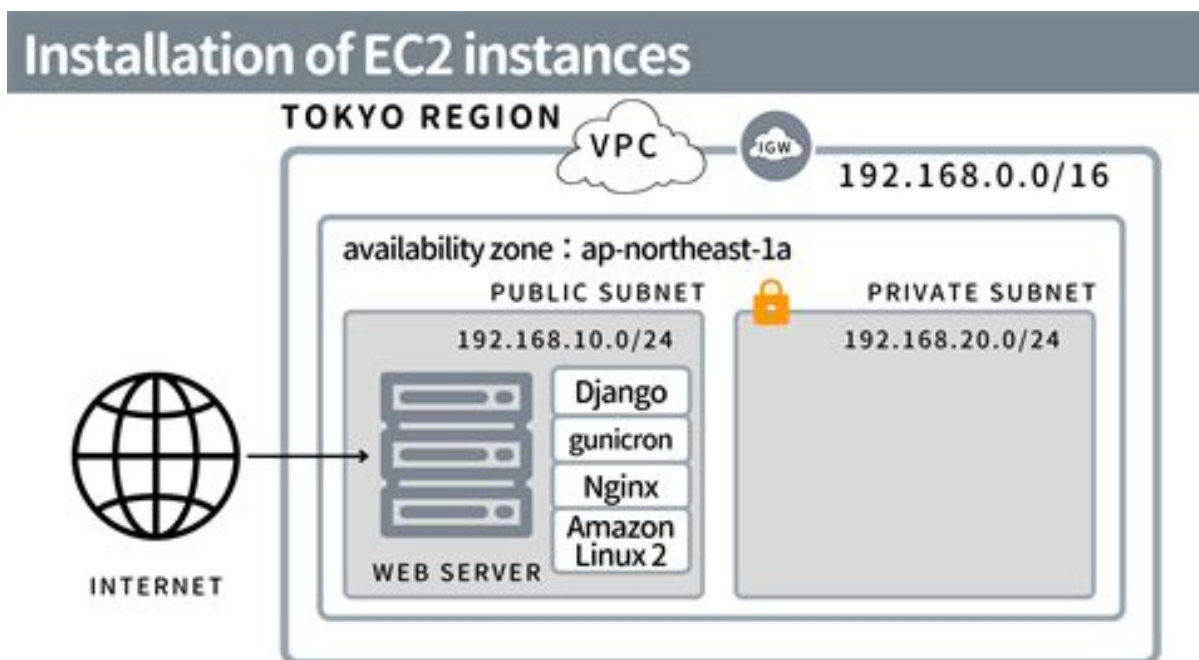
The default route is now set to Internet Gateway. You can see that the destination is 0.0.0.0/0 and the target is aws-infra-igw in the list on the route tab.



We can now create a routing table and associate the public subnet with a routing table whose default route is the Internet gateway. This will allow you to connect to the Internet from the public subnet.

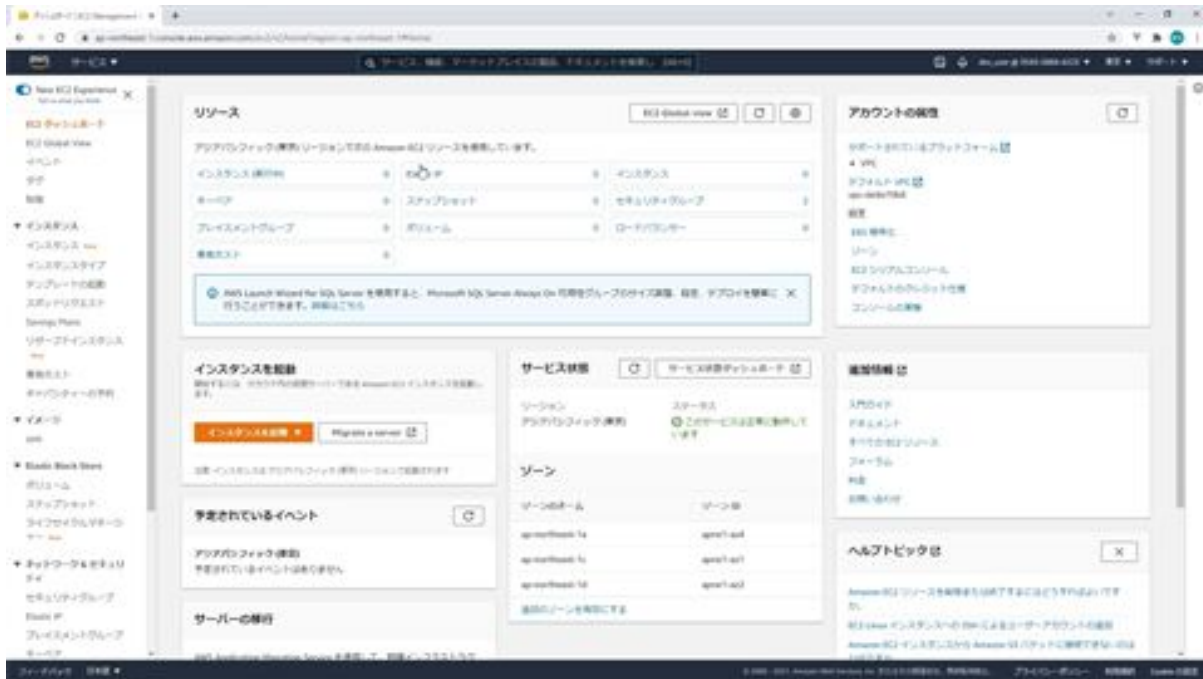
Chapter 4: Let's Build a Web Server

Let's set up an EC2 instance

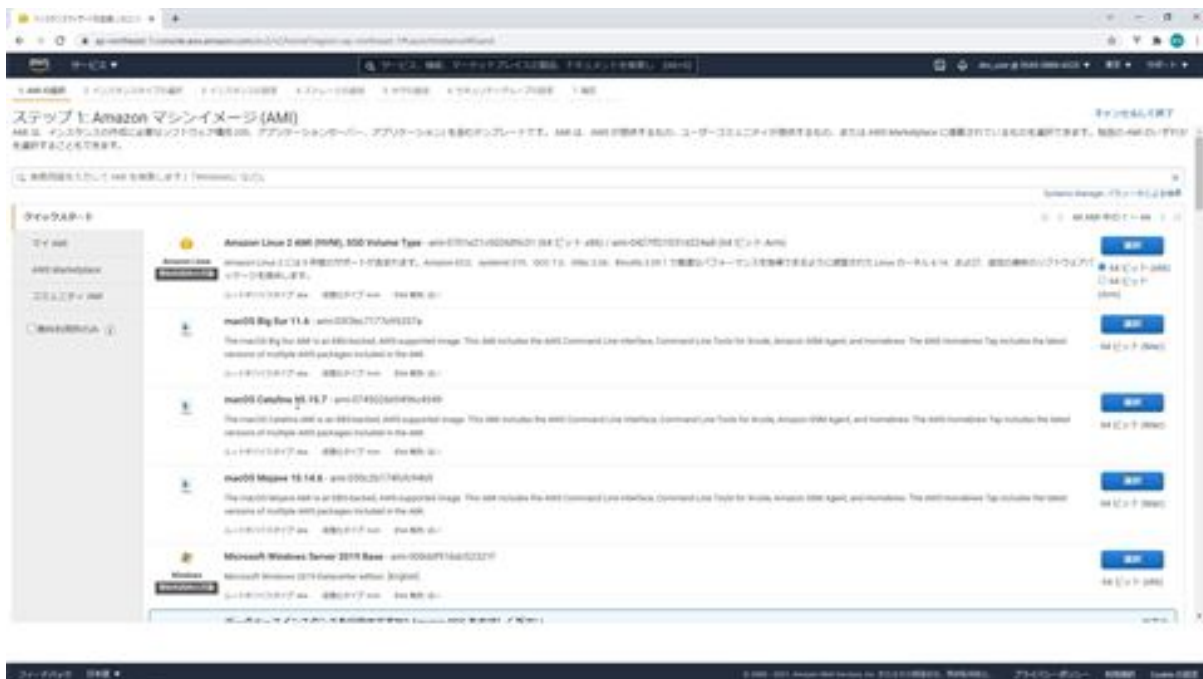


From this chapter, we will build a web server by installing an EC2 instance, which is a virtual server in the AWS Cloud. The servers will be up and running in a few minutes, and you can add or remove servers and change machine specs in a matter of minutes. This makes it easy to handle sudden increases in access. Now, let's set up an EC2 instance.

From the AWS management console, search for EC2 and navigate to the EC2 dashboard.



Next, click the Launch Instance button. Now we will create an instance. First, choose an AMI, Amazon Linux2 Ami.



Next, select the instance type, this time t2.micro, which is available for free use. t2.micro is selected, click the Configure Instance Details button.

ステップバイステップインスタンスタイプの選択

Amazon EC2 では、異なるインスタンスタイプに固有のさまざまなインスタンスタイプが用意されています。インスタンスは、アプリケーションに最も適したタイプで、インスタンスタイプはさまざまな用途、用途、ストレージ、ネットワークやその他の機能の組み合わせによって構成されています。使用するアプリケーションの要件に応じて適切なタイプの組み合わせを選択する必要があります。インスタンスタイプは、その用途に最適なタイプの組み合わせに適用する必要があります。

インスタンスタイプ	アーキテクチャ	vCPU	メモリ	ネットワーク	EBS	置かれたグループ
t3	arm64	1	1	標準的な	-	置かれた
t3	arm64	2	2	標準的な	-	置かれた
t3	arm64	4	4	標準的な	-	置かれた
t3	arm64	8	8	標準的な	-	置かれた
t3	arm64	16	16	標準的な	-	置かれた
t3	arm64	32	32	標準的な	-	置かれた
t3	arm64	64	64	標準的な	-	置かれた
t3	arm64	128	128	標準的な	-	置かれた
t3	arm64	256	256	標準的な	-	置かれた
t3	arm64	512	512	標準的な	-	置かれた
t3	arm64	1024	1024	標準的な	-	置かれた
t3	arm64	2048	2048	標準的な	-	置かれた
t3	arm64	4096	4096	標準的な	-	置かれた
t3	arm64	8192	8192	標準的な	-	置かれた
t3	arm64	16384	16384	標準的な	-	置かれた
t3	arm64	32768	32768	標準的な	-	置かれた
t3	arm64	65536	65536	標準的な	-	置かれた
t3	arm64	131072	131072	標準的な	-	置かれた
t3	arm64	262144	262144	標準的な	-	置かれた
t3	arm64	524288	524288	標準的な	-	置かれた
t3	arm64	1048576	1048576	標準的な	-	置かれた
t3	arm64	2097152	2097152	標準的な	-	置かれた
t3	arm64	4194304	4194304	標準的な	-	置かれた
t3	arm64	8388608	8388608	標準的な	-	置かれた
t3	arm64	16777216	16777216	標準的な	-	置かれた
t3	arm64	33554432	33554432	標準的な	-	置かれた
t3	arm64	67108864	67108864	標準的な	-	置かれた
t3	arm64	134217728	134217728	標準的な	-	置かれた
t3	arm64	268435456	268435456	標準的な	-	置かれた
t3	arm64	536870912	536870912	標準的な	-	置かれた
t3	arm64	1073741824	1073741824	標準的な	-	置かれた
t3	arm64	2147483648	2147483648	標準的な	-	置かれた
t3	arm64	4294967296	4294967296	標準的な	-	置かれた
t3	arm64	8589934592	8589934592	標準的な	-	置かれた
t3	arm64	17179869184	17179869184	標準的な	-	置かれた
t3	arm64	34359738368	34359738368	標準的な	-	置かれた
t3	arm64	68719476736	68719476736	標準的な	-	置かれた
t3	arm64	137438953472	137438953472	標準的な	-	置かれた
t3	arm64	274877906944	274877906944	標準的な	-	置かれた
t3	arm64	549755813888	549755813888	標準的な	-	置かれた
t3	arm64	1099511627776	1099511627776	標準的な	-	置かれた
t3	arm64	2199023255552	2199023255552	標準的な	-	置かれた
t3	arm64	4398046511104	4398046511104	標準的な	-	置かれた
t3	arm64	8796093022208	8796093022208	標準的な	-	置かれた
t3	arm64	17592186044416	17592186044416	標準的な	-	置かれた
t3	arm64	35184372088832	35184372088832	標準的な	-	置かれた
t3	arm64	70368744177664	70368744177664	標準的な	-	置かれた
t3	arm64	140737488355328	140737488355328	標準的な	-	置かれた
t3	arm64	281474976710656	281474976710656	標準的な	-	置かれた
t3	arm64	562949953421312	562949953421312	標準的な	-	置かれた
t3	arm64	1125899906842624	1125899906842624	標準的な	-	置かれた
t3	arm64	2251799813685248	2251799813685248	標準的な	-	置かれた
t3	arm64	4503599627370496	4503599627370496	標準的な	-	置かれた
t3	arm64	9007199254740992	9007199254740992	標準的な	-	置かれた
t3	arm64	18014398509481984	18014398509481984	標準的な	-	置かれた
t3	arm64	36028797018963968	36028797018963968	標準的な	-	置かれた
t3	arm64	72057594037927936	72057594037927936	標準的な	-	置かれた
t3	arm64	144115188075855872	144115188075855872	標準的な	-	置かれた
t3	arm64	288230376151711744	288230376151711744	標準的な	-	置かれた
t3	arm64	576460752303423488	576460752303423488	標準的な	-	置かれた
t3	arm64	1152921504606846976	1152921504606846976	標準的な	-	置かれた
t3	arm64	2305843009213693952	2305843009213693952	標準的な	-	置かれた
t3	arm64	4611686018427387904	4611686018427387904	標準的な	-	置かれた
t3	arm64	9223372036854775808	9223372036854775808	標準的な	-	置かれた
t3	arm64	18446744073709551616	18446744073709551616	標準的な	-	置かれた
t3	arm64	36893488147419103232	36893488147419103232	標準的な	-	置かれた
t3	arm64	73786976294838206464	73786976294838206464	標準的な	-	置かれた
t3	arm64	147573952589676412928	147573952589676412928	標準的な	-	置かれた
t3	arm64	295147905179352825856	295147905179352825856	標準的な	-	置かれた
t3	arm64	590295810358705651712	590295810358705651712	標準的な	-	置かれた
t3	arm64	1180591620717411303424	1180591620717411303424	標準的な	-	置かれた
t3	arm64	2361183241434822606848	2361183241434822606848	標準的な	-	置かれた
t3	arm64	4722366482869645213696	4722366482869645213696	標準的な	-	置かれた
t3	arm64	9444732965739290427392	9444732965739290427392	標準的な	-	置かれた
t3	arm64	18889465931478580854784	18889465931478580854784	標準的な	-	置かれた
t3	arm64	37778931862957161709568	37778931862957161709568	標準的な	-	置かれた
t3	arm64	75557863725914323419136	75557863725914323419136	標準的な	-	置かれた
t3	arm64	151115727451828646838272	151115727451828646838272	標準的な	-	置かれた
t3	arm64	302231454903657293675544	302231454903657293675544	標準的な	-	置かれた
t3	arm64	604462909807314587351088	604462909807314587351088	標準的な	-	置かれた
t3	arm64	1208925819614629174702176	1208925819614629174702176	標準的な	-	置かれた
t3	arm64	2417851639229258349404352	2417851639229258349404352	標準的な	-	置かれた
t3	arm64	4835703278458516698808704	4835703278458516698808704	標準的な	-	置かれた
t3	arm64	9671406556917033397617408	9671406556917033397617408	標準的な	-	置かれた
t3	arm64	19342813113834066795234816	19342813113834066795234816	標準的な	-	置かれた
t3	arm64	38685626227668133590469632	38685626227668133590469632	標準的な	-	置かれた
t3	arm64	77371252455336267180939264	77371252455336267180939264	標準的な	-	置かれた
t3	arm64	154742504910672534361878528	154742504910672534361878528	標準的な	-	置かれた
t3	arm64	309485009821345068723757056	309485009821345068723757056	標準的な	-	置かれた
t3	arm64	618970019642690137447514112	618970019642690137447514112	標準的な	-	置かれた
t3	arm64	1237940039285380274895028224	1237940039285380274895028224	標準的な	-	置かれた
t3	arm64	2475880078570760549790056448	2475880078570760549790056448	標準的な	-	置かれた
t3	arm64	4951760157141521099580112896	4951760157141521099580112896	標準的な	-	置かれた
t3	arm64	9903520314283042199160225792	9903520314283042199160225792	標準的な	-	置かれた
t3	arm64	19807040628566084398320451584	19807040628566084398320451584	標準的な	-	置かれた
t3	arm64	39614081257132168796640903168	39614081257132168796640903168	標準的な	-	置かれた
t3	arm64	79228162514264337593281806336	79228162514264337593281806336	標準的な	-	置かれた
t3	arm64	158456325028528675186563612672	158456325028528675186563612672	標準的な	-	置かれた
t3	arm64	316912650057057350373127225344	316912650057057350373127225344	標準的な	-	置かれた
t3	arm64	633825300114114700746254450688	633825300114114700746254450688	標準的な	-	置かれた
t3	arm64	1267650600228229401492508901376	1267650600228229401492508901376	標準的な	-	置かれた
t3	arm64	2535301200456458802985017802752	2535301200456458802985017802752	標準的な	-	置かれた
t3	arm64	5070602400912917605970035605504	5070602400912917605970035605504	標準的な	-	置かれた
t3	arm64	10141204801825835211940071211008	10141204801825835211940071211008	標準的な	-	置かれた
t3	arm64	20282409603651670423880142422016	20282409603651670423880142422016	標準的な	-	置かれた
t3	arm64	40564819207303340847760284844032	40564819207303340847760284844032	標準的な	-	置かれた
t3	arm64	81129638414606681695520569688064	81129638414606681695520569688064	標準的な	-	置かれた
t3	arm64	162259276829213363391041139376128	162259276829213363391041139376128	標準的な	-	置かれた
t3	arm64	324518553658426726782082278752256	324518553658426726782082278752256	標準的な	-	置かれた
t3	arm64	649037107316853453564164557504512	649037107316853453564164557504512	標準的な	-	置かれた
t3	arm64	12980742146337069071283291150081024	12980742146337069071283291150081024	標準的な	-	置かれた
t3	arm64	25961484292674138142565982300162048	25961484292674138142565982300162048	標準的な	-	置かれた
t3	arm64	51922968585348276285131964600324096	51922968585348276285131964600324096	標準的な	-	置かれた
t3	arm64	103845937170696552570263929200648192	103845937170696552570263929200648192	標準的な	-	置かれた
t3	arm64	207691874341393105140527858401296384	207691874341393105140527858401296384	標準的な	-	置かれた
t3	arm64	415383748682786210281055716802592768	415383748682786210281055716802592768	標準的な	-	置かれた
t3	arm64	8307674973655724205621114336051855536	8307674973655724205621114336051855536	標準的な	-	置かれた
t3	arm64	16615349947311448411242228672103111104	16615349947311448411242228672103111104	標準的な	-	置かれた
t3	arm64	33230699894622896822484457344202222208	33230699894622896822484457344202222208	標準的な	-	置かれた
t3	arm64	66461399789245793644968914688404444416	66461399789245793644968914688404444416	標準的な	-	置かれた
t3	arm64	13292279957849158728993782937688888832	13292279957849158728993782937688888832	標準的な	-	置かれた
t3	arm64	2658455991569831745798756587537777664	2658455991569831745798756587537777664	標準的な	-	置かれた
t3	arm64	5316911983139663491597513175075555328	5316911983139663491597513175075555328	標準的な	-	置かれた
t3	arm64	1063382396627932698319502300015111056	1063382396627932698319502300015111056	標準的な	-	置かれた
t3	arm64	2126764793255865396639004600030222112	2126764793255865396639004600030222112	標準的な	-	置かれた
t3	arm64	42535295865117307932780092000604442224	42535295865117307932780092000604442224	標準的な	-	置かれた
t3	arm64	850705917302346				

want to use the instance that is normally activated, so we leave it unchecked.

The next network is the VPC in which the instance will be installed, so we select aws-infra-vpc. For the next subnet, you are asked which subnet to install, so specify the public subnet where the webserver will be set up. Here we select 1a. For the next Auto Assigned Public IP, you can choose whether or not to add a global IP address that can be accessed via the Internet. Since we want the server to be accessed from the Internet, we select "enable" here.

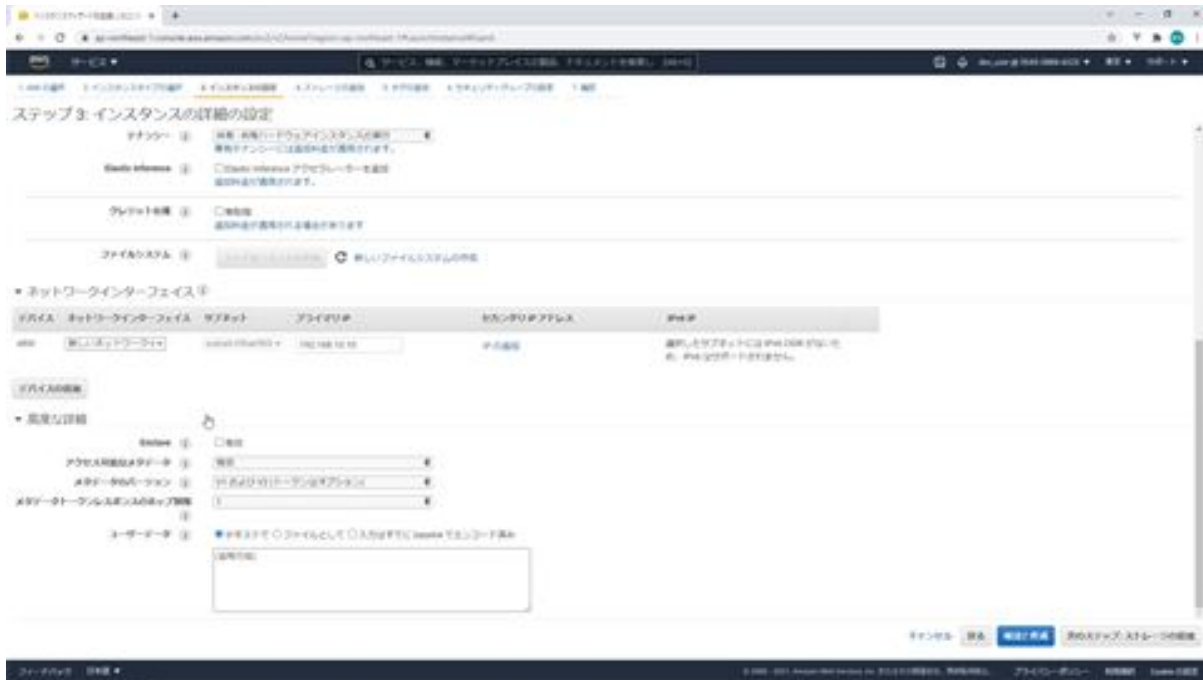
The next deployment group is designed to speed up communication between multiple EC2 instances. In this case, only one instance will be created, so leave it unchecked. The next group, Capacity Reservation, has a fixed resource limit for each availability zone in AWS, and if the limit is exceeded, the EC2 instance will not be able to be launched. If you reserve capacity, you can solve this problem by reserving resources in advance. However, if you have reserved capacity, you will be charged regardless of whether or not the EC2 instance is running. Since there is no particular need for this time, select None.

The next domain binding directory is an option to join the domain environment as is when launching an EC2 instance if you are using simple AD or Microsoft AD in AWS. Since we will not be using it this time, we will select "no directory". The next IAM role allows you to set the permissions when the EC2 instance is linked to other AWS services. In this case, we will not be linking with other services, so we will select None.

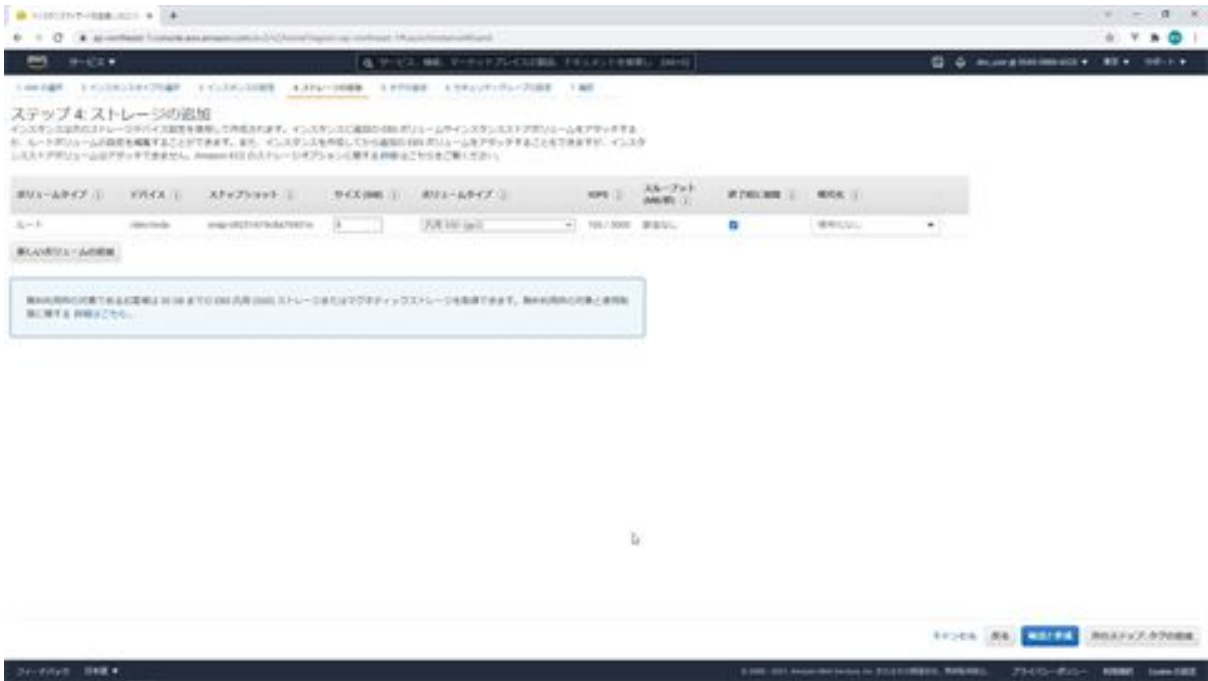
The next shutdown action specifies whether to leave the instance in a stopped state so that it can be started when it is shut down or to remove it. In this case, we want to keep it, so we select stop. The next stop hibernation action specifies whether to put the instance into a state similar to the windows sleep function after it is stopped. If it is dormant, the instance type

overloaded and the site will be slow or unresponsive. Leave this unchecked here.

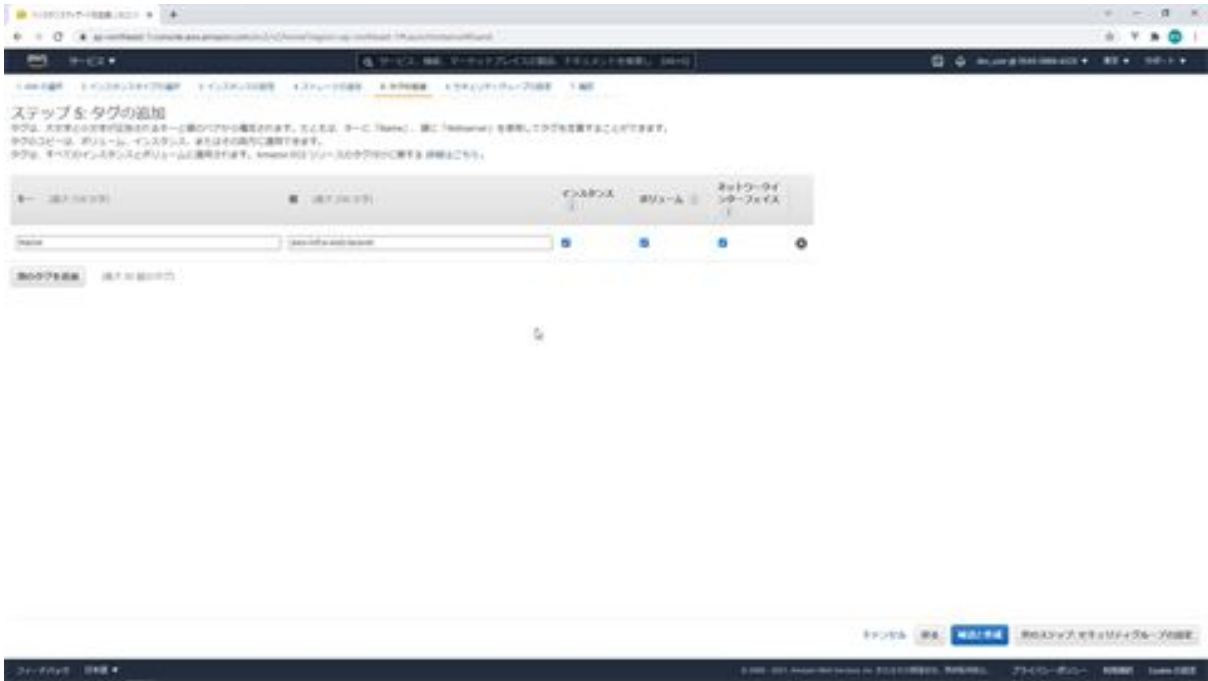
The next network interface can be a private IP as well as a public IP for the instance. In this case, we will set it to 192.168.10.10.



As for the next advanced detail, everything is fine by default. In turn, Enclave specifies whether to use a system called AWS Nitro Enclaves, which can create an isolated computing environment that improves protection and safeguards for highly sensitive information. Next is accessible metadata, access to instance metadata can be disabled by disabling the HTTP endpoint of the instance metadata service. The next version of metadata can only run V1 and V2 or V2, whereas in V1 the metadata can be retrieved by accessing it with a command, whereas in V2 a session token must be obtained before accessing the metadata. The hop limit for the next metadata token response is the number of network hops the metadata token can travel. The next user data can be a script that is run when the instance is started. Then next click on the Add Storage button to go to the Add Storage screen.

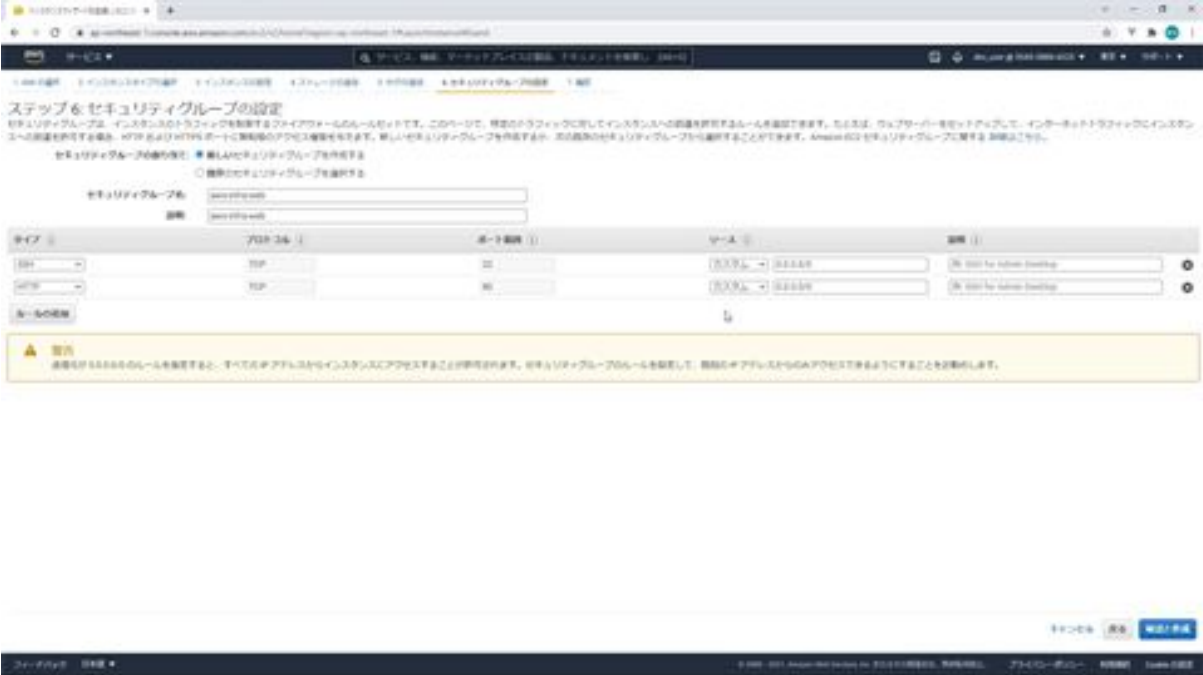


Additional settings, but this time we will not be adding any, so click on the Add Tag button.

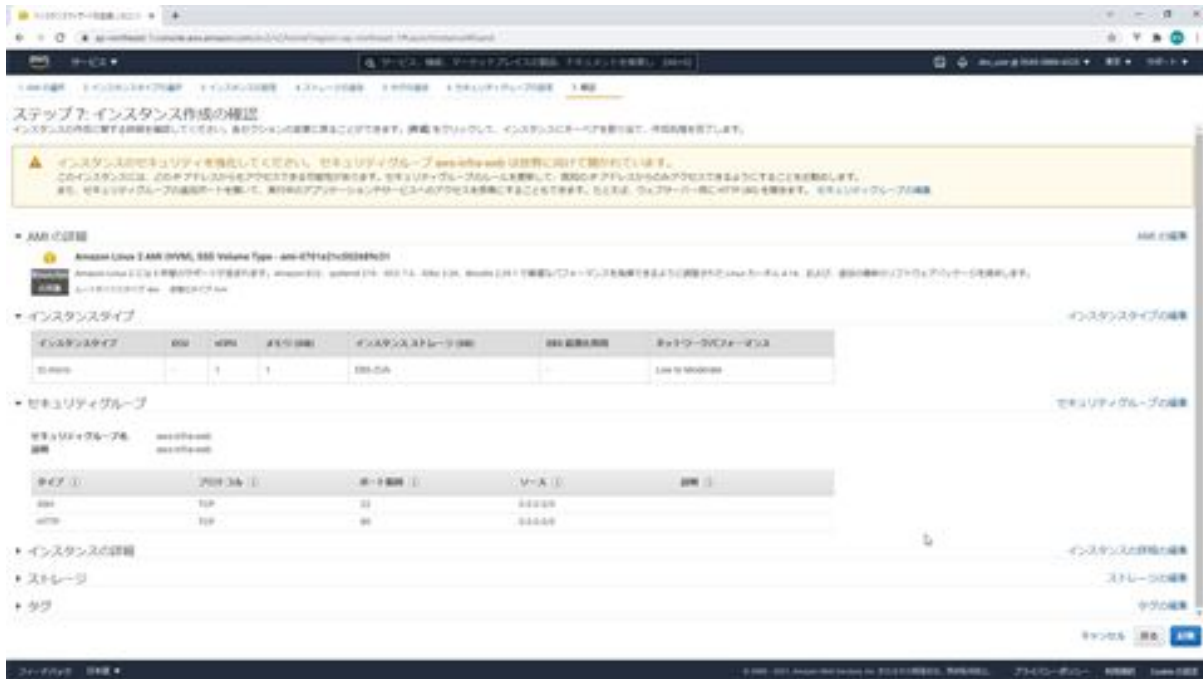


In the Add Tag section, we will add the name of the instance. Click on the link that says Add a name tag and enter aws-infra-web-laravel in the value

section. Once entered, click the Configure Security Group button.



The security group sets security to the instance. Here we create a new security group. Enter aws-infra-web as the security group name. Next, enter the same for the description. Here you see a warning at the bottom, this is because the default security group settings allow connections from anywhere via ssh. Originally, you would specify your office or home IP to restrict the connection. This time, we want to add HTTP because we want to allow HTTP communication from a function called ELB. Click on the Add Rule button, set the Type to HTTP, set this source part to Custom, and enter 0.0.0.0/0. When you are done, click the Confirm and Create button.

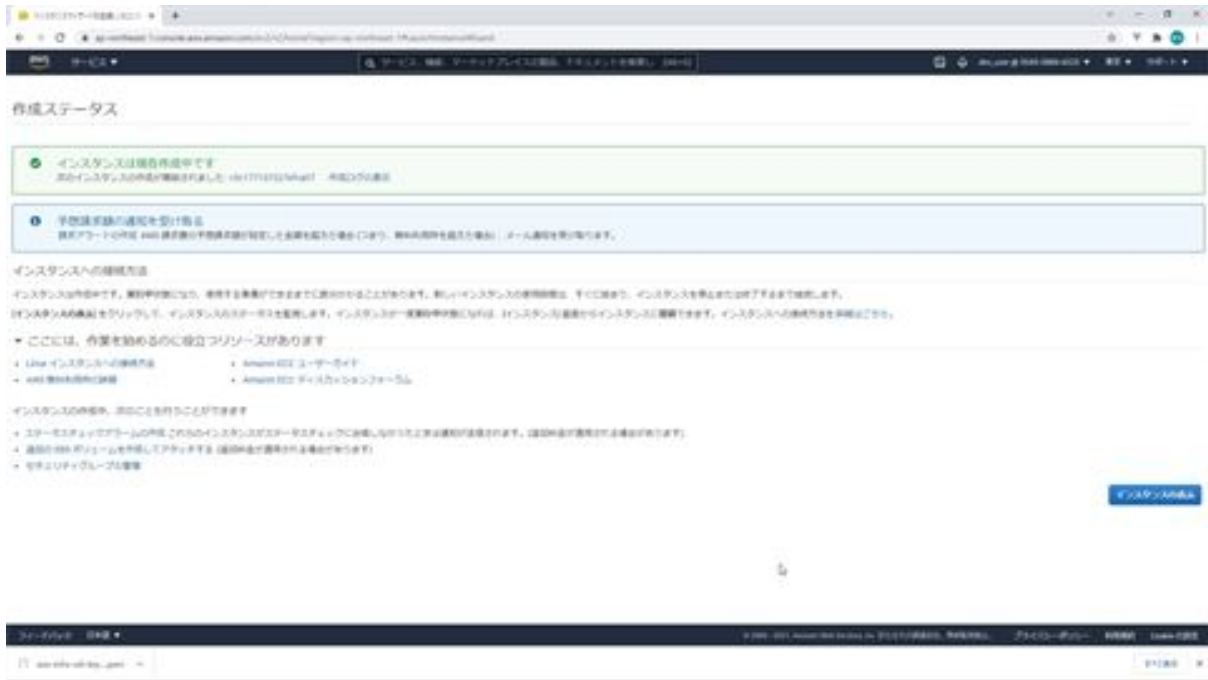


Check the confirmation screen and if the settings are correct, click the Launch button to launch the instance.

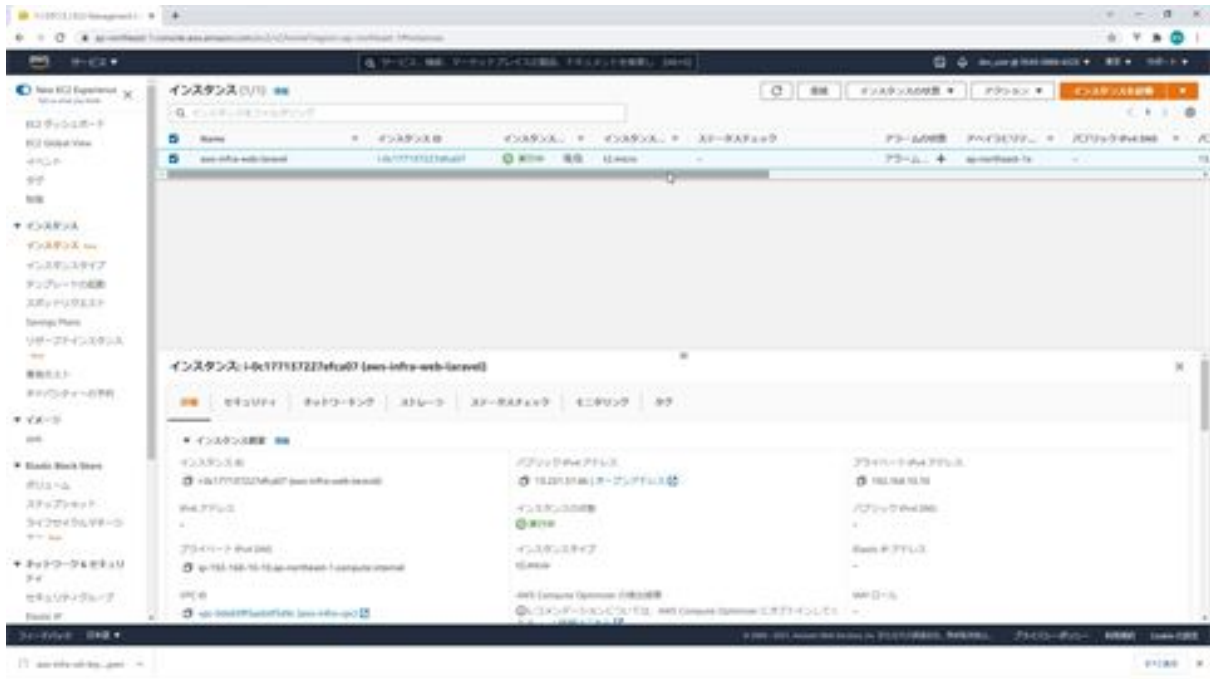


You will then be presented with a screen for specifying a key pair. The key pair is the key to log in to the instance, so create a new key pair. Select

Create a new key pair and enter the name of the key pair as aws-infra-ssh-key. Then click the Download Key Pair button and save the downloaded file. This key pair file cannot be downloaded later, so be sure not to lose it. After downloading the key pair, click the Create Instance button.

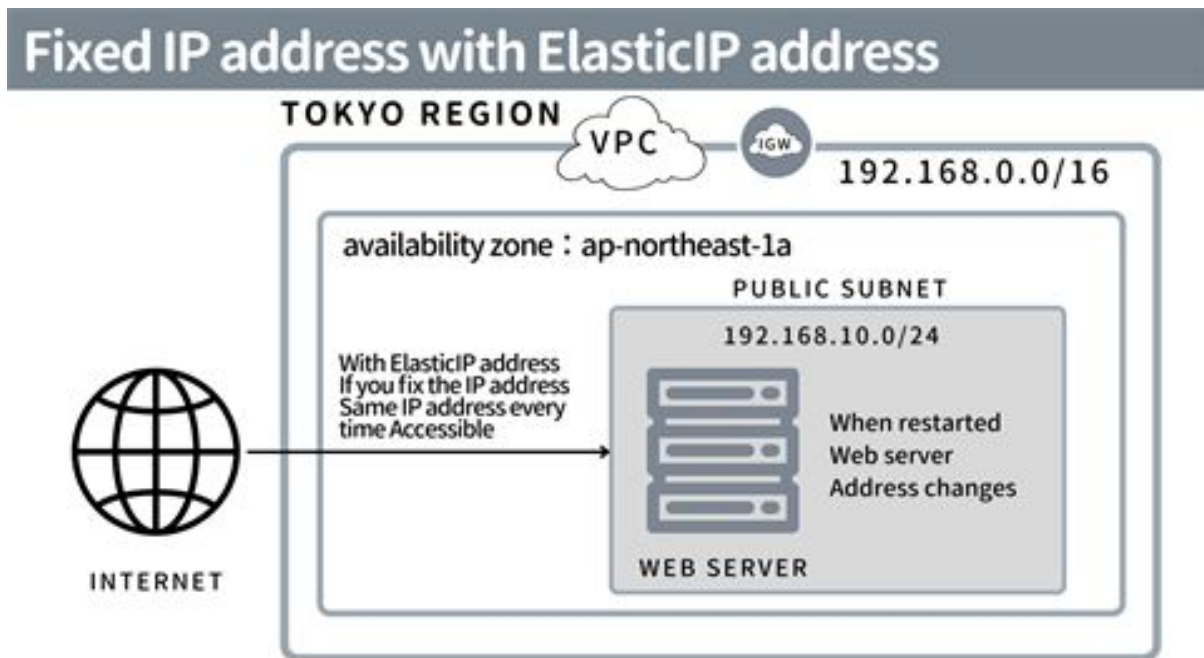


This will start the instance. Return to the instance screen to see if there is an instance.



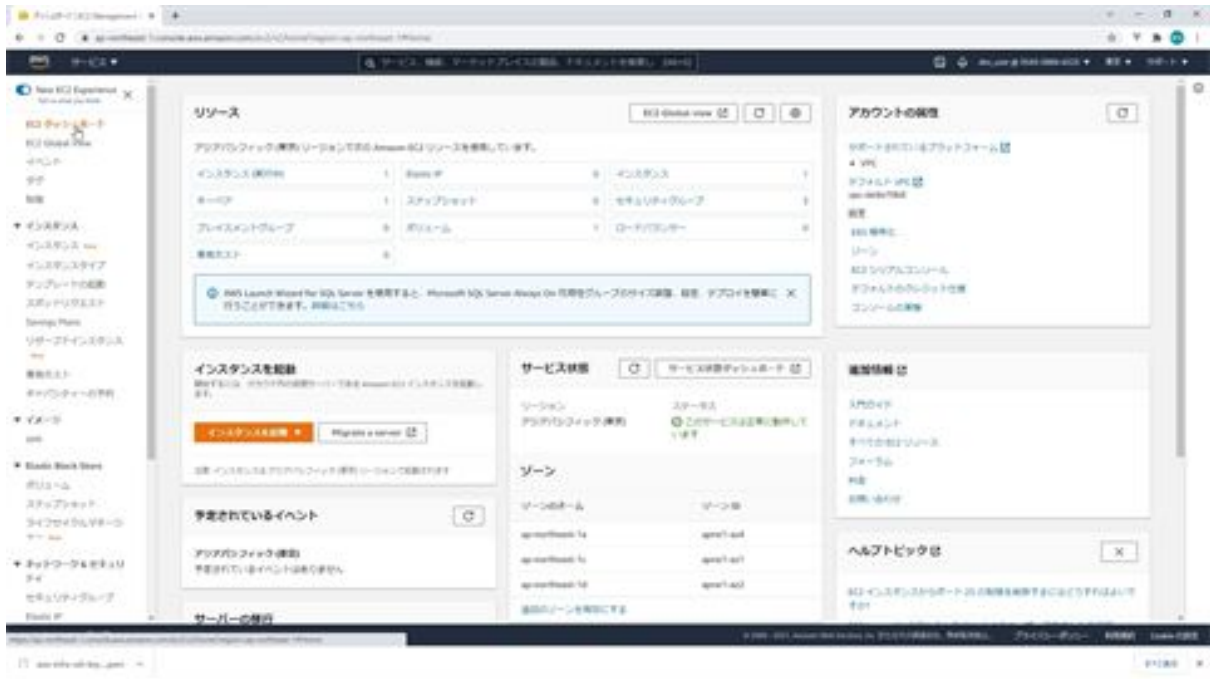
Now that the instance has been confirmed, the EC2 instance configuration is complete.

Let's fix your IP address with ElasticIP address

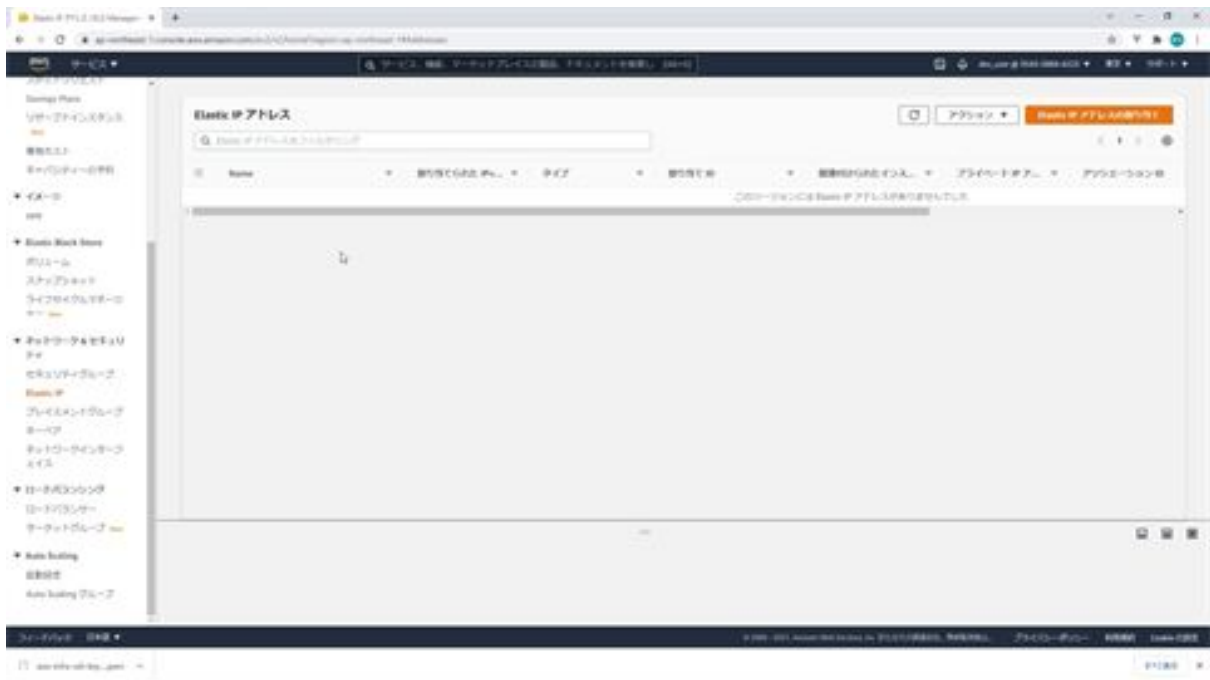


Next, we will fix the IP address with an Elastic IP address, and you may wonder why we set an Elastic IP address when we have an IP address in EC2. The address will be assigned. Since it can be inconvenient to change IP addresses when operating on a server, the Elastic IP address is used this time to fix the public IP address. There is a caveat to the Elastic IP address: if it is associated with an EC2 instance and that instance is running, it is free, but if it is not, you will be charged for it. If you are not using an Elastic IP address, be sure to free up the Elastic IP address. Now let's do the actual work.

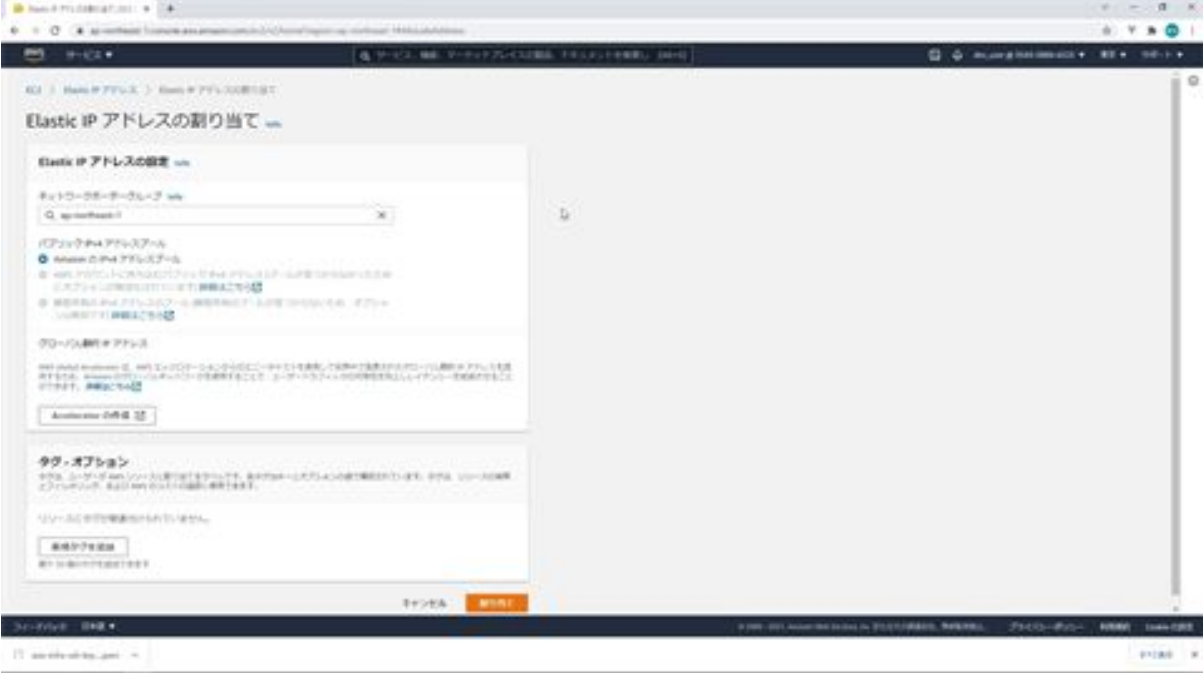
Open the AWS management console and go to the EC2 dashboard.



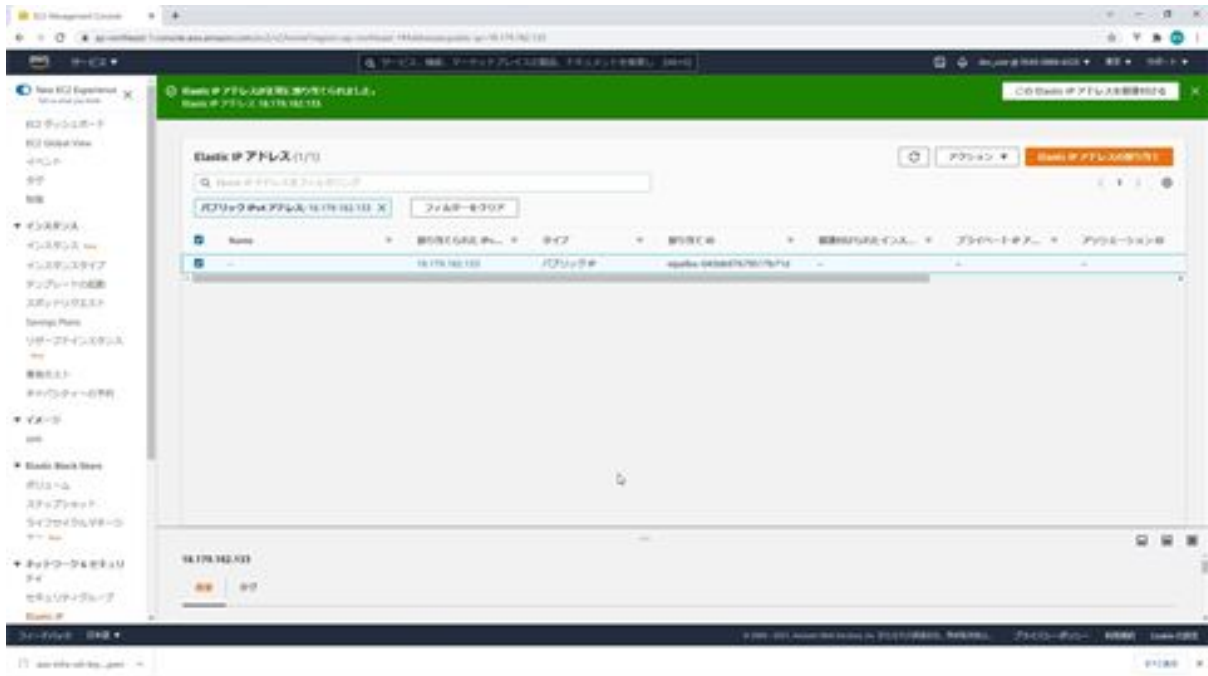
First, we will reserve an Elastic IP address: click on Elastic IP in the menu on the left side of the EC2 dashboard. Next, click the Assign Elastic IP Address button.



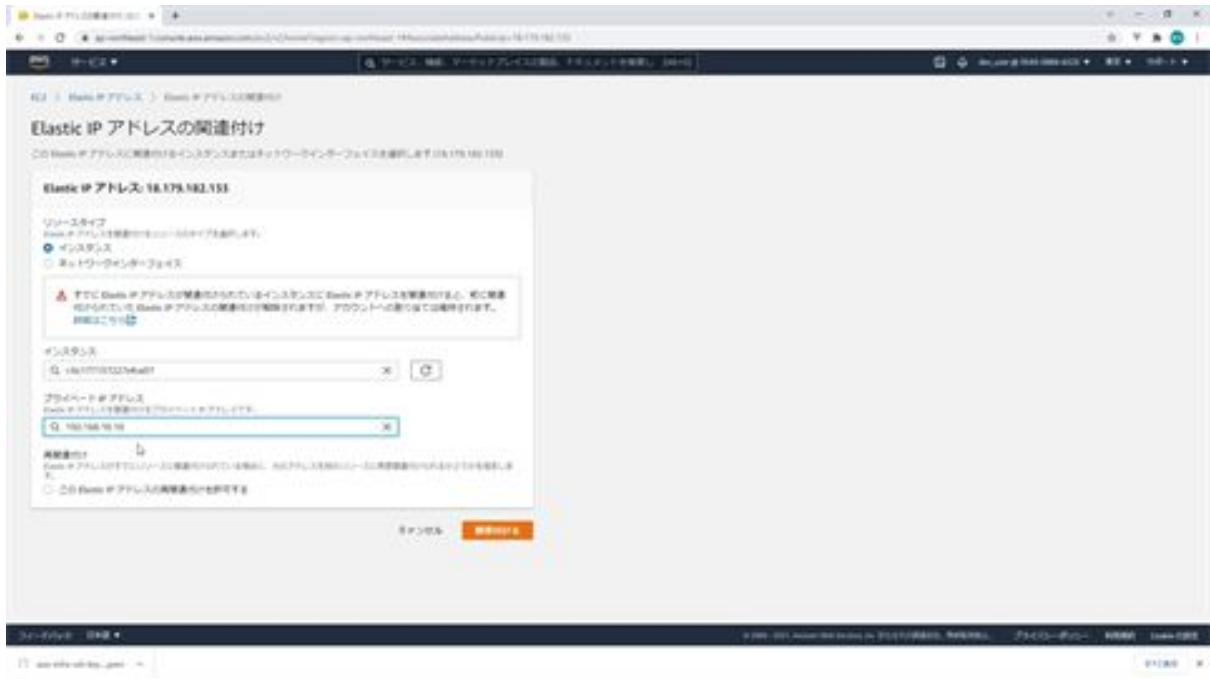
Next, the Elastic IP address configuration screen will appear. Leave the default settings and click the Assign button. You will then be allocated a new IP address.



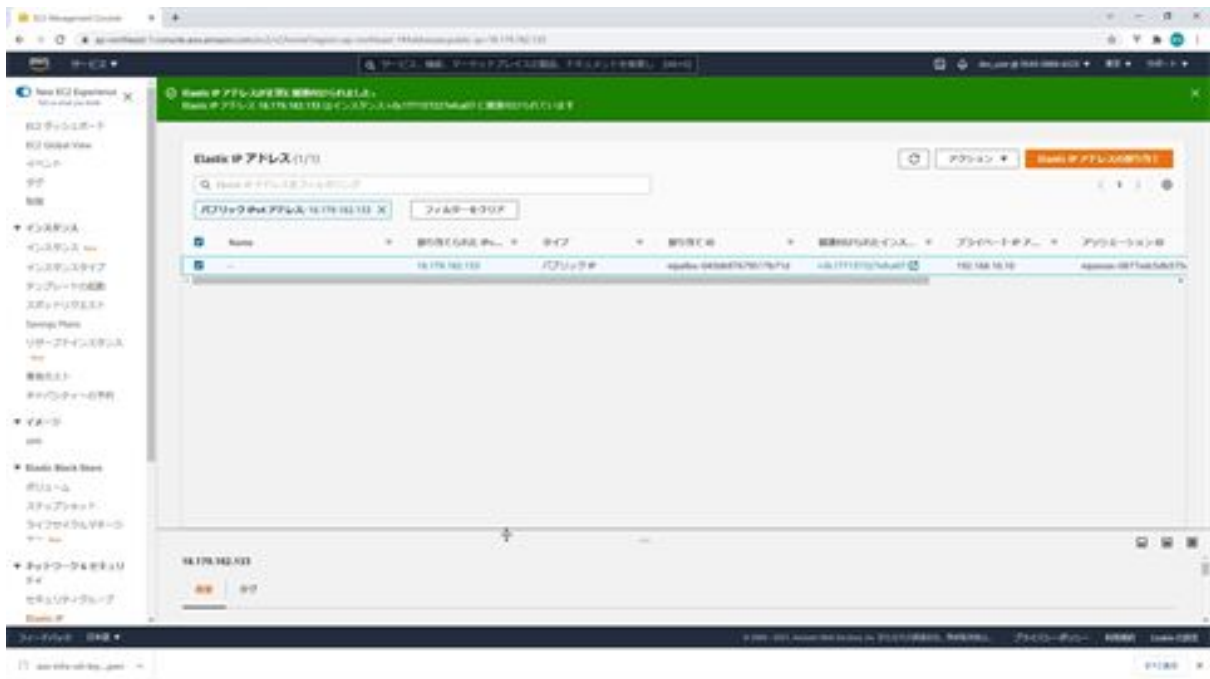
Next, we will assign it to an EC2 instance. Select the IP address you have allocated and click on Associate Elastic IP Address from the Action menu above.



This screen allows you to associate an Elastic IP address. First, select the resource type, which is an instance. Next is the instance selection, here we select aws-infra-web-Laravel. Next, for the Private IP, select 192.168.10.10. Again, for Associate, this should be checked if an Elastic IP address is already associated with it, but in this case, there is no association, so leave it unchecked. Now click on the Associate button.



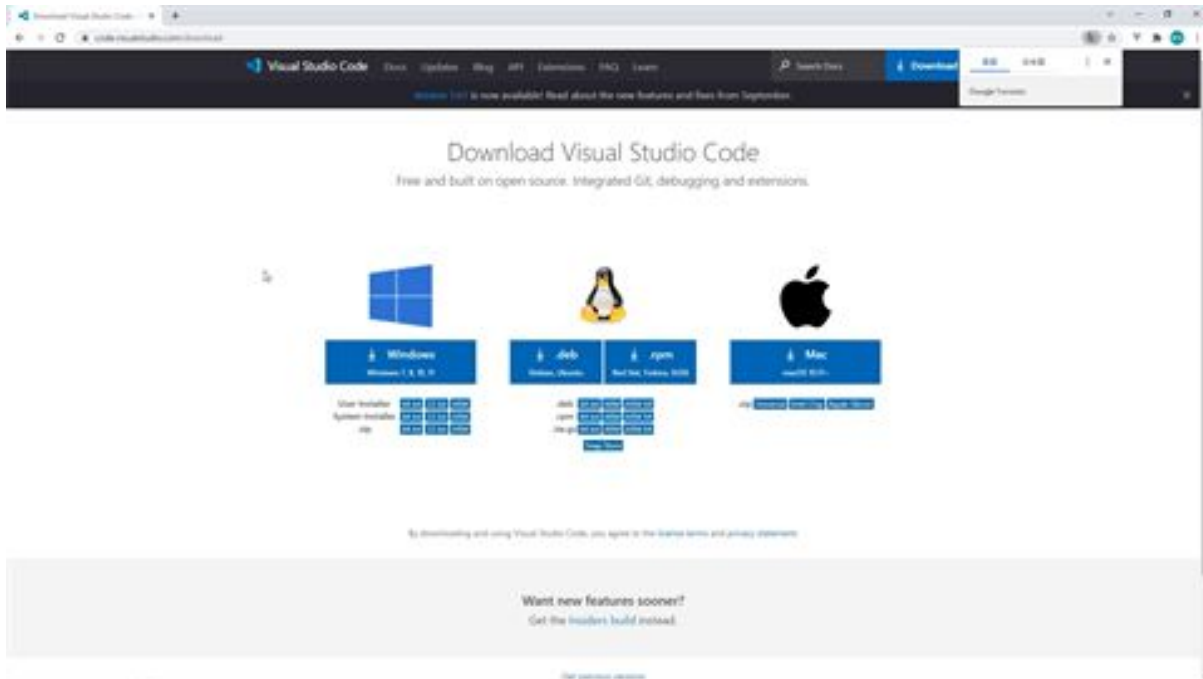
The Elastic IP address association is now complete.



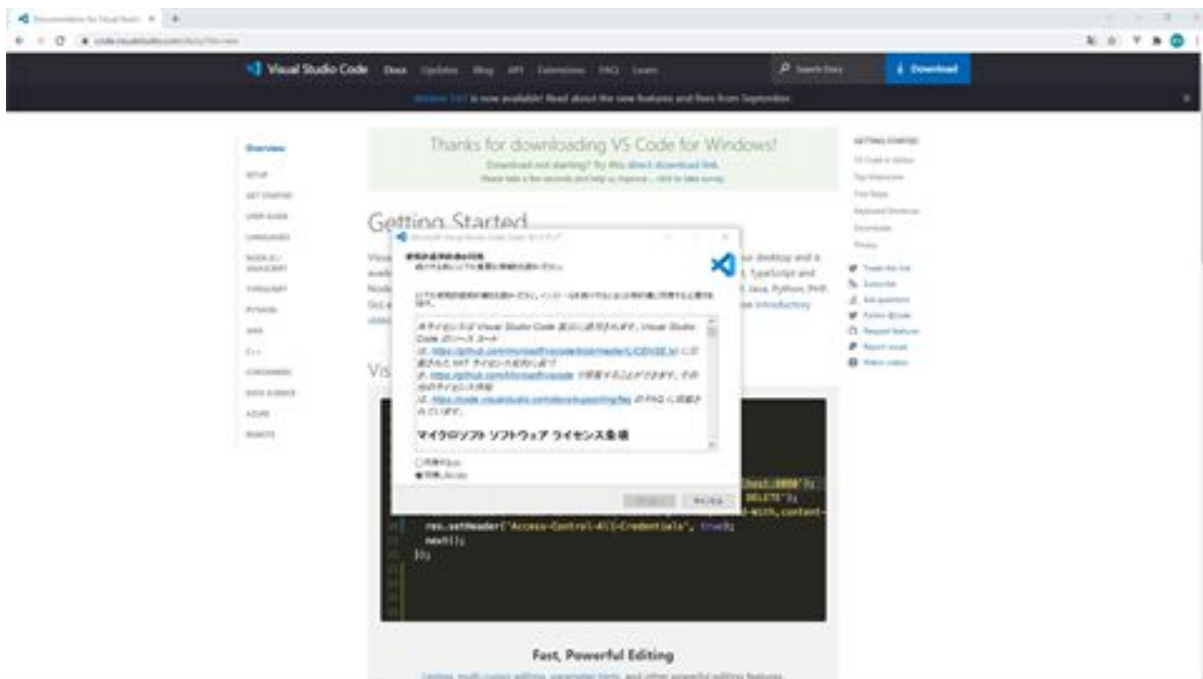
Install Visual Studio Code



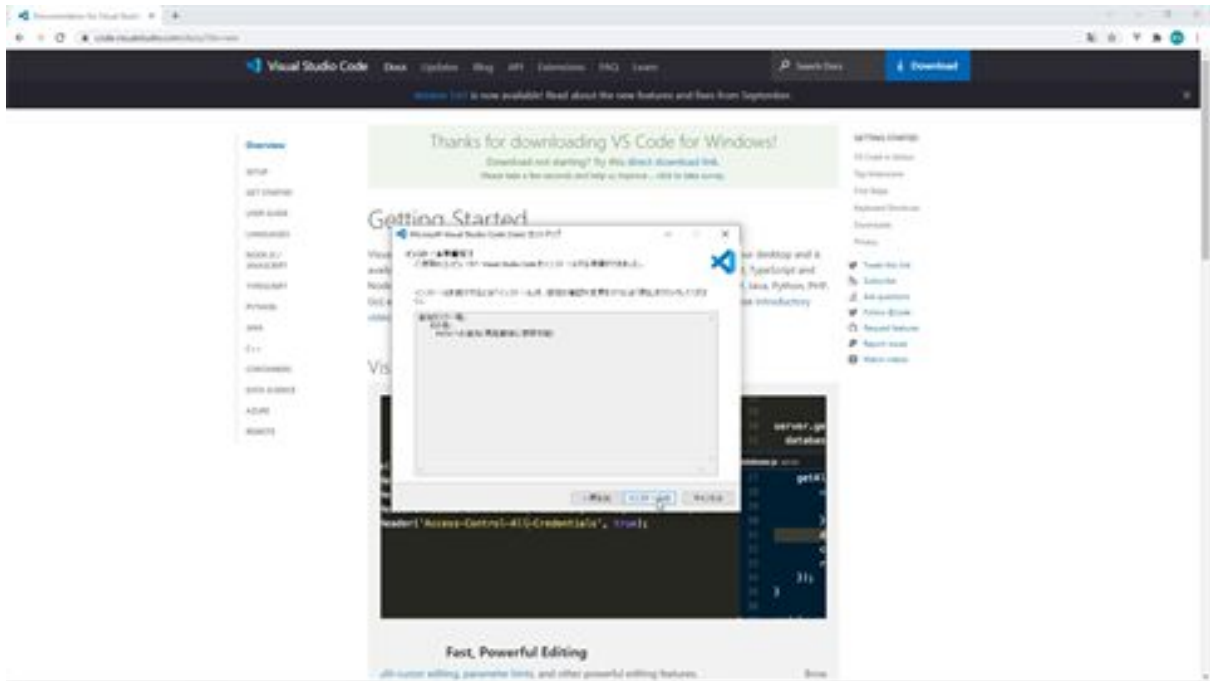
Next, we will install Visual Studio Code to connect to the EC2 instance. First, search for Visual Studio Code on Google. Then the Visual Studio Code site will appear, so access the Visual Studio Code site. Next, click the Download Now button, and the download screen will appear, so select your platform and download.



Now that the download is complete, launch the installer. Select Agree on the license agreement and click the Next button.

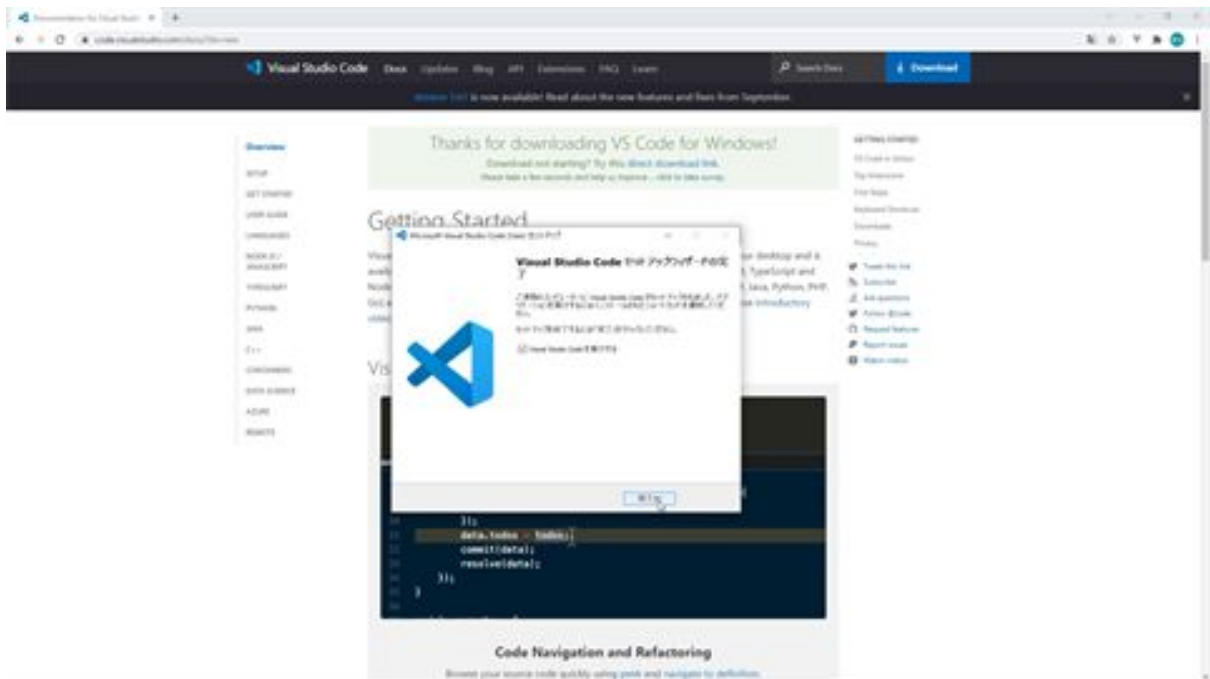


In the selection of additional tasks, leave the defaults as they are and click the Next button. On the next screen, click the Install button.



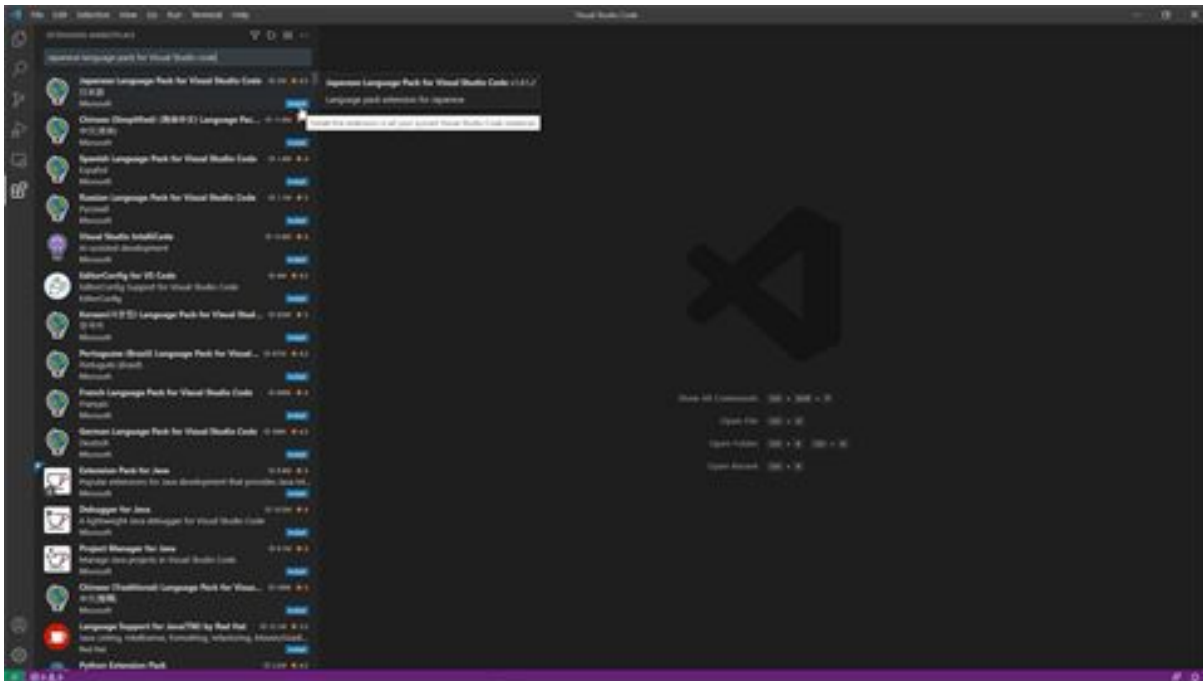
Please wait a few moments for the installation to complete.

The Setup Complete screen appears. This completes the installation of Visual Studio Code.

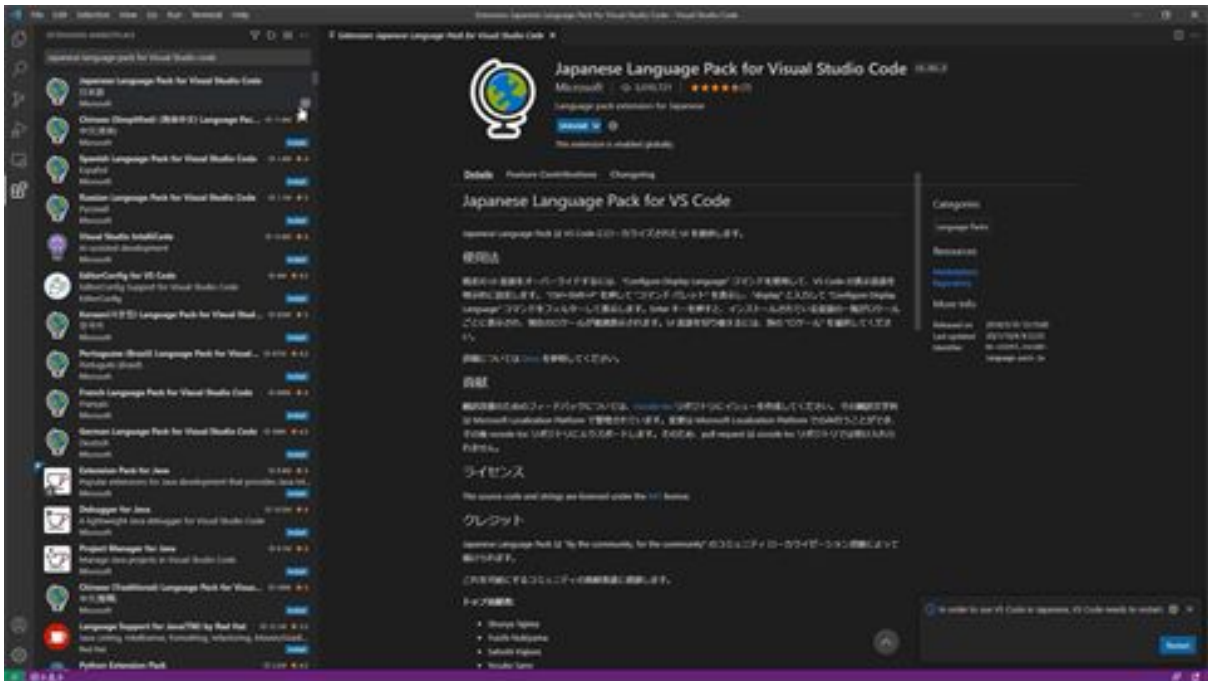


Connect to an EC2 instance with Visual Studio Code

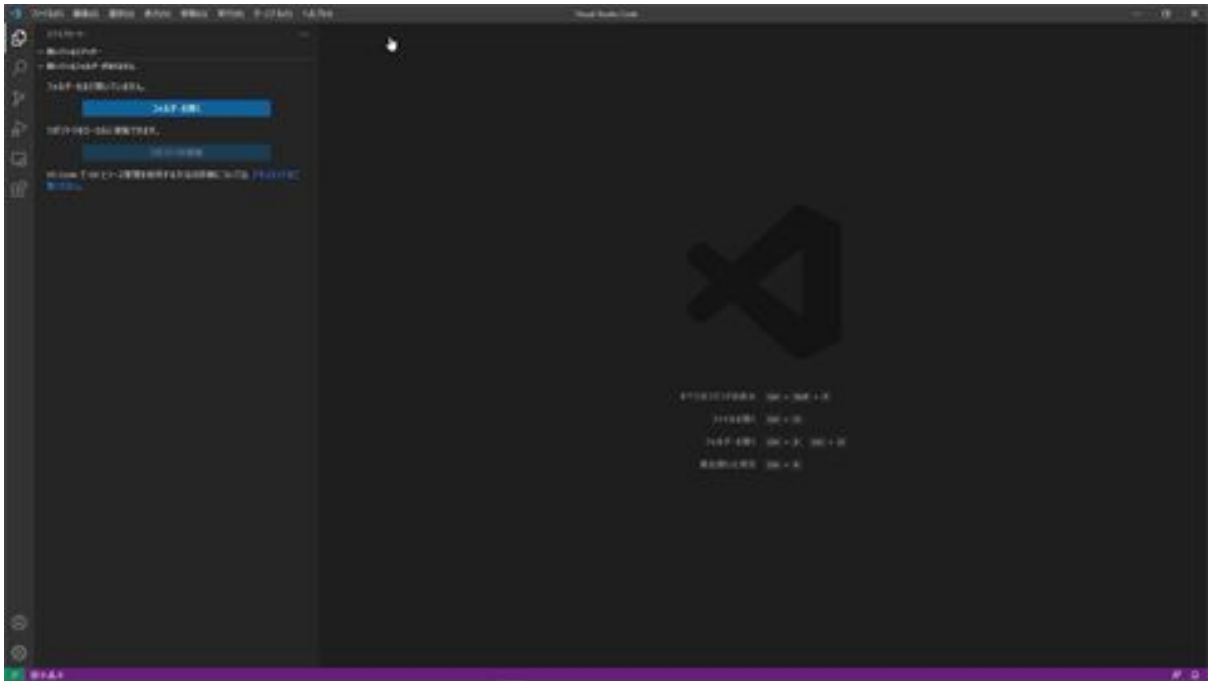
We will now connect to the EC2 instance, but before that, we will convert Visual Studio Code to Japanese. Click on Extensions in the menu on the left and search for "Japanese Language Pack for Visual Studio Code". Then the extension "Japanese Language Pack for Visual Studio Code Japanese" will appear, and install.



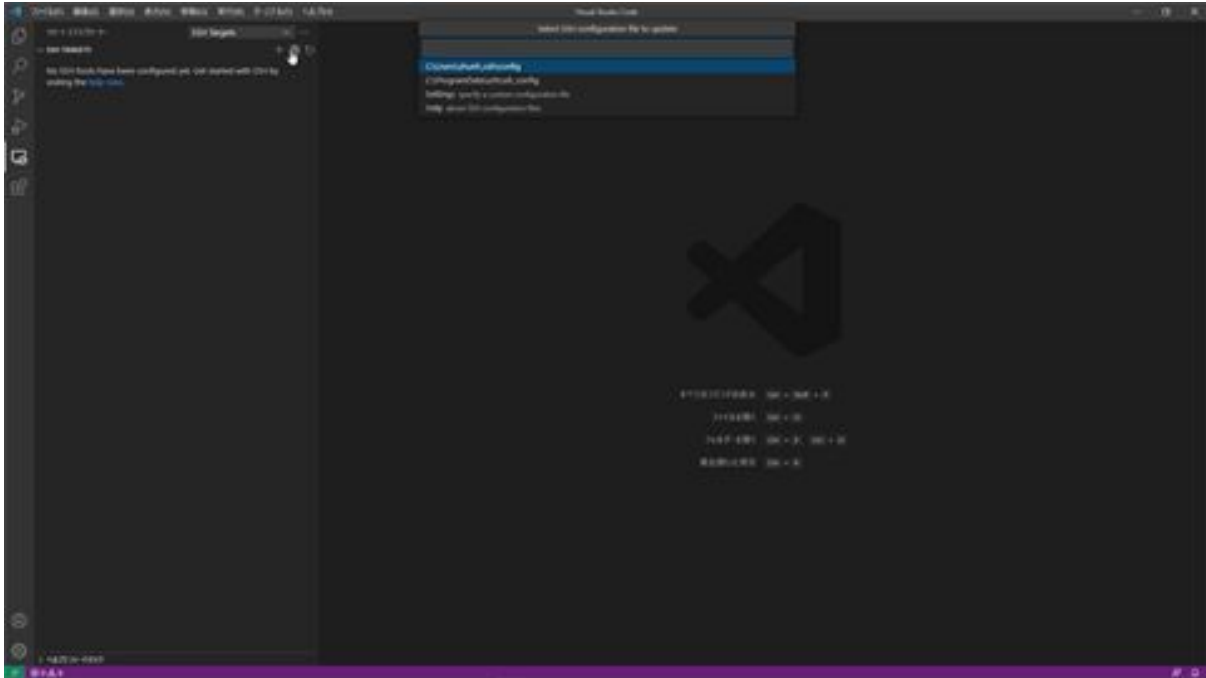
After installation is complete, reboot the system. After the installation is complete, click "Restart" in the lower right corner of the screen.



When Visual Studio Code is restarted, menus and other information are automatically displayed in the newly installed language, Japanese.



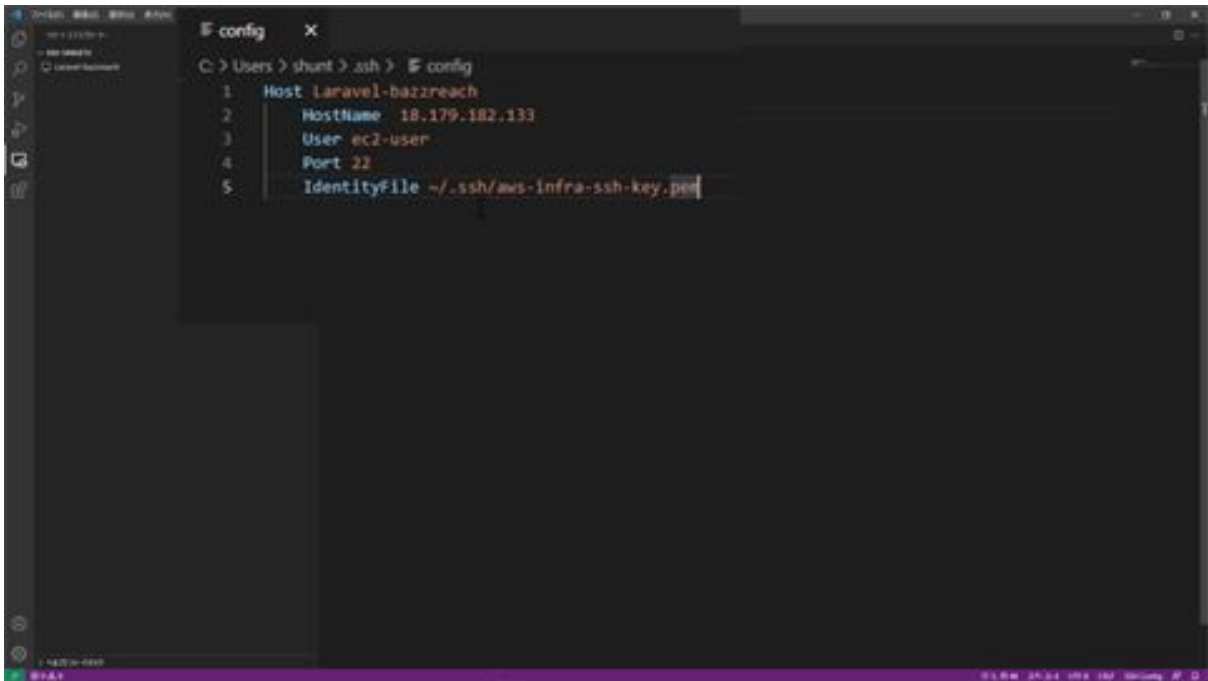
Next, we will install Remote SSH, an extension for connecting to EC2 instances. Type Remote SSH in the search field for extensions. Remote



The next step is to configure the config file, which contains the server IP, login user, port number, and private key path. First, the name of the host, which in this case is Laravel-bazzreach, since we will be creating a Laravel application called bazzreach.

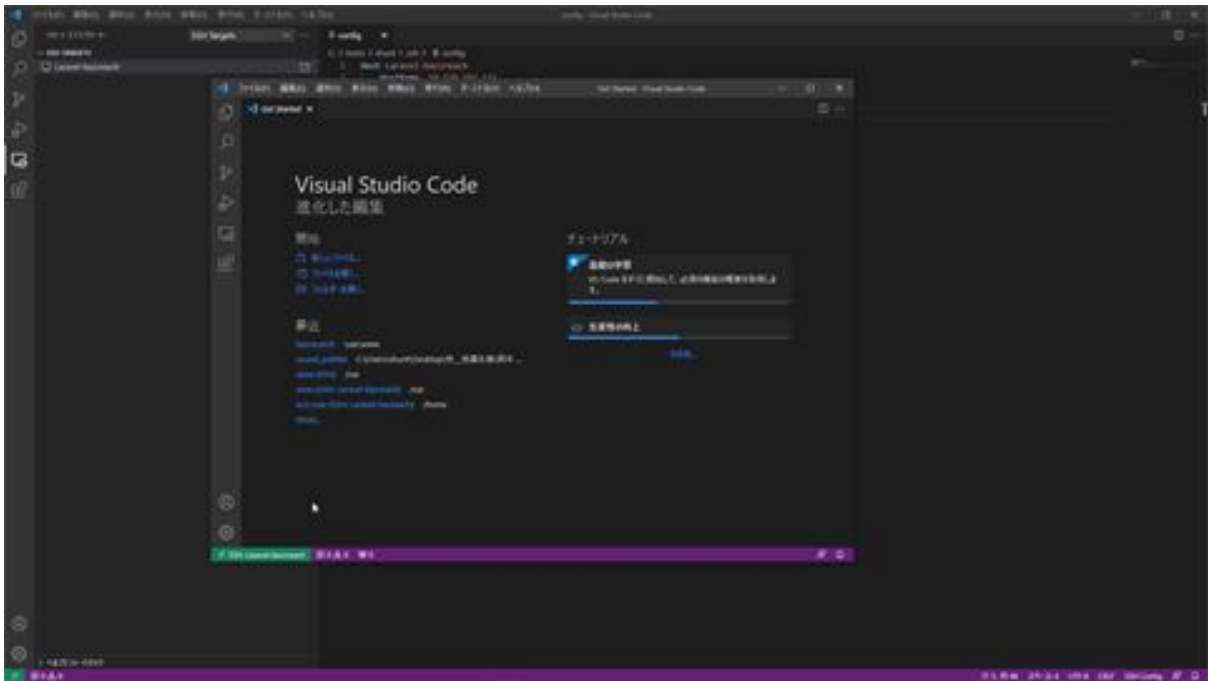
Next, set the server IP to the previously obtained Elastic IP. Next, set the login user to ec2-user. Set the port number to 22, and then set the path to the private key.

This specifies the pem file downloaded when the EC2 instance was created. pem file should be placed in the same location as the config file and set to `~/.ssh/aws-infra-ssh-key.pem`.

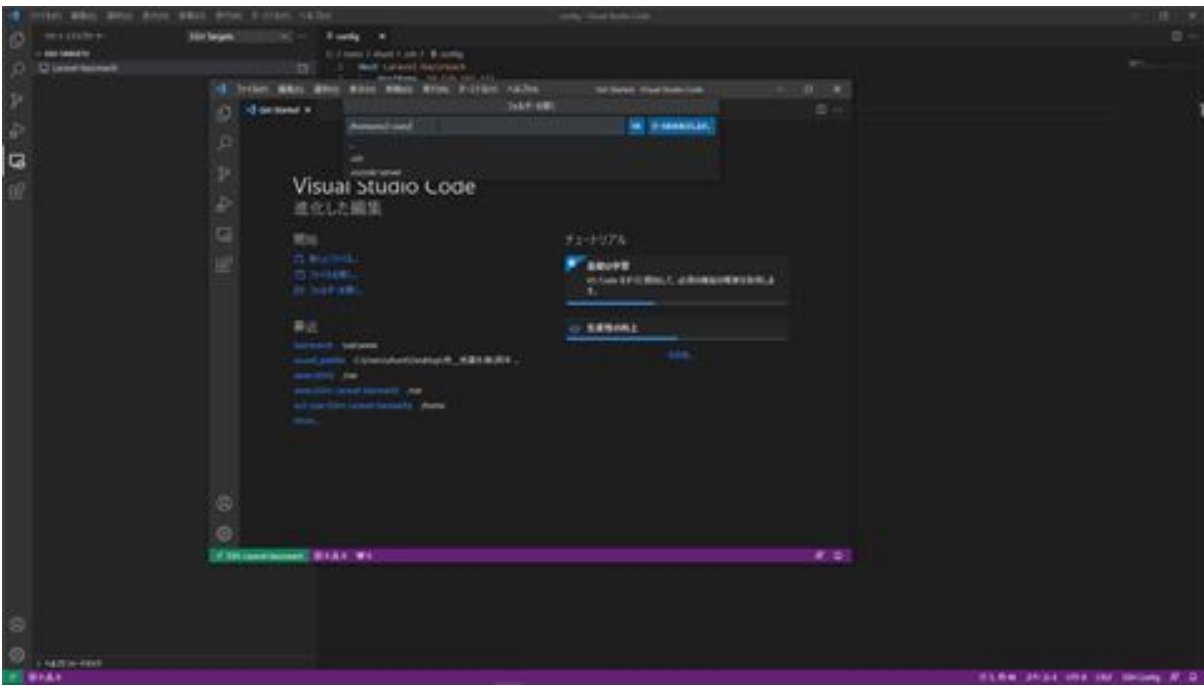


```
C:\Users> shunt > .ssh > vim config
1 Host Laravel-bazzreach
2   Hostname 18.179.182.133
3   User ec2-user
4   Port 22
5   IdentityFile ~/.ssh/aws-infra-ssh-key.pem
```

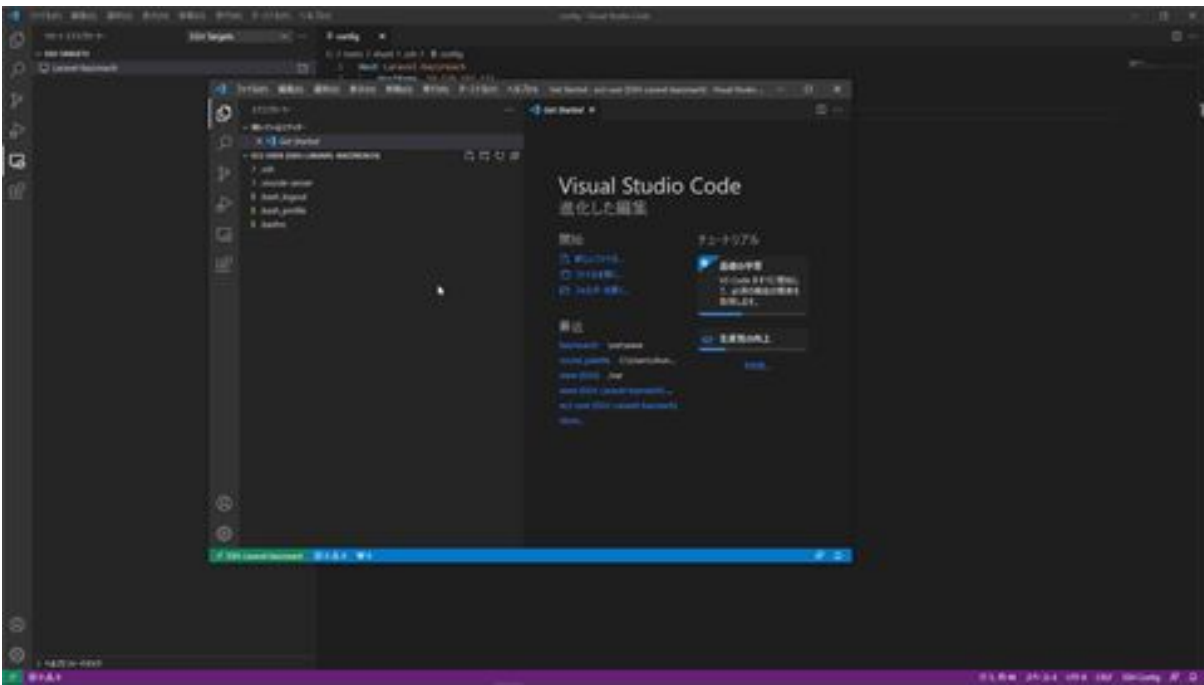
Finally, we will connect to the EC2 instance. From the Remote Explorer, click on the window icon to the right of Laravel-bazzreach listed in the SSH target. After a few moments, you will be connected to the EC2 instance. Here, Laravel-bazzreach is displayed and you are connected to the EC2 instance.

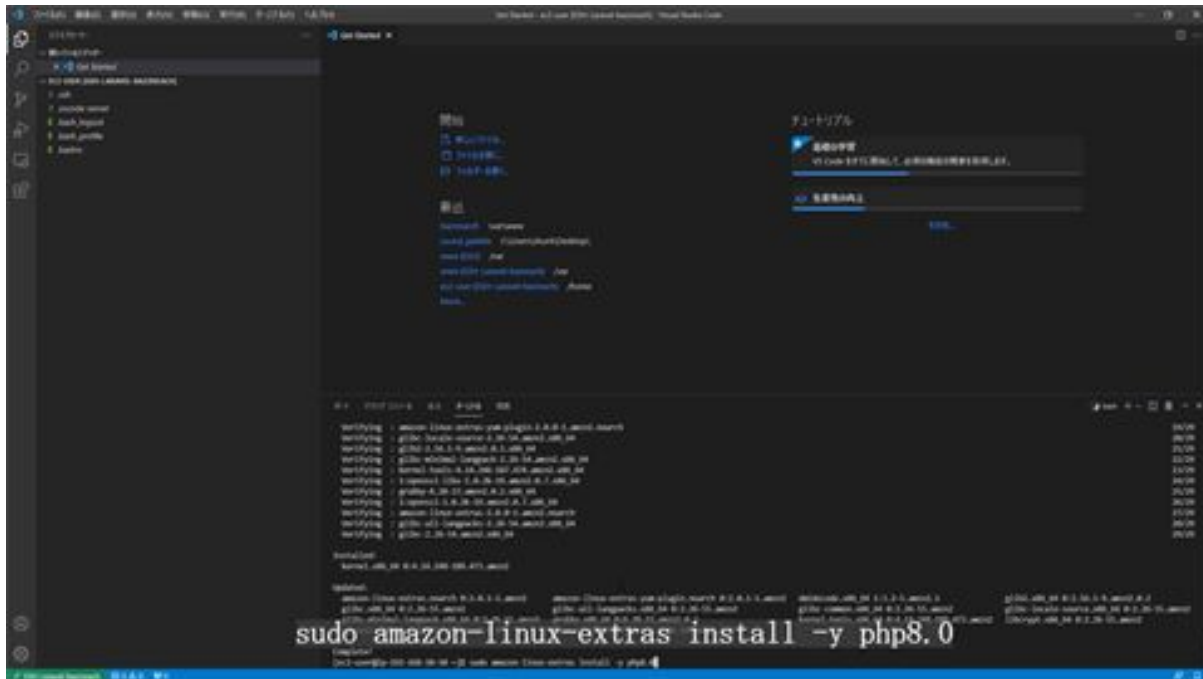


Next, we want to view the ec2-user folder. Click Open Folder from File to access the ec2-user folder.



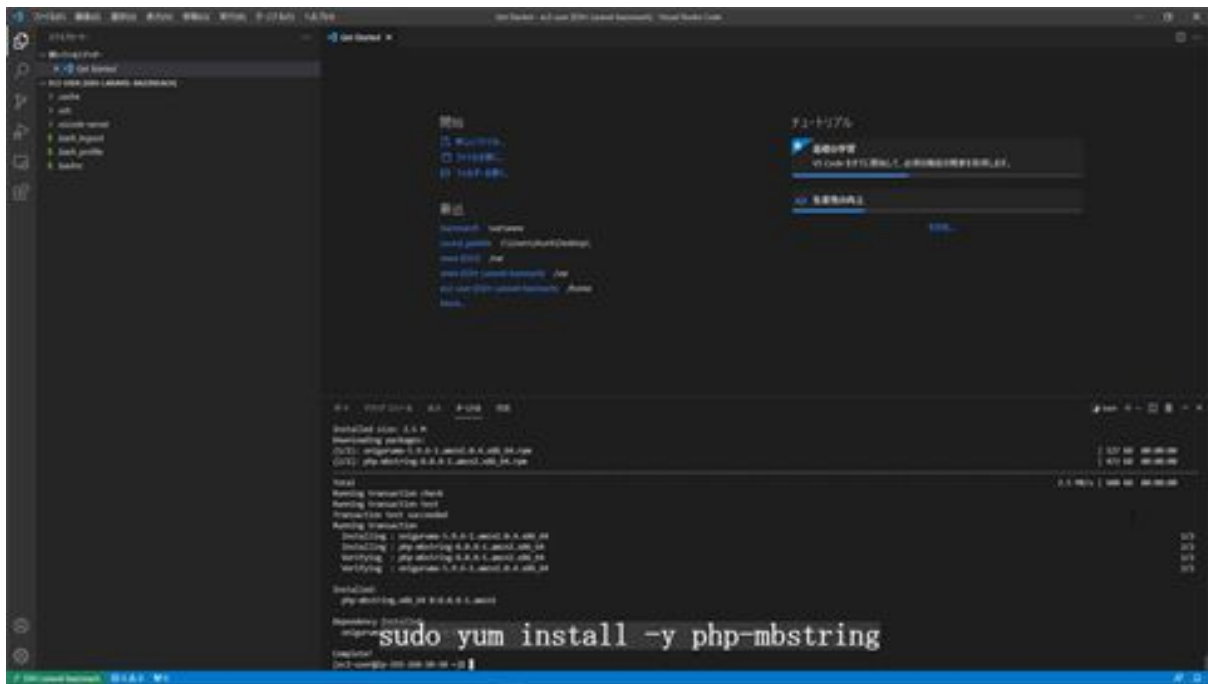
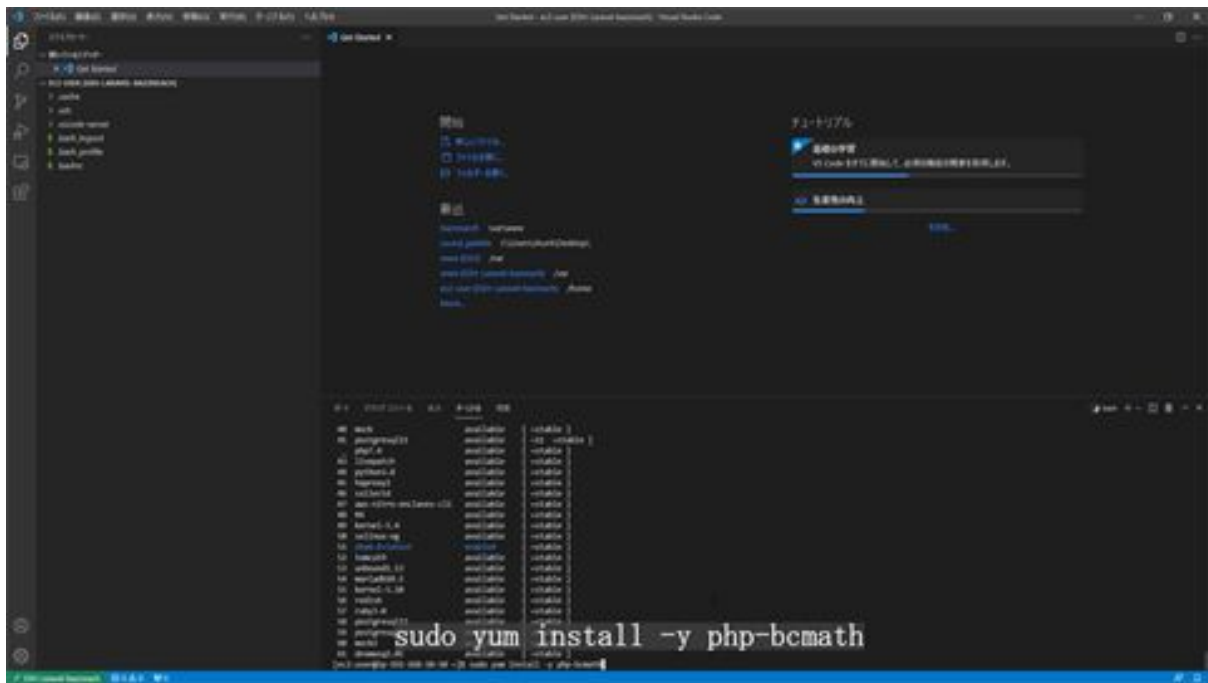
The ec2-user folder is displayed and the files inside can be confirmed. This completes the connection setup with the EC2 instance.

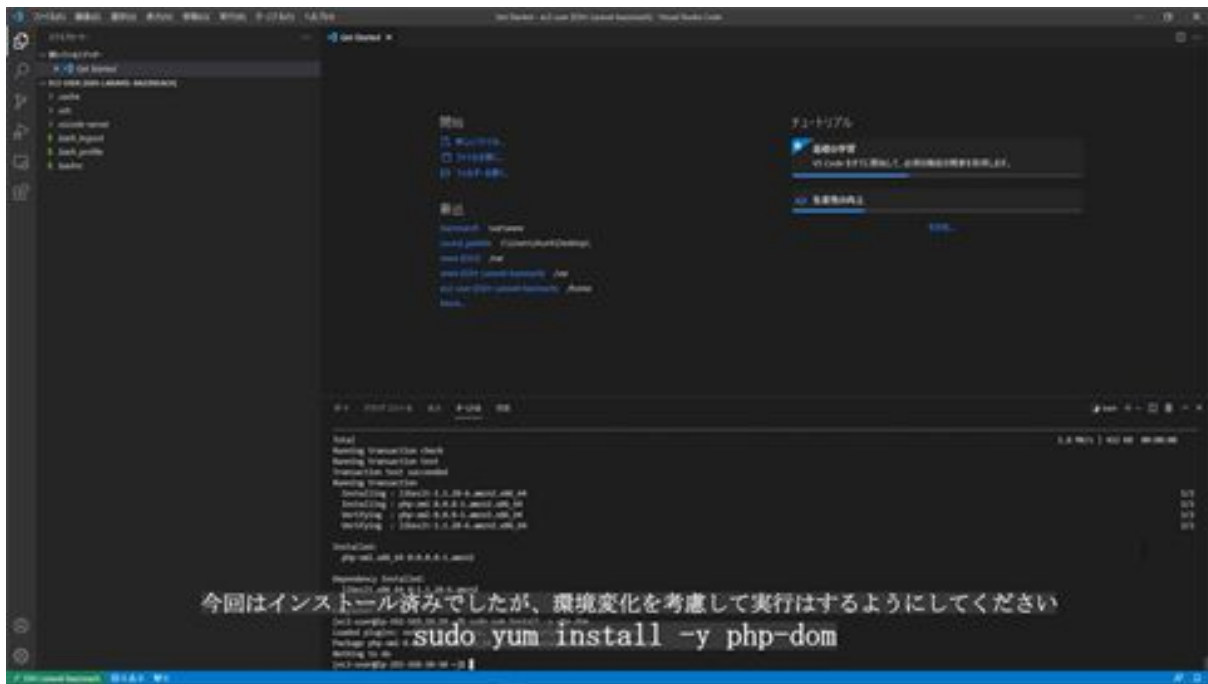
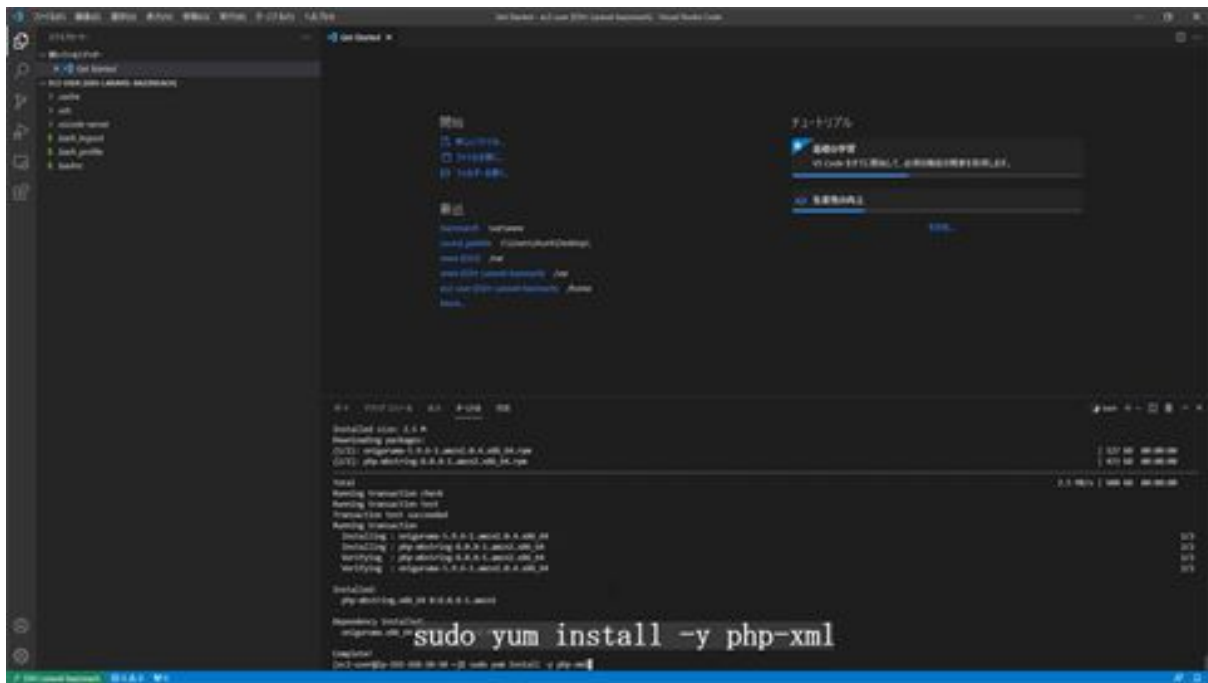


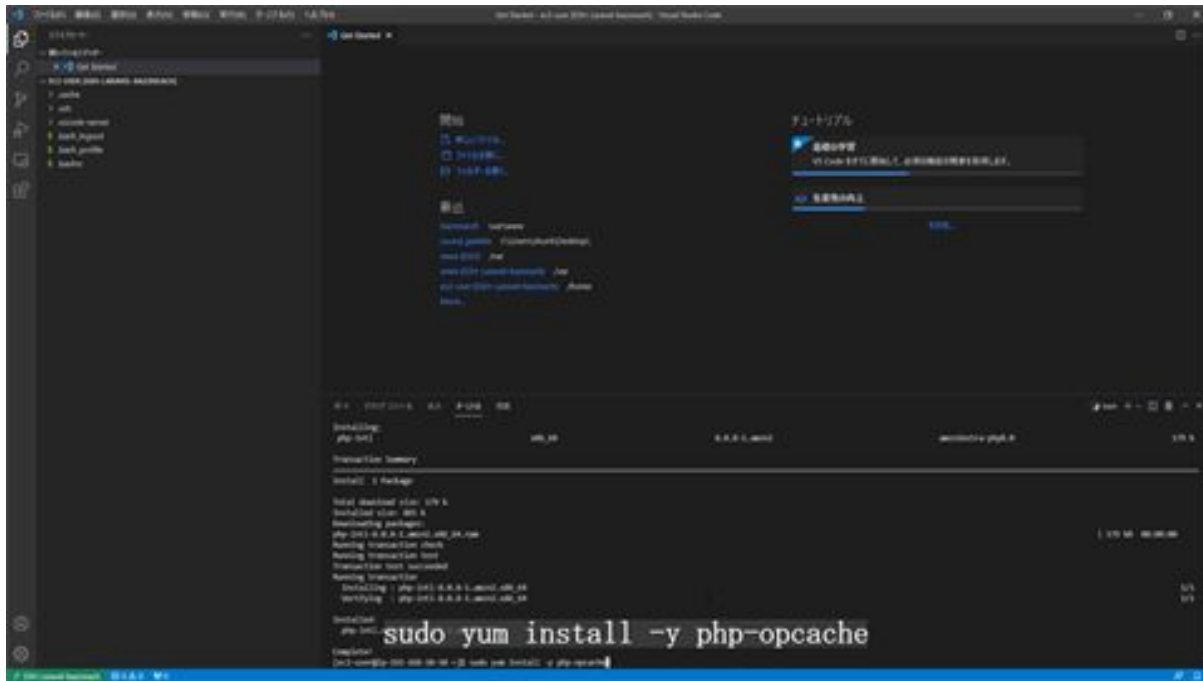


Next, install the necessary modules. There are six required modules, so we will install each of them. Execute the following commands for each.

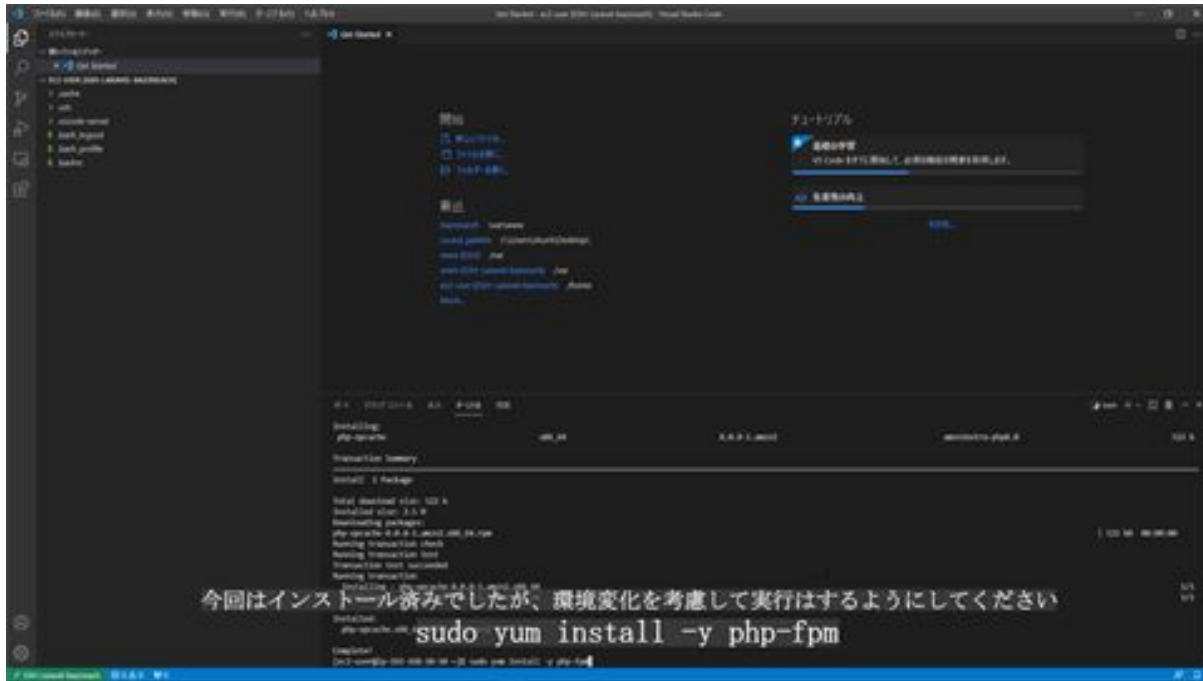
```
sudo yum install -y php-bcmath  
sudo yum install -y php-mbstring  
sudo yum install -y php-xml  
sudo yum install -y php-dom  
sudo yum install -y php-gd  
sudo yum install -y php-intl
```



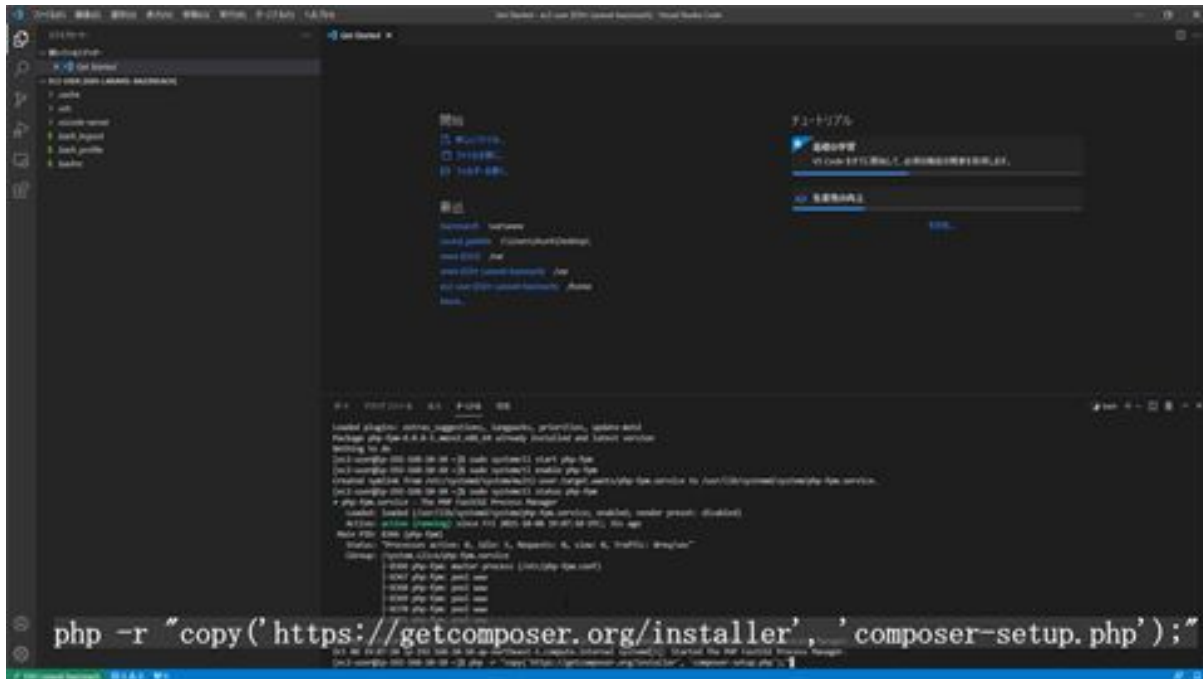




Next, we will install something called PHP-FPM, which stands for FastCGI Process Manager, an officially supported PHP standard application server.



Next, start PHP-FPM. Change the configuration so that PHP-FPM will start automatically even if the server is restarted. Check the status after

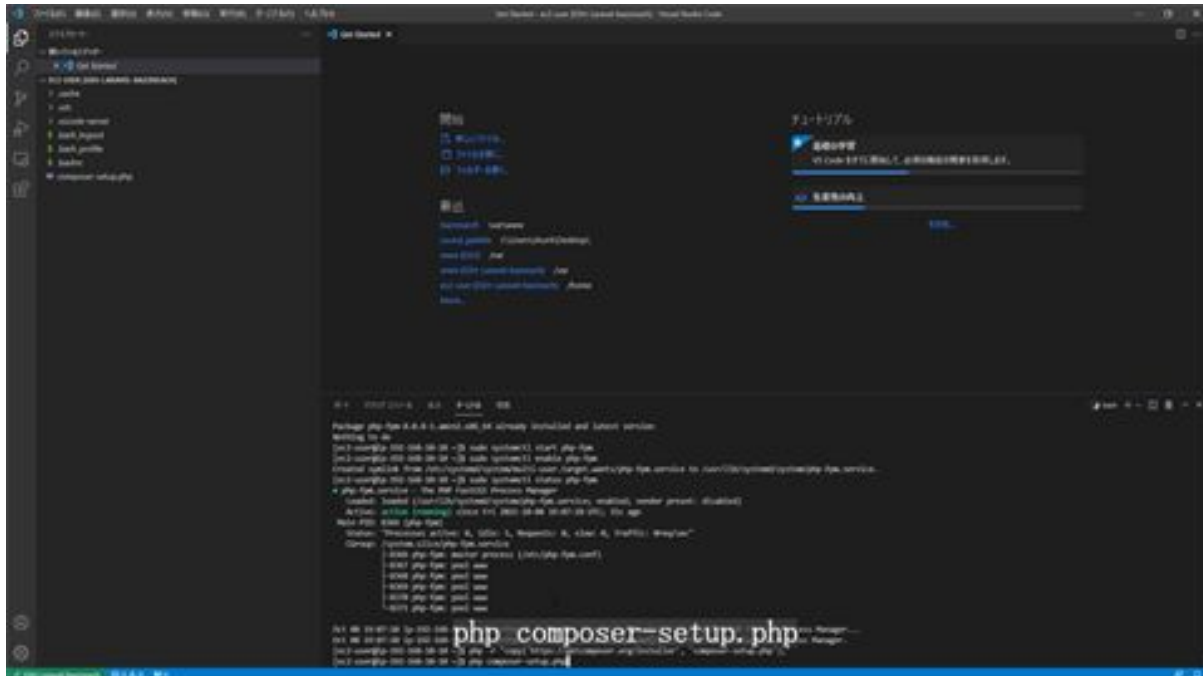


```
Loaded plugins: extras, suggestions, suggestions, protection, update-selinux
Package php-fpm.x86_64-1.0.0-1.el7.centos already installed and latest version
Nothing to do

[1/3] Installing php-fpm.x86_64-1.0.0-1.el7.centos: start php-fpm
[2/3] Installing php-fpm.x86_64-1.0.0-1.el7.centos: install php-fpm
Created symlink from /usr/lib/systemd/systemd/user-runtime-dir/php-fpm.service to /usr/lib/systemd/systemd/php-fpm.service.
[3/3] Installing php-fpm.x86_64-1.0.0-1.el7.centos: status php-fpm
* php-fpm.service - The PHP FastCGI Process Manager
   loaded: loaded (/usr/lib/systemd/systemd/php-fpm.service; enabled; vendor preset: disabled)
   active: active (running) since Fri 2015-04-04 01:07:34 EDT; 10s ago
 Main PID: 4566 (php-fpm)
 Status: "ProcessManager: active, 4, listen: *, Requestor: 4, class: 4, traffic: 0req/sec"
 CGroup: /systemd/systemd/php-fpm.service
          └─ php-fpm: master process (/usr/lib/php.conf)
                php-fpm: pool www
                php-fpm: pool www
                php-fpm: pool www
                php-fpm: pool www
                php-fpm: pool www

php -r 'copy('https://getcomposer.org/installer','composer-setup.php');'
```

Now that the installer has been copied, continue with the installation by running `php composer-setup.php`.



```
Loaded plugins: extras, suggestions, suggestions, protection, update-selinux
Package php-fpm.x86_64-1.0.0-1.el7.centos already installed and latest version
Nothing to do

[1/3] Installing php-fpm.x86_64-1.0.0-1.el7.centos: start php-fpm
[2/3] Installing php-fpm.x86_64-1.0.0-1.el7.centos: install php-fpm
Created symlink from /usr/lib/systemd/systemd/user-runtime-dir/php-fpm.service to /usr/lib/systemd/systemd/php-fpm.service.
[3/3] Installing php-fpm.x86_64-1.0.0-1.el7.centos: status php-fpm
* php-fpm.service - The PHP FastCGI Process Manager
   loaded: loaded (/usr/lib/systemd/systemd/php-fpm.service; enabled; vendor preset: disabled)
   active: active (running) since Fri 2015-04-04 01:07:34 EDT; 10s ago
 Main PID: 4566 (php-fpm)
 Status: "ProcessManager: active, 4, listen: *, Requestor: 4, class: 4, traffic: 0req/sec"
 CGroup: /systemd/systemd/php-fpm.service
          └─ php-fpm: master process (/usr/lib/php.conf)
                php-fpm: pool www
                php-fpm: pool www
                php-fpm: pool www
                php-fpm: pool www
                php-fpm: pool www

php composer-setup.php
```

Next, now that the setup is complete, delete the unnecessary files by executing `php -r "unlink('composer-setup.php');"`.


```
sudo systemctl start nginx
```

Next, change the configuration to autostart even if the server is restarted, enter `sudo systemctl enable nginx`, and change the configuration to autostart.

```
sudo systemctl enable nginx
```

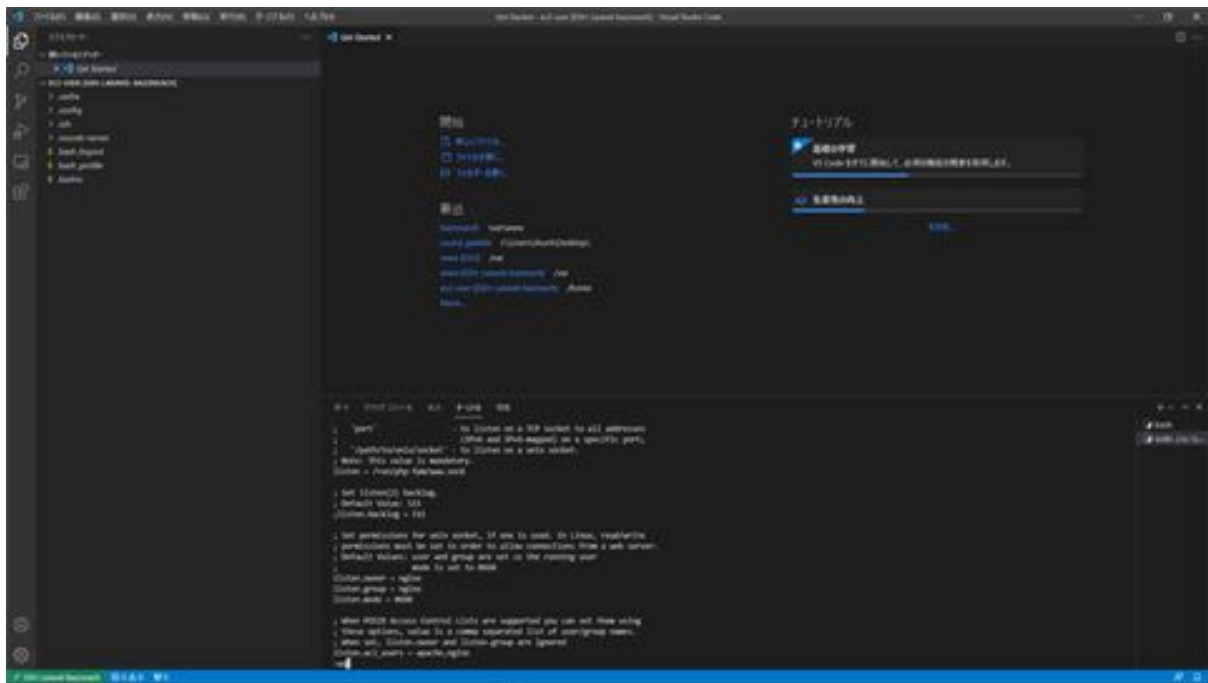


```
root@kali:~# cd /etc/php-fpm.d/
```

Then run `sudo cp www.conf www.conf_bk_20220301` to back up the file.

```
root@kali:~# sudo cp www.conf www.conf_bk_YYYYMMDD
```

Next, open the configuration file and edit its contents by typing `sudo vim www.conf`.



Change the following in the opened file

```
### Per line 24 ###
```

```
# Before change
```

```
user = apache
```

```
# After change
```

```
user = nginx
```

```
### Per line 26 ###
```

```
# Before change
```

```
group = apache
```

```
# After change
```

```
group = nginx
```

```
### Per line 48 ###
```

```
# Before change
```

```
;listen.owner = nobody
```

```
# After change
```

```
listen.owner = nginx
```

```
### Per line 49 ###
```

```
# Before change
```

```
;listen.group = nobody
```

```
# After change
```

```
listen.group = nginx
```

```
### Per line 50 ###
```

```
# Before change
```

```
;listen.mode = 0660
```

```
# After change
```

```
listen.mode = 0660
```


Correction details

```
server {
    listen    80;
    listen    [::]:80;
    server_name _;
    root      /var/www/bazzreach/public;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-Content-Type-Options "nosniff";

    index index.php;

    charset utf-8;

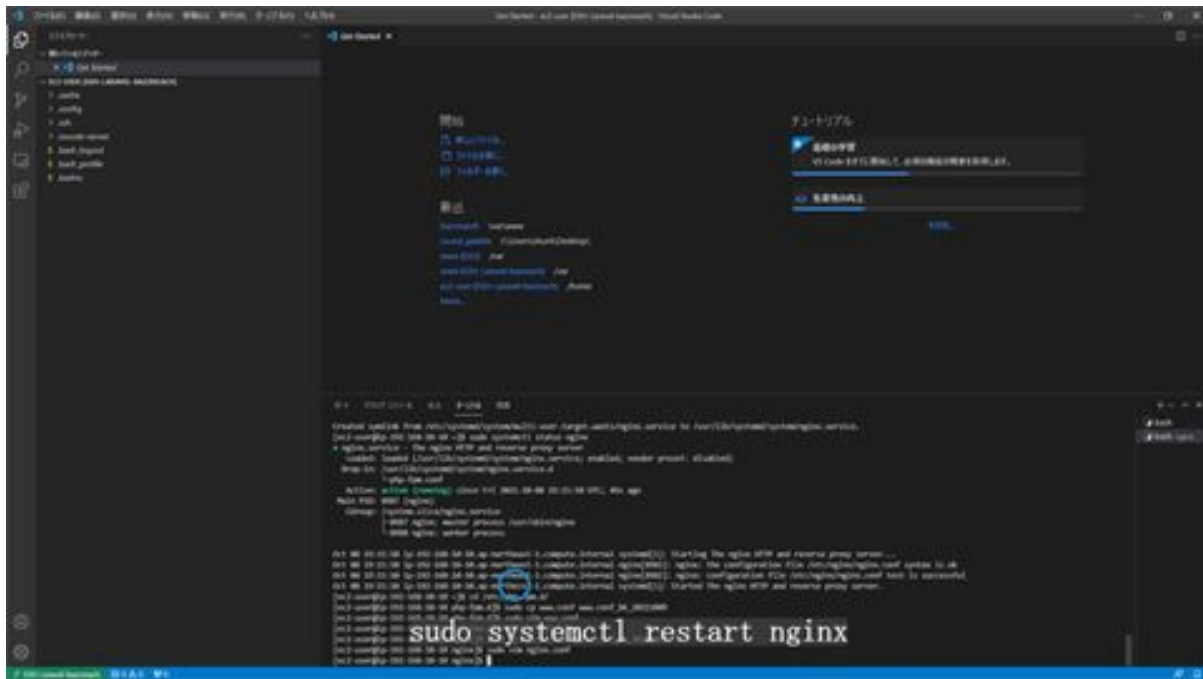
    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ /\.php$ {
        fastcgi_pass  unix:/run/php-fpm/www.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME
    $document_root$fastcgi_script_name;
        include      fastcgi_params;
    }

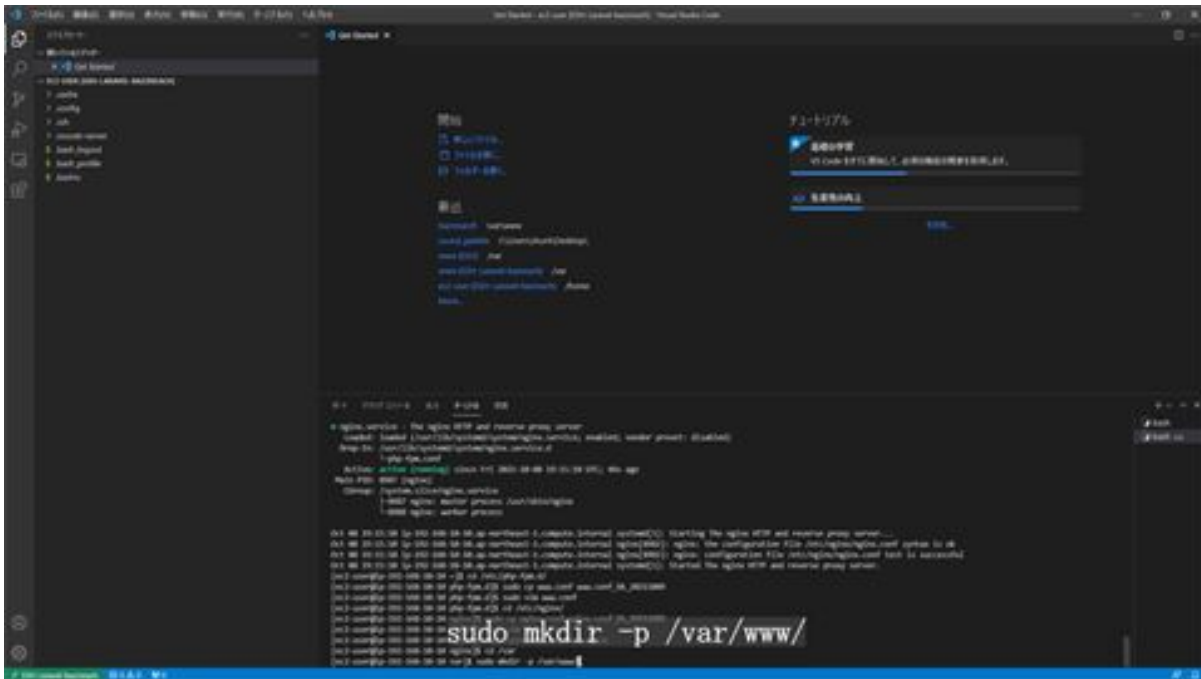
    location ~ /\.(!well-known).* {
        deny all;
    }

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
}
```

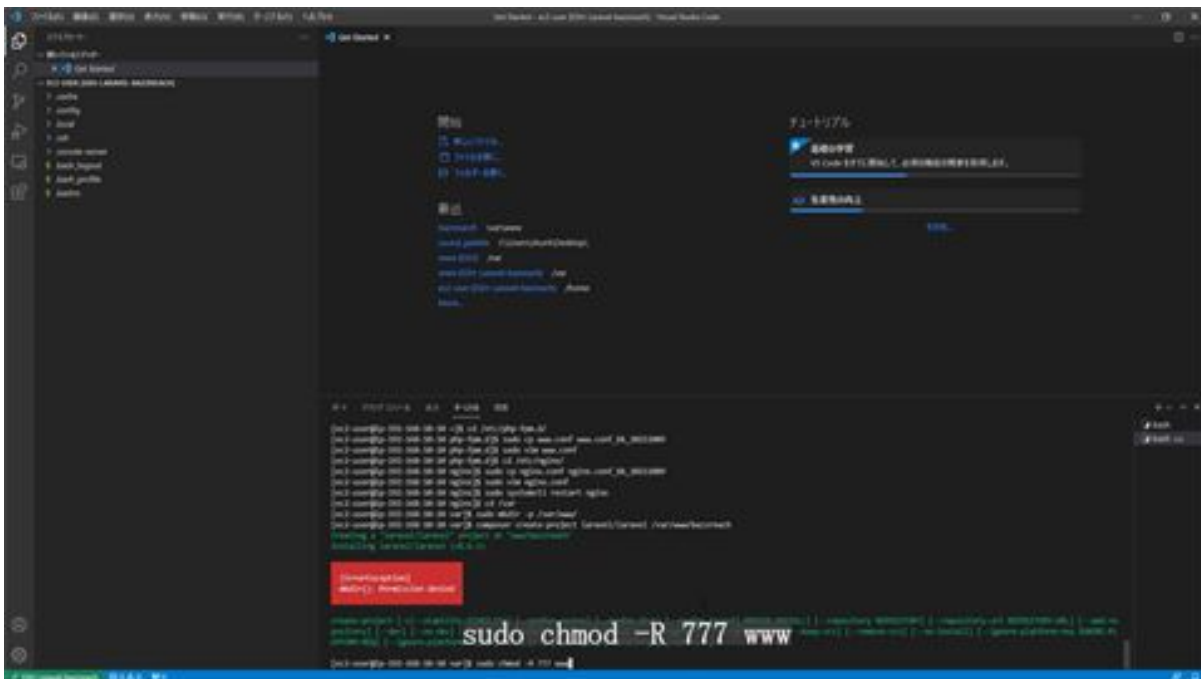
Next, restart Nginx now that the configuration has been changed by running `sudo systemctl restart nginx`.



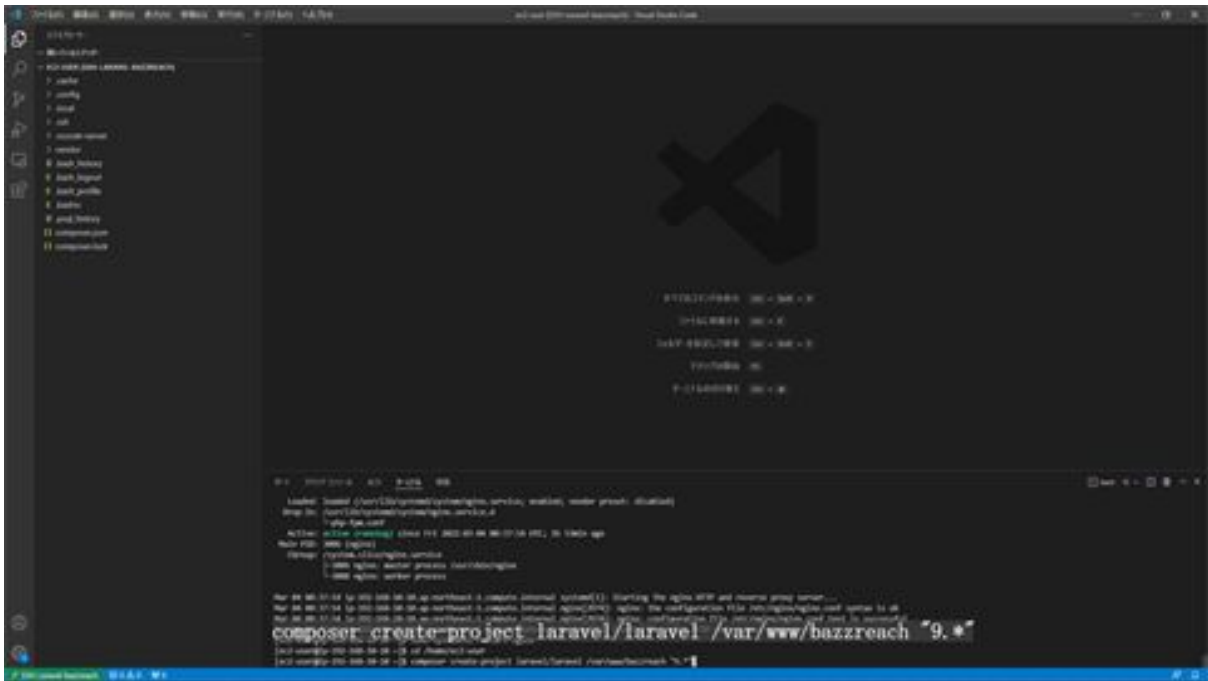
Next, create a document root. Run `cd /var` to move the var folder. Then inside this folder, create a folder called `www`. Run `sudo mkdir -p /var/www/` to create the `www` folder.



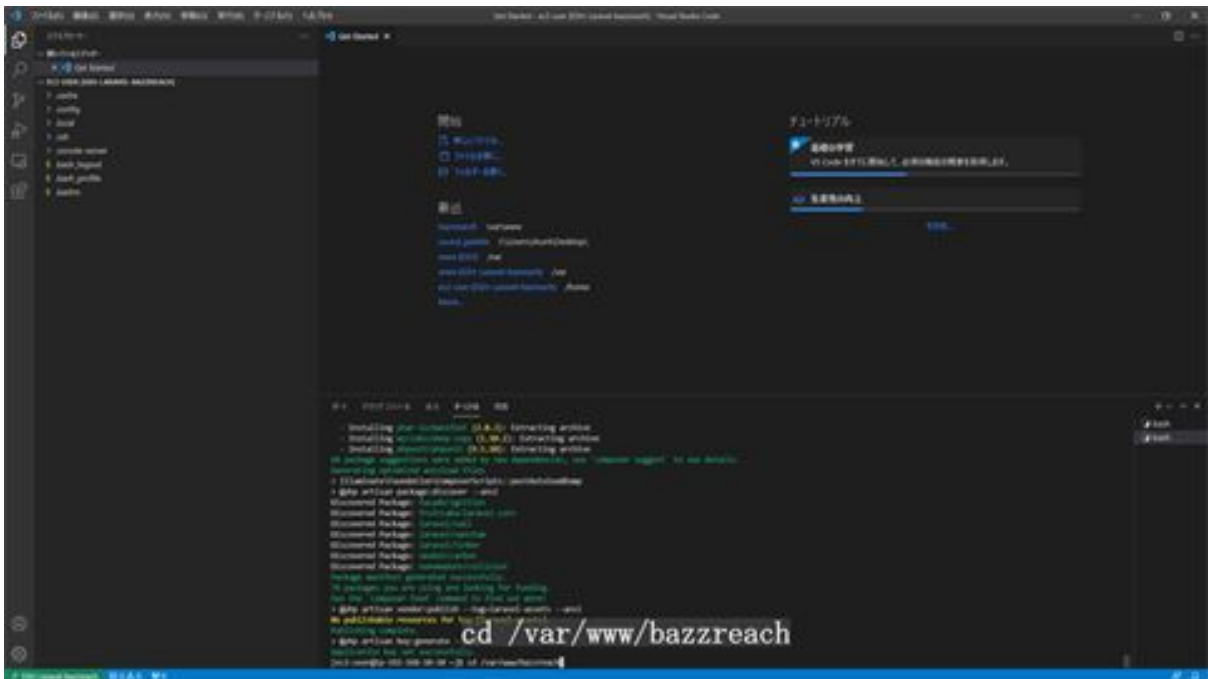
Next, change the permission settings by running `sudo chmod -R 777 www`.



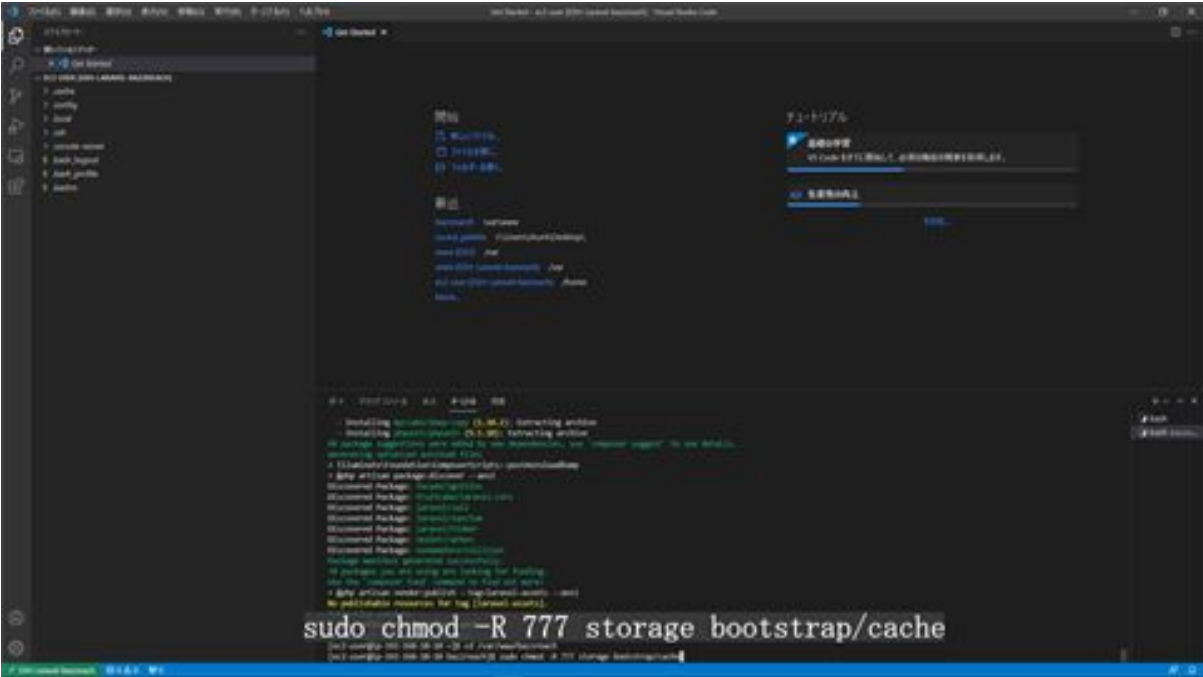
Next, go to the `ec2-user` folder and install the Laravel application by running `cd /home/ec2-user` and then `composer create-project laravel/laravel`. Run `/var/www/bazzreach "9.*"` to install Laravel9.



Once the installation of the Laravel application is complete, the next step is to change the permission settings for the Laravel application. First, navigate to the application folder: `cd /var/www/bazzreach` and navigate to the bazzreach folder.



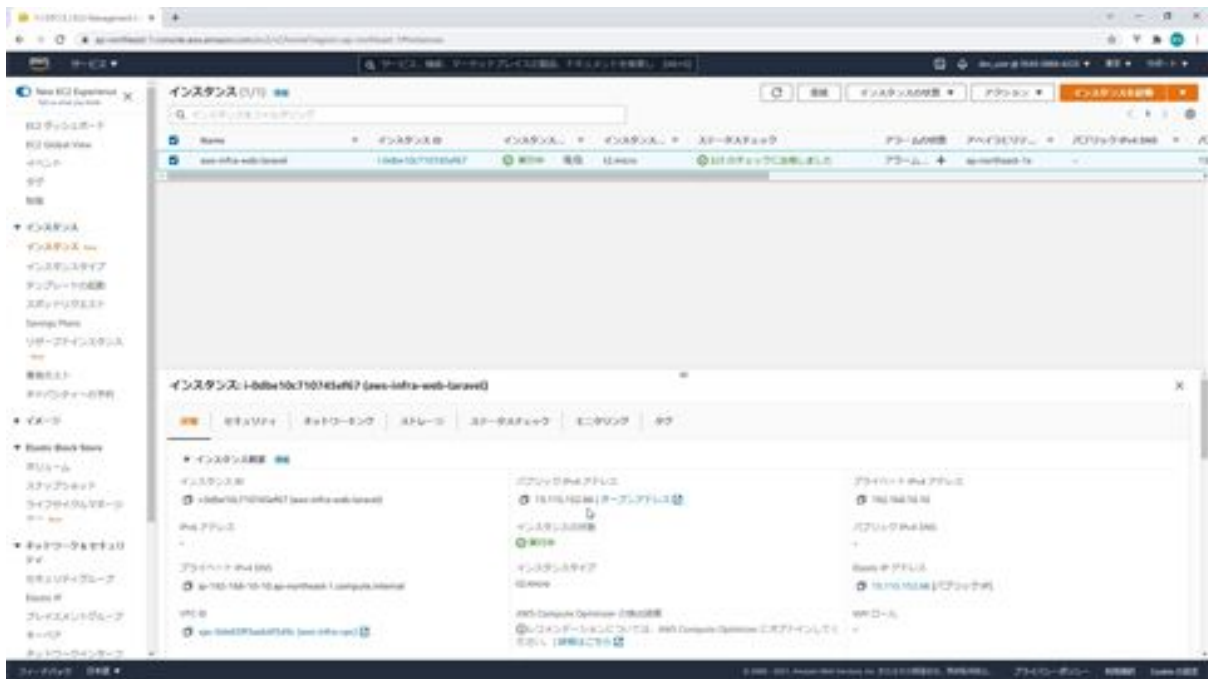
Next, run `sudo chmod -R 777 storage bootstrap/cache` to change the permission settings for storage and cache.



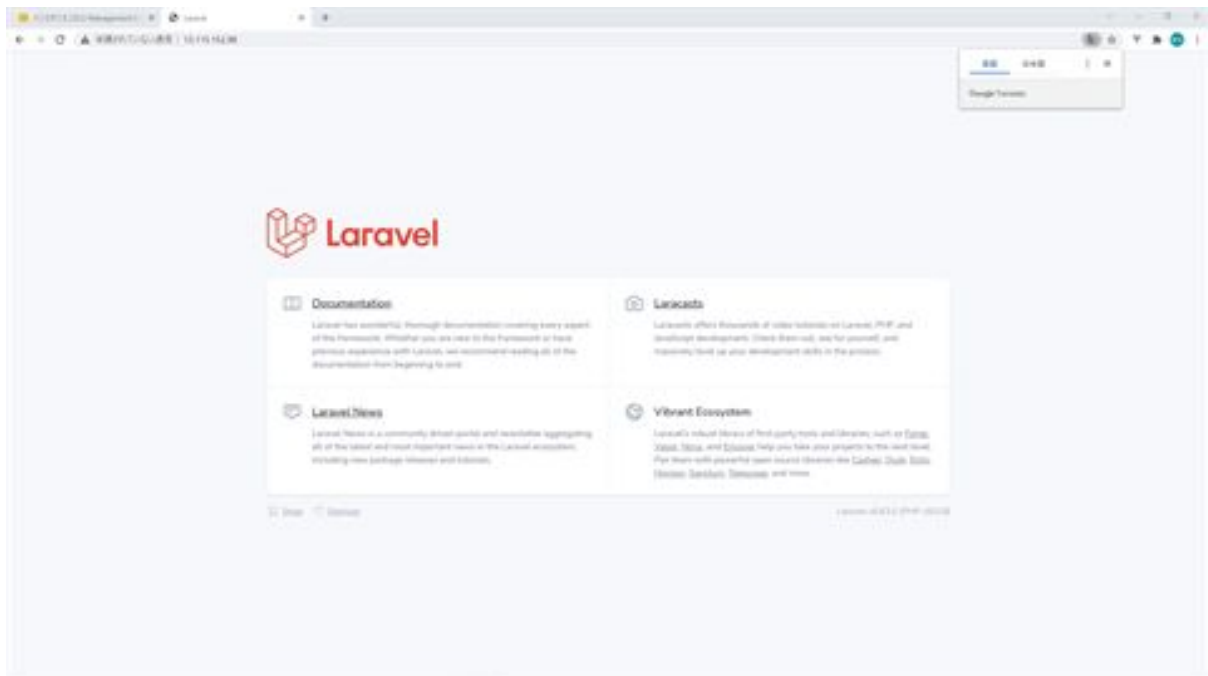
This completes the installation of Laravel.

Let's see if Laravel is displayed on the screen

Now we would like to check if Laravel appears on the screen. Copy the public IPV4 address of the instance aws-infra-web-Laravel, type it in the URL of your browser, and press Enter.



The Laravel screen is then displayed. This completes the Laravel screen verification.



Documentation

Laravel has excellent, thorough documentation covering every aspect of the Framework. Whether you are new to the Framework or have previous experience with Laravel, we recommend reading up on the documentation from beginning to end.



Laravel News

Laravel News is a community driven guide and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new packages, releases and releases.



Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP and Backend Development. Check them out, and for yourself, and improve/boost up your development skills in the process.



Vibrant Ecosystem

Laravel's robust library of third-party tools and libraries, such as Laravel Scout, Laravel Horizon, and Laravel Spark, help you take your projects to the next level. For more with your for open source libraries like Laravel Scout, Laravel Horizon, Laravel Spark, Laravel Nova, and more.

Chapter 5: Convert your website to HTTPS with AWS

Get your domain name for free

In this chapter, we will deal with SSL in AWS. The first step is to acquire a domain name. There are many sites for getting a domain name, but since we want to get a domain name for free, we will use a site called Freenom to get a domain name for free. First, search for Freenom on google. This will bring up the Freenom website, which you can access [here](https://www.freenom.com).



First of all, select Japanese from the language notation. Next, the domain search screen will appear, where you can search for the domain of your

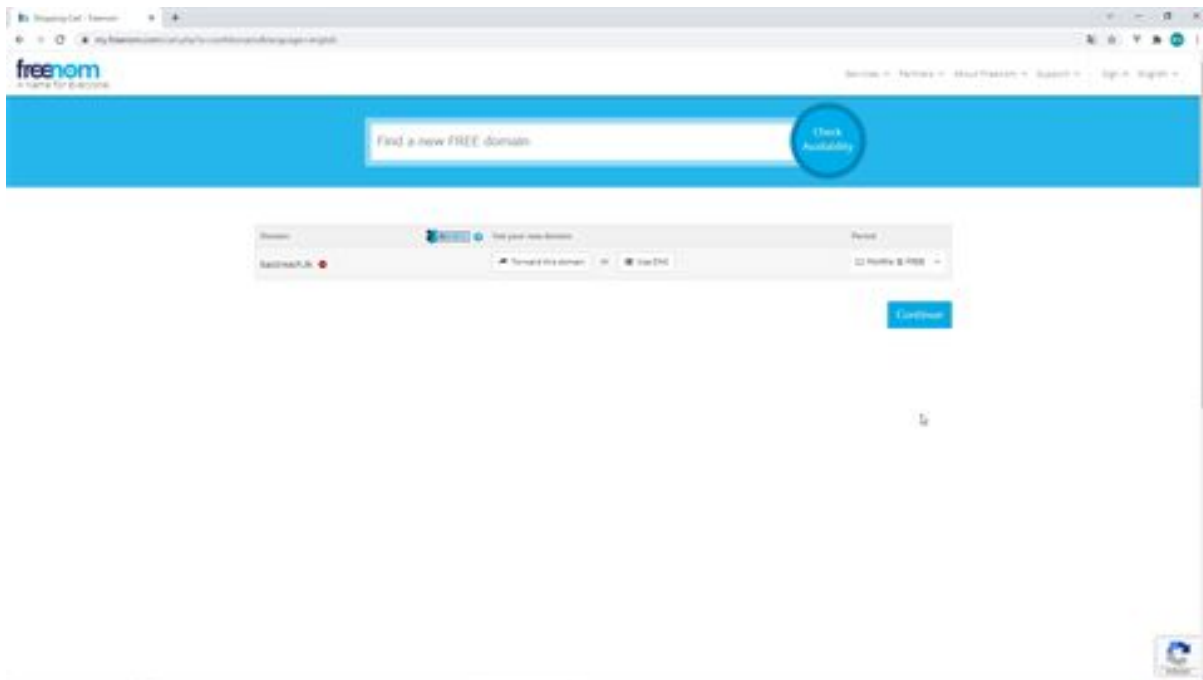
application URL. You can search for any domain you like. In our case, we will search for bazzreach .



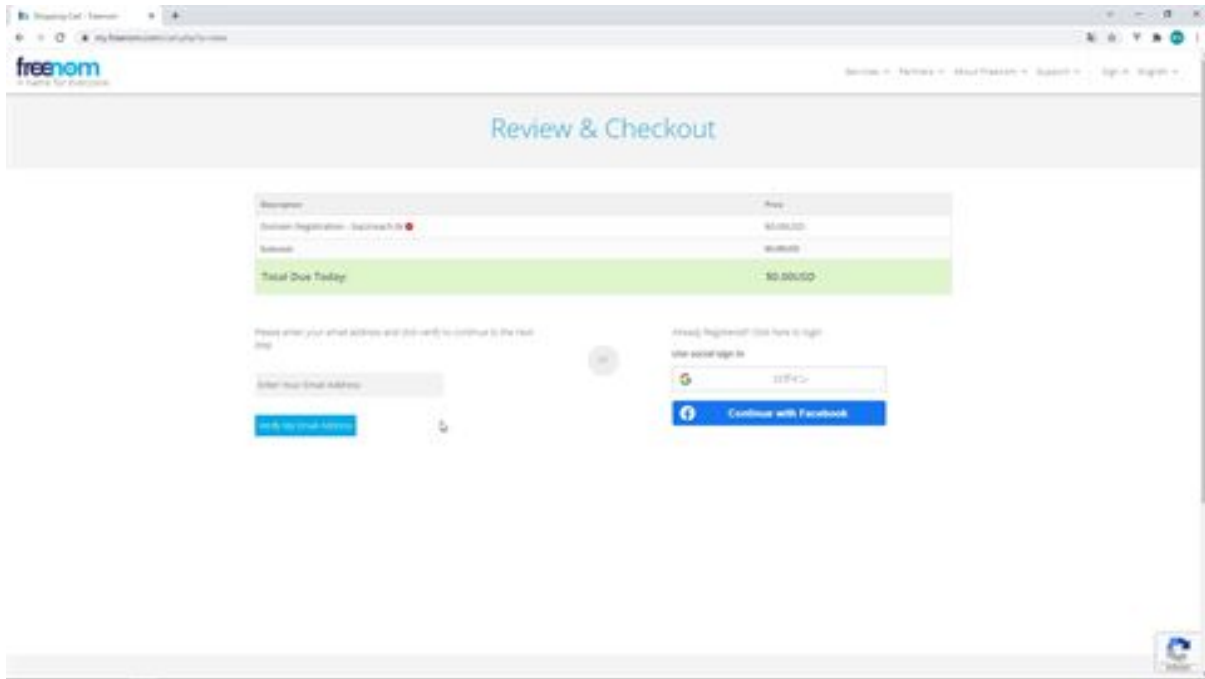
Then you will see some free domains, click on the Get Free Domain Now button. If you get an unavailability message here, search again for something like bazzreach.tk. This time we will proceed to get bazzreach.tk.



Select a domain and click on the Checkout button. The cart screen will appear, and under Period, select the registration period for the domain and click on the Continue button.



Next, the Review & Checkout screen will appear, where you can verify your email address. There will be an email address field, so enter your email address there and click the Verify My Email Address button. You will receive a verification email. Click on the link in the email to verify your email address.

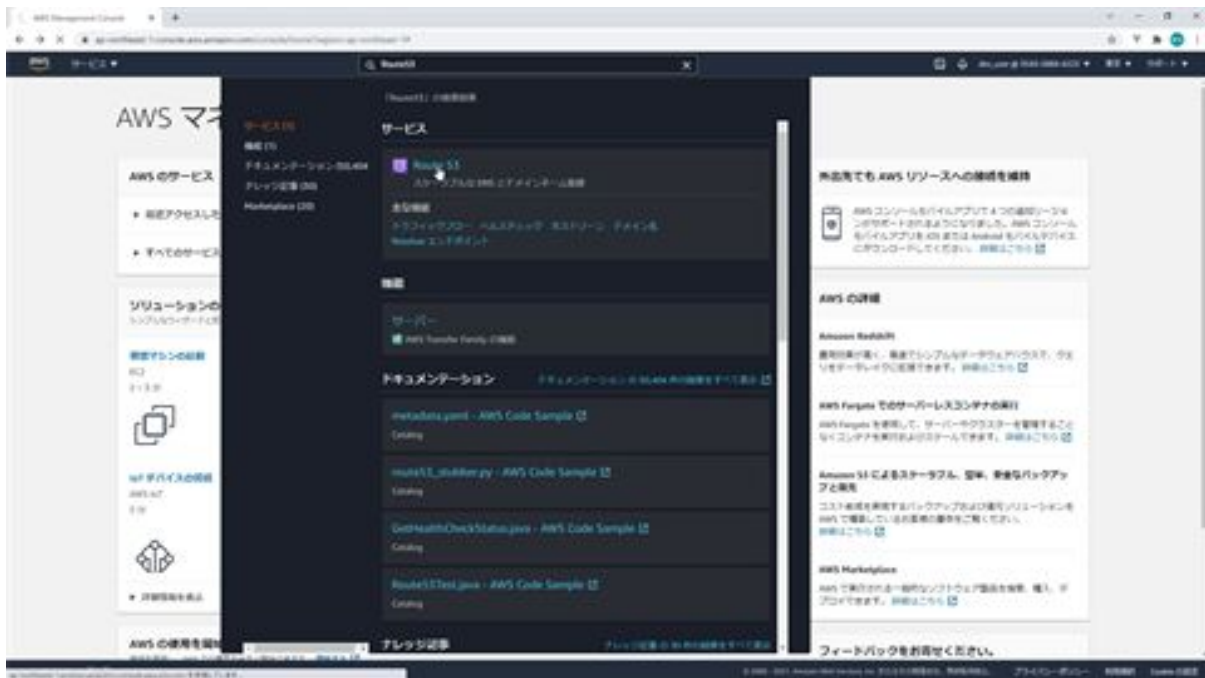


After doing so, you will be prompted to enter your personal information. Check the Terms of Use, and click the Complete Order button.

Next, go to the My Domains screen to check if you have the domain you acquired. Click on My Domains from the Services menu. You can now see your domain name on the My Domains screen. You have now completed the domain acquisition.

Let's create a hosted zone

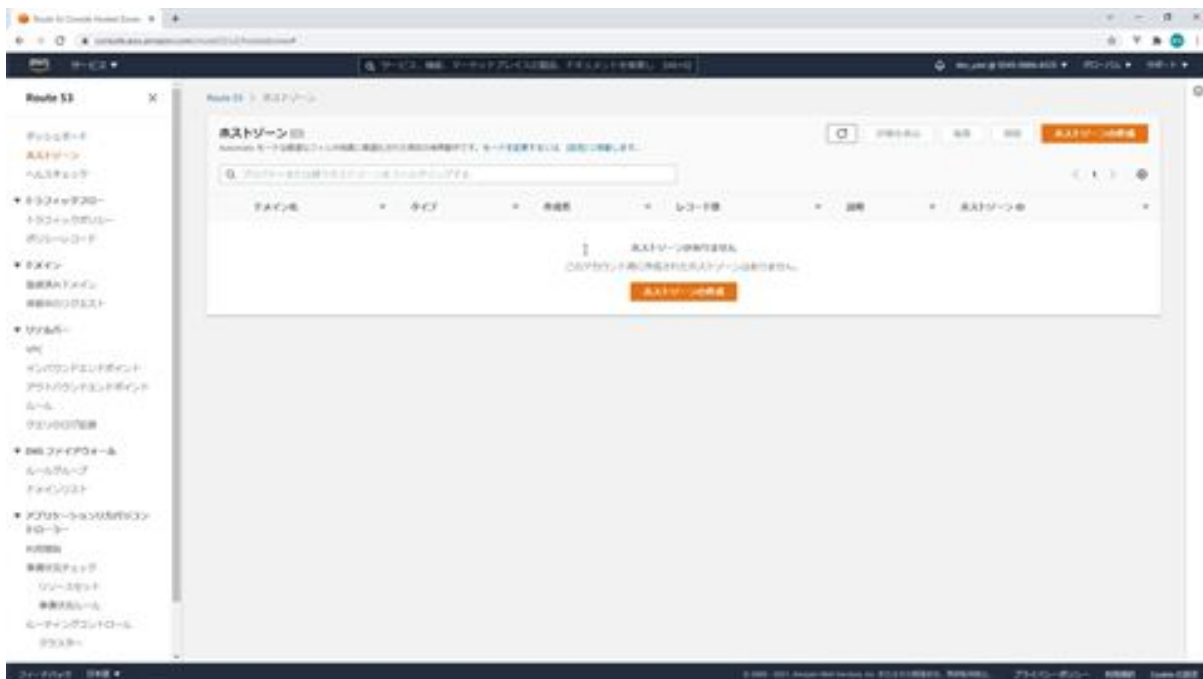
Now we will create a hosted zone for Route53. A hosted zone holds information about how to route traffic for a particular domain, such as example.com or its subdomain sub.example.com. This host zone is a necessary setting to access pages on your domain. Now, let's do some actual work. First, open the Route53 dashboard.



Next, click on Host Zone in the left menu to display the Host Zone screen.

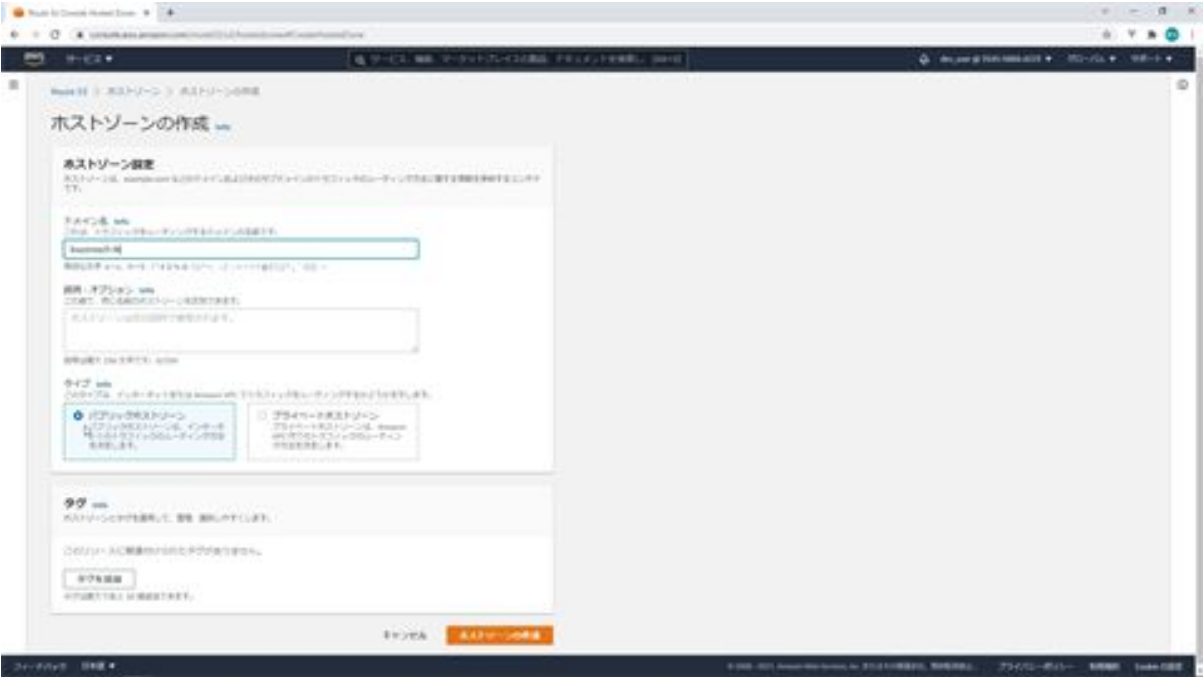


Next, click on the Create Host Zone button and enter the information required to create a hosted zone.

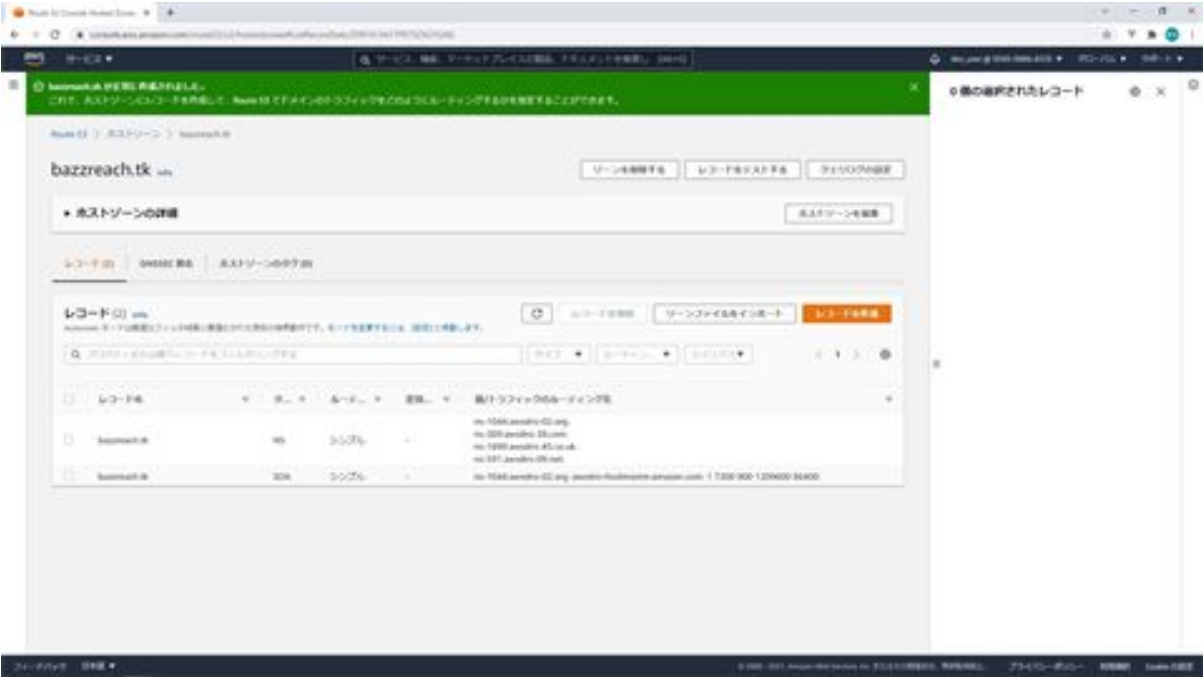


For the domain name, set a domain name that you have acquired yourself. In this case, we will enter bazzreach.tk. You do not need to enter any

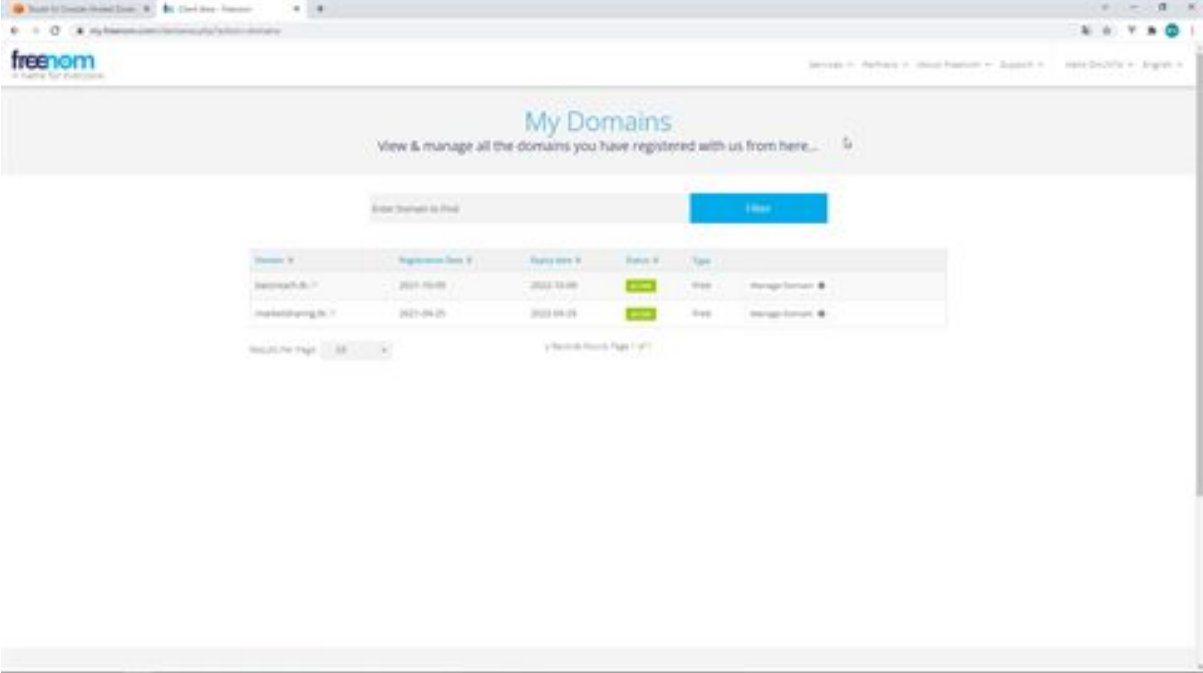
description. For Type, select Public Host Zone. After making your selection, click the Create Host Zone button.



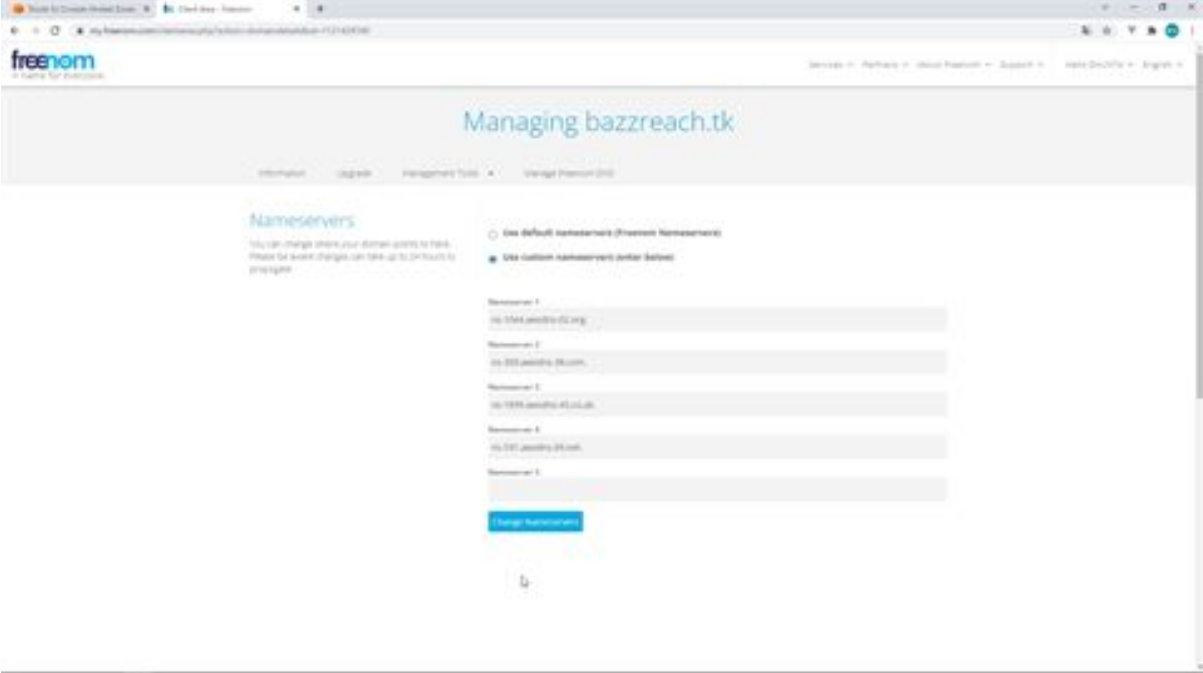
This completes the creation of the hosted zone. When you create a hosted zone, NS records and SOA records are registered from the beginning.



The next step is to set this NS record to the FreeNom nameservers. Click My Domains from the menu and click Manage Domain for the domain you have registered.



Click on the Management Tools tab and click on Nameservers. Now select Custom and enter all of your NS records here. Finally, click on the Change Nameservers button.



The name server settings have now been changed. That's all for creating a hosted zone.

freemom
Managing bazzreach.tk

Information Login Management Tools Manage Premium DNS

Change DNS Records

Information

At the right you can find the details of your domain. You can manage your domain using the tabs above.

[Back to Domains list](#)

Domain

Domain: bazzreach.tk

Registration Date: 20/10/2021

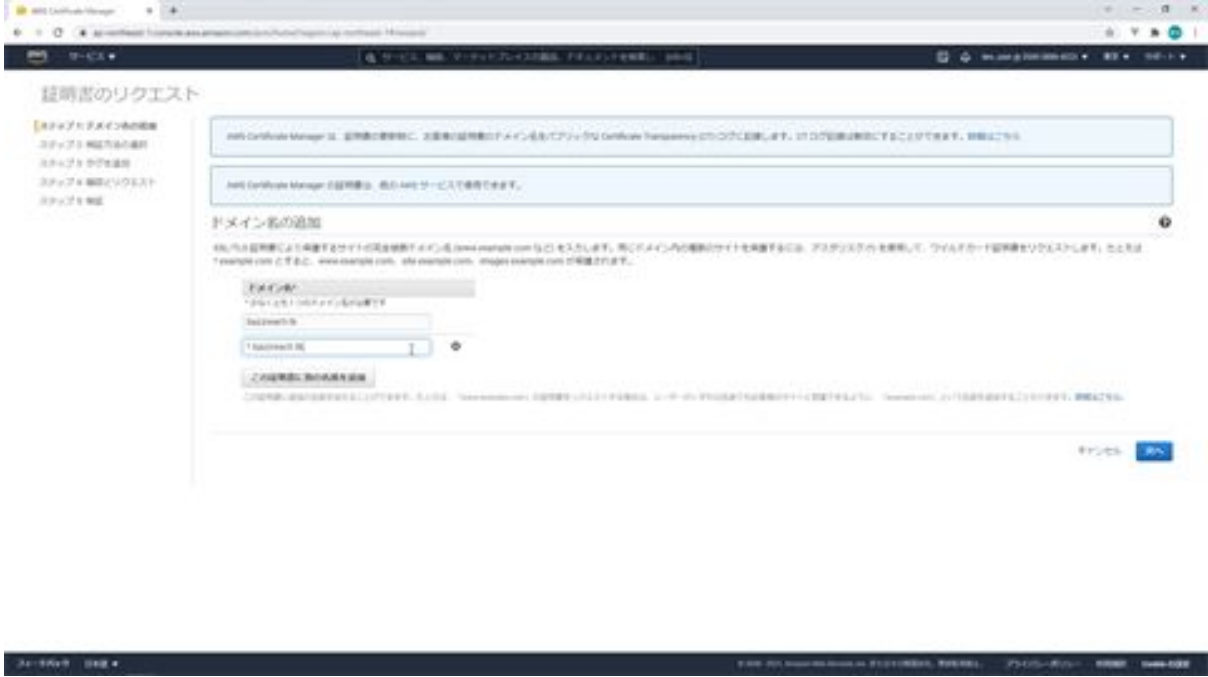
Expiry date:



When the Request Certificate screen appears, click the Request Certificate button.



Next, enter your domain name on the Add Domain page. Please enter your domain here. In my case, I'll enter bazzreach.tk and *.bazzreach.tk. Once you have entered the information, click on the Next button.



In the next verification method selection screen, select DNS verification and click the Next button.

証明書のリクエスト

- ステップドメインの追加
- ステップドメインの削除
- ステップドメインの追加
- ステップドメインのリクエスト
- ステップドメインの管理

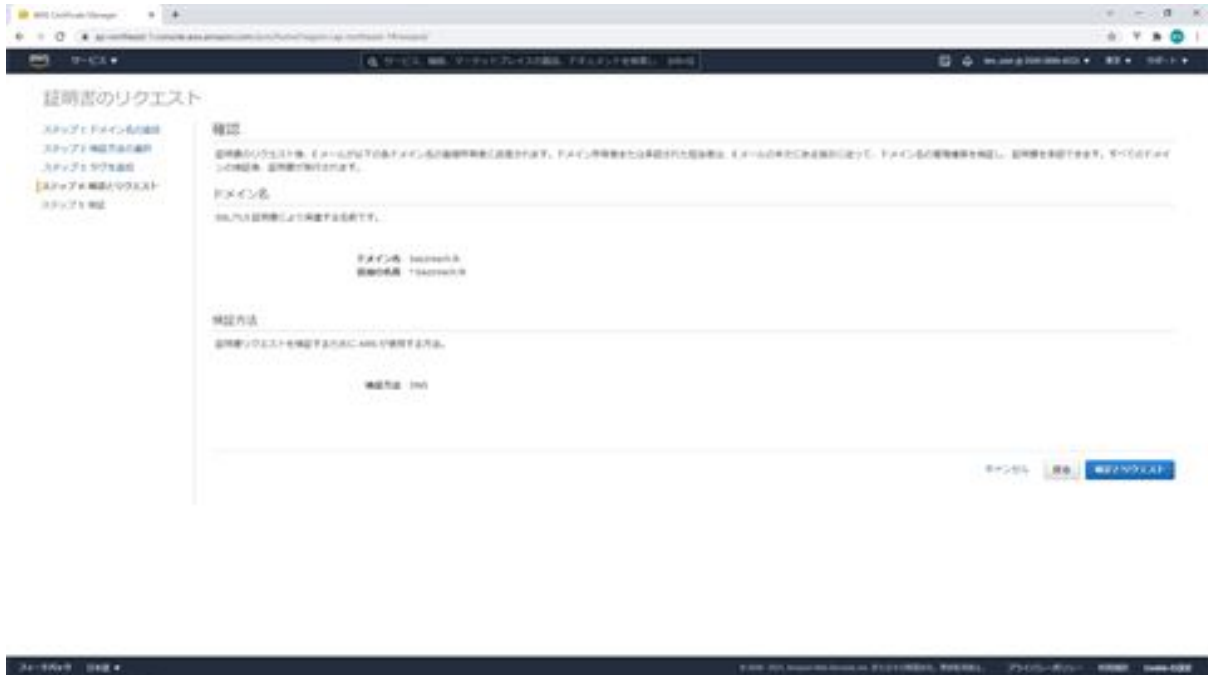
検証方法の選択

Azure Certificate Manager (ACM) による証明書リクエストの検証方法を選択します。または、証明書を作成する前に、承認されたドメインにドメイン名を登録してドメイン名を所有していることを確認してください。ACM は、ACM を使用するドメインに所有権の検証がドメインに登録されていることを確認することにより、所有権を検証することができます。

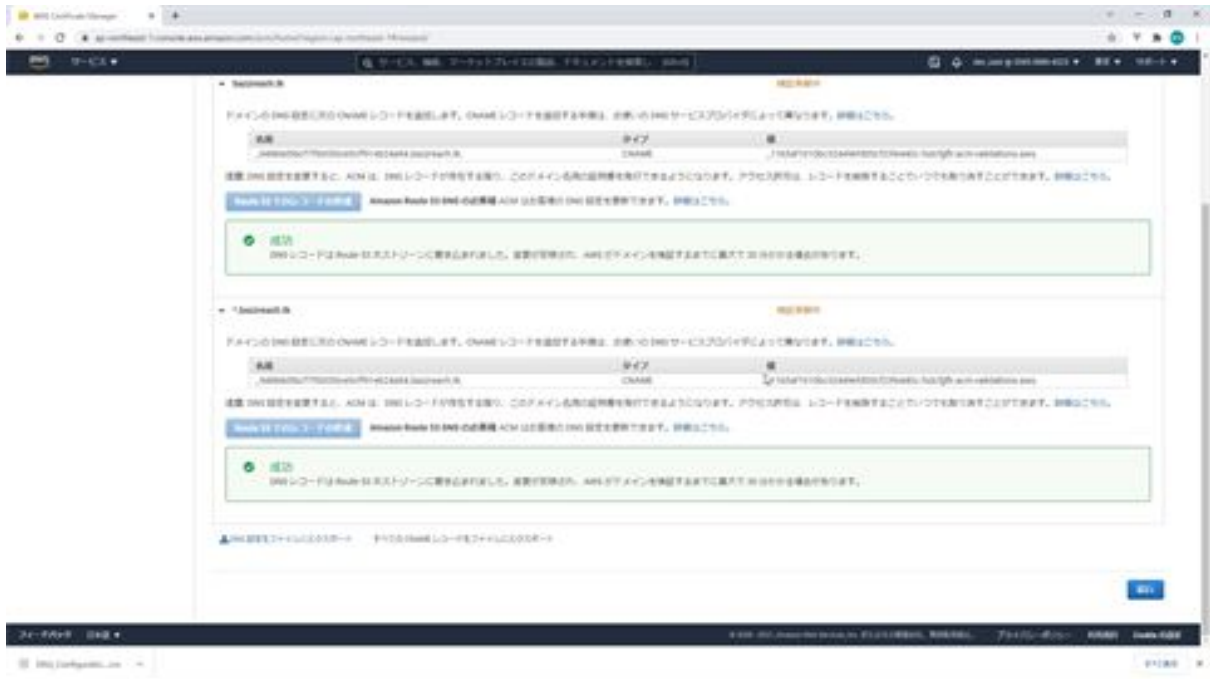
- ドメインの検証**
証明書リクエストドメインの ACM 検証を使用するドメインに所有権が所有されている。または所有する場合は、このオプションを選択します。詳細はこちら。
- ドメインの検証**
証明書リクエストドメインの ACM 検証を使用するドメインに所有権が所有されていない。または所有する場合は、このオプションを選択します。詳細はこちら。

キャンセル 戻る 完了

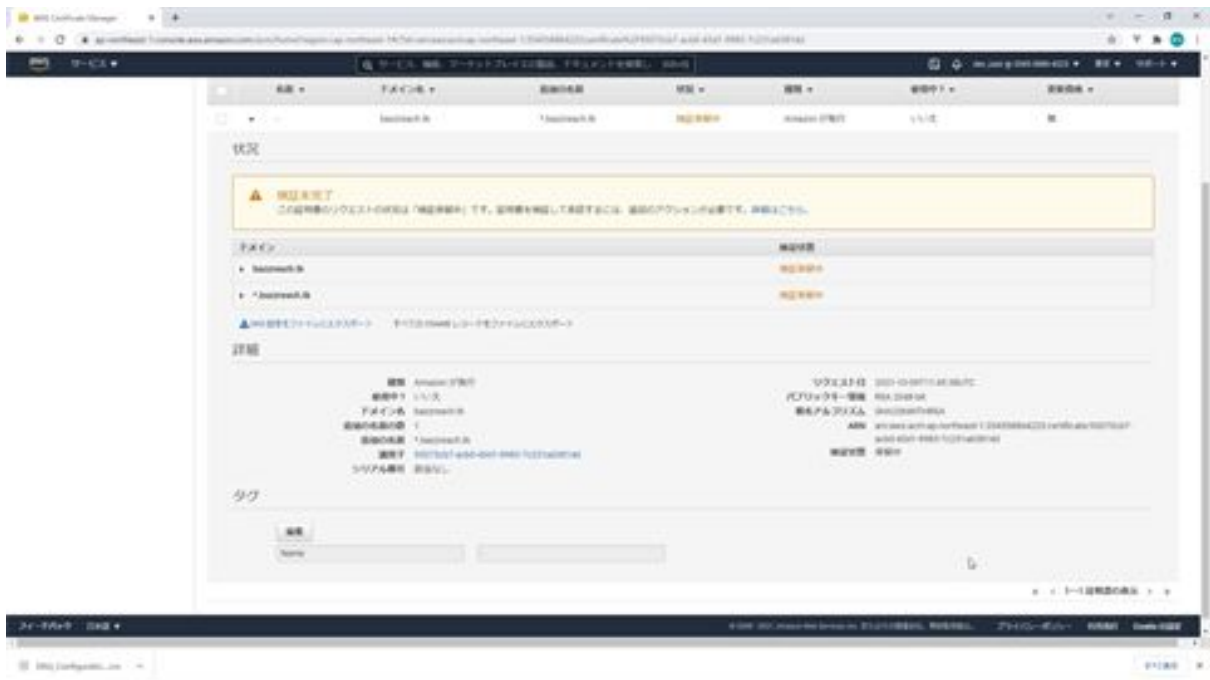
In the next Add Tag screen, click the Confirm button without entering anything. When the confirmation screen appears, check the domain name and verification method, and if everything is OK, click the Confirm and Request button.



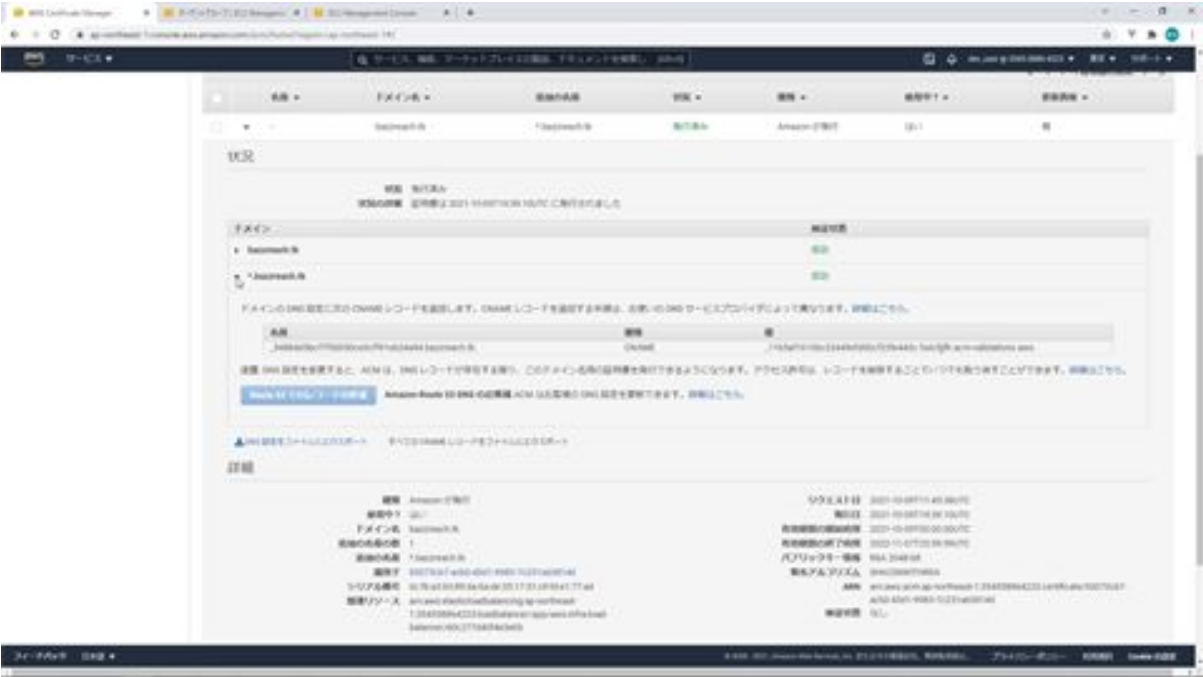
Click on the black triangle in front of the domain in the next validation screen to display additional information about the CNAME record. Then, click the Create Record in Route 53 button at the bottom. The CNAME record will then be created in Route 53.



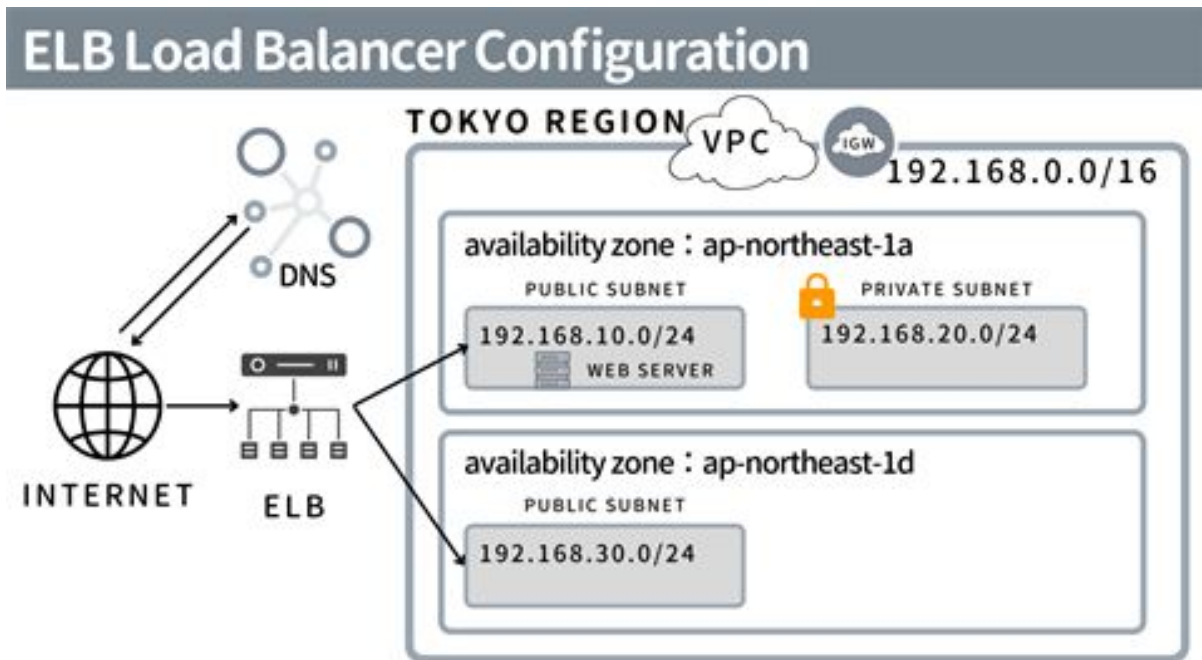
Next, click the Continue button to return to the certificate screen, where the status is pending verification. If you wait for a few hours, the status will be issued, please wait for a while.



The status has been issued. This completes the issuance of your free SSL certificate.



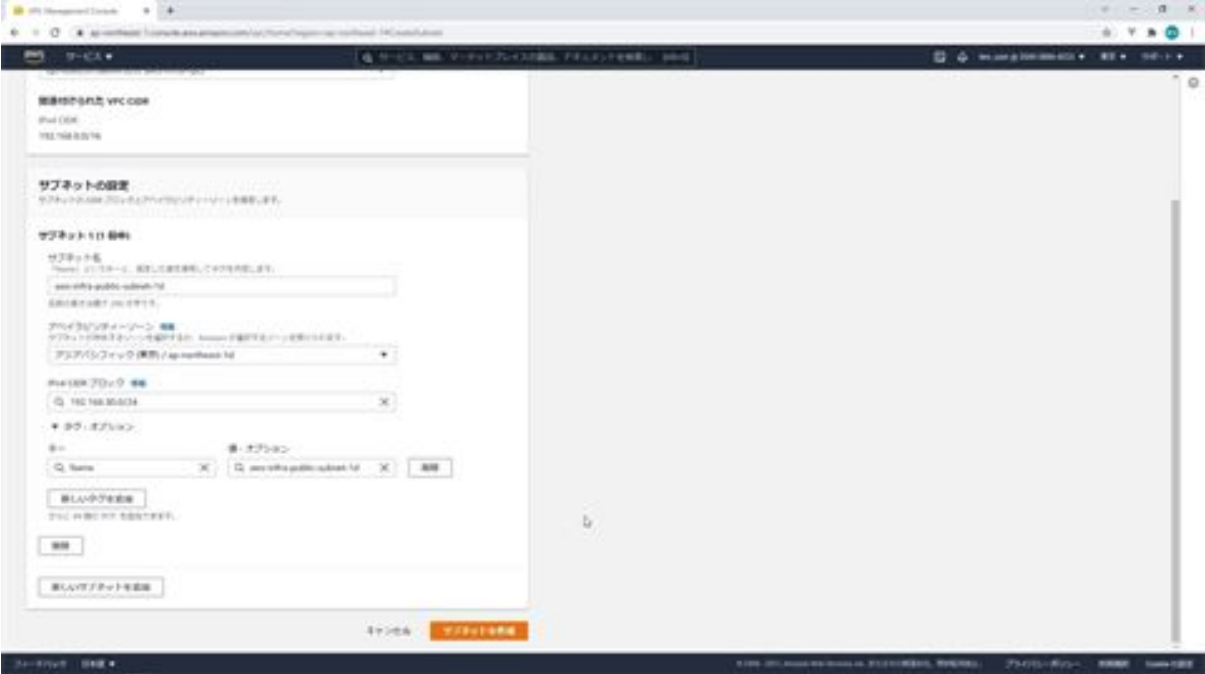
Let's set up a load balancer with AWS ELB



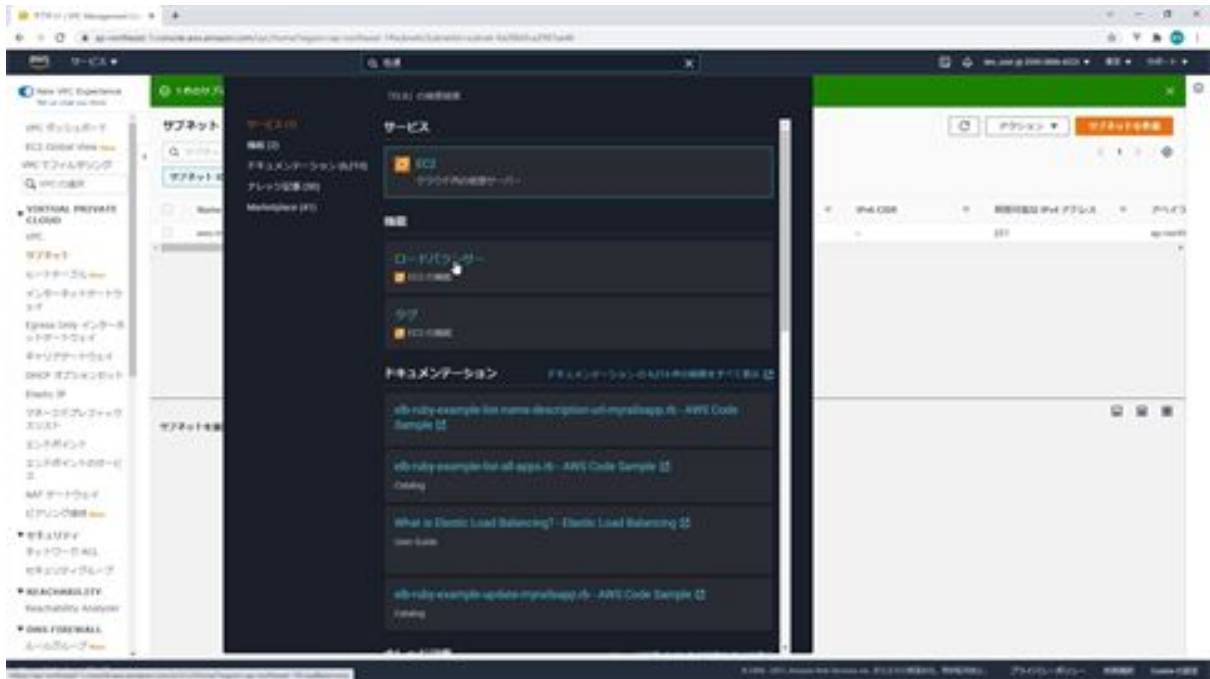
Next, I would like to set up a load balancer, but before that, I will explain about load balancers. A load balancer is called a load balancer, and it is a device that provides a mechanism to distribute communication from the outside to multiple servers.

In this case, we want to set up an SSL certificate, and by building an AWS ELB and tying it to an EC2 instance as a target, we can use it to do the SSL certificate settings for us. We will now create another subnet for the load balancer.

From the AWS Management Console, search for VPC and click Subnet in the menu on the left. Then click the Create Subnet button. On the subnet creation screen, select aws-infra-vpc as the VPCID and enter aws-infra-public-subnet-1d as the subnet name. Select 1d as the availability zone, enter 192.168.30.0/24 as the CIDR block and click the Create Subnet button. A new subnet has now been created.



The next step is to create a load balancer. In the search console above, search for ELB. Then you will see "load balancer" and click on it.

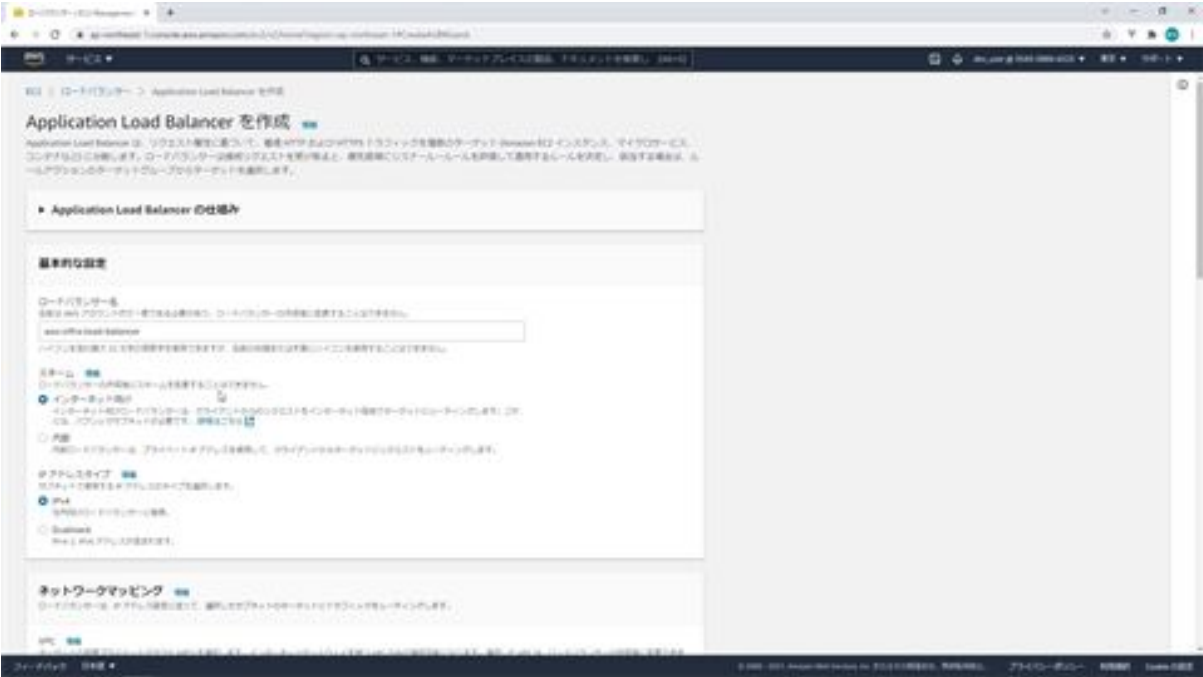


Next, on the Load Balancer screen, click the Create Load Balancer button. Now select the type of load balancer, here click on the Create ALB button.

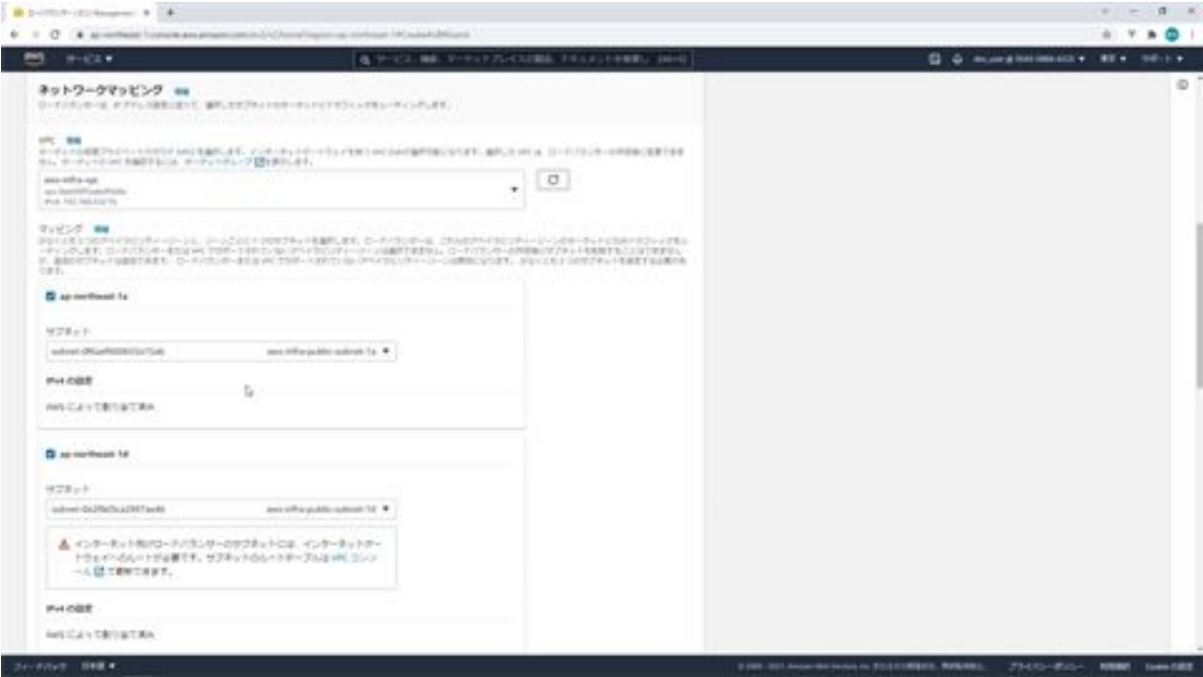


The screen for creating a load balancer will appear, and you can enter the information here. First, enter the load balancer name as aws-infra-load-

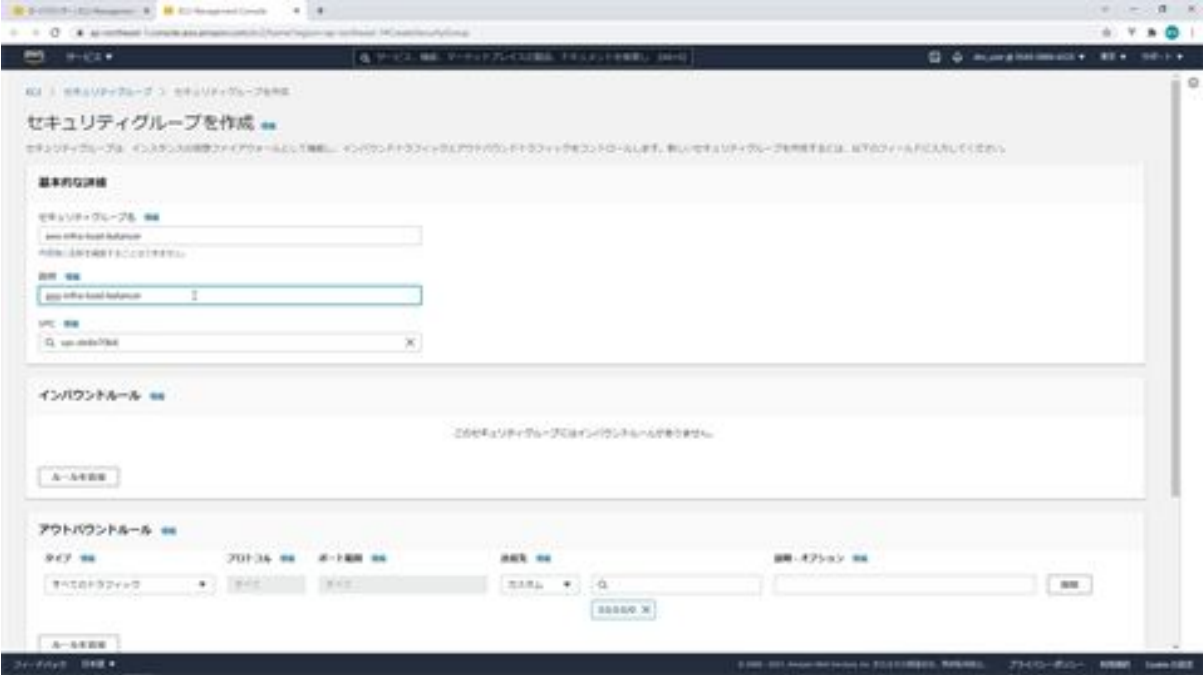
balancer. Select the scheme for the Internet and select IPv4 as the IP address type.



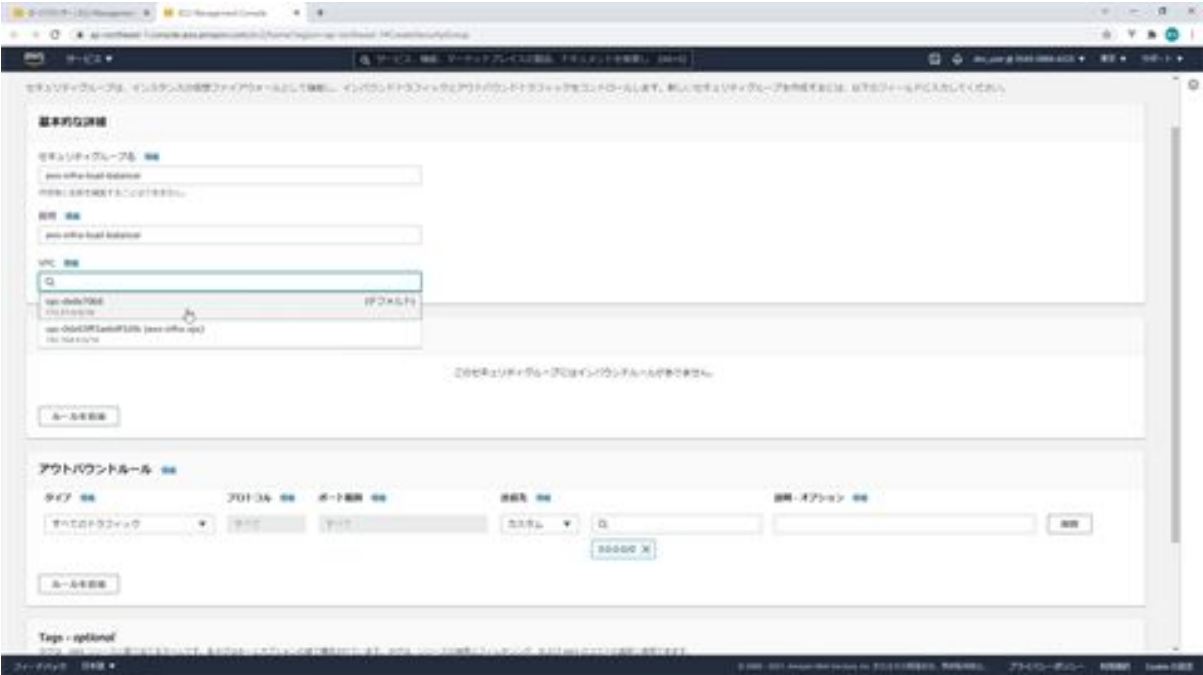
For VPC, select aws-infra-vpc. Once selected, you will see 1a and 1d in the availability zone, check these two and select a public subnet for each.



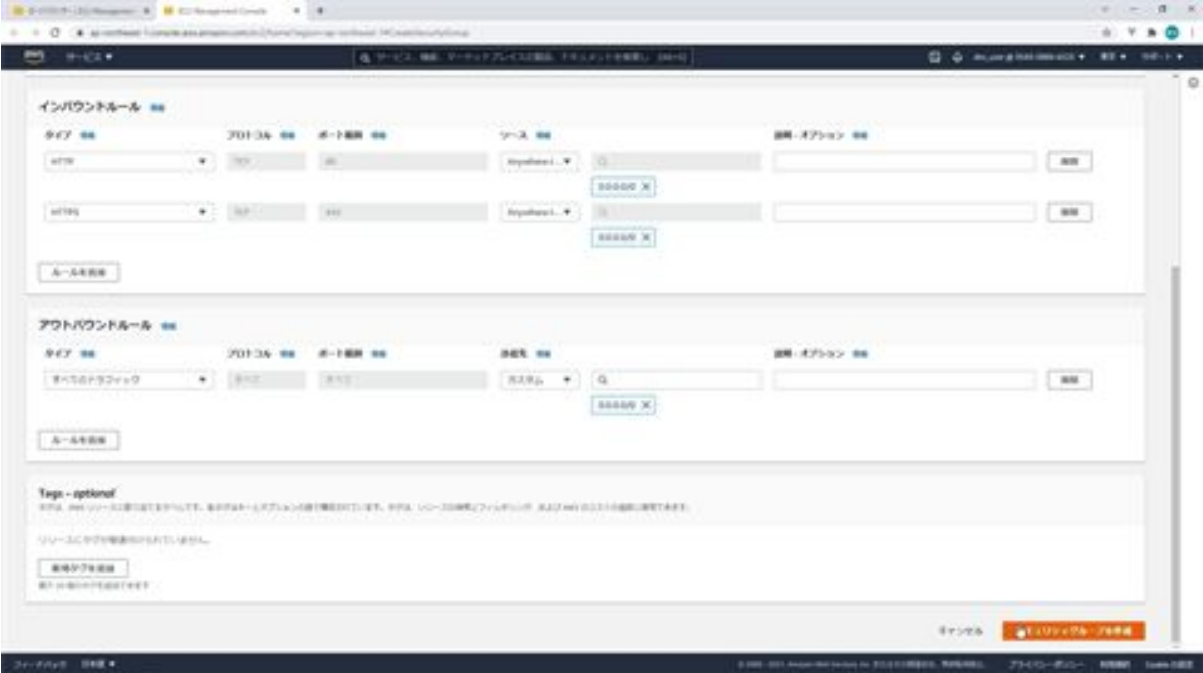
Next, under Security Groups, click the Create new security group link to create a new security group. Enter aws-infra-load-balancer as the security group name and enter the same in the description section.



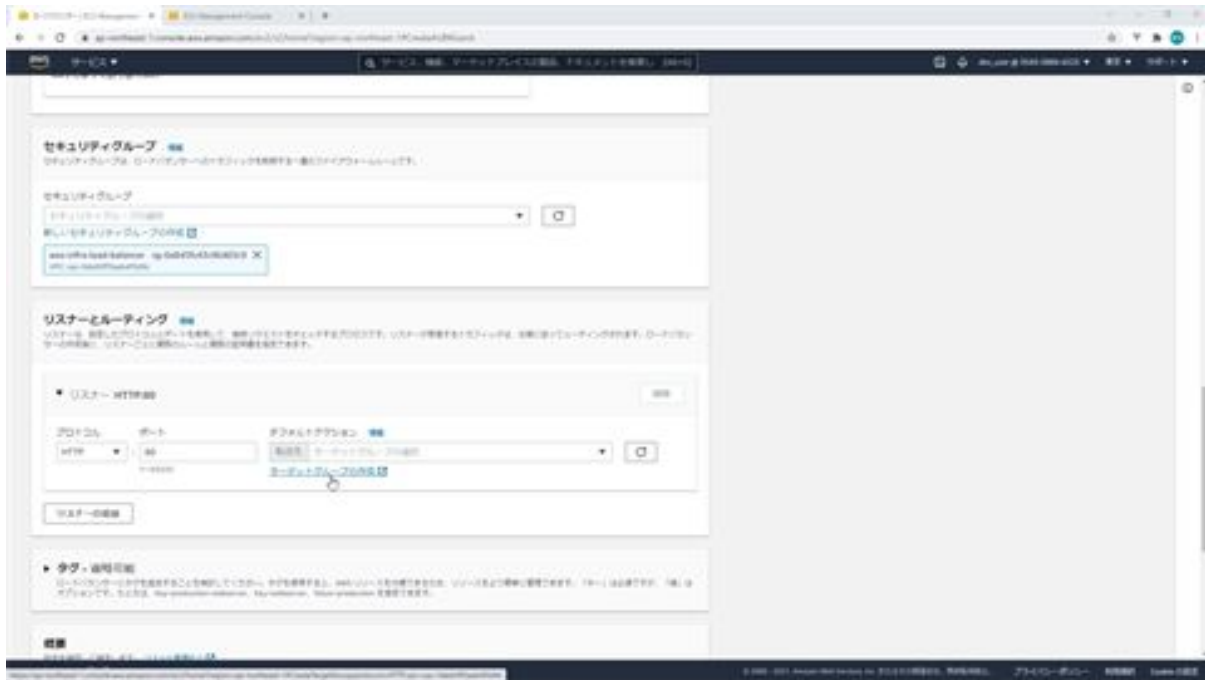
For VPC, select aws-infra-vpc.



Next, click the Add Rule button in Inbound Rules. This will allow you to enter the rules, and you can add HTTP and HTTPS here. Next, enter the CIDR block here, and enter 0.0.0.0/0 here. After doing so, click the Create Security Group button.



Now that you have created a new security group, go back to the load balancer screen, click the refresh button on the right side of this screen, and select the newly created aws-infra-load-balancer. Also, delete the default security group.



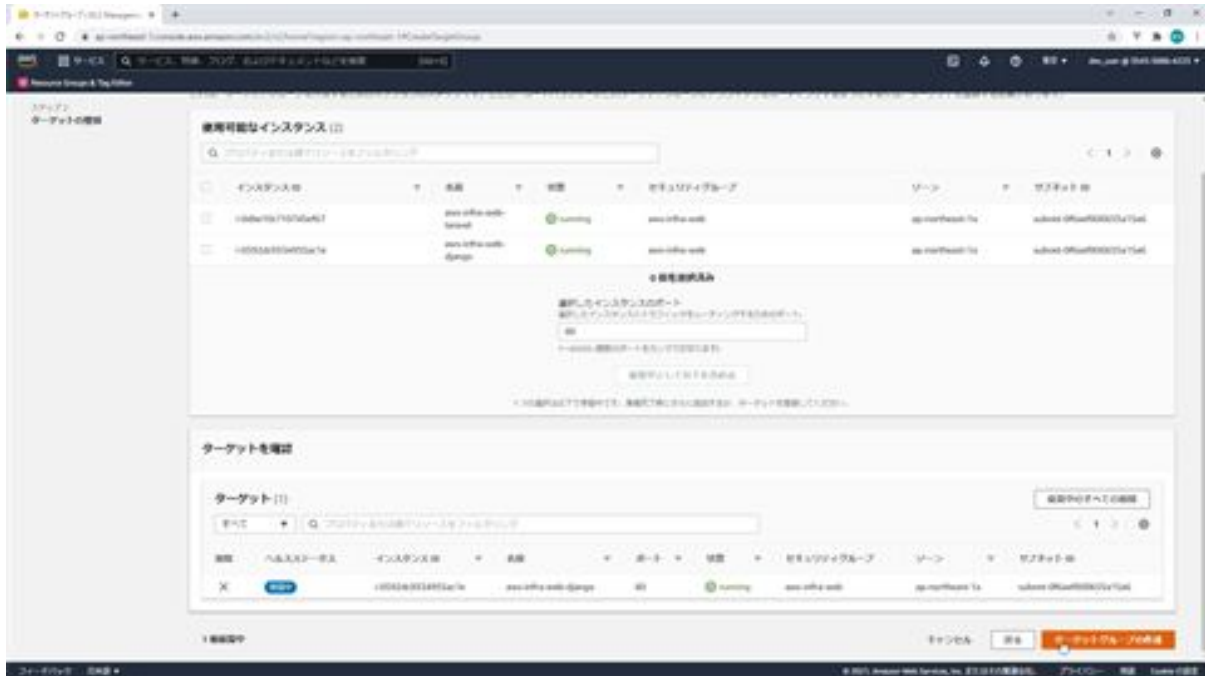
Next, in the listener and routing settings, create a new target group. Click on the Create Target Group link and when the Advanced Group Settings screen appears, select Instance as the target type and enter aws-infra-target-group as the target group name.



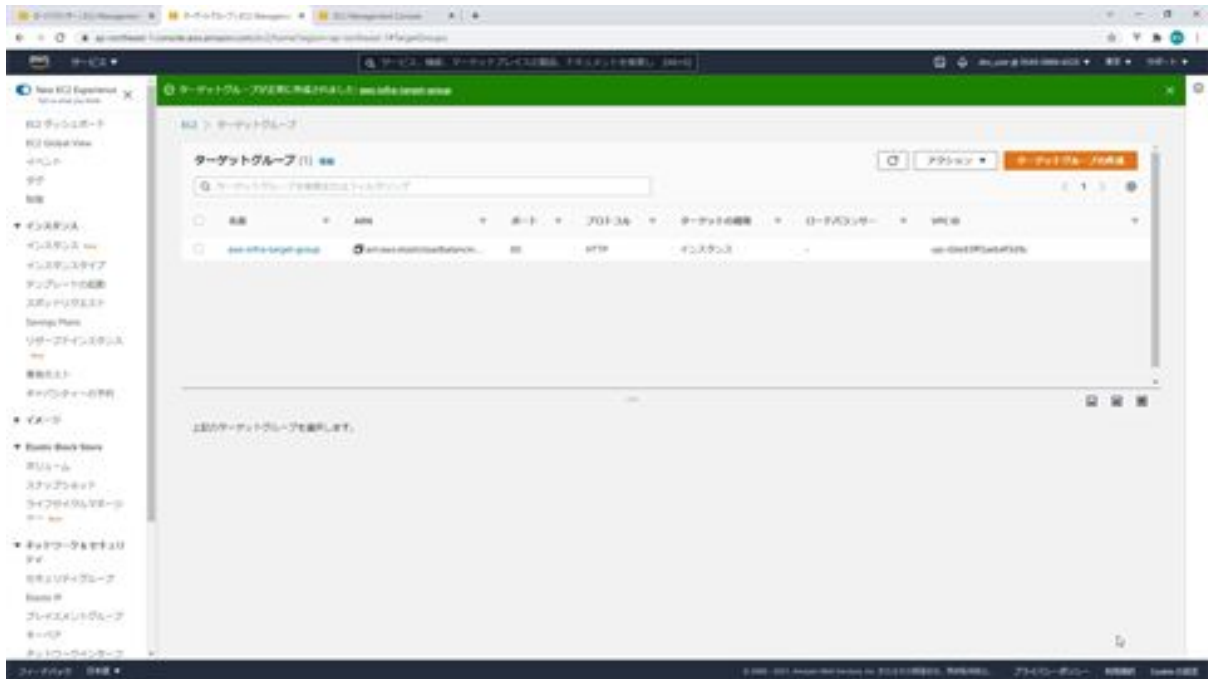
Set the protocol and port number to HTTP only, which is the default, and select aws-infra-vpc as the VPC. Select HTTP1 as the protocol version. The health check is to send periodic requests to the target to test the status. We will leave this as default.



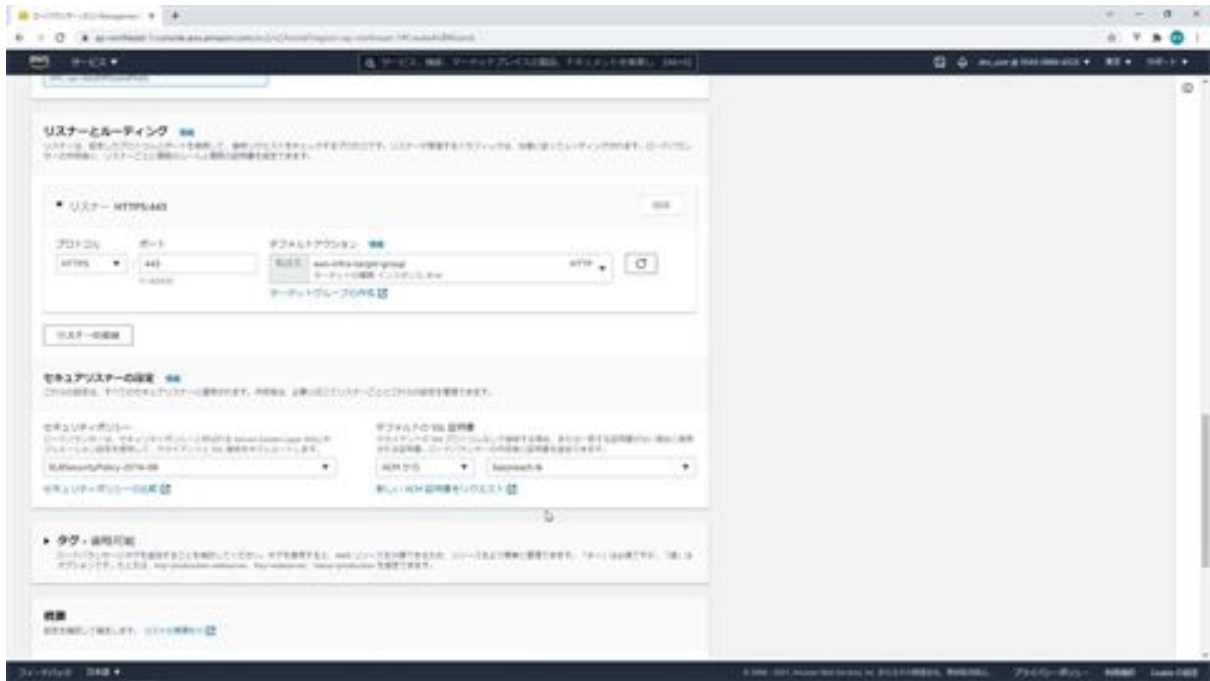
After doing so, click the Next button to display the target registration screen, check the aws-infra-web-django checkbox for the EC2 instance you want to target for the load balancer, click the Include the following as pending button, and then click the Create Target Group button. Click the Create Target Group button. If you check more than one instance here, the instances will be set to be sorted automatically.



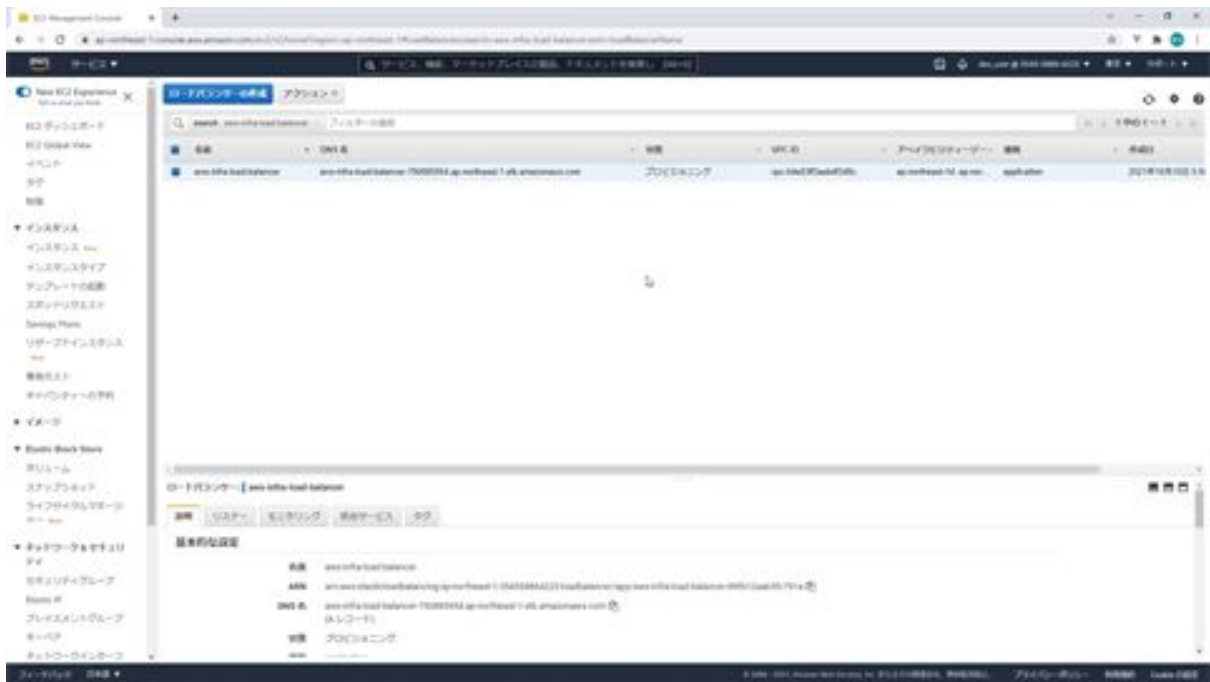
Now that the target has been created, we need to configure the routing.



Click the Update button on the right, select HTTPS as the protocol, and set aws-infra-target-group as the target with port 443. Next, in the secure listener settings, leave the security policy at the default settings, select the default SSL certificate from ACM, and select the domain of the SSL certificate you created. After entering all this information, click on the Create Load Balancer button.



The load balancer has been created. That's it for the load balancer settings.

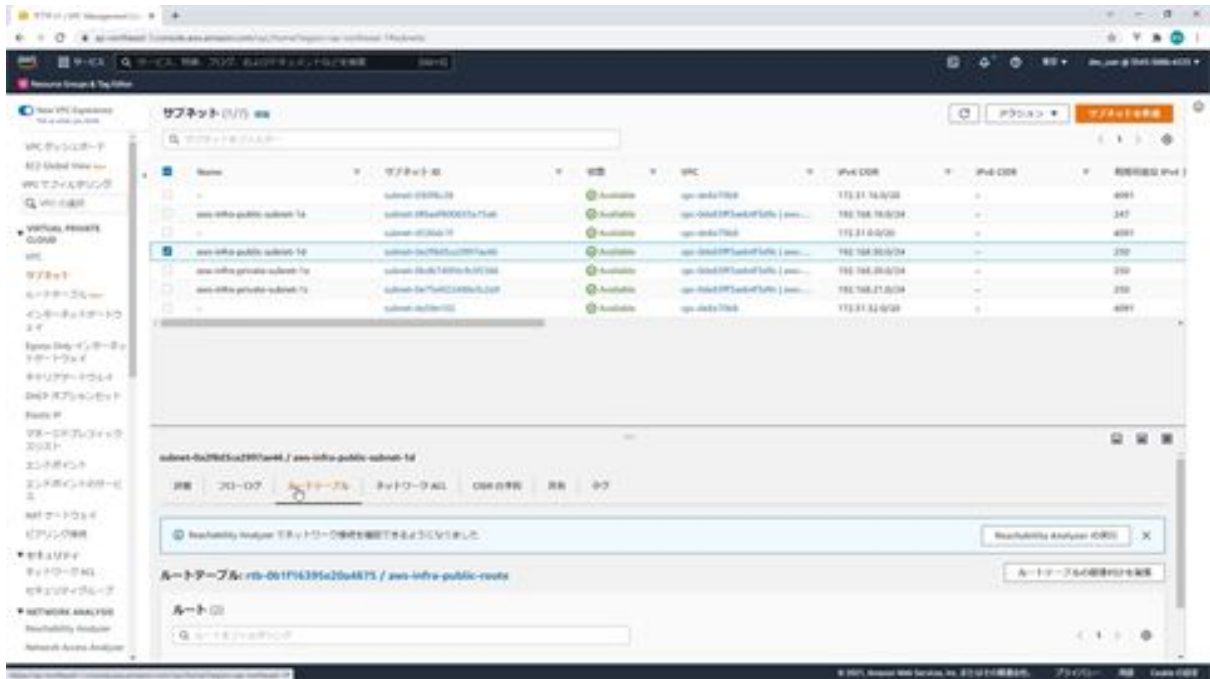


Let's connect the subnet for the load balancer to the Internet gateway

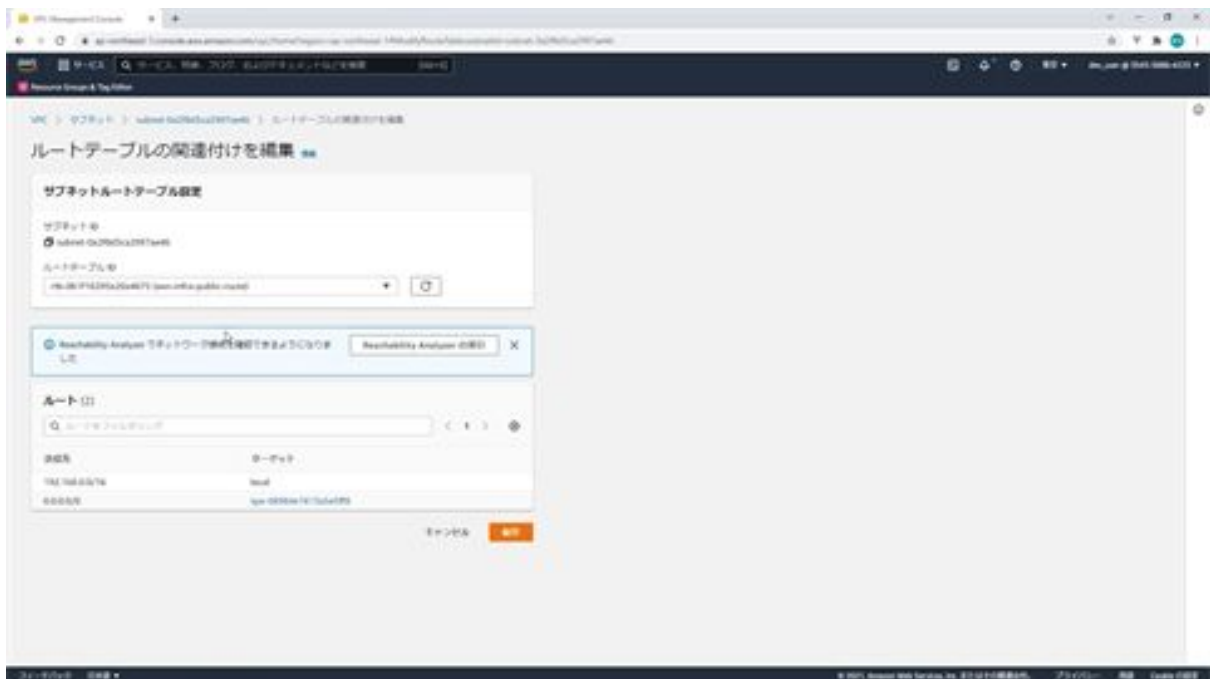
Next, let's configure the subnet created for the load balancer to connect to the Internet gateway. You may wonder why you need to connect a subnet that is not in use to the Internet gateway, but if you don't connect the subnet configured for the load balancer to the Internet gateway, AWS access will be very slow.

In this case, since we will be making an API connection on the Wix side, if the HTTP communication takes longer than 14 seconds on the Wix side, the communication process will be interrupted and an error will occur. To improve access, we will configure the server to connect to the Internet gateway. Now let's do the actual work.

In the management console, search for VPC to display the VPC dashboard. Next, click on Subnets in the menu on the left and select the `aws-infra-public-subnet-1d` that you created. After doing so, click on the Route Table tab at the bottom and click on the Edit Route Table Associations button.

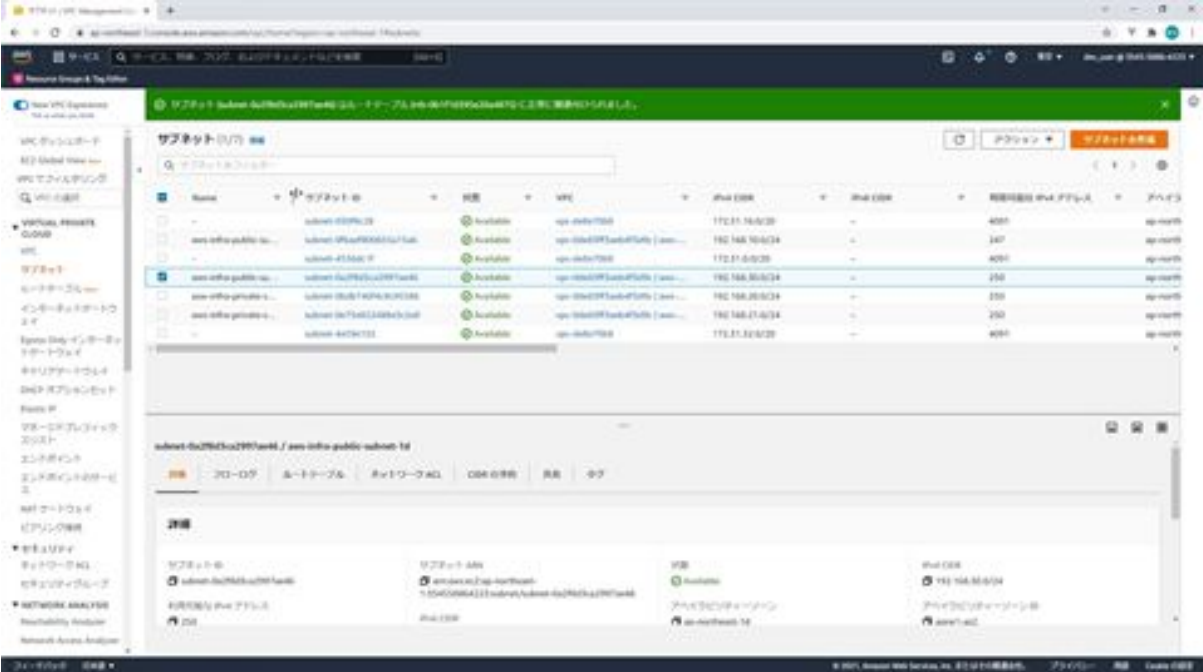


When the Edit Route Table Associations screen appears, change the Route Table ID to aws-infra-public-route, and click the Save button at the bottom.



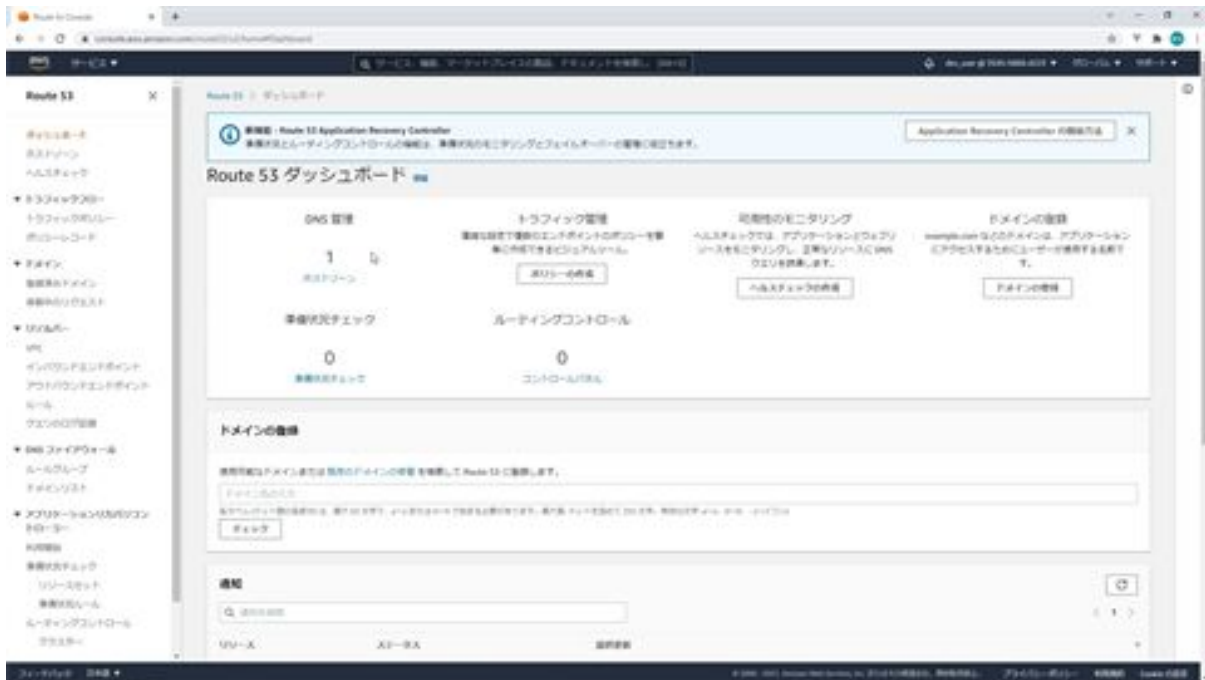
Now, aws-infra-public-subnet-1d is connected to the Internet gateway. That's it for the Internet gateway connection of the subnet for the load

balancer.

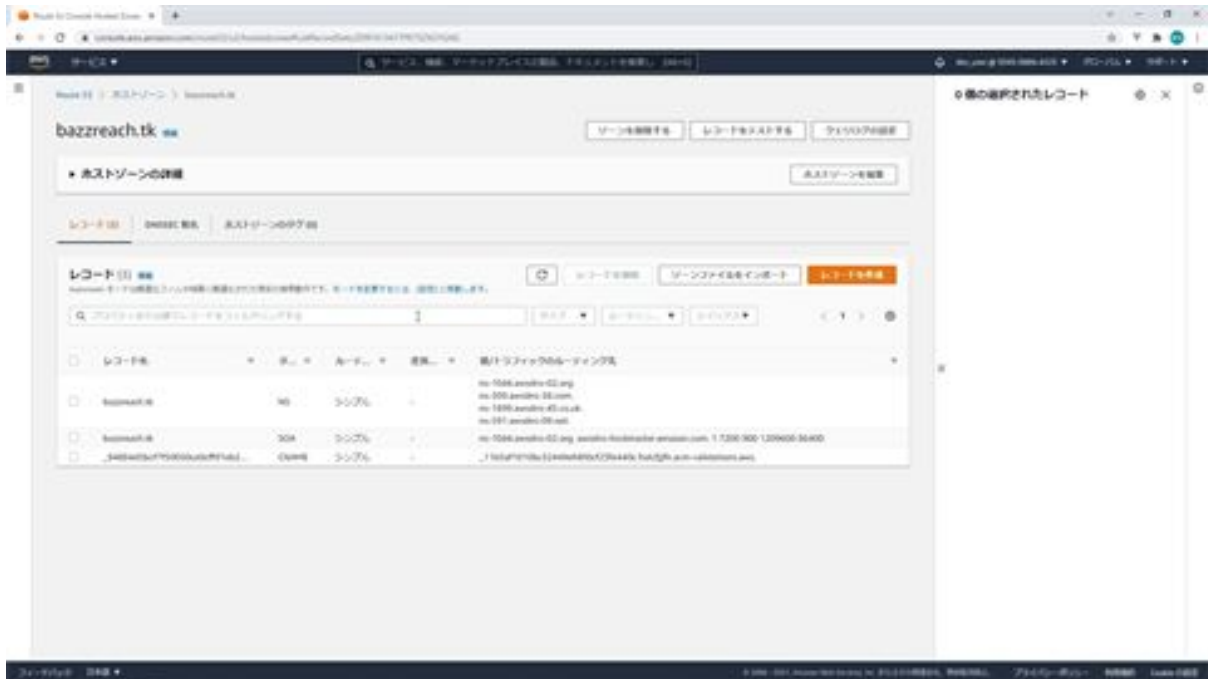


Let's set up an ELB with Route 53

Next, let's configure the ELB on Route53. Once you have completed this configuration, you will be able to specify the domain and even the webserver screen. First, go to Route53 and click on the hosted zone.



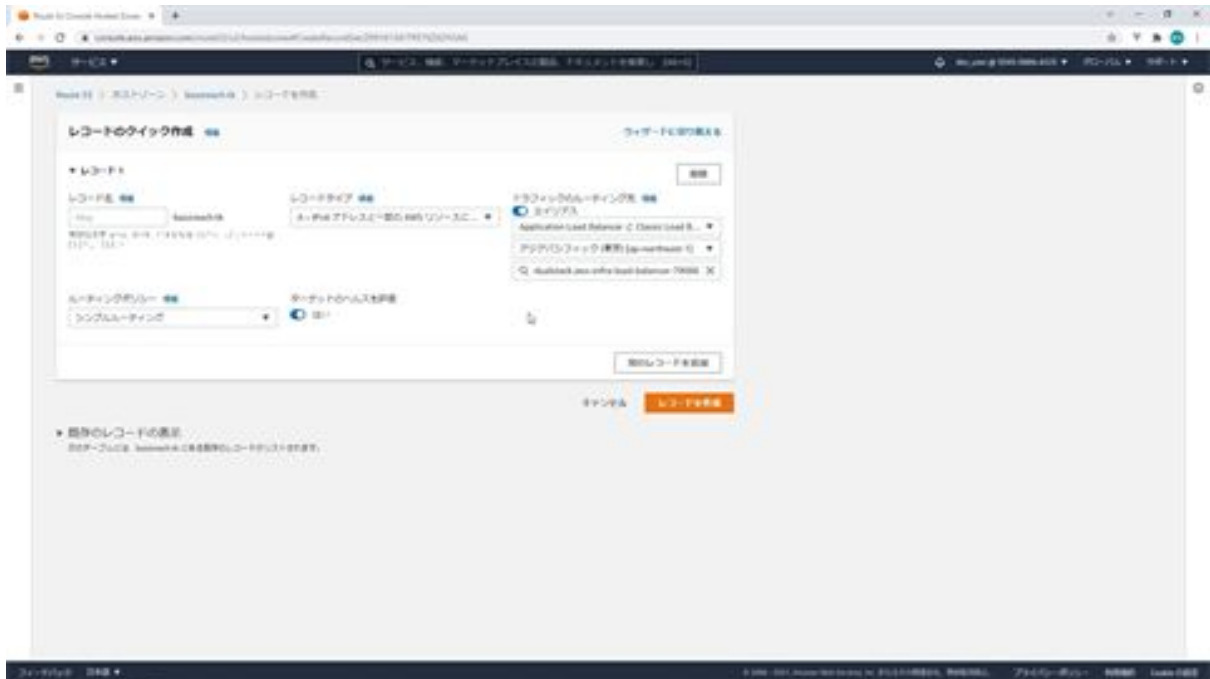
Click Domain in the Host Zone screen, and then click the Create Record button in the Record Settings screen to display the Create Record screen.



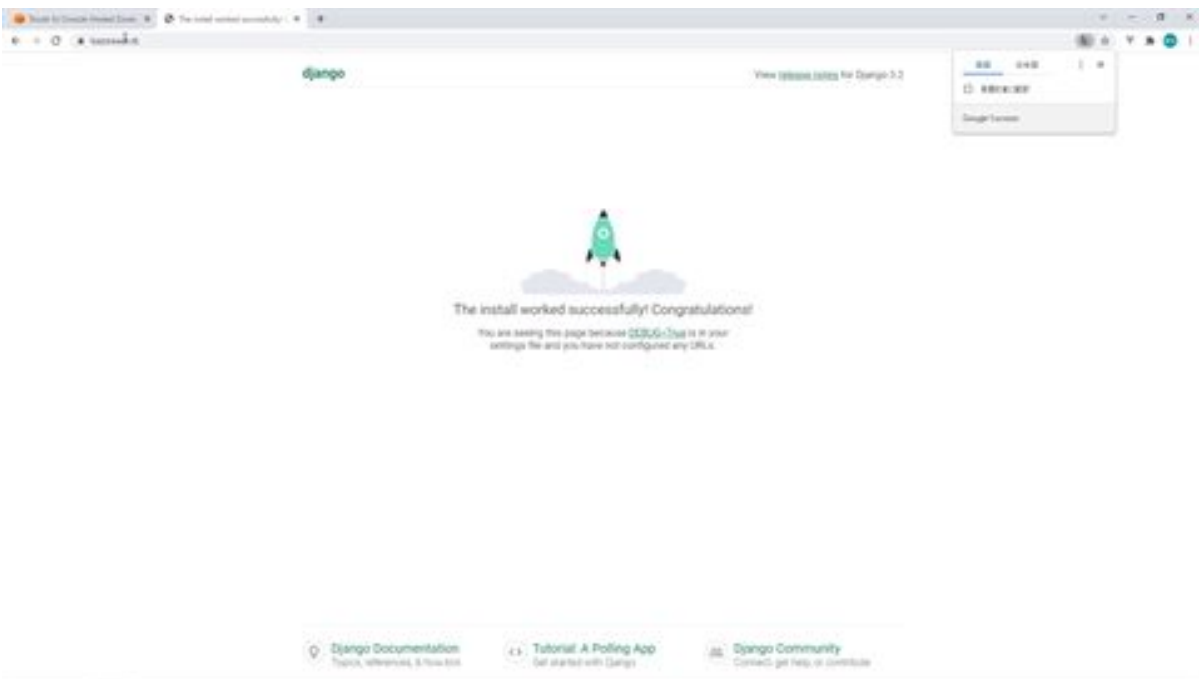
First, leave the record name unentered since we will not be changing the domain. As for the record type, we will set it to A. Alias is an option to specify where to string the AWS-specific name instead of the IP address.

In this case, it is necessary to enable HTTPS, so turn it on, set the traffic routing destination to Application Load Balancer and Classic Load Balancer, and select Tokyo as the region.

Next, for the load balancer, select aws-infra-load-balancer. Next is the routing policy, which can be set when there are multiple servers, but since there is only one server, in this case, we will select simple routing. When you have finished entering the information up to this point, click the Create Record button.

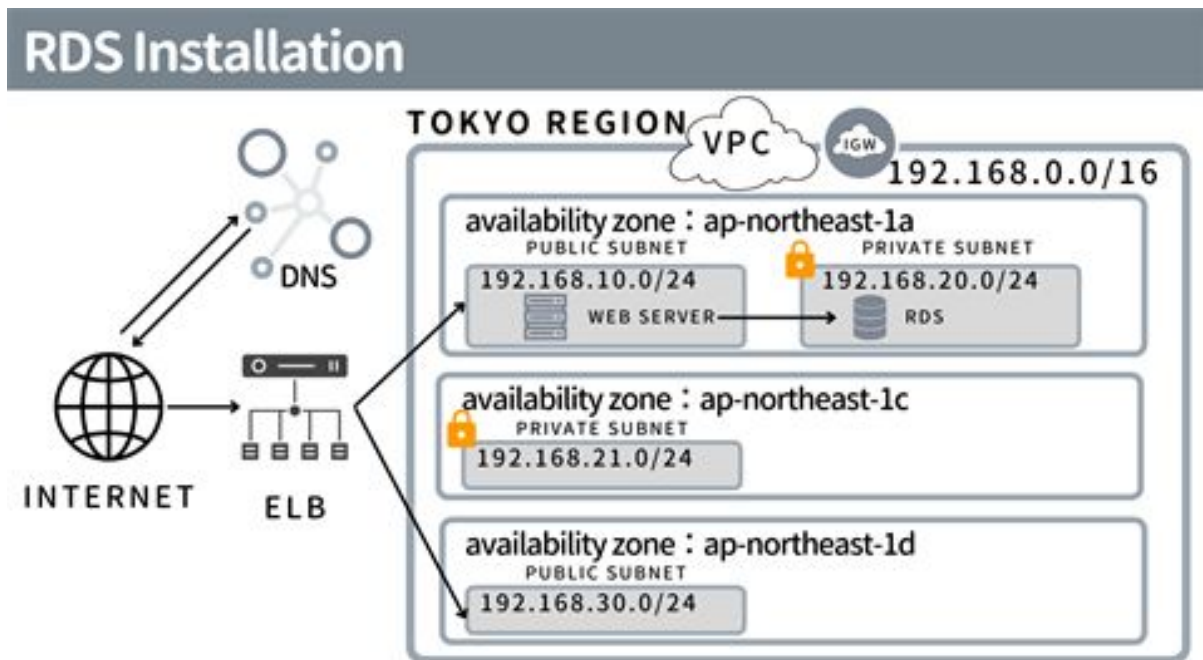


This completes the configuration of the Route53 ELB. Finally, let's check if Django is visible from the domain. That's it for the ELB configuration on Route53.



Chapter 6: Let's build a DB server

Let's create a private subnet for RDS

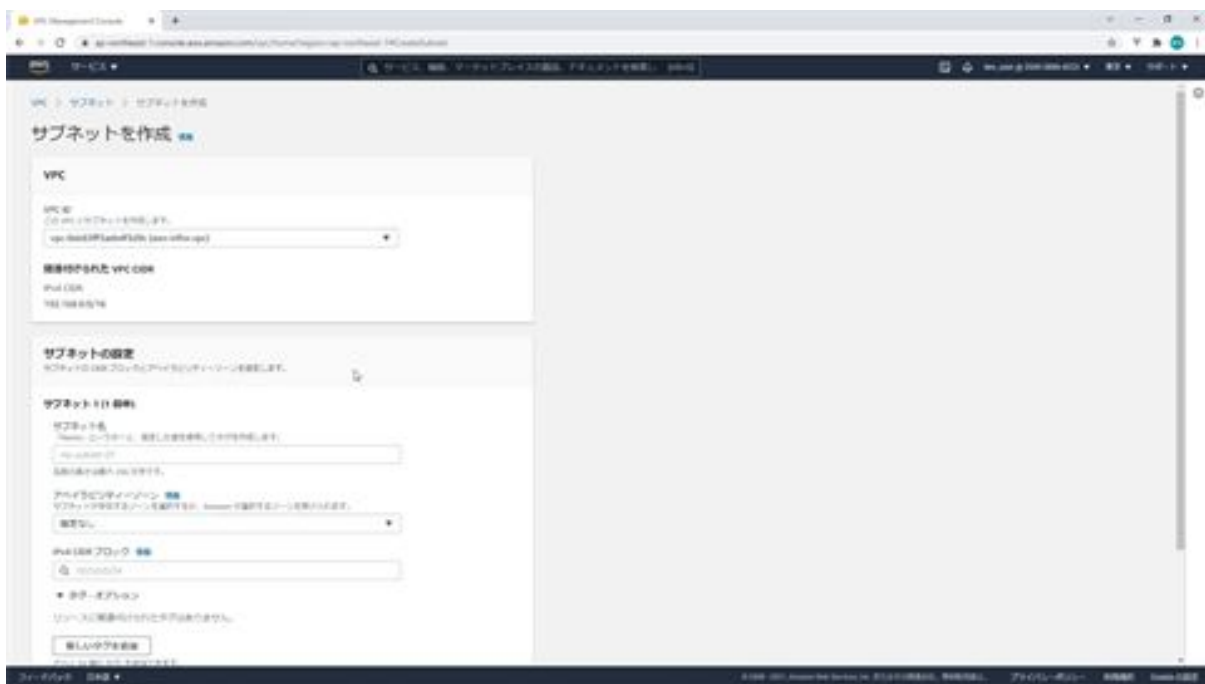


From this chapter, we will build a DB server. RDS is a DB service provided by Amazon on AWS, where the OS and DB engine is managed by AWS, and the user can start and access the DB in a few minutes. As with AmazonEC2, the usage fee is charged for the time the RDS instance is running. In this article, I would like to build a DB using this RDS.

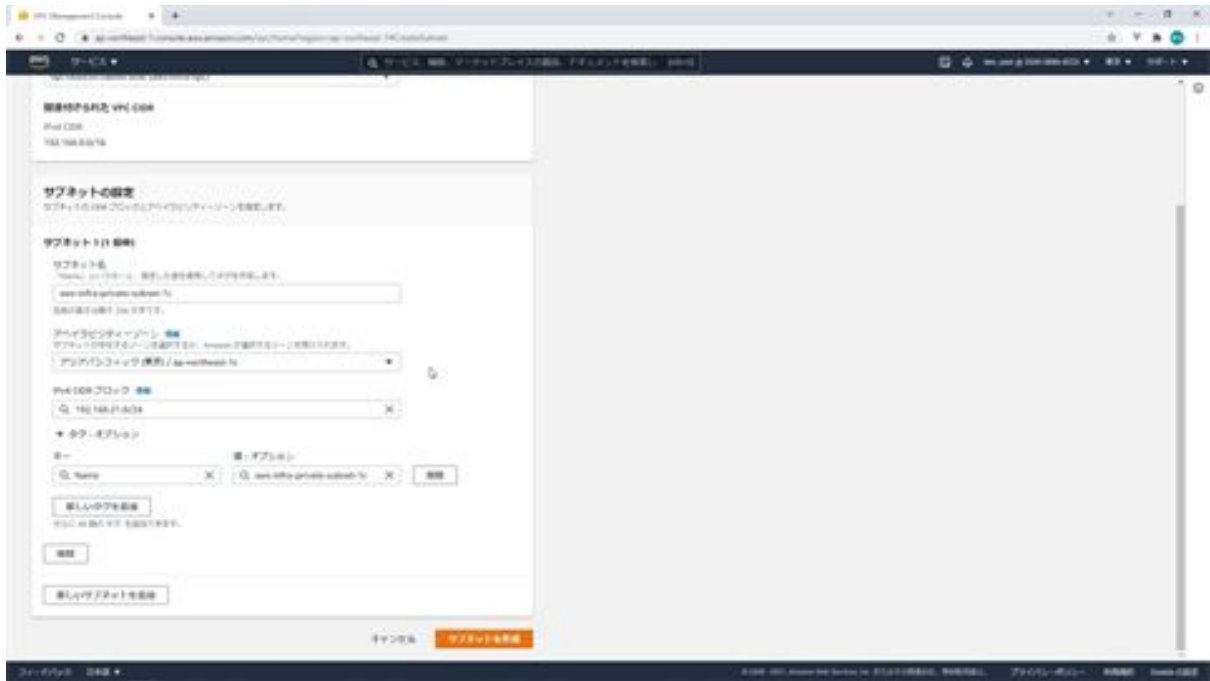
Before we can install RDS, we need to create a private subnet. RDS is a multi-AZ system that refers to multiple availability zones so that if one availability zone fails, we can continue to operate using other available zones. If one availability zone fails, the system will continue to operate using other normal availability zones. To use this multi-AZ, subnets must

be created in multiple availability zones. Currently, only one private subnet has been created, so we will create another one. Now let's go and do the actual work.

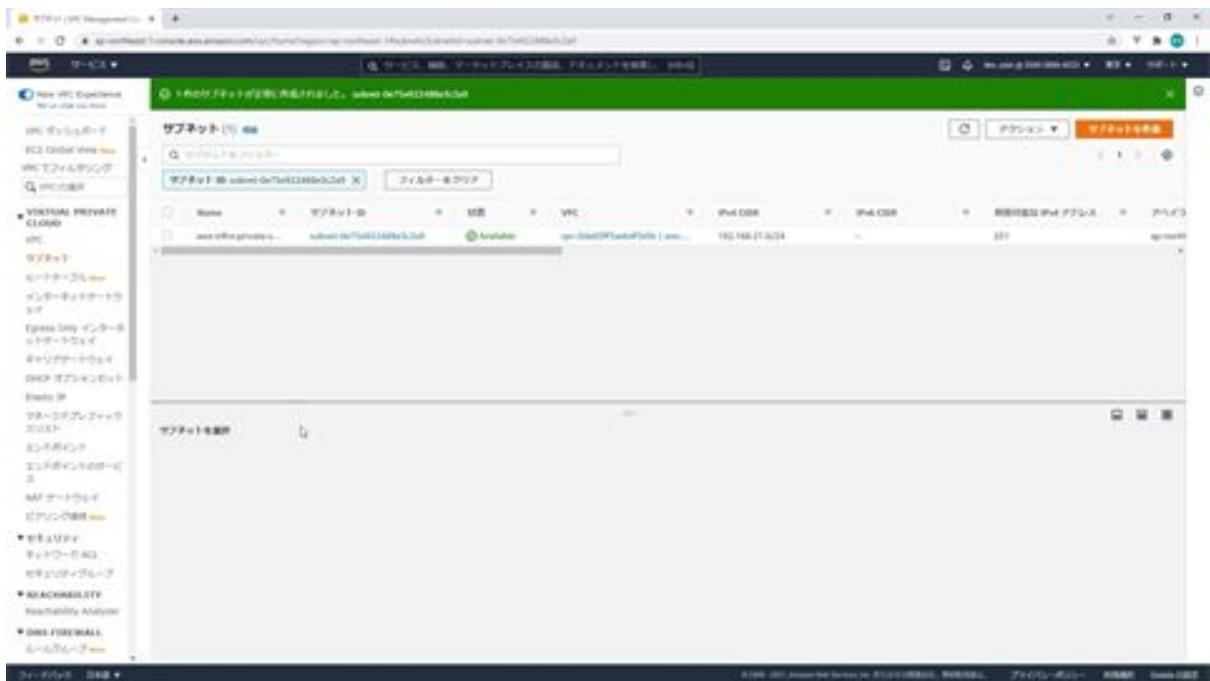
First, open your VPC from the AWS management console. Click on Subnets in the menu on the left. Next, click on the Create Subnet button. Now enter the information on the subnet creation screen: select aws-infra-vpc as the VPC ID, and enter aws-infra-private-subnet-1c as the subnet name.

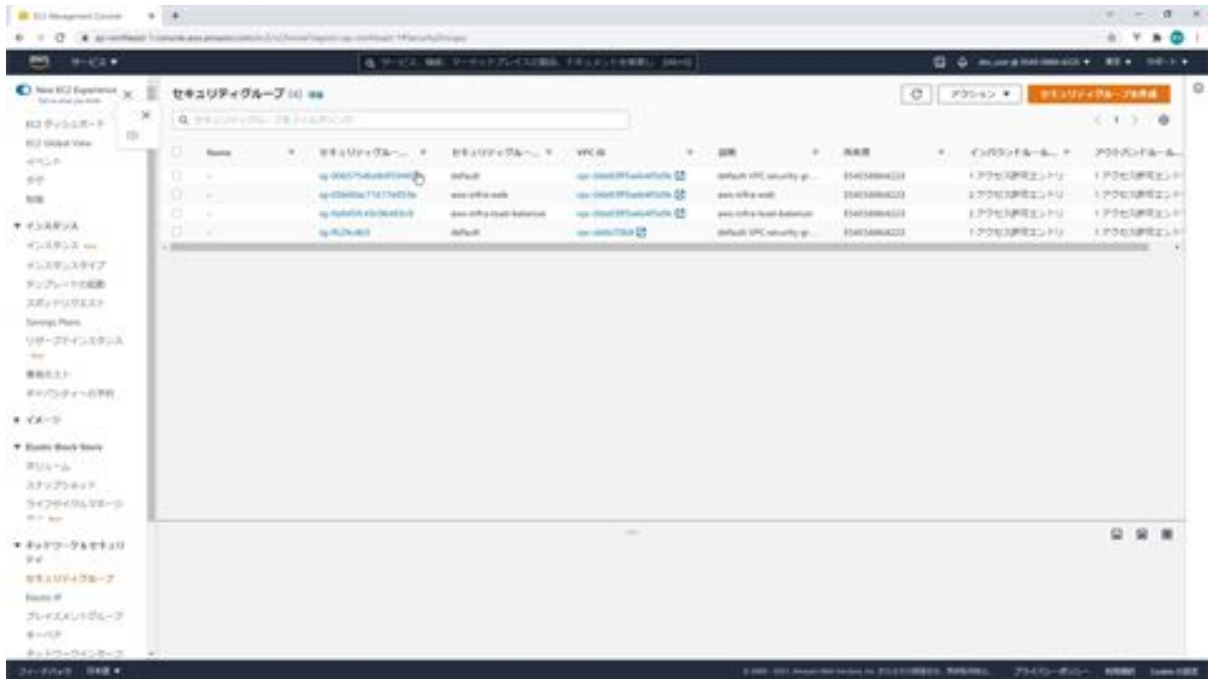


Select 1c for the availability zone and enter 192.168.21.0/24 for the CIDR block. When you are done, click the Create Subnet button.

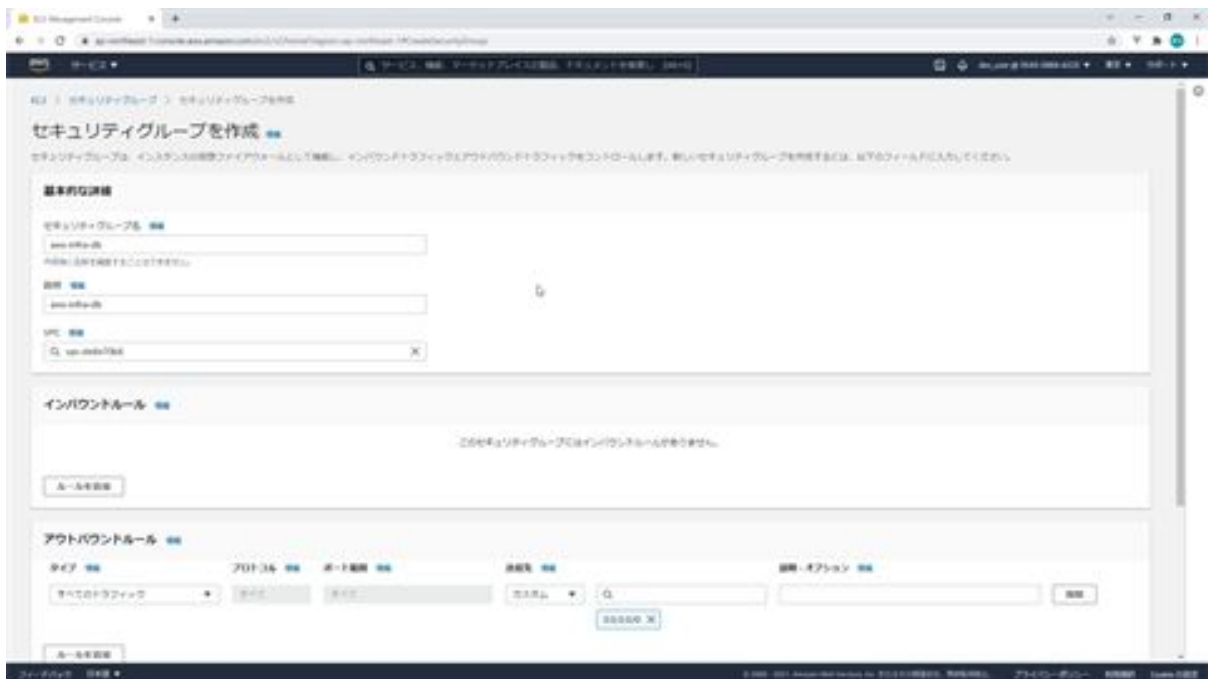


The subnet has now been created. This completes the creation of the subnet for RDS.



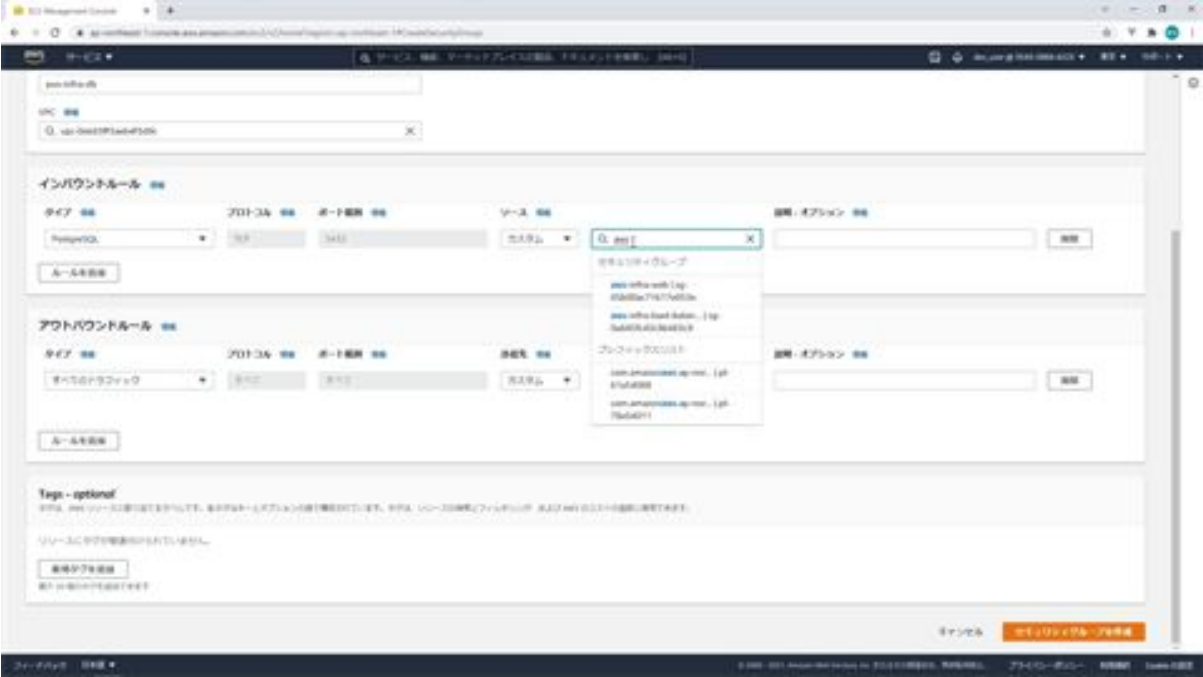


In the security group creation screen, enter aws-infra-db as the security group name and enter the same description. aws-infra-vpc is selected.

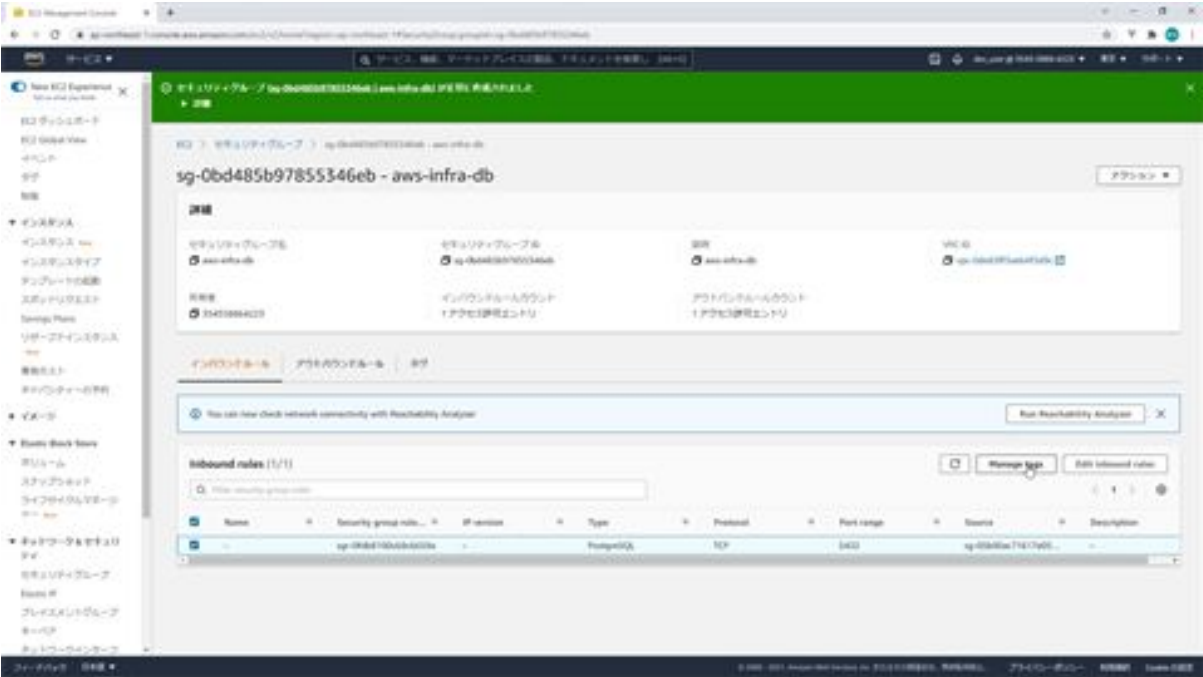


Next, we will add an inbound rule. Click on the Add Rule button and select PostgreSQL as the type. Select Custom as Source and type aws in the input

field. Then you will see aws-infra-web in the security group, select it.



Once you have entered this information, click on the Create Security Group button. The creation of the security group is now complete. This completes the creation of the security group for RDS.



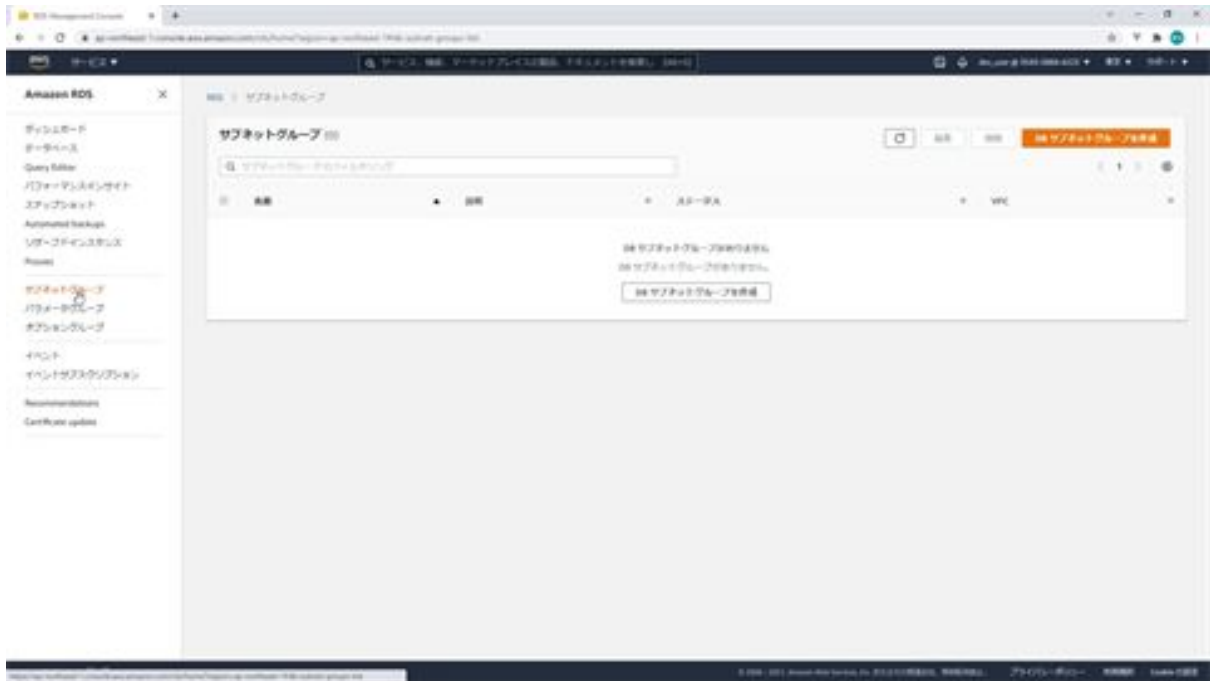
Let's create a subnet group for RDS

Next, we will create a subnet group for RDS. When you launch RDS, you will specify a DB subnet group, which is a configuration that specifies multiple subnets within a VPC on which the RDS instance will launch.

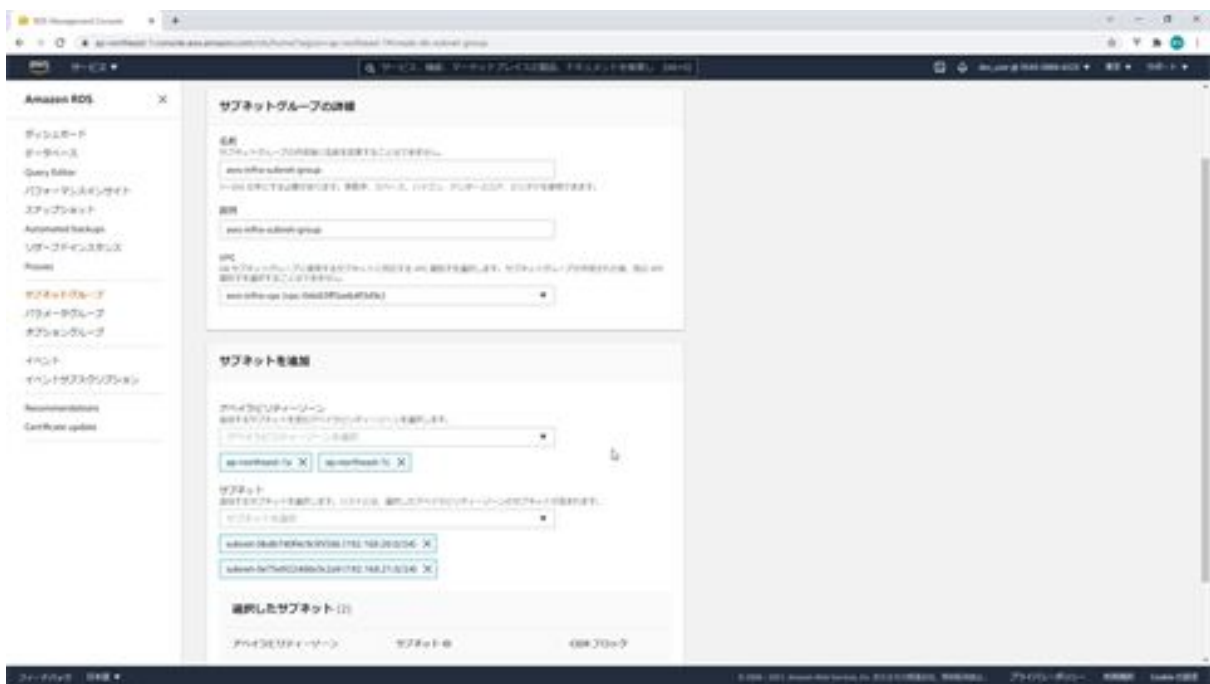
This is because we will be using multiple subnets in a multi-AZ, so we need to specify the DB subnet group when we create the RDS. Now let's proceed with the process. First, open the RDS from the AWS management console. Click on Subnet Group in the left menu.



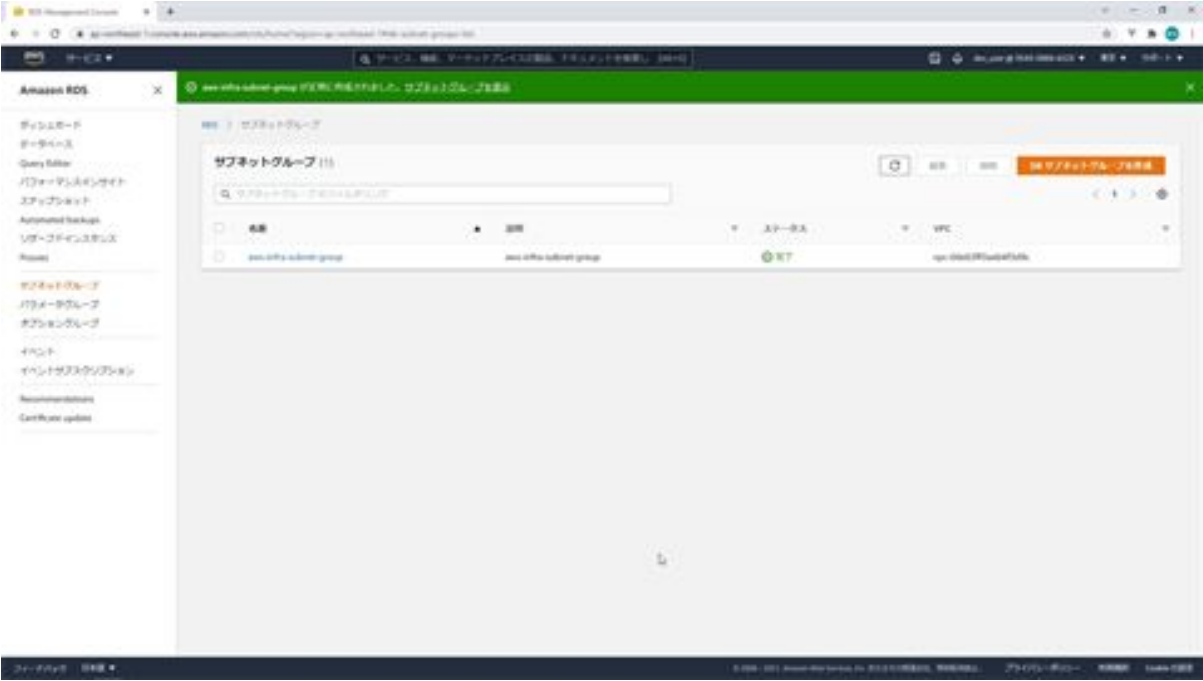
Next, click on the Create DB Subnet Group button.



On the DB subnet group creation screen, enter aws-infra-subnet-group for the name and the same for the description, and select aws-infra-vpc for the VPC. Next, we will add two private subnets in Add Subnet. Check the two availability zones 1a and 1c, and check the two subnets 192.168.20.0 and 192.168.21.0.

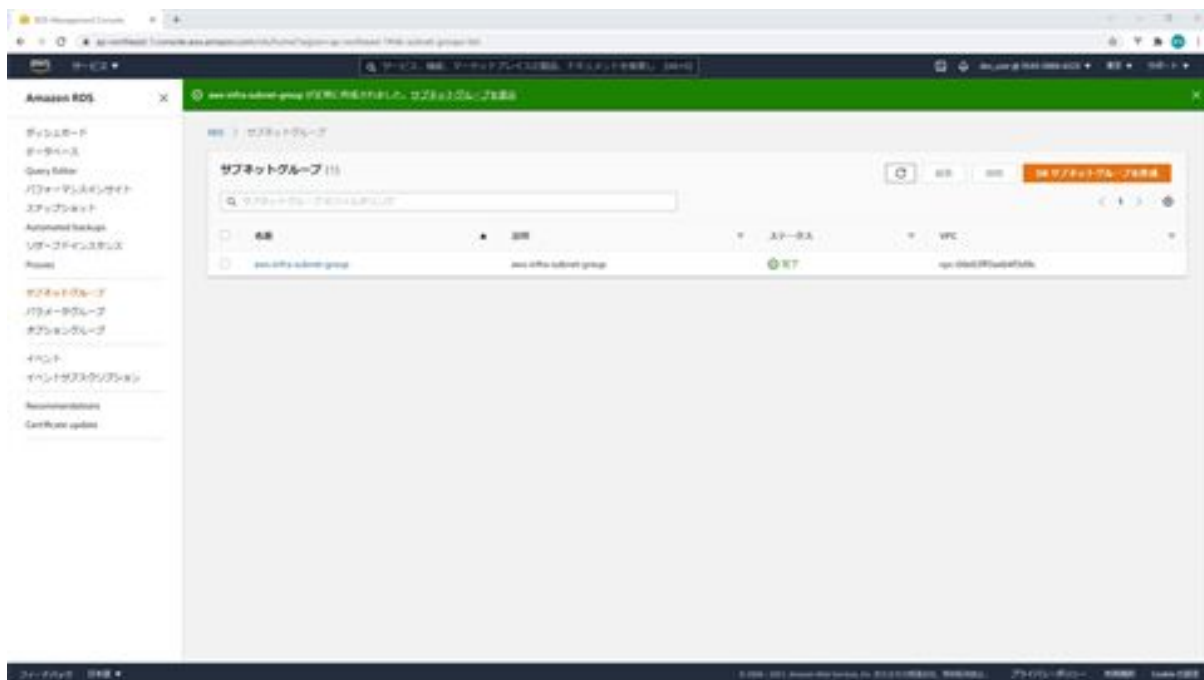


When you are done, click the Create button. This completes the creation of the subnet group for RDS, and if you specify the subnet group when creating RDS, the RDS instance will be created in the two private subnets you specified.

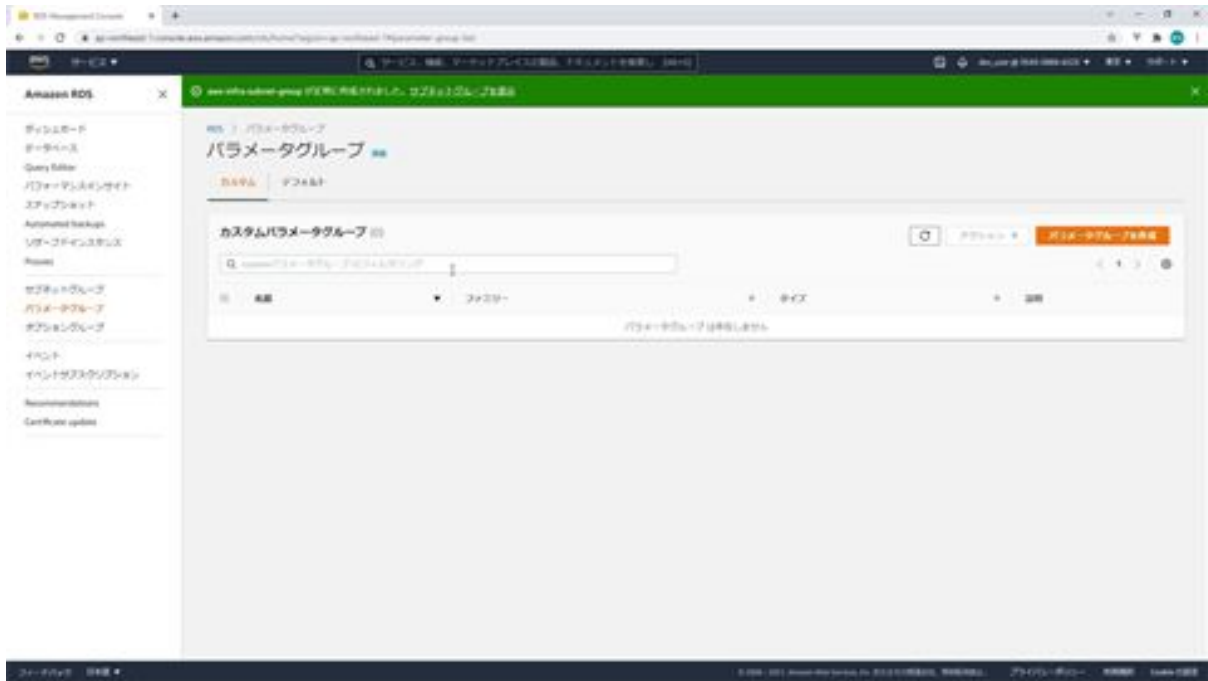


Let's create a DB parameter group for RDS

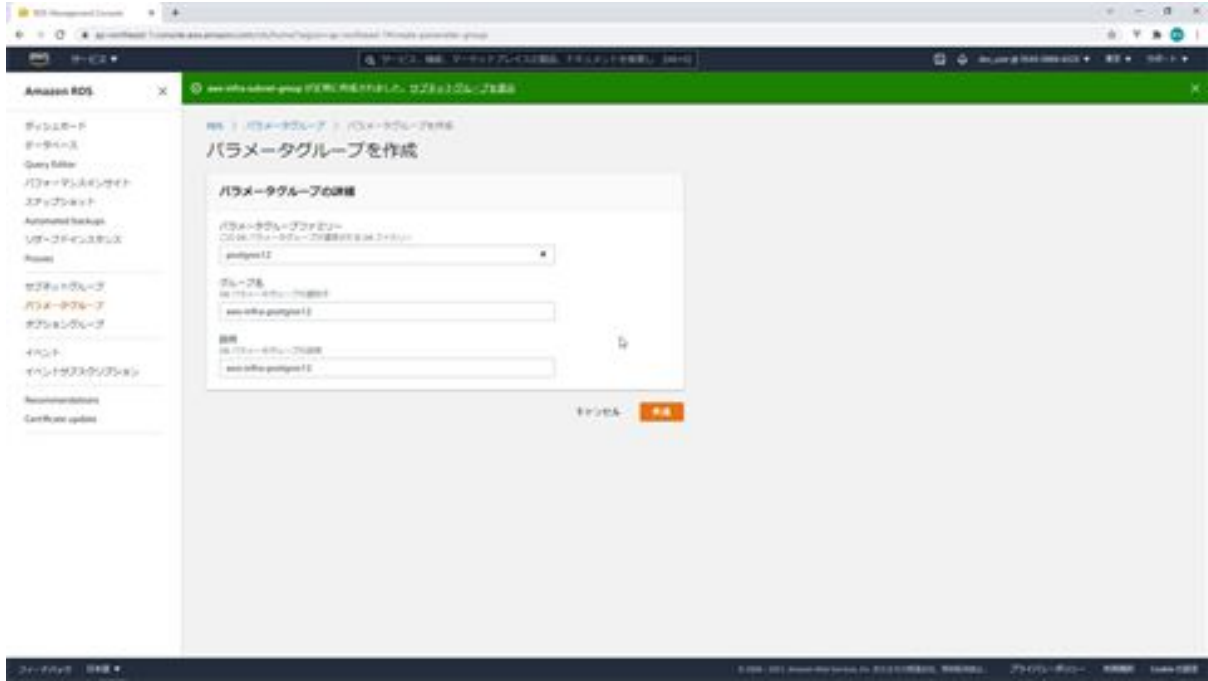
Next, we will create a parameter group for RDS, which is a DB parameter group that specifies the DB configuration values instead of the DB configuration file, since RDS does not allow direct editing of the DB configuration file. Now let's get down to work. Click on Parameter Group in the menu on the left.



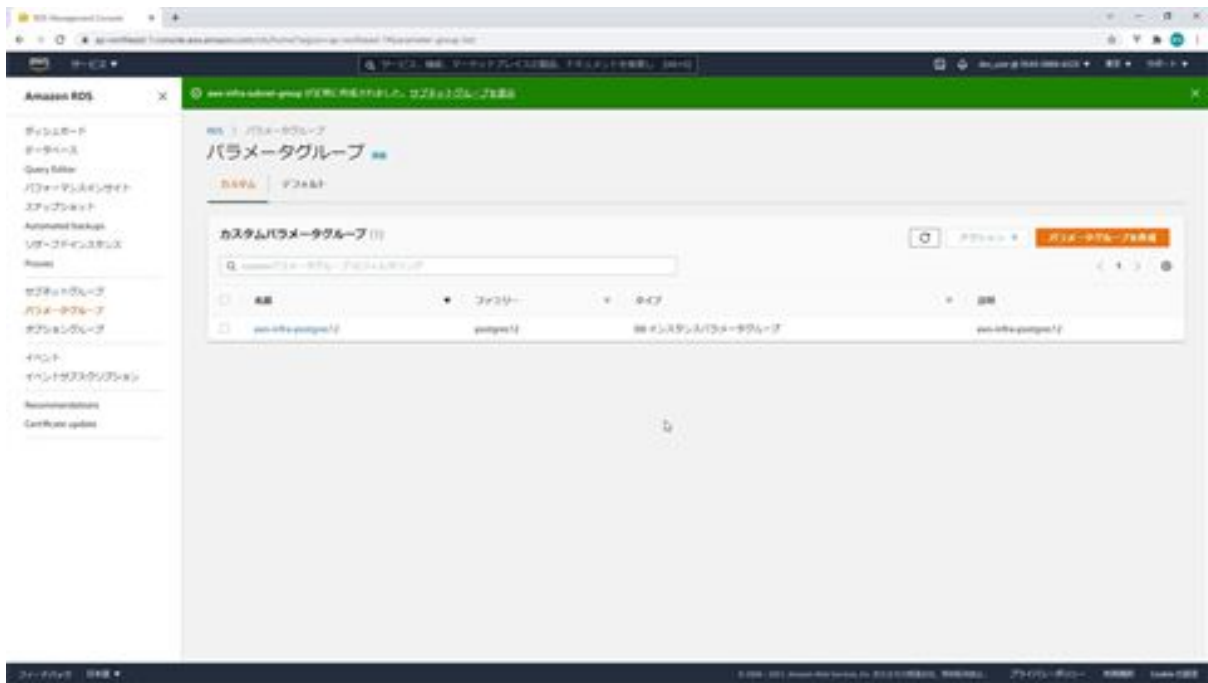
Next, click on the Create Parameter Group button.

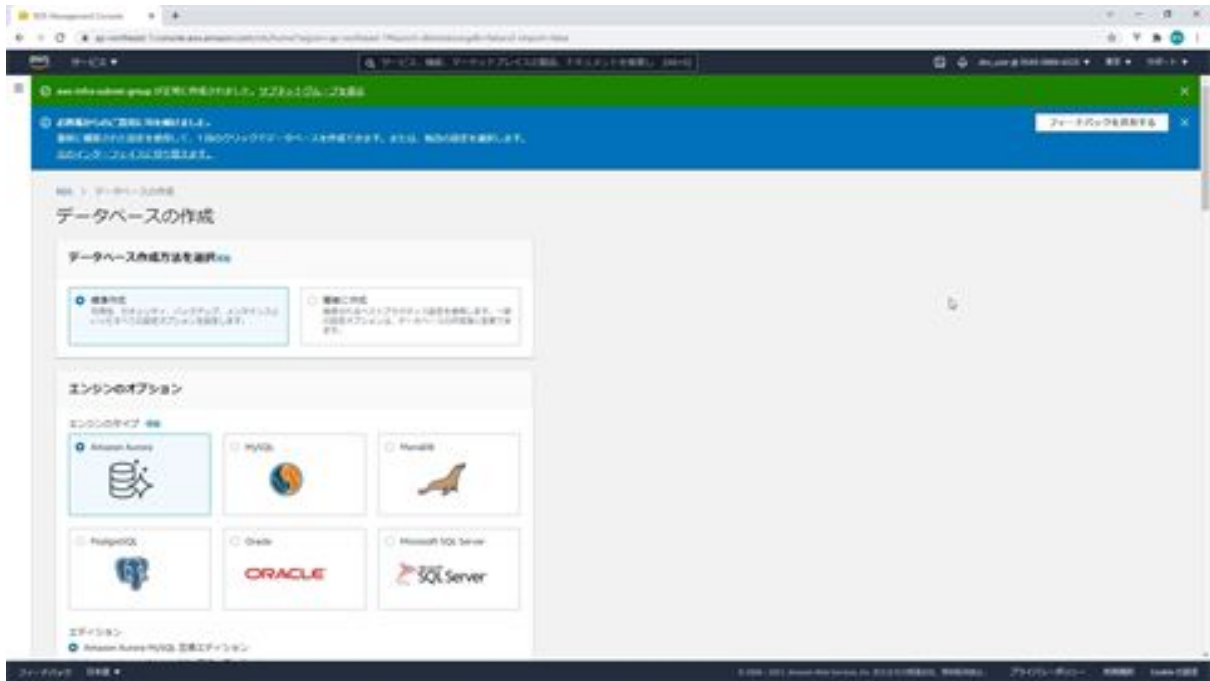


On the parameter group creation screen, select postgres12 for the parameter group family, enter aws-infra-postgres12 for the group name and enter the same for the description. When you are done, click the Create button.

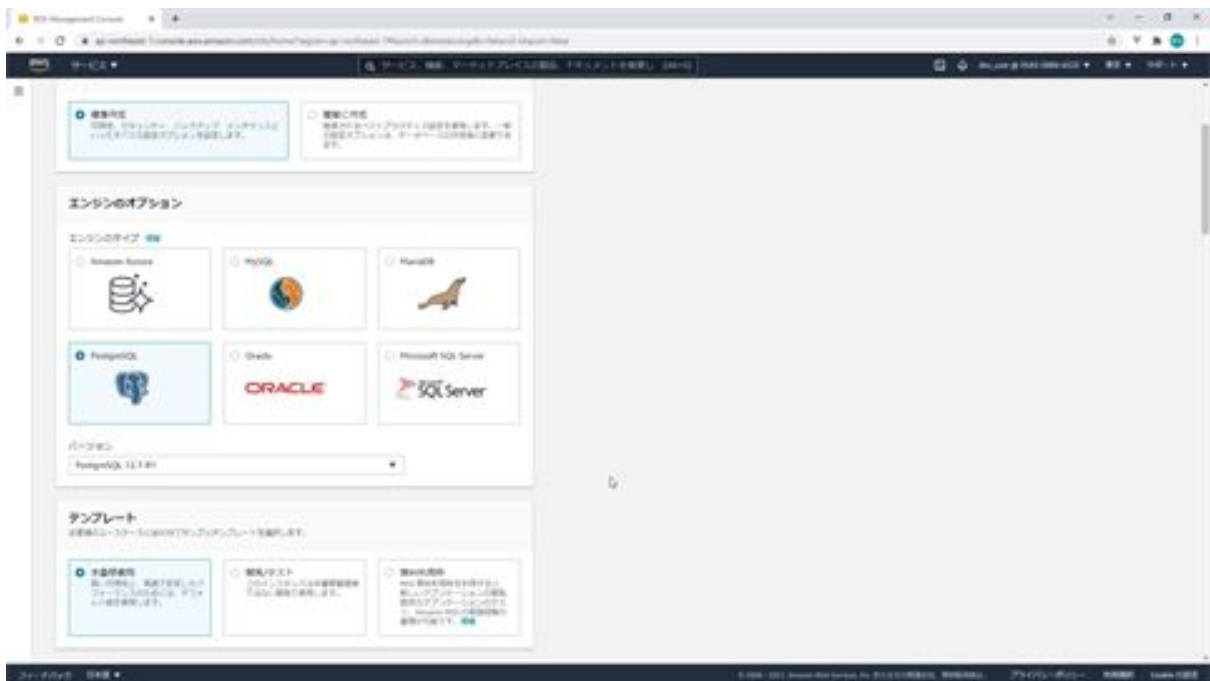


The creation of the DB parameter group for RDS is now complete.



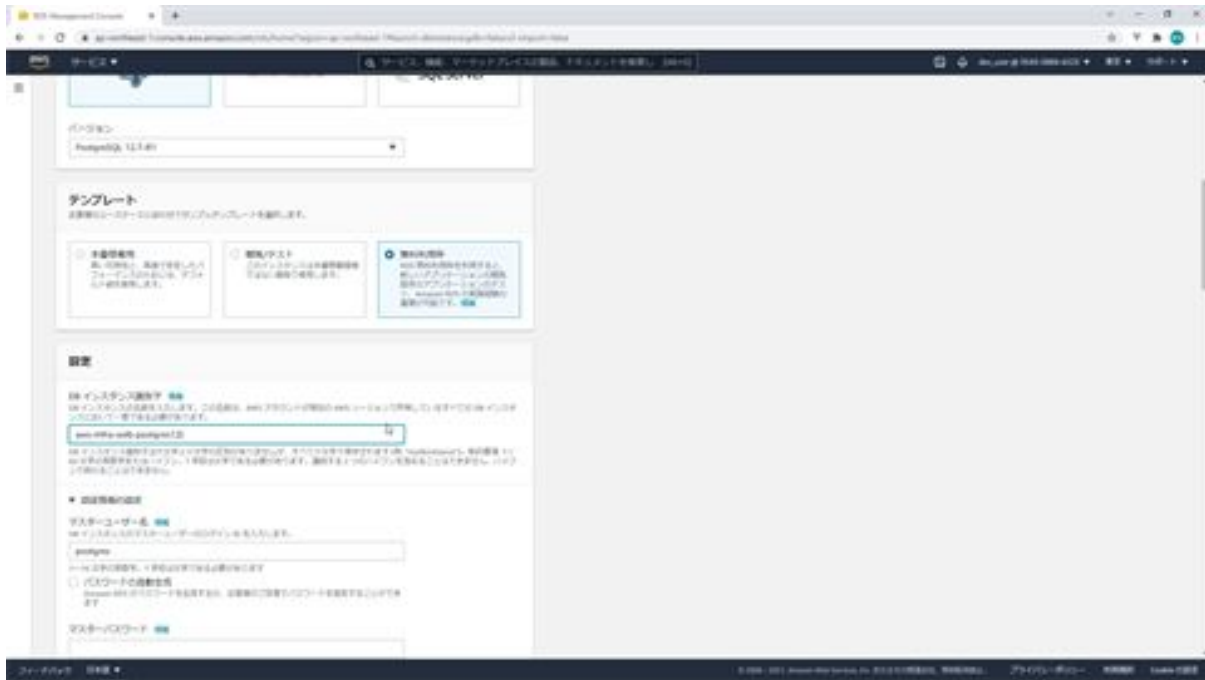


Regarding the version, select 12.7 for which a free usage allowance is available.

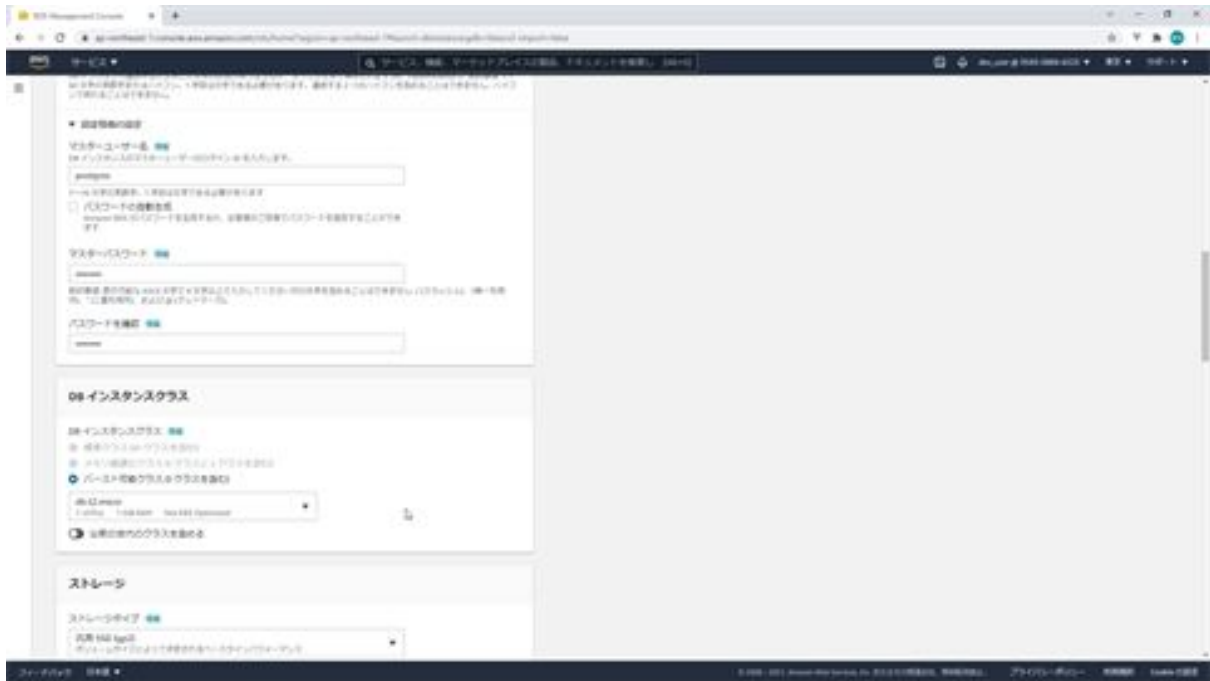


For the template, select the free use frame. The next DB instance identifier will be the name of the RDS instance. Here, enter aws-infra-web-

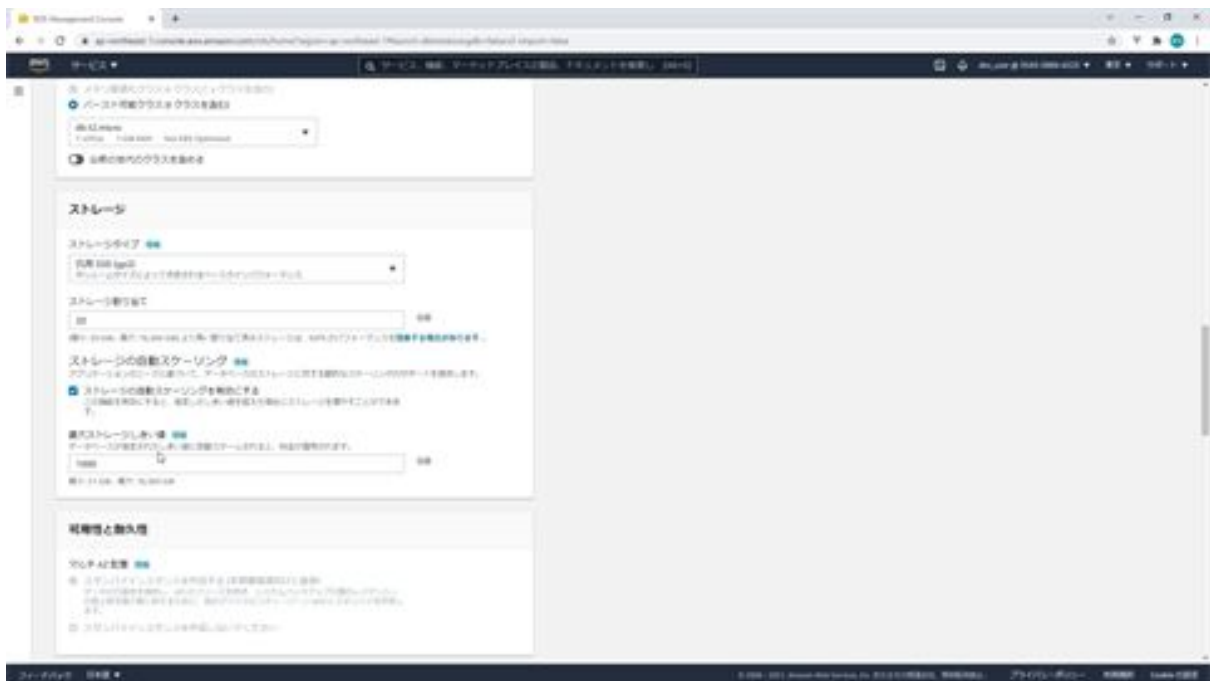
postgres12l.



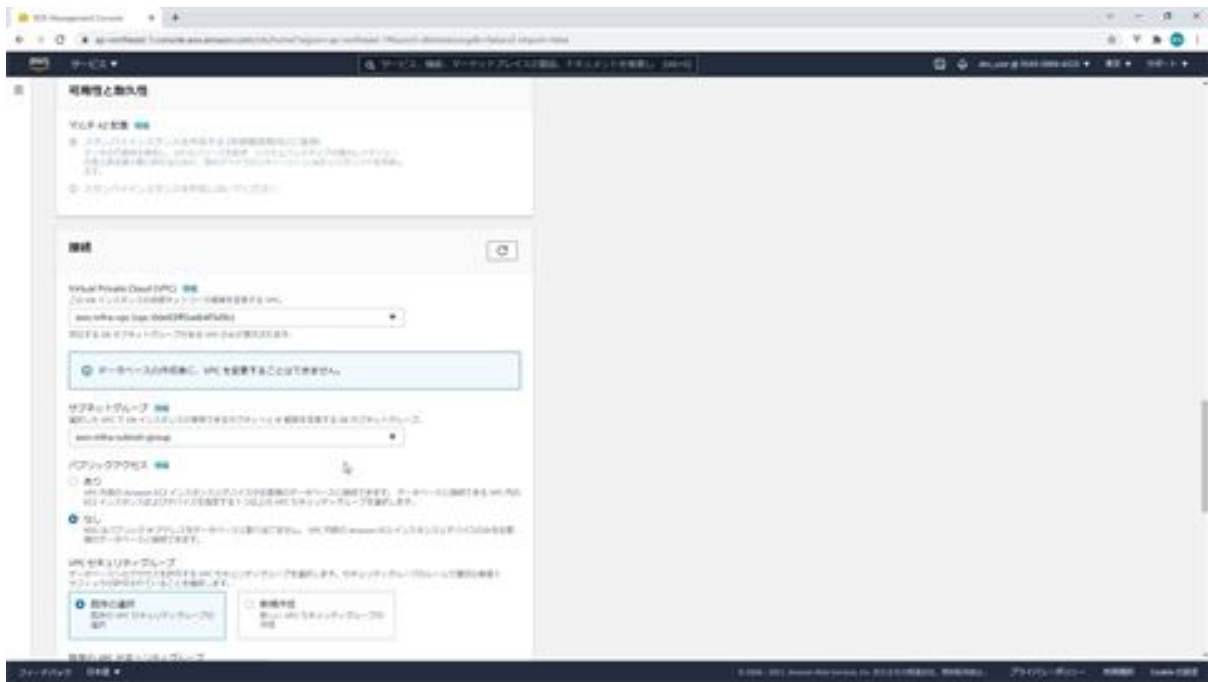
Next is the master user name of the DB instance, but since this is for learning purposes, leave it as postgres. For the automatic password generation, you should check the box, but since this is for learning purposes, leave it unchecked and enter "postgres" as the master password. Next, for the DB instance class, select db.t2.micro, a burstable class that can be used free of charge.



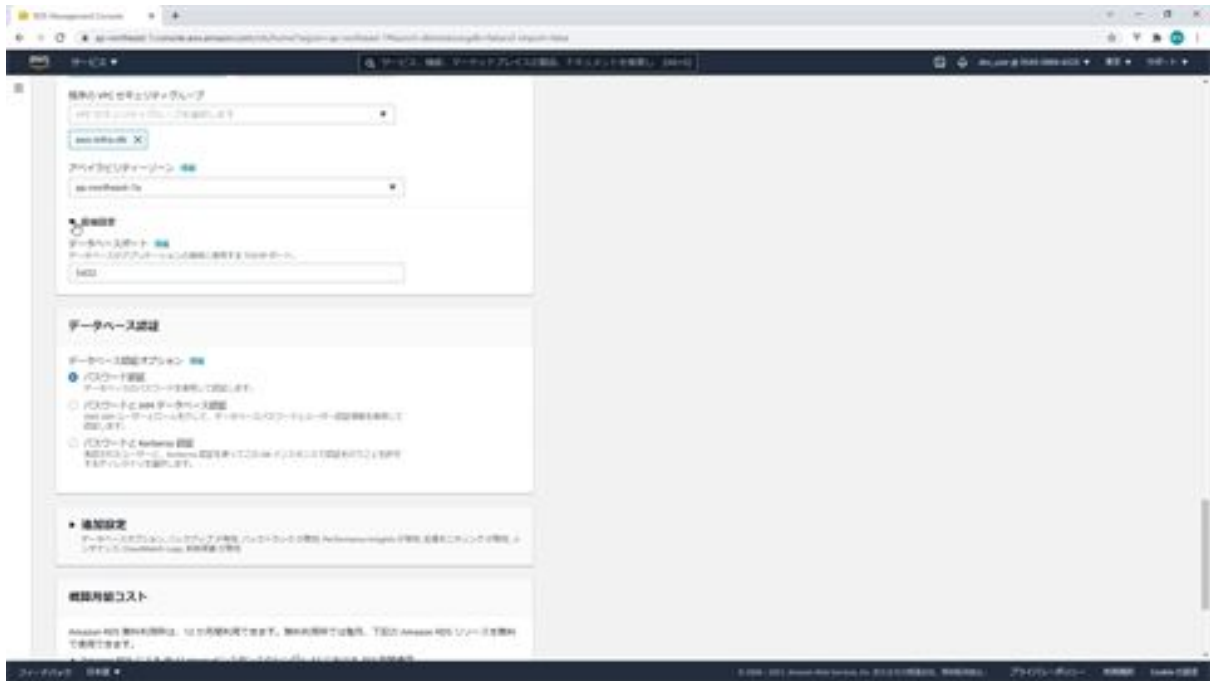
Next, for storage, set the storage type to general-purpose SSD and leave the storage allocation at 20 GB. This is an option to automatically increase the DB storage capacity when the DB load increases. Since we do not expect the load to be high this time, we leave it unchecked.



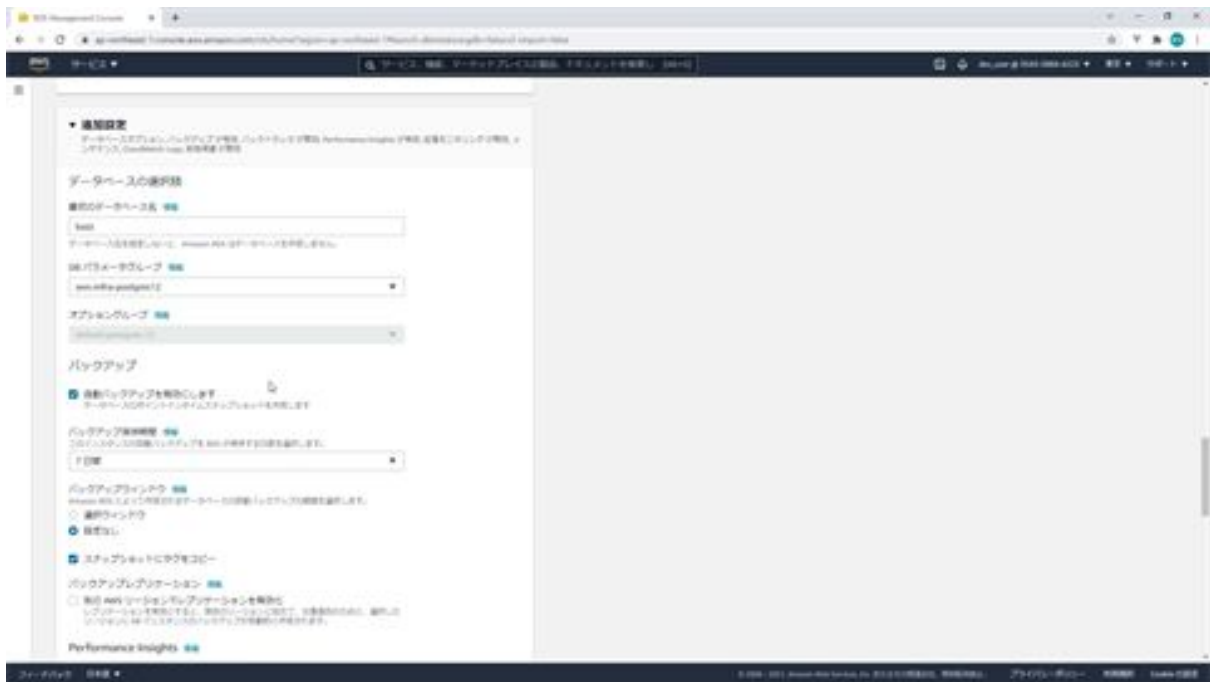
Next comes availability and durability, and you choose whether to deploy multi-AZ or not. In this case, since we have selected the free use slot in the template, we cannot select multi-AZ, but you will be able to select it for other than the free use slot. Since multi-AZ will increase redundancy, but will also increase the cost considerably, we will not create a standby instance this time and leave it as "please do not create a standby instance". Next, for the connection, select aws-infra-vpc for the VPC and aws-infra-subnet-group for the subnet group. Next is public access, which is whether you want to be able to connect to RDS from outside the VPC, but this time we want to prevent access from outside the VPC, so we select none.



Next is the VPC security group, here we select aws-infra-db. The default is already selected, but this should be removed. Next is the availability zone, which should be selected as 1a. The default database port is 5432. Next, for database authentication, leave the default password authentication.



For the first database name, enter bazz, and for the DB parameter group, select aws-infra-postgres12. Next, for backups, first, check the Enable automatic backups checkbox.



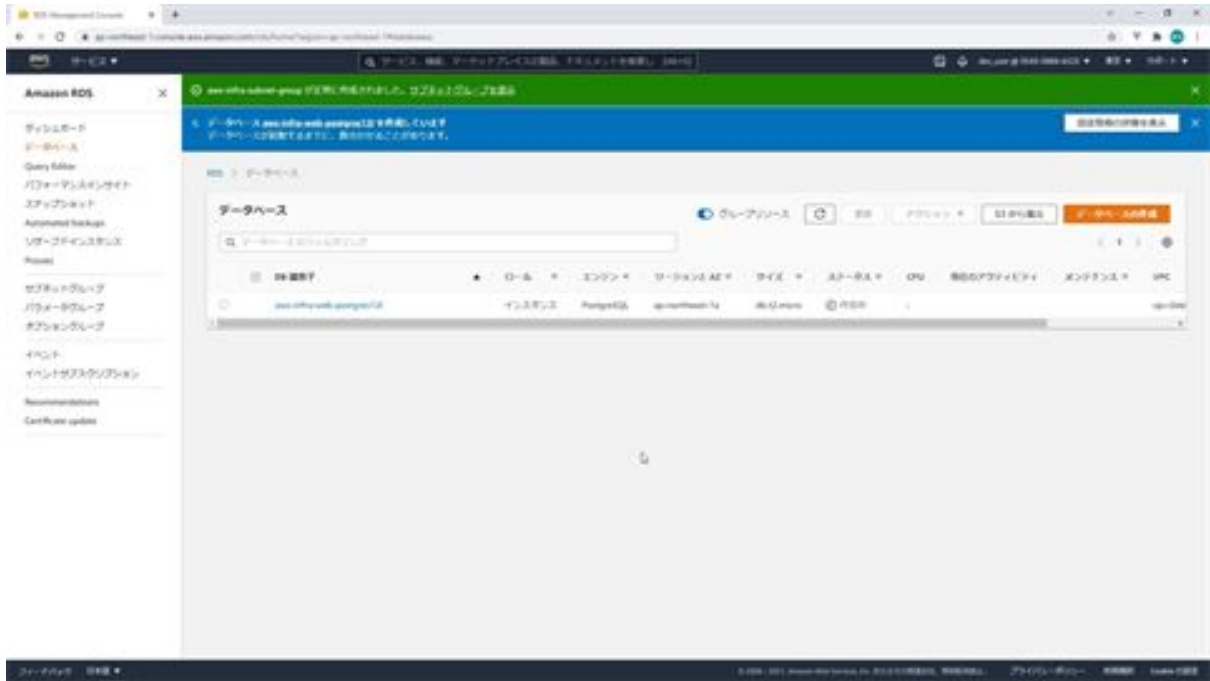
paid option and will not be used in this case, so leave it unchecked. The next option is to export logs to the CloudWatch service configured in the fee alert. Leave none of these unchecked this time.



The next maintenance, automatic minor version upgrade activation will automatically upgrade the minor version when a new minor version is released. Check the box here. The Maintenance window allows you to select a period for when the minor version upgrade will be performed. Here we choose the selection window and specify 6:00 p.m. on Saturday since we want the start time to be 3:00 a.m. on Sunday. We will leave the duration at the default of 30 minutes. The next option is to enable delete protection, which will prevent accidental deletion. In this case, we will leave it unchecked since we may want to delete it for learning purposes. When you have entered the information up to this point, click the Create Database button.



The database is now created and available. Database creation is now complete.



Chapter 7 Creating Screens with Wix

Let's create a Wix account

Now, let's create a Wix account. First, click the New Registration button in the upper right corner.



Next, click on the New Registration link, where you will enter your email address and password. Finally, click on the New Registration button. You have now created a new account with Wix.

新規登録

アカウントをお持ちですか？こちらからログイン

メールアドレス
syoni4ademy@gmail.com

メールアドレス
syoni4ademy@gmail.com

パスワード

パスワードを再入力



*ユーザー登録を行うことで、ユーザーIDとパスワードがシステム上で自動的に生成されます。また、アカウントのパスワードを再入力する必要があります。

Basic editing screen functions

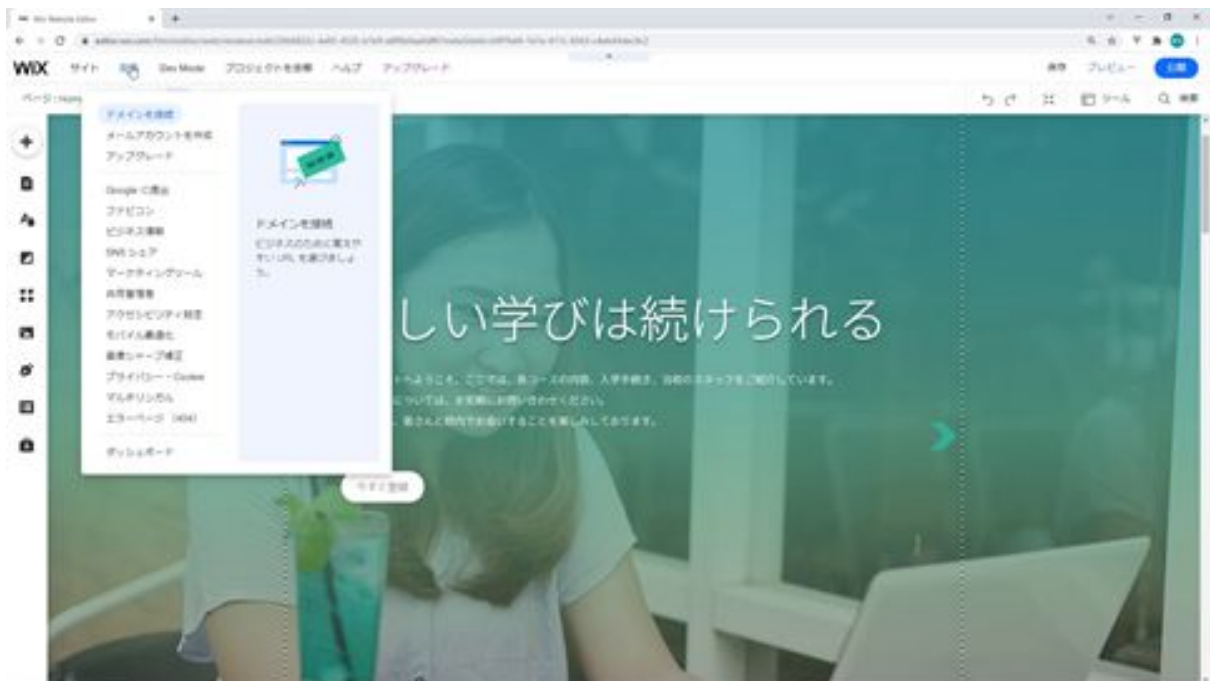
Let me explain the basic functions of the Wix editing screen. First, let's start with the menu in the editing screen, where you will find the Wix logo, then the site, and then the release manager where you can preview, publish, and AB test your work.



This edit history allows you to revert to the state of a previously saved website. By viewing the history of a saved website and clicking on the button, you can revert to the previously saved state.



Next are the settings. There are various settings in the settings. We will not use them here, so we will skip them.



Next is Dev Mode, which is the mode we will be using. Next, you will find the menu for requesting help with your project, upgrade, and upgrade to a

paid Wix plan. On the right side, you will find Save. By default, this is set to auto-save. This is the same as the auto-save function in word and excel. And where will it be saved? So, you need to be connected to the Internet to modify your work in Wix.



Next is the preview, which is a test display of your site. When you click on it, you will see a preview like this.



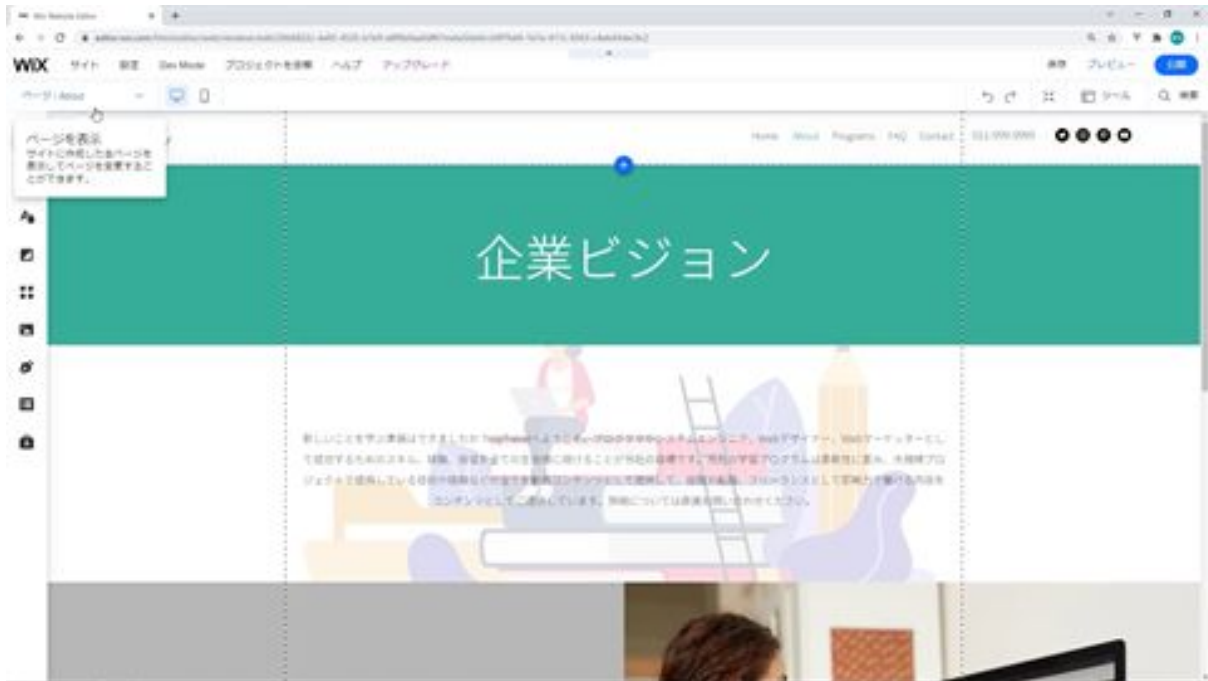
The next step is to publish the site. When you click the publish button, the saved site will be published on the web for people around the world to view. If you click on it, you will see a screen that says "Congratulations" and then a button that says "View Site", click on it and your site will appear on the screen.



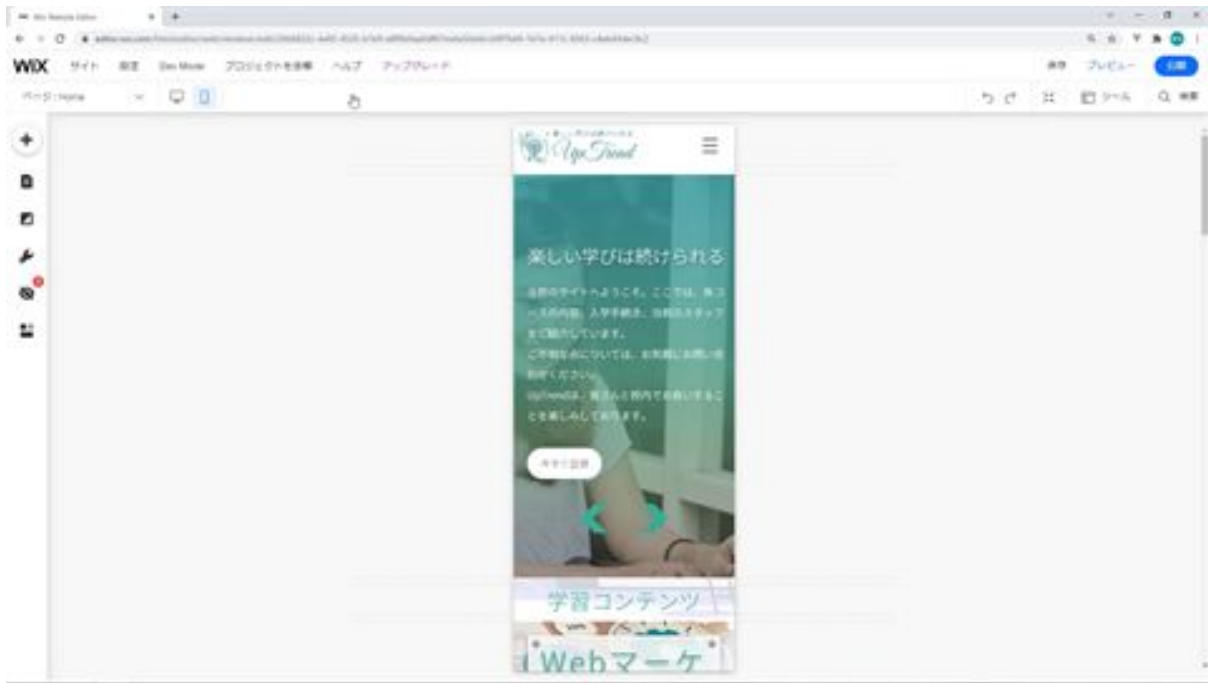
Moving on to the next section, where it says Page: Home, this is because we are managing the pages on this site, and Home is currently displayed, so it is labeled Home. Let's try to display a different page. Click on it and click on another page.



Then the name of another page will be displayed, and other pages can be modified in this way.



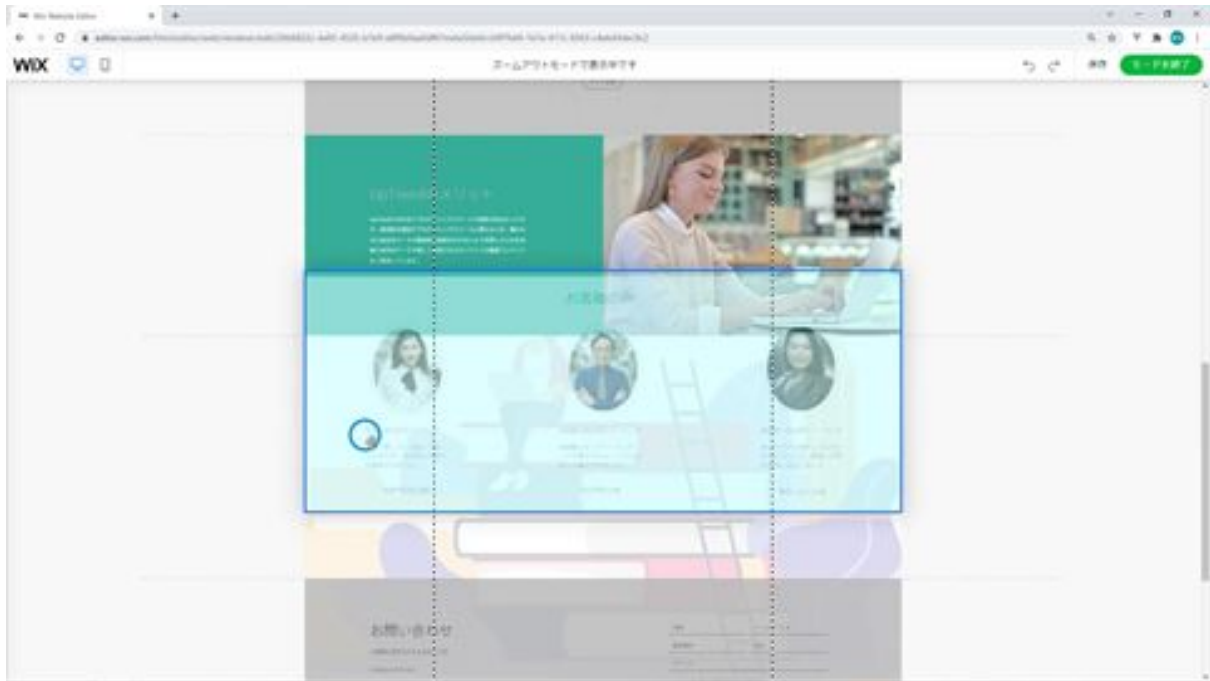
Next, you will see the desktop and mobile editor icons, which allow you to switch between the desktop and mobile screens. When you click on the mobile editor icon, you will be able to edit the site screen for smartphones like this.



The next two buttons, Undo and Redo, are for when you want to undo or redo the changes you have made. For example, if you move a button and want to undo it, press this button to undo it, or press the Redo button to correct the position again.



The next zoom-out sort shows the entire page and allows you to sort each section by its parts. You will see a screen like this, where each section is separated and you can move its position.



The next toolbar allows you to display layers to organize the parts clearly, and a toolbar to change the position of the parts. When you click on the toolbar and the layers, they will appear as shown below. When you click on a layer, a selection is made for each part in the layer.



In the case of toolbars, you can align the parts or move them by specifying their position.



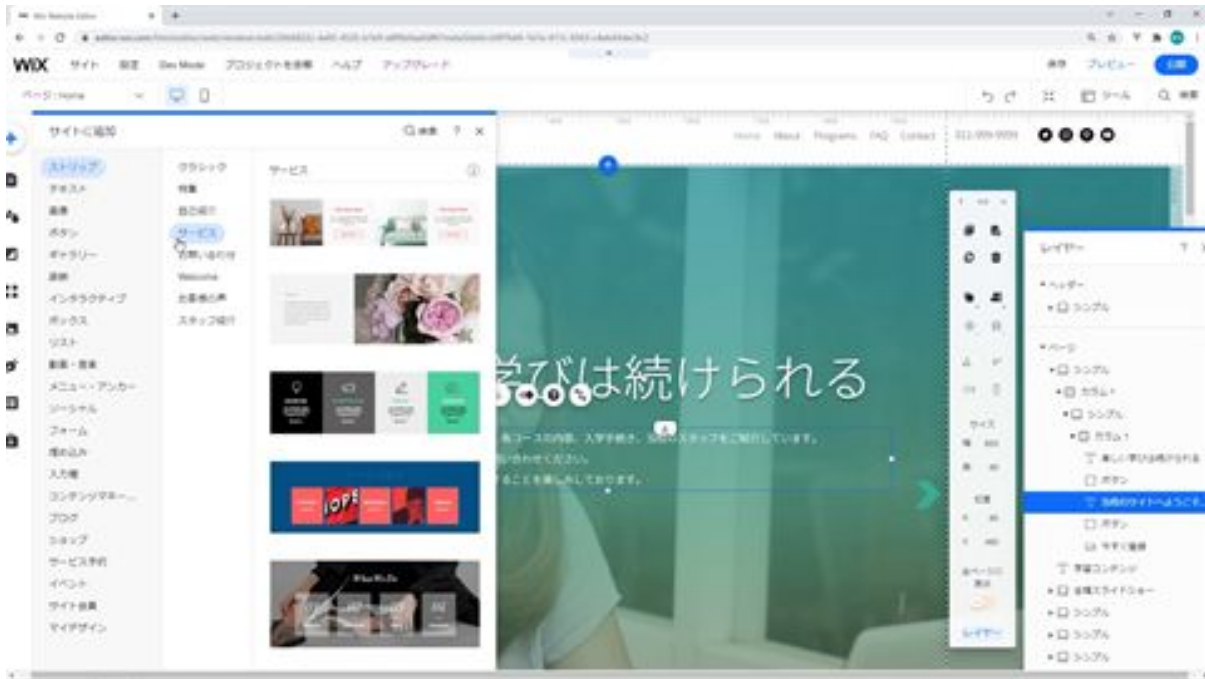
The rulers, gridlines, and snaps will also be displayed. The ruler will show the ruler on the screen, and the grid lines will be the screen area that will be displayed on the tablet or phone.



Next is snapping. Snapping displays guidelines for aligning with other parts. The line in the center indicates that the text above is aligned with the center of the text you are moving.

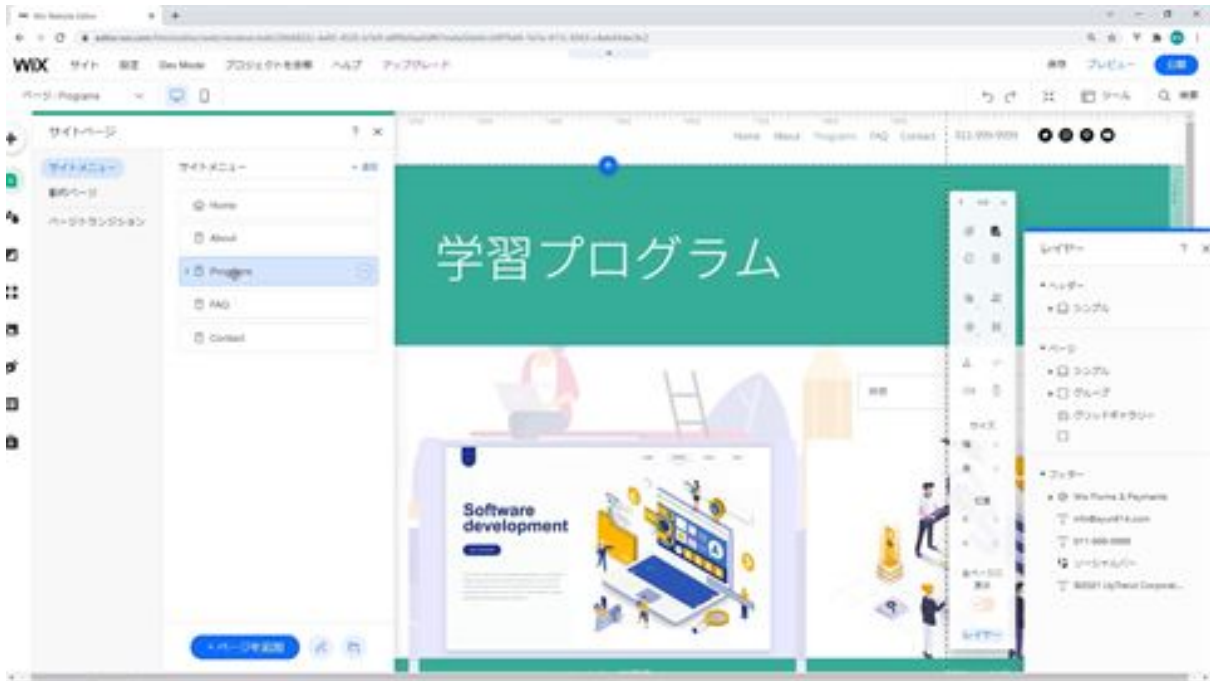


Moving on to the next section, this plus button displays a list of parts, and you can add these to your page by clicking or dragging and dropping them. There are various parts such as strips, text images, buttons, galleries, and decorations. You can also easily add forms, social bars, HTML embedding, and more.



The next menu is Menu & Page, which works in a similar way to the one above, but you can switch pages by selecting them, and you can switch the site menu in the same way in Menu & Page.

You will see a list of screens like this, where anchors, menus, etc. are displayed. When I clicked on it, the page switched.



Next is the site design, which allows you to change the site style, page background, and page transitions.



Clicking on the Edit button in the site style will bring up a screen like this, where you can change the color and text. Clicking on the color in the site

style will change the color scheme of the site, and clicking on the text will change the font type and size.



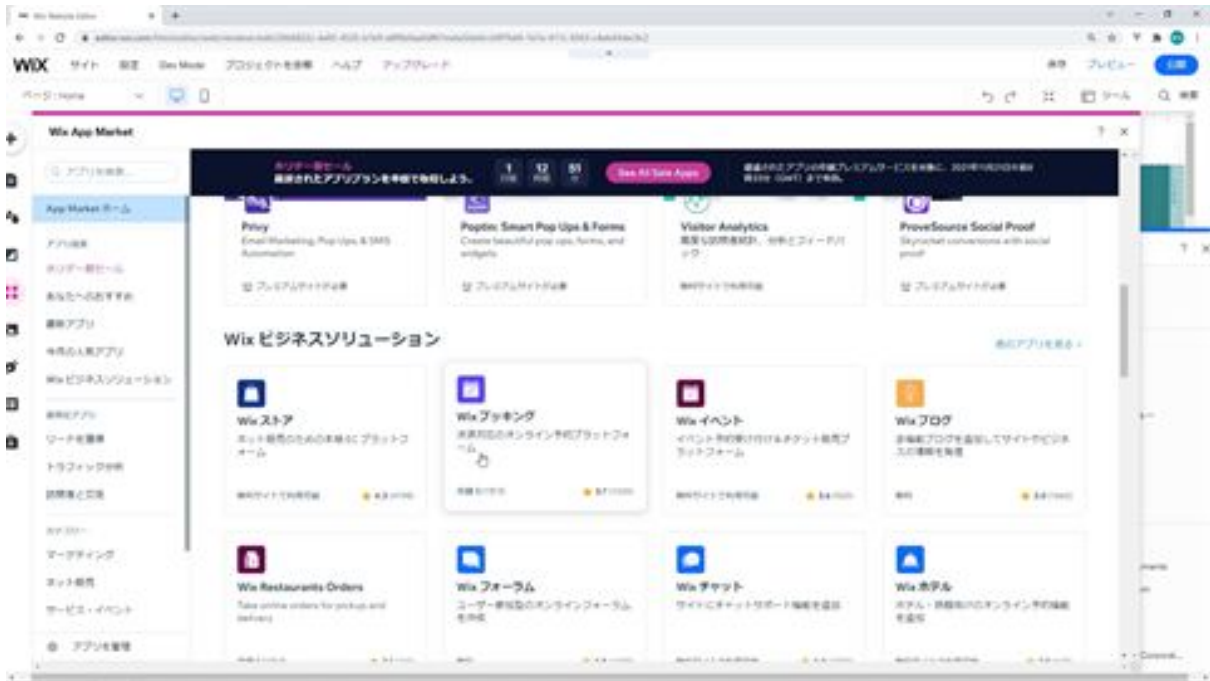
By clicking the Edit Page Background button, you can change the page background of your site to a single color, image, or video.



Page transitions allow you to change the type of animation that is displayed when a page is loaded. The easiest one to use is fade-in. When you fade-in, the screen will appear fluently. You can also swipe vertically, horizontally, and even out.



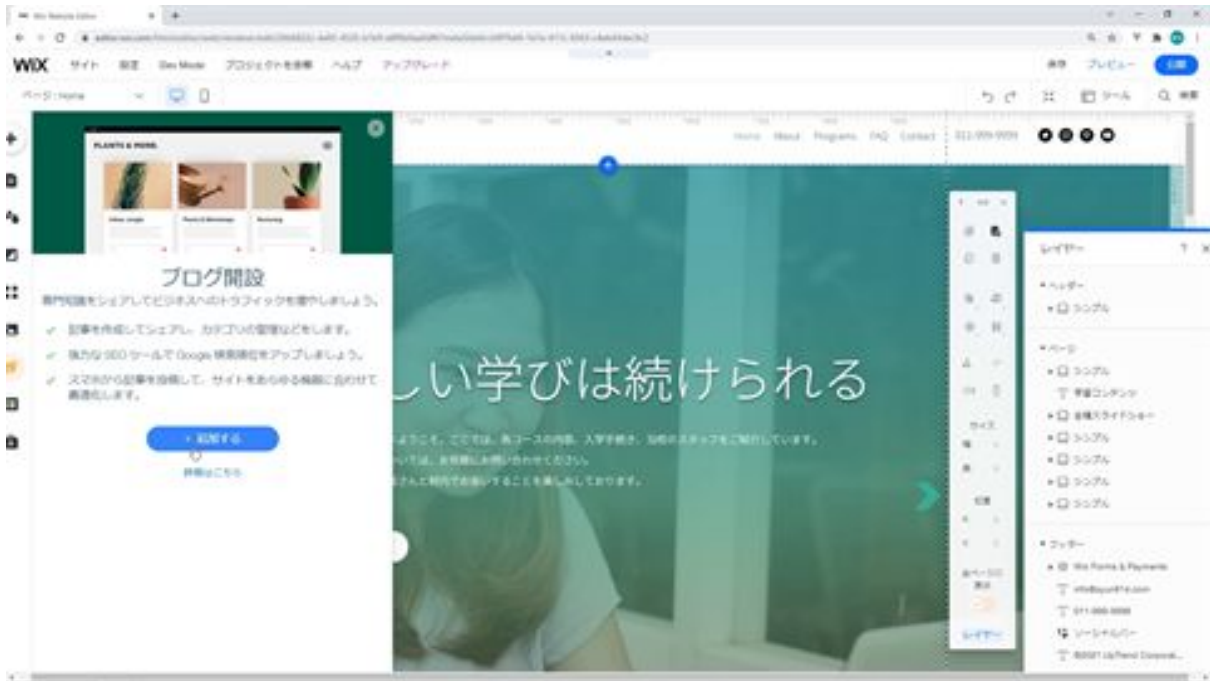
Next are apps, which are systems that can be easily integrated into a website. For example, the Wix Store allows you to easily create a mail-order page, and Wix Booking allows you to easily create a reservation site. These various functions are created under the name of apps.



Next is media, which allows you to upload a variety of files.



Next is Blog, which is a menu that incorporates blog functions. You can set up a blog from this Add button.



After that, there is the Content Manager, where you can create a contact form, a database, and so on. You can create a database by clicking the Create Collection button here.



Next is Ascend, which is a business tool that provides marketing, SEO, customer management, accounting, access analysis, and reports.



Next, click on the header at the top of the page, and you will see the Change Header Design button and the Settings button, where you can change the header design and settings.

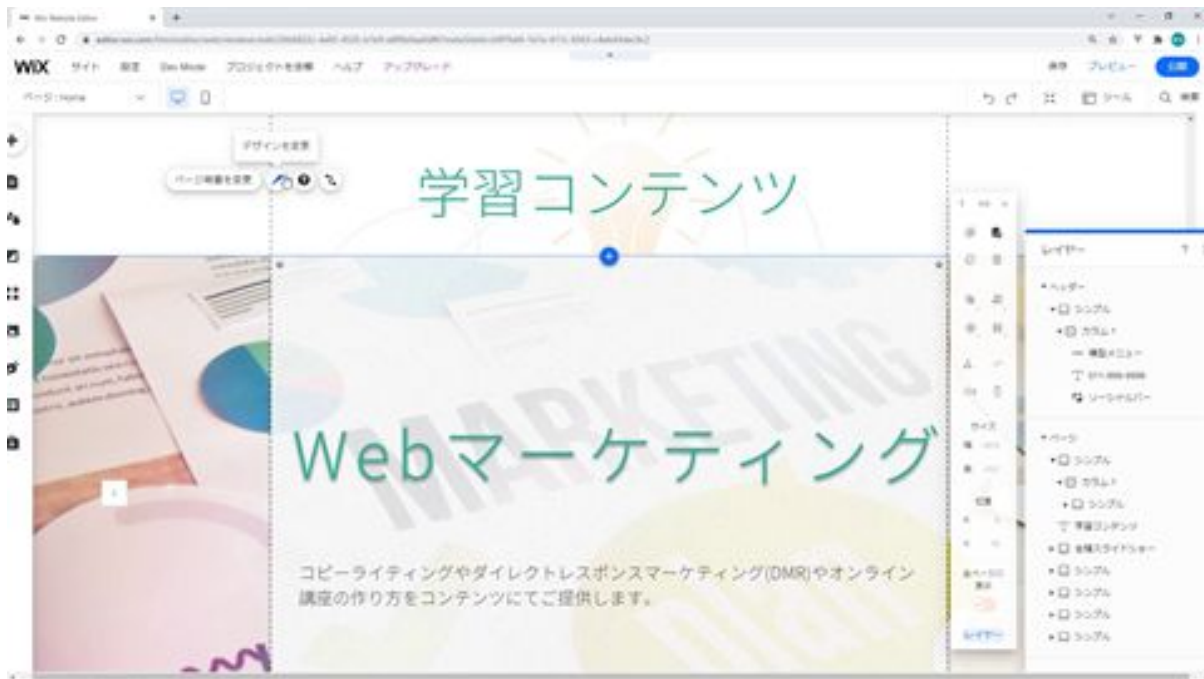


Similarly, in the footer at the bottom of the page, you will see a button to change the design and a button to change the settings as well.



And the area between the header and footer is the main page of the website, where you can click on the page background button and the design button to

change the background and design.



And this main page will be the place where each page will be created. If you want to modify the position of each component, you can display the toolbar from the Tools menu and align it or specify the position.

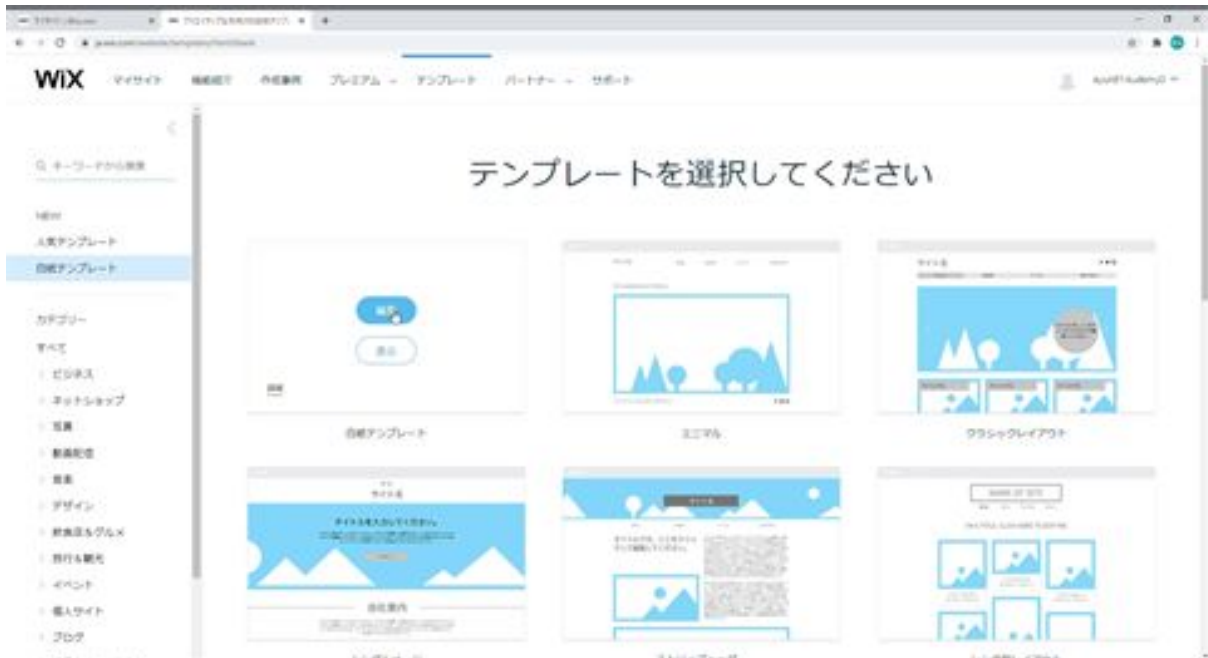
You can also change the design and settings in this way. I think it's best to try it out first to get a feel for what's there. These are the basic functions of the editing screen.

Basic operation and explanation

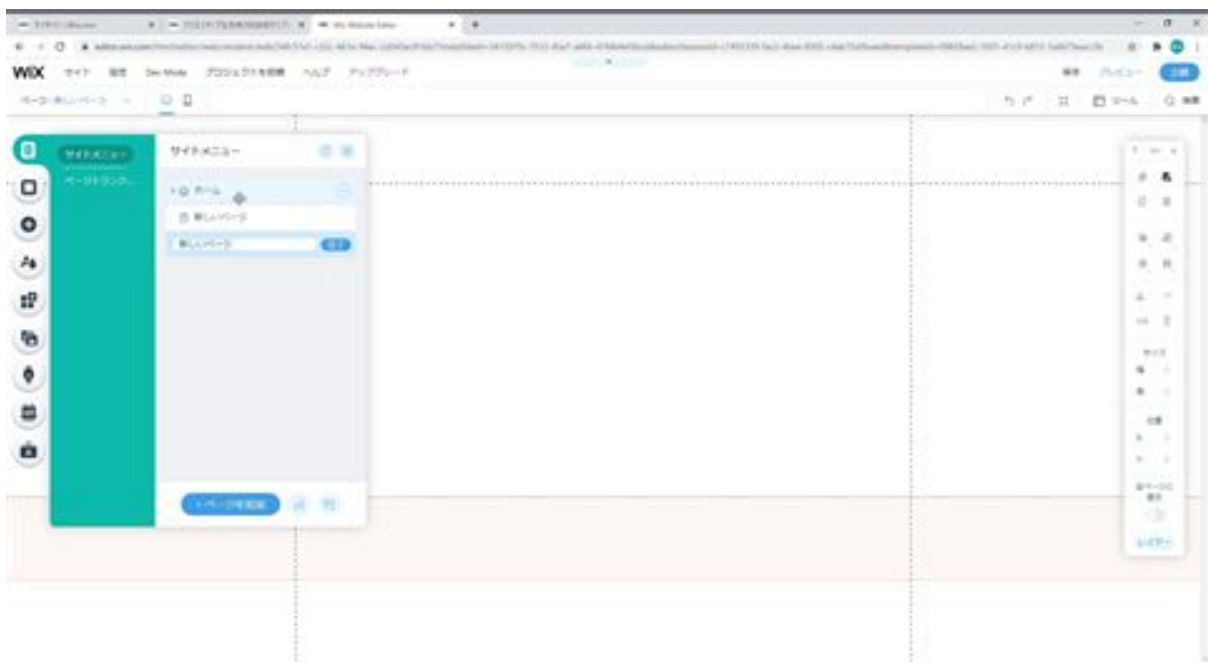
Now let's understand the basic usage of Wix. First, click on Create a new site in the upper right corner.



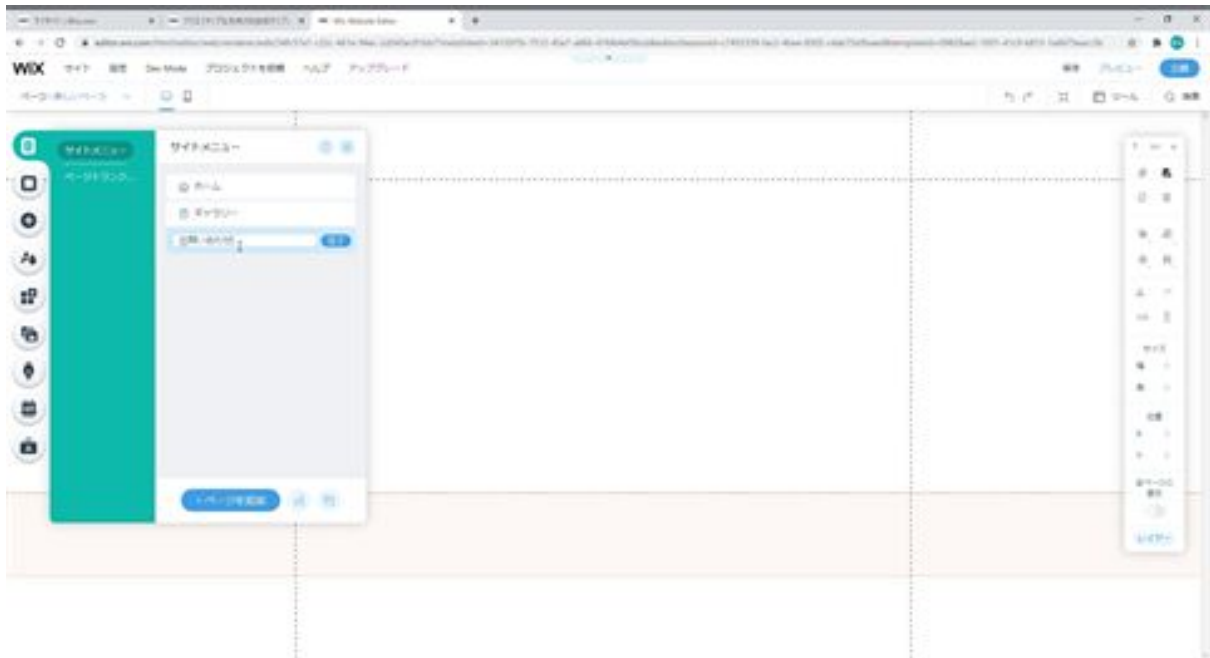
Then click on Other and click on Choose Template. Click the blank template on the left and click Edit.



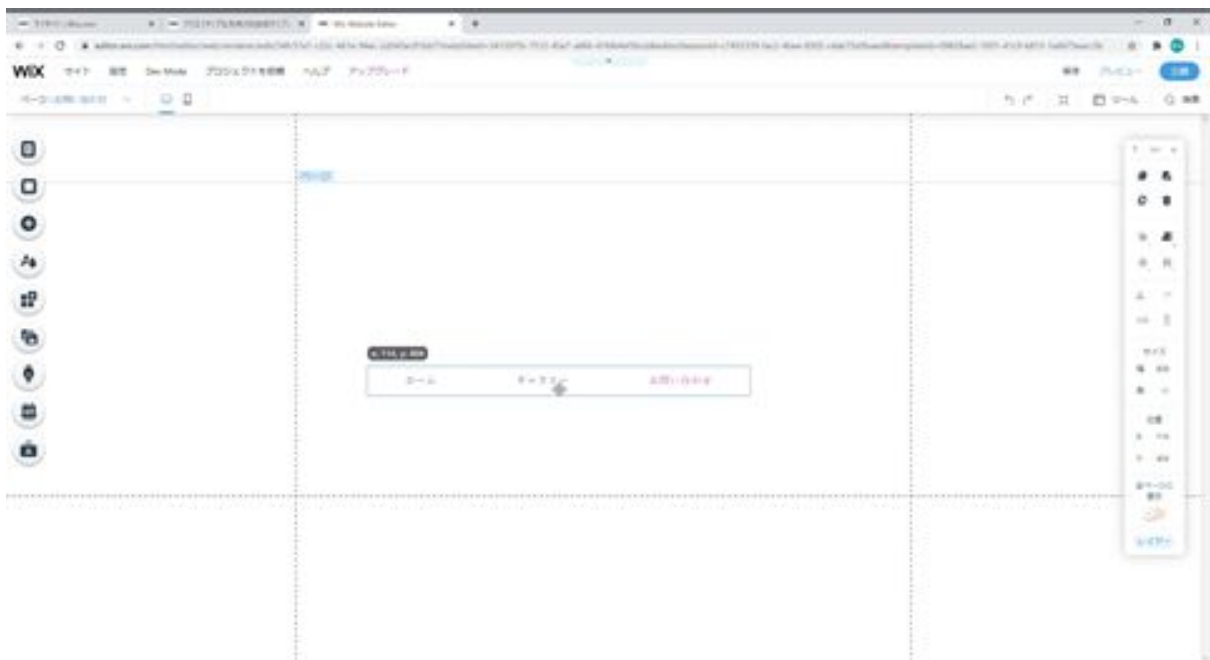
First, we would like to add a menu. Click on Menu & Pages and enter the page you want to add. Click on Add Page at the bottom to add a page.



In this section, we will add a page called Gallery and a page called Contact Us.

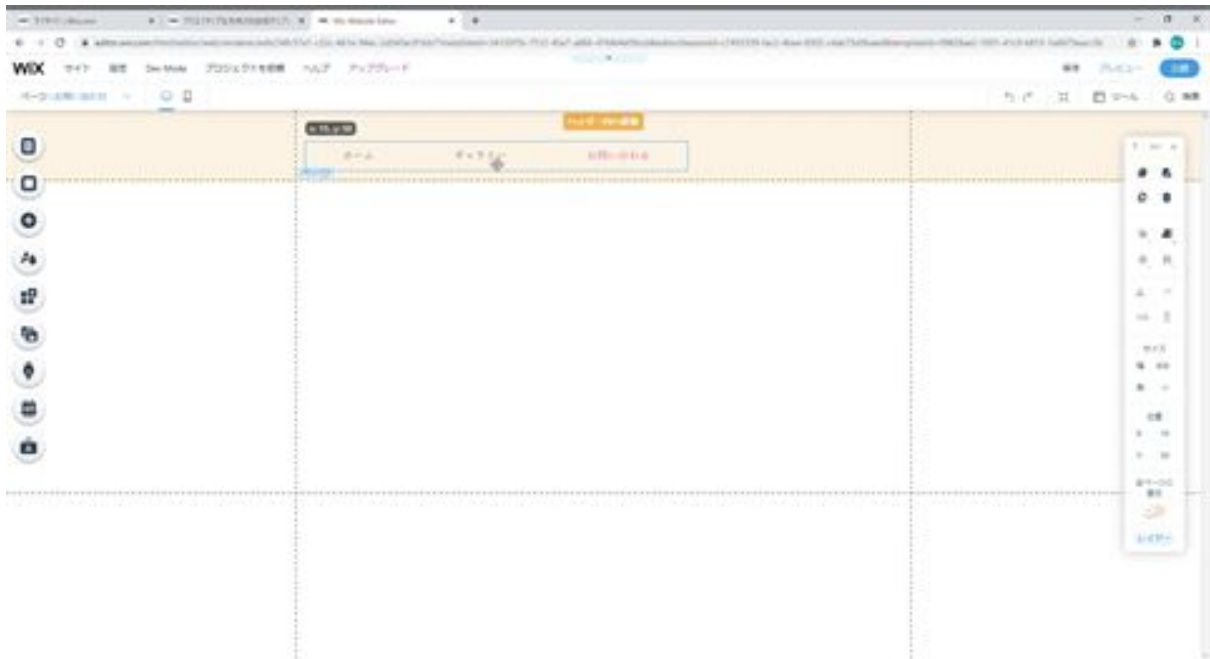


You can add pages, so let's create the menu first. Click on the Add button, and then click on Menu. There are a variety of designs to choose from, but choose the design you want to add and click on it to add the menu.

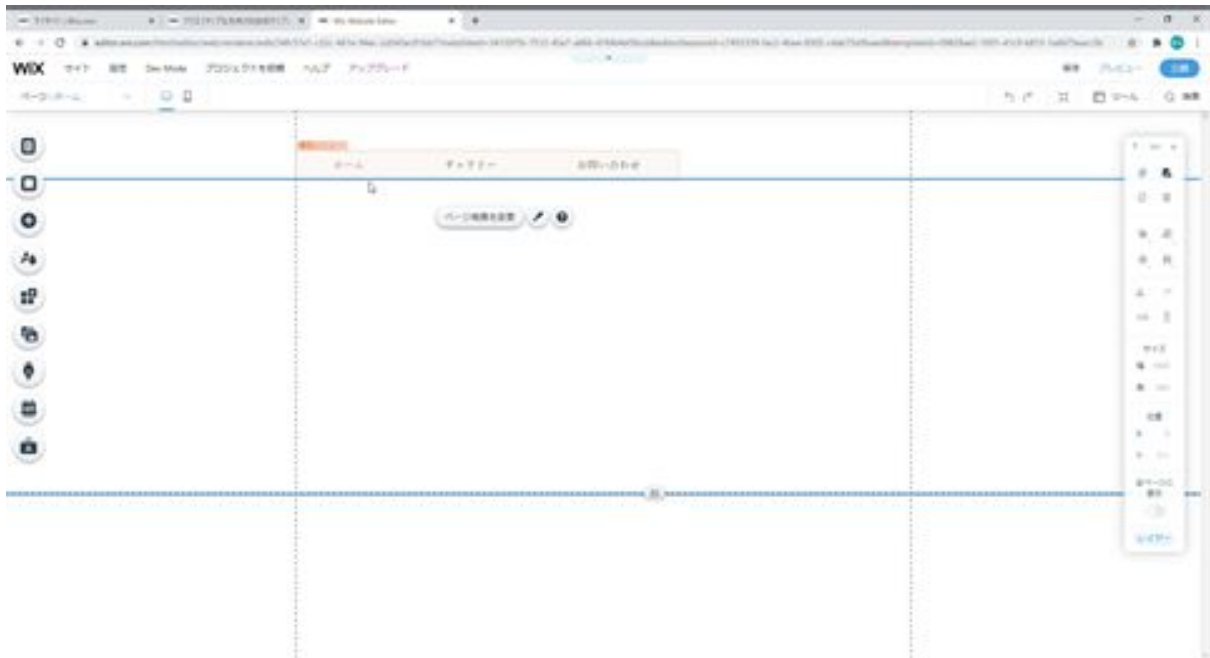


I'd like to bring the added menu up to the header. If you bring it up to the top, you will see the "Move to Header" option, so you can move it to the

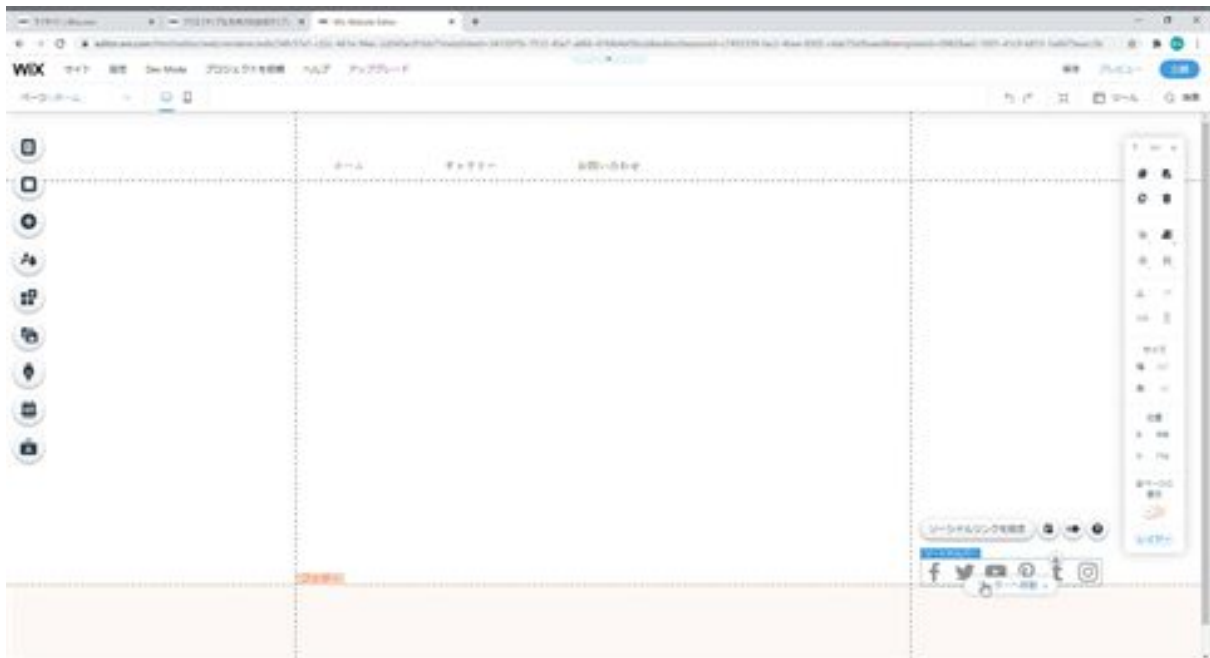
header here.



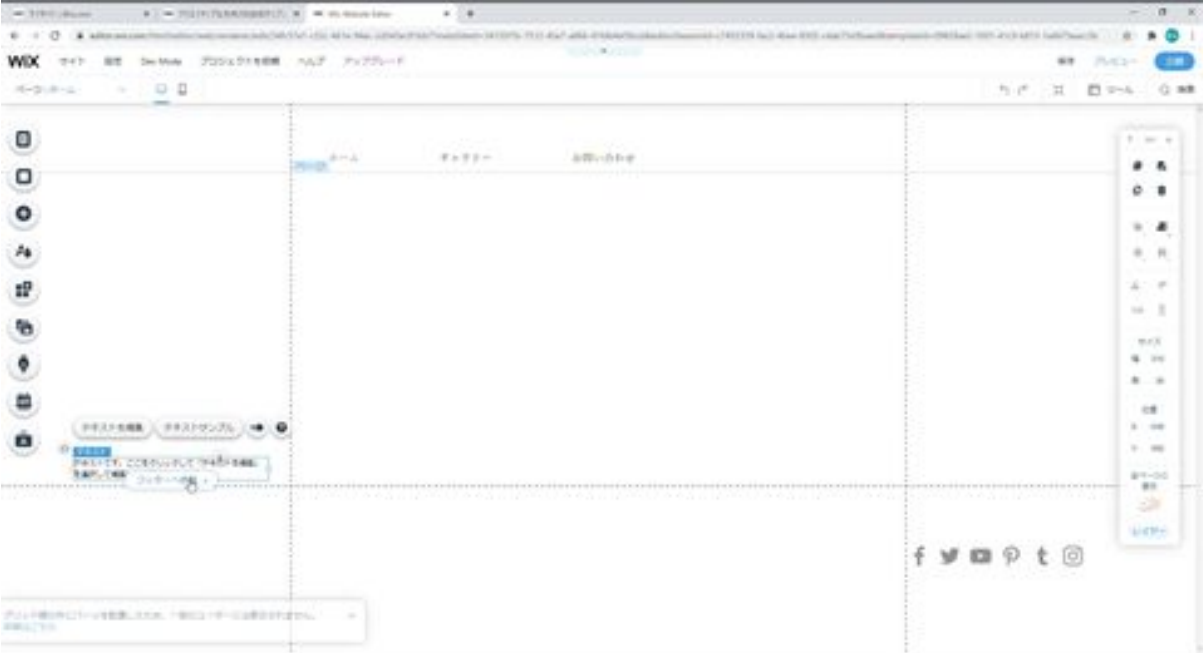
When you move it, move it in line with the guideline. We have now completed adding the menu. Let's see if it is reflected in the other pages. If you click on Home and Gallery in the Menu & Pages section, you will see that the color of the currently displayed menu has changed. The menu has now been added.



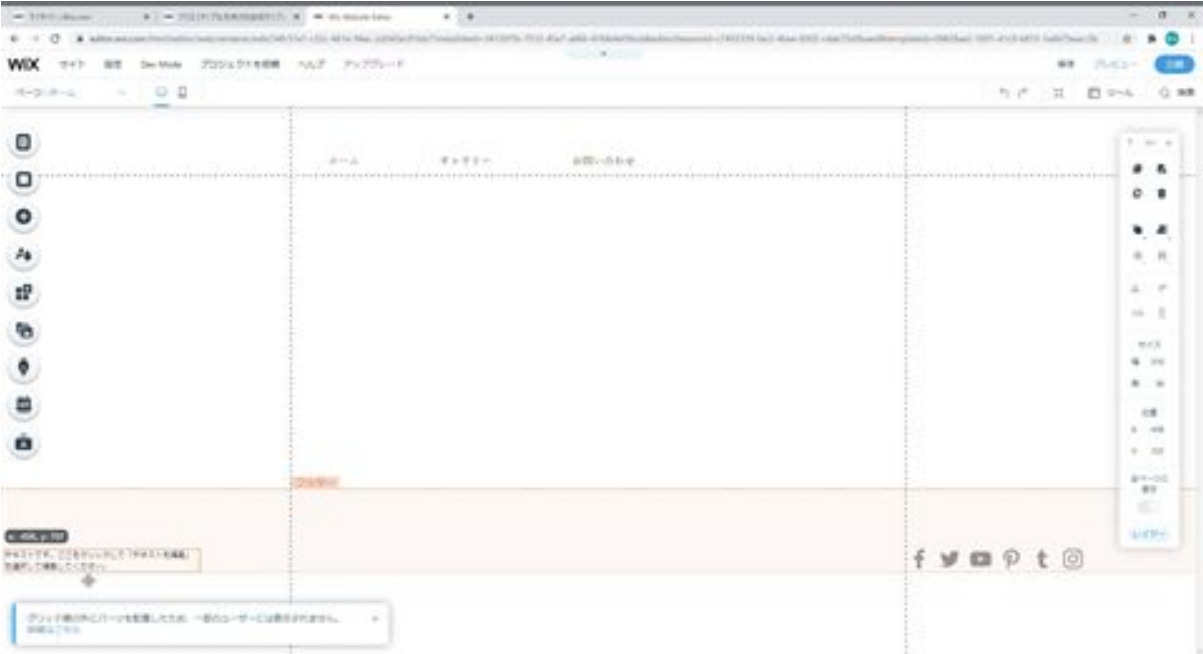
Next, I would like to add the footer. We will add the social bar using the add button. Select Social Bar from Social and move it to the footer. When you move it to the footer, click on the "Move to footer" button because it will not move to the footer. Then the social bar is moved to the footer. This will also move it along the guideline.



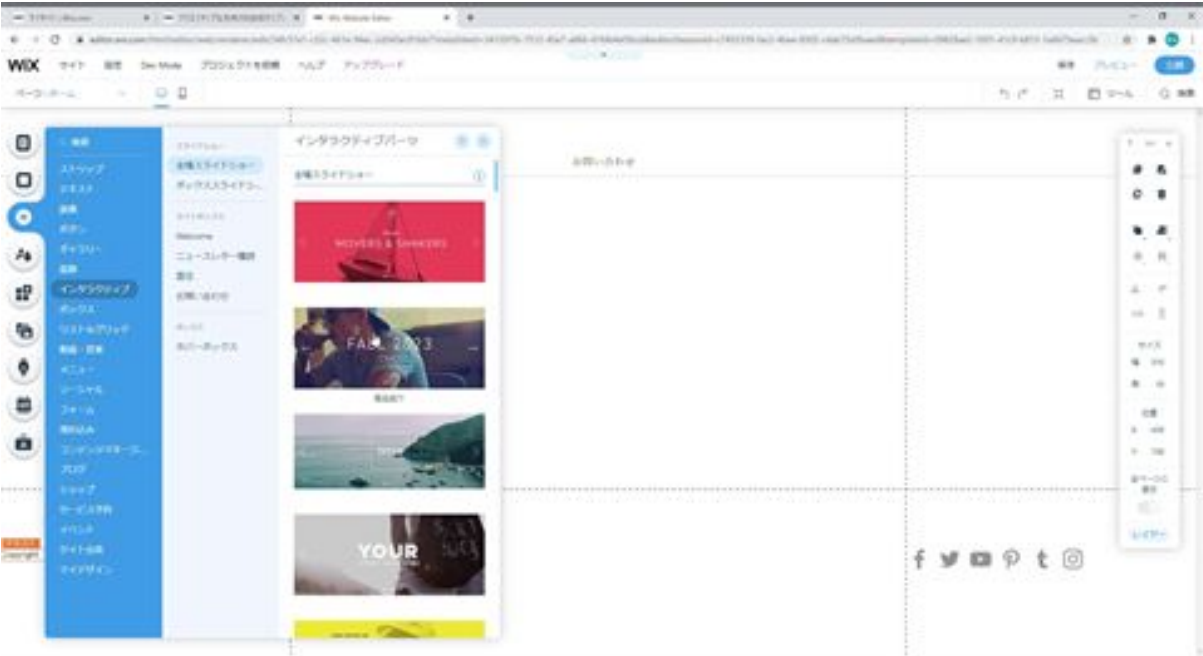
The next step is to add the copyright text. If we select the text and move it, the social bar will be moved along with it, so we'll move it by pressing Move to Footer.



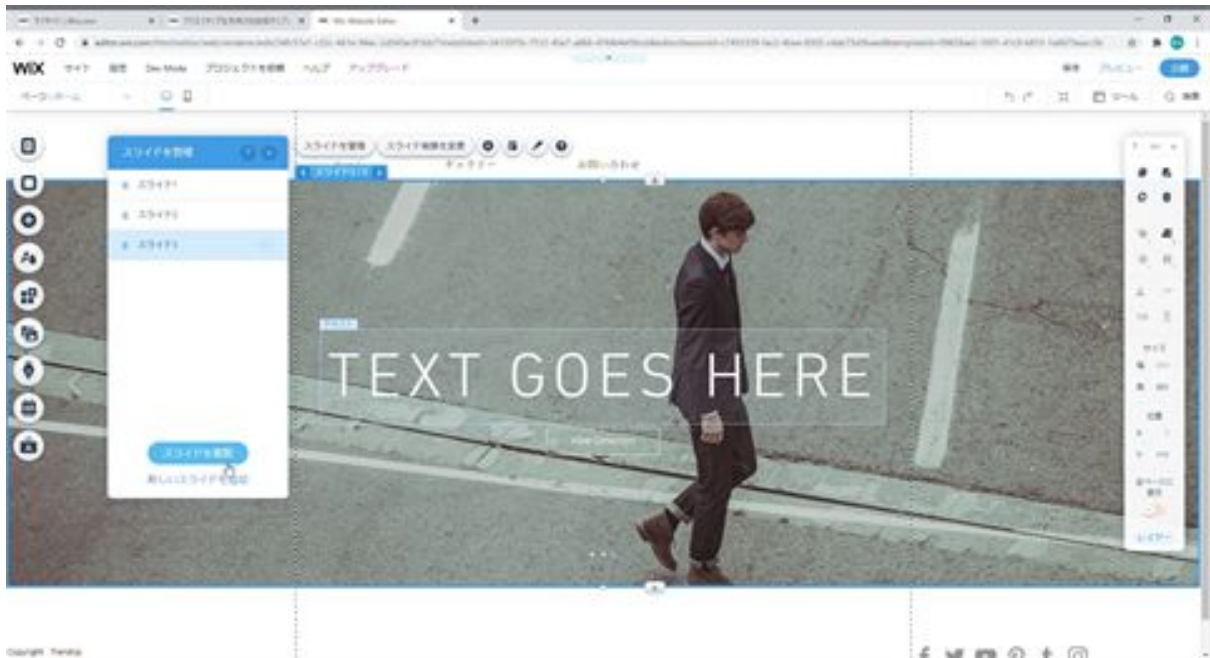
I'm going to go ahead and enter the copyright so that it follows the guideline.



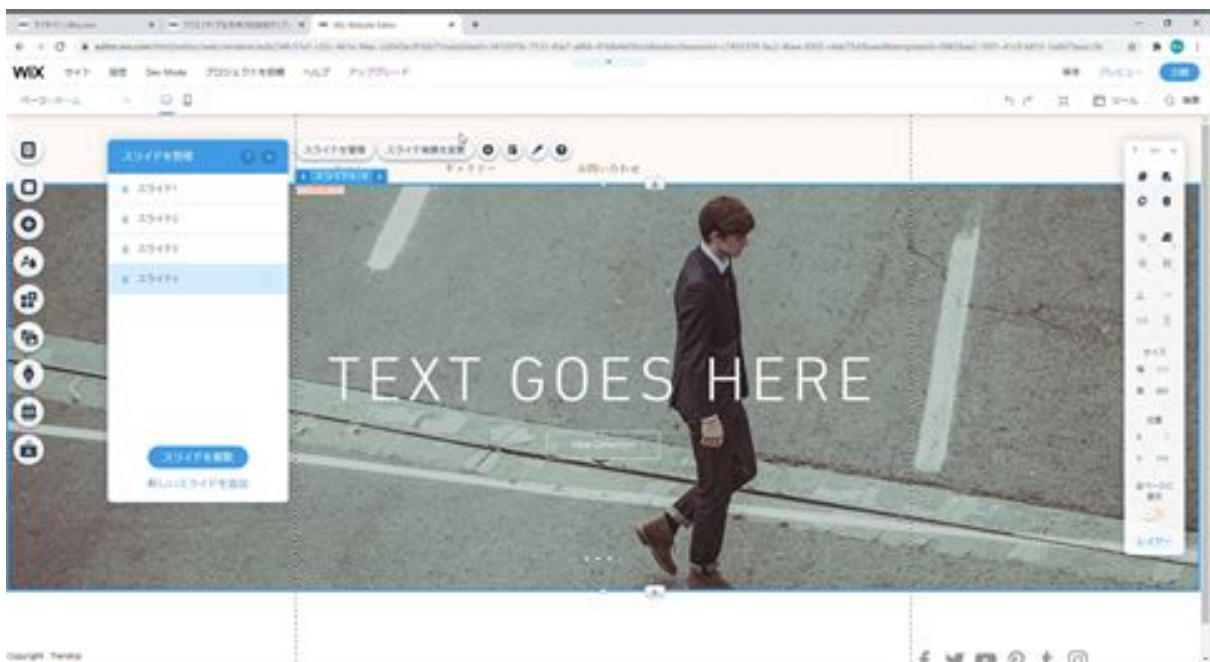
Finally, we will add a slideshow to the main page. Select Slideshow from Interactive.



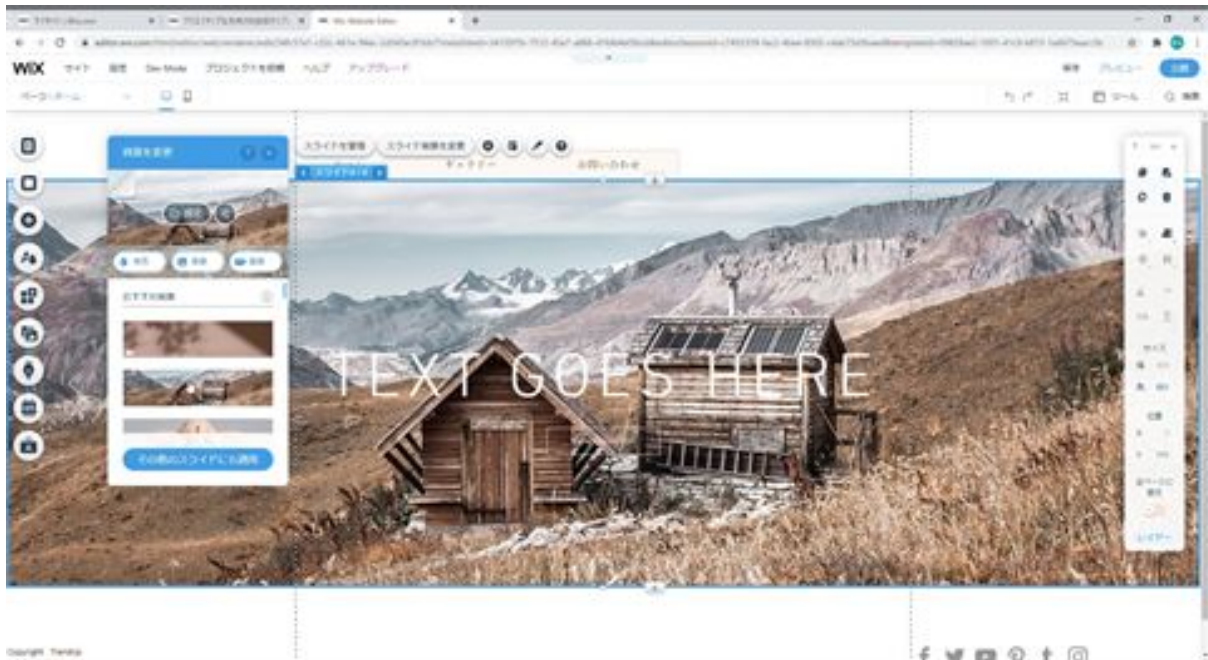
A slideshow has been added. Here is the slideshow, and three slides have been added. I'm going to add another slideshow here, the fourth one. You can add a new slide here from Manage Slides or Duplicate Slide. Here we will add a slide from Duplicate Slide.



We will also change the name to Slide 4. Now we want to change the image on slide 4. With slide 4 selected, click Change Slide Background.



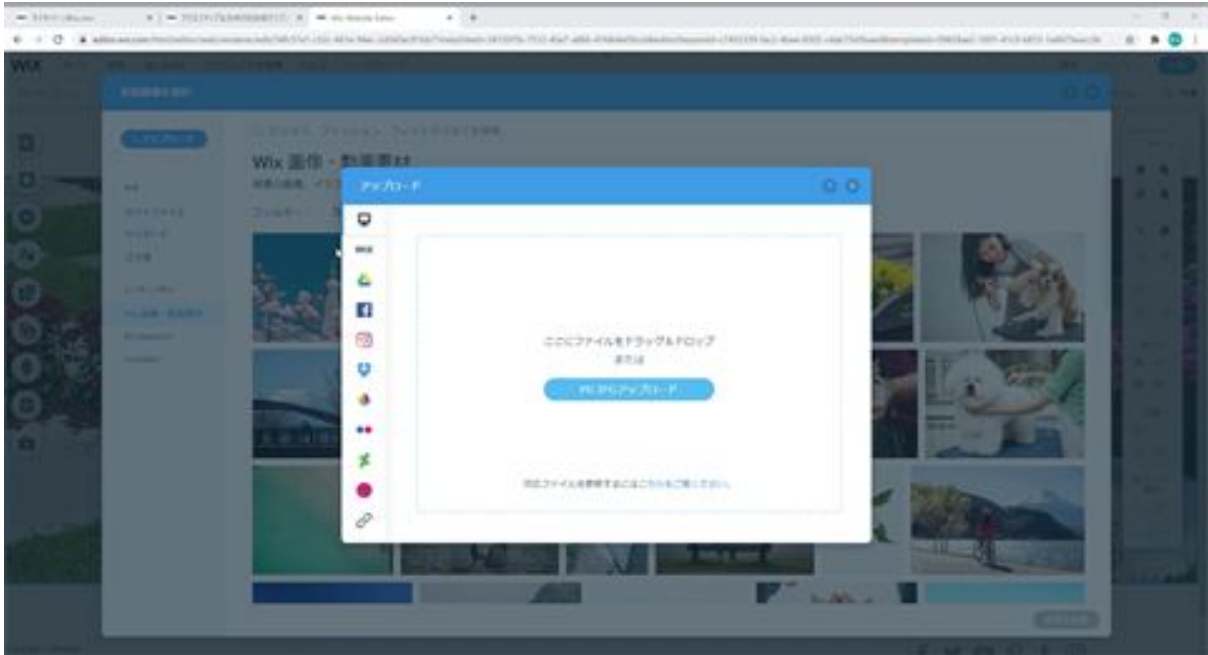
If you select an image here, the image on the slide has been changed.



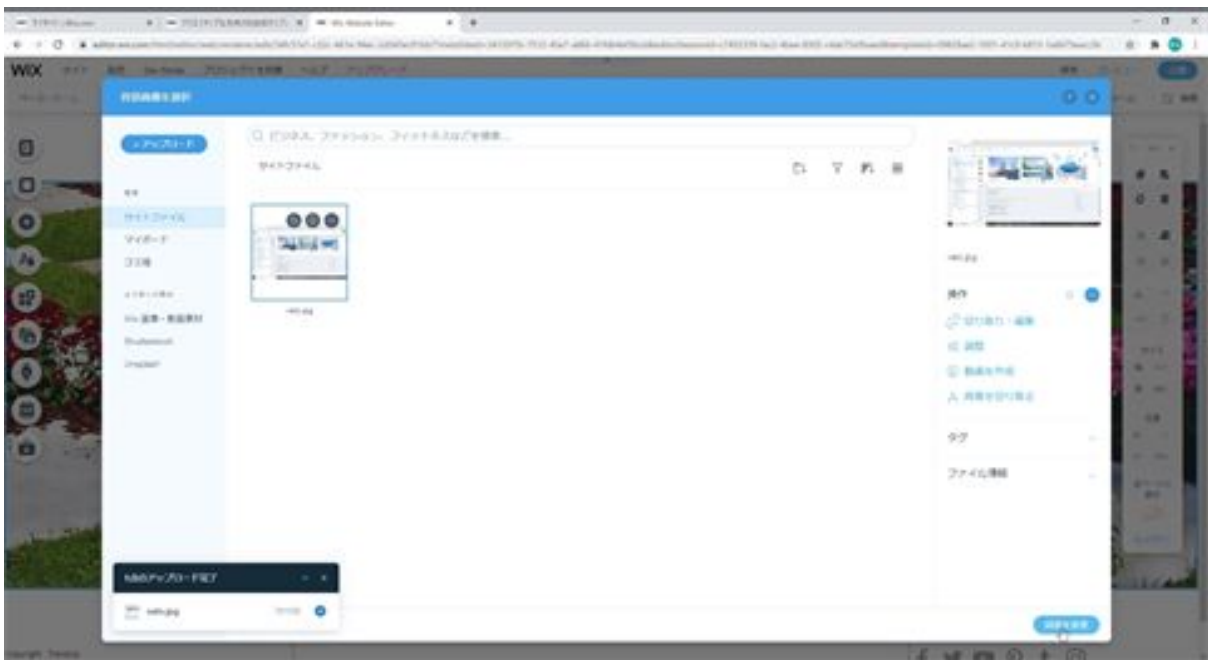
Alternatively, you can click the Image button, select an image from this list, and click Change Background to change the image.



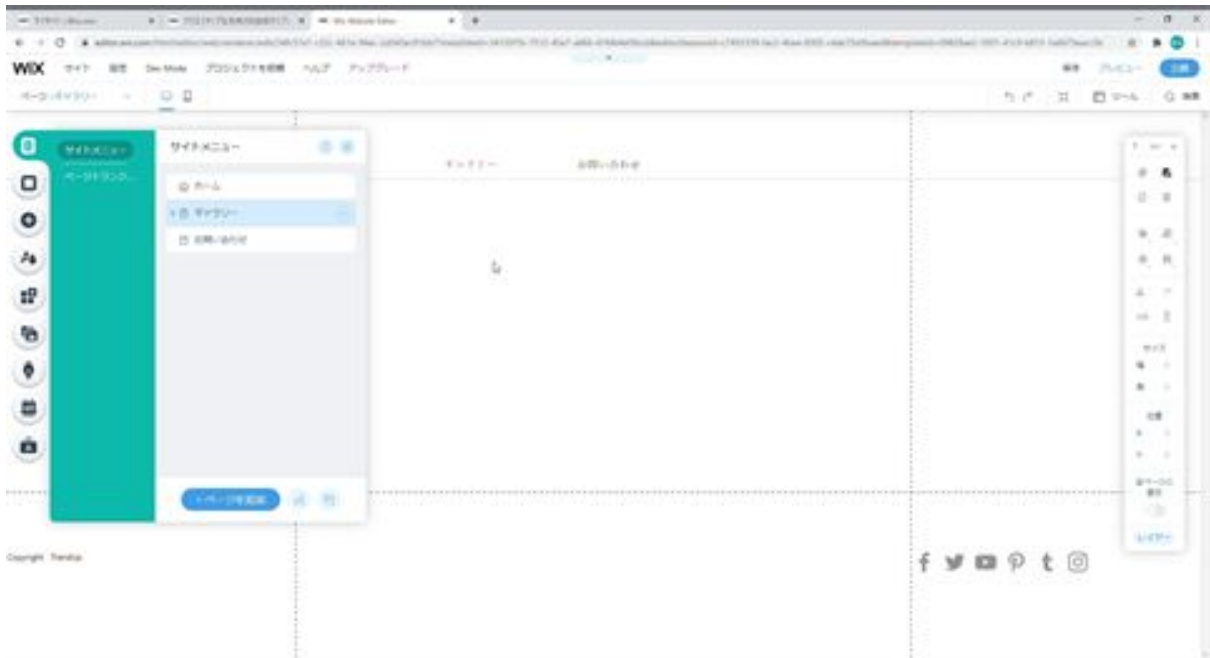
And if you want to upload your image and have it selected, you can add it by clicking on the upload button and selecting the image.



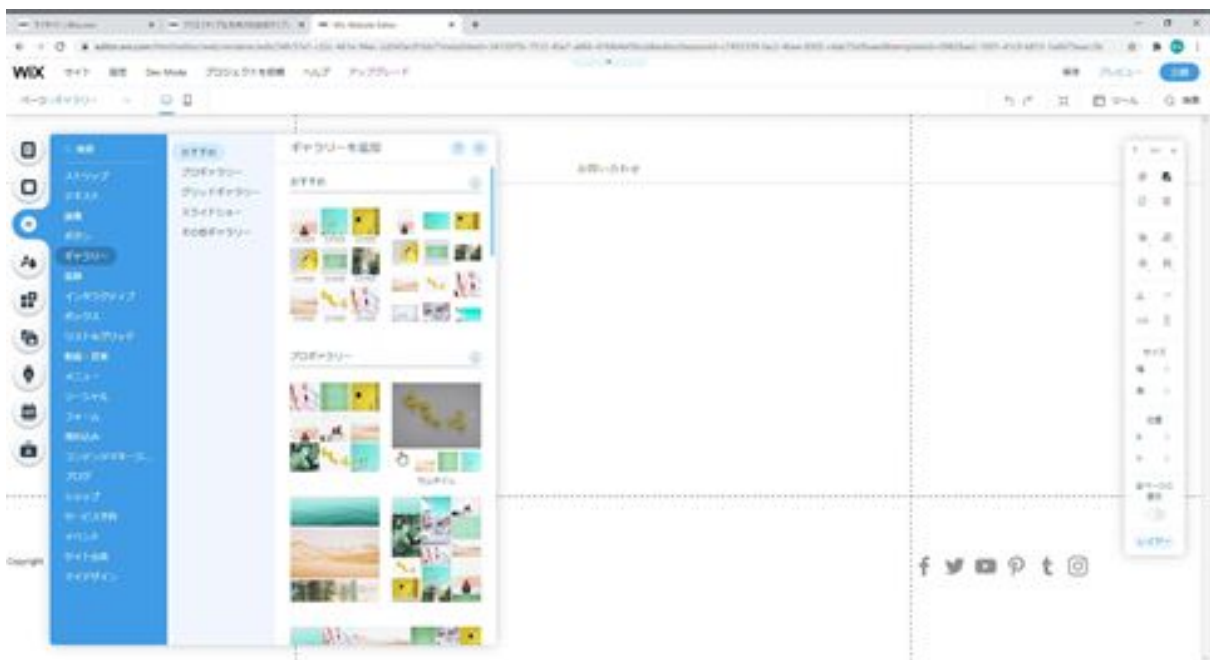
When you do so, the image will be uploaded. Select the uploaded image and set the image to change the image.



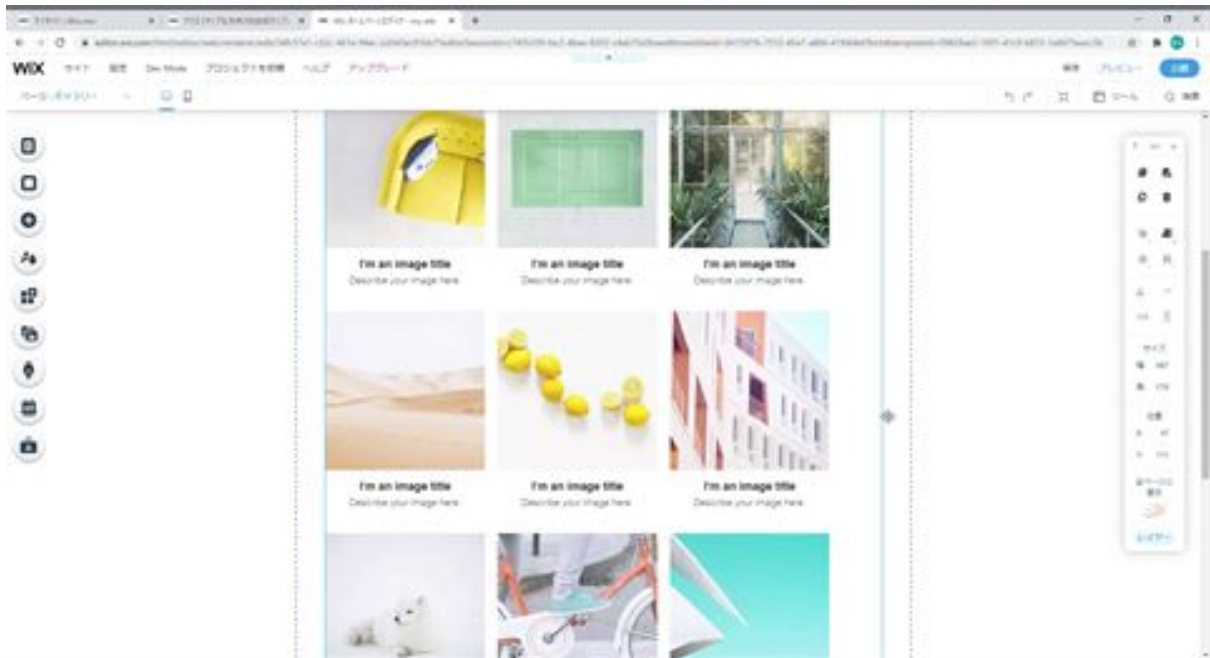
Now we will modify the gallery. Select Gallery from the Menu & Pages on the left.



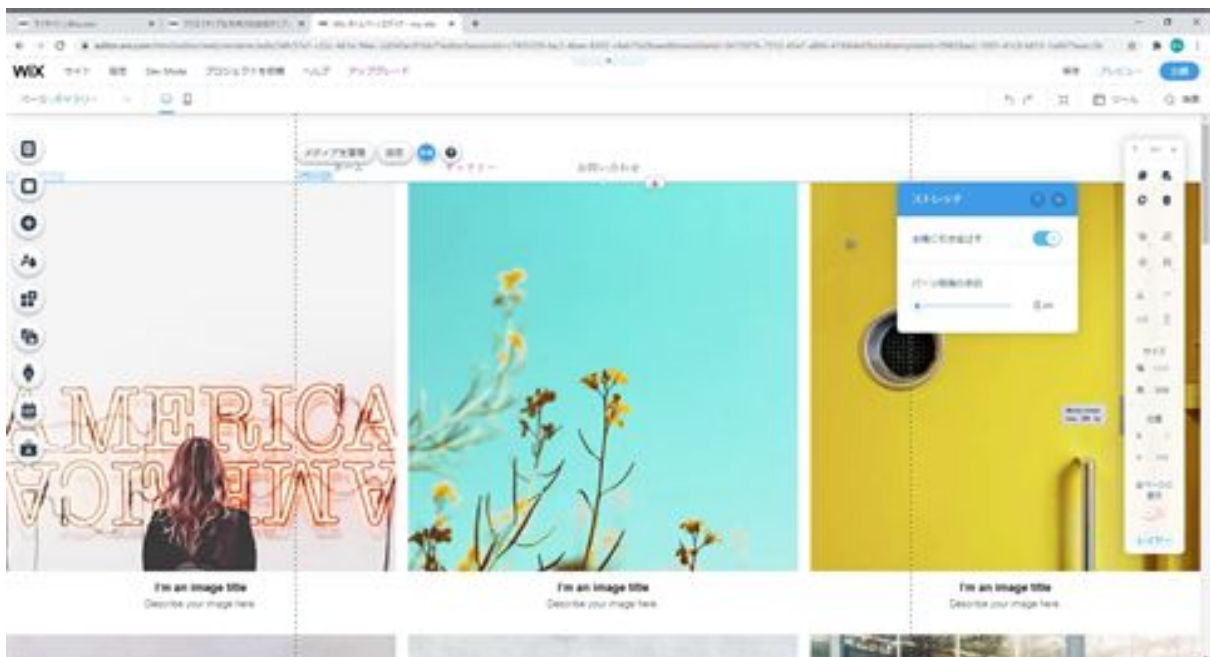
We will add a gallery in the center section here. Select a gallery from the Add button and click on it.



This is the gallery that has been added, and you can change the width and height like this.

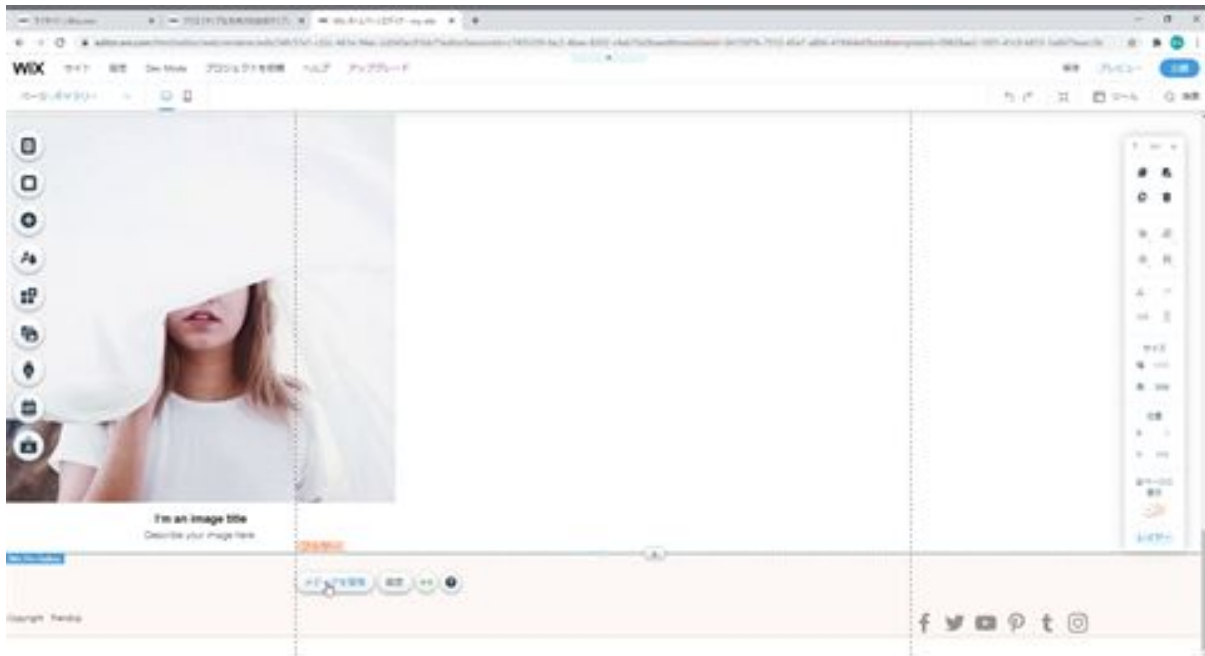


Then there is stretching, which can be stretched to cover the entire screen by clicking on the stretch button. The margin of this part can also be changed by sliding it. In this case, I would like to set it to 0px.

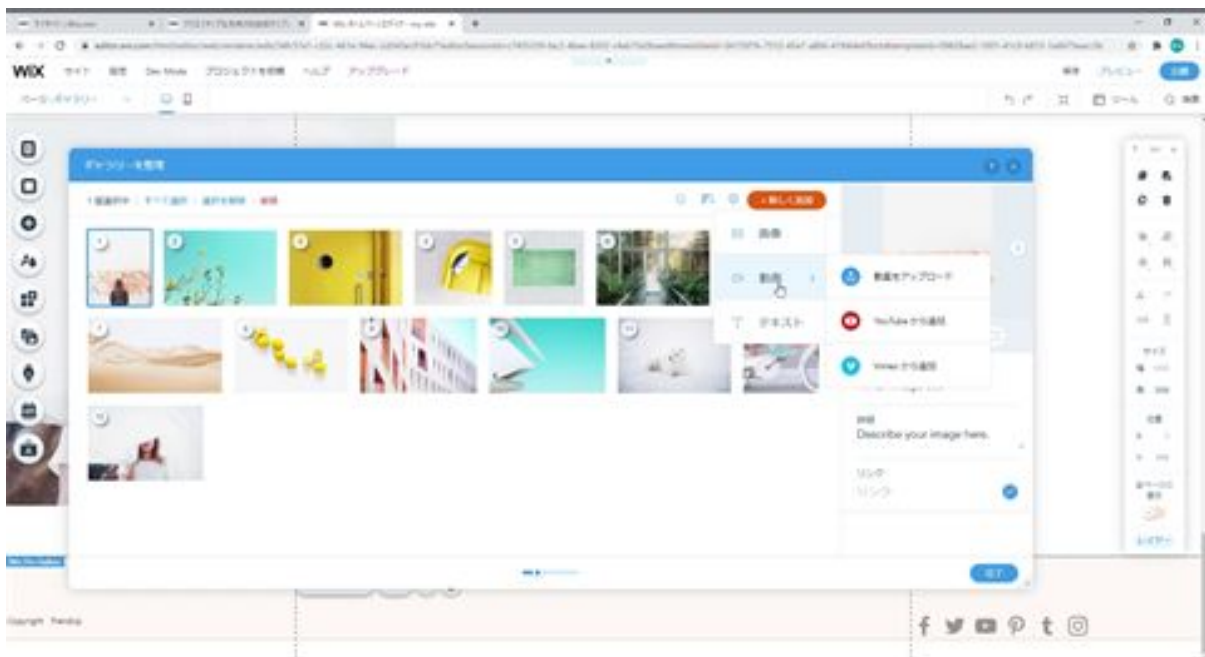


If you want to add images to the gallery or modify the gallery, you can click on Media Management to see the images in the gallery, and you can change

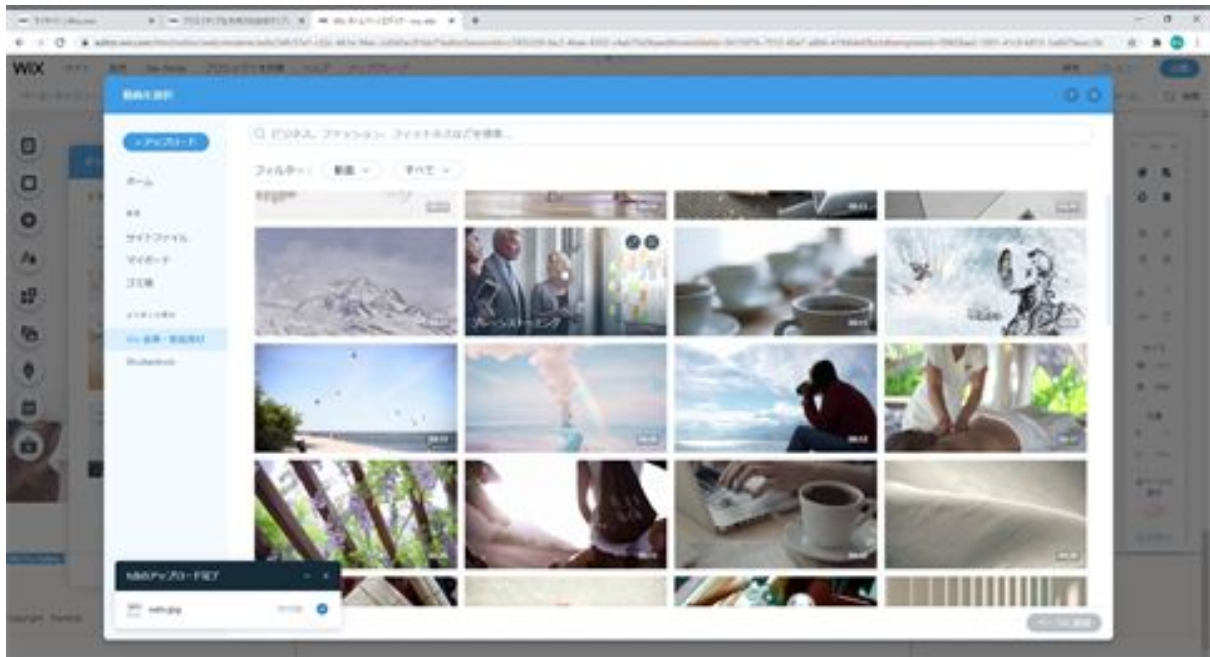
the order by replacing them here.



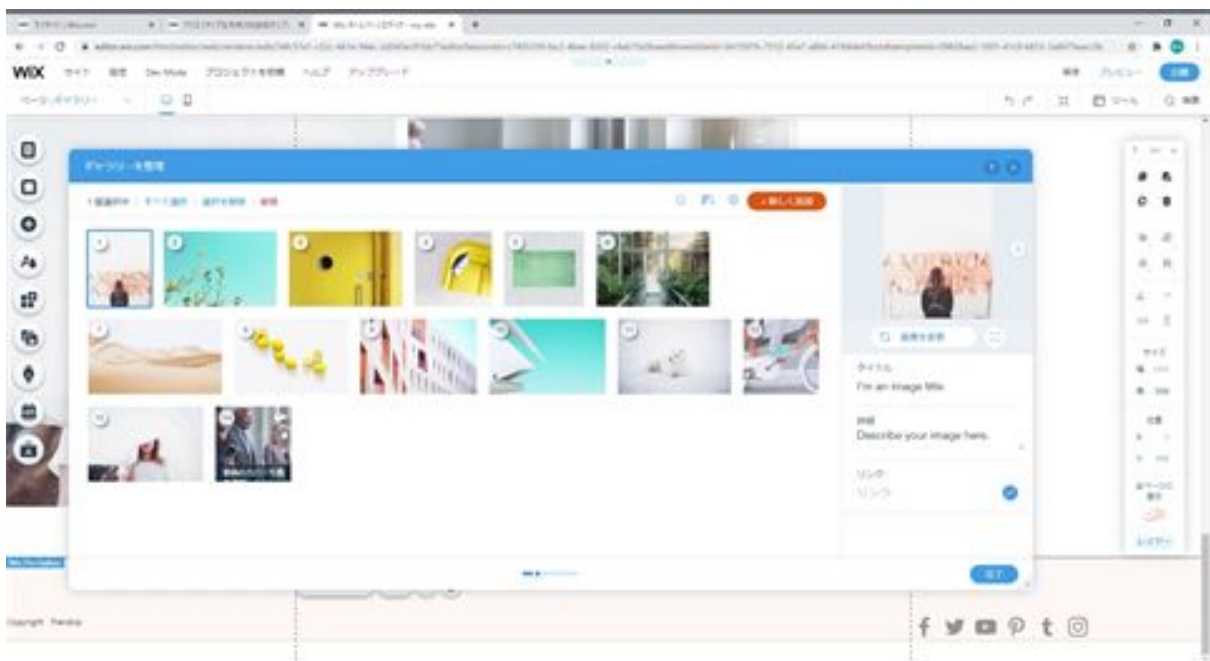
Or you can click on the Add New button to add an image or video. In this case, we would like to add a video.



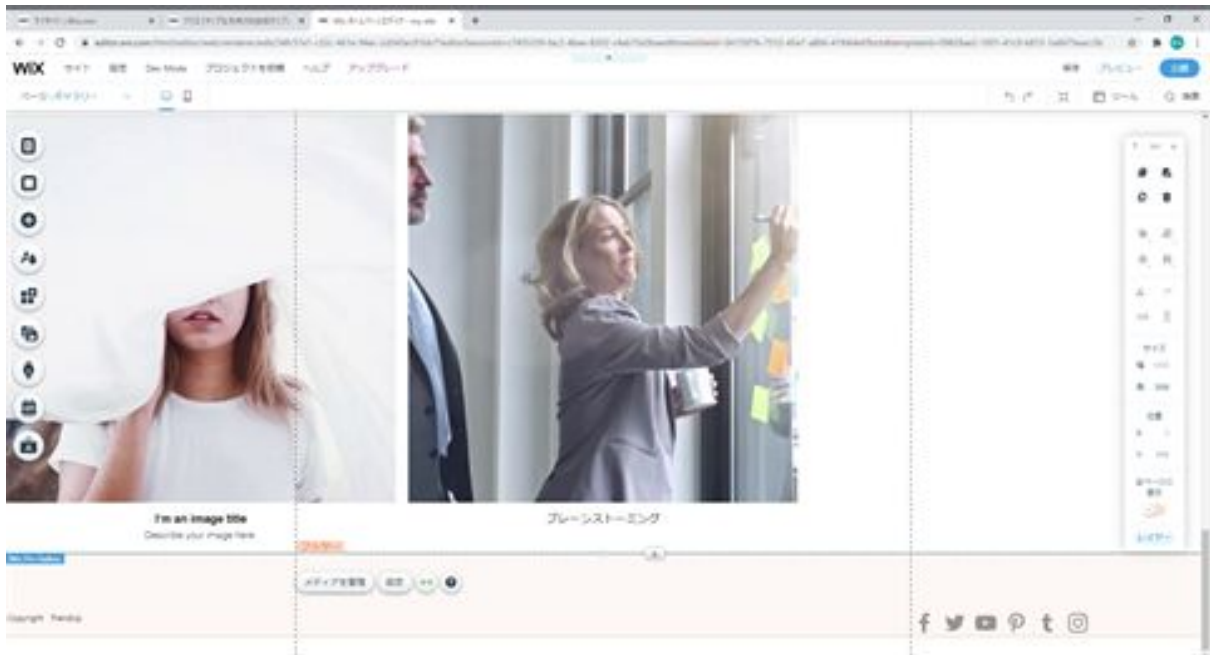
Click Upload Video from Video to see the Wix image and video materials, and add your video here.



When you add it to the page, the video will be added to the end.

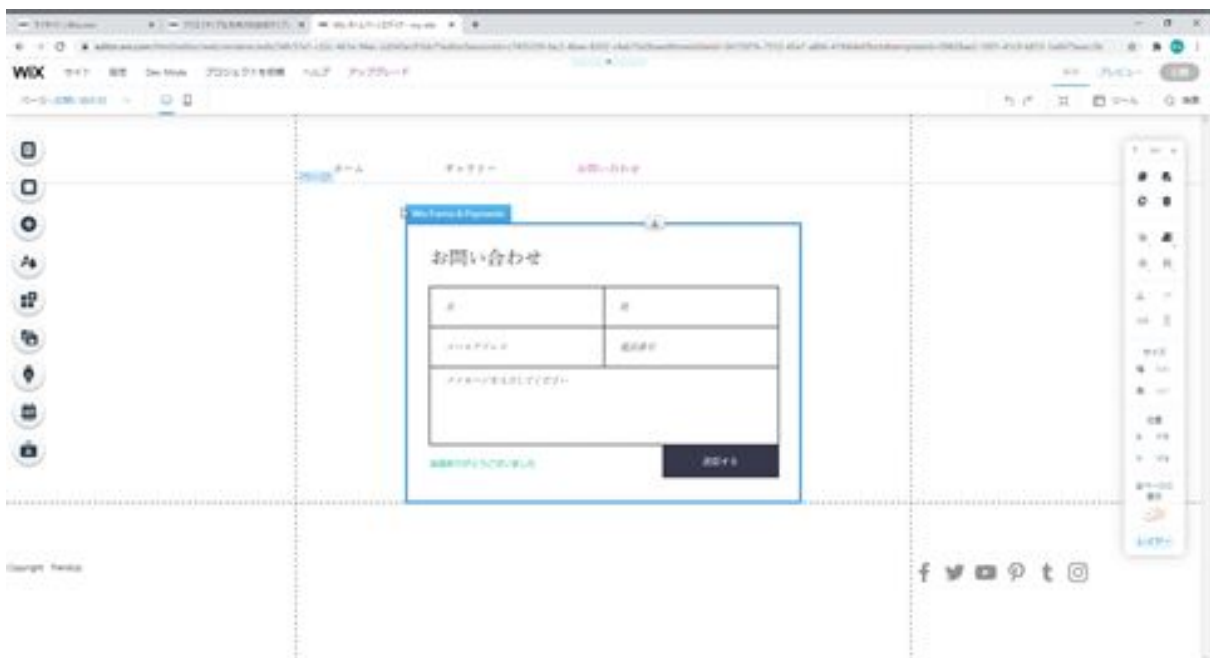


Now click Done to add the video.

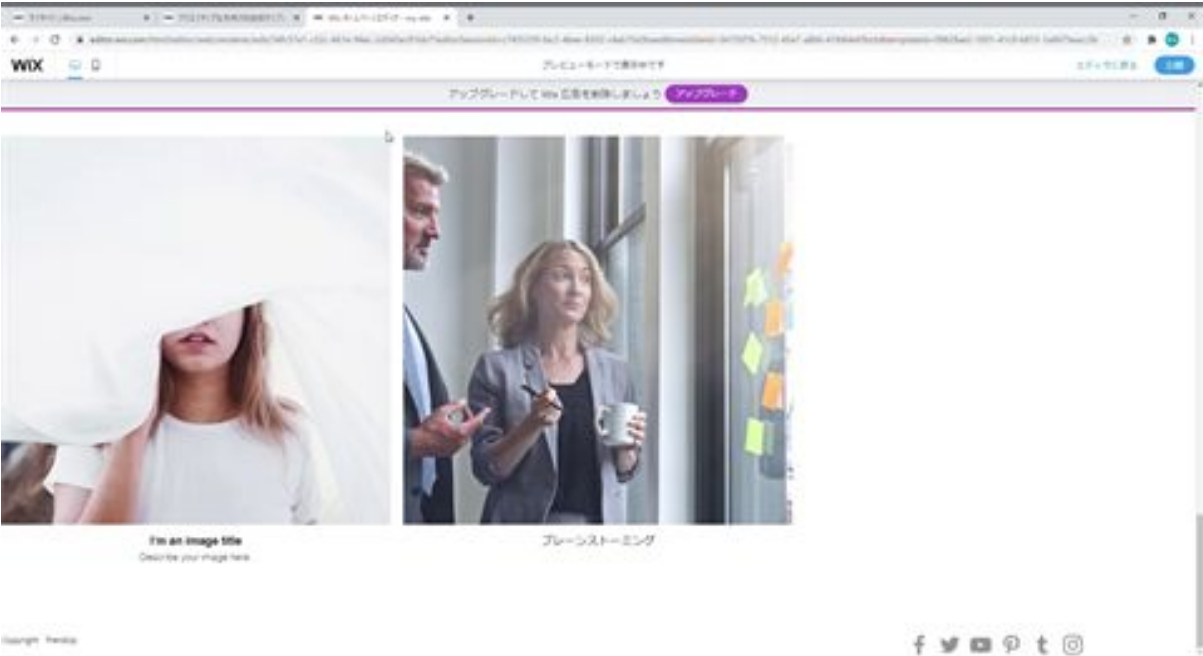


Last but not least, let's modify the Contact Us page. Select the Contact Us page here, and click on the Add button in the center part of the page, then click on the Contact Us form.

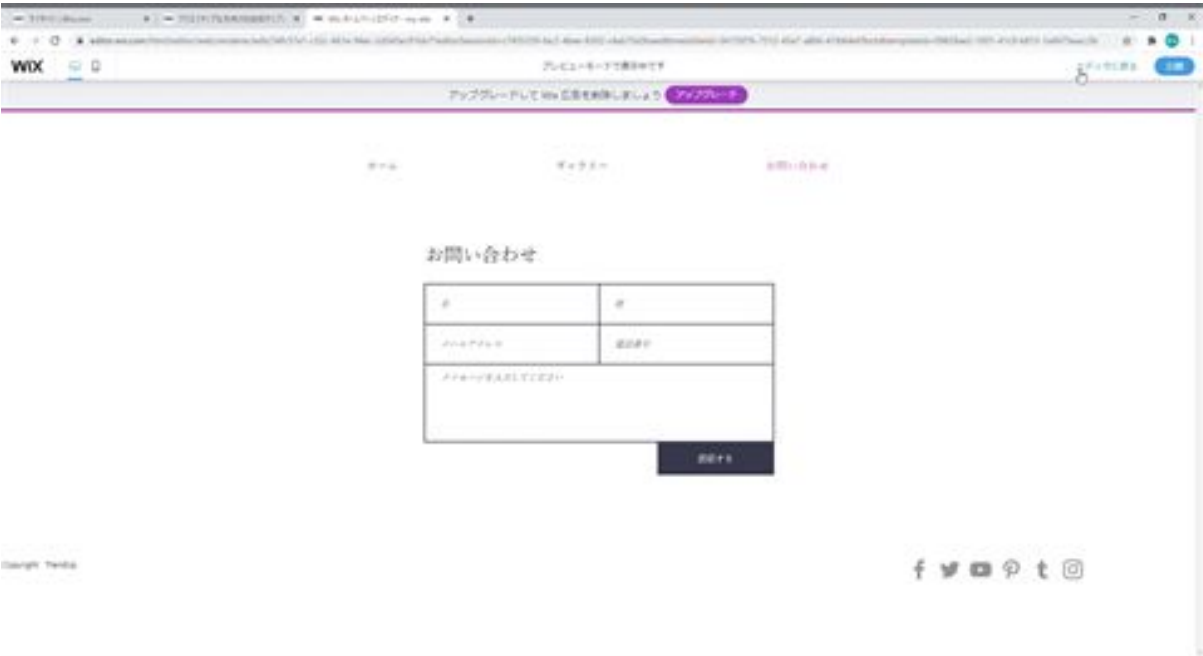
It is ready to use by default, so you can leave it as it is. Once your inquiry has been entered, it will be sent to your registered e-mail address.



Now check the preview once. Then you will see the header, footer, and gallery on the home screen, and the video is playing in the gallery.



If you look at the Contact Us page, you will see that the Contact Us is also displayed. When I stretched the header so that it was centered, it displayed correctly.

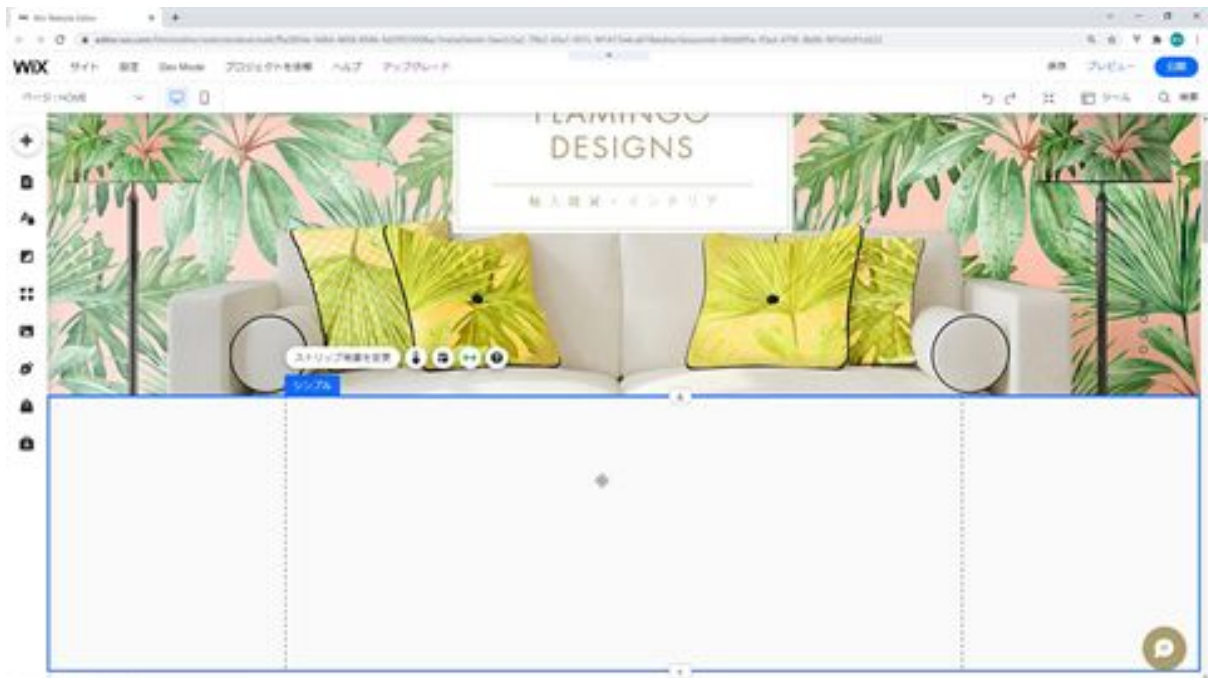


In this way, you can easily create a Wix website by adding more and more things you want to add from the Add button. You can create a Wix website by simply adding and placing the necessary items from the Add button. These are the basic operations and explanations of Wix.

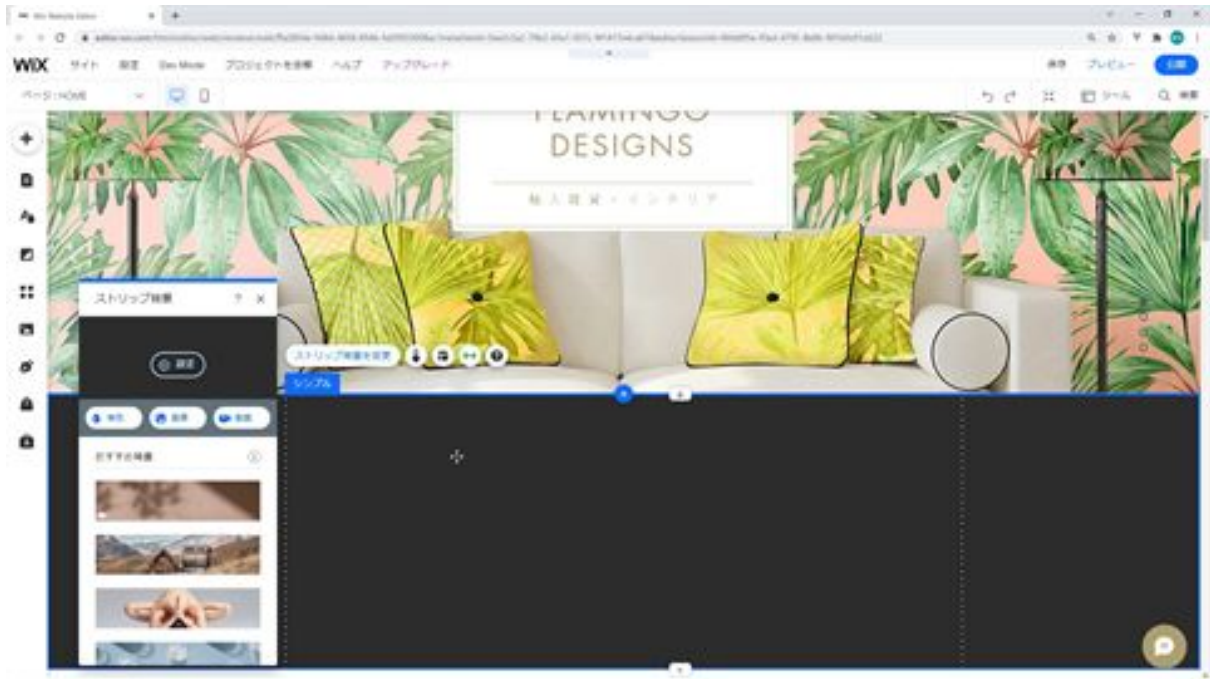
The concept of the strip and its basic usage

Next, I would like to explain the concept of strips and their basic usage. First of all, what is a strip? It is an area where you can place images and text, but you can use it as a component of your design.

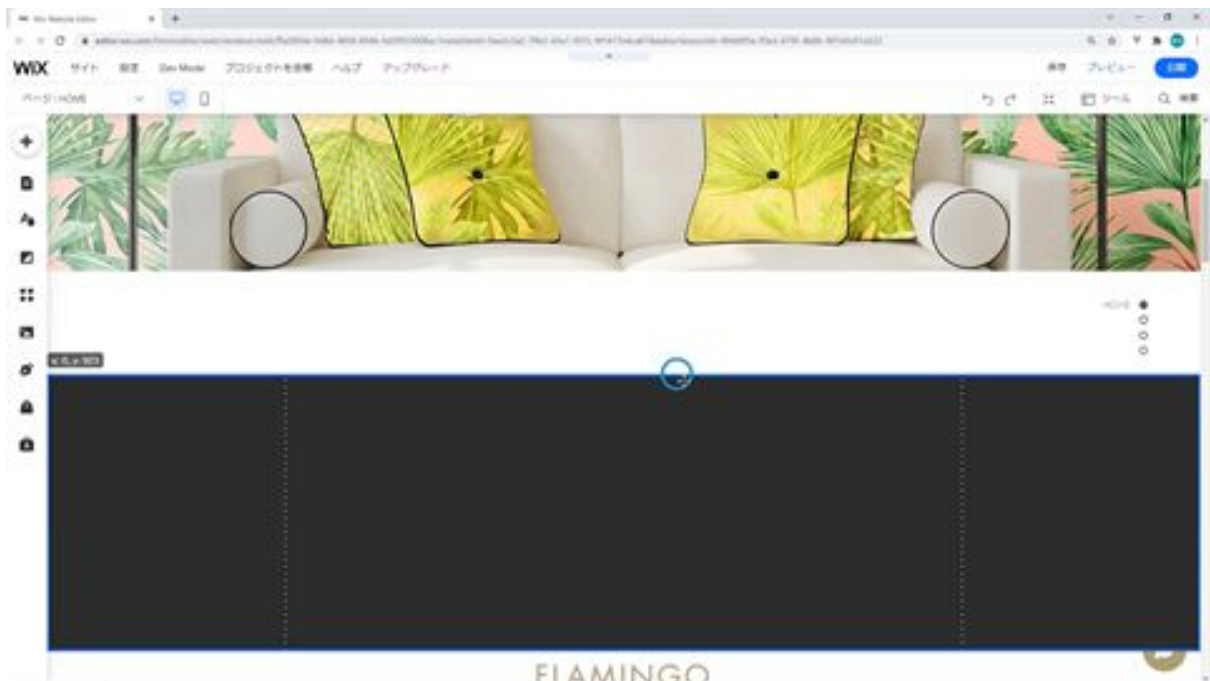
You can place an image or text in this strip, and if you move the strip, it will move at the same time. This can also be thought of as grouping using the strip. You can also use this strip to compose parts to create complex designs. Let's try using the strips in practice. First, let's place the strip. Drag and drop a white strip from the Add button to Strip, Classic.



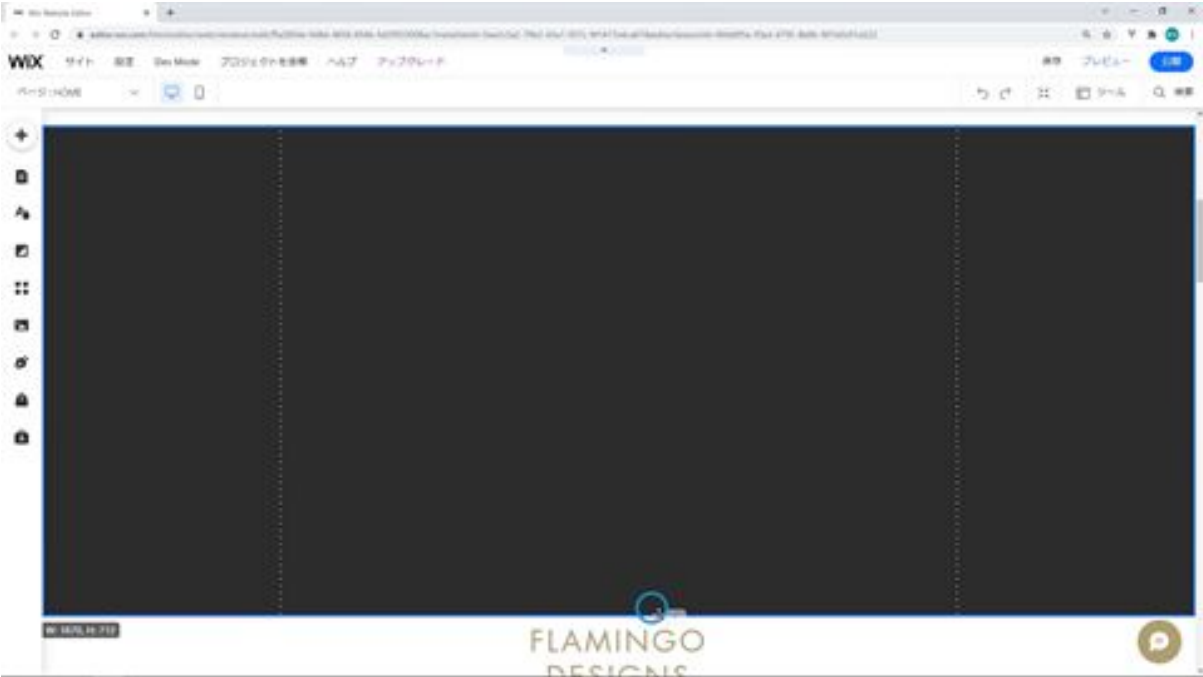
After adding the strip, click on it and you will see a button to change the strip background. First, let's change the background.



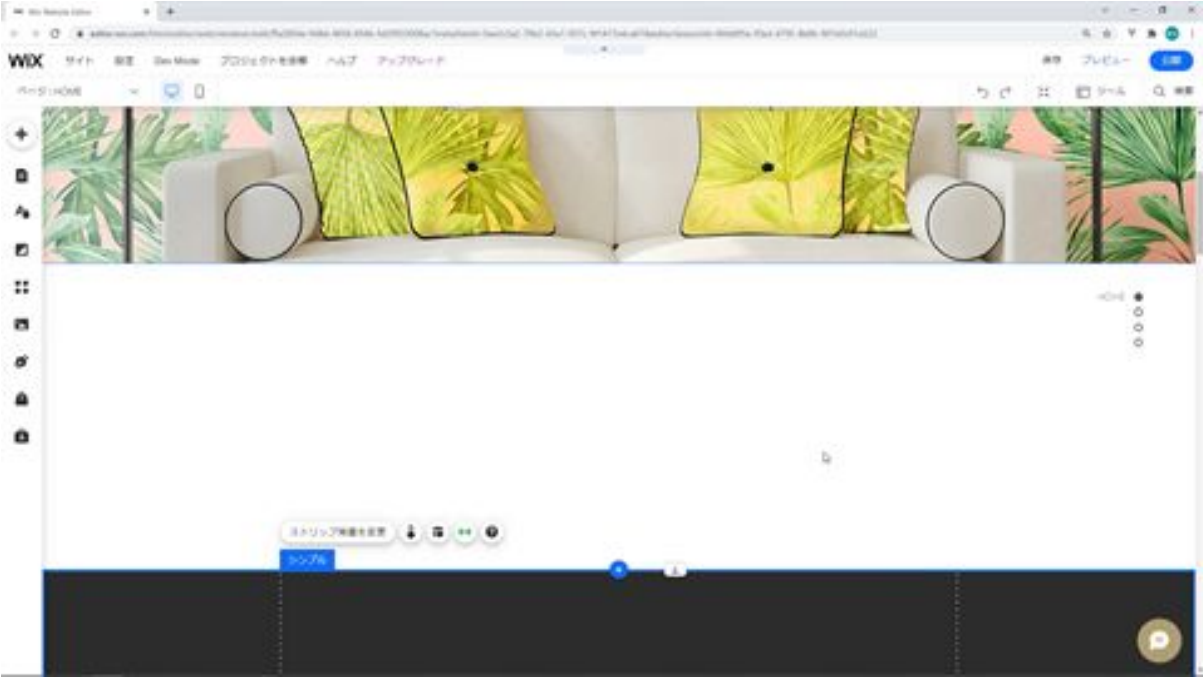
If you look at the strip, you will see that it is surrounded by a blue frame. This blue frame is the area of the strip. There is a drag button on top of the strip and a stretch button on the bottom. If you drag down with the drag button at the top, you can move it in conjunction with the part below it.



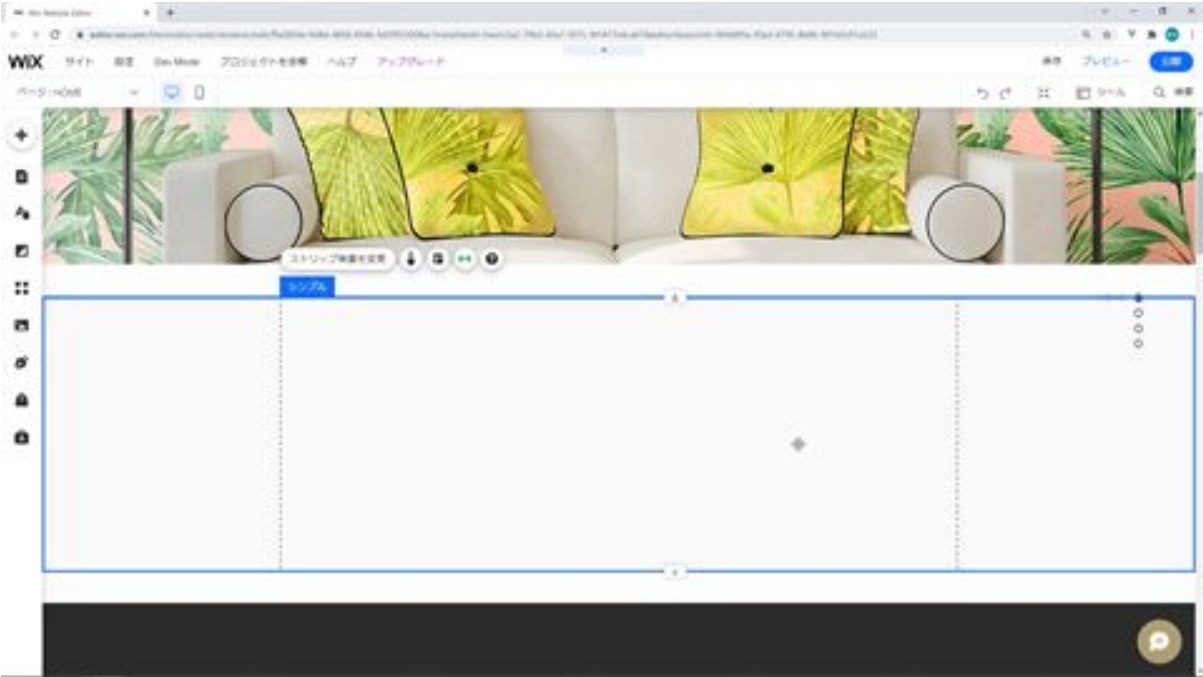
Now drag the stretch button at the bottom down to resize the strip in conjunction with the parts below.



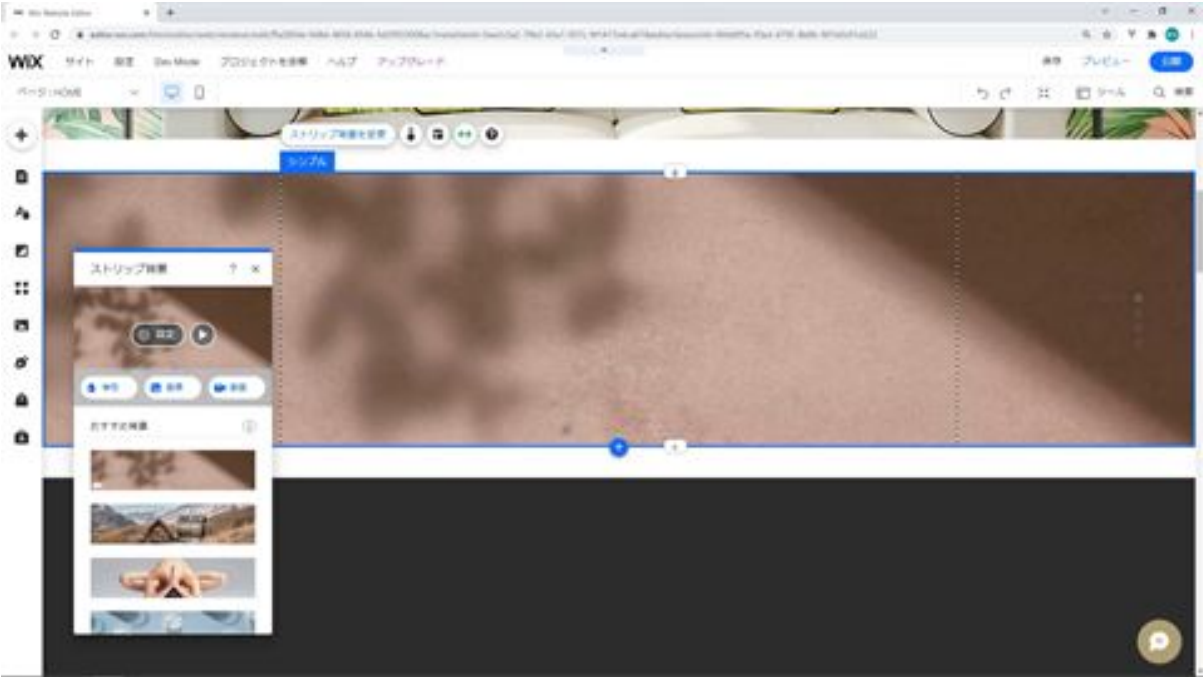
Move the strip down with the drag button to create space.



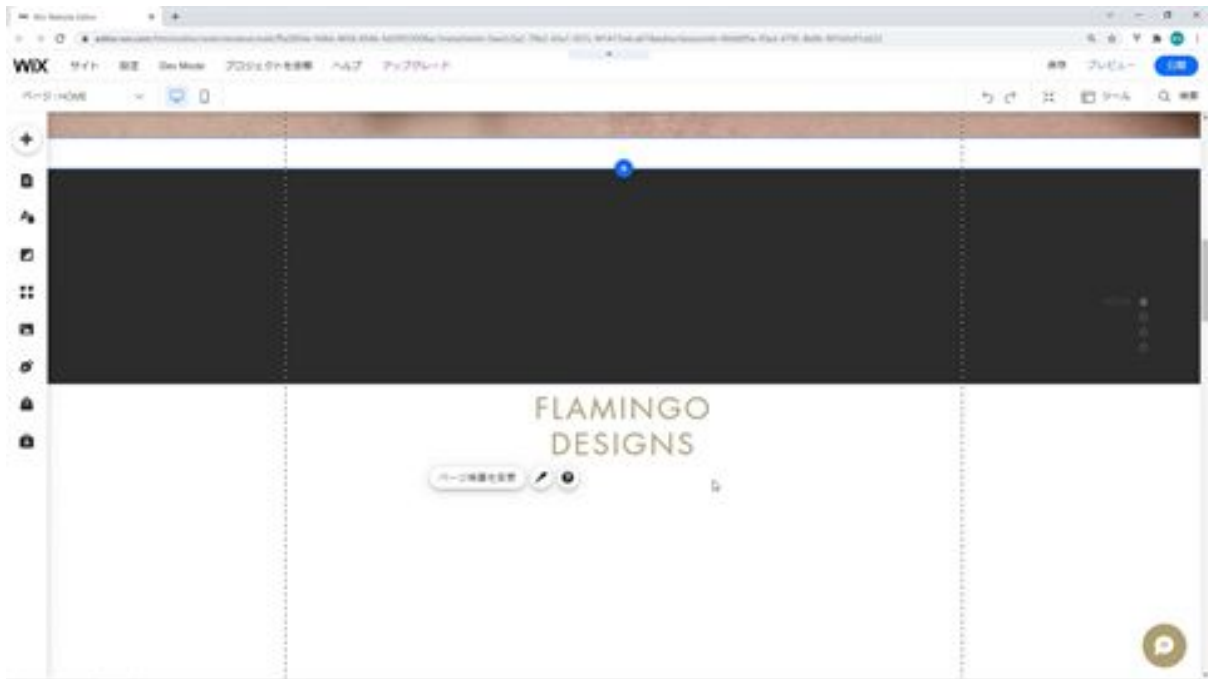
If you want to put some parts in this space, you can use strips. Think of it as a must to use a strip here. Add the strip again.



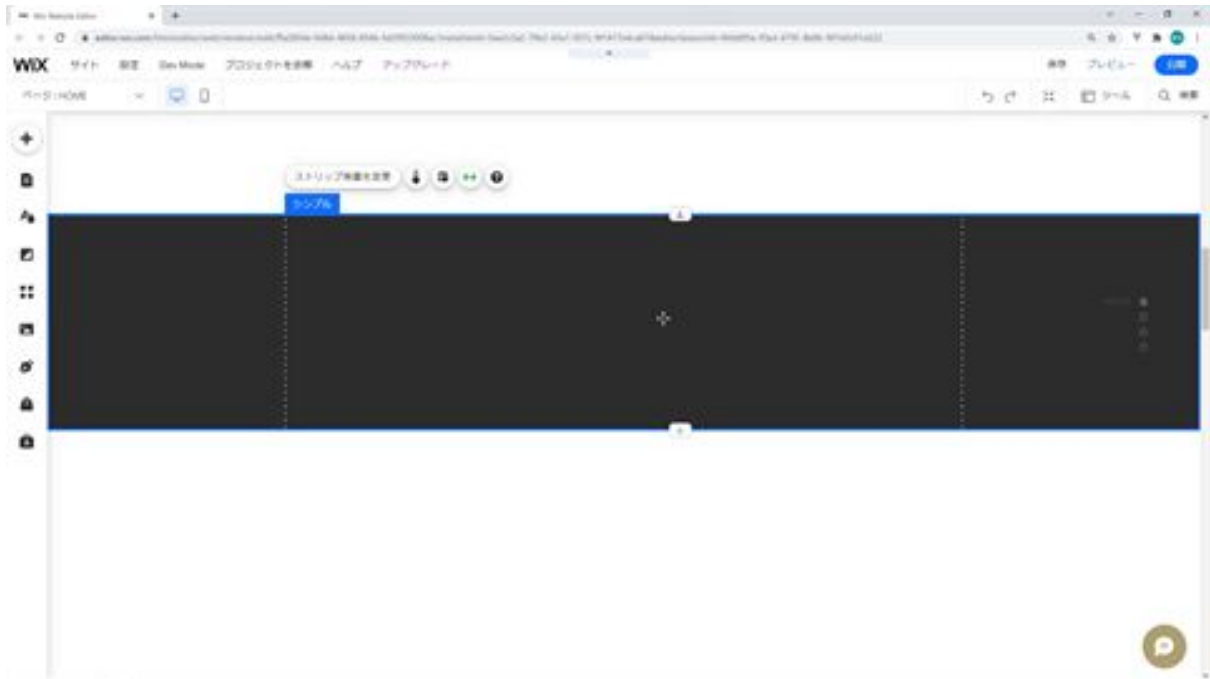
After adding the strip, try to change the background to a photo. I was able to change it to a photo strip.



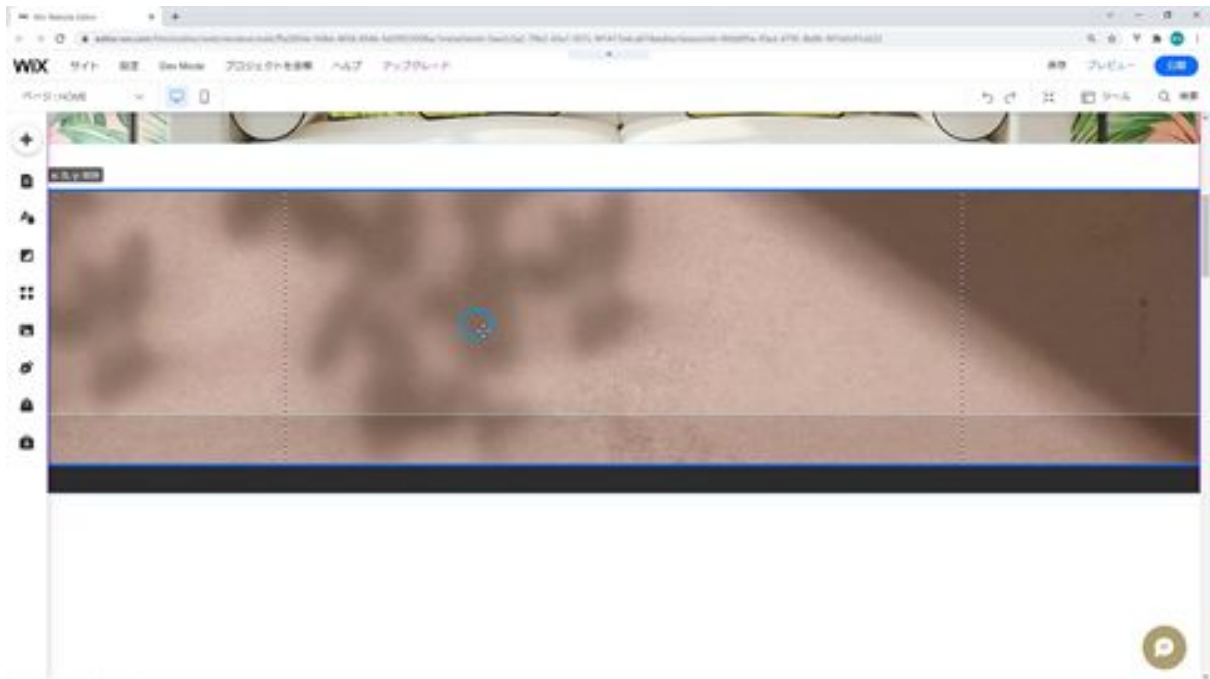
Here you see a background, a black strip, and a photo strip, but what is the difference between the background and these two strips?



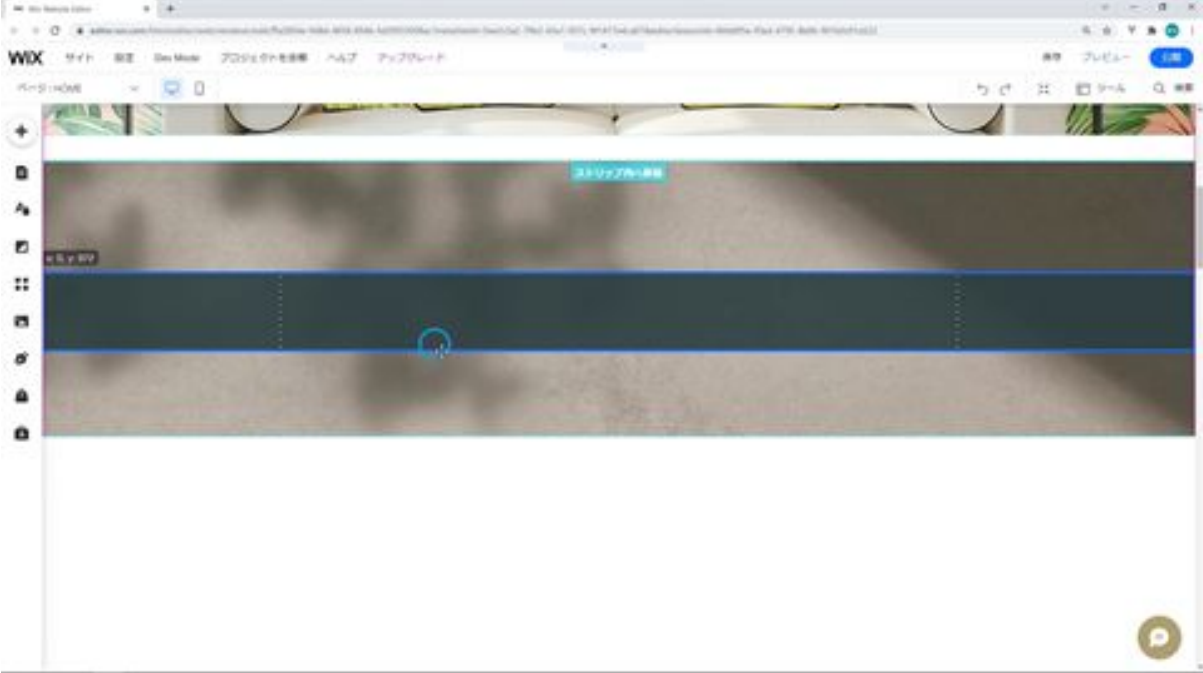
You can see the black strip, the photo strip, and the background, which is the bottom of the Wix screen. The background is at the bottom of the Wix screen, and the black strip and photo strip are on top of it, at the same height, and can be resized and moved freely. If you drag the strip and move it by itself, it will appear above the background text below it, hiding the text.



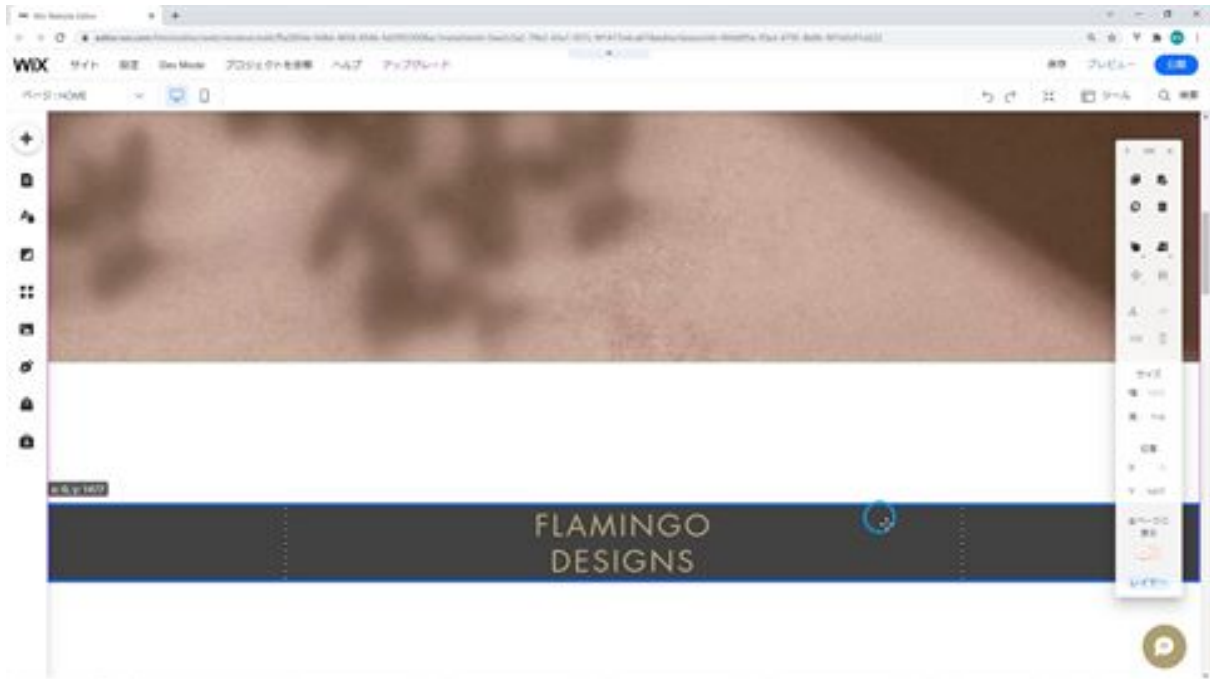
You can also move the strip inside the strip. Change the black strip so that it is smaller and fits into the photo strip. If you move it so that it is partially on top of the photo strip, it will not move at the same time because it is not fully on top of it here yet.



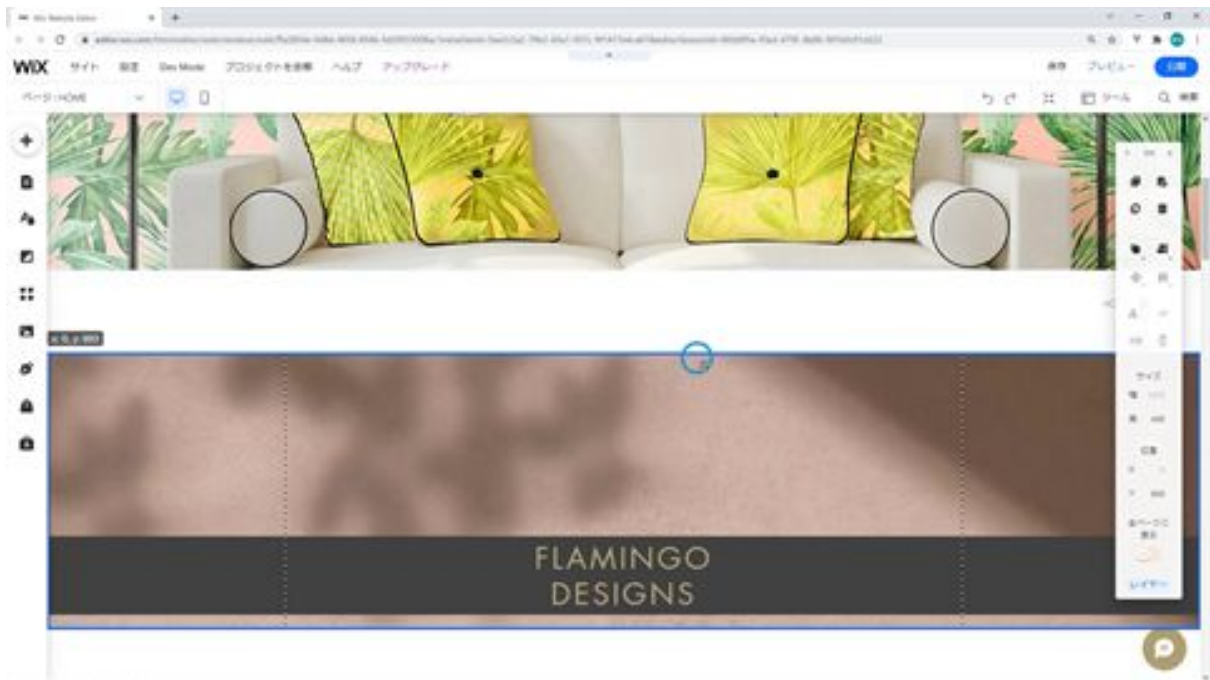
However, if you bring the black strip to the center and drag and drop it while moving it to the strip. This way, the black strip will be completely on top of the photo strip and you can move it at the same time.



The next step is to put the text on the black strip. When you move the black strip, the text will of course move with it.



Then, if you put this strip on the photo strip and move the photo strip, the black strip and the text will all move in tandem. As you can see, this is what I mean when I say it's grouped.



The strips all work together, so when you want to change the layout of a part, you can change it immediately and work very efficiently. You can also create complex designs efficiently by creating parts in strips.

Please try using the strips first, as it will be more efficient for you to learn and create with your hands. That's it for the concept and usage of strips.

Let's check the screen of the application to be created

Next, let's check the screen of the application we are going to create. First, let's run the application and check the screen.

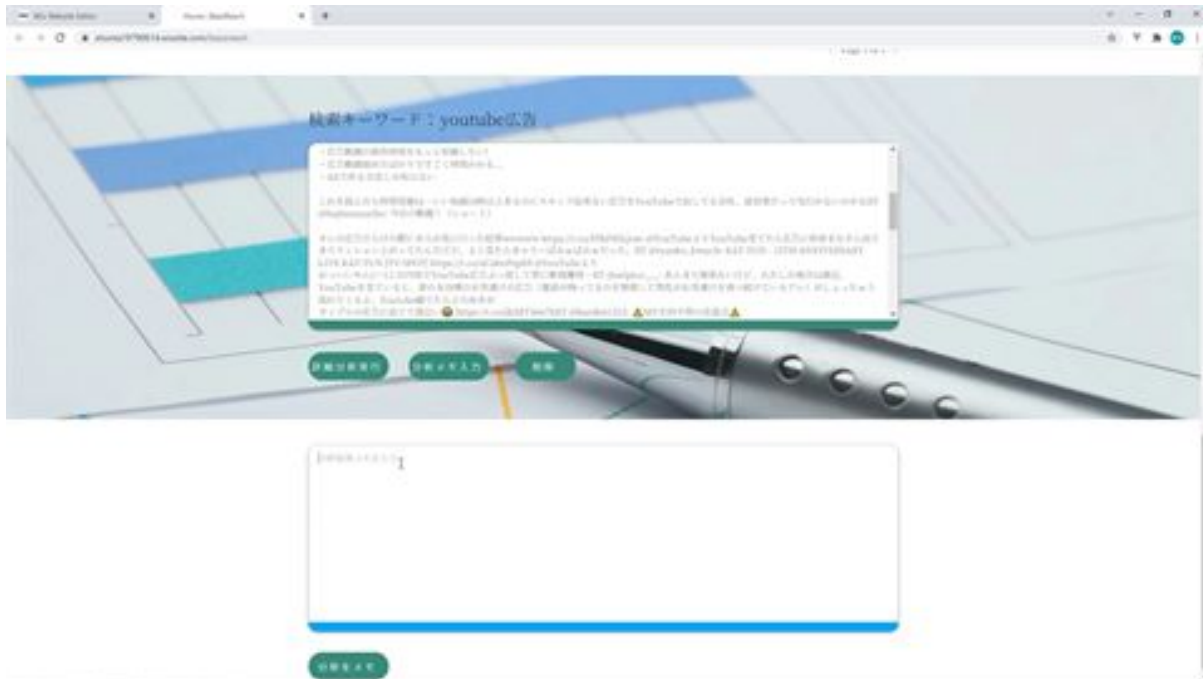


There is a function that allows you to search for tweets by entering the keywords of the tweet with a copy and title. When you do a tweet search, a screen will appear and you can search for tweets.



You will see a button that says "Save search tweet", and when you save it, you will see an animation that says the tweet has been saved, and the tweet will be added to the table.





After entering the analysis memo and clicking the Memo Analysis button, a dialog box will appear with the statement that the memo has been saved.



Next, click on the Delete button, and a confirmation dialog will appear to allow you to delete or cancel.



If you cancel it, it will be canceled as it is, and if you delete it, the data will be lost and there will be 4 entries.



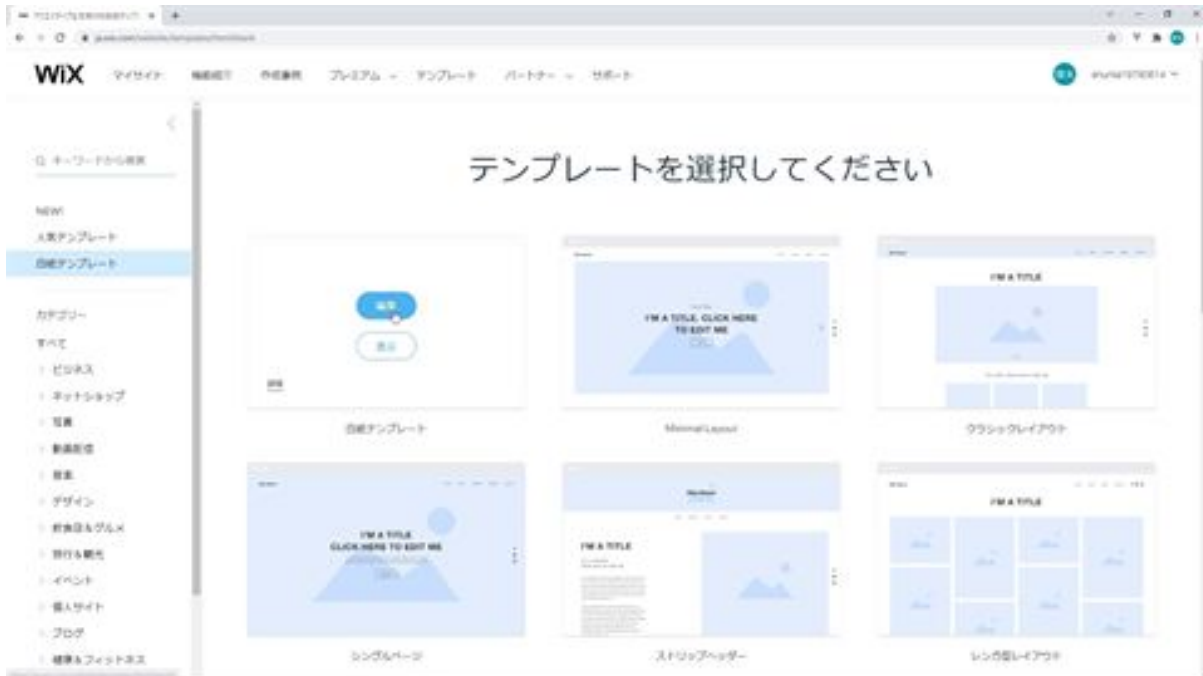
We would like to create the screens and components necessary for such an application from the next step. This is the end of the screen check for the

application to be created.

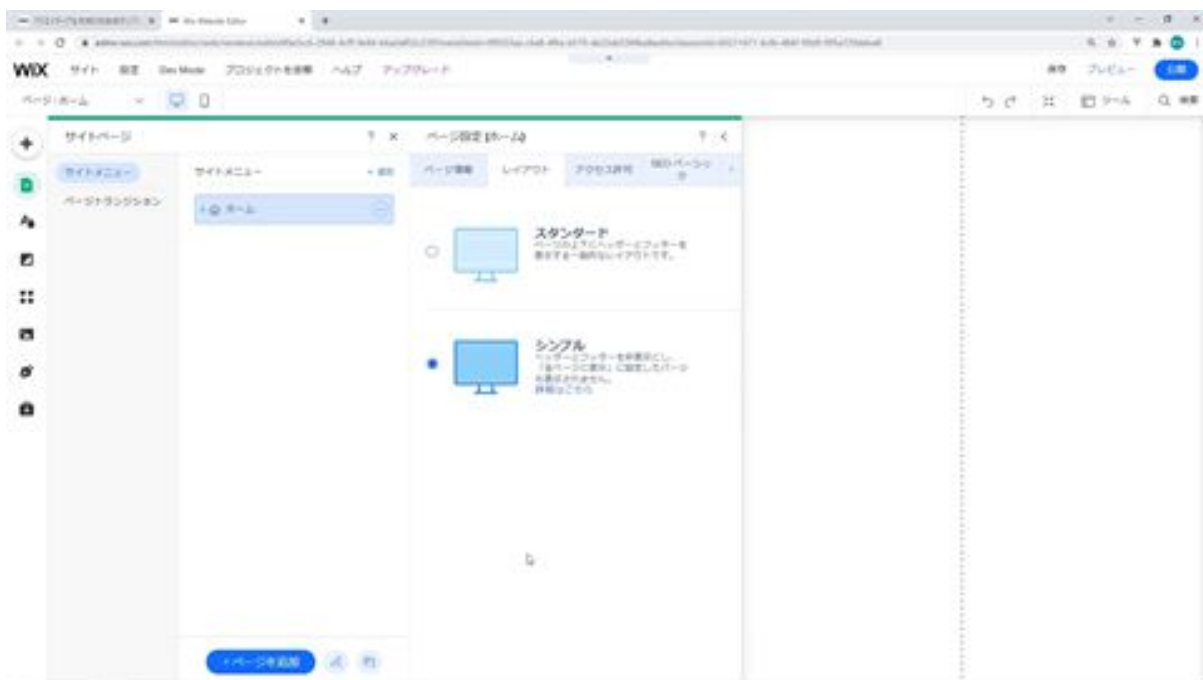
Let's create a strip for the menu



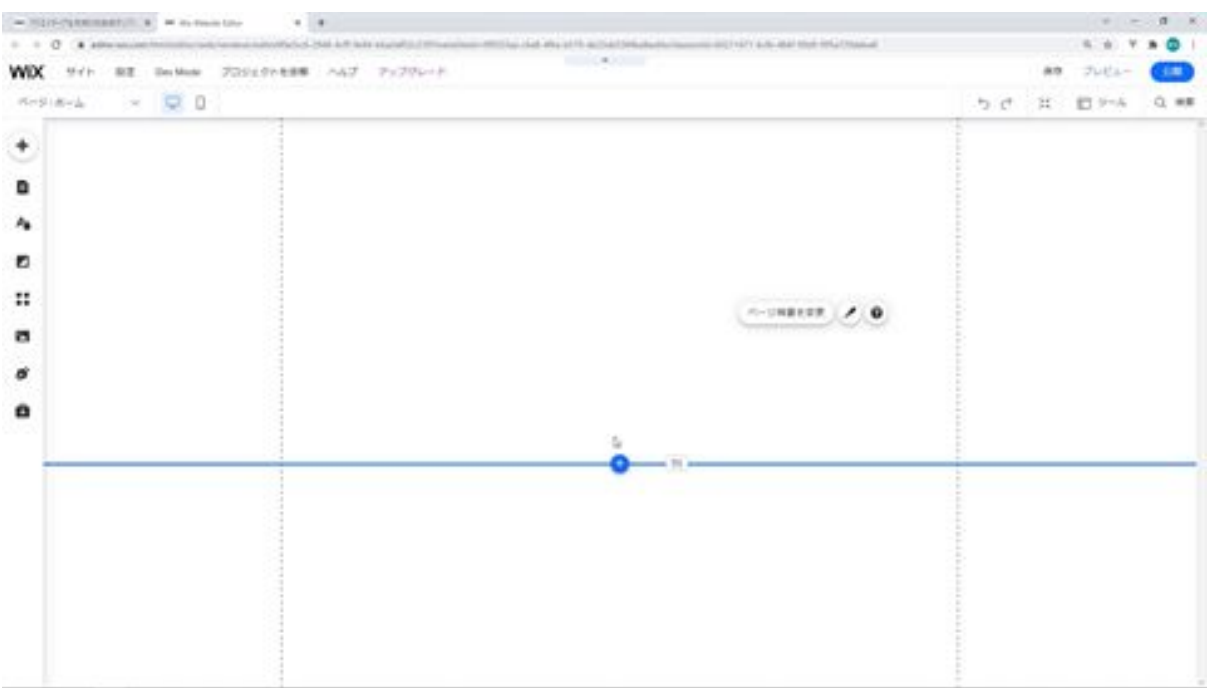
Next, let's create the screen. First, we want to create a strip for the menu. Select "Other" in the screen template and click the Edit Blank Template button from the blank template.



I'd like to create the screen from here. First of all, I'd like to change the layout of the header and footer, but since I won't be using them this time, I'd like to change the layout. I'm not going to use the header and footer this time, so I'd like to change the layout. Go to Menu & Pages, click on Settings for this home, and set Simple from Layout.



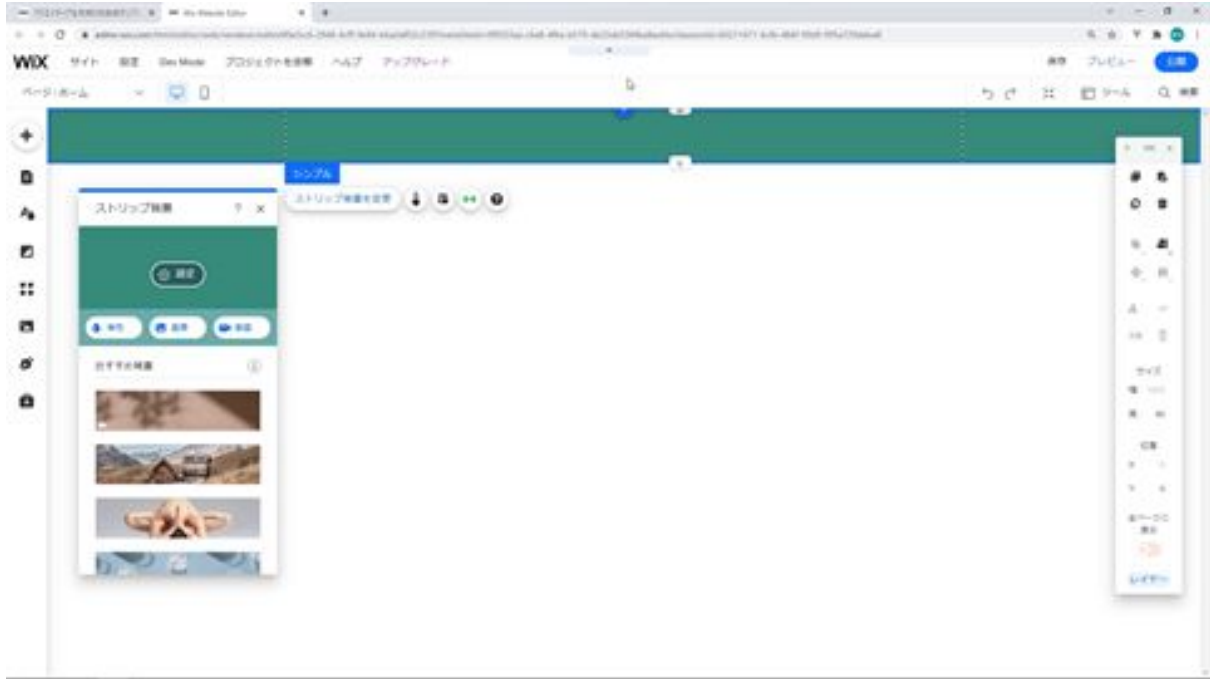
Then, the header and footer are gone. I would like to create the screen in this state.



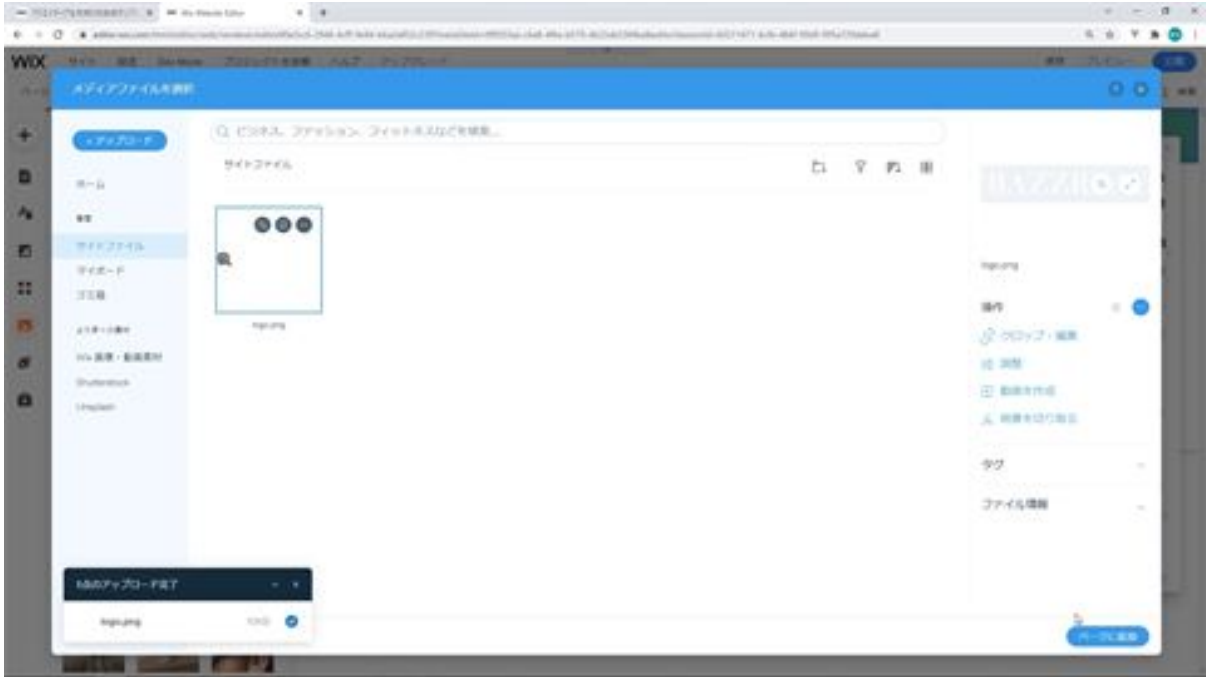
The first step is to create a strip for the menu. Drag and drop the strip from the Add button. Adjust this drag to the top and change the height of this strip to 80px. From Tools, bring up the toolbar and change the height to 80px. Now the height of the strip is 80px.



Next, we will change the color. Click on the Change Strip Background button and add a color from single color, here enter #388C7C and click on the Add button. Then the color of the strip background has been changed.



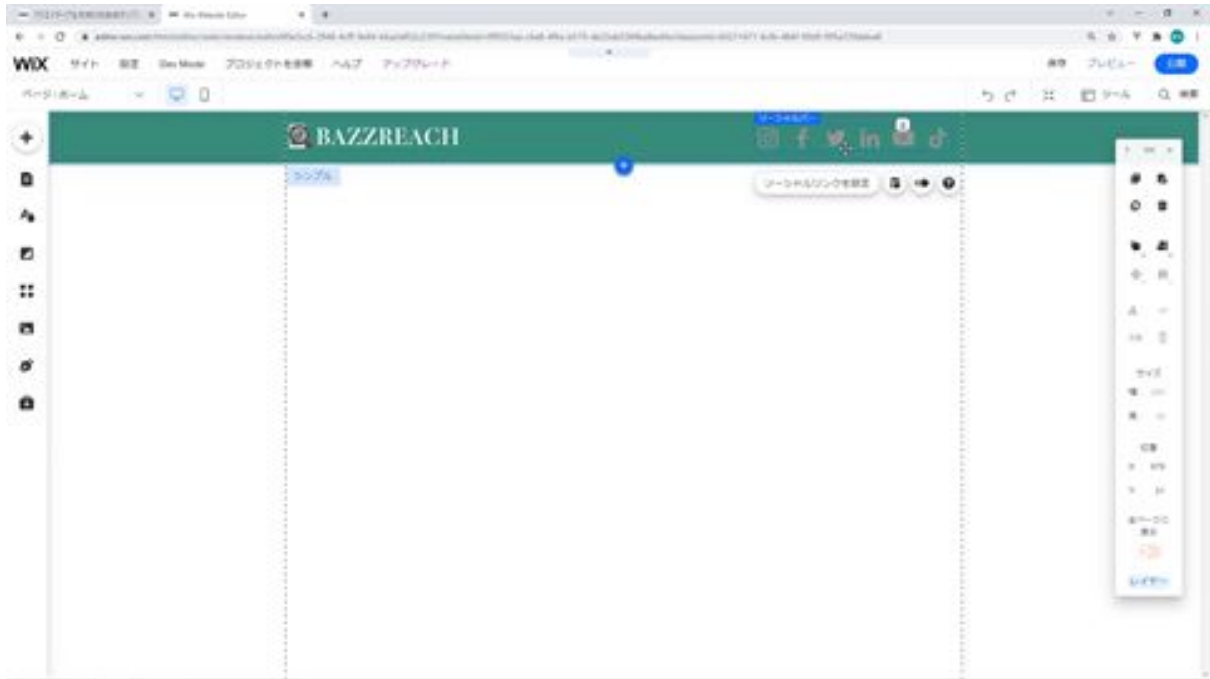
Next, we will display the logo in the frame here, click Upload Media from Media to upload the logo image. Select the logo image and click the Add to Page button.



The next step is to reduce the size of the logo because it is a bit large. Change the height of the logo to 40px. Now move it into the strip, and move it so that it is inside the grid lines.



The next step is to add a social bar here. Select Social from the Add button and drag and drop the social bar.



This is a little hard to see, so we will change the design of the social bar. Click on the Set Social Links button, and then click on the Add Icons

button.



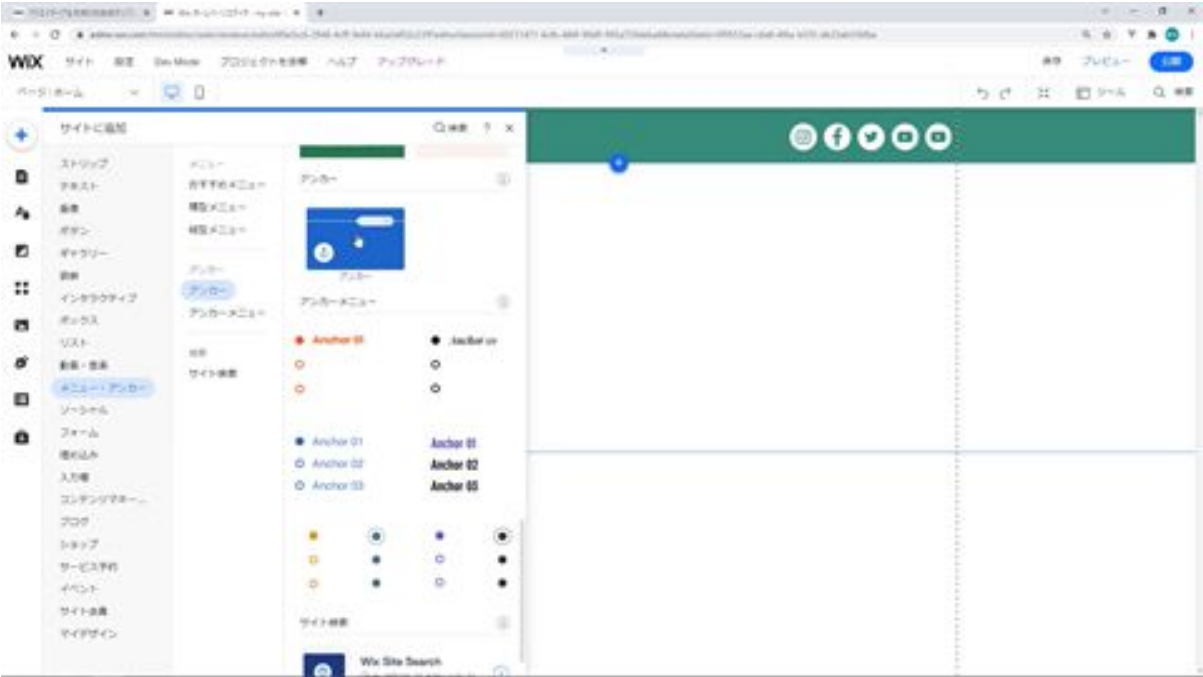
You can choose from different designs of social icons here, so choose the one that is easier to see. The social bar is now much easier to see.



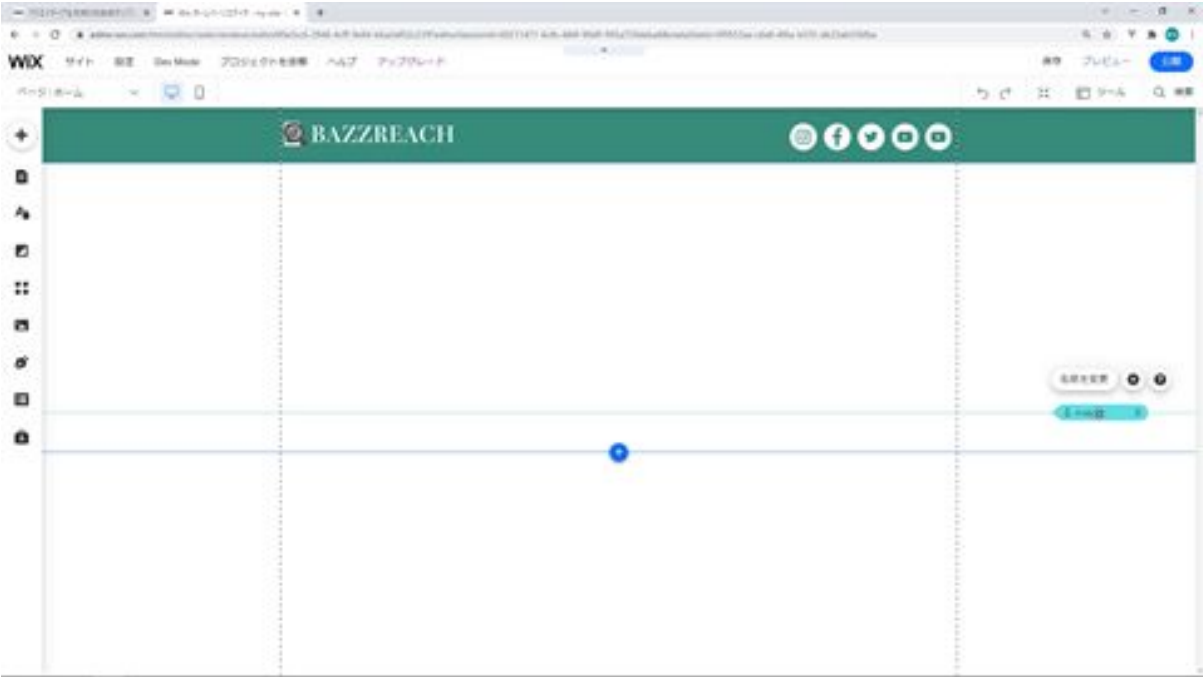
Now we need to modify it so that it is aligned with the logo. The position of the social bar is now centered vertically with the position of the logo.



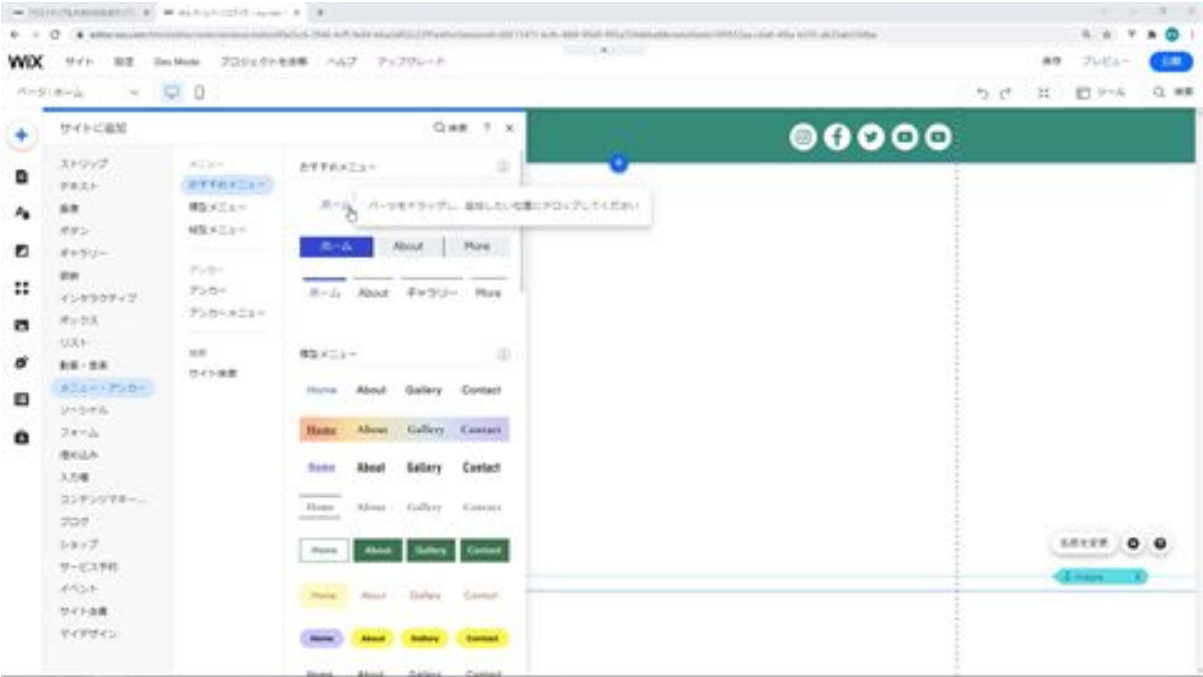
Now we would like to add an anchor. From the Add button, click Menu Anchors, and then click on the anchor here.



For the anchor name, enter Analysis as the anchor name and lower the anchor to an appropriate position.



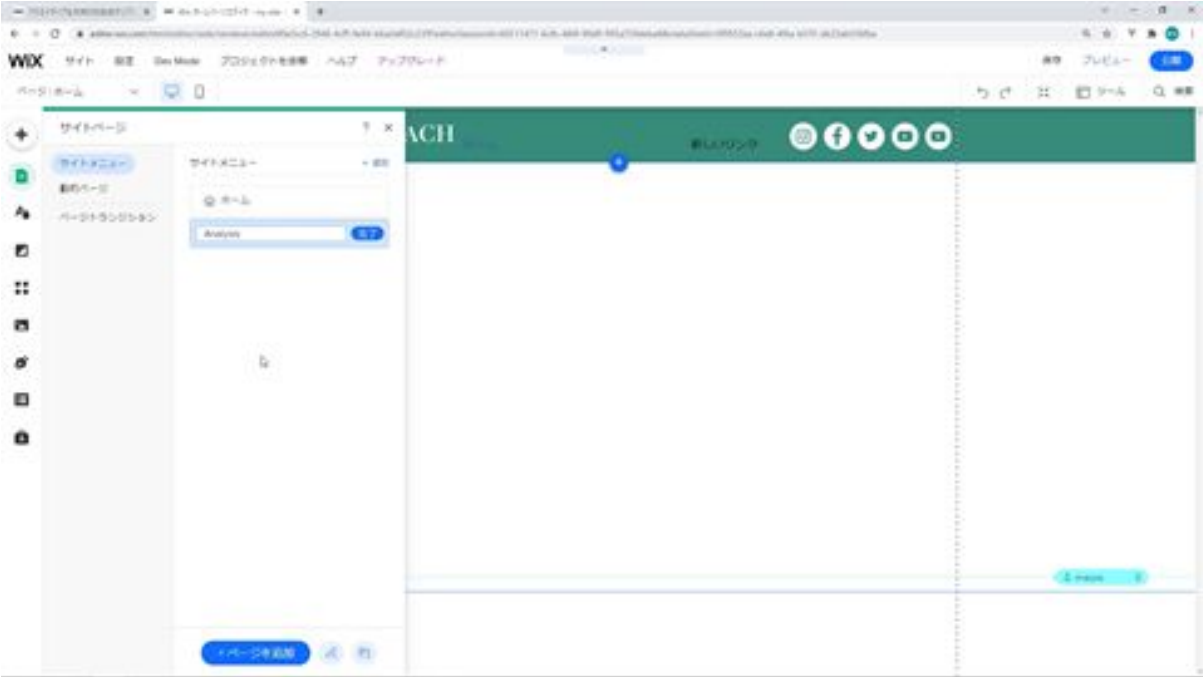
Next, we want to add a menu. Click the Add button and then click Menu to add a menu.



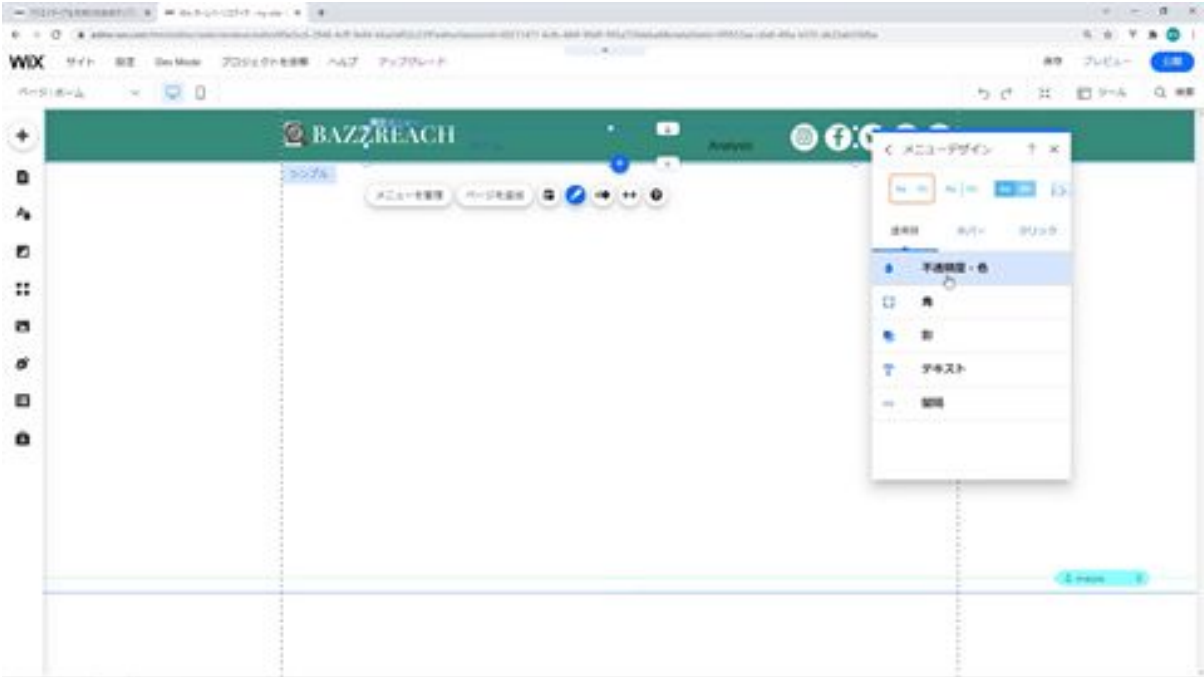
Click on the Manage Menus button. There will be an Add Link button, click on it. Then, select Anchor from the Choose Link to section, select This Analysis and click the Done button.



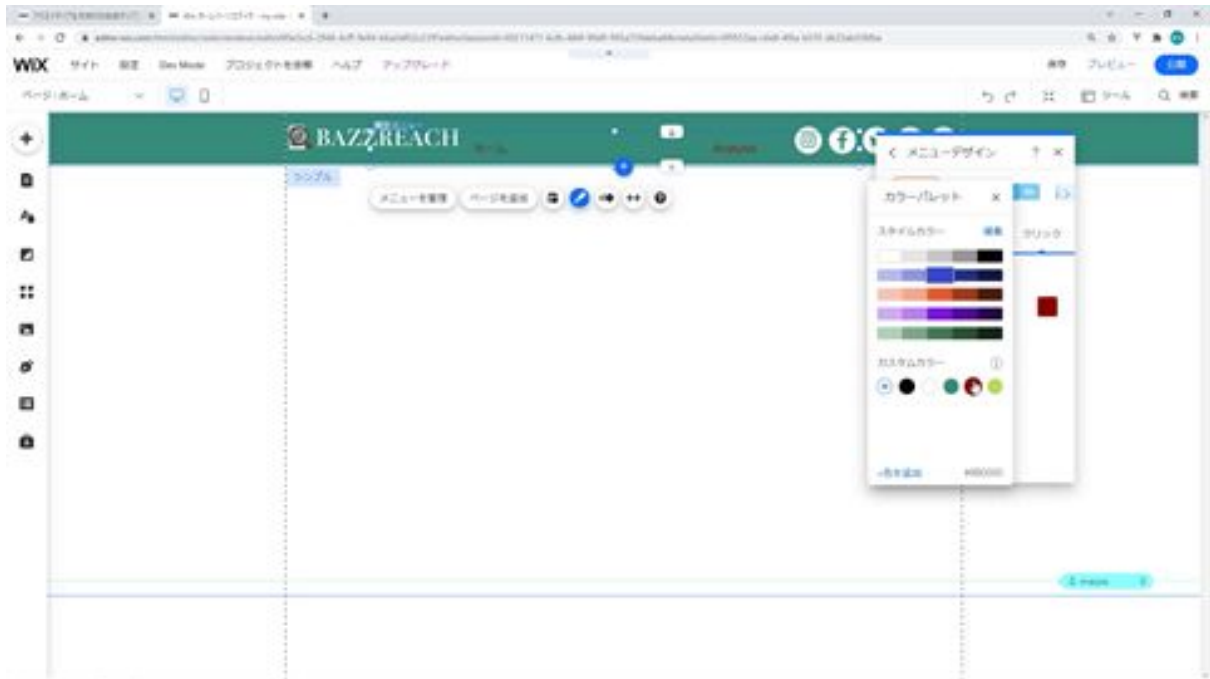
Then a new link will be added, type Analysis here and click the Done button.



Next, we would like to change the color of this menu. Click on this menu and click on the Change Design button. Click on the Customize Design button to change the color.



Select a color of #8B0000 for the hover here. Select a color of #8B0000 when clicking as well. This completes the color change.



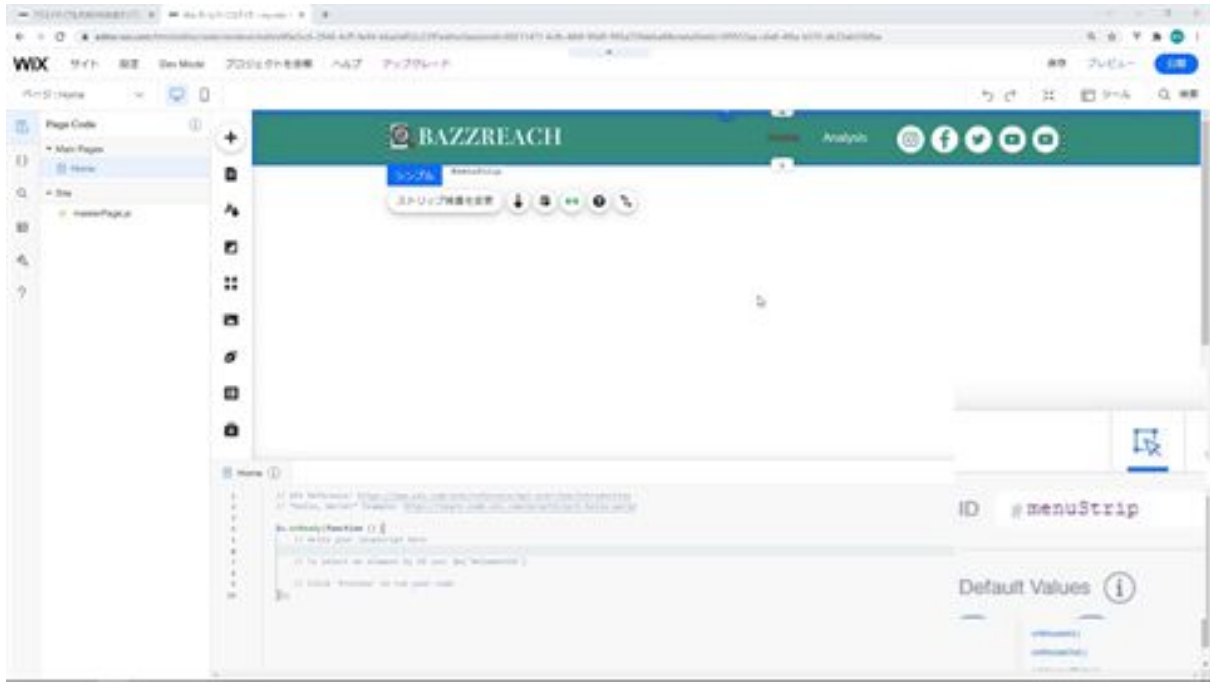
It is also a bit wide, so we will make it smaller. We'll use the Roman alphabet for Home here. The menu is now complete.



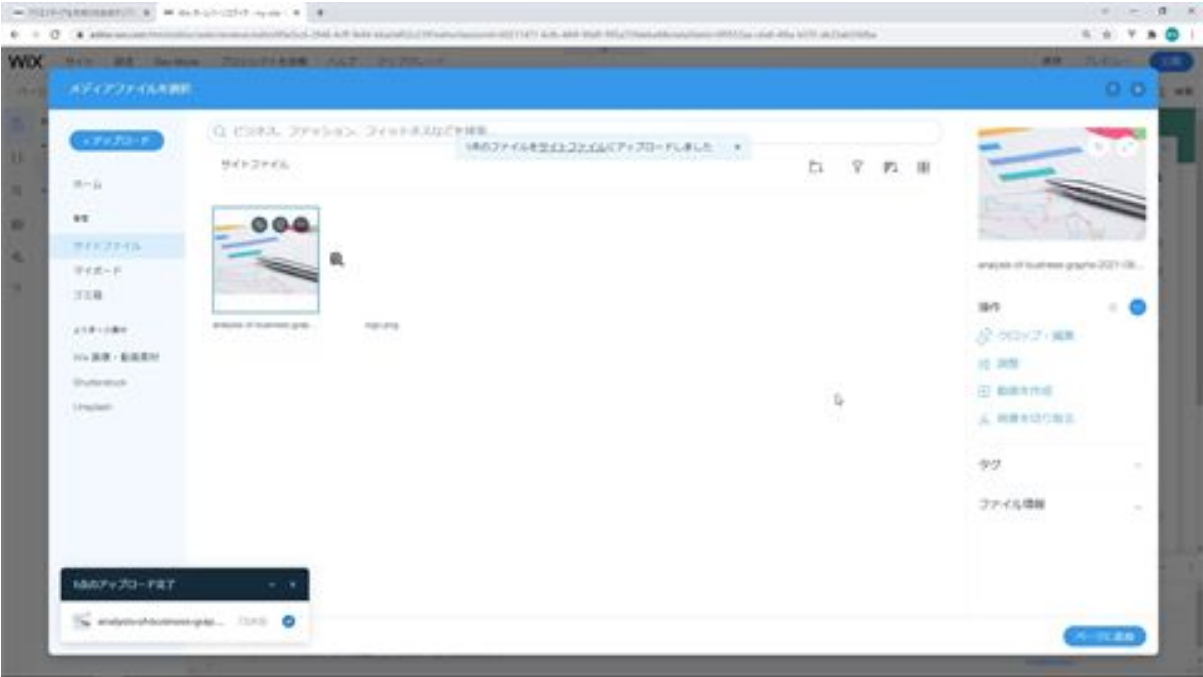
Next, we would like to change the ID name of this strip, click on the Enable Development Mode button from Dev Mode.



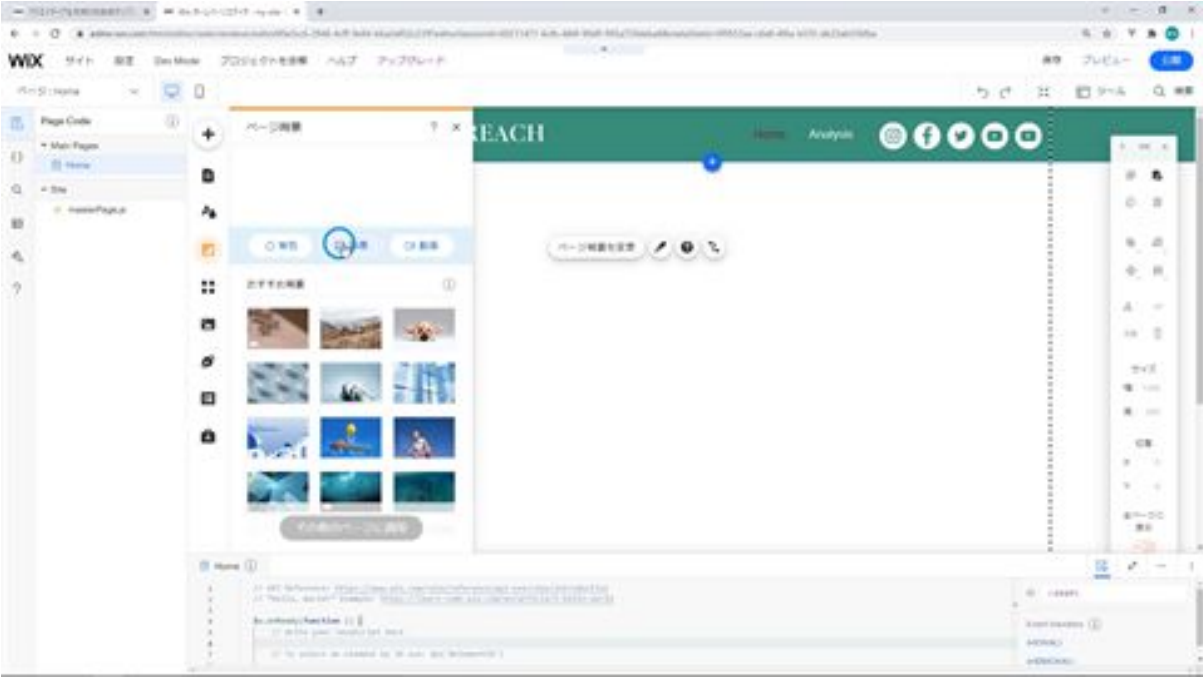
Click on the strip and change the ID name to menuStrip. By specifying this ID, you will be able to show or hide the animation freely in the source code below.



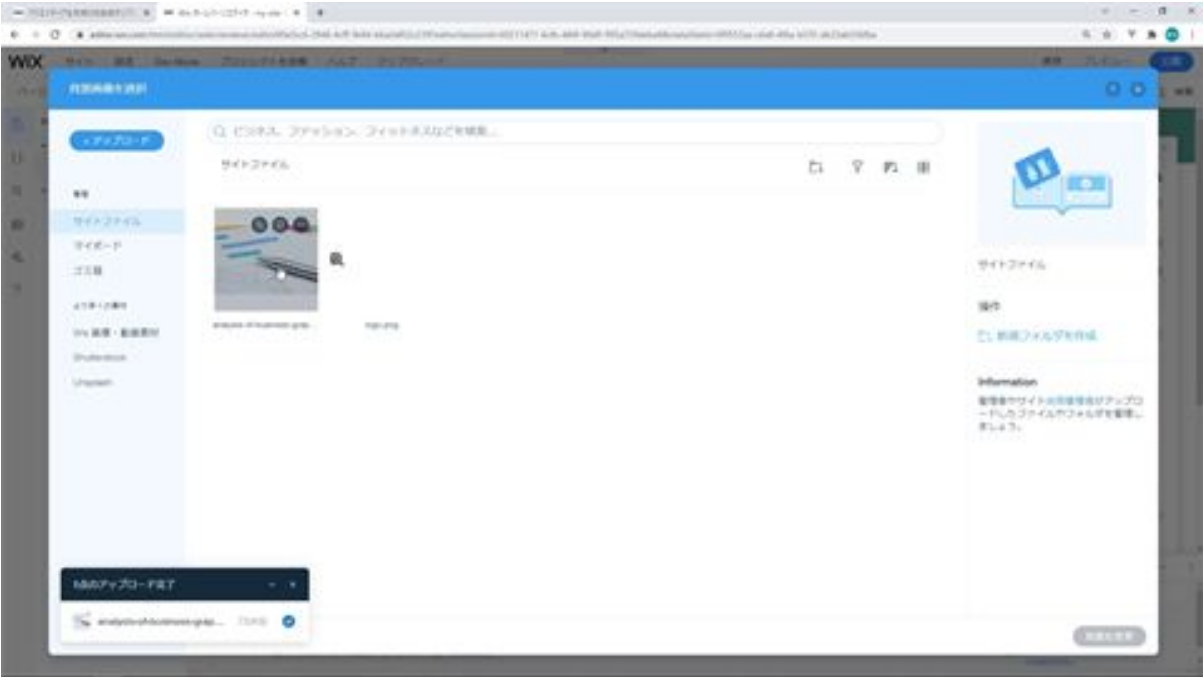
Next, we would like to change the page background. First, go to Media and click Upload Media to upload the background image.



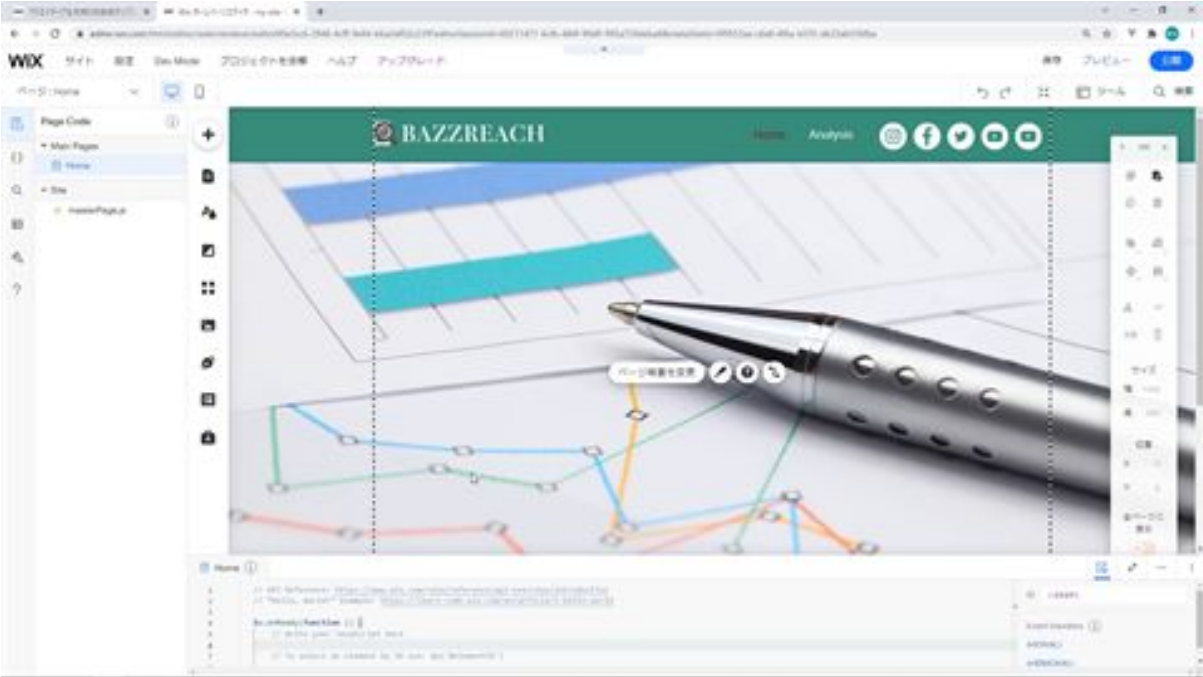
The upload will be completed and you can change the background.



Click Change Page Background and select the file you uploaded from Images.

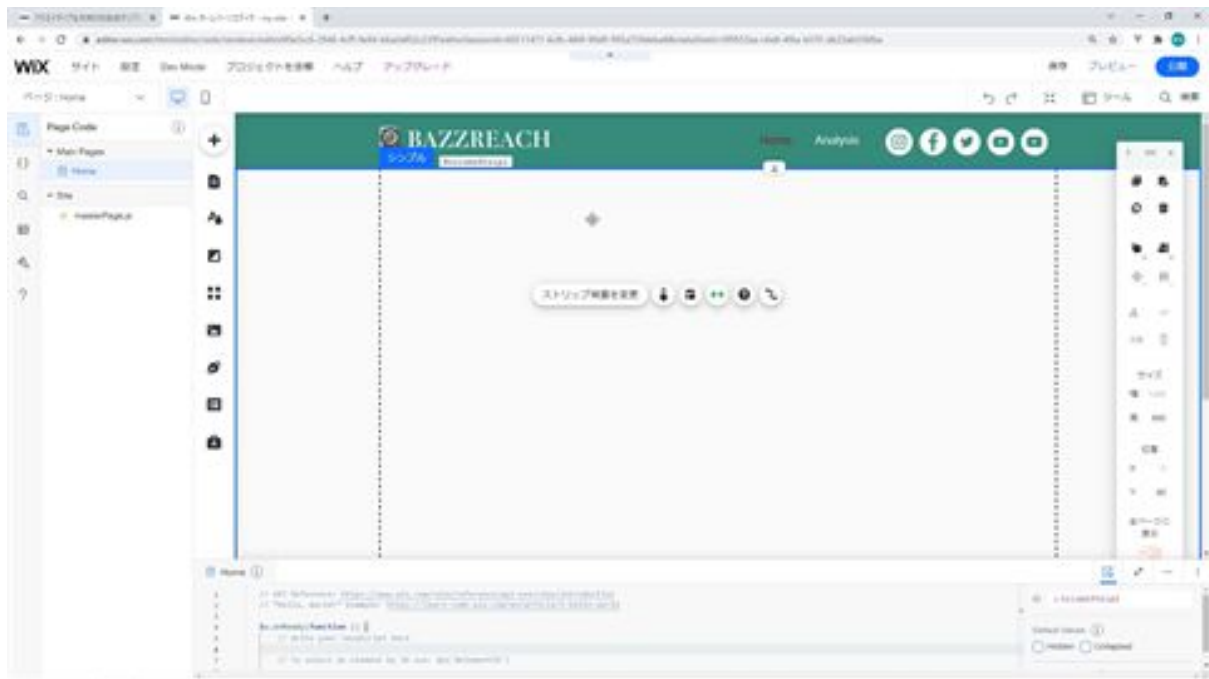


The background has been changed to the image you uploaded. That's it for creating the strip for the menu.

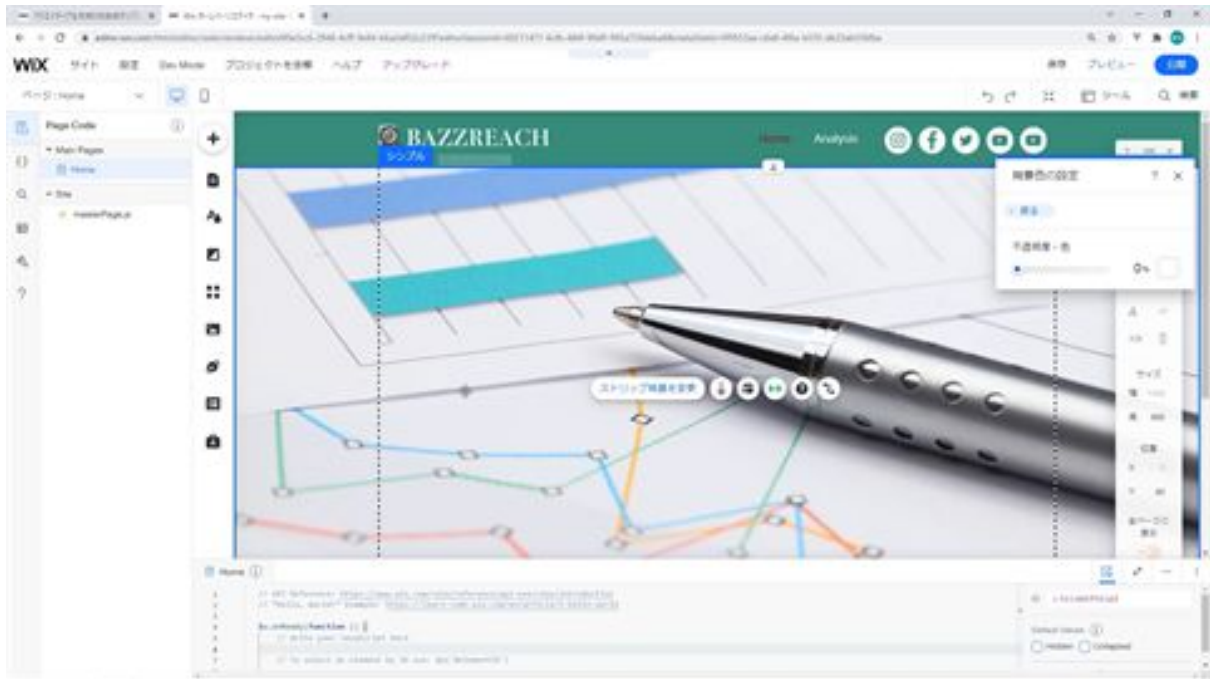


Let's create a strip for the tweet search screen

Next, we will create a strip for the tweet search screen. First, we need to add a strip. Go to Add and add a white strip. Next, change the height of the strip to 860px.



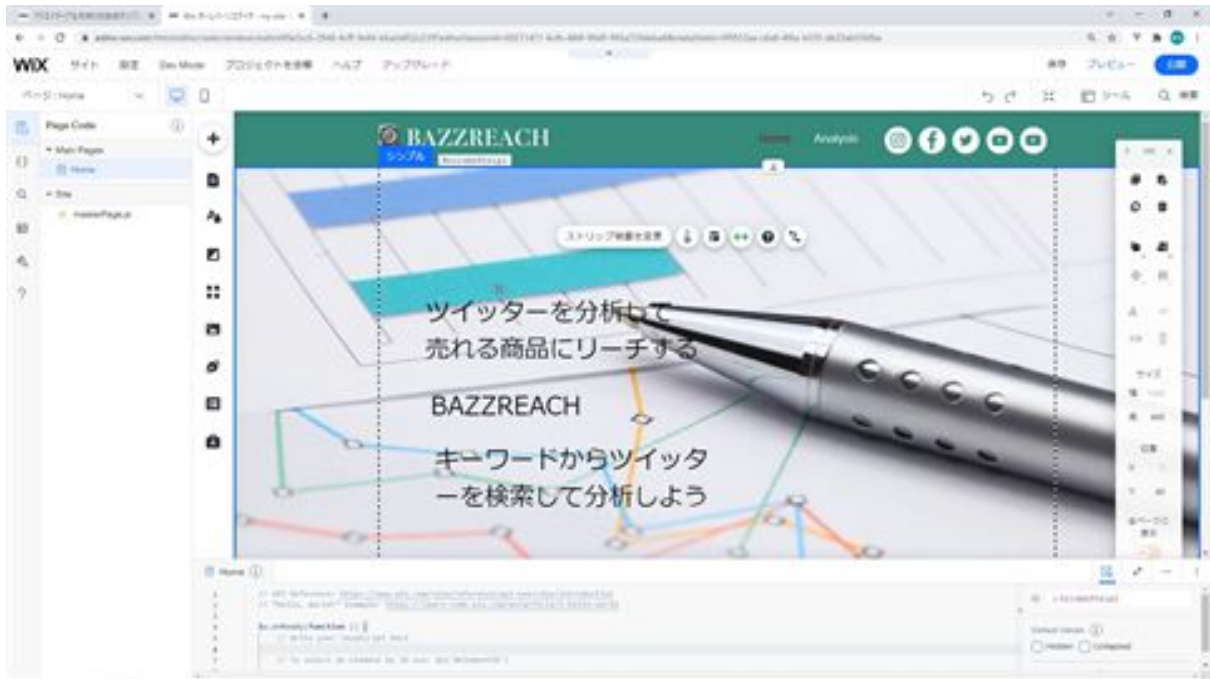
Change the background of the strip so that the background behind it is visible. Change the opacity to 0 from the settings.



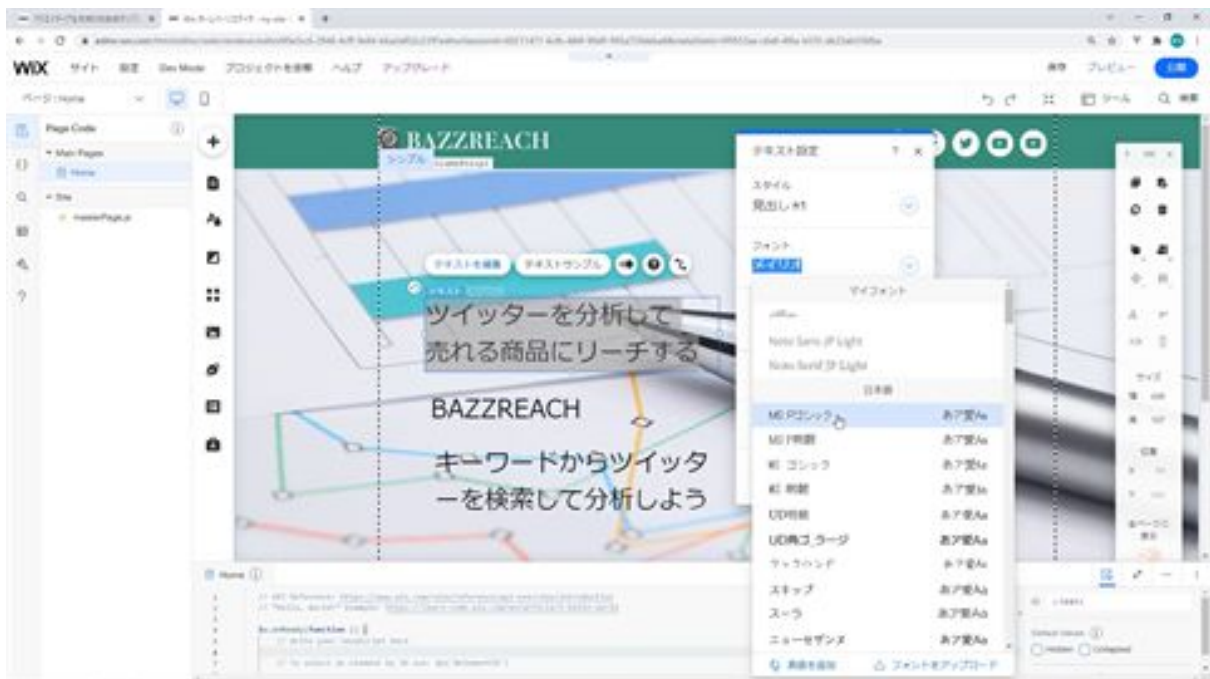
The next step is to add a copy and title to this strip. We will add text from Add and then change this added text.



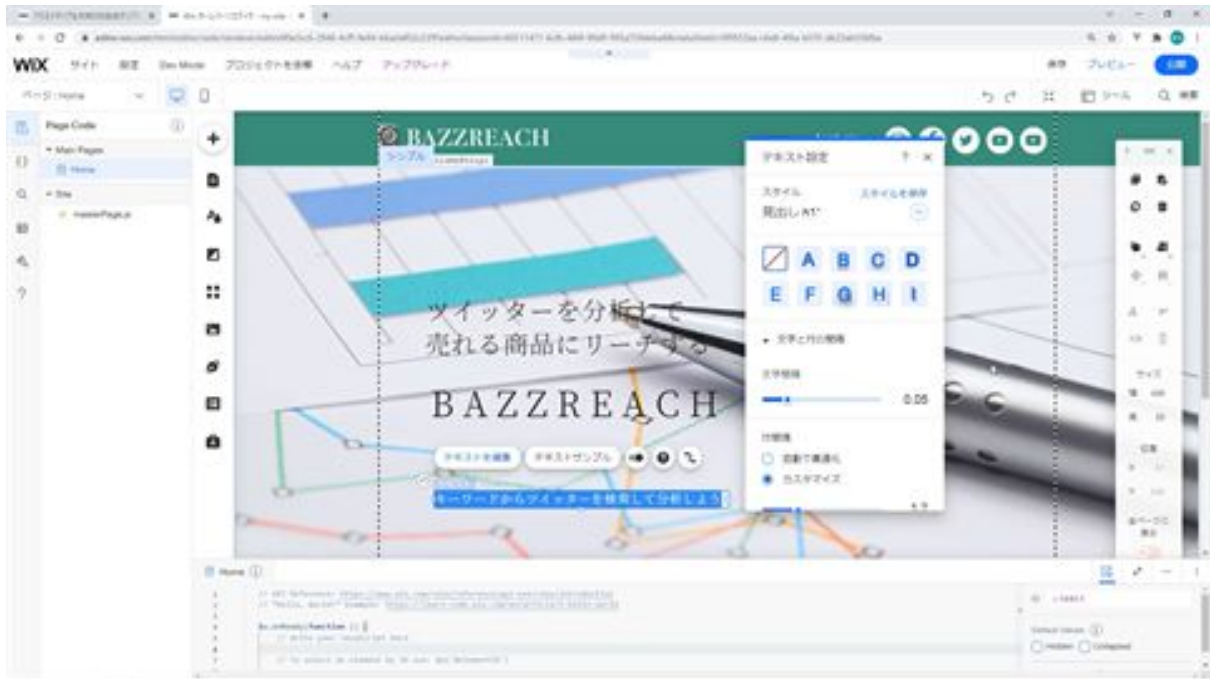
We want two more texts, so we will copy and paste them. Change this text to the title and change the other one to the sub-copy.



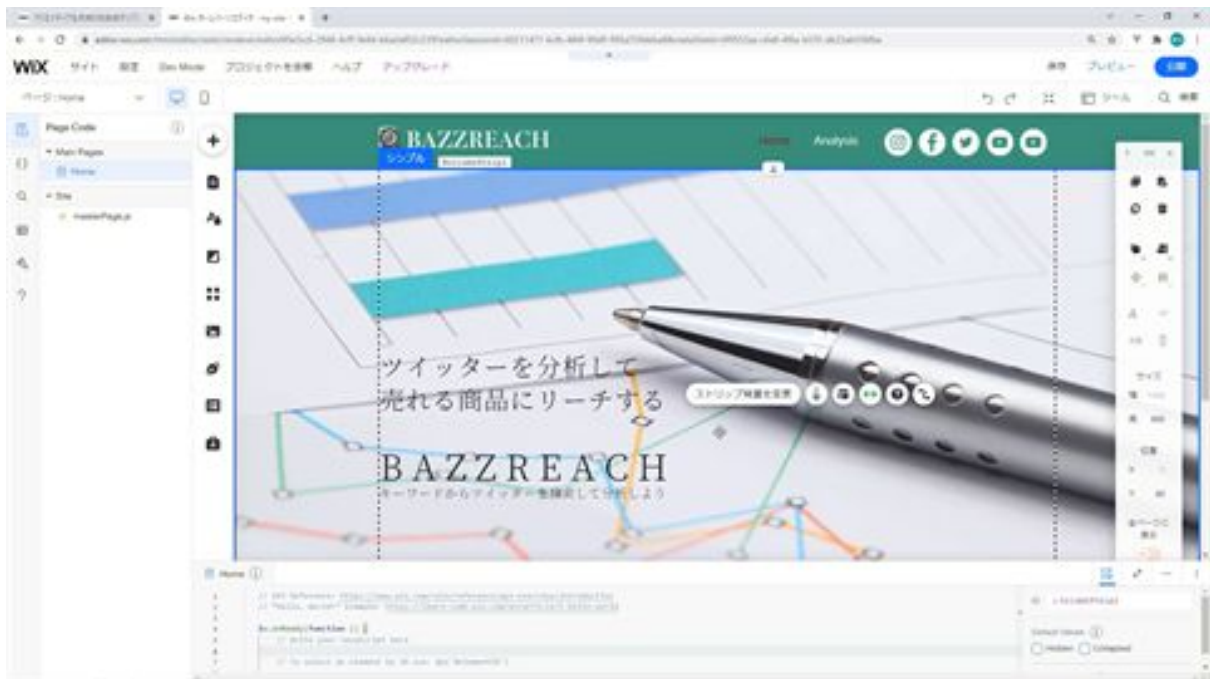
Next, we will change the font and font size of these three pieces of text. For the font, choose the Mincho font.



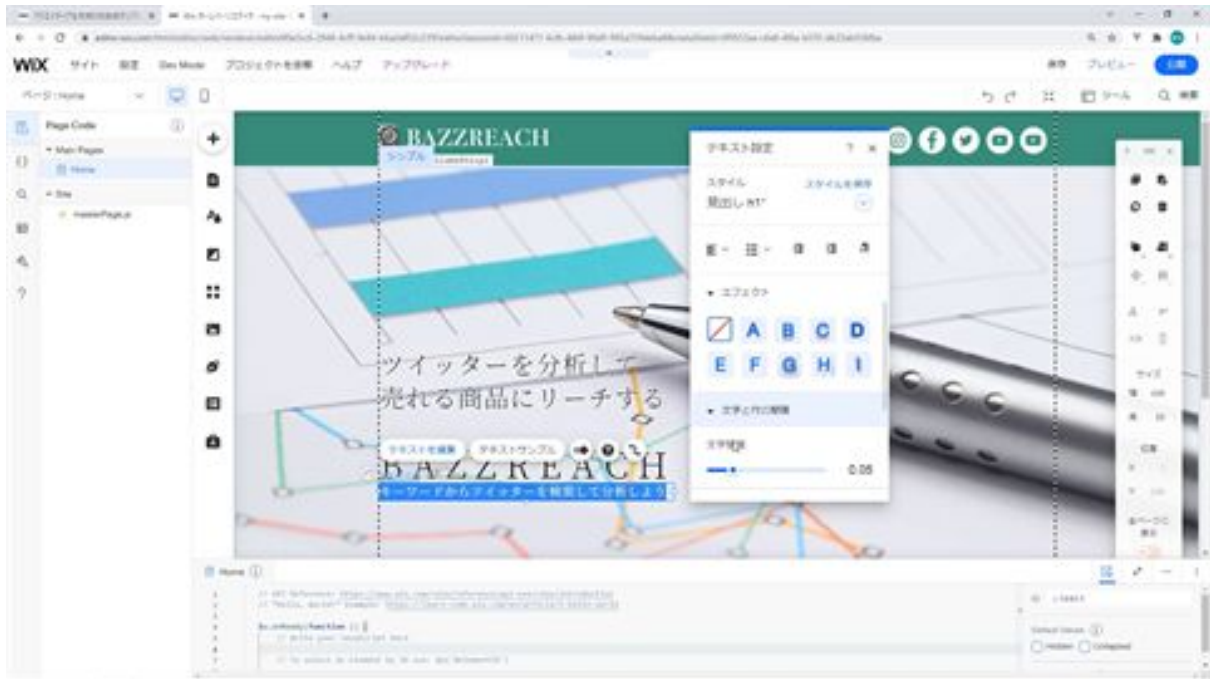
Next, we will modify the character and line spacing and the line spacing.



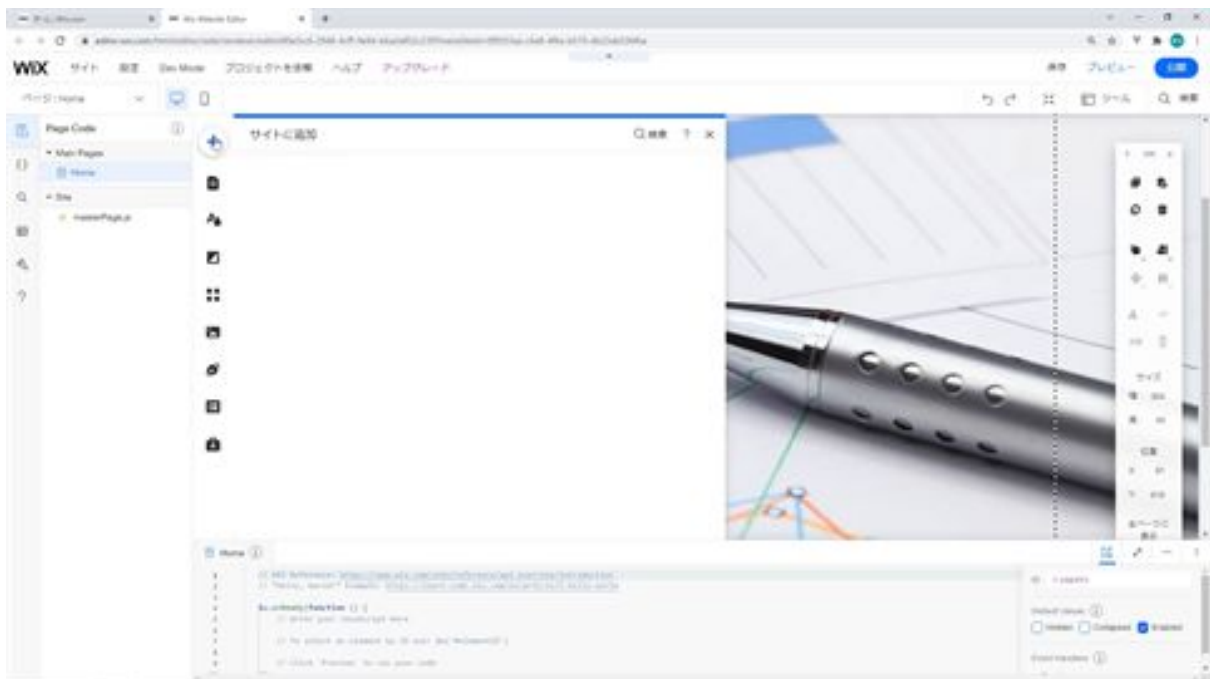
The next step is to modify the position of the copy and title.



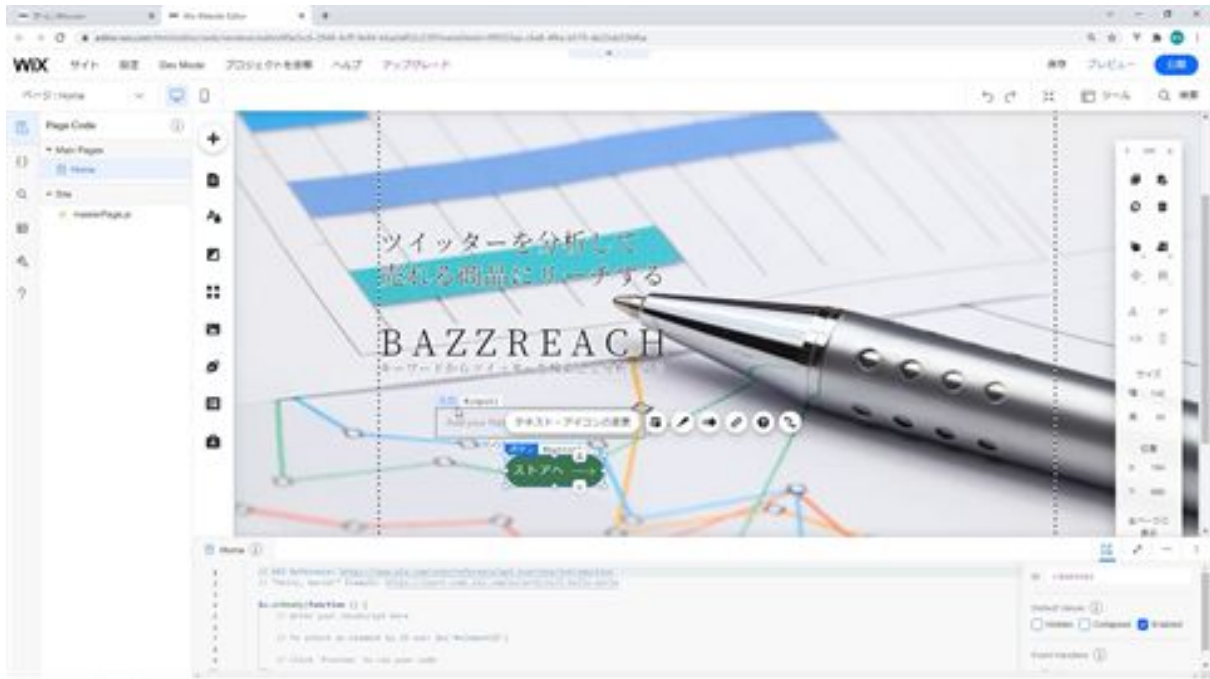
Next, we will add an effect to the text. We will apply the effect of the letter E to all the text.



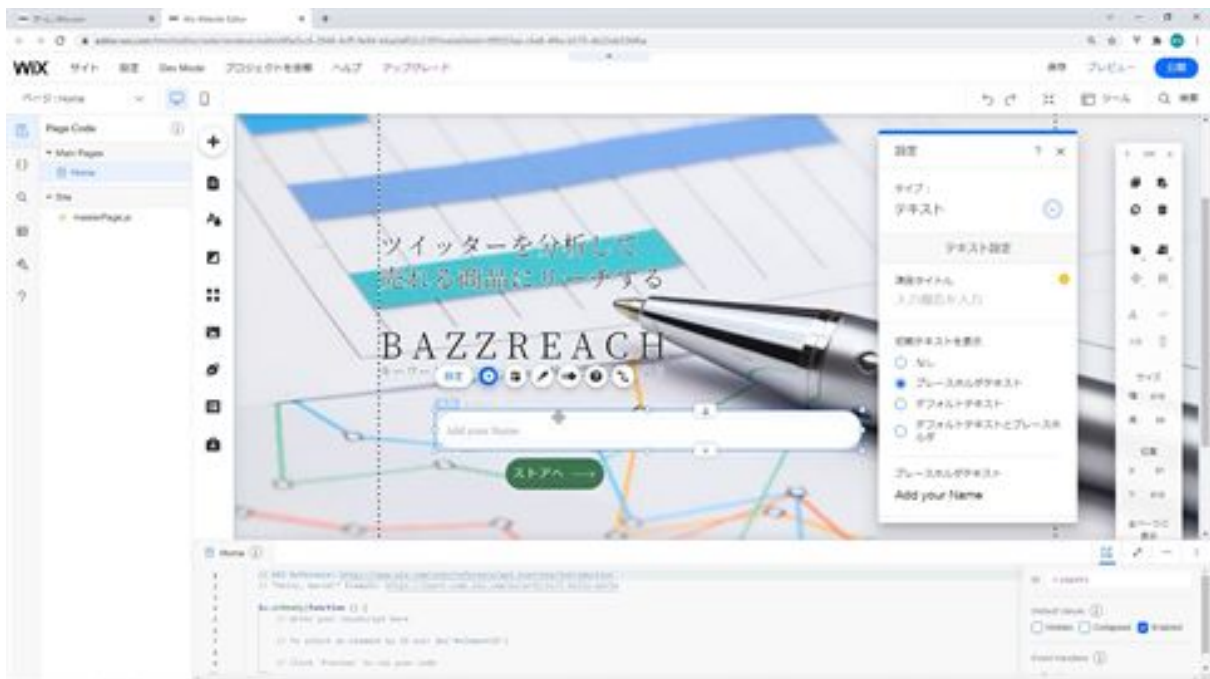
Next, we will add a text field for search and a search button. Select "Enter text" from the input field, and add the top text field this time.



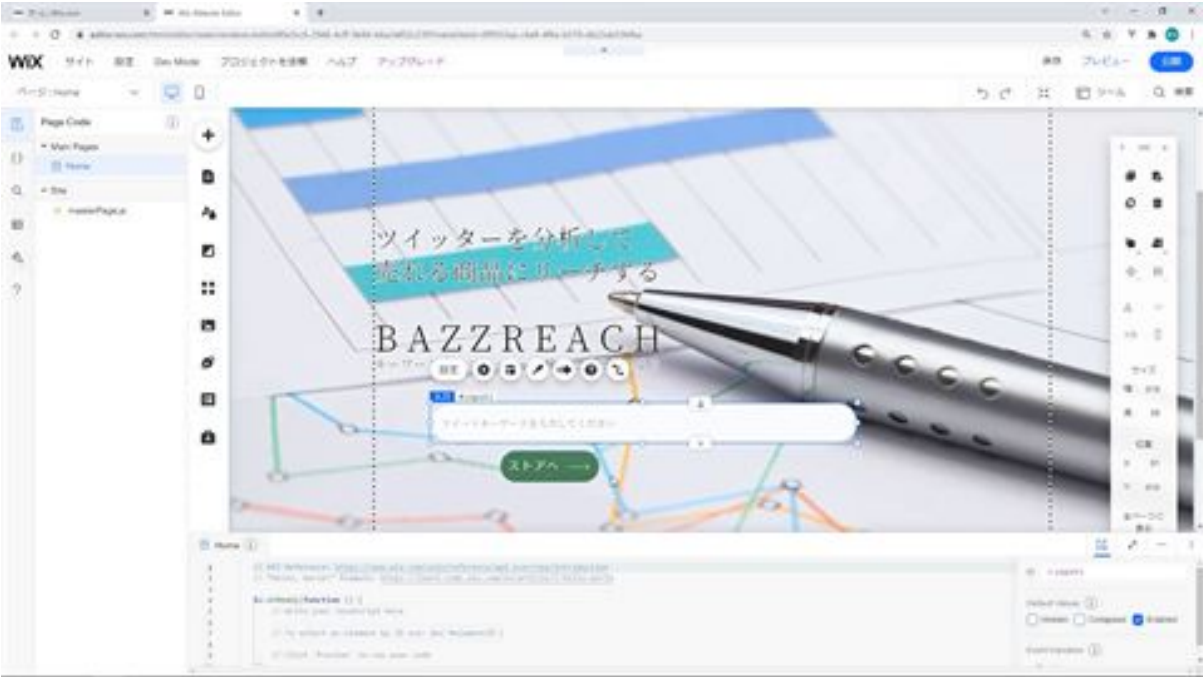
Next, choose a design from the buttons that matches your search button and add a button.



The next step is to modify the design of the text fields and buttons. We will modify the design from Customize Design. Modify the rounded corners to be 50px and the input font to match the title and copy above. Modify the width and height of the text field.



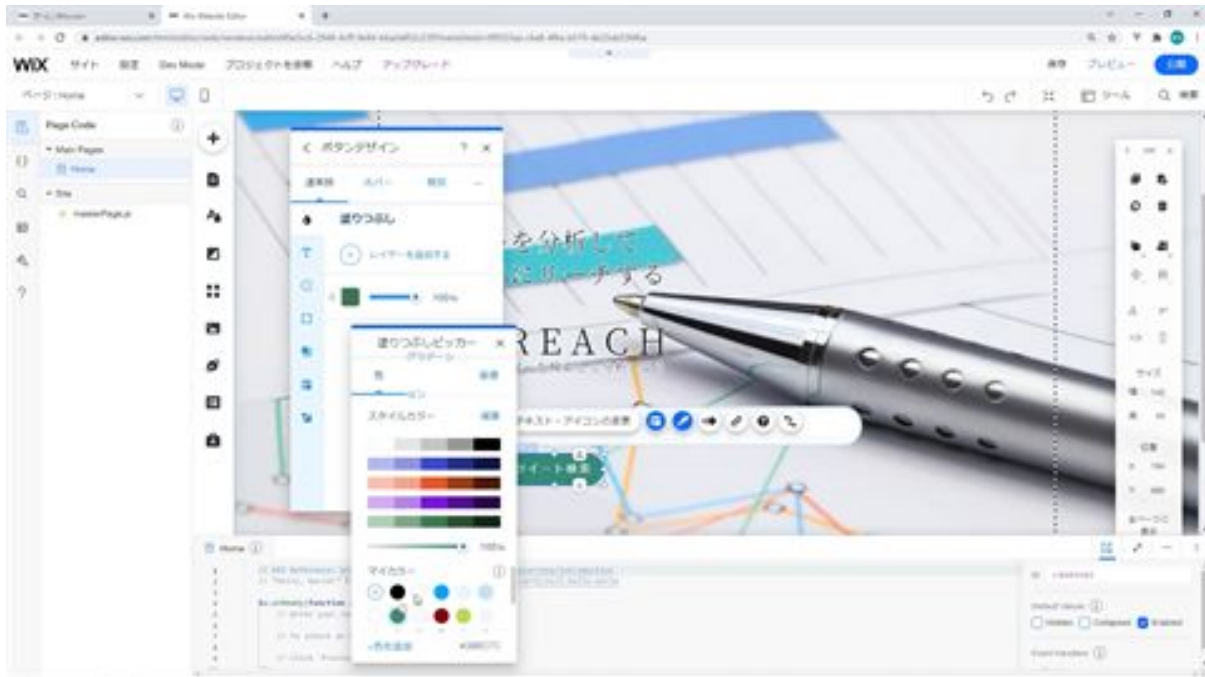
Next, we will modify the placeholder for this text. In this case, we'll modify the placeholder to say "Please enter a tweet keyword."



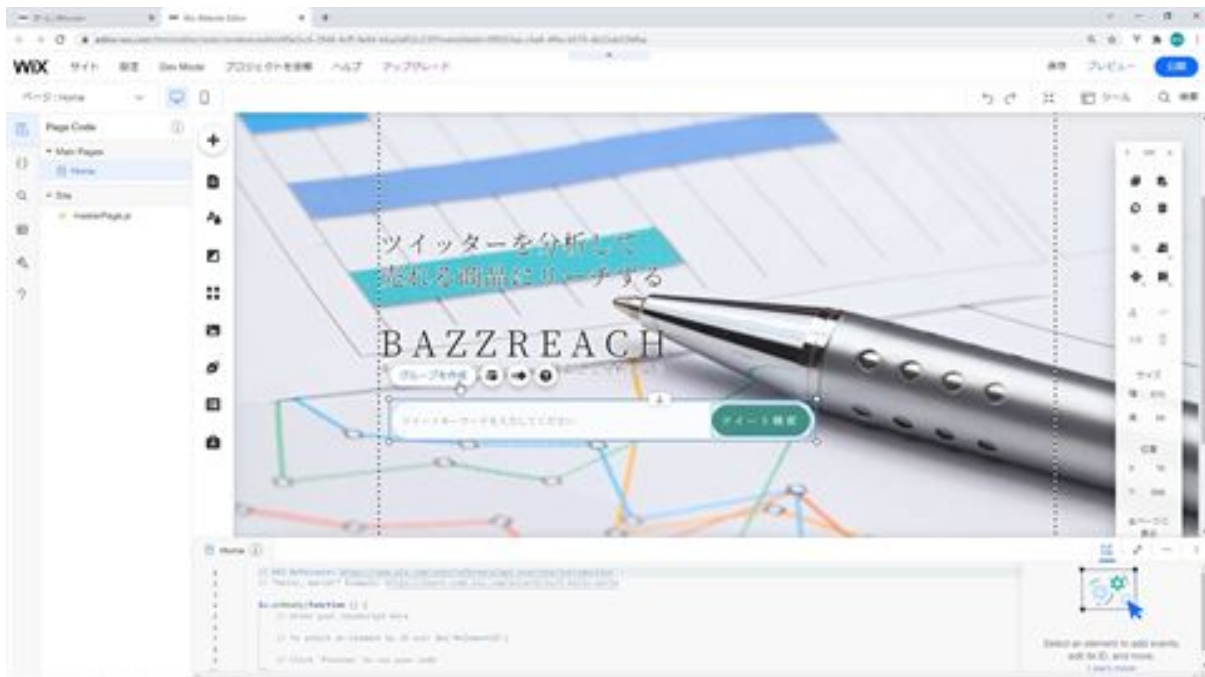
Next, modify the search button. Modify the text to be displayed as Tweet Search. It currently displays text and an icon, change it to text only.



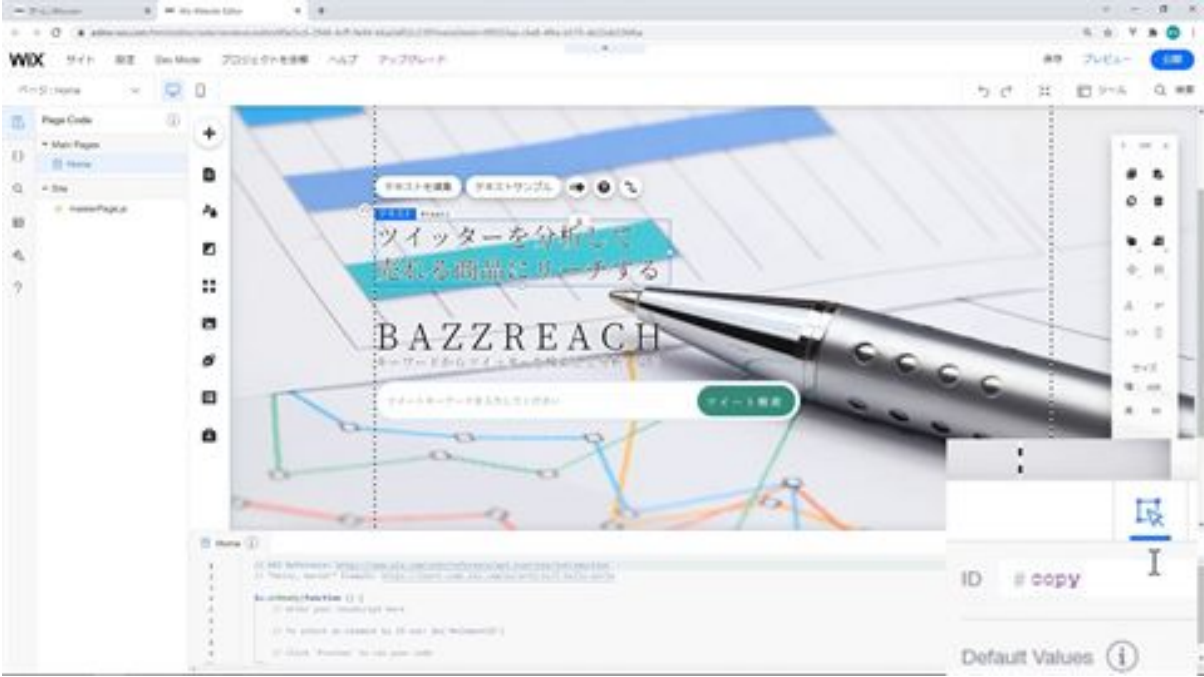
The next step is to change the button design.



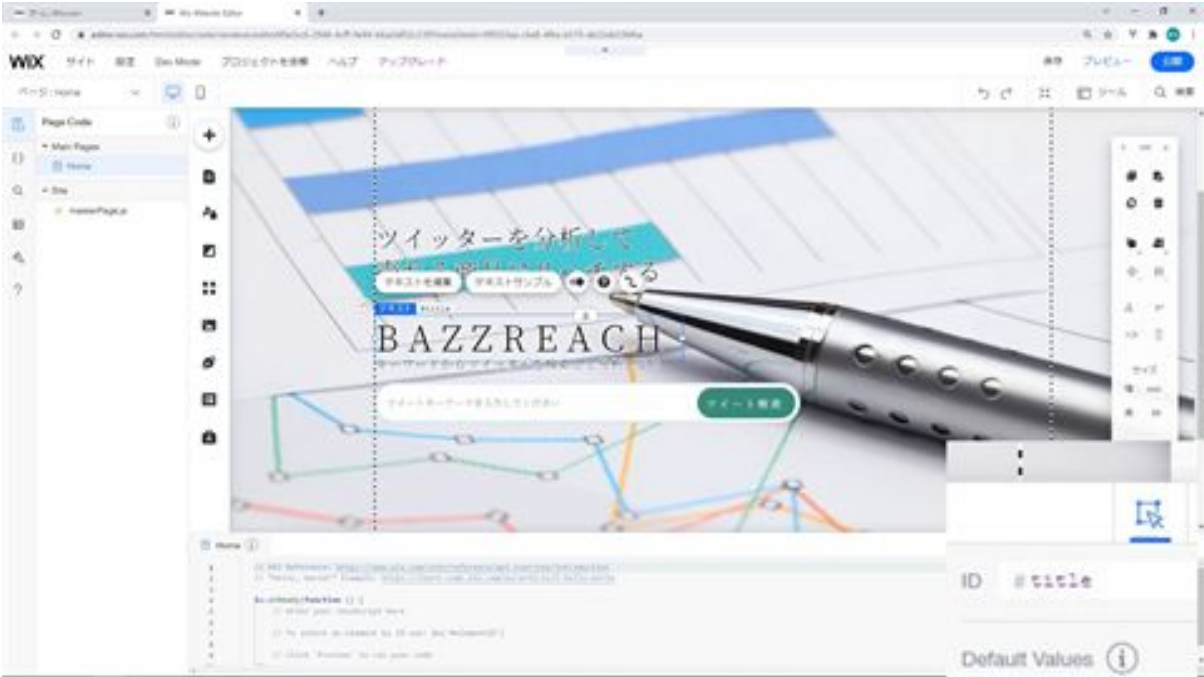
Once the search button is complete, we will group it with the text field. Select both the search button and the text field, and click Create Group.



Next, we will modify the position of the grouped objects. Finally, we will add IDs to the text, text field, and search button that we set up. The topmost copy will be given the ID copy.



The title will be given an ID of title.

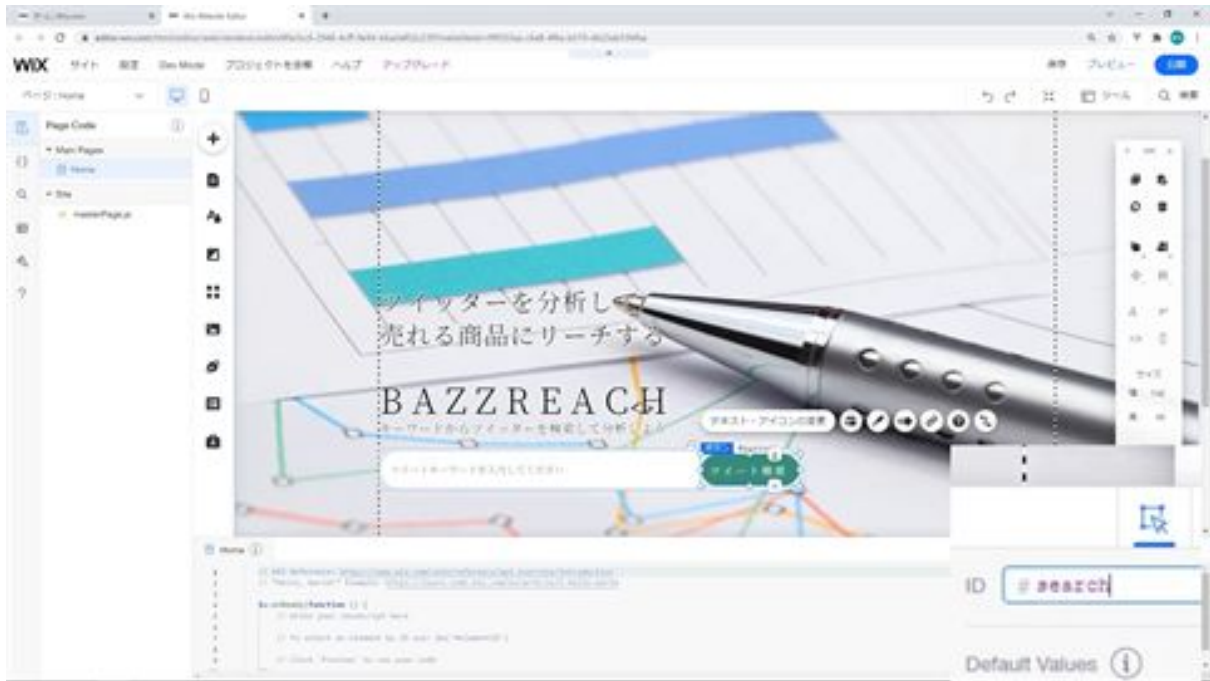


For the ID of the sub-copy, enter subCopy.

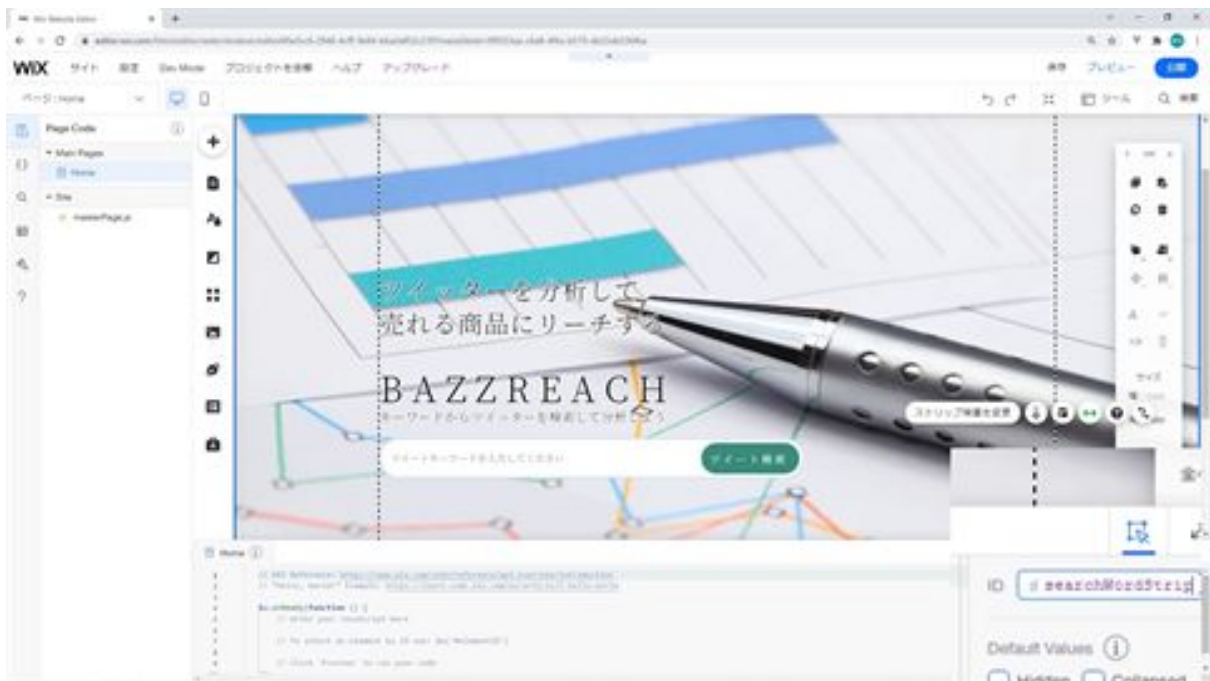


Set the ID searchWord for the text field and set the ID search for the button.



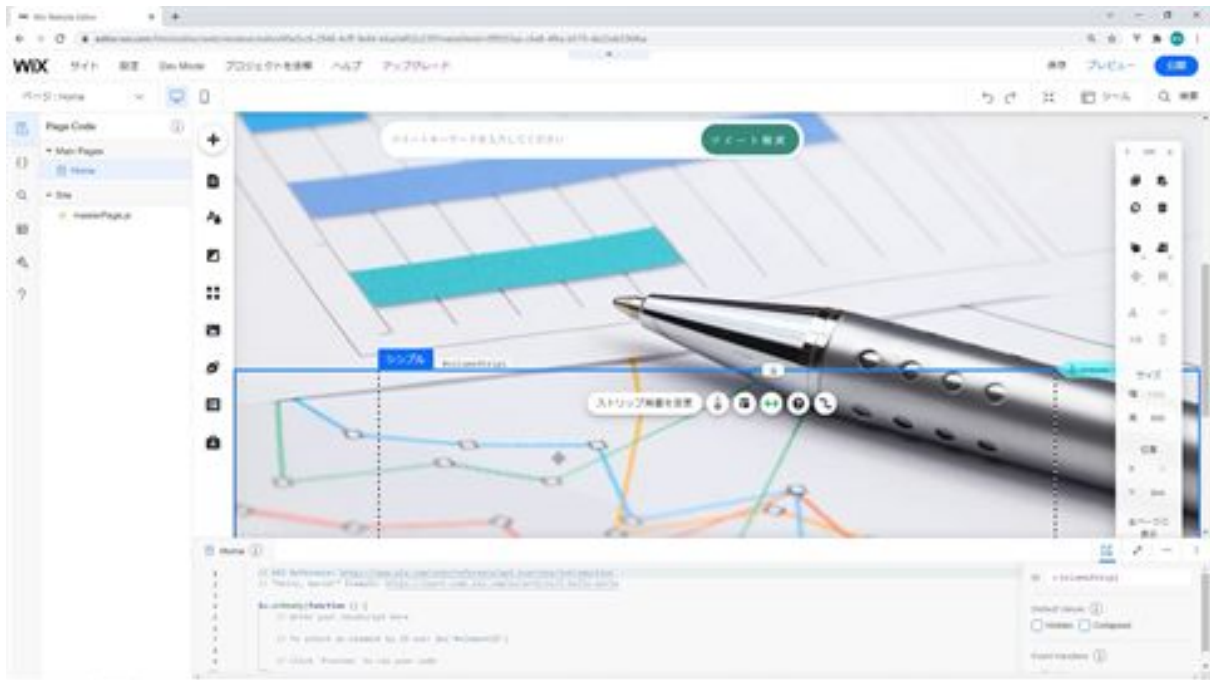


Finally, enter searchWordStrip as the ID of the strip for the search screen. That's all there is to create a strip for the search screen.

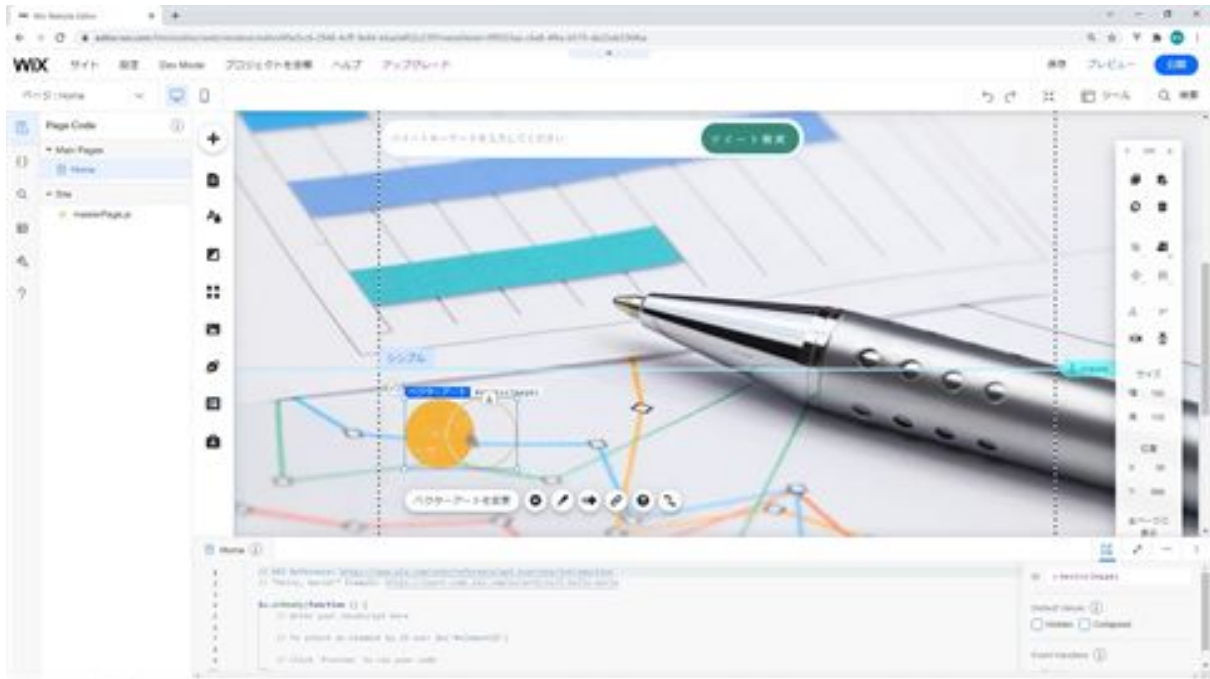


Let's create a strip for saving tweets

The next step is to create a strip for saving tweets. First, add a new strip. Next, change the height of the strip to 650px. Once the height is changed, change the strip to transparent.



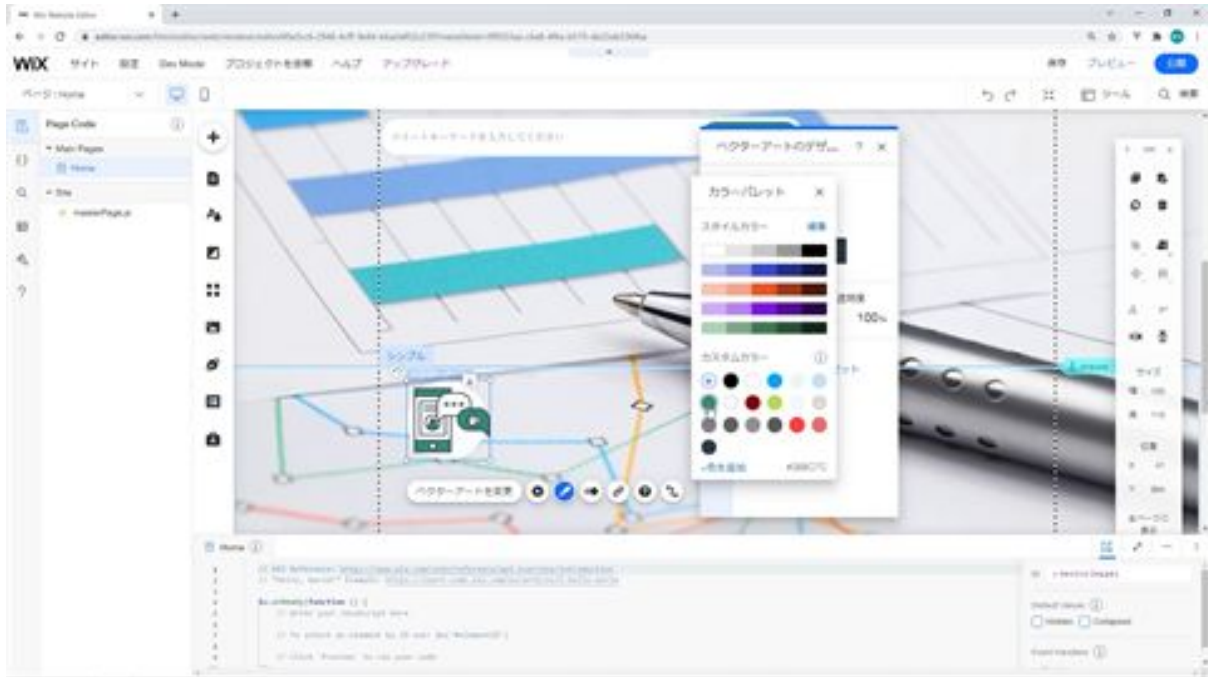
Now let's add the components to the strip. First, we'll add some vector art here.



Once you have added the vector art, you can change the vector art. Enter "communication" in the search field and search. You will see Communication at the top of the list, so select it and change the vector art.



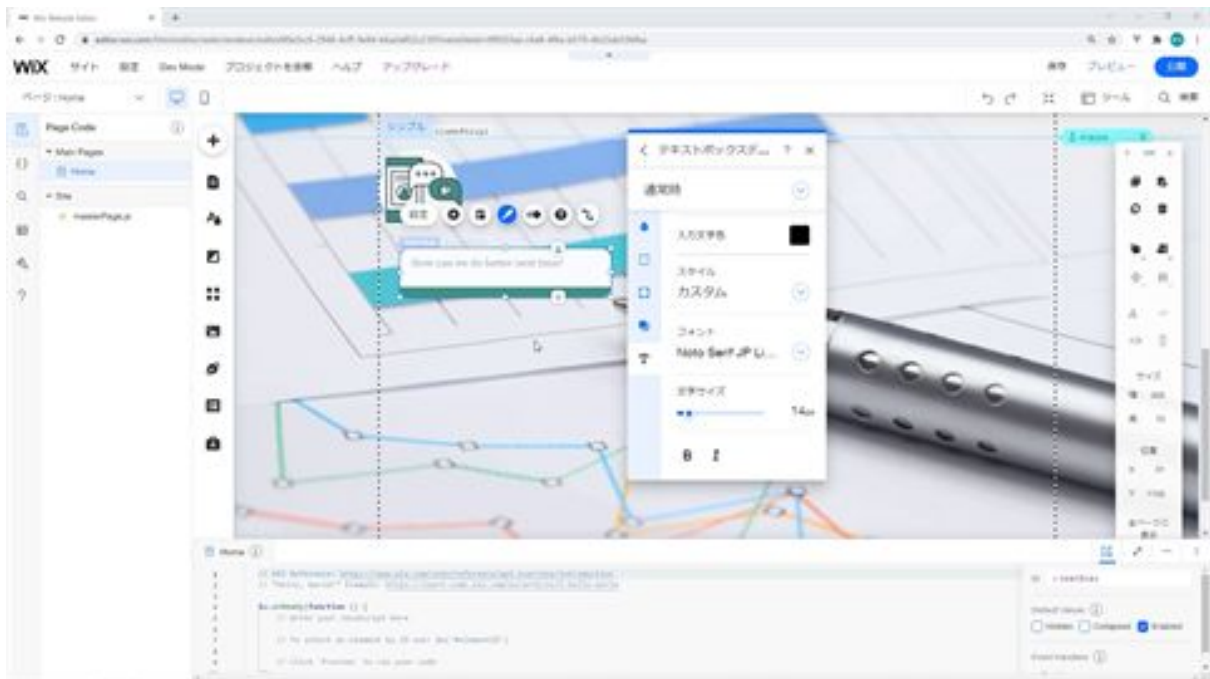
The vector art has been changed. The next step is to change the size of this vector art. Change the width of this vector art to 120px. Next, we will change the design of this vector art. Click on the Change Design button and change the red color to #388C7C green.



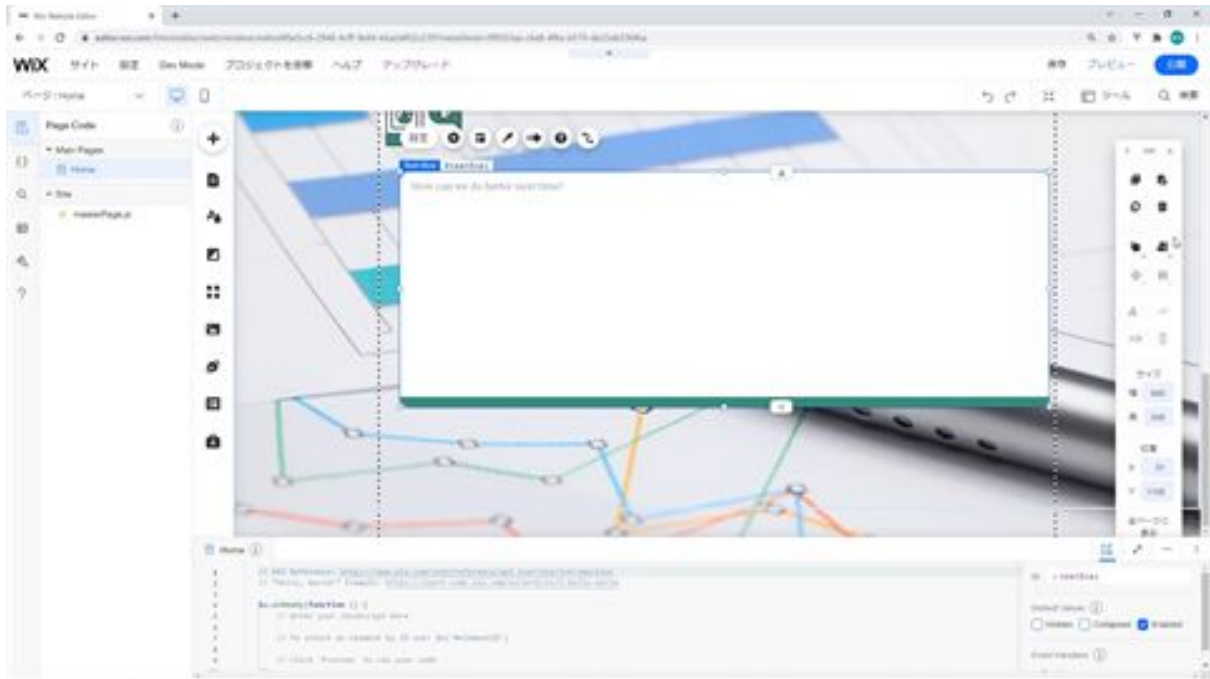
Now we need to change the position along with the guideline. Next, we need to add a text box below it. Select the text box from the input field and add it.



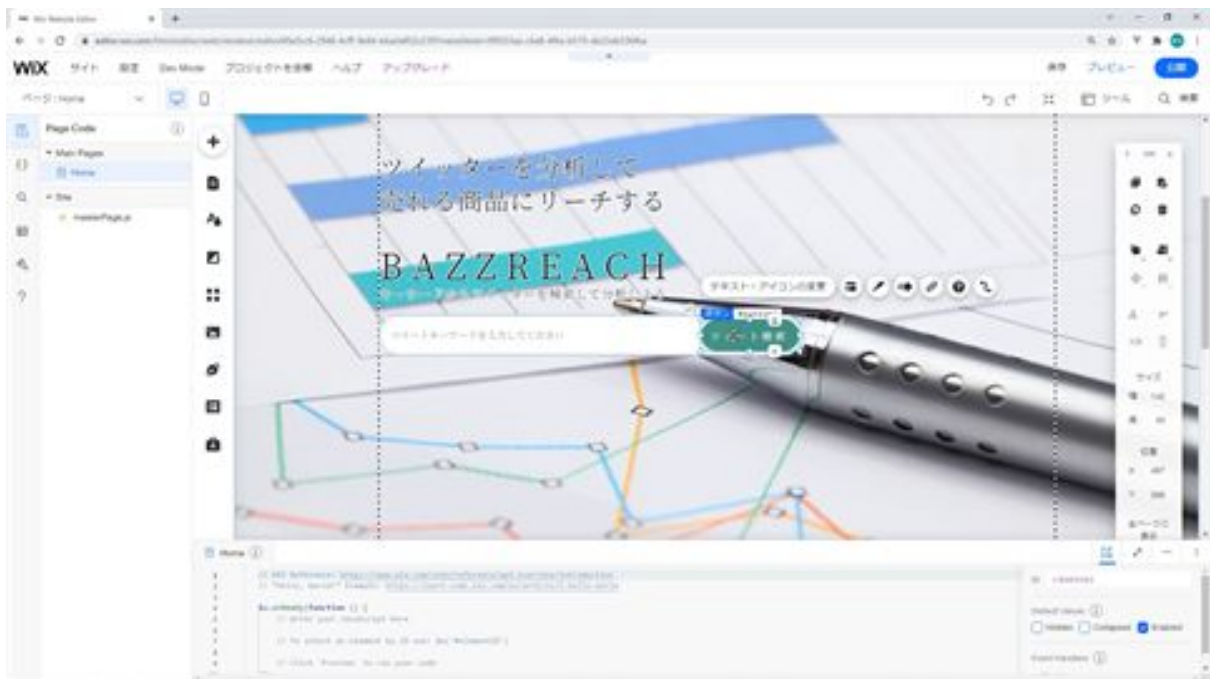
After the text box is added, change the design.



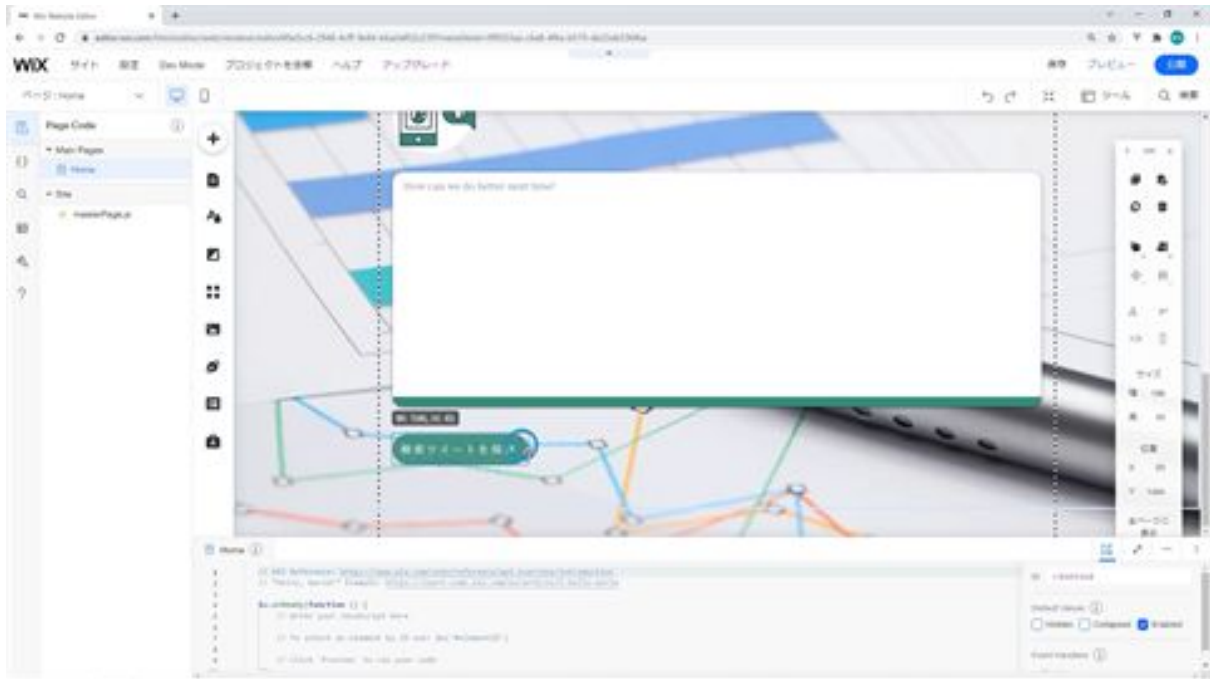
After you have finished modifying the design, adjust the width and height of the text box. Now we can adjust the position.



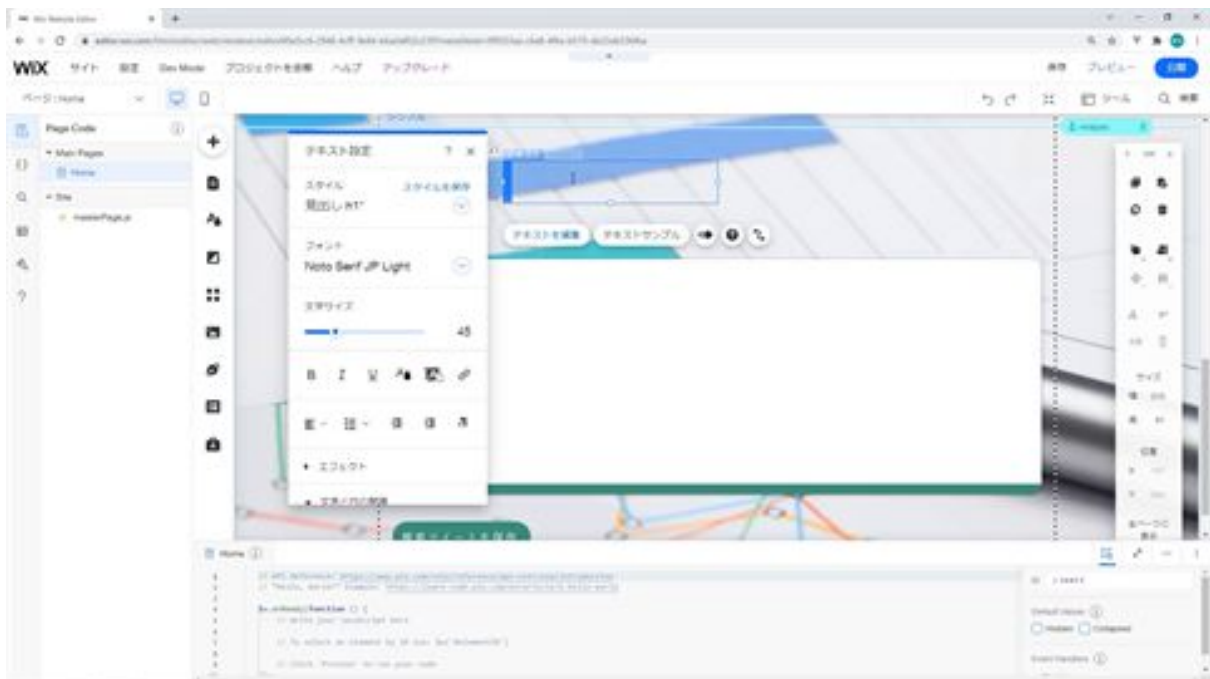
The next step is to add a button here to save the tweet. The button has already been added, so we'll copy it and make it.



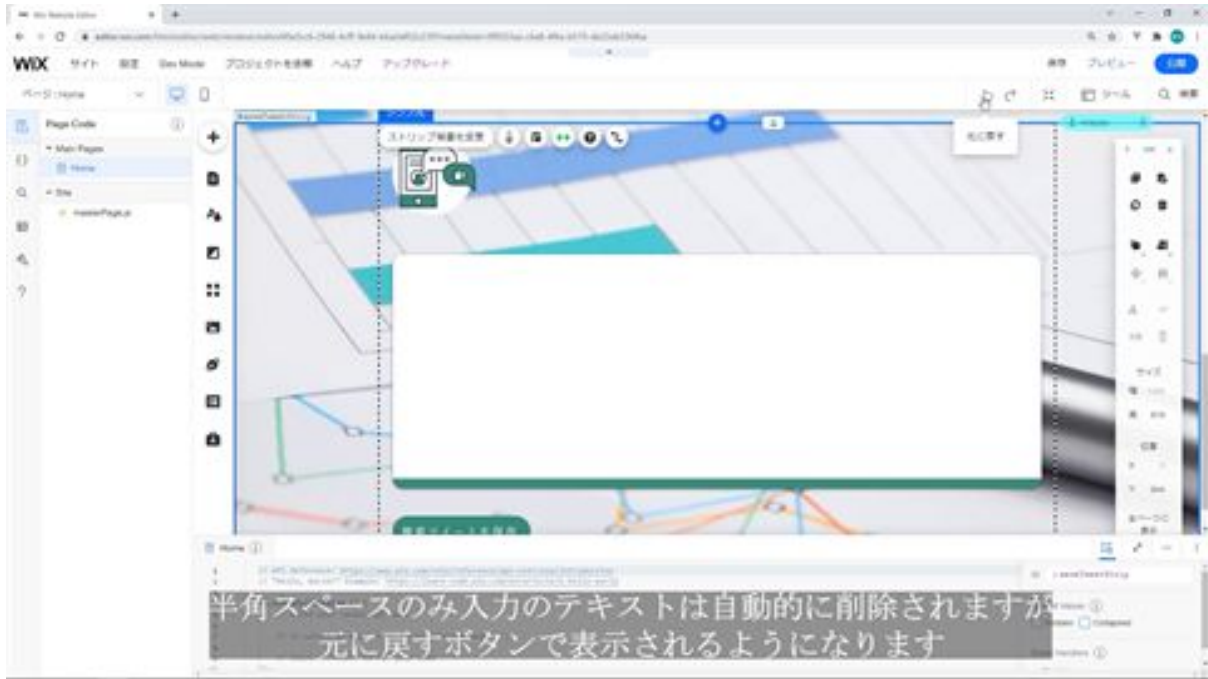
Modify the text of this button to Save Search Tweets. Adjust the size of the button so that all the text is not displayed.



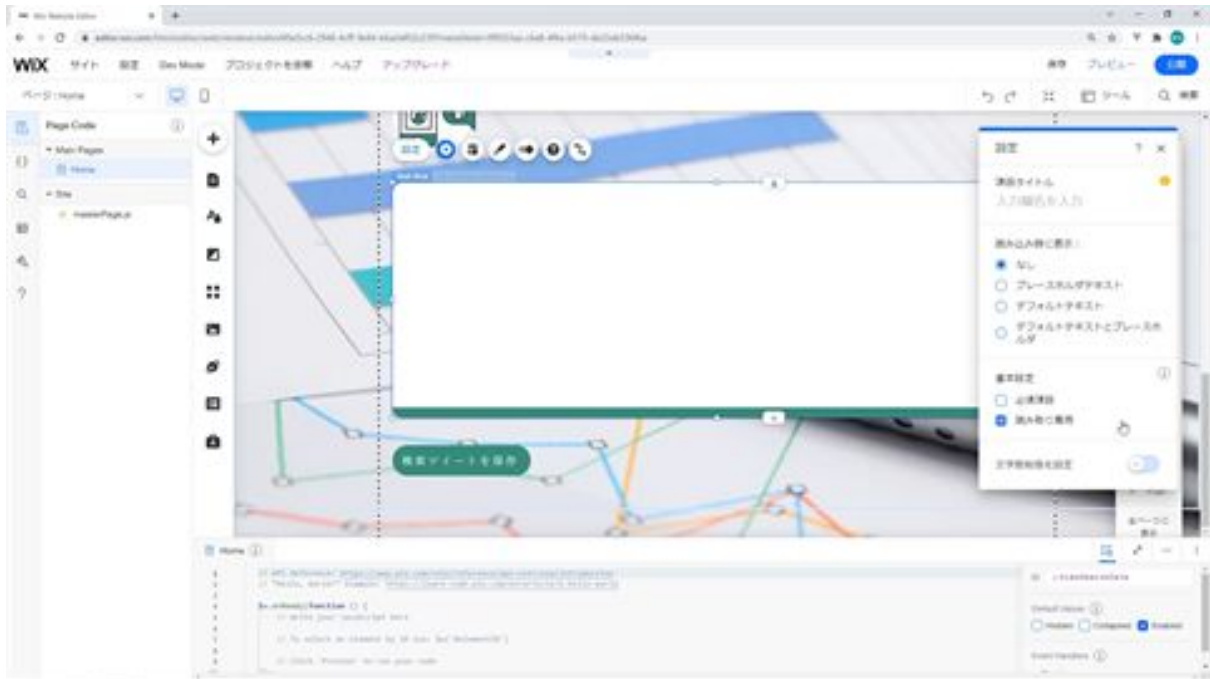
The next step is to add the text to display the search keywords. Select the text from Add and modify its position. Next, edit the text you have added. Set the font to match the other fonts, the font size to 45px lines, and the spacing to Customize. Then enter only spaces in the text.



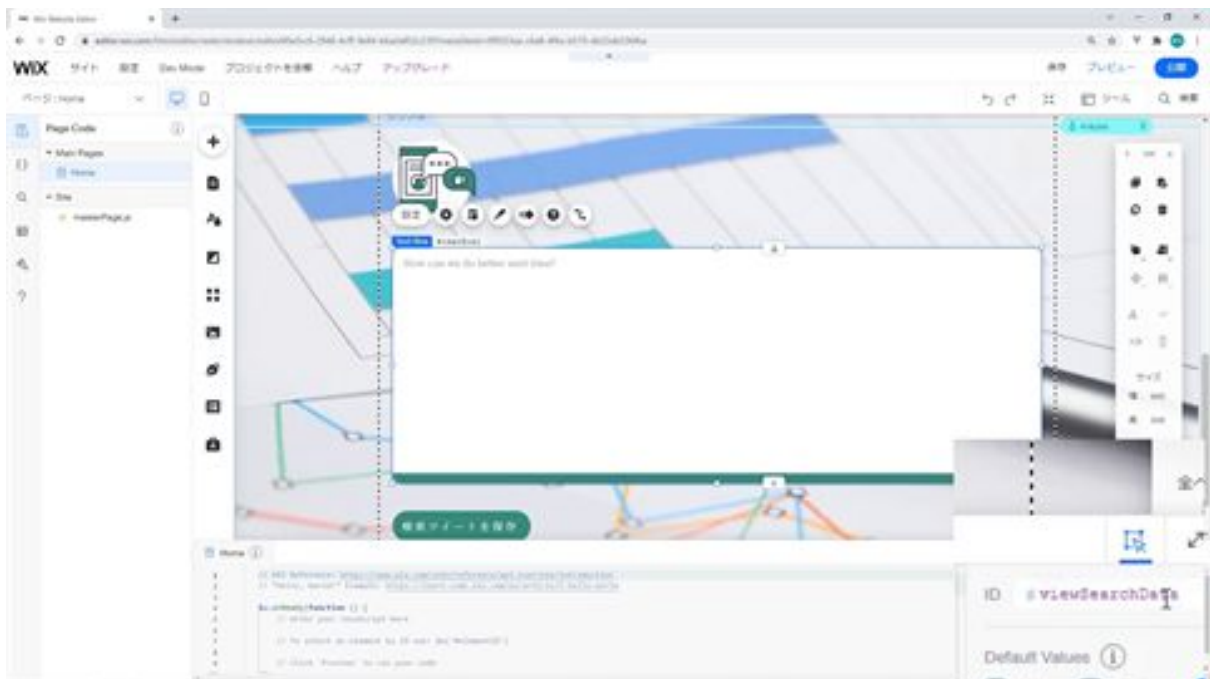
Text with only half-width spaces will be automatically deleted but will be displayed with the Undo button.



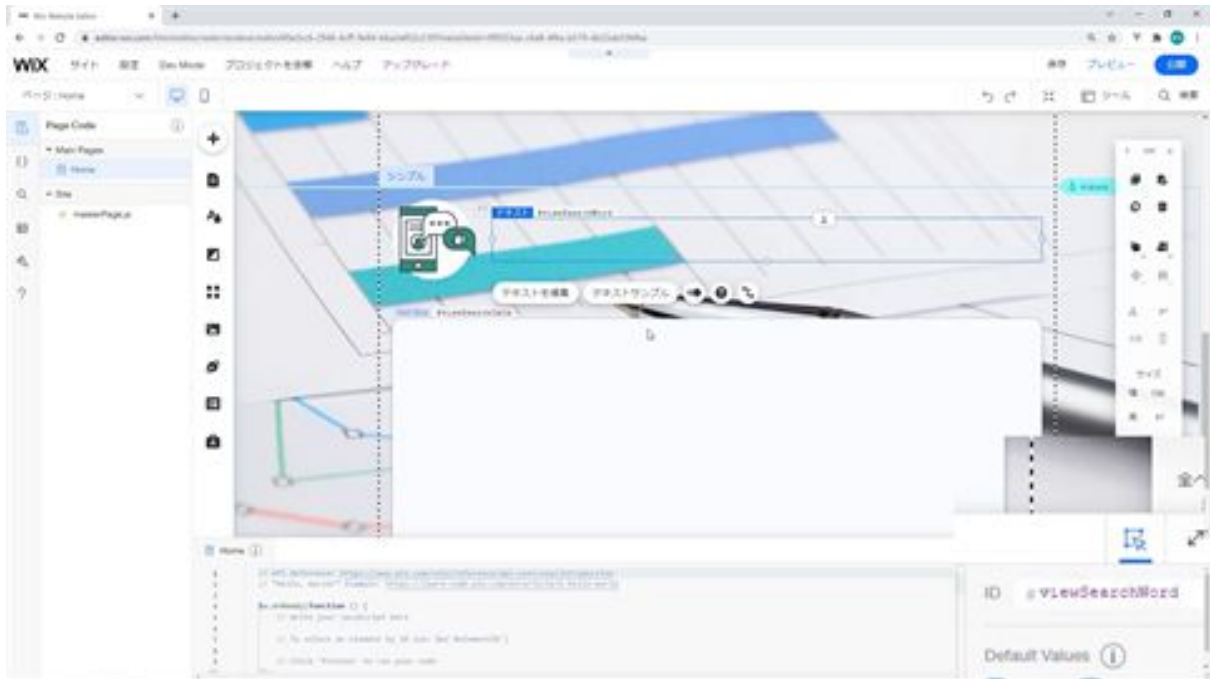
Modify the width and height to correct the position. Next, we will change the text box settings. Click on the Settings button and select No Display on Read. Check the Read-only checkbox in the Preferences.



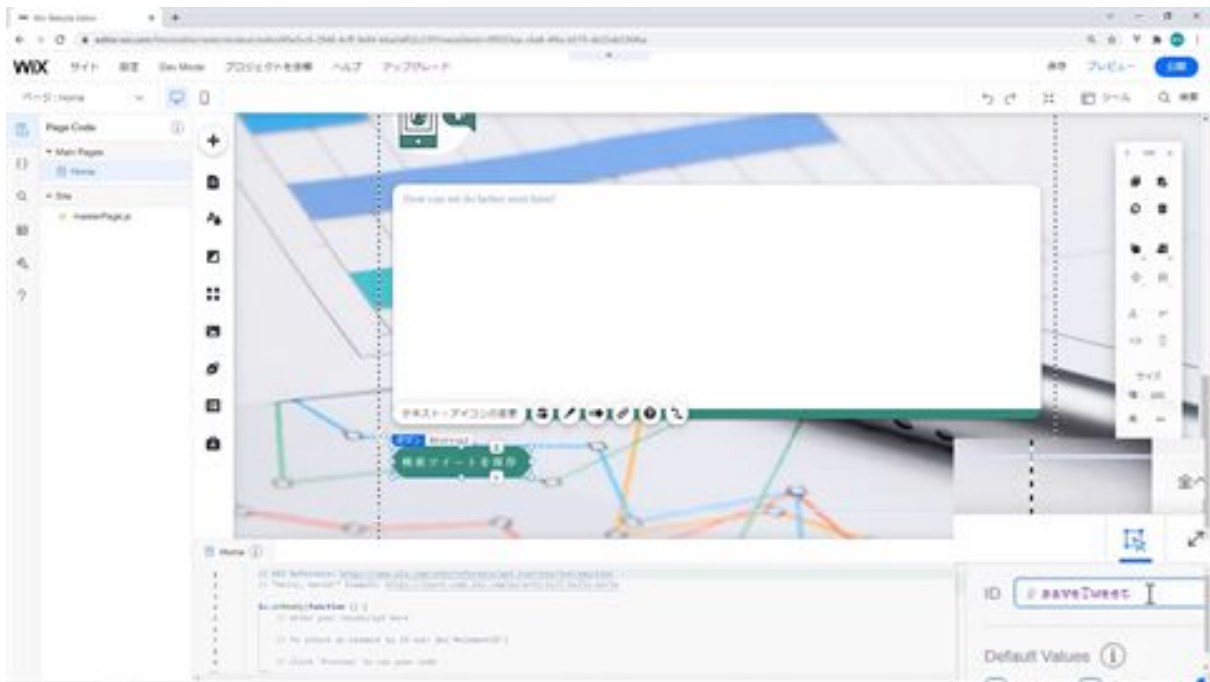
Next, we will add an ID to each component. We will start with the text box and type in viewSearchData.



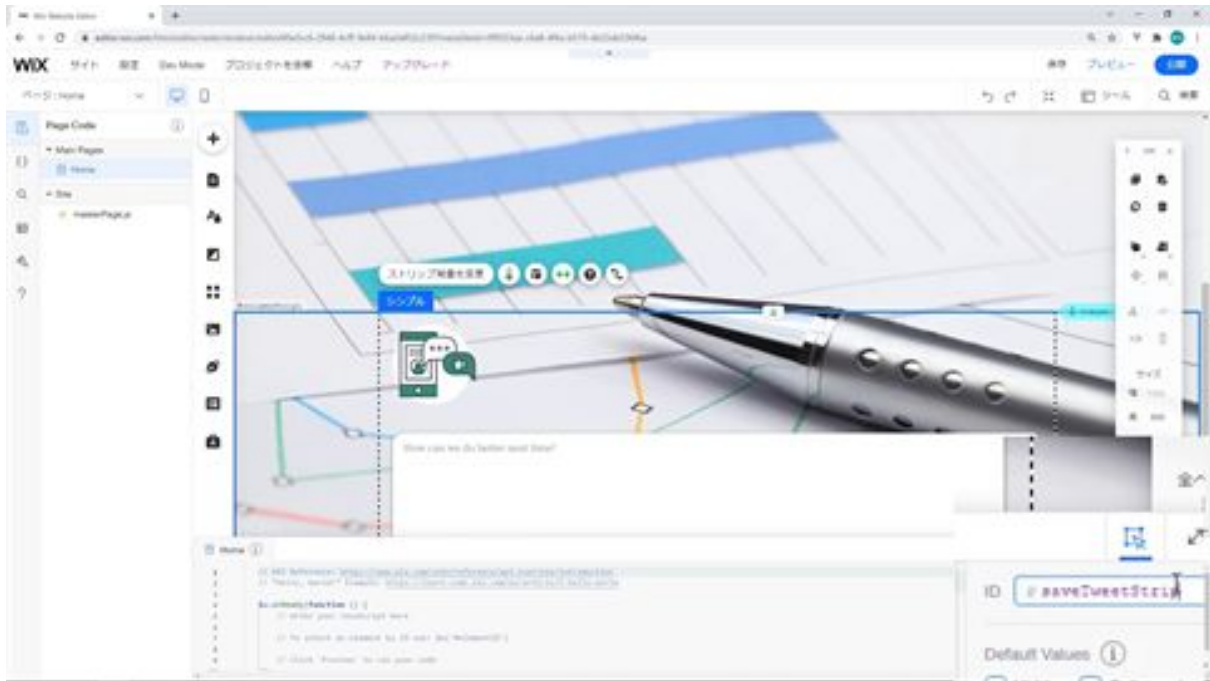
Next, change the ID of the search keyword text to viewSearchWord.



Next, enter saveTweet and ID in the button for saving.



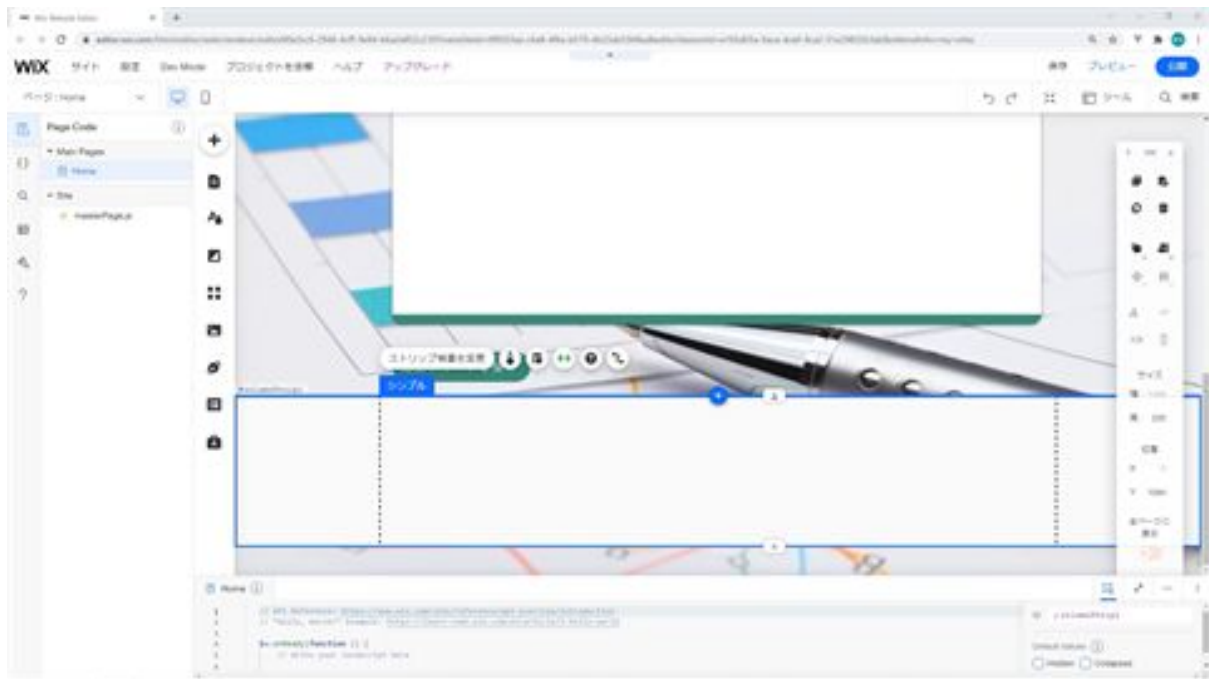
Finally, type saveTweetStrip in this saving strip.



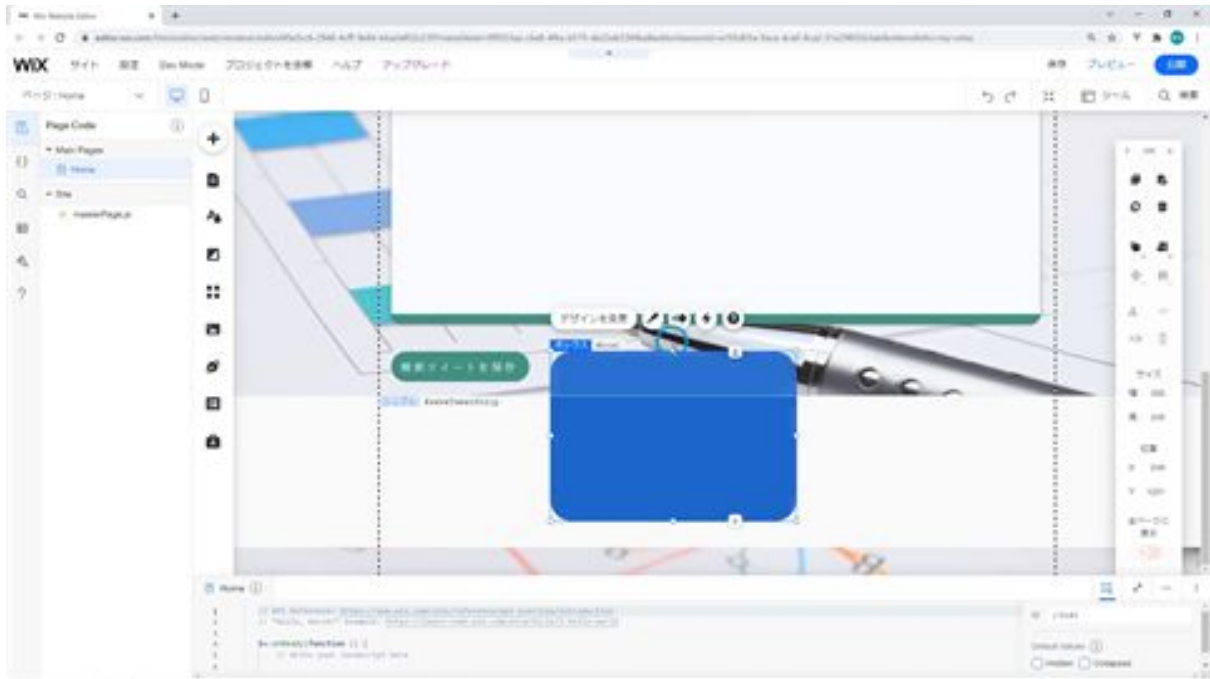
That's all there is to create a strip for saving tweets.

Let's create a strip for searching saved tweet data

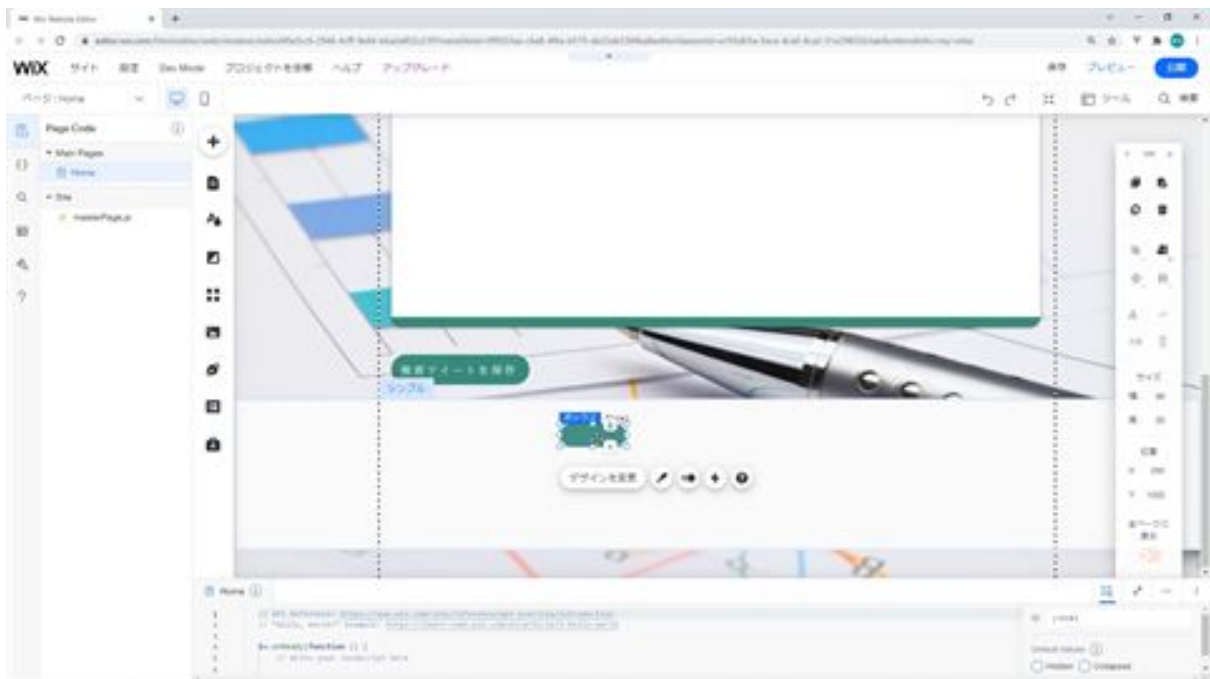
Next, we will create a strip for searching the saved tweet data. First, we will add a new strip. Drag and drop it to the bottom. Next, change the height of the strip. Change the height to 220px.



Next, add a box and text to the top part of the strip.

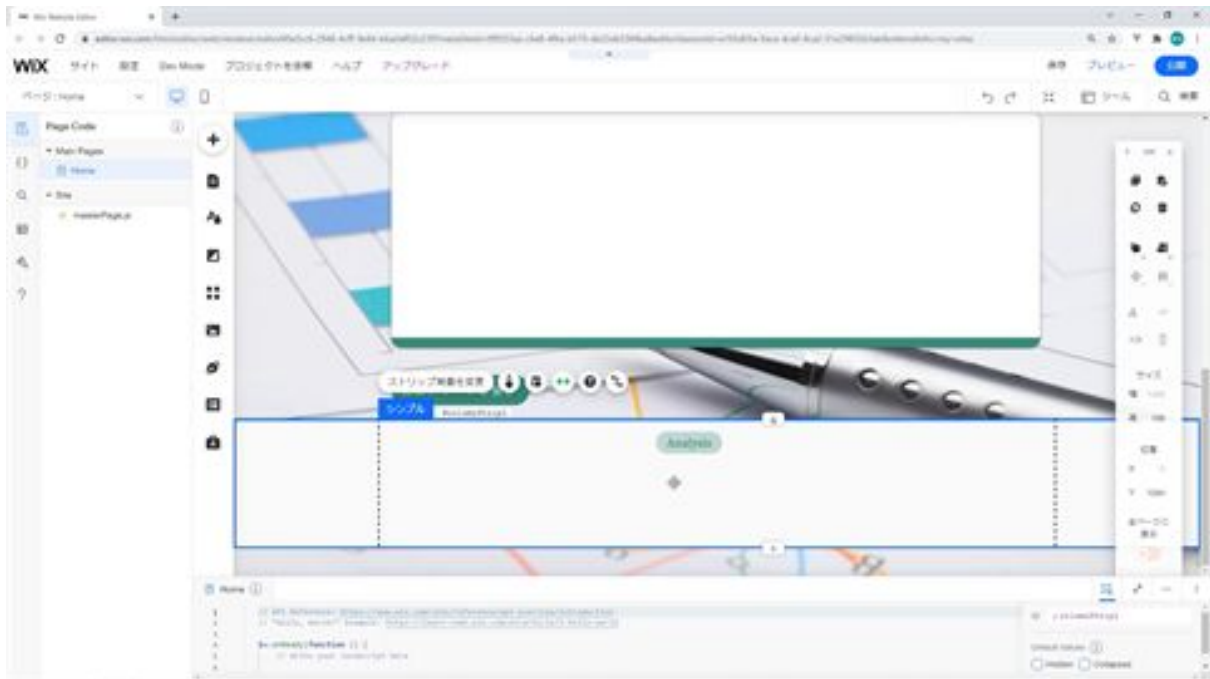


Make the box smaller and add it. Then change the design. Change the background to #388C7C, and change the border to #E6DECA. Leave the rounded corners and shadow as default. Then change the width and height. Change the width to 96px and the height to 32px.



Next, add some text on top of this. Edit the text to match the font to the other font. Change the font size to 16px, make it bold, and center it. Select #388C7C as the text color. Then change the text to Analysis.

Since the color of the box and the text are the same, change the color of the box. Change the opacity to 30% and place the text on top of the box. After selecting the box, select the text and click on Center vertically and Center horizontally respectively. Now that the text is centered, move the box.



This will move the box to the center of the screen. Next, add the text for the search display below this. In this case, we are using three texts. Search Data", a text that displays the number of searches, and finally a text that says "Found". In practice, this means that the value of the search count will change.



The next step is to place a text field and a button for the search. We have the text field and button that we created last time, so we will copy and paste them into the bottom part of the page.



After placing the button, change the text of the button from "Tweet Search" to "Search". Then change the width of the button to 96px. After changing the width, move it to the right to match the text. Next, change the settings of the text field. Change the placeholder text to "Please enter a search term".



The text field for entering search keywords is located at the bottom, so modify its position.



From this strip is where Analysis will be displayed, so align the anchor with Analysis.



Next, we will change the IDs of the components we are placing in the strip. First is this strip, change the ID of this strip to bazzSearchStrip.



The next part is the search count, change the ID of this text to bazzSearchDataCount count.



And the text field, change the ID of the text field here to keyword.



And for the search button, change the ID of the search button to bazzSearch



This is the end of the strip for searching the saved tweet data.

Let's create a strip for the data table

Next, we will create a strip for the data table. First, we will add a new strip. Then we will add the data table. Select a table in the list from the Add button, and this time we will add the first table.



Change the design of the table so that it is centered. In Customize Design, change the header background, previous/last column, hover background, row/cell selection, border color, and column separator color to #388C7C.

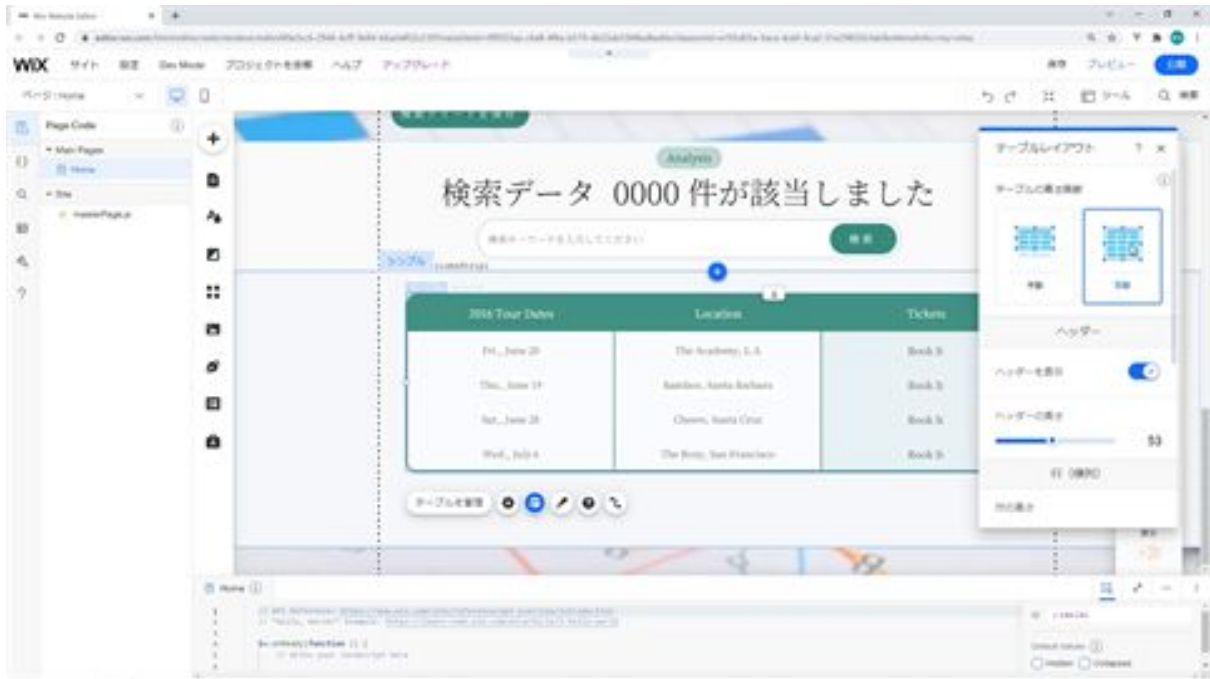
You do not need to modify the page feed. Change the roundness of the corners to 15px. Then turn on Show shadows. Next, change the font of the text to match the rest of the text. Then change the color of the link to #388C7C.



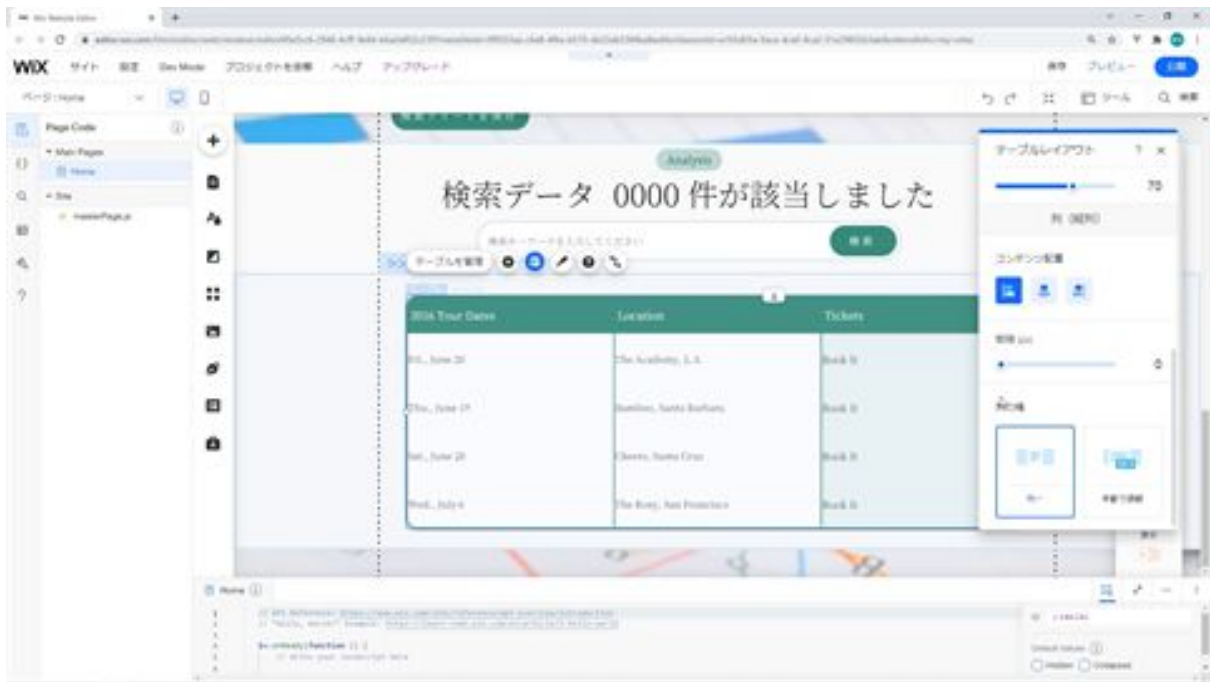
Next, change the color setting of the link on hover to #388C7C.



The next step is to change the width of the table. Change the width of the table to 900px. Then move it to the center. Next, change the layout. In the layout, change the table height adjustment to automatic.



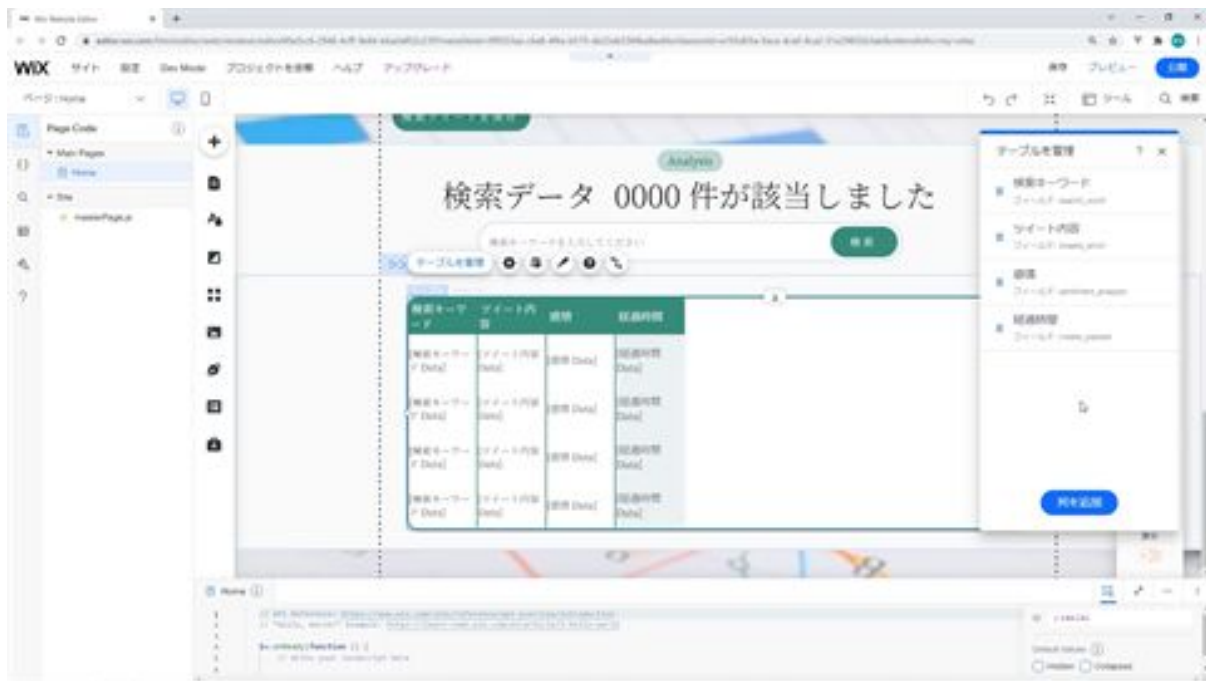
Next, change the row height to 70px. Change the content alignment to left-aligned and select Adjust column width manually. Then click the Apply button.



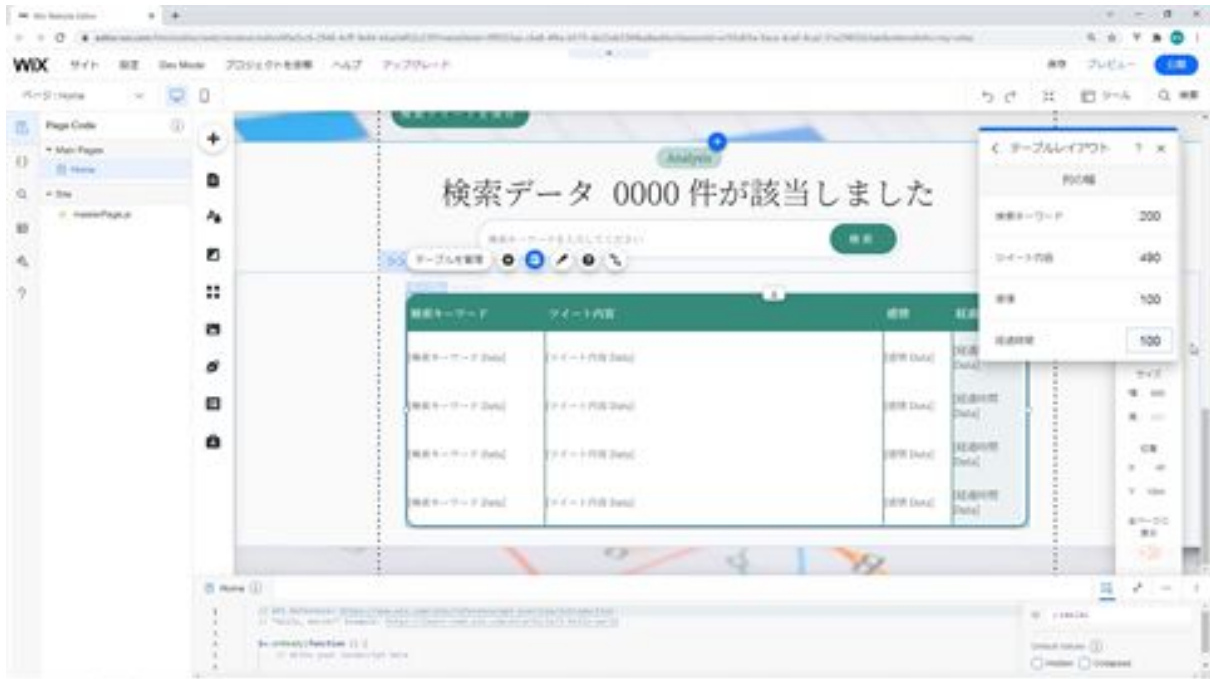
Then you can specify the width of the columns. As for the column width, since we want to add four items here, we will add the items first and then adjust the columns. Double click on the table to display the table management, and set the items you want to display here.

The items to be displayed are search keywords, tweet content, sentiment, and elapsed time. You can add the search terms, tweet content, sentiment, and the elapsed time to this label by clicking the Add column button.

Then set a field for each of these columns. For the search keyword field name, enter search_word and click the Done button. We will modify the other fields in the same way. For the tweet content, enter tweets_short. For the sentiment, enter sentiment_analysis. Enter create_passed for the elapsed time.



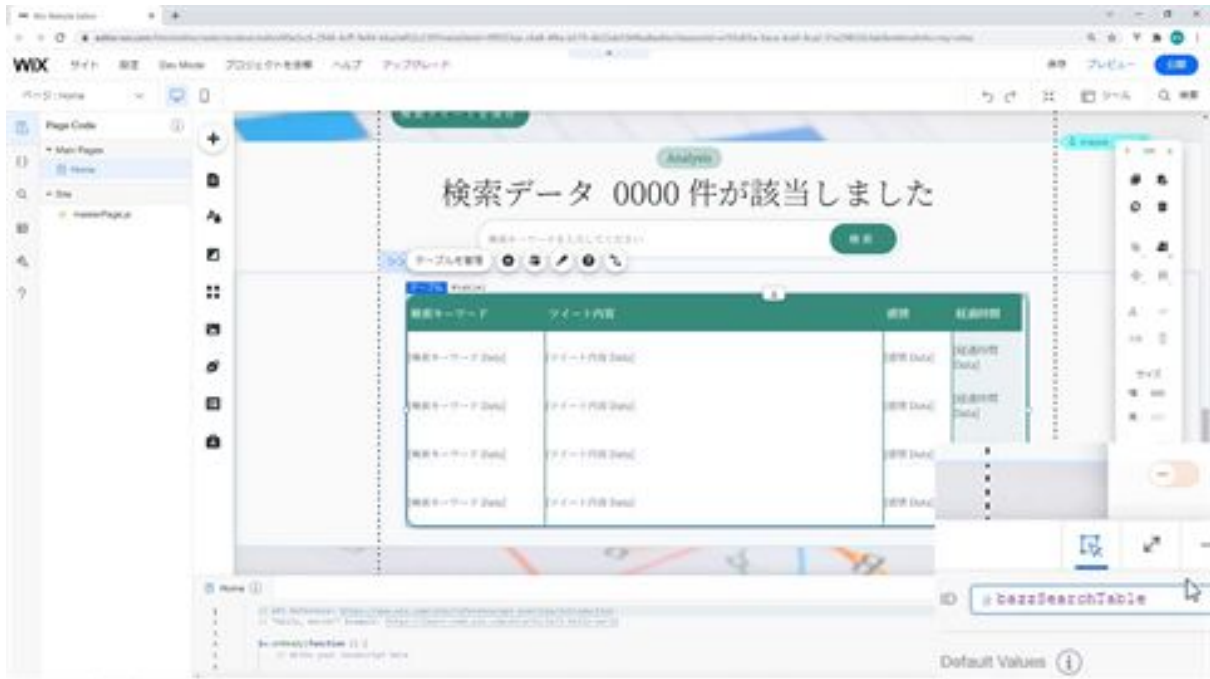
Next, modify the column widths in Design. From Layout, select Adjust Manually at the bottom and click the Apply button to determine the width of each column. We will set 200px for Search Keywords, 490px for Tweet Content, 100px for Emotions, and 100px for Elapsed Time.



Next, change the ID of the strip and the ID of the table. We will change the ID of this strip to `bazzSearchTableStrip`.



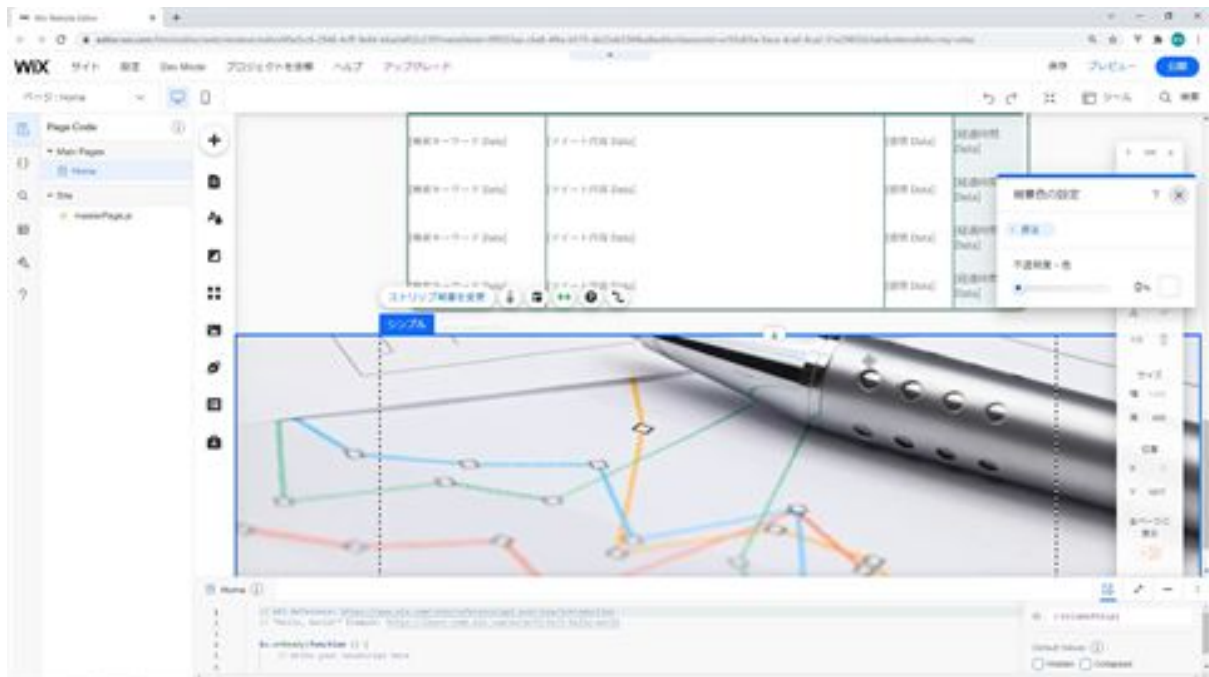
This table will then change its ID to `bazzSearchTable`.



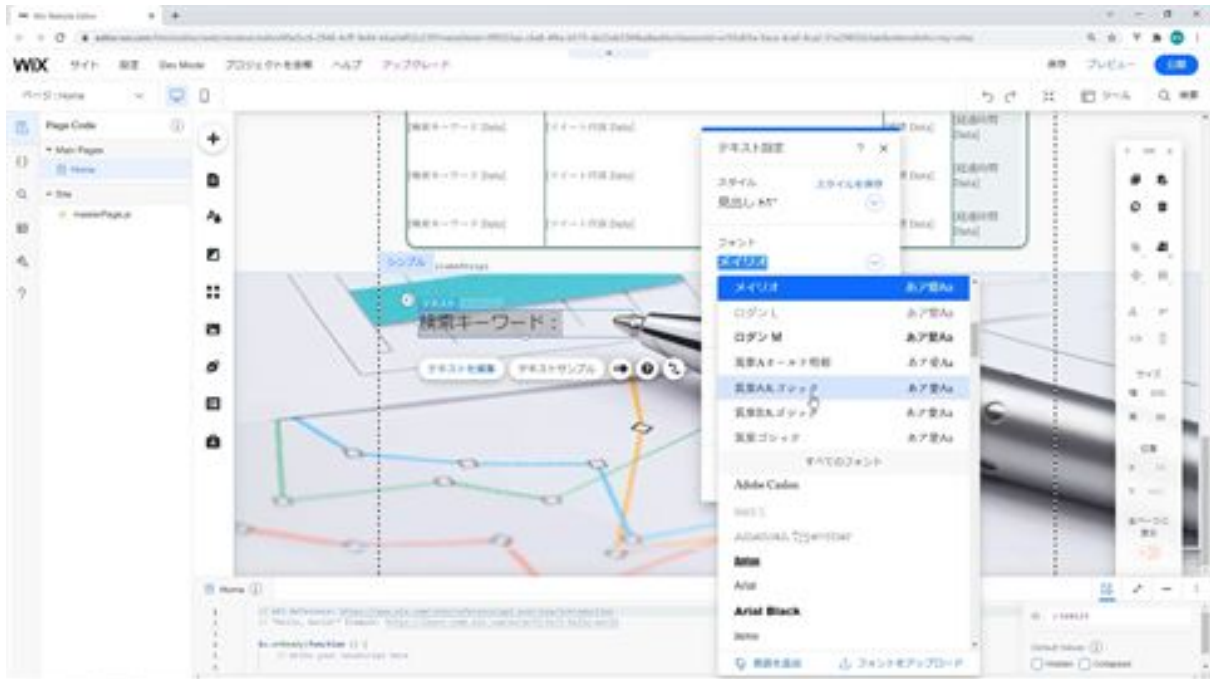
This is the end of creating the strip for the data table.

Let's create a strip for displaying selected data

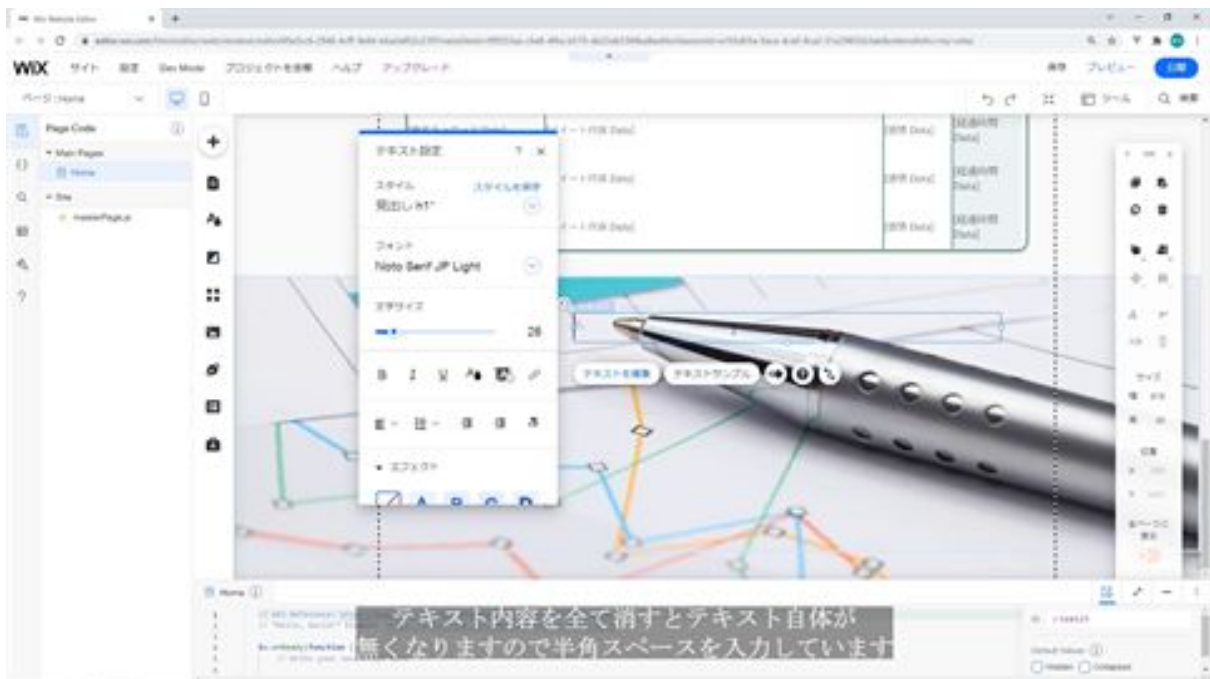
Now we will create a strip for displaying the selected data. First, we will add a new strip. Next, we will change the background of the strip. Go to Settings and change the Opacity to 0%.



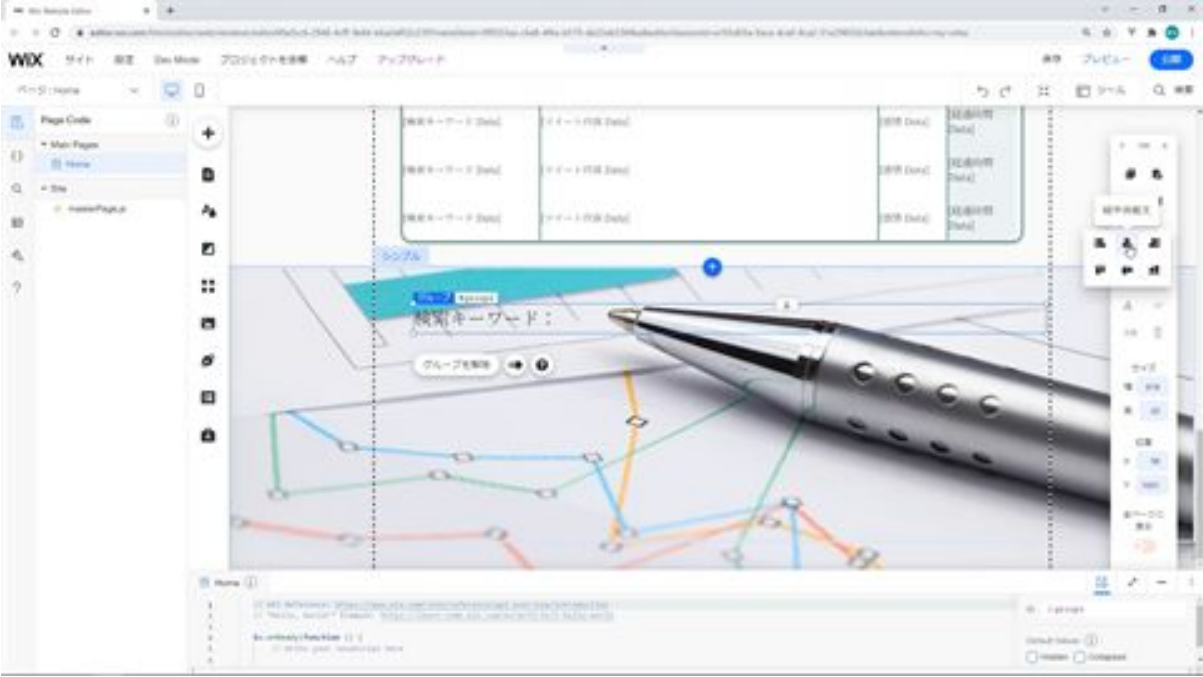
Next, add the text "search terms" and the text to display the search terms on top of this strip. Set the font size of the text to 26px and the font to match the font of the rest of the text.



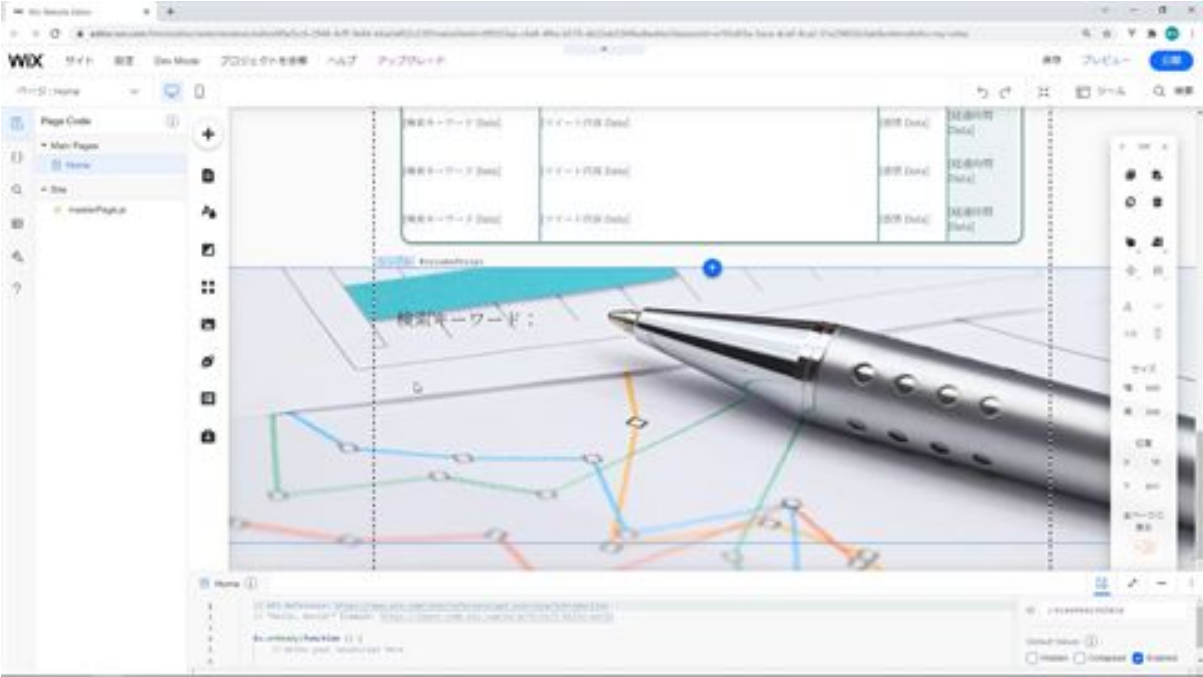
Then copy the added text and use it as the text for displaying the search keywords. It will be moved to the center of the screen. If you enter a single space, the text will disappear, but you can use Wix's back button to display it.



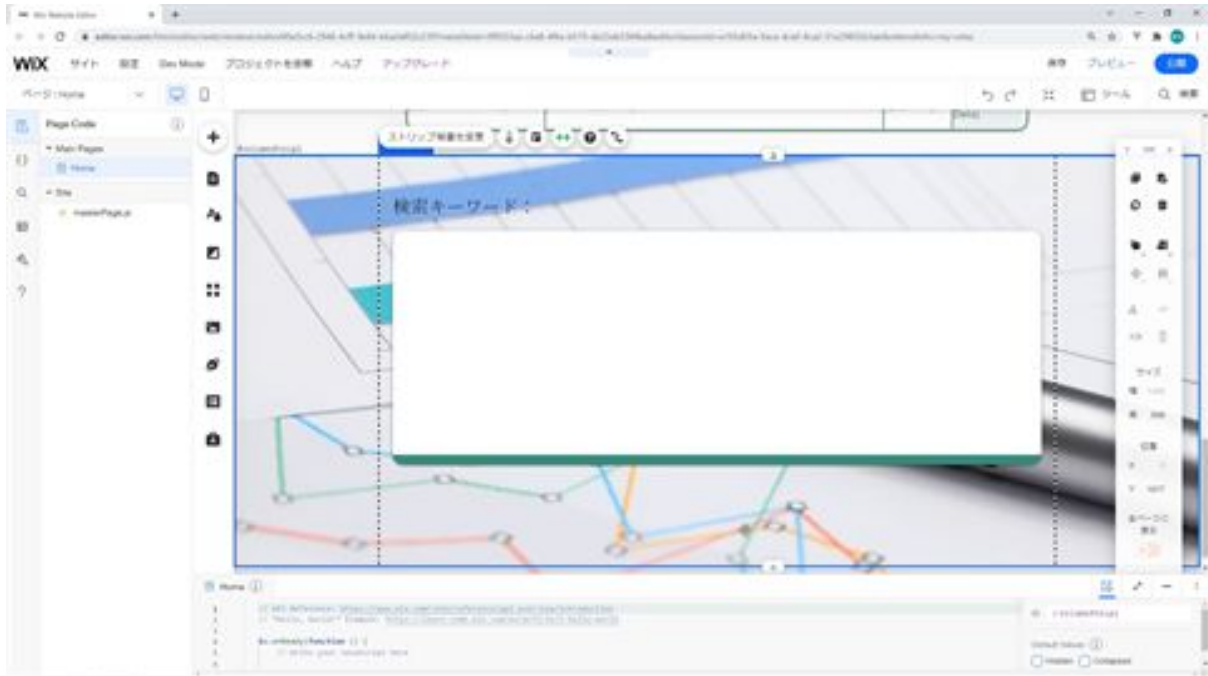
Change the width of the text. Then group these two texts.



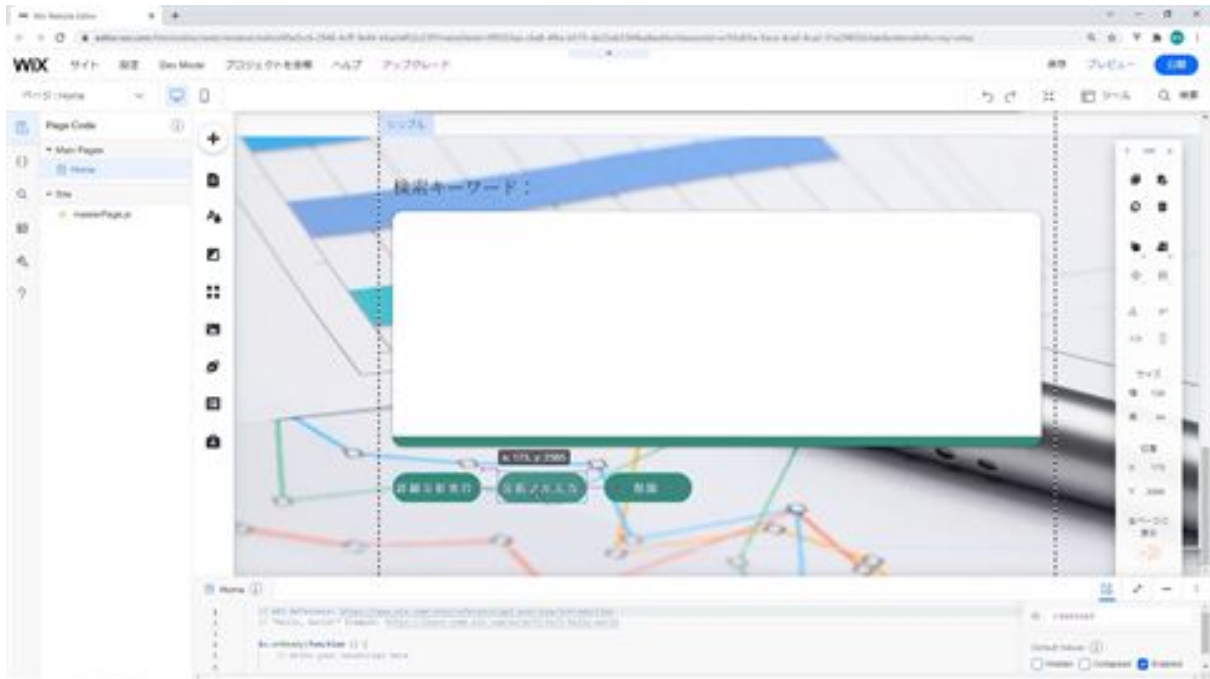
Next, add a text box below the added text. Copy the text box that has already been added and paste it into the strip on this side.



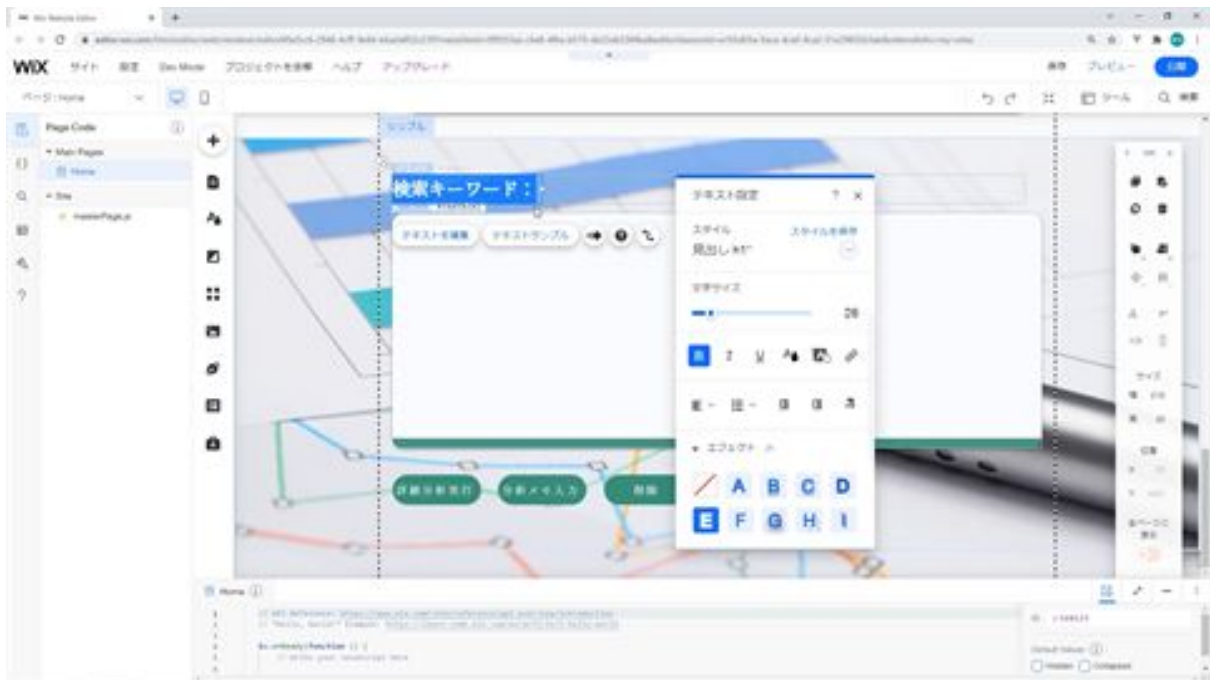
Move the added text to the left to match the text box.



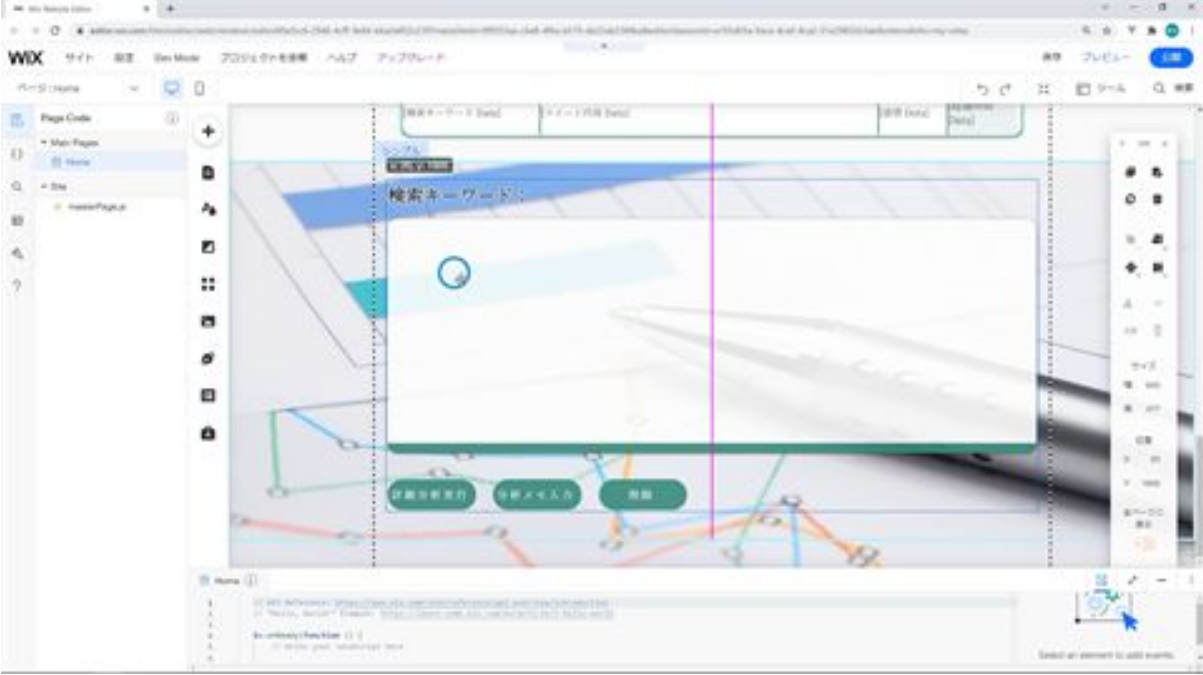
The next step is to increase the height of the strip and add buttons. This time we will add a button called "Detailed Analysis", a button called "Enter Analysis Notes", and a button called "Delete". Copy the buttons you have already created and paste them into this strip. Change the width of the Execute Detailed Analysis button to 128px. Duplicate the button you have created. Change the duplicated button to Analysis Memo Entry and Delete.



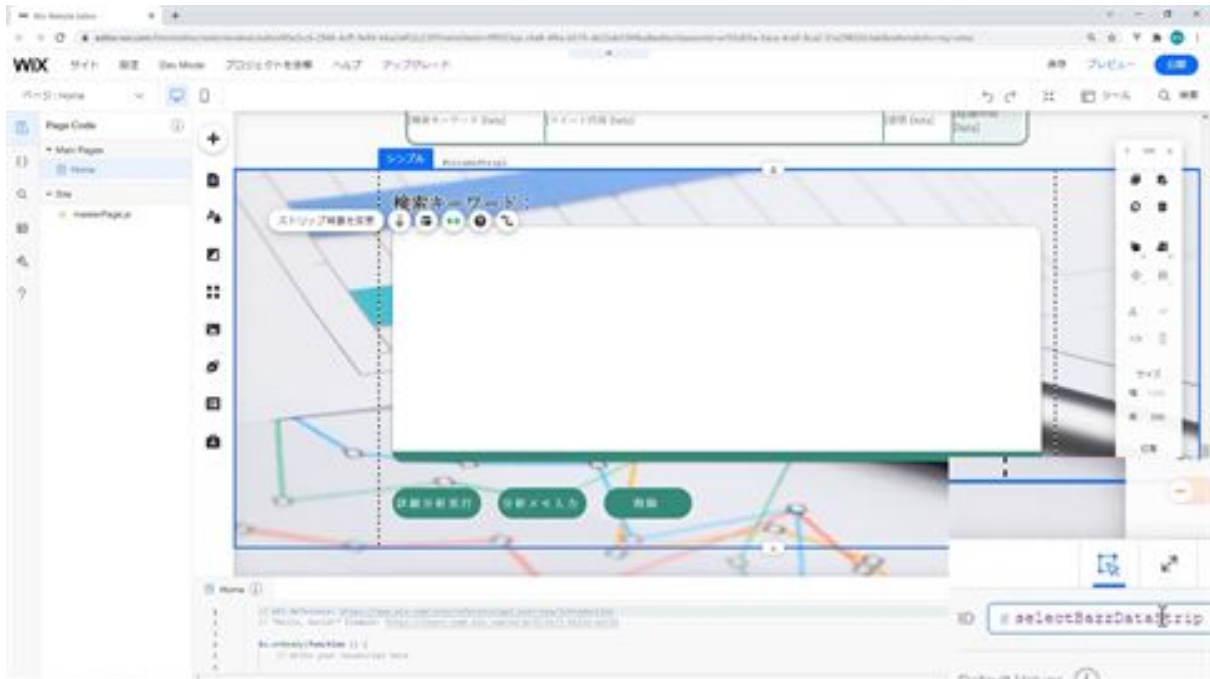
The text that displays the search terms and the text that displays the search terms is hard to read, so we will change it a little. Bold the text and add effects.



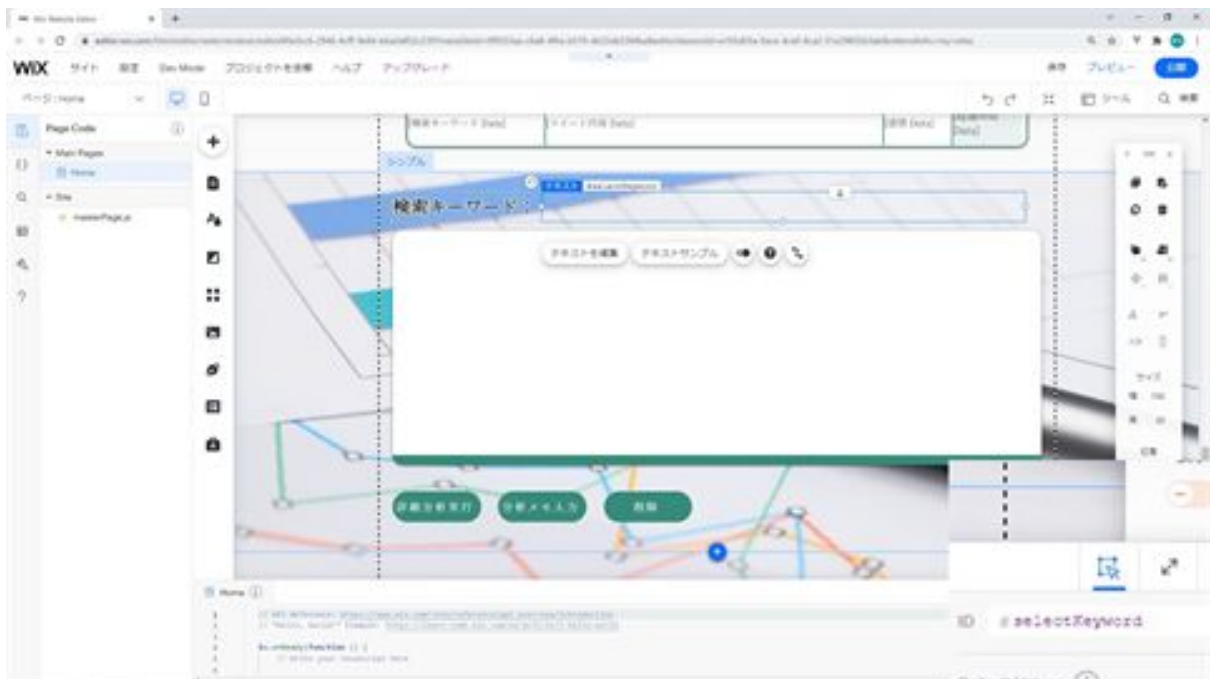
Next, we will change the height of the strip. We'll change the height to 550px. Select all of the parts that are at the bottom of the strip and move them to the top.



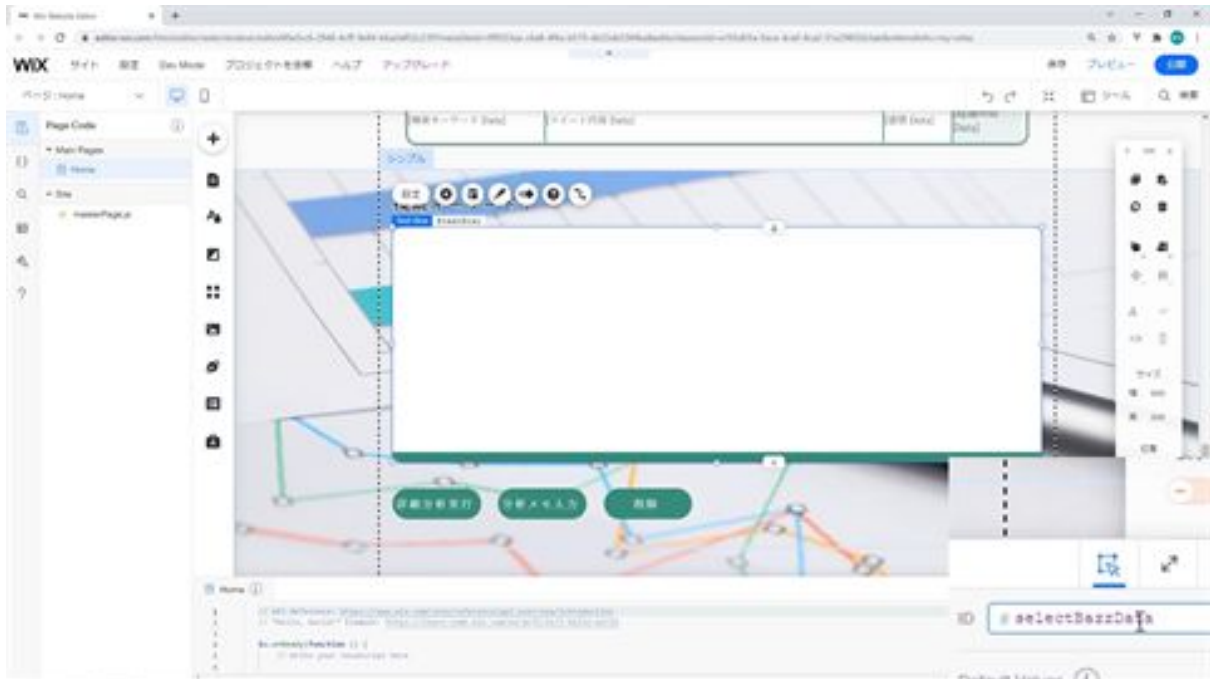
The next step is to change the ID names of the strips and parts. For this strip, we will change the ID to selectBazzDataStrip.



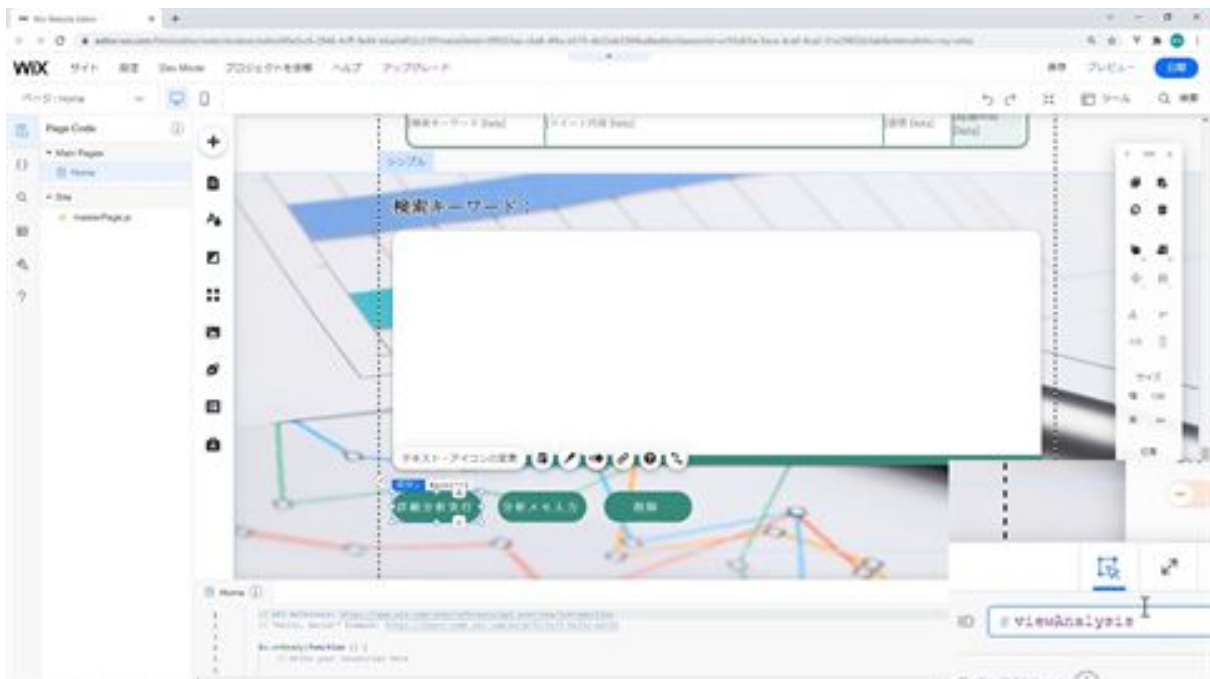
Change the ID of the text that displays the next search keyword to selectKeyword.



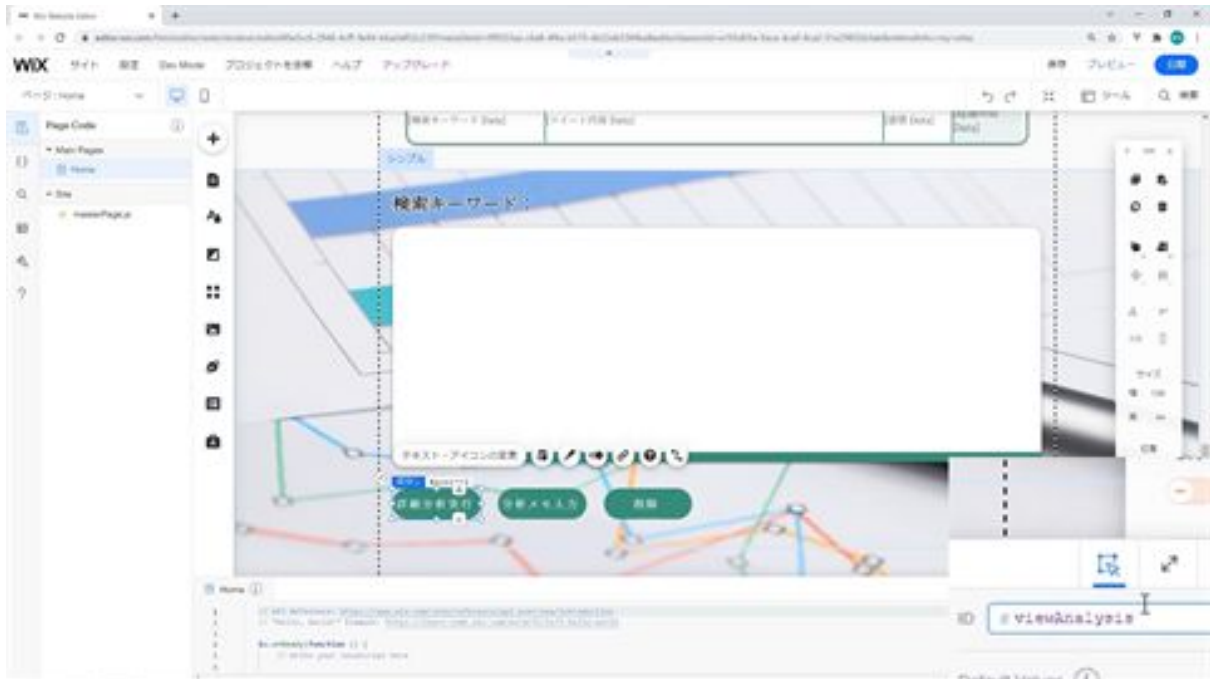
Next, change the ID of the text box to selectBazzData .



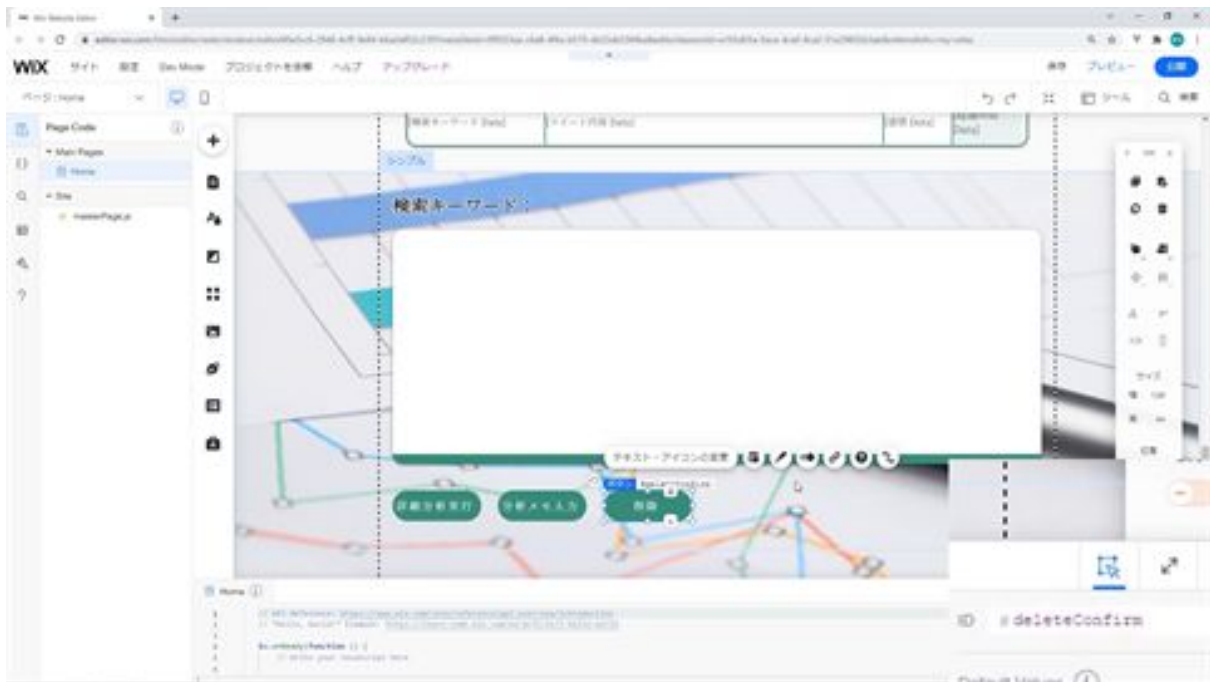
Next, we will change the ID of this Run Detailed Analysis button to viewAnalysis.



Next, change the ID of the analysis memo input button to viewMemoInput.



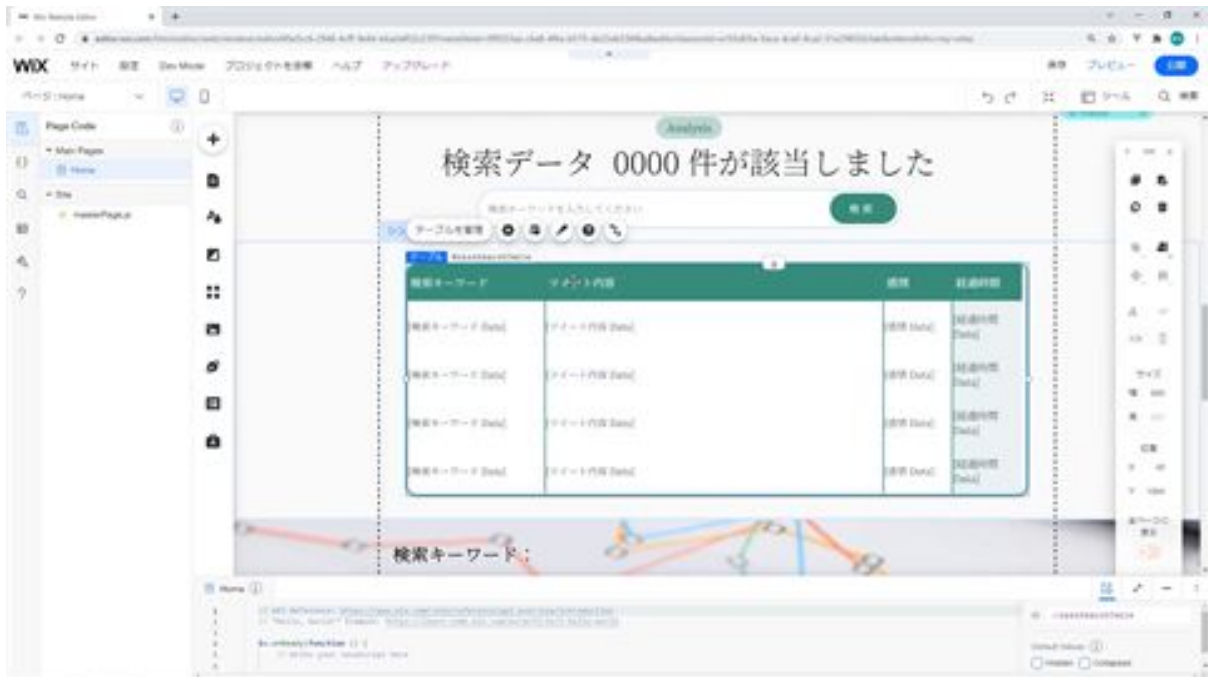
Then modify the ID of the delete button as delete confirm.



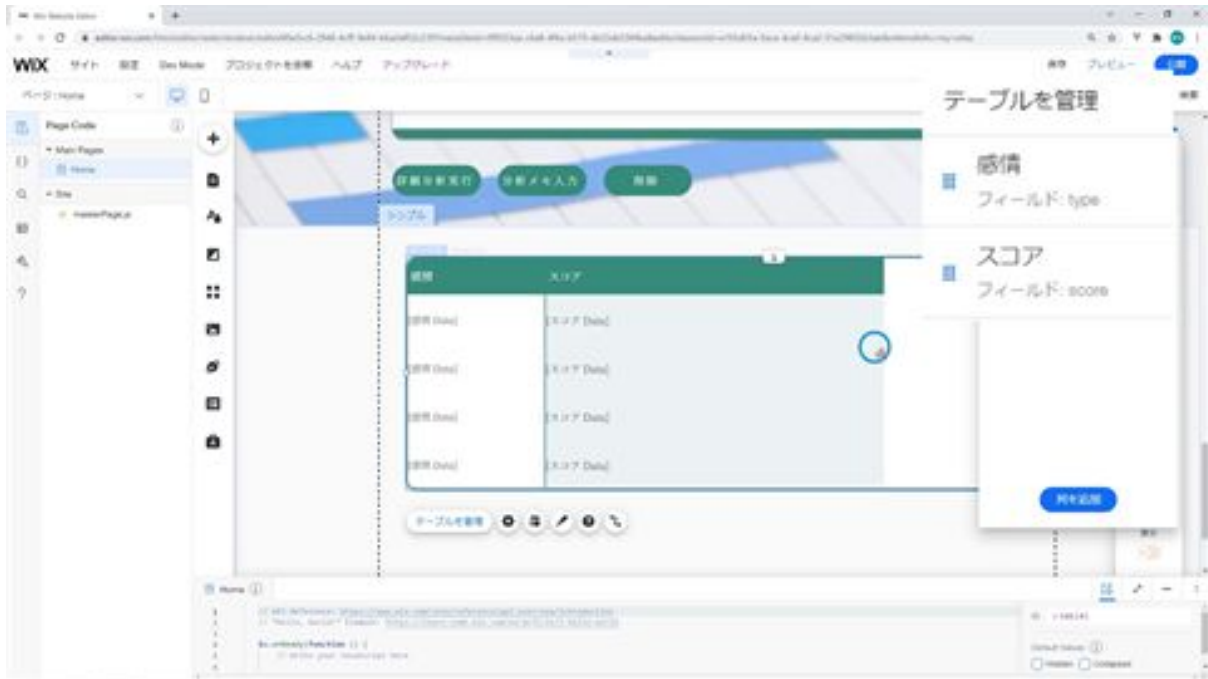
This completes the creation of the strip for displaying the selected data.

Let's create a strip for detailed analysis

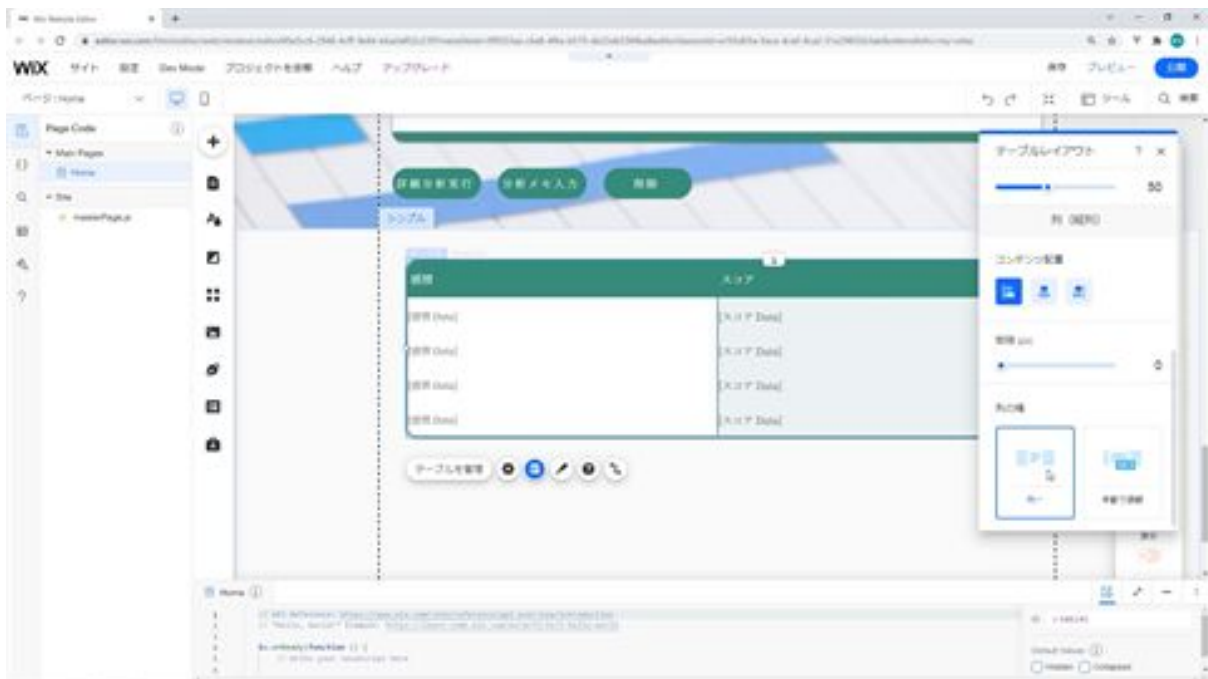
Next, we will create a strip for detailed analysis. First, we will add a new strip. Change the height of this strip to 650px. Next, we will add two tables here. We have already created one table here, so we will copy this table and modify it.



First of all, double click on Manage Tables and change the number of items to two, delete the two items and change the search term to emotion. And change the tweet content field to score. And change the emotion field to type. Then change the score field to score.

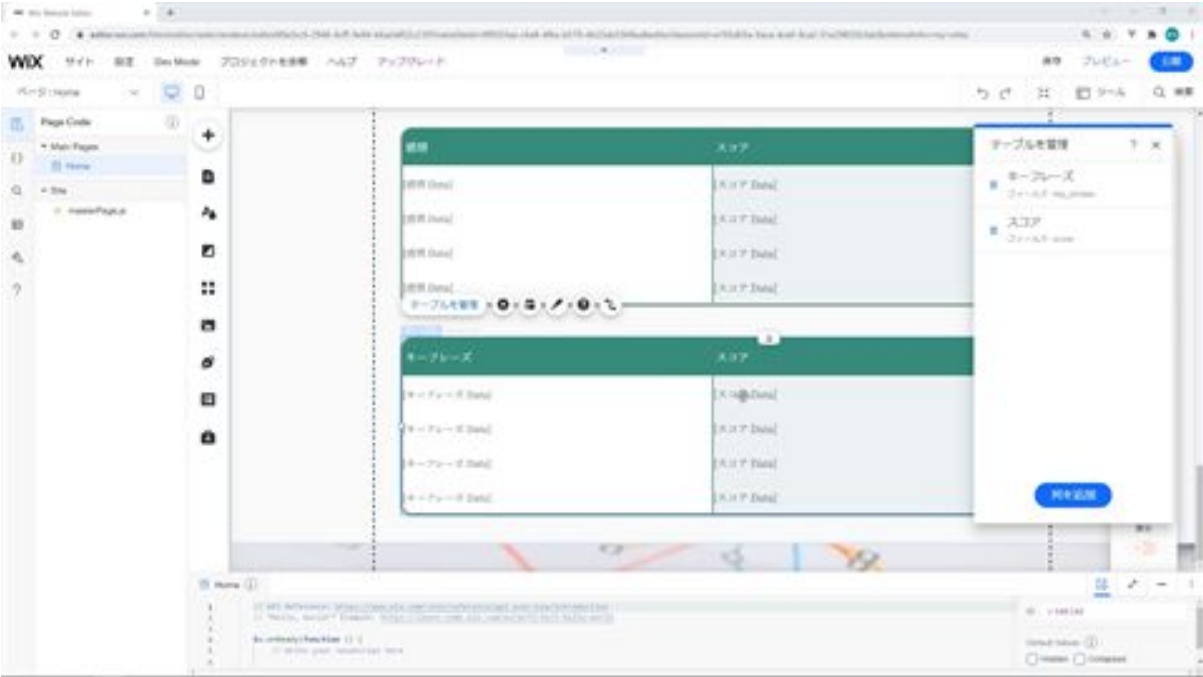


Next, we will change the layout of the table. Change the height of the table to 50px and make the width of the columns uniform.

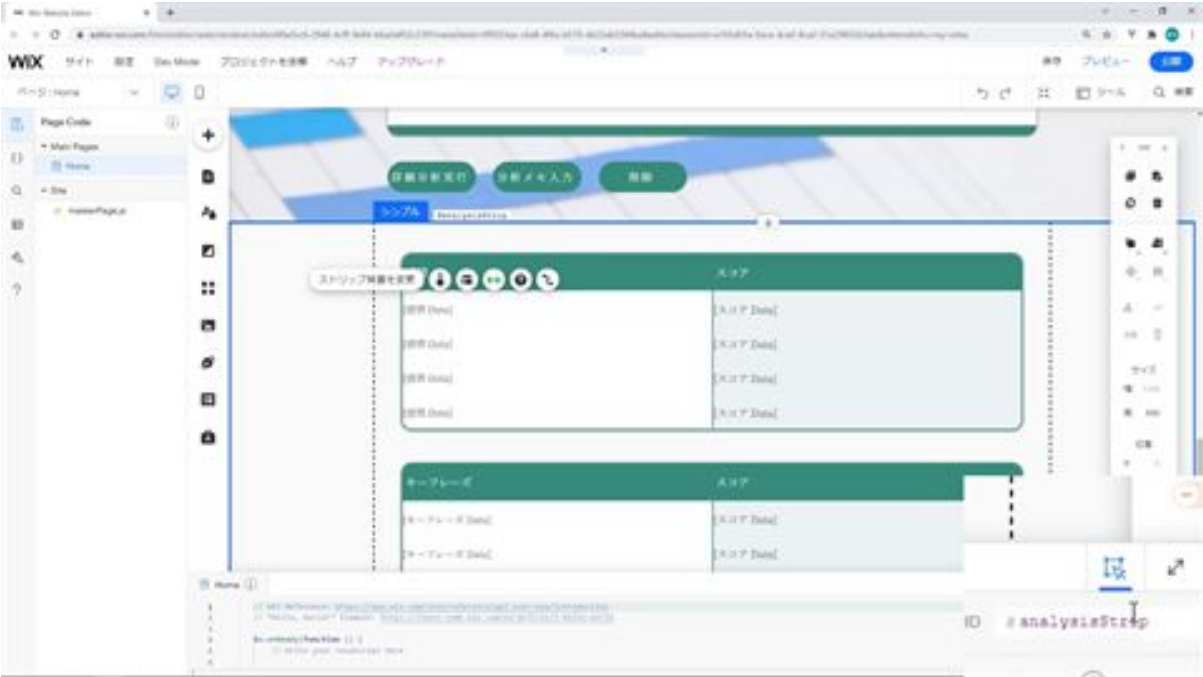


Once you have done this, copy this table and create another one below it. Go to Manage Tables in the copied table and change the fields. Change the

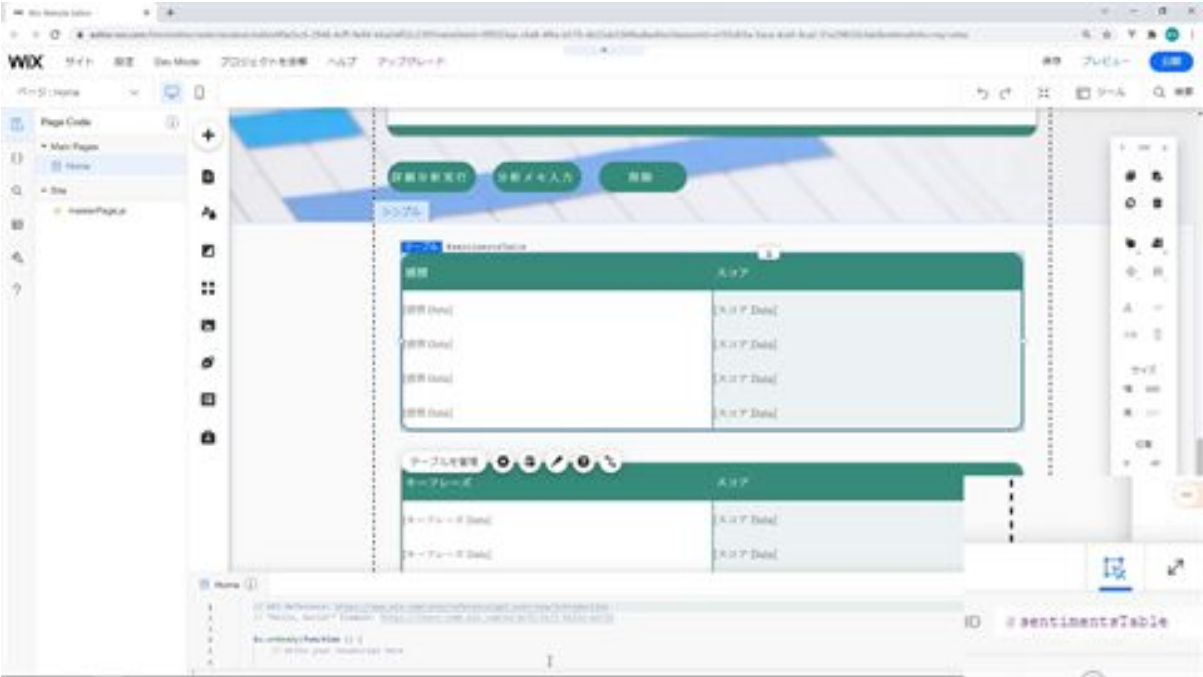
Emotion part to Keyphrase and modify the field name of Keyphrase to key_phrase. You can leave the score part as it is.



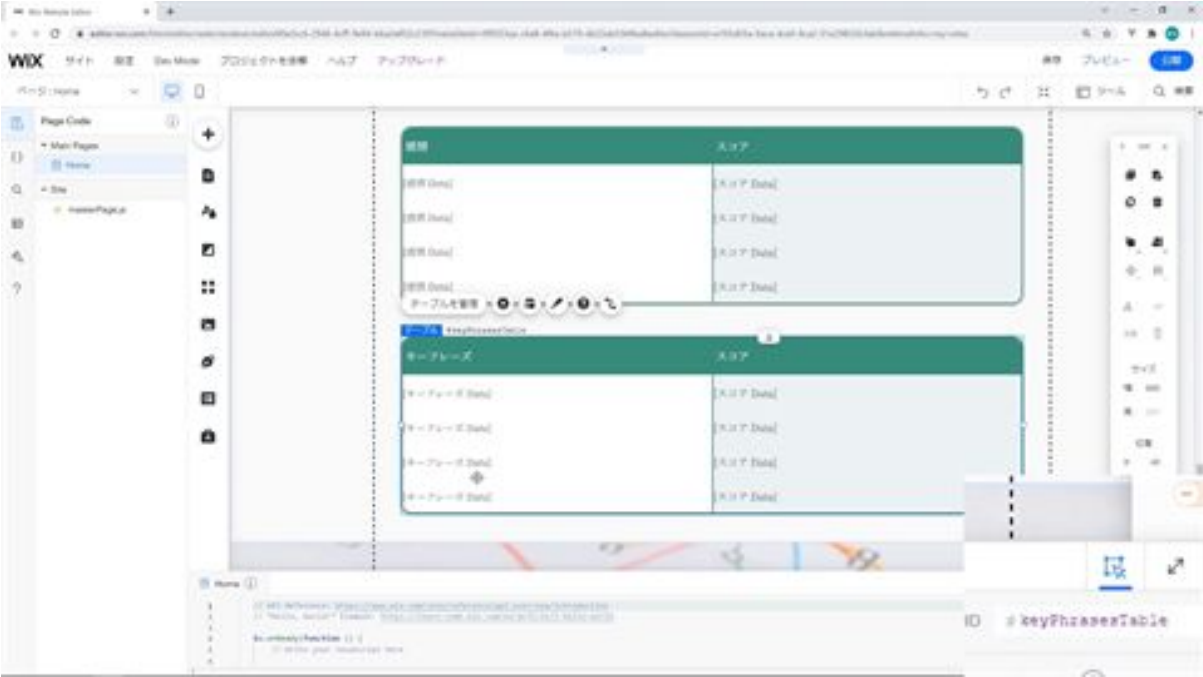
Next, we will modify the ID of this strip and part. Modify the ID of the strip to be analysisStrip.



Next is the ID of this first table, which should be modified as sentimentsTable.



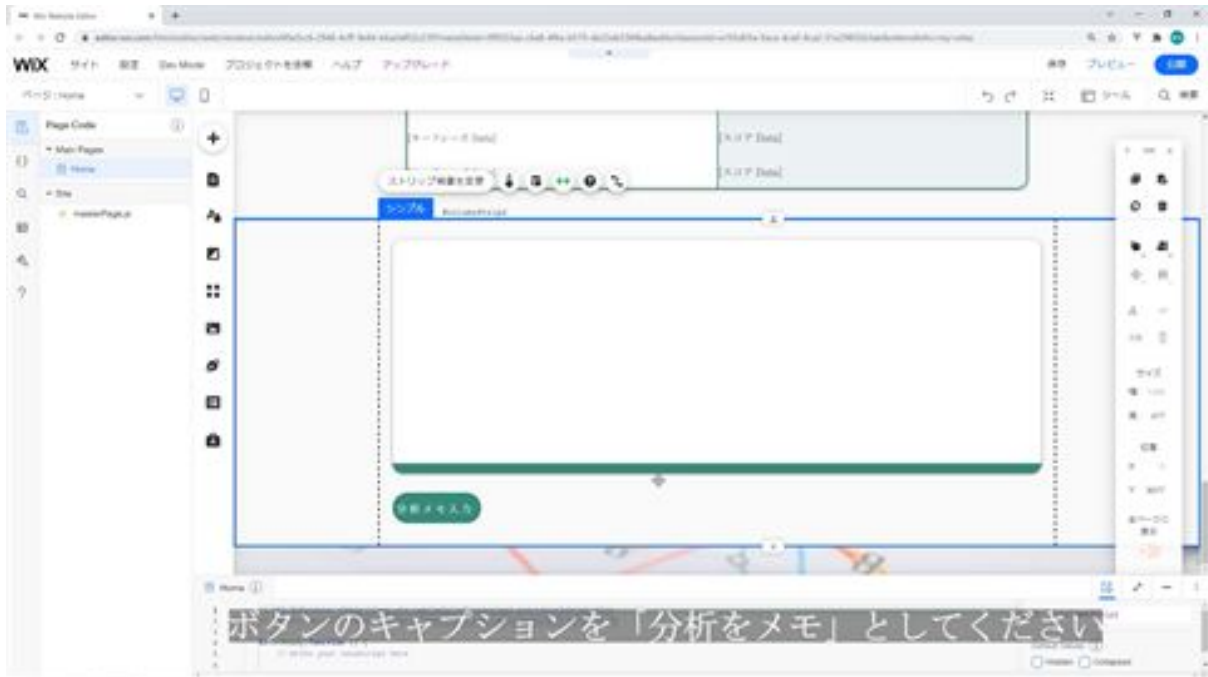
And for the second table, modify the ID as keyPhrasesTable.



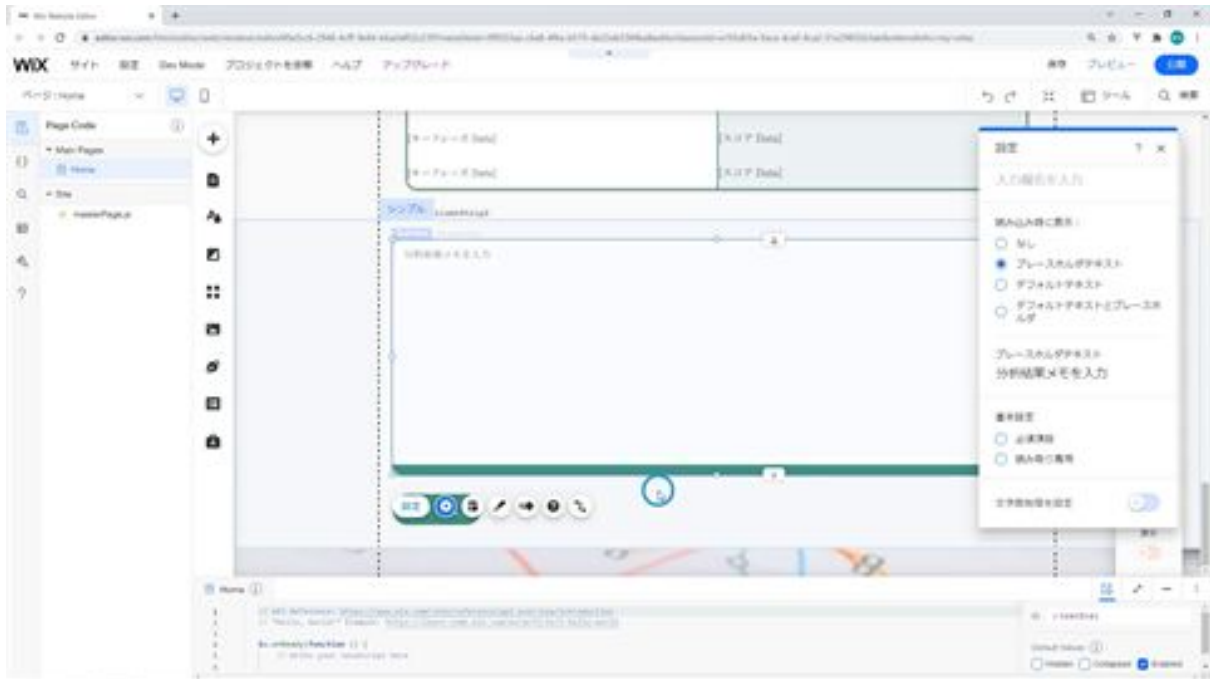
This is the end of creating the strip for detailed analysis.

Let's create a strip for storing notes

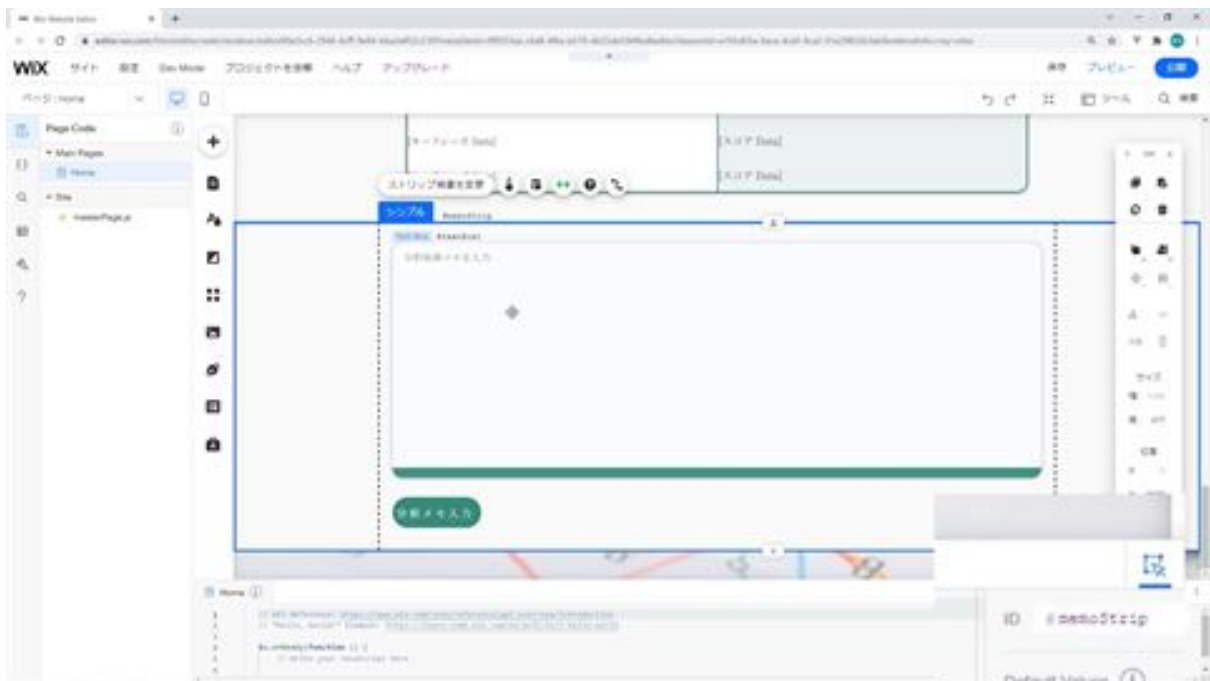
The next step is to create a strip for storing notes. First, we will add a new strip. Next, we will add a text box to this strip. Next, add a button called "Memo Analysis" below this text box.



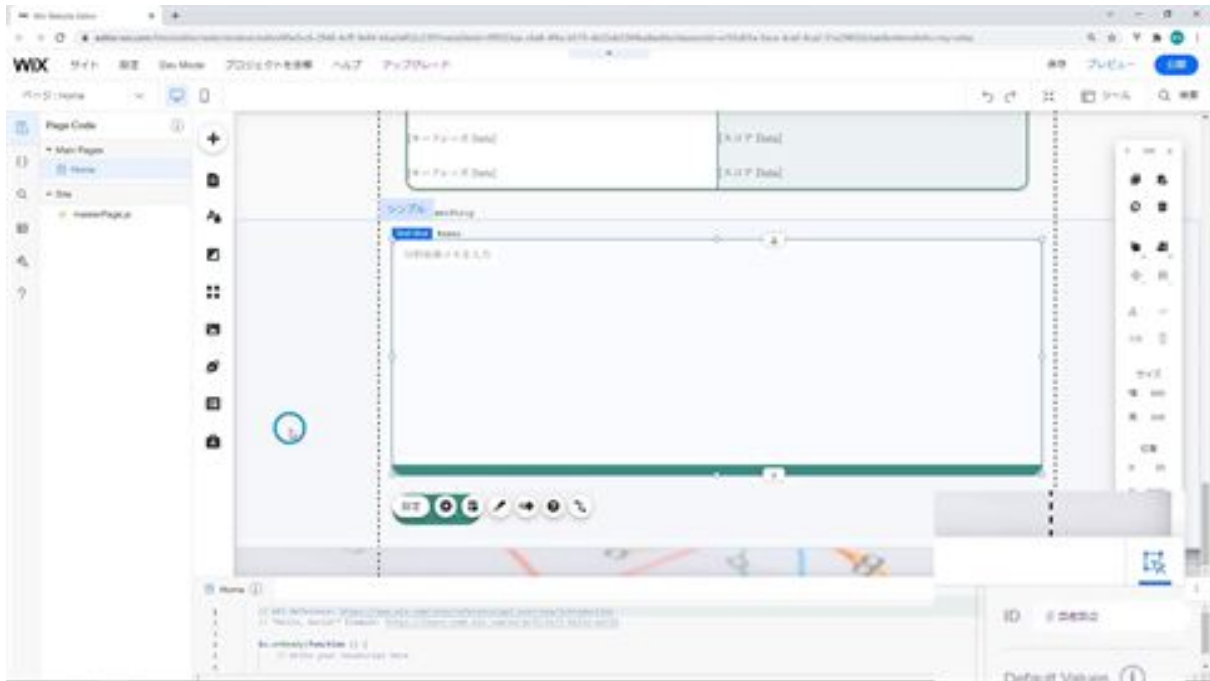
The next step is to change the placeholder in the text box. In the placeholder, type "Enter analysis result memo" and uncheck the read-only option in the Preferences.



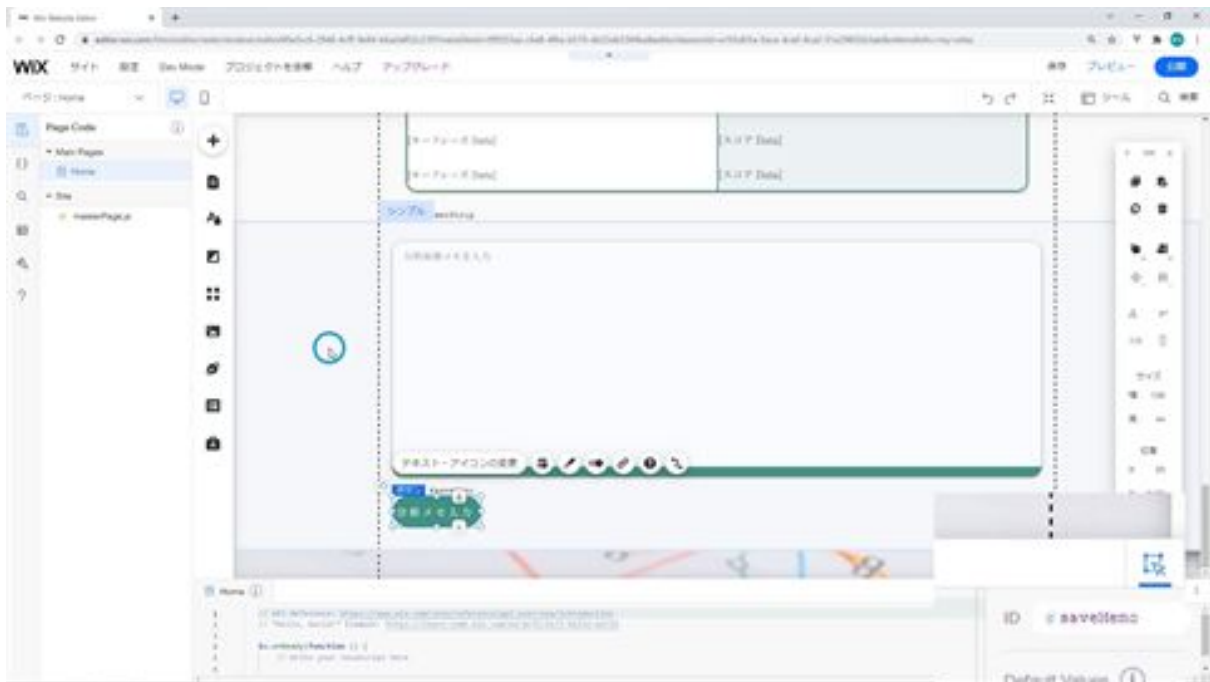
The next step is to change the ID of the strip and the parts inside it. First, we will modify the ID of this strip to memoStrip.



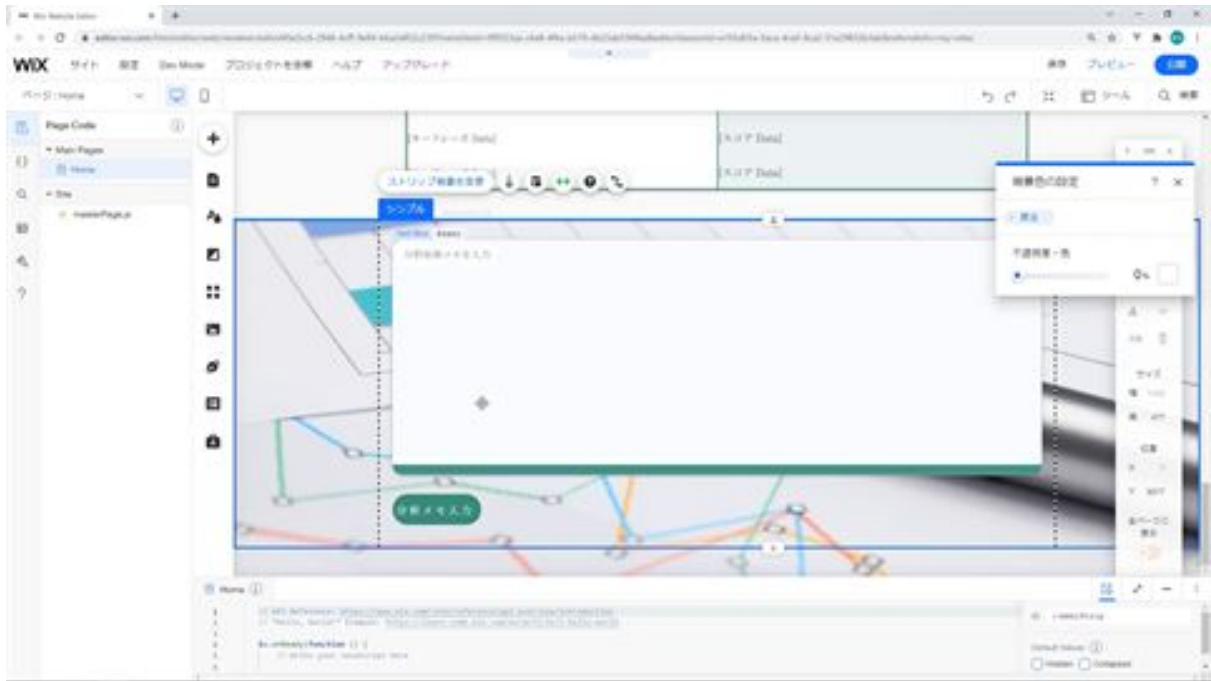
Then modify the ID of the text box as the memo.



Next, modify the ID of this analysis memo input button as saveMemo.

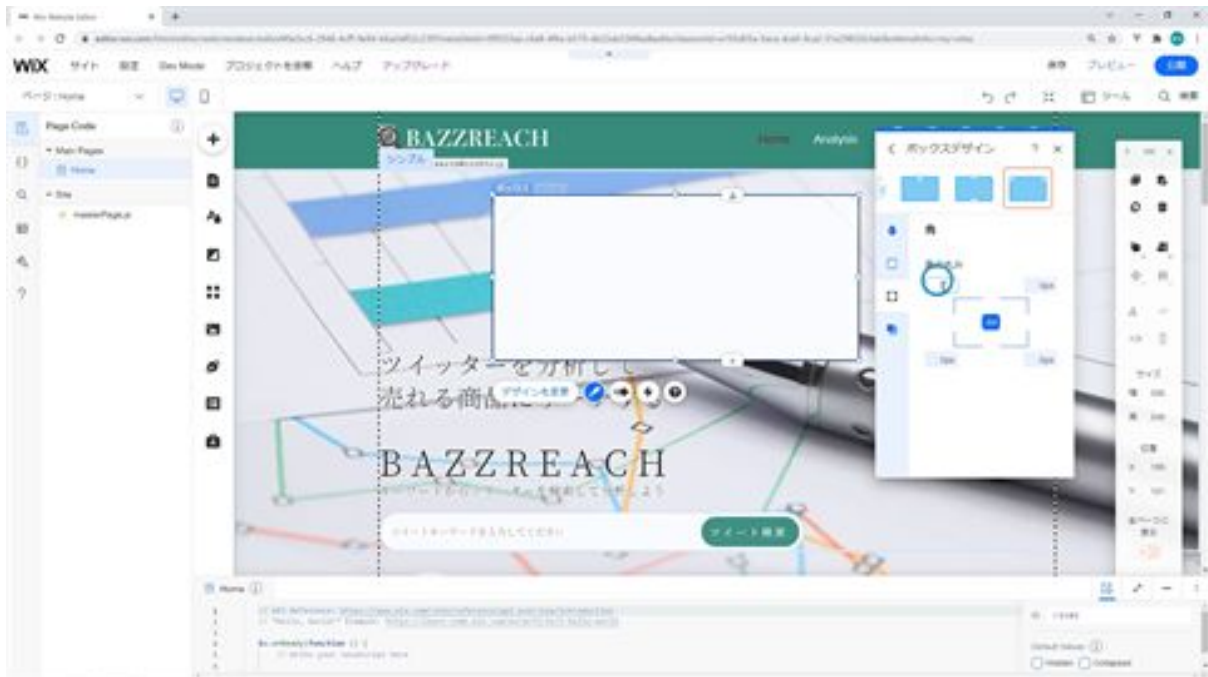


Next, change the background of this strip to transparent. That's all there is to create a strip for saving notes.

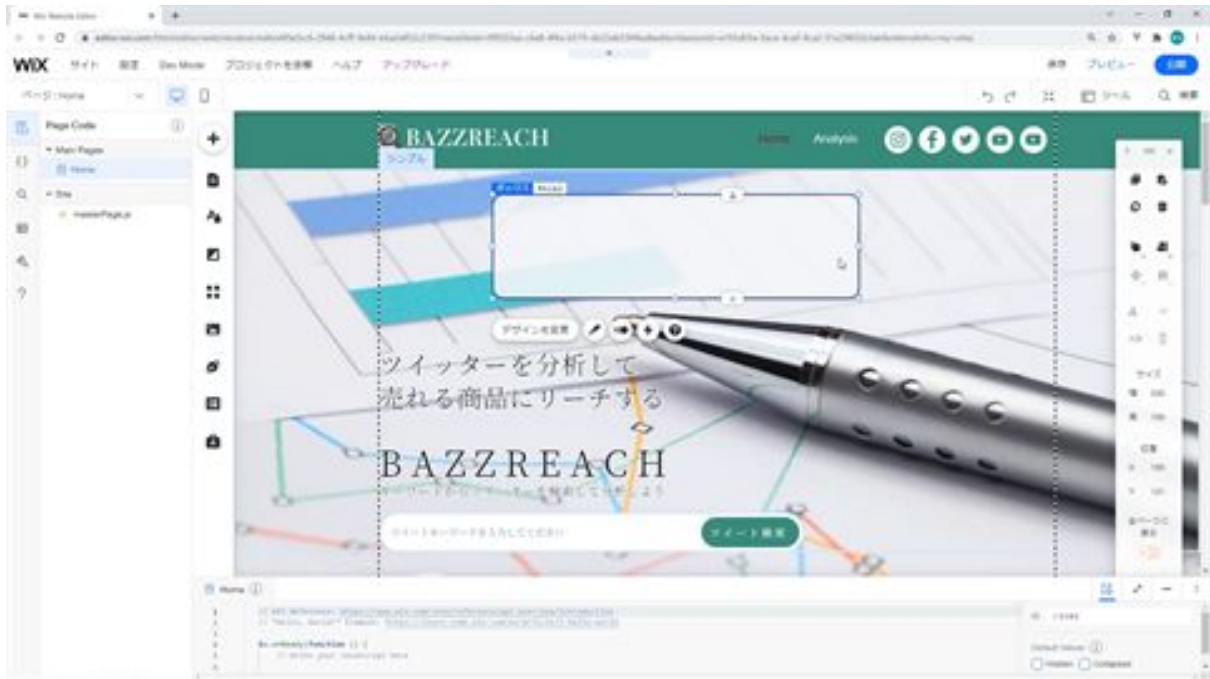


Let's create a confirmation screen for deletion

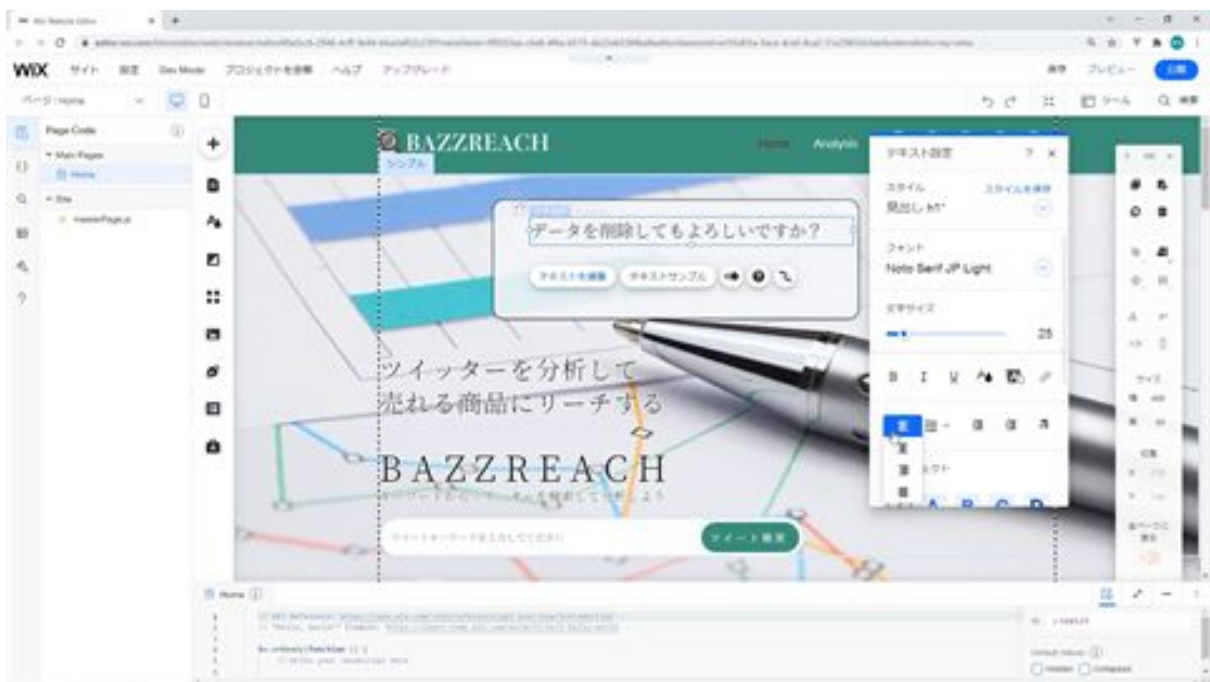
Next, we will create a screen to confirm the deletion. First, we will add a box to the screen. Change the width of the screen to 530px and the height to 150px. Then change the design. Change the box design to a taped design and change the border to 1px.



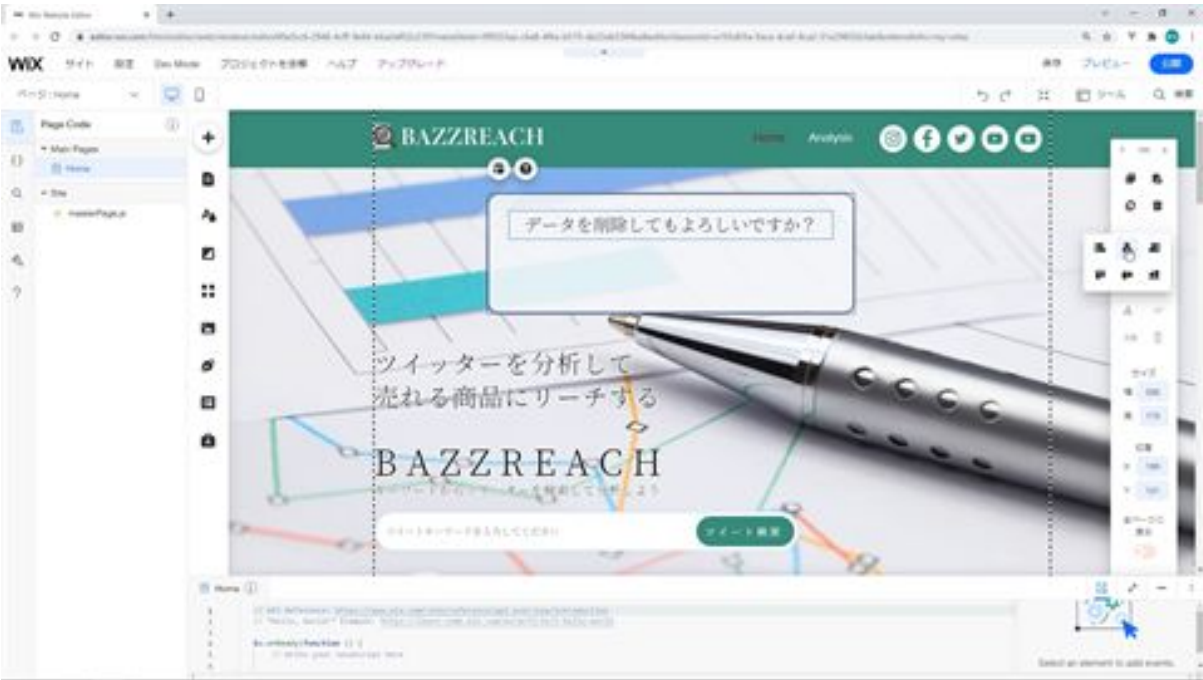
The rounded corners are set to 15px, and the shadow is already added. For the background, set the color to #F5F7FA and the opacity to 90%.



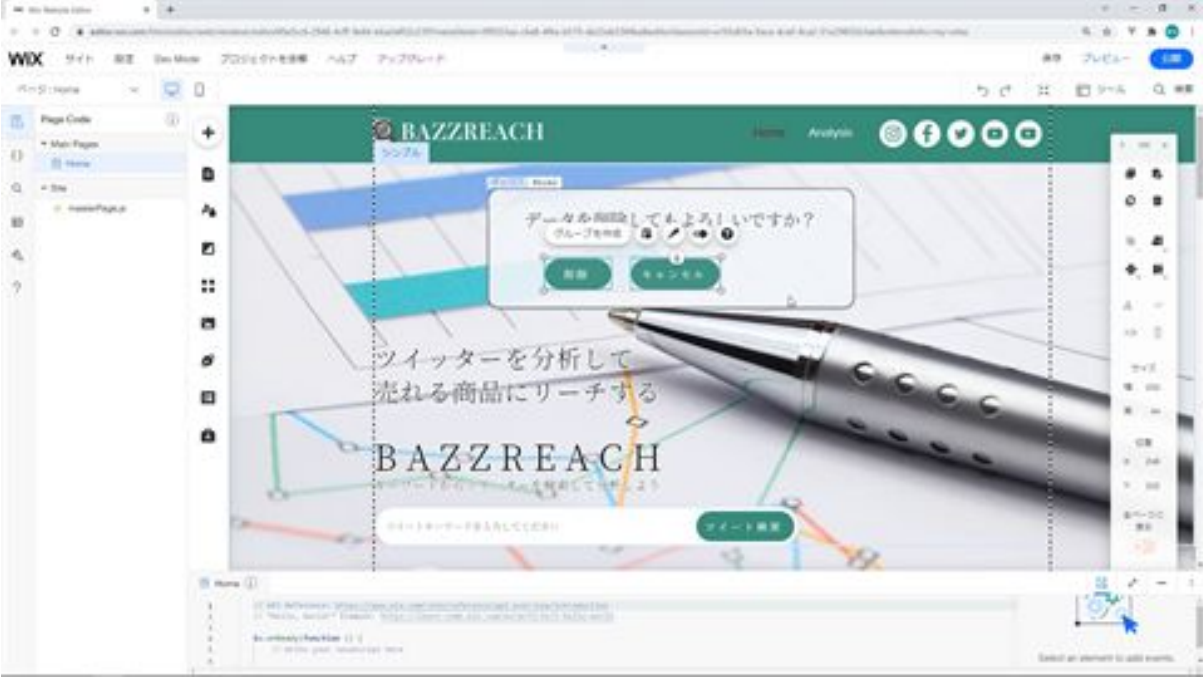
Next, we will add text and a button inside this box. Add the text to the box and change the font size to 25px to match the font of the other text. Change the content of the text to read "Are you sure you want to delete the data?" and change the text to read "Are you sure you want to delete the data? The alignment should be centered.



Select the box and text at the same time and center them with vertical centering.



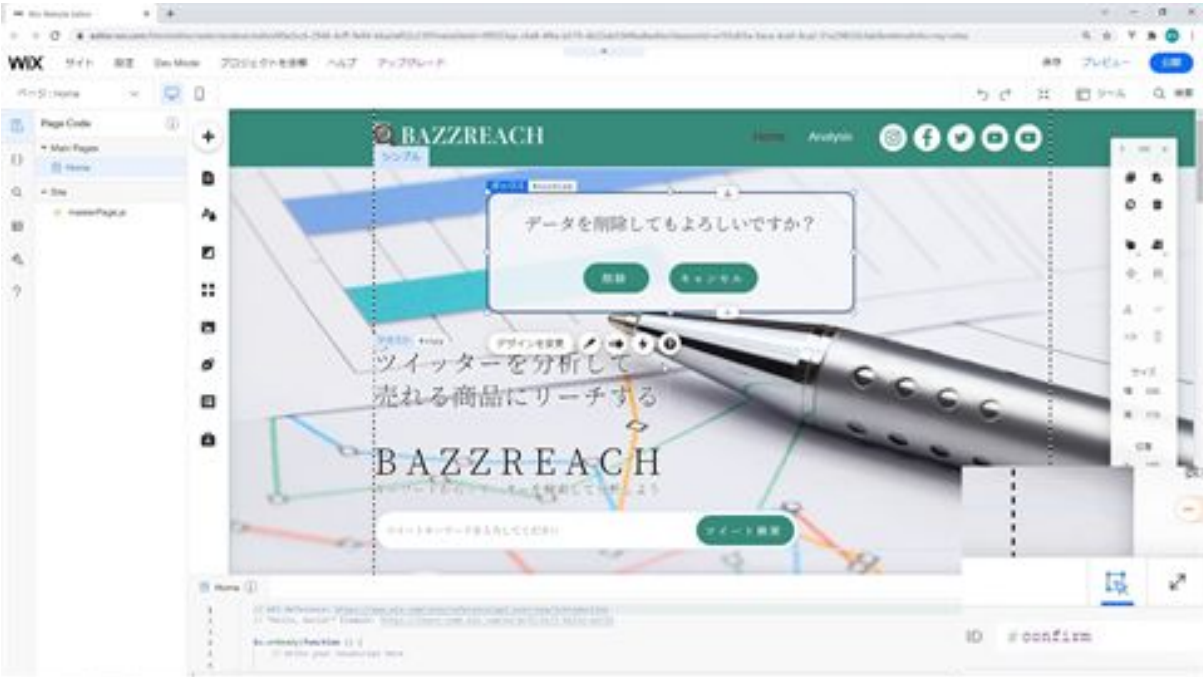
Next, we will add a delete button and a cancel button below this. Copy the delete button below and paste two of them into this box. Make the delete button 96px. Change the text content of the other one to Cancel.



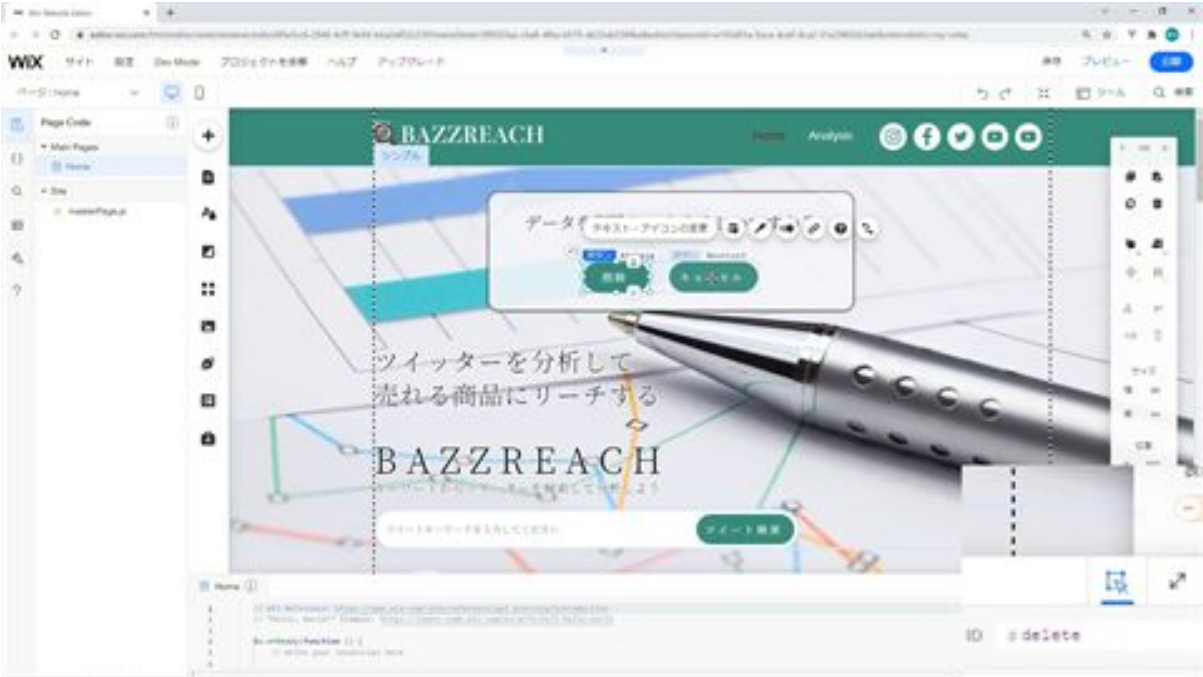
Group these two buttons and move them to the center.



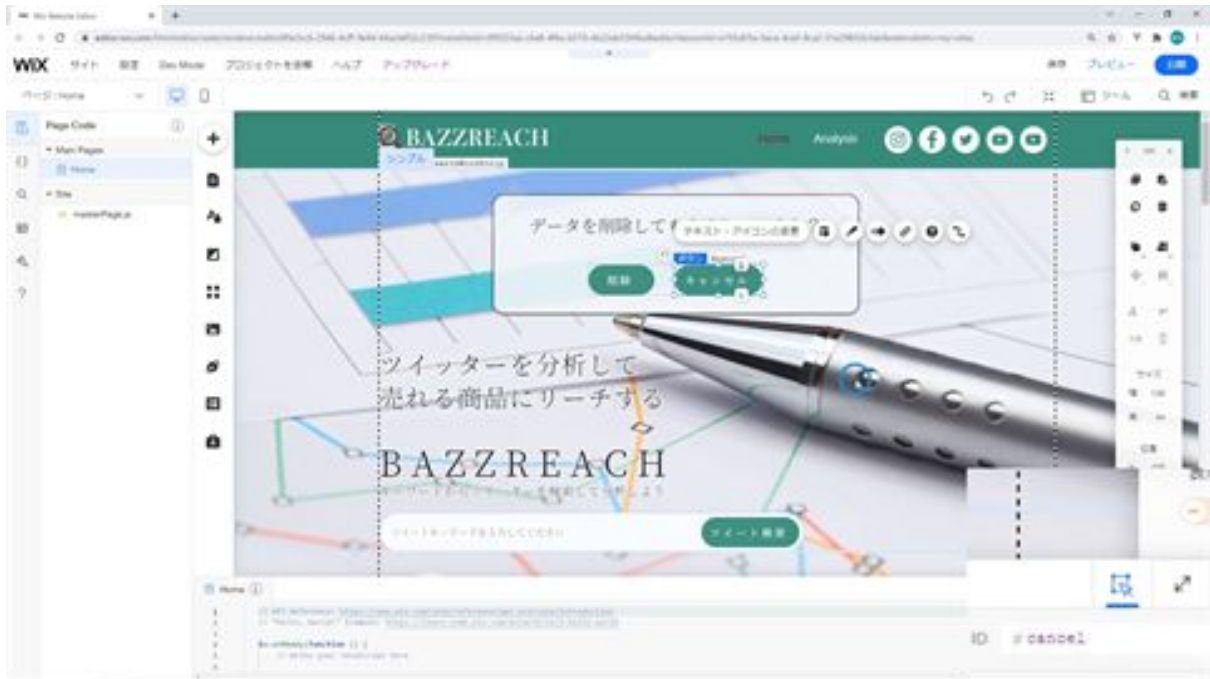
Then change the ID of this box and button. Modify the ID of this box as confirmed.



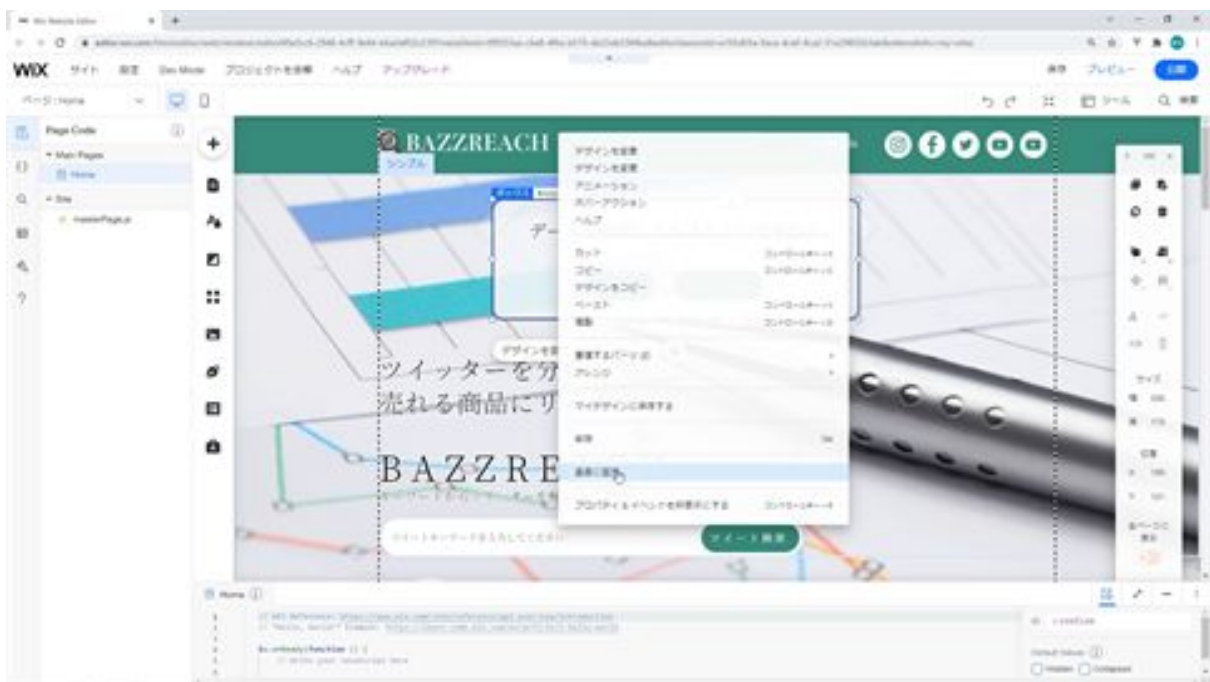
Next, change the ID of this delete button to delete.

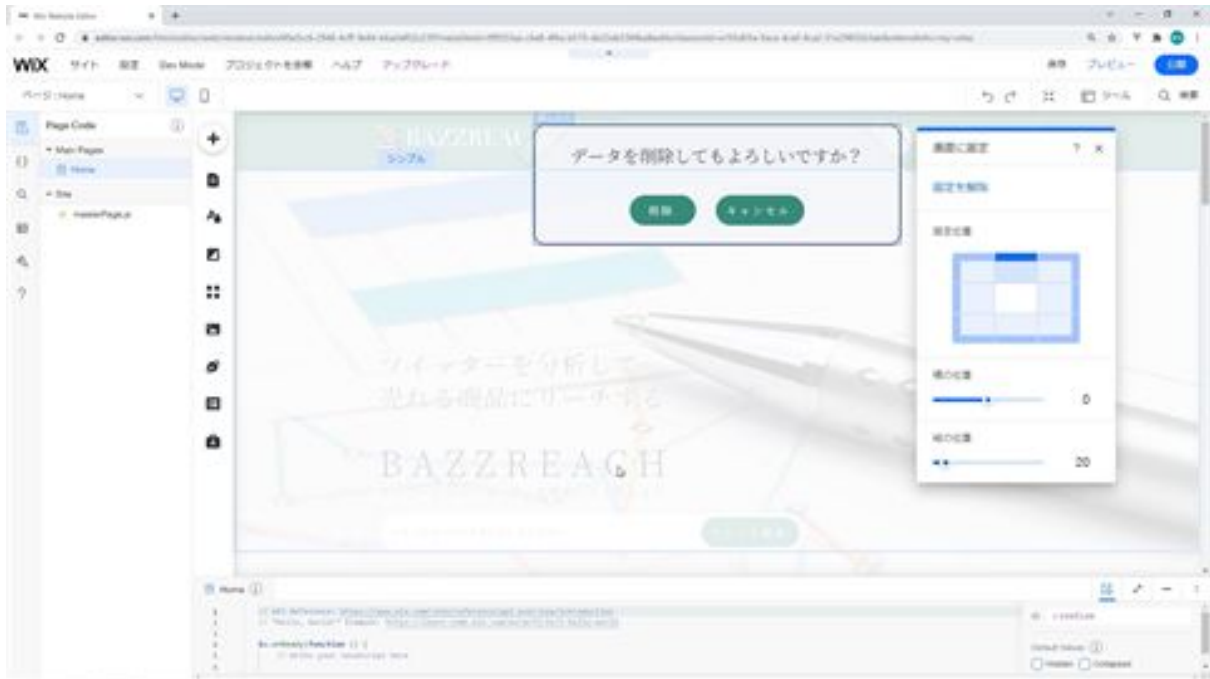


The Cancel button changes the ID to cancel.



Next, fix this box on the screen.

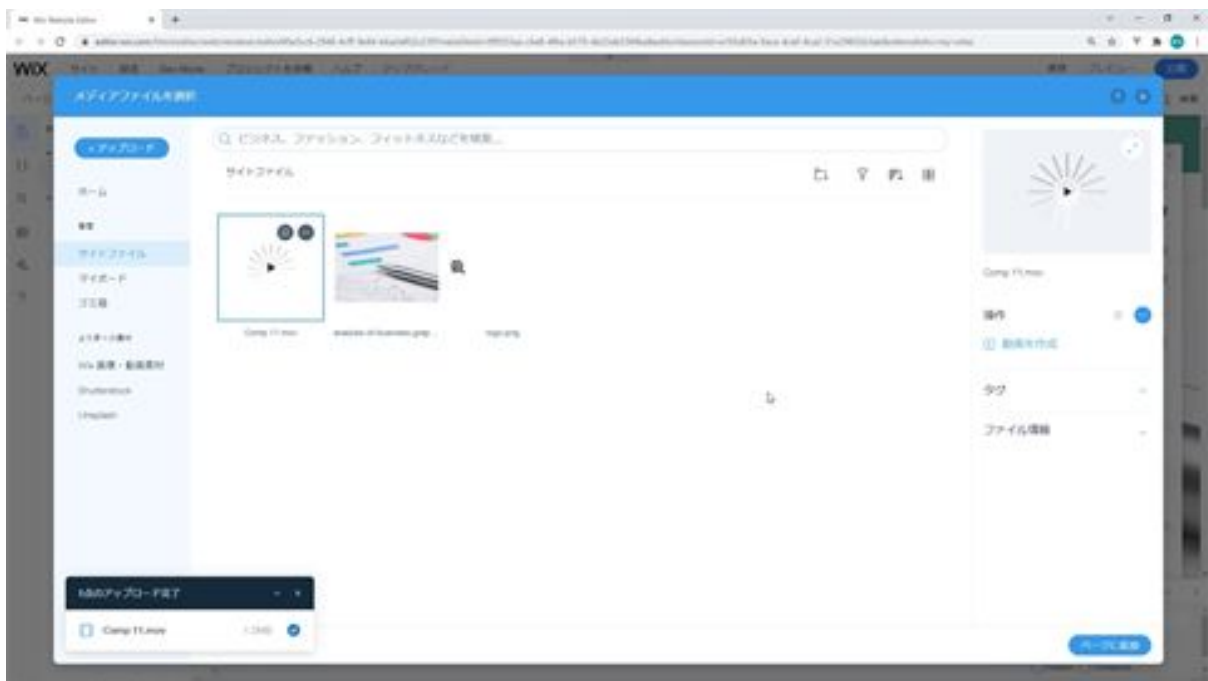




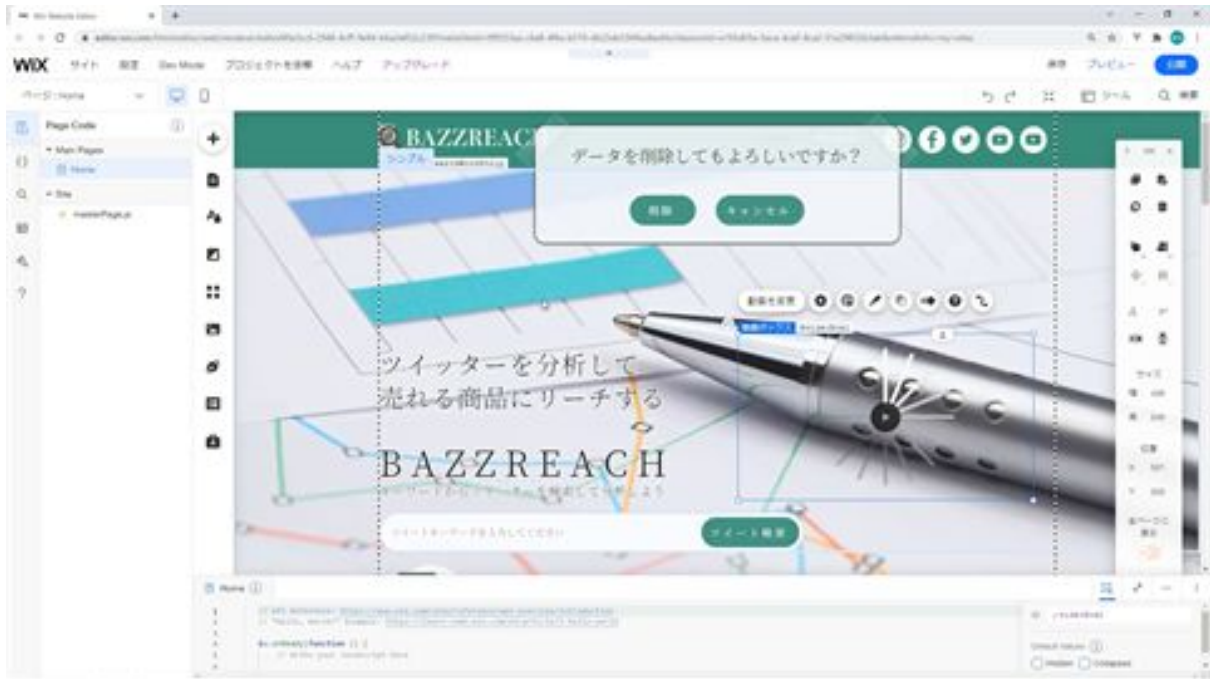
This is the end of creating the confirmation screen for deletion.

Let's create a preloader

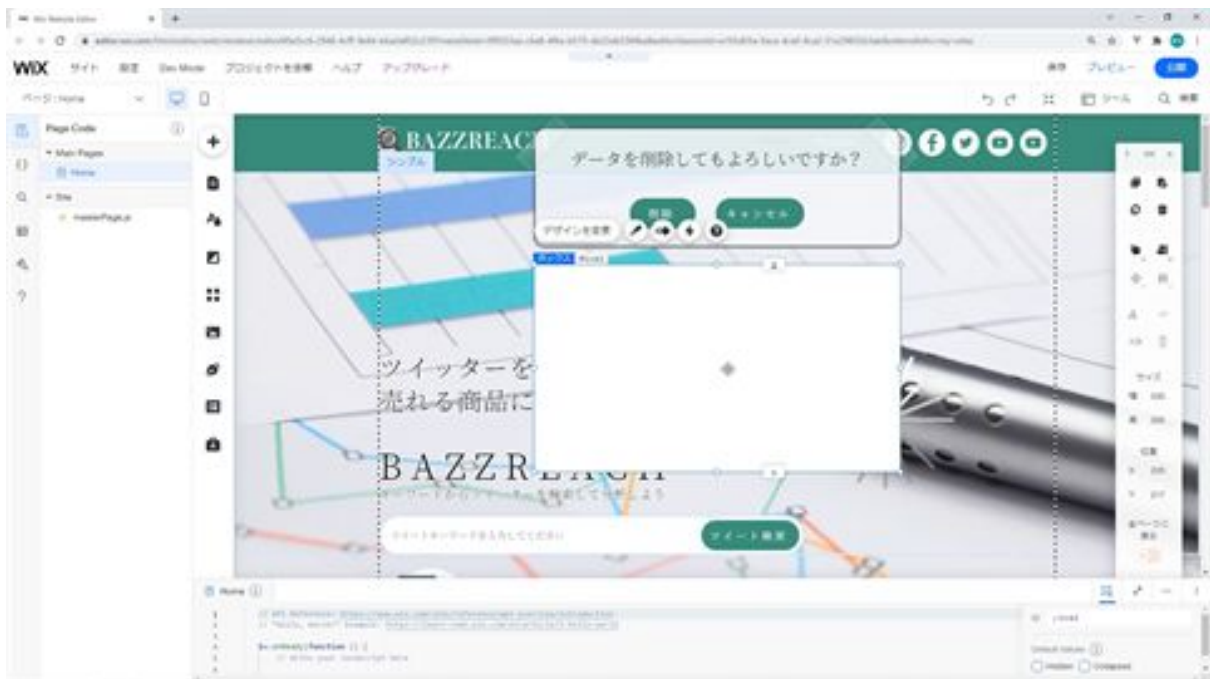
The next step is to create the preloader. The first step is to upload the videos you need to the preloader. Select the media and click Upload Media. Upload the videos you want to display in the preloader here. Once the upload is complete, select it and add it to the page.



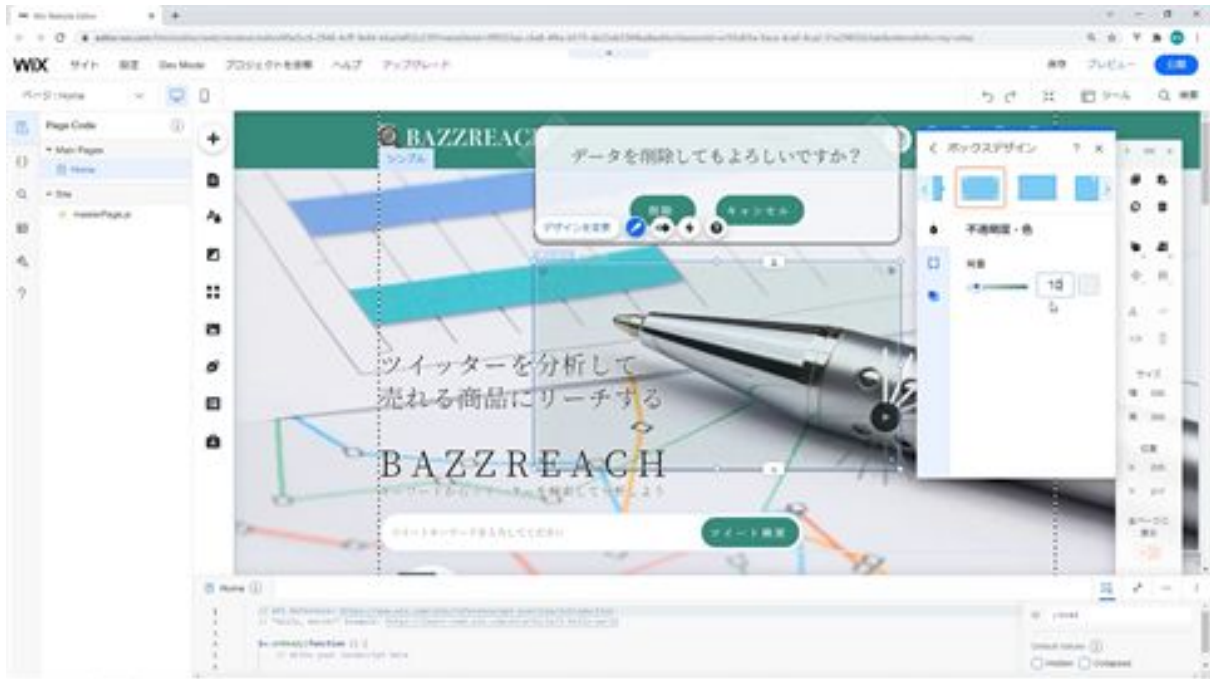
After adding it, reduce the size.



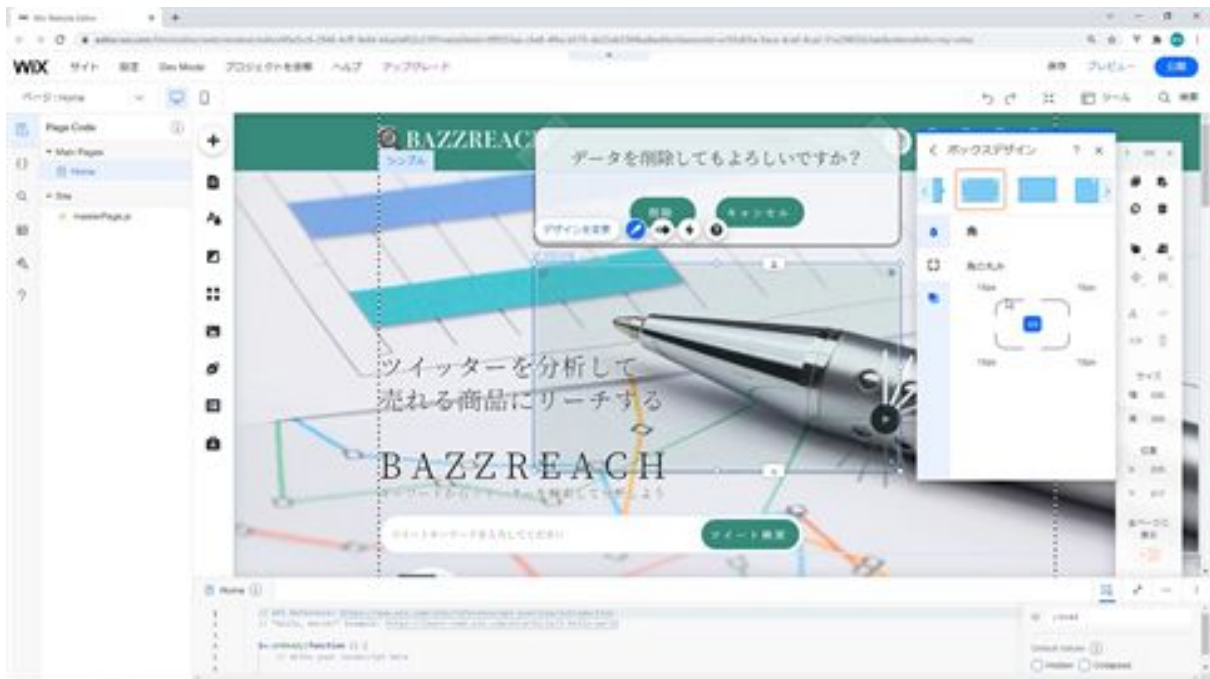
Next, we will create the preloader screen. Add a box from Add. Change the size of the preloader screen to 530px wide and 300px high.



Next, change the design of the preloader screen. Next, change the color of the box to #388C7C and change the opacity to 10%.

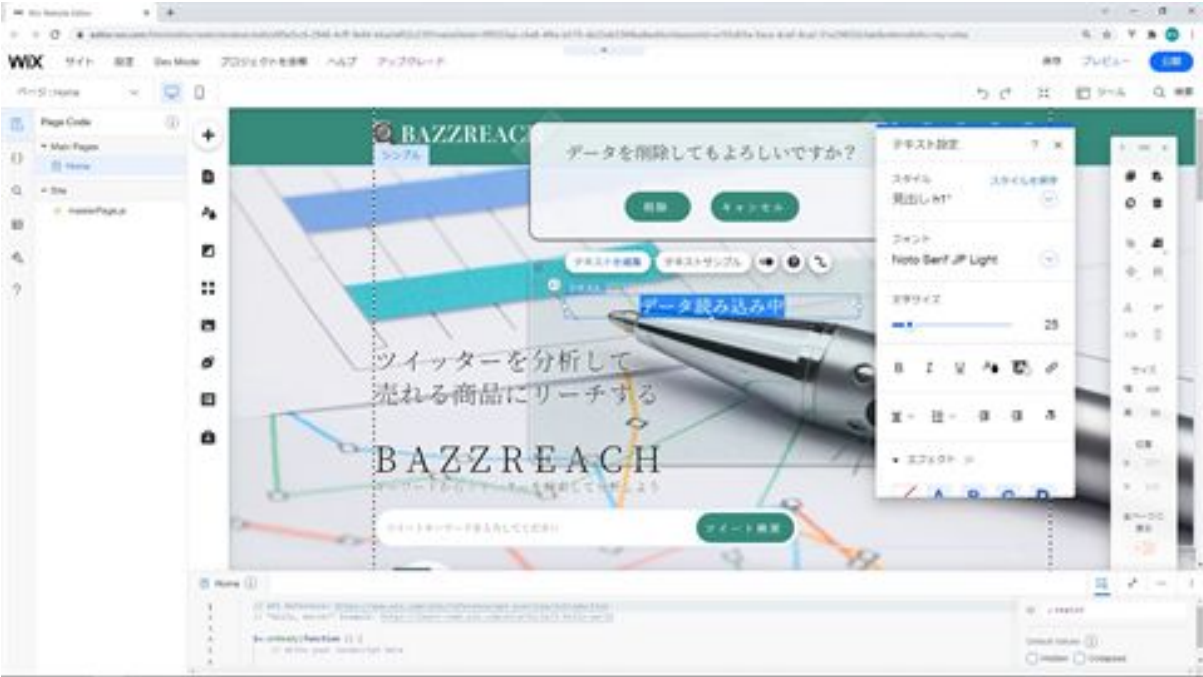


Next, change the roundness of the corners to 15px.

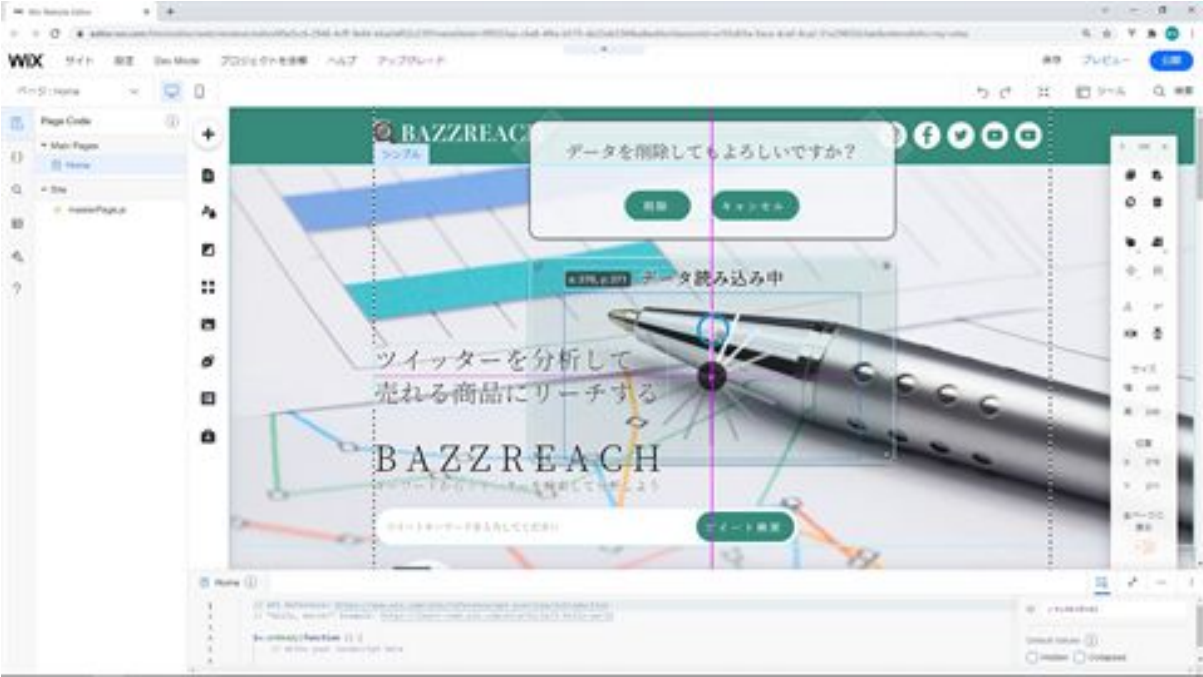


We will add text and a video to this redesigned box. Copy the text and move it into the box. Change the content of the text to Loading Data and change

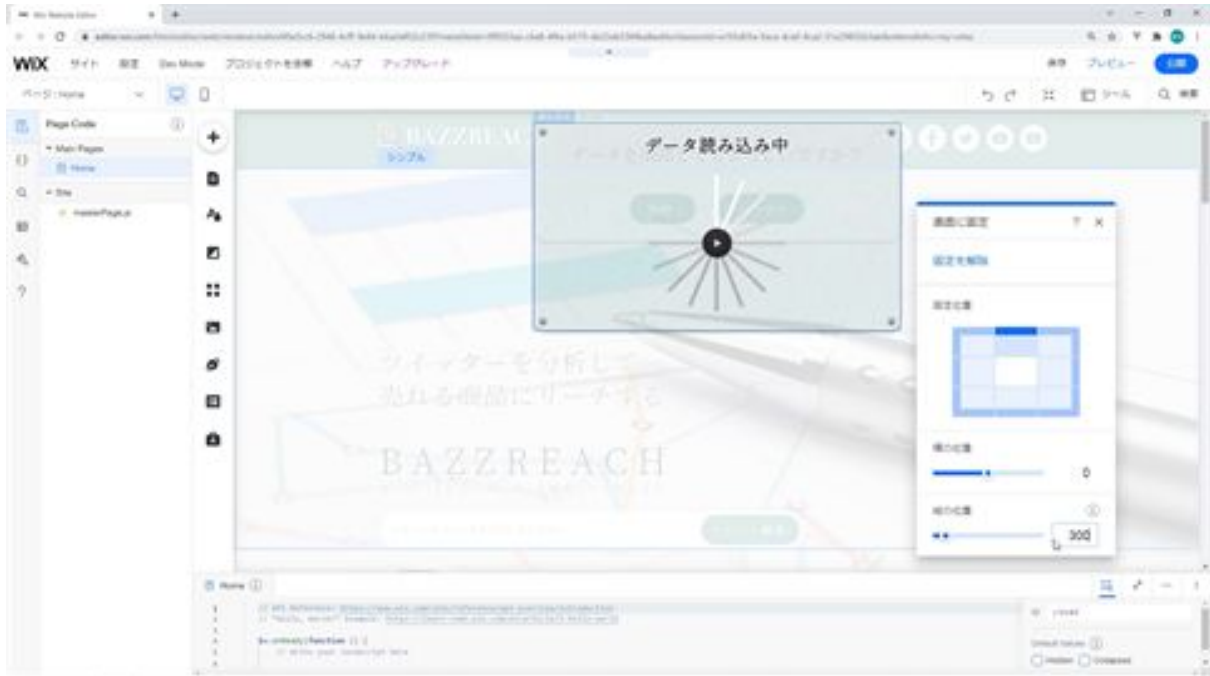
the font size to 25px. Then center the text. The text is hard to read, so change it to bold.



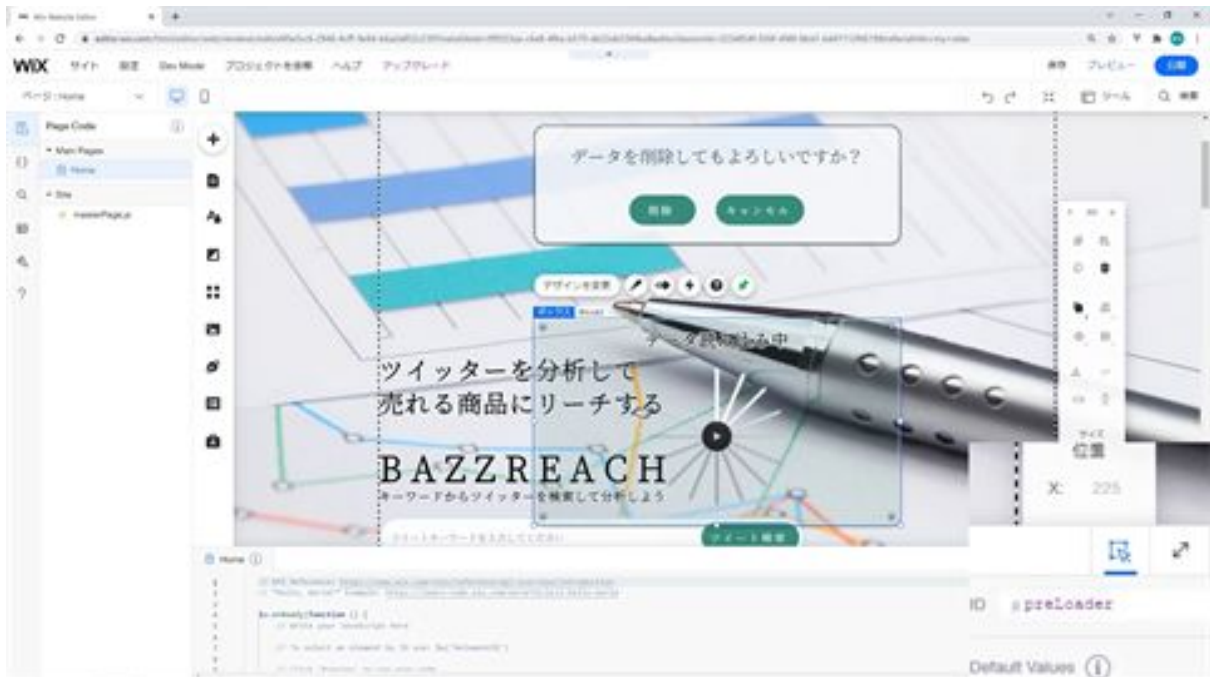
Next, move the added video to this box and adjust its position.



Then move this box to a fixed position on the screen, 300px from the top.



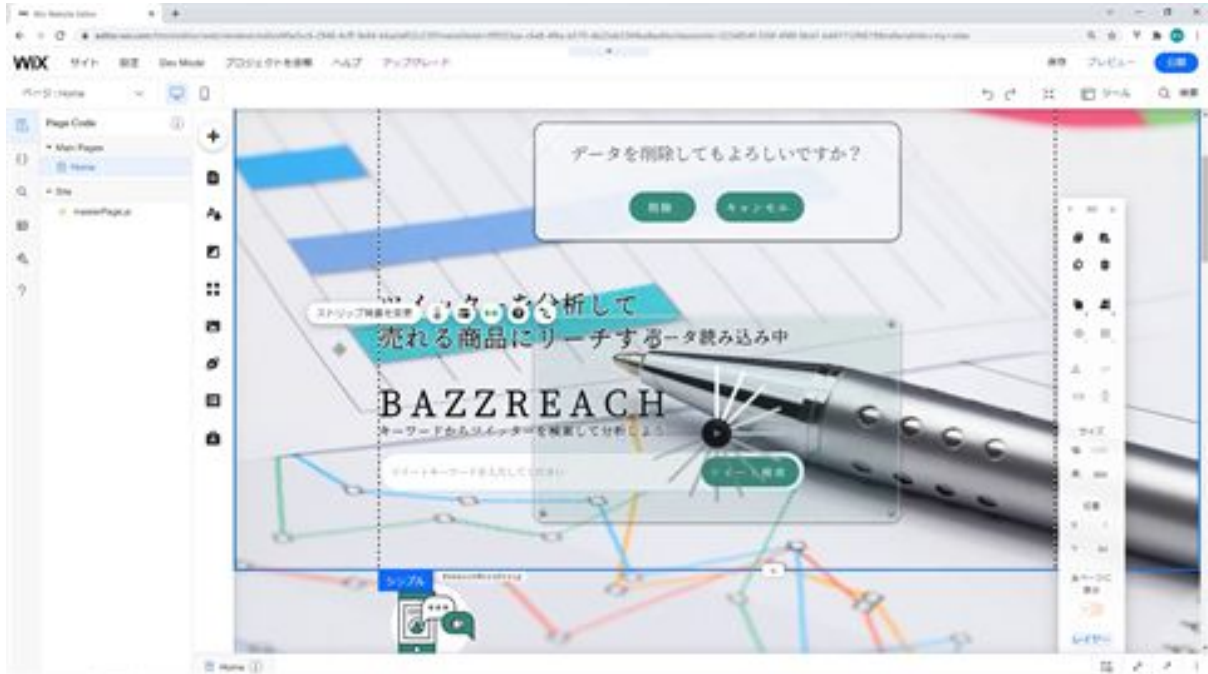
Next, set the ID of the box as preLoader.



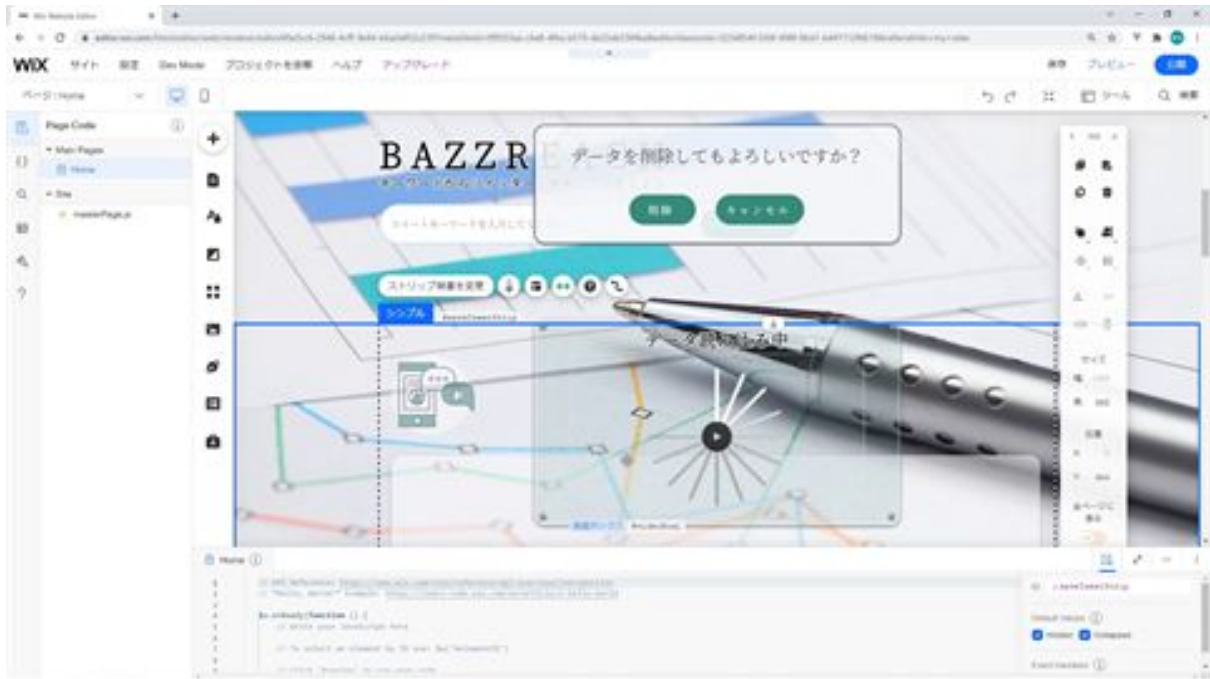
That's all there is to creating a preloader.

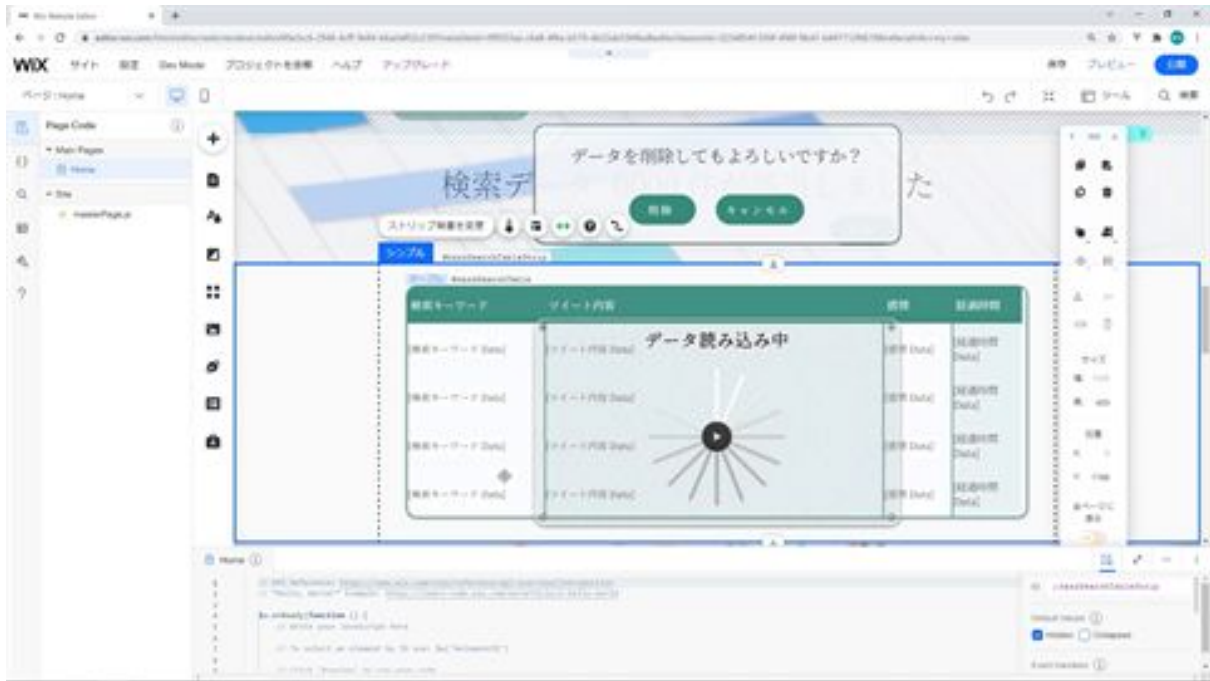
Let's adjust the page design

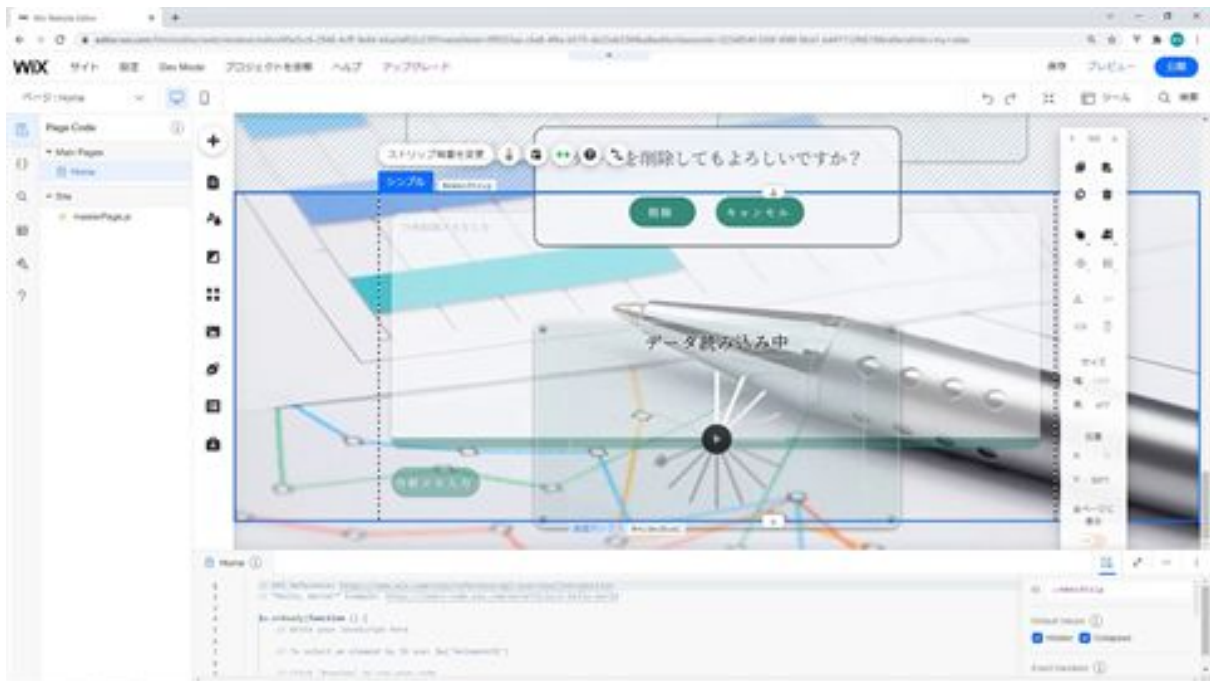
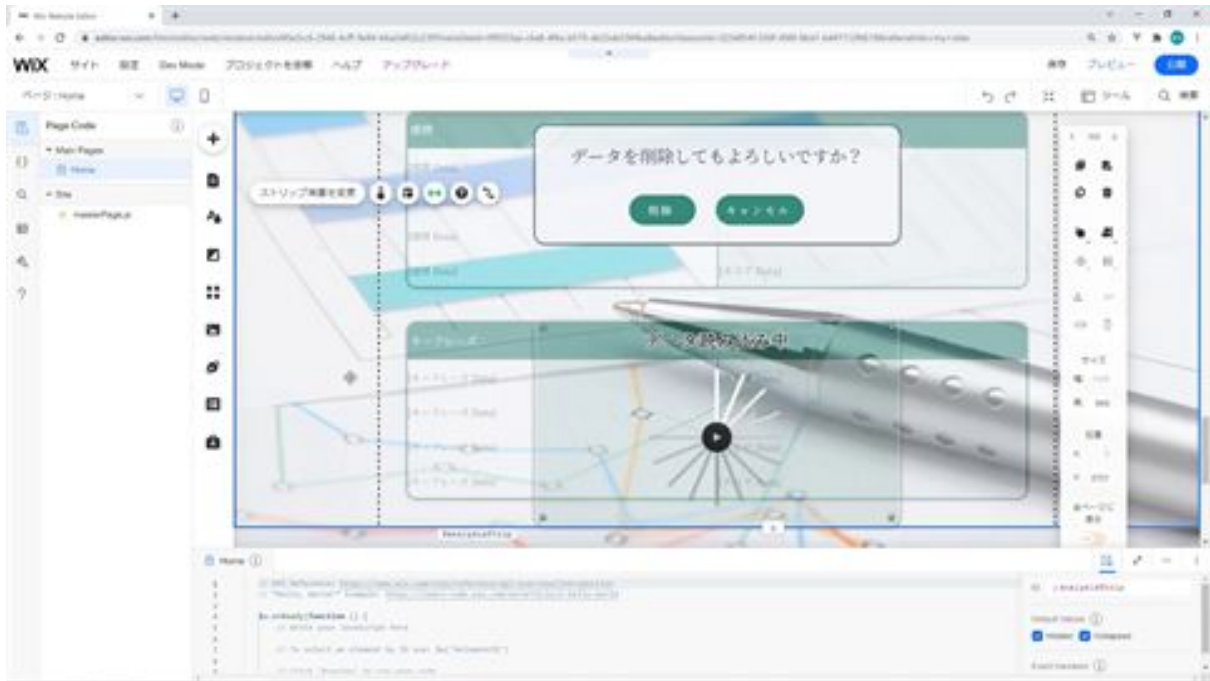
Next, I would like to modify the page design. First, the text in the copy, title, and sub-copy is a little hard to read, so we'll change it to bold text.



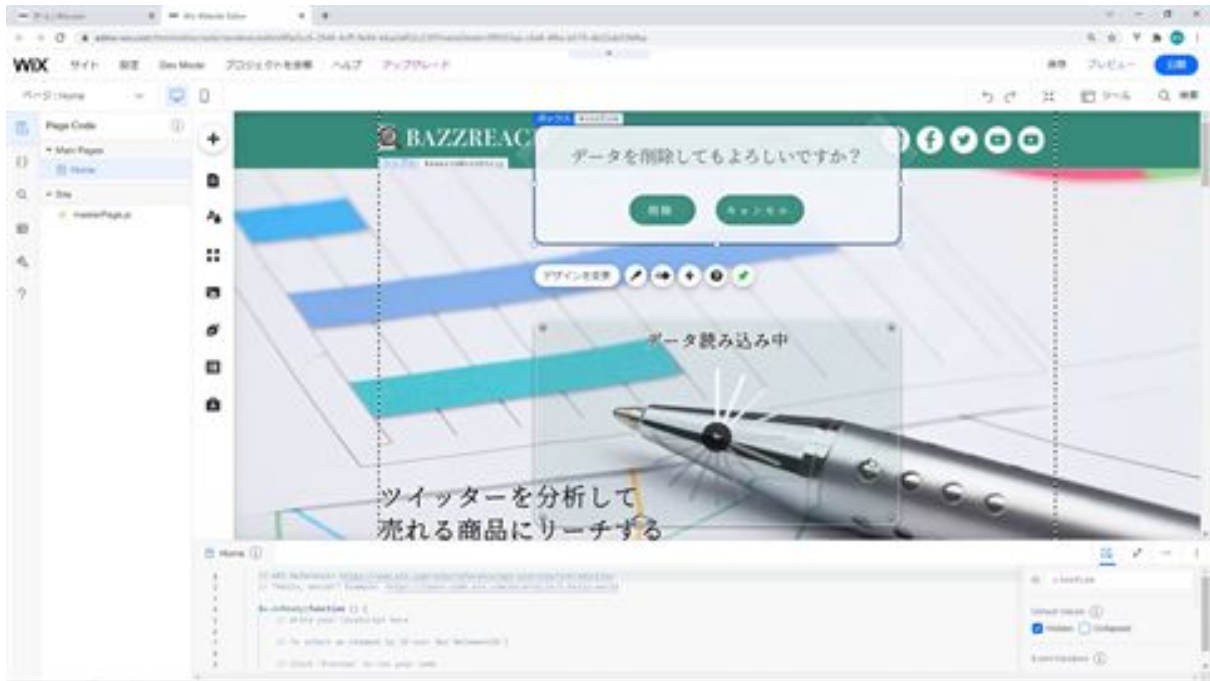
The next step is to set the Hide and Collapse settings for the strip. In the strip's initial display settings, there is a value called Default Values that can be used to hide and fold the strip. Check this box to set the default values for hiding and collapsing.



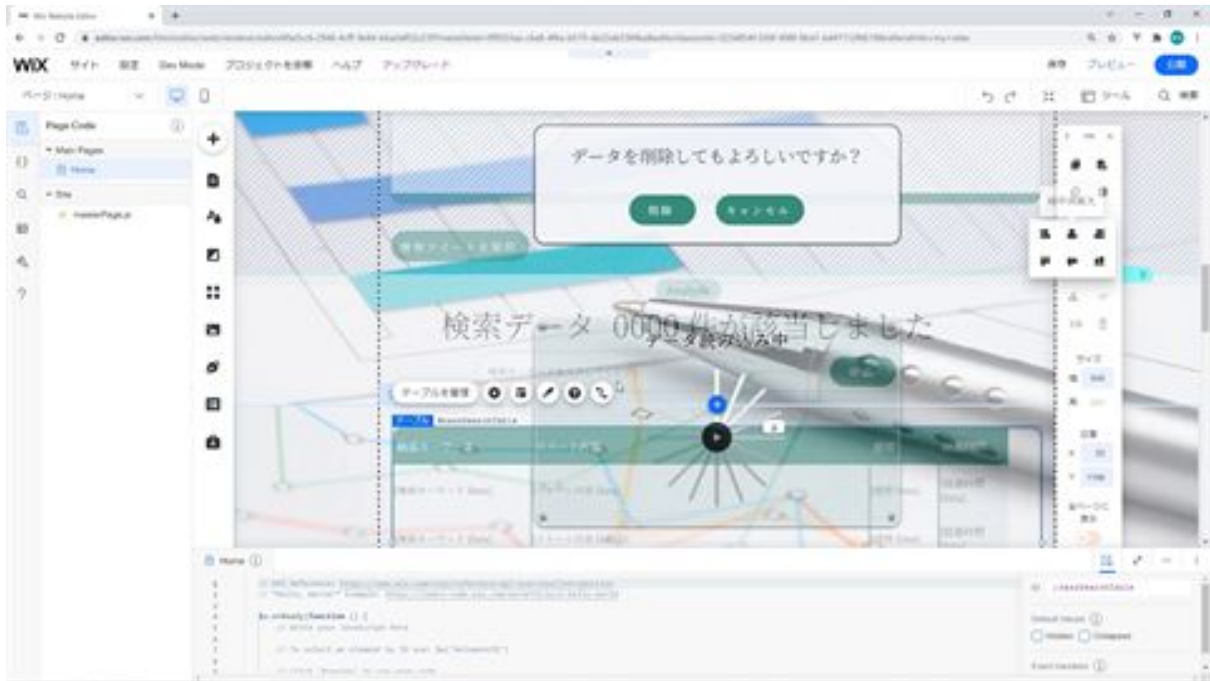




Next, set the confirmation screen and preloader to hidden.



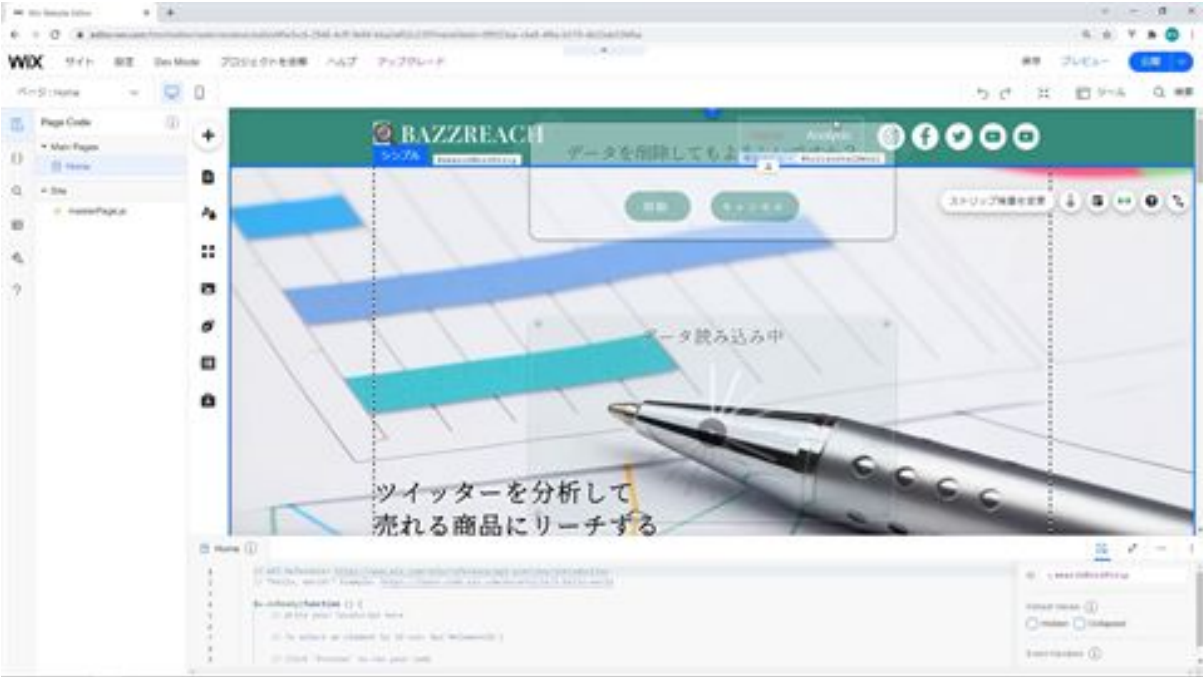
Next, we want the width of all the tables to match the other components, so we will modify it to 940px and center it vertically.



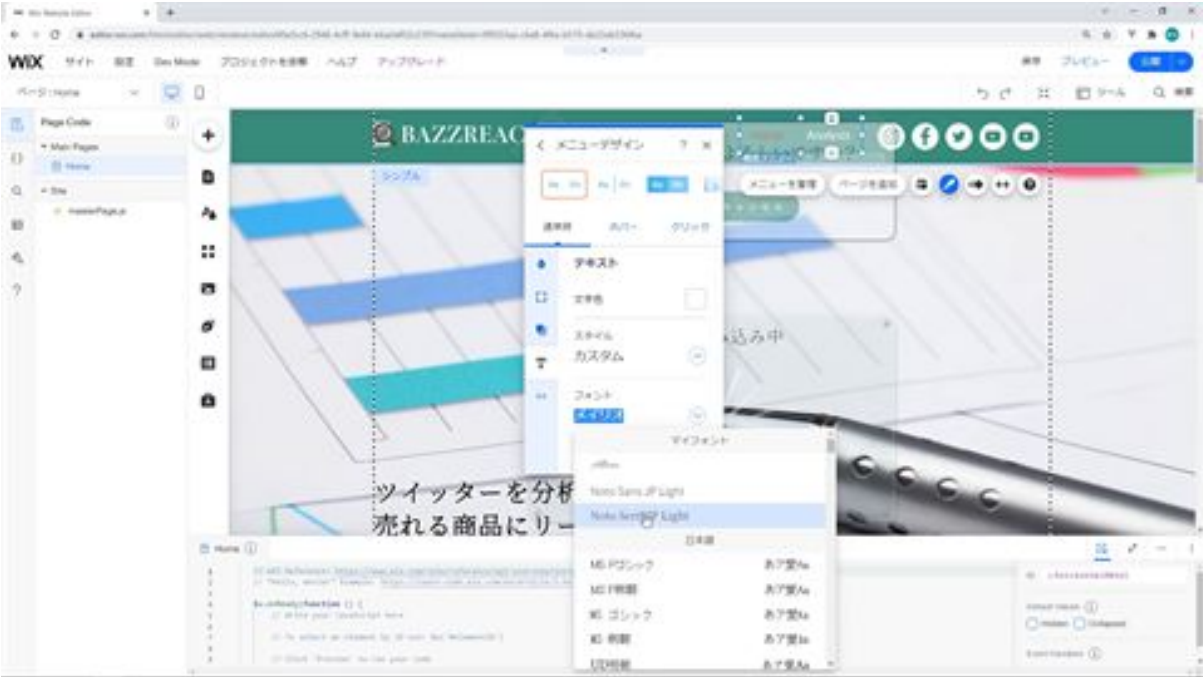
Next, we will modify the length of the items in the table for detailed analysis. Select Adjust Manually from Layout and click the Apply button. Then change the tweet content to 530px.



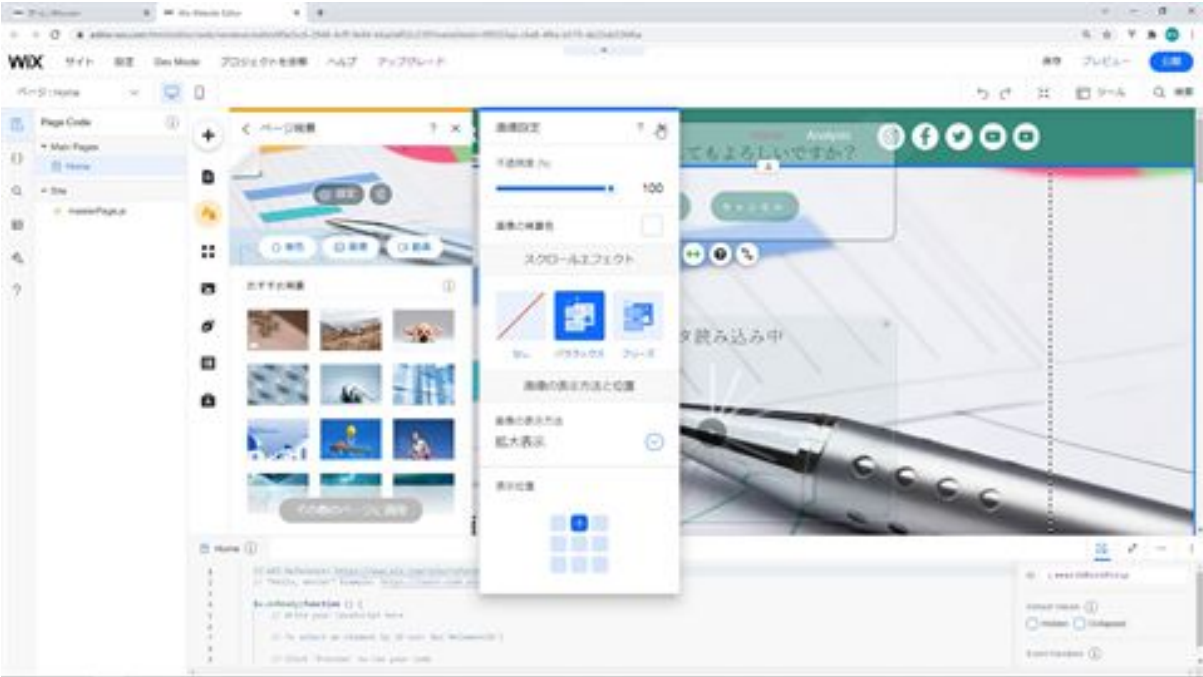
Next, go to Change Design, select Opacity and Color, and change the first/last column to 0%.



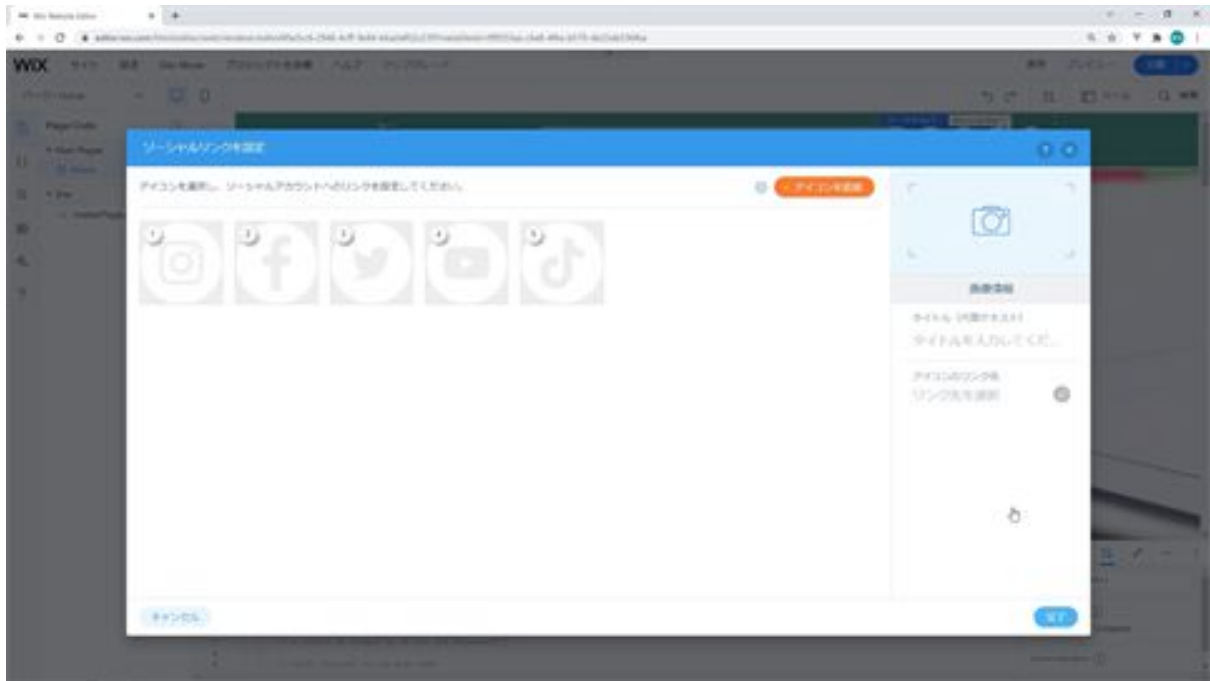
Next, fix the font in the menu as it does not match the other fonts. From Change Design, click Customize Design, select Text, and change the font.



The next step is to change the background scrolling settings. From Site Design, click the Edit Page Background button, and then click Settings. Under Scrolling Effects, click Parallax and change the display position to the top center one.



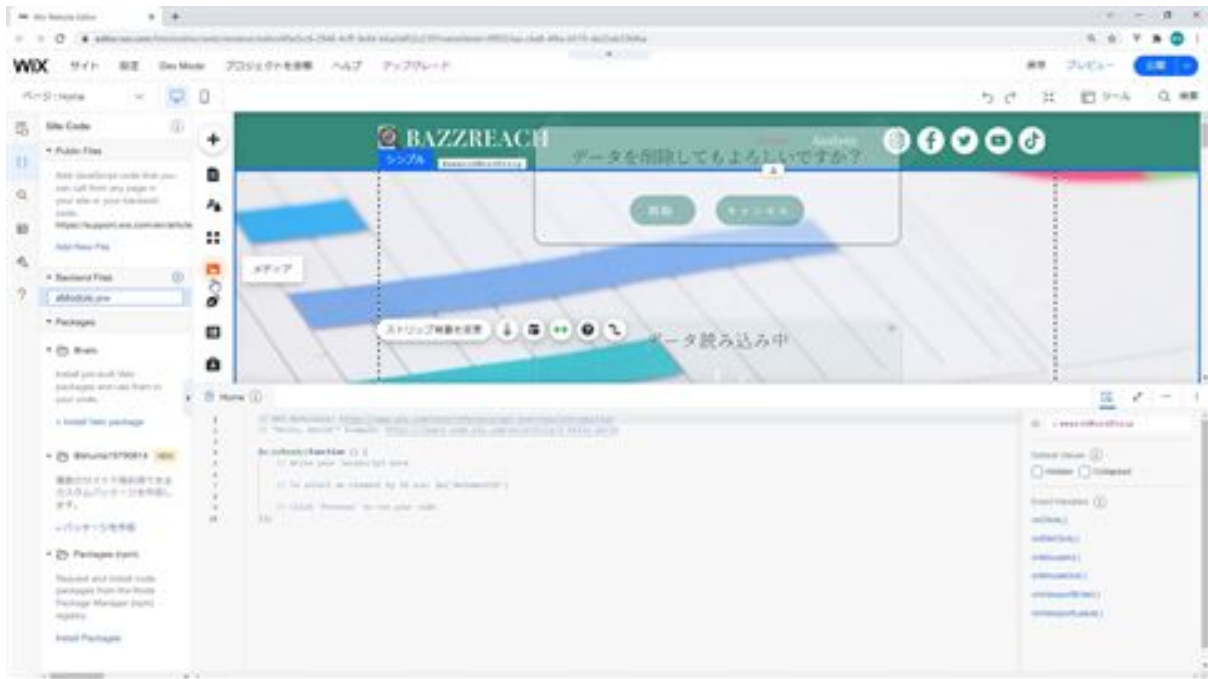
The next step is to correct the settings of this social bar, which is incorrect.



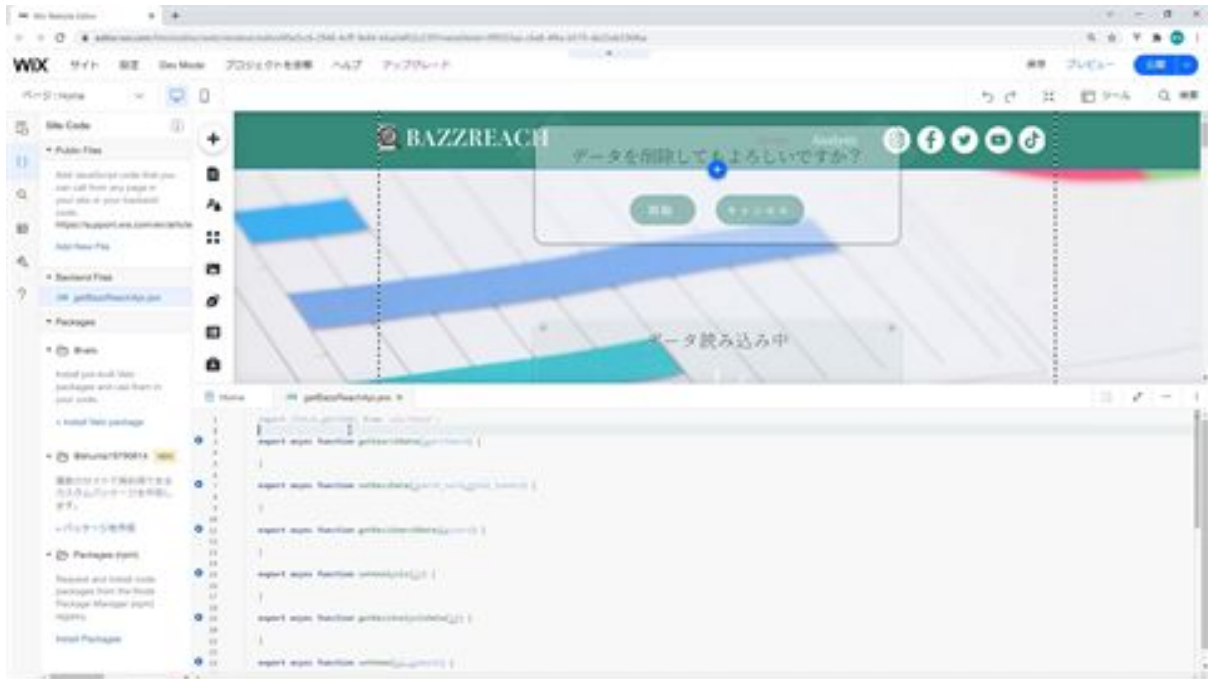
This is the end of the page design adjustments.

Let's create an API call function

Next, we will create a function that will call the API. First, click on Enable Development Mode in Dev Mode in the menu. Next, create a new web module from the Backend Files plus button on the left. The file name should be getBazzReachApi.jsw.



The next step is to create a function that calls the API.



These functions are called the API. These are function names that declare the use of these two functions from WixFetchApi. In practice, these two functions will be used to make API calls.

I will explain the functions in order from the top. First is the getSearchData function, which retrieves the Twitter information from the tweet keywords entered when the tweet search button is clicked.

The next function, setBazzData , will save the tweet information displayed in the text box to the database when the user clicks the "Save Tweet" button.

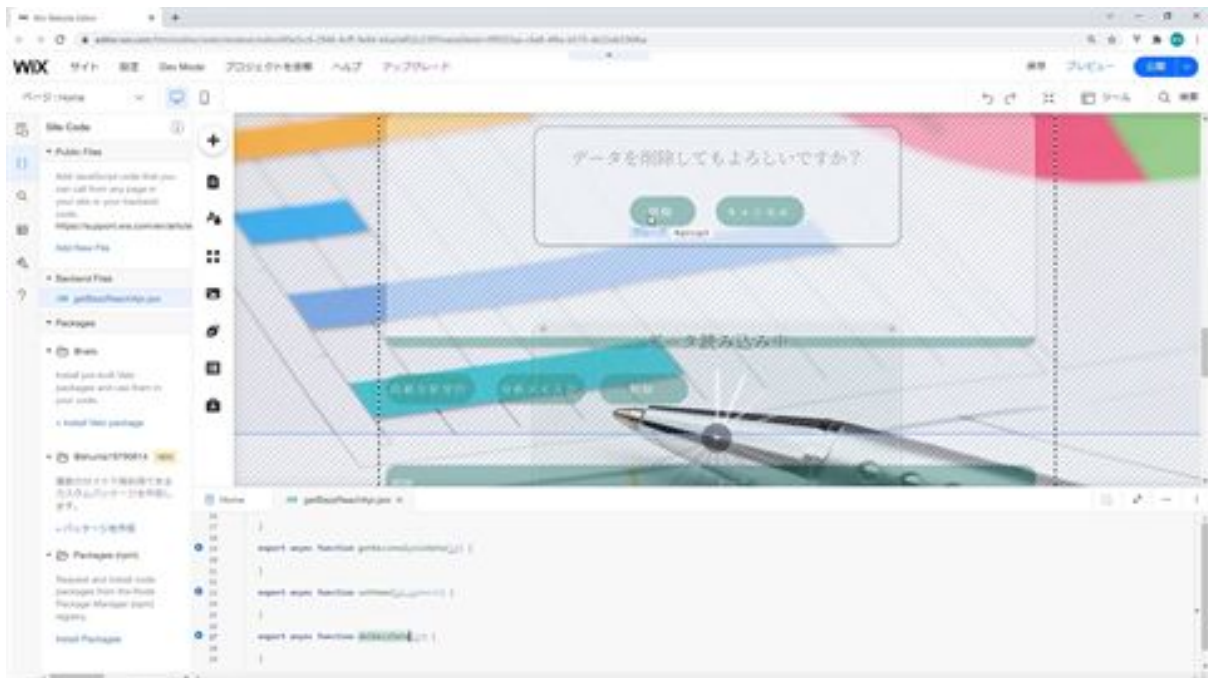
The next function, getBazzSearchData , will retrieve the tweet information stored in the database. The retrieved information will be displayed in a table.

The next function, setAnalysis, is a function to perform a detailed analysis of the tweet information selected in the table. The next function, getBazzAnalysisData , will retrieve the results of the detailed analysis. The results will be displayed in two tables.

The next function, setMemo, will save the memo information entered into the text box.

The next function, delBazzData, deletes the tweet information selected in this table. There is a delete button here, and when you click it, a confirmation dialog will appear. By clicking the delete button in the confirmation dialog, the data will be deleted.

The detailed processing of these functions will be explained when we describe each of them.

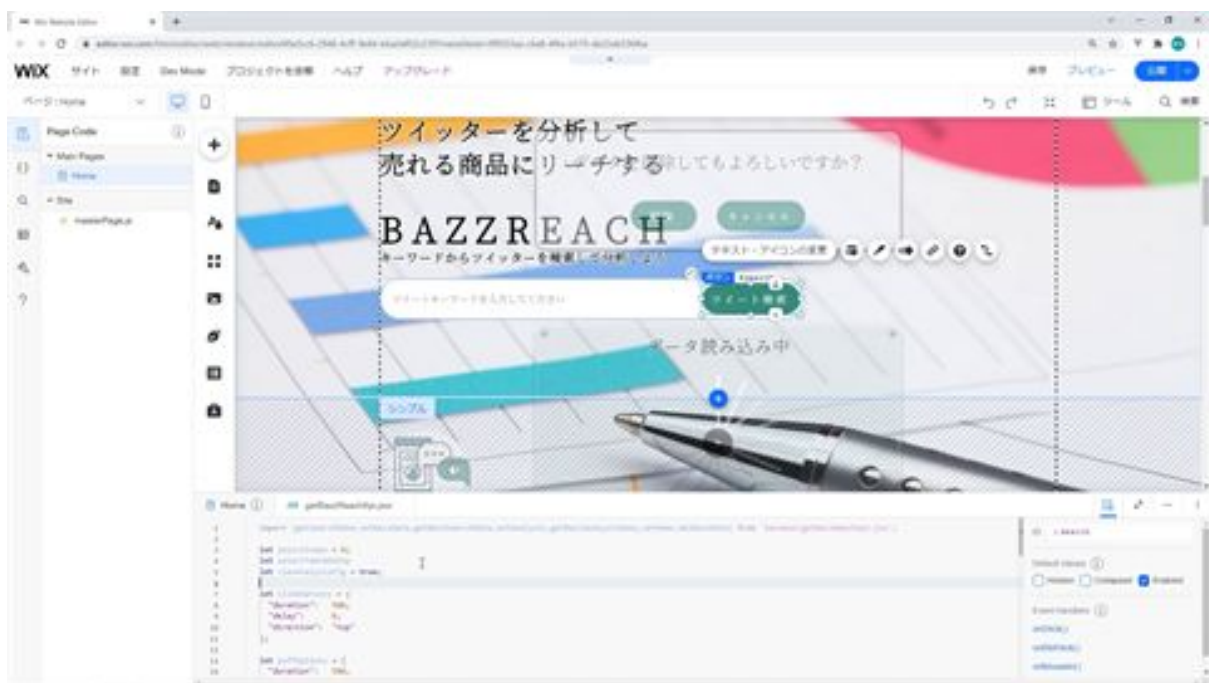


getBazzReachApi.jsw

```
import {fetch,getJSON} from 'wix-fetch';  
export async function getSearchData(searchWord) {  
}  
export async function setBazzData (search_word,total_tweets) {  
}  
export async function getBazzSearchData (keyword) {  
}  
export async function setAnalysis(id) {  
}  
export async function getBazzAnalysisData(id) {  
}  
export async function setMemo(id,comment) {  
}  
export async function delBazzData(id) {  
}
```

Let's create an event function

Next, we would like to create an event function. An event function is a function that is processed when a button is clicked or when a record in a table is selected. Let's start creating it. Click on the Home tab and start writing the process. First, describe the settings necessary for the process.

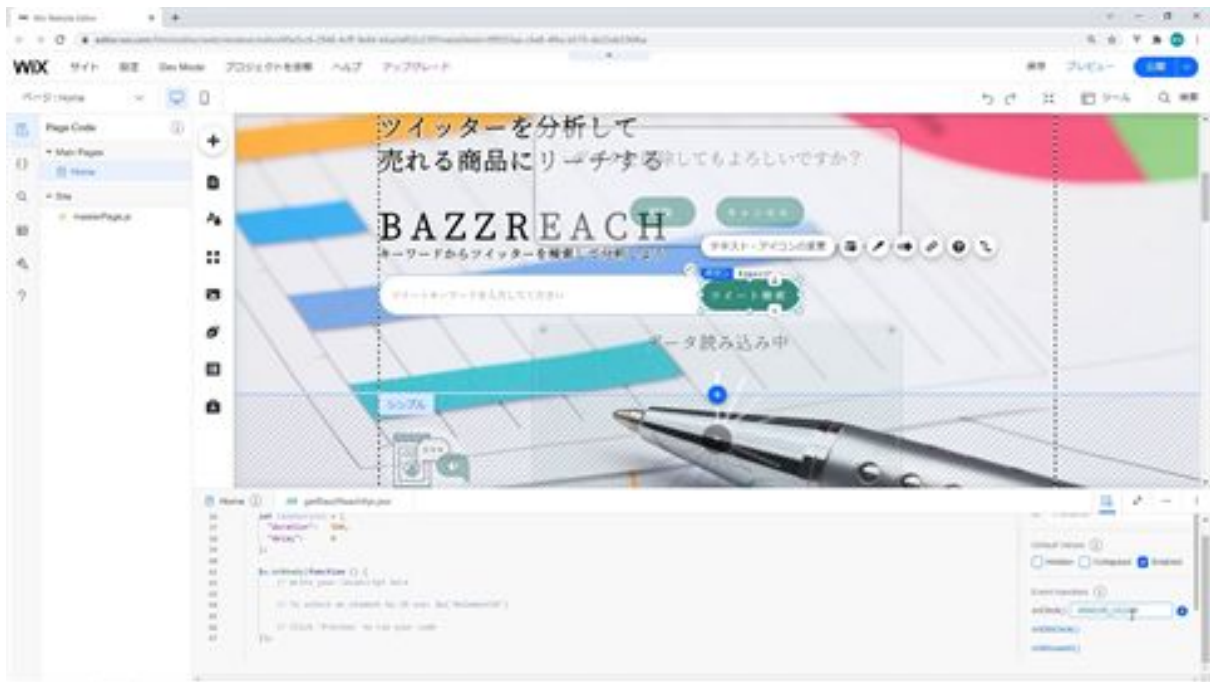


Then add the event function. To add an event function, select a button and you will see the event functions on the right side, such as `onClick`, `onDbClick`, and so on. Click on the event function you want to add. After clicking, click the plus button on the right side. The event function will then be added. We will create similar event functions for other buttons, table selections, etc. in the same way.

The first step is to set the search tweet as the save button.



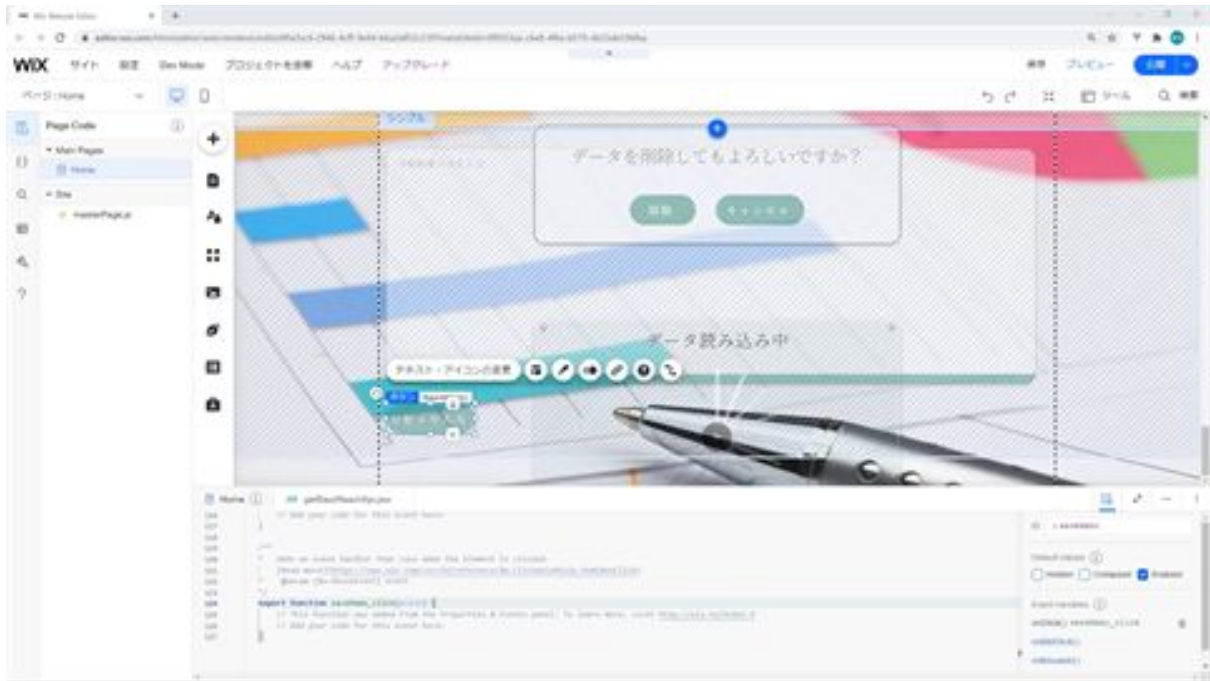
Next, add a click event function for the tweet search button as well.



In the search results table, add an event function when a row is selected.

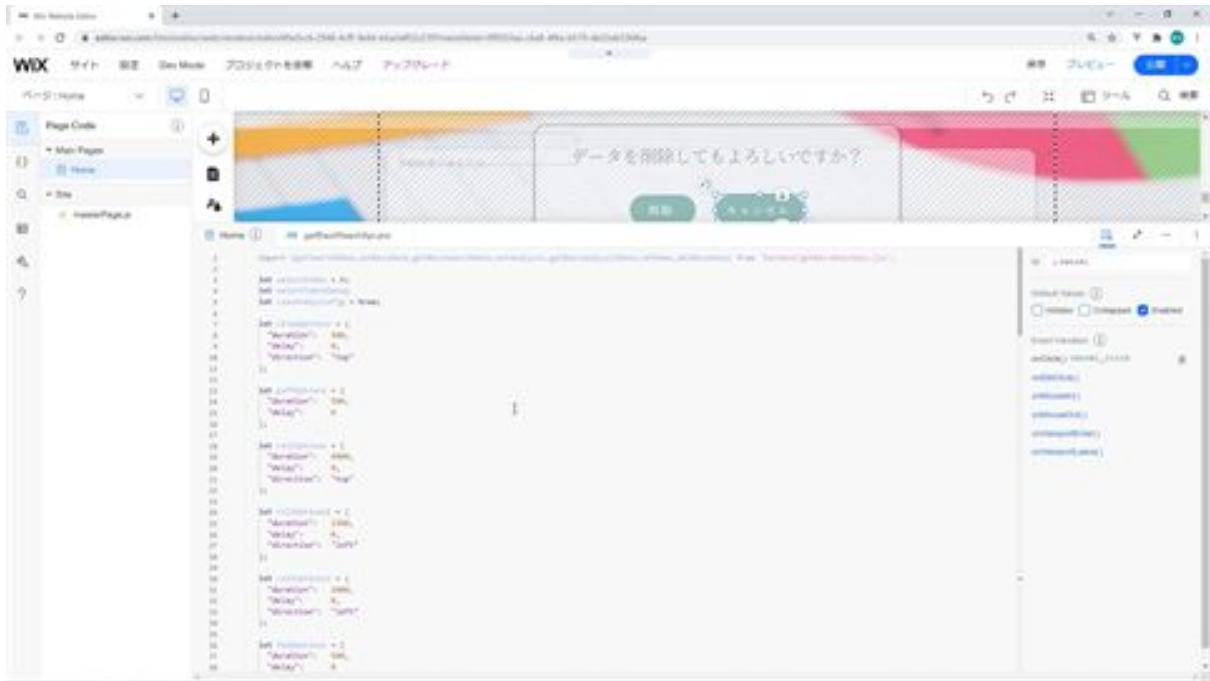


Add an event function for when the analysis memo input button is clicked.



Next, we will also create an event function for when the two buttons of the dialog are clicked.





onReady Function is the function for the initial display, where you will write the necessary processing for the initial display.



This concludes the creation of the event function.

home.js

```
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';
```

```
let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;
```

```
let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};
```

```
let puffOptions = {
  "duration": 500,
  "delay": 0
};
```

```
let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};
```

```
let rollOptions1 = {
  "duration": 1500,
  "delay": 0,
  "direction": "left"
```

```
};
```

```
let rollOptions2 = {  
  "duration": 1000,  
  "delay": 0,  
  "direction": "left"  
};
```

```
let fadeOptions = {  
  "duration": 500,  
  "delay": 0  
};
```

```
$w.onReady(function () {  
  
});
```

```
/**
```

```
* Adds an event handler that runs when the element is clicked.
```

```
[Read more]
```

```
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
```

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function search_click(event) {
```

```
}
```

```
/**
```

```
* Adds an event handler that runs when the element is clicked.
```

```
[Read more]
```

```
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
```

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function saveTweet_click(event) {
```

```
}
```

```
/**
```

```
* Adds an event handler that runs when the element is clicked.
```

```
  [Read more]
```

```
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
```

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function bazzSearch_click(event) {
```

```
}
```

```
/**
```

```
* Adds an event handler that runs when a table row is selected.
```

```
  [Read more]
```

```
(https://www.wix.com/corvid/reference/\$w.Table.html#onRowSelect)
```

```
* @param {$w.TableRowEvent} event
```

```
*/
```

```
export function bazzSearchTable_rowSelect(event) {
```

```
}
```

```
/**
```

```
* Adds an event handler that runs when the element is clicked.
```

```
  [Read more]
```

```
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
```

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function viewAnalysis_click(event) {
```

```
}
```

```
/**
```

```
* Adds an event handler that runs when the element is clicked.
```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function viewMemoInput_click(event) {
```

```
}
```

```
/**
```

```
* Adds an event handler that runs when the element is clicked.
```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function saveMemo_click(event) {
```

```
}
```

```
/**
```

```
* Adds an event handler that runs when the element is clicked.
```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function deleteConfirm_click(event) {
```

```
}
```

```
/**
```

```
* Adds an event handler that runs when the element is clicked.
```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function cancel_click(event) {  
  
}  
  
/**  
 *    Adds an event handler that runs when the element is clicked.  
 *    [Read more]  
 *    (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)  
 *    @param {$w.MouseEvent} event  
 */  
export function delete_click(event) {  
  
}
```


Use the log function to check the log

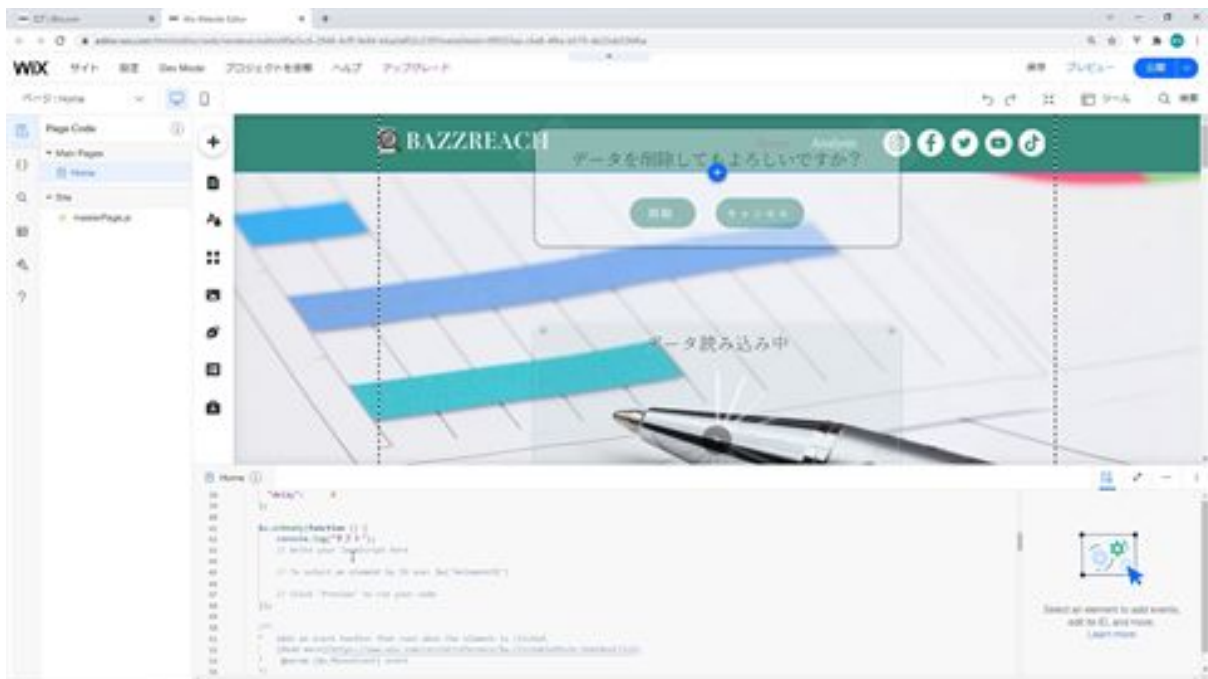
Now let's talk about the log function to check the logs. From the Developer Tools on the left side of your site's dashboard, click Logs.



Open the Log screen and click the Open button in the View Site Events section. This will open the Site Events screen.



Now modify it so that it displays the edit screen and shows "test" in the console in the initial view.



When the fix is complete, publish it and output the log. When you do so, the site event screen will now say test.

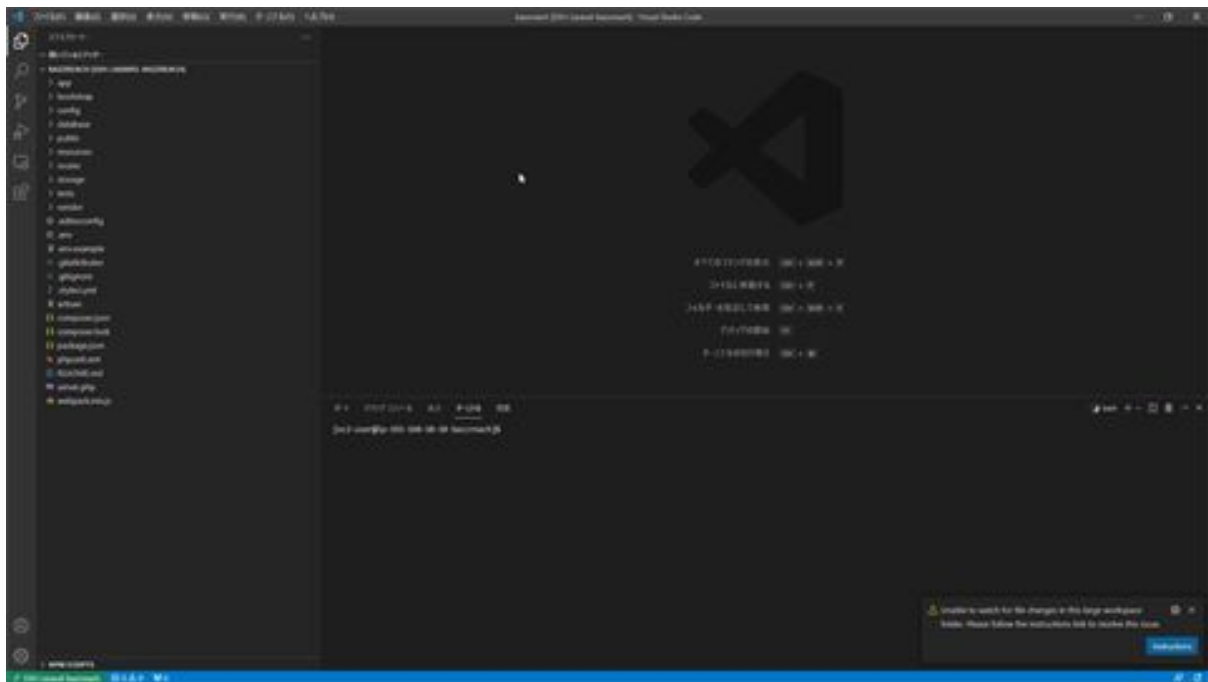


In this site's event screen, the log output to the console log and the error message when connecting to the API are displayed in the log. If you are testing API connections, you can use this log function to develop your system smoothly, so please try using it. This is the end of the explanation of the log function.

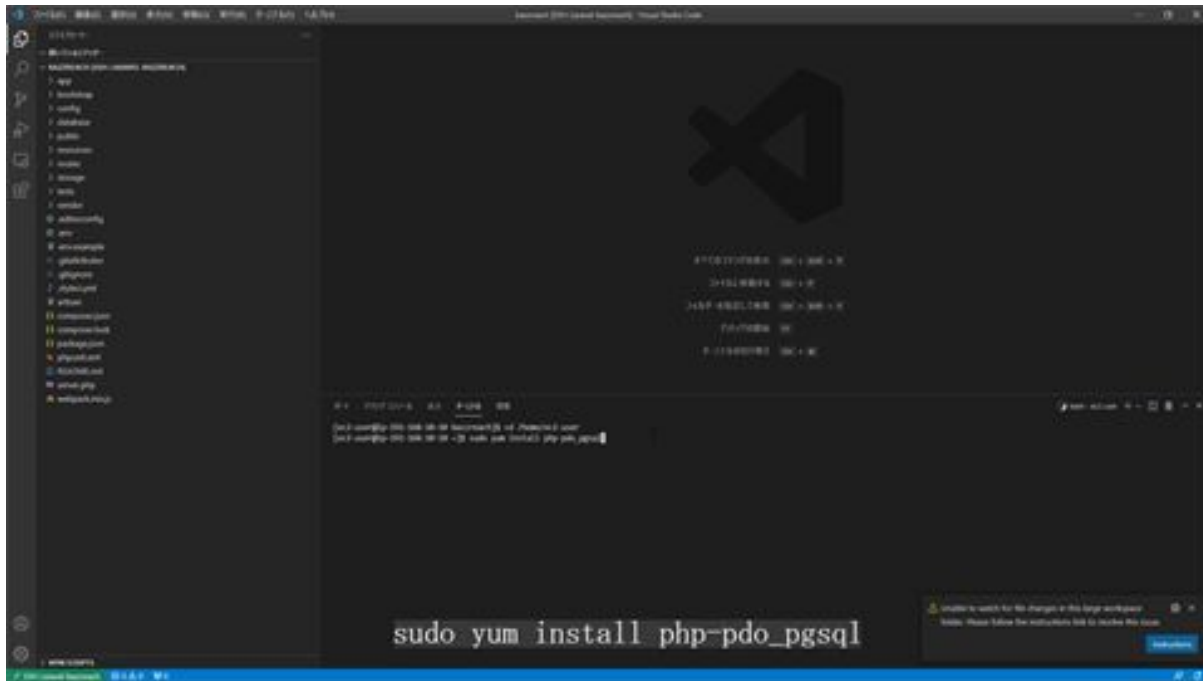
Chapter 8: Create a Wix Application with Laravel API

Let's set up a connection from Laravel to RDS

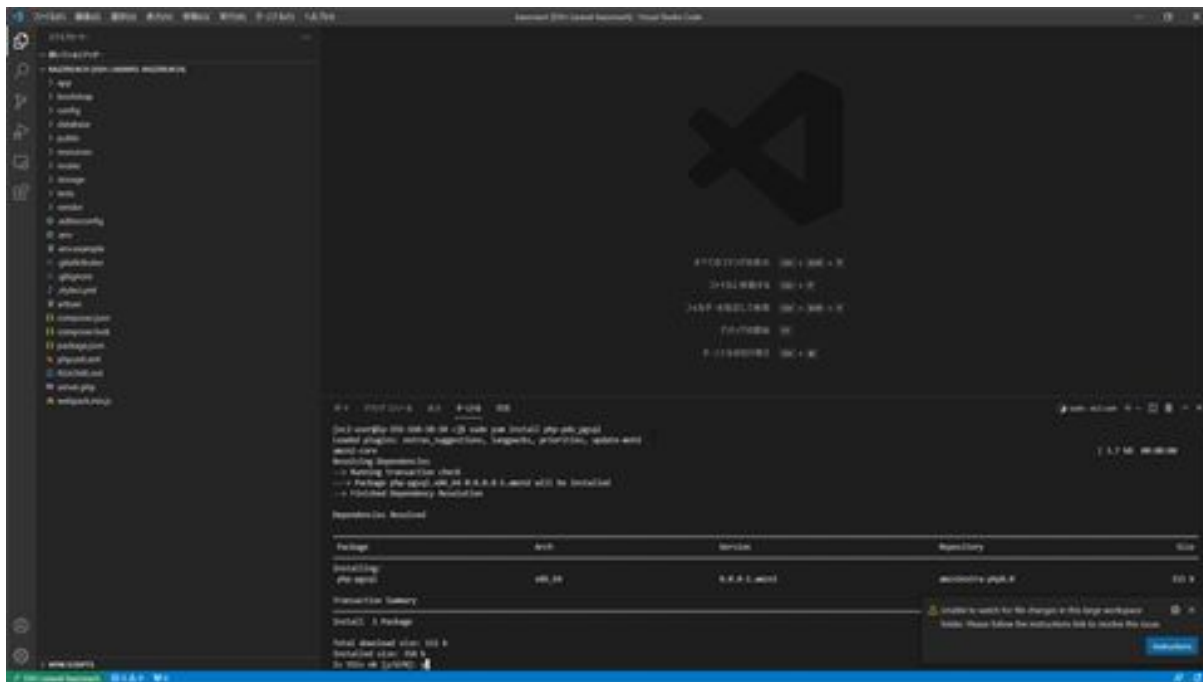
From this chapter, we will create a Laravel application. First, we will set up a connection from Laravel to the RSD: open Visual Studio Code and connect to an EC2 instance.



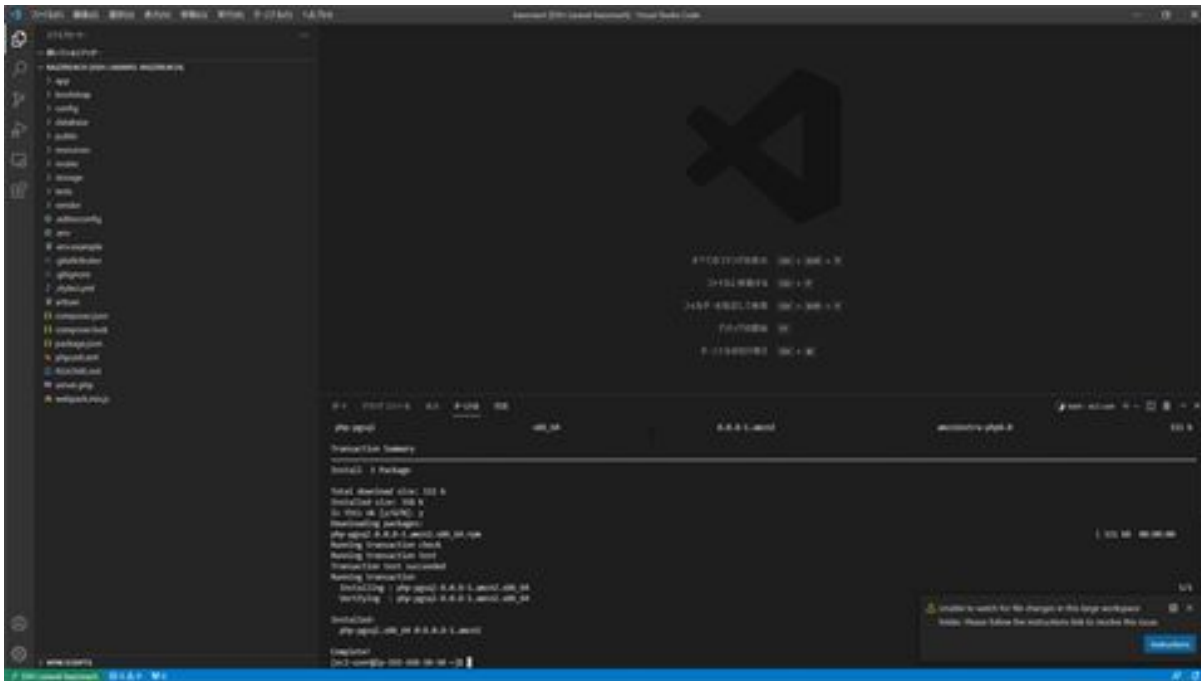
Next, go to the ec2-user folder from the terminal. Then install the PostgreSQL driver by running `sudo yum install php-pdo_pgsql`.



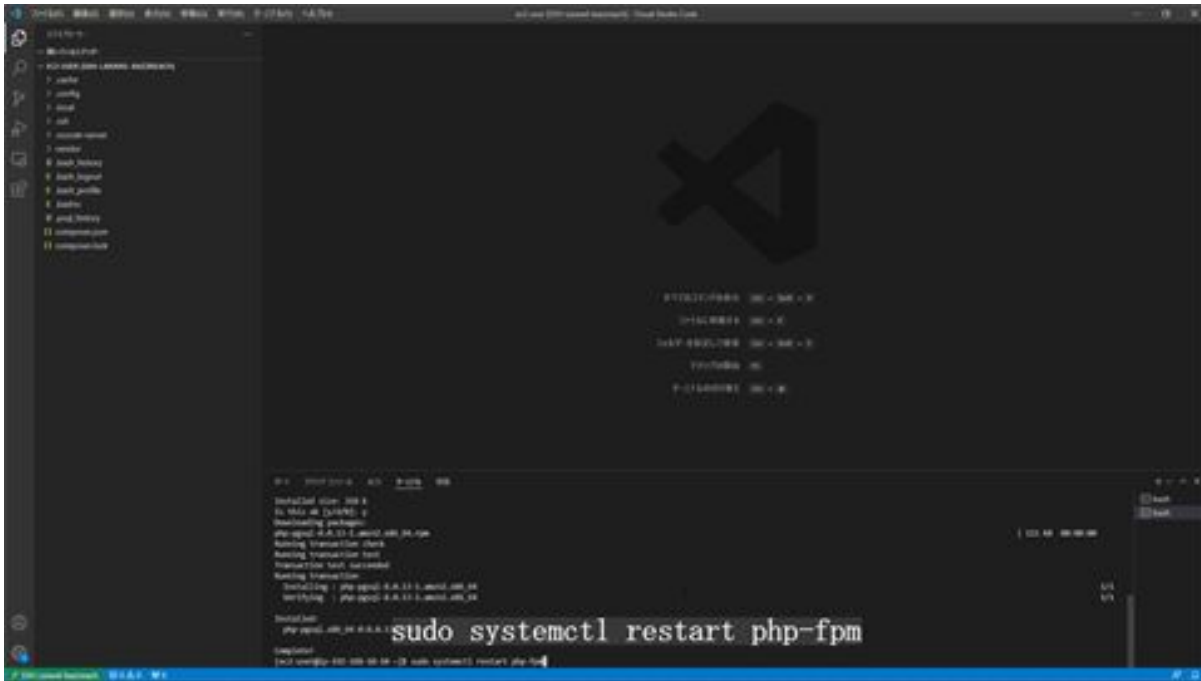
During installation, type y and press enter.



PostgreSQL driver installation is complete.

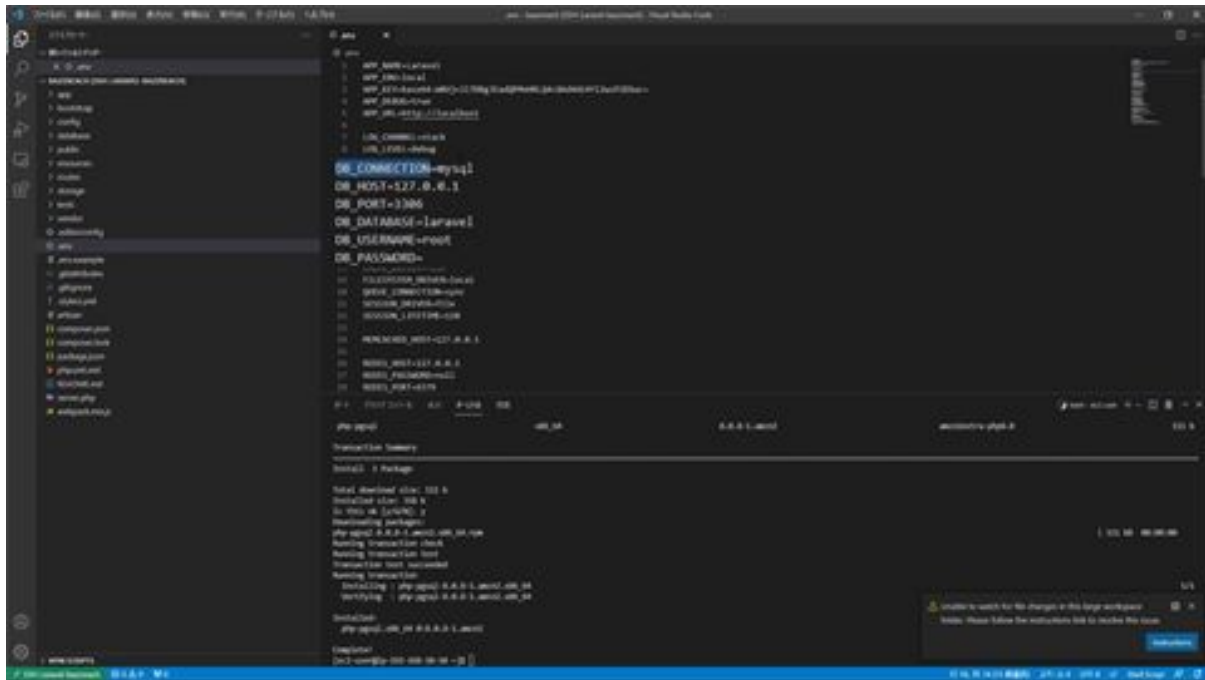


Next, restart PHP-FPM to reflect the changes in the PHP configuration file php.ini.



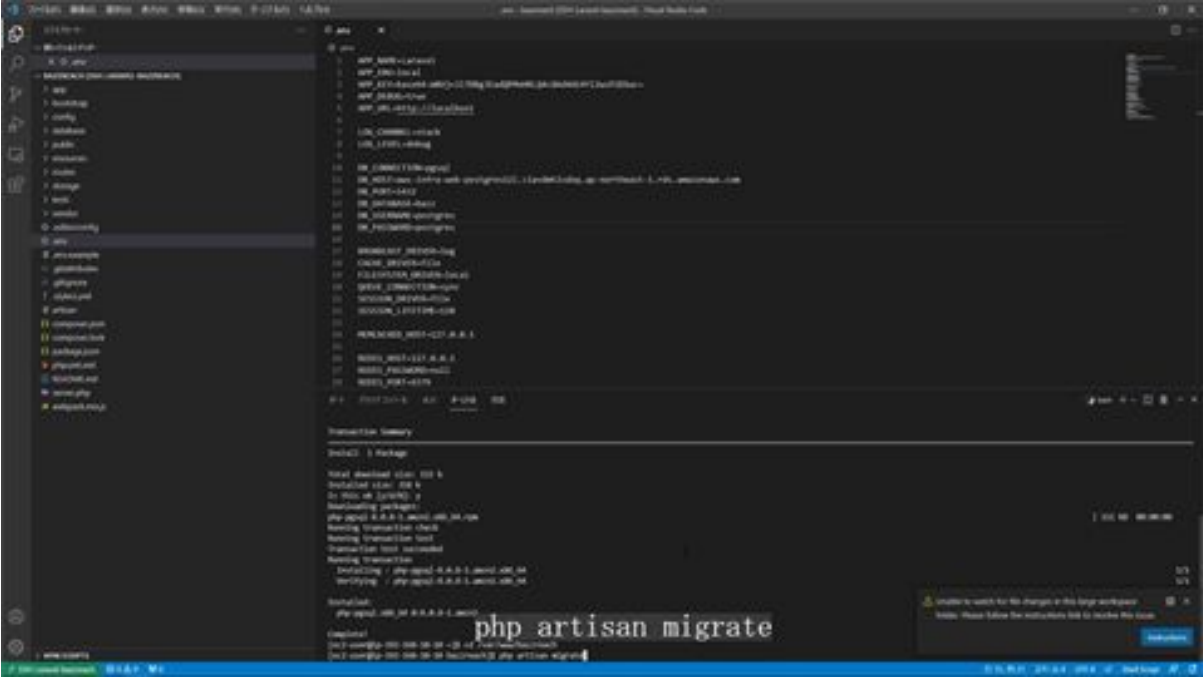
Next, in Visual Studio Code, click on Files in the top menu to open folders and view the bazzreach folder. Then open the .env file in

this bazzreach folder and change the DB settings.



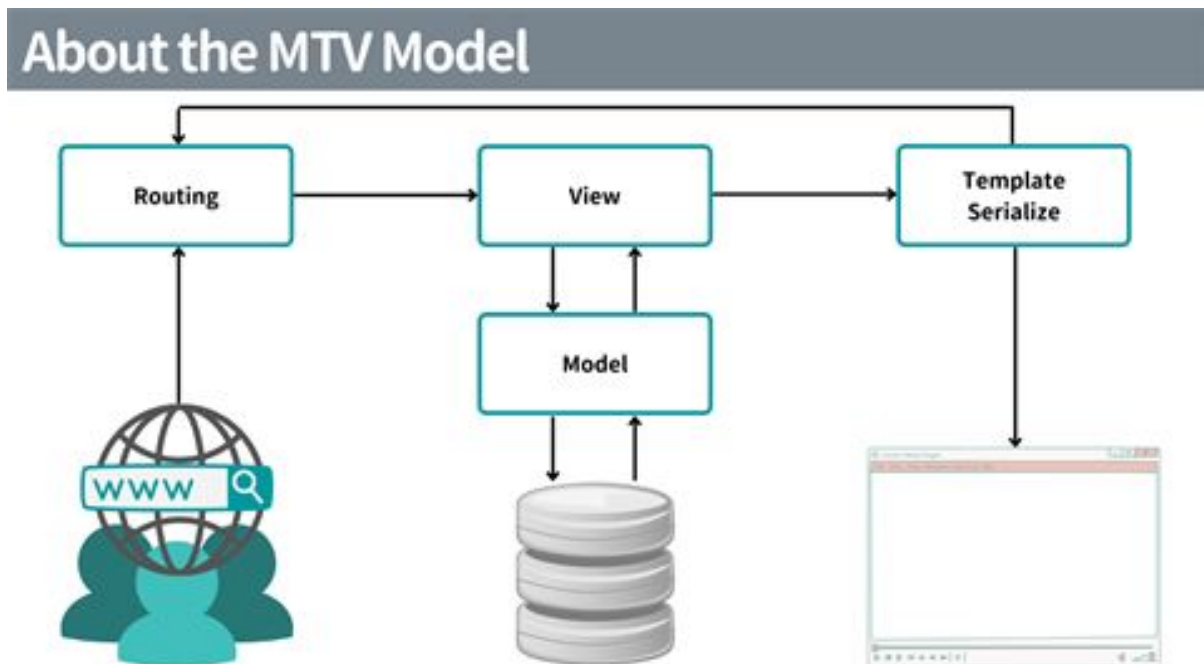
When you open the file, you will see a section marked `DB_CONNECTION`. This `DB_CONNECTION` is marked `mysql`, so enter `pgsql` here. Next, enter the RDS endpoint in the `DB_HOST` field, which can be found in the Connection and Security tab of the RDS screen you just created. Copy the endpoint from there and paste it here. For the next `DB_PORT`, enter 5432. For `DB_DATABASE` enter the DB name `bazz` when you created the RDS instance, for `DB_USERNAME` and `DB_PASSWORD` enter the postgres you set up when you created the instance.

Next, perform the migration. This migration will create the tables and other information configured on the Laravel side in the RDS tables, allowing for data registration and retrieval. Execute "php artisan migrate" in the terminal to execute the migration.



Migration table created successfully. This completes the RDS connection setup from Laravel.

About the MVC Model



The MVC model is a concept that divides the processing of a program into different roles. The term "a" is also used to refer to the following. First, I would like to explain the sequence of steps in this model.

First, when a user accesses the URL of the Laravel application, there is a rule set called "routing" that links the URL and the controller, and the controller side processing is executed according to that rule. This controller is the part that operates the view and the model. It receives requests from the view, sends processing commands to the model, receives processing results from the model, and returns them to the view as a response. It acts as an intermediary between the view and the model. A lot of processing is described here.

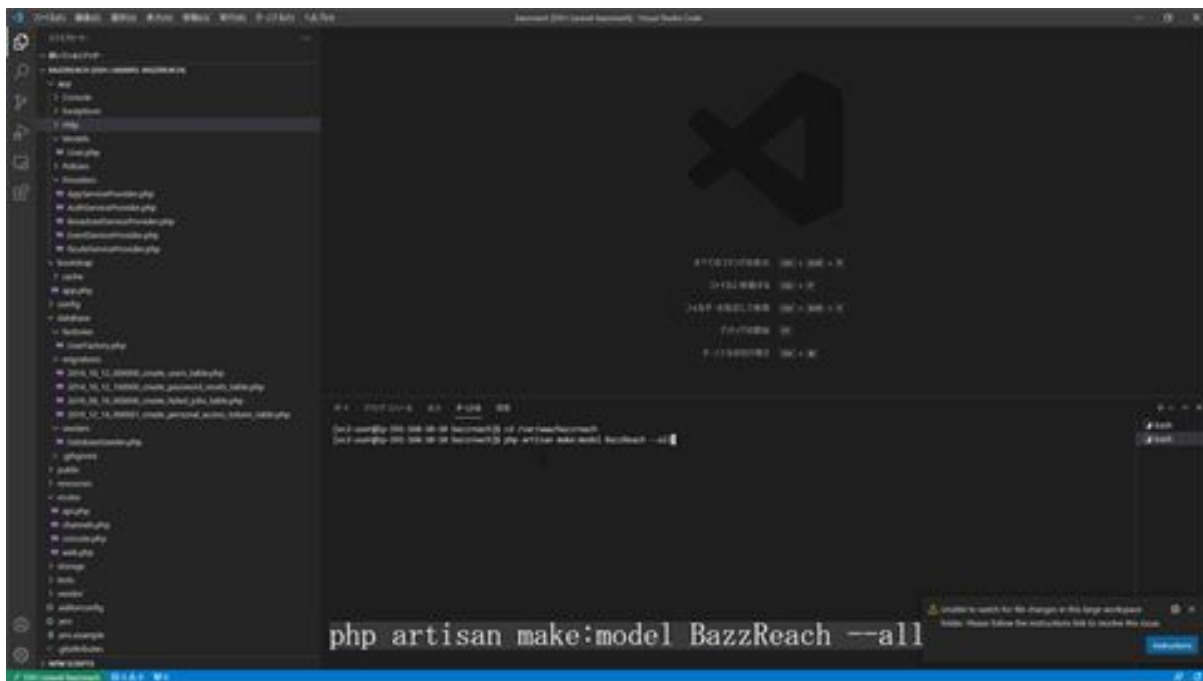
Next is the model, which is responsible for implementing functions for exchanging data with the database and for converting data acquired from the database into a format that is easy to handle programmatically.

Next is the view, which is originally the part that dynamically generates HTML to be displayed in a Web browser. However, since Wix will be used as the screen this time, the model data will be returned in JSON format from the controller without using the view. Wix will then receive the returned data and display it to show the data in the database.

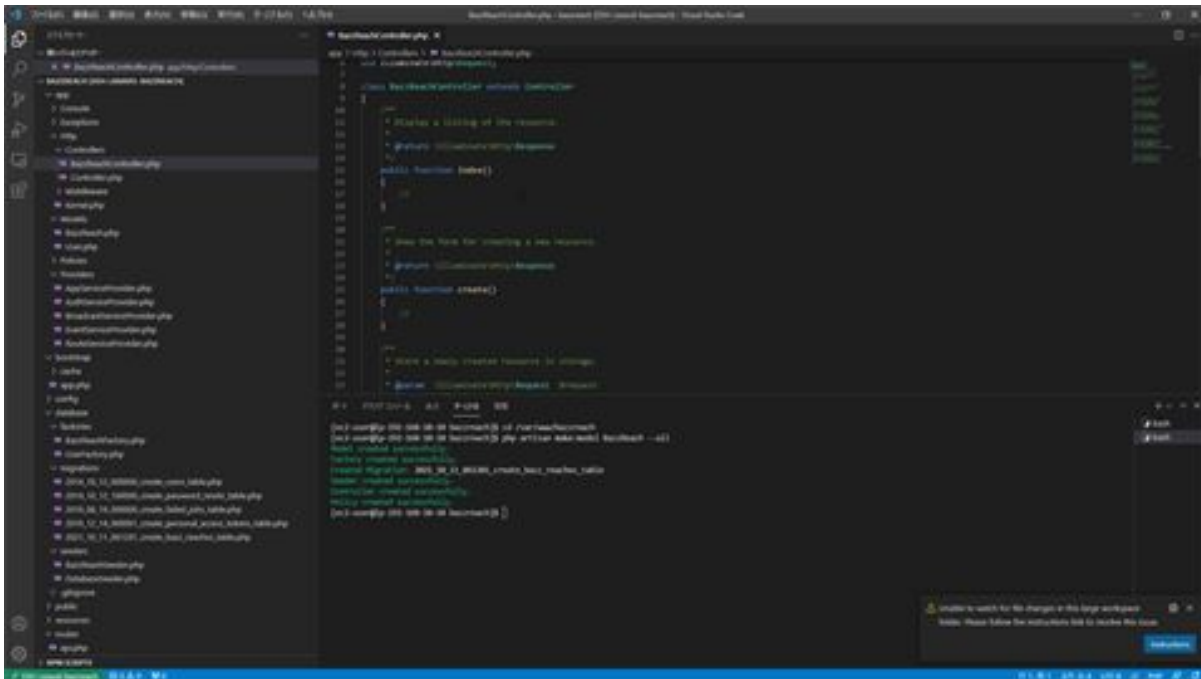
The advantage of the MVC model is that the view, model, and controller are divided into separate roles, which makes application development more efficient. The disadvantage is that controller processing tends to be large. The reason for the increased processing is that the number of models increases too much and the controller becomes more dependent on them. This is the end of the explanation of the MVC model.

Create Controllers and Models

Now we will create the controller and model. First, navigate to the bazzreach folder in the Visual Studio Code terminal. Next, create the controller and model with the command In the terminal, run php artisan make:model BazzReach --all.



Then a BazzReach controller was created in app/Http/Controllers and a BazzReach model was created in app/Http/ModelsModels.



If you check the controller source, you will see the index, create, store, show, edit, update, and destination listed.

This is called an action method. the index is for the initial screen display, create is for registration screen display, the store is for registration processing, the edit is for update screen display, the update is for update processing, the show is for detail display, and delete is for deletion processing. Applications can be easily created. We will explain how to link these processes to URLs in the next section, "Routing Configuration. This is all for the creation of controllers and models.

BazzReachController.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Models\BazzReach;
use Illuminate\Http\Request;

class BazzReachController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
```

```

    //
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function show( BazzReach $ bazzReach )
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function edit( BazzReach $ bazzReach )
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, BazzReach $ bazzReach )

```

```
{
    //
}

/**
 * Remove the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function destroy( BazzReach $ bazzReach )
{
    //
}
}
```

BazzReach.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class BazzReach extends Model
{
    use HasFactory;
}
```

create_bazz_reaches_table.php

```
<?php
```

```
use Illuminate\Database\Migrations\Migration;
```

```
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;
```

```
class CreateBazzReachesTable extends Migration
```

```
{
```

```
    /**
```

```
     * Run the migrations.
```

```
     *
```

```
     * @return void
```

```
     */
```

```
    public function up()
```

```
    {
```

```
        Schema::create('bazz_reaches', function (Blueprint $table) {
```

```
            $table->id();
```

```
            $table->string('search_word');
```

```
            $table->text('total_tweets')->nullable();
```

```
            $table->text('comment')->nullable();
```

```
            $table->timestamps();
```

```
        });
```

```
    }
```

```
    /**
```

```
     * Reverse the migrations.
```

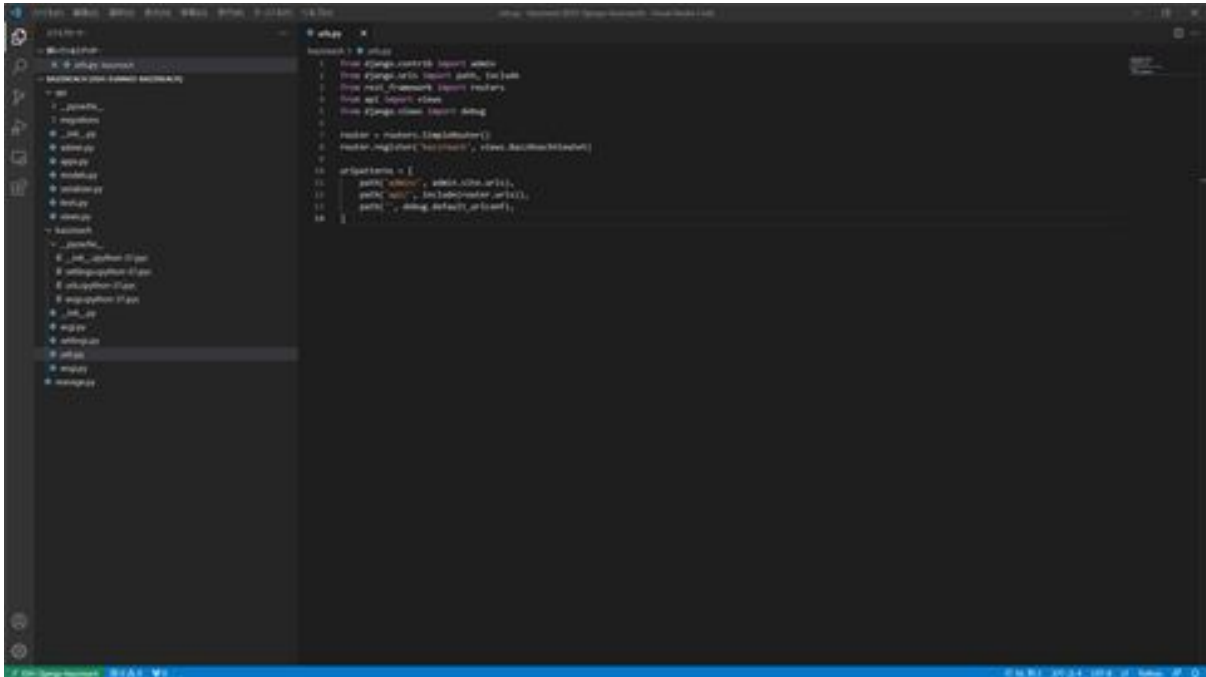
```
     *
```

```
     * @return void
```

```
     */
```

```
public function down()
{
    Schema::dropIfExists('bazz_reaches');
}
}
```

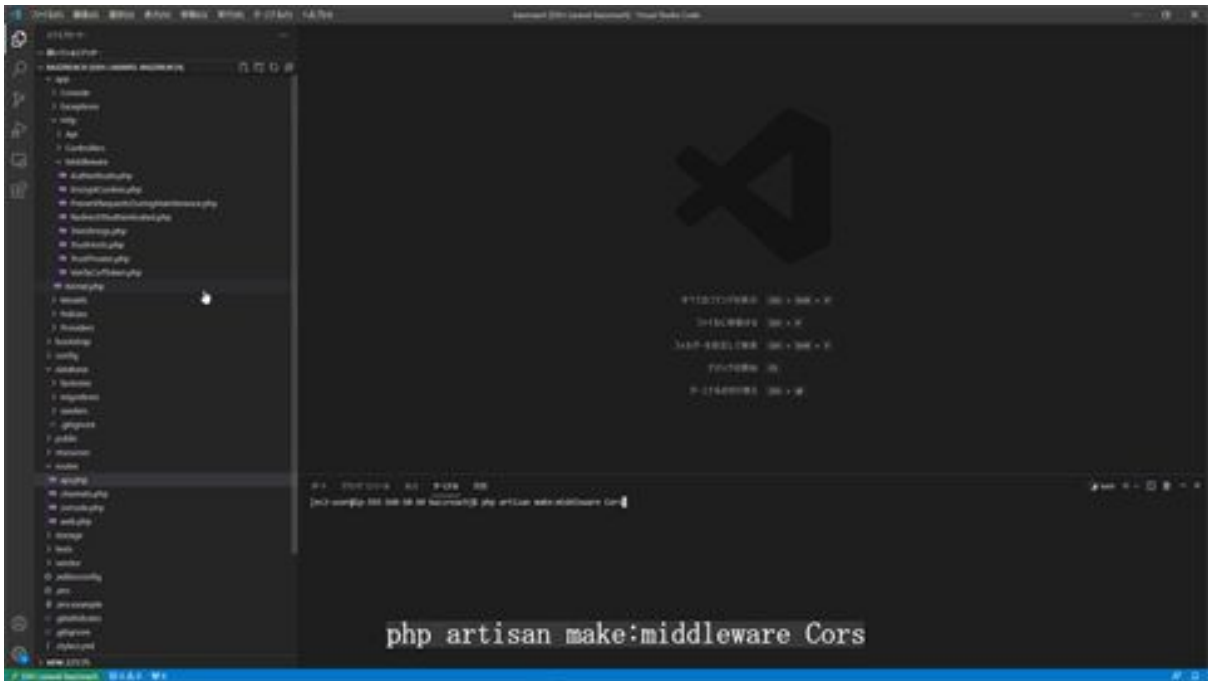
Let's configure Laravel routing



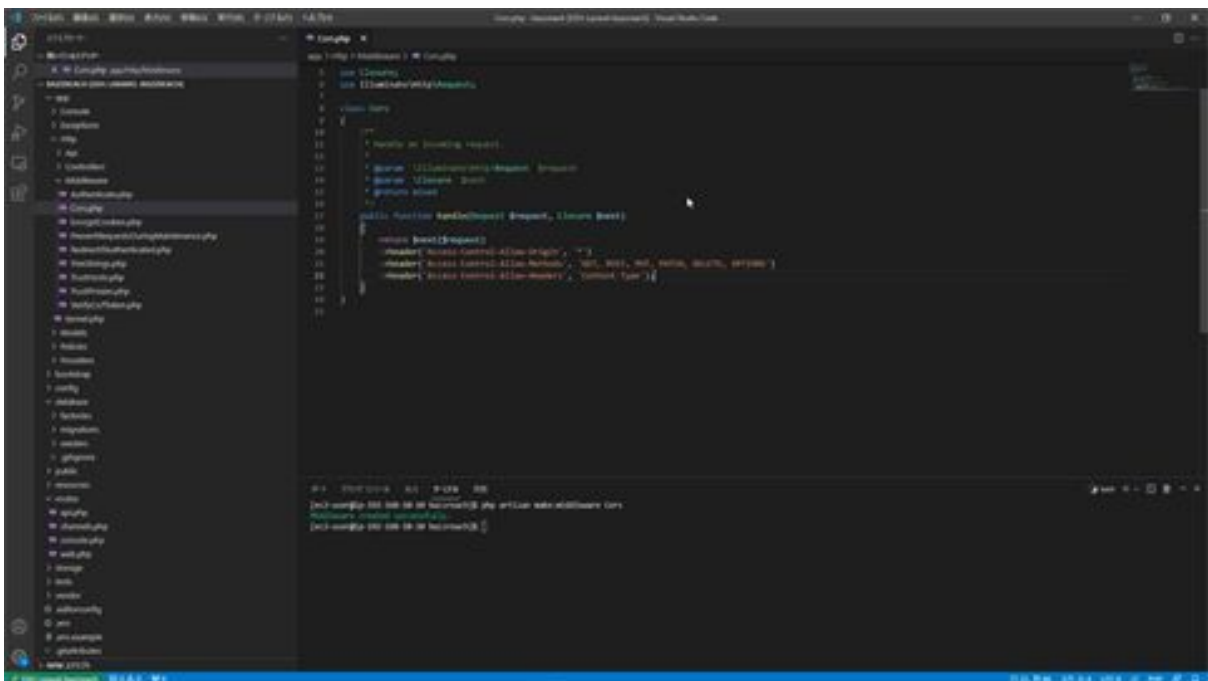
Next, we will configure Laravel's routing settings. What this does is create a mapping rule between URLs and controller functions. By creating a mapping rule, we can determine the rule that this controller's process will be executed when accessed by this URL. Now let's do the work.

First, configure Laravel to allow API connections in visual studio code. Usually, if the URL scheme, HTTP or HTTPS is the same, and the domain and port number are the same, the connection can be made as the same origin. However, since we are connecting from Wix instead of from the AWS Laravel screen, we need to set up CORS so that we can connect even if they are not from the same origin. To do this, we create middleware.

Create the middleware by running `php artisan make:middleware Cors` in a terminal.

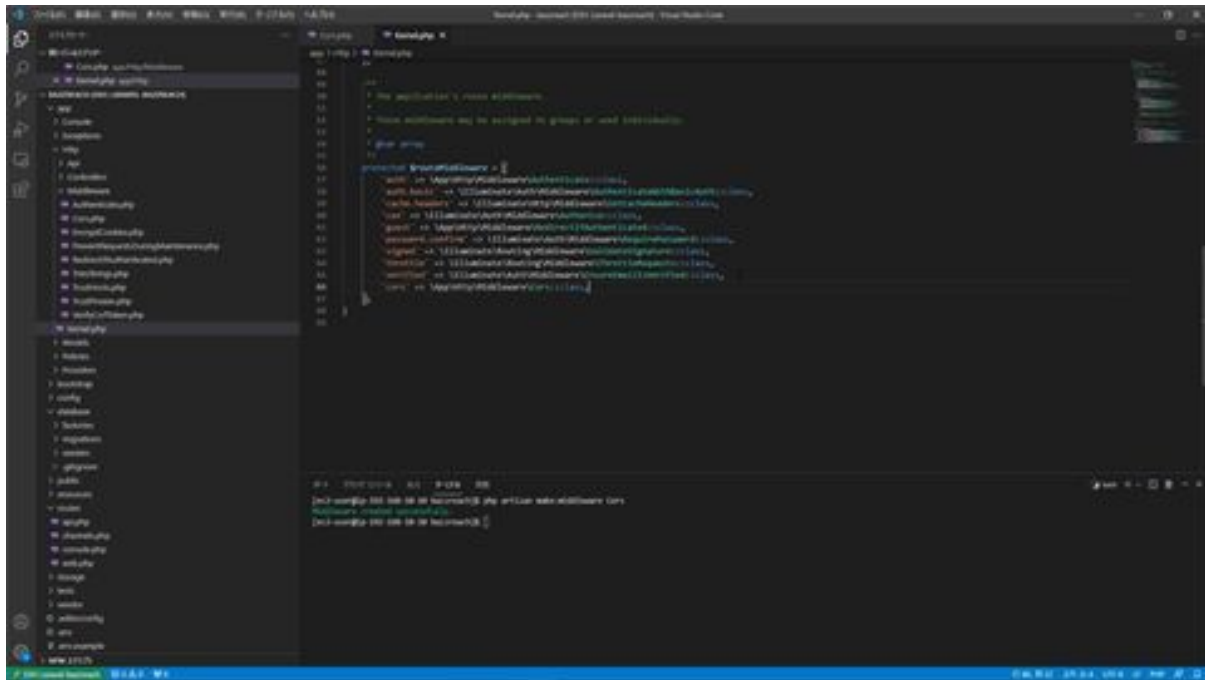


Next, modify the middleware you have created by opening and modifying `Cors.php` in the `app/Http/Middleware` folder.

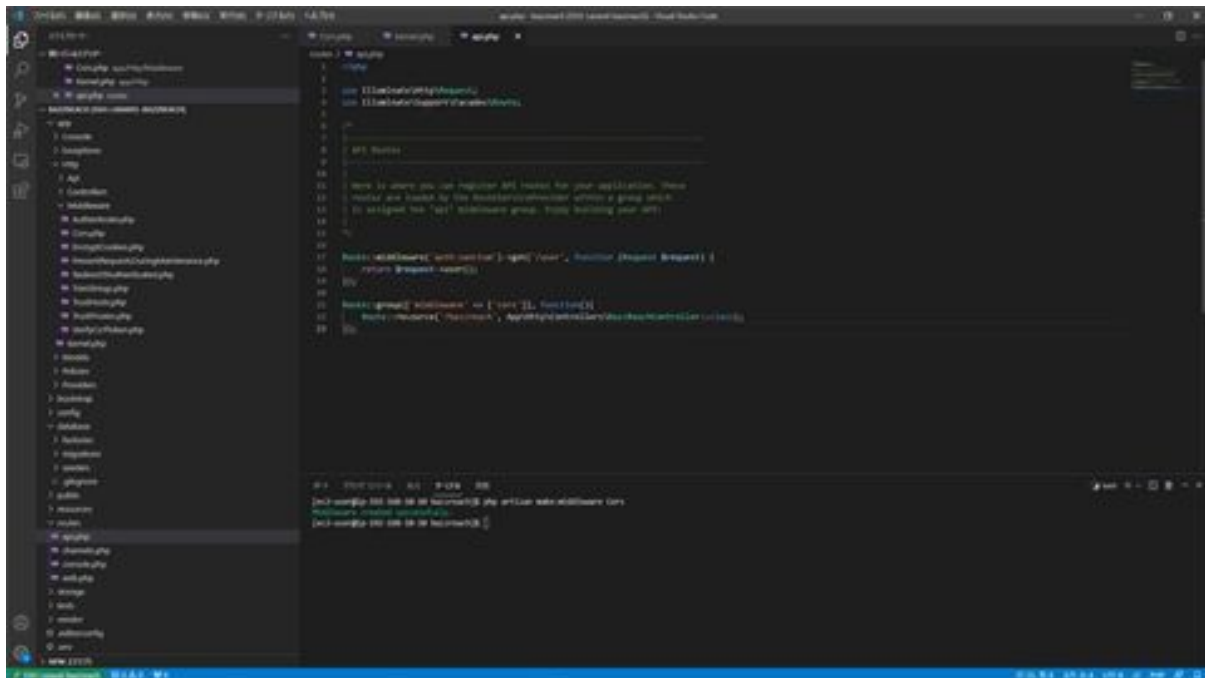


Once the modification is complete, the next step is to add the created middleware to the kernel: open `kernel.php` in the `app/Http` folder and add

the classpath for cors to \$routeMiddleware.



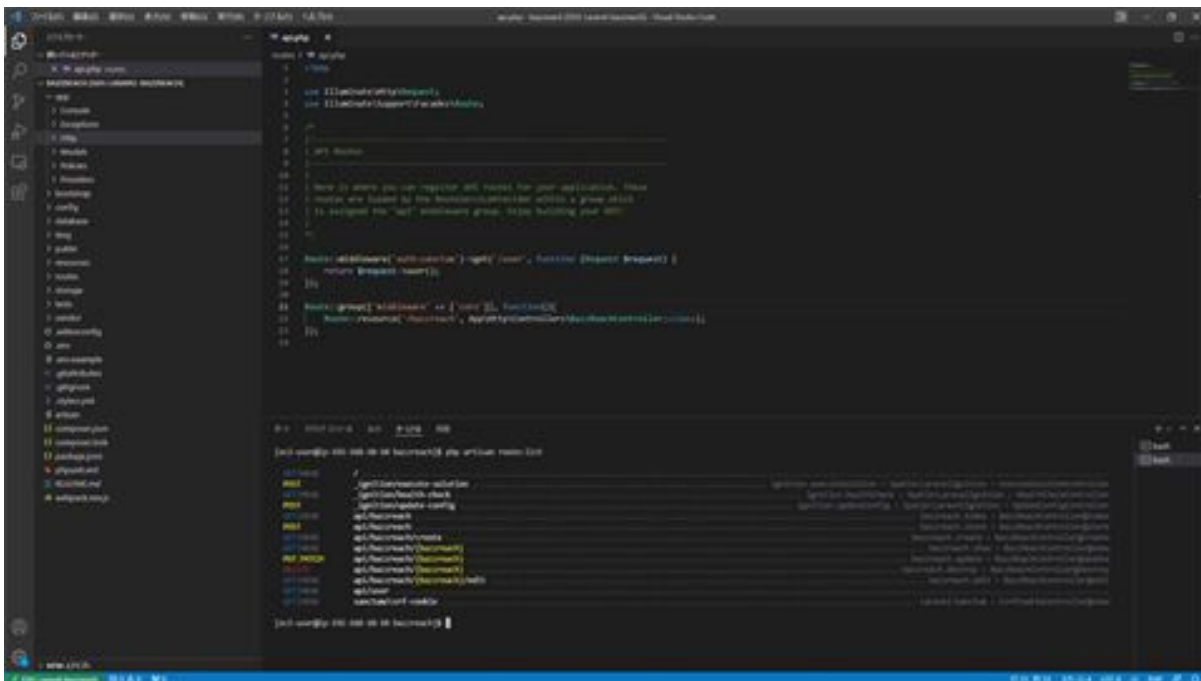
Next, open `api.php` in the `routes` folder and configure the routing settings.



This is a grouping of routing, but this time it is a middleware grouping, and if you write routing settings in this description, the middleware is always

used. Middleware is a function that checks HTTP requests or responses. In this case, Kolus is configured as a middleware so that connections can be made even if they do not originate from the same origin. Routing is configured in this middleware using the resource. By setting up the resource in this way, routing for the initial display screen, registration screen, registration process, update screen, update process, detailed display, and deletion process can be created all at once. Let's see how the routing is created.

Run `php artisan route:list` in a terminal to see the routing correspondence table.

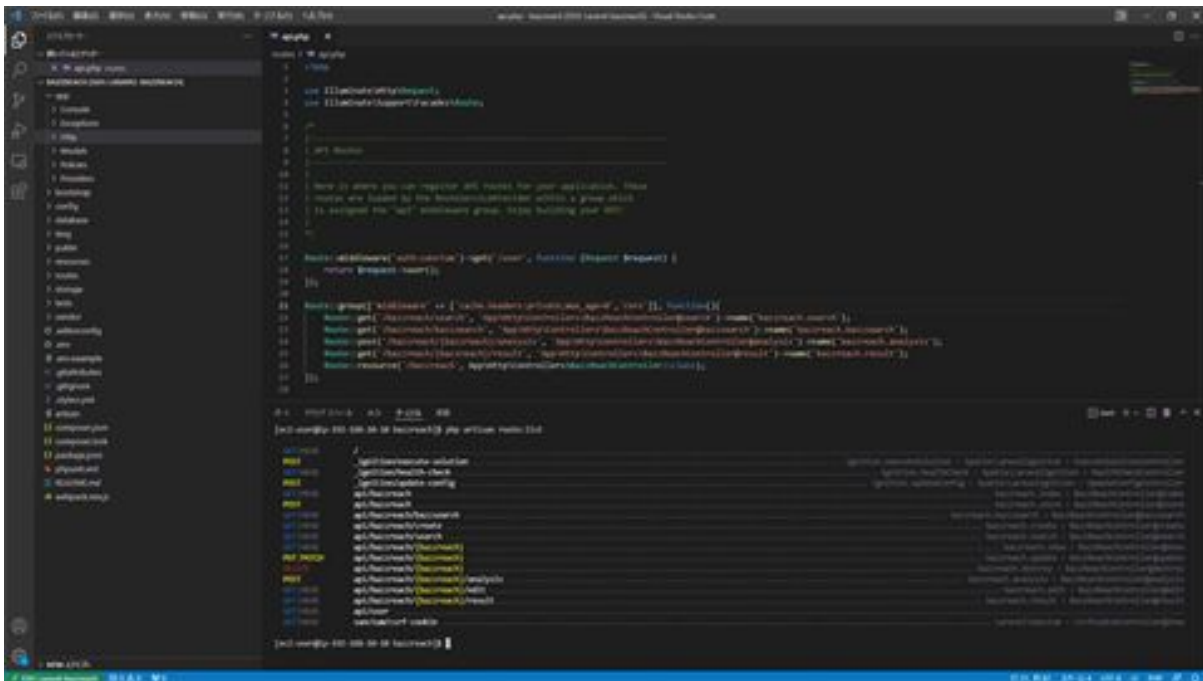


In the correspondence table displayed here, for example, if the URI is `api/bazzreach/create`, it is set to process the `create` action method in the `BazzReachController`. Here is the same thing as the URI.

Looking at `api/bazzreach`, the URI part is the same, and the action method processing is divided into `index` and `store`. The reason why the same URI exists is that the action method to be processed can be changed depending

bazzreach.result will be the routing to retrieve the analysis results. Each of these will be explained in detail when adding the process.

Next, the middleware SetCacheHeaders is used to change the response to not use caching. This is done by setting the API response cache settings to private to disallow caching on the cache server. In addition, max_age is set to 0 seconds to ensure that the cache is always up-to-date. Finally, we check the routing.



Now that we have verified that the routing is configured correctly, we are done with Laravel's routing configuration.

Cors.php

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use Closure;
use Illuminate\Http\Request;

class Cors
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
    public function handle(Request $request, Closure $next)
    {
        return $next($request)
            ->header('Access-Control-Allow-Origin', '*')
            ->header('Access-Control-Allow-Methods', 'GET, POST, PUT,
PATCH, DELETE, OPTIONS')
            ->header('Access-Control-Allow-Headers', 'Content-Type');
    }
}
```

Kernel.php

```
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
```

```

{
  /**
   * The application's global HTTP middleware stack.
   *
   * These middleware are run during every request to your
  application.
   *
   * @var array
   */
  protected $middleware = [
    // \App\Http\Middleware\TrustHosts::class,
    \App\Http\Middleware\TrustProxies::class,
    \Fruitcake\Cors\HandleCors ::class,

    \App\Http\Middleware\PreventRequestsDuringMaintenance::class,

    \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
    \App\Http\Middleware\TrimStrings::class,

    \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsTo
    Null::class,
  ];

  /**
   * The application's route middleware groups.
   *
   * @var array
   */
  protected $middlewareGroups = [
    'web' => [
      \App\Http\Middleware\EncryptCookies::class,

```



```

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::c
lass,
    \Illuminate\Session\Middleware\StartSession::class,
    //
\Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,
    \App\Http\Middleware\VerifyCsrfToken::class,
    \Illuminate\Routing\Middleware\SubstituteBindings::class,
],
'api' => [
    //
\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreSt
ateful::class,
    'throttle:api',
    \Illuminate\Routing\Middleware\SubstituteBindings::class,
],
];

/**
 * The application's route middleware.
 *
 * These middleware may be assigned to groups or used
individually.
 *
 * @var array
 */
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,

```

```

        'auth.basic' =>
\Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'cache.headers' =>
\Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' =>
\App\Http\Middleware\RedirectIfAuthenticated::class,
        'password.confirm' =>
\Illuminate\Auth\Middleware\RequirePassword::class,
        'signed' =>
\Illuminate\Routing\Middleware\ValidateSignature::class,
        'throttle' =>
\Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' =>
\Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
        'cors' => \App\Http\Middleware\Cors::class,
    ];
}

```

api.php

```
<?php
```

```
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
```

```
/*
```

```

|-----|
| API Routes
|-----|
|

```

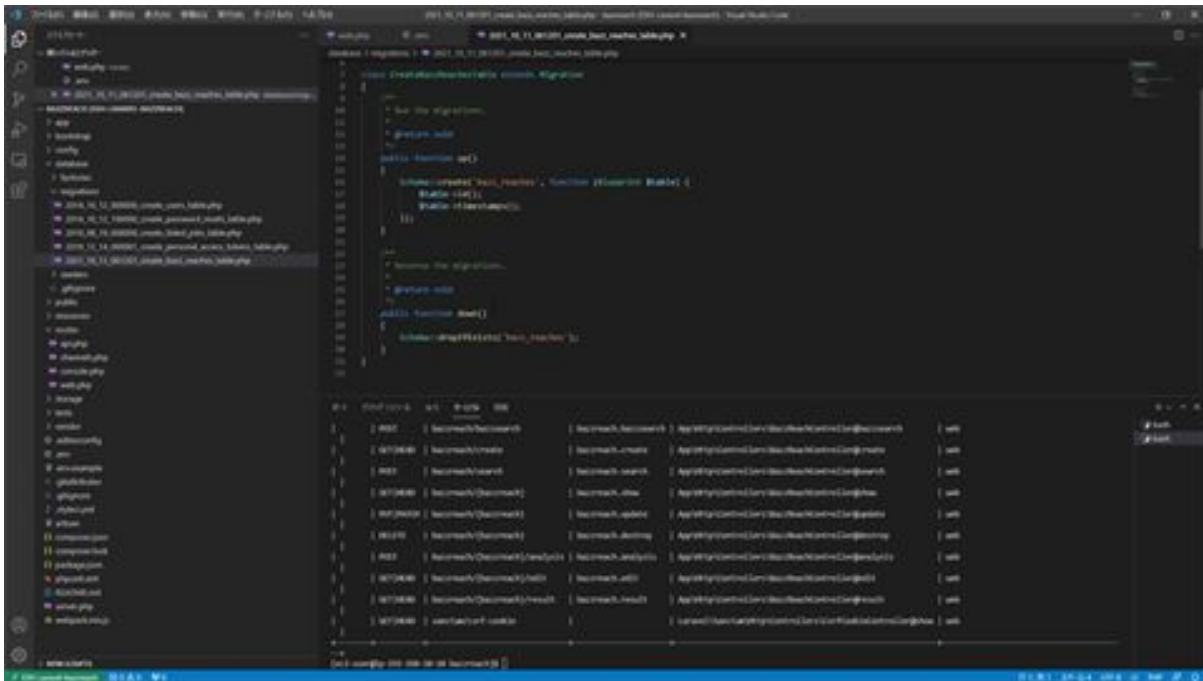
| Here is where you can register API routes for your application.
These
| routes are loaded by the RouteServiceProvider within a group
which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

```
Route::middleware('auth:sanctum')->get('/user', function (Request  
$request) {  
    return $request->user();  
});
```

```
Route::group(['middleware' =>  
['cache.headers:private;max_age=0','cors']], function() {  
    Route::get('/ bazzreach /search',  
'App\Http\Controllers\BazzReachController@search')->  
>name('bazzreach.search');  
    Route::get('/ bazzreach / bazzsearch ',  
'App\Http\Controllers\BazzReachController@bazzsearch')->  
>name('bazzreach.bazzsearch');  
    Route::post('/ bazzreach /{ bazzreach }/analysis',  
'App\Http\Controllers\BazzReachController@analysis')->  
>name('bazzreach.analysis');  
    Route::get('/ bazzreach /{ bazzreach }/result',  
'App\Http\Controllers\BazzReachController@result')->  
>name('bazzreach.result');  
    Route::resource('/ bazzreach ',  
App\Http\Controllers\BazzReachController::class);  
});
```

Let's perform migration

Now we will perform the migration. This migration will create tables in the RDS database. First, we want to change the configuration of the BazzReaches table, so open the file create_date+create_bazz_reaches_table.php in the migrations folder in the database folder.



```
function create($connection)
{
    // Create the migration.
    @mysqli_select_db($connection, $database);

    $sql = "CREATE TABLE `bazz_reaches` (
        `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
        `timestamp` VARCHAR(255) NOT NULL,
        `search_word` VARCHAR(255) NOT NULL,
        `total_tweets` INT(11) NOT NULL,
        `comment` VARCHAR(255) NOT NULL,
        PRIMARY KEY (`id`)
    ) ENGINE=InnoDB";

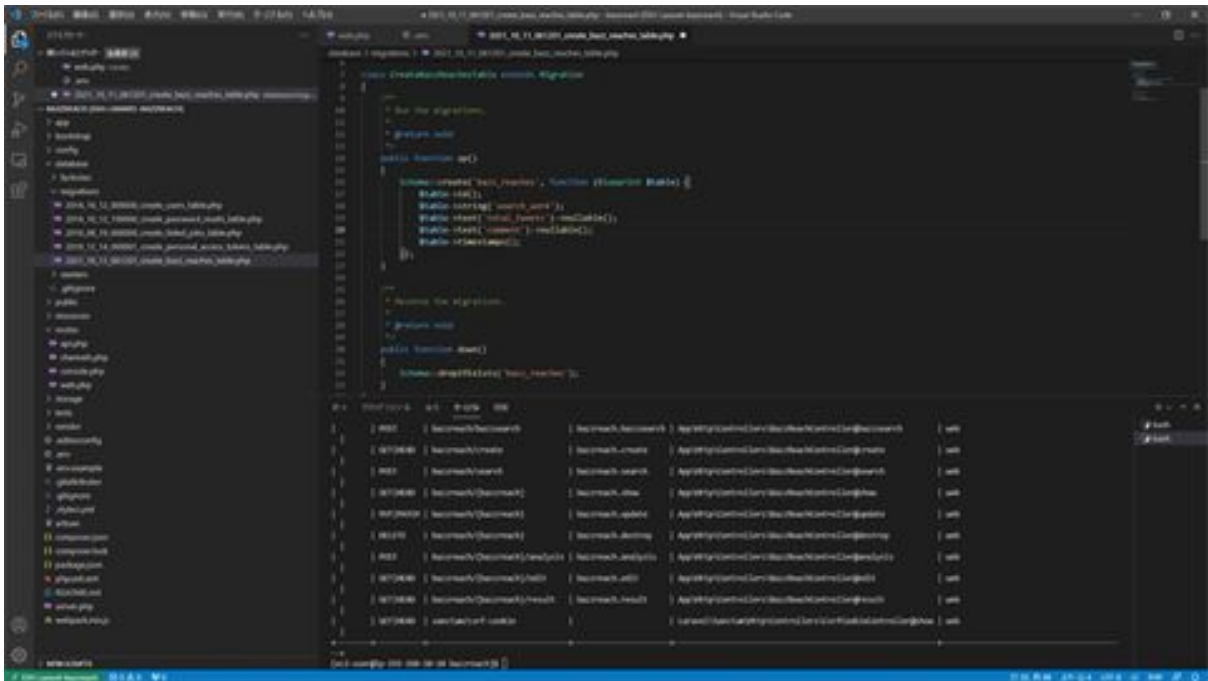
    $connection->query($sql);
}

function down($connection)
{
    // Reverse the migration.
    @mysqli_select_db($connection, $database);

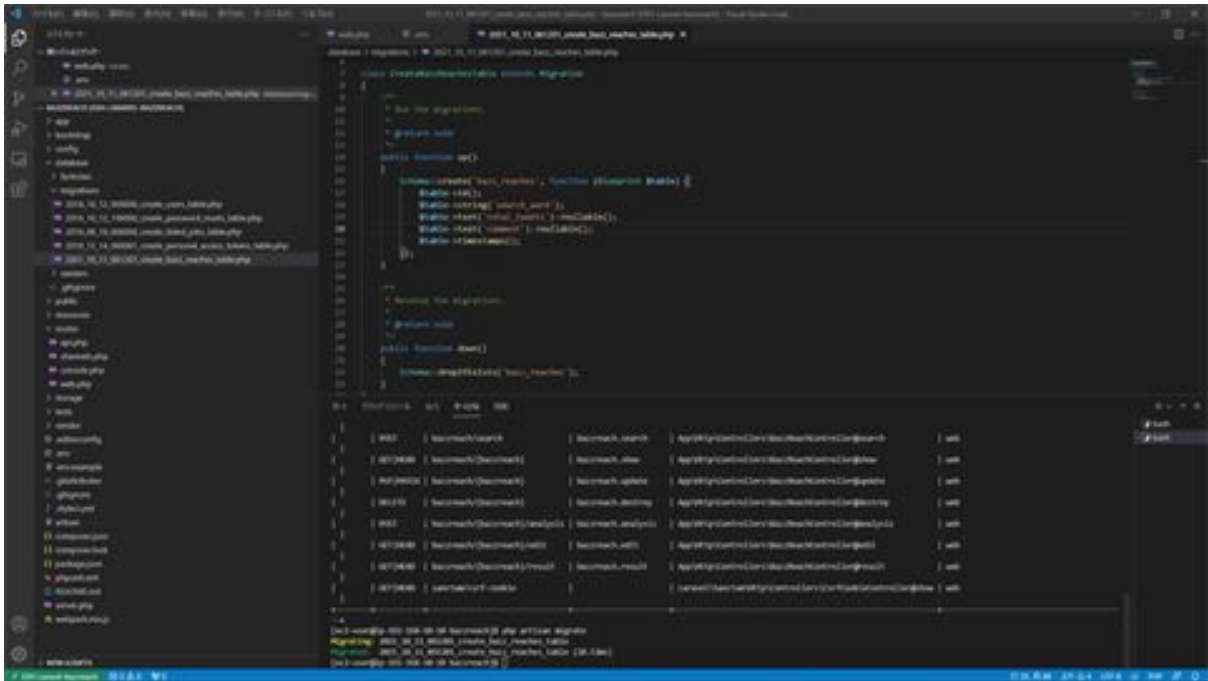
    $sql = "DROP TABLE `bazz_reaches`";

    $connection->query($sql);
}
```

Next, add the items you want to add between the id and timestamp listed in the create section. search_word is the item to store the search keyword, total_tweets is the item to store the searched tweets, the comment is the item to store the comment you entered, and so on. This field is for storing comments.



Once you have made these modifications, perform the migration. Enter the command `php artisan migrate` in the terminal to run the migration.



Migration is now complete. If you made a mistake and want to start from the beginning, you can roll back all migrations and run the migration again

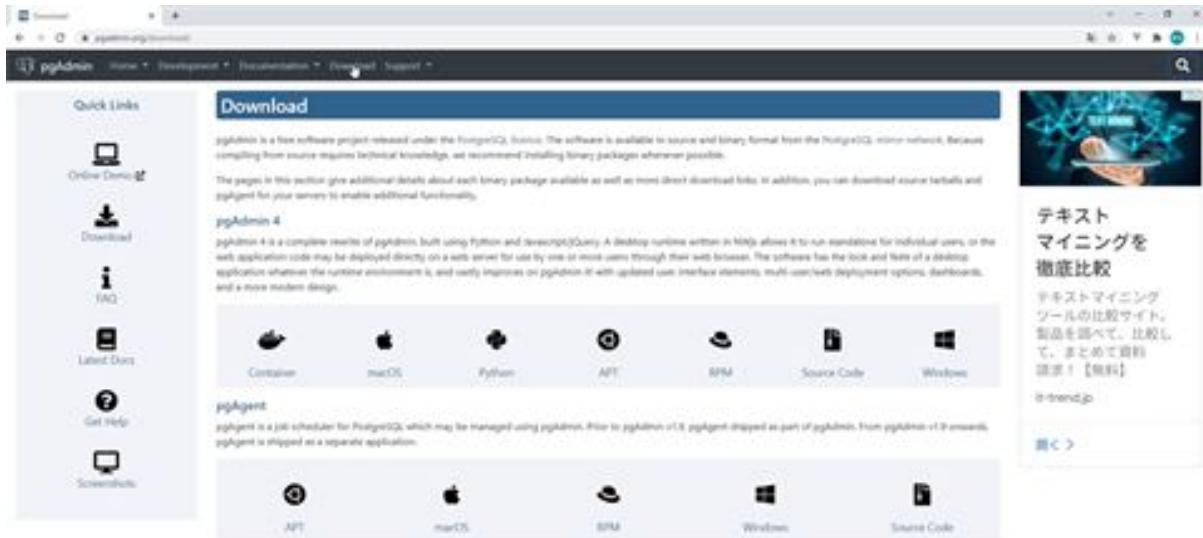
by typing the command `php artisan migrate:refresh`. This completes the migration process.

Connect to RDS with PGAdmin

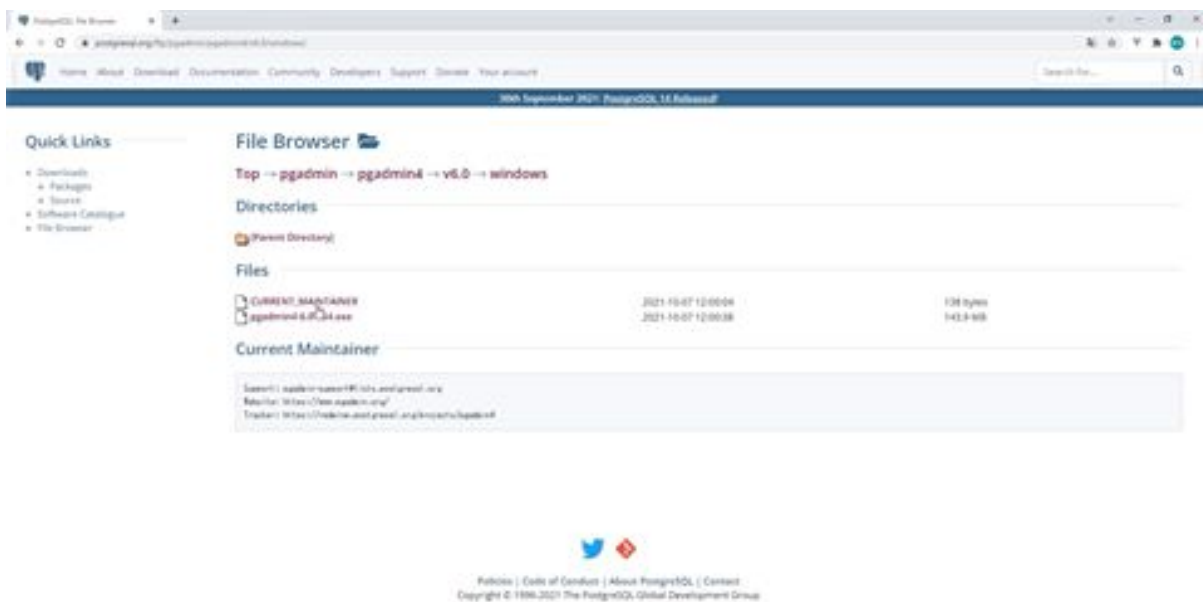
Now, let's connect to RDS with PGAdmin, a tool that allows you to control the database on EC2 using PGAdmin since it is difficult to work with a command-based system. First, access the PGAdmin site by searching for PGAdmin on Google.



Next, install PGAdmin. Click on Downloads in the menu above and click on the button for your platform from PGAdmin.

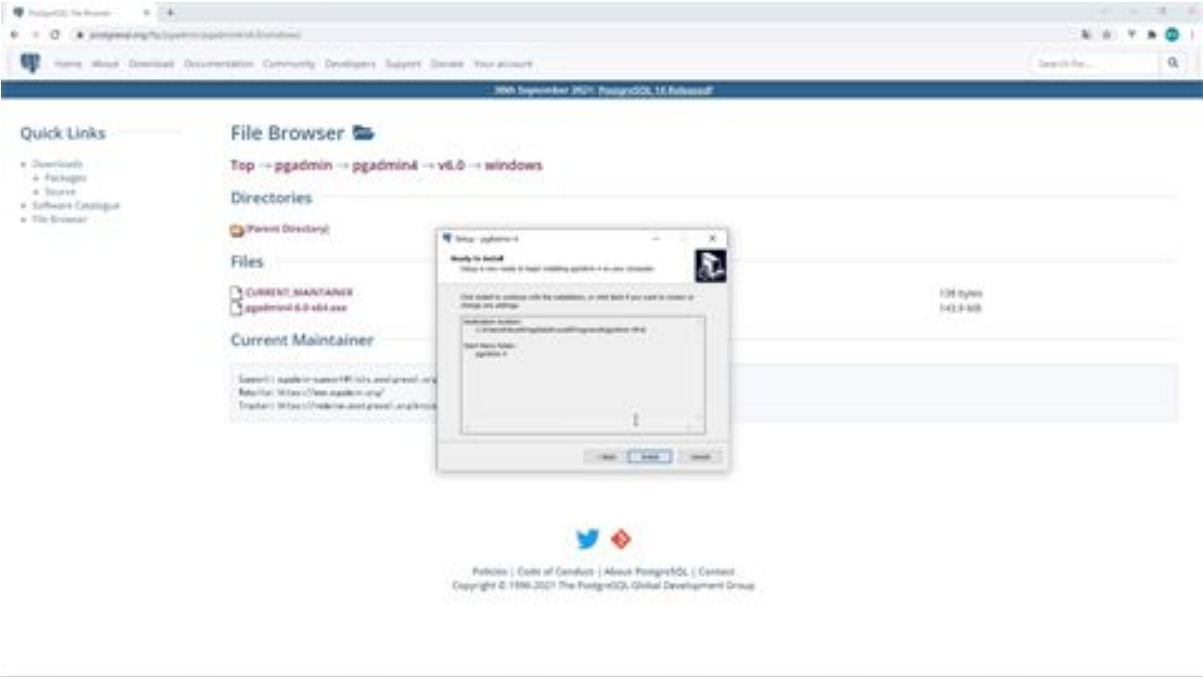


Then the download screen will appear and click on the latest version of PGAdmin. On the next screen, you will see a link to the executable file.

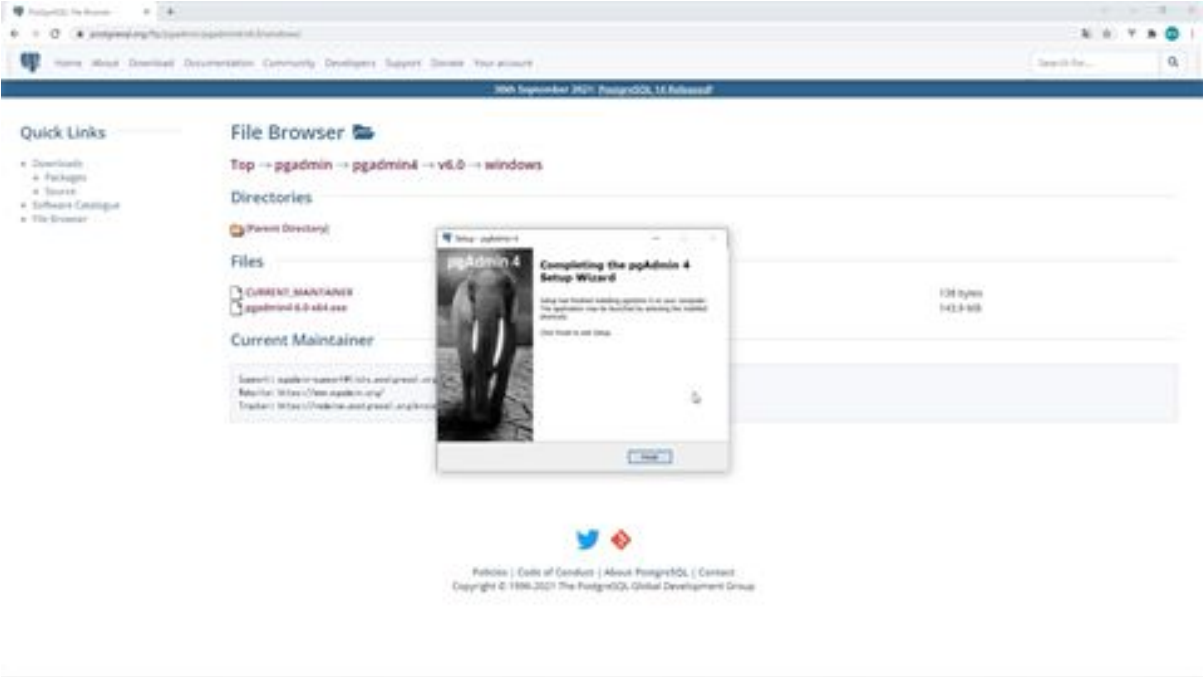


After the download is complete, run the installer to install PGAdmin. Once launched, click the NEXT button, select "I also agree to the license" and click the Next button. Click the Next button again on the next screen, and

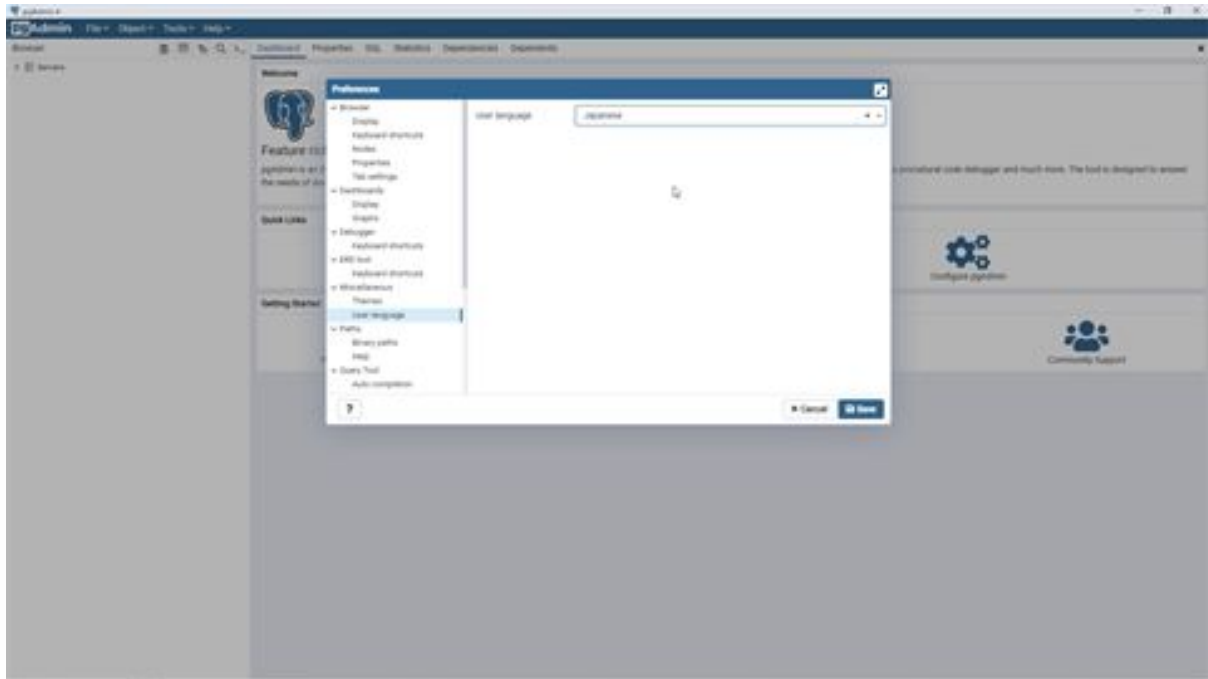
then click the Next button on the screen after that. Click the Install button on the next screen.



The installation will then begin. When the installation is complete, press the Finish button to start PGAdmin.



Once launched, the first step is to translate PGAdmin into Japanese: click on Preferences in the File menu, select Japanese from User Language, and click the Save button.



Then restart PGAdmin and you will see the Japanese display. Next, we will connect to RDS. Click on Servers on the left, click on Create from Object in the menu above, click on Server Group, and enter a name on the Create Server Group screen. Here we will enter BazzServers.



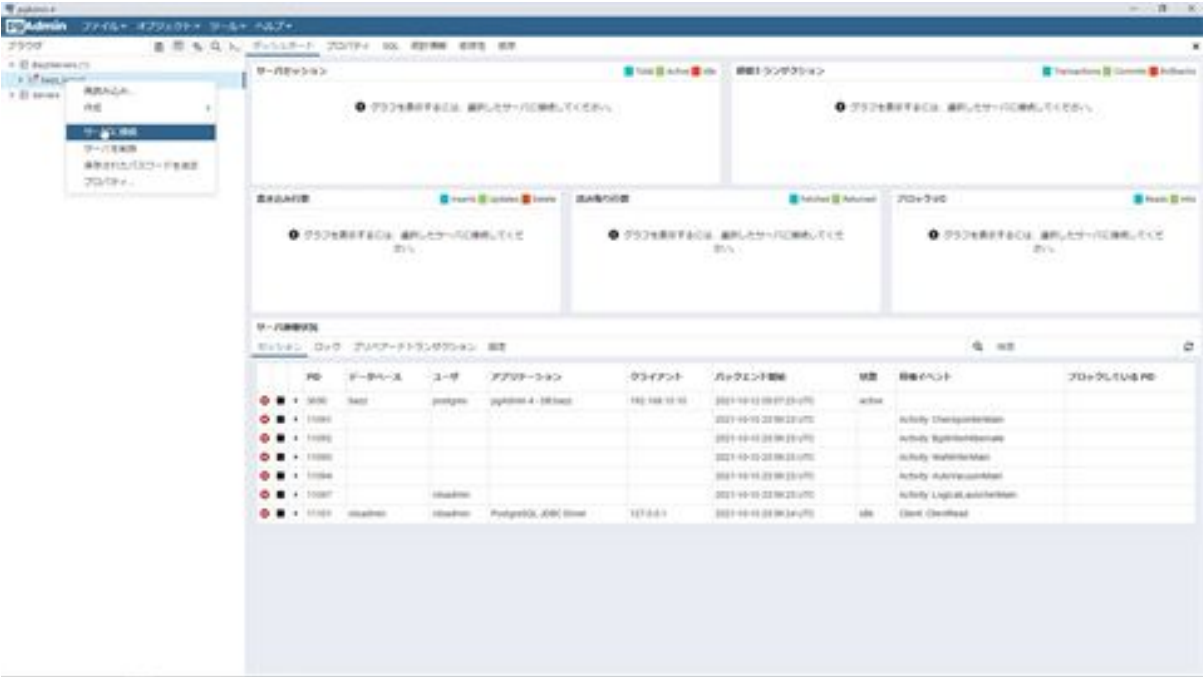
Next, click on Create Object in the top menu and then on Servers. Then the server configuration screen will appear and you will configure the settings. First, the name on the General tab will be the server name displayed in the browser on the left, so enter a name of your choice. In this case, enter `bazz_Laravel`. Then select BazzServers in the server group.



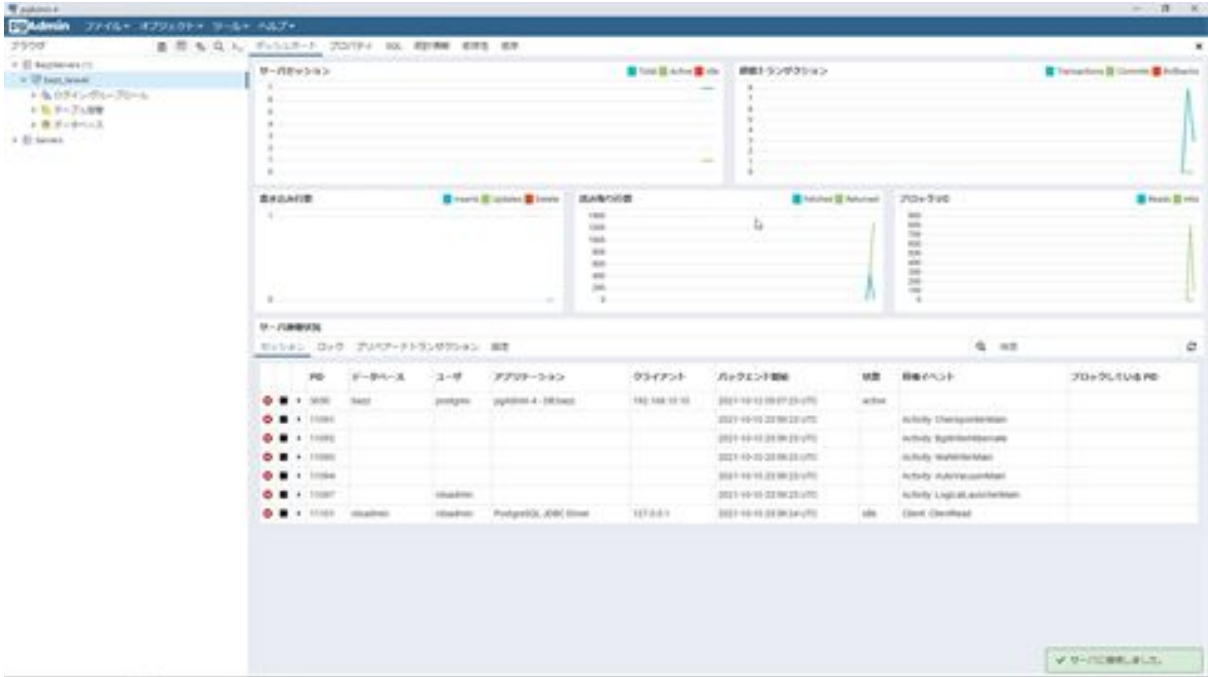
authentication information file. Click the Save button when you have entered the information up to this point.



Next, confirm the connection. Right-click on the server you created and click Connect to Server.



You will then be prompted to enter the password for the identification file, but click the OK button without entering anything. Then you can connect to RDS from PGAdmin.

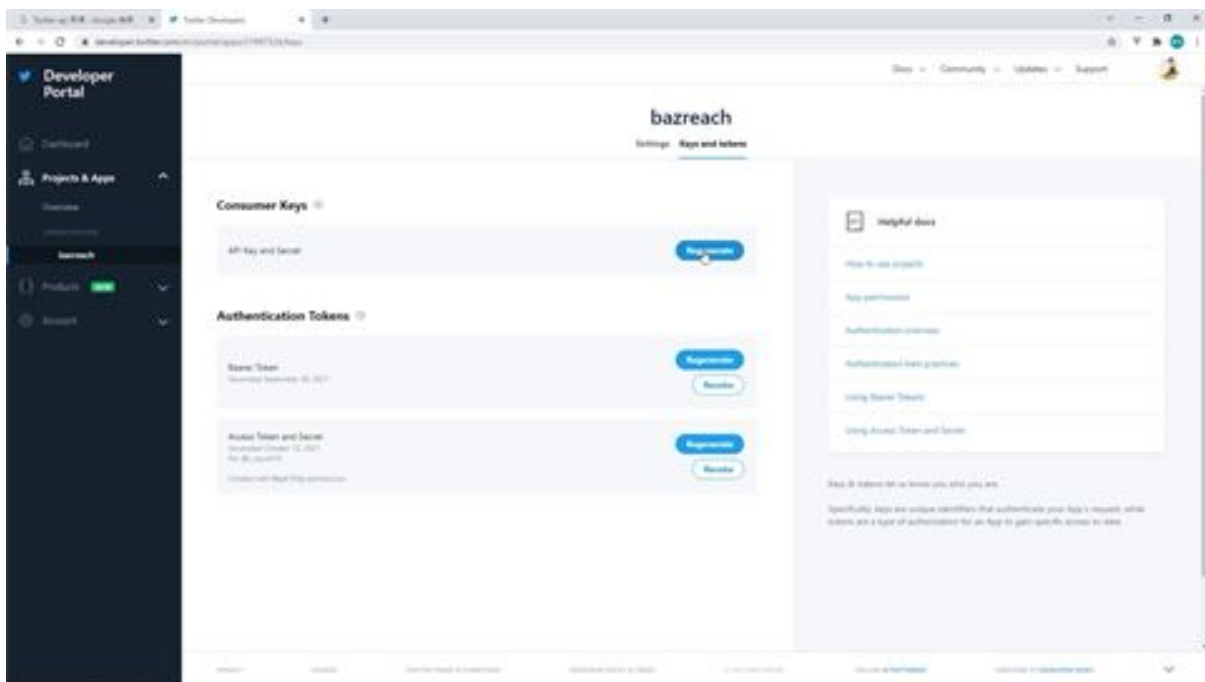


By registering data from here, the application will be able to retrieve data. This is the end of the connection from PGAdmin to RDS.

Let's create a tweet search function with TweeterAPI

Now we want to create a tweet search function with TwitterAPI, but first, we need to apply for TwitterAPI." If you search Google for "Twitter api acquisition", you will see how to acquire TwitterAPI, so please acquire TwitterAPI KEY by yourself.

You will need `api_key`, `api_key_secret`, `access_token`, and `access_token_secret`. Once obtained, `api_key``api_key_secret``access_token`, and `access_token_secret` can be obtained by clicking the Generate button from the screen.



After obtaining the TwitterAPI KEY, install the `abraham/twitteroauth` library to call the Twitter api. Go to the `bazzreach` folder in the terminal and type `composer require abraham/twitteroauth` to run the installation.

Next, create a method to perform a Twitter search: create an Api folder in the Http folder under the app folder in the bazzreach folder and create a file called TwitterApi.php. Once created, we will write the process in TwitterApi.

```
app > http > api > TwitterApi.php
1 </?php
2
3 namespace App\Http\Api;
4
5 use Abraham\TwitterOAuth\TwitterOAuth;
6
7 class TwitterApi
8 {
9
10     private $connection ;
11
12     public function __construct()
13     {
14         $this->connection = new TwitterOAuth(
15             env('TWITTER_CLIENT_KEY'),
16             env('TWITTER_CLIENT_SECRET'),
17             env('TWITTER_CLIENT_ID_ACCESS_TOKEN'),
18             env('TWITTER_CLIENT_ID_ACCESS_TOKEN_SECRET'));
19     }
20
21     // ツイート検索
22     public function serachTweets(String $searchWord)
23     {
24         $totalTweets = "";
25         $searchResults = $this->connection ->get("search/tweets", [
26             'q' => $searchWord,
27             'count' => 100,
28         ]);
29     }
30 }
```

The first part of the process is written as follows: use Abraham\TwitterOAuth\TwitterOAuth;}

This is a declaration to use the library TwitterOAuth that has just been registered. Next, in this __construct section, the initial settings for the search are made, and the TweeterApi information set in the env file is acquired to retrieve the TweeterApi information. Each key corresponds to a key in the env file.

```
38
39 AWS_ACCESS_KEY_ID=
40 AWS_SECRET_ACCESS_KEY=
41 AWS_DEFAULT_REGION=us-east-1
42 AWS_BUCKET=
43 AWS_USE_PATH_STYLE_ENDPOINT=false
44
45 PUSHER_APP_ID=
46 PUSHER_APP_KEY=
47 PUSHER_APP_SECRET=
48 PUSHER_APP_CLUSTER=mt1
49
50 MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
51 MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
52
53 TWITTER_CLIENT_KEY = jk0nuE20P0bBT10mb9Xsy6
54 TWITTER_CLIENT_SECRET = crt7P1YBFzmcXet5DaS3zIf07zDqWuXl91DyVQSEJ31r3Wf5L
55 TWITTER_CLIENT_ID_ACCESS_TOKEN = 1336095832047722496-4T1hffCEvCcJKSp03l1c05InC1Fhcl
56 TWITTER_CLIENT_ID_ACCESS_TOKEN_SECRET = 6oWvqk7RZ3ye0VY9oqoCusyWuTFU60KTMgJ0Nlyw17
```

Next, the serachTweets method is described.

```
app > http > api > TwitterApp.php
14 $this->connection = new TwitterOAuth(
15     env('TWITTER_CLIENT_KEY'),
16     env('TWITTER_CLIENT_SECRET'),
17     env('TWITTER_CLIENT_ID_ACCESS_TOKEN'),
18     env('TWITTER_CLIENT_ID_ACCESS_TOKEN_SECRET'));
19
20
21 // コメント検索
22 public function serachTweets(String $searchWord)
23 {
24     $totalTweets = "";
25     $searchResults = $this->connection->get("search/tweets", [
26         'q' => $searchWord,
27         'count' => 100,
28     ]);
29
30     $searchResults = json_decode(json_encode($searchResults->statuses), true);
31
32     foreach ($searchResults as $searchResult) {
33         $totalTweets .= $searchResult["text"];
34     }
35
36     $totalTweets = mb_strcut($totalTweets, 0, 5000, "UTF-8");
37     return $totalTweets;
38 }
39 }
```

The serachTweets function for retrieving tweets is used to retrieve 100 tweets for the search keyword. The search keyword is retrieved from the \$searchWord argument that came as a keyword, and 100 tweets are

retrieved by searching Twitter with the search keyword. Since it is difficult to use the data as it is, it is converted to JSON format before retrieving the tweets, and all the retrieved tweets are combined.

After that, tweets are retrieved so that they are less than 5,000 bytes. The reason for the 5000 bytes is that if the number of tweets exceeds 5000 bytes, an error will occur in the API that performs the analysis. The `mb_strcut` function is used to make sure that the size of the tweets is less than 5000 bytes.

The final step is to return the tweet. This is all there is to creating a tweet search function with `TweeterApi`.

TwitterApi.php

```
<?php

namespace App\Http\Api;

use Abraham\TwitterOAuth\TwitterOAuth;

class TwitterApi
{

    private $connection ;

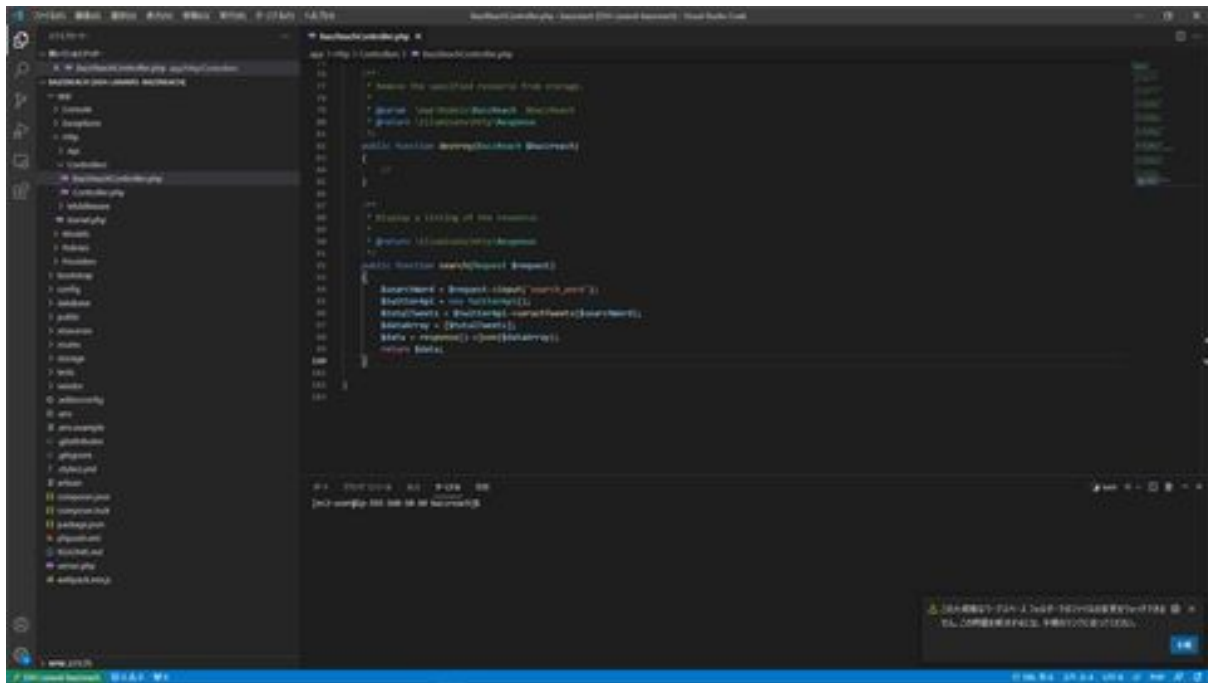
    public function __construct()
    {
        $this->connection = new TwitterOAuth(
            env('TWITTER_CLIENT_KEY'),
            env('TWITTER_CLIENT_SECRET'),
            env('TWITTER_CLIENT_ID_ACCESS_TOKEN'),
            env('TWITTER_CLIENT_ID_ACCESS_TOKEN_SECRET'));
    }

    // ツイート検索
    public function serachTweets(String $searchWord)
    {
        $totalTweets = "";
        $searchResults = $this->connection ->get("search/tweets", [
            'q' => $searchWord,
            'count' => 100,
```

```
]);  
  
$searchResults = json_decode(json_encode($searchResults->statuses),  
true);  
  
foreach ($searchResults as $searchResult) {  
    $totalTweets .= $searchResult["text"];  
}  
  
$totalTweets = mb_strcut($totalTweets, 0 , 5000, "UTF-8");  
return $totalTweets;  
}  
}
```

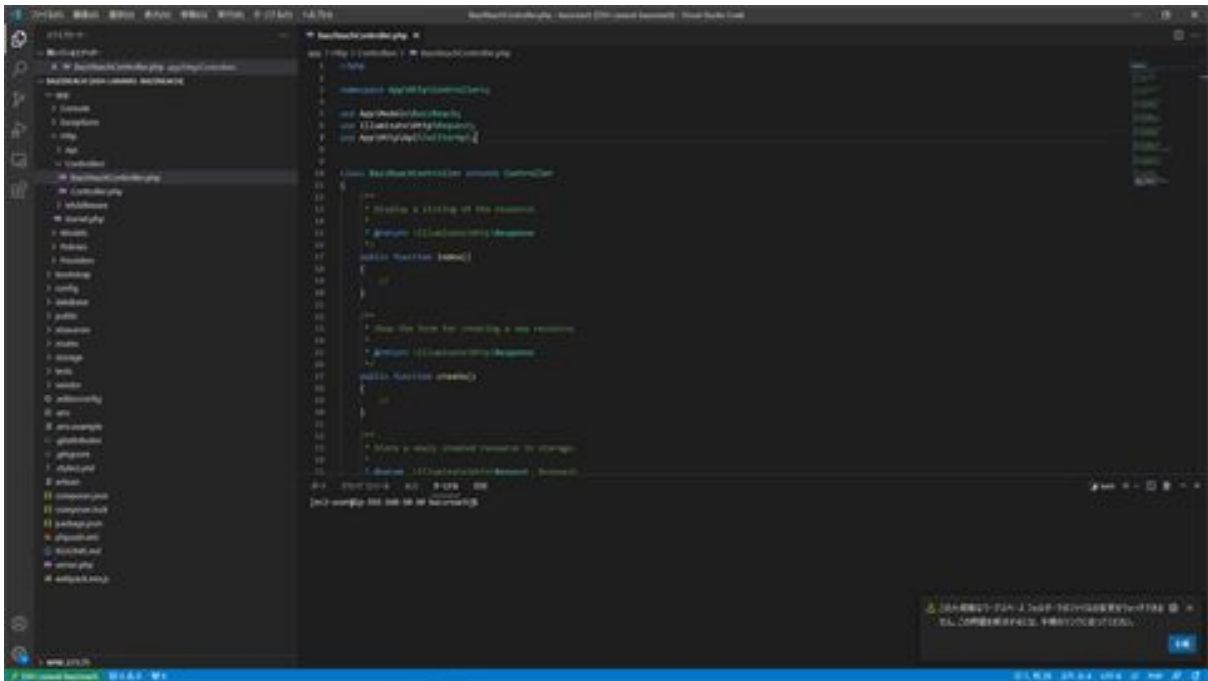

Let's show search tweets

Now we will display the searched tweets on the screen. First, open `BazzReachController` and add the search method.

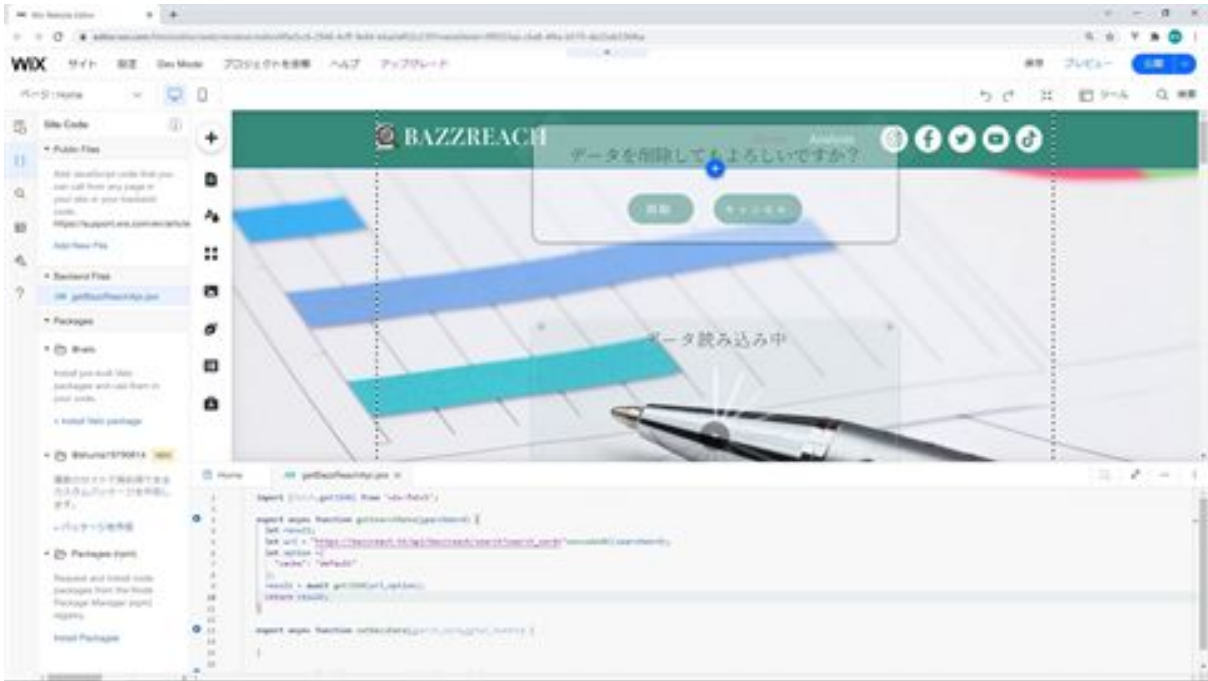


```
1 // Retrieve the searched tweets from storage
2
3 @RequestMapping(value = "/search")
4 @ResponseBody
5 public class BazzReachController {
6
7     // Retrieve a listing of the tweets
8     @RequestMapping(value = "/tweets")
9     public List<Tweet> getTweets() {
10         // Retrieve the tweets from the database
11         return tweetsRepository.findAll();
12     }
13
14     // Retrieve the searched tweets from storage
15     @RequestMapping(value = "/search/{search_word}")
16     @ResponseBody
17     public List<Tweet> searchTweets(@PathVariable("search_word") String search_word) {
18         // Retrieve the searched tweets from storage
19         List<Tweet> tweets = tweetsRepository.findAll();
20         List<Tweet> search_tweets = new ArrayList<>();
21         for (Tweet tweet : tweets) {
22             if (tweet.getText().contains(search_word)) {
23                 search_tweets.add(tweet);
24             }
25         }
26         return search_tweets;
27     }
28 }
```

This method explains that it retrieves the `search_word` data passed from the screen from `$request`. Next, create an instance of the `TwitterApi` you created, retrieve tweets from the instance using the `search_word`, and pass all tweet data in JSON format to the screen side. Next, import `TwitterApi` so that it can be used.



Once this has been done, the next step is to modify it so that it can be displayed on the screen side.



Now we will modify the screen side. First, we will add the `getSearchData` function processing to the `getBazzReachApi.jsw` file. This function explains

that it accesses the search method created in Laravel to retrieve data.

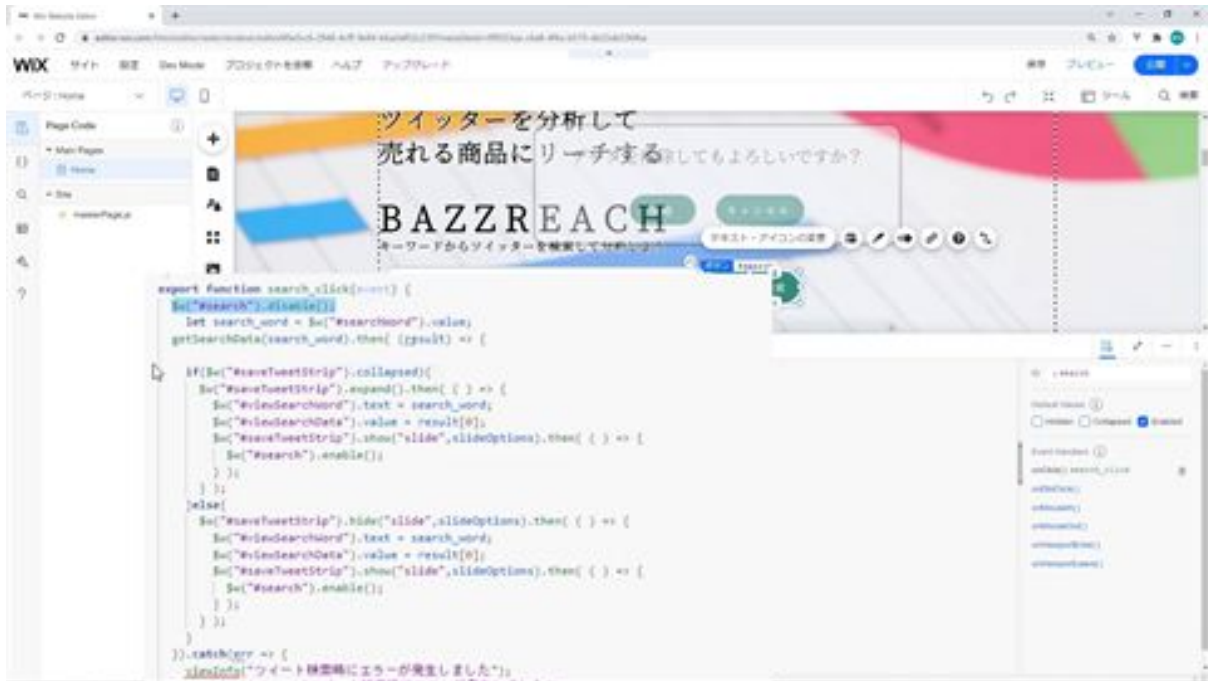
The search_word after the URL is passed as a URL parameter is used in the Get submission method so that it can be received on the Laravel side. The encodeURIComponent function used here is a function to handle Japanese as a URL parameter.

option sets the cache settings, and the request cache is set as the default setting in the cache options settings.

Finally, getJSON is used to retrieve the JSON data and return it to the caller. Here, there is a description of async before function and await before getJson. What this means is that async is placed before the function to create an async function. This function is used in asynchronous communication when using a process that returns an object called Promise.

In this case, the getJson function is used to return a Promise object via asynchronous communication. Also, "await" is a reserved word that can be used only in the async function to make the process wait until the Promise returns the result. In other words, the await and async statements are written to wait until the result is returned in asynchronous processing.

The getJson and fetch functions for API communication are asynchronous communication, which means that the next process is executed even if the result is not returned. Conversely, waiting until the result is returned is called synchronous processing. In Wix, the getBazzReachApi module on the backend side performs API communication and the results are passed to the frontend side home, so this async and await are used for synchronous processing of getJson. Next, on the Home side, we add processing to the onClick event function of the tweet search button.



To explain this process, first, deactivate the search button whose ID is searched. If the button is clicked several times on the Wix side and the same process is performed, a communication error may occur.

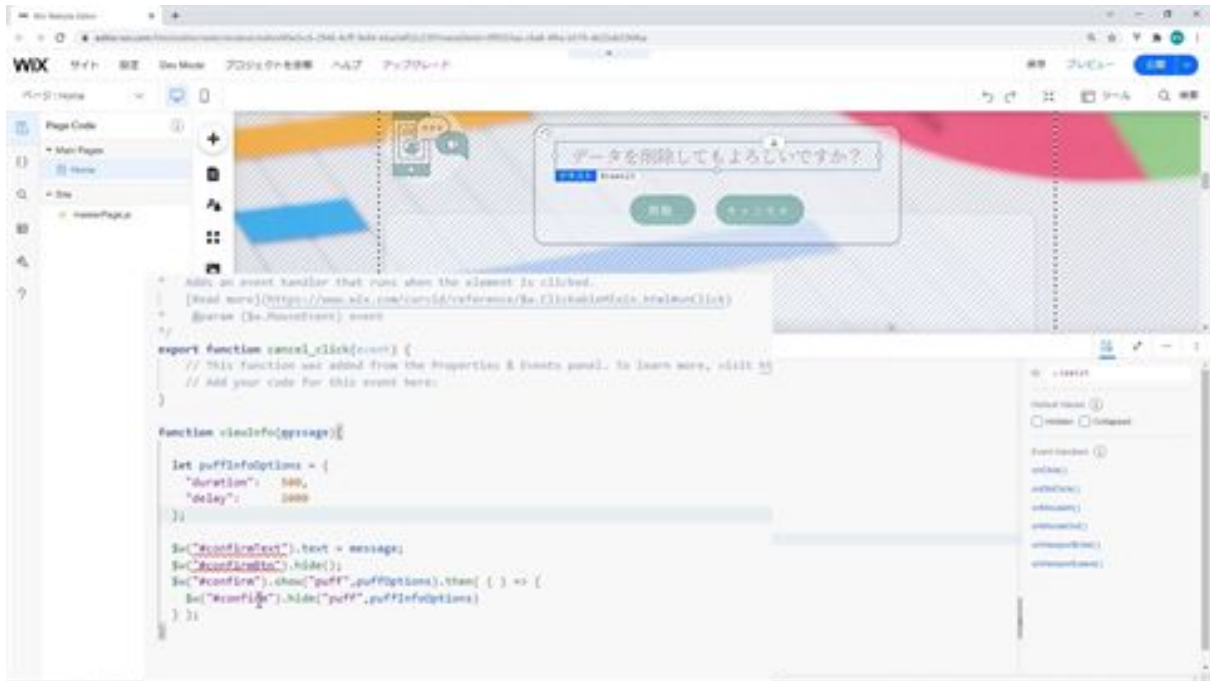
Next, get the tweet keyword from the text field whose ID is searchWord and call the getSearchData function you created. When the resulting tweet data is returned, it is set to the result variable, from which data with an array index of 0 is retrieved.

Data with an array index of 0 is retrieved because it is set and returned in an array. Next, if the strip with ID saveTweetStrip is collapsed, after setting the search keyword in the text with ID viewSearchWord and the acquired tweet data in the textbox with ID viewSearchData, the animation is used to saveTweetStrip and make the search button active.

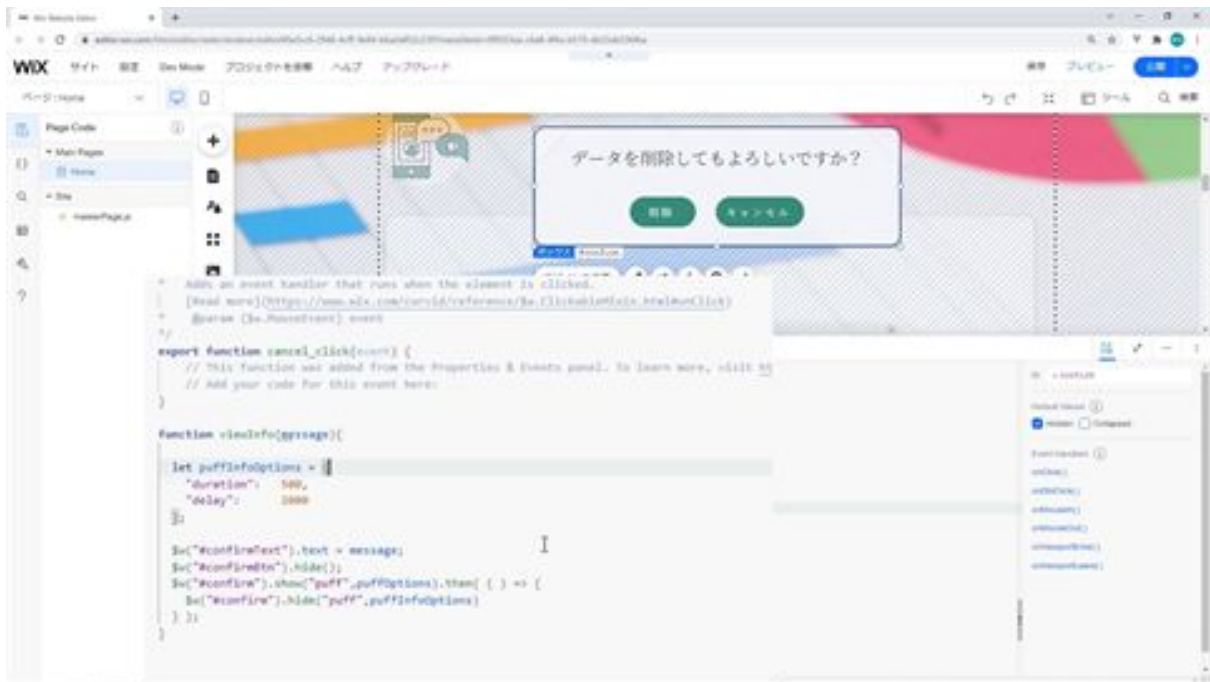


If it is displayed without collapsing, the animation is hidden once and then the search keyword is set in the text whose ID is viewSearchWord and the retrieved tweet data is set in the text box whose ID is viewSearchData, then the animation is displayed and the search button is activated.

If there is an error in data acquisition, the viewInfo function is called to display the error message and log it to the console. Next, the viewInfo function is added.



Add IDs because IDs do not exist in two places here. confirmText is set to the ID of the text in the confirmation dialog. Next is confirmBtn, add the ID for the part that groups the two buttons of the confirmation dialog.





Once you have made this modification, we will check it on the screen. From the screen, enter the tweet keyword and click the Tweet Search button.



Then the results of the search by tweet are displayed on the screen. The search tweets are displayed above.

BazzReachController.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Models\BazzReach;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Api\TwitterApi;
```

```
class BazzReachController extends Controller
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function index()
```

```
    {
```

```
        //
```

```
    }
```

```
    /**
```

```
     * Show the form for creating a new resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function create()
```

```
    {
```

```
        //
```

```

}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    //
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\BazzReach $bazzReach
 * @return \Illuminate\Http\Response
 */
public function show( BazzReach $bazzReach )
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param \App\Models\BazzReach $bazzReach
 * @return \Illuminate\Http\Response
 */
public function edit( BazzReach $bazzReach )
{
    //
}

```

```

}

/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, BazzReach $ bazzReach )
{
    //
}

/**
 * Remove the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function destroy( BazzReach $ bazzReach )
{
    //
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function search(Request $request)
{
    $searchWord = $request->input('search_word');

```

```
$twitterApi = new TwitterApi();  
$totalTweets = $twitterApi->searchTweets($searchWord);  
$dataArray = [$totalTweets];  
$data = response()->json($dataArray);  
return $data;  
}  
}
```

getBazzReachApi.jsw

```
import { fetch,getJSON} from 'wix-fetch';

export async function getSearchData(searchWord) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/search?
search_word="+encodeURIComponent(searchWord);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setBazzData (search_word,total_tweets) {

}

export async function getBazzSearchData (keyword) {

}

export async function setAnalysis(id) {

}

export async function getBazzAnalysisData(id) {

}

export async function setMemo(id,comment) {
```

```
}
```

```
export async function delBazzData(id) {
```

```
}
```

home.js

```
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};

let rollOptions1 = {
  "duration": 1500,
  "delay": 0,
  "direction": "left"
```

```

};

let rollOptions2 = {
  "duration": 1000,
  "delay": 0,
  "direction": "left"
};

let fadeOptions = {
  "duration": 500,
  "delay": 0
};

$w.onReady(function () {

});

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  getSearchData(search_word).then( (result) => {

    if($w("#saveTweetStrip").collapsed){
      $w("#saveTweetStrip").expand().then( ( ) => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( ( ) => {
          $w("#search").enable();

```



```

    } );
  } );
} else {
  $w("#saveTweetStrip").hide("slide", slideOptions).then( () => {
    $w("#viewSearchWord").text = search_word;
    $w("#viewSearchData").value = result[0];
    $w("#saveTweetStrip").show("slide", slideOptions).then( () => {
      $w("#search").enable();
    } );
  } );
}
}).catch(err => {
  viewInfo("An error occurred when searching for tweets");
  console.log(err, "An error occurred when searching for tweets");
  $w("#search").enable();
});
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveTweet_click(event) {
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event

```

```
*/
export function bazzSearch_click(event) {
}

/**
 *   Adds an event handler that runs when a table row is selected.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.Table.html#onRowSelect)
 *   @param {$w.TableRowEvent} event
 */
export function bazzSearchTable_rowSelect(event) {
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function viewAnalysis_click(event) {
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function viewMemoInput_click(event) {
}
```

```
/**
 * Adds an event handler that runs when the element is clicked.
 [Read more]
 (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function saveMemo_click(event) {
```

```
}
```

```
/**
 * Adds an event handler that runs when the element is clicked.
 [Read more]
 (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function deleteConfirm_click(event) {
```

```
}
```

```
/**
 * Adds an event handler that runs when the element is clicked.
 [Read more]
 (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function cancel_click(event) {
```

```
}
```

```
/**
 * Adds an event handler that runs when the element is clicked.
 [Read more]
 (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
```

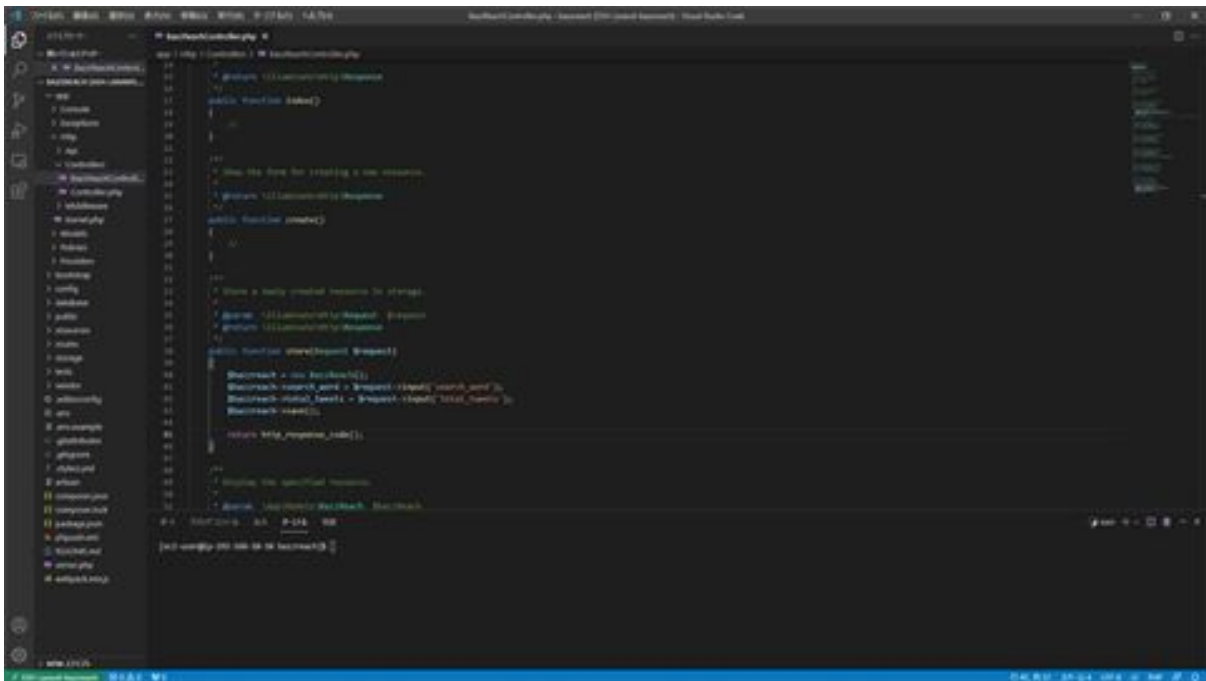
```
*      @param {$w.MouseEvent} event
*/
export function delete_click(event) {
}

function viewInfo(message){
  let puffInfoOptions = {
    "duration": 500,
    "delay": 2000
  };

  $w("#confirmText").text = message;
  $w("#confirmBtn").hide();
  $w("#confirm").show("puff",puffOptions).then( () => {
    $w("#confirm").hide("puff",puffInfoOptions)
  } );
}
```

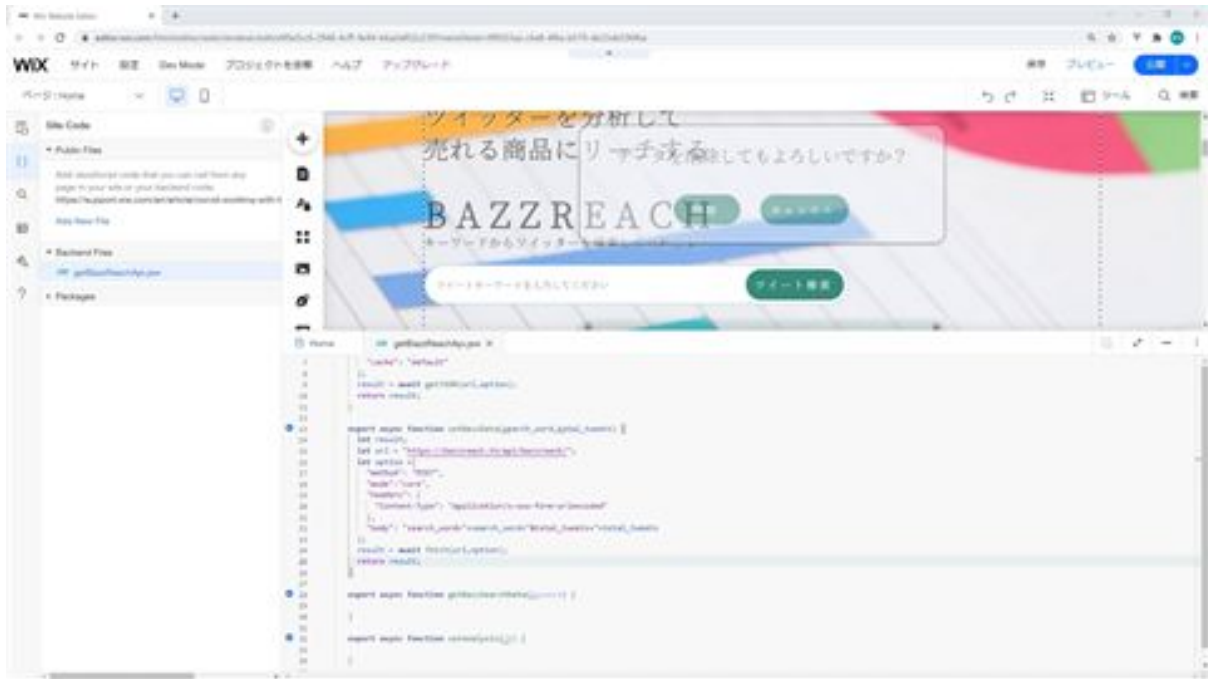
Let's register your search tweets in the DB

Now let's add the process of registering the search tweets to the DB: open BazzReachController and add the store method process.



```
211 public function store()
212 {
213     // Show the form for creating a new resource.
214     // @param \Illuminate\Http\Request $request
215     return $this->show($request);
216 }
217
218 /**
219  * Store a newly created resource in storage.
220  * @param \Illuminate\Http\Request $request
221  * @return \Illuminate\Http\Response
222  */
223 public function store(Request $request)
224 {
225     //
226
227     $bazzreach = new BazzReach();
228     $bazzreach->search_word = $request->input('search_word');
229     $bazzreach->total_tweets = $request->input('total_tweets');
230     $bazzreach->save();
231
232     return $this->response_code();
233 }
234
235 /**
236  * Display the specified resource.
237  * @param \Illuminate\Http\Request $request
238  * @return \Illuminate\Http\Response
239  */
240 public function show($request)
```

This method explains that it retrieves the search_word and total_tweets data passed from the screen from \$request and sets them to the search_word and total_tweets items of the BazzReach model. Once set, the data is registered in the bazz_reaches table in the BazzReach model instance. Finally, the HTTP response code is returned with the return. Now that this has been done, the next step is to modify the data so that it can be displayed on the screen side.



First, add processing to the setBazzData function in the getBazzReachApi Web module. To explain the process of this function, we access the store method created in Laravel and register data bypassing the search_word and total_tweets. tweets=" part, parameters are passed and POST is specified in the method so that the Laravel side can receive them as a POST transmission method.

In the headers section, the content type is specified as the content type for form submission. Finally, POST is sent using fetch, and cross-origin requests are allowed by specifying core in mode to allow access to the API on the Laravel side.

Next, on the Home side, add processing to the onClick event function of the Save Search Tweets button.



To explain this process, first, the save button of a search tweet whose ID is saveTweet is deactivated to prevent double submissions. Next, call the setBazzData function you created. After the call, the data is registered, so we perform an animation to hide the strip with the ID saveTweetStrip, initialize the displayed tweet keywords and tweet data, collapse the strip with the ID saveTweetStrip, and then fold the save button for search tweets. Activate.

If an error occurs in saving the data, the viewInfo function is called to display the error message and log it to the console. Finally, it activates the Save Search Tweets button.

The bazzSearch_click event function is called after calling setBazzData, and it is assumed that the registered data will be read again. We will add more processing to this bazzSearch_click in the future. Now let's check if the data can be registered.

in total_tweet. This is the end of the DB registration of search tweets.

BazzReachController.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Models\BazzReach;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Api\TwitterApi;
```

```
class BazzReachController extends Controller
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function index()
```

```
    {
```

```
        //
```

```
    }
```

```
    /**
```

```
     * Show the form for creating a new resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function create()
```

```
    {
```

```
        //
```

```

}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $ bazzreach = new BazzReach ();
    $ bazzreach->search_word = $request->input('search_word');
    $ bazzreach->total_tweets = $request->input('total_tweets');
    $ bazzreach->save();

    return http_response_code();
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function show( BazzReach $ bazzReach )
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach

```

```

* @return \Illuminate\Http\Response
*/
public function edit( BazzReach $ bazzReach )
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, BazzReach $ bazzReach )
{
    //
}

/**
 * Remove the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function destroy( BazzReach $ bazzReach )
{
    //
}

/**
 * Display a listing of the resource.
 *

```

```
* @return \Illuminate\Http\Response
*/
public function search(Request $request)
{
    $searchWord = $request->input('search_word');
    $twitterApi = new TwitterApi();
    $totalTweets = $twitterApi->searchTweets($searchWord);
    $dataArray = [$totalTweets];
    $data = response()->json($dataArray);
    return $data;
}
}
```

getBazzReachApi.jsw

```
import {fetch,getJSON} from 'wix-fetch';

export async function getSearchData(searchWord) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/search?
search_word="+encodeURIComponent(searchWord);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setBazzData (search_word,total_tweets) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/";
  let option ={
    "method": "POST",
    "mode":"core",
    "headers": {
      "Content-Type": "application/x-www-form-urlencoded"
    },
    "body": "search_word="+search_word+"&total_tweets="+total_tweets
  };
  result = await fetch(url,option);
  return result;
}
```

```
export async function getBazzSearchData (keyword) {  
  }  
export async function setAnalysis(id) {  
  }  
export async function getBazzAnalysisData(id) {  
  }  
export async function setMemo(id,comment) {  
  }  
export async function delBazzData(id) {  
  }
```

home.js

```
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};

let rollOptions1 = {
  "duration": 1500,
  "delay": 0,
  "direction": "left"
```



```

};

let rollOptions2 = {
  "duration": 1000,
  "delay": 0,
  "direction": "left"
};

let fadeOptions = {
  "duration": 500,
  "delay": 0
};

$w.onReady(function () {

});

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  getSearchData(search_word).then( (result) => {

    if($w("#saveTweetStrip").collapsed){
      $w("#saveTweetStrip").expand().then( ( ) => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( ( ) => {
          $w("#search").enable();

```

```

    } );
  } );
} else {
  $w("#saveTweetStrip").hide("slide", slideOptions).then( () => {
    $w("#viewSearchWord").text = search_word;
    $w("#viewSearchData").value = result[0];
    $w("#saveTweetStrip").show("slide", slideOptions).then( () => {
      $w("#search").enable();
    } );
  } );
}
}).catch(err => {
  viewInfo("An error occurred when searching for tweets");
  console.log(err, "An error occurred when searching for tweets");
  $w("#search").enable();
});
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveTweet_click(event) {
  $w("#saveTweet").disable();

  setBazzData($w("#viewSearchWord").text, $w("#viewSearchData").value).t
  hen( result ) => {
    bazzSearch_click(event);
    viewInfo("Saved tweet");
    $w("#saveTweetStrip").hide("slide", slideOptions).then( () => {

```

```

    $w("#viewSearchWord").text = "";
    $w("#viewSearchData").value = "";
    $w("#saveTweetStrip").collapse();
    $w("#saveTweet").enable();
  });
}).catch(err => {
  viewInfo("An error occurred when saving the tweet");
  console.log(err, "An error occurred when saving the tweet");
  $w("#saveTweet").enable();
});
}

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function bazzSearch_click(event) {
}

/**
 * Adds an event handler that runs when a table row is selected.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.Table.html#onRowSelect)
 * @param {$w.TableRowEvent} event
 */
export function bazzSearchTable_rowSelect(event) {
}

/**
 * Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
*/
export function viewAnalysis_click(event) {
}
```

/**

* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
*/
export function viewMemoInput_click(event) {
}
```

/**

* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
*/
export function saveMemo_click(event) {
}
```

/**

* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

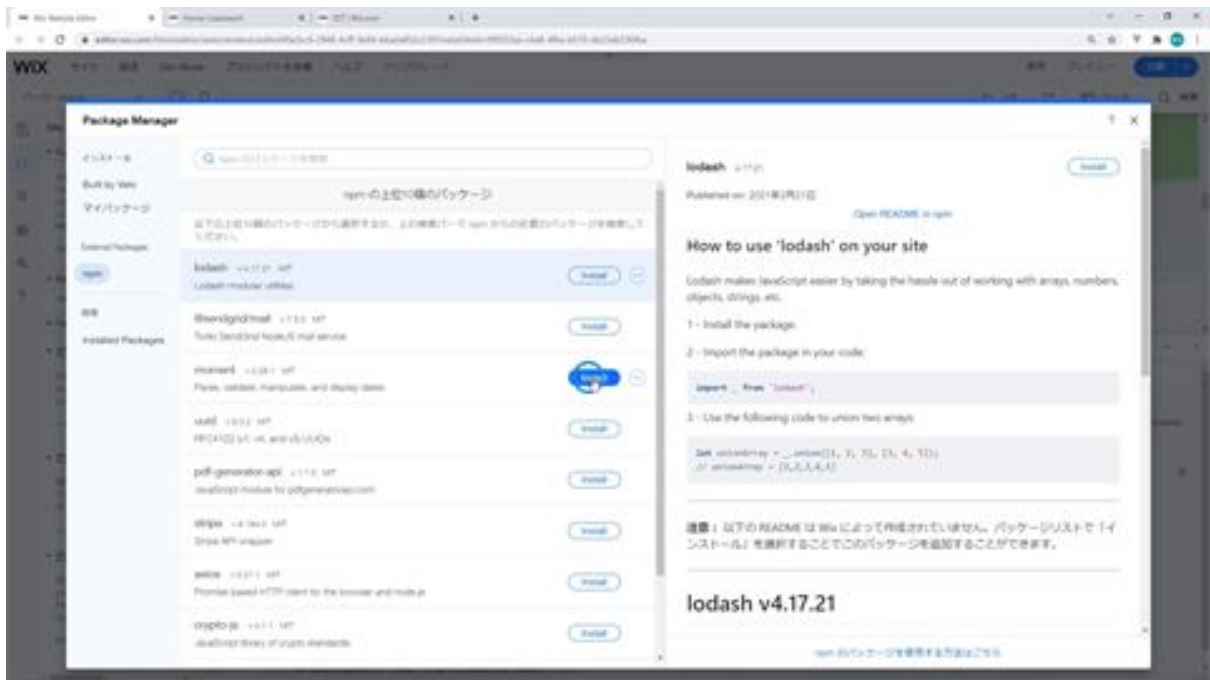
```
* @param {$w.MouseEvent} event
*/
```

```
export function deleteConfirm_click(event) {  
  
}  
  
/**  
*    Adds an event handler that runs when the element is clicked.  
    [Read more]  
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)  
*    @param {$w.MouseEvent} event  
*/  
export function cancel_click(event) {  
  
}  
  
/**  
*    Adds an event handler that runs when the element is clicked.  
    [Read more]  
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)  
*    @param {$w.MouseEvent} event  
*/  
export function delete_click(event) {  
  
}  
  
function viewInfo(message){  
  
    let puffInfoOptions = {  
        "duration": 500,  
        "delay": 2000  
    };  
  
    $w("#confirmText").text = message;  
    $w("#confirmBtn").hide();  
    $w("#confirm").show("puff",puffOptions).then( ( ) => {  
        $w("#confirm").hide("puff",puffInfoOptions)  
    })  
}
```

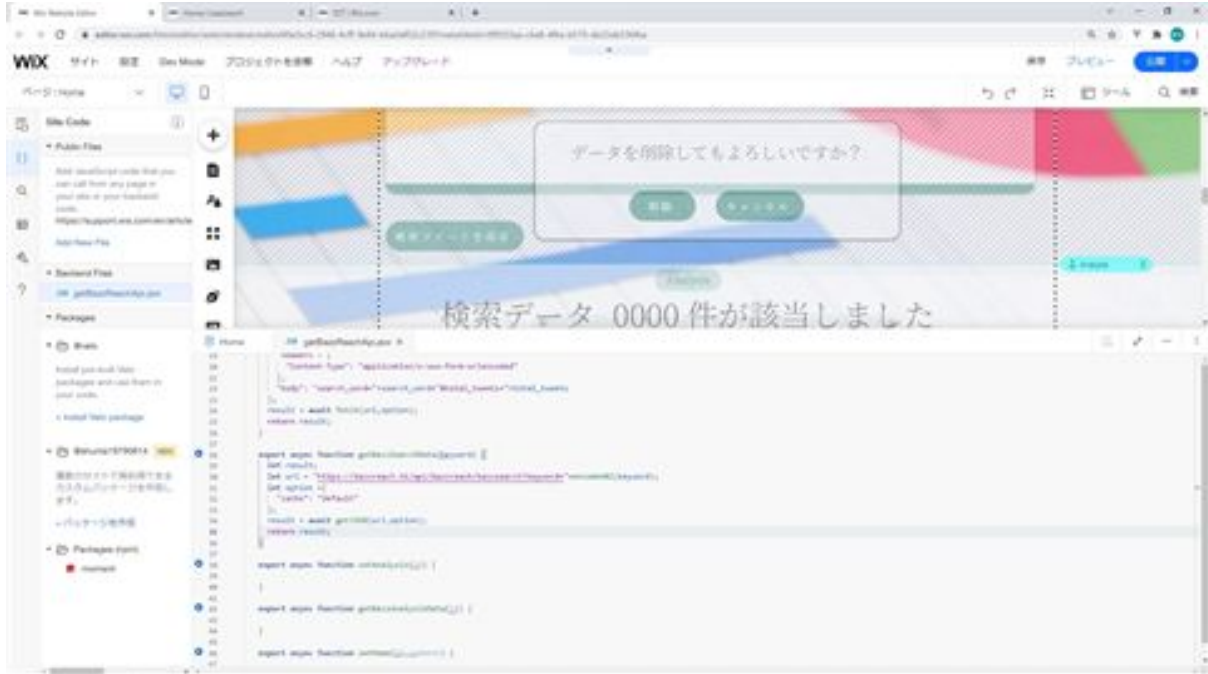
```
} );  
}
```




First, we want to display the elapsed time on the screen, but since it is displayed in English as it is now, we will install something called "moment" to display it in Japanese.



Install Moment by clicking on the Install Velo Package link under Packages in Wix. After installation is complete, add the `getBazzSearchData` function processing to the `getBazzReachApi` Web module.



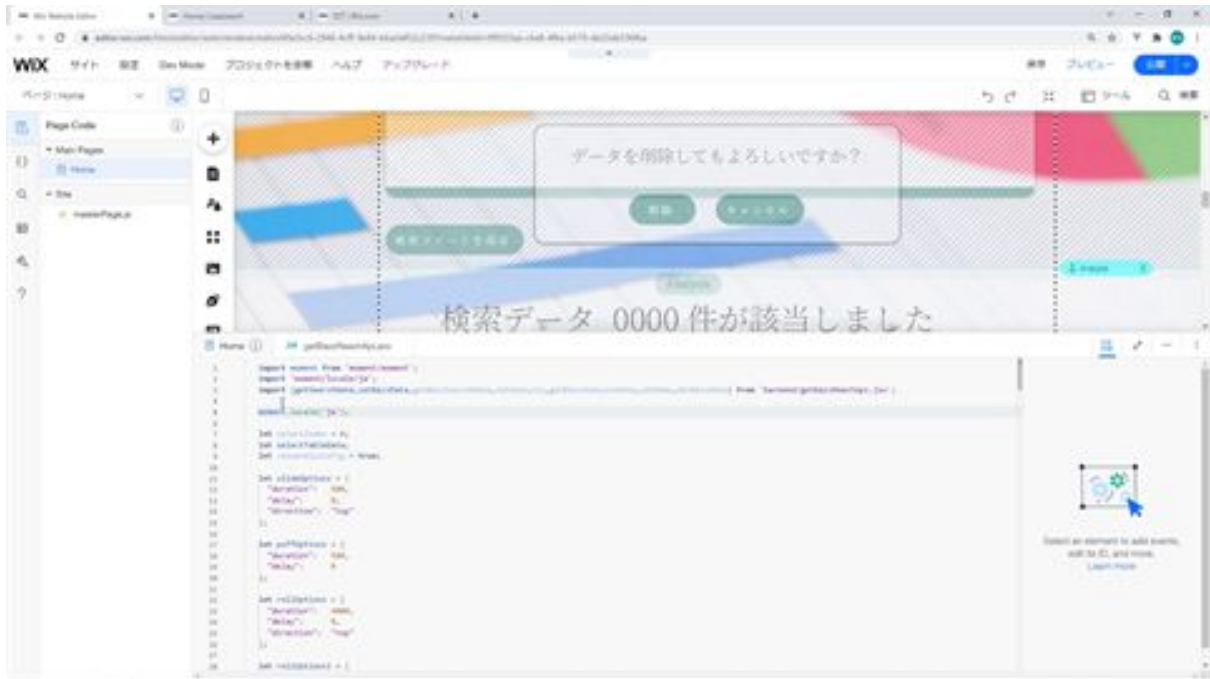
The keyword behind the URL is passed as a URL parameter to be used in the Get submission method so that it can be received by the Laravel side.

The `encodeURI` function used here is a function to allow Japanese characters to be used as URL parameters. the option is used to configure cache settings, and the request cache is set to the default setting in the cache option settings. Finally, `getJSON` is used to retrieve JSON data and return it to the caller. Next, on the Home side, we add the processing necessary for the initial display.

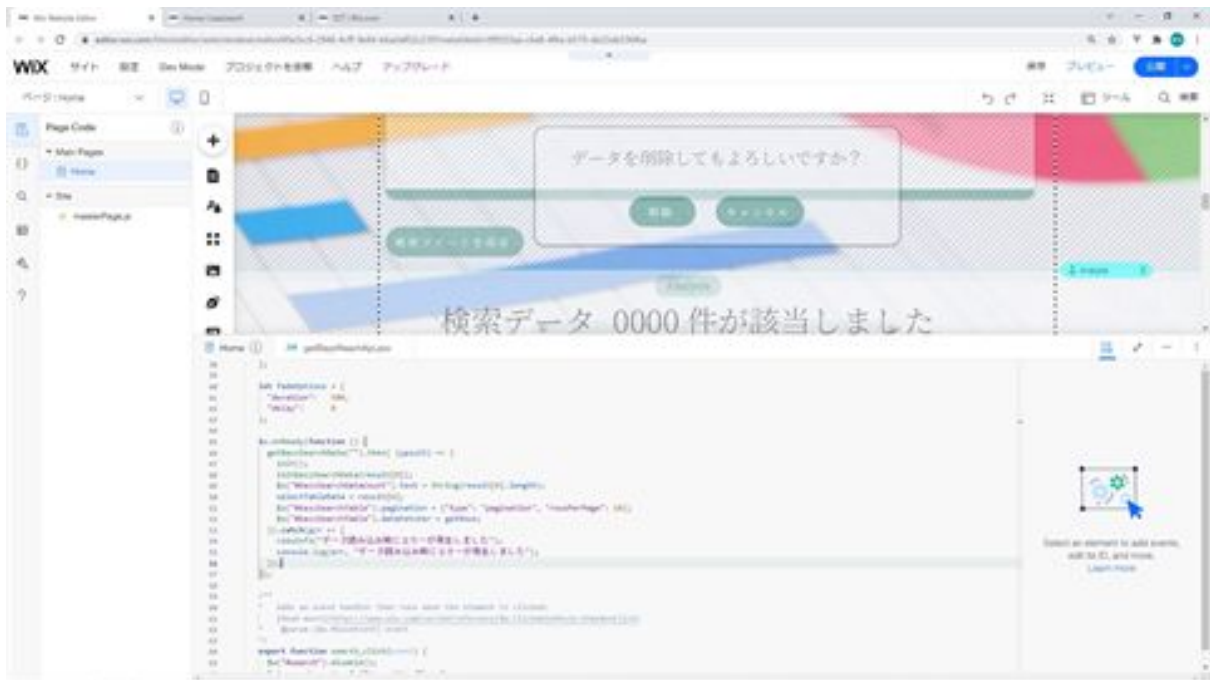
Next is `initBazzSearchData`, which is the process of formatting the table data to be displayed on the screen in a table. The content of the tweet is converted to empty characters so that the content of the tweet is limited to 50 characters and if the content of the sentiment analysis is null, it is not displayed as null.



The next `getRows` function is used to set data in a table. The table data is set in the `selectTableData` variable, and the process to set the data to the table is described from there. The `getRows` function returns a Promise object, which is an object that enables asynchronous processing.



Next, the initial display process is described. First, import the added moment so that it can be used. Then, configure the settings so that the moments can be used in Japanese.

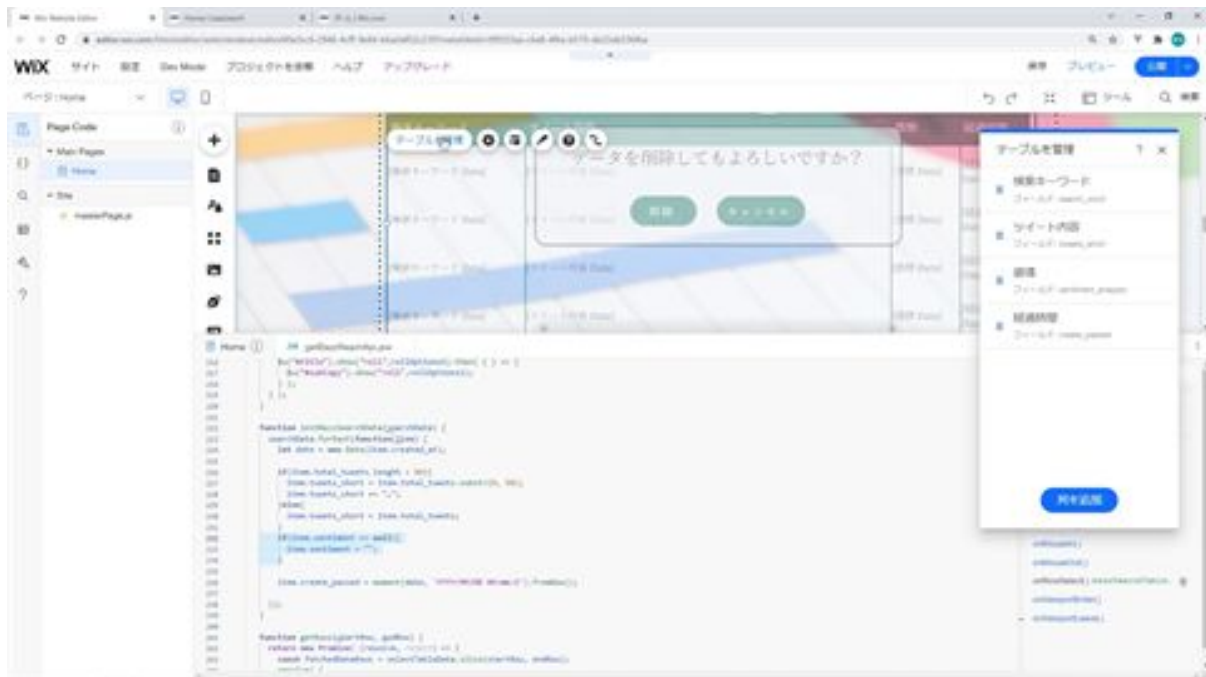


We will describe the initial display process for this onReady function. To explain this initial display process, call the getBazzSearchData function you created. When the resulting tweet data is returned, it is set to "result," so the init function is called first. This process hides the loading and displays the strips that should be displayed since the loading is displayed and all the strips are inactive in the initial screen display.

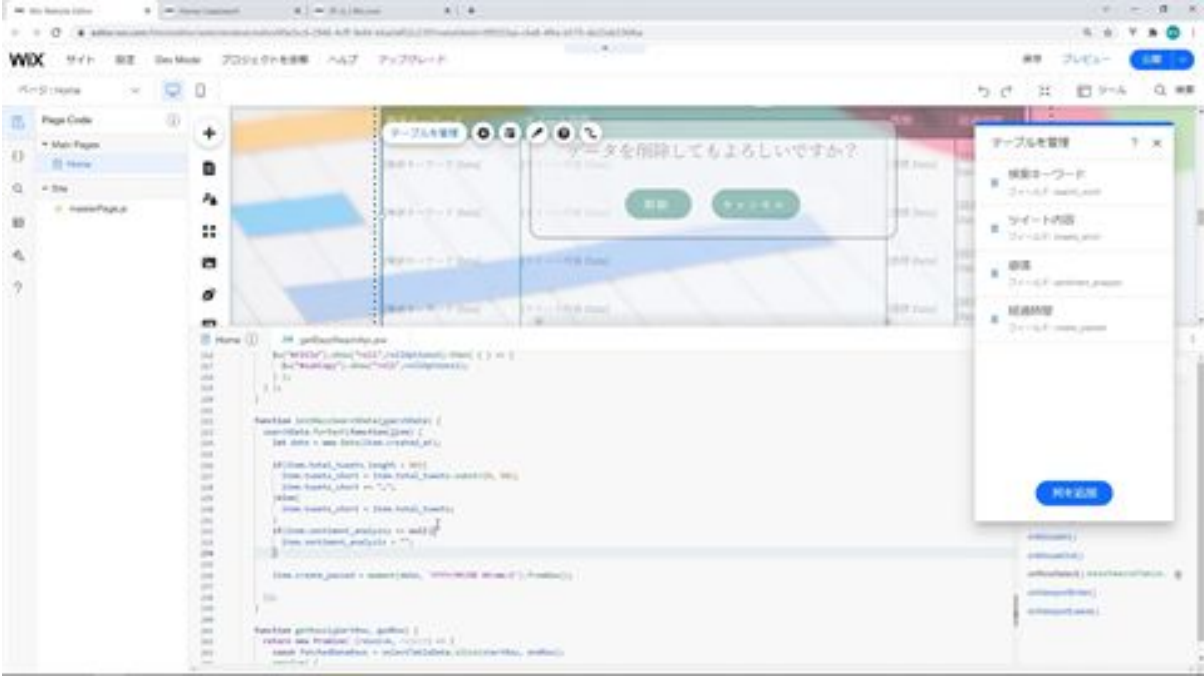
Call initBazzSearchData to process the data to be displayed in the table. The data passed as an argument is a data with an index of 0 in the result array, which is returned as an array, so the index is specified as 0.

After the table data has been formatted, the number of data is set to the text with ID bazzSearchDataCount, and selectTableData is set to the data with index 0 in the result array. The pager is set and the getRows function is set in dataFetcher. getRows function is set in dataFetcher so that the values are displayed in the table.

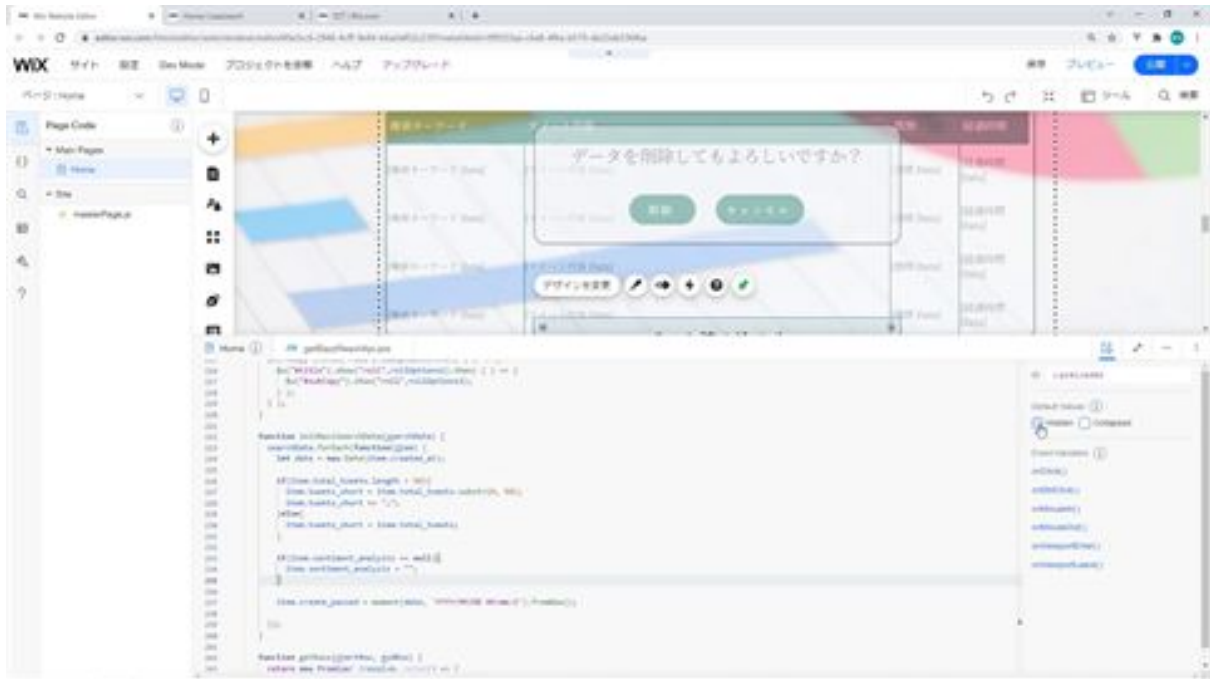
If an error occurs in the getBazzSearchData function, the viewInfo function displays the screen and shows the error in the console log.



In the `initBazzSearchData` function, a conditional branch is performed if `sentiment` is null, but the table displaying the data uses an item called `sentime_analysis`.



We want the `sentime_analysis` item to determine if it is null or not, so we modify `sentiment` to this `sentime_analysis` and modify it to make it empty if it is null.



Next, make sure that preLoader is not set to hidden. Once this has been corrected, display the screen and check that the data is displayed in the table.



We have confirmed that the saved tweet data is correctly displayed in the table. This is all for the table display of registered tweets.

BazzReachController.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Models\BazzReach;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Api\TwitterApi;
```

```
class BazzReachController extends Controller
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function index()
```

```
    {
```

```
        //
```

```
    }
```

```
    /**
```

```
     * Show the form for creating a new resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function create()
```

```
    {
```

```
        //
```



```

}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $ bazzreach = new BazzReach ();
    $ bazzreach->search_word = $request->input('search_word');
    $ bazzreach->total_tweets = $request->input('total_tweets');
    $ bazzreach->save();

    return http_response_code();
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function show( BazzReach $ bazzReach )
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach

```

```

* @return \Illuminate\Http\Response
*/
public function edit( BazzReach $ bazzReach )
{
    //
}

/**
* Update the specified resource in storage.
*
* @param \Illuminate\Http\Request $request
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function update(Request $request, BazzReach $ bazzReach )
{
    //
}

/**
* Remove the specified resource from storage.
*
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function destroy( BazzReach $ bazzReach )
{
    //
}

/**
* Display a listing of the resource.
*

```

```

* @return \Illuminate\Http\Response
*/
public function search(Request $request)
{
    $searchWord = $request->input('search_word');
    $twitterApi = new TwitterApi();
    $totalTweets = $twitterApi->searchTweets($searchWord);
    $dataArray = [$totalTweets];
    $data = response()->json($dataArray);
    return $data;
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function bazzsearch (Request $request)
{
    $keyword = $request->input('keyword');
    $bazzReachs;

    if(empty($keyword)){
        $ bazzReachs = BazzReach ::orderBy('created_at', 'desc')->get();
    }else{
        $query = BazzReach::query();
        $keywords = explode(" ", $keyword);

        foreach ($keywords as $Key => $Value) {
            $query->orWhere('search_word', 'ilike', '%'.$Value.'%');
        }
        $bazzReachs = $query->orderBy('created_at', 'desc')->get();
    }
}

```

```
$dataArray = [$bazzReachs];  
$data = response()->json($dataArray);  
return $data;  
}  
}
```

getBazzReachApi.jsw

```
import {fetch,getJSON} from 'wix-fetch';

export async function getSearchData(searchWord) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/search?
search_word="+encodeURIComponent(searchWord);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setBazzData (search_word,total_tweets) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/";
  let option ={
    "method": "POST",
    "mode":"core",
    "headers": {
      "Content-Type": "application/x-www-form-urlencoded"
    },
    "body": "search_word="+search_word+"&total_tweets="+total_tweets
  };
  result = await fetch(url,option);
  return result;
}
```

```
export async function getBazzSearchData (keyword) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/bazzsearch?
keyword="+encodeURIComponent(keyword);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setAnalysis(id) {

}

export async function getBazzAnalysisData(id) {

}

export async function setMemo(id,comment) {

}

export async function delBazzData(id) {

}
```

home.js

```
import moment from 'moment/moment';
import 'moment/locale/ja';
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

moment.locale('ja');

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};

let rollOptions1 = {
```

```

    "duration": 1500,
    "delay": 0,
    "direction": "left"
};

let rollOptions2 = {
    "duration": 1000,
    "delay": 0,
    "direction": "left"
};

let fadeOptions = {
    "duration": 500,
    "delay": 0
};

$w.onReady(function () {
    getBazzSearchData ("").then( (result) => {
        init();
        initBazzSearchData(result[0]);
        $w("#bazzSearchDataCount").text = String(result[0].length);
        selectTableData = result[0];
        $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
        $w("# bazzSearchTable ").dataFetcher = getRows;
    }).catch(err => {
        viewInfo("An error occurred during data loading");
        console.log(err, "An error occurred during data loading");
    });
});

/**
 *      Adds an event handler that runs when the element is clicked.

```


[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  getSearchData(search_word).then( (result) => {

    if($w("#saveTweetStrip").collapsed){
      $w("#saveTweetStrip").expand().then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }else{
      $w("#saveTweetStrip").hide("slide",slideOptions).then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }
  }).catch(err => {
    viewInfo("An error occurred when searching for tweets");
    console.log(err, "An error occurred when searching for tweets");
    $w("#search").enable();
  });
}
```

```

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveTweet_click(event) {
  $w("#saveTweet").disable();

  setBazzData($w("#viewSearchWord").text,$w("#viewSearchData").value).t
  hen( (result) => {
    bazzSearch_click(event);
    viewInfo("Saved tweet");
    $w("#saveTweetStrip").hide("slide",slideOptions).then( ( ) => {
      $w("#viewSearchWord").text = "";
      $w("#viewSearchData").value = "";
      $w("#saveTweetStrip").collapse();
      $w("#saveTweet").enable();
    } );
  }).catch(err => {
    viewInfo("An error occurred when saving the tweet");
    console.log(err, "An error occurred when saving the tweet");
    $w("#saveTweet").enable();
  });
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */

```

```
export function bazzSearch_click(event) {  
  
}  
  
/**  
*    Adds an event handler that runs when a table row is selected.  
    [Read more]  
(https://www.wix.com/corvid/reference/$w.Table.html#onRowSelect)  
*    @param {$w.TableRowEvent} event  
*/  
export function bazzSearchTable_rowSelect(event) {  
  
}  
  
/**  
*    Adds an event handler that runs when the element is clicked.  
    [Read more]  
(https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick)  
*    @param {$w.MouseEvent} event  
*/  
export function viewAnalysis_click(event) {  
  
}  
  
/**  
*    Adds an event handler that runs when the element is clicked.  
    [Read more]  
(https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick)  
*    @param {$w.MouseEvent} event  
*/  
export function viewMemoInput_click(event) {  
  
}  
  
/**
```

```
*      Adds an event handler that runs when the element is clicked.
      [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
*      @param {$w.MouseEvent} event
*/
export function saveMemo_click(event) {

}

/**
*      Adds an event handler that runs when the element is clicked.
      [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
*      @param {$w.MouseEvent} event
*/
export function deleteConfirm_click(event) {

}

/**
*      Adds an event handler that runs when the element is clicked.
      [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
*      @param {$w.MouseEvent} event
*/
export function cancel_click(event) {

}

/**
*      Adds an event handler that runs when the element is clicked.
      [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
*      @param {$w.MouseEvent} event
```

```

*/
export function delete_click(event) {
}

function viewInfo(message){

  let puffInfoOptions = {
    "duration": 500,
    "delay": 2000
  };

  $w("#confirmText").text = message;
  $w("#confirmBtn").hide();
  $w("#confirm").show("puff",puffOptions).then( () => {
    $w("#confirm").hide("puff",puffInfoOptions)
  } );
}

function init(){
  $w("#preLoader").hide("fade",fadeOptions);
  $w("#searchWordStrip").show("fade",fadeOptions);
  $w("#bazzSearchStrip").show("fade",fadeOptions);
  $w("#bazzSearchTableStrip").show("fade",fadeOptions);
  $w("#copy").show("roll",rollOptions).then( () => {
    $w("#title").show("roll",rollOptions1).then( () => {
      $w("#subCopy").show("roll",rollOptions2);
    } );
  } );
}

function initBazzSearchData (searchData) {
  searchData.forEach(function(item) {
    let date = new Date(item.created_at);

```

```
if(item.total_tweets.length > 50){
  item.tweets_short = item.total_tweets.substr(0, 50);
  item.tweets_short += "...";
}else{
  item.tweets_short = item.total_tweets;
}

if(item.sentiment_analysis == null){
  item.sentiment_analysis = "";
}

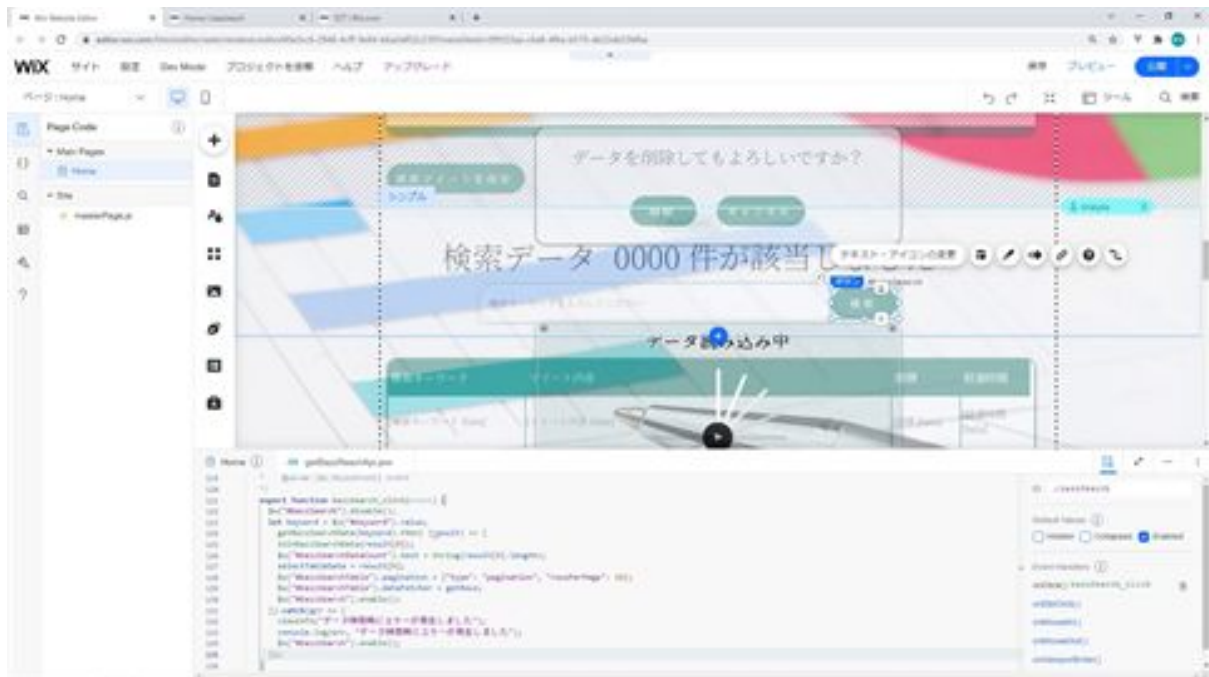
item.create_passed = moment(date, 'YYYY/MM/DD
HH:mm:S').fromNow();

});
}

function getRows(startRow, endRow) {
  return new Promise( (resolve, reject) => {
    const fetchedDataRows = selectTableData.slice(startRow, endRow);
    resolve( {
      pageRows: fetchedDataRows,
      totalRowCount: selectTableData.length
    } );
  } );
}
```

Let's display registered tweets with partial search

Next, we would like to display registered tweets in a partial search. We have already created a bazzsearch method to perform the search, so we will only modify wix this time.



To explain the process of this function, first deactivate the search button whose ID is bazzSearch to prevent double submissions, retrieve the search keyword from the text field whose ID is the keyword, and call the getBazzSearchData function that you created. When the resulting tweet data is returned, it is set to the result variable, so call initBazzSearchData to process the data to be displayed in the table.

After the table data has been formatted, set the number of data in the text with ID bazzSearchDataCount, set the selectTableData variable to the data with index 0 in the result array, and set the table with ID bazzSearchTable to Set the pager. The dataFetcher is then used to set the data in the table.

The data to be set is set using the `getRows` function so that it becomes `selectTableData`. Next, the deactivated search button is activated.

If the call to `getBazzSearchData` results in an error, the `viewInfo` function displays the error screen, outputs the error log to the console, and activates the search button.

Once you have made this modification, check to see if you can filter the data by keywords on the screen.



First, register multiple tweet data.



Next, enter your search terms and click the Search button. You will then see that the data is correctly narrowed down and displayed for multiple keywords. This is the end of the partial search display of registered tweets.

home.js

```
import moment from 'moment/moment';
import 'moment/locale/ja';
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

moment.locale('ja');

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};

let rollOptions1 = {
```

```

    "duration": 1500,
    "delay": 0,
    "direction": "left"
};

let rollOptions2 = {
    "duration": 1000,
    "delay": 0,
    "direction": "left"
};

let fadeOptions = {
    "duration": 500,
    "delay": 0
};

$w.onReady(function () {
    getBazzSearchData ("").then( (result) => {
        init();
        initBazzSearchData(result[0]);
        $w("#bazzSearchDataCount").text = String(result[0].length);
        selectTableData = result[0];
        $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
        $w("# bazzSearchTable ").dataFetcher = getRows;
    }).catch(err => {
        viewInfo("An error occurred during data loading");
        console.log(err, "An error occurred during data loading");
    });
});

/**
 *      Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  getSearchData(search_word).then( (result) => {

    if($w("#saveTweetStrip").collapsed){
      $w("#saveTweetStrip").expand().then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }else{
      $w("#saveTweetStrip").hide("slide",slideOptions).then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }
  }).catch(err => {
    viewInfo("An error occurred when searching for tweets");
    console.log(err, "An error occurred when searching for tweets");
    $w("#search").enable();
  });
}
```

```

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveTweet_click(event) {
  $w("#saveTweet").disable();

  setBazzData($w("#viewSearchWord").text,$w("#viewSearchData").value).t
  hen( (result) => {
    bazzSearch_click(event);
    viewInfo("Saved tweet");
    $w("#saveTweetStrip").hide("slide",slideOptions).then( ( ) => {
      $w("#viewSearchWord").text = "";
      $w("#viewSearchData").value = "";
      $w("#saveTweetStrip").collapse();
      $w("#saveTweet").enable();
    } );
  }).catch(err => {
    viewInfo("An error occurred when saving the tweet");
    console.log(err, "An error occurred when saving the tweet");
    $w("#saveTweet").enable();
  });
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */

```

```

export function bazzSearch_click(event) {
  $w("# bazzSearch ").disable();
  let keyword = $w("#keyword").value;
    getBazzSearchData (keyword).then( (result) => {
  initBazzSearchData(result[0]);
  $w("#bazzSearchDataCount").text = String(result[0].length);
  selectTableData = result[0];
  $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
  $w("# bazzSearchTable ").dataFetcher = getRows;
  $w("# bazzSearch ").enable();
}).catch(err => {
  viewInfo("An error occurred during data retrieval");
  console.log(err, "An error occurred during data retrieval");
  $w("# bazzSearch ").enable();
});
}

/**
 *   Adds an event handler that runs when a table row is selected.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.Table.html#onRowSelect)
 *   @param {$w.TableRowEvent} event
 */
export function bazzSearchTable_rowSelect(event) {

}

/**
 *   Adds an event handler that runs when the element is clicked.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event

```

```
*/
export function viewAnalysis_click(event) {
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function viewMemoInput_click(event) {
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveMemo_click(event) {
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function deleteConfirm_click(event) {
}
```

```

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function cancel_click(event) {
}

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function delete_click(event) {
}

function viewInfo(message){
  let puffInfoOptions = {
    "duration": 500,
    "delay": 2000
  };

  $w("#confirmText").text = message;
  $w("#confirmBtn").hide();
  $w("#confirm").show("puff",puffOptions).then( () => {
    $w("#confirm").hide("puff",puffInfoOptions)
  } );
}

function init(){

```



```

$w("#preLoader").hide("fade",fadeOptions);
$w("#searchWordStrip").show("fade",fadeOptions);
$w("#bazzSearchStrip").show("fade",fadeOptions);
$w("#bazzSearchTableStrip").show("fade",fadeOptions);
$w("#copy").show("roll",rollOptions).then( () => {
  $w("#title").show("roll",rollOptions1).then( () => {
    $w("#subCopy").show("roll",rollOptions2);
  } );
} );
}

```

```

function initBazzSearchData (searchData) {
  searchData.forEach(function(item) {
    let date = new Date(item.created_at);

    if(item.total_tweets.length > 50){
      item.tweets_short = item.total_tweets.substr(0, 50);
      item.tweets_short += "...";
    }else{
      item.tweets_short = item.total_tweets;
    }

    if(item.sentiment_analysis == null){
      item.sentiment_analysis = "";
    }

    item.create_passed = moment(date, 'YYYY/MM/DD
HH:mm:S').fromNow();

  });
}

```

```

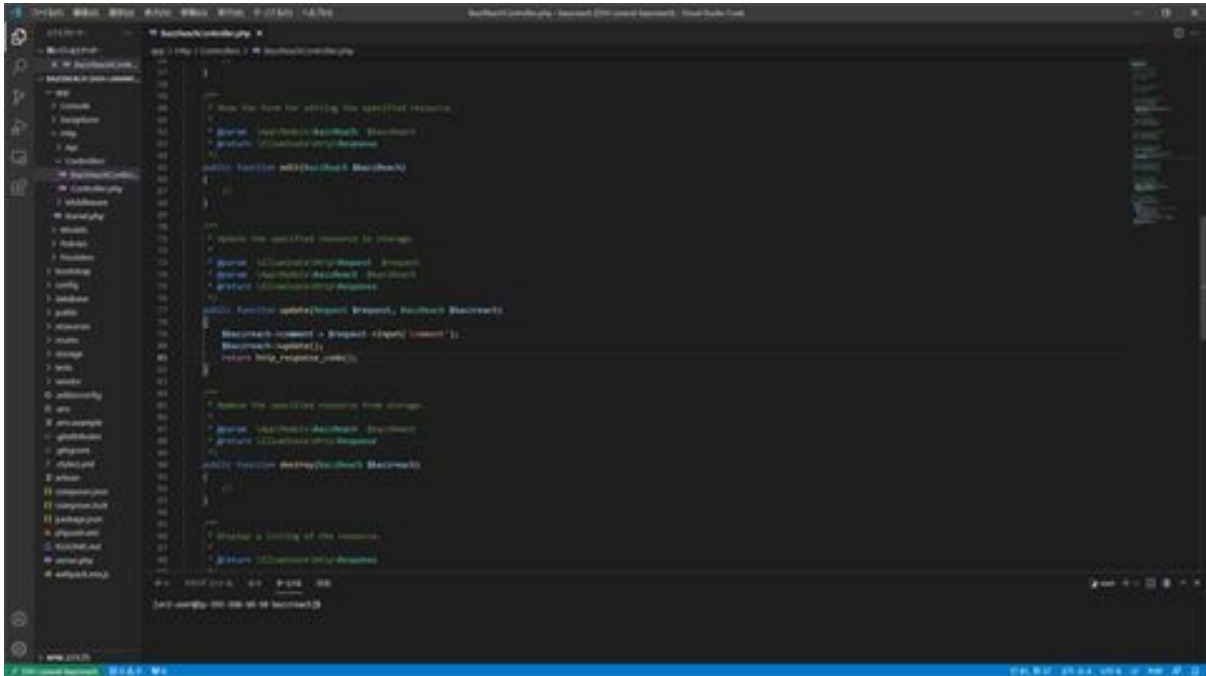
function getRows(startRow, endRow) {
  return new Promise( (resolve, reject) => {

```

```
const fetchedDataRows = selectTableData.slice(startRow, endRow);
resolve( {
  pageRows: fetchedDataRows,
  totalRowCount: selectTableData.length
});
});
}
```

Let's create a memo function for registered tweets

Next, we will create a memo function for the registered tweets.
First, open BazzReachContoroller and modify the update method.



```

    * @param $request BazzReach $request
    * @param $comment BazzReach $comment
    * @return BazzReach $comment

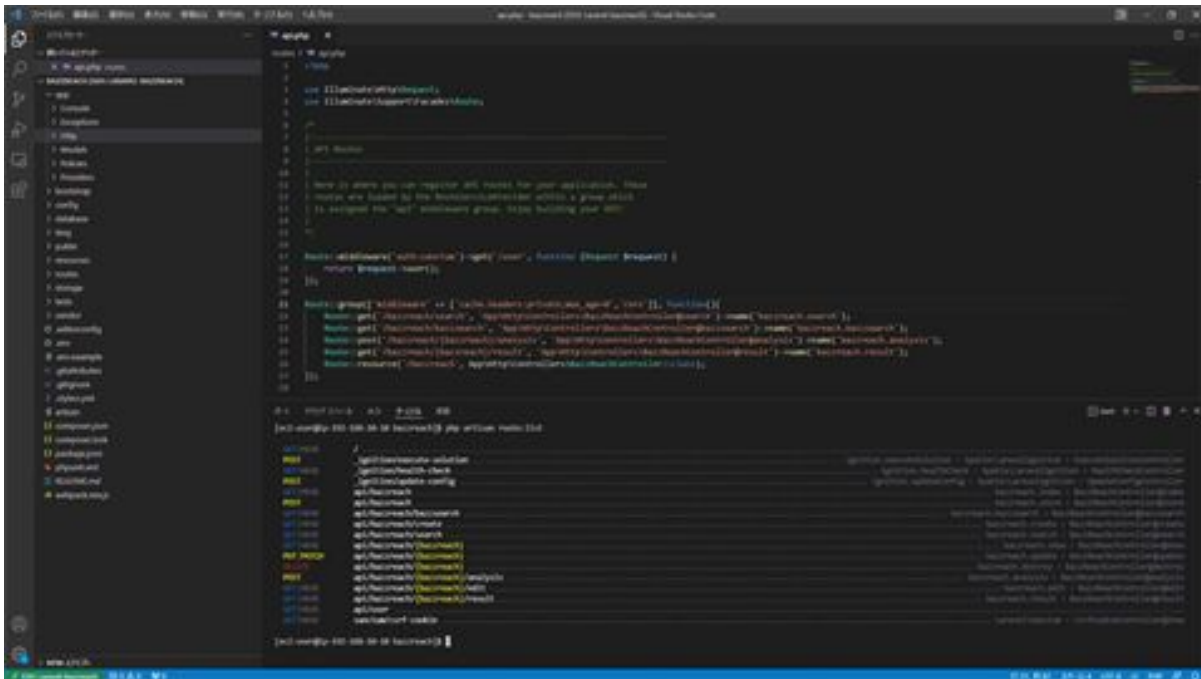
    public function update($request, BazzReach $comment)
    {
        // Update the specified resource to storage
        $comment->comment = $request->input('comment');
        $comment->update();
        return $comment->save();

        // Return the specified resource from storage
        $comment = $comment->find($comment->id);
        return $comment;
    }

    /**
     * Update a listing of the resource.
     * @param $comment BazzReach $comment
     */
}

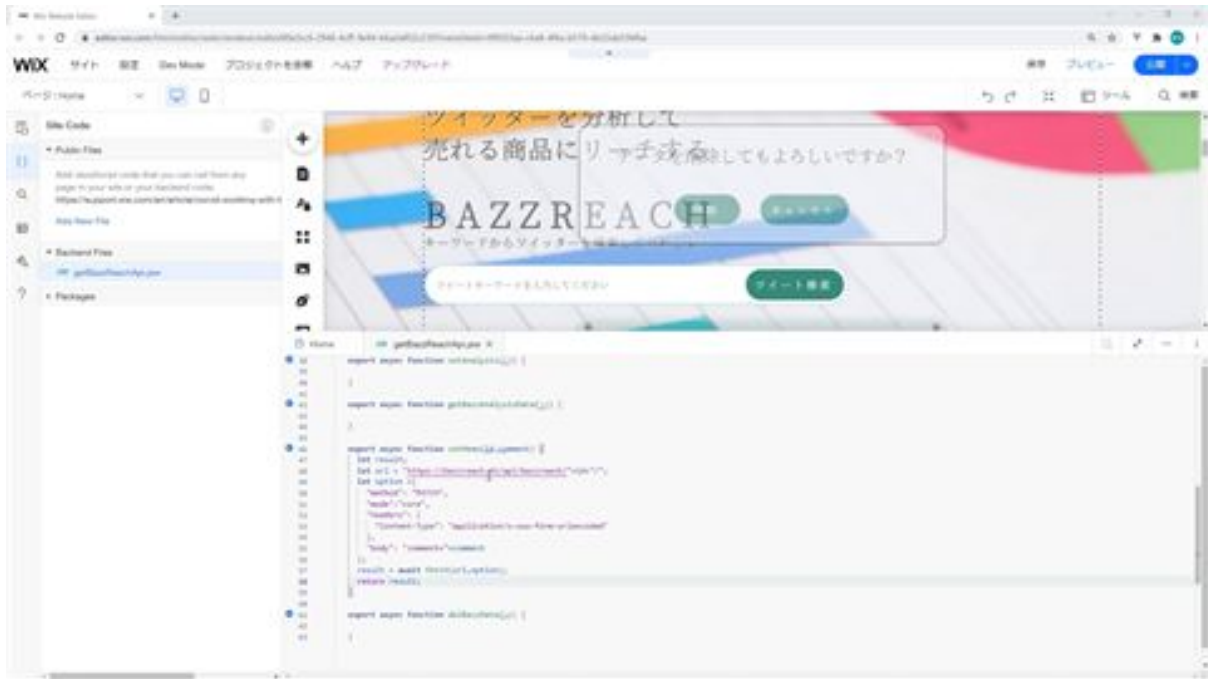
```

This method explains that the data of the comment passed from the screen is retrieved from \$request. Set the comment item in the BazzReach model. Once set, the BazzReach model instance Update data in the bazz_reaches table. Finally, the HTTP response code is returned with the return.



Also, as for the variable name of the argument of the update method, this part must match the variable name in the URI and the argument name of the method. Otherwise, the data will not be retrieved correctly.

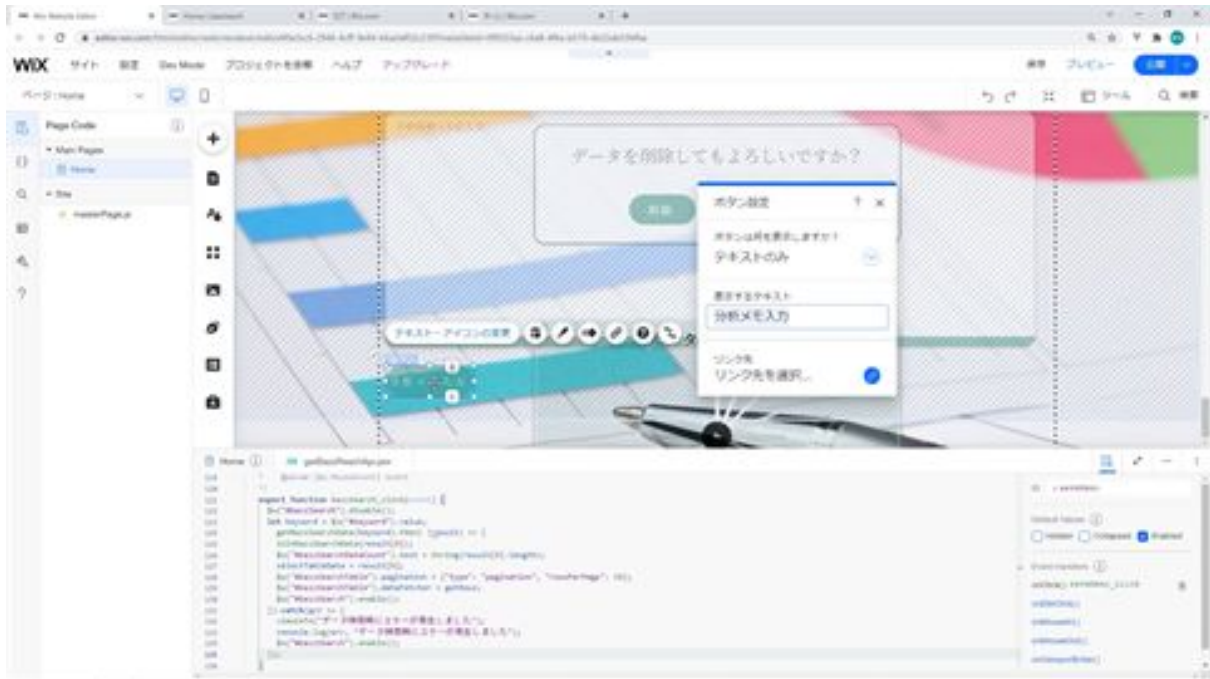
If the variable name \$bazzreach in the method argument and the variable name of the {bazzreach} parameter in the update method's URI do not match, the data cannot be retrieved even if the ID is passed in the URL. Therefore, we need to match the variable names in the routing with the argument names in the method. After this modification, the next step is to modify the data so that it can be displayed on the screen side.



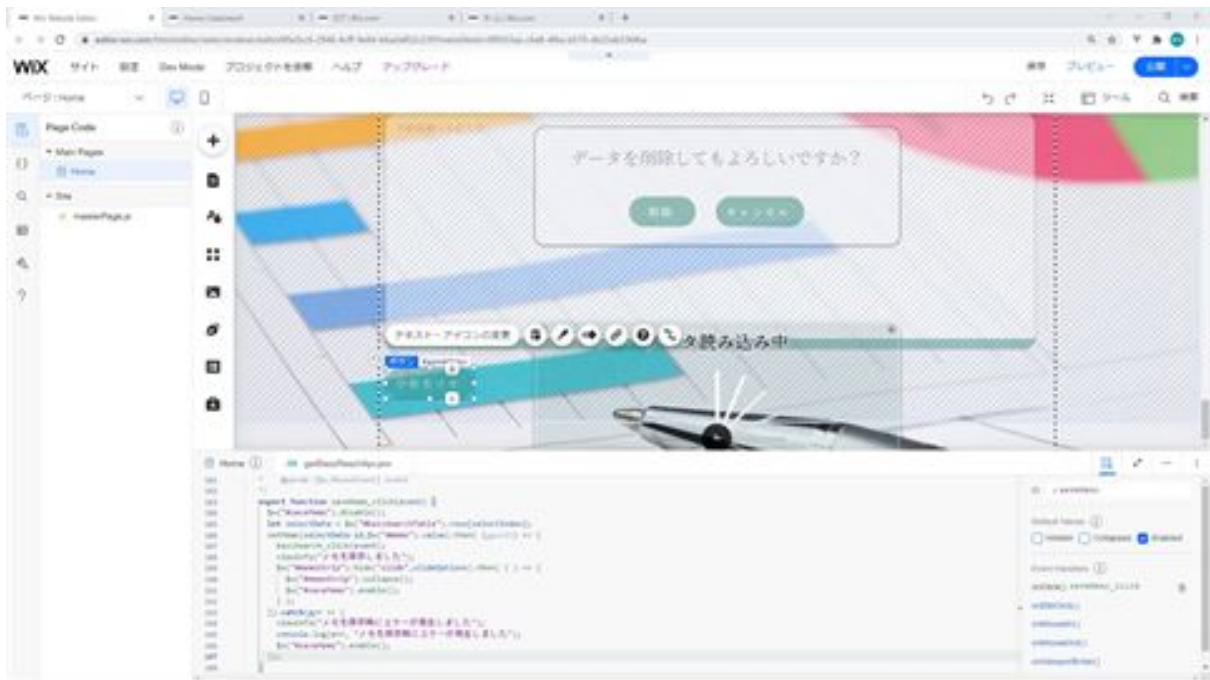
First, add processing to the setMemo function in the getBazzReachApi Web module. The process of this function is to access the update method created in Laravel and update the data by passing a comment. PATCH is specified in the METHOD so that it can be received as a method. The reason why the sending method is PATCH is that it is set in Laravel's routing.

In the headers section, the content type is specified as the content type for submitting the form, and an id is added to the URL because Laravel's routing configuration is set to pass the primary key of the table in the URL.

Finally, PATCH is sent using fetch, which allows cross-origin requests by specifying cors in the mode specification, allowing access to the API on the Laravel side.



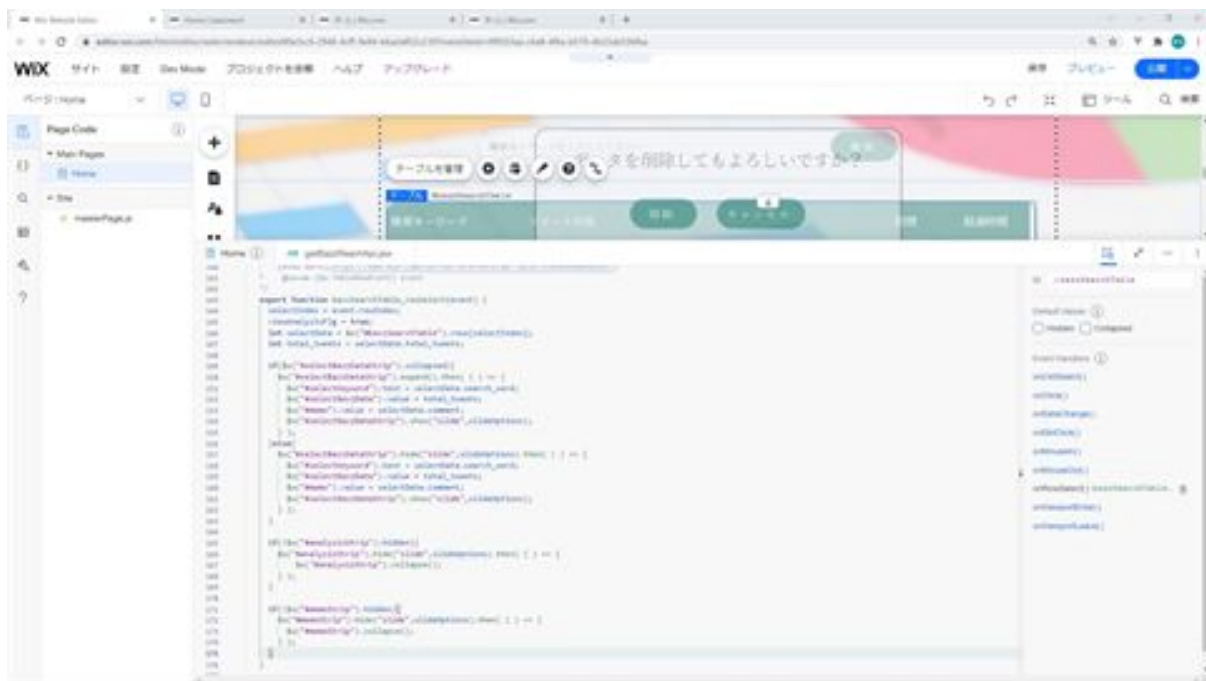
Next, on the Home side, add processing to the onClick event function of the Memo Analysis button. First, for the Memo Analyze button, the button caption was not correct, so we will correct it here.



Next, add processing to the onClick event function of this button. First, deactivate the memo button with the ID saveMemo to prevent double submissions. Next, the table data currently selected is acquired from the table whose ID is bazzSearchTable.

Next, call the setMemo function you created. Since the data will be updated after the call, call the bazzSearch_click function to search the data and update the table information.

Displays a dialog saying that the memo has been saved, hides the strip with ID memoStrip with animation, and activates the memo button for analysis. If the setMemo function fails, it displays an error dialog and outputs an error log to the console. Finally, activate the Memo Analysis button.



Next, we add processing when a row is selected from a table whose ID is bazzSearchTable. To explain the processing of this function, first set the index of the selected row to selectIndex and set viewAnalysisFlg to true; set the data of the selected row to selectData and set total_tweets to the tweets of the selected row The following is a list of the most common problems that can be encountered.

If the strip with ID selectBazzDataStrip is collapsed, set a keyword in the text field with ID selectKeyword, display the tweet of the selected line in the text box with ID selectBazzData, display the saved memo in the text box with ID memo, and animate the selectBazzDataStrip.

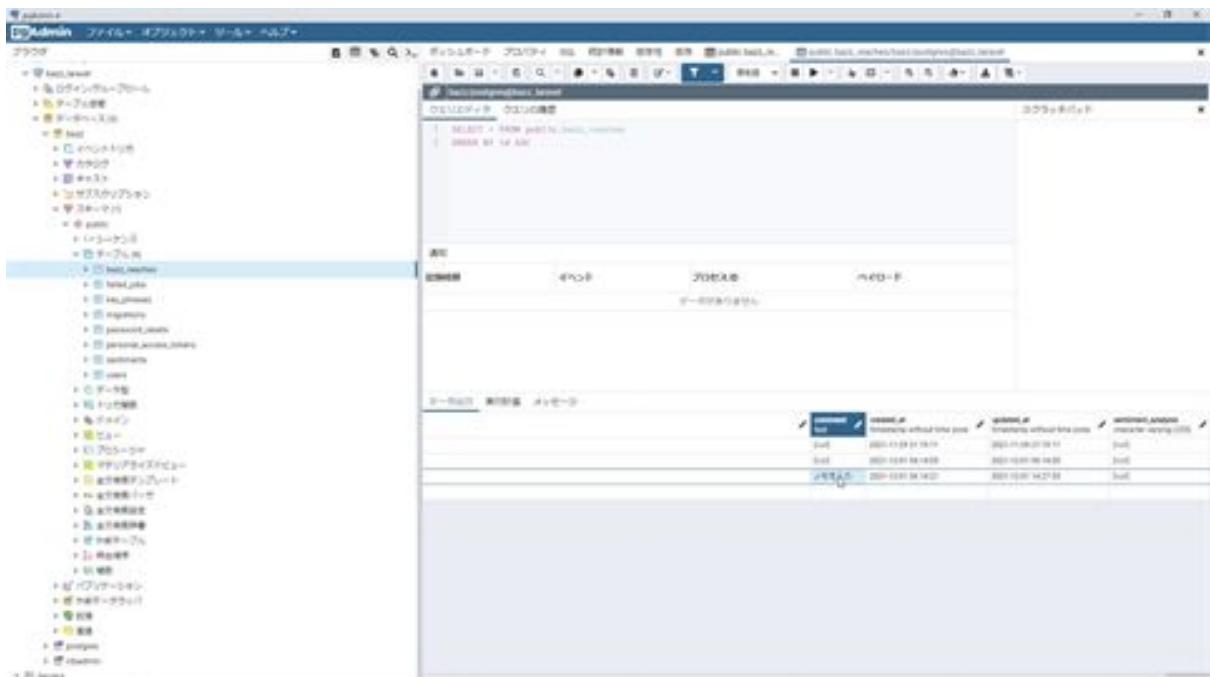
If the selectBazzDataStrip is not collapsed, hide the animated selectBazzDataStrip and set the keywords, tweets, and notes as above. After setting, animate selectBazzDataStrip. If analysisStrip for detailed analysis and memoStrip for the memo is displayed, hide them with animation and collapse them.



Next, add processing to the `onClick` event function of the analysis memo input button. To explain the processing of this function, we deactivate the analysis memo input button, and if the strip with ID `memoStrip` is folded, it is animated, and if it is displayed, it is hidden with animation in the accordion panel. After modifying so far, check if the memo can be saved on the screen.



screen, enter a memo, and click the Memo button for the analysis. The memo is saved and a dialog appears.



Now check to see if it is saved in PGAdmin; we can confirm that the notes we entered are saved in PGAdmin. That's all for creating the memo function

for registered tweets.

BazzReachController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\BazzReach;
use Illuminate\Http\Request;
use App\Http\Api\TwitterApi;

class BazzReachController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }
}
```

```

}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $ bazzreach = new BazzReach ();
    $ bazzreach->search_word = $request->input('search_word');
    $ bazzreach->total_tweets = $request->input('total_tweets');
    $ bazzreach->save();

    return http_response_code();
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function show( BazzReach $ bazzReach )
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach

```

```

* @return \Illuminate\Http\Response
*/
public function edit( BazzReach $ bazzReach )
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, BazzReach $
bazzReach )
{
    $ bazzreach- >comment = $request->input('comment');
    $ bazzreach- >update();
    return http_response_code();
}

/**
 * Remove the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function destroy( BazzReach $ bazzReach )
{
    //
}

```

```

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function search(Request $request)
{
    $searchWord = $request->input('search_word');
    $twitterApi = new TwitterApi();
    $totalTweets = $twitterApi->searchTweets($searchWord);
    $dataArray = [$totalTweets];
    $data = response()->json($dataArray);
    return $data;
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function bazzsearch (Request $request)
{
    $keyword = $request->input('keyword');
    $bazzReachs;

    if(empty($keyword)){
        $ bazzReachs = BazzReach ::orderBy('created_at', 'desc')-
>get();
    }else{
        $query = BazzReach::query();
        $keywords = explode(" ", $keyword);

        foreach ($keywords as $Key => $Value) {

```

```
        $query->orWhere('search_word', 'ilike', '%'.$Value.'%');
    }
    $bazzReachs = $query->orderBy('created_at', 'desc')->get();
}
$dataArray = [$bazzReachs];
$data = response()->json($dataArray);
return $data;
}
}
```

getBazzReachApi.jsw

```
import {fetch,getJSON} from 'wix-fetch';

export async function getSearchData(searchWord) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/search?
search_word="+encodeURIComponent(searchWord);
  let option = {
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setBazzData (search_word,total_tweets) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/";
  let option = {
    "method": "POST",
    "mode":"core",
    "headers": {
      "Content-Type": "application/x-www-form-urlencoded"
    },
    "body": "search_word="+search_word+"&total_tweets="+total_tweets
  };
  result = await fetch(url,option);
  return result;
}
```



```
export async function getBazzSearchData (keyword) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/bazzsearch?
keyword="+encodeURIComponent(keyword);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setAnalysis(id) {

}

export async function getBazzAnalysisData(id) {

}

export async function setMemo(id,comment) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/"+id+"/";
  let option ={
    "method": "PATCH",
    "mode":"core",
    "headers": {
      "Content-Type": "application/x-www-form-urlencoded"
    },
    "body": "comment="+comment
  };
  result = await fetch(url,option);
  return result;
}
```

```
export async function delBazzData(id) {  
}
```

home.js

```
import moment from 'moment/moment';
import 'moment/locale/ja';
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

moment.locale('ja');

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};

let rollOptions1 = {
```

```

    "duration": 1500,
    "delay": 0,
    "direction": "left"
};

let rollOptions2 = {
    "duration": 1000,
    "delay": 0,
    "direction": "left"
};

let fadeOptions = {
    "duration": 500,
    "delay": 0
};

$w.onReady(function () {
    getBazzSearchData ("").then( (result) => {
        init();
        initBazzSearchData(result[0]);
        $w("#bazzSearchDataCount").text = String(result[0].length);
        selectTableData = result[0];
        $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
        $w("# bazzSearchTable ").dataFetcher = getRows;
    }).catch(err => {
        viewInfo("An error occurred during data loading");
        console.log(err, "An error occurred during data loading");
    });
});

/**
 *      Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  getSearchData(search_word).then( (result) => {

    if($w("#saveTweetStrip").collapsed){
      $w("#saveTweetStrip").expand().then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }else{
      $w("#saveTweetStrip").hide("slide",slideOptions).then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }
  }).catch(err => {
    viewInfo("An error occurred when searching for tweets");
    console.log(err, "An error occurred when searching for tweets");
    $w("#search").enable();
  });
}
```

```

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveTweet_click(event) {
  $w("#saveTweet").disable();

  setBazzData($w("#viewSearchWord").text,$w("#viewSearchData").value).t
  hen( (result) => {
    bazzSearch_click(event);
    viewInfo("Saved tweet");
    $w("#saveTweetStrip").hide("slide",slideOptions).then( ( ) => {
      $w("#viewSearchWord").text = "";
      $w("#viewSearchData").value = "";
      $w("#saveTweetStrip").collapse();
      $w("#saveTweet").enable();
    } );
  }).catch(err => {
    viewInfo("An error occurred when saving the tweet");
    console.log(err, "An error occurred when saving the tweet");
    $w("#saveTweet").enable();
  });
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */

```

```

export function bazzSearch_click(event) {
  $w("# bazzSearch ").disable();
  let keyword = $w("#keyword").value;
    getBazzSearchData (keyword).then( (result) => {
  initBazzSearchData(result[0]);
  $w("#bazzSearchDataCount").text = String(result[0].length);
  selectTableData = result[0];
  $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
  $w("# bazzSearchTable ").dataFetcher = getRows;
  $w("# bazzSearch ").enable();
}).catch(err => {
  viewInfo("An error occurred during data retrieval");
  console.log(err, "An error occurred during data retrieval");
  $w("# bazzSearch ").enable();
});
}

/**
 *   Adds an event handler that runs when a table row is selected.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.Table.html#onRowSelect)
 *   @param {$w.TableRowEvent} event
 */
export function bazzSearchTable_rowSelect(event) {
  selectIndex = event.rowIndex;
  viewAnalysisFlg = true;
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  let total_tweets = selectData.total_tweets;

  if($w("#selectBazzDataStrip").collapsed){
    $w("#selectBazzDataStrip").expand().then( ( ) => {

```

```

    $w("#selectKeyword").text = selectData.search_word;
    $w("# selectBazzData ").value = total_tweets;
    $w("#memo").value = selectData.comment;
    $w("#selectBazzDataStrip").show("slide",slideOptions);
    } );
} else {
    $w("#selectBazzDataStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#selectKeyword").text = selectData.search_word;
        $w("# selectBazzData ").value = total_tweets;
        $w("#memo").value = selectData.comment;
        $w("#selectBazzDataStrip").show("slide",slideOptions);
    } );
}

if(!$w("#analysisStrip").hidden){
    $w("#analysisStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#analysisStrip").collapse();
    } );
}

if(!$w("#memoStrip").hidden){
    $w("#memoStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#memoStrip").collapse();
    } );
}
}

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */

```



```

export function viewAnalysis_click(event) {
}

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function viewMemoInput_click(event) {
  $w("#viewMemoInput").disable();
  if($w("#memoStrip").collapsed){
    $w("#memoStrip").expand().then( () => {
      $w("#memoStrip").show("slide",slideOptions).then( () => {
        $w("#viewMemoInput").enable();
      } );
    } );
  }else{
    $w("#memoStrip").hide("slide",slideOptions).then( () => {
      $w("#memoStrip").collapse();
      $w("#viewMemoInput").enable();
    } );
  }
}

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function saveMemo_click(event) {

```

```

$w("#saveMemo").disable();
let selectData = $w("# bazzSearchTable ").rows[selectIndex];
setMemo(selectData.id,$w("#memo").value).then( (result) => {
  bazzSearch_click(event);
  viewInfo("Notes saved");
  $w("#memoStrip").hide("slide",slideOptions).then( ( ) => {
    $w("#memoStrip").collapse();
    $w("#saveMemo").enable();
  } );
}).catch(err => {
  viewInfo("An error occurred when saving the note");
  console.log(err, "An error occurred when saving the note");
  $w("#saveMemo").enable();
});
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function deleteConfirm_click(event) {
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function cancel_click(event) {

```

```

}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function delete_click(event) {

}

function viewInfo(message){

  let puffInfoOptions = {
    "duration": 500,
    "delay": 2000
  };

  $w("#confirmText").text = message;
  $w("#confirmBtn").hide();
  $w("#confirm").show("puff",puffOptions).then( () => {
    $w("#confirm").hide("puff",puffInfoOptions)
  } );
}

function init(){
  $w("#preLoader").hide("fade",fadeOptions);
  $w("#searchWordStrip").show("fade",fadeOptions);
  $w("#bazzSearchStrip").show("fade",fadeOptions);
  $w("#bazzSearchTableStrip").show("fade",fadeOptions);
  $w("#copy").show("roll",rollOptions).then( () => {
    $w("#title").show("roll",rollOptions1).then( () => {
      $w("#subCopy").show("roll",rollOptions2);
    }
  )
}

```

```
    });  
  });  
}
```

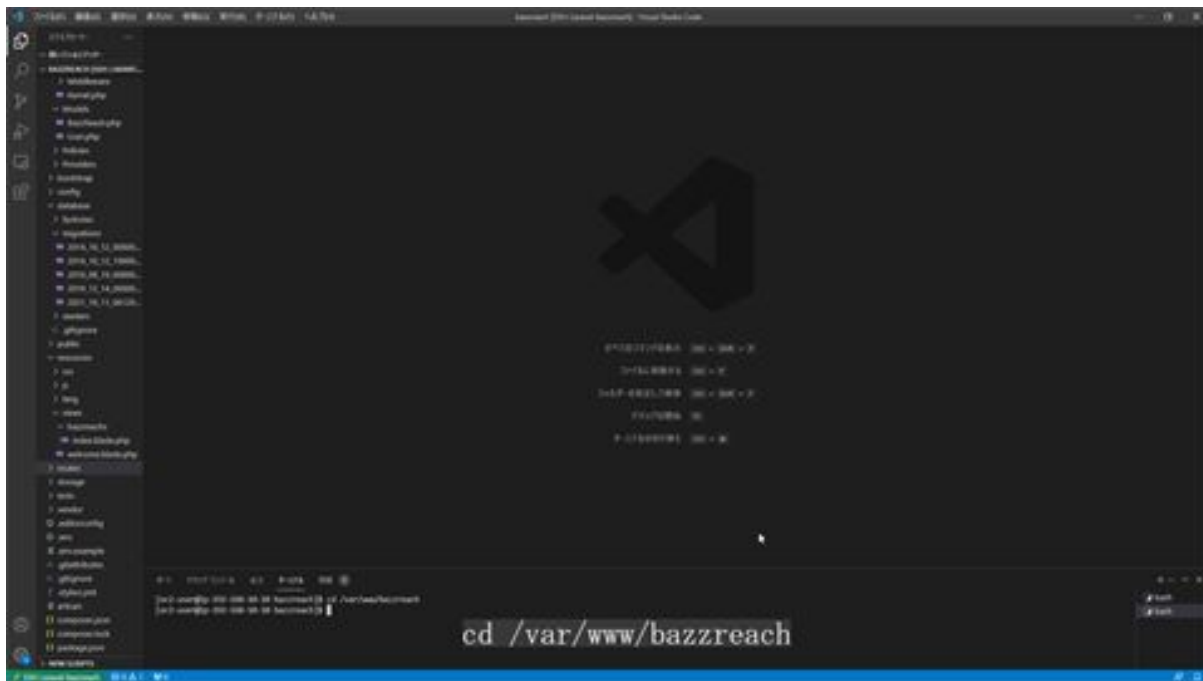
```
function initBazzSearchData (searchData) {  
  searchData.forEach(function(item) {  
    let date = new Date(item.created_at);  
  
    if(item.total_tweets.length > 50){  
      item.tweets_short = item.total_tweets.substr(0, 50);  
      item.tweets_short += "...";  
    }else{  
      item.tweets_short = item.total_tweets;  
    }  
  
    if(item.sentiment_analysis == null){  
      item.sentiment_analysis = "";  
    }  
  
    item.create_passed = moment(date, 'YYYY/MM/DD  
HH:mm:S').fromNow();  
  
  });  
}
```

```
function getRows(startRow, endRow) {  
  return new Promise( (resolve, reject) => {  
    const fetchedDataRows = selectTableData.slice(startRow, endRow);  
    resolve( {  
      pageRows: fetchedDataRows,  
      totalRowsCount: selectTableData.length  
    } );  
  } );  
};  
}
```

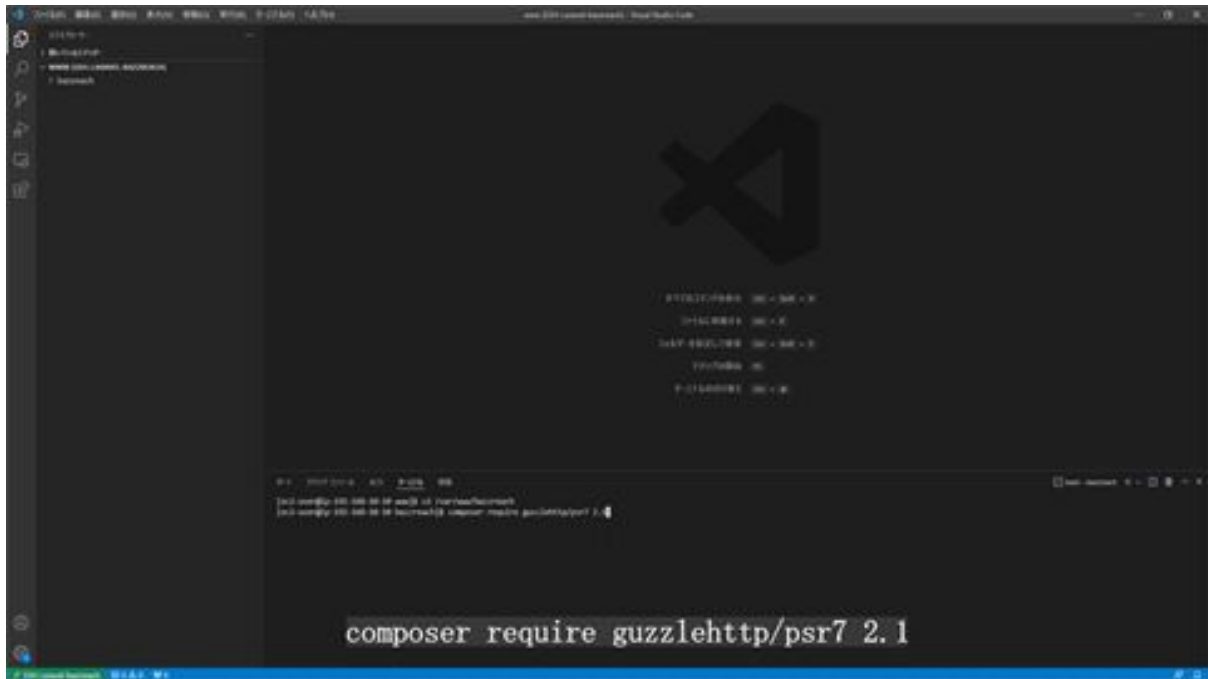
Let's create an analysis function for registered tweets

Now we will analyze the registered tweets. First, we will enable something called Amazon Comprehend for analysis.

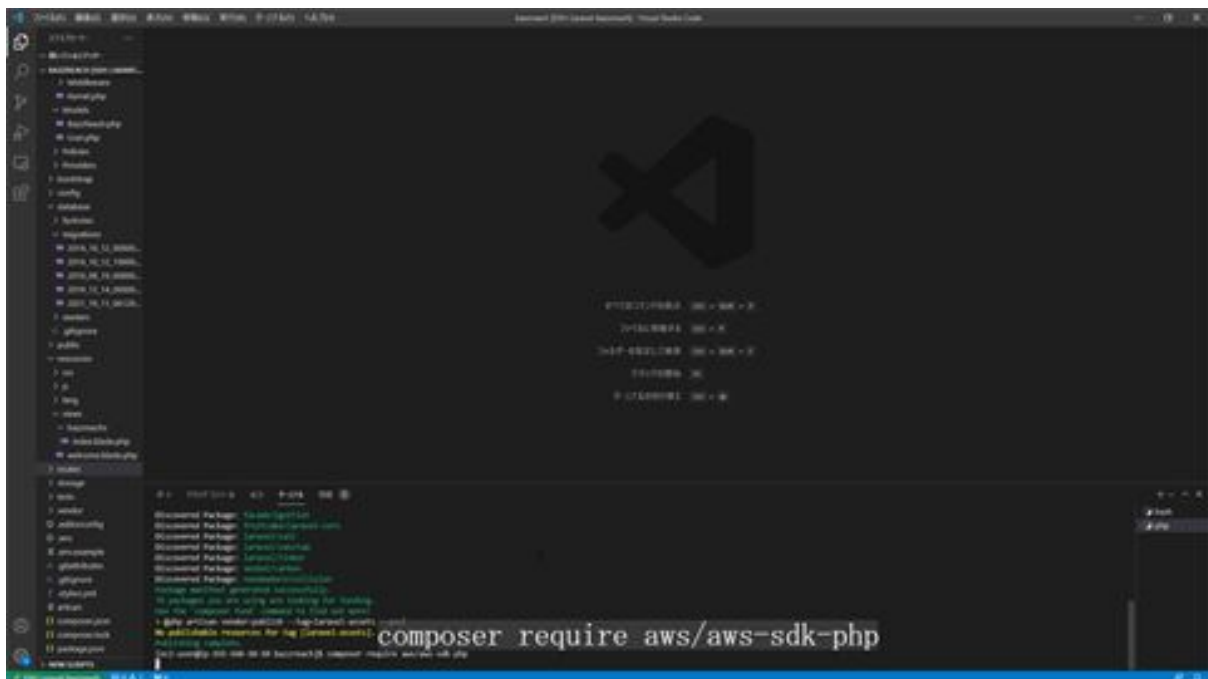
Amazon Comprehend can identify key elements in data such as language, people, and location, and can classify text files by relevant topics. This feature allows you to automatically and accurately detect emotions in your content in real-time. Now let's get to the actual work. To analyze tweets, we will install the AWS package. First, go to the bazzreach folder. `cd /var/www/bazzreach` and navigate to the folder.



Then install `guzzlehttp` and `psr7` required for AWS package installation. `composer require guzzlehttp/psr7 2.1` and run the command to install `guzzlehttp` and `psr7`.

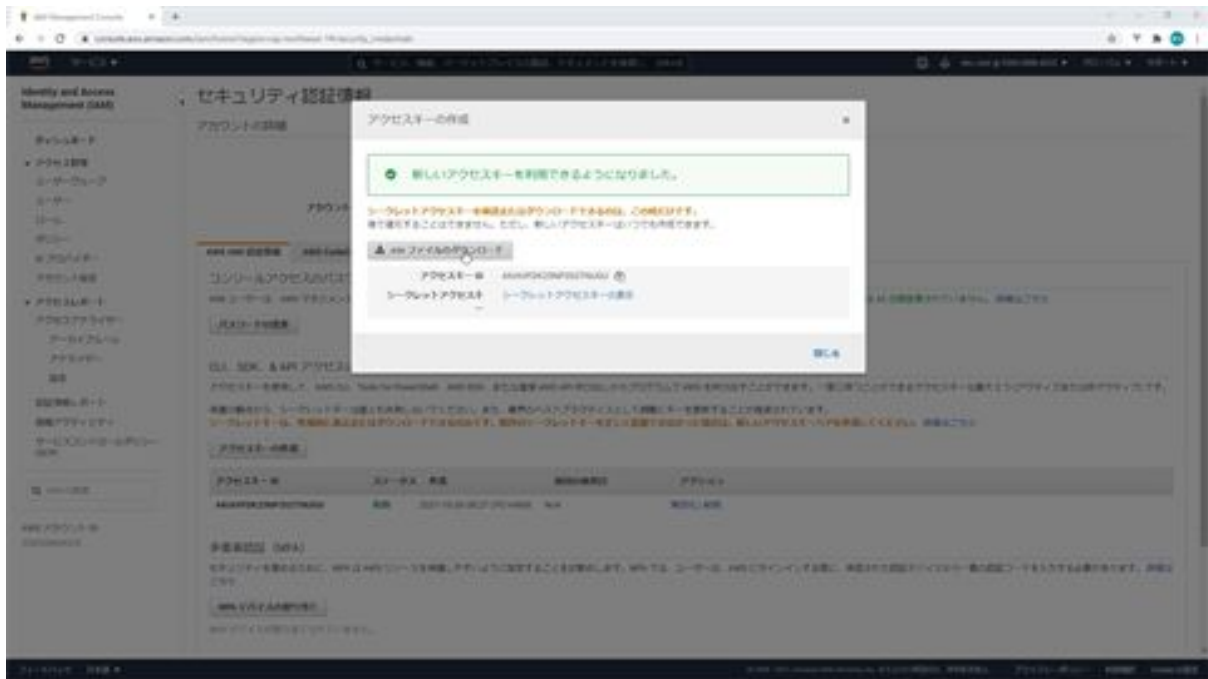


Next, install the AWS packages by running the command `composer require aws/aws-sdk-php`.



After installation is complete, set up the access key and secret access key for the env file: from the AWS management console, click My Security

Credentials in the menu above, click Create Access Key, and download the file.



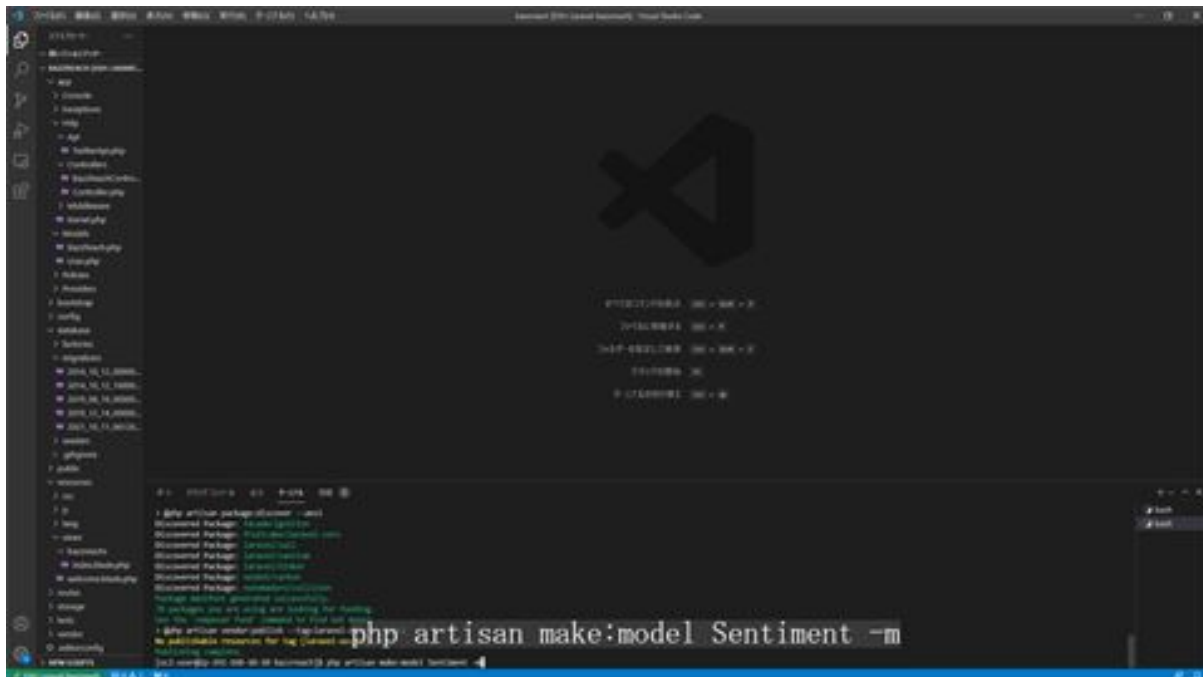
After the download is complete, write the access key and secret access key in the env file. Set `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` in the env file and enter the access key in the `AWS_ACCESS_KEY_ID` section and the secret access key in the `AWS_SECRET_ACCESS_KEY` section.

```
MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

TWITTER_CLIENT_KEY = jk0nuel2DM00bET10ub9XsyE
TWITTER_CLIENT_SECRET = crt7PtYBFzasXet5Da53zIf0z2DqwBuX19tDpVQSEJ31r:
TWITTER_CLIENT_ID_ACCESS_TOKEN = 1336095832047722496-4T1hffCEvCcJkSpmI
TWITTER_CLIENT_ID_ACCESS_TOKEN_SECRET = 6oWVquK78Z3yeKGVY9oqoCusynAaTI

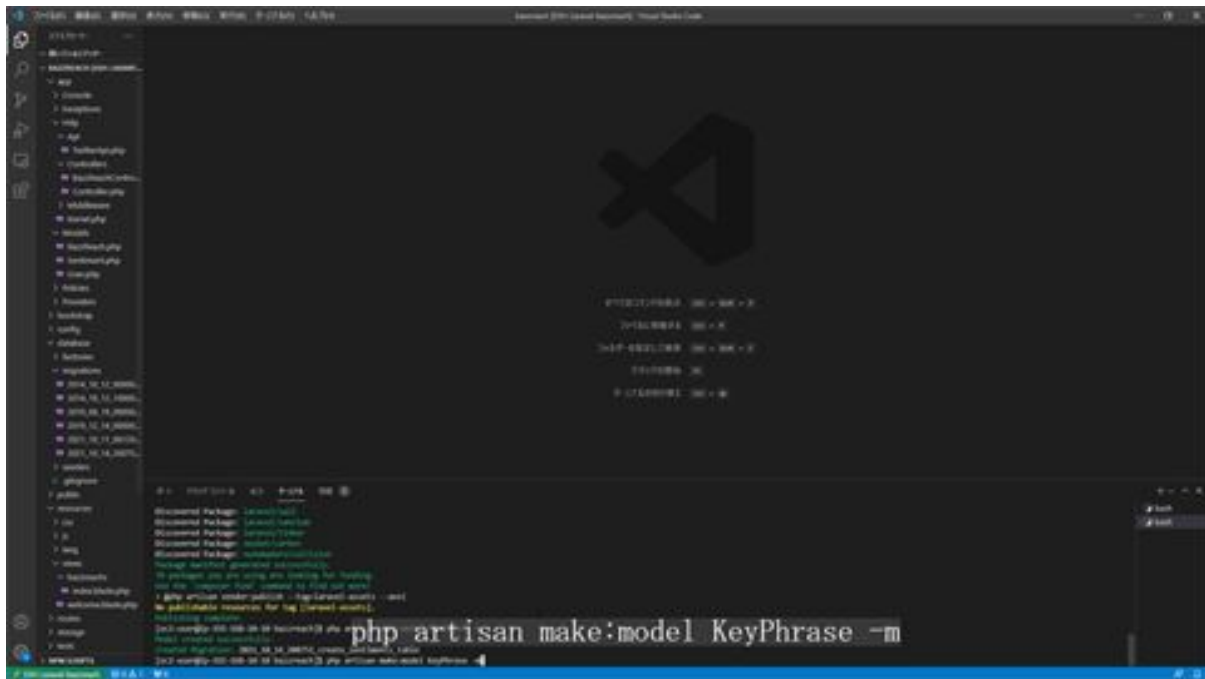
AWS_ACCESS_KEY_ID = AKIAVFDK25NP3DXXDT34
AWS_SECRET_ACCESS_KEY = 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
```

Next, create a model and migration file to store the analysis results. In this case, we will create a model called Sentiment and a migration file called Create a model called KeyPhrase. `php artisan make:model Sentiment -m` where `-m` is the option to create a migration file.



```
php artisan make:model Sentiment -m
```


Key phrases are created in the same way: run `php artisan make:model KeyPhrase -m` to create a KeyPhrase model.



Once the model is created, the `sentiments` and `key_phrases` tables are set up to be associated with the `bazz_reaches` table, and the BazzReach model is modified to be associated with the parent-child relationship.

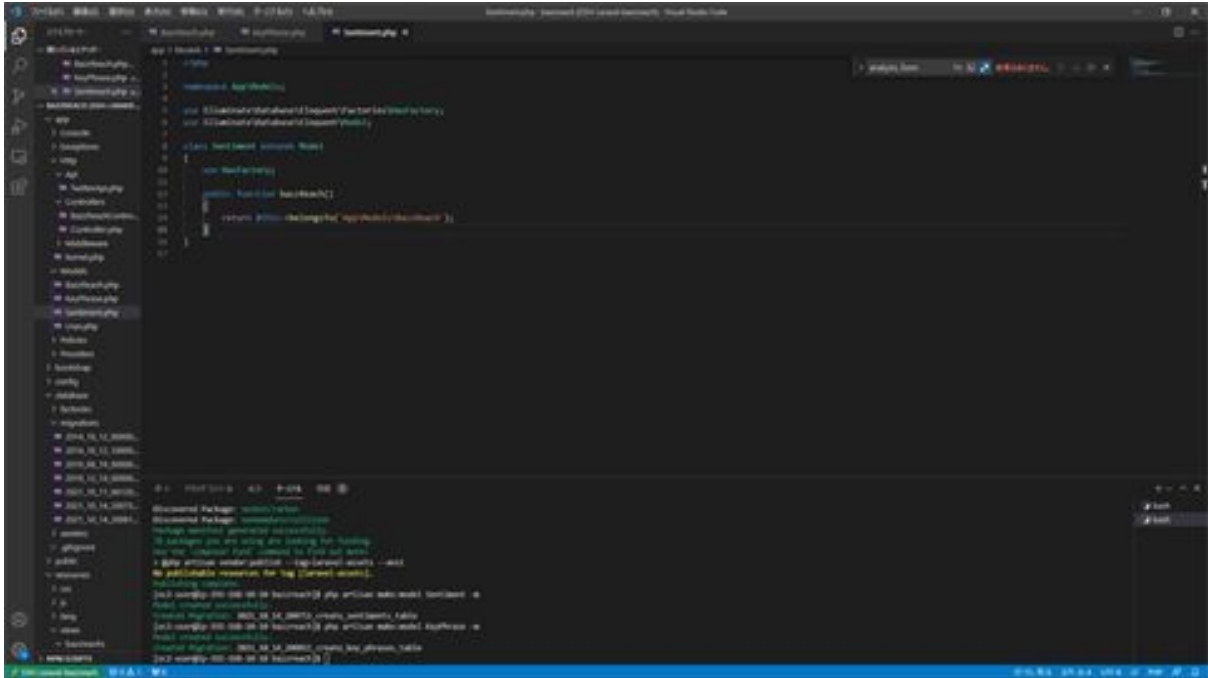
```
class BazzReach {
  hasMany Sentiment;
  hasMany KeyPhrase;
}
```

The screenshot shows a code editor with a file explorer on the left. The main editor displays the BazzReach class with two hasMany associations: Sentiment and KeyPhrase. The code is as follows:

The modification says hasMany, and what this setting means is that there are multiple Sentiment and KeyPhrase records for one BazzReach. Next, open and modify the Sentiment and KeyPhrase models as well.

```
class KeyPhrase {
  belongsTo BazzReach;
}
```

The screenshot shows the same IDE environment, but now the KeyPhrase class is open in the editor. It has a belongsTo association to BazzReach. The code is as follows:



Now that we have set up the parent-child relationships in the table, we have multiple records of Sentiment and KeyPhrase for each record of BazzReach data, and we can retrieve the linked data from the model.

The modification, called "belongsToMany," is a parent-child relationship for the table, where bazzReach is set to have multiple records, and this Sentiment is set to have only one record with bazzReach as the parent, resulting in a one-to-n relationship.

Next, create a migration file to add an entry to the bazz_reaches table. php artisan make:migration add_sentiment_to_bazz_reaches_table --type table=bazz_reaches command.

```
php artisan make:migration add_sentiment_to_bazz_reaches_table --table=bazz_reaches
```

The screenshot shows a terminal window with a dark theme. The command `php artisan make:migration add_sentiment_to_bazz_reaches_table --table=bazz_reaches` has been executed. The terminal output shows the command's success and the creation of the migration file. The file explorer on the left shows the project structure, including the `migrations` directory.

The next step is to modify the migration file you have created. We have created three migration files in total. First, we will modify the migration file `add_sentiment_to_bazz_reaches_table`. Here we will set up the `bazz_reaches` table to add a new entry called `sentiment_analysis`.

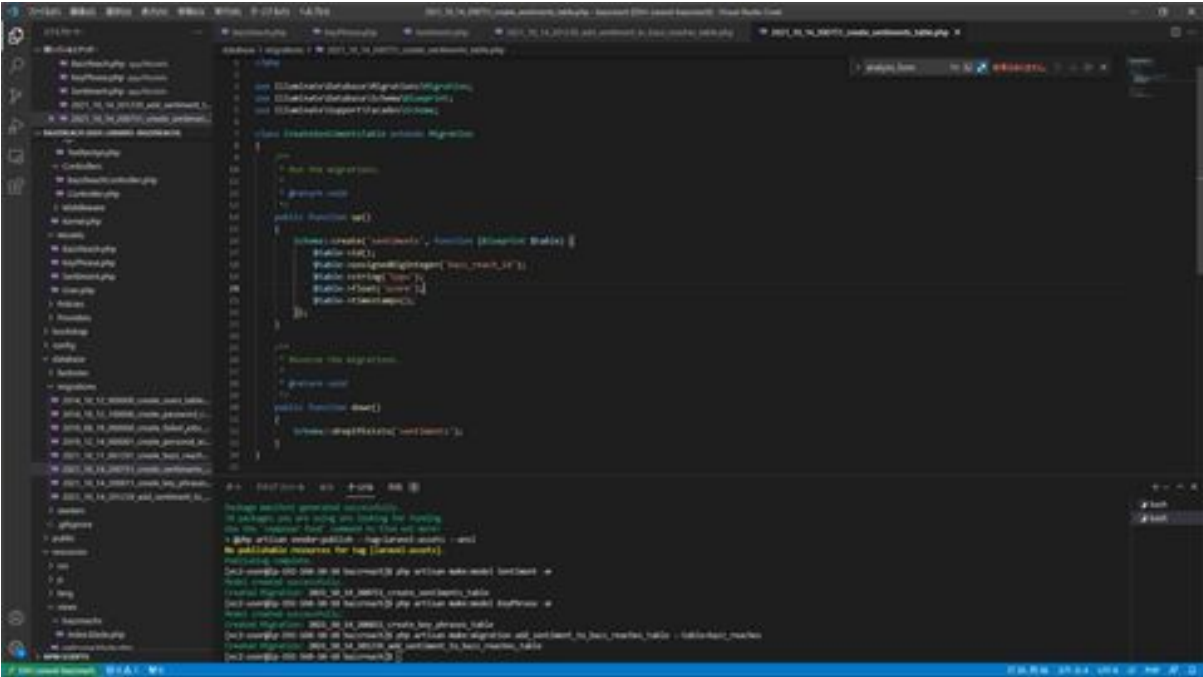
```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddSentimentToBazzReachesTable extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('bazz_reaches', function (Blueprint $table) {
            $table->string('sentiment_analysis', 255);
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('bazz_reaches', function (Blueprint $table) {
            $table->dropColumn('sentiment_analysis');
        });
    }
}
```

The screenshot shows a terminal window with a dark theme. The file explorer on the left shows the project structure, including the `migrations` directory. The terminal output shows the command's success and the creation of the migration file. The file explorer on the left shows the project structure, including the `migrations` directory.

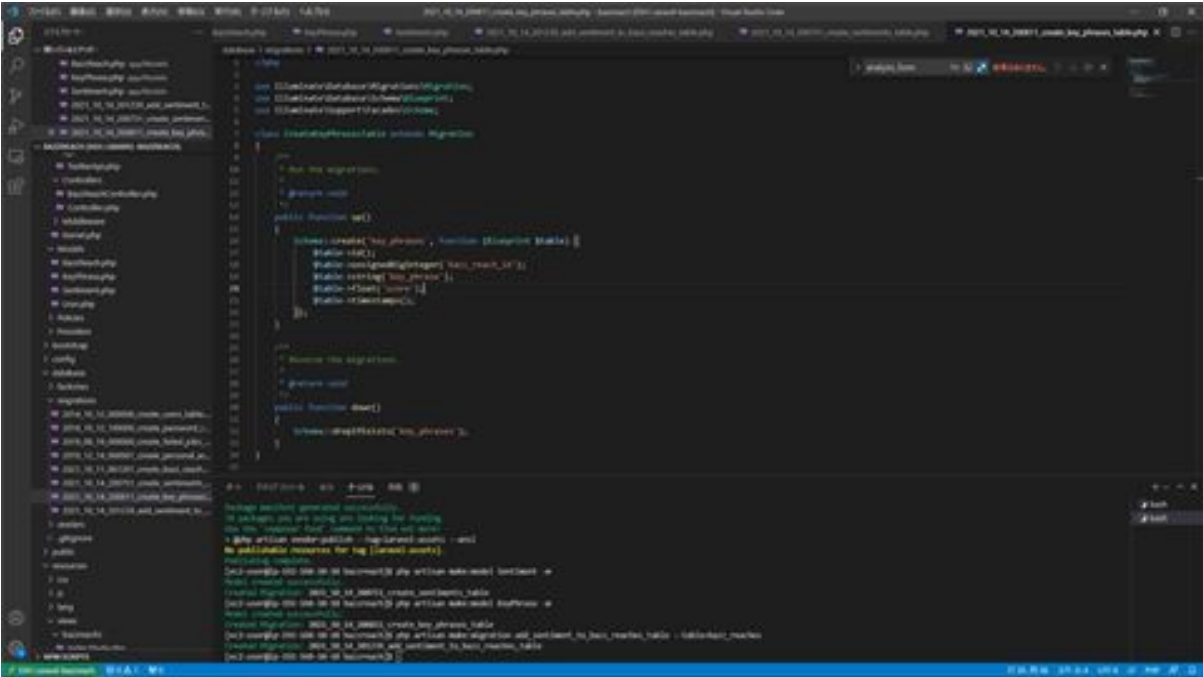
Next, modify the sentiments migration file. New items are added: `bazz_reach_id`, `type`, and `score`.



```
class CreateSentimentsTable < ActiveRecord::Migration[5.2]
  def up
    create_table :sentiments do |t|
      t.string :text
      t.integer :bazz_reach_id
      t.string :type
      t.float :score
    end
  end

  def down
    drop_table :sentiments
  end
end
```

Similarly, the `key_phrases` file is also modified: the `key_phrases` migration adds the entries `bazz_reach_id`, `key_phrase`, and `score`.



```
class CreateKeyPhrasesTable < ActiveRecord::Migration[5.2]
  def up
    create_table :key_phrases do |t|
      t.string :key_phrase
      t.integer :bazz_reach_id
      t.float :score
    end
  end

  def down
    drop_table :key_phrases
  end
end
```



```

class ComprehendApi {
    private ApiClient;

    public ComprehendApi(ApiClient apiClient) {
        this.ApiClient = apiClient;
    }

    // ツイート分析
    public List<Tweet> analyzeTweet(String tweet) {
        Result<List<Tweet>> result = this.ApiClient.execute(new GetTweetsRequest());
        return result.getData();
    }

    // ツイート感情分析
    public List<Tweet> analyzeTweetSentiment(String tweet) {
        Result<List<Tweet>> result = this.ApiClient.execute(new GetTweetsRequest());
        return result.getData();
    }

    // ツイートキーワード抽出
    public List<Tweet> analyzeTweetKeyPhrases(String tweet) {
        Result<List<Tweet>> result = this.ApiClient.execute(new GetTweetsRequest());
        return result.getData();
    }
}

```

コンストラクタはインスタンス作成時つまり
NEWで呼び出された時に呼ばれる処理となります

To explain ComprehendApi, we first perform the initial setup for searching by constructs, analyze sentiment with ComprehendApi's tweetSentiment method, and analyze key phrases with tweetKeyPhrases.

```

class ComprehendApi {
    private ApiClient;

    public ComprehendApi(ApiClient apiClient) {
        this.ApiClient = apiClient;
    }

    // ツイート分析
    public List<Tweet> analyzeTweet(String tweet) {
        Result<List<Tweet>> result = this.ApiClient.execute(new GetTweetsRequest());
        return result.getData();
    }

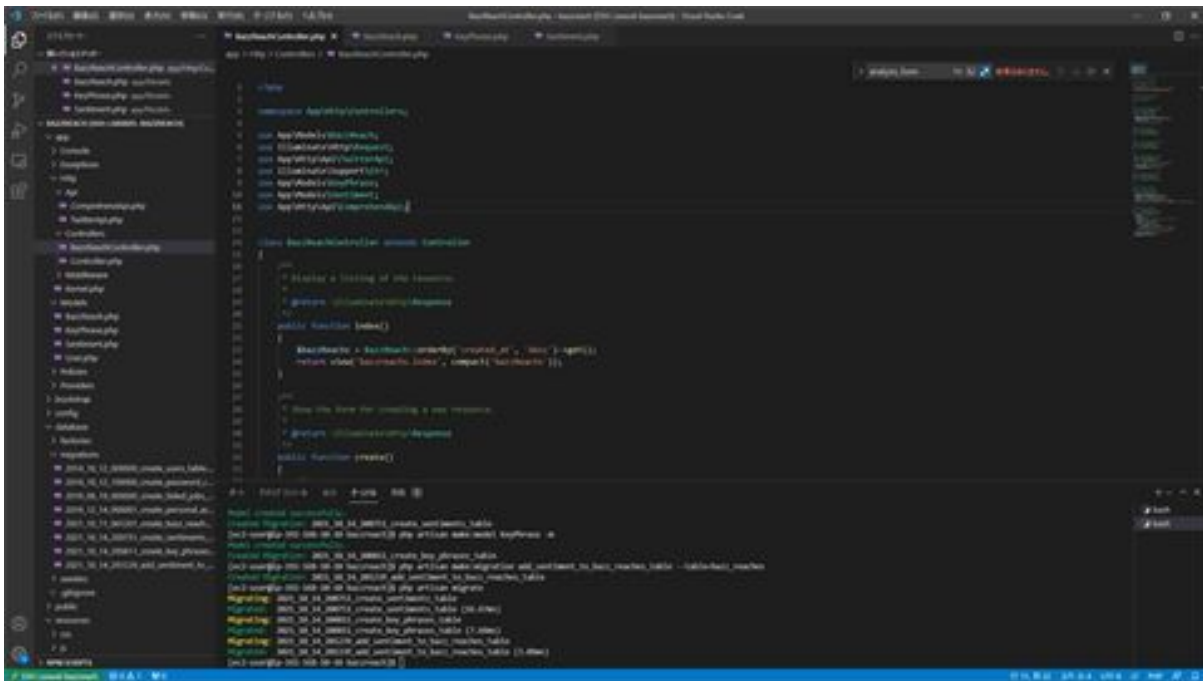
    // ツイート感情分析
    public List<Tweet> analyzeTweetSentiment(String tweet) {
        Result<List<Tweet>> result = this.ApiClient.execute(new GetTweetsRequest());
        return result.getData();
    }

    // ツイートキーワード抽出
    public List<Tweet> analyzeTweetKeyPhrases(String tweet) {
        Result<List<Tweet>> result = this.ApiClient.execute(new GetTweetsRequest());
        return result.getData();
    }
}

```

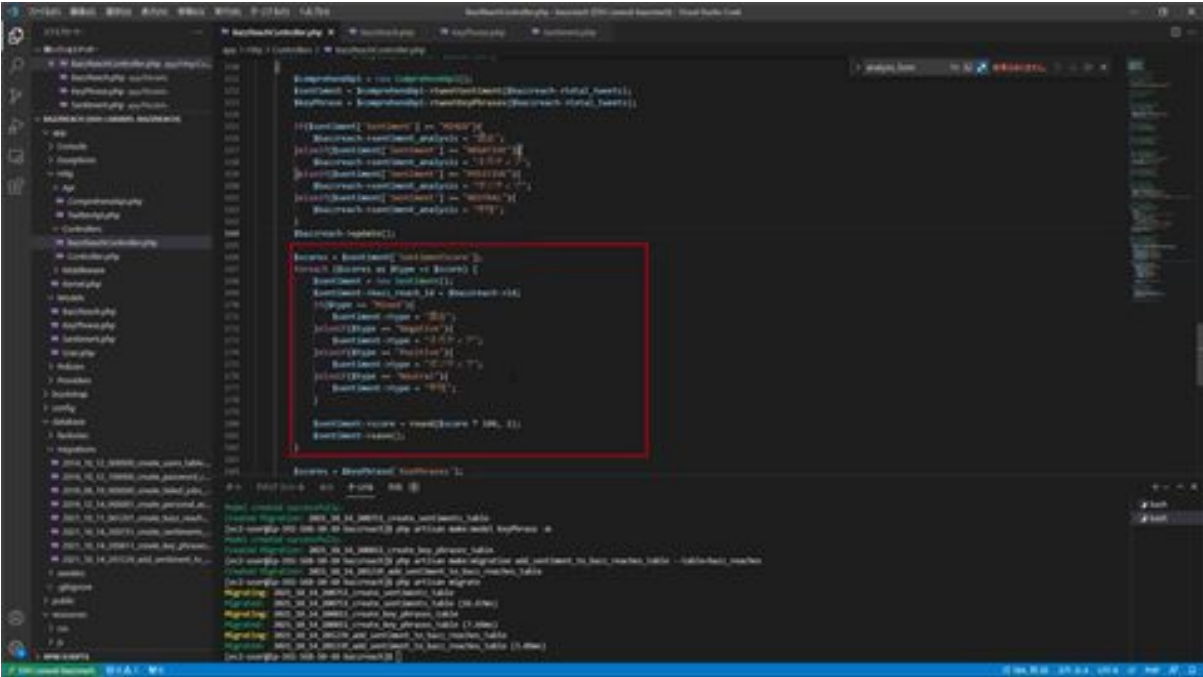

Both use the AWS API, and simply pass the language and string as arguments to be analyzed and the values returned. Both APIs are limited to 5000 bytes of the string, so the string is processed to 5000 bytes and the processed string is passed for analysis.

Now we will next modify BazzReachController. First, we will modify the KeyPhrase model, Sentiment model, and Import ComprehendApi so that they can be used.



Next, create an analysis method in BazzReachController for the analysis action method.

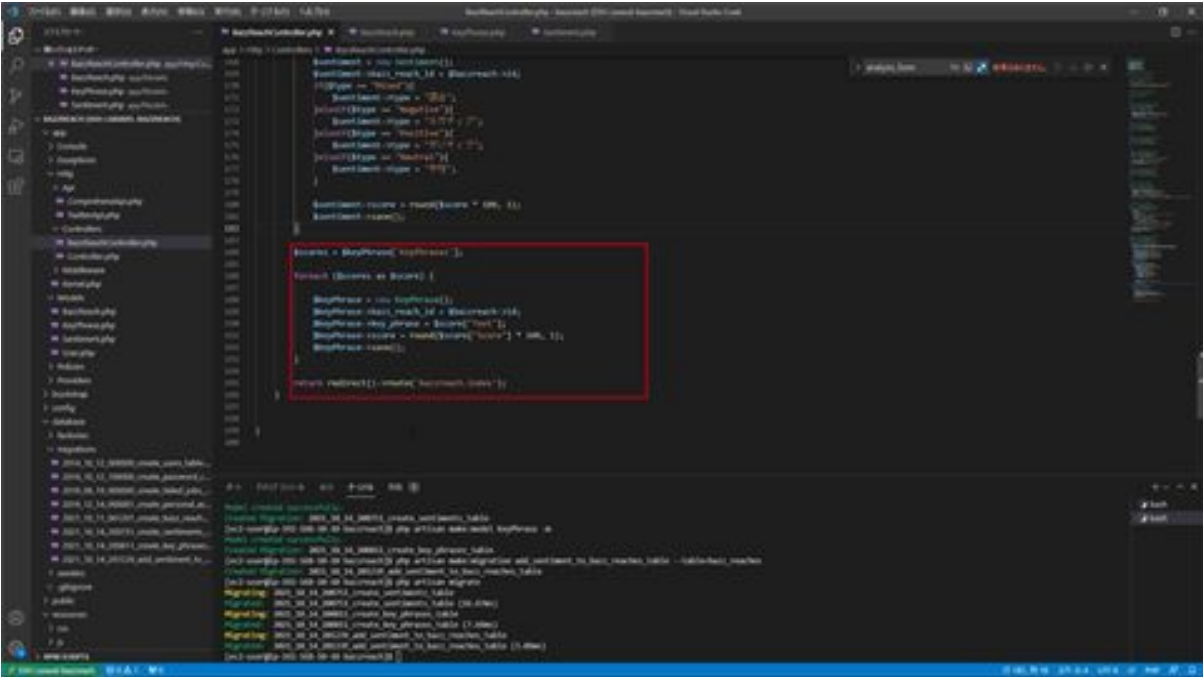
The results of the sentiment analysis are processed using the BazzReach model to update sentiment_analysis items.



```
function processSentimentAnalysisResults() {
    // ...
    BazzReach.updateItem({
        item: {
            id: 'sentiment_analysis_' + item.id,
            score: item.score,
            sentiment: item.sentiment,
            reach: item.reach,
            // ...
        },
        // ...
    });
}

// ...
BazzReach.updateItem({
    item: {
        id: 'sentiment_analysis_' + item.id,
        score: item.score,
        sentiment: item.sentiment,
        reach: item.reach,
        // ...
    },
    // ...
});
```

Next, the results of the sentiment analysis scores are registered using the Sentiment model in a loop process.



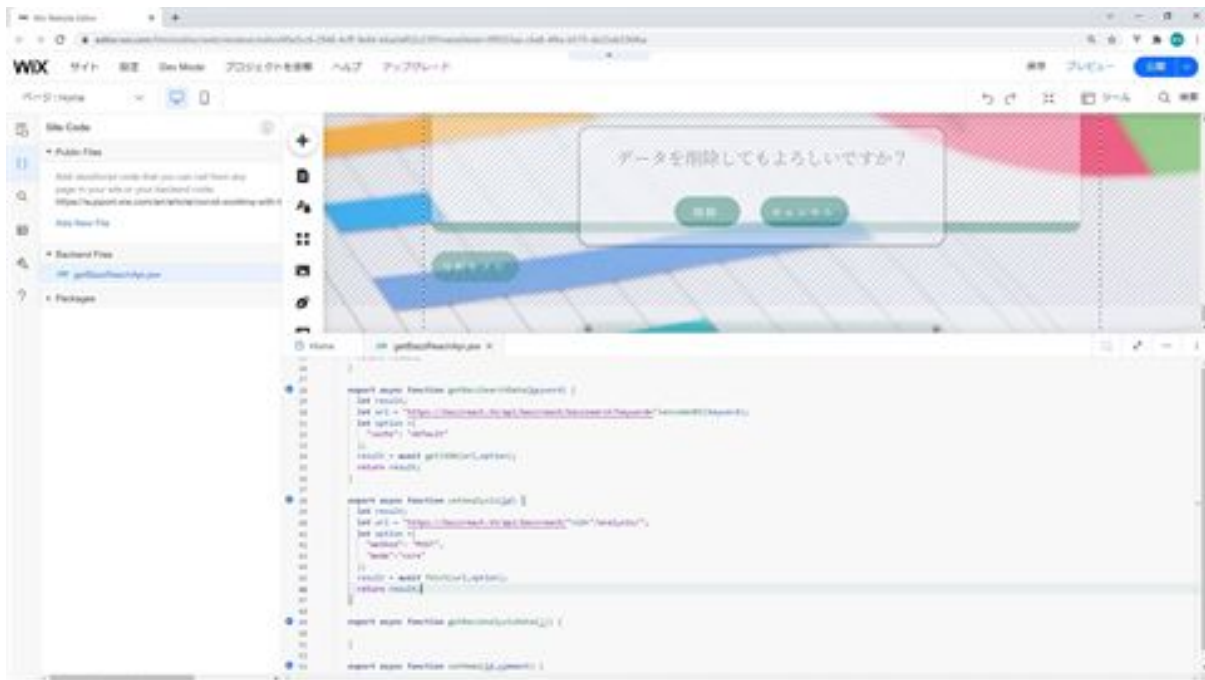
```
function registerSentimentScores() {
    // ...
    for (let i = 0; i < sentimentScores.length; i++) {
        // ...
        Sentiment.registerScore({
            score: sentimentScores[i].score,
            sentiment: sentimentScores[i].sentiment,
            // ...
        });
    }
}

// ...
for (let i = 0; i < sentimentScores.length; i++) {
    // ...
    Sentiment.registerScore({
        score: sentimentScores[i].score,
        sentiment: sentimentScores[i].sentiment,
        // ...
    });
}
```

Similarly, key phrase score results are registered using the KeyPhrase model in a loop process. Once this has been done, the next step is to modify the screen.

Also, as for the variable names of the arguments of the ANALYSIS method, this part must match the variable names in the URI and the method argument names.

Now we will modify the screen side. First, we will add processing to the setAnalysis function in the getBazzReachApi Web module.

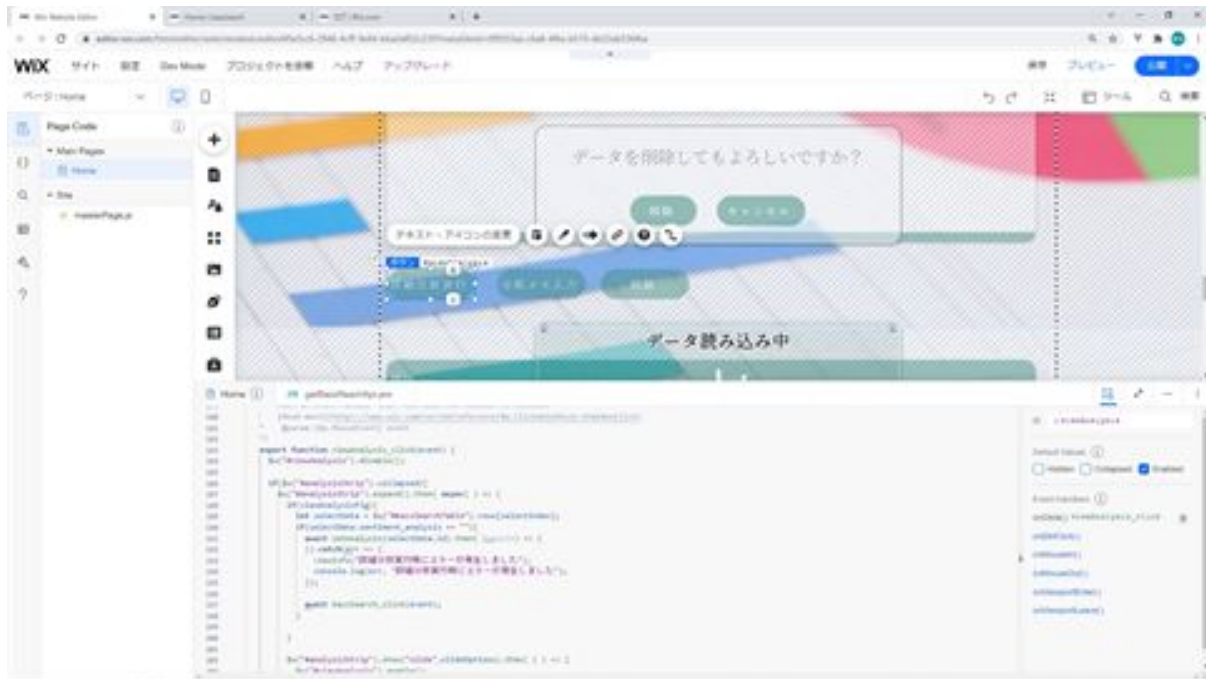


To explain the process of this function, it accesses the analysis method created in Laravel and performs a detailed analysis. the option specifies POST as the method so that Laravel can receive it as a POST submission method.

The reason why the sending method is POST is that it is configured in Laravel routing, and the id is added to the URL because Laravel routing is configured to pass the primary key of the table in the URL.

Finally, we use fetch to send POST, specifying core in the mode specification and allowing cross-origin requests so that we can access the API on the Laravel side.

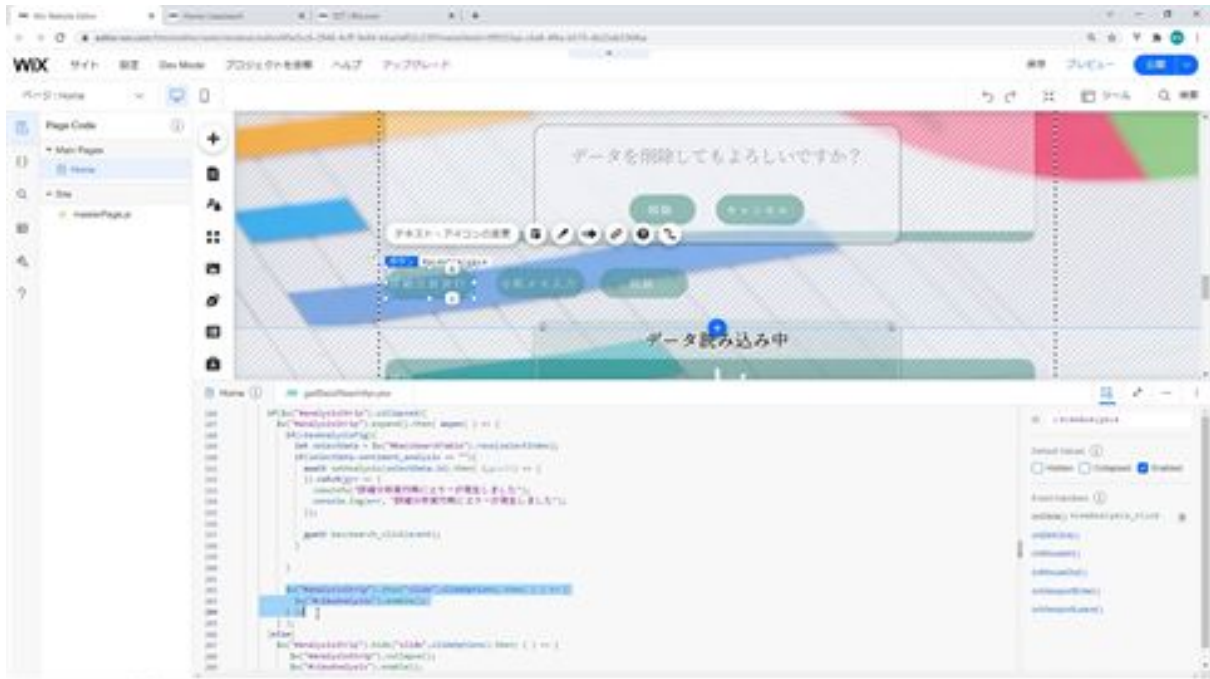
Next, on the Home side, add processing to the onClick event function of the Execute Detailed Analysis button.



To explain the processing of this function, first, the ID deactivates the viewAnalysis detailed analysis execution button to prevent double submissions.

Next, when the strip for detailed analysis with ID analysisStrip is folded, call the setAnalysis function you created to perform the detailed analysis. When a detailed analysis is performed, the sentiment_analysis item is updated, so the sentiment_analysis item is used to determine if a detailed analysis is being performed.

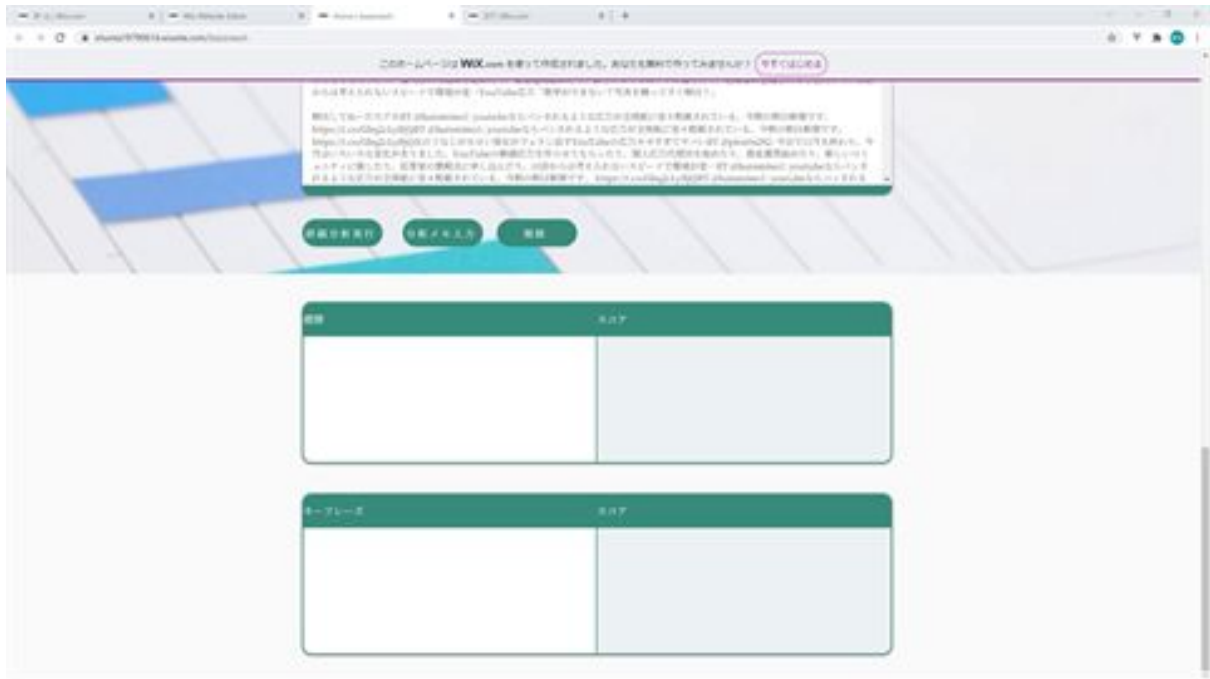
After the detailed analysis is complete, the bazzSearch_click function is called to search the data and update the table information.



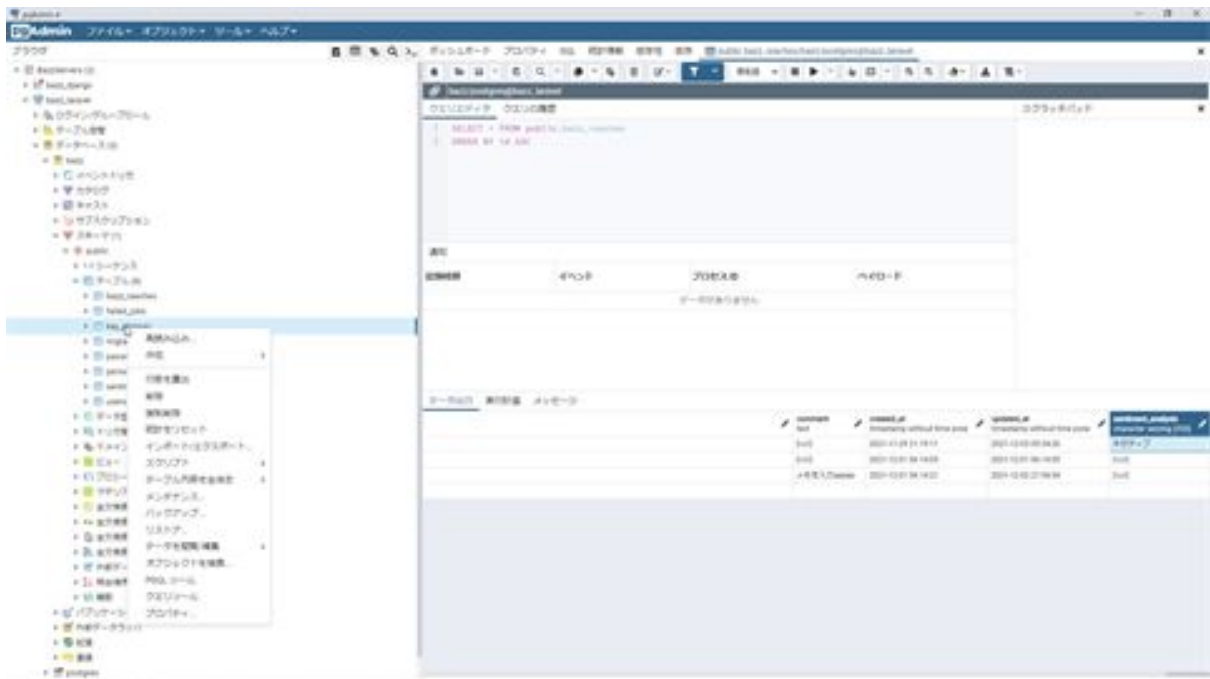
Finally, the animation is used to display the strip whose ID is analysisStrip and activate the button to perform a detailed analysis.



If it is not collapsed and displayed, the animation hides the strip for detailed analysis and activates the button to perform detailed analysis.



Next, open pgAdmin to see if the data has been registered.



BazzReach.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class BazzReach extends Model
{
    use HasFactory;

    public function sentiment()
    {
        return $this->hasMany('App\Models\Sentiment');
    }

    public function keyPhrase()
    {
        return $this->hasMany('App\Models\KeyPhrase');
    }
}
```

Sentiment.php

```
<?php
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Sentiment extends Model
{
    use HasFactory;

    public function bazzReach ()
    {
        return $this->belongsTo('App\Models\BazzReach');
    }
}
```

KeyPhrase.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class KeyPhrase extends Model
{
    use HasFactory;

    public function bazzReach ()
    {
        return $this->belongsTo('App\Models\BazzReach');
    }
}
```

}

add_sentiment_to_bazz_reaches_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddSentimentToBazzReachesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::table('bazz_reaches', function (Blueprint $table) {
            $table->string('sentiment_analysis')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('bazz_reaches', function (Blueprint $table) {
```

```
        $table->dropColumn('sentiment_analysis');
    });
}
}
```

create_sentiments_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateSentimentsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('sentiments', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('bazz_reach_id');
            $table->string('type');
            $table->float('score');
            $table->timestamps();
        });
    }
}
```

```
/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('sentiments');
}
}
```

create_key_phrases_table.php

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateKeyPhrasesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('key_phrases', function (Blueprint $table) {
            $table->id();
        });
    }
}
```

```
        $table->unsignedBigInteger('bazz_reach_id');
        $table->string('key_phrase');
        $table->float('score');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('key_phrases');
}
}
```

ComprehendApi.php

```
<?php

namespace App\Http\Api;

use Illuminate\Http\Request;
use Aws\Comprehend\ComprehendClient;

class ComprehendApi
{

    private $client;

    public function __construct()
    {
        $this->client = new ComprehendClient([
            'credentials' => [
                'key' => env('AWS_ACCESS_KEY_ID'),
                'secret' => env('AWS_SECRET_ACCESS_KEY')
            ],
            'region' => 'ap-northeast-1',
            'version' => '2017-11-27'
        ]);
    }

    // Tweet sentiment analysis
    public function tweetSentiment(String $total_tweets)
    {
```



```

$total_tweets = mb_strcut($total_tweets, 0 , 5000, "UTF-8");
$result = $this->client->detectSentiment([
    'LanguageCode' => 'ja',
    'Text' => $total_tweets
]);

return $result;
}

// keyphrase analysis
public function tweetKeyPhrases(String $total_tweets)
{
    $total_tweets = mb_strcut($total_tweets, 0 , 5000, "UTF-8");
    $result = $this->client->detectKeyPhrases([
        'LanguageCode' => 'ja',
        'Text' => $total_tweets
    ]);

    return $result;
}
}

```

BazzReachController.php

```

<?php

namespace App\Http\Controllers;

use App\Models\BazzReach;
use Illuminate\Http\Request;
use App\Http\Api\TwitterApi;
use App\Models\KeyPhrase;

```

```
use App\Models\Sentiment;
use App\Http\Api\ComprehendApi;

class BazzReachController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
```

```

    $ bazzreach = new BazzReach ();
    $ bazzreach->search_word = $request->input('search_word');
    $ bazzreach->total_tweets = $request->input('total_tweets');
    $ bazzreach->save();

    return http_response_code();
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function show( BazzReach $ bazzReach )
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function edit( BazzReach $ bazzReach )
{
    //
}

/**
 * Update the specified resource in storage.
 *

```

```

* @param \Illuminate\Http\Request $request
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function update(Request $request, BazzReach $ bazzReach )
{
    $ bazzreach->comment = $request->input('comment');
    $ bazzreach->update();
    return http_response_code();
}

/**
 * Remove the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function destroy( BazzReach $ bazzReach )
{
    //
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function search(Request $request)
{
    $searchWord = $request->input('search_word');
    $twitterApi = new TwitterApi();
    $totalTweets = $twitterApi->serachTweets($searchWord);
}

```

```

    $dataArray = [$totalTweets];
    $data = response()->json($dataArray);
    return $data;
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function bazzsearch (Request $request)
{
    $keyword = $request->input('keyword');
    $bazzReachs;

    if(empty($keyword)){
        $ bazzReachs = BazzReach ::orderBy('created_at', 'desc')->get();
    }else{
        $query = BazzReach::query();
        $keywords = explode(" ", $keyword);

        foreach ($keywords as $Key => $Value) {
            $query->orWhere('search_word', 'ilike', '%'.$Value.'%');
        }
        $bazzReachs = $query->orderBy('created_at', 'desc')->get();
    }
    $dataArray = [$bazzReachs];
    $data = response()->json($dataArray);
    return $data;
}

/**
 * Analysis the specified resource from storage.

```

```

*
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function analysis( BazzReach $ bazzreach )
{
    $comprehendApi = new ComprehendApi();
    $sentiment = $comprehendApi->tweetSentiment($ bazzreach-
>total_tweets);
    $keyPhrase = $comprehendApi->tweetKeyPhrases($ bazzreach-
>total_tweets);

    if($sentiment['Sentiment'] == "MIXED"){
        $bazzreach->sentiment_analysis = "mixing";
    }elseif($sentiment['Sentiment'] == "NEGATIVE"){
        $bazzreach->sentiment_analysis = "negative";
    }elseif($sentiment['Sentiment'] == "POSITIVE"){
        $bazzreach->sentiment_analysis = "positive";
    }elseif($sentiment['Sentiment'] == "NEUTRAL"){
        $bazzreach->sentiment_analysis = "neutrality";
    }
    $ bazzreach- >update();

    $scores = $sentiment['SentimentScore'];
    foreach ($scores as $type => $score) {
        $sentiment = new Sentiment();
        $sentiment->bazz_reach_id = $ bazzreach- >id;
        if($type == "Mixed"){
            $sentiment->type = "mixing";
        }elseif($type == "Negative"){
            $sentiment->type = "negative";
        }elseif($type == "Positive"){

```

```
        $sentiment->type = "positive";
    }elseif($type == "Neutral"){
        $sentiment->type = "neutrality";
    }

    $sentiment->score = round($score * 100, 1);
    $sentiment->save();
}

$scores = $keyPhrase['KeyPhrases'];

foreach ($scores as $score) {

    $keyPhrase = new KeyPhrase();
    $keyPhrase->bazz_reach_id = $bazzreach->id;
    $keyPhrase->key_phrase = $score["Text"];
    $keyPhrase->score = round($score["Score"] * 100, 1);
    $keyPhrase->save();
}

return http_response_code();
}
}
```

getBazzReachApi.js

```
import { fetch,getJSON} from 'wix-fetch';

export async function getSearchData(searchWord) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/search?
search_word="+encodeURIComponent(searchWord);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setBazzData (search_word,total_tweets) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/";
  let option ={
    "method": "POST",
    "mode":"core",
    "headers": {
      "Content-Type": "application/x-www-form-urlencoded"
    },
    "body": "search_word="+search_word+"&total_tweets="+total_tweets
  };
  result = await fetch(url,option);
  return result;
}
```



```
export async function getBazzSearchData (keyword) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/bazzsearch?
keyword="+encodeURIComponent(keyword);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setAnalysis(id) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/"+id+"/analysis/";
  let option ={
    "method": "POST",
    "mode":"core"
  };
  result = await fetch(url,option);
  return result;
}

export async function getBazzAnalysisData(id) {

}

export async function setMemo(id,comment) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/"+id+"/";
  let option ={
    "method": "PATCH",
    "mode":"core",
    "headers": {
```

```
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "body": "comment="+comment
};
result = await fetch(url,option);
return result;
}

export async function delBazzData(id) {
}
```

home.js

```
import moment from 'moment/moment';
import 'moment/locale/ja';
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

moment.locale('ja');

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};

let rollOptions1 = {
```

```

    "duration": 1500,
    "delay": 0,
    "direction": "left"
};

let rollOptions2 = {
    "duration": 1000,
    "delay": 0,
    "direction": "left"
};

let fadeOptions = {
    "duration": 500,
    "delay": 0
};

$w.onReady(function () {
    getBazzSearchData ("").then( (result) => {
        init();
        initBazzSearchData(result[0]);
        $w("#bazzSearchDataCount").text = String(result[0].length);
        selectTableData = result[0];
        $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
        $w("# bazzSearchTable ").dataFetcher = getRows;
    }).catch(err => {
        viewInfo("An error occurred during data loading");
        console.log(err, "An error occurred during data loading");
    });
});

/**
 *      Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  getSearchData(search_word).then( (result) => {

    if($w("#saveTweetStrip").collapsed){
      $w("#saveTweetStrip").expand().then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }else{
      $w("#saveTweetStrip").hide("slide",slideOptions).then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }
  }).catch(err => {
    viewInfo("An error occurred when searching for tweets");
    console.log(err, "An error occurred when searching for tweets");
    $w("#search").enable();
  });
}
```

```

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function saveTweet_click(event) {
  $w("#saveTweet").disable();

  setBazzData($w("#viewSearchWord").text,$w("#viewSearchData").value).t
  hen( (result) => {
    bazzSearch_click(event);
    viewInfo("Saved tweet");
    $w("#saveTweetStrip").hide("slide",slideOptions).then( ( ) => {
      $w("#viewSearchWord").text = "";
      $w("#viewSearchData").value = "";
      $w("#saveTweetStrip").collapse();
      $w("#saveTweet").enable();
    } );
  }).catch(err => {
    viewInfo("An error occurred when saving the tweet");
    console.log(err, "An error occurred when saving the tweet");
    $w("#saveTweet").enable();
  });
}

```

```

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */

```

```

export function bazzSearch_click(event) {
  $w("# bazzSearch ").disable();
  let keyword = $w("#keyword").value;
  getBazzSearchData (keyword).then( (result) => {
    initBazzSearchData(result[0]);
    $w("#bazzSearchDataCount").text = String(result[0].length);
    selectTableData = result[0];
    $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
    $w("# bazzSearchTable ").dataFetcher = getRows;
    $w("# bazzSearch ").enable();
  }).catch(err => {
    viewInfo("An error occurred during data retrieval");
    console.log(err, "An error occurred during data retrieval");
    $w("# bazzSearch ").enable();
  });
}

/**
 *    Adds an event handler that runs when a table row is selected.
 *    [Read more]
 *    (https://www.wix.com/corvid/reference/\$w.Table.html#onRowSelect)
 *    @param {$w.TableRowEvent} event
 */
export function bazzSearchTable_rowSelect(event) {
  selectIndex = event.rowIndex;
  viewAnalysisFlg = true;
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  let total_tweets = selectData.total_tweets;

  if($w("#selectBazzDataStrip").collapsed){
    $w("#selectBazzDataStrip").expand().then( ( ) => {

```

```

    $w("#selectKeyword").text = selectData.search_word;
    $w("# selectBazzData ").value = total_tweets;
    $w("#memo").value = selectData.comment;
    $w("#selectBazzDataStrip").show("slide",slideOptions);
    } );
} else {
    $w("#selectBazzDataStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#selectKeyword").text = selectData.search_word;
        $w("# selectBazzData ").value = total_tweets;
        $w("#memo").value = selectData.comment;
        $w("#selectBazzDataStrip").show("slide",slideOptions);
    } );
}

if(!$w("#analysisStrip").hidden){
    $w("#analysisStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#analysisStrip").collapse();
    } );
}

if(!$w("#memoStrip").hidden){
    $w("#memoStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#memoStrip").collapse();
    } );
}
}
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */

```



```

export function viewAnalysis_click(event) {
  $w("#viewAnalysis").disable();

  if($w("#analysisStrip").collapsed){
    $w("#analysisStrip").expand().then( async ( ) => {
      if( viewAnalysisFlg ){
        let selectData = $w("# bazzSearchTable ").rows[selectIndex];
        if(selectData.sentiment_analysis == ""){
          await setAnalysis(selectData.id).then( (result) => {
            }).catch(err => {
              viewInfo("An error occurred while performing detailed analysis");
              console.log(err, "An error occurred while performing detailed
analysis");
            });

            await bazzSearch_click(event);
          }

        }

        $w("#analysisStrip").show("slide",slideOptions).then( ( ) => {
          $w("#viewAnalysis").enable();
        } );
      } );
    } else{
      $w("#analysisStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#analysisStrip").collapse();
        $w("#viewAnalysis").enable();
      } );
    }
  }
}

/**

```

* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function viewMemoInput_click(event) {
  $w("#viewMemoInput").disable();
  if($w("#memoStrip").collapsed){
    $w("#memoStrip").expand().then( () => {
      $w("#memoStrip").show("slide",slideOptions).then( () => {
        $w("#viewMemoInput").enable();
      } );
    } );
  }else{
    $w("#memoStrip").hide("slide",slideOptions).then( () => {
      $w("#memoStrip").collapse();
      $w("#viewMemoInput").enable();
    } );
  }
}
```

```
/**
```

* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function saveMemo_click(event) {
  $w("#saveMemo").disable();
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  setMemo(selectData.id,$w("#memo").value).then( (result) => {
    bazzSearch_click(event);
  } );
}
```

```

viewInfo("Notes saved");
$w("#memoStrip").hide("slide",slideOptions).then( () => {
  $w("#memoStrip").collapse();
  $w("#saveMemo").enable();
} );
}).catch(err => {
  viewInfo("An error occurred when saving the note");
  console.log(err, "An error occurred when saving the note");
  $w("#saveMemo").enable();
});
}

/**
 *   Adds an event handler that runs when the element is clicked.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function deleteConfirm_click(event) {

}

/**
 *   Adds an event handler that runs when the element is clicked.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function cancel_click(event) {

}

/**
 *   Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function delete_click(event) {

}

function viewInfo(message){

  let puffInfoOptions = {
    "duration": 500,
    "delay": 2000
  };

  $w("#confirmText").text = message;
  $w("#confirmBtn").hide();
  $w("#confirm").show("puff",puffOptions).then( () => {
    $w("#confirm").hide("puff",puffInfoOptions)
  } );
}

function init(){
  $w("#preLoader").hide("fade",fadeOptions);
  $w("#searchWordStrip").show("fade",fadeOptions);
  $w("#bazzSearchStrip").show("fade",fadeOptions);
  $w("#bazzSearchTableStrip").show("fade",fadeOptions);
  $w("#copy").show("roll",rollOptions).then( () => {
    $w("#title").show("roll",rollOptions1).then( () => {
      $w("#subCopy").show("roll",rollOptions2);
    } );
  } );
}
```

```

function initBazzSearchData (searchData) {
  searchData.forEach(function(item) {
    let date = new Date(item.created_at);

    if(item.total_tweets.length > 50){
      item.tweets_short = item.total_tweets.substr(0, 50);
      item.tweets_short += "...";
    }else{
      item.tweets_short = item.total_tweets;
    }

    if(item.sentiment_analysis == null){
      item.sentiment_analysis = "";
    }

    item.create_passed = moment(date, 'YYYY/MM/DD
HH:mm:S').fromNow();

  });
}

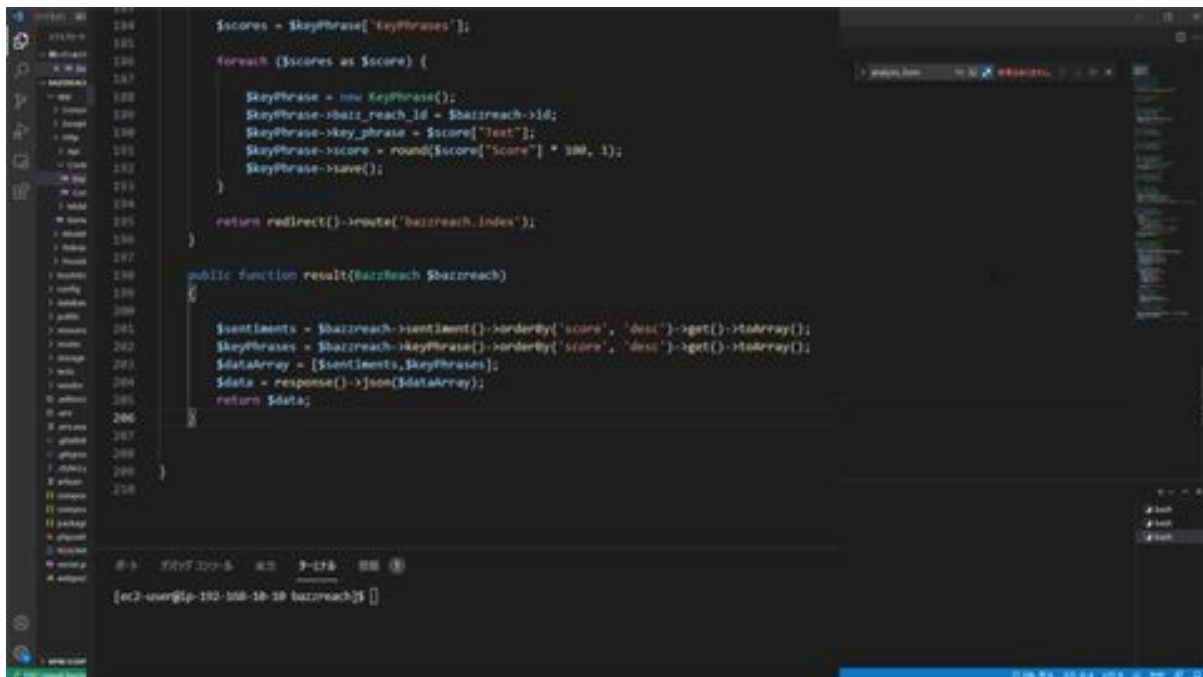
function getRows(startRow, endRow) {
  return new Promise( (resolve, reject) => {
    const fetchedDataRows = selectTableData.slice(startRow, endRow);
    resolve( {
      pageRows: fetchedDataRows,
      totalRowCount: selectTableData.length
    } );
  } );
}

```

Let's view the analysis results of the registered tweets

Next, we will display the results of the analysis. First, we will add a process to retrieve the analysis results.

Add a result method to BazzReachController.



```
    $scores = $keyPhrase['keyPhrases'];  
  
    foreach ($scores as $score) {  
  
        $keyPhrase = new KeyPhrase();  
        $keyPhrase->bazz_reach_id = $bazzreach->id;  
        $keyPhrase->key_phrase = $score['text'];  
        $keyPhrase->score = round($score['score'] * 100, 1);  
        $keyPhrase->save();  
    }  
  
    return redirect()->route('bazzreach.index');  
}  
  
public function result(BazzReach $bazzreach)  
{  
  
    $sentiments = $bazzreach->sentiment()->orderBy('score', 'desc')->get()->toArray();  
    $keyPhrases = $bazzreach->keyphrase()->orderBy('score', 'desc')->get()->toArray();  
    $dataArray = [$sentiments, $keyPhrases];  
    $data = response()->json($dataArray);  
    return $data;  
}
```

First, a method is used to retrieve the sentiment and keyPhrase in BazzReach. This is because the parent-child relationship between BazzReach, sentiment, and KeyPhrase were set up in the previous section. Finally, the data is returned in JSON format. The variable name of the result method argument must match the name of the variable in the URI and the name of the method argument. After this has been done, the next step is to modify the screen.

Now we will modify the screen side. First, we will add processing to the getBazzAnalysisData function in the getBazzReachApi Web module.



To explain this function, it accesses the result method created in Laravel and retrieves the detailed analysis results. id is added to the URL because Laravel's routing configuration is set to pass the table's primary key in the URL.

In option, the cache is configured and the request cache is set to the default setting in the cache option settings. Finally, getJSON is used to retrieve the JSON data and return it to the caller.

Next, on the Home side, modify the onClick event function of the Execute Detailed Analysis button.

The screenshot shows a web browser window with two tables of analysis results. The browser's address bar shows a URL with a search term in a red box. The first table has a green header with 'ID' and 'TEXT' and contains 4 rows of data. The second table also has a green header with 'ID' and 'TEXT' and contains 10 rows of data. The text in the tables is mostly illegible due to low resolution.

ID	TEXT
1111111	
22222	
33333	
44444	

ID	TEXT
1111111	
22222	
33333	
44444	
55555	
66666	
77777	
88888	
99999	
10000	

The results of the detailed analysis are then displayed on the screen. This is the end of the display of the analysis results of the registered tweets.

BazzReachController.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Models\BazzReach;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Api\TwitterApi;
```

```
use App\Models\KeyPhrase;
```

```
use App\Models\Sentiment;
```

```
use App\Http\Api\ComprehendApi;
```

```
class BazzReachController extends Controller
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function index()
```

```
    {
```

```
        //
```

```
    }
```

```
    /**
```

```
     * Show the form for creating a new resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```

public function create()
{
    //
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $ bazzreach = new BazzReach ();
    $ bazzreach->search_word = $request->input('search_word');
    $ bazzreach->total_tweets = $request->input('total_tweets');
    $ bazzreach->save();

    return http_response_code();
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function show( BazzReach $ bazzReach )
{
    //
}

/**

```

```

* Show the form for editing the specified resource.
*
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function edit( BazzReach $ bazzReach )
{
    //
}

/**
* Update the specified resource in storage.
*
* @param \Illuminate\Http\Request $request
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function update(Request $request, BazzReach $ bazzReach )
{
    $ bazzreach->comment = $request->input('comment');
    $ bazzreach->update();
    return http_response_code();
}

/**
* Remove the specified resource from storage.
*
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function destroy( BazzReach $ bazzReach )
{

```

```

    //
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function search(Request $request)
{
    $searchWord = $request->input('search_word');
    $twitterApi = new TwitterApi();
    $totalTweets = $twitterApi->serachTweets($searchWord);
    $dataArray = [$totalTweets];
    $data = response()->json($dataArray);
    return $data;
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function bazzsearch (Request $request)
{
    $keyword = $request->input('keyword');
    $bazzReachs;

    if(empty($keyword)){
        $ bazzReachs = BazzReach ::orderBy('created_at', 'desc')->get();
    }else{
        $query = BazzReach::query();
        $keywords = explode(" ", $keyword);

```

```

        foreach ($keywords as $Key => $Value) {
            $query->orWhere('search_word', 'ilike', '%'.$Value.'%');
        }
        $bazzReachs = $query->orderBy('created_at', 'desc')->get();
    }
    $dataArray = [$bazzReachs];
    $data = response()->json($dataArray);
    return $data;
}

/**
 * Analysis the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function analysis( BazzReach $ bazzreach )
{
    $comprehendApi = new ComprehendApi();
    $sentiment = $comprehendApi->tweetSentiment($ bazzreach->total_tweets);
    $keyPhrase = $comprehendApi->tweetKeyPhrases($ bazzreach->total_tweets);

    if($sentiment['Sentiment'] == "MIXED"){
        $bazzreach->sentiment_analysis = "mixing";
    }elseif($sentiment['Sentiment'] == "NEGATIVE"){
        $bazzreach->sentiment_analysis = "negative";
    }elseif($sentiment['Sentiment'] == "POSITIVE"){
        $bazzreach->sentiment_analysis = "positive";
    }elseif($sentiment['Sentiment'] == "NEUTRAL"){
        $bazzreach->sentiment_analysis = "neutrality";
    }
}

```

```

}
$bazzreach->update();

$scores = $sentiment['SentimentScore'];
foreach ($scores as $type => $score) {
    $sentiment = new Sentiment();
    $sentiment->bazz_reach_id = $bazzreach->id;
    if($type == "Mixed"){
        $sentiment->type = "mixing";
    }elseif($type == "Negative"){
        $sentiment->type = "negative";
    }elseif($type == "Positive"){
        $sentiment->type = "positive";
    }elseif($type == "Neutral"){
        $sentiment->type = "neutrality";
    }

    $sentiment->score = round($score * 100, 1);
    $sentiment->save();
}

$scores = $keyPhrase['KeyPhrases'];

foreach ($scores as $score) {

    $keyPhrase = new KeyPhrase();
    $keyPhrase->bazz_reach_id = $bazzreach->id;
    $keyPhrase->key_phrase = $score["Text"];
    $keyPhrase->score = round($score["Score"] * 100, 1);
    $keyPhrase->save();
}

return http_response_code();
}

```

```

/**
 * Result the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function result( BazzReach $ bazzreach )
{
    $sentiments = $ bazzreach->sentiment()->orderBy('score', 'desc')-
>get()->toArray();
    $keyPhrases = $ bazzreach->keyPhrase()->orderBy('score', 'desc')-
>get()->toArray();
    $dataArray = [$sentiments,$keyPhrases];
    $data = response()->json($dataArray);
    return $data;
}
}

```


getBazzReachApi.js

```
import {fetch,getJSON} from 'wix-fetch';

export async function getSearchData(searchWord) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/search?
search_word="+encodeURIComponent(searchWord);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setBazzData (search_word,total_tweets) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/";
  let option ={
    "method": "POST",
    "mode":"core",
    "headers": {
      "Content-Type": "application/x-www-form-urlencoded"
    },
    "body": "search_word="+search_word+"&total_tweets="+total_tweets
  };
  result = await fetch(url,option);
  return result;
}
```

```
export async function getBazzSearchData (keyword) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/bazzsearch?
keyword="+encodeURIComponent(keyword);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setAnalysis(id) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/"+id+"/analysis/";
  let option ={
    "method": "POST",
    "mode":"core"
  };
  result = await fetch(url,option);
  return result;
}

export async function getBazzAnalysisData(id) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/"+id+"/result/";
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setMemo(id,comment) {
```

```
let result;
let url = "https://bazzreach.tk/api/bazzreach/"+id+"/";
let option = {
  "method": "PATCH",
  "mode": "core",
  "headers": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "body": "comment="+comment
};
result = await fetch(url,option);
return result;
}

export async function delBazzData(id) {
}
```

home.js

```
import moment from 'moment/moment';
import 'moment/locale/ja';
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

moment.locale('ja');

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};

let rollOptions1 = {
```

```

    "duration": 1500,
    "delay": 0,
    "direction": "left"
};

let rollOptions2 = {
    "duration": 1000,
    "delay": 0,
    "direction": "left"
};

let fadeOptions = {
    "duration": 500,
    "delay": 0
};

$w.onReady(function () {
    getBazzSearchData ("").then( (result) => {
        init();
        initBazzSearchData(result[0]);
        $w("#bazzSearchDataCount").text = String(result[0].length);
        selectTableData = result[0];
        $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
        $w("# bazzSearchTable ").dataFetcher = getRows;
    }).catch(err => {
        viewInfo("An error occurred during data loading");
        console.log(err, "An error occurred during data loading");
    });
});

/**
 *      Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  getSearchData(search_word).then( (result) => {

    if($w("#saveTweetStrip").collapsed){
      $w("#saveTweetStrip").expand().then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }else{
      $w("#saveTweetStrip").hide("slide",slideOptions).then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }
  }).catch(err => {
    viewInfo("An error occurred when searching for tweets");
    console.log(err, "An error occurred when searching for tweets");
    $w("#search").enable();
  });
}
```

```

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveTweet_click(event) {
  $w("#saveTweet").disable();

  setBazzData($w("#viewSearchWord").text,$w("#viewSearchData").value).t
  hen( (result) => {
    bazzSearch_click(event);
    viewInfo("Saved tweet");
    $w("#saveTweetStrip").hide("slide",slideOptions).then( ( ) => {
      $w("#viewSearchWord").text = "";
      $w("#viewSearchData").value = "";
      $w("#saveTweetStrip").collapse();
      $w("#saveTweet").enable();
    } );
  }).catch(err => {
    viewInfo("An error occurred when saving the tweet");
    console.log(err, "An error occurred when saving the tweet");
    $w("#saveTweet").enable();
  });
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */

```

```

export function bazzSearch_click(event) {
  $w("# bazzSearch ").disable();
  let keyword = $w("#keyword").value;
    getBazzSearchData (keyword).then( (result) => {
  initBazzSearchData(result[0]);
  $w("#bazzSearchDataCount").text = String(result[0].length);
  selectTableData = result[0];
  $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
  $w("# bazzSearchTable ").dataFetcher = getRows;
  $w("# bazzSearch ").enable();
}).catch(err => {
  viewInfo("An error occurred during data retrieval");
  console.log(err, "An error occurred during data retrieval");
  $w("# bazzSearch ").enable();
});
}

/**
 *   Adds an event handler that runs when a table row is selected.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.Table.html#onRowSelect)
 *   @param {$w.TableRowEvent} event
 */
export function bazzSearchTable_rowSelect(event) {
  selectIndex = event.rowIndex;
  viewAnalysisFlg = true;
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  let total_tweets = selectData.total_tweets;

  if($w("#selectBazzDataStrip").collapsed){
    $w("#selectBazzDataStrip").expand().then( ( ) => {

```



```

    $w("#selectKeyword").text = selectData.search_word;
    $w("# selectBazzData ").value = total_tweets;
    $w("#memo").value = selectData.comment;
    $w("#selectBazzDataStrip").show("slide",slideOptions);
    } );
} else {
    $w("#selectBazzDataStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#selectKeyword").text = selectData.search_word;
        $w("# selectBazzData ").value = total_tweets;
        $w("#memo").value = selectData.comment;
        $w("#selectBazzDataStrip").show("slide",slideOptions);
    } );
}

if(!$w("#analysisStrip").hidden){
    $w("#analysisStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#analysisStrip").collapse();
    } );
}

if(!$w("#memoStrip").hidden){
    $w("#memoStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#memoStrip").collapse();
    } );
}
}

/**
 *   Adds an event handler that runs when the element is clicked.
    [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */

```

```

export function viewAnalysis_click(event) {
  $w("#viewAnalysis").disable();

  if($w("#analysisStrip").collapsed){
    $w("#analysisStrip").expand().then( async( ) => {
      if( viewAnalysisFlg ){
        let selectData = $w("# bazzSearchTable ").rows[selectIndex];
        if(selectData.sentiment_analysis == ""){
          await setAnalysis(selectData.id).then( (result) => {
            }).catch(err => {
              viewInfo("An error occurred while performing detailed analysis.");
              console.log(err, "An error occurred while performing detailed
analysis.");
            });

            await bazzSearch_click(event);
          }

          await getBazzAnalysisData(selectData.id).then( async(result) => {
            selectTableData = result[0];
            $w("#sentimentsTable").pagination = {"type": "pagination",
"rowsPerPage": 10};
            $w("#sentimentsTable").dataFetcher = await getRows;

            selectTableData = result[1];
            $w("#keyPhrasesTable").pagination = {"type": "pagination",
"rowsPerPage": 10};
            $w("#keyPhrasesTable").dataFetcher = await getRows;
            viewAnalysisFlg = false;
          }).catch(err => {
            viewInfo("An error occurred when obtaining detailed analysis
results.");
          });
        }
      }
    });
  }
}

```

```

        console.log(err, "An error occurred when obtaining detailed analysis
results.");
    });

}

$w("#analysisStrip").show("slide",slideOptions).then( () => {
    $w("#viewAnalysis").enable();
} );
} );
}else{
    $w("#analysisStrip").hide("slide",slideOptions).then( () => {
        $w("#analysisStrip").collapse();
        $w("#viewAnalysis").enable();
    } );
}
}

/**
 *   Adds an event handler that runs when the element is clicked.
    [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function viewMemoInput_click(event) {
    $w("#viewMemoInput").disable();
    if($w("#memoStrip").collapsed){
        $w("#memoStrip").expand().then( () => {
            $w("#memoStrip").show("slide",slideOptions).then( () => {
                $w("#viewMemoInput").enable();
            } );
        } );
    }
}else{

```

```

    $w("#memoStrip").hide("slide",slideOptions).then( ( ) => {
      $w("#memoStrip").collapse();
      $w("#viewMemoInput").enable();
    } );
  }
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveMemo_click(event) {
  $w("#saveMemo").disable();
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  setMemo(selectData.id,$w("#memo").value).then( (result) => {
    bazzSearch_click(event);
    viewInfo("Notes saved");
    $w("#memoStrip").hide("slide",slideOptions).then( ( ) => {
      $w("#memoStrip").collapse();
      $w("#saveMemo").enable();
    } );
  }).catch(err => {
    viewInfo("An error occurred when saving the note");
    console.log(err, "An error occurred when saving the note");
    $w("#saveMemo").enable();
  });
}

/**
 *   Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
*/
export function deleteConfirm_click(event) {
}
```

/**

* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
*/
export function cancel_click(event) {
}
```

/**

* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
*/
export function delete_click(event) {
}
```

```
function viewInfo(message){
```

```
  let puffInfoOptions = {
    "duration": 500,
    "delay": 2000
  };
```

```
$w("#confirmText").text = message;
#w("#confirmBtn").hide();
#w("#confirm").show("puff",puffOptions).then( () => {
  $w("#confirm").hide("puff",puffInfoOptions)
} );
}
```

```
function init(){
  $w("#preLoader").hide("fade",fadeOptions);
  $w("#searchWordStrip").show("fade",fadeOptions);
  $w("#bazzSearchStrip").show("fade",fadeOptions);
  $w("#bazzSearchTableStrip").show("fade",fadeOptions);
  $w("#copy").show("roll",rollOptions).then( () => {
    $w("#title").show("roll",rollOptions1).then( () => {
      $w("#subCopy").show("roll",rollOptions2);
    } );
  } );
}
```

```
function initBazzSearchData (searchData) {
  searchData.forEach(function(item) {
    let date = new Date(item.created_at);

    if(item.total_tweets.length > 50){
      item.tweets_short = item.total_tweets.substr(0, 50);
      item.tweets_short += "...";
    }else{
      item.tweets_short = item.total_tweets;
    }

    if(item.sentiment_analysis == null){
      item.sentiment_analysis = "";
    }
  }
}
```

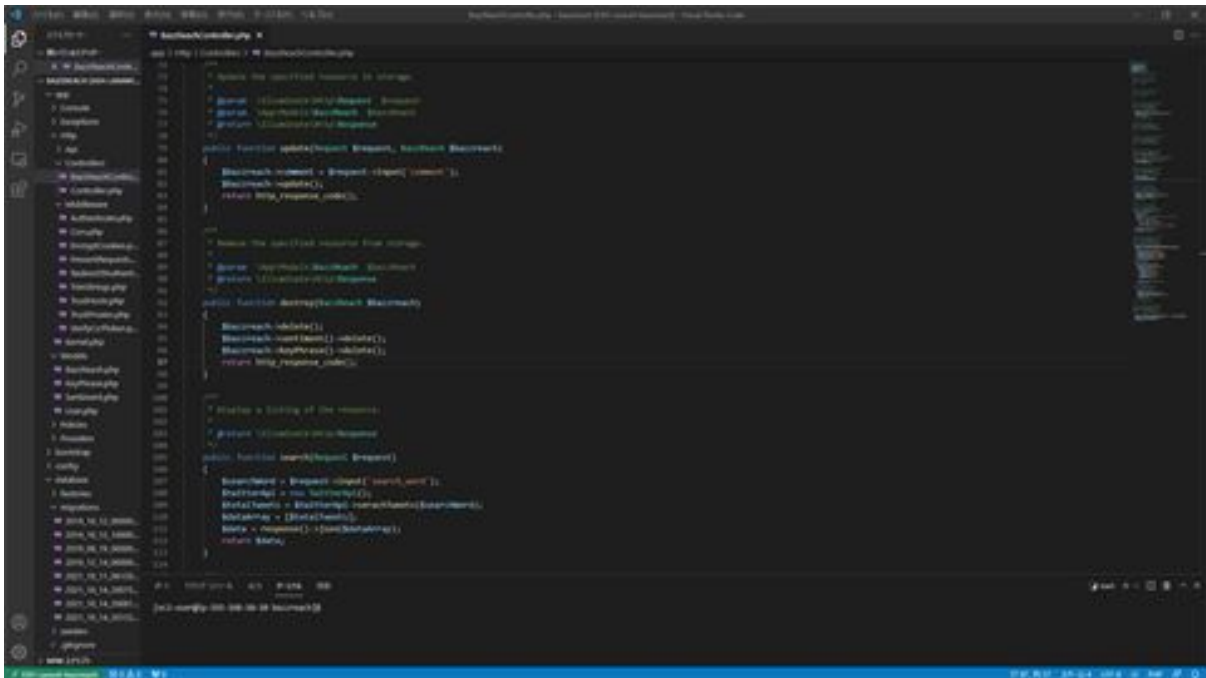
```
    item.create_passed = moment(date, 'YYYY/MM/DD  
HH:mm:S').fromNow();
```

```
  });  
}
```

```
function getRows(startRow, endRow) {  
  return new Promise( (resolve, reject) => {  
    const fetchedDataRows = selectTableData.slice(startRow, endRow);  
    resolve( {  
      pageRows: fetchedDataRows,  
      totalRowCount: selectTableData.length  
    } );  
  } );  
}
```

Let's create a function to delete registered tweets

Next, we will create a function to delete tweets that have been registered. First, open BazzReachController and modify the destroy method.



```

    /**
     * Remove the specified resource from storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Illuminate\Http\Response $response
     *
     * @return \Illuminate\Http\Response
     */
    public function destroy($request, $response, $bazzReach)
    {
        $bazzReach = $request->input('id');
        $bazzReach->delete();
        return $response->json();
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Illuminate\Http\Response $response
     *
     * @return \Illuminate\Http\Response
     */
    public function destroyBazzReach($bazzReach)
    {
        $bazzReach->delete();
        $bazzReach->sentiment->delete();
        $bazzReach->keyPhrase->delete();
        return $response->json();
    }

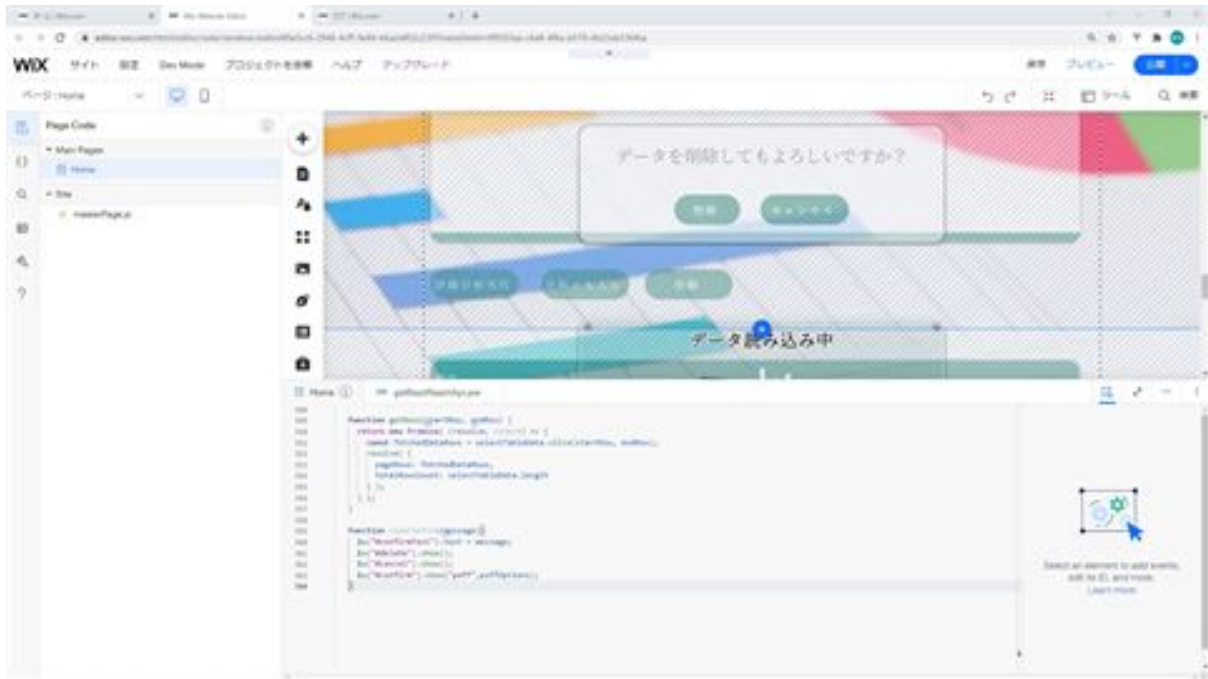
    /**
     * Display a listing of the resource.
     *
     * @param \Illuminate\Http\Request $request
     *
     * @return \Illuminate\Http\Response
     */
    public function search($request)
    {
        $searchWord = $request->input('search_word');
        $bazzReaches = $this->getBazzReach();
        $bazzReaches = $bazzReaches->search($searchWord);
        $bazzReaches = $bazzReaches->paginate(10);
        return $response->json($bazzReaches);
    }

```

This method explains that it deletes data from the bazz_reaches table and also deletes the sentiment and keyPhrase associated with the BazzReach record.

The variable name of the argument of the destroy method must match the name of the variable in the URI and the name of the argument of the method. Now that this has been done, the next step is to modify it so that it can be displayed on the screen.

Now we will modify the screen side. First, add processing to the delBazzData function in the getBazzReachApi Web module.



This process explains that when the Delete button on the screen is clicked, a confirmation dialog box with a Delete button and a Cancel button is displayed. The dialog is animated when it is displayed. The text content of the dialog displays the value passed as an argument. Next, on the Home side, add processing to the onClick event function for the delete button on the screen.



To explain the processing of this function, it calls a function that deactivates the delete button on the screen and displays a confirmation dialog. Next, on the Home side, add processing to the onClick event function for the cancel button on the confirmation screen.



strip for memo entry.



If the deletion process results in an error, an error dialog is displayed and a log of the error is shown in the console log. Once the error log is displayed, activate the Delete button on the screen and the Delete button on the confirmation screen.

Once you have made this modification, check the screen side to see if you can delete the file. screen and click the Delete button.

We have confirmed that the registered tweets are deleted when the Delete button is clicked. This is all about creating a function to delete registered tweets.

BazzReachController.php

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use App\Models\BazzReach;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Api\TwitterApi;
```

```
use App\Models\KeyPhrase;
```

```
use App\Models\Sentiment;
```

```
use App\Http\Api\ComprehendApi;
```

```
class BazzReachController extends Controller
```

```
{
```

```
    /**
```

```
     * Display a listing of the resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```

```
    public function index()
```

```
    {
```

```
        //
```

```
    }
```

```
    /**
```

```
     * Show the form for creating a new resource.
```

```
     *
```

```
     * @return \Illuminate\Http\Response
```

```
     */
```



```

public function create()
{
    //
}

/**
 * Store a newly created resource in storage.
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    $ bazzreach = new BazzReach ();
    $ bazzreach->search_word = $request->input('search_word');
    $ bazzreach->total_tweets = $request->input('total_tweets');
    $ bazzreach->save();

    return http_response_code();
}

/**
 * Display the specified resource.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function show( BazzReach $ bazzReach )
{
    //
}

/**

```

```

* Show the form for editing the specified resource.
*
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function edit( BazzReach $ bazzReach )
{
    //
}

/**
* Update the specified resource in storage.
*
* @param \Illuminate\Http\Request $request
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function update(Request $request, BazzReach $ bazzReach )
{
    $ bazzreach->comment = $request->input('comment');
    $ bazzreach->update();
    return http_response_code();
}

/**
* Remove the specified resource from storage.
*
* @param \App\Models\BazzReach $ bazzReach
* @return \Illuminate\Http\Response
*/
public function destroy( BazzReach $ bazzReach )
{

```

```

    $ bazzreach->delete();
    $ bazzreach->sentiment()->delete();
    $bazzreach->keyPhrase()->delete();
    return http_response_code();
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function search(Request $request)
{
    $searchWord = $request->input('search_word');
    $twitterApi = new TwitterApi();
    $totalTweets = $twitterApi->serachTweets($searchWord);
    $dataArray = [$totalTweets];
    $data = response()->json($dataArray);
    return $data;
}

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function bazzsearch (Request $request)
{
    $keyword = $request->input('keyword');
    $bazzReachs;

    if(empty($keyword)){
        $ bazzReachs = BazzReach ::orderBy('created_at', 'desc')->get();
    }
}

```

```

}else{
    $query = BazzReach::query();
    $keywords = explode(" ", $keyword);

    foreach ($keywords as $Key => $Value) {
        $query->orWhere('search_word', 'ilike', '%'.$Value.'%');
    }
    $bazzReachs = $query->orderBy('created_at', 'desc')->get();
}
$dataArray = [$bazzReachs];
$data = response()->json($dataArray);
return $data;
}

/**
 * Analysis the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function analysis( BazzReach $ bazzreach )
{
    $comprehendApi = new ComprehendApi();
    $sentiment = $comprehendApi->tweetSentiment($ bazzreach->total_tweets);
    $keyPhrase = $comprehendApi->tweetKeyPhrases($ bazzreach->total_tweets);

    if($sentiment['Sentiment'] == "MIXED"){
        $bazzreach->sentiment_analysis = "mixing";
    }elseif($sentiment['Sentiment'] == "NEGATIVE"){
        $bazzreach->sentiment_analysis = "negative";
    }elseif($sentiment['Sentiment'] == "POSITIVE"){

```

```

    $bazzreach->sentiment_analysis = "positive";
} elseif($sentiment['Sentiment'] == "NEUTRAL"){
    $bazzreach->sentiment_analysis = "neutrality";
}
$bazzreach->update();

$scores = $sentiment['SentimentScore'];
foreach ($scores as $type => $score) {
    $sentiment = new Sentiment();
    $sentiment->bazz_reach_id = $bazzreach->id;
    if($type == "Mixed"){
        $sentiment->type = "mixing";
    } elseif($type == "Negative"){
        $sentiment->type = "negative";
    } elseif($type == "Positive"){
        $sentiment->type = "positive";
    } elseif($type == "Neutral"){
        $sentiment->type = "neutrality";
    }

    $sentiment->score = round($score * 100, 1);
    $sentiment->save();
}

$scores = $keyPhrase['KeyPhrases'];

foreach ($scores as $score) {

    $keyPhrase = new KeyPhrase();
    $keyPhrase->bazz_reach_id = $bazzreach->id;
    $keyPhrase->key_phrase = $score["Text"];
    $keyPhrase->score = round($score["Score"] * 100, 1);
    $keyPhrase->save();
}

```

```

    return http_response_code();
}

/**
 * Result the specified resource from storage.
 *
 * @param \App\Models\BazzReach $ bazzReach
 * @return \Illuminate\Http\Response
 */
public function result( BazzReach $ bazzreach )
{
    $sentiments = $ bazzreach->sentiment()->orderBy('score', 'desc')->get()->toArray();
    $keyPhrases = $ bazzreach->keyPhrase()->orderBy('score', 'desc')->get()->toArray();
    $dataArray = [$sentiments,$keyPhrases];
    $data = response()->json($dataArray);
    return $data;
}
}

```

getBazzReachApi.js

```
import { fetch,getJSON} from 'wix-fetch';

export async function getSearchData(searchWord) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/search?
search_word="+encodeURIComponent(searchWord);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setBazzData (search_word,total_tweets) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/";
  let option ={
    "method": "POST",
    "mode":"core",
    "headers": {
      "Content-Type": "application/x-www-form-urlencoded"
    },
    "body": "search_word="+search_word+"&total_tweets="+total_tweets
  };
  result = await fetch(url,option);
  return result;
}
```

```
export async function getBazzSearchData (keyword) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/bazzsearch?
keyword="+encodeURIComponent(keyword);
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setAnalysis(id) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/"+id+"/analysis/";
  let option ={
    "method": "POST",
    "mode":"core"
  };
  result = await fetch(url,option);
  return result;
}

export async function getBazzAnalysisData(id) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/"+id+"/result/";
  let option ={
    "cache": "default"
  };
  result = await getJSON(url,option);
  return result;
}

export async function setMemo(id,comment) {
```



```
let result;
let url = "https://bazzreach.tk/api/bazzreach/"+id+"/";
let option = {
  "method": "PATCH",
  "mode": "core",
  "headers": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "body": "comment="+comment
};
result = await fetch(url,option);
return result;
}
```

```
export async function delBazzData(id) {
  let result;
  let url = "https://bazzreach.tk/api/bazzreach/"+id+"/";
  let option = {
    "method": "DELETE",
    "mode": "core"
  };
  result = await fetch(url,option);
  return result;
}
```

home.js

```
import moment from 'moment/moment';
import 'moment/locale/ja';
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

moment.locale('ja');

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
```

```

};

let rollOptions1 = {
  "duration": 1500,
  "delay": 0,
  "direction": "left"
};

let rollOptions2 = {
  "duration": 1000,
  "delay": 0,
  "direction": "left"
};

let fadeOptions = {
  "duration": 500,
  "delay": 0
};

$w.onReady(function () {
  getBazzSearchData ("").then( (result) => {
    init();
    initBazzSearchData(result[0]);
    $w("#bazzSearchDataCount").text = String(result[0].length);
    selectTableData = result[0];
    $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
    $w("# bazzSearchTable ").dataFetcher = getRows;
  }).catch(err => {
    viewInfo("An error occurred during data loading");
    console.log(err, "An error occurred during data loading");
  });
});

```

```

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  getSearchData(search_word).then( (result) => {

    if($w("#saveTweetStrip").collapsed){
      $w("#saveTweetStrip").expand().then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }else{
      $w("#saveTweetStrip").hide("slide",slideOptions).then( () => {
        $w("#viewSearchWord").text = search_word;
        $w("#viewSearchData").value = result[0];
        $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
          $w("#search").enable();
        } );
      } );
    }
  }
}).catch(err => {
  viewInfo("An error occurred when searching for tweets");
  console.log(err, "An error occurred when searching for tweets");
  $w("#search").enable();
}

```

```

    });
  }

  /**
   * Adds an event handler that runs when the element is clicked.
   [Read more]
   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
   * @param {$w.MouseEvent} event
   */
  export function saveTweet_click(event) {
    $w("#saveTweet").disable();

    setBazzData($w("#viewSearchWord").text,$w("#viewSearchData").value).t
    hen( (result) => {
      bazzSearch_click(event);
      viewInfo("Saved tweet");
      $w("#saveTweetStrip").hide("slide",slideOptions).then( ( ) => {
        $w("#viewSearchWord").text = "";
        $w("#viewSearchData").value = "";
        $w("#saveTweetStrip").collapse();
        $w("#saveTweet").enable();
      } );
    }).catch(err => {
      viewInfo("An error occurred when saving the tweet");
      console.log(err, "An error occurred when saving the tweet");
      $w("#saveTweet").enable();
    });
  }

  /**
   * Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function bazzSearch_click(event) {
  $w("# bazzSearch ").disable();
  let keyword = $w("#keyword").value;
  getBazzSearchData (keyword).then( (result) => {
    initBazzSearchData(result[0]);
    $w("#bazzSearchDataCount").text = String(result[0].length);
    selectTableData = result[0];
    $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
    $w("# bazzSearchTable ").dataFetcher = getRows;
    $w("# bazzSearch ").enable();
  }).catch(err => {
    viewInfo("An error occurred during data retrieval");
    console.log(err, "An error occurred during data retrieval");
    $w("# bazzSearch ").enable();
  });
}
```

```
/**
```

```
*      Adds an event handler that runs when a table row is selected.
```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.Table.html#onRowSelect](https://www.wix.com/corvid/reference/$w.Table.html#onRowSelect))

```
*      @param {$w.TableRowEvent} event
*/
export function bazzSearchTable_rowSelect(event) {
  selectIndex = event.rowIndex;
  viewAnalysisFlg = true;
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
```

```

let total_tweets = selectData.total_tweets;

if($w("#selectBazzDataStrip").collapsed){
  $w("#selectBazzDataStrip").expand().then( () => {
    $w("#selectKeyword").text = selectData.search_word;
    $w("# selectBazzData ").value = total_tweets;
    $w("#memo").value = selectData.comment;
    $w("#selectBazzDataStrip").show("slide",slideOptions);
  } );
}else{
  $w("#selectBazzDataStrip").hide("slide",slideOptions).then( () => {
    $w("#selectKeyword").text = selectData.search_word;
    $w("# selectBazzData ").value = total_tweets;
    $w("#memo").value = selectData.comment;
    $w("#selectBazzDataStrip").show("slide",slideOptions);
  } );
}

if(!$w("#analysisStrip").hidden){
  $w("#analysisStrip").hide("slide",slideOptions).then( () => {
    $w("#analysisStrip").collapse();
  } );
}

if(!$w("#memoStrip").hidden){
  $w("#memoStrip").hide("slide",slideOptions).then( () => {
    $w("#memoStrip").collapse();
  } );
}
}

/**
 * Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function viewAnalysis_click(event) {
  $w("#viewAnalysis").disable();

  if($w("#analysisStrip").collapsed){
    $w("#analysisStrip").expand().then( async( ) => {
      if( viewAnalysisFlg ){
        let selectData = $w("# bazzSearchTable ").rows[selectIndex];
        if(selectData.sentiment_analysis == ""){
          await setAnalysis(selectData.id).then( (result) => {
            }).catch(err => {
              viewInfo("An error occurred while performing detailed analysis");
              console.log(err, "An error occurred while performing detailed
analysis");
            });

            await bazzSearch_click(event);
          }

          await getBazzAnalysisData(selectData.id).then( async(result) => {
            selectTableData = result[0];
            $w("#sentimentsTable").pagination = {"type": "pagination",
"rowsPerPage": 10};
            $w("#sentimentsTable").dataFetcher = await getRows;

            selectTableData = result[1];
            $w("#keyPhrasesTable").pagination = {"type": "pagination",
"rowsPerPage": 10};
            $w("#keyPhrasesTable").dataFetcher = await getRows;
```



```

        viewAnalysisFlg = false;
    }).catch(err => {
        viewInfo("An error occurred when obtaining detailed analysis
results");
        console.log(err, "An error occurred when obtaining detailed analysis
results");
    });

}

$w("#analysisStrip").show("slide",slideOptions).then( () => {
    $w("#viewAnalysis").enable();
} );
} );
} else{
    $w("#analysisStrip").hide("slide",slideOptions).then( () => {
        $w("#analysisStrip").collapse();
        $w("#viewAnalysis").enable();
    } );
}
}

/**
 * Adds an event handler that runs when the element is clicked.
 [Read more]
 (https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function viewMemoInput_click(event) {
    $w("#viewMemoInput").disable();
    if($w("#memoStrip").collapsed){
        $w("#memoStrip").expand().then( () => {
            $w("#memoStrip").show("slide",slideOptions).then( () => {

```

```

    $w("#viewMemoInput").enable();
  });
});
}else{
  $w("#memoStrip").hide("slide",slideOptions).then( () => {
    $w("#memoStrip").collapse();
    $w("#viewMemoInput").enable();
  });
}
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveMemo_click(event) {
  $w("#saveMemo").disable();
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  setMemo(selectData.id,$w("#memo").value).then( (result) => {
    bazzSearch_click(event);
    viewInfo("Notes saved");
    $w("#memoStrip").hide("slide",slideOptions).then( () => {
      $w("#memoStrip").collapse();
      $w("#saveMemo").enable();
    });
  });
}).catch(err => {
  viewInfo("An error occurred when saving the note");
  console.log(err, "An error occurred when saving the note");
  $w("#saveMemo").enable();
});
}

```

```

}

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function deleteConfirm_click(event) {
  $w("#deleteConfirm").disable();
  viewConfirm("Are you sure you want to delete the data?");
}

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function cancel_click(event) {
  $w("#cancel").disable();
  $w("#confirm").hide("puff", puffOptions).then( ( ) => {
    $w("#cancel").enable();
    $w("#deleteConfirm").enable();
  } );
}

/**
 * Adds an event handler that runs when the element is clicked.
 * [Read more]
 * (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */

```

```

export function delete_click(event) {
  $w("#delete").disable();
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  delBazzData (selectData.id).then( (result) => {
    bazzSearch_click(event);
    $w("#confirm").hide("puff",puffOptions).then( () => {
      $w("#delete").enable();
      $w("#deleteConfirm").enable();
    } );

    $w("#selectBazzDataStrip").hide("slide",slideOptions).then( () => {
      $w("#selectKeyword").text = "";
      $w("# selectBazzData ").value = "";
      $w("#memo").value = "";
      $w("#selectBazzDataStrip").collapse();
    } );

    $w("#analysisStrip").hide("slide",slideOptions).then( () => {
      $w("#analysisStrip").collapse();
    } );

    $w("#memoStrip").hide("slide",slideOptions).then( () => {
      $w("#memoStrip").collapse();
    } );

  }).catch(err => {
    viewInfo("An error occurred when deleting data");
    console.log(err, "An error occurred when deleting data");
    $w("#delete").enable();
    $w("#deleteConfirm").enable();
  });
}

function viewInfo(message){

```

```

let puffInfoOptions = {
  "duration": 500,
  "delay": 2000
};

$("#confirmText").text = message;
$("#confirmBtn").hide();
$("#confirm").show("puff",puffOptions).then( () => {
  $("#confirm").hide("puff",puffInfoOptions)
});
}

```

```

function init(){
  $("#preLoader").hide("fade",fadeOptions);
  $("#searchWordStrip").show("fade",fadeOptions);
  $("#bazzSearchStrip").show("fade",fadeOptions);
  $("#bazzSearchTableStrip").show("fade",fadeOptions);
  $("#copy").show("roll",rollOptions).then( () => {
    $("#title").show("roll",rollOptions1).then( () => {
      $("#subCopy").show("roll",rollOptions2);
    });
  });
}

```

```

function initBazzSearchData (searchData) {
  searchData.forEach(function(item) {
    let date = new Date(item.created_at);

    if(item.total_tweets.length > 50){
      item.tweets_short = item.total_tweets.substr(0, 50);
      item.tweets_short += "...";
    }else{
      item.tweets_short = item.total_tweets;
    }
  });
}

```

```

    }

    if(item.sentiment_analysis == null){
        item.sentiment_analysis = "";
    }

    item.create_passed = moment(date, 'YYYY/MM/DD
HH:mm:S').fromNow();

    });
}

function getRows(startRow, endRow) {
    return new Promise( (resolve, reject) => {
        const fetchedDataRows = selectTableData.slice(startRow, endRow);
        resolve( {
            pageRows: fetchedDataRows,
            totalRowCount: selectTableData.length
        } );
    } );
}

function viewConfirm(message){
    $w("#confirmText").text = message;
    $w("#delete").show();
    $w("#cancel").show();
    $w("#confirm").show("puff",puffOptions);
}

```

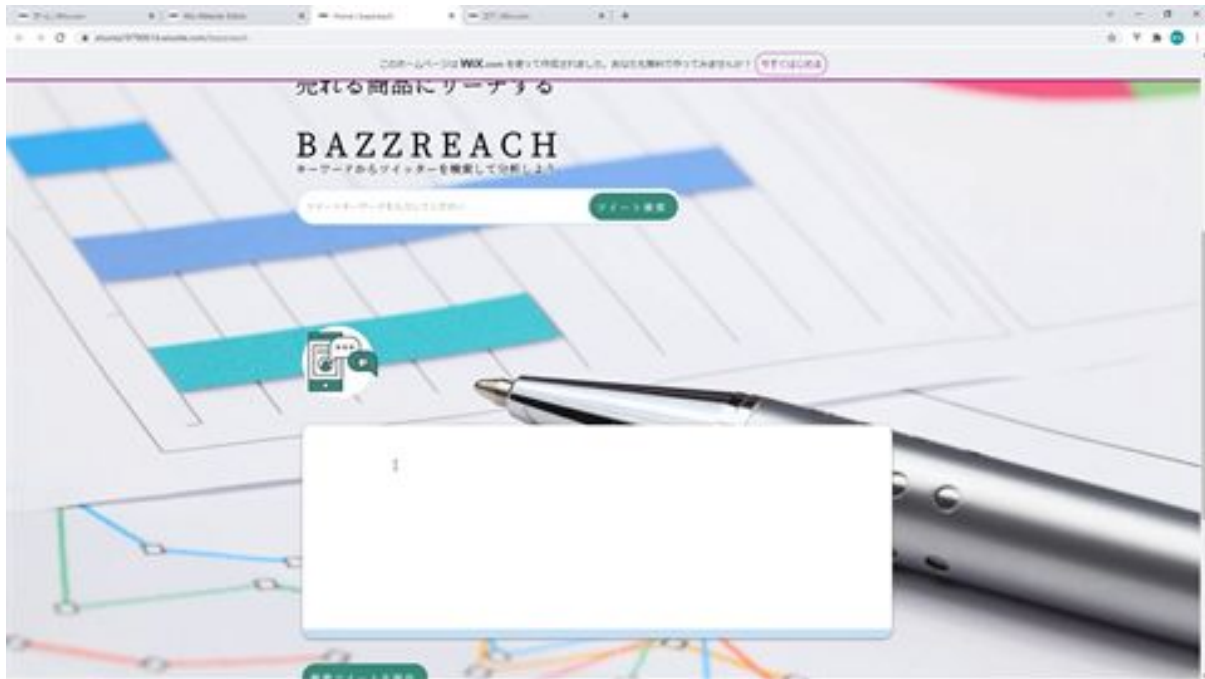
Let's check and adjust the entire application

Now let's check the whole application we have created. First of all, when the loading is finished, the copy and title parts should be displayed with animation, but they are not, so I would like to fix that.



The animation is already set up, so set the copy and title to be hidden. Now the title and copy will be displayed with animation after the loading is complete.

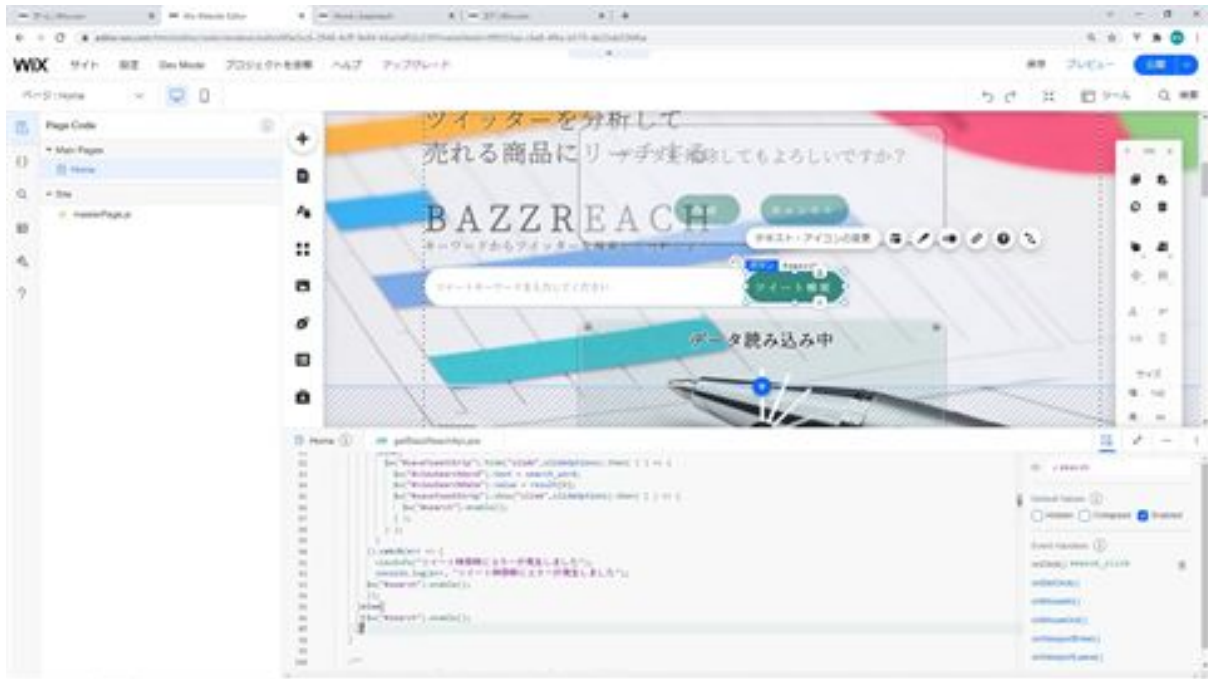
The next time you search for tweets without tweet keywords, you will get a search with no content displayed.



If you don't enter the tweet keyword, it will not be searched.



When the tweet search button is clicked, getSearchData is used to search for tweets, and an if statement is added before the process so that if no tweet keyword is entered, the tweet search is not performed.



The content of the tweet keyword is retrieved by searchWord, and if the variable search_word is non-empty, the search is performed, otherwise, the tweet search button is activated. Otherwise, the tweet search button will be activated.

Now let's check it out on the screen. Click the Tweet Search button without entering any keywords.

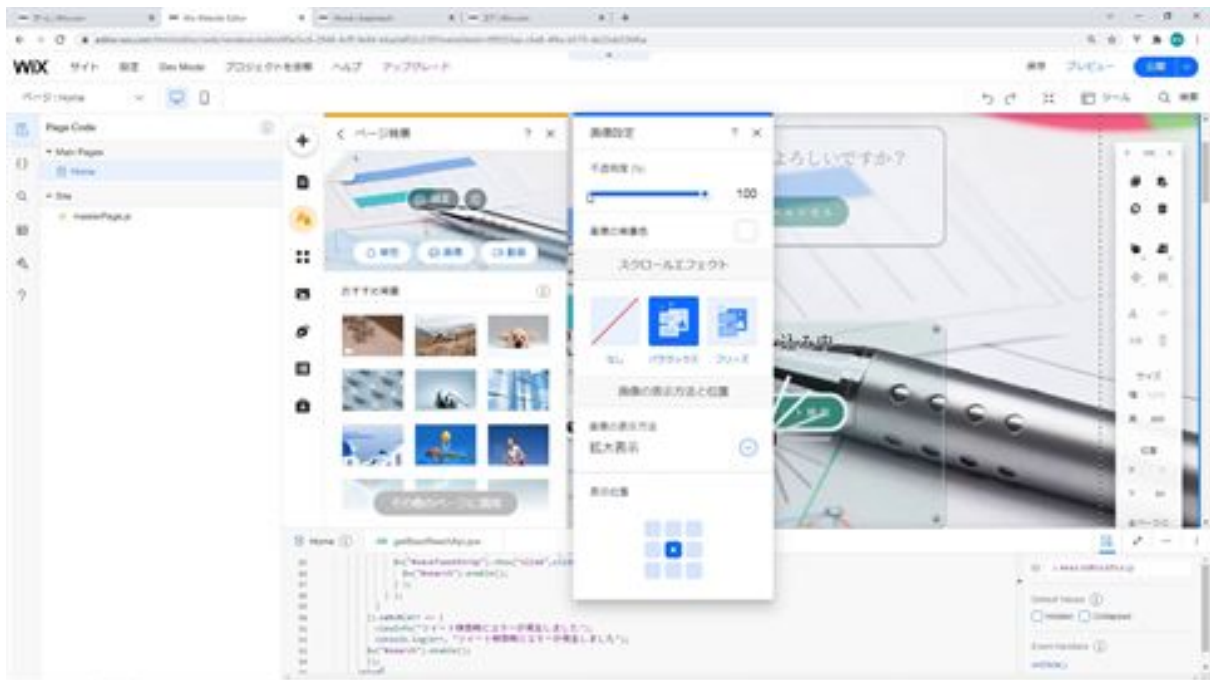


If you do so, no search will be performed and nothing will be displayed. If you click the tweet search button again after entering the tweet keywords, the search results will be displayed.

Next, the keywords, title, and keyword input are slightly off-center, so move them up a bit.



I checked the screen after moving it, and this ballpoint pen diagram and the keyword input are overlapping, so I would like to shift the background picture.



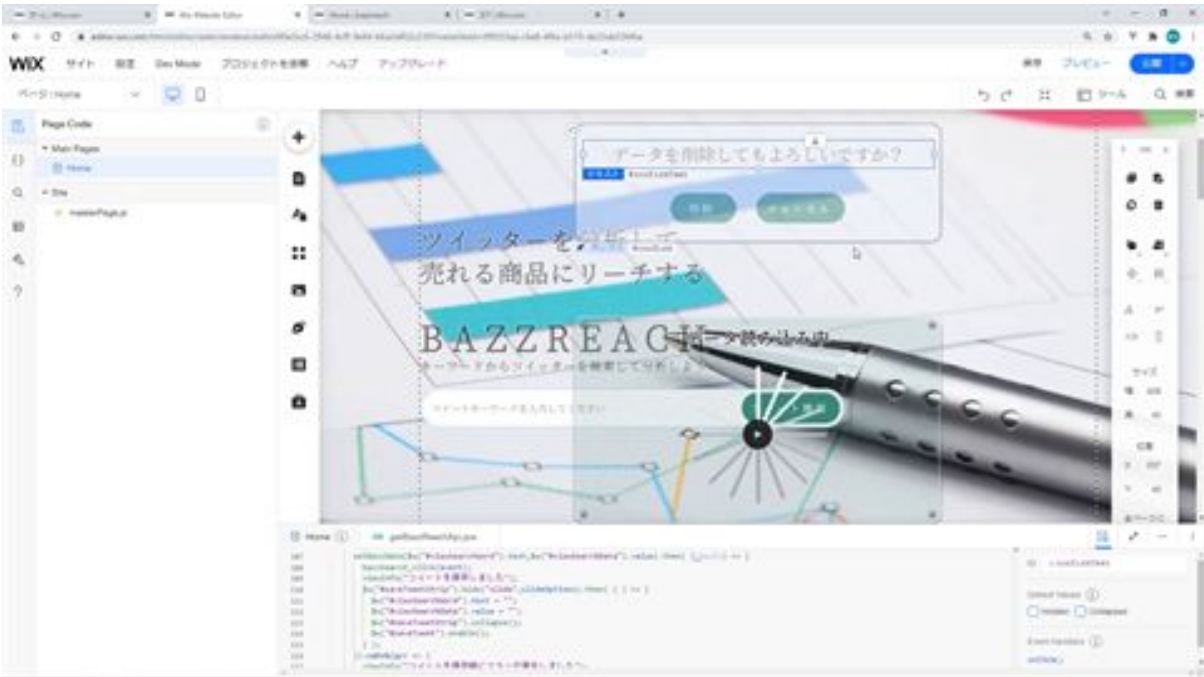
The ballpoint pen part has been fixed so that it does not overlap with the keyword input.



The next step is to fix the wrapped text in the error dialog.

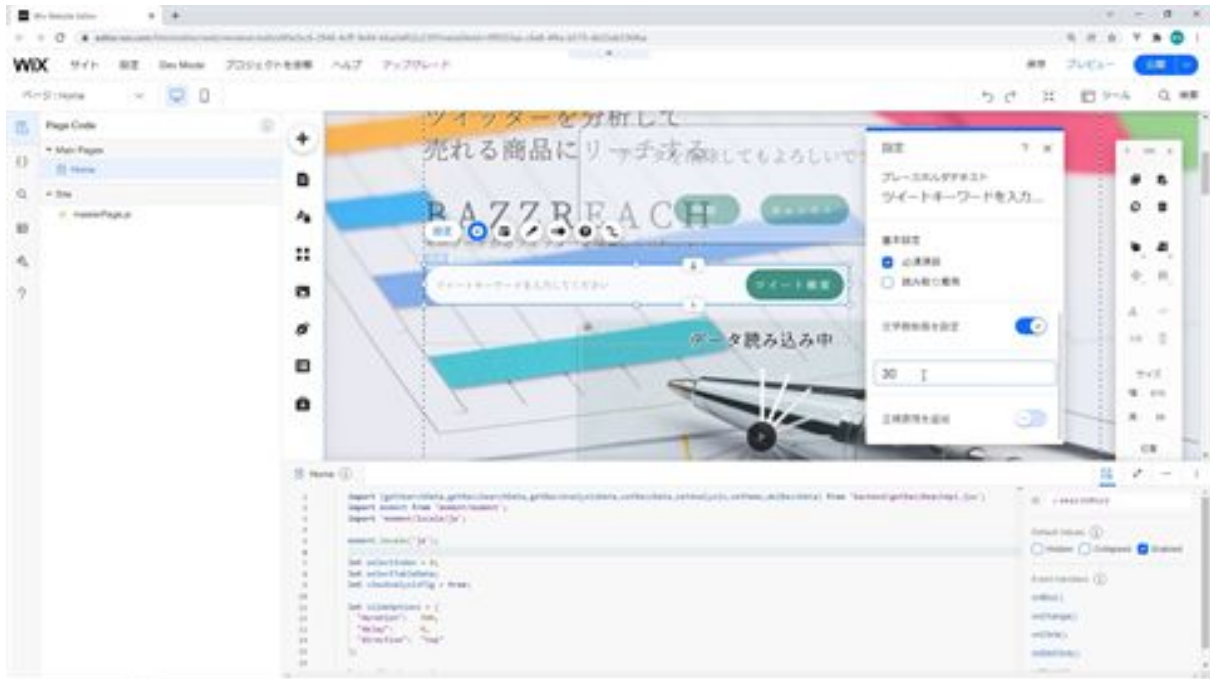


The width of this text field was a little short, so we will make it a little longer.

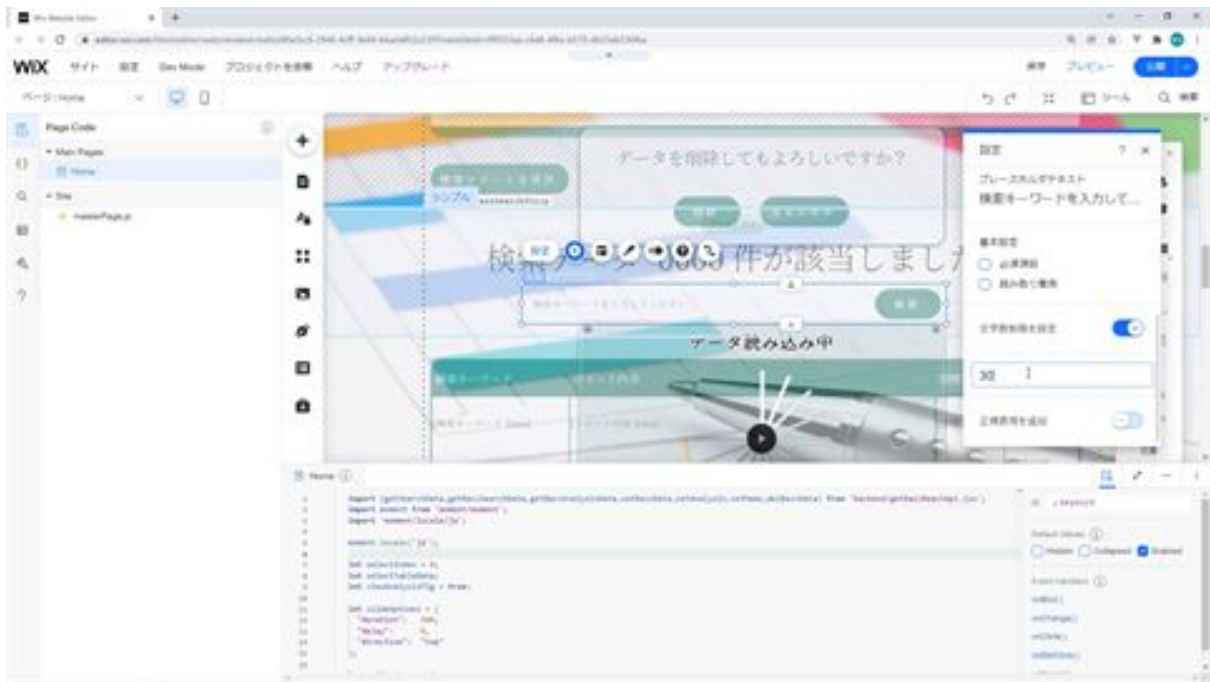


We have confirmed that it is displayed correctly without wrapping in the dialog.





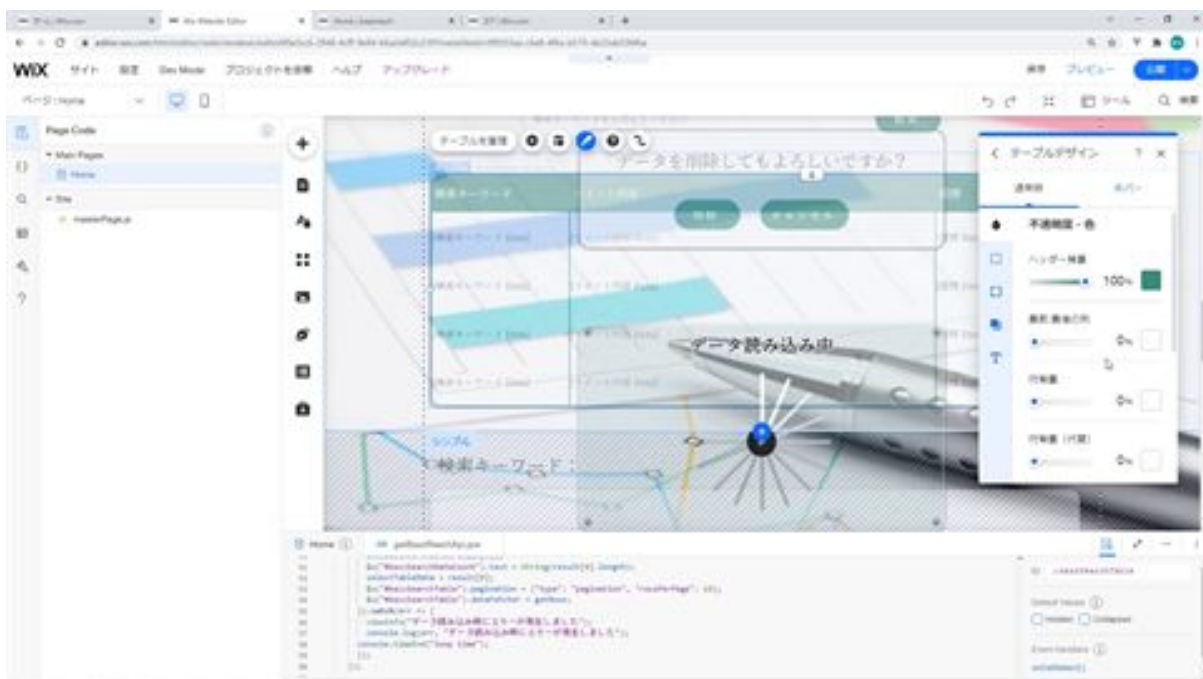
In the same way, you can enter keywords for narrowing down your search.



Next, in the table selection, only the elapsed time will not be selected, so we will fix that.



From Customize Design, select Opacity/Color, and change the opacity of the previous/last column to 0%.



After saving the file and checking it, you can see that the elapsed time is colored when hovering and when selected.



That's it for checking the entire application and making adjustments.

home.js

```
import moment from 'moment/moment';
import 'moment/locale/ja';
import
{getSearchData,setBazzData,getBazzSearchData,setAnalysis,getBazzAnaly
sisData,setMemo,delBazzData} from 'backend/getBazzReachApi.jsw';

moment.locale('ja');

let selectIndex = 0;
let selectTableData;
let viewAnalysisFlg = true;

let slideOptions = {
  "duration": 500,
  "delay": 0,
  "direction": "top"
};

let puffOptions = {
  "duration": 500,
  "delay": 0
};

let rollOptions = {
  "duration": 4000,
  "delay": 0,
  "direction": "top"
};

let rollOptions1 = {
```

```

    "duration": 1500,
    "delay": 0,
    "direction": "left"
};

let rollOptions2 = {
    "duration": 1000,
    "delay": 0,
    "direction": "left"
};

let fadeOptions = {
    "duration": 500,
    "delay": 0
};

$w.onReady(function () {
    getBazzSearchData ("").then( (result) => {
        init();
        initBazzSearchData(result[0]);
        $w("#bazzSearchDataCount").text = String(result[0].length);
        selectTableData = result[0];
        $w("# bazzSearchTable ").pagination = {"type": "pagination",
"rowsPerPage": 10};
        $w("# bazzSearchTable ").dataFetcher = getRows;
    }).catch(err => {
        viewInfo("An error occurred during data loading");
        console.log(err, "An error occurred during data loading");
    });
});

/**
 *      Adds an event handler that runs when the element is clicked.

```

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
*      @param {$w.MouseEvent} event
*/
export function search_click(event) {
  $w("#search").disable();
  let search_word = $w("#searchWord").value;
  if(search_word != ""){
    getSearchData(search_word).then( (result) => {

      if($w("#saveTweetStrip").collapsed){
        $w("#saveTweetStrip").expand().then( () => {
          $w("#viewSearchWord").text = search_word;
          $w("#viewSearchData").value = result[0];
          $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
            $w("#search").enable();
          } );
        } );
      } else{
        $w("#saveTweetStrip").hide("slide",slideOptions).then( () => {
          $w("#viewSearchWord").text = search_word;
          $w("#viewSearchData").value = result[0];
          $w("#saveTweetStrip").show("slide",slideOptions).then( () => {
            $w("#search").enable();
          } );
        } );
      }
    }
  ).catch(err => {
    viewInfo("An error occurred when searching for tweets");
    console.log(err, "An error occurred when searching for tweets");
    $w("#search").enable();
  });
}
```

```

    }else{
      $w("#search").enable();
    }
  }

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveTweet_click(event) {
  $w("#saveTweet").disable();

  setBazzData($w("#viewSearchWord").text,$w("#viewSearchData").value).t
  hen( (result) => {
    bazzSearch_click(event);
    viewInfo("Saved tweet");
    $w("#saveTweetStrip").hide("slide",slideOptions).then( ( ) => {
      $w("#viewSearchWord").text = "";
      $w("#viewSearchData").value = "";
      $w("#saveTweetStrip").collapse();
      $w("#saveTweet").enable();
    } );
  }).catch(err => {
    viewInfo("An error occurred when saving the tweet");
    console.log(err, "An error occurred when saving the tweet");
    $w("#saveTweet").enable();
  });
}

/**

```

* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function bazzSearch_click(event) {  
  $w("# bazzSearch ").disable();  
  let keyword = $w("#keyword").value;  
  getBazzSearchData (keyword).then( (result) => {  
    initBazzSearchData(result[0]);  
    $w("#bazzSearchDataCount").text = String(result[0].length);  
    selectTableData = result[0];  
    $w("# bazzSearchTable ").pagination = {"type": "pagination",  
"rowsPerPage": 10};  
    $w("# bazzSearchTable ").dataFetcher = getRows;  
    $w("# bazzSearch ").enable();  
  }).catch(err => {  
    viewInfo("An error occurred during data retrieval");  
    console.log(err, "An error occurred during data retrieval");  
    $w("# bazzSearch ").enable();  
  });  
}
```

```
/**
```

* Adds an event handler that runs when a table row is selected.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.Table.html#onRowSelect](https://www.wix.com/corvid/reference/$w.Table.html#onRowSelect))

```
* @param {$w.TableRowEvent} event
```

```
*/
```

```
export function bazzSearchTable_rowSelect(event) {  
  selectIndex = event.rowIndex;  
  viewAnalysisFlg = true;
```

```

let selectData = $w("# bazzSearchTable ").rows[selectIndex];
let total_tweets = selectData.total_tweets;

if($w("#selectBazzDataStrip").collapsed){
  $w("#selectBazzDataStrip").expand().then( ( ) => {
    $w("#selectKeyword").text = selectData.search_word;
    $w("# selectBazzData ").value = total_tweets;
    $w("#memo").value = selectData.comment;
    $w("#selectBazzDataStrip").show("slide",slideOptions);
  } );
}else{
  $w("#selectBazzDataStrip").hide("slide",slideOptions).then( ( ) => {
    $w("#selectKeyword").text = selectData.search_word;
    $w("# selectBazzData ").value = total_tweets;
    $w("#memo").value = selectData.comment;
    $w("#selectBazzDataStrip").show("slide",slideOptions);
  } );
}

if(!$w("#analysisStrip").hidden){
  $w("#analysisStrip").hide("slide",slideOptions).then( ( ) => {
    $w("#analysisStrip").collapse();
  } );
}

if(!$w("#memoStrip").hidden){
  $w("#memoStrip").hide("slide",slideOptions).then( ( ) => {
    $w("#memoStrip").collapse();
  } );
}
}

/**

```


* Adds an event handler that runs when the element is clicked.

[Read more]

([https://www.wix.com/corvid/reference/\\$w.ClickableMixin.html#onClick](https://www.wix.com/corvid/reference/$w.ClickableMixin.html#onClick))

```
* @param {$w.MouseEvent} event
```

```
*/
```

```
export function viewAnalysis_click(event) {
```

```
  $w("#viewAnalysis").disable();
```

```
  if($w("#analysisStrip").collapsed){
```

```
    $w("#analysisStrip").expand().then( async( ) => {
```

```
      if( viewAnalysisFlg ){
```

```
        let selectData = $w("# bazzSearchTable ").rows[selectIndex];
```

```
        if(selectData.sentiment_analysis == ""){
```

```
          await setAnalysis(selectData.id).then( (result) => {
```

```
            }).catch(err => {
```

```
              viewInfo("An error occurred while performing detailed analysis");
```

```
              console.log(err, "An error occurred while performing detailed
```

```
analysis");
```

```
            });
```

```
          await bazzSearch_click(event);
```

```
        }
```

```
        await getBazzAnalysisData(selectData.id).then( async(result) => {
```

```
          selectTableData = result[0];
```

```
          $w("#sentimentsTable").pagination = {"type": "pagination",
```

```
"rowsPerPage": 10};
```

```
          $w("#sentimentsTable").dataFetcher = await getRows;
```

```
          selectTableData = result[1];
```

```
          $w("#keyPhrasesTable").pagination = {"type": "pagination",
```

```
"rowsPerPage": 10};
```

```

    $w("#keyPhrasesTable").dataFetcher = await getRows;
    viewAnalysisFlg = false;
  }).catch(err => {
    viewInfo("An error occurred when obtaining detailed analysis
results");
    console.log(err, "An error occurred when obtaining detailed analysis
results");
  });
}

$w("#analysisStrip").show("slide",slideOptions).then( () => {
  $w("#viewAnalysis").enable();
} );
} );
}else{
  $w("#analysisStrip").hide("slide",slideOptions).then( () => {
    $w("#analysisStrip").collapse();
    $w("#viewAnalysis").enable();
  } );
}
}

/**
 * Adds an event handler that runs when the element is clicked.
 [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 * @param {$w.MouseEvent} event
 */
export function viewMemoInput_click(event) {
  $w("#viewMemoInput").disable();
  if($w("#memoStrip").collapsed){
    $w("#memoStrip").expand().then( () => {

```

```

    $w("#memoStrip").show("slide",slideOptions).then( () => {
      $w("#viewMemoInput").enable();
    });
  });
} else {
  $w("#memoStrip").hide("slide",slideOptions).then( () => {
    $w("#memoStrip").collapse();
    $w("#viewMemoInput").enable();
  });
}
}
}

/**
 *   Adds an event handler that runs when the element is clicked.
 *   [Read more]
 *   (https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function saveMemo_click(event) {
  $w("#saveMemo").disable();
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  setMemo(selectData.id,$w("#memo").value).then( (result) => {
    bazzSearch_click(event);
    viewInfo("Notes saved");
    $w("#memoStrip").hide("slide",slideOptions).then( () => {
      $w("#memoStrip").collapse();
      $w("#saveMemo").enable();
    });
  });
}).catch(err => {
  viewInfo("An error occurred when saving the note");
  console.log(err, "An error occurred when saving the note");
  $w("#saveMemo").enable();
});

```

```

    });
  }

/**
 *   Adds an event handler that runs when the element is clicked.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function deleteConfirm_click(event) {
  $w("#deleteConfirm").disable();
  viewConfirm("Are you sure you want to delete the data?");
}

/**
 *   Adds an event handler that runs when the element is clicked.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event
 */
export function cancel_click(event) {
  $w("#cancel").disable();
  $w("#confirm").hide("puff",puffOptions).then( ( ) => {
    $w("#cancel").enable();
    $w("#deleteConfirm").enable();
  } );
}

/**
 *   Adds an event handler that runs when the element is clicked.
   [Read more]
(https://www.wix.com/corvid/reference/\$w.ClickableMixin.html#onClick)
 *   @param {$w.MouseEvent} event

```

```

*/
export function delete_click(event) {
  $w("#delete").disable();
  let selectData = $w("# bazzSearchTable ").rows[selectIndex];
  delBazzData (selectData.id).then( (result) => {
    bazzSearch_click(event);
    $w("#confirm").hide("puff",puffOptions).then( () => {
      $w("#delete").enable();
      $w("#deleteConfirm").enable();
    } );

    $w("#selectBazzDataStrip").hide("slide",slideOptions).then( () => {
      $w("#selectKeyword").text = "";
      $w("# selectBazzData ").value = "";
      $w("#memo").value = "";
      $w("#selectBazzDataStrip").collapse();
    } );

    $w("#analysisStrip").hide("slide",slideOptions).then( () => {
      $w("#analysisStrip").collapse();
    } );

    $w("#memoStrip").hide("slide",slideOptions).then( () => {
      $w("#memoStrip").collapse();
    } );

  }).catch(err => {
    viewInfo("An error occurred when deleting data");
    console.log(err, "An error occurred when deleting data");
    $w("#delete").enable();
    $w("#deleteConfirm").enable();
  });
}

```

```

function viewInfo(message){

    let puffInfoOptions = {
        "duration": 500,
        "delay": 2000
    };

    $w("#confirmText").text = message;
    $w("#confirmBtn").hide();
    $w("#confirm").show("puff",puffOptions).then( () => {
        $w("#confirm").hide("puff",puffInfoOptions)
    } );
}

```

```

function init(){
    $w("#preLoader").hide("fade",fadeOptions);
    $w("#searchWordStrip").show("fade",fadeOptions);
    $w("#bazzSearchStrip").show("fade",fadeOptions);
    $w("#bazzSearchTableStrip").show("fade",fadeOptions);
    $w("#copy").show("roll",rollOptions).then( () => {
        $w("#title").show("roll",rollOptions1).then( () => {
            $w("#subCopy").show("roll",rollOptions2);
        } );
    } );
}

```

```

function initBazzSearchData (searchData) {
    searchData.forEach(function(item) {
        let date = new Date(item.created_at);

        if(item.total_tweets.length > 50){
            item.tweets_short = item.total_tweets.substr(0, 50);
            item.tweets_short += "...";
        }else{

```

```

    item.tweets_short = item.total_tweets;
  }

  if(item.sentiment_analysis == null){
    item.sentiment_analysis = "";
  }

  item.create_passed = moment(date, 'YYYY/MM/DD
HH:mm:S').fromNow();

  });
}

function getRows(startRow, endRow) {
  return new Promise( (resolve, reject) => {
    const fetchedDataRows = selectTableData.slice(startRow, endRow);
    resolve( {
      pageRows: fetchedDataRows,
      totalRowCount: selectTableData.length
    } );
  } );
}

function viewConfirm(message){
  $w("#confirmText").text = message;
  $w("#confirmBtn").show();
  $w("#confirm").show("puff",puffOptions);
}

```

postscript

Author Profile

Shunta Sako

He has been working as a web designer since he was a university student, and then changed to a programmer. He has been working as a web designer, programmer, and full-stack engineer for more than 20 years in the IT industry. I share my know-how of system development with my students. Also active as an instructor at a programming school.

Learn about this book in the video

https://www.udemy.com/course/wix-laravel-api/?couponCode=COUPON_20220421

