Chapter 3 - Deploying Applications to Cloud

Part 1 Concept of deployment

Deployment is the last step and also the most critical step of building software. When you have completed the development of a piece of software, you need to deploy it somewhere so people can use it. The deployment method and platform is different for different types of software.

Type of Software	Example	Deployment Method
Desktop Application	Microsoft Office	Build a installation package and upload to a website for people to download
Web App	Gmail Web Version	Run your app in servers and connect it with a domain (i.e. your website address)
Simple Web Page	Static HTML personal site	Upload to static web page hosting services
Mobile App	Instagram App	Build an app “bundle” and upload to the app store

In this chapter, we will focus on the deployment of Web App and simple web pages.

Part 2 Running an app in the server in traditional way

In the old days, deploying a web app was complicated. You need to first prepare your server. Then you will need to install tons of different software on the server, including web server software, network infrastructure, database software and more. This was just some of the software that you need to install on the server.

After successfully running your app on the server, then you will need to configure the networking, autoscaling, security and fail-safe mechanisms, not to mention the system monitoring and alerts.

For usual small and medium enterprises, this is a huge burden and cost. That's why only the large companies would own a website or app in the past

Part 3 The modern way of running a web app

You might have noticed that a lot of the small and medium enterprises are owning their website and apps now. The technology breakthrough behind this is the advancement and availability of Cloud.

To run a website or web application, you don't need to configure the server like what we mentioned in Part 2 anymore. Instead, the cloud providers would help in managing all these so you can focus on developing the app.

There are 2 major ways of deploying web software on the cloud:

1. Infrastructure as a Service (IaaS)
   It refers to renting the infrastructure from cloud providers. For example, you will be renting networking features, computers (virtual or on dedicated hardware), and data storage space. But you still need to do quite some configurations before you can get your application up and running.

   IaaS provides you with the highest level of flexibility on cloud but that also means that you will need to spend more resources in managing the cloud resources.

2. Platform as a Service (PaaS)
   The concept of PaaS is that the Cloud provider would manage the infrastructure for you. You just need to bring the code to the platform, and it will automatically help you in setting up everything from server software, to networking, monitoring and more.

   For beginners, it is the easiest way to get started on deploying web applications on AWS.

In the next part, we are going to introduce you to 4 ways of deploying web applications, they are all PaaS ways of deploying software on Cloud.

Part 4 Common ways of deploying applications on the cloud

In this section, we are going to teach you 4 ways of deploying web applications. You don't need to memorize the steps as you can always refer to these notes or the official webpage.

You don't need to read through all these parts. Instead, you only need to look at those that are relevant to you.

Type of Application	Section
React	4.1
ExpressJS	4.2
Static HTML webpage	4.3 or 4.4

4.1 Deploy your React App to AWS Amplify

According to the official documentation of AWS,

> AWS Amplify is a set of purpose-built tools and features that lets frontend web and mobile developers quickly and easily build full-stack applications on AWS, with the flexibility to leverage the breadth of AWS services as your use cases evolve.

One of the key features of AWS Amplify is the easy deployment of dynamic websites built by React, Vue and other web frameworks.

That's why we can leverage AWS Amplify to deploy our React assignments, and make our web pages available for the public.

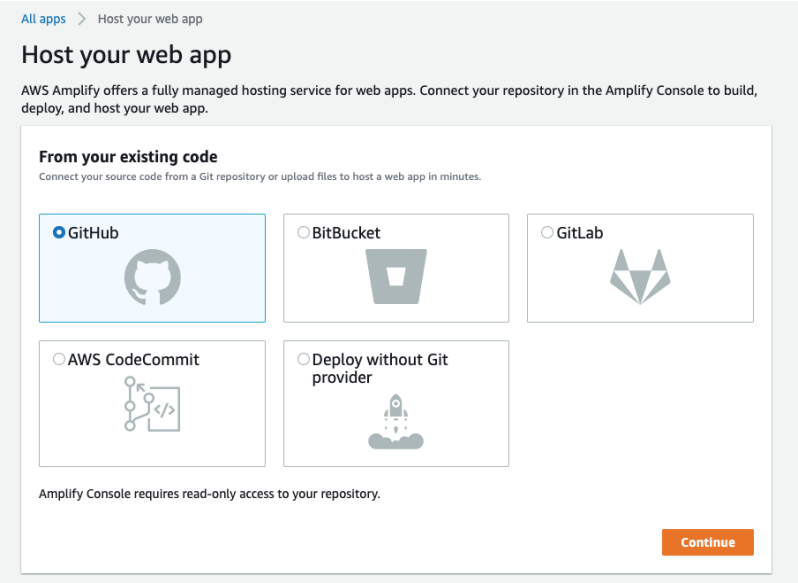
We have listed out the detailed steps below. Do note that the steps updates from time to time, so you might also want to refer to the official documentation at:
https://aws.amazon.com/getting-started/hands-on/build-react-app-amplify-graphql/module-one/

Steps:

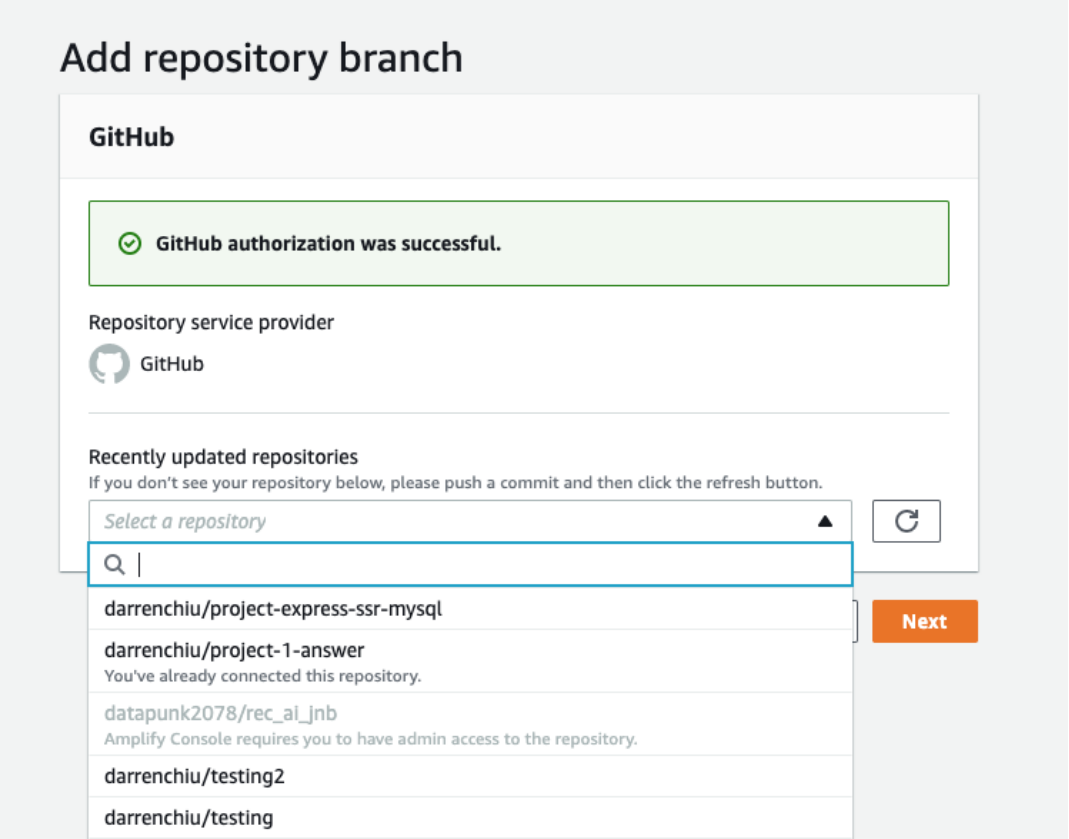
1. Login to AWS Console at https://console.aws.amazon.com, and search for "Amplify" in the search bar.

2. In the Amplify panel, click “New App” -> “Host web app”. You might see a different interface as you are new to Amplify. Look for the “New App” button.

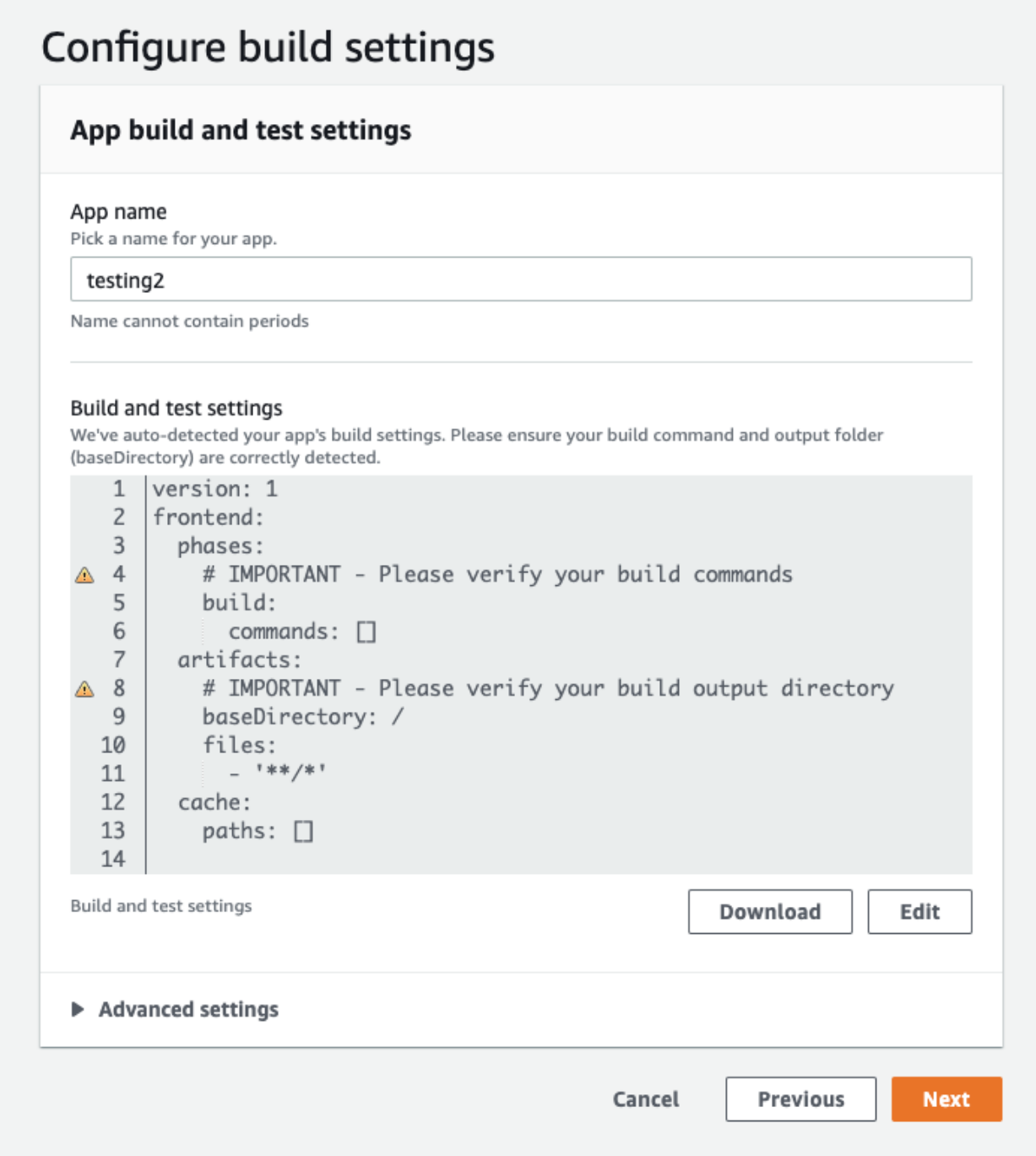
3. You should then be directed to this page. Pick GitHub and follow the steps to connect to your GitHub account.



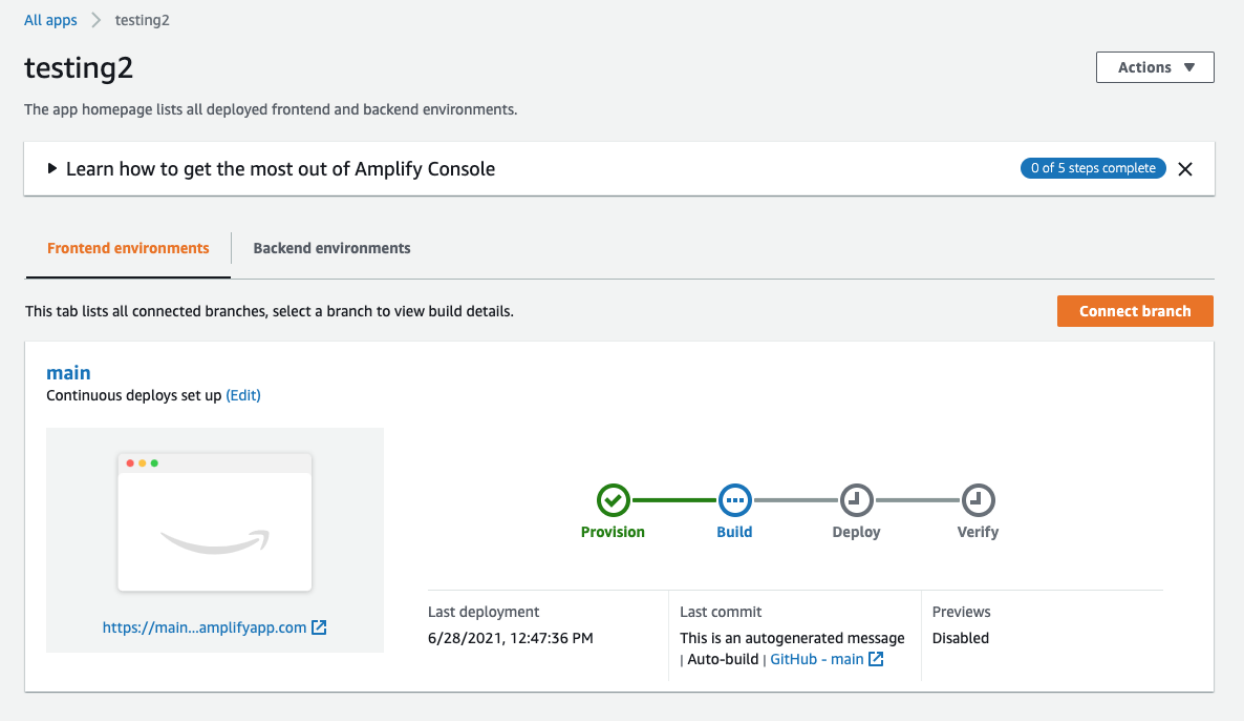
4. After linking up your GitHub account with AWS, you should be able to pick from a list of repositories like the screenshot below. Pick the React project that you want to deploy. For branches, you can pick “main” for now as you only got one branch for now.



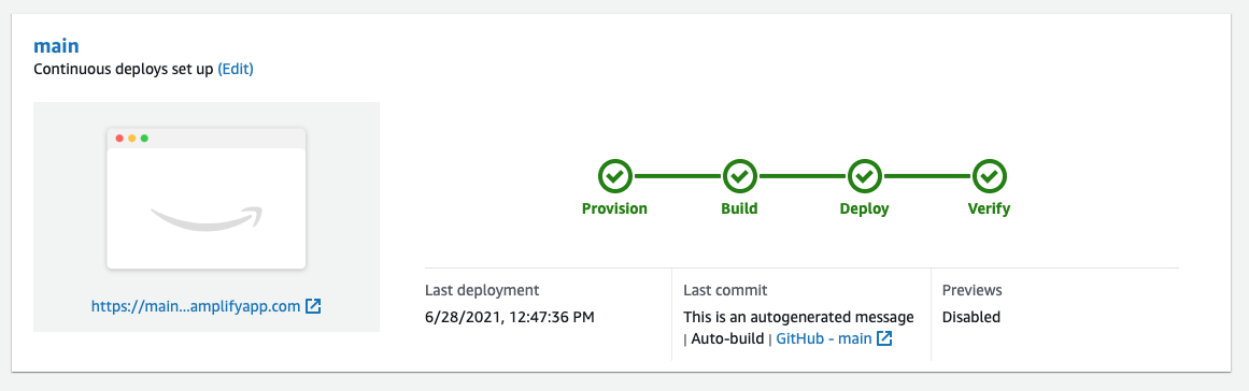
5. Then you will be directed to the “Build Settings” page. As we are deploying a standard React app, no modifications are needed. Just press “Next”. On this page, you can also customize your app name.



6. The next page is the “summary” page of all your setup. Take a look into it and just click “Save and Deploy”. Wait for a moment, Then you will then be redirected to the Amplify console like below:

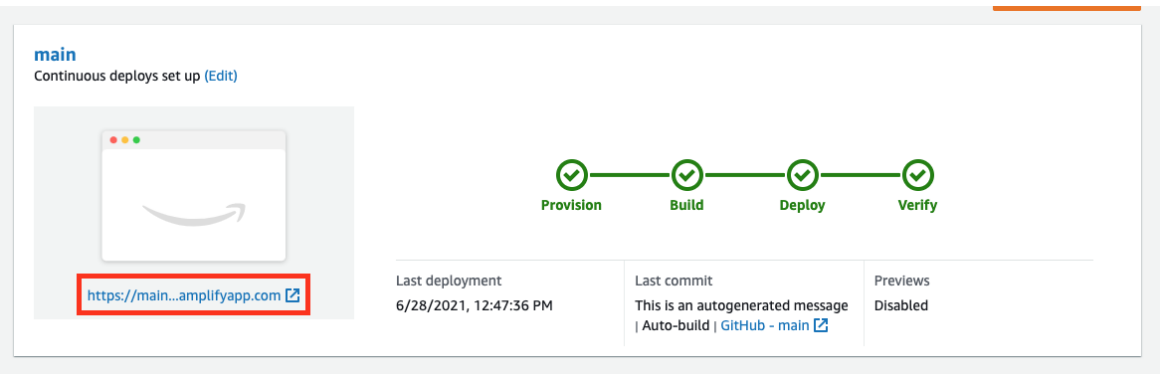


7. On this page, just wait until the deployment is done. If everything is smooth, then you should be able to see all four steps becoming a “green check mark” like below:



8. Once you have all 4 checkmarks, then you can click the link on the left to see if it is working as expected. If it's not working as expected, feel free to reach out to our mentors for help. This is a public URL of your app. You can use it to showcase your projects to others. (If you are using TalentLabs AWS account, be aware that we might remove or pause your project without further notice).

   Once everything is working, please copy the link to your app, and put it in table in Part 3 for submission.



4.2 Deploying ExpressJS Application on AWS Elastic Beanstalk

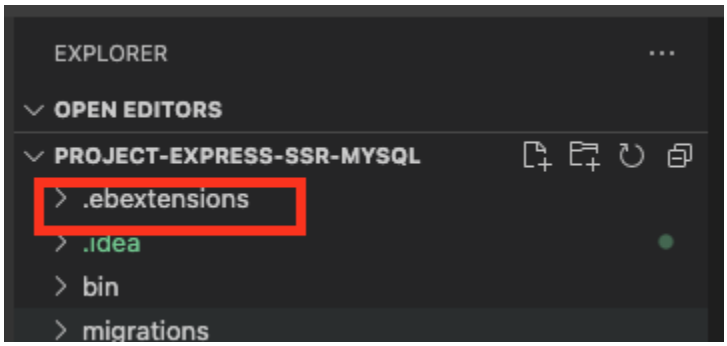
If you want to deploy an auto-scaling backend application (e.g. Express JS backend application), you might want to consider using AWS Elastic Beanstalk. Only minimal configuration is needed, and your server side program will be up and running in almost no time.

If you have built some Express JS applications before, you can consider deploying your Express JS app to AWS Elastic Beanstalk, you will need to first set up some deployment config for your app and zip your code package.

Preparing for the deployment

Steps:

1. Select one of your Express JS projects and open it up in Visual Studio Code.
2. Create a new folder in your project named ".ebextensions". Make sure you include the dot at the beginning of the folder name.



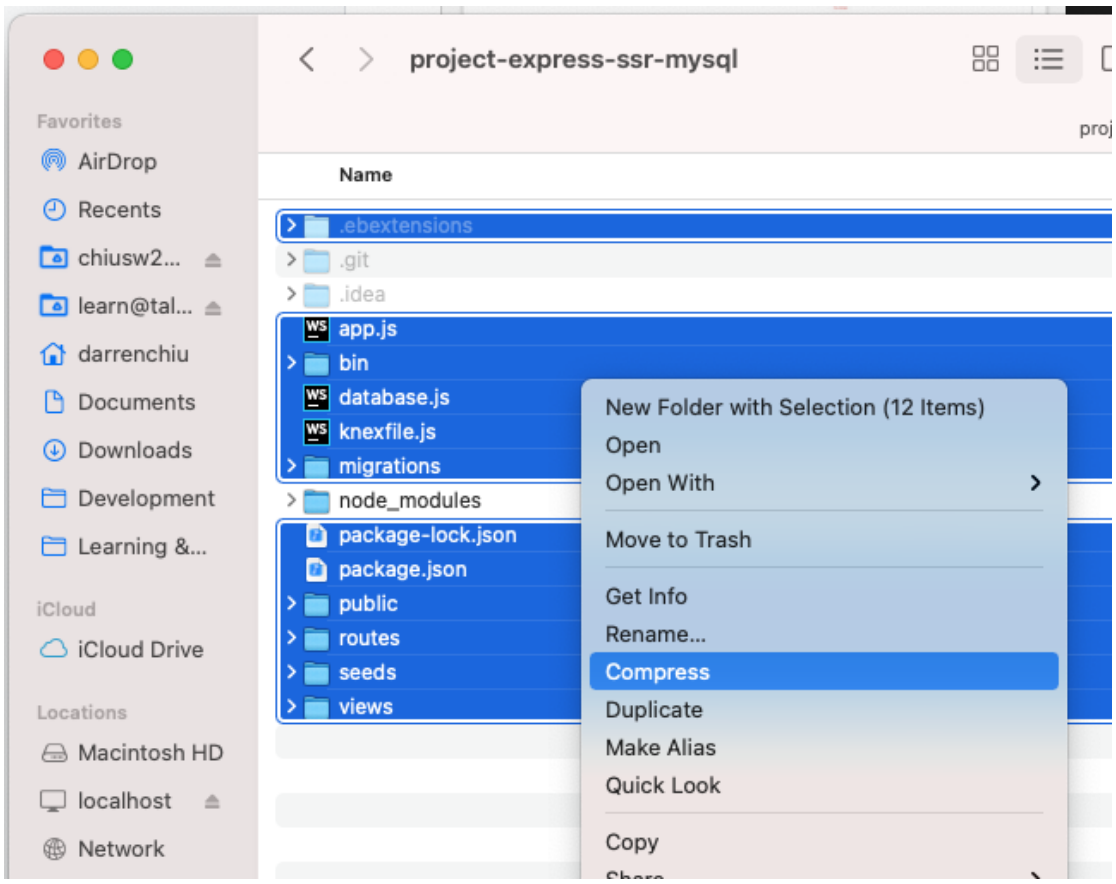
3. Inside the ".ebextensions" folder, create a new file named "staticfiles.config". Copy and paste the following code in the "staticfiles.config" file. This is to tell AWS about the location of your static files (e.g. css, images etc.). Save the file.

```
option_settings:
  aws:elasticbeanstalk:environment:proxy:staticfiles:
    /public: /public
```

4. Go to your project folder in File Explorer (Windows) or Finder (Mac). Open it up and select all the files and folders (except node_modules and .git). Then right click to zip all these files into a zip file. The file name doesn't matter.

   If you are using Windows, the zip function should be named as "Send to" -> "Compressed (zipped) folder".

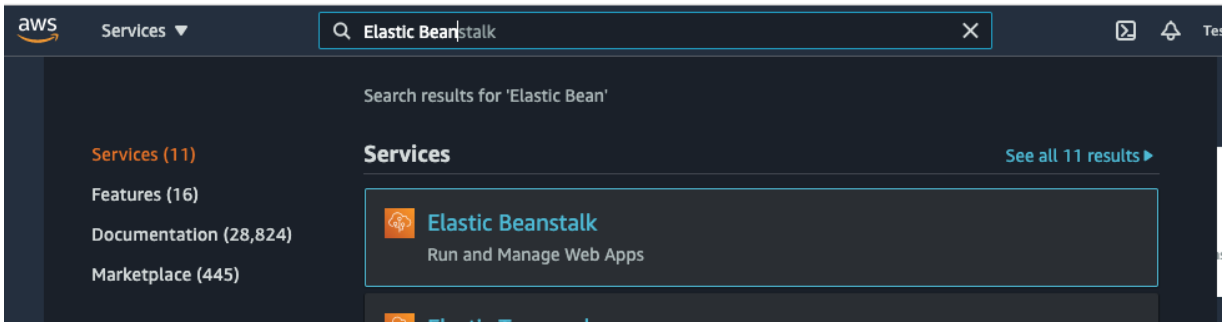
   Make sure you **GO INTO** the folder and select the files for zipping. **NOT** zipping the project folder.



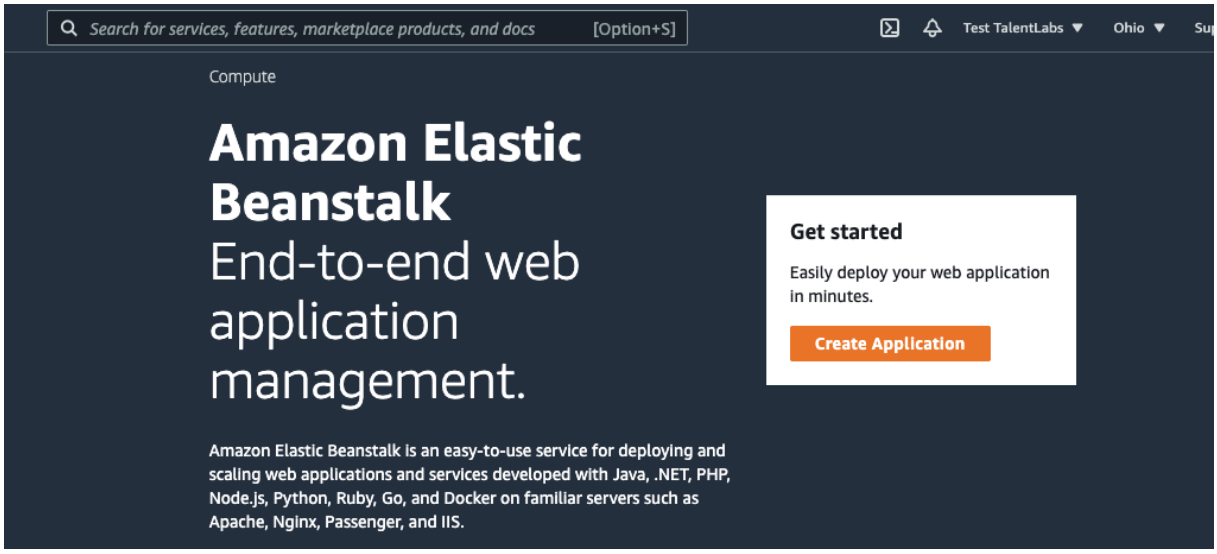
Deployment

Steps:

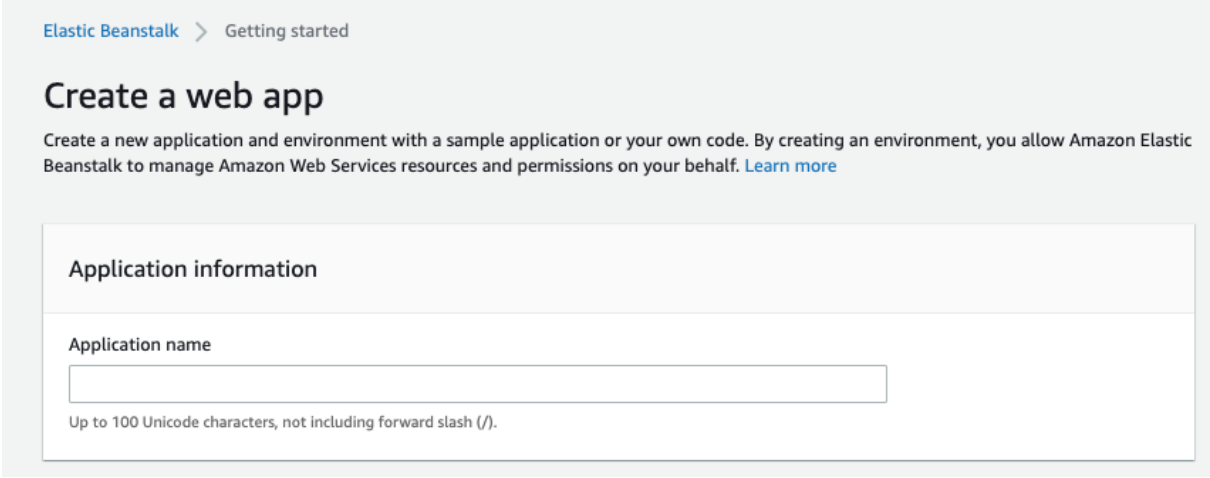
1. Login to the AWS Console and search for Elastic Beanstalk.



2. Click “Create Application” on the Elastic Beanstalk page. You might see a different interface depending if you are a first time user.



3. Then you should see a form like this:

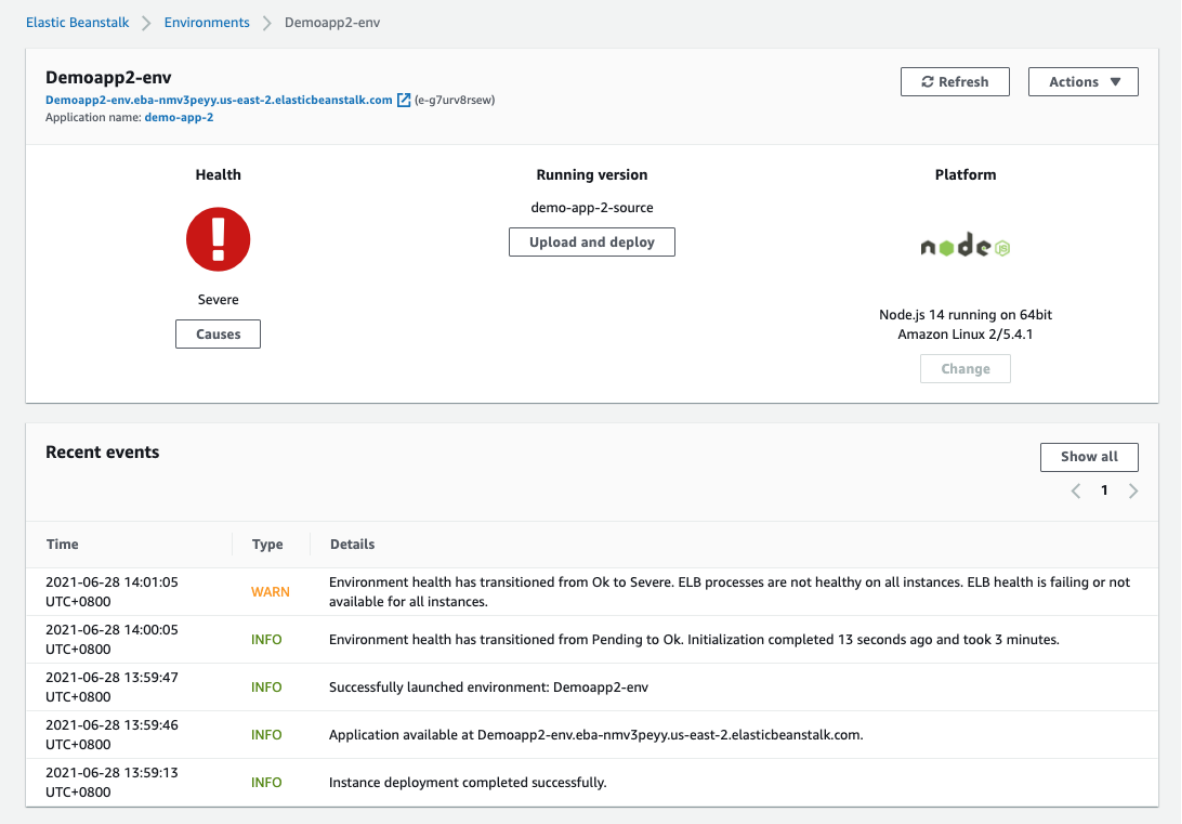


4. Fill the following essential fields in the form. Then click “Create application”

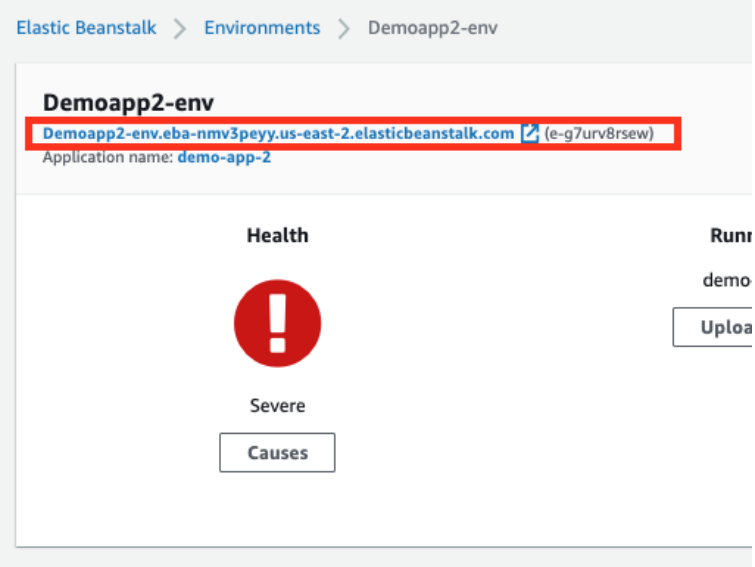
Field	Value
Application Name	Any name you like
Platform - Platform	Node.js
Platform - Platform Branch	Node.js 14 running on 64bit Amazon Linux 2
Platform - Platform version	Pick the “Recommended” version
Application code	Upload your code
Source code origin	Local file -> Choose file -> then pick the zip files you created

5. Wait for a few minutes for the deployment. Then you will be redirected to this page.

   Don’t be scared about the red exclamation mark saying health is “Severe”. This is because we didn’t set up the health check mechanism only. As long as the “Recent events” session doesn’t have anything red, then your deployment should be okay.



6. Next, we are going to test our app and see if it is working. Click on the link generated on the page, and see if you can see your app up and running. If it is working, then copy and paste the link to the table in Part 3.



4.3 Deploying Static HTML Webpage on AWS S3

There are tons of methods in deploying web applications. As a software developer, you should have the capability to read through the documentation and follow the instructions.

For well-known platforms like AWS, they will always provide the latest step by step guide on their website for deployment and the usage of their services. As a software engineer, you can just follow the steps outlined on their website in most of the cases.

One of the common and easy ways of deploying static HTML web pages is leveraging the AWS S3 static website services. The process is very simple, and AWS would provide a free URL for your webpage.

We are not listing the detailed instructions here so you can practice reading documentation. If you find this documentation hard to understand, you can also try the second link we listed below.

Step-by-step guide of hosting static website on AWS S3:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/HostingWebsiteOnS3Setup.html

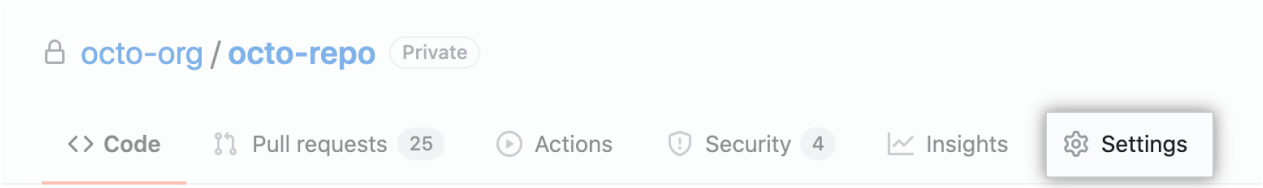
4.4 Deploying Static HTML Webpage on GitHub

Although GitHub is mostly used for storing code, GitHub also provides a handy cloud feature for users to host simple HTML pages. Most users would use this feature to host their personal website or a project introduction page.

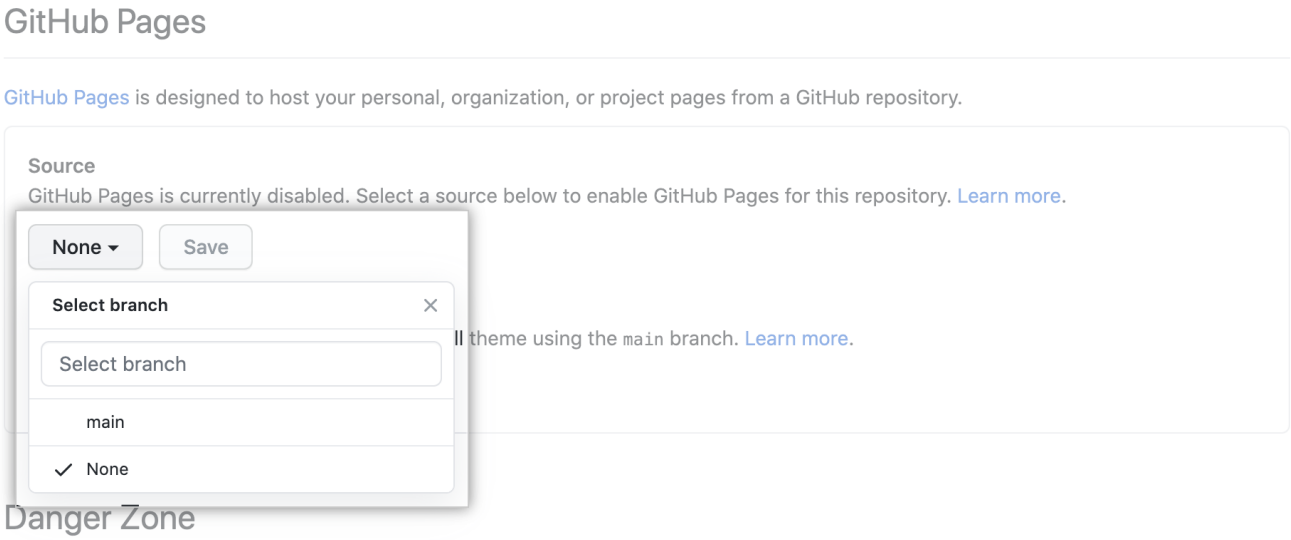
To use this functionality, the steps are very simple.

Step 1: Push your static website files to a GitHub repository. Make sure that you are not wrapping the website with any folders, i.e. index.html should be at the root of the GitHub repository.

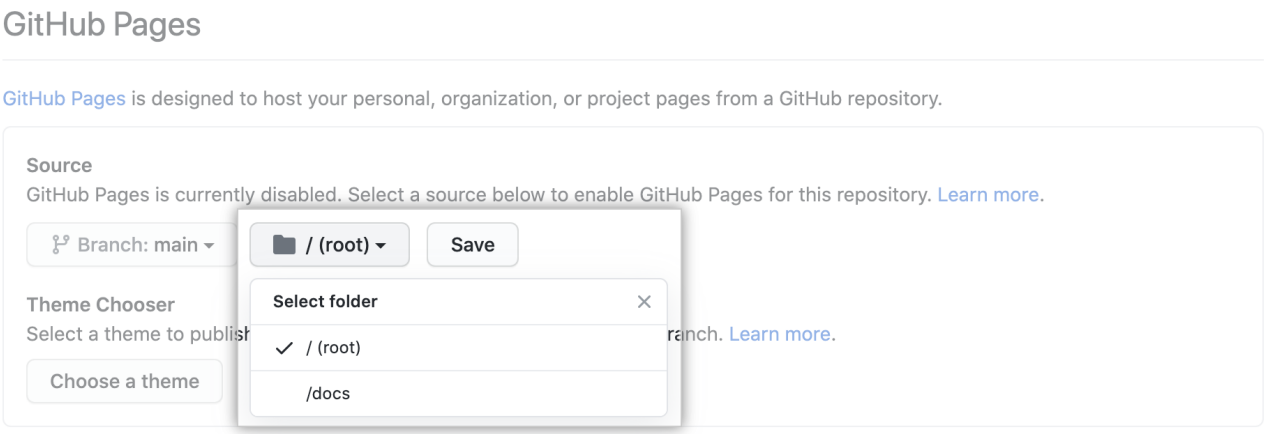
Step 2: Go to the “Settings” page of your GitHub repository.



Step 3: In the "Code and automation" section of the sidebar, click “Pages”.
Step 4: Under "GitHub Pages", use the None or Branch drop-down menu and select a publishing source. For our setup, you should pick “main” or “master”

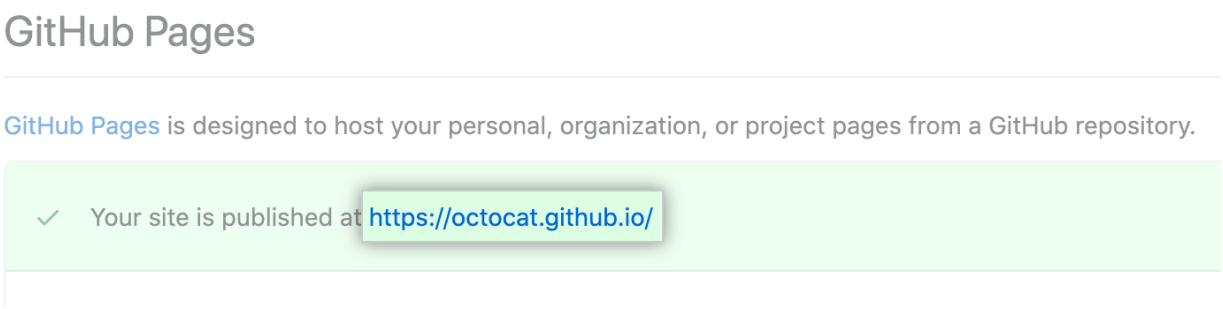


Step 5: Use the drop-down menu to select a folder for your publishing source.



Step 6: Click “Save”.
Step 7: There should be a green box showing up, with the link to your published site. To see your published site, under "GitHub Pages", click your site's URL.

Do note that it might take a few minutes before your site is ready. If you are seeing an not found message after clicking into the link, you might want to try again in a few minutes.



Part 5 Conclusions

Hopefully, after reading this chapter and completing the assignment, you should have an understanding on the different ways of deploying an application to the cloud. There are still many different ways of deploying an application to Cloud that we didn't cover in the chapter. No one could memorize all of these methods. What is more important is the skills of reading through different deployment documentations and being able to follow the instructions.

In the future, when you are facing a new platform or new way of deploying applications, just stay calm and read through the steps, and you should be able to deploy any applications on any platform. Good luck!