Advanced DevOps

AIM: To Build an Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Application on EC2 instance using AWS CodeDeploy.

Pre-Experiment Exercise:

Brief Theory:

AWS CodePipeline:

AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.

CICD:

CodePipeline is a *continuous delivery* service that automates the building, testing, and deployment of your software into production.

Continuous delivery is a software development methodology where the release process is automated. Every software change is automatically built, tested, and deployed to production. Before the final push to production, a person, an automated test, or a business rule decides when the final push should occur. Although every successful software change can be immediately released to production with continuous delivery, not all changes need to be released right away.

Continuous integration is a software development practice where members of a team use a version control system and frequently integrate their work to the same location, such as a main branch. Each change is built and verified to detect integration errors as quickly as possible. Continuous integration is focused on automatically building and testing code, as compared to *continuous delivery*, which automates the entire software release process up to production.

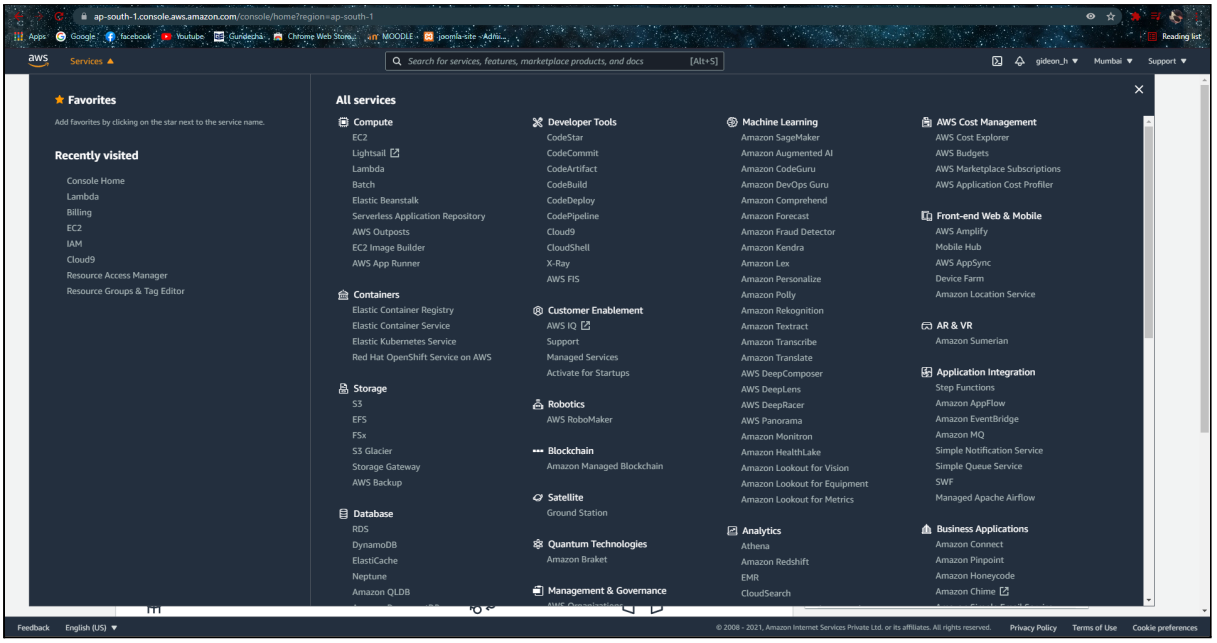
Features:

You can use CodePipeline to help you automatically build, test, and deploy your applications in the cloud. Specifically, you can:

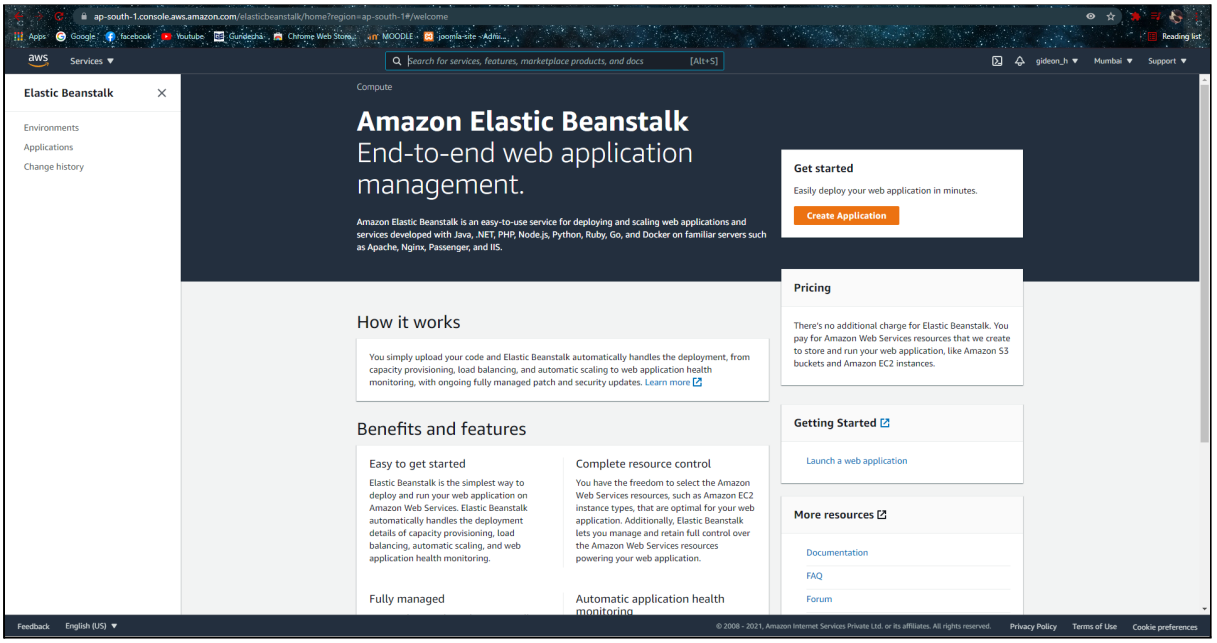
1. **Automate your release processes**: CodePipeline fully automates your release process from end to end, starting from your source repository through build, test, and deployment. You can prevent changes from moving through a pipeline by including a manual approval action in any stage except a Source stage. You can release when you want, in the way you want, on the systems of your choice, across one instance or multiple instances.
2. **Establish a consistent release process**: Define a consistent set of steps for every code change. CodePipeline runs each stage of your release according to your criteria.
3. **Speed up delivery while improving quality**: You can automate your release process to allow your developers to test and release code incrementally and speed up the release of new features to your customers.
4. **Use your favorite tools**: You can incorporate your existing source, build, and deployment tools into your pipeline. For a full list of AWS services and third-party tools currently supported by CodePipeline..
5. **View progress at a glance**: You can review real-time status of your pipelines, check the details of any alerts, retry failed actions, view details about the source revisions used in the latest pipeline execution in each stage, and manually rerun any pipeline.
6. **View pipeline history details**: You can view details about executions of a pipeline, including start and end times, run duration, and execution IDs.

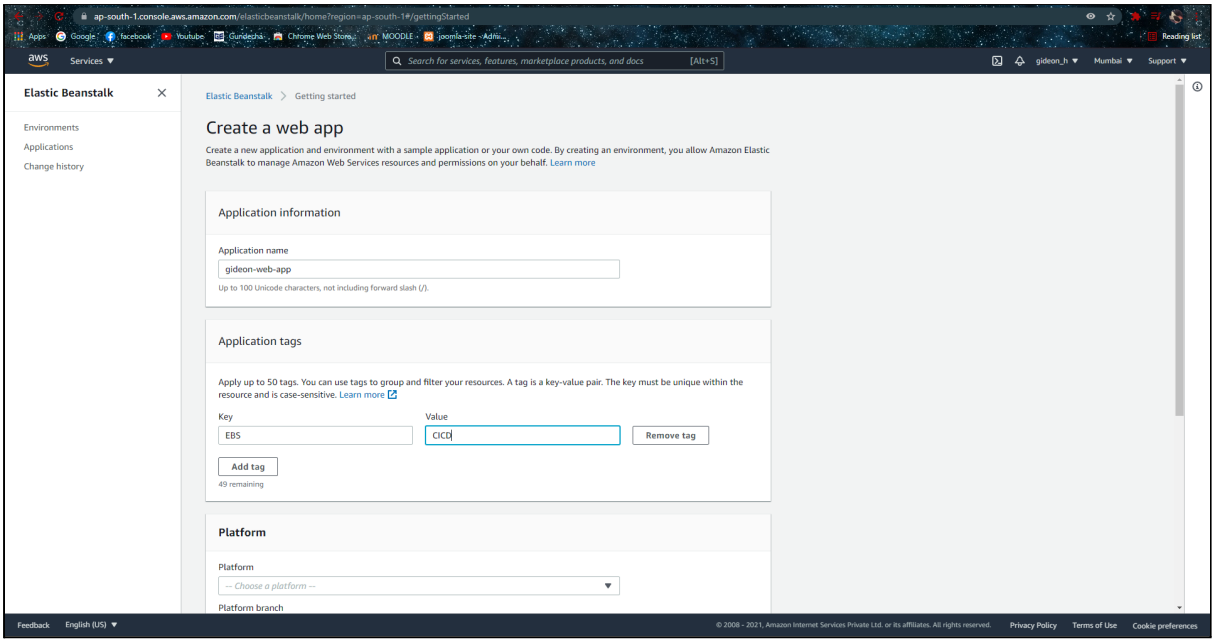
You will create the pipeline using AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change. You will use your GitHub account, an Amazon Simple Storage Service (S3) bucket, or an AWS CodeCommit repository as the source location for the sample app's code. You will also use AWS Elastic Beanstalk as the deployment target for the sample app. Your completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update your live sample app.

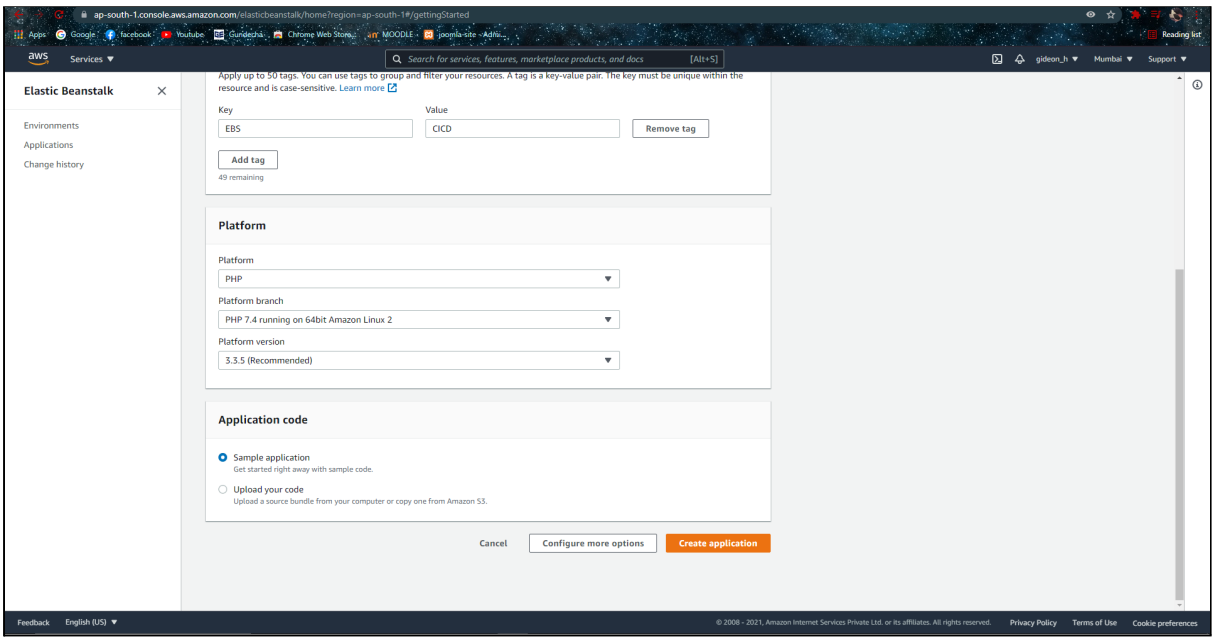
Step1: Create a deployment environment



2) Name your web app and choose PHP from the drop-down menu(or any other language you are interested in) and then click Create Application.

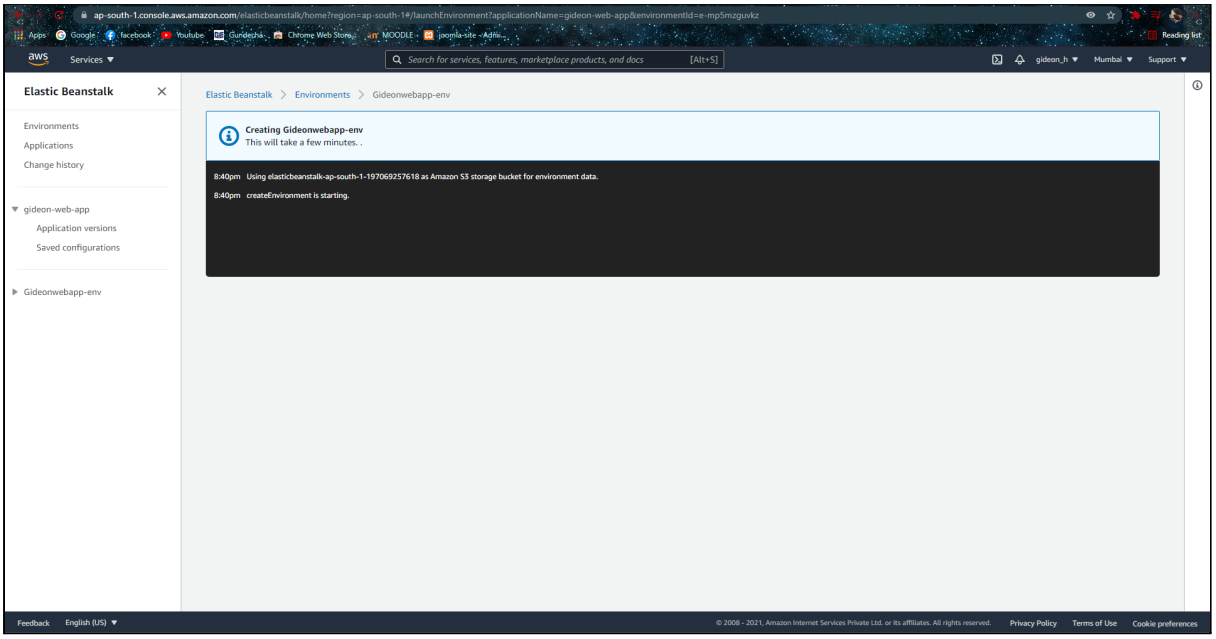






3)Elastic Beanstalk will begin creating a sample environment for you to deploy your application to. It will create an Amazon EC2 instance, a security group, an Auto Scaling group, an Amazon S3 bucket, Amazon CloudWatch alarms, and a domain name for your application.

Note: This will take several minutes to complete.



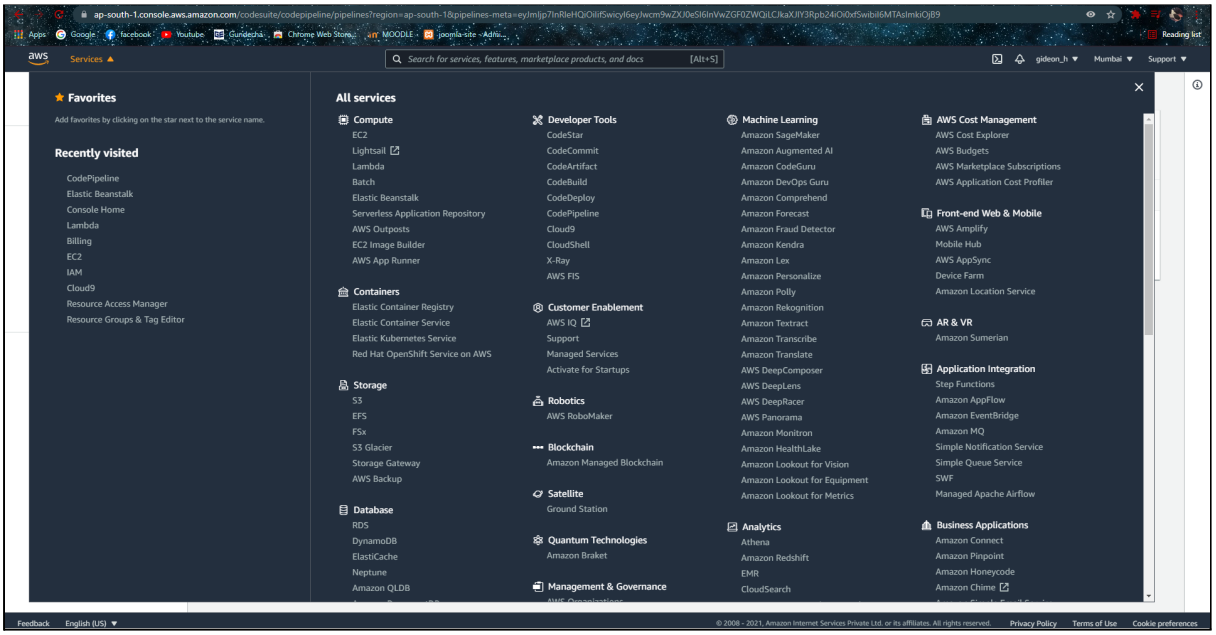
Step 2: Get the copy of Sample Code

In this step you will retrieve the sample app's code and choose a source to host the code. Pipeline takes the code and performs actions on it.

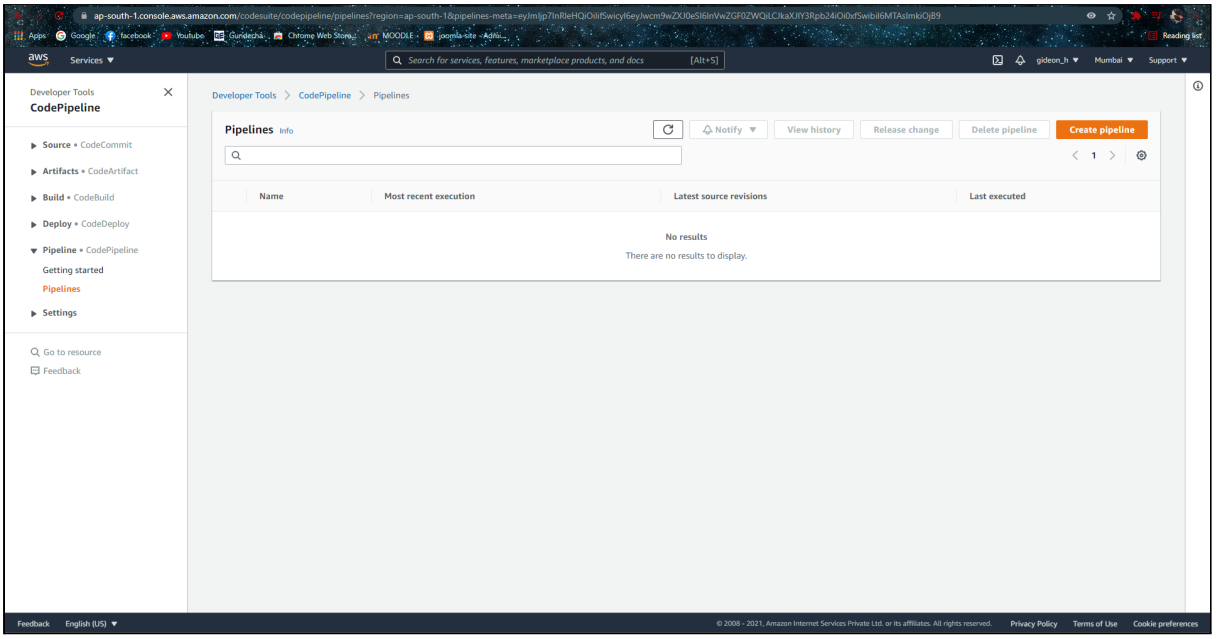
You can use three options.

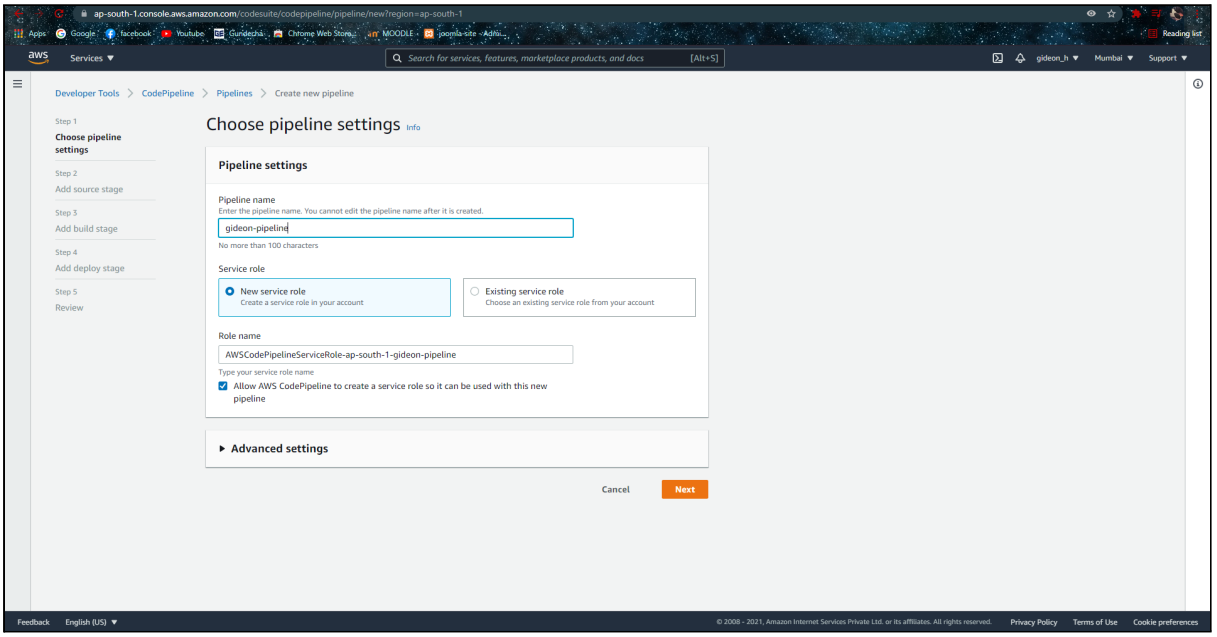
- GitHub Repository
- Amazon S3 Bucket
- AWS CodeCommit Repository

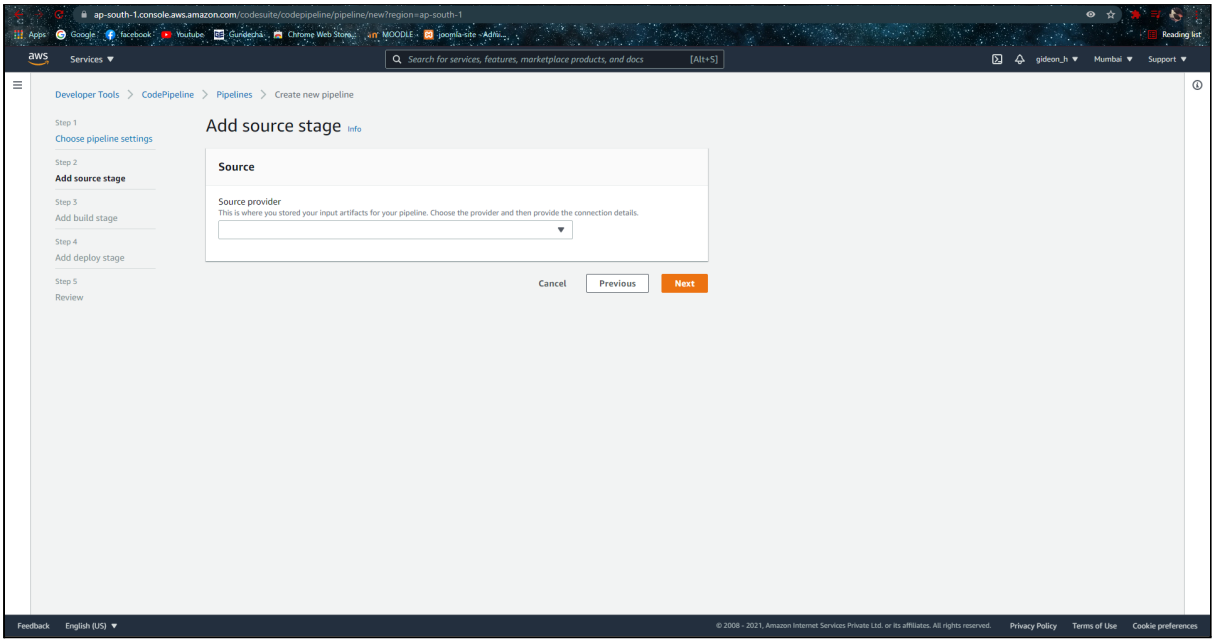
1) Open AWS CodePipeline service(from Developer Tools)



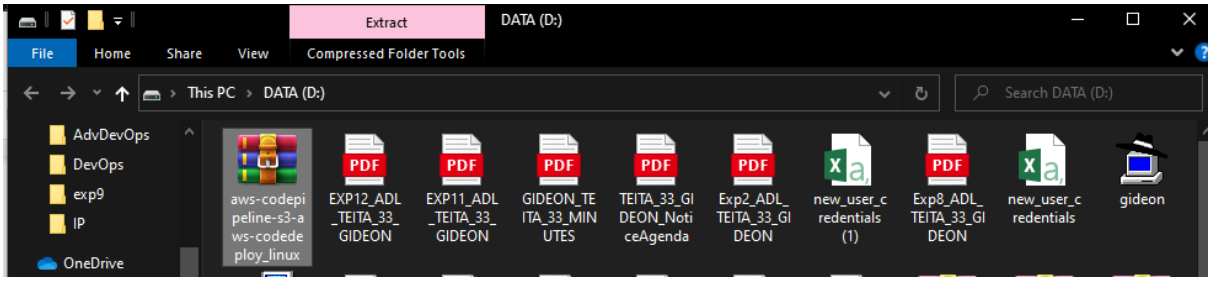
2)Create Pipeline and set settings to it.



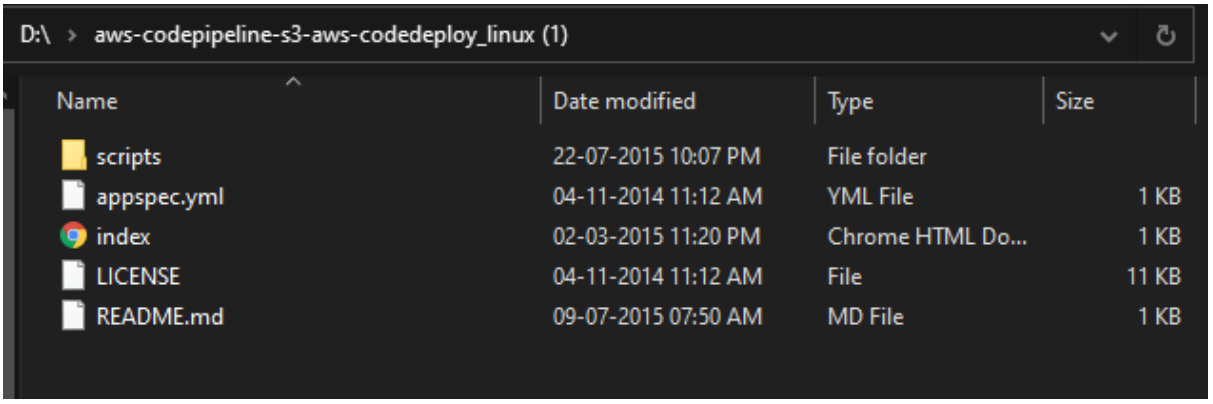




We plan to use Amazon S3 as source



Zip file contents:



YML FILE:

```yaml
version: 0.0
os: linux
files:
  - source: /index.html
    destination: /var/www/html/
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies
      timeout: 300
      runas: root
    - location: scripts/start_server
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server
      timeout: 300
      runas: root
```

HTML FILE

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Sample Deployment</title>
  <style>
    body {
      color: #ffffff;
      background-color: #007f3f;
      font-family: Arial, sans-serif;
      font-size: 14px;
    }

    h1 {
      font-size: 500%;
      font-weight: normal;
```

```
      margin-bottom: 0;
    }

    h2 {
      font-size: 200%;
      font-weight: normal;
      margin-bottom: 0;
    }
  </style>
</head>
<body>
  <div align="center">
    <h1>Congratulations!</h1>
    <h2>You have successfully created a pipeline that retrieved this
source application from an Amazon S3 bucket and deployed it
    to three Amazon EC2 instances using AWS CodeDeploy.</h2>
    <p>For next steps, read the AWS CodePipeline Documentation.</p>
  </div>
</body>
</html>
```

START SERVER

```
#!/bin/bash
service httpd start
```

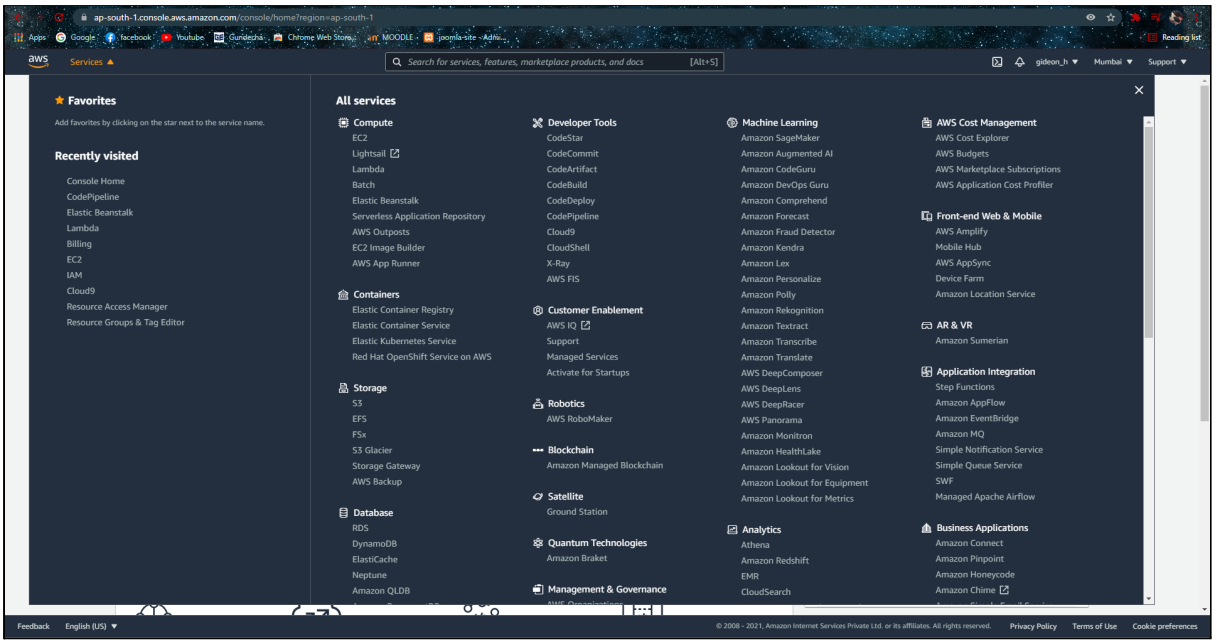
STOP SERVER

```
#!/bin/bash
isExistApp = `pgrep httpd`
if [[ -n  $isExistApp ]]; then
   service httpd stop
fi
```

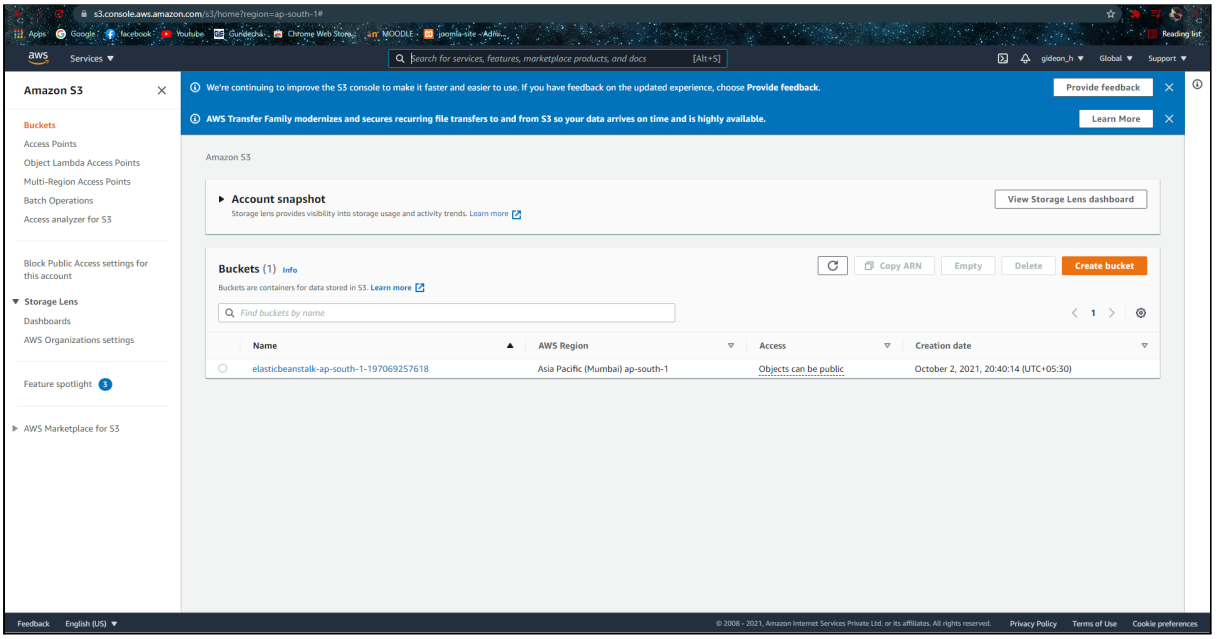
INSTALL DEPENDENCIES

```
#!/bin/bash
yum install -y httpd
```

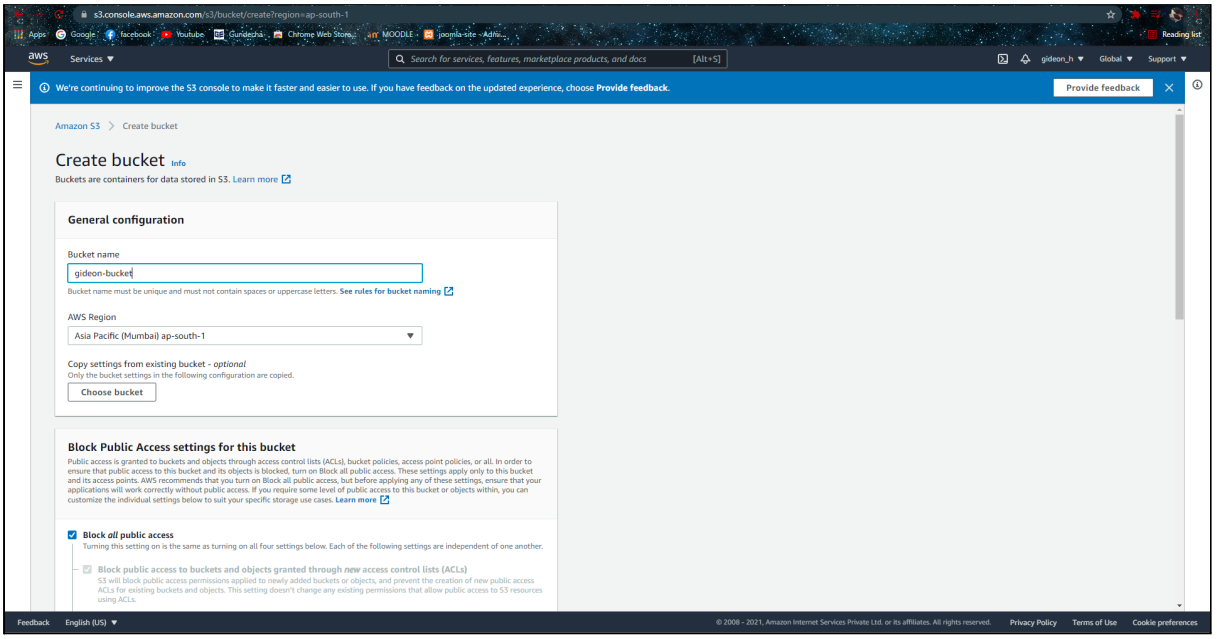
Open Amazon S3 console and Create your S3 Bucket:

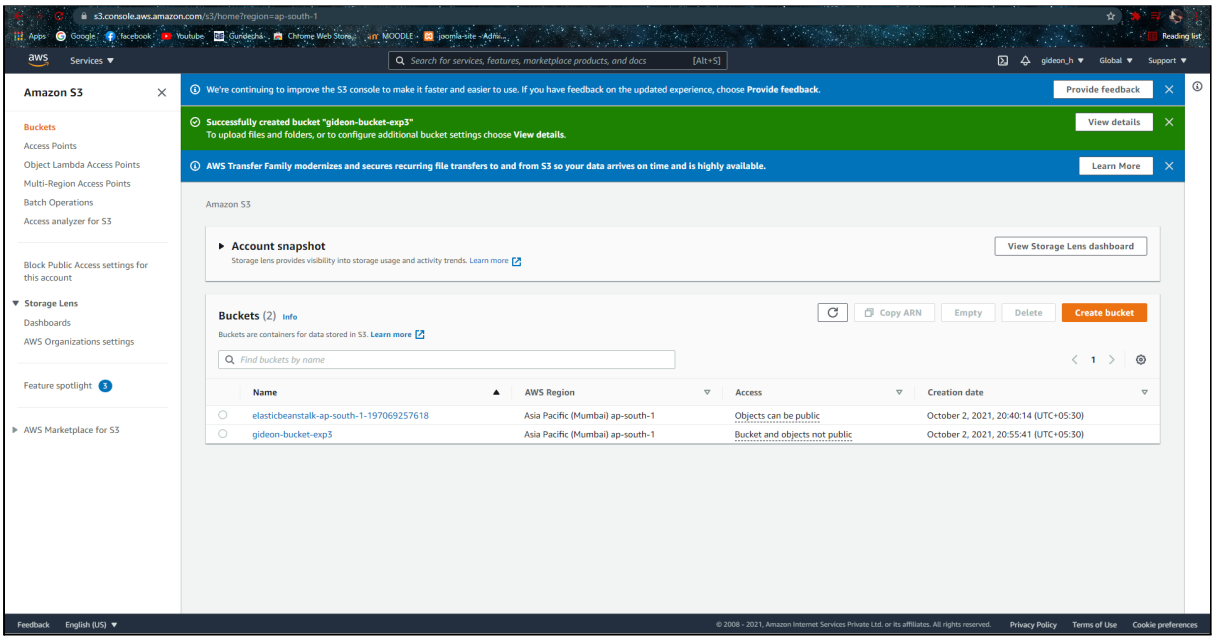


Create Bucket

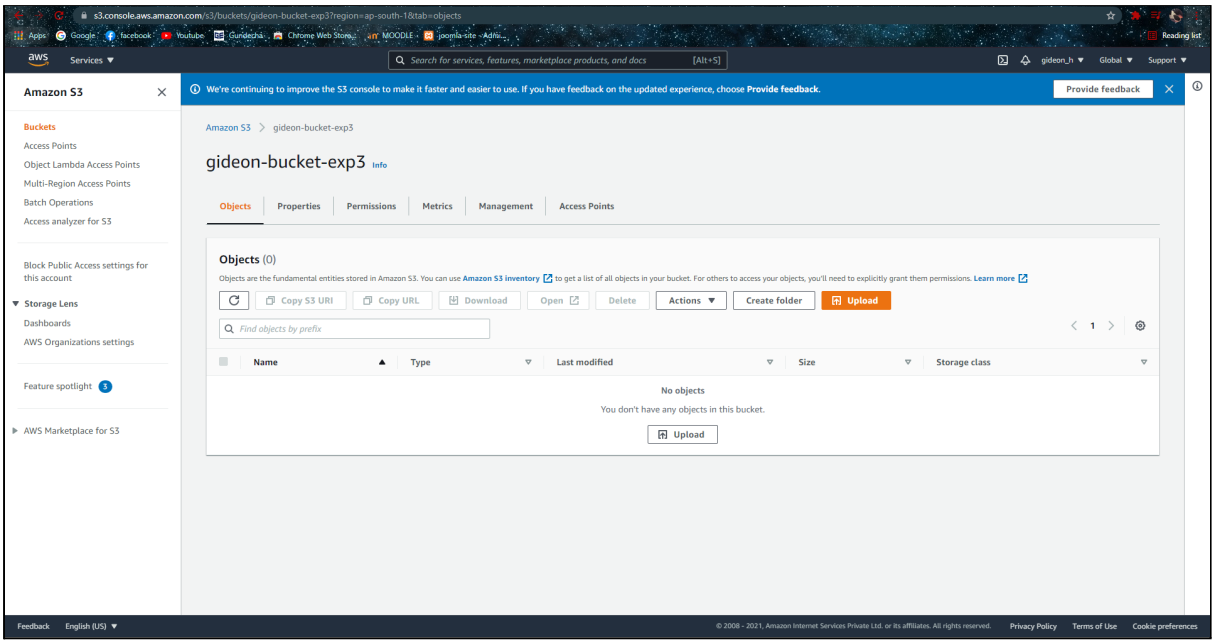


Give Unique name to Bucket and choose default region and Create Bucket



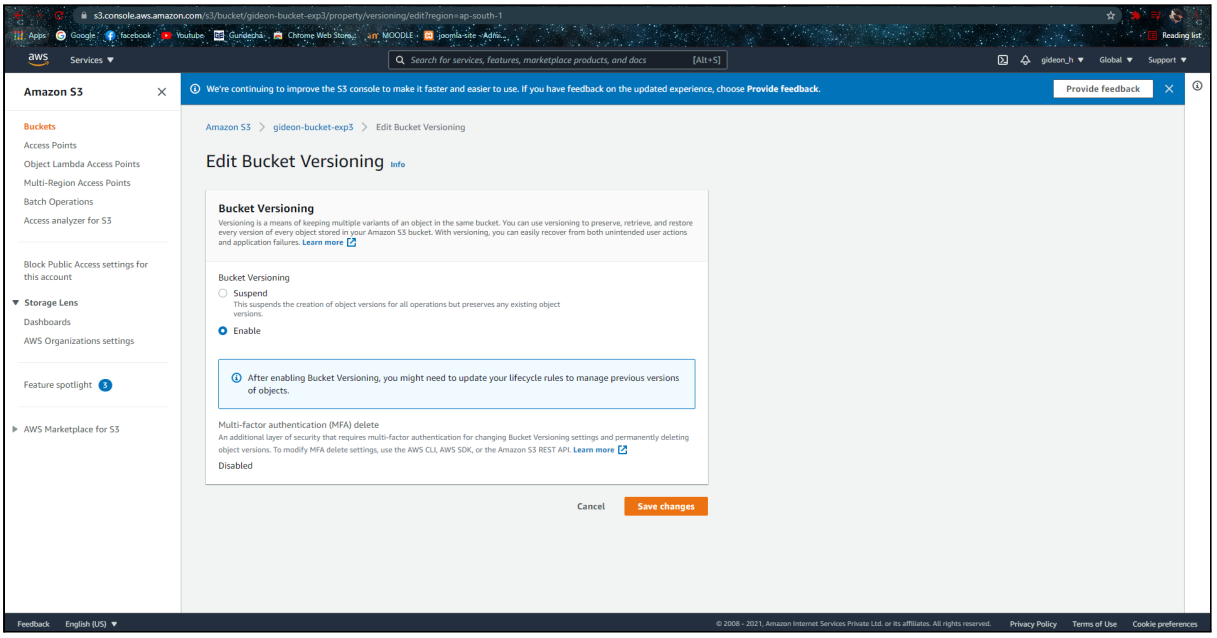


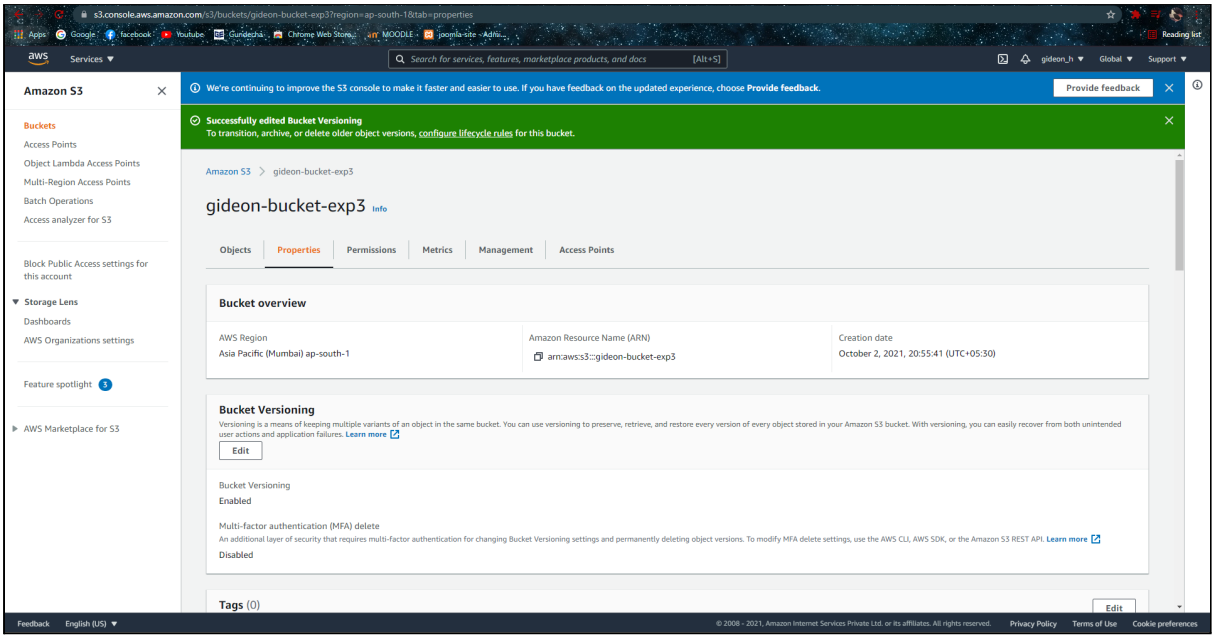
Click on Bucket Name and go to Properties



Edit Bucket Versioning

Click on Bucket Versioning : Enable and save changes.

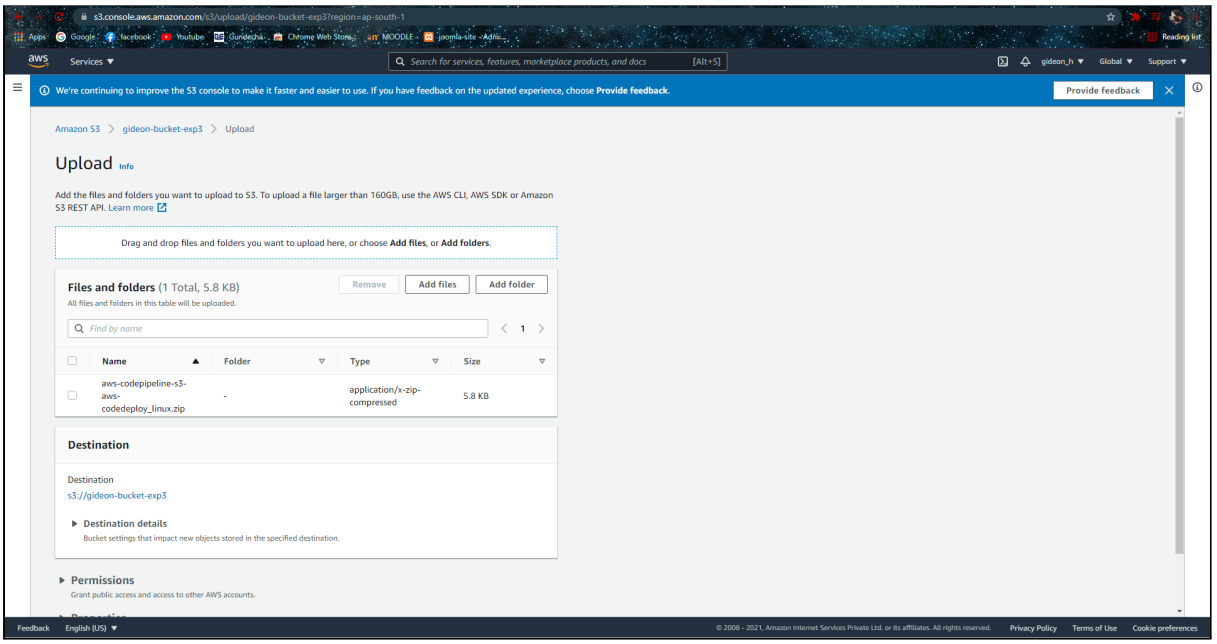


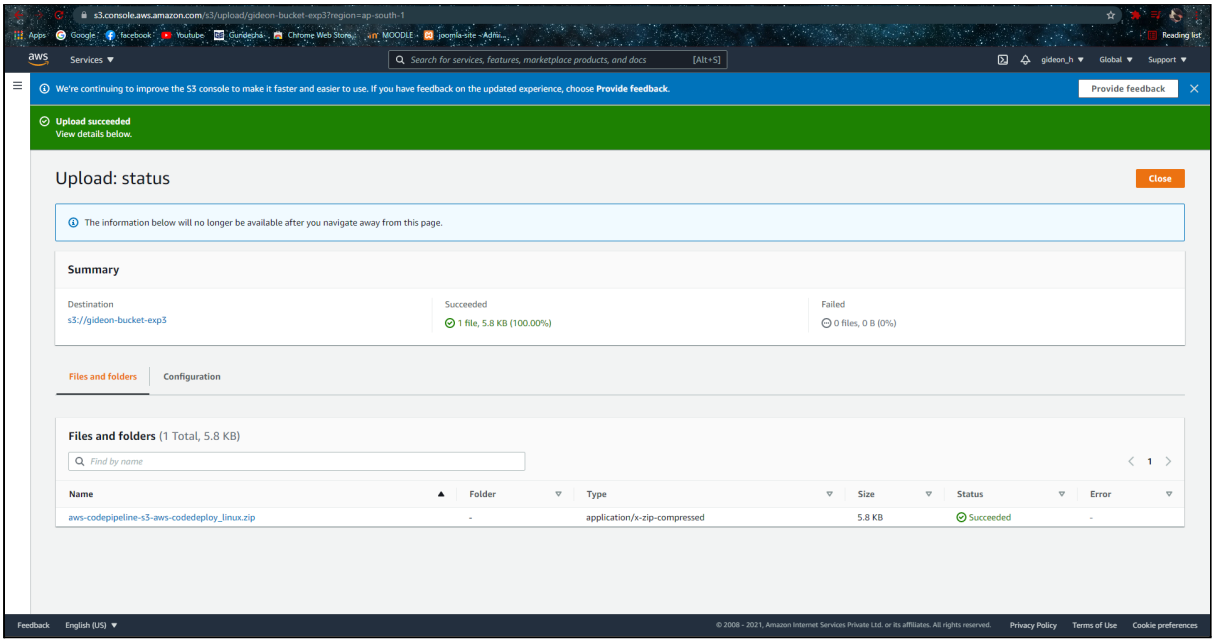


Now Upload the code to Bucket

Select Bucket and Click on Upload (Right Corner)

Add Files -- upload zip file from downloads of your computer. Click on Upload Button





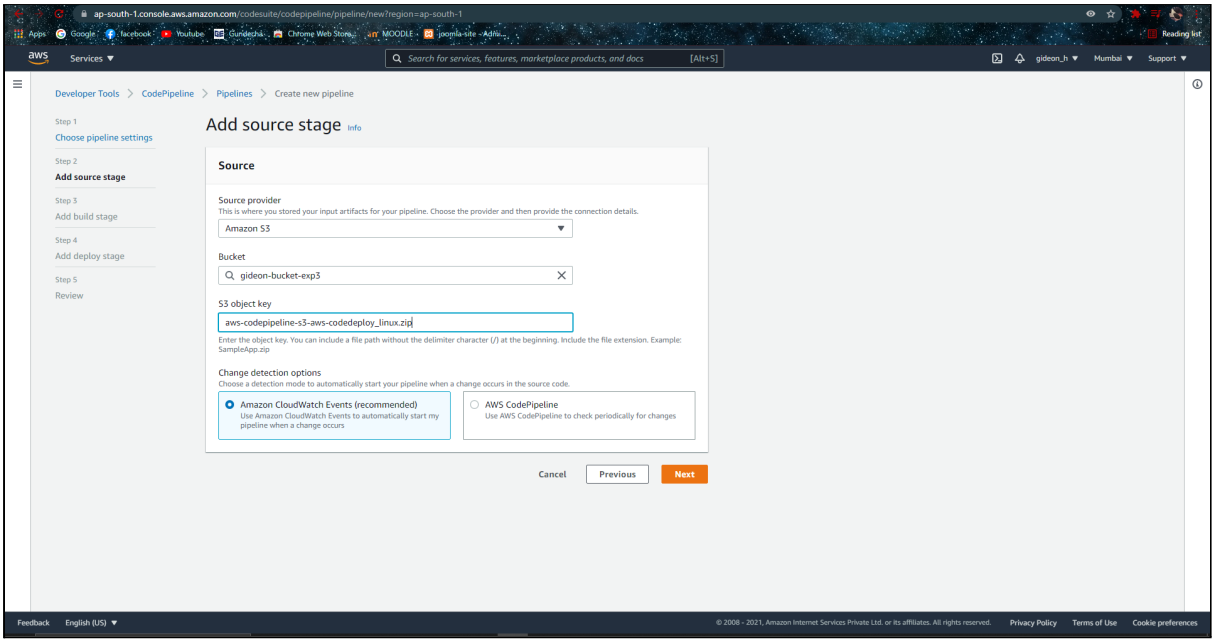
Step 3: Create Pipeline

In this step, you will create and configure a simple pipeline with two actions: source and deploy. You will provide CodePipeline with the locations of your source repository and deployment environment.

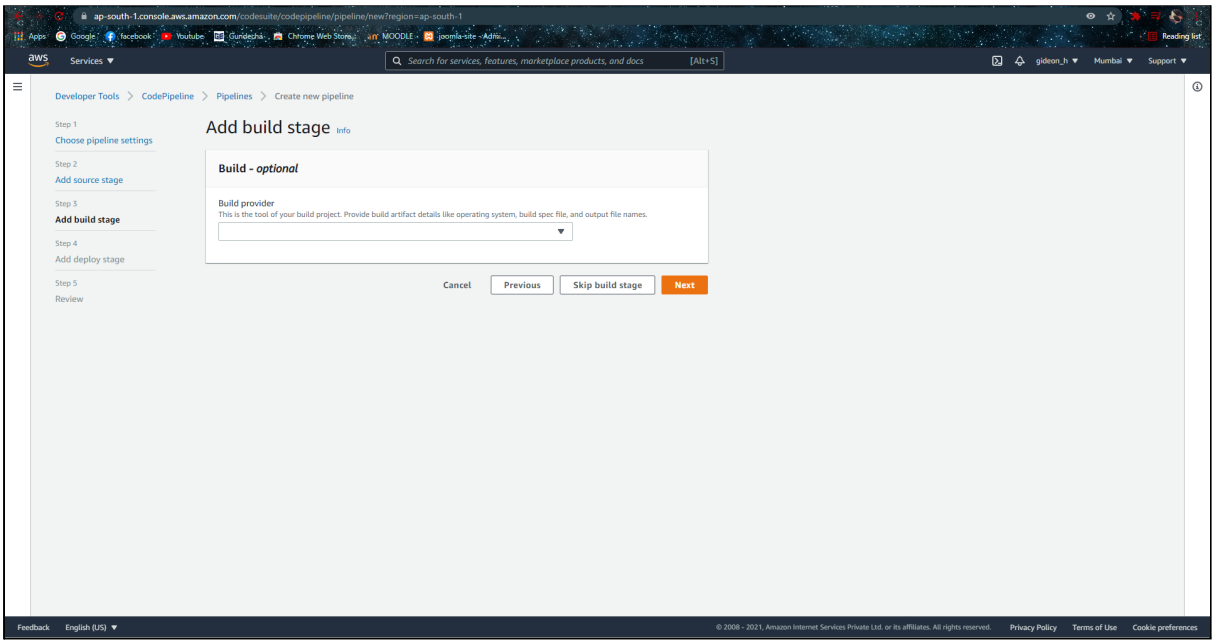
A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this we will skip the build stage.

Goto Pipeline again and create it

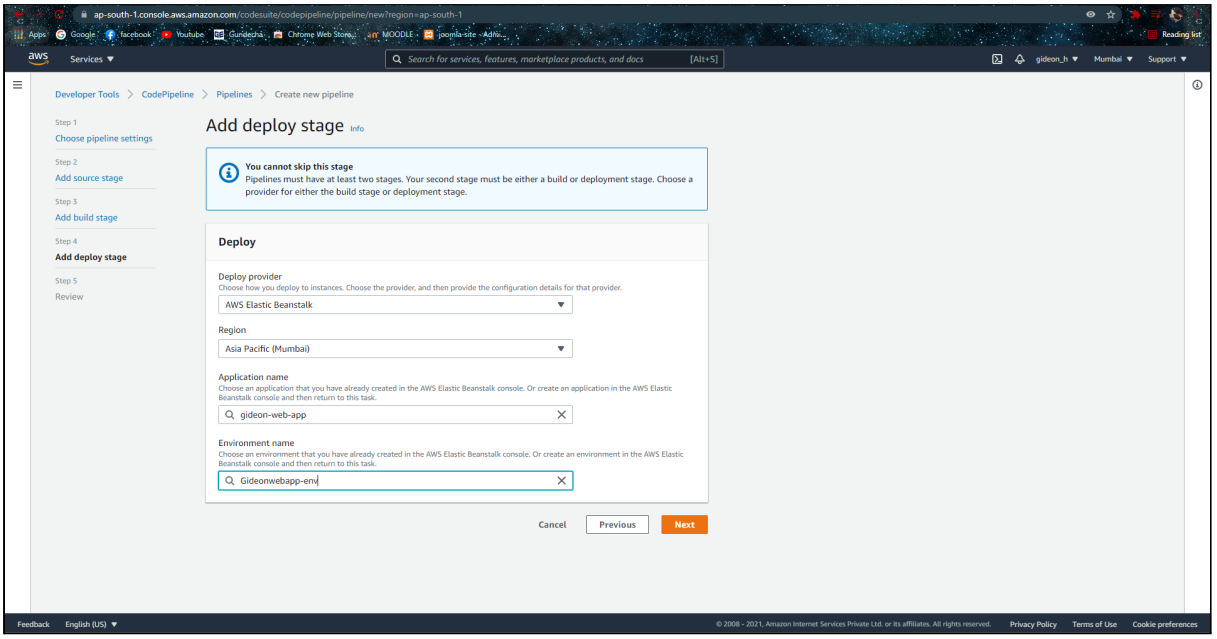
Create new pipeline: Source Provider = Amazon S3, Bucket = . . . , S3 Object Key = Copy S3 Uri from Amazon S3 bucket.



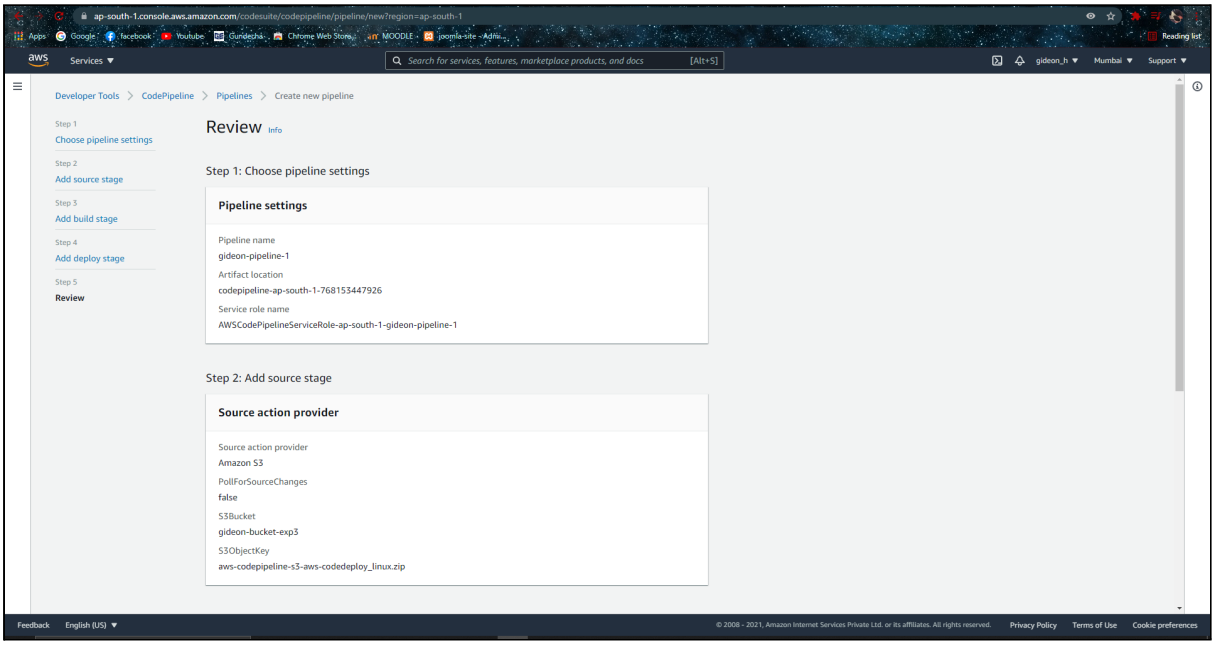
Skip Build Stage

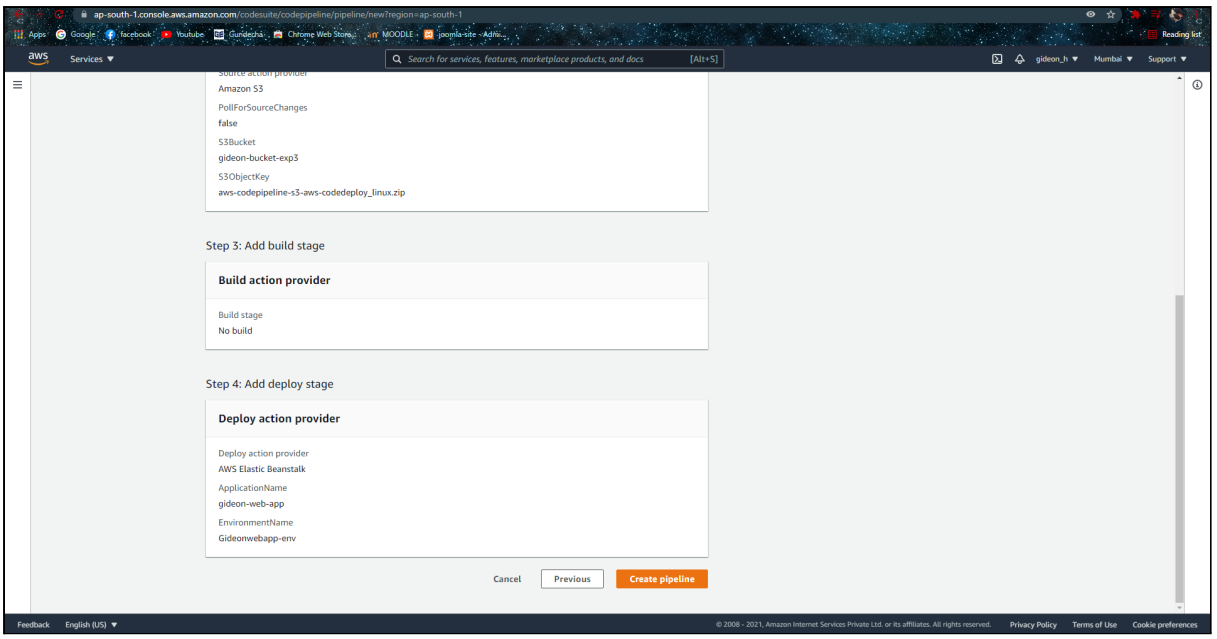


Add Deploy Stage:



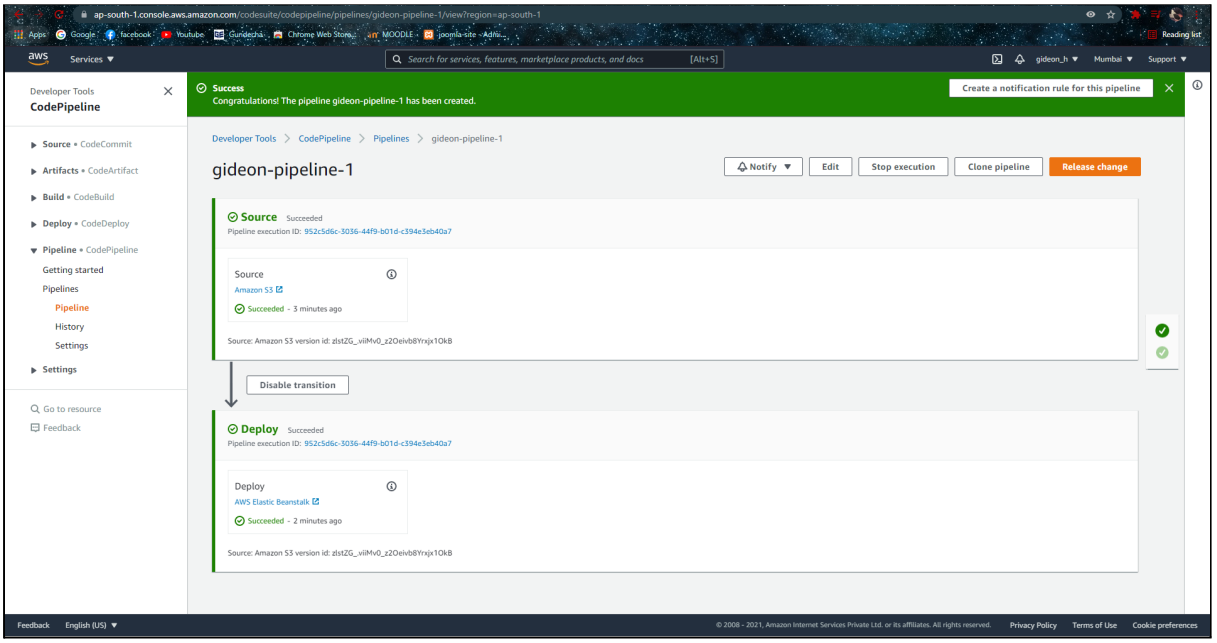
Review the settings and create the pipeline.



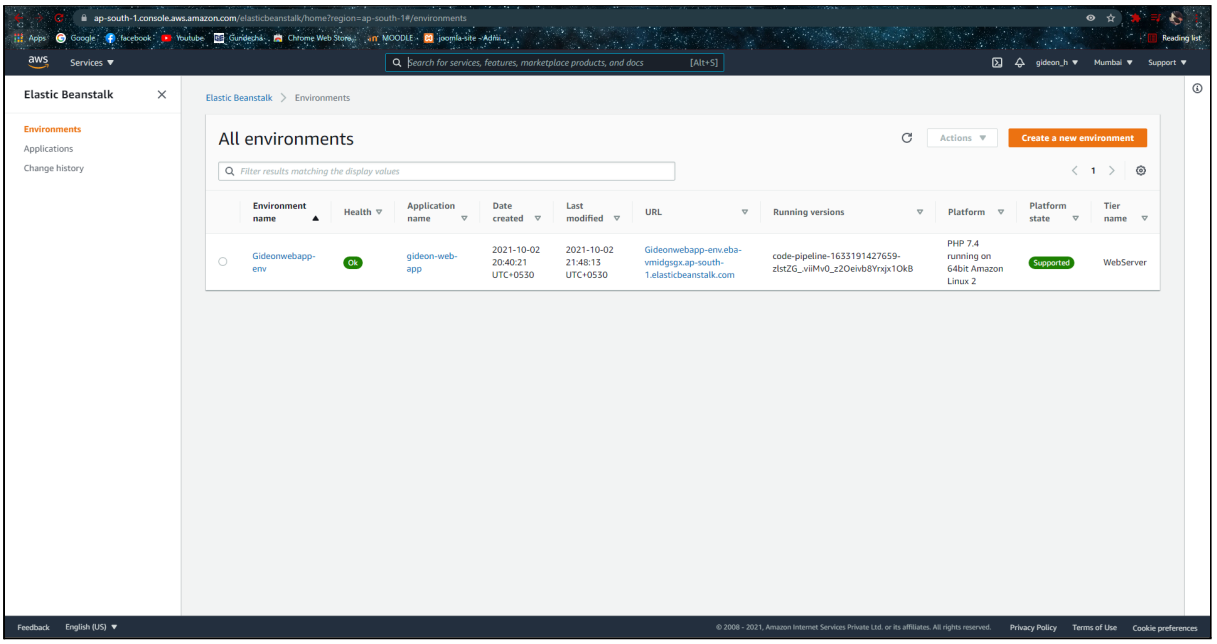


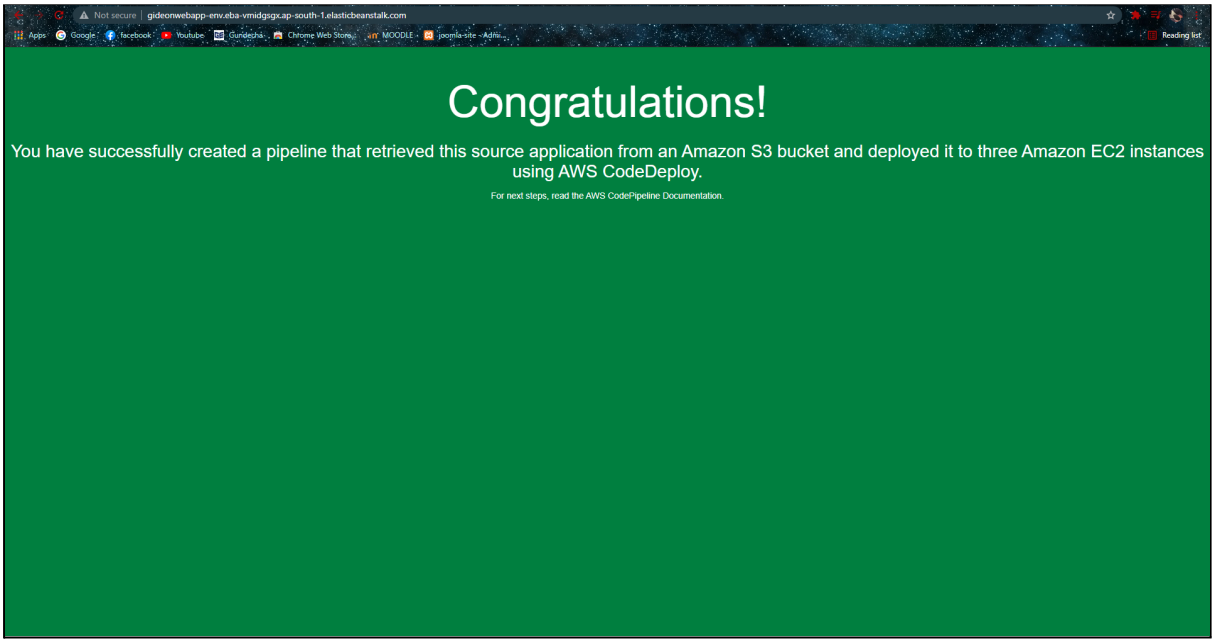
Add deploy stage and Give details : Deployement provider : AWS Elastic Beanstalk, Region,Application Name and Environment Name…(Auto suggested.)

Pipeline created successfully.



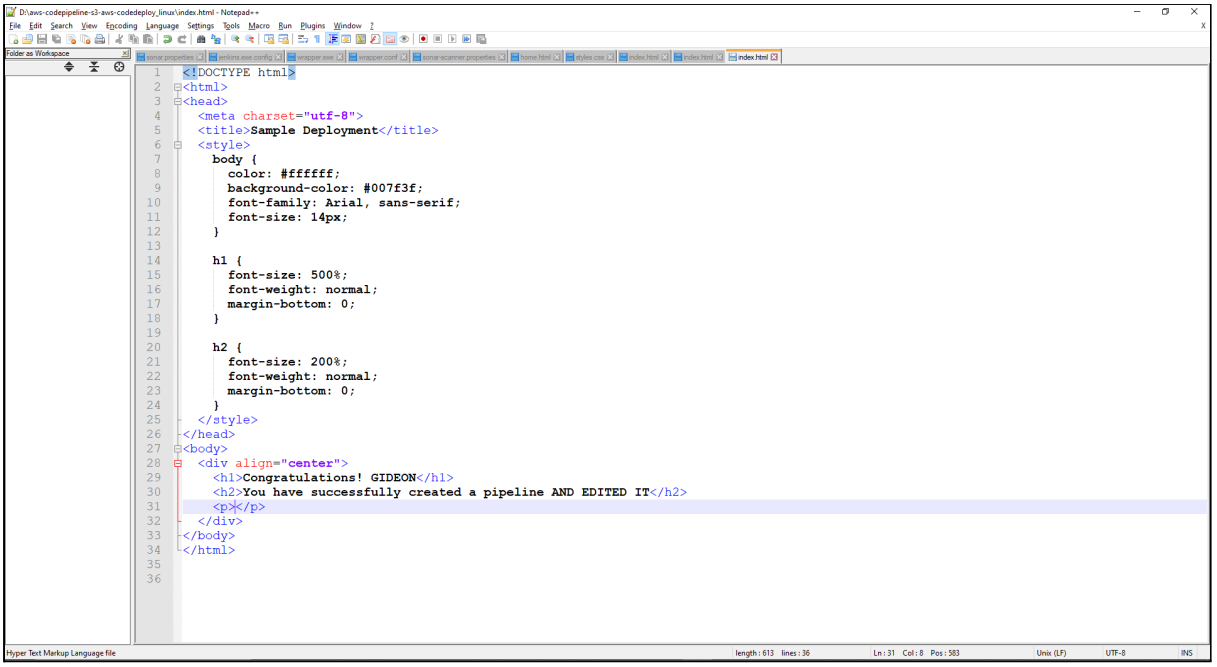
Now go to your EBS environment and click on the URL to view the sample website you deployed.





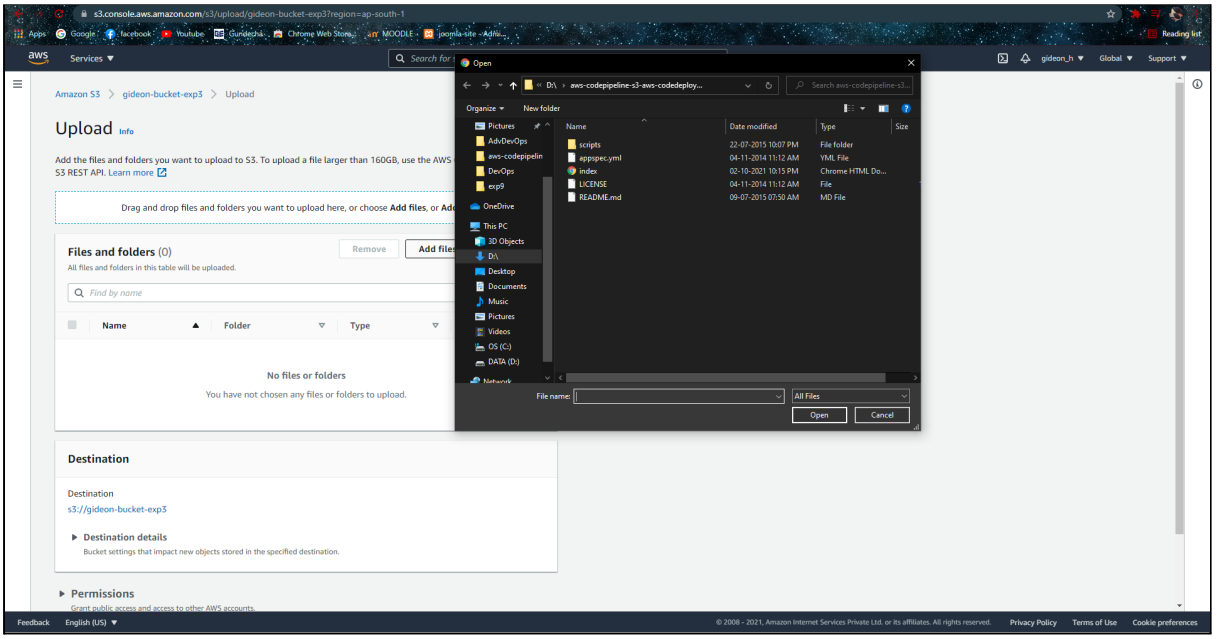
In this step, you will revise the sample code and commit the change to your repository. CodePipeline will detect your updated sample code and then automatically initiate deploying it to your Ec2 instance.

Step 5: Commit a change and then update your app

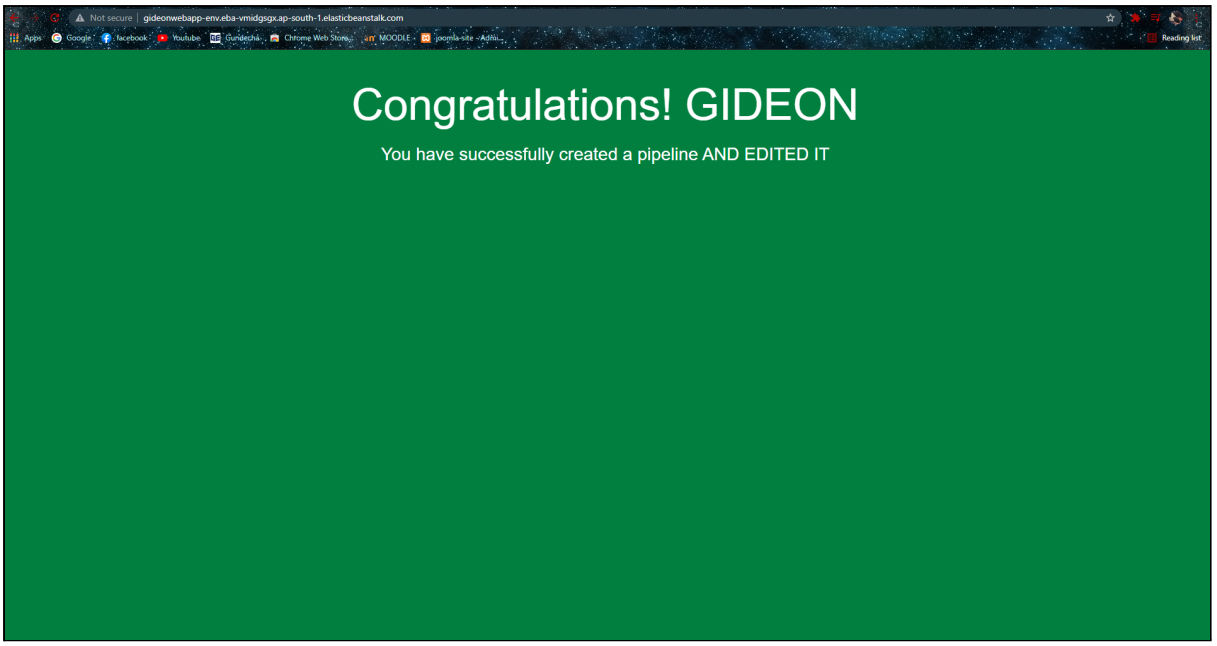


```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Sample Deployment</title>
  <style>
    body {
      color: #ffffff;
      background-color: #007f3f;
      font-family: Arial, sans-serif;
      font-size: 14px;
    }

    h1 {
      font-size: 500%;
      font-weight: normal;
      margin-bottom: 0;
    }

    h2 {
      font-size: 200%;
      font-weight: normal;
      margin-bottom: 0;
    }
  </style>
</head>
<body>
  <div align="center">
    <h1>Congratulations! GIDEON</h1>
    <h2>You have successfully created a pipeline AND EDITED IT</h2>
    <p></p>
  </div>
</body>
</html>
```

uploading the updated zip file

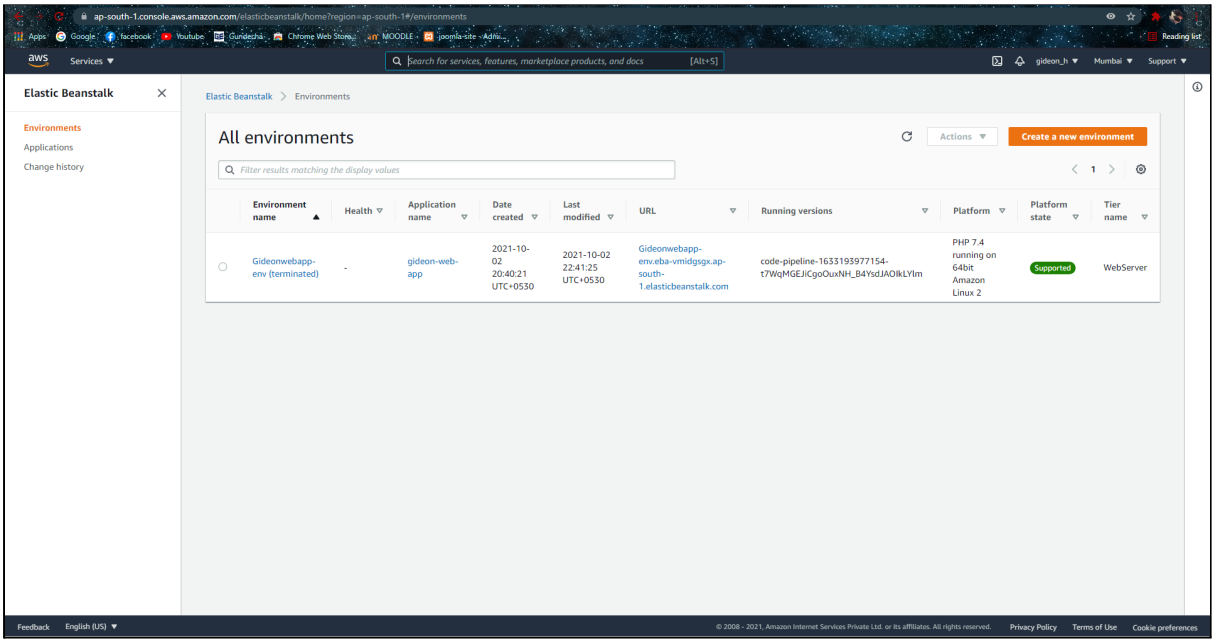


RELOADING WEB APP



Step 6: Clean up the resources

all resources terminated



A. Extended Theory:

- **Deploy Web Application using Elastic BeanStalk**

You can use the AWS Elastic Beanstalk console to upload an updated source bundle and deploy it to your Elastic Beanstalk environment, or redeploy a previously uploaded version.

Each deployment is identified by a deployment ID. Deployment IDs start at 1 and increment by one with each deployment and instance configuration change. If you enable enhanced health reporting, Elastic Beanstalk displays the deployment ID in both the health console and the EB CLI when it reports instance health status. The deployment ID helps you determine the state of your environment when a rolling update fails.

Elastic Beanstalk provides several deployment policies and settings. For details about configuring a policy and additional settings, see Deployment policies and settings. The following table lists the policies and the kinds of environments that support them.

Supported deployment policies

Deployment policy	Load-balanced environments	Single-instance environments	Legacy Windows Server environments†
All at once	✓ Yes	✓ Yes	✓ Yes
Rolling	✓ Yes	✗ No	✓ Yes
Rolling with an additional batch	✓ Yes	✗ No	✗ No
Immutable	✓ Yes	✓ Yes	✗ No
Traffic splitting	✓ Yes (Application Load Balancer)	✗ No	✗ No

The following list provides summary information about the different deployment policies and adds related considerations.

- All at once – The quickest deployment method. Suitable if you can accept a short loss of service, and if quick deployments are important to you. With this method, Elastic Beanstalk deploys the new application version to each instance. Then, the web proxy or application server might need to restart. As a result, your application might be unavailable to users (or have low availability) for a short time.

- Rolling – Avoids downtime and minimizes reduced availability, at a cost of a longer deployment time. Suitable if you can't accept any period of completely lost service. With this method, your application is deployed to your

environment one batch of instances at a time. Most bandwidth is retained throughout the deployment.

- Rolling with additional batch – Avoids any reduced availability, at a cost of an even longer deployment time compared to the Rolling method. Suitable if you must maintain the same bandwidth throughout the deployment. With this method, Elastic Beanstalk launches an extra batch of instances, then performs a rolling deployment. Launching the extra batch takes time, and ensures that the same bandwidth is retained throughout the deployment.

- Immutable – A slower deployment method, that ensures your new application version is always deployed to new instances, instead of updating existing instances. It also has the additional advantage of a quick and safe rollback in case the deployment fails. With this method, Elastic Beanstalk performs an immutable update to deploy your application. In an immutable update, a second Auto Scaling group is launched in your environment and the new version serves traffic alongside the old version until the new instances pass health checks.

- Traffic splitting – A canary testing deployment method. Suitable if you want to test the health of your new application version using a portion of incoming traffic, while keeping the rest of the traffic served by the old application version.

Deploying a new application version

You can perform deployments from your environment's dashboard.

To deploy a new application version to an Elastic Beanstalk environment

1. Open the Elastic Beanstalk console, and in the Regions list, select your AWS Region.

2. In the navigation pane, choose Environments, and then choose the name of your environment from the list.
   Note
   If you have many environments, use the search bar to filter the environment list.

3. Choose Upload and deploy.

4. Use the on-screen form to upload the application source bundle.

5. Choose Deploy.

Redeploying a previous version

You can also deploy a previously uploaded version of your application to any of its environments from the application versions page.

To deploy an existing application version to an existing environment

1. Open the Elastic Beanstalk console, and in the Regions list, select your AWS Region.
2. In the navigation pane, choose Applications, and then choose your application's name from the list.
   Note
   If you have many applications, use the search bar to filter the application list.
3. In the navigation pane, find your application's name and choose Application versions.
4. Select the application version to deploy.
5. Choose Actions, and then choose Deploy.
6. Select an environment, and then choose Deploy.

Other ways to deploy your application

If you deploy often, consider using the Elastic Beanstalk Command Line Interface (EB CLI) to manage your environments. The EB CLI creates a repository alongside your source code. It can also create a source bundle, upload it to Elastic Beanstalk, and deploy it with a single command.

For deployments that depend on resource configuration changes or a new version that can't run alongside the old version, you can launch a new environment with the new version and perform a CNAME swap for a blue/green deployment.

- **AWS Storage Service: S3**

Amazon Simple Storage Service (Amazon S3) is storage for the Internet. It is designed to make web-scale computing easier.

Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, and inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

This introduction to Amazon Simple Storage Service (Amazon S3) provides a detailed summary of this web service. After reading this section, you should have a good idea of what it offers and how it can fit in with your business.

Advantages of using Amazon S3

Amazon S3 is intentionally built with a minimal feature set that focuses on simplicity and robustness. Following are some of the advantages of using Amazon S3:

- Creating buckets – Create and name a bucket that stores data. Buckets are the fundamental containers in Amazon S3 for data storage.
- Storing data – Store an infinite amount of data in a bucket. Upload as many objects as you like into an Amazon S3 bucket. Each object can contain up to 5 TB of data. Each object is stored and retrieved using a unique developer-assigned key.
- Downloading data – Download your data or enable others to do so. Download your data anytime you like, or allow others to do the same.
- Permissions – Grant or deny access to others who want to upload or download data into your Amazon S3 bucket. Grant upload and download permissions to three types of users. Authentication mechanisms can help keep data secure from unauthorized access.
- Standard interfaces – Use standards-based REST and SOAP interfaces designed to work with any internet-development toolkit.

D. References:

https://aws.amazon.com/elasticbeanstalk/faqs/#

https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html

---