Name: Manav Jawrani

Roll No.: 19

Subject: Advanced DevOps

Experiment No.: 2

Experiment 2

Aim: To Build Your Application using AWS CodeBuild and Deploy on S3/SEBS using AWS Codepipeline, deploy sample Application on an EC2 instance using AWS Code Deploy.

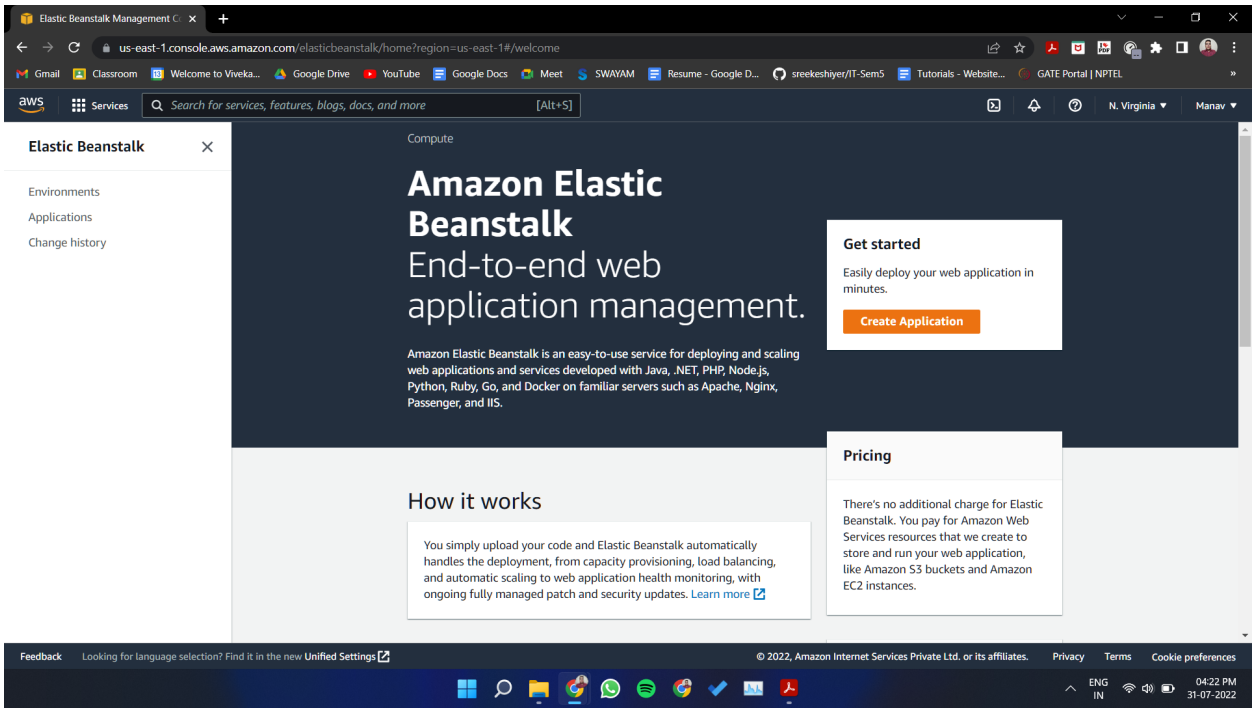
Theory:

Continuous deployment allows you to deploy revisions to a production environment automatically without explicit approval from a developer, making the entire software release process automated. 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 code Commit 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's code. You will also use app. Your completed pipeline will be able to detect changes made to the source respository containing the sample app and then automatically update your live sample app.

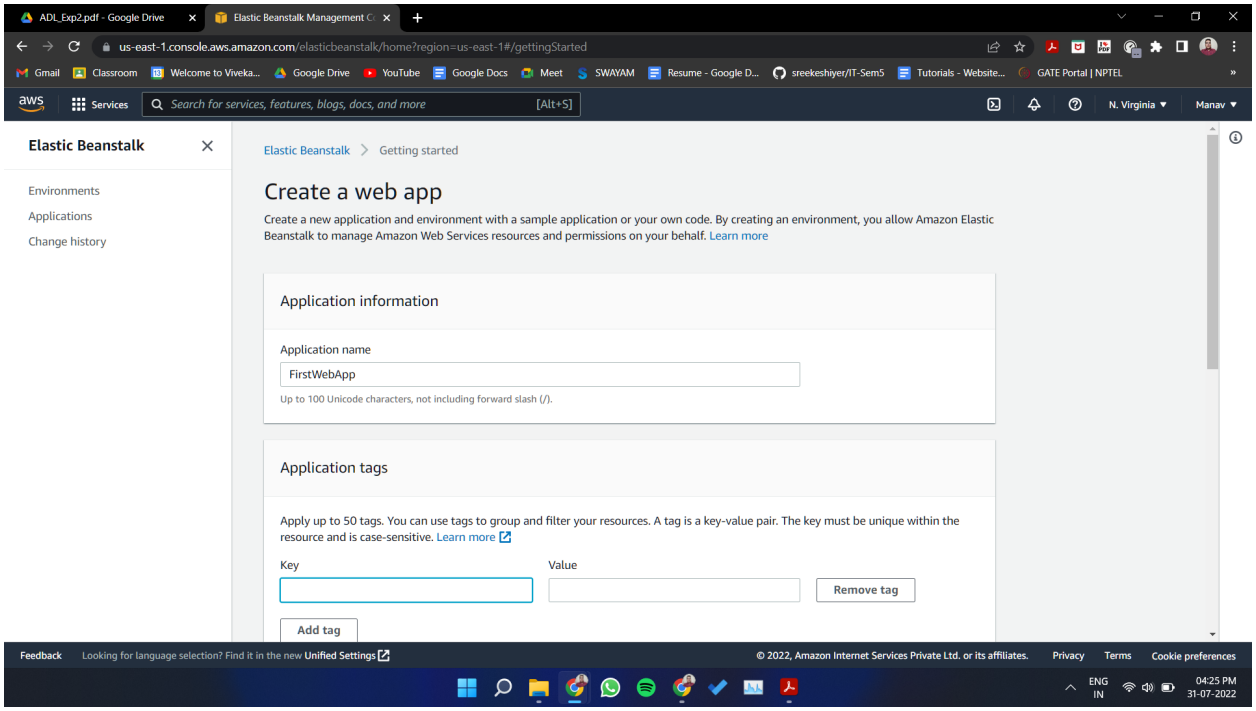
Step 1: Create a Deployment Environment

Your continuous deployment pipeline will need a target environment containing virtual servers, or Amazon EC2 instances, where it will deploy sample code. You will prepare this environment before creating the pipeline.

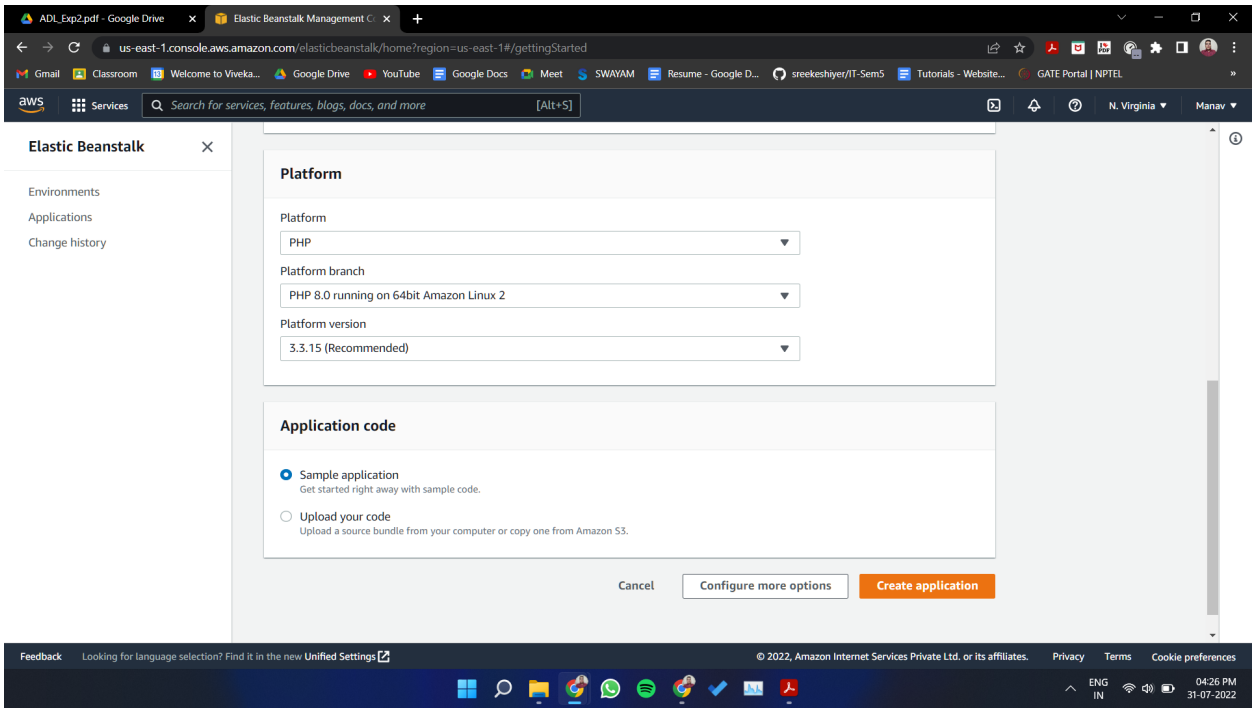
1. Login to your AWS account and search for Elastic Beanstalk in the search box.



2. Open up Elastic Beanstalk and name your web app.

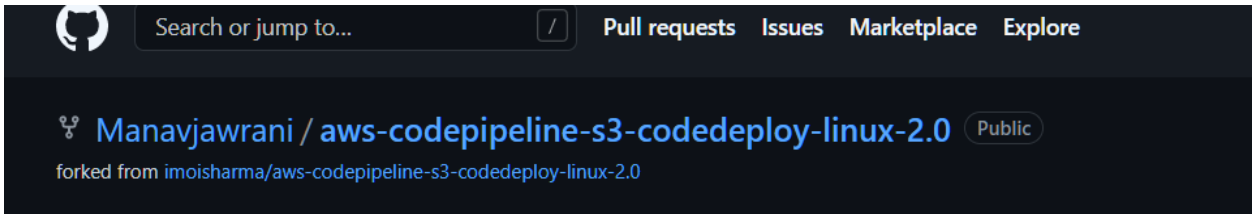


3. Choose PHP from the drop-down menu and then click Create Application.



4. Beanstalk creates a sample environment for you to deploy your application. By default, it creates an EC2 instance, a security group, an Auto Scaling group, an Amazon S3 Bucket, Amazon CloudWatch alarms and a domain name for your Application.

Step 2: Get a copy of your sample code



In this step, we will get the sample code from this GitHub Repository to later host it. The pipeline takes code from the source and then performs actions on it.

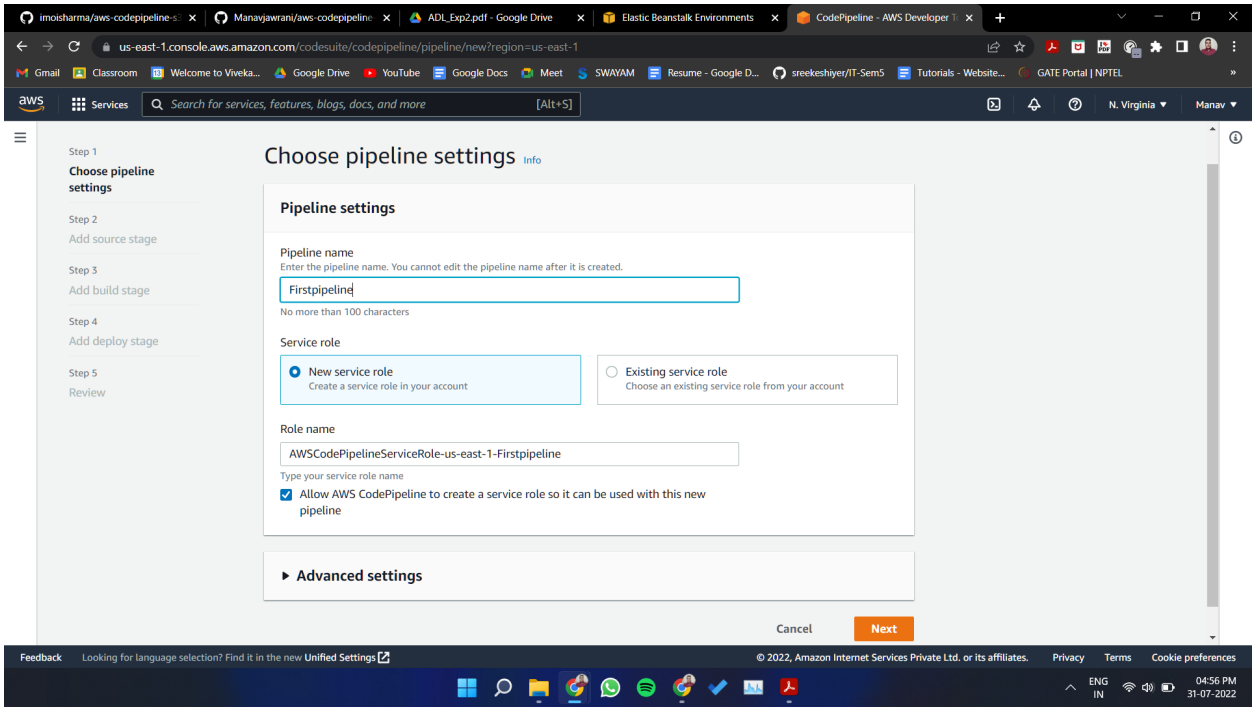
For this experiment, as a source, we will use this forked GitHub repository. We can alternatively also use Amazon S3 and AWS CodeCommit.

Go to the repository shared above and simply fork it.

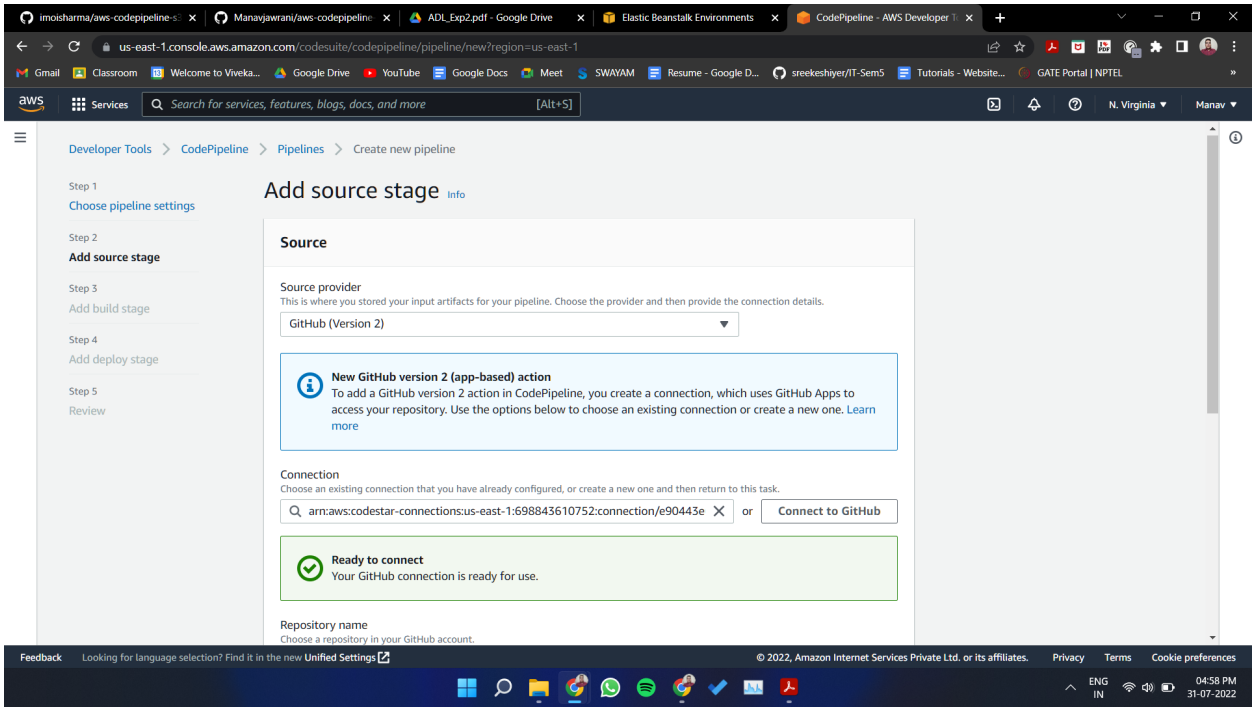
Step 3: Creating a CodePipeline

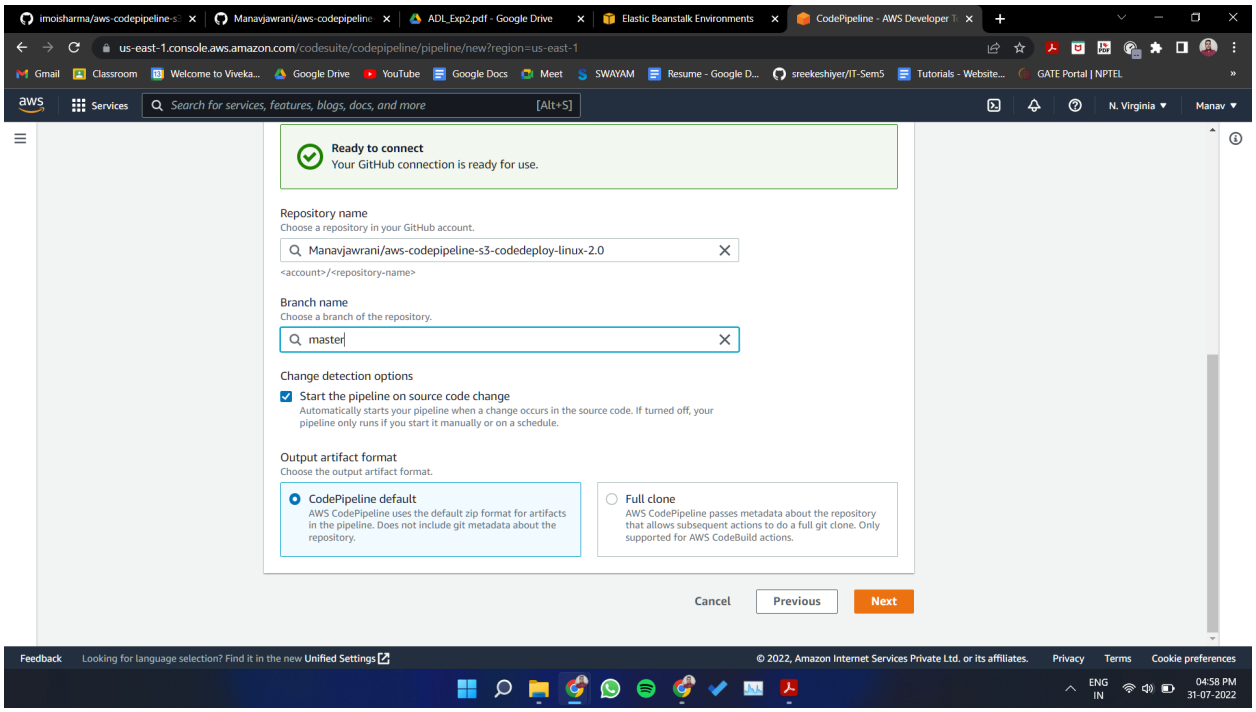
In this step, we'll create a simple pipeline that has its source and deployment information. In this case, however, we will skip the build stage where you get to plug in our preferred build provider.

1. Go to AWS Developer Tools -> CodePipeline and create a new Pipeline. Fill in the initial settings first.



2. In the source stage, choose GitHub v2 as the provider, then connect your GitHub account to AWS by creating a connection. You'd need your GitHub credentials and then you'd need to authorize and install AWS on the forked GitHub Repository.

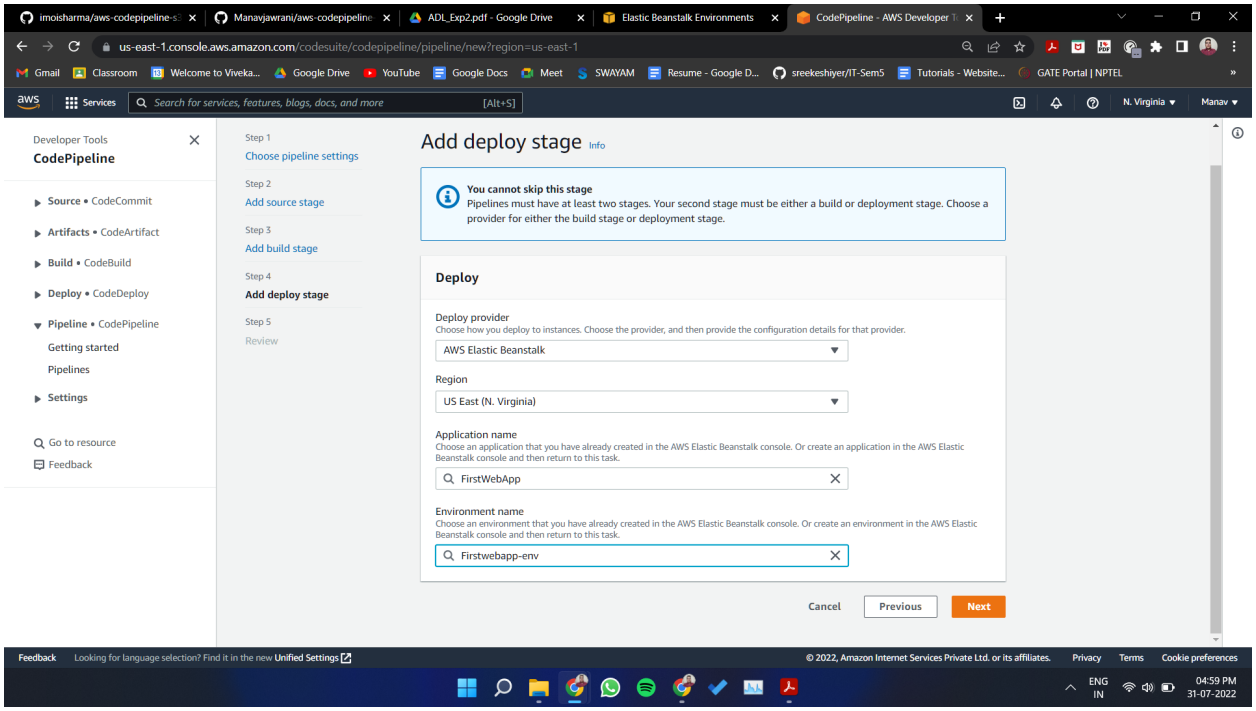




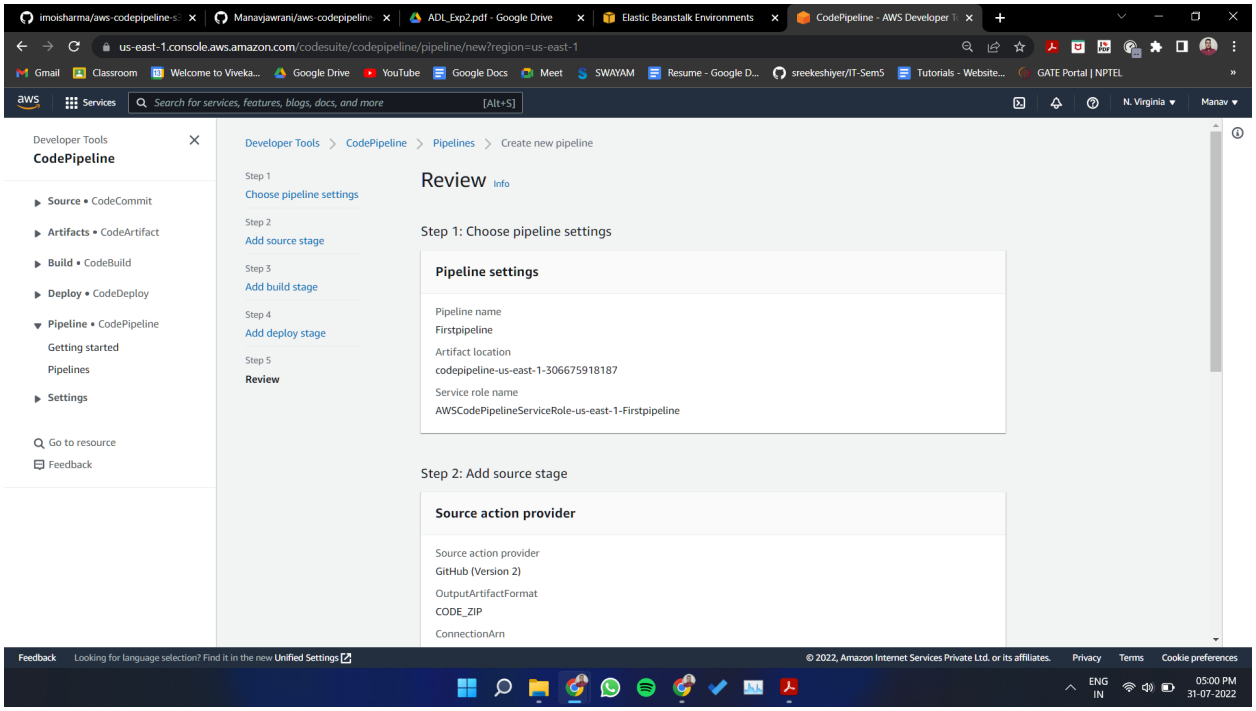
3. Then, simply choose this forked repository and the branch which you will be able to find in the search box. After that, click Continue and skip the build stage. Proceed to the Deployment stage.

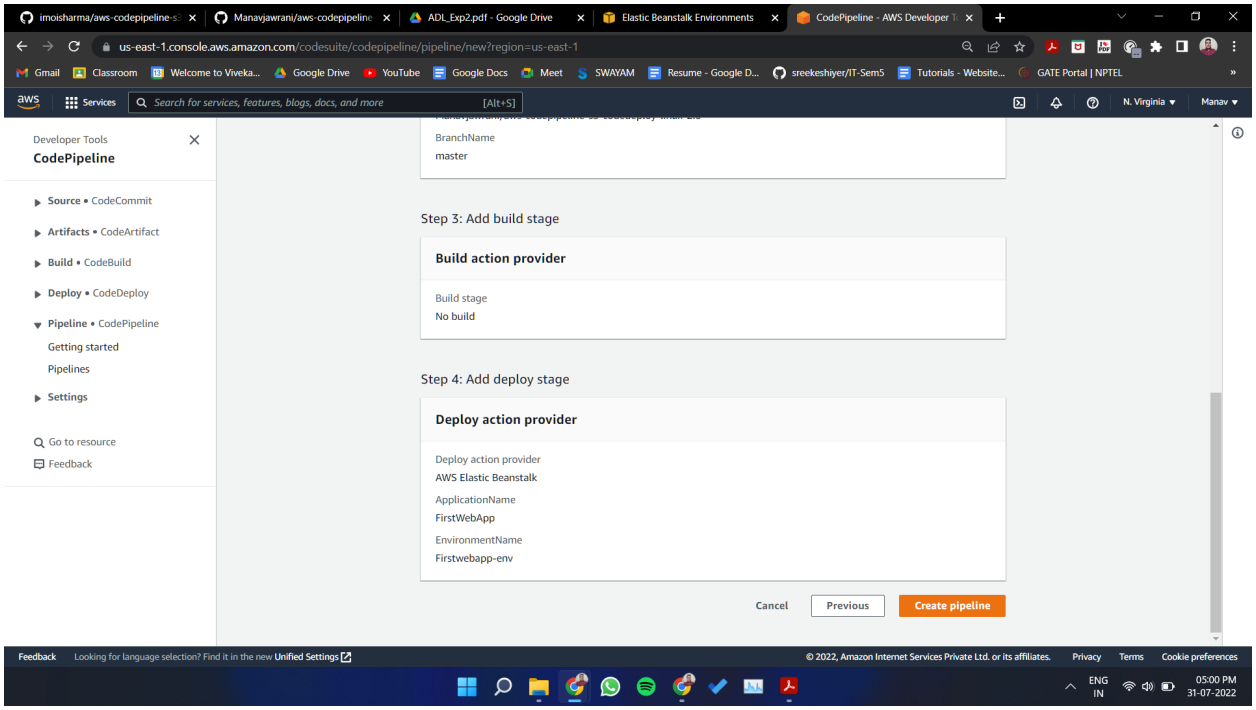
Step 4: Deployment

1. Choose Beanstalk as the Deploy Provider, same region as the Bucket and Beanstalk, name and environment name. Click Next, Review and create the pipeline.

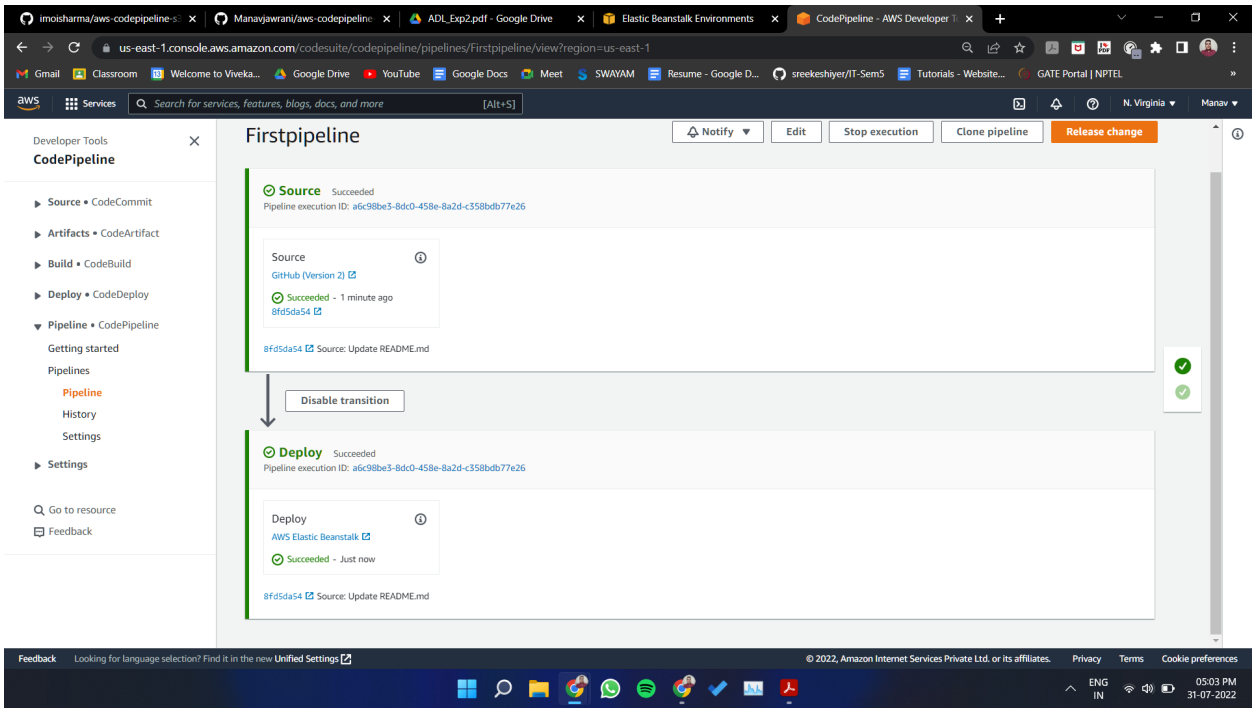


2. Review all the settings and click on create pipeline

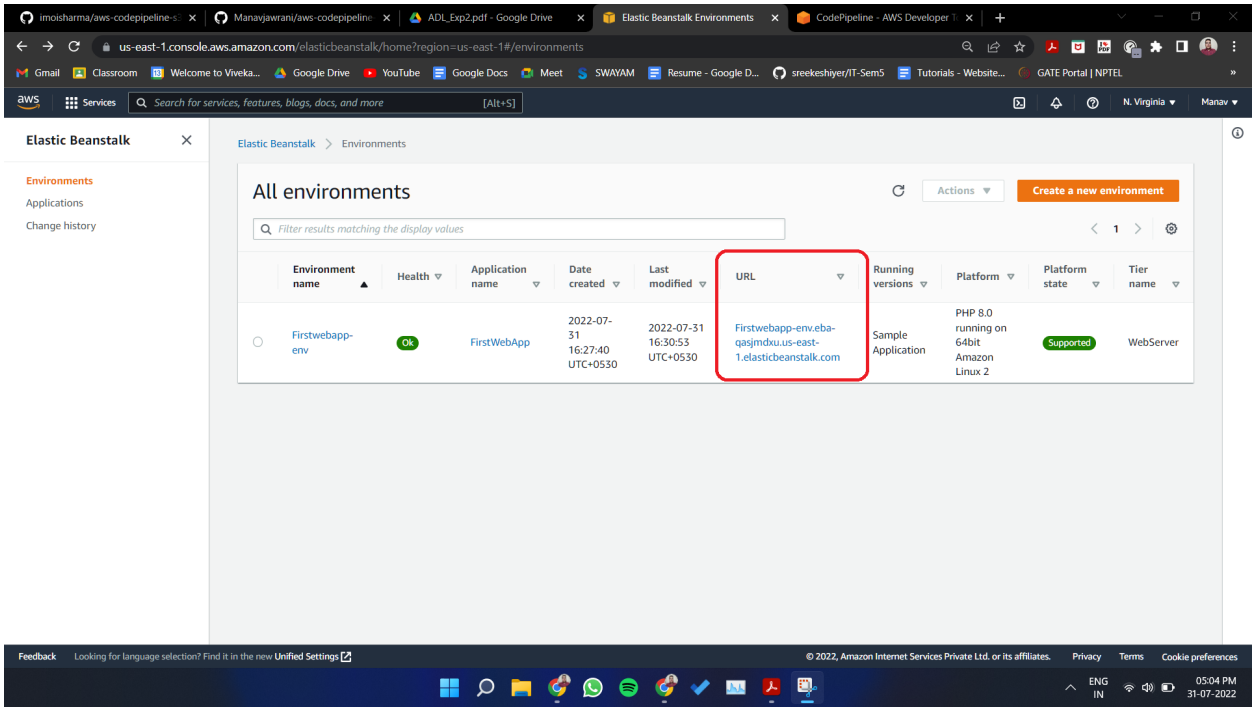




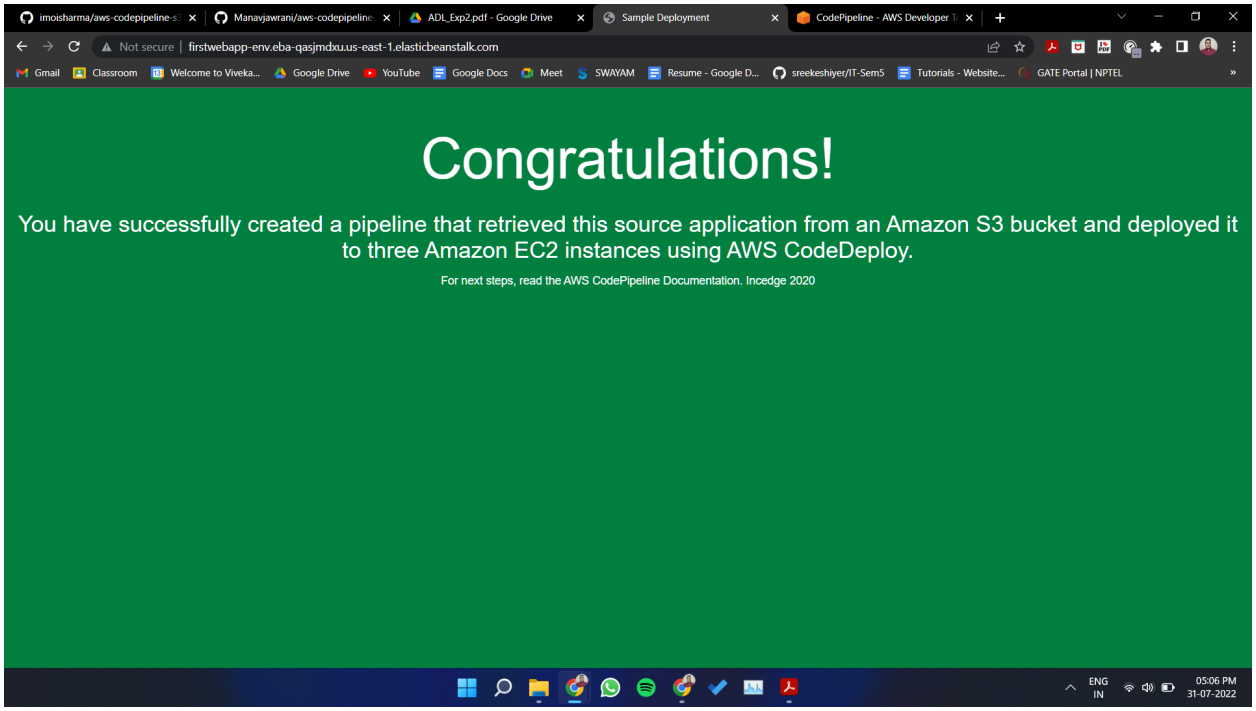
3. If you are able to see the this type of screen it means it is deployed successfully



4. In a few minutes, we will have our pipeline created. Once we have the success message on the Deploy part, we can go ahead and check our URL provided in the EBS environment.



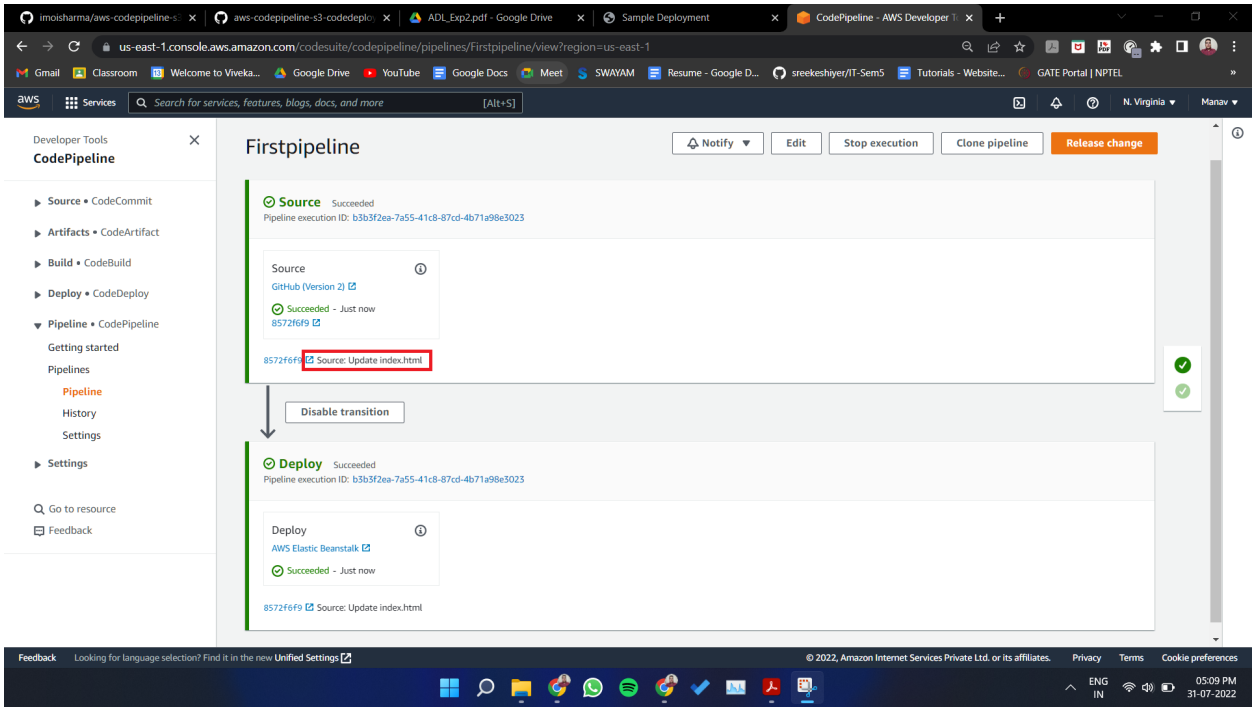
This is the sample website we just created.



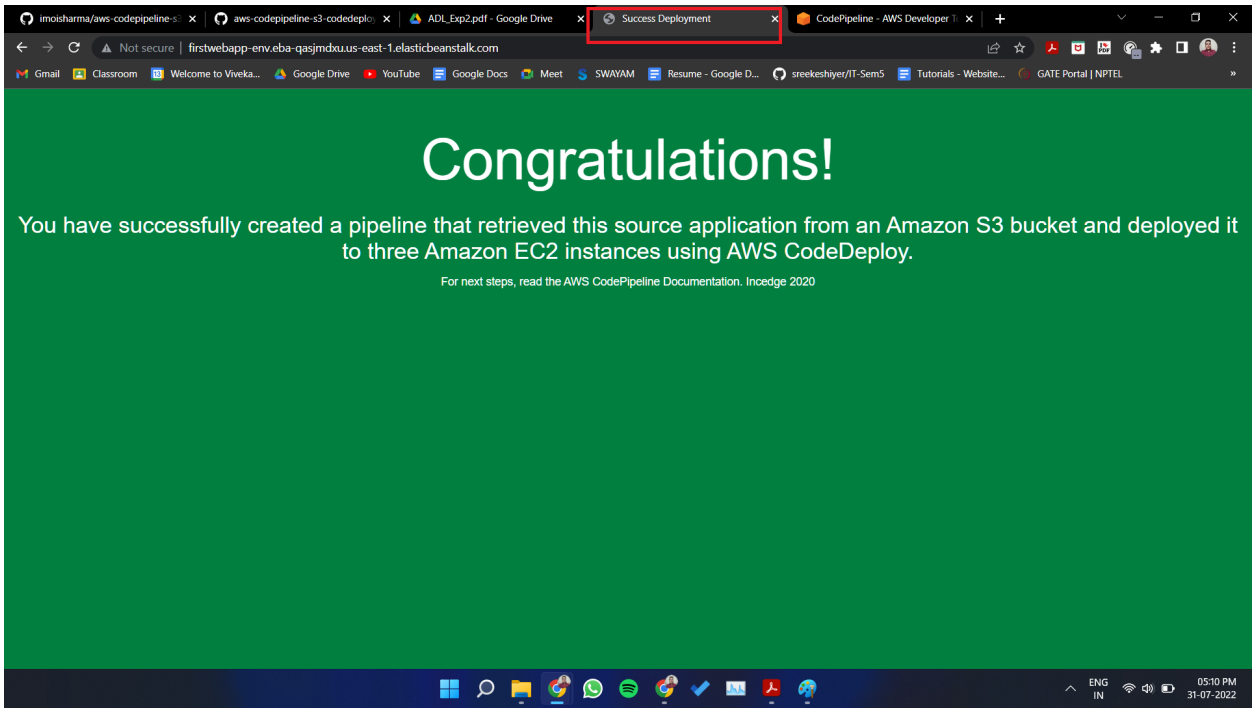
If you can see this, that means that you successfully created an automated software using CodePipeline.

Step 5: Committing changes to update app

1. In this step, we will update the code which we had and make a few changes to the HTML file (keep in mind, this is in our version of the forked repository).
2. In GitHub, open index.html. Then, make changes to either the heading tag or the paragraph tag. Commit these changes on the fly on GitHub.
3. When you commit these changes to your forked version, you'll notice the changes being made in real-time on the Source panel.
4. You can view the changes on the website using the same URL, once the deployment section shows success.



5. Check the changes in the website , here we change the title of the html file.



Conclusion :

In this experiment we learned how to use AWS Elastic Beanstalk Environments to deploy our websites and create a CodePipeline unde the AWS Development Console, source data to the Beanstalk using GitHub and finally, make real-time changes to the website just by pushing updates to GitHub.