|  |  |  |
| --- | --- | --- |
| CONTINUOUS DELIVERY WITH JENKINS AND AWS SERVICES | | |
| Team Number | 53 | |
| Team Names | Bhavani Devi Manyala | Madhulatha Pitta |
| V.S.N.Lakshmi | |
| Mentor Name | Vijay | |
| Team SPOC Number | 9553472784 | |
| Field: DevOps, cloud | | |

INTRODUCTION:

In past, the model used for software development was the “WATERFALL” Model. This model is best suited when all the requirements are present signifies the planning stage. All the requirements should be finalized at that time only. No later changes can be entertained and no work is in parallel for software development in this model. To overcome this, DevOps was comes into the existence.

DevOps is a set of practices that combines software development and IT operations. It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with agile software development; several DevOps aspects came from the agile methodology.

The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. Continuous collaboration is vital, both with team members and project stakeholders.

DevOps helps the developers to automate the entire process i.e., from build process to delivery process. CI/CD is a practice in DevOps that allows developers too quickly and automatically test, package, and deploy their applications. Most commonly, it is often achieved by leveraging a server called Jenkins, which serves as the most common CI/CD orchestrator. Jenkins listens to specific inputs given by the developer and, when triggered, kicks off a pipeline flow. Continuous Delivery increasesthe Business Agility because you are able to deploy anytime with lowrisk. Moreover, bugs and other issues can easily be identified and corrected because feedback for changes is immediate.

SCOPE OF THE PROJECT:

The project called “CONTINUOUS DELIVERY WITH JENKINS AND AWS SERVICE” is to automate the process entire process by using open source tool called Jenkins for creating pipeline and with the help of AWS services.

The goal of the continuous integration and continuous delivery (CI/CD) pipeline is to enable teams to release a constant flow of software updates into production to quicken release cycles, lower costs, and reduce the risks associated with development.

In this project, the entire process of Continuous delivery is done with the help of EC2 instances in AWS service and creating a pipeline using Jenkins. A continuous delivery pipeline is an automated expression of the process for getting software from version control right through to users and customers. **Continuous Delivery** (CD) is a DevOps methodology that enables development teams to deploy changes such as new features, configuration, bug fixes, and experiments into production safely, quickly, and sustainably.

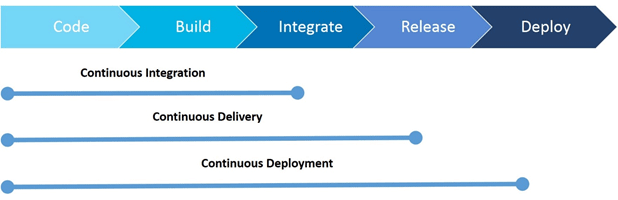


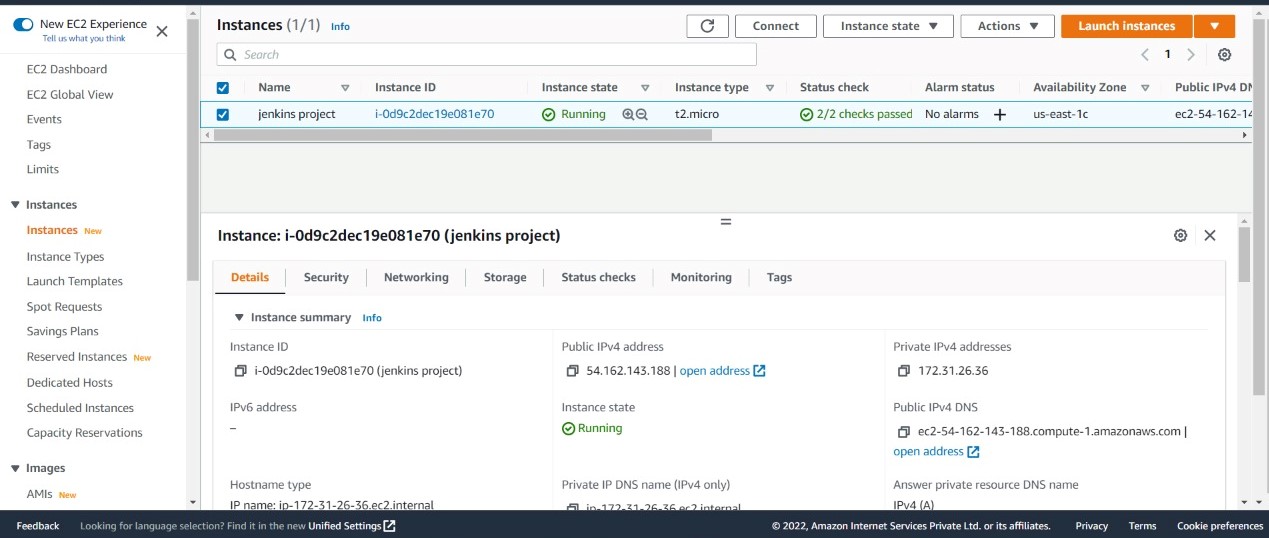
Fig: CONTINUOUS DELIVERY PIPELINE

Continuous Delivery is an extension of Continuous integration. Continuous delivery is a software development practice that uses automation to speed the release of new code. It establishes a process through which a developer's changes to an application can be pushed to a code repository through automation. With continuous delivery, every code change is built, tested, and then pushed to a non-production testing or staging environment. Simply say that continuous delivery is the presence of a manual approval to update to production. The final decision to deploy to a live production environment is triggered by the developer*.*

**PROCESS OF THE PROJECT:**

Let us discuss the entire process:

Step1: Launching an EC2 instance in AWS with the configuration of Linux operating system.



Step 2: After creating an EC2 instance, connect the instance and install java and Jenkins in that Linux machine by using the below commands.

* To download and install Jenkins, first go to root account by using the command

[ec2-user ~]$ Sudo su

* To ensure that the software packages are up to date on the instance, use the following command to perform a quick software update:

[ec2-user ~]# yum update -y

* Add the Jenkins repo using the following command:

[ec2-user ~]# wget -O /etc/yum.repos.d/jenkins.repo \

https://pkg.jenkins.io/redhat-stable/jenkins.repo

* Import a key file from Jenkins-CI to enable installation from the package:

[ec2-user ~]# rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

[ec2-user ~]# yum upgrade

* Install java:

[ec2-user ~]# amazon-linux-extras install java-openjdk11 -y

* Install Jenkins:

[ec2-user ~]# yum install Jenkins -y

* Enable the Jenkins to start a boot:

[ec2-user ~]# systemctl enable jenkins

* Start jenkins as a service

[ec2-user ~]# systemctl start jenkins

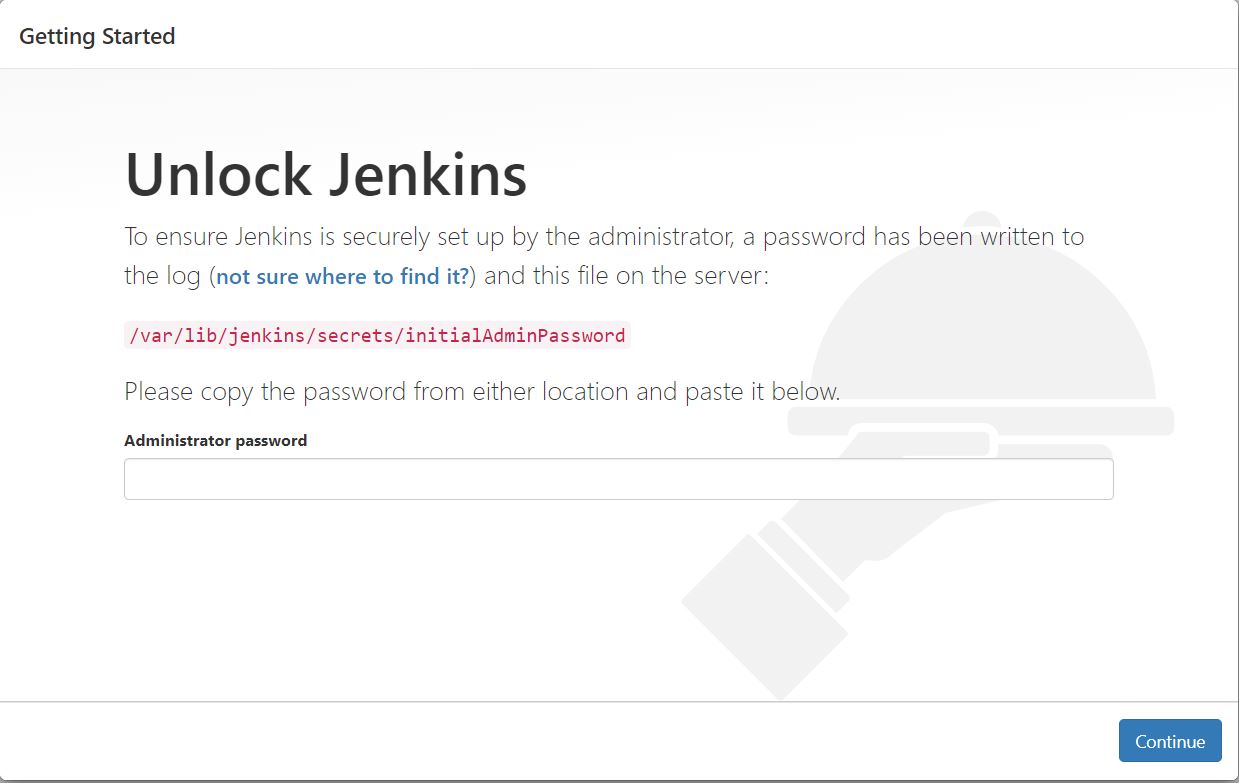
* checking the status of the Jenkins service using the command:

[ec2-user ~]# systemctl status jenkins

Step 3: configuring Jenkins

Now Jenkins is installed and running on EC2 instances.

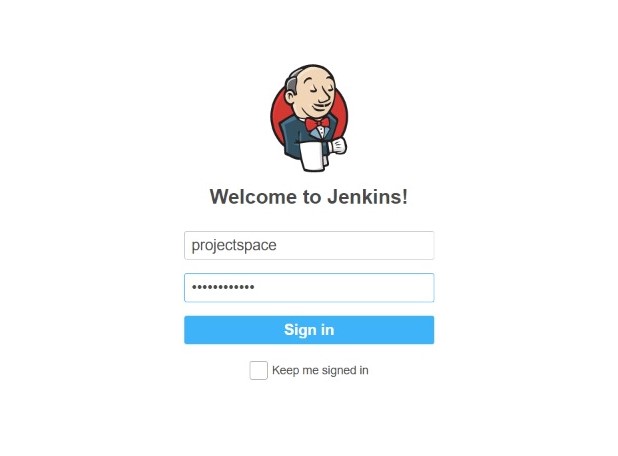
* Connect to http://<your\_server\_public\_DNS>:8080 from the browser. It will be able to access Jenkins through its management interface:



* Command display the Administrator password

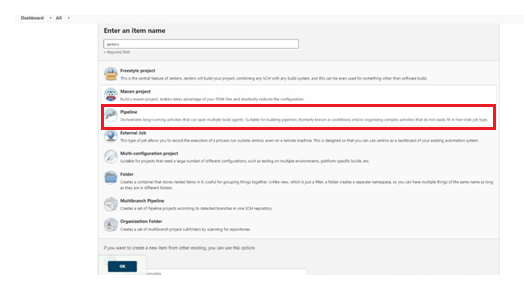
[ec2-user ~]# cat /var/lib/jenkins/secrets/initialAdminPassword

* The Jenkins installation script directs you to the **Customize Jenkins page**. Click **Install suggested plugins**.
* Once the installation is complete, **Create First Admin User** then login into jenkins account.



Step 4: create a new item in Jenkins.

* After log in into our Jenkins account, create a new item ->enter the item name ->select pipeline ->click on ok. It will redirect to the configuration page.



**INSTALLING PLUGINS**:

We have to install some plugins for supporting the process of pipeline. The plugins used in this project is mentioned below.

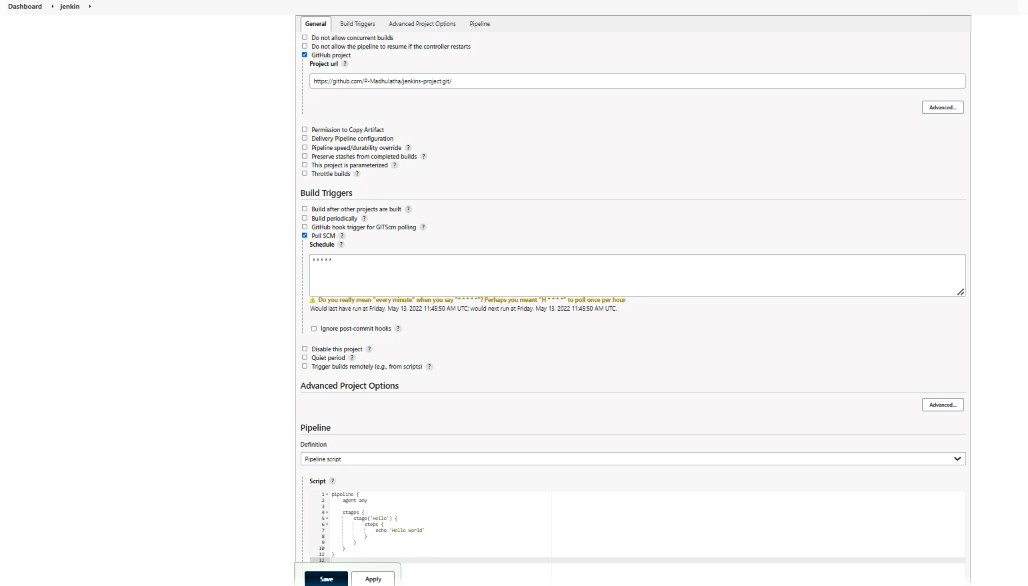
* Git plugin:

In order to integrate Jenkins with Github, all you require is a plugin. The Github plugin for Jenkins allows you to schedule your build and facilitates easy transfer of data from the Github repository to Jenkins machine. Moreover, it also triggers each build automatically after each commit. In that we have to install Git bash. Git bash allow developers to use Git in a command line interface.

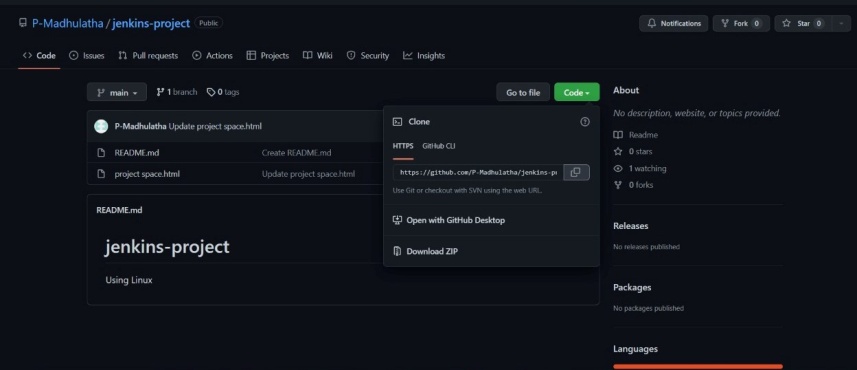
* Build pipeline plugin:

This plugin provides a Build Pipeline View of upstream and downstream connected jobs that typically form a build pipeline.  In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins.

**CONFIGURATION**:



* Go to “**general”** and enable the github project option ad then give the github repository link.

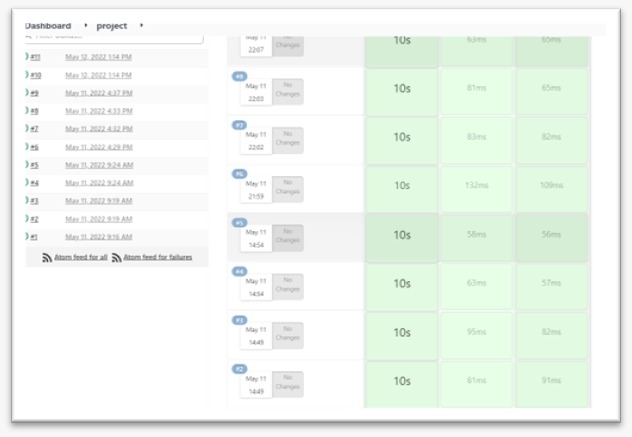


* In build triggers, select poll SCM. Typically, SCM attempts to centrally control or link the production and distribution of a product. By managing the supply chain, companies can cut excess costs and deliver products to the consumer faster.
* Now go to pipeline and select pipeline script.
* Give the desired code in script section. Click on apply and save.

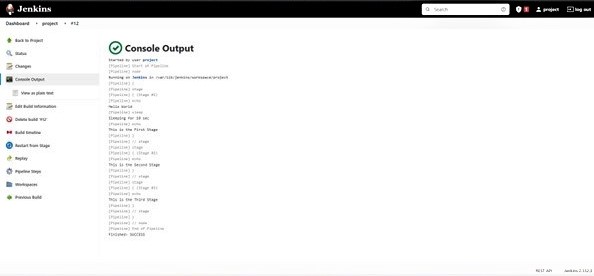
Step 5: Execution

* Go to dashboard, select the project which we previously created. Click on “**build now”** then it will start its execution. It will produce an entire pipeline process.

**IMPACT AND OUTCOME**:



The entire pipeline is created and the process gets automated such as connection among build, test, deploy and release to the production. It repeats the pipeline automatically based on poll SCM configuration.



If we want to check the output in a statement manner, go to console output, there we can observe the scripted output what we gave previously in configuration panel.

Continuous delivery provides an ideal scenario for the organization’s application teams. So developers simply push code to a repository. This code will be integrated, tested, deployed, tested again, merged with infrastructure, go through security and quality reviews, and be ready to deploy with extremely high confidence.

Now, the industry needs to streamline testing by reducing development tool complexity. Tools are generating ample data, but the next evolution is efficiently taking that data and using it to drive DevOps to even faster CI/CD cycles.

When CI/CD is used, code quality is improved and software updates are delivered quickly and with high confidence that there will be no breaking changes. The impact of any release can be correlated with data from production and operations. It can be used for planning the next cycle, too—a vital DevOps practice in the organization’s cloud transformation.

As CI/CD evolves, it has further broken down the wall between development and operations, empowering developers to spend the bulk of their time enhancing business applications. Typically, the gains have come by automating various parts of the build, test, and deploy processes.