# **Setting up a CI/CD Pipeline Process with Jenkins and Docker in AWS**



CI/CD is an essential part of DevOps and any modern software development practice. A purpose-built CI/CD platform can maximize development time by improving an organization’s productivity, increasing efficiency, and streamlining workflows through built-in automation, testing, and collaboration.

**What is Continuous Integration**

Continuous integration is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Here we will utilize Jenkins as our Continuous Integration tool to automatically build and test our application every time code changes are made. This will help identify issues early in the development process, allowing for faster feedback and remediation.

**What is Continuous Delivery/Deployment**

Continuous delivery is a software development practice where code changes are automatically built, tested, and prepared for a release to production. Here again, we will use Jenkins to automatically deploy our application to a server or Docker registry when the build and test phases are successful, ensuring that the latest version of our application is always available to end-users. This ensures Continuous Delivery/Deployment and reduces the time-to-market for our application.

**What is Continuous Testing**

Continuous testing is a core DevOps practice that provides a continual assessment of incremental changes throughout the development process. Our application will be continuously tested while the Docker image is run as part of the pipeline process, helping to identify any issues early in the development cycle.

**Infrastructure as Code**

Infrastructure as code is a practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration. We will leverage Jenkins pipeline scripting to automate the build, test, and deployment process of our application. This pipeline script can be version-controlled and treated as code, making it easier to manage and reproduce the pipeline.

**Webhook Triggers:**

We will set up webhook triggers to automatically trigger pipeline builds whenever changes are pushed to the Git repository. This ensures that our pipeline is always up-to-date with the latest changes, and our application is continuously built and tested.

A Webhook, also known as a Web Callback or HTTP push API is a mechanism for an application to provide real-time data to other apps. A Webhook sends data to other applications in real-time,

**Docker**:

We will containerize our application using Docker to make it more portable and scalable. Docker allows us to package our application and its dependencies into a single, portable unit that can be run consistently on any platform.

**Git:**

We will use Git to manage the source code of our application, making it easier to collaborate and version-control changes.

**Introduction**

**\*\*Set up a CI/CD pipeline using Jenkins and Docker by building a simple Flask application, testing it, and deploying it to Docker Hub.**

* Containerize a simple Flask application using Docker to make it more portable and scalable.
* Use Git to manage the source code of the application to make it easier to collaborate and version-control changes.
* Implement Infrastructure as Code for automated build, test, and deployment process using Jenkins pipeline script.
* Ensure Continuous Integration of the application by configuring Jenkins to automatically build and test every time code changes are made.
* Implement Continuous Delivery/Deployment by configuring Jenkins to automatically deploy the application to a Docker registry when the build and test phases are successful.

**Steps:**

1.Create an EC2 instance.

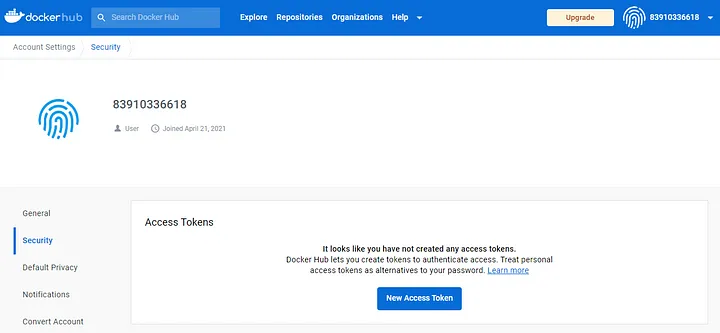
2.Install Jenkins

3.Install Docker

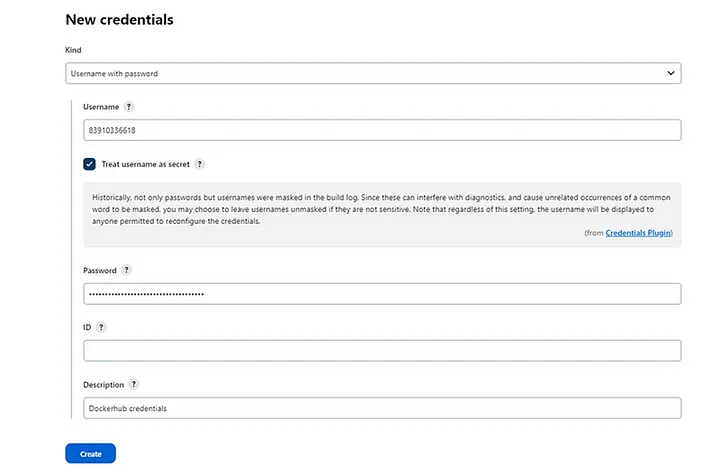
4.Install GIT

5.Configure docker hub credentials

Application link - https://github.com/yasminjeelani/newcicd.git



7. In your Jenkins server go to Manage Jenkins -> Credentials -> System -> Global credentials -> + Add credentials. Enter your Docker Hub username, the token you created and a brief description of the credentials.



8. Create the environment variables used in the Jenkinsfile for the Docker image and for the Docker Hub credentials.

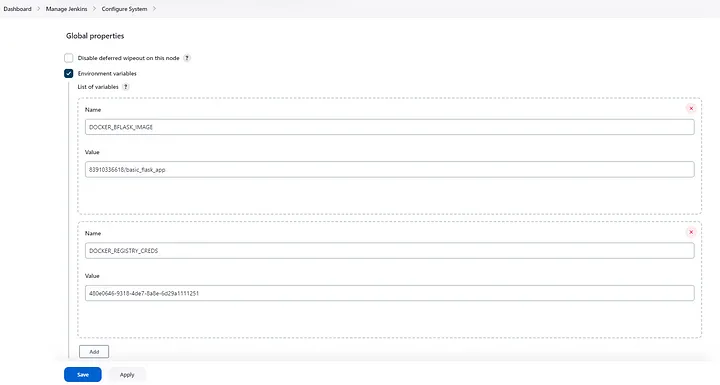
Go to Manage Jenkins -> Configure System and under ‘Global properties’ add them as shown in the picture below.

Name: DOCKER\_BFLASK\_IMAGE

Value: [your-dockerhub-username/repository-name]

Name: DOCKER\_REGISTRY\_CREDS

Value: ID generated when adding the Docker Hub credentials into Jenkins.



Click on Apply and Save to Continue.

9. Create a new Jenkins job.

In your Dashboard go to New Item -> Pipeline and enter the name of your project.

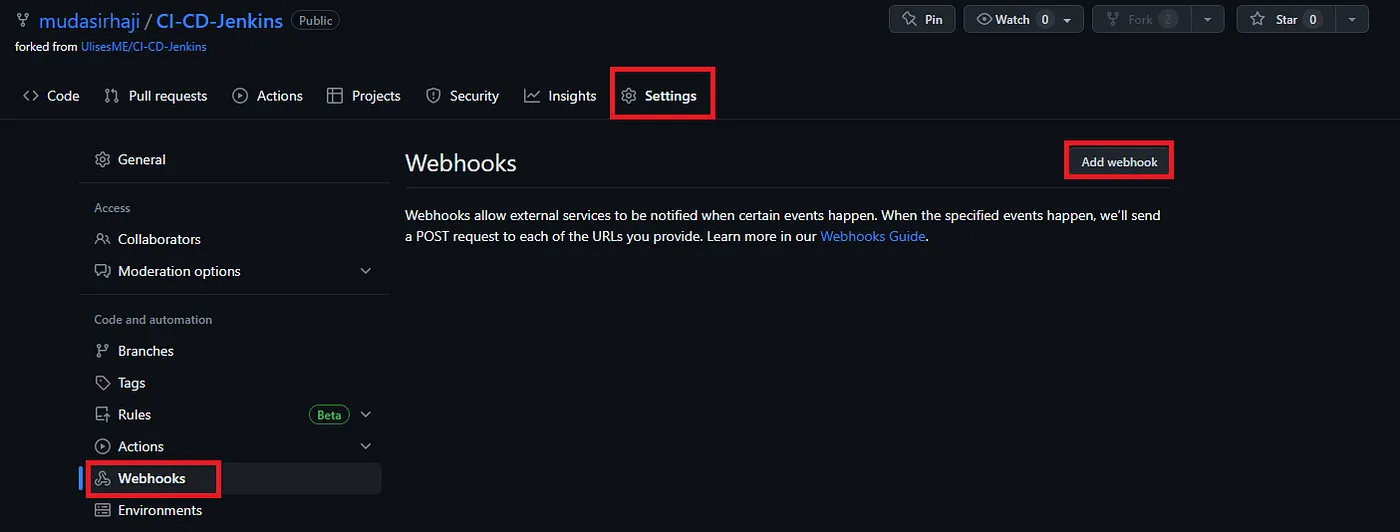
10. Select ‘Github project’ and add your Github repository.

12. Create pipeline and build it

13. Adding Webhook to our Jenkins Pipeline

A Webhook, also known as a Web Callback or HTTP push API is a mechanism for an application to provide real-time data to other apps. A Webhook sends data to other applications in real-time, so you get it right away.

To add a webhook in our GitHub repo go to the settings, click on webhooks, and then on Add webhook as shown below:

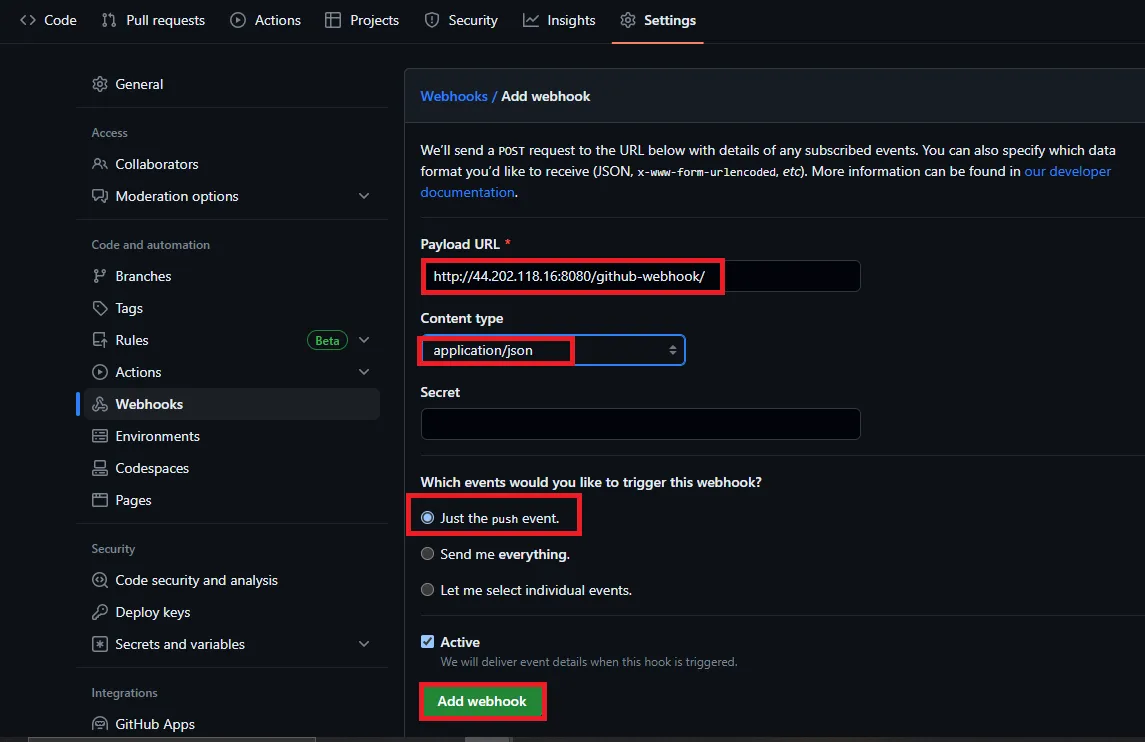


On the next screen provide the following info:

Payload URL: [EC2-IP-Address:8080/github-webhook/]

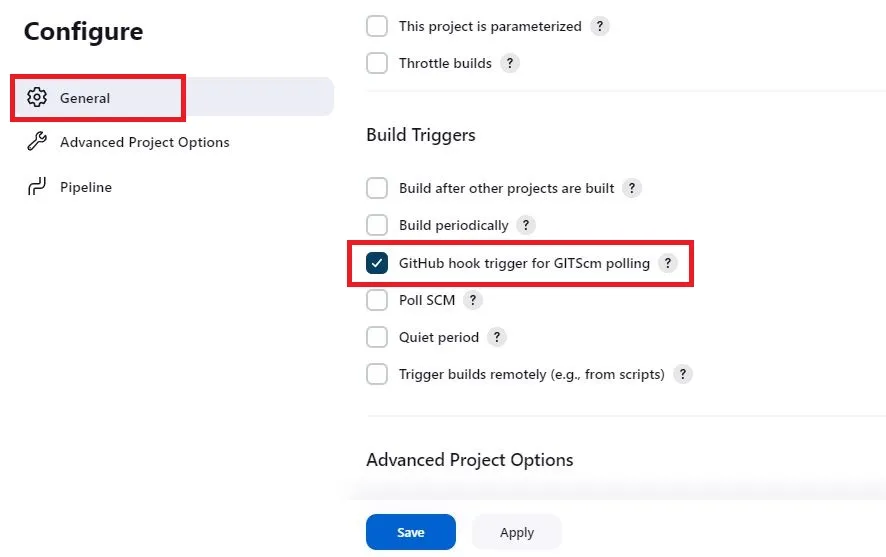
Content type: application/json

Trigger only on the push event



Click on Add Webhook to proceed.

Now under Build Triggers select the option GitHub hook trigger for GITScm polling, click on Apply and Save to continue.



When Jenkins receives a GitHub push hook, GitHub Plugin checks to see whether the hook came from a GitHub repository which matches the Git repository defined in SCM/Git section of this job. If they match and this option is enabled, GitHub Plugin triggers a one-time polling on GITScm.

Now After that, we will install the GitHub Integration plugin in our Jenkins Server

Jenkins -> Manage Jenkins -> Manage Plugins -> Install github integration plugin

Let’s do some minor changes in our GitHub repo, commit and push your changes. which should trigger the Jenkins Job, thus displaying the successful integration of Github webhooks with Jenkins that ensured that the latest changes are tested, built, and deployed as quickly as possible.

