**CI/CD (Continuous Integration and Continuous Deployment/Delivery):**

It is a backbone of DevOps

**Scenario:** A company wants to ensure that any code changes made by developers are automatically tested and deployed.

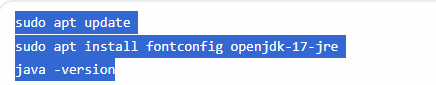
**CI: -** Once Developer push their code that code should be tested then security Check then application build and it should be ready to deploy called Continuous Integration.

**CD (Continuous Deployment): -** Once CI is completed, automatically triggered called continuous Deployment.

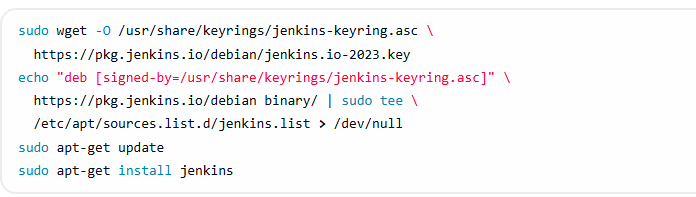
**CD (Continuous Delivery): -** Once CI is completed, we need manually triggered called continuous Deployment.

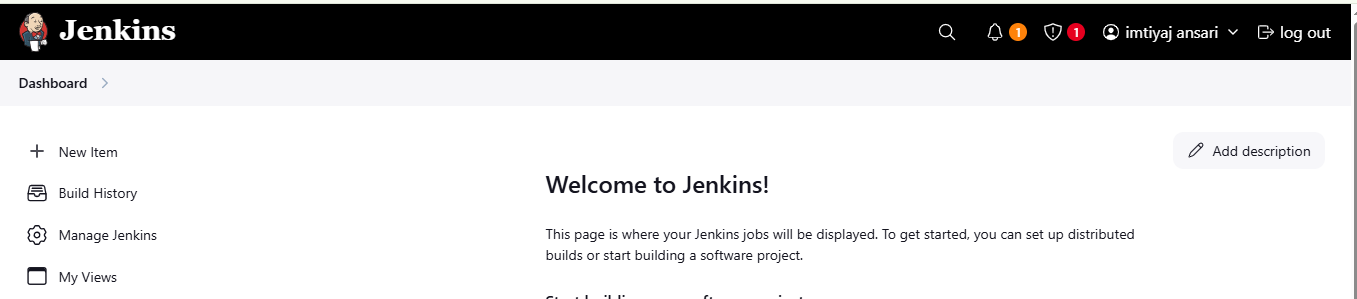
**Installation of Jenkins:-**

**First Install Java…….**



**Now Install Jenkins……**





**Discard old builds: -** Automatically removes older build records to save storage space.

**Do not allow concurrent builds: -** Prevent multiple builds from running simultaneously to avoid conflicts and ensure consistency.

**Do not allow the pipeline to resume if the controller restarts: -** Ensures a pipeline won't continue after a Jenkins controller reboot, forcing a fresh start.

**GitHub project: -** Specifies the GitHub repository from which the pipeline retrieves its source code.

**Pipeline speed/durability override: -** Adjust pipeline execution settings to prioritize either faster runs or higher durability.

**Preserve stashes from completed builds: -** Keeps stashed data (temporary files) even after a build finishes.

**This project is parameterized: -** Enables the pipeline to accept user-defined input values during execution.**Throttle builds: -** Limits the rate at which builds are executed to manage system load.

**Triggers:-**

**Build after other projects are built: -** Triggers a build when another specified Jenkins project completes successfully.

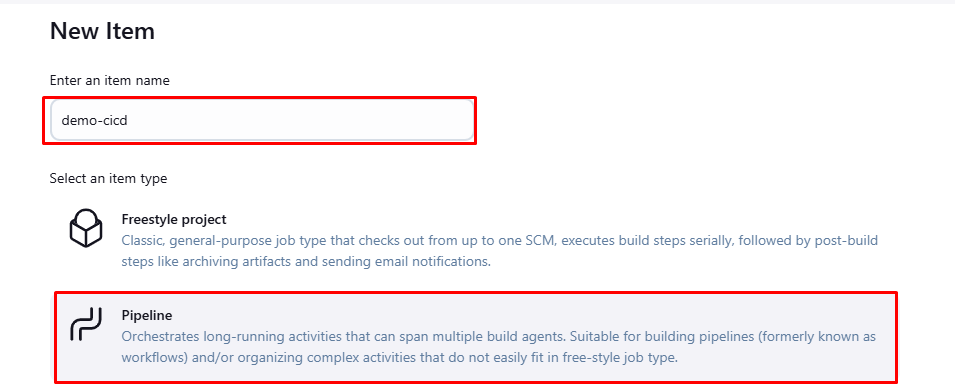
**Build periodically: -** Schedules builds to run at regular intervals, defined by a cron expression.

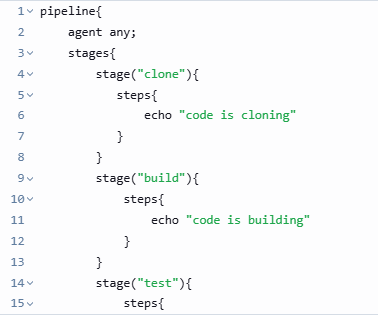
**GitHub hook trigger for GITScm polling: -** Automatically triggers a build when GitHub detects a push to the repository.

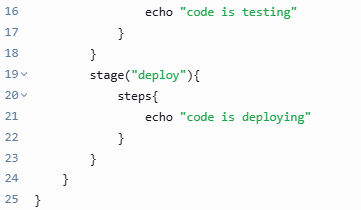
**Poll SCM: -** Regularly checks the source code management (SCM) system for changes and triggers a build if any are found.

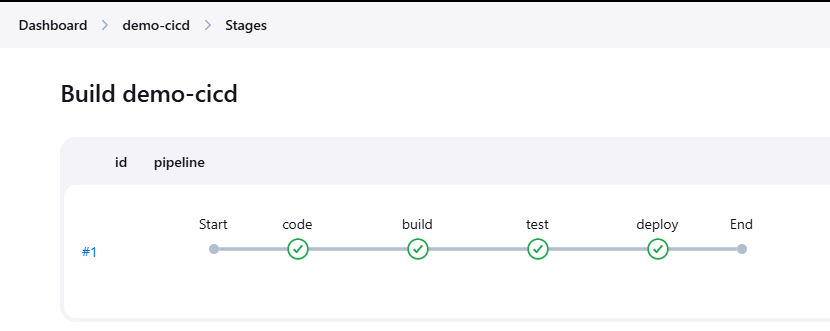
**Trigger builds remotely (e.g., from scripts): -** Allows builds to be initiated via external scripts or tools using a specific URL and authentication.

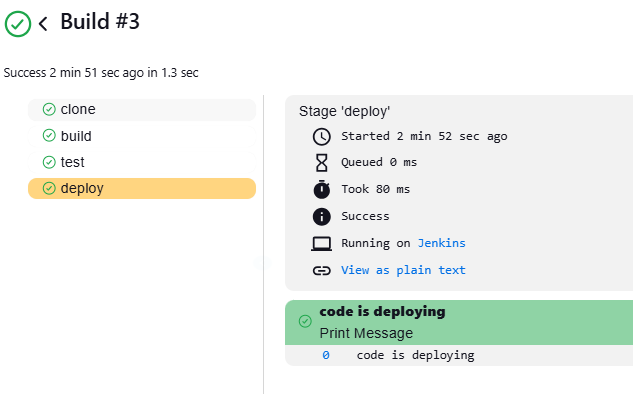
**Create Demo pipeline…**





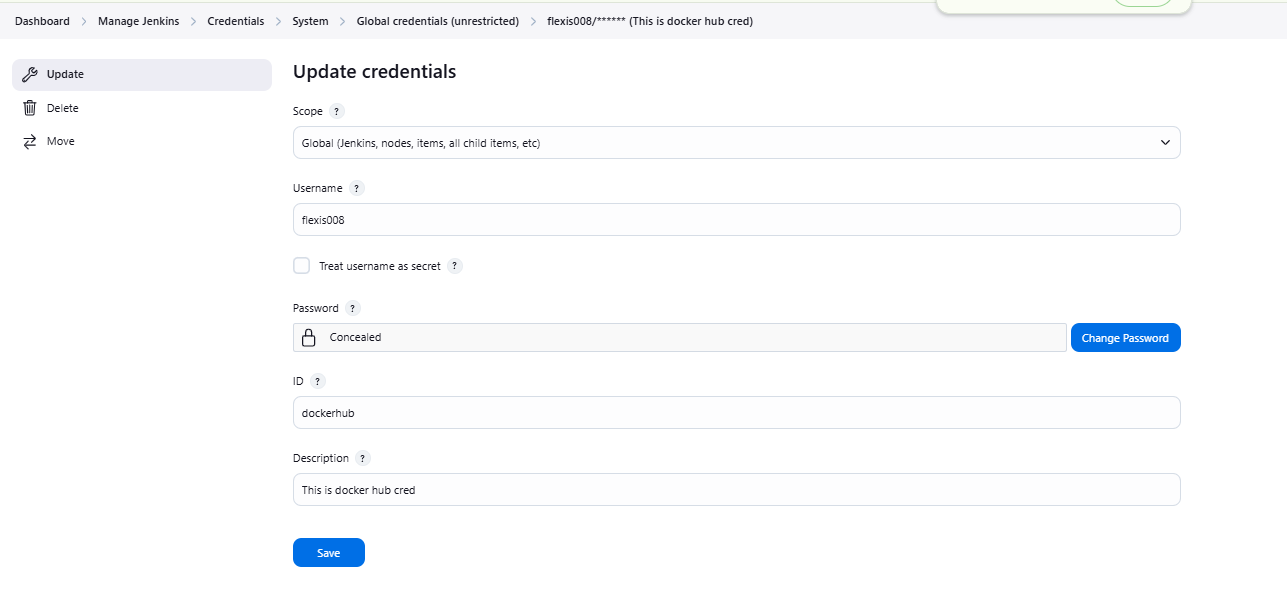






**Create two-tier-flask-app pipeline…**

First set the id and password in Jenkins credential



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Create two-tier-flask-app pipeline automatically…**

Need to set up webhook.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**In Jenkins pipeline select**

A screenshot of a computer

AI-generated content may be incorrect.

**Create two-tier-flask-app pipeline automatically in Jenkinsfile…**

Create a file named Jenkinsfile in git hub and write a pipeline in Jenkinsfile

A screenshot of a computer

AI-generated content may be incorrect.

**Go to Jenkins pipeline and select Pipeline script from SCM**

A screenshot of a computer

AI-generated content may be incorrect.

**Master Slave architecture**

Create a slave machine, copy public key from master to slave (authorized foiled) and install java on slave machine.

On Dashboard, set up agent

A close up of a screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Name- Slave (anything)

Remote root Directory- create a folder in slave and provide the path

Labels- dev or test (this label pass in Jenkins file)

A screenshot of a computer

AI-generated content may be incorrect.

Launch Method- via ssh

Host- slave machine IP

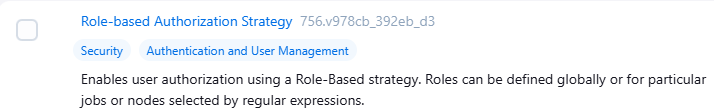
Credential- fill the details and put master private key

A screenshot of a chat

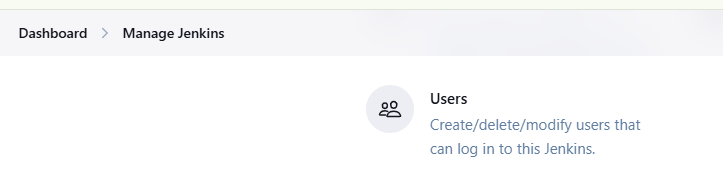
AI-generated content may be incorrect.

**Role Based Access: -**

**Install Role-Based Authorization Strategy**

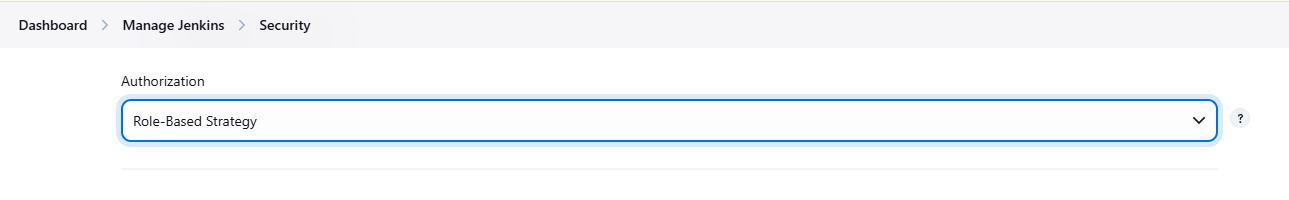


**Create User**

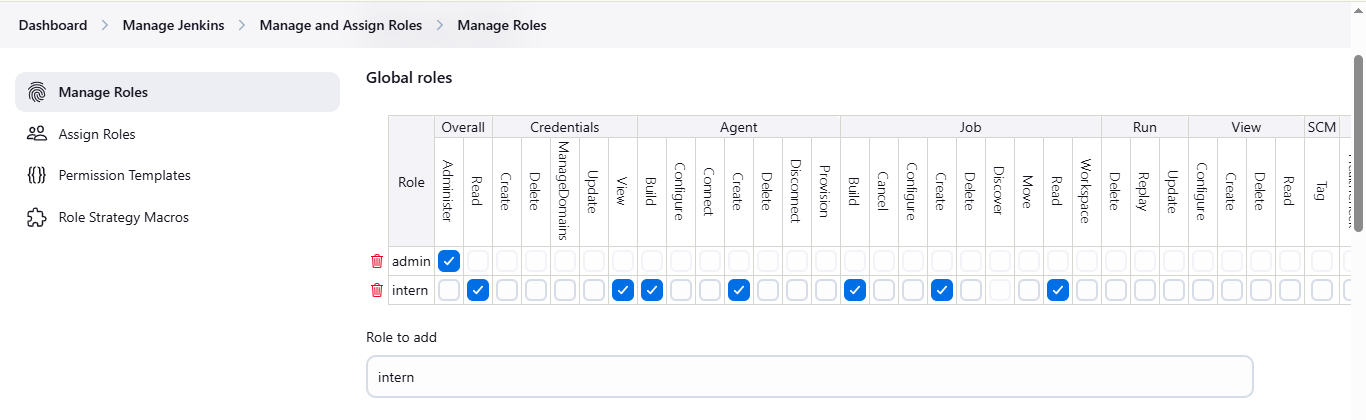




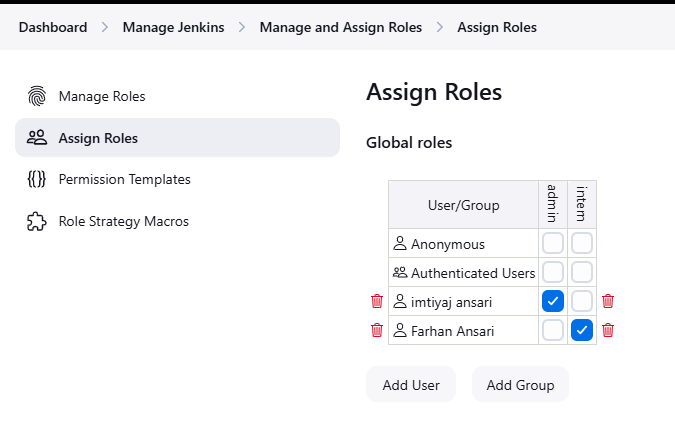
**In Security set Role-Based Strategy**



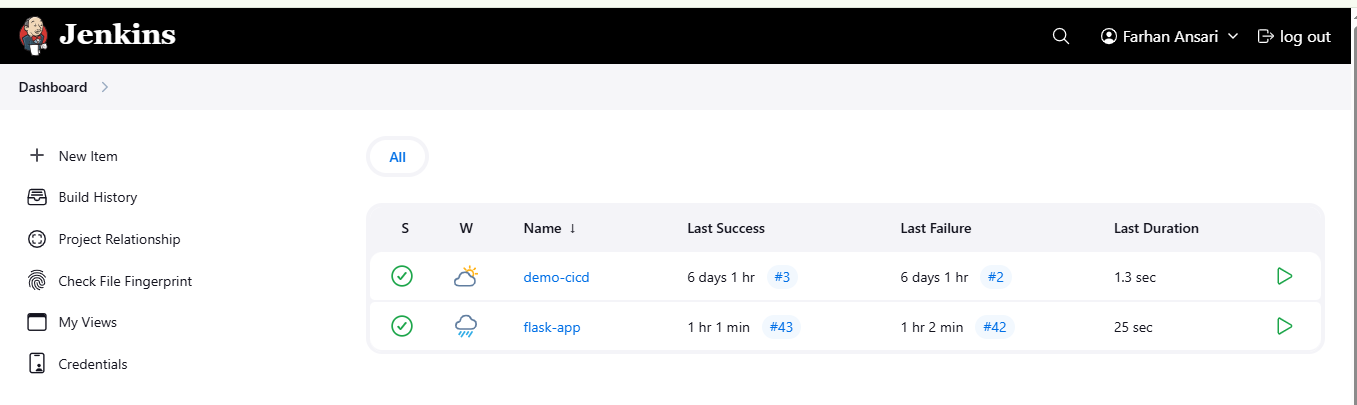
**Create a Manage Role line Intern and give permission**



**Assign Intern role to created user**

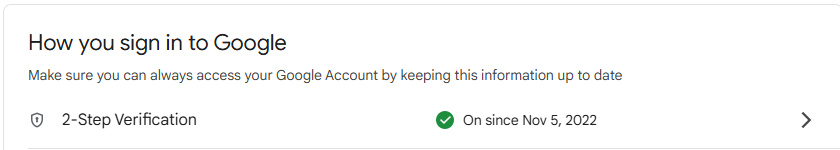


**Login and Check**

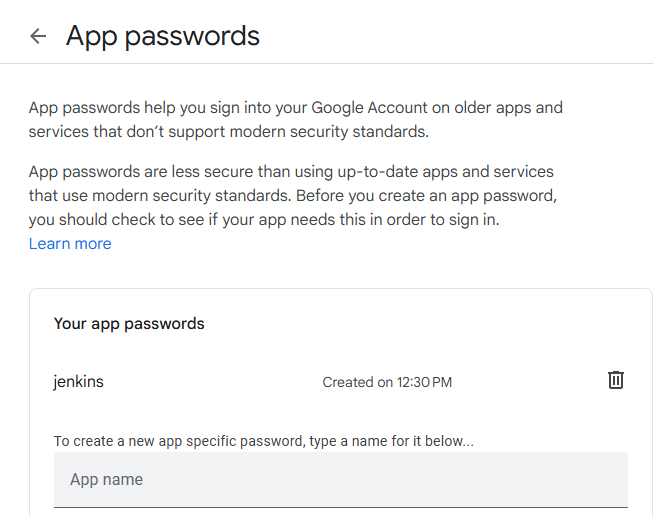


**Email Notification: -**

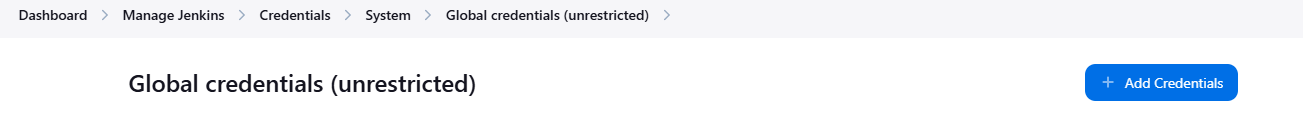
**Allow SMTP (465) in inbound rule**

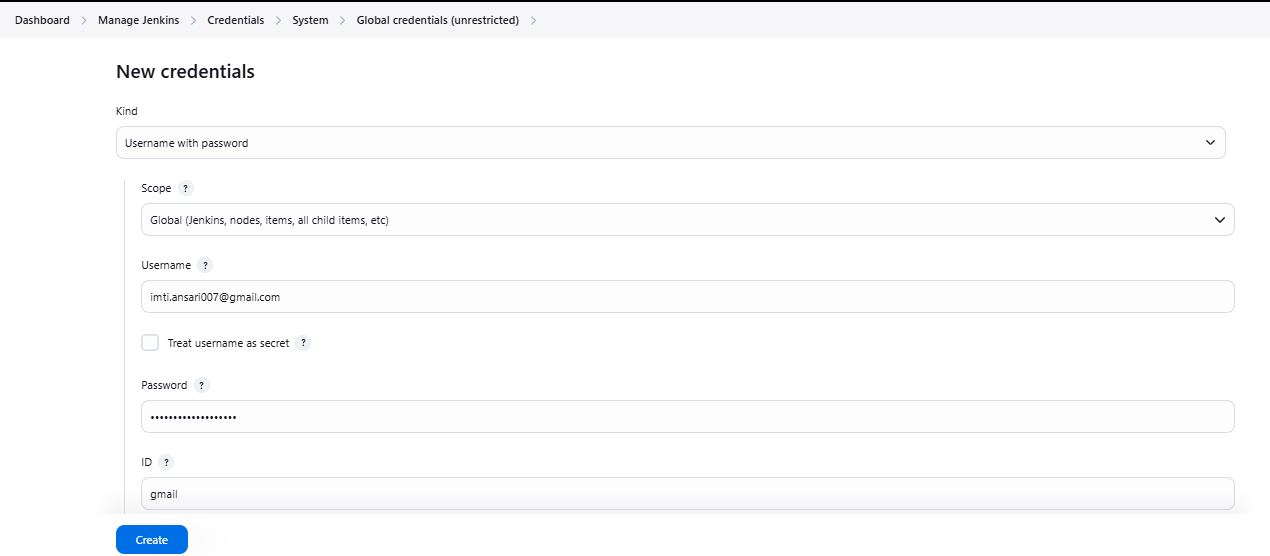
**Enable 2-step verification on your mail  
**

**Create App Password --- Gmail---Manage Security---search App.**

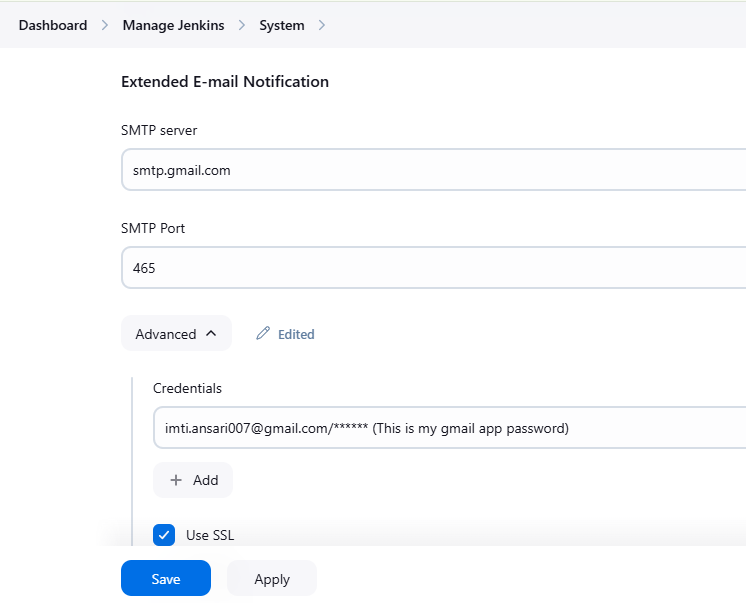
****

**Add credential in jenkins**

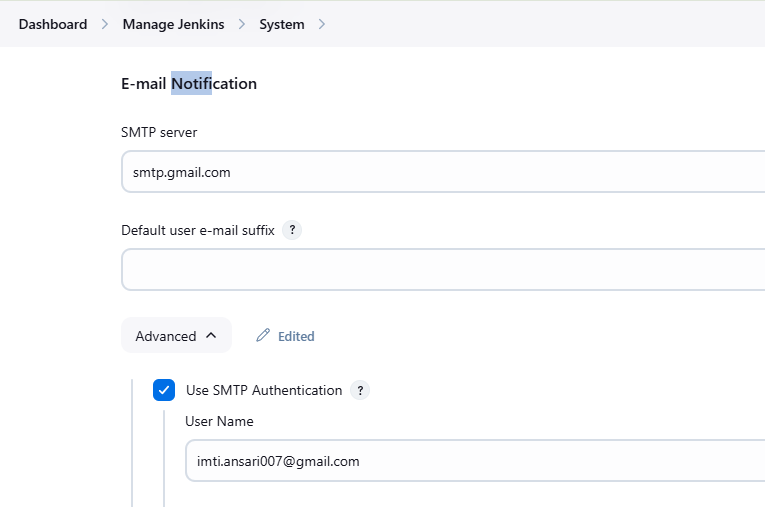
****

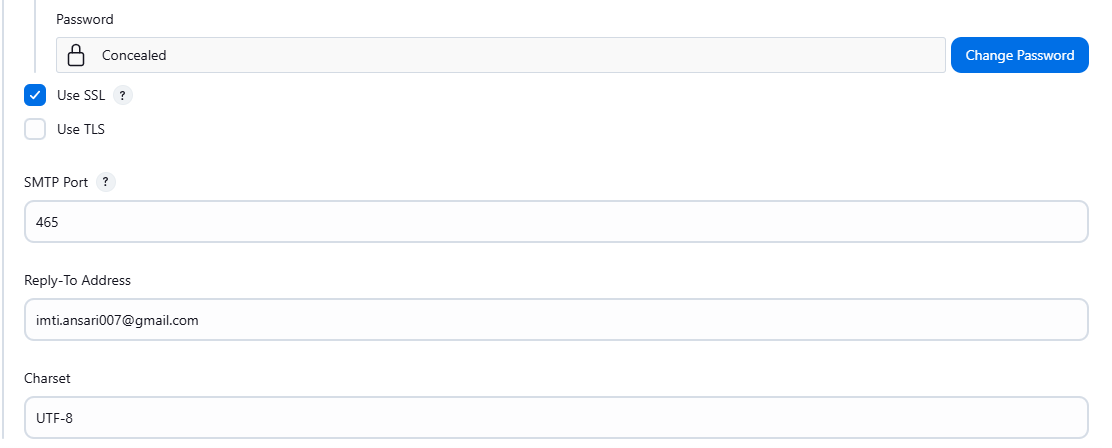
****

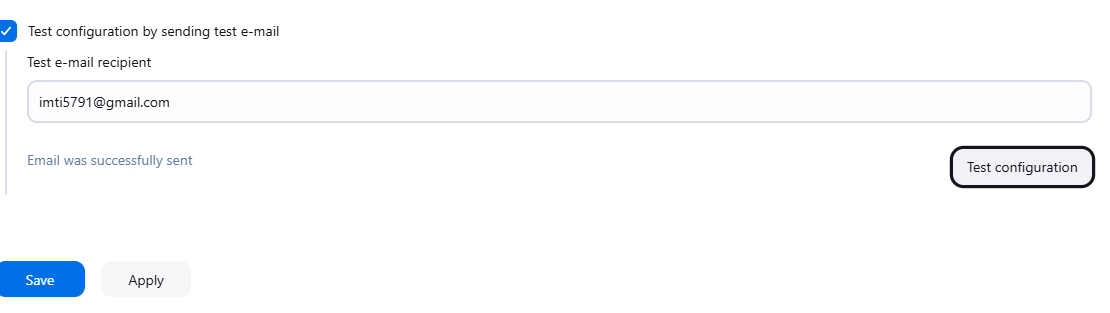
**Set Email notification on jenkins**

****

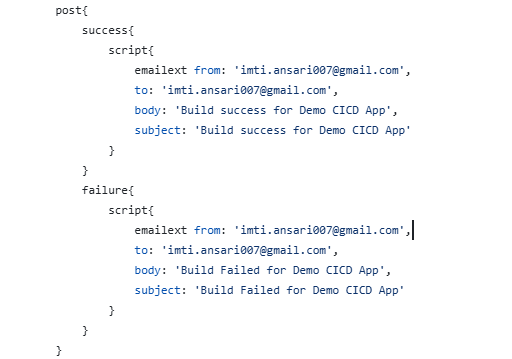
**Test Email sent or not**

****

****

****

**Now go to Jenkinsfile and set the script.**

****

**Multibranch Pipeline: -**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Go to git hub and set up webhook.  
http://52.19.64.169:8080/multibranch-webhook-trigger/invoke?token=mytoken**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Multi branch showing, in which branch Jenkins file update auto build the same branch.**

**Jenkins Troubleshoot**

**Gitcheckout --- repo url /user/token ---- 128 error if not provide user and token/password**

**Gitcheckout --- repo url /user/token ---403 provide wrong user and token/password**

**Pipeline fails --- 137 not enough memory**

**127---command or tool not found like mvn not found**

**When we run sonar analysis with Jenkins and we forget to configure the sonar server--- sonar not available**

**500 – You should check immediately server up or not (if push nexus artifact and getting error then check nexus server up or not if it is up then restart the server)**

**Jenkins Backup + Restore 1st type**

1. **Create a Jenkins server---create a job and build**
2. **Create a S3 bucket**
3. **Systemctl stop Jenkins**
4. **tar -zcvf jenkinsbackup.tar.gz /var/lib/jenkins/**
5. **IAM Roles with s3 full access policy and attach with Server**
6. **aws s3 cp jenkinsbackup.tar.gz s3:// jenkinsbackupaseem/jenkinsbackup.tar.gz**
7. **Now terminate the server**
8. **Create new Instance and install Jenkins for restore and attach IAM role**
9. **Systemctl stop Jenkins**
10. **aws s3 cp s3://jenkinsbackupaseem/jenkinsbackup.tar.gz jenkinsbackup.tar.gz**
11. **rm -rf /var/lib/Jenkins**
12. **tar -zxvf jenkinsbackup.tar.gz -C /**
13. **system start Jenkins**

**Jenkins Backup 2nd type**

**Install the plugin thin backup and configure it set crontab for automatic backup**

**Parameterized: -**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**