**Jenkins Installation on Ubuntu   
  
STEP\_1: Create a EC2 Machine in AWS (t2.micro)**

SSH to EC2 Machine created above -> sudp apt update -y

**STEP\_2: Install Openjdk Java 17**

 sudo apt install -y openjdk-17-jdk

**STEP\_3: Install Jenkins with bellow commands**

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key  
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null  
sudo apt-get update  
sudo apt-get install -y jenkins

**STEP\_4: Start Jenkins application suing systemctl**

To start Jenkins service  
      sudo systemctl start jenkins

To stop Jenkins service   
      sudo systemctl stop jenkins

To check current status Jenkins service   
      sudo systemctl status jenkins

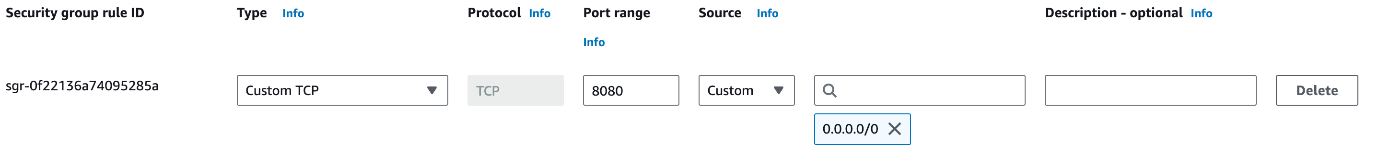
To edit current Jenkins service settings  
      sudo systemctl edit jenkins

**STEP\_5: To access Jenkins from client (Browser)**

Jenkins by default runs on 8080 port  
To open 8080 port to access   
     Goto -> EC2 -> jenkins\_instance -> Security (Tab)

  
Then  
A blue text on a white background

Description automatically generated  
Edit inbound rules  
  
     A white rectangular object with a black line

Description automatically generated with medium confidence  
  
Add inbound rule for 8080   
  
       
Access the jenkins with url <jenkins\_ec2\_public\_ip\_address>:<port> from any browser  
        A close up of a text

Description automatically generated  example: 13.233.131.159:8080

**STEP\_6: Initaial setup of jenkins (Unlock Jenkins)**

**A screenshot of a password

Description automatically generated**

**Run the below command in jenins EC2 machine**         
       sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
         (Note: copy and paste printed key in the above windows)

A close-up of a sign

Description automatically generated  
                       If you get above error means you copied some extra charactes with initial\_admin\_password in the terminal (like including new line)  
                      **Solution:  copy the password into any notepad application and copy only the password to jenkins  
A screenshot of a computer program

Description automatically generated**

**Add Users to jenkins**

Navigate to **Jenkins Dashboard** → **Manage Jenkins** → **Users** → **Create User**. Enter the user name, password, full name and email-address and then click **Create User** button.

**Project-based Matrix Authorization Strategy (User security and access control)**

Project-based Matrix Authorization Strategy allows you to define specific permissions for each project (job) within your Jenkins instance. With this strategy, you can specify which users or groups have particular permissions (such as build, configure, or delete) for individual projects.

Here's how you can set up Project-based Matrix Authorization in Jenkins:

* **Access Jenkins Configuration**: Log in to Jenkins with administrative privileges.
* **Navigate to Manage Jenkins**: Click on the "Manage Jenkins" link on the Jenkins dashboard.
* **Configure Global Security**: Under the "Configure Global Security" section, select the "Enable security" checkbox to enable security settings.
* **Select Project-based Matrix Authorization Strategy**: Scroll down to the "Authorization" section and choose "Project-based Matrix Authorization Strategy" from the available options.
* **Define Permissions**: You will see a matrix where you can define permissions for different users or groups. Here are some common permissions:
  + Overall: Permissions for overall Jenkins access.
  + Job: Permissions related to specific jobs or projects.
  + Run: Permissions to run jobs.
  + Configure: Permissions to configure jobs.
  + Read: Permissions to view job configurations and build results.
  + Workspace: Permissions to access the workspace of a job.
  + Cancel: Permissions to cancel builds.
  + Delete: Permissions to delete jobs.
* **Grant Permissions**: For each permission, you can specify which users or groups should have that permission by checking the corresponding checkboxes.
* **Save Configuration**: Once you have configured the permissions as per your requirements, click on the "Save" or "Apply" button to save the changes.
* **Test Permissions**: After configuring the permissions, it's essential to test them to ensure that users and groups have the appropriate access to Jenkins projects.

**Jenkins Freestyle Project/Job**

**General Configuration:**

* **Job Name:** Provide a meaningful name for the job to identify its purpose.
* **Description:** Optionally, add a description to provide additional context about the job's purpose.
* **Discard Old Builds:** Specifies the strategy for managing old builds to free up disk space  
  Options typically include keeping all builds, keeping only the last N builds, or keeping builds based on days.
* **Execute concurrent builds if necessary:**Allows Jenkins to execute multiple builds of this job concurrently if necessary.
* **This project is parameterized:** Allows you to define parameters for the job, enabling users to customize build behavior.  
  + Most used parameter types,  
    **String Parameter:**Accepts a single string value.  
    **Boolean Parameter:** Accepts a true/false value.  
    **Choice Parameter:**Presents a dropdown menu or radio buttons to select one option from a predefined list.

**Source Code Management (SCM):**

* **SCM:**Choose the version control system (e.g., Git) for your project.
* **Repository URL:** Provide the URL of your version control repository.
* **Credentials:** If required, provide credentials to authenticate with the version control system.

**Build Triggers:**

Specify conditions that trigger a build.

* **Build periodically:** Trigger builds based on a schedule (using cron syntax).
* **Poll SCM**: Trigger a build when changes are detected in the version control repository.
* **Build after other projects are built:** Trigger this job after other specified projects are built.
* **Trigger builds remotely:** Trigger builds remotely via HTTP.

**Build Environment:**

* **Build Environment:** Set up the environment for the build process.
* **Delete workspace before build starts:**Clean up the workspace before starting a new build.
* **Set Environment Variables:** Define environment variables for the build process.

**Build:**

* **Build:**Define the build steps for the job.
* **Execute shell:** Execute shell commands on Unix-like platforms.
* **Execute Windows batch command:** Execute batch commands on Windows.
* **Invoke Maven:** Execute Maven goals.

**Post-Build Actions:**

* **Post-Build Actions:** Define actions to be performed after the build completes.
* **Archive the artifacts:** Archive files (e.g., JAR, WAR) to keep them for future reference.
* **Email Notification:** Send email notifications to specified recipients after the build.
* **Trigger parameterized build:** Trigger another project with parameters.
* **Publish JUnit test result report:** Publish JUnit test results to Jenkins.

**Crontab (Job Scheduler)**

The crontab is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list. Crontab stands for “cron table, ” because it uses the job scheduler cron to execute tasks; cron itself is named after “chronos, ” the Greek word for time.cron is the system process which will automatically perform tasks for you according to a set schedule. The schedule is called the crontab, which is also the name of the program used to edit that schedule.

Linux Crontab Format:     \*         \*            \*          \*         \*              <command to execute>

       MIN    HOUR     DOM     MON     DOW                  CMD

FIELD DESCRIPTION ALLOWED VALUE

* MIN Minute field 0 to 59
* HOUR Hour field 0 to 23
* DOM Day of Month 1-31
* MON Month field 1-12
* DOW Day Of Week 0-6
* CMD Command Any command to be executed.

To view the Crontab entries: crontab -l

To edit Crontab Entries: crontab -e

To edit crontab entries of other Linux users: crontab -u username -e

example

1) To schedule a job for every minute using Cron

\* \* \* \* \* command/script

2) How to Execute a Linux Cron Jobs Every Second Using Crontab.

You cannot schedule an every-second cronjob. Because in cron the minimum unit you can specify is minute. 

3) To schedule a background Cron job for every 10 minutes.

\*/10 \* \* \* \* /home/maverick/check-disk-space

4) Schedule a Job for More Than One Instance (e.g. Twice a Day)

executes the specified script at 11:00 and 16:00 every day

00 11,16 \* \* \* /home/ramesh/bin/incremental-backup

5) Schedule a Job for Specific Range of Time (e.g. Only on Weekdays)

    This example checks the status of the database every day (including weekends) during the working hours 9 a.m – 6 p.m

00 09-18 \* \* \* /home/ramesh/bin/check-db-status

6) Cron job to run on the last day of the month

55 23 28-31 \* \* [[ "$(date --date=tomorrow +\%d)" == "01" ]] && myscript.sh 0 23 28-31 \* \* [ $(date -d +1day +%d) -eq 1 ] && myscript.sh

7) Cron special keywords and their meaning Keyword Equivalent

@yearly 0 0 1 1 \*

@daily 0 0 \* \* \*

@hourly 0 \* \* \* \*

@reboot Run at startup.

a) To schedule a job for the first minute of every year using @yearly

@yearly /home/maverick/bin/annual-maintenance

b) To schedule a Cron job beginning of every month using @monthly

@monthly /home/maverick/bin/tape-backup

c) To schedule a background job every day using @daily

@daily /home/maverick/bin/cleanup-logs "day started"

d) To execute a Linux command after every reboot using @reboot

@reboot CMD

**Add Users to jenkins**

Navigate to **Jenkins Dashboard** → **Manage Jenkins** → **Users** → **Create User**. Enter the user name, password, full name and email-address and then click **Create User** button.

**Project-based Matrix Authorization Strategy (User security and access control)**

Project-based Matrix Authorization Strategy allows you to define specific permissions for each project (job) within your Jenkins instance. With this strategy, you can specify which users or groups have particular permissions (such as build, configure, or delete) for individual projects.

Here's how you can set up Project-based Matrix Authorization in Jenkins:

* **Access Jenkins Configuration**: Log in to Jenkins with administrative privileges.
* **Navigate to Manage Jenkins**: Click on the "Manage Jenkins" link on the Jenkins dashboard.
* **Configure Global Security**: Under the "Configure Global Security" section, select the "Enable security" checkbox to enable security settings.
* **Select Project-based Matrix Authorization Strategy**: Scroll down to the "Authorization" section and choose "Project-based Matrix Authorization Strategy" from the available options.
* **Define Permissions**: You will see a matrix where you can define permissions for different users or groups. Here are some common permissions:
  + Overall: Permissions for overall Jenkins access.
  + Job: Permissions related to specific jobs or projects.
  + Run: Permissions to run jobs.
  + Configure: Permissions to configure jobs.
  + Read: Permissions to view job configurations and build results.
  + Workspace: Permissions to access the workspace of a job.
  + Cancel: Permissions to cancel builds.
  + Delete: Permissions to delete jobs.
* **Grant Permissions**: For each permission, you can specify which users or groups should have that permission by checking the corresponding checkboxes.
* **Save Configuration**: Once you have configured the permissions as per your requirements, click on the "Save" or "Apply" button to save the changes.
* **Test Permissions**: After configuring the permissions, it's essential to test them to ensure that users and groups have the appropriate access to Jenkins projects.

**Creadetials to Integrate with other tools**

Navigate to**Jenkins Dashboard → Manage Jenkins → Security → Credentials → (global) → Add Credentials**

* + For "Username with password", you'll need to enter the username and password.
  + For "SSH username with private key", you'll need to provide the username and paste the private key.
  + For "Secret text", you'll enter the secret value.
  + For "Secret file", you'll upload the file containing the secret.

**Select Credential Type**: Choose the type of credentials you want to add from the dropdown menu. Jenkins supports various types of credentials, such as username with password, SSH username with private key, secret text, secret file, etc. Select the appropriate type for your use case.

* **Save Credentials**: After entering the necessary details, click on the "OK" or "Save" button to add the credentials to Jenkins.
* **Use Credentials in Jobs**: Once the credentials are saved, you can use them in your Jenkins jobs and plugins. During job configuration, you'll have the option to select the appropriate credential from the list of available credentials.