Master:

Master is the main server (Admin Server). The Master's job is to handle: Scheduling build jobs. Dispatching builds to the slaves for the actual execution. Monitor the slaves.

Slave:

Responsible for executing the jobs assigned to them by the master.

**Steps to Execute Jenkins from the Command Prompt:**

**1. Navigate to the Directory Where jenkins.war Is Located**

Open a command prompt or terminal window, then navigate to the directory where you downloaded or placed the jenkins.war file.

Example:

bash

Copy code

**cd path/to/jenkins-directory**

**2. Start Jenkins with Java**

To run Jenkins, you simply use the following command:

bash

Copy code

java -jar jenkins.war

This will start Jenkins on the default port 8080. If the port is already in use or you want to specify a different port, you can do that using the --httpPort option.

Example to run Jenkins on port 8081:

bash

Copy code

**java -jar jenkins.war --httpPort=8081**

**3. Access Jenkins in Your Browser**

Once Jenkins has started successfully, you can access it by navigating to the following URL in your web browser:

arduino

Copy code

**http://localhost:8080**

If you specified a custom port (e.g., 8081), then you would use:

arduino

Copy code

http://localhost:8081

**4. Unlock Jenkins (First Time Setup)**

* The first time you run Jenkins, it will ask you to unlock it using an initial password.
* The password can be found in a file located in the Jenkins home directory. The path to this file is shown in the command prompt where Jenkins started.
* The file is usually located in:

javascript

Copy code

~/.jenkins/secrets/initialAdminPassword

You can open this file to retrieve the password.

**5. Complete the Setup**

Once you've unlocked Jenkins, follow the on-screen instructions to complete the setup, which typically involves installing recommended plugins and setting up an admin user

Note: To setup password we need follow below steps

S1: Navigate to the folder which is mentioned Jenkins home page via command prompt using ‘**cd’**

S2: To open file password file enter “**cat <file\_name>”**

**To check Jenkins is running:** Go to the respective Jenkins folder enter the command **“jenkins.exe status”**

**To stop Jenkins is running:** Open admin command promt, then enter the following command **“net stop jenkins”**

**To Open admin command promt:**

 Press Win + R to open the Run dialog.

 Type cmd in the box, but don’t press Enter yet.

 Hold down Ctrl + Shift and then press Enter. This opens the Command Prompt with administrator privileges.

**Jenkins User Name and Password – Palanisankar, Prakash#18**

**URL -** [**http://localhost:8080/**](http://localhost:8080/)

**Upstream:** Job that triggers another job.

Downstreen: job triggered by another job.

**Interview Questions:**

1. What’s the difference between continuous integration, continuous delivery, and continuous deployment?

**Continuous integration:**

Code changes are frequently pushed to a shared repository and tested regularly. The primary goal of CI is to detect integration issues early, which can improve the quality of the codebase.

**Continuous Delivery:**

After code changes pass CI, they are automatically deployed to a staging or testing environment. This is the final step before the code can be made live.

**Continuous Deployment:**

Continuous Deployment is the next step after Continuous Delivery. In this practice, every change that passes automated tests is automatically deployed to production without manual approval.

2. Role Automation Testers in CI/CD

 **Automating Test Creation**: Automation testers write tests for various application layers (unit, integration, functional, etc.) and **ensure they’re automated** to run in every part of the CI/CD pipeline.

 **Integrating Tests with CI/CD Tools**: The tests are integrated into CI/CD tools (e.g., Jenkins, GitLab CI, CircleCI), ensuring that every code commit triggers automated tests.

 **Monitoring and Feedback**: Automation testers monitor test results, and provide **immediate feedback** to developers. If tests fail, they are notified and can fix issues quickly.

 **Running Tests in Multiple Environments**: Automation testers ensure that tests run in both **staging** and **production-like environments** to ensure that the app works in real-world conditions.

 **Ensuring Fast and Reliable Releases**: Automation testers help make sure that tests are **fast, reliable**, and thorough so that the CI/CD pipeline remains efficient. The faster the tests, the faster the feedback loop, allowing developers to fix issues quickly and deploy with confidence.

3. How do you trigger CI/CD pipeline

* Push the code
* Schedule the trigger
* Manual Initiation
* Continuous integration