Day 29 Task: Jenkins Important interview Questions.

Jenkins Interview
Here are some Jenkins-specific questions related to Docker that one can use during a DevOps Engineer interview:

Questions: -

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

→

Continuous integration: -
Developers practicing continuous integration merge their changes back to the main branch as often as possible. The developer's changes are validated by creating a build and running automated tests against the build. By doing so, you avoid integration challenges that can happen when waiting for release day to merge changes into the release branch.

Continuous integration puts a great emphasis on testing automation to check that the a application is not broken whenever new commits are integrated into the main branch.

Continuous delivery: -

Continuous delivery is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage.

This means that on top of automated testing, you have an automated release process and you can deploy your application any time by clicking a button.

In theory, with continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements. However, if you truly want to get the benefits of continuous delivery, you should deploy to production as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem.

Continuous deployment: -
Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.

Continuous deployment is an excellent way to accelerate the feedback loop with your customers and take pressure off the team as there isn't a "release day" anymore. Developers can focus on building software, and they see their work go live minutes after they've finished working on it.

2. Benefits of CI/CD

→

Let’s look at five benefits of implementing a CI/CD pipeline to understand why many organizations have shifted toward this approach.

1. Reduce risk
Finding and fixing bugs late in the development process is expensive and time-consuming. This is especially true when there are issues with features that have already been released to production.

With a CI/CD pipeline, you can test and deploy code more frequently, giving testers the ability to detect issues as soon as they occur and to fix them immediately. You are essentially mitigating risks in real time.

2. Deliver faster
Organizations are moving toward releasing features multiple times a day. This is not an easy task; only a handful of companies like Netflix, Amazon, and Facebook have been

able to achieve this goal. But, with a seamless CI/CD pipeline, multiple daily releases can be made a reality.

Teams can build, test and deploy features automatically with almost no manual intervention. This is accomplished using various tools, frameworks, and systems like Travis CI, Docker, Kubernetes, and LaunchDarkly.

3. Expend less manual effort
Once you build features and check in code, tests should be automatically triggered to make sure that the new code does not break existing features and that the new features are working correctly.

After the tests run, the code gets deployed to different environments, including QA, staging and production. Throughout this process, you will be getting constant notifications through different channels, giving you plenty of information about the build, test and deploy cycles.

4. Generate extensive logs
One key aspect of observability is logging information. Logs are a rich source of information to understand what is happening beneath the UI and study application behavior.

With a CI/CD pipeline, extensive logging information is generated in each stage of the development process. There are various tools available to analyze these logs effectively and get immediate feedback about the system.

5. Make easier rollbacks
One of the biggest advantages of a CI/CD pipeline is you can roll back changes quickly. If any new code changes break the production application, you can immediately return the application to its previous state. Usually, the last successful build gets immediately deployed to prevent production outages.

3.What is meant by CI-CD?

→

Continuous Integration (CI): -

CI or **Continuous Integration** is the practice of automating the integration of code changes from multiple developers into a single codebase. It is a software development practice where the developers commit their work frequently into the central code repository (GitHub or Stash). Then there are automated tools that build the newly committed code and do a code review, etc. as required upon integration.

Continuous Delivery: -

CD or **Continuous Delivery** is carried out after Continuous Integration to make sure that we can release new changes to our customers quickly in an error-free way. This includes running integration and regression tests in the staging area (similar to the production environment) so that the final release is not broken in production. It ensures to automate the release process so that we have a release-ready product at all times and we can deploy our application at any point in time.

Continuous Delivery automates the entire software release process. The final decision to deploy to a live production environment can be triggered by the developer/project lead as required. Some popular CD tools are AWS Code Deploy, Jenkins, and GitLab.

4. What is Jenkins Pipeline?

→

In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence. In a Jenkins Pipeline, every job has some sort of dependency on at least one or more jobs or events.

Jenkins Pipeline can be defined by a text file called JenkinsFile. You can implement pipeline as code using JenkinsFile, and this can be defined by using a DSL (Domain Specific Language). With the help of JenkinsFile, you can write the steps required for running a Jenkins Pipeline.

JenkinsFile can be defined by using either Web UI or with a JenkinsFile.

Pipeline syntax
Two types of syntax are used for defining your JenkinsFile.

- Declarative -
  Declarative pipeline syntax offers a simple way to create pipelines. It consists of a predefined hierarchy to create Jenkins pipelines
- Scripted -

Scripted Jenkins pipeline syntax runs on the Jenkins master with the help of a lightweight executor.

5. How do you configure the job in Jenkins?

→

There are Multiple ways to configure the jobs. Here for demonstration purpose we will use the Freestyle project to configure the job.

- Go to the Jenkins dashboard and Click on New Item
- In the next screen, enter the Item name, in this case we have named it Hello world. Choose the ‘Freestyle project option’
- The next screen will come up in which you can specify the details of the job. You can Add the project Name and Description.
- you can also enter the GitHub URL of the repository there, in addition to this, you would need to click on the Add button for the credentials to add a user name and

password to the GitHub repository so that the code can be picked up from the remote repository.

- Now go to the Build section and click on Add build step → Execute shell
- In the command window, enter some commands and then click on the Save button.
- Once saved, you can click on the Build Now option to see if you have successfully defined the job.
- Once the build is completed, a status of the build will show if the build was successful or not.
- Click on the Console Output link to see the details of the build

5. Where do you find errors in Jenkins?

→

Here are the below ways, to find the errors in Jenkins: -

- Take a look at Log generated by the Jenkins
- you can look at /var/log/Jenkins/Jenkins.log via doing "tail -f /var/log/Jenkins/Jenkins.log" and check carefully what's breaking it.
- Check the system log file
- Few Jenkins plugins which can you help you https://wiki.jenkins-ci.org/display/JENKINS/Monitoring
- Make sure your Jenkins & installed plugins version is installed with most UpToDate stable release.

6.In Jenkins how can you find log files?

→

Jenkins offers an open-source CI/CD automation solution for developers. One of the features Jenkins offers is automatically logging the performance of builds.

How to View Jenkins Logs?

- To access the log, click the **Manage Jenkins** link on the right-hand side of the dashboard.
- Click the **System Log** button in the *Status Information* section.
- Click the **All-Jenkins Logs** link to access the default log.

Linux

The default location for Jenkins logs on Linux is **/var/log/Jenkins/Jenkins.log**. To view the log file, open it using a text editor such as Nano:

```
Sudo nano /var/log/Jenkins/Jenkins.log
```

7.Jenkins workflow and write a script for this workflow?

→

Jenkins Workflow is a plugin for **Jenkins**. Once installed, a new item type becomes available: a "Workflow". Workflow projects can be used for the same purposes as regular "Freestyle" Jenkins projects, but they also have the ability to orchestrate much larger tasks that can span multiple projects, and even create and manage multiple workspaces **in a single Workflow**.

To get started, head to the Jenkins Dashboard and click **New Item**. Name the new item "Shell Test (Workflow)", and select the **Workflow** type.

You'll notice that the configuration options differ from standard Jenkins projects. There are no longer any options to add a GitHub repo, build steps, or post-build actions. Instead, there's a new section called **Workflow**.

Within the Workflow section is a text box labeled **Script**. That is where you'll be defining the Workflow script Jenkins will run when it initializes a build of the project.

Simple Workflow Script: -

```
Node {
git 'https://github.com/redhotvengeance/hello-jenkins.git'
Sh 'uptime'
         }
```

8.How to create continuous deployment in Jenkins?

→

To create continuous deployment in Jenkins, you need to create a pipeline that implements the CI/CD process and includes steps for building, testing, and deploying code changes to production.

9.How build job in Jenkins?

→

To build a job in Jenkins, go to the Jenkins web interface, select a job, and click on "Build Now."

10.Why we use pipeline in Jenkins?

→

Jenkins Pipelines are used to automate the CI/CD process and make it easier to implement complex builds and deployments.

11.Is Only Jenkins enough for automation?

→

Jenkins is not enough for automation, it can be combined with other tools such as Ansible, Docker, and Kubernetes for a complete automation solution.

12.How will you handle secrets?

→

Secrets in Jenkins can be handled using the Jenkins Credentials plugin, which allows you to store sensitive information, such as passwords and API keys, securely.

13.Explain diff stages in CI-CD setup

→

Common stages in a CI/CD setup include building, testing, and deploying code changes.

14.Name some of the plugins in Jenkin?

→

Some popular plugins in Jenkins include the Blue Ocean plugin for creating pipelines, the Slack plugin for notifications, the Git plugin for version control, and the Docker plugin for containerization.