DEVOPS INTERVIEW QUESTIONS

Name two ways a Jenkins node agent can be configured to communicate back with the Jenkins master.

These are the mechanisms for starting a Jenkins node agent:

1. From the browser window launch a Jenkins node agent.
2. From the command line launch a Jenkins node agent.

When you launch a Jenkins node agent it will download a JNLP file. A new process is launched on the client machine by the JNLP when it runs

How to take a backup of your Jenkins build jobs?

Within the XML configuration each Jenkins build is stored. When this folder is copied, the configuration of all the build jobs that are managed by the Jenkins master are backed up. If you can perform a Jenkins Git integration, then it is good. When you copy the contents of the folder, you will see that the build jobs described in the folder will be restored when the Jenkins server is started the next time.

what are the steps included in a Jenkins pipeline.

A complete Jenkins pipeline will include building a project from the source code, putting it through a variety of unit, integrating, testing for user acceptance and performance and then finally deploying the packaged application on an application server.

So the steps in a Jenkins pipeline will include:

- Build
- Test
- Deploy

What is the process for creating a backup and copy files in Jenkins?

If you want to create a backup for your file then you need to regularly backup your Jenkins_Home directory. This will include all the build jobs configuration, all the slave node configuration and the build history. If you want to create a Jenkins backup, you can copy a job directory to the clone or can rename the directory.

From one server to another how do you copy or move your Jenkins jobs?

First you need to copy your jobs directory from the old to the new server. There are multiple ways to do it. You can either move the job from installation to simply copying the corresponding job directory, or you can make a clone of the job directory by making an existing job copy. First you need to have a different name which you can rename later.

What are the fundamental differences between DevOps & Agile?

The differences between the two are listed down in the table below.

DevOps vs Agile		
Features	**DevOps**	**Agile**
Agility	Agility in both Development & Operations	Agility in only Development
Processes/ Practices	Involves processes such as CI, CD, CT, etc.	Involves practices such as Agile Scrum, Agile Kanban, etc.
Key Focus Area	Timeliness & quality have equal priority	Timeliness is the main priority
Release Cycles/ Development Sprints	Smaller release cycles with immediate feedback	Smaller release cycles
Source of Feedback	Feedback is from self (Monitoring tools)	Feedback is from customers
Scope of Work	Agility & need for Automation	Agility only

How do all Devops tools work together?

Given below is a generic logical flow where everything gets automated for seamless delivery. However, this flow may vary from organization to organization as per the requirement.

1. Developers develop the code and this source code is managed by Version Control System tools like Git etc.
2. Developers send this code to the Git repository and any changes made in the code is committed to this Repository.
3. Jenkins pulls this code from the repository using the Git plugin and build it using tools like Ant or Maven.

4. Configuration management tools like puppet deploys & provisions testing environment and then Jenkins releases this code on the test environment on which testing is done using tools like selenium.
5. Once the code is tested, Jenkins send it for deployment on the production server (even production server is provisioned & maintained by tools like puppet).
6. After deployment It is continuously monitored by tools like Nagios.
7. Docker containers provides testing environment to test the build features.

What are the anti-patterns of DevOps?

A pattern is common usage usually followed. If a pattern commonly adopted by others does not work for your organization and you continue to blindly follow it, you are essentially adopting an anti-pattern. There are myths about DevOps. Some of them include:

- DevOps is a process
- Agile equals DevOps?
- We need a separate DevOps group
- Devops will solve all our problems
- DevOps means Developers Managing Production
- DevOps is Development-driven release management
    1. DevOps is not development driven.
    2. DevOps is not IT Operations driven.
- We can't do DevOps – We're Unique
- We can't do DevOps – We've got the wrong people

Describe branching strategies you have used.

This question is asked to test your branching experience so tell them about how you have used branching in your previous job and what purpose does it serves, you can refer the below points:

- Feature branching
  A feature branch model keeps all of the changes for a particular feature inside of a branch. When the feature is fully tested and validated by automated tests, the branch is then merged into master.
- Task branching
  In this model each task is implemented on its own branch with the task key included in the branch name. It is easy to see which code implements which task, just look for the task key in the branch name.
- Release branching
  Once the develop branch has acquired enough features for a release, you can clone that branch to form a Release branch. Creating this branch starts the next release cycle, so no new features can be added after this point, only bug fixes, documentation generation,

and other release-oriented tasks should go in this branch. Once it is ready to ship, the release gets merged into master and tagged with a version number. In addition, it should be merged back into develop branch, which may have progressed since the release was initiated.

What is Git rebase and how can it be used to resolve conflicts in a feature branch before merge?

According to me, you should start by saying git rebase is a command which will merge another branch into the branch where you are currently working, and move all of the local commits that are ahead of the rebased branch to the top of the history on that branch.
Now once you have defined Git rebase time for an example to show how it can be used to resolve conflicts in a feature branch before merge, if a feature branch was created from master, and since then the master branch has received new commits, Git rebase can be used to move the feature branch to the tip of master.
The command effectively will replay the changes made in the feature branch at the tip of master, allowing conflicts to be resolved in the process. When done with care, this will allow the feature branch to be merged into master with relative ease and sometimes as a simple fast-forward operation.

How will you secure Jenkins?

The way I secure Jenkins is mentioned below. If you have any other way of doing it, please mention it in the comments section below:

- Ensure global security is on.
- Ensure that Jenkins is integrated with my company's user directory with appropriate plugin.
- Ensure that matrix/Project matrix is enabled to fine tune access.
- Automate the process of setting rights/privileges in Jenkins with custom version controlled script.
- Limit physical access to Jenkins data/folders.
- Periodically run security audits on same.

What is configuration management in terms of infrastructure and mention a few popular tools used?
Ans: Configuration management consists of practices and the various tools involved to automate the delivery and infrastructure operations. It is all about keeping the server ready (**E.g.** Installing system packages, network configuration settings) for application deployment once the application is developed.
So the Ops or the system admin needs to ensure parity in different environments (Dev, QA, PROD etc…) by provisioning the systems.

What is Vagrant?

A vagrant is a tool which can create and manage virtualized environments for testing and developing software.

What are the containers?

Containers are from of lightweight virtualization. They offer isolation among processes.

Explain Pair Programming with reference to DevOps

Pair programming is an engineering practice of Extreme Programming Rules. In this method, two programmers work on the same system, on the same design/algorithm/code.

One programmer act as a “driver.” Other acts as an “observer” who continuously monitor the progress of a project to identify problems. The roles can be reversed at any point of time without any prior intimation.