Q: **Can you explain the CICD process in your current project? or can you talk about any CICD process that you have implemented?**

A: In the current project we use the following tools orchestrated with Jenkins to achieve CICD.

- Maven, Sonar, AppScan, ArgoCD, and Kubernetes

Coming to the implementation, the entire process takes place in 8 steps

1. Code Commit: Developers commit code changes to a Git repository hosted on GitHub.
2. Jenkins Build: Jenkins is triggered to build the code using Maven. Maven builds the code and runs unit tests.
3. Code Analysis: Sonar is used to perform static code analysis to identify any code quality issues, security vulnerabilities, and bugs.
4. Security Scan: AppScan is used to perform a security scan on the application to identify any security vulnerabilities.
5. Deploy to Dev Environment: If the build and scans pass, Jenkins deploys the code to a development environment managed by Kubernetes.
6. Continuous Deployment: ArgoCD is used to manage continuous deployment. ArgoCD watches the Git repository and automatically deploys new changes to the development environment as soon as they are committed.
7. Promote to Production: When the code is ready for production, it is manually promoted using ArgoCD to the production environment.
8. Monitoring: The application is monitored for performance and availability using Kubernetes tools and other monitoring tools.

Q: **What are the different ways to trigger Jenkins pipelines?**

A: This can be done in multiple ways,

To briefly explain about the different options,

```

- Poll SCM: Jenkins can periodically check the repository for changes and automatically build if changes are detected.

  This can be configured in the "Build Triggers" section of a job.

- Build Triggers: Jenkins can be configured to use the Git plugin, which allows you to specify a Git repository and branch to build.

  The plugin can be configured to automatically build when changes are pushed to the repository.
```

- Webhooks: A webhook can be created in GitHub to notify Jenkins when changes are pushed to the repository.
Jenkins can then automatically build the updated code. This can be set up in the "Build Triggers" section of a job and in the GitHub repository settings.
```

Q: **How to backup Jenkins?**

A: Backing up Jenkins is a very easy process, there are multiple default and configured files and folders in Jenkins that you might want to backup.
```
- Configuration: The `~/.jenkins` folder. You can use a tool like rsync to back up the entire directory to another location.

- Plugins: Backup the plugins installed in Jenkins by copying the plugins directory located in JENKINS_HOME/plugins to another location.

- Jobs: Backup the Jenkins jobs by copying the jobs directory located in JENKINS_HOME/jobs to another location.

- User Content: If you have added any custom content, such as build artifacts, scripts, or job configurations, to the Jenkins environment, make sure to backup those as well.

- Database Backup: If you are using a database to store information such as build results, you will need to back up the database separately. This typically involves using a database backup tool, such as MySQL dump for MySQL, to export the data to another location.
```
One can schedule the backups to occur regularly, such as daily or weekly, to ensure that you always have a recent copy of your Jenkins environment available. You can use tools such as CRON or Windows Task Scheduler to automate the backup process.

Q: **How do you store/secure/handle secrets in Jenkins?**

A: Again, there are multiple ways to achieve this,
Let me give you a brief explanation of all the possible options.
```
- Credentials Plugin: Jenkins provides a credentials plugin that can be used to store secrets such as passwords, API keys, and certificates. The secrets are encrypted and stored securely within Jenkins, and can be easily retrieved in build scripts or used in other plugins.

- Environment Variables: Secrets can be stored as environment variables in Jenkins and referenced in build scripts. However, this method is less secure because environment variables are visible in the build logs.

- Hashicorp Vault: Jenkins can be integrated with Hashicorp Vault, which is a secure secrets management tool. Vault can be used to store and manage sensitive information, and Jenkins can retrieve the secrets as needed for builds.

- Third-party Secret Management Tools: Jenkins can also be integrated with third-party secret management tools such as AWS Secrets Manager, Google Cloud Key Management Service, and Azure Key Vault.
```

Q: **What is latest version of Jenkins or which version of Jenkins are you using?**

A: This is a very simple question interviewers ask to understand if you are actually using Jenkins day-to-day, so always be prepared for this.

Q: **What is shared modules in Jenkins?**

A: Shared modules in Jenkins refer to a collection of reusable code and resources that can be shared across multiple Jenkins jobs. This allows for easier maintenance, reduced duplication, and improved consistency across multiple build processes.
For example, shared modules can be used in cases like:
```
- Libraries: Custom Java libraries, shell scripts, and other resources that can be reused across multiple jobs.

- Jenkins file: A shared Jenkins file can be used to define the build process for multiple jobs, reducing duplication and making it easier to manage the build process for multiple projects.

- Plugins: Common plugins can be installed once as a shared module and reused across multiple jobs, reducing the overhead of managing plugins on individual jobs.

- Global Variables: Shared global variables can be defined and used across multiple jobs, making it easier to manage common build parameters such as version numbers, artifact repositories, and environment variables.
```
```

Q: **can you use Jenkins to build applications with multiple programming languages using different agents in different stages?**

A: Yes, Jenkins can be used to build applications with multiple programming languages by using different build agents in different stages of the build process.

Jenkins supports multiple build agents, which can be used to run build jobs on different platforms and with different configurations. By using different agents in different stages of the build process, you can build applications with multiple programming languages and ensure that the appropriate tools and libraries are available for each language.

For example, you can use one agent for compiling Java code and another agent for building a Node.js application. The agents can be configured to use different operating systems, different versions of programming languages, and different libraries and tools.

Jenkins also provides a wide range of plugins that can be used to support multiple programming languages and build tools, making it easy to integrate different parts of the build process and manage the dependencies required for each stage.

Overall, Jenkins is a flexible and powerful tool that can be used to build applications with multiple programming languages and support different stages of the build process.

Q: **How to setup auto-scaling group for Jenkins in AWS?**

A: Here is a high-level overview of how to set up an autoscaling group for Jenkins in Amazon Web Services (AWS):
```
- Launch EC2 instances: Create an Amazon Elastic Compute Cloud (EC2) instance with the desired configuration and install Jenkins on it. This instance will be used as the base image for the autoscaling group.

- Create Launch Configuration: Create a launch configuration in AWS Auto Scaling that specifies the EC2 instance type, the base image (created in step 1), and any additional configuration settings such as storage, security groups, and key pairs.

- Create Autoscaling Group: Create an autoscaling group in AWS Auto Scaling and specify the launch configuration created in step 2. Also, specify the desired number of instances, the minimum number of instances, and the maximum number of instances for the autoscaling group.
```

- Configure Scaling Policy: Configure a scaling policy for the autoscaling group to determine when new instances should be added or removed from the group. This can be based on the average CPU utilization of the instances or other performance metrics.

- Load Balancer: Create a load balancer in Amazon Elastic Load Balancer (ELB) and configure it to forward traffic to the autoscaling group.

- Connect to Jenkins: Connect to the Jenkins instance using the load balancer endpoint or the public IP address of one of the instances in the autoscaling group.

- Monitoring: Monitor the instances in the autoscaling group using Amazon CloudWatch to ensure that they are healthy and that the autoscaling policy is functioning as expected.

By using an autoscaling group for Jenkins, you can ensure that you have the appropriate number of instances available to handle the load on your build processes, and that new instances can be added or removed automatically as needed. This helps to ensure the reliability and scalability of your Jenkins environment.
```

Q: **How to add a new worker node in Jenkins?**

A: Log into the Jenkins master and navigate to Manage Jenkins > Manage Nodes > New Node. Enter a name for the new node and select Permanent Agent. Configure SSH and click on Launch.

Q: **How to add a new plugin in Jenkins?**

A: Using the CLI,
`java -jar jenkins-cli.jar install-plugin <PLUGIN_NAME>`

Using the UI,

1. Click on the "Manage Jenkins" link in the left-side menu.
2. Click on the "Manage Plugins" link.
```

Q: **What is JNLP and why is it used in Jenkins?**

A: In Jenkins, JNLP is used to allow agents (also known as "slave nodes") to be launched and managed remotely by the Jenkins master instance. This allows Jenkins to distribute build tasks to multiple agents, providing scalability and improving performance.

When a Jenkins agent is launched using JNLP, it connects to the Jenkins master and receives build tasks, which it then executes. The results of the build are then sent back to the master and displayed in the Jenkins user interface.

Q: **What are some of the common plugins that you use in Jenkins?**

A: Be prepared for answer, you need to have at least 3-4 on top of your head, so that interview feels you use Jenkins on a day-to-day basis.