1. ***What is continuous integration?***

Continuous Integration (CI) is a software development methodology where developers — following the trunk-based model — merge their changes to the main branch many times per day.

CI is supported by automated tests and a build server that runs them on every change. As a result, failures are made visible as soon as they are introduced and can be fixed within minutes.

2. ***How do CI and version control relate to one another?***

Every change in the code must trigger a continuous integration process. This means that a CI system must be connected with a Git repository to detect when changes are pushed, so tests can be run on the latest revision.

3. ***What's the difference between continuous integration, continuous delivery, and continuous deployment?***

Continuous integration (CI) executes the sequence of steps required to build and test the project. CI runs automatically on every change committed to a shared repository, offering developers quick feedback about the project's state.
Continuous delivery is an extension of CI. Its goal is to automate every step required to package and release a piece of software. The output of a continuous delivery pipeline takes the form of a deployable binary, package, or container.
Continuous deployment is an optional step-up from continuous delivery. It is a process that takes the output from the delivery pipeline and deploys it to the production system in a safe and automated way.

4. ***What is the build stage?***

The build stage is responsible for building the binary, container, or executable program for the project. This stage validates that the application is buildable and provides a testable artifact.

5. ***What's the difference between a hosted and a cloud-based CI/CD platform?***

A hosted CI server must be managed like any other server. It must be first installed, configured, and maintained. Upgrades and patches must be applied to keep the server secure. Finally, failures in the CI server can block development and stop deployments.
On the other hand, a cloud-based CI platform does not need maintenance. There's nothing to install or configure, so organizations can immediately start using them. The cloud provides all the machine power needed, so scalability is not a problem. Finally, the reliability of the platform is guaranteed by SLA.

6. ***How do you use variables in the GitLab pipeline?***

GitLab variables can be used to pre-define the values. Go to your project page, Settings tab -> CI/CD, find Variables and click on the Expand button. Here you can define variable names and values, which will be automatically passed into the GitLab pipelines, and are available as environment variables there.

7. *What are CI/CD pipeline stages?*

There are four stages of a CI/CD pipeline:

- Source Stage
- Build Stage
- Test Stage
- Deploy Stage

8. *How does a CI CD pipeline works?*

A continuous integration/continuous delivery pipeline automates the software delivery process. The pipeline builds code, runs tests, and safely deploys a new version of the application. Automated pipelines remove manual errors, provide standardized feedback loops to developers, and enable fast product iterations.

9. *What's the difference between a hosted and a cloud-based CI/CD platform?*

A hosted CI server must be managed like any other server. It must be first installed, configured, and maintained. Upgrades and patches must be applied to keep the server secure. Finally, failures in the CI server can block development and stop deployments.
On the other hand, a cloud-based CI platform does not need maintenance. There's nothing to install or configure, so organizations can immediately start using them. The cloud provides all the machine power needed, so scalability is not a problem. Finally, the reliability of the platform is guaranteed by SLA.

10. *When is the best time to implement CI/CD?*

The transition to DevOps requires a complete reshaping of your software development culture, including the workflow, organizational structure as well as infrastructure. Therefore, organizations must prepare themselves for a major change when implementing DevOps.