# CI/CD Pipelines and Concepts Quiz (100 Questions)

DevOps Learning Module

This quiz covers fundamental concepts related to CI/CD pipelines, tools, and best practices. Choose the best answer for each question.

1. What does "CI" stand for in CI/CD?

   A. Continuous Integration

   B. Constant Integration

   C. Code Integration

   D. Continuous Inspection

   **Answer: A**
   **Explanation:** CI stands for Continuous Integration, the practice of frequently merging all developers' working copies to a shared mainline.

2. What is the primary goal of Continuous Integration (CI)?

   A. To deploy code to production automatically.

   B. To find and fix integration bugs early and often.

   C. To manage source code repositories.

   D. To run security scans.

   **Answer: B**
   **Explanation:** By integrating code frequently (often multiple times per day) and running automated builds and tests, CI helps catch integration issues before they become large problems.

3. What does "CD" stand for?

   A. Continuous Delivery

   B. Continuous Deployment

   C. Code Distribution

   D. Both A and B.

   **Answer: D**
   **Explanation:** CD can stand for either Continuous Delivery or Continuous Deployment, which are related but distinct practices that follow Continuous Integration.

4. What is the key difference between Continuous Delivery and Continuous Deployment?

A. There is no difference.

B. Continuous Delivery includes automated testing; Continuous Deployment does not.

C. Continuous Delivery: every change is releasable, but a *manual* step triggers the deploy. Continuous Deployment: every change is *automatically* deployed.

D. Continuous Deployment is for containers; Continuous Delivery is for VMs.

**Answer: C**
**Explanation:** This is the key distinction. Continuous Delivery ensures the code is *always* in a releasable state, but a human makes the final decision to deploy. Continuous Deployment automates that final step as well.

5. What is a "build artifact"?

A. The source code for a project.

B. A log file from a failed build.

C. The compiled, packaged, and versioned output of a build (e.g., a `.jar`, `.war`, Docker image, or `.zip` file).

D. A bug found during the build.

**Answer: C**
**Explanation:** The artifact is the "thing" that is built by the CI process. This artifact is then promoted through subsequent stages like testing and deployment.

6. Which popular open-source CI/CD server is known for its extensive plugin ecosystem and is written in Java?

A. GitLab CI

B. Travis CI

C. Jenkins

D. CircleCI

**Answer: C**
**Explanation:** Jenkins is one of the oldest and most widely-used CI/CD automation servers, famous for its flexibility and massive library of plugins.

7. What is a "pipeline" in the context of CI/CD?

A. The network connection to the server.

B. A feature in the Git protocol.

C. A defined set of automated steps (e.g., Build, Test, Deploy) that a code change goes through.

D. A list of project dependencies.

**Answer: C**
**Explanation:** A pipeline is the core of CI/CD. It defines the workflow of automated tasks that are triggered by a code change.

8. In a Jenkins pipeline, what is a "stage"?

   A. A single command.

   B. A distinct part of the pipeline, used for logical grouping (e.g., "Build", "Test", "Deploy").

   C. The server that Jenkins runs on.

   D. A plugin.

   **Answer: B**
   **Explanation:** Stages are used to visualize and organize the pipeline. Each stage can contain one or more steps.

9. What is the name of the file used to define a Jenkins Pipeline-as-Code?

   A. `.jenkins.yml`

   B. `Jenkinsfile`

   C. `config.xml`

   D. `pipeline.groovy`

   **Answer: B**
   **Explanation:** By convention, a `Jenkinsfile` is placed in the root of a source code repository. Jenkins reads this file to define and run the pipeline.

10. Jenkins Pipelines are written in which programming language?

    A. Python

    B. YAML

    C. Groovy (a DSL based on Groovy)

    D. Java

    **Answer: C**
    **Explanation:** Jenkins Pipelines use a Domain-Specific Language (DSL) that is written in Groovy.

11. What are the two syntax types for writing a Jenkinsfile?

    A. Declarative and Imperative

    B. Scripted and Declarative

    C. Simple and Advanced

    D. YAML and Groovy

    **Answer: B**
    **Explanation:** "Scripted" is the older, more flexible syntax, which is a full-fledged Groovy environment. "Declarative" is a newer, more structured, and opinionated syntax.

12. What is the name of the file used to configure CI/CD pipelines in GitLab?

    A. `gitlab-ci.yml`

B. `.gitlab-ci.yml`

C. `config.yml`

D. `Jenkinsfile`

**Answer: B**
**Explanation:** GitLab CI/CD pipelines are configured by a YAML file named `.gitlab-ci.yml` in the root of the repository.

13. What is a "Runner" in GitLab CI?

A. The main GitLab server.

B. The agent (a separate server or container) that executes the jobs defined in the pipeline.

C. The developer who triggers the pipeline.

D. A log file.

**Answer: B**
**Explanation:** A Runner is an agent that picks up CI jobs from the GitLab server, runs them, and reports the results back. Runners can be shared (for all projects) or specific (for one project).

14. What is "GitHub Actions"?

A. A set of Git commands.

B. A CI/CD and automation platform built directly into GitHub.

C. A project management tool within GitHub.

D. A tool for managing permissions.

**Answer: B**
**Explanation:** GitHub Actions is a powerful, event-driven automation platform that allows you to build, test, and deploy code directly from GitHub.

15. Where are GitHub Actions workflows defined?

A. In a `Actionsfile` in the root of the repo.

B. In `.github/workflows/` as YAML files.

C. In the repository settings on the website.

D. In a `.github.yml` file.

**Answer: B**
**Explanation:** Workflows are defined as YAML files (e.g., `main.yml`) inside the `.github/workflows/` directory.

16. In GitHub Actions, what is an "action"?

A. The entire workflow file.

B. A reusable, individual task (e.g., `actions/checkout@v2`).

C. A trigger, such as a `push`.

D. A log file.

**Answer: B**
**Explanation:** An "action" is a pre-built, reusable command or script. You combine multiple "actions" into "steps" to build a "job".

17. In GitHub Actions, what is an "event"?

    A. A failed build.

    B. A single command.

    C. A specific activity that triggers a workflow (e.g., `on: [push, pull_request]`).

    D. A build artifact.

    **Answer: C**
    **Explanation:** An event is the trigger for the pipeline. This can be a `push` to a branch, the creation of a `pull_request`, a new `release`, or even a scheduled `cron` event.

18. What is a "unit test"?

    A. A test of the entire application, end-to-end.

    B. A test of a small, isolated piece of code (like a single function or class).

    C. A manual test performed by a QA team.

    D. A test of the deployment process.

    **Answer: B**
    **Explanation:** Unit tests are the foundation of automated testing. They are fast, small, and verify the correctness of individual "units" of code in isolation.

19. What is an "integration test"?

    A. The same as a unit test.

    B. A test that verifies that multiple components (units) of the application work together correctly.

    C. A manual test.

    D. A test of the `.gitlab-ci.yml` file.

    **Answer: B**
    **Explanation:** Integration tests check the interactions *between* components, such as checking if your application code can correctly query the database.

20. What is an "end-to-end (E2E) test"?

    A. A test of a single function.

    B. A test that simulates a real user's workflow from start to finish (e.g., using a tool like Selenium to click buttons in a browser).

    C. A build script.

D. A unit test.

**Answer: B**
**Explanation:** E2E tests are the broadest tests. They verify the entire application flow (front-end, back-end, database, etc.) from a user's perspective.

21. What is "Infrastructure as Code" (IaC)?

    A. Writing your application code in a file.

    B. The practice of managing and provisioning infrastructure (servers, networks, etc.) through code and automation.

    C. A type of CI/CD pipeline.

    D. A security policy.

    **Answer: B**
    **Explanation:** IaC tools (like Terraform, Ansible, Chef, or Puppet) allow you to define your infrastructure in text files, which can then be versioned, tested, and deployed in a CI/CD pipeline.

22. Which IaC tool is primarily "declarative" and used for provisioning and managing multi-cloud infrastructure?

    A. Ansible

    B. Terraform

    C. Chef

    D. Jenkins

    **Answer: B**
    **Explanation:** Terraform is a declarative provisioning tool. You *declare* the desired state of your infrastructure (e.g., "I want 3 AWS EC2 instances"), and Terraform figures out how to make it happen.

23. Which IaC tool is primarily "procedural" (or "imperative") and used for configuration management (configuring software on existing servers)?

    A. Ansible

    B. Terraform

    C. Docker

    D. GitLab

    **Answer: A**
    **Explanation:** Ansible (like Chef and Puppet) is a configuration management tool. It's often procedural, as you define "plays" and "tasks" that are executed in order.

24. What is a "Blue-Green Deployment"?

    A. Deploying on Mondays (blue) and Wednesdays (green).

B. A deployment strategy where you have two identical environments ("Blue" and "Green"). You deploy to the inactive one, test, then switch traffic.

C. A deployment strategy that slowly rolls out the change to a few users at a time.

D. A deployment strategy that uses two colors on the dashboard.

**Answer: B**
**Explanation:** This strategy allows for zero-downtime deployments. The "Blue" environment is live, you deploy to "Green," and once "Green" is verified, you switch the load balancer to point to "Green."

25. What is a "Canary Release"?

    A. A deployment strategy that slowly releases the new version to a small subset of users (like 1% or 5%) before rolling it out to everyone.

    B. A deployment that fails and "sings" (alerts) the team.

    C. A Blue-Green deployment.

    D. A deployment that happens at night.

    **Answer: A**
    **Explanation:** This strategy is used to test the new version in production with minimal risk. If the "canary" group shows errors or poor performance, the deployment can be rolled back.

26. What is a "rollback" in CI/CD?

    A. The process of reverting a failed deployment to the last known-good version.

    B. The "roll forward" strategy of fixing a bug.

    C. Deleting the pipeline logs.

    D. The "Build" stage.

    **Answer: A**
    **Explanation:** A key part of automated deployment is having an automated rollback plan. If a deployment fails, the pipeline should ideally be able to automatically redeploy the previous stable version.

27. What is a "webhook"?

    A. A Git command.

    B. An automated HTTP callback sent from one system to another to notify it of an event.

    C. A type of deployment script.

    D. A security vulnerability.

    **Answer: B**
    **Explanation:** Webhooks are the "glue" of CI/CD. For example, GitHub sends a webhook to Jenkins on every `git push`, which tells Jenkins to trigger the build pipeline.

28. What technology is commonly used to create lightweight, portable, and consistent environments for building and deploying applications?

    A. Virtual Machines (VMs)

    B. Jenkins

    C. Git

    D. Containers (e.g., Docker)

    **Answer: D**
    **Explanation:** Containers (like Docker) are a core technology for modern CI/CD. They allow you to build an "immutable" image that contains your application and all its dependencies, ensuring it runs the same in dev, test, and prod.

29. What is a `Dockerfile`?

    A. A file that configures the Docker daemon.

    B. A text file containing instructions for building a Docker image.

    C. A running instance of a Docker image.

    D. A Docker CI/CD pipeline.

    **Answer: B**
    **Explanation:** A `Dockerfile` is a set of instructions (e.g., `FROM ubuntu`, `COPY . /app`, `RUN npm install`) that `docker build` uses to create an image.

30. What is a "Container Registry"?

    A. A server that runs containers.

    B. A repository (like Docker Hub, GCR, or ECR) for storing and distributing Docker images.

    C. The `docker build` command.

    D. A `Dockerfile`.

    **Answer: B**
    **Explanation:** In a CI/CD pipeline, the "Build" stage typically builds a Docker image, and the "Push" stage pushes that image to a container registry. The "Deploy" stage then pulls it from the registry.

31. What is "Kubernetes" (K8s)?

    A. A CI/CD server.

    B. A container runtime (like Docker).

    C. A container orchestration platform for automating the deployment, scaling, and management of containerized applications.

    D. A configuration management tool.

    **Answer: C**
    **Explanation:** Kubernetes is often the "deployment" target in a CI/CD pipeline. The

pipeline builds an image, and then (e.g., using `kubectl`) tells Kubernetes to update the running application with the new image.

32. What is a "declarative" pipeline?

    A. A pipeline where you define *what* the pipeline should do, not *how* it should do it.

    B. A pipeline where you write procedural scripts.

    C. A pipeline that is run manually.

    D. A Jenkins Scripted Pipeline.

    **Answer: A**
    **Explanation:** Declarative syntax (used by Jenkins Declarative, GitLab CI, GitHub Actions) is a structured format where you *declare* the stages, steps, and conditions.

33. What is "static code analysis" (or "linting")?

    A. Running unit tests.

    B. Analyzing source code for potential errors, style violations, and security vulnerabilities without actually running the code.

    C. Running integration tests.

    D. Manual code review.

    **Answer: B**
    **Explanation:** Static Analysis (SAST) tools (like SonarQube, `eslint`, `puppet-lint`) are almost always run in the "Test" or "Build" stage of a CI pipeline.

34. What is a common branching strategy that works well with CI/CD?

    A. Committing all changes directly to the `main` branch.

    B. Feature Branch Workflow (e.g., GitHub Flow).

    C. SVN Workflow.

    D. Having one branch per developer that never gets merged.

    **Answer: B**
    **Explanation:** A Feature Branch Workflow, where each new feature is developed in its own branch and then merged into `main` (often via Pull Request), is the standard for CI/CD.

35. What is the purpose of a "Pull Request" (PR) in a CI/CD workflow?

    A. To download code.

    B. To provide a place for manual code review and to trigger automated CI builds (tests) *before* the code is merged.

    C. To deploy code to production.

    D. To create a new branch.

**Answer: B**
**Explanation:** Modern CI systems (like GitHub Actions) are heavily integrated with PRs. Opening a PR automatically runs the test suite against the proposed changes.

36. What is an "artifact repository"?

    A. A Git repository.

    B. A server (like Artifactory, Nexus, or GitHub Packages) that stores binary build artifacts (e.g., `.jar`, `.zip`, `.rpm`).

    C. A Docker registry.

    D. A database of failed builds.

    **Answer: B**
    **Explanation:** You don't store large binary artifacts in Git. The CI build produces an artifact, and the "Push" stage uploads it to an artifact repository, where it is versioned.

37. What is "DevSecOps"?

    A. A security-focused CI/CD tool.

    B. A team of security engineers.

    C. The practice of integrating security testing and best practices into every stage of the DevOps (CI/CD) lifecycle.

    D. A tool for managing secrets.

    **Answer: C**
    **Explanation:** DevSecOps is a cultural shift. It means "shifting left" on security—running security scans (SAST, DAST, dependency checks) as part of the automated pipeline, not just at the end.

38. What is "SAST"?

    A. Static Application Security Testing (analyzing source code for vulnerabilities).

    B. Dynamic Application Security Testing (testing the running application).

    C. Secure Application Software Test.

    D. A type of integration test.

    **Answer: A**
    **Explanation:** SAST tools scan the "static" source code. They are "white-box" tests.

39. What is "DAST"?

    A. Static Application Security Testing.

    B. Dynamic Application Security Testing (attacking a running application to find vulnerabilities).

    C. Data Application Security Test.

    D. A type of unit test.

**Answer: B**
**Explanation:** DAST tools are "black-box" tests. They run against a deployed, "dynamic" application and try to find vulnerabilities (like SQL injection) from the outside.

40. What is "SCA" (Software Composition Analysis)?

    A. A test of the application's performance.

    B. A tool that scans your project's *dependencies* (e.g., npm, Maven) for known vulnerabilities (CVEs).

    C. A static code analysis tool.

    D. A deployment strategy.

    **Answer: B**
    **Explanation:** SCA is a critical part of DevSecOps. It checks all your open-source libraries against a database of known vulnerabilities.

41. What is a "build matrix"?

    A. A list of all build artifacts.

    B. A feature in CI/CD tools that allows you to run the same job multiple times with different configurations (e.g., different OS, different language versions).

    C. A dashboard showing build health.

    D. A Jenkins plugin.

    **Answer: B**
    **Explanation:** A build matrix is used to test a project's compatibility. You might define a matrix to test against Python 3.8, 3.9, and 3.10, and on both `ubuntu-latest` and `windows-latest`.

42. What is a "hosted" CI/CD platform?

    A. A tool you install and manage on your own servers (e.g., Jenkins).

    B. A tool that is managed for you by a vendor (e.g., GitHub Actions, CircleCI, Travis CI).

    C. A tool that only works with Git.

    D. A tool that is open-source.

    **Answer: B**
    **Explanation:** A "hosted" or "SaaS" (Software-as-a-Service) platform means you don't have to manage the CI/CD server itself. A "self-hosted" tool (like Jenkins) requires you to manage the server.

43. What is the purpose of caching in a CI/CD pipeline?

    A. To speed up builds by saving and reusing files that don't change often (e.g., dependencies).

    B. To store build artifacts.

C. To store build logs.

D. To store secrets.

**Answer: A**
**Explanation:** Caching (e.g., ~/.npm or ~/.m2) saves time and bandwidth by not re-downloading all project dependencies on every single build.

44. What is "Monitoring" in the context of CI/CD?

A. Watching the build logs scroll by.

B. The process of collecting and analyzing data (metrics, logs) from a deployed application to check its health and performance.

C. Running security scans.

D. Manual testing.

**Answer: B**
**Explanation:** Monitoring is the "feedback loop" of DevOps. After deployment, you must monitor the application (e.g., with Prometheus, Datadog) to ensure the deployment was successful.

45. What is a "rolling deployment"?

A. A deployment that fails and rolls back.

B. A deployment strategy where the new version is deployed to servers one by one, replacing the old version slowly.

C. A Blue-Green deployment.

D. A deployment that happens on a schedule.

**Answer: B**
**Explanation:** In a 10-server cluster, a rolling deployment updates server 1, then server 2, then server 3, and so on. This ensures the application stays online (on the other 9 servers) during the update.

46. What is "GitFlow"?

A. A simple branching model with just `main` and feature branches.

B. A complex, strict branching model with long-lived `main` and `develop` branches, plus `feature`, `release`, and `hotfix` branches.

C. A CI/CD tool from GitHub.

D. A Git command.

**Answer: B**
**Explanation:** GitFlow is a very structured (and often complex) branching model. It is often contrasted with the simpler "GitHub Flow."

47. What is "GitHub Flow"?

A. The same as GitFlow.

B. A simple branching model where `main` is always deployable, and all work is done in short-lived feature branches.

C. The workflow file for GitHub Actions.

D. A tool for reviewing pull requests.

**Answer: B**
**Explanation:** GitHub Flow is a lightweight model: create a branch, add commits, open a pull request, review and test, then merge and deploy. It is very well-suited for Continuous Deployment.

48. What is a "trigger" in CI/CD?

A. The event that starts a pipeline (e.g., a git push, a PR, a schedule).

B. The agent that runs the job.

C. The final "deploy" step.

D. A failed build.

**Answer: A**
**Explanation:** The trigger is the "cause" of the pipeline run. The most common trigger is a `push` to the version control system.

49. What is "Trunk-Based Development"?

A. The same as GitFlow.

B. A practice where all developers commit directly to a single shared branch (the "trunk", e.g., `main`), or use very short-lived branches.

C. A practice of using many long-lived branches.

D. A CI/CD tool.

**Answer: B**
**Explanation:** This is a core practice that enables true Continuous Integration. By keeping branches short-lived (or non-existent), everyone integrates their code constantly.

50. What is "secrets management" in CI/CD?

A. Hiding the build logs.

B. The practice of securely storing and providing sensitive data (like API keys, passwords, certificates) to the pipeline.

C. Encrypting the source code.

D. A type of security scan.

**Answer: B**
**Explanation:** You must *never* hard-code secrets in source code. Secrets management involves using a tool (like HashiCorp Vault, or the built-in secrets in GitHub/GitLab) to inject secrets at runtime.

51. Which tool is a popular, open-source tool dedicated to secrets management?

A. Jenkins

B. HashiCorp Vault

C. Terraform

D. Docker

**Answer: B**
**Explanation:** Vault is the industry-standard tool for securely storing, accessing, and rotating secrets, and is often integrated into CI/CD pipelines.

52. What is a "linting" (or "linter") tool?

A. A tool that runs unit tests.

B. A tool that checks code for stylistic errors, formatting, and simple bugs (e.g., `eslint` for JavaScript).

C. A tool that deploys code.

D. A tool that merges branches.

**Answer: B**
**Explanation:** Linting is a form of static analysis. It's almost always the first check in a CI pipeline, as it's very fast and catches simple mistakes.

53. What is a "mutable" artifact?

A. An artifact (like a Docker image) that can be changed after it is created.

B. An artifact that is perfect and can never be changed.

C. An artifact that is built from a template.

D. An artifact that is stored in Git.

**Answer: A**
**Explanation:** A mutable artifact (e.g., a Docker `:latest` tag that gets overwritten) is dangerous because the "thing" you tested is not the same "thing" you deploy.

54. What is an "immutable" artifact?

A. An artifact that can be easily changed.

B. An artifact (like a Docker image with a unique tag, e.g., `my-app:1.2.5`) that is never changed once it's built.

C. A failed build artifact.

D. A build script.

**Answer: B**
**Explanation:** This is a core best practice. You build an immutable artifact *once*, then promote that *exact* artifact from dev to test to prod. This ensures consistency.

55. What does the "Build" stage in a typical pipeline do?

A. Runs unit tests and linter.

B. Deploys the code to production.

C. Compiles the code, runs unit tests/linting, and packages it into an artifact (e.g., a Docker image).

D. Runs security scans.

**Answer: C**
**Explanation:** The "Build" (or "CI") stage is responsible for taking the source code and turning it into a deployable, versioned artifact, while also running fast, local tests.

56. What is a "pipeline as code"?

    A. The practice of defining your CI/CD pipeline (stages, steps, etc.) in a text file that is stored in version control.

    B. Writing your pipeline in a GUI.

    C. Storing your pipeline logs in Git.

    D. A tool for writing pipelines.

    **Answer: A**
    **Explanation:** This is the modern standard. By storing the pipeline definition (e.g., `Jenkinsfile`, `.gitlab-ci.yml`) in Git, it is versioned, reviewable, and can be changed along with the application code.

57. What is a "golden image"?

    A. A Docker image that is very large.

    B. A base VM or container image that is pre-configured with the OS, patches, and common tools, used as a starting point.

    C. The `ubuntu:latest` image.

    D. A failed build.

    **Answer: B**
    **Explanation:** "Baking" a golden image (e.g., with Packer) is a common CI task. It speeds up deployments because you don't have to install/patch the OS every time.

58. Which tool is often used to build "golden images" (VMs, AMIs, Docker images) in an automated way?

    A. Jenkins

    B. Packer (by HashiCorp)

    C. Terraform

    D. Git

    **Answer: B**
    **Explanation:** Packer is a tool specifically for building machine images from a single JSON template, and it's often run as a step in a CI pipeline.

59. In GitLab CI, what is a "job"?

A. The entire pipeline.

B. A "stage" (like "test").

C. A specific command (e.g., `unit-test`) that runs as part of a "stage".

D. A GitLab Runner.

**Answer: C**
**Explanation:** The hierarchy is: Pipeline -> Stages -> Jobs. A "stage" (e.g., `test`) can contain multiple "jobs" (e.g., `unit-test`, `lint-test`) that can run in parallel.

60. In GitLab CI, what does the `image` keyword define?

A. The application artifact.

B. The Docker image to use for the job's environment.

C. The logo for the project.

D. A VM image.

**Answer: B**
**Explanation:** GitLab CI (and many others) are container-native. The `image: node:16` keyword tells the Runner to spin up a Docker container using that image and run the job's `script` inside it.

61. In GitLab CI, what is the `script` keyword used for?

A. To define the Docker image.

B. To define the "stage" of the job.

C. To define the shell commands that the Runner should execute for this job.

D. To define the pipeline trigger.

**Answer: C**
**Explanation:** The `script` block contains the actual commands (e.g., `npm install`, `npm test`) that are run to perform the job.

62. In a `Jenkinsfile`, what is an "agent"?

A. The developer.

B. The `Jenkinsfile` itself.

C. The environment (e.g., a node, label, or Docker container) where the pipeline or stage will run.

D. The trigger for the pipeline.

**Answer: C**
**Explanation:** The `agent` directive (e.g., `agent any`, `agent { label 'linux' }`, `agent { docker 'node:16' }`) tells Jenkins *where* to execute the steps.

63. In a Jenkins Declarative Pipeline, what is the `post` section used for?

A. To define the "Deploy" stage.

B. To define actions that run at the *end* of the pipeline or stage, based on the result (e.g., `always`, `success`, `failure`).

C. To post a message to Slack.

D. To define the trigger.

**Answer: B**
**Explanation:** The `post` block is crucial for cleanup and notifications. You can have a `failure` block to send a Slack alert, and an `always` block to run `junit` (test reporting).

64. What is CircleCI?

A. A configuration management tool.

B. A popular, hosted (SaaS) CI/CD platform.

C. A Jenkins plugin.

D. A security scanner.

**Answer: B**
**Explanation:** CircleCI is a major competitor to GitHub Actions and GitLab CI. It is known for its high performance, caching, and use of a `.circleci/config.yml` file.

65. What is Travis CI?

A. A popular, hosted (SaaS) CI/CD platform, one of the first to become popular for open-source projects.

B. A self-hosted CI server.

C. A tool from HashiCorp.

D. A CI/CD platform for mobile apps only.

**Answer: A**
**Explanation:** Travis CI was a pioneer in hosted CI/CD. It uses a `.travis.yml` file and was extremely popular in the open-source community.

66. What is "observability"?

A. A more advanced form of monitoring, focusing on logs, metrics, and traces.

B. The practice of manual testing.

C. The name of a CI/CD tool.

D. The act of running `tail -f` on a log.

**Answer: A**
**Explanation:** Observability is the "feedback" part of the loop. It's about having the "three pillars" (logs, metrics, traces) to understand the *internal* state of your application after deployment.

67. What is a "distributed tracing"?

A. A type of build matrix.

B. A pillar of observability, used to trace a single user request as it flows through multiple microservices.

C. A Git command.

D. A type of unit test.

**Answer: B**
**Explanation:** In a microservices architecture, a single click can touch 10+ services. Tracing (e.g., with Jaeger or OpenTelemetry) gives you a "trace" of that request's entire journey.

68. What is "value stream mapping" in DevOps?

A. A security scan.

B. A diagram showing the flow of a user request.

C. An exercise to identify and visualize all the steps (and waste) from a code idea to production.

D. A database schema.

**Answer: C**
**Explanation:** This is a key "DevOps" practice. The CI/CD pipeline is the *automation* of the value stream, which is the process of delivering value to the customer.

69. What is "mean time to recovery" (MTTR)?

A. The average time it takes to build the code.

B. The average time it takes to run the unit tests.

C. The average time it takes to recover from a failure in production.

D. The average time it takes to get a PR approved.

**Answer: C**
**Explanation:** This is a key metric for DevOps success. A good CI/CD pipeline (with good testing and rollback procedures) *dramatically* lowers MTTR.

70. What is "deployment frequency"?

A. How often the CI server crashes.

B. How often developers commit code.

C. How often new code is deployed to production.

D. How often unit tests are run.

**Answer: C**
**Explanation:** This is another key DevOps metric. "Elite" teams, enabled by CI/CD, deploy multiple times per day, while low-performing teams might deploy once every few months.

71. What is "gated check-in" (or "gated commit")?

A. A system where a developer's `push` is blocked *until* the CI server has successfully built and tested the changes.

B. A manual approval step on a Pull Request.

C. A security feature of Git.

D. A CI/CD tool.

**Answer: A**
**Explanation:** This is a very strict form of CI. Instead of "breaking the build" after a push, the system *prevents* the push from ever reaching `main` if it fails the tests.

72. What is the purpose of a `before_script` block in GitLab CI?

A. To run commands *after* the main `script`.

B. To run commands *before* every job's main `script` (e.g., to install dependencies).

C. To define the Docker image.

D. To run only if the job fails.

**Answer: B**
**Explanation:** The `before_script` block is used for setup tasks (like `apt-get update` or logging in to a registry) that are needed by the main `script` block.

73. What is an "environment variable" in a CI/CD pipeline?

A. A variable that is hard-coded in the source code.

B. A key-value pair (e.g., `DATABASE_URL`) that is injected into the build environment.

C. A type of artifact.

D. A build log.

**Answer: B**
**Explanation:** Environment variables are used to pass configuration and secrets to the pipeline, so the scripts are not hard-coded.

74. Why should you *never* hard-code secrets (like passwords) in a `.gitlab-ci.yml` or `Jenkinsfile`?

A. It is too difficult to type them.

B. Because these files are stored in Git, the secrets would be visible to everyone with access.

C. The pipeline will fail.

D. It slows down the build.

**Answer: B**
**Explanation:** This is a critical security rule. All secrets must be stored in the CI/CD tool's "secrets" or "variables" section, or in an external vault.

75. What is "dependency scanning" in DevSecOps?

A. The same as SAST.

B. The same as DAST.

C. The process of checking your `package.json`, `pom.xml`, etc., for third-party libraries with known vulnerabilities (CVEs).

D. Manually reviewing dependencies.

**Answer: C**
**Explanation:** This is another name for Software Composition Analysis (SCA). Tools like `npm audit` or `Snyk` perform this scan.

76. What is a "multi-stage" Docker build?

    A. A `Dockerfile` that builds multiple images.

    B. A `Dockerfile` that uses multiple `FROM` statements to separate the build environment from the final runtime environment.

    C. A `Dockerfile` that is run in a CI pipeline.

    D. A `Dockerfile` that fails.

    **Answer: B**
    **Explanation:** This is a best practice. You use a large "builder" image (e.g., `FROM golang:1.18`) to compile your app, then use a small "runtime" image (e.g., `FROM alpine`) and just `COPY` the compiled binary. This results in a very small, secure final image.

77. In a Kubernetes deployment, what is a "rolling update"?

    A. A Blue-Green deployment.

    B. A Canary deployment.

    C. The default strategy, where Kubernetes slowly terminates old pods and creates new ones, one by one.

    D. A manual deployment.

    **Answer: C**
    **Explanation:** This is Kubernetes's built-in "rolling deployment" strategy. It ensures zero downtime by slowly replacing the application's pods with the new version.

78. What is "GitOps"?

    A. Using Git from the command line.

    B. A way of implementing Continuous Delivery where Git is the "single source of truth" for the desired state of the infrastructure.

    C. A CI/CD tool.

    D. A security policy for Git.

    **Answer: B**
    **Explanation:** In GitOps, the *entire* state of the cluster (e.g., Kubernetes manifests) is stored in a Git repo. An agent (like Argo CD or Flux) inside the cluster "pulls" changes from Git and applies them.

79. What is a key difference between "push-based" and "pull-based" deployments?

    A. "Push" is for Git; "Pull" is for SVN.

    B. "Push": The CI server (e.g., Jenkins) *pushes* the changes to the server. "Pull": An agent *pulls* the changes from a repo (GitOps).

    C. "Push" is manual; "Pull" is automatic.

    D. There is no difference.

    **Answer: B**
    **Explanation:** Traditional CI/CD is "push-based" (Jenkins runs `kubectl apply`). GitOps is "pull-based" (an agent like Argo CD watches Git and applies changes).

80. What is a "Helm Chart"?

    A. A dashboard of build metrics.

    B. A "package manager" for Kubernetes, bundling all the YAML manifests for an application.

    C. A type of Git branch.

    D. A security policy.

    **Answer: B**
    **Explanation:** Helm Charts are "packages" of templated Kubernetes YAML. A CI/CD pipeline will often package the application as a Helm Chart and push it to a chart repository.

81. What is "Test-Driven Development" (TDD)?

    A. A practice where you write the (failing) unit test *before* you write the application code.

    B. A practice where you write tests *after* you write the code.

    C. A type of integration test.

    D. A manual testing strategy.

    **Answer: A**
    **Explanation:** TDD follows a "Red-Green-Refactor" cycle: write a failing test (Red), write the minimal code to pass the test (Green), then clean up the code (Refactor). It's highly compatible with CI.

82. What is "SonarQube"?

    A. A CI/CD server.

    B. An artifact repository.

    C. A popular, open-source platform for continuous inspection of code quality (static analysis, security, code coverage).

    D. A deployment tool.

**Answer: C**
**Explanation:** A CI pipeline is often configured to send build results and code coverage reports to SonarQube, which provides a dashboard and can "fail the build" if quality gates aren't met.

83. What is "code coverage"?

    A. A measure of how many developers have reviewed the code.

    B. A metric (in percentage) of how much of your application code is executed by your automated tests.

    C. The number of lines of code in your project.

    D. A tool for static analysis.

    **Answer: B**
    **Explanation:** Code coverage tools (like JaCoCo, Istanbul) are run during the "Test" stage. A low coverage (e.g., 20%) indicates that 80% of your code is not being tested.

84. What is the "build breaker"?

    A. A person who intentionally breaks the build.

    B. A practice where a CI build is configured to "fail" (and thus "break") if it does not meet a quality threshold (e.g., <80% test coverage).

    C. A tool for testing CI/CD.

    D. A failed deployment.

    **Answer: B**
    **Explanation:** Integrating tools like SonarQube as a "build breaker" is a key part of CI. It prevents code that fails quality or security gates from being merged.

85. What is a "cron" trigger in CI/CD?

    A. A trigger that runs when a file named `cron` is changed.

    B. A trigger that runs the pipeline on a schedule (e.g., every night at midnight).

    C. A trigger that runs when a cron job on the server fails.

    D. A manual trigger.

    **Answer: B**
    **Explanation:** Most CI/CD tools allow you to use cron syntax to schedule pipelines. This is often used for "nightly builds," "nightly security scans," or "weekly reports."

86. What is "ChatOps"?

    A. A CI/CD tool that runs in a chat room.

    B. The practice of integrating your CI/CD tools (and other ops tools) into a chat client (like Slack or Teams).

    C. A tool for managing team conversations.

    D. A manual deployment.

**Answer: B**
**Explanation:** ChatOps allows you to run commands (e.g., `/deploy my-app to prod`) from a chat window, with the results and notifications being posted back to the channel.

87. In Jenkins, what is a "Freestyle Project"?

    A. The modern, "Pipeline-as-Code" method.

    B. The older, GUI-based method of creating a build job.

    C. A project with no source code.

    D. A project that is not in Git.

    **Answer: B**
    **Explanation:** Before the `Jenkinsfile`, you configured jobs by clicking buttons and filling in text fields in the Jenkins UI. This is a "Freestyle Project."

88. In a `Jenkinsfile`, what is the purpose of the `sh` step?

    A. To execute a shell command (e.g., `sh 'npm install'`).

    B. To checkout source code.

    C. To switch to a different agent.

    D. To send a Slack message.

    **Answer: A**
    **Explanation:** The `sh` step is one of the most common steps. It runs a command in the default shell (usually `bash`) on the agent.

89. What is a "feature flag" (or "feature toggle")?

    A. A Git branch.

    B. A line in a `.gitignore` file.

    C. A configuration (in code or a service) that allows you to turn a feature on or off in production without deploying new code.

    D. A Pull Request.

    **Answer: C**
    **Explanation:** Feature flags are a powerful technique. They allow you to merge and deploy unfinished code (wrapped in an `if (feature_flag_on)` block) safely. This decouples "deployment" from "release."

90. What is "monitoring as code"?

    A. The practice of defining your monitoring dashboards and alerts in code (e.g., in Terraform or YAML).

    B. A tool for monitoring source code.

    C. A CI/CD pipeline for monitoring.

    D. A manual monitoring process.

**Answer: A**
**Explanation:** Similar to IaC, this means your monitoring configuration (dashboards, alerts) is versioned in Git and applied automatically, ensuring consistency.

91. What is the "shifting left" principle in DevSecOps?

    A. Moving all your code to the left side of the screen.

    B. Integrating security (scans, reviews, etc.) earlier in the development lifecycle (i.e., "left" on a timeline).

    C. Only using left-handed developers.

    D. Deploying from a branch on the left.

    **Answer: B**
    **Explanation:** "Shifting left" means moving security from a final "gate" before production to an automated step that happens at commit, build, and test time.

92. What is an "agent pool" in CI/CD?

    A. A single, powerful agent.

    B. A collection of build agents (Runners) with similar capabilities (e.g., "linux-pool", "windows-pool").

    C. The source code for an agent.

    D. A failed agent.

    **Answer: B**
    **Explanation:** An agent pool allows the CI/CD server to distribute jobs to any available agent in that pool, which enables parallel builds and high availability.

93. What is "YAML"?

    A. A programming language.

    B. A human-readable data serialization format, commonly used for configuration files.

    C. A build artifact.

    D. A CI/CD server.

    **Answer: B**
    **Explanation:** YAML (YAML Ain't Markup Language) is the configuration file format for most modern CI/CD tools (GitLab CI, GitHub Actions, CircleCI).

94. What is a "Service Container" (or "sidecar") in a CI/CD job?

    A. The main container running the job.

    B. A second container (e.g., a database like `postgres`) that is linked to the main job container to run integration tests.

    C. A container that has failed.

    D. A Docker image.

**Answer: B**
**Explanation:** If your unit tests need a real database, many CI tools let you define a "service" container. The CI platform starts both your job container and the service container and networks them together.

95. What does `on: pull_request` in a GitHub Actions workflow mean?

    A. The workflow is triggered when a pull request is created or updated.

    B. The workflow will run `git pull`.

    C. The workflow will create a pull request.

    D. The workflow runs only on the `main` branch.

    **Answer: A**
    **Explanation:** This is a very common and powerful trigger. It ensures that every PR is automatically tested *before* it can be merged.

96. What is "Spinnaker"?

    A. A CI (build) tool.

    B. A (CD) Continuous Delivery platform, specializing in complex, multi-cloud deployments (e.g., Blue-Green, Canary).

    C. A source code editor.

    D. A monitoring tool.

    **Answer: B**
    **Explanation:** Spinnaker (originally from Netflix) is an open-source, enterprise-grade CD platform. It's often used *after* a tool like Jenkins builds the artifact.

97. What is "Tekton"?

    A. A CI/CD tool for mobile apps.

    B. A Kubernetes-native framework for creating and running CI/CD pipelines.

    C. A security scanner.

    D. A web-based GUI for Jenkins.

    **Answer: B**
    **Explanation:** Tekton is a powerful, flexible, open-source framework that defines pipelines as Kubernetes Custom Resources (CRDs), making it "cloud-native."

98. What is "Argo CD"?

    A. A CI (build) tool.

    B. A declarative, GitOps-based Continuous Delivery tool for Kubernetes.

    C. A monitoring tool.

    D. An artifact repository.

**Answer: B**
**Explanation:** Argo CD is a popular GitOps tool. It runs in your cluster, watches a Git repo, and automatically "pulls" and applies any changes to manifests, ensuring the cluster matches the repo.

99. What is a "Feedback Loop" in CI/CD?

    A. The time it takes for a developer to get feedback (e.g., test failure, deployment status) after a commit.

    B. A pipeline that runs forever.

    C. A notification from a webhook.

    D. The `post` block in a Jenkinsfile.

    **Answer: A**
    **Explanation:** A primary goal of CI/CD is to *shorten* the feedback loop. Fast builds and tests mean a developer finds out about a bug in minutes, not days.

100. What is "Mean Time to Recovery" (MTTR) a metric for?

    A. Stability

    B. Speed

    C. Security

    D. Cost

    **Answer: A**
    **Explanation:** MTTR is one of the four key DORA (DevOps Research and Assessment) metrics. It measures the stability and resilience of a system by tracking how quickly it can recover from a failure.