CI/CD

---

1. What does CI stand for in DevOps practices?

A. Continuous Inspection
B. Continuous Integration
C. Continuous Implementation
D. Continuous Innovation

Answer: B

Explanation: Continuous Integration emphasizes merging developer changes frequently and validating them automatically to detect issues early.

2. Which statement best describes Continuous Delivery?

A. Automatic deployment to production without human oversight
B. Ability to deploy any build to production on demand
C. Manual deployment once per month
D. Delivery of documentation only

Answer: B

Explanation: Continuous Delivery keeps software in a deployable state so a release can happen at any time, often with a manual approval gate.

3. What is the main goal of Continuous Integration?

A. Merge code changes frequently and run automated tests
B. Deploy nightly builds
C. Monitor servers
D. Manage infrastructure

Answer: A

Explanation: CI reduces integration problems by integrating work regularly and verifying builds with automated tests.

4. Which tool is widely used for CI/CD pipelines and uses declarative Jenkinsfile?

A. Jenkins
B. GitLab Runner
C. Azure DevOps
D. Bamboo

Answer: A

Explanation: Jenkins offers declarative and scripted pipelines defined in Jenkinsfiles, making pipeline logic version-controlled and reproducible.

5. What does a pipeline stage typically represent?

A. A logical grouping of tasks like build, test, deploy

B. Individual command
C. A monolithic application
D. Source repository
Answer: A
Explanation: Stages divide pipeline flow into segments (build, test, deploy), clarifying progression and enabling targeted visibility or approvals.

6. What is the purpose of a build agent?

A. Execute pipeline jobs on assigned infrastructure
B. Manage SCM
C. Replace developers
D. Store artifacts
Answer: A
Explanation: Build agents (runners, executors) run the tasks defined in pipelines, provisioning environments and executing scripts or builds.

7. Which CI/CD practice reduces integration issues by merging changes often?

A. Feature flagging
B. Trunk-based development
C. Blue-green deployment
D. Canary release
Answer: B
Explanation: Trunk-based development encourages short-lived branches and frequent merges into mainline, aligning with CI goals.

8. What is an artifact repository used for?

A. Store build outputs like binaries, packages
B. Store source code
C. Manage environment variables
D. Monitor logs
Answer: A
Explanation: Artifacts are produced by builds and stored in repositories for versioned, traceable distribution to downstream environments.

9. Which tool is commonly used to manage artifacts?

A. JFrog Artifactory
B. Git
C. Terraform
D. Ansible
Answer: A
Explanation: Tools like Artifactory or Nexus manage binary artifacts, complementing source control and enabling reliable deployments.

10. What do automated tests in CI ensure?

A. Code changes don’t break existing functionality
B. Deployments run faster
C. Manual testing is replaced entirely
D. Performance metrics improve

Answer: A

Explanation: Automated tests catch regressions early, maintaining software quality with each integration.

11. In a Jenkins pipeline, what does ‘agent any‘ denote?

A. Run pipeline on any available agent
B. Run pipeline on local machine only
C. Run pipeline in Docker
D. Run pipeline on master node only

Answer: A

Explanation: ‘agent any‘ tells Jenkins to schedule the pipeline on any available executor node with suitable labels.

12. What is a benefit of using Infrastructure as Code in pipelines?

A. Reproducible environments with version control
B. Manual configuration
C. Slower provisioning
D. More documentation

Answer: A

Explanation: IaC ensures environments are defined declaratively and reproducibly, reducing drift and enabling consistent pipeline provisioning.

13. Which branching strategy is often paired with CI?

A. Trunk-based or short-lived feature branches
B. Long-lived release branches only
C. Forking workflow exclusively
D. No branching

Answer: A

Explanation: Short-lived branches or direct commits to trunk integrate quickly, minimizing divergence and easing automated testing.

14. What does pipeline-as-code provide?

A. Version-controlled definition of pipeline logic
B. Manual pipeline creation
C. GUI-based pipeline editing only
D. No change tracking

Answer: A

Explanation: Defining pipelines as code allows review, reuse, and change tracking of automated workflows alongside application code.

15. Which type of testing is typically executed early in CI pipeline?

A. Unit tests
B. Load tests
C. Production tests
D. Chaos engineering

Answer: A

Explanation: Unit tests run quickly and validate individual components, serving as the first safeguard in CI pipelines.

16. What is blue-green deployment?

A. Rolling deployment with two identical environments, switching traffic after verification
B. Deploying with canary nodes only
C. Deploying on weekends
D. Deploying to staging only

Answer: A

Explanation: Blue-green uses two environments (blue and green) where one serves traffic while the other receives updates, enabling instant cutover and rollback.

17. Continuous Deployment differs from Continuous Delivery by:

A. Automatically deploying every change to production without manual approval
B. Requiring manual deployment
C. Removing testing
D. Only building artifacts

Answer: A

Explanation: Continuous Deployment automates the last mile, pushing successful builds to production without human gating, whereas Continuous Delivery may still require approval.

18. Which open-source CI tool uses YAML '.gitlab-ci.yml' file?

A. GitLab CI
B. Jenkins
C. CircleCI
D. Travis CI

Answer: A

Explanation: GitLab CI/CD pipelines are defined in '.gitlab-ci.yml', specifying stages, jobs, and scripts executed by GitLab Runners.

19. What is a canary release?

A. Gradually routing traffic to new version to monitor issues

B. Deploying to all users instantly
C. Manual deployment
D. Deploying without tests
Answer: A
Explanation: Canary releases incrementally introduce changes to a subset of users, monitoring metrics before wider rollout.

20. Why use containerized build environments?

A. Ensure consistent build dependencies
B. Increase resource usage
C. Reduce isolation
D. Avoid automation
Answer: A
Explanation: Containers provide reproducible, isolated environments ensuring builds run with consistent dependencies across agents.

21. What does ‘pipeline‘ block define in Jenkins declarative pipeline?

A. Entire pipeline, including agent, stages, post actions
B. Only build step
C. Only environment variables
D. Only triggers
Answer: A
Explanation: The top-level ‘pipeline‘ block encapsulates the entire pipeline structure, housing agent declarations, environment, stages, and post conditions.

22. Which service from AWS provides managed CI/CD?

A. AWS CodePipeline
B. AWS Lambda
C. AWS CloudTrail
D. Amazon S3
Answer: A
Explanation: CodePipeline orchestrates CI/CD workflows, integrating with CodeBuild, CodeDeploy, and partner services for automation.

23. What is typically stored in ‘.gitlab-ci.yml‘?

A. Stages, jobs, scripts, and environment definitions for pipeline
B. Terraform state
C. Deployment manifests only
D. Helm charts
Answer: A
Explanation: ‘.gitlab-ci.yml‘ describes the pipeline including stages, job definitions, scripts, variables, and environment configuration for GitLab CI/CD.

24. In Azure DevOps pipelines, what is a ‘stage‘?

A. Logical boundary for jobs representing phases like build/test/deploy
B. Container registry
C. Single task
D. Service connection

Answer: A

Explanation: Stages group jobs into phases, enabling approvals, dependencies, and visualization across the pipeline flow.

25. What is the role of webhooks in CI/CD?

A. Trigger pipelines based on repository events like pushes or PRs
B. Manage secrets
C. Provide logging
D. Deploy artifacts

Answer: A

Explanation: Webhooks notify CI/CD systems of repository changes, automatically triggering builds or deployments.

26. Why use feature flags?

A. Toggle features without redeploying, enabling safer releases
B. Enforce longer release cycles
C. Replace version control
D. Force downtime

Answer: A

Explanation: Feature flags enable progressive rollouts and quick disablement of features without redeployment, supporting experimentation.

27. What is artifact promotion?

A. Moving build artifact through environments (dev->test->prod) after validation
B. Advertising product
C. Removing old artifacts
D. Encrypting build

Answer: A

Explanation: Promotion ensures the same artifact travels across environments, maintaining consistency and traceability of deployments.

28. Which pipeline stage assures code quality with static analysis?

A. Quality gate stage
B. Deploy stage
C. Monitoring stage
D. Notification stage

Answer: A

Explanation: Incorporating static analysis or code quality checks in a dedicated stage ensures quality gates are met before progressing.

29. What does ‘retry‘ keyword do in GitLab CI?

A. Re-run job automatically on failure specified number of times
B. Skip stage
C. Rollback release
D. Retry merge request

Answer: A

Explanation: Setting ‘retry‘ allows jobs to retrigger automatically when they fail, mitigating transient issues like network glitches.

30. What is the primary outcome of Continuous Testing?

A. Feedback on code quality delivered rapidly throughout pipeline
B. Manual QA cycles
C. Deployment scheduling
D. Monitoring uptime

Answer: A

Explanation: Continuous Testing ensures test feedback is integrated across the pipeline, enabling quick detection of defects before production.

31. Which protocol is commonly used by CI servers to pull code?

A. Git over HTTPS/SSH
B. FTP
C. SMB
D. RDP

Answer: A

Explanation: CI servers clone or fetch repositories via Git protocols (HTTPS or SSH), aligning with standard version control practices.

32. Why is pipeline visualization helpful?

A. Understand flow, bottlenecks, and failure points quickly
B. Replace logs
C. Increase deploy time
D. Manage secrets

Answer: A

Explanation: Visual pipelines highlight stage status and durations, helping teams identify fail points or slow steps for optimization.

33. What is a manual approval gate?

A. Step requiring human validation before proceeding further
B. Automatic deploy
C. Monitoring tool

D. Notification pipeline

Answer: A

Explanation: Approval gates halt pipeline execution until an authorized person reviews and approves, often before production deploys.

34. Which Kubernetes-native CI/CD tool defines pipelines as CRDs?

A. Tekton

B. Jenkins X (uses Tekton internally)

C. Argo Workflows

D. All of the above

Answer: D

Explanation: Tekton provides CRDs for pipelines; Jenkins X leverages Tekton; Argo Workflows also uses CRDs for workflow definitions.

35. What is the purpose of code coverage reports in CI?

A. Measure percentage of code executed by tests

B. Measure deployment frequency

C. Monitor security

D. Manage repositories

Answer: A

Explanation: Coverage reports reveal how much code the test suite exercises, guiding test improvement.

36. Which metric is part of DORA metrics?

A. Deployment frequency

B. Code review time

C. Sprint velocity

D. Team size

Answer: A

Explanation: DORA metrics include deployment frequency, lead time for changes, change failure rate, and mean time to recovery.

37. What does ‘gitlab-runner register‘ do?

A. Registers a runner with GitLab instance to execute jobs

B. Register user

C. Register branch

D. Register artifact

Answer: A

Explanation: Registering a runner connects it to GitLab, allowing it to pick up CI jobs according to tags or configuration.

38. Why implement secrets management in CI/CD?

A. Protect credentials used during builds and deployments

B. Increase logs
C. Share passwords publicly
D. Reduce automation
Answer: A
Explanation: Secure secrets management prevents exposure of credentials, API keys, and tokens used in automation.

39. What is pipeline caching?

A. Reusing previously downloaded dependencies or build outputs to speed jobs
B. Storing pipeline logs
C. Storing artifacts
D. Storing environment variables
Answer: A
Explanation: Caching reduces redundant downloads and builds, improving pipeline performance by reusing intermediate results.

40. What is the role of ‘stages‘ array in GitLab CI?

A. Define execution order of job groups
B. Define environment variables
C. Store artifacts
D. Manage secrets
Answer: A
Explanation: The ‘stages‘ array sets the pipeline flow order; jobs assigned to each stage run concurrently and stages run sequentially.

41. Which CI/CD concept ensures production-like environment for testing?

A. Staging environment
B. Development environment
C. Sandbox only
D. Local environment
Answer: A
Explanation: Staging mirrors production to catch issues before release, providing realistic validation of builds.

42. What is rollback strategy?

A. Plan to revert to previous stable version if deployment fails
B. Plan to accelerate deployments
C. Plan to refactor
D. Plan to scale
Answer: A
Explanation: Rollback plans ensure teams can quickly return to a known good state when new releases cause issues.

43. In Jenkins, what does a ‘post‘ block define?

A. Actions after entire pipeline or stage completes (success/failure/always)
B. Pre-build tasks
C. Environment variables
D. Agent selection

Answer: A

Explanation: The ‘post‘ section declares steps to run after pipeline completion, such as notifications on success or failure.

44. Which service provides hosted CI/CD pipelines integrated with GitHub?

A. GitHub Actions
B. Bitbucket Pipelines
C. Azure DevOps
D. TeamCity

Answer: A

Explanation: GitHub Actions runs workflows directly in GitHub repositories, integrating with GitHub events and marketplace actions.

45. What is the purpose of ‘matrix‘ builds?

A. Run same job across multiple configurations (e.g., OS, versions)
B. Run sequential builds
C. Upgrade matrix library
D. Manage dependencies

Answer: A

Explanation: Matrix builds execute permutations of parameters (like Python versions) in parallel, improving test coverage across environments.

46. Why is idempotent deployment important?

A. Ensures repeated deployments yield same state without side effects
B. Speeds up compile
C. Removes rollback need
D. Eliminates testing

Answer: A

Explanation: Idempotent deployments avoid inconsistent environments when pipelines rerun the same deployment script.

47. What does ‘needs‘ keyword define in GitLab CI?

A. Job dependencies to enable directed acyclic graph execution
B. Required artifacts
C. Required manual approval
D. Required variables

Answer: A

Explanation: ‘needs‘ allows jobs to run as soon as prerequisites succeed, enabling more parallel execution than stage-based dependencies alone.

48. Which technique gradually shifts traffic between old and new versions?

A. Canary deployment
B. Blue-green
C. Rolling update
D. Both A and C (canary incremental, rolling sequential)

Answer: D

Explanation: Canary gradually increases traffic percentage; rolling updates replace instances in batches. Both provide controlled transitions.

49. What is artifact retention policy?

A. Rules for how long artifacts are stored before cleanup
B. Rule for artifact security
C. Rule for pipeline triggers
D. Rule for environment

Answer: A

Explanation: Retention policies determine storage duration for build outputs to manage storage costs and compliance requirements.

50. What does ‘pipeline.trigger‘ do in GitLab?

A. Starts downstream pipeline using trigger token
B. Adds stage
C. Cancels pipeline
D. Notifies user

Answer: A

Explanation: Pipeline triggers in GitLab allow programmatic or chained pipeline execution via tokens, enabling multi-project orchestration.

51. Why use container registries in CI/CD?

A. Store and version Docker images used for deployment
B. Store source code
C. Store binaries only
D. Manage logs

Answer: A

Explanation: Container registries hold built images tagged with versions, ensuring consistent artifacts for deployments across environments.

52. Which command validates GitHub Actions workflow syntax locally?

A. act (third-party tool)
B. gh workflow validate
C. Both (depending on tooling)

D. git actions check

Answer: C

Explanation: The ‘act‘ CLI simulates Actions locally, while ‘gh workflow validate‘ checks workflow syntax using GitHub’s CLI.

53. What is the purpose of ‘approval‘ steps in deployment pipelines?

A. Provide manual control gate before promoting to critical environments

B. Automate deployments

C. Remove accountability

D. Manage secrets

Answer: A

Explanation: Approvals ensure stakeholders validate changes before production deployment, balancing automation with governance.

54. Which object defines pipeline default environment variables in GitLab?

A. variables:

B. env:

C. settings:

D. config:

Answer: A

Explanation: In ‘.gitlab-ci.yml‘, the ‘variables‘ section sets default environment variables available to all jobs unless overridden.

55. What is the benefit of automated rollbacks?

A. Rapid recovery from failed deployments with minimal manual intervention

B. Slower pipeline

C. Increased downtime

D. More manual work

Answer: A

Explanation: Automation enables quick rollback to previous stable versions, reducing mean time to recovery when incidents occur.

56. What does ‘pipeline as code‘ enable for auditing?

A. Track changes to pipeline definitions via version control

B. Remove history

C. Mask commits

D. Delete logs

Answer: A

Explanation: Pipelines stored alongside source code provide traceability, code review, and history for workflow changes.

57. Which Jenkins plugin supports declarative pipelines?

A. Pipeline plugin

B. Blue Ocean
C. Git plugin
D. Email-ext
Answer: A
Explanation: The Pipeline plugin adds Jenkinsfile-based pipeline support, enabling declarative syntax and pipeline-as-code capabilities.

58. What is ephemeral build agent?

A. Agent created on demand and destroyed after job completion
B. Always-on server
C. Manual worker
D. Static host
Answer: A
Explanation: Ephemeral agents provision clean environments per job, preventing contamination between builds and simplifying scaling.

59. Why configure branch protection rules with CI?

A. Enforce successful builds/tests before merges
B. Prevent branching
C. Disable CI
D. Force direct commits to main
Answer: A
Explanation: Branch protection can require status checks from CI pipelines to pass before merging to critical branches, improving quality.

60. Which pipeline stage ensures compliance checks?

A. Governance stage
B. Build stage
C. Rollback stage
D. Code review stage
Answer: A
Explanation: A governance or compliance stage runs security checks, policy validations, or audit tasks prior to release.

61. What is trunk-based development?

A. Developers commit small changes frequently to mainline
B. Developers work on long branches for months
C. Developers clone nightly
D. Developers avoid merging
Answer: A
Explanation: Trunk-based development supports CI by encouraging frequent merges to main, reducing merge conflicts and enabling rapid feedback.

62. Why integrate security scans (DevSecOps)?

A. Identify vulnerabilities early in pipeline
B. Delay deployments
C. Replace QA
D. Remove tests

Answer: A

Explanation: Integrating security testing (SAST, DAST, container scanning) into pipelines catches issues early, aligning with DevSecOps practices.

63. What does ‘parallel‘ directive enable in Jenkins?

A. Execute multiple branches of pipeline simultaneously
B. Repeat stage sequentially
C. Execute only one job
D. Reuse artifacts

Answer: A

Explanation: Jenkins declarative pipeline supports ‘parallel‘ to run multiple branches (e.g., tests on different platforms) concurrently within a stage.

64. Which metric measures time between code committed and deployed?

A. Lead time for changes
B. Mean time to recover
C. Deployment frequency
D. Change fail rate

Answer: A

Explanation: Lead time for changes, one of the DORA metrics, tracks how quickly changes move from commit to production.

65. What is a service connection in Azure DevOps?

A. Defines credentials to external services used by pipelines
B. Defines deployment environment
C. Defines pipeline schedule
D. Defines repository

Answer: A

Explanation: Service connections store authentication details for services like Azure subscriptions, Docker registries, or GitHub repositories.

66. How do you trigger GitHub Actions workflow manually?

A. workflow_dispatch event
B. manual trigger command
C. scheduler
D. comment trigger only

Answer: A

Explanation: Adding ‘workflow_dispatch‘ allows manual triggering from the GitHub UI or API with optional input parameters.

67. Why maintain pipeline documentation?

A. Help teams understand stages, dependencies, and recovery procedures
B. Replace code
C. Increase approvals
D. Remove automation

Answer: A

Explanation: Documentation clarifies pipeline design, responsibilities, and rollback steps, facilitating onboarding and incident response.

68. What does ‘allow_failure: true‘ do in GitLab CI?

A. Marks job as optional; pipeline succeeds even if job fails
B. Retries job
C. Cancels pipeline
D. Skips job

Answer: A

Explanation: Setting ‘allow_failure‘ permits a job to fail without failing the pipeline, useful for non-critical tests or experimental checks.

69. Which deployment strategy keeps old version running until new version passes tests, then switches traffic?

A. Blue-green deployment
B. Rolling deployment
C. Direct cutover
D. Hotfix deployment

Answer: A

Explanation: Blue-green maintains two production environments, promoting the new one only after validation, facilitating instant rollback.

70. What is pipeline orchestration?

A. Coordinated execution of multiple pipelines or jobs with dependencies
B. Running single script
C. Manual approval only
D. Logging

Answer: A

Explanation: Orchestration manages complex workflows across projects or microservices, ensuring correct order and dependencies among pipelines.

71. What is the role of ‘artifacts‘ keyword in GitLab CI?

A. Specify files to be uploaded after job completion

B. Define dependencies
C. Set environment
D. Set timeouts
Answer: A
Explanation: Declaring ‘artifacts‘ captures files (binaries, logs, reports) from a job and persists them for download or downstream stages.

72. Why use caching of Docker layers?

A. Accelerate image builds by reusing unchanged layers
B. Increase image size
C. Remove reproducibility
D. Avoid build automation
Answer: A
Explanation: Docker layer caching prevents rebuilding unchanged layers, significantly reducing build time during CI.

73. Which tool automates progressive delivery with metrics-based rollouts on Kubernetes?

A. Argo Rollouts
B. Spinnaker
C. Flagger
D. All of the above
Answer: D
Explanation: Argo Rollouts, Spinnaker, and Flagger all support progressive delivery patterns (canary, blue-green) driven by metrics and analysis.

74. What is dynamic environment in GitLab?

A. Temporary environment created per branch or merge request for review
B. Static production environment
C. Local workspace
D. Pipeline log
Answer: A
Explanation: Dynamic environments provide per-branch review apps, enabling testing and stakeholder review before merge.

75. Why incorporate smoke tests post-deployment?

A. Quickly verify critical functionality before full rollout
B. Replace unit tests
C. Delay release
D. Provide documentation
Answer: A
Explanation: Smoke tests validate essential system functions immediately after deployment, catching major issues early.

76. What is pipeline failure notification used for?

A. Alert stakeholders about issues for quick remediation
B. Increase noise
C. Remove logs
D. Document success

Answer: A

Explanation: Notifications via email, chat, or incident systems ensure the right people respond promptly to broken pipelines.

77. Which command line tool interacts with Jenkins via scripts?

A. Jenkins CLI (java -jar jenkins-cli.jar)
B. Jenkinsctl
C. jnk
D. Jenkins API only

Answer: A

Explanation: The Jenkins CLI jar connects to Jenkins to create jobs, trigger builds, or manage configuration from scripts.

78. Why use multi-branch pipelines?

A. Automatically create pipeline per branch with shared Jenkinsfile
B. Only run on main
C. Avoid branching
D. Merge branches

Answer: A

Explanation: Multi-branch pipelines auto-detect branches and run their respective Jenkinsfiles, aligning CI with branch workflows.

79. What is the purpose of pipeline timeouts?

A. Prevent jobs from running indefinitely if stuck
B. Slow down pipeline
C. Ensure manual intervention
D. Increase resource usage

Answer: A

Explanation: Timeouts guard against hung builds, freeing resources and prompting investigation of long-running steps.

80. Which GitHub Actions syntax defines job dependencies?

A. needs:
B. depends_on:
C. requires:
D. after:

Answer: A

Explanation: The ‘needs‘ keyword lists jobs that must complete before the current job begins, establishing dependency graphs in workflows.

81. Why run security scans on container images?

A. Detect vulnerabilities before deployment
B. Increase image size
C. Remove dependencies
D. Delay releases

Answer: A

Explanation: Scanning images for CVEs ensures containerized workloads meet security standards before reaching production.

82. What is a release pipeline?

A. Pipeline focusing on deploying artifacts through staging/production with approvals
B. Pipeline building code only
C. Pipeline for documentation
D. Pipeline for tests only

Answer: A

Explanation: Release pipelines orchestrate packaging, approvals, and deployments across environments, managing releases end-to-end.

83. What is the role of configuration management in CD?

A. Ensure environments are configured consistently during deployments
B. Replace CI
C. Manage developers
D. Provide logging

Answer: A

Explanation: Configuration management tools (Ansible, Chef, Puppet) enforce consistent server state, supporting automated deployments.

84. Why use automated database migrations in CD?

A. Keep schema in sync with application deployments reliably
B. Replace backups
C. Avoid schema changes
D. Delay deploys

Answer: A

Explanation: Automated migrations ensure database changes deploy alongside application code, reducing manual errors and drift.

85. What does ‘CI‘ pipeline badge indicate in GitHub/GitLab?

A. Current build status (passing/failing)
B. Number of commits
C. Number of contributors

D. Tag count

Answer: A

Explanation: Pipeline badges display build status, allowing quick visibility on README pages or dashboards.

86. Which pipeline step prepares infrastructure before deploy?

A. Provision stage using IaC

B. Cleanup stage

C. Notification stage

D. Monitoring stage

Answer: A

Explanation: Provisioning stage applies infrastructure templates to ensure required resources exist before application deployment.

87. What is environment drift?

A. Differences between environments caused by manual changes

B. Monitoring metric

C. Pipeline failure

D. Deployment frequency

Answer: A

Explanation: Drift occurs when environments diverge from desired configuration due to manual edits or inconsistent automation; pipelines aim to prevent or detect it.

88. Why monitor pipeline performance metrics?

A. Identify bottlenecks and improve speed/reliability

B. Increase costs

C. Reduce automation

D. Replace documentation

Answer: A

Explanation: Tracking job durations, queue times, and success rates helps optimize pipeline efficiency and reliability.

89. What does ‘rules‘ keyword in GitLab CI define?

A. Conditions for job execution based on branch, variables, or changes

B. Job dependencies

C. Artifact storage

D. Timeout

Answer: A

Explanation: ‘rules‘ provides fine-grained control over job execution, replacing ‘only‘/‘except‘ with more flexible conditions.

90. Which strategy avoids downtime by updating subsets of servers sequentially?

A. Rolling updates

B. Blue-green
C. Big bang
D. Hotfix
Answer: A
Explanation: Rolling updates update a portion of instances at a time, maintaining service availability during deployment.

91. Why include performance tests in CD pipeline?

A. Ensure releases meet performance SLAs before production
B. Replace unit tests
C. Delay features
D. Increase manual work
Answer: A
Explanation: Automated performance tests verify that changes don't degrade responsiveness or capacity, protecting user experience.

92. What does 'only: changes' do in GitLab CI?

A. Runs job only when specified files change
B. Runs job only on main
C. Runs job only on tags
D. Runs job only on schedule
Answer: A
Explanation: The 'changes' rule triggers jobs based on file patterns, limiting work to relevant commits (e.g., running docs build when docs change).

93. Which tool visualizes pipeline metrics and DORA scores?

A. GitLab Analytics
B. Jenkins Dashboard
C. Azure DevOps Insights
D. All (A and C have features; Jenkins via plugins)
Answer: D
Explanation: GitLab and Azure DevOps provide built-in analytics; Jenkins offers plugins for visualizing trends, enabling teams to track DORA metrics.

94. What is deployment window?

A. Predefined time when deployment is allowed to minimize risk
B. CI stage
C. Monitoring interval
D. Build slot
Answer: A
Explanation: Deployment windows schedule releases during low-risk periods, coordinating teams and minimizing user impact.

95. Why use automated secrets rotation?

A. Reduce exposure by regularly updating credentials used in pipelines
B. Increase manual updates
C. Avoid automation
D. Keep secrets static

Answer: A

Explanation: Rotating secrets automatically limits risk of compromise, ensuring credentials used by pipelines remain secure.

96. What is ChatOps in CI/CD context?

A. Triggering and monitoring pipeline activities through chat platforms
B. Chatting with support
C. Creating chat rooms
D. Using chat for deployment approvals only

Answer: A

Explanation: ChatOps integrates automation with chat tools (Slack, Teams) to run commands, view build status, and collaborate in real time.

97. Why implement pipeline linting tools?

A. Validate syntax and best practices before pipeline execution
B. Increase runtime
C. Break builds
D. Remove automation

Answer: A

Explanation: Linting ensures pipeline definitions adhere to syntax and style rules, catching issues before pipelines fail at runtime.

98. Which deployment pattern keeps multiple versions active and routes specific users to each?

A. A/B testing
B. Rolling
C. Blue-green
D. Hotfix

Answer: A

Explanation: A/B testing runs two versions simultaneously, directing user segments to measure performance or user response.

99. What is mean time to recovery (MTTR)?

A. Average time to restore service after failure
B. Time to deploy
C. Time to test
D. Time to review code

Answer: A

Explanation: MTTR, a DORA metric, measures how quickly teams recover from incidents, indicating resilience of delivery process.

100. Why maintain pipeline templates?

A. Encourage consistency and reuse across teams/projects

B. Increase duplication

C. Restrict customization

D. Avoid automation

Answer: A

Explanation: Templates accelerate new pipeline creation with proven patterns, ensuring consistent practices and reducing duplication.