

Gitlab GitLab-Certified-CI-CD-Associate

GitLab Certified CI/CD Associate

**Thank You for Downloading GitLab-Certified-CI-CD-Associate
Updated Exam Questions**

<https://certs4sale.com/Gitlab/gitlab-certified-ci-cd-associate-pdf-exam-dumps>

QUESTION: 1

A DevOps engineer is configuring a GitLab CI/CD pipeline with multiple jobs. They need to ensure that certain jobs execute before others while optimizing the pipeline's efficiency. Which of the following best describes how GitLab determines the order of job execution in a pipeline?

- Option A : Jobs execute sequentially from top to bottom as they appear in .gitlab-ci.yml.
- Option B : Jobs are executed in parallel by default, unless defined in separate stages.
- Option C : The needs keyword forces all jobs to run in strict sequential order, one after another.
- Option D : Jobs in different stages execute randomly, without any predefined sequence.

Correct Answer: B

QUESTION: 2

In GitLab CI/CD, different types of executors can be used to run jobs in different environments. Which of the following is an uncommon executor for running CI/CD jobs in GitLab?

- Option A : Shell Executor
- Option B : Kubernetes Executor
- Option C : Docker Executor
- Option D : Azure Pipelines Executor

Correct Answer: D

QUESTION: 3

You are managing a project with multiple environments (development, staging, production) in GitLab CI/CD. You need to ensure that any changes pushed to the development branch trigger deployments to the development environment, changes to the staging branch trigger deployments to the staging environment, and only approved changes can be deployed to production from the main branch. Which two .gitlab-ci.yml configurations should you use to achieve this?

- Option A : Set up manual deployment jobs for all environments and trigger them manually.
- Option B : Use the rules keyword with conditions for each environment-specific job.
- Option C : Define a single pipeline for all environments and use environment variables to switch between environments.
- Option D : Define separate jobs for each environment with the only keyword specifying the respective branch.
- Option E : Combine all deployment jobs into a single job and use the script section to handle branching logic.

Correct Answer: B,D

QUESTION: 4

You want to integrate API Security Scanning into your GitLab CI/CD pipeline to automatically detect vulnerabilities in your APIs. What steps should you take to properly configure API Security Scanning?

Option A : Manually review API endpoints and test them for vulnerabilities in the pipeline configuration.

Option B : Add an `api_security_scan` job in the `.gitlab-ci.yml` file and include `security/api-scanning` template.

Option C : Write a custom script to perform API security scans and integrate it into the build stage of the pipeline.

Option D : Install a third-party API security scanning tool on your CI runners and configure it to run during the pipeline.

Correct Answer: B

QUESTION: 5

You have multiple GitLab Runners in your project, each configured to run specific types of jobs. You want to ensure that your Docker-based CI/CD jobs run only on runners that have been explicitly configured for Docker. Which of the following is the correct way to ensure this behavior in your `.gitlab-ci.yml` file?

Option A :

```
1. job:
2.   script: echo "Running job on Docker runner"
3.   assign-runner: docker-runner
```

Option B :

```
1. job:
2.   script: echo "Running job on Docker runner"
3.   executor: docker
```

Option C :

```
1. job:
2.   script: echo "Running job on Docker runner"
3.   runner: docker-runner
```

Option D :

```
1. job:
2.   script: echo "Running job on Docker runner"
3.   tags:
4.     - docker-runner
```

Correct Answer: D

QUESTION: 6

In a GitLab CI/CD pipeline, you need to execute a script that requires specific environment variables to be set dynamically based on the branch name. How can you configure the job to set these variables?

- Option A : Use the `tags` keyword to specify different runners that set different environment variables.
- Option B : Use the `before_script` section to write a script that sets the environment variables dynamically.
- Option C : Set the environment variables in the job script section.
- Option D : Define the environment variables in the `.gitlab-ci.yml` file under the `variables` section.

Correct Answer: B

QUESTION: 7

A DevOps engineer is setting up a GitLab Child Pipeline to dynamically trigger jobs based on changes in different components of their repository. Which of the following is required to correctly configure a child pipeline?

- Option A : A child pipeline must be defined in the same `.gitlab-ci.yml` file as the parent pipeline.
- Option B : Use the `trigger` keyword in a job configuration to specify the child pipeline's configuration file.
- Option C : Define a job in `.gitlab-ci.yml` that uses `extends: child-pipeline`.
- Option D : Add `child-pipeline: true` as a global setting in the GitLab project's CI/CD settings.

Correct Answer: B

QUESTION: 8

You want to integrate Dependency Scanning into your GitLab CI/CD pipeline to automatically detect vulnerabilities in your project's dependencies. What steps should you take to properly configure Dependency Scanning?

- Option A : Write a custom script to parse dependencies and check for vulnerabilities during the build stage.
- Option B : Configure a webhook to trigger a dependency scan after every commit.
- Option C : Add a `dependency_scan` job in the `.gitlab-ci.yml` file and include `security/dependency-scanning` template.
- Option D : Install a third-party tool on your CI runners and manually configure it to scan dependencies.

Correct Answer: C

QUESTION: 9

You want to integrate Coverage-guided Fuzz Testing into your GitLab CI/CD pipeline to identify security vulnerabilities. What steps should you take to properly configure Coverage-guided Fuzz Testing?

- Option A : Add a `fuzz_test` job in the `.gitlab-ci.yml` file and include `security/fuzz-testing` template.
- Option B : Use the `before_script` section to run fuzz tests before any other job in the pipeline.
- Option C : Install a third-party fuzz testing tool on your CI runners and configure it to run during the pipeline.
- Option D : Write a custom script to perform fuzz testing and integrate it into the build stage of the pipeline.

Correct Answer: A

QUESTION: 10

A software engineering team wants to speed up their CI/CD pipeline execution by reducing redundant computations in GitLab CI/CD. They decide to use caching to store dependencies between runs. Which of the following is true about caching in GitLab CI/CD?

- Option A : Cached files are automatically synchronized across different runner machines in a multi-runner setup.
- Option B : GitLab automatically caches all files used in previous jobs unless explicitly disabled.
- Option C : A cache is always removed after a job completes, and it cannot be retained for future jobs.
- Option D : The cache keyword allows caching of files or directories between pipeline jobs and can be shared across different pipelines.

Correct Answer: D

QUESTION: 11

What is the primary goal of implementing CI in a software development workflow?

- Option A : To ensure that all code changes are deployed to production immediately.
- Option B : To replace manual testing with automated production monitoring.
- Option C : To automate the process of creating virtual machines for testing purposes.
- Option D : To reduce the time it takes to merge code changes into the main branch.

Correct Answer: D

QUESTION: 12

A DevOps engineer is setting up a GitLab CI/CD pipeline to deploy applications to different environments:

staging and production. The goal is to ensure that deployments are tracked and associated with the correct environment in GitLab. How should they configure the deployment in `.gitlab-ci.yml`?

- Option A : Use when: manual for the staging environment to make sure it always deploys automatically.
- Option B : Define environment globally in `.gitlab-ci.yml` to apply the same setting to all jobs.
- Option C : Use environment: name: production in all jobs so that every job is associated with the production environment.
- Option D : Use environment: name: staging for the staging job and environment: name: production for the production job

Correct Answer: D

QUESTION: 13

Your GitLab CI/CD pipeline is set up for a large application with multiple stages and jobs. Recently, you noticed that a specific job in the middle of the pipeline is failing, causing the subsequent jobs to be skipped. You want to identify the root cause efficiently using the GitLab Pipeline Graph. Which of the following steps should you take to pinpoint the issue?

- Option A : Rerun the entire pipeline from the beginning.
- Option B : Use the pipeline graph to identify the failing job and inspect its logs.
- Option C : Manually execute each job on a local machine to find the error.
- Option D : Check the log files of the skipped jobs.

Correct Answer: B

QUESTION: 14

You are managing a CI/CD pipeline in GitLab that needs to deploy to different environments based on the branch being merged (e.g., develop branch to staging, main branch to production). How would you configure the pipeline to ensure the correct deployment for each branch?

- Option A : Use except keyword to exclude branches from job execution.
- Option B : Use the only keyword in the job definitions to specify the branches they should run on.
- Option C : Use a single job with environment variables to switch between deployment environments.
- Option D : Configure separate `.gitlab-ci.yml` files for each environment.

Correct Answer: B

QUESTION: 15

You need to optimize a GitLab CI/CD pipeline to include caching to speed up subsequent pipeline runs, and also ensure that artifacts from the build stage are available to the deployment stage. Which two configurations should you use in the `.gitlab-ci.yml` file?

- Option A : Define artifacts in the build stage and specify the paths and when to be used.
- Option B : Use the dependencies keyword to define job dependencies.
- Option C : Set up a before_script section to manually handle caching and artifact creation.
- Option D : Use the cache keyword to define paths to be cached between jobs.
- Option E : Combine all stages into a single job to avoid the need for caching and artifact sharing.

Correct Answer: A,D

QUESTION: 16

A team wants to configure GitLab Group Runners for multiple projects but needs to ensure that only specific types of jobs run on them while other jobs should execute on shared GitLab-hosted runners. What is the best way to achieve this?

- Option A : Configure job-level tags in .gitlab-ci.yml to match the group runner's tags and define specific jobs to use it.
- Option B : Assign runner tags at the instance level and let projects override them in .gitlab-ci.yml.
- Option C : Configure the group runner as locked so that only default pipeline jobs can use it.
- Option D : Disable shared runners at the group level to force all jobs to run on the group runner.

Correct Answer: A

QUESTION: 17

A DevOps engineer wants to optimize a GitLab CI/CD pipeline using Directed Acyclic Graph (DAG) execution to allow jobs in different stages to run in parallel when there are no explicit dependencies. The engineer writes the following .gitlab-ci.yml configuration:

```
1. stages:
2.   - build
3.   - test
4.   - deploy
5.
6. job1:
7.   stage: build
8.   script: echo "Building project"
9.
10. job2:
11.   stage: test
12.   script: echo "Running tests"
13.
14. job3:
15.   stage: deploy
16.   script: echo "Deploying"
```

Which statement best describes how this pipeline will execute?

- Option A : The pipeline will allow jobs to run in parallel within the same stage but still enforce stage ordering.
- Option B : The pipeline will execute all jobs simultaneously because there are no explicit dependencies.
- Option C : The pipeline will use a DAG by default and optimize execution based on detected dependencies.
- Option D : The pipeline will execute all jobs in a strictly sequential manner based on the stages definition.

Correct Answer: A

QUESTION: 18

You are working on a project that requires deploying a web application to a Kubernetes cluster. The deployment process should include running unit tests, building a Docker image, pushing the image to a container registry, and then deploying it to the Kubernetes cluster. Which of the following CI/CD configurations would best achieve this in GitLab?

- Option A : Use a single pipeline with stages for testing, building, and deploying, utilizing the Docker and Kubernetes executors.
- Option B : Use separate pipelines for testing and deployment to isolate the deployment process from the main CI/CD pipeline.
- Option C : Use a single pipeline with a manual job for deployment to ensure it is only deployed when manually triggered.
- Option D : Use GitLab Pages to deploy the web application to a Kubernetes cluster after building and testing.

Correct Answer: A

QUESTION: 19

You are managing a CI/CD pipeline in GitLab for a large application with multiple services. You notice that the pipeline is taking too long to complete due to numerous sequential jobs. How can you optimize the pipeline to reduce the total execution time?

- Option A : Use pipeline caching.
- Option B : Disable artifacts.
- Option C : Increase the number of sequential jobs.
- Option D : Implement parallel jobs.

Correct Answer: D

QUESTION: 20

You need to create a GitLab CI/CD pipeline with the following requirements: The build stage compiles the code. The test stage runs unit tests. The integration stage runs integration tests but should only run if the test

stage is successful. The deploy stage deploys the application but should only run if both the test and integration stages are successful and only on the production branch. Which of the following .gitlab-ci.yml configurations correctly implements this scenario?

Option A :

```
1. stages:
2.   - build
3.   - test
4.   - integration
5.   - deploy
6.
7. test:
8.   stage: test
9.   script:
10.    - run_tests
11.
12. integration:
13.   stage: integration
14.   script:
15.    - run_integration_tests
16.   needs:
17.    - test
18.
19. deploy:
20.   stage: deploy
21.   script:
22.    - deploy_application
23.   only:
24.    - production
25.   needs:
26.    - integration
```

Option B :

```
1. stages:
2.   - build
3.   - test
4.   - integration
5.   - deploy
6.
7. test:
8.   stage: test
9.   script:
10.    - run_tests
11.
12. integration:
13.   stage: integration
14.   script:
15.    - run_integration_tests
16.   only:
```

```
17.     - production
18.
19.  deploy:
20.    stage: deploy
21.    script:
22.      - deploy_application
23.  only:
24.    - production
```

Option C :

```
1.  stages:
2.    - build
3.    - test
4.    - integration
5.    - deploy
6.
7.  integration:
8.    stage: integration
9.    script:
10.      - run_integration_tests
11.
12.  deploy:
13.    stage: deploy
14.    script:
15.      - deploy_application
16.  only:
17.    - production
```

```
1. stages:
2.   - build
3.   - test
4.   - integration
5.   - deploy
6.
7. test:
8.   stage: test
9.   script:
10.    - run_tests
11.
12. integration:
13.   stage: integration
14.   script:
15.    - run_integration_tests
16.
17. deploy:
18.   stage: deploy
19.   script:
20.    - deploy_application
21.   only:
22.    - production
23.   dependencies:
24.    - integration
```

Option D :

Correct Answer: A

QUESTION: 21

You are setting up a GitLab CI/CD pipeline that needs to execute specific jobs only when certain conditions are met, such as changes to a particular file or directory. Which GitLab CI/CD feature allows you to define these conditions within your pipeline?

- Option A : Rules
- Option B : Job Dependencies
- Option C : Only/Except Keywords
- Option D : CI/CD Templates

Correct Answer: A

QUESTION: 22

In GitLab CI/CD, which keyword is used in the .gitlab-ci.yml file to define different environments such as development, staging, and production?

- Option A : environment
- Option B : jobs
- Option C : deploy

Option D : stages

Correct Answer: A

QUESTION: 23

Your team is working on a GitLab project that handles sensitive user data. To prevent accidental exposure of secrets like API keys and passwords, you need to enable Secret Detection in your CI/CD pipeline. Which of the following steps is NOT required to set up Secret Detection in GitLab?

Option A : Review and act on the findings from the Secret Detection reports.

Option B : Add the Secret Detection template to your .gitlab-ci.yml file.

Option C : Configure a secret scanning tool outside of GitLab to run alongside the pipeline.

Option D : Ensure that GitLab Runners have access to the necessary Docker images for Secret Detection.

Correct Answer: C

QUESTION: 24

How can a downstream pipeline in another project be triggered in a Multi-Project Pipeline configuration?

Option A : By using the `include` keyword with a reference to the other project's YAML file

Option B : By using the `trigger` keyword with a `project` parameter

Option C : By specifying a `multi-project` keyword at the top level of the pipeline configuration

Option D : By setting the `parallel` keyword in the pipeline definition

Correct Answer: B

QUESTION: 25

You are working on a project in GitLab that involves multiple environments (development, staging, production). Each environment requires different configurations and credentials. Recently, you've observed that the deployment to the production environment sometimes fails due to incorrect configurations being used. Which approach is best to ensure that each environment gets the correct configurations and credentials during deployment?

Option A : Manually change the configurations and credentials in the code before each deployment.

Option B : Store all configurations and credentials directly in the .gitlab-ci.yml file.

Option C : Use GitLab environment variables to manage different configurations and credentials.

Option D : Deploy only to the development environment and manually move the code to staging and production.

Correct Answer: C

QUESTION: 26

You have a CI/CD pipeline in GitLab with several jobs. One of your jobs, `critical_task`, must run and complete successfully before any other jobs in its stage can execute. You have configured the `.gitlab-ci.yml` as follows: `stages: - setup - test` `critical_task: stage: setup script: - echo "Critical Task Running"` `secondary_task: stage: setup script: - echo "Secondary Task Running"` `needs: - critical_task` `unit_tests: stage: test script: - echo "Running unit tests"`

- Option A : `secondary_task` will run only after `critical_task` completes successfully.
- Option B : `unit_tests` will run regardless of the outcome of `critical_task`.
- Option C : `unit_tests` will run only if both `critical_task` and `secondary_task` succeed.
- Option D : The `needs` keyword ensures `critical_task` and `secondary_task` run sequentially.
- Option E : If `critical_task` fails, `secondary_task` will not run.

Correct Answer: A,D,E

QUESTION: 27

What is the primary benefit of integrating coverage-guided fuzz testing into your GitLab CI/CD pipeline?

- Option A : It identifies potential security vulnerabilities by generating unexpected input data and observing the program's behavior.
- Option B : It automatically documents all discovered vulnerabilities and suggests fixes.
- Option C : It provides real-time security monitoring of the production environment.
- Option D : It ensures that all dependencies are secure and up-to-date.

Correct Answer: A

QUESTION: 28

You manage a GitLab CI/CD setup where different jobs have varying resource requirements. How should you configure your GitLab CI/CD pipeline to ensure that each job is assigned to the most appropriate runner?

- Option A : Use shared runners for all jobs regardless of their resource requirements
- Option B : Use a single powerful runner for all jobs
- Option C : Allow GitLab to automatically assign jobs to any available runner
- Option D : Define runner tags and use them to assign specific jobs to appropriate runners

Correct Answer: D

QUESTION: 29

A GitLab CI/CD pipeline is taking too long to execute because all jobs run sequentially. The DevOps team wants to optimize it by running independent jobs in parallel while maintaining proper execution order for dependent jobs. Which approach should they take?

- Option A : Remove stages and rely on job naming conventions to determine execution order.
- Option B : Use when: always for all jobs so they execute immediately when a commit is pushed.
- Option C : Use stages: [build, test, deploy] and configure jobs within the same stage to run in parallel.
- Option D : Set needs: [test] in the deploy job to force deployment to happen before testing.

Correct Answer: C

QUESTION: 30

Your team is developing a large-scale application with multiple dependencies and frequent code changes. The CI/CD pipeline is configured to run tests and build the application upon each commit. However, you notice that the pipeline often fails due to intermittent network issues when downloading dependencies, causing delays. What is the best approach to improve the reliability and speed of the pipeline?

- Option A : Run the pipeline less frequently to reduce the impact of network issues.
- Option B : Decrease the number of dependencies to minimize network usage.
- Option C : Increase the timeout settings for network operations in the pipeline.
- Option D : Use a local cache to store dependencies and reuse them across pipeline runs.

Correct Answer: D

THANK YOU FOR DOWNLOADING GITLAB-CERTIFIED-CI-CD-ASSOCIATE UPDATED EXAM QUESTIONS

Note: Thanks For Trying The Demo Of Our GitLab-Certified-CI-CD-Associate Exam Product

Visit Our Site to Purchase the Full Set of Actual GitLab-Certified-CI-CD-Associate Exam Questions With Answers.

Money Back Guarantee



Click The Link Below

<https://certs4sale.com/Gitlab/gitlab-certified-ci-cd-associate-pdf-exam-dumps>