CERTLABZ CERTIFICATION BOARD

Certified Infrastructure as Code Professional (ClaCP)

Ten Sample Questions

These ten questions are written at the difficulty of the live exam. Six carry a figure, a data table, a configuration or a code listing; four are plain scenario questions, as on the live paper. Every one is a scenario a working infrastructure engineer would meet, and none of them appears in the live question bank.

The live exam is multistage adaptive: how you score on one stage selects the difficulty of the next, so no two candidates sit the same paper. It is a performance-based exam: each form can carry up to 5 to 12 performance-based questions alongside the multiple-choice questions, which means you build, sequence or complete something rather than only choosing from a list. Answers and full explanations follow each question here; the live exam does not disclose item-level answers.

Question 1

Domain 2: Configuration Language and Resources

CODE

Emma Wilson reviews an infrastructure configuration snippet.

```
resource "db" "main" {
  name     = "prod-db"
  password = "S3cr3t-in-the-repo!"   # hard-coded
  size     = "large"
}
```

Listing 1.1 A database resource definition.

What is the most serious problem, and how should it be fixed?

A. The size is hard-coded, so it should be moved into a variable to let the same code deploy different database sizes per environment.

B. The password is hard-coded, so it lands in version control and state; reference it from a secrets manager or injected variable instead.

C. The resource name is hard-coded, so it should be generated randomly to avoid any chance of a naming collision across environments.

D. Nothing is seriously wrong, because the configuration is short and the password will be encrypted automatically once it is applied.

Correct answer: B
A hard-coded password ends up in the repository history and in state, where it can leak; it should come from a secrets manager or an injected variable and never be written in the code. Parameterising the size is a good idea but minor, the resource name is fine, and nothing encrypts a committed secret away.

Question 2

Domain 2: Configuration Language and Resources

DATA

James Carter labels the elements of a configuration.

Element	Role
Variable	Configurable input value
Output	Exposed result such as an IP
Data source	Read-only lookup of existing infra
Resource	A managed infrastructure object

Table 2.1 Configuration elements and their roles.

Which statement about these elements is correct?

A. A data source creates and manages the lifecycle of a new resource, while a plain resource only reads existing infrastructure.

B. A variable exposes a computed result to other configurations, while an output passes configurable values into the code.

C. A resource is a managed object the tool creates and destroys, while a data source only reads existing infrastructure read-only.

D. An output creates infrastructure objects, while a resource simply reports values such as IP addresses back to the user.

Correct answer: C
A resource is a managed object the tool creates, updates and destroys; a data source is a read-only lookup of existing infrastructure or information. Variables are inputs and outputs are exposed results, so the options that swap those roles or claim a data source creates resources are wrong.

Question 3

Domain 4: Provisioning Workflow and CI/CD

CONFIG

Olivia Davis reviews an IaC pipeline definition.

```
on: push
steps:
  - apply            # every commit applies immediately
  - auth: static admin key stored in the repo
  # no plan step, no approval, no locking
```

Listing 3.1 The pipeline as configured.

Which set of changes makes this pipeline safe?

A. Apply faster on more powerful runners and rotate the static key weekly, since quicker applies and rotation remove the main risks here.

B. Add a reviewed plan step and a production approval gate, use short-lived federated credentials, and enable remote state locking.

C. Apply only on the main branch and keep the static admin key, because restricting the branch is sufficient protection for production changes.

D. Remove the apply step entirely so nothing is ever applied automatically, and have engineers apply by hand from their own laptops instead.

Correct answer: B
The pipeline auto-applies every commit with no plan, no approval, a committed static admin key and no locking. Safe practice adds a reviewed plan and an approval gate for production, replaces the static key with short-lived federated credentials, and enables remote state locking. Faster runners, branch restriction alone or hand-applying from laptops do not address these gaps.

Question 4

Domain 3: State, Modules and Reuse

DATA

Liam Anderson reviews how state is managed across environments.

Aspect	Current setup
State location	On each engineer's laptop
Locking	None
Dev and prod state	One shared state file
Access control	None

Table 4.1 Current state management.

Which problem is the most dangerous, and what fixes it?

A. Laptops holding state is the only real issue, and simply copying the same shared state file to a network drive fully resolves it.

B. A shared dev and prod state means a dev change can destroy prod resources; split state per environment and restrict who can apply to prod.

C. The lack of locking is harmless because engineers coordinate verbally, so no change to the state backend or access model is required here.

D. There is no dangerous problem, because state files are just metadata and losing or sharing them has no effect on real infrastructure.

Correct answer: B
A single shared state for dev and prod is the most dangerous flaw: a dev apply can modify or destroy production resources. The fix is separate state per environment plus access control restricting production applies. Local state, missing locking and no access control are all real problems too, but the shared prod state is the one that can cause immediate, severe damage.

Question 5 — Domain 5: Testing, Policy and Security — CONFIG

Ava Thompson runs a security scan over an IaC change before it is applied.

```
scan results:
  security_group ingress: 0.0.0.0/0 on port 22   (SSH open to internet)
  storage bucket: public-read = true
  required tag "owner": missing
```

Listing 5.1 Findings from the pre-apply scan.

How should these findings be handled?

A. Apply the change now and open tickets to fix the findings later, because scan findings are advisory and do not block a deployment.

B. Fix the open SSH rule and public bucket in the code and add the owner tag before applying, enforced by policy as code failing the plan.

C. Ignore the findings because a scanner cannot know the business context, and only a human reviewer should decide on security settings.

D. Apply only the parts of the change the scanner did not flag, and leave the SSH rule, public bucket and missing tag exactly as they are.

Correct answer: B

These are real, deployable misconfigurations: SSH open to the internet, a public bucket and a missing required tag. They should be fixed in the code before applying, and policy as code should fail the plan so such changes cannot be deployed at all. Applying first and fixing later, or ignoring the findings, would put the insecure resources into production.

Question 6 — Domain 6: Drift, Operations and Collaboration — DATA

A scheduled plan reports differences between the code and reality.

Resource	Difference found
Firewall rule	A port was opened manually in the console
Instance size	Changed by hand during an incident
Tag set	An owner tag removed manually

Table 6.1 Drift detected by the scheduled plan.

What does this indicate, and what is the correct response?

A. It indicates the code is now wrong, so the console should become the source of truth and the code files should simply be deleted.

B. It indicates configuration drift from manual changes; reconcile by updating the code or re-applying, and discourage out-of-band changes.

C. It indicates the scan tool is faulty, so the scheduled plan should be turned off to stop it reporting these harmless differences again.

D It indicates nothing actionable, because manual console changes always take permanent precedence over whatever the IaC code declares.

Correct answer: B
The differences are configuration drift caused by manual console changes, the classic gap between code and reality. The correct response is to reconcile, either bring the changes into the code or re-apply to restore the declared state, and to discourage future out-of-band edits so the code stays authoritative. Deleting the code or disabling drift detection would abandon the single source of truth.

Question 7 Domain 1: IaC Foundations and Principles

A team edits infrastructure both by hand in the console and through IaC. What problem does mixing these create?

A None, because using both the console and IaC together gives the team maximum flexibility with no downside worth worrying about.

B It causes configuration drift: real state diverges from the code, and the next apply may revert or clash with the manual changes.

C It makes deployments faster, because console changes take effect instantly while the IaC code serves purely as after-the-fact documentation.

D It only affects billing, because the resources still work correctly and the sole consequence of mixing the two methods is higher cost.

Correct answer: B
Mixing manual and IaC changes causes drift: the real infrastructure no longer matches the code, and the next apply may revert the manual change or fail unexpectedly. For IaC to work, the code must be the single source of truth, with changes made through it rather than by hand in the console.

Question 8 Domain 4: Provisioning Workflow and CI/CD

How should an IaC pipeline authenticate to the cloud when it applies changes?

A With a long-lived static admin key committed to the repository, so the pipeline can always authenticate without any extra setup.

B With short-lived credentials obtained through workload identity or OIDC federation, so no long-lived secret is stored anywhere.

C With the individual engineer's personal cloud password, entered once and cached, so applies run under a real named human account.

D With no authentication at all in non-production, because only production applies need credentials and lower environments are exempt.

Correct answer: B
Pipelines should authenticate with short-lived credentials from workload identity or OIDC federation, so there is no long-lived secret to leak. A committed static admin key is the leading cause of IaC credential leaks, personal passwords do not belong in automation, and every environment that provisions resources needs proper authentication.

Question 9 Domain 3: State, Modules and Reuse

A company wants every project to use the same compliant, standard network setup. What is the best way to achieve this with IaC?

A. Copy and paste the network configuration into each project, so every team owns its own independent copy that it can freely change.

B. Build a versioned module that encapsulates the standard network and have every project consume a pinned version of that module.

C. Have each team invent its own network setup, because local ownership matters more than consistency across the organisation's projects.

D. Configure the network by hand in the console for each project, using a shared checklist to keep the manual steps roughly consistent.

Correct answer: B
A versioned, tested module encapsulates the standard once, and projects consume a pinned version, giving consistency, reuse and deliberate upgrades. Copy-paste copies diverge over time, letting each team invent its own defeats the goal of a standard, and manual console setup is neither reproducible nor reliably consistent.

Question 10 Domain 5: Testing, Policy and Security

A team wants to guarantee that no resource is ever created without a required "owner" tag. What enforces this reliably?

A. A reminder email to engineers and a note on the team wiki, because clearly communicating the rule is enough to ensure it is followed.

B. A policy-as-code rule in the pipeline that fails any plan which would create a resource missing the required owner tag.

C. A quarterly manual audit that finds untagged resources after the fact, so they can be tagged retrospectively once they are discovered.

D. Trusting each engineer to remember the tag, because experienced engineers rarely forget and occasional lapses are not a real concern.

Correct answer: B
Only automated enforcement reliably guarantees a standard: a policy-as-code rule that fails any non-compliant plan means an untagged resource can never be applied. Reminders, wiki notes, after-the-fact audits and trusting memory all allow untagged resources to slip through, because they depend on people never forgetting.

Published by the CertLabz Certification Board, a division of CertLabz LLC. Companion to objectives document CIACP-001. https://certlabz.com/certifications/certified-infrastructure-as-code-professional.html