CGE-P EXAM BLUEPRINT

Certified GRC Engineer — Practitioner

Official Examination Guide

GRC Engineering Club Training Academy

Instructors: AJ Yawn & Abdie Mohamed

Version 1.0 | March 2026

Built by the community, for the community

www.grcengclub.com

TABLE OF CONTENTS

ABOUT THIS DOCUMENT

Purpose: This exam blueprint serves as the official guide for the Certified GRC Engineer — Practitioner (CGE-P) certification. It outlines the knowledge, skills, and competencies required to pass the examination and provides detailed specifications for all exam domains.

How to Use: Candidates should use this blueprint to understand exam scope, identify knowledge gaps, focus study efforts on high-weight domains, and practice with question types aligned to Bloom's Taxonomy levels. Each domain description includes task statements, knowledge requirements, and skill statements to guide preparation.

Certification Value: The CGE-P certification validates that engineers can design, implement, and maintain automated governance, risk, and compliance (GRC) solutions using modern infrastructure and policy-as-code practices. This certification demonstrates competency in bridging the gap between compliance requirements and engineering execution.

EXAM STRUCTURE & FORMAT

Attribute	Details
Format	Computer-based, proctored
Time Limit	90 minutes
Total Questions	60
Multiple-Choice	50 questions (4 options, single correct answer)
Scenario-Based	10 questions (read scenario, answer 1–2 follow-up questions)
Passing Score	72% (43/60 correct)
Scoring	All questions weighted equally; scenarios count as individual questions
Negative Marking	None
Retake Policy	14-day waiting period between attempts; maximum 3 attempts per 12 months
Breaks	No section breaks; 90 minutes total

Question Distribution: The exam covers all seven domains with questions distributed by domain weight. Higher-weighted domains have more questions, allowing comprehensive assessment of critical competencies.

Difficulty Levels: The exam includes questions at multiple Bloom's Taxonomy levels, from foundational knowledge to advanced analysis and evaluation. This ensures assessment of both depth of understanding and practical application ability.

Adaptive Features: Scenario-based questions require candidates to apply knowledge in realistic situations. These questions assess ability to synthesize concepts and make informed decisions in context.

DOMAIN OVERVIEW

#	Domain	Weight	Questions	Focus
1	GRC Engineering Foundations	15%	9	Core concepts, maturity model, frameworks, career paths
2	Infrastructure as Code	20%	12	Terraform, compliant resources, modules, evidence collection
3	Policy-as-Code	15%	9	OPA/Rego, compliance policies, enforcement
4	CI/CD for GRC Engineers	15%	9	Pipelines, security testing, evidence automation
5	Cloud-Native Security & Monitoring	15%	9	Control validation, continuous monitoring, dashboards
6	OSCAL & Continuous Authorization	10%	6	OSCAL models, cATO, transformation roadmaps
7	Applied GRC Engineering	10%	6	Integration, portfolio projects, end-to-end workflows
TOTAL		100%	60	

Domain Weight Explanation: Domain weights reflect the importance and scope of each area in real-world GRC Engineering practice. Higher-weighted domains require deeper knowledge and represent critical competencies. Infrastructure as Code and Cloud-Native Security & Monitoring are weighted highest because automation and continuous validation are central to modern GRC practices.

COURSE → DOMAIN CROSSWALK

Every video lesson and every hands-on lab is mapped to the domain it is tested under. Use this table to make sure your study time is distributed by exam weight.

#	Domain	Speaker decks covered	Hands-on labs
1	GRC Engineering Foundations	01_01 · 01_02 · 01_03 · 01_04 · 01_05	(none required)
2	Infrastructure as Code	02_01 · 02_02 · 02_03 · 02_04 · 02_05	2.3 · 2.4 · 2.5
3	Policy-as-Code	03_01 · 03_02 · 03_03 · 03_04 · 03_05	3.3 · 3.4
4	CI/CD for GRC Engineers	04_01 · 04_02 · 04_03 · 04_04	4.3 · 4.4

#	Domain	Speaker decks covered	Hands-on labs
5	Cloud-Native Security & Monitoring	05_01 · 05_02 · 05_03 · 05_04	5.2 · 5.4
6	OSCAL & Continuous Authorization	06_01 · 06_02 · 06_03	6.1
7	Applied GRC Engineering	07_01 (Capstone Brief)	Capstone (assembles all prior labs)

DOMAIN 1: GRC ENGINEERING FOUNDATIONS

Weight: 15% (9 questions)

TASK STATEMENTS

1.1 Define GRC Engineering and its role in organizational risk management

1.2 Place an organization on the 5-level GRC Engineering Maturity Model (Reactive, Documented, Automated, Continuous, Adaptive)

1.3 Explain the three pillars of GRC Engineering (Automate, Evidence, Continuous)

1.4 Map compliance frameworks to engineering practices and toolsets

1.5 Identify required skills and career progression paths for GRC Engineers

KNOWLEDGE STATEMENTS

- Definition and principles of GRC Engineering
- The 5-level GRC Engineering Maturity Model: Reactive, Documented, Automated, Continuous, Adaptive
- The Three Pillars of GRC Engineering: Automate (IaC + PaC), Evidence (machine-verifiable receipts), Continuous (every merge is an audit event)
- The big five compliance frameworks: NIST CSF, NIST 800-53, ISO 27001, SOC 2, FedRAMP
- Role responsibilities and required competencies

SKILLS STATEMENTS

- Build business case for GRC Engineering adoption
- Assess organizational maturity using established models
- Map compliance requirements to engineering practices
- Communicate GRC Engineering value to stakeholders

BLOOM'S TAXONOMY DISTRIBUTION

Level	Questions
Remember	2
Understand	4
Apply	2
Analyze	1
Evaluate	0

Level	Questions
Create	0

DOMAIN 2: INFRASTRUCTURE AS CODE

Weight: 20% (12 questions)

TASK STATEMENTS

2.1 Explain Infrastructure as Code (IaC) principles, declarative vs. imperative, and the diff/review workflow

2.2 Author a compliant cloud resource (e.g., S3 bucket) with hardcoded SC-28 encryption, AC-3 public-access block, AU-3 access logging, and CM-6 required tags

2.3 Build a reusable Terraform module that enforces compliance defaults (encryption, versioning, retention, required labels) and emits a compliance attestation output

2.4 Capture pre-deployment plan output and post-deployment state as machine-readable JSON evidence

2.5 Wire IaC artifacts into an immutable evidence vault (S3 Object Lock or equivalent)

KNOWLEDGE STATEMENTS

- Declarative IaC principles, HCL syntax, and provider model
- Terraform plan / apply / state lifecycle and remote backends with locking
- Provider default_tags pattern and variable validation blocks for compliance enforcement
- Module composition: primitive, stack, and landing-zone patterns
- Object Lock retention modes (GOVERNANCE vs. COMPLIANCE) and when to use each
- Mapping resource arguments to NIST 800-53 control families (SC-28, AC-3, AU-3, AU-6, CM-6)

SKILLS STATEMENTS

- Write, plan, and apply Terraform with a remote backend
- Compose modules that hide compliance choices from consumers
- Hand-trace a terraform plan -json output and identify the controls it satisfies
- Capture and sign evidence bundles from a Terraform workspace
- Diagnose state-lock and provider-config errors

BLOOM'S TAXONOMY DISTRIBUTION

Level	Questions
Remember	1
Understand	3
Apply	6
Analyze	2
Evaluate	0
Create	0

Hands-On Labs: 2.3 First Compliant Resource (AWS S3); 2.4 Terraform Modules for Compliance (GCP GCS); 2.5 IaC as Compliance Evidence (AWS Object Lock).

DOMAIN 3: POLICY-AS-CODE

Weight: 15% (9 questions)

TASK STATEMENTS

3.1 Define Policy-as-Code and the three problems it fixes (ambiguity, latency, scale)

3.2 Author Rego policies with structured # METADATA blocks (control_id, severity, remediation)

3.3 Write Rego rules against a terraform plan -json input and assert deny-set contents in unit tests (opa test)

3.4 Run a Conftest gate against a Terraform plan, fail closed, and produce machine-readable JSON results

3.5 Make the case for Policy-as-Code adoption: stakeholder mapping, observe/educate/enforce/expand rollout

KNOWLEDGE STATEMENTS

- OPA architecture: input + policy = decision
- Rego v1 syntax: package, deny rules, set comprehensions, set subtraction
- Policy metadata blocks and self-documenting deny messages with control IDs
- Conftest namespaces and the difference between opa eval, opa test, and conftest test
- Cross-cloud policy reuse: control IDs are portable, resource-type filters are not

SKILLS STATEMENTS

- Write a deny rule and a passing/failing fixture
- Run opa test and interpret PASS/FAIL output
- Add an AWS variant to a GCP-targeted policy without breaking existing tests
- Wrap a Conftest run in a shell script that the pipeline calls
- Drive a stakeholder rollout: warn-only, then enforce, then expand

BLOOM'S TAXONOMY DISTRIBUTION

Level	Questions
Remember	1
Understand	3
Apply	4
Analyze	1
Evaluate	0
Create	0

Hands-On Labs: 3.3 Writing Compliance Policies in Rego (GCP fixtures); 3.4 Integrating PaC with Terraform via Conftest (AWS plan).

DOMAIN 4: CI/CD FOR GRC ENGINEERS

Weight: 15% (9 questions)

TASK STATEMENTS

4.1 Map the five CI/CD stages (source, build, test, deploy, observe) to compliance control insertion points

4.2 Connect existing developer security tools (SAST/SCA/DAST/container scanners) to NIST 800-53 control families

4.3 Build a GitHub Actions workflow that runs Terraform plan + Conftest + tfsec on every PR and uploads a named evidence artifact

4.4 Sign evidence bundles with Cosign keyless via GitHub OIDC and upload them to an Object Lock vault

4.5 Verify chain of custody end-to-end: integrity (SHA-256), authenticity (Cosign + Rekor), preservation (Object Lock retention)

KNOWLEDGE STATEMENTS

- Pipeline stages and control insertion points
- GitHub Actions OIDC trust to AWS (sts:AssumeRoleWithWebIdentity, sub-claim scoping)
- Conftest exit codes, --output=json, namespace selection, and the policy-gate.sh wrapper pattern
- Cosign keyless signing flow: OIDC token to Fulcio cert to Rekor entry (the .sig.bundle)
- S3 Object Lock retention modes and how Object Lock blocks DeleteObject and overwrite
- How the workflow YAML itself is CM-3 / CM-6 / CA-2 / RA-5 / AU-9 evidence

SKILLS STATEMENTS

- Author a workflow that fails closed on policy or scan findings while still uploading evidence
- Configure an OIDC trust relationship scoped to one repository
- Inspect a CI run, locate the evidence artifact, and download/verify it
- Trace a verify-evidence.sh failure to the specific broken property (integrity, authenticity, preservation)
- Diagnose common failure modes: missing id-token: write permission, Conftest path errors, Rekor propagation race

BLOOM'S TAXONOMY DISTRIBUTION

Level	Questions
Remember	1
Understand	2
Apply	4
Analyze	2
Evaluate	0
Create	0

Hands-On Labs: 4.3 Building a GRC Evidence Pipeline (AWS + GitHub Actions); 4.4 Evidence Management & Chain of Custody (Cosign keyless + S3 Object Lock).

DOMAIN 5: CLOUD-NATIVE SECURITY & MONITORING

Weight: 15% (9 questions)

TASK STATEMENTS

5.1 Identify the five functional families of cloud security services (Identity, Logging, Detection, Secrets, Posture) and place each cloud's catalog inside them

5.2 Stand up the AWS-native compliance backbone via Terraform: CloudTrail (multi-region, log-file-validation), Security Hub (NIST 800-53 standard), and AWS Config where SCP allows

5.3 Stand up the GCP identity-first baseline: Org Policy enforcing API-level rejections, Workload Identity Federation in place of service account keys, and Data Access logs explicitly enabled per service

5.4 Translate findings into evidence: capture Security Hub findings or GCP IAM audit configs as JSON and route through the evidence pipeline

KNOWLEDGE STATEMENTS

- The Rosetta Stone: AWS CloudTrail, Azure Activity Logs, GCP Cloud Audit Logs are the same job; AWS Config, Azure Policy, GCP Security Command Center are the same job
- AWS chain: CloudTrail feeds Config and GuardDuty; Security Hub aggregates
- GCP chain: Organization-Folder-Project hierarchy; Org Policy enforces at the API call (not audit-after-the-fact)
- Service-to-control mapping: CloudTrail to AU-2/AU-12/AU-10; Config to CM-2/CM-6/CM-8; GuardDuty/SCC to SI-4/RA-5; Security Hub aggregator to CA-7
- Watch-outs: Config and Security Hub bill per check; Data Access logs in GCP are off by default; service account JSON keys are the #1 GCP breach vector

SKILLS STATEMENTS

- Read a Security Hub finding and identify the underlying NIST control family
- Configure GitHub Actions to assume an AWS role via OIDC and assume a GCP service account via Workload Identity Federation, both without long-lived keys
- Recognize when an org-level SCP is blocking a Config enable and route around it
- Choose the right service per family for a given workload (Identity Center vs ABAC; CloudTrail data events vs management-only)
- Defend a posture-management strategy that uses native tools first and third-party only where there's a real cross-cloud gap

BLOOM'S TAXONOMY DISTRIBUTION

Level	Questions
Remember	1
Understand	2
Apply	3
Analyze	2
Evaluate	1
Create	0

Hands-On Labs: 5.2 AWS Security Services Baseline (CloudTrail + Security Hub); 5.4 GCP Security Services Baseline (Org Policy + WIF + Data Access logs).

DOMAIN 6: OSCAL & CONTINUOUS AUTHORIZATION

Weight: 10% (6 questions)

TASK STATEMENTS

6.1 Name OSCAL's five models (Catalog, Profile, Component, SSP, Assessment) and place a component-definition.json in the right place

6.2 Author a Component Definition that references real Terraform resources and links to signed evidence URIs in the vault

6.3 Validate OSCAL documents with compliance-trestle (UUID v4 format, schema-strict requirements)

6.4 Apply the DoD cATO three pillars (Continuous Monitoring, Active Cyber Defense, DevSecOps) to a roadmap from current state

6.5 Use the Now / Next / Later framework to sequence GRC engineering investments without overcommitting

KNOWLEDGE STATEMENTS

- OSCAL five-model hierarchy and which models a Practitioner authors versus consumes
- Component Definition structure: components, control-implementations, implemented-requirements, links
- Profile resolution against a catalog (trestle profile-resolve)
- cATO definition and the three DoD memo pillars
- Now / Next / Later vs. Gantt: why the ritual matters more than the artifact

SKILLS STATEMENTS

- Generate v4 UUIDs and pass trestle validate
- Wire link href values to objects in an Object Lock vault so the audit becomes a graph traversal
- Build a one-page roadmap with three buckets and an explicit Later column
- Map a workload to the relevant cATO pillars and identify the next phase
- Read an existing OSCAL document and tell whether it was hand-authored or generated from a template

BLOOM'S TAXONOMY DISTRIBUTION

Level	Questions
Remember	1
Understand	2
Apply	2
Analyze	1
Evaluate	0
Create	0

Hands-On Labs: 6.1 Introduction to OSCAL (compliance-trestle + Component Definition for a Terraform module).

DOMAIN 7: APPLIED GRC ENGINEERING

Weight: 10% (6 questions)

TASK STATEMENTS

7.1 Inherit a deliberately non-compliant workload and assemble the four required capstone layers (Terraform baseline, OPA suite, GitHub Actions pipeline, OSCAL component) around it

7.2 Declare a primary compliance framework (HIPAA Security Rule, SOC 2 TSC, or CMMC L2) and structure every capstone layer around that choice

7.3 Demonstrate end-to-end integration: an opened pull request triggers the policy gate, fail or pass drives the apply, the apply produces a signed evidence bundle in the vault

7.4 Defend design trade-offs in a written stakeholder-facing document (scope choices, framework choice, what was not done)

KNOWLEDGE STATEMENTS

- The four capstone layers and how they compose: baseline -> policies -> pipeline -> OSCAL
- Capstone scoring rubric: end-to-end integration, working evidence pipeline, clear design reasoning
- The three common capstone failure modes: too much scope, copy-paste OSCAL, unsigned evidence
- Framework selection rationale and how it propagates from policy metadata into OSCAL implementation links
- Honest gap reporting: documenting what you did NOT get to is part of passing

SKILLS STATEMENTS

- Decompose a workload into named gaps and pick the smallest set of layers that closes them
- Trace a single change from PR open to signed evidence in the vault
- Choose between closing a gap in Terraform vs. enforcing it only in policy, and defend the choice
- Write a five-page write-up that explains why, not just what
- Estimate residual risk and report it without hand-waving

BLOOM'S TAXONOMY DISTRIBUTION

Level	Questions
Remember	0
Understand	1
Apply	2
Analyze	1
Evaluate	1
Create	1

Hands-On Labs: 7.1 Capstone Project Brief (assemble all prior labs into one repo).

IAC PORTFOLIO ASSESSMENT

Portfolio Purpose

The capstone portfolio demonstrates that the candidate can assemble the four course layers into one working system on a real cloud account. The candidate inherits a deliberately non-compliant workload and wraps it with a compliant baseline, a policy gate, an evidence pipeline, and an OSCAL component. The capstone is the same skills the multiple-choice exam tests, this time end-to-end.

Step 0: The Deploy Gate

Before building anything, the candidate forks a public starter workload, deploys it to their own cloud sandbox, and confirms it runs (a make test smoke check). Real GRC engineers inherit working systems. Demonstrating a clean deploy is the floor.

Required Deliverables (Four Layers)

1. **Terraform Baseline** — KMS keys, an S3 evidence bucket with Object Lock, a multi-region CloudTrail trail, and the gap-closing overrides on the inherited workload (encryption with customer CMK, public access blocks, IAM least privilege, etc.)
2. **OPA Policy Suite** — Five or more Rego policies with # METADATA blocks. Each policy cites a control ID from the candidate's declared primary framework (HIPAA Security Rule, SOC 2 TSC, or CMMC L2). Each policy has a passing and failing test fixture.
3. **GitHub Actions Pipeline** — One workflow with five named steps in order: Plan, Policy Check, Apply, Sign (Cosign keyless via GitHub OIDC), Upload to vault. Two pull requests must exist in the repo's history: one that passed and merged, one that failed the policy gate and was blocked.
4. **OSCAL Component** — A component-definition.json that describes what was actually built. The control-implementation source points at the declared framework's catalog. Evidence links resolve to real signed objects in the candidate's vault.

Plus a five-page WRITEUP.md in the repo explaining design decisions, framework choice, trade-offs, and an honest list of what the candidate did not get to.

Grading Criteria

Category	Weight	Criteria
End-to-End Integration	35%	Open a PR, the gate runs, the gate decides whether apply happens, apply triggers signing, signing uploads to the vault. Not four disconnected demos.
Working Evidence Pipeline	30%	Reviewer pulls a recent run and verifies it: Cosign signature against the public Sigstore log, SHA-256 recompute, Object Lock retention check. All three must hold.
Clear Design Reasoning	20%	WRITEUP.md explains why each tool was chosen and what trade-offs were accepted. Honest gap reporting.
Framework Alignment	15%	Policies cite the declared framework's control IDs; OSCAL's control-implementation source matches. Multi-framework cross-references in props are encouraged.

Submission Process

Submission: Public GitHub repository with all deliverables. Allow 5–7 business days for review.

Grading: Two independent reviewers evaluate each portfolio. Both must agree on pass/fail.

Feedback: Candidates receive written feedback regardless of outcome. Failed portfolios may be resubmitted after addressing feedback (one resubmission allowed).

CERTIFICATION MAINTENANCE

Validity Period

The CGE-P certification is valid for three years from the date of issuance. To keep the credential active, candidates must complete an annual renewal each year of the three-year cycle. After three years, candidates must re-take the exam to continue holding the certification.

Annual Renewal Options

Each year of the three-year cycle, the candidate completes one of the following to keep the certification active. Both options are accepted; only one is required per year.

Option 1: Active Club Membership — Maintain an active GRC Engineering Club membership. Certification auto-renews each year while membership is in good standing.

Option 2: CPE Hours — Submit 20 approved Continuing Professional Education (CPE) hours within the past 12 months.

Approved CPE Activities

Activity	CPE Hours
Conference presentation on GRC Engineering topic	4 hours
Published article or blog post on GRC Engineering	2 hours
Open-source contribution to GRC tools	2–4 hours
Completing advanced training courses	1 hour per course hour
Mentoring new GRC Engineers	1 hour per session (max 10)

BLOOM'S TAXONOMY REFERENCE

Taxonomy Levels

Level	Description	Example Question Stem
Remember	Recall facts and basic concepts	"What does OSCAL stand for?"
Understand	Explain ideas or concepts	"Which best describes the purpose of..."
Apply	Use information in new situations	"Given this Terraform configuration, what..."
Analyze	Draw connections among ideas	"Which combination of tools would best..."
Evaluate	Justify a decision or course of action	"Your organization needs to choose between..."
Create	Produce new or original work	"Design an approach that integrates..."

Distribution Across Exam

Level	Total Questions
Remember	6
Understand	17
Apply	23
Analyze	10
Evaluate	2
Create	2

CANDIDATE PREPARATION RECOMMENDATIONS

Study Strategy

1. **Start with the Study Guide** — Work through all seven domains in order, beginning with GRC Engineering Foundations
2. **Focus on High-Weight Domains** — Infrastructure as Code (20%) and Cloud-Native Security (15%) deserve the most study time
3. **Practice with Scenarios** — Scenario questions test application of knowledge; practice by working through real-world situations
4. **Build Your Portfolio Early** — Start the IaC portfolio alongside your studies to reinforce learning with hands-on practice
5. **Use the Video Course** — Watch corresponding video lectures for each domain to reinforce study guide content
6. **Take Practice Exams** — Use the practice exam to identify knowledge gaps and adjust your study plan

Recommended Resources

- GRC Engineering Club Training Academy (video course)
- CGE-P Study Guide (companion to this blueprint)
- Terraform documentation (terraform.io)
- Open Policy Agent documentation (openpolicyagent.org)
- NIST OSCAL documentation (pages.nist.gov/OSCAL)
- NIST 800-53 Control Catalog

Contact Information

Website: www.grcengclub.com

Email: academy@grcengclub.com

Community: GRC Engineering Club (Patreon)