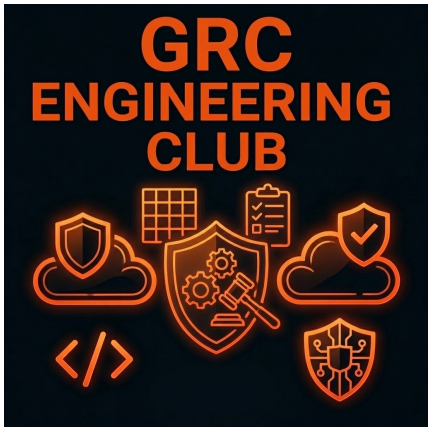

CGE-AUD

Study Guide

Certified GRC Engineer, Auditor Specialty

GRC Engineering Club Training Academy

Instructor: AJ Yawn, former SOC 2 auditor

Version 1.0 (Beta) | June 2026 | Launches July 15, 2026

www.grcengclub.com

HOW TO USE THIS STUDY GUIDE

Overview of the Certification

The Certified GRC Engineer, Auditor Specialty (CGE-AUD) validates your ability to audit a modern, GRC-engineered organization. It gives traditional auditors the cloud, code, pipeline, and policy literacy to read engineering artifacts as evidence, test automated and continuous controls, and partner with GRC engineers instead of slowing them down. The one-line goal: audit the modern stack without becoming an engineer.

Who Should Take This Exam

Internal auditors, external (CPA-firm) IT auditors, SOC 2 and ISO 27001 auditors, PCI QSAs, HIPAA and HITRUST assessors, risk assurance professionals, and GRC analysts who now audit cloud-native, SaaS-first, DevOps-driven organizations that look nothing like the systems they were trained on.

Prerequisites

None. The CGE-AUD assumes near-zero prior technical knowledge and is standalone, with no CGE-P prerequisite. We skip a generic frameworks chapter on purpose: commercial auditors already know SOC 2, ISO 27001, PCI DSS, and HIPAA. This guide opens straight into the technical literacy they are missing.

Recommended Study Timeline

Timeline	Activities	Estimated Hours
Week 1	Domains 1 and 2 (Cloud & Code Literacy, Auditing IaC)	2–3 hours
Week 2	Domains 3 and 4 (CI/CD, Cloud-Native Monitoring)	2–3 hours
Week 3	Domains 5 and 6 (Claude Code, Evidence Evaluation)	2–3 hours
Week 4	Domain 7, frameworks review, practice questions	2–3 hours

Total learner study time is roughly 8 to 10 hours: about four hours of video plus reading, practice questions, and the exam. There are no labs to set up.

How to Use This Guide with the Video Course

This study guide complements the CGE-AUD video course taught by AJ Yawn, a former SOC 2 auditor. Use the guide as your primary reference, working through each domain in order. Every domain pairs a short technical primer (here is what this thing actually is) with auditor-specific guidance (here is how to test it and what bad looks like).

EXAM OVERVIEW

Exam Format

Attribute	Details
Total Questions	50 questions
Question Types	40 Multiple Choice + 10 Scenario-Based
Time Limit	75 minutes
Pass Score	70% (35/50 correct answers)
Portfolio	None (knowledge exam only)
Retake Policy	14-day wait between attempts; max 3 attempts per 12 months
Format	Computer-based (online proctored)

Domain Breakdown and Question Distribution

Domain	Topic	Weight	Questions	Study Focus
1	Cloud & Code Literacy	20%	~10	Cloud, shared responsibility, JSON/YAML, Git, reading code
2	Auditing IaC	15%	~7	Reading Terraform, sampling, state & drift evidence
3	Auditing CI/CD	15%	~8	Pipeline anatomy, SoD, build logs, approval gates
4	Cloud-Native Monitoring	15%	~7	Automated controls, drift, audit logs, KSIs
5	Claude Code & AI Tooling	15%	~8	AI-assisted evidence, test procedures, guardrails
6	Evidence Evaluation	10%	~5	Evidence quality, sampling, workpapers, Audit Playbook
7	Applied Commercial Scenarios	10%	~5	SOC 2, ISO 27001, PCI DSS in the cloud

No Portfolio Required

Auditors evaluate, they do not build. The CGE-AUD is a knowledge certification with no capstone and no lab setup. Ten of the fifty questions are scenario-based: each presents a short artifact (a Terraform snippet, a pull request description, a CloudTrail event, a build log) and asks you to evaluate it, identify the finding, or choose the right test procedure.

Scoring and Certification

Exam Score: You receive a scaled score. 70% or higher (35 of 50) signifies passing.

Certification Awarded: Upon passing, you earn the Certified GRC Engineer, Auditor Specialty (CGE-AUD) credential, valid for three years.

DOMAIN 1: CLOUD & CODE LITERACY FOR AUDITORS

20% of Exam, Approximately 10 Questions

What You Need to Know

Domain 1 is the foundation for the whole certification. An auditor who cannot read JSON, navigate a repository, or interpret a Terraform file cannot evaluate the evidence the other six domains produce. You will build a working mental model of cloud, learn to read the file formats every cloud artifact uses, and learn to read code without writing it.

1.1 Why Auditors Need Technical Literacy

Cloud-native organizations replaced manual checklists with Terraform modules, OPA policies, CI/CD gates, and continuous monitoring. Most audits still ask for screenshots and spreadsheets. The result is longer audits, weaker assurance, and frustrated engineers. The fix is not to become an engineer. It is to read engineering artifacts directly as evidence.

1.2 Cloud 101 and the Shared Responsibility Model

A cloud account (AWS), subscription (Azure), or project (GCP) is the top-level billing and isolation boundary. Resources live in regions and availability zones. The shared responsibility model splits control ownership between the cloud provider (security of the cloud) and the customer (security in the cloud). Scoping a SOC 2 or ISO audit correctly starts with knowing which side of that line each control falls on.

1.3 Reading JSON and YAML Without Writing Code

Almost every cloud artifact an auditor receives is JSON or YAML: CloudTrail events, IAM policies, pipeline definitions, configuration exports. JSON is keys, values, nesting, and arrays. YAML is the same data in a more readable layout. You do not write these; you read them well enough to evaluate them as evidence. Reading a real CloudTrail event and an IAM policy is the core skill.

1.4 Git and GitHub: Where Modern Compliance Lives

Modern compliance lives in version control. A repository holds the code; commits record every change with an author and timestamp; branches and pull requests record who proposed and who approved a change. That history is an audit trail. An auditor can navigate a GitHub repository and pull commit history as evidence without ever touching the command line.

1.5 Reading Code You Do Not Write: HCL and Terraform

Terraform uses HCL (HashiCorp Configuration Language). At a glance: resource blocks declare infrastructure, variables parameterize it, and modules package it for reuse. You do not need to write HCL. You need to read a Terraform file well enough to know what infrastructure it deploys and which controls that infrastructure implements.

Key Terms and Definitions

Region / Availability Zone: Geographic and isolation boundaries for cloud resources.

Shared Responsibility Model: The split of security duties between cloud provider and customer.

JSON / YAML: The two text formats nearly every cloud artifact is expressed in.

Repository: A version-controlled store of code and its full change history.

Pull Request (PR): A proposed change that records who authored and who approved it.

HCL: HashiCorp Configuration Language, the syntax Terraform is written in.

Sample Study Question, Domain 1

Question: An auditor is scoping a SOC 2 audit of a SaaS company running on AWS. Management says physical security of the data centers is out of scope. Is this correct?

A) No. Physical security is always the customer's responsibility.

B) Yes. Under the shared responsibility model, physical security of the cloud is the provider's responsibility and is covered by the provider's own attestations.

C) No. Physical security cannot be audited in the cloud.

D) Yes, but only if the customer has no production data in AWS.

Answer: B) Yes. Under the shared responsibility model, physical security of the cloud is the provider's responsibility and is covered by the provider's own attestations.

Explanation: Security of the cloud (physical facilities, hardware) belongs to AWS and is typically covered by AWS's own SOC 2 / ISO reports, which the customer inherits. The customer is responsible for security in the cloud (configuration, access, data). Scoping starts with placing each control on the right side of that line.

DOMAIN 2: AUDITING INFRASTRUCTURE AS CODE

15% of Exam, Approximately 7 Questions

What You Need to Know

Infrastructure as code (IaC) defines infrastructure declaratively in files instead of clicking through a console. For an auditor, that changes everything: the population of resources is defined in code, control configuration is visible before deployment, and state and drift reports become point-in-time evidence.

2.1 What IaC Is and Why Auditors Should Care

Declarative infrastructure means the code describes the desired end state and the tool makes reality match it. Because the configuration is in version control, an auditor can see exactly what was provisioned, when it changed, and who approved the change. Sampling and evidence both shift from after-the-fact collection to reading the source of truth.

2.2 Reading a Terraform Module Like an Auditor

When you open a module that provisions, say, an S3 bucket, look first for the control-relevant arguments: encryption at rest, access logging, public-access blocks, and required tags. What is present tells you which controls are implemented. What is missing, or set to a permissive value, is a red flag and a likely finding.

2.3 Sampling Resources from IaC

Traditional sampling assumes you cannot test the whole population. When the population is generated from a module, you often can: every bucket made from the same module inherits the same controls. That enables complete-population testing for the templated attributes, with judgmental sampling reserved for exceptions and resources created outside the module.

2.4 IaC as Evidence: State, Drift, and Workpapers

The Terraform state file records what is actually deployed. Drift is the gap between the code and reality. A drift report is point-in-time evidence: it shows whether the deployed environment still matches the approved configuration. Document the state and drift output in the workpaper, referencing the specific files and the date captured.

Sample Study Question, Domain 2

Question: A Terraform module for an S3 bucket includes an aws_s3_bucket_server_side_encryption_configuration resource and an aws_s3_bucket_public_access_block with all four settings true, but no access logging resource. As the auditor, what do you conclude?

A) Encryption and public-access controls are implemented in code; access logging appears to be a gap to investigate.

B) The module is non-compliant and must be rejected outright.

C) Nothing can be concluded without a screenshot of the console.

D) Logging is automatic in AWS, so its absence in code is irrelevant.

Answer: A) Encryption and public-access controls are implemented in code; access logging appears to be a gap to investigate.

Explanation: Reading the module tells you which controls are present (encryption, public-access blocks) and which are absent (access logging). The absence is a finding to investigate, not an automatic fail: logging may be configured elsewhere. The point is that the code itself is the evidence; you do not need a screenshot to evaluate it.

DOMAIN 3: AUDITING CI/CD PIPELINES

15% of Exam, Approximately 8 Questions

What You Need to Know

A CI/CD pipeline builds, tests, scans, and deploys code automatically. For an auditor, the pipeline is where change management, segregation of duties, and security testing controls actually operate, and every run produces logs that serve as evidence in place of screenshots.

3.1 Pipeline Anatomy: GitHub Actions, GitLab CI, and Jenkins

A pipeline is defined in a file (for GitHub Actions, a YAML workflow). It runs stages: build, test, scan, deploy. Different platforms (GitHub Actions, GitLab CI, Jenkins) use different syntax but the same shape. Learning to read a workflow file tells you what the pipeline does and which checks gate a deployment.

3.2 Segregation of Duties in a Pipeline World

SoD in modern engineering is enforced by configuration: who can push to a branch, who must approve a pull request, and who can trigger a deploy. Branch protection rules requiring a separate reviewer are the control. Testing SoD means reading those settings, not interviewing people about intent.

3.3 Build Logs as Control Evidence

Every pipeline run produces a log: what ran, when, who triggered it, and whether each check passed. Those logs map directly to change management and security testing controls. A build log that shows tests and scans passing before deploy is stronger, time-stamped evidence than a screenshot of a dashboard.

3.4 Approval Gates, Security Tests, and What Good Looks Like

Mature pipelines combine automated gates (SAST, DAST, SCA, vulnerability scanning) with required human approvals for sensitive changes. Good looks like: required reviews, failing builds that block deploys, and scans wired into the gate. Common gaps include optional checks, self-approval, and scans that run but never block.

Sample Study Question, Domain 3

Question: In a GitHub repository, the main branch has a protection rule requiring one approving review, but the rule allows the author to approve their own pull request. What control concern should the auditor raise?

A) None; one approval is sufficient.

B) Segregation of duties: self-approval lets one person both author and approve a change, defeating the review control.

C) Encryption at rest is not configured.

D) The repository should not be public.

Answer: B) Segregation of duties: self-approval lets one person both author and approve a change, defeating the review control.

Explanation: The branch protection rule is the SoD control. Allowing the author to approve their own PR means a single person can author and authorize a production change, which is the classic SoD failure. The auditor reads the setting directly and raises the finding; no interview or screenshot is required.

DOMAIN 4: CLOUD-NATIVE MONITORING & CONTINUOUS CONTROLS

15% of Exam, Approximately 7 Questions

What You Need to Know

Cloud-native controls often operate continuously rather than at a single point in time. That changes the audit: instead of sampling a moment, you evaluate whether a control operates effectively across the whole period, using drift detection, audit logs, and key security indicators.

4.1 Testing Automated Controls vs. Point-in-Time Sampling

When a control runs automatically on every change, evidence of operating effectiveness is different: you assess the control's configuration and its continuous output across the audit period rather than pulling one sample. This is a shift in test procedure design, not a reason to fall back on monthly screenshots.

4.2 Drift Detection and What It Means for the Audit

Drift is when deployed infrastructure deviates from its coded, approved state. Tools such as Terraform and AWS Config detect it. A drift report that shows an unauthorized change is, in effect, an audit finding generated by the system. Interpreting drift output lets the auditor identify control failures directly.

4.3 CloudTrail, Azure Monitor, and GCP Audit Logs

Each cloud records its own audit log: AWS CloudTrail, Azure Monitor / Activity Logs, and GCP Cloud Audit Logs. They record who did what, when. An auditor requests, reads, and samples these logs as evidence for access and change controls. Watch for common gaps: data-access logging is often off by default, and retention may be too short.

4.4 Key Security Indicators (KSIs) Auditors Can Trust

Management often presents KSIs and KPIs from continuous monitoring. A KSI is only as trustworthy as the calculation behind it. The auditor's job is to validate the KSI itself: what data feeds it, how it is computed, and whether it actually measures the control it claims to measure before relying on it for comfort.

Sample Study Question, Domain 4

Question: An automated control re-checks every S3 bucket for public access every hour and alerts on violations. Management offers the control's full year of run history. How should the auditor test operating effectiveness?

A) Pull one screenshot from a single day as the sample.

B) Evaluate the control's configuration and review its continuous output across the audit period, investigating any gaps or alerts.

C) Decline to test because automated controls cannot be audited.

D) Require management to perform the check manually each month instead.

Answer: B) Evaluate the control's configuration and review its continuous output across the audit period, investigating any gaps or alerts.

Explanation: A continuously-operating control calls for assessing the configuration plus the full-period output, not a single point-in-time sample. The year of run history is stronger evidence than a screenshot. Reverting to manual monthly checks would discard the control's actual strength.

DOMAIN 5: CLAUDE CODE & AI TOOLING FOR AUDITORS

15% of Exam, Approximately 8 Questions

What You Need to Know

This domain is where the CGE-AUD differs from every other auditor certification. You will learn to use Claude Code and the GRC Engineering Club's auditor skills to pull configurations, draft test procedures, and analyze JSON evidence without writing code, and just as important, when not to trust the output.

5.1 Why Claude Code Matters for Auditors

AI-assisted audit is becoming the baseline for evidence work, not a nice-to-have. An auditor who can ask an AI tool to extract and summarize configurations works through engineered evidence far faster than one collecting screenshots by hand, while keeping full review responsibility.

5.2 Setting Up Claude Code as an Auditor

Claude Code is set up and run without a coding background. The course provides a first-run walkthrough tailored to non-technical auditors so you can follow along on your own machine. The detailed setup steps are delivered with the video lessons.

5.3 The GRC Engineering Club Claude Code Project

The Club maintains a Claude Code project built for auditors. It packages the prompts and skills an auditor needs to work with cloud evidence. The repository, contents, and install steps are provided with the course materials.

5.4 Auditor Skills Library

The Auditor Skills Library is a catalog of reusable skills: evidence-pull skills that retrieve configurations, control-mapping skills that align evidence to framework controls, and workpaper-drafting skills that turn findings into review-ready language. You learn which skill to invoke for which task.

5.5 Worked Examples: Auditing with Claude Code

The course walks through concrete examples, such as using Claude Code to validate a SOC 2 CC6.1 logical-access control across an AWS account: pull the relevant configuration, analyze it against the control, and draft the test procedure and result.

5.6 Guardrails: When NOT to Trust the Output

AI assists; it does not sign the opinion. You remain responsible for reviewing every output before it enters a workpaper. Understand the independence implications of AI use, document how and where AI was used in the engagement, and verify AI-produced conclusions against the underlying evidence.

Sample Study Question, Domain 5

Question: An auditor uses Claude Code to summarize 200 IAM policies and the tool reports that none grant wildcard admin access. What is the appropriate next step before relying on this in the workpaper?

A) Cite the AI summary as conclusive evidence; no further work is needed.

B) Review the underlying evidence to verify the conclusion and document that AI was used to assist the analysis.

C) Delete the AI output to preserve independence.

D) Re-run the same prompt until the answer changes.

Answer: B) Review the underlying evidence to verify the conclusion and document that AI was used to assist the analysis.

Explanation: AI accelerates the work but the auditor retains review responsibility. The conclusion must be verified against the underlying policies, and the use of AI must be documented to preserve independence and a clear review trail. Treating the summary as conclusive, or hiding that AI was used, both fail the guardrails.

DOMAIN 6: EVIDENCE EVALUATION IN A GRC-ENGINEERED ORG

10% of Exam, Approximately 5 Questions

What You Need to Know

In an automated environment, evidence is generated by systems rather than assembled by people. This domain covers what makes that evidence good, how to sample it, a real tool for collecting it, and how to document it in a workpaper.

6.1 Evidence Quality in Automated Environments

Good evidence is authentic, has integrity, and is complete. System-generated evidence (logs, config exports, signed artifacts) is generally stronger than human-generated evidence (screenshots, narratives) because it is harder to alter and easier to attribute. Recognizing this hierarchy is the core skill of evidence evaluation.

6.2 Sampling Strategies for Continuously-Generated Evidence

When evidence is produced continuously, classic statistical and judgmental sampling still apply, but you can also use anomaly-based sampling and, where feasible, complete-population testing. Choose the strategy that matches how the evidence is generated and the assurance you need.

6.3 AWS Audit Playbook Walkthrough

The AWS Audit Playbook by AJ Dehn (AuditOps.io), tagline AWS audits without screenshots, is an open-source Python project that uses boto3 to collect AWS evidence as JSON and generate machine-readable and PDF reports. It is a working example of how modern evidence collection should look. You learn to run it against a sample environment and interpret the output in a real audit.

6.4 Workpaper Documentation for Engineered Controls

Document an automated control test by referencing the actual artifacts: the IaC files, the build log, the JSON output, and any hash or signature attestations, with the date captured. Avoid the common anti-patterns: pasting a screenshot with no source, or describing a control without linking the evidence that proves it operates.

Sample Study Question, Domain 6

Question: For a change-management control, an auditor can obtain either a screenshot of a ticket or the signed CI/CD build log showing the change was tested and approved before deploy. Which is the stronger evidence, and why?

A) The screenshot, because it is easier to read.

B) The build log, because it is system-generated, time-stamped, and attributable, giving it higher integrity and authenticity.

C) They are equivalent.

D) Neither can be used as audit evidence.

Answer: B) The build log, because it is system-generated, time-stamped, and attributable, giving it higher integrity and authenticity.

Explanation: System-generated evidence such as a signed build log is harder to alter and easier to attribute than a screenshot, so it ranks higher on evidence-quality criteria (authenticity, integrity, completeness). Preferring system-generated evidence is the central habit of auditing a GRC-engineered org.

DOMAIN 7: APPLIED COMMERCIAL AUDIT SCENARIOS

10% of Exam, Approximately 5 Questions

What You Need to Know

This domain ties the toolkit together with end-to-end walkthroughs of the engagements commercial auditors actually run: a SOC 2 Type II of a cloud-native SaaS, ISO 27001 in a cloud-native environment, and PCI DSS scoping in the cloud.

7.1 SOC 2 Type II of a Cloud-Native SaaS

Walk through scoping, control selection, and test procedures for a typical cloud-native SaaS, using IaC and CI/CD evidence in place of screenshots. The full CGE-AUD toolkit applies: read the Terraform for control configuration, the pipeline for change management and SoD, and the logs for access and monitoring.

7.2 ISO 27001 in Cloud-Native Environments

Map ISO 27001 Annex A controls to cloud-native evidence and scope the ISMS for a cloud environment. Many Annex A controls have a direct cloud-native artifact: access control to IAM policies, logging and monitoring to cloud audit logs, change management to the pipeline.

7.3 PCI DSS Scoping in the Cloud

Scope a PCI DSS assessment in the cloud: identify the cardholder data environment (CDE), evaluate segmentation, and apply shared responsibility. Network segmentation evidence in the cloud comes from security groups, network ACLs, and routing configuration. Watch for common cloud PCI pitfalls such as over-broad scope and unproven segmentation.

Sample Study Question, Domain 7

Question: During a PCI DSS assessment of a cloud workload, management claims the analytics environment is out of scope because it is segmented from the cardholder data environment. What evidence best supports or refutes that claim?

A) A verbal confirmation from the DevOps lead.

B) The security group, network ACL, and routing configuration showing whether traffic can actually flow between the two environments.

C) A screenshot of the cloud console home page.

D) The company's marketing website.

Answer: B) The security group, network ACL, and routing configuration showing whether traffic can actually flow between the two environments.

Explanation: Segmentation in the cloud is implemented in network configuration. The auditor evaluates the actual security groups, network ACLs, and routes to confirm the analytics environment cannot reach the CDE. Configuration evidence proves or disproves segmentation; a verbal claim or a console screenshot does not.

FRAMEWORKS IN THE CLOUD: QUICK REFERENCE

You already know these frameworks. The CGE-AUD value is mapping their controls to cloud-native evidence. Use this as a quick reference for Domain 7 and the scenario questions.

SOC 2 Trust Services Criteria

Common Criteria that map cleanly to cloud-native evidence: **CC6.1** logical access (IAM policies, identity configuration); **CC6.6** boundary protection (security groups, network ACLs); **CC6.7** transmission security (TLS configuration); **CC7.2** system monitoring (CloudTrail, cloud audit logs); **CC8.1** change management (CI/CD pipeline and pull request history).

ISO 27001 Annex A

Representative Annex A themes and their cloud-native artifacts: access control to IAM and identity configuration; logging and monitoring to cloud audit logs; operations security and change management to the pipeline; cryptography to encryption-at-rest and in-transit configuration in IaC.

PCI DSS in the Cloud

Key cloud scoping concepts: the cardholder data environment (CDE) and what brings a resource into scope; segmentation evidence from security groups, network ACLs, and routing; shared responsibility for who secures what; and common pitfalls such as over-broad scope and segmentation that is claimed but not demonstrated.

Exam Day and Next Steps

Read each scenario artifact carefully before answering: the finding is usually visible in the snippet. Manage time across 50 questions in 75 minutes. After you pass, keep the credential active through an active GRC Engineering Club membership (auto-renewal) or 15 CEU hours across the three-year cycle through Club events, published writing, open-source audit tooling, or mentoring new CGE-AUD candidates.

Website: www.grcengclub.com **Email:** academy@grcengclub.com **Community:** GRC Engineering Club (Patreon)