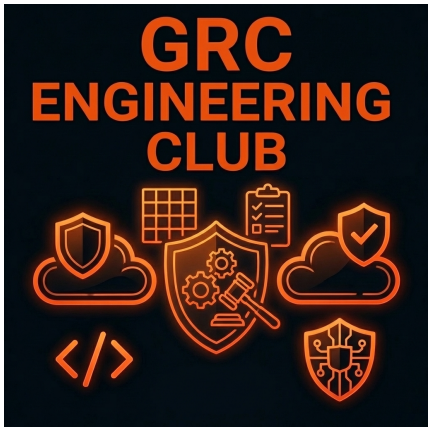

CGE-AZ

Study Guide

Certified GRC Engineer, Azure Specialty

GRC Engineering Club Training Academy

Instructor: AJ Yawn, author of GRC Engineering for AWS

Version 1.0 | September 2026 | Launches September 18, 2026

www.grcengclub.com

HOW TO USE THIS STUDY GUIDE

Overview of the Certification

The Certified GRC Engineer, Azure Specialty (CGE-AZ) is the first cloud specialty stacked on CGE-P. It takes an engineer who can already write Terraform, encode policy, and defend evidence to an auditor, and teaches them to do it on Azure by building one thing: a complete, automated GRC engineering pipeline in their own subscription. The pipeline discovers what is running, activates what is missing, stores immutable evidence, generates auditor deliverables, narrates them weekly, and remediates non-compliant resources on its own. The one-liner: do not map controls to Azure. Build the pipeline that proves them.

Who Should Take This Exam

GRC engineers, compliance engineers, security engineers, and cloud-savvy auditors who hold CGE-P and work in, or are moving into, Microsoft shops. Most enterprise GRC teams live in Microsoft environments while almost all cloud GRC training starts with AWS and stays there. CGE-AZ closes that gap.

Prerequisites

CGE-P is a hard prerequisite, enforced in the platform. The course assumes you can read and write Terraform, use Git and pull requests, and explain a control objective, because CGE-P taught you all three, and it re-teaches none of it. No Microsoft certification is required.

Course Format and Study Time

About 110 minutes of bite-sized video: 25 lessons averaging under 5 minutes, plus two text articles (00_02 How to Use This Course, 07_04 Certification Exam Guide) and six hands-on labs. Total learner study time is 10 to 14 hours: roughly 1.5 hours of video plus the labs, the capstone build, practice questions, and the exam. The learning happens in the labs, not in the lectures.

The Six Labs

Every lab runs in your own subscription from your fork of the starter repo, github.com/GRCEngClub/cgeaz, and every lab folder has a teardown script. Timing matters: create the Azure free account and start the Defender 30-day trial together, so the $200 credit window and the trial window both cover Labs 2 through 6 and the capstone.

Lab	Name	Hands-on Time	Cost
1	Stand Up Your Azure GRC Sandbox	~30 min	$0; the $10/month budget you create is the guardrail for every later lab
2	Turn It All On	~30 min	$0 during the Defender for Storage 30-day trial; Log Analytics is pennies at lab volume
3	Deploy Your Foundation	~60 min	$0; state storage holds kilobytes, policies are free
4	Evidence Flowing End to End	~60 min	Pennies; Cosmos serverless and consumption Functions bill per use
5	Generate Your First ATO Deliverables	~45 min	Pennies; one more consumption Function App

Lab	Name	Hands-on Time	Cost
6	Close the Loop	~60 min at the keyboard, plus async policy-scan waits	$0; policies, remediation tasks, and Actions on a public fork are free

Recommended Study Timeline

Timeline	Activities	Estimated Hours
Week 1	Domains 1 and 2 (Azure Foundations, The Azure GRC Toolkit) + Labs 1 and 2. Starting Lab 2 starts the Defender trial clock.	3–4 hours
Week 2	Domain 3 (Terraform on Azure) + Lab 3	2–3 hours
Week 3	Domains 4 and 5 (The Evidence Plane, Reporting & Narrative) + Labs 4 and 5	3–4 hours
Week 4	Domains 6 and 7 + Lab 6, capstone build and submission, practice questions, exam	3–4 hours plus capstone

How to Use This Guide with the Video Course

This study guide complements the CGE-AZ video course, taught end to end by AJ Yawn. Each domain chapter below follows the lessons in order and condenses what each one teaches into prep notes. Watch the lesson, do the lab, then use these notes and the practice prompts to check yourself. Domains 1 through 3 get you fluent; Domains 4 through 6 build the pipeline stage by stage; Domain 7 is the capstone.

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)
Capstone	REQUIRED: the pipeline, deployed as code, graded against a published rubric
Retake Policy	14-day wait between attempts; max 3 attempts per 12 months
Format	Computer-based (online proctored)
Price	$350, or free for active Club members

Domain Breakdown and Question Distribution

Domain	Topic	Weight	Questions	Study Focus
1	Azure Foundations	15%	~8	Hierarchy, Entra ID, RBAC, managed identities, shared responsibility
2	The Azure GRC Toolkit	15%	~7	Defender, Policy, Log Analytics + KQL, Functions, Cosmos, Blob
3	Terraform on Azure	15%	~8	azurerm provider, auth paths, remote state, governance as code
4	The Evidence Plane	20%	~10	Discovery, activation, Cosmos + WORM Blob evidence store
5	Reporting & Narrative	15%	~7	SAR, POA&M, OSCAL SSP from Cosmos only; AI digest guardrails
6	Enforcement & Operations	10%	~5	Auto-remediation, least-privilege identity, OPA gate, drift
7	Capstone	10%	~5	Requirements, rubric, failure modes, submission

The Capstone Is Required

CGE-AZ certifies builders, so the credential takes both parts: the exam and a graded capstone. The capstone is the pipeline itself: foundation, evidence plane, at least two reports generated from the evidence store only, and enforcement in dry-run, deployed as code in your own subscription and submitted as a public repo link. It is graded automatically against five published dimensions, each weighted 20%, with a pass at a weighted average of 70 or higher and no auto-fail triggers. You get 2 attempts with a 24-hour cooldown. Read the CGE-AZ Capstone Overview and the published rubric at cert.grcengclub.com/rubric/cge-az before you build. If you did the six labs, the capstone is assembly, not invention.

Scoring and Certification

Exam Score: 70% or higher (35 of 50) passes. Scenario questions score as individual questions, weighted equally with multiple choice.

Certification Awarded: Passing the exam and the capstone earns the Certified GRC Engineer, Azure Specialty (CGE-AZ) credential, valid for 365 days. Renew annually with an active Club membership (auto-renews) or 30 CPE hours including at least 20 Group A.

DOMAIN 1: AZURE FOUNDATIONS

15% of Exam, Approximately 8 Questions

What You Need to Know

Domain 1 gives you the Azure mental model: the hierarchy, the identity system, and the responsibility boundary. The thesis of the whole domain is that the org chart of your cloud is your control architecture. Teams that force the AWS mental model onto Azure scope their compliance programs wrong, sometimes for years.

1.1 The Azure Mental Model (01_01)

Four levels, top to bottom: the tenant (your Entra ID directory, the root of the tree), management groups (your OUs, nestable up to six levels), subscriptions (the closest thing to an AWS account: a billing and isolation boundary), and resource groups (no AWS equivalent; every resource lives in exactly one). Policy and role assignments flow downward from wherever you attach them, so where you attach is a control decision. Scope to management groups, not subscriptions: enterprises run hundreds of subscriptions, and an initiative assigned at the management group means every subscription that joins is born compliant. Two traps: subscription-based scoping breaks at Azure scale, and deleting a resource group deletes everything inside it, which makes that permission a data-destruction control.

1.2 Entra ID, RBAC and Managed Identities (01_02)

Azure has two role systems side by side. Entra directory roles (Global Administrator and friends) govern the directory world: users, apps, authentication. Azure RBAC (Owner, Contributor, Reader, and hundreds of granular built-ins) governs the resource world. Global Administrator has no standing power over resources, and Owner cannot touch the directory: auditing one world tells you nothing about the other. Four principal types can hold access: users, groups (prefer these; roles on individuals breed orphaned access), service principals, and managed identities. Every RBAC assignment is role plus principal plus scope, and az role assignment list turns that blade into timestamped JSON evidence. Control plane and data plane are separate permission sets: Contributor on a storage account still cannot read a blob without a data-plane role like Storage Blob Data Reader. The pipeline runs entirely on managed identities, so there are no credentials to rotate.

1.3 Shared Responsibility and What You Engineer (01_03)

Microsoft's attestations (SOC 2, ISO, FedRAMP) cover the platform: data centers, hosts, hypervisor. Everything you configure on top is yours, and nobody has attested it. Five control categories are always customer-owned on Azure: identity and access, configuration state, log routing and retention, policy guardrails, and evidence. The pipeline exists to discharge them: it discovers and assesses configuration, models least-privilege identity, routes and retains logs, enforces policy, and stages three and four exist for nothing but evidence. Judgment work (vendor reviews, risk acceptance) stays human on purpose.

1.4 Lab 1: Stand Up Your Azure GRC Sandbox (01_04)

Four deliverables that every later lab builds inside: the management group hierarchy (mg-grc → mg-grc-sandbox → your subscription), a tagged resource group (rg-grc-sandbox-dev, tagged from birth because discovery reads tags as inventory metadata), a scoped Reader assignment to a grc-auditors group (role plus principal plus scope, saved as your first evidence artifact), and a $10/month budget with actual and forecast alerts. Boring, predictable names are a control: they make anomalies visible.

Key Terms and Definitions

Tenant: The Microsoft Entra ID directory at the top of the hierarchy; every subscription trusts exactly one.

Management Group: A container for subscriptions, nestable to six levels; the primary scope for policy and access.

Subscription: The billing and isolation boundary; the closest Azure equivalent to an AWS account.

Resource Group: A deployment and lifecycle unit every resource must live in; deleting one deletes its contents.

Entra Directory Role: A role governing the directory world (users, apps, auth), separate from Azure RBAC.

Managed Identity: A service principal Azure creates and rotates itself; no key to leak, commit, or forget.

Control Plane / Data Plane: Managing a resource versus touching its data; separate permission sets.

Shared Responsibility: Microsoft attests the platform; the customer engineers everything they configure.

Practice Prompts (no answers here; the graded bank is separate)

1. A policy initiative must apply to every current and future subscription in a sandbox program. Which scope do you assign it at, and what happens to a subscription that joins later?

2. A user holds Global Administrator. What can they do to a storage account in a subscription, and what assignment would change that answer?

3. An engineer with Contributor on a storage account gets AuthorizationPermissionMismatch reading a blob. Explain why, and name the kind of role that fixes it.

DOMAIN 2: THE AZURE GRC TOOLKIT

15% of Exam, Approximately 7 Questions

What You Need to Know

Azure has around two hundred services; the pipeline runs on six. This domain is the translation table from AWS and the deep dives on the three services you will read artifacts from on the exam: Defender for Cloud, Azure Policy, and Log Analytics with KQL. If a service does not earn a stage in the pipeline, it is not in the course.

2.1 The Services That Run the Pipeline (02_01)

The translation table: Security Hub becomes Microsoft Defender for Cloud (posture engine); AWS Config splits into the Activity Log plus Azure Resource Graph (recording) and Azure Policy (rules and remediation); Lambda becomes Azure Functions; S3 becomes Blob Storage with WORM immutability; and Audit Manager becomes Cosmos DB. That last one is deliberate and is the most important architecture decision in the course: AWS closed Audit Manager to new customers in April 2026, and the lesson is permanent. When you rent your evidence store you rent your schema, retention, and roadmap. Own the database and adding a framework becomes a data change, not a procurement. Supporting players: Log Analytics with KQL for who-did-what-when, and AI Foundry with Communication Services for the weekly digest, on a short leash.

2.2 Defender for Cloud: Discovery and Posture (02_02)

Defender has two halves with two price tags. Foundational CSPM is free and always on: continuous assessment against the Microsoft cloud security benchmark, producing recommendations and a secure score. Defender plans are paid per resource type (Storage, Servers, Containers) and add threat detection and deeper posture; every plan has a 30-day free trial per subscription. The unit of work is the assessment: a rule evaluated continuously against resources, each result carrying a status and timestamp, queryable by API. That is a machine-generated control test, and the collector Function in Domain 4 sweeps exactly these. The regulatory compliance dashboard maps the same assessments to framework controls (assign the NIST CSF 2.0 standard); nothing is re-tested. Collect once, map to every framework. Know the boundary: the dashboard proves configuration state, not processes or risk decisions.

2.3 Azure Policy: The Enforcement Engine (02_03)

A first-party policy engine AWS does not have an equal for. A definition is JSON with an if block (matching resources via aliases, the engine's map of every settable property) and a then block with an effect. It is Rego-shaped: read the if, read the effect, and you know what it does. The effects ladder in escalation order: audit (mark non-compliant, touch nothing; the starting point for every new control), deny (block at the API before the resource exists), modify (fix a property, like stamping a tag), and deployIfNotExists (deploy a missing companion, like diagnostic settings). The remediation effects execute through a managed identity attached to the assignment, which is how the platform answers who fixed it. Definitions group into initiatives; assignments take parameters (audit in dev, deny in prod); exemptions are resource-scoped, time-boxed risk acceptances living in the platform. Effect judgment: audit first, escalate to deny only for clear-cut, low-false-positive requirements, and remember that enforcement you cannot sustain is worse than observation you can.

2.4 Logs and KQL: The Raw Material of Evidence (02_04)

Three log families: the Activity Log (control-plane operations, free, but retention is on you), resource logs (per-resource telemetry, off until diagnostic settings route them), and Entra logs (sign-ins and directory changes). Route what matters to a Log Analytics workspace via diagnostic settings, then ask questions in KQL: where filters, project selects columns, summarize aggregates. The course drills three evidence queries that answer auditor questions directly from the data, and the retention decision is itself a control decision.

2.5 Lab 2: Turn It All On (02_05)

Activation day, and the trial clock starts: enable Defender for Storage inside its 30-day trial, assign the NIST CSF 2.0 standard to the subscription, create the Log Analytics workspace and route the Activity Log into it, seed a storage account for Defender to assess, and pull your first assessment from the API. That API pull is the pipeline's raw material; Domain 4 automates it.

Key Terms and Definitions

Foundational CSPM: The free, always-on half of Defender: continuous assessment, recommendations, secure score.

Defender Plan: A paid, per-resource-type layer adding threat detection; each plan has a 30-day trial per subscription.

Assessment: One rule evaluated against one resource with a status and timestamp; a continuous control test.

Policy Effect: What a matched policy does: audit, deny, modify, or deployIfNotExists.

Alias: The dotted field path a policy uses to reach a resource property.

Initiative: A group of policy definitions assigned as one unit (azurerm_policy_set_definition in code).

Exemption: A resource-scoped, time-boxed policy carve-out with a documented category: a risk acceptance record.

KQL: Kusto Query Language; where, project, and summarize over logs in a Log Analytics workspace.

Practice Prompts (no answers here; the graded bank is separate)

1. A new control has never run in this environment and its false-positive rate is unknown. Which policy effect do you assign first, and what data tells you when to escalate?

2. An assessor asks how many storage accounts allowed public network access last Tuesday. Which pipeline component answers that without touching a live service, and why does the answer hold up?

3. Your team proposes buying a compliance SaaS as the evidence store. Using the Audit Manager story, argue for or against, naming what you give up in each case.

DOMAIN 3: TERRAFORM ON AZURE

15% of Exam, Approximately 8 Questions

What You Need to Know

You already write Terraform; this domain moves you from the aws provider to azurerm and puts governance itself in code. By the end of Lab 3 your sandbox stops being hand-built and starts being governed by code, with a denied deployment as the proof.

3.1 From aws to azurerm: Provider, Auth and State (03_01)

The azurerm provider needs a features block and explicit subscription targeting; pin the provider version (4.x conventions apply). Authentication has one right answer per context: az login for a human at a laptop, OIDC federation for CI (GitHub Actions gets short-lived tokens, no stored secrets), and managed identity for anything running inside Azure. Remote state lives in an Azure storage account with versioning and locking, and the state storage is itself audit-relevant: treat it as a controlled resource, because state can contain sensitive values and its history is change evidence.

3.2 Governance as Code: Policy in Terraform (03_02)

Custom policy definitions ship as azurerm_policy_definition with jsonencode carrying the if/then rule; initiatives are azurerm_policy_set_definition; assignments scope to a management group so inheritance does the distribution. The detail the exam will probe: an assignment carrying a remediation effect (modify or deployIfNotExists) requires an identity block, and that identity needs the roles its fixes require. Policy-as-code beats portal clicks because every control change gets a pull request, a review, and a history.

3.3 Lab 3: Deploy Your Foundation (03_03)

Bootstrap remote state, import the Lab 1 sandbox into Terraform, and deploy the foundation stage from the starter repo: the Log Analytics workspace, the evidence resource group, tag and diagnostic policies as an initiative, and the remediation identity. Then prove enforcement: attempt a deployment the deny policy forbids and watch it fail at the API. Teardown is terraform destroy from here on.

Key Terms and Definitions

features block: The required (even when empty) configuration block on the azurerm provider.

OIDC Federation: CI authenticating with short-lived tokens instead of stored credentials.

Remote State: Terraform state in a versioned, locked storage account; itself a controlled resource.

jsonencode: The Terraform function embedding a policy's JSON rule inside azurerm_policy_definition.

Identity Block: The assignment block that gives remediation effects a managed identity to execute as.

Denied Deployment: The foundation's proof of enforcement: the API refuses a non-compliant resource.

Practice Prompts (no answers here; the graded bank is separate)

1. A pipeline stores a service principal secret in CI variables to run terraform apply. Name the two zero-secret alternatives and match each to the context it belongs in.

2. A plan shows an azurerm_policy_assignment with effect deployIfNotExists and no identity block. What happens at remediation time, and what is the fix?

3. Why does the course treat the state storage account as a controlled resource? Give two concrete risks.

DOMAIN 4: THE EVIDENCE PLANE

20% of Exam, Approximately 10 Questions

What You Need to Know

The heaviest domain on the exam, by design: stages 1 through 3 of the pipeline are where evidence is born. Discovery interrogates before anything acts, activation enables only what is missing, and the evidence store holds every finding once, immutably, with lineage. By the end of Lab 4 you can trace a single finding from the Defender portal to an immutable stored record.

4.1 Stage 1, Discovery: Detect What's Running (04_01)

Discover before you act. Terraform data sources interrogate the environment (subscriptions, Defender plan states, workspace presence) and Azure Resource Graph answers estate-wide questions. Discovery emits a typed output contract: an inventory of what exists and a gap map of what is missing. Because discovery only reads, re-runs are safe and idempotent, and downstream stages consume its outputs instead of assuming.

4.2 Stage 2, Activation: Enable What's Missing (04_02)

Activation is conditional by construction: for_each over the gap map, so Terraform enables only the Defender plans that are off (azurerm_security_center_subscription_pricing), wires workspace settings, and assigns the CSF 2.0 standard as code. What already exists is respected, not recreated. Unconditional activation is the anti-pattern the rubric checks for.

4.3 Stage 3, The Evidence Store: Cosmos, WORM Blob and Collectors (04_03)

The store you own: Cosmos DB containers for assessments, frameworks, and mappings (the crosswalk to 800-53 lives as data, so collect once holds), plus a WORM-immutable Blob container for report-grade artifacts. The collector is a Python Function on a timer: it calls the Defender assessments API and writes run-stamped documents, with runId and collectedAt on every record and deterministic IDs so re-collection upserts instead of duplicating. Identity wiring follows the data-plane split: the collector gets data-plane write, humans get read, and nobody holds account keys. Be able to defend every component of this design to an assessor.

4.4 Lab 4: Evidence Flowing End to End (04_04)

Deploy stages 02-activation and 03-evidence-store, trigger a collection run, verify documents landed in Cosmos, verify WORM on the reports container, and trace one finding from the portal to its stored record. Practical gotcha the lab validates: consumption (Y1) Functions quota is regional and zero in most US regions on free accounts, so run probe-quota.sh first and set functions_location accordingly. The lab ends with a cost checkpoint against the Lab 1 budget.

Key Terms and Definitions

Data Source: A Terraform block that reads existing state instead of creating it; discovery's engine.

Azure Resource Graph: The estate-wide query service discovery uses for inventory questions.

Gap Map: Discovery's output contract: what is missing, consumed conditionally by activation.

Collector Function: The timer-triggered Function that sweeps the assessments API into Cosmos.

WORM: Write once, read many: the immutability policy on the artifacts container.

Lineage: runId and collectedAt stamped on every document, making each record traceable to a run.

Idempotent Collection: Deterministic document IDs and upserts, so re-runs never duplicate evidence.

Collect Once: One assessment sweep feeding every framework through the crosswalk data.

Practice Prompts (no answers here; the graded bank is separate)

1. A reviewer finds azurerm_security_center_subscription_pricing resources with no reference to discovery outputs. Which rubric expectation does this violate, and what breaks when the pipeline runs against an already-configured subscription?

2. A collector writes documents with random UUIDs each run. What goes wrong over time, and which two fields plus what ID strategy fix it?

3. An assessor asks you to prove a finding in last month's report was not altered after collection. Walk the trace: which stores, which properties, and which failed operation make the case?

DOMAIN 5: REPORTING & NARRATIVE

15% of Exam, Approximately 7 Questions

What You Need to Know

Stages 4 and 5 turn stored evidence into the deliverables an assessor actually wants, plus a weekly narrative a CISO actually reads. The load-bearing rules: reports read from Cosmos only, and AI describes, never decides.

5.1 Stage 4, Reporting: ATO Deliverables from Cosmos Only (05_01)

Six reports, each answering a question for a reader: the SAR (what is our assessed posture), the POA&M (what is broken and when will it be fixed), the framework report (how do we score against a given framework), the OSCAL 1.1.2 SSP (the machine-readable system security plan), the weekly digest (what changed this week, in prose), and the ATO package (the bundle). Report Functions run on schedules and read from the evidence store only, never from live platform APIs. That rule is what makes every report number reproducible: any figure traces to a stored query over stored documents.

5.2 Lab 5: Generate Your First ATO Deliverables (05_02)

Deploy the reporting stage and run the POA&M and SAR generators against your own evidence. Note the reporter identity's whitelist: Cosmos read plus Blob write, no Security Reader, no Cosmos write; separation of duties enforced by role scopes. Verify WORM storage and dated paths, then complete the auditor's test on your own pipeline: pick a number in the report and trace it back to its Cosmos document.

5.3 Stage 5, AI Narrative: Describe, Never Decide (05_03)

The narrative layer turns report JSON into prose: an AI Foundry model deployment, a digest Function (report JSON in, digest out), and Communication Services for email delivery. The guardrails are the point: AI is additive, never load-bearing; prompts are grounded in the report data; inputs are human-auditable; and no model touches a finding, a score, or a remediation decision. This stage is walkthrough-only in the course and is never graded in the capstone.

Key Terms and Definitions

SAR: Security Assessment Report: assessed posture, generated from stored evidence.

POA&M: Plan of Action and Milestones: open weaknesses with remediation plans and dates.

OSCAL SSP: The machine-readable System Security Plan, OSCAL version 1.1.2.

Cosmos-Only Rule: Report generators read the evidence store, never live platform APIs.

Number-to-Source Trace: Following a report figure back to the stored documents that produced it.

Describe, Never Decide: The AI guardrail: models write prose about established facts, and nothing else.

Practice Prompts (no answers here; the graded bank is separate)

1. A teammate speeds up the SAR generator by querying the Defender API directly at render time. The numbers are fresher. What did the change break, and how would an assessor catch it?

2. Match each reader to the deliverable they ask for first: a federal authorizing official, a remediation-tracking engineering lead, and a CISO skimming weekly. Justify each.

3. A proposal adds AI-based prioritization that reorders POA&M items by predicted risk. Does this pass the narrative-stage guardrails? Say precisely which line it crosses or why it does not.

DOMAIN 6: ENFORCEMENT & OPERATIONS

10% of Exam, Approximately 5 Questions

What You Need to Know

Stage 6 closes the loop: from observing non-compliance to fixing it, through an identity you can audit, with a human at the gate, and with the fix documenting itself in the next collection. Operations keeps the pipeline honest: a CI gate on its own repo and drift detection watching both the code and the people.

6.1 Stage 6, Enforcement: Close the Loop (06_01)

Remediation at scale runs through Azure Policy's remediation effects, and every fix executes as one dedicated, user-assigned, least-privilege identity whose roles are a whitelist of its job, never Owner or Contributor. Escalation is a ladder with humans on every rung: audit first, then dry-run (remediation tasks proposed, human approves), then enforce, and each escalation is a reviewed parameter change in a pull request. The Activity Log records the remediation identity's actions, so who fixed it has an auditable answer, and the next collection run picks up the corrected state: detected, fixed, re-collected, documented.

6.2 Lab 6: Close the Loop (06_02)

You break your sandbox on purpose. Deploy enforcement in dry-run, enable the CI gate on your fork, introduce a deliberate violation, approve the proposed remediation task, and watch the fix flow into the next collection and the reports, with drift detection firing along the way. A design note the lab teaches in reverse: the foundation's deny policy makes the obvious sabotage impossible (deny fires on updates too), so you de-escalate deny to audit through a parameter change, sabotage, remediate, and re-escalate. Expect asynchronous waits; policy compliance scans ran about 15 minutes in validation, and re-running commands does not speed Azure up.

Key Terms and Definitions

Remediation Task: The unit of policy-driven fixing: proposed by the platform, approved by a human in dry-run.

Remediation Identity: The dedicated user-assigned identity every fix executes as; a whitelist of its job.

Escalation Ladder: audit → dry-run → enforce, each step a reviewed change with a human approving.

CI Gate: conftest/OPA evaluating the repo's own Terraform plans on every pull request, failing closed.

Drift Detection: Two questions: does reality match code (scheduled plan) and who is touching reality (activity-log tripwire).

Practice Prompts (no answers here; the graded bank is separate)

1. A remediation identity holds Contributor at subscription scope because it was easier. List what an auditor concludes, and rewrite the design in one sentence.

2. Your scheduled drift plan is clean, but a resource changed yesterday. Which drift question did the scheduled plan miss, and which component catches it?

3. Describe the evidence trail one approved remediation leaves behind, from proposal to the report that shows the resource healthy again.

DOMAIN 7: CAPSTONE

10% of Exam, Approximately 5 Questions

What You Need to Know

The capstone is the credential's second half and it is required: your pipeline, in your subscription, from your repo, graded automatically against a rubric published from day one. Domain 7's exam questions test whether you know what the rubric demands and how graders check it.

7.1 Capstone Brief: Ship Your Pipeline (07_01)

The requirements: the governed foundation, the evidence plane (discovery, activation, Cosmos plus WORM Blob store), at least two reports generated from the evidence store only with real run history, and enforcement in dry-run, all deployed as code from a public repo that starts from github.com/GRCEngClub/cgeaz. Five dimensions, each 20%: IaC Quality, Control Implementation and Identity Design, Evidence Integrity and Traceability, Pipeline Operations, and Documentation and Control Mapping. Pass is a weighted average of 70 or higher with no auto-fail triggers. Auto-fails: a private or inaccessible repo, active secrets, real PII or production credentials, a trivially modified fork of the reference, or no README. Grading is automated with 2 attempts and a 24-hour cooldown, and every score returns per-dimension results with quoted file evidence.

7.2 Capstone Walkthrough: The Reference Build (07_02)

A complete passing capstone toured at review pace: repo structure, the modifications that show comprehension, and the rubric checks run live: plan cleanliness, identity scopes, the number-to-source trace, the gate test, and the docs. What distinguishes strong from passing: run history that accumulated over time rather than a burst the night before, a gate proven to block with a closed test PR, and a CONTROLS.md that matches the code.

7.3 Course Wrap-Up and What's Next (07_03)

The six stages recapped as one system, and the pattern taken to work: brownfield first steps start with discovery, because discovery is safe to run anywhere. The credential says you build and operate the loop: discovered, activated, stored, reported, narrated, enforced.

7.4 Certification Exam Guide (07_04, article)

Logistics: scheduling, domain weight recap with question counts, scenario strategy (read the artifact first; the gap is usually visible in it), capstone submission timing relative to the exam and the Defender trial window, and the retake and renewal policies covered in the Exam Overview of this guide.

Key Terms and Definitions

Rubric Dimension: One of five published grading axes, each weighted 20%.

Auto-Fail Trigger: A condition that fails a submission regardless of score, such as active secrets or no README.

Run History: Accumulated timer and workflow executions proving the pipeline lives; graders check for bursts.

Reference Build: The complete passing capstone toured in 07_02; calibrate against it, do not copy it.

CONTROLS.md: The mapping from every policy, collector, and gate rule to NIST CSF 2.0 categories.

Practice Prompts (no answers here; the graded bank is separate)

1. A submission scores 90, 85, 80, 75, 72 across the five dimensions but the repo's Actions history shows every workflow ran once, manually, the day before submission. Which dimension takes the hit, and can the submission still pass?

2. List three auto-fail triggers and, for each, the pre-submission check that catches it in under five minutes.

3. Your capstone is ready but your Defender trial expires in four days and your exam is in two weeks. Order your submissions and defend the timing.

EXAM DAY AND NEXT STEPS

Read each scenario artifact before the answer options: the gap is usually visible in the policy definition, KQL query, Terraform plan, pipeline log, or Defender assessment in front of you. Manage time across 50 questions in 75 minutes. Submit the capstone while your pipeline's run history is live and your Defender trial window is open.

After you pass both parts, the credential is valid for 365 days. Renew annually through an active GRC Engineering Club membership (auto-renews) or 30 CPE hours including at least 20 Group A. Then take the pattern to work: start brownfield engagements with discovery, and build from there.

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