# Roadmap - Cloud Security

**Goals and Intentions:**

- **Project-First, Non-Linear Learning:** I am of the belief that building—rather than consuming lectures—keeps people engaged. Each lab should produce an artifact (a GitHub repo, a Terraform deployment, a penetration-demo write-up) that you all can reference as you go along.

  Even if it's a small lab, it's *something to show* and the important thing should be that feeling that you're creating something or protecting something.
- **Introducing progressive complexity over time with some real-world context:** We'll begin with simple constructs (an EC2 + security group) and layer in complexity (IAM roles, RDS, ALB, Kubernetes, DAST/CSPM/SAST tooling) over time.

  At each step, I hope we can tie the technical work back to a business use case or security program requirement (e.g., “Why would an organization enforce IAM roles on servers?”). I hope you will start to cultivate this viewpoint on all the infrastructure you learn about.
- **Portfolio-Centric Approach:** Every lab should result in a tangible deliverable: Terraform code, annotated screenshots, vulnerability scan reports, or even just short write-ups explaining security trade-offs and build notes on the labs.

  By the end of six months, each one of you who stuck with it, should have a GitHub repository (or several) showcasing your journey from “spin up an EC2” to “deploy a containerized app behind an ALB with CSPM scanning enabled,” complete with README.md files explaining what you did and why.

  *As we progress through each lab, encouragement and (time providing) support will be given to you for a Python script to come from it and/or a creative bonus stretch to add a more advanced artifact to your portfolio that showcases more scripting or automation.
- **Discussion of Politics & Processes:** After you build something, pause to talk about the real-world constraints: what happens when a Lead or Head of Engineering refuses to track low-severity CSPM findings with an audit coming up? How do you approach DevSecOps champions internally? Who do you need buy-in from to introduce Semgrep or a CSPM tool in a mid-market startup? We'll try to develop a simple stakeholder-mapping table (e.g., “Position → Concerns → Ask → Win Condition”) to begin understanding this and develop it as we go.

## Scope of the mentorship:

This program is designed to immerse you all (mostly beginners and some career shifters) into real-world AWS cloud security practices rather than simply preparing for certification exams. Over the next six months, you'll **build and secure AWS environments** using predominantly **Linux-based VMs** (no Windows), leveraging **limited Bash scripting to learn foundational Linux** administration and Terraform and Python for automation and security tooling.

We'll emphasize how **Cloud Security integrates with DevOps workflows**—showing you, for example, how to embed security controls into **CI/CD pipelines** and I**nfrastructure as Code** (Terraform). By focusing on hands-on projects (EC2 servers, IAM roles, VPCs, RDS, container or Kubernetes deployments, and open-source scanning tools) that mirror entry-level roles, you'll develop **the practical experience hiring managers actually look for**. My goal and hope is to position you for junior or **internship-level IT Cloud Support, Jr. Cloud Security and/or Jr. DevSecOps opportunities**—with a GitHub based portfolio and creative security strategic skills that I think will show employers real AWS expertise matters more than a credential alone.

Please remember, I am working full time. I live in Europe, so I may not be able to respond right away to questions, and I am doing this because I believe that when most people who are struggling with career shifts or just trying to find a new career, it can be scary and frustrating. I went through it and I wouldn't have landed if it weren't for a few people mentoring me. So I'm paying it back. The community for mentorship should also depend on all of you mentoring one another, whether it be sharing what you learned or interview tips, etc.. This will also give you skills companies look for: people with investment in themselves and their teams.

**I do not guarantee anything from this experiment.** I'm simply trying something new. Just like you are!

---

# Six-Month Roadmap (Weekly Labs + Skill Milestones)

Below is a month-by-month breakdown. Each week (or so, let's be realistic) is one lab/project. We can adjust pace according to how you all are keeping up or feeling. It's not school. It's mentorship!

**Feedback welcome** via Discord DM or my email (bprofane@gmail.com). I'll be putting this into an online tool one of you all showed me earlier (https://roadmap.sh/). We'll give that a try and see if it generates something more easy to comprehend then my PDFs. This is a first draft (preview).

# Month 1: Foundations in AWS & Infrastructure as Code

1. **Lab 1: AWS Account Setup & “Hello, EC2”**
   - *Task:* You will create or connect to an AWS Free Tier account (or LocalStack if no AWS). Launch a basic EC2 instance (Amazon Linux 2) via the console.
   - *Deliverable:* Screenshot of running EC2 with public IP + brief README.md explaining what an EC2 is.
   - *Concepts:* Cloud fundamentals, AWS console navigation, Linux basics.
2. **Lab 2: Secure the EC2 – Security Groups & SSH Access**
   - *Task:* Create a new security group that allows only SSH (port 22) from your own IP. Confirm by attempting an SSH connection from another IP (e.g., from mobile hotspot blocked).
   - *Deliverable:* Terraform (or CloudFormation) code defining the security group + a short write-up explaining ingress/egress rules and why restricting SSH is critical.
   - *Concepts:* Network security groups, principle of least privilege, Terraform basics.
3. **Lab 3: IAM Roles & Permissions for EC2**
   - *Task:* Create an IAM role that allows read-only S3 access. Attach it to the EC2. Demonstrate via SSH that the instance can run `aws s3 ls` on a public bucket without storing credentials.
   - *Deliverable:* Terraform code for IAM role + instance profile + README explaining IAM role vs. IAM user.
   - *Concepts:* IAM basics, instance profiles, AWS credentials best practices.
4. **Lab 4: Version Control & GitHub Integration** - You will be learning about GitHub best practices and basic versioning commands. You will build on these going forward.
   - *Task:* You will initialize a Git repository, push your Terraform code and documentation to GitHub. You should open a pull request (PR) for review.
   - *Deliverable:* A GitHub repository URL, at least one merged PR, basic `.gitignore` and `README.md`.
   - *Concepts:* Git commit/push/PR workflow, collaborative code review, importance of version control in Security Engineering.
5. **Bonus**: Build a web server on your EC2 or upload a CV static website to S3 bucket to begin charting your journey for potential employers. This can be a demo day project to recap what you've learned.

# Month 2: Expanding the AWS Environment & Baseline Security Scanning

1. **Lab 5: Build an RDS Database (MySQL) within VPC**

- *Task:* Using Terraform, provision a private RDS MySQL 8.0 instance in the same VPC. Restrict access so only your EC2 (from Month 1) can connect on port 3306.
- *Deliverable:* Terraform module for VPC, subnets, security group, and RDS instance + screenshot of a successful `mysql -h <rds-endpoint>` from the EC2 instance.
- *Concepts:* VPC networking, subnets, private vs. public subnets, multi-AZ basics (optional).

2. **Lab 6: Introduce an Open-Source CSPM Tool (e.g., Prowler or ScoutSuite)**
   - *Task:* You will run a baseline scan of your AWS account using Prowler (or ScoutSuite) to identify critical/high misconfigurations (e.g., publicly open S3 buckets, IAM policy issues).
   - *Deliverable:* PDF/HTML report from Prowler, plus a bullet-point summary in yourrepo of top 3 critical/high findings and remediation steps.
   - *Concepts:* Continuous security monitoring, CIS AWS Foundations Benchmark, interpreting CSPM output, compliance baselines.
3. **Lab 7: Triage Critical/High Findings & Logging**
   - *Task:* For each high/critical finding, write a brief triage note: “What is the risk? How would you remediate? Who in an organization would you call to fix it?” Additionally, enable CloudTrail logging & send logs to an S3 bucket.
   - *Deliverable:* A `vulnerability_triage.md` document listing findings + proposed remediation + stakeholder. Screenshot of CloudTrail trails enabled.
   - *Concepts:* Risk analysis, stakeholder communication, audit logging best practices.
4. **Lab 8: Hands-On “Red Team Light”: S3 Bucket Enumeration & Subdomain Takeover Demo**
   - *Task:* You will create an S3 bucket named `<your-username>-bprofanelab-bucket`, intentionally misconfigure bucket ACL (make it public) in Terraform. Then use AWS CLI or a simple Python script to enumerate the bucket. Demonstrate how a “bad actor” could read/write to it. For bonus, register a subdomain in Route 53 pointing to a nonexistent bucket to demonstrate takeover.
   - *Deliverable:* Terraform code + a 1-minute screencast of bucket enumeration + short blog-style write-up: “How S3 misconfig leads to subdomain takeover.”
   - *Concepts:* S3 security best practices (Block Public Access, bucket policy vs. ACL), domain services, basic offensive cloud security.
5. Lab 9: BONUS lab - SIEM research and creation
   * We need a SIEM at this point and let's have you all bring your own suggestions to your projects. It'll be a demo day project.

## Month 3: Load Balancers, Auto Scaling & Infrastructure Hardening

1. **Lab 9: Deploy a Simple Web App Behind an ALB/ELB**
    - *Task:* Using Terraform, deploy a small “Hello, World” Node.js (or NGINX) application on the EC2 instance(s). Create an Application Load Balancer (ALB) in front, public-facing. Confirm via browser.
    - *Deliverable:* Terraform code, architecture diagram (hand-drawn or drawn in Lucidchart/diagrams.net), and a screenshot of the working ALB endpoint.
    - *Concepts:* ALB vs. NLB, listeners, target groups, health checks, auto scaling group (ASG) basics if desired.
2. **Lab 10: Harden the ALB & EC2 Instances**
    - *Task:*
        - a) Configure an AWS WAF Web ACL to block basic OWASP Top 10 threats (e.g., SQLi, XSS) with pre-built rules.
        - b) Enable EC2 instance’s SSH only from the ALB’s security group (for internal troubleshooting).
    - *Deliverable:* Terraform code for WAF ACL + security group rules and a brief write-up explaining how WAF protects web apps.
    - *Concepts:* WAF fundamentals, managed rule groups, network segmentation.
3. **Lab 11: CSPM “Continuous” Integration with GitHub Actions**
    - *Task:* We'll add a GitHub Actions workflow that runs Prowler/ScoutSuite scan against the Terraform-built environment on every commit/PR. If any critical/high findings appear, fail the job.
    - *Deliverable:* `.github/workflows/cspm.yml` with steps to authenticate to AWS, run Prowler, and parse exit codes. Screenshot of an intentional failure (e.g., open S3 bucket) causing a failed check.
    - *Concepts:* CI/CD integration, “shift-left” security, automating compliance scans.
4. **Lab 12: Identity & Access Management Deep Dive (IAM Policies & SCPs)**
    - *Task:*
        - a) Create a custom IAM policy that only allows `DescribeInstances` and `StartInstances` on EC2.
        - b) Attach it to an IAM group (e.g., “JuniorSecEng”).
        - c) Demo an AWS Organizations Service Control Policy (SCP) blocking the creation of public S3 buckets.
    - *Deliverable:* JSON documents for IAM policy and SCP + a short explanation in your repo: “How do IAM policy vs. SCP interact?”
    - *Concepts:* IAM policy syntax, policy simulator, Organizations and SCP layering.

# Month 4: Containers & Kubernetes on AWS

1. **Lab 13: ECS Fargate or EKS Baseline Deployment**

- *Task:*
  - Option A (ECS): Deploy the same “Hello” app as a Docker container on ECS Fargate behind an ALB.
  - Option B (EKS): Deploy a managed EKS cluster with a node group (or use AWS Fargate profiles), and deploy the “Hello” container as a Deployment + Service.
- *Deliverable:* Terraform/EKS YAML manifests + architecture diagram + screenshot of `kubectl get pods` /service endpoint working.
- *Concepts:* Container orchestration basics, ECS vs. EKS, container security considerations.

2. **Lab 14: Kubernetes Hardening & CIS Benchmark**
   - *Task:* Run the open-source kube-bench scanning the EKS cluster for CIS Kubernetes Benchmark compliance. Review critical/high findings, and remediate at least two (e.g., ensure RBAC is enabled, enforce PodSecurity).
   - *Deliverable:* kube-bench JSON/HTML output + a remediation plan in markdown.
   - *Concepts:* Kubernetes security best practices, CIS Benchmarks, RBAC, PodSecurity Standards.
3. **Lab 15: Infrastructure as Code for Kubernetes**
   - *Task:* Convert the EKS deployments/manifests into a Helm chart or Terraform Kubernetes provider module. If you used ECS, convert the ECS task definitions/service into Terraform modules.
   - *Deliverable:* A reusable Helm chart (with `values.yaml` and `Chart.yaml`) or a Terraform module folder with examples.
   - *Concepts:* DRY principles, templating, Semantic Versioning of modules/charts.
4. **Lab 16: DAST Scanning with OWASP ZAP Against a Deployed App**
   - *Task:*
     - a) Deploy OWASP ZAP as a Docker container locally or on an EC2.
     - b) Run an authenticated or unauthenticated DAST scan against your ECS/EKS “Hello” app.
     - c) Review findings; classify true positives vs. false positives.
   - *Deliverable:* ZAP scan report (HTML) + a short post-mortem: “Which findings were real? How would you triage/close them?”
   - *Concepts:* DAST fundamentals, false positives vs. true positives, remediation workflows.

## Month 5: Securing CI/CD & Application Security

1. **Lab 17: Integrate Semgrep for SAST in a GitHub Workflow**
   - *Task:*
     - a) Choose a basic Node.js application (or Python/Go) with known security bugs.

    - b) Add a Semgrep GitHub Action to scan PRs. Configure rules to catch at least two OWASP Top 10 code issues (e.g., SQL injection in a simple Python example).
  - *Deliverable:* Semgrep config file ( `.semgrep.yml` ) + GitHub Actions workflow + screenshot of a PR failing due to detected code smell.
  - *Concepts:* SAST vs. DAST, rule precision vs. noise, integrating SAST early.
2. **Lab 18: Discussion & Stakeholder Mapping for Security Tool Adoption**
  - *Task:* Each one of you picks either CSPM or SAST integration you implemented. Then write a short “Stakeholder Outreach Plan”:
    - Who are the key decision-makers (CTO, Dev Manager, CISO)?
    - What are your pain points? (e.g., “CTO fears slowdowns in release cycle.”)
    - How would you position Semgrep/Prowler so adoption seems to accelerate, not hinder, development?
  - *Deliverable:* A 1-page PDF or markdown “Stakeholder Outreach Plan” for your chosen tool.
  - *Concepts:* Organizational buy-in, ROI arguments, balancing security vs. speed.
3. **Lab 19: Container Image Scanning & Supply Chain Security**
  - *Task:*
    - a) Build a Docker image for your web app & push it to an ECR registry.
    - b) Use an open-source scanner (e.g., Trivy or Clair) to scan the image. Identify high/critical CVEs.
    - c) Demonstrate patching: update base image to mitigate at least two vulnerabilities, rebuild, and re-scan.
  - *Deliverable:* Dockerfile + Trivy scan output before/after + short summary of patch strategy.
  - *Concepts:* Image hardening, CVE triage, minimum permissible privileges in container.
4. **Lab 20: Cross-Account Roles & Multi-Account Strategy**
  - *Task:*
    - a) Simulate a multi-account AWS setup (e.g., a “Dev” account and a “Prod” account using separate AWS profiles or IAM credentials).
    - b) Create an IAM role in Prod that can be assumed by a principal in Dev. From Dev’s CLI, assume the role and perform a restricted action (e.g., list Prod’s S3 buckets).
  - *Deliverable:* IAM role/trust policy JSON + demo commands + explanatory doc: “Why multi-account?”
  - *Concepts:* Least privilege across accounts, AWS Organizations, trust policies.

## Month 6: Advanced Topics & Capstone Project Kickoff

1. **Lab 21: AWS Security Hub & Aggregated Findings**
    - *Task:* Enable AWS Security Hub in your primary AWS account. Integrate findings from CSPM (Prowler) and from Container Scanning (Trivy) into Security Hub.
    - *Deliverable:* Screenshot of Security Hub dashboard showing findings from multiple sources + a “mapping doc”: “Which findings came from Prowler? Which from Trivy?”
    - *Concepts:* Centralized finding aggregation, Security Hub standards, subscriptions.
2. **Lab 22: Introducing Drift Detection & Automated Remediation (Optional)**
    - *Task:* Using AWS Config, enable a rule (e.g., `restricted-common-ports`) to detect if any security group allows 0.0.0.0/0 on port 22. Configure an AWS Lambda that auto-remediates by removing that rule.
    - *Deliverable:* AWS Config rule JSON, Lambda function code, and a test where a misconfigured SG is auto-fixed.
    - *Concepts:* Infrastructure drift, policy-as-code, automated remediation, Lambda basics.
3. **Lab 23: Capstone Project – Project Definition & Team Formation**
    - *Task:* You will propose 1–2 capstone ideas in a shared spreadsheet. Examples:
        - “Deploy a complete three-tier web application with auto-scaling front end, RDS backend, ECS/EKS mid-tier, all secured with WAF, CIS Benchmarks enforced, SAST/DAST integrated, and a vulnerability management pipeline that sends alerts to Slack.”
        - “Simulate a multi-tenant SaaS environment with multiple VPCs, cross-account IAM, and demonstrate how a CSPM baseline could continuously detect misconfigurations.”
    - *Deliverable:* Capstone proposal doc (1-page outline with objectives, tech stack, expected deliverables) + grouping into small teams of 2–3.
    - *Concepts:* Full-stack cloud security, project scoping, teamwork.
4. **Lab 24: Capstone Project Kickoff**
    - *Task:* Form teams, decide on roles (Terraform engineer, Kubernetes lead, DevSecOps pipeline lead, documentation lead). Create a GitHub org/repository for the project. Define milestones for Month 7–9.
    - *Deliverable:* Team charter (roles, communication channels, timeline), GitHub org URL, initial commit with empty Terraform scaffold.
    - *Concepts:* Agile/Scrum basics (sprints, standups), version control at scale, clear deliverable definition.