DevOps Technical Task

Simulated EKS-Style Secure Deployment Using Minikube

🗂️ Overview

In this technical task, you'll simulate a secure microservices deployment in a local Kubernetes cluster using Minikube. Your focus will be on infrastructure, security, observability, and incident response — modeled after a real-world AWS EKS environment.

You are not required to write application logic; we've provided example container images to use as your services.

This task is not about copying from Stack Overflow — it's about showcasing your ability to reason, secure, and operate modern Kubernetes platforms.

🎯 Objective

> Build and secure a local Kubernetes microservices environment that mimics AWS EKS production patterns, using Minikube, and respond to a simulated production incident involving a security leak and network policy violations.

🛠️ Environment Setup

✅ Requirements:

- Install:
    - Minikube
    - Docker
    - `kubectl`
    - `helm`
    - (Optional) `k9s`, `OPA`, `Kyverno`, `Falco`

✅ Minikube Setup:

Start your cluster:

```
minikube start --cpus=2 --memory=4096 --addons=ingress,metrics-server
```

- Install:
    - NGINX Ingress Controller (via Minikube addon)
    - Prometheus & Grafana (via Helm)

🧱 Part 1: Microservice Stack

Deploy the following three services:

Service	Purpose	Docker Image
`gateway`	API gateway, exposed via ingress	`nginxdemos/hello`
`auth-service`	Auth logic, logs headers	`kennethreitz/httpbin`
`data-service`	Mock business logic	`hashicorp/http-echo`

✅ Requirements:

- Containerized deployment using Helm or Kustomize
- Liveness/readiness probes
- Proper resource requests/limits
- `gateway` should be publicly accessible via ingress
- `auth-service` and `data-service` should be internal-only
- Use separate namespaces for system and application

🛡️ Part 2: Simulate IAM with MinIO

✅ Task:

- Deploy MinIO (`minio/minio`) inside the cluster to act as mock S3
- Create:
    - A `Secret` for MinIO credentials
    - A `ServiceAccount` bound to `data-service` only
- Prove:
    - `data-service` can access the bucket
    - `auth-service` cannot (even if misconfigured)

Bonus: Use OPA or Kyverno to enforce this IAM-like policy.

🚨 Part 3: Security Incident Simulation

📜 Scenario:

> "The `auth-service` has been leaking `Authorization` headers to logs, and it's been discovered that it can reach services it should not — including external endpoints."

✅ Your tasks:

1. Investigate the issue
    - Use `kubectl logs` or `k9s` to find sensitive logs
2. Fix the problem
    - Update the deployment to avoid leaking sensitive headers
    - Apply a NetworkPolicy:
        - Allow `auth-service` to only communicate with `data-service`
        - Block all egress traffic unless explicitly needed
3. Prevent future violations
    - Use OPA, Kyverno, or Falco to:
        - Detect if pods log Authorization headers
        - Alert or block such behavior

> You may simulate logging leaks with HTTPBin by hitting the `/headers` or `/get` endpoint with Authorization headers.

📊 Part 4: Observability

✅ Tasks:

Install Prometheus & Grafana using Helm:

```
Helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm install monitoring prometheus-community/kube-prometheus-stack
```

- Create a Grafana dashboard showing:
    - Pod CPU/memory usage
    - HTTP request rates and errors
    - Pod restarts

> Bonus: Set up an alert (in Prometheus or Grafana) for abnormal restarts or failed probes.

🧪 Part 5: Failure Simulation

Simulate a partial failure:

- Kill a pod or simulate node pressure using `kubectl delete pod`
- Ensure system:
    - Recovers via ReplicaSets or HPA
    - Is observable during the event (e.g., spikes in Grafana)

📦 Deliverables

Please submit your solution as a Git repo or ZIP file with the following:

📝 Must Include:

1. `README.md` with:
    - Setup instructions
    - Architecture diagram (ASCII, draw.io, or hand sketch)
    - Description of your design decisions
    - Explanation of the security incident & how you fixed it
    - Any assumptions or known issues

2. Code:
    - Helm charts or Kustomize manifests
    - YAML configs for Deployments, Services, NetworkPolicies, Secrets
    - OPA/Kyverno/Falco policy files (if used)
    - Dockerfiles if you modified anything (optional)
    - Exported Grafana dashboards (as JSON)

✅ Evaluation Criteria

Category	What We Look For
Kubernetes Knowledge	Namespaces, probes, RBAC, ingress, service accounts
Security Thinking	Least privilege, network policies, runtime policy enforcement
Observability	Useful dashboards, metrics, ability to detect issues
Failure Handling	Can you identify, simulate, and recover from issues?
Documentation	Clear, reproducible, and well-reasoned README
Real-World Thinking	Infrastructure decisions mimic EKS-style production practices

---

⏳ Time Estimate

~4–6 hours depending on your experience. You're welcome to take more time if needed — depth matters more than speed.

✨ Notes

- You are encouraged to make decisions and assumptions — just explain them in your README.
- You may reuse small snippets, but do not just paste from tutorials. We're interested in *your thought process*.
- If something is unclear, feel free to request clarification.
