

DevOps / Infra Engineer: Technical Challenge

- Candidates should submit their solution within **three** business days of receiving the challenge.
- This timeline strikes a balance between being rigorous and achievable for senior-level engineers.

Objective


Build a Kubernetes (K8s) environment to deploy a backend API application and database of your choice while incorporating SSL and demonstrating fault tolerance, auto-scaling, and operational efficiency. The solution must use Terraform and Helm for provisioning and deployment, be cloud-agnostic, and include a Makefile for automation.

Challenge Requirements

1. Environment Setup

- Create a Kubernetes cluster **locally** (MiniKube, Docker).
- Use Terraform to provision the infrastructure.
- Ensure the solution is **cloud-agnostic** (compatible with at least AWS, GCP, or Azure). Since these will be separate solutions, no cloud-agnostic infrastructure provision exists. However, tools like [Crossplane](#) can incorporate multiple solutions under the hood and expose a single API. (🌟 Nice to have)

2. Application Deployment

- Deploy any backend API application (e.g., Node.js, Python, Go).
 You could use any tool to generate the App or download an open-source app like [this one](#) to bypass this step.
- Use **Helm charts** to manage application and database deployments.
- Configure [SSL](#) for secure communication between the app and the DB. (🌟 Nice to have)

3. Fault Tolerance

- Design the system to handle failures gracefully, ensuring the following:
 - Automatic recovery of pods (e.g., using `livenessProbe` and `readinessProbe` in Kubernetes).
 - Database resilience (e.g., through replication or clustering).

4. Auto-scaling

- Configure auto-scaling for:

- API application (horizontal pod autoscaler).
 - Database (vertical or horizontal scaling where applicable).
- Ensure the cluster adapts to workload spikes and reduces resources during inactivity.

6. Morning

- Use Grafana and Prometheus.

7. Deliverables

- **Documentation:**
 - A README file with:
 - Steps to provision the environment.
 - Instructions to deploy and test the application.
 - Explanation of your fault-tolerance and scaling approach.
 - Architectural diagram of the solution.
- **Code Repository:**
 - Terraform scripts for infrastructure provisioning.
 - Helm charts for application and database deployment.
 - Kubernetes manifests (if applicable).

Evaluation Criteria

- The architecture should be entirely provisioned via some infrastructure as a code tool.
- The presented solution must handle server (instance) failures.
- Components must be updated without downtime in service.
- The deployment of new code should be completely automated (bonus points if you create tests and include them in the pipeline).
- The database and any mutable storage must be backed up at least daily.
- All relevant logs for all tiers need to be easily accessible (having them on the hosts is not an option).
- You should be able to deploy it on one more prominent Cloud provider: AWS, Google Cloud, Azure, DigitalOcean, or RackSpace.
- The system should present relevant historical metrics to spot and debug bottlenecks.
- During the interview, an architectural diagram / PPT will be provided to explain your architecture.
- All the relevant configuration scripts (Terraform/Ansible/Cloud Formation/ARM Templates)
- All the relevant runtime handling scripts (start/stop/scale nodes).
- All the relevant backup scripts.
- You can use another git provider to leverage hooks, CI/CD, or other features that are not enabled in Toptal's git. Everything else, including the code for the CI/CD pipeline, must be pushed to Toptal's git.
- The complete solution must be deployed and working before the interview.