TECHNICAL CHALLENGE

Sr. Cloud/DevOps Consultant

Comprehensive documentation covering the solution design and implementation of the challenge.

Objective

This document demonstrates various aspects of the project, including what the Terraform code builds, how the CI/CD pipeline operates, details of the Helm chart, and the reasoning behind the architectural choices.

Summary

This is a nearly high-level overview of how this challenge was approached. More technical details will be shared during the discussion, as agreed.

Infrastructure Provisioning

The Terraform code is built as modules for ease of management, maintenance, and enhancement. There are five modules under the terraform/modules directory (bastion, gke, iam, redis, and vpc). All five modules share the same architectural structure, with each containing a main.tf, variables.tf, and outputs.tf file keeping the code clean and easy to edit or refactor. The previously mentioned modules provision the following infrastructure resources and components:

1- **GKE Cluster**: A private regional cluster with a single worker node (configurable via google_container_cluster). Other cluster-specific configurations—such as intra-cluster networking, Kubernetes version, and node hardware specifications—can also be configured here.

2- **Node Pool and Node Configuration:** Defined custom settings for node configuration and autoscaling policies. Workload Identity is also configured to bind the Kubernetes service account (KSA) with the corresponding Google Cloud service account (GSA), enhancing security.

3- **Networking:** Created a Virtual Private Cloud (VPC) with a “mgmt” subnet, where the Bastion VM resides, and a “private” subnet for the GKE cluster nodes. Firewall policies and NAT configurations were also defined. As requested, the GKE cluster is completely isolated and can only be accessed from the Bastion. The Bastion VM has internet access via a NAT IP.

4- **Artifact Registry:** Created a GCP artifact/docker registry (name: devops-demo-repo) to push built images by the GitHub actions workerflow runner.

5- **Service Accounts and IAM Roles:** Implemented least privilege IAM policies across three service accounts to strengthen security:

   a. gke-nodes-sa@devoteam-463111.iam.gserviceaccount.com : Grant the GKE nodes to pull images from Artifact Registry.
   b. github-actions-sa@devoteam-463111.iam.gserviceaccount.com : Granted access to the GitHub Actions runner (hosted on the Bastion VM) to access the GKE cluster for application deployment. This integration was implemented using Workload Identity Federation (WIF) with OIDC tokens, a workload_identity_pool, and a provider configured with GitHub as the issuer. An attribute condition was set to restrict access to only my repository. The final step in this integration was assigning a principal to allow the GitHub workflow to act as the github-actions-sa service account.
   c. tfm-sa@devoteam-463111.iam.gserviceaccount.com : Granting Terraform needed privileges to provision infrastrucure resources.

6- **Redis:** reated a highly available Redis setup by implementing a primary-failover configuration, where the primary instance is in the me-central1-a zone and the failover instance is in me-central1-b.

7- **Bastion:** Created a Bastion VM running Ubuntu 24.04 LTS, with all external access blocked except through Identity-Aware Proxy (IAP) requests. This VM serves as the sole access point to interact with the GKE nodes, and access is restricted to the github-actions-sa service account only.

Containerization

Containerized the application using a Dockerfile based on python:3.10-slim, exposing the required environment variables during the build process. The default user was changed from root to a non-root user named devoteam to enhance container security. Finally, the application port was exposed and the application was launched.

Helm

Created the “devo-chart” Helm chart to manage Kubernetes deployments and releases, making the process configurable and reusable. A LoadBalancer service (backed by GCP's load balancer) was used to expose the application. An autoscaling policy was configured using Kubernetes Horizontal Pod Autoscaler (HPA), with the trigger set to an average memory utilization of 80%.

CI/CD Pipeline

The project uses a GitHub Actions workflow (located in the .github/workflows/ directory) to implement Continuous Integration and Continuous Delivery (CI/CD). The pipeline is automatically triggered upon a push to the main branch and ensures that code changes follow a standardized process—build, push, and deploy—without manual intervention.

The pipeline consists of two jobs:

1. Build-push – This job includes the following steps:
   a. Check out the code.
   b. Authenticate to GCP using Workload Identity Federation (WIF).
   c. Install the Google Cloud SDK on the runner and set the project.
   d. Build, push export the image SHA tag (usingGITHUB_SHA as the tag).

2. Deploy – This job includes:
    a. Check out the code.
    b. Authenticate to GCP using Workload Identity Federation (WIF).
    c. Install the Google Cloud SDK on the runner and set the project.
    d. Fetch the GKE cluster's internal endpoint and populate the kubeconfig.
    e. Deploy a Helm release.

Monitoring

To ensure the deployed application is reachable and to enable quick alerts in case of downtime, **Cloud Monitoring Uptime Checks** were leveraged along with an alerting policy. The uptime check periodically sends HTTP requests to the application's endpoint every 60 seconds from multiple global locations. If the application becomes unreachable (with a timeout threshold of 10 seconds), an alert notification is sent to my personal email.

Security Measures

Below are some of the security best practices followed during the setup:

1. Restricted firewall rules to allow only the absolute minimum access.
2. Applied minimal IAM scopes on GKE nodes.
3. Configured containers to run as non-root users.
4. Used Workload Identity Federation for keyless CI/CD authentication.
5. Assigned IAM roles based on the principle of least privilege.
6. Deployed the application in a non-default namespace (assuming you meant Kubernetes namespace; if you meant "non-default database," clarify accordingly).