Stateful Applications With Kubernetes and Cloud SQL

- Creating DB with cloud sql
- Setting up Kubernetes
- Deploying an Application
- Examining fault tolerance

The application

Simple hello world happy birthday application

Description: saves/updates the given user's name and date of birth in the database
Request: PUT /hello/<username> { "dateOfBirth" : "YYYY-MM-DD" }
Response: 204 No Content

App checks username is all letters and date of birth is larger or equal to today.

Description: returns hello happy birthday message for given user

a. If username's birthday is in N day(s):

```
{
    "messege": "Hello <username>! Your birthday is in N day(s)!"
}
```

b. If username's birthday is today:

```
{
    "messege": "Hello <username>! Your birthday is today!"
}
```

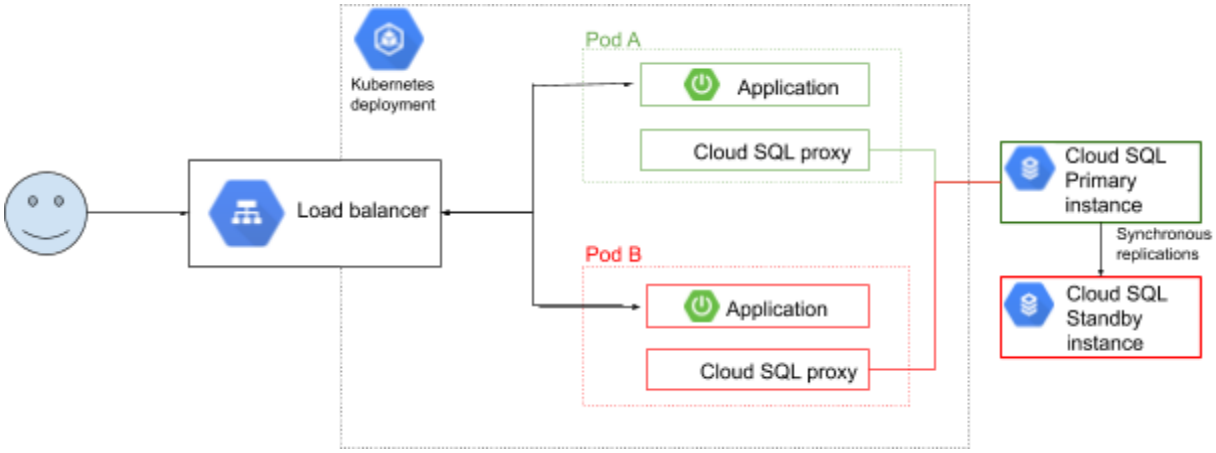
Goals:

- Handle traffic
- Low maintenance
- (Almost) always available

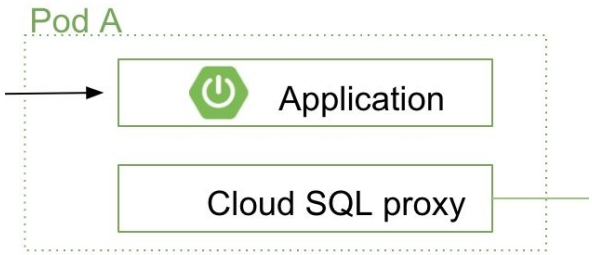
High availability:

- Use a **primary** instance (master) and a **standby** instance in different zones.
- Backed by a regional **persistent disk** that uses **synchronous replication** to sync any writes.
- When an unhealthy instance is detected, the standby instance is promoted.
- Failovers typically take ~60s from once triggered.

Solution architecture:



- The end user is interacting through a load balancer, the load balancer is going to be distributing traffic between the pods.
- Pods, the atomic unit is inside a deployment.
- The deployment manages the pods for us.
- The load balancer is going to direct traffic to the pods.
- Pods are connected to the cloud sql instance.
- The Cloud SQL instance had HA so it has asynchronous replication to a standby instance.
- The “sidecar” pattern:



Prerequisites

The steps described in this document require installations of several tools and the proper configuration of authentication to allow them to access your GCP resources.

Cloud Project

If you do not have a Google Cloud account, please signup for a free trial here. You'll need access to a Google Cloud Project with billing enabled. See Creating and Managing Projects for creating a new project. To make cleanup easier it's recommended to create a new project.

Required GCP APIs

The following APIs will be enabled:
Compute Engine API
https://console.developers.google.com/apis/api/compute.googleapis.com/

Service Networking API
https://console.developers.google.com/apis/api/servicenetworking.googleapis.com/

Cloud SQL Admin API
https://console.developers.google.com/apis/api/sqladmin.googleapis.com/

Kubernetes engine api
https://console.developers.google.com/apis/api/container.googleapis.com/

To deploy everything:

Tools

When not using Cloud Shell, the following tools are required:

- Access to an existing Google Cloud project.
- Bash and common command line tools (Make, etc.)
- Terraform v0.12.3+
- gcloud v255.0.0+
- kubectl that matches the latest generally-available GKE cluster version.

Install Terraform

Terraform is used to automate the manipulation of cloud infrastructure. Its installation instructions are also available online.

Install Cloud SDK

The Google Cloud SDK is used to interact with your GCP resources. Installation instructions for multiple platforms are available online.

Install kubectl CLI

The kubectl CLI is used to interteract with both Kubernetes Engine and Kubernetes in general. Installation instructions for multiple platforms are available online.

Authenticate gcloud

Prior to running this demo, ensure you have authenticated your gcloud client by running the following command:
gcloud auth login

Configure gcloud settings

Run gcloud config list and make sure that compute/zone, compute/region and core/project are populated with values that work for you. You can choose a region and zone near you. You can set their values with the following commands:
Where the region is us-central1
gcloud config set compute/region us-central1

Updated property [compute/region].
Where the zone inside the region is us-central1-c
gcloud config set compute/zone us-central1-c

Updated property [compute/zone].
Where the project name is my-project-name
gcloud config set project my-project-name

Updated property [core/project].

Deploy everything:

Prerequisite:
Create terraform/terraform.tfvars
As follows:

```
# GCP Settings
gcp_location = your_gcp_location
gcp_zone =  your_gcp_zone
gcp_auth_file = "account.json"
app_project =  your_gcp_project_name
initial_node_count = 3
machine_type = "n1-standard-1"
```

Terraform init
Terraform apply -auto-approve

#create secret vault as a sidecar
#init with your secrets
Kubctl create -f terraform/secrets.yaml

#deploy app container & cloud sql proxy
Kubectl create -f terraform/deployment.yaml

#deploy load balancer
Kubctl create -f terraform/loadbalancer.yaml

#deploy autoscaler for zero downtime
Kubctl create -f terraform/autoscaler.yaml

To Destroy:
Terraform destroy -auto-approve