Sensyne Tech Test
Notes and explainations

Spike Lampson

May 9, 2019

Contents

1 Initial design

Requirements are: Simple web app, with backend and database, deployed on a cluster of nodes via an orchestration layer. Expectation: Couple of hours, which seems very hopeful for setting up a project at all, let alone finishing it to me. Still, what to do.

Initial plan: Create in terraform/AWS. Find basic docker webapp container, deploy that onto EKS, with RDS instance as the backend.

Things to check:

1. How much VPC/IG/routing etc still needs to be done by hand
2. Does EKS handle the loadbalancer?
3. Find basic docker container

2 Cluster

Cluster based on https://learn.hashicorp.com/terraform/aws/eks-intro

3 Kubeconfig

To create the kubectl config, run `terraform output kubeconfig > /.kube/config`.

To create the config map, run `terraform output config_map_aws_auth > config_map_aws_auth.yaml` followed by `kubectl apply -f config_map_aws_auth.yaml`

This relies on `aws-iam-authenticator`, which is installed via https://docs.aws.amazon.com/eks/latest/us aws-iam-authenticator.html

4 Creating a basic service

Unashamedly copied from step 4 of https://docs.aws.amazon.com/eks/latest/userguide/getting-started.html

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/examples/master/guestbook-go
kubectl apply -f https://raw.githubusercontent.com/kubernetes/examples/master/guestbook-go
kubectl apply -f https://raw.githubusercontent.com/kubernetes/examples/master/guestbook-go
kubectl apply -f https://raw.githubusercontent.com/kubernetes/examples/master/guestbook-go
kubectl apply -f https://raw.githubusercontent.com/kubernetes/examples/master/guestbook-go
kubectl apply -f https://raw.githubusercontent.com/kubernetes/examples/master/guestbook-go
```

This will create a basic guestbook service, with a Redis backend. Sadly, it appears that the guide is out of date, as the docker container for the redis-slave services (`kubernetes/redis-slave:v2`) appears to not be working today. Still, the service starts, and displays a nice error message, and the failure in the pods can be seen with `kubectl get pods` and `kubectl describe pods redis-slave`.

I have chosen not to use a seperate service for redis, or another database, due to having spent some time on this already.