Ort Singalovski Final project

Introduction:

This project will cover all the subjects we learned throughout the 2 months of the course. During this project we will implement the technologies to provide an end to end solution for a sample application

The application will have the following components:

- Frontend
- Backend
- DB

You are required to choose the frontend and backend technologies you are most interested in.

The Final result will be a complete Solution to build and deploy the application to the chosen technologies.

Technologies

The technologies that we will use:

- Git
- Docker
- Kubernetes
    - Ingress
    - Deployment/Statefulset
    - Job
    - Service
- Helm
- Terraform
- Jenkins/github actions
- ArgoCD
- Prometheus/Grafana
- Elasticsearch/Kibana
- AWS
- Slack

Implementation

Infrastructure

All the infrastructure will be defined as code using Terraform (or any other tool that you desire) The code will be in a separate infrastructure repository.

Make sure state is saved into a state bucket

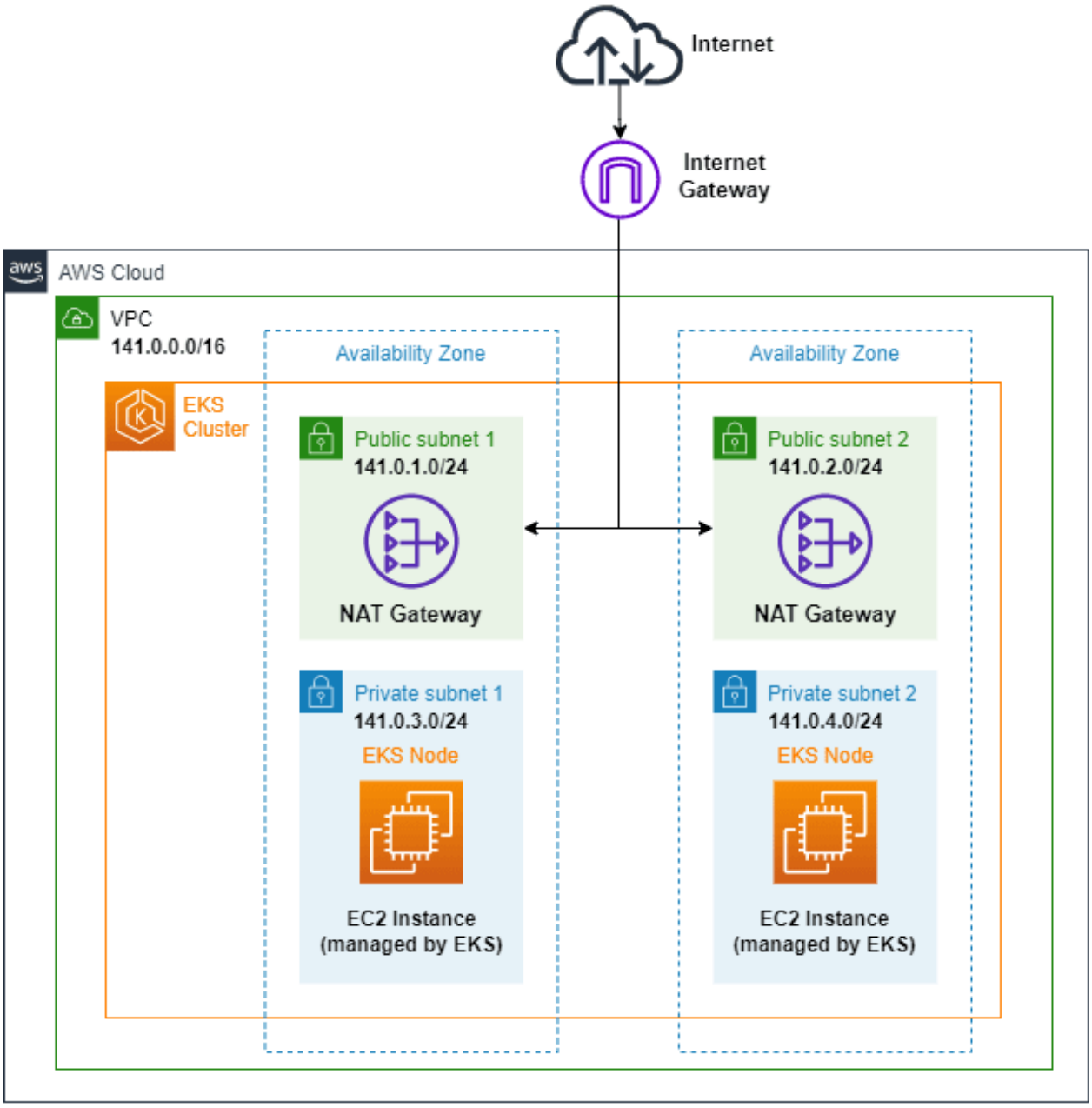
The implementation should be as a module that will represent a single environment and will be easy to reuse for additional ones.

The infrastructure should use as many modules as possible instead of rewriting modules for example:

- https://registry.terraform.io/modules/terraform-aws-modules/vpc/aws/latest
- https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest

The module you create will contain the above modules and additional ones as required

The architecture you will need to create is:



On the cluster you will need to install additional resources using helm - this should be as part of the terraform:

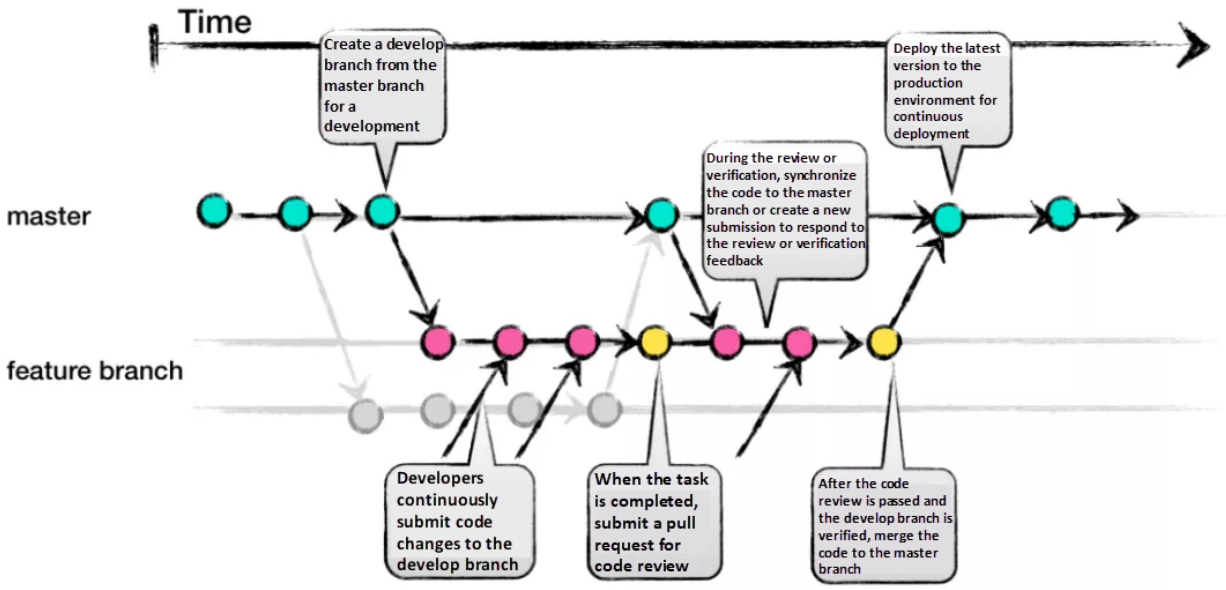
- Prometheus/Grafana
- Ingress controller
- Elasticsearch/Kibana (or cloudwatch logs instead)
- ArgoCD
- If you prefer to use Jenkins instead of github actions
    - Install Jenkins and setup all the configuration is JCaSC plugin

Add instructions on the repo how to access the ArgoCD and deploy the application

Git and CI

Based on the CI Tool you have chosen implement the following:

The git and CI flow we have chosen to implement is Github Flow as seen below



In the CI system (Jenkins/Github actions) We will implement a CI/CD flow based on the source of the change - When the commit is to a master branch we will deploy to staging, based on the results of the tests we've implemented.

A push tag with version will trigger a deployment to production.

The Development flow will be as following:

- Create a feature branch with the following naming convention:
    - feature/RND-123-add-new-feature
- All pushed commits to the feature will trigger a CI process which will run the following tests:
    - Code lints
    - Build
- Following a successful build and lint we will be able to merge to master with a pull request

    - Hint: create a protected branch in github to prevent changes that haven't finished successfully the CI pipeline
- A merge into master will run additional tests and deploy to staging

Build

The applications that we create will create a docker image as an artifact and will push it into the ECR

Deployment

The deployment into the cluster will be implemented using ArgoCD and with an additional repository which will contain the desired state of each one of the environments (test/staging/production)

Application

The application you will create is a multi-tier microservice application which includes:

- Frontend
- Backend
- Database

This technologies you will choose are up to you and you should select technologies you are interested in or familiar with.

Each of the services should implement the CI flow described above and be deployed using ArgoCD as explained in the previous section.

Create a Helm chart for each one of the services and include it in the source repo for the application.

Monitoring

Make sure you have dashboards that will show you something about the nodes.

You can use the kube-prometheus-stack helm chart to get a basic useful dashboard installed out of the box if you are lazy.

Bonus

Add alerts that will send a slack notification when the pods are restarting and/or any other issues occurs during the build and/or deployment.

This is the original assignment you received during the lesson which you should base the solution above on

This Challenge - If you choose to accept it - Will help you understand the flow of building an application from scratch and deploying it into kubernetes

Make sure Everything you do is pushed into Github

Required steps:

Step 1

- Open a new Git Repository for this project
- Create a new python application - https://pythonbasics.org/flask-tutorial-hello-world/
- Don't go further than the first hello world
- Wrap the application in a Docker container
- Create a new repository in a private ECR - https://docs.aws.amazon.com/AmazonECR/latest/userguide/repository-create.html
- Push the docker container created previously into the ECR repo
- Git commit and push the code into the repository

Step 2

- Use the same repository for the above application and create a subdirectory for the helm chart

- Create a helm chart for the above application
- Make sure you configure the imagepullsecrets so you will be able to pull the container into the kubernetes cluster
- Make sure you enable an ingress and service
- Make sure that you configure the correct resource limits and requests
- Commit and push the helm chart to the github

Step 3

- Deploy the helm chart

Step 4

- Create a Simple Jenkins build that will automatically create your application
    - Build the docker container you created from your github
    - Push the docker container to the ECR repository
        - Remember This will need to authenticate with the ECR

Step 5

- Migrate the jenkins build to a pipeline with a jenkinsfile
    - Make sure the build is running from the Jenkinsfile from the git
    - Change the jenkinsfile and see if the build was updated