Senior DevOps Engineer Take Home Challenge

The Task

To create and demonstrate a CICD pipeline, that includes:

- At least 1 quality assurance gate during the CI process.
- The CD pipeline has to include a configuration management tool.

Technologies can be selected by the candidate, essentially, we are looking for a repeatable way to build some code and deploy it somewhere.

Solution

Though there are many ways to implement the solution, considering the time and cost, I tried to use as many technologies as possible to demonstrate my knowledge on the tools.

What Technologies I used:

The following technologies are used to complete the challenge

1) Terraform
2) Kubernetes
3) Ansible
4) Sonar Kube
5) Jenkins
6) Docker
7) AWS
8) Azure
9) Maven
10) Git

High level Solution:

1) Application code will be comiited to Git
2) SCM will polled every minute and the Jenkins Pipeline will be triggered as soon as the changes are pushed to GIT
3) Maven will initiate the sonarQube analysis to check the quality of the code
4) Based on the set criteria in Quality gate, SonarQube will mark the build as Pass or fail and It will also highlight any code smells
5) The code will packaged by maven
6) The .war will be pushed to S3 (Instead of Artifiact manager like JFrog)
7) Ansible will pull the code from S3 and then an Image will be created and pushed to docker hub

8) Implementer approval will be required before publishing the code to the site
9) Once the approval Is given, the kubernetes deployment will be done to refresh the pods

How I used the technologies

1) **Terraform** -
All the Infrastructure related components such as Jenkins server, SonarQube server and Kubernetes were created using Terraform. The scripts can be repurposed to deploy the components In multiple areas. The dependency on the code is reduced as much as possible (considering the available time to complete the activity)

2)**Kubernetes**-
Initially tried to deploy the cluster manually and then EKS but due to the software compatibility and other reasons, I finally used AKS to deploy the application. I initially used Jenkins plugin to automate the deployment but there is a bug In Jenkins plugin and hence I developed a simple playbook to deploy the kubernetes objects. The deployment of cluster and kubernetes is automated and hence we can reuse to deploy multiple Images across multiple applications and environments (such as Dev, Prod etc)

3) **Ansible**: To reduce the number of tools used In the solution, I tried using S3 to store the artifacts versions Instead of using tool like JFrog but as I', doing the task on small application I tried using the docker plugin to simplify the task but I still Included the ansible which I used to generate Docker Image and this can be reused for other functionalities In future

Due to the Jenkins limitation on kubernetes plugin, I wrote ansible to apply the kubectl configurations during any changes in the Docker Image. I understand that one of the main objective of this task to use configuration management tool to the maximum but I'm also conscious that you want to see my experience on other tools as well and hence I tried my best to accommodate all the technologies by giving Importance to your expectations

To overcome the Issue of my local setup, I configured the IP of my server In the pipeline script and added few customisations using S3 but if we have ansible on our jenkins server then the customisations can be removed.

4)Sonar Qube-
This Is used as a Quality gate and I used the default tests to mark a particular build as success or failures. As this tool requires separate DB, I Installed Sonarqube and postgres on a separate server to avoid any software compatibility Issues. This approach also enabled us to reuse the same server for Quality check the builds across multiple projects and environments. Once all the software's were Installed, an Image has been taken and with the help of the Terraform scripts, the server can be launched as many times as possible

5) Jenkins :
This is used as a deployment server. A terraform script Is created to launch the Jenkins server with all the required dependencies. We can even add this to Auto scaling

group to upscale or downscale based on the demand and it can also be reused In other areas. Pipleline is configured as code so that the same can be reused based on the requirement

6) Docker: The Java code will be Installed on top of existing tomcat container to ship the code easily. Once the Image Is created, It will be pushed to the docker hub so that the kubernetes application can use the latest container Images

7) AWS - The AWS Infrastructure creation Is automated with Terraform . As part of Jenkins and other required server configurations, a VPC will be created and then subnet, route table, Internet gateway, Security group and S3 bucket etc were Implemented as a code . So the same can be reused In multiple places

8) Azure- Azure is used to host the container orchestration tool (AKS). The deployment of AKS is automated with the help of Terraform. Based on the requirement, the same code can be reused to host multiple kubernetes clusters.

9) Maven : Used as build tool to Invoke the sonarQube analysis and also to produce the artifcats. Initially planned to Integrate with JFrog but due to time constarints, I Implemented logic to push the code to S3 and from there the Images can be built and we can reuse this Implementation even If we want to avoid hosting the application as docker Image

10)Git : Used as a version control tool to maintain different versions of Java code. I also added the pipeline code In Jenkins File in case If we want to reuse the pipeline process for other environments