Devops Case Study 1

Implementation steps:

1. Creation of base VM
2. Installing terraform, Visual studio code, Eclipse and git.
3. Installing terraform, git, ansible extensions in VS Code.
4. Login to github and create a new repository.
5. Integrating terraform with azure.
6. Creating terraform scripts to create master, slave node and install java and jenkins in master node & java, maven, ansible, docker, azure cli, git, ansible modules and sshpass using VS Code.
7. Creating the application code and docker file in eclipse.
8. Creating the ansible playbooks to create an VM and to install docker using VS code.
9. Push the terraform scripts, ansible playbook and application code to git hub.
10. Run the terraform scripts to create 2 VMs.
11. Log in to Master node, start jenkins, configure jenkins and ssh authentication with key to slave node.
12. Configure slave VM as slave node in jenkins.
13. Log in to slave server, login to azure, login to git and dockerhub.
14. Configure ansible to refer to host file.
15. Create an shell script to grep the IP of docker vm and update in ansible host files.
16. Integrate ansible with azure by creating the credentials.
17. Login to jenkins console, install the maven, docker, ssh and ansible plugins.
18. Configure the slave node.
19. In slave node configure the jenkins such that it will run on port 4243, Configure the slave docker cloud in jenkins.
20. Configure maven home and ansible home in Global Configuration tools of manage jenkins.
21. Configure docker hub credentials in configure of manage jenkins.

I have done the project in two different ways.

1) First Approach:

Single pipeline which involves configuring the GIT repository, application packaging, docker build and push to docker hub, Creation of VM and installing docker using ansible and deploying the container with application image on the new vm.

2) It consists of 2 pipelines

**Pipeline 1** will be running on slave node which involves configuring the GIt repository, maven build, docker image build and push to docker hub, Creation of VM and installing docker using ansible.

**Pipeline 2** will execute docker commands to kill, remove the earlier running container, removing the old images and running the container with new updated docker image.

Approach 1:

1. Create a jenkins freestyle pipeline which will refer to new repository created in github as **source code management**.
2. In pipeline build trigger, select GitHub hoot trigger for GITScm polling.
3. In pipeline build select **Invoke top level maven targets** select the maven configured earlier and select package as goals and give the path of pom.xml file.
4. Next in pipeline build select **Build/Publish Docker image**, mention the Docker file path, select the docker cloud, give the docker preferred image name, tick push image option and give docker hub credentials.
5. Next in pipeline select **Invoke Ansible Playbook**, select earlier configured ansible and give the path of ansible vm creation playbook.
6. Next in pipeline select **Execute Shell** and try to run the shell script to fetch IP of newly created VM.
7. Next in pipeline select **Invoke Ansible Playbook**, select earlier configured ansible and give the path of ansible docker creation playbook and give server credentials.
8. Configure the ssh site for new vm created.
9. Next in pipeline select **Execute shell script on remote host using ssh** and select confiured ssh site and enter the commands to stop and remove the earlier running docker images on docker server, remove the docker images.
10. Enter the command to start the container with earlier pushed docker image containing the packaged application.
11. In git hub repository add jenkins webhook.
12. Test the pipeline by modifying the code.

Approach 2:

Pipeline 1: Which will be running on slave node.

1. Create a jenkins freestyle pipeline which will refer to new repository created in github as **source code management**.
2. In pipeline build trigger, select GitHub hoot trigger for GITScm polling.
3. In pipeline build select **Invoke top level maven targets** select the maven configured earlier and select package as goals and give the path of pom.xml file.
4. Next in pipeline build select **Build/Publish Docker image**, mention the Docker file path, select the docker cloud, give the docker preferred image name, tick push image option and give docker hub credentials.
5. Next in pipeline select **Invoke Ansible Playbook**, select earlier configured ansible and give the path of ansible vm creation playbook.
6. Next in pipeline select **Execute Shell** and try to run the shell script to fetch IP of newly created VM.
7. Next in pipeline select **Invoke Ansible Playbook**, select earlier configured ansible and give the path of ansible docker creation playbook and give server credentials.

Pipeline 2:

1. Create a jenkins freestyle pipeline and in **build triggers** select **Build after other projects are built** and select pipeline 1 as select **trigger only if build is stable option.**
2. Next in pipeline, in build select **Execute docker command**, select **stop Containers,** enter container ID.
3. Next in pipeline, in build select **Execute docker command**, select **Remove Containers,** enter container ID.
4. Next in pipeline, in build select **Execute docker command**, select **Remove image,** enter container ID.
5. Next in pipeline, in build select **Execute shell** and enter the command to run the container with earlier pushed docker image containing the packaged application.
6. Modify the code and test the pipeline.