CLOUD DEVOPS ENGINEER

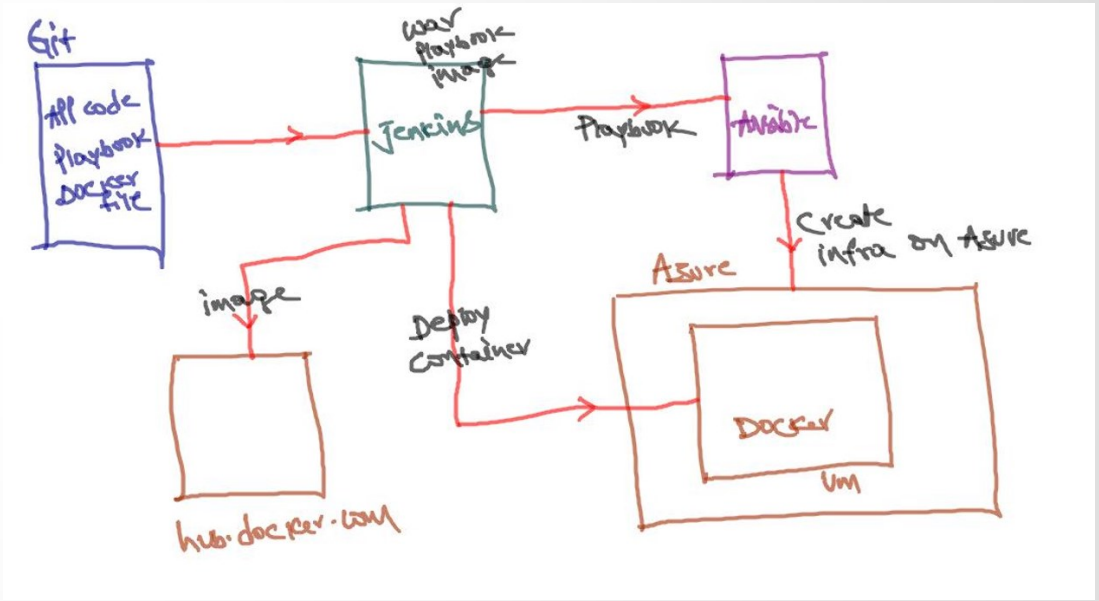
PROJECT CASE STUDY

BUSINESS REQUIREMENT

- At Wolkentech Pvt Ltd, there was a separate team that provided dedicated Jenkins pipelines with a stable master-slave node setup, but the environment was only used for quality assurance (QA), staging, and production environments. The development environment was still very manual, and the team needed to automate it to gain as much flexibility as possible while accelerating the development effort. This is the reason they decided to build a CI/CD pipeline for DevOps. And the open source version of Jenkins was the obvious choice due to its flexibility, openness, powerful plugin-capabilities, and ease of use.

SOLUTION:

- Build a multi-staged Java build pipeline that takes from the phases of pulling dependencies from JAR repositories like Maven, compiling Java codes, running the unit tests, packaging into a JAR/WAR file, and deploying to a cloud server.
- Construct a multi-pipeline automating the tasks of executing Ansible playbooks to deploy the required infrastructure for Application.
- Design a complete end-to-end DevOps pipeline that pulls the infrastructure resource files and configuration files stored in SCM like GitHub and executing the scripts through various runtime programs.



IMPLEMENTATION:

Part-1

- Create Jenkins multi-server environment in Azure using Terraform

IMPLEMENTATION:

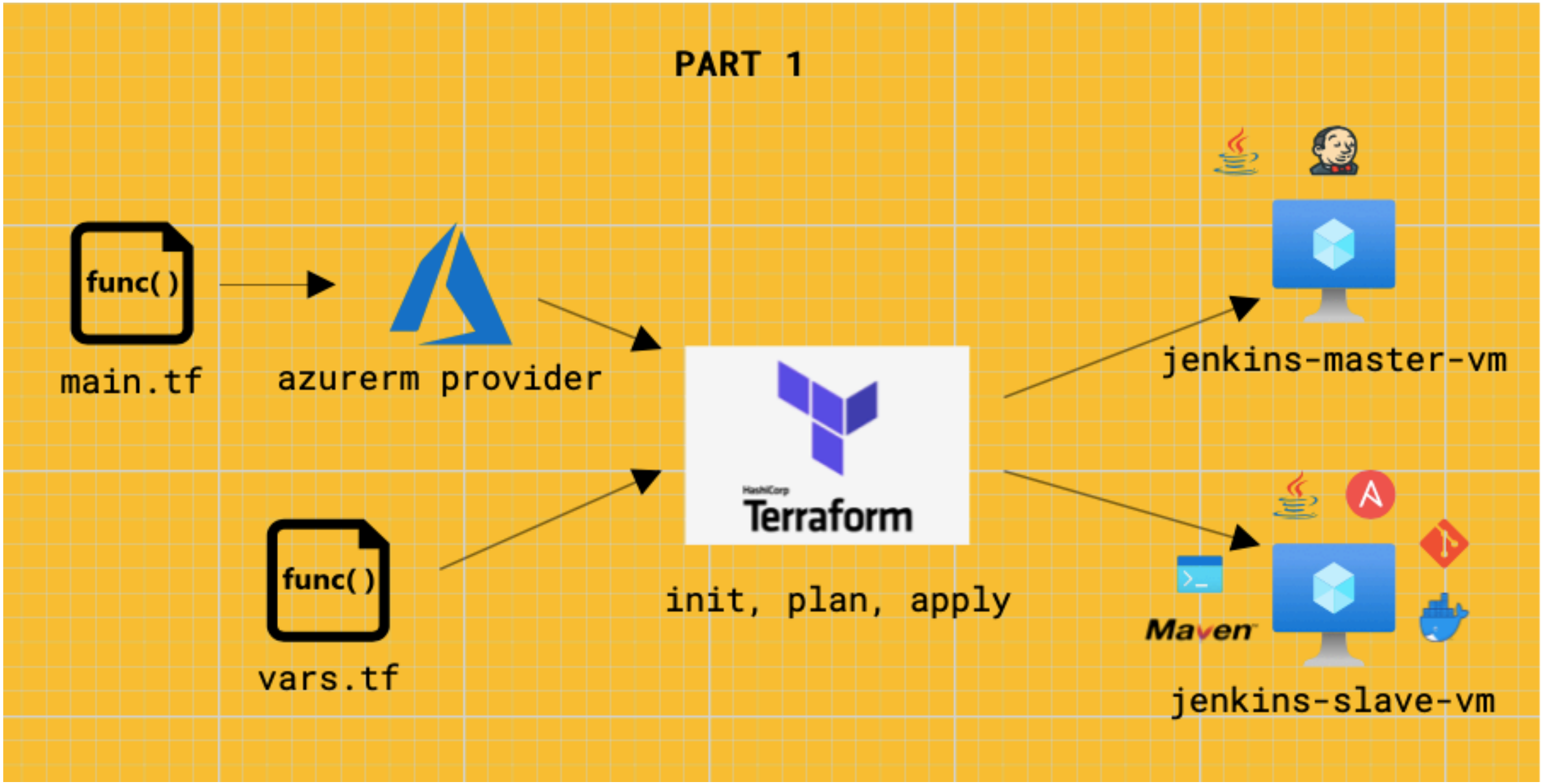
Part-2:

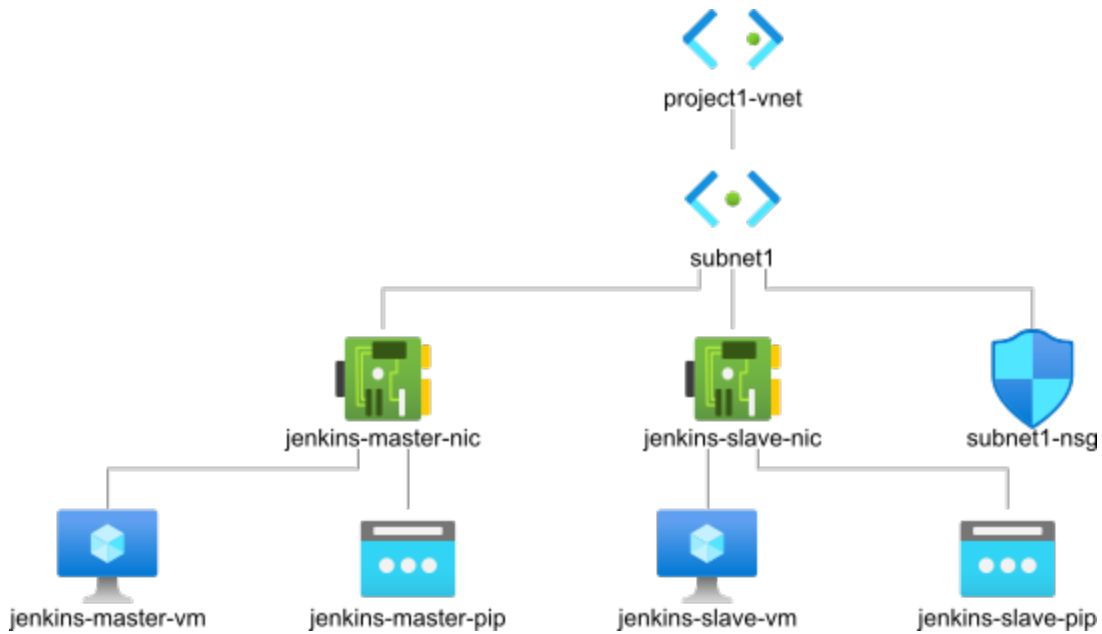
- Phase 1: Simple Java WebApplication
- Phase 2: Containerize the webapplication using Dockerfile
- Phase 3: Pushing Code and Dockerfile to GIT
- Phase 4: Deploy a Change
- Phase 5: Create a Ansible Playbook to Automate Machine Setup with Docker Engine
- Phase 6: Push Ansible code to the same Git
- Phase 7: Deploy Your Ansible Playbook to Azure and Test it
- Phase 8: Workflow Automation with Jenkins

IMPLEMENTATION:

Pipeline Steps

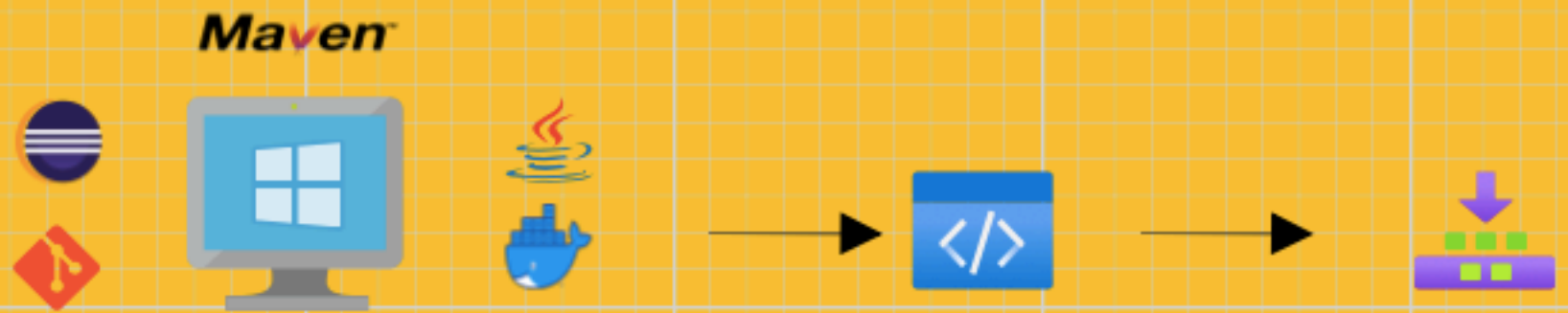
- Fetch Application Code and Ansible Playbook to workspace
- Build Application Code
- Build the Docker Image
- Push the Docker image to hub.docker.com/Azure Container registry
- Store artifacts and Ansible playbook in workspace
- Run Ansible playbook to deploy infra to Azure
- Deploy Docker container to Docker host VM


PART 1
func( )
main.tf
azurerm provider
HashiCorp
Terraform
init, plan, apply
jenkins-master-vm
func( )
vars.tf
Maven
jenkins-slave-vm


project1-vnet
subnet1
jenkins-master-nic
jenkins-slave-nic
subnet1-nsg
jenkins-master-vm
jenkins-master-pip
jenkins-slave-vm
jenkins-slave-pip

PART 2

Phase 1



Windows VM

On Windows VM running eclipse, created a Maven project with archetype as web application

Modified index.jsp to display a custom message

Run maven clean install to build code into war file

PART 2

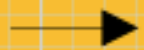
Phase 2

Created a Dockerfile that takes tomcat as base image and deploys the previously built war file

Created a GitHub repository "srikanthkarra/projectone"

Phase 3

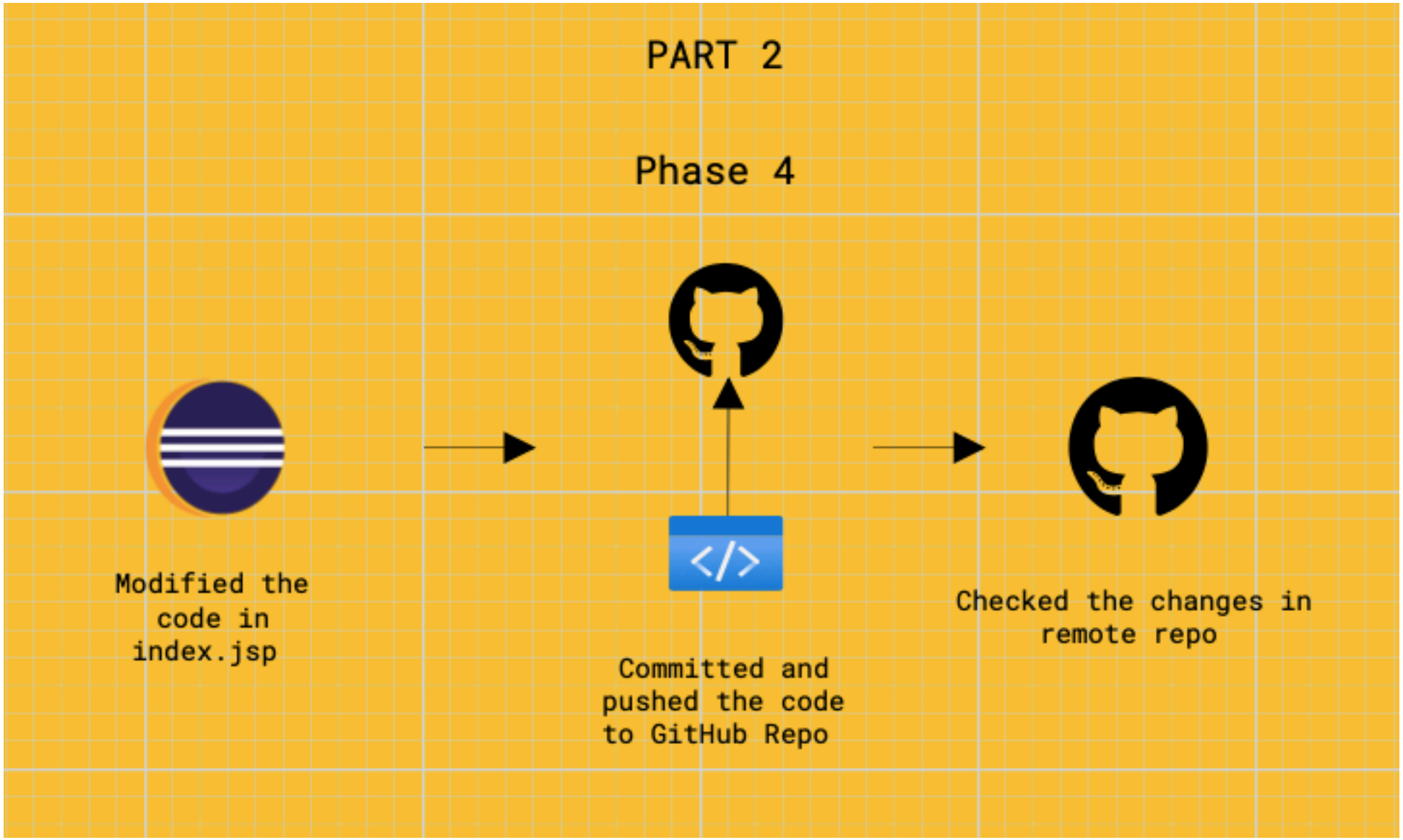
Converted the app into a local repo

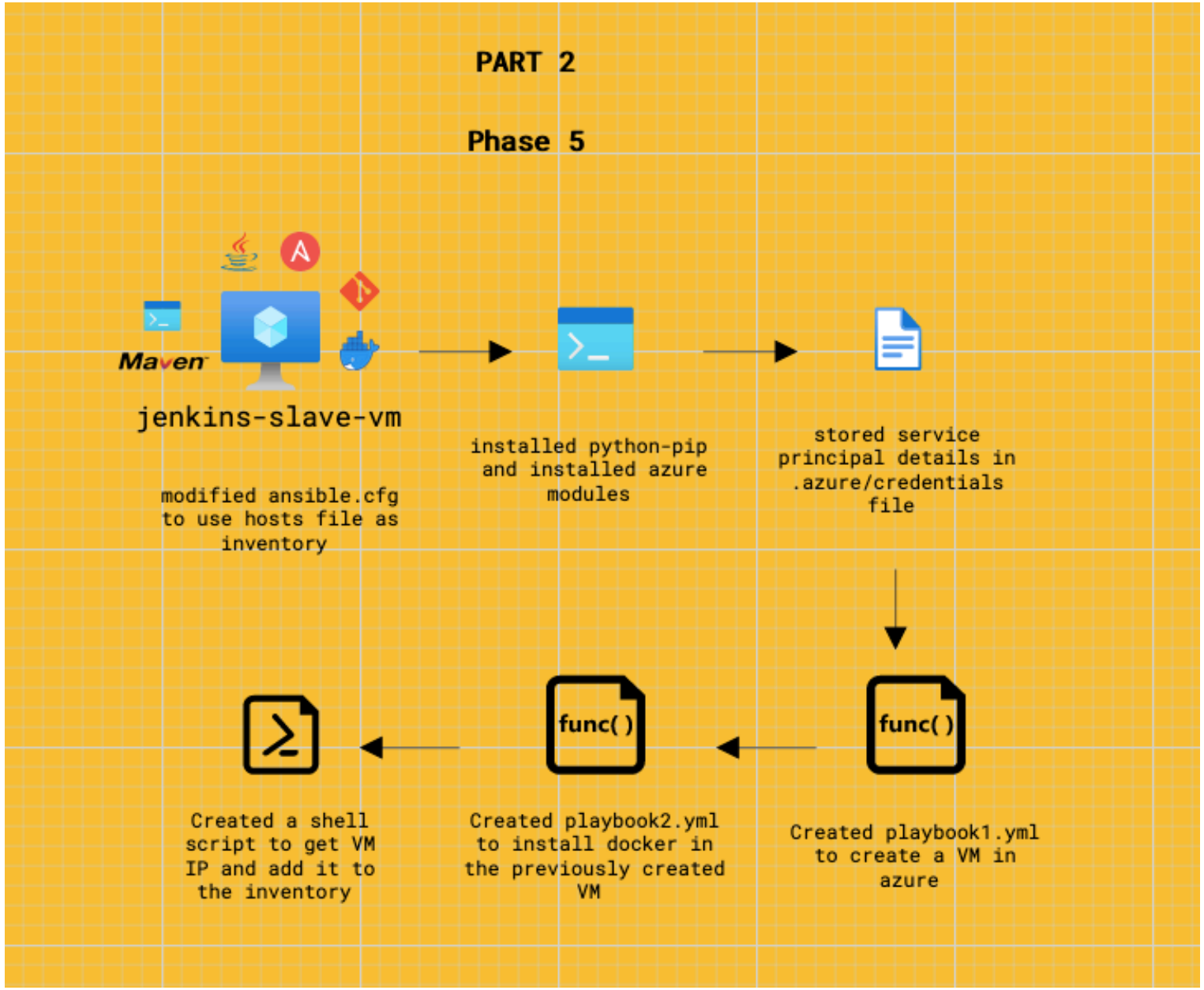


Committed and pushed the code to GitHub Repo

PART 2

Phase 4




PART 2
Phase 5
Maven
jenkins-slave-vm
modified ansible.cfg to use hosts file as inventory
installed python-pip and installed azure modules
stored service principal details in .azure/credentials file
func( )
Created playbook1.yml to create a VM in azure
func( )
Created playbook2.yml to install docker in the previously created VM
Created a shell script to get VM IP and add it to the inventory

PART 2

Phase 6

Pushed playbook1.yml, playbook2.yml and getIP.sh to GitHub repo in a different branch pipeline2

PART 2

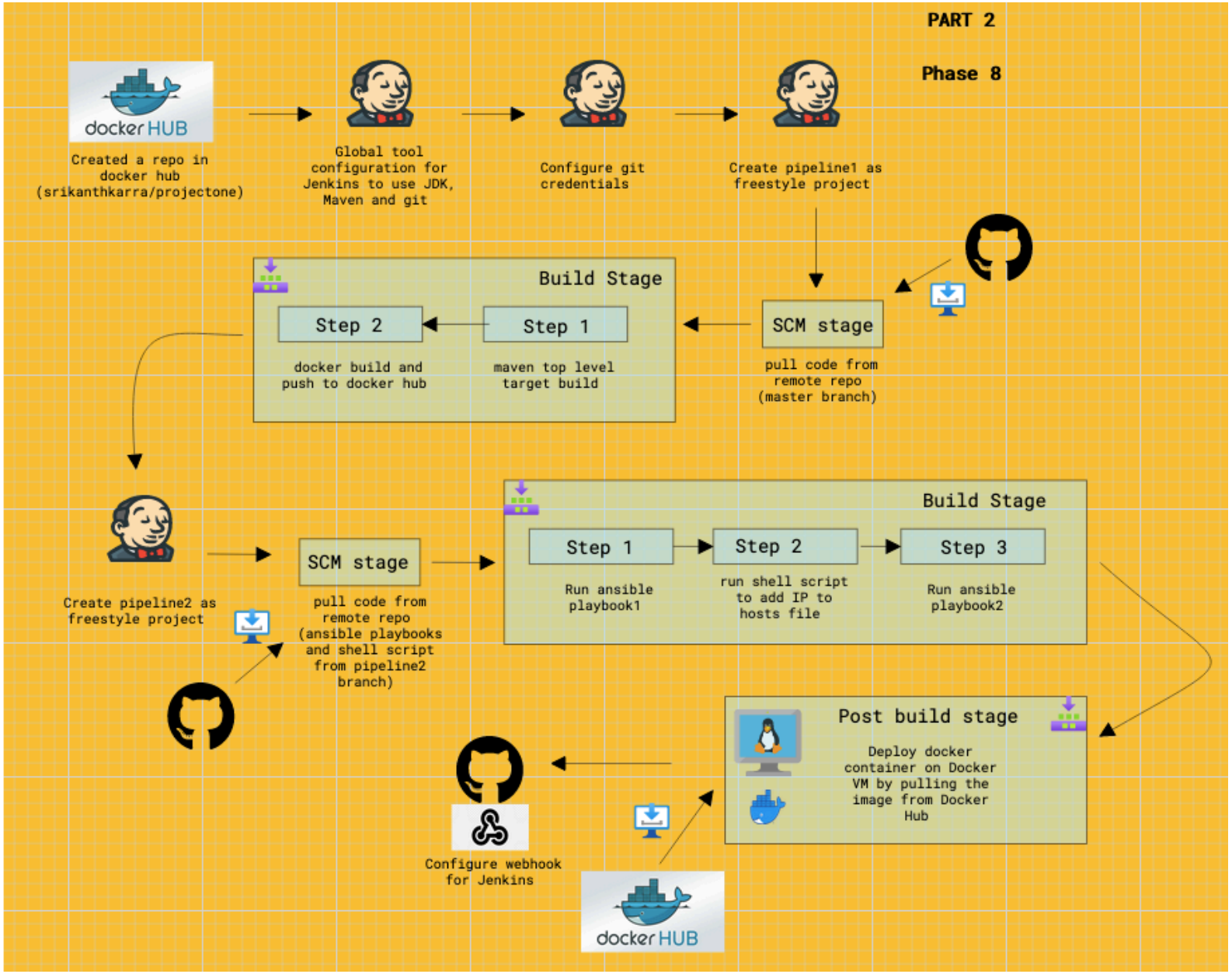
Phase 7

jenkins-slave-vm

Run playbook1.yml to create a VM in Azure

Run the script getIP.sh to add VM's IP to hosts file

Run playbook2.yml to install docker on the created VM


PART 2
Phase 8
docker HUB
Created a repo in docker hub (srikanthkarra/projectone)
Global tool configuration for Jenkins to use JDK, Maven and git
Configure git credentials
Create pipeline1 as freestyle project
Build Stage
Step 2
Step 1
docker build and push to docker hub
maven top level target build
SCM stage
pull code from remote repo (master branch)
Create pipeline2 as freestyle project
SCM stage
pull code from remote repo (ansible playbooks and shell script from pipeline2 branch)
Build Stage
Step 1
Step 2
Step 3
Run ansible playbook1
run shell script to add IP to hosts file
Run ansible playbook2
Post build stage
Deploy docker container on Docker VM by pulling the image from Docker Hub
Configure webhook for Jenkins
docker HUB