ASSIGNMENT

3 tier app deployment HandsOn with terraform modules

1) Clone the code - https://github.com/heydevopsorg/terrform-threetierarch.git

2) Create the AWS account - https://aws.amazon.com/console/

3) Install Docker and terraform on windows laptop/Mac/Linux

https://docs.docker.com/desktop/install/windows-install/

https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

4) Execute the linux command to give permission

chmod +x setup-ecrs.sh

Terraform Commands

There are a couple of commands to check the Terraform's built-in command-line documentation:

- `terraform`
- `terraform -h`
- `terraform --help`

The resulting help page will have the main commands at the top, followed by the less common or more complex commands below.

```
Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
```

We can also enter the terraform command and then a subcommand with `-h` or `--help` to pull up a list of commands that are specific to that subcommand.

200 Terraform Interview Q&A

1. What is Terraform?

Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp that allows users to define and provision data center infrastructure using a high-level configuration language called HCL (HashiCorp Configuration Language) or JSON.

Example:

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
}
```

2. What are the main features of Terraform?

Terraform's main features include Infrastructure as Code (IaC), execution plans, resource graphs, change automation, and state management.

Example:

```
terraform {
  backend "s3" {
    bucket = "my-terraform-state"
    key    = "global/s3/terraform.tfstate"
    region = "us-west-2"
  }
}
```

3. What is the difference between Terraform and other IaC tools like Ansible, Puppet, and Chef?

Exploring the essentials of Terraform with 100 concise questions for quick learning!

1. **What Is Terraform?**
   Terraform is an open-source infrastructure as code (IaC) tool developed by HashiCorp. It allows you to define and provision infrastructure using a high-level configuration language. Terraform is cloud-agnostic, meaning it supports multiple cloud providers like AWS, Azure, GCP, and on-premise systems. It uses a declarative approach, meaning you define the desired state of your infrastructure, and Terraform makes it happen. Its main features include versioning, state management, and modular configurations.

2. **What are the main components of Terraform?**
   Terraform consists of the following key components:
   - **Providers**: Manage resources in cloud services or on-prem systems.
   - **Modules**: Group reusable configurations for efficient scaling.
   - **State**: Maintains the current status of your infrastructure.
   - **Configuration Files**: Written in HashiCorp Configuration Language (HCL) to define infrastructure.
   - **Backend**: Defines where Terraform state data is stored, such as local files or remote storage.

3. **What is the difference between Terraform and other IaC tools like Ansible?**
   Terraform focuses on provisioning and managing infrastructure resources declaratively, while Ansible is used for configuration management and automation tasks. Terraform maintains a state file to track changes, enabling resource lifecycle management. Ansible uses an imperative approach, describing the steps needed to achieve the desired state. Terraform is cloud-agnostic, whereas Ansible works better for managing server configurations after infrastructure is provisioned. Both can complement each other in DevOps workflows.

4. **What is Terraform State? Why is it important?**
   Terraform State is a critical file that tracks the current status of your infrastructure managed by Terraform. It acts as a single source of truth, allowing Terraform to determine what resources exist and their configuration. The state file helps in resource dependency management and change detection. It supports team collaboration by enabling remote backends like S3 or Terraform Cloud. Proper state management is essential to prevent data corruption and ensure smooth Terraform operations.

5. **What are the different types of variables in Terraform?**
   Terraform supports three types of variables:

My Last two Months Interview Questions

Note - These all questions are from various top product based companies, Please take this as an challenge and complete all questions including coding in python🙂

If you are not able to complete the coding in python then go over the solution in internet

Thanks
Singam

Zeta - Meal cards / HDFC / Payments gateway

1) How the pods are communicating with each other
2) You have 500GB data how you will store in S3 and what steps you will take to make sure you have time, cost and security implemented
3) What is vpc endpoint
4) In other words, the function needs to find out if we can get the sequence array from the array, when we delete some or no elements in the array without changing the order of the remaining elements.
array: [3, 1,1, 7, 5, 10, 2];
sequence: [1,1, 5, 2]
Output: true
array: [3,1,7,5,10,2]
sequence: [1,3]
output: false

Terraform modules are a crucial part of managing scalable and reusable infrastructure as code.

1. What is a Terraform module and why are they important?

A Terraform module is a collection of configuration files that manage a set of related resources as a single unit. It is essentially a reusable package of Terraform configuration that can be used to simplify complex infrastructure setups and promote best practices.

Components of a Terraform Module

A typical Terraform module consists of the following files:

1. `main.tf`: Contains the primary set of resource configurations.
2. `variables.tf`: Defines input variables that the module can accept to customize its behavior.
3. `outputs.tf`: Defines the values that the module will output after execution, which can be used by other modules or configurations.
4. `providers.tf`: Specifies the providers the module uses (optional but often included for completeness).

Importance of Terraform Modules

1. **Reusability**:
   - Modules allow you to encapsulate common patterns and configurations, making it easy to reuse code across different projects or environments. This reduces the amount of code duplication and improves maintainability.
2. **Abstraction**:
   - By using modules, you can abstract complex infrastructure configurations into simple, reusable components. This helps in managing infrastructure as code (IaC) more efficiently and makes it easier for teams to understand and use the configurations without needing to dive into the complexities.
3. **Consistency**:
   - Modules help enforce best practices and standardize the way resources are configured and managed. This consistency ensures that infrastructure is provisioned in a predictable and reliable manner.
4. **Collaboration**:
   - Teams can share modules across the organization, enabling collaboration and reducing the effort required to set up common infrastructure components. This fosters a culture of shared knowledge and infrastructure as code.
5. **Maintainability**:
   - By breaking down infrastructure into smaller, manageable modules, you can more easily update, maintain, and troubleshoot configurations. This modular approach also allows for easier testing and validation of individual components.

1. What do you know about AWS Region?

Answer: An AWS Region is a completely independent entity in a geographical area. There are two more Availability Zones in an AWS Region.

Within a region, Availability Zones are connected through low-latency links.

Since each AWS Region is isolated from another Region, it provides very high fault tolerance and stability.

For launching an EC2 instance, we have to select an AMI within the same region.

2. What are the important components of IAM?

Answer: The important components of IAM are as follows:

IAM User: An IAM user is a person or service that will interact with AWS. User can sign in to AWS Management Console for performing tasks in AWS.

IAM Group: An IAM Group is a collection of IAM users. We can specify permission to an IAM Group. This helps in managing a large number of IAM users. We can simply add or remove an IAM User to an IAM Group to manage the permissions.

IAM Role: An IAM Role is an identity to which we give permissions. A Role does not have any credentials (password or access keys). We can temporarily give an IAM Role to an IAM User to perform certain tasks in AWS.

IAM Permission: In IAM we can create two types of Permissions. Identity-based and Resource-based. We can create a Permission to access or perform an action on an AWS Resource and assign it to a User, Role or Group. We can also create Permissions on resources like S3 bucket, Glacier vault etc and specify who has access to the resource.

IAM Policy: An IAM Policy is a document in which we list permissions to specify Actions, Resources, and Effects. This document is in JSON format. We can attach a policy to an IAM User or Group.

TOP 250+ Interviews Questions on AWS

Q1) What is AWS?

Answer:AWS stands for Amazon Web Services. AWS is a platform that provides on-demand resources for hosting web services, storage, networking, databases and other resources over the internet with a pay-as-you-go pricing.

Q2) What are the components of AWS?

Answer:EC2 – Elastic Compute Cloud, S3 – Simple Storage Service, Route53, EBS – Elastic Block Store, Cloudwatch, Key-Paris are few of the components of AWS.

Q3) What are key-pairs?

Answer:Key-pairs are secure login information for your instances/virtual machines. To connect to the instances we use key-pairs that contain a public-key and private-key.

Q4) What is S3?

Answer:S3 stands for Simple Storage Service. It is a storage service that provides an interface that you can use to store any amount of data, at any time, from anywhere in the world. With S3 you pay only for what you use and the payment model is pay-as-you-go.

Q5) What are the pricing models for EC2instances?

Answer:The different pricing model for EC2 instances are as below,

- On-demand
- Reserved
- Spot
- Scheduled
- Dedicated

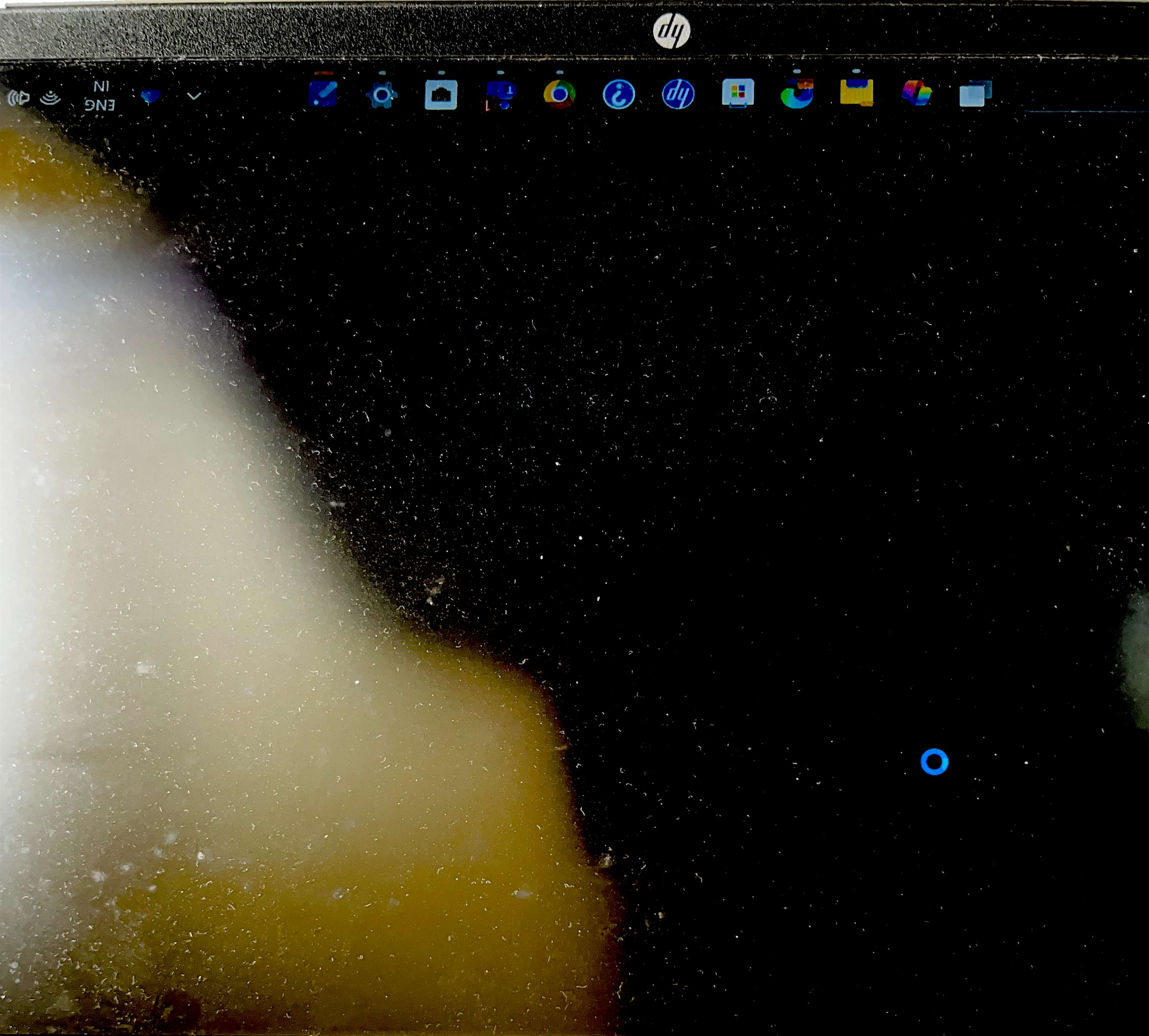
Q6) What are the types of volumes for EC2 instances?

Question 1: What are your daily responsibilities in your current project?

Answer:

In my current role as a DevOps Engineer, my daily responsibilities involve a blend of infrastructure management, automation, and collaboration to ensure smooth and efficient operations.

- **Monitoring and Maintenance:**
  - **Health Checks:** Start the day by reviewing health dashboards and monitoring alerts using tools like **Prometheus** and **Grafana**.
  - **Issue Mitigation:** Prioritize troubleshooting any detected anomalies to prevent impact on end-users.
- **CI/CD Pipeline Management:**
  - **Automation:** Work with **Jenkins** and **GitLab CI** to streamline the deployment process.
  - **Pipeline Development:** Write and maintain pipeline scripts, integrate automated testing, and ensure smooth deployments with rollback strategies.
- **Infrastructure as Code (IaC):**
  - **Provisioning:** Use **Terraform** and **Ansible** to provision and manage AWS cloud resources.
  - **Resource Management:** Configure EC2 instances, manage VPCs, set up S3 buckets, and handle IAM roles and permissions.
- **Security and Compliance:**
  - **System Updates:** Regularly update and patch systems, manage SSL certificates.
  - **Best Practices:** Enforce security best practices, conduct regular audits, and address vulnerabilities with the security team.
- **Performance Optimization:**
  - **Analysis:** Analyze system performance metrics to optimize resource utilization.
  - **Optimization:** Adjust autoscaling policies, optimize database queries, refine caching strategies to improve responsiveness and reduce costs.
- **Collaboration:**
  - **Team Alignment:** Participate in daily stand-ups and planning meetings using Agile methodologies.
  - **Cross-Functional Work:** Ensure alignment with development teams on deployment strategies and environment consistency.
- **On-Call Support:**
  - **Availability:** Participate in on-call rotations to provide 24/7 support for critical issues.



PROJECT1 👍

***0-14 years of exp

**INTERVIWER/HR will not check your projects with your current organization

JENKINS CICD PIPELINE:

Client Name: Mercedes Benz

Project Name : DMS [Dealer Management System]

Mercedes Benz dealer onboarding applications were handled by the DevOps/SRE team, the 16 microservices were onboarded into the CICD pipeline with end to end integration of all tools where we had various suites like Security, build, docker, ansible and k8 deployment.The Jenkins shared library along with the Jenkinsfile design for each microservice was done by both in an individual contribution and as a team work.

- Integrated all tools with the Jenkins and designed various automation scripts in shell and python to check the various stages conditions in pipeline
- Sonarqube QG was checked with shell script for each module and each microservice was on boarded to sonar
- Automation of GITLAB/SONAR/JENKINS/JIRA rest apis was done where I have received appreciation emails
- Maintained the build servers of jenkins where we had docker installed which converted the Jar into image
- Jfrog rest api automation for cleanup of old binaries was done
- Kubernetes manifest file design as per the development teams was handled where we had various kinds of deployments, daemonset, services etc.
- Troubleshooting the pod deployment along with RCA was performed

ROLES AND RESPONSIBILITIES PART2

***0-14 years of exp

**INTERVIEWER/HR will not check your projects with your current organisation

DEVOPS/LINUX SYSTEM ADMINISTRATION:

Client Name: Zeta

Project Name : Payzapp

HDFC Bank's Payzapp is an online payment app that doubles as a digital wallet and a virtual card. You can use PayZapp to pay for online and offline services like paying bills, sending money, shopping online, etc., without having to use your bank account or cards.

- Maintaining the linux centos servers of Payzapp by taking care of OS patching and software upgrades with the help of ansible playbooks

- Disk management and volume management were performed on all the servers to support the server uptime

- Log optimization scripts were deployed with the help of ansible on all the Payzapp servers to make sure the server logs are archived and stored in s3

- The user management and the log management along with the SELINUX was setup to make sure the enhanced security is provided to the servers.

- Performing systems administration, maintenance, and engineering in multiple large enterprise server environments (Linux, Red Hat, CentOS)

- Maintain Linux-based servers, both hardware and software

- Manage in-house Java developed applications that run on the Linux systems

- Demonstrate knowledge of essential network services such as DNS, NFS, Apache, NTP, sendmail, and OpenSSH

ANSIBLE HANDSON

(Push Based Methodology)

Step 1 - Create the EC2 instance with the below data

Type - LINUX AMAZON
Size - 8 GB T2.MICRO
Region - As per your choice

Step 1.1 - Install Ansible and check for python and git setup

yum install ansible - y

python3 -v

git -v

Step 2 - Clone the code from this repo

https://github.com/praveen1994dec/Ansible.git

Step 3 - Execute each set of playbook from the above repo

https://github.com/praveen1994dec/Ansible/blob/main/write_content_to_file.yml

ansible-playbook write_content_to_file.yml

ANSIBLE

(Push Based Methodology)

Ansible works on push based methodology. Ansible is agentless such that it requires nothing to be installed on the target host machine except ssh connection and python.

Difference between chef and ansible

Chef →

- It works with client server architecture.
- It works with ruby.
- We need to register each and every host individually to the host machine.
- Chef is more secure than ansible.
- It is difficult to set up.

Ansible →

- It works with push based methodology. It works with python.
- It supports dynamic inventory.
- Ansible is not secure by default. We need to add an external module called vault.
- It is easy to set up.

RAW Module

It is also called a dirty module. If there is no python in the target machine we use raw module to install python or run it by some shell commands.

Ansible Inventory (/etc/ansible/hosts)

We have 2 types of inventory

→ **Static Inventory**

→ **Dynamic Inventory**

Static Inventory:

Static inventory is a file which contains the list of IP addresses of target host machines and groups them together based on user requirements in json or ini format. Default location of inventory is **/etc/ansible/hosts**

Dynamic Inventory :

Ansible team will provide two files, python script and ini file. When we execute the script it will automatically fetch the address of target host machines and store it in the inventory file. Only thing we have to update is the path of the inventory file in the python script.

Ansible

What is Ansible?

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation. It is designed to simplify the process of managing large-scale IT environments, making it easier to deploy and manage applications and infrastructure consistently.

Create Environment to Practice Ansible

- **Create two ec2 instance ansible server and node server**
- **To install ansible in ansible server**

Userdata script:

```
#!/bin/bash
sudo su
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible -y
```

- **Connect to the ec2 instances**
- **Generate a key for passwordless ssh on both servers**

```
ssh-keygen
cd ~/.ssh/
```

Copy public key of ansible server and paste in node server

Ansible server:

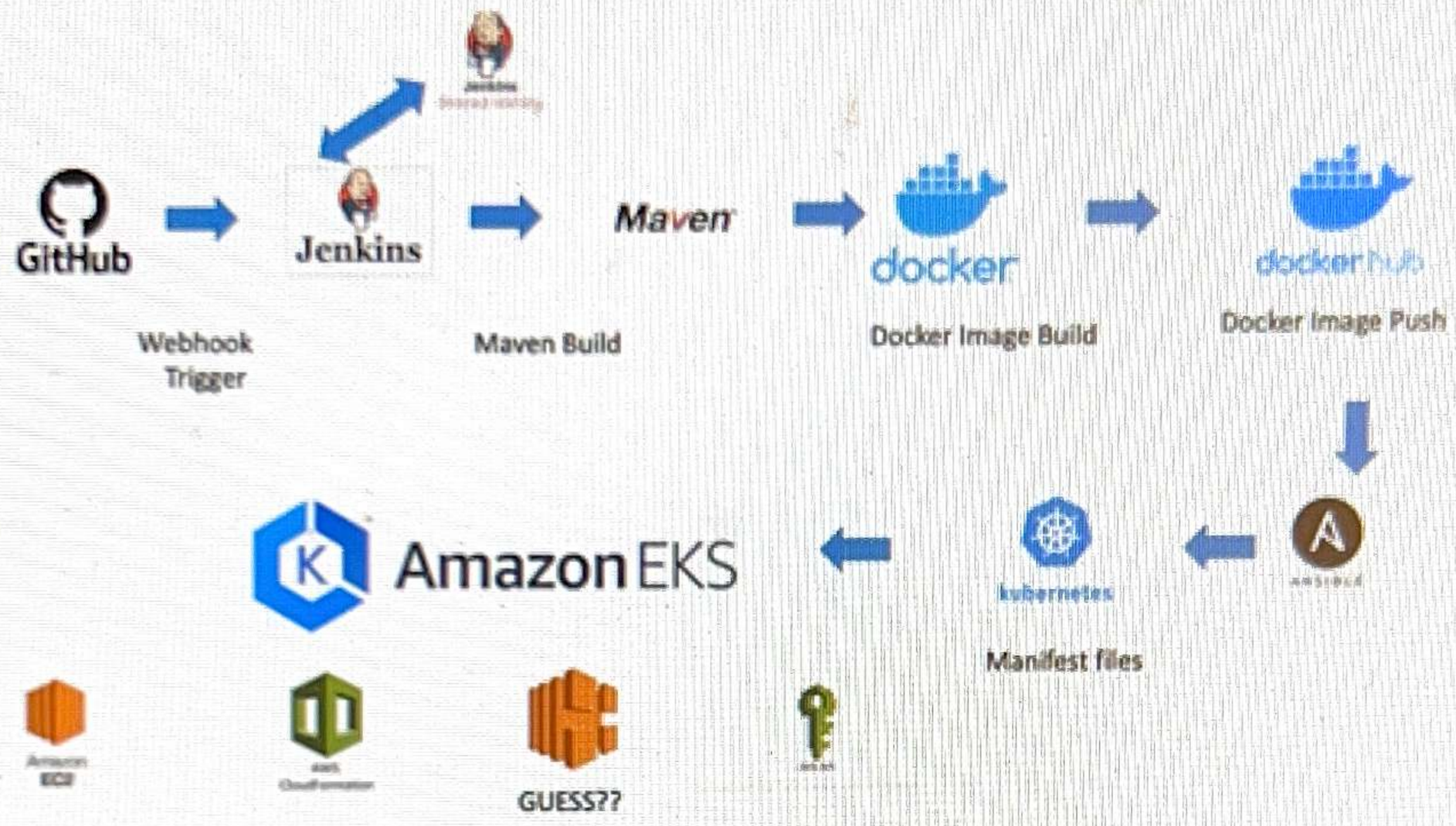
```
cat <public-key> #copy key
```

node server:

```
sudo vi authorized_keys # paste key in the file and save
```

- Configure Ansible server

```
cd /etc/ansible
ls
sudo vi hosts
[webserver] #node group name
<node-ip>
```



https://github.com/DEVOPS-WITH-WEB-DEV/jenkins-shared-library1.git

https://github.com/DEVOPS-WITH-WEB-DEV/spring-cloud-kubernetes.git

PREREQUISITES

1) AWS LOGIN
2) DOCKER HUB LOGIN
3) GITHUB HUB LOGIN

**** COST WILL OCCUR IN AWS ACCOUNT***

Ansible Definition

Ansible is a powerful, open-source automation tool designed to streamline IT tasks such as configuration management, application deployment, and orchestration. Its core strengths are its simplicity, scalability, and ease of use. Automation tasks in Ansible are defined using human-readable YAML files called Playbooks.

Key Features of Ansible

Agentless Operation

Ansible does not require additional software or agents on managed systems. It connects using SSH for Linux/Unix or WinRM for Windows.

Push-based Model

The control node sends commands directly to the managed nodes, ensuring efficient task execution.

Platform Independence

Ansible works seamlessly across diverse environments—cloud, on-premises, or hybrid.

Security by Design

Uses OpenSSH for secure communication and avoids storing sensitive information by default.

Benefits of Using Ansible

- Automates repetitive IT tasks, saving time and effort.
- Reduces human error in configuration and system setup.
- Supports multiple operating systems and environments.
- Simple to learn, making it accessible even for teams new to automation.

Installation Steps

PHASE3 IQ WITH HANDS-ON TASKS

Note - These Questions have been shared from your 6.0 Students Community

1)What are the things that you check for approval of PR requests on Github?
2)Before triggering the pipeline and checking the sonarqube report for analysis, at code or git level what does the pipeline check?
3)Possible security measures that can be taken before CI pipeline gets triggered?
4)How would you optimise CI/CD pipeline for large-scale apps to reduce build and deployment time?
5) Is the webhook trigger on git applied for all branches or for all pushes done by the developer?
6) OS and its troubleshooting
7) Networking and its troubleshooting

Jenkins

1) How do you set up a jenkins server in your organisation, can you walk me through?
2)What is the memory capacity and size of servers for jenkins master, slave, sonarqube, maven,docker, kubernetes, Prometheus, Grafana?
3)what database you use in your organisation, what work you do with database, how to
4)secure database, what disaster recovery strategy do you follow?
What strategy follows for Jenkins master slave architecture in your organisation?
5)Where do you store credentials for jenkins in your organisation?
6)What type of authentication is used for Jenkins in your organisation?
7)What is Jenkins' single sign on ?
8)how to troubleshoot and resolve failed builds or jobs in Jenkins

Github :