**Ansible Setup**

**What is Ansible?**

Ansible is an open-source IT automation tool that simplifies configuration management, application deployment, and orchestration of infrastructure. It uses a declarative language (YAML) to define tasks, ensuring systems are in a desired state.

**Main Role of Ansible:**

1. **Configuration Management**: Ensure systems are consistently configured (e.g., installing and configuring a web server).
2. **Application Deployment**: Automate the deployment of applications (e.g., deploying a containerized app on Kubernetes).
3. **Infrastructure Provisioning**: Provision virtual machines, containers, or cloud resources.
4. **Orchestration**: Automate workflows involving multiple components (e.g., setting up a database, web server, and load balancer).

**Key Features of Ansible:**

1. Agentless: No need to install any agents on the target systems; it communicates over SSH (Linux) or WinRM (Windows).
2. Declarative: You define the desired state, and Ansible ensures the system matches that state.
3. Scalable: Manage infrastructure from a single server to thousands of servers.
4. Extensible: Supports modules and plugins for various systems (e.g., cloud providers, networking devices).

**Ansible vs. Terraform**

**A screenshot of a computer

Description automatically generated**

**How Ansible Works on VMs and Kubernetes**

On Virtual Machines:

1. Communication: Uses SSH for Linux VMs or WinRM for Windows.
2. Playbooks: Defines tasks to configure or deploy software on the VMs.

**What is an Ansible Playbook?**

An Ansible **playbook** is a YAML file that defines a set of tasks to be executed on one or more target machines. It describes the desired state of a system or application and automates the process of achieving that state.

**Key Components of a Playbook:**

1. **Hosts**: Defines the target machines on which the tasks will run.
2. **Tasks**: A sequence of actions to be executed (e.g., installing software, starting services).
3. **Modules**: Reusable units (like plugins) that perform specific tasks (e.g., apt, yum, kubernetes.core.k8s).
4. **Handlers**: Special tasks triggered only when notified (e.g., restart a service after configuration changes).
5. **Variables**: Used to make playbooks dynamic and reusable.

**Installing and configuring Nginx on 10 VMs**

---

- name: Install and Start Nginx

hosts: webservers

become: true # Run tasks with sudo privileges

tasks:

- name: Update apt cache

apt:

update\_cache: yes

- name: Install Nginx

apt:

name: nginx

state: present

- name: Start and enable Nginx

service:

name: nginx

state: started

enabled: true

**ansible-playbook -i inventory.yml nginx\_setup.yml**

**How It Works**

1. **Play Name**: "Install and Start Nginx": Describes the goal of this play.
2. **Hosts**: "webservers": Specifies the target group of servers (defined in the inventory file).
3. **become: true**: Ensures tasks run with root privileges.
4. **Tasks**:
   * **Update apt cache**: Updates the package manager cache.
   * **Install Nginx**: Ensures Nginx is installed.
   * **Start and enable Nginx**: Ensures Nginx is running and will restart automatically on reboot.

**Ansible – Setup & Configure**

**Prequisites**

sudo apt update -y

sudo apt update

sudo apt install git -y

sudo apt install maven -y

mvn -version

apt install openjdk-17-jdk -y

java -version

sudo apt install update

sudo apt install curl -y

sudo apt install iputils-ping -y

sudo apt-get install python3 -y

sudo apt-get install python3-pip -y

sudo apt update

sudo apt install openssh-client

**Ansible -Tower ( Ubuntu Machine)**

sudo apt install software-properties-common -y

sudo apt install ansible -y

sudo add-apt-repository ppa:ansible/ansible --yes --update

sudo apt install ansible-core -y

ansible --version

**On the Control Node and Target Node:**

# Verify Ansible installation

ansible --version

# Create Ansible user and set up sudo permissions

sudo adduser --disabled-password --gecos "" ansible

sudo usermod -aG sudo ansible

vi /etc/sudoers

# Ansible user with no password for sudo

ansible ALL=(ALL) NOPASSWD:ALL

**Add the ansible User to the sudo Group** (optional, but a good practice):

sudo usermod -aG sudo ansible

**2. Set Up SSH Access for the Ansible User**

su – ansible

ssh-keygen -t rsa -b 4096

cat ~/.ssh/id\_rsa.pub ( Copy the public key)

ssh-rsa  ansible@vm1

**Grant Write Access**: To allow the ansible user to create a directory, you can either:

sudo chmod u+w /home/dhrangdhariaritesh

**3. Configure Ansible to Use the ansible User**

sudo vi /etc/ansible/ansible.cfg

[defaults]

remote\_user = ansible

become = True

become\_method = sudo

cat ~/.ssh/id\_rsa.pub ( Copy the public key)

4. **Edit the Inventory File**:

**Ensure your inventory file is set up to use the ansible user:**

sudo vi /etc/ansible/hosts

[local]

localhost ansible\_connection=local

[mytargets]

vm2 ansible\_host=10.128.0.52 ansible\_user=user

**Ansible -Target ( Ubuntu Machine)**

--Create ansible user (you can also use root user , best practice is to use separate user , i.e ansible user)

**# Verify Ansible installation**

**ansible --version**

**# Create Ansible user and set up sudo permissions**

**sudo adduser --disabled-password --gecos "" ansible**

**sudo usermod -aG sudo ansible**

**vi /etc/sudoers**

**ansible ALL=(ALL) NOPASSWD: ALL**

**sudo usermod -aG sudo ansible**

**su - ansible**

**mkdir -p ~/.ssh**

**chmod 700 ~/.ssh**

**touch ~/.ssh/authorized\_keys**

**chmod 600 ~/.ssh/authorized\_keys**

**sudo chmod 700 /home/ansible/.ssh**

**sudo chmod 600 /home/ansible/.ssh/authorized\_keys**

**sudo chown -R ansible:ansible /home/ansible/.ssh**

**sudo vi /etc/ssh/sshd\_config**

**Make sure the following lines are set (uncomment them if necessary):**

**PubkeyAuthentication yes**

**AuthorizedKeysFile .ssh/authorized\_keys**

**PubkeyAuthentication yes**

**sudo systemctl restart ssh**

**vi ~/.ssh/authorized\_keys**

**--Paste the public key from control vm into the file and save it.--**

**sudo systemctl restart ssh**

**ls -ld ~/.ssh**

**ls -l ~/.ssh/authorized\_keys**

**cat ~/.ssh/authorized\_keys**

**------Validation from Control------**

**ssh ansible@10.128.0.20**

**Type 'Yes'**

**ssh-copy-id** [**ansible@10.150.0.22**](mailto:ansible@10.150.0.22)

**ansible all -m ping**

**ansible all --list-hosts**

**ansible targets -m ping**

**ansible local -m ping**

**Test Passwordless Sudo Access**

On the Target Node:

1. Switch to the ansible User:

**su - ansible**

1. Test Sudo Without a Password: Run a command that requires sudo, such as:

**sudo whoami**

The command should output root without prompting for a password. If it does, the passwordless sudo is correctly configured.

**Run to Ansible Target VM**

**Create inventory.yml or using existing host**

all:

hosts:

vm2:

ansible\_host: 10.128.0.56

ansible\_user: ansible # Replace 'user' with the SSH username for vm2

OR

sudo vi /etc/ansible/hosts

[mytargets]

vm2 ansible\_host=10.128.0.52 ansible\_user=user

ansible -i /etc/ansible/hosts mytargets -m ping

**3. Run a Ping Test**

Use the Ansible ping module to test the connection to vm2:

**ansible -i inventory.yml all -m ping**

vm2 | SUCCESS => {

"changed": false,

"ping": "pong"

}

ansible -i inventory.yml all -a "ls -l"

1. **Run a Test Playbook**

**test.yml**

---

- name: Install and configure Nginx

hosts: all

become: yes

tasks:

- name: Update the package index

apt:

update\_cache: yes

- name: Install Nginx

apt:

name: nginx

state: present

- name: Ensure Nginx is enabled and running

service:

name: nginx

state: started

enabled: yes

**Detailed Explanation of Each Section**

1. **Playbook Header**

- name: Install and configure Nginx

* + name: A description of the playbook for readability in Ansible output.

hosts: all

* + hosts: Specifies the target group (all in this case) from the inventory file.

become: yes

* + become: Elevates privileges to sudo for tasks requiring root access.

**Tasks Section**

**Task 1: Update the Package Index**

- name: Update the package index

apt:

update\_cache: yes

* + - name: A description of the task for readability.
    - apt: The Ansible module for managing packages on Debian-based systems.
    - update\_cache: Refreshes the local package cache to ensure the latest package metadata.

**Task 2: Install Nginx**

- name: Install Nginx

apt:

name: nginx

state: present

* + - name: Description of the task.
    - apt:
      * name: Specifies the package name to install (nginx).
      * state: Ensures the package is installed (present).

**Task 3: Enable and Start Nginx**

- name: Ensure Nginx is enabled and running

service:

name: nginx

state: started

enabled: yes

**Explanation**

name: Describes the task.

service: The Ansible module to manage services.

name: Specifies the service name (nginx).

state: Ensures the service is running (started).

enabled: Ensures the service starts on boot (yes).

**Run the Playbook**

**ansible-playbook -i inventory.yml setup\_nginx.yml**

**ansible-playbook -i inventory.yml test.yml --ask-become-pass**

**or**

**ansible-playbook -i /etc/ansible/hosts test.yml**

**Verify the Playbook Execution**

ssh user@10.128.0.52

sudo systemctl status nginx

**Revert the changes**

ansible all -i inventory.yml -m service -a "name=nginx state=stopped"

ansible all -i inventory.yml -m service -a "name=nginx enabled=no"

ansible all -i inventory.yml -m apt -a "name=nginx state=absent"

ansible all -i inventory.yml -m file -a "path=/etc/nginx state=absent"

**RUN Kuberntes from VM**

gcloud auth login

gcloud container clusters get-credentials cluster-1 --zone us-central1-c --project artful-talon-443506-d1

kubectl get nodes

Install Required Dependencies:

ansible-galaxy collection install community.kubernetes

pip3 install kubernetes google-auth

**Verify the Kubernetes Configuration File**

**cat ~/.kube/config**

**ls -l ~/.kube/config**

**chmod 600 ~/.kube/config**

**kubectl get nodes**

**export KUBECONFIG=~/.kube/config**

**2 Ways**

1. **Create deployment file and create ansible playbook to use that**
2. **Create ansible playbook by adding deployment code directly.**

**Approach 1 :** **nginx-deployment.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

namespace: default

labels:

app: nginx

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: nginx-service

namespace: default

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

**Update the Ansible Playbook**

---

- name: Deploy Nginx to Kubernetes

hosts: localhost

tasks:

- name: Apply Nginx Deployment

kubernetes.core.k8s:

kubeconfig: ~/.kube/config

namespace: default

state: present

definition: "{{ lookup('file', 'nginx-deployment.yml') }}"

ansible-playbook deploy\_nginx.yml

kubectl get pods -n default

kubectl get svc -n default

**Approach 2 : Create a Deployment Playbook**

Create sre\_automation.yml to handle deployment, monitoring setup, scaling, and incident response:

---

- name: Deploy Nginx Application on Kubernetes

hosts: localhost

tasks:

- name: Create Nginx Deployment

kubernetes.core.k8s:

kubeconfig: ~/.kube/config

state: present

definition:

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

namespace: default

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

- name: Expose Nginx Deployment as a Service

kubernetes.core.k8s:

kubeconfig: ~/.kube/config

state: present

definition:

apiVersion: v1

kind: Service

metadata:

name: nginx-service

namespace: default

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

ansible-galaxy collection install kubernetes.core

pip3 install kubernetes google-auth

**ansible-playbook deploy\_nginx\_k8s.yml**

**kubectl get deployments**

**kubectl get pods**

**kubectl get svc**