

Master of Science (Computer Application) (M.Sc. CA) Programme

M.Sc. CA Sem - I

AY 2024-25

104-01 : Web Development Operations

***by***

|  |  |  |
| --- | --- | --- |
| Seat No. | Roll No. | Name of Student |
|  | 01 | Jenil Nareshkumar Ambawala |

|  |
| --- |
| **Submitted to :** |
| Prof. Tushar Bhadak |

# Create an Ansible playbook that installs Python, Node.js, and Apache Web Server on remote machines. Test the playbook on a target server and document the process.

## Answer:-

---

# playbook.yml

* name: Install Python, Node.js, and Apache Web Server hosts: web\_servers

become: true # Run commands as sudo tasks:

* + name: Update apt package list (Debian/Ubuntu) ansible.builtin.apt:

update\_cache: yes

when: ansible\_os\_family == "Debian"

* + name: Install Python ansible.builtin.package:

name: python3 state: present

* + name: Install pip (Python Package Manager) ansible.builtin.package:

name: python3-pip state: present

* + name: Install Node.js (Debian/Ubuntu) ansible.builtin.apt:

name:

* + - nodejs

* + - npm

state: present

when: ansible\_os\_family == "Debian"

* + name: Install Node.js (RedHat/CentOS) ansible.builtin.yum:

name:

* + - nodejs
    - npm

state: present

when: ansible\_os\_family == "RedHat"

* + name: Install Apache Web Server (Debian/Ubuntu) ansible.builtin.apt:

name: apache2 state: present

when: ansible\_os\_family == "Debian"

* + name: Install Apache Web Server (RedHat/CentOS) ansible.builtin.yum:

name: httpd state: present

when: ansible\_os\_family == "RedHat"

* + name: Ensure Apache is started and enabled (Debian/Ubuntu) ansible.builtin.systemd:

name: apache2 state: started enabled: yes

when: ansible\_os\_family == "Debian"

* + name: Ensure Apache is started and enabled (RedHat/CentOS)

ansible.builtin.systemd:

name: httpd state: started enabled: yes

when: ansible\_os\_family == "RedHat"

## Running the playbook:-

ansible-playbook -i inventory.ini playbook.yml

# Develop an Ansible playbook that demonstrates multiple ways to create variables and use it.

## Answer:-

---

# variables\_demo.yml

* name: Demonstrating Variable Usage in Ansible hosts: localhost

gather\_facts: false vars:

# Inline variables inside playbook inline\_var: "This is an inline variable"

# List variable my\_list:

* + item1
  + item2
  + item3

# Dictionary variable my\_dict:

name: "Ansible" version: "2.10"

# Variable based on expressions sum\_of\_numbers: "{{ 5 + 3 }}"

vars\_files:

* vars/external\_vars.yml

tasks:

* name: Print inline variable ansible.builtin.debug:

msg: "{{ inline\_var }}"

* name: Print list variable ansible.builtin.debug:

msg: "{{ my\_list }}"

* name: Print dictionary variable ansible.builtin.debug:

msg: "Name: {{ my\_dict.name }}, Version: {{ my\_dict.version }}"

* name: Use variable from expression ansible.builtin.debug:

msg: "The sum of 5 and 3 is {{ sum\_of\_numbers }}"

* name: Print external variable ansible.builtin.debug:

msg: "{{ external\_var }}"

* name: Print variable from inventory group ansible.builtin.debug:

msg: "{{ group\_var }}"

* name: Print host-specific variable ansible.builtin.debug:

msg: "{{ host\_var }}"

* name: Use command-line variable ansible.builtin.debug:

msg: "{{ cli\_var }}"

## Running the playbook:-

ansible-playbook -i inventory.ini variables\_demo.yml --extra-vars "cli\_var=This is a CLI variable"

# Create an Ansible playbook that demonstrates how to create handlers?

## Answer:-

---

# handlers\_demo.yml

* name: Demonstrating Handlers in Ansible hosts: localhost

gather\_facts: false tasks:

* + name: Create a configuration file ansible.builtin.copy:

content: "Configuration data" dest: /tmp/sample\_config.conf notify: "Restart Apache"

* + name: Ensure Apache is installed ansible.builtin.apt:

name: apache2 state: present update\_cache: yes

notify: "Restart Apache"

when: ansible\_os\_family == "Debian"

* + name: Ensure Apache is installed (RedHat/CentOS) ansible.builtin.yum:

name: httpd state: present

notify: "Restart Apache"

when: ansible\_os\_family == "RedHat"

handlers:

* + name: Restart Apache ansible.builtin.systemd:

name: apache2 state: restarted

when: ansible\_os\_family == "Debian"

* + name: Restart Apache (RedHat/CentOS) ansible.builtin.systemd:

name: httpd state: restarted

when: ansible\_os\_family == "RedHat"

## Running the playbook:-

ansible-playbook -i inventory.ini handlers\_demo.yml

# Create an Ansible playbook that handles multiples methods of creating and using Roles?

## Answer:-

1. **Create Role: Apache**

* roles/apache/tasks/main.yml:

```yaml

---

# roles/apache/tasks/main.yml

* name: Install Apache ansible.builtin.apt:

name: apache2 state: present update\_cache: yes notify: Restart Apache

* name: Copy Apache config template ansible.builtin.template:

src: apache2.conf.j2

dest: /etc/apache2/apache2.conf notify: Restart Apache

* name: Copy sample index.html ansible.builtin.copy:

src: sample\_index.html

dest: /var/www/html/index.html mode: '0644'

```

## - \*\*roles/apache/handlers/main.yml\*\*:

```yaml

---

# roles/apache/handlers/main.yml

- name: Restart Apache ansible.builtin.systemd: name: apache2

state: restarted

```

## - \*\*roles/apache/templates/apache2.conf.j2\*\*:

```jinja

# roles/apache/templates/apache2.conf.j2 # Basic Apache configuration ServerName localhost

DocumentRoot /var/www/html

<Directory /var/www/html>

Options Indexes FollowSymLinks AllowOverride None

Require all granted

</Directory>

```

## \*\*roles/apache/files/sample\_index.html\*\*:

```html

<!-- roles/apache/files/sample\_index.html -->

<html>

<head>

<title>Welcome to Apache!</title>

</head>

<body>

<h1>Hello from Ansible Apache Role!</h1>

</body>

</html>

```

## Create Role: Node.js

* **\*\*roles/nodejs/tasks/main.yml\*\*:**

```yaml

---

# roles/nodejs/tasks/main.yml

* name: Install Node.js and npm ansible.builtin.apt:

name:

* + "{{ nodejs\_package }}"
  + "{{ npm\_package }}" state: present

tags:

- nodejs

```

* \*\*roles/nodejs/vars/main.yml\*\*:

```yaml

---

# roles/nodejs/vars/main.yml nodejs\_package: nodejs npm\_package: npm

```

## Create Role: Python

**- \*\*roles/python/tasks/main.yml\*\*:**

```yaml

---

# roles/python/tasks/main.yml

* name: Install Python and pip ansible.builtin.apt:

name:

* + "{{ python\_package }}"
  + "{{ pip\_package }}" state: present

```

* \*\*roles/python/defaults/main.yml\*\*:

```yaml

---

# roles/python/defaults/main.yml python\_package: python3 pip\_package: python3-pip

```

## Playbook

```yaml

---

# playbook.yml

* name: Use Apache, Node.js, and Python Roles hosts: localhost

become: true

roles:

* + role: apache tags: apache
  + role: nodejs tags: nodejs
  + role: python tags: python

```

## Running the playbook:-

```bash

ansible-playbook -i inventory.ini playbook.yml

```

# Develop an Ansible playbook that handles Control Structures?

## Answer:-

---

# control\_structures.yml

* name: Playbook Demonstrating Control Structures hosts: localhost

gather\_facts: false vars:

package\_list:

* + vim
  + git
  + curl is\_apache\_needed: true

sample\_file\_path: "/tmp/sample\_file.txt" user\_list:

* + name: "user1" state: "present"
  + name: "user2" state: "absent"

tasks:

### CONDITIONAL TASKS ###

* name: "Install Apache when it is required" ansible.builtin.apt:

name: apache2 state: present

when: is\_apache\_needed tags: apache

* name: "Remove Apache if not needed"

ansible.builtin.apt:

name: apache2 state: absent

when: not is\_apache\_needed tags: apache

### LOOPS ###

* name: "Install multiple packages" ansible.builtin.apt:

name: "{{ item }}" state: present

loop: "{{ package\_list }}" tags: packages

* name: "Create or delete users based on their state" ansible.builtin.user:

name: "{{ item.name }}"

state: "{{ item.state }}"

loop: "{{ user\_list }}" tags: users

### BLOCKS WITH ERROR HANDLING ###

* block:
  + name: "Create a sample file" ansible.builtin.file:

path: "{{ sample\_file\_path }}" state: touch

* + name: "Write content to the sample file" ansible.builtin.copy:

content: "This is a sample file" dest: "{{ sample\_file\_path }}"

* + name: "Fail this task deliberately" ansible.builtin.command:

cmd: "/bin/false"

rescue:

* + name: "Handle error by notifying" ansible.builtin.debug:

msg: "The previous task failed. Handling the error."

always:

* + name: "Ensure the file is removed" ansible.builtin.file:

path: "{{ sample\_file\_path }}" state: absent

## Running the playbook:-

ansible-playbook control\_structures.yml

# Create exception handling program in playbook.

## Answer:-

---

# exception\_handling.yml

* name: Playbook to demonstrate exception handling hosts: localhost

gather\_facts: false tasks:

### Block of tasks that might fail ###

* + block:
    - name: "Create a sample file" ansible.builtin.file:

path: "/tmp/sample\_file.txt" state: touch

notify: "Cleanup Sample File"

* + - name: "Write content to the sample file" ansible.builtin.copy:

content: "This is a sample file created by Ansible." dest: "/tmp/sample\_file.txt"

* + - name: "Deliberately fail this task" ansible.builtin.command:

cmd: "/bin/false" register: failure\_result ignore\_errors: false

rescue:

* + - name: "Handle the failure by notifying and reporting error" ansible.builtin.debug:

msg: "The task failed. The error: {{ failure\_result }}"

* + - name: "Send a notification of failure" ansible.builtin.debug:

msg: "Error handled. Proceeding with rescue."

always:

* + - name: "Always run this task to clean up" ansible.builtin.file:

path: "/tmp/sample\_file.txt" state: absent

* + - name: "Always notify, regardless of success or failure" ansible.builtin.debug:

msg: "The playbook has completed the execution. This task is executed always."

handlers:

* + name: "Cleanup Sample File" ansible.builtin.debug:

msg: "Cleanup action: Removing sample file after error handling."

## Running the playbook:-

ansible-playbook exception\_handling.yml

# Set up Jenkins on Kubernetes Engine.

## Answer:-

---

# setup\_jenkins\_on\_gke.yml

* name: Set Up Jenkins on Google Kubernetes Engine hosts: localhost

gather\_facts: no tasks:

* + name: Authenticate with Google Cloud command: >

gcloud auth activate-service-account --key-file={{ gcp\_service\_account\_key }} vars:

gcp\_service\_account\_key: "/path/to/your/service-account-key.json" # Update with your service account key file

* + name: Set project ID

command: gcloud config set project {{ gcp\_project\_id }} vars:

gcp\_project\_id: "your-gcp-project-id" # Update with your project ID

* + name: Create a GKE cluster command: >

gcloud container clusters create jenkins-cluster

--zone us-central1-c

--num-nodes 3

register: gke\_cluster\_creation until: gke\_cluster\_creation.rc == 0 retries: 3

delay: 60

* + name: Get credentials for the new cluster

command: gcloud container clusters get-credentials jenkins-cluster --zone us- central1-c

* + name: Install Helm command: >

curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |

bash

args:

warn: false

* + name: Add Jenkins Helm repository

command: helm repo add jenkins https://charts.jenkins.io args:

warn: false

* + name: Update Helm repositories

command: helm repo update

* + name: Create a namespace for Jenkins command: kubectl create namespace jenkins

ignore\_errors: yes # Ignore if the namespace already exists

* + name: Install Jenkins using Helm command: >

helm install jenkins jenkins/jenkins

--namespace jenkins

--set controller.serviceType=LoadBalancer register: jenkins\_installation

* + name: Wait for Jenkins service to be assigned an external IP command: kubectl get svc -n jenkins -o

jsonpath='{.status.loadBalancer.ingress[0].ip}' register: jenkins\_external\_ip

until: jenkins\_external\_ip.stdout | length > 0 retries: 10

delay: 30

* + name: Get Jenkins admin password command: >

kubectl exec --namespace jenkins -it svc/jenkins -c jenkins -- /bin/cat

/run/secrets/chart-admin-password register: jenkins\_admin\_password

* + name: Output Jenkins Information debug:

msg:

* + - "Jenkins is installed!"
    - "Access Jenkins at: http://{{ jenkins\_external\_ip.stdout }}:8080"
    - "Admin Password: {{ jenkins\_admin\_password.stdout }}"

## Running the playbook:-

ansible-playbook setup\_jenkins\_on\_gke.yml

# Create CI/CD with Jenkins in Kubernetes Engine.

## Answer:-

---

# ci\_cd\_jenkins\_gke.yml

* name: Set Up CI/CD with Jenkins on Google Kubernetes Engine hosts: localhost

gather\_facts: no vars:

gcp\_service\_account\_key: "/path/to/your/service-account-key.json" # Update with your service account key file

gcp\_project\_id: "your-gcp-project-id" # Update with your GCP project ID jenkins\_admin\_password: "admin" # Default admin password for Jenkins

tasks:

* + name: Authenticate with Google Cloud

command: gcloud auth activate-service-account --key-file={{ gcp\_service\_account\_key }}

* + name: Set project ID

command: gcloud config set project {{ gcp\_project\_id }}

* + name: Create a GKE cluster command: >

gcloud container clusters create jenkins-cluster

--zone us-central1-c

--num-nodes 3

register: gke\_cluster\_creation until: gke\_cluster\_creation.rc == 0 retries: 3

delay: 60

* + name: Get credentials for the new cluster

command: gcloud container clusters get-credentials jenkins-cluster --zone us- central1-c

* + name: Install Helm command: >

curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |

bash

args:

warn: false

* + name: Add Jenkins Helm repository

command: helm repo add jenkins https://charts.jenkins.io args:

warn: false

* + name: Update Helm repositories command: helm repo update
  + name: Create a namespace for Jenkins command: kubectl create namespace jenkins

ignore\_errors: yes # Ignore if the namespace already exists

* + name: Install Jenkins using Helm command: >

helm install jenkins jenkins/jenkins

--namespace jenkins

--set controller.serviceType=LoadBalancer

--set controller.adminPassword={{ jenkins\_admin\_password }} register: jenkins\_installation

* + name: Wait for Jenkins service to be assigned an external IP command: kubectl get svc -n jenkins -o

jsonpath='{.status.loadBalancer.ingress[0].ip}' register: jenkins\_external\_ip

until: jenkins\_external\_ip.stdout | length > 0 retries: 10

delay: 30

* + name: Get Jenkins admin password command: >

kubectl exec --namespace jenkins -it svc/jenkins -c jenkins -- /bin/cat

/run/secrets/chart-admin-password

register: jenkins\_admin\_password\_output

* + name: Output Jenkins Information debug:

msg:

* + - "Jenkins is installed!"
    - "Access Jenkins at: http://{{ jenkins\_external\_ip.stdout }}:8080"
    - "Admin Password: {{ jenkins\_admin\_password\_output.stdout }}"
  + name: Install Jenkins Plugins command: >

kubectl exec --namespace jenkins -it svc/jenkins -c jenkins -- jenkins-plugin-cli -

-plugins git pipeline

register: jenkins\_plugins\_installation

* + name: Create a sample pipeline job command: >

kubectl exec --namespace jenkins -it svc/jenkins -c jenkins -- curl -X POST -u admin:{{ jenkins\_admin\_password\_output.stdout }} -H "Content-Type: application/json" -d '{

"name": "Sample-Pipeline", "mode":

"org.jenkinsci.plugins.workflow.multibranch.WorkflowMultiBranchProject",

"pipeline": { "definition": {

"script": "pipeline { agent any; stages { stage('Build') { steps { echo 'Building...'; } } stage('Test') { steps { echo 'Testing...'; } } stage('Deploy') { steps { echo 'Deploying...'; } } } }"

}

}

}'

## Ruuning the playbook:-

ansible-playbook ci\_cd\_jenkins\_gke.yml