**Project Title: Automated Web Server Deployment Using Ansible**

**Description:** This project demonstrates the automation of web server deployment using Ansible, a popular configuration management tool. Ansible allows us to automate the process of configuring and deploying a web server along with hosting a simple web page.

**Real-World Application**

**For example:** A company wants to launch its website on 10 cloud servers. Instead of manually logging into each server to install the web server, configure settings, and deploy content, they use Ansible. By running an Ansible playbook, the setup is automated, and within minutes, all 10 servers are ready to host the website.

**Why is it Important?**

* Saves Time: Automating tasks reduces the time taken to configure servers.
* Consistency: Ensures every server is set up in the same way, avoiding configuration drift.
* Scalability: Easily deploy the same configuration on hundreds of servers.
* Error Reduction: Eliminates human errors caused by repetitive manual tasks

**Key Features**

* Automates web server installation and configuration (e.g., Nginx or Apache).
* Deploys a static HTML webpage (index.html).
* Supports multiple servers via an inventory file.
* Scales easily for dynamic infrastructures.

**Prerequisites of this project Automated Web Server Deployment Using Ansible**

* Linux Knowledge: Basic understanding of Linux commands.
* Ansible Knowledge: Understanding of how to write and execute Ansible playbooks.
* Python & SSH: Ensure Python is installed and SSH is configured on both control and target nodes.
* Web Server Software: Apache or Nginx installed and ready to be configured.
* Firewall Management: Ensure firewalls on target machines allow HTTP/HTTPS traffic.
* Target Server Access: SSH access to all servers you intend to deploy the web server to.
* Website Files: Access to the website content to be deployed.

**Step by Step instructions**

**Step 1: Prerequisites**

* Before starting the project, ensure the following are ready: Ansible Installed
* Install Ansible on a control node (your local machine or another Linux instance).

sudo yum install -y epel-release

sudo yum install -y ansible

**AWS EC2 Instances:**

* Create 2-3 AWS EC2 instances (use Amazon Linux or Ubuntu) to act as target servers.
* Ensure they are accessible from the control node via SSH.
* Control Node SSH Access:
* Copy your SSH key to the target servers for passwordless SSH access.
* Copy code
* ssh-copy-id ec2-user@<target\_server\_ip>

**Step 2: Project Directory Structure**

Create the required directory structure for the project:

ansible-webserver/

├── playbook.yml

├── inventory

├── roles/

│ ├── webserver/

│ ├── tasks/

│ │ └── main.yml

│ ├── files/

│ └── index.html

**Step 3: Inventory File**

* Define the target servers in the inventory file (inventory):

[webservers]

<target\_server\_ip\_1> ansible\_user=ec2-user

<target\_server\_ip\_2> ansible\_user=ec2-user

Replace <target\_server\_ip\_1> and <target\_server\_ip\_2> with your EC2 instance IPs.

**Step 4: Sample HTML Page**

Add a sample HTML file (roles/webserver/files/index.html) to deploy to the web server:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Welcome to Ansible Automated Server</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f9;

color: #333;

text-align: center;

}

header {

background-color: #007bff;

color: #fff;

padding: 20px;

}

h1 {

margin: 0;

font-size: 2.5rem;

}

p {

margin-top: 10px;

font-size: 1.2rem;

}

main {

padding: 20px;

}

footer {

background-color: #333;

color: #fff;

padding: 10px;

position: fixed;

width: 100%;

bottom: 0;

font-size: 0.9rem;

}

</style>

</head>

<body>

<header>

<h1>Welcome to Ansible Automated Server</h1>

<p>Your server was deployed seamlessly with automation!</p>

</header>

<main>

<h2>About This Page</h2>

<p>This is a simple website deployed using Ansible automation. Enjoy your newly configured web server!</p>

<h3>Features:</h3>

<ul style="list-style-type: square; text-align: left; display: inline-block;">

<li>Deployed using Ansible Playbooks</li>

<li>Simple and elegant design</li>

<li>Customizable content</li>

</ul>

</main>

<footer>

© 2024 - Automated by Ansible

</footer>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Welcome to Ansible Automated Server</title>

</head>

<body>

<h1>Web Server Deployed Using Ansible</h1>

</body>

</html>

**Step 5: Ansible Role Task**

Write the main role tasks in roles/webserver/tasks/main.yml:

yaml

- name: Install Apache

yum:

name: httpd

state: present

- name: Start and Enable Apache Service

service:

name: httpd

state: started

enabled: true

- name: Copy Web Page to Document Root

copy:

src: files/index.html

dest: /var/www/html/index.html

- name: Open HTTP Port in Firewall

firewalld:

port: 80/tcp

state: enabled

permanent: true

- name: Reload Firewall

command: firewall-cmd --reload

**Step 6: Playbook**

Write the main playbook file (playbook.yml) to call the role:

yaml

Copy code

- name: Setup Web Server

hosts: webservers

become: true

roles:

- webserver

**Step 7: Test Your Setup**

* Ping Target Servers:
* Run the ping module to test connectivity.

**ansible -i inventory webservers -m ping**

* Run the Playbook:
* Execute the playbook to set up the web servers.

**ansible-playbook -i inventory playbook.yml**

**Step 8: Verify the Deployment**

Open the browser and enter the IP address of any EC2 instance.

Example: http://<target\_server\_ip>.

You should see the HTML page with the message "Web Server Deployed Using Ansible"