Documentation: Using Ansible Playbooks to Provision Infrastructure on AWS and Azure

1. **Ansible Installation:**

Install Ansible on your local machine or on a control node. You can install it via pip:
bash
Copy code

```
pip install ansible
```

   -

2. **Cloud Account:**
   - **AWS: Make sure you have an AWS account and access to your AWS Access Key ID and Secret Access Key.**
   - **Azure: Make sure you have an Azure account and access to your Azure credentials.**
3. **Cloud Provider Modules:**
   - **Ansible uses different modules to interact with AWS and Azure. Make sure these modules are installed:**

For AWS: Install the boto3 and botocore libraries:
bash
Copy code

```
pip install boto3 botocore
```

For Azure: Install the azure libraries:
bash
Copy code

```
pip install azure-cli
```

4. **Authentication Configuration:**

For **AWS**, configure your AWS CLI with your credentials:
bash
Copy code
```
aws configure
```

For **Azure**, log in using the Azure CLI:
bash
Copy code
```
az login
```

1. Provisioning Infrastructure on AWS with Ansible Playbooks

Step 1: Create an Ansible Inventory File

An inventory file defines the hosts you want to manage with Ansible. For AWS, you can define your AWS instance information in this file.

Example `inventory.ini`:

ini
Copy code
```
[aws]
ec2_instance
ansible_host=ec2-18-234-56-78.compute-1.amazonaws.com
```

Step 2: Define the Playbook to Provision AWS Resources

In this step, we'll create a playbook to provision an EC2 instance on AWS. A playbook is a YAML file that defines the tasks Ansible should perform.

Example aws_provision.yml**:**

yaml
Copy code

```
---
- name: Provision EC2 instance on AWS
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Create EC2 instance
      ec2_instance:
        key_name: your-key-name
        id: "{{ ec2_instance_id }}"
        instance_type: t2.micro
        image: ami-0c55b159cbfafe1f0  # Replace with an
actual image ID
        region: us-east-1
        count: 1
        wait: yes
        group: default
        vpc_subnet_id: subnet-12345678  # Replace with
your subnet ID
        assign_public_ip: yes
        security_group: your-security-group
```

```
    instance_tags:
      Name: MyEC2Instance
  register: ec2
- name: Show instance details
  debug:
    var: ec2
```

Step 3: Run the Playbook

Execute the playbook to provision the EC2 instance:

bash
Copy code

```
ansible-playbook -i inventory.ini aws_provision.yml
```

Step 4: Verify the EC2 Instance

After the playbook runs successfully, verify that the EC2 instance is created in your AWS console under the EC2 dashboard.

2. Provisioning Infrastructure on Azure with Ansible Playbooks

Step 1: Create an Ansible Inventory File

For Azure, the inventory file will define the Azure VMs you want to manage.

Example `inventory.ini`:

ini

Copy code

```
[azure]
azure_vm ansible_host=your-vm-ip-address
```

Step 2: Define the Playbook to Provision Azure Resources

Now, let's create a playbook that provisions a virtual machine in Azure. This playbook will create a resource group, a virtual network, a subnet, and then deploy a virtual machine.

Example azure_provision.yml:

yaml
Copy code

```
---
- name: Provision resources on Azure
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Create a resource group
      azure_rm_resourcegroup:
        name: MyResourceGroup
        location: eastus
      register: resource_group

    - name: Create a virtual network
      azure_rm_virtualnetwork:
        resource_group: MyResourceGroup
        name: MyVNet
        address_prefixes: "10.0.0.0/16"
```

```
      register: virtual_network

    - name: Create a subnet
      azure_rm_subnet:
        resource_group: MyResourceGroup
        virtual_network_name: "{{ virtual_network.name
}}"
        name: MySubnet
        address_prefix: "10.0.1.0/24"
      register: subnet

    - name: Create a virtual machine
      azure_rm_virtualmachine:
        resource_group: MyResourceGroup
        name: MyAzureVM
        vm_size: Standard_B1ms
        admin_username: azureuser
        admin_password: "YourPassword123!"  # Change
this to a secure password
        image:
          offer: UbuntuServer
          publisher: Canonical
          SKU: 18.04-LTS
          version: latest
        network_interfaces:
          - name: MyNIC
            subnet_name: MySubnet
            public_ip: yes
            security_group: default
      register: vm
```

```
    - name: Show virtual machine details
      debug:
        var: vm
```

Step 3: Run the Playbook

Execute the playbook to provision the resources:

bash
Copy code

```
ansible-playbook -i inventory.ini azure_provision.yml
```

Step 4: Verify the Azure Resources

Once the playbook runs successfully, verify that the resources (VM, resource group, subnet, etc.) are created in your Azure portal.

3. Variables and Dynamic Provisioning

For both AWS and Azure, you can define variables in the playbook to make the process more flexible and reusable. For example:

1. **AWS:**

You can pass variables in a `vars.yml` file or directly in the playbook.

Example `vars.yml`:

yaml
Copy code

```yaml
ec2_instance_id: "ami-0c55b159cbfafe1f0"
region: "us-east-1"
instance_type: "t2.micro"
```

Then reference these variables in the playbook:

yaml
Copy code

```yaml
image: "{{ ec2_instance_id }}"
region: "{{ region }}"
instance_type: "{{ instance_type }}"
```

2. **Azure:**

Similarly, you can define variables for Azure resources:

Example `azure_vars.yml`:

yaml
Copy code

```yaml
resource_group: "MyResourceGroup"
location: "eastus"
vm_name: "MyAzureVM"
```

Reference the variables in the playbook:

yaml
Copy code

```
resource_group: "{{ resource_group }}"
location: "{{ location }}"
name: "{{ vm_name }}"
```

4. Using Dynamic Inventory

Ansible can automatically fetch information about the cloud infrastructure if you have a dynamic inventory (e.g., from **AWS EC2 instances** or **Azure VMs**).

1. **AWS Dynamic Inventory:**

Install the Boto plugin and use it to generate inventory based on AWS EC2 instances dynamically.

Example configuration for AWS:

ini
Copy code

```
[ec2]
plugin = aws_ec2
regions = us-east-1
keyed_groups = ['tags']
```

2. **Azure Dynamic Inventory:**

For Azure, Ansible provides the azure_rm dynamic inventory plugin. Set it up in the inventory.ini file:

ini
Copy code

```
[azure]
plugin = azure_rm
subscription_id = <your_subscription_id>
client_id = <your_client_id>
secret = <your_client_secret>
tenant = <your_tenant_id>
```

This setup allows Ansible to pull in live Azure resources automatically.

5. Cleanup and Deletion of Resources

To clean up resources, you can create a separate playbook for destruction, using the `state: absent` parameter in your playbook tasks.

Example for AWS EC2 Instance Cleanup:

yaml
Copy code

```
---
- name: Terminate EC2 instance
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Terminate the EC2 instance
      ec2_instance:
        instance_ids:
          - "{{ ec2_instance_id }}"
        state: absent
```

Example for Azure VM Cleanup:

yaml

Copy code

```yaml
---
- name: Delete Azure VM
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Delete virtual machine
      azure_rm_virtualmachine:
        resource_group: MyResourceGroup
        name: MyAzureVM
        state: absent
```