**Lab Guide: Setting Up and Configuring 3 Ubuntu Instances in AWS**

This lab guide will walk you through the process of deploying and configuring three Ubuntu instances in the Ohio region with the specified settings.

**1. Create 3 Ubuntu Instances in the Ohio Region**

1. **Login to AWS Management Console:**
   * Navigate to the [AWS Management Console](https://aws.amazon.com/console/).
   * Switch to the Ohio (us-east-2) region from the top-right corner.
2. **Launch Instances:**
   * Go to **EC2** service.
   * Click on **Launch Instance**.
   * Provide a name for your instances, such as raman-lumen-1, raman-lumen-2, and raman-lumen-3.
3. **AMI Selection:**
   * Search for **Ubuntu Server 20.04 LTS** (or a newer version available as of 24th).
   * Select the AMI.
4. **Instance Type:**
   * Choose **t2.micro** for all instances.
5. **Key Pair Creation:**
   * In the **Key Pair** section, click **Create a new key pair**.
   * Provide a name, e.g., raman-key.
   * Select **RSA** and **.pem** format.
   * Download the .pem file and keep it secure for SSH access.
6. **Network Settings (Security Group):**
   * In the **Network Settings** section:
     + Create a new security group.
     + Add a rule for **SSH access**:
       - **Type:** SSH
       - **Protocol:** TCP
       - **Port Range:** 22
       - **Source:** My IP (or customize as needed).
     + Save the security group as raman-ssh-sg.
7. **Storage:**
   * Keep the default 8 GiB (adjust if necessary).
8. **Launch:**
   * Click **Launch Instance**.
   * Repeat the process for all three instances.

**2. Connect to the Instances**

1. **Change Permissions of the Key Pair:**
   * Open your terminal or command prompt.
   * Navigate to the directory where your key pair file (raman-key.pem) is saved.
   * Run:

bash

Copy code

chmod 400 raman-key.pem

1. **SSH into the Instances:**
   * Retrieve the **public IP** of each instance from the EC2 console.
   * Connect to the instances one by one using the following commands:

bash

Copy code

ssh -i raman-key.pem ubuntu@<instance-public-ip>

**3. Change Hostnames for Each Instance**

1. **Login as Root:**
   * Once connected to each instance:

bash

Copy code

sudo -i

1. **Set the Hostname:**
   * Use the hostnamectl command to set unique hostnames for each instance:
     + **Instance 1:**

bash

Copy code

hostnamectl set-hostname raman-lumen-1

* + - **Instance 2:**

bash

Copy code

hostnamectl set-hostname raman-lumen-2

* + - **Instance 3:**

bash

Copy code

hostnamectl set-hostname raman-lumen-3

1. **Reload the Shell:**
   * Refresh the shell for changes to reflect:

bash

Copy code

bash

* + Verify the hostname:

bash

Copy code

hostname

**4. Verify Connectivity Between Instances**

1. **Enable SSH Access Between Instances:**
   * Ensure all instances are part of the same security group (raman-ssh-sg) to allow internal communication.
2. **Test Connectivity:**
   * Login to one instance and test SSH access to the others:

bash

Copy code

ssh ubuntu@<other-instance-private-ip>

* + Replace <other-instance-private-ip> with the private IPs of the other instances (available in the EC2 console).

**Summary of Steps**

* Deployed 3 Ubuntu instances in the Ohio region using the **t2.micro** instance type.
* Created and used a key pair for SSH access.
* Configured a security group for SSH access from specific sources.
* Connected to all three instances via SSH.
* Changed hostnames to raman-lumen-1, raman-lumen-2, and raman-lumen-3.
* Verified internal connectivity between instances.

**Lab Guide: Setting Up Ansible with Configuration and SSH Access**

This guide details the setup of Ansible on a master node (controller) and configuring SSH access to managed nodes (m1 and m2) for Ansible to communicate and execute commands.

**1. Setup Ansible on the Master Node**

1. **Update the System:**
   * Connect to the master node (controller):

bash

Copy code

sudo apt update

1. **Install Prerequisites:**
   * Install essential software properties:

bash

Copy code

sudo apt install software-properties-common

1. **Add Ansible PPA:**
   * Add the official Ansible repository:

bash

Copy code

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

1. **Install Ansible:**
   * Install Ansible on the master node:

bash

Copy code

sudo apt install ansible

1. **Verify Installation:**
   * Check the installed version of Ansible:

bash

Copy code

ansible --version

**2. Modify Ansible Configuration**

1. **Edit Ansible Configuration File:**
   * Open the Ansible configuration file:

bash

Copy code

sudo vi /etc/ansible/ansible.cfg

* + Replace the contents of the file with the configuration from the Gist:  
    [Ansible Configuration](https://gist.github.com/wbcurry/f38bc6d8d1ee4a70ee2c).

1. **Change Default Inventory Location:**
   * Update the configuration to use /etc/ansible/hosts as the default inventory file:
     + Locate the inventory setting.
     + Set it to:

javascript

Copy code

inventory = /etc/ansible/hosts

* + Save and exit (:wq).

1. **Verify Configuration:**
   * Ensure changes are saved:

bash

Copy code

cat /etc/ansible/ansible.cfg

**3. Prepare Managed Nodes (m1 and m2)**

1. **Update SSH Configuration on Managed Nodes:**
   * SSH into m1:

bash

Copy code

ssh ubuntu@<m1-public-ip>

* + Open the SSH daemon configuration file:

bash

Copy code

sudo vi /etc/ssh/sshd\_config

* + Update the following lines:

plaintext

Copy code

PermitRootLogin yes

PasswordAuthentication yes

* + Save and exit (:wq).

1. **Restart SSH Service:**
   * Apply the changes by restarting the SSH service:

bash

Copy code

sudo systemctl restart ssh

1. **Set the Root Password:**
   * Set a password for the root user:

bash

Copy code

sudo passwd root

* + Enter and confirm a strong password.

1. **Repeat Steps for m2:**
   * Follow the same process on the second managed node (m2).

**4. Configure SSH Access from Master to Managed Nodes**

1. **Test SSH Access:**
   * From the master node, connect to m1 and m2 using the root user:

bash

Copy code

ssh root@<m1-private-ip>

ssh root@<m2-private-ip>

* + Verify connectivity and authentication.

1. **Add Managed Nodes to Inventory:**
   * Open the inventory file on the master node:

bash

Copy code

sudo vi /etc/ansible/hosts

* + Add the managed nodes:

plaintext

Copy code

[all]

m1 ansible\_host=<m1-private-ip> ansible\_user=root

m2 ansible\_host=<m2-private-ip> ansible\_user=root

1. **Save and Exit Inventory File:**
   * Save and close the file.

**5. Test Ansible Connectivity**

1. **Ping Managed Nodes:**
   * Run the Ansible ping module to check connectivity:

bash

Copy code

ansible all -m ping

* + If prompted for a password, use:

bash

Copy code

ansible all -m ping -k

1. **Verify Successful Response:**
   * A successful response will look like this:

plaintext

Copy code

m1 | SUCCESS => {

"changed": false,

"ping": "pong"

}

m2 | SUCCESS => {

"changed": false,

"ping": "pong"

}

**Summary of Steps**

* Installed Ansible on the master node and verified the installation.
* Modified the Ansible configuration file to use a custom inventory location.
* Updated SSH configurations on managed nodes (m1 and m2) to allow root login and password authentication.
* Configured root passwords and restarted SSH services.
* Verified SSH access from the master to managed nodes.
* Added managed nodes to the Ansible inventory and tested connectivity using the Ansible ping module.

Your Ansible setup is now ready for executing tasks and managing configurations on the nodes.

**Lab Guide: Configuring a Custom Ansible Inventory File**

This guide details the steps to configure Ansible to use a custom inventory file located at /home/ubuntu/ramanhost instead of the default inventory file (/etc/ansible/hosts).

**1. Prepare the Environment**

1. **Navigate to Home Directory:**
   * On the master node (controller), switch to the home directory:

bash

Copy code

cd /home/ubuntu/

1. **Verify Directory Structure:**
   * List files in the directory to ensure no conflicts:

bash

Copy code

ls

**2. Modify Ansible Configuration File**

1. **Open the Ansible Configuration File:**
   * Edit the ansible.cfg file:

bash

Copy code

vi ansible.cfg

1. **Change Inventory Location:**
   * Locate the inventory directive in the [defaults] section.
   * Update it to point to the custom inventory file /home/ubuntu/ramanhost:

plaintext

Copy code

[defaults]

inventory = /home/ubuntu/ramanhost

1. **Save Changes:**
   * Save and exit the editor:
     + Press Esc, type :wq, and hit Enter.
2. **Verify Changes:**
   * Confirm that the ansible.cfg file is updated:

bash

Copy code

cat ansible.cfg

**3. Create the Custom Inventory File**

1. **Create and Open ramanhost:**
   * Create a file named ramanhost in the /home/ubuntu directory:

bash

Copy code

vi ramanhost

1. **Add Managed Nodes:**
   * Add the details of your managed nodes (m1 and m2) in the following format:

plaintext

Copy code

[all]

m1 ansible\_host=<m1-private-ip> ansible\_user=root

m2 ansible\_host=<m2-private-ip> ansible\_user=root

1. **Save the Inventory File:**
   * Save and close the file (:wq).
2. **Verify Inventory File:**
   * Confirm the contents of the inventory file:

bash

Copy code

cat ramanhost

**4. Test Ansible Connectivity**

1. **Ping All Nodes:**
   * Run the ping module to test connectivity:

bash

Copy code

ansible all -m ping -k

* + The -k flag prompts for a password if SSH keys are not used.

1. **Validate Successful Connectivity:**
   * A successful response for both nodes will look like:

plaintext

Copy code

m1 | SUCCESS => {

"changed": false,

"ping": "pong"

}

m2 | SUCCESS => {

"changed": false,

"ping": "pong"

}

**5. Verify Directory and File Paths**

1. **Check Current Directory:**
   * Ensure you are in the correct working directory:

bash

Copy code

pwd

* + Output:

plaintext

Copy code

/home/ubuntu

1. **List Files in the Directory:**
   * Confirm the presence of ansible.cfg and ramanhost:

bash

Copy code

ls

**Summary of Steps**

1. Changed the default inventory file location in ansible.cfg to /home/ubuntu/ramanhost.
2. Created a custom inventory file (ramanhost) with the managed nodes.
3. Tested connectivity to managed nodes using the ping module.
4. Verified directory structure and file paths.

With these steps completed, Ansible is now configured to use the custom inventory file for managing nodes. This setup provides flexibility for managing environments with specific inventory configurations.

**Lab Guide: Setting Up Passwordless SSH Authentication**

This guide details the steps for setting up passwordless SSH authentication between the Ansible master node (controller) and managed nodes (m1 and m2). It enables seamless communication for Ansible without requiring a password for each operation.

**1. Generate SSH Key Pair on the Master Node**

1. **Login to the Master Node:**
   * Connect to the master node (controller).
2. **Generate an SSH Key Pair:**
   * Run the following command to generate a new RSA key pair:

bash

Copy code

ssh-keygen -t rsa

* + Follow the prompts:
    - Press Enter to accept the default location (/home/ubuntu/.ssh/id\_rsa).
    - Leave the passphrase blank for passwordless authentication.
    - Output example:

plaintext

Copy code

Generating public/private rsa key pair.

Enter file in which to save the key (/home/ubuntu/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/ubuntu/.ssh/id\_rsa.

Your public key has been saved in /home/ubuntu/.ssh/id\_rsa.pub.

1. **Verify Key Pair:**
   * List the generated key files:

bash

Copy code

ls ~/.ssh

* + Output example:

plaintext

Copy code

id\_rsa id\_rsa.pub

**2. Transport the Public Key to Managed Nodes**

1. **Copy the Public Key to m1:**
   * Use the ssh-copy-id command to copy the public key to m1:

bash

Copy code

ssh-copy-id root@<m1-private-ip>

* + Enter the root password for m1 when prompted.

1. **Copy the Public Key to m2:**
   * Repeat the same process for m2:

bash

Copy code

ssh-copy-id root@<m2-private-ip>

* + Enter the root password for m2 when prompted.

1. **Verify Key Transfer:**
   * On m1 and m2, the public key will be added to the ~/.ssh/authorized\_keys file.
   * Connect to each node and check the file:

bash

Copy code

ssh root@<node-private-ip>

cat ~/.ssh/authorized\_keys

* + Ensure the master node's public key is present in the file.

**3. Test Passwordless SSH Authentication**

1. **Connect to m1 and m2 Without Passwords:**
   * From the master node, test SSH access to m1:

bash

Copy code

ssh root@<m1-private-ip>

* + Exit the session:

bash

Copy code

exit

* + Repeat for m2:

bash

Copy code

ssh root@<m2-private-ip>

1. **Verify Passwordless Access:**
   * Ensure you are not prompted for a password when connecting to m1 and m2.

**4. Test Ansible Connectivity**

1. **Ping All Managed Nodes:**
   * Use the Ansible ping module to test connectivity:

bash

Copy code

ansible all -m ping

1. **Expected Output:**
   * A successful response will look like this:

plaintext

Copy code

m1 | SUCCESS => {

"changed": false,

"ping": "pong"

}

m2 | SUCCESS => {

"changed": false,

"ping": "pong"

}

**5. Verify History and Steps**

1. **Confirm Commands Executed:**
   * View the command history:

bash

Copy code

history

* + Verify that all steps were executed correctly.

**Summary of Steps**

1. Generated an SSH key pair on the master node.
2. Copied the public key to m1 and m2 using ssh-copy-id.
3. Verified passwordless SSH access to the managed nodes.
4. Tested Ansible connectivity using the ping module.

With passwordless SSH authentication set up, the master node can now seamlessly manage the nodes using Ansible.

4o