Ansible Master-Slave Architecture Project

Objective: The objective of this project is to set up an Ansible master-slave architecture on two EC2 instances, one as the master and the other as the slave, to demonstrate secure and automated management of remote servers using Ansible.

Architecture Overview:

The project involves setting up two EC2 instances:

1. **Master Instance:** This instance will act as the Ansible control node, responsible for managing and configuring the slave instance.
2. **Slave Instance:** This instance will be managed and configured by the master instance using Ansible.

Step-by-Step Implementation:

Step 1: Launch EC2 Instances

- Launch two EC2 instances with the same Amazon Machine Image (AMI) and configure the necessary security groups.
- Ensure that both instances have a public IP address and can communicate with each other.

Step 2: Generate RSA Keys

- On both instances, generate a pair of RSA keys using the **ssh-keygen** command.
- The generated keys will be used for secure SSH communication between the master and slave instances.

Step 3: Configure SSH on Master Instance

- On the master instance, create a new SSH configuration file (**~/.ssh/config**) and add the slave instance's IP address and username.
- Copy the master instance's public RSA key (**~/.ssh/id_rsa.pub**) to the slave instance's **~/.ssh/authorized_keys** file.

Step 4: Configure SSH on Slave Instance

- On the slave instance, create a new SSH configuration file (**~/.ssh/config**) and add the master instance's IP address and username.
- Ensure that the slave instance's **~/.ssh/authorized_keys** file contains the master instance's public RSA key.

Step 5: Install Ansible on Master Instance

- Install Ansible on the master instance using the package manager (e.g., **yum** or **apt**).
- Configure the Ansible inventory file (**/etc/ansible/hosts**) to include the slave instance's IP address and username.

Step 6: Test Ansible Connection

- Run the **ansible -m ping** command on the master instance to test the connection to the slave instance.
- Verify that the master instance can successfully connect to the slave instance using Ansible.

Step 7: Create Ansible Playbook

- Create an Ansible playbook (e.g., **slave_config.yml**) on the master instance to configure the slave instance.
- The playbook should include tasks to manage and configure the slave instance, such as:
    - Installing software packages
    - Configuring system settings
    - Deploying applications

Step 8: Run Ansible Playbook

- Run the Ansible playbook on the master instance using the **ansible-playbook** command.
- Verify that the playbook successfully configures the slave instance according to the defined tasks.

Benefits:

- This project demonstrates the use of Ansible for automating and managing remote servers in a secure and efficient manner.
- The master-slave architecture enables centralized management of multiple servers, making it easier to maintain and update infrastructure.
- The use of SSH keys ensures secure communication between the master and slave instances, reducing the risk of unauthorized access.

Conclusion: The Ansible Master-Slave Architecture Project provides a hands-on experience in setting up and configuring an Ansible-based infrastructure on EC2 instances. This project showcases the power of Ansible in automating and managing remote servers, while ensuring secure communication and access control.