Ansible Master-Slave Setup Guide on AWS

This guide walks you through setting up Ansible on AWS EC2 instances with a Master-Slave SSH configuration. Follow each step carefully. Commands assume Ubuntu on both instances and that you have SSH access.

Prerequisites

• 2 AWS EC2 instances running Ubuntu (one Master, one Slave).
• Hostnames changed to Master and Slave (use sudo hostname <name> to set).
• You can SSH into both instances and have sudo/root access.
• Connectivity between Master and Slave (security group allowing SSH from Master to Slave).

High-level Steps

1. Create and SSH into two EC2 instances (Master and Slave).
2. Install Ansible on the Master.
3. Generate SSH keys on Master and copy public key to Slave authorized_keys.
4. Configure SSH on Slave to allow root login via key and enable PubkeyAuthentication.
5. Test Ansible connectivity and run example modules to install/remove packages.

Master Node Setup

Switch to root and update packages:

```
sudo su

sudo apt update -y
```

Add Ansible PPA and install Ansible:

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

apt-get install ansible -y
```

Verify installation:

```
ansible --version
```

Edit the Ansible inventory to add the Slave IP:

```
nano /etc/ansible/hosts
```

Add at the end of the file:

```
[client_1]
<IP_Address_of_Slave>
```

Create SSH keypair on Master (if not already created):

```
ssh-keygen -t rsa
```

Switch to root and list .ssh:

```
cd /root
cd .ssh/
ls
```

Slave Node Configuration

Switch to root on Slave and open SSH folder:

```
sudo su

cd /root/.ssh/
ls
```

Open authorized_keys and paste the Master public key (id_rsa.pub):

```
nano authorized_keys
```

• Paste the full content of id_rsa.pub from Master into this file and save.

Edit SSH daemon configuration to allow root login and pubkey auth:

```
nano /etc/ssh/sshd_config
```

Ensure the following lines are set:

```
PermitRootLogin yes
```

After editing, restart SSH:

```
systemctl restart sshd || systemctl restart ssh
```

Back to Master — Test Ansible connectivity

From Master, test ping via Ansible:

ansible -m ping all

If host key prompt appears when using ssh, accept it once: (optional if error found)

ssh root@<IP_Address_of_Slave>
When prompted 'Are you sure you want to continue connecting (yes/no)?' type 'yes' and press Enter.

ansible -m ping all

ansible client_1 -m setup

Example: Install/Remove packages via Ansible

Verify package presence on Slave (run on slave):

git --version
nano --version

From Master, install git and remove nano on Slave using apt module:

ansible client_1 -m apt -a "name=git state=present" --become

ansible client_1 -m apt -a "name=nano state=absent" --become

Verify changes on Slave:

git --version # Should show version if installed
nano test.txt # Should fail if nano was removed

Troubleshooting & Tips

• Ensure security groups allow SSH (port 22) from the Master to the Slave.
• Make sure the correct IP is in /etc/ansible/hosts or use an FQDN.
• Permission issues: authorized_keys must be owned by root and have mode 600. .ssh folder should be 700.
• If PermitRootLogin=no for security reasons, you can use a non-root user with sudo and configure Ansible to use ansible_user and become method.
• To avoid host key prompt in automation, add the Slave to known_hosts using ssh-keyscan or use ansible_ssh_common_args to disable strict host key checking (use with caution).

Security note: Enabling PermitRootLogin yes is convenient for quick setups but reduces security. Prefer key-based login for a specific sudo-capable user and disable root login in production.

Optional: Add Slave to known_hosts (non-interactive)

On Master, run:

```
ssh-keyscan -H <IP_Address_of_Slave> >> /root/.ssh/known_hosts
```

Generated by: Ansible Master-Slave Setup Guide Generator
Date: 2025-10-29
