Contents

[Step 1: Update the System Package List 1](#_Toc177798508)

[Step 2: Verify Python Installation 1](#_Toc177798509)

[Step 3: Install Software Dependencies 2](#_Toc177798510)

[Step 4: Add the Ansible PPA 2](#_Toc177798511)

[Step 5: Install Ansible 2](#_Toc177798512)

[Step 6: Verify Ansible Installation 2](#_Toc177798513)

[Step 7: Configure SSH for Passwordless Authentication 2](#_Toc177798514)

[Step 8: Add the SSH Key to the Remote Host (Slave) 2](#_Toc177798515)

[Step 9: Create an AMI from the Instance 3](#_Toc177798516)

[Step 10: Define Remote Hosts in the Ansible Inventory File 4](#_Toc177798517)

[Step 11: Test the Ansible Connection to the Hosts 5](#_Toc177798518)

# How to Install Ansible and Configure Hosts for Automation

Ansible is a powerful open-source automation tool that simplifies IT management tasks such as configuration management, application deployment, and orchestration. It works by communicating with remote hosts (nodes) over SSH and executing tasks using playbooks or ad-hoc commands.

In this article, we will cover the steps to install Ansible on a master node, generate SSH keys, connect remote hosts (slaves) for passwordless authentication, and configure the Ansible inventory file for managing multiple hosts.

**Prerequisites:**

1. A master node (control machine) running Ubuntu or a similar Debian-based Linux system.
2. One or more remote hosts (slaves) that you want to manage using Ansible.
3. SSH access between the master and slave nodes.

## Step 1: Update the System Package List

Before installing any software, update your system's package list to ensure you're installing the latest versions of software:

**sudo apt update -y**

## Step 2: Verify Python Installation

Ansible is built in Python, so Python must be installed on your master node. Check if Python is installed by running:

**python3 --version**

If Python is not installed, install it with:

**sudo apt install python3 -y**

## Step 3: Install Software Dependencies

Ansible requires certain dependencies to manage PPAs (Personal Package Archives) and install the latest Ansible version. Install these using:

**sudo apt install software-properties-common -y**

## Step 4: Add the Ansible PPA

Ansible is available in a dedicated PPA that contains the latest release. Add this PPA to your repository list:

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

## Step 5: Install Ansible

Once the PPA is added, install Ansible:

**sudo apt install ansible -y**

## Step 6: Verify Ansible Installation

Check if Ansible is installed correctly by verifying the version:

**ansible --version**

You should see the installed Ansible version and Python information. At this point, Ansible is installed and ready to manage hosts.

## Step 7: Configure SSH for Passwordless Authentication

Ansible uses SSH to communicate with remote hosts. To automate tasks without being prompted for a password every time, configure passwordless SSH access.

**Generate an SSH Key Pair on the Master Node**

On your master node, run the following command to generate an SSH key pair:

**ssh-keygen**

Press **Enter** to accept the default file location and leave the passphrase empty (optional).

## Step 8: Add the SSH Key to the Remote Host (Slave)

After generating the SSH key pair, you need to copy the public key to the remote host for passwordless SSH authentication.

First, display the content of your public key:

**cat ~/.ssh/id\_rsa.pub**

Copy the entire output. Then, log into your remote host (slave):

**ssh user@slave\_ip\_address**

On the remote host, append the copied public key to the authorized\_keys file:

**echo "ssh-rsa AAAAB3Nza..." >> ~/.ssh/authorized\_keys**

Ensure the permissions are set correctly:

**chmod 600 ~/.ssh/authorized\_keys**

Exit the remote host:

**exit**

Now, test the SSH connection from the master node to ensure passwordless access is working:

**ssh user@slave\_ip\_address**

If configured correctly, you should log in without being prompted for a password.

## Step 9: Create an AMI from the Instance

After setting up SSH keys and ensuring passwordless access, you can create an AMI of the instance. This step assumes you are using AWS CLI.

**Install AWS CLI** (if not already installed):

**sudo apt install awscli -y**

**Configure AWS CLI** with your credentials:

**aws configure**

**Create an AMI** from the instance:

**aws ec2 create-image --instance-id i-1234567890abcdef0 --name "MyServerImage" --no-reboot**

Replace i-1234567890abcdef0 with your actual instance ID and "MyServerImage" with your desired AMI name.

This will create an AMI of your instance, which you can use to launch new instances with the same configuration.

Or use management console to create ami.

1. **Select the Instance**:
   * select the instance you want to create an AMI from.
2. **Create Image (AMI)**:
   * With the instance selected, click on the **Actions** button at the top.
   * From the dropdown menu, select **Image and templates** > **Create image**.
3. **Configure the Image**:
   * In the **Create Image** dialog box, provide a name and description for your AMI.
4. **Create the Image**:
   * Click the **Create image** button at the bottom of the dialog box.
   * You will see a confirmation message that the image creation process has started.
5. **Monitor the Image Creation**:
   * To monitor the progress, go to the **AMIs** section in the left-hand menu under **Images**.
   * Your new AMI will appear in the list. It may take a few minutes to complete.

Once the AMI is created, you can use it to launch new instances with the same configuration.

## Step 10: Define Remote Hosts in the Ansible Inventory File

Ansible uses an inventory file to define the hosts it will manage. Hosts can be grouped and configured with specific parameters.

**Open the Ansible Inventory File**

Open the /etc/ansible/hosts file to define your remote hosts:

**sudo nano /etc/ansible/hosts**

**Add Slave Hosts**

In the file, add your slave hosts under a group (e.g., [host]) and specify their IP addresses using ansible\_ssh\_host. Here's how you can structure the file:

**[host]**

**Slave1 ansible\_ssh\_host=<Host ip address>**

**Slave2 ansible\_ssh\_host=<Host ip address>**

This configuration tells Ansible that Slave1 and Slave2 are reachable at the specified IP addresses.

**Additional Host Parameters:**

You can also include additional SSH connection details, such as specifying the SSH user, private key, or port:

**[host]**

**Slave1 ansible\_ssh\_host=<Host ip address> ansible\_user=ubuntu ansible\_ssh\_private\_key\_file=/path/to/private/key**

**Slave2 ansible\_ssh\_host=<Host ip address> ansible\_user=ubuntu ansible\_port=2222**

This example includes the SSH user, private key location, and a custom SSH port for Slave2.

**Save the Inventory File**

Once the hosts are added, save and close the file:

* Press Ctrl + O to save the changes.
* Press Ctrl + X to exit the editor.

## Step 11: Test the Ansible Connection to the Hosts

To ensure that Ansible can communicate with the remote hosts, run a simple ping test:

**ansible -m ping host**

This command sends a ping to all hosts in the [host] group. If the connection is successful, you will see a pong response from each slave host.

**Conclusion**

By following these steps, you’ve successfully installed Ansible on your master node, configured passwordless SSH access, and set up an inventory file to manage remote hosts. Now, you're ready to automate tasks across multiple systems using Ansible's playbooks and commands. You can add more hosts or groups to your inventory as your infrastructure grows, making Ansible a scalable solution for automation.