

# Ansible

All documentation related to Ansible.

- [Ansible Installation](#)
  - [\[How-To\] Install Ansible on Ubuntu 24.04 LTS](#)
- [Ansible Configuration](#)

# Ansible Installation

All Ansible installation docs

# [How-To] Install Ansible on Ubuntu 24.04 LTS

## Purpose

This doc will explain how to take a base Ubuntu 24 server and turn it into a hub for Ansible automation that can interact with Proxmox.

## Prerequisites

List of prerequisites:

- Sudo user
- Ubuntu 24.04 LTS Server (VM)

## Instructions

### Step 1: Install Ansible on Ubuntu

Update the package list:

```
sudo apt update
```

Install Ansible:

```
sudo apt install -y ansible
```

Verify the Ansible installation:

```
ansible --version
```

### Step 2: Install Required Ansible Collections and Modules for Proxmox

Install the community.general collection, which includes the proxmox modules:

```
ansible-galaxy collection install community.general
```

Add the Debian repository to your sources list:

```
echo "deb http://deb.debian.org/debian stable main" | sudo tee /etc/apt/sources.list.d/debian.list
```

Install the proxmoxer Python library, which the Proxmox API module depends on:

```
sudo apt update
sudo apt install -y python3-proxmoxer
```

Disable the Debian repository after installing to avoid pulling other packages from it:

```
sudo rm /etc/apt/sources.list.d/debian.list
sudo apt update
```

Verify proxmoxer is installed on the system:

```
python3 -c "import proxmoxer; print(proxmoxer.__version__)"
```

## Step 3: Set Up API Access for Proxmox

**Create a Proxmox API user** specifically for Ansible access:

1. Log into Proxmox and navigate to **Datacenter > Permissions > Users**.
2. Add a new user (e.g., `ansible_user`) with **API Token** privileges.

**Create an API token** for the user:

1. Go to **API Tokens** under the new user and create a token (e.g., `ansible_token`).
2. Note the `username@realm!tokenid` and the token's secret, as you'll need them in Ansible.

**Assign the necessary permissions** to the API user:

1. Go to **Datacenter > Permissions > Add > User Permission**.
2. Assign `VM.Allocate`, `VM.Config.Disk`, `VM.Config.HWType`, and `VM.Console` permissions (or others as needed).

## Step 4: Set Up Passwordless SSH Authentication Between Ansible and Proxmox

To set up passwordless SSH authentication between your Ansible server and each Proxmox VE node, here's a step-by-step guide to generate an SSH key pair on the Ansible server and copy the public key to each Proxmox VE node. This will enable Ansible to access Proxmox nodes without needing a password each time.

1. **Generate SSH Key Pair on the Ansible Server:** Open a terminal on your Ansible server and run the following command to create an SSH key pair:

bash

```
ssh-keygen -t rsa -b 4096
```

- When prompted, you can press `Enter` to accept the default location (`~/.ssh/id_rsa`) and leave the passphrase empty for automated access.
- This command generates two files in your `~/.ssh` directory:
  - `id_rsa` (the private key)
  - `id_rsa.pub` (the public key)

2. **Copy the Public Key to Each Proxmox Node:** Use the `ssh-copy-id` command to install the public key on each Proxmox node, which enables passwordless authentication.

Replace `<proxmox-node-ip>` with the IP address or hostname of each Proxmox server.

bash

```
ssh-copy-id -i ~/.ssh/id_rsa.pub root@<proxmox-node-ip>
```

- You'll need to enter the password for `root` on the Proxmox node just this once.
- Repeat this step for each Proxmox node you want Ansible to manage.

3. **Test Passwordless SSH Authentication:** After copying the public key, test the SSH connection to each Proxmox node to confirm it works without a password:

bash

```
ssh root@<proxmox-node-ip>
```

If the setup is successful, you should be logged in directly without any password prompt.

4. **Confirm Ansible Can Connect to Proxmox Nodes:** Now, you can test Ansible's connection to each Proxmox node by running:

bash

```
ansible all -m ping -i <inventory_file>
```

This should return a successful `ping` response if the connection works as expected.

## Step 5: Configure Ansible Inventory for Proxmox

Create a directory for your Ansible project and configure the inventory:

```
cd /etc/ansible
sudo nano inventory.yml
```

Add the Proxmox server to the inventory file:

```
---
proxmox:
  hosts:
    192.168.1.10:
  vars:
    ansible_user: root
    ansible_password: *****
    ansible_ssh_common_args: '-o StrictHostKeyChecking=no'
    ansible_python_interpreter: auto_silent
```

Replace IP address under hosts with your proxmox ve host IP and ansible\_password with your proxmox ve root password. For better security use API token and ansible vault.

## Step 6: Create Ansible Playbook for Ping Test to Proxmox

Create a playbook file:

```
sudo nano test-ping.yml
```

Add the following content to the playbook to test ping to proxmox ve node:

```
---
- name: Test connectivity to all hosts
  hosts: all
  gather_facts: no
  tasks:
    - name: Ping test
      ansible.builtin.ping:
```

Run the playbook with the following command:

```
ansible-playbook test-ping.yml
```

If everything works, you should see output like this:

```
user@ansible-server:/etc/ansible$ ansible-playbook test-ping.yml

PLAY [Test connectivity to all hosts]
*****

TASK [Ping test]
*****

ok: [192.168.1.10]

PLAY RECAP
*****
192.168.1.10      : ok=1   changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

This confirms you can ping the proxmox ve host via Ansible playbooks from your Ansible server.

## Step 7: Create Ansible Playbook for Connection Test to Proxmox

Create a playbook file:

```
sudo nano test-connection.yml
```

Add the following content to the playbook to test the connection to proxmox ve node:

```
---
- name: Proxmox example task
  hosts: proxmox
  tasks:
    - name: Example Proxmox task
      ansible.builtin.debug:
```

Run the playbook with the following command:

```
ansible-playbook test-connection.yml
```

If everything works, you should see output like this:

```
user@ansible-server:/etc/ansible$ ansible-playbook test-connection.yml
```

PLAY [Proxmox example task]

\*\*\*\*\*

TASK [Gathering Facts]

\*\*\*\*\*

ok: [192.168.1.10]

TASK [Example Proxmox task]

\*\*\*\*\*

ok: [192.168.1.10] => {

"msg": "Hello world!"

}

PLAY RECAP

\*\*\*\*\*

192.168.1.10 : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

This confirms you can connect and authenticate to the proxmox ve host via Ansible playbooks from your Ansible server.

At this point, you have a fully working Ansible server built and configured with proxmox ve nodes for automation. Start creating playbooks for your automation.



# Ansible Configuration

All Ansible configuration docs.