**How To Install and Configure Ansible on Ubuntu 20.04**

## [Prerequisites](https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-20-04#prerequisites)-

1. **One Ansible Control Node**: The Ansible control node is the machine we’ll use to connect to and control the Ansible hosts over SSH.
2. A non-root user with sudo privileges.
3. An SSH keypair associated with this user.
4. **One or more Ansible Hosts**:

## [Step 1 — Installing Ansible](https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-20-04#step-1-installing-ansible)-

1. sudo apt-add-repository ppa:ansible/ansible

Press ENTER when prompted to accept the PPA addition.

Next, refresh your system’s package index so that it is aware of the packages available in the newly included PPA:

1. sudo apt update

Following this update, you can install the Ansible software with:

1. sudo apt install ansible

Your Ansible control node now has all of the software required to administer your hosts

## [Step 2 — Setting Up the Inventory File](https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-20-04#step-2-setting-up-the-inventory-file)

The inventory file contains information about the hosts you’ll manage with Ansible. You can include anywhere from one to several hundred servers in your inventory file, and hosts can be organized into groups and subgroups. The inventory file is also often used to set variables that will be valid only for specific hosts or groups, in order to be used within playbooks and templates. Some variables can also affect the way a playbook is run, like the ansible\_python\_interpreter variable that we’ll see in a moment.

To edit the contents of your default Ansible inventory, open the /etc/ansible/hosts file using your text editor of choice, on your Ansible control node:

sudo nano /etc/ansible/hosts

**Note: Although Ansible typically creates a default inventory file at etc/ansible/hosts, you are free to create inventory files in any location that better suits your needs. In this case, you’ll need to provide the path to your custom inventory file with the -i parameter when running Ansible commands and playbooks. Using per-project inventory files is a good practice to minimize the risk of running a playbook on the wrong group of servers.**

The default inventory file provided by the Ansible installation contains a number of examples that you can use as references for setting up your inventory. The following example defines a group named [servers] with three different servers in it, each identified by a custom alias: **server1**, **server2**, and **server3**. Be sure to replace the highlighted IPs with the IP addresses of your Ansible hosts.

/etc/ansible/hosts

[servers]

server1 ansible\_host=203.0.113.111

server2 ansible\_host=203.0.113.112

server3 ansible\_host=203.0.113.113

[all:vars]

ansible\_python\_interpreter=/usr/bin/python3

The all:vars subgroup sets the ansible\_python\_interpreter host parameter that will be valid for all hosts included in this inventory. This parameter makes sure the remote server uses the /usr/bin/python3 Python 3 executable instead of /usr/bin/python (Python 2.7), which is not present on recent Ubuntu versions.

When you’re finished, save and close the file by pressing CTRL+X then Y and ENTER to confirm your changes.

Whenever you want to check your inventory, you can run:

1. ansible-inventory --list -y

You’ll see output similar to this, but containing your own server infrastructure as defined in your inventory file:

Output

all:

children:

servers:

hosts:

server1:

ansible\_host: 203.0.113.111

ansible\_python\_interpreter: /usr/bin/python3

server2:

ansible\_host: 203.0.113.112

ansible\_python\_interpreter: /usr/bin/python3

server3:

ansible\_host: 203.0.113.113

ansible\_python\_interpreter: /usr/bin/python3

ungrouped: {}

Now that you’ve configured your inventory file, you have everything you need to test the connection to your Ansible hosts.

## [**Step 3 — Testing Connection**](https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-20-04#step-3-testing-connection)

After setting up the inventory file to include your servers, it’s time to check if Ansible is able to connect to these servers and run commands via SSH.

For this guide, we’ll be using the Ubuntu **root** account because that’s typically the only account available by default on newly created servers. If your Ansible hosts already have a regular sudo user created, you are encouraged to use that account instead.

You can use the -u argument to specify the remote system user. When not provided, Ansible will try to connect as your current system user on the control node.

From your local machine or Ansible control node, run:

1. ansible all -m ping -u root

This command will use Ansible’s built-in [ping module](https://docs.ansible.com/ansible/latest/modules/ping_module.html) to run a connectivity test on all nodes from your default inventory, connecting as **root**. The ping module will test:

* if hosts are accessible;
* if you have valid SSH credentials;
* if hosts are able to run Ansible modules using Python.

You should get output similar to this:

Output

server1 | SUCCESS => {

"changed": false,

"ping": "pong"

}

server2 | SUCCESS => {

"changed": false,

"ping": "pong"

}

server3 | SUCCESS => {

"changed": false,

"ping": "pong"

}

If this is the first time you’re connecting to these servers via SSH, you’ll be asked to confirm the authenticity of the hosts you’re connecting to via Ansible. When prompted, type yes and then hit ENTER to confirm.

Once you get a "pong" reply back from a host, it means you’re ready to run Ansible commands and playbooks on that server.

**Note**: If you are unable to get a successful response back from your servers, check our [Ansible Cheat Sheet Guide](https://www.digitalocean.com/community/tutorials/how-to-use-ansible-cheat-sheet-guide) for more information on how to run Ansible commands with different connection options.

## [Step 4 — Running Ad-Hoc Commands (Optional)](https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-20-04#step-4-running-ad-hoc-commands-optional)

After confirming that your Ansible control node is able to communicate with your hosts, you can start running ad-hoc commands and playbooks on your servers.

Any command that you would normally execute on a remote server over SSH can be run with Ansible on the servers specified in your inventory file. As an example, you can check disk usage on all servers with:

1. ansible all -a "df -h" -u root

Output

server1 | CHANGED | rc=0 >>

Filesystem Size Used Avail Use% Mounted on

udev 3.9G 0 3.9G 0% /dev

tmpfs 798M 624K 798M 1% /run

/dev/vda1 155G 2.3G 153G 2% /

tmpfs 3.9G 0 3.9G 0% /dev/shm

tmpfs 5.0M 0 5.0M 0% /run/lock

tmpfs 3.9G 0 3.9G 0% /sys/fs/cgroup

/dev/vda15 105M 3.6M 101M 4% /boot/efi

tmpfs 798M 0 798M 0% /run/user/0

server2 | CHANGED | rc=0 >>

Filesystem Size Used Avail Use% Mounted on

udev 2.0G 0 2.0G 0% /dev

tmpfs 395M 608K 394M 1% /run

/dev/vda1 78G 2.2G 76G 3% /

tmpfs 2.0G 0 2.0G 0% /dev/shm

tmpfs 5.0M 0 5.0M 0% /run/lock

tmpfs 2.0G 0 2.0G 0% /sys/fs/cgroup

/dev/vda15 105M 3.6M 101M 4% /boot/efi

tmpfs 395M 0 395M 0% /run/user/0

...

The highlighted command df -h can be replaced by any command you’d like.

You can also execute [Ansible modules](https://docs.ansible.com/ansible/latest/modules/modules_by_category.html) via ad-hoc commands, similarly to what we’ve done before with the ping module for testing connection. For example, here’s how we can use the apt module to install the latest version of vim on all the servers in your inventory:

1. ansible all -m apt -a "name=vim state=latest" -u root

You can also target individual hosts, as well as groups and subgroups, when running Ansible commands. For instance, this is how you would check the uptime of every host in the servers group:

1. ansible servers -a "uptime" -u root

We can specify multiple hosts by separating them with colons:

1. ansible server1:server2 -m ping -u root

In this guide, you’ve installed Ansible and set up an inventory file to execute ad-hoc commands from an Ansible Control Node.

Once you’ve confirmed you’re able to connect and control your infrastructure from a central Ansible controller machine, you can execute any command or playbook you desire on those hosts.