**Topics**

**What is Ansible**

**Why Ansible**

**Inventory**

**YAML**

**Playbook**

**Ansible Installation**

**Hosts Configuration(client)**

**Inventory configuration**

**Write a playbook**

**Execute playbook to all hosts**

# Pull Based Configuration Management Tool

1. In this type of configuration management tool, the nodes pull the configuration information from the server (hence, the name).
2. A small software (called agent or client) is installed on every node. This agent/client will:
3. at regular intervals, get the configuration from the server
4. compare the configuration received from the server with the current configuration of the node
5. if there is any miss-match, take the steps required to match the configuration of the node with the configuration received from the server.
6. This means that, its always the agent/client that initiates communication, not the main server.
7. Chef & Puppet are good examples of such configuration management tools.

# Push Based Configuration Management Tool

1. In this type of configuration management tool, the main server (where the configuration data is stored) pushes the configuration to the node (hence, the name). So, it is the main server that initiates communication, not the nodes. Which means that an agent/client may or may not be installed on each node.

# Installation

### Installing Ansible:

*sudo apt update*

The APT package repository cache should be updated.

Now, install Ansible with the following command:

*sudo apt install ansible*

To confirm the installation, press Y and then press <Enter>.

Now, run the following command to check if ansible is working correctly.

$ ansible --version

As we can see, ansible command is available and working correctly.

### Generating SSH Key:

Now, you have to generate an SSH key on the computer where you have installed Ansible.

To generate an SSH key, run the following command:

$ ssh-keygen

Now, press <Enter>.

Press <Enter>.

An SSH key should be generated.

### Configuring Ubuntu Hosts for Ansible Automation:

In this section, I will show you how to configure an Ubuntu host (host1) for Ansible automation. If you have more than one host which you want to automate using Ansible, then repeat the same process on each of the hosts.

The Ubuntu Ansible hosts (which you want to configure for Ansible automation) must have SSH server package installed.

First, update the APT package repository cache with the following command:

$ sudo apt update

Then, install OpenSSH server with the following command:

$ sudo apt install openssh-server

In my case, the OpenSSH server package is already installed. If it is not installed in your case, it should be installed.

Now, check if the sshd service is running with the following command:

$ sudo systemctl status sshd

As you can see, the sshd service is active (running) and enabled (will automatically start on system boot).

If the sshd service is not active (running) in your case, start it manually with the following command:

$ sudo systemctl start sshd

If the sshd service is not enabled (not added to the system startup) in your case, add it to the system startup manually with the following command:

$ sudo systemctl enable sshd

Now, configure the firewall to allow SSH access with the following command:

$ sudo ufw allow ssh

You should also create an ansible user and allow password-less sudo access to the ansible user.

To create an ansible user, run the following command:

$ sudo adduser ansible

Now, type in a password for the ansible user and press <Enter>.

An ansible user should be created.

Now, configure password-less sudo access to the ansible user with the following command:

$ echo "ansible ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/ansible

Now, find the IP address of the Ansible host (host1) with the following command:

$ hostname -I

Here, the IP address in my case is 192.168.20.162. It will be different for you. So, make sure to replace it with yours form now on.

Now, from the computer where you have installed Ansible, copy the SSH public key to the Ansible host (host1) as follows:

$ ssh-copy-id ansible@192.168.20.162

Type in yes and press <Enter>.

Now, type in the password for the ansible user and press <Enter>.

The public SSH key should be copied to host1.

Now, disable password-based login for the ansible user with the following command:

$ sudo usermod -L ansible

Now, you can only SSH into the Ansible host (host1) as ansible user without any password from the computer you have copied the SSH public key from (in this case, the computer where you have installed Ansible). But you won’t be able SSH into the Ansible host (host1) as ansible user from any other computer. I have configured the Ansible hosts this way for security reasons. As the ansible user don’t need any password for running administrative commands, it’s risky to allow password-based login for the ansible user.

Now, you should be able to SSH into the ansible host host1 from the computer where you have installed Ansible as follows:

$ ssh ansible@192.168.20.162

As you can see, I am able to access the Ansible host (host1) as ansible user without any password. So, the Ansible host (host1) is ready for Ansible automation.

If for some reason, you want to allow password-based login for the ansible user again, run the following command in the Ansible host (host1):

$ sudo usermod -U ansible

You can configure as many hosts as you want for Ansible automation the same way.

In this article, I have configured only 2 hosts, host1 and host2 for the demonstration.

### Testing Ansible:

Now, create a new project directory ~/ansible-demo/ in the computer where you have installed Ansible as follows:

$ mkdir ~/ansible-demo

Now, navigate to the ~/ansible-demo/ directory as follows:

$ cd ~/ansible-demo/

Now, create a new hosts file in the project directory as follows:

$ nano hosts

Now, type in the IP addresses or DNS names of the Ansible hosts (host1 and host2 in my case) in the hosts file as follows:

192.168.20.162  
192.168.20.153

Now, save the file by pressing followed by Y and then .

Now, try to ping all the hosts using Ansible as follows:

$ ansible all -i ./hosts -u ansible -m ping

NOTE: Here, -u option is used to specify the username (ansible in this case) which Ansible will use to SSH into the hosts.

As you can see, all the hosts can be pinged. So, the hosts are ready for Ansible automation.

The same way, you can run any command in the hosts using Ansible as follows:

$ ansible all -i ./hosts -u ansible -m shell -a 'echo "$(hostname) - $(hostname -I)"'

As you can see, the command ran successfully in each of the hosts and the output is displayed.

# Advantages

1. Agentless
2. ssh
3. push-based
4. build with python

**Yaml**

* begins with ---
* next line with -
* hosts
* become
* tasks

**Useful Commands**

ansible all -m ping -u username

nano *etc*ansible/hosts

ansible-playbook pname -K

**references**

https://docs.ansible.com/ansible/latest/user\_guide/playbooks\_intro.html

https://docs.ansible.com/ansible/latest/modules/script\_module.html

https://linuxhint.com/install\_ansible\_ubuntu/