Documentations on Ansible Playbook for installation of 5 Linux Packages on Slave Server

*Detailed documentations on Ansible Playbook for installation of 5 Linux Packages*

* *What is Ansible?*

*Ansible is an open-source IT automation tool that automates provisioning, configuration management, application deployment, orchestration, and many other manual IT processes. Unlike more simplistic management tools, Ansible users (like system administrators, developers and architects) can use Ansible automation to install software, automate daily tasks, provision infrastructure, improve security and compliance, patch systems, and share automation across the entire organization.*

* *What is Ansible Playbook?*

*An ansible playbook is a file that contains a list of tasks that automatically execute against hosts. It is an organized unit of scripts that defines work for a server configuration managed by the automation tool Ansible. Each module within an Ansible Playbook performs a specific task, and contains metadata that determines when and where a task is executed, as well as which user executes it.*

* *Introduction*

*Configuration management systems are designed to streamline the process of controlling large numbers of servers, for administrators and operations teams. They allow you to control many different systems in an automated way from one central location.*

*While there are many popular configuration management tools available for Linux systems, such as Chef and Puppet, these are often more complex than many people want or need. Ansible is a great alternative to these options because it offers an architecture that doesn’t require special software to be installed on nodes, using SSH to execute the automation tasks and YAML files to define provisioning details.*

* *Prerequisites*

*To follow this tutorial, you will need:*

*One Ansible Control Node: The Ansible control node is the machine we’ll use to connect to and control the Ansible hosts over SSH. Your Ansible control node can either be your local machine or a server dedicated to running Ansible, though this guide assumes your control node is an Ubuntu 20.04 system. Make sure the control node has:*

*A non-root user with Sudo privileges. To set this up, you can follow Steps 2 and 3 of our Initial Server Setup Guide for Ubuntu 20.04. However, please note that if you’re using a remote server as your Ansible Control node, you should follow every step of this guide. Doing so will configure a firewall on the server with ufw and enable external access to your non-root user profile, both of which will help keep the remote server secure.*

*An SSH keypair associated with this user. To set this up, you can follow Step 1 of our guide on How to Set Up SSH Keys on Ubuntu 20.04.*

*One or more Ansible Hosts: An Ansible host is any machine that your Ansible control node is configured to automate. This guide assumes your Ansible hosts are remote Ubuntu 20.04 servers. Make sure each Ansible host has:*

*The Ansible control node’s SSH public key added to the authorized keys of a system user. This user can be either root or a regular user with Sudo privileges. To set this up, you can follow Step 2 of How to Set Up SSH Keys on Ubuntu 20.04.*

*Step 1 - Installing Ansible*

* *To begin using Ansible as a means of managing your server infrastructure, you need to install the Ansible software on the machine that will serve as the Ansible control node.*
* *From your control node, run the following command to include the official project’s PPA (personal package archive) in your system’s list of sources:*

|  |
| --- |
| *$ 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:*
* *Following this update, you can install the Ansible software with:*

|  |
| --- |
| *$ sudo apt install ansible* |

* *Your Ansible control node now has all the software required to administer your hosts. Next, we will go over how to add your hosts to the control node’s inventory file so that it can control them.*

## [*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, 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:*
* *The default inventory file provided by the Ansible installation contains several 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.*

|  |
| --- |
| *$ sudo nano /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:*

|  |
| --- |
| *$ ansible-inventory --list -y* |

* *You’ll see output like 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 can 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:*

|  |
| --- |
| *$ 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 can run Ansible modules using Python.*
* *You should get output like 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 from a host, it means you’re ready to run Ansible commands and playbooks on that server.*

|  |
| --- |
| *---*  *- hosts: server1*  *user: himanshuw*  *become: yes*  *tasks:* |

* *First, we will specify the hosts name in /etc/ansible/hosts I.e server1 along with the IP of the slave server. Followed by specifying user and its name, then become: yes, for privilege escalation systems to execute tasks with root privileges or with another user’s permissions. Because this feature allows you to ‘become’ another user, different from the user that logged into the machine (remote user), we call it become.*
* *Then finally specify tasks: to perform.*

|  |
| --- |
| *#######Install-Postfix#######*  *- name: Postfix package installation*  *apt:*  *name: postfix*  *state: present*  *- name: Ensure postfix service is running*  *service:*  *name: postfix*  *state: started*  *- name: Enable postfix on System Boot*  *service:*  *name: postfix*  *enabled: yes* |

* *Beginning with Installing Postfix as we have ubuntu server we use apt as a package manager, giving it a name as postfix where state is present.*
* *Next, we must ensure postfix service is running fine and its state should be started.*
* *Here after, enabling postfix on System Boot so that package could get installed correctly.*

|  |
| --- |
| *#######Install-Git#######*  *- name: Git package installation*  *apt:*  *name: git*  *state: present*  *update\_cache: yes* |

* *Now for installing Git package by apt, specifying name as git where state is present and update cache allows ansible’s apt module to refresh the caches before applying whatever change is necessary (if any)*

|  |
| --- |
| *#######Install-JDK#######*  *- name: Update APT package manager repositories cache*  *become: yes*  *apt:*  *update\_cache: yes*    *- name: JDK package installation*  *become: yes*  *apt:*  *name: openjdk-8-jdk*  *state: present* |

* *Now installing JDK first by updating APT package manager repositories cache and become: yes, by allowing root privileges and apt for installing in ubuntu server & update\_cache: yes*
* *Next step is to name the JDK package while allowing root privileges and giving name: openjdk-8-jdk and state: present for building stack/app whether app deploy or dependency version bump.*

|  |
| --- |
| *#######Install-tree#######*  *- name: apt update && apt install tree -y*  *apt:*  *update\_cache: yes*  *name: tree*  *state: present*  *#######Install-Nginx#######* |

* *Similarly, for installing tree package by apt-update (system update) and install tree –y (by command and allowing yes by –y)*
* *Also, update\_cache: yes, giving name as tree and its state to be present to install successfully.*

|  |
| --- |
| *#######Install-Nginx#######*  *- name: Nginx package installation*  *apt:*  *name: nginx*  *state: present* |

* *At last ending with installing Nginx package with name: nginx & state: present.*

*Complete Ansible Playbook as follows:*

|  |
| --- |
| *---*  *- hosts: server1*  *user: himanshuw*  *become: yes*  *tasks:*  *#######Install-Postfix#######*  *- name: Postfix package installation*  *apt:*  *name: postfix*  *state: present*  *- name: Ensure postfix service is running*  *service:*  *name: postfix*  *state: started*  *- name: Enable postfix on System Boot*  *service:*  *name: postfix*  *enabled: yes*  *#######Install-Git#######*  *- name: Git package installation*  *apt:*  *name: git*  *state: present*  *update\_cache: yes*  *#######Install-JDK#######*  *- name: Update APT package manager repositories cache*  *become: yes*  *apt:*  *update\_cache: yes*    *- name: JDK package installation*  *become: yes*  *apt:*  *name: openjdk-8-jdk*  *state: present*  *#######Install-tree#######*  *- name: apt update && apt install tree -y*  *apt:*  *update\_cache: yes*  *name: tree*  *state: present*  *#######Install-Nginx#######*  *- name: Nginx package installation*  *apt:*  *name: nginx*  *state: present* |