**Table Of Contents**

1. **Virtualization**
2. **Install VirtualBox**
3. **Install Vagrant**
4. **Verify Vagrant Installation**
5. **Create Directory for Vagrant**
6. **Initialize Vagrant**
7. **Configure Linux flavour**
8. **Configure MySql and Backup Server**
9. **Configure Ansible Controller for Configuration Management**
10. **Start the servers**
11. **SSH in to the servers**
12. **Ansible Installation**
13. **File Structure**
14. **MySql Installation**
15. **Backing up MySql**
16. **Verifying Backups**

Conventions used in the document :

Code : **vagrant**

Directory : **vagrant-servers**

File : **Vagrantfile**

Servers : **ansible.controller**

Important points : **Start the servers**

1. **Virtualization**

There are a few options when it comes to virtualization and provisioning virtual servers like VMware, Vagrant, VirtualBox, etc.

We can provision VMware’s using Ansible but we will move ahead with Vagrant as it has the below advantages.

1. Can be setup and configured easily

2. Provisions virtual servers quickly by executing a few commands

3. Reproduces the exact same environment no matter the number of servers we provision

4. Servers can be provisioned with packages preinstalled even during the first launch

5. Consumes less space as compared to VM's

6. Can clean up the machines easily after its work is done with a single command

1. **Install VirtualBox**

Vagrant is dependent on VirtualBox. For Vagrant to work, we need to install Virtual Box first.

The link to download VirtualBox is <https://www.virtualbox.org/wiki/Downloads>

We will be using Windows platform to install VirtualBox.

1. **Install Vagrant**

After successful installation of VirtuaBox, we will proceed with Vagrant installation.

The link to download Vagrant is <https://www.vagrantup.com/downloads.html>

We will be using Windows platform to install Vagrant.

1. **Verify Vagrant Installation**

Once Vagrant is installed, we can verify by typing the below command in the terminal/command prompt.

**vagrant**

**Note** - We will be using **GitBash** to interact with Vagrant

1. **Create Directory for Vagrant**

Create a directory on your machine at any location. Vagrant does not take much space but ensure that there is enough space for us to provision vagrant servers. We will name the directory as **vagrant-servers**.

1. **Initialize Vagrant**

We will now initialize the vagrant file in order to provision linux servers for us.

Go inside the **vagrant-servers** directory and start the Git bash terminal.

Type the command**vagrant init** and hit enter. This should create a **Vagrantfile** to configure vagrant.

1. **Configure Linux flavour**

Open **Vagrantfile** to configure which servers to create. We will go ahead with centos

flavour of linux.

Look for the configuration **config.vm.box**. Change its value to to become as below.

**config.vm.box = "bento/centos-7.3"**

This will ensure that Centos flavour is provisioned in the Linux servers.

1. **Configure MySql and Backup Server**

Now we will assign hostnames, Ip address and network to the servers.

Add the below code to the **Vagrantfile**.

**config.vm.define "mysql1.local" do |server|**

**server.vm.hostname = "mysql1.local"**

**server.vm.network :private\_network, ip: "192.168.10.6"**

**end**

**config.vm.define "backup1.local" do |server|**

**server.vm.hostname = "backup1.local"**

**server.vm.network :private\_network, ip: "192.168.10.7"**

**end**

1. **Configure Ansible Controller for Configuration Management**

We will also provision an **ansible.controller** linux server for configuration management. Add the below code in the **Vagrantfile**.

**config.vm.define "ansible.controller" do |server|**

**server.vm.hostname = "ansible.controller"**

**server.vm.network :private\_network, ip: "192.168.20.5"**

**end**

1. **Start the servers**

We will now start the server and then **ssh** into it to verify if the servers are provisioned and are able to connect to each other.

To start the servers use the below commands.

**vagrant up mysql1.local**

**vagrant up backup1.local**

**vagrant up ansible.controller**

1. **SSH into in the servers**

To SSH into the servers, we can enter the below commands on distinct terminals.

**vagrant ssh mysql1.local**

**vagrant ssh backup1.local**

**vagrant ssh ansible.controller**

1. **Ansible Installation**
2. **Install Ansible**

We will now install Ansible on **ansible.controller** linux server that we provisioned using Vagrant in Section A. This server will act as a controller machine which will be responsible to install and configure packages on other machines (**mysql1.local** and **backup1.local** in our case)

SSH into the **ansible.controller** server using vagrant using the below command.

**vagrant ssh ansible.controller**

Install ansible using yum package manager in Centos by entering this command.

**sudo yum install ansible –y**

Use **ansible --version** to verify if ansible is installed successfully. It should return the latest ansible version that we installed.

1. **Sharing the public key**

Create new user "ansible" : adduser ansible

Set user password : passwd ansible

Add user to the wheel group : usermod -aG wheel ansible

This needs to be done on all the servers viz controller and the nodes.

SSH into the **ansible.controller** now and then create and copy the ssh key to the nodes.

Create SSH key : **ssh-keygen**

Copy SSH public key to nodes :

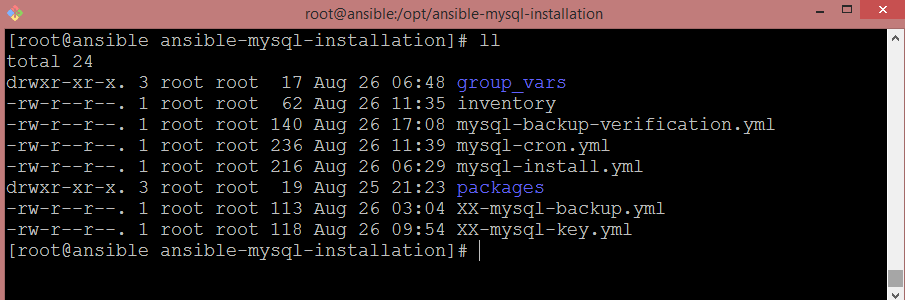
**sudo ssh-copy-id –i ~/.ssh/id\_rsa.pub 192.168.10.6**

**sudo ssh-copy-id –i ~/.ssh/id\_rsa.pub 192.168.10.7**

We can now access the nodes using Ansible.

1. **File Structure**

The file structure is as below:



group\_vars: Generated using ansible vault. Stores data for groups.

inventory: Stores the inventory rg. Mysql1.local and backup1.local servers

packages: Its purpose is to store each package/software directory which itself would contain ansible roles for better organization and reusability.

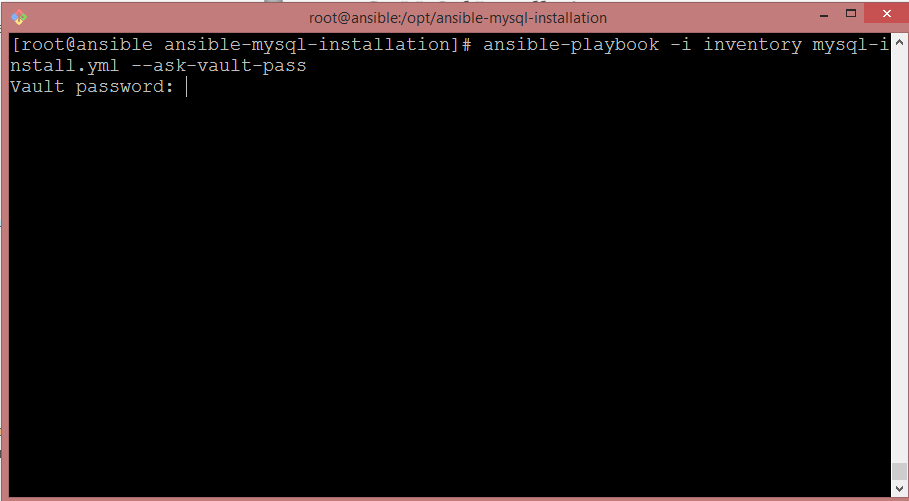
\*.yml : Run ansible playbooks using yml files and inventory. Demonstrated in below screenshots

Vault key: secretpassword

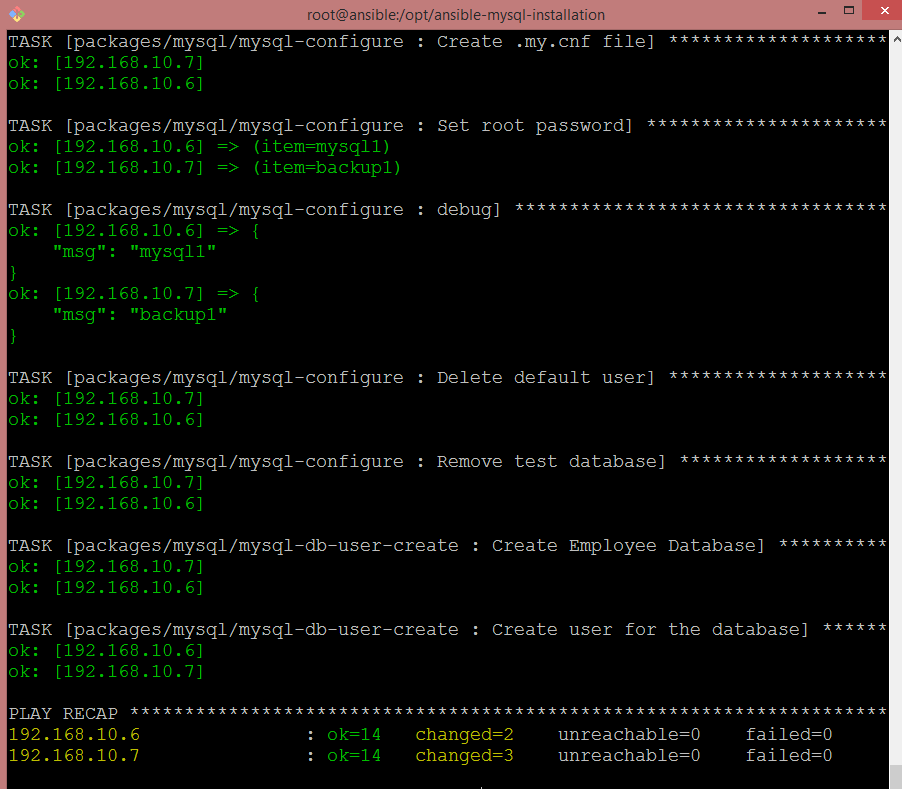
1. **MySql Installation**

To install MySql on the mysql1.local server, simply execute the ansible playbook as below

**sudo ansible-mysql-installation]# ansible-playbook -i inventory mysql-install.yml --ask-vault-pass**



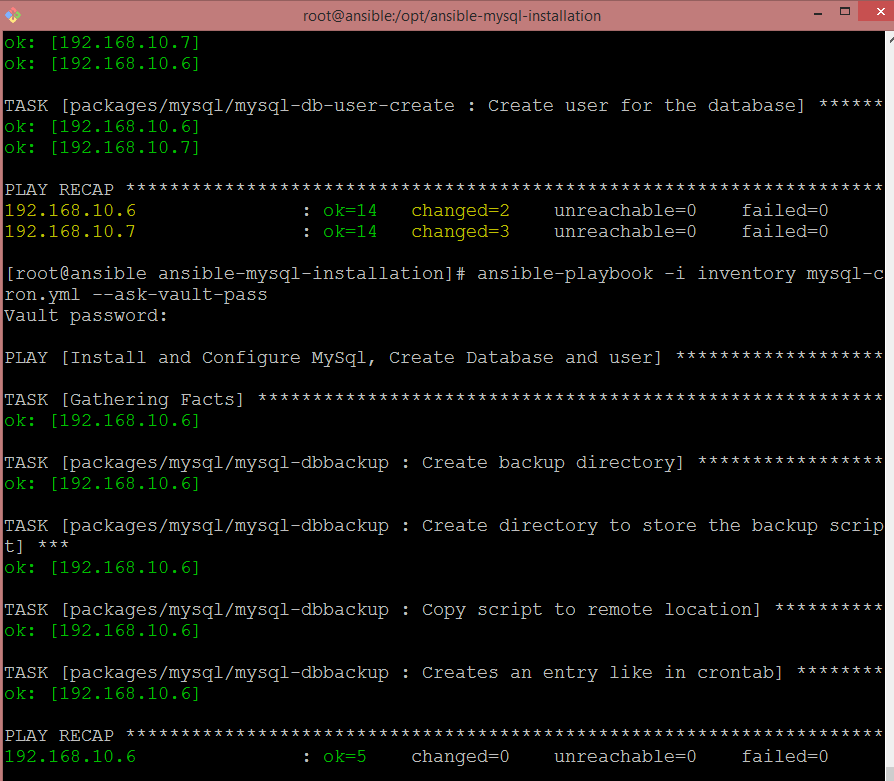
The playbook will prompt for Ansible vault password. We have used ansible vault for better security so that we do not expose the passwords or file path to anonymous user. Once, executed, we will get the following output.



1. **Backing up MySql**

Similarly, as above, run the ansible playbook as below.

**sudo ansible-playbook -i inventory mysql-cron.yml --ask-vault-pass**



This will create a cron job entry on **mysql1.local** server. It will run the job to backup the database

1. **Verifying the Backups**

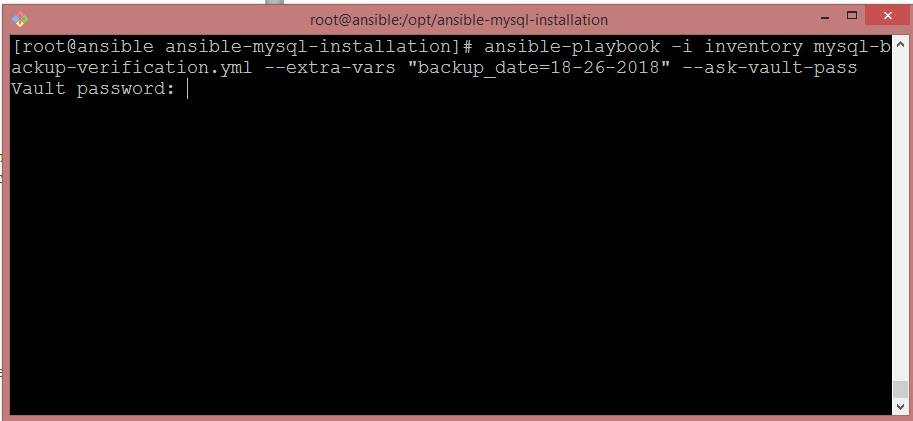
To verify the backup just run the below ansible command.

**sudo ansible-playbook -i inventory mysql-backup-verification.yml --extra-vars "backup\_date=18-26-2018" --ask-vault-pass**

Note that we have used ansible vault. So we prompt the user (--ask-ansible-vault) to enter

the password using which the user can access the ansible vault. The user also needs to enter

the date for which he needs to verify the backup for using the **backup\_date** variable.



Enter the vault password and the playbook will execute.

