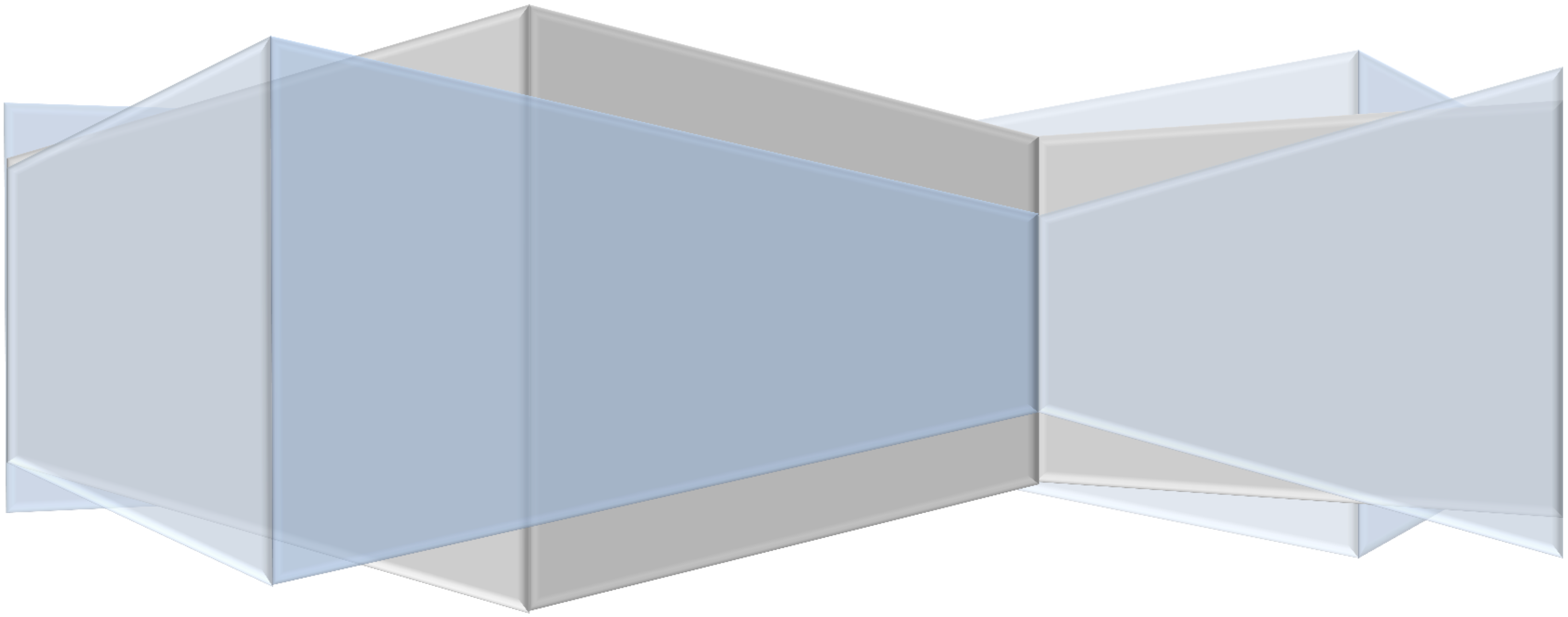
Automation Factory

Ansible – Configuration Management tool

Ganesh Palnitkar



Readying VMs for for Ansible installations.

This can be on a Debian or Centos machine. Using vagrant, use below vagrant file syntax to get the VMs up and running.

```
-------------------------------------------------------------------
 Vagrant.configure("2") do |config|

  config.vm.define "anserv" do |anserv|
    anserv.vm.box = "base"
    anserv.vm.network "forwarded_port", guest: 80, host: 8080
    anserv.vm.network "private_network", ip: "192.168.33.20"
    anserv.vm.provider "virtualbox" do |vb|
            vb.gui = true
            vb.memory = "1024"
    end
    anserv.vm.provision "shell", inline: <<-SHELL
            apt-get update
            SHELL
  end

  config.vm.define "webserv" do |webserv|
            webserv.vm.box = "nrel/Centos-6.5-x86_64"
            webserv.vm.hostname = "webserv"
            webserv.vm.network "private_network", ip: "192.168.33.21"
            webserv.vm.network "forwarded_port", guest: 80, host:8080
            webserv.vm.provider "virtualbox" do |vb|
            vb.gui = true
            vb.memory = "512"
    end
  end

  config.vm.define "db" do |db|
            db.vm.box = "nrel/Centos-6.5-x86_64"
            db.vm,hostname = "db"
            db.vm.network "private_network", ip: "192.168.33.22"
            db.vm.provider "virtualbox" do |vb|
            vb.gui = true
            vb.memory = "512"
    end
  end
end
---------------------------------------------
```

Making ready the Centos machine for Ansible installation

Once logged into the vm,

Update the repository with command,

```
$ sudo yum install -y epel-release

$ sudo yum install -y ansible
```

For installing ansible on RHEL8.

```
$ sudo dnf -y install python3-pip

$ sudo pip3 install --upgrade pip

$ sudo dnf -y install
https://dl.fedoraproject.org/pub/epel/epel-release-latest-
8.noarch.rpm

$ sudo dnf install  --enablerepo epel-playground  ansible
```

Install Ansible using Python way.

`$ sudo yum install -y gcc` … install compiler repository update

`$ sudo yum install -y python-setuptools`

`$ sudo easy_install pip` … python egg.. Manager

`$ sudo yum install -y python-devel` … install python dev library

`$ sudo pip install ansible` …. Download code, compile and install the package.

In order to have your Ansible Controller server talk to remote nodes over SSH, make sure to add the public key pair to the remote server. This will enable the user on Ansible server to login to remote server using **SSH protocol.**

If you face difficulty authenticating, try removing the entry for that remote server from the 'known_hosts' file on the Ansible server. This can be done by running below command,

```
$ ssh-keygen -f "/root/.ssh/known_hosts" -R <remote server name
/ IPaddress>
```

To install Ansible on Ubuntu server, use below commands.,

Installing Ansible on Linux variants:

- **Debian/Ubuntu:** The easiest way to install Ansible on a Debian or Ubuntu system is to use the official apt package.

```
$ sudo apt-add-repository -y ppa:ansible/ansible

$ sudo apt-get update

$ sudo apt-get install -y ansible
```

- Incase there's an error about Python, run below command.

```
$ sudo apt-get install python-software-properties
```

To start using the Ansible server for managing your infrastructure, at very first create an inventory file as below,

```
    $ nano inventory
    192.168.33.21
    192.168.33.22
    $ ansible 192.168.33.21 -i inventory -u vagrant -m ping -k
```

Here as the remote node is not a valid known_host for the Ansible server, this Ansible command will return an error stating the remote host is not trusted.

TO work around this error, run a command ssh to the remote server. This will add a fingerprint to the known_host file.

We can run the Ansible command in the debug mode as well.

This is done using below command,

```
$ ansible 192.168.33.21 -i inventory -u vagrant -m ping -k -vv
```

….. here –vv is for verbose mode (debug level 2)

Or,

```
$ ansible 192.168.33.21 -i inventory -u vagrant -m ping -k -vvv
```

… here vvv stands for verbose mode with more detailed report (debug level 3).

Starting with Ansible

A very powerful way to mass execute a command or a script over large number of environment machines, we can use Ansible using the adhoc command.

```
$ ansible 192.168.33.21 -i inventory -u vagrant -m command -a
"/usr/sbin/yum update -y"
```

Alternatively we can use the shell module as well which enables us to run shell commands, variables etc.

Inventory Files:

```
-------------------------------------------

dbserver ansible_ssh_host=192.168.33.22 ansible_ssh_user=vagrant
ansible_ssh_passwd=kjd&shd
[db]
dbserver
[web]
Webserver1.autofact.com
Webserver2.autofact.com
[datacenter:children]
web
[datacenter:vars]
ansible_ssh_user=ansible_user
ansible_ssh_passwd=%679jd
-------------------------------------------
```

Try running below adhoc ansible commands,

```
$ ansible dbserver -i inventory -m ping
```

Or

```
$ ansible db -i inventory -m ping
```

Or,

```
$ ansible datacenter -i inventory -m ping
```

In all these commands the vars are defined inside the inventory file.

Variable file:

- Create group variables in separate files
- Show order of precedence

Variable files are written in YAML

Directory structure

```
Production
        group_vars
                all
                db
        host_vars
                web1
        inventory_prod
Test
        group_vars
                all
                db
        host_vars
                web1
        inventory_test
```