Ansible

- Ansible
  - Ansible Setup
    - Launching Machines with Amazon Linux
  - Installing Ansible on Control Node:
    - Managing `ansible.cfg` file
    - Setup Passwordless ssh login from control node to managed node.
      - 1. Using Existing AWS EC2 Key Pair
      - 2. Generating new Key Pair
        - How it works
  - Ansible Modules
  - Running Ansible Ad Hoc Commands
    - Command vs Shell Modules
    - file module
    - copy module
    - setup module
  - Note
  - Reference

Ansible Setup

Launching Machines with Amazon Linux

- To Setup ansible on EC2 instances with Amazon Linux :
- Launch `3 EC2 instances` and tag one of the instance as control and managed nodes
- Change the hostname for ec2 instances on Amazon Linux AMI only, use below :
- On Control Node:

```
sudo hostnamectl set-hostname control-node.example.com
```

- On Managed Node:

```
sudo hostnamectl set-hostname managed-node-01.example.com
sudo hostnamectl set-hostname managed-node-02.example.com
```

- On Control Node:

```
[ec2-user@control-node ssm-user]$ ping managed-node-01.example.com
ping: managed-node-01.example.com: Name or service not known
```

- Edit the `/etc/hosts` file similar with below details:

```
172.31.26.166   control-node.example.com    control-node
172.31.27.151   managed-node-01.example.com managed-node-01
172.31.27.141   managed-node-02.example.com managed-node-02
```

Installing Ansible on Control Node:

- Check if Python 3 is installed on Linux:
- Installing ansible only on one node i.e only `Control Node`:
- For amazon Linux :

```
sudo yum install python3
sudo amazon-linux-extras install ansible2
```

- For Redhat Family `sudo yum install -y ansible`
- For debian Family `sudo apt-get install -y ansible`
- Create and change directory to the `/home/ec2-user/ansible-demo`

```
mkdir ansible-demo && cd ansible-demo
```

- Ansible works against multiple managed nodes or “hosts” in your infrastructure at the same time, using a list or group of lists know as `inventory`.

> The `/etc/ansible/hosts` file is considered the system's default static inventory file.

- Create inventory file with name `inventory` in the same current working directory with below content

```
[myhost]
localhost ansible_connection=local

[dev]
managed-node-01.example.com hostVariableName=hostVariableValue
ansible_connection=ssh

[web]
managed-node-02.example.com

[centos]
managed-node-03.example.com     ansible_user=centos

[myhost:vars]
username=myusername
password=mypassword
```

- These variables are defined in inventory file.
  - `host variables` : You can easily assign a variable to a single host, then use it later in playbooks.
  - `group variables` : These are variable values that are to be shared for all hosts in a group
- To view all the list of hosts from inventory file

```
ansible -i inventory all --list-hosts

ansible -i inventory dev --list-hosts -v

ansible --version
```

Managing `ansible.cfg` file

- Ansible searches for `ansible.cfg` in these locations in order for preceedence of config file:
  1. ANSIBLE_CONFIG (`environment variable if set`)
  2. ansible.cfg (`in the current directory`)
  3. ~/.ansible.cfg (`in the home directory as a hidden file`)
  4. `/etc/ansible/ansible.cfg`

```
export ANSIBLE_CONFIG=""
unset ANSIBLE_CONFIG
```

- Using `./ansible.cfg`
  - If an `ansible.cfg` file exists in the directory in which the `ansible` command is executed, it is used instead of the global file or the user's personal file.
  - This allows administrators to create a directory structure where different environments or projects are stored in separate directories, with each directory containing a configuration file tailored with a unique set of settings.
- Using `~/.ansible.cfg`
  - Ansible looks for a `~/.ansible.cfg` in the user's home directory.
  - If this file exists, this configuration is used instead of the `/etc/ansible/ansible.cfg` if there is no ansible.cfg file in the current working directory.
- Using `/etc/ansible/ansible.cfg`
  - The ansible package provides a base configuration file located at /etc/ansible/ansible.cfg. This file is used if no other configuration file is found.
- Lets create `ansible.cfg` file, it assumes that you can connect to the managed hosts as `ec2-user` using SSH key-based authentication, and that `ec2-user` can use sudo to run commands as root without entering a password:

```
```

```
[defaults]
interpreter_python=/usr/bin/python3
inventory = ./inventory
remote_user = ec2-user
ask_pass = false
private_key_file=/path/to/file.pem
host_key_checking = False

[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ask_pass = false
```

- Check details of the config file using below command from different directories:

```
ansible --version
```

- Here, the config file options changes as per path from where ansible command is executed.
- verify sections inside a `ansible.cfg` file

```
grep "^\[" /etc/ansible/ansible.cfg
```

Below are the sections in ansible.cfg file

- `[defaults]` - Most of the settings in the configuration file are grouped here
- `[privilege_escalation]` - This section contains settings for defining how operations that require escalated privileges are executed on managed hosts.
- **[defaults]**
    - `interpreter_python` - Python Executable binary path
    - `inventory` - specifies the path of your inventory file
    - `remote_user` - specifies the user who will connect to the managed hosts and run the playbooks
    - `private_key_file` - specifies the private key identity file to be used when connecting to remote server.
- **[privilege_escalation]**
    - `become` - specify where to allow/disallow privilege escalation; default is False.
    - `become_method` - specify the privilege escalation method; default is `sudo`.
    - `become_user` - specify the user you become through privilege escalation; default is root.
    - `become_ask_pass` - specify whether to ask or not ask for privilege escalation password; default is False

Setup Passwordless ssh login from control node to managed node.

- Passwordless ssh can be setup by using one of the below methods:

1. Using Existing AWS EC2 Key Pair

- As these EC2 instances are creating with same Key Pair, the `key.pem` used to connect to the instance can be copied on the Control Node Path under the same path as inventory file.
- Property `private_key_file=/path/to/file.pem` under ansible.cfg file can be used to specify the path to private key.
- Test the ssh connection from control node to managed nodes.

```
ssh -i /path/to/file.pem ec2-user@managed-node-01
```

2. Generating new Key Pair

`ssh ec2-user@managed-node-01`

- Since we are on linux node, if we can log in to the remote user with a password then you can probably set up SSH key-based authentication, which would allow you to set `ask_pass = false`.
- The first step is to make sure that the `user on the control node` has an SSH key pair configured in `~/.ssh`.
- We can run below command to accomplish this.
- Here we will configure ssh keypair and copy the public key to all the remote machines.
- On Control Node:

```
ssh-keygen
```

For a single existing managed host, you can install your public key on the managed host and populate your local `~/.ssh/known_hosts` file with its host key using the `ssh-copy-id` command:

- Using `ssh-copy-id` to copy keys
- To configure passwordless ssh from control node:
- setup password for managed nodes using `passwd` command for ec2-user.
- On Managed Node
    - Execute below command to Update the password.

      ```
      sudo passwd ec2-user
      ```

    - change `PasswordAuthentication` property inside `/etc/ssh/sshd_config` to `yes` and restart the `sshd` service on all managed nodes `sudo service sshd restart`
- Once a Key Pair is created using `ssh-keygen` command on control node, we can use below command on control node to copy the public keys to all managed nodes under `~/.ssh/authorized_keys` file.

```
[ec2-user@control-node ~]$ ssh-copy-id ec2-user@managed-node-01.example.com
------------------Command Output--------------------
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ec2-
user/.ssh/id_rsa.pub"
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now
it is to install the new keys
ec2-user@managed-node-01.example.com's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ec2-user@managed-node-
01.example.com'"
and check to make sure that only the key(s) you wanted were added.

------------------Command Output--------------------
ssh-copy-id ec2-user@managed-node-02.example.com
```

- Verify whether above commands have copied the public key on managed nodes under `~/.ssh/authorized_keys`.

How it works

- The `ssh-copy-id` command logs onto a server using another authentication method (normally a password).
- It then checks the permissions of the user's .ssh directory and copies the new public key into the `~/.ssh/authorized_keys` file.

Ansible Modules

Running Ansible Ad Hoc Commands

- General Syntax of Ansible Ad-Hoc Command is:
- `ansible host-pattern -m module [-a 'module arguments'] [-i inventory]`

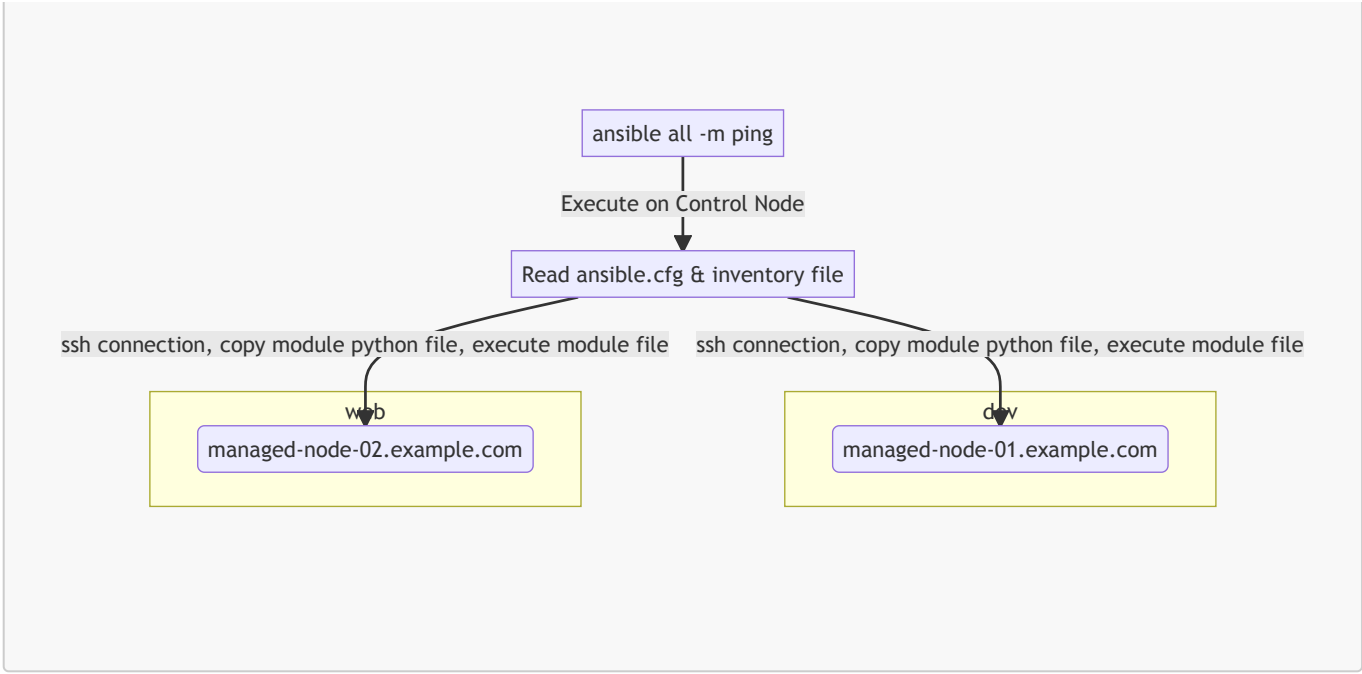
```
ansible all -m ping
ansible dev -m ping
ansible web -m ping
```

```
[ec2-user@control-node ansible-demo]$ ansible all -m ping
localhost | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
managed-node-01.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
managed-node-02.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ec2-user@control-node ansible-demo]$ ansible dev -m ping
managed-node-01.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ec2-user@control-node ansible-demo]$ ansible web -m ping
managed-node-02.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ec2-user@control-node ansible-demo]$
```

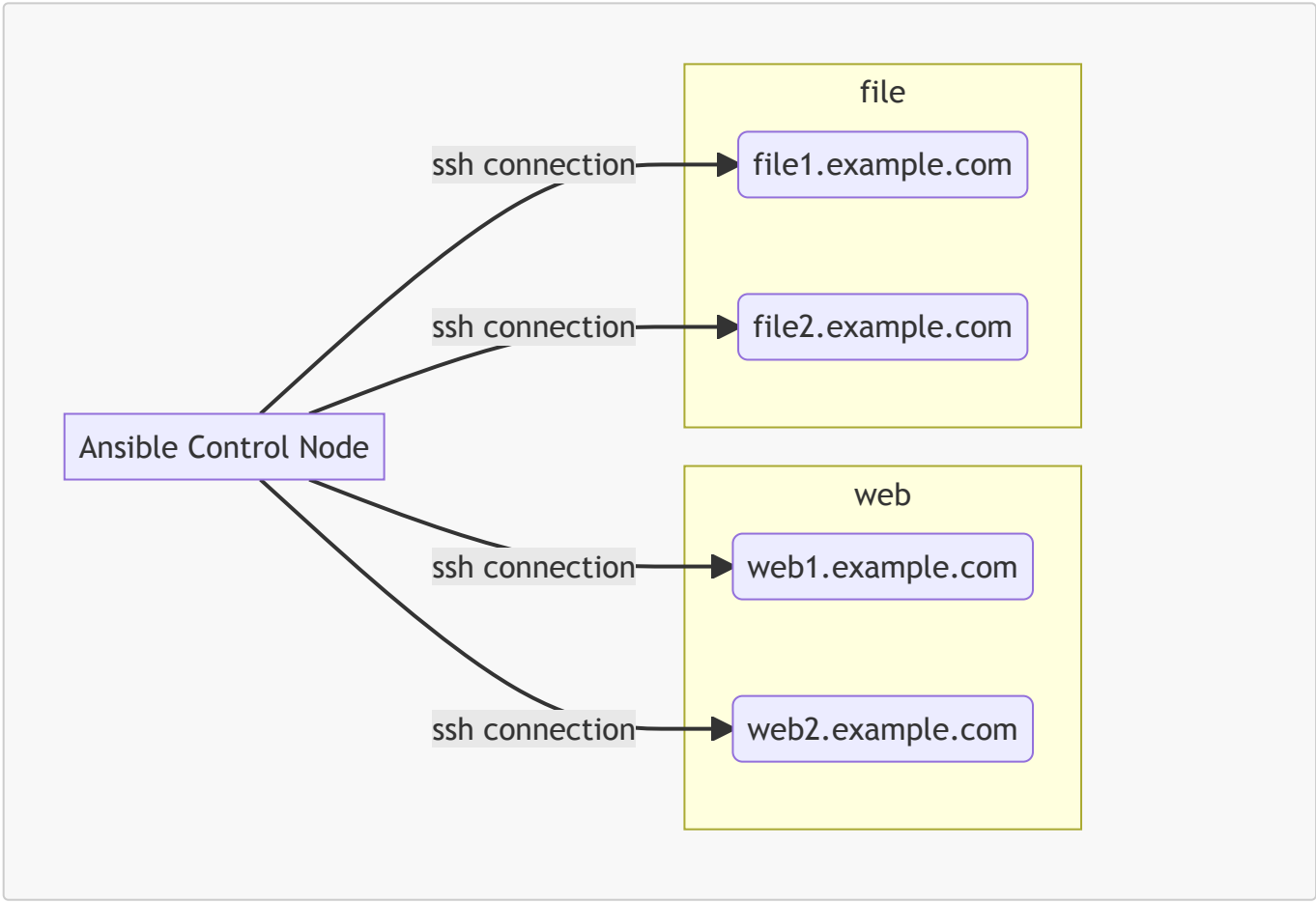
- When you get `SUCCESS` response, it means the module has be successfully executed on remote host.
- To get more verbose mode of the execution use:

```
ansible all -m ping -v
ansible all -m ping -vv
ansible all -m ping -vvv
```

- Execution information of **ansible ping module**



- Sample scenario of ssh connection into remote managed nodes.



- In above verbose mode information, ansible uses SSH authentication in the background to connect to managed nodes, uses the module file (for e.g `ping.py`), copies this module file from control node over managed node in a temporary location and executes the file using python `/usr/bin/python`.
- On execution of this module file, there is `success/failure` response that is returned.
- If there is ping issue for `localhost`, then try ssh localhost command or append public key to authorized_keys file on local server.

- To view all the modules that are present in the module: `ansible-doc -l`
- To get the number of modules supported by Ansible: `ansible-doc -l | wc -l`
- Add a new linux user

```
ansible dev -m user -a 'name=test_user uid=2000 state=present'
```

- `Idempotent`
  - Idempotent means modules that can run repeatedly to ensure systems are in a particular state without disrupting those systems if they already are.
  - To check this, we can run the previous ad-hoc again.
  - Here, if ansible has previously made some change, if the same command is executed again, there will be no change i.e `changed : false`

```
ansible dev -m user -a 'name=test_user uid=2000 state=present'
```

```
[ec2-user@control-node ansible-demo]$ ansible dev -m user -a 'name=test_user uid=2000 state=present'
managed-node-01.example.com | CHANGED => {
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 2000,
    "home": "/home/test_user",
    "name": "test_user",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 2000
}
[ec2-user@control-node ansible-demo]$ ansible dev -m user -a 'name=test_user uid=2000 state=present'
managed-node-01.example.com | SUCCESS => {
    "append": false,
    "changed": false,
    "comment": "",
    "group": 2000,
    "home": "/home/test_user",
    "move_home": false,
    "name": "test_user",
    "shell": "/bin/bash",
    "state": "present",
    "uid": 2000
}
[ec2-user@control-node ansible-demo]$
```

- Remove the Linux User

```
ansible dev -m user -a 'name=test_user uid=2000 state=absent'
```

- If we change the `./ansible.cfg` file and comment the `privilege_escalation`, the above command will not allow `ec2-user` to run `sudo` operations.

```
[ec2-user@control-node ansible-demo]$ cat ansible.cfg
[defaults]
interpreter_python=/usr/bin/python3
inventory = ./inventory
remote_user = ec2-user
ask_pass = false
private_key_file=./aws-linux-mumbai.pem
host_key_checking = False

#[privilege_escalation]
#become = true
#become_method = sudo
#become_user = root
#become_ask_pass = false
[ec2-user@control-node ansible-demo]$ ansible dev -m user -a 'name=new_test_user uid=2001 state=present'
managed-node-01.example.com | FAILED! => {
    "changed": false,
    "cmd": "/sbin/useradd -u 2001 -m new_test_user",
    "msg": "[Errno 13] Permission denied: b'/sbin/useradd'",
    "rc": 13
}
[ec2-user@control-node ansible-demo]$
```

- The following modules are useful:
  - `copy` (copy a local file to the managed host)
  - `get_url` (download a file to the managed host)
  - `file` (set permissions and other properties of a file)
  - `synchronize` (to synchronize content like rsync)
  - `lineinfile` (make sure a certain line is or isn't in a file)
  - Software package management modules, such as `yum`, `apt`, `pip` and so on
  - System administration tools, such as
    - `service` to control daemons
    - `user` module to add, remove and configure users
    - `uri`, which interacts with a web server and can test functionality or issue API requests

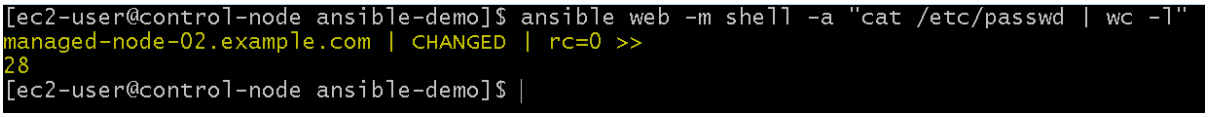
Command vs Shell Modules

- The `command` module allows administrators to execute arbitrary commands on the command line of managed hosts.

```
ansible all -m command -a /usr/bin/hostname
ansible localhost -m command -a 'id'
ansible localhost -m command -a 'df -h'
ansible localhost -m command -a 'cat /etc/passwd' -vvv
```

There are very significant differences in below modules:

- `command`
  - `piping or redirection` operations are not supported with the `command` module
  - Below command will result into an error.
  - `ansible web -m command -a "cat /etc/passwd | wc -l"`
  -
```
[ec2-user@control-node ansible-demo]$ ansible web -m command -a "cat /etc/passwd | wc -l"
managed-node-02.example.com | FAILED | rc=1 >>
cat: invalid option -- 'l'
Try 'cat --help' for more information.non-zero return code
[ec2-user@control-node ansible-demo]$
```

- `shell`
    - `piping or redirection` operations are supported with the `shell` module
    - Below command will result into an output
    - `ansible web -m shell -a "cat /etc/passwd | wc -l"`
    - 

- Examples of linux commands that can be executed on a ad-hoc basis.

```
ansible all -m shell -a "cat /etc/passwd | grep -i '/bin/bash' | awk -F: '{print $1}'"
```

- 

file module

- `file` module commands
    - Create a directory directly to many servers

```
ansible all -m file -a 'dest=/tmp/new-directory mode=755 owner=ec2-user group=ec2-user state=directory'
```

    - Delete a directory directly from many servers

```
ansible all -m file -a 'dest=/tmp/new-directory mode=755 owner=ec2-user group=ec2-user state=absent'
```

copy module

- `copy`
    - Transfer a file from control node to managed servers

```
ansible all -m copy -a 'src=/etc/hosts dest=/tmp/hosts'
```

- Write some content to this file

```
ansible dev -m copy -a 'content="This file is used and Managed by
Ansible\n" dest=/tmp/test_file'
```

- 

- To view the content of the file

```
ansible all -m command -a 'cat /tmp/test-file'
```

setup module

- gather facts

```
ansible all -m setup
```

- filter facts

```
ansible all -m setup -a 'filter=ansible_hostname'
ansible all -m setup -a 'filter=ansible_fqdn'
ansible all -m setup -a 'filter=ansible_pkg_mgr'
ansible all -m setup -a 'filter=ansible_os_family'
ansible all -m setup -a 'filter=ansible_eth0'
ansible all -m setup -a 'filter=ansible_distribution'
```

Note

> As running multiple instances running in EC2 might incur costs over and above the Free Tier Limit, make sure EC2 instances are stopped if they are not in use.

Reference

- The below command will open up the `ping` module documentation page to get more detail information of all the options supported by this module.

```
```

```
ansible-doc ping
```

```
[ec2-user@control-node ansible-demo]$ ansible-doc ping
> PING    (/usr/lib/python2.7/site-packages/ansible/modules/system/ping.py)

        A trivial test module, this module always returns `pong' on successful contact. It does not make
        sense in playbooks, but it is useful from `/usr/bin/ansible' to verify the ability to login and that
        a usable Python is configured. This is NOT ICMP ping, this is just a trivial test module that
        requires Python on the remote-node. For Windows targets, use the [win_ping] module instead. For
        Network targets, use the [net_ping] module instead.

  * This module is maintained by The Ansible Core Team
OPTIONS (= is mandatory):

- data
        Data to return for the `ping' return value.
        If this parameter is set to `crash', the module will cause an exception.
        [Default: pong]
        type: str
```