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.
    - How it works
    - Running Ad Hoc Commands
  - Working with playbooks
    - Executing Ansible Playbooks

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
```

- 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:

- Installing ansible only on one node i.e only `Control Node`:
- For amazon Linux : `sudo amazon-linux-extras install ansible2`
- For Redhat Family `sudo yum install -y ansible`
- For debian Family `sudo apt-get install -y ansible`

- 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 and change directory to the `/home/ec2-user/ansible-demo`

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

- Create inventory file with name `inventory` 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

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

- `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
```

- `hostvars` and `groupvars`

Managing `ansible.cfg` file

- Ansible searches for ansible.cfg in these locations in order:

    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`

- 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 someuser using SSH key-based authentication, and that someuser can use sudo to run commands as root without entering a password:

```
[defaults]
inventory = ./inventory
remote_user = ec2-user
ask_pass = 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 .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]` - Most of the settings in the configuration file are grouped here

Setup Passwordless ssh login from control node to managed node.

`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.

```
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.
- change `PasswordAuthentication` property inside `/etc/ssh/sshd_config` to yes and restart the sshd service on all managed nodes.
- use `ssh-keygen` command to generate you own keypair on control node and then using below command on control node to copy the public key to all managed nodes
- Now try the ssh-copy-id command from control node to all your managed nodes.

```
ssh-copy-id ec2-user@managed-node-01.example.com
```

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 key into the authorized_keys file.

Running Ad Hoc Commands

```
ansible host-pattern -m module [-a 'module arguments'] [-i inventory]
```

```
ansible all -m ping
```

- If there is ping issue for localhost, then try ssh localhost command and setup password for ec2-user
- To view all the modules that are present in the module: `ansible-doc -l`
- The following modules might be immediately useful:
- File modules, such as copy (copy a local file to the managed host), get_url (download a file to the managed host), synchronize (to synchronize content like rsync), file (set permissions and other properties of a file), and lineinfile (make sure a certain line is or isn't in a file)
- Software package management modules, such as yum, dnf, apt, pip, gem, 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
- Add a new linux user

```
ansible dev -m user -a 'name=new_user uid=4000 state=present' > managed-node-
01.example.com
```

- `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.
- Remove the Linux User

```
ansible dev -m user -a 'name=new_user uid=4000 state=absent' > managed-node-
01.example.com
```

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

- `file` - 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` - 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 localhost -m copy -a 'content="This file is used and Managed by Ansible\n" dest=/etc/test-file'
```

- To view the content of the file

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

- gather facts

```
ansible all -m setup
```

- filter facts

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

Working with playbooks

- Ad hoc commands can run a single, simple task against a set of targeted hosts as a one-time command.
- The real power of Ansible, however, is in learning how to use playbooks to run multiple, complex tasks against a set of targeted hosts in an easily repeatable manner.
- A play is an ordered set of tasks which should be run against hosts selected from your inventory.
- A playbook is a text file that contains a list of one or more plays to run in order.

> In simple words :
> *`Playbook`* ***contains*** *`Plays`*
> *`Plays`* ***contains*** *`tasks`*
> *`Task`* ***runs*** *`modules`*.

Plays allow you to change a lengthy, complex set of manual administrative tasks into an easily repeatable routine with predictable and successful outcomes. In a playbook, you can save the sequence of tasks in a play into a human-readable and immediately runnable form.

- Ansible Playbook execution

```
ansible-playbook playbook.yml
```

- Syntax Verification

```
ansible-playbook --syntax-check main.yml
```

- Executing a Dry Run

```
ansible-playbook 1b_dry_run.yml --check
```

Executing Ansible Playbooks

> View the .yml files and execute with ansible-playbook command

- Magic Variables
    - `hostvars`
    - Contains the variables for managed hosts, and can be used to get the values for another managed host's variables. It won't include the managed host's facts if they haven't been gathered yet for that host.
    - `group_names`

- Lists all groups the current managed host is in.
- `groups`
- Lists all groups and hosts in the inventory.
- `inventory_hostname`
- Contains the hostname for the current managed host as configured in the inventory. This may be different from the hostname reported by facts for various reasons.
- To view more magic variables that are associated with a particular hosts, use below command:

```
ansible localhost -m debug -a 'var=hostvars[inventory_hostname]'
```