**WHAT CAN YOU DO WITH ANSIBLE**

* Configuration management
* Application deployment
* Provisioning (make availible)
* Ochestration (automation)
* Security
* Continuous delivery

**ANSIBLE USES SSH**

Ansible depends on SSH access to the servers you are managing. There is no client software needed on host your want to manage.

**INSTALL macOS**

The control node.

brew install ansible

Version,

ansible --version

**INSTALL UBUNTU**

sudo apt-get install software-properties-common

sudo apt-add-repository ppa:ansible/ansible

sudo apt-get update

sudo apt-get install ansible

**SETUP ON MAC - INVENTORY FILE**

Ansible uses an inventory file to determine what hosts to work against.

Make an inventory file of your hosts (use hostname or ip),

nano /etc/ansible/hosts,

p-stack-to-graph-ksf5

p-stack-to-graph-ksf5:2222

Tell ansible where you host file is,

nano ~/.ansible.cfg,

[defaults]

inventory = /etc/ansible/hosts

Your public SSH key should be located in authorized\_keys on those systems.

There are many more things you can do with a inventory file.

**COMMANDS**

All commands have the same format,

ansible server\_or\_group -m module\_name -a arguments

If you didn't have an inventory file you would have to do,

ansible all -i p-stack-to-graph-ksf5, -m ping

* all look at all the host
* -m --module-name like ping
* -i what is the inventory path

But since we have an inventory file (the hosts), the following works fine,

ansible all --module-name ping

ansible all -m ping

**MODULES**

Ansible’s way of abstracting certain system management or configuration tasks.

Control things you automate.

There are over 450 modules here

**AD HOC COMMANDS**

Use the module commands to send a command to the host,

ansible all --module-name command --args "uptime"

ansible all -m command -a "uptime"

ansible all -m command -a "/bin/date"

**PLAYBOOK**

Allow you to organize your configuration and management tasks in simple, human-readable files.

Playbooks can be combined with other playbooks and organized into Roles which allow you to define sophisticated infrastructures and then easily provision and manage them.

* Playbooks contain plays
* Plays contain Tasks
* Tasks call modules
* Tasks run sequentially
* Handlers are triggered by taks and are run once, at the end of the plays.

A playbook template looks like,

---

- hosts: [target hosts]

remote\_user: [yourname]

tasks:

- [task 1]

- [task 2]

Lets create a task,

nano test.yml

---

- hosts: all

tasks:

- name: Ensure git is installed

apt: name=git state=installed

sudo: yes

- name: Install cheat sheets repo

git: repo=https://github.com/JeffDeCola/my-cheat-sheets.git

dest=~/my-cheats

remote\_user: jeffdecola

sudo: false

- name: Copy file

command: cp ~/my-cheats/ansible-cheat-sheet/README.md ~/copied-this-README.MD

remote\_user: jeffdecola

sudo: false

Run,

ansible-playbook test.yml

**ROLES**

Roles are special kind of playbook.

Folders are as follows:

* defaults *The default variables (lower priority variables).*
  + main.yml
* files *If need to add files.*
* handlers *Targets for notify directives, and are almost always associated with services.*
  + main.yml
* meta *Meta data.*
  + main.yml
* tasks *Series of Ansible plays to install, configure, and run software.*
  + main.yml
* templates *Similar to files except that templates support modification.*
* vars *The default variables (higher priority variables).*
  + main.yml