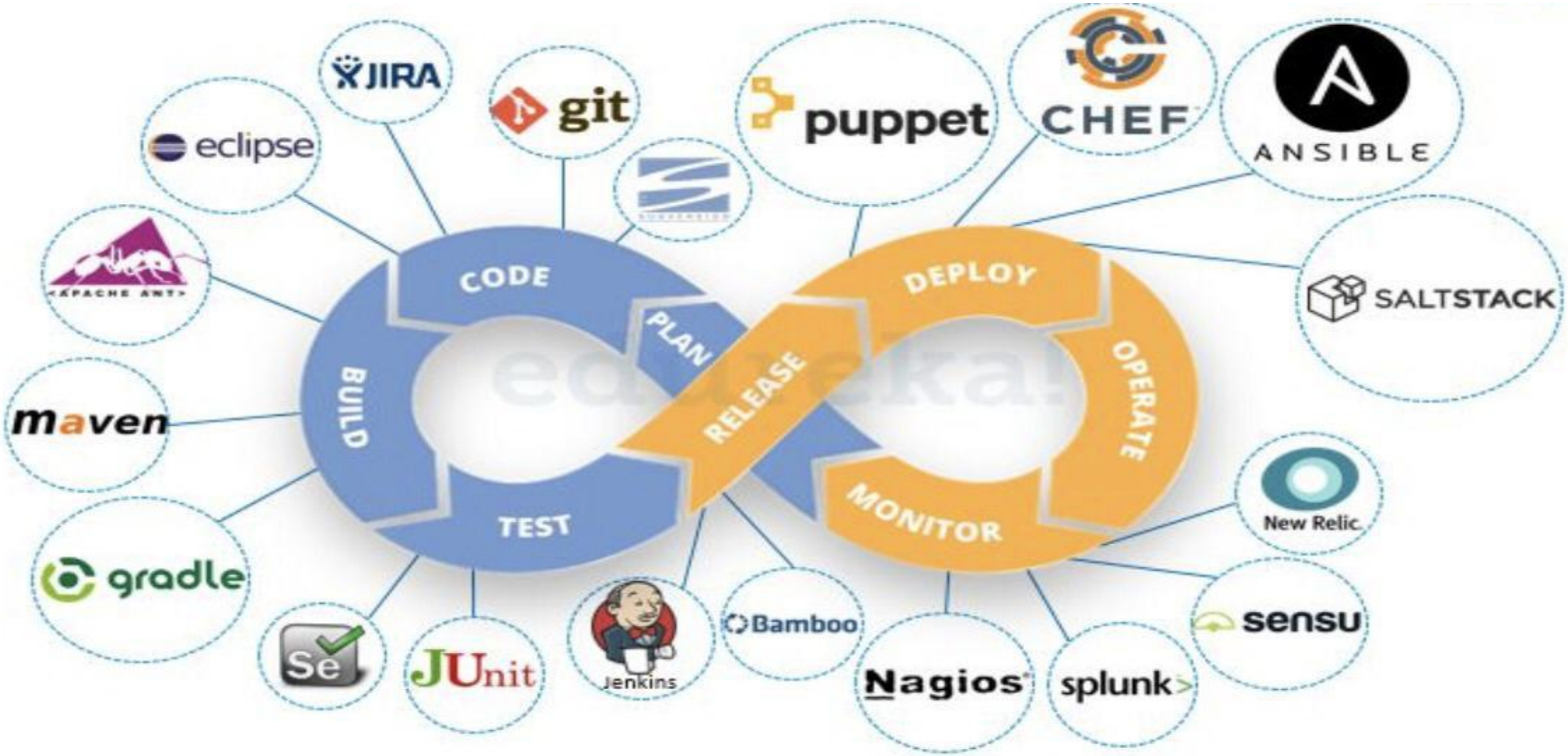
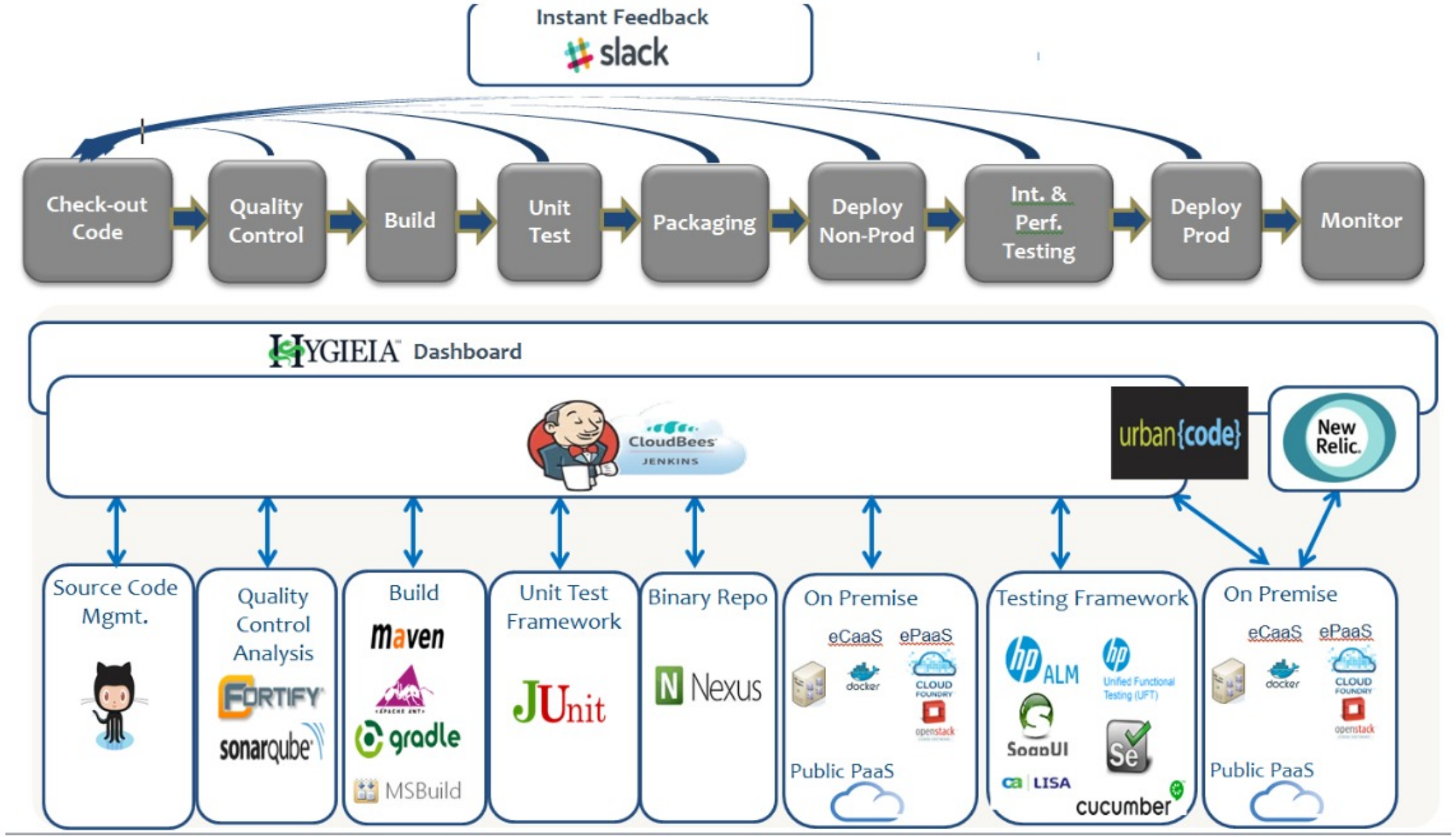
Introduction to Infrastructure Automation and Ansible - Mount Everest Consulting

CI/CD Pipeline Example - A



CI/CD Pipeline Example - B



Introduction to Ansible

- Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy.

- It support configuration management with examples as below.
    - Configuration of servers
    - Application deployment
    - Continuous testing of already install application
    - Provisioning
    - Orchestration
    - Automation of tasks

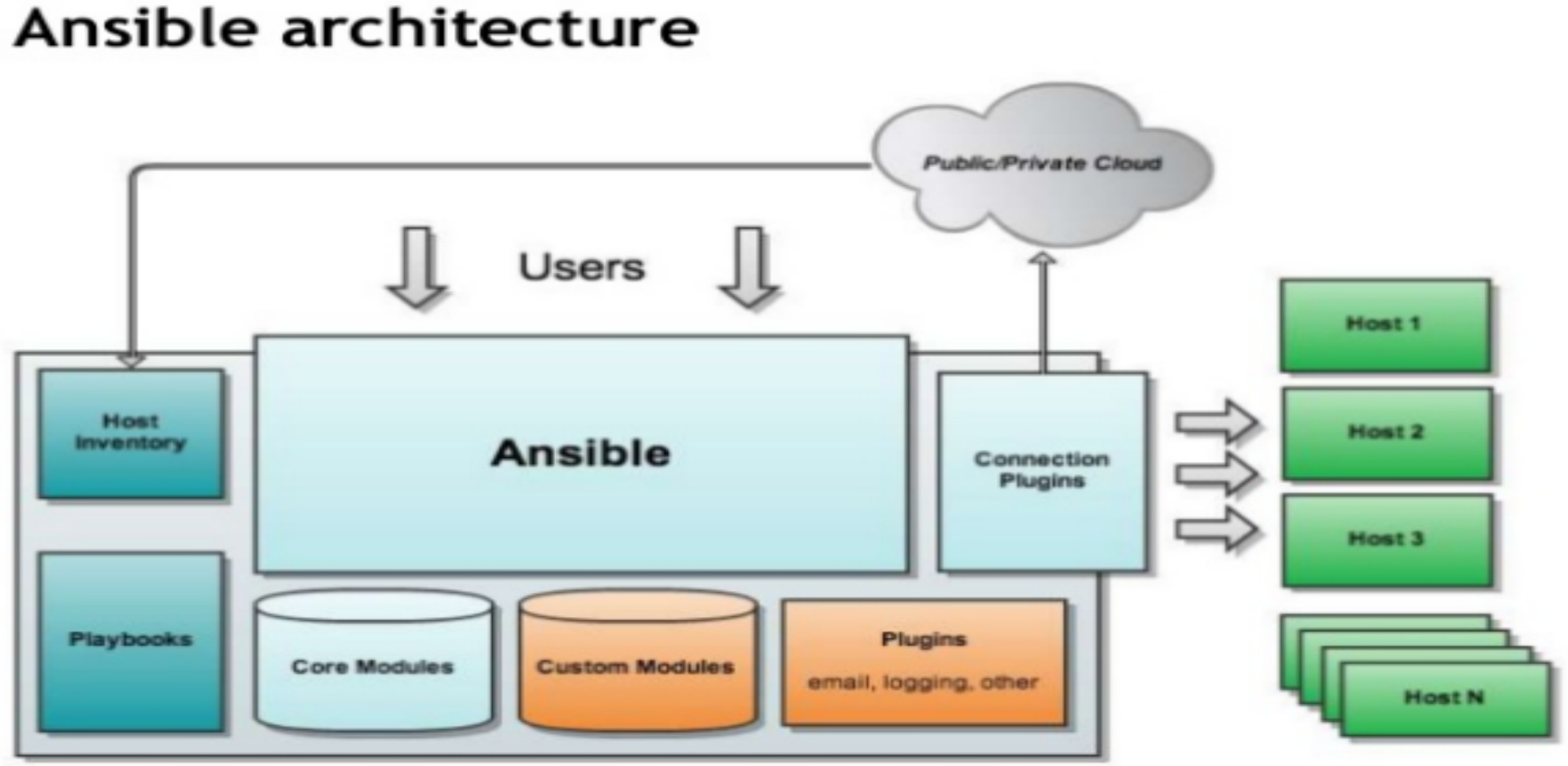
Why Automation?

- Tasks in code
- Collaboration
- Eliminate errors
- Write once
- Laziness
- Large scale provisioning
- Maintenance

Why Ansible

- It is a free open source application
- Agent-less – No need for agent installation and management
- Python/yaml based
- Highly flexible and configuration management of systems.
- Large number of ready to use modules for system management
- Custom modules can be added if needed
- Configuration roll-back in case of error
- Simple and human readable
- Self documenting

Ansible Architecture



Installation of Ansible

- Install packages below on the Server Machine
- sudo apt-get install python-yaml python-jinja2 python-paramiko python-crypto python-keyczar ansible

- Install packages below on the client Machines
- sudo apt-get install python-crypto python-keyczar

Create the RSA Key Pair

- The first step is to create the key pair on the Server machine
- ssh-keygen –t rsa

- Once you have entered the Gen Key command, you will get a few more questions:
- Enter file in which to save the key (/home/test/.ssh/id_rsa):
- Enter no password for the next prompt
- Copy the Public Key
- ssh-copy-id test@192.168.85.135
- Repeat the same process for other machines you wish to login automatically with.
- Ensure the test username has sudo access to the remote clients

Configuration of ansible

- Do the following on the Server machine
- Create the list of client machines you wish to access via this server
- vi /etc/ansible/hosts (And enter the following lines and save file)
- [Servers]
- 192.168.85.135
- 192.168.85.136

- Run the ping command below to see if indeed you are reaching both client nodes
- ansible -m ping all

Examples of ansible commands

- The output show ping result success as shown below

```
test@devx4:~$ ansible -m ping all
192.168.85.135 | success >> {
    "changed": false,
    "ping": "pong"
}

192.168.85.136 | success >> {
    "changed": false,
    "ping": "pong"
}
```

Examples of ansible commands (Cnt)

- How to run commands to fetch hard drives utilization
- ansible -m command -a 'df -h' Servers

- How to run commands to fetch system uptime
- ansible -m command -a ‘uptime' Servers

```
test@devx4:~$ ansible -m command -a 'uptime' Servers
192.168.85.136 | success | rc=0 >>
 10:36:02 up 33 min,  2 users,  load average: 0.00, 0.01, 0.05

192.168.85.135 | success | rc=0 >>
 10:36:01 up 33 min,  2 users,  load average: 0.00, 0.01, 0.03
```

Examples of ansible commands (Cnt)

- The full configuration environment inventory of a particular client machine can be obtain using the command below.
- ansible -m setup 192.168.85.135 (output as shown below)

```
test@devx4:~$ ansible -m setup 192.168.85.135
192.168.85.135 | success >> {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.85.135"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::20c:29ff:fe40:c86a"
        ],
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "07/02/2015",
        "ansible_bios_version": "6.00",
```

Creating an ansible-playbook template

- Create a template to enable the installation of an NTP service with content as shown below and file saved as ntp.yml

```
test@devx4:~$ cat ntp.yml
---
- hosts: Servers
  tasks:
  - name: ensure ntp packages is installed
    action: apt pkg=ntp
    sudo: yes
  - name: copy ntp configuration file
    action: copy src=/etc/ansible/ntp.conf dest=/etc/ntp.conf
            owner=root group=root mode=0644
    sudo: yes
  - name: ensure ntp service is restarted
    action: service name=ntp state=restarted
    sudo: yes
```

Understanding ansible playbook configurations

- In order to use ansible with SSH passwords you will need to install the program below
- sudo apt-get install sshpass
- Ansible-playbook command can be executed to run the ntp.yml file as below
- ansible-playbook -k -K ntp.yml
- The –k –K switches allow you to be able to use your ssh key and passwordless sudo.
- Every playbook configuration begins with triple dash (----)
- The hosts, tasks, name, action are various instructions commands to help automate your ntp installation process.

Understanding ansible playbook configurations (cnt)

- The output of the ansible-playbook command as below

```
test@devx4:~$ ansible-playbook -k -K ntp.yml
SSH password:
sudo password [defaults to SSH password]:

PLAY [Servers] ***************************************************************

GATHERING FACTS ***************************************************************
ok: [192.168.85.140]
ok: [192.168.85.142]

TASK: [ensure ntp packages is installed] **************************************
changed: [192.168.85.142]
changed: [192.168.85.140]

TASK: [copy ntp configuration file] *******************************************
changed: [192.168.85.140]
changed: [192.168.85.142]

TASK: [ensure ntp service is restarted] ***************************************
changed: [192.168.85.140]
changed: [192.168.85.142]

PLAY RECAP ********************************************************************
192.168.85.140             : ok=4    changed=3    unreachable=0    failed=0
192.168.85.142             : ok=4    changed=3    unreachable=0    failed=0
```

Ansible Documentations

- You can find more explanation in the Ansible Docs.
  - Ad-hoc commands
  - Inventories
  - Variables
  - Modules
  - Playbook Roles

- Similar tools that does the same function as Ansible are as below.
  - Puppet
  - Chef
  - Salt