NetApp

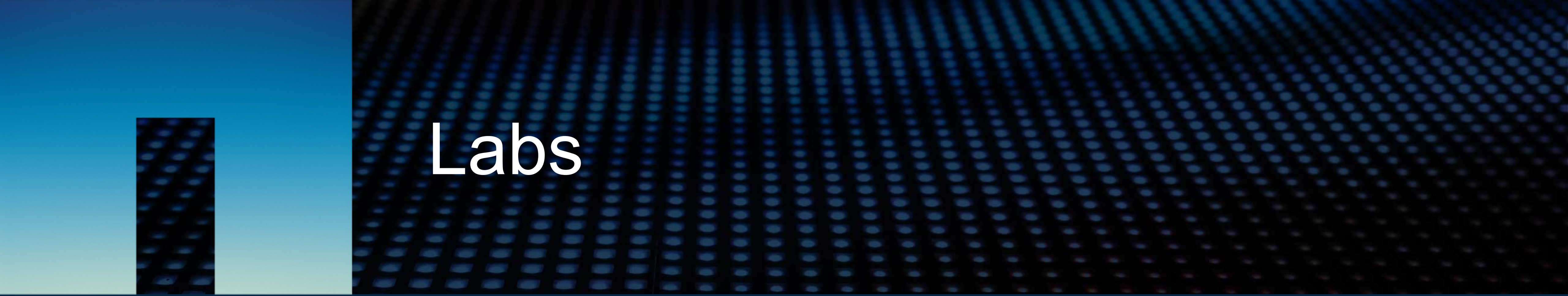
Ansible NetApp Automation Workshop

Introduction to Ansible NetApp Automation for Storage Administrators and Operators

Housekeeping

- Timing
- Breaks
- Takeaways
- Lab

NetApp



Labs

- https://lod-bootcamp.netapp.com
- Password: **Treble58**
- https://github.com/schmots1/ansible_workshop/
- Exercise Guide

NetApp

What you will learn

- Introduction to Ansible Automation
- How it works
- Ad-hoc commands and Inventories
- Understanding modules, tasks & playbooks
- Working with Playbooks
  - Using variables
  - Conditionals and Loops
  - Templates
  - Roles
- Tower
  - About
  - Inventories and Credentials
  - Projects and Job Templates
  - Surveys
  - RBAC
  - Workflows

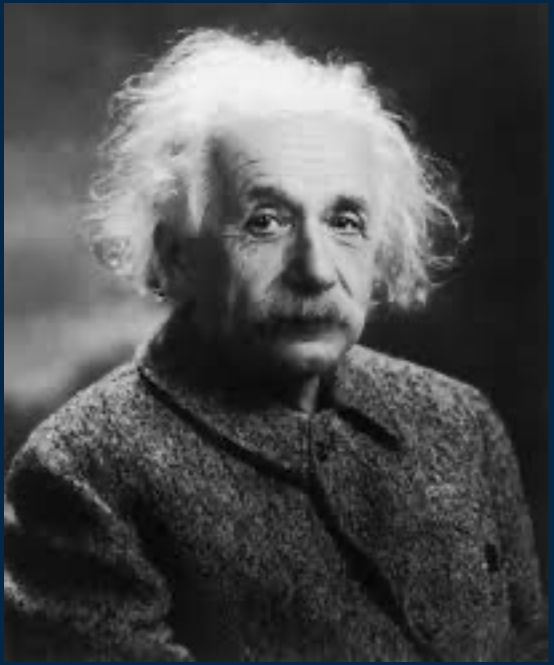
NetApp

Introduction

- What Ansible Automation is
- What it can do

“Insanity is doing the same thing over and over again and expecting different results”

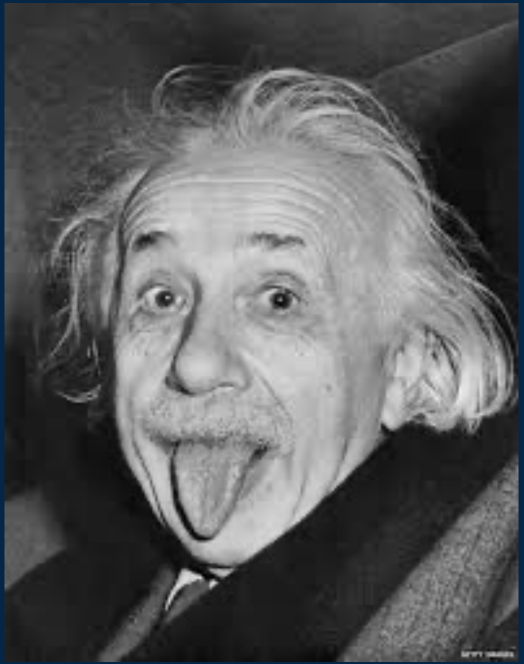
Albert Einstein



NetApp

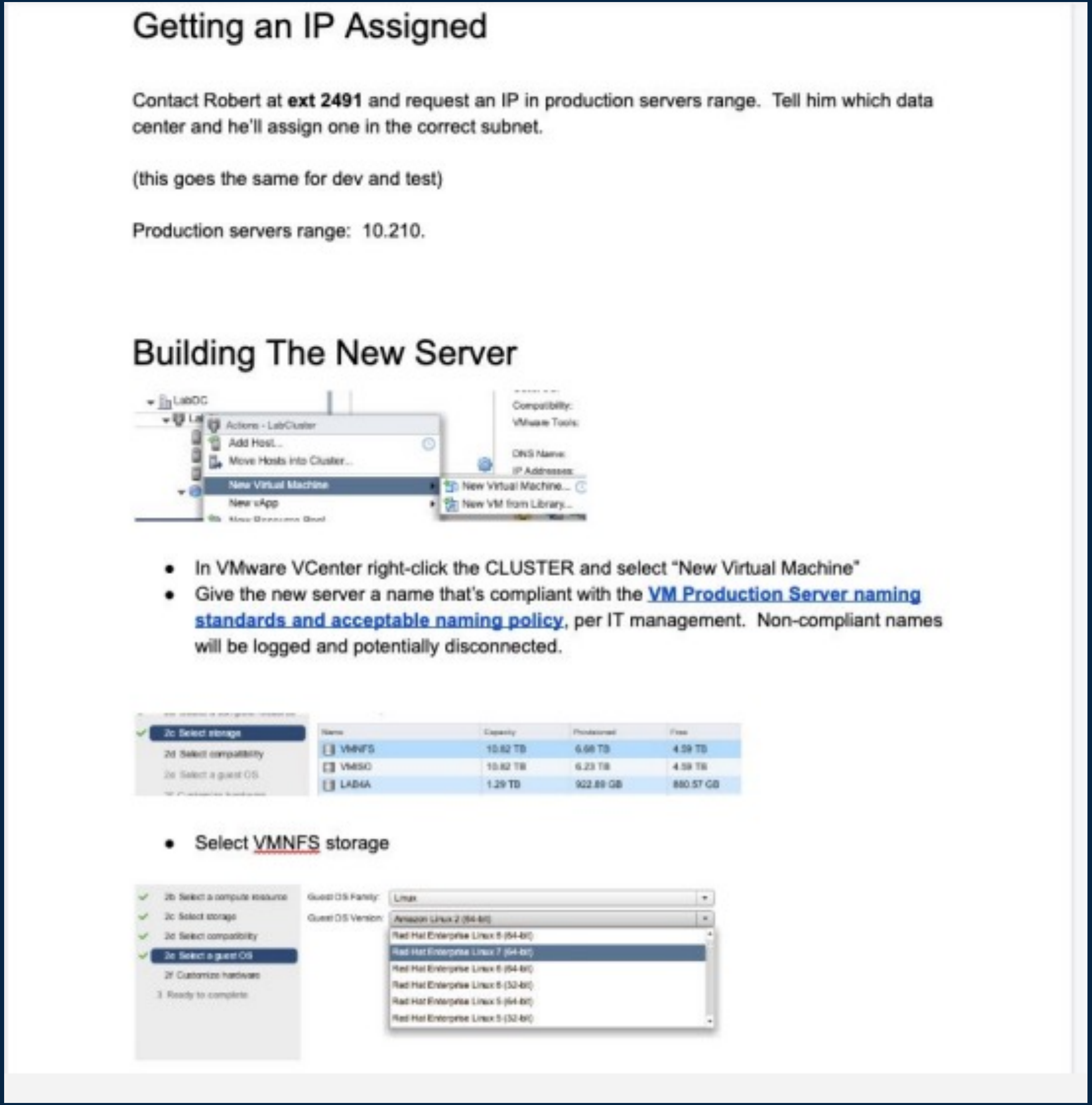
“Insanity is doing the same thing over and over again manually when you could have automated it with Ansible”

probably not Albert Einstein



Ansible Automation

- Repeatable processes
  - Usually manual steps to assure end solution



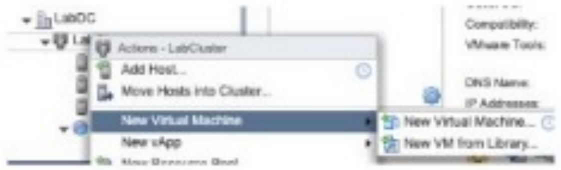
Getting an IP Assigned

Contact Robert at **ext 2491** and request an IP in production servers range. Tell him which data center and he'll assign one in the correct subnet.

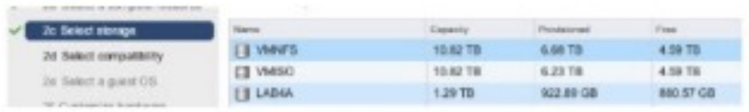
(this goes the same for dev and test)

Production servers range: 10.210.

Building The New Server

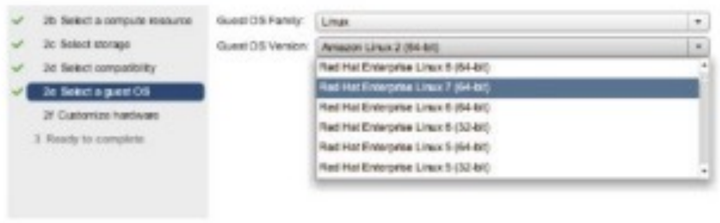


- In VMware VCenter right-click the CLUSTER and select "New Virtual Machine"
- Give the new server a name that's compliant with the VM Production Server naming standards and acceptable naming policy, per IT management. Non-compliant names will be logged and potentially disconnected.



Name	Capacity	Provisioned	Free
VMNFS	10.82 TB	6.68 TB	4.59 TB
VMISO	10.82 TB	6.23 TB	4.59 TB
LAB4A	1.29 TB	922.89 GB	880.57 GB

- Select VMNFS storage



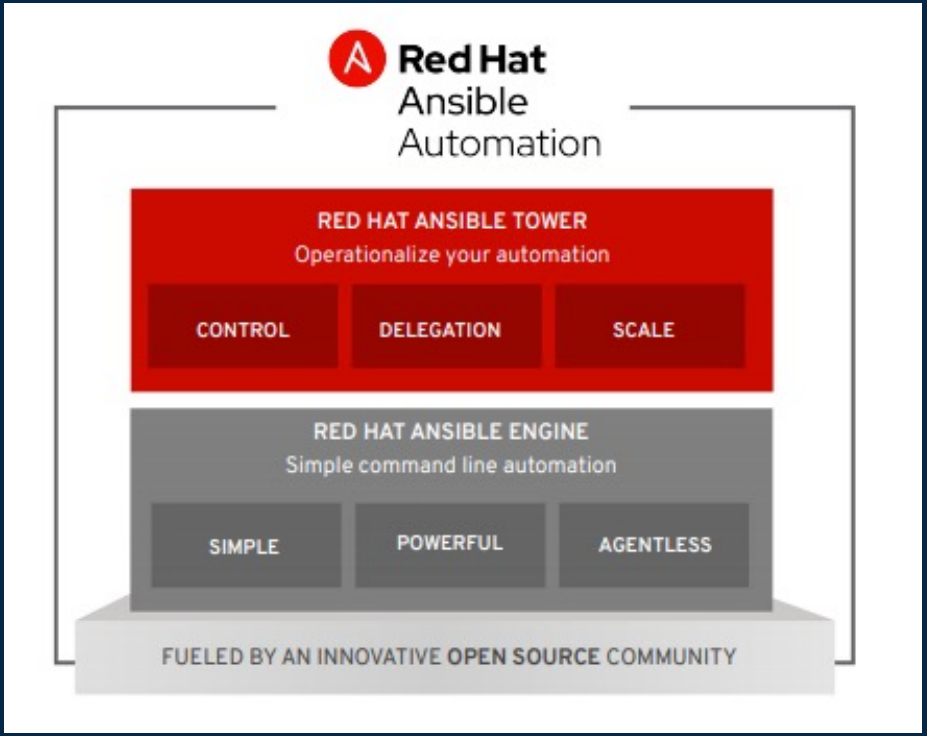
Guest OS Family: Linux
Guest OS Version: Amazon Linux 2 (64-bit)

Red Hat Enterprise Linux 8 (64-bit)
Red Hat Enterprise Linux 7 (64-bit)
Red Hat Enterprise Linux 6 (64-bit)
Red Hat Enterprise Linux 6 (32-bit)
Red Hat Enterprise Linux 5 (64-bit)
Red Hat Enterprise Linux 5 (32-bit)

Ansible Automation

What is Ansible automation?

- Ansible Automation is the enterprise **framework** for automating across IT operations
- Ansible Engine runs Ansible Playbooks, the automation **language** to describe an IT application infrastructure
- Ansible Tower allows you to **scale** IT automation, manage complex deployments and speed productivity.


Red Hat Ansible Automation
RED HAT ANSIBLE TOWER
Operationalize your automation
CONTROL
DELEGATION
SCALE
RED HAT ANSIBLE ENGINE
Simple command line automation
SIMPLE
POWERFUL
AGENTLESS
FUELED BY AN INNOVATIVE OPEN SOURCE COMMUNITY

Ansible Automation

- Why Ansible?
  - Simple
    - YAML
    - No Coding
  - Procedural
    - Tasks are executed in order
    - Playbooks are procedural, modules are declarative
  - Idempotent
    - Ability to run over and over, without error or duplicates
- Agentless
  - Nothing to load or update
  - Get started right away
- EcoSystem
  - Usable by every team

NetApp

Ansible Automation

- What can I Automate using Ansible?
  - Orchestration
  - Configuration Management
  - Application Deployment
  - Provisioning
  - Continuous Delivery
  - Security and Compliance

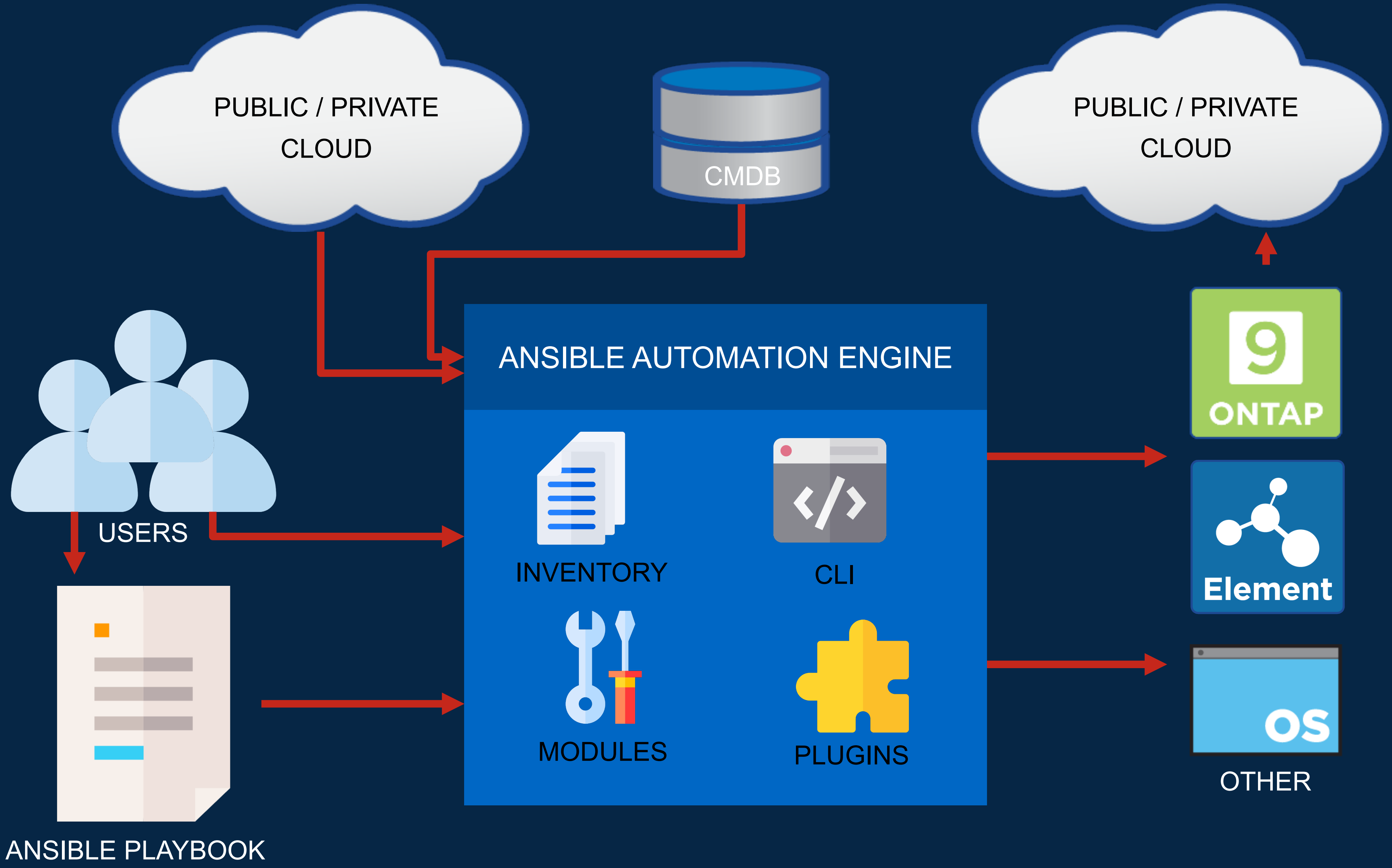
Firewalls	Load Balancers
Applications	Containers
Clouds	Servers
Infrastructure	Storage
Network Devices	And More

NetApp

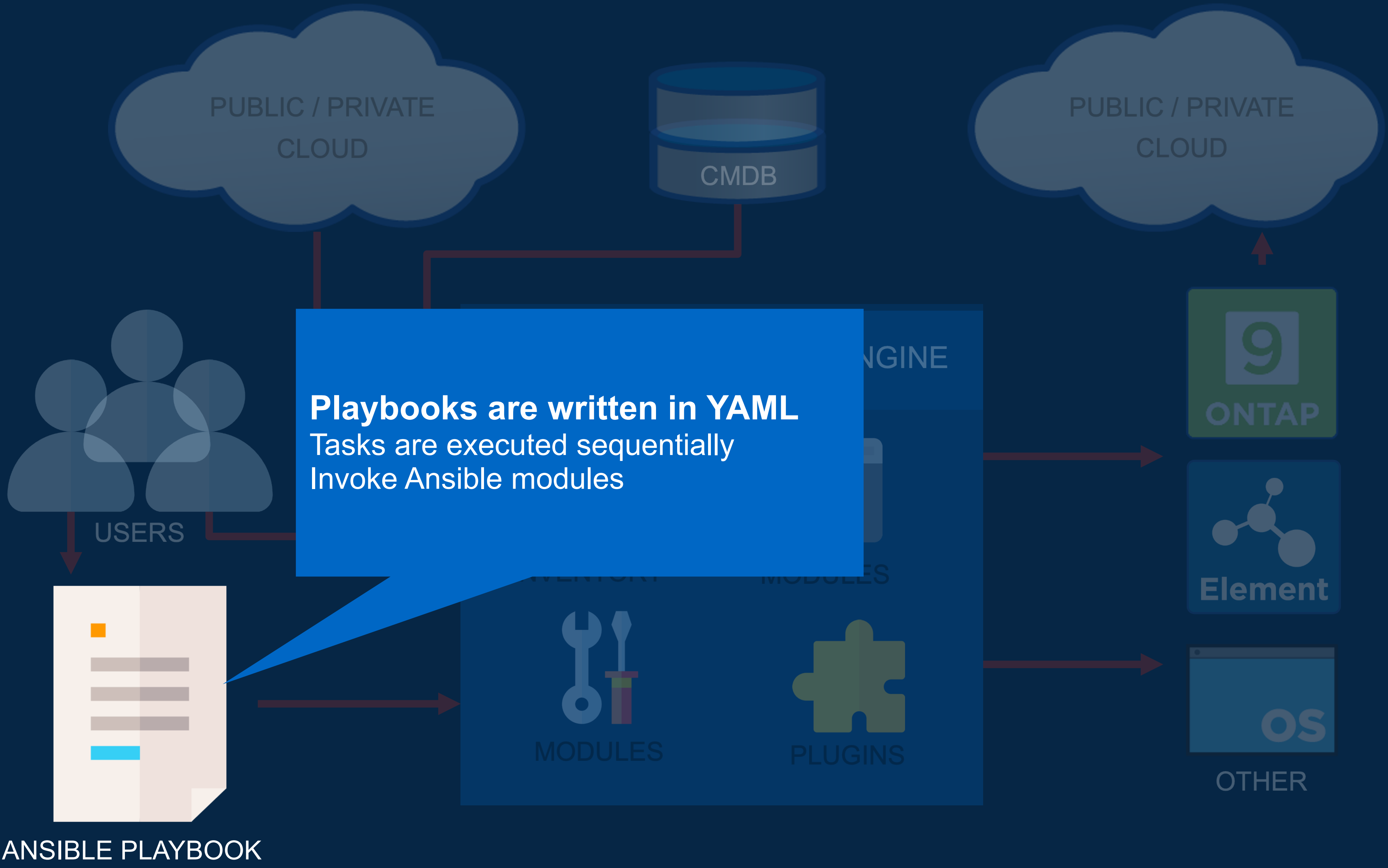
1.1 Basics

Understanding the Ansible Infrastructure

Installing Ansible

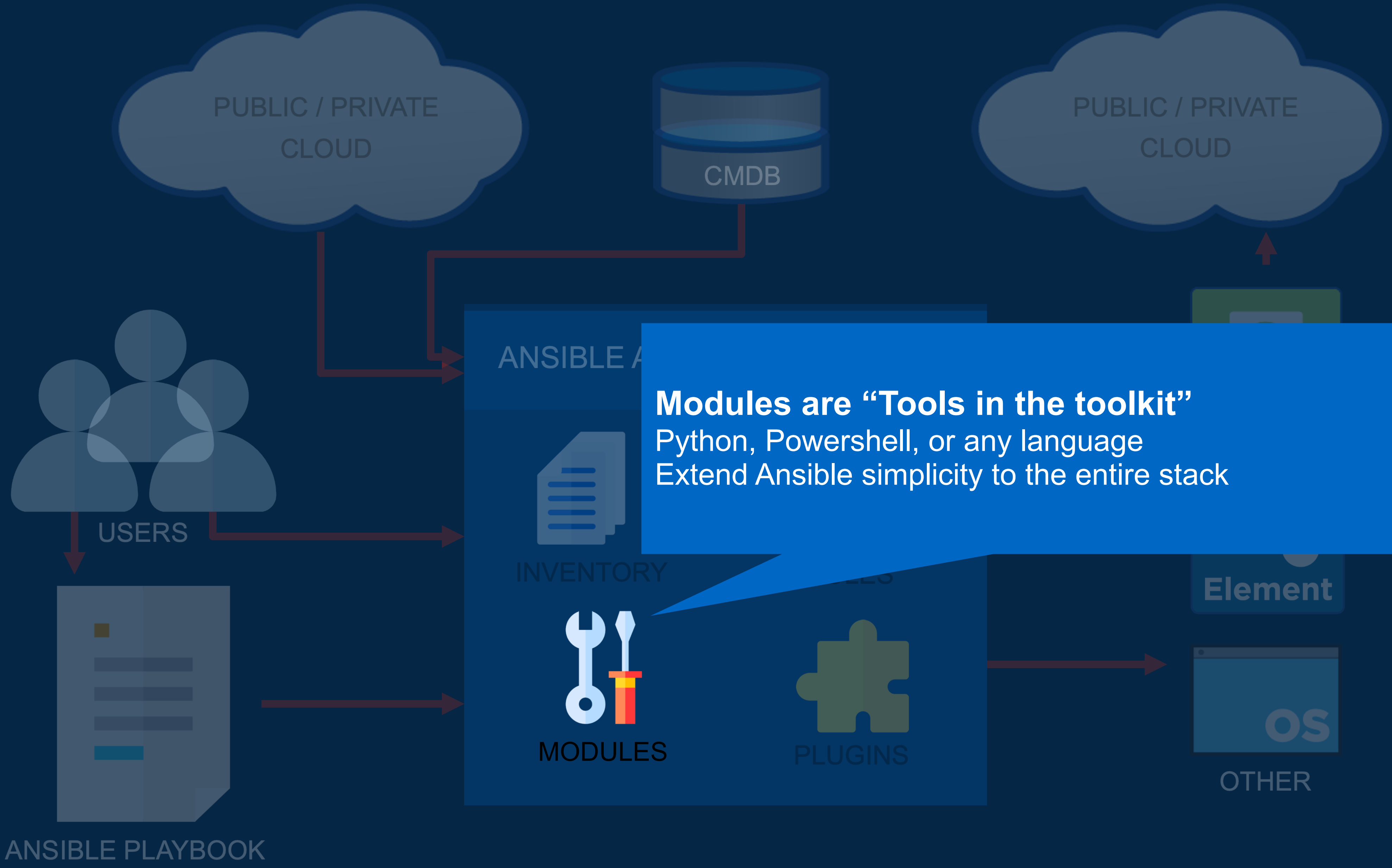


Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf



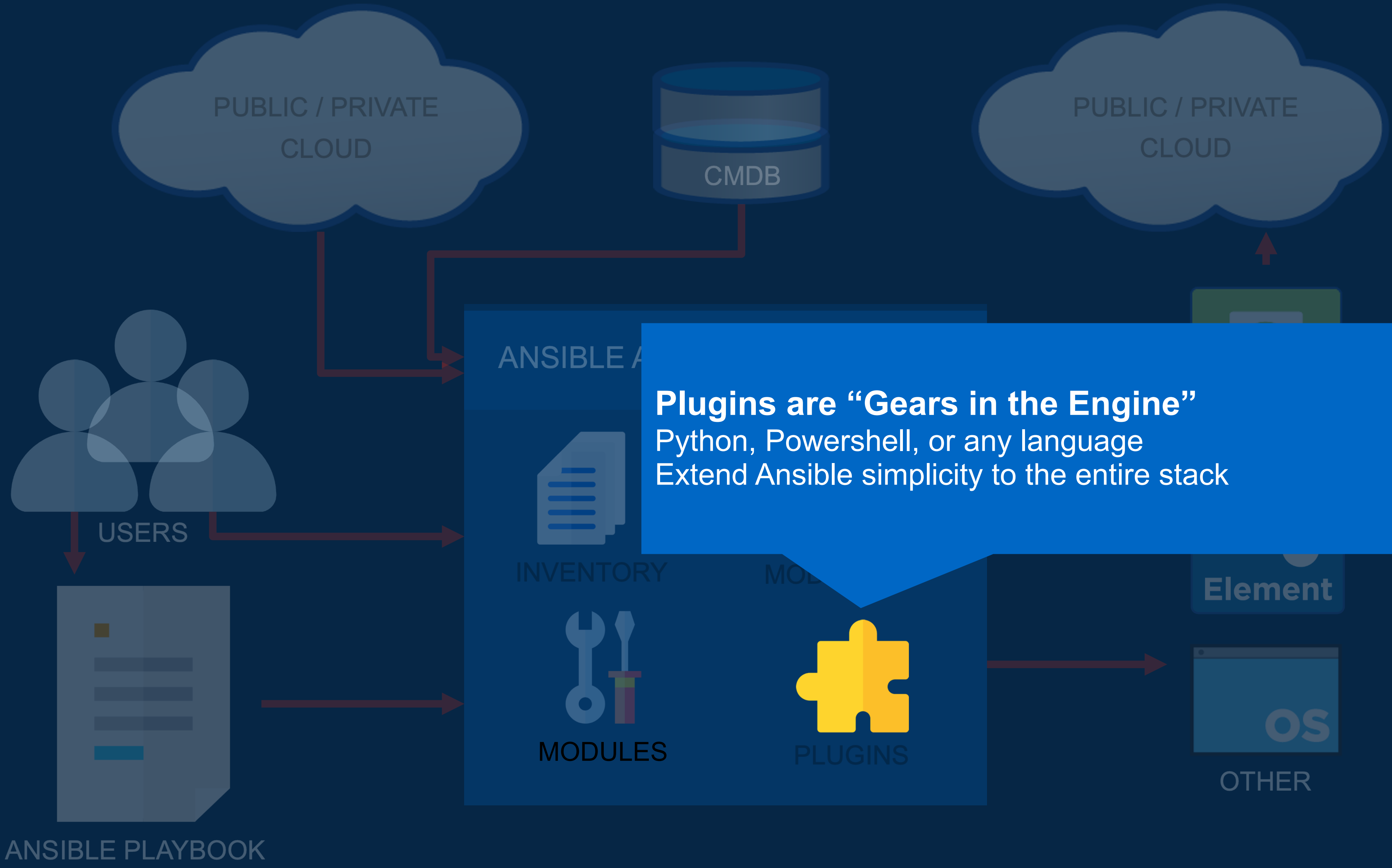
Playbooks are written in YAML
Tasks are executed sequentially
Invoke Ansible modules

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf



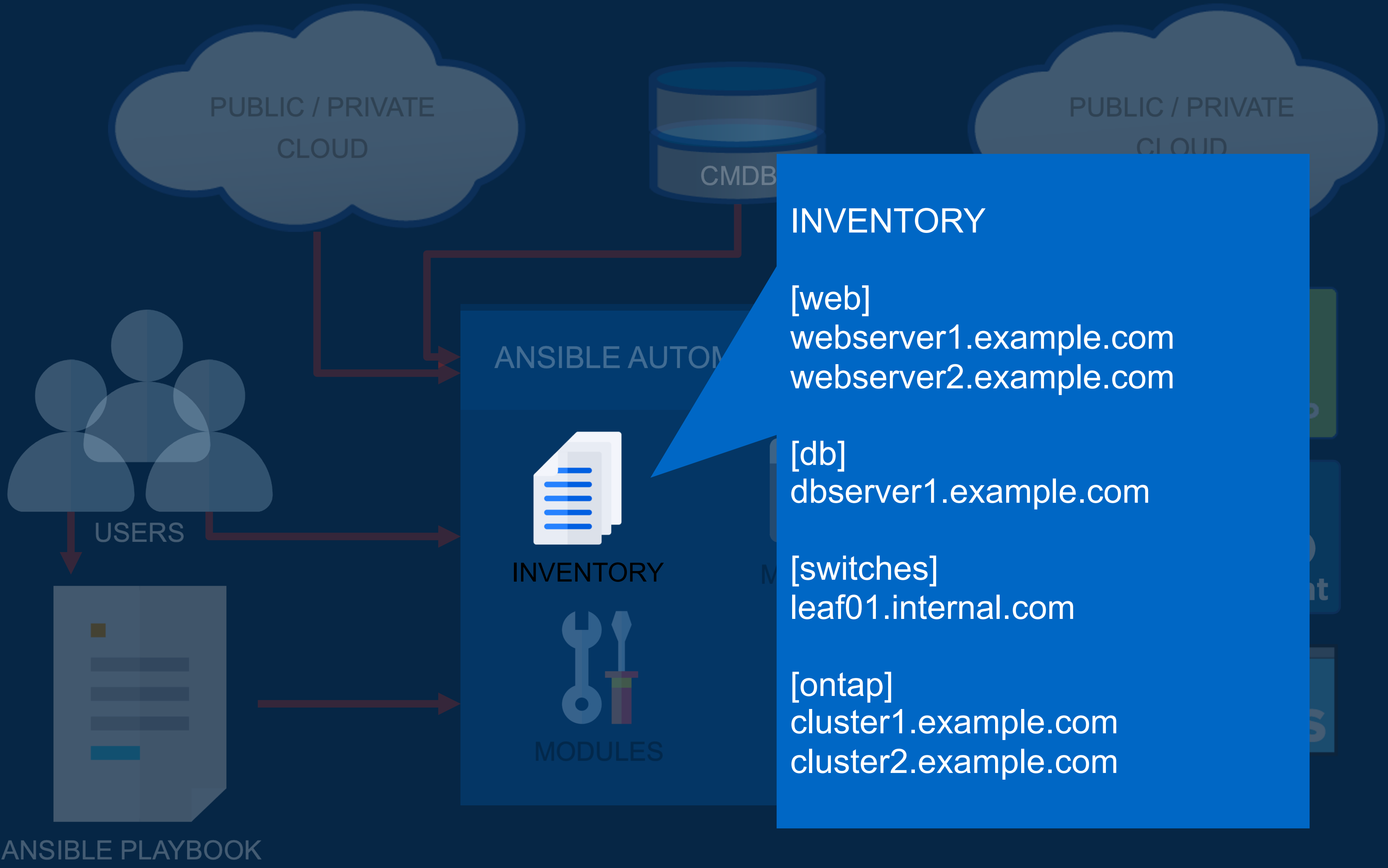
Modules are "Tools in the toolkit"
Python, Powershell, or any language
Extend Ansible simplicity to the entire stack

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf



Plugins are “Gears in the Engine”
Python, Powershell, or any language
Extend Ansible simplicity to the entire stack

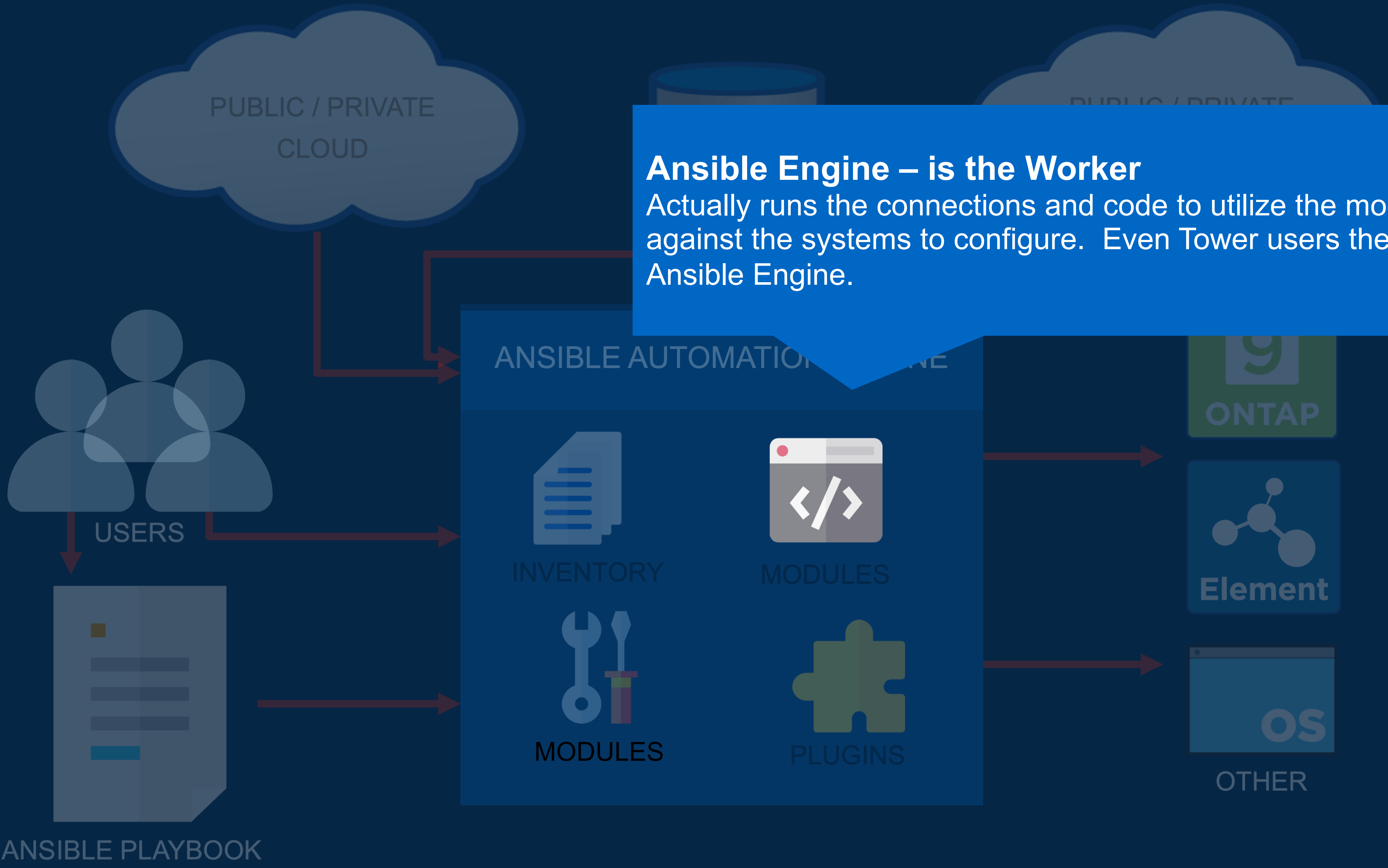
Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf



INVENTORY

```
[web]
webserver1.example.com
webserver2.example.com

[db]
dbserver1.example.com

[switches]
leaf01.internal.com

[ontap]
cluster1.example.com
cluster2.example.com
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

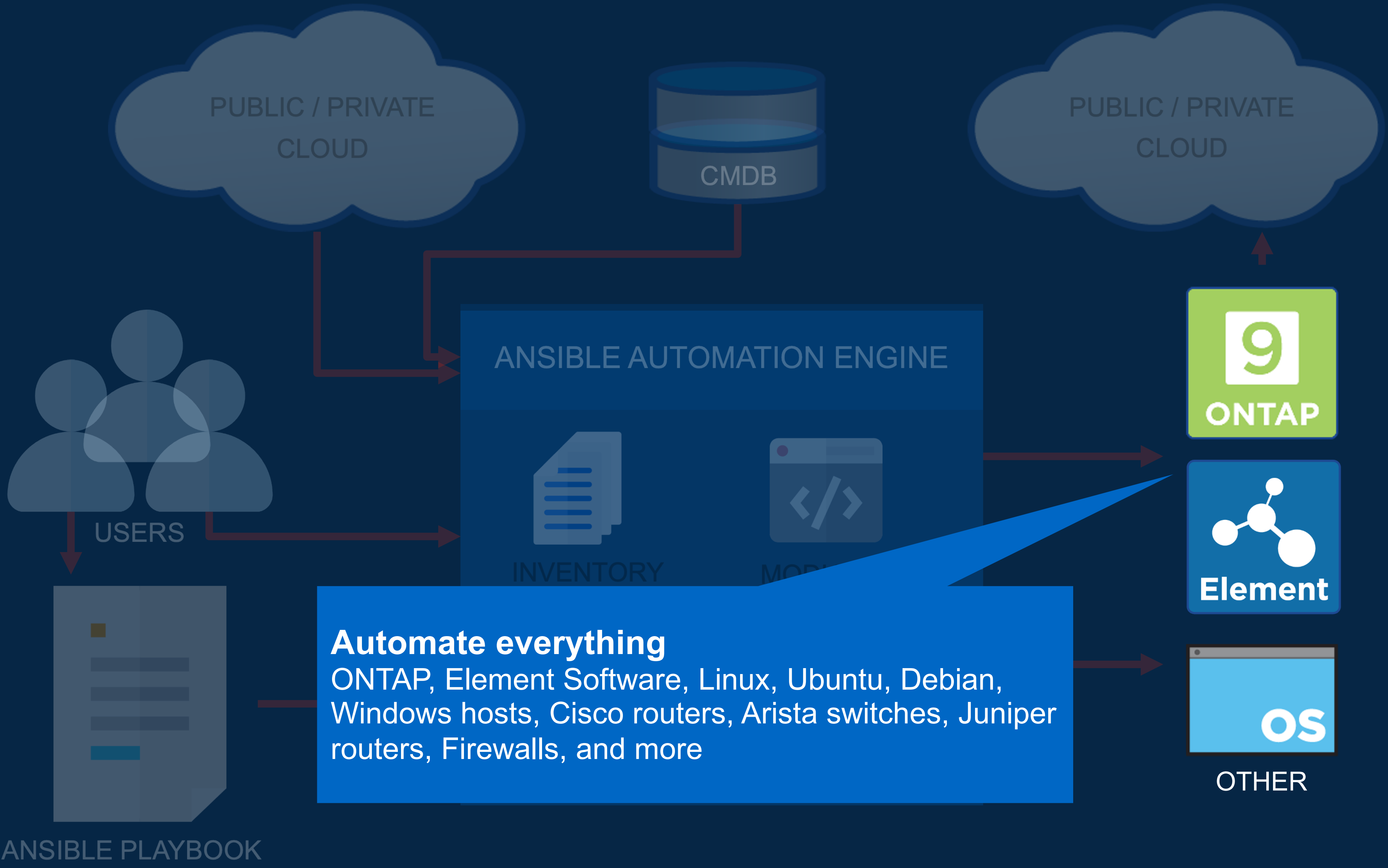


Ansible Engine – is the Worker

Actually runs the connections and code to utilize the modules against the systems to configure. Even Tower users the Ansible Engine.

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

NetApp



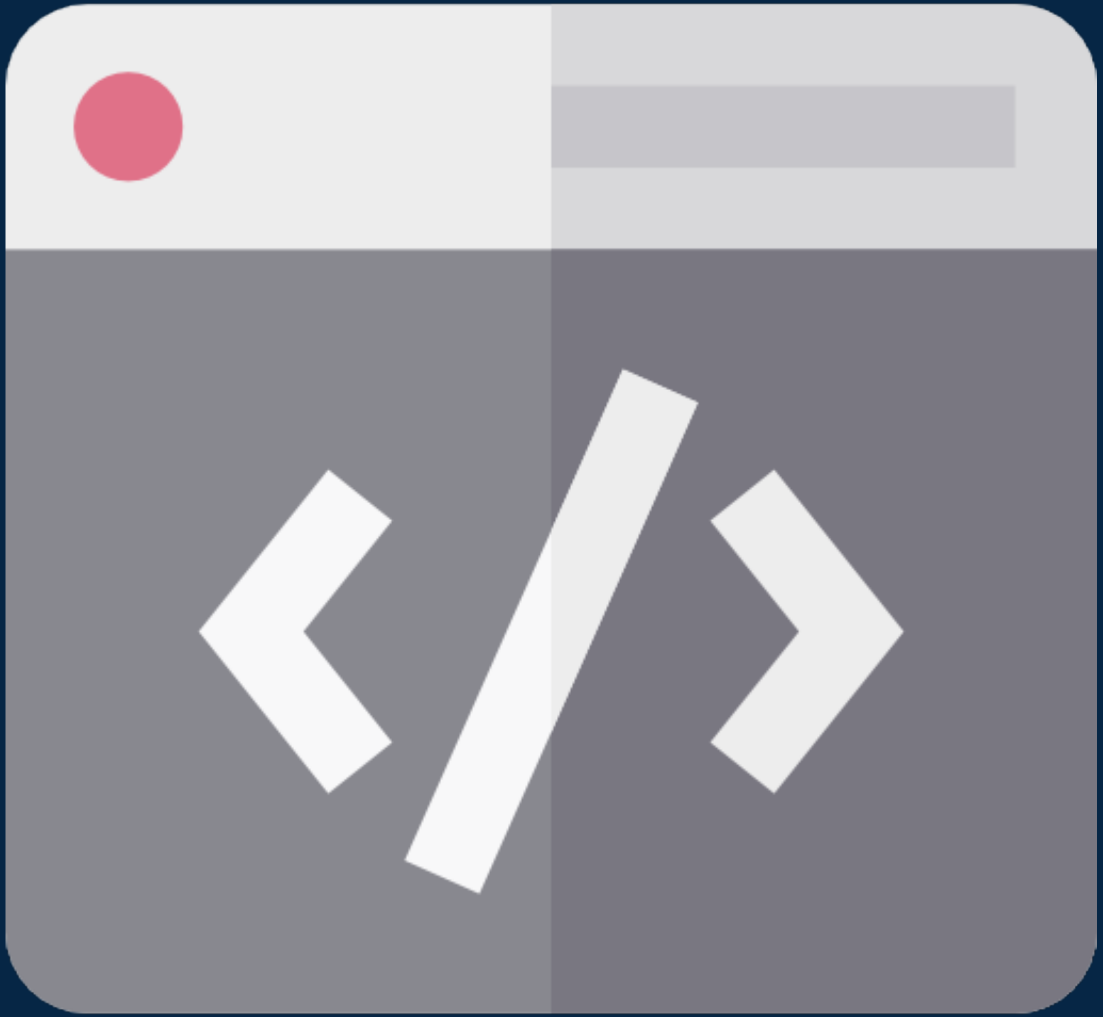
Automate everything
ONTAP, Element Software, Linux, Ubuntu, Debian, Windows hosts, Cisco routers, Arista switches, Juniper routers, Firewalls, and more

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Automation

How Ansible ONTAP Automation Works

- Code is run via ZAPI or REST API
  - <= 9.5 Defaults to ZAPI
  - >= 9.6 Defaults to REST API (can be overwritten)



1.2 Running Commands

Ansible inventories

Ansible config file

Modules and ad-hoc commands

Understanding Inventories

- Ansible works against multiple systems in an inventory
- Inventory is usually file based
- Can have multiple groups
- Can have variables for each group or even host

- **NETAPP MODULES RUN AGAINST LOCALHOST**

Understanding Inventories

Basic

```
# Static inventory example:
[myservers]
10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.100
host.example.com
```

NetApp

Understanding Inventories

Variables

```
[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[web]
Node-[1:30] ansible_host=10.42.0.[31:60]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=kev
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Understanding Inventories

Variable Precedence

Host variables apply to the host and override group vars

```
[app1srv]
appserver01 ansible_host=10.42.0.2 tmp_dir=/tempdir
appserver02 ansible_host=10.42.0.3 tmp_dir=/tmpwsdir

[web]
Node-[1:30] ansible_host=10.42.0.[31:60]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=kev
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa
```

Group variables apply for all devices in that group

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Understanding Inventories

Managing Variables in Files

```
[user@ansible ~]$ tree /somedir

/somedir
|--group_vars
|     --app1srv
|     --db
|     --web
|--inventory
|--host_vars
    --app01
    --app02
    --app03
```

```
[ user@ansible ~] cat /somedir/inventory

[web]
node-[1:30] ansible_host=10.42.0.[31:60]

[appxsrv]
app01
app02
app03
```

```
[ user@ansible ~] cat /somedir/group_vars/web

apache_listen_port: 8080
apache_root_path: /var/www/mywebdocs/
```

```
[ user@ansible ~] cat /somedir/host_vars/app01

owner_name: Chris P. Bacon
owner_contact: ‘cbacon@mydomain.tld’
server_purpose: Application X
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Understanding Inventories

Groups

```
[nashville]
bnaapp01
bnaapp02

[atlanta]
atlapp03
atlapp04

[south:children]
atlanta
nashville
hsvapp05
```

Some examples sourced from
https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Configuration File

- Basic configuration for Ansible
- Can be in multiple locations, with different precedence
- Here: `.ansible.cfg` in the home directory
- Configures where to find the inventory

Ansible Configuration File

Search order

- ANSIBLE_CONFIG (environment variable if set)
- ansible.cfg (in the current directory)
- ~/.ansible.cfg (in the home directory)
- /etc/ansible/ansible.cfg (installed as Ansible default

Ad-Hoc Commands

- Single Ansible command to perform a task quickly, directly on the command line
- Most basic operation that can be performed
- Here is an example using Ansible ping – which is not ICMP
- `$ ansible all -m ping`

Ad-Hoc Commands

ping

```
# Check connections (submarine ping, not ICMP)
[user@ansible]$ ansible all -m ping


web1 | SUCCESS => {
   “ansible_facts”: {
         “discovered_interpreter_python”: “/usr/bin/python”
    },
    “changed”: false,
    “ping”: “pong”
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ad-Hoc Commands

The Ansible Command –help (Display some basic and extensive options)

```
[user@ansible]$ ansible --help

usage: ansible [-h] [--version] [-v] [-b] [--become-method BECOME_METHOD][--become-user BECOME_USER] [-K] [-i INVENTORY] [--list-hosts]
[-l SUBSET] [-P POLL_INTERVAL] [-B SECONDS] [-o] [-t TREE] [-k] [--private-key PRIVATE_KEY_FILE] [-u REMOTE_USER] [-c CONNECTION] [-T
TIMEOUT] [--ssh-common-args SSH_COMMON_ARGS] [--sftp-extra-args SFTP_EXTRA_ARGS] [--scp-extra-args SCP_EXTRA_ARGS] [--ssh-extra-args
SSH_EXTRA_ARGS] [-C] [--syntax-check] [-D] [-e EXTRA_VARS] [--vault-id VAULT_IDS] [--ask-vault-pass | --vault-password-file
VAULT_PASSWORD_FILES] [-f FORKS] [-M MODULE_PATH] [--playbook-dir BASEDIR] [-a MODULE_ARGS] [-m MODULE_NAME] pattern

Define and run a single task 'playbook' against a set of hosts

positional arguments:

  pattern               host pattern

optional arguments:

  --ask-vault-pass      ask for vault password

  --list-hosts          outputs a list of matching hosts; does not execute anything else

  --playbook-dir BASEDIR

                        Since this tool does not use playbooks, use this as a substitute playbook directory.This sets the relative path
                        for many features including roles/ group_vars/ etc.
```

… and about another 100 lines

Ad-Hoc Commands

Useful common options

- -m MODULE_NAME, --module-name=MODULE_NAME
  Module name to execute the ad-hoc command
- -a MODULE_ARGS, --args=MODULE_ARGS
  Module arguments for the ad-hoc command
- -b, --become
  Run ad-hoc command with elevated rights such as sudo, the default method
- -e EXTRA_VARS, --extra-vars=EXTRA_VARS
  Set additional variables as key=value, @variable_file or YAML/JSON

Ad-Hoc Commands

Common use examples

```
# Check connections to all (submarine ping, not ICMP)
[user@ansible]$ ansible all -m ping


# Run a command on all the hosts in the web group
[user@ansible]$ ansible web -m command -a “uptime”


# Collect and display known facts for server “web1”
[user@ansible]$ ansible web1 -m setup
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Modules

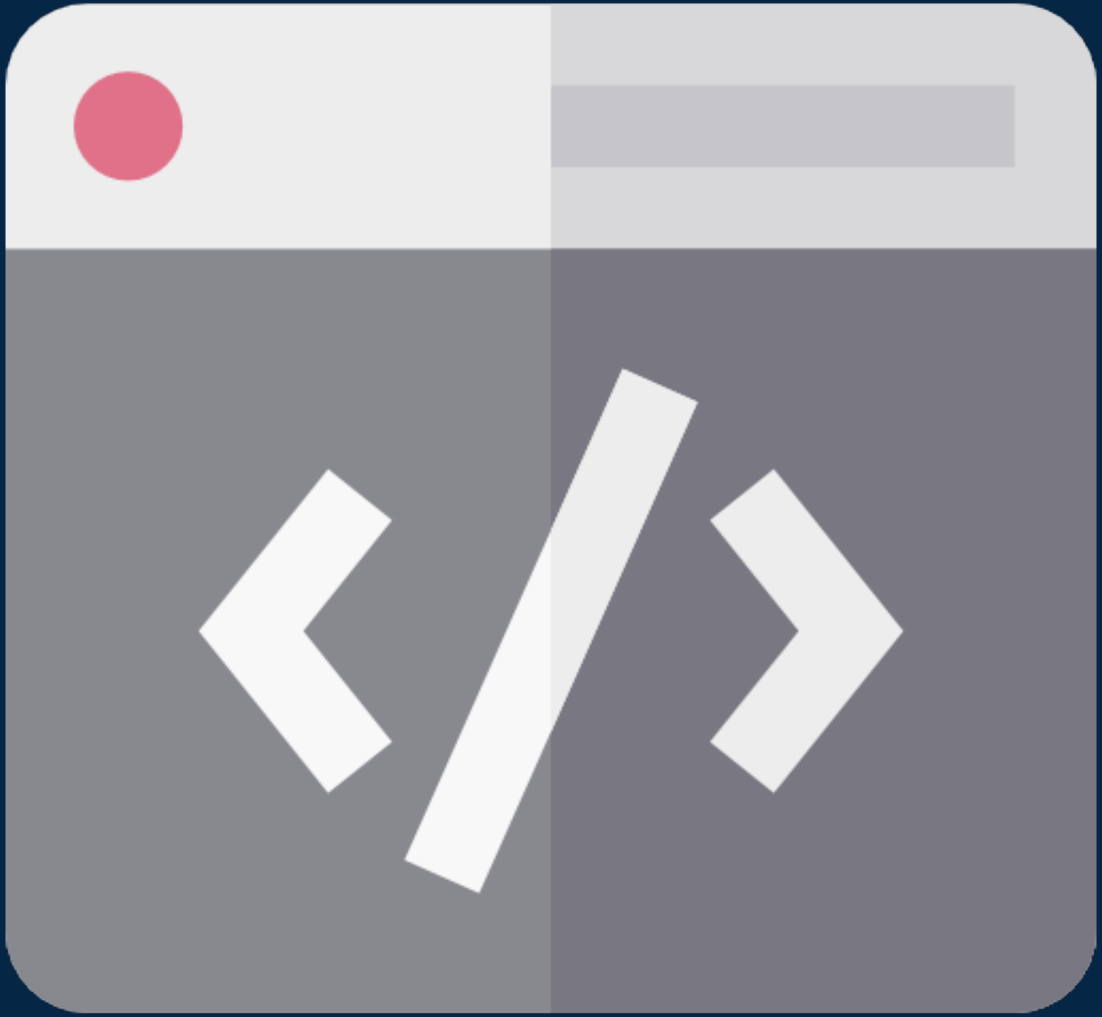
Using ansible-doc to read a modules documentation

```
[user@ansible]$ ansible-doc netapp.ontap.na_ontap_volume

> NA_ONTAP_VOLUME    (/usr/lib/python2.7/site-packages/ansible/modules/storage/netapp/na_ontap_volume.py)
        Create or destroy or modify volumes on NetApp ONTAP.

  * This module is maintained by an Ansible Partner
OPTIONS (= is mandatory):

- aggr_list

        an array of names of aggregates to be used for FlexGroup constituents.
        [Default: (null)]
        type: list
        version_added: 2.8

- aggr_list_multiplier

        The number of times to iterate over the aggregates listed with the aggr_list parameter when creating a FlexGroup.
        [Default: (null)]
        type: int
        version_added: 2.8
```

Ansible Modules

“I can’t find a module that does what I need it to do!”

- na_ontap_command (pre9.6)
- na_ontap_rest_cli
- Command
- Shell
- raw



1.3 Playbooks

Playbook basics

Running a playbook

Ansible Playbooks

Ansible Plays

A play

Another play

```
---
- hosts: db
  vars:
    software:
      - mariadb-server
  roles:
    - install_wordpress_db

- hosts: web
  vars:
    software:
      - httpd
      - php
  roles:
    - install_wordpress_web
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Playbooks

Common Elements - Connections

- hosts – The declarative list of hosts or groups against which this play will run.
- connection – Allows you to change the connection plugin used for tasks to execute on the target
- port – Used to override the default port used in a connection
- remote_user – User to define/override which user is connecting to the remote system
- become – Boolean that controls if privilege escalation is used or not on Task execution.
  (also become_flags, become_user, become_method)

- **NetApp plays will usually have ‘hosts: localhost’ and will not use any of the other elements**

Ansible Playbooks

Common Elements – Inventory and Variable Handling

- order – Controls the sorting of hosts as they are used for executing the play. Possible values are inventory, sorted, reverse_sorted, reverse_inventory and shuffle. Not used with NetApp plays

- vars – Dictionary/map of variables
- vars_files – List of files that contain vars to include in the play
- vars_prompt – list of variables to prompt for on launch

Ansible Playbooks

Common Elements – Information Handling

- name – Identifier. Can be used for documentation, in or tasks/handlers
- gather_facts – Boolean (default yes) allows the bypass of fact gathering. This can speed up connection time where facts are not needed in a playbook. For NetApp plays should always be set to 'false'.
- no_log – Boolean that controls information disclosure and logging.
- ignore_errors – Boolean. When set to **yes**, errors will be ignored unless absolutely fatal to the playbook execution
- check_mode – Also known as "dry run" mode, will evaluate but not execute. For modules that support check mode, the module will report the expected result without making any changes as a result of the tasks.

Ansible Playbooks

Common Elements – Task Handling

- pre_tasks – A list of tasks to execute before roles.
- roles – List of roles to be imported into the play
- tasks – Main list of tasks to execute in the play. They run after roles and before post_tasks.
- post_tasks – A list of tasks to execute after the roles section.
- handlers – Just like regular tasks but are only run if the Task contains a “notify” directive and also indicates that it changed something. For example, if a config file is changed then the task referencing the config file templating operation may notify a service restart **handler**.

Ansible Playbooks

Ansible Plays

```
---
- name: install a LAMP stack
  hosts: web,db,appserver01
  become: yes
  vars:
    my_greeting: Welcome to my awesome page
    favorite_food: fried pickles

  roles:
  - install_lamp_elements

  tasks:
  - name: write the index file
    copy:
      content: {{ my_greeting }}. Enjoy some {{ favorite_food }}”
      dest: /var/www/html/index.html
    notify: reload_apache

  handlers:
  - name: reload_apache
    service:
      name: httpd
      state: reloaded
```

Some examples sourced from
https://github.com/ansible/workshops/blob/master/decks/ansible_rhe
l.pdf

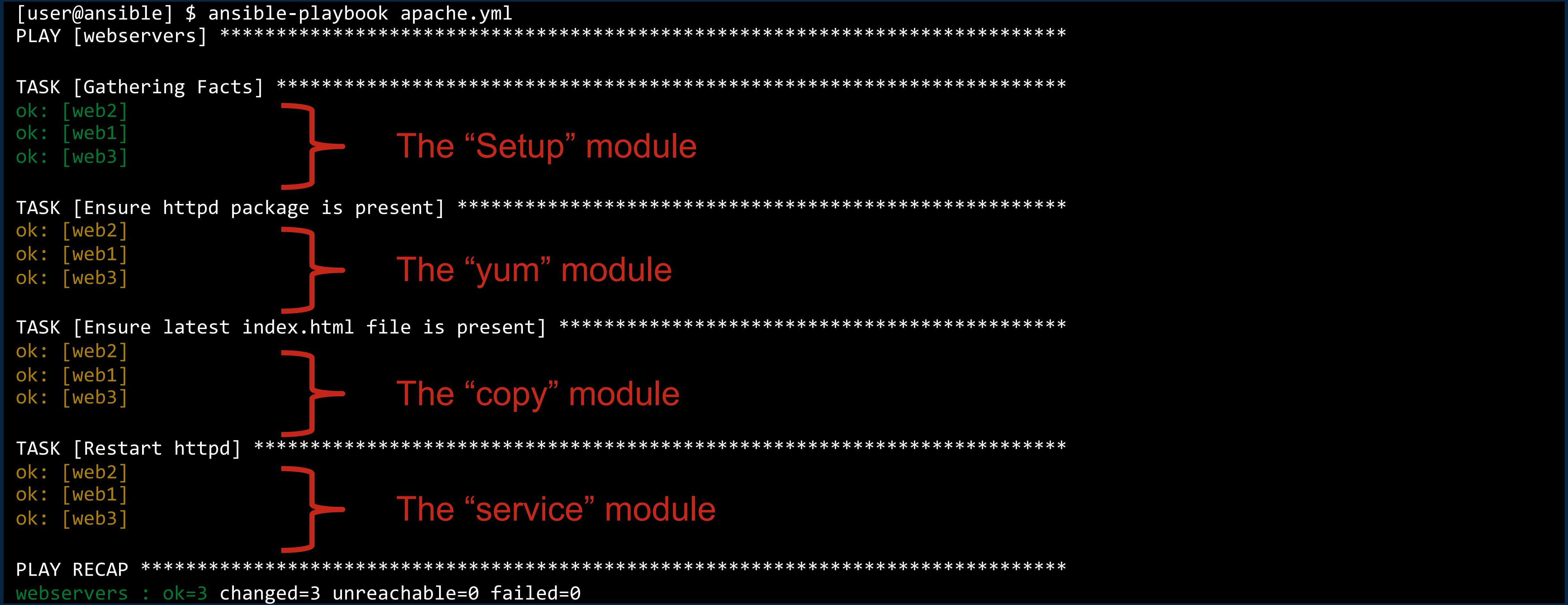
Ansible Playbooks

Using Tasks

```
---
tasks:
 - name: Ensure httpd package is present
   yum:
     name: httpd
     state: latest

 - name: Ensure latest index.html file is present
   copy:
     src: files/index.html
     dest: /var/www/html/

- name: Restart httpd
  service:
    name: httpd
    state: restart
```

Ansible Playbooks

Running the Playbook



```
[user@ansible] $ ansible-playbook apache.yml
PLAY [webservers] ****************************************************************************

TASK [Gathering Facts] ***********************************************************************
ok: [web2]
ok: [web1]
ok: [web3]

TASK [Ensure httpd package is present] *******************************************************
ok: [web2]
ok: [web1]
ok: [web3]

TASK [Ensure latest index.html file is present] **********************************************
ok: [web2]
ok: [web1]
ok: [web3]

TASK [Restart httpd] *************************************************************************
ok: [web2]
ok: [web1]
ok: [web3]

PLAY RECAP ***********************************************************************************
webservers : ok=3 changed=3 unreachable=0 failed=0
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Playbooks

Running Outputs

A task executed as expected, no change was made.

A task executed as expected, making a change

General text information and headers

A conditional task was skipped

A bug or deprecation warning

A task failed to execute successfully

NetApp

1.4 Working with Playbooks

Variables

Conditionals, Handlers, Loops

Templates

Roles

Ansible Variables

Variable examples

```
---
- hosts: all

  vars:
    var_one: one is the loneliest number
    var_two: two can be as sad as one
    var_three: three dog night said that
    var_four: "{{ var_three }} {{ var_one }}"
    var_five: "and that {{ var_two }}."
```

```
three dog night said that one is the loneliest number

and that two can be as sad as one.
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Variables

Variables and Facts/Info

```
"ontap_info": {
             "aggregate_info": {
                "aggr0_ansible1_01": {
                    "aggr_fs_attributes": {
                        "block_type": "64_bit",
                        "fsid": "1201229146",
                        "type": "aggr”
                    },
```

A variable defined in our playbook

```
vars:
  volume: vol_name1
```

With in a playbook you can use a mix of hardcoded info, variables, or collected facts or info.

```
name: “{{ volume }}”
size: 10
aggregate: “{{ ontap_info.aggregate_info }}”
```

Ansible Variables

Variables Precedence

1. extra vars (highest, overwrites all)
2. task vars (overridden just for that task)
3. block vars (overridden just for that block)
4. role and include vars
5. play vars_files
6. play vars_prompt
7. play vars
8. set_facts
9. registered vars
10. host facts
11. playbook host_vars
12. playbook group_vars
13. inventory host_vars
14. inventory group_vars
15. inventory vars
16. role defaults (lowest, will be overwritten by anything)

Ansible Conditionals

When this, do that

```
vars:
  my_mood: happy

tasks:
 - name: conditional task, based on my_mood var

  debug:
     msg: "Come talk to me. I am {{ my_mood }}!"
   when: my_mood == ‘happy’
```

Alternatively

```
  debug:
    msg: "Feel free to interact. I am {{ my_mood }}"
  when: my_mood != ‘grumpy’
```

Ansible Handlers

What to run when something runs

```
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest
  notify: restart_httpd

  handlers:
  - name: restart_httpd
    service:
      name: httpd
      state: restart
    when: httpd_results.changed
```

Ansible Loops

Don’t Do This

```
- yum:
    name: httpd
    state: latest

- yum:
    name: httpd-tools
    state: latest

- yum:
    name: mysql-server
    state: latest

- yum:
    name: php56-mysql
    state: latest
```

Ansible Loops

This is better

```
- name: ensure a list of packages are installed
  yum:
    name: “{{ packages }}”
    state: latest
  vars:
    packages:
      - httpd
      - httpd-tools
      - mysql-server
      - php56-mysqlnd
```

Ansible Templates

Advanced Playbooks

```
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
    - name: Verify correct config file is present
      template:
        src: templates/httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
```

```
## Excerpt from httpd.conf.j2

# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
# Listen 80 ## original line
Listen {{ http_port }}

# DocumentRoot: The directory out of which you will server your documents
# DocumentRoot "/var/www/html"
DocumentRoot {{ http_docroot }}
```

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

NetApp

Ansible Roles

- Think Ansible packages
- Provide Ansible with a way to load tasks, handlers, and variables from separate files
- Group content, allowing easy sharing of code with others
- Make larger projects more manageable
- Can be developed in parallel by different administrators

Ansible Roles

Structure

- Defaults: default variables with lowest precedence
- Handlers: contains all handlers
- Meta: role metadata including dependencies to other roles
- Tasks: plays or tasks
- Templates: templates to deploy
- Tests: placement for playbook tests
- Vars: variables (override user defined variables)

```
user/
|- defaults
|    - main.yml
|- handlers
|    - main.yml
|- meta
|    - main.yml
|- README.md
|- tasks
|    - main.yml
|   - templates
|- tests
|    - inventory
|    - test.yml
|- vars
|    - main.yml
```

Ansible Collections and Roles

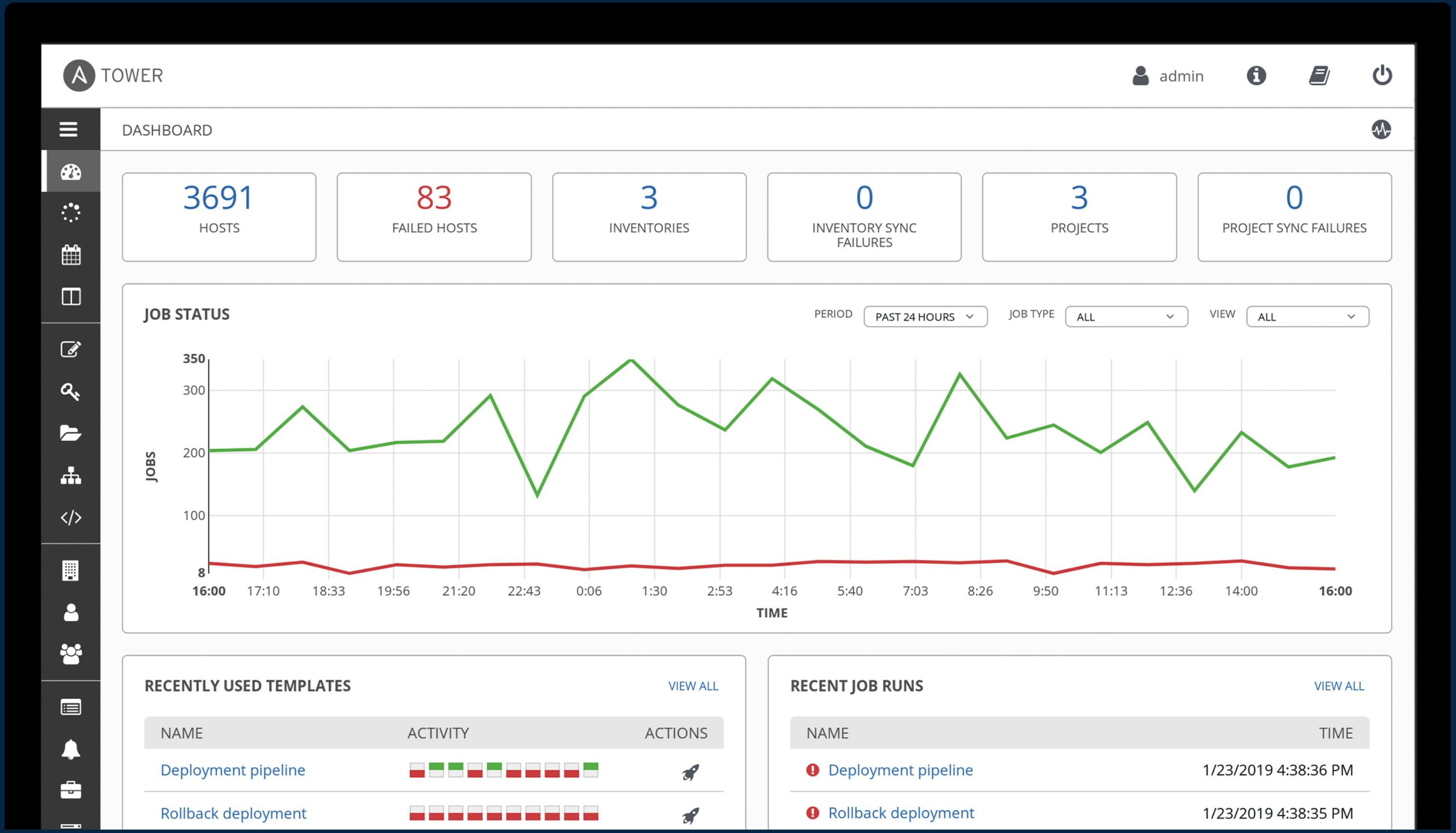
Galaxy

- https://galaxy.ansible.com
- Sharing Content
- Certified and Community
- Collections and Roles

2.1 Tower

Ansible Tower

- Subscription RH Ansible Tower
- Ansible Tower AWX



3691	83	3	0	3	0
HOSTS	FAILED HOSTS	INVENTORIES	INVENTORY SYNC FAILURES	PROJECTS	PROJECT SYNC FAILURES

JOB STATUS

PERIOD: PAST 24 HOURS · JOB TYPE: ALL · VIEW: ALL

RECENTLY USED TEMPLATES · VIEW ALL

NAME	ACTIVITY	ACTIONS
Deployment pipeline	🟥🟩🟩🟥🟩🟥🟥🟥🟥🟩	
Rollback deployment	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥	

RECENT JOB RUNS · VIEW ALL

NAME	TIME
Deployment pipeline	1/23/2019 4:38:36 PM
Rollback deployment	1/23/2019 4:38:35 PM

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Tower

RBAC

Allow restricting playbook access to authorized users. One team can use playbooks in check mode (read-only) while others have full administrative abilities.

Push button

An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

RESTful API

With an API first mentality every feature and function of Tower can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

Workflows

Ansible Tower's multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials

Enterprise integrations

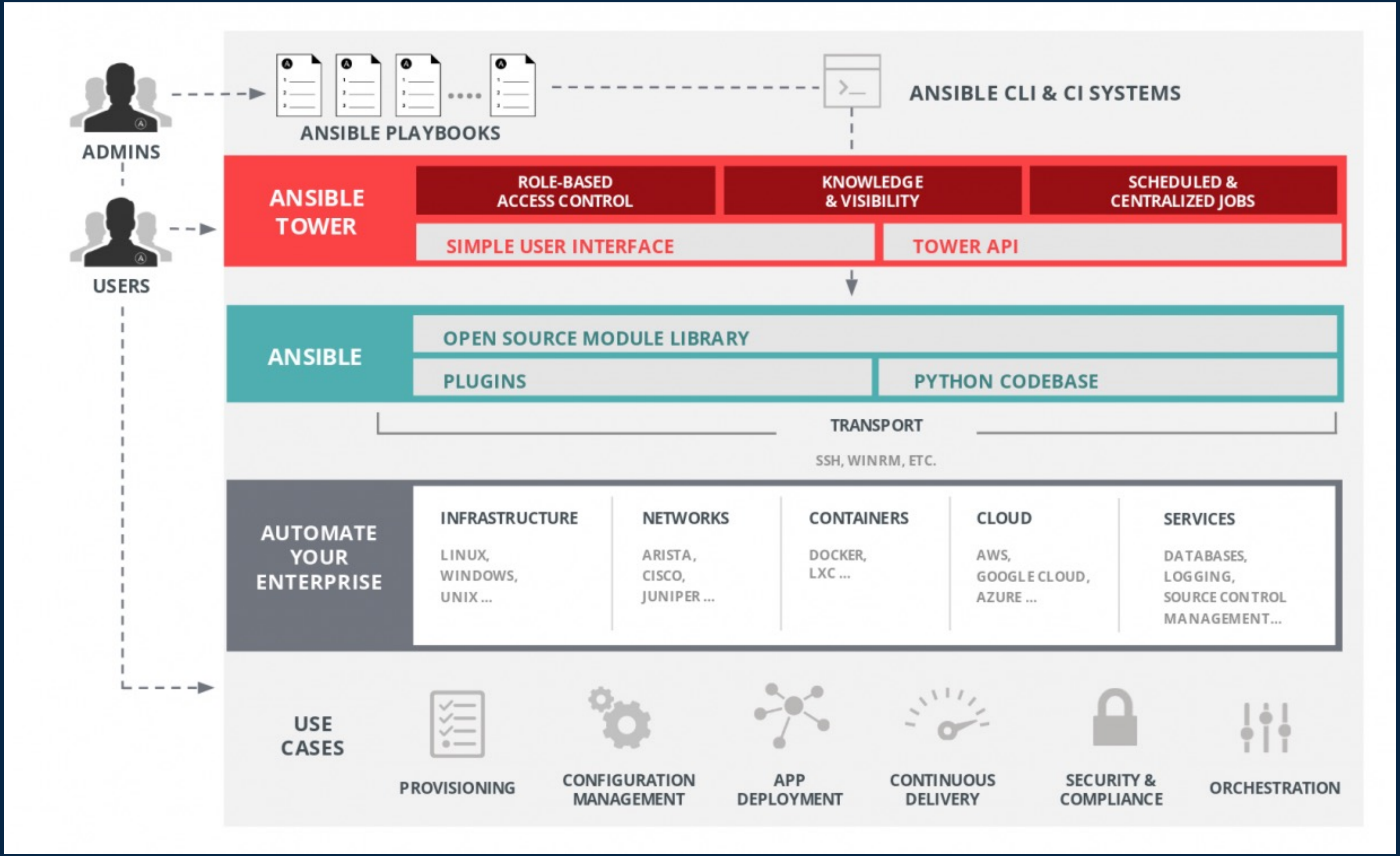
Integrate with enterprise authentication like TACACS+, RADIUS, Azure AD. Setup token authentication with OAuth 2. Setup notifications with PagerDuty, Slack, and Twilio

Centralized logging

All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened – all securely stored and viewable later, or exported through Ansible Tower's API.

NetApp

Ansible Tower



Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

2.2 Tower

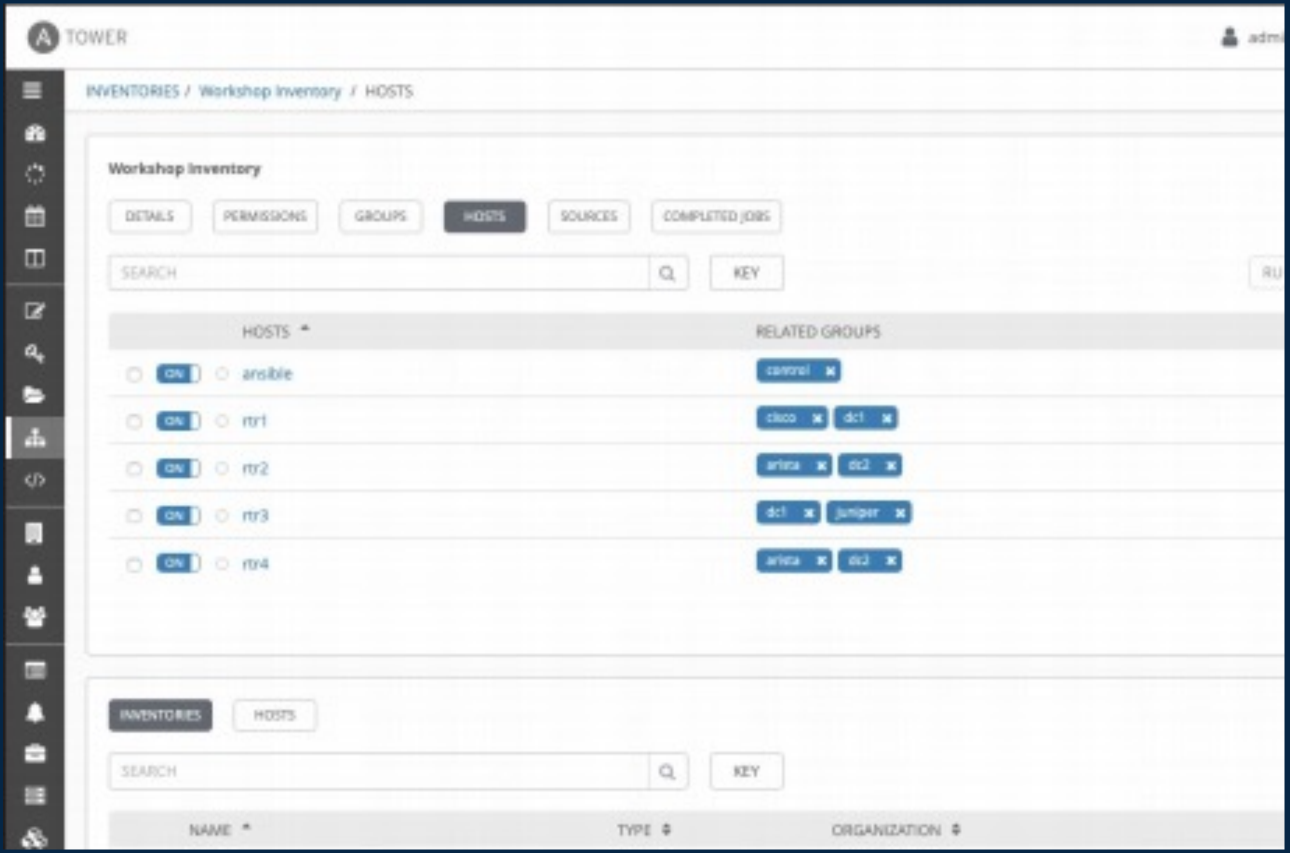
Inventories

Credentials

Ansible Tower

Inventory

- Hosts (nodes)
- Groups
- Inventory-specific data (vars)
- Static or dynamic sources



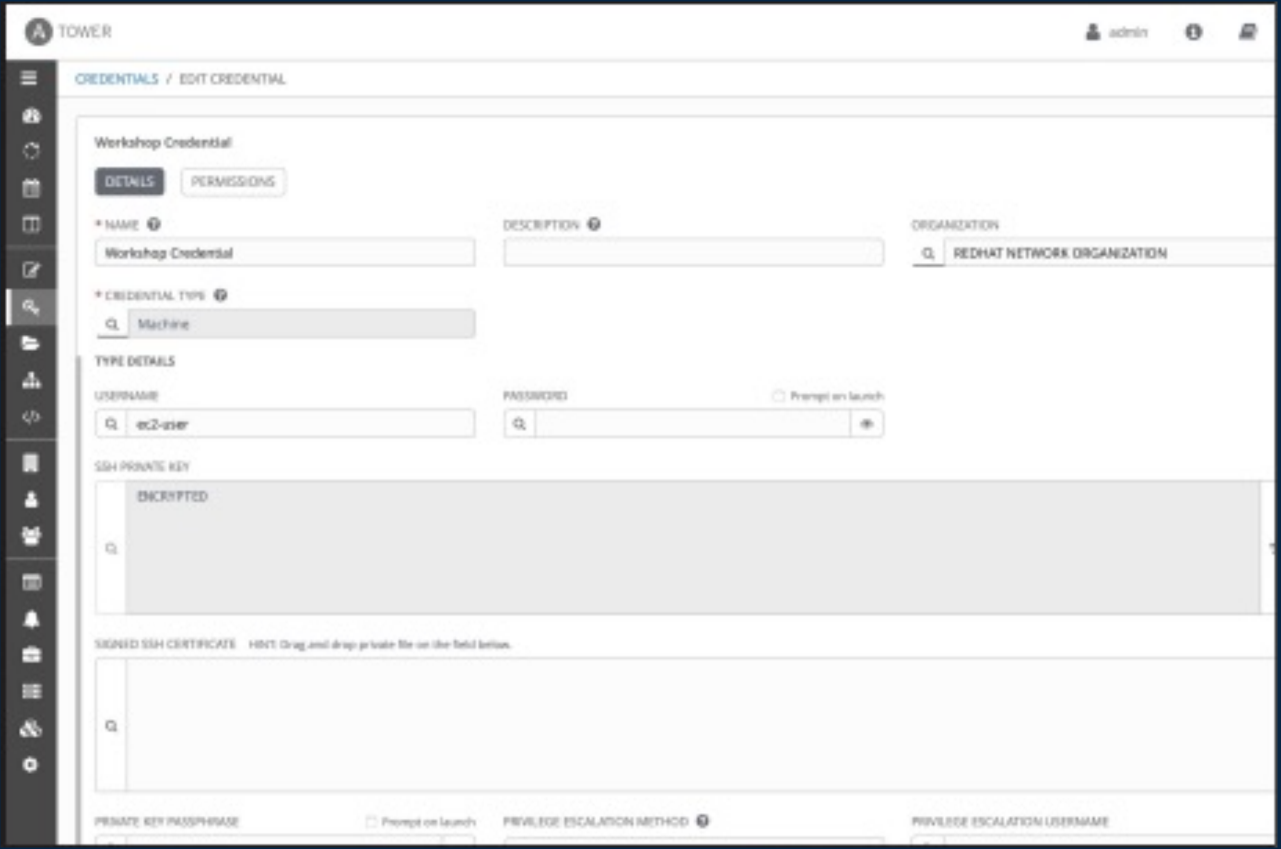
HOSTS	RELATED GROUPS
ansible	control
rtr1	cisco, dc1
rtr2	arista, dc2
rtr3	dc1, juniper
rtr4	arista, dc2

Some examples sourced from
https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Tower

Credentials

- Connecting to remote machines to run jobs
- Syncing with inventory sources
- Importing project content from version control systems
- Connecting to and managing NetApp devices



TOWER admin

CREDENTIALS / EDIT CREDENTIAL

Workshop Credential

DETAILS PERMISSIONS

* NAME: Workshop Credential

DESCRIPTION:

ORGANIZATION: REDHAT NETWORK ORGANIZATION

* CREDENTIAL TYPE: Machine

TYPE DETAILS

USERNAME: ec2-user

PASSWORD: Prompt on launch

SSH PRIVATE KEY: ENCRYPTED

SIGNED SSH CERTIFICATE HINT: Drag and drop private file on the field below.

PRIVATE KEY PASSPHRASE: Prompt on launch

PRIVILEGE ESCALATION METHOD

PRIVILEGE ESCALATION USERNAME

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

2.3 Tower

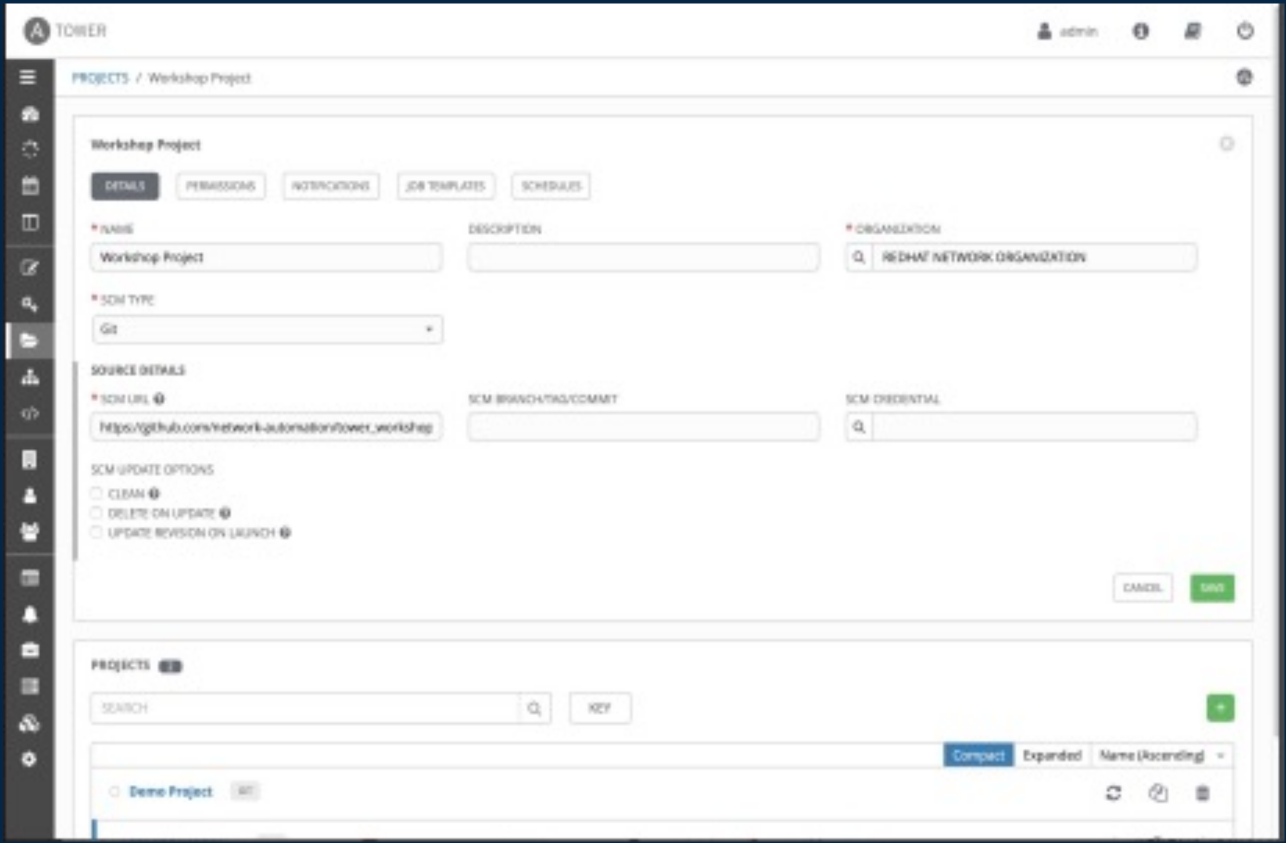
Projects

Job Templates

Ansible Tower

Projects

- Collection of Ansible Playbooks
- Usually used with a SCM
  - Git
  - Subversion
  - Mercurial



Workshop Project

* NAME	DESCRIPTION	* ORGANIZATION
Workshop Project		REDHAT NETWORK ORGANIZATION
* SCM TYPE		
Git		

SOURCE DETAILS

* SCM URL	SCM BRANCH/TAG/COMMIT	SCM CREDENTIAL
https://github.com/network-automation/tower_workshop		

SCM UPDATE OPTIONS

- ☐ CLEAN
- ☐ DELETE ON UPDATE
- ☐ UPDATE REVISION ON LAUNCH

PROJECTS

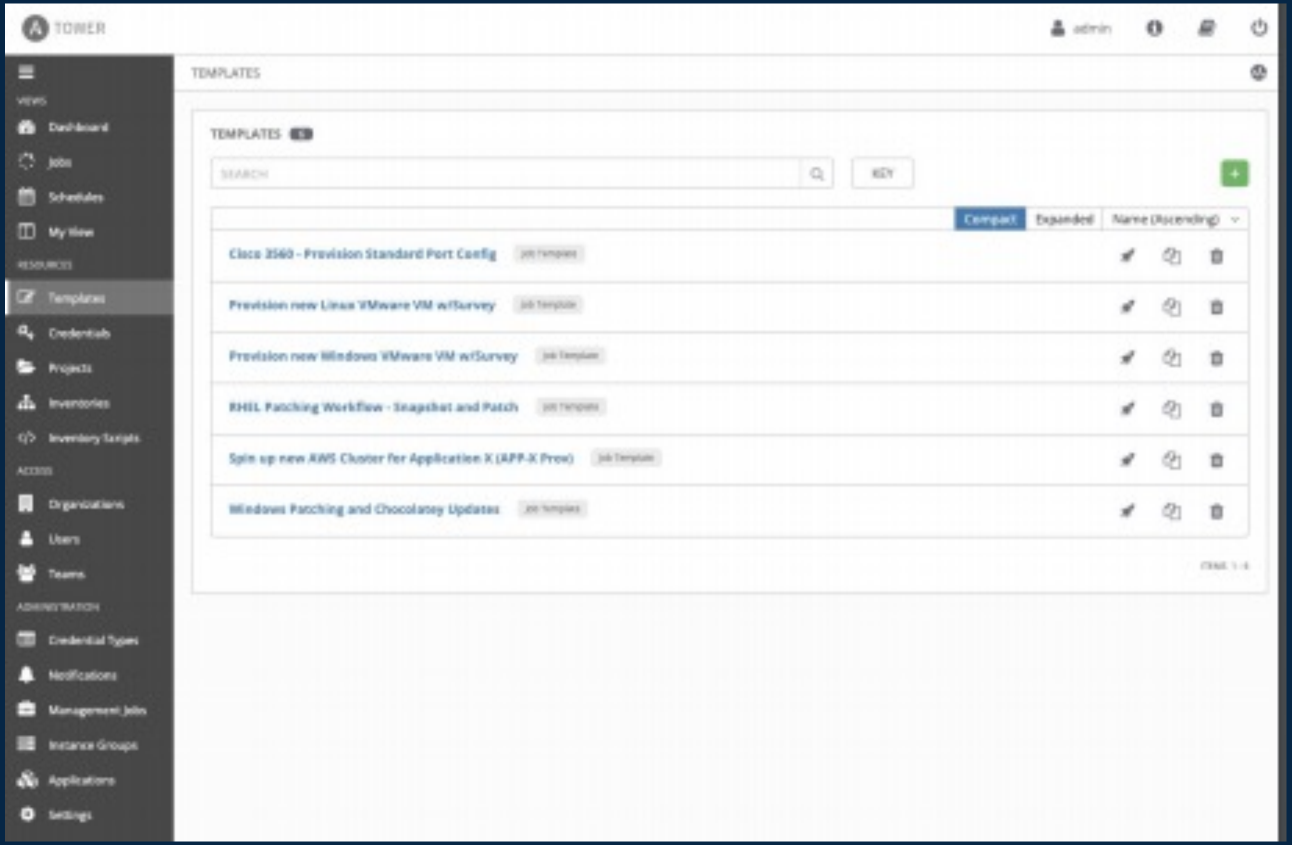
Demo Project GIT

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

Ansible Tower

Job Templates

- Define how a job will run
  - Inventory
  - Project (containing a playbook)
  - Credentials
  - Survey or optional vars
  - Can be launched via GUI or API



TEMPLATES

Cisco 3560 - Provision Standard Port Config — Job Template
Provision new Linux VMware VM w/Survey — Job Template
Provision new Windows VMware VM w/Survey — Job Template
RHEL Patching Workflow - Snapshot and Patch — Job Template
Spin up new AWS Cluster for Application X (APP-X Prox) — Job Template
Windows Patching and Chocolatey Updates — Job Template

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

2.4 Tower

Surveys

Ansible Tower

Surveys

- Runs a series of customized questions before the job runs in a user-friendly manner
- Question and answer format to allow self-service without understanding underlying concepts about setting variables.

2.5 Tower

Role Based Access Control (RBAC)

Ansible Tower

RBAC

- Role-Based Access Controls (RBAC) are built into Ansible Tower and allow administrators to delegate access to inventories, organizations, and more. These controls allow Ansible Tower to help you increase security and streamline management of your Ansible automation.

Ansible Tower

RBAC

- Organization is a logical collection of users, teams, projects, inventories, etc.
  - All entities belong to an organization with the exception of users
- A user is an account to access Tower and its services provided permissions are granted
- Teams allow easier role-based access across organizations

NetApp

2.6 Tower

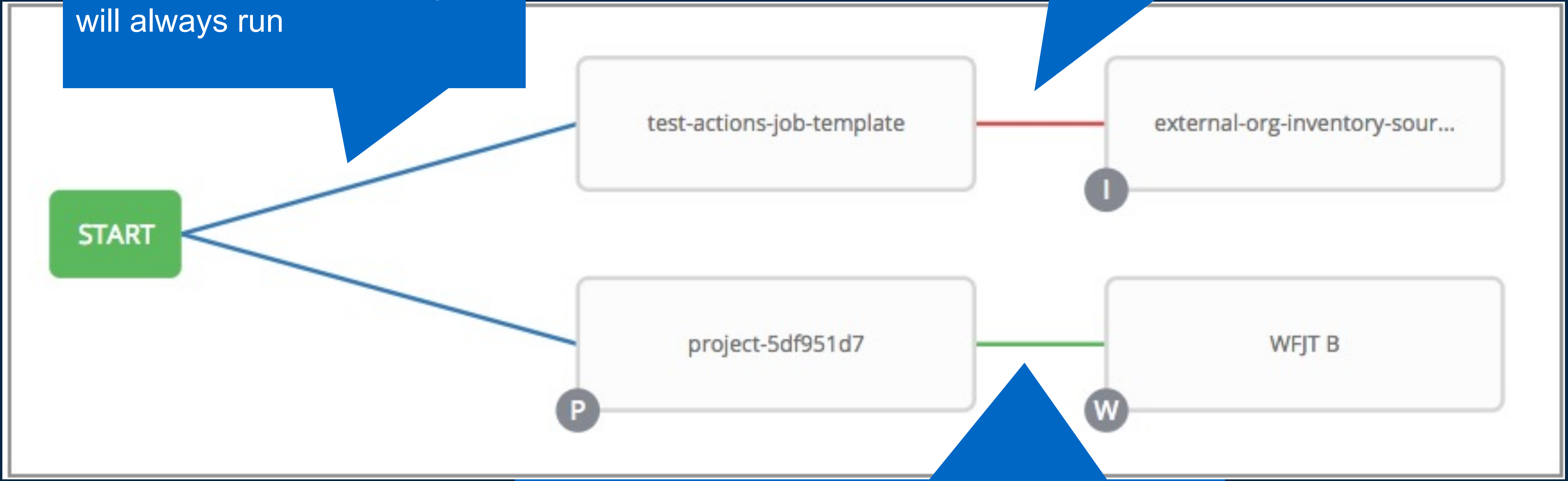
Workflows

Ansible Tower

Workflows

Red indicates this Job Template will only be run if the previous Job Template fails

Blue indicates a Job Template will always run



Green indicates this Job Template will only run if the previous Job Template is successful

Some examples sourced from https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf

NetApp

Learn More

- https://www.netapp.io
- Slack (www.netapp.io/slack)
- https://netapp.io/2018/10/08/getting-started-with-netapp-and-ansible-install-ansible/
- https://netapp.io/2021/08/19/how-to-guide-setting-up-awx-on-a-single-host/
- https://www.github.com/netapp-automation/ansible

Thank You

Some examples sourced from
https://github.com/ansible/workshops/blob/master/decks/ansible_rhel.pdf