

ansible

simple IT automation



Prerequisites



- ❖ Familiarity with Linux commands
- ❖ Basic knowledge on networking concepts
- ❖ Basic Knowledge on Networking(CIDR blocks, subnet etc.)
- ❖ Target Audience
 - System administrators
 - Software developers in a DevOps role
 - **Anyone who wants to learn!!**

About you

- ❖ Please tell me about Yourself:
 - Your Name
 - Your background
 - What is the purpose of this course?
 - Where and how you will be using this knowledge?
 - What do you currently know about Ansible?

About me

- ❖ Your Trainer : Deepak Gupta(@hellodk01)
- ❖ Experience : 9+ Years
- ❖ Certifications
 - Blockchain for Developers
 - Interfacing with the Raspberry Pi
 - Big Data, Cloud Computing, & CDN Emerging Technologies

About me

❖ Industry Roles

- Devops Lead, MoveinSync
- Systems Engineer, Myntra Designs
- Devops Engineer, Knowlarity Communications
- Software Engineer, Wipro Technologies

❖ Hobbies :

- Photography
- Travelling
- Trekking

Trainings Delivered

- ❖ Cloud Computing : AWS Solutions, Azure DevOps
- ❖ Container Technologies : Docker, Kubernetes
- ❖ Monitoring Tools : Sensu, Zabbix, Nagios, Icinga2
- ❖ SQL Databases : MySQL, PostgreSQL, MariaDB
- ❖ NoSQL Databases : MongoDB, Cassandra, Redis, Gemfire
- ❖ Web Server : Nginx Setup and Configurations
- ❖ Messaging Tools : RabbitMQ, Kafka
- ❖ Configuration Management: Ansible, Chef, Puppet, Saltstack
- ❖ Architecture : Microservices, DevOps, DevSecOps
- ❖ Programming : Java, Python, Golang, rust, haskell

Course Organization

- ❖ Hours: 9:30 hrs to 17:30 hrs IST

- ❖ Breaks:



- ❖ We would be using Ubuntu 18.04 as our primary OS

Course Organization

- ❖ Organize yourself into groups
- ❖ Make sure that members of each group sit together
- ❖ I hope lab details are already shared with you all
 - your VM's are up and running
 - if not execute the steps as mentioned in the document already provided

Credits

- ❖ some of the images/materials may be borrowed from the internet and not owned by us
- ❖ we would extend our gratitude to the original content authors of those images/contents

Materials

❖ Slides

- Day wise slides – Before the session starts
- Final sides – On the last day

❖ Additional Reading materials

- First day

❖ Specific references/materials

- Upon request and time frames

Guidelines for the session

- ❖ Please login 10 minutes before the time
- ❖ Please do a check on your network connection and audio before the class to have a smooth session
- ❖ All participants will be on mute, by default
- ❖ Unmute yourself when requested or as needed
- ❖ Ask and answer questions to make your learning interactive
- ❖ Most often logging off or rejoining will help solve the tool related issues

AGENDA



❖ Introduction to Ansible

- history and reason for development of Ansible
- Brief comparison with Saltstack and others
- Benefits and limitations of using Ansible
- Ansible Architecture & core components
- Learning Environment
- Yaml Syntax

❖ Quick Examples

- What is an ad-hoc command
- Ad-hoc commands examples
- discussions about the ansible command

AGENDA



- ❖ Inventories in Ansible
 - Static Inventories
 - Dynamic Inventories
- ❖ Ansible Playbooks
 - Commonly used Modules
 - Using modules in playbooks
 - Register, Debug, stdout & stderr
- ❖ Using Conditionals
- ❖ Error Handling in Playbooks
- ❖ Tagging tasks in Playbooks

AGENDA



- ❖ Templates
- ❖ Using Ansible facts
- ❖ Using variables to gather server info
- ❖ roles and includes
- ❖ create a role to install apache

AGENDA



- ❖ Ansible galaxy and how its used
- ❖ Using multiple roles
- ❖ What is parallelism
 - Parallelism in a playbook
- ❖ Adding windows nodes
- ❖ Setting up patch management
- ❖ Installing softwares using chocolatey
- ❖ setting repository for the custom module
- ❖ ansible-vault
- ❖ Ansible Tower
 - add data to inventory
 - run a sample task

AGENDA



❖ Other

- Ansible Galaxy
- AWX Project
- Testing Strategies
- YAML Syntax

For Ansible Tower Demo, please fill in the form from the below link

❖ <https://www.ansible.com/products/tower/trial>

Alternatively we can utilize AWX – open source version of Tower



Introduction to Ansible

Introduction

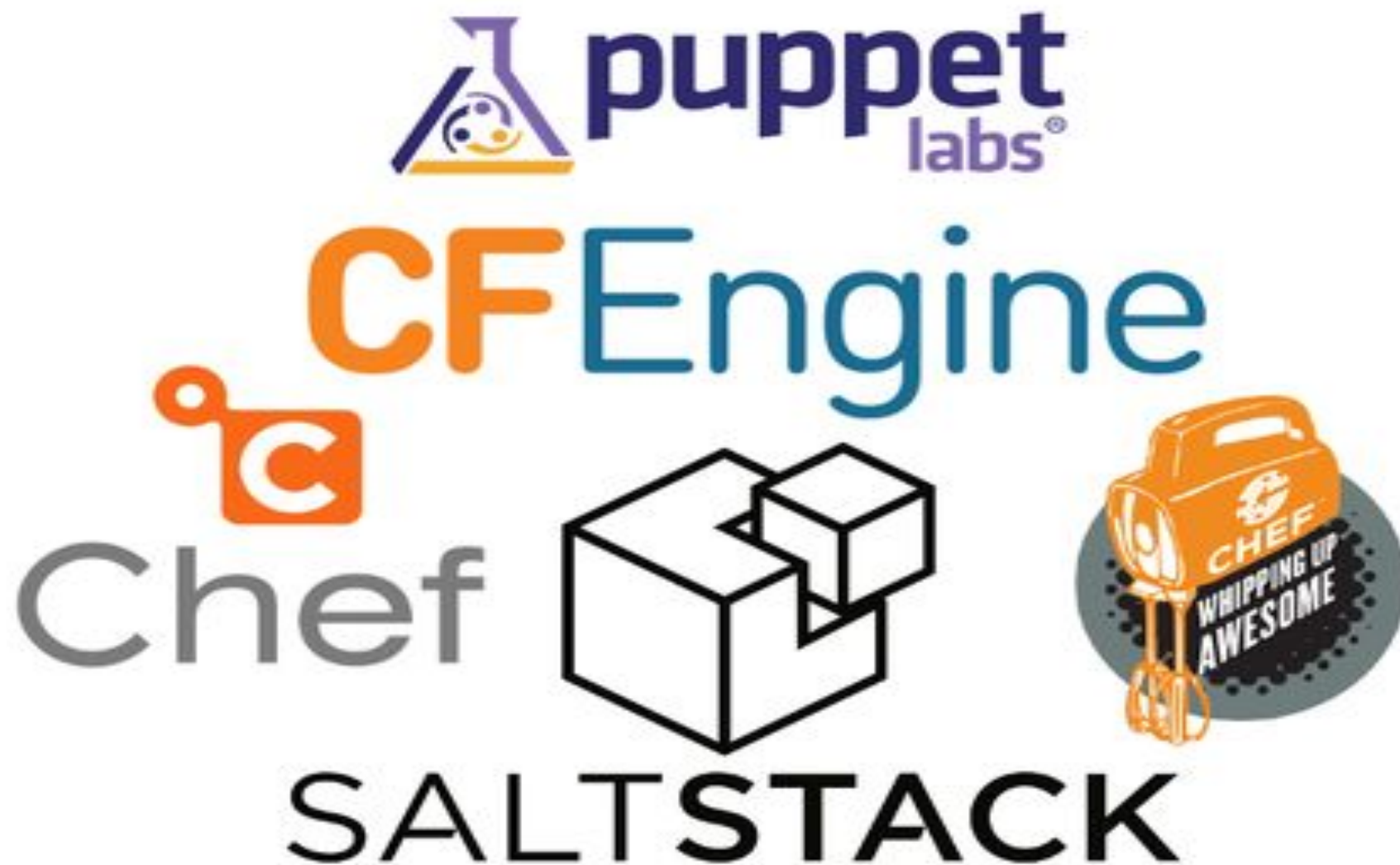


- ❖ What is Configuration management?
 - practices & tooling to automate the delivery and operation of infrastructure
- ❖ solutions model infrastructure, continually monitor & enforce desired configurations, and automatically remediate any unexpected changes or configuration drift
- ❖ deliver better software faster, configuration management helps lay the foundation for DevOps

Introduction

- ❖ Before we can deploy our software , we need to do a number of things:
 - Add user accounts and passwords
 - Configure security settings and privileges
 - Install all the packages needed to run the app
 - Customize the configuration files for each of these packages
 - Create databases and database user accounts; load some initial data
 - Configure the services that should be running
 - Deploy the app code and static assets
 - Restart any affected services
 - Configure the machine for monitoring

Introduction



Introduction

- ❖ term "ansible" was coined by Ursula K. Le Guin in 1966
 - in the novel Rocannon's World
 - refers to fictional instantaneous communication systems
- ❖ Ansible tool was developed by Michael DeHaan
 - author of the provisioning server application Cobbler
 - co-author of the Fedora Unified Network Controller (Func) framework for remote administration
- ❖ Ansible, Inc. (originally AnsibleWorks, Inc.) was the company set up to commercially support and sponsor Ansible
- ❖ Red Hat acquired Ansible in October 2015

Introduction

- ❖ included as part of the Fedora distribution of Linux, owned by Red Hat
- ❖ available for
 - Red Hat Enterprise Linux
 - CentOS
 - OpenSUSE
 - SUSE Linux Enterprise
 - Debian
 - Ubuntu
- ❖ Not available for
 - Windows

Introduction

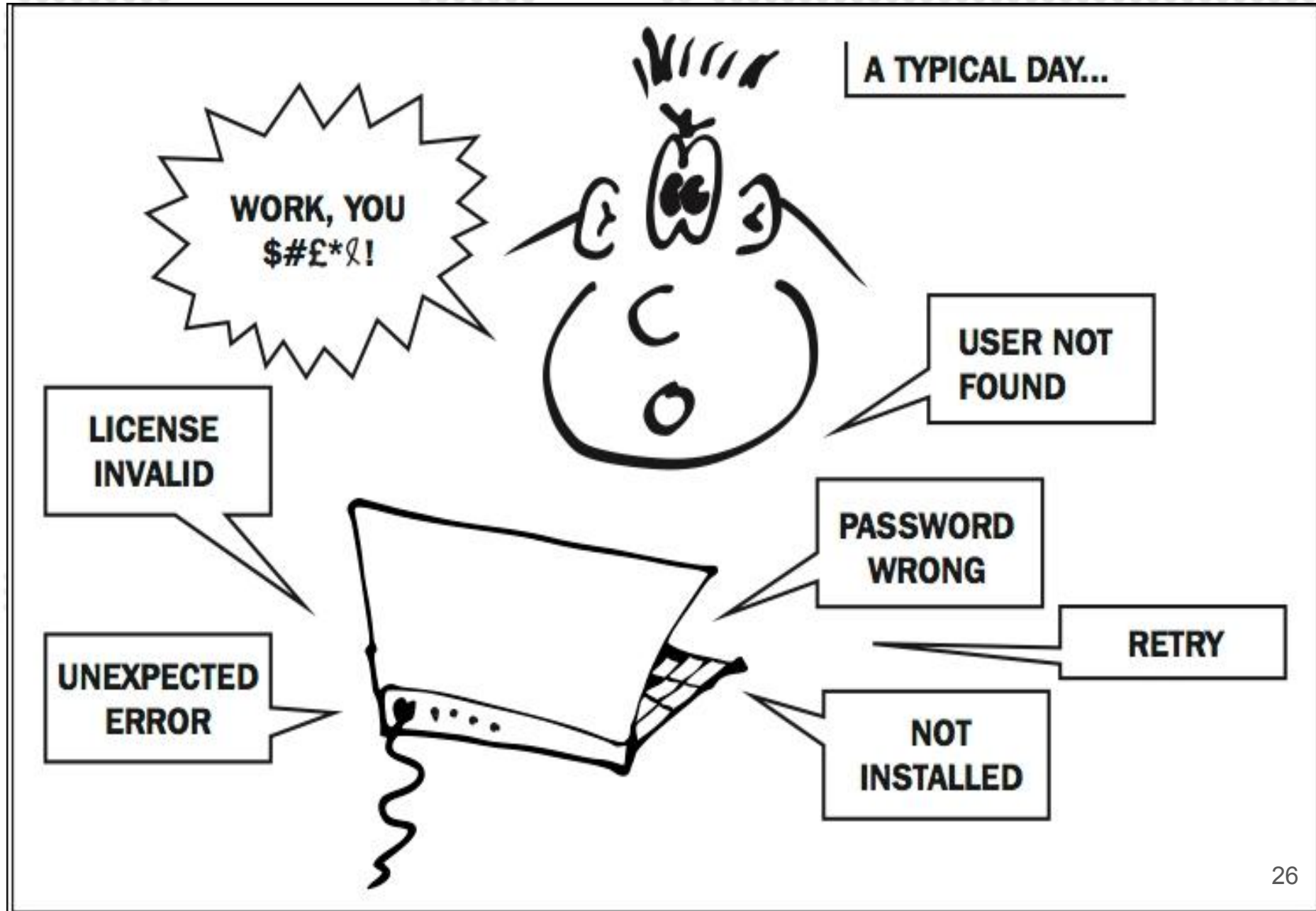
- ❖ Ansible website says
 - doing the same tasks over and over?
 - solve problems once and then automate your solutions going forward?
 - Ansible is here to help



Introduction

- ❖ What is Ansible?
 - open-source software provisioning, configuration management tool
 - runs on many Unix-like systems
 - can configure both Unix-like systems as well as Microsoft Windows
 - includes its own declarative language to describe system config
 - written by Michael DeHaan and acquired by Red Hat in 2015
 - uses push approach
 - centralized infrastructure, configuration management
 - Ansible is agentless
 - connecting remotely via SSH or remote PowerShell to do its tasks
 - acquired by redhat in October 2015

Introduction



Introduction

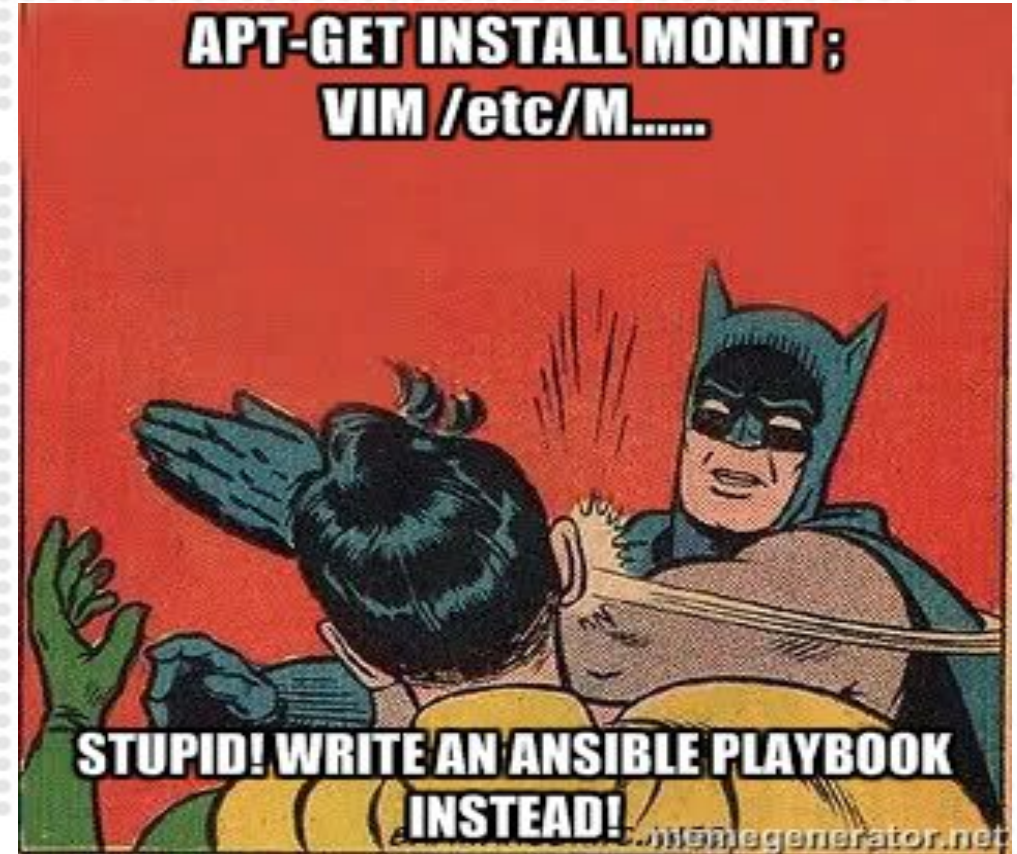
- ❖ used for
 - remote task execution
 - configuration management
 - Infrastructure as Code
 - Network deployment and management
 - hybrid cloud control

Introduction

- ❖ design goals
 - Minimal in nature
 - Consistent
 - Secure
 - Highly reliable
 - Minimal learning required

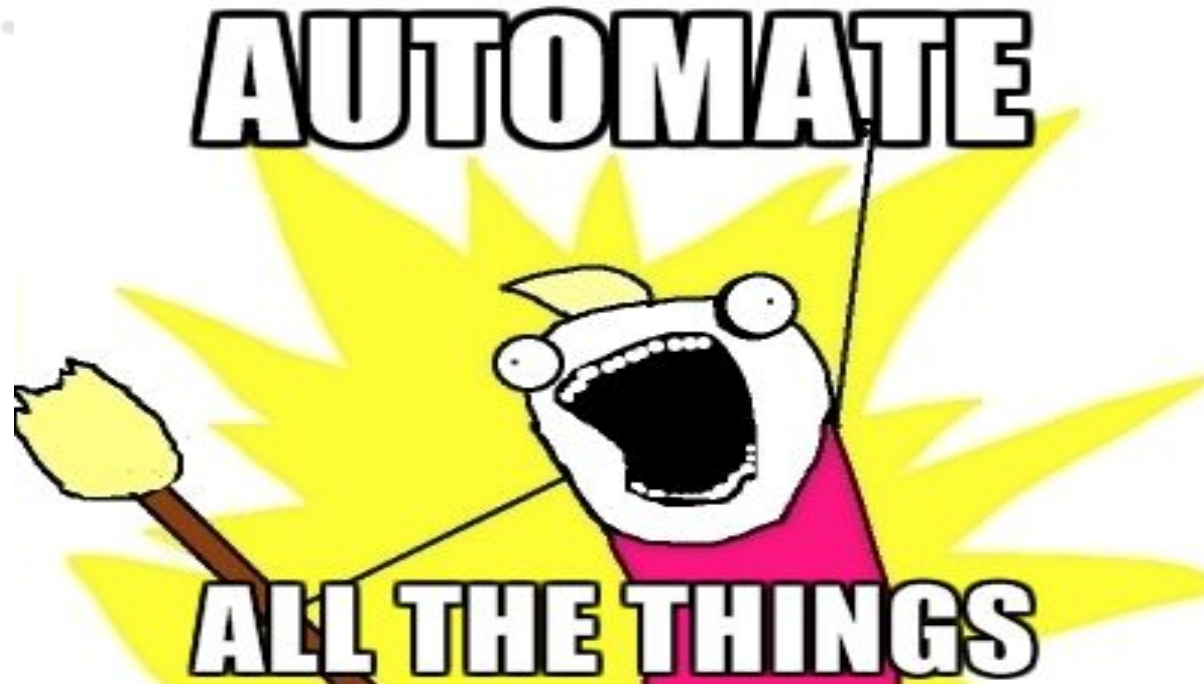
What the heck is Ansible

- ❖ software platform for CM systems
- ❖ Agent-less
- ❖ Secure
- ❖ Scalable
- ❖ Easy learning curve



....so what can
we do with
ansible?

- ❖ package installation
- ❖ shell commands
- ❖ install/update package
- ❖ management systems, clone git
- ❖ stop, start, restart service etc.



ansible comes with batteries included!!

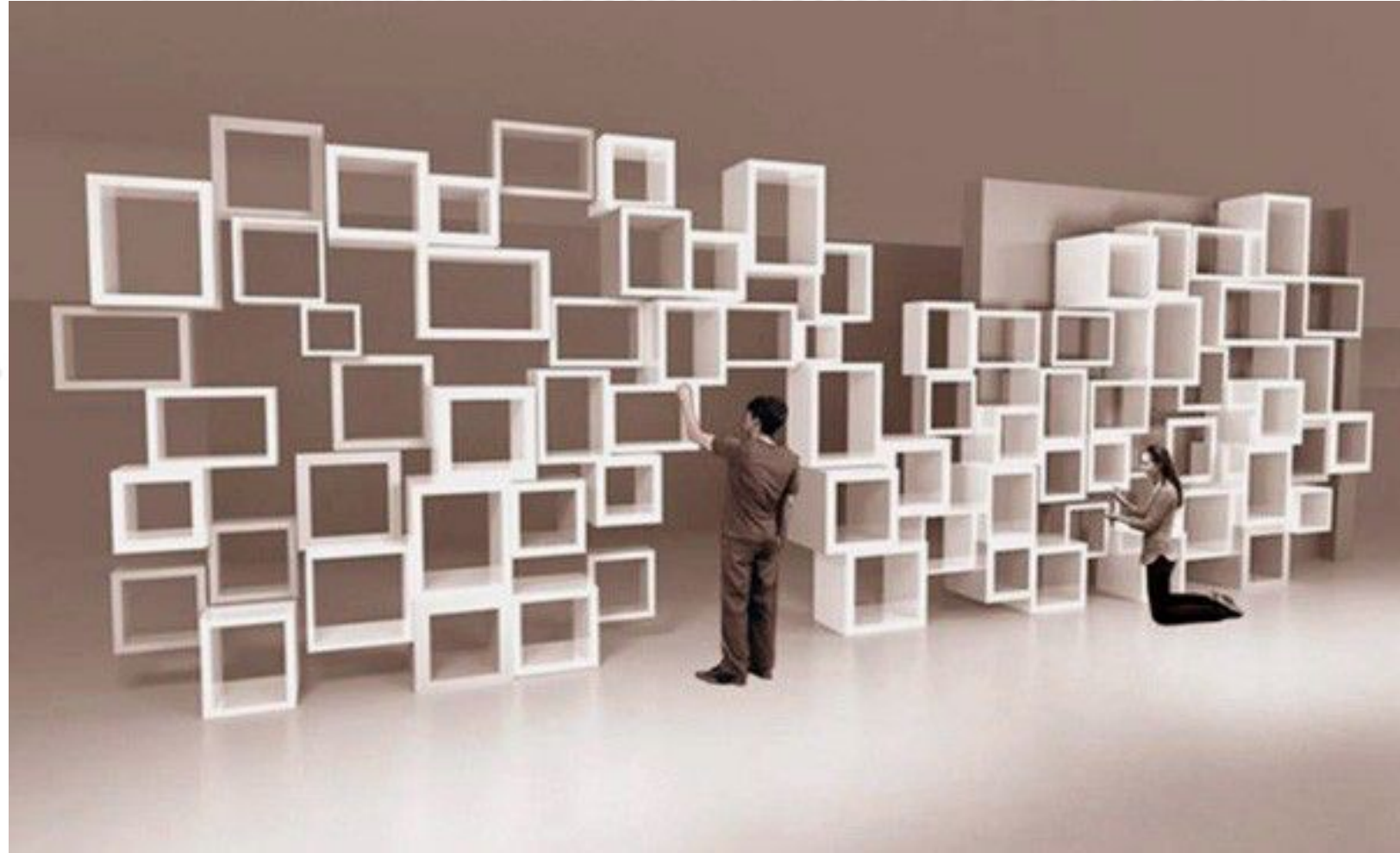
- ❖ Ansible comes bundled with nearly all the mostly used applications/protocols etc in computing world

➤ http://docs.ansible.com/list_of_all_modules.html

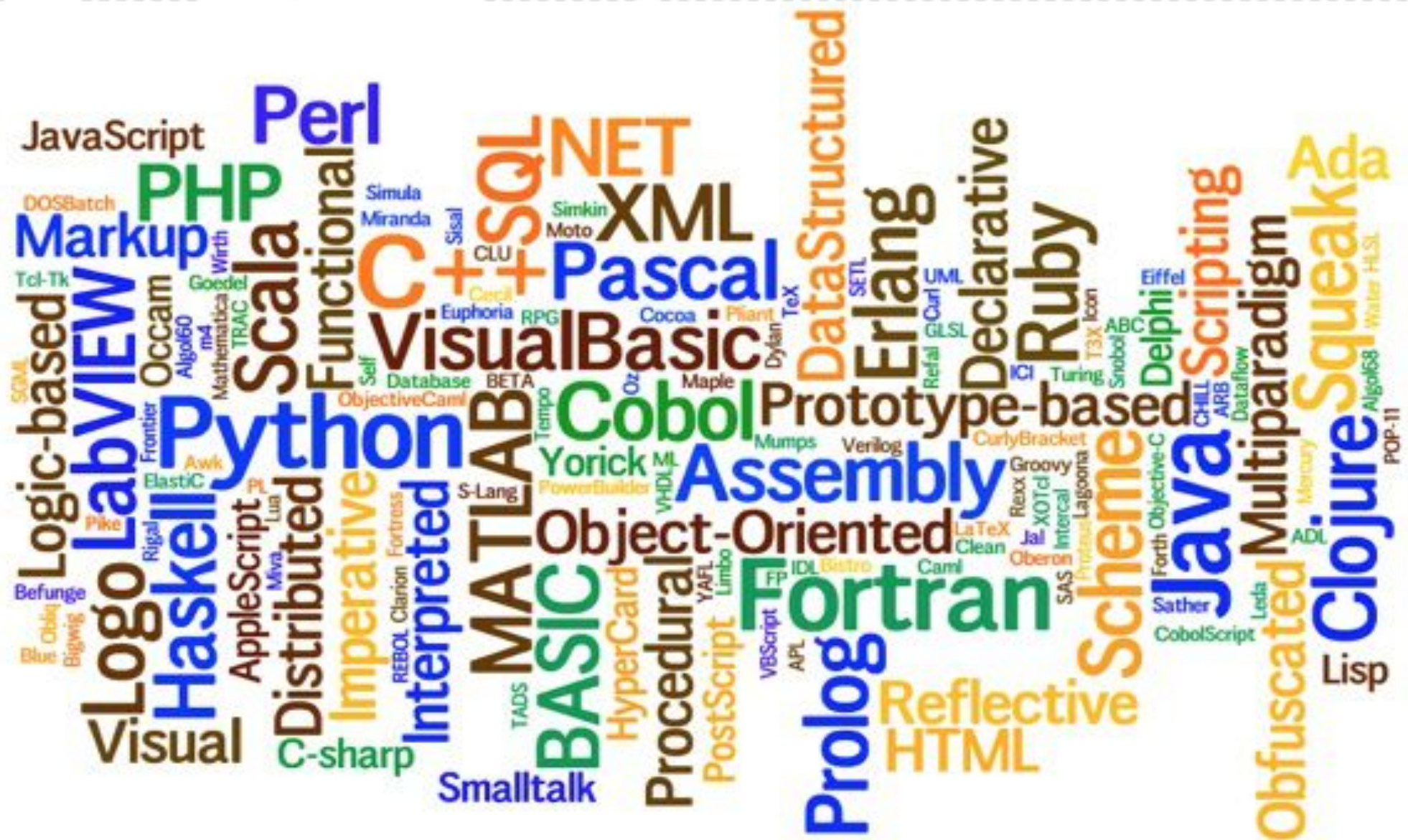


...need anything
out of the box??

- ❖ write down our own custom modules...and the best part...guess??



It's free of
language
barriers





How Ansible Works

Wondering How Ansible works ??

- ❖ uses no agents – Ansible manages machines in an agentless manner.
- ❖ no additional custom security infrastructure – it's easy to deploy
- ❖ Uses YAML – Friendly Language(YAML, in the form of Ansible Playbooks)
- ❖ Uses OpenSSH for transport
- ❖ Highly scalable
- ❖ Idempotent

Wondering How Ansible works ??

- ❖ Ansible works by
 - connecting to the nodes
 - pushing out small programs, called "Ansible modules" to them
 - Ansible then executes these modules (over SSH by default)
 - removes them when finished
- ❖ programs are written to be resource models of the desired state of the system
- ❖ Passwords are supported
- ❖ SSH keys with ssh-agent are one of the best ways to use Ansible
- ❖ `ssh vagrant@192.168.10.30 -T hostname`

Why Ansible

- ❖ much easier to work with compared to the likes of Puppet, Chef etc
- ❖ does not require agents set up on individual nodes
- ❖ supports the pull architecture
- ❖ simple enough for new users
- ❖ works at high enough level to work with other tools as well



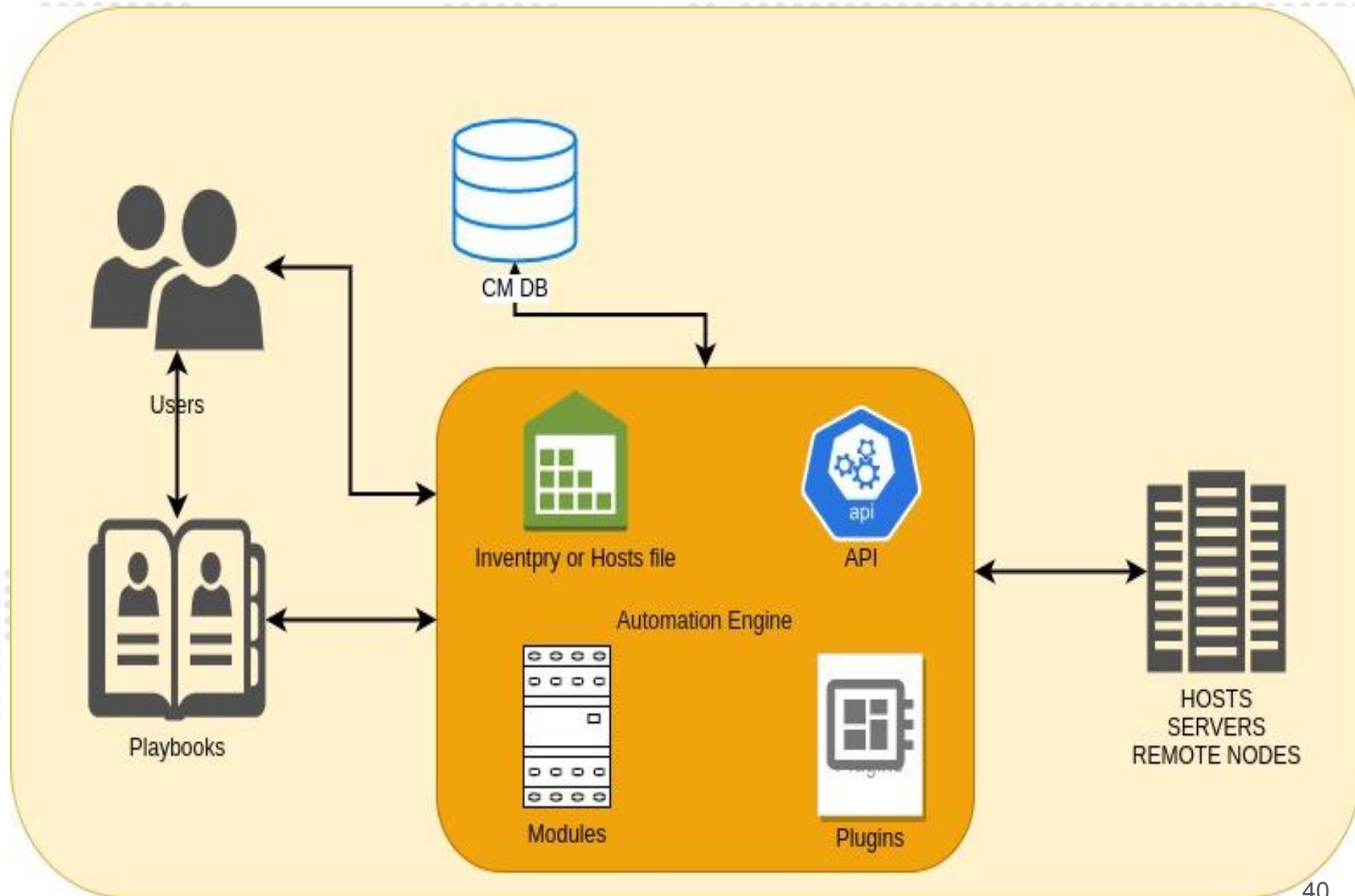
Push vs Pull

<u>Push</u>	<u>Pull</u>
<ul style="list-style-type: none">• No agents required	<ul style="list-style-type: none">• Agents required
<ul style="list-style-type: none">• Synchronous Architecture	<ul style="list-style-type: none">• Async Architecture
<ul style="list-style-type: none">• Central server architecture	<ul style="list-style-type: none">• Master Agent architecture

Ansible Drawbacks

- ❖ Prone to performance issues at times
- ❖ because of ssh based communication

Ansible Architecture



Ansible Architecture

- ❖ Ansible automation engine
 - inventory file
 - playbooks
 - modules
 - API's
 - plugins

Terminologies

- ❖ Control Machine / Node
 - system where Ansible is installed and configured
 - ansible connects to remote nodes and execute commands on them
- ❖ Managed Nodes/Remote Nodes/Remote Servers
 - a server controlled by Ansible
- ❖ Inventory File
 - file that contains information about the servers Ansible controls
 - located at `/etc/ansible/hosts`
- ❖ Playbook
 - file containing a series of tasks to be executed on a remote server

Terminologies

❖ Role

- collection of playbooks and other files that are relevant to a goal such as installing a web server

❖ Play

- a full Ansible run
- can have several playbooks and roles, included
- a single playbook can act as an entry point for other playbooks

❖ Playbooks

- collection of tasks, written in yaml syntax with .yaml extension

❖ tasks

- uses ansible modules to accomplish the job(eg: creating a file)

Terminologies

- ❖ templates
 - file which contains all configuration parameters
 - dynamic values are given as variables
 - During playbook execution, depending on conditions like which OS we are using, variables gets replaced with relevant values
- ❖ modules
 - executed on remote nodes via tasks or directly invoked from the CLI
 - uses python interpreter and implements some functionality
- ❖ hosts
 - remote machines or nodes, defined in the inventory files

Terminologies

❖ groups

- set of hosts performing a specific business goal
- eg: webserver groups, db groups

Ansible vs Salt

<u>Ansible</u>	<u>Salt</u>
Python	Python
Master Less	Master Minion
SSH Based	Commands are issued on the master command line
Stateless	Maintains State



Set up learning environment

Learning Environment

- ❖ developers and engineers, often aim for minimalism and modularity
- ❖ minimalism and modularity overlooked in local development
- ❖ dependencies and tools needed to complete our job
- ❖ dependencies quickly grow in size and spiral into disrepair
- ❖ conflicting versions of the same tools or programs
- ❖ never-ending OS updates – issues
- ❖ with virtual machines – little-to-no risk to our development machine
- ❖ vagrant – takes care of all VM configurations via a Vagrantfile
- ❖ different directory for different Vagrantfile
 - directories defines the environment boundaries in Vagrant

Learning Environment

- ❖ Check if our base/physical system supports virtualization:
 - `cat /etc/cpuinfo | grep -i vmx` #no output for virtual machines
- ❖ already installed VirtualBox from Ubuntu or Debian repository?
- ❖ remove it with:
 - `sudo apt remove virtualbox* -y`
- ❖ Windows?Download the latest virtualbox from the link below:
 - https://www.virtualbox.org/wiki/Linux_Downloads
- ❖ Download the vagrant tool from the link below
 - <https://www.vagrantup.com/downloads.html>

Learning Environment

- ❖ Working on Windows?
 - Ensure Hyper-V is disabled
 - Download the latest virtualbox from the link below:
 - https://www.virtualbox.org/wiki/Linux_Downloads
 - Download the vagrant tool from the link below
 - <https://www.vagrantup.com/downloads.html>

Learning Environment

- ❖ Now install both the .deb packages using the below commands
 - `sudo dpkg -i <VIRTUAL BOX FILE NAME>`
 - `sudo dpkg -i <VAGRANT PACKAGE NAME>`
- ❖ Now create a directory as a non-root user on the Desktop
 - `mkdir ~/Desktop/ansible`
 - `cd ~/Desktop/ansible`
- ❖ Vagrant uses a config file – **Vagrantfile** for it's environment
- ❖ Create the file **Vagrantfile** in the directory
 - `~/Desktop/ansible`
- ❖ Check the `synced_folder` location in the **Vagrantfile**

Learning Environment

- ❖ To turn our VM on, navigate to the directory with our Vagrantfile
 - `vagrant up`
 - `vagrant up ansible`
- ❖ To pause our VM, navigate to the directory with our Vagrantfile
 - `vagrant suspend`
 - `vagrant suspend ansible`
- ❖ To turn our VM off, navigate to the directory with our Vagrantfile
 - `vagrant halt`
 - `vagrant halt ansible`

Learning Environment

- ❖ Destroy our VM, navigate to the directory with our Vagrantfile:
 - `vagrant destroy`
 - `vagrant destroy ansibleM`
- ❖ Possible vm images
 - `ansibleM.vm.box = "ubuntu/xenial64"`
 - `ansibleM.vm.box = "centos/7"`
- ❖ Install a good text editor like Sublime 3
- ❖ Have a good SSH client like MobaXterm

Learning Environment

- ❖ Logging to the VM's
 - Go to the directory where you have the Vagrant file
 - `cd ~/Desktop/ansible`
 - `vagrant ssh ansible`
 - Verify your IP Address with the below command
 - `ip a`
 - Alternatively you can also use normal ssh from any location
 - `ssh vagrant@192.168.10.70`
 - password is **vagrant**

Learning Environment

- ❖ Enable bash completion
 - `sudo apt update`
 - `sudo apt install bash-completion -y`
 - `cat /etc/profile.d/bash_completion.sh`
 - `echo "source /etc/profile.d/bash_completion.sh" >> ~/.bashrc`
 - `exit`
 - `sudo su`

Learning Environment

- ❖ Windows machine
 - Use the **Vagrantfile** provided for this
 - Create a folder on the desktop with name windows
 - Put the Vagrantfile referencing windows in this directory
 - now from powershell go to this windows directory and hit
 - **vagrant up**



Ansible Installation

Ansible Installation

❖ Dependencies

- `sshpass`
- `python`
- `openssl`

❖ Commands for Ansible Installation

- `sudo apt update`
- `sudo apt install software-properties-common sshpass -y`
- `sudo apt-add-repository --yes --update ppa:ansible/ansible`
- `sudo apt install ansible openssh-server -y`
- `ansible --version`

Ansible Configuration

- ❖ configuration files
 - `/etc/ansible/ansible.cfg`
- ❖ default inventory file
 - `/etc/ansible/hosts`

Ansible Configuration

- ❖ Inventory file location
 - `vim /etc/ansible/hosts`
- ❖ Ansible configuration file location
 - `vim /etc/ansible/ansible.cfg`
- ❖ Disabling strict host key checking
 - `vim /etc/ansible/ansible.cfg`
 - # uncomment this to disable SSH key host checking
 - `host_key_checking = False`

Ansible Configuration Parameters

- ❖ `hostname`
 - the hostname of the remote machine
- ❖ `ansible_ssh_host`
 - the ip or domain of the remote host
- ❖ `ansible_port`
 - the port of the remote host which is usually 22
- ❖ `ansible_connection`
 - the connection where we set, we want to connect with ssh
- ❖ `ansible_user`
 - the ssh user

Ansible Configuration Parameters

- ❖ `ansible_ssh_extra_args`
 - extra arguments what we want to specify for the ssh

Ansible Modules

- ❖ units which gets the work done
- ❖ works like:
 - creating file
 - setting cron jobs
 - issuing shell commands
 - executing remote scripts
 - tar/untar files
 - unzip operations
 - package installations etc.
- ❖ Ansible has built in module library for day to day use cases

Ansible Module Examples

- ❖ file
- ❖ cron
- ❖ shell
- ❖ command
- ❖ script
- ❖ copy
- ❖ template
- ❖ unarchive
- ❖ lineinfile
- ❖ user
- ❖ group

Ansible Modules

- ❖ List of available Ansible Modules
 - https://docs.ansible.com/ansible/latest/modules/modules_by_category.html

Ansible Ad Hoc Commands

- ❖ Ansible Ad-Hoc commands are used to accomplish tasks quickly
- ❖ These commands are mostly used for one-off tasks
- ❖ Ad-Hoc commands are handy to get small tasks done quickly

Ansible Ad Hoc Commands

- ❖ `ansible -m ping all`
- ❖ `ansible -m ping all -u vagrant -k -K`
- ❖ `ansible -m ping web`
- ❖ `ansible -m ping web -u vagrant -k -K`
- ❖ `ansible -m shell all -a whoami -u vagrant -k -K`
- ❖ `ansible -bm shell all -a whoami -u vagrant -k -K`
- ❖ `ansible -m command all -a whoami -u vagrant -k -K`
- ❖ `ansible -b -m user -a 'name=admin' db -u vagrant -k -K`
- ❖ `ansible -b -m apt -a 'name=tree' web -u vagrant -k -K`
- ❖ `ansible -b -m apt -a 'name=nginx' web -u vagrant -k -K`
- ❖ `ansible -b -m service -a 'name=nginx state=started' all -u vagrant -k -K`

Exercises

- ❖ Try to create the below tasks
 - Create a user with your name and verify it
 - Create a directory
 - Create a file
 - Install a package tree
 - Execute a remote command
 - Stop Nginx and verify from the server

Solution

- ❖ Create a user with your name and verify it
 - `ansible -bm user all -u vagrant -K -k -a 'name=admin shell=/bin/bash'`
- ❖ Create a directory
 - `ansible -bm file all -u vagrant -K -k -a 'path=/tmp/dir state=directory'`
- ❖ Create a file
 - `ansible -bm file all -u vagrant -K -k -a 'path=/tmp/dir/myfile state=touch'`
- ❖ Install a package tree
 - `ansible -bm package all -u vagrant -K -k -a 'name=tree state=present'`
- ❖ Execute a remote command
 - `ansible -bm command all -a uptime -u vagrant -k -K`

Solution

❖ Stop Nginx

➤ `ansible -b -m service -a 'name=nginx state=started' all -u vagrant -k -K`





YAML Introduction

YAML Syntax

- ❖ YAML Ain't Markup Language
- ❖ More human readable than XML or JSON
- ❖ Parsers are commonly available (yaml-lint)
- ❖ Ansible uses a subset of YAML in a specific way
- ❖ All documents begin with `---`
- ❖ Ansible: only one YAML document per file (the YAML spec allows more)
- ❖ *almost* all Ansible YAML files start with a list (`- [key]`)
- ❖ Indentation is key
- ❖ .yaml or .yml extension

YAML Syntax

❖ Primary elements are

- Lists
- Hashes
- Booleans

YAML Example

```
---  
# This Playbook would deploy the whole mongodb cluster with replication and  
sharding.  
- hosts: all  
  roles:  
    - role: common  
- hosts: mongo_servers  
  roles:  
    - role: mongod  
- hosts: mongoc_servers  
  roles:  
    - role: mongoc  
- hosts: mongos_servers  
  roles:  
    - role: mongos  
- hosts: mongo_servers  
tasks:  
- include: roles/mongod/tasks/shards.yml
```

Lists

- ❖ A series of values
- ❖ Can be long

```
---  
mylist:  
- item1  
- item2  
- item3
```

- ❖ or short form

```
---  
mylist: ['item1', 'item2', 'item3']
```


Hash or Dictionary

- ❖ Key-Value pairs
- ❖ Can be long

employees:

- dave:

name: Davey Jones

job: Sailor

location: In locker

- ❖ or short form

employees:

- dave: {name: Davey Jones, job: Sailor, location: In locker}

Yaml Lint

- ❖ syntax checker for yaml
- ❖ Simple to install
- ❖ Easy to use
- ❖ light-weight alternative to the Ansible parser
- ❖ Installation
 - `sudo apt-get install yamllint`
- ❖ Usage
 - `yamllint myplaybook.yml`

Yaml Exercise

- ❖ `touch abc.yml`
- ❖ `yamllint abc.yml`
- ❖ `echo 'Demo YAML' > abc.yml`
- ❖ `yamllint abc.yml`



That's All for Day 1



Inventory File

Inventory Files

- ❖ Inventory file for ansible can be located on this location
 - `/etc/ansible/hosts`
- ❖ Rules of this hosts file
 - Comments begin with the '#' character
 - Blank lines are ignored
 - Groups of hosts are delimited by [header] elements
 - we can enter hostnames or ip addresses
 - hostname/ip can be a member of multiple groups

Inventory Files

- ❖ Individual hosts, Ungrouped hosts, specify before any group headers

green.example.com

blue.example.com

192.168.100.1

192.168.100.10

- ❖ Grouping of hosts – collection of hosts belonging to the 'webservers' group

[webservers]

alpha.example.org

beta.example.org

192.168.1.100

192.168.1.110

Inventory Files

- ❖ have multiple hosts following a pattern? specify them like this

`www[001:006].example.com`

`db-[99:101]-node.example.com`

- ❖ A collection of database servers in the 'dbservers' group

`[dbservers]`

`db01.intranet.mydomain.net`

`db02.intranet.mydomain.net`

`10.25.1.56`

`10.25.1.57`

Inventory Files

❖ Group of Groups

[childgroup2]

host1

host2

[childgroup1]

host2

host3

[parent1:children]

childgroup1

childgroup2

More Examples Inventory Files

```
[web]  
mastery.example.name ansible_host=192.168.10.25
```

```
[dns]  
backend.example.name
```

```
[database]  
backend.example.name
```

```
[frontend:children]  
web
```

```
[backend:children]  
dns  
database
```

```
[web:vars]  
http_port=88  
proxy_timeout=5
```

```
[backend:vars]  
ansible_port=314
```

```
[all:vars]  
ansible_ssh_user=otto
```

Dynamic Inventory

- ❖ Ansible inventory fluctuates over time?
- ❖ hosts spinning up and shutting down in response to business demands?
- ❖ `/etc/ansible/hosts` – static inventory will not serve our needs in this case
- ❖ tracking hosts from multiple sources?
 - cloud providers
 - LDAP
 - Cobbler
 - enterprise CMDB systems
- ❖ Ansible integrates to all of these via a dynamic external inventory system

Dynamic Inventory

- ❖ Ansible supports two ways to connect with external inventory
 - Inventory Plugins
 - can be enabled from the `ansible.cfg` file
 - `aws_ec2`, `openstack` etc.
 - inventory scripts
- ❖ plugins recommended over scripts for dynamic inventory
- ❖ ok to write our own plugin to connect to other dynamic inventory sources

Dynamic Inventory

- ❖ Ansible supports two ways to connect with external inventory
 - Inventory Plugins
 - can be enabled from the `ansible.cfg` file
 - `aws_ec2`, `openstack` etc.
 - inventory scripts
- ❖ plugins recommended over scripts for dynamic inventory
- ❖ ok to write our own plugin to connect to other dynamic inventory sources

Dynamic Inventory

- ❖ When ansible or ansible-playbook is directed at an executable file for an inventory source, Ansible will execute that script with a single argument, `--list`
- ❖ allows Ansible to get a listing of the entire inventory in order to build up its internal objects to represent the data
- ❖ Once that data is built up, Ansible will then execute the script with a different argument for every host in the data to discover variable data
- ❖ The argument used in this execution is `--host <hostname>`, which will return any variable data specific to that host

Dynamic Inventory

❖ Cobbler

- <https://github.com/hellodk/ansible-provider-docs/tree/master/content/inventory>
- copy `cobbler.py` to `/etc/ansible` and `chmod +x` the file
- Run `cobblerd` any time we use Ansible
- to communicate with Cobbler using Cobbler's XMLRPC API use the `-i` command line option
 - `ansible-playbook -i /etc/ansible/cobbler.py`
- Add a `cobbler.ini` file in `/etc/ansible` so Ansible knows where the Cobbler server is and some cache improvements can be used

Dynamic Inventory

- ❖ `/etc/ansible/cobbler.ini`

```
[cobbler]
```

```
# Set Cobbler's hostname or IP address
```

```
host = http://127.0.0.1/cobbler_api
```

```
cache_path = /tmp
```

```
cache_max_age = 900
```

- ❖ EC2

- `ansible -i ec2.py -u ubuntu us-east-1d -m ping`

- ❖ References

- https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html

Dynamic Inventory

- ❖ Custom Scripts
 - `python3 get_inventory.py`
- ❖ Ensure `get_inventory.py` is executable
 - `cp get_inventory.py /home/vagrant`
 - `cd /home/vagrant`
 - `chmod +x get_inventory.py`
 - `ls -ltr`
- ❖ Run ansible and pass this dynamic inventory
 - `ansible all -i get_inventory.py -m ping -k -K`

Inventory Exercises

- ❖ Add your servers in inventory with the below specifications
 - [main] group comprising of anisblem
 - [slaves] group comprising of ansibles1, ansibles2, ansibles3
 - [windows] group with ip 192.168.10.35 (we will use it later)

Inventory Exercises Solution

- ❖ Add your servers in inventory with the below specifications

- [main] group comprising of anisblem

```
[main]
```

```
192.168.10.30
```

- [slaves] group comprising of ansibles1, ansibles2, ansibles3

```
[slaves]
```

```
192.168.10.30
```

```
192.168.10.30
```

```
192.168.10.30
```

- [windows] group with ip 192.168.10.35 (we will use it later)

```
[windows]
```

```
192.168.10.30
```



Introduction to Tasks and Playbooks

Tasks

- ❖ Individual piece of a job
- ❖ Example
 - creating a file
 - starting a service etc.
- ❖ 2 ways to implement tasks
- ❖ to create a user:
 - `ansible -b -K -m user -a 'name=admin' db -u vagrant -k -K`
or
 - `- name: Add the user 'admin' # This is the title`
`user: # This is the module name`
`name: admin # These are the list of supported parameters`

Playbooks

- ❖ A collection of tasks along with host definitions

```
---  
- hosts: 192.168.122.196  
  become: yes  
  remote_user: vagrant  
  tasks:  
    - name: "Create demo group"  
      group:  
        name="demo"  
        state="present"
```

Executing Playbooks

- ❖ Syntax
 - `ansible-playbook <playbook name>`
- ❖ Creating a group
 - `ansible-playbook create_group.yml`
- ❖ Creating a user
 - `ansible-playbook create_user.yml`
- ❖ Create a file
 - `ansible-playbook create_file.yml`
- ❖ Create a directory
 - `ansible-playbook create_directory.yml`

Executing Playbooks with Verbosity

❖ Verbosity levels

➤ v

➤ vv

➤ vvv

➤ vvvv

❖ Create a directory

➤ `ansible-playbook create_directory.yml -v`

➤ `ansible-playbook create_directory.yml -vv`

➤ `ansible-playbook create_directory.yml -vvv`

➤ `ansible-playbook create_directory.yml -vvvv`

Including Playbooks

- ❖ Include multiple plays into a single playbook
 - `ansible-playbook all.yml`
 - `ansible-playbook all.yml -v`

Problem with this Playbook

- ❖ Issues
 - Fixed group name
 - Fixed host name
- ❖ Solution
 - Variables

Variables in Ansible

Variables

- ❖ May be defined through:
 - Playbooks
 - External YAML
 - Facter
 - Command line
 - Host and Group directories
- ❖ Must only consist of:
 - letters
 - numbers
 - underscores

Variables in Inventory Files

❖ `cat /etc/ansible/hosts`

```
[webservers]
```

```
web-01 http_port=80 maxRequests=200
```

```
web-02 http_port=80 maxRequests=200
```

```
[webservers:vars]
```

```
ntp_server=0.centos.pool.ntp.org
```

```
proxy=webproxy
```

Variables in Facter

❖ `ansible all -m setup`

```
host-10-23-5-21 | SUCCESS => {  
  "ansible_facts": {  
    "ansible_all_ipv4_addresses": [  
      "10.23.5.21"  
    ],  
    "ansible_all_ipv6_addresses": [  
      "fe80::f816:3eff:fef7:6e80"  
    ],  
    "ansible_architecture": "x86_64",
```

Variables in Playbook

- name: Deploy our webserver

hosts: webserver

vars:

http_port: 80

tasks:

- name: deploy apache

Variables in External YAML

```
# The Playbook
```

```
---
```

```
- name: Deploy our webserver
```

```
hosts: webserver
```

```
vars:
```

```
http_port: 80
```

```
vars_files: /vars/external_vars.yml
```

```
tasks:
```

```
- name: deploy apache
```


Variables in External YAML

The YAML file

myvariable1: myvalue1

myvariable2: myvalue2

foo:

subkey1: one

subkey2: two

Variables in Modules in External File

```
# The Playbook
```

```
---
```

```
- name: Deploy our webserver
```

```
hosts: webserver
```

```
tasks:
```

```
- name: include default step variables
```

```
include_vars: default_step.yml
```

Variables in Modules in External File

```
# The YAML file
```

```
---
```

```
myvariable1: myvalue1
```

```
myvariable2: myvalue2
```

```
foo:
```

```
  subkey1: one
```

```
  subkey2: two
```

Variables in split-out yaml/json files

- ❖ `/etc/ansible/host_vars`
- ❖ `/etc/ansible/group_vars`

Variables in Command line

- ❖ `ansible-playbook release.yml --extra-vars "version=1.23.45
other_variable=foo"`
- ❖ `ansible-playbook release.yml --extra-vars
'{"pacman":"mrs","ghosts":["inky","pinky","clyde","sue"]}'`
- ❖ `ansible-playbook release.yml --extra-vars "@some_file.json"`
- ❖ In any section, redefining a variable will overwrite the previous instance
- ❖ If multiple groups have the same variable, the last one loaded wins

Using Variables in Playbooks

tasks:

- debug: msg="System {{ inventory_hostname }} has gateway {{

- ansible_default_ipv4.gateway }}"

when: ansible_default_ipv4.gateway is defined

- name: Deploy my file

template: src=foo.cfg.j2 dest={{ remote_install_path }}/foo.cfg

Using Variables in Templates

Port {{ ssh_port }}

Protocol {{ ssh_protocol }}

HostKey /etc/ssh/ssh_host_rsa_key

HostKey /etc/ssh/ssh_host_dsa_key

UsePrivilegeSeparation yes

KeyRegenerationInterval 3600

ServerKeyBits 1024

SyslogFacility AUTH

LogLevel INFO

LoginGraceTime 120

PermitRootLogin {{ ssh_permit_root_login }}

StrictModes yes

...

Flexible Playbook with variables

- ❖ copy the previous file into a new file – flexible-playbook.yaml

```
---  
- name: My playbook  
  hosts: '{{ hosts }}'  
  remote_user: '{{ user }}'  
  tasks:.....
```

- ❖ Execution command

➤ `ansible-playbook flexible-playbook.yaml --extra-vars "hosts=web
user=vagrant"`

Variable Scopes

- ❖ determines where the variable is valid
- ❖ 3 main scopes
 - Global
 - Set by config, environment variables and the command line
 - Play
 - Each play and its contained structures, vars entries, include_vars, role defaults and vars
 - Host
 - Variables directly associated to a host
 - eg: inventory, facts or registered task outputs

Variable Scopes

❖ Nested data structures are accessed slightly differently:

➤ `{{ ansible_eth0["ipv4"]["address"] }}`

OR alternatively:

➤ `{{ ansible_eth0.ipv4.address }}`

❖ To access the first element of an array:

➤ `{{ foo[0] }}`

Variable Precedence

- ❖ role defaults (least important)
- ❖ inventory vars [2]
- ❖ inventory group_vars
- ❖ inventory host_vars
- ❖ playbook group_vars
- ❖ playbook host_vars
- ❖ host facts
- ❖ registered vars
- ❖ set_facts
- ❖ play vars

Variable Precedence

- ❖ `play vars_prompt`
- ❖ `play vars_files`
- ❖ `role` and `include vars`
- ❖ `block vars` (only for tasks in block)
- ❖ `task vars` (only for the task)
- ❖ `extra vars` (most important)

Splitting out Variables

- ❖ Host and Group variables can be stored in individual files relative to inventory file
 - `host_vars`
 - `group_vars`
- ❖ `ls /etc/ansible/host_vars`
 - `webserver1.yaml`
 - `webserver2.yaml`
- ❖ `ls /etc/ansible/group_vars`
 - `dbservers.yaml`
 - `webservers.yaml`
 - `all_infrastructure.yaml`

Splitting out Variables

- ❖ You can also create directories named after your groups or hosts.
- ❖ All files in the directories will be read
 - `/etc/ansible/group_vars/webserver/db_settings`
 - `/etc/ansible/group_vars/webserver/cluster_settings`
- ❖ the `group_vars/` and `host_vars/` directories can exist in either the playbook directory OR the inventory directory
- ❖ If both paths exist, variables in the playbook directory will override variables set in the inventory directory

Splitting out Variables

- ❖ All group variables must be defined in a `group_vars` area
- ❖ All host variables must be defined in a `host_vars` area

Exercises

- ❖ Have an entry like this in your inventory file

[web]

192.168.10.71

- ❖ `ansible web -m ping`
- ❖ `ansible web -m command -a "uptime" -o`
- ❖ `ansible web -m setup`
- ❖ `sudo yum list available subversion.x86_64 --showduplicates`
- ❖ `ansible web -m package -a "name=subversion-1.7.14-11.el7_4 state=present" -b`
- ❖ `ansible web -m package -a "name=subversion state=latest" -b`
- ❖ `ansible web -m package -a "name=* state=latest" -b`
- ❖ `ansible web -m yum -a "name=* security=yes state=latest" -b`

Exercises

- ❖ `ansible web -m package -a "name=rh-maven35 state=present" -b`
- ❖ `ansible web -m package -a "name=git state=present" -b`
- ❖ `ansible web -m package -a "name=@jboss-eap7 state=present" -b`
- ❖ `ansible web -m service -a "name=eap7-standalone state=started" -b`
- ❖ `ansible web -m git -a "repo=https://github.com/jboss-developer/jboss-eap-quickstarts.git dest=/tmp/checkout"`
- ❖ `ansible web -m shell -a "scl enable rh-maven35 'mvn clean install wildfly:deploy -Dmaven.test.skip=true' chdir=/tmp/checkout/helloworld" -b`
- ❖ `ansible web -m uri -a "url=http://localhost:8080/helloworld/ return_content=yes"`

Exercises

- ❖ Demo site (open this in your web browser):
 - <http://192.168.10.70:8080/helloworld/>
- ❖ `ansible web -m service -a "name=eap7-standalone state=stopped" -b`
- ❖ `ansible web -m package -a "name=@jboss-eap7 state=absent" -b`
- ❖ `ansible web -m package -a "name=eap7-* state=absent" -b`
- ❖ `ansible web -m package -a "name=rh-maven35 state=absent" -b`
- ❖ `ansible web -m package -a "name=git state=absent" -b`

Exercises

- ❖ Execute the playbook

- `playbook1.yaml`

Prompt

- hosts: all

remote_user: vagrant

vars_prompt:

- name: "name"

prompt: "what is your name?"

- name: "quest"

prompt: "what is your quest?"

- name: "favcolor"

prompt: "what is your favorite color?"

Playbook Exercises

- ❖ write a playbook to accomplish the following tasks with user input, group input and host name input
 - Create a user with name admin, group admin
 - Create a folder
 - mydir
 - Create a file
 - myfile
 - Create a symlink to the file myfile
 - mysymlink
 - Copy a file from your local system to the directory mydir

Variable Exercises

- ❖ Write a playbook to print all the variables
- ❖ Write a playbook to create a user where user is a variable
- ❖ Write a playbook to prompt user for details

Variable Exercises Solution

- ❖ Write a playbook to print all the variables
 - `ansible-playbook debug_module_usage.yml`
- ❖ Write a playbook to create a user where user is a variable
 - `ansible-playbook create_user.yml`
- ❖ Write a playbook to prompt user for details
 - `ansible-playbook vars_prompt.yml`

SSH Based Authentication

- ❖ Generate ssh keys if not already present
 - `ssh-keygen`
 - `ls -ltr ~/.ssh`
- ❖ copy the public keys of your centralized server
 - `cat /home/vagrant/.ssh/id_rsa.pub`
- ❖ put the same public keys in the authorized keys of the remote machines
 - `vim /home/vagrant/.ssh/authorized_keys`
- ❖ verify the same by doing a ssh from the central machine to the remote
 - `ssh vagrant@192.168.10.31`
- ❖ You should be able to connect automatically without password

SSH Based Authentication Exercise

- ❖ copy the public keys of your centralized server to all the remote servers
 - source: `/home/vagrant/.ssh/id_rsa.pub`
 - destination: `/home/vagrant/.ssh/authorized_keys`
- ❖ verify the same by doing a ssh from the central machine to the remote
 - `ssh vagrant@192.168.10.30`
 - `ssh vagrant@192.168.10.31`
 - `ssh vagrant@192.168.10.32`
 - `ssh vagrant@192.168.10.33`
- ❖ You should be able to connect automatically without password

SSH Based Authentication Solution

- ❖ copy the public keys of your centralized server to all the remote servers
 - source: `/home/vagrant/.ssh/id_rsa.pub`
 - destination: `/home/vagrant/.ssh/authorized_keys`
 - `ansible-playbook generate_ssh_keys.yml`
- ❖ verify the same by doing a ssh from the central machine to the remote
 - `ssh vagrant@192.168.10.30`
 - `ssh vagrant@192.168.10.31`
 - `ssh vagrant@192.168.10.32`
 - `ssh vagrant@192.168.10.33`
- ❖ You should be able to connect automatically without password



Controlling Play

Execution

Tags

```
tasks:

  - yum: name={{ item }} state=installed
    with_items:
      - httpd
      - memcached
    tags:
      - packages

  - template: src=templates/src.j2 dest=/etc/foo.conf
    tags:
      - configuration
```


Tags

- ❖ `ansible-playbook playbook1.yml --tags "configuration,packages"`
- ❖ `ansible-playbook playbook1.yml --skip-tags "notification"`
- ❖ `ansible-playbook playbook1.yml --step`
- ❖ `ansible-playbook playbook1.yml --check`

Play Information

- ❖ list all tasks that would be executed by a play without making changes
 - `ansible-playbook playbook1.yml --list-tasks`
- ❖ list all hosts that would be affected by a play, without running any tasks
 - `ansible-playbook playbook1.yml --list-hosts`
- ❖ use tags to limit the execution of a play
 - `ansible-playbook playbook1.yml --list-tags`

Controlling Play Execution

- ❖ use option `--start-at-task` to define a new entry point for our playbook
- ❖ Ansible will skip anything that comes before the specified task
- ❖ requires a valid task name as argument:
 - `ansible-playbook playbook1.yml --start-at-task="Set Up Nginx"`
- ❖ only execute tasks associated with specific tags, use the option `--tags`
 - `ansible-playbook playbook1.yml --tags=mysql,nginx`
- ❖ skip all tasks that are under specific tags, use `--skip-tags`
 - `ansible-playbook playbook1.yml --skip-tags=mysql`

Maximum Failure Percentage

- ❖ By default, Ansible will continue executing actions as long as there are hosts in the group that have not yet failed
- ❖ For situations such as rolling updates, it may be desirable to abort the play when a certain threshold of failures have been reached
- ❖ We do this with "`max_fail_percentage`"

```
- hosts: webservers
  max_fail_percentage: 30
  serial: 10
```


Delegation

- ❖ want to perform a task on one host with reference to other hosts?
- ❖ use the 'delegate_to' keyword

- hosts: webserver

serial: 5

tasks:

- name: take out of load balancer pool

command: /usr/bin/take_out_of_pool {{ inventory_hostname }}

delegate_to: 10.0.0.65

- name: actual steps would go here

yum: name=acme-web-stack state=latest

- name: add back to load balancer pool

command: /usr/bin/add_back_to_pool {{ inventory_hostname }}

delegate_to: 10.0.0.65

Localhost Delegation

- ❖ delegate to localhost
- ❖ There is a shorthand for this: "local_action"

tasks:

- name: take out of load balancer pool

local_action: command /usr/bin/take_out_of_pool {{ inventory_hostname }}

- name: add back to load balancer pool

local_action: command /usr/bin/add_back_to_pool {{ inventory_hostname }}

Error Handling

- ❖ Ignoring failed commands
- ❖ Ensuring Handler behaviour
- ❖ Specifying Failure States
- ❖ Aborting the Play
- ❖ Blocks

Error Handling

❖ Ignoring Errors

– name: this will not be counted as a failure

command: /bin/false

ignore_errors: yes

❖ Forcing handlers

– name: this will not be counted as a failure

command: /bin/false

force_handlers: True

❖ Failed in

– name: this command prints FAILED when it fails

command: /usr/bin/example-command -x -y -z

register: command_result

failed_when: "'FAILED' in command_result.stderr"

Accelerated Mode

- ❖ Launches a daemon on port 5099 for 30 minutes
- ❖ Requires Python 2.5+ on the Controller
- ❖ Significantly faster than Paramiko
- ❖ In a Play:
 - hosts: all
 - accelerate: true
 - tasks:
 - name: some task
- ❖ Options available in `ansible.cfg`

```
[accelerate]
accelerate_port = 5099
accelerate_timeout = 30
accelerate_connect_timeout = 5.0
accelerate_multi_key = yes
```

Asynchronous Actions

```
---  
- hosts: all  
  remote_user: root  
  tasks:  
  
  - name: simulate long running op, wait 45 sec, poll every 5 sec  
    command: /bin/sleep 15  
    async: 45  
    poll: 5
```

Exercises

- ❖ Convert `install_docker.sh` to an Ansible playbook



That's All for Day 2



Facts in Ansible

Facts

- ❖ System information provided by "Facter"
- ❖ Gathered as a discrete step during a run
- ❖ Accessible via the "setup" module
- ❖ An extensible framework
 - Custom Facts
 - Local Facts
- ❖ Fact gathering can be disabled via the "gather_facts" setting:
 - – hosts: whatever
 - gather_facts: no
- ❖ You may wish to do this for performance

Facts

- ❖ `ansible -m setup main -a filter=*ipv4`
- ❖ Display facts from all hosts and store them
 - `ansible all -m setup --tree /tmp/facts`
- ❖ Display only facts regarding memory
 - `ansible all -m setup -a 'filter=ansible_*_mb'`
- ❖ Display only facts returned by facter.
 - `ansible all -m setup -a 'filter=facter_*'`
- ❖ Collect only facts returned by facter
 - `ansible all -m setup -a 'gather_subset=!all,facter'`

Facts

❖ Available gather_subset are

- all
- min
- hardware
- network
- virtual
- ohai
- facter

Local Facts

- ❖ Also known as "External Facts"
- ❖ Loaded from `/etc/ansible/facts.d` from every remote server – hence local facts
- ❖ Must end in `.fact`
- ❖ Must either contain JSON
- ❖ OR be executable and return JSON
- ❖ Return as part of the "ansible_local" fact
- ❖ local fact file returning JSON:
 - `cat /etc/ansible/facts.d/cluster.fact`
`{ "state": "green", "backup": "complete" }`
 - `ansible -m setup -a 'filter=ansible_local' main`

Filtering Facts

❖ want to re-read facts after deploying a new one – **localfacts.yaml**

– hosts: webservers

tasks:

– name: create directory for ansible custom facts

file: state=directory recurse=yes path=/etc/ansible/facts.d

– name: install custom impi fact

copy: src=ipmi.fact dest=/etc/ansible/facts.d

– name: re-read facts after adding custom fact

setup: filter=ansible_local

Facts

- ❖ `ansible -m setup all -l 192.168.10.30 -a "filter=ansible_python.executable"`
- ❖ `ansible -m setup all -a "filter=ansible_local"`

Fact Caching

- ❖ allows to view Facts
 - from previous runs
 - from other Hosts
- ❖ possible for one server to reference variables about another
 - `{{ hostvars['asdf.example.com']['ansible_os_family'] }}`
- ❖ save facts between playbook runs
- ❖ feature must be manually enabled
- ❖ useful in very large infrastructure with thousands of hosts
- ❖ Fact caching could be configured to run nightly

Fact Caching

- ❖ allows to view Facts
 - from previous runs
 - from other Hosts
- ❖ possible for one server to reference variables about another
 - `{{ hostvars['asdf.example.com']['ansible_os_family'] }}`
- ❖ save facts between playbook runs
- ❖ feature must be manually enabled
- ❖ useful in very large infrastructure with thousands of hosts
- ❖ Fact caching could be configured to run nightly

Fact Caching

- ❖ Ansible ships with two persistent cache plugins
 - redis
 - jsonfile
- ❖ get redis up and running, perform the equivalent OS commands:
 - yum install redis
 - service redis start
 - pip install redis
- ❖ Redis plugin does not support port or password configuration

Fact Caching

- ❖ enable fact caching via redis by enabling it in ansible.cfg as follows:

```
[defaults]
```

```
gathering = smart
```

```
fact_caching = redis
```

```
fact_caching_timeout = 86400
```

```
# seconds
```

Fact Caching

- ❖ configure Fact Caching with a jsonfile:

```
[defaults]
```

```
gathering = smart
```

```
fact_caching = jsonfile
```

```
fact_caching_connection = /home/vagrant
```

```
fact_caching_timeout = 86400
```

```
# seconds
```

- ❖ Execute some playbooks

- `ansible-playbook local-facts.yml`

- `ls -ltr ~`

- `ansible -m setup all -a "filter=ansible_local"`

- `ls -ltr ~`

Set Facts

❖ `ansible-playbook set_facts`



Templating in Ansible

Jinja2 Templates

- ❖ Ansible uses the Jinja2 templating system
- ❖ All variables in-scope are available inside a template
 - `ansible-playbook local-facts-template.yml`
- ❖ Templates are essential for managing services that vary their configurations

Printing Variables

```
---  
- hosts: 127.0.0.1
```

```
  sudo: yes
```

```
  remote_user: runuser
```

```
  tasks:
```

```
    # Example that prints the loopback address and gateway for each host
```

```
    - debug: msg="System"
```

```
    - debug: msg="System {{ inventory_hostname }} has gateway {{  
ansible_default_ipv4.gateway }}"
```

```
      when: "ansible_default_ipv4.gateway is defined"
```

```
    - shell: /usr/bin/uptime
```

```
      register: result
```

```
    - debug: var=result verbosity=2
```

```
    - name: Display all variables/facts known for a host
```

```
      debug: var=hostvars[inventory_hostname] verbosity=4
```


Magic Variables

- ❖ few additional variables – cannot be set directly by the user
- ❖ Ansible will always override them to reflect internal state
 - `hostvars`
 - `group_names`
 - `groups`
- ❖ `hostvars`
 - Ask about the variables of another host
 - `{{ hostvars['test.example.com']['ansible_distribution'] }}`
- ❖ `group_names`
 - list of all the Groups the Host is in
- ❖ `groups`
 - list of all the Groups and their Hosts in the Inventory

Magic Variables

- ❖ Using group_names in template logic:
 - `{% if 'webserver' in group_names %}`
`# some part of a configuration file that only applies to webserver`
`{% endif %}`
- ❖ Using groups in template logic:
 - `{% for host in groups['slaves'] %}`
`# something that applies to all app servers.`
`{% endfor %}`

Magic Variables

❖ Finding all IP addresses in a Group:

➤ `{% for host in groups['slaves'] %}`

`{{ hostvars[host]['ansible_eth0']['ipv4']['address'] }}`

`{% endfor %}`

Few more Magic Variables

- ❖ `inventory_hostname`: hostname as configured in Ansible's inventory file
- ❖ `play_hosts`: list of hostnames that are in scope for the current play
- ❖ `inventory_dir`: Pathname of the directory holding Ansible's inventory file
- ❖ `inventory_file`: pathname and the filename pointing to Ansible's inventory file

Registered Variables

- ❖ major use of variables is running a command, saving the result, and performing actions with that variable – "Registering a Variable"

– hosts: web_servers

tasks:

– shell: /usr/bin/foo

register: foo_result

ignore_errors: True

– shell: /usr/bin/bar

when: foo_result.rc == 5

- ❖ Registered variables have the same scope and lifetime as Facts

Exercises

- ❖ Using the Ansible debug statement print the following:
 - `hostvars`
 - `group_names`
 - `groups`
 - `inventory_hostname`
 - `play_hosts`
 - `inventory_dir`
 - `inventory_file`
 - `role_path`
- ❖ Use register to register the output of a task and print it

Exercises

❖ `ansible-playbook template_example.yml -e "id=1"`



Conditionals & Filters in Ansible

Conditionals

- ❖ "when" is the conditional keyword:

tasks:

- name: "shutdown Debian flavored systems"

command: /sbin/shutdown -t now

when: ansible_os_family == "Debian"

- ❖ use parentheses to group conditions:

```
tasks:
```

```
- name: "shutdown CentOS 6 and Debian 7 systems"
```

```
  command: /sbin/shutdown -t now
```

```
  when: (ansible_distribution == "CentOS" and ansible_distribution_major_version == "6") or  
        (ansible_distribution == "Debian" and ansible_distribution_major_version == "7")
```

Conditionals

- ❖ Booleans can also be checked:

tasks:

- shell: echo "Executing Action!"

when: myvariable

- ❖ Jinja2 contains tests that combine well with "when":

tasks:

- shell: echo "I've got '{{ foo }}' and am not afraid to use it!"

when: foo is defined

- fail: msg="Bailing out. this play requires 'bar'"

when: bar is undefined

Conditionals

- ❖ As do registered variables

```
- name: test play
  hosts: all

  tasks:

    - shell: cat /etc/motd
      register: motd_contents

    - shell: echo "motd contains the word hi"
      when: motd_contents.stdout.find('hi') != -1
```

Conditionals Exercises

❖ `ansible-playbook conditionals.yml -e 'myvariable=true'`



Ansible Galaxy & Roles

Ansible Galaxy

- ❖ Ansible Galaxy is a free site for finding, downloading, rating, and reviewing all kinds of community developed Ansible Roles
- ❖ `ansible-galaxy` is a command-line tool for managing and creating Roles
- ❖ `ansible-galaxy search mysql`
- ❖ `ansible-galaxy search --server https://galaxy-qa.ansible.com mysql --author geerlingguy`
- ❖ `ansible-galaxy info username.role_name`
- ❖ `ansible-galaxy install username.role_name`

Ansible Galaxy

- ❖ `ansible-galaxy list`
- ❖ `ansible-galaxy remove username.rolename`
- ❖ `ansible-galaxy init rolename`

Ansible Configurations

- ❖ fork
- ❖ roles_path

Roles

- ❖ structured set of Playbooks, Handlers,
- ❖ Files, Variables, and Templates, that fulfil a defined purpose
- ❖ Problem
 - Playbooks can include plays from other playbooks, and handlers and variables from elsewhere
- ❖ Solution
 - Roles are an abstraction to cover everything
 - We bundle everything into a structured environment

Roles

- ❖ Roles provide the following benefits:
 - Defined scope of functionality
 - Promote re-use of configuration steps
 - Should represent business logic (e.g. Deploy a new webserver, or application layer)
- ❖ Community roles are available in the "ansible-galaxy"

Roles Construction

- ❖ `ansible-galaxy init demo`
- ❖ `tree demo`

Roles Folder Structure

- ❖ Defaults
 - main.yaml
 - all default values to be used by the roles
- ❖ Files
 - contains all files used in the role, and referred by copy module
- ❖ Handlers
 - place for all the handlers
- ❖ Meta
 - metadata like authors, dependencies, date, versions etc.
- ❖ Tasks
 - Playbooks used by the role, main.yml is the entry point

Roles Folder Structure

- ❖ Templates
 - files to be modified at runtime, .j2 extension in ideal scenarios
 - powered by Jinja2 templating engine
- ❖ Vars
 - another place for variables
 - more priority than default variables

Parameterized Includes

- ❖ Created when you pass parameters to an include:

tasks:

- include: core_app.yml admin=timmy

- include: core_app.yml admin=alice

- include: core_app.yml admin=bob

- include: core_app.yml

vars:

admin: sharon

ssh_keys:

- keys/one.txt

- keys/two.txt

Parameterized Includes

- ❖ Variables are used as normal within Tasks and Templates:

- name: Deploying core_app

 - yum: name=core_app state=present

- name: Configuration

 - copy: src=adminfile dest="/home/{{ admin }}/keyfile"

Using Roles

- ❖ Inside of our top-level Playbook, we need very little information
 - define hosts
 - declare the role we are using

```
---  
- hosts: wordpress_hosts  
roles:  
  - nginx  
  - php  
  - mysql  
  - wordpress
```


Using Roles

- ❖ Include your Role with the "roles" keyword:

```
---
```

```
- name: deploy-databases
```

```
  hosts: dbusers
```

```
  roles:
```

```
    - sshd
```

```
    - ntpd
```

```
    - { role: mysql, mysql_root_db_pass: foobar, mysql_db: none,
```

```
      mysql_users: none }
```

Parameterized Roles

- ❖ Define defaults in "[role]/defaults/main.yml"

```
---  
sshd_port: 22  
sshd_permit_root_login: true  
sshd_manage_service: "{{ false if ansible_virtualization_type ==  
'docker' else true }}"  
sshd_config_owner: root  
sshd_config_group: root  
sshd_config_mode: "0600"
```

Parameterized Roles

- ❖ Parameters are then passed in from a calling Playbook

- name: deploy-databases

hosts: dbusers

roles:

- { role: sshd, sshd_port: 22 }

- ntpd

- { role: mysql, mysql_root_db_pass: foobar, mysql_db: none, mysql_users:
nobleprog }

Structured Playbooks

- ❖ Playbooks can include other Playbooks
- ❖ This means we can have a master Playbook, and sub-Playbooks

- name: this is a play at the top level of a file

- hosts: all

- remote_user: root

- tasks:

- name: say hi

- tags: foo

- shell: echo "hi..."

- include: load_balancers.yml

- include: webservers.yml

Exercises

- ❖ Role to create Apache Installation
- ❖ User Installation for Ansible
- ❖ Create a role for Java 6 Installation
- ❖ Reference
 - <https://github.com/hellodk/swachalit>



Ansible Vault

Ansible Vault

- ❖ allows keeping sensitive data such as passwords or keys in encrypted files, rather than as plaintext in our playbooks or roles
- ❖ Vault can encrypt:
 - Any structured data file used by Ansible
 - “group_vars/” and “host_vars/”
 - Inventory variables
 - “include_vars” and “vars_files”
 - variable files from the ansible command line: “-e
 - @file.yml” or “-e @file.json”
 - Role variables and defaults are also included
 - Handlers and Tasks

Ansible Vault

- ❖ `ansible-vault create myfile.yml`
- ❖ `ansible-vault edit myfile.yml`
- ❖ `ansible-vault rekey myfile.yml mbar.yml mbaz.yml`
- ❖ `ansible-vault encrypt myfile.yml mbar.yml mbaz.yml`
- ❖ `ansible-vault decrypt myfile.yml mbar.yml mbaz.yml`
- ❖ `ansible-vault view myfile.yml mbar.yml mbaz.yml`
- ❖ `ansible-playbook site.yml --ask-vault-pass`



That's All for Day 3



Managing Windows Server

Setting Things Up

- ❖ Ansible can manage desktop OSs

- Windows 7

- 8.1

- 10

- ❖ server OSs including

- Windows Server 2008

- 2008 R2

- 2012

- 2012 R2

- 2016

- 2019

Setting Things Up

- ❖ On the Ansible controller
 - `sudo pip install pywinrm`
- ❖ To add Kerberos auth to pywinrm, run the following
 - `sudo apt-get install gcc python-dev libkrb5-dev -y`
 - `pip install pywinrm[kerberos]`
- ❖ To add CredSSP auth to pywinrm, run the following
 - `sudo apt-get install gcc python-dev libssl-dev`
 - `pip install pywinrm[credssp]`
- ❖ If on Centos/Redhat then use the below command + the above pip
 - `sudo yum install gcc python-devel krb5-devel krb5-workstation -y`
 - `sudo yum install gcc python-devel openssl-devel -y`

Setting Things Up

- ❖ Ansible requires PowerShell 3.0 or newer
 - verify the powershell version
 - GET-HOST | Select-Object version
- ❖ at least .NET 4.0 to be installed on the Windows host
 - verify the .Net version
- ❖ A WinRM listener should be created and activated
- ❖ two main components of the WinRM service
 - listener
 - service
- ❖ governs how Ansible can interface with the Windows host

Setting Things Up

❖ Setup winrm

- `New-Item -Path C:\Temp -ItemType Directory`
- `$config_script = "C:\Temp\ConfigureRemotingForAnsible.ps1"`
- `(New-Object -TypeName System.Net.WebClient).DownloadFile("https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1", $config_script)`
- `powershell.exe -ExecutionPolicy Bypass -File $config_script -EnableCredSSP`

❖ Verify winrm is listening or not

- `winrm enumerate winrm/config/Listener`

Setting Things Up

- ❖ Another way to Setup winrm
 - `$url =`
`"https://raw.githubusercontent.com/ansible/ansible/devel/examples`
`/scripts/ConfigureRemotingForAnsible.ps1"`
 - `$file = "$env:temp\ConfigureRemotingForAnsible.ps1"`
 - `(New-Object -TypeName System.Net.WebClient).DownloadFile($url,`
`$file)`
 - `powershell.exe -ExecutionPolicy Bypass -File $file`
- ❖ Verify winrm is listening or not
 - `winrm enumerate winrm/config/Listener`

Setting Things Up

- ❖ test if verify WinRM is up and running over a HTTPS listener:
 - `$username = "vagrant"`
 - `$password = ConvertTo-SecureString -String "vagrant" -AsPlainText -Force`
 - `$cred = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $username, $password`
 - `$session_option = New-PSSessionOption -SkipCACheck -SkipCNCheck -SkipRevocationCheck`
 - `Invoke-Command -ComputerName 127.0.0.1 -UseSSL -ScriptBlock { whoami } -Credential $cred -SessionOption $session_option`

Setting Things Up

- ❖ if everything is up & running over a HTTPS listener we get the below output

```
PS C:\Windows\system32> Invoke-Command -ComputerName 127.0.0.1 -UseSSL -ScriptBlock { whoami } -Credential $cred -SessionOption $session_option  
win-i7tmc13v2rr\vagrant  
PS C:\Windows\system32> .
```

Setting Things Up

- ❖ Next is setting up the inventory
- ❖ 2 ways

➤ `w_hosts.yml`

```
windows:
```

```
  hosts:
```

```
    vagrant-windows2012R2:
```

```
      ansible_host: 192.168.10.35
```

```
vars:
```

```
  ansible_user: vagrant
```

```
  ansible_password: vagrant
```

```
  ansible_connection: winrm
```

```
  ansible_port: 5986
```

```
  ansible_winrm_transport: ntlm
```

```
  ansible_winrm_server_cert_validation: ignore
```

Setting Things Up

- ❖ Next is setting up the inventory
- ❖ 2 ways

➤ `w_hosts1.yml`

```
[windows]
```

```
192.168.10.35
```

```
[windows:vars]
```

```
ansible_user="vagrant"
```

```
ansible_password="vagrant"
```

```
ansible_connection="winrm"
```

```
ansible_port="5986"
```

```
ansible_winrm_transport="ntlm"
```

```
ansible_winrm_server_cert_validation="ignore"
```

Setting Things Up

- ❖ Execute ansible commands
 - `ansible -i w_hosts.yml windows -m win_ping`
 - `ansible -i w_hosts1.yml windows -m win_ping`
 - `ansible -i w_hosts.yml windows -m win_ping -vvvv`
 - `ansible -i w_hosts1.yml windows -m win_ping -vvvv`
- ❖ Try executing some playbooks
 - `ansible-playbook -i w_hosts.yml w_playbook.yml -vvvv`
 - `ansible-playbook -i w_hosts1.yml w_playbook1.yml -vvvv`
 - `verify on localhost:8080`

Setting Things Up

- ❖ Ansible modules for Windows

- https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html



Ansible Tower and AWX

Ansible Tower

- ❖ Role-based access control
- ❖ Ansible job scheduling
- ❖ Graphical inventory management
- ❖ Full Ansible job history for an audit trail
- ❖ Real-time output of playbook runs
- ❖ Self-service IT features to delegate jobs

Ansible Tower

- ❖ Installation
 - Download tower tar
 - untar it
 - go to the folder and run setup.sh
- ❖ `sudo ansible-tower-service status`
- ❖ `sudo ansible-tower-service restart`

- ❖ Installation steps
 - Install Docker
 - Install Node and NPM
 - Install python-pip git pwgen vim
 - requests module
 - Git clone AWX repository
 - Modify inventory file
 - Run the install.yml playbook
 - Access AWX on IP:80

❖ Install Docker

- `sudo apt-get update`
- `sudo apt -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common`
- `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
- `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`
- `sudo apt update`
- `sudo apt -y install docker-ce docker-compose`
- `sudo docker version`

- ❖ Install Docker
 - `sudo systemctl enable docker`
 - `sudo systemctl status docker`
 - `sudo systemctl start docker`

AWX

- ❖ Install Node and NPM
 - `sudo apt install -y nodejs npm`
 - `sudo npm install npm --global`
- ❖ Install python-pip git pwgen vim and requests module
 - `sudo apt -y install python-pip git pwgen vim`
 - `sudo pip install requests==2.14.2`
- ❖ Git clone AWX repository
 - `git clone --depth 50 https://github.com/ansible/awx.git`
- ❖ Modify inventory file
 - `cd awx/installer/`
 - `vim inventory`

AWX

- ❖ Generate secret
 - `pwgen -N 1 -s 30`
- ❖ Run the install.yml playbook
 - `ansible-playbook -i inventory install.yml`
- ❖ Access AWX on IP:80
 - User name: `admin`
 - Password: `password`

AWX Exercises

- ❖ Add the following
 - Organization
 - Users
 - Inventory
 - Hosts
 - credentials
 - Project
 - Template
- ❖ Launch Templates
- ❖ Bulk inventory addition
 - `awx-manage inventory_import --inventory-name Linux_UA_Hosts --source hosts_add`

Additional Playbooks

OCI Playbooks

- ❖ <https://docs.cloud.oracle.com/en-us/iaas/Content/API/SDKDocs/ansiblegetstarted.htm>
- ❖ <https://github.com/oracle/oci-ansible-modules>
- ❖ <https://www.martinberger.com/?p=4819>
- ❖ <https://javiermugueta.blog/2019/04/12/ansible-for-oracle-cloud-infrastructure-oci/>

MySQL Playbooks

- ❖ <https://github.com/geerlingguy/ansible-role-mysql/tree/master/tasks>
- ❖ <https://github.com/ontic/ansible-role-mysql>

AWS Related Playbooks

- ❖ https://docs.ansible.com/ansible/latest/scenario_guides/guide_aws.html
- ❖ <https://www.freecodecamp.org/news/ansible-manage-aws/>

Ansible Certification

Certification

- ❖ Exam

- Redhat EX407

- ❖ Costs

- 400 USD

- ❖ Dumps

- https://github.com/rilindo/ex407_ansible_specialist_exam_labs



Exercises

Exercise

- ❖ Create a final role, as per the instructions
 - `webapp`
- ❖ Run the role and see if the webapp got deployed successfully
 - target nodes only `slaves` group
- ❖ Now Push the role to your git repository
- ❖ Deploy the same role to your ansible environment via `AWX`

Exercise

- ❖ Don't have a github.com account?
 - Create one at <https://github.com>
 - Create a repository
 - Clone the repository
 - be familiar with only the below 4 commands for this demo
 - `git add`
 - `git commit`
 - `git push`
 - `git pull`

Best Practices

- ❖ Default your Parameters
- ❖ Always define the state
- ❖ Use Tags
- ❖ Version Control
- ❖ Comments your code
- ❖ Readability over strict code
- ❖ Use Roles
- ❖ Split playbooks into logical groupings

Parting Talk

- ❖ I hope we were able to give you all a great kick into the world of Ansible
- ❖ Take it as a journey, go step by step while automating things around
- ❖ All the best for your journey!!



References

- ❖ Ansible Official Documentations
- ❖ <http://ansible.redhatgov.io/standard/core/index.html>
- ❖ <https://ansible.github.io/workshops/>
- ❖ <https://medium.com/quick-code/top-tutorials-to-learn-ansible-33afd23ea160>
- ❖ <https://www.ansible.com/blog/ansible-best-practices-essentials>

Thank you for being an
awesome Audience!

