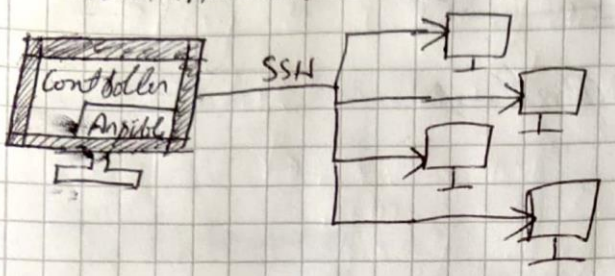
Ansible → Powerful automation tool that uses YAML to perform complex task such as patching OS to deployments.

Ansible inventory

Ansible is agentless ie it donot need any additional client software to run a simple SSH connection is enough to perform task from the controller machine to other machines on the network.



- Agentless
- SSH / Powershell Remoting is enough.

Detail needed to connect to these system / remote m/c is stored in "Inventory" file. Default inventory file is at ~~/etc~~ "/etc/ansible/hosts"

Inventory file

→ Grouping (Action can be applied to whole group at once)

```
[app]
web   ansible_host=a.b.com ansible_connection=ssh ansible_port=22
      ansible_user=root    ansible_ssh_pass=pw
db    ansible_host=b.b.com ansible_connection=winrm ansible_port=5986
      ansible_user=admin   ansible_password=pw
mail  ansible_host=c.c.com ansible_connection=ssh ansible_port=22
      ansible_user=root    ansible_ssh_pass=pw
web2  ansible_host=d.d.com ansible_connection=winrm ansible_port=5986
      ansible_user=admin   ansible_password=pw
```

[app1]
web
db

(Alias for connection)

Feilds →
- ansible_host → FQDN of the server
- ansible_connection → type of connection default SSH
- ansible_port → port to connect default 22
- ansible_user → user
- ansible_ssh_pass → Pass Linux / ansible_password = winpw

Example:

Web Servers

web1 ansible_host=w1.dc.com ansible_connection=ssh ansible_user=u1 ansible_ssh_pass=pw

web2 —"— w2.dc.com —"— ssh —"— u2 —"— pw

for ssh

DB Servers

db ansible_host=db.dc.com ansible_connection=winrm ansible_user=admin ansible_password=pw

winrm

Group

[db_n]
db

[web_n]
web1
web2

[all_n:children]
db_n
web_n

→ Group of groups we must use ":children"

To check connectivity

Cmd : ansible web1 -m ping -i ./inventory.txt

alias

ping module to ping

inventory file

↳ inventory.txt

↳ when specified it doesn't use the default one.

YAML - all config are done in yaml in ansible

Ansible Playbook

Playbook are the orchestration language of ansible.
All playbook is an YAML file

→ Playbook has set of Play
  → Play is a set of activities/task to run on hosts
    → Task is an action to be performed on the host
      ↳ Exec a cmd
        Run a script
        Install or deploy
        Shutdown/restart

Running Ansible

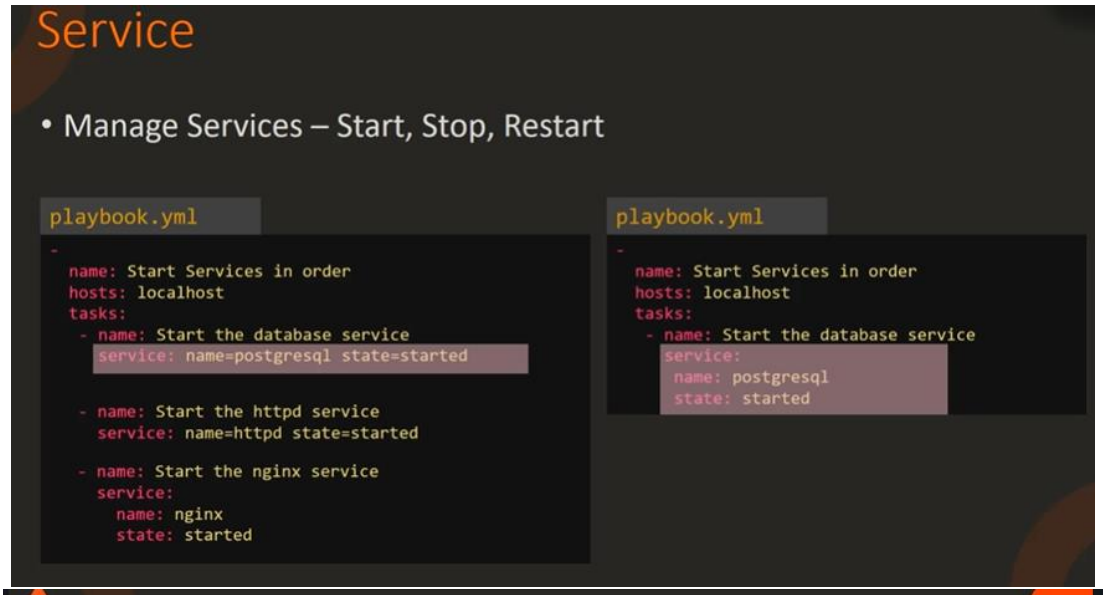
cmd:	ansible	ansible-playbook
Usecase:	for simple task to execute a cmd or check connectivity	Used for complex tasks to run set of complex tasks build, test & deploy
eg:	ansible <hosts> -a <cmd> ansible all -a "/sbin/reboot" ansible web1 -m ping ansible <host> -m <module>	ansible-playbook <playbook name> ansible-playbook run-web.yaml

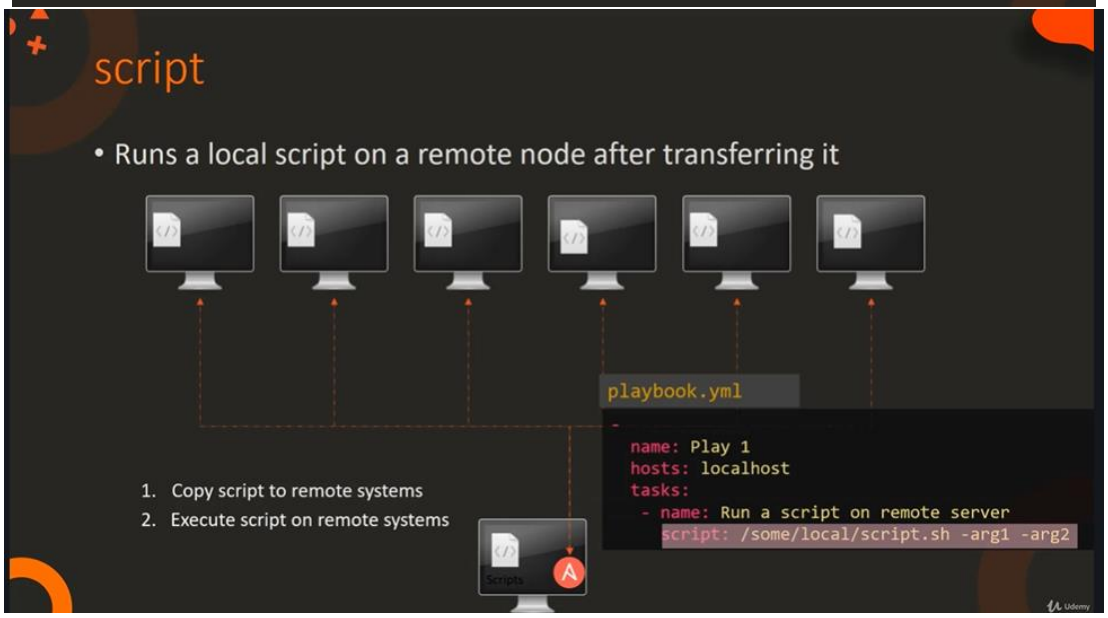
Note:

Ansible creates a default group "all" that has all the hosts as a part of it.

Modules in Ansible:

Built in modules help us to carry out task with ease





command

Executes a command on a remote node

parameter	comments
chdir	cd into this directory before running the command
creates	a filename or (since 2.0) glob pattern, when it already exists, this step will **not** be run.
executable	change the shell used to execute the command. Should be an absolute path to the executable.
free_form	the command module takes a free form command to run. There is no parameter actually named 'free form'. See the examples!
removes	a filename or (since 2.0) glob pattern, when it does not exist, this step will **not** be run.
warn (added in 1.8)	if command warnings are on in ansible.cfg, do not warn about this particular line if set to no/false.

playbook.yml

```
-
  name: Play 1
  hosts: localhost
  tasks:
   - name: Execute command 'date'
     command: date

   - name: Display resolv.conf contents
     command: cat /etc/resolv.conf

   - name: Display resolv.conf contents
     command: cat resolv.conf chdir=/etc

   - name: Display resolv.conf contents
     command: mkdir /folder creates=/folder
```

```
- name: Copy file from source to destination
  copy: src=/source_file dest=/destination
```

lineinfile

- Search for a line in a file and replace it or add it if it doesn't exist.

/etc/resolv.conf

```
nameserver 10.1.250.1
nameserver 10.1.250.2
```

```
nameserver 10.1.250.10
```

playbook.yml

```
-
  name: Add DNS server to resolv.conf
  hosts: localhost
  tasks:
   - lineinfile:
       path: /etc/resolv.conf
       line: 'nameserver 10.1.250.10'
```

/etc/resolv.conf

```
nameserver 10.1.250.1
nameserver 10.1.250.2
nameserver 10.1.250.10
```

script.sh

```
#Sample script

echo "nameserver 10.1.250.10" >> /etc/resolv.conf
```

/etc/resolv.conf

```
nameserver 10.1.250.1
nameserver 10.1.250.2
nameserver 10.1.250.10
nameserver 10.1.250.10
nameserver 10.1.250.10
```

idempotency

Why "started" and not "start"?

"Start" the service httpd

"Started" the service httpd

Ensure service httpd is started

If httpd is not already started => start it
If httpd is already started, =>do nothing

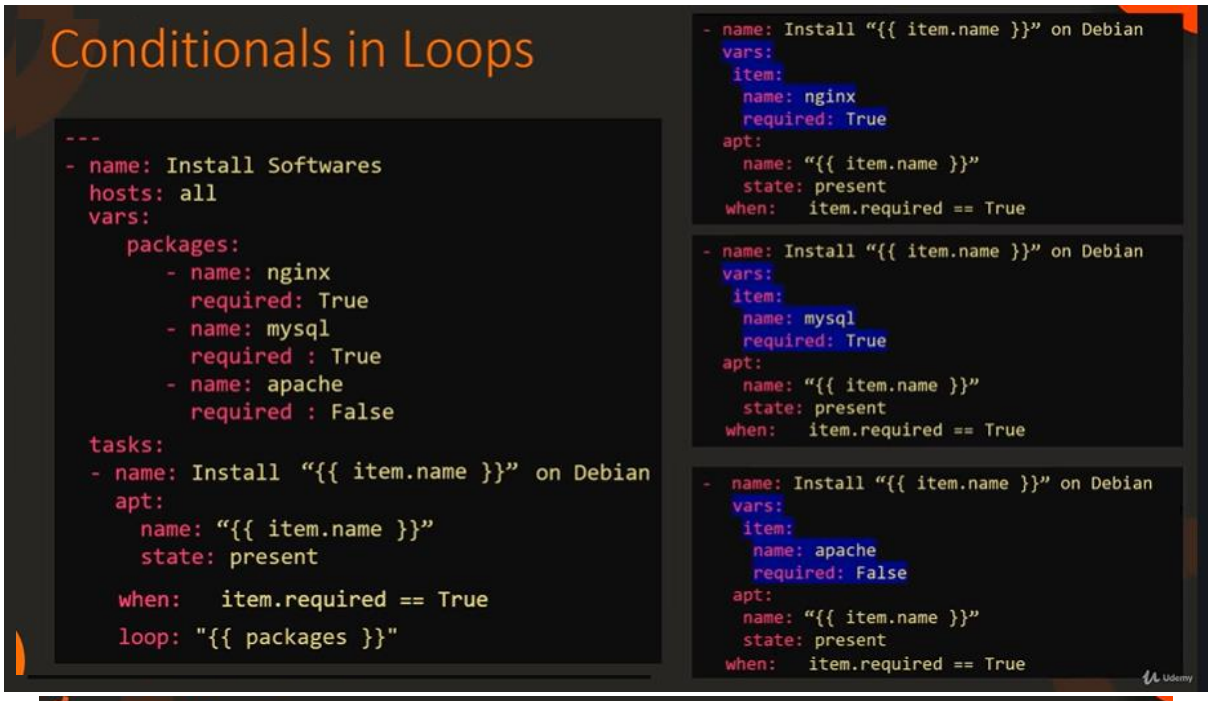
Idempotency
An operation is idempotent if the result of performing it once is exactly the same as the result of performing it repeatedly without any intervening actions.

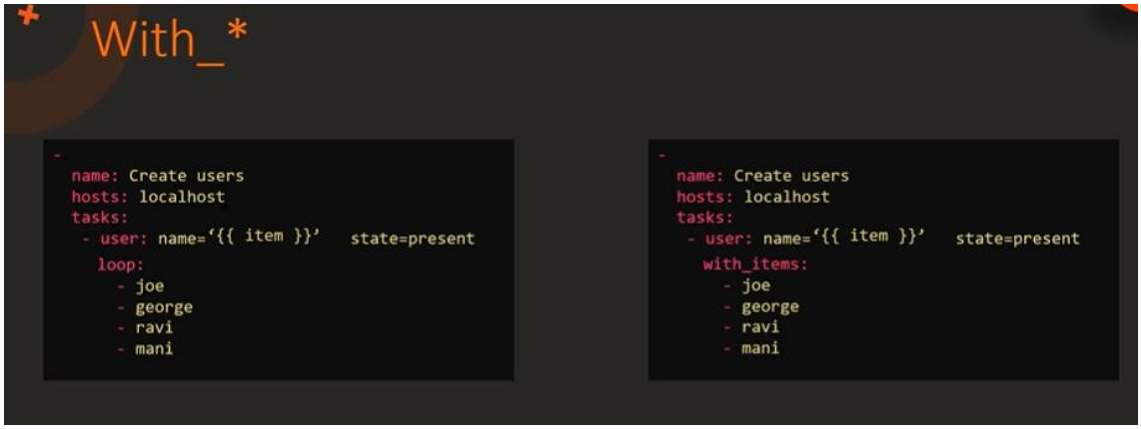
Modules example:

```
# Update the playbook with a play to Execute a script on all web server nodes.
 The script is located at /tmp/install_script.sh
- name: "Execute a script on all web server nodes"
  hosts: web_nodes
  tasks:
    - name: "Execute a script on all web server nodes"
      script: /tmp/install_script.sh
# Update the playbook to add a new task to start httpd services on all web nod
es
- name: "Execute a script on all web server nodes"
  hosts: web_nodes
  tasks:
    - name: "Execute a script"
      script: /tmp/install_script.sh
    - name: "Start httpd service"
      service: "name=httpd state=started"
# Update the playbook to add a new task in the beginning to add an entry into
/etc/resolv.conf file for hosts. The line to be added is nameserver 10.1.250.1
0
- name: "Execute a script on all web server nodes and start httpd service"
  hosts: web_nodes
  tasks:
    - name: "Update entry into /etc/resolv.conf"
      lineinfile:
        path: /etc/resolv.conf
        line: "nameserver 10.1.250.10"
    - name: "Execute a script"
      script: /tmp/install_script.sh
    - name: "Start httpd service"
      service:
        name: httpd
        state: started
# Update the playbook to add a new task at second position (right after adding
 entry to resolv.conf) to create a new web user.
# Use the user module for this. User details to be used are given below:
# Username: web_user
# uid: 1040
# group: developers
- name: "Execute a script on all web server nodes and start httpd service"
  hosts: web_nodes
  tasks:
    - name: "Update entry into /etc/resolv.conf"
      lineinfile:
        path: /etc/resolv.conf
        line: "nameserver 10.1.250.10"
    - name: "Create a new user"
      user:
```

```
      name: web_user
      uid: 1040
      group: developers
  - name: "Execute a script"
    script: /tmp/install_script.sh
  - name: "Start httpd service"
    service:
      name: httpd
      state: started
```

Loops in ansible: we can use look or with_item to iterate





With_*

with_items
with_file
with_url
with_mongodb
with_dict
with_etcd
with_env
with_filetree
With_ini
With_inventory_hostnames
With_k8s
With_manifold
With_nested
With_nios
With_openshift
With_password
With_pipe
With_rabbitmq
With_redis
With_sequence
With_skydive
With_subelements
With_template
With_together
With_varnames

With_*

```
-
  name: Create users
  hosts: localhost
  tasks:
   - user: name='{{ item }}'   state=present
     with_items:
       - joe
       - george
       - ravi
       - mani
```

```
-
  name: View Config Files
  hosts: localhost
  tasks:
   - debug: var=item
     with_file:
       - "/etc/hosts"
       - "/etc/resolv.conf"
       - "/etc/ntp.conf"
```

```
-
  name: Get from multiple URLs
  hosts: localhost
  tasks:
   - debug: var=item
     with_url:
       - "https://site1.com/get-servers"
       - "https://site2.com/get-servers"
       - "https://site3.com/get-servers"
```

```
-
  name: Check multiple mongodbs
  hosts: localhost
  tasks:
   - debug: msg="DB={{ item.database }} PID={{ item.pid}}"
     with_mongodb:
       - database: dev
         connection_string: "mongodb://dev.mongo/"
       - database: prod
         connection_string: "mongodb://prod.mongo/"
```

Loops & loops with conditions example:

```yaml
# to loop we can use "Loop" and "with_items"

# loop directive (with_items) to the task to print all fruits defined in the f
ruits variable.
- name: "Print list of fruits"
  hosts: localhost
  vars:
    fruits:
      - Apple
      - Banana
      - Grapes
      - Orange
  tasks:
    - command: 'echo "{{ item }}"'
      with_items: "{{ fruits }}"

# loop to install packages
- name: "Install required packages"
  hosts: localhost
  vars:
    packages:
      - httpd
      - binutils
      - glibc
      - ksh
      - libaio
      - libXext
      - gcc
      - make
      - sysstat
      - unixODBC
      - mongodb
      - nodejs
      - grunt
  tasks:
    - yum:
        name: "{{ item }}"
        state: present
      loop: "{{ packages }}"
```

Conditional example:

```
#The given playbook attempts to start mysql service on all_servers.
#Use the when condition to run this task if the host (ansible_host) is the dat
abase server.
# We have created a group for web servers. Similarly create a group for databa
se servers named 'db_servers' and add db1 server to it

# # Sample Inventory File
# # Web Servers
# web1 ansible_host=server1.company.com ansible_connection=ssh ansible_user=ro
ot ansible_ssh_pass=Password123!
# web2 ansible_host=server2.company.com ansible_connection=ssh ansible_user=ro
ot ansible_ssh_pass=Password123!
# web3 ansible_host=server3.company.com ansible_connection=ssh ansible_user=ro
ot ansible_ssh_pass=Password123!
# # Database Servers
# db1 ansible_host=server4.company.com ansible_connection=winrm ansible_user=a
dministrator ansible_ssh_pass=Password123!
# [web_servers]
# web1
# web2
# web3
# [db_servers]
# db1
- name: "Execute a script on all web server nodes"
  hosts: all_servers
  tasks:
    - service: "name=mysql state=started"
      when: 'ansible_host=="server4.company.com"'

##AGE check
- name: "Am I an Adult or a Child?"
  hosts: localhost
  vars:
    age: 25
  tasks:
    - command: 'echo "I am a Child"'
      when: "age < 18"
    - command: 'echo "I am an Adult"'
      when: "age >= 18"

#Install NGINX with correct package manager
# ansible_os_family holds the value of OS family
- name: "Install NGINX"
  hosts: all
  tasks:
    - name: "Install nginx on DEBIAN"
      apt:
```

```
        name: nginx
        state: present
      when: ansible_os_family == "Debian" and
        ansible_distribution_version == "18.04"
    - name: "Install nginx on RED HAT"
      yum:
        name: nginx
        state: present
      when: ansible_os_family == "RedHat"    or
        ansible_os_family == "SUSE"

#Check if service is down and trigger a mail
- name: "Check and alert"
  hosts: all
  tasks:
    - command: service httpd status
      register: result
    - mail:
        to: nithinbs18@gmail.com
        subject: service down
        body: httpd service is down
        when: result.stdout.find('down') != -1
# find() in the above when line checks the content if found it returns the pos
ition of the string if not fount it returns -1
# any value other than -1 indicated the service is down
```

Task description

Write an ansible playbook, that performs the following tasks:

- builds a new docker image using a Dockerfile you provide (please use this base image from hub.docker.com: centos:7)

  - the new image should have:

    - all recent system updates - nginx installed and running

    - nginx should serve at /index.html a simple html page only containing the timestamp of time when the file was created

- starts the new image (port 80 should be reachable from the outside)

The expectation is, that we can run the playbook and then easily can reach the html page inside the running container on port 80

Technical requirements

Please provide *one* zip file that contains the following:

- playbook, dockerfile

- documentation how to start the playbook and access the aforementioned html page

```yaml
- name: "DevOps challange  Nithin Bhardwaj Sridhar"
  hosts: localhost
  tasks:
    - name: Ensure context directory exists
      file:
        path: ./docker-content
        state: directory
    - name: Check if default.conf exists
      stat:
        path: ./docker-content/default.conf
      register: default_result
    - name: Ansible create file with content if default.conf is not present
      copy:
        dest: "./docker-content/default.conf"
        content: |
          server {
            listen       80;
            server_name  localhost;
            location /index.html {
                root   /usr/share/nginx/html;
                index  index.html index.htm;
          try_files $uri /index.html;
            }
          }
      when: not default_result.stat.exists
    - name: Record current time
      command: date
      register: date_result
    - name: Create/Overwrite file index.html
      copy:
        dest: "./docker-content/index.html"
        content: |
          <html lang="en">
            <head>
              <title>Time</title>
              <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/c
ss/bootstrap.min.css" rel="stylesheet" />
              <style>
                html,
                body {
                  height: 100%;
                  background-color: #333;
                }
                body {
                  display: -ms-flexbox;
                  display: flex;
                  color: #fff;
                  text-shadow: 0 0.05rem 0.1rem rgba(0, 0, 0, 0.5);
```

```
                box-shadow: inset 0 0 5rem rgba(0, 0, 0, 0.5);
              }
            </style>
          </head>
          <body class="text-center">
            <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-
column">
              <header class="masthead mb-auto">
              <h5>Nithin Bhardwaj Sridhar</h5>
              </header>
              <main role="main" class="inner cover">
                <h1 class="cover-
heading">Created Time: {{date_result.stdout}}</h1>
              </main>
              <footer class="mastfoot mt-auto"></footer>
            </div>
          </body>
        </html>
    - name: Build docker image
      docker_image:
        path: .
        name: task-app
        tag: latest
        force: yes
    - name: Create/Re-create and start the container
      docker_container:
        name: nithin-task
        image: task-app
        state: started
        recreate: yes
        ports:
          - "80:80"
```

Docker file:

```
FROM centos:7
RUN yum update -y
RUN yum install -y epel-release && \
    yum install -y \
    nginx && \
    yum clean all
COPY ./docker-content/default.conf /etc/nginx/conf.d/
COPY ./docker-content/index.html /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```