↓ execute Adhok-command}
→ Create a file, seeing Server memory - execute command
memory Utilization
→ Adhock command - create a file (Not Saved anywhere in Ansible)

ansible
> Cfg (conf.)
> Hosts
> inventory

Controller —create a file→ [ ] {Adhok - oneLiner on server}
→ allserver, [Specific group]

> ansible all -a " touch /tmp/ravi " # one Liner
Which server, you executing this adhock command

✗ Something changed.

> ansible all -a " ls -l /tmp"
→ Ravi file created.

> ansible all -a " free -m" (target command, executed Running on P)

> ansible Localhost -a " free -m" (Memory Utilization)
> " all -b - a " df -h" (directory Structure)
> " apt-get update"
↳ will Run as a Root user
-b ↑ become a Root user by default

> ps -ef | grep apache2 ← [Search apache2]
↑ Running process.

| Capital K - Root user Password | small -k (Normal user password) |

" apt-get install apache2"
" apt-get install -y apache2"
↑ {will ask yes or no user}
-y (will bypass)

-b [-k] -a
↑ become Root user
BEcome password:

> ansible all -a " curl Localhost" - [executed Remote machine]

(Create file only on one server) > ansible all -a " touch /tmp/test" --limit "3.135.22"
↑ ip

> Limit → the no. of Server where we want to execute task.

> ansible all -a " ls -l /temp/"
↑ - Limit
- Serial Fashion

set target then limit them

> " -a " root/labs/ docker ... .sh"
↑ sh ————→

> Modules → copy File from one Location to other
CP ← [Linux Commands]

1. COPY 3. Stat ← {predefine modules} ... [3000 modules]
2. file 4. find

→ Ansible Documentation
script module ansible — pred-defined ansible command

> ansible all -m script -a "- - - /sh"

-b (Root user)

consider as a script module & execute it (after transfering it to be (module))

> Script Run it on production machine

> ansible all -b -a "docker version".

* > ansible all -b -m setup (

3.1 → "ansible_facts" : → os, disk space, hardware (info. of Remote machine)

> date & time. (ansible collect data from Remote machine that is called facts)

> ansible all -b -m setup (→ facts collects) [all data] [info. collected from Remote machine]

> ansible-doc -- list (show list of modules)

> adhock (Req. without plane, immediate execution)

> Copy the Feedback to file (> text.data)
(Collect the verbose output & save them to file)

** Playbook - IO -COPY (Simple Scripts Set)
[Contains Set of activity] > java, jenkins. [Set of activity]

> Copy something on remote machine (cp touch)

> execute task in a single shots (Set of activity performed by administ.)
[what Syntax is being used]

> YAML - data structure | serialization Lang.)

* yaml Syntax → List -
dictionary -

who all are in class? →
participants (list)
- ravi - ravikaran
- kalyan

What you want to execute? → List of Fruits.

{environment}
(key-value pair)
(dictionary)

ravi (want to know about ravi)
- roles: mast
cert: dca
occ: trainer

→ key-value pair

List of fruits
fruits:
- Apple
- Banana

dictionary of martin
martin:
name: Rakesh
Job: Developer
Skills:
- Python

- List
- Dictionary
- Spaces.

* yaml or .yml

- hosts: all (- Symbol Start list)
↑ specific group. (all or specific group)
become: true (become a root user) (or ignore it)
Keywords
↳ tasks: (what you want to perform)
- name: copy the data. (any name) (what want
copy: (module Name) wher to copy)
src: /root/ravi/~~module~~ inv (~~Copy~~ the File)
dest: /tmp

> ansible-playbook copy.yml (

task: [Gathering Facts] → By default info.
ok: [3.22.10]

task: [copy the files]
changed: [ ]

play Recap: ********
3.10.12 : ok=2 changed=1

> ansible-playbook copy.yml [P1] Removed Ravi
↳ changed happends at one place. [P2] file (intelligent)

> don't use tab (use dots)

> copy.yml (Aws) [File module]

- hosts: aws
become: yes
tasks:
- name: Create ne
file:
path: /tmp/ravifile
state: touch

path: /-
state:
directory
state: absent] folder

* - hosts: aws
  become: yes
  tasks:
    - name: Install Git
      apt:

    - name:

?

* ---
  - hosts: aws
    tasks:
      - name: printing
        debug:
          message: "hello world"

output to this variable.

Register: out

(?) print data out of variables

msg: "{{ out }}"

x=5, echo x, echo $x → (V)

→ x (Not one Red)

* Facts (collects Facts & then add conditions)

* when - conditions (conditions)

↳ stats (statics) → ~~com~~

getting Facts (once facts are gethered then other things are possible)