ANSIBLE

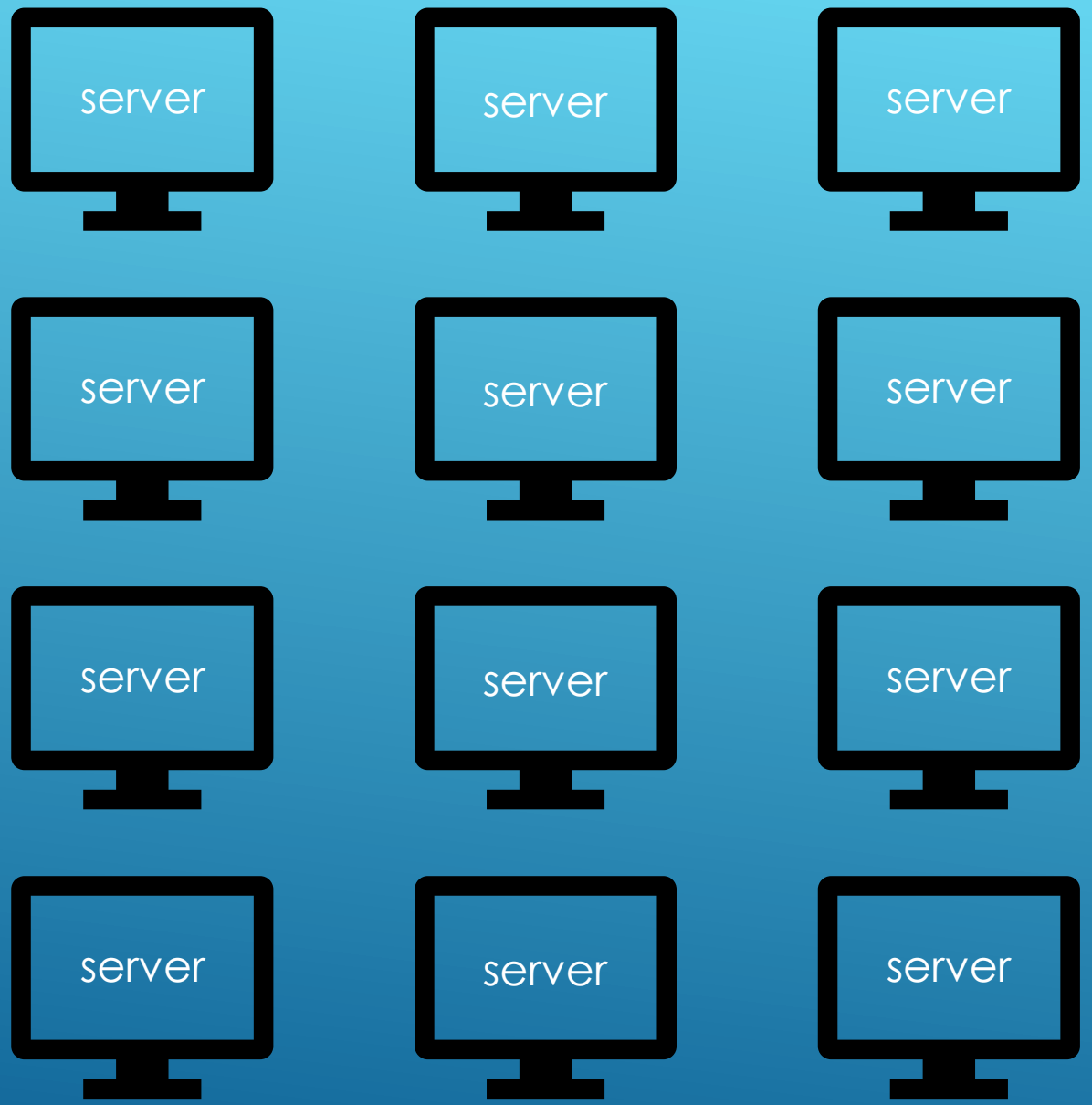
By Mostafa Yehia

DAY 1 AGENDA

- What is Ansible
- Why Ansible
- SSH overview
- Ansible & SSH
- Installing Ansible & preparing SSH
- Ad-hoc commands
- Inventory file
- Ansible.cfg file
- Ad-hoc commands escalation
- Ansible playbook
- Ansible modules

WHAT IS ANSIBLE ?

- Ansible is a software tool that provides simple but powerful automation for cross-platform computer support.
- It is used for updates on workstations and servers, cloud provisioning, configuration management, and nearly anything a systems administrator does on a daily basis.


server
server
server
server
server
server
server
server
server
server
server
server

WHY ANSIBLE ?

- Idempotent: 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.
- Agentless: Other tools like (Puppet & Chef) require an agent to be installed on the target device. Ansible only requires an SSH connection to the target device.
- Open-source: Ansible is an open-source community project sponsored by Red Hat.

SSH OVERVIEW

- OpenSSH is the premier connectivity tool for remote login with the SSH protocol.
- It encrypts all traffic to eliminate connection hijacking and other attacks.

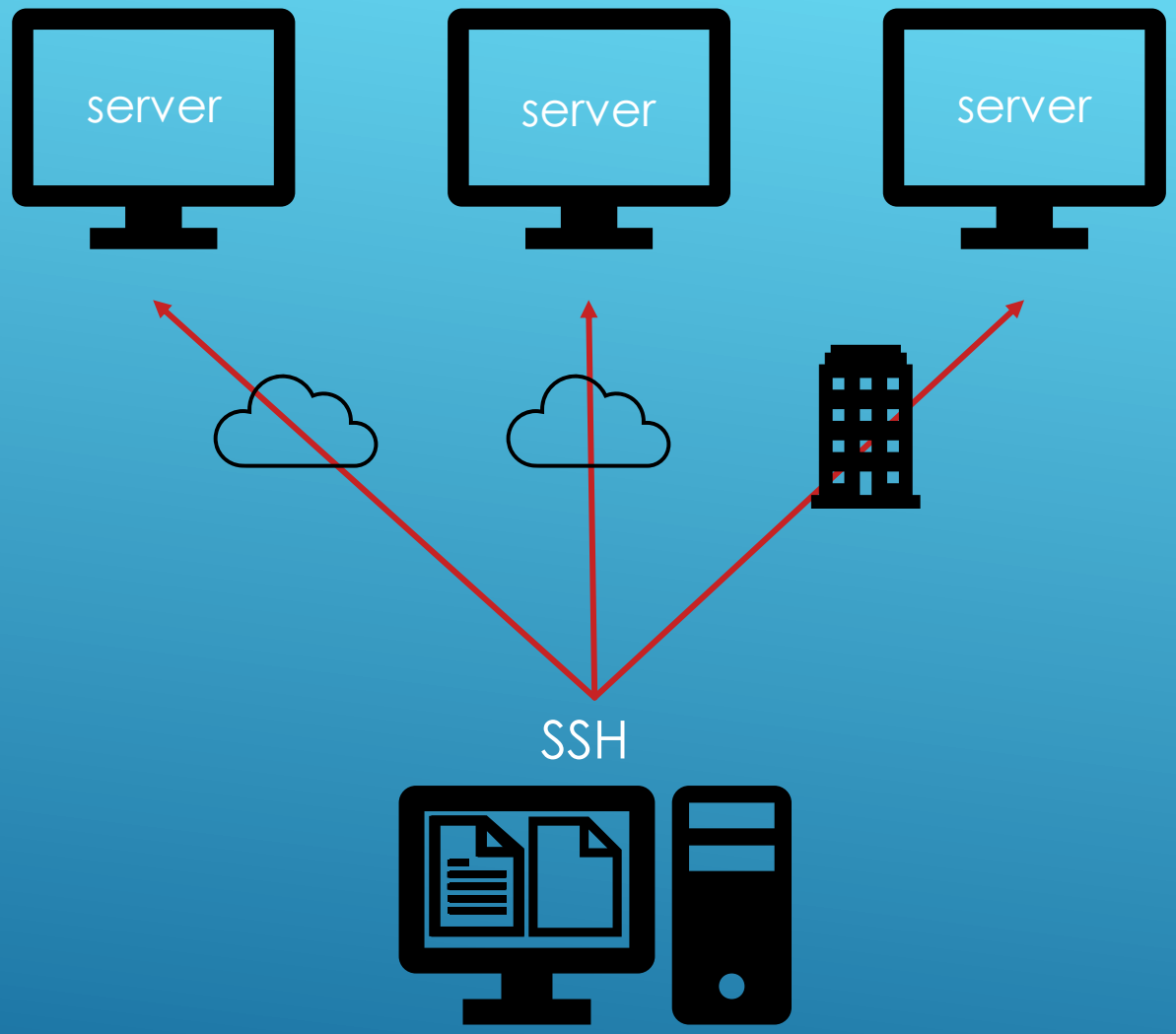
PC 2

SSH

PC 1

ANSIBLE & SSH

How Ansible connects to servers ?



Ansible control machine

INSTALLING ANSIBLE & PREPARING SSH

- Install ansible
- Create a new user on control machine and new user on host 1
- Make sure you can ssh into host 1 (using password)
- Generate SSH key pair on control machine
- Copy the public key to host 1
- Make sure you can ssh into host 1 (using prv/pub)

AD-HOC COMMANDS

Ad-hoc: Running Ansible to perform some quick command.

An example of an ad hoc command might be rebooting 50 machines in your infrastructure.

Structure:

ansible [pattern] -i [inventory] --private-key [/path/to/private/key] -u [remote_user] -m [module_name]

Example:

ansible all -i 3.87.24.251, --private-key ~/.ssh/devops -u ubuntu -m ping

SSH equivalent:

ssh ubuntu@3.87.24.251 -i ~/.ssh/devops -o 'RemoteCommand echo hi; bash;' -t

INVENTORY FILE

inventory file: A file that describes Hosts and Groups in Ansible.

Examples:

[web_servers]

3.87.24.251

[database_servers]

3.87.24.252

3.87.24.253

INVENTORY FILE

- Create the inventory file
- Put the IP of host 1 in the inventory file
- Use the inventory file path in your ad-hoc command instead of using the IP hard-coded
- Example:
  ansible all -i inventory --private-key ~/.ssh/devops -u ubuntu -m ping

CONFIGURATION FILE

ansible.cfg file: This is the brain and the heart of Ansible.

The file that governs the behavior of all interactions performed by the control machine.

Locations:

ANSIBLE_CONFIG (environment variable if set)

ansible.cfg (in the current directory)

~/.ansible.cfg (in the home directory)

/etc/ansible/ansible.cfg

Example:

```
[defaults]
inventory = ./inventory
private_key_file = ~/.ssh/devops
remote_user = ubuntu
```

CONFIGURATION FILE

- Create the configuration file
- Insert some values in the configuration file
- Run the minimized ad-hoc command
- Example: ansible all -m ping

AD-HOC COMMANDS ESCALATION

Ad-hoc: Running Ansible to perform some quick command with SUDO permissions.

Structure:

ansible [pattern] -m [module_name] --become

Example:

ansible all -m command -a "whoami" --become

Example ansible.cfg:

```
[privilege_escalation]
become = true
```

AD-HOC COMMAND ESCALATION USING ROOT USER

- Insert the correct values in the configuration file
- Example: ansible all -m command -a "whoami"
- What is the output of the command ?

PLAYBOOK

- A playbook is a list of plays.
- playbook: Playbook is the language Ansible uses to orchestrate, configure, administer, or deploy systems.
- Play: is a mapping between a set of hosts (groups, hostnames, or IPs) and the tasks which run on those hosts to define the role that those systems will perform.
- There can be one or many plays in a playbook.

Examples:

```
- name: your play name
  hosts: all
  tasks:
  - name: your task name
    ping:
```

PLAYBOOK

- Write your first playbook file
- Stop gather_facts and update cache

MODULES

Modules: are units of code that can control system resources or execute system commands.

Ansible provides a module library that you can execute directly on remote hosts or through playbooks.

Playbook example:

```
- name: your play name
  hosts: all
  tasks:
  - name: your task name
    ping:
```

Ad-hoc command example:

```
ansible all –m ping
```

MODULES

- Explore some built-in modules like:

(**apt, dnf, package, service, command, copy, user, group, lineinfile, authorized_key, etc.**)

ansible-builtin modules

- Update cache
- Install latest nginx
- Copy index.html from controller to host 1
- Restart nginx service
- Can you see your index.html file when you hit host 1 on port 80 ?

QUESTIONS ?

THANK YOU

DAY 2 AGENDA

- Tags
- Variables
- Loops
- When
- Register

TAGS

Tags: Running only specific parts of a playbook instead of running the entire playbook.

Example:

```
- name: my play with tags
  hosts: all
  tasks:

  - name: my task1 with tags
    tags: my_first_tag
    ping:

  - name: my task2 with tags
    tags: my_second_tag
    ping:
```

```
ansible-playbook my-playbook.yml --tags my_first_tag
```

TAGS

- Write simple playbook file
- Add two tasks (apt update – apt install nginx)
- Add tags to first task: update
- Add tags to second task: install
- Run only the (apt update) task
- Example: ansible-playbook my-playbook.yml --tags update
- Add one task with “tags: always” and run the previous command again

VARIABLES

Variables: Ansible uses variables to manage differences between systems.

Locations:

- Playbooks
- Inventory
- Command Line
- Register
- Files or Roles

example:

```
- name: my play with variables
  hosts: all
  vars:
    package: nginx
    version: latest
  tasks:

  - name: my task with variables
    apt:
      name: "{{ package }}"
      state: "{{ varsion }}"
```

VARIABLES

Define these variables (package_name, package_version)

- on playbook level
- on inventory level
- on command line level

Use apt module with the package name and version from your variables

LOOPS

Loops: Ansible uses loops to execute a task multiple times.

example:

```
- name: my play with loops
  hosts: all
  tasks:

  - name: my task with loops
    apt:
      name: "{{ item }}"
      state: latest
    loop:
      - nginx
      - mariadb-server
      - curl
```

example:

```
- name: my play with loops
  hosts: all
  tasks:

  - name: my task with loops
    apt:
      name: "{{ item.package_name }}"
      state: "{{ item.package_state }}"
    loop:
      - { package_name: "nginx", package_state: “present"}
      - { package_name: “mariadb-server", package_state: “latest"}
      - { package_name: “curl", package_state: “absent"}
```

LOOPS

- Loop over a list of packages and install latest versions.
- Loop over a list of packages and perform different actions as per input.

WHEN

When: You want to execute different tasks depending on the value of a fact, a variable, or the result of a previous task.

Example:

```
- name: my play with conditions
  hosts: all
  tasks:

  - name: my task1 with conditions
    apt:
      name: nginx
    when: ansible_facts['distribution'] == “Ubuntu“

  - name: my task2 with conditions
    apt:
      name: httpd
    when: ansible_facts['distribution'] == “CentOS“
```

WHEN

- Install nginx or apache2 depending on distribution
- Restart nginx service if distribution is ubuntu and variable value is true

REGISTER

Register: Ansible register is a way to capture the output from task execution and store it in a variable.

Example:

```
- name: my play with register
  hosts: all
  tasks:

  - name: my task1 with register
    command: cat /var/www/html/index.html
    register: my_result

  - name: my task2 with register
    debug:
      var: my_result
```

REGISTER & WHEN

- View the value of your register variable using debug module
- Restart service if the installation task was changed or was not failed

QUESTIONS ?

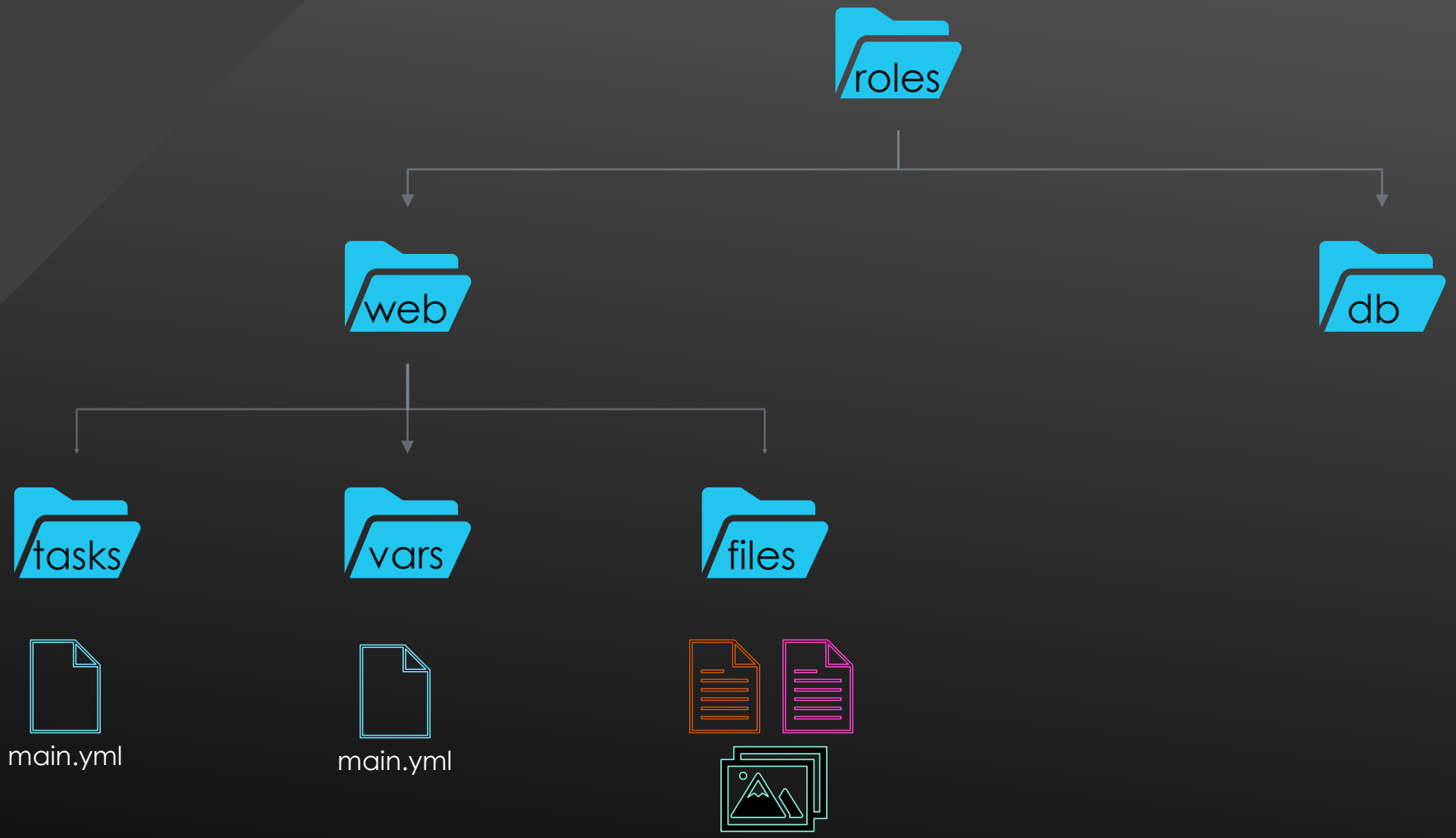
THANK YOU

DAY 3 AGENDA

- Roles
- Handlers
- Templates “.j2”
- Ansible-Vault
- Ansible-Galaxy

ROLES

Roles: pack related vars, files, templates, tasks, and handlers, based on a known file structure so you can easily reuse them and share them with other users.


roles
web
db
tasks
vars
files
main.yml
main.yml

ROLES

- Create your first role with name (web)

- The task book will include:

1. installing a package
   (get the package name from vars)
2. copying a list of files from controller to host using loop
   (get the list of file names from vars)
   (the actual files will be stored in ./roles/web/files)
   (will be executed only when the install task is in state: changed)

- Restart the service of the installed package
  (will be executed only when the copy task is in state: changed)

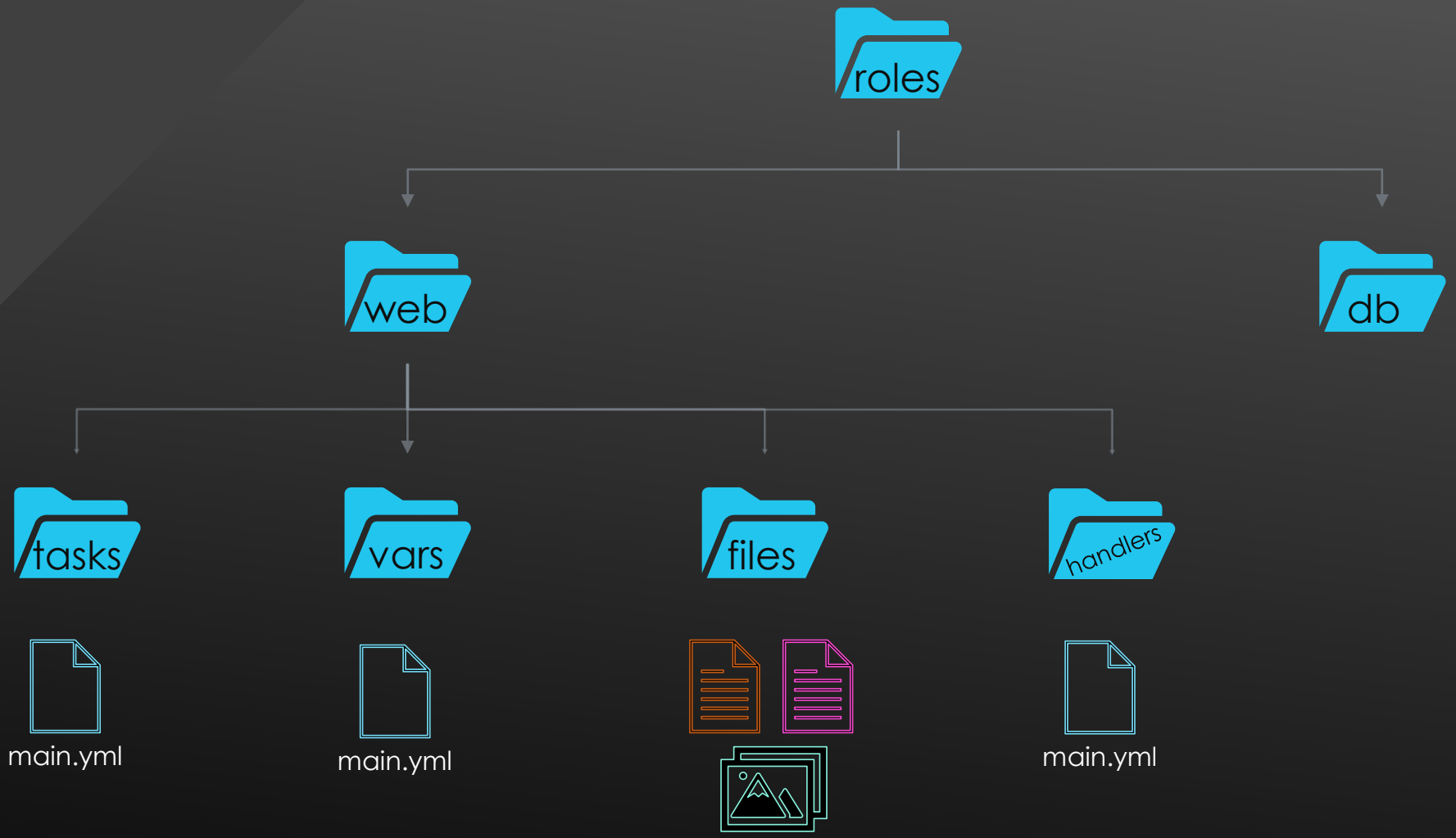
HANDLERS

Handlers: Handlers are tasks that only run when notified.

Example:

```
- name: my play with handlers
  hosts: all
  tasks:
  - name: my task1 with handlers
    apt:
      name: nginx
    notify: my_handler

  handlers:
  - name: my_handler
    service:
      name: nginx
      state: restarted
```


roles
web
db
tasks
vars
files
handlers
main.yml
main.yml
main.yml

HANDLERS

- Create your first role with name (web)

- The task book will include:

1. installing a package
   (get the package name from vars)
2. copying a list of files from controller to host using loop
   (get the list of file names from vars)
   (the actual files will be stored in ./roles/web/files)
   (will be executed using Handlers)

- Restart the service of the installed package
  (will be executed using Handlers chaining)

TEMPLATES

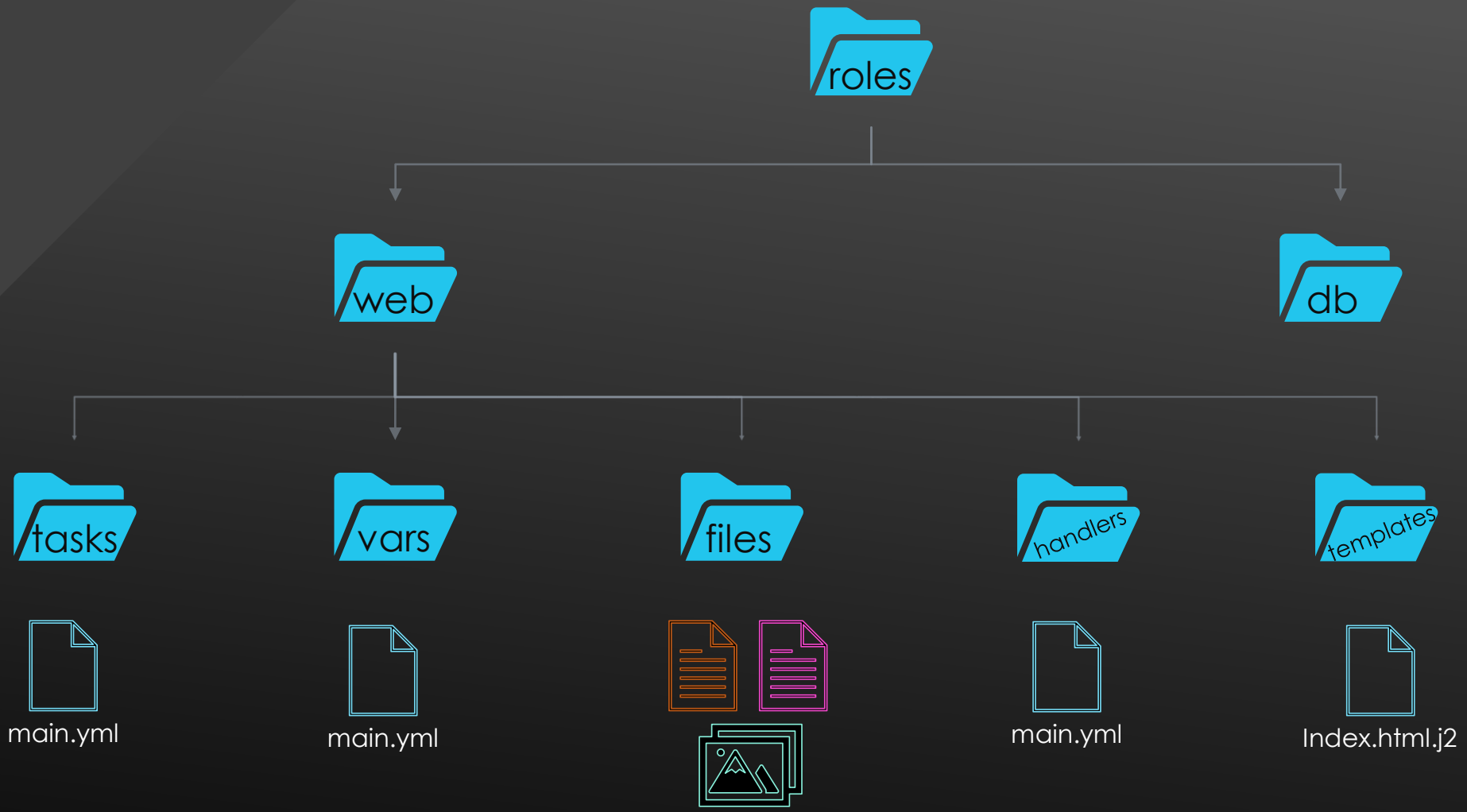
Templates: Ansible uses Jinja2 templates to create dynamic content at the controller end, and render it as static content at the host end.

index.html.j2

```
<html>
<body>
<h1> {{ my_message }} <h1>
</body>
</html>
```

index.html.j2

```
<html>
<body>
<h1> {{ my_message }} <h1>
</body>
</html>
```


roles
web
db
tasks
vars
files
handlers
templates
main.yml
main.yml
main.yml
Index.html.j2

TEMPLATES

- Create your first role with name (web)

- The task book will include:

1. installing a package
   (get the package name from vars)
2. Copying a file from controller to host using template
   (get the template name & template message from vars)
   (the actual template file will be stored in ./roles/web/templates)
   (will also notify the restart handler)
3. copying a list of files from controller to host using loop
   (get the list of file names from vars)
   (the actual files will be stored in ./roles/web/files)
   (will be executed using Handlers)

- Restart the service of the installed package
  (will be executed using Handlers chaining)

ANSIBLE-VAULT

Ansible-Vault: provides a way to encrypt and manage sensitive data such as passwords.

Example:

```
- name: my play with vault
  hosts: all
  vars:
    user_name: ahmed
  var_files:
    - ./passwords.yml
  tasks:
  - name: my task1 with vault
    user:
      name: "{{ user_name }}"
      password: "{{ user_pass }}"
```

passwords.yml

```
user_pass: 123456
```

Commands:

```
$ ansible-vault encrypt ./passwords.yml
$ ansible-playbook my-playbook.yml --ask-vault-pass
```

ANSIBLE-GALAXY

Ansible-Galaxy: refers to the Galaxy website, a free site for finding, downloading, and sharing community developed roles and collections.

Commands:

- $ ansible-galaxy info username.role_name
- $ ansible-galaxy install username.role_name
- $ ansible-galaxy install username.role_name,1.0.0
- $ ansible-galaxy list
- $ ansible-galaxy remove username.role_name

QUESTIONS ?

THANK YOU