ANSIBLE

* Ansible is an open source software that automates software provisioning, configuration management, and application deployment.
* Ansible is commonly used for tasks like software installation, configuration, and system updates across multiple servers or devices in a network.
* Orchestration, Security and compliance.
* Uses YAML Scripting language which works on KEY-VALUE PAIR
* Ansible GUI is called as Ansible Tower. It was just Drag and Drop.
* It helps reduce manual work, improve consistency, and save time in managing complex environments.

# The Keys Features of Ansible:

Agentless :There is no software or agent to be installed on the client that communicates back to the server.

Simple and extensible: Ansible is written in Python and uses YAML for playbook language, both of which are considered relatively easy to learn.

# PLAYBOOK:

Ansible playbooks are a way to send commands to remote computers in a scripted way. Instead of using Ansible commands individually to remotely configure computers from the command line, you can configure entire complex environments by passing a script to one or more systems.

# WHY ANSIBLE:

While managing the multiple servers its hard to keep their configuration identical. If you have multiple servers which needs to configure the same setup in all. while doing the one to one

server their might be a chances to miss some configuration steps in some servers. Thats why automation tools come into play! The automation tools like Ansible, Chef, Puppet and

SaltStack all are based on a same principle.

DESCRIBE THE DESIRED STATE OF THE SYSTEM

# MASTER-SLAVE CONCEPT :

STEP-1: LAUNCH 5 INSTANCE (1-MASTER, 4-SLAVE)

STEP-2: INSTALL ANSIBLE, PYTHON AND PIP ON MASTER SERVER

amazon-linux-extras install ansible2 -y yum install python-pip -y

STEP-3: ADD ANSIBLE USER IN ANSIBLE SERVER (useradd ansible)

STEP-4: SET A PASSWORD TO USER IN ANSIBLE SERVER (passwd ansible)

STEP-5: GIVE ROOT PERMISSIONS TO ANSIBLE USER

visudo ---> 100 line (100gg)

add these second line (ansible ALL=(ALL) NOPASSWD: ALL) now save & quit from the flle

STEP-6: NOW WE HAVE TO SAY YES TO PASSWORD AUTHNETICATION

vi /etc/ssh/sshd\_config > 63 line (63gg)

change the password authentication from no to yes STEP-7: RESTART SSHD (systemctl restart sshd)

# NOTE: REPETE ALL THESE STEPS ON ALL SLAVE SERVERS FROM STEP-3 TO STEP-7

STEP-8: LOGIN AS ANSIBLE USER (su - ansible)

STEP-9: GENERATE A KEY IN ANSIBLE USER ON MASTER SERVER (ssh-keygen)

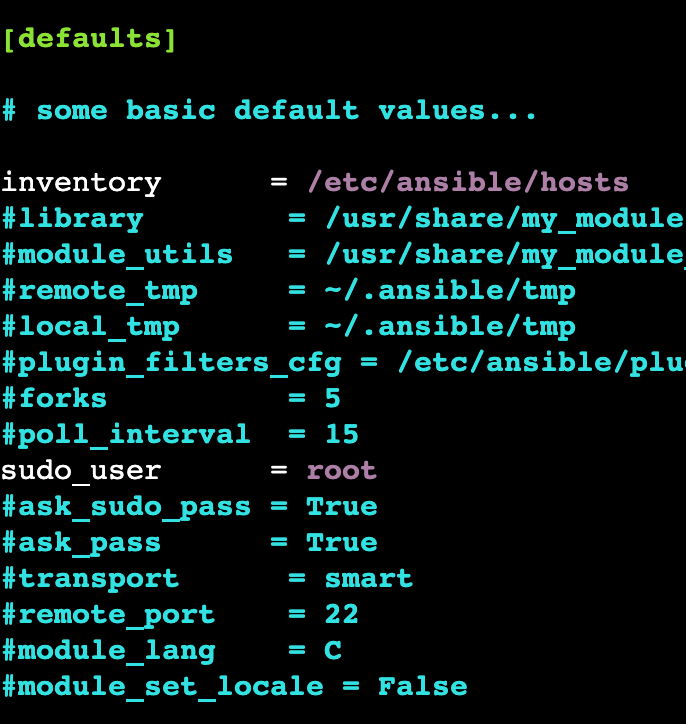
It will generate 2 keys (public & private)

STEP-10: COPY THE PUBLIC KEY TO ALL SLAVE SERVERS (ssh-copy-id ansible@slave\_ip)

STEP-11: LOGIN TO THE SLAVE (ssh ansible@slave\_ip)

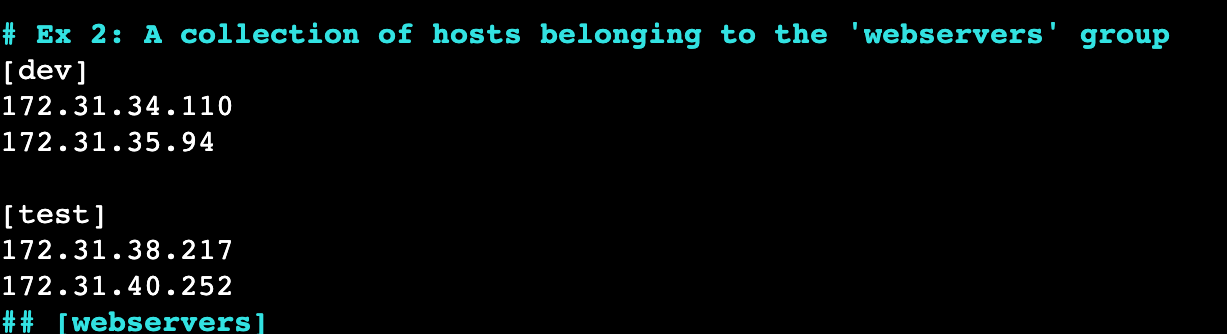
# NOW ITS TIME TO CHANGE ANSIBLE CONFIGURATIONS:

STEP-12: EXIT FROM ANSIBLE USER ON MASTER SERVER (exit), so we will be in root user

STEP-13 : ENBALE ANSIBLE INVENTORY AND SUDO USER (vi /etc/ansible/ansible.cfg)

save & quit from the file

STEP-14: ADD INVENTORIES (vi /etc/ansible/hosts)



HERE dev & test is the group names save & quit from the file

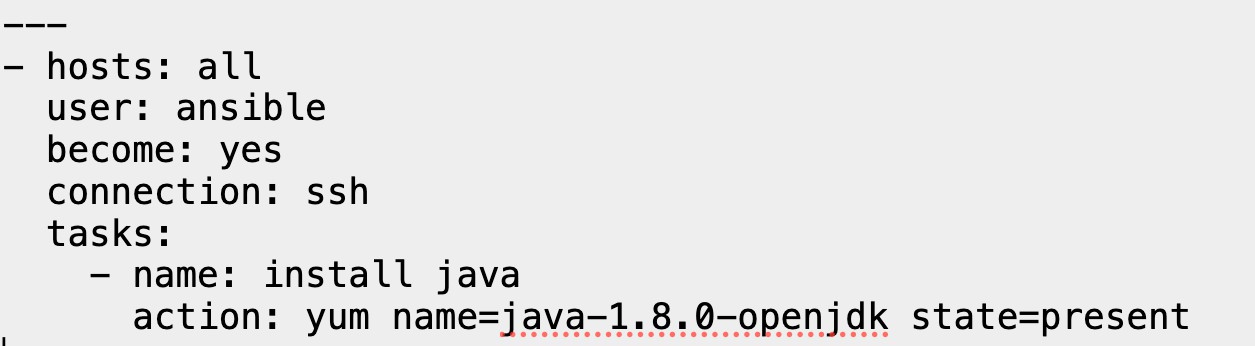
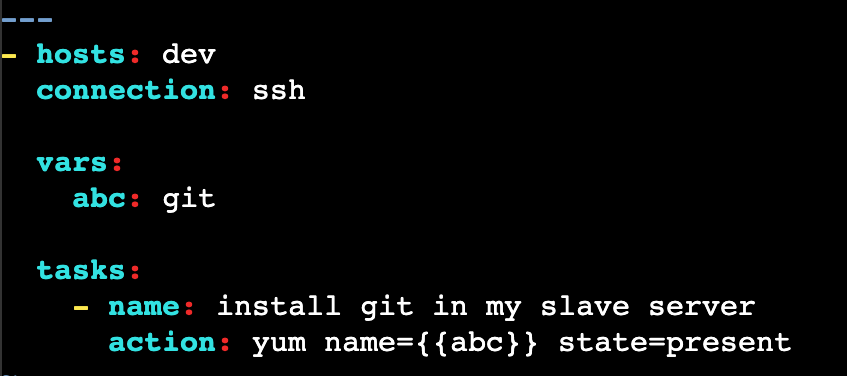
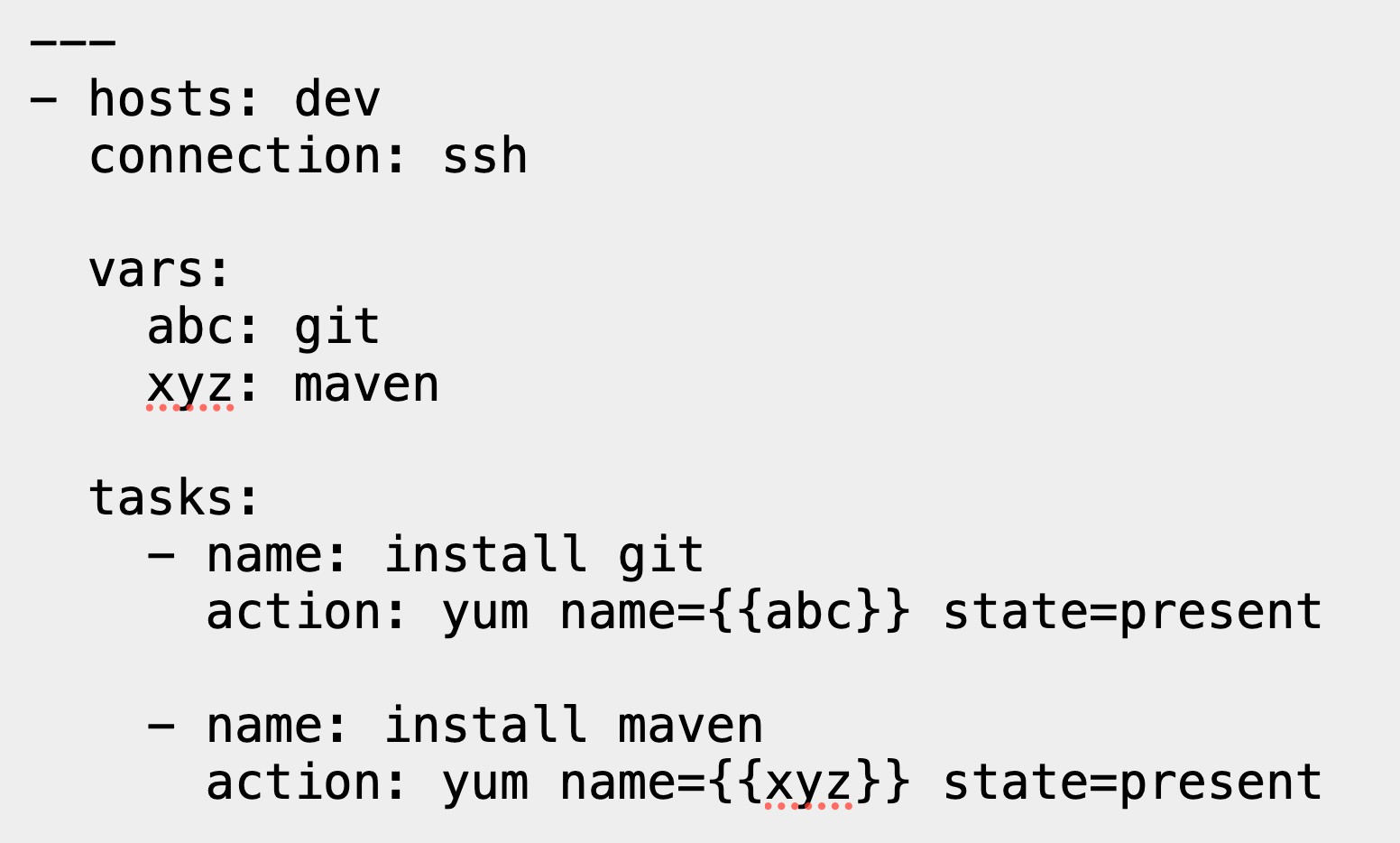
STEP-15: TO CHECK WITH SLAVE SERVER CONNECTION

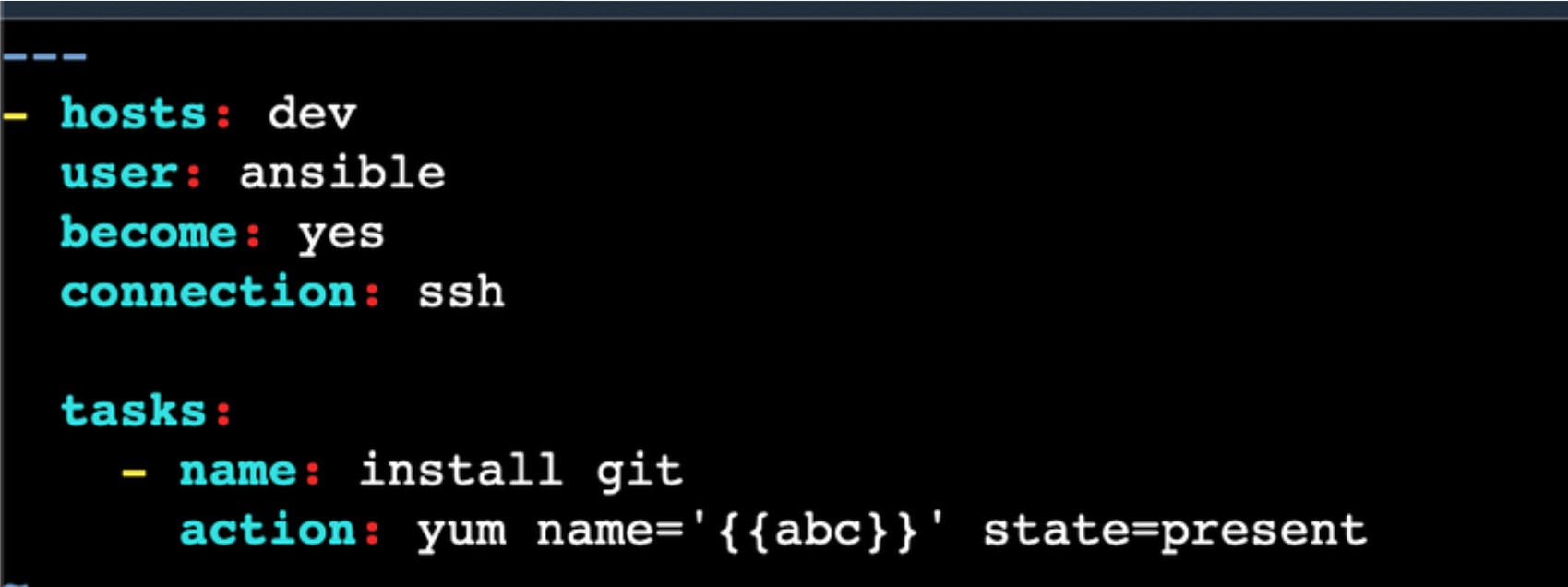
login as ansible user : su - ansible

to check the connection : ansible all --list-hosts

* To see the list of hosts in inventory : ansible all --list-hosts
* To see the list of particular group hosts in inventory : ansible group\_name --list-hosts
* To see the 1st hosts in inventory : ansible all[0] --list-hosts
* To check the network connection between master & slave : ansible all -m ping

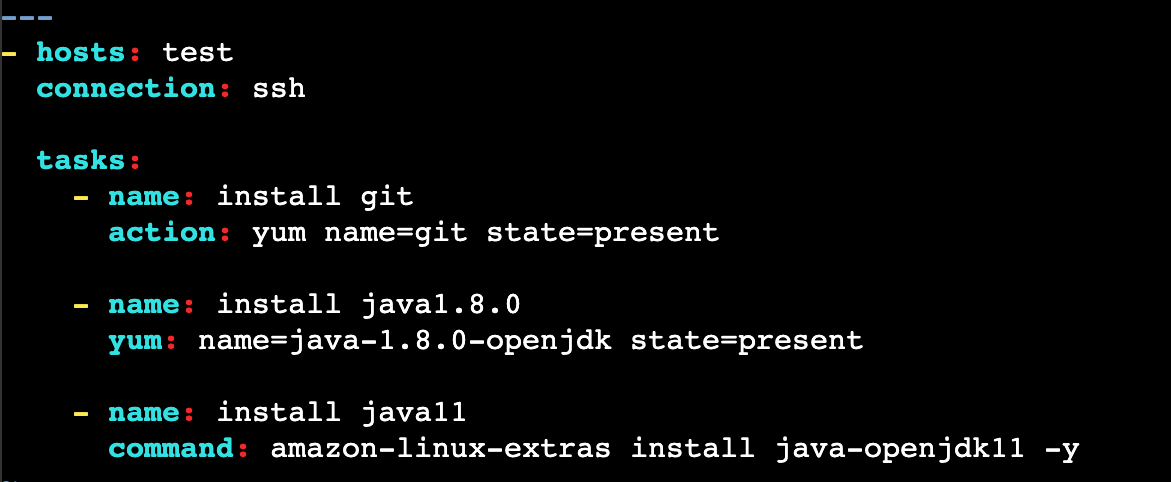
# PLAYBOOKS:

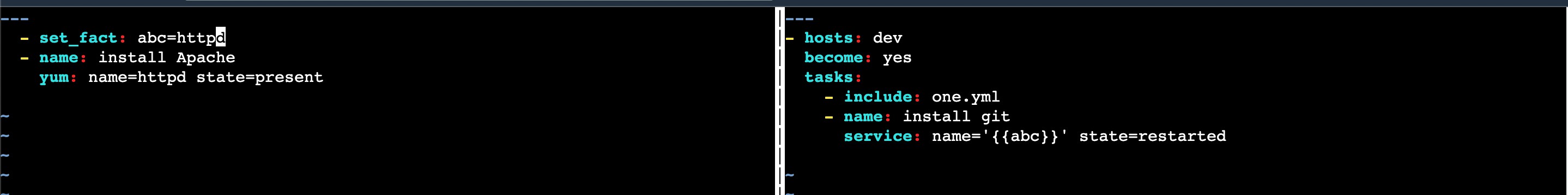
1. Playbooks in ansible are written in YAML language.
2. It is human readable & serialization language commonly used for configuration files.
3. You can write codes consists of vars, tasks, handlers, files, templates and roles.
4. Each playbook is composed of one or more modules in a list.
5. Playbooks are mainly divided into sections like
6. TARGET SECTION: Defines host against which playbooks task has to be executed.
7. VARIABLE SECTION: Defines variables.
8. TASK SECTION: action you are perfoming.
9. WRITE A PLAYBOOK TO INSTALL GIT IN DEV GROUP:
10. WRITE A PLAYBOOK TO INSTALL JAVA1.8.0 ON ALL THE SERVERS
11. WRITE A PLAYBOOK TO INSTALL WEB SERVER & START THE WEB SERVER:
12. WRITE A PLAYBOOK WITH VARIABLE:
13. WRITE A PLAYBOOK WITH MULTIPLE VARIABLES:
14. WRITE A PLAYBOOK TO ADD VARIABLES DYNAMICALLY:

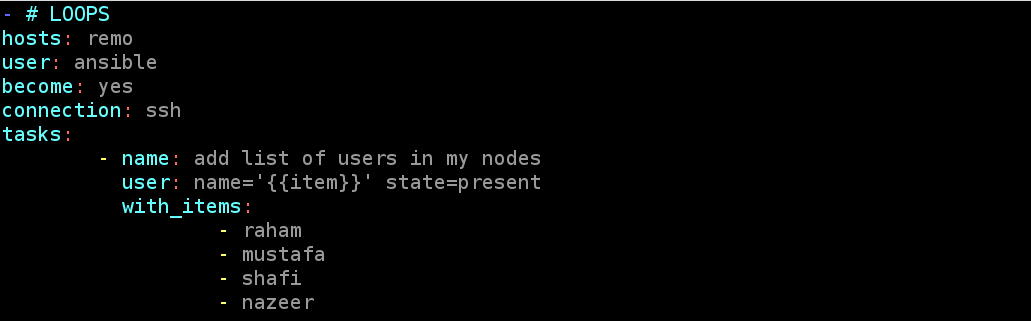
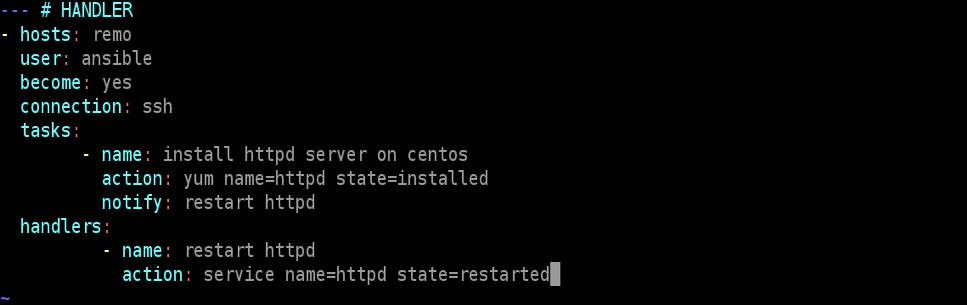
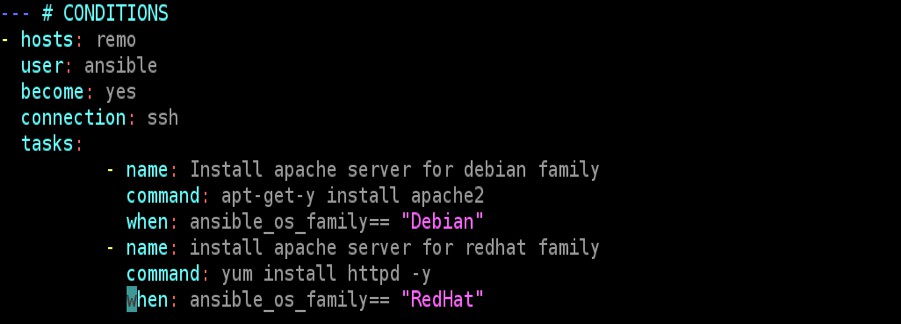


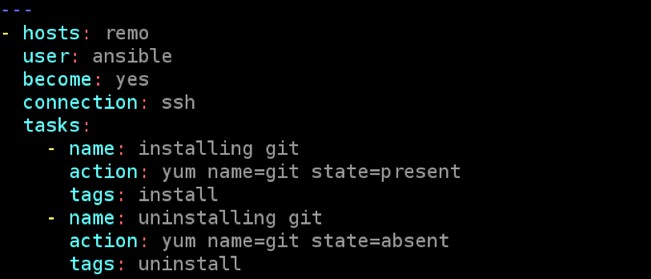
for single var: ansible-playbook one.yml --extra-vars "abc=git"

for multiple vars: ansible-playbook one.yml --extra-vars "abc=git def=maven"

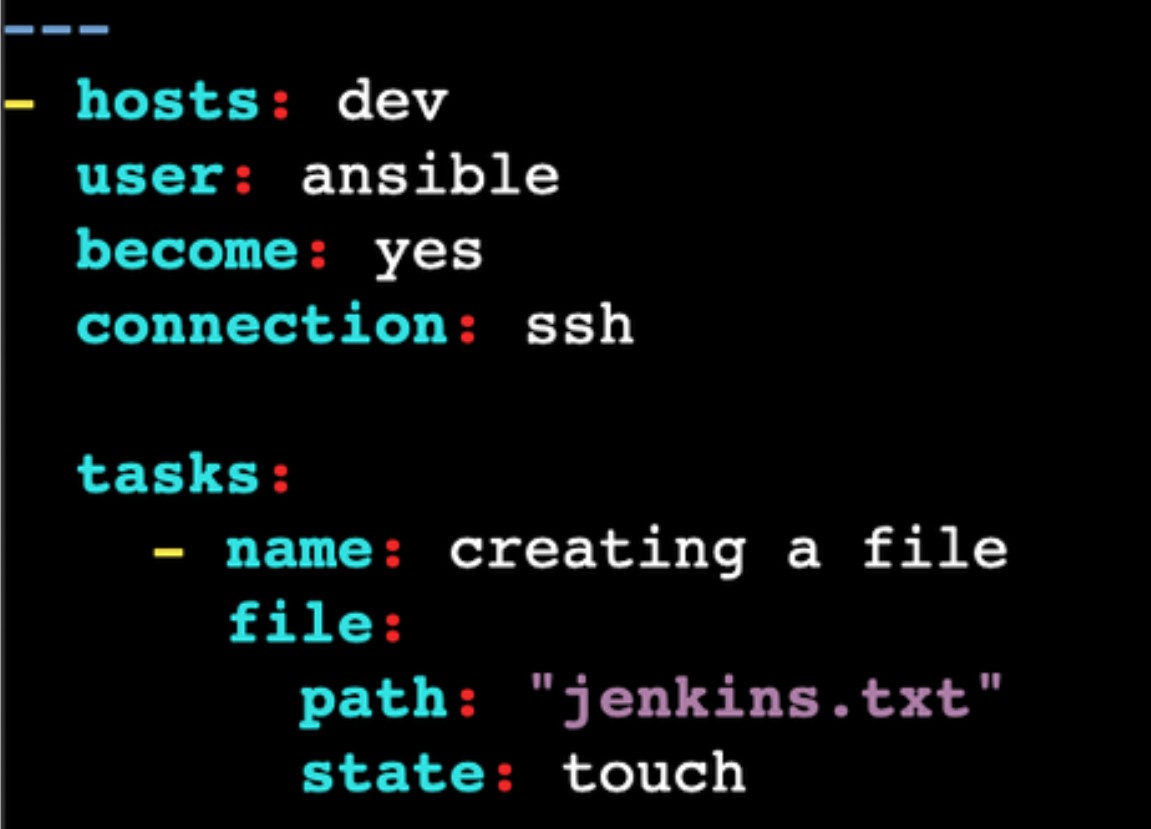
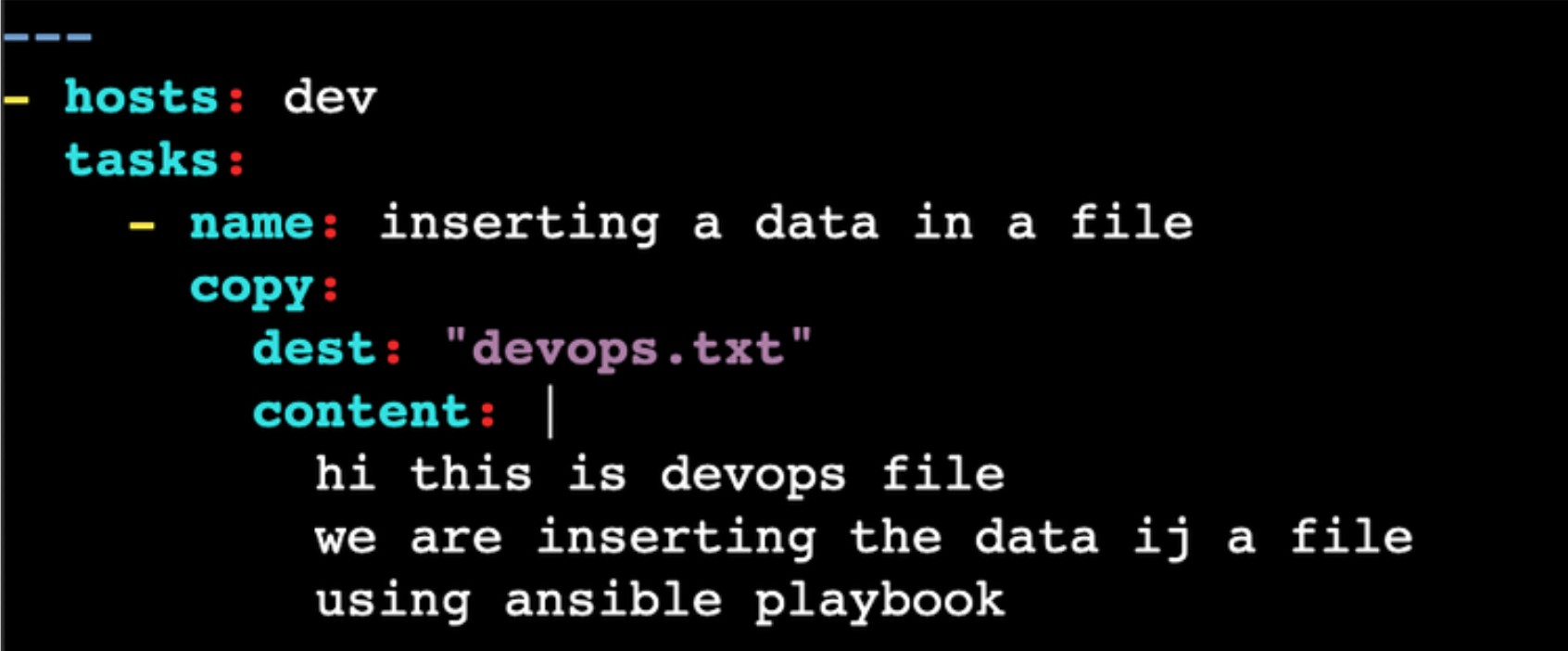
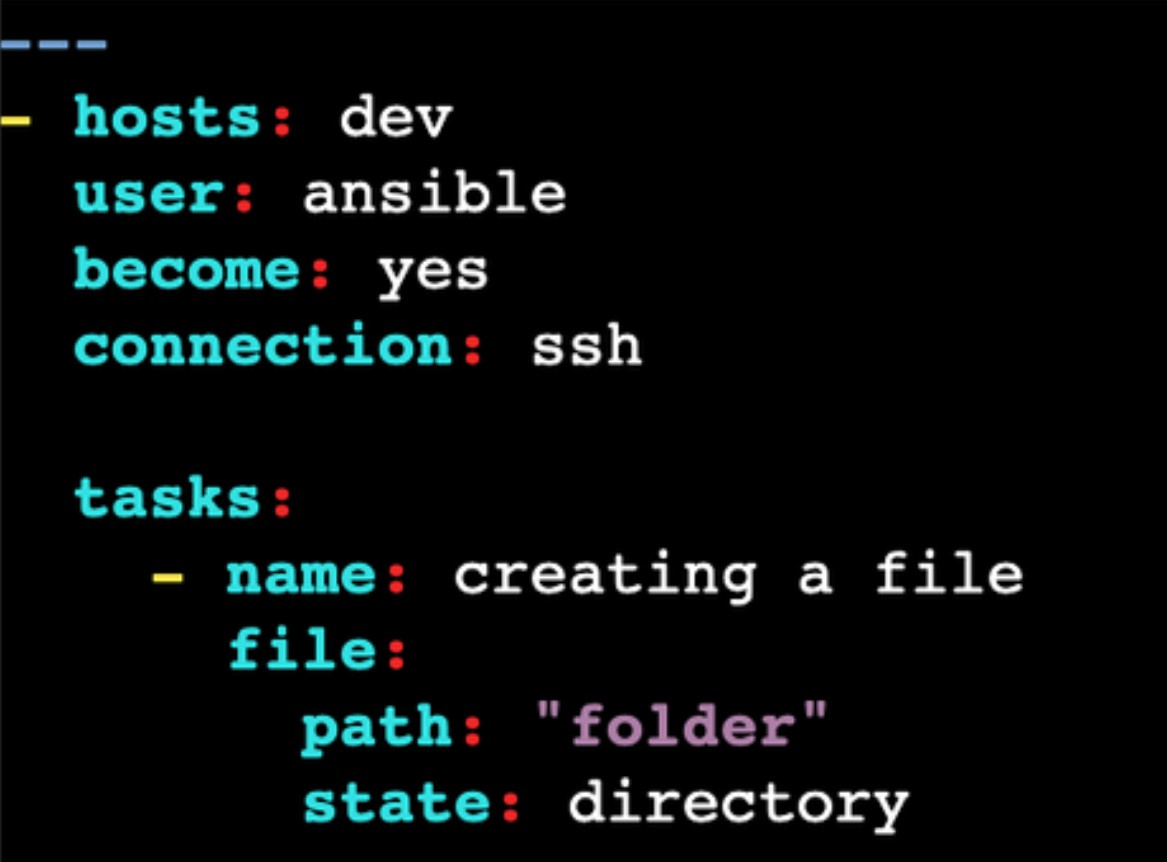
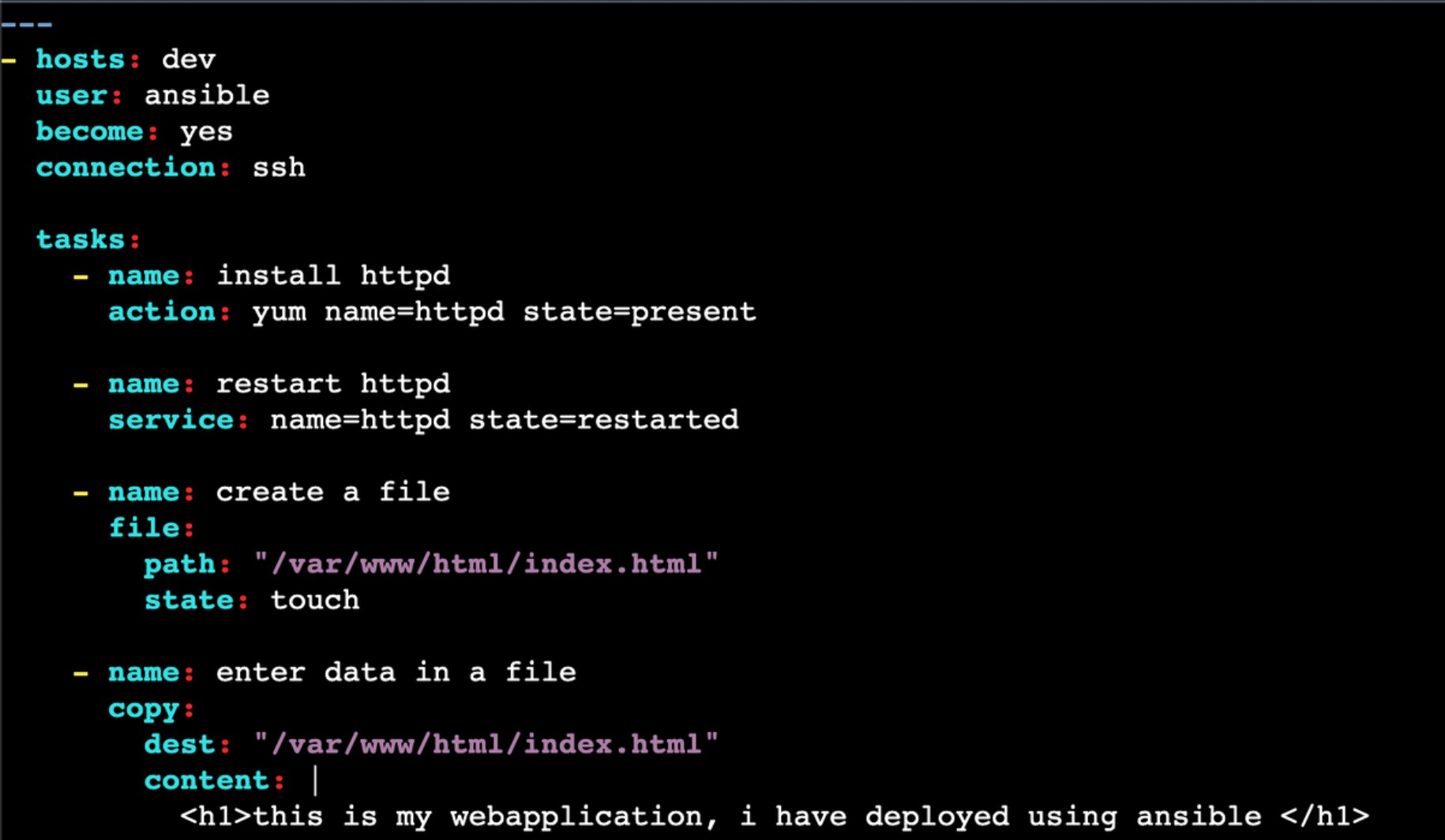
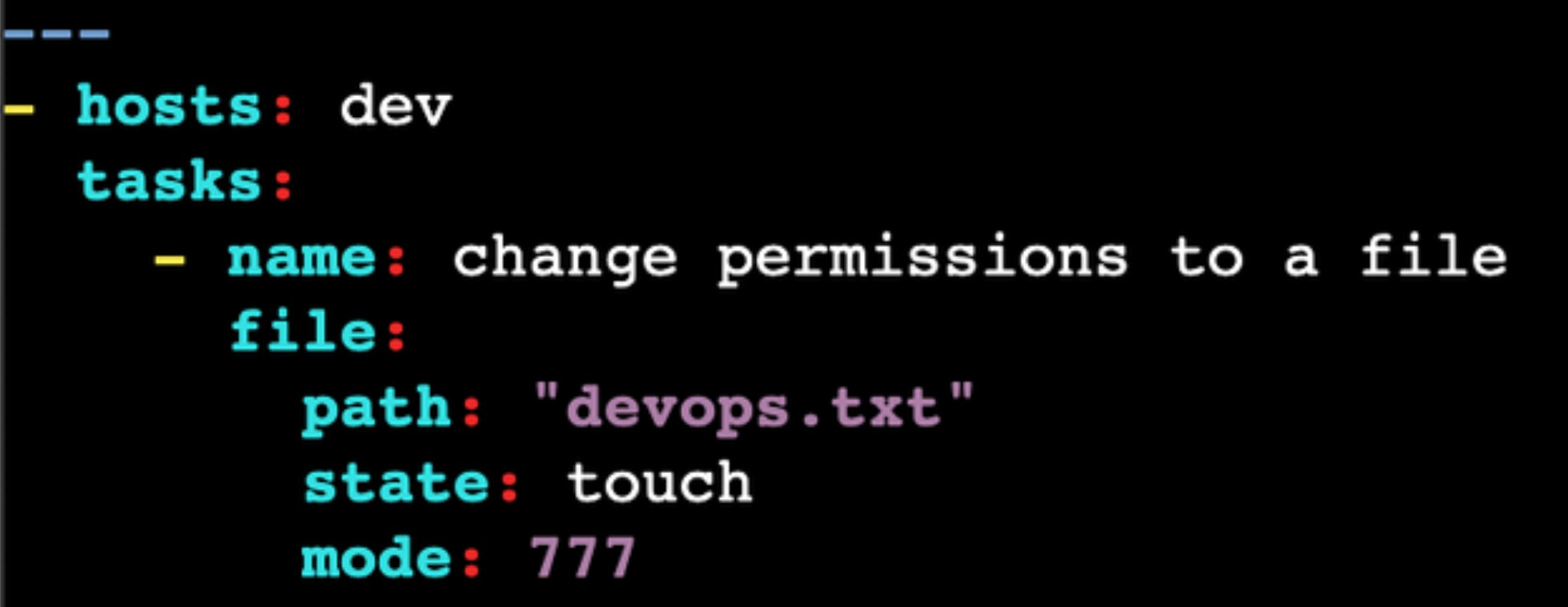
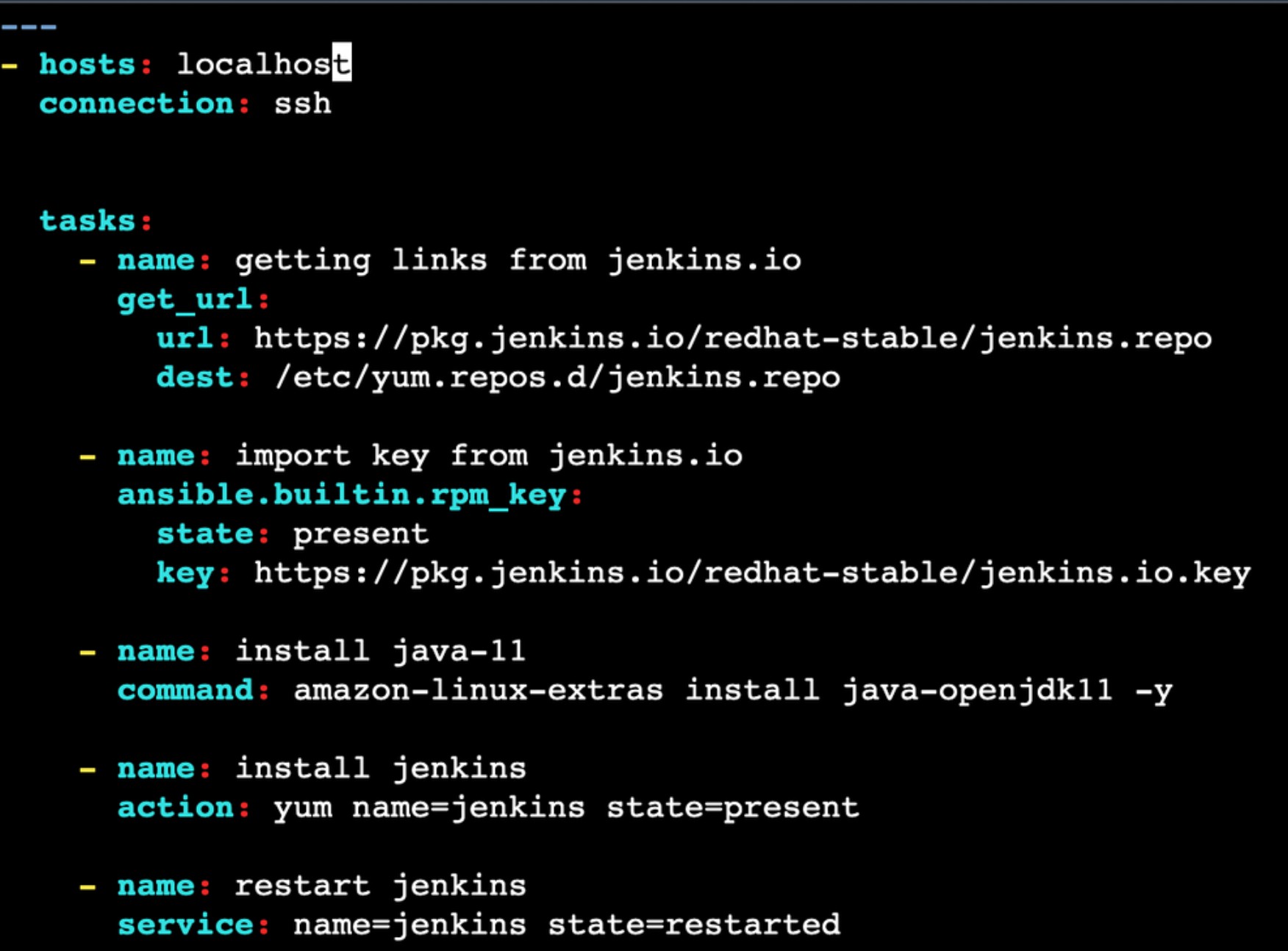
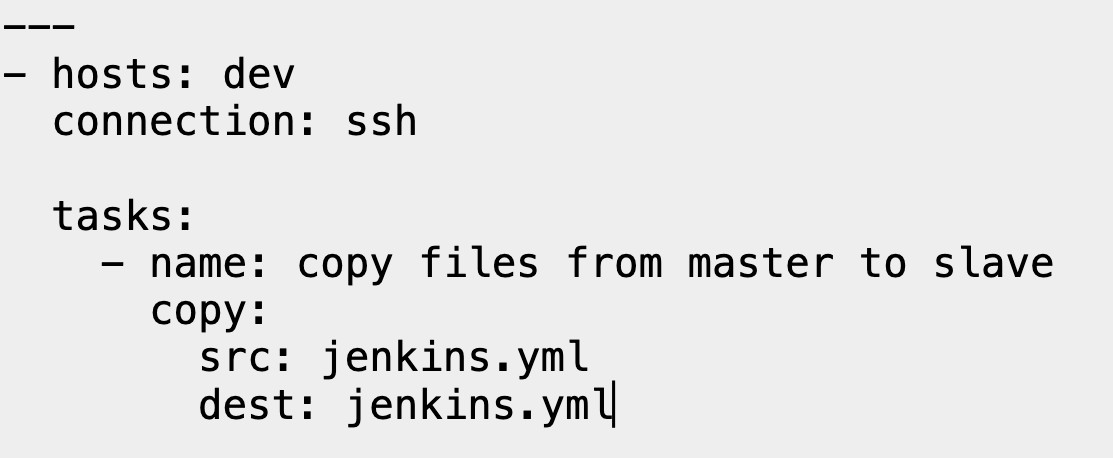
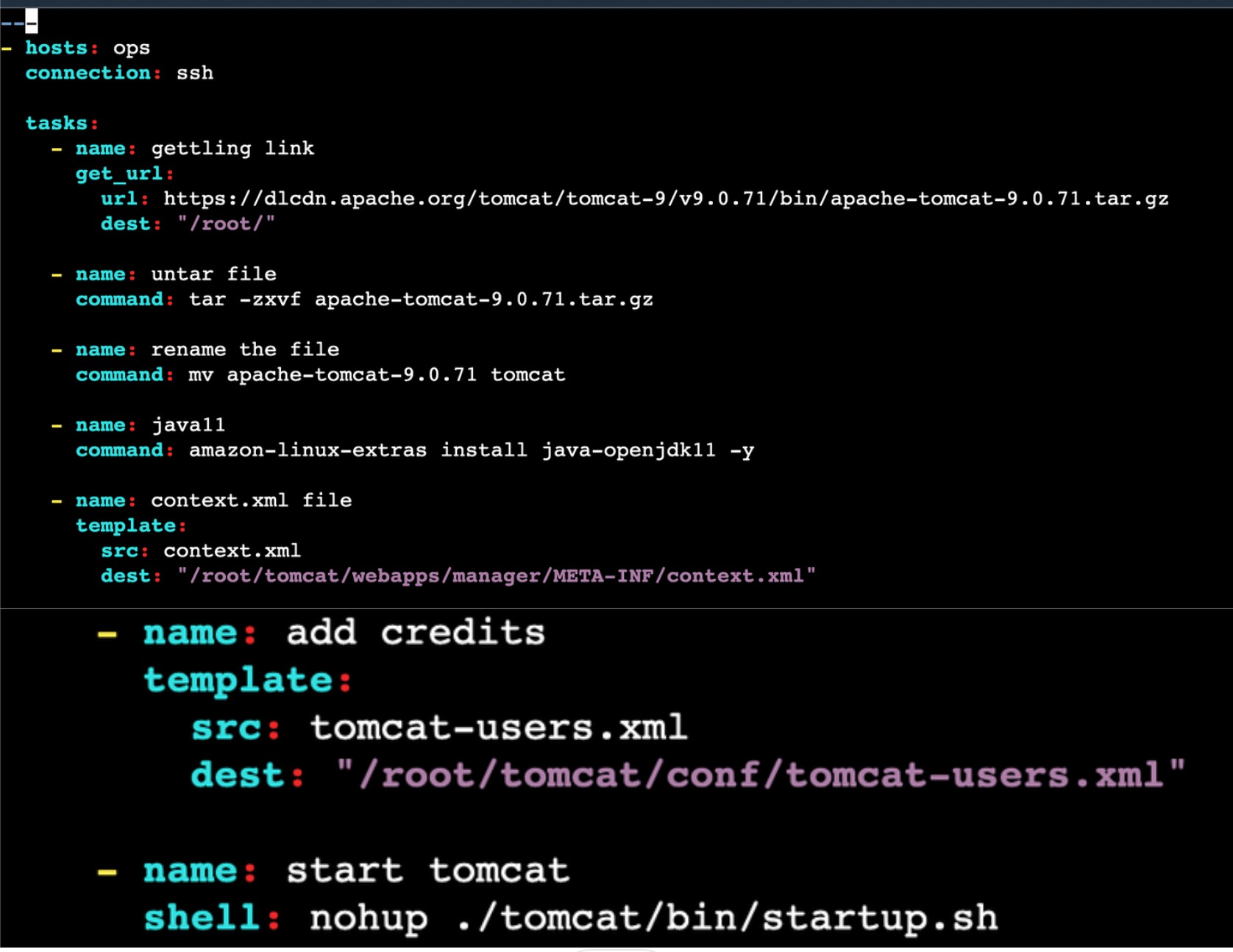
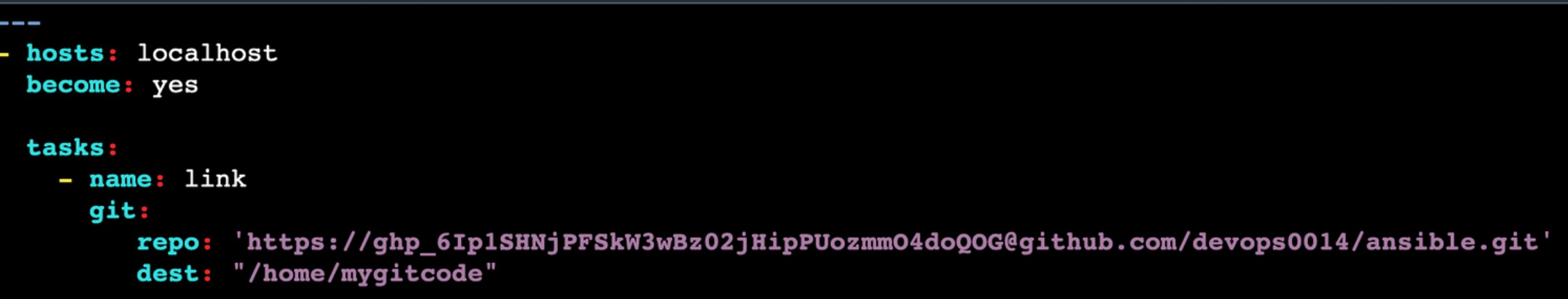
1. WRITE A PLAYBOOK TO INSTALL PACKAGES ON DIFFERENT WAYS:
2. Passing a Varaible flle - A Varaible can be deflned in a variable flle and can be passed to a playbook using the include

one.yml two.yml

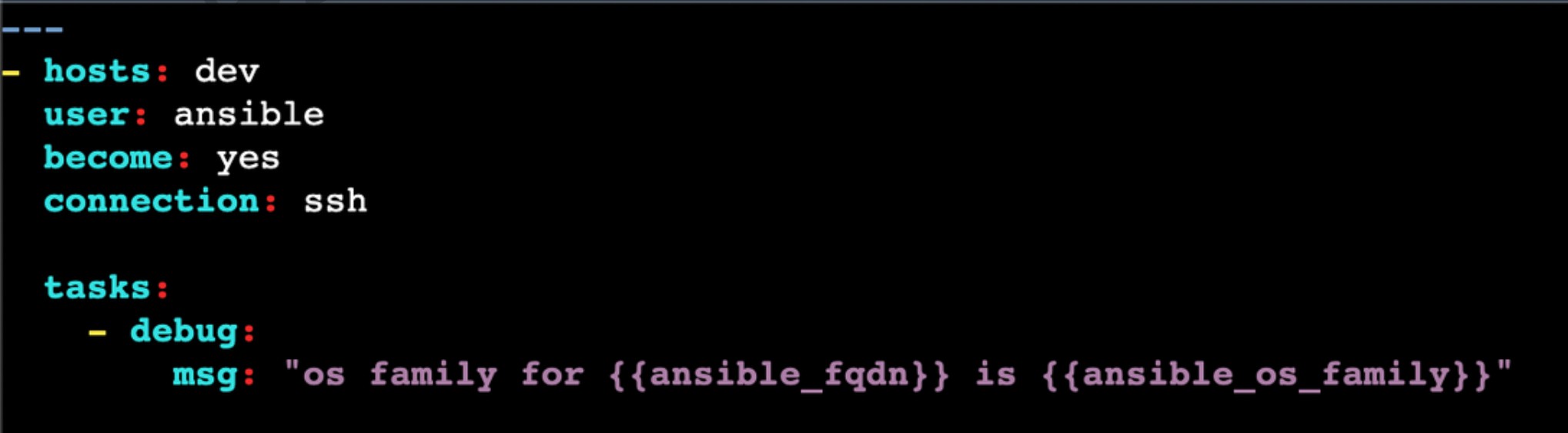
1. WRITE A PLAYBOOK TO ADD MULTIPLE USERS:
2. WRITE A PLAYBOOK USING HANDLERS:
3. WRITE A PLAYBOOK USING CONDITIONS:
4. WRITE A PLAYBOOK USING TAGS:

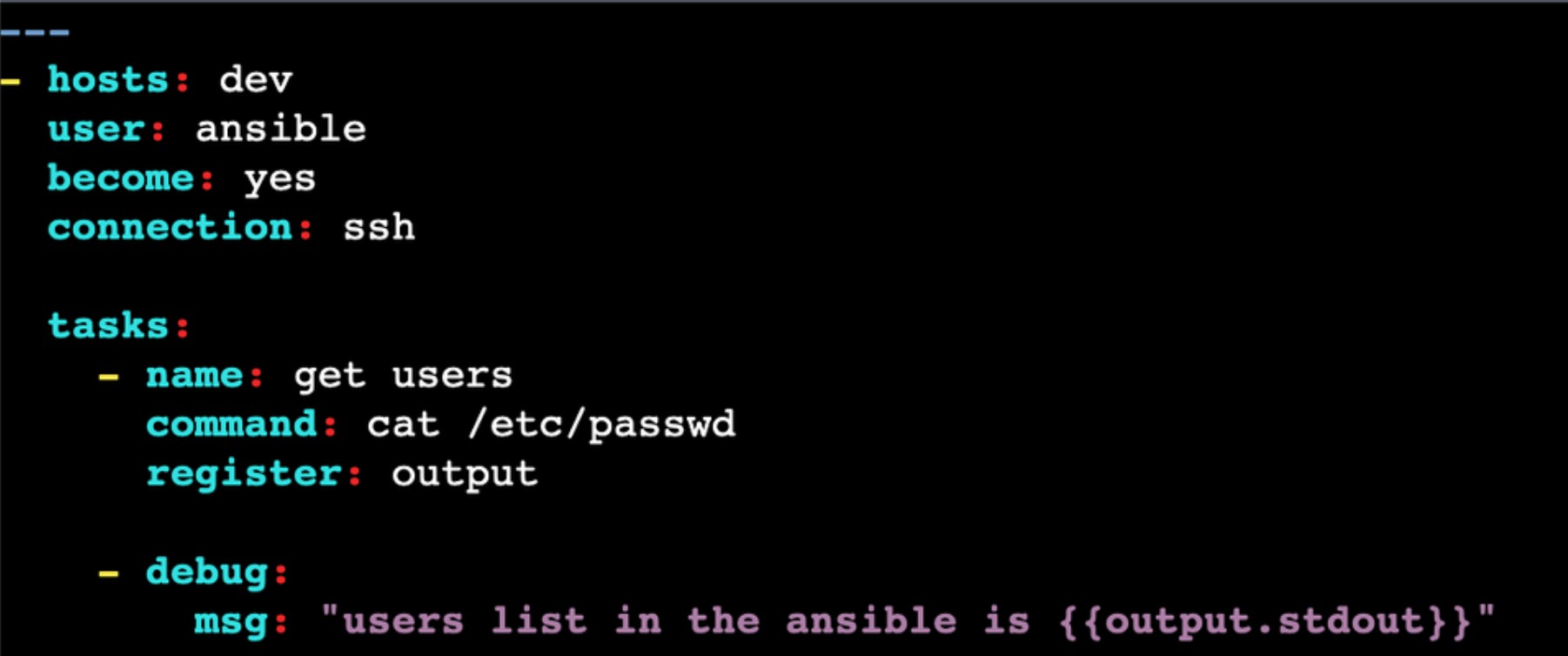


* + TO EXECUTE A SINGLE TASK: ansible-playbook abc.yml --tags tagname
  + TO EXECUTE A MULTIPLE TASK: ansible-playbook abc.yml --tags tagname1,tagname2
  + TO SKIP A TASK: ansible-playbook abc.yml --skip-tags “uninstall”

1. WRITE A PLAYBOOK FOR CREATING A FILE:
2. WRITE A PLAYBOOK FOR CREATING A FILE:
3. WRITE A PLAYBOOK FOR ENTERING A DATA IN A FILE:
4. WRITE A PLAYBOOK TO CHANGE THE PERMISSIONS OF A FILE:
5. WRITE A PLAYBOOK TO DEPLOY A WEBSITE:
6. WRITE A PLAYBOOK TO SETUP JENKINS:
7. WRITE A PLAYBOOK TO SETUP TOMCAT:
8. WRITE A PLAYBOOK TO COPY A FILE:
9. WRITE A PLAYBOOK TO GET A CODE FROM GITHUB(PUBLIC-REPO):
10. WRITE A PLAYBOOK TO GET A CODE FROM GITHUB(PRIVATE-REPO):

SYNTAX: token@github.com/username/repo.git

1. WRITE A PLAYBOOK USING DEBUG MODULE:
2. WRITE A PLAYBOOK TO SEE LIST OF USERS:



## ANSIBLE ROLES:

Ansible roles are a way to organize and structure your Ansible playbooks in a more modular and reusable manner. They provide a means to group related tasks, variables, and files

together, making your playbooks more organized and easier to manage. Roles can be thought of as a collection of tasks, templates, and variables that are designed for a specific purpose or function, such as setting up a web server, configuring a database, or managing a specific

application.

1. Create the role directory structure:

You can create a role using the ansible-galaxy command or by manually creating the directory structure. Let's create the directory structure manually:

roles/

├── webserver/

├── tasks/

│ └── main.yml

├── handlers/

│ └── main.yml

├── templates/

│ └── index.html.j2

├── vars/

│ └── main.yml

├── defaults/

│ └── main.yml

└── meta/

└── main.yml

1. Define the role tasks in **roles/webserver/tasks/main.yml:**

---

* + name: Install Apache web server yum: name=httpd state=present
  + name: Ensure Apache service is running service: name=httpd state=started

1. Define role variables in **roles/webserver/vars/main.yml:**

---

apache\_port: 80

1. Create a handler in **roles/webserver/handlers/main.yml** (optional) to restart the Apache service if needed:

---

* + name: Restart Apache

service: name=httpd state=restarted

1. Create a template for the index page in **roles/webserver/templates/index.html.j2** (optional):

<!DOCTYPE html>

<html>

<head>

<title>Welcome to My Website</title>

</head>

<body>

<h1>Welcome to my web server!</h1>

</body>

</html>

1. Specify any necessary metadata for the role in **roles/webserver/meta/main.yml:**

---

dependencies: []

1. With this role structure in place, you can now use the webserver **role** in your Ansible playbook by specifying it in the roles section. For example:

---

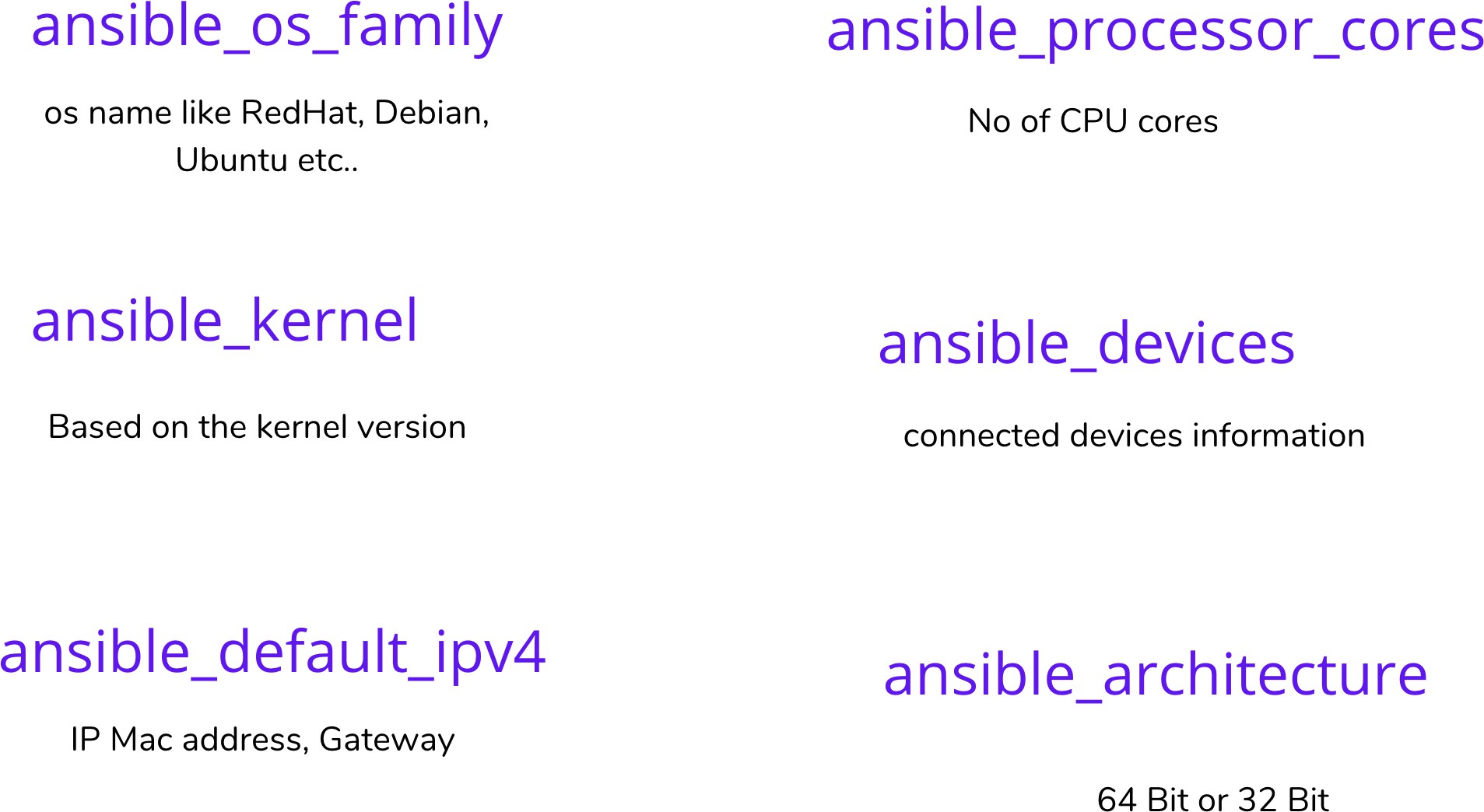
* + name: Configure Web Server hosts: web\_servers

become: yes

roles:

- webserver

# ANSIBLE SETUP MODULES:



After executing a playbook, if you want to see the output in json format ansible -m setup private\_ip

if you want to apply a see particular output, you can apply filter.

 ansible -m setup -a "filter=ansible\_os\_family" private\_ip

 ansible -m setup -a "filter=ansible\_devices" private\_ip

 ansible -m setup -a "filter=ansible\_kernel" private\_ip

# ADHOC COMMANDS:

Ansible ad-hoc commands are quick, one-time instructions you give to Ansible on the command line to perform simple tasks on remote servers. These commands are not part of Ansible's usual automation playbook and are typically used for tasks like running a single

command, checking server status, or making minor changes without writing full automation scripts. Ad-hoc commands are handy for immediate, one-off tasks.

 ansible remo -a “ls” [remo: Group name, -a: argument, ls: command]

 ansible remo [0] -a “touch file1”

 ansible all -a “touch file2”

 ansible remo -a “sudo yum install httpd -y”

 ansible remo -ba “yum install httpd -y” (b: become you will become sudo user)

 ansible remo -ba “yum remove httpd -y”

## ANSIBLE MODULES:

Ansible modules are like individual commands or tools that perform specific tasks on target machines. They are the building blocks for Ansible automation. Modules can do things like create files, install software, restart services, and more.

 ansible remo -b -m yum -a “pkg=httpd state=present” (install: present)

 ansible remo -b -m yum -a “pkg=httpd state=latest” (update: latest)

 ansible remo -b -m yum -a “pkg=httpd state=absent” (uninstall: absent)

 ansible remo -b -m service -a “name=httpd state=started” (started: start)

 ansible remo -b -m user -a “name=raj” (to check go to that servers and sudo cat

/etc/passwd).

 ansible remo -b -m copy -a “src=filename dest=/tmp” (to check go to that server and give ls /tmp)

## ANSIBLE GALAXY:

Ansible Galaxy is a website and command-line tool for sharing and managing collections of Ansible roles and playbooks. In simple terms, it's like an online marketplace or repository for Ansible automation content.

 ansible-galaxy init raham

 ansible-galaxy search elasticsearch

 ansible-galaxy search elasticsearch --author alikins

 ansible-galaxy install alikns.elasticsearch

 cd /home/ansible/.ansible/roles/

## ANSIBLE VALUT:

Ansible Vault is a feature of the Ansible automation tool that is used to securely encrypt sensitive data, such as passwords, API keys, and other secrets, so that they can be safely stored and shared within Ansible playbooks and roles.

USE CASES:

 Encryption

 Secure Storage

 Password Prompt

 Automation

 Secrets Management

COMMANDS FOR ANSIBLE PASSWORD

 ansible-vault create vault.yml : creating a new encrypted playbook.

 ansible-vault edit vault.yml : Edit the encrypted playbook.

 ansible-vault rekey vault.yml : To edit the password.

 ansible-vault view vault.yml : To view the playbook without decrypt.

 ansible-vault encrypt vault.yml : To encrypt the existing playbook.

 ansible-vault decrypt vault.yml : To decrypt the encrypted playbook.