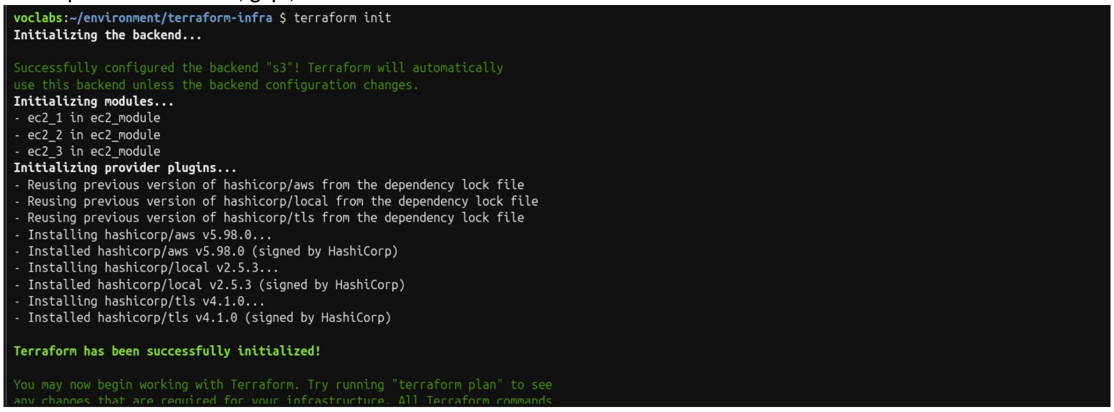
What have I learned in creation of the project

terraform init

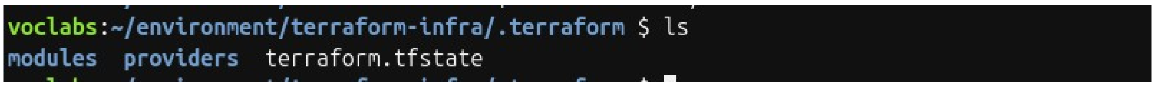
1- it scans and initialize all variable resources of modules like ec2_1 , ec2_2 ,ec2_3
2- It initializes provider like aws , gcp , azure ……………



```
voclabs:~/environment/terraform-infra $ terraform init
Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing modules...
- ec2_1 in ec2_module
- ec2_2 in ec2_module
- ec2_3 in ec2_module
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/tls from the dependency lock file
- Installing hashicorp/aws v5.98.0...
- Installed hashicorp/aws v5.98.0 (signed by HashiCorp)
- Installing hashicorp/local v2.5.3...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
- Installing hashicorp/tls v4.1.0...
- Installed hashicorp/tls v4.1.0 (signed by HashiCorp)

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
```

if you created a new resource instance module, you will need to rerun terraform init.

3- it collect all these data and put them in .terraform

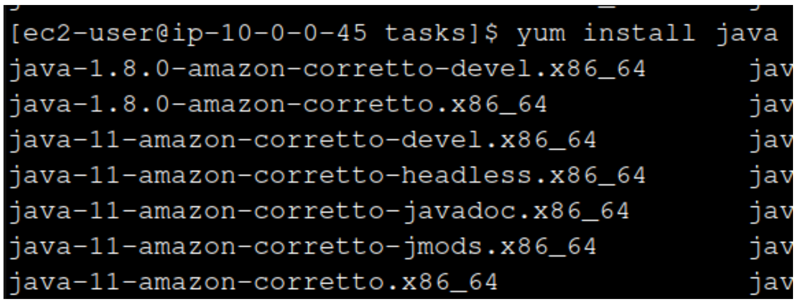


```
voclabs:~/environment/terraform-infra/.terraform $ ls
modules  providers  terraform.tfstate
```

INSTALLATION OF JAVA in Amazon EC2 instance

NOTE:

corretto	→ Java 11 runtime only
corretto-headless	→ Java runtime without GUI stuff
devel	→ Java 11 **runtime + compiler**
doc	**→ documentation**



```
[ec2-user@ip-10-0-0-45 tasks]$ yum install java
java-1.8.0-amazon-corretto-devel.x86_64      jav
java-1.8.0-amazon-corretto.x86_64            jav
java-11-amazon-corretto-devel.x86_64         jav
java-11-amazon-corretto-headless.x86_64      jav
java-11-amazon-corretto-javadoc.x86_64       jav
java-11-amazon-corretto-jmods.x86_64         jav
java-11-amazon-corretto.x86_64               jav
```

to install jenkins you need java → corretto light weight and you don't need compiler or GUI.
java-17-amazon-corretto.x86_64

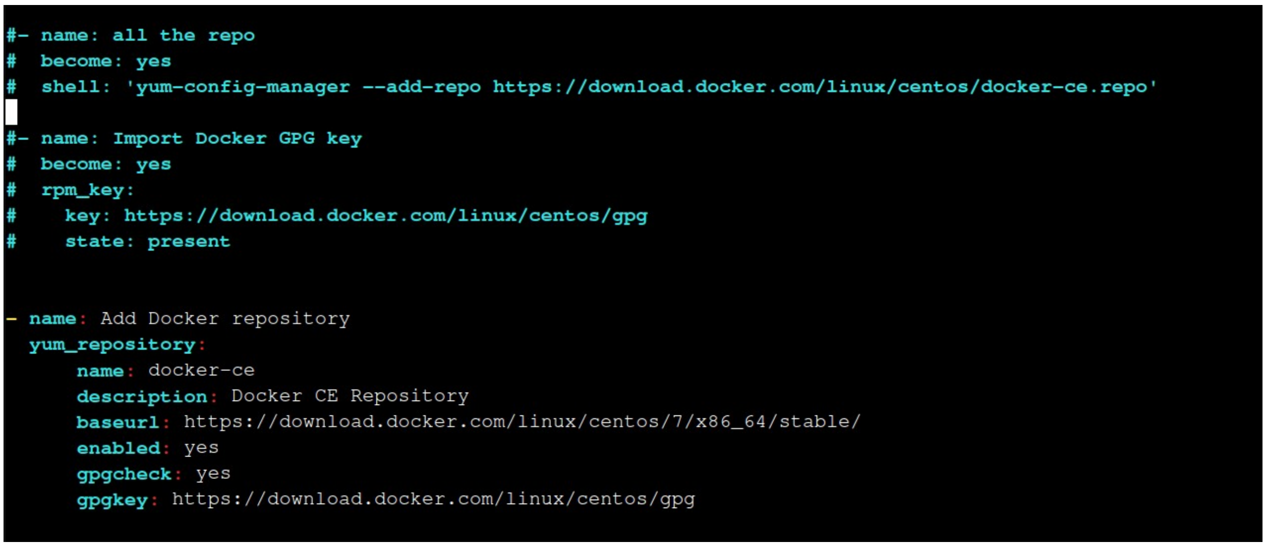
To install docker on redhat

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install -y docker-ce docker-ce-cli containerd.io
```

append the docker repo and install docker:

the commented commands didn’t make me able to install docker-ce , docker-ce-cli and containerd

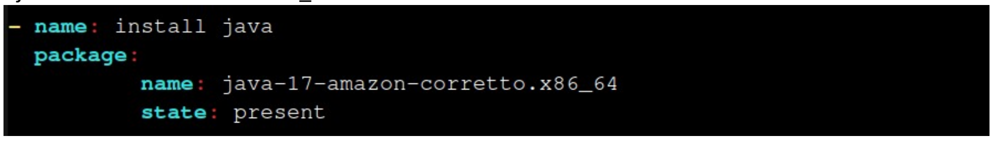
- I think because the key which installed need to me integrated with the repo that we installed , but this way makes them separated.
The `yum_repository` module with `gpgkey` downloads and imports the key automatically.



```
#- name: all the repo
#  become: yes
#  shell: 'yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo'

#- name: Import Docker GPG key
#  become: yes
#  rpm_key:
#    key: https://download.docker.com/linux/centos/gpg
#    state: present


- name: Add Docker repository
  yum_repository:
      name: docker-ce
      description: Docker CE Repository
      baseurl: https://download.docker.com/linux/centos/7/x86_64/stable/
      enabled: yes
      gpgcheck: yes
      gpgkey: https://download.docker.com/linux/centos/gpg
```

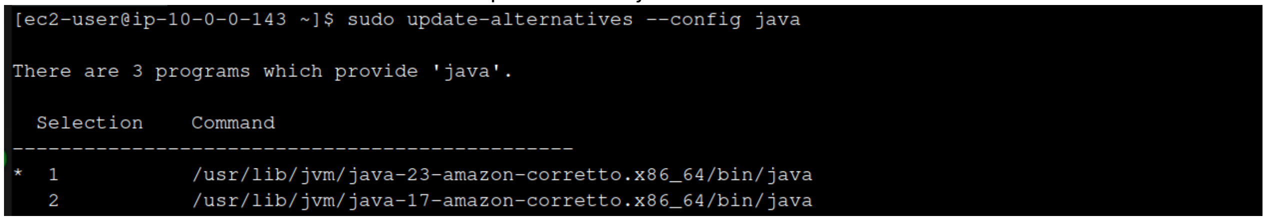
----
to operate jenkins on the server → it must work on java 17

you install java java-17-amazon-corretto.x86_64



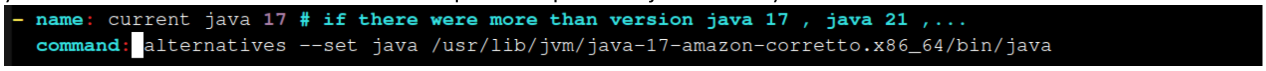
```
- name: install java
  package:
          name: java-17-amazon-corretto.x86_64
          state: present
```

if there are 100 servers and all of them have multiple versions of java



```
[ec2-user@ip-10-0-0-143 ~]$ sudo update-alternatives --config java

There are 3 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
*  1           /usr/lib/jvm/java-23-amazon-corretto.x86_64/bin/java
   2           /usr/lib/jvm/java-17-amazon-corretto.x86_64/bin/java
```

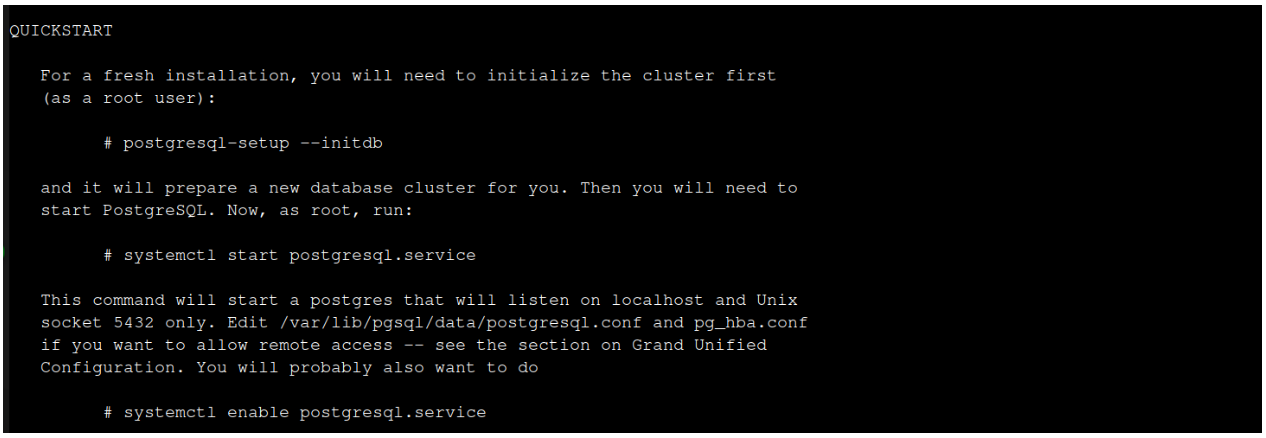
you can run alternative command with --set option and paste the java version you need



```
- name: current java 17 # if there were more than version java 17 , java 21 ,...
  command: alternatives --set java /usr/lib/jvm/java-17-amazon-corretto.x86_64/bin/java
```

----
when you install postgres-server

createuser CLI command → creates a db user inside postgres

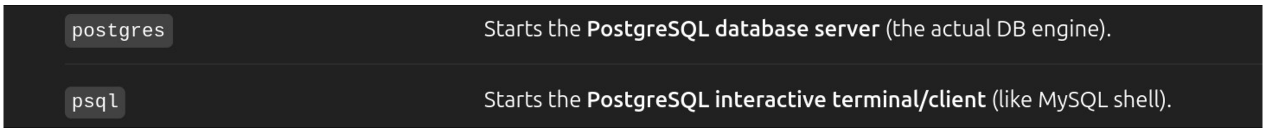


```
QUICKSTART

   For a fresh installation, you will need to initialize the cluster first
   (as a root user):

         # postgresql-setup --initdb

   and it will prepare a new database cluster for you. Then you will need to
   start PostgreSQL. Now, as root, run:

         # systemctl start postgresql.service

   This command will start a postgres that will listen on localhost and Unix
   socket 5432 only. Edit /var/lib/pgsql/data/postgresql.conf and pg_hba.conf
   if you want to allow remote access -- see the section on Grand Unified
   Configuration. You will probably also want to do

         # systemctl enable postgresql.service
```

postgresql-setup --initdb → After postgres installation → run this command , it creates a db cluster for you
/var/lib/pgsql/data in one condition → /var/lib/pgsql/data must be empty

What is the difference between postgres and psql command?

Postgres command is the engine → you login using username and password to enter your postgres db
psql command → you can run SQL queries and instructions in the command line.



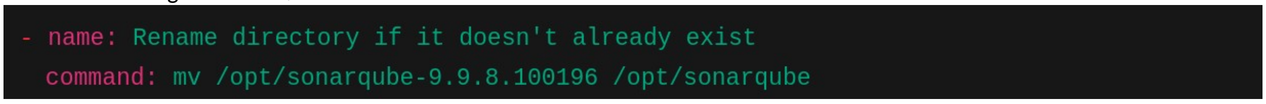
`postgres`	Starts the **PostgreSQL database server** (the actual DB engine).
`psql`	Starts the **PostgreSQL interactive terminal/client** (like MySQL shell).

What have we learned in ansible?

1- adding repos to the package manager → it needs cache_update for package managers like yum and dnf
1- install postgres-server
2- create postgres db cluster
3- create postgres user → sudo -u postgres createuser <db_user> بتستخدم يوزر البوتجري علشان ينفذ حاجة علي الداتابيز
4- get the user password → sudo -u postgres psql -c "ALTER USER <db_user> WITH ENCRYPTED PASSWORD 'password' ";
5- give a db_user all privileges → sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE <db_name> TO <db_user>"

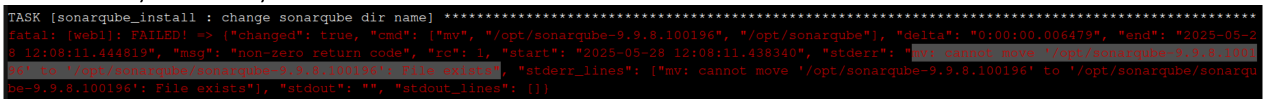
learned with sonarqube
1- wget or ansible module get_url to download zip file on the target servers
2- use 'unarchive' ansible module to unarchive zip file
3- create 'sonar' group
4- create 'sonar' user
5- use file module to give ownership to sonar:sonar user:group on /opt/sonar recursively

command moving directories/files



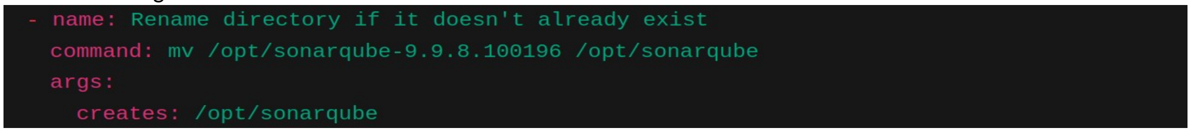
```yaml
- name: Rename directory if it doesn't already exist
  command: mv /opt/sonarqube-9.9.8.100196 /opt/sonarqube
```

If the directory was already exists...... → error



```
TASK [sonarqube_install : change sonarqube dir name] ***************************************************************************************
fatal: [web1]: FAILED! => {"changed": true, "cmd": ["mv", "/opt/sonarqube-9.9.8.100196", "/opt/sonarqube"], "delta": "0:00:00.006479", "end": "2025-05-2
8 12:08:11.444819", "msg": "non-zero return code", "rc": 1, "start": "2025-05-28 12:08:11.438340", "stderr": "mv: cannot move '/opt/sonarqube-9.9.8.1001
96' to '/opt/sonarqube/sonarqube-9.9.8.100196': File exists", "stderr_lines": ["mv: cannot move '/opt/sonarqube-9.9.8.100196' to '/opt/sonarqube/sonarqu
be-9.9.8.100196': File exists"], "stdout": "", "stdout_lines": []}
```

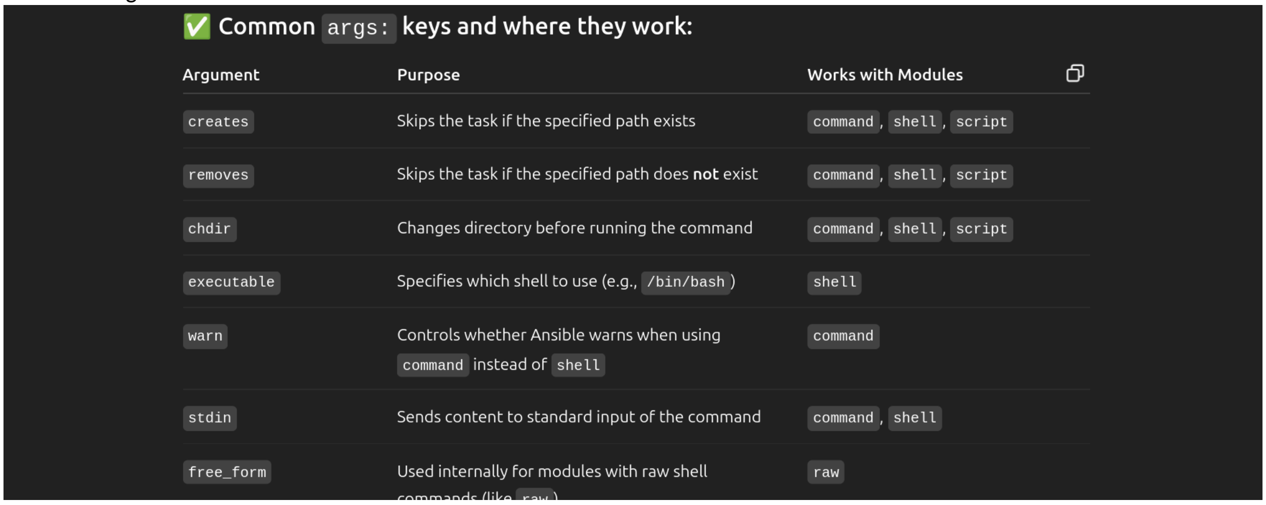
to solve this use args:



```
- name: Rename directory if it doesn't already exist
  command: mv /opt/sonarqube-9.9.8.100196 /opt/sonarqube
  args:
    creates: /opt/sonarqube
```

it just checks if the file/directory already exists →   skill the command

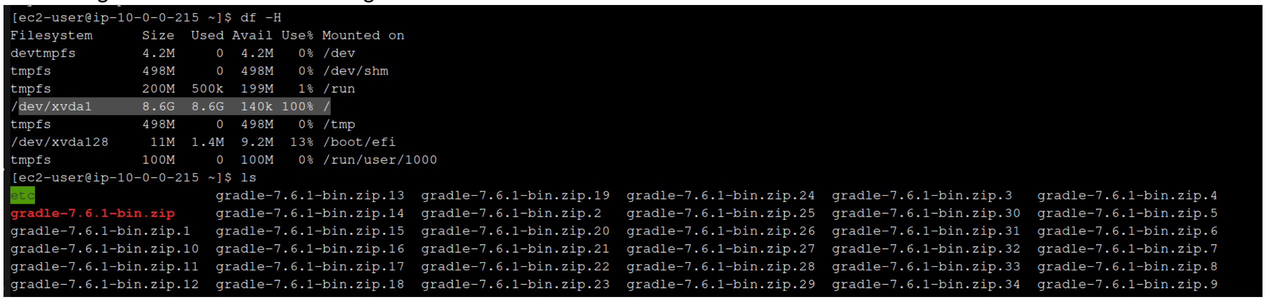
Details on args:



✅ Common `args:` keys and where they work:

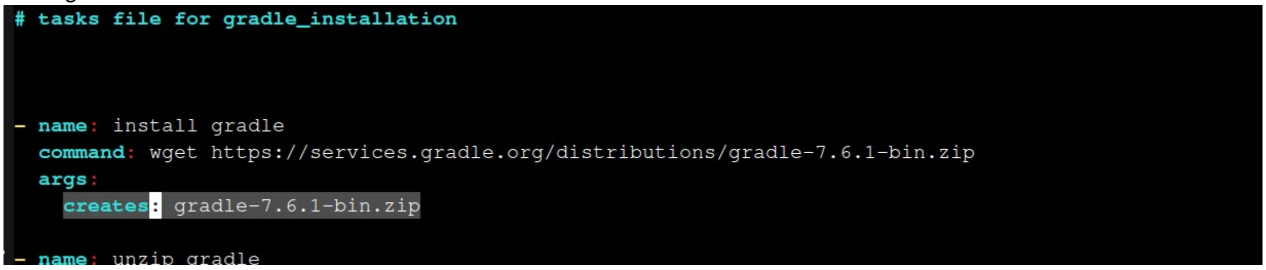
Argument	Purpose	Works with Modules
`creates`	Skips the task if the specified path exists	`command`, `shell`, `script`
`removes`	Skips the task if the specified path does **not** exist	`command`, `shell`, `script`
`chdir`	Changes directory before running the command	`command`, `shell`, `script`
`executable`	Specifies which shell to use (e.g., `/bin/bash`)	`shell`
`warn`	Controls whether Ansible warns when using `command` instead of `shell`	`command`
`stdin`	Sends content to standard input of the command	`command`, `shell`
`free_form`	Used internally for modules with raw shell commands (like `raw`)	`raw`

in one target instances   → the Storage was full because of that



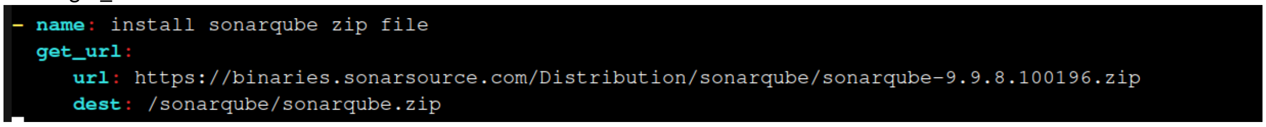
```
[ec2-user@ip-10-0-0-215 ~]$ df -H
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        4.2M     0  4.2M   0% /dev
tmpfs           498M     0  498M   0% /dev/shm
tmpfs           200M  500k  199M   1% /run
/dev/xvda1      8.6G  8.6G  140k 100% /
tmpfs           498M     0  498M   0% /tmp
/dev/xvda128     11M  1.4M  9.2M  13% /boot/efi
tmpfs           100M     0  100M   0% /run/user/1000
[ec2-user@ip-10-0-0-215 ~]$ ls
etc                     gradle-7.6.1-bin.zip.13  gradle-7.6.1-bin.zip.19  gradle-7.6.1-bin.zip.24  gradle-7.6.1-bin.zip.3   gradle-7.6.1-bin.zip.4
gradle-7.6.1-bin.zip    gradle-7.6.1-bin.zip.14  gradle-7.6.1-bin.zip.2   gradle-7.6.1-bin.zip.25  gradle-7.6.1-bin.zip.30  gradle-7.6.1-bin.zip.5
gradle-7.6.1-bin.zip.1  gradle-7.6.1-bin.zip.15  gradle-7.6.1-bin.zip.20  gradle-7.6.1-bin.zip.26  gradle-7.6.1-bin.zip.31  gradle-7.6.1-bin.zip.6
gradle-7.6.1-bin.zip.10 gradle-7.6.1-bin.zip.16  gradle-7.6.1-bin.zip.21  gradle-7.6.1-bin.zip.27  gradle-7.6.1-bin.zip.32  gradle-7.6.1-bin.zip.7
gradle-7.6.1-bin.zip.11 gradle-7.6.1-bin.zip.17  gradle-7.6.1-bin.zip.22  gradle-7.6.1-bin.zip.28  gradle-7.6.1-bin.zip.33  gradle-7.6.1-bin.zip.8
gradle-7.6.1-bin.zip.12 gradle-7.6.1-bin.zip.18  gradle-7.6.1-bin.zip.23  gradle-7.6.1-bin.zip.29  gradle-7.6.1-bin.zip.34  gradle-7.6.1-bin.zip.9
```

add args :



```
# tasks file for gradle_installation


- name: install gradle
  command: wget https://services.gradle.org/distributions/gradle-7.6.1-bin.zip
  args:
    creates: gradle-7.6.1-bin.zip

- name: unzip gradle
```

or use get_url



```
- name: install sonarqube zip file
  get_url:
     url: https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.8.100196.zip
     dest: /sonarqube/sonarqube.zip
```

What is the cgroup and how it’s related to docker?

Control Groups (cgroups) is a Linux kernel feature that **limits, accounts for, and isolates the resource usage** (CPU, memory, disk I/O, network, etc.) of a collection of processes.

Conclude → It provides isolation between processes , it's used by docker to provide a full isolation between containers and each other.

There are two versions of cgroups:

- **cgroup v1:** The original version with separate hierarchies for different resources.
- **cgroup v2:** A newer unified hierarchy combining all resources under one tree.

ISSUE:!!!!
when I try to run any docker command → it struggles and print error

```
cgroups: cgroup mountpoint does not exist: unknown
```

why?
When I try to print cgroup mounts , it prints

```
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,...)
```

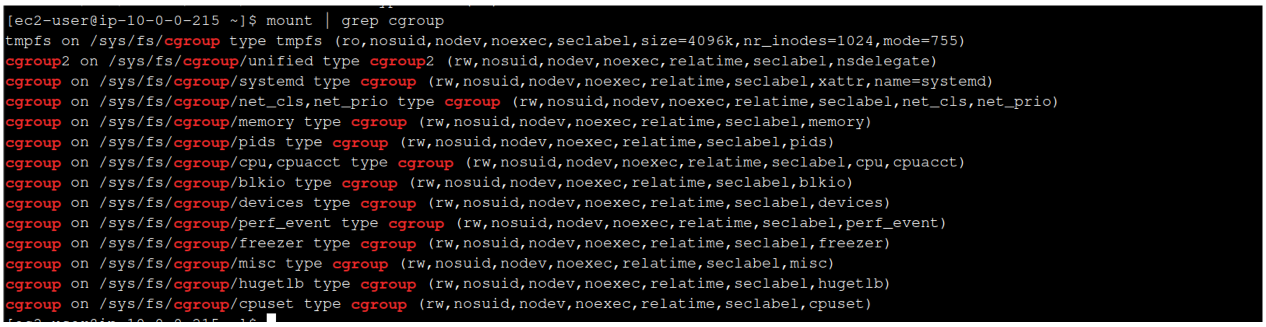
It means that our system has cgroup2 only and our docker version works on cgroup1

- This means your system was running **only cgroup v2** (called "unified cgroup hierarchy").
- But your **Docker version (19.03.15)** expects **cgroup v1** controllers to exist and be mounted.
- Docker could not find the cgroup v1 mountpoints it needed, so it failed to start containers.

Run this command to make your kernel works on cgroup1 and cgroup2

```
sudo grubby --update-kernel=ALL --args="systemd.unified_cgroup_hierarchy=0"
```

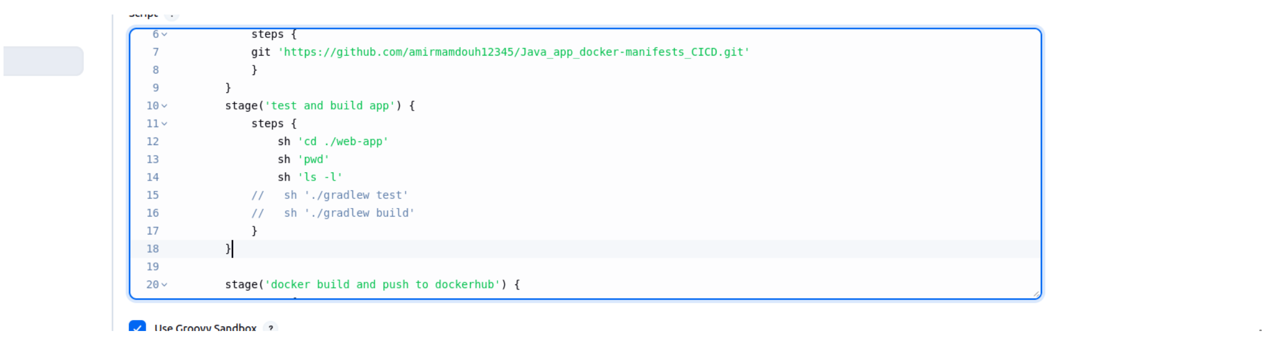
- This **adds a kernel boot parameter** called `systemd.unified_cgroup_hierarchy=0`.
- What this does is tell **systemd** (the init system) to **disable pure cgroup v2 mode** and switch back to **hybrid mode**:
- Hybrid mode means **both cgroup v1 and v2 are active simultaneously**.



```
[ec2-user@ip-10-0-0-215 ~]$ mount | grep cgroup
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,seclabel,size=4096k,nr_inodes=1024,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,seclabel,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,xattr,name=systemd)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,net_cls,net_prio)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,memory)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,pids)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,cpu,cpuacct)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,blkio)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,devices)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,perf_event)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,freezer)
cgroup on /sys/fs/cgroup/misc type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,misc)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,hugetlb)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,cpuset)
```

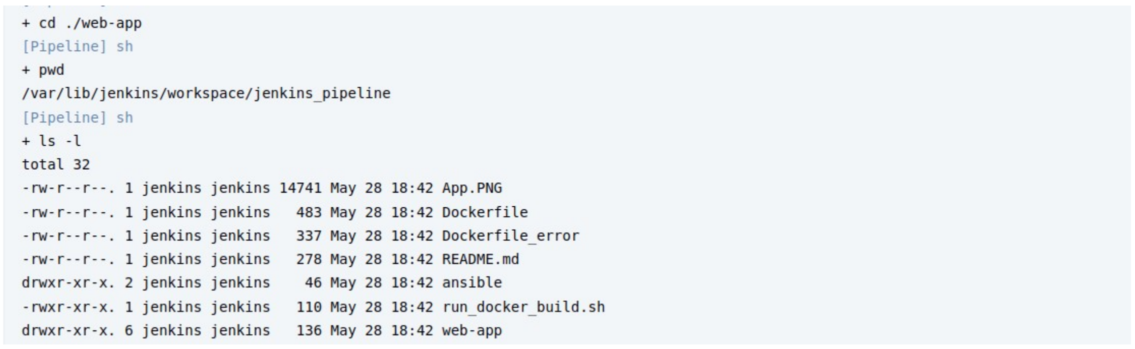
In jenkins

'cd' command can't be run in this way



```
            steps {
            git 'https://github.com/amirmamdouh12345/Java_app_docker-manifests_CICD.git'
            }
        }
        stage('test and build app') {
            steps {
                sh 'cd ./web-app'
                sh 'pwd'
                sh 'ls -l'
            //   sh './gradlew test'
            //   sh './gradlew build'
            }
        }

        stage('docker build and push to dockerhub') {
```

It didn't change directory to web-app



```
+ cd ./web-app
[Pipeline] sh
+ pwd
/var/lib/jenkins/workspace/jenkins_pipeline
[Pipeline] sh
+ ls -l
total 32
-rw-r--r--. 1 jenkins jenkins 14741 May 28 18:42 App.PNG
-rw-r--r--. 1 jenkins jenkins   483 May 28 18:42 Dockerfile
-rw-r--r--. 1 jenkins jenkins   337 May 28 18:42 Dockerfile_error
-rw-r--r--. 1 jenkins jenkins   278 May 28 18:42 README.md
drwxr-xr-x. 2 jenkins jenkins    46 May 28 18:42 ansible
-rwxr-xr-x. 1 jenkins jenkins   110 May 28 18:42 run_docker_build.sh
drwxr-xr-x. 6 jenkins jenkins   136 May 28 18:42 web-app
```

VERY IMPORTANT NOTE:
if you added user to a group and wait to take permissions of this group → you need to re-login first or to re-boot

If you want to resize a mounted partition

to extend the partition
`sudo growpart /dev/xvda 1` → check `lsblk`

to extend the the filesystem
`sudo xfs_growfs /` → check `df -H`

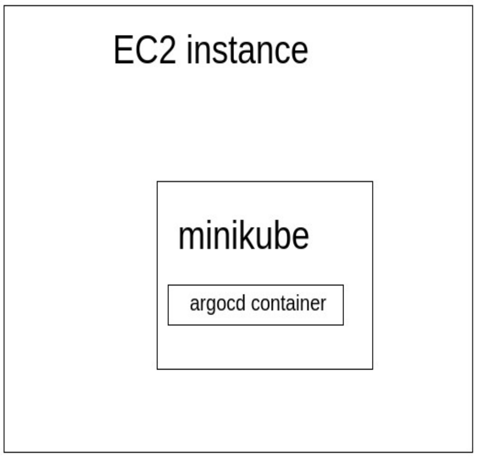
install minikube
```
curl -LO https://github.com/kubernetes/minikube/releases/latest/download/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
```

install kubectl
```
   curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

VERY VERY VERY VERY IMPORTANT!!!

How to make your Argocd accessed from public regions in case of minikube?
You have argocd as a container inside your minikube cluster
minikube is a local pod in your VM or EC2 instance→ (has public IP)

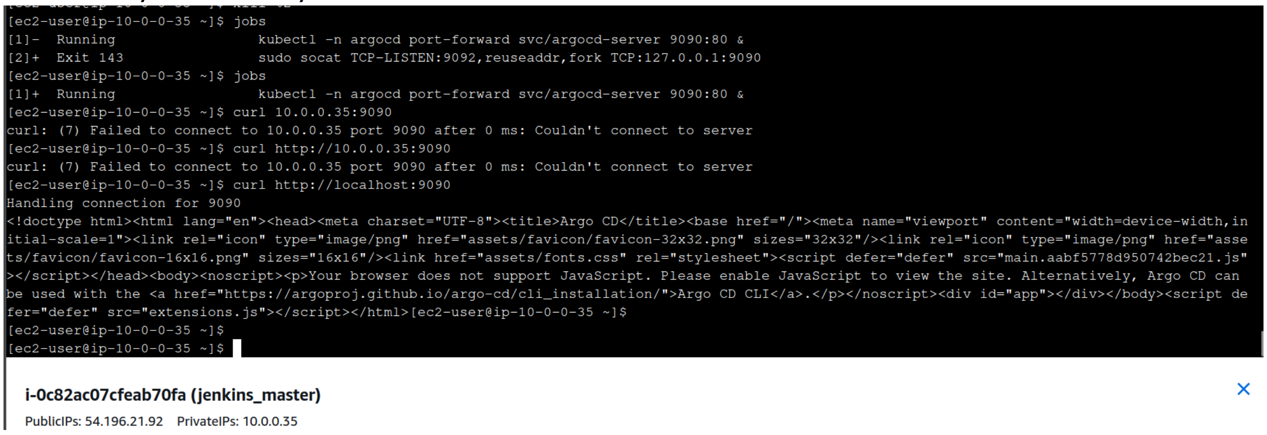
Our mission is to expose argocd pod to outside!!



Wrong Solution in case minikube:
If you made the argocd-server service be nodeport → you will expose the argocd-server pod using minikube ip address not the EC2 instance IP address.
It will work if the control plane was on EC2 instance itself.

Right Solution: → port forwarding
using port forwarding will make the service be accessible from the localhost of the Instance runs the command.
قصدي انك بترن Kubectl port-forward من علي الماشين نفسها , ف هتقدر تاكسيس السيرفيس من اللوكال هوست

I means If you currently on the EC2 instance CMD:12


```
[ec2-user@ip-10-0-0-35 ~]$ jobs
[1]-  Running                 kubectl -n argocd port-forward svc/argocd-server 9090:80 &
[2]+  Exit 143                sudo socat TCP-LISTEN:9092,reuseaddr,fork TCP:127.0.0.1:9090
[ec2-user@ip-10-0-0-35 ~]$ jobs
[1]+  Running                 kubectl -n argocd port-forward svc/argocd-server 9090:80 &
[ec2-user@ip-10-0-0-35 ~]$ curl 10.0.0.35:9090
curl: (7) Failed to connect to 10.0.0.35 port 9090 after 0 ms: Couldn't connect to server
[ec2-user@ip-10-0-0-35 ~]$ curl http://10.0.0.35:9090
curl: (7) Failed to connect to 10.0.0.35 port 9090 after 0 ms: Couldn't connect to server
[ec2-user@ip-10-0-0-35 ~]$ curl http://localhost:9090
Handling connection for 9090
<!doctype html><html lang="en"><head><meta charset="UTF-8"><title>Argo CD</title><base href="/"><meta name="viewport" content="width=device-width,in
itial-scale=1"><link rel="icon" type="image/png" href="assets/favicon/favicon-32x32.png" sizes="32x32"/><link rel="icon" type="image/png" href="asse
ts/favicon/favicon-16x16.png" sizes="16x16"/><link href="assets/fonts.css" rel="stylesheet"><script defer="defer" src="main.aabf5778d950742bec21.js"
></script></head><body><noscript><p>Your browser does not support JavaScript. Please enable JavaScript to view the site. Alternatively, Argo CD can
be used with the <a href="https://argoproj.github.io/argo-cd/cli_installation/">Argo CD CLI</a>.</p></noscript><div id="app"></div></body><script de
fer="defer" src="extensions.js"></script></html>[ec2-user@ip-10-0-0-35 ~]$
[ec2-user@ip-10-0-0-35 ~]$
[ec2-user@ip-10-0-0-35 ~]$
```
i-0c82ac07cfeab70fa (jenkins_master)
PublicIPs: 54.196.21.92 PrivateIPs: 10.0.0.35

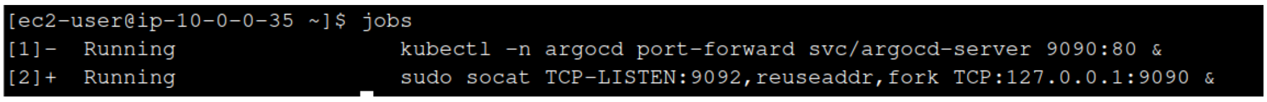
لو رنيته ب الاي بي بتاع الماشين مش هينفع , لكن لو رنيته ب127.0.0.1
اوlocalhost هيرن عادي
mn al a5er bisma7 b al loopback bs

Then Here we need to use something to expose localhost to outside → socat

```
sudo yum install socat -y
```

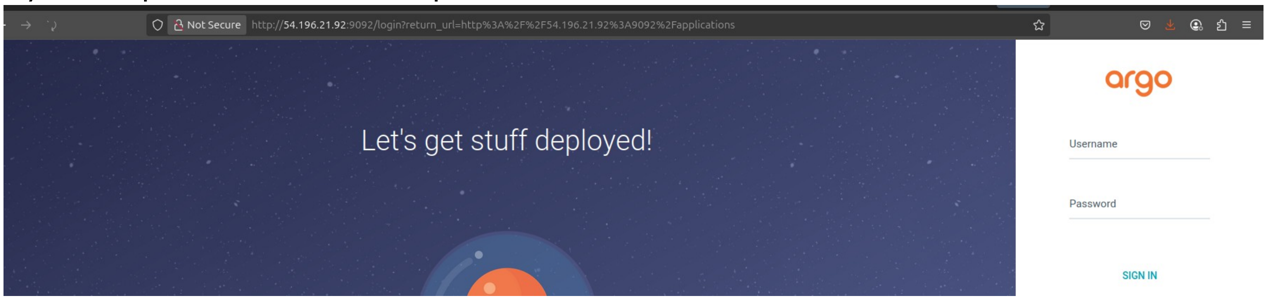
```
sudo socat TCP-LISTEN:9092,reuseaddr, fork TCP:127.0.0.1:9090
```

so any request comes to private or public IP address on port 9092 → forwarded to localhost:9090



```
[ec2-user@ip-10-0-0-35 ~]$ jobs
[1]-  Running                   kubectl -n argocd port-forward svc/argocd-server 9090:80 &
[2]+  Running                   sudo socat TCP-LISTEN:9092,reuseaddr,fork TCP:127.0.0.1:9090 &
```

try to use public IP address with port 9092


Not Secure
http://54.196.21.92:9092/login?return_url=http%3A%2F%2F54.196.21.92%3A9092%2Fapplications
Let's get stuff deployed!
argo
Username
Password
SIGN IN

install argocd

```
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```

nginx-ingress

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.44.0/deploy/static/provider/cloud/deploy.yaml