

\$ Watch date

\$ Watch ls -la

\$ find / -name *.log

PROBLEM : Add a user to the linux system and give only the access to a particular file only ?

Answer :

```
sudo useradd -m chaitanya      # To add a user chaitanya
su - chaitanya                # switch the user
sudo userdel -r chaitanya      # Delete the User and Their Home Directory
sudo passwd chaitanya          # set the password: new@2803NCD for the user
id chaitanya                   # check the uid
sudo groupadd dev              # create a dev group
getent group dev               # check if the group was created
sudo usermod -aG dev chaitanya # Add Chaitanya to the "dev" Group. -aG: Adds the user to the
                                group without removing existing groups.
sudo usermod -g dev chaitanya  # Add Chaitanya to the "dev" Group. -g: Sets the primary group
                                for the user.
sudo chown chaitanya:chaitanya custom.pub #owner of the file is chaitanya and group is chaitanya
id chaitanya                   # Chaitanya is now part of the dev group,
groups chaitanya               # list all the group chaitanya is part of it
sudo chown :dev filelog        # change the group of the user associated with the filelog
sudo groupdel chaitanya        # delete the group created
sudo systemctl status ssh      # check the status of ssh service
cat /etc/group
```

1. chaitanya:x:1001:

Group Name: chaitanya

Group Password: x (default, indicating no password)

Group ID (GID): 1001

Members: No members listed (indicating no users are in this group directly)

2. dev:x:1002:chaitanya

Group Name: dev

Group Password: x

Group ID (GID): 1002

Members: chaitanya is a member of the dev group.

ls -la

-rw-r--r--. 1 chaitanya chaitanya 20 Dec 9 07:33 filelog

- : directory

rw- : owner permissions (Read (r) = 4, Write (w) = 2, Execute (x) = 1)

-r- : group permissions (Read (r) = 4, Write (w) = 2, Execute (x) = 1)

r-- : others permissions (Read (r) = 4, Write (w) = 2, Execute (x) = 1)

chaitanya: The owner of the file.

chaitanya: The group associated with the file.

20: The size of the file in bytes

sudo nano /etc/ssh/sshd_config : set PasswordAuthentication yes

: BE CAREFUL WITH PubkeyAuthentication

: if PubkeyAuthentication is set no, means u cant login with .pem

key

sudo systemctl restart ssh

ssh chaitanya@ip

. chaitanya can only read and write only filelog(which he is owner) can't do anything in the linux System. if chaitanya put sudo vi anyfilename , he can't write any file in linux system because he should be the member of sudoers file also.

Giving chaitanya elevated permissions (superuser)

option 1 : sudo usermod -aG sudo chaitanya

option 2 : sudo visudo

root ALL=(ALL:ALL) ALL

ADD Below:

chaitanya ALL=(ALL:ALL) ALL

ctrl+o: save the file , Enter , ctrl+x: exit from the file.

PUTTY

FROM THE .PEM TO .PPK

1. LOAD THE .PEM (SELECT ALL THE FILES AT THE BOTTOM)
2. SAVE PRIVATE KEY (.PPK) WITH NO PARAPHRASE

MODIFY TEXT COLOR TO GREEN

1. Select Colours and then select "Default Foreground" from the list on the right.
2. .Click the "Modify" button and choose your desired text colour.
3. Click "OK" and then "Apply" to save the change

LOGIN TO PUTTY:

- > Select the session : enter the Hostname and port :22
- > select + SSH > SELECT + AUTH > CREDENTIALS > BROWSE THE PRIVATE KEY(.PPK)FILE HERE
- > ONCE THE TERMINAL IS OPEN , TERMINAL WILL ASK FOR THE USERNAME ENTER

GIT

`git config --global user.name "[name]"`

`git config --global user.email "[email address]"`

`git init [repository name]`

`git clone [url]`

`git add [file]`

`git commit -m "[Type in the commit message]"`

`git commit -a`

`git status`

`git rm [file]`

This command deletes the file from your working directory and stages the deletion.

`git log`

`git branch`

`git branch -d [branch name]`

This command deletes the branch

`git checkout [branch name]`

only switches the branch after clone

`git checkout -b [branch name]`

This command creates a new branch and also switches to it.

`git branch -a`

List all branches (local and remote)

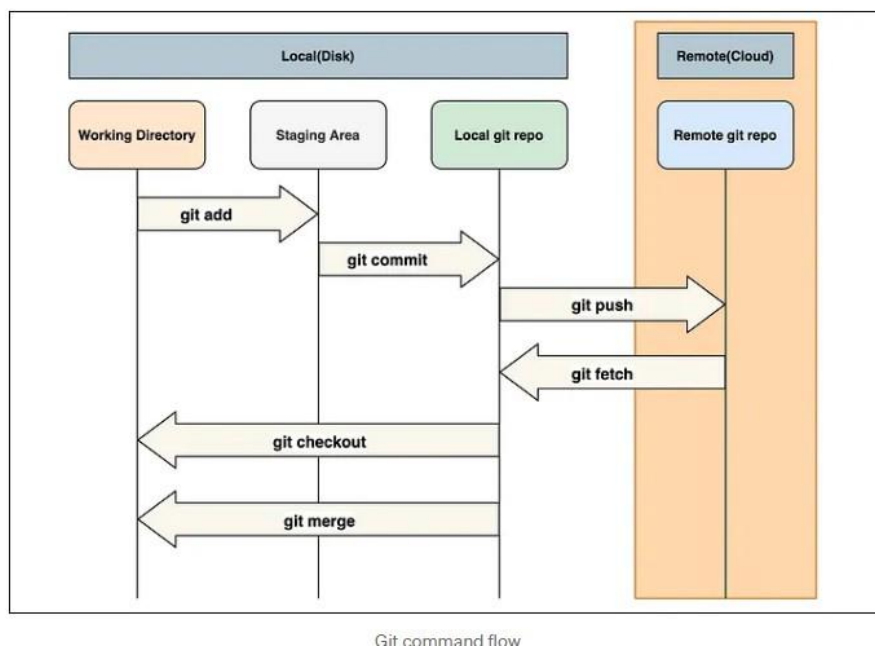
`git clone --mirror <original-repo-url>` #--mirror clones the repository along with all branches, tags, and other references.

`git fetch --all`

`cd <repository-name>`

`git remote add new-remote <new-repo-url>` #Add the new repository as a remote:

`git push new-remote --all` #Push all branches:



`git merge [branch name]`

This command merges the specified branch's history into the current branch.

`git remote add [variable name] [Remote Server Link]`

1. **EXAMPLE:** `git remote add origin https://github.com/chaitan28/Java_app_3.0.git`

git push [variable name] [master]

2.EXAMPLE: git push origin master

git push --all origin

This command pushes all branches to your remote repository.

git pull [Repository Link]

EXAMPLE: git pull https://github.com/chaitan28/Java_app_3.0.git

Git remote add origin <url> or # git remote add new-remote <new-repo-url>

Git init .

Git add .

Git commit -m "message"

git remote -v

git remote remove origin

git rm --cached -r . # Remove All Files from Cache

git remote add new-remote <new-repo-url>

git push - - all <url> - -force ## Username and password asked

Always use git pull before git push

Git push origin master/main/branch

Git pull <url>

DOCKER

INSTALLATION

sudo apt update

sudo apt install docker.io

sudo service docker start

sudo systemctl enable docker

sudo systemctl status docker

docker --version

sudo groupadd docker

sudo usermod -aG docker \$USER # user to the Docker group:

newgrp docker # Log out and log back in to apply the group change

BUILD FROM DOCKERFILE

`docker image build -t <image_name>:<tag> <context_dir>`

#Dockerfile is present current directory presented by dot(.).

`docker build -t centos_buddy .`

`docker build -t <image_name>:<tag or version> .`

`docker build --no-cache -t <my-image> .`

If your Dockerfile is not in your current directory, you can specify it by adding the --file option:

`docker build -f </path/to/dockerfilename> -t <image_name> .`

IMAGES

`docker login`

`docker pull <image name>:tag`

To push the image to dockerhub, you need to change the image name first then the push the image

`docker tag <name of the image>:<tag> <repository_name/imagename>:tag`

`docker push <repository_name/imagename>:tag`

```
root@ip-172-31-43-206:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
busybox       1.37.0    27a71e19c956   10 days ago   4.27MB
tomcat        jre17     d82e93920526   2 weeks ago   276MB
root@ip-172-31-43-206:~# docker tag busybox:1.37.0 chaitan28/poc_docker:busybox_image
root@ip-172-31-43-206:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
chaitan28/poc_docker  busybox_image  27a71e19c956   10 days ago   4.27MB
busybox       1.37.0    27a71e19c956   10 days ago   4.27MB
tomcat        jre17     d82e93920526   2 weeks ago   276MB
root@ip-172-31-43-206:~# docker push chaitan28/poc_docker:busybox_image
```

`docker run -d --name <container_name> <image_name>`

`docker run -d --name <container_name> -p <host_post>:<container_port> <image_name>`

when u don't put -d after run ,docker runs all the logs in foreground(on the live terminal). It will difficult for you to excute commuand again on the live terminal .To keep the live terminal free we use -d (background).

If you run a container without -d but want to detach (return to the terminal) without stopping it, you can use the following key combination

Ctrl + P, then Ctrl + Q

This will detach from the container without stopping it. Stop with Ctrl + C

#remove images

docker system prune -a #remove all unused images and containers

docker rmi <image name or image id>

docker image prune

docker image prune -a

docker images

#list of unused images

docker images -a

Containers

#To list all container

docker ps -a

docker ps -aq

#docker container which are running

docker ps

docker start <container_id or container_name>

docker stop <container_id or container_name>

docker restart <container_id or container_name>

docker pause <container_id or container_name>

docker unpause <container_name or container_id>

docker rm -f <container_name or container_id>

remove all the used and unused containers

docker rm -f \$(docker ps -aq)

login to container

login to the container

docker exec -it <container_id or container_name> /bin/bash

docker exec -it <container_id or container_name> /bin/sh

docker exec -u root -it <container_id_or_name> /bin/sh

Ship Docker Image Between Hosts

#Save the Docker Image on the Source Host

On the source host, use the docker save command to export the image to a .tar file:

```
docker save -o my-image.tar my-image:latest
```

#my-image:latest: Replace this with the name and tag of your image.

#my-image.tar: The name of the file to which the image will be saved.

#You can transfer the .tar file to the target host using methods like SCP command

To check the Linux distribution running inside a Docker container, you can follow these methods:

Check /etc/os-release File

The /etc/os-release file contains information about the Linux distribution. You can read it directly to find the distribution name and version.

Command:

After login inside container

```
$ cat /etc/os-release
```

Example Output (Ubuntu):

```
NAME="Ubuntu"
```

```
VERSION="20.04 LTS (Focal Fossa)"
```

```
ID=ubuntu
```

```
ID_LIKE=debian
```

```
VERSION_ID="20.04"
```

To check the CPU and memory usage for all running Docker containers, you can use the following command:

```
$docker stats
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
fbbd9c31fdb4	my_app	0.06%	50.9MiB / 2GiB	2.49%	1.2kB / 1.5kB	0B / 0B	5
4d4e6e41f383	nginx	0.01%	27.4MiB / 512MiB	5.35%	3.4kB / 1.4kB	0B / 0B	2

Check Disk Usage of a Specific Container

To see the disk usage for a specific container, you can use the docker system df command, which provides details about the disk space used by images, containers, and volumes.

Command:

```
$ docker system df
```

example:

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	3	1	2.5GB	1.2GB (48%)

Containers 2 2 500MB 0B (0%)

Steps to Change Docker's Storage Location

First, stop the Docker service so that the configuration file can be safely modified.

```
$sudo systemctl stop docker
```

Create a New Directory: Create the new directory where you want Docker to store its data. For example, you might want to mount a new disk or partition here.

```
$sudo mkdir -p /new/disk/location
```

```
$sudo chown root:docker /new/disk/location
```

Ensure that Docker has proper permissions to write to this new location.

Edit the Docker Daemon Configuration (daemon.json): Edit Docker's configuration file (/etc/docker/daemon.json) to specify the new storage location. If the file doesn't exist, you can create it.

```
sudo nano /etc/docker/daemon.json
```

Add the following JSON configuration:

```
{  
  "data-root": "/new/disk/location"  
}
```

After saving the configuration, restart the Docker service for the changes to take effect.

```
$sudo systemctl start docker
```

Verify the Change: To ensure Docker is now using the new location for its data, run:

```
$ docker info | grep "Docker Root Dir"
```

You should see the updated directory path under the "Docker Root Dir" section.

No, you cannot modify the memory allocation for a running Docker container using the docker run command. The --memory flag (and other resource limits) can only be set when creating a new container. Once the container is running, its resource limits are fixed.

```
$ docker stop my_container
```

```
$ docker rm my_container
```

```
$ docker rm my_container
```

After removing the old container, you can now create a new one with the desired memory allocation.

```
$ docker run -d --memory=4g --name my_container my_image
```

This will start the container with a memory limit of 4GB.

To Avoid Downtime:

If you are looking to avoid downtime while increasing resources, consider using Docker Compose for container orchestration, or tools like **Docker Swarm** or **Kubernetes**, which allow you to manage containers more dynamically with resource scaling.

PORT MAPPING

```
docker run -d --name <container_name> -p <host_post>:<container_port> <image_name>
```

```
docker logs -f <container_id or container_name>
```

REMOVE CACHE

```
docker builder prune
```

```
docker build --no-cache -t your_image_name .
```

DOCKER VOLUME

```
docker run -d --name <container-name> -v <path on host>:<path-in-container-where-volume-is-mounted> <image-name>
```

```
$ docker run -v /host/path:/container/path my_image
```

This will mount the /host/path directory on your host machine to the /container/path directory inside the Docker container.

Key points about this setup:

Bidirectional syncing: Any changes made to the files in /host/path on your host system will immediately reflect inside the container at /container/path, and vice versa.

Live updates: The data is shared between the host and the container, so if you modify or add files in /host/path, those changes will be visible inside the container at /container/path, and if you modify files inside the container at /container/path, they will also be reflected on the host.

```
$ docker volume inspect <volume-name>
```

```
$ docker volume rm <volume-name>
```

NETWORK:

```
docker inspect <container_name or container_id >
```

```
docker network ls
```

```
docker network inspect <network_id or network_name >
```

Firewall

```
sudo ufw status  
sudo ufw enable  
sudo ufw app list  
sudo ufw allow 'Nginx HTTP'  
sudo ufw status  
sudo ufw allow OpenSSH  
sudo ufw allow <pdockerort>
```

JENKINS:

```
.jar files stored directory  
/var/lib/jenkins/workspace
```

Maven

mvn clean

This command cleans the Maven project by deleting the target directory

mvn package

This command builds the Maven project and packages it into a JAR, WAR.

mvn install

This command builds the Maven project and installs the project files (JAR, WAR, pom.xml, etc.) to the local repository

mvn deploy

This command deploys the artifact to the remote repository

mvn validate

This command validates the Maven project to ensure syntax is correct

mvn clean install -DskipTests

This command clean and install the war/jar in the current directory

JFROG

```
curl -X PUT [URL] -u [username:password] -T [file-to-upload]
```

Example: `curl -X PUT -u "admin:password" -T /path/to/myfile.war http://artifactory-url/artifactory/libs-release-local/com/mycompany/myfile.war`

JFROG:

JFROG : <https://jfrog.com/help/r/jfrog-installation-setup-documentation/install-artifactory-single-node-with-linux-archive>.

#You can use wget to download the .deb package directly

Wget https://downloads.mysql.com/archives/get/p/3/file/mysql-connector-j_8.4.0-1ubuntu24.04_all.deb

Once the download completes, install the package with dpkg

```
sudo dpkg -i mysql-connector-j_8.4.0-1ubuntu24.04_all.deb
```

If there are dependency issues, you can resolve them by running:

```
sudo apt-get install -f
```

To verify the location of the MySQL Connector .jar file, you can search for it using find

```
find /usr/share/java -name "mysql-connector-java*.jar"
```

copy the mysql-connector-java-<version> .jar file into

\$JFROG_HOME/artifactory/var/bootstrap/artifactory/tomcat/lib directory.

```
sudo ln -s /usr/share/java/mysql-connector-java-8.4.0.jar
```

```
/home/ubuntu/tools_installation_scripts/jfrog/artifactory/var/bootstrap/artifactory/tomcat/lib
```

Ansible installations:

https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html

```
sudo apt update
```

```
sudo apt install software-properties-common
```

```
sudo add-apt-repository --yes --update ppa:ansible/ansible
```

```
sudo apt install ansible
```

```
python3 --version
```

```
ansible --version
```

<https://devopscube.com/setup-ansible-aws-dynamic-inventory/>

LINUX MOUNT DISK

```
slave@slave2:~$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda          8:0    0   30G  0 disk
├─sda1       8:1    0   29G  0 part /
├─sda14      8:14   0    4M  0 part
├─sda15      8:15   0  106M  0 part /boot/efi
└─sda16     259:0   0   913M  0 part /boot
sdb          8:16   0   16G  0 disk
└─sdb1       8:17   0   16G  0 part /mnt
sr0         11:0    1  628K  0 rom
```

Step 1: Verify the New Disk

Use the `lsblk` command to check if the new disk (`/dev/sdc`) is available and confirm its unmounted status:

```
lsblk
```

You should see `/dev/sdc` listed, but it may not yet have a partition (e.g., `/dev/sdc1`).

Step 2: Partition the Disk

If you want to create a partition on `/dev/sdc` (recommended for new disks):

1. Start the `fdisk` utility:

```
sudo fdisk /dev/sdc
```

2. Inside `fdisk`:
 - Type `n` to create a new partition.
 - Press `p` to make it a primary partition.
 - Press `1` to select partition number 1.
 - Accept the defaults for the first and last sectors.
 - Type `w` to write changes and exit.
3. Verify the partition is created:

```
lsblk
```

You should now see something like `/dev/sdc1`.

Step 3: Format the New Disk

1. Format the partition with a filesystem (e.g., `ext4`):

```
sudo mkfs.ext4 /dev/sdc1
```

2. Confirm formatting is successful by running:

```
sudo blkid /dev/sdc1
```

This will display information about the filesystem and its UUID.

Step 4: Create a Mount Point

1. Create a directory where you want to mount the new disk. For example:

```
sudo mkdir /mnt/newdisk
```

Step 5: Mount the Disk

1. Mount the disk to the directory you just created:

```
sudo mount /dev/sdc1 /mnt/newdisk
```

2. Verify that the disk is mounted successfully:

```
df -h
```

You should see `/dev/sdc1` mounted on `/mnt/newdisk`.

Changing the default (root) disk in Linux, particularly on a virtual machine (such as in Azure), typically involves more complex steps since the root filesystem is where the operating system runs.

This message is normal when you're creating a new partition table on a blank disk. It's simply warning you that the changes you've made so far (like creating a new partition) are only in memory and haven't yet been written to disk.

Here's how to proceed carefully:

1. **Proceed with Partitioning:** Since the disk is empty, you'll likely want to create a new partition:
 - Type `n` and press **Enter** to create a new partition.
 - Press **Enter** to accept the default values for partition type, partition number, and start and end sectors unless you have specific requirements.
2. **Write the Partition Table:**
 - After defining the partition, type `w` and press **Enter** to write the changes to disk.
 - This will save the new partition table to `/dev/sdc`

```

slave@slave2:~$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda          8:0    0   30G  0 disk
├─sda1       8:1    0   29G  0 part /
├─sda14      8:14   0    4M  0 part
├─sda15      8:15   0  106M  0 part /boot/efi
└─sda16     259:0   0   913M  0 part /boot
sdb          8:16   0   16G  0 disk
└─sdb1       8:17   0   16G  0 part /mnt
sdc          8:32   0   32G  0 disk
└─sdc1       8:33   0   32G  0 part
sr0         11:0    1   628K  0 rom
slave@slave2:~$ sudo mkfs.ext4 /dev/sdc1

```

OpenSSH is Installed

To verify if OpenSSH is installed, run:

```
dpkg -l | grep openssh
```

For **Debian/Ubuntu**-based systems, this will list the installed OpenSSH packages. If OpenSSH is not installed, you can install it using:

```

sudo apt update
sudo apt install openssh-server

```

For **Red Hat/CentOS**-based systems, use:

```
sudo yum install openssh-server
```

2. Enable and Start the SSH Service

After installing OpenSSH, start and enable the SSH service to start at boot:

```

sudo systemctl enable ssh
sudo systemctl start ssh

```

If you're using a systemd-based system and `sshd.service` is not found, the service might be named `ssh` (as used by some distributions). You can check the status of SSH with:

```
sudo systemctl status ssh
```

ANSIBLE

- `ssh-keygen -t rsa -b 2048`

1. SSH Directory and File Permissions:

Run the following commands on the slave:

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

- `~/.ssh` should have **700** permissions. 700 (drwx-----)
- `~/.ssh/authorized_keys` should have **600** permissions. 600 (-rw-----)

2. Check the Ownership:

Make sure the `.ssh` directory and `authorized_keys` file are owned by the correct user:

```
chown -R user:user ~/.ssh
```

Replace `user` with the actual username on the slave machine.

2. Verify SSH Configuration on the Slave

Check the SSH configuration file (`/etc/ssh/sshd_config`) on the slave machine to ensure it allows public key authentication.

1. Open the SSH configuration file:

```
sudo nano /etc/ssh/sshd_config
```

2. Look for the following lines and ensure they are set as follows:

```
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

3. Save any changes and restart the SSH service:

```
sudo systemctl restart sshd
```

3. Use `ssh-copy-id` with a Different Method

If `ssh-copy-id` isn't working as expected, you can manually copy the public key.

1. On the Master: Display the public key content.

```
cat ~/.ssh/id_rsa.pub
```

2. On the Slave: Append the public key to the `authorized_keys` file.


```
echo "ssh-rsa AAAA... your_public_key_here" >> ~/.ssh/authorized_keys
```

Replace "ssh-rsa AAAA... your_public_key_here" with the public key you copied.

If `ssh-copy-id` fails, ensure that:

- The remote machine is reachable and SSH is running.
- You have correct permissions on the `~/.ssh/authorized_keys` file.
- The remote machine's SSH configuration allows public key authentication (`PubkeyAuthentication yes`).

Static Inventory: Static inventory is a file which contains the list of IP addresses of target host machines and groups them together based on user requirements in json or ini format. Default location of **inventory** is **/etc/ansible/hosts**

Dynamic Inventory : Ansible team will provide two files, python script and .ini file. When we execute the script it will automatically fetch the address of target host machines and store it in the inventory file. Only thing we have to update is the **path of the inventory file in the python script**.

ANSIBLE ROLES

1. Install the Role (if not already installed)

If you are using a role from Ansible Galaxy, first install the role:

```
ansible-galaxy install <role_name>
```

For example:

```
ansible-galaxy install geerlingguy.apache
```

2. Create a Playbook with the Role

Next, create an Ansible playbook that references the role. Here's an example playbook that uses the `geerlingguy.apache` role to install and configure Apache:

```
Playbook.yml
---
- name: Install and configure Apache
  hosts: webserver
  become: yes
  roles:
    - geerlingguy.apache
```

- `hosts: webserver` specifies which group of hosts in your inventory the role should be applied to.
- `become: yes` means the role will run with elevated privileges (sudo).
- `roles:` specifies the role you want to run.

3. Run the Playbook

Once your playbook is ready, run it using the `ansible-playbook` command:

```
ansible-playbook -i inventory_file playbook.yml
```

- `-i inventory_file` specifies your inventory file (e.g., `hosts` file, or dynamic inventory).
- `playbook.yml` is the playbook you created with the role.

YUM FIREWALLD SAME AS UFW IN APT

```
sudo yum install firewalld -y
```

```
sudo systemctl enable firewalld
```

```
sudo systemctl start firewalld
```

```
sudo firewall-cmd --state
```

```
sudo systemctl status firewalld
```

```
sudo firewall-cmd --permanent --add-port=8080/tcp
```

```
sudo firewall-cmd --reload
```

```
sudo firewall-cmd --list-ports
```

```
sudo firewall-cmd --permanent --add-service=http
```

```
sudo firewall-cmd --reload
```

```
sudo firewall-cmd --list-ports
```

```
sudo firewall-cmd --permanent --add-service=http
```

```
sudo firewall-cmd --reload
```

NGINX TO PRINT IP ON URL:

```
#!/bin/bash
```

```
sudo yum update -y
```

```
sudo amazon-linux-extras enable nginx1
```

```
sudo yum install nginx -y
```

```
sudo systemctl start nginx
```

```
sudo systemctl enable nginx
```

```
sudo yum update -y
```

```

sudo yum install firewalld -y
sudo systemctl enable firewalld
sudo systemctl start firewalld
sudo firewall-cmd --permanent --add-port=80/tcp
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --reload
sudo systemctl status nginx
sudo systemctl status firewalld
sudo firewall-cmd --list-ports

nginx -v

sudo chmod 644 /usr/share/nginx/html/index.html

export PUBLIC_IP=$(curl -s http://checkip.amazonaws.com)

echo "From a EC2 With Public IP: $PUBLIC_IP" | sudo tee /usr/share/nginx/html/index.html
> /dev/null

sudo systemctl restart nginx

```

NGINX CONTAINER

```

$ sudo usermod -aG docker $USER ( run without sudo login in and out )

$ docker pull nginx:stable-perl

$ docker cp <container_id>:/usr/share/nginx/html /docker_vol/nginx/data/

$ docker cp <container_id>: /etc/nginx/conf.d /etc/nginx/conf.d

$ docker run -d -p 9000:80 --name nginx -v /docker_vol/nginx/data:/usr/share/nginx/html -v
/etc/nginx/conf.d:/etc/nginx/conf.d nginx:stable-perl (before that create a conf.d and
index.html otherwise empty directory will be copied)

$ docker stop <container_id>

# update the nginx.conf

server {

    listen 80;

    server_name hoog.com; # Your domain

    location / {

```

```

    proxy_pass http://localhost:9000; # Forward requests to port 9000
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}

# Optional: Custom error page
error_page 500 502 503 504 /50x.html;

location = /50x.html {
    root /usr/share/nginx/html;
}
}

```

S3 Policy for a bucket:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "deny all s3 access except owner",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::dev-app1-v1.23-static/*",

```

```

        "arn:aws:s3:::dev-app1-v1.23-static"
    ],
    "Condition": {
        "StringNotEquals": {
            "aws:PrincipalArn": "arn:aws:iam::216989137624:root"
        }
    }
}
]
}

```

Key Components:

1. Effect: "Deny"

This explicitly denies any S3 operation (s3:*) on the bucket or its objects.

2. Principal: "*"

The * means the policy applies to all users, roles, and services (global denial).

3. Resource:

Specifies the bucket (arn:aws:s3:::dev-app1-v1.23-static) and all objects inside it (arn:aws:s3:::dev-app1-v1.23-static/*).

4. Condition:

This ensures the denial is only effective if the requester's ARN (aws:PrincipalArn) is not equal to the provided root account ARN (arn:aws:iam::2169891374:root).

S3FS MOUNT

- Add useradd username
- Set the password to username : passwd username
- sudo su – username
- grant username with sudo permission from sudo visudo ,ADD username ALL=(ALL:ALL) ALL
- su - username
- sudo whoami

- `sudo apt update && sudo apt upgrade -y`
- `sudo apt install curl unzip`
- `curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
- `unzip awscliv2.zip`
- `aws --version`
- go the username IAM console -> users -> security credentials -> create access keys
- `aws configure`
- `aws s3 ls`
- `sudo apt install -y s3fs`
- `mkdir ~/s3bucket`
- `s3fs my-bucket-name ~/s3bucket`
- `df -h | grep s3`
- **try to copy the file into s3bucket , then you can see all the files into console**

AWS EFS MOUNT

What is /etc/fstab?

The /etc/fstab file lists the file systems that should be mounted automatically when the system boots/restarts, along with their mount points and mount options.

The format of each line in /etc/fstab generally consists of 6 fields:

1. **Device:** The device or file system to be mounted (e.g., /dev/sda1 or an NFS share).Mount
2. **Point:** The directory where the device will be mounted (e.g., /mnt/data).
3. **File System Type:** The type of the file system (e.g., ext4, nfs, xfs).
4. **Mount Options:** Any options to be used during mounting (e.g., defaults, rw, noatime).
5. **Dump:** Used by the dump utility; typically set to 0 for NFS.
6. **Pass:** Used by fsck to determine the order of file system checks during boot (typically 0 for NFS).

```
$sudo apt update
```

```
$sudo apt install nfs-common -y
```

```
$sudo systemctl enable nfs-common.service
```

IGNORE: Failed to start nfs-common.service: Unit nfs-common.service is masked.

```
$sudo systemctl start nfs-common.service
```

INGORE: Failed to start nfs-common.service: Unit nfs-common.service is masked.

only mount through ip =

```
sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsz=1048576,hard,timeo=600,retrans=2,noresvport 172.31.83.46:/mnt/efs
```

change the SG in aws efs in network

```
$ sudo vi /etc/fstab
```

```
ADD 172.31.83.46:/ /mnt/efs nfs4
```

```
$ sudo systemctl daemon-reload
```

```
$ sudo mount -a
```

```
$ sudo umount /mnt/efs
$ cd /mnt/efs/
$ watch ls -la
```

System logs

System installations logs

\$ tail -f /var/log/syslog

You won't see Jenkins job logs in `syslog` because Jenkins maintains its own logging system and does not directly integrate with the system logger like `syslog` by default

if u want see specific job logs in Jenkins:

```
$ cat /var/lib/jenkins/jobs/sample-freestyle/builds/2
```

TOMCAT

- The configuration you posted is from a `Tomcat server.xml` file. This file defines how the Tomcat server should handle incoming requests, set up connectors (for HTTP, AJP, and SSL), configure listeners, and define other behaviors like access logs and security realms.
-
- Based on the community request, Webapps folder is moved to the webapps.dist folder, which means webapps folder is empty and there are no files to serve on the browser. That is when you saw the error message The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.
- To re-enable the webapps, we need to move the files from webapps.dist to webapps. Here are the steps to run to make sure that the tomcat server is up and running without any issues. Please run the below commands step by step:
- `cd /usr/local/tomcat/`
- `mv webapps webapps2`
- `mv webapps.dist/ webapps`
- `ls webapps`

```
cd webapps
ls
docs  examples  host-manager  manager  ROOT
pwd
```

- `exit`
- `cd /usr/local/tomcat/bin`

- `./catalina.sh start`: Starts Tomcat.
- `./catalina.sh stop`: Stops Tomcat.
- `./catalina.sh restart`: Restarts Tomcat

```
$ docker run -d --name tomcat-container -p 9000:8080 -v /var/log/tomcat-docker:/usr/local/tomcat/logs tomcat:9.0.97-jdk17
```

```
$ cp /usr/local/tomcat/logs/SampleWebApp.war  
/usr/local/tomcat/webapps
```


CLOUDWATCH AGENT

- Creating a role with following access

Step 2: Add permissions

[Edit](#)

Permissions policy summary

Policy name 	Type	Attached as
AmazonEC2RoleforSSM	AWS managed	Permissions policy
AmazonSSMFullAccess	AWS managed	Permissions policy
AmazonSSManagedInstanceCore	AWS managed	Permissions policy
AWSDataLifecycleManagerSSMFullAccess	AWS managed	Permissions policy
CloudWatchAgentAdminPolicy	AWS managed	Permissions policy
CloudWatchAgentServerPolicy	AWS managed	Permissions policy
CloudWatchLogsFullAccess	AWS managed	Permissions policy

- Attach the role to the running instance.
- Install cloudwatch awslogs.service from below:
\$sudo apt-get update
\$sudo apt-get install -y python-is-python3 or sudo apt-get install -y python
\$python --version
- Your copying the cloudwatch file into ec2 instance
systems manager > Node Management > run command >
AWS-ConfigureAWSPackage > Choose instances manually > select the ec2
instance
The above will be copied configured in /opt/aws/amazon-cloudwatch-agent/bin

AWS-ConfigureAWSPackage is an AWS Systems Manager (SSM) document (SSM Document) that allows you to install, update, or remove software packages on instances managed by AWS Systems Manager. It is often used to manage software agents like the CloudWatch Agent.

- After successful run , we need to install the agent

ASG TEMPLATE RULES

ASG Settings:

Example:

Minimum Capacity: 1

Desired Capacity: 2

Maximum Capacity: 5

Initial State:

The ASG launches 2 instances to match the desired capacity.

Scaling Down:

If traffic decreases, the ASG reduces the desired capacity but will not drop below 1 instance (minimum capacity).

Scaling Up:

If traffic spikes, the ASG scales up to meet the demand, increasing the desired capacity up to a maximum of 5 instances.

`$ scp -i .\aws-demo.pem .\aws-demo.pem ec2-user@ec2-3-83-84-91.compute-1.amazonaws.com:/tmp`

Create SGs

Create target group

Create ALB

Create ASG Template

Create ASG

ROUTE53:

A RECORD: maps ip to route53 DNS record

CNAME RECORD: associate ssl/tls certificates

NS RECORDS: associate the custom domain name

MX RECORDS: associates mailing services