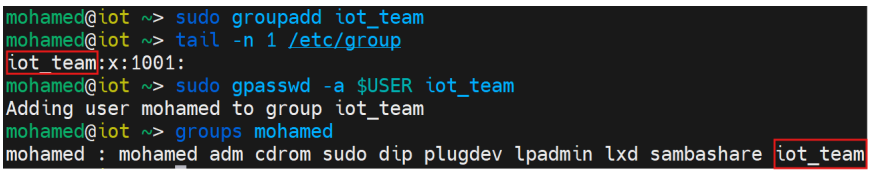
Day 3 – Phase 3: User, Group, and Permissions Management

Boss's Request: Secure the project and restrict access to authorized users only.

Tasks:

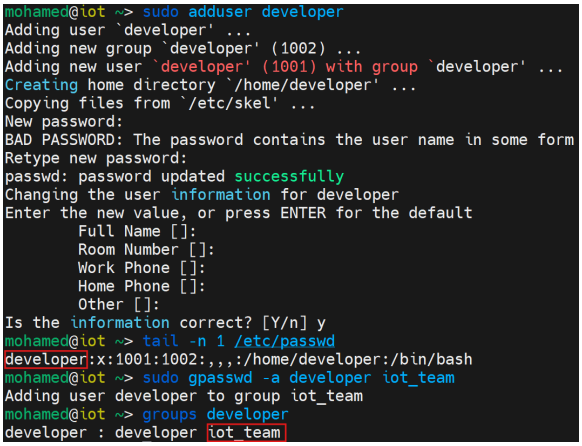
- Create a new group `iot_team` and add your user to it.

```
sudo groupadd iot_team
sudo gpasswd -a $USER iot_team
```



- Create a new developer user, add it to the group.

```
sudo adduser developer
sudo gpasswd -a developer iot_team
```

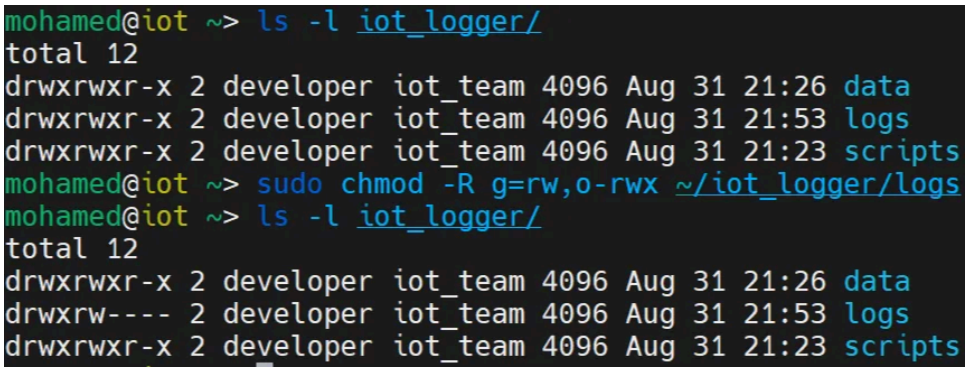


- Change ownership of `iot_logger` to the developer + group.

```
sudo chown -R developer:iot_team ~/iot_logger
```

- Set permissions: group can read/write logs, others blocked.

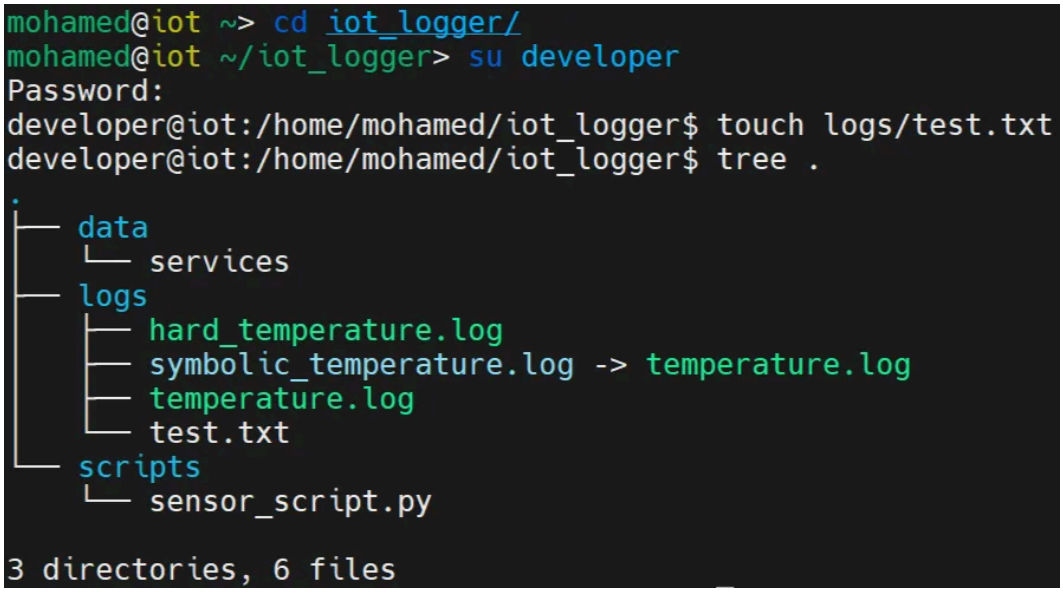
```
sudo chmod -R g=rw,o-rwx ~/iot_logger/logs
# or
sudo chmod -R 760 ~/iot_logger/logs    # that may change owner permissi
ons too
```



- Test access as new user, then remove test user.
  - **developer** is owner of `iot_logger` , so he can access `iot_logger` not `/home/mohamed/iot_logger`

    ```
    cd iot_logger/
    # change to developer user at same path
    su developer

    # to test access
    touch logs/test.txt
    ```

  - remove test user

    ```
    sudo userdel -r developer
    ```

Open-Ended Questions:

- How do Linux file permissions (`r`, `w`, `x`) work for files vs directories? Give an example using `ls -l`.

Permission	Files	Directories
`r`	read file contents	list directory contents
`w`	modify file contents	create/delete files inside
`x`	execute file	access (cd into) the directory

  - Example:

```
drwxrwxr-x 5 mohamed mohamed 4096 Aug 31 21:50 iot_logger
```

    - Owner & group can do everything, others can read+enter

- Explain octal notation for permissions and what the `umask` command does. Give one calculation example.
  - Octal notation → r=4, w=2, x=1
  - Example:
    - `chmod 775 iot_logger`

```
drwxrwxr-x 5 mohamed mohamed 4096 Aug 31 21:50 iot_logger
```

    - Owner = `111` = 7, Group = `111` = 7, Other = `101` = 5
  - `umask` sets default permissions when new files are created.
    - Example: if `umask 022` , If default file permissions = `666`, actual = `644`.
- What is the difference between the root user and a normal user? Why is root considered dangerous?
  - **Root user** = superuser, full control (can change system files, delete anything).
  - **Normal user** = limited, safer.
  - Root is dangerous because mistakes (like `rm -rf /`) can break the system.