Connessione alla propria macchina

1. Aprire il client SSH .
2. Posizionarsi nella directory di salvataggio del file .pem allegato alla mail.
3. Tramite il client SSH, utilizza il seguente comando al fine di impostare le autorizzazioni del file della chiave privata in **read only** per l'utente owner.
   chmod 400 keyPairName.pem
4. Nella finestra del terminale, utilizzare il comando ssh per connettersi all'istanza. Specificare il percorso e il nome del file della chiave privata (.pem), il nome utente per l'istanza e il nome DNS pubblico per l'istanza.
5. Utilizzare la username **centos** per la connessione al sistema

Esempio:
ssh -i "**keyPairName.pem**"
centos@ec2-xx-xxx-xx-x.eu-central-1.compute.amazonaws.com

Per passare all'utente **root** utilizzare il comando "**sudo su -**"
[centos@ip-xxx-xx-xx-xxx ~]$ **sudo su -**

Exercise 1: Managing Files with Shell Expansion and Command substitution

- Utilizzare l'account **student** (se non presente crearlo)
- Creare sotto il path **/exam/exercise1/** le directory **exercise_directoryX** con **X** compreso tra 1 e 100
- all'interno di ogni directory sotto **/exam/exercise1/exercise_directoryX** creare i files
    - **file(a..z)_HOSTNAME_DATE.txt**
        - con **(a..z)=** tutte le lettere dalla a alla z
        - con **HOSTNAME =** hostname del sistema (**hostname -s**)
        - **DATE** la data di creazione file nel formato H:M:S

Il risultato sarà il seguente:

```
|-- exercise1_directory1
|   |-- filea_desktop_00:36:47
|   |-- fileb_desktop_00:36:47
|   `-- filec_desktop_00:36:47
etc...
|-- exercise1_directory2
|   |-- filea_desktop_00:36:47
|   |-- fileb_desktop_00:36:47
|   `-- filec_desktop_00:36:47
etc...
```

Exercise 2: Managing pipeline and regular expression

- ricercare sotto la directory **/usr** tutti i nomi dei soli file presenti che iniziano con il carattere **a** oppure il carattere **e** oppure il carattere **c.** Redirigere lo standard output sul file **/exam/exercise2/ls.txt**.

Exercise 3: User and Group

- Create due nuovi gruppi **teachers** e **students**
    - **teachers** con GID 3000
    - **students** con GID 3001
- Creare l'utente appartenente al gruppo **teachers**: **cavatorta**
    - l'utente **cavatorta** avrà le seguenti caratteristiche:
        - **UID** 3000
        - password **cavatorta**
        - dovrà cambiare password una volta ogni 2 mesi e al primo accesso
        - dovrà poter accedere a file e directory di tutti gli utenti appartenenti al gruppo **students**
        - dovrà avere tra i gruppi secondari **anche** il gruppo **wheel**
- Creare l'utente appartenente al gruppo **students**: **rossi**
    - L'utente **rossi** avrà le seguenti caratteristiche:
        - **UID** 3010
        - l'account scadrà dopo un anno dalla sua creazione
        - dovrà avere come gruppo secondario: **users**
        - dovrà utilizzare **/bin/sh** come login shell, al posto di **/bin/bash**
        - questo utente dovrà risultare in stato di **Lock** (non potrà fare accesso al sistema fino a che l'utente root non disabiliterà il blocco)

Exercise 4: file permission

- Creare una directory sotto **/exam/exercise4** dove gli utenti che possono accedere al gruppo **students** potranno condividere files
- Tutti i file creati sotto la directory **/exam/exercise4** dovranno essere assegnati automaticamente al gruppo **students**

Exercise 5: Shell environment and alias command

- Creare un nuovo comando o alias command chiamato ***userinfo*** disponibile al login per **tutti** gli utenti del sistema. Chiamato dovrà stampare la seguente stringa
  `user: <username> - working directory: <print working directory> - home_directory: /home/directory`

Esempio:

```
[student@hostname tmp]$ userinfo
user: student - working directory: /tmp - home_directory:
/home/student
```

Exercise 6: Bash script

- Create uno script bash sotto **/exam/exercise6** chiamato **check.sh** con le seguenti caratteristiche:
    - Accetti in ingresso o un file o una directory
    - Se quanto passato in ingresso non è un file o una directory restituisca il messaggio "**Usage: check.sh <file/directory>"**
    - Se passato in ingresso un file restituisca il messaggio **"Is a file!!"**
    - Se passato in ingresso una directory restituisca il messaggio **"Is a directory!!"**

```
bash check.sh fileName
Is a file!!
bash check.sh directoryName
Is a directory
bash check.sh nonExistentFile
Usage:  check.sh <file/directory>
bash check.sh
Usage:  check.sh <file/directory>
```

Exercise 7: Bash script

- Creare uno script bash sotto **/exam/exercise7** chiamato **manage-package.sh** con le seguenti caratteristiche:
    - Accetti in ingresso due parametri
        - il primo contenente il nome del package
        - il secondo l'azione da intraprendere
            - **search**: effettuerà una ricerca del possibile software da installare restituendo la lista trovata
            - **install**: installerà il software passato come argomento
            - **remove**: rimuoverà il software installato passato come argomento
            - **info**: restituirà le informazioni del software passato come argomento
    - Se non verranno passati argomenti o saranno più o meno di due, o non quelli permessi restituisca il messaggio "**Usage: manage-package.sh <packageName> <search/install/remove/info>" N.B.** Si utilizzi una funzione per stampare a video l'eventuale messaggio di errore.
    - Per le sole azioni **install remove** al termine se andate a buon fine stampi a video il messaggio
        - **<packageName> installed/removed successfully**

      Se non completata con successo stampi a video il messaggio
        - **<packageName> install/remove error**

Exercise 8: Systemd

- Visualizzare la lista di tutte le unit di tipo **service** in stato **active** presenti sul sistema e redirigere tale lista all'interno file **/exam/exercise8/service.unit**

Exercise 9: Docker

- Creare la propria immagine basata su **centos:8**
- La directory **/exam/exercise9** dovrà contenere i files:
  - **Dockerfile**
  - **dockerexam.sh**
- **dockerexam.sh** sarà lo script bash il cui compito sarà quelli di stampare "hello <name>!!" il parametro name dovrà essere una variabile di ambiente contenente di default il valore "exam"
- L'immagine dovrà chiamarsi `exam/myhelloexam`

```
Risultato:
docker run exam/myhelloexam
Hello Exam!!
docker run -e ENV_VAR_NAME="Luca" exam/myhelloexam
Hello Luca!!
```

Exercise 10: Docker compose

- Creare una applicazione wordpress mysql tramite docker-compose con le seguenti caratteristiche
  - porta **locale** wordpress 8081
  - DB_USER=exam
  - DB_PASSWORD=eXaM@1
  - DB_NAME=exercise10
  - Utilizzate un bind locale per i dati delle applicazioni sotto
    - **/exam/exercise10/worpress** per wordpress
    - **/exam/exercise10/mysq**l per mysql
- Utilizzate le immagini ufficiali dal Docker Hub (**hub.docker.com**)
- Una volta lanciata la compose l'applicazione dovrà essere accessibile dall'esterno collegandosi al DNS della vostra istanza AWS su porta 8081 quindi modificate il Firewall per permetterne l'accesso.
- Il file **docker-compose.yml** dovrà essere messo sotto **/exam/exercise10**

Question :

- Si descriva a parole la differenza tra metodo dichiarativo e imperativo riportando un esempio associato anche alla creazione di oggetti all'interno del cluster Kubernetes
    - Salvare la risposta sotto **/exam/question/**