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 **ec2-user** per la connessione al sistema

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

Per passare all'utente **root** utilizzare il comando "sudo su -"
[ec2-user@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_directory_mm_yyyy** con **mm** compreso tra 1 e 12; **yyyy**=2021 e **yyyy**=2024
- sotto ogni directory creata **/exam/exercise1/exercise_directory_mm_yyyy** creare i files
    - **file(a..z)_DATE.txt**
        - con **(a..z)=** tutte le lettere dalla a alla z
        - **DATE** la data di creazione file nel formato ymd (**man date**)

Il risultato sarà il seguente:

```
|-- exercise1_directory_1_2021
|   |-- filea_20210122
|   |-- fileb_20210122
|   `-- filec_20210122
|   etc...
|-- exercise2_directory_2_2021
|   |-- filea_20210122
|   |-- fileb_20210122
|   `-- filec_20210122
|   etc...
etc...
|-- exercise1_directory_1_2024
|   |-- filea_20210122
|   |-- fileb_20210122
|   `-- filec_20210122
|   etc...
|-- exercise2_directory_2_2024
|   |-- filea_20210122
|   |-- fileb_20210122
|   `-- filec_20210122
|   etc...
etc...
```

All'interno della directory /exam/exercise1 creare il file **/exam/exercise1/command.txt** contenente i due comandi utilizzati per creare directory e files

Exercise 2: Managing pipeline and regular expression

Fare una ricerca su tutto il file system ricercando con l'utente **student** tutti i file contenenti il pattern **lib**
Salvare l'output sul file **exam/exercise2/findlib** e gli errori sul file **exam/exercise2/findliberror**

Exercise 3: User and Group

- Create due nuovi gruppi **open-source** e **linux**
    - **open-source** con GID 10000
    - **linux** con GID 10001
- Creare l'utente appartenente al gruppo primario **open-source**: **stallman**
    - l'utente **stallman** avrà le seguenti caratteristiche:
        - **UID** 10010
        - password **stallman**
        - dovrà cambiare password una volta ogni 6 mesi e al primo accesso
        - dovrà poter accedere a file e directory degli utenti appartenenti al gruppo **linux**
        - home directory: **"/home/open-source/stallman"**
        - L'account dovrà scadere a **5 anni** dalla creazione dello stesso
        - non creare il gruppo **stallman**
- Creare l'utente appartenente al gruppo primario **linux**: **torvalds**
    - L'utente **torvalds** avrà le seguenti caratteristiche:
        - **UID** 10020
        - dovrà avere come gruppo secondario: **gnu**
        - dovrà utilizzare **/bin/sh** come login shell, al posto di **/bin/bash**
        - questo utente dovrà risultare in stato di Lock

Exercise 4: Alias and Environment

- Creare un nuovo alias chiamato **hello** disponibile per tutti gli utenti del sistema
- **hello** una volta richiamato dovrà restituire:
    - hello **USERNAME** today is **DATA**
        - **USERNAME** = lo userid dell'utente che lancerà **hello**
        - **DATA =** la data e ora attuale nel formato yyyy-mm-dd HH:MM:SS

Exercise 5: Yum package manager and Systemd

- Installare il servizio "**redis**" sul vostro sistema
- Abilitare il servizio al boot

Exercise 6: Firewalld

- Abilitare le connessioni in ingresso al vostro sistema sulla porta **8081 TCP** solo per la zona **exam1**
- Rendere la regola permanente.

Exercise 7: Bash

- Creare uno script bash sotto **/exam/exercise7** chiamato **filelcounter.sh** che accetti in ingresso un parametro.
- Il parametro in ingresso sarà un file **ASCII text**
- Verificare che il parametro in ingresso sia definito, altrimenti esca con un errore.
- Verificare che il parametro passato contenga un file presente sul sistema altrimenti esca con un errore
- Verificare che il file passato in ingresso sia un file **ASCII text** altrimenti esca con un errore
- Stampi in output il numero di linee presenti all'interno del file (**man wc**)

```
esempio:
# bash filelcounter.sh
ERROR: filelcounter.sh nessun parametro passato o file non trovato
# bash filelcounter.sh /bin/ls
il formato del file deve essere ASCII text
# bash filelcounter.sh /etc/passwd
il file passato contiene linee: 61
```

Exercise 8: Bash

- Creare uno script bash sotto **/exam/exercise8** chiamato **rgb.sh** che accetti in ingresso un parametro.
- Il parametro in ingresso sarà la stringa "**red"** oppure "**green"** oppure "**blue"**
- Passata la stringa **red** l'output dello script sarà "**is red color**", se **green** "**is green color**", se **blue** "**is blue color**".
    - Per qualunque altro parametro passato in ingresso (o in assenza di parametro) lo script dovrà uscire con exit code **255** e restituire l'output: **Usage: rgb.sh <red/green/blue>**
- L'output **Usage** e **exit code** in caso di parametri non corretti **DEVONO** essere gestiti da una **funzione**
- Si utilizzi il case statement per gestire il parametro in ingresso

```
exempio:
# bash rgb.sh
Usage: rgb.sh <red/green/blue>
# echo $?
255
# bash rgb.sh yellow
Usage: rgb.sh <red/green/blue>
# bash rgb.sh red
is red color
```

Exercise 9: Httpd

- Configurare due virtual host per servire i contenuti presenti sotto le directory:
    - **/exam/exercise9/www**
    - **/exam/exercise9/www.exam1.com/www**
- Creare il file /exam/exercise9/www/**index**.**html** con contenuto
  "*Default httpd exam 22/01/2021*"
  Creare il file /exam/exercise9/www.exam1.com/www**/index.html** con contenuto
  "*www.exam1.com*"
- Configurare i due virtual host per fare in modo che il contenuto servito da URL: **www.exam1.com** e **exam1.com** sia "*www.exam1.com*". Per ogni altra richiesta via http, il contenuto che il server dovrà erogare sarà "Default httpd exam 22/01/2021"
- Si aggiungano le regole firewall necessarie per connessione dall'esterno in maniera permanente.
- Per fare il test dalla vostra macchina locale dovrete inserire nel file **hosts** del vostro sistema la entry
  IPPUBBLICO www.exam1.com exam1.com

  ```
  esempio:
  sul sistema ec2-35-152-64-166.eu-south-1.compute.amazonaws.com
  IPPUBBLICO=35.152.64.166
  # curl 35.152.64.166
  Default httpd exam 22/01/2021
  # curl exam1.com
  www.exam1.com
  ```

Exercise 10: Docker and Docker-compose

- Creare la propria immagine basata su **centos:8** chiamandola **exam/exercise10:1.0**
- La directory **/exam/exercise10** dovrà contenere i files:
    - **Dockerfile**
    - **docker-compose.yml**
    - **hello.sh**
- **hello.sh** dovrà contenere lo script bash il cui compito sarà di stampare “hello **exam**!!”. La stringa **exam** dovrà poter essere sostituita passando una variabile di ambiente allo start del container.
- preparare il file **docker-compose.yml** per avviare il container tramite **docker-compose**.
    - la build della immagine può essere gestita tramite la compose, definendola nel file **docker-compose.yml** , oppure potete scegliere di creare l’immagine da shell tramite la cli docker (**docker build).**

```
esempio:
docker-compose up
[...]
Successfully built 27ae437e5975
Successfully tagged exam/exercise10:1.0
[...]
Creating root_hello_1 ... done
Attaching to root_hello_1
hello_1  | Hello exercise10
root_hello_1 exited with code 0
```

Question:

creare il file /exam/question/question.txt
Abbiamo analizzato durante il corso l’oggetto **deployment** di kubernetes.
Con deployment abbiamo visto come sia possibile fornire aggiornamenti dichiarativi alle applicazioni in container, consentendo di descriverne il ciclo di vita, con aspetti quali le immagini da utilizzare, il numero di pod necessari e le modalità di aggiornamento.
Si descriva a parole quali passaggi vengono messi in atto nel caso in cui si debba ad esempio, aggiornare l’immagine di una applicazioni in container definita sul nostro oggetto deployment.
Quali oggetti kubernetes vengono utilizzati, come il controller di kubernetes interagisce per gestire la procedura di update tra i vari oggetti impattati. Si riporti inoltre un esempio di oggetto deployment descritto tramite file yaml **(exam.yaml** da creare nel path **/exam/question/exam.yaml)**, supponendo di dover gestire una applicazione basata sulla immagine “**exam/applicationexam:1.0**” con lable exam su 4 repliche attive e strategy rollingUpdate.