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: Regular Expressions

- Utilizzate il file **/exam/exercise1/text_file** per trovare tutte le occorrenze che:
    - iniziano con un **numero**
    - iniziano e finiscono con una **A**
    - finiscono con una lettera **n** oppure **e**
    - contengono il carattere **a** oppure **b** oppure **c** o combinazioni dei 3
- Riportare in **/exam/exercise1/regex_file** le regular expression
- Potete testare la vostra regular expression con il comando **grep** (grep <regex> /exam/exercise1/text_file)

Exercise 2: File permission and users

- Creare uno script bash che stampi in output “hello world” chiamato **hello**. Il file dovrà appartenere all’utente e gruppo **root.**
- Assicuratevi possa essere lanciato da **qualunque** utente del sistema come comando

example:

```
[student@ip-172-31-35-174 ~]$ hello
hello world!!
[root@ip-172-31-35-174 exercise1]# hello
hello world!!
```

Exercise 3: Default file permission

- Creare l’utente **bill** appartenente al gruppo **microsoft**.
- Fare in modo che nuovi file e directory creati dall’utente **bill**, di default (al login quindi), non possano essere letti scritti o visti, da nessuno al di fuori dell’utente **bill**.
- l’utente dovrà poter accedere a file e directory appartenenti al gruppo **collaboration**
- L’account utente dovrà scadere a 235 giorni dalla sua creazione
- L’utente dovrà cambiare password ogni 37 giorni

Exercise 4: HTTPD

- Installare sul sistema il servizio HTTP/Apache
- Fare in modo che HTTPD venga lanciato al boot della macchina
- Il servizio sarà in ascolto sulla **SOLA** porta **8088**
- Aggiungere la regole firewall per poter accedere dall’esterno al servizio sulla porta indicata
- Fare in modo che la Document Root impostata per il vostro servizio sia **/exam/exercise4**
- creare il file **/exam/exercise4/index.html** contenente la stringa “Hello Exam!”
- Potete verificare che il tutto funzioni collegandovi all’indirizzo IP della vostra macchina AWS dal browser locale alla vostra postazione:
    - http://ec2-xx-xxx-xx-x.eu-central-1.compute.amazonaws.com:8088

Exercise 5: Bash script

- Create uno script bash sotto **/exam/exercise5** chiamato **sum_even_odd.sh** con le seguenti caratteristiche:
    - accetti in ingresso un lista di numeri
    - lo script dovrà stampare in output:
        - la somma dei soli numeri pari
        - la somma dei soli numeri dispari
        - Se non vengono passati almeno due parametri in ingresso riporti un errore generico a piacere
        - Se il parametro passato non è un numero riporti l'errore: ERROR only number please!

example:

```
bash /exam/exercise5/sum_even_odd.sh 1 2 3 4 5
Somma dei numeri pari: 6
Somma dei numeri dispari: 9
```

Exercise 6: docker-compose

- La directory **/exam/exercise6** dovrà contenere i seguenti files e directories:
  - **httpd/Dockerfile**
  - **httpd/entrypoint.sh**
  - **content_management/Dockerfile**
  - **content_management/entrypoint.sh**
  - **content/index.html**
  - **docker-compose.yml**
- Il servizio tramite **docker compose** dovrà gestire due applicazioni. La prima costituita da un servizio *HTTPD* e la seconda da un *content management* con le seguenti caratteristiche:
  - HTTPD: sarà un servizio Apache installato su immagine di base **fedora:latest** in ascolto sulla **porta 80** che andrà a montare sulla DocumentRoot il contenuto della **index.html** presente sul sistema sotto la directory **content** tramite bind locale.
  - Content Management: sarà un'immagine contenente un processo (**script bash)** il cui compito sarà quello di modificare il contenuto della **index.html** presente sotto la directory **content** montata in bind al container ogni due minuti. Il contenuto della **index.html** dovrà essere definito a partire da una variabile di ambiente chiamata **TIME_FORMAT** nel seguente modo:
    - **TIME_FORMAT**=MM-YYYY il contenuto sarà l'output del comando **date +%m-%Y**
    - **TIME_FORMAT =**MM-DD-YYYY il contenuto sarà l'output del comando **date "+%A %d-%B, %Y"**
- **docker-compose.yml** verrà utilizzato per gestire start/build delle due immagini, gestire la variabile TIME_FORMAT, e il mount verso i due servizi in container della directory **content**
  - **ATTENZIONE:** sul sistema linux utilizzato docker compose si lancia tramite command: ***docker-compose*** (non *"docker compose"*)
  - per gestire la build su più directory utlizzare `build` nel seguente modo

```
services:
  ServiceName:
    build:
      context: ./httpd <--- directory contenente Dockerfile
```

Question :

le risposte andranno messa sotto la directory **/exam/question/**

- Applicazioni installate su sistema operativo standard, e in container. Quali le differenze principali.
- Cosa si intende per *platform as a service* e perché possiamo considerare kubernetes un servizio di questo tipo.