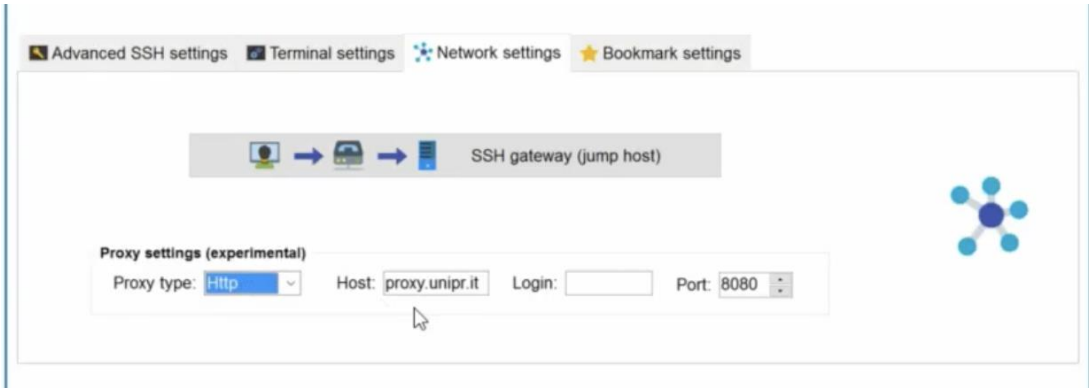
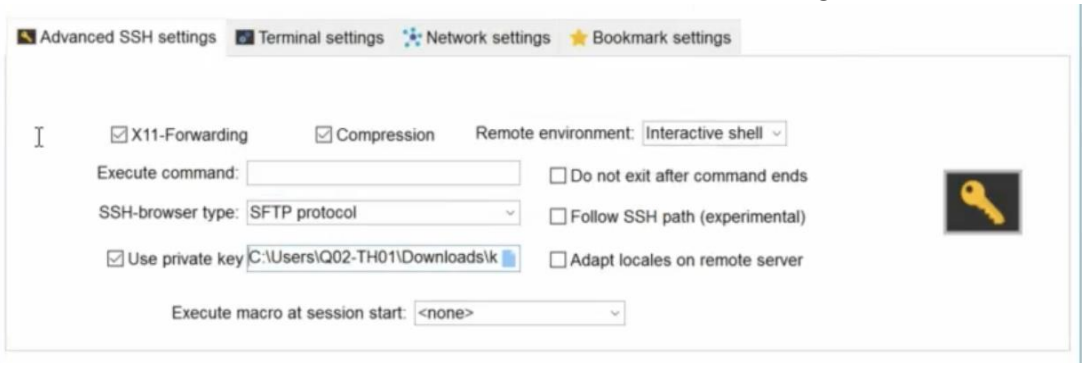
Connessione alla propria macchina

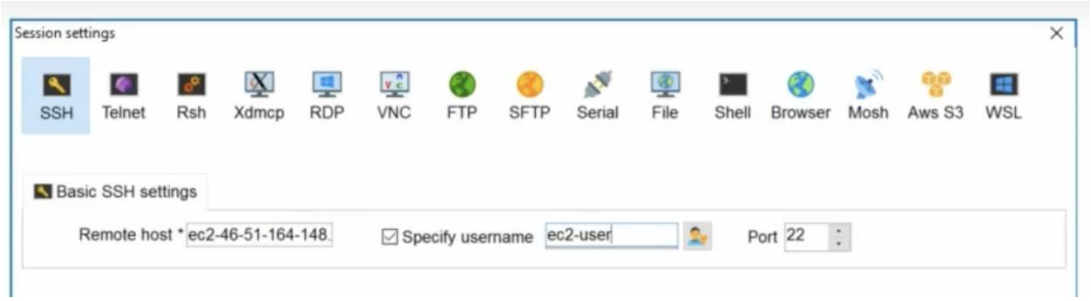
1. Aprire MobaXterm.
2. Tab session / SSH
3. Configurare il proxy dal tab Network Setting
    a. Httpd
    b. proxy.unipr.it
    c. 8080



4. Passare la chiave SSH dal tab “Advanced SSH settings”



5. Inserire in Remote Hosts il dns della vostra istanza Linux
    a. esempio: ec2-xx-xxx-xx-x.eu-central-1.compute.amazonaws.com
6. Username: **ec2-user**
7. port: **22**



Video accesso ai sistemi cloud dalle postazioni UNIPR:
accesso_aws_pld_laboratori.mov

Per passare all’utente **root** utilizzare il comando “**sudo su -**”
[ec2-user@ip-xxx-xx-xx-xxx ~]$ **sudo su -**

Exercise 1: Regular Expressions

- Create il file **/exam/exercise1/text_file** contenente l'output del comando "**ls -1 /"** per trovare tutte le occorrenze che:
  - contengano un **numero**
  - iniziano e finiscono con una **t**
  - finiscano con un numero
  - 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)

```
cd exam/exercise1/
touch text_file
ls -1 / > text_file
touch regex_file
vim regex_file (e inserire:)
      [0-9]
      ^t.*t$
      [0-9]$
      [abc]
grep -E -f regex_file text_file
```

Exercise 2: File permission and users

- Creare uno script (**NON UN ALIAS**) bash chiamato **user_name** che possa essere richiamato come comando senza specificare il path in cui si trova, e che stampi in output il nome dell'utente che lo lancia.
- Assicuratevi possa essere lanciato da **qualunque** utente del sistema

example:

```
[student@ip-172-31-35-174 ~]$ user_name
student
[root@ip-172-31-35-174 exercise1]# user_name
Root
```

```
cd /usr/local/bin
vim user_name (e inserire:)
      echo $USER
chmod +x user_name
```

Exercise 3: Default file permission

- Creare l'utente **john** appartenente al gruppo **smith**.
- Fare in modo che nuovi file e directory creati dall'utente **john**, di default (al login quindi), non possano essere letti scritti o visti, da nessuno al di fuori dell'utente **john**.

```
groupadd smith
useradd -g smith john
(cambio utente in john)
umask 077 (non toglie niente ad user, mentre toglie tutto a group e other)
```

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 **8081**
- 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 collegandosi 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:8081

Exercise 5: Bash script

- Create uno script bash sotto **/exam/exercise5** chiamato **even_odd.sh** con le seguenti caratteristiche:
    - accetti in ingresso un lista di numeri
    - lo script dovrà stampare in output:
        - la lista dei soli numeri pari inseriti
        - la lista dei soli numeri dispari inseriti
        - 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/even_odd.sh 1 2 3 4 5
numeri pari: 2 4
numeri dispari: 1 3 5
```

```
#!/bin/bash

if [ "$#" -lt 2 ]; then
    echo "Almeno 2 parametri richiesti."
    exit 1
fi

## non numeri
for parameter in "$@"; do
    if ! [[ $parameter =~ ^[0-9]+$ ]]; then
        echo "ERROR, only number please!"
        exit 1
    fi

done

## numeri pari
echo -n "numeri pari:"
for parameter in "$@"; do
    if (( $parameter % 2 == 0 )); then
        echo -n " $parameter"
    fi
done

## numeri dispari
echo
echo -n "numeri dispari:"
for parameter in "$@"; do
    if ! (( $parameter % 2 == 0 )); then
        echo -n " $parameter"
    fi
done
echo

exit 0
```

Exercise 6: docker

- Creare una nuova immagine Docker basata su fedora che chiamerete **exam/exercise10:1.0**
- Il compito di questa immagine una volta lanciata, sarà quello di stampare a video la stringa “Hello Student” ogni 2 secondi, per un massimo di 5 volte, e poi uscire stampando la stringa “goodbye!!”
- Fare in modo che la parola Student possa essere modificata tramite variabile di ambiente passata allo start del container.

```
mkdir exercise10
cd exercise10/
```

```
vi Dockerfile (e inserire: )
# Usa l'immagine base di Fedora
FROM fedora

# Copia lo script all'interno dell'immagine
COPY script.sh /script.sh

# Imposta la variabile d'ambiente con il valore di default "Student"
ENV STUDENT_NAME Student

# Esegui lo script quando il container viene avviato
CMD ["bash", "/script.sh"]
```

```
vi script.sh (e inserire: )
#!/bin/bash

# Funzione per la stampa periodica
print_message() {
    local counter=0
    while [ $counter -lt 5 ]; do
        echo "Hello $STUDENT_NAME"
        counter=$((counter+1))
        sleep 2
    done
    echo "Goodbye!!"
}
# Esegui la funzione per la stampa periodica
print_message
```

```
chmod +x script.sh
sudo systemctl start docker (SE NON SI HA DOCKER RUNNATO)
docker build -t exam/exercise10:1.0 .
docker run -e STUDENT_NAME="Pippo" exam/exercise10:1.0
```

Question :

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

- Quali sono le funzioni presenti all'interno di un orchestratore di container come Kubernetes più vantaggiose rispetto ad una o più applicazioni in container gestite tramite ad esempio il tool Docker Compose.
- Cosa si intende per *platform as a service* e perché possiamo considerare kubernetes un servizio di questo tipo.