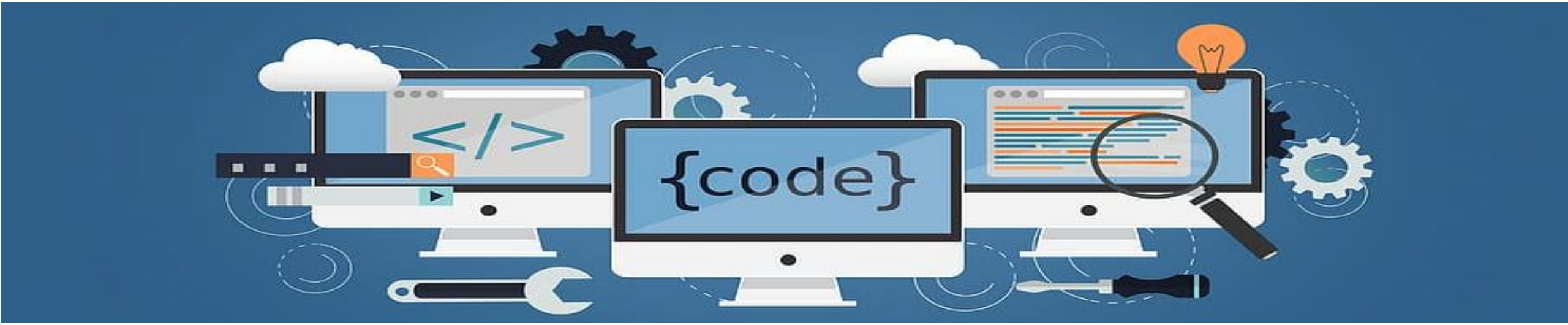
DES424
Cloud-based Application Development

Tutorial: Basic Docker Tutorial



Dr. Apichon Witayangkurn
(apichon@siit.tu.ac.th)

Roadmap

- Install Docker Engine
- List and reuse container
- Run Ubuntu container
- Run Web server (nginx) container
- Build custom image with your web

Install VM → learner lab create EC2.

- Ubuntu version 20.x on Virtual machine environment (VMware player, VirtualBox, EC2)
- For EC2, Ubuntu 20.04LTS, t2.large,disk 80GB

Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE L

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-06aa3f7caf3a30282 (64-bit (x86)) / ami-0a75bd84854bc95c9 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Canonical, Ubuntu, 20.04 LTS, amd64 focal image build on 2023-10-25

Architecture: 64-bit (x86)
AMI ID: ami-06aa3f7caf3a30282 — Verified provider

Instance type Info

Instance type: t2.large
Family: t2 2 vCPU 8 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.1208 USD per Hour
On-Demand RHEL base pricing: 0.1528 USD per Hour
On-Demand SUSE base pricing: 0.1928 USD per Hour
On-Demand Linux base pricing: 0.0928 USD per Hour

All generations
Compare instance types

Additional costs apply for AMIs with pre-installed software

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group | Select existing security group

We'll create a new security group called **'launch-wizard-3'** with the following rules:

- [x] Allow SSH traffic from — Anywhere 0.0.0.0/0
  Helps you connect to your instance
- [x] Allow HTTPS traffic from the internet
  To set up an endpoint, for example when creating a web server
- [x] Allow HTTP traffic from the internet
  To set up an endpoint, for example when creating a web server

Storage (volumes) Info — Simple

EBS Volumes — Hide details

Volume 1 (AMI Root) (Custom)

Storage type: EBS | Device name - *required*: /dev/sda1 | Snapshot: snap-0c031a46fe3d3cc7c

80GB

Size (GiB): 80 | Volume type: gp2 | IOPS: 240 / 3000

Delete on termination: Yes | Encrypted: Not encrypted | KMS key: Select

KMS keys are only applicable when encryption is set on this volume.

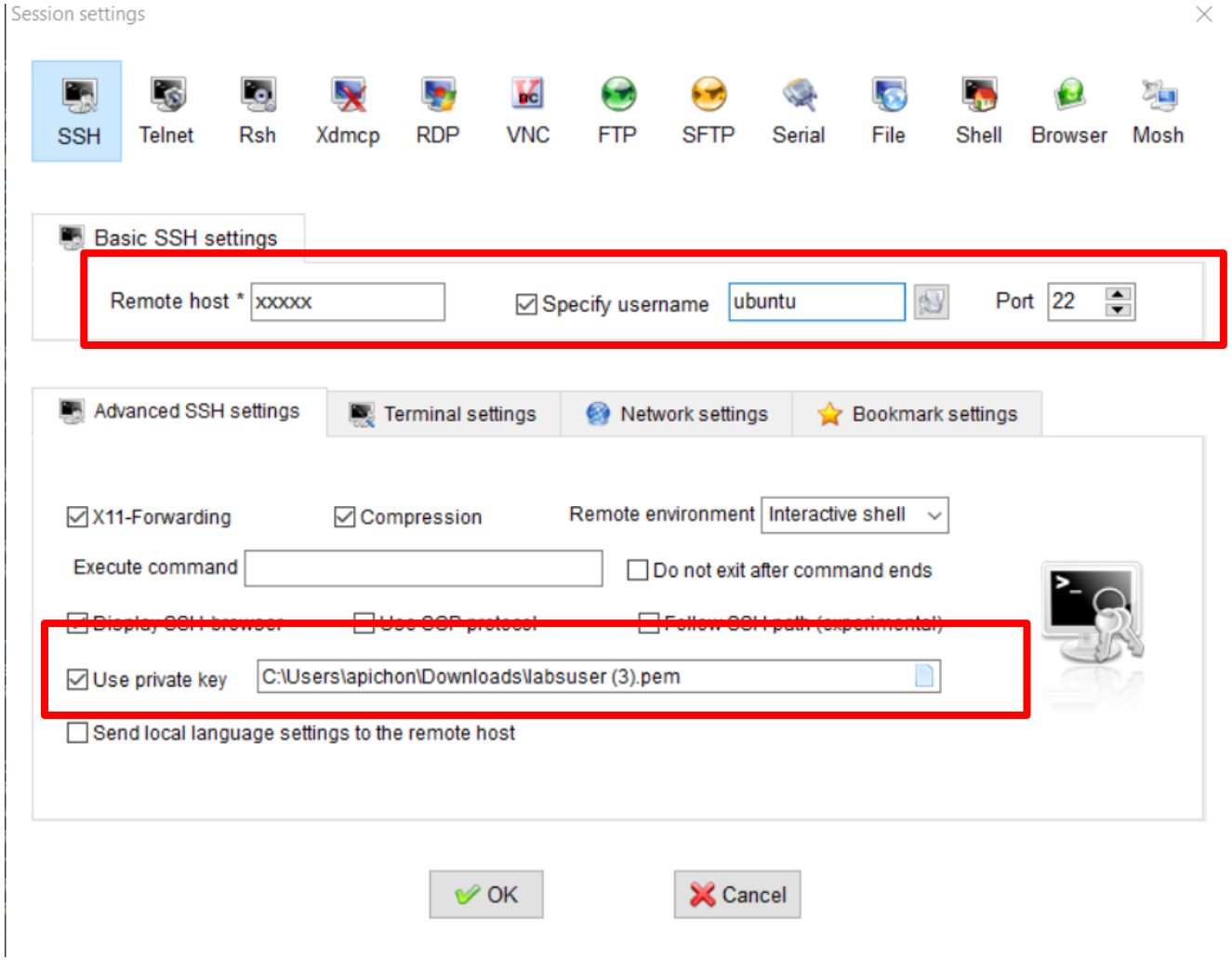
Key pair (login) Info

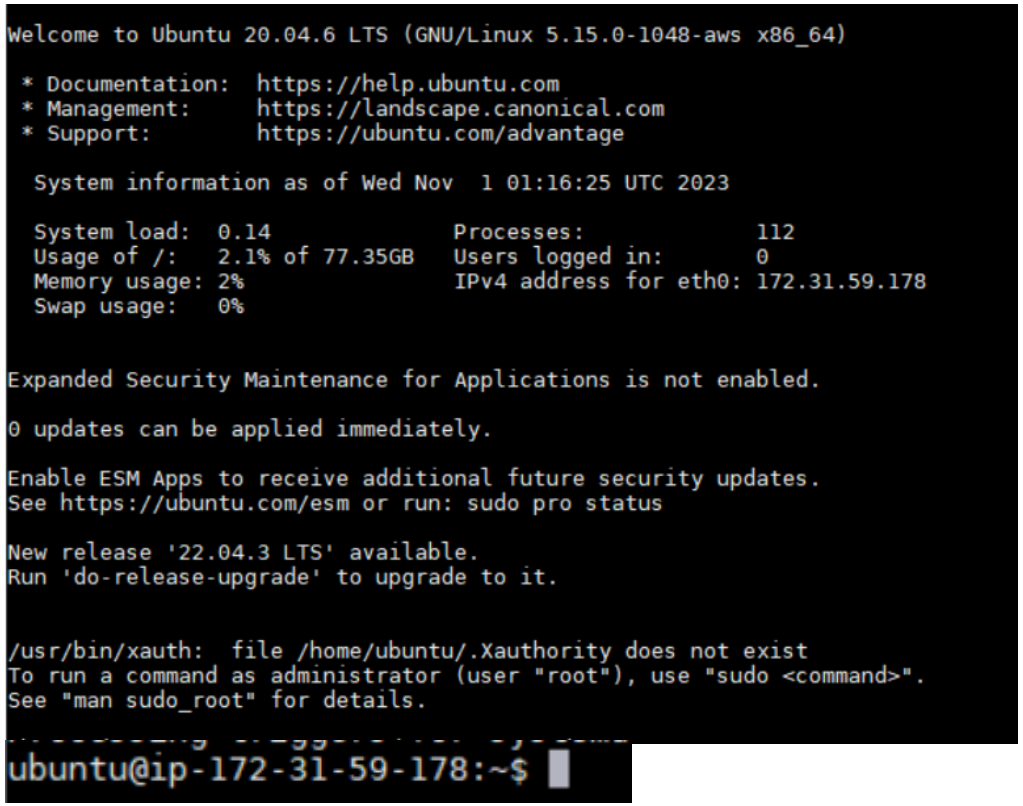
You can use a key pair to securely
before you launch the instance.

Key pair name - *required*: vockey

Connect to your VM with ssh

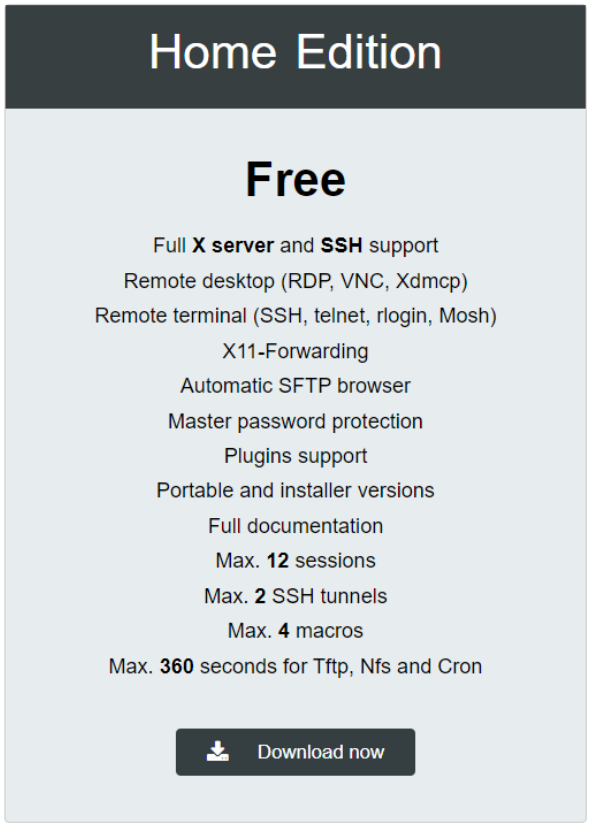
- Identify IPv4 and use user: ubuntu and key for access

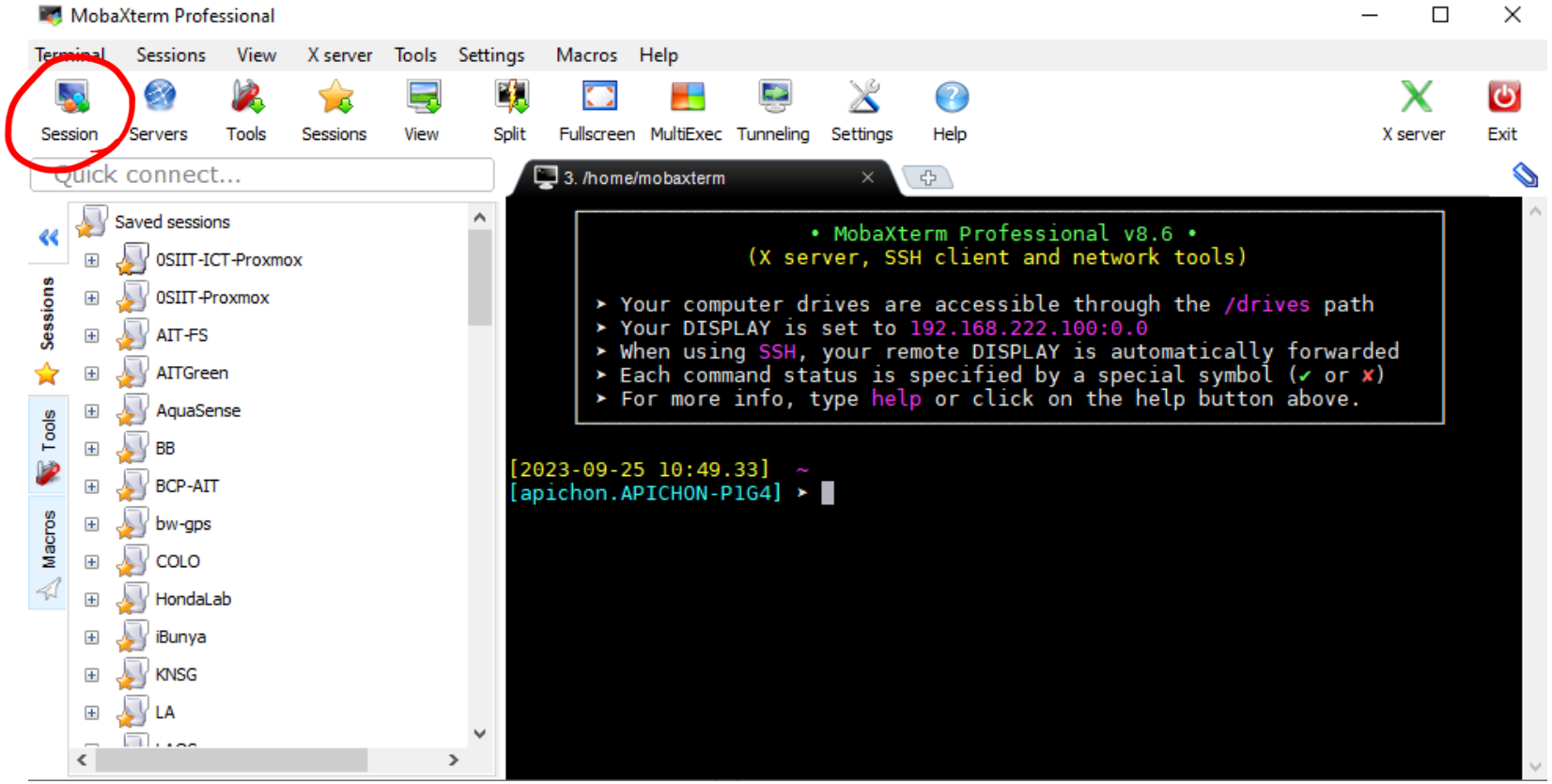




Use Mobaxterm to remote ssh

- For Windows, https://mobaxterm.mobatek.net/download.html
- Select Home Edition (Free)

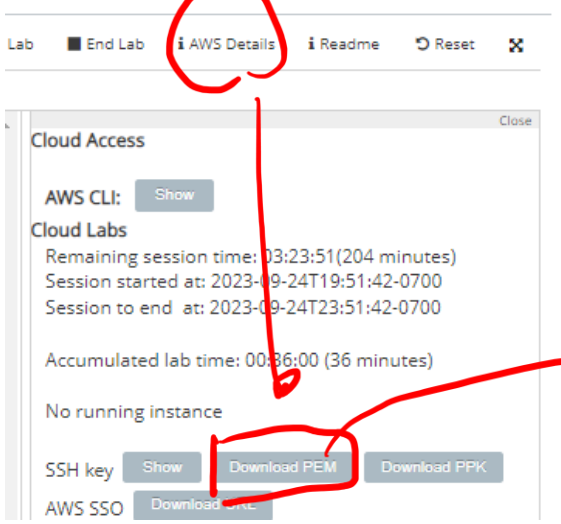


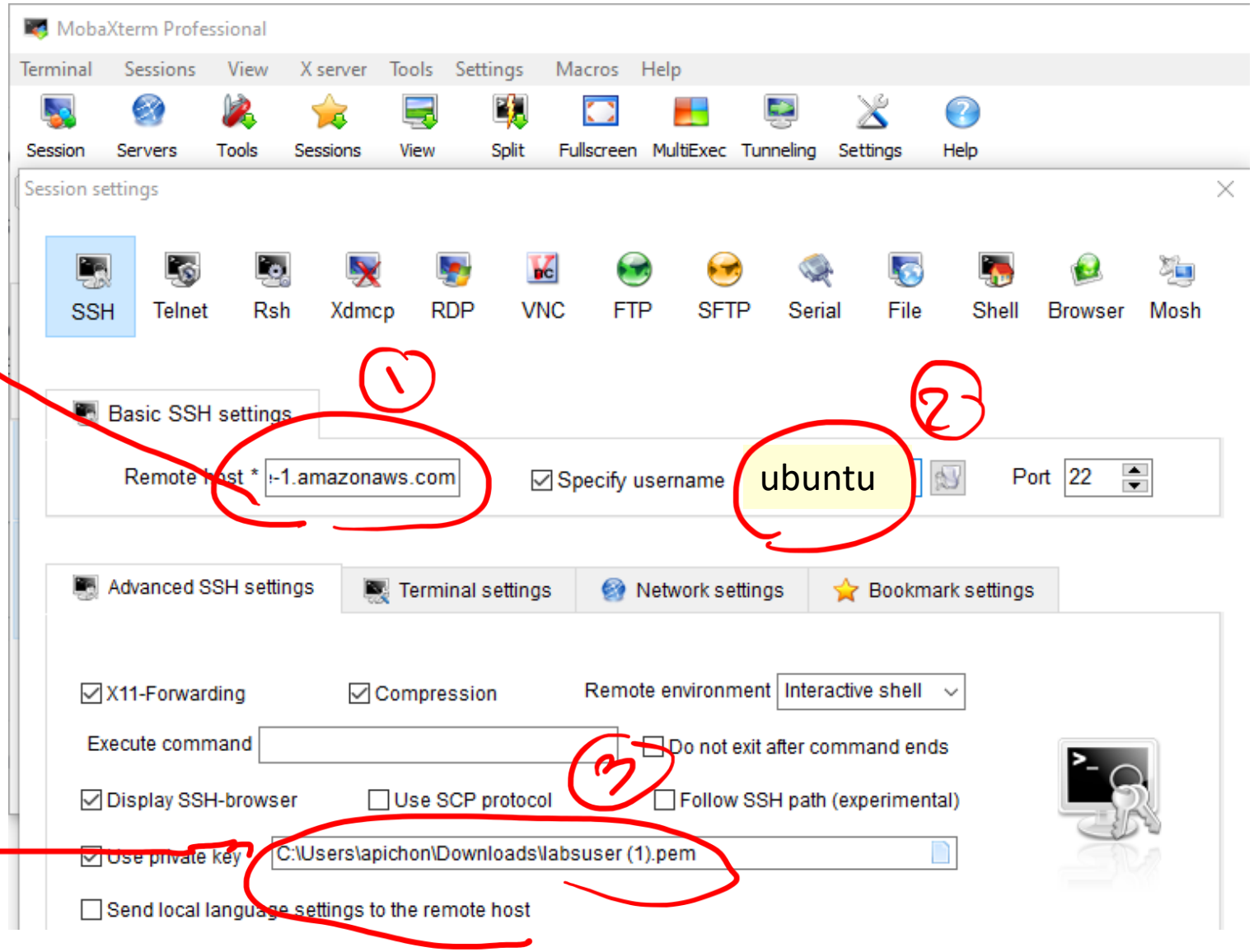


Use Mobaxterm to remote ssh

- Create new session
- Host: copy from primary node DNS

  Primary node public DNS
  ec2-3-91-90-3.compute-1.amazonaws.com
- User: ubuntu
- Private key: download PEM from AWS detail of lab learner





Ready to Run

- If it work, you should see prompt for running cli.

```
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1048-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Nov  1 01:16:25 UTC 2023

  System load:  0.14              Processes:             112
  Usage of /:   2.1% of 77.35GB   Users logged in:       0
  Memory usage: 2%                IPv4 address for eth0: 172.31.59.178
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


/usr/bin/xauth:  file /home/ubuntu/.Xauthority does not exist
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Install Docker Engine

- Ubuntu version 20.x on Virtual machine environment (VMware player, VirtualBox)
- Install docker engine (https://docs.docker.com/engine/install/ubuntu/)
- Add docker group to our user.

```
sudo usermod -aG docker $USER
```

- Try running docker command

```
apichon@ubuntu:~/Desktop$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/apichon/.docker")
  -c, --context string     Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context
                           set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
```

Test Docker Engine

- Test your docker engine by running the hello-world image.

```
docker run hello-world
```

- If everything is ok, you should see the following screen.

```
apichon@ubuntu:~/Desktop$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

List and reuse the containers on our system

- docker ps -a lists the containers on our system (-a including exited containers):

```
apichon@ubuntu:~/Desktop$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS                      PORTS     NAMES
1d6db4fab4c3   hello-world   "/hello"   3 minutes ago   Exited (0) 3 minutes ago            stupefied_leakey
bb3567b1ec62   hello-world   "/hello"   7 minutes ago   Exited (0) 7 minutes ago            priceless_meninsky
1afb4a83a119   hello-world   "/hello"   3 days ago      Exited (0) 3 days ago               stupefied_faraday
e71bbac9b352   hello-world   "/hello"   4 days ago      Exited (0) 4 days ago               cool_ganguly
apichon@ubuntu:~/Desktop$
```

- docker start --attach <container name> instead of docker run to reuse the container

```
apichon@ubuntu:~/Desktop$ docker start --attach cool_ganguly

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

Stop and remove the containers on our system

- docker stop <container name> to stop the container:

- docker rm <container name> to remove the container

```
apichon@ubuntu:~/Desktop$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
1d6db4fab4c3   hello-world   "/hello"   8 minutes ago    Exited (0) 8 minutes ago              stupefied_leakey
bb3567b1ec62   hello-world   "/hello"   12 minutes ago   Exited (0) 12 minutes ago             priceless_meninsky
1afb4a83a119   hello-world   "/hello"   3 days ago       Exited (0) 3 days ago                 stupefied_faraday
apichon@ubuntu:~/Desktop$ docker rm stupefied_faraday
stupefied_faraday
apichon@ubuntu:~/Desktop$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
1d6db4fab4c3   hello-world   "/hello"   10 minutes ago   Exited (0) 10 minutes ago             stupefied_leakey
bb3567b1ec62   hello-world   "/hello"   15 minutes ago   Exited (0) 14 minutes ago             priceless_meninsky
```

Try running ubuntu container

- docker run -it ubuntu bash, will download an Ubuntu Linux image and started a login shell as root inside it. The -it flags allow us to interact with the shell

```
apichon@ubuntu:~/Desktop$ docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
cf92e523b49e: Pull complete
Digest: sha256:35fb073f9e56eb84041b0745cb714eff0f7b225ea9e024f703cab56aaa5c7720
Status: Downloaded newer image for ubuntu:latest
root@a0e25010b532:/#
```

Ubuntu shell (try exit to exit shell and exit container)

- Run docker ps to check running container

```
apichon@ubuntu:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS              PORTS     NAMES
a0e25010b532   ubuntu    "bash"    2 minutes ago    Up About a minute             practical_hawking
```

- Execute docker top <container name> to show process in container

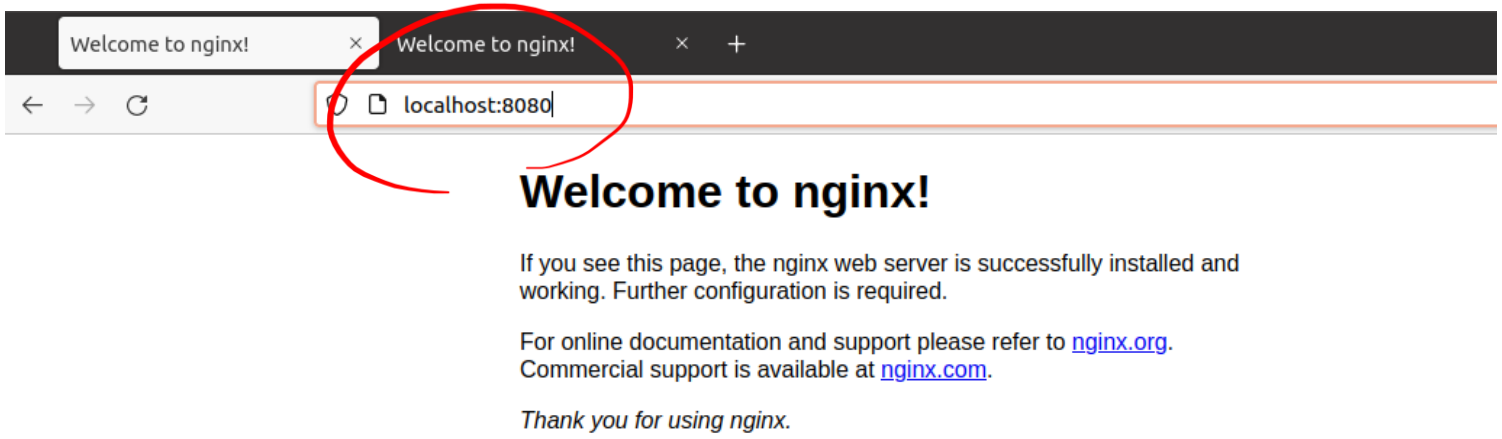
```
apichon@ubuntu:~/Desktop$ docker top practical_hawking
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                2982                2956                0                   20:42               pts/0               00:00:00            bash
```

Test running Web server (nginx) container

- We will download a nginx image and run as daemon with port mapping 8080

```
docker run -p 8080:80 -d nginx
```

- -p for mapping port from host (8080) to container (80)
- -d for leave the container running in non-interactive mode
- Open browser and try http://localhost:8080 or http://127.0.0.1:8080



Test running Web server (nginx) container

- docker exec -it <container_name> bash to enter shell in container

```
apichon@ubuntu:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                                     NAMES
e83ec91fe21d   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp, :::8080->80/tcp   bold_wilson
apichon@ubuntu:~/Desktop$ docker exec -it bold_wilson bash
root@e83ec91fe21d:/# cd /usr/share/nginx/html
root@e83ec91fe21d:/usr/share/nginx/html# ls
50x.html  index.html
root@e83ec91fe21d:/usr/share/nginx/html#
```

- Stop and remove container

```
apichon@ubuntu:~/Desktop$ docker stop xenodochial_rhodes
xenodochial_rhodes
apichon@ubuntu:~/Desktop$ docker rm xenodochial_rhodes
xenodochial_rhodes
```

Let build custom image with your webpage

- Create directory name mytestweb and enter that directory.
- Create directory name html and enter that directory

```
apichon@ubuntu:~/Desktop$ mkdir mytestweb
apichon@ubuntu:~/Desktop$ cd mytestweb/
apichon@ubuntu:~/Desktop/mytestweb$ mkdir html
apichon@ubuntu:~/Desktop/mytestweb$ cd html
apichon@ubuntu:~/Desktop/mytestweb/html$
```

- Create a file named "index.html" with the following content and save it

```
<!DOCTYPE html>
<html>
    <body>
        <h1>My Name: Your name</h1>
        <p>Id: Your ID</p>
    </body>
</html>
```

```
apichon@ubuntu:~/Desktop/mytestweb/html$ ls
index.html
```

Let build custom image with your webpage

- Go back to your mytestweb directory. (cd ..)
- Create a file named "Dockerfile" (no extension) with the following content and save it.

```
FROM nginx
COPY html /usr/share/nginx/html
```

- Check files (ls)

```
apichon@ubuntu:~/Desktop/mytestweb$ ls
Dockerfile  html
```

- Run command to build image ("." = current directory)

```
docker build -t mytestweb .
```

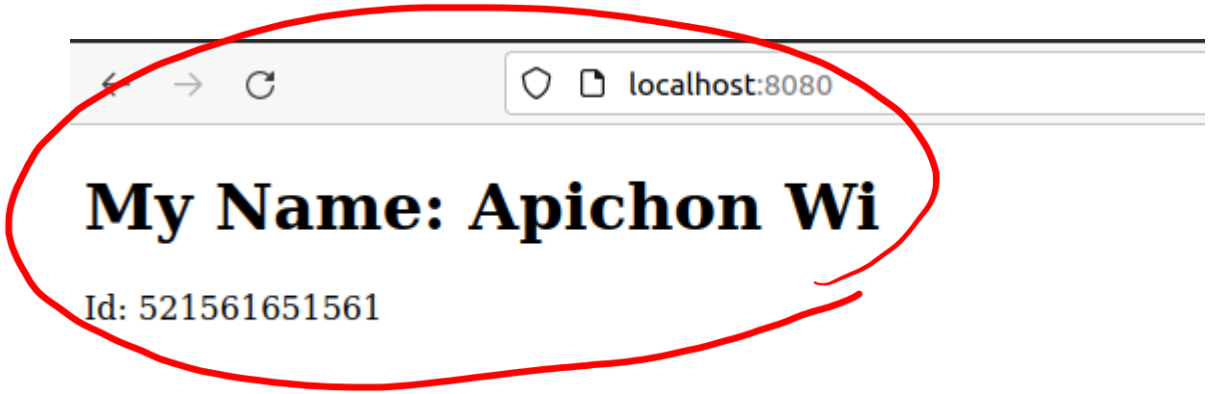
```
apichon@ubuntu:~/Desktop/mytestweb$ docker build -t mytestweb .
Sending build context to Docker daemon  3.584kB
Step 1/2 : FROM nginx
 ---> 51086ed63d8c
Step 2/2 : COPY html /usr/share/nginx/html
 ---> Using cache
 ---> 58ef19a965a7
Successfully built 58ef19a965a7
Successfully tagged mytestweb:latest
```

Let build custom image with your webpage

- Check your image in the registry with docker image ls

```
apichon@ubuntu:~/Desktop/mytestweb$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
mynginx       latest    58ef19a965a7   13 minutes ago   142MB
mytestweb     latest    58ef19a965a7   13 minutes ago   142MB
nginx         latest    51086ed63d8c   13 days ago      142MB
ubuntu        latest    216c552ea5ba   2 weeks ago      77.8MB
hello-world   latest    feb5d9fea6a5   13 months ago    13.3kB
```

- Run your image: docker run --name myweb -d -p 8080:80 mytestweb
- Check your web on browser



If you use EC2, change localhost to public IPv4 and allow port 8080 on security group


To be continued.