Elastic Beanstalk vs EC2

Elastic beanstalk is a wizard for creating EC2 instance of specific platform

Ex:

When we create elastic beanstalk of “Java”, EC2 instance is created and java is installed there.

When we create elastic beanstalk of “Docker”, EC2 instance is created and Docker is installed there.

Demo:

Create an elastic beanstalk application of platform “Docker”

1. Login to aws.amazon.com/console
2. Navigate to Elastic Beanstalk
3. Create a new application

Name: munish

Click the create button

1. Create environment
2. Platform: Docker
3. Sample Application is fine now. Next
4. Service access:

Service role: Use an existing service role

aws-elasticbeanstalk-service-role

EC2 key pair: always recommended to create or select a key pair (else, cannot connect to ec2 instance)

EC2 instance profile: aws-elasticbeanstalk-ec2-role

Next

1. VPC (select default vpc)

Activate public ip

Select instance subnets

Next

1. Skip to review
2. Submit

Now, in EC2 instances, I able to see “Munish-env”

When I try to connect to the ec2 instance, it is not allowed because, we did not create .pem key pair.

In elastic beanstalk, environment, configuration,

I am allowed to edit the “service access” tab and add a key-pair.

This will cause the env to terminate and re launch.

After that,

We can go to EC2 instance and connect to Munish-env instance.

Connected.

Check docker version

But when we run a docker image

docker run –p 5000:5000 jagindia/18-jul-employee-h2

we got error because, t3-micro does not have memory to run this image.

How to upgrade to t2-small?

In ec2 we do not find option.

In elastic beanstalk, env configuration, “Instance traffic and scaling”, edit

And add an instance type “t2-small” do not remove other types.

<http://ashwin.ap-south-1.elasticbeanstalk.com:5000/employee>

Task:

Create an elastic beanstalk application of “Docker” platform and use instance type as “t3-medium”

And connect to EC2 instance of that env.

Check docker version

Run a docker image

Curl in ec2 conntected using ssh client

Then check in browser, replace localhost by public ip

If not working, then go to security group and enable “all traffic” and anywhere.

Podman:

Prerequisite:

Open cmd and run:

winget install Microsoft.WindowsTerminal

install podman: version 4.1 or above:

<https://github.com/containers/podman/releases/tag/v4.6.0>

scroll down to Assets:

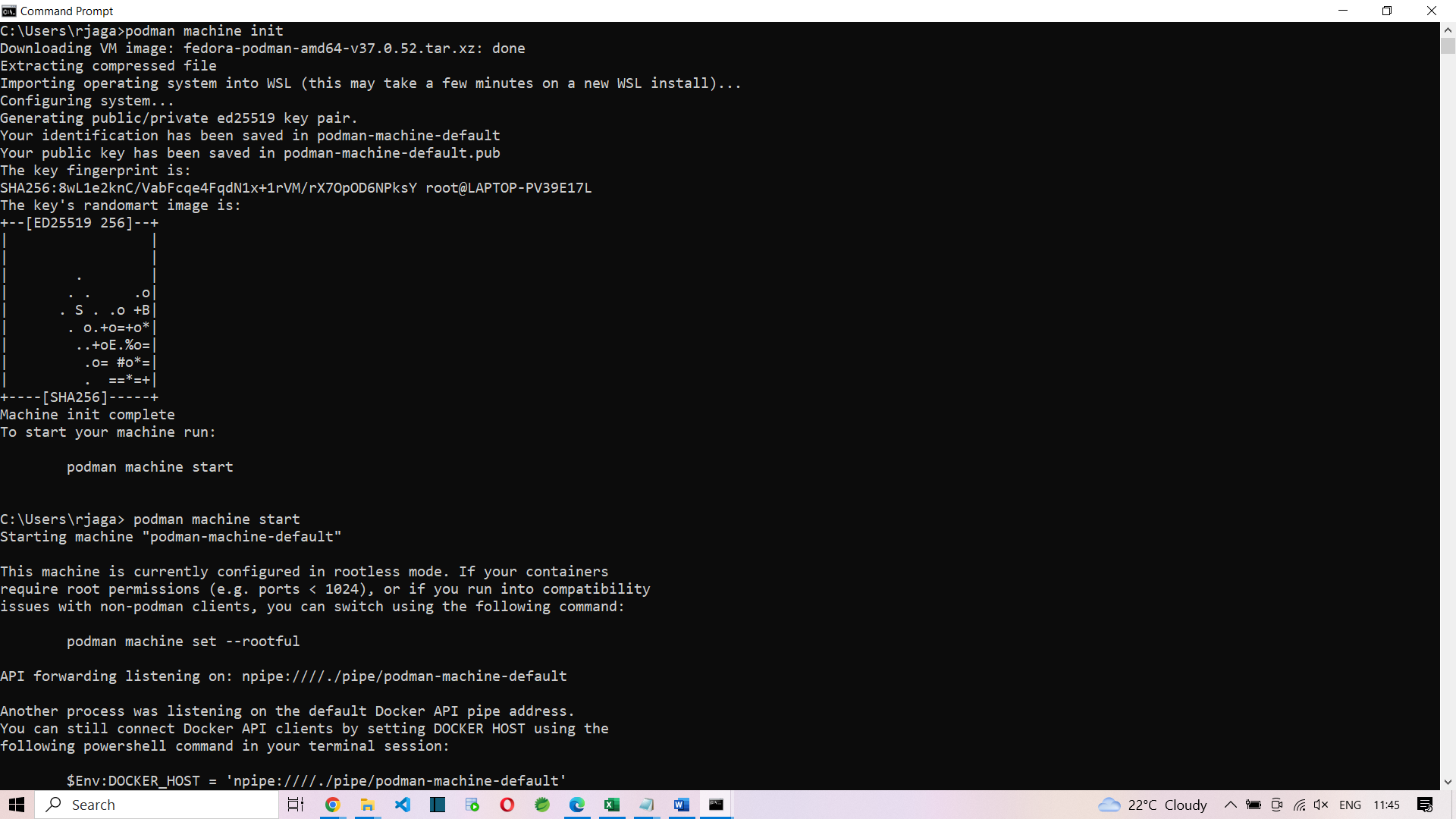
[podman-4.6.0-setup.exe](https://github.com/containers/podman/releases/download/v4.6.0/podman-4.6.0-setup.exe)

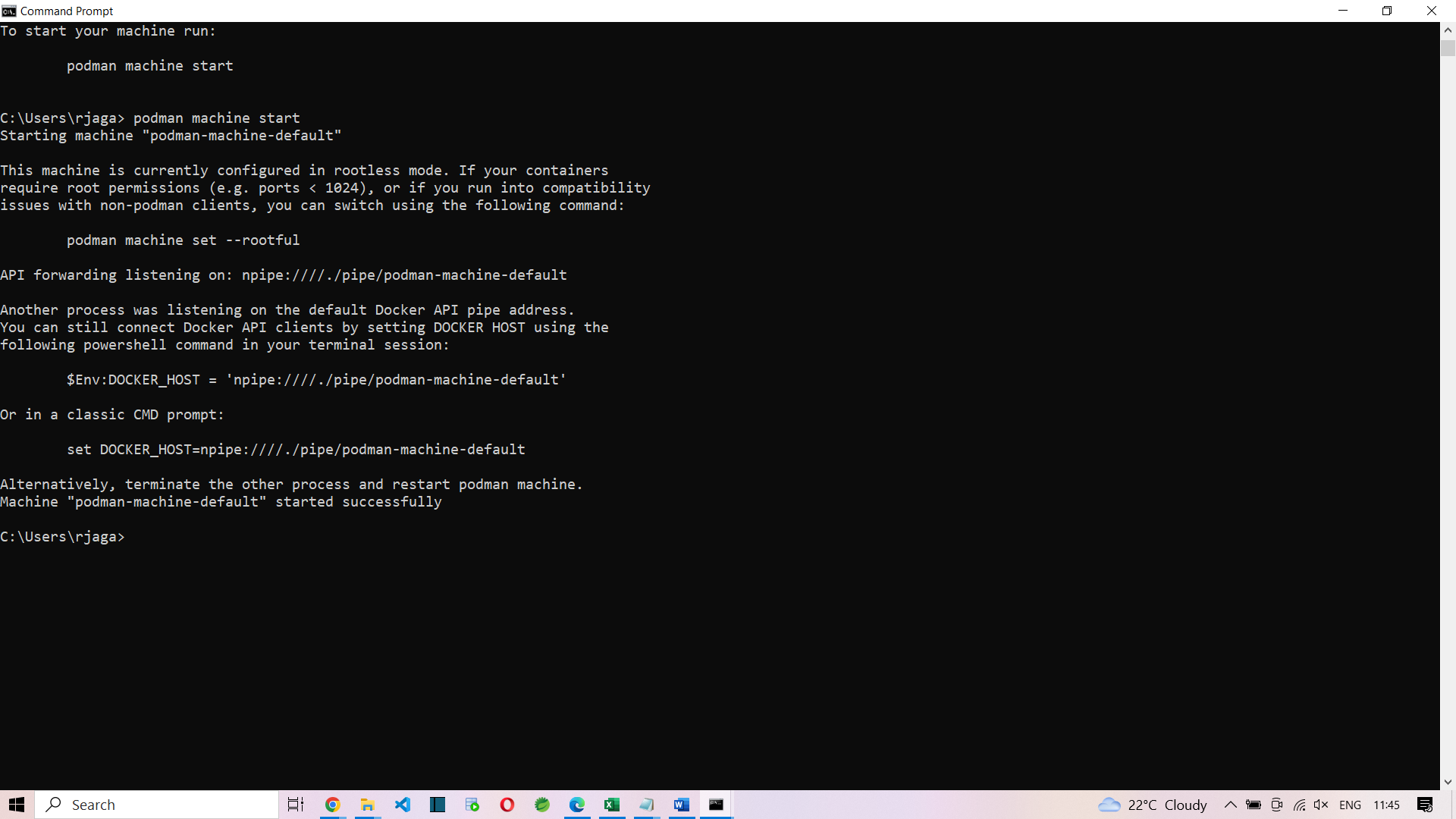
download and install podman.

Tick the “wsl” if not installed

Open new cmd prompt:

podman machine init

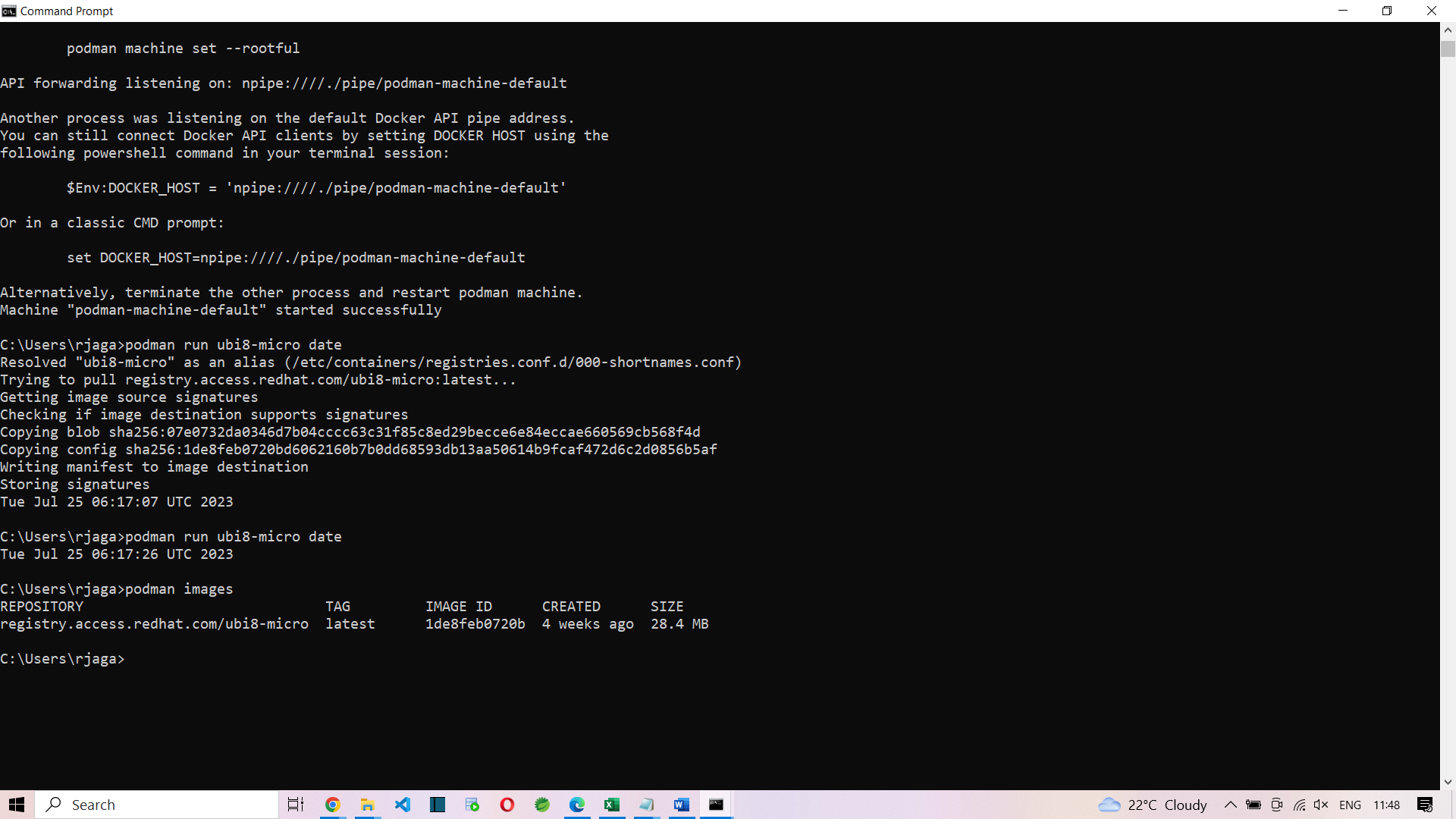




Run your first command in podman:-

podman run ubi8-micro date

first, it checks if this images is present locally. Else it will pull the image and run.



To view all images present locally:

podman images

podman image ls

podman container ls

podman container ls -a

podman ps

podman ps -a

<https://github.com/containers/podman/blob/main/docs/tutorials/podman-for-windows.md>

podman system prune -a

podman login -u myuser -p mypassword docker.io/myuser/myimage

podman build -t localhost/myimage .

$ podman images

REPOSITORY TAG IMAGE ID CREATED SIZE

localhost/myimage latest 1546573dd25d About a minute ago 716 MB

docker.io/library/docker 19.03.12-dind 66dc2d45749a 20 months ago 227 MB

ex:

podman push 1546573dd25d docker://docker.io/myuser/myimage:1.0.0

ex:

podman push 7aba14b55f49 docker://docker.io/jagindia/kalyan