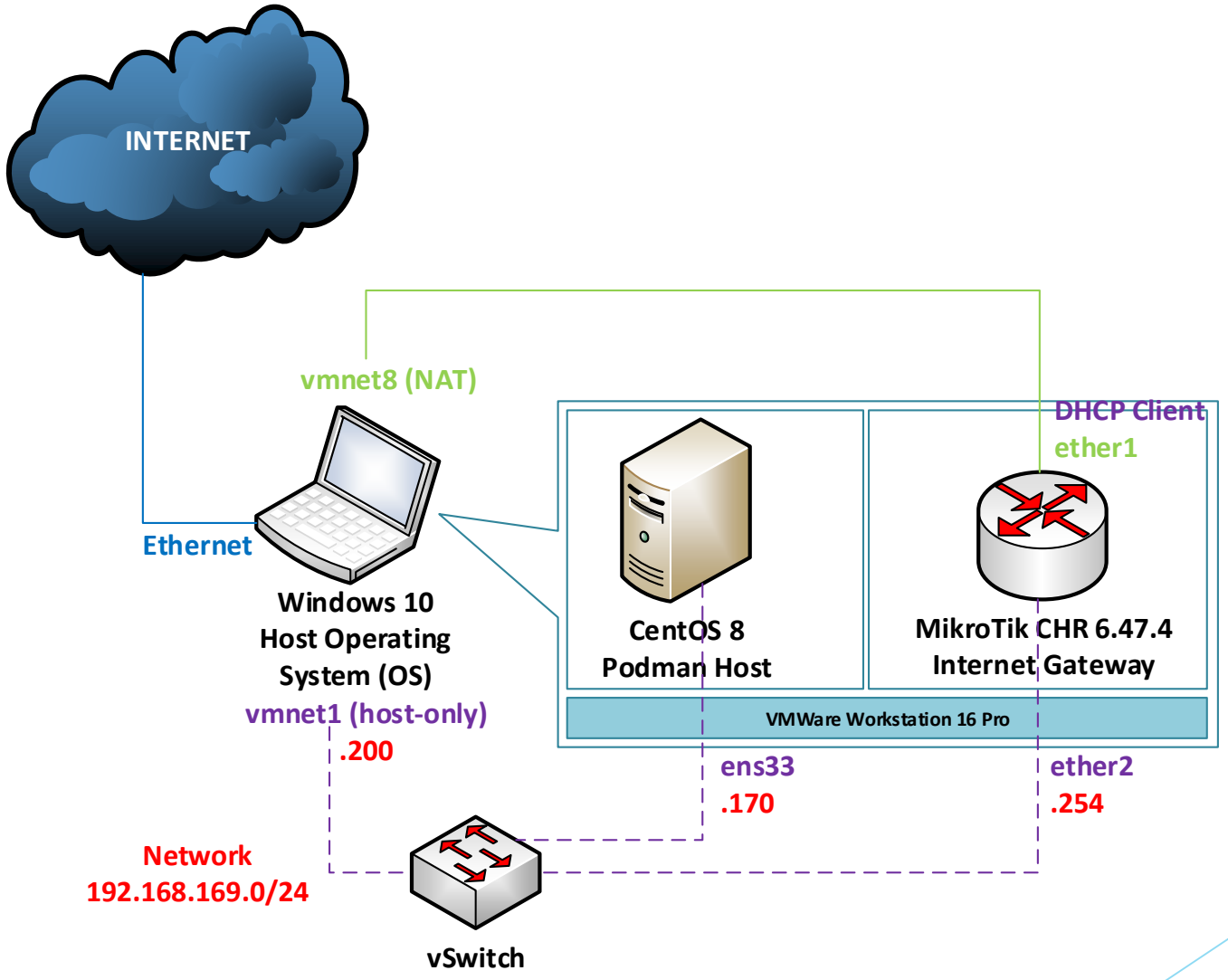
PODMAN

The Next Generation of Linux Container Tools

I PUTU HARIYADI

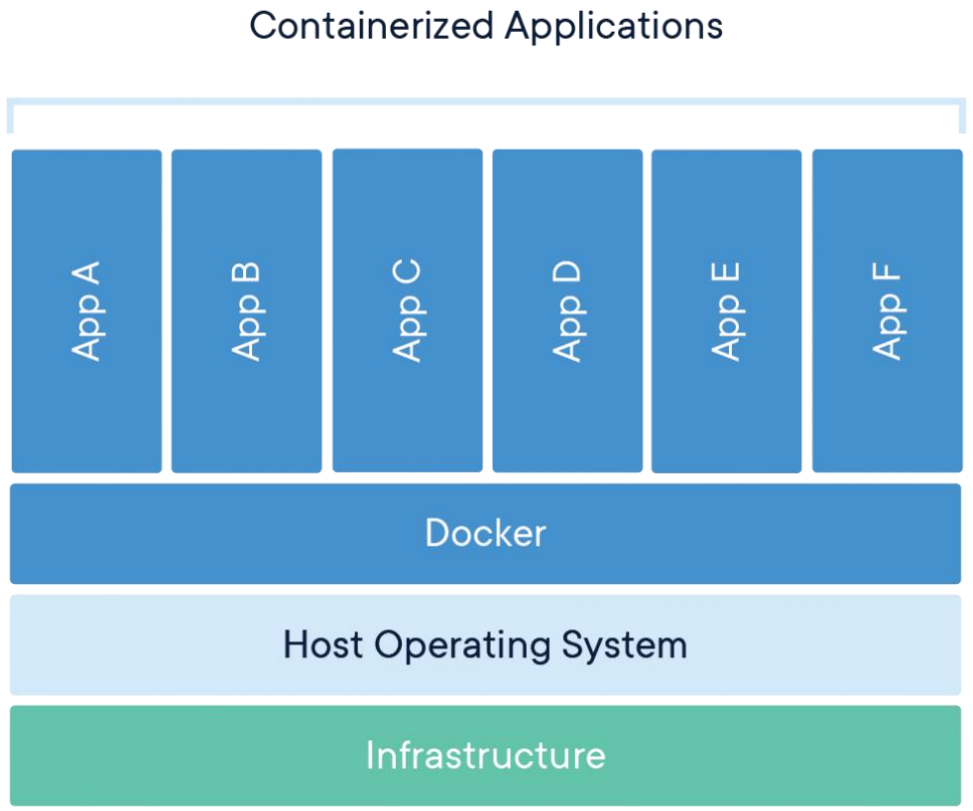
putu.hariyadi@universitasbumigora.ac.id

RANCANGAN JARINGAN UJICOBA



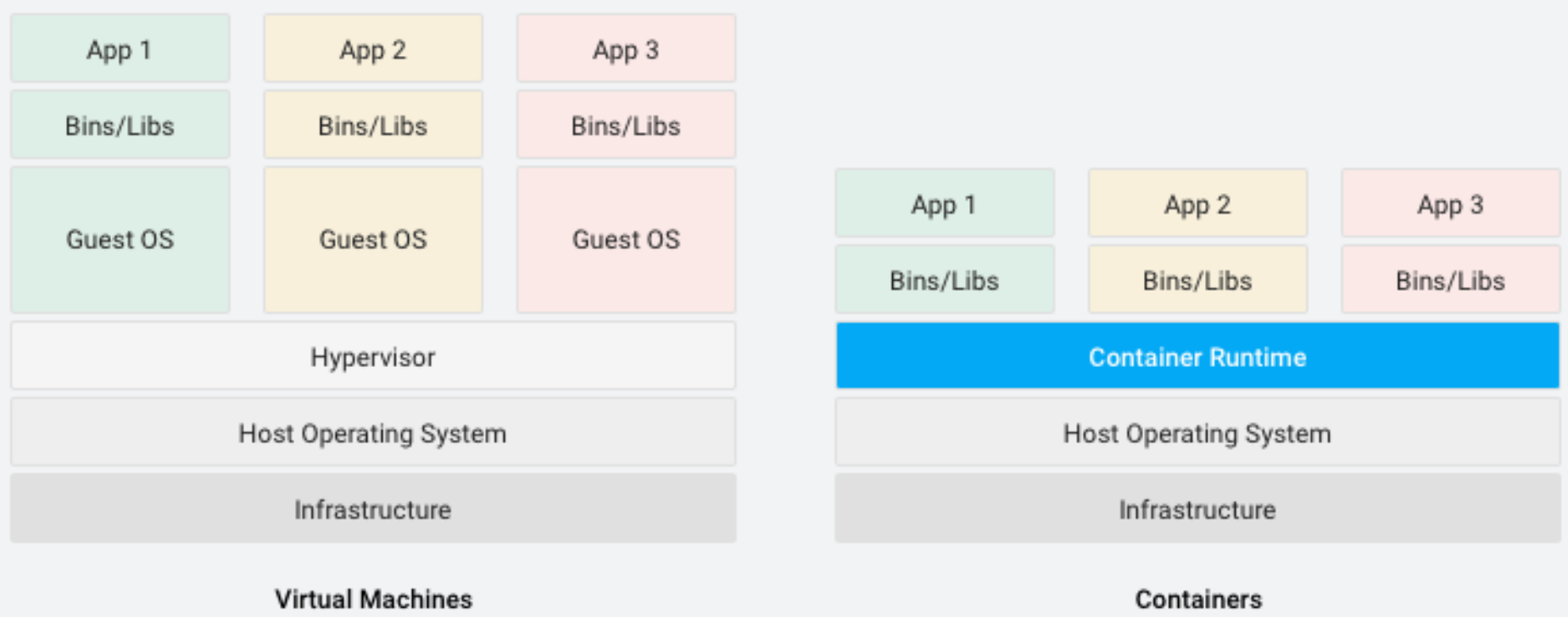
APA ITU CONTAINER?

- Menurut Docker.com, Container merupakan standar unit perangkat lunak yang mengemas kode dan semua dependensinya sehingga aplikasi berjalan dengan cepat dan andal dari satu lingkungan komputasi ke lingkungan komputasi lainnya.



Sumber gambar: https://www.docker.com/resources/what-container

PERBANDINGAN VIRTUAL MACHINE (VM) DENGAN CONTAINER (1)



Sumber gambar: https://cloud.google.com/containers/

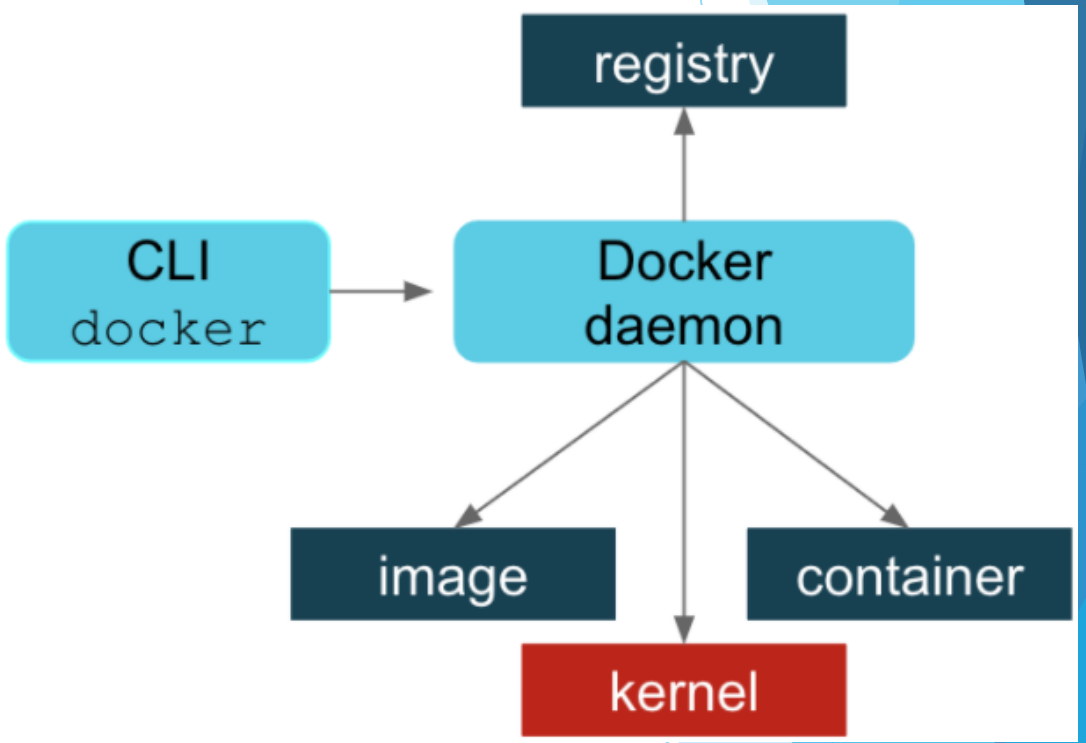
PERBANDINGAN VIRTUAL MACHINE (VM) DENGAN CONTAINER (2)

VMs	Containers
Heavyweight	Lightweight
Limited performance	Native performance
Each VM runs in its own OS	All containers share the host OS
Hardware-level virtualization	OS virtualization
Startup time in minutes	Startup time in milliseconds
Allocates required memory	Requires less memory space
Fully isolated and hence more secure	Process-level isolation, possibly less secure

Sumber: https://www.backblaze.com/blog/vm-vs-containers/

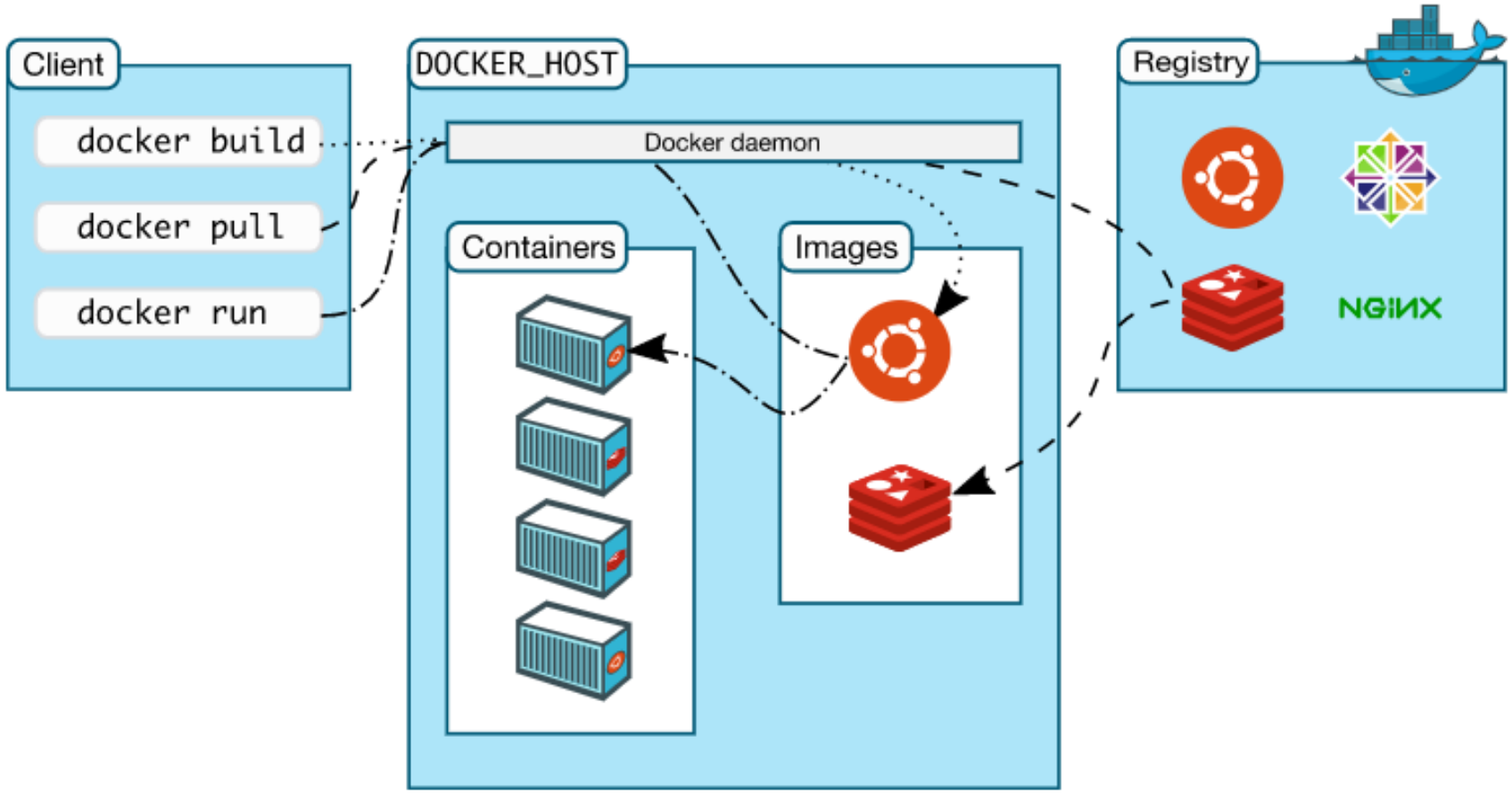
APA ITU CONTAINER IMAGES?

- *Container images* adalah cara untuk mengemas (*package*) aplikasi agar dapat dijalankan sebagai *container*.
- *Package* memuat aplikasi dan dependensi *run-time*.
- *Container images* merupakan direktori yang memuat *file* terkait metadata tentang cara menjalankan *container*.
- Docker merupakan *tool* yang paling populer untuk bekerja dengan *container images*.



Sumber gambar: developers.redhat.com

DOCKER ARCHITECTURE



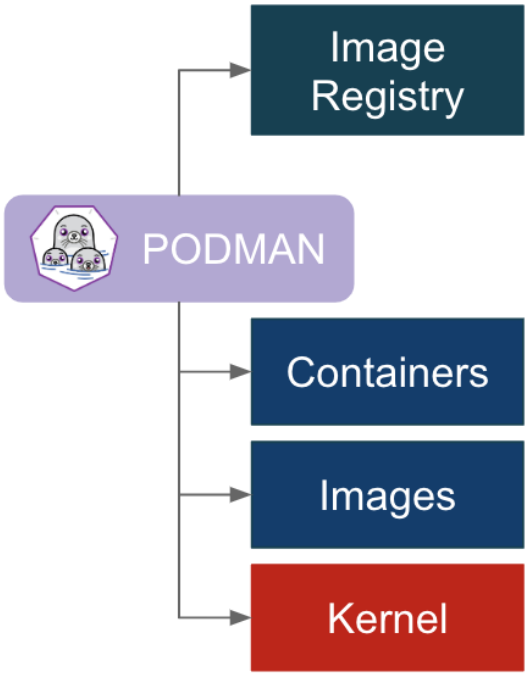
Sumber gambar: https://docs.docker.com/get-started/overview/

PERMASALAHAN PADA DOCKER

- Docker berjalan menggunakan proses tunggal sehingga dapat mengakibatkan sebuah titik kegagalan.
- Proses tersebut memiliki keseluruhan *child processes* atau *container* yang berjalan.
- Apabila terjadi kegagalan maka keseluruhan *child process* kehilangan jejaknya sehingga akan masuk ke dalam *orphaned state*.
- Membangun *container* menyebabkan kerentanan keamanan.
- Keseluruhan operasi pada *Docker* dilakukan oleh *root*.

PENGENALAN PODMAN (1)

- Merupakan daemonless container engine untuk mengembangkan, memanajemen dan menjalankan Open Container Initiative (OCI) *containers* dan *container images* pada sistem *Linux*.
- Podman berinteraksi secara langsung dengan *Image Registry*, *containers* dan *images storage*, serta *kernel Linux* melalui runC container runtime process (tidak menggunakan *daemon*).



Sumber gambar: developers.redhat.com

PENGENALAN PODMAN (2)

podman

- Podman menyediakan *command line front end* yang mendukung kompatibilitas dengan Docker. Secara sederhana dilakukan dengan membuat alias Docker CLI, `alias docker=podman`.
- *Container* yang dikelola oleh Podman dapat dijalankan oleh root atau menggunakan mode rootless (non-privileged user).
- Podman mengelola keseluruhan ekosistem *container* meliputi pods, containers, container images, dan container volumes menggunakan *library* libpod.
- Pod adalah pengelompokkan satu atau lebih *container* yang diterapkan bersama pada *host* yang sama.

INSTALASI PODMAN PADA CENTOS 8

1. Memperbaharui sistem.

   ```
   # dnf -y update
   ```

2. Instalasi Podman.

   ```
   # dnf -y install podman
   ```

3. Memverifikasi versi Podman yang terinstall.

   ```
   # podman --version
   ```

   ```
   [root@oks ~]# podman --version
   podman version 1.6.4
   ```

4. Menampilkan informasi sistem terkait Podman.

   ```
   # podman info
   ```

   Terlampir cuplikan *output* dari eksekusi perintah tersebut.

```
[root@oks ~]# podman info
host:
  BuildahVersion: 1.12.0-dev
  CgroupVersion: v1
  Conmon:
    package: conmon-2.0.6-1.module_el8.2.0+305+5e198a41.x86_64
    path: /usr/bin/conmon
    version: 'conmon version 2.0.6, commit: a2b11288060ebd7abd20e0b4eb1a834bbf0aec3e'
```

MANAJEMEN CONTAINER IMAGES PADA PODMAN (1)

- Menampilkan informasi daftar *container images* yang terdapat pada penyimpanan lokal.

  ```
  # podman images
  ```

  ```
  [root@oks ~]# podman images
  REPOSITORY   TAG   IMAGE ID   CREATED   SIZE
  [root@oks ~]#
  ```

- Melakukan pencarian *container images* pada registry.

  ```
  # podman search image
  ```

  Sebagai contoh mencari *image* hello-world:

  ```
  # podman search hello-world
  ```

MANAJEMEN CONTAINER IMAGES PADA PODMAN (2)

- Hasil dari pencarian *image* hello-world.

```
[root@oks ~]# podman search hello-world
INDEX       NAME                                          DESCRIPTION                                     STARS   OFFICIAL   AUTOMATED
docker.io   docker.io/library/hello-world                 Hello World! (an example of minimal Dockeriz...   1307    [OK]
docker.io   docker.io/tutum/hello-world                   Image to test docker deployments. Has Apache...   72                 [OK]
docker.io   docker.io/ansibleplaybookbundle/hello-world-apb   An APB which deploys a sample Hello World! a...   1                  [OK]
docker.io   docker.io/ansibleplaybookbundle/hello-world-db-apb   An APB which deploys a sample Hello World! a...   1                  [OK]
docker.io   docker.io/dockercloud/hello-world             Hello World!                                    19                 [OK]
docker.io   docker.io/kitematic/hello-world-nginx         A light-weight nginx container that demonstr...   148
docker.io   docker.io/crccheck/hello-world                Hello World web server in under 2.5 MB          13                 [OK]
docker.io   docker.io/infrastructureascode/hello-world    A tiny "Hello World" web server with a healt...   0                  [OK]
docker.io   docker.io/ppc64le/hello-world                 Hello World! (an example of minimal Dockeriz...   2
docker.io   docker.io/freddiedevops/hello-world-spring-boot                                               0
docker.io   docker.io/strimzi/hello-world-producer                                                       0
docker.io   docker.io/strimzi/hello-world-consumer                                                       0
docker.io   docker.io/strimzi/hello-world-streams                                                        0
docker.io   docker.io/datawire/hello-world                Hello World! Simple Hello World implementati...   1                  [OK]
docker.io   docker.io/vad1mo/hello-world-rest              A simple REST Service that echoes back all t...   4                  [OK]
docker.io   docker.io/koudaiii/hello-world                                                               0
docker.io   docker.io/rancher/hello-world                                                               1
docker.io   docker.io/burdz/hello-world-k8s               To provide a simple webserver that can have ...   0                  [OK]
docker.io   docker.io/businessgeeks00/hello-world-nodejs                                                 0
docker.io   docker.io/souravpatnaik/hello-world-go        hello-world in Golang                           1
docker.io   docker.io/markmnei/hello-world-java-docker    Hello-World-Java-docker                         1                  [OK]
docker.io   docker.io/kevindockercompany/hello-world                                                     0
docker.io   docker.io/carinamarina/hello-world-app        This is a sample Python web application, run...   1                  [OK]
docker.io   docker.io/nirmata/hello-world                                                               0                  [OK]
docker.io   docker.io/okteto/hello-world                                                                0
```

MANAJEMEN CONTAINER IMAGES PADA PODMAN (3)

- Secara default `podman search` akan mencari *container images* berdasarkan daftar registry yang terdapat pada bagian **`[registries.search]`** dari *file* konfigurasi **`/etc/containers/registries.conf`**.

```
# head -35 /etc/containers/registries.conf | tail -2
```

```
[root@oks ~]# head -35 /etc/containers/registries.conf | tail -2
[registries.search]
registries = ['registry.access.redhat.com', 'registry.redhat.io', 'docker.io']
```

Terlihat `podman search` akan mencari *image* yang diminta secara berurutan di:

1. registry.access.redhat.com
2. registry.redhat.io
3. docker.io

MANAJEMEN CONTAINER IMAGES PADA PODMAN (4)

- Mengunduh *images* tertentu sebagai contoh hello-world.

  `# podman pull docker.io/library/hello-world`

```
[root@oks ~]# podman pull docker.io/library/hello-world
Trying to pull docker.io/library/hello-world...
  Get https://auth.docker.io/token?scope=repository%3Alibrary%2Fhello-world%3Apull&service=registry.docker.io: dial tcp: lookup auth.docker.io on 192
.168.169.254:53: server misbehaving
Error: error pulling image "docker.io/library/hello-world": unable to pull docker.io/library/hello-world: unable to pull image: Error parsing image c
onfiguration: Get https://auth.docker.io/token?scope=repository%3Alibrary%2Fhello-world%3Apull&service=registry.docker.io: dial tcp: lookup auth.dock
er.io on 192.168.169.254:53: server misbehaving
```

  Terlihat proses unduh gagal dilakukan karena diperlukan *login* terlebih dahulu ke *registry server* tersebut.

- Login ke *registry server* menggunakan *username* dan *password* yang telah didaftarkan dengan mengeksekusi perintah `podman login registry`, sebagai contoh ke docker.io.

  `# podman login docker.io`.

```
[root@oks ~]# podman login docker.io
Username: [redacted]
Password:
Login Succeeded!
```

  Apabila proses *login* sukses dilakukan maka akan muncul pesan Login Succeeded!.

MANAJEMEN CONTAINER IMAGES PADA PODMAN (5)

- Ujicoba mengunduh kembali *image* hello-world.

```
[root@oks ~]# podman pull docker.io/library/hello-world
Trying to pull docker.io/library/hello-world...
Getting image source signatures
Copying blob 0e03bdcc26d7 done
Copying config bf756fb1ae done
Writing manifest to image destination
Storing signatures
bf756fb1ae65adf866bd8c456593cd24beb6a0a061dedf42b26a993176745f6b
```

Terlihat proses unduh telah berhasil dilakukan.

- Verifikasi *image* yang telah diunduh dengan mengeksekusi perintah:

```
# podman images
```

```
[root@oks ~]# podman images
REPOSITORY                      TAG      IMAGE ID       CREATED         SIZE
docker.io/library/hello-world   latest   bf756fb1ae65   9 months ago    20 kB
```

MANAJEMEN CONTAINER IMAGES PADA PODMAN (6)

- Mengunduh *image* nginx.

```
[root@oks ~]# podman pull nginx
Trying to pull registry.access.redhat.com/nginx...
  can't talk to a V1 docker registry
Trying to pull registry.redhat.io/nginx...
  unable to retrieve auth token: invalid username/password: unauthorized: Please login to the Red Hat Registry using your Customer Portal credentials
. Further instructions can be found here: https://access.redhat.com/RegistryAuthentication
Trying to pull docker.io/library/nginx...
Getting image source signatures
Copying blob 1a0022e444c2 done
Copying blob bb79b6b2107f done
Copying blob 111447d5894d done
Copying blob a95689b8e6cb done
Copying blob 32b7488a3833 done
Copying config f35646e839 done
Writing manifest to image destination
Storing signatures
f35646e83998b844c3f067e5a2cff84cdf0967627031aeda3042d78996b68d35
```

Terlihat proses unduh telah berhasil dilakukan.

- Verifikasi *image* yang telah diunduh dengan mengeksekusi perintah:

```
# podman images
```

```
[root@oks ~]# podman images
REPOSITORY                  TAG      IMAGE ID       CREATED        SIZE
docker.io/library/nginx     latest   f35646e83998   4 days ago     137 MB
docker.io/library/hello-world   latest   bf756fb1ae65   9 months ago   20 kB
```

MANAJEMEN CONTAINER IMAGES PADA PODMAN (7)

- Menampilkan informasi konfigurasi dari *container* atau *image* dengan mengeksekusi perintah:

  `# podman inspect name`

  Sebagai contoh untuk *image* hello-world.

  `# podman inspect hello-world`

  Terlampir cuplikan *output* dari eksekusi perintah tersebut.

```
[root@oks ~]# podman inspect hello-world
[
    {
        "Id": "bf756fb1ae65adf866bd8c456593cd24beb6a0a061dedf42b26a993176745f6b",
        "Digest": "sha256:8c5aeeb6a5f3ba4883347d3747a7249f491766ca1caa47e5da5dfcf6b9b717c0",
        "RepoTags": [
            "docker.io/library/hello-world:latest"
        ],
        "RepoDigests": [
            "docker.io/library/hello-world@sha256:8c5aeeb6a5f3ba4883347d3747a7249f491766ca1caa47e5da5dfcf6b9b717c0",
            "docker.io/library/hello-world@sha256:90659bf80b44ce6be8234e6ff90a1ac34acbeb826903b02cfa0da11c82cbc042"
        ],
```

MANAJEMEN CONTAINER PADA PODMAN (1)

- Menjalankan *container* dari *image* menggunakan perintah:

  `# podman run image`

  Sebagai contoh *image* hello-world dengan mengeksekusi perintah:

  `# podman run hello-world`

```
[root@oks ~]# podman 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/
```

MANAJEMEN CONTAINER PADA PODMAN (2)

- Menampilkan informasi *container* yang berjalan pada sistem.

```
# podman ps
```

```
[root@oks ~]# podman ps
CONTAINER ID  IMAGE  COMMAND  CREATED  STATUS  PORTS  NAMES
```

Terlihat tidak terdapat *container* yang sedang berjalan pada sistem.

- Menampilkan informasi keseluruhan *container* yang pernah dibuat oleh Podman baik yang berjalan maupun berhenti pada sistem.

```
# podman ps -a
```

```
[root@oks ~]# podman ps -a
CONTAINER ID  IMAGE                                COMMAND  CREATED         STATUS                     PORTS  NAMES
1f663d4f3810  docker.io/library/hello-world:latest  /hello   17 seconds ago  Exited (0) 17 seconds ago         goofy_hermann
```

Terlihat pernah dibuat *container* dari *image* hello-world.

MANAJEMEN CONTAINER PADA PODMAN (3)

- Menjalankan *container* di *background* dari *image* nginx dan mengekspose *container port* 80 sebagai *localhost:80*.

```
# podman run -d -p 80:80 nginx
```

```
[root@oks ~]# podman run -d -p 80:80 nginx
d14ef463b54700b87768c5778de7deb77b39bc1394da8f28e837ecdc4eb844ab
```

- Menampilkan informasi *container* yang berjalan pada sistem.

```
# podman ps
```

```
[root@oks ~]# podman ps
CONTAINER ID  IMAGE                           COMMAND               CREATED         STATUS             PORTS                   NAMES
d14ef463b547  docker.io/library/nginx:latest  nginx -g daemon o...  2 minutes ago  Up 2 minutes ago  0.0.0.0:80->80/tcp  silly_wescoff
```

Terlihat *container* dari *image* nginx sedang berjalan pada sistem.

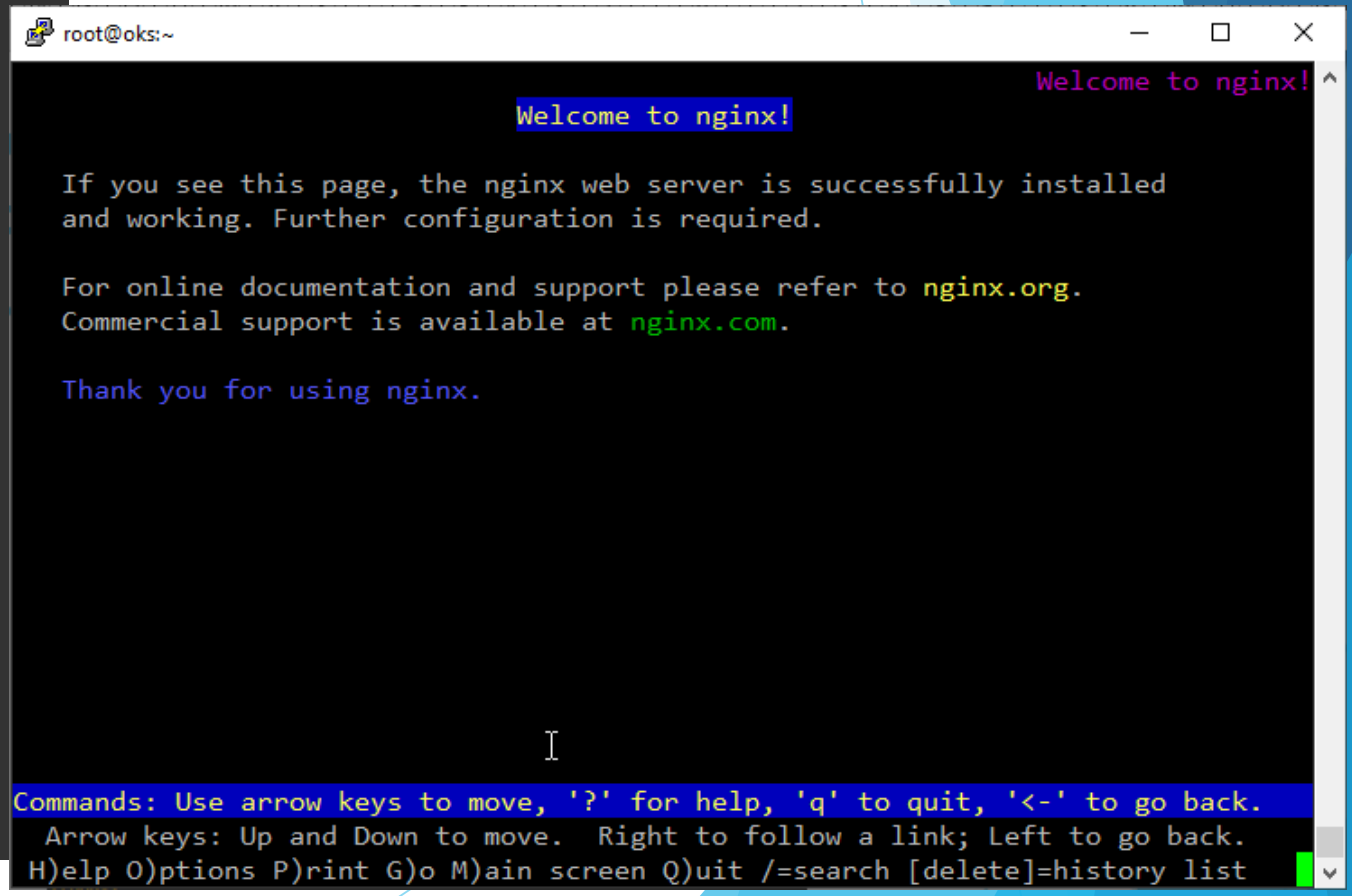
MANAJEMEN CONTAINER PADA PODMAN (4)

- Memverifikasi akses ke *server web* nginx dari localhost menggunakan utilitas curl dan lynx. Perintah yang dieksekusi adalah `curl localhost` dan `lynx localhost`.

```
[root@oks ~]# curl localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```



MANAJEMEN CONTAINER PADA PODMAN (5)

- Menghentikan *container* yang sedang berjalan menggunakan perintah:

  ```
  # podman stop container
  ```

  Container dapat berupa ID atau *name*. Sebagai contoh untuk menghentikan container dengan *image* nginx yang memiliki ID d14ef463b547 maka perintah yang dieksekusi adalah:

  ```
  # podman stop d14ef463b547
  ```

```
[root@oks ~]# podman stop d14ef463b547
d14ef463b54700b87768c5778de7deb77b39bc1394da8f28e837ecdc4eb844ab
```

- Menampilkan informasi *container* yang berjalan pada sistem.

  ```
  # podman ps
  ```

```
[root@oks ~]# podman ps
CONTAINER ID  IMAGE  COMMAND  CREATED  STATUS  PORTS  NAMES
```

Terlihat tidak terdapat *container* yang sedang berjalan pada sistem.

MANAJEMEN CONTAINER PADA PODMAN (6)

- Menampilkan informasi keseluruhan *container* yang pernah dibuat oleh Podman baik yang berjalan maupun berhenti pada sistem.

```
# podman ps -a
```

```
[root@oks ~]# podman ps -a
CONTAINER ID  IMAGE                                COMMAND               CREATED             STATUS                         PORTS                   NAMES
d14ef463b547  docker.io/library/nginx:latest       nginx -g daemon o...  23 minutes ago      Exited (0) 8 seconds ago       0.0.0.0:80->80/tcp      silly_
wescoff
1f663d4f3810  docker.io/library/hello-world:latest  /hello                About an hour ago   Exited (0) About an hour ago                           goofy_
hermann
```

Terlihat terdapat 2 (dua) *container* yang pernah dibuat pada sistem.

- Menghapus seluruh *container* dari host dapat dilakukan dengan mengeksekusi perintah:

```
# podman rm -a
```

```
[root@oks ~]# podman rm -a
1f663d4f3810036eb6d3341e40ac24196a0a591fe846bfb40fa9fc02a547a2e7
d14ef463b54700b87768c5778de7deb77b39bc1394da8f28e837ecdc4eb844ab
```

MANAJEMEN CONTAINER PADA PODMAN (7)

- Verifikasi dengan menampilkan informasi keseluruhan *container* yang pernah dibuat oleh Podman baik yang berjalan maupun berhenti pada sistem.

```
# podman ps -a
```

```
[root@oks ~]# podman ps -a
CONTAINER ID  IMAGE  COMMAND  CREATED  STATUS  PORTS  NAMES
```

Terlihat keseluruhan *container* telah berhasil dihapus.

MANAJEMEN CONTAINER IMAGES (7)

- Menampilkan informasi *local images*.

```
# podman images
```

```
[root@oks ~]# podman images
REPOSITORY                    TAG      IMAGE ID       CREATED        SIZE
docker.io/library/nginx       latest   f35646e83998   4 days ago     137 MB
docker.io/library/hello-world latest   bf756fb1ae65   9 months ago   20 kB
```

- Menghapus *local container image* dari *local cache*.

```
# podman rmi image
```

Sebagai contoh *image* nginx.

```
# podman rmi nginx
```

```
[root@oks ~]# podman rmi nginx
Untagged: docker.io/library/nginx:latest
Deleted: f35646e83998b844c3f067e5a2cff84cdf0967627031aeda3042d78996b68d35
```

- Verifikasi penghapusan *image* nginx dengan mengeksekusi perintah:

```
# podman images
```

```
[root@oks ~]# podman images
REPOSITORY                    TAG      IMAGE ID       CREATED        SIZE
docker.io/library/hello-world latest   bf756fb1ae65   9 months ago   20 kB
```

MANAJEMEN CONTAINER IMAGES (8)

- Keluar atau *logout* dari container registry dengan mengeksekusi perintah:

  `# podman logout registry`

  Sebagai contoh untuk *registry server* docker.io.

  `# podman logout docker.io`

```
[root@oks ~]# podman logout docker.io
Removed login credentials for docker.io
```

ROOTLESS PODMAN (1)

- Membuat *user* baru pada *CentOS 8* dengan nama login "oks" dan *password* "12345678".

```
# useradd oks

# passwd oks
```

```
[root@oks ~]# useradd oks
[root@oks ~]# passwd oks
Changing password for user oks.
New password:
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic
Retype new password:
passwd: all authentication tokens updated successfully.
```

- *Logout* dari *user* root dengan mengeksekusi perintah `logout` dan *login* ke *CentOS 8* sebagai user "oks".

```
oks@oks:~
login as: oks
oks@192.168.169.170's password:
Web console: https://oks.ubg.local:9090/ or https://192.168.169.170:9090/

[oks@oks ~]$
```

ROOTLESS PODMAN (2)

- Login ke *registry server* docker.io menggunakan *username* dan *password* yang telah didaftarkan dengan mengeksekusi perintah:

```
$ podman login docker.io.
```

```
[oks@oks ~]$ podman login docker.io
Username:
Password:
Login Succeeded!
```

Apabila proses *login* sukses dilakukan maka akan muncul pesan Login Succeeded!.

- Mengunduh *image* httpd-24-centos7 dari docker.io.

```
# podman pull centos/httpd-24-centos7
```

```
[oks@oks ~]$ podman pull centos/httpd-24-centos7
Trying to pull registry.access.redhat.com/centos/httpd-24-centos7...
  name unknown: Repo not found
Trying to pull registry.redhat.io/centos/httpd-24-centos7...
  unable to retrieve auth token: invalid username/password: unauthorized: Please
 login to the Red Hat Registry using your Customer Portal credentials. Further i
nstructions can be found here: https://access.redhat.com/RegistryAuthentication
Trying to pull docker.io/centos/httpd-24-centos7...
Getting image source signatures
Copying blob e2c4942f4189 done
Copying blob f1498894b11c done
Copying blob da56c9694723 done
Copying blob 75f829a71a1c done
Copying blob 063ff8a1435c done
Copying blob 09351523c1de done
Copying blob 17f560ff6bfd done
Copying blob 9a03c4ed0deb done
Copying config 24aec7b98d done
Writing manifest to image destination
Storing signatures
24aec7b98d6bc27d1a9af66958328cf4193c6f29b770b3005da1ae00baa13e61
```

ROOTLESS PODMAN (3)

- Menampilkan informasi *local images*.

```
$ podman images
```

```
[oks@oks ~]$ podman images
REPOSITORY                          TAG      IMAGE ID       CREATED       SIZE
docker.io/centos/httpd-24-centos7   latest   24aec7b98d6b   3 weeks ago   357 MB
```

Terlihat *image* httpd-24-centos7 telah berhasil diunduh.

- Membuat direktori webcontent/html pada *home* direktori dari *user* yang digunakan sebagai lokasi penyimpanan konten dari *website* dan memverifikasi hasil pembuatan direktori tersebut menggunakan utilitas `tree`.

```
# mkdir -p webcontent/html

# tree webcontent
```

```
[oks@oks ~]$ mkdir -p webcontent/html
[oks@oks ~]$ tree webcontent
webcontent
└── html

1 directory, 0 files
```

Terlihat direktori webcontent/html telah berhasil dibuat.

ROOTLESS PODMAN (4)

- Membuat file index.html dan disimpan pada direktori webcontent/html serta menambahkan konten “Selamat Datang di Situs Universitas Bumigora” di dalam *file* tersebut.

```
# echo “Selamat Datang di Situs Universitas Bumigora” >
webcontent/html/index.html
```

```
[oks@oks ~]$ echo "Selamat Datang di Situs Universitas Bumigora" > webcontent/html/index.html
```

- Memverifikasi konten dari file index.html yang telah dibuat.

```
# cat webcontent/html/index.html
```

```
[oks@oks ~]$ cat webcontent/html/index.html
Selamat Datang di Situs Universitas Bumigora
```

PERSISTENT STORAGE PADA CONTAINER (1)

- Menjalankan *container* di *background* menggunakan *image* httpd-24-centos7 dengan nama webserver dan mengekspose container port 8080 sebagai localhost:8080. Selain itu melakukan mount direktori ~/webcontent pada *host* ke direktori /var/www pada *container*.

```
$ podman run –d --name webserver –p 8080:8080 –v
~/webcontent:/var/www:Z httpd-24-centos7
```

Penjelasan *option*:

-d digunakan untuk menjalankan *container* di *background* (*detached mode*).

--name digunakan untuk mengatur nama *container*.

-p digunakan untuk mempublikasikan *container port* 8080 yang terbuka ke *port 8080* pada *interface host*.

-v digunakan untuk membuat *bind mount*.

:Z pada *volume mount* digunakan untuk melabel ulang direktori dan konten didalamnya.

PERSISTENT STORAGE PADA CONTAINER (2)

- Menampilkan informasi *container* yang sedang berjalan.

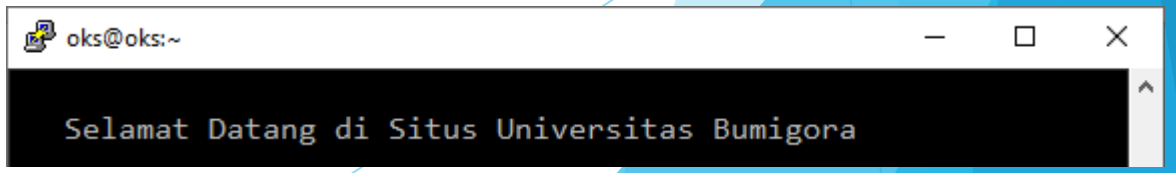
  `$ podman ps`

```
[oks@oks user]$ podman ps
CONTAINER ID  IMAGE                                COMMAND               CREATED         STATUS             PORTS                   NAMES
4f08a37257eb  docker.io/centos/httpd-24-centos7:latest  /usr/bin/run-http...  5 seconds ago  Up 3 seconds ago  0.0.0.0:8080->8080/tcp  webserver
```

  Terlihat *container* dengan nama webserver telah berhasil dibuat dan dijalankan.

- Memverifikasi akses ke *container* webserver menggunakan utilitas curl dan lynx. Perintah yang dieksekusi adalah `curl localhost:8080` dan `lynx localhost:8080`.

```
[oks@oks ~]$ curl localhost:8080
Selamat Datang di Situs Universitas Bumigora
```



MANAJEMEN CONTAINER SEBAGAI SERVICE (1)

- Membuat unit *file* systemd untuk mengatur *container* webserver dengan perintah systemctl.
- Membuat direktori `~/.config/systemd/user`.

```
$ mkdir -p ~/.config/systemd/user
```

- Memverifikasi hasil dari pembuatan direktori tersebut menggunakan utilitas tree.

```
$ tree .config
```

```
[oks@oks ~]$ tree .config
.config
├── containers
│   └── storage.conf
└── systemd
    └── user

3 directories, 1 file
```

MANAJEMEN CONTAINER SEBAGAI SERVICE (2)

- Berpindah direktori ke `~/.config/systemd/user`

```
$ cd ~/.config/systemd/user
```

- Mengeksekusi perintah `podman generate systemd` untuk membuat unit *file* bagi *container* webserver.

```
$ podman generate systemd --name webserver –files
```

```
[oks@oks user]$ podman generate systemd --name webserver --files
/home/oks/.config/systemd/user/container-webserver.service
```

- Menampilkan isi dari *file* container-webserver.service

```
$ cat container-webserver.service
```

MANAJEMEN CONTAINER SEBAGAI SERVICE (3)

```
[oks@oks user]$ cat container-webserver.service
# container-webserver.service
# autogenerated by Podman 1.6.4
# Tue Oct 20 05:46:09 WITA 2020

[Unit]
Description=Podman container-webserver.service
Documentation=man:podman-generate-systemd(1)

[Service]
Restart=on-failure
ExecStart=/usr/bin/podman start webserver
ExecStop=/usr/bin/podman stop -t 10 webserver
KillMode=none
Type=forking
PIDFile=/run/user/1000/overlay-containers/a7347f10b0f140eb462920928e733f6dab68c94f7da81d567ff9de8254aeffe7/userdata/conmon.pid

[Install]
WantedBy=multi-user.target
```

Lakukan penyesuaian isi dari *file* container-webserver.service agar *service* terkait *container* webserver tersebut dapat beroperasi dengan baik.

```
$ nano container-webserver.service
```

MANAJEMEN CONTAINER SEBAGAI SERVICE (4)

```
# container-webserver.service
# autogenerated by Podman 1.6.4
# Sun Oct 20 05:46:09 WITA 2020

[Unit]
Description=Podman container-webserver.service
Documentation=man:podman-generate-systemd(1)
Wants=network.target
After=network-online.target

[Service]
Restart=on-failure
ExecStartPre=/bin/rm -f %t/container-webserver.pid %t/container-webserver.ctr-id
ExecStart=/usr/bin/podman run --conmon-pidfile %t/container-webserver.pid --cidfile %t/container-webserver.ctr-id -d --name webserver -p 8080:8080 -v
 /home/oks/webcontent:/var/www:Z httpd-24-centos7
ExecStop=/usr/bin/sh -c "/usr/bin/podman rm -f `cat %t/container-webserver.ctr-id`"
PIDFile=%t/container-webserver.pid
KillMode=none
Type=forking

[Install]
WantedBy=multi-user.target default.target
```

Simpan perubahan dengan menekan tombol CTRL+O dan tekan Enter.

Keluar dari editor nano dengan menekan tombol CTRL+X.

MANAJEMEN CONTAINER SEBAGAI SERVICE (5)

- Menghentikan *container* webserver.

```
$ podman stop webserver
```

```
[oks@oks user]$ podman stop webserver
4f08a37257eb3d78a4a092d19adf8acf228971c41fc0ea7b6cefdbf477d3edf9
```

- Menghapus *container* webserver.

```
$ podman rm webserver
```

```
[oks@oks user]$ podman rm webserver
4f08a37257eb3d78a4a092d19adf8acf228971c41fc0ea7b6cefdbf477d3edf9
```

- Melakukan reload konfigurasi dari systemd

```
$ systemctl –user daemon-reload
```

- Mengaktifkan *service* container-webserver.

```
$ systemctl --user enable container-webserver.service
```

```
[oks@oks user]$ systemctl --user enable container-webserver.service
Created symlink /home/oks/.config/systemd/user/multi-user.target.wants/container
-webserver.service → /home/oks/.config/systemd/user/container-webserver.service.
```

MANAJEMEN CONTAINER SEBAGAI SERVICE (6)

- Menjalankan *service* container-webserver.

```
$ systemctl --user start container-webserver
```

- Menampilkan informasi *container* yang sedang berjalan.

```
$ podman ps
```

```
[oks@oks user]$ podman ps
CONTAINER ID  IMAGE                                COMMAND               CREATED        STATUS            PORTS                   NAMES
83015f21ca8b  docker.io/centos/httpd-24-centos7:latest  /usr/bin/run-http...  5 seconds ago  Up 5 seconds ago  0.0.0.0:8080->8080/tcp  webserver
```

- Menghentikan *service* container-webserver.

```
$ systemctl --user stop container-webserver
```

- Menampilkan informasi keseluruhan *container* yang pernah dibuat oleh Podman baik yang berjalan maupun berhenti pada sistem.

```
$ podman ps -a
```

```
[oks@oks user]$ podman ps -a
CONTAINER ID  IMAGE   COMMAND  CREATED  STATUS  PORTS   NAMES
[oks@oks user]$
```

MANAJEMEN CONTAINER SEBAGAI SERVICE (7)

- Memastikan *service* dari *user* oks berjalan ketika *server* diaktifkan dan berhenti ketika *server* di shutdown.

```
$ loginctl enable-linger
```

- Memverifikasi status dari *option* Linger untuk *user* oks telah diatur.

```
$ loginctl show-user oks
```

```
[oks@oks user]$ loginctl show-user oks
UID=1000
GID=1000
Name=oks
Timestamp=Sun 2020-10-18 18:46:49 WITA
TimestampMonotonic=85760097
RuntimePath=/run/user/1000
Service=user@1000.service
Slice=user-1000.slice
Display=5
State=active
Sessions=5 1
IdleHint=no
IdleSinceHint=0
IdleSinceHintMonotonic=0
Linger=yes
```

MANAJEMEN CONTAINER SEBAGAI SERVICE (8)

- Berpindah sebagai user root menggunakan perintah:

```
$ su –
```

```
[oks@oks user]$ su -
Password:
Last login: Sun Oct 18 19:15:06 WITA 2020 from 192.168.169.200 on pts/0
[root@oks ~]#
```

- Lakukan *reboot* sistem untuk menguji apakah *service* container-webserver dari *user* oks akan secara otomatis diaktifkan oleh systemd dengan mengeksekusi perintah:

```
# reboot
```

Tunggu hingga proses *reboot* selesai dilakukan.

MANAJEMEN CONTAINER SEBAGAI SERVICE (9)

- Lakukan *login* kembali ke sistem menggunakan *user* oks.

```
oks@oks:~
login as: oks
oks@192.168.169.170's password:
Web console: https://oks.ubg.local:9090/ or https://192.168.169.170:9090/

Last login: Sun Oct 18 19:25:35 2020 from 192.168.169.200
[oks@oks ~]$
```

- Verifikasi apakah *container* webserver telah berjalan.

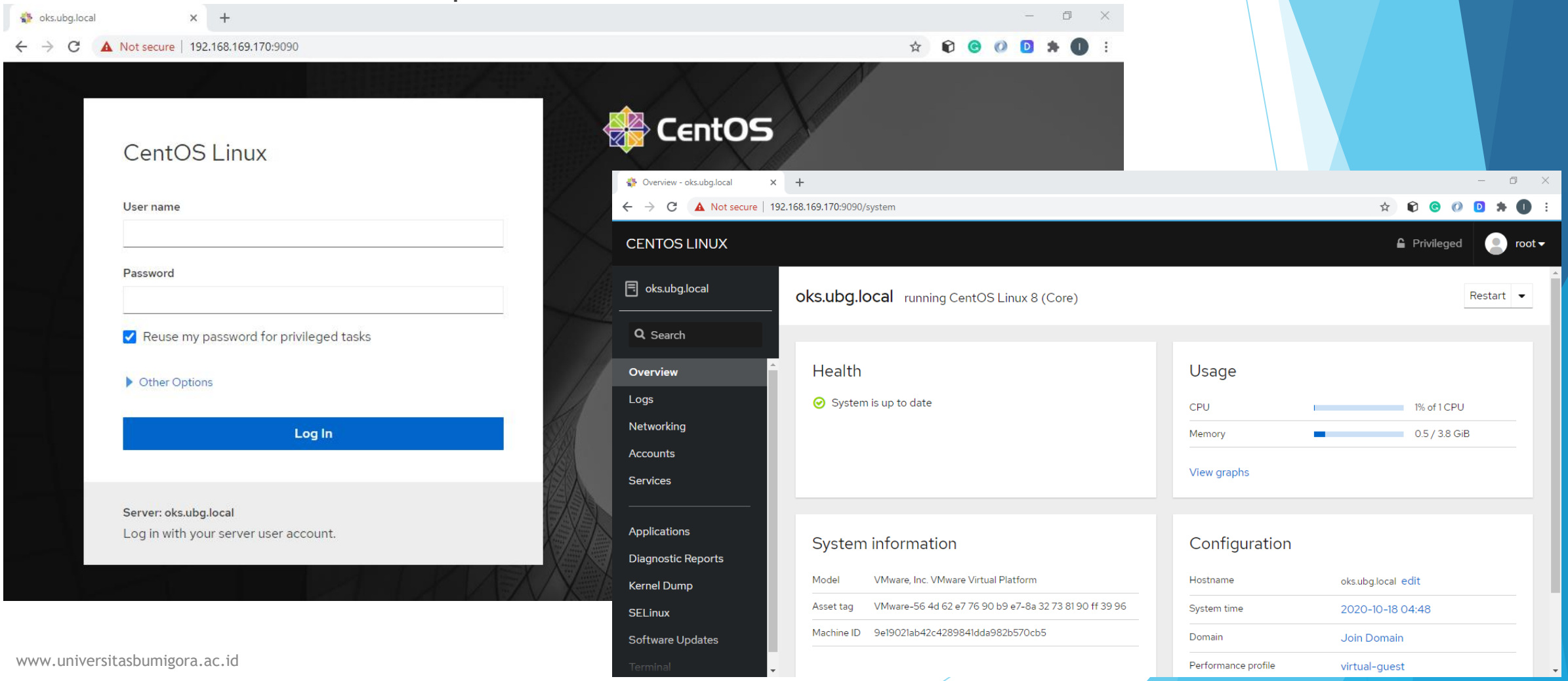
```
$ podman ps
```

```
[oks@oks ~]$ podman ps
CONTAINER ID  IMAGE                                COMMAND               CREATED        STATUS            PORTS                   NAMES
eb675c460876  docker.io/centos/httpd-24-centos7:latest  /usr/bin/run-http...  2 minutes ago  Up 2 minutes ago  0.0.0.0:8080->8080/tcp  webserver
```

Terlihat *container* webserver telah berjalan.

PENGENALAN COCKPIT

- Antarmuka berbasis web untuk manajemen dan pengawasan (*monitoring*) *server Linux* baik lokal maupun *remote*.



FITUR ADMINISTRASI COCKPIT

- Manajemen *Services*.
- Manajemen Akun Pengguna.
- Manajemen dan pengawasan *system service*.
- Konfigurasi *interface* jaringan dan *firewall*.
- Tinjauan *system logs*.
- Manajemen *Virtual Machines*.
- Pembuatan *Diagnostic Reports*.
- Konfigurasi *Kernel Dump*.
- Konfigurasi *SELinux*.
- Pembaharuan perangkat lunak (*Updating software*).
- Terminal berbasis web.

SISTEM OPERASI YANG MENDUKUNG COCKPIT

	Supported	Tested	Included
fedora	✓	✓	✓
Red Hat Enterprise Linux	✓	✓	✓
fedora COREOS	✓	✓	✓
PROJECT ATOMIC	✓	✓	✓
CentOS	✓	✓	✓
debian	✓	✓	✓
ubuntu	✓	✓	✓
Clear Linux			✓
archlinux			✓
Tumbleweed			✓

INSTALASI DAN KONFIGURASI COCKPIT PADA CENTOS 8

- Instalasi *Cockpit*.

```
# dnf install -y cockpit
```

- Mengaktifkan *service Cockpit* secara permanen.

```
# systemctl enable cockpit.socket
```

- Menjalankan *service Cockpit*.

```
# systemctl start cockpit
```

- Memverifikasi status *service Cockpit*.

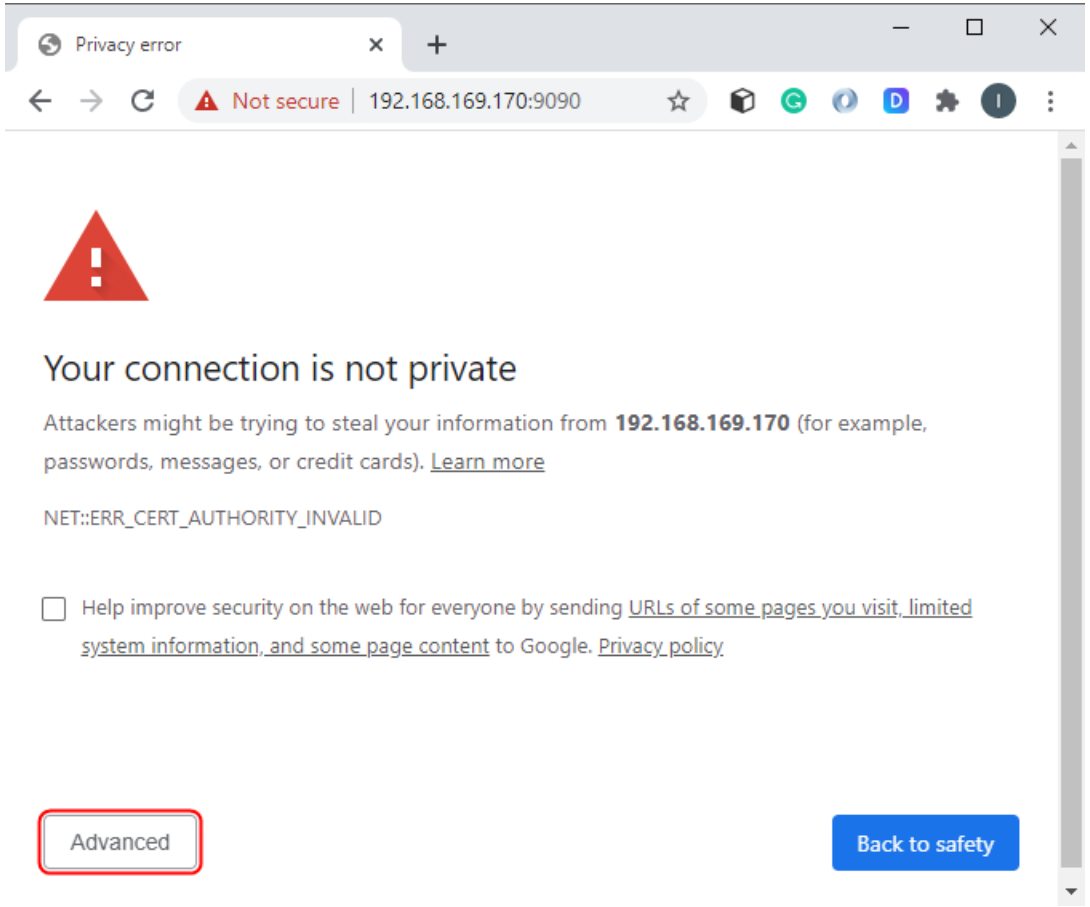
```
# systemctl status cockpit
```

- Menambahkan aturan **firewall** untuk mengijinkan akses ke **Dashboard** *Cockpit* dari mesin lainnya dan manajemen *remote machine*.

```
# firewall-cmd --permanent --add-service=cockpit
# firewall-cmd --reload
```

VERIFIKASI AKSES COCKPIT (1)

- Buka browser dan pada *address bar* masukkan alamat https://alamat.ip.server.centos:9090 sebagai contoh **https://192.168.169.170:9090**

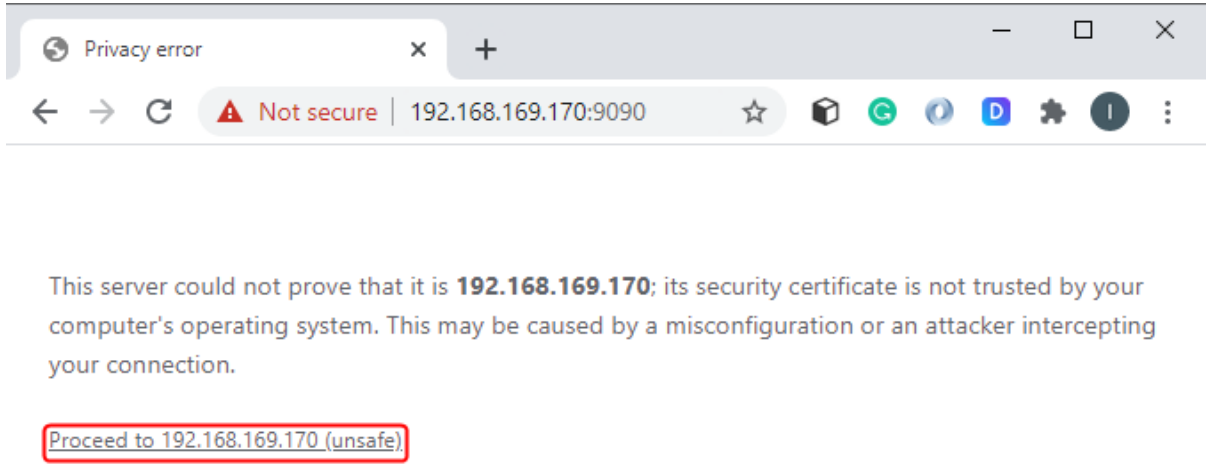


Tampil pesan peringatan "Your connection is not private".

Klik Advanced untuk melanjutkan pengaksesan.

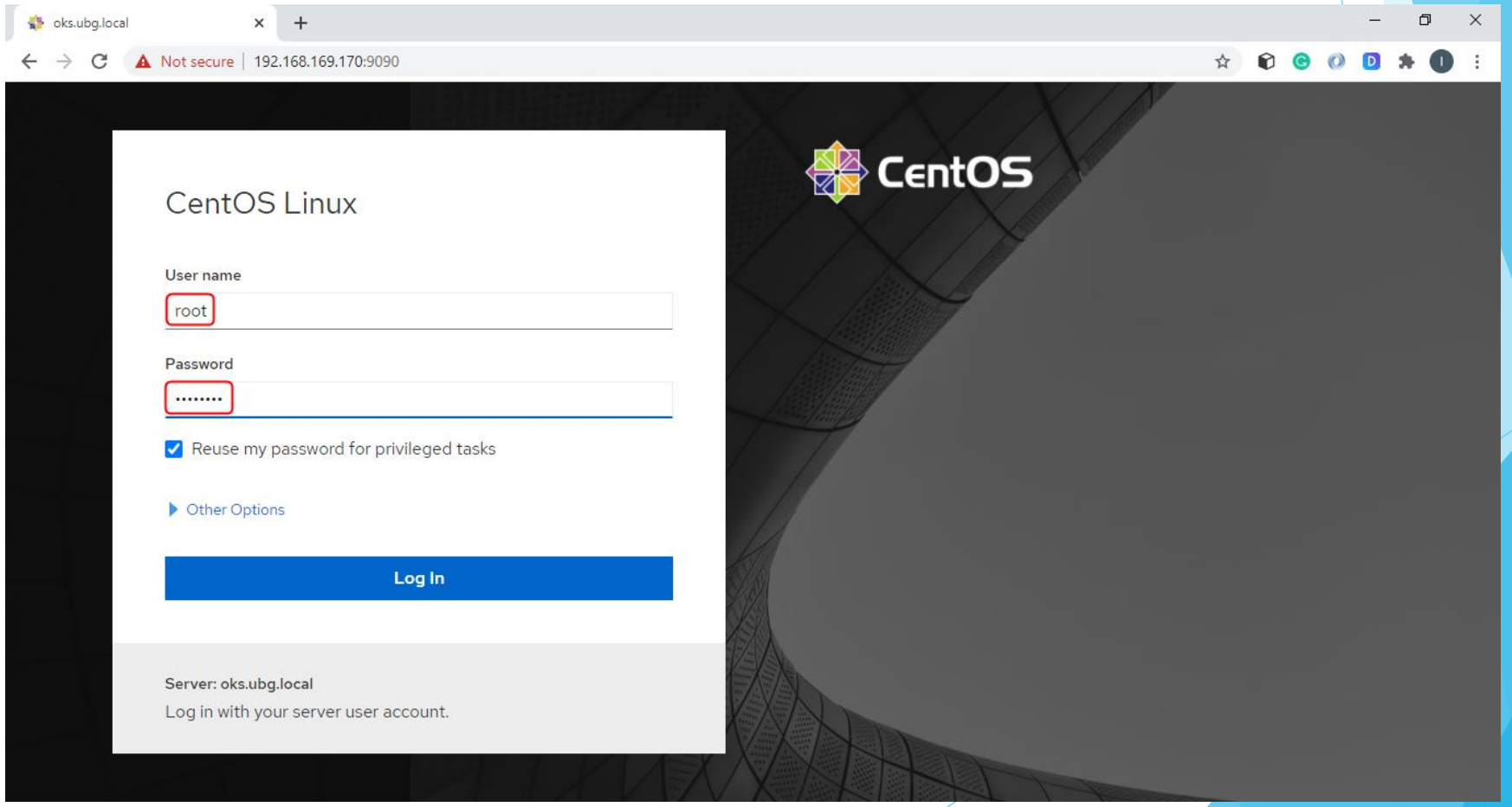
VERIFIKASI AKSES COCKPIT (2)

- Klik pada *link* "Proceed to 192.168.169.170 (unsafe)" untuk melanjutkan pemrosesan.

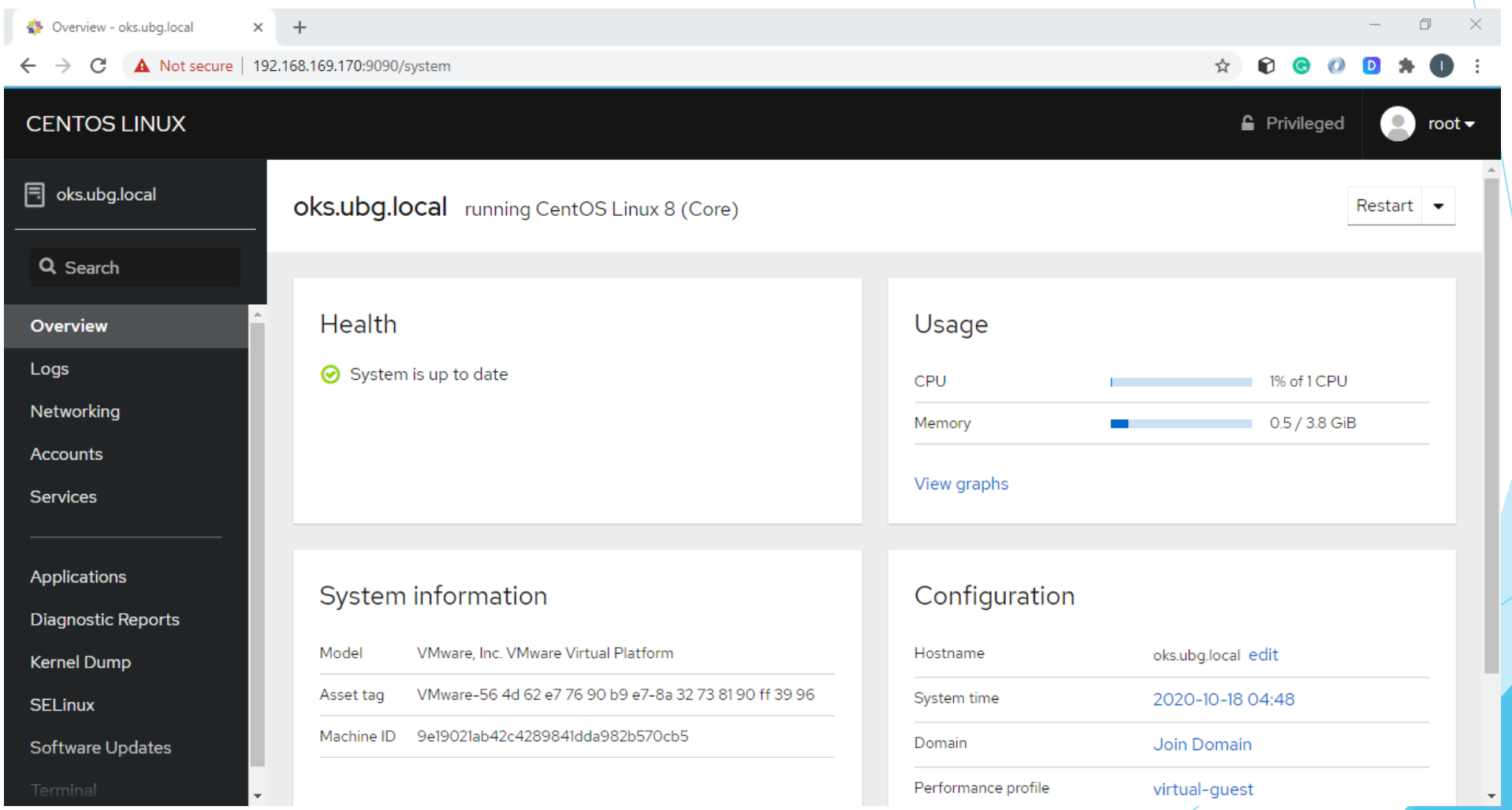


VERIFIKASI AKSES COCKPIT (3)

- *Login* menggunakan akun pengguna sistem lokal.
- Sebagai contoh menggunakan **User name root**. Lengkapi isian **Password** dari *user root* dan klik tombol **Log In**.

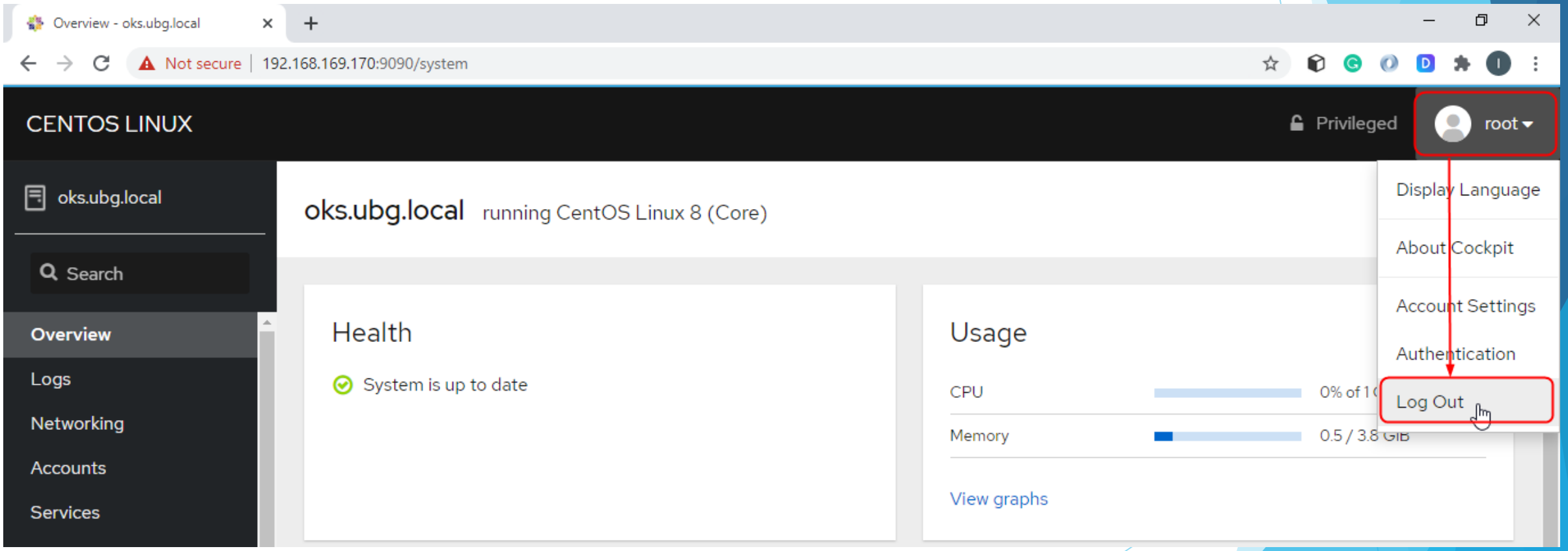


DASHBOARD DARI COCKPIT



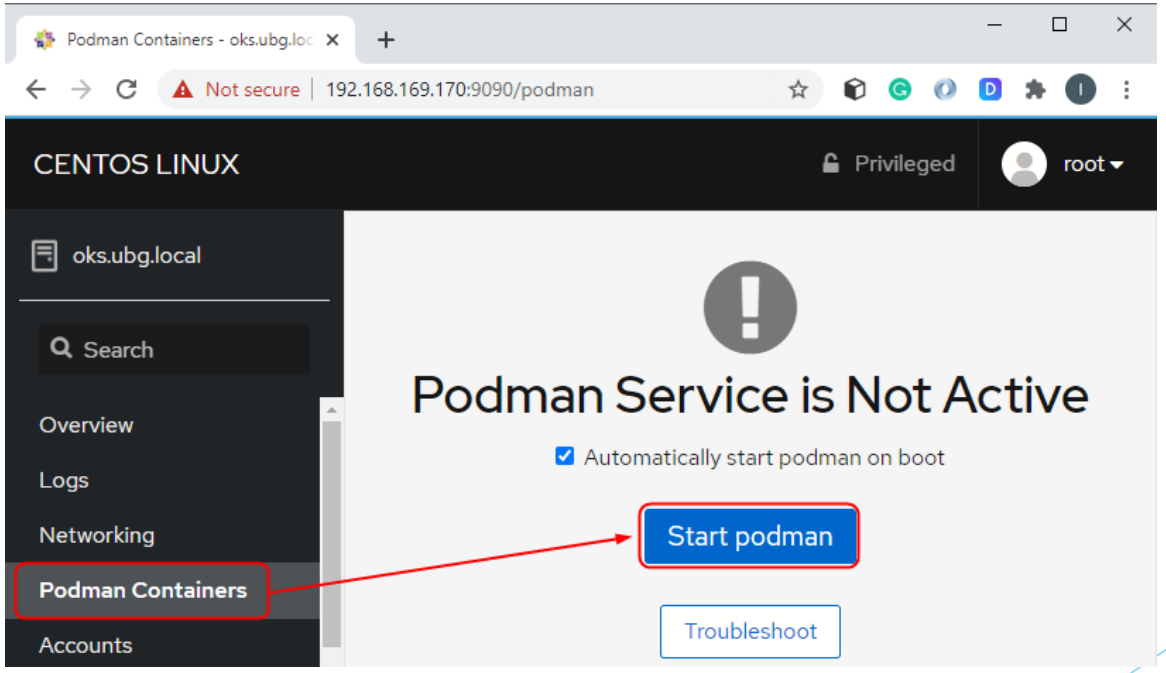
LOGOUT DARI COCKPIT

- Klik pada menu *dropdown* di pojok kanan atas dan pilih Log Out.



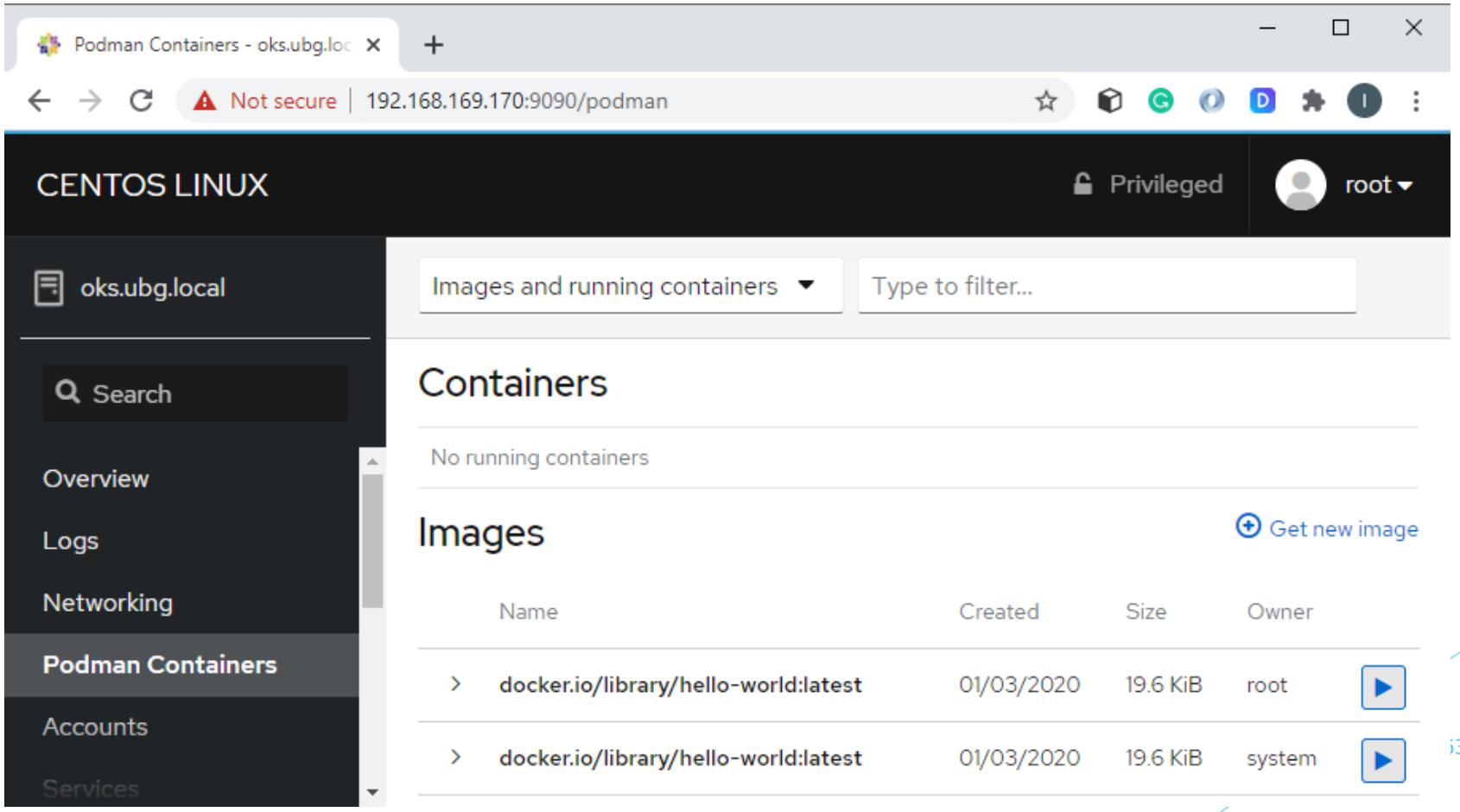
ADD-ONS PODMAN PADA COCKPIT

1. Menginstalasi dukungan Podman pada *Cockpit*.
   ```
   # dnf install -y cockpit-podman
   ```
2. *Login* ke *Cockpit* sebagai *user* root. Tampil Dashboard dari *Cockpit*. Pada sidebar sebelah kiri terlihat menu Podman Containers.
3. Klik pada menu Podman Containers maka pada panel detail sebelah kanan memperlihatkan pesan "Podman Service is Not Active" yang menginformasikan bahwa layanan Podman belum aktif. Klik pada tombol Start podman untuk mengaktifkan.



ADD-ONS PODMAN PADA COCKPIT

- Podman Containers akan memperlihatkan daftar **Containers** dan **Images** yang terdapat pada sistem.



DEMO MANAJEMEN PODMAN MENGGUNAKAN COCKPIT

ADA PERTANYAAN?

REFERENSI

- Docker, What is a Container?, 2020, https://www.docker.com/resources/what-container
- Google, Containers at Google, 2020, https://cloud.google.com/containers/
- Backblaze, What's the Diff: VMs vs Containers, 2018, https://www.backblaze.com/blog/vm-vs-containers/
- Build Containers the Hard Way (WIP), 2020, https://containers.gitbook.io/build-containers-the-hard-way/
- Docker, Docker Overview, 2020, https://docs.docker.com/get-started/overview/
- Podman Get Started, 2020, https://podman.io/getting-started/
- Podman and Buildah for Docker Users, 2019, https://developers.redhat.com/blog/2019/02/21/podman-and-buildah-for-docker-users/

REFERENSI

- Chetansingh Rajput, Docker Vs Podman, 2020, https://medium.com/technopanti/docker-vs-podman-c03359fabf77
- Red Hat, Red Hat Enterprise Linux 8.2 RHCSA650E RHCSA Running Containers Edition 1, 2020

TERIMAKASIH