

WARNING!

**DILARANG KERAS MELAKUKAN PEMBAJAKAN MAUPUN MENGAKUI ISI
MODUL INI. SELURUH ISI MODUL INI ADALAH HAK CIPTA DARI CILSY
(WWW.CILSY.ID).**

**SILAHKAN DISEBARKAN LUASKAN KEMBALI TANPA MENGUBAH APAPUN ISI
DARI MODUL INI.**



DASAR MANAJEMEN DOCKER DENGAN PORTAINER



FEBRUARY 10, 2017
CILSY FIOLUTION INDONESIA

BAB I

Pengenalan Docker

Docker adalah sebuah project open source yang ditujukan untuk developer atau sysadmin untuk membangun, mengemas dan menjalankan aplikasi dimana pun di dalam sebuah container.

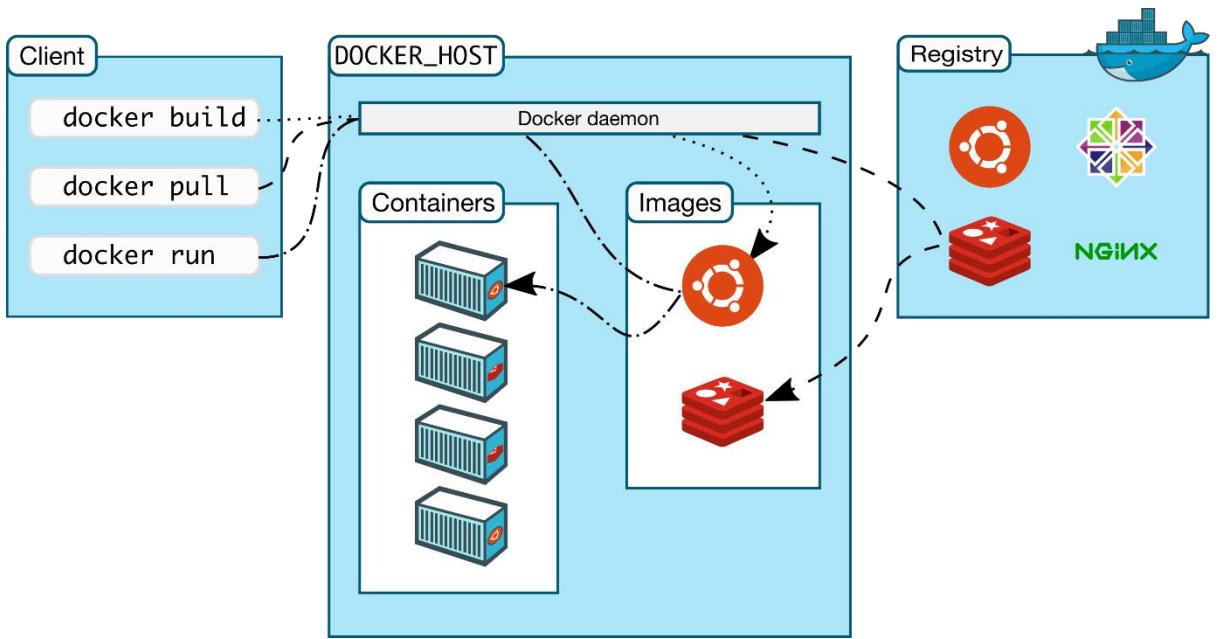
Mungkin anda sedikit bingung dengan pengertian diatas dikarenakan terlalu sulit untuk membayangkan bagaimana pengembangan aplikasi yang sebenarnya. Docker berfungsi sebagai virtualisasi sebuah sistem operasi atau sebuah server atau sebuah web server atau bahkan sebuah database server, dimana dengan menggunakan virtualisasi ini, diharapkan developer dapat mengembangkan aplikasi sesuai dengan spesifikasi server atau dengan kata lain, jika kita mengembangkan sebuah aplikasi lalu kita jalankan pada komputer kita sendiri maka secara otomatis aplikasi akan berjalan dengan baik, nah bagaimana jika server yang akan menjalankan aplikasi kita memiliki banyak perbedaan dengan komputer kita seperti perbedaan sistem operasi, arsitektur processor dan sebagainya. Dengan menggunakan virtualisasi ini maka para developer lebih mudah untuk mengatur mengenai deployment atau menjalankan aplikasi di server production.

Sebelum kita membahas mengenai docker lebih lanjut, kita akan mencoba membahas sedikit mengenai docker dan vagrant. Docker dan vagrant merupakan tool yang sama atau dapat dikatakan merupakan tool developer yang mempunyai fungsi yang sama, akan tetapi meski memiliki fungsi yang sama terdapat beberapa perbedaan sehingga kita perlu menentukan tool yang terbaik untuk melakukan development sebuah aplikasi. Beberapa perbedaan dapat dilihat melalui tabel berikut.

	Docker	Vagrant
Virtualization	Linux container	Virtual machine
Resource isolation	Weak	Strong
OS	Linux	Linux, Windows, MacOS, ...
Starting time	Seconds	Minutes
Size	100M+	1G+
CM integration	No	Yes
Building image time	Short (mins)	Long (10+ mins)
Hosted number	>50	<10
Deployment tools	CoreOS, Mesos,...	Terraform
Public images	Yes (Docker Hub)	Yes (Vagrant Cloud)

gambar diatas dapat anda lihat [disini](#). Dari gambar dapat dilihat bahwa terdapat banyak sekali perbedaan antara docker dan vagrant. Perbedaan yang sangat mencolok adalah docker menggunakan resource atau memory yang lebih sedikit ketimbang vagrant, ini dapat dilihat dari penggunaan RAM, penggunaan images sistem operasi dan juga dapat dilihat perbedaannya, jika menggunakan vagrant maka kita wajib melakukan instalasi virtual machine seperti virtual box atau vmware, berbeda dengan docker menggunakan linux container sehingga kita tidak perlu melakukan instalasi virtual machine.

Arsitektur Docker



gambar diatas merupakan arsitektur docker, dimana docker terdiri dari beberapa element yaitu docker client, docker daemon, docker container, docker images dan docker registry. Docker menggunakan teknologi client server untuk menghubungkan antara docker client dan docker daemon. Penulis akan menjelaskan sedikit mengenai istilah - istilah penting pada docker.

Docker Daemon

Docker daemon berfungsi untuk membangun, mendistribusikan dan menjalankan container docker. User tidak dapat langsung menggunakan docker daemon, akan tetapi untuk menggunakan docker daemon maka user menggunakan docker client sebagai perantara atau cli.

Docker Images

Docker images adalah sebuah template yang bersifat read only. Template ini sebenarnya adalah sebuah OS atau OS yang telah diinstall berbagai aplikasi. Docker images berfungsi untuk membuat docker container, dengan hanya 1 docker images kita dapat membuat banyak docker container.

Docker Container

Docker container bisa dikatakan sebagai sebuah folder, dimana docker container ini dibuat dengan menggunakan docker container. Setiap docker container disimpan maka akan terbentuk layer baru tepat diatas docker images atau base image diatasnya. Contohnya misalkan kita menggunakan image ubuntu, kemudian kita membuat sebuah container dari image ubuntu tersebut dengan nama ubuntuku, kemudian kita lakukan instalasi sebuah software misalnya nginx maka secara otomatis container ubuntuku akan berada diatas layer image atau base image ubuntu. Anda dapat membuat banyak docker container dari 1 docker images. Docker container ini nantinya dapat dibuild sehingga akan menghasilkan sebuah docker images, dan docker images yang dihasilkan dari docker container ini dapat kita gunakan kembali untuk membuat docker container yang baru.

Docker Registry

Docker registry adalah kumpulan docker image yang bersifat private maupun public yang dapat anda akses di [docker hub](#). Dengan menggunakan docker registry, anda dapat menggunakan docker image yang telah dibuat oleh developer yang lain, sehingga mempermudahkan kita dalam pengembangan aplikasi.

BAB II

Pengenalan Teknologi Portainer

[Portainer](#) adalah aplikasi yang ringan, cross-platform, dan open source UI untuk manajemen Docker. Portainer memberikan gambaran rinci tentang Docker dan memungkinkan Anda untuk mengelola kontainer, Docker image, jaringan dan volume masing masing kontainer melalui web dashboard sederhana.



Mengapa kita menggunakan portainer pada docker?

1. Mudah digunakan.

Dengan menggunakan portainer melakukan managemen docker pun menjadi lebih mudah, karena portainer memberikan gambaran rinci tentang Docker dan memungkinkan Anda untuk mengelola kontainer, gambar, jaringan dan volume.

2. Mudah dalam Instalasi

Karna sudah memiliki docker maka untuk melakukan instalasi pun sangat mudah, kita hanya perlu download portainer menggunakan docker dengan perintah “pull” pada docker itu sendiri. Tidak membutuhkan waktu yang sangat banyak untuk melakukan instalasi pada portainer.

3. Portainer dibuat untuk docker.

Portainer telah dibangun untuk berjalan pada mesin Docker sehingga dapat dijalankan di mana saja di mana Docker berjalan. Hal ini kompatibel dengan Docker untuk Linux dan Docker untuk Windows.

4. Penyesuaian dengan versi docker

Portainer telah dirancang untuk mendukung segala sesuatu fitur yang API Docker tawarkan. Docker API akan berkembang dan mengusulkan fitur baru, dan portainer akan menyesuaikan.

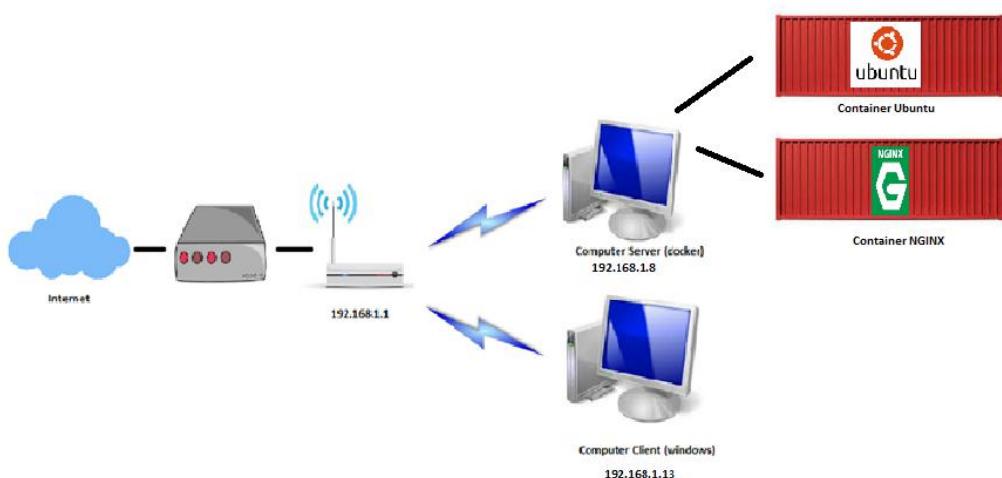
BAB III

Perencanaan dan Tujuan Akhir Dengan Docker

Sebelum masuk kedalam instalasi, dan realisasi docker. Pada bab ini akan menjelaskan tentang topologi dan alur dari topologi yang akan kita buat agar lebih mudah untuk dipahami, dan penjelasan tujuan daripada realisasi docker.

1. Topologi pada docker.

Berikut ini merupakan topologi yang akan kita buat untuk realisasi docker itu sendiri.



2. Penjelasan topologi

Modem terhubung ke Internet dan meneruskan kepada router, Pada router memiliki IP:192.168.1.1 dan terdapat 2 pc yang terhubung kepada router.

- PC Docker dengan IP : 192.168.1.8
- PC Client dengan IP : 192.168.1.13

Pada docker server memiliki Container Ubuntu, dan juga Web server NGINX yang nantinya akan diakses oleh computer client dengan operating system windows.

3. Hasil akhir daripada docker

Pada docker server terdapat 2 images, dan sudah dibuat berupa container untuk masing masing image. Dimana images yang terdapat pada docker server yaitu:

- Ubuntu 16.04
- NGINX (web server)

Selanjutnya client yang menggunakan operating system Windows akan mengakses server docker dan menjalankan Container tersebut.

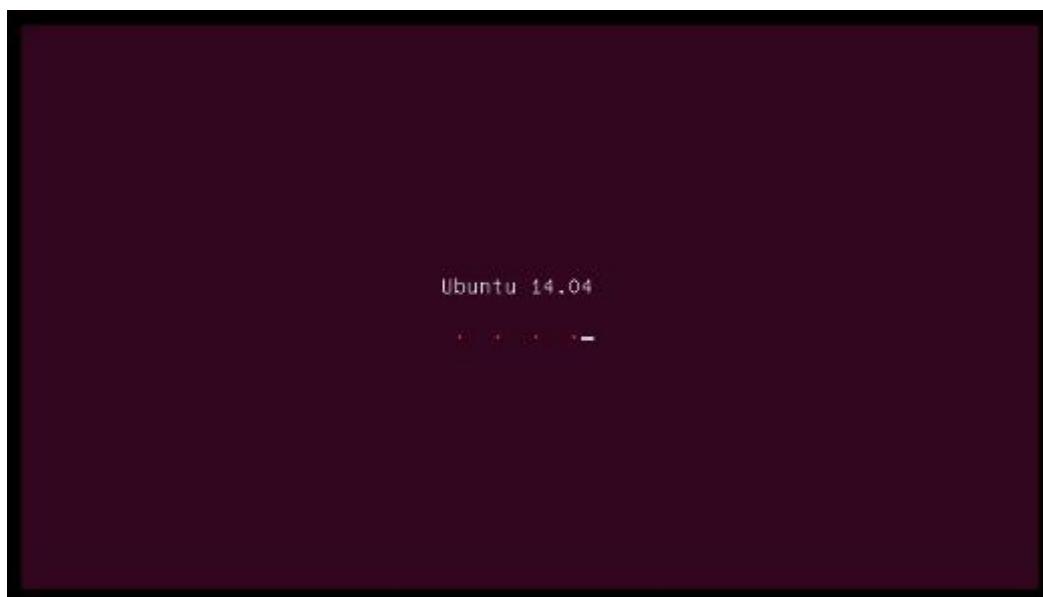
BAB IV

CARA MENGINSTAL SISTEM OPERASI LINUX UBUNTU

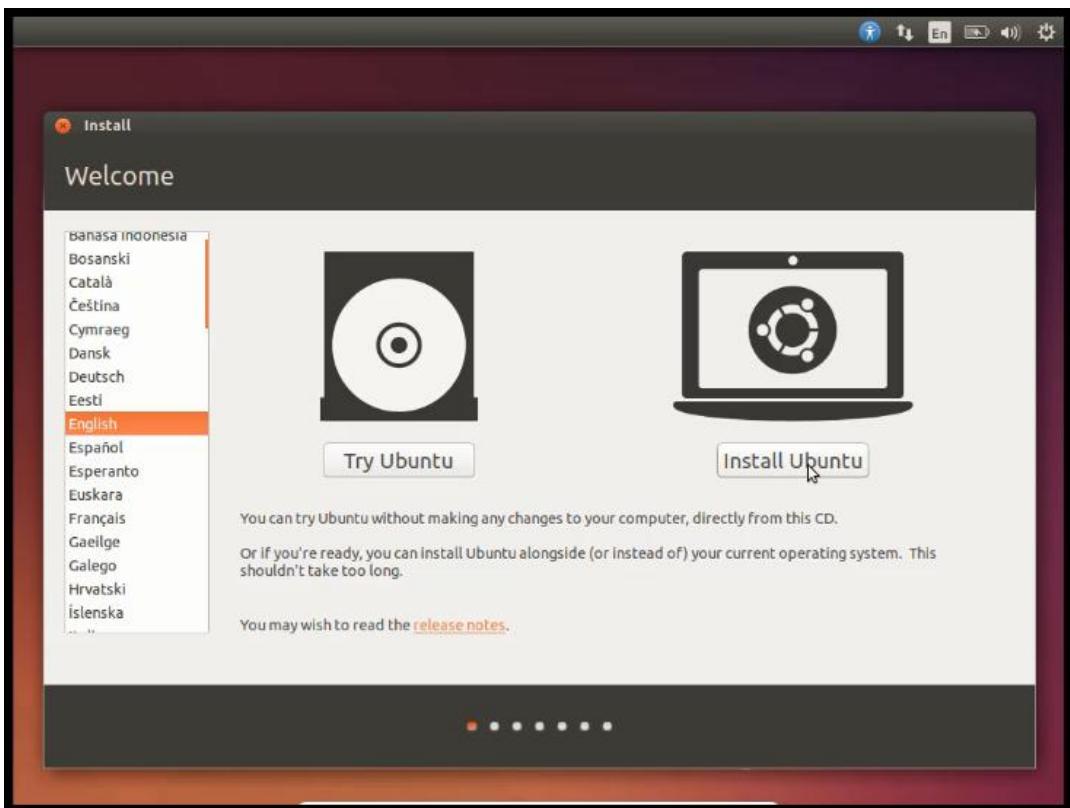
Banyak orang yang menggunakan Sistem Operasi Linux untuk dijadikan server karena Linux ini bersifat open source yang artinya pengguna tidak harus membayar untuk mendapatkannya dan dapat digunakan sesuai yang diinginkan. Keuntungan dari open source yaitu perangkat lunak bebas sering tersedia tanpa biaya, dan ketika perangkat lunak ini dikembangkan oleh orang lain maka pengembangan dari perangkat lunak dapat digunakan oleh semua orang.

Tahapan instalasi pada Linux sebenarnya bervariasi, tergantung dari distro yang digunakan, metodenya pun beragam. Terdapat distro yang menggunakan consol, ada yang menggunakan text-based seperti instalasi Windows, ada juga yang benar-benar menggunakan grafis antarmuka. Kali ini distro yang digunakan adalah Ubuntu versi 14.04. Tahapan instalasinya sebagai berikut:

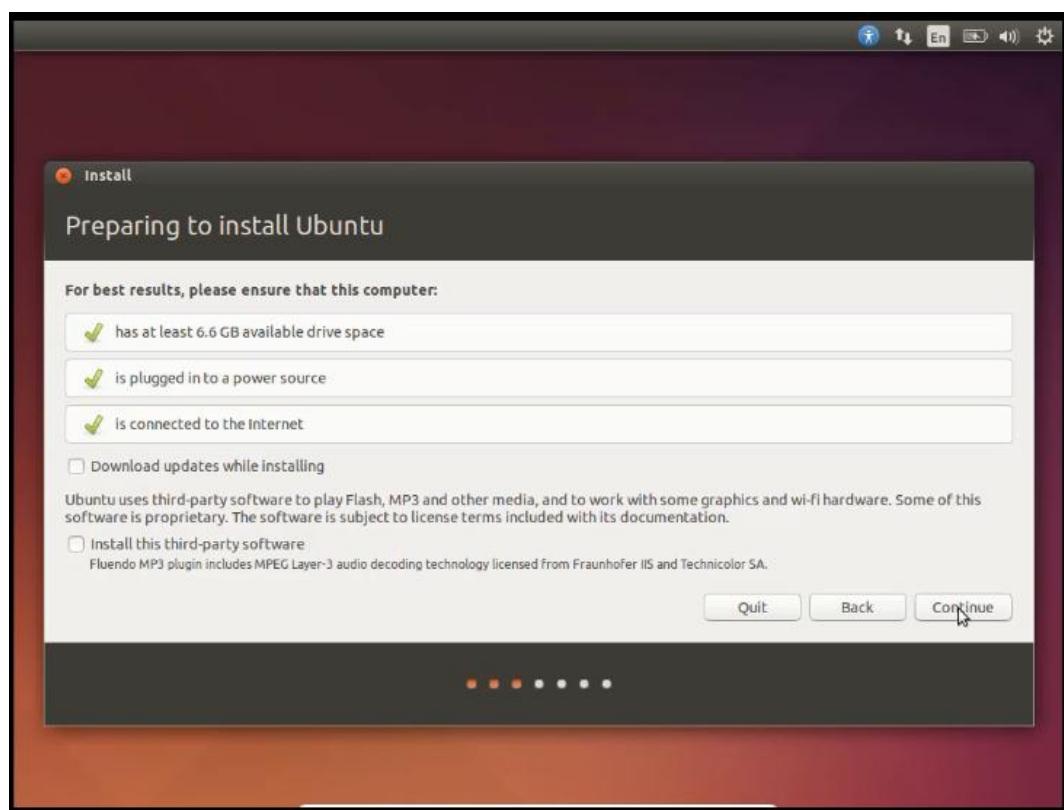
1. Pertama yang dilakukan adalah memilih ISO Ubuntu yang akan digunakan dalam penginstalan. Bisa dilihat disini saya menggunakan Ubuntu versi 14.04.



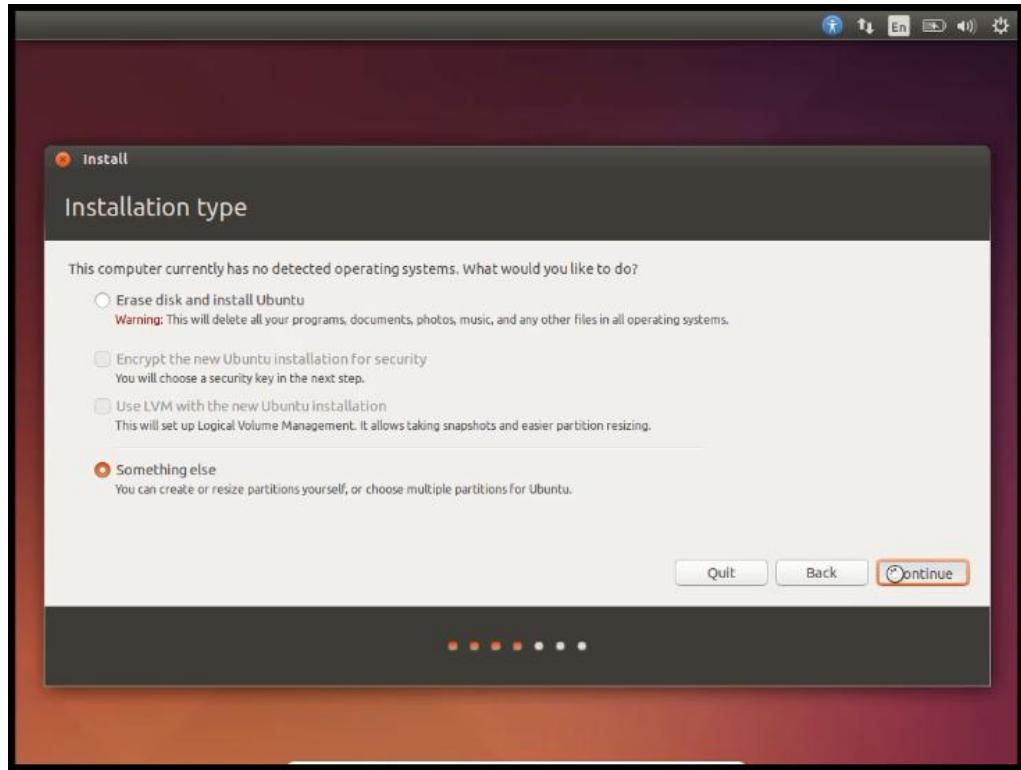
2. Tunggu beberapa saat hingga tampilan Welcome Window muncul. Langkah awal dalam penginstalan Ubuntu yaitu memilih bahasa yang akan digunakan saat kita menggunakan sistem operasi Ubuntu ini. Default dari sistemnya menggunakan English. Jika hendak mencoba dulu Ubuntu tanpa melakukan instalasi, maka dapat memilih menu "Try Ubuntu". Namun jika hendak melakukan instal Ubuntu, maka pilih menu "Install Ubuntu".



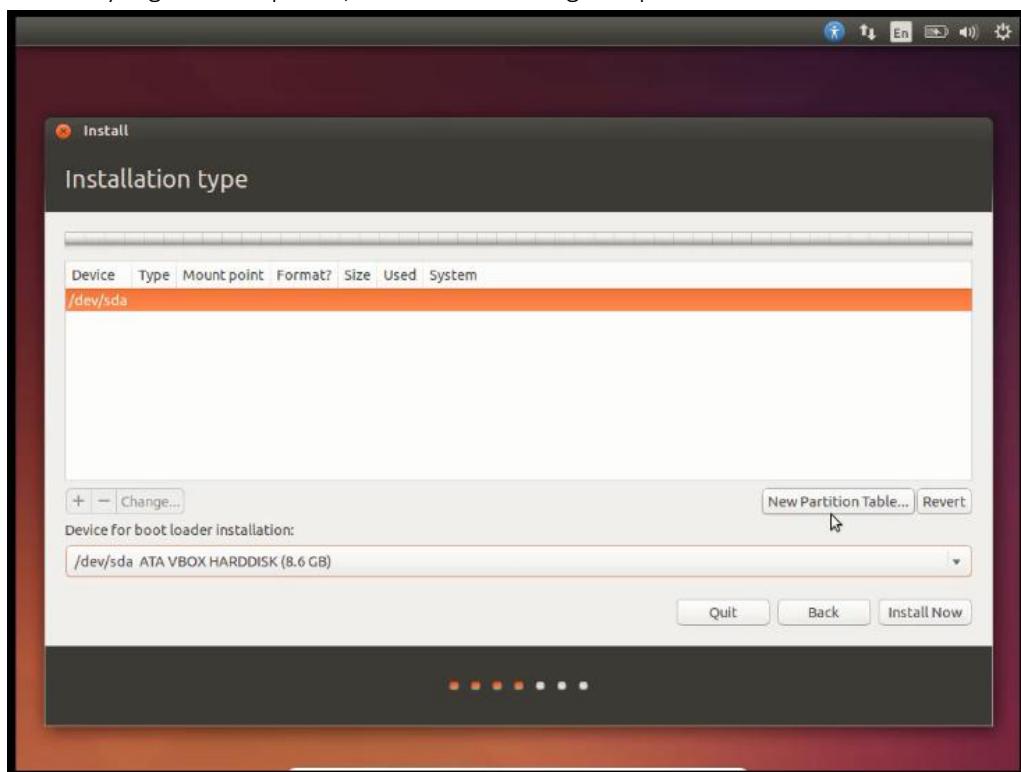
3. Selanjutnya akan muncul anjuran untuk menyediakan 6.6 space harddisk, terhubung ke daya listrik, an terkoneksi ke internet. Apabila sudah ada tanda ceklist (V) langsung klik **Continue**.



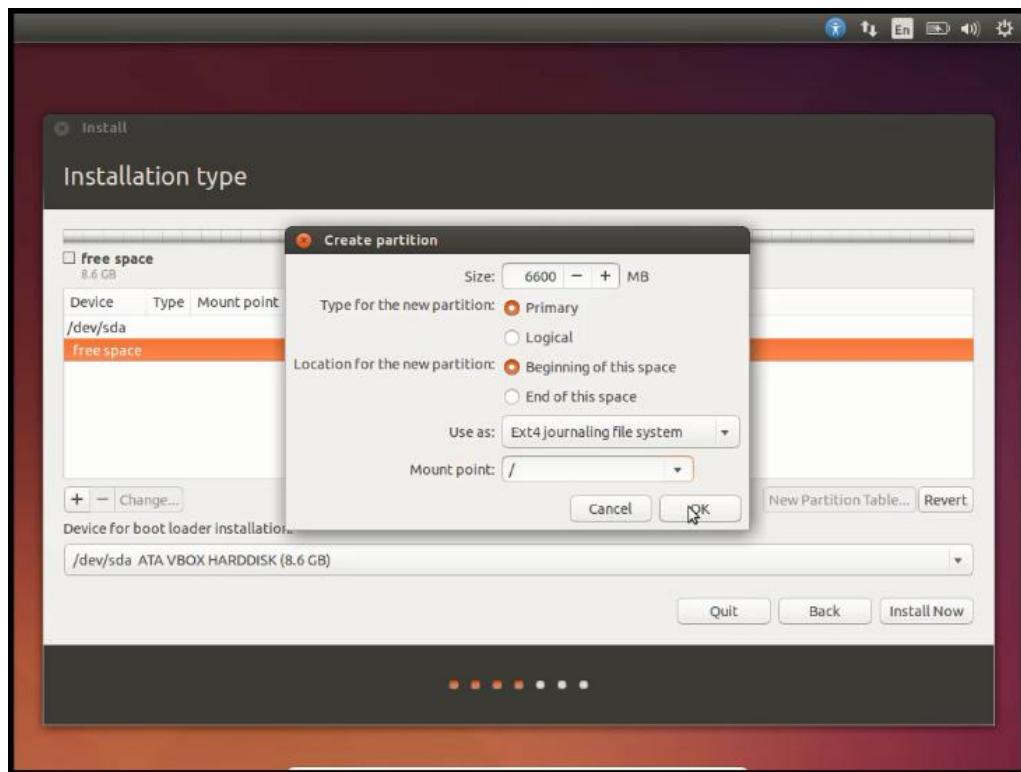
4. Pada tahap ini yang kita lakukan adalah mempersiapkan partisi disk, namun terdapat beberapa pilihan. Pilihan pertama “Erase Disk and Install Ubuntu” digunakan jika keseluruhan harddisk akan dipakai oleh ubuntu. Pilihan terakhir “Somthing Else” digunakan untuk mengatur partisi harddisk secara manual. Lanjut dengan klik **Install Now**.



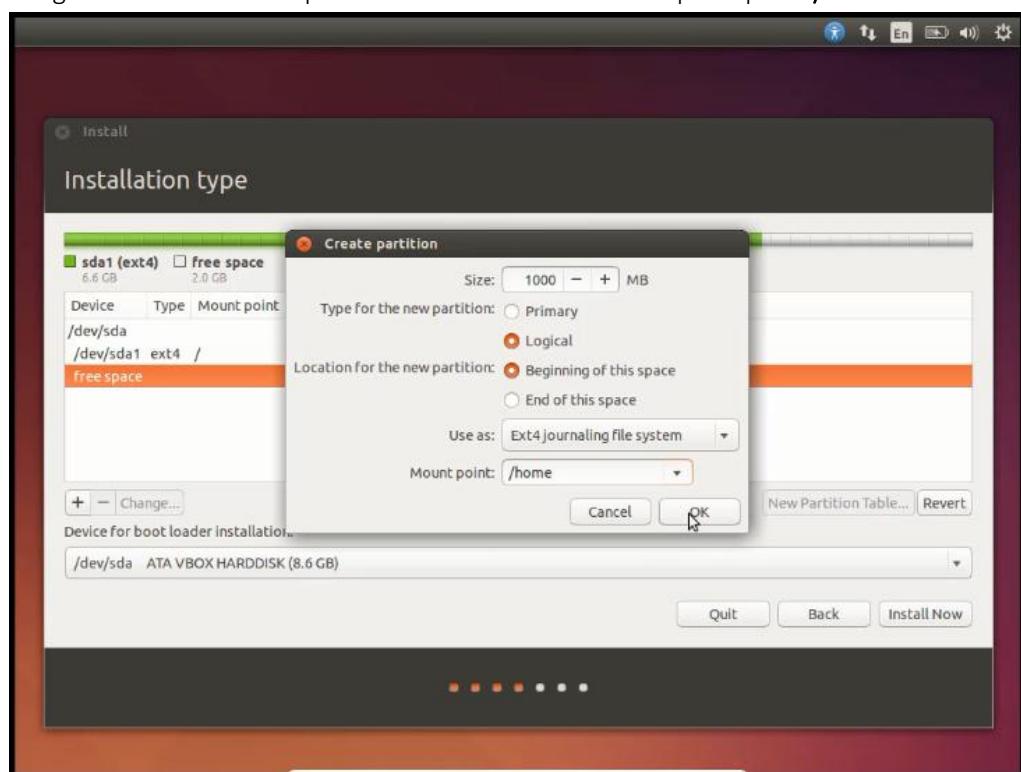
5. Pilih free space yang akan digunakan, kemudian klik **New Partitions**. Free space disini merupakan area harddisk yang belum dipartisi, bukan area kosong dari partisi itu sendiri.



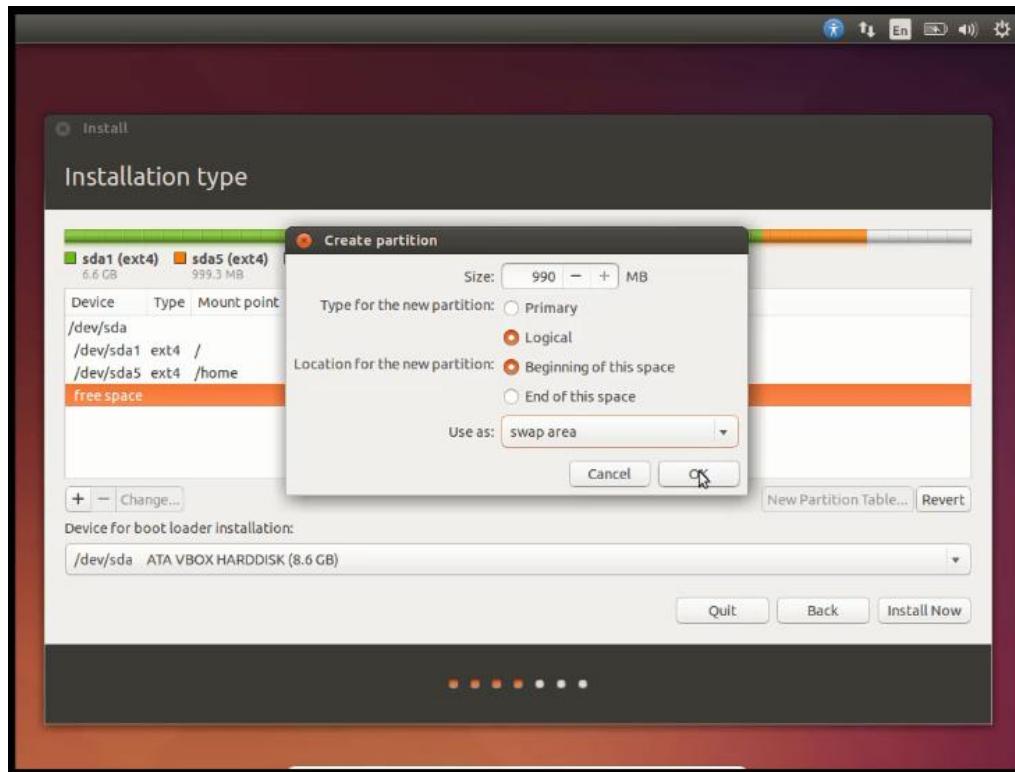
6. Kemudian **klik + (plus)**. Ini untuk membuat partisi utamanya dengan **ukuran 6000 MB**. Pilih **primary** sebagai tipe partisi. Pilih **beginning** sebagai lokasi partisi. Dan pilih **format Ext4** sebagai format terbaru di partisi. Kemudian untuk mount point pilih **“/”**. Lalu klik **OK**.



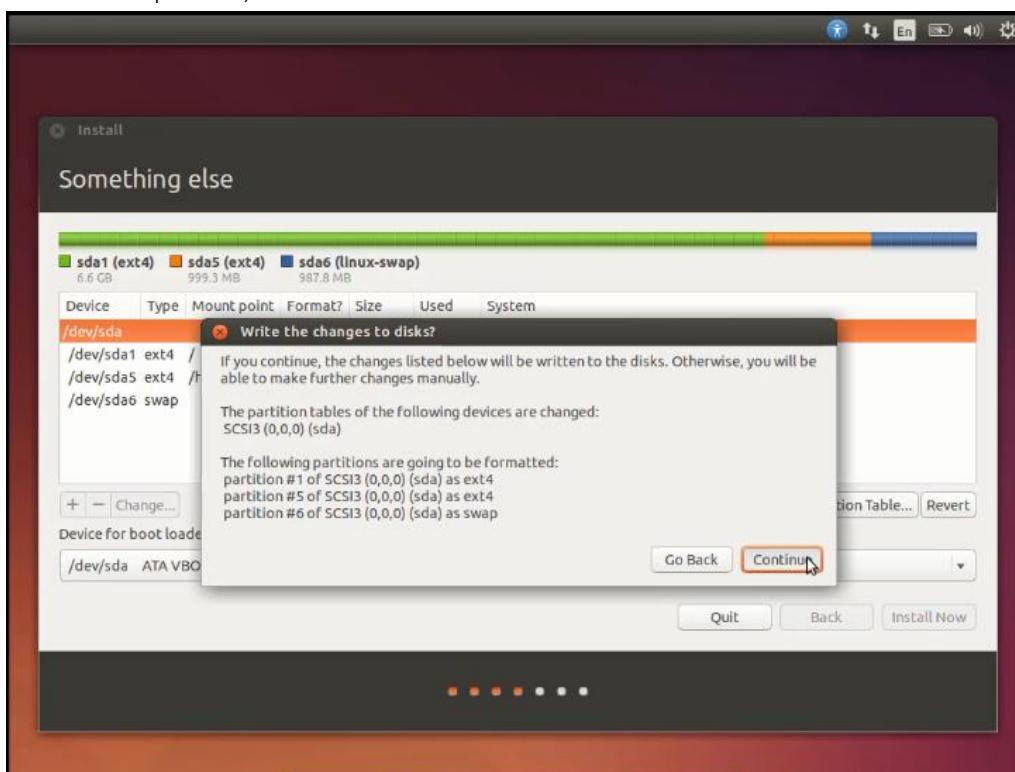
7. Setelah partisi utama terbentuk, buatlah partisi kedua dengan cara klik + (plus). Dengan **ukuran 1000 MB**. Pilih **logical** sebagai tipe partisi. Pilih **beginning** sebagai lokasi partisi. Dan pilih **format Ext4** sebagai format terbaru di partisi. Kemudian untuk mount point pilih **“/home”**. Lalu klik **OK**.



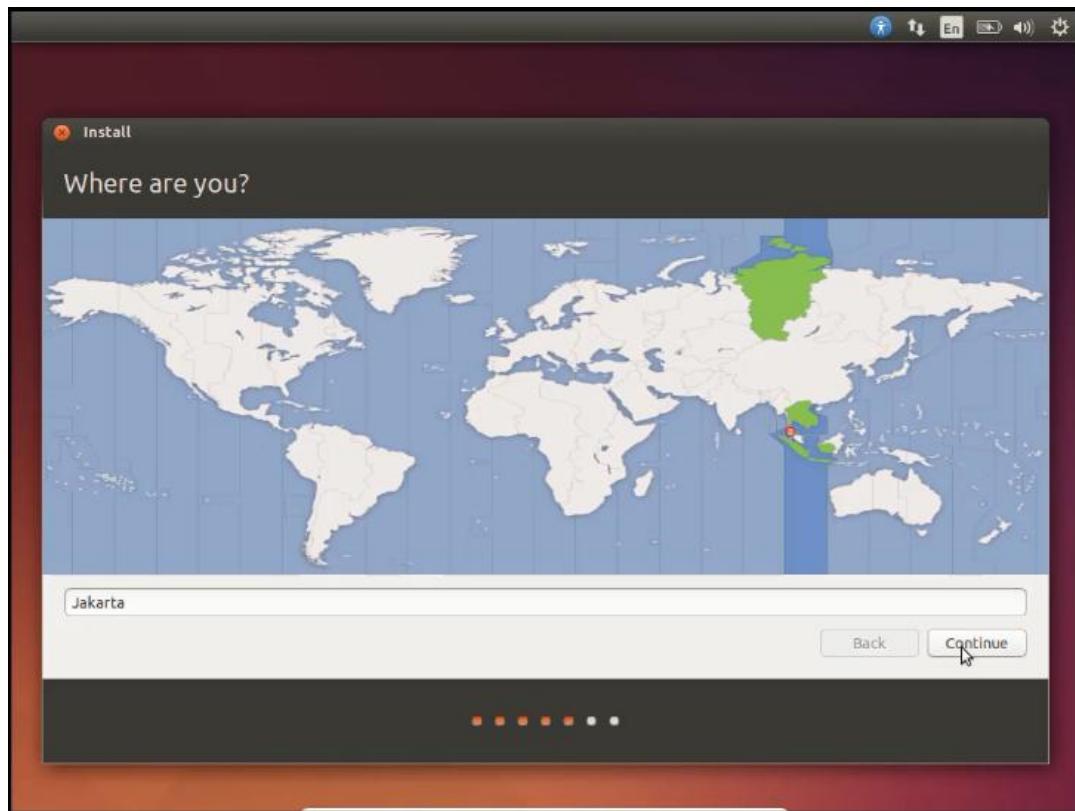
8. Setelah itu buatlah swap area menggunakan **sisa free space** tadi. Klik **+** (plus). Tipe partisi untuk swap area bisa menggunakan **primary maupun logical**. Jangan lupa pilih **format swap area**. Lalu klik **OK**. Kalau sudah semua maka langsung aja klik install now di bagian kanan bawah



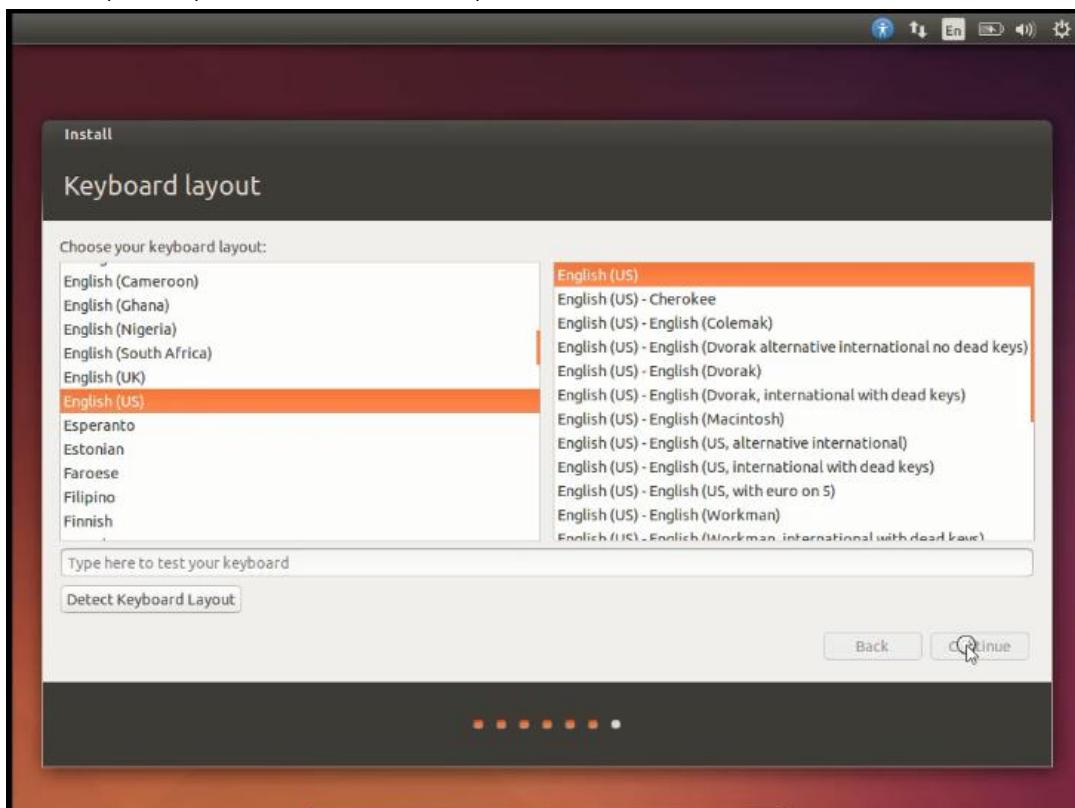
9. Apabila muncul seperti ini, klik continue.



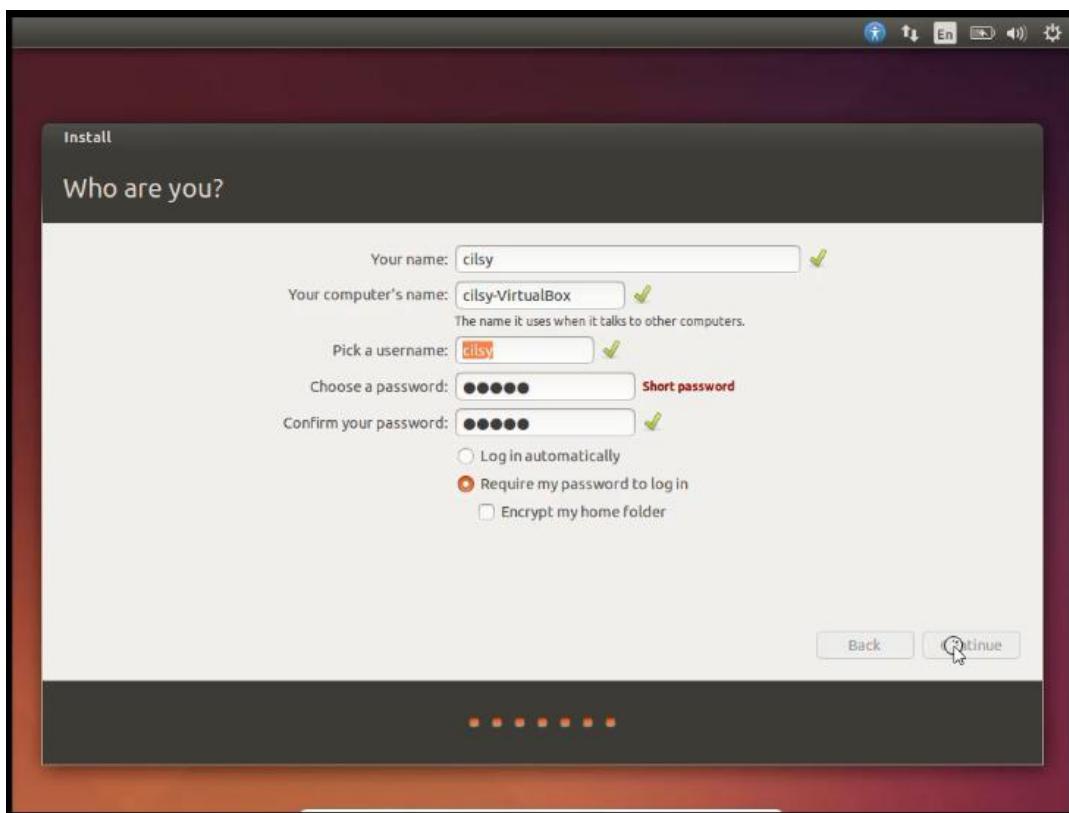
10. Kemudian atur lokasi dimana kita berada. Pilih jakarta. Lalu continue



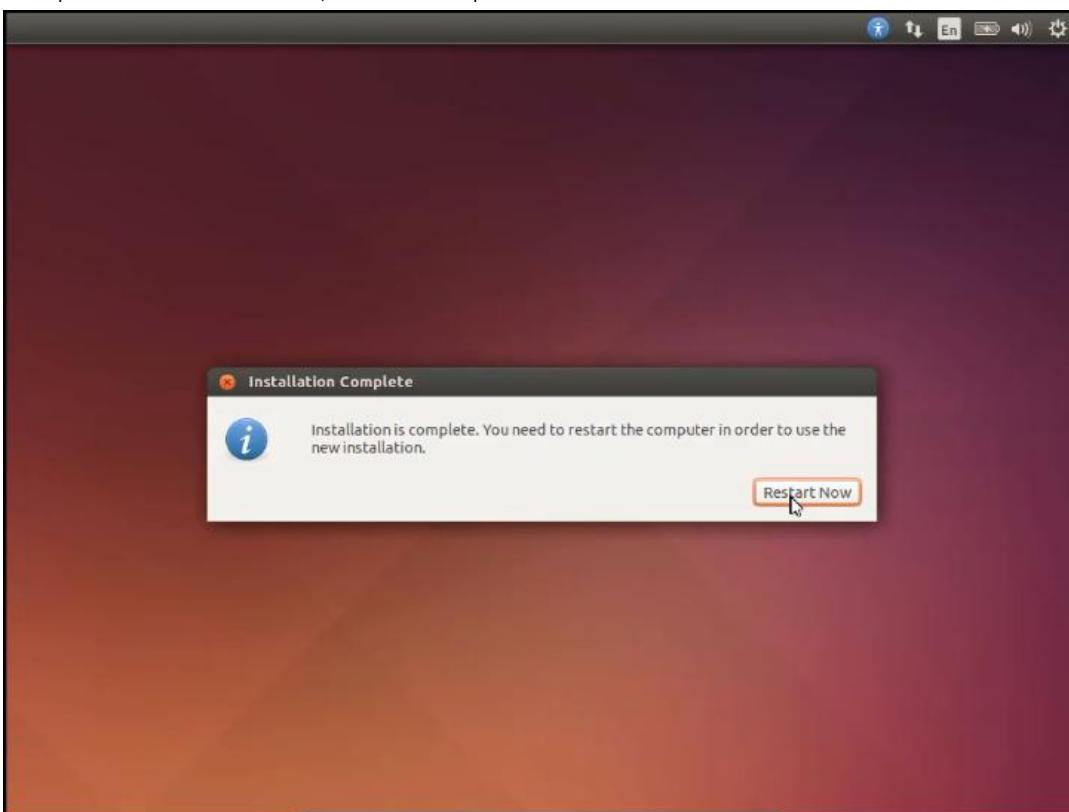
11. Tentukan layout keyboard. Default sistemnya adalah US. Lalu continue



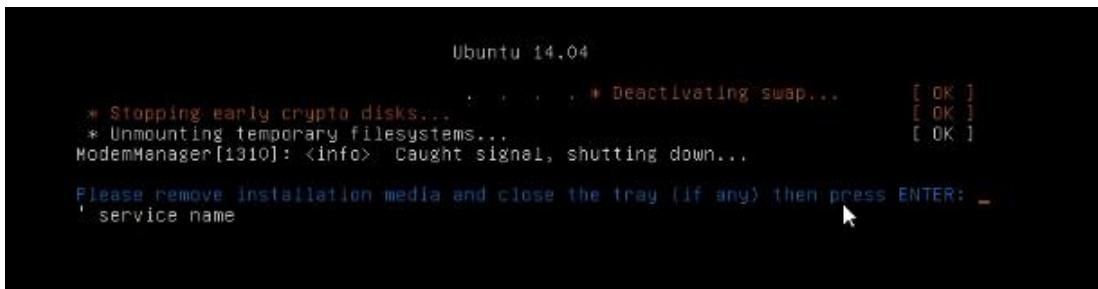
12. Ini tampilan untuk menuliskan profil komputer kita. Kemudian masukkan password. Penulisan password harus diingat terus karena akan digunakan saat akan masuk ke ubuntu setelah selesai instalasi nanti.



13. Setelah proses instalasi selesai, restart komputer.

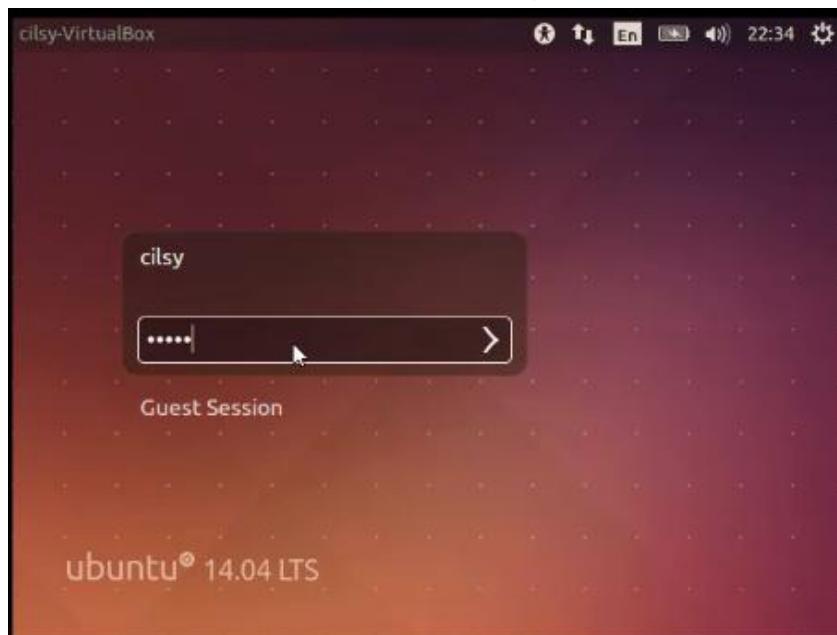


14. Apabila muncul seperti ini maka tekan enter.

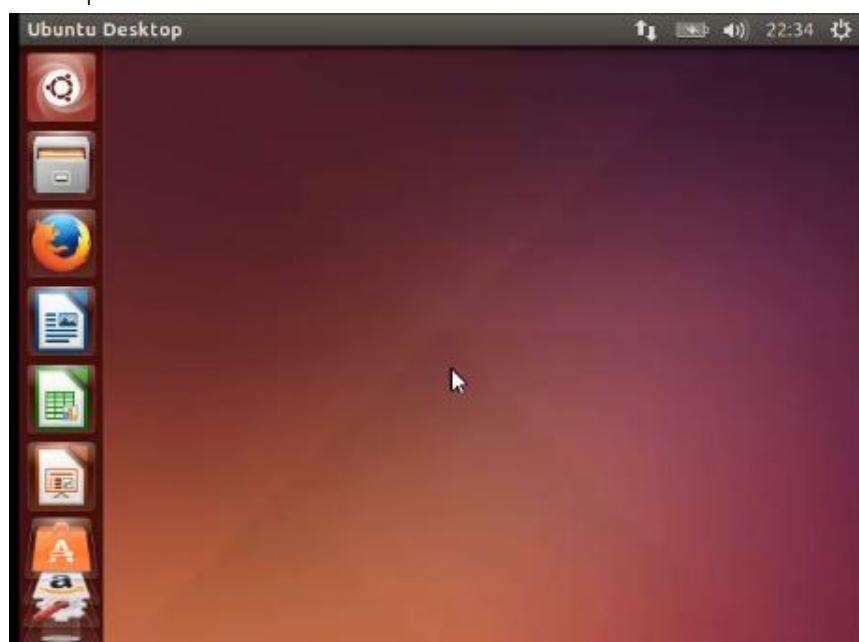


```
Ubuntu 14.04
* Stopping early crypto disks... [ OK ] * Deactivating swap... [ OK ]
* Unmounting temporary filesystems... [ OK ]
ModemManager[1310]: <info> Caught signal, shutting down...
Please remove installation media and close the tray (if any) then press ENTER:
' service name'
```

15. Ini adalah tampilan awal ubuntu. Masukkan password yang tadi telah dibuat.



16. Ini tampilan desktop dari ubuntu 14.04.

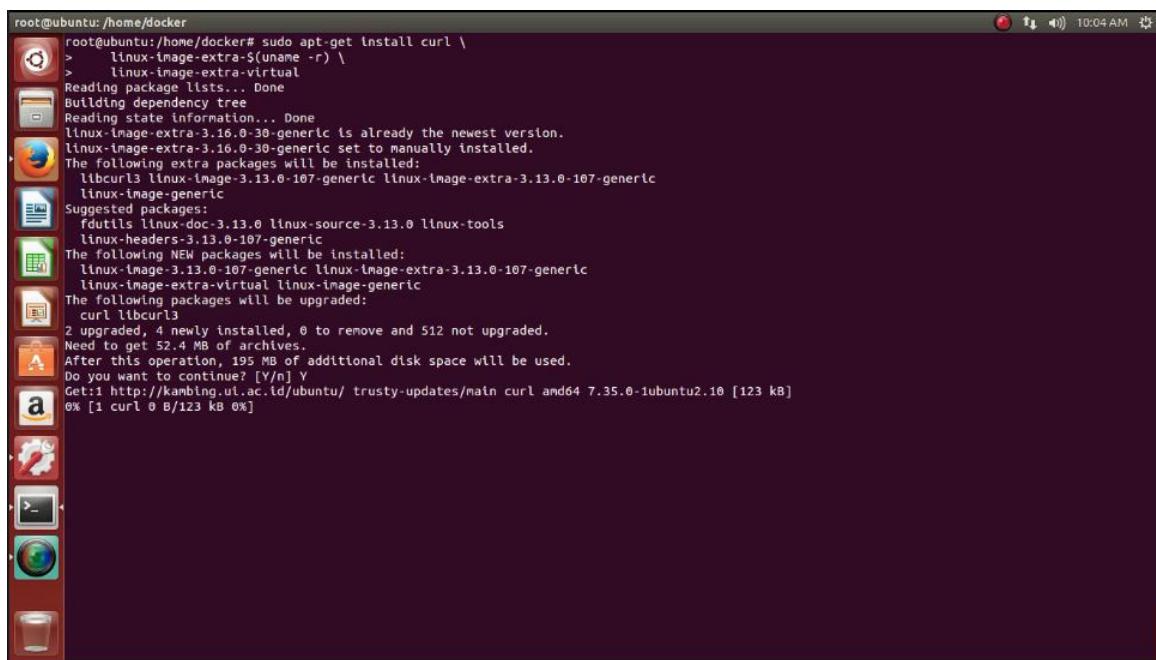


BAB V

CARA INSTAL DOCKER PADA UBUNTU

1. Lakukan penginstalan curl pada Ubuntu apabila tidak memilikinya. Kemudian muncul tulisan “Do you want to continue?” → Ketik Y.

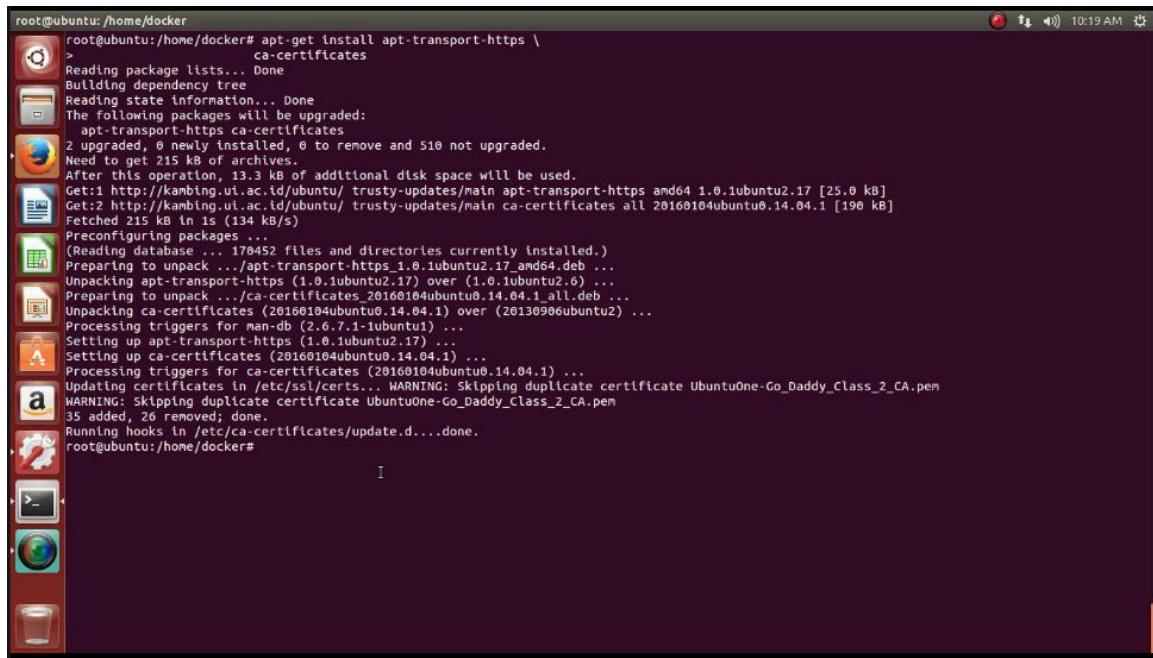
```
$ sudo apt-get install curl \ linux-image-extra-$(uname -r) \ linux-image-extra-virtual
```



```
root@ubuntu:/home/docker# sudo apt-get install curl \
>   linux-image-extra-$(uname -r) \
>   linux-image-extra-virtual
Reading package lists... Done
Building dependency tree
Reading state information... Done
linux-image-extra-3.16.0-30-generic is already the newest version.
linux-image-extra-3.16.0-30-generic set to manually installed.
The following extra packages will be installed:
  libcurl3 linux-image-3.13.0-107-generic linux-image-extra-3.13.0-107-generic
  linux-image-generic
Suggested packages:
  fdutils linux-doc-3.13.0 linux-source-3.13.0 linux-tools
  linux-headers-3.13.0-107-generic
The following NEW packages will be installed:
  linux-image-3.13.0-107-generic linux-image-extra-3.13.0-107-generic
  linux-image-extra-virtual linux-image-generic
The following packages will be upgraded:
  curl libcurl3
2 upgraded, 4 newly installed, 0 to remove and 512 not upgraded.
Need to get 52.4 MB of archives.
After this operation, 195 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://kamling.ul.ac.id/ubuntu/ trusty-updates/main curl amd64 7.35.0-1ubuntu2.10 [123 kB]
0% [1 curl 0 B/123 kB 0%]
```

2. Instal paket-paket untuk menggunakan repository.

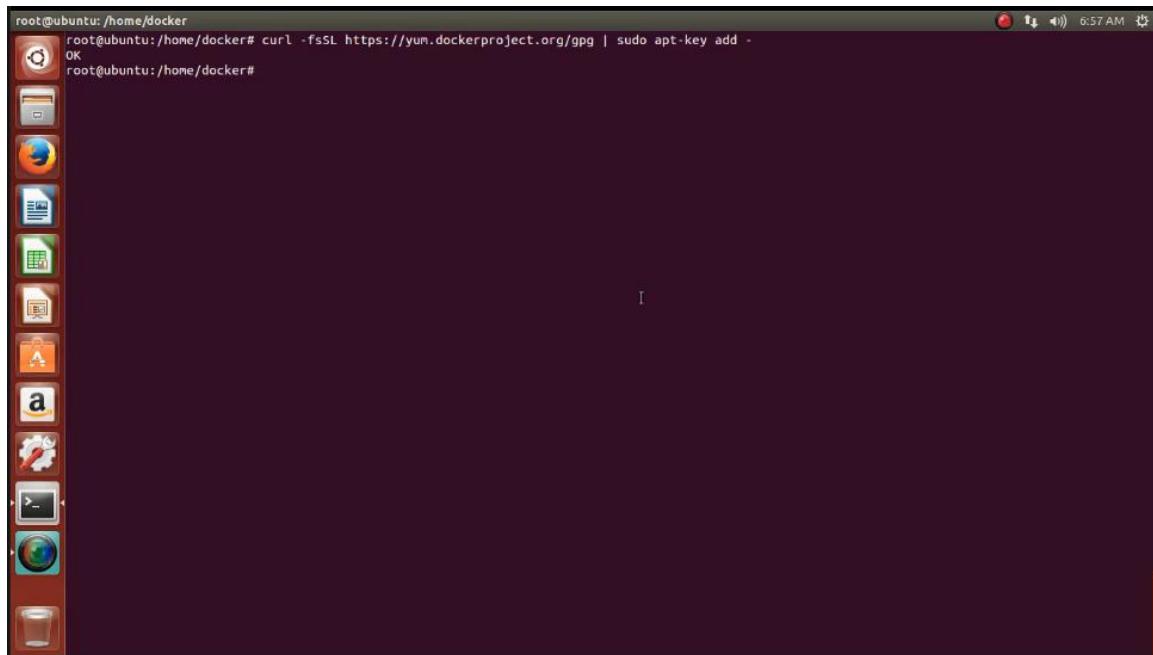
```
$ sudo apt-get install apt-transport-https \ ca-certificates
```



```
root@ubuntu:/home/docker# apt-get install apt-transport-https \
>                               ca-certificates
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  apt-transport-https ca-certificates
2 upgraded, 0 newly installed, 0 to remove and 510 not upgraded.
Need to get 215 kB of archives.
After this operation, 13.3 kB of additional disk space will be used.
Get:1 http://kanbing.ui.ac.id/ubuntu/ trusty-updates/main apt-transport-https amd64 1.0.1ubuntu2.17 [25.0 kB]
Get:2 http://kanbing.ui.ac.id/ubuntu/ trusty-updates/main ca-certificates all 20160104ubuntu0.14.04.1 [190 kB]
Fetched 215 kB in 1s (134 kB/s)
Preconfiguring packages ...
(Reading database ... 178452 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_1.0.1ubuntu2.17_amd64.deb ...
Unpacking apt-transport-https (1.0.1ubuntu2.17) over (1.0.1ubuntu2.6) ...
Preparing to unpack .../ca-certificates_20160104ubuntu0.14.04.1_all.deb ...
Unpacking ca-certificates (20160104ubuntu0.14.04.1) over (20130906ubuntu2) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up apt-transport-https (1.0.1ubuntu2.17) ...
Setting up ca-certificates (20160104ubuntu0.14.04.1) ...
Processing triggers for ca-certificates (20160104ubuntu0.14.04.1) ...
Updating certificates in /etc/ssl/certs... WARNING: Skipping duplicate certificate UbuntuOne-Go_Daddy_Class_2_CA.pem
WARNING: Skipping duplicate certificate UbuntuOne-Go_Daddy_Class_2_CA.pem
35 added, 26 removed; done.
Running hooks in /etc/ca-certificates/update.d....done.
root@ubuntu:/home/docker#
```

- Setelah itu tambahkan GPG key dari Docker Resminya.

```
$ curl -fsSL https://yum.dockerproject.org/gpg | sudo apt-key add -
```



```
root@ubuntu:/home/docker# curl -fsSL https://yum.dockerproject.org/gpg | sudo apt-key add -
OK
root@ubuntu:/home/docker#
```

- Memverifikasi key ID nya adalah 58118E89F3A912897C070ADBF76221572C52609D.

```
$ apt-key fingerprint 58118E89F3A912897C070ADBF76221572C52609D
```

```
root@ubuntu:/home/docker# apt-key fingerprint 58118E89F3A912897C070ADBF76221572C52609D
/etc/apt/trusted.gpg
-----  
pub 1024D/437D05B5 2004-09-12  
Key fingerprint = 6302 39CC 130E 1A7F D81A 27B1 4097 6EAF 437D 05B5  
uid Ubuntu Archive Automatic Signing Key <ftpmaster@ubuntu.com>  
sub 2048g/79164387 2004-09-12  
  
pub 1024D/FB875451 2004-12-30  
Key fingerprint = C598 6B4F 1257 FFA8 6632 CBA7 4618 1433 FBB7 5451  
uid Ubuntu CD Image Automatic Signing Key <cdimage@ubuntu.com>  
  
pub 4096R/C0B21F32 2012-05-11  
Key fingerprint = 7908 C727 7767 219C 42C8 6F93 3B4F E6AC C0B2 1F32  
uid Ubuntu Archive Automatic Signing Key (2012) <ftpmaster@ubuntu.com>  
  
pub 4096R/EFE21092 2012-05-11  
Key fingerprint = 8439 380F 228D 22F7 B374 28C0 D94A A3F0 EFE2 1092  
uid Ubuntu CD Image Automatic Signing Key (2012) <cdimage@ubuntu.com>  
  
pub 1024D/3E5C1192 2010-09-20  
Key fingerprint = C474 15DF F48C 0964 5B78 6094 1612 6D3A 3E5C 1192  
uid Ubuntu Extras Archive Automatic Signing Key <ftpmaster@ubuntu.com>  
  
pub 4096R/2C52609D 2015-07-14  
Key fingerprint = 5811 8E89 F3A9 1289 7C07 0ADB F762 2157 2C52 609D  
uid Docker Release Tool (releasedocker) <docker@docker.com>  
  
/etc/apt/trusted.gpg.d/maarten-baert-simplescreenrecorder.gpg  
-----  
pub 1024R/283EC8CD 2013-05-26  
Key fingerprint = 40ED B3E0 5F04 3CA1 8517 6AC0 409C 8B51 283E C8CD  
uid Launchpad PPA for Maarten Baert  
root@ubuntu:/home/docker#
```

5. Gunakan perintah berikut untuk mengatur repository.

```
$ sudo add-apt-repository \ "deb https://apt.dockerproject.org/repo/ \ ubuntu-(lsb_release -cs) \ main"
```

```
root@ubuntu:/home/docker# sudo add-apt-repository \
>     "deb https://apt.dockerproject.org/repo/ \
>     ubuntu-$(lsb_release -cs) \
>     main"
root@ubuntu:/home/docker#
```

6. Update paket index apt.

```
$ sudo apt-get update
```

```
root@ubuntu:/home/docker
> main"
root@ubuntu:/home/docker# sudo apt-get update
Ign http://kambing.ul.ac.id trusty InRelease
Hit http://kambing.ul.ac.id trusty-updates InRelease
Hit http://kambing.ul.ac.id trusty-backports InRelease
Hit http://kambing.ul.ac.id trusty-security InRelease
Hit http://kambing.ul.ac.id trusty Release.gpg
Ign http://extras.ubuntu.com trusty InRelease
Hit http://extras.ubuntu.com trusty Release.gpg
Hit http://kambing.ul.ac.id trusty Release
Hit http://ppa.launchpad.net trusty InRelease
Hit http://kambing.ul.ac.id trusty-updates/main Sources
Hit http://extras.ubuntu.com trusty Release
Hit http://kambing.ul.ac.id trusty-updates/restricted Sources
Hit http://kambing.ul.ac.id trusty-updates/universe Sources
Hit http://kambing.ul.ac.id trusty-updates/multiverse Sources
Hit http://kambing.ul.ac.id trusty-updates/main amd64 Packages
Hit http://kambing.ul.ac.id trusty-updates/restricted amd64 Packages
Hit http://kambing.ul.ac.id trusty-updates/universe amd64 Packages
Hit http://kambing.ul.ac.id trusty-updates/multiverse amd64 Packages
Hit http://kambing.ul.ac.id trusty-updates/main i386 Packages
Hit http://kambing.ul.ac.id trusty-updates/restricted i386 Packages
Hit http://ppa.launchpad.net trusty/main amd64 Packages
Hit http://extras.ubuntu.com trusty/main Sources
Hit http://extras.ubuntu.com trusty/main amd64 Packages
Hit http://ppa.launchpad.net trusty/main i386 Packages
Hit https://apt.dockerproject.org ubuntu-trusty InRelease
Ign https://apt.dockerproject.org ubuntu-trusty/main Translation-en_US
Ign https://apt.dockerproject.org ubuntu-trusty/main Translation-en
Ign https://apt.dockerproject.org ubuntu-trusty/main Translation-en_US
Ign https://extras.ubuntu.com trusty/main Translation-en
Ign https://extras.ubuntu.com trusty/main Translation-en
Hit http://kambing.ul.ac.id trusty-updates/universe i386 Packages
Hit http://kambing.ul.ac.id trusty-updates/multiverse i386 Packages
Hit http://kambing.ul.ac.id trusty-updates/main Translation-en
Hit http://kambing.ul.ac.id trusty-updates/multiverse Translation-en
100% [Translation-en 6,211 kB] 2,875 kB/s 0s
```

7. Instal Docker versi terbaru, atau skip langkah ini dan lanjut ke langkah selanjutnya untuk melihat versi tertentu. Gunakan perintah berikut ini untuk menginstal versi terbaru dari Docker.

```
$ sudo apt-get install docker-engine
```

```
root@ubuntu:/home/docker
root@ubuntu:/home/docker# sudo apt-get -y install docker-engine
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  docker-engine
0 upgraded, 1 newly installed, 0 to remove and 513 not upgraded.
Need to get 0 B/19.1 MB of archives.
After this operation, 88.5 MB of additional disk space will be used.
Selecting previously unselected package docker-engine.
(Reading database ... 171270 files and directories currently installed.)
Preparing to unpack .../docker-engine_1.13.0-0~ubuntu-trusty_amd64.deb ...
Unpacking docker-engine (1.13.0-0~ubuntu-trusty) ...
Processing triggers for Man-db (2.6.7.1-1ubuntu1) ...
Processing triggers for ureadahead (0.100.0-16) ...
ureadahead will be reprofiled on next reboot
Setting up docker-engine (1.13.0-0~ubuntu-trusty) ...
  docker start/running, process 6402
Processing triggers for ureadahead (0.100.0-16) ...
root@ubuntu:/home/docker#
```

8. Pada saat tertentu, kita diharuskan untuk menginstall versi Docker tertentu, tidak selalu menggunakan yang terbaru. Berikut perintah untuk melihat versi yang tersedia.

```
$ apt-cache madison docker-engine
```

```
root@ubuntu:/home/docker# apt-cache madison docker-engine
docker-engine | 1.13.0~0-ubuntu-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.12.6~0-ubuntu-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.12.5~0-ubuntu-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.12.4~0-ubuntu-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.12.3~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.12.2~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.12.1~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.12.0~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.11.2~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.11.1~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.11.0~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.10.3~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.10.2~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.10.1~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.10.0~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.9.1~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.9.0~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.8.3~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.8.2~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.8.1~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.8.0~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.7.1~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.7.0~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.6.2~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.6.1~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.6.0~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
docker-engine | 1.5.0~0-trusty | https://apt.dockerproject.org/repo/ ubuntu-trusty/main amd64 Packages
root@ubuntu:/home/docker#
```

9. Memverifikasi apakah instalasi docker telah benar atau belum dengan menjalankan images hello-world.

```
$ sudo docker run hello-world
```

```
root@ubuntu:/home/docker# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
78445dd45222: Pull complete
Digest: sha256:c551578d4c5e1e838e9cd307f6c6a0d620b5e07e6f927b07d05f6d12a1ac8d7
Status: Downloaded newer image for hello-world:latest

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.
 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.

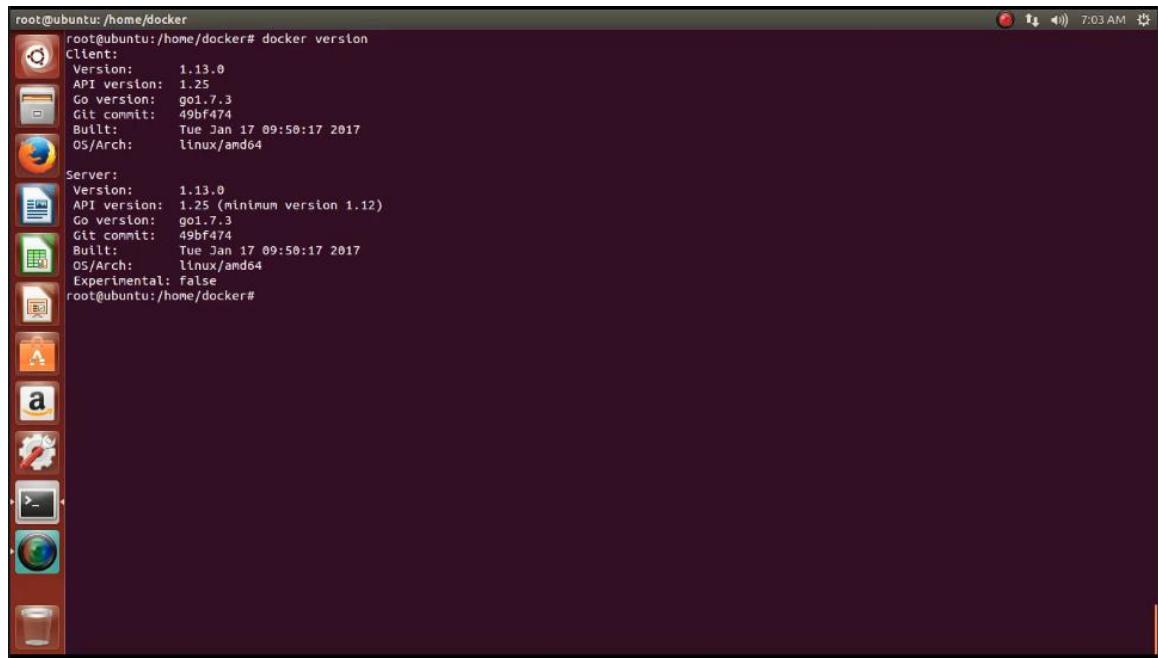
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

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

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/
root@ubuntu:/home/docker#
```

10. Melihat versi dari Docker yang terinstall di Ubuntu.

```
$ docker version
```



```
root@ubuntu:/home/docker
root@ubuntu:/home/docker# docker version
Client:
  Version: 1.13.0
  API version: 1.25
  Go version: go1.7.3
  Git commit: 49bf474
  Built: Tue Jan 17 09:50:17 2017
  OS/Arch: linux/amd64

Server:
  Version: 1.13.0
  API version: 1.25 (minimum version 1.12)
  Go version: go1.7.3
  Git commit: 49bf474
  Built: Tue Jan 17 09:50:17 2017
  OS/Arch: linux/amd64
  Experimental: false
root@ubuntu:/home/docker#
```

BAB VI

Cara Daftar Docker ID

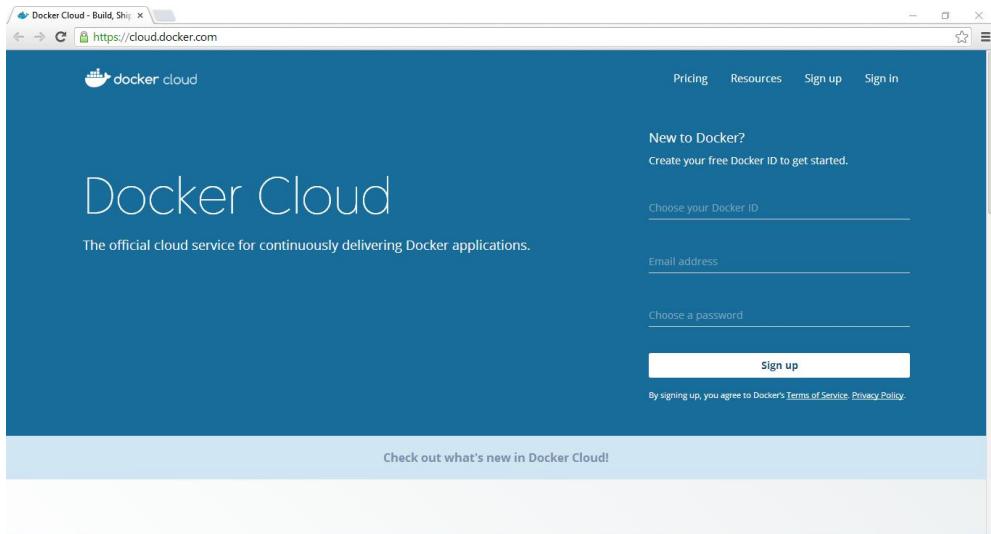
Docker ID memberi Anda akses ke layanan Docker seperti : Docker Store, Docker Cloud, repositori Docker Hub, dan beberapa program beta.

Docker ID Anda menjadi namespace repositori yang digunakan oleh host layanan seperti Docker Hub dan Docker Cloud. Yang Anda butuhkan adalah alamat email.

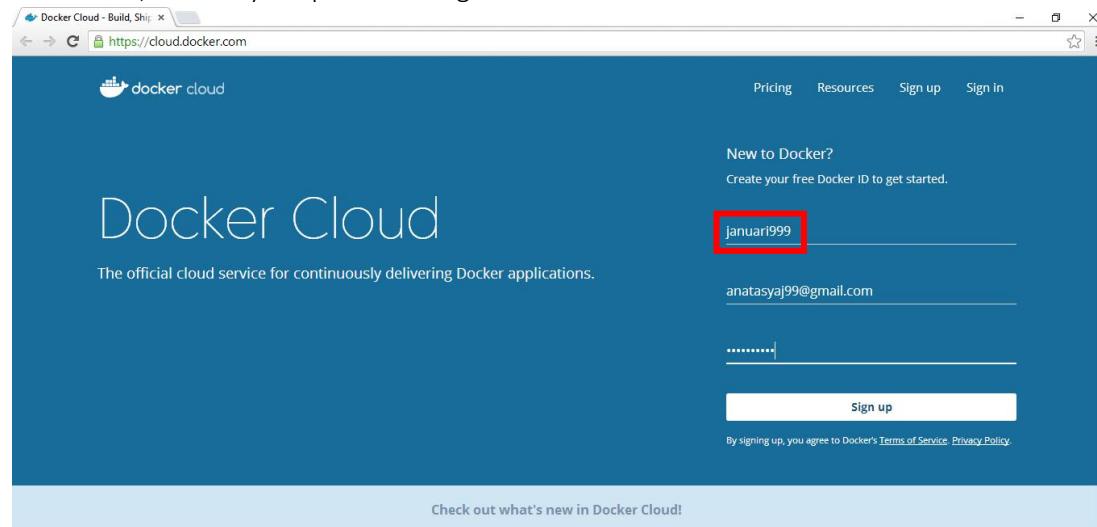
akun ini juga memungkinkan Anda untuk login ke layanan seperti Pusat Docker Support, Forum Docker, dan portal Docker Sukses.

Mendaftar Docker ID

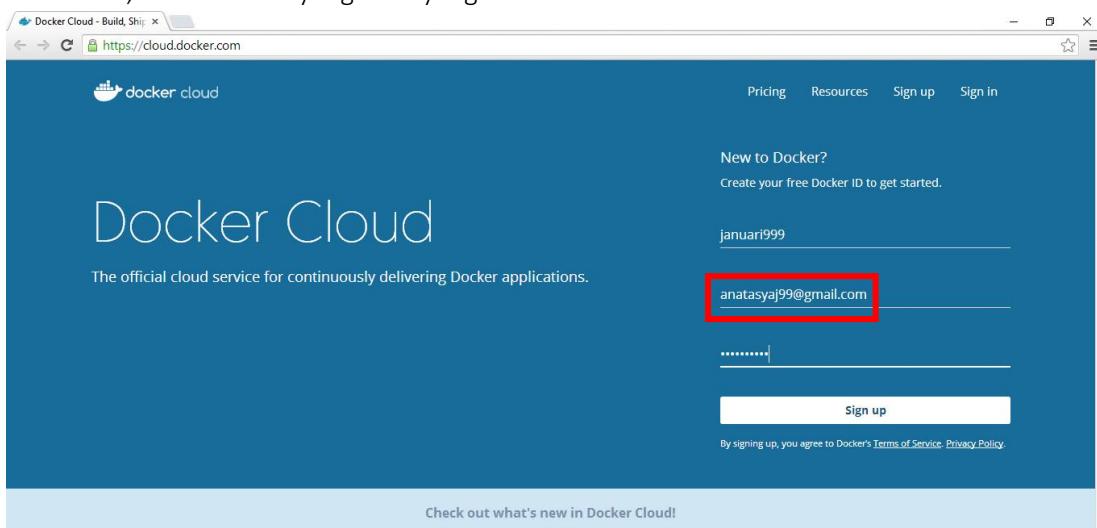
1. Buka halaman Docker Cloud mendaftar.



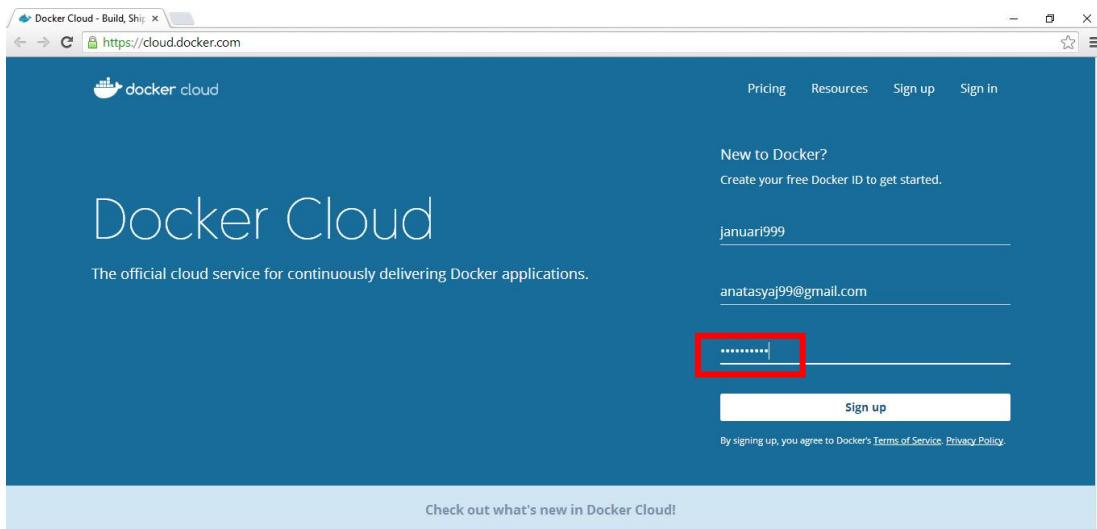
2. Masukkan username yang akan menjadi Docker ID Anda. Docker ID Anda harus antara 4 dan 30 karakter, dan hanya dapat berisi angka dan huruf kecil.



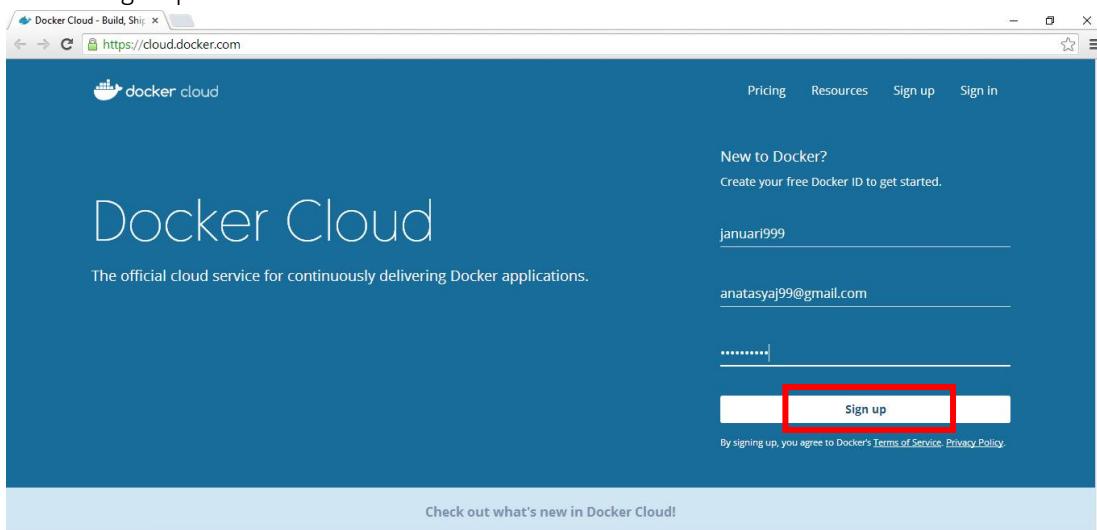
3. Masukkan, alamat email yang valid yang unik.



4. Masukkan password antara 6 dan 128 karakter.

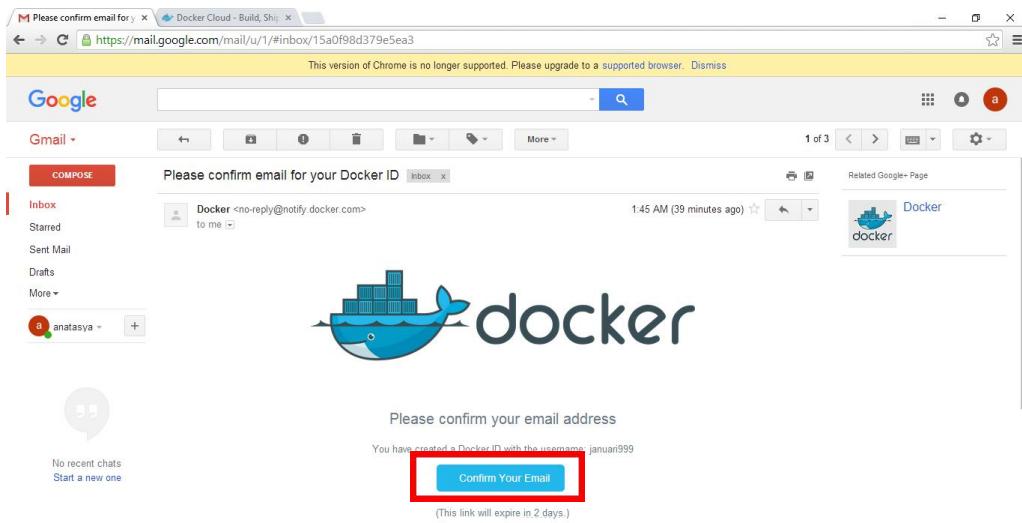


6. Klik Sign Up.



*Docker mengirimkan email verifikasi ke alamat yang Anda berikan.

7. Klik link dalam email untuk memverifikasi alamat Anda.



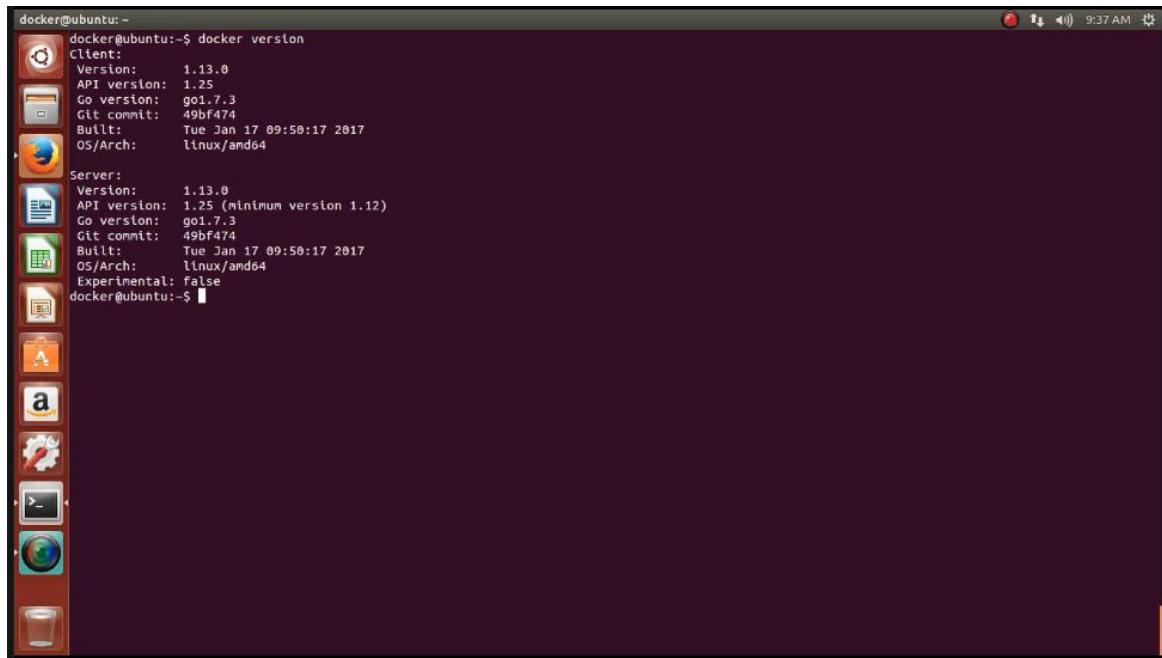
Catatan: Anda tidak bisa login dengan ID Docker jika belum memverifikasi alamat email Anda.

BAB VII

CARA INSTAL PORTAINER PADA UBUNTU

1. Pastikan pada Ubuntu anda sudah terinstall Docker.

```
$ docker version
```

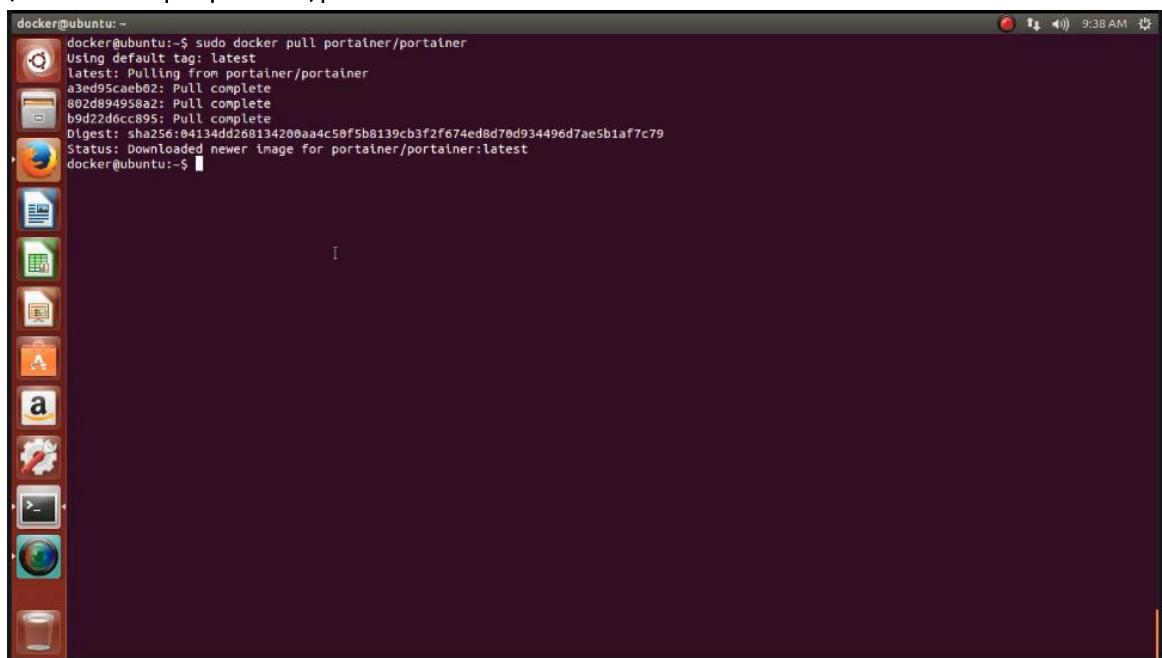


```
docker@ubuntu:~$ docker version
Client:
  Version: 1.13.0
  API version: 1.25
  Go version: go1.7.3
  Git commit: 49bf474
  Built: Tue Jan 17 09:50:17 2017
  OS/Arch: linux/amd64

Server:
  Version: 1.13.0
  API version: 1.25 (minimum version 1.12)
  Go version: go1.7.3
  Git commit: 49bf474
  Built: Tue Jan 17 09:50:17 2017
  OS/Arch: linux/amd64
  Experimental: false
docker@ubuntu:~$
```

2. Pull

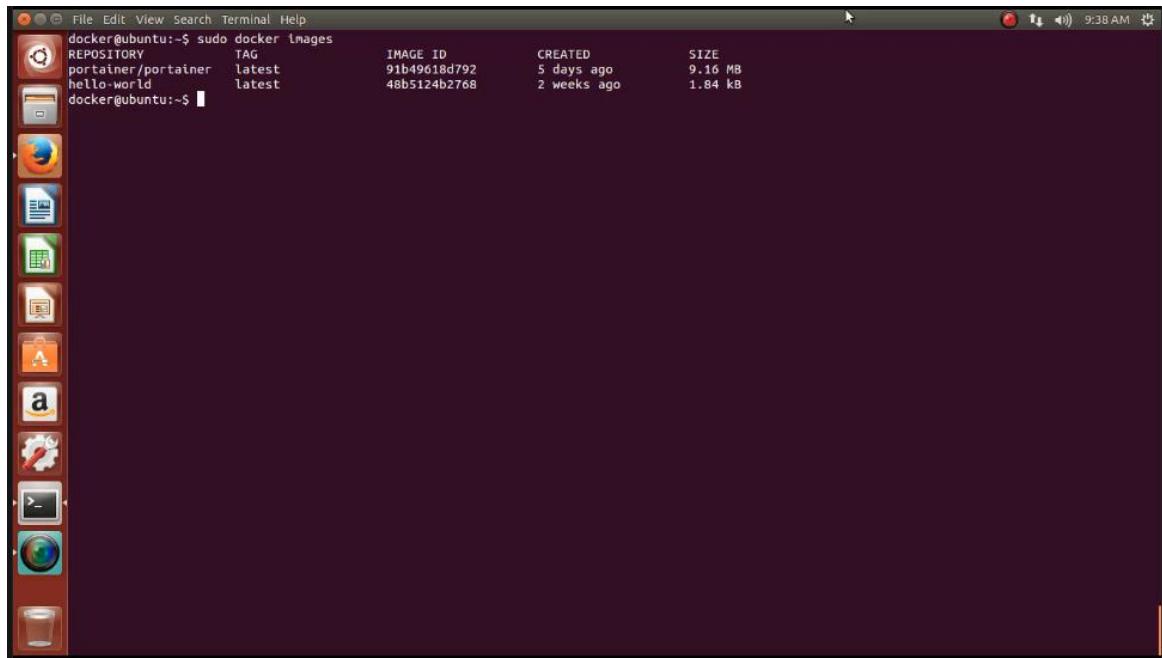
```
$ sudo docker pull portainer/portainer
```



```
docker@ubuntu:~$ sudo docker pull portainer/portainer
Using default tag: latest
latest: Pulling from portainer/portainer
a3ed95caeb02: Pull complete
802d894958a2: Pull complete
b9d27d6cc895: Pull complete
Digest: sha256:84134dd268134200aa4c50f5b8139cb3f2f674ed8d70d934496d7ae5b1af7c79
Status: Downloaded newer image for portainer/portainer:latest
docker@ubuntu:~$
```

3. Melihat docker images yang ada

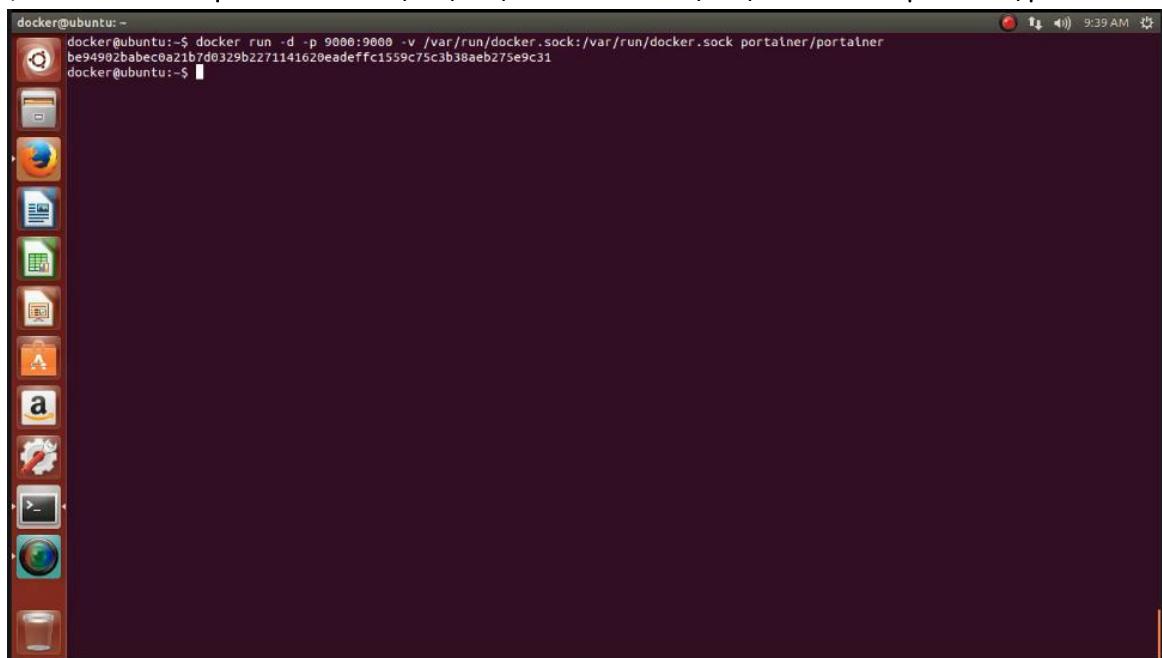
```
$ sudo docker images
```



```
File Edit View Search Terminal Help
docker@ubuntu:~$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
portainer/portainer latest 91b49618d792 5 days ago 9.16 MB
hello-world latest 48b5124b2768 2 weeks ago 1.84 kB
docker@ubuntu:~$
```

4.

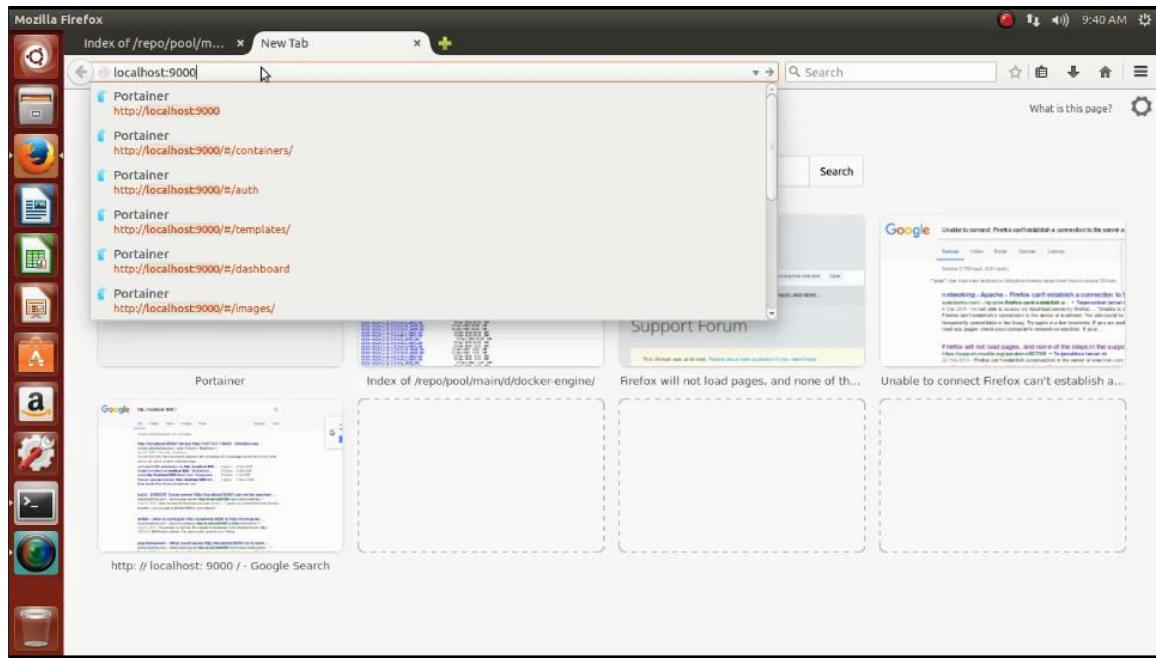
```
$ docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer
```



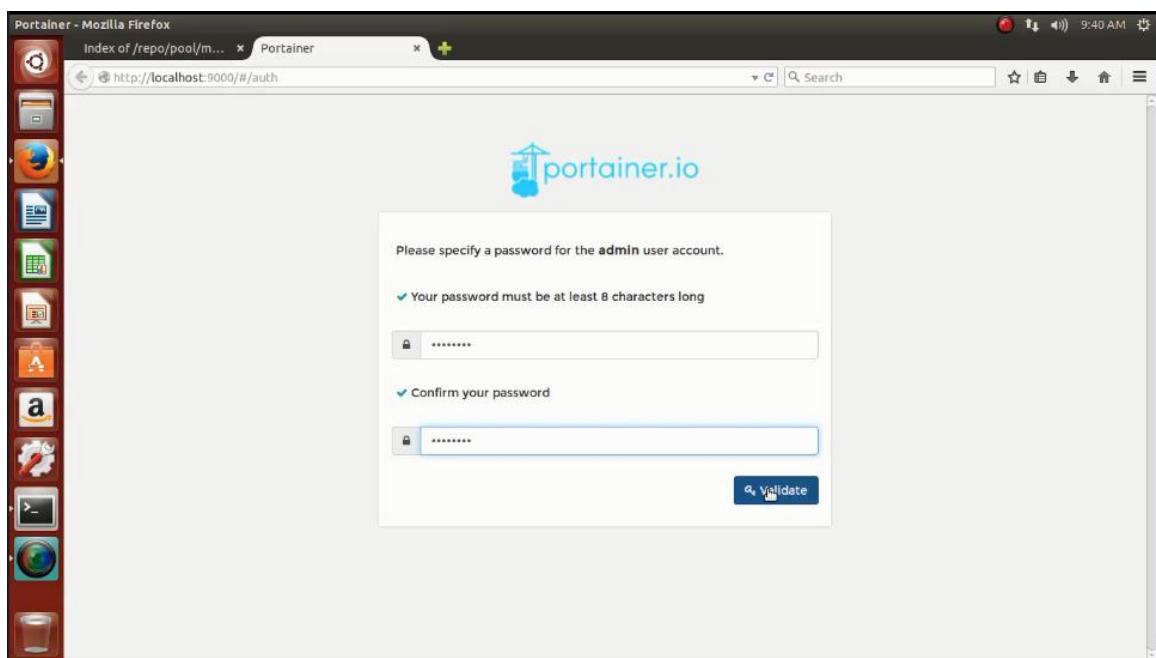
```
File Edit View Search Terminal Help
docker@ubuntu:~$ docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer
be94902babec0a21b7d0329b2271141620eadeffc1559c75c3b38aeb275e9c31
docker@ubuntu:~$
```

5. Buka browser anda. Lalu ketikkan alamat berikut pada address bar :

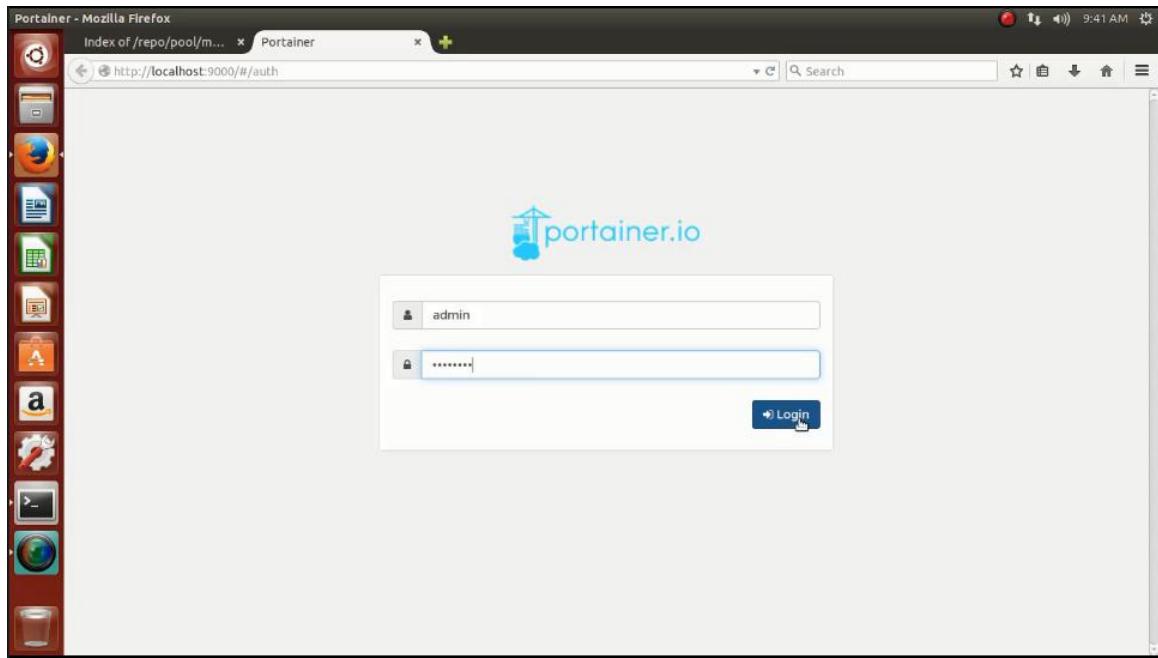
➔ <http://localhost:9000> atau http://<IP_Ubuntu>:9000



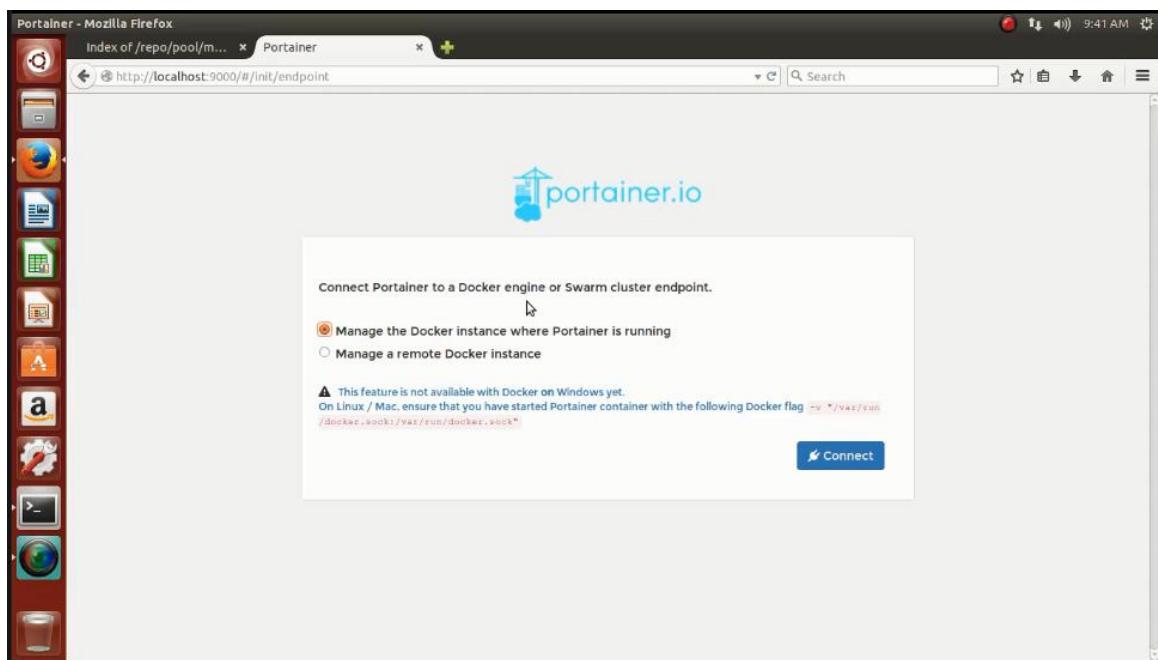
- Setelah itu akan muncul tampilan web portainer dan ketikkan password baru untuk login sebagai admin. **Minimal 8 karakter**. Kalau sudah langsung klik **validate**.



- Selanjutnya masuk ke tampilan login web portainer. Masukkan password yang telah dibuat di tahap sebelumnya → klik login.



8. Pilih yang "Manage the docker instance where Portainer is Running"



9. Apabila muncul tulisan seperti berikut ini masukkan perintah yang tertera dan buka kembali terminal.

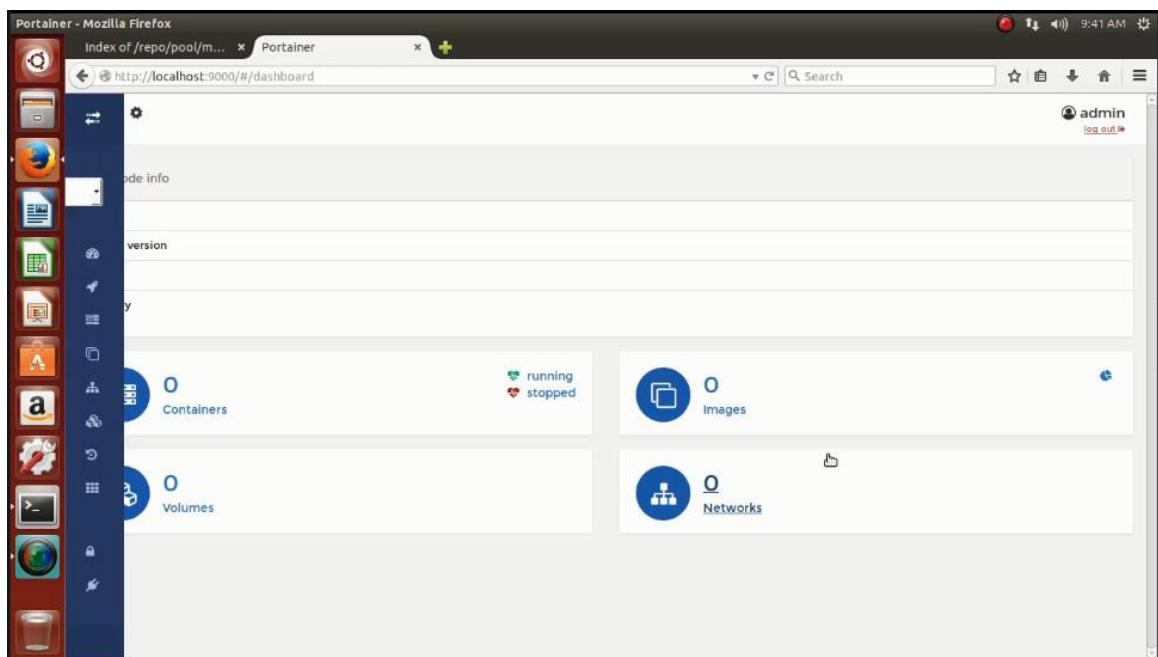
**⚠ This feature is not available with Docker on Windows yet.
On Linux / Mac, ensure that you have started Portainer container with the following Docker flag -v /var/run/docker.sock:/var/run/docker.sock**

10. Kemudian ketikkan perintah berikut:

```
$ docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock
```

```
docker@ubuntu:~  
docker@ubuntu:~$ docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock  
"docker run" requires at least 1 argument(s).  
See 'docker run --help'.  
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]  
Run a command in a new container  
dockergubuntu:~$
```

11. Setelah itu muncul tampilan web portainer



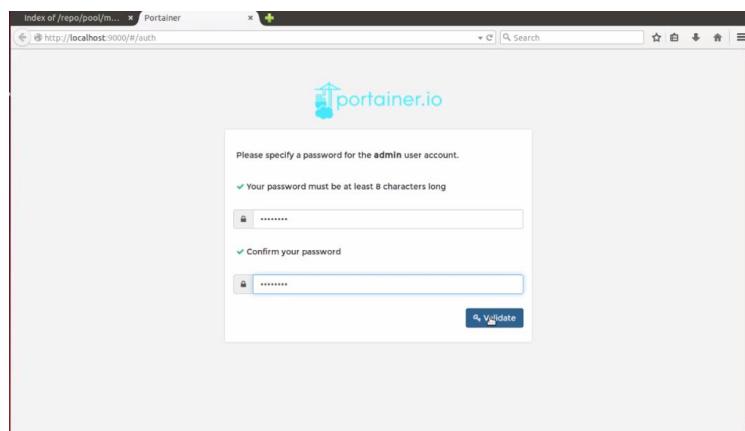
BAB VIII

Mengakses portainer

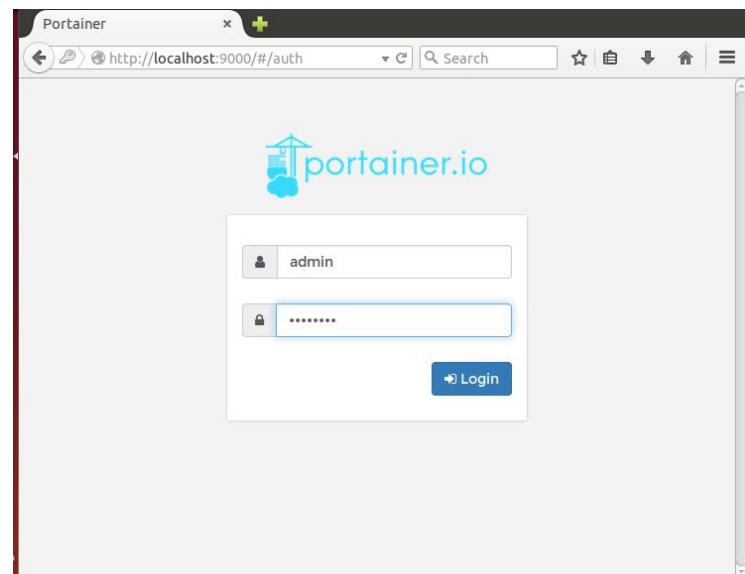
- Setelah portainer diinstall dalam sistem ubuntu. Untuk menjalankan portainer menggunakan perintah:

```
sudo docker run -d -p 9000:9000 -v /var/run/docker.sock:/var/run/docker.sock
portainer/portainer
```

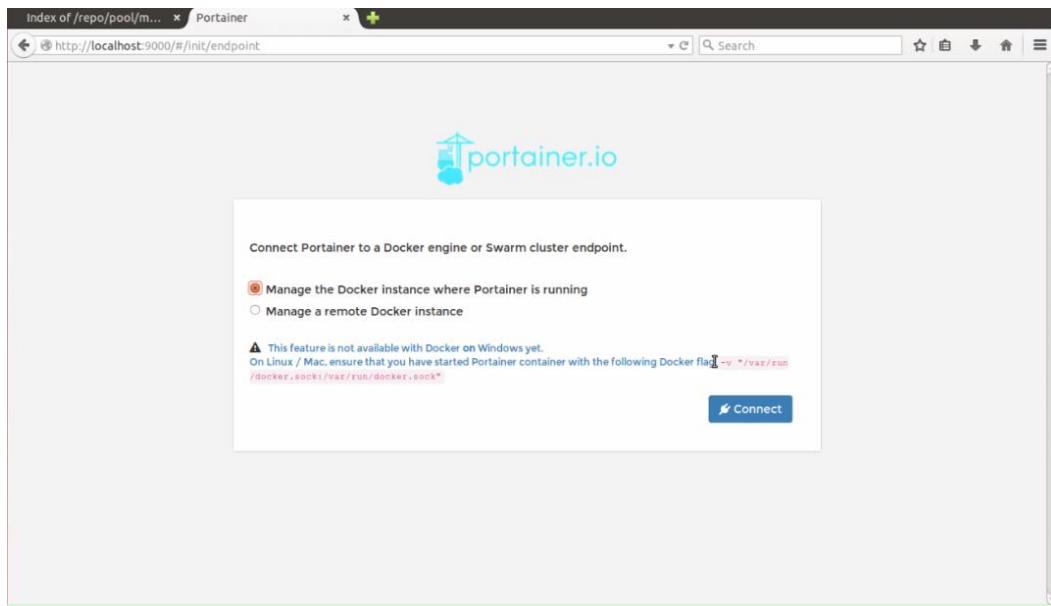
- Portainer telah diaktifkan, untuk mengakses portainer buka web browser dan masukkan alamat <http://localhost:9000> atau http://IP_Address:9000. Tampil pada web browser seperti pada gambar berikut, silakan setup password untuk user admin. Masukkan password dua kali dan tekan tombol *Validate* untuk membuat password user admin.



- pada langkah selanjutnya masukkan password yang telah anda buat sebelumnya lalu login.



- Connect Portainer ke mesin Docker atau Swarm klaster endpoint. Karena saya tidak memiliki mesin Docker pada perangkat lain, maka saya memilih "**Manage the Docker instance where Portainer is running**".



5. Setelah terhubung, Anda akan disajikan dengan dashboard intuitif seperti di bawah ini.

A screenshot of the Portainer dashboard. The URL is http://localhost:9000/#/dashboard. The dashboard has a dark blue sidebar on the left with various navigation links: ACTIVE ENDPOINT (local), ENDPOINT ACTIONS (Dashboard, App Templates, Containers, Images, Networks, Volumes, Events, Docker), and PORTAINER SETTINGS (Password, Endpoints). The main area shows "Node info" for "ubuntu" (Docker version 1.13.0, CPU 1, Memory 1GB). Below this are four cards: "Containers" (2 running, 0 stopped), "Images" (2), "Volumes" (18 aufs driver), and "Networks" (3).

BAB IX

Menagemen Images Pada Docker Portainer

Docker images adalah sebuah template yang bersifat read only. Template ini sebenarnya adalah sebuah OS atau OS yang telah diinstall berbagai aplikasi. Docker images berfungsi untuk membuat docker container, dengan hanya 1 docker images kita dapat membuat banyak docker container. Pada pembahasan kali ini penulis akan melakukan managemen pada Images yaitu pull Images menggunakan docker portainer.

Buka halaman portainer “<http://localhost:9000>” atau “http://IP_ADDRESS:9000”.

The screenshot shows the Portainer.io dashboard for a local endpoint. On the left sidebar, under the 'Images' section, there is a summary of resources: 1 Container (1 running, 0 stopped), 2 Images (9.2 MB total), 12 Volumes (aufs driver), and 3 Networks. The main panel displays node information for 'docker-VirtualBox' with Docker version 1.15.0, 1 CPU, and 2.1 GB Memory.

Buka menu Images pada halaman Portainer.io maka akan terlihat Images portainer yang tersimpan pada ubuntu, Mengapa ada Images portainer dalam menu Portainer? Images portainer ini ada dikarenakan proses “run” dan “pull” pada portainer disaat penambahan portainer pada docker.

The screenshot shows the 'Image list' page in Portainer. The 'Pull image' section has a 'Name' field set to 'e.g. ubuntu:trusty'. Below it, a table lists a single image entry:

ID	Tags	Size	Created
sha256:d449426bc5...	portainer/portainer:latest	9.2 MB	2017-02-01 05:18:51

Kolom bagian “Name” merupakan tempat untuk menentukan jenis Images yang akan kita “pull” baik itu Aplikasi, maupun Sistem Operasi. Uniknya dalam kolom “Name” ini kita dapat menentukan versi yang digunakan dalam Aplikasi, atau Sistem Operasi itu sendiri, dengan cara menggunakan tag “:”. Contohnya jika kita ingin menginstall Sistem Operasi

Ubuntu versi 14.04 maka kita tuliskan “ubuntu:14.04” dsb. Namun jika dalam kolom “Name” ini tidak diberi versi atau tag maka portainer akan melakukan “pull” pada versi yang terakhir.

Pada kolom “Registry” merupakan tempat untuk memilih server untuk melakukan “pull” Images, jika kolom “Registry” ini dikosongkan maka portainer akan otomatis mengakses server docker registry.

Penulis akan melakukan “pull” Ubuntu versi 14.04 dengan registry default. Isi kolom Name dengan “ubuntu:14.04” lalu tekan tombol “pull”.

The screenshot shows the Portainer.io interface for managing Docker images. On the left, there's a sidebar with navigation links like Dashboard, App Templates, Containers, Images (which is currently selected), Networks, Volumes, Events, and Docker. The main area has a title 'Image list' and a sub-section 'Pull image'. In the 'Name' field, 'ubuntu:14.04' is typed. A note below says 'Note: if you don't specify the tag in the image name, latest will be used.' Below the input fields is a large red 'Pull' button. To the right of the pull section is a table titled 'Images' with columns: Id, Tags, Size, and Created. It lists two entries: one with Id 'sha256:d449426bc5...' and Tags 'portainer/portainer:latest', and another with Id 'sha256:b969ab9f92...' and Tags 'ubuntu:14.04'. At the bottom right of the table, there are buttons for 'Items per page' (set to 10) and 'Filter...'. The top right corner shows the user 'admin' and a 'log out' link.

Jika sudah maka akan terlihat jumlah images pun bertambah dan kita telah berhasil melakukan “pull” dari server registry docker.

This screenshot shows the same Portainer.io interface as the previous one, but with a different set of images listed. The 'Images' table now shows three entries. The first entry is identical to the one in the previous screenshot: Id 'sha256:d449426bc5...' with Tags 'portainer/portainer:latest'. The second entry has Id 'sha256:b969ab9f92...' and Tags 'ubuntu:14.04'. The third entry has Id 'sha256:449426bc5...' and Tags 'portainer/portainer:latest'. The rest of the interface, including the sidebar and the 'Pull image' dialog, remains the same.

BAB X

Managemen Container Pada Docker Portainer

Docker container sendiri merupakan sebuah image yang dapat dikemas dan dibaca tulis, container berjalan diatas image. Pada setiap perubahan yang disimpan pada container akan menyebabkan terbentuknya layer baru di atas base image. Kita dapat melakukan instalasi aplikasi didalamnya dan melakukan penyimpanan.

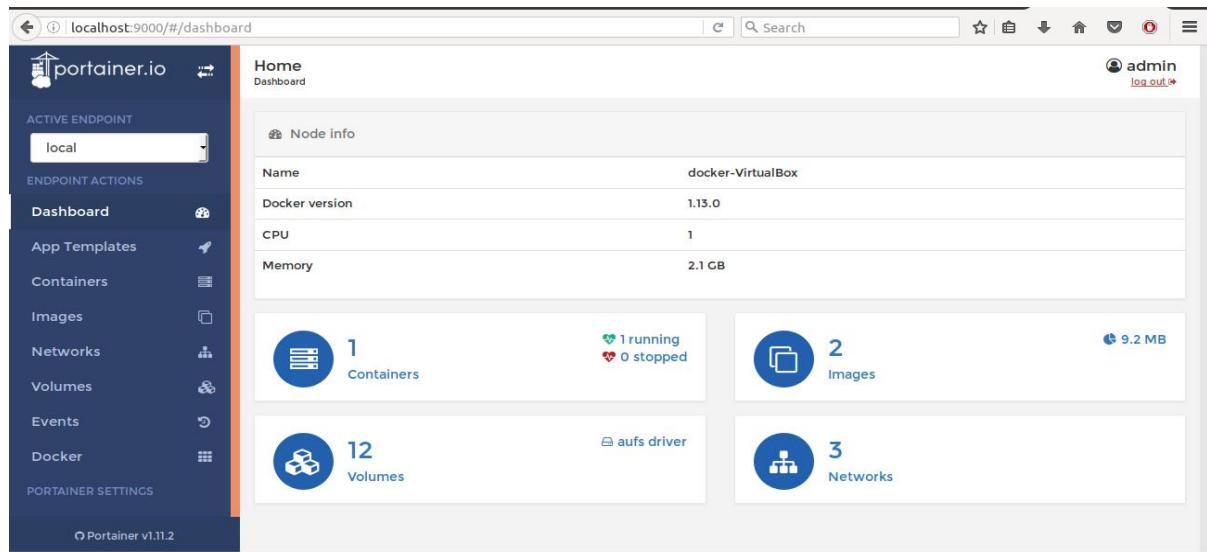
Dalam penerapannya managemen kontainer menjadi lebih mudah dengan menggunakan portainer, karena portainer menggunakan sistem berbasis gui (graphic user

interface) yaitu antarmuka pada sistem aplikasi yang menggunakan tampilan grafis pada web browser.

Terdapat beberapa cara untuk menambahkan container baru pada portainer, disini penulis akan menjelaskan pembuatan kontainer dengan beberapa metode tersebut.

1. Menambahkan container pada menu container

Buka halaman portainer “<http://localhost:9000>” atau “http://IP_ADDRESS:9000”.



Buka menu containers pada halaman portainer, dan akan terlihat container yang sedang aktif dan running. Selain dapat melihat container yang sedang aktif dan running pada menu container terdapat beberapa fitur dengan fungsi masing - masing yaitu :

Start	Menjalankan container yang telah dibuat
Stop	Menghentikan container yang sedang berjalan
Kill	Menutup container yang sedang aktif ataupun tidak
Restart	Merestart container
Pause	Memberhentikan container yang sedang aktif
Resume	Melanjutkan container yang sedang dipause
Remove	Menghapus container yang ada dalam list container
Add Container	Menambahkan container, atau membuat container

The screenshot shows the Portainer.io web interface. On the left, there's a sidebar with various menu items: ACTIVE ENDPOINT (local), ENDPOINT ACTIONS (Dashboard, App Templates, Containers, Images, Networks, Volumes, Events, Docker), and PORTAINER SETTINGS (Portainer v1.11.2). The main area is titled "Container list" and shows a table of containers. The table has columns for State, Name, Image, IP Address, and Published Ports. One row is visible, showing a green "running" status for the container named "suspicious_bhaskara", which uses the image "portainer/portainer" and is accessible via IP 172.17.0.2 on port 9000.

Selanjutnya tekan “ADD CONTAINER” yang terdapat pada menu bar.

This screenshot is similar to the one above, showing the Portainer.io interface with the "Containers" menu item selected in the sidebar. The "Add container" button in the top right of the main container list area is highlighted with a black rectangle.

Disini penulis ingin menambahkan container yang berisi images ubuntu. Maka selanjutnya isi pada kolom :

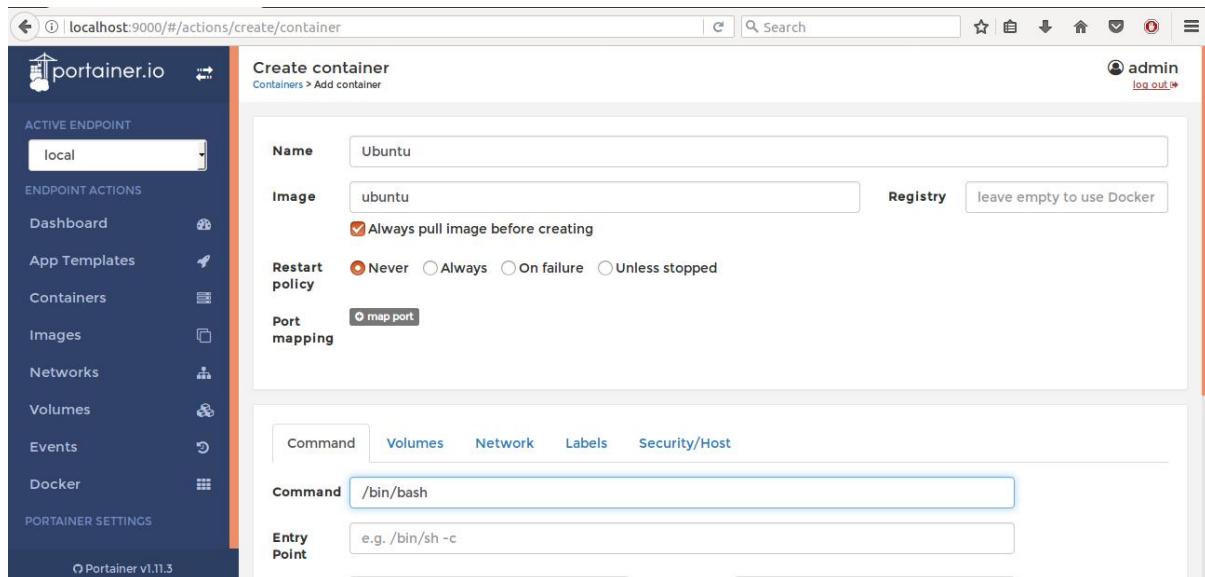
- ➔ Name = Ubuntu
- ➔ Image = ubuntu.
- ➔ command = /bin/bash.

Dengan menuliskan “ubuntu” pada kolom image maka container pun akan terisi dengan image ubuntu versi terakhir, namun jika ingin menggunakan versi tertentu semisal “ubuntu 14.04”, maka isi kolom tersebut dengan menggunakan tag, atau dapat disimbolkan dengan “:”, contoh :

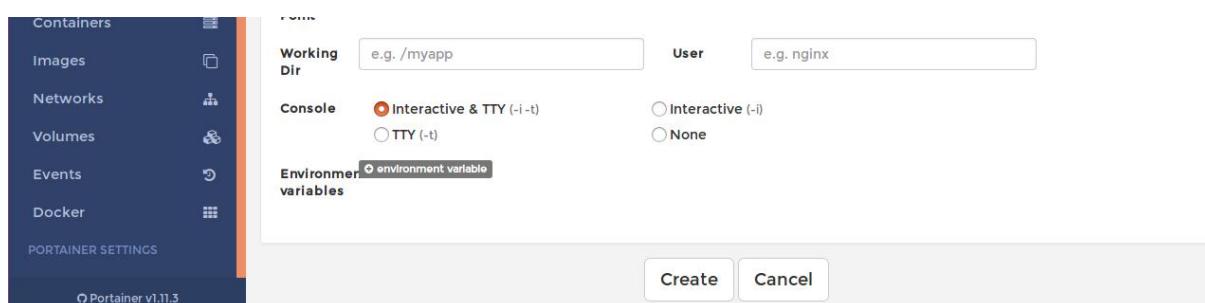
- ➔ Image = ubuntu:14.04.

Pada kolom imagepun dapat di isi nama aplikasi yang ingin dibuat dalam bentuk container, contohnya seperti aplikasi mysql, nginx, apache, dan lain lain. Namun ada beberapa aplikasi yang sudah disediakan pada menu “App Templates” sehingga lebih praktis untuk membuat suatu container yang hanya berisikan aplikasi. Dalam aplikasi kondisi tag “:” untuk menentukan versi aplikasi pun berlaku.

Dengan menceklis Always pull image before creating maka saat membuat container portainer akan melakukan pull dari server. Dan karna registry tidak diisi maka portainer akan meminta pull images melalui alamat docker registry resmi milik docker.



Setelah itu pada menu console centang pada radiobutton interactive and TTY. Selanjutnya tekan create untuk memulai pembuatan container yang terdapat images ubuntu, pembuatan container akan memakan waktu dikarnakan melakukan pull dari server.



Jika sudah maka container akan bertambah 1 dengan image “ubuntu:latest” dan memiliki nama container yaitu “ubuntu”, jika container ini tidak diberi nama maka portainer akan memberikan nama secara otomatis. Setelah container terbuat maka container akan otomatis running. Container sendiri pun memiliki sebuah IP, dimana IP tersebut merupakan IP yang di “bridge” kan dengan IP docker yang berada pada ubuntu.

State	Name	Image	IP Address	Published Ports
running	Ubuntu	ubuntu:latest	172.17.0.3	-
running	practical_habit	portainer/portainer	172.17.0.2	9000:9000

Jika ingin menghentikan container maka centang pada checkbox container, dan klik stop pada menu bar container, dan begitu pula jika ingin menggunakan beberapa menu lain seperti kill, restart, pause, resume, dan remove. Namun jika ingin meremove suatu container pastikan container dalam keadaan stopped.

State	Name	Image	IP Address	Published Ports
<input checked="" type="checkbox"/>	Ubuntu	ubuntu:latest	172.17.0.3	-
<input type="checkbox"/>	practical_habit	portainer/portainer	172.17.0.2	9000:9000

Untuk mengakses container yang berisi images ubuntu kita dapat mengaksesnya menggunakan terminal dengan menggunakan perintah command line:

➔ sudo docker attach "NAMA CONTAINER".

```
root@300c187b2fcc: /  
docker@docker-VirtualBox:~$ docker attach Ubuntu  
^C  
root@300c187b2fcc:/# ls  
bin dev home lib64 mnt proc run srv tmp var  
boot etc lib media opt root sbin sys usr  
root@300c187b2fcc:/#
```

Ketika mengakses images ubuntu, terlihat jelas bahwa terdapat perbedaan pada nama user root yang seharusnya adalah "docker" berubah menjadi id yang terdapat pada container yaitu "300c187b2fcc".

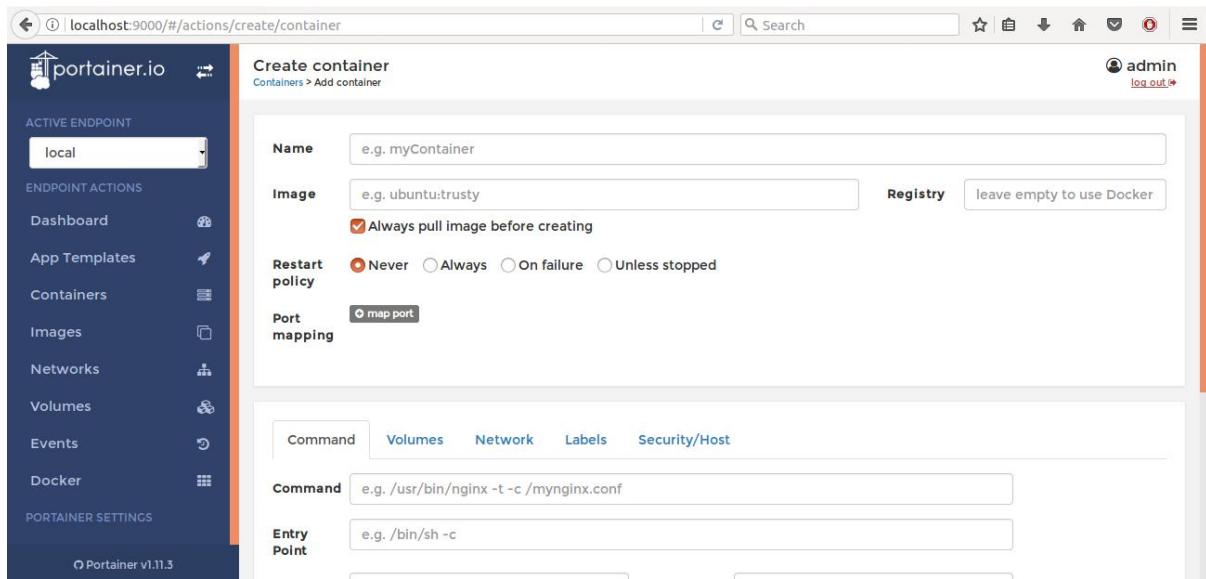
2. Pembuatan container menggunakan Images yang tersimpan pada ubuntu.

Pada BAB sebelumnya kita telah membahas bagaimana cara melakukan pull Images pada menu Images dalam portainer, dan Images berfungsi untuk membuat docker container dan dengan 1 docker Images kita dapat membuat berbagai macam docker container.

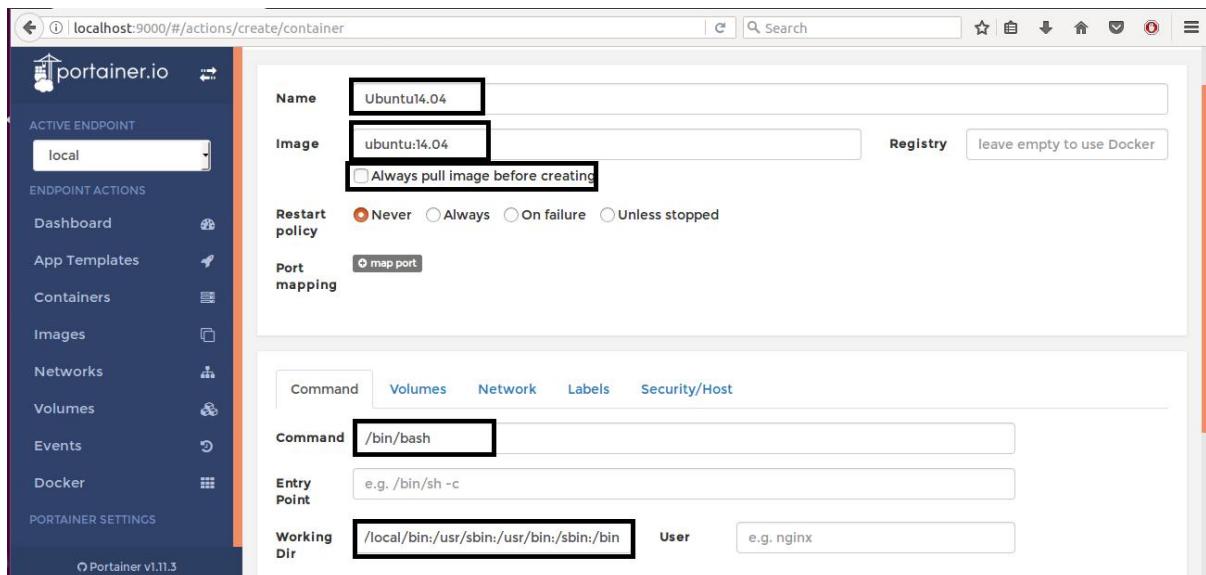
Buka menu Images pada halaman Portainer maka akan terlihat beberapa Images yang sudah kita “pull” atau “download” pada BAB sebelumnya, maupun pada metode pertama.

Klik Id pada Images Ubuntu 14.04 yang merupakan Images yang telah kita pull dari bab sebelumnya, lalu scroll kebawah hingga terdapat konten dari halaman yang menjelaskan informasi tentang PATH atau lokasi dari pada dockerfile itu sendiri.

Salin lokasi dockerfile tersebut dan buka menu Container kembali, dan klik Add Container.



Isi bebas kolom Name, pada kolom Image isi dengan “ubuntu:14.04”, pada kolom Command isi dengan “/bin/bash”, tempel lokasi yang tadi kita salin pada kolom Working Dir, dan jangan lupa di uncentang pada “Always Pull Image before creating” karna kita akan menggunakan Images yang sudah kita “pull” sebelumnya.



Sebelum menekan tombol “create” jangan lupa untuk memilih “Interactive and TTY” pada opsi “Console”. Setelah itu create.

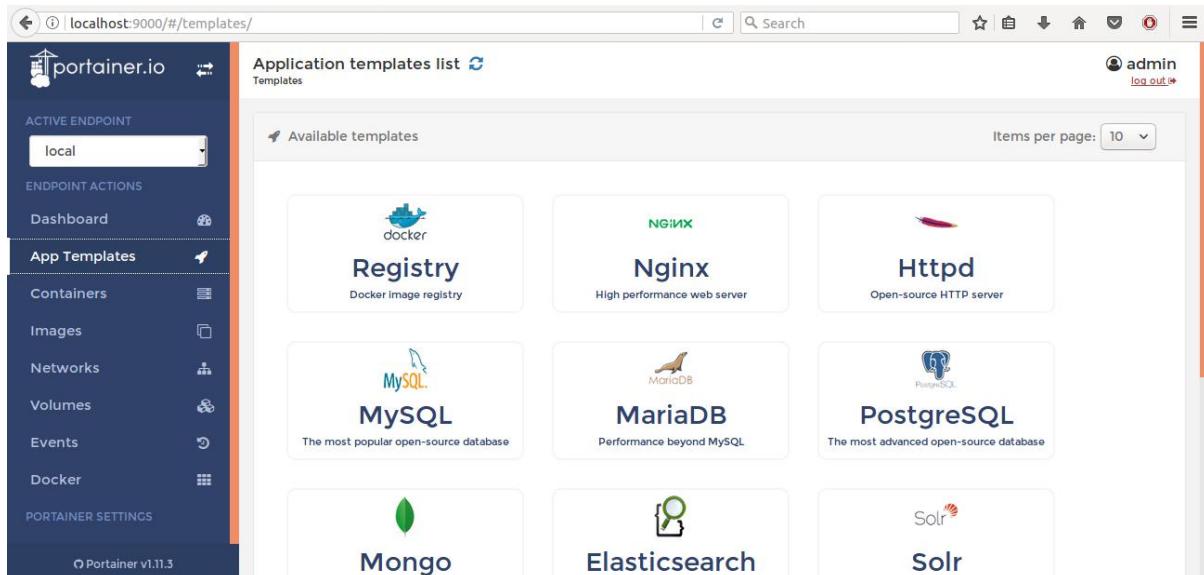


Setelah itu maka akan terlihat bahwa bertambahnya 1 container yang aktif, dengan nama Ubuntu14.04

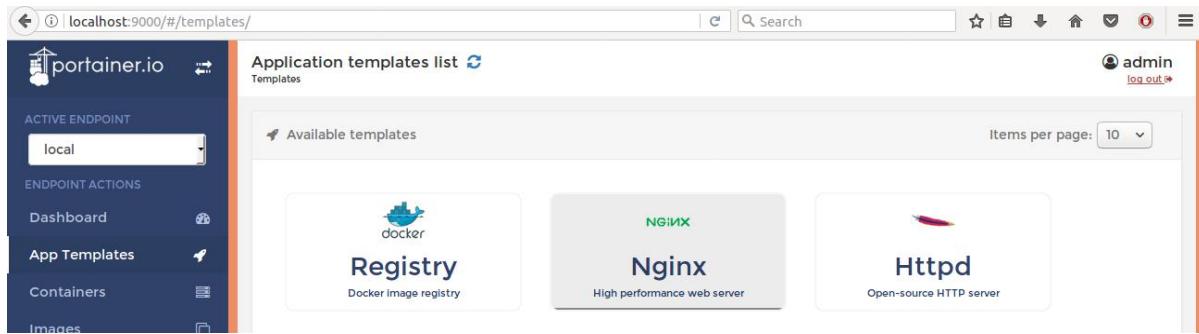


3. Pembuatan container melalui menu “App Templates”.

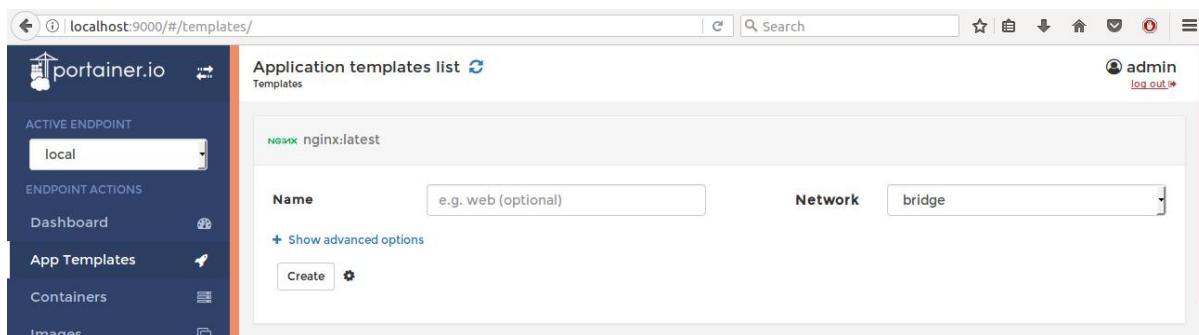
Pada menu App Templates terdapat beberapa images yang sudah disediakan oleh portainer itu sendiri, namun App Templates hanya menyediakan images berupa aplikasi. Untuk melakukan penambahan container pada menu App Templates, maka klik menu App Templates.



Pilih aplikasi yang diinginkan, disini penulis akan menggunakan aplikasi nginx.



Selanjutnya maka akan tampil menu seperti dibawah ini. Kolom Name berfungsi memberikan nama untuk container dan Network berfungsi untuk menentukan jenis Network apa yang ingin digunakan pada aplikasi tersebut.



Selanjutnya beri nama pada kolom Name, disini penulis akan memberikan nama "NGINX", dengan kondisi Network default. Lalu setelah itu tekan tombol create. Proses ini akan memakan waktu yang cukup lama karena sepertinya halnya image ubuntu, aplikasi NGINX pun akan melakukan proses pull kepada docker registry.



Jika semua proses sudah selesai maka akan terlihat container "NGINX" sudah terbuat dan berjalan.

The screenshot shows the Portainer.io web interface. On the left, there's a sidebar with 'ACTIVE ENDPOINT' set to 'local'. Under 'ENDPOINT ACTIONS', 'Containers' is selected. The main area is titled 'Container list' and shows a table of three running containers:

State	Name	Image	IP Address	Published Ports
running	NGINX	nginx:latest	172.17.0.4	:32769:443 → 32770:80
running	Ubuntu	ubuntu:latest	172.17.0.3	-
running	practical_habit	portainer/portainer	172.17.0.2	:9000:9000

Untuk melakukan ujicoba terhadap aplikasi NGINX, dapat dilakukan dengan membuka web browser dengan menggunakan IP pada NGINX dan portnya. "http://IP_NGINX:80".

The screenshot shows a web browser window with the URL '172.17.0.4:80'. The page displays the standard NGINX 'Welcome to nginx!' welcome message.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working.
Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Selain mengakses container yang telah kita buat menggunakan PC ubuntu yang terinstall docker kita akan mencoba mengakses melalui PC client yang menggunakan Operating System Windows. Namun agar client dapat mengakses server, pastikan bahwa PC server Ubuntu dengan PC client Windows berada dalam satu jaringan.

Setelah dalam satu jaringan coba lakukan test ping, antar kedua PC tersebut.

The screenshot shows a Windows Command Prompt window titled 'Administrator: Command Prompt'. The command 'ping 192.168.1.8' is run, and the output shows successful ping results:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Riki>ping 192.168.1.8

Pinging 192.168.1.8 with 32 bytes of data:
Reply from 192.168.1.8: bytes=32 time<1ms TTL=64

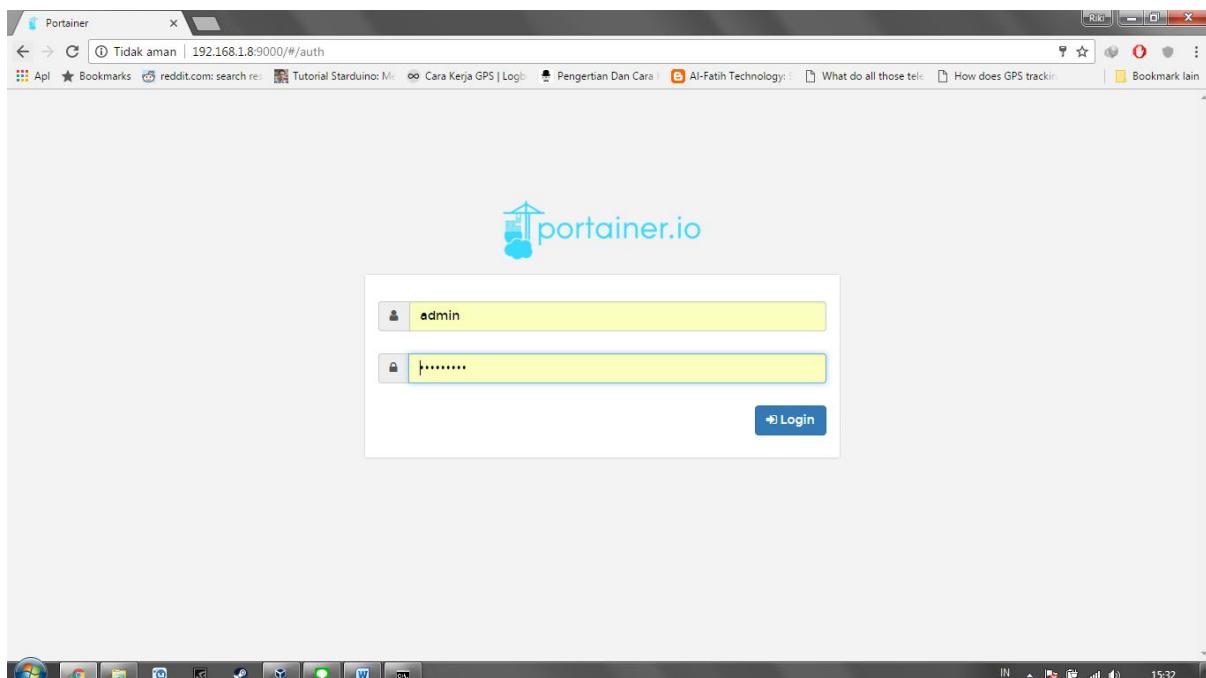
Ping statistics for 192.168.1.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\Riki>
```

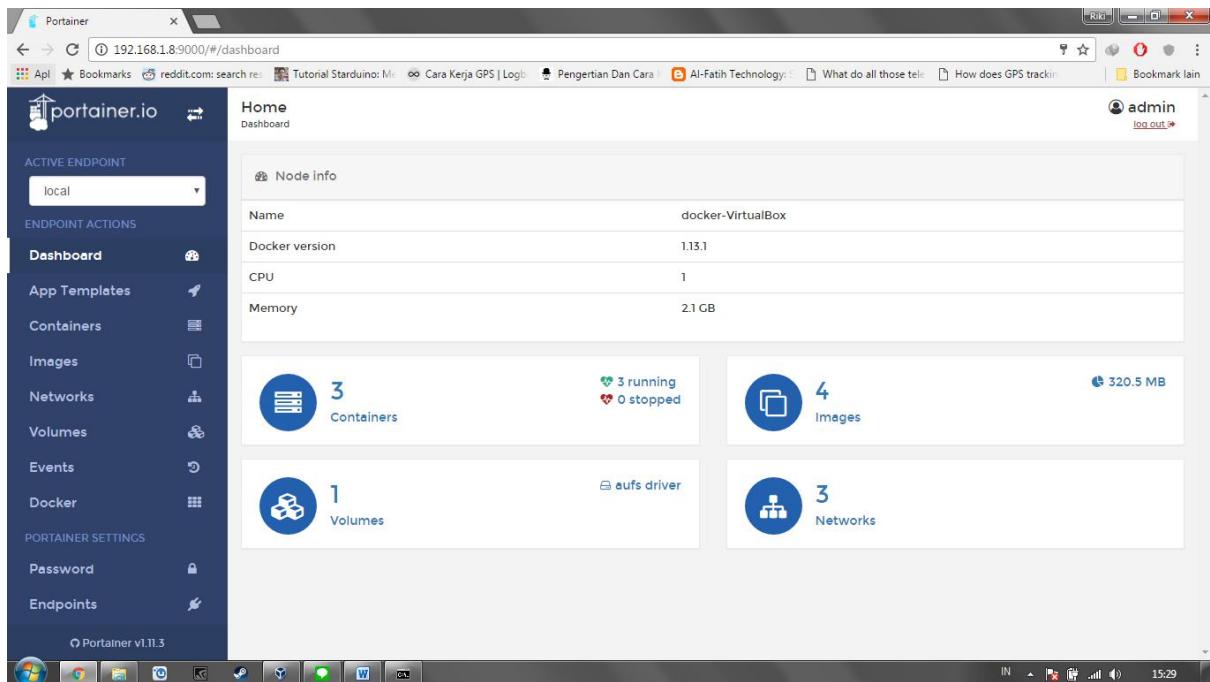
```
docker@docker-VirtualBox: ~
docker@docker-VirtualBox:~$ ping 192.168.1.13
PING 192.168.1.13 (192.168.1.13) 56(84) bytes of data.
64 bytes from 192.168.1.13: icmp_seq=1 ttl=128 time=0.290 ms
64 bytes from 192.168.1.13: icmp_seq=2 ttl=128 time=0.200 ms
64 bytes from 192.168.1.13: icmp_seq=3 ttl=128 time=0.264 ms
64 bytes from 192.168.1.13: icmp_seq=4 ttl=128 time=0.370 ms
^C
--- 192.168.1.13 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.200/0.281/0.370/0.060 ms
docker@docker-VirtualBox:~$
```

Disini pada Ubuntu Server menggunakan IP : **192.168.1.8**, dan Pada Windows Menggunakan IP : **192.168.1.13**.

Jika proses ping berhasil, maka pastikan docker dan container dalam keadaan aktif dan sedang berjalan. Karna kita menggunakan portainer maka langkah selanjutnya adalah buka halaman Portainer pada windows, dengan cara "<http://192.168.1.8:9000>" atau "<http://IP UBUNTU:9000>".

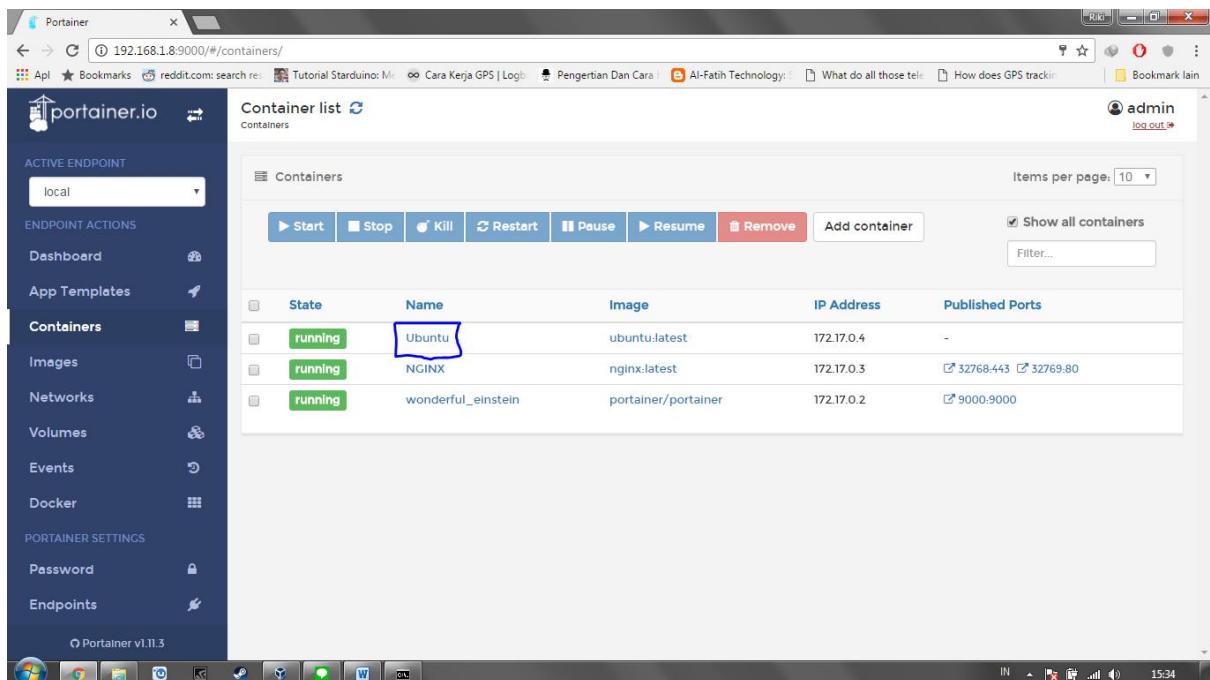


Login dengan password yang kita buat pada docker portainer pada ubuntu server.



Pada saat ini kita akan mengakses container ubuntu dan juga NGINX yang telah kita jadikan container pada docker server ubuntu. Pertama tama kita akan mengakses container ubuntu kita pada windows.

Buka menu container, lalu klik container Ubuntu yang sedang berjalan.



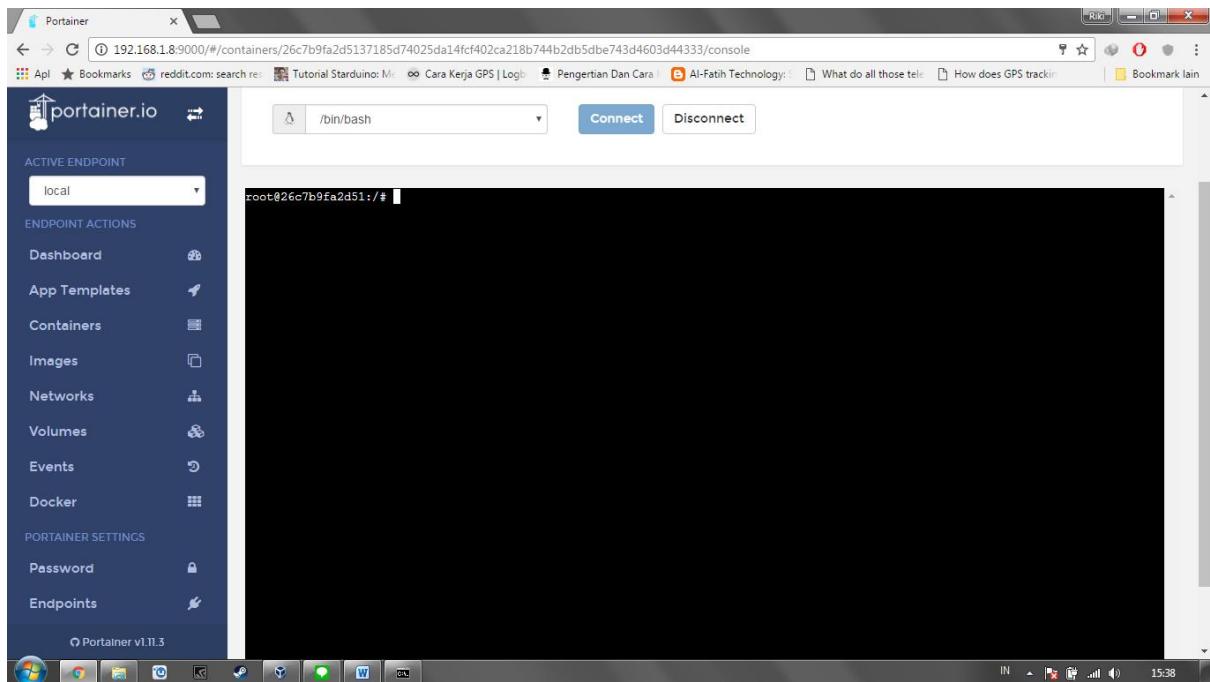
Klik console pada Menu container status.

The screenshot shows the Portainer web interface. On the left, a sidebar lists various Docker management options like Dashboard, App Templates, Containers, Images, Networks, Volumes, Events, Docker, and Portainer Settings. The 'Containers' option is selected. In the main panel, it says 'Container details' under 'Containers > Ubuntu'. It shows an 'ACTIVE ENDPOINT' dropdown set to 'local'. Below that is a row of redaction buttons: Stop, Kill, Restart, Pause, and Remove. A blue box highlights the 'Console' button. Under 'Container status', it lists the container's name as 'Ubuntu', IP address as '172.17.0.4', status as 'Running since 4 hours', and start time as '2017-02-14 11:29:43'. There are also 'Stats', 'Logs', and 'Console' tabs. A 'Create image' section allows creating an image from the container. At the bottom, there's a search bar for 'Name' and a 'Registry' dropdown.

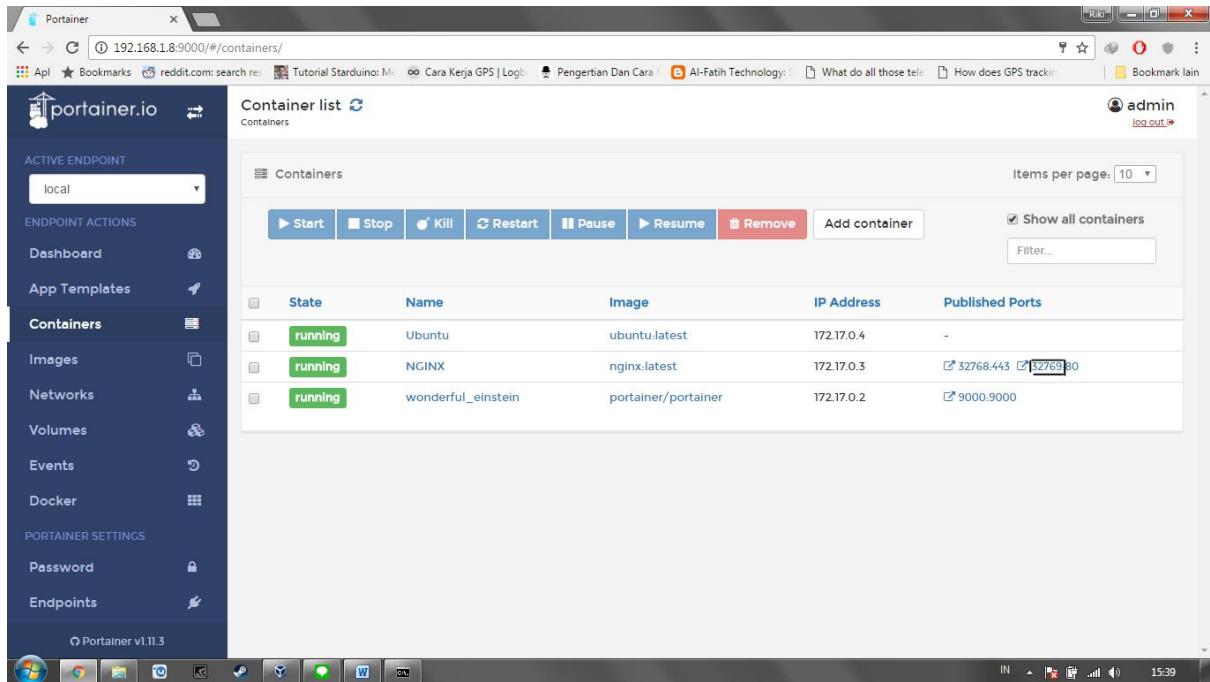
Selanjutnya klik connect dengan menggunakan /bin/bash. Agar kita mengakses perintah bash.

This screenshot shows the 'Container console' page for the same Ubuntu container. The sidebar and top navigation are identical to the previous screenshot. The main area is titled 'Container console' under 'Containers > Ubuntu > Console'. It features a 'Console' input field containing '/bin/bash' and a 'Connect' button, which is highlighted with a blue box. To the right of the input field are 'Disconnect' and other control buttons. The bottom of the screen shows a taskbar with various icons and system status indicators.

Kita telah berhasil mengakses container tersebut



Selanjutnya kita akan melakukan cek pada NGINX dengan mengakses container NGINX pada client Windows. Buka menu container dan lihat pada published port yang digunakan pada container tersebut.



Selanjutnya buka browser baru dengan alamat "<http://192.168.1.8:32769>" atau "<http://IP UBUNTU:PORT>"



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



Terimakasih.