

## LAB 1/1 - Ambiente do curso

### Objetivo

Nesse primeiro laboratório prático iremos configurar o ambiente de trabalho do curso, realizando a instalação e configuração dos softwares necessários.

Para o curso, será utilizado o seguinte ambiente:

- **Sistema Operacional:** Ubuntu 14.04 LTS
- **Softwares necessários:**
  - Simulador de rede: Mininet
  - *Switch* virtual: Open vSwitch
  - Container para execução do controlador: Docker
  - IDE de programação JAVA: IntelliJ

## Open vSwitch

### Instalação

O Open vSwitch é um *switch* virtual que implementa o protocolo OpenFlow. Algumas de suas principais características são:

- Implementado em software
- Plano de dados executa dentro do kernel do Linux
- Plano de controle executa em nível usuário
- Pode operar como um *switch* Ethernet padrão

*A utilização do Open vSwitch será detalhada em uma prática separada*

Para instalar o pacote execute:

```
#> sudo apt-get install -y openvswitch-switch
```

Para validar o funcionamento do Open vSwitch, execute:

```
#> sudo ovs-vsctl show  
b5c1bb5a-faf1-4aec-9ebb-c673848aeddb  
    ovs_version: "2.0.2"
```

Verifique se uma saída semelhante a acima é mostrada em seu terminal.

# Mininet

## Introdução

O Mininet é um emulador de redes que faz uso do recurso de *Network Namespaces* do *kernel* do Linux para prover o isolamento entre os *hosts* da rede emulada.

O suporte a redes OpenFlow é obtido com a utilização do Open vSwitch para a criação dos *switches* aos quais os *hosts* se conectam.

Por ser extremamente leve, ele é bastante escalável, permitindo a rápida prototipação de grandes redes virtuais.

*A utilização do Mininet será detalhada em uma prática separada*

## Instalação

Para instalar o Mininet

```
#> git clone git://github.com/mininet/mininet
#> mininet/util/install.sh -fnv
```

**Nota:** Outra opção seria instalar utilizando o **apt-get** mas, para o curso, iremos utilizar alguns recursos da versão mais nova, disponível no github do projeto.

Para testar o funcionamento do Mininet

```
#> sudo mn --test pingall --switch user --topo linear,2
```

Para testar a integração ao Open vSwitch

```
#> sudo mn --switch ovsk --topo single,3
... ..
*** Stopping 2 hosts
h1 h2
*** Done
completed in 0.797 seconds
```

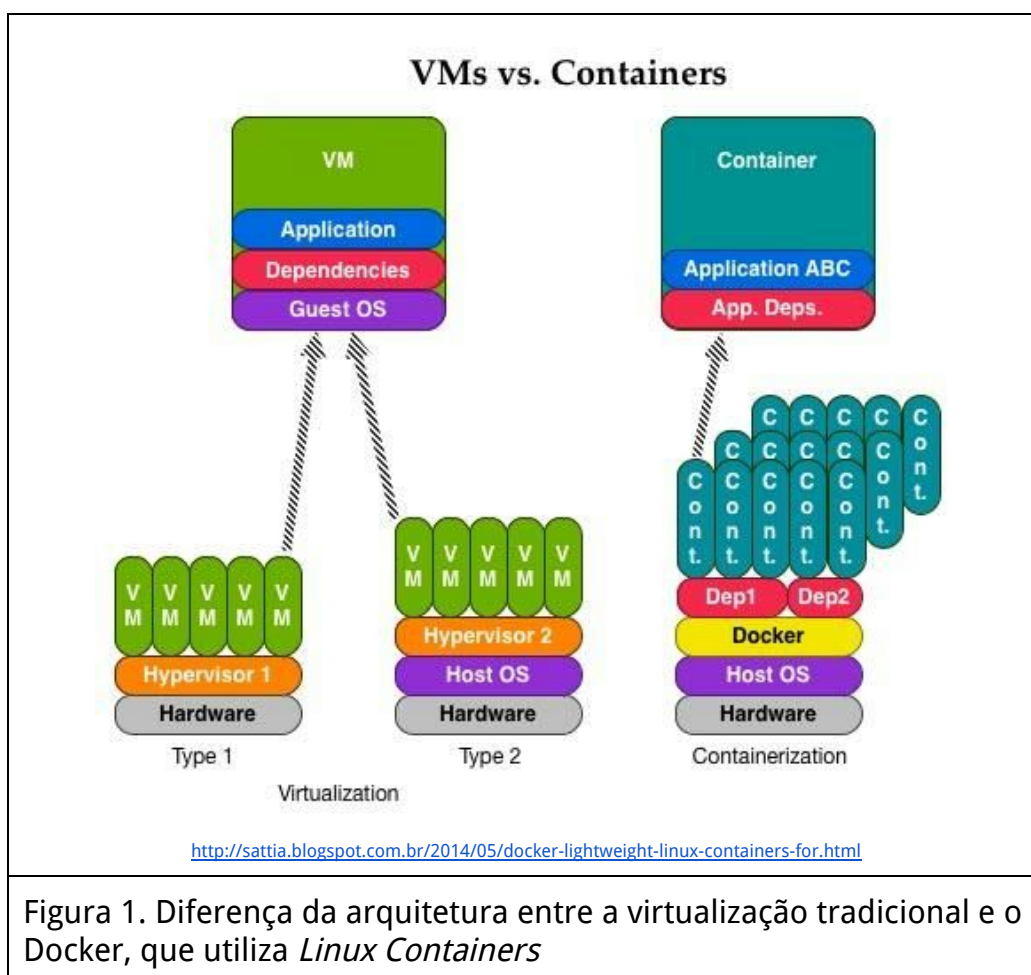
# Docker

## Introdução

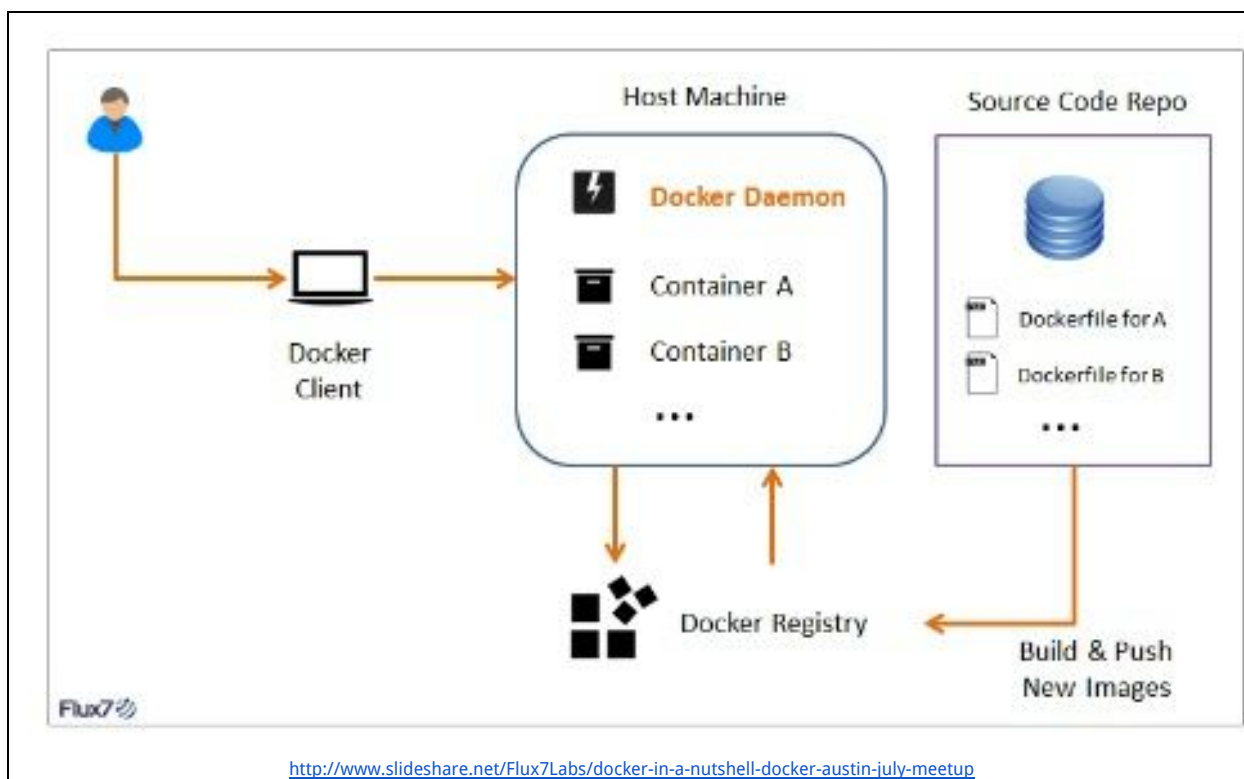
O Docker, uma proposta alternativa às máquinas virtuais convencionais, automatiza o uso de aplicações isoladas com o uso de containers (LXC), que é uma virtualização leve a nível de kernel. O kernel permite a execução de múltiplos espaços de usuário independentes e isolados.

Cada container é composto por uma imagem do SO incluindo o sistema de arquivos, bibliotecas e executáveis. Porém sua execução sofre pouco *overhead*, uma vez que o kernel é compartilhado entre os containers e as chamadas de sistema são realizadas diretamente, sem emulação ou o controle de um hypervisor.

Os containers podem ser iniciados, reiniciados e terminados em questão de segundos.



## Componentes do Docker



- **Docker client:** o binário `docker` é o principal cliente usado para interação do usuário com o sistema Docker;
- **Docker daemon:** monitor de execução permanentemente ativo no *host* que gerencia os containers e suas interações com as imagens e o *docker registry*;
- **Docker images:** um template somente leitura, que contém exatamente as configurações e o estado inicial quando um novo container é criado. No curso, iremos disponibilizar uma imagem de Ubuntu, com o controlador SDN já instalado e configurado
- **Docker registry:** repositório onde as imagens ficam armazenadas e podem ser recuperadas para um *host*;
- **Docker container:** um container consiste de um sistema operacional, os arquivos do usuário e meta dados que foram gerados a partir de uma imagem.

## Instalação

Adicionar a chave publica do repositório

```
#> sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80  
--recv-keys 58118E89F3A912897C070ADBF76221572C52609D
```

Criar source do repositório:

```
#> echo "deb https://apt.dockerproject.org/repo ubuntu-$(lsb_release  
-s -c) main" | sudo tee /etc/apt/sources.list.d/docker.list
```

Atualizar repositório:

```
#> sudo apt-get update
```

Instalar pacote:

```
#> sudo apt-get install -y docker-engine
```

## Utilização

Para listar os containers em execução e os já criados, execute:

```
#> docker ps -a
```

Para baixar uma imagem do repositório do Docker, você pode utilizar o comando abaixo. No entanto, já disponibilizamos a imagem localmente para evitar o tempo de *download* da imagem, necessitando apenas carregá-la (mostraremos a seguir):

```
* Nota: não é necessário executar esse comando *  
#> docker pull heitormotta/sci-onos:run
```

Para criar e iniciar um container a partir de uma imagem existente, execute:

```
* Nota: não é necessário executar esse comando *  
#> docker run -ti --name <nome_container> <imagem>
```

Para listar todas as meta informações do container:

```
* Nota: não é necessário executar esse comando *  
#> docker inspect <nome_container>
```

Como dito anteriormente, utilizaremos uma imagem local do container ONOS criado para o curso, por fins meramente de poupar o tempo gasto para baixar as imagens.

Importe a imagem com o comando **load**:

```
#> docker load -i docker-images/sci-onos.tar.gz
#> docker load -i docker-images/sci-onos-run.tar.gz
```

Verifique se imagem carrega está na lista:

```
#> docker images heitormotta/sci-onos
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
heitormotta/sci-onos	latest	ee9aa25f5783	2 weeks ago	756.4 MB
heitormotta/sci-onos	run	b8d0e6347090	2 days ago	839.7 MB

Crie um container a partir da imagem:

```
#> docker run -td -h teste1 --name teste1 heitormotta/sci-onos:run
```

No comando acima, criamos um container chamado teste1 (**--name**), setamos o *hostname* para teste1 (**-h**) e indicamos para que seja criado um container com o ponto de entrada sendo a execução de um *shell* bash (**--entrypoint**).

Podemos utilizar qualquer comando como *entrypoint* do container, como por exemplo o serviço de inicialização do ONOS (**onos-service**). No entanto, não poderíamos reiniciar o serviço do ONOS porque o container iria perder o ponto de entrada e ser finalizado. Por isso, escolhemos o *entrypoint* como */bin/bash*.


## Acessando o *container* do ONOS

Para acessar o container, é preciso iniciar um *shell* dentro dele:

```
#> docker exec -ti teste1 bash
root@teste1:~/onos#
```

Para executar o serviço do ONOS no container e acessar a CLI do controlador de dentro do *shell* do *container*, utilizamos o comando **onos-service**:

```
teste1#> /root/onos/bin/onos-service
Welcome to Open Network Operating System (ONOS)!



Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown ONOS.
```

Com o serviço em execução, podemos acessar um novo terminal na CLI do controlador, usando o SSH. Para tanto, primeiramente precisamos obter o IP do *container* com um dos comandos abaixo.

```
#> docker inspect teste1 | grep -w IPAddress
#> docker inspect --format '{{.NetworkSettings.IPAddress }}' teste1
```

Para realizar a conexão ssh, utilize o IP obtido e “karaf” como usuário/senha:

```
ssh karaf@<IPAddress> -p 8101
```

Finalize a execução do controlador utilizando o comando **shutdown** na CLI do controlador ou apertando as teclas **CTRL+D**

```
onos> shutdown
Confirm: halt instance root (yes/no): yes
```

Por fim, remova o container de teste criado para o aprendizado do docker.

```
#> docker stop teste1
#> docker rm teste1
```

## Criando o container para as práticas iniciais do curso

Após termos aprendido a criar containers no docker, vamos criar o container, **onos1**, que será utilizado nas práticas do curso.

```
#> docker run -td -p 8181:8181 -p 6633:6633 -h onos1 --name onos1  
heitormotta/sci-onos:run
```

No comando acima, o parâmetro “-p” serve para redirecionar as portas no container criado para a máquina local, de forma que possamos acessar a interface WEB do container pelo endereço **localhost:8181**, ao invés de utilizarmos o IP privado do container (172.17.0.x/16).

Verifique se o container **onos1** foi criado com sucesso, através do comando:

```
#> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
66cde4ff4947	heitormotta/sci-onos:run	"/bin/bash"	About a minute ago Up
About a minute	5005/tcp, 0.0.0.0:6633->6633/tcp, 8101/tcp, 0.0.0.0:8181->8181/tcp, 9876/tcp	onos1	

Outros comandos úteis do Docker podem ser visualizados nas URLs abaixo:

<https://coderwall.com/p/2es5jw/docker-cheat-sheet-with-examples>

<https://github.com/wsargent/docker-cheat-sheet>

## IntelliJ (IDE)

O IntelliJ é uma IDE de desenvolvimento JAVA bastante popular, que é a linguagem do controlador ONOS.

Usaremos o IntelliJ para a implementação e depuração das aplicações para o ONOS desenvolvidas no curso. Por simplicidade, as máquinas do curso já possuem uma versão do IntelliJ instalada, que pode ser acessado pelo ícone **IntelliJ** no Desktop.

A versão *community* do IntelliJ está disponível gratuitamente em:

<https://www.jetbrains.com/idea/download/>