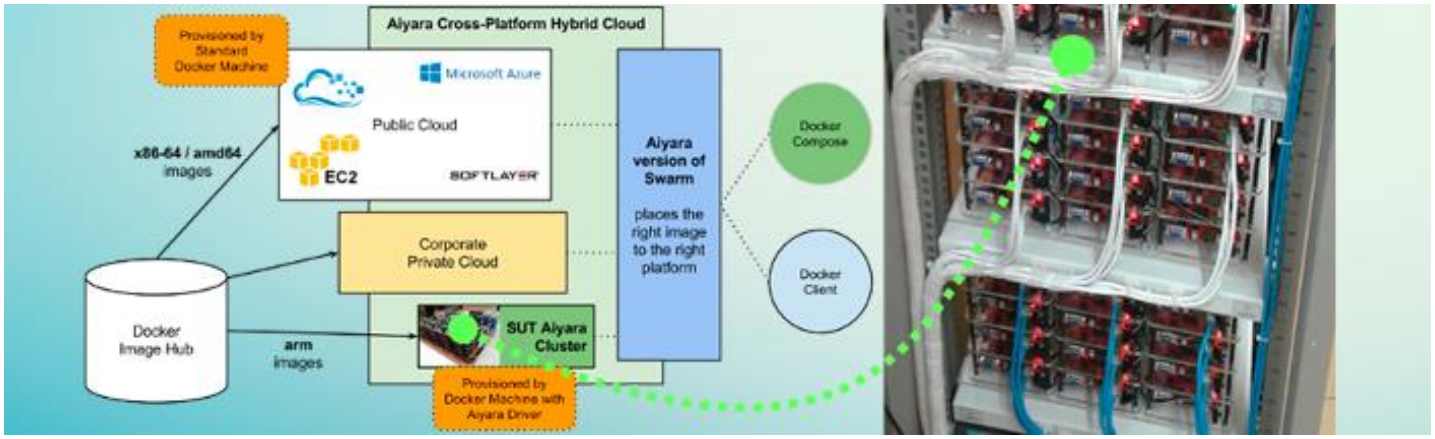

Nuvem Híbrida e Docker

Com o conjunto de ferramentas Docker, configuramos uma nuvem híbrida com facilidade. Docker Machine nos ajuda a criar e gerenciar máquinas container-centric. Ele vem com um grande conjunto de drivers, cobrindo desde VirtualBox local para IaaS, como Amazon EC2 ou DigitalOcean. Docker Machine apenas faz com que instancie essas máquinas Docker-ready.

Docker Swarm nos ajuda a formar um cluster de instâncias Docker, que estão provisionados por Docker Machine. Não importa onde estão em execução. Se seus endereços IP são acessíveis, o Swarm pode formá-los para ser um host Docker único virtual. Em seguida, ele permite que qualquer cliente Docker controle o cluster. Com máquina padrão e Swarm, já temos uma maneira fácil de construir uma nuvem híbrida baseada em Docker.

Apesar de rodar em qualquer lugar, uma limitação do Docker é que ele suporta oficialmente uma única arquitetura Linux / x86-64 (aka AMD64). Isso é compreensível já que o Linux / x86-64 é a plataforma principal.

Comece pequeno. Uma nuvem híbrida de 2 nodes.

Aqui está uma lista da nossa menor nuvem híbrida. Nós temos um node rodando em uma placa ARM, e outra é uma máquina de 512mb DigitalOcean. Um node local mestre extra está agindo como Swarm Master.

Docker Machine faz a manipulação fácil. Nós podemos apenas usar o config machine para fornecer toda a configuração necessária para o cliente Docker.

```
$ docker $(machine config master --swarm) info
Containers: 2
Strategy: spread
Filters: affinity, health, constraint, port, dependency
Nodes: 2
ocean-1: 128.199.108.67:2376
 └ Containers: 1
 └ Reserved CPUs: 0 / 1
 └ Reserved Memory: 0 B / 514.5 MiB
 └ Labels: executiondriver=native-0.2, kernelversion=3.13.0-43-generic, operatingsystem=Ubuntu 14.04.1 LTS, provider=digitalocean, storagedriver=aufs

rack-1-node-4: 192.168.1.4:2376
 └ Containers: 1
 └ Reserved CPUs: 0 / 2
 └ Reserved Memory: 0 B / 2.069 GiB
 └ Labels: architecture=arm, executiondriver=native-0.2, kernelversion=3.19.4, operatingsystem=Debian GNU/Linux 7 (wheezy), provider=aiyara, storagedriver=aufs
```

O comando Docker Info, acima, percorreu o Swarm Mestre. Ele mostrou que temos um cluster de 2 nodes. O primeiro provedor usa DigitalOcean do Docker Machine. A segunda é a nossa node Aiyara. Ambos usam AUFS como seu mecanismo de armazenamento.

Em seguida tentamos extrair as imagens do Debian para cada node antes de executá-lo.

```
$ docker $(machine config master --swarm) pull debian

rack-1-node-4: Pulling debian:latest...

ocean-1: Pulling debian:latest...

ocean-1: Pulling debian:latest... : downloaded

rack-1-node-4: Pulling debian:latest... : downloaded
```

Bem, Ocean-1 puxa claramente a imagem mais rápido. Em seguida, testamos a execução de um simples comando uname -a duas vezes por meio do Swarm.

Usamos a imagem Debian.

O Swarm escolheria um node para a primeira execução, em seguida, um outro node para a segunda execução, porque sua estratégia de agendamento padrão é o Spread, um algoritmo que vai colocar os recipientes o mais propagável possível.

```
$ docker $(machine config master --swarm) run debian uname -a

Linux e75d5877493e 3.19.4 #2 SMP Mon Apr 20 02:39:39 ICT 2015 armv7l GNU/Linux



$ docker $(machine config master --swarm) run debian uname -a

Linux 6d6b9d406f88 3.13.0-43-generic #72-Ubuntu SMP Mon Dec 8 19:35:06 UTC 2014
x86_64 GNU/Linux
```

Vamos ver o que está por trás disso. Executamos o ps -a para mostrar todos os recipientes dentro do cluster. Você vai notar que o nosso node ARM executou um recipiente com a imagem diferente do utilizado pelo node DigitalOcean.

O aiyara / debian: latest.arm é a nossa imagem Debian ARM.

Aiyara versão do Swarm é inteligente o suficiente para saber que vamos colocar um novo recipiente em uma máquina ARM, por isso escolheu a imagem específica da plataforma correta para nós.

```
$ docker $(machine config master --swarm) ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6d6b9d406f88	debian:latest	"uname -a"	14 minutes ago	Exited (0) 13 minutes ago	___	ocean-1/focused leakey
e75d5877493e	aiyara/debian:latest.arm	"uname -a"	14 minutes ago	Exited (0) 14 minutes ago	___	rack-1-node-4/modest_ritchie

Um 100-node Cluster

Bem, um cluster de 2 nodes é muito simples. Aqui está uma 100-Node-Cloud em ação.

Como já temos 50 nodes ARM em nosso cluster, criaremos outros 50 nodes em DigitalOcean. Aqui está o comando para criar um node DigitalOcean.

```
$ machine create -d digitalocean --digitalocean-access-token=token --digitalocean-region=sgp1
ocean-1
```

Executamos o comando acima para o node ocean-1 ao ocean-50. Nós usamos principalmente os parâmetros padrão, então nós temos VMs 512mb a partir de plano de R$ 15,00/mês.

Seria muito trabalhoso escalar número de nodes manualmente, mas o Docker Compose nos ajuda a lidar com isso facilmente. Nós criamos um diretório e o nomeamos de aiyara_cloud. Em seguida, nós colocamos o compose.yml, um arquivo de descrição para a nossa unidade de implantação.

Iniciaremos um servidor Nginx web em cada node, e ligaremos a porta 80 do servidor a porta exposta do recipiente. Aqui está a nossa descrição YML:

```
$ cat docker-compose.yml
web:
 image: nginx
 ports:
 - "80:80"
```

Começamos o primeiro node com o seguinte comando:

```
$ docker-compose up -d
```

Para dimensionar o servidor web para 100 nodes, usamos:

```
$ docker-compose scale web=100
```

Então, você vai ver que outros 99 recipientes seriam criados a partir de cada imagem específica da plataforma e colocada corretamente para seu hardware, um por um.

Aqui está a lista de todos os recipientes que funcionam ao longo do / cloud-based Docker x86-64 ARM híbrido.

Referência: cubieboard.org

Compartilhe e Divirta-se