

Cloud ML

Advanced Institute for Artificial Intelligence – AI2

<https://advancedinstitute.ai>

Agenda

- ☐ Origem e Motivação
- ☐ Modelo de Uso
- ☐ Estratégias de Uso
- ☐ Virtualização
- ☐ Docker

Grid Computing (computação em grade ou computação em malha) ¹

- ❑ Iniciativa pioneira para computação distribuída
- ❑ Grid é uma analogia a rede elétrica, poder computacional a disposição, no mesmo molde que energia elétrica
- ❑ Aplicação prática para projetos científicos, que utilizavam supercomputadores em conjunto

¹I. Foster, "The anatomy of the grid: enabling scalable virtual organizations," Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001

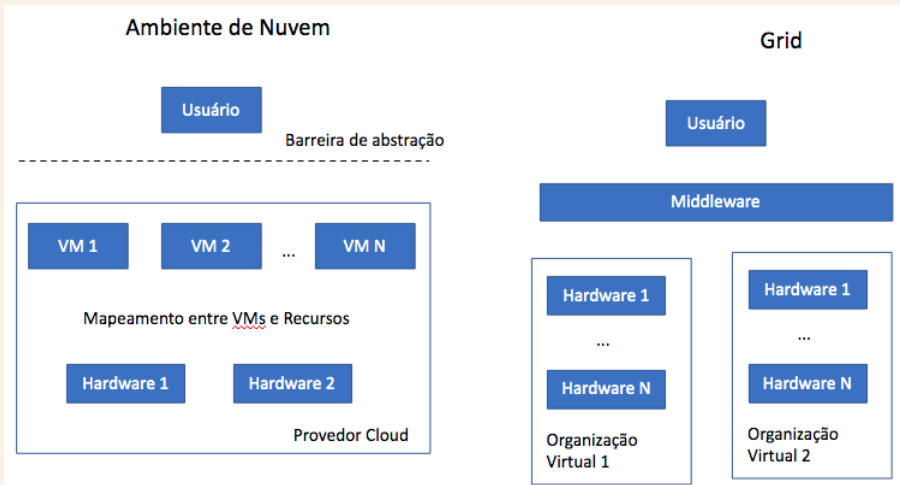
Grid Computing (computação em grade ou computação em malha)

- ❑ Esse modelo é baseado no uso do hardware diretamente
- ❑ Alta complexidade de uso. Desenvolvedores responsáveis por montar e executar a aplicação
- ❑ Limitação para uso de Virtualização:
 - Complexidade de implementação
 - Limitação de hardware
 - Hardware monoprocessoado
 - Time sharing entre virtualização e processo, tornaria inviável a execução de aplicações

Transição de grid para cloud (Motivações):

- ☐ Inviabilidade do Uso do grid para aplicações comerciais
- ☐ Nós computacionais com múltiplos processadores
 - Possibilidade de executar virtualização de modo eficiente
- ☐ Avanços nas tecnologias de virtualização
 - Um nó inteiro poderia ser controlado por um sistema de virtualização

Diferenças entre Grid e Cloud



Transição de grid para cloud (Motivações):

- ☐ Possibilidade de explorar o balanço entre o número de máquinas virtuais oferecidas em relação ao desempenho da máquina virtual
- ☐ Não precisa de um middleware complexo. Interface simples para interagir com recursos
- ☐ Possibilidade clara de otimizar custos com TI de acordo com as demandas por processamento

Desafio do rateio entre demanda e poder computacional



Modos de uso de recursos oferecidos por um ambiente de nuvem:

- ☐ IaaS
- ☐ PaaS
- ☐ SaaS

Ambientes podem oferecer todos esses modos ou se especializarem em alguns deles

IaaS (Infrastructure as a Service)

- ☐ Ambiente oferece os recursos básicos para montar um ambiente de TI
 - Recursos de rede
 - Computadores (virtuais ou em hardware dedicado)
 - Espaço de armazenamento de dados recursos diversos
- ☐ Recursos podem ser criados, destruídos e combinados de acordo com demandas diversas

PaaS (Platform as a Service)

- ☐ Recursos com foco na montagem e execução da aplicação
- ☐ A infraestrutura SO, hardware, servidor de aplicação, e outras tecnologias subjacentes, não controláveis pelo desenvolvedor
- ☐ Nesse contexto, a plataforma é responsável por prover suporte a recursos que a aplicação utilize

SaaS (Software as a Service)

- ☐ Nesse modelo o software é usado como serviço
- ☐ O foco é apenas no uso do software ou funcionalidade
- ☐ Normalmente, referenciado como aplicações para o usuário final
 - Exemplo: serviço de email para usuário final
 - Equipe de TI não faz backup, não muda o servidor de local, não se preocupa com variação de carga, invasões, etc

Estratégias de uso e montagem

- ☐ Cloud: aplicação é inteiramente montada em algum provedor de nuvem. Ou foi migrada de um ambiente local para a nuvem
- ☐ Híbrida: parte da aplicação roda localmente e parte na nuvem
 - Aumentar o número de recursos usados para executar a aplicação
- ☐ On-Premise: ambiente de nuvem montado localmente
 - Permite utilizar recursos legados com metodologias modernas de montagem de software

Um exemplo de PaaS é o Heroku

- ☐ Permite montar uma aplicação informando apenas os requisitos e forma de execução
- ☐ Implementa um git próprio para subir uma aplicação
- ☐ Painel na área do usuário permite executar e parar uma aplicação

Um exemplo de IaaS é o EC2 do AWS

- ☐ EC2 elastic compute cloud: unidade de hardware com sistema operacional e acesso por IP público
- ☐ Gerenciamento das instâncias pode ser feito pelo terminal
- ☐ É possível automatizar o processo de criar, executar uma aplicação e distribuir instâncias por scripts no terminal

- Um grande desafio em colocar aplicações em produção é montar o ambiente adequado de Sistema operacional, versões de bibliotecas e parametrizações do sistema.
- Quando há duas ou mais aplicações para colocar em produção o problema se torna ainda maior, pois um único ambiente deve atender duas demandas potencialmente distintas de sistema operacional
- Uma solução para esse problema é a virtualização, que refere-se a mecanismo de criar uma visão do sistema operacional para cada aplicação

- ❑ Um mecanismo de virtualização muito simples são as máquinas virtuais
- ❑ Outro mecanismo mais simples os ambientes virtuais como o conda
- ❑ Duas desvantagens desses dois métodos:
 - Máquinas virtuais prejudicam o desempenho das aplicações
 - Ambientes virtuais são desprovidos de flexibilidade quanto a configuração do sistema operacional

- ❑ Sistemas operacionais modernos oferecem recursos de virtualização no nível do sistema operacional
- ❑ Tal virtualização (chamadas containeres) parte da premissa de que o kernel do SO permite a existência de múltiplas instâncias isoladas do espaço do usuário
- ❑ Tais instâncias permitem criar um ambiente de SO próprio que acessa os recursos do computador e do SO instalado na máquina
- ❑ Do ponto de vista dos programas em execução neles, parecem computadores reais
- ❑ Containeres são mais rápidos de iniciar, produzem pouca sobrecarga no desempenho da aplicação e são muito flexíveis quanto a montagem do Sistema operacional



Contêineres Docker

- ❑ <https://docker-curriculum.com/>
- ❑ <https://docs.microsoft.com/pt-br/visualstudio/docker/tutorials/docker-tutorial>

Agenda

- ☐ Conceitos e definições
- ☐ Arquitetura Docker
- ☐ Criação e manipulação de contêineres
- ☐ Construindo imagens personalizadas

Definição

- ☐ Segregação de processos no mesmo kernel;
- ☐ Isolamento máximo possível de todo o resto do ambiente;
- ☐ File Systems, criados a partir de uma “imagem”;
- ☐ Torna a reprodutibilidade muito mais fácil
- ☐ Conceitualmente semelhante a máquinas virtuais

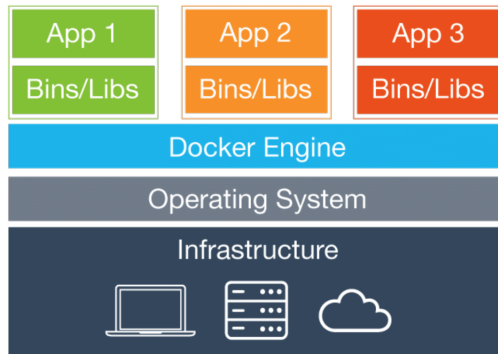


Figure: Aplicações utilizando Contêineres

Vantagens

- ❑ Leves porque não contêm um sistema operacional;
- ❑ Têm o mesmo desempenho que o código executado no sistema operacional host:
 - Até três vezes mais desempenho do que as máquinas virtuais, quando executados no mesmo hardware).
- ❑ Tempo de inicialização em milissegundos (em comparação com minutos para uma máquina virtual).
- ❑ Requerem pouca memória RAM
- ❑ São definidos usando código
 - Podemos tirar vantagem de sistemas de controle de código como o Git.
- ❑ Incentivam o reúso, para que você possa minimizar o retrabalho dispendioso.

Desvantagens

- ❑ Sempre são executados no sistema operacional Linux (os contêineres compartilham o sistema operacional do host):
 - Máquinas virtuais podem executar um sistema operacional diferente para cada máquina virtual
- ❑ Os contêineres usam isolamento no nível do processo
 - potencialmente menos seguro do que as máquinas virtuais totalmente isoladas


```
1
2 > docker run hello-world
3
4 Unable to find image 'hello-world:latest' locally
5 latest: Pulling from library/hello-world
6 b8dfde127a29: Pull complete
7 Digest: sha256:f2266cbfc127c960fd30e76b7c792dc23b588c0db76233517e1891a4
   e357d519
8 Status: Downloaded newer image for hello-world:latest
9
10 Hello from Docker!
11 This message shows that your installation appears to be working
   correctly.
12
13 ...
14
15 Share images, automate workflows, and more with a free Docker ID:
16 https://hub.docker.com/
17
18 For more examples and ideas, visit:
19 https://docs.docker.com/get-started/
```

Conceitos

- ❑ **Imagem:** um conjunto estático de arquivos binários que armazenam todas as informações necessárias para iniciar um contêiner
- ❑ **Contêiner:** um sistema operacional isolado usando containerização (neste caso via Docker) para rodar em um sistema operacional host (neste caso MacOS)

Importante

É importante ressaltar que um contêiner é uma instância em execução de uma imagem.

Imagens

- ☐ Materialização de um modelo de um sistema de arquivos
- ☐ Produzido através de um processo de build;
- ☐ Representada por um ou mais arquivos e pode ser armazenada em um repositório como Github;
- ☐ O Docker utiliza file systems especiais para otimizar o uso, transferência e armazenamento das imagens, containers e volumes.
 - O principal é o AUFS, que armazena os dados em camadas sobrepostas, e somente a camada mais recente é gravável.

Arquitetura

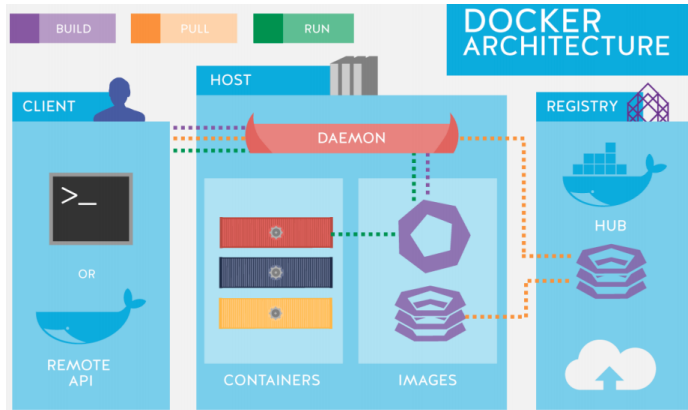


Figure: Arquitetura Docker

Parâmetros de Execução

☐ Execução Interativa:

- `--interactive` - isso nos permitirá digitar comandos de forma interativa no contêiner
- `--tty` - aloca um pseudo-TTY que permitirá que o contêiner imprima sua saída na tela.

```
1 > docker run --interactive --tty ubuntu:20.04 bash
2
3 Unable to find image 'ubuntu:20.04' locally
4 20.04: Pulling from library/ubuntu
5 a70d879fa598: Pull complete
6 c4394a92d1f8: Pull complete
7 10e6159c56c0: Pull complete
8 Digest: sha256:3c9c713e0979e9bd6061ed52ac1e9e1f246c9495aa063619d9d695fb
   8039aa1f
9 Status: Downloaded newer image for ubuntu:20.04
10 root@e2dd9abd3559:/#
```

Parâmetros de Execução

```
1 root@e2dd9abd3559:/# uname -r
2 4.19.121-linuxkit
3
4 root@e2dd9abd3559:/# cat /etc/lsb-release
5 DISTRIB_ID=Ubuntu
6 DISTRIB_RELEASE=20.04
7 DISTRIB_CODENAME=focal
8 DISTRIB_DESCRIPTION="Ubuntu 20.04.2 LTS"
```

Principais comandos:

- ❑ Listar Imagens:
 - `docker image ls`
- ❑ Listar Contêineres
 - `docker ps -a`
- ❑ Remover Contêineres/Imagens
 - `docker [image] rm 6f0684d58dca`
- ❑ Mapeamento de Volumes
 - `docker run -it -v [host]:[container] ubuntu bash`

Imagens Personalizadas

- ☐ A imagem é a abstração da infraestrutura em estado somente leitura, de onde será instanciado o contêiner.
- ☐ Imagens podem ser oficiais ou não oficiais
- ☐ As imagens oficiais são mantidas pela empresa docker e disponibilizadas no [docker hub](#);

Importante

- ☐ Todo contêiner é iniciado a partir de uma imagem
- ☐ Nunca teremos uma imagem em execução
- ☐ Um contêiner só pode ser iniciado a partir de uma única imagem

Imagens Oficiais

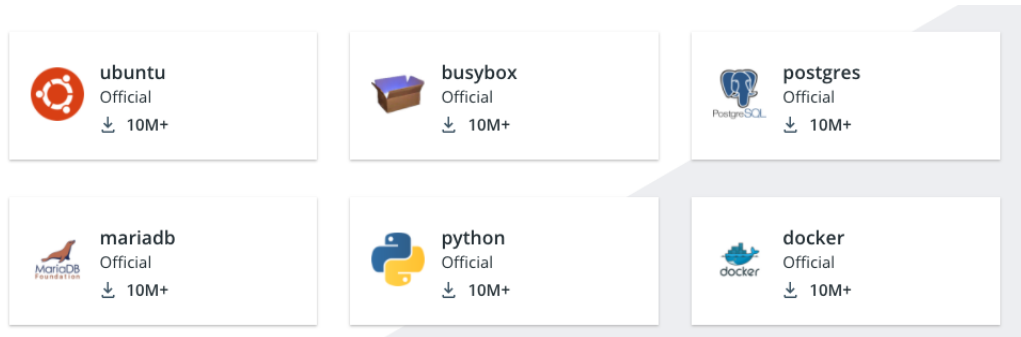


Figure: Imagens oficiais no Docker Hub

Imagens Personalizadas

- ❑ O objetivo das imagens oficiais é prover um ambiente básico;
 - um ponto de partida para criação de imagens pelos usuários
- ❑ As imagens não oficiais são mantidas pelos usuários que as criaram
- ❑ Nomeclatura:
 - Nome de uma imagem é composto por duas partes: *repositório* e *tag*;
 - `ubuntu:20.04`: repositório `ubuntu` e tag `20.04`
 - Cada dupla `repositório:tag` representa uma imagem diferente;

Utilizando o comando `commit` para criação de imagem:

- ❑ Criando um contêiner e realizando alguma modificação na imagem padrão:

```
1 > docker container run -it --name ubuntu-fun ubuntu:20.04 bash
2 root@fb4cdc61c273:/# apt update
3 root@fb4cdc61c273:/# apt install cowsay fortune -y
4 root@fb4cdc61c273:/# exit
```

Utilizando o comando `commit` para criação de imagem:

- Agora vamos criar uma nova imagem utilizando o comando `commit`:

```
1 > docker container commit ubuntu-fun ubuntu:fun
2 sha256:d88fd6d76bcba0de4c4bb2304330d0b662a5cadd4db447ba83ae54fc6bba848c
3 > docker image ls
4 REPOSITORY    TAG       IMAGE ID        CREATED         SIZE
5 ubuntu        fun       d88fd6d76bcb   4 seconds ago  149MB
```

Utilizando o comando `commit` para criação de imagem:

- Vamos testar a nova imagem:

```
1 > docker run --rm ubuntu:fun bash -c "/usr/games/fortune | /usr/games/  
    cowsay"  
2 -----  
3 / Q: What's tan and black and looks great \  
4 \ on a lawyer? A: A doberman. /  
5 -----  
6      \      ^__^  
7      \      (oo)\_____  
8          (__)\        )\/\  
9              ||----w |  
10             ||     ||
```

Dockerfiles:

- ❑ Conjunto de instruções aplicadas em uma imagem para geração de outra;
- ❑ Controle de diferenças entre uma imagem (base), e a imagem que se deseja criar;

```
1 > cat Dockerfile
2 FROM ubuntu:20.04
3 RUN apt-get update && apt-get install cowsay fortune -y
4 COPY arquivo_teste /tmp/arquivo_teste
5 CMD bash
```

Diretivas do Dockerfile:

- ☐ FROM: informa qual a imagem base
- ☐ RUN: informa quais comandos serão executados durante a criação da imagem
- ☐ COPY: copia arquivos do *host* para a imagem;
- ☐ CMD: informa qual comando será executado por padrão, caso nenhum seja informado na inicialização do contêiner

Construção da imagem:

```
1  > docker image build -t ubuntu:fun .
2  Sending build context to Docker daemon   2.56kB
3  Step 1/4 : FROM ubuntu:20.04
4  ----> 26b77e58432b
5  Step 2/4 : RUN apt-get update && apt-get install cowsay fortune -y
6
7  ...
8
9  Step 3/4 : COPY arquivo_teste /tmp/arquivo_teste
10  ----> dea0a06e75d4
11  Step 4/4 : CMD bash
12  ----> Running in 9e38da969f42
13  Removing intermediate container 9e38da969f42
14  ----> 5a1f0278b1c3
15  Successfully built 5a1f0278b1c3
16  Successfully tagged ubuntu:fun
```


Outras Diretivas do Dockerfile:

- ❑ ADD: faz o mesmo que o COPY, porém permite que a cópia seja feita de uma URL
- ❑ ENTRYPOINT: Define um executável (e argumentos padrão) a ser executado quando o contêiner é iniciado - não é redefinido pela linha de comando
- ❑ ENV: Define variáveis de ambiente dentro da imagem;
- ❑ WORKDIR: Define o diretório de trabalho para qualquer RUN, CMD, ENTRYPOINT, ADD ou COPY subsequente;

Ambiente AWS

- ☐ Fornece capacidade de computação redimensionável na nuvem
- ☐ Projetado para tornar a computação em nuvem em escala
- ☐ ambiente de computação virtual
 - Variedade de SOs
 - Escalável para diferentes volumes

Regiões

- ❑ Área geográfica onde o Amazon EC2 lançará as instâncias criadas
- ❑ Critérios para escolha de região
 - Desempenho para transferências de arquivos
 - Requisitos regulatórios
- ❑ Zonas de disponibilidades: regiões que atendem critérios mínimos de desempenho, custo ou disponibilidade definido pelo usuário

Instâncias

- ☐ Amazon EC2 oferece uma ampla seleção de tipos de instância
- ☐ otimizado para se adequar a diferentes casos de uso
- ☐ Os tipos de instância compreendem combinações variadas de CPU, capacidade de memória, armazenamento e rede

Modelo de custo

- ☐ Sob demanda: pagamento por hora
- ☐ Reserva: pagamento antecipado com garantia da disponibilidade do recurso
- ☐ Spot: recursos disponibilizados em formato de leilão

Configuração inicial

- ☐ Por segurança, todo acesso só pode ser realizado por meio de chave pública/privada
- ☐ No momento da criação da instância, uma chave é disponibilizada para esse fim
- ☐ É necessário também criar um grupo de segurança para determinar a política de autorização a ser aplicada a instância EC2 criada

AWS-cli

- ☐ Ferramenta para controlar via terminal o uso dos recursos oferecidos pela AWS
- ☐ Pode ser instrumentado por scripts bash
- ☐ Biblioteca boto3, permite manipular o aws-cli através do Python