

# Preparação do Ambiente

---

Advanced Institute for Artificial Intelligence – AI2

<https://advancedinstitute.ai>



# Sistemas Unix

---

# Agenda

1. Introdução/Motivação
2. Interação com o SO - Comandos Básicos
3. Gerenciando Processos
4. Manipulação de Arquivos

## Motivação

Como cientista de dados, **é quase impossível evitar os sistemas Unix**. Unix vs Windows não é uma questão de preferência como R vs Python; se você não conhece o Unix, **é provável que tenha dificuldades no local de trabalho**. Unix é o **sistema operacional da ciência de dados**, então você precisa saber como usá-lo.

## O que é Unix?

“

*Os sistemas Unix são caracterizados por um **design modular** que às vezes é chamado de “filosofia Unix”. Este conceito implica que o sistema operacional fornece um conjunto de **ferramentas simples**, cada uma desempenhando uma **função limitada e bem definida**, com um sistema de arquivos unificado (o sistema de arquivos Unix) como principal meio de comunicação, e script shell e linguagem de comando (o Unix shell) **para combinar as ferramentas** para realizar **fluxos de trabalho complexos**.*

Wikipedia

## Cenários de Uso

Você pode encontrar sistemas Unix como um cientista de dados:

- ☐ ao trabalhar na linha de comando usando sistemas operacionais macOS ou Linux,
- ☐ ao trabalhar com servidores remotos no data center do seu empregador,
- ☐ ao trabalhar com servidores remotos fornecidos pela AWS, Azure, GCP, etc,
- ☐ ao trabalhar com servidores Jupyter ou RStudio compartilhados (eles são hospedados em servidores Unix)

## Cenários de Uso

Alguns exemplos de uso do Unix para ciência de dados incluem:

- ☐ usando `ssh` para se conectar a um servidor de análise remoto
- ☐ usando `scp` para mover arquivos entre servidores
- ☐ usando um gerenciador de pacotes (por exemplo, `brew`, `apt`, `yum`) para instalar ferramentas de linha de comando
- ☐ usando ferramentas de linha de comando para interagir com serviços de terceiros
- ☐ usando `wget` para baixar arquivos da internet
- ☐ usando `top` para monitorar a utilização de recursos do sistema
- ☐ usando `docker` para gerenciar e executar contêineres

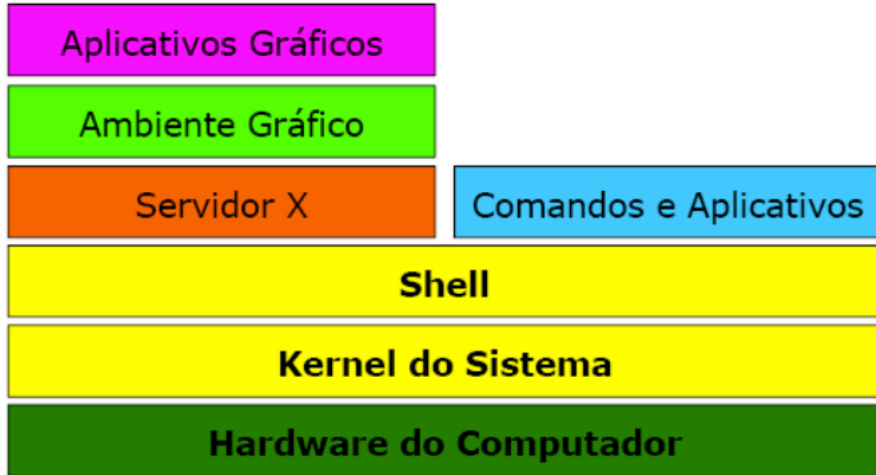
O Sistema Operacional (SO) é o programa que controla o computador, servindo de Interface entre o usuário e a máquina. O Sistema Operacional faz isso através de dois componentes: O Kernel e o Shell

- ❑ Kernel é o nome dado ao núcleo do Sistema Operacional. É o módulo deste programa que se comunica com o hardware do computador
- ❑ Shell é a “fachada” do Sistema Operacional. Essa é a parte do programa que se comunica com o usuário, recebendo seus comandos e repassando-os ao Kernel



## Distribuição Linux

- É o nome dado ao conjunto de programas formado pelo Kernel Linux e por mais alguns softwares distintos (como Shells, aplicativos, jogos, utilitários, etc.)
- Várias empresas (ou pessoas) podem agrupar os programas que acham interessantes e criar suas próprias distros
- O Que Há Numa Distribuição?
  - Kernel, shell e ambiente gráfico
    - KDE, Gnome, ABlackBox, WindowMaker, Fluxbox



- Protege usuários de lidar com as *entranhas* do SO;
- Quando as pessoas dizem "shell" hoje em dia, quase sempre se referem ao **B**ourne **A**gain **S**hell (**bash**)
- Exemplo de um REPL - *Read-Evaluate-Print Loop*.
  1. o shell lê seu comando;
  2. o shell avalia seu comando e calcula a saída;
  3. o shell imprime a saída para o terminal;
  4. o shell imprime um novo prompt, pronto para seu próximo comando;

Comandos de sessão:

- ☐ Login: iniciar sessão: Interface Gráfica ou Terminal
- ☐ Logoff : sair da sessão
  - ☐ `exit`
- ☐ Reboot : reinicia o sistema
  - ☐ `reboot`
- ☐ `poweroff` ou `shutdown -h now` desliga o sistema

Atalhos úteis em uma sessão:

- ☐ TAB auto completa comandos
- ☐ History mostra a lista de comandos executados
- ☐ ! id-histórico executa o comando do histórico com o id id-histórico
- ☐ Man mostra ajuda de comandos

## Estrutura de arquivos

- ❑ Referência a um arquivo/diretório é feita através de um caminho, que é formado da seguinte forma:
- ❑ [diretório raiz] [diretório 1] [diretório 2] ... [diretório n] [arquivo]
- ❑ Cada um desses itens são separados por uma / (barra) no Linux
- ❑ o diretório raiz chama-se / (barra)
  - Exemplo de caminho: `/home/rmcobe/doc/1`
- ❑ exemplo de caminho: `/home/rmcobe/doc/1`

## Comandos Básicos - Manipulação do Sistema de Arquivos

- ☐ **cd**: Comando para acessar um diretório
- ☐ **pwd**: retorna o caminho do diretório desde a raiz /
- ☐ **cp**: copia um arquivo ou diretório de uma caminho para outro
- ☐ **find**: procurar por arquivos ou diretórios no sistema de arquivos
- ☐ **mkdir**: cria um novo diretório
- ☐ **mv**: move um arquivo ou diretório de um caminho para outro
- ☐ **ls**: lista um diretório

## Comandos Básicos - Manipulação do Sistema de Arquivos

- ☐ **rm**: remove um diretório, desde que esteja vazio
- ☐ **touch**: cria um arquivo vazio
- ☐ **du**: retorna o espaço utilizado por um diretório ou arquivo
- ☐ **df**: retorna as partições presentes no sistema
- ☐ **tree**: retorna a árvore do sistema
- ☐ **chmod**: muda a permissão de arquivos e diretórios



## Alguns diretórios possuem notações especiais ou "atalho"

- `~`: ao referir-se ao diretório `~` (til), o sistema entende como o diretório pessoal do usuário, ou seja, `/home/[usuário]`, onde `[usuário]` é o nome de login do usuário atual.
- `-`: o kernel Linux armazena um histórico dos diretórios que acessamos. O `-` (hífen) refere-se ao último diretório acessado
- `.`: o símbolo `.` (ponto) refere-se ao diretório atual, ou seja, aquele em que estamos trabalhando
- `..`: o `..` (ponto ponto) refere-se ao diretório acima do qual estamos.

No Linux existem três tipos de permissão para definir os acessos a arquivos e diretórios:

- ☐ **r** : Permissão de leitura para um arquivo; e permitir listar o conteúdo de um diretório através do comando `ls` diretório
- ☐ **w** : Permissão de gravação e exclusão para um arquivo/diretório.
- ☐ **x** : Permissão para executar um arquivo, se for um arquivo binário ou um script; e se for um diretório permitir acesso a ele através do comando `cd` diretório
- ☐ Tais permissões são concedidas aos seguintes papéis de usuário
  - **Dono** : O proprietário do arquivo ou criador do arquivo
  - **Grupo**: Usuários que fazem parte do grupo do proprietário
  - **Outros**: Não são os proprietários e nem fazem parte do grupo



## Filtros

- ❑ **grep** Examina cada linha de dados que recebe da entrada padrão e produz cada linha que contém um padrão especificado de caracteres.
- ❑ **sort** Classifica a entrada padrão e produz o resultado classificado na saída padrão
- ❑ **uniq** Dado um fluxo de dados classificado da entrada padrão, ele remove linhas de dados duplicadas (ou seja, garante que cada linha seja única).
- ❑ **head** Exibe as primeiras linhas de sua entrada. Útil para obter o cabeçalho de um arquivo.
- ❑ **tail** Exibe as últimas linhas de sua entrada. Útil para coisas como obter as entradas mais recentes de um arquivo de log.

## Variáveis de Ambiente

- ☐ **export** : cria uma variável
- ☐ **env** : mostra as variáveis criadas
- ☐ **unset** : apaga uma variável
- ☐ **echo** : mostra o valor atribuído a uma variável

## Tipos de processos:

- Processos interativos: são iniciados a partir de uma sessão de usuário no terminal de comandos e são controlados por ele
- Processos em lote (batch): o processo entra em execução e não permite nenhuma interação com o usuário do SO
- Daemons: são processos servidores normalmente executados quando o Linux é inicializado, permanecendo em execução enquanto o sistema estiver em funcionamento esperando em background que outro processo solicite o seu serviço.
  - Utiliza-se o operador & para que o processo execute e libere o terminal

Pesquisando processos em execução comando ps

- ☐ Exibe informações sobre os processos ativos
- ☐ `ps` [opções]
- ☐ `a` exibe também informações de outros usuários
- ☐ `u` exibe o nome do usuário e a hora de início do processo
- ☐ `x` exibe também os processos não associados a um terminal de controle
- ☐ `-p pid` exibe o processo cujo número é pid

kill: finaliza um processo por meio do pid;

- ❑ `kill [opções] [sinal] pid`
- ❑ `-n` sinal aplicado ao processo
- ❑ `-l` lista todos os nomes e números de sinais
  - `kill -9 1029`



Comandos para exibir informações sobre o computador de forma geral

- ❑ **free**: exibe a quantidade de memória livre;
- ❑ **lscpu**: exibe informações sobre o CPU

## Redirecionamento

- Redirecionamento de saída:
  - `>` e `>>`
- Redirecionamento de entrada:
  - `<` e `<<`
  - `sort < arquivo`
- Pipe (`|`) - redireciona a saída de um comando para a entrada do próximo:
  - `cat arquivo | uniq`

**cat** : concatena arquivos e lista na saída padrão

- ❑ Sintaxe: **cat** [opções] parametros
- ❑ Parâmetros pode ser uma lista contendo arquivos
- ❑ Exemplo de uso: **cat** `arq1.txt` `arq2.txt`
  - Concatena `arq1.txt` e `arq2.txt` e exibe na saída padrão
- ❑ Outro Exemplo: **cat** `>` `arq3.txt`



# Contêineres Docker

---

## 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

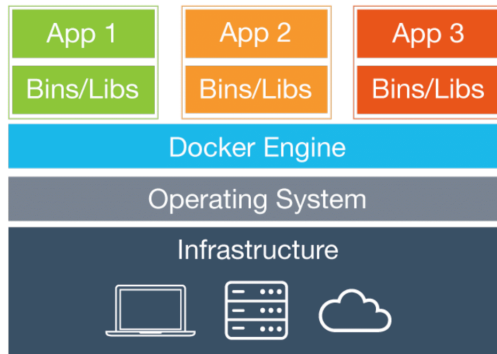


Figure: Aplicações utilizando Contêineres

## 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:f2266cbfc127c960fd30e76b7c792dc23b588c0db76233517e1891a4e35
   7d519
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

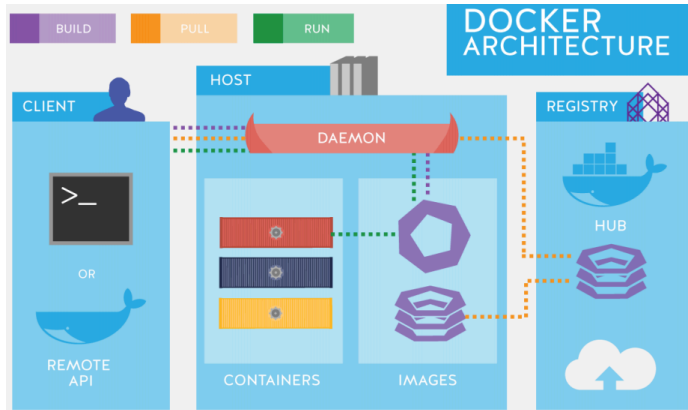


Figure: Arquitetura Docker

## 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: sha256:3c9c713e0979e9bd6061ed52ac1e9e1f246c9495aa063619d9d695fb803
   9aa1f
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`