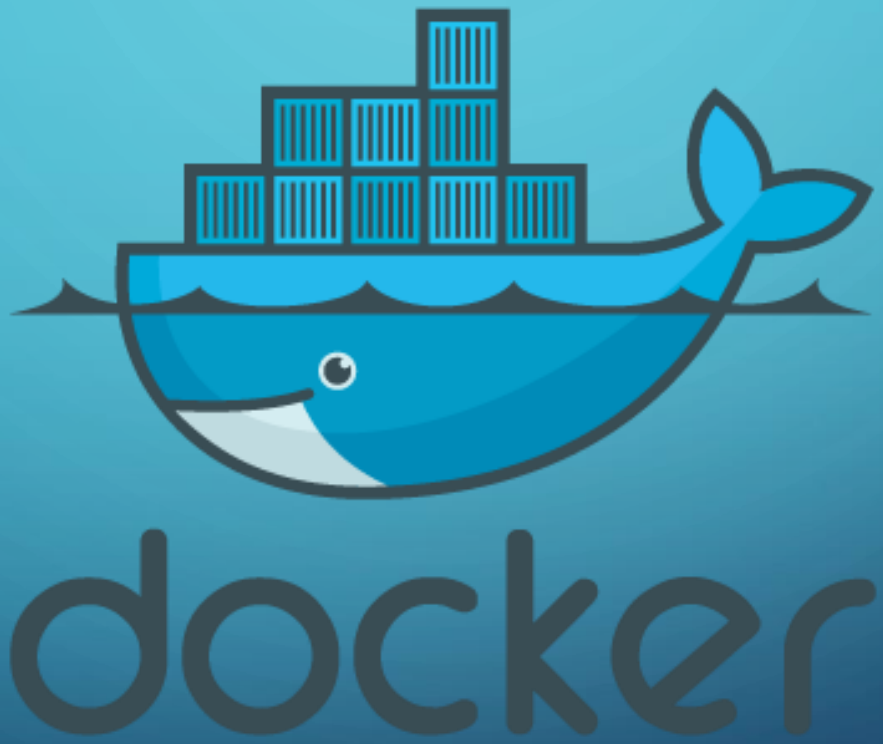
INTRODUÇÃO AO DOCKER



DOCUMENTAÇÃO

- **Links**
  - https://docs.docker.com
  - https://github.com/ViniFTex/Docker

O QUE É O DOCKER ?

- O Docker são containers que permitem empacotar aplicativos ou trechos de código, em um sistema de arquivos completo, que contém tudo o que é necessário para execução. Garantindo assim sua autenticidade.
- Ou seja, trata-se de uma tecnologia Open Source que permite criar, executar, testar e implantar aplicações distribuídas dentro de containers de software.

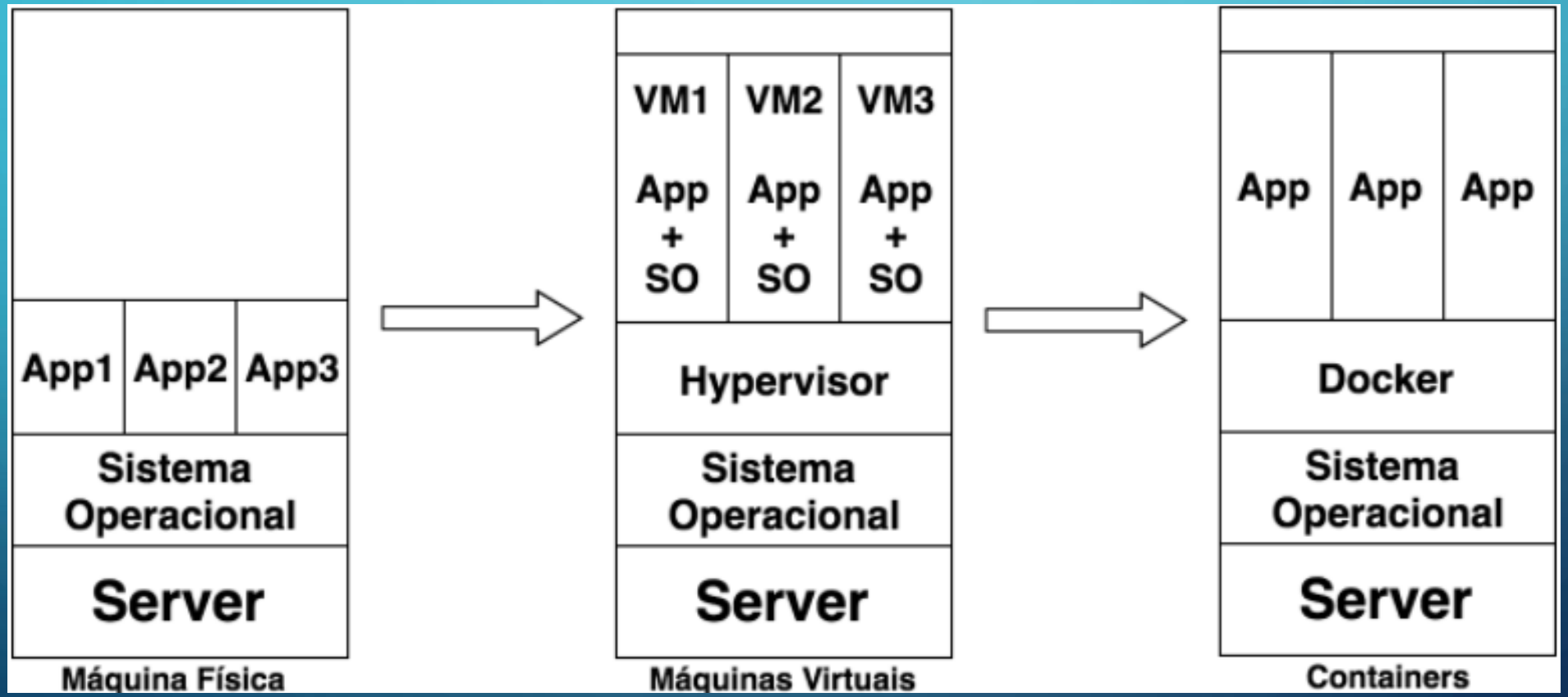
O QUE SÃO CONTAINERS ?

- Os containers são um método de virtualização em nível de sistema operacional que permite executar uma aplicação e suas dependências em processos com recursos isolados.
- É um agrupamento de uma aplicação junto com suas dependências , que compartilham o kernel do sistema operacional, ou seja da máquina (Virtual ou Física) onde está rodando.
- Containers são similares às maquinas virtuais, porém mais leves e mais integrados ao sistema operacional da maquina host.

CARACTERÍSTICAS

- Leve
  - Compartilhamento do kernel do S.O. e uso mais eficiente da memória RAM.
- Aberto (Open Source):
  - Sistema disponível nas principais distribuições:
    - Mac OS, Linux e Windows (Open Source no Windows 10 e Enterprise no Windows Server).
- Seguro
  - Os containers isolam as aplicações, umas das outras e de sua infraestrutura, com uma camada adicional de proteção.

DIFERENÇAS ENTRE FÍSICA, VM E DOCKER



VM VS DOCKER

- VM
  - O objetivo desse modelo é compartilhar os recursos físicos entre vários ambientes isolados sendo que cada um deles tem sob sua tutela uma máquina inteira, com memória, disco, processadores, rede e outros periféricos, todos entregues via abstração de virtualização.
  - É como se dentro de uma máquina física criasse máquinas menores e independentes entre si. Cada máquina dessa tem seu próprio sistema operacional completo, que por sua vez interage com todos os hardwares virtuais que lhe foi entregue pelo modelo de virtualização a nível de máquina.
  - O sistema operacional instalado dentro de uma máquina virtual fará interação com os hardwares virtuais e não o hardware real.

VM VS DOCKER

- Docker
  - Esse modelo de virtualização está no nível de sistema operacional, ou seja ao contrário da máquina virtual um container não tem visão de uma máquina inteira, ele é apenas um processo em execução em um kernel compartilhado entre todos os outros containers.
  - Ele utiliza o **Namespace** para prover o devido isolamento de memória RAM, processamento, disco e acesso a rede. Mesmo compartilhando o kernel, o Docker em execução tem a visão de estar usando um sistema operacional dedicado.

NAMESPACES

- Namespaces foram adicionados no kernel Linux na versão 2.6.24 e são eles que permitem o isolamento de processo quando estamos utilizando o Docker.
- São os responsáveis por fazer com que cada container possua seu próprio environment, ou seja, cada container terá a sua árvore de processo, pontos de montagens, etc.
- Faz com que um container não interfira na execução de outro.

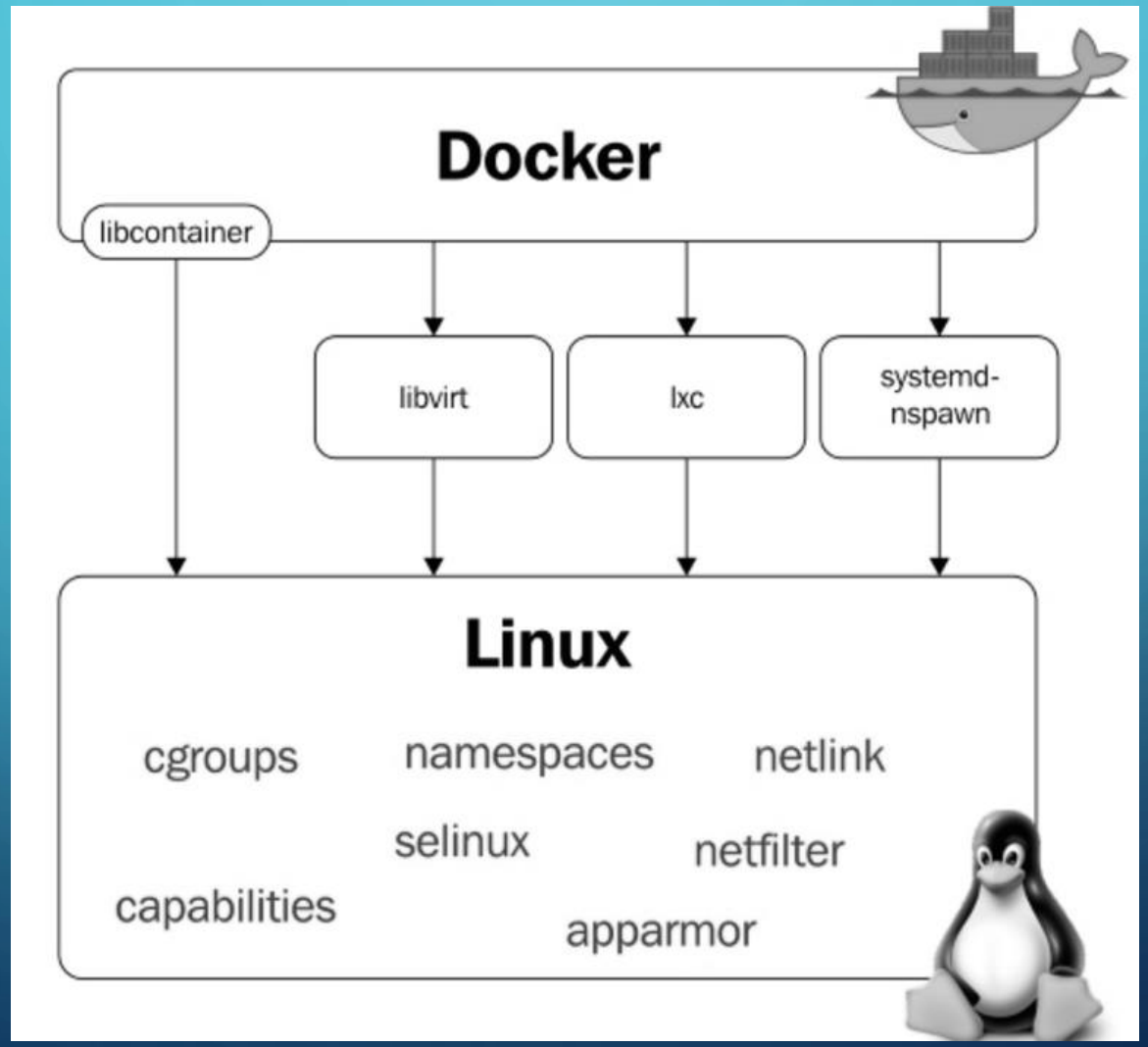
NAMESPACES E SUAS UTILIDADES

- PID namespace
  - O PID permite que cada container tenha seus próprios identificadores de processos.
- Net namespace
  - O Net permite que cada container possua sua interface de rede e portas.
- Mnt namespace
  - Com o Mnt namespace cada container pode ser dono de seu ponto de montagem, bem como o de um sistema de arquivos raiz.
  - Ele garante que um processo rodando em um sistema de arquivos não consiga acessar outro sistema de arquivos montado por outro Mnt namespace.

NAMESPACES E SUAS UTILIDADES

- IPC namespace
  - Ele provê um SystemV IPC isolado, além de uma fila de mensagens POSIX própria.
- UTS namespace
  - Responsável por prover o isolamento de hostname, nome de domínio, versão do SO, etc.
- User namespace
  - É responsável por manter o mapa de identificação de usuário em cada container.

DOCKER INTERNALS



CGROUP E NETFILTER

- Cgroups
  - É responsável por permitir a limitação da utilização de recursos do host pelos containers.
  - Com o cgroups você consegue gerenciar a utilização de CPU, memória, dispositivos, I/O, etc.
- Netfilter
  - Para que os containers consigam se comunicar, o Docker constrói diversas regras de roteamento através do iptables; inclusive utiliza o NAT.

STORAGE DRIVERS

- OverlayFS2
  - OverlayFS2 é a versão atualizada do Union Filesystem, versão atualmente recomendada pela Docker que tem correções que foram sendo melhoradas a partir de versões mais antigas como OverlayFS e AUFS.
- BTRFS
  - É a geração seguinte do Union Filesystem, que trás vários outros benefícios, mas ainda só está disponível em versões CE e na Versão EE do SLES (Suse Linux Enterprise Server).

ELEMENTOS DOCKER

- Docker Image
  - Imagens são templates para criação de containers, imagens são imutáveis, para executa-las é necessário criar uma instância dela o “Container”.
  - É importante falar também, que as imagens são construídas em camadas, o que facilita sua reutilização e manutenção.
  - Em resumo uma imagem nada mais é do que um ambiente totalmente encapsulado e pronto para ser replicado onde desejar.
- Dockerfile
  - São scripts com uma série de comando para criação de uma imagem, nesses scripts é possível fazer uma série de coisas como executar comandos sh, criar variáveis de ambiente, copiar arquivos e pastas do host para dentro da imagem etc.

ELEMENTOS DOCKER

- Docker Registry
  - É similar a um repositório GIT, onde as imagens podem ser versionadas, comitadas e puxadas.
  - Quando recuperamos uma imagem, usando o comando Docker pull por exemplo, estamos normalmente baixando a imagem de um registro Docker.
  - Pode ser enviado ou baixado imagens do repositório default que é o dockerHub.com, ou em registrys privados como é o caso do harbor.

DOCKER COMPOSE

- Docker compose trata-se de uma forma de você conseguir escrever em um único arquivo todos os detalhes do ambiente de sua aplicação.
- Com o Docker compose faz com que possamos construir um ambiente inteiro. Por exemplo: no Docker compose vamos configurar para ele subir um container com Frontend, Backend e banco de dados cada um com sua estrutura mas utilizando apenas uma configuração para isso.

LABORATÓRIO DOCKER

- Acesso aos passos do Laboratório
  - https://github.com/ViniFTex/Docker

REFERÊNCIAS

- https://pt.slideshare.net/AndrJusti/apresentao-Docker-73035181
- https://livro.descomplicandodocker.com.br

THE END