

Introdução ao Docker

- ▼ O que é o Docker
- ▼ Vantagens
- ▼ Porque eu devo usar o Docker
- ▼ Como a estrutura de servidores é projetada
- ▼ Conhecendo a arquitetura e o conceito
- ▼ Alguns comandos basicos do Docker

O que é o Docker?

O docker é uma plataforma aberta criada com o intuito de

- ▼ Facilitar no desenvolvimento
- ▼ Agilizar a implementação
- ▼ E a possibilidade de ter aplicações em ambientes isolados

O docker foi abraçado pela comunidade e graças a

- ▼ Facilidade de gerenciar
- ▼ A agilidade para subir um novo ambiente
- ▼ E a simplicidade de realizar modificações

Vantagens - Principais vantagens

Algumas das vantagens do uso do Docker é ter um ambiente

- ▼ Extremamente leve
- ▼ Isolado e ajudando com projetos legados

Temos a possibilidade de criar diversos containers e serem executados simultaneamente em um mesmo host.

Vantagens - Dependências

Podemos ter um gerenciamento de dependência melhor e assim dando a possibilidade de cada contêiner contem as dependências de cada aplicação.

Depois de um projeto criado podemos facilmente

- ▼ Importar
- ▼ Exportar

Assim podemos subir de forma fácil para um outro ambiente ou até para a produção.

Vantagens - Diversas versões

- ▼ Podemos ter diversos imagens e containers para cada versão do projeto.
- ▼ Depois de criado podemos ter a portabilidade de testar em qualquer ambiente e facilmente ter uma escalabilidade para produção.
- ▼ A facilidade de testar diversas versões de um mesmo projeto em diversos ambientes com agilidade

Porque eu devo usar o Docker

Afinal , porque mudar o modo de trabalhar hoje?

- ▼ Atualmente o Docker está sendo considerada um "idioma" , pois ele auxilia na comunicação entre o código e a infraestrutura.
- ▼ A facilidade para encontrar exemplos , projetos opensource e ver como outros desenvolvedores trabalham.
- ▼ Poupar tempo de novos desenvolvedores e da equipe.
- ▼ A agilidade de varias pessoas ter o mesmo ambiente

Ainda não sei se uso o Docker - Desenvolvedor

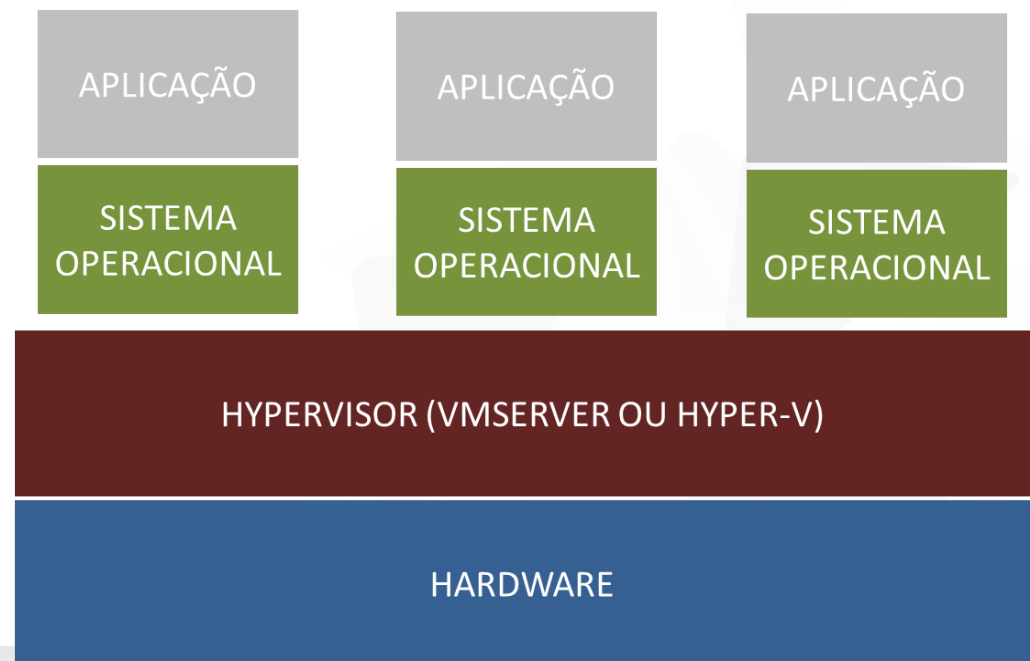
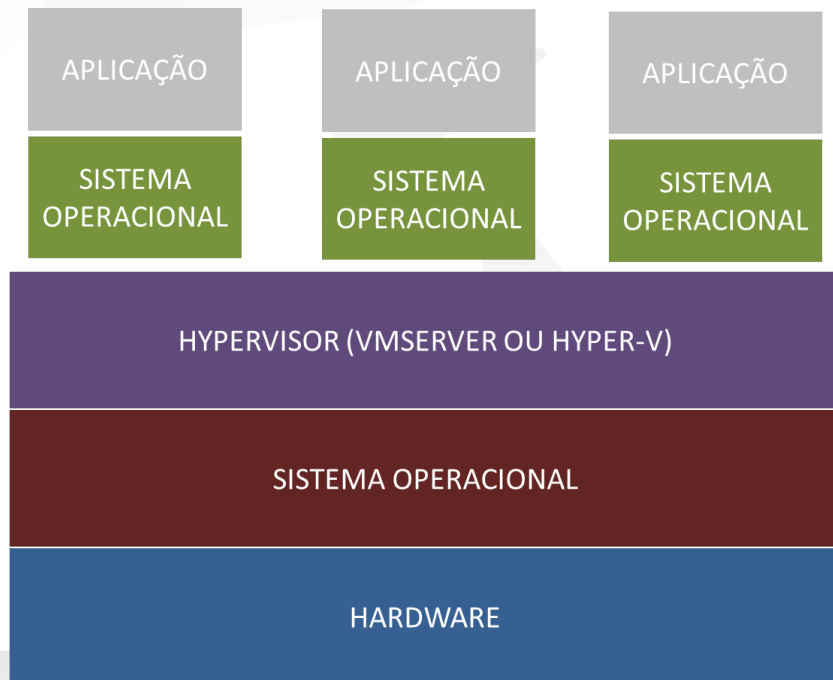
- ▼ Só precisa se preocupar em desenvolver uma vez e pode executar em qualquer local.
- ▼ Não é necessário se preocupar com dependências ou pacotes.
- ▼ Só precisa ter o foco no desenvolvimento.
- ▼ Com o Docker é possível ter diversos ambientes para testes.
- ▼ Assim evitamos o clássico no localhost funciona

Ainda não sei se uso o Docker - Syadmin

- ▼ O responsável pela infraestrutura configura uma vez e executa em qualquer lugar
- ▼ Podemos eliminar incertezas na entrega das aplicações ou serviços
- ▼ Com o Docker temos um ciclo de trabalho mais eficiente e ágil
- ▼ É possível ter uma infraestrutura escalável facilmente

Sem o uso da plataforma de containers - Infra

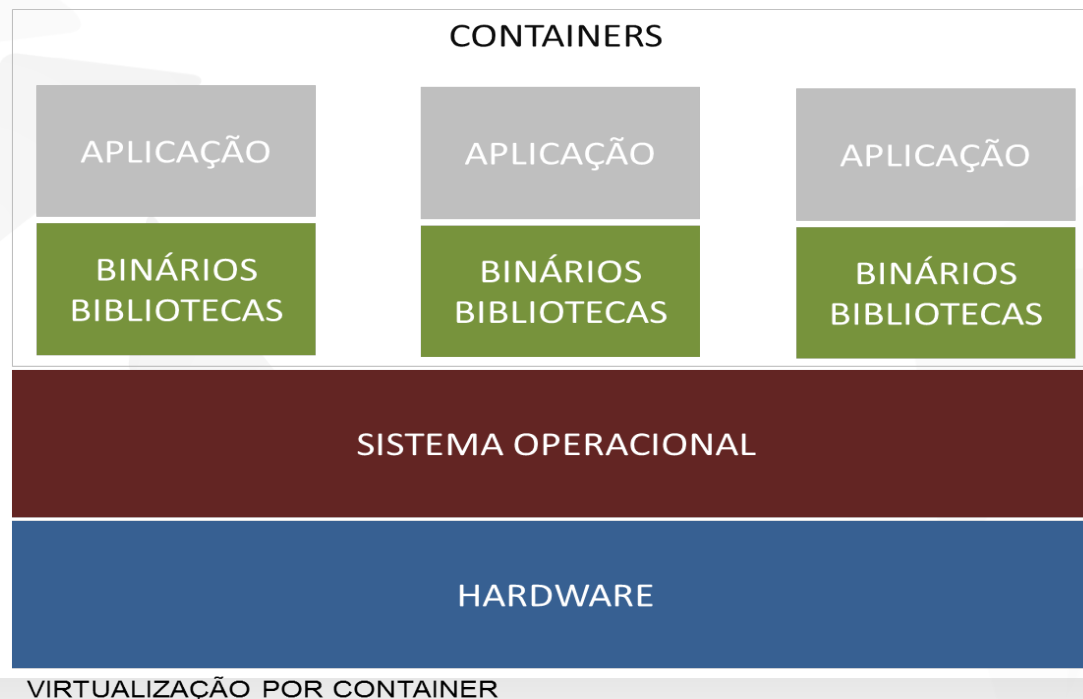
- Uma infraestrutura sem o uso de containers deixa todo o projeto mais pesado e lento.
- Em toda maquina virtual tem um sistema operacional instalado , deixando o projeto muito mais lento , pesado e até aumentando o custo



Como funciona com o uso de containers - Infra

Já com o uso de containers podemos ver que só é necessário obter os binários e as bibliotecas.

- ▼ O Docker usa um modelo de isolamento utilizado é o virtualização a nível de sistema operacional



Conhecendo a arquitetura - Imagens

- ▼ As imagens podemos entender como formas de bolo ou template usados para criar containers
- ▼ As imagens não containers , mais dão base consistente para que aja um container.
- ▼ Temos imagens oficiais ou criadas pela comunidades.
- ▼ Podemos armazenar elas localmente usando o Docker Registry ou publicamente no Dockerhub

Conhecendo a arquitetura - Containers

- ▼ Já os containers podemos entender como bolos prontos , não podemos criar um container sem uma imagem previamente.
- ▼ Os containers contem o necessario para executar uma aplicação e são baseados nas images.
- ▼ Os mantem o isolamento da aplicação e de recursos.
- ▼ Os containers são voláteis e depois de desligado todo o seu conteúdo é perdido.

Conhecendo o conceito – Docker Engine

O docker engine é um daemon , ele é nos auxilia na

- ▼ Construção
- ▼ No envio
- ▼ E na execução dos nossos containers.

Conhecendo o conceito - Docker Client

- ▼ Já o Docker client é responsável por receber as entradas do usuário e as enviar para a engine.
- ▼ Podemos ter um client e o engine na mesma maquina. Porem eles podem ser executados em hosts diferentes.

Conhecendo o conceito - Docker Registry

- ▼ O Docker registry é responsável por armazenar nossas imagens , ele pode ser usado de forma local ou criar o nosso próprio servidor de imagens.
- ▼ Esse servidor pode ser privado ou publico , um exemplo de servidor publico é o Dockerhub onde podemos encontrar diversas imagens e até subir a nossa propria imagem.

Conhecendo o conceito - Dockerfile

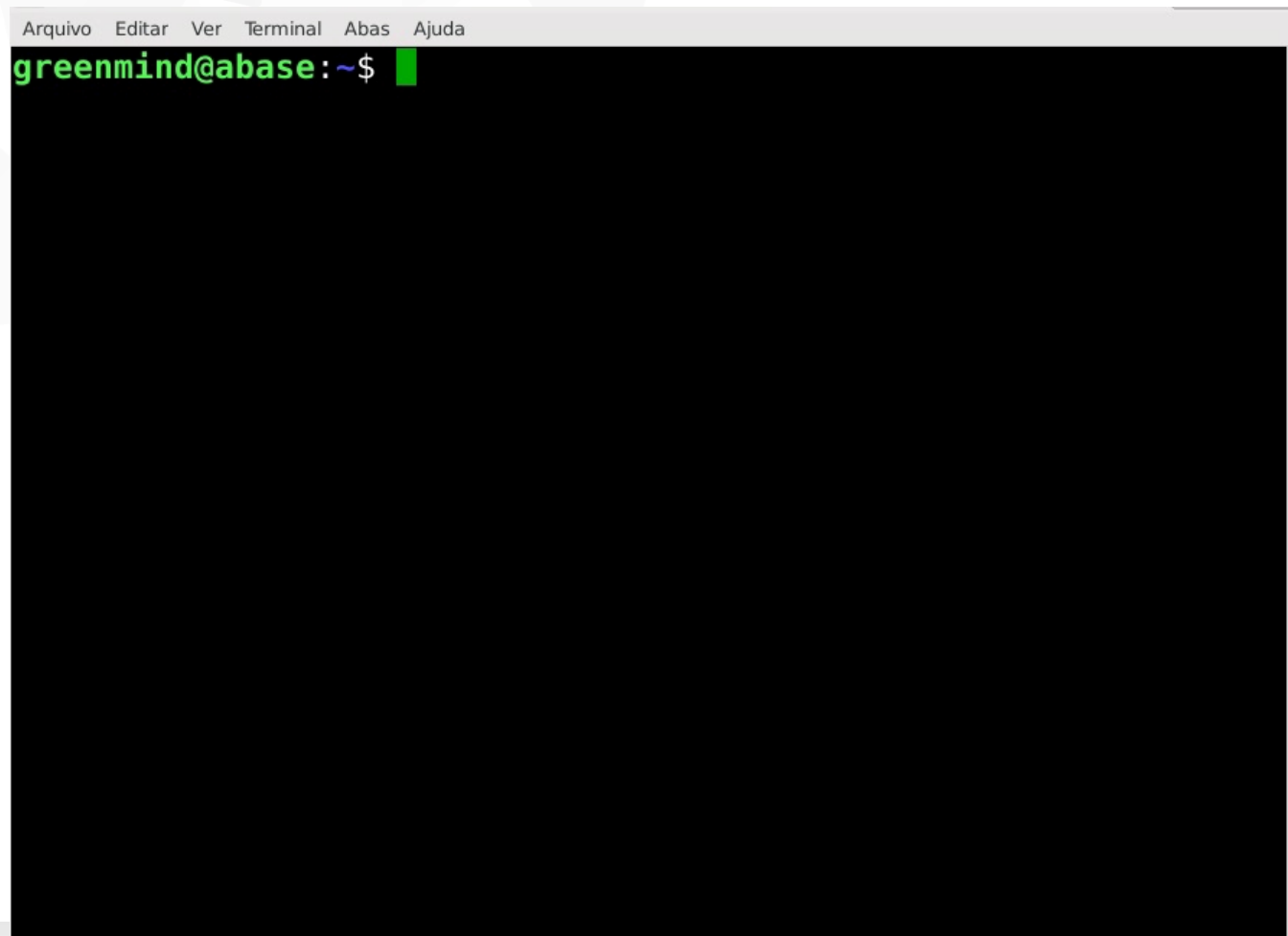
- ▼ O dockerfile é um arquivo que auxiliar na criação de uma imagem.
- ▼ É usado instruções e elas são aplicadas em uma determinada imagem para que outra imagem seja criada baseada nas modificações.
- ▼ Depois de criada podemos subir nossa imagem para o dockerhub.

Conhecendo o conceito – Docker compose

- ▼ O docker compose é uma ferramenta para nos auxiliar na criação de múltiplos containers Docker
- ▼ Assim como no Dockerfile podemos configurar todos os parâmetros necessários para executar um container a partir de um arquivo que é o docker-compose.yml
- ▼ Nesse arquivo de execução podemos selecionar definir determinados serviços , podemos setar portas abertas , variáveis de ambiente , volumes , configurar redes e muitas possibilidade que não conseguimos apenas com o Dockerfile.

Comandos basicos - help

Podemos obter mais informações sobre o docker e o docker-compose usando a opção

A screenshot of a terminal window. The title bar at the top contains the menu items: "Arquivo", "Editar", "Ver", "Terminal", "Abas", and "Ajuda". The terminal content shows a green prompt "greenmind@abase:~\$" followed by a green cursor block. The rest of the terminal area is black and empty.

```
Arquivo  Editar  Ver   Terminal  Abas  Ajuda  
greenmind@abase:~$
```

Comandos basicos - pull

O comando **pull** é responsável por obter uma imagem, essa imagem pode ser do DockerHub ou de outro servidor de imagens.

Podemos querer escolher uma determinada tag, senão ele vai baixar a versão latest.

Comandos basicos - images

O comando **images** é responsável por listar as imagens que temos em nossa maquina.

Dessa forma podemos ver qual imagem usar e até qual TAG.

Comandos basicos - ps

Podemos listar os container que estão em funcionamento usando o **ps** e ele nos auxilia para saber quais containers estão em funcionamento.

Vamos ter informações como por exemplo

- ▼ CONTAINER ID
- ▼ IMAGE
- ▼ COMMAND
- ▼ CREATED
- ▼ STATUS
- ▼ PORTS
- ▼ NAMES

Comandos basicos - run

Agora que já sabemos como obter uma imagem e como listar elas vamos executar um container.

- ▼ Podemos executar ele tambem em background
- ▼ Podemos executar ele interagindo diretamente com ele e quando terminarmos ele ser excluido.

Comandos basicos - Volumes

Vamos realizar mapeamento da seguinte forma.

- ▼ Primeiro precisamos especificar qual origem do no host e segundo onde devemos montar dentro do container.

Comandos basicos - Portas

- ▼ Podemos realizar o mapeamento de portas da seguinte forma.
- ▼ Primeiro precisamos saber qual porta será mapeada no host e qual deve receber essa conexão dentro do container.

Comandos basicos - exec

Depois de criar um container tambem podemos ter acesso a eles , para isso vamos usar o exec.

Vamos supor que temos um container com o nome **web_server** e quero ter acesso a ele.

Comando basicos – criando container

Depois que aprendemos a inserir nome em um container e subir ele para funcionar em background podemos realizar comandos como por exemplo

- ▼ stop
- ▼ start

Dessa forma podemos iniciar e parar container com maior facilidade.

Bonus – gerenciando containers com Portainer

Agora que já aprendemos o básico podemos conhecer um projeto chamado Portainer.

O portainer nos auxilia na administração do docker e ainda

- ▼ Gerenciar container
- ▼ Auxilia na criação de imagens
- ▼ Redes
- ▼ Volumes

E tudo isso com duas linhas de comando.

Perguntas?





Referencias e agradecimentos ...

- ▼ <https://github.com/gomex/docker-para-desenvolvedores/>
- ▼ <https://www.linuxtips.com.br/>
- ▼ E toda a comunidade foda que está a nossa volta



All text and image content in this document is licensed under the [Creative Commons Attribution-Share Alike 3.0 License](#) (unless otherwise specified). "LibreOffice" and "The Document Foundation" are registered trademarks. Their respective logos and icons are subject to international copyright laws. The use of these therefore is subject to the [trademark policy](#).



Obrigado ...

- ▼ <https://github.com/greenmind-sec/>
- ▼ <https://github.com/ABase-BR/Docker-intro>
- ▼ <https://www.juliusec.com/>
- ▼ <https://wiki.juliusec.com/>
- ▼ <https://ezdevs.com.br/>



All text and image content in this document is licensed under the [Creative Commons Attribution-Share Alike 3.0 License](#) (unless otherwise specified). "LibreOffice" and "The Document Foundation" are registered trademarks. Their respective logos and icons are subject to international copyright laws. The use of these therefore is subject to the [trademark policy](#).