Habilitando Aplicações Nativas de Nuvem

Introdução a Contêineres e Kubernetes

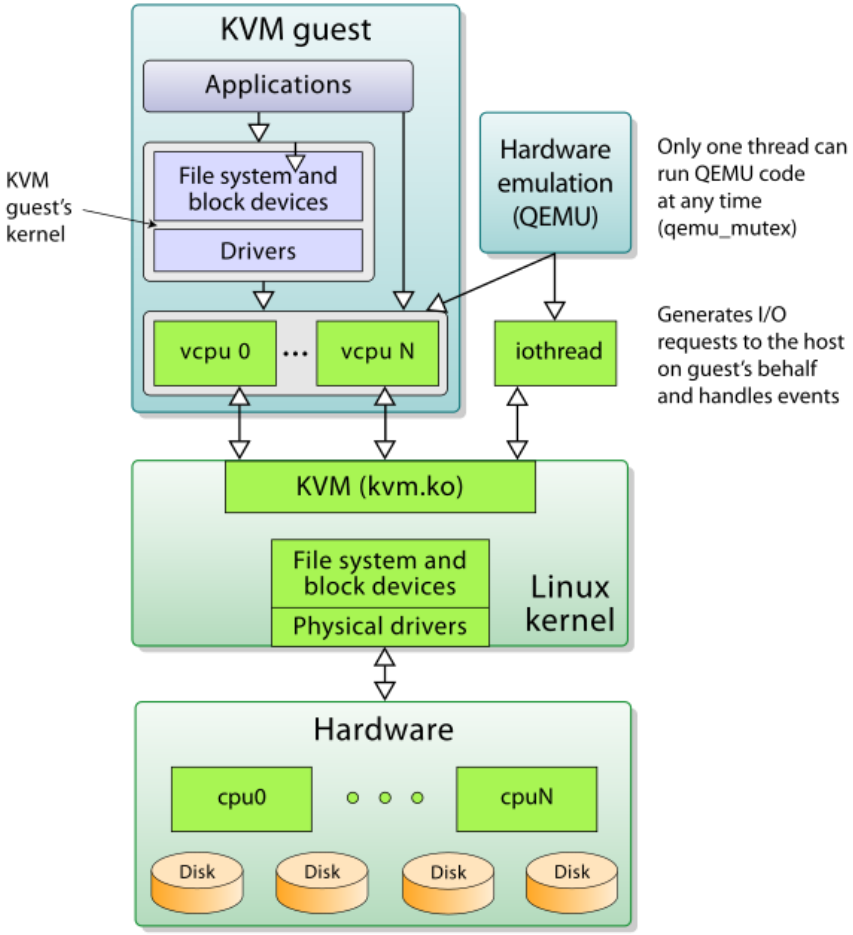
2. Aplicativos Modernos e Contêineres

Sergio Rio
www.sergiorio.tech

Refrescando nossa memória

- Na aula passada estudamos os centros de dados definidos por software, a motivação econômica para os serviços de nuvem e a tecnologias para virtualização que os viabilizam.
- Atividades preparatórias para a aula de hoje:
  - Tour no data center IBX SP3 da Equinix em São Paulo (15 minutos)
  - Why does Microsoft have underwater data centers? (5 minutos)
  - Over 200,000 Servers in One Place! Visiting Hetzner in Falkenstein (Germany) (20 minutos)

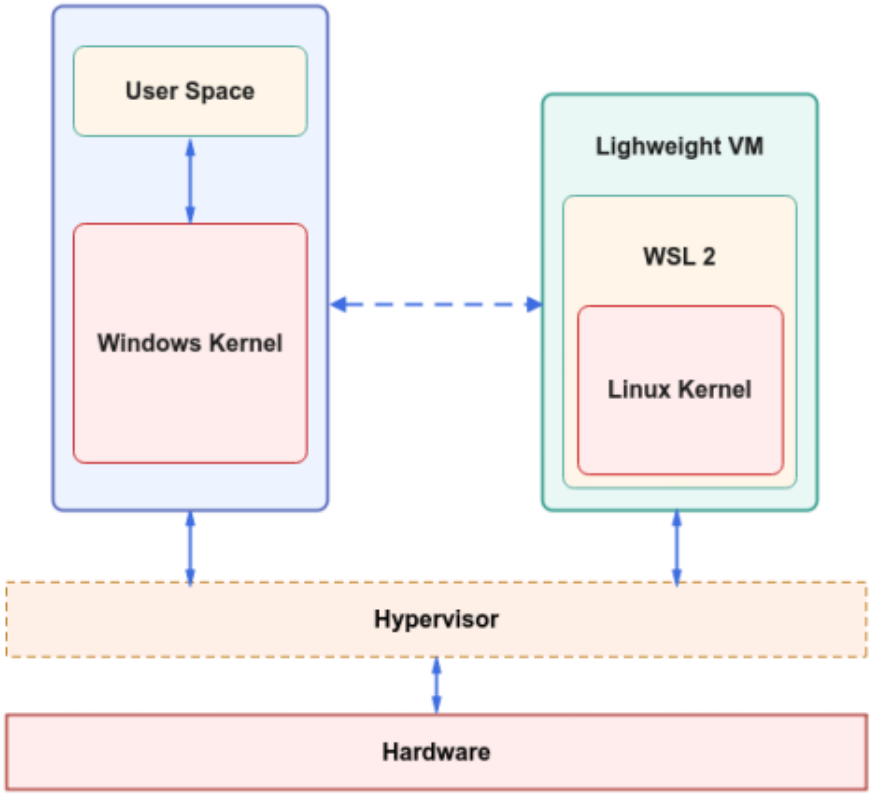
Qual a arquitetura de virtualização do Linux KVM?



Fonte: Kernel-based Virtual Machine

- *Kernel-based Virtual Machine (KVM)* é um **modulo** de código aberto do **kernel Linux** que permite ao mesmo kernel atuar como **Hypervisor**.
- Incorporado ao kernel padrão do Linux na versão 2.6.20 em 2007.
- Requer suporte de microarquiteturas de hardware como Intel VT e AMD-V.
- Suporta vários sistemas operacionais incluindo Windows e MacOS.

E o Windows for Linux Subsystem (WSL)?



- *O Window for Linux Subsystem (WSL)* permite executar **comandos Linux no Windows** sem necessidade de particionamento de disco e boot dual.
- Existem duas versões:
  - ***WSL1***: a mais usada atualmente foi criada como uma camada de tradução de código binário em tempo de execução.
  - ***WSL2***: a mais recente(2019) e baseada em virtualização com o Windows Hypervisor Platform onde com kernel Linux simples (leves).
- ***WSL2*** é mais rápida e apresenta menos erros/bugs.

Fontes: The Underlying Architecture of WSL and why is WSL 2 better e Introduction to WSL 2

Dúvidas, questões ou comentários?

Programa: Introdução a Contêineres e Kubernetes

1. Conceitos Básicos

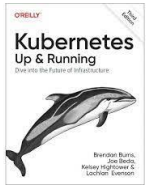
- ✓ Abstrações em Ciência da Computação
- ✓ Virtualização de Computadores
- ✓ MicroVMs e Unikernels

2. Contêineres

- ➢ Origem
- ➢ Fundamentos
- ▪ Criação e execução
- ▪ Registro e reuso
- ▪ Infraestrutura como Código
- ▪ Aplicativos Modernos

3. Kubernetes

- ▪ Origem
- ▪ Arquitetura
- ▪ Pods
- ▪ Abstrações de Recursos
- ▪ Descoberta de Serviços
- ▪ Serviços de Rede
- ▪ Instalação e administração básica
- ▪ Implantação de um caso de uso (exemplo)



Kubernetes Up & Running
Brendan Burns, Joe Beda, Kelsey Hightower, and Lachlan Evenson
Cortesia da VMware Inc.

Contêineres

Origem e Fundamentos

“I once heard that hypervisors are the living proof of operating system’s incompetence”

Glauber Costa, *2012*

LCE: The failure of operating systems and how we can fix it, talk at LinuxCon Europe 2012

Origem

“I don’t think anybody expected it to be as successful as it has been.”
Joe Beda, VMware.

Origem 1/2

- Início dos anos 70 com a tentativa de **isolamento** de aplicativos (teste/produção) **num mesmo ambiente**.
- Permitir testes no mesmo sistema operacional e hardware do ambiente de produção.
- Sistemas operacionais evoluíram no isolamento de usuários para isolamento de arquivos, redes e etc. atraes de melhoras como *chroot, CGroups, Namespaces, Apparmore* e *SELinux profiles*.
- Em 1979 foi lançado o Unix version 7 com capacidade rudimentares de isolamento no sistema de arquivos (chroot). Adotado pelo BSD OS em 1982.
- Free BSD Jails no início dos anos 2000 (particionamentos de recursos). Combinado com o OpenVZ resultou no Solaris Containers em 2004.

Origem 2/2

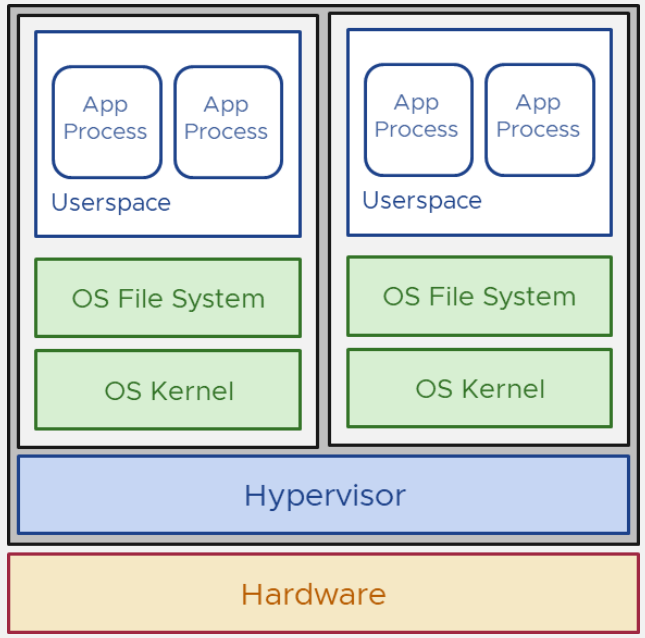
- LXC (Linux Containers) em 2008: a versão mais completa e estável de tecnologia de contêineres.
- Posteriormente o uso de contêineres em larga escala avançou com a chegada do Apache Mesos, Google Borg e o Facebook Tupperware: capacidades de orquestração de contêineres e administração de clusters de servidores.
- Em 2013 a empresa dotCloud criou o projeto de código aberto Docker baseado na sua tecnologia de contêineres. Logo depois a empresa mudou seu nome para Docker.
- O Docker foi criado com base no LXC e libcontainers (open source container stack LMCTFY - Let Me Contain That For You).
- Adoção generalizada pelas comunidades do CNCF.

Fundamentos

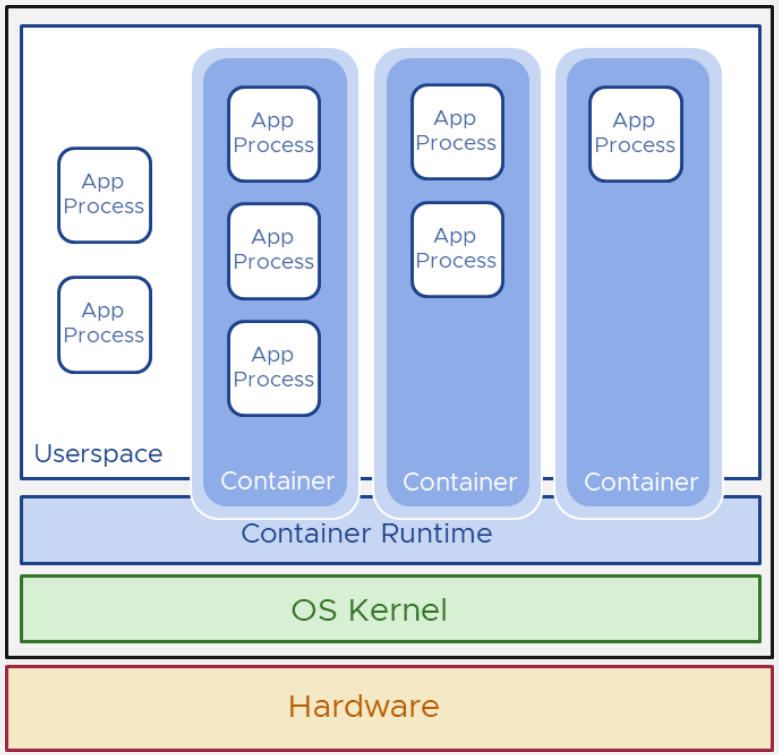
"This containers revolution is changing the basic act of software consumption. It's redefining this much more lightweight, portable unit, or atom, that is much easier to manage… It's a gateway to dynamic management and dynamic systems."

Craig McLuckie, Google.

Virtualização a nível de sistema operacional



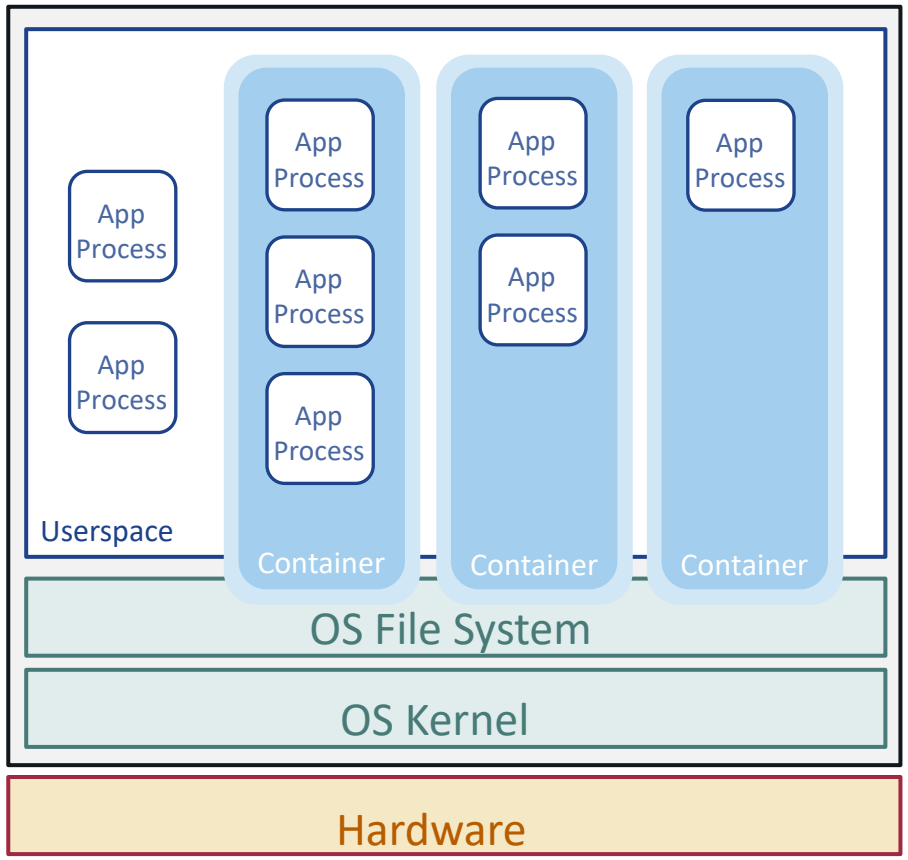
Máquinas Virtuais



Contêineres

- Isolamento de processos a nível de subsistema individual do kernel do sistema operacional.
- Contêineres são mais “leves” que maquinas virtuais: carregam mais rápido e ocupam menos recursos de memória e armazenamento.

Principais benefícios



- ✓ **Infraestrutura imutável:** uma imagem de contêiner não pode ser alterada depois de criada.
- ✓ **Configuração declarativa das imagens de contêineres:** definição do que e não do como facilita a compreensão e é muito menos susceptível a erros.
- ✓ **Desacoplamento:** habilita aplicativos com arquiteturas desacopladas, onde cada componente (contêiner) se encarrega de uma funcionalidade ou serviço (*separation of concerns*) e permite que equipes trabalhem de maneira independente em cada componente ou serviço.

Contêineres versus Maquinas Virtuais

Recurso	Contêiner	Máquina virtual
Sistema operacional	Compartilha o kernel do sistema operacional do host	Tem um kernel próprio
Portabilidade	Mais portátil	Menos portátil
Velocidade	Mais rápido para iniciar e desligar	Mais lento para iniciar e desligar
Uso de recursos	Usa menos recursos	Usa mais recursos
Casos de uso	Bom para aplicativos portáteis e escalonáveis	Bom para aplicativos isolados

Os três pilares do Linux Containers

1. CGroups
2. Namespaces
3. Unionfs

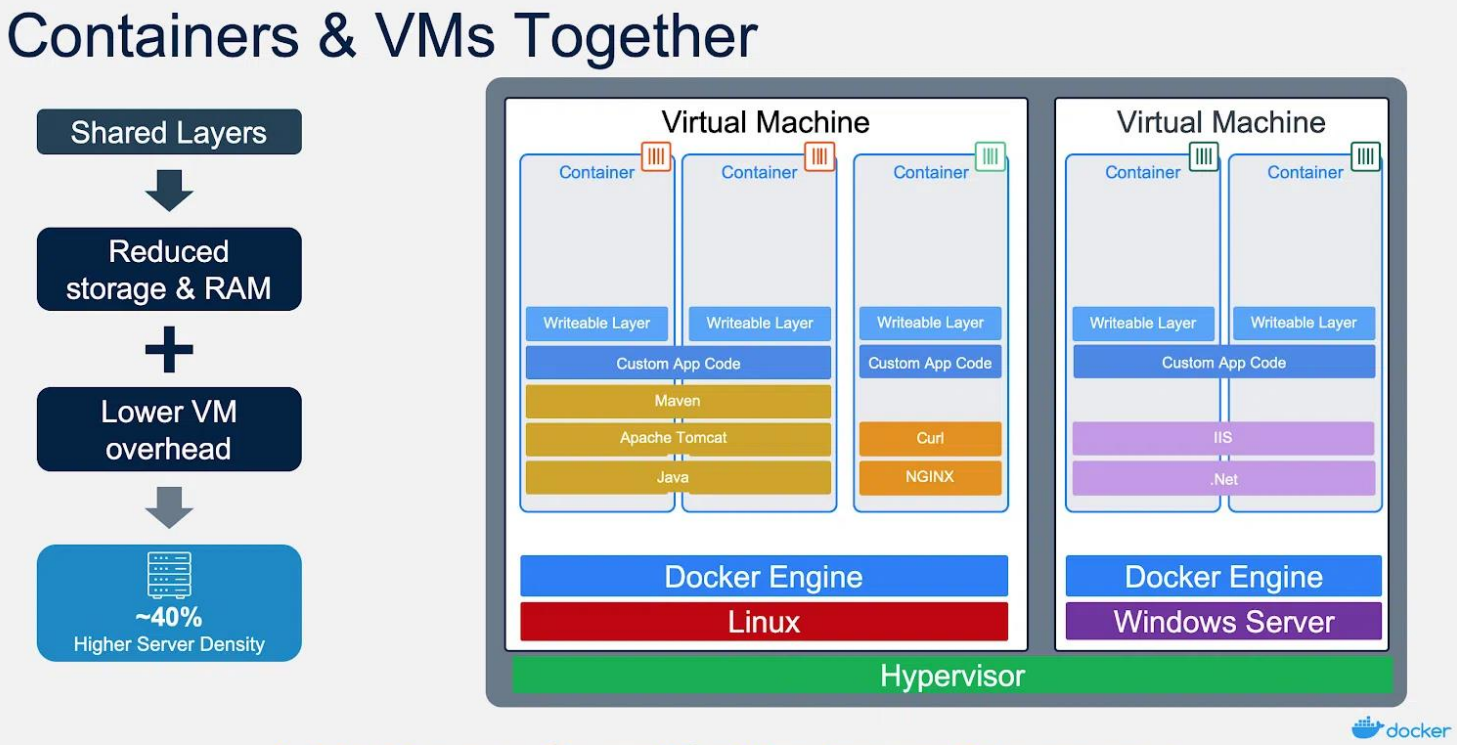
Chroot não!

O *chroot* não permite isolamento seguro

- Em um sistema operacional Unix o diretório raiz (/) é o topo da hierarquia do sistema de arquivos.
- É o home do usuário *root* (*sysadmin*).
  - Cada processo tem sua própria definição de onde está seu próprio diretório raiz
  - Por default é o raiz do sistema.
- Mas pode ser mudando com a chamada de sistema *chroot()*.
  - chroot() altera de maneira aparente o diretório raiz do processo corrente e seus processos filhos.
  - Apenas altera a referência de caminhos de busca.
- Diretório atual não é modificado e os caminhos relativos de busca podem fazer referência e acessar qualquer outro diretório ou arquivos fora do “novo” diretório raiz.
- O próprio processo poder executar *chroot()* para reconfigurar seu diretório raiz para a raiz do sistema (/).

Dúvidas, questões ou comentários?

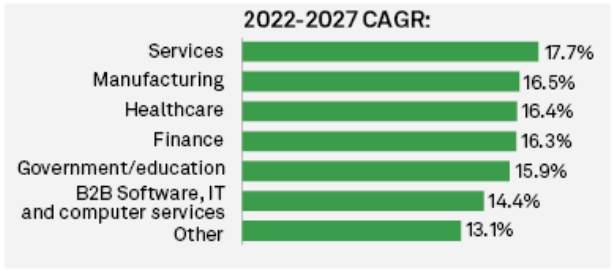
VMs ou Contêineres?
Não é um ou outro, são ambos!

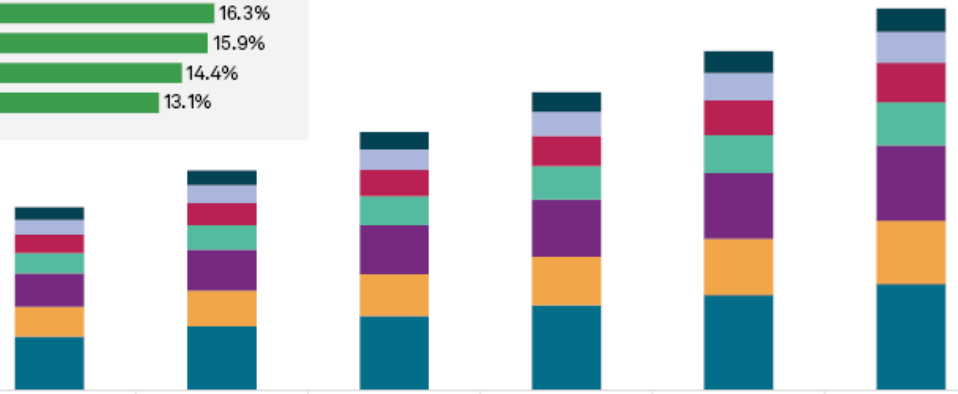


Added bonus: fewer “Golden Images” to manage

De uma ferramenta emergente a uma ampla adoção empresarial

Application container ecosystem CAGR and revenue by vertical market ($M)



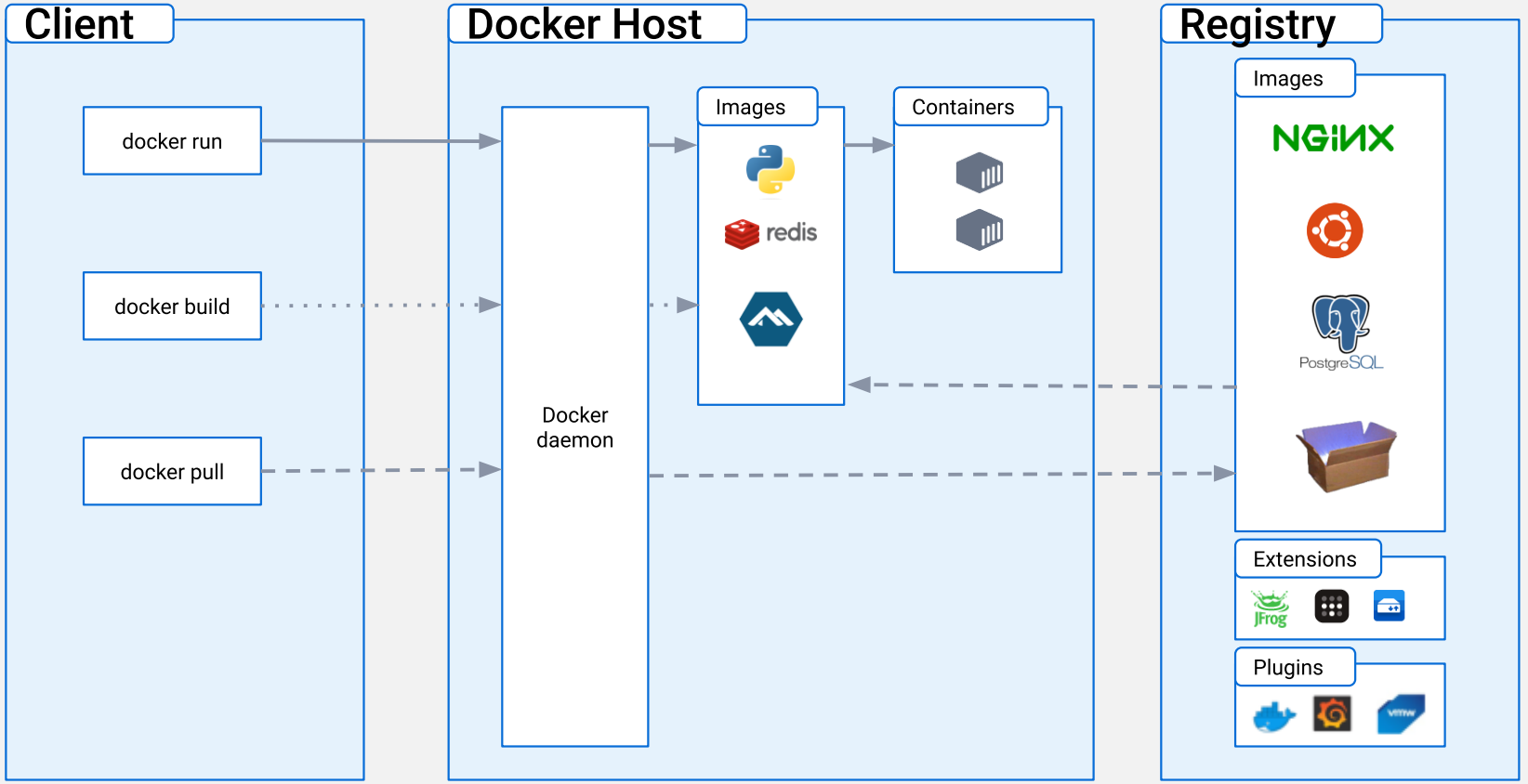


	2022	2023E	2024E	2025E	2026E	2027E
Other	345.9	406.1	465.7	524.1	581.6	638.8
Government/education	399.8	480.8	565.1	652.2	742.6	837.4
Healthcare	501.7	605.8	714.8	828.2	946.7	1,071.8
Finance	551.6	665.8	785.3	909.4	1,039.0	1,175.8
Services	913.5	1,115.6	1,331.1	1,559.0	1,801.0	2,060.3
Manufacturing	799.4	966.4	1,141.8	1,324.6	1,515.9	1,718.2
B2B Software, IT and computer services	1,480.1	1,757.4	2,038.8	2,322.3	2,609.1	2,903.1

CAGR = compound annual growth rate; B2B = business to business; E = estimates.
Source: Applied Infrastructure & DevOps Market Monitor: Application Container Ecosystem, Q1 2023.

Fonte: Application containers still driving broader market growth, innovation

Docker: Componentes e Arquitetura



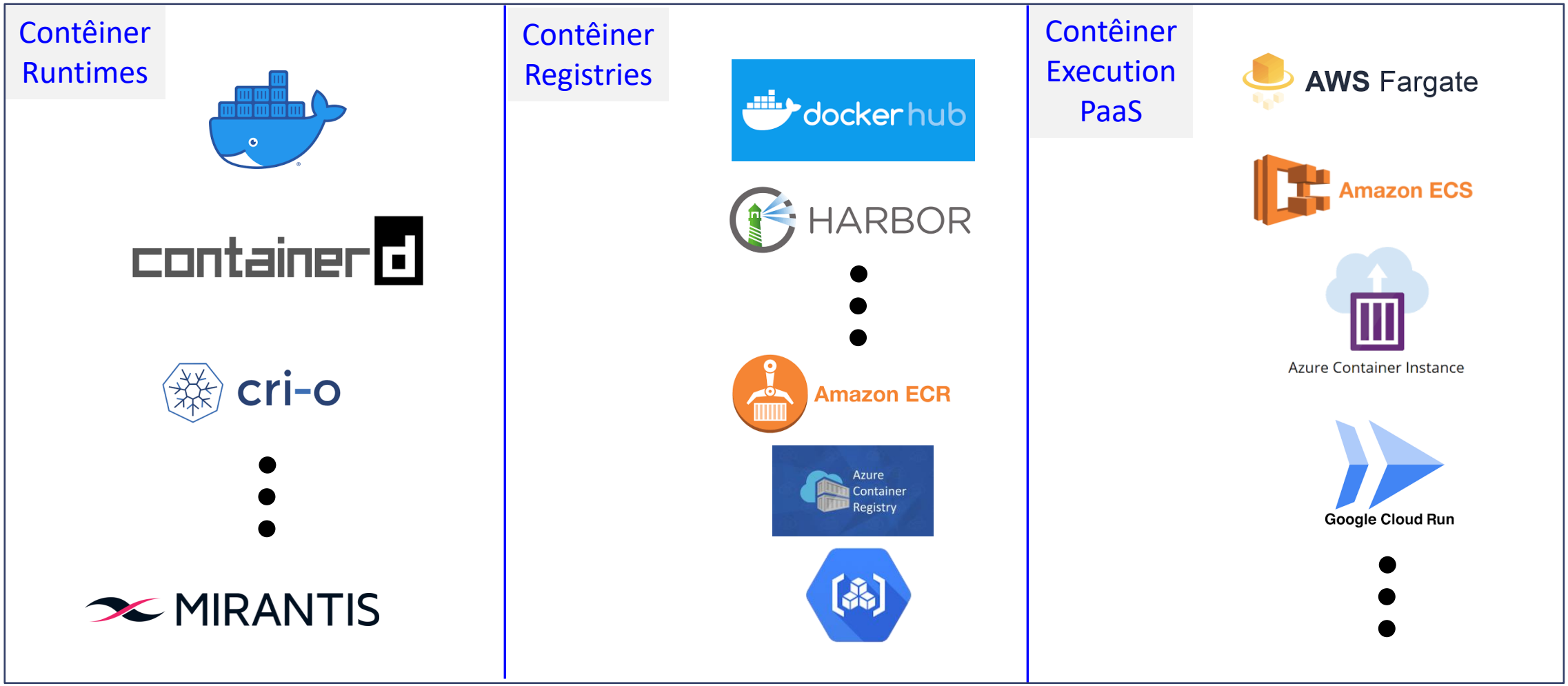
Fonte: Docker architecture

Open Container Initiative[1]

- Projeto de código aberto patrocinado pela Linux Foundation
- Define padrões para formatos e módulos de execução (*runtimes*) para Contêineres.
- Início baseado na especificação da empresa Docker que abriu seu formato de imagens e o código de seu modulo de execução.
- Possui três especificações:
  1. Especificação de módulos de execução
  2. Especificação de imagens
  3. Especificação de distribuição

1- https://opencontainers.org/about/overview/

Ecossistema



Dúvidas, questões ou comentários?

Atividades para a próxima aula

- Ler Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications e responder as seguintes perguntas:
  1. Quais tipos de aplicações são apropriadas para utilizaram Unikernels? Por que? Mencionar exemplos.
  2. Na sua opinião quais as principais limitações de uso de Unikernels?
  3. Conforme os resultados reportados no artigo, quais são as diferenças de desempenho entre Unikernels e Contêiners?
  4. Na sua opinião, quais os critérios para uma aplicação utilizar Unikernel ou Contêiner? Explique com exemplos.
- Assistir vídeos com tutoriais de números 1 ao 7 do Docker Tutorial:
  1. Introdução (baixar e instalar)
  2. Imagens, contêineres e servidor Docker
  3. Comandos básicos
  4. Listar, criar e executar contêineres
  5. Visualizar logs, suspender e abortar execução.
  6. Remover e inspecionar contêineres.
  7. Executar um comando dentro de um contêiner em execução.