



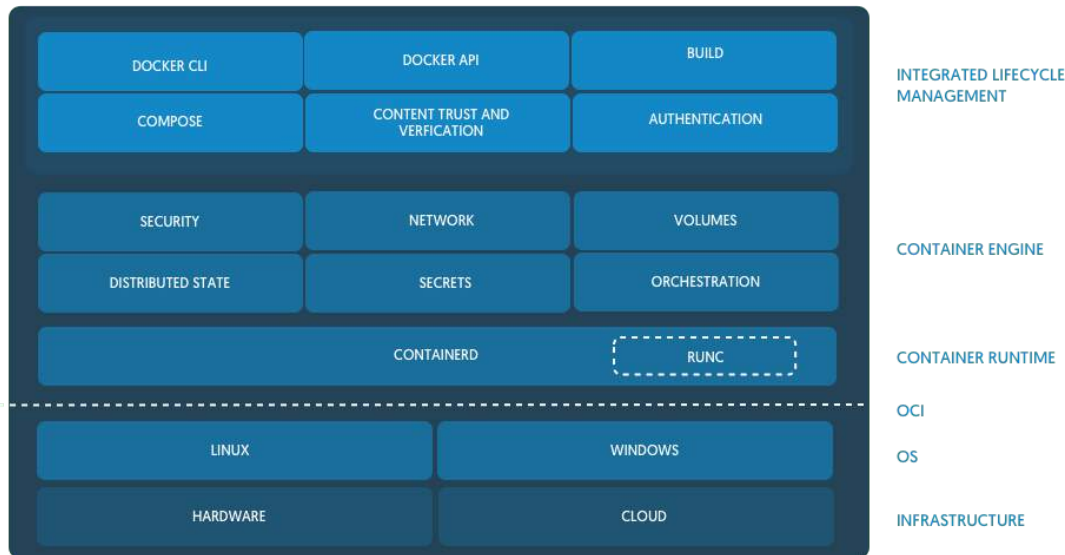
Um *Container Runtime* com ênfase em **simplicidade**,
robustez e **portabilidade**



Um **container runtime** disponibilizado como um projeto de Software de Código Aberto (#OSS) que **gerencia o Ciclo de Vida Completo do contêiner**, desde a transferência e armazenamento de imagens para a execução de contêineres até a supervisão do armazenamento de baixo nível, e das configurações de network.

containerd

- Originalmente construído como um ponto de integração para *runtimes* OCI* (*runC*) mas absorveu funcionalidades necessárias para plataformas como Docker e K8s.
- Fornece uma “camada cliente” para as aplicações se abstraírem do nível de Kernel (API-driven)



Fonte: <https://blog.docker.com/2017/08/what-is-containerd-runtime/>

Uma Breve História



Jun 2015

• Docker doou **runC** para a Open Container Initiative (OCI)

Containerd - Daemon Controle para **runC**

Dez 2015

<https://blog.docker.com/2015/12/containerd-daemon-to-control-runc/>

Containerd “0.2”, Docker 1.11

Abr 2016

<https://blog.docker.com/2016/04/docker-engine-1-11-runc/>

Dez 2016

• Expansão do Projeto #OSS **Containerd 1.0**

<https://blog.docker.com/2016/12/introducing-containerd/>



Containerd doado para a CNCF

Mar 2017

<https://blog.docker.com/2017/03/docker-donates-containerd-to-cncf/>

Sir Robert Dudley's *Dell' Arcano Del Mar*, 1646. Image courtesy of the [Harvard Map Collection](#).

Cloud Native Computing Foundation Anuncia a Graduação do Projeto containerd



<https://www.cncf.io/announcement/2019/02/28/cncf-announces-containerd-graduation/>

Cloud Native Computing Foundation

- Parte da Fundação Linux, Sem fins lucrativos; fundada em Dez 2015

Graduados



kubernetes
Orquestração



Prometheus
Monitoramento



envoy
Serviço de Proxy



CoreDNS
Descoberta de Serviços



Container Runtime



fluentd
Logging



OPENTRACING
API Rastreabilidade Distribuída



Mensageria



Remote Procedure Call



LINKERD

Service Mesh



Container Runtime



Gerenciamento de Pacotes



CNI
API de Rede



ROOK
Armazenamento



JAEGER
Rastreabilidade Distribuída



HARBOR
Registro



Software Update Spec



etcd
Armazenamento Chave/Valor



Segurança



Open Policy Agent
Política



Armazenamento



cri-o
Container Runtime

- Membros Platinum :

Alibaba Cloud



DELL Technologies



Google Cloud



IBM Cloud



Microsoft Azure

ORACLE

Pivotal



SAMSUNG
SAMSUNG SDS



vmware



Arquitetura

Direcionadores da Arquitetura

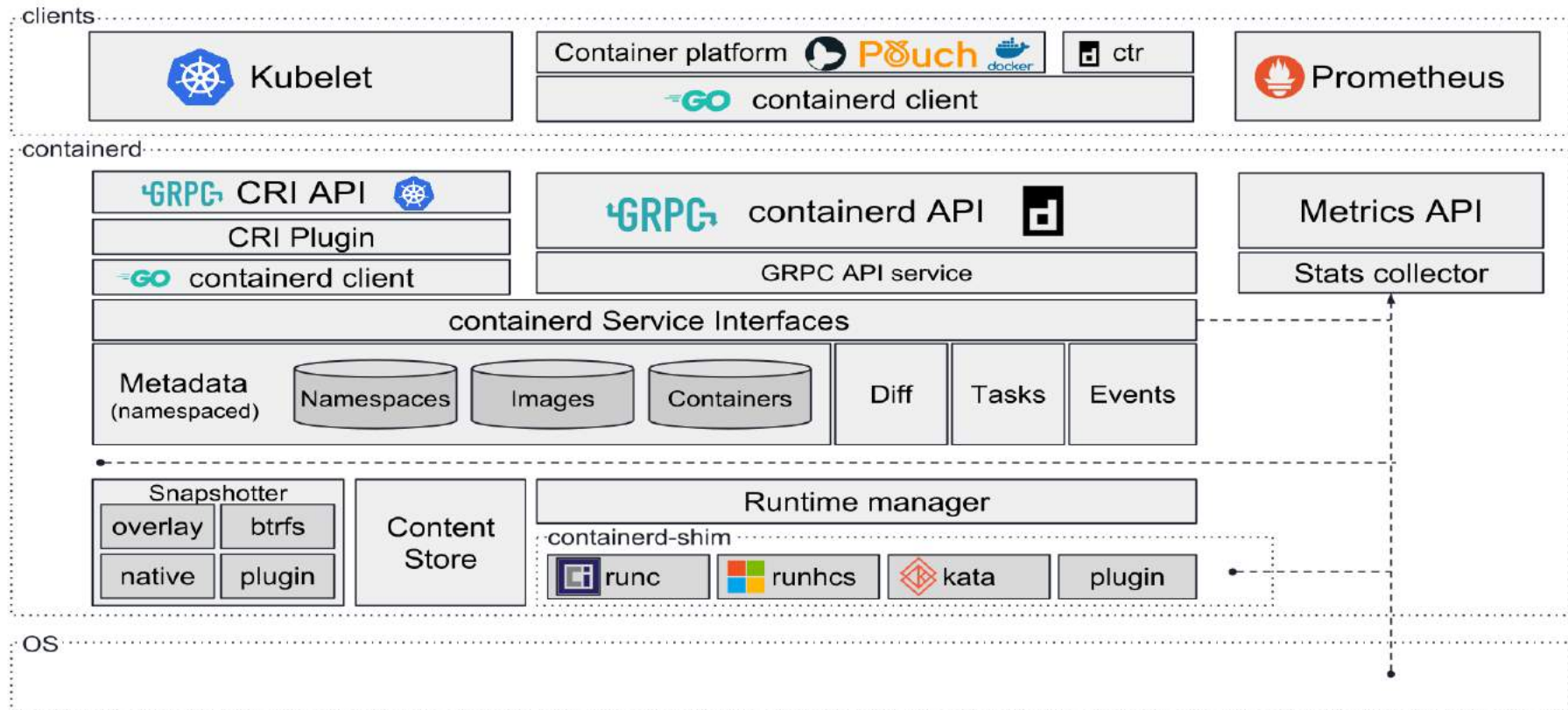
Requisitos

- **Just Enough** : Use apenas o necessário
- **Just in Time** : Agilidade *runtime*
- **Desacoplamento**
- **Compatibilidade OCI** e relação configuração direta com OCI
- Métricas com **Prometheus**
- Tecnologias conhecidas

Objetivos Técnicos

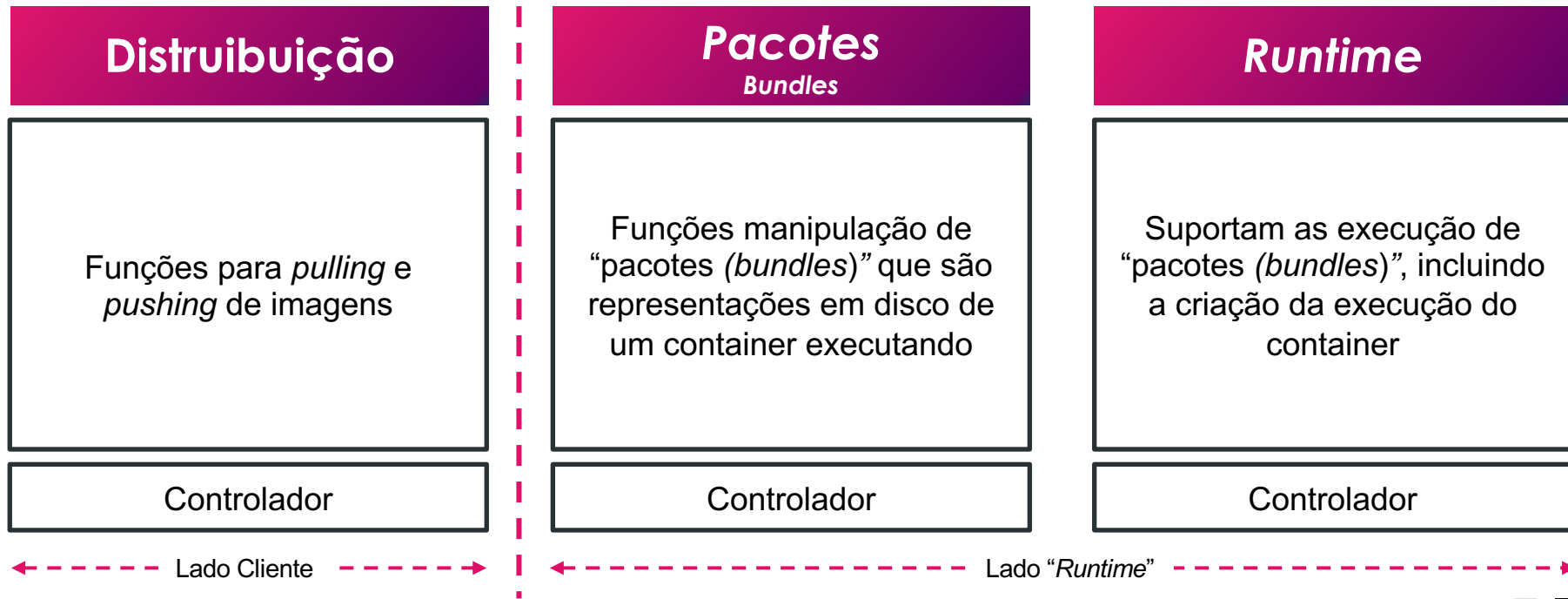
- **Estabilidade e desempenho** nas funções principais para os containers
- **Desacoplamento e modularidade** nos sistemas principais (*imagens, filesystem, runtime*) permitindo reuso e conectividade
- **Suporte total a OCI**
- **API baseada em gRPC**

Arquitetura



Subsistemas

Conjunto de Componentes da Arquitetura



Modulos

- Além dos subsistemas, existem vários componentes que podem cruzar os limites do subsistema

Executor

Realmente implementa o mecanismo de execução

Supervisor

Monitora e reporta o estado dos containers

Metadata

Armazena os metadados (images e *bundles*)

Conteúdo

Fornece o acesso ao armazenamento de conteúdos (como imagem)

Snapshotter

Gerencia os sistemas de “snapshot” para as imagens de containers

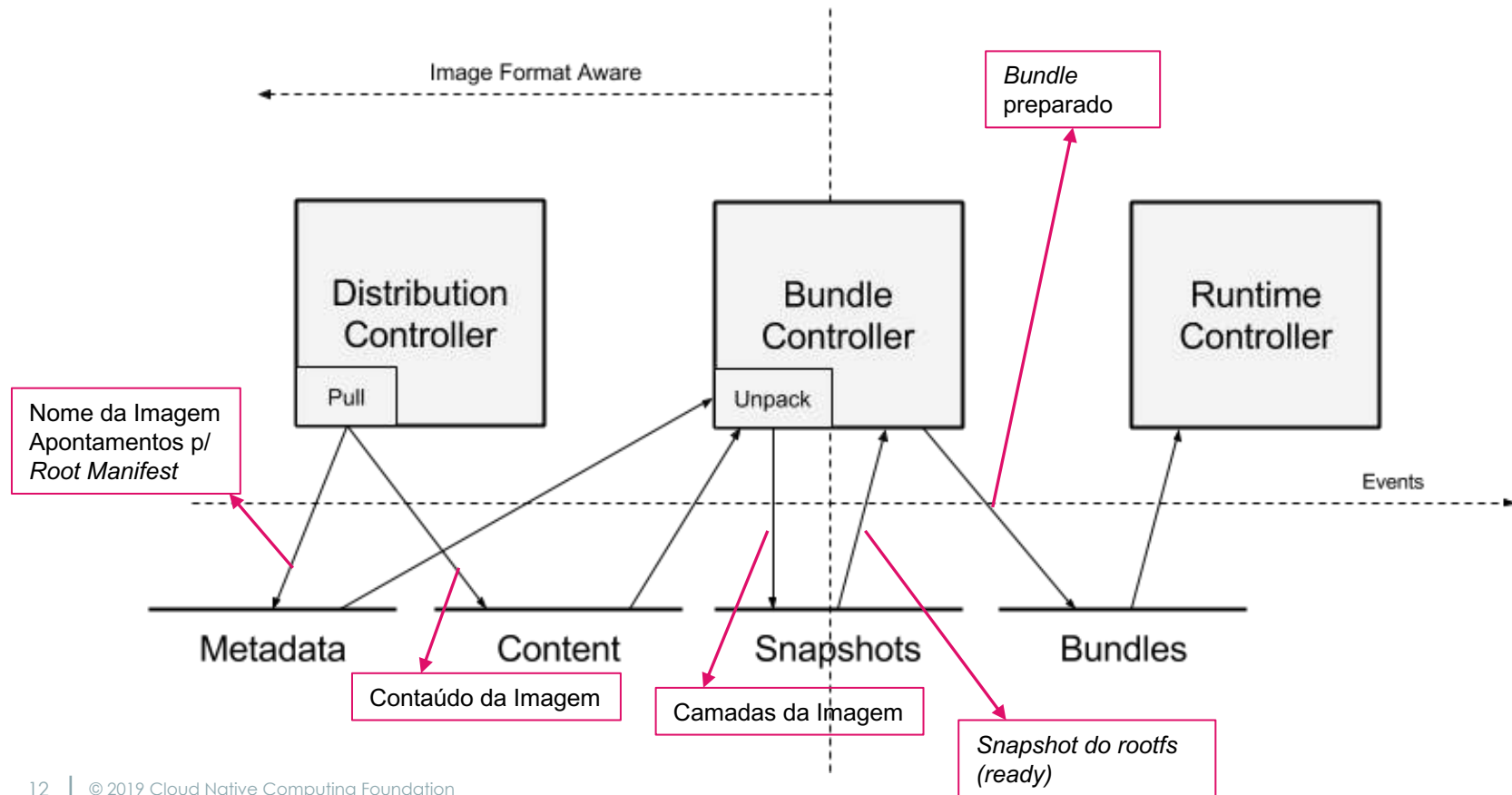
Eventos

Suporta a coleção e consumo de eventos (orientado a eventos, pub/sub pattern)

Tarefas

Permitem ao cliente gerenciar o estado de um container executando

Criação de um *Bundle*



Executando

Conectando com Containerd

main.go

```
package main

import (
    "log"

    "github.com/containerd/containerd"
)

func main() {
    if err := redisExample(); err != nil {
        log.Fatal(err)
    }
}

func redisExample() error {
    client, err := containerd.New("/run/containerd/containerd.sock")
    if err != nil {
        return err
    }
    defer client.Close()
    return nil
}
```

Criação de um novo Cliente Containerd

Contexto com namespace

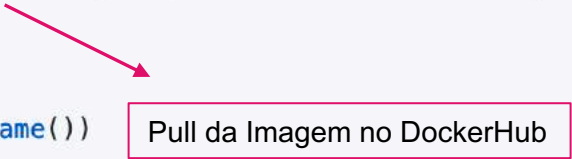
```
ctx := namespaces.WithNamespace(context.Background(), "example")
```

Obtendo uma Imagem

main.go

```
func redisExample() error {
    client, err := containerd.New("/run/containerd/containerd.sock")
    if err != nil {
        return err
    }
    defer client.Close()

    ctx := namespaces.WithNamespace(context.Background(), "example")
    image, err := client.Pull(ctx, "docker.io/library/redis:alpine", containerd.WithPullUnpack)
    if err != nil {
        return err
    }
    log.Printf("Successfully pulled %s image\n", image.Name())
    return nil
}
```



Criando um Spec e um Container OCI

main.go

```
container, err := client.NewContainer(  
    ctx,  
    "redis-server",  
    containerd.WithNewSnapshot("redis-server-snapshot", image),  
    containerd.WithNewSpec(oci.WithImageConfig(image)),  
)  
if err != nil {  
    return err  
}  
defer container.Delete(ctx, containerd.WithSnapshotCleanup)
```


Executando o Container

main.go

```
task, err := container.NewTask(ctx, cio.NewCreator(cio.WithStdio))
if err != nil {
    return err
}
defer task.Delete(ctx)
```

Task em Estado "Criado"

```
exitStatusC, err := task.Wait(ctx)
if err != nil {
    return err
}

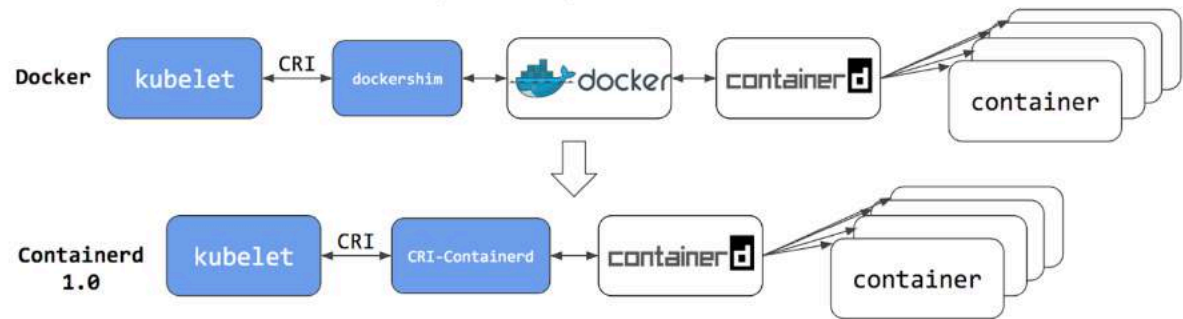
if err := task.Start(ctx); err != nil {
    return err
}
```

Task Executando

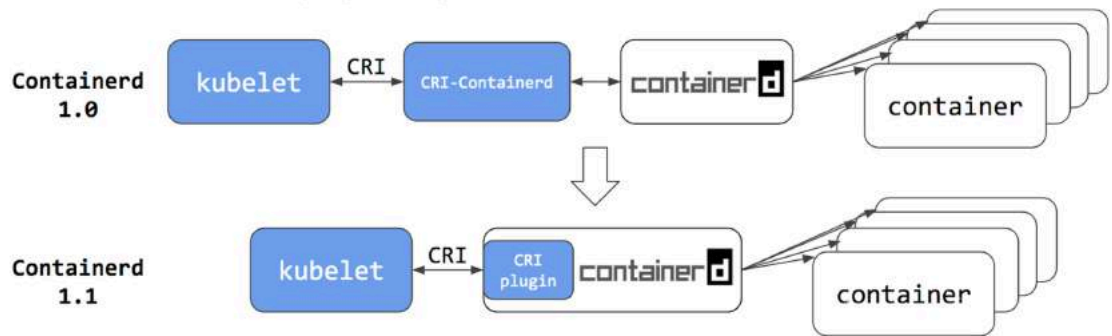
Adotando

Integração do Containerd com Kubernetes

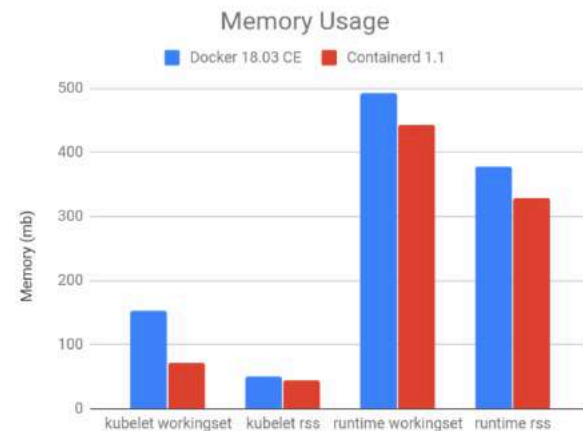
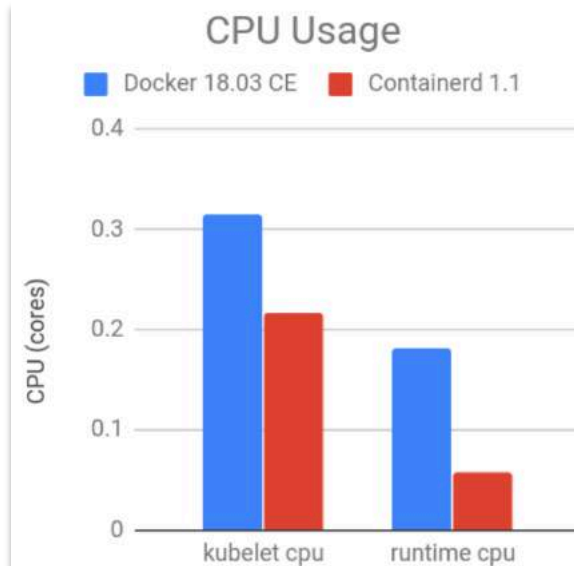
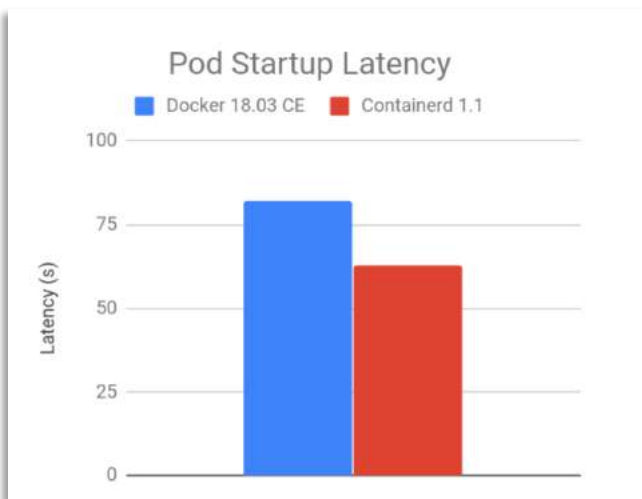
Containerd 1.0 - CRI-Containerd (end of life)



Containerd 1.1 - CRI Plugin (current)



Testes de Performance



Provedores



Obrigado

Paulo Alberto Simoes

Principal Cloud Solution Engineer at
Oracle

[@pasimoes](#)



Paulo Alberto Simoes .:

Principal Solution Engineer | Sr Enterprise
Architect | Software Developer | Cloud-Nativ...

