

## This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

## Objetivos da Aula

 Introdução ao Kubernetes.

## Competências da Aula

Esta aula possui conteúdo que atende aos objetivos da seguinte certificação:

### Certified Kubernetes Administrator (CKA)

Core Concepts (19% of exam);



# Introdução ao Kubernetes

## Anotações

---

---

---

---

---

---

---

---

---

---

---

## Kubernetes

O Kubernetes é uma plataforma portátil e extensível de código aberto para gerenciamento de cargas de trabalho e serviços containerizados, que facilita tanto a configuração declarativa quanto a automação. Tem um ecossistema grande e de rápido crescimento. Os serviços suporte e ferramentas do Kubernetes, estão amplamente disponíveis.



### Sobre o Kubernetes

O Google abriu o projeto Kubernetes em 2014. O Kubernetes baseia-se em uma década e meia de experiência do Google com a execução de cargas de trabalho de produção em grande escala, combinadas com as melhores ideias e práticas da comunidade.

Por que preciso do Kubernetes e o que ele pode fazer?

O Kubernetes possui vários recursos. Pode ser pensado como:

- ✓ Uma plataforma de contêiner;
- ✓ Uma plataforma de microsserviços;
- ✓ Uma plataforma de nuvem portátil e muito mais.

O Kubernetes fornece um ambiente de gerenciamento centrado em contêineres. Ele orquestra a infraestrutura de computação, rede e armazenamento em nome das cargas de trabalho dos usuários. Isso fornece grande parte da simplicidade da Plataforma como Serviço (PaaS), com a flexibilidade da Infraestrutura como Serviço (IaaS) e permite a portabilidade entre os provedores de infraestrutura.

### Por que as empresas precisam de Kubernetes?

Os tempos de execução dos contêineres foram projetados, como uma alternativa para a execução de imagens de máquinas virtuais imutáveis (VM). Imagens de VM são muito mais pesadas (requerem mais recursos) do que contêineres e, portanto, precisam de mais servidores para serem implantados. Em contraste, as modernas tecnologias de contêineres simplificam a execução de milhares de contêineres leves em um único host, o que leva a economia radical de recursos de computação.

### Uso de Containers

Os contêineres permitem separar aplicativos da infraestrutura subjacente e isolá-los do ambiente do host, usando um sistema de arquivos autônomo, redes virtuais e dependências. Esse isolamento torna os contêineres muito mais portáteis e fáceis de implantar que as Vms.

No entanto, as tecnologias de contêineres, como o Docker, não abordam totalmente o desafio de executar aplicativos em contêiner na produção. O Docker ofereceu o Swarm (e ainda faz) para a orquestração, mas o Swarm, no final das contas, não ofereceu tanto quanto o K8s. Pense nisso por um momento: os aplicativos de produção reais incluem vários contêineres implantados em vários hosts de servidor. Quando você gerencia centenas ou milhares de contêineres em vários nós em produção, você precisa escaloná-los dependendo da carga do aplicativo, habilitar comunicação e acesso externo (por exemplo, por meio de microsserviços), armazenamento, execução de verificações regulares de integridade, gerenciamento de atualizações e muitas outras tarefas.

Essas são todas as tarefas de orquestração que não saem da caixa nos tempos de execução do contêiner. Desenvolver sua própria estrutura de orquestração para executar essas tarefas seria uma sobrecarga desnecessária para o seu negócio.

## Recursos do Kubernetes: Escalabilidade

A escalabilidade está entre as principais preocupações das aplicações modernas de nível de produção, expostas a milhões de usuários em potencial. Quando se trata de executar aplicativos em clusters de centenas ou até milhares de servidores, o escalonamento se torna uma tarefa de administração complexa que envolve muitos pré-requisitos e ressalvas.

### Escalabilidade

Algumas das perguntas que podem surgir ao longo do caminho:

- ✓ O número desejado de instâncias de aplicativos está em execução?
- ✓ Quantas dessas instâncias são saudáveis e estão prontas para atender ao tráfego?
- ✓ Qual é a carga atual do aplicativo? Quantas réplicas são necessárias para atender essa carga?
- ✓ Quantos nós estão disponíveis atualmente para agendamento?
- ✓ Quantos recursos estão disponíveis no cluster?

Estas são apenas algumas perguntas que podem tornar um administrador de cluster sem ferramentas apropriadas e enlouquecer. A solução óbvia é a automação, e é aí que o Kubernetes realmente brilha.

Por exemplo, os controladores da plataforma, como implantações, podem monitorar quantos aplicativos estão sendo executados no cluster e, se por algum motivo, esse número for diferente do estado desejado, o Kubernetes aumentará ou diminuirá a implantação para alcançá-lo. Sob o capô, o Kubernetes também manterá um registro dos nós e recursos disponíveis para agendar de maneira inteligente seus aplicativos durante o processo de escalonamento. Ao mesmo tempo, você sempre pode fazer uma escala manual, usando as ferramentas de linha de comando do Kubernetes (kubectl).

## Recursos do Kubernetes: Rede eficiente e descoberta de serviço

O objetivo da rede Kubernetes é transformar contêineres em “hosts virtuais” autênticos que podem se comunicar entre si através de nós, combinando os benefícios de VMs, contêineres e microsserviços. A rede do Kubernetes é baseada em várias camadas, atendendo ao seguintes objetivos:

- ✓ Comunicação contêiner à contêiner;
- ✓ Comunicação Pod à Pod;
- ✓ Serviços.

### Rede eficiente

- ✓ **Comunicação contêiner à contêiner:** No host local e utilizando o namespace de rede de um pod. Essa camada de rede permite que as interfaces de rede do contêiner, contenham contêineres fortemente acoplados que podem se comunicar entre si em portas especificadas, muito semelhantes às aplicações convencionais;
- ✓ **Comunicação de pod-para-pod:** Que permite a comunicação de pods nos nós. O Kubernetes pode transformar o cluster em uma rede virtual, onde todos os pods podem se comunicar uns com os outros, não importando em quais nós eles se conectam;
- ✓ **Serviços:** Uma abstração de Serviço define uma política (microsserviço) para acessar pods por outros aplicativos. Os serviços agem como balanceadores de carga que distribuem solicitações em pods de back-end gerenciados pelo serviço.

O Kubernetes é muito flexível sobre soluções de rede que você pode usar. No entanto, a plataforma impõe os seguintes princípios para as interfaces de rede no nível do cluster:

- ✓ Todos os contêineres podem se comunicar uns com os outros, sem o NAT;
- ✓ Todos os nós podem se comunicar com todos os containers (e vice-versa), sem o NAT;
- ✓ O IP visto por um contêiner é o mesmo IP visto por outros contêineres.

## Recursos do Kubernetes: Suporte nativo para aplicativos Stateful

O Kubernetes vêm com um suporte nativo para aplicativos stateful, como bancos de dados e armazenamentos de valores-chave. Em particular, seu subsistema de volumes persistentes, fornece uma API que abstrai detalhes da infraestrutura de armazenamento subjacente (por exemplo, AWS EBS, Azure Disk, etc.), permitindo que usuários e administradores concentrem-se na capacidade de armazenamento e tipos de armazenamento que seus aplicativos consumirão, do que os detalhes sutis da API de cada provedor de armazenamento.

### Volumes Persistente

Volumes persistentes permitem reservar a quantidade necessária de recursos, usando reivindicações de volume persistentes. A reivindicação é vinculada automaticamente à volumes que correspondem ao tipo de armazenamento, à capacidade e a outros requisitos especificados na reivindicação. As reclamações também são automaticamente desvinculadas se um volume correspondente não existir. Esse recurso permite reservar de forma eficiente o armazenamento por aplicativos em execução no cluster do Kubernetes.

Outro ótimo recurso para aplicativos com informações de estado é o provisionamento de armazenamento dinâmico. No Kubernetes, os administradores podem descrever vários tipos de armazenamento disponíveis no cluster e suas políticas específicas de recuperação, montagem e backup. Depois que essas classes de armazenamento forem definidas, o Kubernetes poderá provisionar automaticamente a quantidade solicitada de recursos do provedor de armazenamento subjacente, como o AWS EBS ou o disco do Azure.



## Recursos do Kubernetes: Extensibilidade e capacidade de conexão


O Kubernetes enfatiza a filosofia de extensibilidade e capacidade de conexão, o que significa que a plataforma preserva a escolha do usuário e a flexibilidade, onde isso é importante. O Kubernetes visa suportar a maior variedade de cargas de trabalho e tipos de aplicativos possíveis, e ser fácil de integrar com qualquer ambiente e ferramenta.

## Plugins

Alguns frameworks plugin suportados pelo Kubernetes incluem:

- ✓ Plugins CNI (Container Network Interface): Implementam o modelo de rede CNI e são projetados para interoperabilidade;
- ✓ Os plug-ins de volume fora da árvore, como a Container Storage Interface (CSI) e o FlexVolume. Eles permitem que os fornecedores de armazenamento criem plug-ins de armazenamento personalizados, sem adicioná-los ao repositório do Kubernetes.

### Recapitulando

 Introdução ao Kubernetes.

#### Anotações

---

---

---

---

---

---

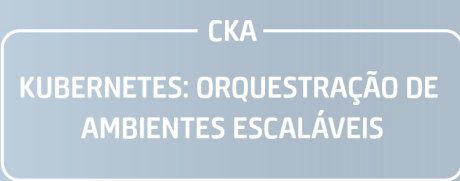
---

---

---

---

---

[illegible]

## Objetivos da Aula

- Conhecer os componentes do Master;
- Conhecer os componentes dos Nodes.

## Competências da Aula

Esta aula possui conteúdo que atende aos objetivos da seguinte certificação:

### Certified Kubernetes Administrator (CKA)

Core Concepts (19% of exam);



# Componentes do Master

Anotações

---

---

---

---

---

---

---

---

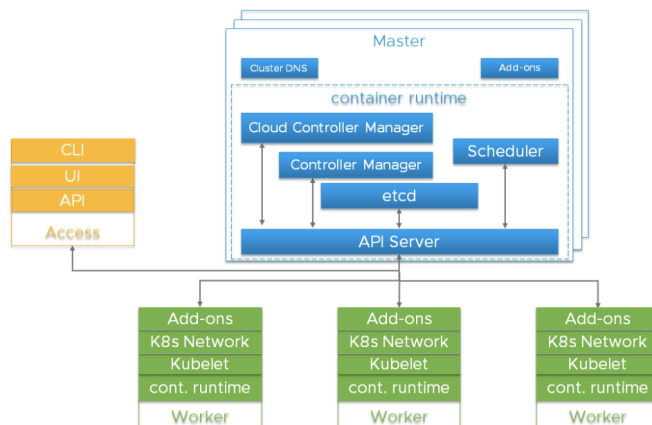
---

---

---

---

## Arquitetura do Cluster Kubernetes



### Sobre o Kubernetes

O Google abriu o projeto Kubernetes em 2014. O Kubernetes baseia-se em uma década e meia de experiência do Google com a execução de cargas de trabalho de produção em grande escala, combinadas com as melhores ideias e práticas da comunidade.

Por que preciso do Kubernetes e o que ele pode fazer?

O Kubernetes possui vários recursos. Pode ser pensado como:

- ✓ Uma plataforma de contêiner;
- ✓ Uma plataforma de microsserviços;
- ✓ Uma plataforma de nuvem portátil e muito mais.

O Kubernetes fornece um ambiente de gerenciamento centrado em contêineres. Ele orquestra a infraestrutura de computação, rede e armazenamento em nome das cargas de trabalho dos usuários. Isso fornece grande parte da simplicidade da Plataforma como Serviço (PaaS) com a flexibilidade da Infraestrutura como Serviço (IaaS) e permite a portabilidade entre os provedores de infraestrutura.

## Componentes do Cluster Kubernetes

### Componentes do Master

- ✓ Os componentes principais fornecem o plano de controle do cluster. Os componentes mestres tomam decisões globais sobre o cluster, detectam e respondem a eventos de cluster;
- ✓ Os componentes principais podem ser executados em qualquer máquina no cluster. No entanto, para simplificar, os scripts de configuração geralmente iniciam todos os componentes principais na mesma máquina e não executam contêineres de usuários nesta máquina.

### Anotações

---

---

---

---

---

---

---

---

---

---

---

---

## Componentes do Cluster Kubernetes

**kube-apiserver**

etcd

kube-scheduler

kube-controller

cloud-controller

- ✓ Componente no mestre que expõe a API do Kubernetes. É o front-end do plano de controle do Kubernetes;
- ✓ Ele é projetado para dimensionar horizontalmente, isto é, é dimensionado com a implantação de mais instâncias.

### kube-apiserver

O servidor da API do Kubernetes valida e configura dados para os objetos da API que incluem pods, serviços, controladores de replicação e outros. O API Server presta serviços às operações REST e fornece a interface para o estado compartilhado do cluster, através do qual todos os outros componentes interagem.



## Componentes do Cluster Kubernetes

kube-apiserver

**etcd**

kube-scheduler

kube-controller

cloud-controller

- ✓ Armazenamento de valor de chave consistente e altamente disponível, usado como armazenamento de apoio do Kubernetes para todos os dados de cluster;
- ✓ Se o seu cluster do Kubernetes usa o etcd como seu armazenamento de apoio, certifique-se de ter um plano de backup para esses dados.

### etcd

O Kubernetes usa o etcd para armazenar todos os seus dados - seus dados de configuração, seu estado e seus metadados. O Kubernetes é um sistema distribuído, então ele precisa de um armazenamento de dados distribuído como o etcd. O etcd permite que qualquer um dos nós no cluster do Kubernetes leia e grave dados.

## Componentes do Cluster Kubernetes

kube-apiserver

etcd

kube-scheduler

kube-controller

cloud-controller

- ✓ Componente no mestre que observa os pods recém-criados, que não têm nenhum nó designado e seleciona um nó para eles serem executados;
- ✓ Os fatores considerados para decisões de programação, incluem requisitos de recursos individuais e coletivos, restrições de hardware / software / política, especificações de afinidade e antiafinidade, localidade dos dados, interferência entre cargas de trabalho e prazos finais.

### kuber-scheduler

O agendador do Kubernetes é uma função específica da carga de trabalho, rica em políticas e com reconhecimento de topologia que afeta significativamente a disponibilidade, o desempenho e a capacidade. O escalonador precisa levar em conta requisitos de recursos individuais e coletivos, requisitos de qualidade de serviço, restrições de hardware / software / política, especificações de afinidade e antiafinidade, localidade de dados, interferência entre cargas de trabalho, prazos e assim por diante. Os requisitos específicos da carga de trabalho serão expostos por meio da API, conforme necessário.

## Componentes do Cluster Kubernetes

kube-apiserver

etcd

kube-scheduler

kube-controller

cloud-controller

- ✓ Componente no mestre que executa os controladores;
- ✓ Logicamente, cada controlador é um processo separado, mas para reduzir a complexidade, todos eles são compilados em um único binário e executados em um único processo.

### kube-controller-manager

O cloud-controller-manager permite que o código do fornecedor de serviços na nuvem e o código do Kubernetes evoluam independentemente uns dos outros. Em versões anteriores, o código principal do Kubernetes dependia de código específico do provedor de nuvem para funcionalidade. Em versões futuras, o código específico para os fornecedores de nuvem deve ser mantido pelo próprio fornecedor da nuvem e vinculado ao gerenciador do controlador de nuvem, durante a execução do Kubernetes.

Os seguintes controladores possuem dependências de provedor de nuvem:

- ✓ **Controlador de nó:** Para verificar o provedor de nuvem para determinar se um nó foi excluído na nuvem, após parar de responder;
- ✓ **Controlador de rotas:** Para configurar rotas na infraestrutura de nuvem subjacente;
- ✓ **Controlador de serviço:** Para criar, atualizar e excluir balanceadores de carga do provedor de nuvem;
- ✓ **Controlador de volume:** Para criar, anexar, montar volumes e interagir com o provedor de nuvem para orquestrar volumes.

kube-apiserver

etcd

kube-scheduler

kube-controller

cloud-controller

- ✓ O Cloud-controller-manager executa controladores que interagem com os provedores de nuvem subjacentes. O cloud-controller-manager executa apenas loops de controladores específicos do provedor de nuvem. Você deve desabilitar esses loops de controlador no kube-controller-manager. Você pode desabilitar os loops do controlador, definindo o sinalizador `--cloud-provider` como `external` ao iniciar o kube-controller-manager.

### cloud-controller-manager

O Docker Community Edition (CE), é ideal para desenvolvedores e pequenas equipes que querem começar a utilizar o Docker e experimentar aplicativos baseados em contêiner. O Docker CE possui três tipos de canais de atualização, **stable**, **test** e **nightly**:

- ✓ **Stable**: Fornece os últimos lançamentos para disponibilidade geral;
- ✓ **Test**: Fornece pré-versões que estão prontas para teste, antes da disponibilidade geral;
- ✓ **Nightly**: Oferece as mais recentes versões do trabalho em andamento, para o próximo grande lançamento.

# Componentes do Node

Anotações

---

---

---

---

---

---

---

---

---

---

---

---

## kubelet

## kube-proxy

## Container Runtime

- ✓ Um agente que é executado em cada nó no cluster. Isso garante que os contêineres estejam sendo executados em um pod;
- ✓ O kubelet usa um conjunto de PodSpecs fornecidos por vários mecanismos e garante que os contêineres descritos nesses PodSpecs, estejam em execução e sejam saudáveis. O kubelet não gerencia contêineres que não foram criados pelo Kubernetes.

## Anotações

---

---

---

---

---

---

---

---

---

---

---

---

## Componentes do Cluster Kubernetes

kubelet

kube-proxy

Container Runtime

- ✓ O kube-proxy é um proxy de rede que é executado em cada nó no cluster. Ele permite a abstração de serviço do Kubernetes, mantendo as regras de rede no host e realizando o encaminhamento de conexão;
- ✓ O kube-proxy é responsável pelo encaminhamento de solicitações. O kube-proxy, permite o encaminhamento de fluxo TCP e UDP ou o encaminhamento TCP e UDP de round robin em um conjunto de funções de back-end.

### Anotações

---

---

---

---

---

---

---

---

---

---

---

---

kubelet

kube-proxy

Container Runtime

- ✓ O tempo de execução do contêiner é o software responsável pela execução de contêineres;
- ✓ O Kubernetes suporta vários tempos de execução do contêiner: Docker, containerd, cri-o, rktlet e qualquer implementação do Kubernetes CRI (Container Runtime Interface).

Anotações

---

---

---

---

---

---

---

---

---

---

---

---



## Componentes do Cluster Kubernetes

### Outros Componentes (addons)



Os addons são pods e serviços que implementam recursos de cluster. Os pods podem ser gerenciados por Implantações, ReplicationControllers e assim por diante. Objetos addon namespaced são criados no namespace kube-system. Segue alguns addons para o cluster Kubernetes:

- ✓ DNS;
- ✓ Web UI (Dashboard);
- ✓ Monitoramento de Recurso de Contêiner;
- ✓ Log de nível de cluster.

### Addons

- **DNS:** Embora os outros complementos não sejam estritamente necessários, todos os clusters do Kubernetes devem ter o DNS do cluster, pois muitos exemplos dependem dele;
- **Web UI (Dashboard):** O Dashboard é uma IU de uso geral, baseada na Web para clusters do Kubernetes. Ele permite que os usuários gerenciem e solucionem problemas de aplicativos em execução no cluster, bem como o próprio cluster;
- **Monitoramento de Recurso de Contêiner:** O Monitoramento de Recurso de Contêiner registra métricas genéricas de séries temporais sobre contêineres em um banco de dados central e fornece uma interface do usuário para navegação desses dados;
- **Log de nível de cluster:** Um mecanismo de criação de log em nível de cluster é responsável por salvar os logs de contêiner em um armazenamento de log central com interface de pesquisa / navegação.

Recapitulando

-  Conhecer os componentes do Master;
-  Conhecer os componentes dos Nodes.

Anotações

---

---

---

---

---

---

---

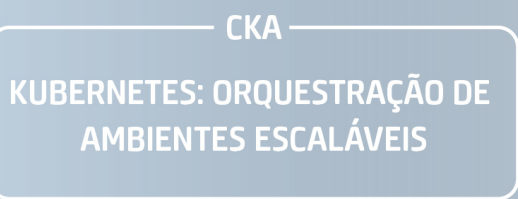
---

---

---

---

---

[illegible]

## Objetivos da Aula

Pod, Service, Volume, Namespace, ReplicaSet, Deployment, StatefulSet e Job.

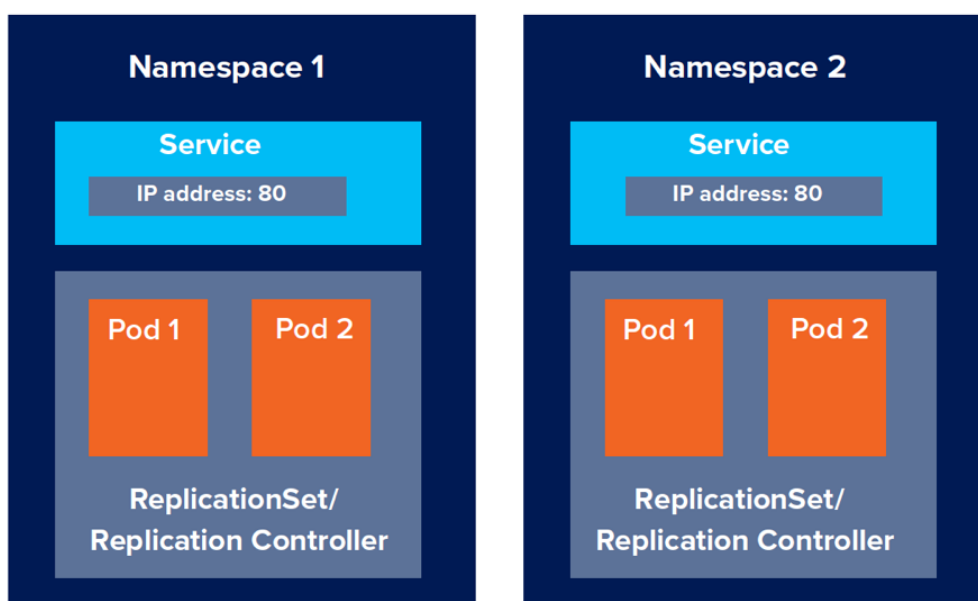
## Competências da Aula

Esta aula possui conteúdo que atende aos objetivos da seguinte certificação:

### Certified Kubernetes Administrator (CKA)

Core Concepts (19% of exam);





## Sobre o Kubernetes

O Kubernetes contém várias abstrações que representam o estado de seu sistema: aplicativos e cargas de trabalho em contêiner implantados, seus recursos de rede, discos associados e outras informações sobre o que seu cluster está fazendo.

## Objetos do Cluster Kubernetes

**Pod**

Service

Volume

Namespace

ReplicaSet

Deployment

StatefulSet

Job

✓ Um Pod é uma unidade de implementação no Kubernetes, que fornece um conjunto de abstrações e serviços para aplicativos em execução no cluster do Kubernetes. Ele pode incluir um ou vários contêineres (por exemplo, Docker e Rkt) que compartilham recursos de armazenamento e de rede. cgroups e namespaces do Linux, são programados / colocados em conjunto e compartilham o mesmo ciclo de vida.

### Uso de Pods

O uso mais básico dos pods são os pods que executam um único contêiner, onde um Pod serve como wrapper em torno de um único contêiner (por exemplo, contêiner Docker) e o Kubernetes gerencia os serviços do Pod para contêineres, em vez de contêineres diretamente.

O uso avançado de pods, inclui pods executando vários contêineres fortemente acoplados. Nesse cenário, um Pod é um wrapper em torno de vários contêineres co-localizados que compartilham recursos e têm responsabilidades distintas. Por exemplo, pode-se imaginar um Pod encapsulando dois contêineres, um dos quais age como um servidor estático para arquivos e o segundo serve como um contêiner de "sidecar", executando operações com esses arquivos (por exemplo, atualização e transformação).

## Objetos do Cluster Kubernetes

Pod

Service

Volume

Namespace

ReplicaSet

Deployment

StatefulSet

Job

- ✓ Uma maneira abstrata de expor um aplicativo em execução em um conjunto de pods, como um serviço de rede;
- ✓ Não há necessidade de modificar seu aplicativo para usar um mecanismo de descoberta de serviço não familiar. O Kubernetes fornece aos pods seus próprios endereços IP e um único nome DNS para um conjunto de pods e pode balancear a carga entre eles.

### Kubernetes Services

Os Kubernetes Pods são mortais. Nascer e quando morrem não são ressuscitados. Se você usa uma implantação para executar seu aplicativo, ele pode criar e destruir os pods dinamicamente (por exemplo, ao dimensionar ou reduzir).

Cada Pod recebe seu próprio endereço IP, no entanto, o conjunto de pods para uma implantação em execução num momento, pode ser diferente do conjunto de pods que executam esse aplicativo no momento seguinte.

Pod

Service

Volume

Namespace

ReplicaSet

Deployment

StatefulSet

Job

- ✓ Um volume do Kubernetes, tem uma vida útil explícita - o mesmo que o Pod que o contém. Consequentemente, um volume ultrapassa todos os Containers que são executados no Pod e os dados são preservados nas reinicializações do Container. Naturalmente, quando um Pod deixa de existir, o volume deixará de existir também.

### Anotações

---

---

---

---

---

---

---

---

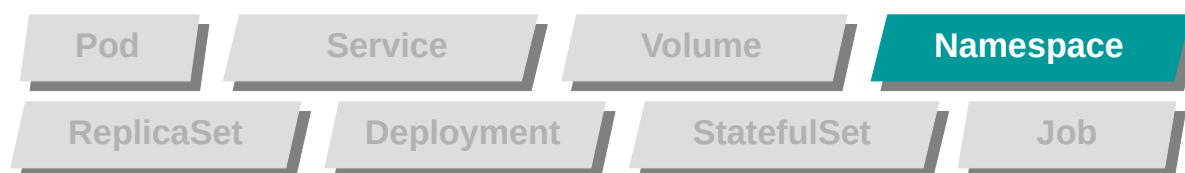
---

---

---

---





- ✓ Namespaces são um conceito muito básico no Kubernetes. Eles são simples de criar, excluir e são usados para subdividir seu cluster para que várias equipes possam trabalhar nele. Isso não apenas economiza custos de servidor, mas também pode aumentar a qualidade, fornecendo uma plataforma conveniente para testes de integração e outros testes de fumaça, antes da implantação na produção.

### Uso de vários namespaces

Os namespaces são destinados ao uso em ambientes com muitos usuários espalhados por várias equipes ou projetos. Para clusters com poucos à dezenas de usuários, você não precisa criar ou pensar em namespaces. Comece a usar namespaces quando precisar dos recursos que eles fornecem.

Namespaces fornecem um escopo para nomes. Os nomes dos recursos precisam ser exclusivos em um namespace, mas não nos namespaces. Os namespaces não podem ser aninhados dentro um do outro e cada recurso do Kubernetes pode estar apenas em um namespace.

Os namespaces são uma maneira de dividir recursos de cluster entre vários usuários (por meio de cota de recursos).

## Objetos do Cluster Kubernetes

Pod

Service

Volume

Namespace

**ReplicaSet**

Deployment

StatefulSet

Job

✓ A finalidade de ReplicaSet é manter um conjunto estável de pods de réplica em execução a qualquer momento. Como tal, é frequentemente utilizado para garantir a disponibilidade de um número especificado de Pods idênticos.

### Como funciona um ReplicaSet?

Um ReplicaSet é definido com campos, incluindo um seletor que especifica como identificar os pods que ele pode adquirir, várias réplicas que indicam quantos Pods devem ser mantidos e um modelo de pod que especifica os dados dos novos Pods que ele deve criar para atender ao número de réplicas de critérios. Um ReplicaSet, em seguida, cumpre o seu objetivo, criando e excluindo pods conforme necessário para alcançar o número desejado. Quando um ReplicaSet precisa criar novos Pods, ele usa seu modelo Pod.

## Objetos do Cluster Kubernetes



✓ Os deployments representam um conjunto de vários Pods idênticos, sem identidades exclusivas. Elas executam várias réplicas do aplicativo e substituem automaticamente todas as instâncias que falham ou não respondem. Assim, os deployments ajudam a garantir que uma ou mais instâncias do aplicativo estejam disponíveis para veicular solicitações dos usuários.

### Anotações

---

---

---

---

---

---

---

---

---

---

---

---

## Objetos do Cluster Kubernetes

Pod

Service

Volume

Namespace

ReplicaSet

Deployment

StatefulSet

Job

✓ Como um Deployment, um StatefulSet gerencia os Pods que são baseados em uma especificação de contêiner idêntica. Ao contrário de uma implantação, um StatefulSet mantém uma identidade fixa para cada um dos seus pods. Esses pods são criados a partir da mesma especificação, mas não são intercambiáveis: cada um tem um identificador persistente que é mantido em qualquer reprogramação.

### Anotações

---

---

---

---

---

---

---

---

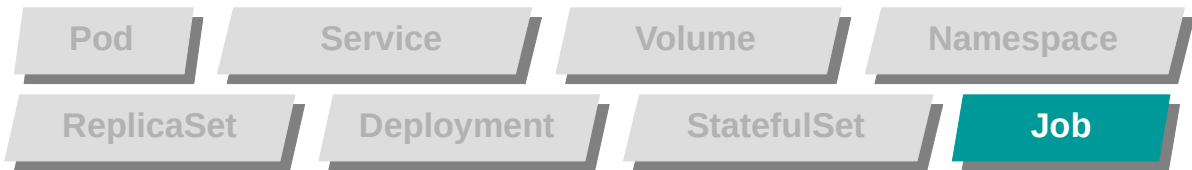
---

---

---

---

## Objetos do Cluster Kubernetes



✓ Um Job cria um ou mais pods e garante que um número especificado deles seja finalizado com sucesso. Quando os pods são concluídos com sucesso, o Job rastreia as conclusões bem-sucedidas. Quando um número especificado de conclusões bem-sucedidas é atingido, a tarefa (isto é, Job) é concluída. Excluir um trabalho limpará os pods criados.

### Anotações

---

---

---

---

---

---

---

---


---

---

---

---

## Recapitulando

 Pod, Service, Volume, Namespace, ReplicaSet, Deployment, StatefulSet e Job.

### Anotações

---

---

---

---

---

---

---

---

---

---

---

---