Kubernetes

Facef - 2020

Diego Osse Fernandes

- Ciência da Computação Unip(2009);
- TechLead LuizaLabs;
- github.com/diegofernandes;
- diego.osse@gmail.com

Kubernetes Parte 1

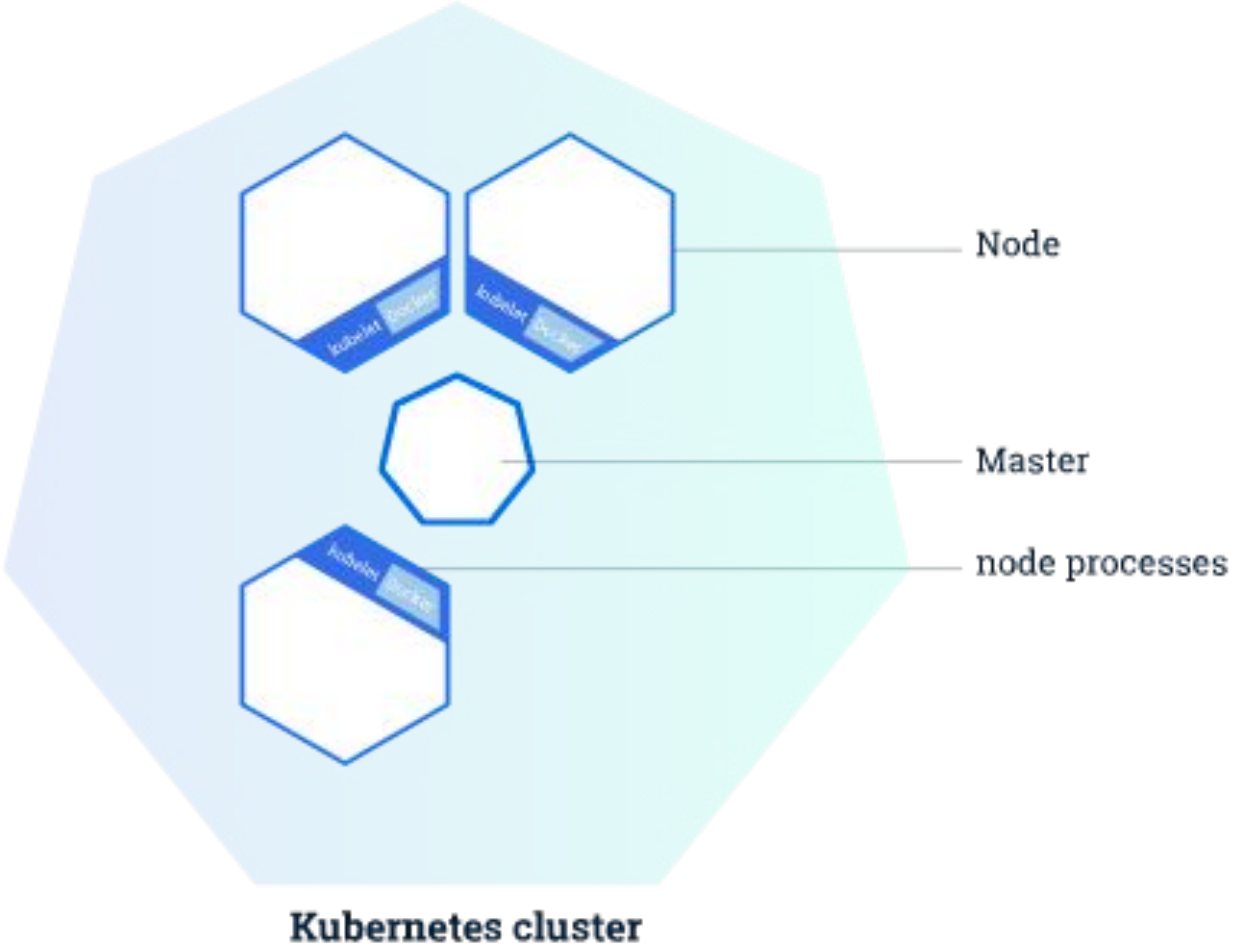
1. Criar um Cluster;
2. Implantar uma aplicação;
3. Explorar a aplicação e seus recursos;
4. Expor a Aplicação;
5. *Scalar* a aplicação;
6. Atualizar a aplicação.

Material adicional https://github.com/diegofernandes/k8s-facef

1 - Criar um Cluster

- microk8s: https://microk8s.io/docs
  - Leve, Zero OPS, Instalação simples
  - Configurar o kubectl https://microk8s.io/docs/working-with-kubectl
- Minikube: https://kubernetes.io/docs/tasks/tools/install-minikube/
  - Desenvolvido pela comunidade do kubernetes
- kubeclt: https://kubernetes.io/docs/tasks/tools/install-kubectl/
  - Utilitário de comando para interação com as API do Kubernetes

O Cluster

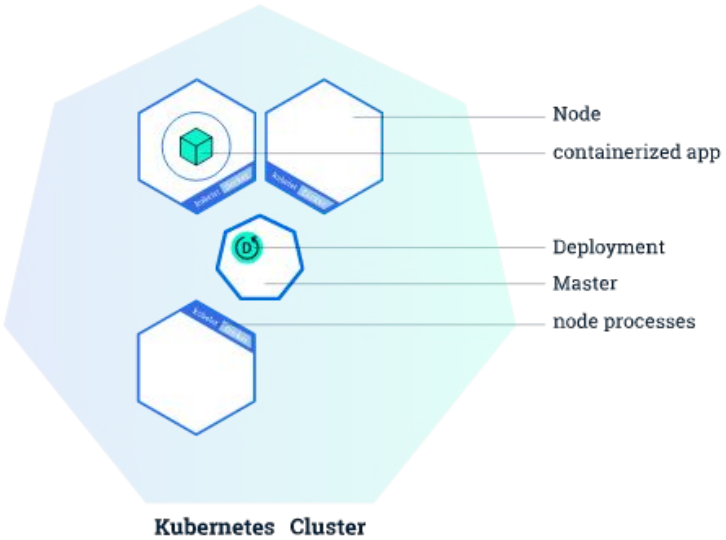


2 - Implantar uma aplicação

1. Vamos criar um deployment:
    a. `kubectl create deployment hello-node --image=k8s.gcr.io/echoserver:1.4`
2. Visualizar o deployment:
    a. `kubectl get deployments`
3. Visualizar os pods:
    a. `kubectl get pods`
4. Apagar o deployment:
    a. `kubectl delete deployment hello-node`

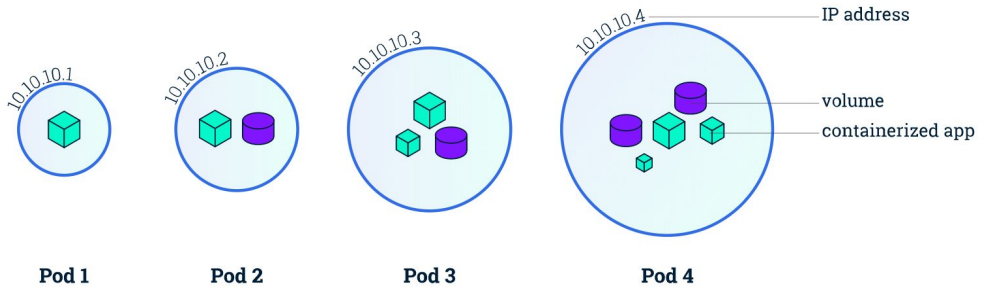
2 - Implantar uma aplicação

- Deployment - Instrui o Kubernetes como:
  - Criar;
  - Atualizar;
  - Iniciar;
  - Parar;
  - Cuidar - Self Healing.



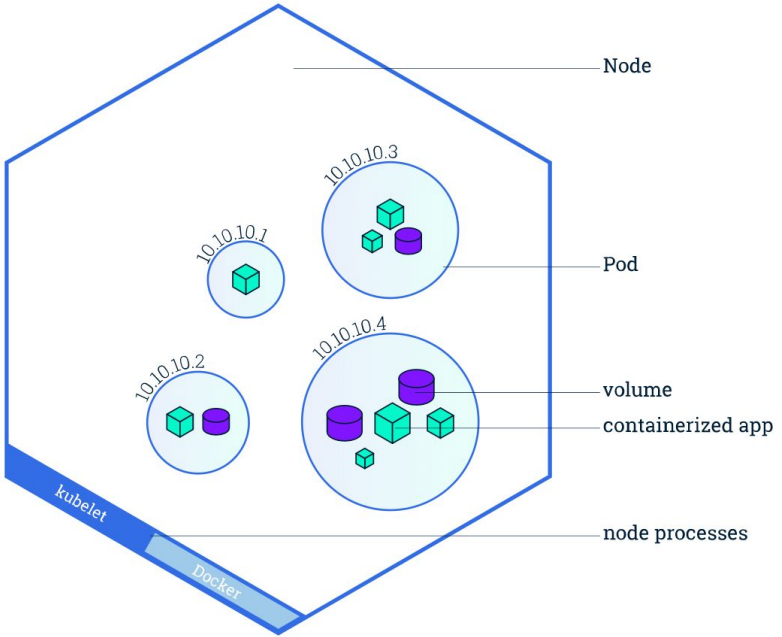
3 - Explorando a Aplicação

- Pods
  - Grupo de containers(Docker, rkt);
  - Compartilha o armazenamento como volume;
  - Compartilhar rede (Logical Host);
  - Pods são atômicos (Únicos);
  - Amarrado a um Node;
  - Mortal, descartável.



3 - Explorando a Aplicação

- Nodes
  - VM ou Máquina física;
  - Nó de trabalho - Onde os Pods são schedulados;
  - Kubelet - Processo responsável por se comunicar com o Master e manter as coisas em ordem;
  - Container runtime(Docker RKT, etc).



3 - Explorar a Aplicação

- Comandos para analisar sua aplicação ao nível de Pods:
  - **kubectl get pods** - lista os recursos do tipo pod;
  - **kubectl describe** - exibe informações detalhadas sobre os recursos;
  - **kubectl logs** - Exibe os logs de um container pertencente a um pod;
  - **kubectl exec** - Executa um comando dentro de um container pertencente a um pod;
  - **kubectl port-forward** - Cria um proxy entre sua máquina e o container pertencente a um pod.

3 - Explorando a Aplicação

Exercício 1:

Usando os comandos anteriores vamos analisar o estado de nossa aplicação hello-api:

1. Implante a aplicação hello-api (`diegofernandes/k8s-facef:0.0.1`);
2. Liste seus pods;
3. Analise seus logs;
4. Descubra os containers de um pod;
5. Crie um proxy para a nossa api que está rodando no cluster;
6. Faça chamadas para nossa api (curl, postman).

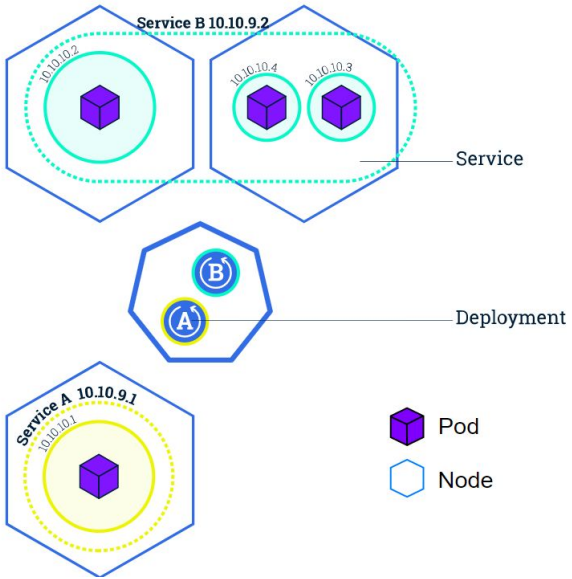
4 - Expondo uma aplicação

Criando um Service:

- **kubectl expose deployment hello-api --type NodePort --port 8080**
- kubectl get nodes -o wide

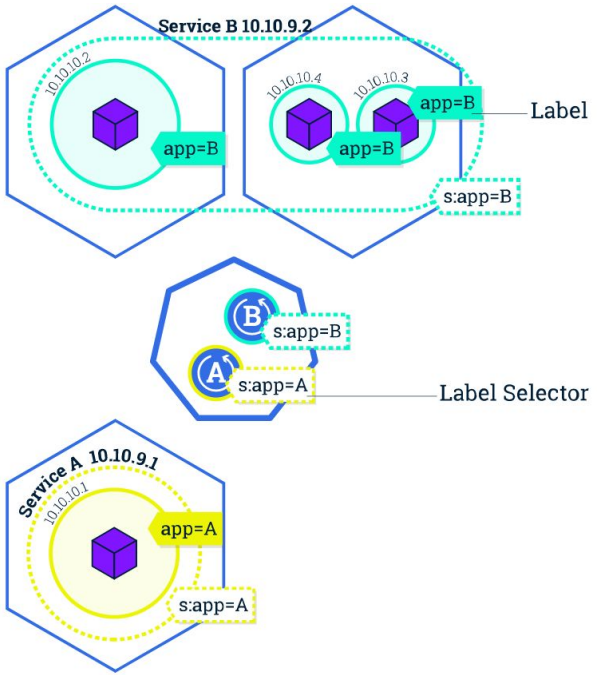
4 - Expondo uma aplicação

- Service:
  - Abstração lógica de Pods e política de acesso a eles;
  - Acompanha as mudanças dos apps;
  - Baixo acoplamento entre apps;
  - Receber tráfego externo;
  - ClusterIP: IP Interno no Cluster(Tráfego somente interno);
  - NodePort: Abre uma porta em cada Node do Cluster;
  - LoadBalance: Cria um LoadBalance Externo na Cloud;
  - ExternalName: Expõem a aplicação com um nome de DNS(Depende de Addon de DNS).



4 - Expondo uma aplicação

- Service e Labels:
  - Service descobre Pods através das labels;
  - Labels são chave-valor;
  - Versionamento;
  - Ambiente de execução;
  - Classificação.



5 - *Scalar* a Aplicação

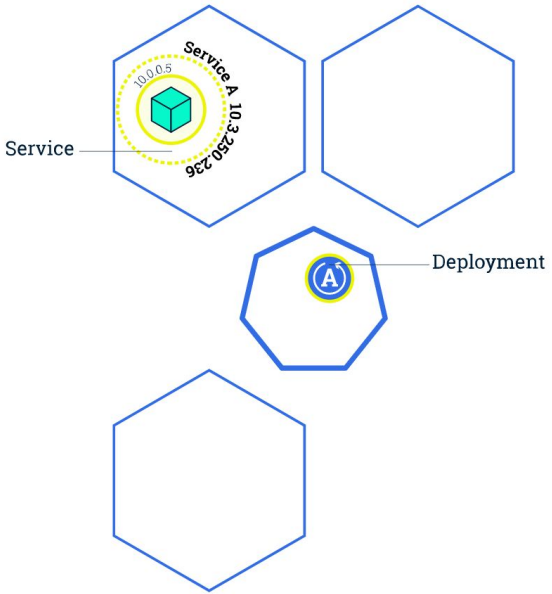
Modificando o número de réplicas de um Deployment:

- **kubectl scale --replicas=3 deployment hello-api**

- **kubectl create deployment --replicas=3**

5 - *Scalar* a Aplicação

- *Scale:*
  - Aplicação acompanha a demanda;
  - Services distribuem o tráfego entre os pods;
  - Alta disponibilidade - Agora podemos modificar sem gerar downtimes;
  - Autoscaling - adicionar ou remover pods automaticamente.



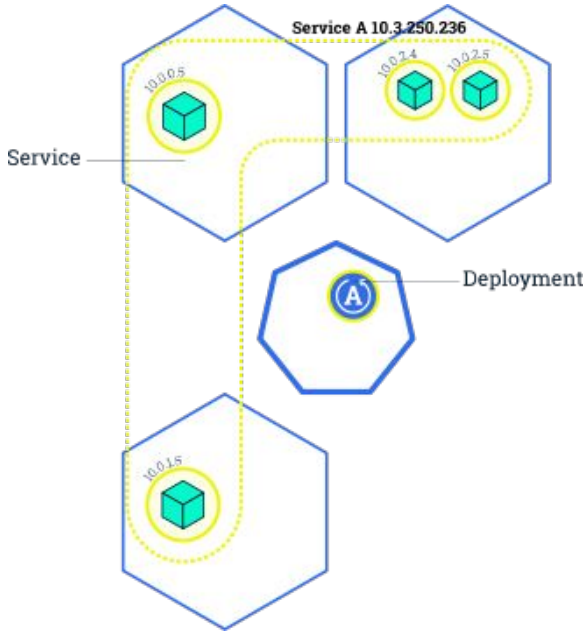
6 - Atualizando a Aplicação

Rolling Update:

- **kubectl set image deployment hello-api hello-api=diegofernandes/hello-api:0.0.2**
- **kubectl rollout status deployment hello-api**
- **kubectl rollout undo deployment hello-api**

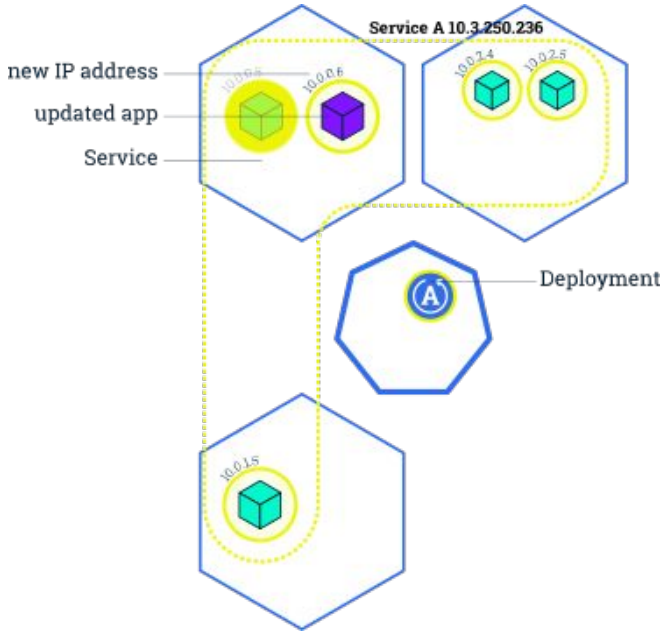
6 - Atualizando a Aplicação

- *Rolling Update*
  - Zero downtime;
  - Substituição gradual;
  - Rollback;
  - Promoção entre ambientes;
  - Integração com CD.



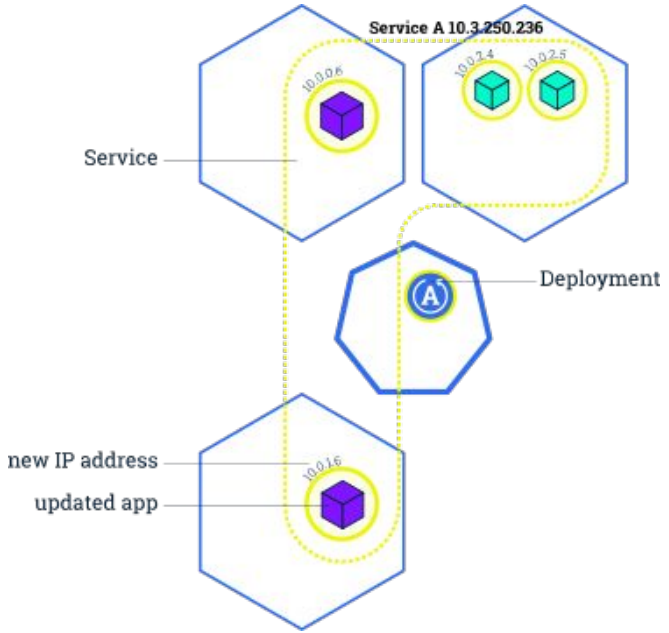
6 - Atualizando a Aplicação

- *Rolling Update*
  - Zero downtime;
  - Substituição gradual;
  - Rollback;
  - Promoção entre ambientes;
  - Integração com CD.



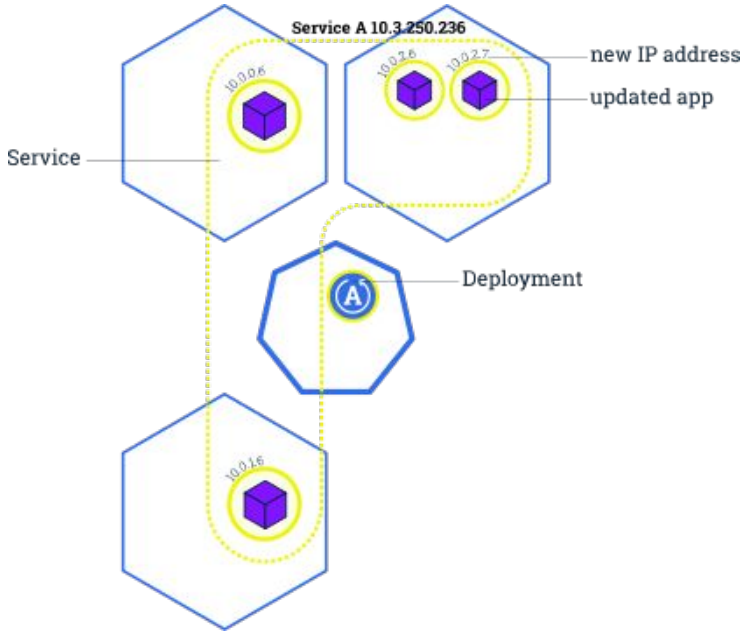
6 - Atualizando a Aplicação

- *Rolling Update*
  - Zero downtime;
  - Substituição gradual;
  - Rollback;
  - Promoção entre ambientes;
  - Integração com CD.



6 - Atualizando a Aplicação

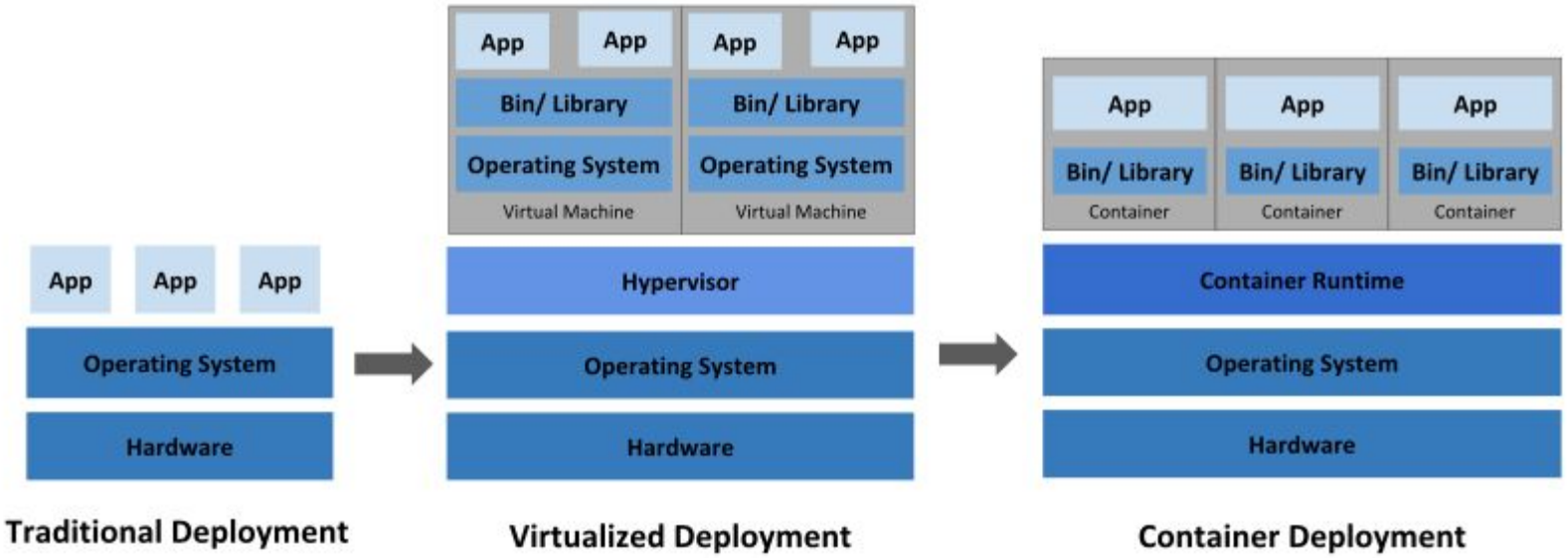
- *Rolling Update:*
  - Zero downtime;
  - Substituição gradual;
  - Rollback;
  - Promoção entre ambientes;
  - Integração com CD.



O que é Kubernetes?

- Portável;
- Extensível;
- OpenSource;
- Gerenciador de serviços e aplicações conteinerizadas;
- Kubernetes vem do Grego, que significa piloto.

Evolução dos ambientes



Visão Geral

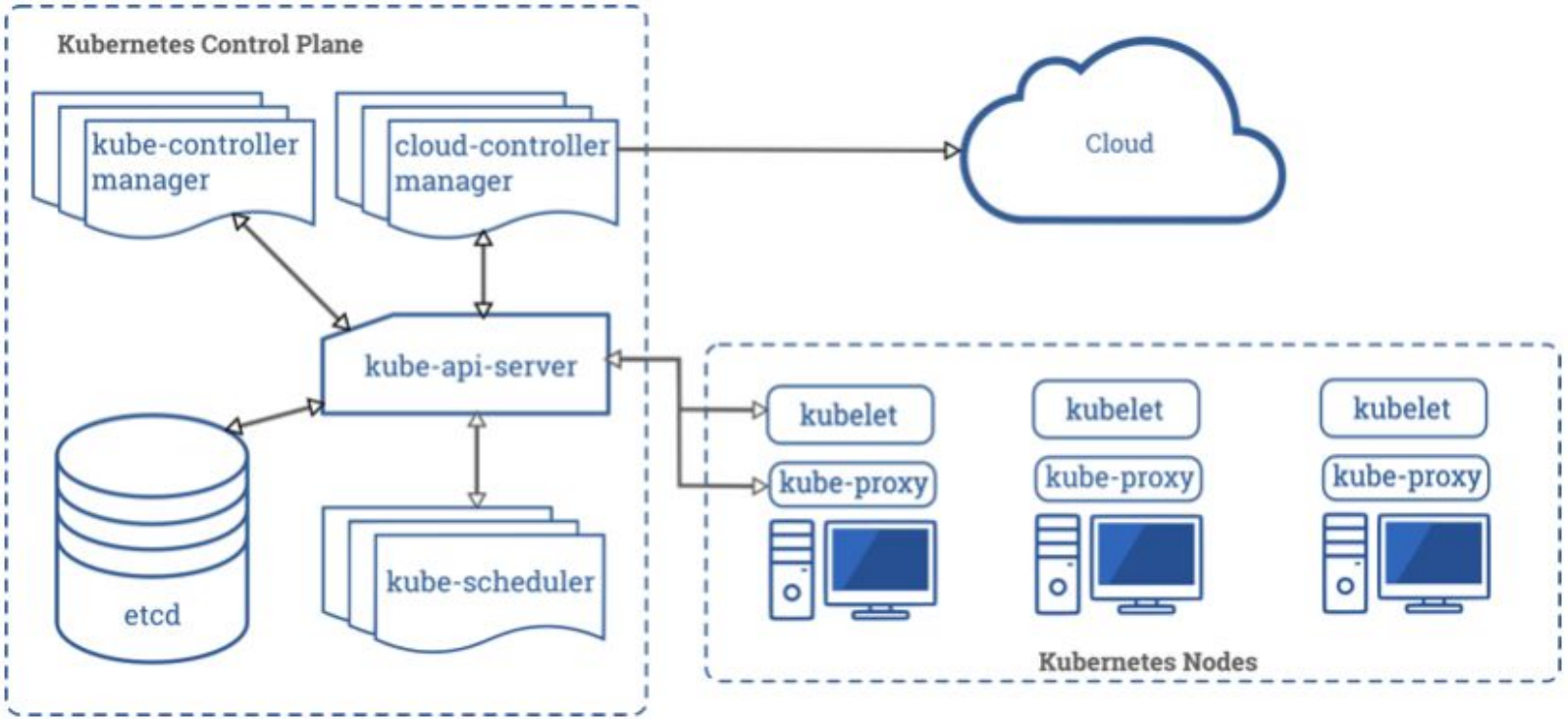
O que posso fazer com o Kubernetes?

- Service discovery e Load balancing;
- Orquestração de armazenamento;
- Rollouts e rollback automatizados;
- Bin packing automáticos(Organiza para caber);
- Self-healing;
- Gerenciamento de configurações e segredos.

O que o Kubernetes não é/faz?

- Não vai implantar ou compilar sua aplicação;
- Não fornece serviços de nível de aplicação(Mensageria, Banco de dados, etc);
- Não fornece nem adota nenhum sistema abrangente de configuração, manutenção, gerenciamento ou autocorreção da máquina;
- Não dita qual solução usar para logging, monitoria e alertas.

Componentes


Kubernetes Control Plane
kube-controller manager
cloud-controller manager
Cloud
kube-api-server
kubelet
kube-proxy
kubelet
kube-proxy
kubelet
kube-proxy
etcd
kube-scheduler
Kubernetes Nodes

Componentes *Control Plane*

- Componentes de tomada de decisões globais sobre o cluster:
  - Ex Quando iniciar um novo POD;
- kube-apiserver - Componente que expoem as interfaces de controle(Kubernetes API);
- etcd - Banco chave/valor de alta disponibilidade e consistência;
- kube-scheduler:
  - Observa pods recém criados sem nós atribuídos e seleciona um nó respeitando seus requisitos(Recursos, Afinidade, políticas, etc);
- kube-controller-manager - Processos de controle:
  - Node Controller: Monitora/notifica quando nós ficam on e off;
  - Replication Controller: Responsável por manter o número correto de pods para cada replication controller;
  - Endpoint Controller: Mantém objetos de Endpoints(O que junta services e pods;)
  - Service Account e Token Controllers: Cria os accounts e tokens de acessos para os novos Namespaces.

Componentes Control Plane

- cloud-controller-manager:
  - Opcional;
  - Faz a liga entre o seu cluster e os serviços de Cloud;
  - Específicos para cada cloud(GCP, AWS, Azure, etc);
  - Controle de Nodes - Adicionar ou remover nós no cluster;
  - Controle de Rotas - gerenciar as rotas na camada de rede da cloud;
  - Controle de Service - Criar, atualizar, deletar load balance na cloud;
  - Etc.

Nós, Nodos, Nodes

- kubelet - garante que todos os containers em um pod estejam saudáveis;
- kube-proxy - Proxy de rede, faz parte do conceito do Service;
- Container runtime:
    - Docker;
    - containerd;
    - CRI-O;
    - Outra implementações .

Addons

- DNS:
  - CoreDNS
  - Kube-DNS
- WebUI -Painéis;
- Monitoramento:
  - Metrics-server
  - Prometheus
- Log:
  - cluster-level-logging
  - logging agent
    - Stackdriver
    - fluentd
    - ELK

Exercício: Deploy da sua aplicação

1. Conteinizar a aplicação de api node feita nas aulas de React;
2. Publicar o container no docker hub(http://hub.docker.com);
3. Implantar o container no cluster;
4. Expor a porta da aplicação.

Próxima Aula

- API Kubernetes (yaml files);
- Configurações e segredos;
- Health check;
- Alocação de recursos;
- Auto *scaling* horizontal.