Administração de Infraestrutura Web com Kubernetes e Google Cloud

Entregando serviços web altamente disponíveis, escaláveis e automatizados.

Márcio Jales

Site Reliabilty Engineer, AppProva

AppProva

SOMOS EDUCAÇÃO

Motivação

Motivação

- Uma nova cultura de desenvolvimento de software: **DevOps**
  - Mais integração entre desenvolvimento e operações;
  - Entregas contínuas, rápido *feedback*;
  - *The three ways*.
- Como era antes:
  - Aplicações monolíticas;
  - Meses para consolidar novas funcionalidades;
  - Testes e implantação (*deploy*) apenas ao final;
- Como é com DevOps:
  - Aplicações fracamente acopladas (microserviços);
  - Pequenos fragmentos de códigos de cada vez;
  - Testes constantes e automáticos até produção.

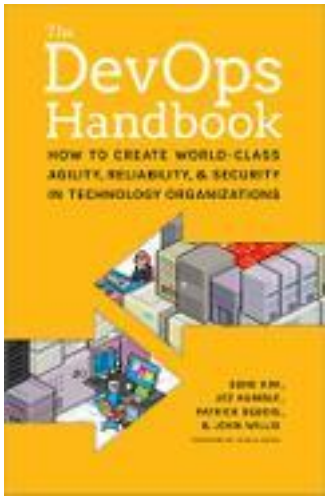
Motivação

- Desafios comuns de uma infraestrutura complexa:
  - Automação;
  - Alta disponibilidade;
  - Tolerância a falhas;
  - Escalabilidade.

- DevOps na infraestrutura:
  - *Infrasctructure as Code*;
  - *Containers*;
  - Computação na Nuvem.

Referências

Referências



The DevOps Handbook

John Willis, Patrick Debois, Jez Humble, Gene Kim

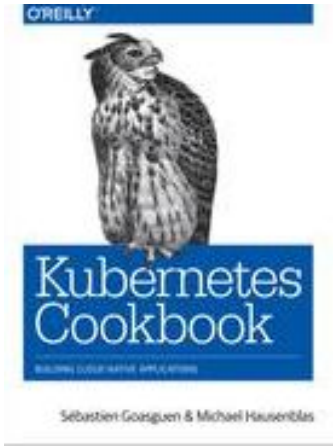
Outubro 2016

Site Reliability Engineering

Niall R. Murphy, Jennifer Petofl, Chris Jones, Betsy Beyer

Abril 2016

Referências



Kubernetes Cookbook

Sébastien Goasguen; Michael Hausenblas

Março 2018

Kubernetes: Up and Running

Joe Beda, Brendan Burns, Kelsey Hightower

Setembro 2017

Arquitetura e Definições

Arquitetura

- Um **mestre** comanda um série de **nós**. A aplicação é executada apenas nos nós;

- Kubernetes suporta apenas orquestramento de containers **Docker** e **rkt**;

- A intereção do usuário é feita através de um cliente chamado **kubectl**.

- As entidades do Kubernetes são os **objetos**. Cada um corresponde a um **recurso** no servidor da API, que fica no mestre.

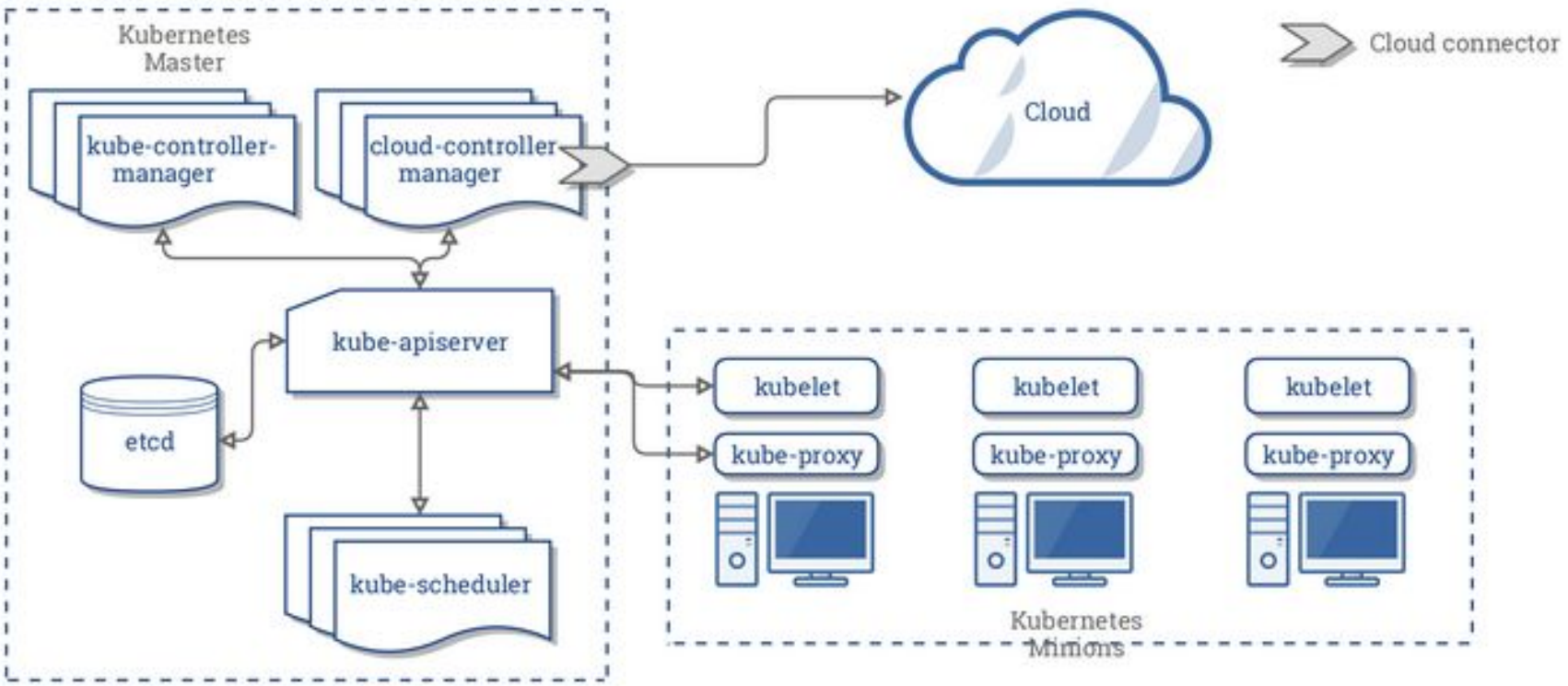
Arquitetura

- Componentes do cluster no mestre:
  - kube-apiserver:
    - Comanda a criação de todos os objetos
  - etcd:
    - Serviço que armazena as informações do cluster;
    - Recebe as informações do kuber-apiserver;
  - kube-scheduler:
    - Decide em quais nós cada Pod executará;
  - kube-controller-manager:
    - Controladores que administram criação de certos objetos;
    - Consultam kuber-apiserver para saber se tais objetos foram criados e agir.

Arquitetura

- Componentes do cluster nos nós:
  - Kube proxy:
    - Deve estar presente em todos os nós;
    - Responsável pelo roteamento do tráfego
    - Objeto do Kubernetes;
  - Kube DNS:
    - Pode ou não estar presente em todos os nós;
    - Objeto do Kubernetes;
  - Kubelet:
    - Garante que os Pods permaneçam executando;
    - Não é um objeto do Kubernetes;
    - Administra o funcionamento do *runtime* dos containers;

Arquitetura



Fonte: Concepts Underlying the Cloud Controller Manager. Disponível em <https://kubernetes.io/docs/concepts/architecture/cloud-controller/>

Administração dos objetos

- Comandos imperativos:
  - Age diretamente sobre os objetos.
- Configuração imperativa:
  - Especifica uma ação sobre um objeto descrito em um **manifesto**;
- Configuração declarativa:
  - Dado um manifesto, kubernetes identifica qual ação executar.

Management technique	Operates on	Recommended environment	Supported writers	Learning curve
Imperative commands	Live objects	Development projects	1+	Lowest
Imperative object configuration	Individual files	Production projects	1	Moderate
Declarative object configuration	Directories of files	Production projects	1+	Highest

Fonte: Kubernetes Object Management. Disponível em <https://kubernetes.io/docs/concepts/overview/object-management-kubectl/overview/>

Declaração de objetos

- Arquivos com extensão “ymal”;

- Definem as especificações do objeto;

- 5 componentes principais:
  - apiVersion;
  - kind;
  - metadata (preenchido tanto pelo usuário quanto servidor);
  - spec (preenchido pelo usuário);
  - status (preenchido pelo servidor).

Google Cloud Platform

GCP

- Conjunto de recursos físicos e virtuais que residem em *datacenters* do Google;
- *Datacenters* se localizam em **regiões**, as quais podem ser divididas por **zonas**;
- Os acessos aos recursos são feitos através de **serviços**;
- Cada serviço requer uma **ativação de API.**
- A interação com o GCP é feita por um cliente chamado **gcloud.**

GKE

- Serviço do Google Cloud que administra e dispõe clusters com Kubernetes configurado;
- O nó mestre fica sob responsabilidade do GCP;
- Inclui outros componentes no cluster, além do proxy e DNS:
  - Fluentd (para logs);
  - Heapster/metrics-server (monitoramento);
- Utiliza-se de ferramentas avançadas:
  - Balanceamento de carga;
  - *Node pools*;
  - Dimensionamento horizontal e atualização automáticos;
  - Autorreparação de nós;
  - Monitoramento integrado;

Clusters

- Conjunto de máquinas que trabalham em conjunto;
  - Clusters de máquinas x Cluster no Kubernetes;

- $ gcloud container clusters ...

- $ gcloud container node-pools …

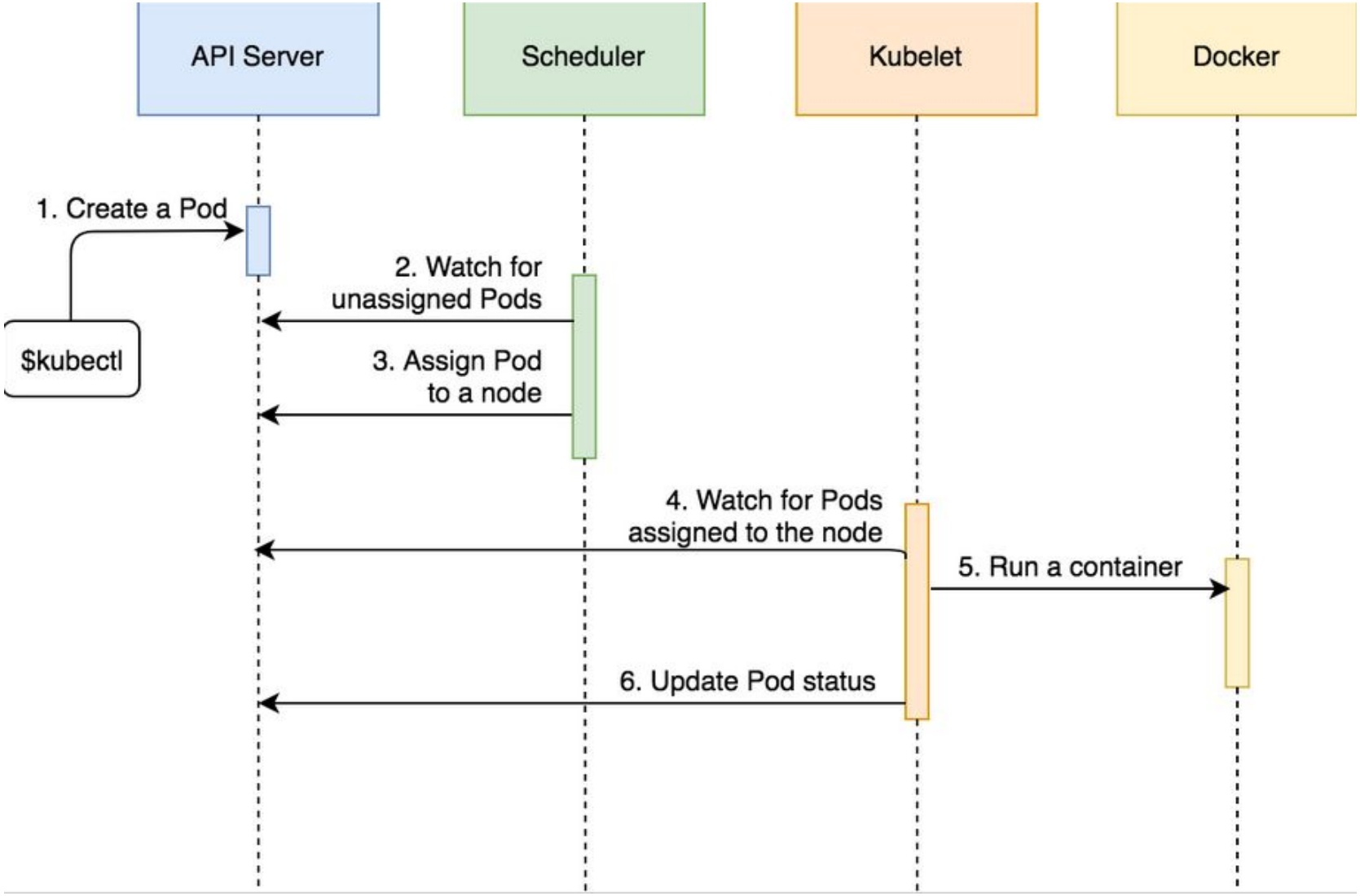
- $ gcloud compute firewall-rules ...

Workloads

Pods

- ➔ Conjunto de containers que trabalham em cooperação;
- ➔ A unidade fundamental do Kubernetes;
- ➔ Compartilham o mesmo espaço de rede (IP, portas) e volumes;
- ➔ O que deve ser colocado junto em um Pod?

Pods



Fonte: The DevOps 2.3 Toolkit: Kubernetes Workshop. Viktor Farcic. Disponível em <http://vfarcic.github.io/devops23/workshop.html#/11/6>

Pods

- Executar um Pod:
  - $ kubectl run kuard --image=gcr.io/kuar-demo/kuard-amd64:1
  - $ kubectl apply -f https://raw.githubusercontent.com/kubernetes-up-and-running/examples/master/5-1-kuard-pod.yaml

- Obter informações:
  - $ kubectl get pods [-o wide]
  - $ kubectl describe pod kuard

Pods

- Apagar Pod:
  - $ kubectl delete pod kuard
- Acessar Pod:
  - $ kubectl port-forward kuard 8080:8080
- Acessar logs do Pod:
  - $ kubectl logs kuard
- Executar comando dentro do Pod:
  - $ kubectl exec kuard printenv

Pods

- *Health checks*:
  - *liveness probe* x *readiness probe*;
  - $ kubectl apply -f https://raw.githubusercontent.com/kubernetes-up-and-running/examples/master/5-2-kuard-pod-health.yaml

- Controle de recursos:
  - requerer x limitar;
  - kubectl apply -f https://raw.githubusercontent.com/kubernetes-up-and-running/examples/master/5-3-kaurd-pod-resreq.yaml
  - kubectl apply -f https://raw.githubusercontent.com/kubernetes-up-and-running/examples/master/5-4-kaurd-pod-reslim.yaml

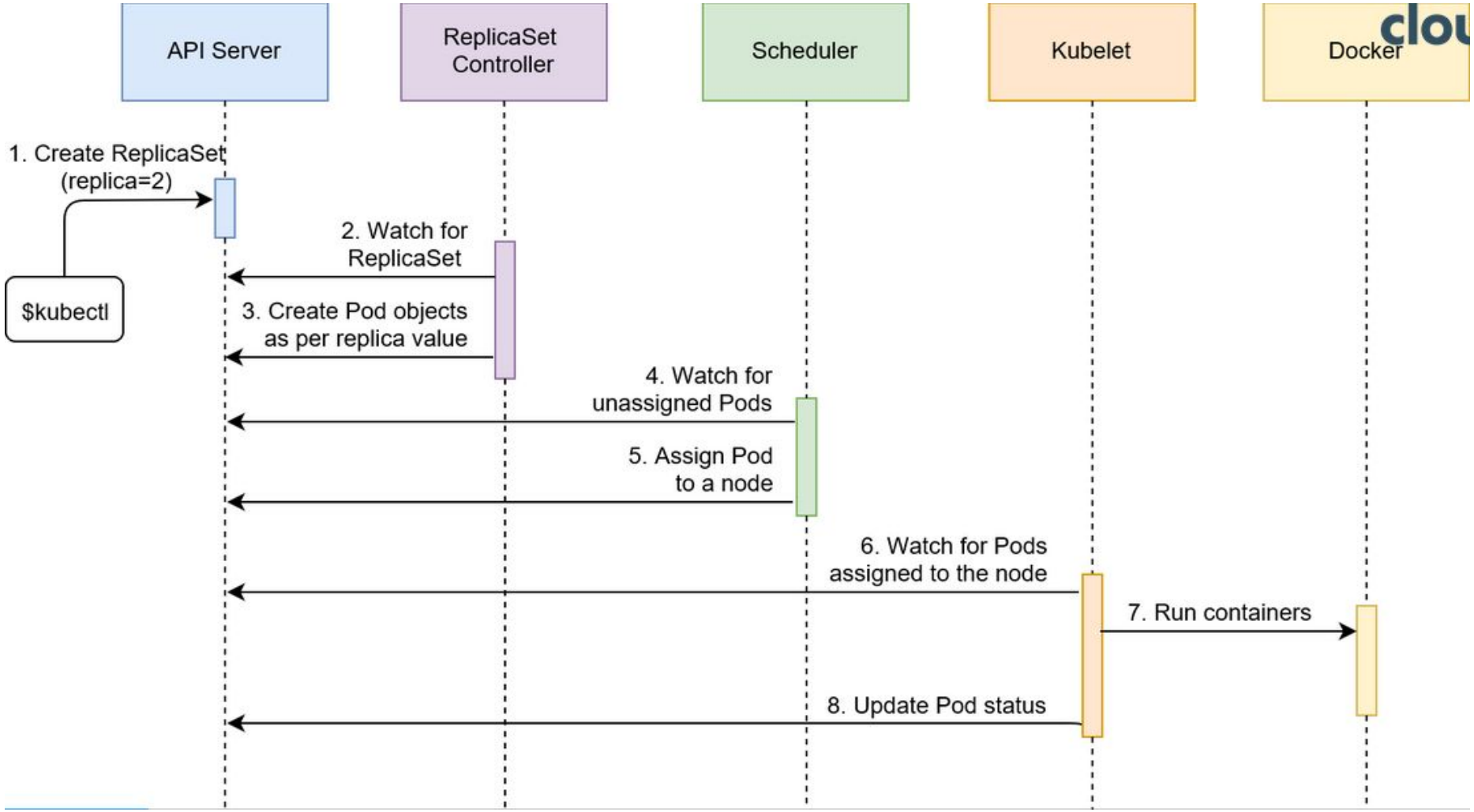
Pods

- Volumes:
  - kubectl apply -f https://raw.githubusercontent.com/kubernetes-up-and-running/examples/master/5-5-kuard-pod-vol.yaml
  - “spec.volumes” + “spec.containers.volumeMounts”.

ReplicaSets

- Criar Pods isolados não é um antipadrão e sem muita utilidade;
- **ReplicaSets** controla um grupo de Pods;
- Pods sob um ReplicaSet são realocados automaticamente em caso de falhas com o cluster;
- Especificação de um Pod + quantidade de réplicas desejadas.

ReplicaSets



Fonte: The DevOps 2.3 Toolkit: Kubernetes Workshop. Viktor Farcic. Disponível em <http://vfarcic.github.io/devops23/workshop.html#/13/4>

ReplicaSets

- Utilizam ***labels*** (rótulos) para identificar quais Pods devem administrar;
- Pods e ReplicaSets são fracamente acoplados;
- Pensado para aplicações *stateless*;
- kubectl apply -f kuard-rs.yaml

ReplicaSets

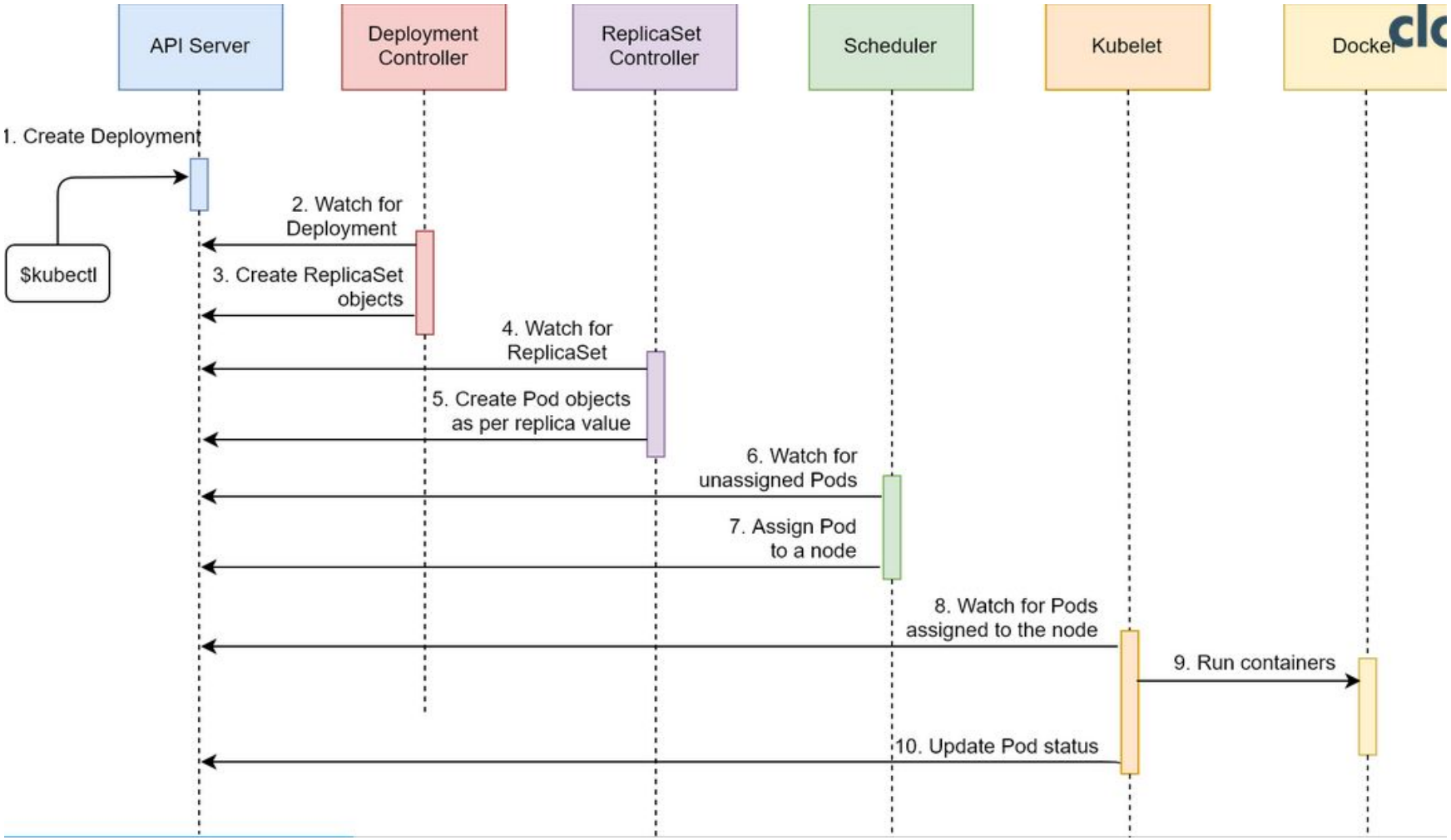
- Dimensionamento de ReplicaSets:
  - kubectl scale rs kuard --replicas=4;
  - kubectl edit rs kuard (mudar “spec.replicas”);
  - Automático com ***HorizontalPodAutoscaler*** (HPA)

- HPA
  - kubectl apply -f kuard-hpa.yaml;
  - Na versão v1 da API 1.10, apenas por uso de CPU.

Deployments

- ➔ ReplicaSets estão presos a apenas uma imagem de container;
- ➔ **Deployments** são os objetos que administram novas versões da imagem, sem *downtime*;
- ➔ Deployment cria um ReplicaSet que, por suz vez, controla os Pods.

Deployments



Fonte: The DevOps 2.3 Toolkit: Kubernetes Workshop. Viktor Farcic. Disponível em <http://vfarcic.github.io/devops23/workshop.html#/17/4>

Deployments

- “spec.strategy” define como a atualização de imagem é feita:
  - “RollingUpdate” x “Recreate”;

- “spec.minReadySeconds” especifica quanto o deployment deve esperar para atualizar o próximo pod;

- $ kubectl apply -f kuard-deploy.yaml;

- $ kubectl rollout history deployment kuard;

- $ kubectl rollout undo deployments kuard --to-revision=1.

Mais tipos de workloads

- **Jobs** criam Pods que executam até suas tarefas serem completadas;

- **DaemonSets** criam Pods que devem ser executados em todos os nós no cluster;

Service Discovery

Services

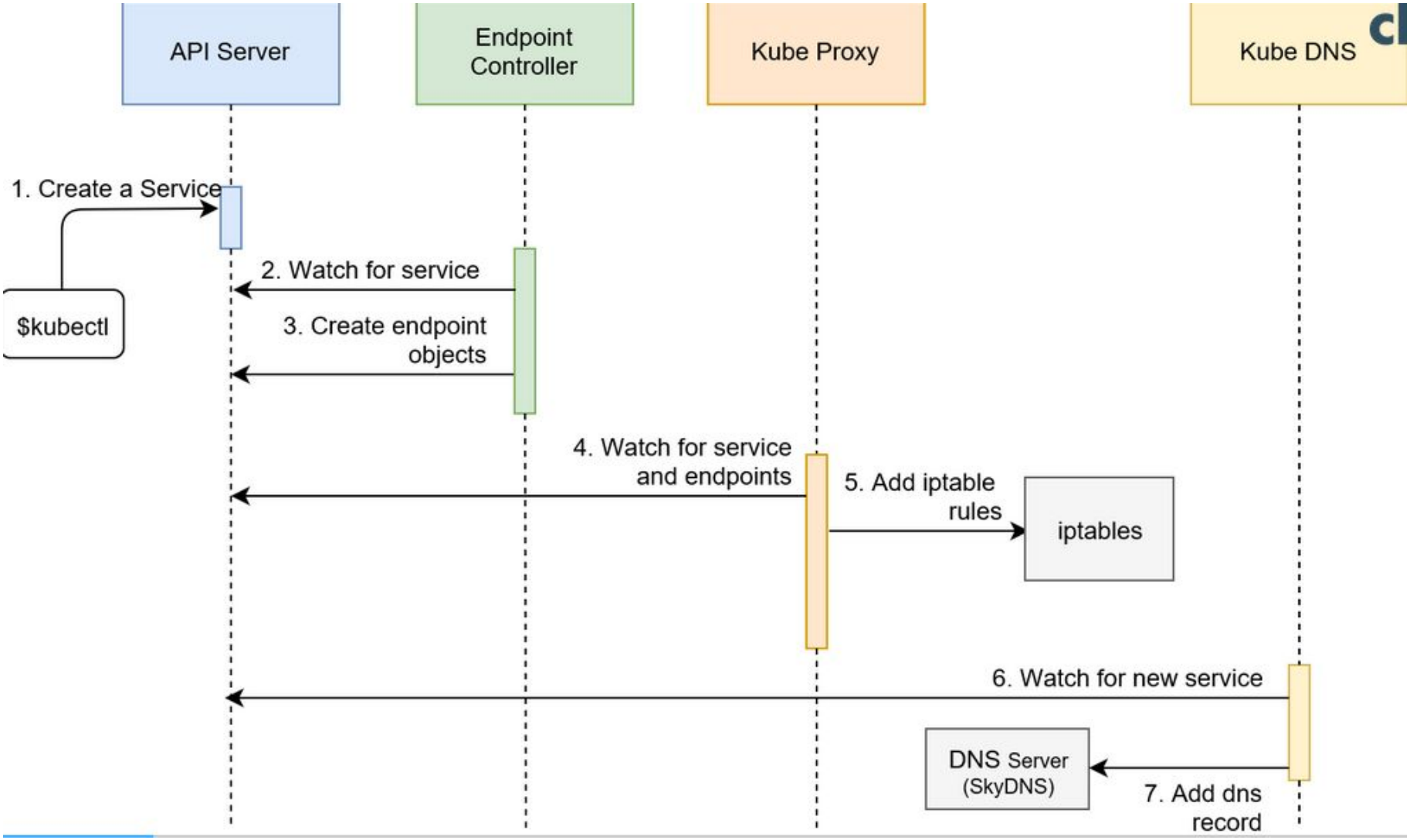
- Em um cenário com várias réplicas, é preciso saber quais pods correspondem a uma certa funcionalidade e onde estão;

- ***Services*** são os objetos que permitem isso;

- Dependem do DNS do Kubernetes;
  - Serviço *kube-dns*;

- Criam ***endpoints***, objetos que identificam os Pods que recebem tráfego.

Services



Fonte: The DevOps 2.3 Toolkit: Kubernetes Workshop. Viktor Farcic. Disponível em <http://vfarcic.github.io/devops23/workshop.html#/15/3>

Services

- ClusterIP:
  - Serviço acessível apenas dentro do cluster;
  - Um IP virtual que identifica os Pods que recebem tráfego.

- NodePort:
  - Abre um porta na máquina que redireciona tráfego externo para o serviço;
  - Constituído sobre um ClusterIP.

- LoadBalancer:
  - Necessita suporte da Cloud;
  - Constituído sobre um NodePort;
  - Cria um IP único que redireciona tráfego para o serviço.

Configurações

Configurações

- Os dois objetos principais são **ConfigMaps** e **Secrets**;

- ConfigMaps:
  - Informações não sensíveis;
  - Podem ser incorporados no Pod de dois modos:
    - Volumes;
    - Variáveis de ambiente.

- Secrets:
  - Semelhante aos configMaps, porém para informações sensíveis;
  - Valores com codificação base64.

Configurações

- kubectl apply -f https://raw.githubusercontent.com/kubernetes-up-and-running/examples/master/11-2-kuard-config.yaml

- kubectl apply -f https://raw.githubusercontent.com/kubernetes-up-and-running/examples/master/11-3-kuard-secret.yaml

- Para secrets, às vezes é mais fácil criá-los imperativamente com “kubectl create secret”.