Habilitando Aplicações Nativas de Nuvem

Introdução a Contêineres e Kubernetes

3. Kubernetes

Sergio Rio
www.sergiorio.tech

Programa: Introdução a Contêineres e Kubernetes

1. Conceitos Básicos

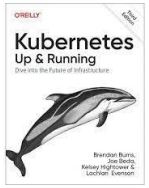
- ✓ Abstrações em Ciência da Computação
- ✓ Virtualização de Computadores
- ✓ MicroVMs e Unikernels

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.

Refrescando nossa memória

- Nas aula passada estudamos os princípios fundamentais do K8s e suas implicações.
- Atividades preparatórias para a aula de hoje:
  - Ler capítulos 3, 4 e 5 do livro “Kubernetes Up and Running”
    - Deploying a Kubernetes Cluster
    - Common *kubectl* Commands
    - Pods
  - Laboratório tutorial Kubernetes *minikube:*
    1. Instalar e configurar *minikube* em instancia EC2 da *AWS:*
       - Setup Minikube on AWS EC2 Ubuntu
    2. Implementar um aplicativo na instancia *minikube*:
       - Hello Minikube
    3. Terminar e remover instancia EC2 ao concluir tutorial!

Abstrações de Recursos

Manifestos

- Os documentos de manifestos do K8s servem de base para descrição e definição de recursos, que podemos criar e editar posteriormente em formato YAML.
- Os manifestos representam a especificação “do estado desejado” de um objeto, além de informações básicas sobre o objeto (como seu nome).
- Em essência, existem quatro campos essenciais nos manifestos do K8s que devem estar sempre presentes:
  - *apiVersion:*
  - *kind:*
  - *metadata:*
  - *spec:*

apiVersion

- Define qual versão da API um determinado recurso usará. Pode ser simplesmente v1, o que significa que fará parte da API principal especificada em /api/v1.
- Também pode ser <nome>/<versão>, por exemplo batch/v1, especificando que o recurso usa uma API que está em /api/<nome>.
- Podemos descobrir mais sobre quais APIs e versões existem em um determinado cluster executando

```
$ kubectl api-version
```

- Os resultados serão diferentes de cluster para cluster e entre versões do K8s. Podemos ter APIs personalizadas, APIs desabilitadas ou APIs recentes que podem ter sido implementadas em diferentes versões do K8s.
- *ReplicationController* e *ReplicaSet* são dois objetos populares que esclarecem isso e diferem na apiVersion
  - *ReplicaController* é um componente da API principal na v1, então escreveríamos apiVersion: v1 em suas especificações.
  - *ReplicaSet*, que evoluiu do ReplicaController, é um componente de uma API mais recente servida em apps/v1. Esse tipo de controle de versão é incrivelmente poderoso e flexível, permitindo que o K8s evolua enquanto mantém muita compatibilidade com versões anteriores.

kind

- Representa o tipo de objeto especificado por meio de um manifesto.
- Cada tipo de recurso estará disponível em uma API específica. Isso torna essencial que *kind* e *apiVersion* em um manifesto específico sejam coerentes.
- Podemos inspecionar que tipo podemos usar para objetos executando:

```
$ kubectl api-resources
```

- Assim é possível ver rapidamente quais tipos de recursos existe(m) em cada apiVersion.
- Exemplo:

```
kind: Pod
```

metadata

- Descreve informações de um objeto que permitem a identificação exclusiva desse objeto.
- Ao criar um manifesto, este campo deverá ter pelo menos um nome associado. Normalmente também veremos um campo chamado *labels*.
- Exemplo:

```
metadata:
  name: kuard
```

spec

- Define o estado desejado do objeto no K8s.
- Varia amplamente entre diferentes recursos e versões de API (difícil de memorizar todos os campos necessários).
- Exemplo

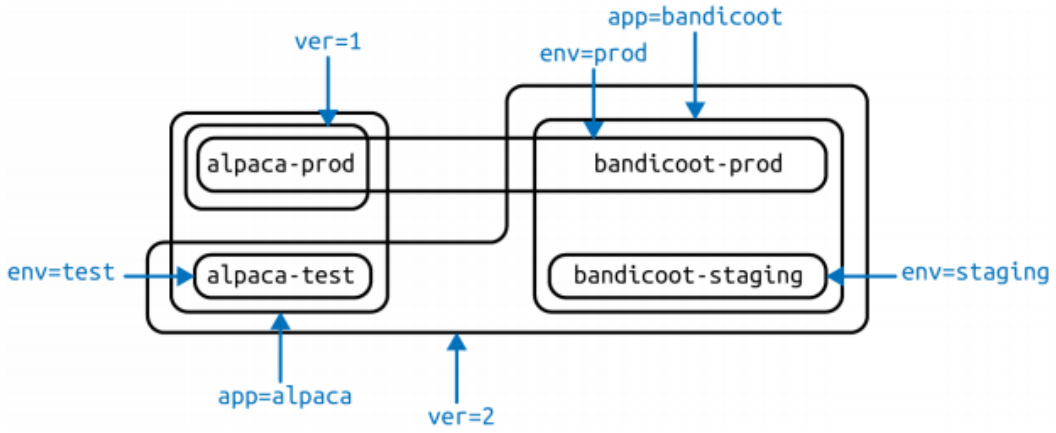
```
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 4 # tells deployment to run 4 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16.1
        ports:
        - containerPort: 80
```

Labels

- São pares chave/valor que podem ser anexados a objetos K8s, como *pods* e *ReplicaSets* por exemplo.
- Eles podem ser arbitrários e são úteis para anexar informações de identificação e fornecem a base para agrupar objetos.
- E permitem que os usuários mapeiem as estruturas de suas próprias organizações nos objetos do sistema K8s para facilitar a sua administração.

```
$ kubectl get deployments --show-labels

NAME                ... LABELS
alpaca-prod         ... app=alpaca,env=prod,ver=1
alpaca-test         ... app=alpaca,env=test,ver=2
bandicoot-prod      ... app=bandicoot,env=prod,ver=2
bandicoot-staging   ... app=bandicoot,env=staging,ver=2
```



Label Selectors 1/2

- Os seletores de rótulos são usados para filtrar objetos do K8s com base em um conjunto de rótulos.
- Usam uma sintaxe simples para expressões booleanas e são usados tanto por usuários finais (via ferramentas como *kubectl*) como também por diferentes tipos de objetos (como a forma como um *ReplicaSet* se relaciona aos seus *pods*).
- Cada implantação (por meio de um *ReplicaSet*) cria um conjunto de pods usando os rótulos especificados.

```
$ kubectl get pods --show-labels

NAME                                ... LABELS
alpaca-prod-3408831585-4nzfb        ... app=alpaca,env=prod,ver=1,...
alpaca-prod-3408831585-kga0a        ... app=alpaca,env=prod,ver=1,...
alpaca-test-1004512375-3r1m5        ... app=alpaca,env=test,ver=2,...
bandicoot-prod-373860099-0t1gp      ... app=bandicoot,env=prod,ver=2,...
bandicoot-prod-373860099-k2wcf      ... app=bandicoot,env=prod,ver=2,...
bandicoot-staging-1839769971-3ndv ... app=bandicoot,env=staging,ver=2,...
```

Label Selectors 2/2

```
$ kubectl get pods --selector="ver=2"

NAME                                READY     STATUS    RESTARTS   AGE
alpaca-test-1004512375-3r1m5        1/1       Running   0          3m
bandicoot-prod-373860099-0t1gp      1/1       Running   0          3m
bandicoot-prod-373860099-k2wcf      1/1       Running   0          3m
bandicoot-staging-1839769971-3ndv5  1/1       Running   0          3m
```

```
$ kubectl get pods --selector="app=bandicoot,ver=2"

NAME                                READY     STATUS    RESTARTS   AGE
bandicoot-prod-373860099-0t1gp      1/1       Running   0          4m
bandicoot-prod-373860099-k2wcf      1/1       Running   0          4m
bandicoot-staging-1839769971-3ndv5  1/1       Running   0          4m
```

Annotations

- Armazenam metadados adicionais de objetos K8s com o único propósito de auxiliar as ferramentas e bibliotecas.
- Podem ser usados para a própria ferramenta ou para passar a informações de configuração entre sistemas externos.
- São uteis por exemplo:
  - Acompanhar o “motivo” da atualização mais recente de um objeto.
  - Anexar informações de compilação, lançamento ou imagem que não sejam apropriadas para rótulos (pode incluir um hash Git, carimbo de data/hora, número PR, etc.).
  - Fornecer dados extras para melhorar a qualidade visual ou a usabilidade de uma UI. Por exemplo, os objetos podem incluir um link para um ícone (ou uma versão codificada em base64 de um ícone).
  - Protótipo de funcionalidade alfa no K8s (em vez de criar uma API de primeira classe campo, os parâmetros para essa funcionalidade são codificados em uma anotação).

Atividades para a próxima aula

- Ler capítulo 7 do livro “Kubernetes Up and Running”
  - Service Discovery