Configurando um Cluster e Node Groups AWS EKS

Carlos Diego C. P. | www.cdiego.com

SUMÁRIO

SUMÁRIO **2**

1. ARQUITETURA DO KUBERNETES **3**

KUBERNETES CONTROL PLANE 4

KUBERNETES NODES 5

PODS E SERVIÇOS 6

SERVIÇOS DO KUBERNETES 7

REDES KUBERNETES 7

ARMAZENAMENTO PERSISTENTE NO KUBERNETES 8

DESCOBRINDO E PUBLICANDO SERVIÇOS NO KUBERNETES 9

NAMESPACES, RÓTULOS E NOTATIONS 10

2. CONFIGURANDO UM CLUSTER AWS EKS **12**

3. CONFIGURANDO NODE GROUPS NO EKS **14**

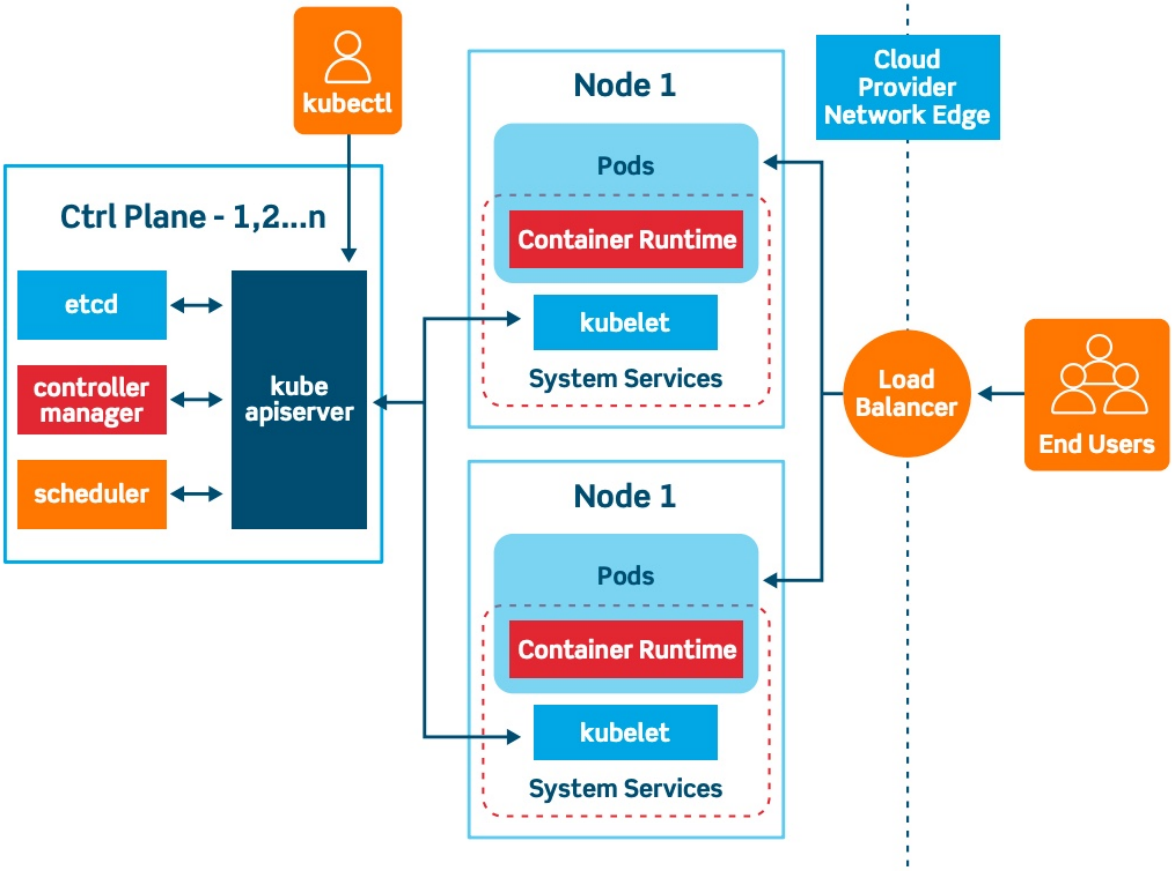
4. DEPLOY NO EKS A PARTIR DE UMA IMAGEM PÚBLICA NO ECR **15**

5. CRIANDO UMA IMAGEM ECR CUSTOMIZADA **17**

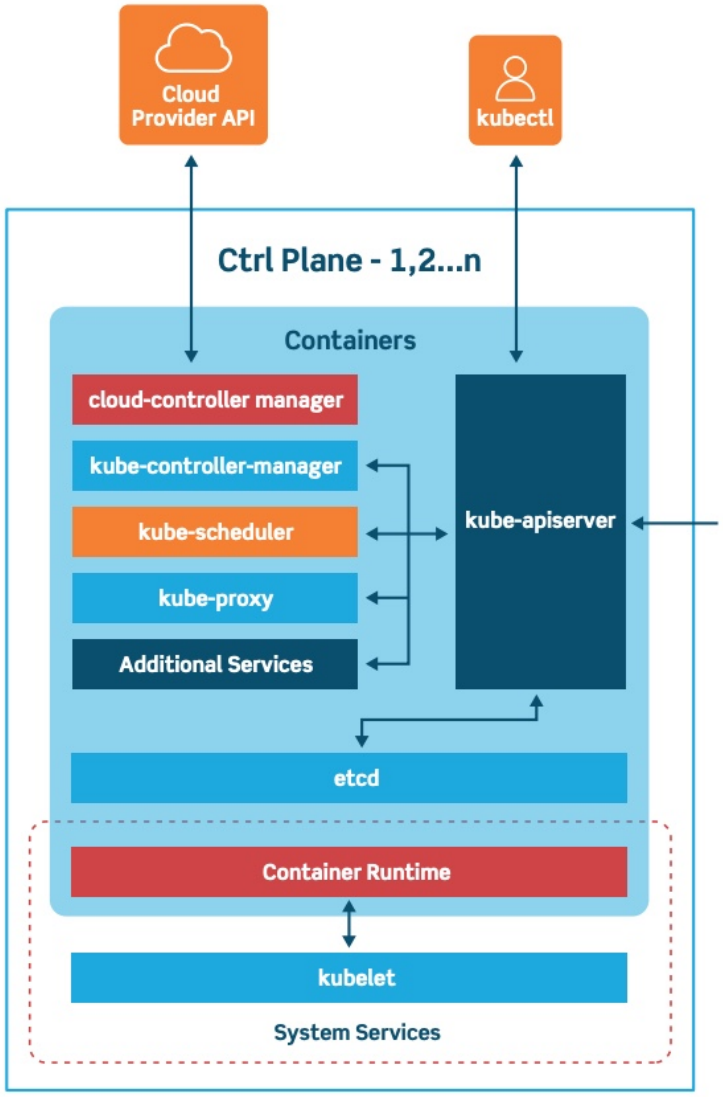
ANEXOS | PRINCIPAIS COMANDOS KUBERNETES **18**

1. ARQUITETURA DO KUBERNETES

MASTER (CLUSTER)
WORKER-NODES
REPLICA-SET
PODS
CONTAINERS



KUBERNETES CONTROL PLANE



O Control Plane é o sistema que mantém um registro de todos os objetos do Kubernetes. Ele gerencia continuamente os estados dos objetos, respondendo às alterações no cluster; Ele também trabalha para fazer com que o estado real dos objetos do sistema corresponda ao estado desejado. Como mostra a ilustração acima, o Control Plane é composto por três componentes principais: kube-apiserver, kube-controller-manager e kube-scheduler. Todos eles podem ser executados em um único nó principal ou podem ser replicados em vários nós principais para alta disponibilidade.

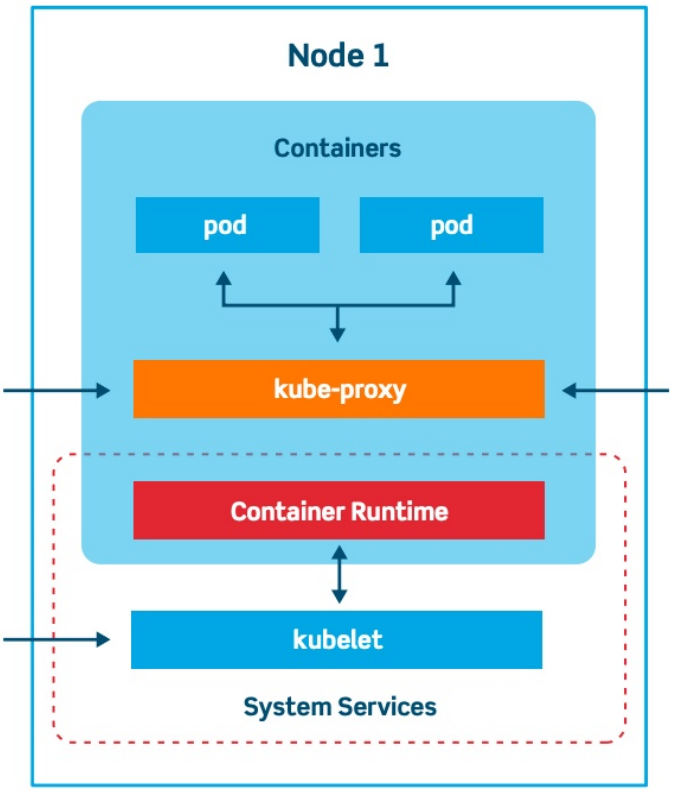
O servidor de API fornece APIs para oferecer suporte à orquestração do ciclo de vida (dimensionamento, atualizações etc.) para diferentes tipos de aplicativos. Ele também atua como o gateway para o cluster, portanto, o servidor da API deve estar acessível por clientes de fora do cluster. Os clientes se autenticam pelo servidor de API e também o usam como proxy / túnel para nós e pods (e serviços).

A maioria dos recursos contém metadados, como rótulos e notations, estado desejado (especificação) e estado observado (status atual). Os controladores trabalham para direcionar o estado real para o estado desejado.

Existem vários controladores para controlar o estado de nós, replicação (escalonamento automático), pontos de extremidade (serviços e pods), contas de serviço e tokens (namespaces). O Controller Manager é um daemon que executa os loops de controle principal, observa o estado do cluster e faz alterações para direcionar o status para o estado desejado. O Cloud Controller Manager integra-se a cada nuvem pública para oferecer suporte ideal a zonas de disponibilidade, instâncias de VM, serviços de armazenamento e serviços de rede para DNS, roteamento e balanceamento de carga.

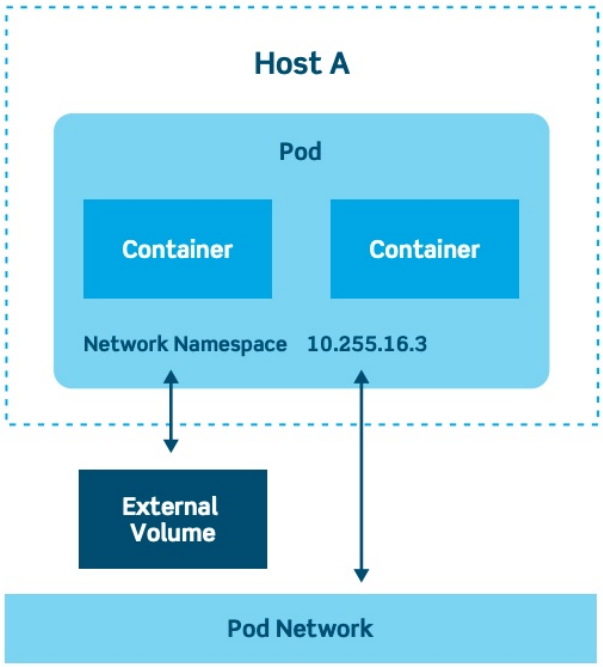
O Scheduler é responsável pelo agendamento de contêineres nos nós do cluster; leva em consideração várias restrições, como limitações ou garantias de recursos e especificações de afinidade e anti-afinidade.

KUBERNETES NODES



Nós de cluster são máquinas que executam contêineres e são gerenciadas pelos nós principais. O Kubelet é o controlador principal e mais importante em Kubernetes. É responsável por direcionar a camada de execução do contêiner, normalmente o Docker.

PODS E SERVIÇOS



O Pod é um dos conceitos cruciais no Kubernetes, pois são a construção principal com a qual os desenvolvedores interagem. Os conceitos anteriores são arquitetura interna e focada em infraestrutura.

Essa construção lógica empacota um único aplicativo , que pode consistir em vários contêineres e volumes de armazenamento. Normalmente, um único contêiner (às vezes com algum programa auxiliar em um contêiner adicional) é executado nessa configuração - conforme mostrado no diagrama abaixo.

Como alternativa, os pods podem ser usados para hospedar pilhas de aplicativos verticalmente integrados, como um aplicativo WordPress LAMP (Linux, Apache, MySQL, PHP). Um pod representa um processo em execução em um cluster .

Os vagens são efêmeros, com uma vida útil limitada. Ao reduzir novamente ou atualizar para uma nova versão, por exemplo, os pods acabam morrendo. Os pods podem fazer o dimensionamento automático horizontal (ou seja, aumentar ou diminuir o número de instâncias) e executar atualizações contínuas e implantações de canários.

Existem vários tipos de pods:

- ReplicaSet , o padrão, é um tipo relativamente simples. Garante que o número especificado de pods esteja em execução
- A implantação é uma maneira declarativa de gerenciar pods via ReplicaSets. Inclui mecanismos de reversão e atualização de rolagem
- Daemonset é uma maneira de garantir que cada nó execute uma instância de um pod. Usado para serviços de cluster, como monitoramento de integridade e encaminhamento de log
- StatefulSet é adaptado para gerenciar pods que devem persistir ou manter o estado

- Job e CronJob executam tarefas de vida curta como pontuais ou programadas.

Essa transitoriedade inerente cria o problema de como rastrear quais pods estão disponíveis e executar um aplicativo específico. É aqui que entram os Serviços .

SERVIÇOS DO KUBERNETES

Os serviços são a maneira de o Kubernetes configurar um proxy para encaminhar o tráfego para um conjunto de pods. Em vez de atribuições estáticas baseadas em endereços IP, os Serviços usam seletores (ou rótulos) para definir quais pods usam qual serviço. Essas atribuições dinâmicas tornam muito fácil o lançamento de novas versões ou a adição de pods a um serviço. Sempre que um Pod com as mesmas etiquetas de um serviço é ativado, ele é atribuído ao serviço.

Por padrão, os serviços são acessíveis apenas dentro do cluster usando o tipo de serviço clusterIP. Outros tipos de serviço permitem acesso externo; o tipo LoadBalancer é o mais comum em implantações em nuvem. Ele ativará um balanceador de carga por serviço no ambiente de nuvem, o que pode ser caro. Com muitos serviços, também pode se tornar muito complexo.

Para resolver essa complexidade e custo, o Kubernetes suporta o Ingress , uma abstração de alto nível que governa como usuários externos acessam serviços em execução em um cluster Kubernetes usando regras de roteamento HTTP baseadas em host ou URL.

Existem muitos controladores de Ingress diferentes (Nginx, Ambassador) e há suporte para balanceadores de carga nativos da nuvem (do Google, Amazon e Microsoft). Os controladores de ingresso permitem expor vários serviços sob o mesmo endereço IP, usando os mesmos balanceadores de carga.

A funcionalidade do Ingress também vai além das regras de roteamento simples. O Ingress permite a configuração de resiliência (tempos limite, limite de taxa), roteamento baseado em conteúdo, autenticação e muito mais.

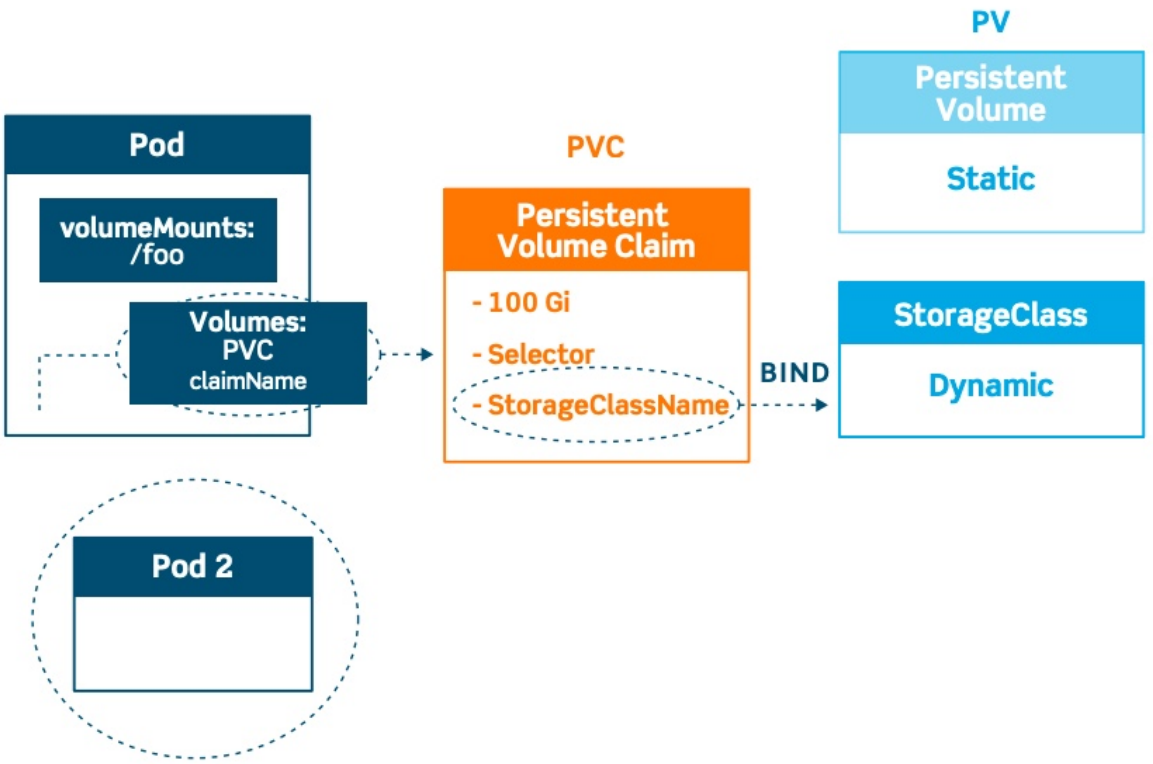
REDES KUBERNETES

Rede O Kubernetes possui um modelo de rede distinto para redes de podto-pod em todo o cluster. Na maioria dos casos, a CNI (Container Network Interface) usa uma rede de sobreposição simples (como Flannel) para ocultar a rede subjacente do pod usando encapsulamento de tráfego (como VXLAN); Ele também pode usar uma solução totalmente roteada como o Calico. Nos dois casos, os pods se comunicam por uma rede de pods em todo o cluster, gerenciada por um provedor CNI como Flannel ou Calico.

Dentro de um pod, os contêineres podem se comunicar sem restrições. Os contêineres em um pod existem no mesmo espaço de nome de rede e compartilham um IP. Isso significa que os contêineres podem se comunicar através do host local. Os pods podem se comunicar usando o endereço IP do pod, acessível em todo o cluster.

Mover de pods para serviços, ou de fontes externas para serviços, requer passar pelo kube-proxy.

ARMAZENAMENTO PERSISTENTE NO KUBERNETES



Kubernetes usa o conceito de volumes . Na sua essência, um volume é apenas um diretório, possivelmente com alguns dados, acessíveis a um pod. Como esse diretório é criado, a mídia que o suporta e seu conteúdo é determinado pelo tipo de volume específico usado.

O Kubernetes possui vários tipos de armazenamento, que podem ser misturados e combinados em um pod (veja a ilustração acima). O armazenamento em um pod pode ser consumido por qualquer contêiner. O armazenamento sobrevive à reinicialização do pod, mas o que acontece após a exclusão do pod depende do tipo de armazenamento específico.

Existem muitas opções para montar arquivos e bloquear o armazenamento em um pod. Os mais comuns são serviços de armazenamento em nuvem pública, como AWS EBS e gcePersistentDisk, ou tipos que se conectam a uma infraestrutura de armazenamento físico, como CephFS, Fibre Channel, iSCSI, NFS, Flocker ou glusterFS.

Existem alguns tipos especiais, como configMap e Secrets, usados para injetar informações armazenadas no Kubernetes no pod ou noDir vazio, comumente usado como espaço temporário.

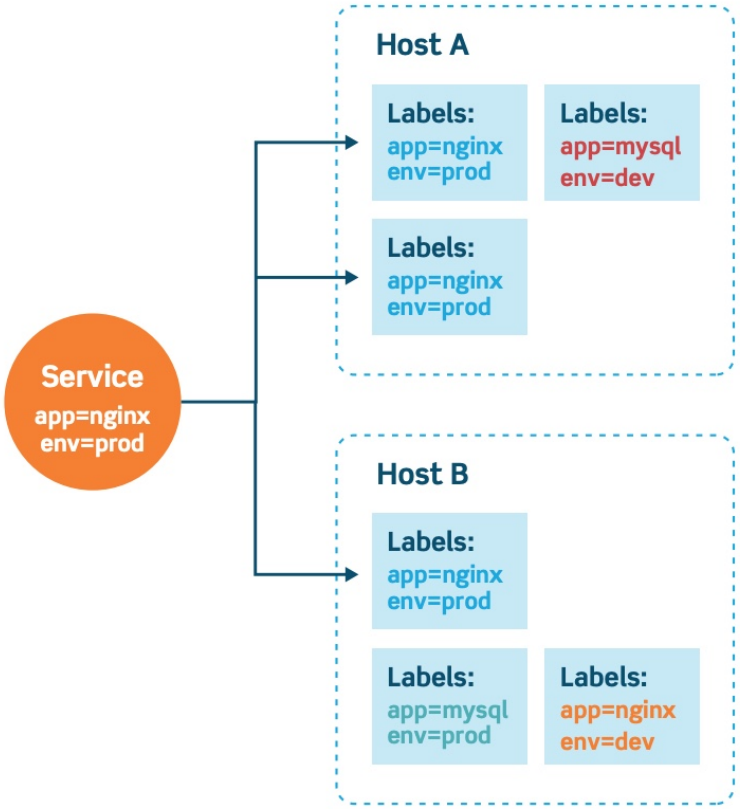
Os PersistentVolumes (PVs) estão vinculados a um recurso de armazenamento existente e geralmente são provisionados por um administrador. São objetos de todo o cluster

vinculados ao provedor de armazenamento de backup que disponibiliza esses recursos para consumo.

Para cada pod, um PersistentVolumeClaim faz uma solicitação de consumo de armazenamento em um namespace. Dependendo do uso atual do PV, ele pode ter diferentes fases ou estados: disponível, vinculado (indisponível para outros), liberado (requer intervenção manual) e com falha (o Kubernetes não pôde recuperar o PV).

Finalmente, StorageClasses são uma camada de abstração para diferenciar a qualidade do armazenamento subjacente. Eles podem ser usados para separar características diferentes, como desempenho. StorageClasses não são diferentes de etiquetas; os operadores os usam para descrever diferentes tipos de armazenamento, para que o armazenamento possa ser provisionado dinamicamente com base nas declarações recebidas dos pods. Eles são usados em conjunto com o PersistentVolumeClaims, que é como os pods solicitam dinamicamente novo armazenamento. Esse tipo de alocação dinâmica de armazenamento é comumente usado quando o armazenamento é um serviço, como em provedores de nuvem pública ou sistemas de armazenamento como o CEPH.

DESCOBRINDO E PUBLICANDO SERVIÇOS NO KUBERNETES



A descoberta de serviços é uma parte crucial de um ambiente saudável do Kubernetes, e o Kubernetes depende muito de seu serviço DNS integrado (Kube-DNS ou CoreDNS , dependendo da versão do cluster) para fazer isso. O Kube-DNS e o CoreDNS criam,

atualizam e excluem registros DNS de serviços e pods associados, conforme mostrado na ilustração acima. Isso permite que os aplicativos segmentem outros serviços ou pods no cluster por meio de um esquema de nomeação simples e consistente.

Um exemplo de registro DNS para um serviço Kubernetes:

`service.namespace.svc.cluster.local`
Um pod teria um registro DNS, como:

`10.32.0.125.namespace.pod.cluster.local`

Existem quatro tipos de serviço diferentes, cada um com comportamentos diferentes:

1. O ClusterIP expõe o serviço apenas em um IP interno. Isso torna o serviço acessível apenas de dentro do cluster. Este é o tipo padrão.
2. NodePort expõe o serviço no IP de cada nó em uma porta específica. Isso dá aos desenvolvedores a liberdade de configurar seus próprios balanceadores de carga, por exemplo, ou configurar ambientes não totalmente suportados pelo Kubernetes.
3. O LoadBalancer expõe o serviço externamente usando o balanceador de carga de um provedor de nuvem. Isso geralmente é usado quando o balanceador de carga do provedor de nuvem é suportado pelo Kubernetes, pois automatiza sua configuração.
4. ExternalName apenas mapeia um registro CNAME no DNS. Nenhum proxy de qualquer tipo é estabelecido. Isso é comumente usado para criar um serviço no Kubernetes para representar um armazenamento de dados externo, como um banco de dados executado externamente no Kubernetes. Um caso de uso potencial seria usar o AWS RDS como banco de dados de produção e um contêiner MySQL para o ambiente de teste.

NAMESPACES, RÓTULOS E NOTATIONS

Os namespaces são clusters virtuais em um cluster físico. Eles têm como objetivo oferecer a várias equipes, usuários e projetos um ambiente virtualmente separado para trabalhar e impedir que as equipes atrapalhem, limitando o que os objetos do Kubernetes podem ver e acessar.

Os rótulos distinguem recursos em um único namespace. Eles são pares de chave / valor que descrevem atributos e podem ser usados para organizar e selecionar subconjuntos de objetos. Os rótulos permitem consultas e relógios eficientes e são ideais para uso em interfaces orientadas ao usuário para mapear estruturas da organização em objetos Kubernetes.

Os rótulos são frequentemente usados para descrever o estado da liberação (estável, canário), ambiente (desenvolvimento, teste, produção), camada de aplicativo (front-end, back-end) ou identificação do cliente. Os seletores usam rótulos para filtrar ou selecionar objetos e são usados em todo o Kubernetes. Isso evita que os objetos sejam vinculados fisicamente.

As notations, por outro lado, são uma maneira de adicionar metadados arbitrários não identificáveis, ou bagagem, a objetos. As notations são frequentemente usadas para

ferramentas de configuração declarativa; construir, liberar ou criar informações de imagem; ou informações de contato das pessoas responsáveis.

Este é um trecho do The Gorilla Guide to Kubernetes in the Enterprise , escrito por Joep Piscaer.

Veja Capítulo 1: O panorama do desenvolvimento Alterar .

Você pode baixar o guia completo aqui.

2. CONFIGURANDO UM CLUSTER AWS EKS

1. Criar uma *role* para o EKS
    a. IAM > Roles > EKS
    b. Selecione a opção “Service” e “EKS”
2. Criar role conforme link https://docs.aws.amazon.com/eks/latest/userguide/service_IAM_role.html#create-service-role
3. Acessar o CloudFormation e criar uma stack com o nome “EKS-VPC” para provisionar uma VPC com subnets alocadas em 3 (três) diferentes zonas de disponibilidade. Utilizar o *template* disponível em https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/amazon-eks-vpc-sample.yaml
    a. O stack acima irá criar: 1(uma) VPC, 3 (três) subnets e 1 (um) security group.
4. Criar o cluster no EKS. O nome do cluster será EKSDeepDive e deverá ser associada a role criada no passo [2].
5. Provisionar uma instância EC2 com o Amazon Linux 2, na mesma VPC do cluster do EKS.
6. Configurar o kubectl em na instância para gerenciar o cluster EKS.
    a. mkdir $HOME/bin
    b. curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/\1.12.10/2019-08-14/bin/linux/amd64/kubectl
    c. chmod +x ./kubectl
    d. cp ./kubectl $HOME/bin/kubectl
    e. export PATH=$HOME/bin:$PATH
    f. echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
    g. kubectl version --client
7. Instalar o aws-iam-authenticator
    a. mkdir $HOME/bin
    b. curl -o aws-iam-authenticator https://amazon-eks.s3-us-west-2.amazonaws.com/\1.10.3/2018-07-26/bin/linux/amd64/\aws-iam-authenticator
    c. chmod +x ./aws-iam-authenticator
    d. cp ./aws-iam-authenticator $HOME/bin
    e. export PATH=$HOME/bin:$PATH
    f. echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
    g. aws-iam-authenticator help
8. Configurar o kubectl para o EKS
    a. curl -O https://bootstrap.pypa.io/get-pip.py
    b. python get-pip.py --user
    c. pip install awscli --upgrade --user
    d. export PATH=$HOME/.local/bin:$PATH
    e. echo 'export PATH=$HOME/.local/bin:$PATH' >> ~/.bashrc
    f. aws configure
    g. aws eks update-kubeconfig --name EKSDeepDive

h. kubectl config view
i. kubectl get svc
j. kubectl cluster-info

3. CONFIGURANDO NODE GROUPS NO EKS

1. Criando um node group gerenciado no EKS: https://docs.aws.amazon.com/eks/latest/userguide/create-managed-node-group.html
2. Criando um EKS node IAM Role: https://docs.aws.amazon.com/eks/latest/userguide/create-node-role.html#create-worker-node-role

4. DEPLOY NO EKS A PARTIR DE UMA IMAGEM PÚBLICA NO ECR

1. Vamos conhecer o ECR Public Gallery, disponível em: https://gallery.ecr.aws/?page=1
2. Primeiramente, precisam ser criados dois arquivos YAML para o deploy no Kubernetes: o “deployment” e o “service”.
3. Para realizar o deployment:
    a. kubectl apply -f deployment.yaml
4. Após realizar o deployment das aplicações, pode-se consultar os deployments:
    a. kubectl get deployments
5. Agora, deve-se realizar a criação dos serviços.
    a. kubectl apply -f service.yaml
6. Após realizar o service das aplicações, pode-se consultar os services:
    a. kubectl get svc
7. Para remover os serviços ou os deployments, basta acessar as pastas onde estão os .yaml e executar:
    a. kubectl delete -f service.yaml
    b. kubectl delete -f deployment.yaml

Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld
spec:
  selector:
    matchLabels:
      run: helloworld-backend
  replicas: 2
  template:
    metadata:
      labels:
        run: helloworld-backend
    spec:
      containers:
        - name: helloworld
          image: public.ecr.aws/nginx/nginx:1.21-perl
          ports:
            - containerPort: 80
              protocol: TCP
```

Service.yaml

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  name: helloworld-backend
spec:
  ports:
  - port: 80
    nodePort: 31479
    protocol: TCP
    targetPort: 80
  selector:
    run: helloworld-backend
  type: LoadBalancer
```

5. CRIANDO UMA IMAGEM ECR CUSTOMIZADA

1. Execute o passo a passo descrito no procedimento disponível em: https://docs.aws.amazon.com/AmazonECR/latest/userguide/getting-started-cli.html
2. Após a execução do fluxo, realize as etapas refaça a etapa 4, considerando a imagem criada neste passo.

ANEXOS | PRINCIPAIS COMANDOS KUBERNETES

- kubectl config view
- kubectl cluster-info
- kubectl get nodes
- kubectl get deployments
- kubectl get svc
- kubectl describe deployment **deploymentname**
- kubectl describe service **servicename**

INFORMAÇÕES ADICIONAIS IMPORTANTES

Como corrigir um erro de servidor não autorizado ao me conectar ao servidor da API do Amazon EKS?

https://aws.amazon.com/pt/premiumsupport/knowledge-center/eks-api-server-unauthorized-error/