Comandos básicos para Kubernetes

Minikube:

* + minikube start (inicializa minikube)
  + minikube stop (finaliza minikube)
  + minikube status (status do ambiente criado)
  + minikube delete (limpa tudo - deleta clusters)

Kubectl:

* + kubectl version —client
  + kubectl run <nome-atribuído ao pod> —image <nome-pod>
  + kubectl get pods (lista pods ativos)
  + kubectl get pods -o wide (apresenta mais detalhes dos pods ativos)
  + kubectl describe pod <nome\_do\_pod> (apresenta detalhes do pod especificado)
  + kubectl create -f <nome\_arq\_yaml> (cria pod a partir de um arquivo yaml)
  + kubectl apply -f <nome\_arq\_yaml> (outra forma de criar um pod a partir de um arquivo yaml)
  + kubectl delete pod <nome\_do\_pod>

Replication Controller e ReplicaSets

Verificar:

* + kubectl get replication controller
  + kubectl get replicates

Criar:

* + kubectl create -f <nome\_arq\_yaml>
  + kubectl create -f <caminho\_arquivo>/<nome\_arq\_yaml>

Atualiza número de réplicas:

* + Alterar o número de réplicas no arquivo .yaml usado para starter o ReplicationSet
  + kubectl scale —replicates <nome\_metadata> —replicas=<número\_de\_replicas>

Deletar:

* + kubectl delete replicaset

Deployments:

* + kubectl create -f <caminho\_arquivo>/<dp.yaml> (cria um objeto deployment)
  + kubectl get deployment (consulta deployments criados)
  + kubectl get pods (consulta pods criados através do deployment)
  + kubectl get replicates (consulta replicaset criado pelo próprio deployment)
  + kubectl get all (permite que se consulte tudo o que foi criado a partir do deployment)
  + kubectl describe deployment <nome\_do\_deployment> (verifica detalhes do deployment criado)
  + kubectl rollout status <nome\_do\_repositório>/<nome\_do\_arq.yaml> (retorna o status do deployment)
  + kubectl rollout history deployment/<nome\_do\_deployment> (retorna o histórico de roll out)
  + kubectl apply -f <nome\_do\_repositório>/<nome\_do\_arq.yaml> —save-config (aplica uma mudança na sua aplicação, por exemplo, uma atualização de versão. Evite usar apply - prefira a versão abaixo)
  + kubectl create -f <nome\_do\_repositório>/<nome\_do\_arq.yaml> —save-config (aplica uma mudança na sua aplicação, por exemplo, uma atualização de versão)
  + kubectl rollout undo <nome\_do\_repositório>/<nome\_do\_deployment> (se tudo der errado, volte para estado anterior)
  + kubectl rollout undo <nome\_do\_repositório>/<nome\_do\_deployment> —to-revision=<número\_do\_deploy\_desejado> (se tudo der errado, volte para estado específico)
  + Kubectl scale <nome\_diretório>/<nome\_arq.yaml> —replicas=<num\_desejado> (para escalar up ou down sua aplicação.
  + kubectl run nginx —image=nginx (exemplo de publicação direta usando kubectl - este comando cria um pod e um ReplicaSet. Deve ser evitado seu uso)

Redes com Kubernetes

Fundamentos: idealmente cada serviço deve ser criado no seu próprio container, isso não significa que não se possa ter mais de um serviço em um container, mas essa é uma prática a ser evitada.

Se tivermos um único pod no cluster, tudo fica mais fácil, isso é boa prática de desenvolvimento. Tendo um único pod no cluster, tudo fica fácil, pois os contadores podem se comunicar entre eles através de *localhost*.

Um cluster ao “nascer" traz consigo dois endereços IP, sendo um deles o endereço IP do próprio cluster e o outro o endereço da apiServer (endereço utilizado para acessar o cluster via SSH e etc). Há três comandos básicos aqui que nos permite ter essa visão:

Atenção: em se tratando de kubernets, diferentemente daquilo que acontece no docker (o container é que recebe o endereço IP), quem recebe o endereço IP (Kubernetes) é o POD.

Quando criamos um cluster, além do IP principal do cluster e do IP da apiServer, é criada uma rede que irá endereçar os PODs a medida que são criados.

Obs.: os podes podem se comunicar entre eles através dos endereços IP gerados.

Em todo caso, não é recomendável fazer a comunicação/acesso de um pod com outro através do endereço IP, pois este endereço pode mudar com a recriação do pod, que ocorre por exemplo, com uma atualização. Recomenda-se para solucionar esse problema trabalhar com nomes dos serviços criados.

KubeDNS

Dentre os serviços que são instalados junto com o Kubernetes está o KubeDNS que faz toda a parte de tradução de nomes para endereços IP assim como um servidor DNS normal.

Nota: para evitar problemas de redes com Kubernetes lembre-se que todos os containers/PODs podem se comunicar entre-si sem o uso de NAT. Na verdade, todos os nodes podem se comunicar com todos os containers sem o uso de NAT. Para se resolver todas essas questões, utilize técnicas de roteamento.

Funcionamento de namespace no Kubernetes

Além de conhecermos os conceitos básicos de redes no Kubernetes devemos entender também o funcionamento de um conceito chamado namespaces.

NameSpaces é um “espaço" na qual separamos nossos recursos dentro do Kubernetes em áreas separadas.

É aqui que inicia o conceito de arquitetura de micro-serviços…

Basicamente o conceito de NameSpace consiste em agrupara pods que fazem parte de uma mesma área em um NameSpace. Mas atenção, se ao criar um objeto Kubernetes não especificar um NameSpace para ele, será colocado no NameSpace padrão (default) do Kubernetes.

Atenção: não é recomendado criar objetos Kubernetes nos NameSpaces internos do Kubernetes - objetos internos são todos aqueles que iniciam seus nomes com a palavra kube\_.

Podemos criar objetos sem especificar nenhum NameSpace, que irá criar os objetos no NameSpace default, ou mesmo especificar um NameSpace próprio para o objeto.

O problema de criar NameSpaces próprios ao invés de usar o NameSpace default do Kubernetes é que você deverá lembrar e informar os NameSpaces próprios existentes sempre que for criar um objeto, logo, a dica aqui é: *use o padrão (isso para iniciantes ok)*.

Nota: é possível criar NameSpace utilizando YAML, tendo como base a seguinte estrutura como top-level:

apiVersion: V1

kind: NameSpace

metadata:

name: <nome\_NameSpace>

Descoberta de Serviços em Kubernetes

O serviço Kubernetes está acessível a partir do endereço NameServer: <IP> que esta presente no arquivo de configuração <resolv.conf> que pode ser acessado a partir do diretório /etc.

cd /etc - muda para o diretório /etc

No diretório /etc

cat resolv.conf - você tem acesso aos dados de configuração do pod apontando para o servidor DNS a partir do qual será possível acessar todos os serviços.

É preciso também compreender o FQDN - Fully Qualified Domain Name no Kubernetes - ao usar o comando kubectl get all, dentre as informações de retorno uma é importante para esse entendimento, trata-se do service/kubernetes. Para analisar entre no pod e use o comando: nslookup <nome\_serviço>, dentre os valores de retorno você tem o Name: kubernetes.default.svc.cluster.local, que é o valor do FQDN.

Observe agora ao usar o comando cat em /etc o resultado (search **default.svc.cluster.local** svc.cluster.local cluster.local), note que é o nome do FQDN:

root@webapp-5db44756bc-997vr:/# cat /etc/resolv.conf

nameserver 10.96.0.10

search default.svc.cluster.local svc.cluster.local cluster.local

options ndots:5

Decifrando cada uma das partes:

kubernetes - nome do serviço

default - NameSpace padrão do Kubernetes

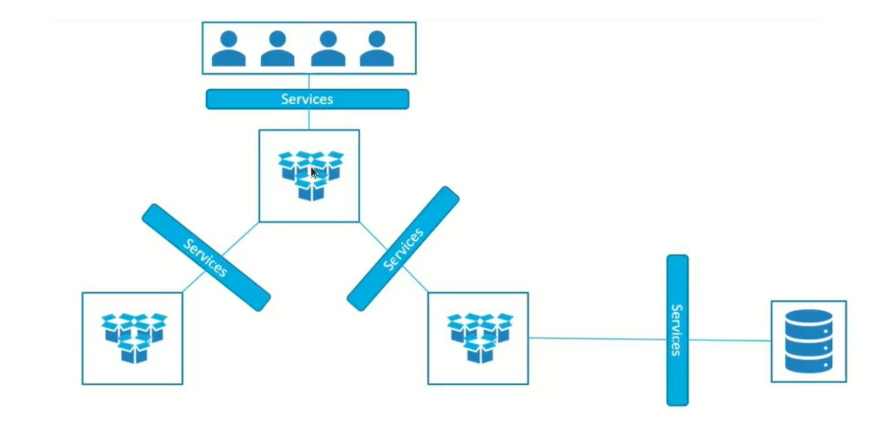
Sic - Serviço (kind: Service)

cluster.local - Nome do cluster

Serviços com Kubernetes

Services em Kubernetes permite a comunicação entre os diversos componentes de uma aplicação com o mundo externo a ela.

Permite também que se possa conectar facilmente uma aplicação com outra e claro com usuários externos.



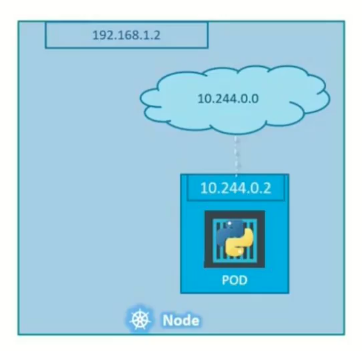
Serviços em Kubernetes.

Nota: os services permitem que se crie uma arquitetura de microserviços para uma aplicação.

Importante: sabemos como ocorre a comunicação via com objetos do Kubernetes. Até o momento não conseguimos acessar nossa aplicação web através do nosso navegador web, isso porque, os pods não possuem comunicação externa ao cluster, esse é um problema a ser resolvido com o Services.

Note que o cluster tem um endereço IP, mas internamente ele possui outra rede na qual distribui endereçamento IP para os pods criados.

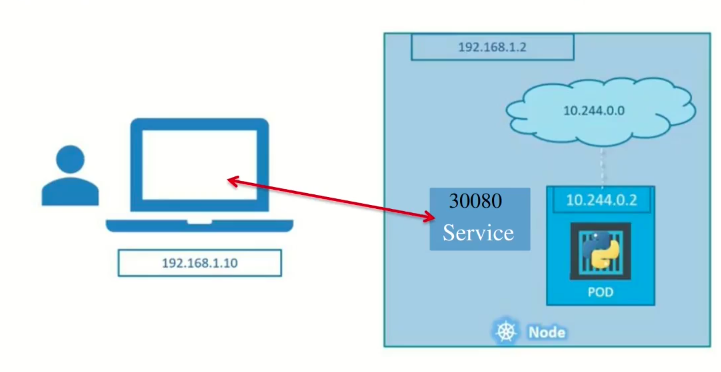
Nosso computador tem um endereço IP que é da mesma rede do cluster, mas difere dos endereços dos pods, logo, se estão em redes diferentes não se comunicam diretamente.



Legenda

Se nós acessarmos o shell de um dos pods e tentarmos acessar o serviço HTTP do outro pod termos acesso. Mas essa não é uma operação interessante.

É aí que entra o Service, que habilita uma porta de acesso para que possamos acessar o serviço e que estiver disponível. Pode parecer estranho o número alto da porta disponível, mas isso ocorre apenas durante o desenvolvimento, para evitar conflito de portas com outros serviços. Na produção, serviços como Load Balance fará o roteamento para a porta padrão do serviço.



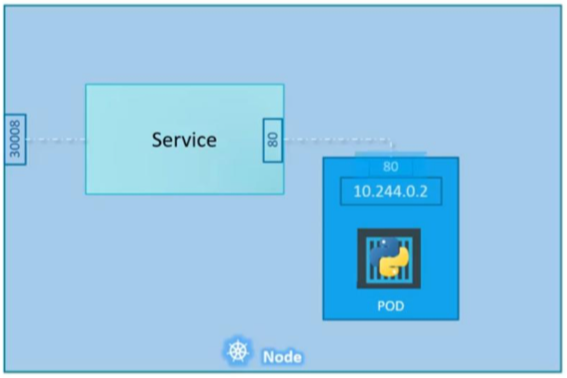
Legenda

O NodePort no Kubernetes

A figura acima reporta os três tipos de serviços disponíveis para uso em Kubernetes, NodePort (permite acesso externo ao node), ClusterIP (serviços internos, não libera acesso ao mundo externo) e LoadBalancer (para balancear a carga de sua aplicação e permite acesso ao mundo externo).

NodePort (serviço público) - abre uma comunicação entre o pod e agentes externos, como navegadores web se o serviço disponibilizado for web, por exemplo.

ClusterIP - é usado quando a comunicação entre os serviços é apenas interna (privada) ao cluster, sem acesso à agentes externos.

LoadBalancer - habilita a distribuição de carga entre os diversos pods que estão realizando o serviço. Este recurso não funciona localmente, mas sim em provedores cloud.

A porta 30008 é a porta NodePort - que faz a comunicação externa com serviços (esta porta precisa estar no range entre 30000 e 32767).

O nosso service esta na porta 80(Service) que faz comunicação com o pod (port)

E por fim o 80(Pod) - porta do servidor web (Target Port).

Exemplo de uma estrutura mínima para se criar um NodePort:

Atenção: caso o targetPort não seja informado irá assumir o mesmo valor de port. Caso nodePort não seja informado irá assumir um valor aleatório dentro do range definido anteriormente.

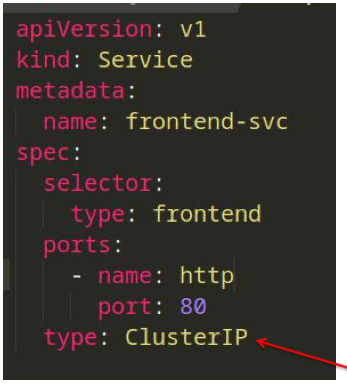
Nota: fazendo uso do serviço NodePort, a partir de então podemos acessar o servidor, por exemplo se o serviço for um servidor web, no endereço do cluster, exemplo: http://192.168.20.2:30008



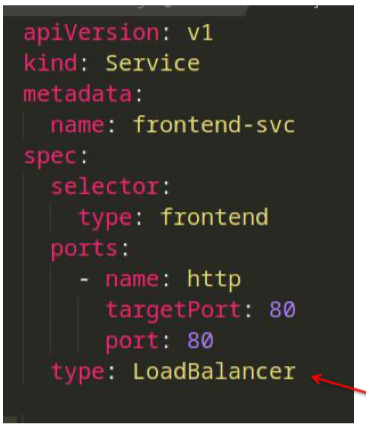
Tipo de Serviço: NodePort

Para um serviço interno, privado temos a seguinte estrutura básica - ClusterIP:

Por fim temos o LoadBalancer que a configuração básica pode ser vista na figura a seguir, mas lembre-se, somente provedores em nuvem com suporte a balanceamento de cargas para Kubernetes permitem seu uso.



Tipo de serviço ClusterIP



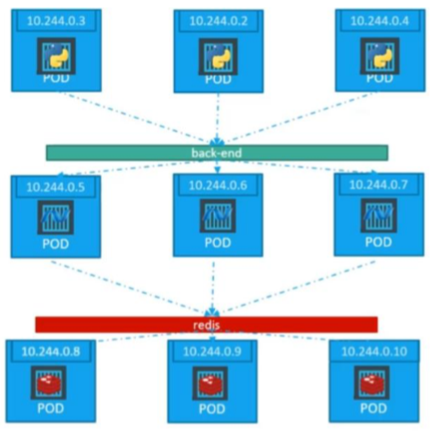
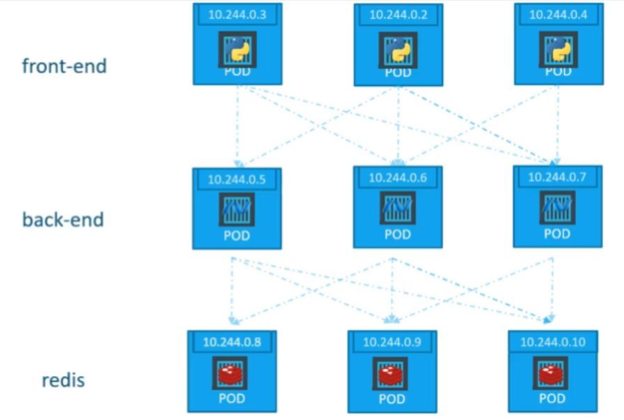
Tipo de serviço LoadBalancer

Importante: na presença de múltiplos PODs dividindo a carga da aplicação, por exemplo, aplicação web, nada muda, pois todos os pods possuem o mesmo labels “app” com valor “myapp”. Desta forma tudo irá funcionar sem nenhuma mudança, fantástico… E mais se os PODs estiverem distribuídos entre múltiplos nodes, o kubernetes cria um serviço englobando os nodes, fazendo com que o acesso ocorresse normalmente.

Trabalhando com ClusterIp no Kubernetes

Numa aplicação Fullstack é muito comum trabalharmos com múltiplos pods, estes pods precisam se comunicar entre si e esta comunicação não deve sair dos domínios do cluster.

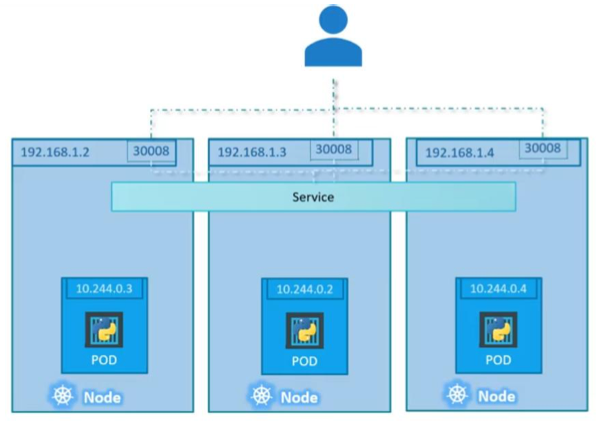
Nota: os pods não possuem IP estático, ou seja, se um pod apresentar problemas e tiver que ser substituído poderá ter um novo endereço IP, uma vez que este é gerado dinamicamente.

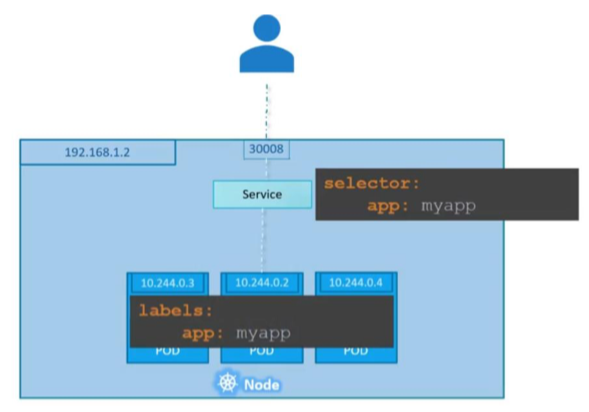
Para solucionar este problema o Service no Kubernetes nos ajuda a “agrupar" os pods que oferecem o mesmo serviço e fornecer uma interface única de acesso aos serviços. A partir do momento em que o serviço foi criado, o serviço passa a ser a interface, não mais sendo preciso depender o IP.

Pode-se dizer que o Service ClusterIP serve justamente para este fim, deixando a comunicação entre os pods funcionarem apenas internamente ao cluster, sem permitir acesso externo.

Desta forma, podemos criar microserviços que juntos, farão parte de toda aplicação.

Cada microserviço recebe um nome e um endereço IP que deve ser usado pelos outros microserviços para realizar o acesso.



Alguns comandos interessantes

* + kubectl get all - traz um resumo dos dados dos clusteres (todos os objetos do nosso cluster Kubernetes que pertencem ao NameSpace padrão)
  + kubectl get namespaces - pega todos os objetos de todos os namespaces
  + kubectl get ns - (atalho para o comando acima) pega todos os objetos de todos os namespaces
  + kubectl get pods -n default - pega todos os pods do namespace default/padrão
  + kubectl get pods -n <nome do NameSpace> - pega todos os pods do namespace <nome do NameSpace>
  + kubectl get all -n <nome\_NameSpace> - permite consultar todos os objetos especificando o nome do nameSpace
  + kubectl describe service Kubernetes - descreve os dados referentes ao funcionamento do kubernetes
  + kubectl get service kubernetes - outra forma de se obter dados de endereço IP do Cluster criado
  + kubectl cluster-info - traz um conjunto de informações adicionais sobre os serviços ativos no cluster criado, por exemplo o KubeDNS.
  + kubectl create namespace <nome\_NameSpace> —save-config - cria um NameSpace como o nome <nome\_NameSpace>
  + kubectl create ‘f pods/<nome\_pod> —save-config —namespace=<nome\_NameSpace> - cria um objeto no NameSpace especificado através de seu nome <NameSpace>
  + kubectl config set-context —current —namespace=<NomeSpace> - define um NameSpace criado tendo o nome <NameSpace> para ser o padrão no seu projeto. A partir desse comando, qualquer objeto será criado no NameSpace definido anteriormente, a menos que seja especificado outro NameSpace.
  + kubectl create -f namespaces/<nome\_NameSpace.yaml> —save-config - cria um NameSpace utilizando um arquivo yaml.
  + kubectl exec -it <nome\_do\_pod> -- <comando a executar> - conecta dois pods e executa o comando <comando a executar>
  + kubectl exec -it webapp-5db44756bc-997vr -- bash - exemplo do comando acima executando o pod web no pod MySql rodando o shell.
  + apt update - atualiza o pod em execução
  + apt search mysql-client - busca por um cliente MySql dentro de um pod
  + apt install default-mysql-client -y - para instalar o cliente desejado, no caso default-mysql-client
  + apt install dnsutils -y - instala ferramentas DNS caso não estejam disponíveis
  + nslookup kubernetes - para visualizar informações do serviço chamado kubernetes. Valores de retorno de nslookup, importante para este caso: Name que é na verdade o nosso procurado FQDN

Server: 10.96.0.10

Address: 10.96.0.10#53

Name: kubernetes.default.svc.cluster.local

Address: 10.96.0.1

* + mysql -h 10.244.0.3 -uroot -ppassword accol - exemplo de conexão ao banco de dados mysql, que está no endereço 10.244.0.3 conectando como usuário root (urrot) com a senha (password) no banco accol. Como isso, fecha-se um exemplo de comunicação via rede entre pods.
  + Para sair do pod digite - <CTRL + D> - sai do nível corrente de pod que estiver conectado, continue com <CTRL+D> até que saia de todos os pods desejados
  + kubectl create -f service-definition.yml - este comando permite a criação de um serviço no contexto de Kubernetes
  + kubectl get services - este comando exibe os Services disponíveis presentes no NameSpace padrão/default.
  + minikube ip - retorna o ip da rede
  + minikube service <nome\_serviço> —url - devolve o IP e a porta do serviço que tem por nome <nome\_serviço>
  + minikube start —driver=hyperkit - esta opção é importante para se trabalhar com serviços no Windows e MacOs, isso, aparentemente acontece pelo fato do minkube utilizar por default o driver do Docker, que utiliza um esquema de rede diferentemente quando se trata destes dois sistemas operacionais (Windows e MacOS). Atenção pode haver problemas aqui, fique atento …
  + kubectl create -f <nome\_diretorio>/ —save-config - cria todos arquivos que estiverem descritos no <nome\_diretorio>
  + minikube service <nome\_Serviço> —url -n <nome\_NameSpace> - retorna o endereço IP do <nome\_Serviço> dentro do <nome\_Namespace> desejado, caso não especifique <nome\_NameSpace> terá como retorno o IP do name\_Space default ou um erro, como este "*Exiting due to SVC\_NOT\_FOUND: Service 'result' was not found*”. O serviço não foi encontrado no NameSpace default.
  + kubectl get nodes - exibe quantos nodes estão disponíveis - para aplicação em nuvem.