Inicializando o cluster

Agora que temos tudo configurado, vamos iniciar o nosso cluster:

```
sudo kubeadm init --pod-network-cidr=10.10.0.0/16 --apiserver-advertise-address=<O IP QUE VAI FALAR COM OS NODES>
```

Substitua <O IP QUE VAI FALAR COM OS NODES> pelo endereço IP da máquina que está atuando como control plane.
Após a execução bem-sucedida do comando acima, você verá uma mensagem informando que o cluster foi inicializado com sucesso. Além disso, você verá um comando para configurar o acesso ao cluster com o kubectl. Copie e cole esse comando em seu terminal:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Essa configuração é necessária para que o kubectl possa se comunicar com o cluster, pois quando estamos copiando o arquivo admin.conf para o diretório .kube do usuário, estamos copiando o arquivo com as permissões de root, esse é o motivo de executarmos o comando sudo chown $(id -u):$(id -g) $HOME/.kube/config para alterar as permissões do arquivo para o usuário que está executando o comando.

Entendendo o arquivo admin.conf

Agora precisamos entender o que temos dentro do arquivo admin.conf. Antes de mais nada precisamos conhecer alguns pontos importantes sobre o a estrutura do arquivo admin.conf:

- É um arquivo de configuração do kubectl, que é o cliente de linha de comando do Kubernetes. Ele é usado para se comunicar com o cluster Kubernetes.

- Contém as informações de acesso ao cluster, como o endereço do servidor API, o certificado de cliente e o token de autenticação.
- Eu posso ter mais de um contexto dentro do arquivo admin.conf, onde cada contexto é um cluster Kubernetes. Por exemplo, eu posso ter um contexto para o cluster de produção e outro para o cluster de desenvolvimento, simples como voar.
- Ele contém os dados de acesso ao cluster, portanto, se alguém tiver acesso a esse arquivo, ele terá acesso ao cluster. (Desde que tenha acesso ao cluster, claro).
- O arquivo admin.conf é criado quando o cluster é inicializado.

Vou copiar aqui o conteúdo de um exemplo de arquivo admin.conf:

```
apiVersion: v1

clusters:
- cluster:
    certificate-authority-data: SEU_CERTIFICADO_AQUI
    server: https://172.31.57.89:6443
  name: kubernetes

contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes

current-context: kubernetes-admin@kubernetes

kind: Config

preferences: {}

users:
- name: kubernetes-admin
  user:
```

```
    client-certificate-data: SUA_CHAVE_PUBLICA_AQUI
    client-key-data: SUA_CHAVE_PRIVADA_AQUI
```

Simplificando, temos a seguinte estrutura:

```
apiVersion: v1
clusters:
#...
contexts:
#...
current-context: kind-kind-multinodes
kind: Config
preferences: {}
users:
#...
```

Vamos ver o que temos dentro de cada seção:

Clusters

A seção clusters contém informações sobre os clusters Kubernetes que você deseja acessar, como o endereço do servidor API e o certificado de autoridade. Neste arquivo, há somente um cluster chamado kubernetes, que é o cluster que acabamos de criar.

```
- cluster:
    certificate-authority-data: SEU_CERTIFICADO_AQUI
    server: https://172.31.57.89:6443
  name: kubernetes
```

Contextos

A seção contexts define configurações específicas para cada combinação de cluster, usuário e namespace. Nós somente temos um contexto configurado. Ele é chamado kubernetes-admin@kubernetes e combina o cluster kubernetes com o usuário kubernetes-admin.

```
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
```

Contexto atual

A propriedade current-context indica o contexto atualmente ativo, ou seja, qual combinação de cluster, usuário e namespace será usada ao executar comandos kubectl. Neste arquivo, o contexto atual é o kubernetes-admin@kubernetes.

```
current-context: kubernetes-admin@kubernetes
```

Preferências

A seção preferences contém configurações globais que afetam o comportamento do kubectl. Aqui podemos definir o editor de texto padrão, por exemplo.

```
preferences: {}
```

Usuários

A seção users contém informações sobre os usuários e suas credenciais para acessar os clusters. Neste arquivo, há somente um usuário chamado kubernetes-admin. Ele contém os dados do certificado de cliente e da chave do cliente.

```
- name: kubernetes-admin
  user:
    client-certificate-data: SUA_CHAVE_PUBLICA_AQUI
    client-key-data: SUA_CHAVE_PRIVADA_AQUI
```

Outra informação super importante que está contida nesse arquivo é referente as credenciais de acesso ao cluster. Essas credenciais são usadas para autenticar o usuário que está executando o comando kubectl. Essas credenciais são:

- Token de autenticação: É um token de acesso que é usado para autenticar o usuário que está executando o comando kubectl. Esse token é gerado automaticamente quando o cluster é inicializado. Esse token é usado para autenticar o usuário que está executando o comando kubectl. Esse token é gerado automaticamente quando o cluster é inicializado.

- certificate-authority-data: Este campo contém a representação em base64 do certificado da autoridade de certificação (CA) do cluster. A CA é responsável por assinar e emitir certificados para o cluster. O certificado da CA é usado para verificar a autenticidade dos certificados apresentados pelo servidor de API e pelos clientes, garantindo que a comunicação entre eles seja segura e confiável.

- client-certificate-data: Este campo contém a representação em base64 do certificado do cliente. O certificado do cliente é usado para autenticar o usuário ao se comunicar com o servidor de API do Kubernetes. O certificado é assinado pela autoridade de certificação (CA) do cluster e inclui informações sobre o usuário e sua chave pública.

- client-key-data: Este campo contém a representação em base64 da chave privada do cliente. A chave privada é usada para assinar as solicitações enviadas ao servidor de API do Kubernetes, permitindo que o servidor verifique a autenticidade da solicitação. A chave privada deve ser mantida em sigilo e não compartilhada com outras pessoas ou sistemas.

Esses campos são importantes para estabelecer uma comunicação segura e autenticada entre o cliente (geralmente o kubectl ou outras ferramentas de gerenciamento) e o servidor de API do Kubernetes. Eles permitem que o servidor de API verifique a identidade do cliente e vice-versa, garantindo que apenas usuários e sistemas autorizados possam acessar e gerenciar os recursos do cluster.

Você pode encontrar os arquivos que são utilizados para adicionar essas credentiais ao seu cluster em `/etc/kubernetes/pki/`. Lá temos os seguintes arquivos que são utilizados para adicionar essas credenciais ao seu cluster:

- client-certificate-data: O arquivo de certificado do cliente geralmente é encontrado em /etc/kubernetes/pki/apiserver-kubelet-client.crt.

- client-key-data: O arquivo da chave privada do cliente geralmente é encontrado em /etc/kubernetes/pki/apiserver-kubelet-client.key.
- certificate-authority-data: O arquivo do certificado da autoridade de certificação (CA) geralmente é encontrado em /etc/kubernetes/pki/ca.crt.

Vale lembrar que esse arquivo é gerado automaticamente quando o cluster é inicializado, e são adicionados ao arquivo admin.conf que é utilizado para acessar o cluster. Essas credenciais são copiadas para o arquivo admin.conf já convertidas para base64.

Pronto, agora você já sabe o porquê copiamos o arquivo admin.conf para o diretório ~/.kube/ e como ele funciona.
Caso você queira, você pode acessar o conteúdo do arquivo admin.conf com o seguinte comando:

```
kubectl config view
```

Ele somente irá omitir os dados de certificados e chaves privadas, que são muito grandes para serem exibidos no terminal.