Troubleshooting Kubernetes Cluster Access Issues

0s left

100% **Open Cloud Environment**
Setup completed
Average setup time: 2m 27s

100% **Open Code Environment**
Setup completed

Credentials

Account ID
33584053 Copy

Username
student Copy

Password
Ca1_q0Eg5 Copy

Region
US West 2 Copy

PEM Download PPK Download

Bastion Public Ip
34.212.51 Copy

Cluster SSH
ssh ubunt

Introduction

In this lab step, you will troubleshoot issues that relate to communicating with and accessing a Kubernetes cluster. The bastion host has `kubectl` installed, but has not been configured correctly to talk to the cluster. You will learn how to diagnose and resolve this problem and more related to accessing a cluster.

Instructions

1. Attempt to list the nodes in the cluster:

Copy code

```
kubectl get nodes
```

```
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

The result is a **connection refused** message. By default, `kubectl` will attempt to connect to the Kubernetes API server on port 8080 of the localhost. The bastion host does not have an API server so there is nothing to connect to. You need to configure kubectl to connect to the API server running on the control-plane node. `kubectl` is configured to connect to clusters by using kubeconfig files. By default, `kubectl` looks for a kubeconfig file `~/.kube/config`. This file does not exist on the bastion host. You can copy the kubeconfig file on the control-plane to get it working on the bastion host.

2. Enter the following commands to copy the control-plane node's kubeconfig file, and enter *yes* when prompted about the authenticity of the host:

Copy code

```
mkdir .kube  # create the .kube directory
scp -o "ForwardAgent yes" control-plane:.kube/confi
```

Support

The kubeconfig file that you copied from the control-plane node includes:

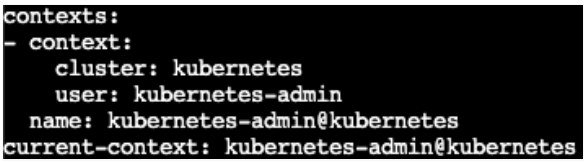
1. Information about the cluster, such as the server address
2. Information about users to authenticate as including certificates that were generated when the cluster was created

3. View the contents of the kubeconfig file:

Copy code

```
cat ~/.kube/config
```

Most of the contents are certificate and key data, but you can notice that there are several top-level keys including **clusters**, **contexts**, **current-context**, and **users**. The following image highlights the **contexts** and **current-context** keys:



A context is a triple of a **cluster** (**kubernetes**), a **user** (**admin**), and a namespace (if not specified, the default namespace is used). The context is also given a name for reference (**admin@kubernetes**). The **current-context** sets the context that will be used by `kubectl` by default. To manage the configuration of `kubectl`, you can use its `config` command.

4. View the `config` commands provided by `kubectl`:

Copy code

```
kubectl config --help
```

Lab Steps

1. Connecting to the K8s Cluster
2. **Troubleshooting Kubernetes Cluster Access Issues**

Need help? Contact our support team

```
your system). These paths are merged. When a value is modified, it is modified in the file that defines the stanza. When
a value is created, it is created in the first file that exists. If no files in the chain exist, then it creates the
last file in the list.
  3.  Otherwise, ${HOME}/.kube/config is used and no merging takes place.

Available Commands:
  current-context   Display the current-context
  delete-cluster    Delete the specified cluster from the kubeconfig
  delete-context    Delete the specified context from the kubeconfig
  delete-user       Delete the specified user from the kubeconfig
  get-clusters      Display clusters defined in the kubeconfig
  get-contexts      Describe one or many contexts
  get-users         Display users defined in the kubeconfig
  rename-context    Rename a context from the kubeconfig file
  set               Set an individual value in a kubeconfig file
  set-cluster       Set a cluster entry in kubeconfig
  set-context       Set a context entry in kubeconfig
  set-credentials   Set a user entry in kubeconfig
  unset             Unset an individual value in a kubeconfig file
  use-context       Set the current-context in a kubeconfig file
  view              Display merged kubeconfig settings or a specified kubeconfig file
```

These commands can be used to safely write to kubeconfig files using the **delete**, **set**, **unset**, and **use** commands. Assuming the configuration file has the correct cluster and user information, you may have issues connecting to a cluster because the current context is not configured at all; it is set to a different context than you expect.

5. Enter the following `config` command to get a summary of the contexts available in a kubeconfig file:

Copy code

```
kubectl config get-contexts
```

```
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin
```

This view is useful for showing you all of the configured contexts and which is the **CURRENT** context. If there is no current context set, there will be no * in the first column. In this lab, there is only one context, but you could have several contexts in practice. There are also multiple contexts for different clusters in the Kuberenetes certification exams. If the current context is not what you want, you can use the `use-context` command to change it.

6. Re-attempt to list the nodes in the cluster:

Copy code

```
kubectl get nodes
```

```
NAME                                        STATUS   ROLES
ip-10-0-0-10.us-west-2.compute.internal     Ready    <none>
ip-10-0-0-100.us-west-2.compute.internal    Ready    control-plane
ip-10-0-0-11.us-west-2.compute.internal     Ready    <none>
```

With the kubeconfig file in the default location of ~/.kube/config and the
unicate with the
ror, it would most
ues. If you received

7. Confirm the admin user is authorized to list all nodes:

Copy code

```
kubectl auth can-i list nodes -A
```

The `can-i` command will return a binary response for whether or not a user is authorized to perform a specified action. You can learn more about the actions from the command's help page (`kubectl auth can-i --help`). The admin user is authorized because it is bound to an RBAC role that grants permission. Kubernetes does not store any user resources. Instead, role binding resources store information about the names of users or groups that are assigned to a role. Roles can be for a specific namespace (Role) or for an entire cluster (ClusterRole). Similarly, role bindings can be for a specific namespace (RoleBinding) or for an entire cluster (ClusterRoleBinding).

8. List all of the cluster roles:

Copy code

```
kubectl get clusterroles
```

Kubernetes creates several cluster roles by default. The one that is germane to this exercise is the **cluster-admin** role.

9. Show the cluster-admin cluster role resource YAML:

Copy code

```
kubectl get clusterrole cluster-admin -o yaml
```

```
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  creationTimestamp: "2022-08-09T19:47:23Z"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: cluster-admin
  resourceVersion: "78"
  uid: b698d652-4d9f-4e38-b9fe-d5ec31b2121a
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

The **rules** key allows all actions (**verbs**) on all **resources**. You can get the same information from `kubectl describe clsuterrole cluster-admin`, but it is useful to use to `-o yaml` option on `get` commands to quickly create templates from existing resources. This is very useful when you need to create new resources and cannot remember which **apiVersion** to use or what the names of certain keys are.

10. Describe the admin-cluster-binding cluster role binding to understand how the cluster-admin role is bound to users:

Copy code

```
kubectl describe clusterrolebinding admin-cluster-binding
```

```
Name:         admin-cluster-binding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  cluster-admin
Subjects:
  Kind  Name   Namespace
  ----  ----   ---------
  User  admin
```

The **Role** map specifies the **Name** of the role that is being bound, and the **Subjects** map lists all the subjects (users, groups, or service accounts) that are bound to the role. In this case, the role is bound to

side of Kubernetes, For completeness,

11. Extract the admin certificate from the kubeconfig file, and use OpenSSL to show the certificate details:

Copy code

```
grep "client-cert" ~/.kube/config | \
  sed 's/\(.*client-certificate-data: \)\(.*\)/\2/' | \
  base64 --decode \
  > cert.pem
openssl x509 -in cert.pem -text -noout
```

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4780407784204344386 (0x42576b8d53031c42)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = kubernetes
        Validity
            Not Before: Aug  9 19:47:04 2022 GMT
            Not After : Aug  9 19:47:06 2023 GMT
        Subject: O = system:masters, CN = kubernetes-admin
```

The **Subject** shows the **admin** common name. Kubernetes maps the common name to users and the organization (if present) to groups. You now understand the full authorization chain:

1. The context uses an admin client certificate
2. There is a cluster role binding between the admin user and the cluster-admin cluster role
3. The cluster-admin cluster role grants all actions on all resources

Summary

In this lab step, you learned how to diagnose and resolve issues related to communicating with the desired Kuberentes cluster using `kubectl`. You understood the role of kubeconfig files, contexts, and `config` commands to ensure that `kubectl` is properly configured to communicate with the target cluster. You also reviewed roles and role bindings, and how they allow the admin user to perform any action on the cluster.

VALIDATION CHECKS

1 Checks Start check