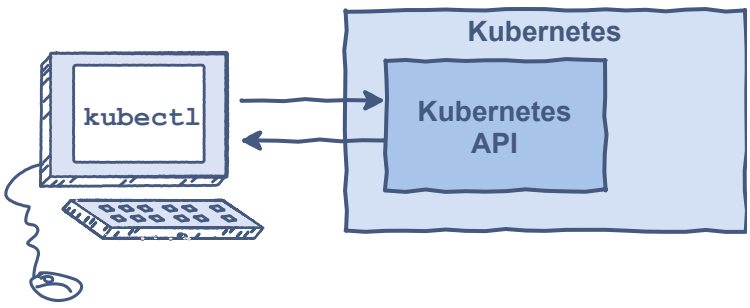
Accessing Kubernetes API

In this lesson, we will explore the Kubernetes API and the process to access it.

WE'LL COVER THE FOLLOWING

The API

Every interaction with Kubernetes goes through its API and needs to be authorized. That communication can be initiated through a user or a service account. All Kubernetes objects currently running inside our cluster are interacting with the API through service accounts. We won't go deep into those. Instead, we'll concentrate on the authorization of human users.



> ℹ All the commands from this chapter are available in the 12-auth.sh Gist.

Checking out the Port

Typically, the Kubernetes API is served on a secured port. Our Minikube cluster is no exception. We can check the port from the `kubectl` config.

```
kubectl config view \
    -o jsonpath='{.clusters[?(@.name=="minikube")].cluster.server}'
```

We used `jsonpath` to output the `cluster.server` entry located in the cluster with the name `minikube`.

The **output** is as follows.

```
https://192.168.99.105:8443
```

We can see that `kubectl` accesses the Minikube Kubernetes API on the port `8443`. Since the access is secured, it requires certificates which are stored as the `certificate-authority` entry. Let's take a look.

```
kubectl config view \
    -o jsonpath='{.clusters[?(@.name=="minikube")].cluster.certificate-authority}'
```

The **output** is as follows.

```
/Users/vfarcic/.minikube/ca.crt
```

The `ca.crt` certificate was created with the Minikube cluster and, currently, provides the only way we can access the API.

The Real-World Scenario

If this was a "real" cluster, we'd need to enable access for other users as well. We could send them the certificate we already have, but that would be very insecure and would lead to a lot of potential problems. Soon, we'll explore how to enable other users to access the cluster securely. For now, we'll focus on the exploration of the process Kubernetes uses to authorize requests to its API.

Understanding the Process

Each request to the API goes through **three stages**.

- Authentication
- Authorization
- Passing the admission control

Authentication

Kubernetes uses client certificates, bearer tokens, an authenticating proxy, or HTTP basic auth to authenticate API requests through authentication plugins. In the authentication process, the username is retrieved from the HTTP request. If the request cannot be authenticated, the operation is aborted with the status code 401.

Authorization

Once the user is authenticated, the authorization validates whether it is allowed to execute the specified action. The authorization can be performed through ABAC, RBAC, or Webhook modes.

Passing the Admission Control

Finally, once a request is authorized, it passes through admission controllers. They intercept requests to the API before the objects are persisted and can modify them. They are advanced topics that we won't cover in this chapter.

Authentication is pretty standard, and there's not much to say about it. On the other hand, admission controllers are too advanced to be covered just yet.

We're left with the user authorization which we'll explore in the next lesson.