

# Installing kubectl

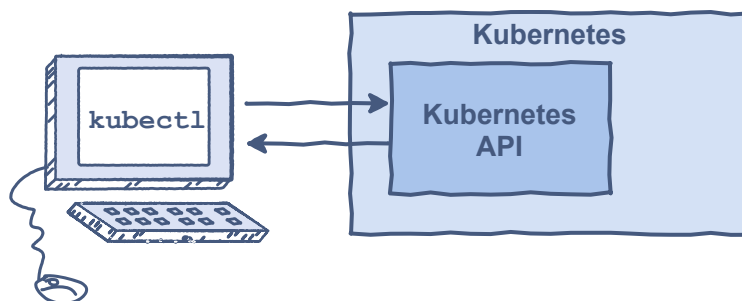
In this lesson, we will learn how to install the Kubernetes command-line tool: kubectl.

## WE'LL COVER THE FOLLOWING ^

- Understanding kubectl
- Installation
  - MacOS
  - Linux
  - Windows
- Verification

## Understanding kubectl #

Kubernetes' command-line tool, `kubectl`, is used to manage a cluster and applications running inside it. We'll use `kubectl` a lot throughout the course, so we won't go into details just yet. Instead, we'll discuss its commands through examples that will follow shortly. For now, think of it as your interlocutor with a Kubernetes cluster.



! All the commands from this chapter are available in the [02-minikube.sh](#) Gist.

## Installation #

Let's install `kubect1`.

Feel free to skip the installation steps if you already have `kubect1`. Just make sure that it is version 1.8 or above.

## MacOS #

If you are a **MacOS user**, please execute the commands that follow.

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.g
chmod +x ./kubect1
sudo mv ./kubect1 /usr/local/bin/kubect1
```

If you already have [Homebrew](#) package manager installed, you can “brew” it with the command that follows.

```
brew install kubect1
```

## Linux #

If, on the other hand, you're a **Linux user**, the commands that will install `kubect1` are as follows.

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.
chmod +x ./kubect1
sudo mv ./kubect1 /usr/local/bin/kubect1
```

## Windows #

Finally, **Windows users** should download the binary through the command that follows.

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.
```

Feel free to copy the binary to any directory. The important thing is to add it to your `PATH`.

## Verification #

Let's check `kubect1` version and, at the same time, validate that it is working correctly. No matter which OS you're using, the command is as follows.

```
kubectl version
```



The **output** is as follows.

```
Client Version: version.Info{Major:"1", Minor:"9", GitVersion:"v1.9.0", GitCommit:"925c127ec6  
The connection to the server localhost:8080 was refused - did you specify the right host or p
```



That is a very ugly and unreadable output. Fortunately, **kubectl** can use a few different formats for its output. For example, we can tell it to output the command in **yaml** format

```
kubectl version --output=yaml
```




The **output** is as follows.

```
clientVersion:  
  buildDate: "2019-04-08T17:11:31Z"  
  compiler: gc  
  gitCommit: b7394102d6ef778017f2ca4046abbbaa23b88c290  
  gitTreeState: clean  
  gitVersion: v1.9.0  
  goVersion: go1.9.2  
  major: "1"  
  minor: "9"  
  platform: darwin/amd64  
The connection to the server localhost:8080 was refused - did you specify the right host or p
```



That was a much better (more readable) output.

We can see that the client version is 1.14.0. At the bottom is the error message stating that **kubectl** could not connect to the server. That is expected since we did not yet create a cluster. That's our next step.

 At the time of writing this course kubectl version was 1.14.0. Your version might be different when you install it on your machine.

---

Our next step is the installation of Minikube.

