Kubernetes in Depth
Debugging Guide

1 Working with Minikube as the K8s installation

It is recommended to start Minikube with Docker as the driver. For this, make sure you have Docker Desktop running and then start it with:

```
minikube start --driver=docker
```

To verify that the Minikube single-node Kubernetes cluster is up and running, check with:

```
minikube status
```

```
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

1.1 Context issues

Occasionally, when running any Minikube command, you will obtain an error message similar to this below:

```
W0721 08:21:36.757796   22204 main.go:291] Unable to resolve the current Docker CLI
context "default": context "default": context not found:
```

This can be resolved with one of the techniques explained in this post (ensure that you have Docker Desktop running before you start working with Minikube).
The first command is the one that is most likely to succeed: `docker context use default`

This issue usually arises if you have multiple installations of Kubernetes (for e.g. using the Docker on Desktop option or Rancher desktop) on a single physical machine, all of which use different contexts when running. In that case, you can check for all the existing contexts matching to these installations with:

```
kubectl config get-contexts
```

and if you wish to use minikube as the Kubernetes cluster for the lab session, then you will need to set:

```
kubectl config use-context minikube
```

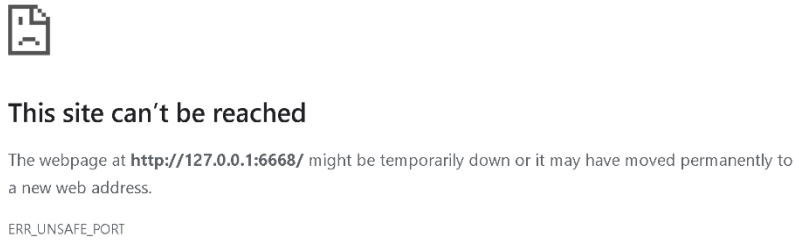
1.2 Port issues when accessing service tunnelling ports via Chrome

Services need to be created in order to expose pods for access: whether this access is from within the cluster or externally.
If you are using Minikube to implement your K8s cluster, you must access the service associated with a deployment with:

```
minikube service deployment-name
```

The PowerShell terminal will freeze at this point and needs to be open in order to keep the tunnelling active. If you press Ctrl+C, the IP address and port specified are no longer valid.

If you are using Chrome and in the new browser tab view, you may obtain an error message similar to the one shown below related to the tunnelling port that Minikube attempts to create on localhost. This is an issue with Chrome blocking access to specific ports for security reasons:



You should still be able to access the port by sending the HTTP request from a REST client such as Postman, or just press Ctrl+C to end the tunneling service and reissuing the command:

```
minikube service first-app-service
```

which will should now provide tunnelling from a new random localhost port, that hopefully will not be among the list of ports blocked by Chrome.

2 General

2.1 Port-related issues

Kubernetes services are used to expose pods within a deployment for access on specific ports on localhost. Occasionally, you may encounter port issues which produce error messages similar to the one below, indicating a potential conflict with another existing process listening on the same port.

```
error: listen tcp 127.0.0.1:8001: bind: An attempt was made to access a socket in a
way forbidden by its access permissions.
```

You can run the following command in a command prompt with admin privilege to identify any process that might be listening on a given *port-number*, for e.g. 8080

```
netstat -aon | findstr :port-number
```

This returns to you the PID of any process that is listening on that given port, for e.g.

```
TCP    0.0.0.0:port-number    0.0.0.0:0              LISTENING       PID
TCP    [::]:port-number       [::]:0                 LISTENING       PID
```

Next you can find the actual name of the process using its *PID* with the command:

```
tasklist | findstr PID
```

This returns the actual name of the process executable

```
someprogram.exe     PID Console                    11    219,532 K
```

Once you have identified the process, you can decide to shut it down if it's not an important process or application that is required. To do this, you can terminate the process with:

```
taskkill /F /PID PID
```

```
SUCCESS: The process with PID PID has been terminated.
```

If there is no process running at that port, then the issue is usually related to NAT service reserving some port ranges by default which cannot be used. To address this, you can simply restart the NAT service from a command prompt with admin privilege as follows:

```
net stop winnat
```

```
net start winnat
```

which should then typically resolve the issue.

2.2 Multiple K8s installations

You may have multiple K8s installations on your machine (as well as active cloud based clusters such as AWS EKS or GKE), in which case, you should always verify which cluster the kubectl CLI tool is configured to work with currently. This is done by examining all the contexts (which contains info for all these different K8s clusters) with:

```
kubectl config get-contexts
```

The current context is highlighted with *

```
CURRENT   NAME              CLUSTER           AUTHINFO          NAMESPACE
          docker-desktop    docker-desktop    docker-desktop
*         minikube          minikube          minikube          default
          rancher-desktop   rancher-desktop   rancher-desktop
```

```
          main-user@demoeks.ap-southeast-1.eksctl.io   demoeks.ap-southeast-1.eksctl.io   main-
user@demoeks.ap-southeast-1.eksctl.io
```

Make sure that you have the correct context set for the particular Kubernetes cluster installation that you wish to interact with. If you are not in the correct context, then switch to the correct one with

`kubectl config use-context` *`new-context`* (which is likely to be `minikube` or `rancher-desktop` or cloud-based cluster)

Note that different K8s installations may implement different versions of K8s, and again you will need to adhere to the recommendation for your `kubectl` client tool to be within one minor version difference of the current K8s installation being used to avoid unforeseen issues. For example, a 1.32 kubectl client works with Kubernetes cluster version 1.31, 1.32 and 1.33.

You can determine the current version of your kubectl client by typing in a PowerShell terminal:

```
kubectl version --client
```