Install Minikube in WSL 2

- Why Minikube for Kubernetes?
- Install Minikube in WSL 2
    - Enable and configure WSL 2 on Windows
    - Install Docker in WSL 2
    - Install Minikube prerequisites
        - Install systemctl
        - Install Conntrack
    - Install Kubectl and Minikube and start the cluster
    - Install Helm to work with Minikube
- Minikube Dashboard
    - Overview
    - Basic usage
    - Getting just the dashboard URL
- Summary

Why Minikube for Kubernetes?

Minikube is a tool that lets you run Kubernetes locally. Minikube runs a single-node Kubernetes cluster on your personal computer (including Windows, macOS, and Linux PCs) so that you can try out Kubernetes, or for daily development work.

Install Minikube in WSL 2

Following is step-by-step guide to install Minikube in WSL 2 with Kubectl and Helm.

Enable and configure WSL 2 on Windows

Please refer How to enable and configure WSL v2 in Windows

Install Docker in WSL 2

Please refer Docker Page for details or use below commands in sequence.

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get -y install apt-transport-https ca-certificates curl gnupg lsb-
release
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
-o /usr/share/keyrings/docker-archive-keyring.gpg

sudo apt-get update

echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo
tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get -y install docker-ce docker-ce-cli containerd.io

sudo usermod -aG docker $USER && newgrp docker

sudo service docker start

sudo service docker status
```

Install Minikube prerequisites

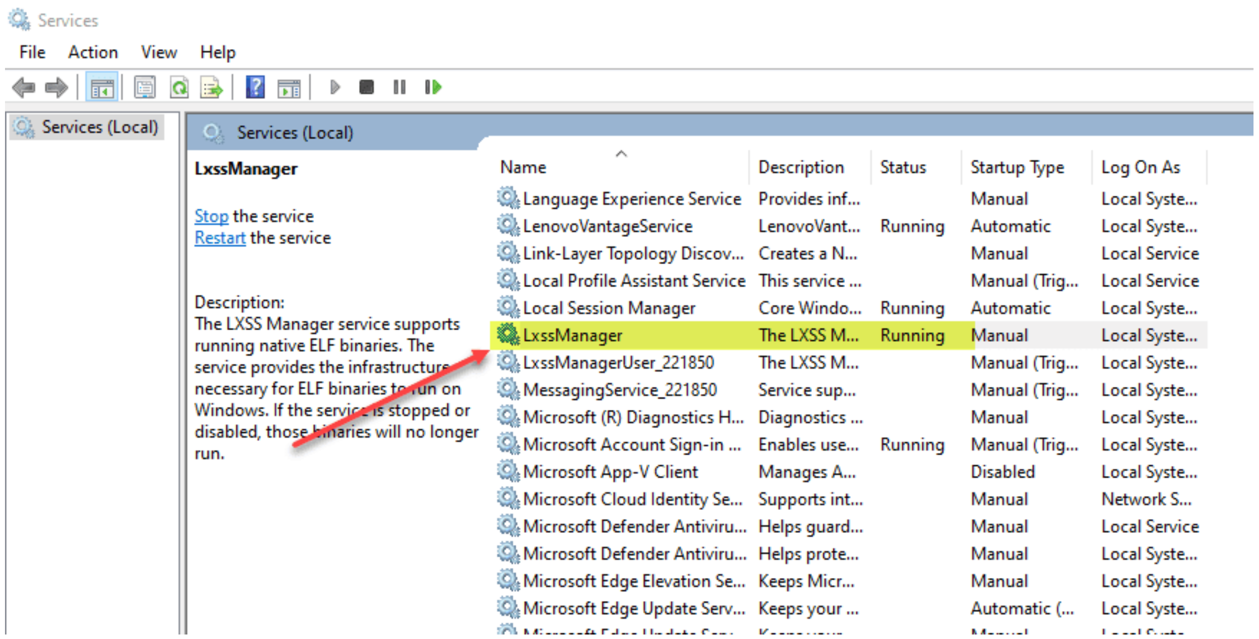
There are a couple of prerequisites needed for installation of Minikube in WSL 2.

Install systemctl

- Pulldown a script from github and execute using following commands on WSL 2.

```
git clone https://github.com/DamionGans/ubuntu-wsl2-systemd-script.git
cd ubuntu-wsl2-systemd-script/
bash ubuntu-wsl2-systemd-script.sh
```

- Restart the **LxssManager** service in Windows to initialize systemctl with WSL 2.



LxssManager

Stop the service
Restart the service

Description:
The LXSS Manager service supports running native ELF binaries. The service provides the infrastructure necessary for ELF binaries to run on Windows. If the service is stopped or disabled, those binaries will no longer run.

Name	Description	Status	Startup Type	Log On As
Language Experience Service	Provides inf...		Manual	Local Syste...
LenovoVantageService	LenovoVant...	Running	Automatic	Local Syste...
Link-Layer Topology Discov...	Creates a N...		Manual	Local Service
Local Profile Assistant Service	This service ...		Manual (Trig...	Local Service
Local Session Manager	Core Windo...	Running	Automatic	Local Syste...
LxssManager	The LXSS M...	Running	Manual	Local Syste...
LxssManagerUser_221850	The LXSS M...		Manual (Trig...	Local Syste...
MessagingService_221850	Service sup...		Manual (Trig...	Local Syste...
Microsoft (R) Diagnostics H...	Diagnostics ...		Manual	Local Syste...
Microsoft Account Sign-in ...	Enables use...	Running	Manual (Trig...	Local Syste...
Microsoft App-V Client	Manages A...		Disabled	Local Syste...
Microsoft Cloud Identity Se...	Supports int...		Manual	Network S...
Microsoft Defender Antiviru...	Helps guard...		Manual	Local Service
Microsoft Defender Antiviru...	Helps prote...		Manual	Local Syste...
Microsoft Edge Elevation Se...	Keeps Micr...		Manual	Local Syste...
Microsoft Edge Update Serv...	Keeps your ...		Automatic (...	Local Syste...

Install Conntrack

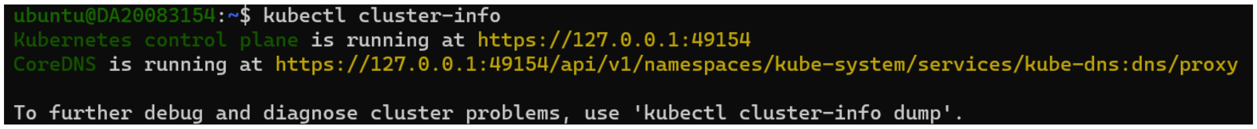
```
sudo apt install -y conntrack
```

Install Kubectl and Minikube and start the cluster

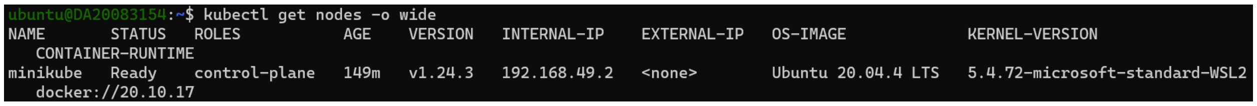
Kubectl is the defacto standard tool for working with Kubernetes in general. It is great to get it installed and set the context for Kubectl to Minikube.

```
# Download the Google Cloud public signing key
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg

# Add the Kubernetes apt repository
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list

# Update apt package index with the new repository
sudo apt-get update

# Install kubectl
sudo apt-get install -y kubectl

# Enable kubectl autocompletion Bash
echo 'source <(kubectl completion bash)' >>~/.bashrc
kubectl completion bash | sudo tee /etc/bash_completion.d/kubectl > /dev/null

# Install the latest Minikube stable release on x86-64 Linux using binary
download
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-
linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube

# Set VM driver to none, as we cannot virtualize on virtualization
sudo minikube config set vm-driver none

# Delete all existing Minikube profiles
sudo minikube delete --purge=true --all=true

# Start Minikube
minikube start --driver=docker --delete-on-failure

# Set Kubectl context to  Minikube
kubectl config use-context minikube

# Access your shiny new cluster
kubectl get po -A

# Check for cluster info on your localhost
kubectl cluster-info
```

Below cluster-info should be shown after successful installation & configuration.



```
ubuntu@DA20083154:~$ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:49154
CoreDNS is running at https://127.0.0.1:49154/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```



```
ubuntu@DA20083154:~$ kubectl get nodes -o wide
NAME       STATUS   ROLES           AGE    VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION
   CONTAINER-RUNTIME
minikube   Ready    control-plane   149m   v1.24.3   192.168.49.2   <none>        Ubuntu 20.04.4 LTS   5.4.72-microsoft-standard-WSL2
   docker://20.10.17
```

Install Helm to work with Minikube

The process to install helm involves the following steps.

```
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh
```

Minikube Dashboard

Dashboard

minikube has integrated support for the Kubernetes Dashboard UI.

Overview

The Dashboard is a web-based Kubernetes user interface. You can use it to:

- deploy containerized applications to a Kubernetes cluster
- troubleshoot your containerized application
- manage the cluster resources
- get an overview of applications running on your cluster
- creating or modifying individual Kubernetes resources (such as Deployments, Jobs, DaemonSets, etc)

For example, you can scale a Deployment, initiate a rolling update, restart a pod or deploy new applications using a deploy wizard.

Basic usage

To access the dashboard:

```
minikube dashboard
```

This will enable the dashboard add-on, and open the proxy in the default web browser.

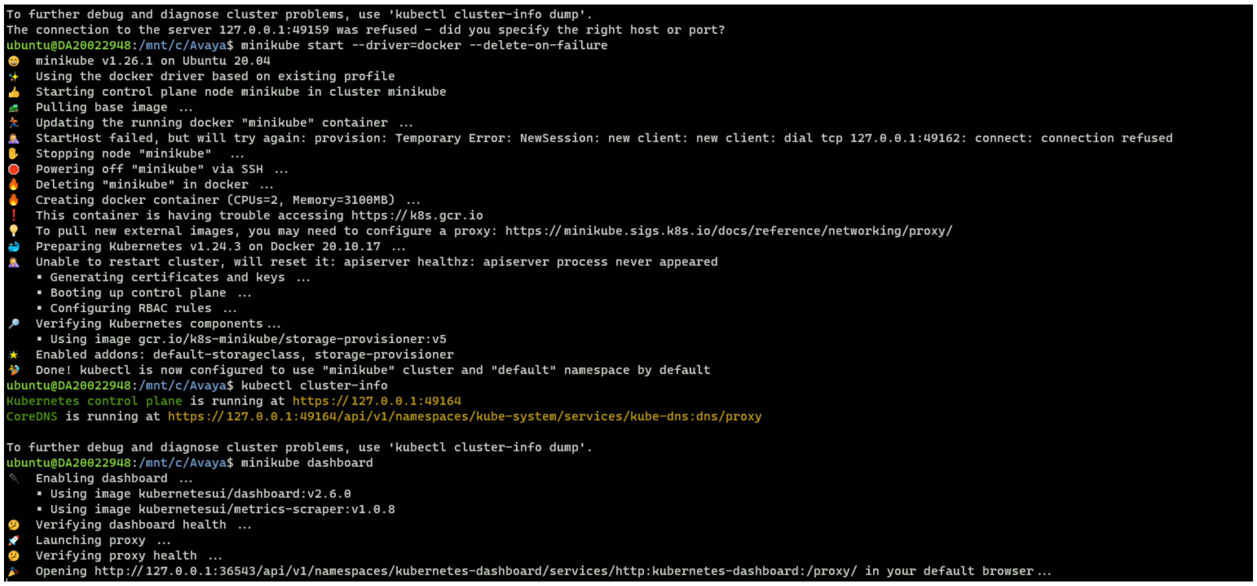
It's worth noting that web browsers generally do not run properly as the root user, so if you are in an environment where you are running as root, see the URL-only option.

To stop the proxy (leaves the dashboard running), abort the started process (`Ctrl+C`).

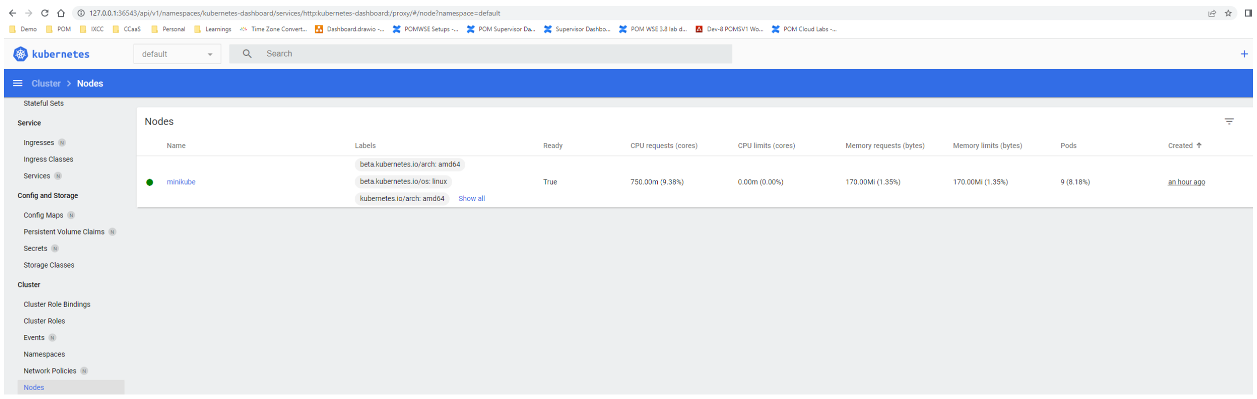
Getting just the dashboard URL

If you don't want to open a web browser, the dashboard command can also simply emit a URL:

```
minikube dashboard --url
```



```
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
The connection to the server 127.0.0.1:49159 was refused - did you specify the right host or port?
ubuntu@DA20022948:/mnt/c/Avaya$ minikube start --driver=docker --delete-on-failure
    minikube v1.26.1 on Ubuntu 20.04
    Using the docker driver based on existing profile
    Starting control plane node minikube in cluster minikube
    Pulling base image ...
    Updating the running docker "minikube" container ...
    StartHost failed, but will try again: provision: Temporary Error: NewSession: new client: new client: dial tcp 127.0.0.1:49162: connect: connection refused
    Stopping node "minikube"  ...
    Powering off "minikube" via SSH ...
    Deleting "minikube" in docker ...
    Creating docker container (CPUs=2, Memory=3100MB) ...
    This container is having trouble accessing https://k8s.gcr.io
    To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
    Preparing Kubernetes v1.24.3 on Docker 20.10.17 ...
    Unable to restart cluster, will reset it: apiserver healthz: apiserver process never appeared
      ▪ Generating certificates and keys ...
      ▪ Booting up control plane ...
      ▪ Configuring RBAC rules ...
    Verifying Kubernetes components...
      ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    Enabled addons: default-storageclass, storage-provisioner
    Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
ubuntu@DA20022948:/mnt/c/Avaya$ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:49164
CoreDNS is running at https://127.0.0.1:49164/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
ubuntu@DA20022948:/mnt/c/Avaya$ minikube dashboard
    Enabling dashboard ...
      ▪ Using image kubernetesui/dashboard:v2.6.0
      ▪ Using image kubernetesui/metrics-scraper:v1.0.8
    Verifying dashboard health ...
    Launching proxy ...
    Verifying proxy health ...
    Opening http://127.0.0.1:36543/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
```



127.0.0.1:36543/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/node?namespace=default

Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Pods	Created ↑
● minikube	beta.kubernetes.io/arch: amd64 beta.kubernetes.io/os: linux kubernetes.io/arch: amd64 Show all	True	750.00m (9.38%)	0.00m (0.00%)	170.00Mi (1.35%)	170.00Mi (1.35%)	9 (8.18%)	an hour ago

Summary

Using WSL 2 and Minikube is a great way to start playing around with Kubernetes clusters without the need for standing up VMs and other lab environment prerequisites. WSL2 has its quirks, however, using the process listed above, you should be able to get a Minikube Kubernetes cluster up and running fairly quickly.

Kubectl and Helm are additional components that aren’t absolutely required to work with Minikube. However, they allow using the same tools you will use in production to interact with your Kubernetes clusters.