# Networking

## Create a testing Pod

Create a Pod for testing networking. We can use for that for example the busybox – publicly available Docker image.

We can use this command to create a Pod, get access to its terminal and remove it once we exit its terminal:

* kubectl run dns-test --image=busybox:1.28 --restart=Never -it --rm -- /bin/sh

From the Pod’s terminal execute commands for testing networking.

## CoreDNS

Create a testing Pod using this command:

* kubectl run -rm -it <pod-name> --image=your-docker-image -- /bin/bash

As image we can use for example the busybox:1.28 or our own one.

From that Pod’s terminal we can test resolving DNS names:

* nslookup kubernetes.default – DNS name for API server
* nslookup google.com – check a public domain
* nslookup <service-name>.<namespace> - domain name for some Kubernetes Service

Other things to check:

* Check CoreDNS logs.
* Check firewall rules using iptables
* Check CoreDNS configMap, especially section with the ‘forward’ keyword
* Check the resolv.conf file of a Pod.
* After making changes to CoreDNS restart a coredns deployment:
  + kubectl rollout restart deployment coredns -n kube-system

# Testing a Docker image

If we want to test how a Docker image behaves in a Pod, we can quickly create a testing Pod using that image:

* kubectl run -rm -it <pod-name> --image=your-docker-image -- /bin/bash

After creating it we can get access to its terminal and execute some bash commands or we can check its logs.

More information about how to create such a Pod is in the ‘Kubernetes useful commands’ file in the ‘Creating a Pod’ section.

# Check resource specification

We can check a YAML specification for a given resource using the ‘kubectl get …. -o yaml’ command.

More information about that command is in the ‘Kubernetes useful commands’ file.

# Check logs

We can check resources logs using the ‘kubectl logs’ and ‘kubectl describe’ commands.

More information about those commands is in the ‘Kubernetes useful commands’ file.

# Setting up Kubernetes

Here are usefull tools for debugging.

* crictl ps -a – list the running and stopped containers.
* Crictl logs <containerID> - check logs for a given container
* ss -tuln | grep 6443 – check processes listening on the 6443 port
* ps aux | grep kube-apiserver – check processes in which the ‘kube-apiserver’ string appears.
* journalctl -u kubelet -f – view logs from the kubelet process
* kubectl get pods -n kube-system – Check pods in the kube-system namespace
* telnet 10.0.1.4 6443 – try to connect over TCP to the server with the 10.0.1.4 IP address over the 6443 port.