**Basic Skupper Setup**

**Summary:** Allow users to take current Mac setup and initialize a skupper installation connecting two separate namespaces allowing for communication between private docker instances. By setting up this infrastructure users will be able to set up their kubernetes environment to communicate in a namespace agnostic ecosystem.

**Prerequisite:** MacOSX, Docker, Minikube, Kind, docker-mac-net-connect

**Log**

| Change Made | Signature |
| --- | --- |
| Created Initial Document | AG, SP |
|  |  |
|  |  |
|  |  |

brew install go

go install sigs.k8s.io/kind@v0.18.0

brew install kind

Create kind cluster

Initialize and set-up kind cluster

| > kind create cluster  **Output >**  Creating cluster "kind" ...  ✓ Ensuring node image (kindest/node:v1.25.3) 🖼  ✓ Preparing nodes 📦  ✓ Writing configuration 📜  ✓ Starting control-plane 🕹️  ✓ Installing CNI 🔌  ✓ Installing StorageClass 💾  Set kubectl context to "kind-kind"  You can now use your cluster with:  kubectl cluster-info --context kind-kind  Thanks for using kind! 😊 |
| --- |

Kind install verification

Status checking - kind installation

| > kubectl get pods **Output >** No resources found in default namespace. |
| --- |

Create kubernetes config files

This command will allow us to save these configurations for use at a later time

| > export KUBECONFIG=~/.kube/config-ns1  > kind export kubeconfig  > export KUBECONFIG=~/.kube/config-ns2  > kind export kubeconfig |
| --- |

Set-up Load Balancer

loadbalancer needed to get Docker instances to generate IP addresses for use in the skupper install

| > kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.12.1/manifests/namespace.yaml  > kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.12.1/manifests/metallb.yaml  > kubectl get pods -n metallb-system  **Output >**  NAME READY STATUS RESTARTS AGE  controller-6658b8446c-zqdjt 1/1 Running 0 34s  speaker-8f495 1/1 Running 0 34s |
| --- |

Docker network inspect

The next step will involve getting the docker network interface and grabbing the correct range of IPs that we will need for spawning the skupper install. In this case, the CIDR notation we want to note down is 172.19.0.0/16

| > docker network inspect -f '{{.IPAM.Config}}' kind **Output >** [{172.19.0.0/16 172.19.0.1 map[]} {fc00:f853:ccd:e793::/64 map[]}] |
| --- |

Create yaml file

Using that CIDR range we got earlier, we want to set up the metal configuration file for generating the IPs. Below will be the 'metal.yaml' file you should create and modify the IP range at the bottom to what works in your system.

| > nano metal.yaml  apiVersion: v1  kind: ConfigMap  metadata:  namespace: metallb-system  name: config  data:  config: |  address-pools:  - name: default  protocol: layer2  addresses:  - 172.19.255.200-172.19.255.250 |
| --- |

Apply yaml file

| > kubectl apply -f metal.yaml **Output >** configmap/config created |
| --- |

Clone the hello world skupper example from their github repo:

| > git clone https://github.com/GemaSoftware/Skupper-Hello-World-Mac |
| --- |

Connect namespaces

Once in the folder, run the test application that will spawn the frontend and backend where a skupper install lives connecting the two namespaces

| > ./plano run-external ~/.kube/config-west ~/.kube/config-east |
| --- |

Final Connection

Once running, find the IP of the frontend docker instance by running this command and grabbing the <External-IP> located with this docker container.  
Attempt to curl to that IP address on port 8080. If this command runs without error, you will see a Hello World meanign the Skupper namespace connection worked.

| > kubectl get svc  **>** curl http://172.19.255.204:8080 **Output >** <!DOCTYPE html>  <html>  <head>  <title>Skupper Hello World</title>  <link rel="icon" href="data:,">  <link rel="stylesheet" href="/static/main.css"/>  <script type="module" src="/static/main.js"></script>  </head>  <body id="content">  </body>  </html> |
| --- |