Configuring Network Policies for Applications

Network Policies allow you to limit connections between Pods. Therefore, using Network Policies provide better security by reducing the compromise radius.

> ***Note*** *Network Policies determine whether a connection is allowed, and they do not offer higher level features like authorization or secure transport (like SSL/TLS).*

Create a cluster

To create a container cluster with Network Policy enforcement, run the following command:

```
gcloud container clusters create admatic-network-test --enable-network-policy
```

> ***Warning*** *If you omit the `--enable-network-policy` flag, any NetworkPolicy resources you create are silently ignored.*

```
WARNING: Starting in 1.12, new clusters will have basic authentication disabled by d
efault. Basic authentication can be enabled (or disabled) manually using the `--[no-
]enable-basic-auth` flag.
WARNING: Starting in 1.12, new clusters will not have a client certificate issued. Y
ou can manually enable (or disable) the issuance of the client certificate using the
 `--[no-]issue-client-certificate` flag.
WARNING: Currently VPC-native is not the default mode during cluster creation. In th
e future, this will become the default mode and can be disabled using `--no-enable-i
p-alias` flag. Use `--[no-]enable-ip-alias` flag to suppress this warning.
WARNING: Starting in 1.12, default node pools in new clusters will have their legacy
 Compute Engine instance metadata endpoints disabled by default. To create a cluster
 with legacy instance metadata endpoints disabled in the default node pool, run `clu
sters create` with the flag `--metadata disable-legacy-endpoints=true`.
This will enable the autorepair feature for nodes. Please see https://cloud.google.c
om/kubernetes-engine/docs/node-auto-repair for more information on node autorepairs.
WARNING: Starting in Kubernetes v1.10, new clusters will no longer get compute-rw an
d storage-ro scopes added to what is specified in --scopes (though the latter will r
emain included in the default --scopes). To use these scopes, add them explicitly to
 --scopes. To use the new behavior, set container/new_scopes_behavior property (gclo
ud config set container/new_scopes_behavior true).
Creating cluster admatic-network-test in us-west1-c...done.
Created [https://container.googleapis.com/v1/projects/espblufi-android/zones/us-west
1-c/clusters/admatic-network-test].
```

```
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kub
ernetes/workload_/gcloud/us-west1-c/admatic-network-test?project=espblufi-android
kubeconfig entry generated for admatic-network-test.
NAME                  LOCATION    MASTER_VERSION  MASTER_IP       MACHINE_TYPE   NOD
E_VERSION  NUM_NODES  STATUS
admatic-network-test  us-west1-c  1.10.9-gke.5    35.233.184.239  n1-standard-1  1.1
0.9-gke.5  3          RUNNING
```

Restrict incoming traffic to Pods

Kubernetes `NetworkPolicy` resources let you configure network access policies for the Pods. `NetworkPolicy` objects contains with the following information:

- Pods to which the network policies apply, usually designated by a label selector
- Type of Internet traffic the network policy affects: `Ingress` for incoming traffic, `Egress` for outgoing traffic, or both
- For Ingress policies, which Pods can connect to the specified Pods
- For Egress policies, the Pods to which the specified Pods can connect.

First, run a simple web server application with label `app=hello` and expose it internally in the cluster:

```
kubectl run hello-web --labels app=hello \
  --image=gcr.io/google-samples/hello-app:1.0 --port 8080 --expose
```

```
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a fu
ture version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
service/hello-web created
deployment.apps/hello-web created
```

Next, you need to configure a `NetworkPolicy` to allow traffic to the `hello-web` Pods from only the `app=foo` Pods. Other incoming traffic from Pods that do not have this label, external traffic, and traffic from Pods in other namespaces are blocked.

```
cat << EOF > hello-allow-from-foo.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: hello-allow-from-foo
spec:
  policyTypes:
    - Ingress
  podSelector:
    matchLabels:
      app: hello
  ingress:
```

```
    - from:
        - podSelector:
            matchLabels:
              app: foo
EOF
```

This policy selects Pods with label `app=hello` and specifies an ingress policy to allow traffic only from Pods with the label `app=foo`.

To apply this policy to the cluster, run the following command:

```
kubectl apply -f hello-allow-from-foo.yaml
```

```
networkpolicy.networking.k8s.io/hello-allow-from-foo created
```

Validate the ingress policy

First, run a temporary Pod with the label `app=foo` and get a shell in the Pod:

```
kubectl run -l app=foo --image=alpine --restart=Never --rm -i -t test-1
```

```
If you don't see a command prompt, try pressing enter.
/ #
```

Make a request to the `hello-web:8080` endpoint to verify that the incoming traffic is allowed:

```
wget -qO- --timeout=2 http://hello-web:8080
```

```
Hello, world!
Version: 1.0.0
Hostname: hello-web-8b44b849-9c2vl
```

Traffic from Pod `app=foo` to the `app=hello` Pods is enabled.

Next, run a temporary Pod with a different label (`app=other`) and get a shell inside the Pod:

```
kubectl run -l app=other --image=alpine --restart=Never --rm -i -t test-1
```

```
If you don't see a command prompt, try pressing enter.
/ #
```

Make the same request to observe that the traffic is **not allowed** and therefore the request times out, then exit from the Pod shell:

```
wget -qO- --timeout=2 http://hello-web:8080
```

```
wget: download timed out
```

```
exit
```

```
pod "test-1" deleted
pod default/test-1 terminated (Error)
```

Restrict outgoing traffic from the Pods

You can restrict outgoing (egress) traffic as you would incoming traffic.

However, to be able to query internal hostnames such as `hello-web` or external hostnames such as `www.example.com`, you must allow DNS (domain name system) resolution in your egress network policies. DNS traffic occurs on port 53 using TCP and UDP protocols.

To exercise egress network policies, deploy a `NetworkPolicy` controlling outbound traffic from Pods with the label `app=foo` while allowing traffic only to Pods with the label `app=hello`, as well as the DNS traffic.

```
cat << EOF > foo-allow-to-hello.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: foo-allow-to-hello
spec:
  policyTypes:
    - Egress
  podSelector:
    matchLabels:
      app: foo
  egress:
    - to:
        - podSelector:
            matchLabels:
              app: hello
    - ports:
        - port: 53
          protocol: TCP
        - port: 53
          protocol: UDP
EOF

kubectl apply -f foo-allow-to-hello.yaml
```

```
networkpolicy.networking.k8s.io/foo-allow-to-hello created
```

This manifest specifies a network policy controlling the egress traffic from Pods with label `app=foo` with two allowed destinations:

1. Pods in the same namespace with the label app=hello.
2. Cluster Pods or external endpoints on port 53 (UDP and TCP)

Validate the egress policy

First, deploy a new web application called `hello-web-2` and expose it internally in the cluster:

```
kubectl run hello-web-2 --labels app=hello-2 \
  --image=gcr.io/google-samples/hello-app:1.0 --port 8080 --expose
```

```
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a fu
ture version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
service/hello-web-2 created
deployment.apps/hello-web-2 created
```

Next, run a temporary Pod with `app=foo` label and get a shell prompt inside the container:

```
kubectl run -l app=foo --image=alpine --rm -i -t --restart=Never test-3
```

```
If you don't see a command prompt, try pressing enter.
/ #
```

Validate that the Pod can establish connections to `hello-web:8080`:

```
wget -qO- --timeout=2 http://hello-web:8080
```

```
Hello, world!
Version: 1.0.0
Hostname: hello-web-8b44b849-9c2vl
```

Validate that the Pod cannot establish connections to `hello-web-2:8080`:

```
wget -qO- --timeout=2 http://hello-web-2:8080
```

```
wget: download timed out
```

```
exit
```

```
pod "test-3" deleted
pod default/test-3 terminated (Error)
```

Cleaning up

Delete the container cluster

This step will delete the resources that make up the container cluster, such as the compute instances, disks and network resources.

```
gcloud container clusters delete admatic-network-test
```

```
The following clusters will be deleted.
 - [admatic-network-test] in [us-west1-c]

Do you want to continue (Y/n)?  Y

Deleting cluster admatic-network-test...done.
Deleted [https://container.googleapis.com/v1/projects/espblufi-android/zones/us-west1-c/clusters/admatic-network-test].
```