Network Policies implementation in FSCP 3.0

Last updated by | William Artero | Sep 14, 2023 at 10:49 AM GMT+2

⚠️ AAB alert

AKS clusters are imposed a default `deny-all` global network policy.
Create higher priority network policies to allow pods to communicate with each other.

Contents

About Network Policy

Network policies are a way to control the network traffic between pods in the cluster. By defining network policies we can enforce security, isolation and prevent unauthorized access to pods.

The page linked in the header has a lot of information which you should read. Below are some of the most important points to know when it comes to Network Policy on Kubernetes.

Types of Network Policy API

There are two types of API you can use to define your network policies.

- kubernetes native network policy API (`networking.k8s.io`)
- Calico CRD API (`projectcalico.org/v3`)

Stratus Recommendation

- Use the native **kubernetes API** (`networking.k8s.io/v1`) for defining your network policies

- For more complex edge cases, you can use calico CRD API (`projectcalico.org/v3`)

---

networking.k8s.io/v1 - NetworkPolicy

This is the API defined as part of Kubernetes itself. This standard API is limited in its functionality when compared with alternatives, although this does mean that Network Policies are portable between Kubernetes clusters of different distributions or CSPs.

This API will not operate without a network policy plugin. Kubernetes itself does not ship with a mechanism for creating or enforcing any kind of network policy resource; it simply defines a standardized API for doing so.

Summary:

- a provider-agnostic API defined as part of Kubernetes itself no need of any add-ons.
- Simple and easy to use
- Namespace-scoped
- limited in capabilities
- deployed via `kubectl`

Ways to deploy a Kubernetes API(`networking.k8s.io/v1`) NetworkPolicy

This is trivial; you simply need to run `kubectl apply -f policy.yaml`

where `policy.yaml` is a file containing a NetworkPolicy resource of the `networking.k8s.io/v1` API.

You can find examples on how to implement common scenarios in the FSCP Azure Community repository. Check the official Kubernetes documentation for more information about the network policies API and details on how they work.

Projectcalico.org/v3 - NetworkPolicy

This is the API provided by Calico. Calico is the Network Policy engine created by Tigera. Calico is made available by Azure for use on AKS and will be automatically installed on your cluster when deployed.

Calico is a 3rd party tool which is not created or maintained by the Kubernetes project. However, Calico does provide the required infrastructure to facilitate both the Kubernetes Network Policy API **and** a proprietary, more sophisticated Calico Network Policy API.

This means that once Calico is installed, both the standardized Kubernetes API and the Calico API will be usable. That does not mean that you should use both, however; it's recommended to use only one where possible.

Summary:

- Calico-specific API defined by Calico
- Global and Namespace-scoped resources available
- more functionality than Kubernetes-only
- deployed via `calicoctl` or `kubectl` + Calico API Server

Ways to deploy a Calico API NetworkPolicy

Calico provides two ways to deploy NetworkPolicy resources which use the Calico API (i.e. `projectcalico.org/v3`):

Use `calicoctl` directly

This is the most straightforward method to set up. You simply need to ensure you have the `calicoctl` binary available wherever you want to run it (e.g. on your laptop or build agent).

You then use the `calicoctl` command to manipulate relevant Calico YAML manifests or cluster resources as you wish.

Use `kubectl` with the Calico API Server

This is the most pleasant and natural way to use Calico, but also requires some extra setup. For this, you need to deploy Calico API Server using the Calico Operator. This causes a Pod to be spun up in a `calico-apiserver` Namespace; you will also need to ensure that) this Pod has the required access to the Kubernetes API Server.

Once done, you will be able to do things like `kubectl apply -f policy.yaml` with files that directly contain a `projectcalico.org/v3` manifest, rather then using `calicoctl`.

For more information on Calico API network policy refer Calico official documentation

---