Aporeto

Azure Integration Guide

Cloud-Native Security

www.aporeto.com

Securing Kubernetes Workloads in Hybrid Settings with Aporeto

Centralized Visibility and Security for Applications Distributed on AKS and Private Clouds

Aporeto

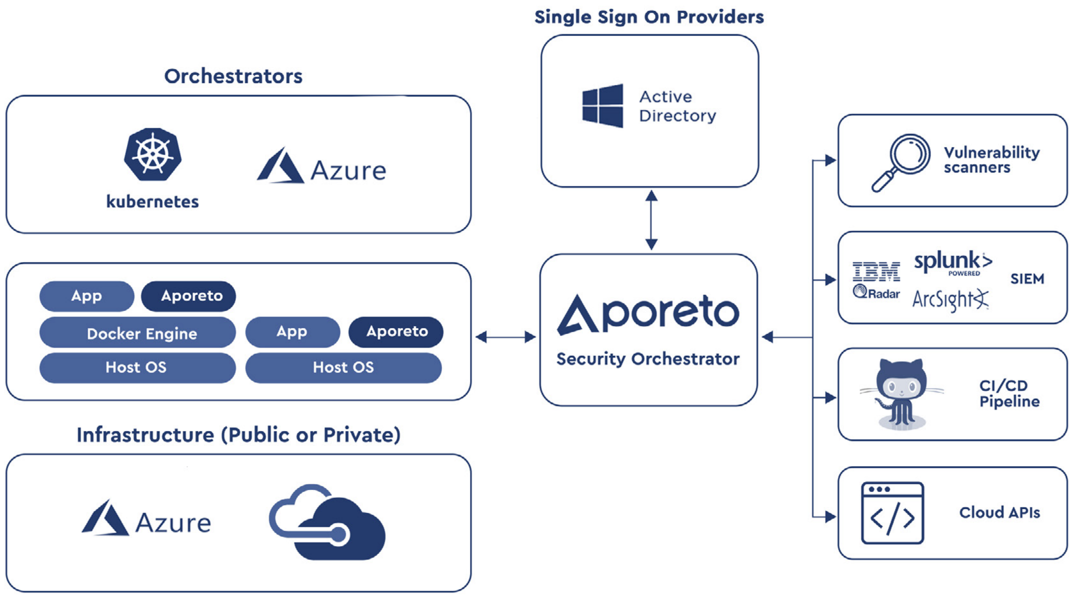
Azure Kubernetes Service (AKS) makes it simple to deploy a managed Kubernetes cluster in Azure. AKS reduces the complexity and operational overhead of managing Kubernetes by off loading much of that responsibility to Azure and handling critical tasks like health monitoring and maintenance. However, your operational needs may require you to deploy your Kubernetes cluster in a hybrid setting. For instance, your data services may be running in your private cloud while application logic services could be running in AKS.

Without the proper toolset and configuration, maintaining visibility and security for your distributed environment can be hard to configure and maintain. In this whitepaper, we focus on providing centralized visibility and monitoring for these types of distributed workloads in a manner that is easy to deploy and manage.

About Aporeto

Aporeto is a Zero Trust security solution for microservices, containers and cloud. Fundamental to Aporeto's approach is the principle that everything in an application is accessible to everyone and could be compromised at any time. Aporeto uses vulnerability data, identity context, threat monitoring and behavior analysis to build and enforce authentication, authorization and encryption policies for applications. With Aporeto, enterprises implement a uniform security policy decoupled from the underlying infrastructure, enabling workload isolation, API access control and application identity management across public, private or hybrid cloud.

Because Aporeto transparently binds to application components to provide them with identity, the result is security independent from infrastructure and network and reduction of complexity at any scale on any cloud.



Aporeto is simple to deploy and operate:

- **Pick an application and visualize it;**
- **Generate and simulate security policy;**
- **Enforce the security policy.**

You can visualize the application of your choice by deploying Aporeto as an AKS DaemonSet. If you control the virtual machines on which your application component run, you may also deploy Aporeto as a Docker container or a userland process.

Aporeto auto-generates application security policy by ingesting Kubernetes Network Policies. You also have the option of leveraging your application dependency graph that Aporeto creates to describe the application's behavioral intent as policies. In every case, you may audit and edit auto-generated policies and inject human wisdom when necessary.

Once you have policies, you may simulate their enforcement at runtime to evaluate the effects of your security policies without interrupting operations. When satisfied that your security policies are solid, you may lockdown your application and protect it with a Zero Trust approach.

Because Aporeto untethers application security from the network and infrastructure, one key benefit of Aporeto's approach for protecting your containers, microservices and cloud applications is that you can have a consistent security approach even in a hybrid or multi-cloud setting. As you gain experience with Aporeto in a single cluster setting, you will quickly realize how easy it is to have a consistent security posture in multi-cluster and multi-cloud settings without any infrastructure or operational complexity.

Setting up Aporeto in AKS

STEP 1 Prepare environment

You will need the following binaries installed in your path.

1. az (see https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest)
2. kubectl (see https://kubernetes.io/docs/tasks/tools/install-kubectl/)

If you have not already done so, log into azure with

```
az login
```

and create a working directory

```
mkdir -p aks; cd aks
```

The assumption will be made that you are in the working directory for the remainder of this document.

STEP 2 Aporeto Setup

You will need to login to Aporeto and create the Kubernetes cluster. This can be done from the Web UI or by command line.

Using the UI (NOT THE CLI)

Using the UI login to https://console.aporeto.com/ then click on Switch to Accounts (top right corner user icon, immediate right of the "?" mark icon. On the navigation pane click on Kubernetes Clusters and then click on the "+" sign icon (top right). Give the cluster a name such as aks1 and select a namespace to create the cluster under. Finally click on the create button.

This will create the cluster and cause a gzipped tar file to be downloaded. The naming format is cluster-name.tar.gz (ssuming you named the cluster aks1 the file will be named aks1.tar.gz) to be downloaded to your default download directory. Move this file to the working directory created in the previous step.

Using CLI (NOT THE UI)

You must have a valid Aporeto token set in your environment. We will assume you have this set as APOCTL_TOKEN. You will also need to set TARGET_NS to your desired namespace. Run the commands or use this script:

```
#APOCTL_TOKEN= (already exported into environment)
TARGET_NS="/YOUR_NAMESPACE_GOES_HERE"
CLUSTER_NAME="aks"

curl -o cluster.json 'https://api.console.aporeto.com/kubernetesclusters?' \
-H 'Accept-Encoding: gzip,deflate' \

-H "Authorization: Bearer $APOCTL_TOKEN" \
-H 'Content-Type: application/json' \
--data-binary "{\"name\":\"$CLUSTER_NAME\",\"targetNamespace\":\"$TARGET_NS\"}"\
--compressed
```

This command will download the result in the JSON format. The JSON file should contain a key with the name kubernetes Definitions. The value is a base64 encoded gzip tar file. We can extract the field into a file with the command.

```
cat cluster.json | awk -F"kubernetesDefinitions" '{print $2}' | awk -F\" '{print $3}'
> cluster.base64
```

Then we need to convert the base64 file to a gzip tar file. On Mac we can do this with the command

```
cat cluster.base64 | base64 -D -o ${CLUSTER_NAME}.tar.gz
```

and on Linux we can use the command

```
cat cluster.base64 | base64 -d > ${CLUSTER_NAME}.tar.gz
```

You should now have a valid gzip file.

STEP 3 Create AKS (Kubernetes Cluster on AKS)

Run the following commands:

```
az group create --name aporeto_lab --location eastus

az aks create --resource-group aporeto_lab --name aks1 \
   --node-count 2 --generate-ssh-keys

az aks get-credentials --resource-group aporeto_lab \
   --name aks1 -f aks1.kube.cfg

export KUBECONFIG=$PWD/aks1.kube.cfg
```

Verify that the nodes are operational with the command:

```
kubectl get nodes
```

You should see something like this

```
->;kubectl get nodes
NAME                         STATUS    ROLES    AGE     VERSION
aks-nodepool1-82983338-0     Ready     agent    3m      v1.9.6
aks-nodepool1-82983338-1     Ready     agent    3m      v1.9.6
```

STEP 4 Join the AKS Cluster to Aporeto

Extract the contents of the file aks1.tar.gz and create the Kubernetes resources with these commands:

```
mkdir -p kube_aporeto

tar xfv aks1.tar.gz -C kube_aporeto

kubectl create -f kube_aporeto
```

Then, check the status with this command:

```
kubectl get pods -n kube-system
```

You should see something like the following:

```
->;kubectl get pods -n kube-system
NAME                                    READY    STATUS     RESTARTS     AGE
aporeto-enforcer-fkf46                  1/1      Running    0            23s
aporeto-enforcer-v4k5r                  1/1      Running    0            23s
aporeto-kubesquall-h4m5d                1/1      Running    0            21s
azureproxy-79c5db744-t2654              1/1      Running    2            4m
heapster-55f855b47-drbb2                2/2      Running    0            3m
kube-dns-v20-7c556f89c5-mcg6z           3/3      Running    0            4m
kube-dns-v20-7c556f89c5-xhts7           3/3      Running    0            4m
kube-proxy-h5rqq                        1/1      Running    0            4m
kube-proxy-s7rkq                        1/1      Running    0            4m
```

```
kube-svc-redirect-92tvv                   1/1       Running     0            4m
kube-svc-redirect-h2dmp                   1/1       Running     0            4m
kubernetes-dashboard-546f987686-7gzln     1/1       Running     2            4m
tunnelfront-66fd996c74-dlpdm              1/1       Running     0            4m
```

STEP 5 Enjoy your AKS cluster as secured by Aporeto!

Now that you have connected your AKS Kubernetes cluster to Aporeto, you can visualize it in real time and on historical bases using the Aporeto UI (https://console.aporeto.com/). You can also connect your private cloud workload to your Aporeto account and view your distributed application's end-to-end operations centrally.

You can find instructions for connecting non-AKS workloads to Aporeto by perusing the document set in https://console.aporeto.com/accounts/welcome (click on "Switch to Accounts" (top right corner user icon, immediate right of the "?" mark icon). As always, you can request support directly in Aporeto's Console or via this link.

For more information, visit:
www.aporeto.com