Aporeto

AWS Integration Guide

AWS Integration Guide

Aporeto integrates with AWS to help enterprises efficiently deploy, manage, and secure applications at scale and the compute platforms they run on including Kubernetes, Docker, Linux, Mesos, and others.

Aporeto utilizes the AWS Identity Document – an API-accessible cryptographically signed metadata available on all AWS instances – as contextual identity. This identity is inherited by workloads which can be a process on an EC2 instance or a container/pod running on an EC2 instance. The contextual identity is combined with additional static metadata from the workload.

Aporeto utilizes this workload identity as means to authorize all network communications between workloads within a Virtual Private Cloud (VPC), across VPCs independent of their region or availability zone and across hybrid cloud environments.

Aporeto's identity driven network authorization model has some distinct benefits for AWS users

1. VPCs can have overlapping addresses but the workloads in each VPC will have distinct identity
2. Network access policies work independent of NAT gateways or load balancers allowing very granular protection
3. No changes to AWS security groups when elastic workloads are replicated or re-scheduled across hosts

Feature and Capabilities Description

Feature: EC2 & ECS Instance auto-generated identity

The Aporeto Enforcer runs as a Linux service or container, and protects your applications that run on your compute platforms anywhere, including AWS EC2. For AWS EC2 or EC2 conainer service (ECS) instances, the Enforcer extracts AWS-specific identity attributes in addition to system and user generated attributes.

Figure 1 – Identity for a workload hosted in AWS

Aporeto is able to extract the following AWS specific attributes:

- AWS Region, AWS Availability Zone
- AWS Image ID, AMI launch index
- Local IPv4 address, MAC address, AWS Hostname
- AWS Instance Action, Reservation ID, Version, architecture

Aporeto combines AWS attributes with additional metadata from the the application such as the type of container (nginx, mongo, redis), roles assigned to a container and potential vulnerabilities (CVEs) assigned to the container as part of an image scan. Aporeto offers image vulnerability scanning with continuous vulnerability updates. All of these attributes create a unique fingerprint and define the security posture of this workload.

Benefits: EC2 & ECS Instance Auto-generated Identity

A unique workload identity allows security teams to define network access control policies independent of network infrastructure. This has substantial implications to security teams when deploying dynamic microservices on AWS.

1. No longer have to rely on AWS security groups for workload segmentation. For dynamic workloads maintaining security groups becomes extremely difficult. Automation improves some operational hurdles but in the event of network failure troubleshooting 10s or 100s of security groups is not feasible. When EC2 or ECS workloads replicate to cope with load, access control policies need to be updated if they are tied to IP address. Defining authorization policies based on workload identity allows for seamless workload replication and re-scheduling across hosts without updates to network policies.

2. Factoring in the security posture of an application through its associated vulnerability data – part of its contextual identity – allows security teams to quarantine network communications or be alerted of network communications to/from a vulnerable container. Security teams can assess the impact of a potential exploit and take the appropriate remediation actions.

3. Overlapping IP spaces in VPCs or between VPCs and private cloud is a very common challenge. Defining segmentation policies in these scenarios is done by using the NAT gateway IP address which weakens access control. Leveraging unique workload identity allows granular segmentation independent of IP address and presence of NAT gateways.

4. Most applications in cloud environments use load balancers for scale-out deployments. A load balancer masks workload IP addresses and creates very similar challenges to enabling granular segmentation as the NAT gateway does. Aporeto's approach to workload identity works transparently with L4 and L7 load balancers remediating this issue.

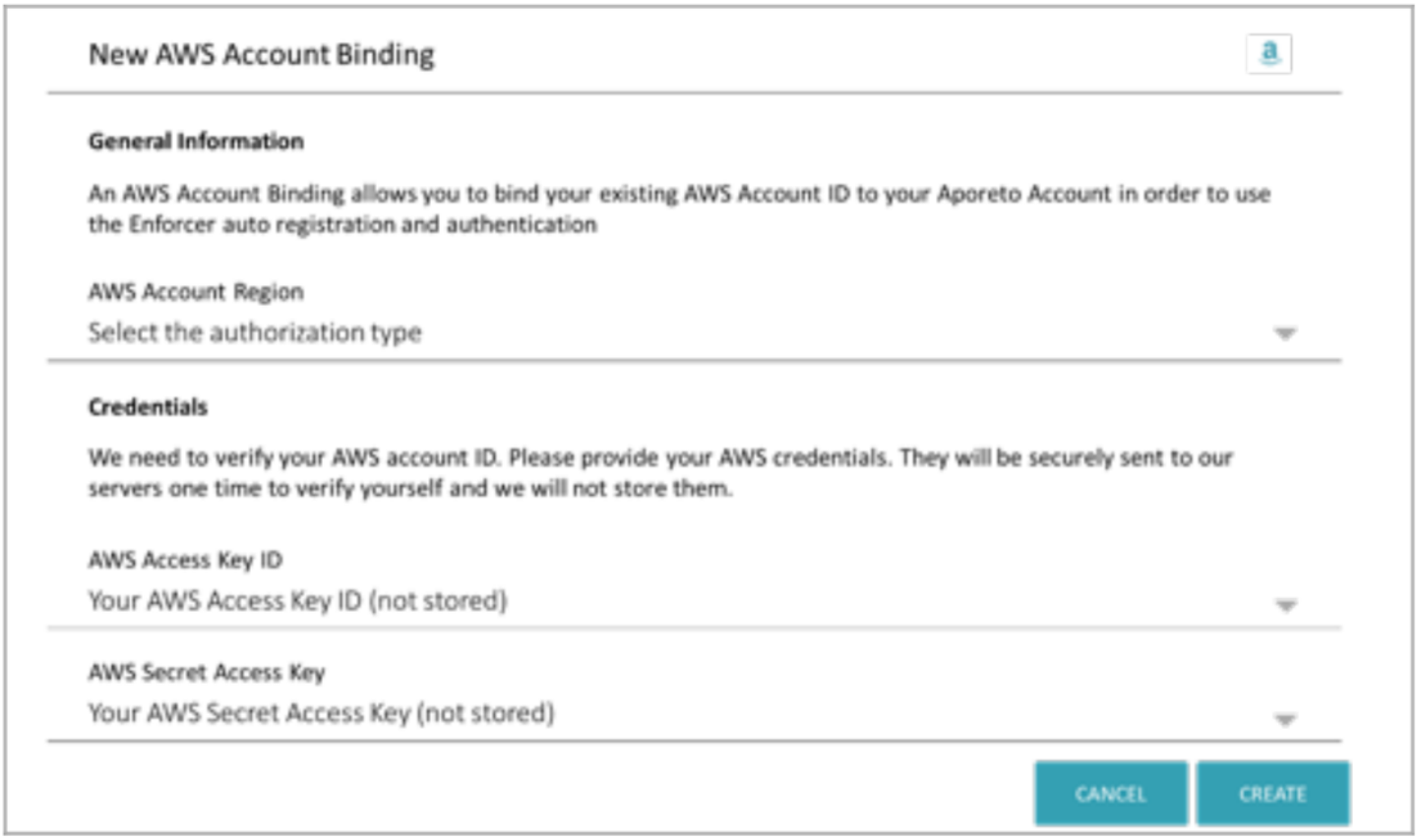
5. Since the identity of an application has AWS specific context security teams are able to create abstract rules to maintain compliance. One example is to ensure any inter AWS region traffic is always encrypted or to ensure any AWS to non AWS cloud traffic is encrypted.

AWS Account Verification

The Aporeto namespace account owner allows the Aporeto Service to verify the the user's AWS account (see Figure 2, below). After specifying the AWS Region, AWS Access Key ID, and AWS Secret Access Key, the Aporeto Service will use those credentials one time to verify the user's AWS account. Aporeto verifies that the AWS account actually belongs to the user.

With the AWS account verified, Aporeto can at a future time, access AWS Instance Identity Documents for identity for applications and their hosting EC2 instances.

Aporeto does not store the AWS keys given to verify the user's AWS account.



Feature: Enforcer Node Auto-registration

An Aporeto-protected environment is composed of two main Aporeto components: the centralized Aporeto Service, and an Enforcer instance on each of the customer compute platform nodes.

Generally, users apply certificates or Aporeto JWT (JSON Web Token) tokens to register a new Enforcer node.

Aporeto Enforcer Node Autoregistration speeds up the Enforcer registration process. When the enforcerd process on the compute platform node (examples include Linux, Docker, Kubernetes) is registered via an "AWS" parameter, enforcerd retrieves the AWS Instance Identity document that is signed by the AWS private key and sends it to the Aporeto Security Orchestrator. Aporeto validates the document is signed correctly using the AWS public key and compares the account ID with with the one provided in the the one-time AWS account verification (described above). If the account IDs match, the instance is accepted and Aporeto issues credentials for the Enforcer to participate in the Aporeto security domain. This process allows Aporeto enforcers to utilize the AWS identity document as a root of trust when attesting the enforcers.

Enforcer Node Autoregistration can be used via an AWS Amazon Machine Image (AMI) that already has the Enforcer as part of the image, or after a virtual machine instance has been created and the Enforcer installed.

Benefits: Enforcer Node Auto-registration

By eliminating the need to use certificates or tokens obtained from the Aporeto Service, Enforcer nodes can be registered with the Aporeto Service in a more easily automated manner (as part of an Infrastructure-as-Code effort), and more securely since no secret data is transmitted over the network.

Feature: AWS External Service Autodiscovery

This feature periodically (30 seconds) polls AWS with the user's credentials to create External Services in Aporeto that correspond to select AWS EC2, ELB, and RDS instances. Aporeto Network access policies can then be created with labels on the AWS resource.

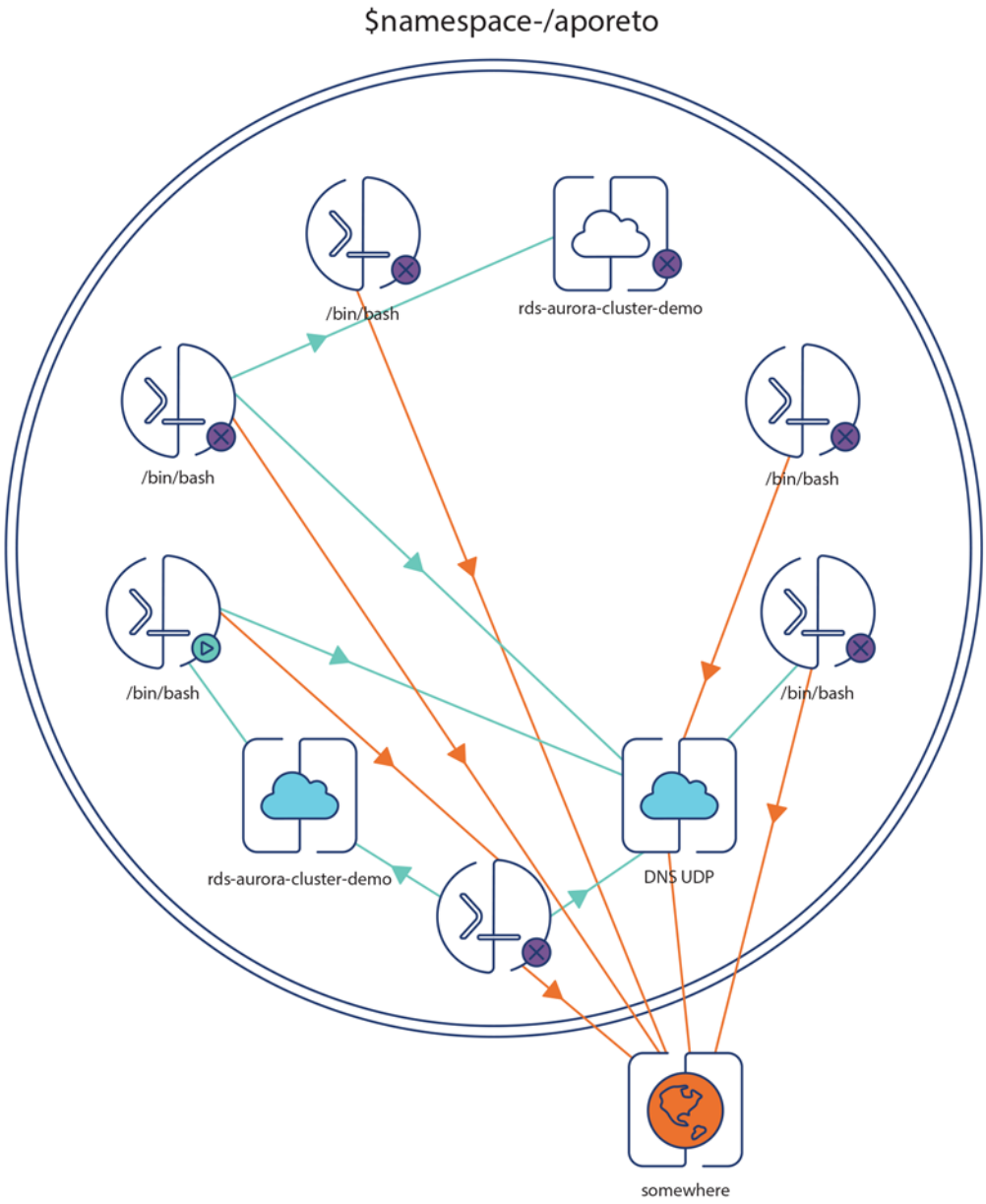


Figure 3 – Zoom in of the Aporeto Platform view depicting an automatically detected AWS EC2 instances and RDS clusters as an External Services, as well as an unauthorized external service

Benefits: AWS External Service Auto-discovery

With AWS EC2, ELB, and RDS clusters and instances registered as External Services, any network communication attempted between those services and services in Aporeto-protected compute platforms can be visualized in the Aporeto Platform view.

Whether for forensics investigation for security attacks, unintended company application network behavior, or reviewing the effects of network policy, the Aporeto Platform view provides visualization for security, network, and developer teams.

AWS Installation Options

Compute platforms (examples include Linux, Docker, and Kubernetes) with the Aporeto Enforcer can be installed in AWS via one of the following methods:

Method:	Amazon Machine Image (AMI) with Enforcer already installed + Enforcer Auto-registration.
Description /what happens:	The AMI is created with Aporeto already installed. /etc/enforcerd/enforcerdenv is part of the image and is pre-populated with the parameters needed to autoregister with the Aporeto Service.
Who this is for /benefits:	This is for customers who want a highly AWS-integrated, Aporeto-transparent installation method. Once the VM is instantiated, the Enforcer auto-registers with the Aporeto service.

Method:	AMI + installation automation + Enforcer Auto-registration.
Description /what happens:	The AMI is created without Aporeto installed. After the VM is instantiated, customer automation installs the Enforcer, and the Enforcer is added to the Service via Enforcer Auto-registration.
Who this is for /benefits:	Some customers may not want a fully, pre-baked AMI – possibly they use multiple cloud providers and want to minimize the amount of cloud-specific workflow.
Option 1:	post-VM bringup automation via AWS cloudinit.
Option 2:	VM installation + post-VM bring-up automation via Terraform (or other tool).

Method:	Manual/custom automation.
Description /what happens:	Per our in-product documentation, manually create an AWS instance, then manually install/register/run the Enforcer.
Who this is for /benefits:	Customers with no/custom automation capability, users who want to/need to manually perform each step of the installation. Parts of this workflow may be automated via various means (e.g. Terraform/Ansible).

Conclusion

Aporeto integration with AWS provides customers fast and secure installation of Aporeto Enforcer nodes.

Aporeto identity integration features provides cloud infrastructure, security, and development teams rich identity for applications services in AWS or any cloud that interacts with AWS hosted services to enforce network policy at scale and ensure secure communication wherever your services reside.