Managing inbound traffic to your application

Document Level Classification

200

Introduction

Allowing inbound network access and traffic to your application (*Pods*) in a Substrate cluster involves the following steps:

1. Create a Kubernetes *Service* object.
2. Create a Kubernetes *Ingress* object (required if your application is internet-facing)
3. Allow inbound network traffic to your *Ingress* resource using Kubernetes Annotations for Service Network Load Balancers or using security group rules for Internet Facing Load Balancers

Creating a Kubernetes *Service*

A Kubernetes *Service* is an abstract way to expose an application running on a set of Pods as a network service. While *Pods* are typically ephemeral (they are created and destroyed as part of deployments and/or scaling), a *Service* is typically not ephemeral.

To create a Kubernetes *Service*:

- If you are using epic-app, use the example below for reference.
- If you are not using epic-app, use this reference to create a manifest.

Example: Define a Service using epic-app

```
epic-app:
  service:
    enabled: true
    type: ClusterIP
    ports:
      - 80
```

A Kubernetes *Service* provides you with a hostname an IP address local to your cluster - for example, `<service-name>.<namespace>.svc.cluster.local`. You can use this hostname to access your application from within the cluster.

Creating a Kubernetes *Ingress* for Internal Access (Service Network)

A Kubernetes *Ingress* object allows you to expose your Kubernetes *Service* outside of the cluster. An *Ingress* may be configured to give *Services* externally reachable URLs, load balance traffic, terminate SSL / TLS, and offer name-based virtual hosting. For Substrate applications, this is the preferred way to expose your application to others over the service-network.

The service network allows services within Epic to be be exposed internally without exposing to the internet, to achieve this the Ingress is configured to be placed on particular service-network subnets within in the VPC.

To create an *Ingress* resource:

- If you are using epic-app, use the example below for reference.
- If you are not using epic-app, use this reference for Ingress resources and this reference for Application Load Balancer annotations to create a manifest.

Example: Define Ingress using epic-app

```
epic-app:
  loadbalancers:
    alb: # `alb` is an arbitrary name. You can use any other meaningful
      hosts:
        # Define the hostname(s) where your services can be reached, an
        # under <account>.internal.epicgames.net is automatically added
        - host: myservice.abcd-dev.internal.epicgames.net
        # Create the certificate in ACM and add it to the load-balanc
      createSSL: true
      port:
        number: 80
      annotations:
        #
        # Reference:
        # * https://kubernetes-sigs.github.io/aws-load-balancer-control
        #

        # Use Application Load Balancer (ALB)
        kubernetes.io/ingress.class: alb

        # The ALB is not reachable via the internet
        alb.ingress.kubernetes.io/scheme: internal

        # Configure listeners on ports 80 and 443, and redirect 80 -> 4
        alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS
        alb.ingress.kubernetes.io/ssl-redirect: "443"

        # Specify how to route traffic to pods
        # This is `ip` since `service.type = ClusterIP`
        alb.ingress.kubernetes.io/target-type: ip

        # Configure health-checks
        alb.ingress.kubernetes.io/healthcheck-path: /health
        alb.ingress.kubernetes.io/success-codes: "200"

        # Place the load balancer on the service-network
        service.epicgames.com/subnet-tag: epic/service-network/k8s-serv

        # Allow traffic from all internal sources
        service.epicgames.com/prefix-list-names: service-network.all-in
```

Manage inbound network traffic to your Internal Application Load Balancer (with Service Network Annotations)

The recommended way to do this for internal load balancers on the Service Network is via the annotations for the Kubernetes Ingress. For a full example of the values, expand the Example in the **Creating a Kubernetes Ingress for internal access** section above. The annotation service.epicgames.com/prefix-list-names: allows you to manage what managed prefix lists have access to your app. For example, if you wanted to allow access to the entire Service Network, you could use the following annotation:

```
service.epicgames.com/prefix-list-names: service-network.all-internal
```

If you wanted to also allow access to office and vpn networks you could use the following annotation:

```
service.epicgames.com/prefix-list-names: service-network.all-internal,s
```

Note: When using multiple prefix lists in the annotation they must be separated by a comma ,.

Create a Kubernetes *Ingress* for Internet Facing access

A Kubernetes *Ingress* object allows you to expose your Kubernetes *Service* outside of the cluster. An *Ingress* may be configured to give

Services externally reachable URLs, load balance traffic, terminate SSL / TLS, and offer name-based virtual hosting. For Substrate applications, this is the preferred way to expose your application to others using an internet-facing URL.

Unlike a *Service*, only creating an *Ingress* resource has no effect, rather an *Ingress controller* is responsible for fulfilling the *Ingress*. Substrate clusters support AWS Load Balancer Controller to satisfy *Ingress* using an Application Load Balancer.

The AWS Load Balancer Controller will provision and configure the Application Load Balancer based on *Ingress* resource configurations and annotations. The AWS Load Balancer Controller is also configured for Subnet Discovery – if you create an *internet-facing* Load Balancer, the controller will automatically find and use the *public* subnets without you having to define subnet ids. However, to use an HTTPS listener, you need a valid SSL certificate provisioned in AWS Certificate Manager (ACM). Follow these instructions to create a certificate in ACM.

To create an *Ingress* resource:

- If you are using epic-app, use the example below for reference.
- If you are not using epic-app, use this reference for Ingress resources and this reference for Application Load Balancer annotations to create a manifest.

Example: Define Ingress using epic-app

```
epic-app:
  loadbalancers:
    alb: # `alb` is an arbitrary name. You can use any other meaningful
      hosts:
        # Define the hostname(s) where your services can be reached
        - host: myservice.abcd.dev.use1a.on.epicgames.com
      port:
        number: 80
      annotations:
```

```
    #
    # Reference:
    # * https://kubernetes-sigs.github.io/aws-load-balancer-control
    #

    # Use Application Load Balancer (ALB)
    kubernetes.io/ingress.class: alb

    # The ALB is reachable via the internet
    alb.ingress.kubernetes.io/scheme: internet-facing

    # Configure listeners on ports 80 and 443, and redirect 80 -> 4
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS
    alb.ingress.kubernetes.io/ssl-redirect: "443"

    # Specify how to route traffic to pods
    # This is `ip` since `service.type = ClusterIP`
    alb.ingress.kubernetes.io/target-type: ip

    # Configure health-checks
    alb.ingress.kubernetes.io/healthcheck-path: /health
    alb.ingress.kubernetes.io/success-codes: "200"

    # Configure ACM certificate used by the TLS listener
    alb.ingress.kubernetes.io/certificate-arn: <FIXME_ACM_CERTIFICA

            # Allowing traffic with Security Groups, Prefix Lists,
            # The following examples are mutually exclusive. You ca
            # with one of these annotation options.

            # Option 1 Security Groups
            # Reference https://kubernetes-sigs.github.io/aws-load-
            # If you have created a Security Group with Terraform
        alb.ingress.kubernetes.io/security-groups: <SG_NAME>

            # Option 2 Prefix Lists
            # Reference https://github.ol.epicgames.net/charts/epic
            # If you have created a managed prefix list with Terraf
        service.epicgames.com/prefix-list-names: <PL_NAME>

            # Option 3 Inbound CIDRs
            # If you want to specify specific CIDR ranges, for exam
        alb.ingress.kubernetes.io/inbound-cidrs: 0.0.0.0/0
```

For additional examples of configurations (fixed responses, AWS WAF, etc.) see Controlling your Load Balancer in Kubernetes.

Manage inbound network traffic to your Public Application Load Balancer (with Security Groups, Prefix Lists, or Inbound CIDR annotations)

When managing inbound internet facing traffic to your Application Load Balancer, the preferred method to do this is by using the native tools, i.e. Security Groups, Prefix Lists, or Inbound CIDR annotations. There are 3 different options for doing this, but all offer the flexibility of allow you to create your own resources with Terraform that can be managed via code and not have to rely on IxM, which is considered deprecated.

If you expand the example in the Create a Kubernetes Ingress for Internet Facing access section, you can see all three options. These options are mutually exclusive. Meaning you will want to chose one option.

Security Groups

If you have created a Security Group that you manage via Terraform, you can specify that security group using the following annotation:

```
alb.ingress.kubernetes.io/security-groups: <SG_NAME>
```

Reference https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/#security-groups

Prefix Lists

If you have created a Managed Prefix List that you manage via Terraform, you can specify that prefix list using the following annotation:

```
service.epicgames.com/prefix-list-names: <PL_NAME>
```

Reference https://github.ol.epicgames.net/charts/epic-app/blob/main/docs/values/values.md#loadbalancers

Inbound CIDRs

If you want to specify inbound CIDR blocks or all CIDRs, you can specify that using the following annotation.

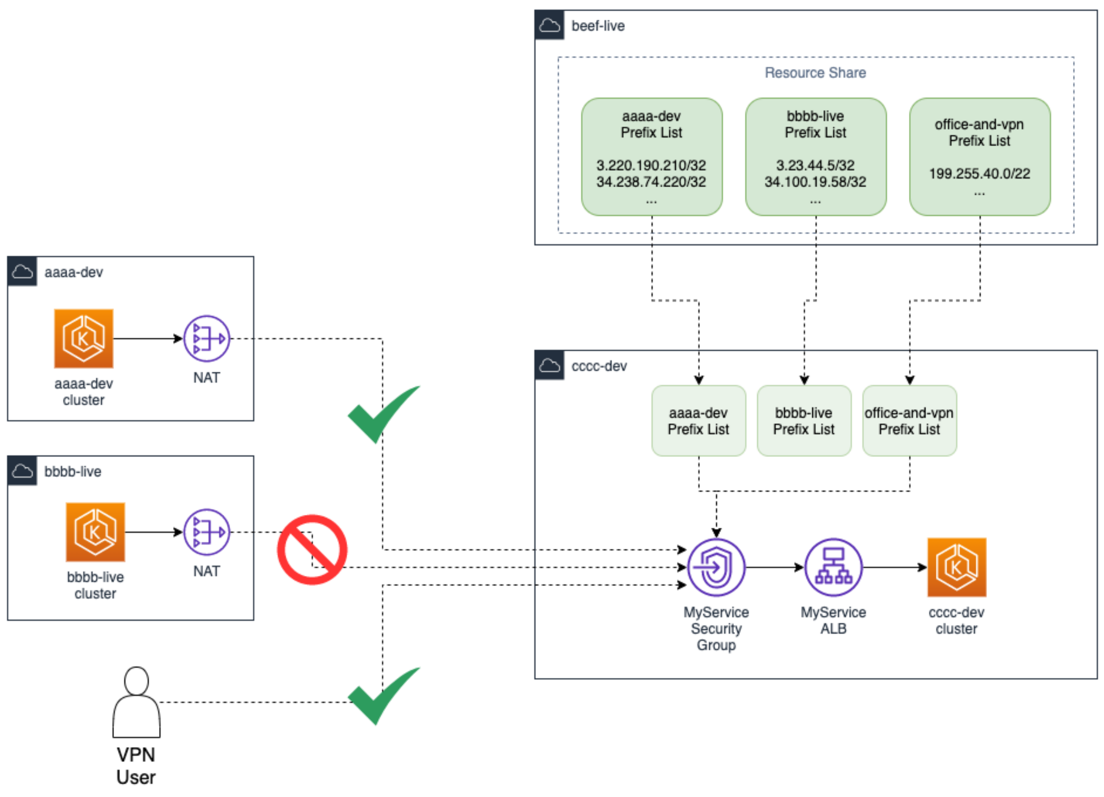
```
alb.ingress.kubernetes.io/inbound-cidrs: 0.0.0.0/0
```

Reference https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/#inbound-cidrs

Manage inbound network traffic to your Public Application Load Balancer (with IxM - DEPRECATED)

Inbound network traffic to an Internet facing Application Load Balancer is controlled by security groups and associated rules. In Substrate, internet facing security group rules are managed using Interconnect Manager (IxM).

Substrate clusters use a custom controller that automatically creates a security group and attaches it to your Application Load Balancer. By default, no inbound traffic is allowed (there are no inbound rules defined). As a service owner, you will need to explicitly allow inbound traffic. The recommended way to do this is using the `ixm` CLI for public load balancers.



A common first step is to allow traffic from Epic VPN and Office IP addresses – using the following command in an authenticated terminal session:

```
ixm service authorize 'myservice.abcd.dev.use1a.on.epicgames.com' -s ix
```

To allow traffic from other services, you will need to know the Substrate account name of the other service. Use the following command in an authenticated terminal session:

```
ixm service authorize 'myservice.abcd.dev.use1a.on.epicgames.com' -s ix
```

For other ways to use `ixm` , see IxM - CLI Reference.

Using `ixm` to allow inbound traffic will allow all services in the source Substrate cluster to communicate with your service. The inbound traffic authorization is *source cluster to destination service*, and **not** *source service to destination service*.

Page Information:
Page ID: 81068466
Space: Cloud Developer Platform
Downloaded: 2025-07-12 04:08:06