우선은 보다 편하게 calico network와 security policy를 관리하기 위해서 Calicoctl을 설치해준다.

|  |
| --- |
| $ curl -L https://github.com/projectcalico/calico/releases/download/v3.24.2/calicoctl-linux-amd64 -o calicoctl  $ chmod +x calicoctl  $ sudo cp calicoctl /usr/bin/ |

그리고 이제 EKS 환경에 calico를 설치해주자.

|  |
| --- |
| $ kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/config/master/calico-operator.yaml  $ kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/config/master/calico-crs.yaml |

그리고 아래와 같은 명령어를 입력 했을 때 DESIRED의 값과 READY의 값이 서로 일치한지 확인해준다.

|  |
| --- |
| $ kubectl get daemonset calico-node --namespace calico-system |



우선 아래의 명령어로 Front-end, Back-end, Client 및 관리사용자 인터페이스 서비스를 적용해준다.

|  |
| --- |
| $ kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/tutorials/stars-policy/manifests/00-namespace.yaml  $ kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/tutorials/stars-policy/manifests/01-management-ui.yaml  $ kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/tutorials/stars-policy/manifests/02-backend.yaml  $ kubectl apply -f https://docs.projectcalico.org/v3.5/getting-started/kubernetes/tutorials/stars-policy/manifests/03-frontend.yaml  $ kubectl apply -f <https://docs.projectcalico.org/v3.5/getting-started/kubernetes/tutorials/stars-policy/manifests/04-client.yaml>  $ kubectl port-forward service/management-ui -n management-ui 9001 |

그리고 해당 port-forward 명령어를 실행한 인스턴스의 공인 IP 주소로 접근하면 아래와 같은 사이트가 출력되는 것을 볼 수 있다.

텍스트, 전자기기, 스크린샷이(가) 표시된 사진

자동 생성된 설명

※ 위 사이트는 Frontend, Backend, Client 간의 통신 상태를 간단하게 눈으로 확인 할 수 있다.

일단 각 네임스페이스 별로 서비스를 각각 격리 시켜주자

|  |
| --- |
| $ vim default-deny.yaml |
| kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  name: default-deny  spec:  podSelector:  matchLabels: {} |

|  |
| --- |
| $ kubectl apply -f default-deny.yaml -n stars  $ kubectl apply -f default-deny.yaml -n client |

이제 새로고침하면 사이트에는 아무런 인터페이스가 출력되지 않은 것을 볼 수 있다.

인터페이스를 서비스에 접근할 수 있도록 management-ui namespace로부터 들어오는 통신만 허용해주자. Network policy를 적용시켜주자.

|  |
| --- |
| $ vim allow-ui.yaml |
| kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  namespace: stars  name: allow-ui  spec:  podSelector:  matchLabels: {}  ingress:  - from:  - namespaceSelector:  matchLabels:  role: management-ui  ---  kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  namespace: client  name: allow-ui  spec:  podSelector:  matchLabels: {}  ingress:  - from:  - namespaceSelector:  matchLabels:  role: management-ui |

위와 같이 작성하면 management-ui 네임스페이스에 있는 오브젝트만 client, stars 네임스페이스에 있는 오브젝트에 네트워크 통신이 가능하다 하지만, 그 외 다신 네임스페이스에서는 접근이 불가하다.

management-ui -> client

텍스트이(가) 표시된 사진

자동 생성된 설명

default -> client

텍스트이(가) 표시된 사진

자동 생성된 설명

동일 네임스페이스에서 특정 파드와의 통신만 허용하기

이러면 pod label이 role: backend인 pod와 role: frontend인 POd끼리 TCP 6379 포트의 통신만 허용이 된 것이다.

|  |
| --- |
| $ vim backend-policy.yaml |
| kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  namespace: stars  name: backend-policy  spec:  podSelector:  matchLabels:  role: backend  ingress:  - from:  - podSelector:  matchLabels:  role: frontend  ports:  - protocol: TCP  port: 6379 |

다른 네임 스페이스의 특정 파드와의 통신 허용

|  |
| --- |
| $ vim front-policy.yaml |
| kind: NetworkPolicy  apiVersion: networking.k8s.io/v1  metadata:  namespace: stars  name: frontend-policy  spec:  podSelector:  matchLabels:  role: frontend  ingress:  - from:  - namespaceSelector:  matchLabels:  role: client  ports:  - protocol: TCP  port: 80 |

**Logging하기 (Generate Log)**