**Security Groups for Pods란?**

AWS EC2 보안그룹을 Kubernetes에서 사용할 수 있는 기능이다. Amazon Ec2보안그룹을 사용하여 Inbound 및 Outbound등에 대한 네트워크 트래픽을 관리할 수 있다.

배포하기 전에 아래의 제한 사항을 확인해준다.

* Amazon EC2 노드에서 실행되는 Pods는 모두 EKS 지원 Kubernetes 버전에 있을 수 있지만 Fargate에서 실행되는 Pods는 1.18 이상의 클러스터에 있어야한다.
* Pods의 보안 그룹은 Windows Node에서는 사용할 수 없다.
* Amazon EC2 노드가 포함된 IPv6 패밀리용으로 구성된 클러스터에서 사용할 수 없다. 하지만 Fargate 노드만 포함하는 IPv6 패밀리용으로 구성된 클러스터가 있는 Pods의 보안 그룹을 사용할 수 있다.
* pod 변형에 대한 액세스를 제한하는 경우 psp가 할당된 role에 대해 Kubernetes ClusterRoleBinding에 eks-vpc-resource-controller 및 vpc-resource-controller Kubernetes 서비스 계정이 지정되어 있어야한다.

|  |
| --- |
| subjects:  - kind: Group  apiGroup: rbac.authorization.k8s.io  name: system:authenticated  - kind: ServiceAccount  name: vpc-resource-controller  - kind: ServiceAccount  name: eks-vpc-resource-controller |

* 사용자 지정 네트워킹 및 Pods의 보안 그룹을 함께 사용하는 경우 ENIconfig에 지정된 보안 그룹 대신 pods의 보안 그룹에서 지정한 보안 그룹이 사용된다.

**실습**

|  |
| --- |
| $ aws eks describe-cluster --name awsvpccnitest --query cluster.roleArn --output text |

그리고 출력된 IAM Role의 이름을 기억하고 아래의 파란 색으로 표시한 부분을 수정해준다.

|  |
| --- |
| $ aws iam attach-role-policy \  --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController \  --role-name eksctl-awsvpccnitest-cluster-ServiceRole-1ET6XVF2KLRS4 |

※ 이는 해당 Cluster Role에 AmazonEKSVPCResourceController 권한을 부여해주는 작업이다.

그 후 aws-node Daemonset에서 ENABLE\_POD\_ENI 변수를 true로 설정하여 Amazon VPC CNI 추가 기능이 pods의 네트워크 인터페이스를 관리하도록 설정해준다. 이 설정이 true로 설정되면, 클러스터의 각 노드에 대해 추가 기능이 vpc.amazonaws.com/has-trunk-attached=true 값이 있는 레이블을 추가한다. vpc resource controller는 aws-k8s-trunk-eni를 사용하여 트렁크 네트워크 인터페이스라는 하나의 특수 네트워크 인터페이스를 생성하고 연결한다.

|  |
| --- |
| $ kubectl set env daemonset aws-node -n kube-system ENABLE\_POD\_ENI=true |

그리고 Trunk Network Interface가 생성이되면 트렁크 또는 표준네트워크 인터페이스에서 pods에 보조 IP 주소를 할당한다. 노드가 삭제되면 트렁크 인터페이스가 자동으로 삭제된다. 이후 단계에서 pod에 대한 보안 그룹을 배포하면 VPC 리소스 컨트롤러가 aws-k8s-branch-enu의 설명과 함께 Branch Network Interface라는 특수한 네트워크 인터페이스를 생성하고 거기에 보안 그룹을 연결한다. Branch Network Interface는 노드에 연결된 표준 및 트렁크 네트워크 인터페이스와 함께 생성된다. 라이브니스 또는 준비 프로브를 사용하는 경우 kubelet이 TCP를 사용해 Branch Network Interface의 pods에 연결할 수 있도록 TCP 초기 demux를 사용 정지해야한다. TCP 초기 Demux를 비활성화해주자.

|  |
| --- |
| $ kubectl patch daemonset aws-node \  -n kube-system \  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":[{"name":"DISABLE\_TCP\_EARLY\_DEMUX","value":"true"}],"name":"aws-vpc-cni-init"}]}}}}' |

※ POD\_SECURITY\_GROUP\_ENFORCING\_MODE=standard로 설정된 1.11.0 이상의 Amazon VPC CNI plugin for kubernetes 추가 기능을 사용하는 경우 다음 단계에 설명한대로 이전 명령을 실행할 필요가 없다.

클러스터에서 NodeLocal DNSCache를 사용하거나, 자체 보안 그룹이 있는 pods를 통해 Calico Network Policy를 사용하거나, 보안 그룹을 할당하려는 pods에 대해 externalTrafficPolicy를 Local로 설정한 인스턴스 대상을 사용하는 NodePort 및 LoadBalancer 유형의 Kubernetes 서비스가 있는 경우 버전 1.11.0이 상의 Amazon VPC CNI Plugin for Kubernetes 추가 기능을 사용하고 있어야하며 다음 설정을 사용 설정해야한다.

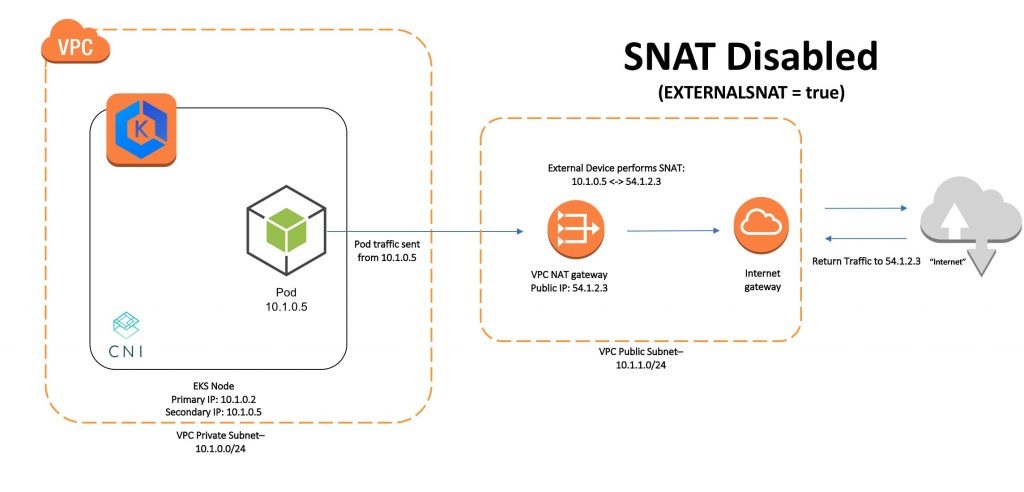
|  |
| --- |
| $ kubectl set env daemonset aws-node -n kube-system POD\_SECURITY\_GROUP\_ENFORCING\_MODE=standard |

※ Pod 보안 그룹의 규칙은 kubelet 또는 nodeLocalDNS와 같이 동일한 노드에 있는 pods간 또는 Pods와 service간의 트래픽이 적용되지 않는다.

또한 외부와 통신을 하기 위해서 아래와 같은 옵션을 추가해준다.

만약 NAT가 외부 디바이스(예: 인스턴스 자체가 아닌 NAT Gatway)에서 처리되도록 지정하기 위해 AWS\_VPC\_K8S\_CNI\_EXTERNALSNAT 환경 변수를 true로 설정하여 인스턴스에서 SNAT을 비활성화할 수 있다.

즉, 퍼블릭 서브넷에 있는 노드의 경우 SNAT이 필요하다. 외부 SNAT을 사용하려면 노드가 Private Subnet에 있어야하며, NAT Gateway 또는 다른 외부 NAT 디바이스를 통해 인터넷에 연결이 되어야한다.



1. Worker Node가 NAT Gateway를 거쳐서 외부랑 통신 되는 경우

|  |
| --- |
| $ kubectl set env daemonset -n kube-system aws-node AWS\_VPC\_K8S\_CNI\_EXTERNALSNAT=true  $ kubectl rollout restart daemonset aws-node -n kube-system |

1. Worker Node가 NAT Gateway를 안 거치고 외부랑 통신 되는 경우

|  |
| --- |
| $ kubectl set env daemonset -n kube-system aws-node AWS\_VPC\_K8S\_CNI\_EXTERNALSNAT=false  $ kubectl rollout restart daemonset aws-node -n kube-system |

※ pods에서 vpc 외부의 주소로 향하는 아웃바운드 트래픽은 인스턴스의 기본 네트워크 인터페이스의 IP 주소로 변환된 네트워크 주소이다. (AWS\_VPC\_K\*S\_CNI\_EXTERNALNAT=true 로 성정하지 않은 경우 제외). 이 트래픽의 경우 pods 보안 그룹의 규칙이 아닌 기본 네트워크 인터페이스에 대한 보안 그룹의 규칙이 사용된다.

※ 이 설정을 기존 pods에 적용하려면 pods 또는 pods가 실행되고 있는 노드를 다시 시작해야한다.

이제 테스트를 위해 namespace를 생성해주자.

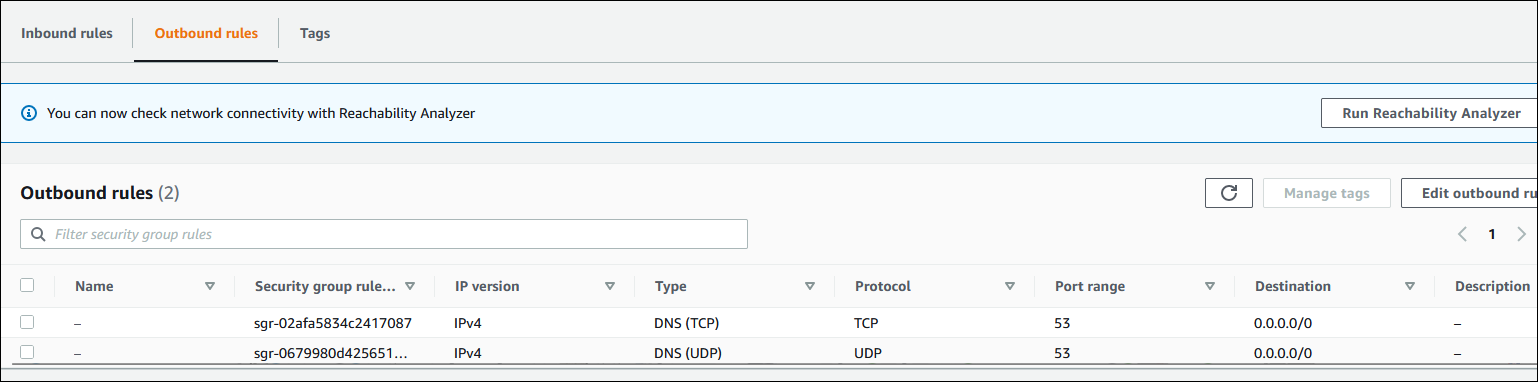
|  |
| --- |
| $ kubectl create ns pod-sg |

그리고 아래의 예제 보안 정책을 작성해주자. 서비스 계정 레이블에 따라 pods를 선택하려는 경우 podSelector를 serviceAccountSelector로 바꿀 수 있다. 두 선택시 중 하나만 지정해야한다. 비어 있는 podSelector(예: podSelector: {})는 네임스페이스의 모든 pods를 선택한다. 비어 있는 serviceAccountSelector는 네임스페이스의 모든 서비스 계정을 선택한다. groupIds에 대해 1~5개의 보안 그룹 id를 지정해야한다. 둘 이상의 ID를 지정하면 모든 보안 그룹의 모든 규칙 조합이 선택한 pods에 적용된다.

일단 아래와 같은 규칙을 가진 Security Group을 생성해주자.

텍스트이(가) 표시된 사진

자동 생성된 설명



|  |
| --- |
| apiVersion: vpcresources.k8s.aws/v1beta1  kind: SecurityGroupPolicy  metadata:  name: my-security-group-policy  namespace: pod-sg  spec:  podSelector:  matchLabels:  role: my-role  securityGroups:  groupIds:  - sg-04579a6c457bf43a6 |

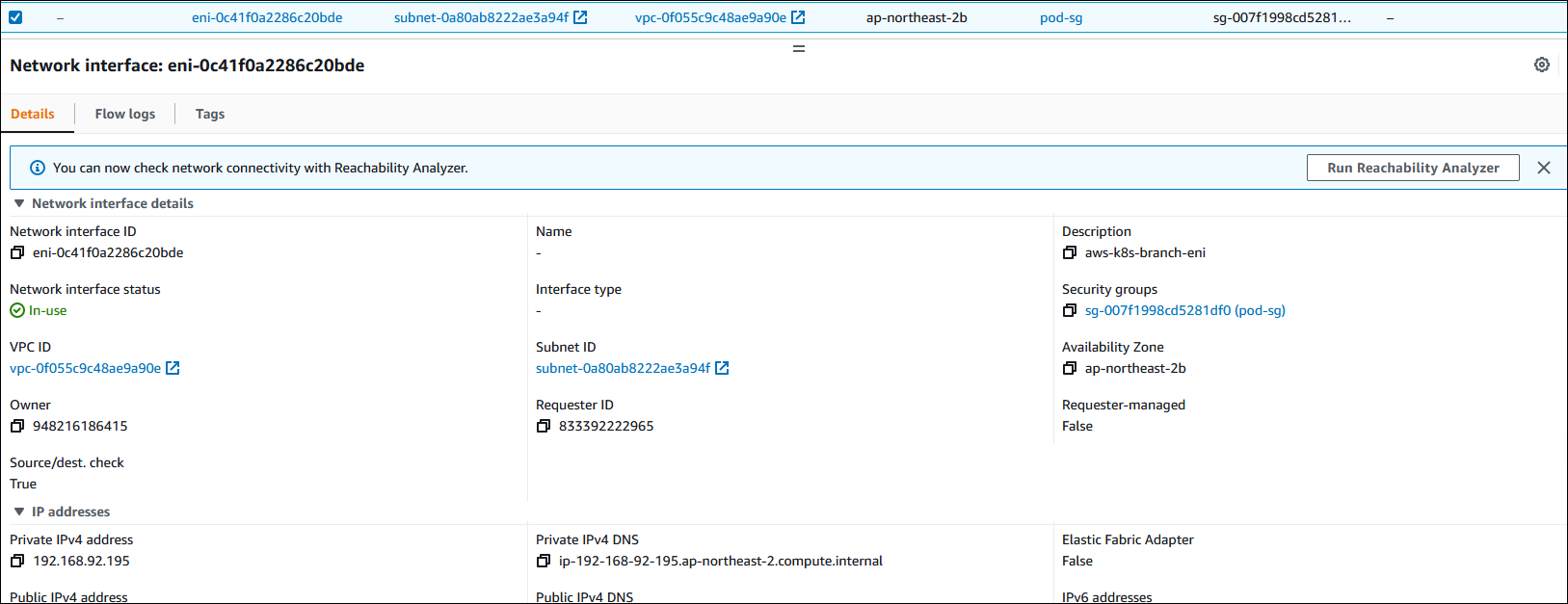
그리고 이제 deployment를 생성해주자.

|  |
| --- |
| apiVersion: apps/v1  kind: Deployment  metadata:  name: my-deployment  namespace: pod-sg  labels:  app: my-app  spec:  replicas: 1  selector:  matchLabels:  app: my-app  template:  metadata:  labels:  app: my-app  role: my-role  spec:  containers:  - name: my-container  image: my-image  ports:  - containerPort: 80 |

※ 만약 Deployment의 Pod가 생성되지 않고 아래와 같은 오류가 발생했다면, 인스턴스 유형에서 실행할 수 있는 최대 pods수를 초과할 수 없기 때문에 발생하는 오류이다.

|  |
| --- |
| Failed to create pod sandbox: rpc error: code = Unknown desc = failed to set up sandbox container "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f" network for pod "my-deployment-59f5f68b58-c89wx": networkPlugin  cni failed to set up pod "my-deployment-59f5f68b58-c89wx\_my-namespace" network: add cmd: failed to assign an IP address to container |

정상 적으로 pod가 생성이 되었다면 아래와 같이 branch interface인 eni가 생성된 것을 볼 수 있다.



그리고 Pod를 10개 생성하면 아래와 같이 인터페이스도 10개가 생성된다.

테이블이(가) 표시된 사진

자동 생성된 설명

그리고 아무 EC2를 생성해서 해당 인스턴스에 Ping을 날려보면 아래와 같이 정상 적으로 통신이 되는 것을 볼 수 있다.

