역할 부여 삼총사

IMDS, IRSA, Pod Identity

톺아보기

발표자 소개

- 카카오뱅크 클라우드엔지니어 천강민
- 편안한 노후를 위해 열심히 달리는 중
  - 실무에서 사용하는 클라우드 프로그래밍
  - 실무에서 사용중인 AWS 클라우드 IAM 이해와 보안
  - Python을 활용한 AWS FinOps 어플리케이션 제작
  - 빠르고 안전한 어플리케이션 배포 파이프라인(CI/CD) 만들기

빠르게 본론으로

1. IMDS (**I**nstance **M**eta**D**ata **S**ervice)
2. IRSA (**I**AM **R**oles for **S**ervice **A**ccounts)
3. Pod Identity

란 무엇인지?

어떻게 동작하는지?

어떨 때 취약해지는지?

안전하려면?

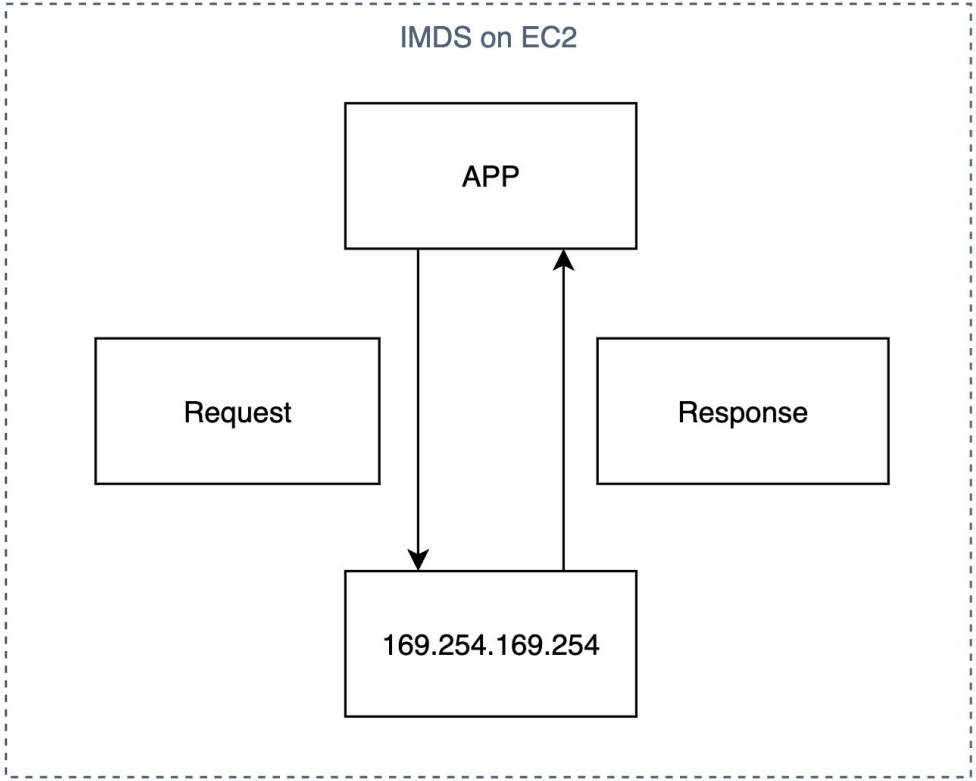
IMDS (Instance MetaData Service)

IMDS (**I**nstance **M**eta**D**ata **S**ervice) 란?

1. 인스턴스에 대한 데이터를 보유한 서비스
    a. 인스턴스에 사용된 AMI ID (/latest/meta-data/ami-id)
    b. 인스턴스 프로파일에 대한 정보 (/latest/meta-data/iam/info)
    c. 인스턴스 자격 증명 문서 (/latest/dynamic/instance-identity/document, pkcs7, signature)
        i. Vault 를 통한 EC2 인증 등에 사용
    d. Systems Manager 기본 호스트 관리 구성, GuardDuty 런타임 모니터링 등 사용을 위한 인스턴스 자체 자격 증명 (권한 없음)
    (latest/meta-data/identity-credentials/ec2/security-credentials/ec2-instance)
    e. 인스턴스 프로파일의 임시 자격 증명 (latest/meta-data/iam/security-credentials/role-name)
    f. Userdata (/latest/user-data)
    g. 등등

동작 방식 이해

IMDS 동작 방식

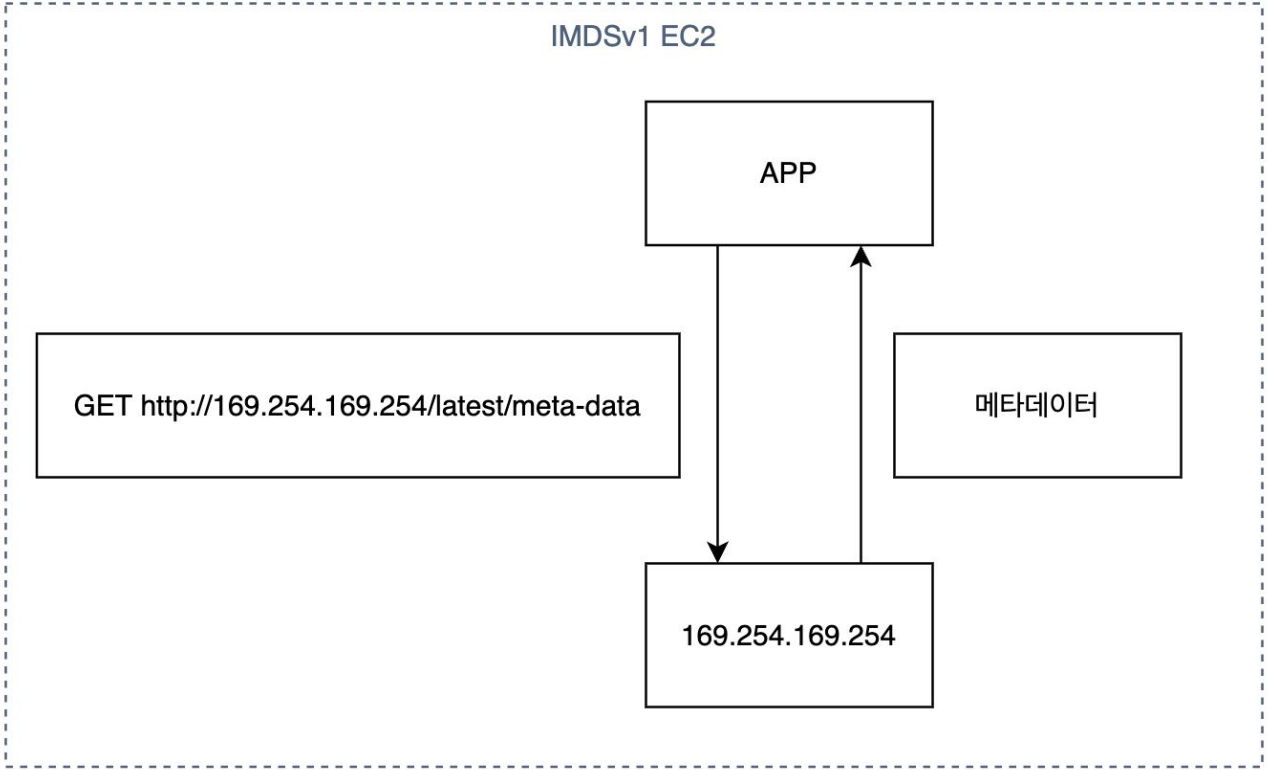


IMDSv1 vs IMDSv2

IMDSv1

1. IMDSv1 을 사용한다는 것이 IMDSv2 를 사용하지 못하는 것은 아님
    a. 메타데이터 사용 (http-endpoint: enabled) 설정 시, IMDS 버전 선택은 **선택적**임 **(http-tokens: optional)**
    b. 기본적으로 IMDSv1, IMDSv2 모두 사용 가능 상태
2. 다만, 2022년 10월 부터 인스턴스 실행 시 IMDSv2 가 기본으로 설정됨
    a. **http-tokens: required**
3. HTTP GET 메소드를 통한 “**요청/응답**” 방식으로, 별도의 토큰 발급 필요없이 요청만 할 수 있다면 **임시자격증명**을 획득 가능

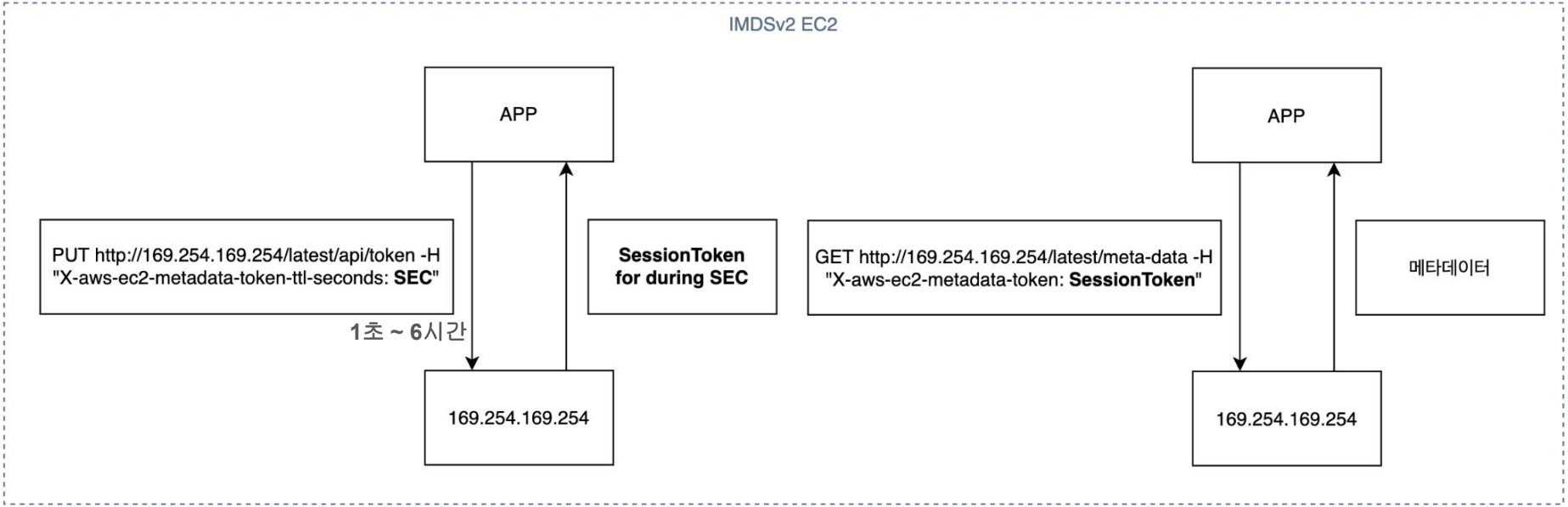
IMDSv1 동작 방식


IMDSv1 EC2
APP
GET http://169.254.169.254/latest/meta-data
메타데이터
169.254.169.254

IMDSv2

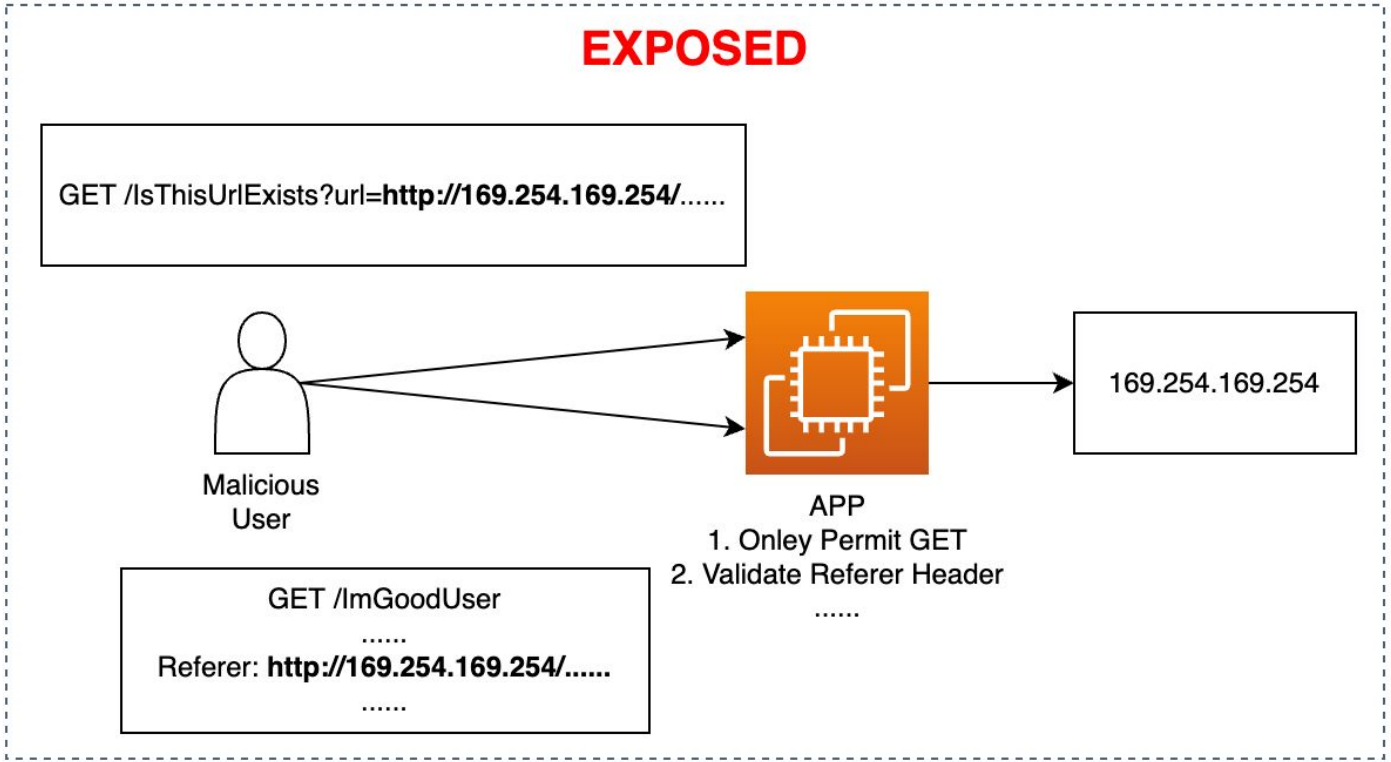
1. IMDSv2 사용을 강제하게 되면, **IMDSv1 방식으로 요청이 불가**
    a. IMDS 버전 선택 강제 **(http-tokens: required)**
2. HTTP PUT 메소드를 통해 **세션토큰**을 받고, 해당 세션토큰을 GET 메소드의 헤더에 추가하여 **임시자격증명**을 획득 가능
    a. IMDSv1 과 달리 **“세션 지향"** 방식이라고 함
3. 이외에도 다양한 보안적 이점을 누릴 수 있음
    a. GET 메소드를 통한 **SSRF (Server Side Request Forgery) 방지** (세션토큰이 있어야 하니까)
    b. **X-Forwarded-For 헤더** 포함된 요청에 대한 **세션토큰 미발급**

IMDSv2 동작 방식

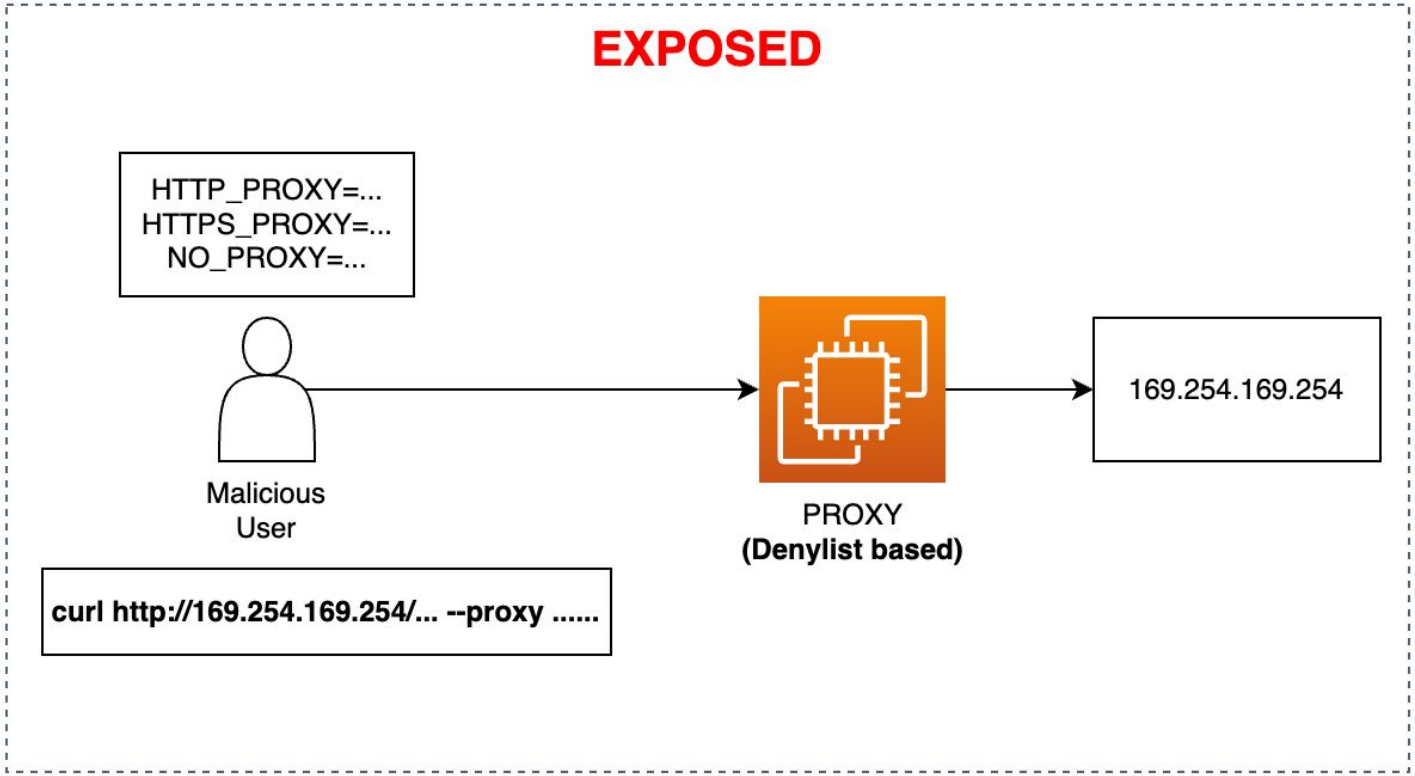

IMDSv2 EC2
APP
PUT http://169.254.169.254/latest/api/token -H "X-aws-ec2-metadata-token-ttl-seconds: SEC"
SessionToken for during SEC
1초 ~ 6시간
169.254.169.254
APP
GET http://169.254.169.254/latest/meta-data -H "X-aws-ec2-metadata-token: SessionToken"
메타데이터
169.254.169.254

IMDSv1 은 왜 취약하다고 하는걸까?

빈번하게 사용되는 GET 요청

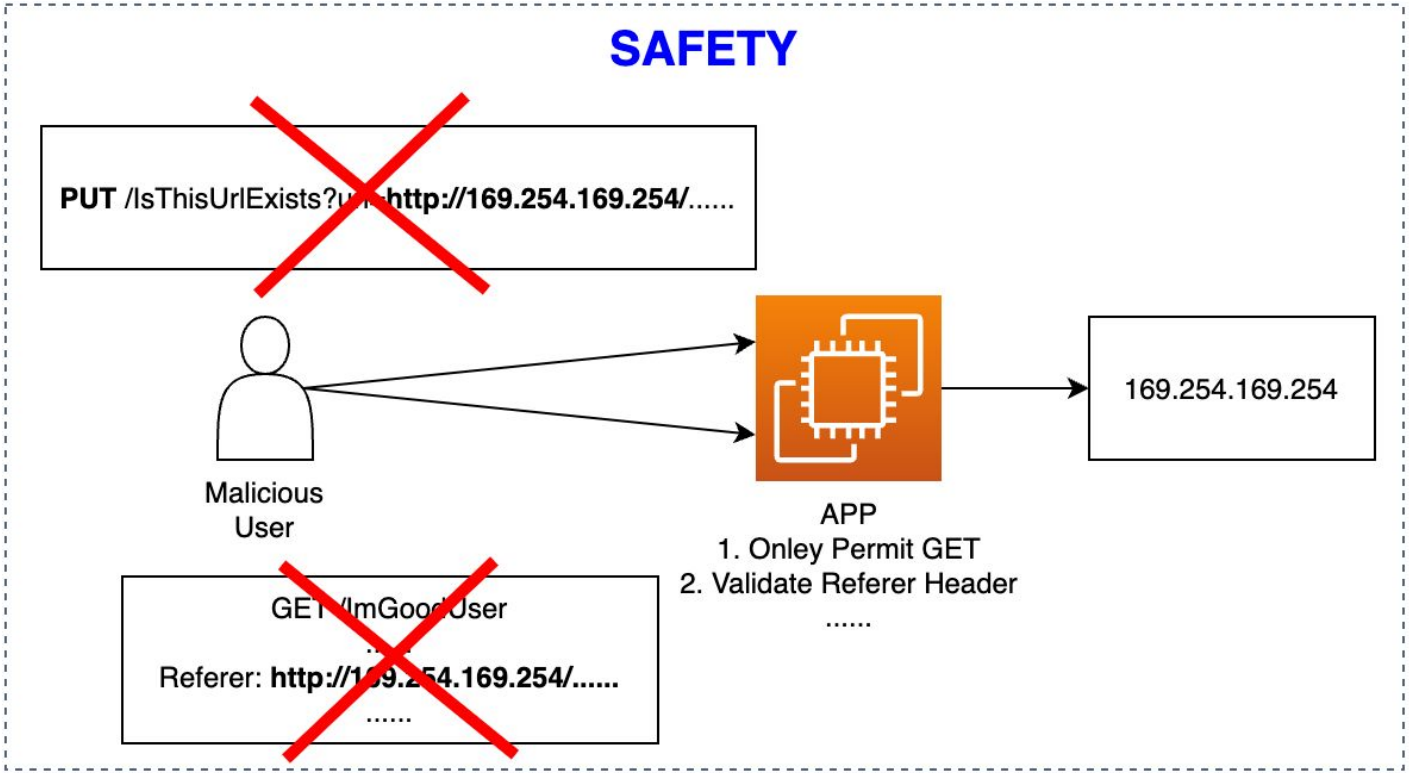


프록시 서버, 웹훅은 잘못이 없어요...

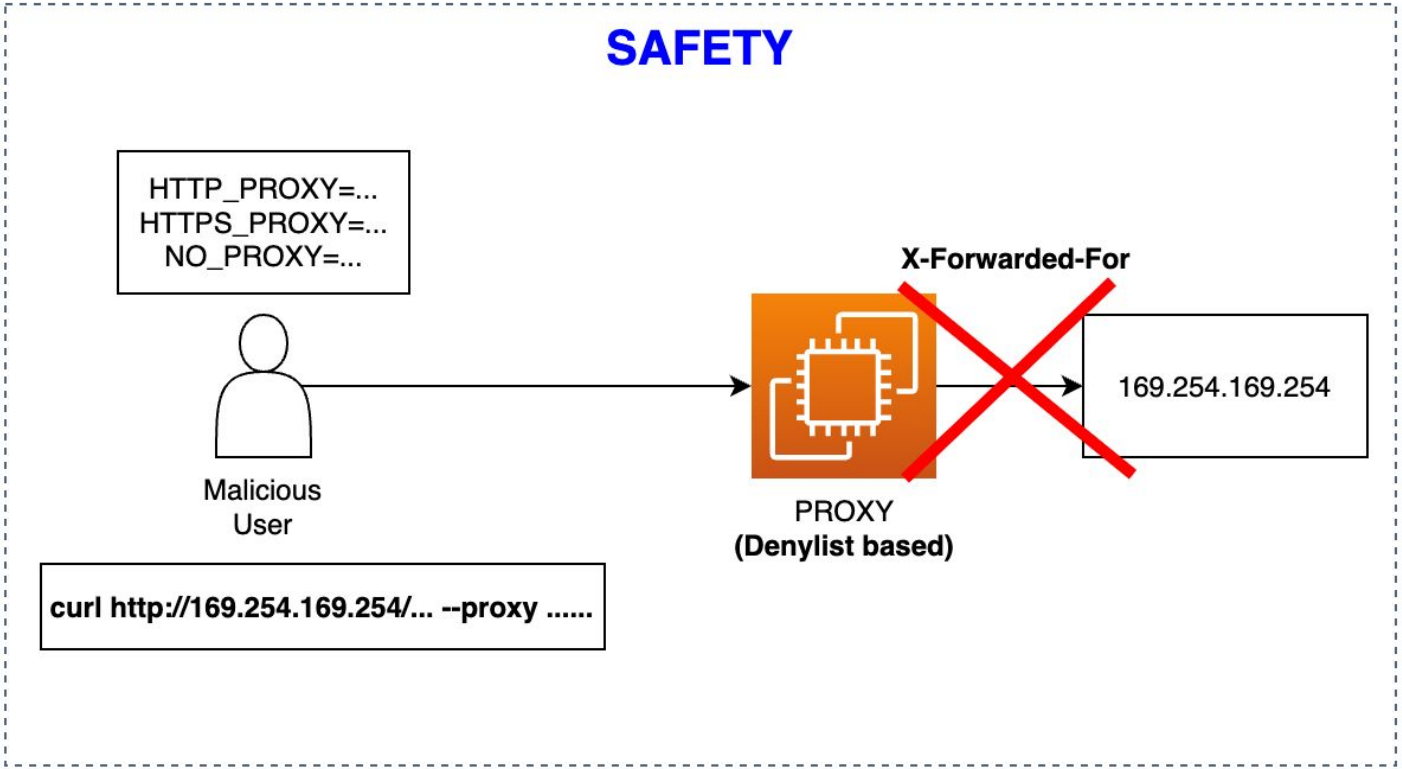


IMDSv2 로 전환하기

PUT 요청이 선행되어야 함



X-Forwarded-For 는 허용하지 않음



IMDSv1 사용 현황 확인하기

1. AWS CLI 를 통해 **“HttpTokens: optional”** 인스턴스 파악
    a. aws ec2 describe-instances --query "Reservations[*].Instances[*].[InstanceId, MetadataOptions]" --region [REGION]
    b. 실제 IMDSv1 사용 여부와 상관없이 설정된 인스턴스 파악
2. CloudWatch 를 통한 **“MetadataNoToken”** 지표 확인
    a. 실제 IMDSv1 사용 중인 인스턴스 파악
3. (필요 시) aws-imds-packet-analyzer 를 통한 프로세스 단위 파악 가능
    a. 명령 인수도 파악 가능
    b. 세부 출력 정보는 링크 참고

안전한 IMDS 환경 구성하기

IMDS 안전하게 사용하기

1. SCPs 를 통한 정책 강화
    a. EC2 시작 시, IMDSv2 강제
    b. IMDSv1 자격증명 동작 제한
    c. EC2 메타데이터 정보 변경 금지
    d. 자격증명 반환 홉수 제한
    e. 인스턴스 프로파일 외부 사용 제한
2. 프록시 서버 구성 시, 메타데이터 서비스 주소 요청 제한
3. 웹훅 기능에 대한 제어 가능 여부 검토

EC2 시작 시, IMDSv2 강제 (기존 인스턴스 영향 없음)

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "RequireImdsV2",
        "Effect": "Deny",
        "Action": "ec2:RunInstances",
        "Resource": "arn:aws:ec2:*:*:instance/*",
        "Condition": {
            "StringNotEquals": {
                "ec2:MetadataHttpTokens": "required"
            }
        }
    }
}
```

IMDSv1 자격증명 동작 제한 (자격증명 발급은 됨, 매우 주의, 장애남)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireAllEc2RolesToUseV2",
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "NumericLessThan": {
                    "ec2:RoleDelivery": "2.0"
                }
            }
        }
    ]
}
```

EC2 메타데이터 정보 변경 금지 (기존 인스턴스 영향 없음)

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "ec2:ModifyInstanceMetadataOptions",
        "Resource": "*"
    }
}
```

자격증명 반환 홉수 제한 (컨테이너 사용 시 매우 주의, 기본 값은 2)

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "MaxImdsHopLimit",
        "Effect": "Deny",
        "Action": "ec2:RunInstances",
        "Resource": "arn:aws:ec2:*:*:instance/*",
        "Condition": {
            "NumericGreaterThan": {"ec2:MetadataHttpPutResponseHopLimit": "1"}
        }
    }
}
```

인스턴스 프로파일 외부 사용 제한

```
{
    "Sid": "RequireSameVPC",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "StringNotEquals": {
            "aws:SourceVpc": "${aws:Ec2InstanceSourceVpc}"
        },
        "Null": {
            "ec2:SourceInstanceARN": "false"
        },
        "NotIpAddressIfExists": {
            "aws:SourceIp": [
                "[redacted]"
            ]
        },
        "BoolIfExists": {
            "aws:ViaAWSService": "false"
        }
    }
},
```

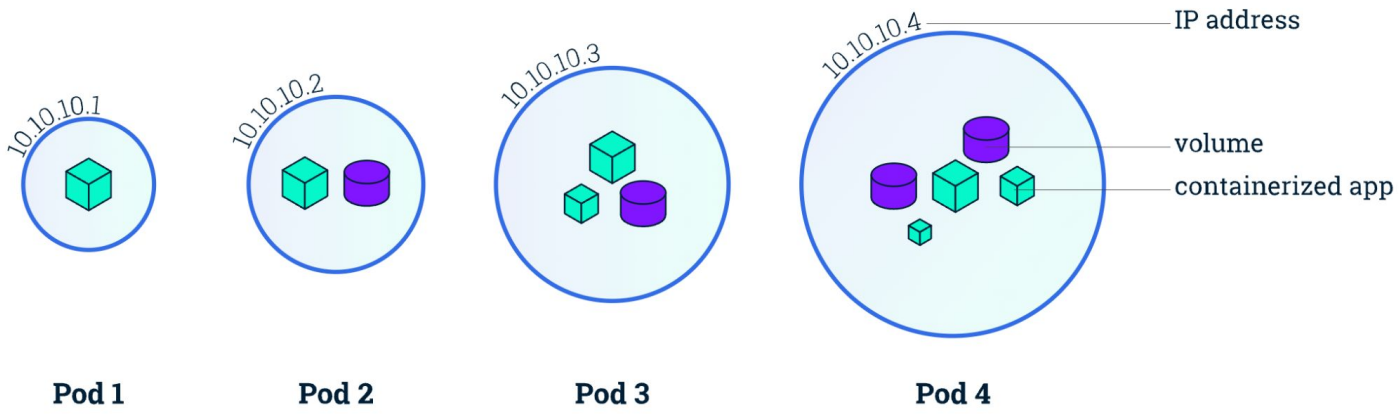
```
{
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "StringNotEquals": {
            "aws:VpcSourceIp": "${aws:Ec2InstanceSourcePrivateIPv4}"
        },
        "Null": {
            "ec2:SourceInstanceARN": "false"
        },
        "NotIpAddressIfExists": {
            "aws:SourceIp": [
                "[redacted]"
            ]
        },
        "BoolIfExists": {
            "aws:ViaAWSService": "false"
        }
    }
}
```

IRSA (IAM Roles for Service Accounts)

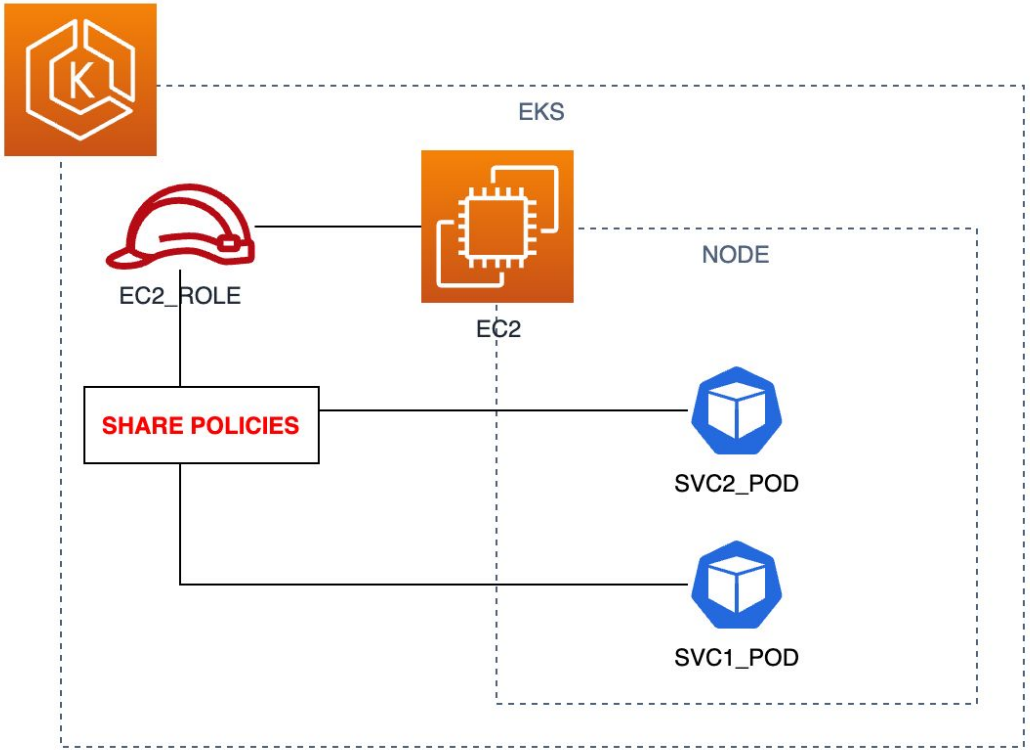
IRSA 란?

IRSA (**I**AM **R**oles for **S**ervice **A**ccounts) 란?

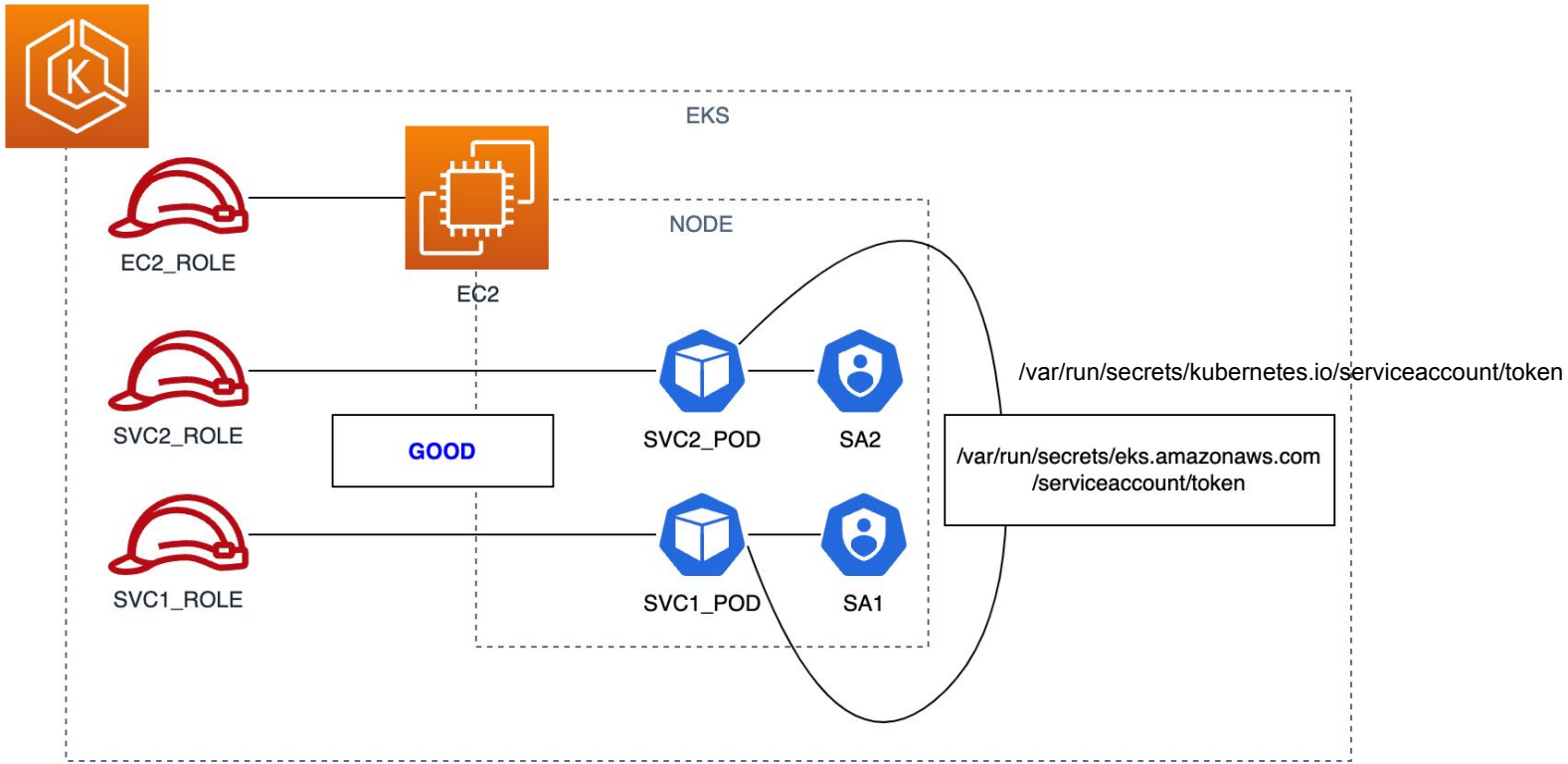
1. Pod 내부 어플리케이션 **컨테이너**가 **역할을 사용**할 수 있게 해주는 기능
    a. Pod 이란?
        i. Kubernetes 배포의 최소 단위



IRSA 등장 배경 - IRSA가 없었을 때 (오픈소스 제외)

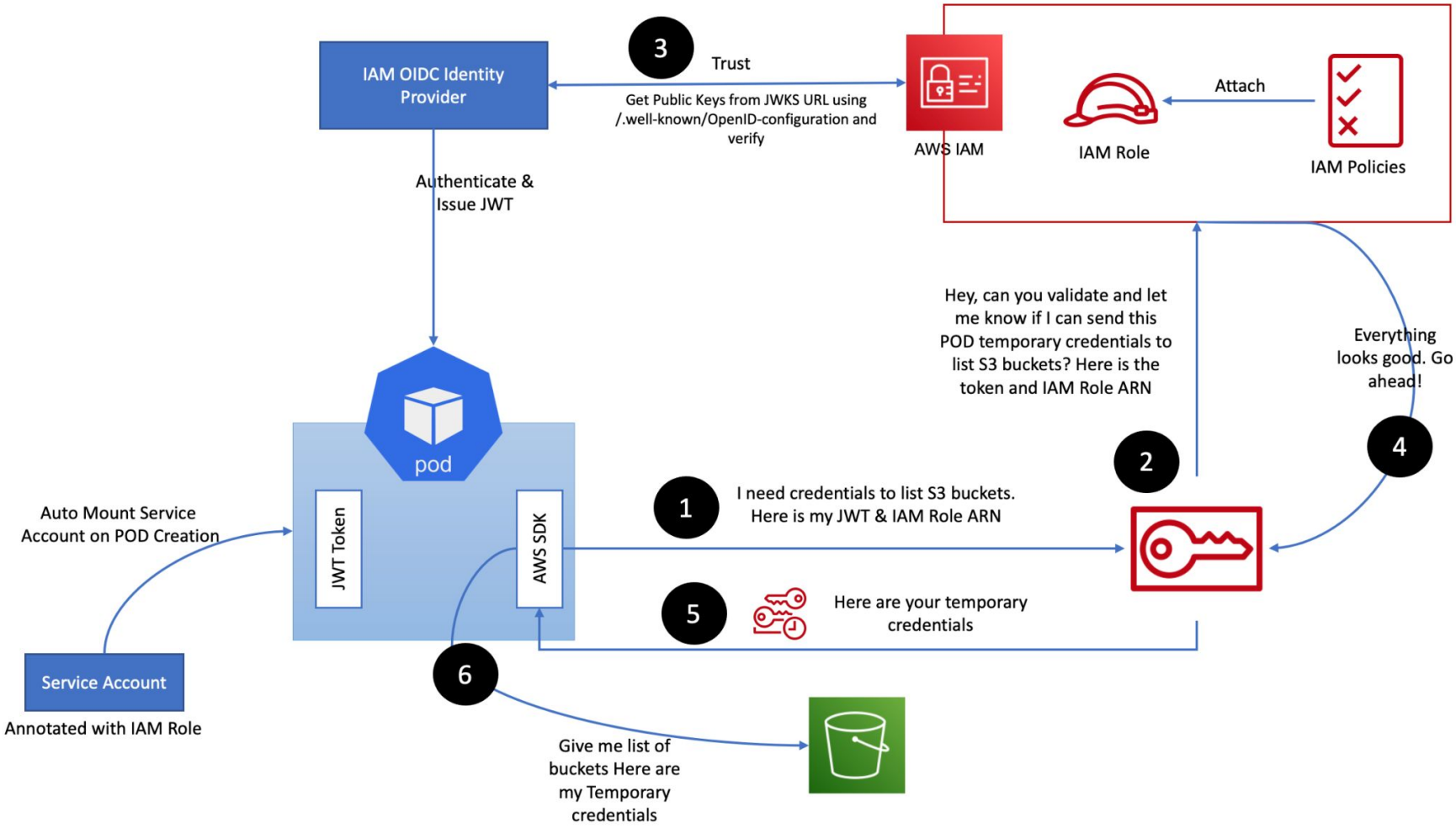


IRSA 등장 배경 - IRSA 등장 이후 (간단 버전)



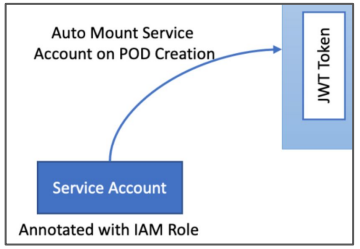
동작 방식 이해

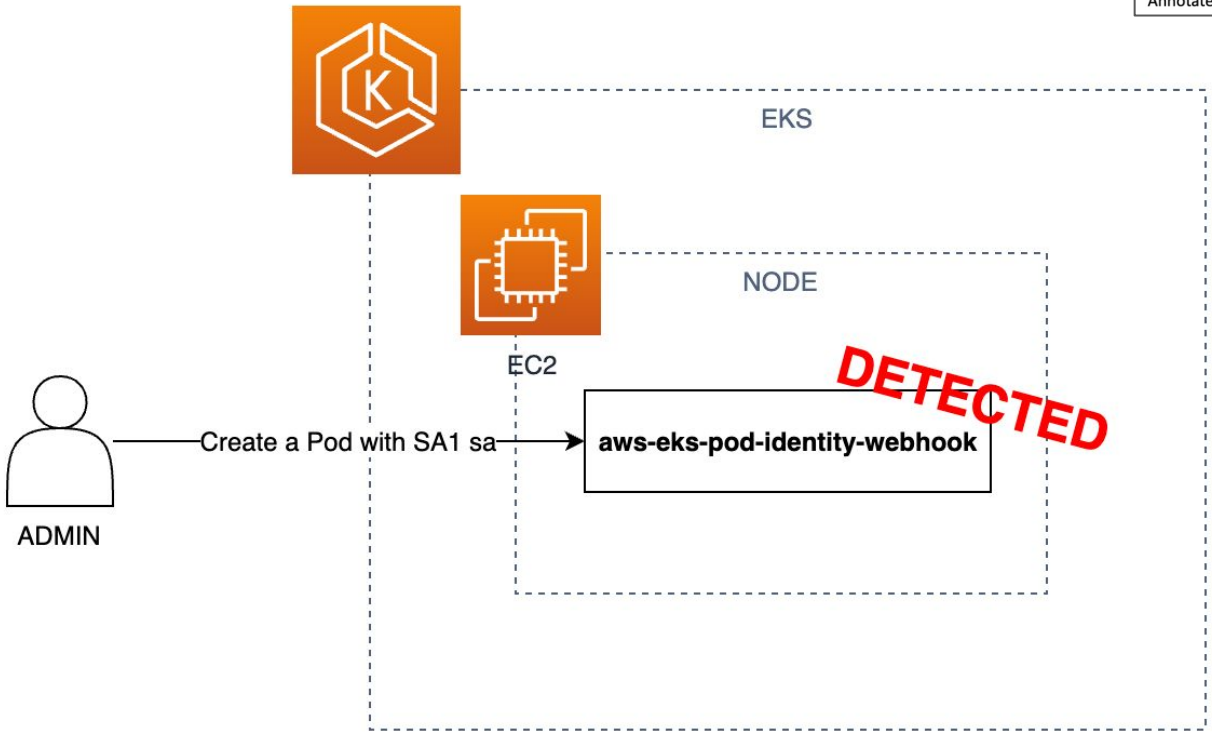
크게 살펴보기


IAM OIDC Identity Provider
3
Trust
Get Public Keys from JWKS URL using /.well-known/OpenID-configuration and verify
AWS IAM
Attach
IAM Role
IAM Policies
Authenticate & Issue JWT
Hey, can you validate and let me know if I can send this POD temporary credentials to list S3 buckets? Here is the token and IAM Role ARN
Everything looks good. Go ahead!
pod
JWT Token
AWS SDK
Auto Mount Service Account on POD Creation
1
I need credentials to list S3 buckets. Here is my JWT & IAM Role ARN
2
4
5
Here are your temporary credentials
Service Account
Annotated with IAM Role
6
Give me list of buckets Here are my Temporary credentials

토큰 발급 과정 살펴보기

톺아보기 - Pod 생성 감지

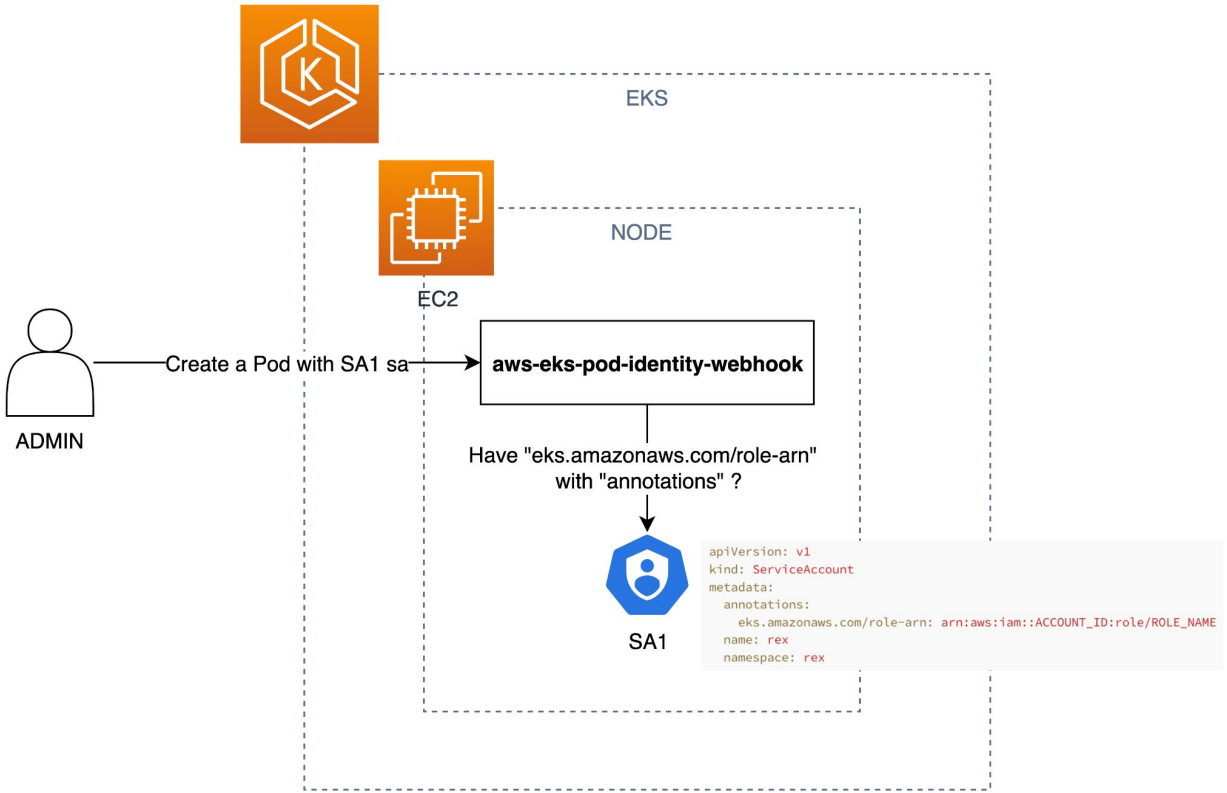




툐아보기 - Pod 생성 감지 (설정)

```
webhooks:
...<SNIP>...
  - operations: [ "CREATE" ]
    apiGroups: [""]
    apiVersions: ["v1"]
    resources: ["pods"]
  sideEffects: None
  admissionReviewVersions: ["v1beta1"]
```

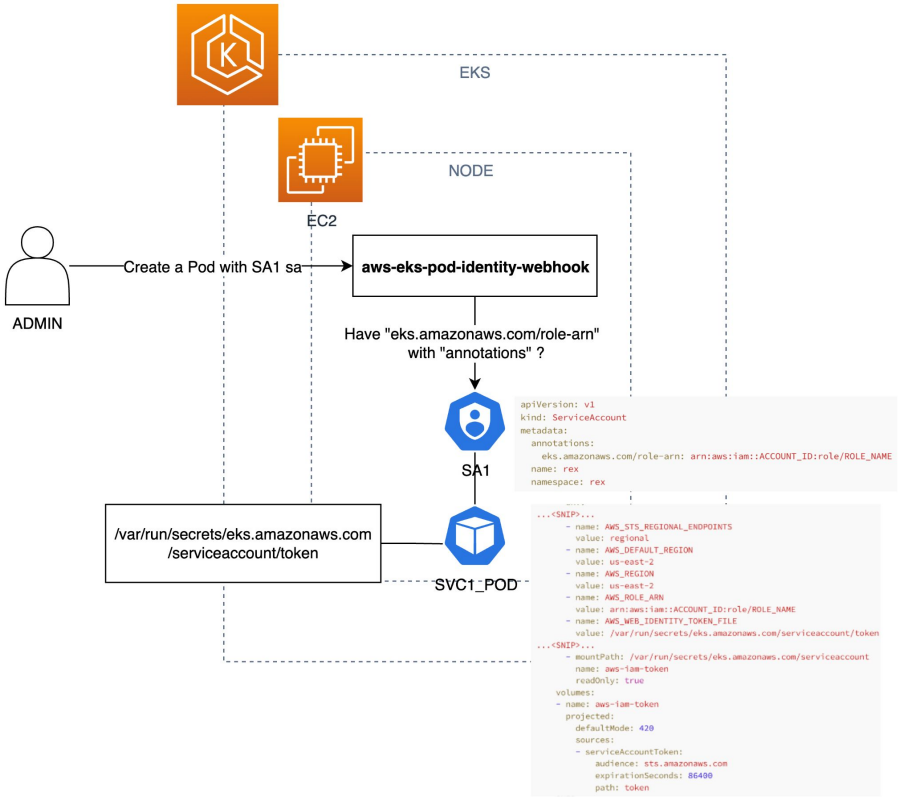
톺아보기 - 서비스 계정 검사



톺아보기 - 서비스 계정 검사 (설정)

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::ACCOUNT_ID:role/ROLE_NAME
  name: rex
  namespace: rex
```

톺아보기 - Pod 에 볼륨마운트 및 환경변수 주입

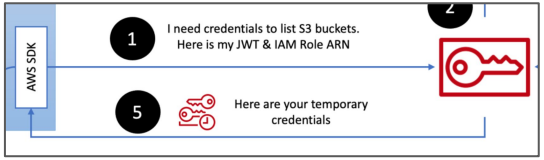


톺아보기 - Pod 에 볼륨마운트 및 환경변수 주입 (설정)

```
apiVersion: v1
items:
- apiVersion: v1
  kind: Pod
...<SNIP>...
  spec:
    containers:
    - env:
...<SNIP>...
      - name: AWS_STS_REGIONAL_ENDPOINTS
        value: regional
      - name: AWS_DEFAULT_REGION
        value: us-east-2
      - name: AWS_REGION
        value: us-east-2
      - name: AWS_ROLE_ARN
        value: arn:aws:iam::ACCOUNT_ID:role/ROLE_NAME
      - name: AWS_WEB_IDENTITY_TOKEN_FILE
        value: /var/run/secrets/eks.amazonaws.com/serviceaccount/token
...<SNIP>...
      - mountPath: /var/run/secrets/eks.amazonaws.com/serviceaccount
        name: aws-iam-token
        readOnly: true
    volumes:
    - name: aws-iam-token
      projected:
        defaultMode: 420
        sources:
        - serviceAccountToken:
            audience: sts.amazonaws.com
            expirationSeconds: 86400
            path: token
...<SNIP>...
```

AWS 와의 통신 과정 살펴보기

AssumeRoleWithWebIdentity 동작 방식



1. 역할의 ARN, 세션명, **암호화된 서명이 포함된 토큰** (JWT) 을 통해 임시 자격 증명 발급
2. API 호출을 위한 자격증명이 필요하지 않음
3. IRSA 의 경우, 토큰의 **“iss”** 항목을 통해 **키가 변조되지 않았음을 확인**할 수 있는 엔드포인트 접근 후, **공개키를 통해 토큰의 무결성을 검사**
4. 이후, **수** 결정

```
response = client.assume_role_with_web_identity(
    RoleArn='string',
    RoleSessionName='string',
    WebIdentityToken='string',
```

토큰

```
sh-4.2$ pwd
/var/run/secrets/eks.amazonaws.com/serviceaccount
sh-4.2$ cat token
eyJhbGciOiJSUzI1NiIsImtpZCI6IjQ5ZDRmMTJkNTE4N2FmMzg
NDMyNTQsImlzcyI6Imh0dHBzOi8vb2lkYy5la3MudXMtZWFzdC0
hcnRzIiwicG9kIjp7Im5hbWUiOiJjYXJ0cy1mOGY3NzhiZGMtbn
widWlkIjoiMTRiMjE4MmYtYWM0Mi00NGY1LWI3YjUtZjM0ZGQ0M
yxI5k4eRGtFcVGxwhyWd1akOB5AsvnEKGEv4b9XE1fC2Ycvn0F8
p0CB4jMN5IwqwtH4-8EGTiQgvedZ4vaTV5KpBlV1HMZou7NXi0m
```

토큰 디코드


```
eyJhbGciOiJSUzI1NiIsImtpZCI6IjQ5ZDRmMTJkNTE4N2FmMzgxYTIzYWU1NDBhNjYyNTU1NTZkZmYyYmMifQ.eyJhdWQiOlsic3RzLmFtYXpvbmF3cy5jb20iXSwiZXhwIjoxNjg1ODI5NjU0LCJpYXQiOjE2ODU3NDMyNTQsImlzcyI6Imh0dHBzOi8vb2lkYy5la3MudXMtZWFzdC0yLmFtYXpvbmF3cy5jb20vaWQvOTI3ODkyMEJEQTkyOUE4QTU0RTkyRTE4QTRERUNCMkEiLCJrdWJlcm5ldGVzLmlvIjp7Im5hbWVzcGFjZSI6Imhhcnp1biwicG9kIjp7Im5hbWUiOiJjYXJ0cy1mOGY3NzhiZGMtbndiNnYiLCJ1aWQiOiI0YTRhNDM3My0wNzUxLTRiMDMtYWZhOC0yZWZiYWI5MzBjMDkifSwic2VydmljZWFjY291bnQiOnsibmFtZSI6Imhhcnp1biwidWlkIjoiMTRiMjE4MmYtYWM0Mi00NGY1LWI3YjUtZjM0ZGQ0MWM0YzYyIn19LCJuYmYiOjE2ODU3NDMyNTQsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDpjYXJ0czpjYXJ0cyJ9.EgXzAmGZcpbn2Ejpdn4OyxI5k4eRGtFcVGxwhyWd1akOB5AsvnEKGEv4b9XE1fC2Ycvn0F8Tg3GcXbL3lAfnvcMvKSeUtbfarBETJg_xhkkZHauYkLomnggdJxKbq2QKzdf8B7bMjYeitX6Eu5HSynZdu3XWRc-ZBdp-TnB7zowW9vygXMdnZC7p0CB4jMN5IwqwtH4-8EGTiQgvedZ4vaTV5KpBlV1HMZou7NXi0mq1nPinZ6CO2hf4X6N1FPMxzRlax29OfAfmp94bh9wRsaHGP8qeTZOrEoulV9HWJaCZFoMIDsZX-6jyBzg6M91uMGrRxzS5aq1XHFyjEEsVlg
```


HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "RS256",
  "kid": "49d4f12d5187af381a23ae540a66255556dff2bc"
}
```


PAYLOAD: DATA


```
{
  "aud": [
    "sts.amazonaws.com"
  ],
  "exp": 1685829654,
  "iat": 1685743254,
  "iss": "https://oidc.eks.us-east-2.amazonaws.com/id/9278920BDA929A8A54E92E18A4DECB2A",
  "kubernetes.io": {
    "namespace": "carts",
    "pod": {
      "name": "carts-f8f778bdc-nwb6v",
      "uid": "4a4a4373-0751-4b03-afa8-2efbab930c09"
    },
    "serviceaccount": {
      "name": "carts",
      "uid": "14b2182f-ac42-44f5-b7b5-f34dd41c4c62"
    }
  },
  "nbf": 1685743254,
  "sub": "system:serviceaccount:carts:carts"
}
```


VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key in SPKI, PKCS #1, X.509 Certificate, or JWK string format.
  ,
  Private Key in PKCS #8, PKCS #1, or JWK string format. The key never leaves your browser.
)
```

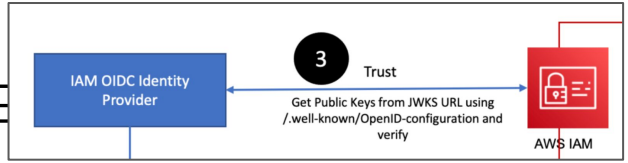
역할 신뢰 정책

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam:[redacted]:oidc-provider/oidc.eks.us-east-2
                    .amazonaws.com/id/9278920BDA929A8A54E92E18A4DECB2A"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-2.amazonaws.com/id/9278920BDA929A8A54E92E18A4DECB2A:sub":
                        "system:serviceaccount:carts:carts"
                }
            }
        }
    ]
}
```

임시 자격증명 발급

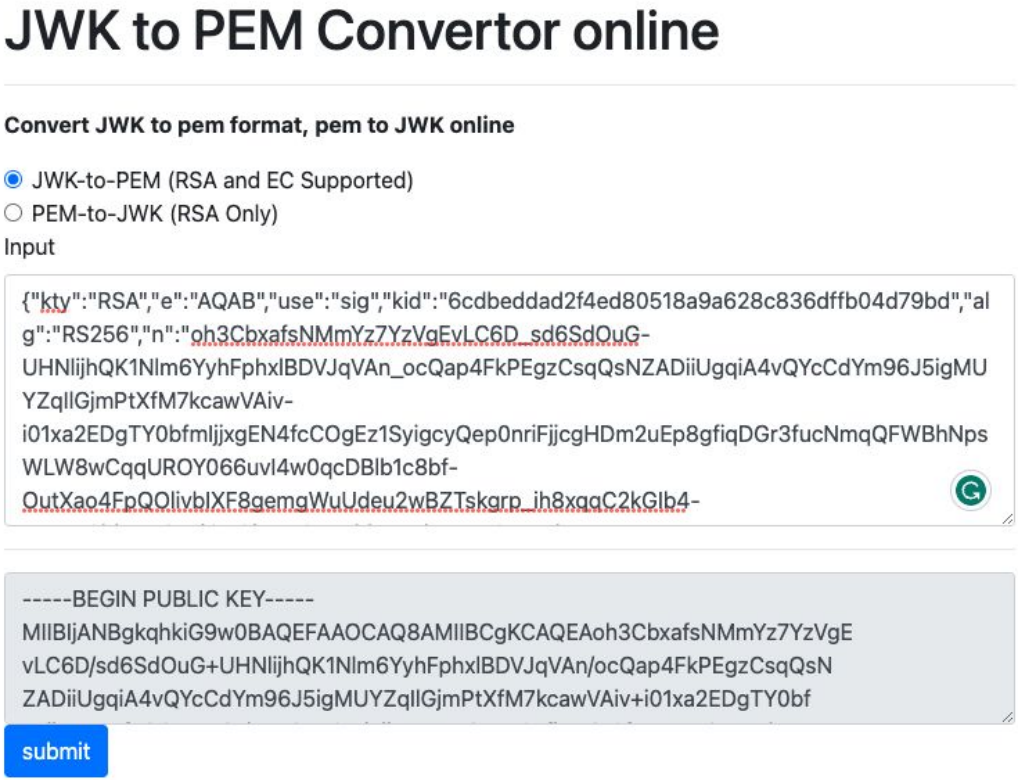
```
rex:~/environment $ aws sts assume-role-with-web-identity \
--role-arn arn:aws:iam::ACCOUNT_ID:role/eks-workshop-carts-dynamo \
--role-session-name test --web-identity-token eyJhbGciOiJSUzI1<SNIP>
{
    "Credentials": {
        "AccessKeyId": "ASIA<SNIP>",
        "SecretAccessKey": "KZ9V<SNIP>",
        "SessionToken": "IQoJb<SNIP>",
        "Expiration": "2023-06-03T01:00:43+00:00"
    },
    "SubjectFromWebIdentityToken": "system:serviceaccount:carts:carts
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAWU22AXDEAGQGMY3XZ:test",
        "Arn": "arn:aws:sts::ACCOUNT_ID:assumed-role/eks-workshop-car
    },
    "Provider": "arn:aws:iam::ACCOUNT_ID:oidc-provider/oidc.eks.us-ea
    "Audience": "sts.amazonaws.com"
}
```

퍼블릭 OIDC 엔드포인트를 통한 토큰 검

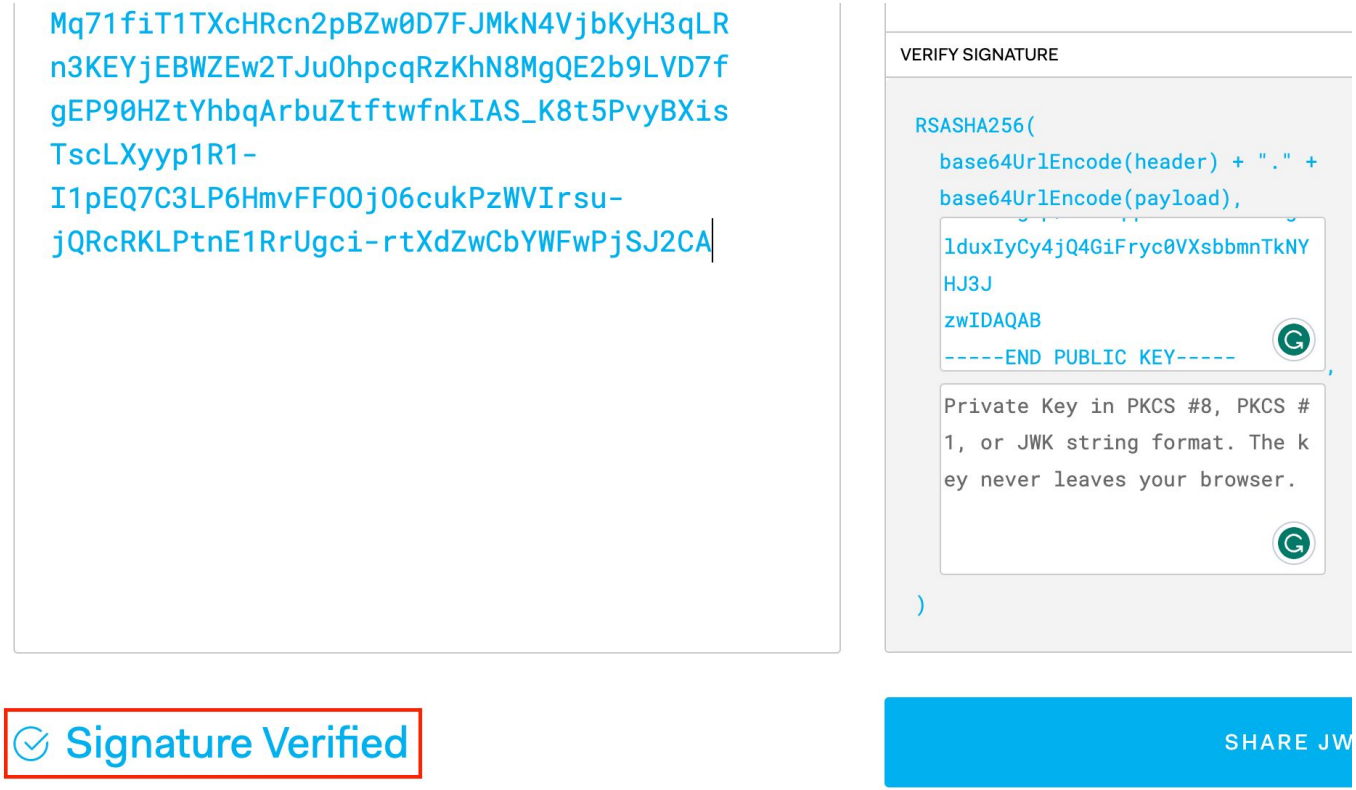


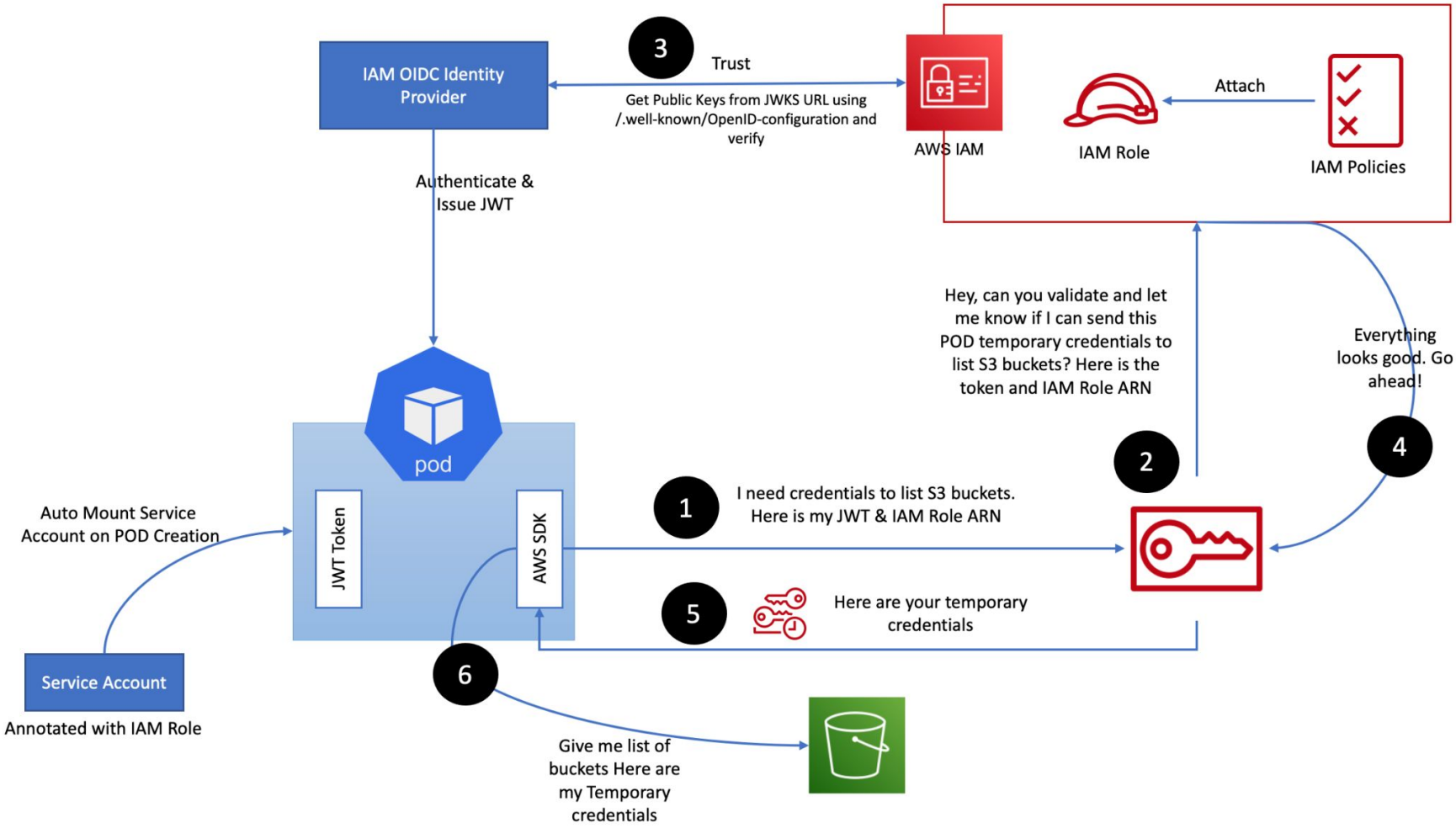

```
curl https://oidc.eks.ap-northeast-2.amazonaws.com/id/23099A8AB33FA460238FE7A20CB79EF6/.well-known/openid-configuration
{"issuer":"https://oidc.eks.ap-northeast-2.amazonaws.com/id/23099A8AB33FA460238FE7A20CB79EF6"
"jwks_uri":"https://oidc.eks.ap-northeast-2.amazonaws.com/id/23099A8AB33FA460238FE7A20CB79EF6/keys","authorization_endpoint":"urn:kubernetes:programmatic_authorization","response_types_supported":["id_token"],"subject_types_supported":["public"],"claims_supported":["sub","iss"],"id_token_signing_alg_values_supported":["RS256"]}

curl https://oidc.eks.ap-northeast-2.amazonaws.com/id/23099A8AB33FA460238FE7A20CB79EF6/keys
{"keys":[{"kty":"RSA","e":"AQAB","use":"sig","kid":"ccfbc0b8c44ddfb9dbaff064a03d9f6c0b32302a",
"alg":"RS256","n":"nsezoviVMf9hRl7nkYunnXFvRT9r-Lhx0zg2p6GEpE-j5nwKSvce1ZT7b78uPSRQXdbVAQ54yYo
DMaozMS82oHgjqX8OIfIhGwS-eM_QPfdE5Rt-7VbG3Hhn0w4C3RGYivmXRd3skLd2baDxX1zpd-RibJewDYDuY33aMUeOi
7bVrYja86_GO-PBX2EK2xtW30xlpXUHr7LnvV1TBXIaeTopeBe4OXMzPYAxTMl1cYwmPXDFJbk6vA1MnSXHl_97N94maLT
dkFI17D0aUWzmtwhxCCWWY19d8KSyfawoJK_l4MPCDkprRXj_UWLAHmMmX-i8ADUsY24hK0YuIzE7CQ"}]}
```

JWK to PEM


JWK to PEM Convertor online
Convert JWK to pem format, pem to JWK online
JWK-to-PEM (RSA and EC Supported)
PEM-to-JWK (RSA Only)
Input
{"kty":"RSA","e":"AQAB","use":"sig","kid":"6cdbeddad2f4ed80518a9a628c836dffb04d79bd","alg":"RS256","n":"oh3CbxafsNMmYz7YzVgEvLC6D_sd6SdOuG-UHNlijhQK1Nlm6YyhFphxlBDVJqVAn_ocQap4FkPEgzCsqQsNZADiiUgqiA4vQYcCdYm96J5igMUYZqIlGjmPtXfM7kcawVAiv-i01xa2EDgTY0bfmljjxgEN4fcCOgEz1SyigcyQep0nriFjjcgHDm2uEp8gfiqDGr3fucNmqQFWBhNpsWLW8wCqqUROY066uvI4w0qcDBlb1c8bf-OutXao4FpQOlivblXF8gemgWuUdeu2wBZTskgrp_jh8xqqC2kGlb4-

JWT 검증


Mq71fiT1TXcHRcn2pBZw0D7FJMkN4VjbKyH3qLR
n3KEYjEBWZEw2TJuOhpcqRzKhN8MgQE2b9LVD7f
gEP90HZtYhbqArbuZtftwfnkIAS_K8t5PvyBXis
TscLXyyp1R1-
I1pEQ7C3LP6HmvFFOOjO6cukPzWVIrsu-
jQRcRKLPtnE1RrUgci-rtXdZwCbYWFwPjSJ2CA
VERIFY SIGNATURE
RSASHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
lduxIyCy4jQ4GiFryc0VXsbbmnTkNY
HJ3J
zwIDAQAB
-----END PUBLIC KEY-----
Private Key in PKCS #8, PKCS #
1, or JWK string format. The k
ey never leaves your browser.
)
Signature Verified
SHARE JW


IAM OIDC Identity Provider
3
Trust
Get Public Keys from JWKS URL using /.well-known/OpenID-configuration and verify
AWS IAM
Attach
IAM Role
IAM Policies
Authenticate & Issue JWT
Hey, can you validate and let me know if I can send this POD temporary credentials to list S3 buckets? Here is the token and IAM Role ARN
Everything looks good. Go ahead!
pod
JWT Token
AWS SDK
Auto Mount Service Account on POD Creation
1
I need credentials to list S3 buckets. Here is my JWT & IAM Role ARN
2
4
5
Here are your temporary credentials
Service Account
Annotated with IAM Role
6
Give me list of buckets Here are my Temporary credentials

안전한 IRSA 환경 구성하기

AssumeRoleWithWebIdentity… 뭔가 빠지지 않았나?

1. 토큰에 명시된 **서비스 계정이 보유**하고 있는 실제 **역할 ARN 과의 비교 과정은?**

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::ACCOUNT_ID:role/ROLE_NAME
  name: rex
  namespace: rex
```

없어요... 있었는데? 아뇨 없어요...

1. AWS 는 **토큰의 변조 유무만 검사**할 뿐, **서비스 계정에 부여된 역할**과 **실제 수임하려는 역할이 같은지 검사하지 않음**
2. 그렇기에, 신뢰 정책에 별(*)이 내리게 되면 취약해짐
   a. **"OIDC_ENDPOINT/id/CLUSTER_ID:sub" : "system:serviceaccount:*:*"**
3. 첫번째 “*” 은 네임스페이스 이며, 두번째 “*” 은 서비스 계정임
4. 최악의 경우, 클러스터 내 토큰을 탈취당하면 위 구문이 포함된 동일한 Federated 주소를 사용중인 역할들을 모두 수임 가능

별이 내릴 때,

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::457065937096:oidc-provider/oidc.eks.us-east-2
                    .amazonaws.com/id/9278920BDA929A8A54E92E18A4DECB2A"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringLike": {
                    "oidc.eks.us-east-2.amazonaws.com/id/9278920BDA929A8A54E92E18A4DECB2A:sub":
                        "system:serviceaccount:*:*"
                }
            }
        }
    ]
}
```

역할도 내려온다

```
rex:~/environment $ aws sts assume-role-with-web-identity \
--role-arn arn:aws:iam::ACCOUNT_ID:role/eks-workshop-carts-dynamo \
--role-session-name test --web-identity-token eyJhbGciOiJSUzI1<SNIP>
{
    "Credentials": {
        "AccessKeyId": "ASIA<SNIP>",
        "SecretAccessKey": "KZ9V<SNIP>",
        "SessionToken": "IQoJb<SNIP>",
        "Expiration": "2023-06-03T01:00:43+00:00"
    },
    "SubjectFromWebIdentityToken": "system:serviceaccount:carts:carts",
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAWU22AXDEAGQGMY3XZ:test",
        "Arn": "arn:aws:sts::ACCOUNT_ID:assumed-role/eks-workshop-carts-
    },
    "Provider": "arn:aws:iam::ACCOUNT_ID:oidc-provider/oidc.eks.us-east-
    "Audience": "sts.amazonaws.com"
}
```

```
rex:~/environment $ aws sts assume-role-with-web-identity \
--role-arn arn:aws:iam::[redacted]role/rextest1234 --role-session-name test
--web-identity-token eyJhbGciOiJSUzI1<SNIP>
{
    "Credentials": {
        "AccessKeyId": "ASIA<SNIP>",
        "SecretAccessKey": "HJ8N<SNIP>",
        "Expiration": "2023-06-03T01:06:31+00:00"
    },
    "SubjectFromWebIdentityToken": "system:serviceaccount:carts:carts",
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAWU22AXDEPTXHKPKQC:test",
        "Arn": "arn:aws:sts::ACCOUNT_ID:assumed-role/rextest1234/test"
    },    "Provider": "arn:aws:iam::ACCOUNT_ID:oidc-provider/oidc.eks.us-east-
    "Audience": "sts.amazonaws.com"
}
```

근데 클러스터 내부에서만 되는거면 그나마 괜찮지 않나요?

1. EKS OIDC 엔드포인트는 공개 (public) 되어 있음
2. 이로 인해, 토큰 탈취 후, 외부에서 편하게 수임 시도 가능
    a. **토큰의 무결성을 검사하는 것이지, 실제로 어디에서 사용하는지는 검사하지 않음**
3. 토큰의 유효기간은 기본적으로 24 시간임 (86400초)

신뢰 정책만 잘 구성하면 될까요?

1. IMDS 의 경우 다양한 정책을 활용하여 쉽게(?) 외부 사용 제한이 가능했음
    a. aws:EC2InstanceSourceVpc
    b. aws:Ec2InstanceSourcePrivateIPv4
2. 하지만, IRSA 는 정책적 지원이 없기 때문에 우리가 잘 만들어서 사용해야 함

```
"Statement": [
    {
        "Effect": "Deny",
        "Action": [
            "*"
        ],
        "Resource": [
            "*"
        ],
        "Condition": {
            "StringNotEquals": {
                "aws:SourceVpc": [
                    "vpc-12341234"
                ]
            },
            "NotIpAddress": {
                "aws:SourceIp": [
                    "123.123.123.123/32"
                ]
            },
            "BoolIfExists": {
                "aws:ViaAWSService": "false"
            }
        }
    },
```

```
{
    "Effect": "Deny",
    "Action": [
        "*"
    ],
    "Resource": [
        "*"
    ],
    "Condition": {
        "NotIpAddress": {
            "aws:VpcSourceIp": [
                "10.1.1.1/32"
            ]
        },
        "Null": {
            "aws:SourceVpc": "false"
        }
    }
}
```


{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Deny",
"Action": "*",
"Resource": "*",
"Condition": {
"Null": {
"aws:SourceVpc": "false",
"ec2:SourceInstanceARN": "false"
},
"StringNotEquals": {
"aws:SourceVpc": "vpc-12341234"
},
"BoolIfExists": {
"aws:ViaAWSService": "false"
}
}
}
]
}

EKS Pod Identity

EKS Pod Identity?

1. IRSA와 달리 **외부 단일 지점**에서 Pod이 역할을 사용할 수 있도록 하는 기능
2. 기존에 ECS에서 사용하던 방식을 도입
3. 다만, Kubernetes 1.24 버전 이상을 사용해야 함
    a. ~~기존 고객들이 1.23에서 잘 안올리려고 해서 그런게 아닐까~~

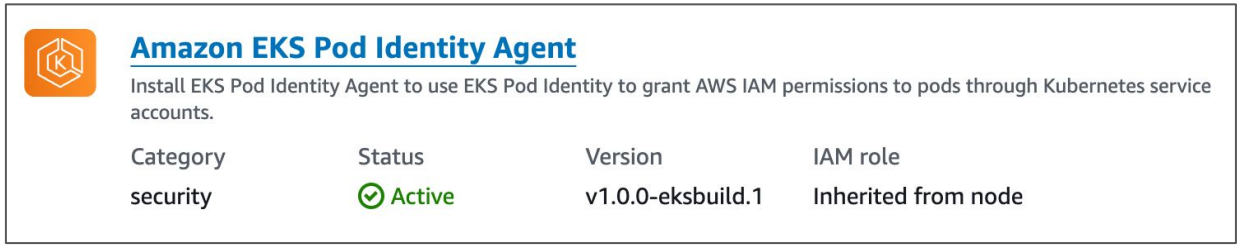
왜 출시한걸까? 뭐가 좋아진거야?

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

1. 버전 업그레이드 시 고충 개선
    a. 신규 버전의 클러스터를 구성
    b. IRSA를 통해 사용하던 기존 역할의 신뢰정책 전부 수정
        i. 신규 클러스터의 OIDC Provider 생성 및 기존 역할 신뢰정책에 추가
    c. 마이그레이션 진행
        i. **끔찍하리만큼 불편**하고, 역할을 정확히 파악하지 못하면 대참사 발생
2. 보안성 향상
    a. IRSA에서는 신뢰정책에 별(*)을 많이 쓰게 됨 (아무래도 불편하다 생각하다보니..)
3. 사용성 향상
    a. (기존) 서비스 계정의 **annotations** / 역할의 **신뢰정책**
    b. (개선) Pod Identity 등록

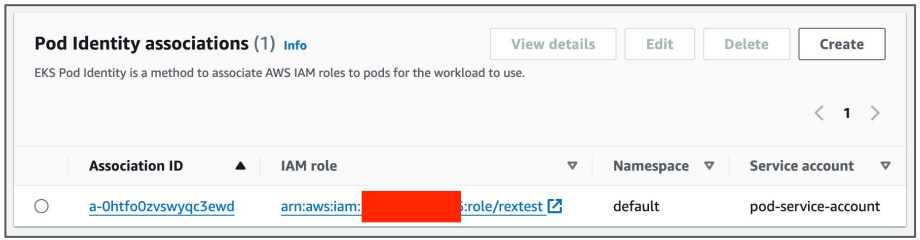
왜 출시한걸까? 뭐가 좋아진거야?

1. add-on만 추가해주면 됨



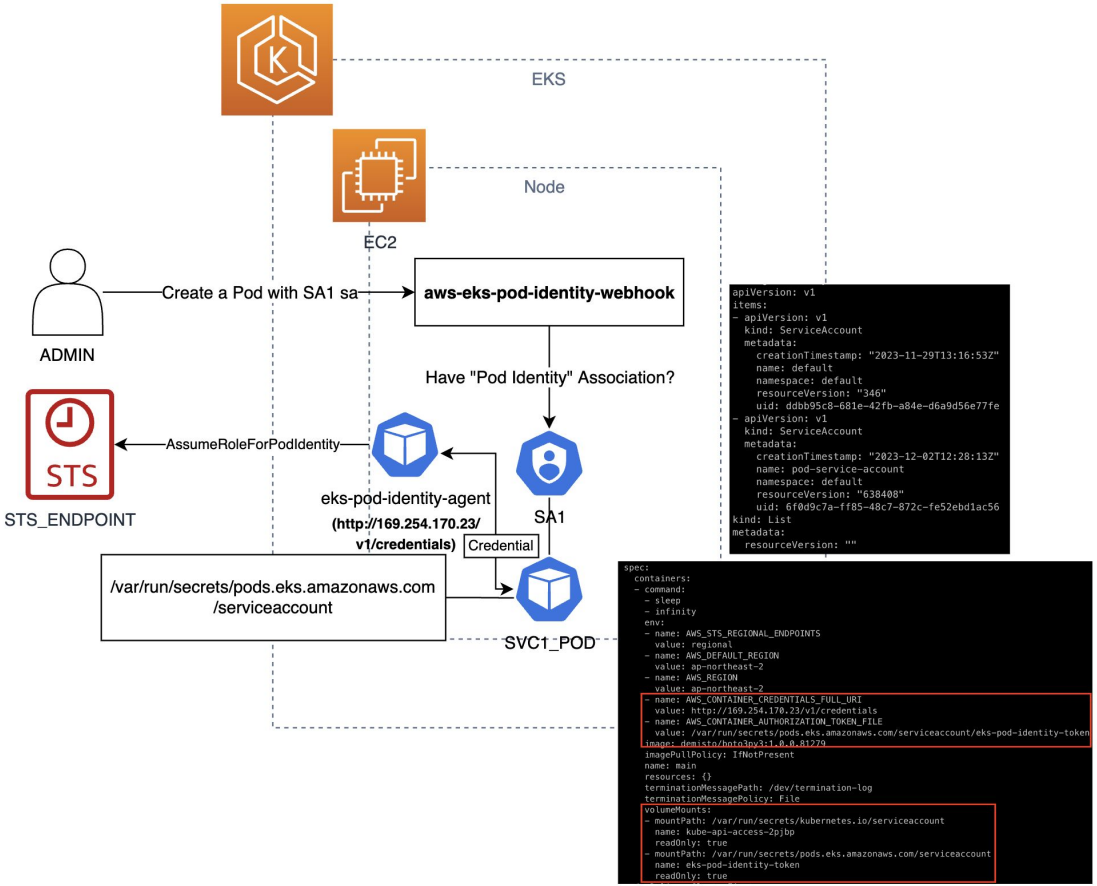

```
NAME                  DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
aws-node              2         2         2       2            2           <none>
eks-pod-identity-agent 2        2         2       2            2           <none>
kube-proxy            2         2         2       2            2           <none>
```

```
NAME                          READY   STATUS    RESTARTS
aws-node-fnj8n                1/1     Running   0
aws-node-zbffm                1/1     Running   0
coredns-6d47776b78-hrlnf      1/1     Running   0
coredns-6d47776b78-xkcsg      1/1     Running   0
eks-pod-identity-agent-957rz  1/1     Running   0
eks-pod-identity-agent-rp26v  1/1     Running   0
kube-proxy-9g4dm              1/1     Running   0
kube-proxy-fbg2n              1/1     Running   0
```



eks-pod-identity-agent

동작방식


EKS
Node
EC2
ADMIN
Create a Pod with SA1 sa
aws-eks-pod-identity-webhook
Have "Pod Identity" Association?
AssumeRoleForPodIdentity
STS
STS_ENDPOINT
eks-pod-identity-agent
(http://169.254.170.23/
v1/credentials)
Credential
SA1
/var/run/secrets/pods.eks.amazonaws.com
/serviceaccount
SVC1_POD
apiVersion: v1
items:
- apiVersion: v1
kind: ServiceAccount
metadata:
creationTimestamp: "2023-11-29T13:16:53Z"
name: default
namespace: default
resourceVersion: "346"
uid: ddbb95c8-681e-42fb-a84e-d6a9d56e77fe
- apiVersion: v1
kind: ServiceAccount
metadata:
creationTimestamp: "2023-12-02T12:28:13Z"
name: pod-service-account
namespace: default
resourceVersion: "638408"
uid: 6f0d9c7a-ff85-48c7-872c-fe52ebd1ac56
kind: List
metadata:
resourceVersion: ""
spec:
containers:
- command:
- sleep
- infinity
env:
- name: AWS_STS_REGIONAL_ENDPOINTS
value: regional
- name: AWS_DEFAULT_REGION
value: ap-northeast-2
- name: AWS_REGION
value: ap-northeast-2
- name: AWS_CONTAINER_CREDENTIALS_FULL_URI
value: http://169.254.170.23/v1/credentials
- name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
value: /var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
image: demista/hnto3ny3:1.0.0.81279
imagePullPolicy: IfNotPresent
name: main
resources: {}
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
volumeMounts:
- mountPath: /var/run/secrets/kubernetes.io/serviceaccount
name: kube-api-access-2pjbp
readOnly: true
- mountPath: /var/run/secrets/pods.eks.amazonaws.com/serviceaccount
name: eks-pod-identity-token
readOnly: true

169.254.170.23?

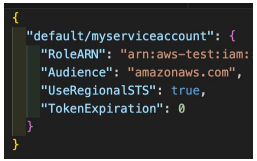
1. 링크 로컬 주소로, EC2의 IMDS와 유사한 역
2. 어떤 방식으로 접근이 되는걸까?
   a. 이제 **모든 EC2가 해당 주소를 오픈**하고 있나요?
      i. 놉!
   b. Capabilities와 네트워크 인터페이스의 조합임
      i. **80포트 오픈 시, 해당 인터페이스에 바인딩!**

```
  ports:
  - containerPort: 80
    hostPort: 80
    name: proxy
    protocol: TCP
  - containerPort: 2703
    hostPort: 2703
    name: probes-port
    protocol: TCP
  readinessProbe:
    failureThreshold: 30
    httpGet:
      host: localhost
      path: /readyz
      port: probes-port
      scheme: HTTP
    initialDelaySeconds: 1
    periodSeconds: 10
    successThreshold: 1
    timeoutSeconds: 10
  resources: {}
  securityContext:
    capabilities:
      add:
      - CAP_NET_BIND_SERVICE
  terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
  volumeMounts:
  - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
    name: kube-api-access-m8k6j
    readOnly: true
dnsPolicy: ClusterFirst
enableServiceLinks: true
hostNetwork: true
```

```
pod-id-link0: flags=195<UP,BROADCAST,RUNNING,NOARP>  mtu 1500
        inet 169.254.170.23  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fd00:ec2::23  prefixlen 128  scopeid 0x0<global>
        inet6 fe80::e890:e5ff:fe7f:80c1  prefixlen 64  scopeid 0x20<link>
        ether ea:90:e5:7f:80:c1  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 78  bytes 5460 (5.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

근데 맵핑은 어디서 관리하고 판단하나요?

1. 23년 8월에 이미 amazon-eks-pod-identity-webhook에 기능이 추가됨
   a. --watch-config-file=<path>

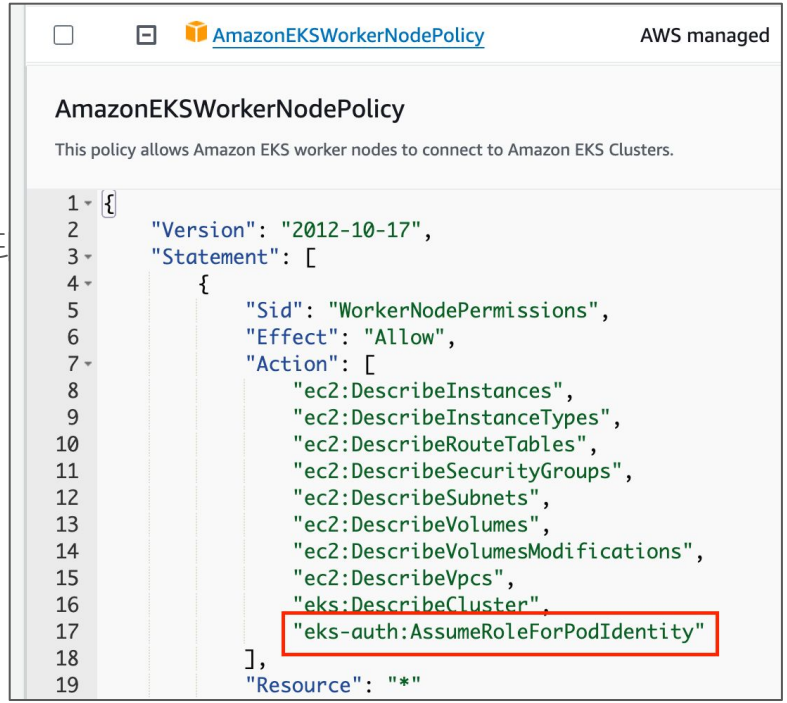


2. **(여기서부턴 추정)**
   a. 컨트롤 플레인에서 해당되는 **"네임스페이스"**와 **"서비스 계정"**들에 대한 설정 저장 (+ 역할 등)
      i. 토큰을 주입하고, 환경변수에 등록해줘야 하는 대상을 알아야
   b. 이후, SDK 또는 AWS CLI 또는 엔드포인트 직접 질의 시,
   c. 내부적으로 별도의 확인을 거쳐,
   d. 역할에 대한 자격증명을 획득 및 반환하는 것으로 보임.
   e. **(주의) 서비스 계정에 annotations 존재하면 우선순위로 인해 적용 ?**

마이그레이션 시 주의할 점

주의할 점

1. 사용중인 SDK 버전 확인!
2. 워커노드의 정책 확인
   a. 기본적으로 관리형 정책을 사용할 것으로 예상도
   b. 우측과 같이 권한이 추가 됐음.
3. 혹시!
   a. 보안 솔루션으로 인한 링크 로컬 주소 생성 불가
   b. 이미 내부적으로 사용중인 링크 로컬 주소인지,
   c. iptables에 별도의 규칙이 있는지
   d. 등을 꼭 확인할 것!!



보안은?

AssumeRoleForPodIdentity

PDF

The Amazon EKS Auth API and the `AssumeRoleForPodIdentity` action are only used by the EKS Pod Identity Agent.

We recommend that applications use the AWS SDKs to connect to AWS services; if credentials from an EKS Pod Identity association are available in the pod, the latest versions of the SDKs use them automatically.

Request Syntax

```
POST /clusters/clusterName/assume-role-for-pod-identity HTTP/1.1
Content-type: application/json

{
   "token": "string"
}
```

Q & A ?

관리형 클러스터의 “구조적" 취약점

왜 “구조적" 취약점 일까?

관리형 클러스터들은,

1. 기존 클라우드 인프라와 연계가 돼야 함
2. 예를 들면,
    a. Pod 이 생성되면, VPC 에 ENI 를 생성하고 붙여줘야 함
    b. 삭제되면, 같이 삭제해야 함
    c. 마이그레이션과 같은 작업이 있으면 분리/결합 수행 필요
    d. 관리를 위한 태그 부여
    e. 이를 위한 정보 조회 등

그렇기 때문에 다양한 권한이 필요함 - 1

AmazonEC2ContainerRegistryReadOnly

Provides read-only access to Amazon EC2 Container Registry repositories.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetRepositoryPolicy",
                "ecr:DescribeRepositories",
                "ecr:ListImages",
                "ecr:DescribeImages",
                "ecr:BatchGetImage",
                "ecr:GetLifecyclePolicy",
                "ecr:GetLifecyclePolicyPreview",
                "ecr:ListTagsForResource",
                "ecr:DescribeImageScanFindings"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonEKS_CNI_Policy

This policy provides the Amazon VPC CNI Plugin (amazon-vpc-cni-k8s) the permissions it re

복사

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignPrivateIpAddresses",
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes",
                "ec2:DetachNetworkInterface",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:UnassignPrivateIpAddresses"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
```

그렇기 때문에 다양한 권한이 필요함 - 2

AmazonEKSWorkerNodePolicy

This policy allows Amazon EKS worker nodes to connect to Amazon EKS Clusters.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVolumes",
                "ec2:DescribeVolumesModifications",
                "ec2:DescribeVpcs",
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

AmazonSSMManagedInstanceCore

The policy for Amazon EC2 Role to enable AWS Systems Manager service core functionality.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeAssociation",
                "ssm:GetDeployablePatchSnapshotForInstance",
                "ssm:GetDocument",
                "ssm:DescribeDocument",
                "ssm:GetManifest",
                "ssm:GetParameter",
                "ssm:GetParameters",
                "ssm:ListAssociations",
                "ssm:ListInstanceAssociations",
                "ssm:PutInventory",
                "ssm:PutComplianceItems",
                "ssm:PutConfigurePackageResult",
                "ssm:UpdateAssociationStatus",
                "ssm:UpdateInstanceAssociationStatus",
                "ssm:UpdateInstanceInformation"
            ],
            "Resource": "*"
        },
```

다시 IMDS (Instance MetaData Service)

```
response = client.modify_network_interface_attribute(
    Attachment={
        'AttachmentId': 'string',
        'DeleteOnTermination': True|False
    },
    Description={
        'Value': 'string'
    },
    DryRun=True|False,
    Groups=[
        'string',
    ],
```

1. Pod 에서도 메타데이터 서비스에 접근이 가능함
2. 이를 통해,
   a. ECR 을 통한 **악성 이미지 다운로드 및 모든 정보 수집** 가능
   b. **EC2, 보안그룹 등 다양한 정보 수집** 가능
   c. **ENI 에 대한 서비스 거부 공격** 가능
3. 이외에도 시스템 상황에 따라,
   a. 노드 역할을 통한 kubernetes 접근 가능
   b. 서비스 계정을 통한 권한 상승
   c. 결과적으로 AWS 전체 장악 가능

```
└─ k exec -it irsa -- sh
sh-4.2# curl http://169.254.169.254/latest/meta-data/iam/security
{
  "Code" : "Success",
  "LastUpdated" : "2023-07-02T00:05:21Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAW
  "SecretAccessKey" : "N
  "Token" : "IQoJb3JpZ2l
////////8BEAEaDDQ1NzA2N
5g9w/R2AjULcwLD9FZoAfB3q
S/mpi30GIYBYbsUJzqEtKDeR
NoQDrsIH73gLSzW1FS2c9sCM
g45vNZhEGKsISFmrSO474dc2
XvwXSFB6g+9QaqRqW+pBc7sn
pFMDhTw1RvDX+rWMJP+gqUGO
Ioq8lp73RLMMvtjkFcXJouuX
  "Expiration" : "2023-07-02T06:24:15Z"
```

안전하게 사용하기

자격증명 반환 홉수 제한 (컨테이너 사용 시 매우 주의)

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "MaxImdsHopLimit",
        "Effect": "Deny",
        "Action": "ec2:RunInstances",
        "Resource": "arn:aws:ec2:*:*:instance/*",
        "Condition": {
            "NumericGreaterThan": {"ec2:MetadataHttpPutResponseHopLimit": "1"}
        }
    }
}
```

1. 홉을 1로 제한한다면?

1. SCP를 통한 가장 간단하고 효율적인 해결 방법
2. **호스트 네트워크**를 사용 중인 Pod 을 제외하고, **IMDS 접근 불가**
3. k get pod

   -o=custom-columns=NS:.metadata.namespace,NODE:.spec.nodeName,NAM

   E:.metadata.name,HOST:.spec.hostNetwork --all-namespaces

홉을 1로 제한한다면?

NS	NODE	NAME	HOST
amazon-guardduty	ip-192-168-64-16.ap-northeast-2.compute.internal	aws-guardduty-agent-wxzrl	true
calico-apiserver	ip-192-168-64-16.ap-northeast-2.compute.internal	calico-apiserver-544496f4b7-5xxbf	<none>
calico-apiserver	ip-192-168-64-16.ap-northeast-2.compute.internal	calico-apiserver-544496f4b7-jxkcp	<none>
calico-system	ip-192-168-64-16.ap-northeast-2.compute.internal	calico-kube-controllers-6b8fb97d67-fwgk6	<none>
calico-system	ip-192-168-64-16.ap-northeast-2.compute.internal	calico-node-jbncg	true
calico-system	ip-192-168-64-16.ap-northeast-2.compute.internal	calico-typha-866b9f78c4-5j4ql	true
calico-system	ip-192-168-64-16.ap-northeast-2.compute.internal	csi-node-driver-hht2l	<none>
default	ip-192-168-64-16.ap-northeast-2.compute.internal	irsa	<none>
kube-system	ip-192-168-64-16.ap-northeast-2.compute.internal	aws-node-bphnd	true
kube-system	ip-192-168-64-16.ap-northeast-2.compute.internal	coredns-dc4979556-qplx9	<none>
kube-system	ip-192-168-64-16.ap-northeast-2.compute.internal	coredns-dc4979556-vx6wj	<none>
kube-system	ip-192-168-64-16.ap-northeast-2.compute.internal	kube-proxy-gnk9q	true
tigera-operator	ip-192-168-64-16.ap-northeast-2.compute.internal	tigera-operator-7585d47b69-2kvq5	true

2. NetworkPolicy 사용

1. calico 를 활용하여 네트워크 정책 추가 가능
    a. AWS 공식 문서에 소개되어 있는 내용이며, 이외에도 다양한 오픈소스 활용 가능
2. 다만, 몇 가지 단점 존재
    a. 네트워크 정책으로 인한 클러스터 부담
    b. 클러스터 생성 시에 항상 챙겨야 함
3. 그럼에도, SCP 로 일괄로 적용하는 것 보단 라벨을 이용하거나 여러가지 정책을 활용해 조금 더 유연하게 정책 적용 가능한 장점도 존재

2. NetworkPolicy 사용

```
apiVersion: projectcalico.org/v3
kind: GlobalNetworkPolicy
metadata:
  name: block-imds-access-for-all
spec:
  selector: all()
  types:
  - Egress
  egress:
  - action: Deny
    destination:
      nets: [169.254.169.254/32]
  - action: Allow
    destination:
      notNets: [169.254.169.254/32]
EOF
```

Q & A