

12. k8s의 인증 & 인가

I k8s의 인증

- k8s의 자체적인 인증 기능 없음
 - k8s 클러스터를 운영하는 환경의 인증 기능을 적용하라는 것
- 예시
 - 예1) AWS : AWS IAM
 - 예2) GCP : GCP IAM
 - 예3) Onpremise

X.509 Certificate 인증

Webhook 외부 인증 연동, Webhook LDAP 연동

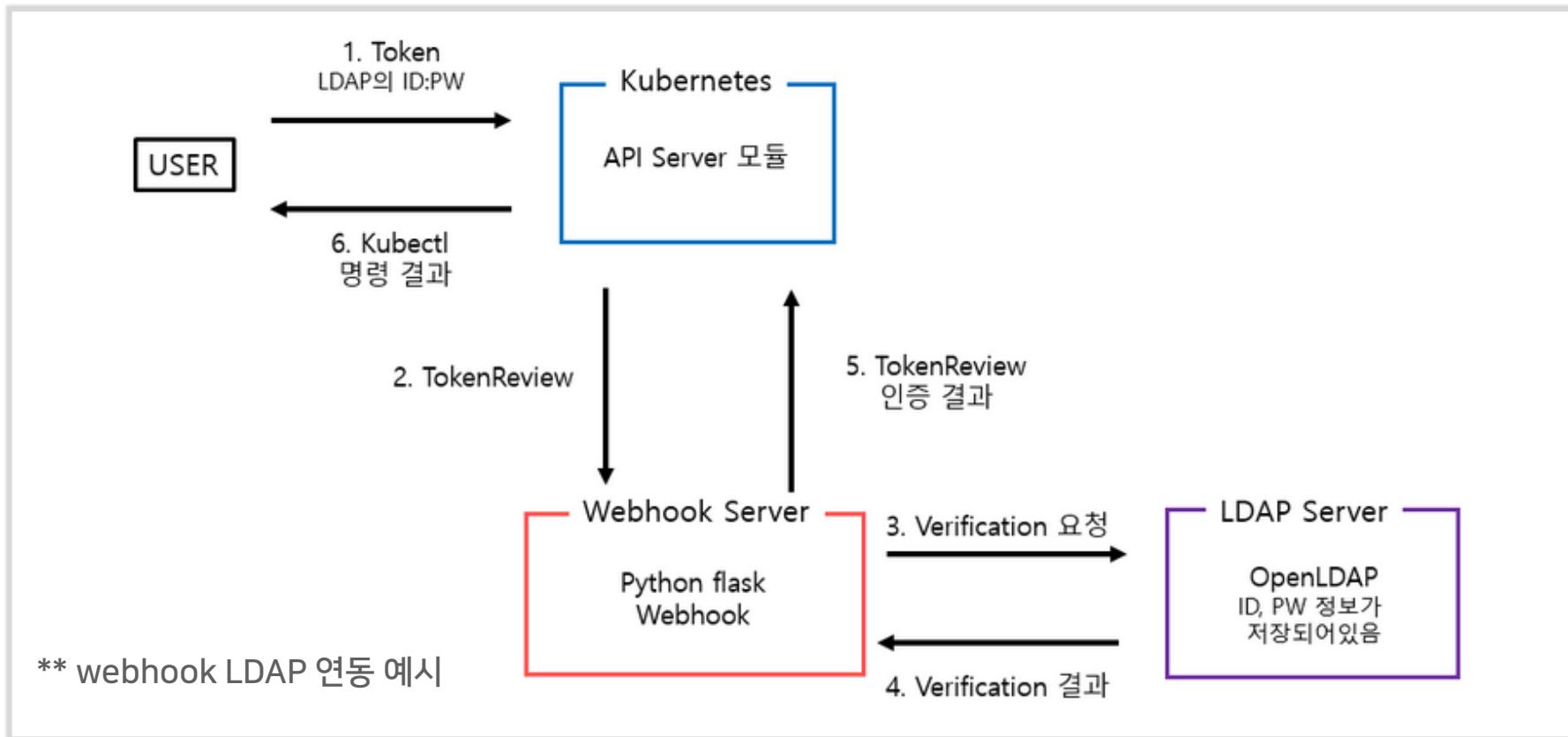
OpenID Connector Provider

- 인증 기능은 없지만 보안주체는 존재함
 - User, Group
 - ServiceAccount

12. k8s의 인증 & 인가

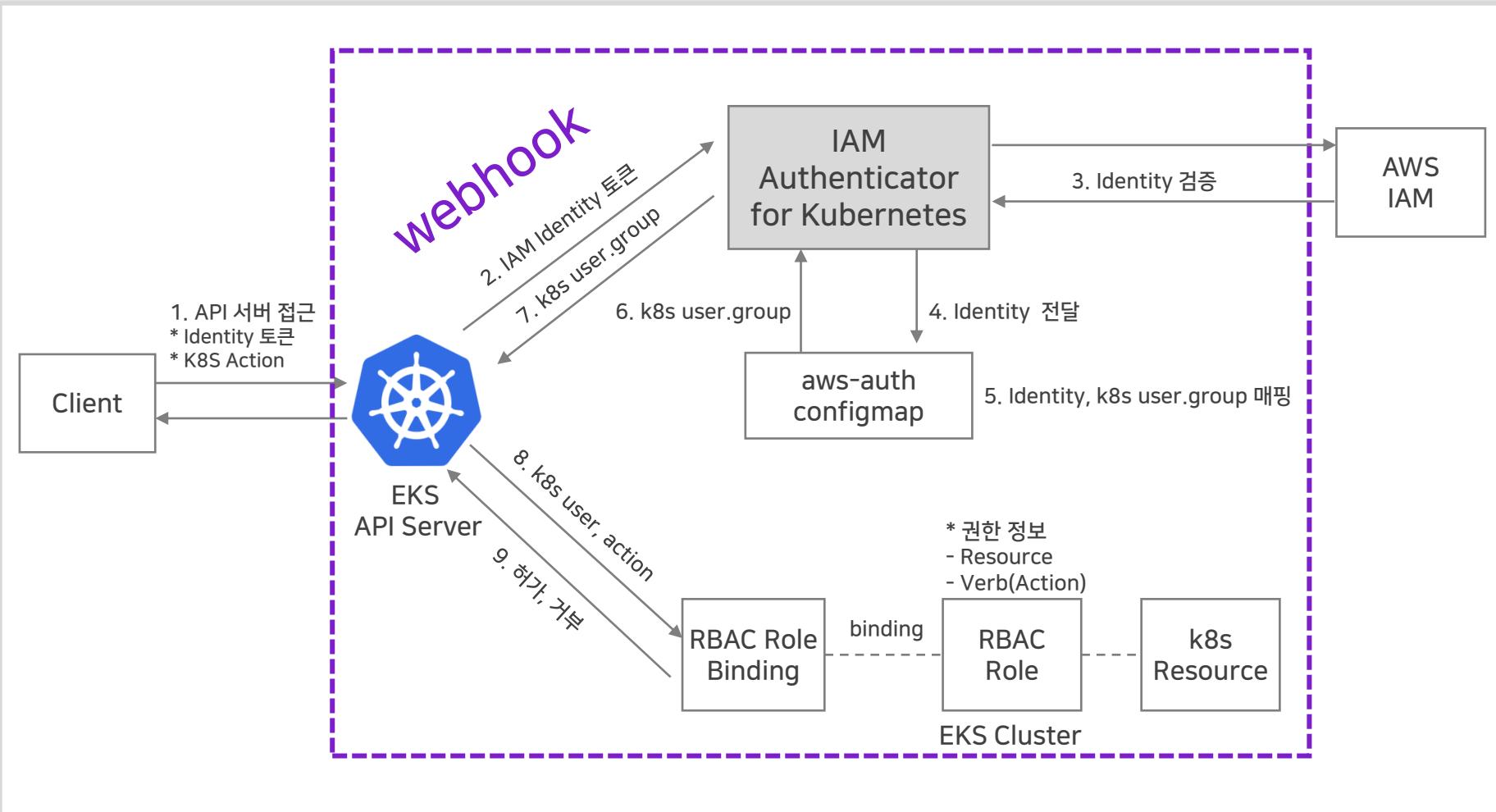
I k8s의 인가 기능

- k8s RBAC 기반 인가
- RBAC Role & Cluster Role



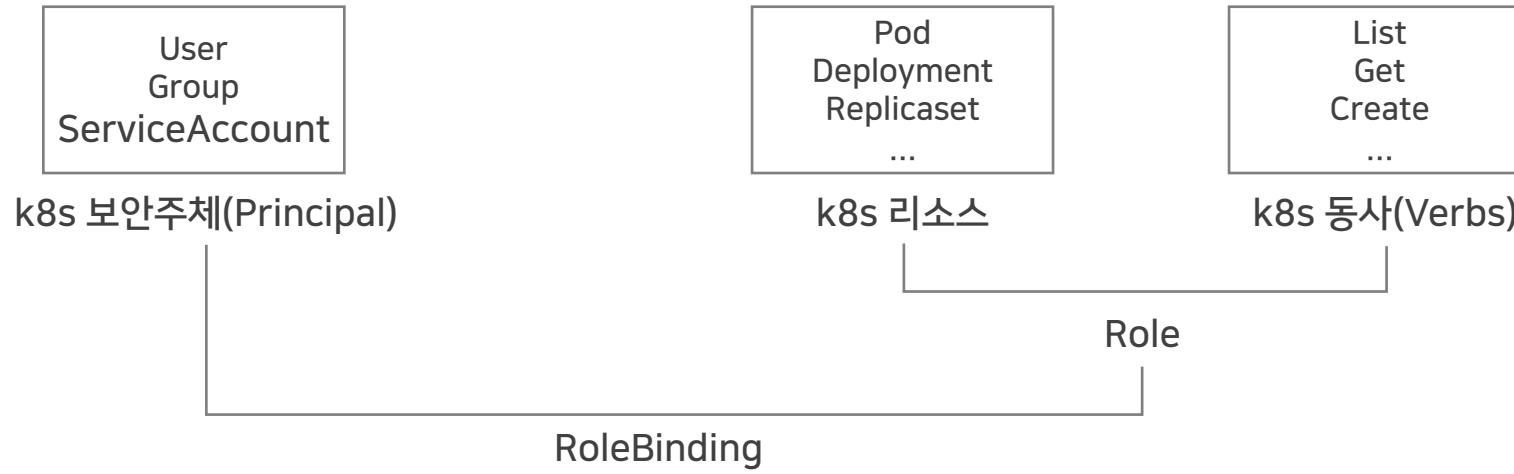
12. k8s의 인증 & 인가

I EKS의 인증 기능 - AWS IAM 연동



12. k8s의 인증 & 인가

I k8s RBAC



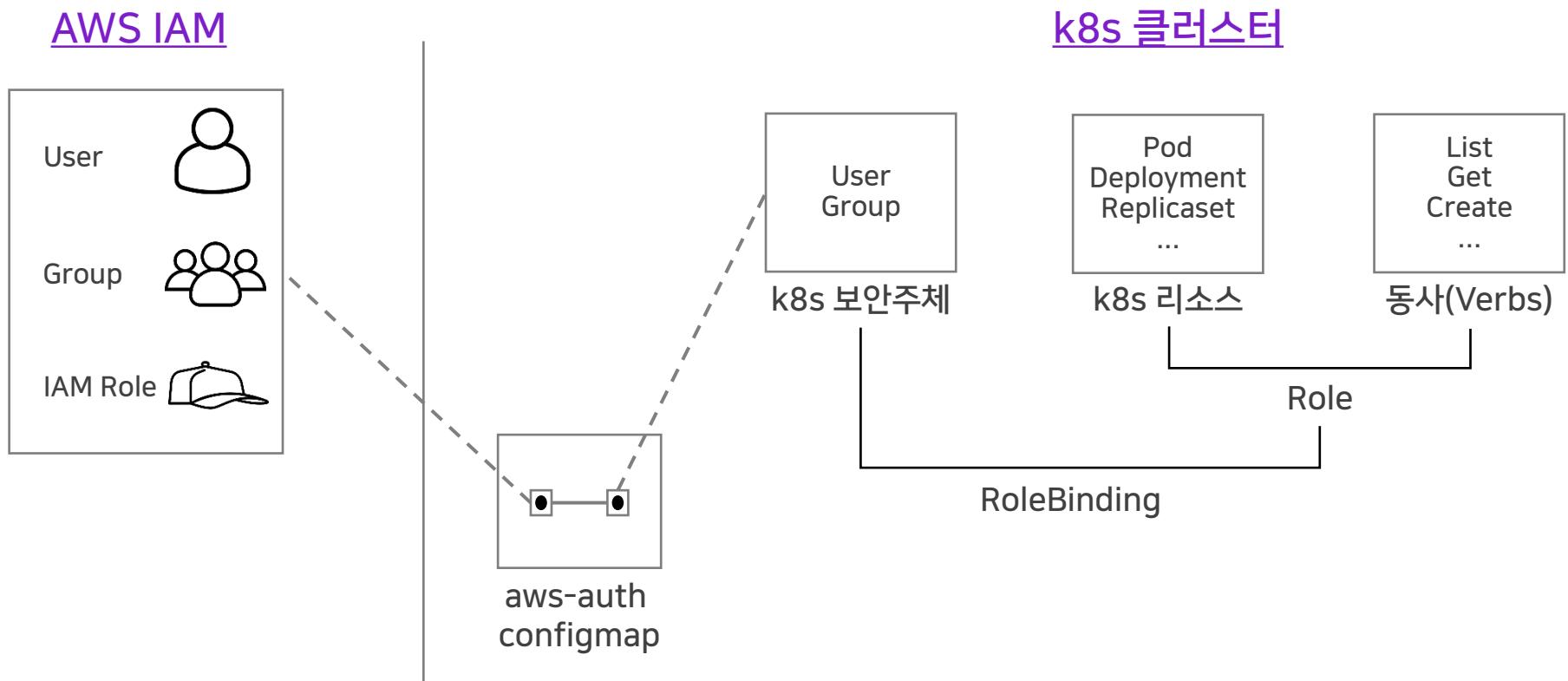
I k8s 인증,인가의 몇가지 유형

- AWS --> k8s
- k8s --> k8s
- k8s --> AWS

12. k8s의 인증 & 인가

I AWS --> k8s

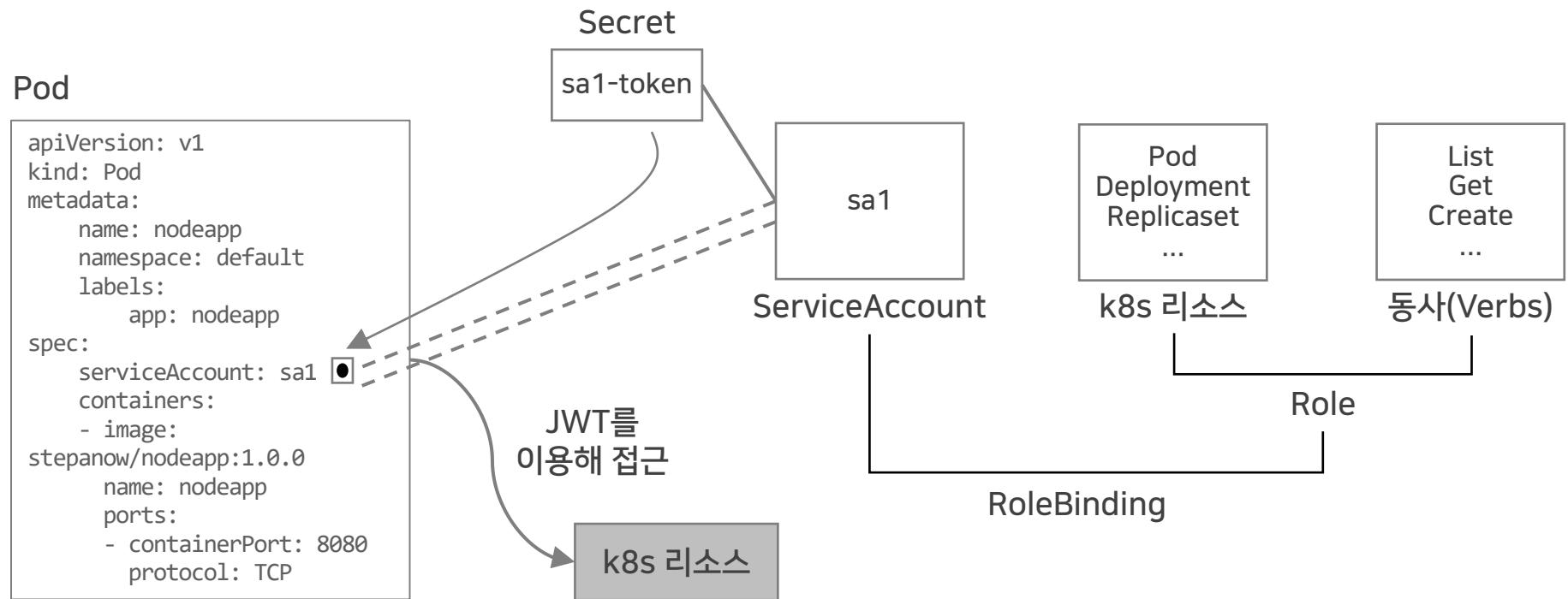
- 예 : AWS IAM Role --> k8s Pod 생성



12. k8s의 인증 & 인가

I k8s --> k8s

- 예) Pod 내의 애플리케이션 --> k8s Pod 생성

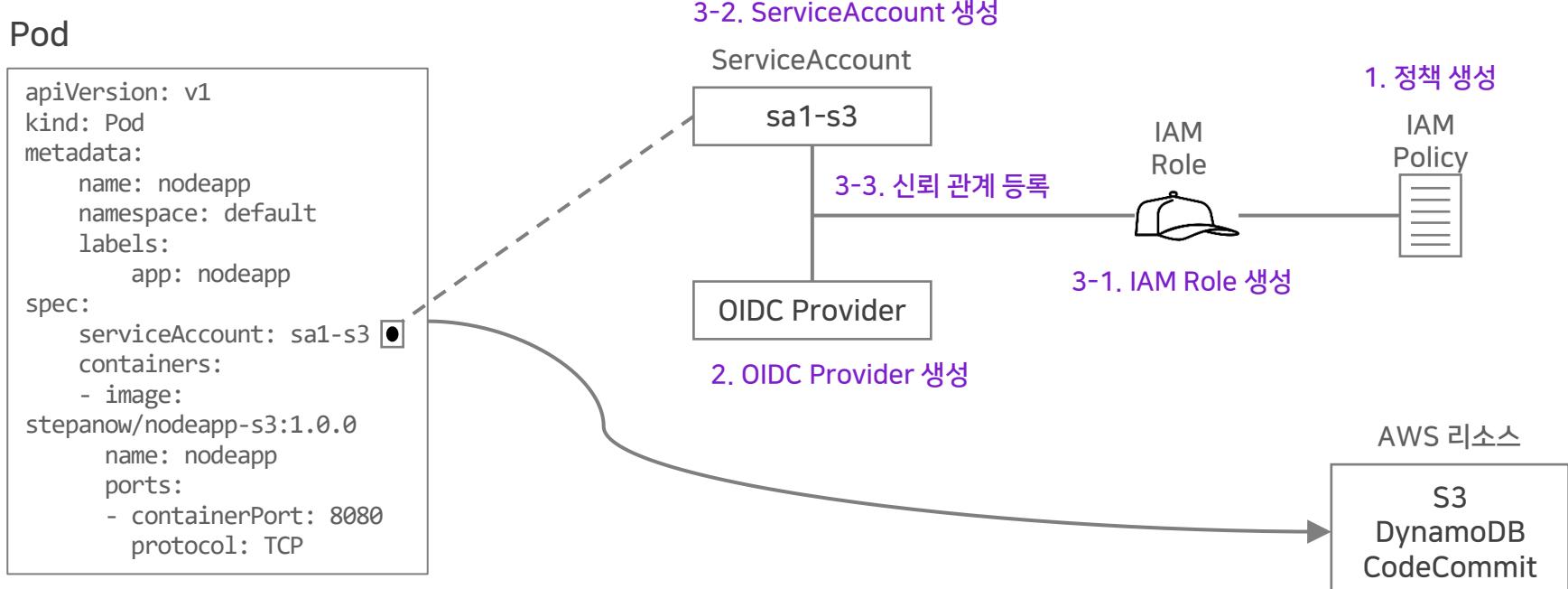


12. k8s의 인증 & 인가

I k8s --> AWS : IRSA(IAM Role for ServiceAccount)

- 예) Pod 내의 애플리케이션 ---> S3와 같은 AWS 리소스 접근

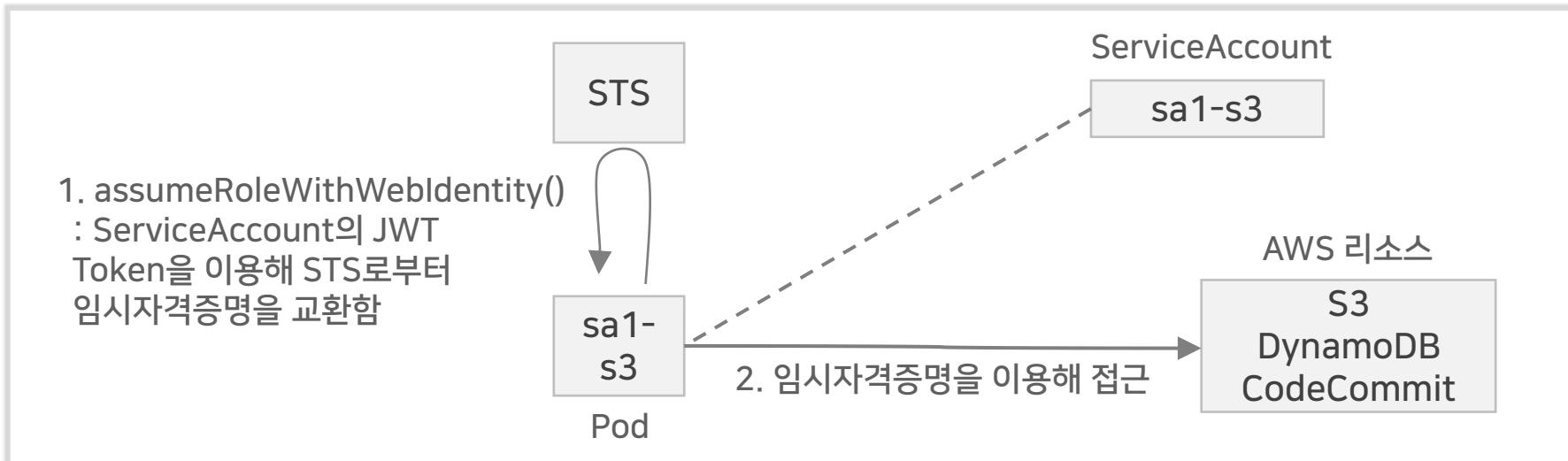
3. eksctl create iamserviceaccount 명령 --> 3-1, 3-2, 3-2 한번에 등록



13. IRSA

I IAM Role for Service Account

- ServiceAccount의 JWT를 IAM Role의 자격증명으로 교환하여 AWS 리소스에 접근 허용하는 방법
- OAuth2를 이용한 Social 인증과 유사한 메커니즘
 - Facebook이 발급한 JWT를 신뢰하는 엔티티로 지정한 IAM Role
 - assumeRoleWithWebIdentity() : JWT --> 임시자격증명으로 교환
- ServiceAccount의 JWT를 신뢰하는 엔티티로 이용할 수 있도록 하기 위해 OIDC Provider 등록이 필요함.



13. IRSA

I 예제 애플리케이션 소개

- stepanowon/nodeapp-s3:1.0.0 : s3 버킷의 객체 접근

```
//--- nodeapp-s3/server.js
import express from "express";
import bodyParser from "body-parser";
import os from 'os';

import { S3Client } from "@aws-sdk/client-s3";
import { GetObjectCommand } from "@aws-sdk/client-s3";

const app = express();

const BUCKET = "k8s-irsa-test";
const OBJECTKEY = "contacts.json";

app.use(bodyParser.json());

app.get("/", (req, res, next) => {
  res.status(200).send(`

    <div>
      <h1>nodeapp-s3</h1>
      <h2> Version: 1.0 </h2>
      <h2> 호스트명 : ${os.hostname()} </h2>
    </div>
  `)
});
```

```
app.get("/contacts", async (req, res, next) => {
  const s3Client = new S3Client({ region: "ap-northeast-2" });
  const bucketParams = { Bucket: BUCKET, Key:OBJECTKEY };

  try {
    const streamToString = (stream) =>
      new Promise((resolve, reject) => {
        const chunks = [];
        stream.on("data", (chunk) => chunks.push(chunk));
        stream.on("error", reject);
        stream.on("end", () =>
          resolve(Buffer.concat(chunks).toString("utf8")));
      });

    const data = await s3Client.send(new
      GetObjectCommand(bucketParams));
    const bodyContents = await streamToString(data.Body);
    res.json(bodyContents);
  } catch (err) {
    console.log("Error", err);
    res.json({ error : err.message });
  }
}

app.listen(8080, () => {
  console.log(`Server is running : PORT 80`);
});
```

13. IRSA

I IRSA 미적용 Pod로 접근 불가 확인

- `kubectl apply -f pod-no-sa.yaml`
 - `kubectl get pods` 명령으로 Pod 생성 확인
- `kubectl exec -it pod-no-sa -- /bin/bash`
 - `curl localhost:8080` --> 액세스 성공
 - `curl localhost:8080/contacts` --> 액세스 거부
- 테스트 완료 후 Pod 삭제
 - `kubectl delete pods pod-no-sa`

```
## pod-no-sa.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-no-sa
  labels:
    app: nodeapp
spec:
  containers:
    - image: stepanow/nodeapp-s3:1.0.0
      imagePullPolicy: Always
      name: nodeapp
      ports:
        - containerPort: 8080
```

```
user00:~/environment/eks/serviceaccount/irsa $ kubectl apply -f pod-no-sa.yaml
pod/pod-no-sa created
user00:~/environment/eks/serviceaccount/irsa $ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
pod-no-sa  1/1     Running   0          8s
user00:~/environment/eks/serviceaccount/irsa $ kubectl exec -it pod-no-sa -- /bin/bash
root@pod-no-sa:/usr/src/app# curl localhost:8080

<div>
  <h1>nodeapp-s3</h1>
  <h2> Version: 1.0 </h2>
  <h2> 호스트명 : pod-no-sa </h2>
</div>
root@pod-no-sa:/usr/src/app# curl localhost:8080/contacts
[{"error": "Access Denied"}root@pod-no-sa:/usr/src/app#
root@pod-no-sa:/usr/src/app#
```

13. IRSA

I OIDC Provider 생성 방법 1

- OIDC Provider URL 확인 후 IAM에서 OIDC Provider 등록

The screenshot shows the 'Add Federated User' wizard in the AWS IAM console. On the left, there is a sidebar with tabs: '개요' (Overview) is selected, followed by '리소스' (Resources), '컴퓨팅' (Computing), '네트워킹' (Networking), '추가 기능' (Additional Features), '인증' (Authentication), '로깅' (Logging), '업데이트 기록' (Update History), and '태그' (Tags). Below these tabs is a section titled '세부 정보' (Detailed Information) containing two tables:

API 서버 엔드포인트	OpenID Connect 공급자 URL
https://3BB82FF706CD7E9956475789BF4373B1.gr7.ap-northeast-2.eks.amazonaws.com	https://oidc.eks.ap-northeast-2.amazonaws.com/id/3BB82FF706CD7E9956475789BF4373B1

인증 기관	클러스터 IAM 역할 ARN
LS0tLS1CR0JTIbDRVJUSUZJQ0FURS0tLS0tCk1JSUMvakNDQWVhZ0F3SUJBZ0lCQUR	arn:aws:iam::242337484181:role/eksctl-demo00-cluster-ServiceRole-1F2S68DONSTFR

A large purple arrow points from the 'OpenID Connect 공급자 URL' field in the sidebar to the '공급자 URL' input field in the main configuration pane.

The main configuration pane has the following sections:

- 자격 증명 공급자 추가**: The title of the wizard.
- 공급자 구성**: A summary of the provider configuration.
- 공급자 유형 정보**: A section where the **OpenID Connect** option is selected.
 - SAML**: Describes SAML 2.0 support.
 - OpenID Connect**: Describes OpenID Connect support, mentioning AWS accounts, Google, and Salesforce.
- 공급자 URL**: Input field containing <https://oidc.eks.ap-northeast-2.amazonaws.com/id/3BB82FF706CD7E9956475789BF4373B1>.
- 대상 정보**: Input field containing `demo00`.

13. IRSA

I OIDC Provider 생성 방법 2

- eksctl utils associate-iam-oidc-provider --cluster demoNN --approve

I IAM Policy 검토(EKSPodToS3Policy)

- arn:aws:iam::111122223333:policy/EKSPodToS3Policy

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:GetObject"  
            ],  
            "Resource": [  
                "arn:aws:s3::::k8s-irsa-test/*"  
            ]  
        }  
    ]  
}
```

13. IRSA

I eksctl create iamserviceaccount 명령

- 입력 : IAM Policy, ServiceAccount명, 클러스터
- 생성 : ServiceAccount, IAM Role, 신뢰관계 --> 6페이지 전 그림 참조
- 다음 명령에서 bold 처리된 부분만 변경

```
eksctl create iamserviceaccount \
--name sa1-s3-demoNN \
--namespace default \
--cluster demoNN \
--attach-policy-arn arn:aws:iam::111122223333:policy/EKSPodToS3Policy \
--approve \
--override-existing-serviceaccounts
```

I ServiceAccount, Token 확인

- kubectl get sa sa1-s3-demoNN
- kubectl create token sa1-s3-demoNN

```
user00:~/environment/eks/serviceaccount/irsa $ kubectl get sa sa1-s3-demo00
NAME      SECRETS   AGE
sa1-s3-demo00  0        6m29s
user00:~/environment/eks/serviceaccount/irsa $ kubectl create token sa1-s3-demo00
eyJhbGciOiJSUzI1NiIsImtpZCI6ImFjMDJjNTAwYzcxZTdmMDUyOWVlMGYxYTkxMTczODlmMzzhNWI5Mj(
F1bHQuc3ZjI10sImV4CCI6MTY4Nji3NDc0NCwiaWF0IjoxNjg2MjcxMTQ0LCJpc3MiOiJodHRwczovL29p;
vaWQvM0JC0DJGRjcwNkNEN0U5OTU2NDc1Nzg5QkY0MzczQjEiLCJrdWJlcml5ldGVzLmlvIjp7Im5hbWVzc(
YW1lIjoic2ExLXMzLWR1bW8wMCIsInRpZCI6ImJmNDUyODdiLWM3NDAtNGMyYi04NGY3LWMzY2M20TQwWE(
W06c2VydmIjZWfjY291bnQ6ZGVmYXVsDpzYTEtczMTZGvtbzAwIn0.Gdgjnt3F07iexXkg1fu35EYB4sd\(
ENRHQPUpkAz821fI2401bEBKJDKzB2PN7a5QmM2MOD1KLeorguPYBkm58LfRtaK05646uop4u_uD-8Flb(
o026Aty0d113dYBT6-E89bKRDC1D8wmx9Zy-W701DXEWCFUMKK4SJj-FGoh2WMy8b4YSimPWZMpCP95Kd(
r7u6Cjd6Zg
```

13. IRSA

I Service Account를 사용하는 Pod 생성 후 접근 여부 확인

- kubectl apply -f pod-sa.yaml
- kubectl exec -it pod-sa -- /bin/bash
- 다음 명령 실행 : 둘 다 성공
 - curl localhost:8080
 - curl localhost:8080/contacts

```
user00:~/environment/eks/serviceaccount/irsa $ kubectl apply -f pod-sa.yaml
pod/pod-sa created
user00:~/environment/eks/serviceaccount/irsa $ kubectl exec -it pod-sa -- /bin/bash
root@pod-sa:/usr/src/app# curl localhost:8080

<div>
  <h1>nodeapp-s3</h1>
  <h2> Version: 1.0 </h2>
  <h2> 호스트명 : pod-sa </h2>
</div>
root@pod-sa:/usr/src/app# curl localhost:8080/contacts
"[{"no":1569603605606,"name":"Keandra Lee","tel":"010-3456-8299","address":"서울시"}, {"no":1569603605605,"name":"Katherine White","tel":"010-3456-8298","address":"서울시"}, {"no":1569603605604,"name":"Katchi Wilson","tel":"010-3456-8297","address":"서울시"}, {"no":1569603605603,"name":"Kathy Miller","tel":"010-3456-8296","address":"서울시"}]"
```

```
## pod-sa.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-sa
  labels:
    app: nodeapp
spec:
  serviceAccountName: sa1-s3-demonN
  containers:
    - image: stepanow/nodeapp-s3:1.0.0
      imagePullPolicy: Always
      name: nodeapp
      ports:
        - containerPort: 8080
```

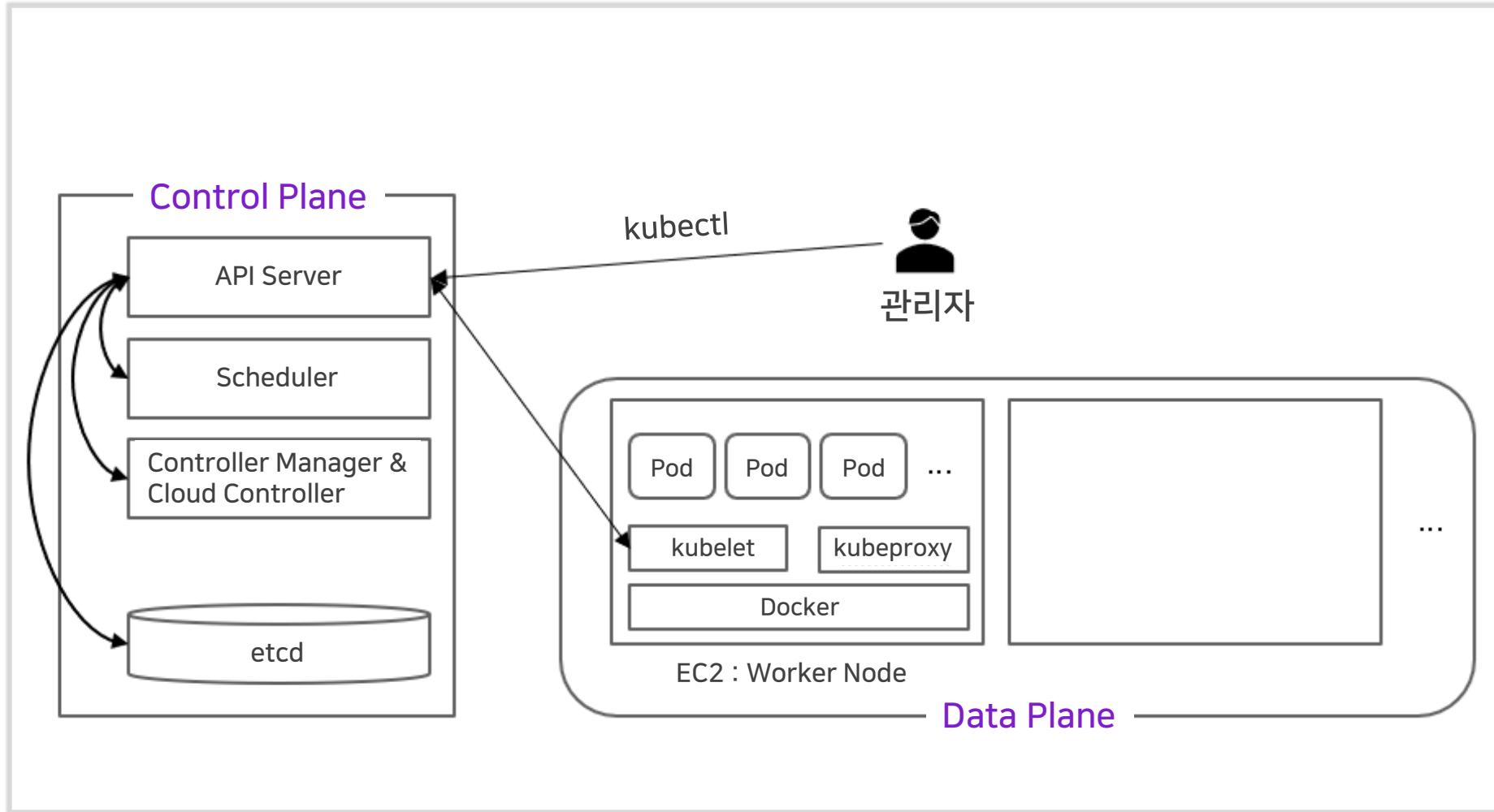
13. IRSA

I 리소스 정리

- kubectl delete -f pod-no-sa.yaml
- kubectl delete -f pod-sa.yaml
- eksctl delete iamserviceaccount --cluster demoNN --name sa1-s3-demoNN

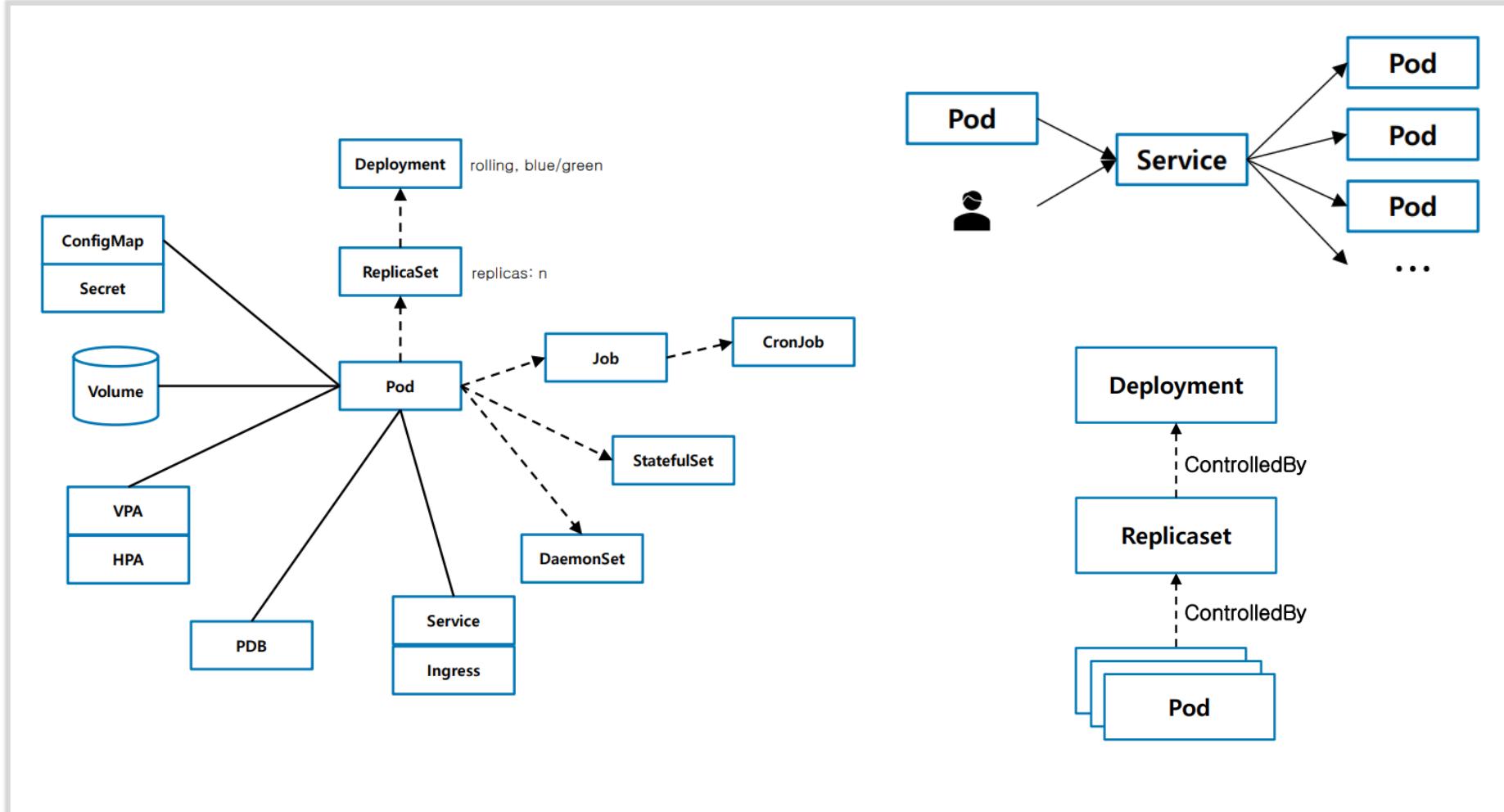
* 내용 정리

I Container Orchestration 도구 --> 대표적인 예가 k8s



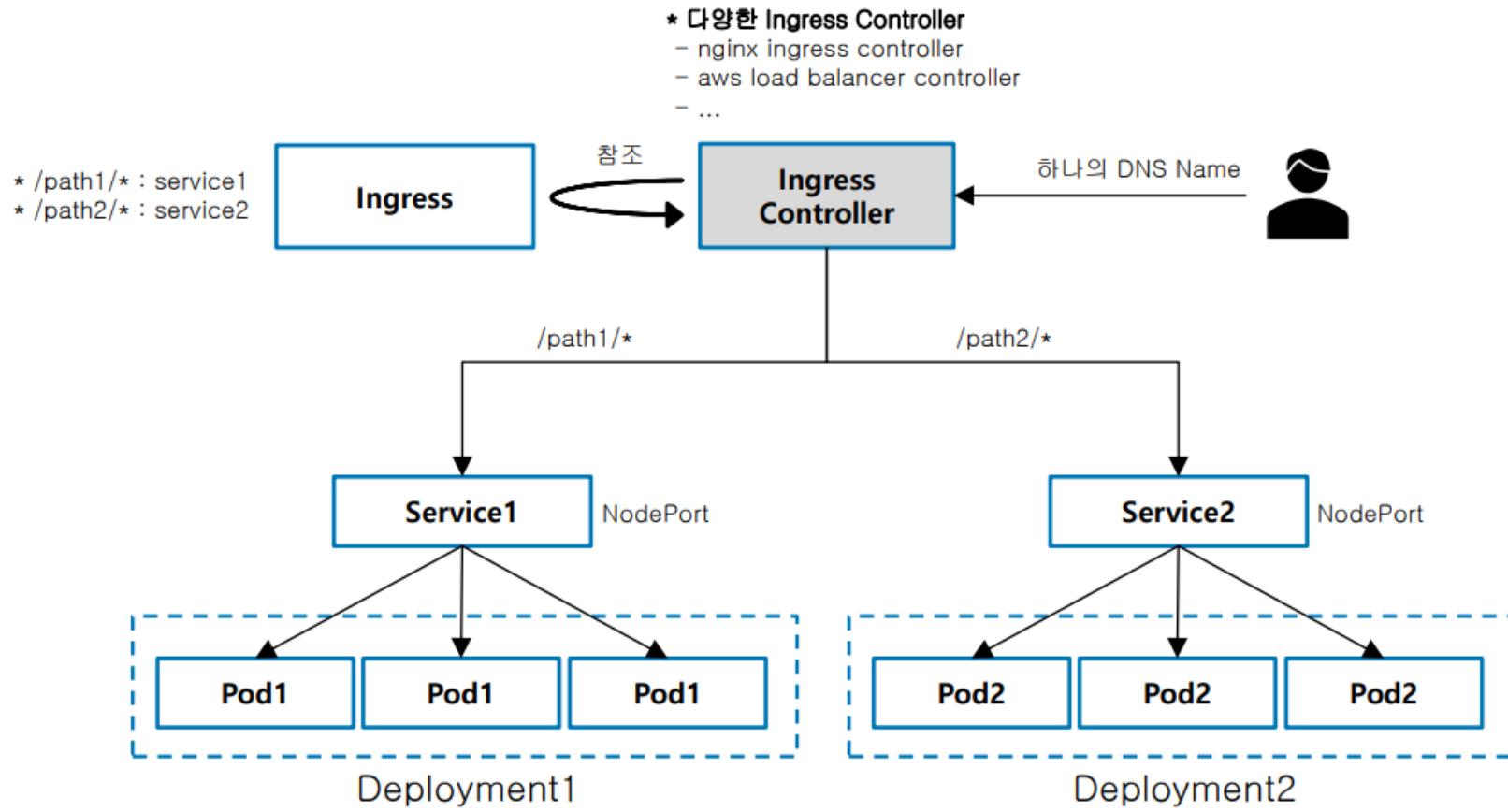
* 내용 정리

I k8s 리소스 객체



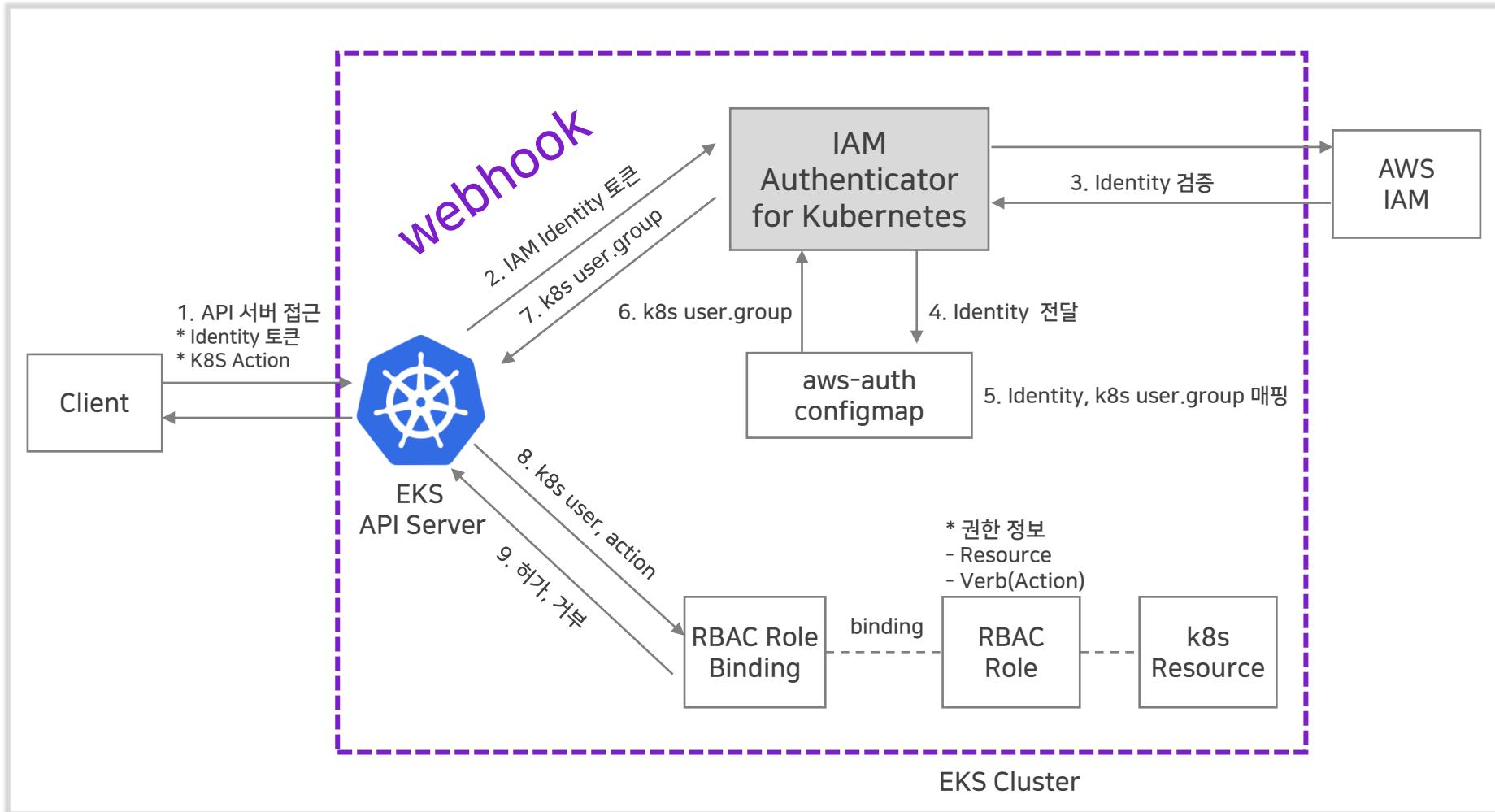
* 내용 정리

I Ingress



* 내용 정리

I 인증 및 인가 1



* 내용 정리

I 인증 및 인가 2 : IRSA

3. eksctl create iamserviceaccount 명령 --> 3-1, 3-2, 3-2 한번에 등록

