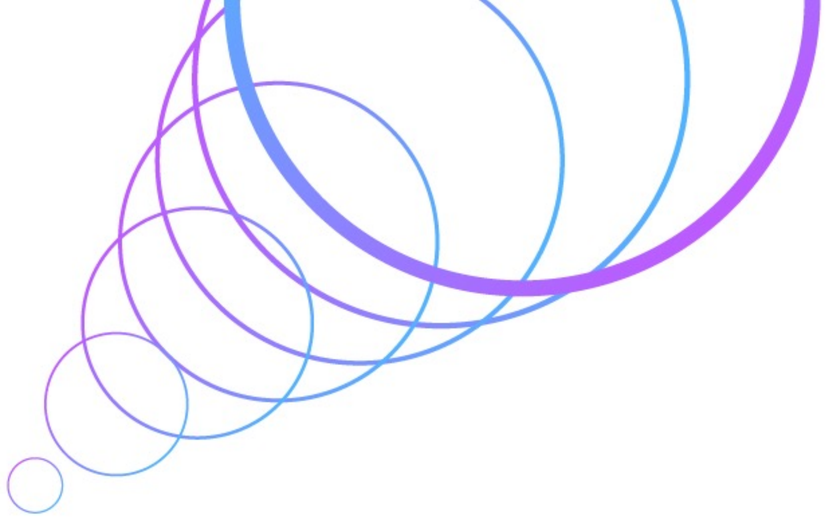
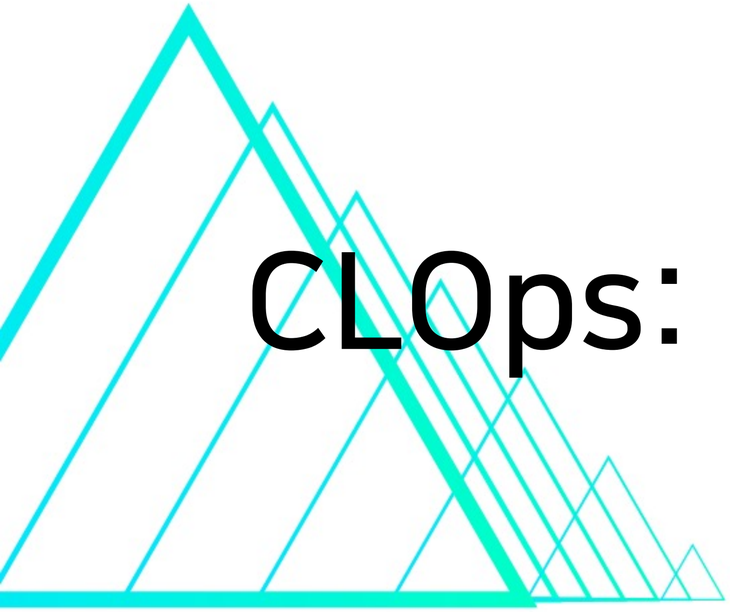




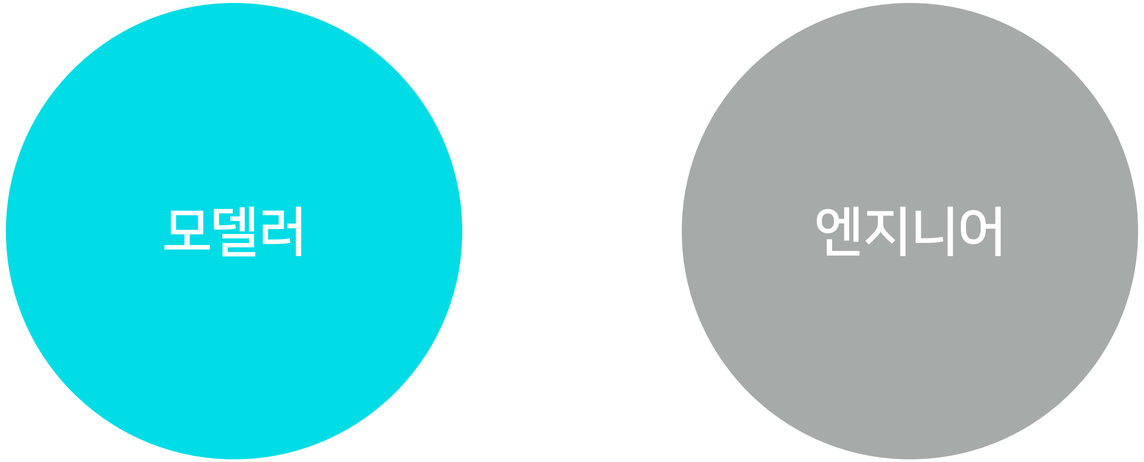
CLOps: 넌 모델링만해, 난 서빙할테니

이현동 ML System, NAVER CLOVA
칼 길리우스 ML System, NAVER CLOVA
한지승 ML System, NAVER CLOVA

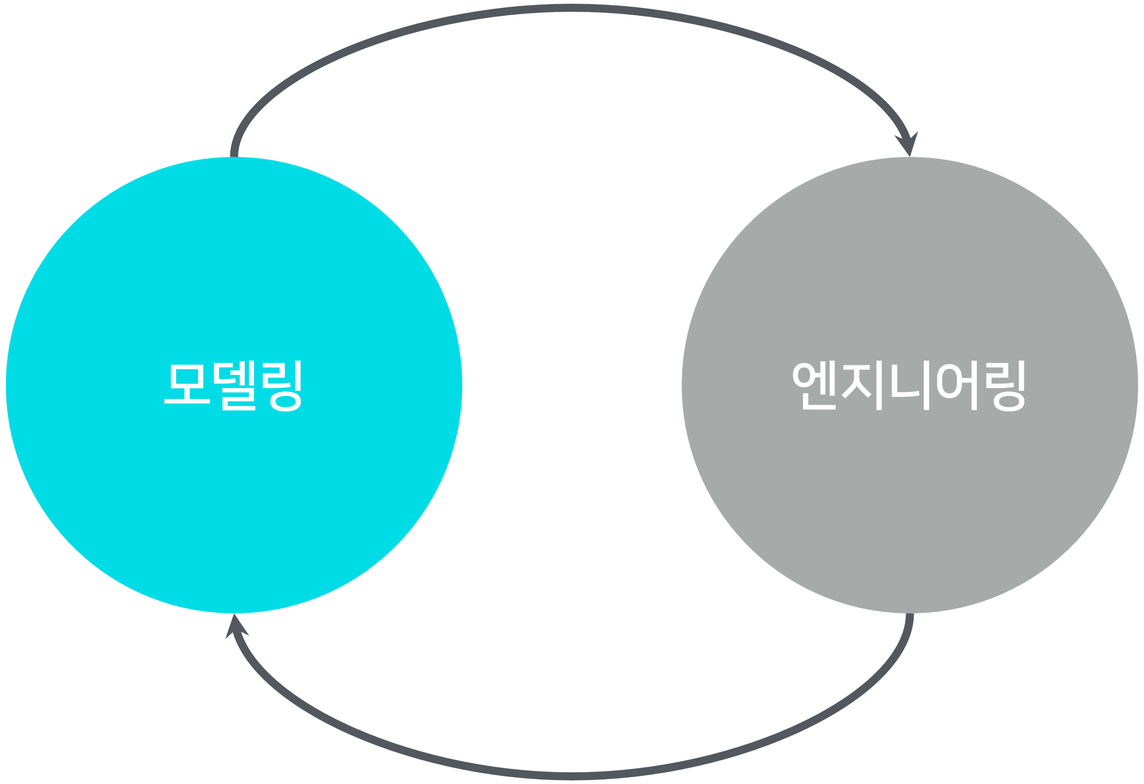
CONTENTS

1.Background

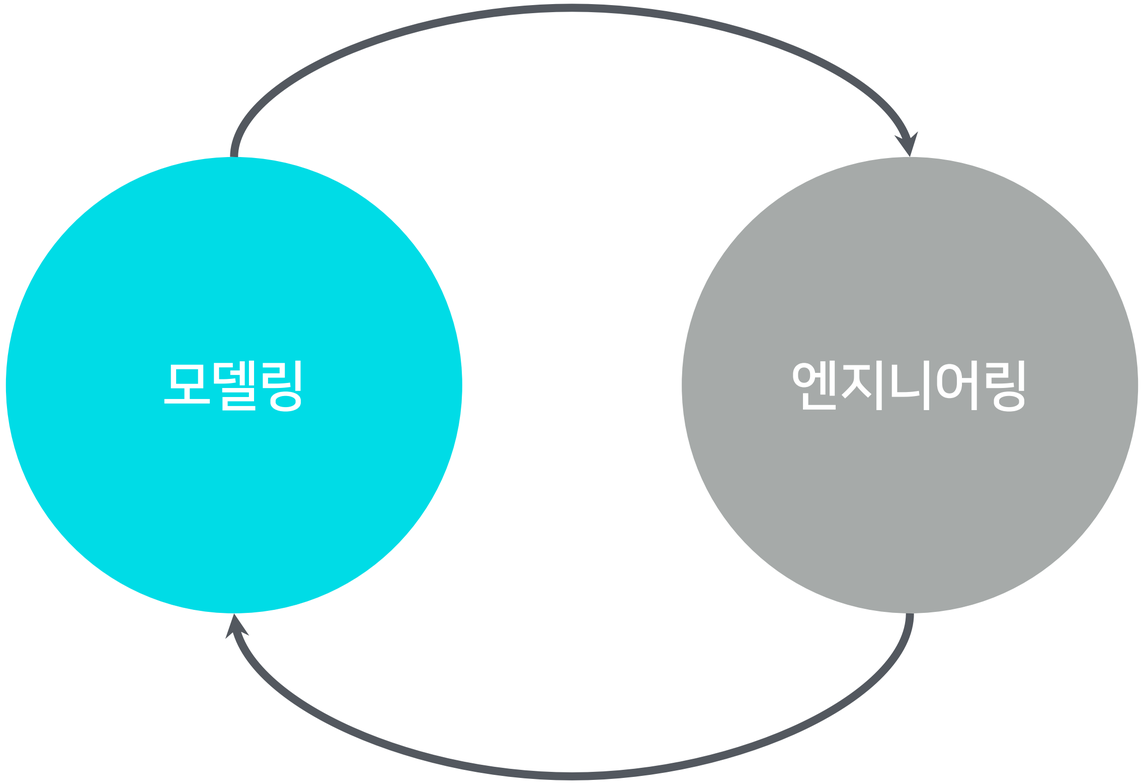
1.1 모델러와 엔지니어



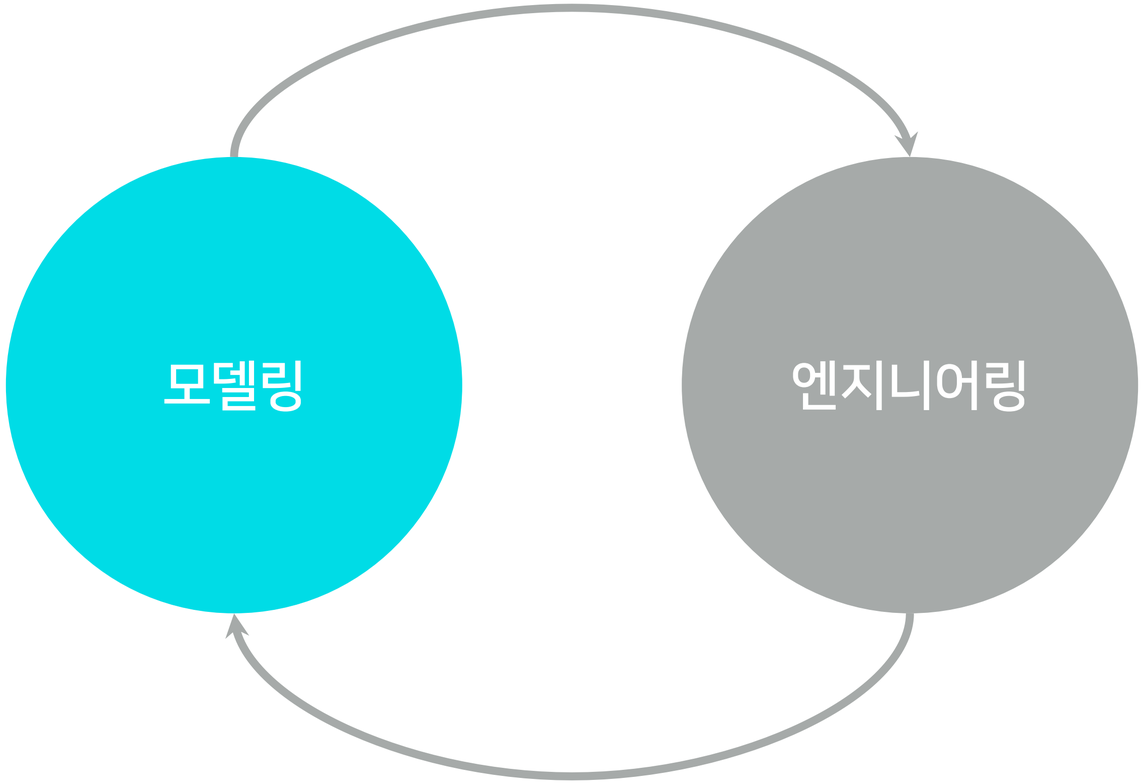
1.2 모델링과 엔지니어링



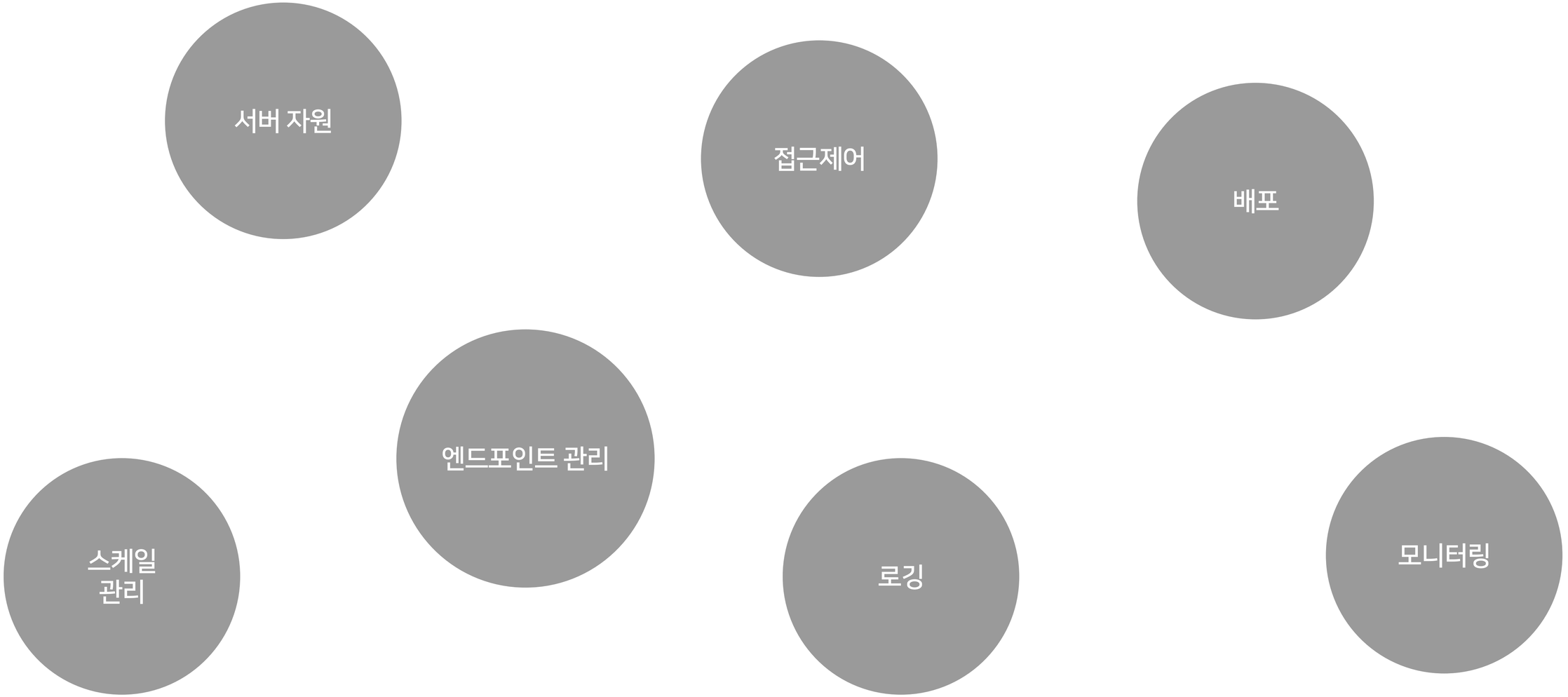
1.2 모델링과 엔지니어링



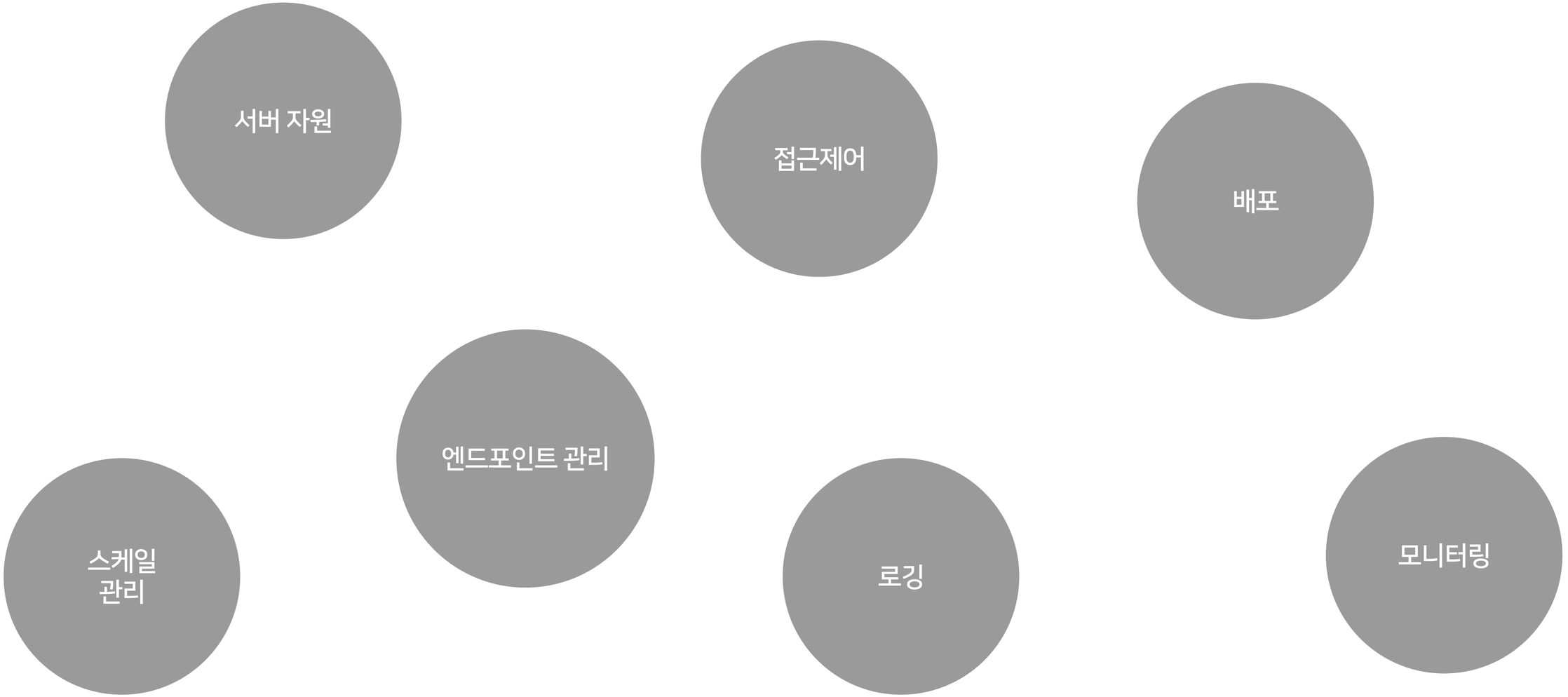
1.2 모델링과 엔지니어링



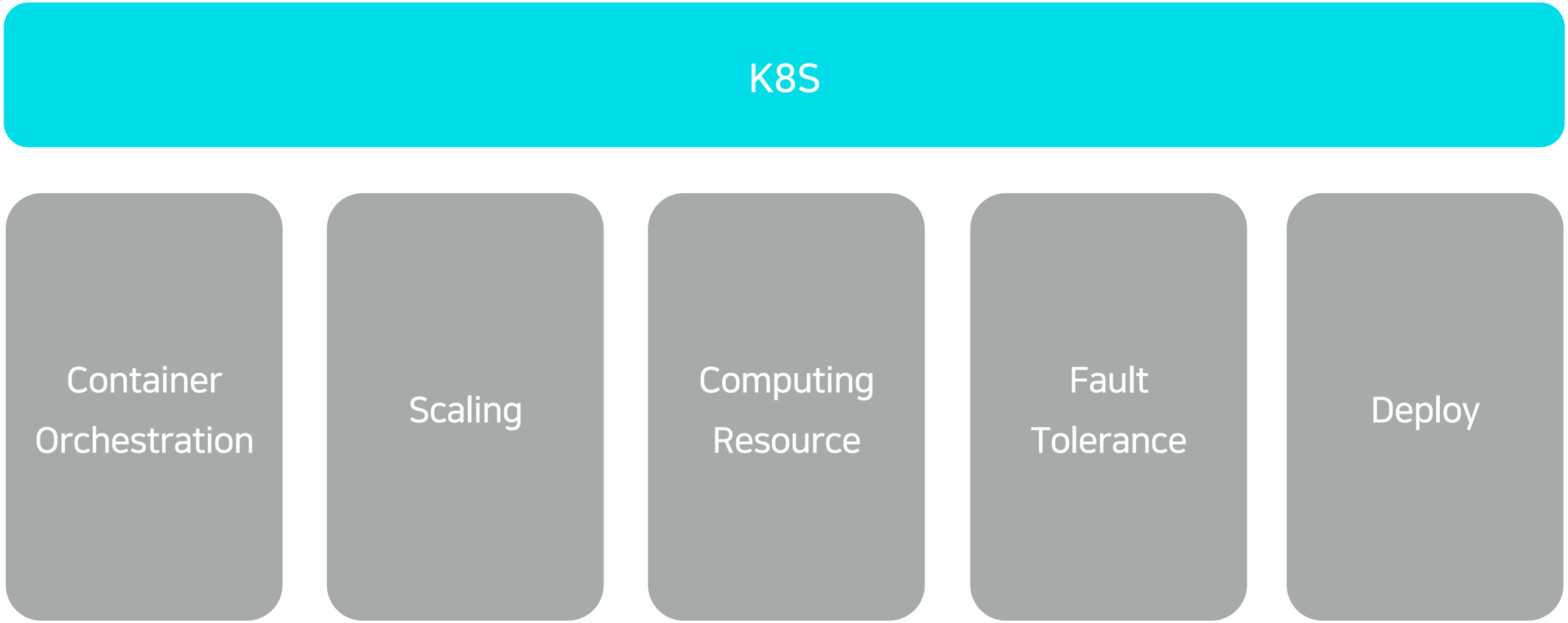
1.2 모델링과 엔지니어링



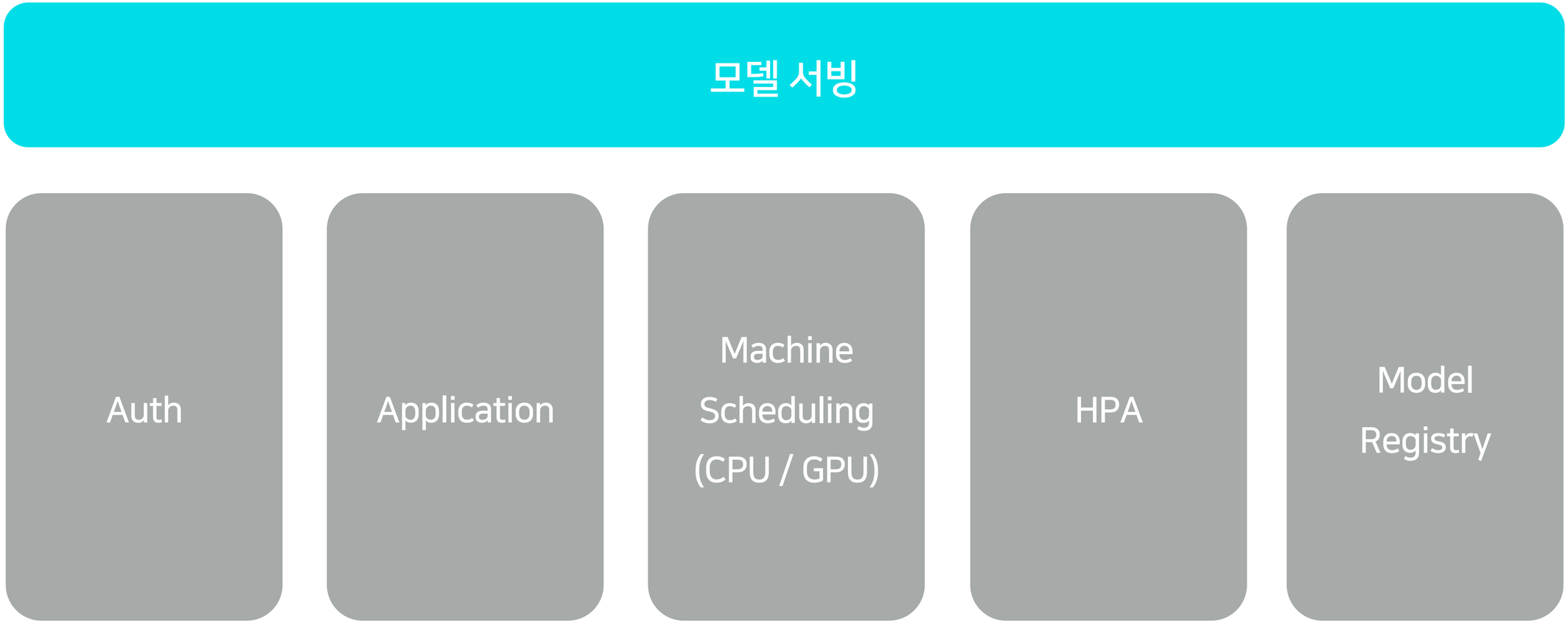
1.2 모델링과 엔지니어링



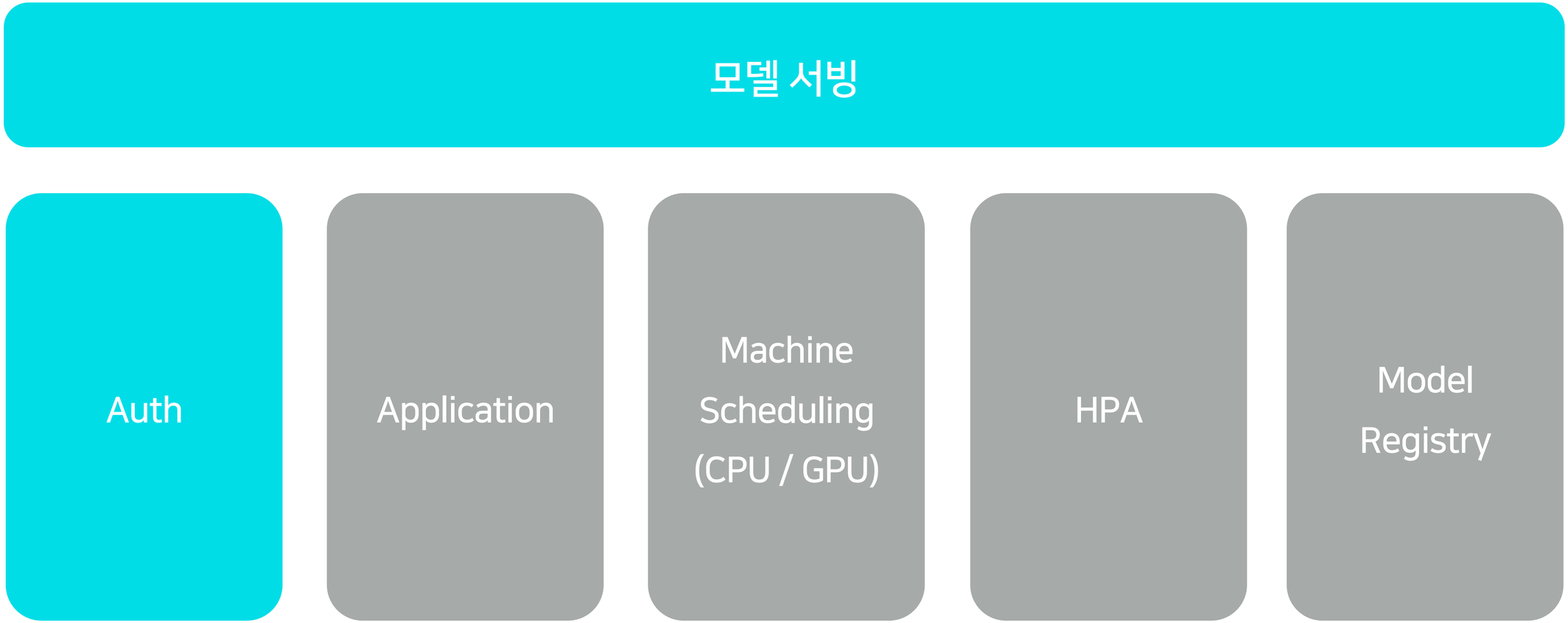
1.3 K8S를 선택한 이유



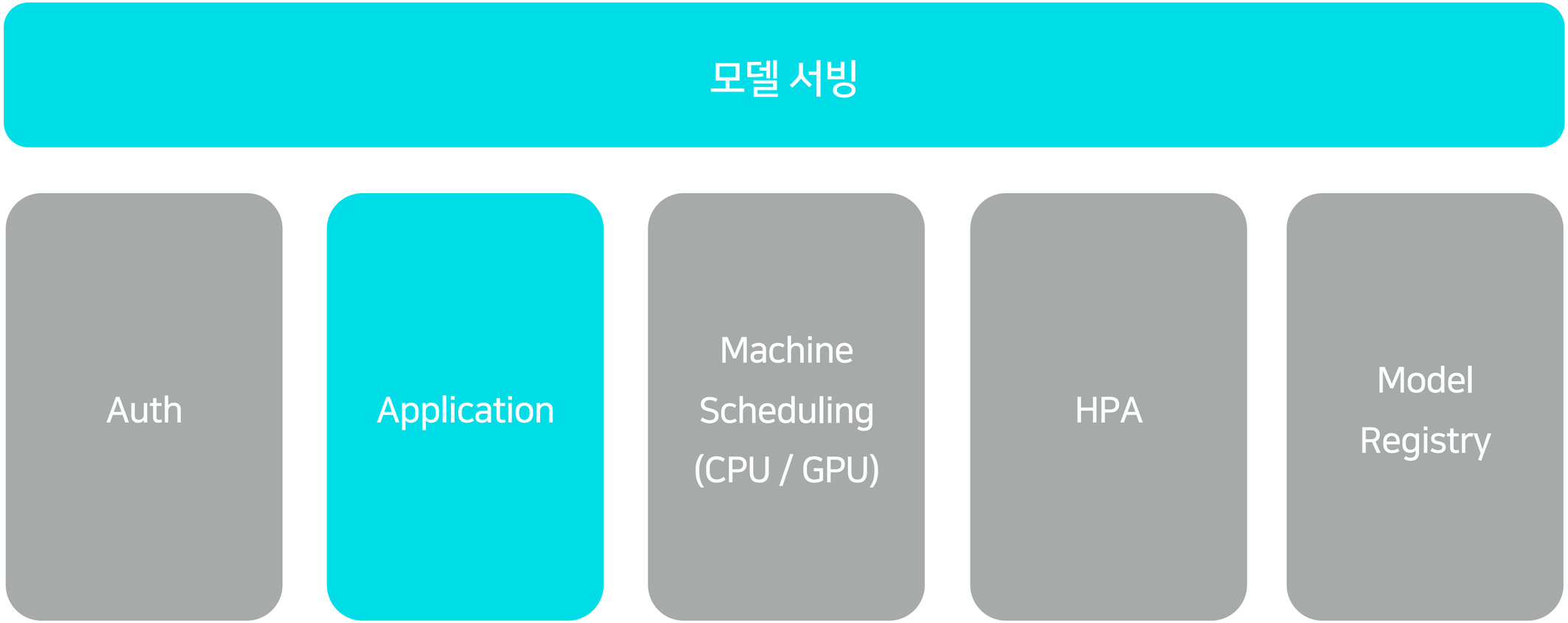
1.4 모델 서빙을 위한 필수 요소



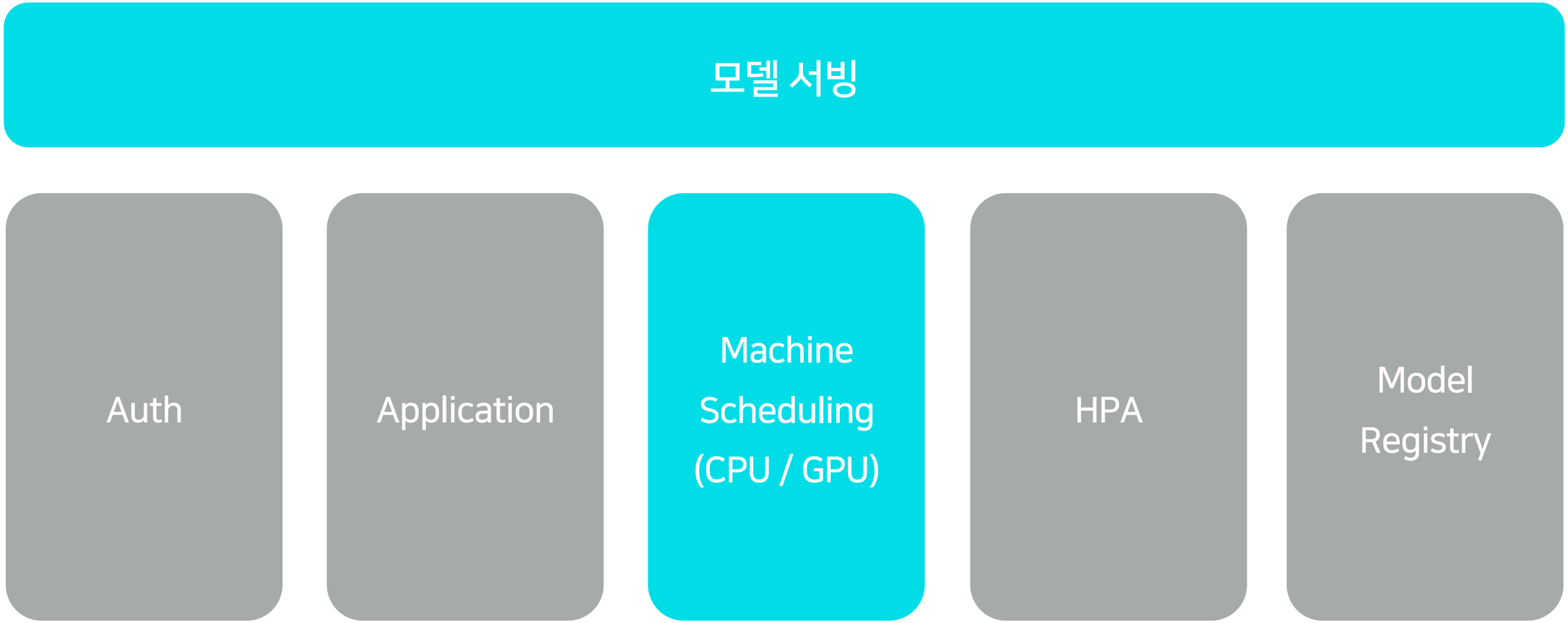
1.4 모델 서빙을 위한 필수 요소



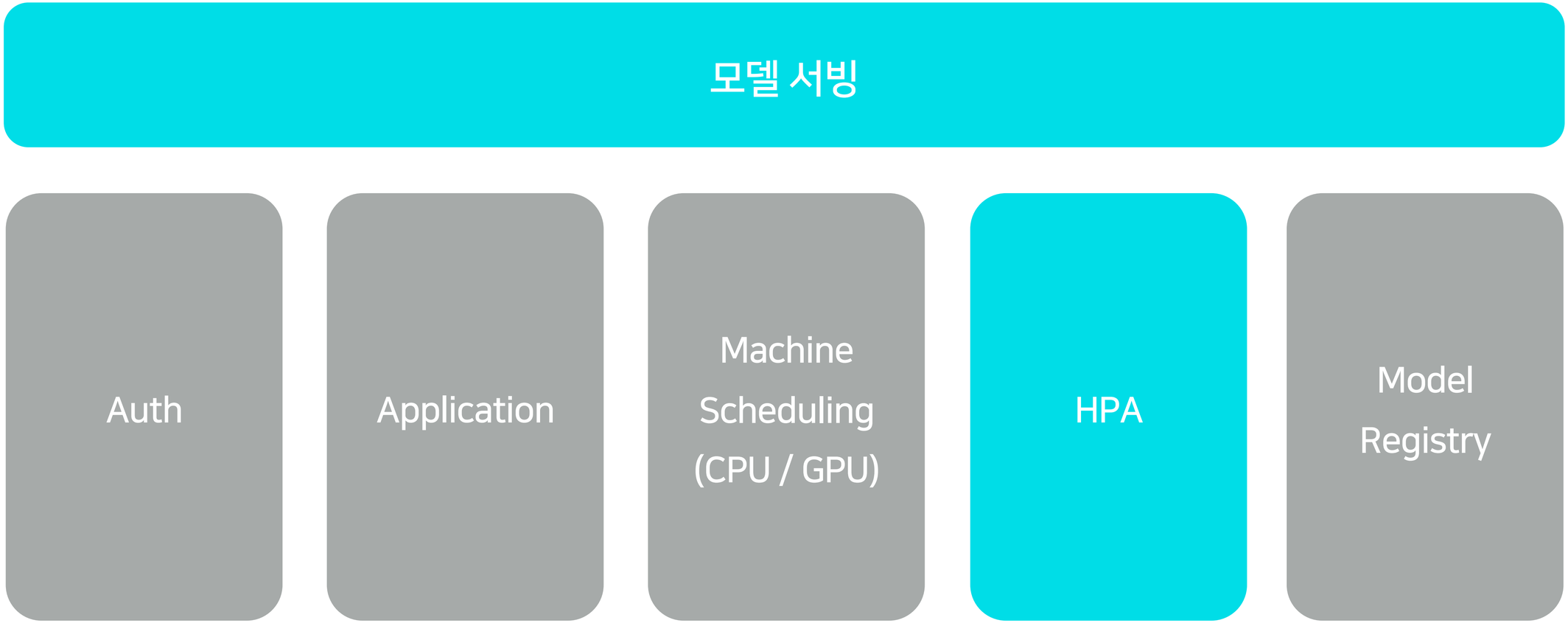
1.4 모델 서빙을 위한 필수 요소



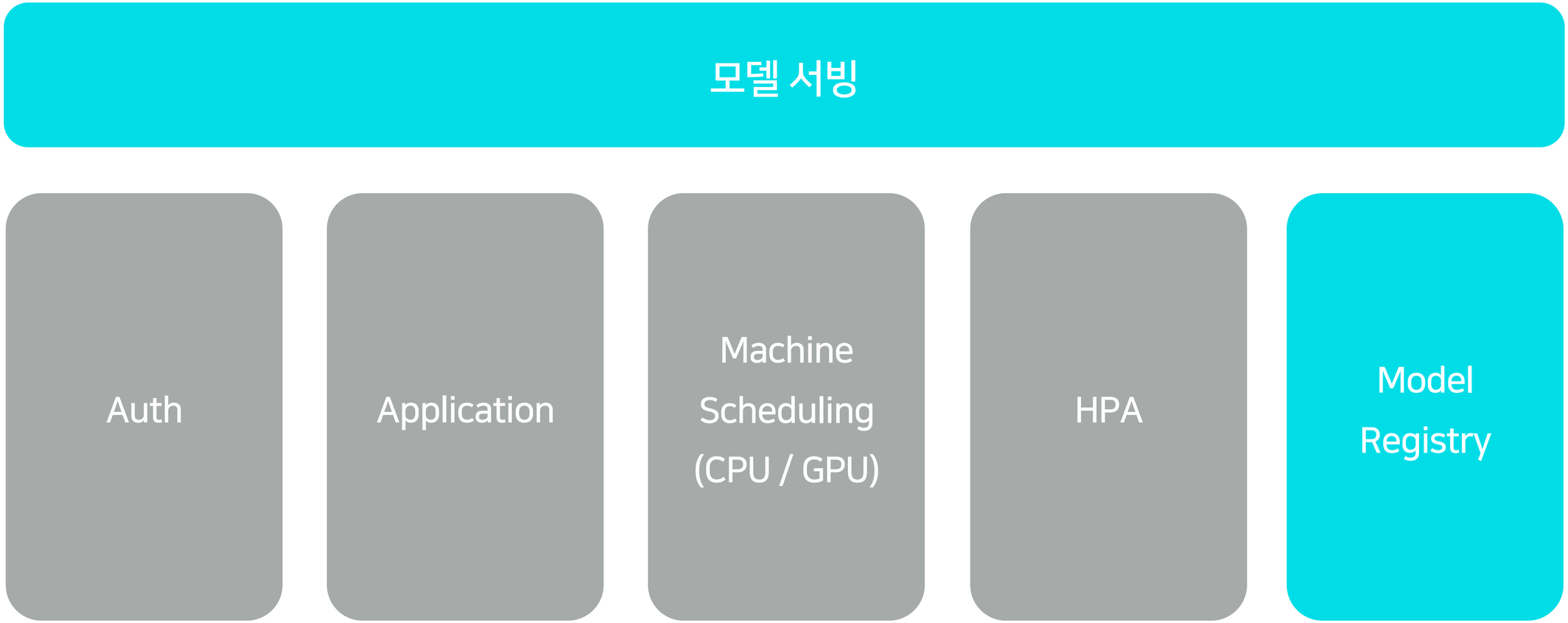
1.4 모델 서빙을 위한 필수 요소



1.4 모델 서빙을 위한 필수 요소



1.4 모델 서빙을 위한 필수 요소

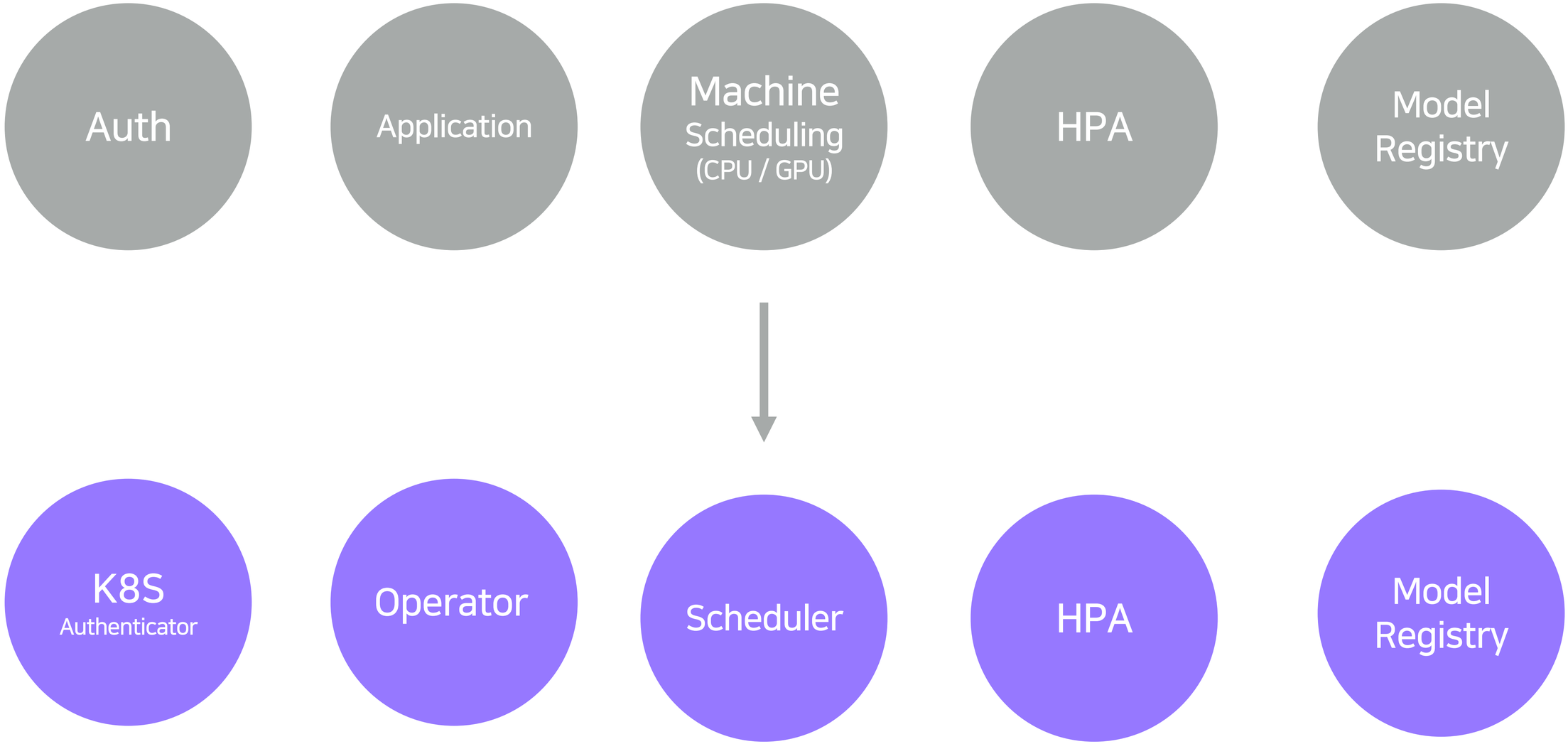


DEVIEW 2021

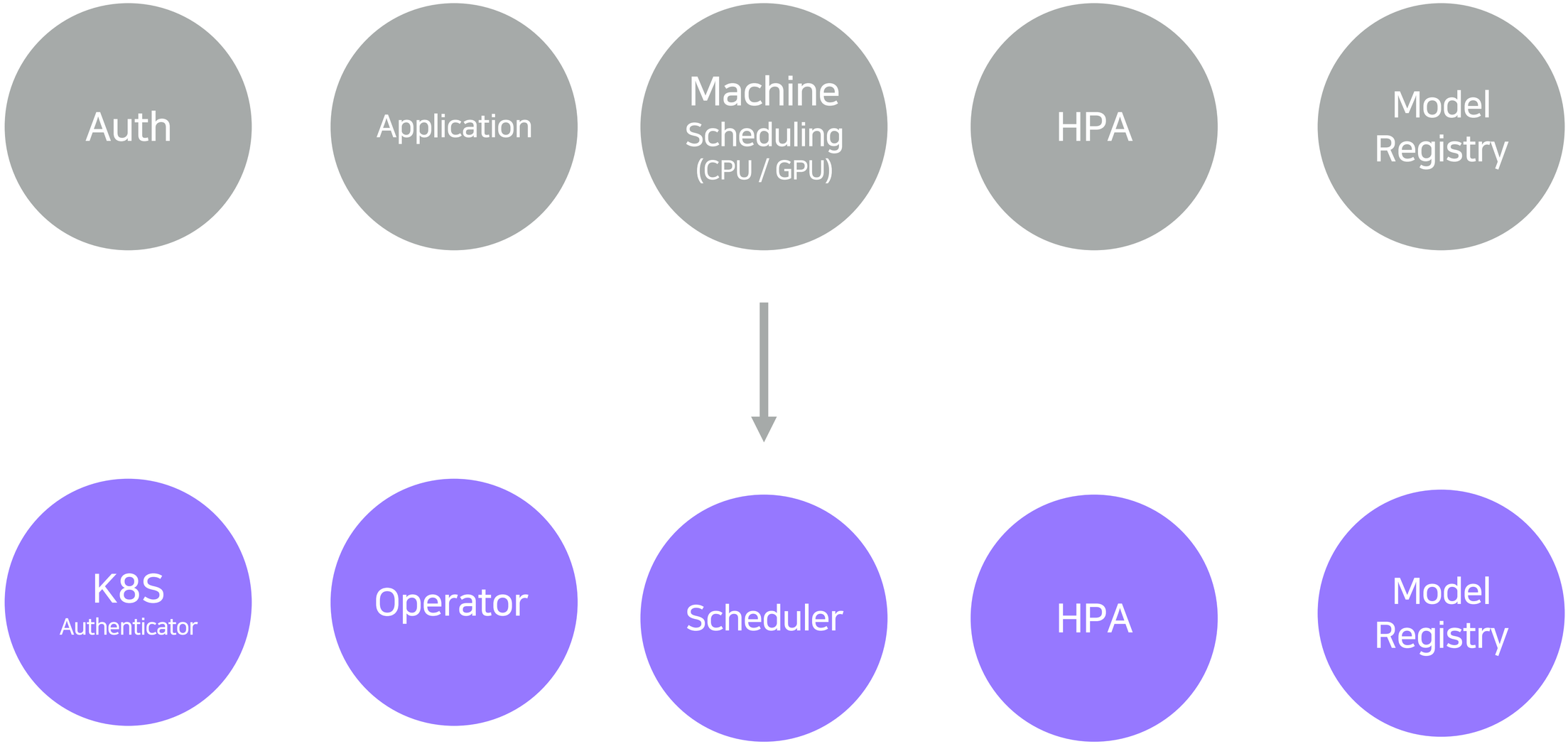
2.Architecture & Components

2.1 CLOps

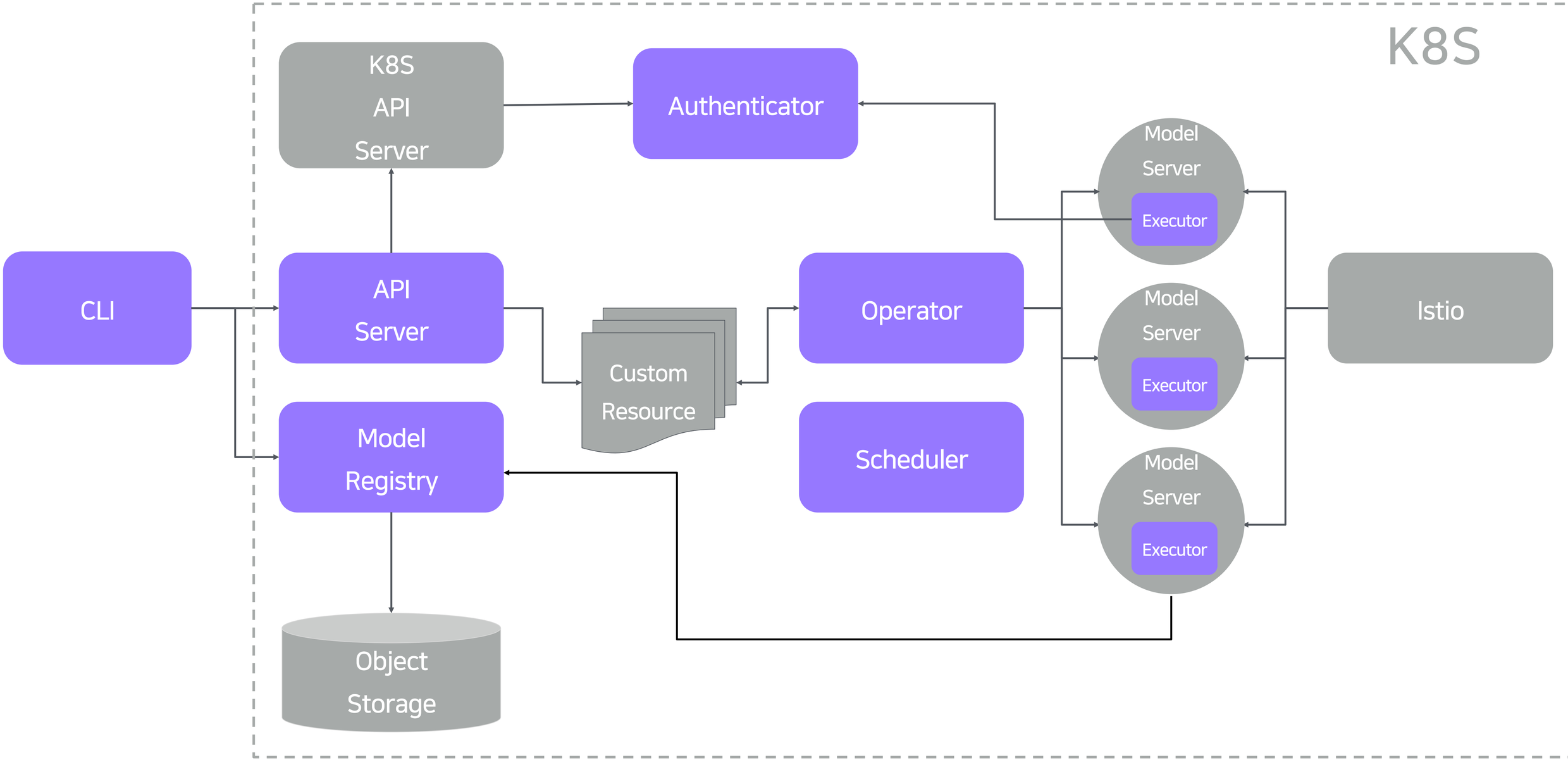
2.2 Overview


Auth
Application
Machine
Scheduling
(CPU / GPU)
HPA
Model
Registry
K8S
Authenticator
Operator
Scheduler
HPA
Model
Registry

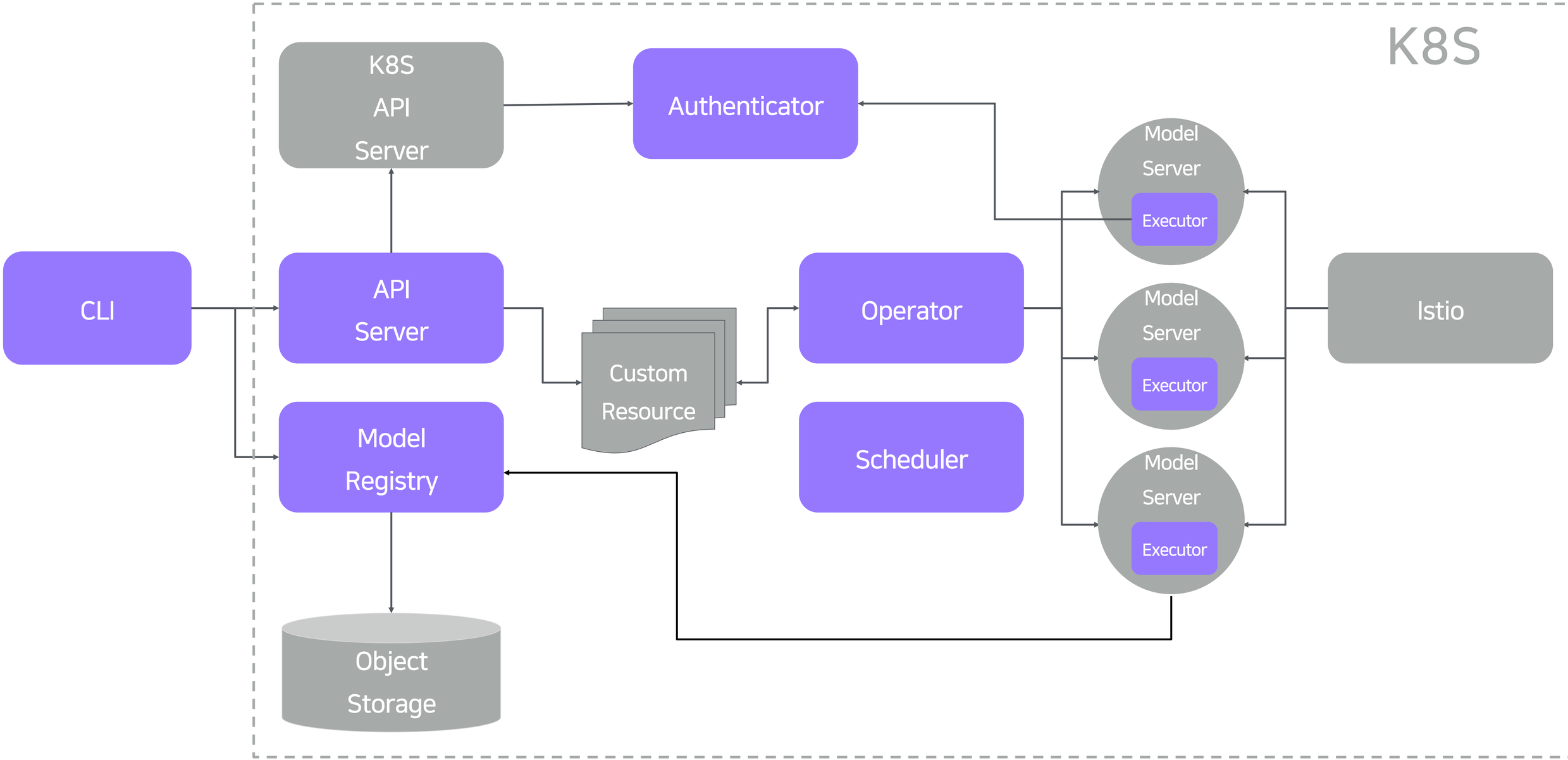
2.2 Overview


Auth
Application
Machine
Scheduling
(CPU / GPU)
HPA
Model
Registry
K8S
Authenticator
Operator
Scheduler
HPA
Model
Registry

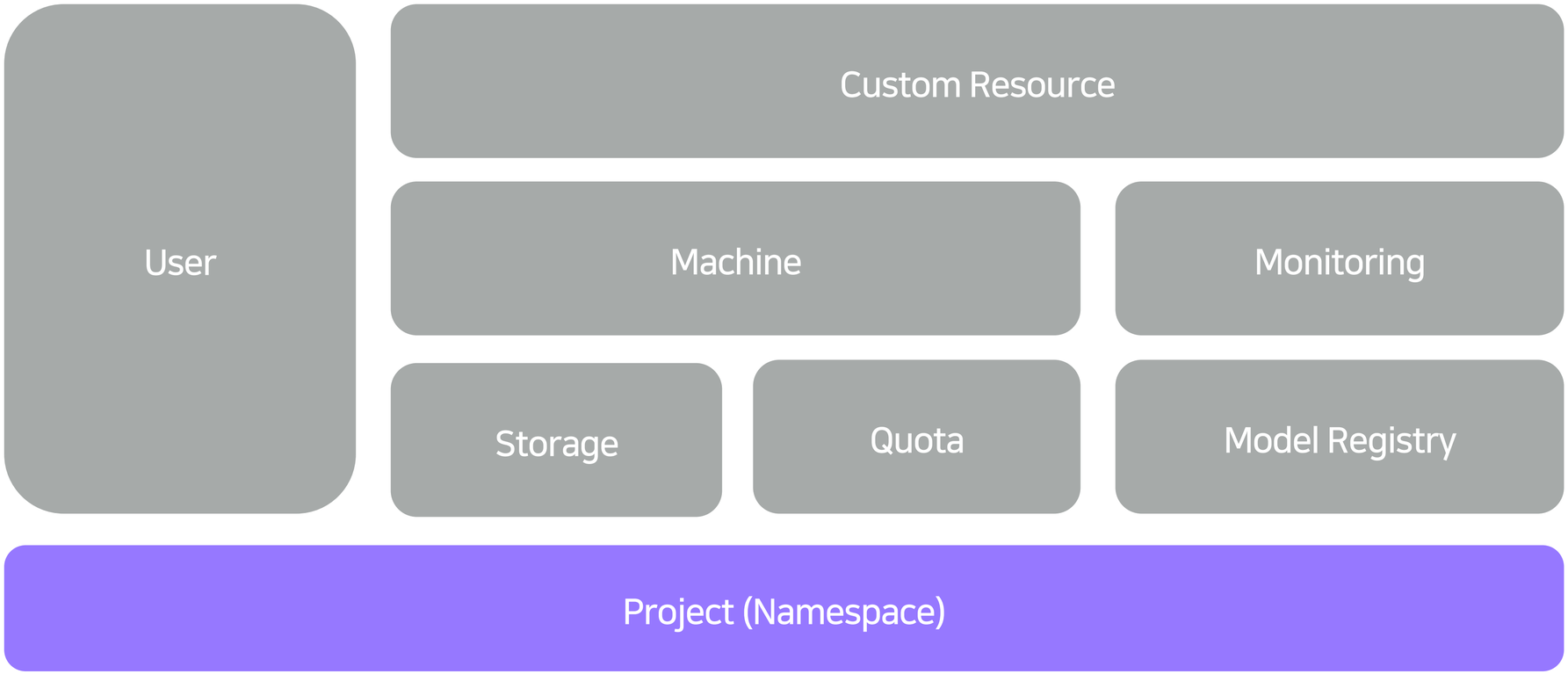
2.2 Overview


K8S
K8S API Server
Authenticator
Model Server
Executor
CLI
API Server
Custom Resource
Operator
Istio
Model Server
Executor
Model Registry
Scheduler
Model Server
Executor
Object Storage

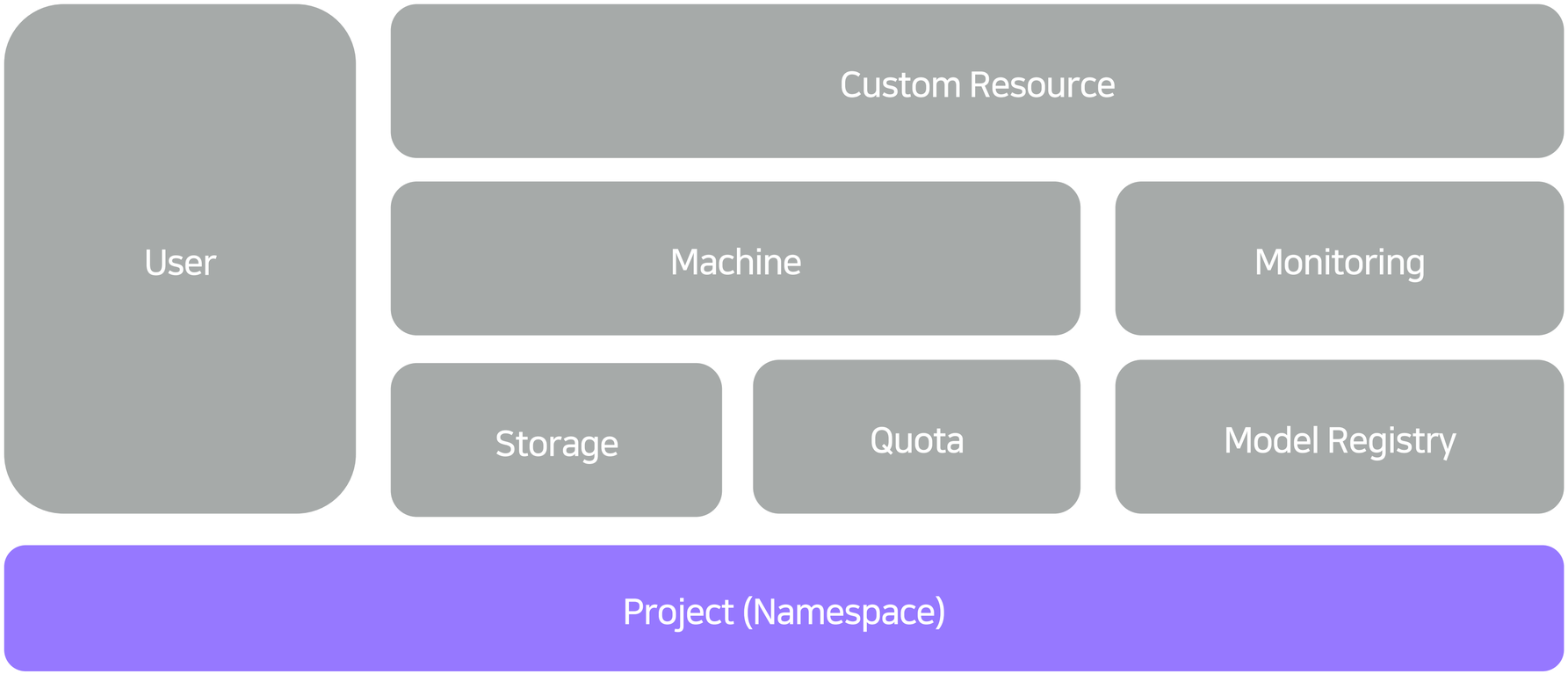
2.2 Overview


K8S
K8S API Server
Authenticator
Model Server
Executor
CLI
API Server
Custom Resource
Operator
Istio
Model Server
Executor
Model Registry
Scheduler
Model Server
Executor
Object Storage

2.2 Overview


User
Custom Resource
Machine
Monitoring
Storage
Quota
Model Registry
Project (Namespace)

2.2 Overview


User
Custom Resource
Machine
Monitoring
Storage
Quota
Model Registry
Project (Namespace)

2.3 Authenticator – Why

DEVIEW 2021

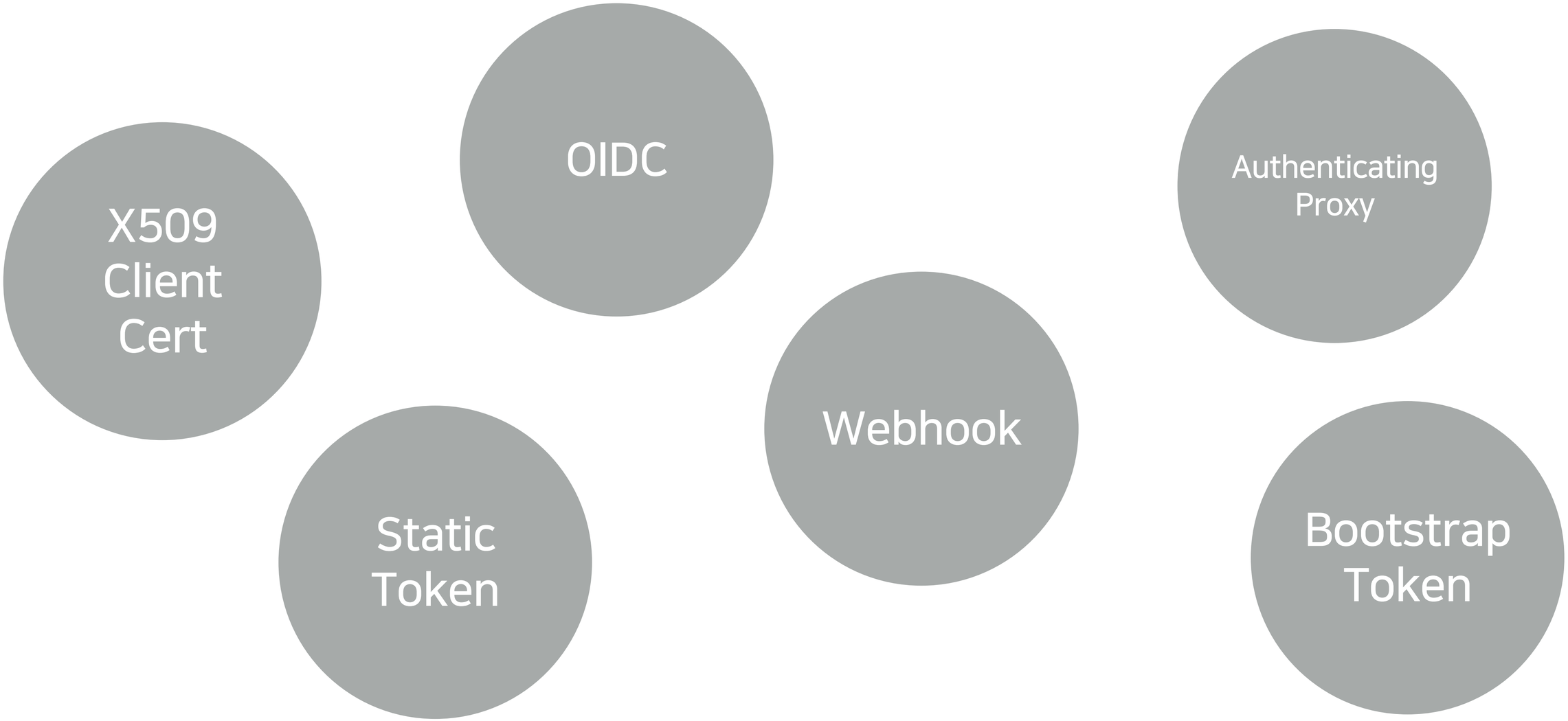
접근 제어

- 인가된 사용자에 의해서만 CLOps 사용 가능

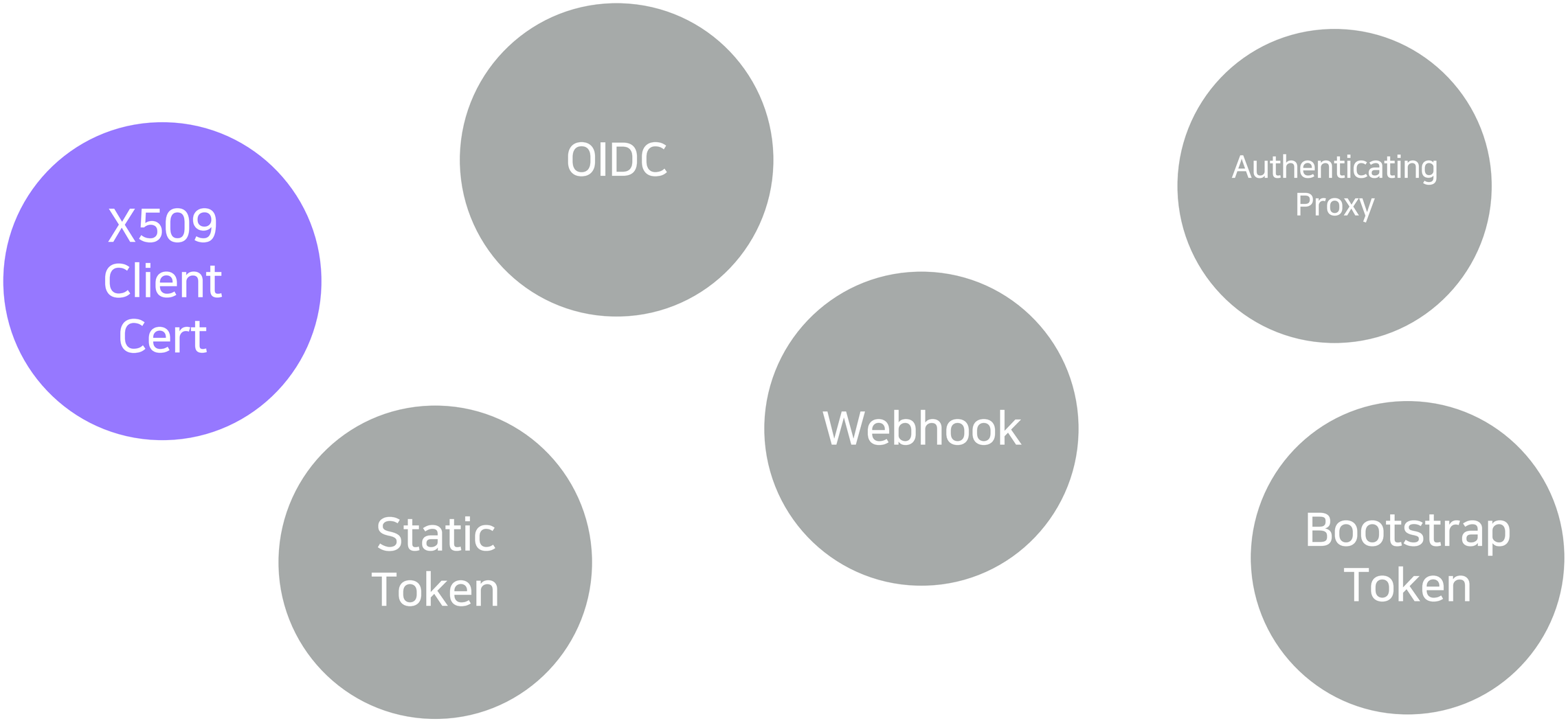
프로젝트별 권한 관리

- 권한에 따라 각각 다른 기능 사용 가능

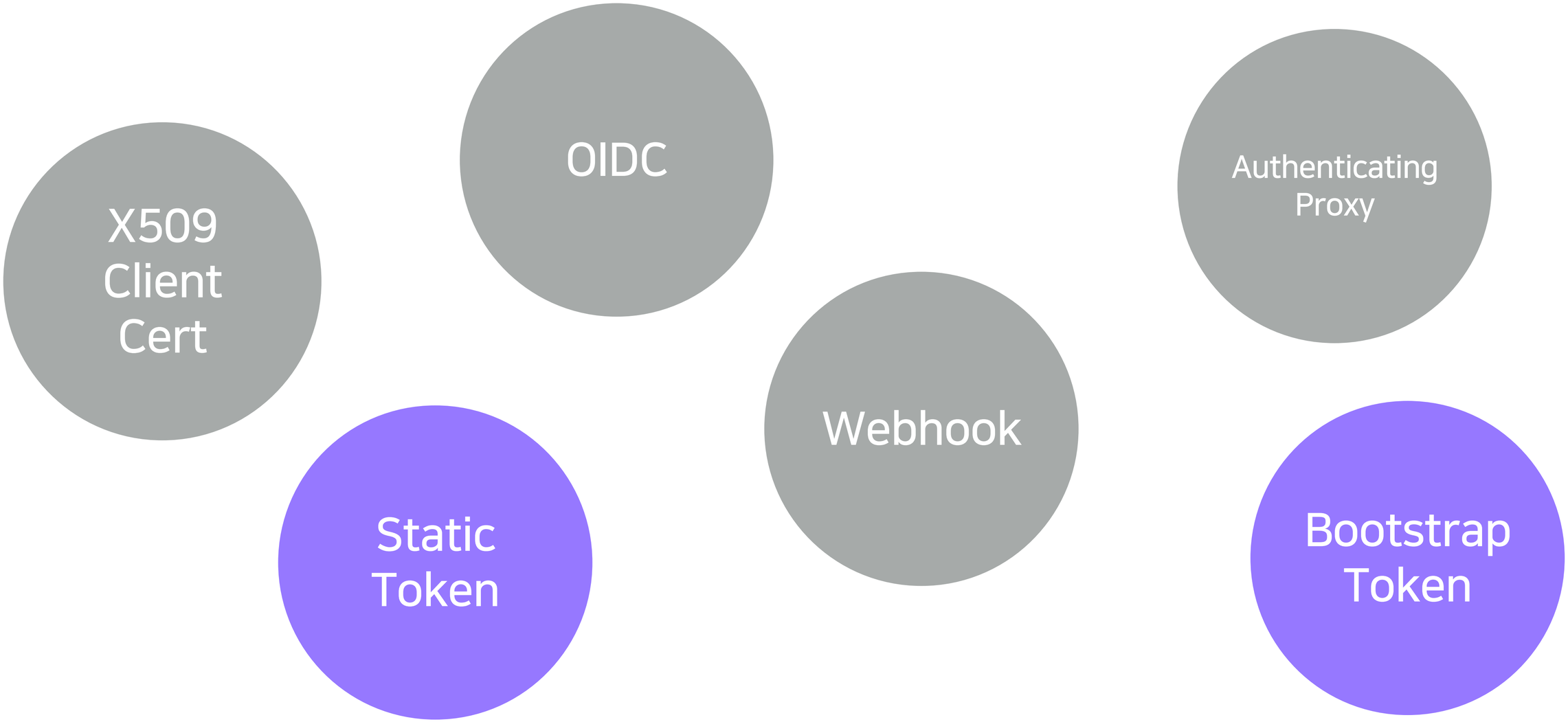
2.3 K8S Authenticator - Detail



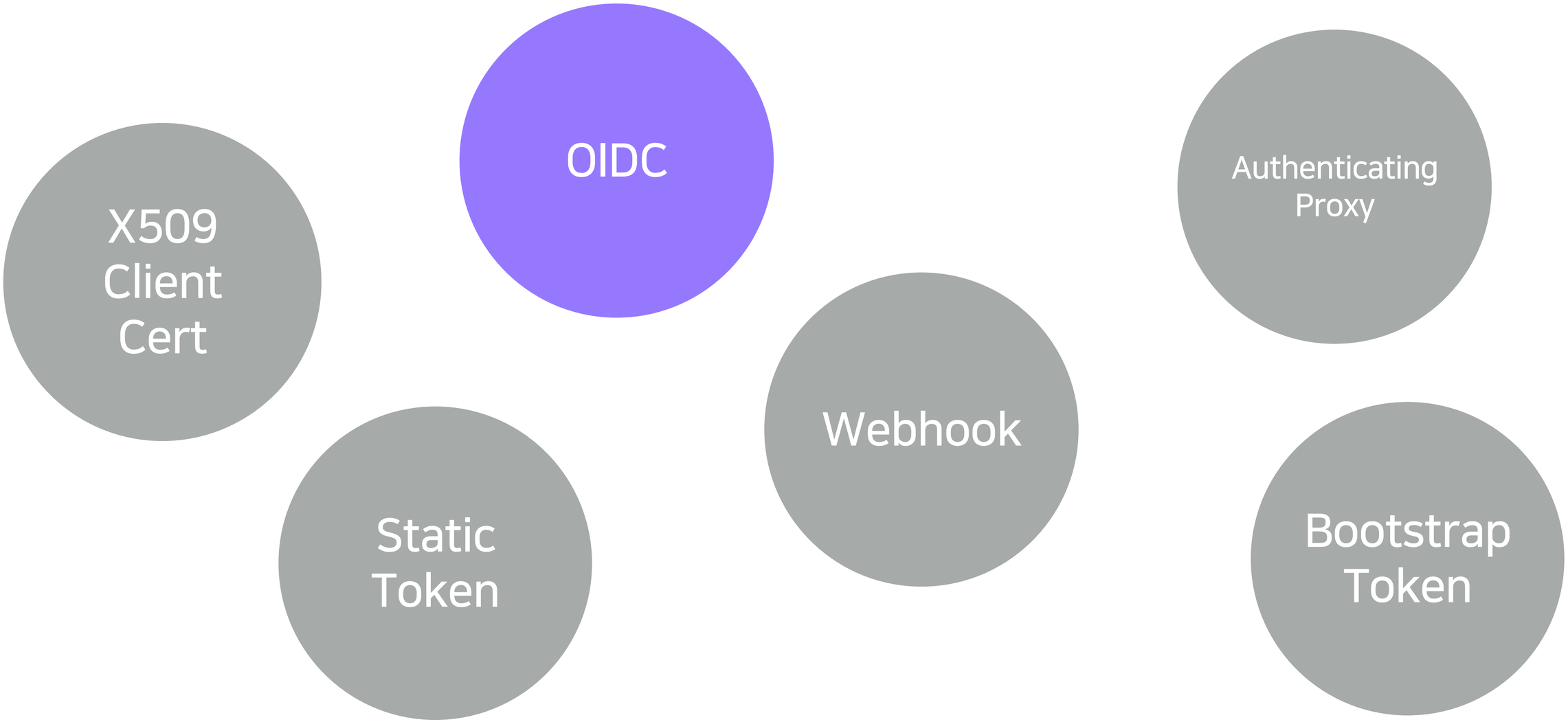
2.3 K8S Authenticator - Detail



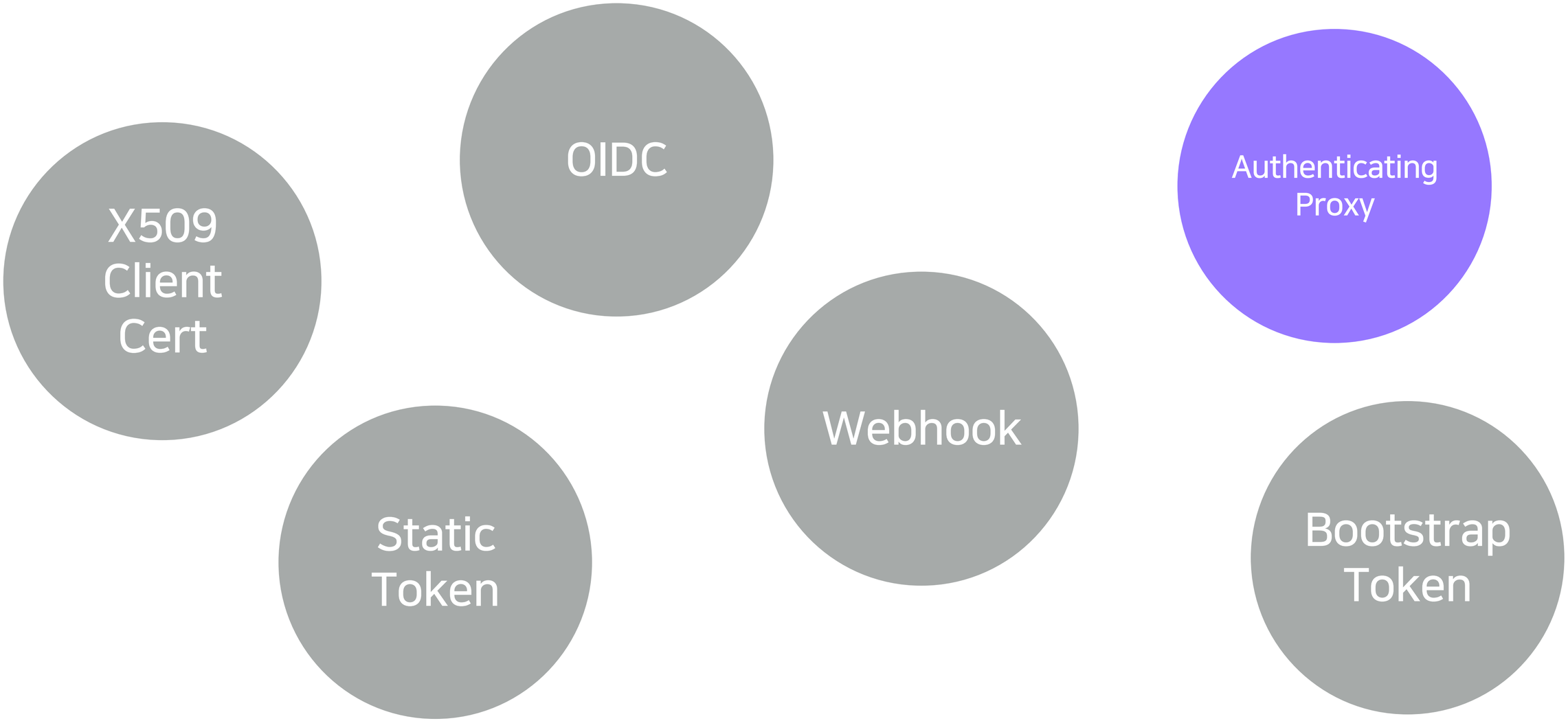
2.3 K8S Authenticator - Detail



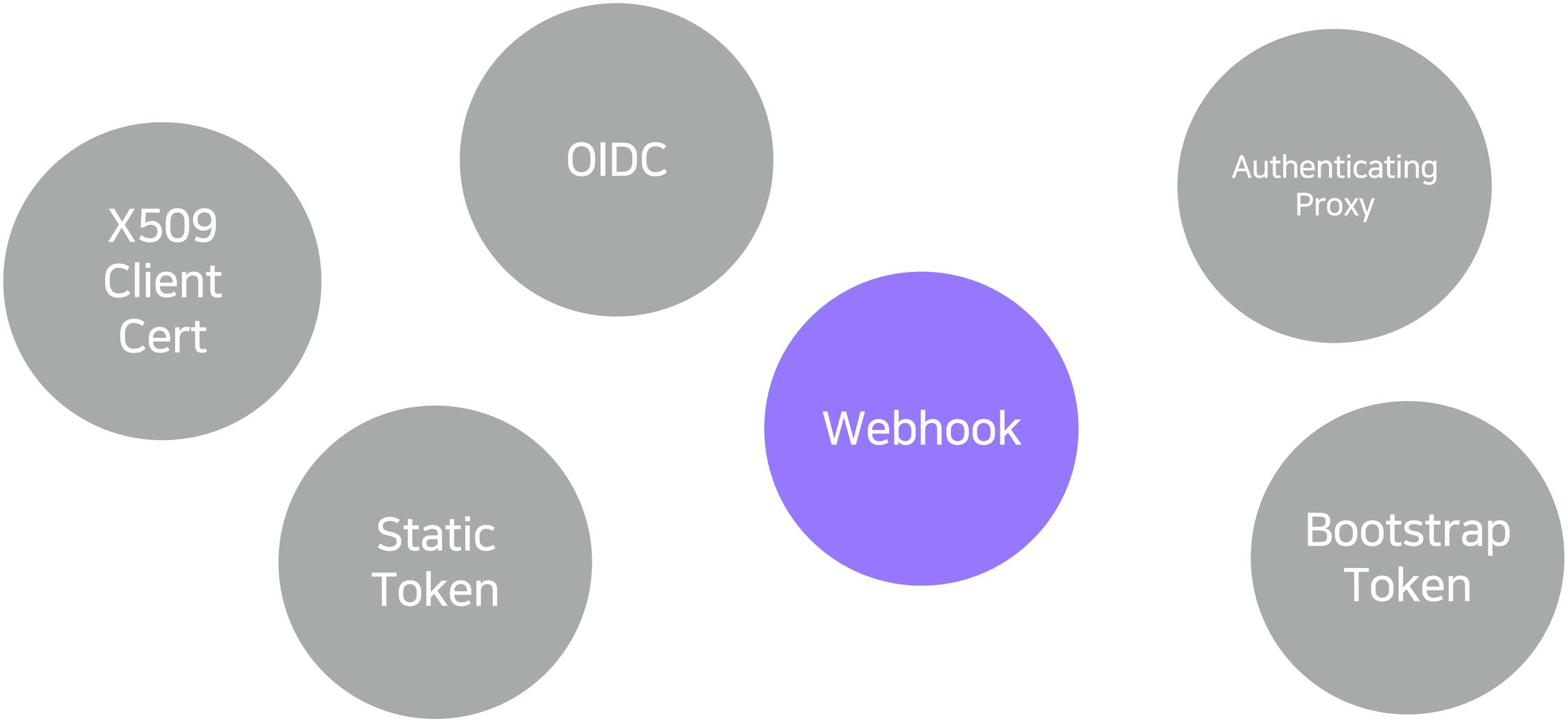
2.3 K8S Authenticator - Detail



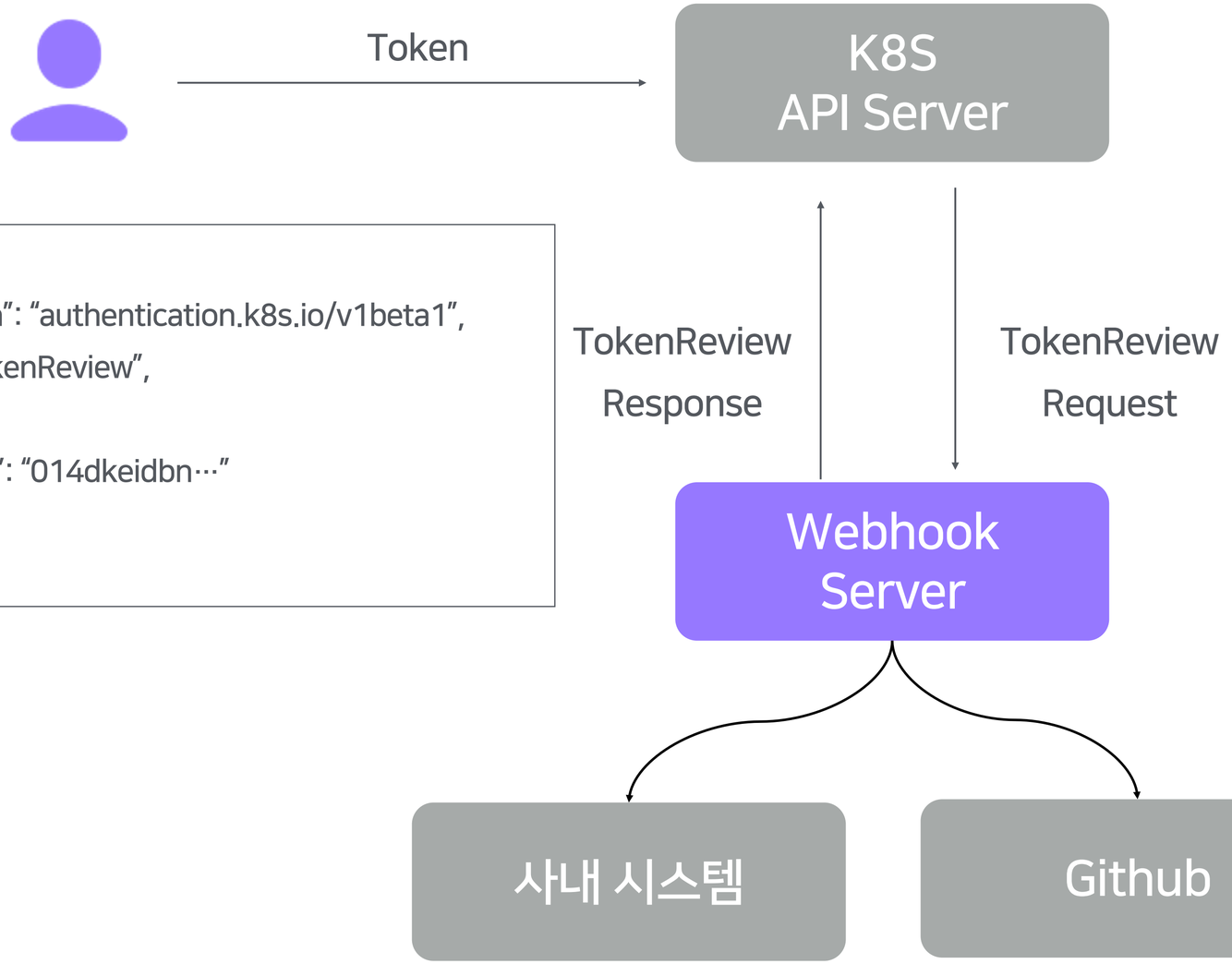
2.3 K8S Authenticator - Detail



2.3 K8S Authenticator - Detail



2.3 K8S Authenticator - Detail




```
{
    "apiVersion": "authentication.k8s.io/v1beta1",
    "kind": "TokenReview",
    "spec": {
        "token": "014dkeidbn…"
    }
}
```

```
{
    "apiVersion": "authentication.k8s.io/v1beta1",
    "kind": "TokenReview",
    "status": {
        "authenticated": true,
        "user": {
            "username": janedoe@example.com,
            "uid": "40",
            "groups", ["developers", "admin"]
        }
    }
}
```

2.3 K8S Authenticator - Detail

Configuration

- --authentication-token-webhook-config-file

```
clusters:
- name: webhook-token-auth-cluster
  cluster:
    server: https://{host}
    insecure-skip-tls-verify: False

users:
- name: webhook-token-auth-user

current-context: webhook-token-auth
contexts:
- context:
    cluster: webhook-token-auth-cluster
    user: webhook-token-auth-user
  name: webhook-token-auth
```

2.4 Operator - Why

모델 서빙을 위한 필요 기능 자동화

- 모델 애플리케이션 구현 외 서빙 및 운영 기능을 쉽게 배포 목표
- (모델러) 모델 배포를 위한 애플리케이션 및 설정만 선언

SDK

- Kubebuilder (https://github.com/kubernetes-sigs/kubebuilder)
- Operator-sdk (https://sdk.operatorframework.io)

2.4 Operator - Detail

DEVIEW 2021

Resource

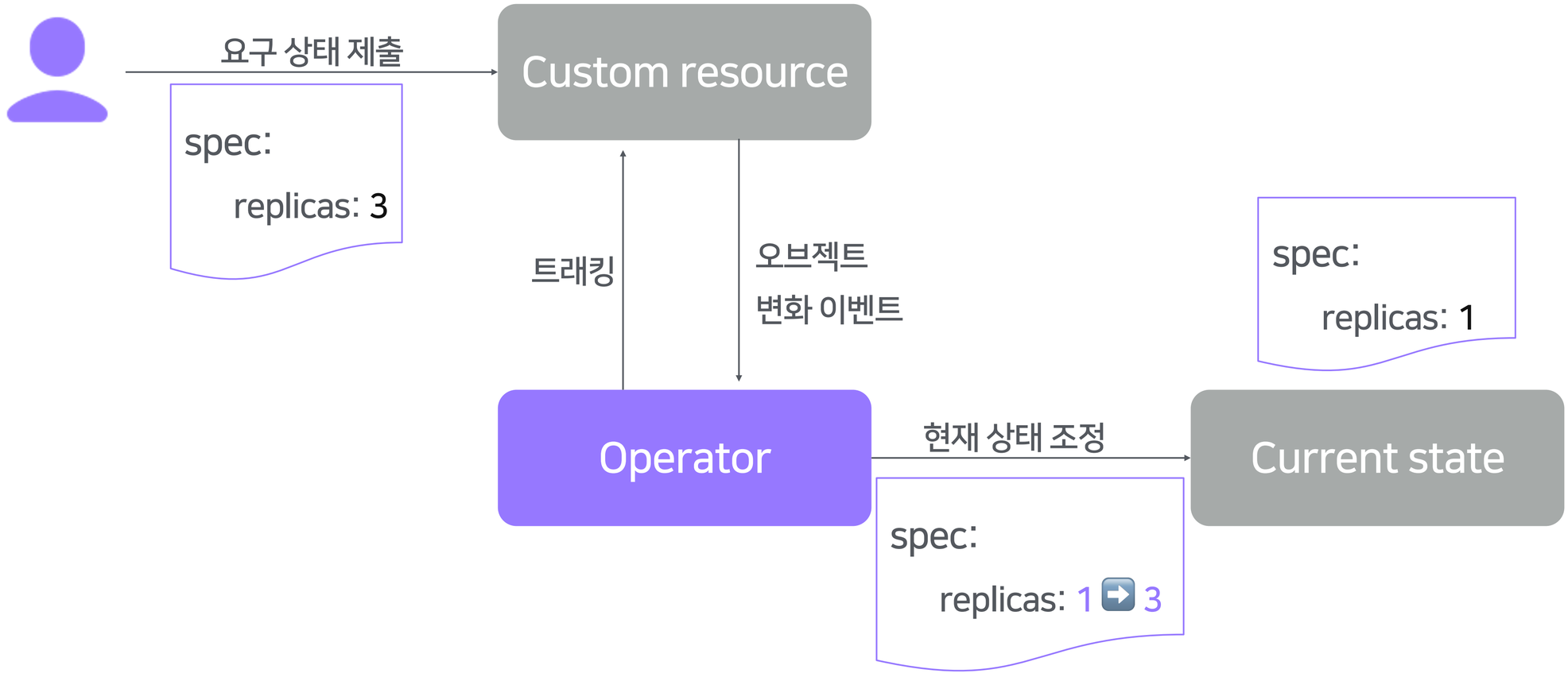
- 리소스는 “의도를 담은 레코드”
- 사용자가 K8S API 사용을 하기 위한 인터페이스 역할
- 코어 리소스로 Deployment, Service, DaemonSet 등을 제공

Custom Resource (CR)

- 기본 리소스 외 어플리케이션 제공자가 정의한 쿠버네티스 리소스
- 모든 오브젝트 리소스는 동일한 형태로 안정적으로 확장 가능한 구조

2.4 Operator - Detail

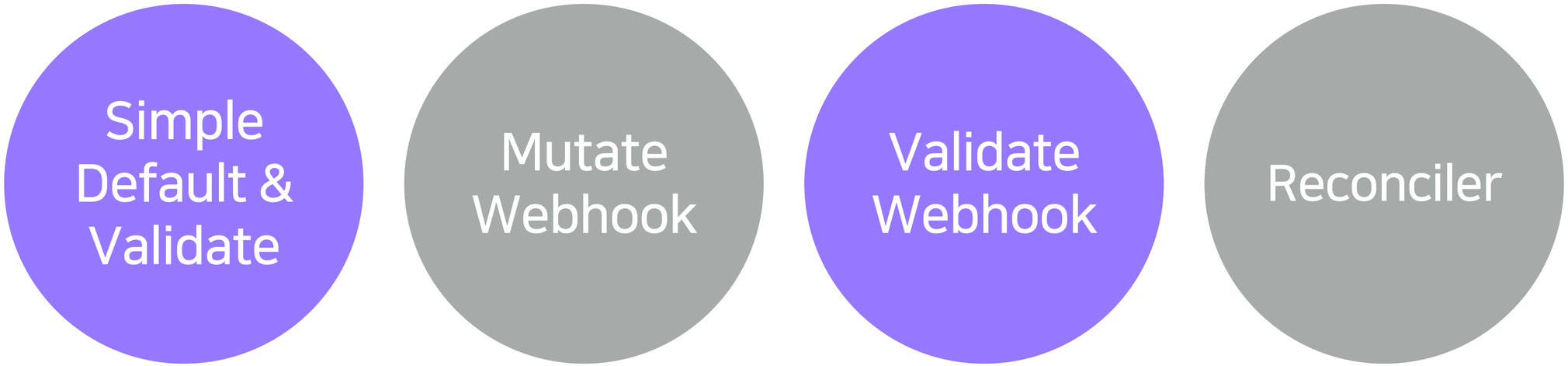
오브젝트 리소스의 현재 상태(current state)를 사용자의 요구 상태(desired state)와 동기화를 보장하기 위해 끊임없이 조정(reconcile)하는 행위자



2.4 Operator - Detail

오퍼레이터 메커니즘

- 정의 기반 단순 기본값 및 검증 => 필드 변경(기본값) 설정 => 검증 => 조정



2.4 Operator - Detail

Simple Default & Validate

- CRD Struct에 정의된 단일 필드에 대한 기본값 설정 및 검증

Mutate(Default) Webhook

- (리소스 생성) 타 리소스 및 복합 필드에 대한 기본값 설정 Webhook

Validate Webhook

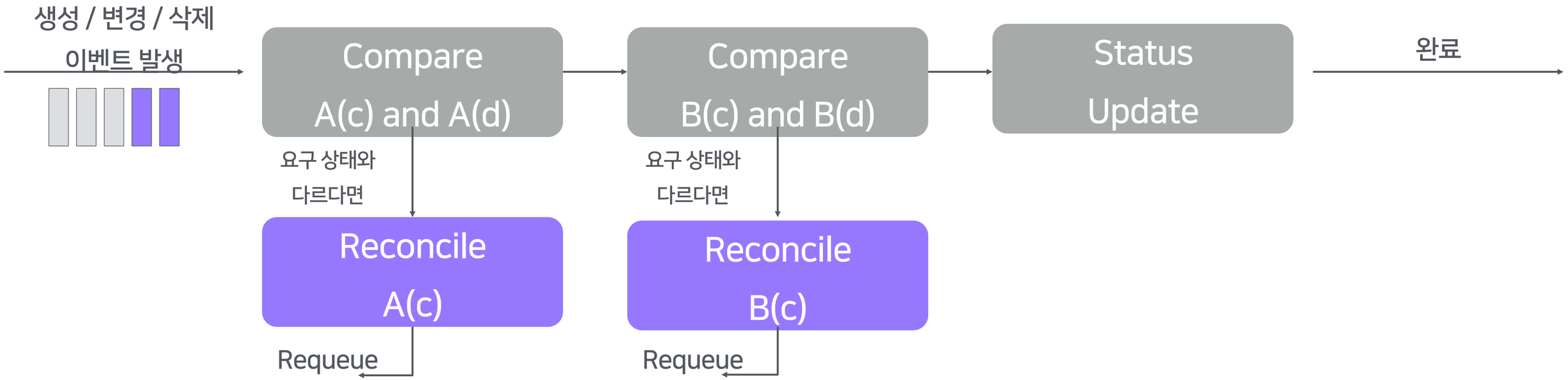
- 타 리소스 및 복합 필드에 대한 필드 검증 Webhook
- 리소스 생성, 업데이트, 삭제 시점에 따른 필드 검증

2.4 Operator - Detail

DEVIEW 2021

Reconciler

- 기본값 및 검증을 통과한 요구 상태를 현재 상태와 비교 및 동기화
- 현재 상태(c), 요구 상태(d)



2.4 Operator - Detail

DEVIEW 2021

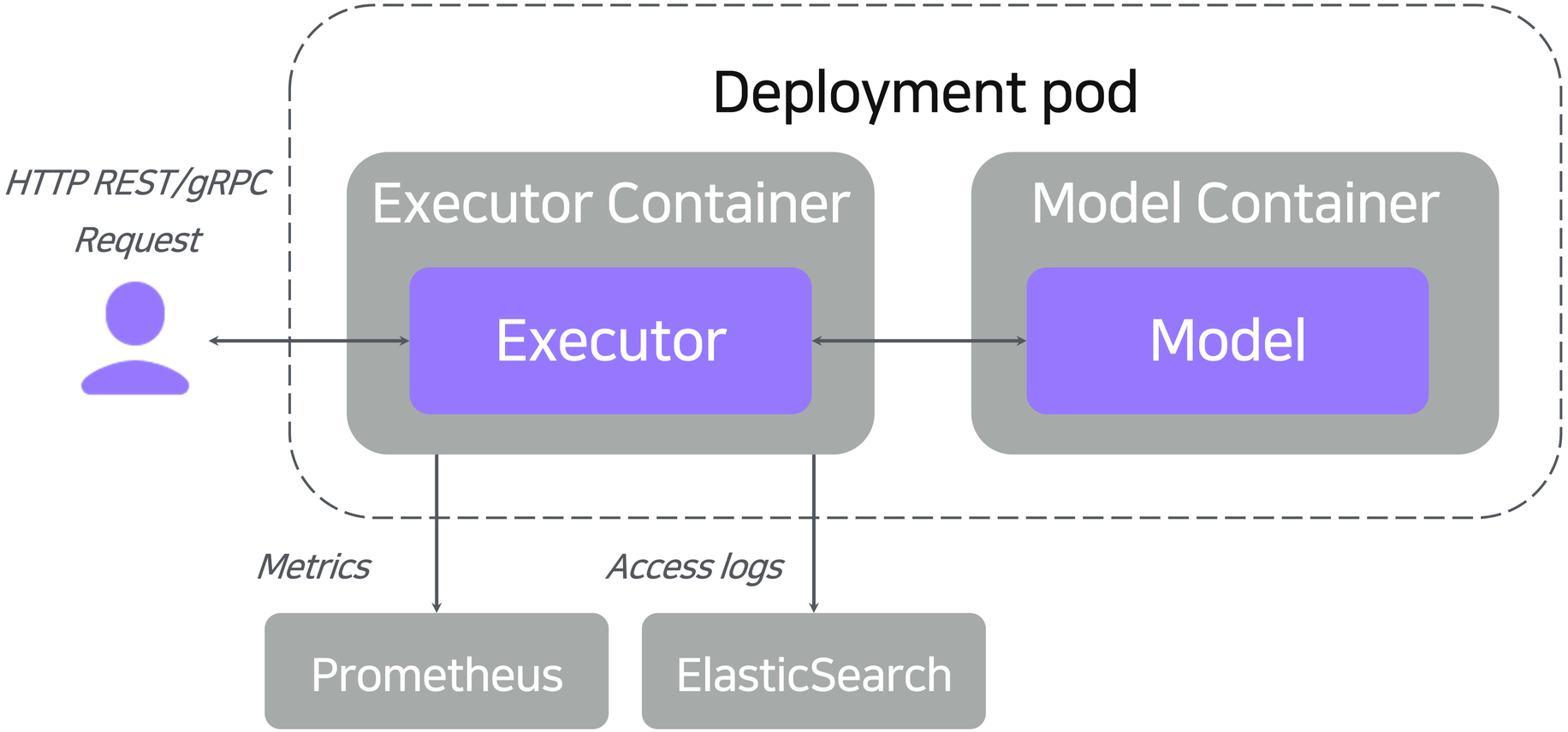
모델 배포 일반화 및 부가기능 주입

- 모델 어플리케이션 변경 없이 사이드카 형태로 Executor 주입
- 모델 엔드포인트 일반화
- 로깅, 메트릭과 같은 미들웨어 추가 용이
- 모델레지스트리 마운트, 요청 형태 제어, 요청량 제어 등
- 복잡한 모델을 그래프 추론 관리 및 조정

2.4 Operator - Where

- 모델 엔드포인트 일반화 및 미들웨어 주입

```
spec:
    predictor:
        registries:
        - name: detector
          mountPath: /data
        containers:
        - name: main
          image: alpine
    graph:
        entrypoint: main
```



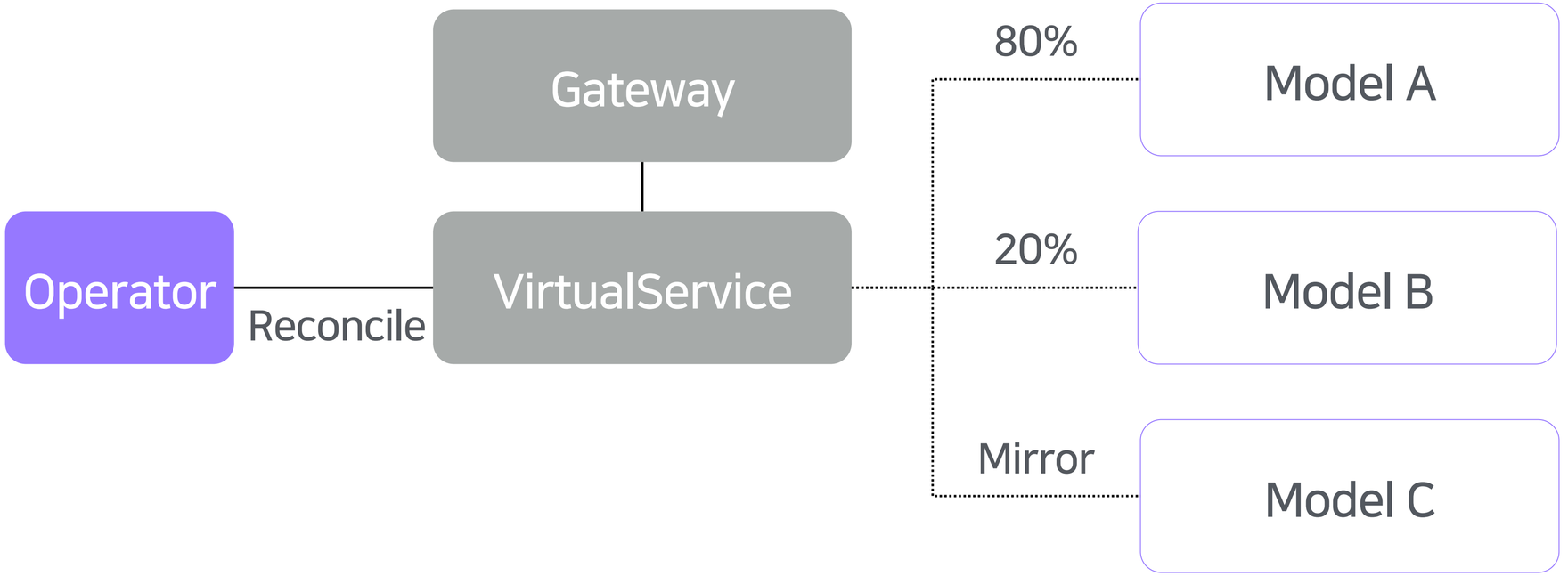
2.4 Operator - Where

DEVIEW 2021

트래픽 제어

- (모델러) 모델 트래픽 설정
- (오퍼레이터) A/B(N) Test, Mirror, Timeout, Retry 제어

```
spec:
    routes:
    - name: model-a
      weight: 80
    - name: model-b
      weight: 20
    mirror:
    - name: model-c
```



2.4 Operator - Where

DEVIEW 2021

로깅 포워드

- (모델러) 로깅 포워드 타겟 설정
- (오퍼레이터) 컨테이너 내 로깅을 타겟으로 포워딩 사이드카 주입

이미지 빌드

- (모델러) CLOps CLI 도커 빌드 입력
- (오퍼레이터) 이미지 빌드 파이프라인 실행 및 로컬 데이터 전송 자동화

2.4 Operator - Where

DEVIEW 2021

이외에도.. (추후 슬라이드에서 설명 예정)

- 오토스케일러
- 모델 레지스트리 및 캐시
- 모델 요청 형태 제어 (Sync / Async)
- 요청량 제어 (Provider / Consumer)

2.4 Operator - TIP

DEVIEW 2021

멱등성(Idempotent) 보장

- 여러번 실행하더라도 결과가 달라 지지 않도록 구현
- 요구 상태에 따른 변화는 항상 똑같아야한다.

컨트롤러 리소스 참조 관계

- 상위 리소스와의 관계를 ‘SetControllerReference’ 함수를 통해 참조 설정
- 리소스가 삭제될 때, 함께 Garbage Collection(GC)를 위한 필수사항

2.4 Operator - TIP

DEVIEW 2021

Custom Resource 필드 타입

- int32 / *int32, string / *string 차이 이해 필요
- Golang의 기본값 인식에 따른 비정상 작동 방지

리소스 비교 및 Reconcile

- 특정 필드가 런타임 시점에 값이 생성되거나 Immutable 필드이거나
- 비교 필드 특성에 따라 비교 로직 변경 필요
- 필드 비교 시 요구 상태와 현재 상태가 계속 달라 무한히 Reconcile 하는 문제 방지

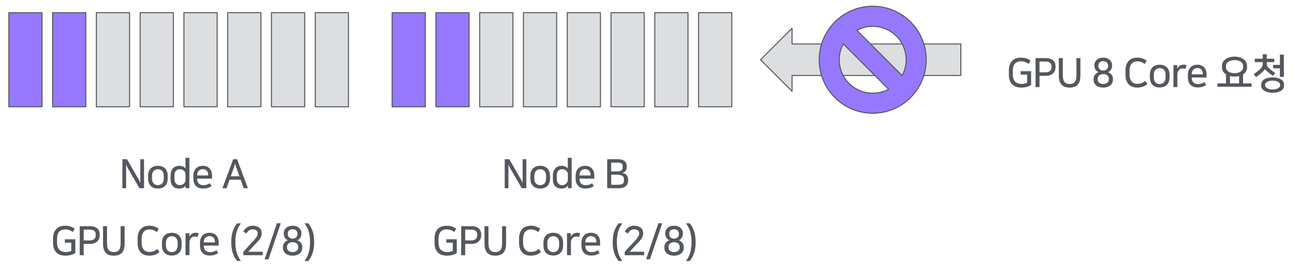
2.5 Scheduler - Why

프로젝트당 모델 종류에 따라 다양한 필요 리소스

- CPU / GPU 사용 여부, GPU 필요 코어, FP32 / FP16에 따른 아키텍처
- 프로젝트별 머신 타입 사용량 제한 스케쥴링

효율적인 리소스 관리를 위한 스케쥴링 정책

- GPU 노드 단편화 문제를 해결하여 가용 리소스 최대화
- 예) 8코어 노드에 2코어 애플리케이션 할당된 상황에서 8코어 애플리케이션 할당 요청



2.5 Scheduler - Detail

DEVIEW 2021

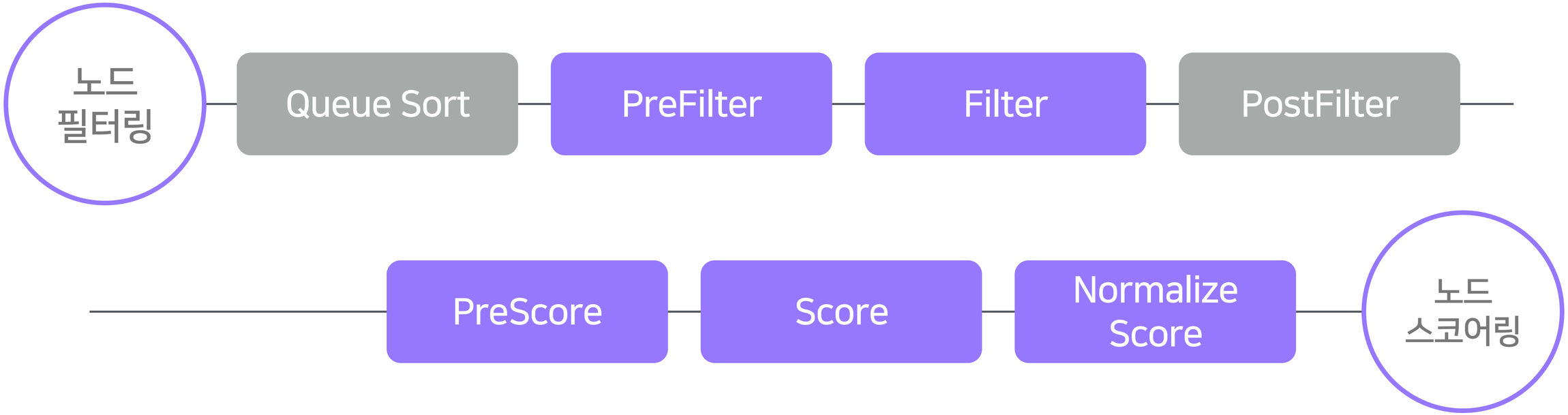
Scheduling Framework

- K8S 1.19 부터 지원하는 커스텀 스케쥴러 프레임워크
- 스케쥴링 노드 필터링 및 스코어링과 같은 플러그인 포인트 제공
- 노드 필터링, 스코어링, 예약, 할당 단계

https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/

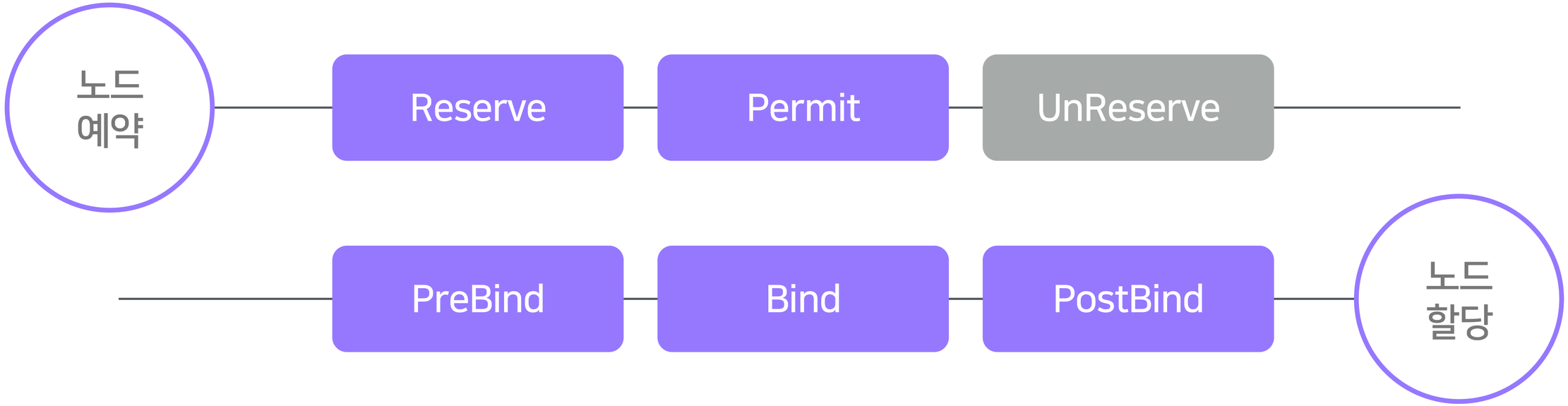
2.5 Scheduler - Detail

- 스케쥴링 프레임워크 워크플로우 (Extension points)



2.5 Scheduler - Detail

- 스케쥴링 프레임워크 워크플로우 (Extension points)

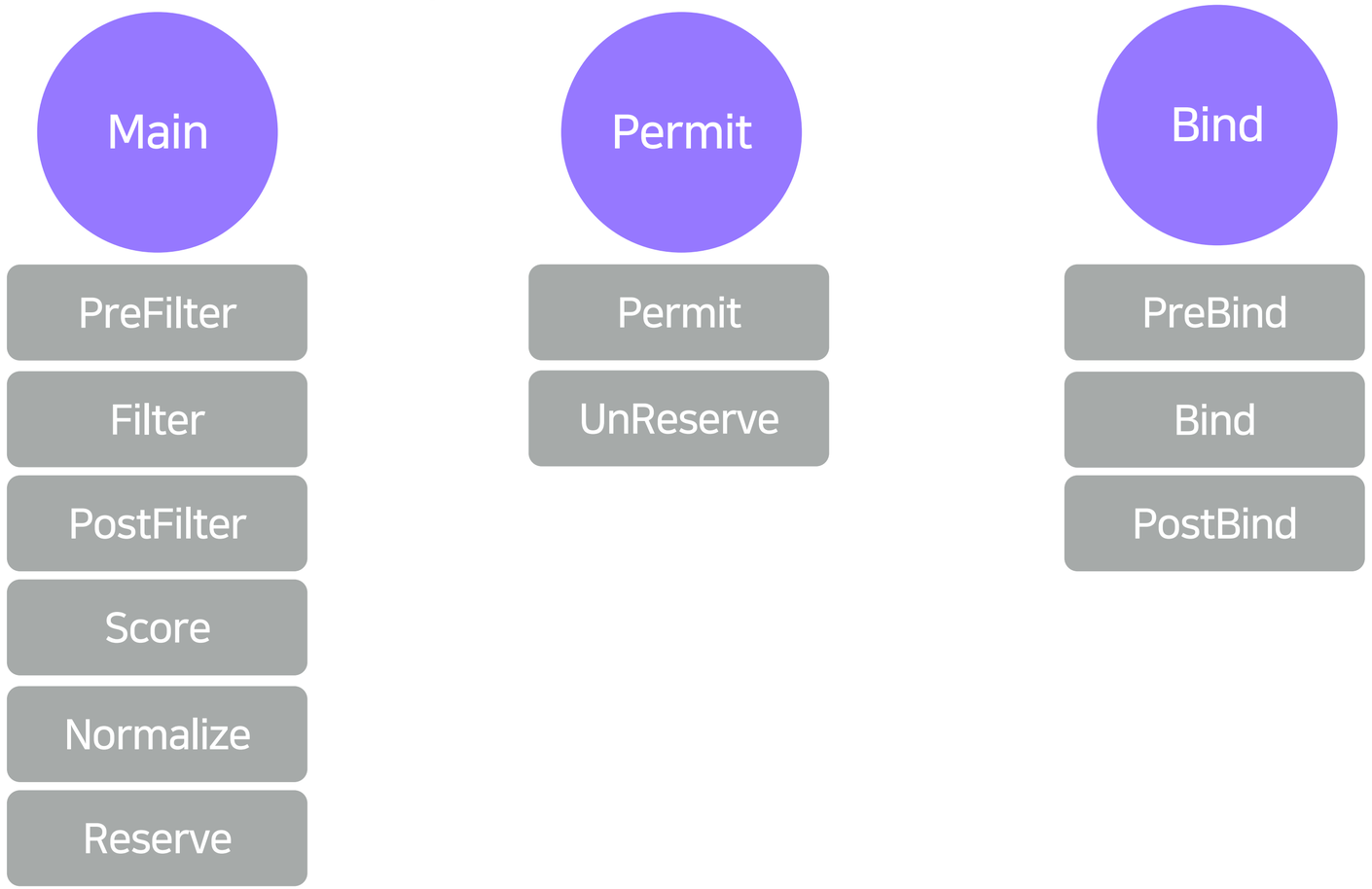


2.5 Scheduler - Detail

DEVIEW 2021

스케쥴링 프레임워크 스레드

- 세개의 스레드로 병렬 실행하며, 동일 스레드 내 실행 순서 및 단일 실행 보장



2.5 Scheduler - Where

DEVIEW 2021

프로젝트 별 노드 스케쥴링

- CR 기반 CLOps 서비스 내 프로젝트별 사용 가능한 노드 할당
- 단일 클러스터에서도 프로젝트(네임스페이스별) 노드 스케쥴링 가능

커스텀 스코어링

- 노드 할당을 위한 점수 부여 로직을 커스텀 적용
- 단편화 방지 및 가용 노드 확보

2.5 Scheduler - TIP

DEVIEW 2021

병렬 스레드 간 경쟁 상태가 없도록 구현

- 스레드 간 Extension Point 특성 파악 필요
- 스케쥴링 로직 내 리소스 경쟁 상태로 데드락 또는 데이터 부정합 발생 가능

단일 스케쥴 사이클 내 정보 공유

- CycleState 구조체를 통해 스케쥴링 라이프사이클 내 데이터 공유 가능
- 동일 데이터 API 호출로 인해 성능 저하 예방

2.5 Scheduler - TIP

Active / StandBy 사용 시, 리더 여부 확인

- 스케쥴러 내 추가적인 Informer 구성 시, 자체 리더 선출 리소스를 사용하여 구현 필요
- 스케쥴링 프레임워크 내 구현부에서 리더 여부를 알 수 없는 구조

노말라이즈 스코어 단계 고려 사항

- 노말라이즈를 위해 스코어에 대한 가중치에 대한 고려 필요
- 기본 스코어링도 함께 작용하기 때문에 의도치 않은 스케쥴링이 될 수 있음

2.6 Horizontal Pod Autoscaler (HPA) - Why

DEVIEW 2021

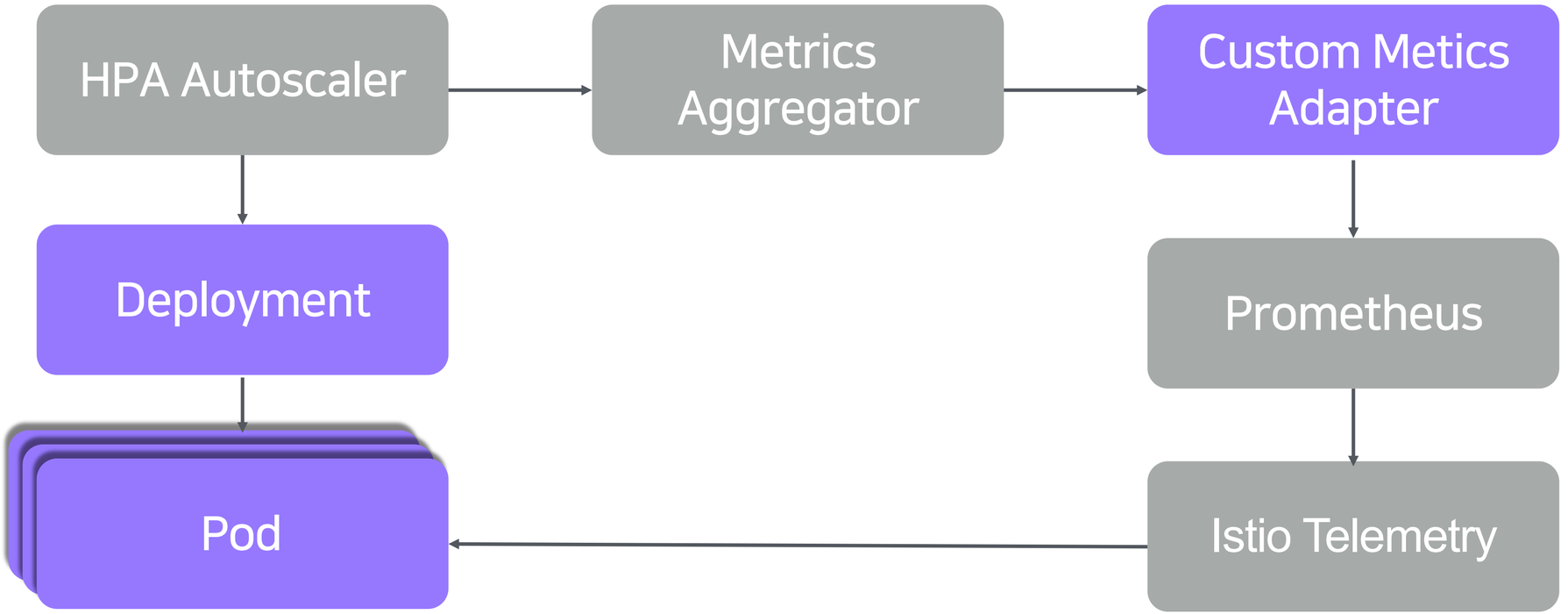
기존 HPA 항목의 한계

- CPU/메모리 기반 HPA 는 모델 서빙을 위한 HPA 메트릭으로 적합하지 않음
- 사용자 정의 (요청수, Latency, GPU 등) 기반 HPA 메트릭 필요

2.6 Horizontal Pod Autoscaler (HPA) - Detail

DEVIEW 2021

- HPA Metrics Aggregator 구조



2.6 Horizontal Pod Autoscaler (HPA) - Where

요청량에 따른 오토스케일링

- (모델러) 요청량 임계값(Threshold) 설정
- (HPA) 임계값을 기준으로 스케일아웃 / 인 진행

```
spec:
    hpa:
        maxReplicas: 10
        minReplicas: 1
        metrics:
        - resource:
          name: hpa-exam
          target:
            type: RequestRate
            averageUtilization: 60
```

2.7 Model Registry - Why

Contains trained ML models

- Allows us to easily access the models to deploy from one location

Model versioning

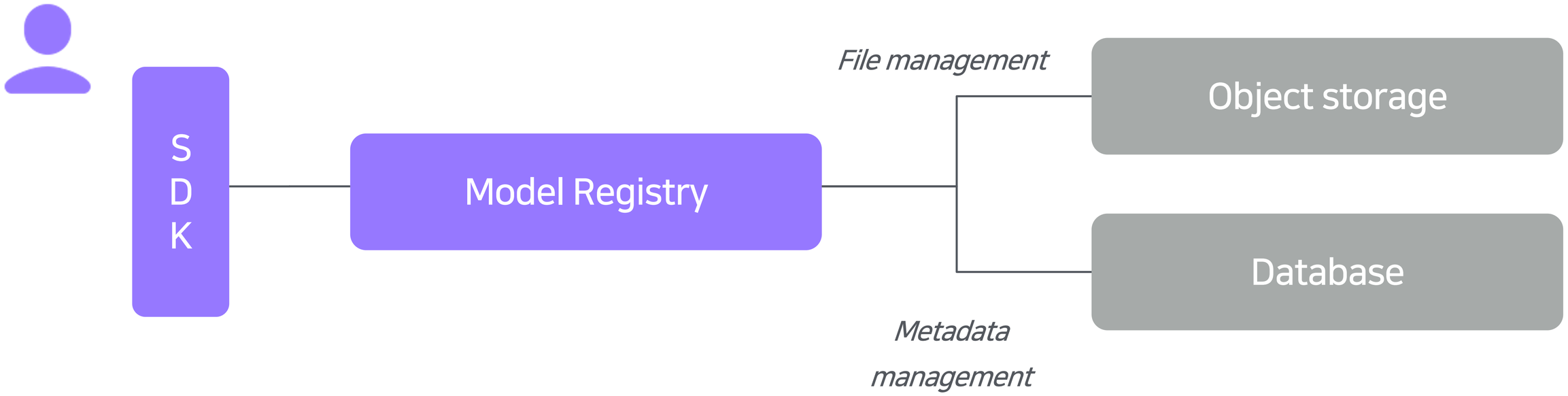
- Easy rollback to previous versions
- Enables *A/B/n* testing between different versions of same model

Model metadata

- Ability to trace which in environment model was trained and with what dataset *(reproducibility)*

2.7 Model Registry - Detail

- Client SDK for interacting with registry
- Store files and metadata in respective backends

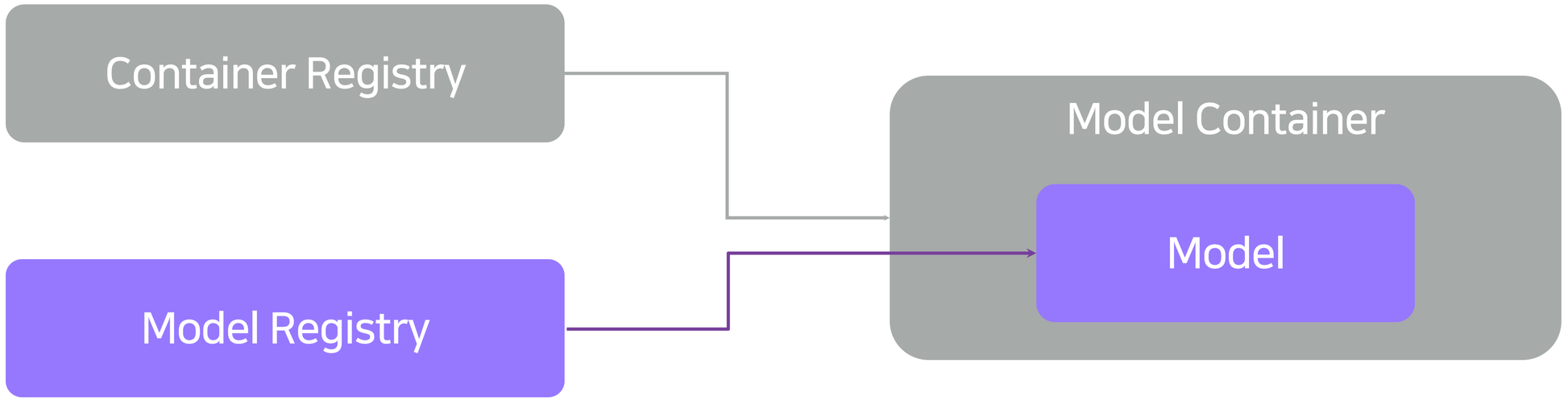


2.7 Model Registry - Detail

DEVIEW 2021

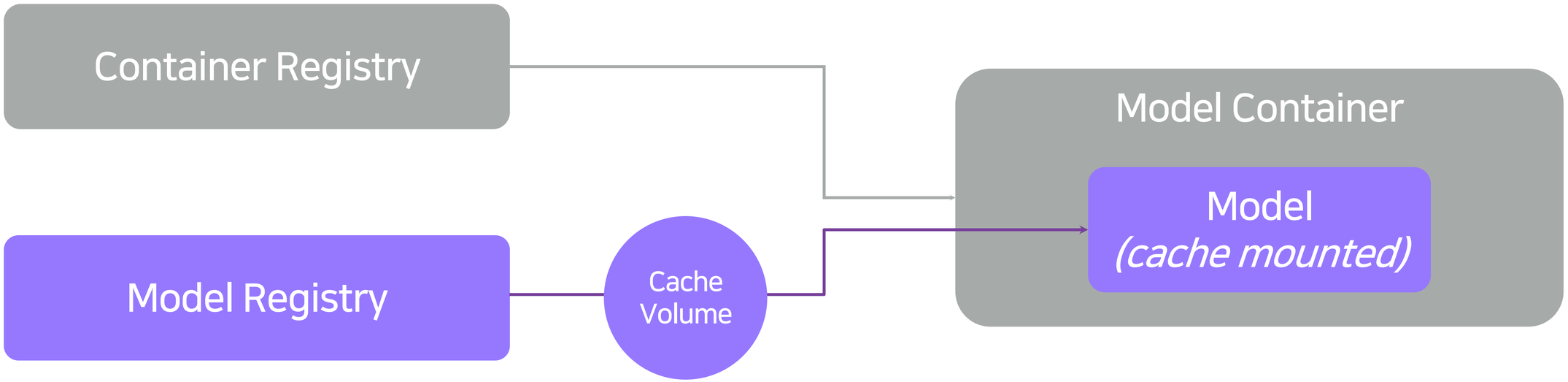
Model downloaded when deploying

- Separate the trained model and its execution environment
- Easier to manage with their own life-cycles



2.7 Model Registry - Detail

- For large models caching is supported



2.7 Model Registry - TIP

Parallel upload/download

- Many models are large in size and uploading/downloading can be time consuming
- Using multiple threads can increase total throughput

- When to cache models
- Network storage is usually more expensive than ethereal *(local) disk storage*
- Consider if caching is actually required when deploying each model
- Highly critical or very large models are good candidates

2.8 Synchronous/Asynchronous - Why

DEVIEW 2021

Synchronous predictions

- Client requests and waits until prediction is done
- Default prediction mode

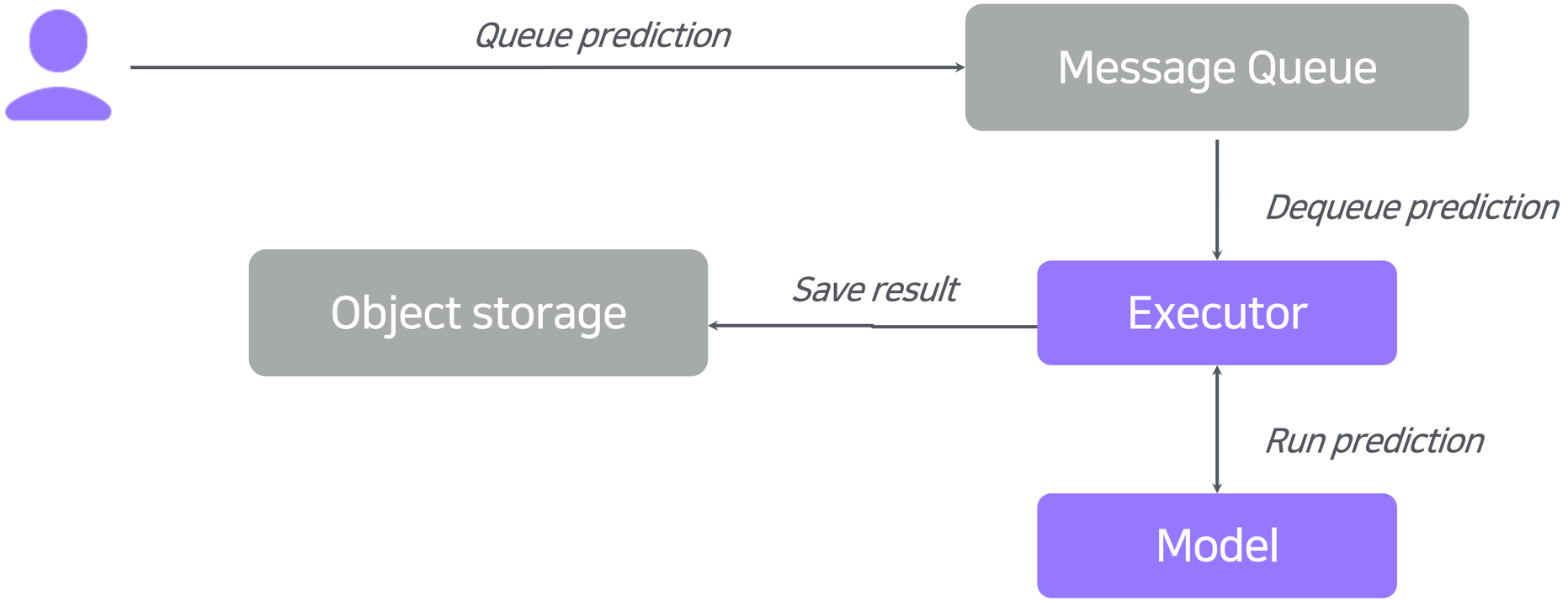
Asynchronous

- Client queues predictions
- Can retrieve result when prediction is done
- Preferred when prediction workloads take long *(e.g. video processing)*

2.8 Synchronous/Asynchronous - Detail

DEVIEW 2021

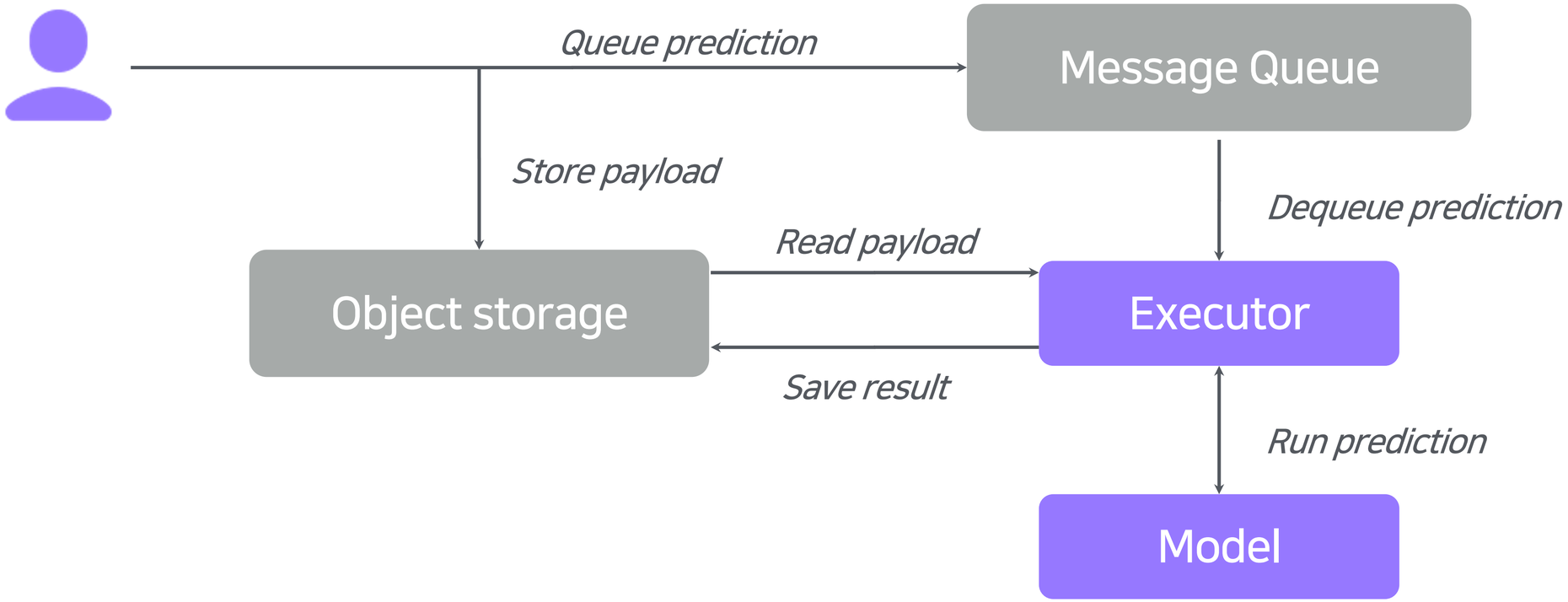
- Use message queue as asynchronous input



2.8 Synchronous/Asynchronous - Detail

DEVIEW 2021

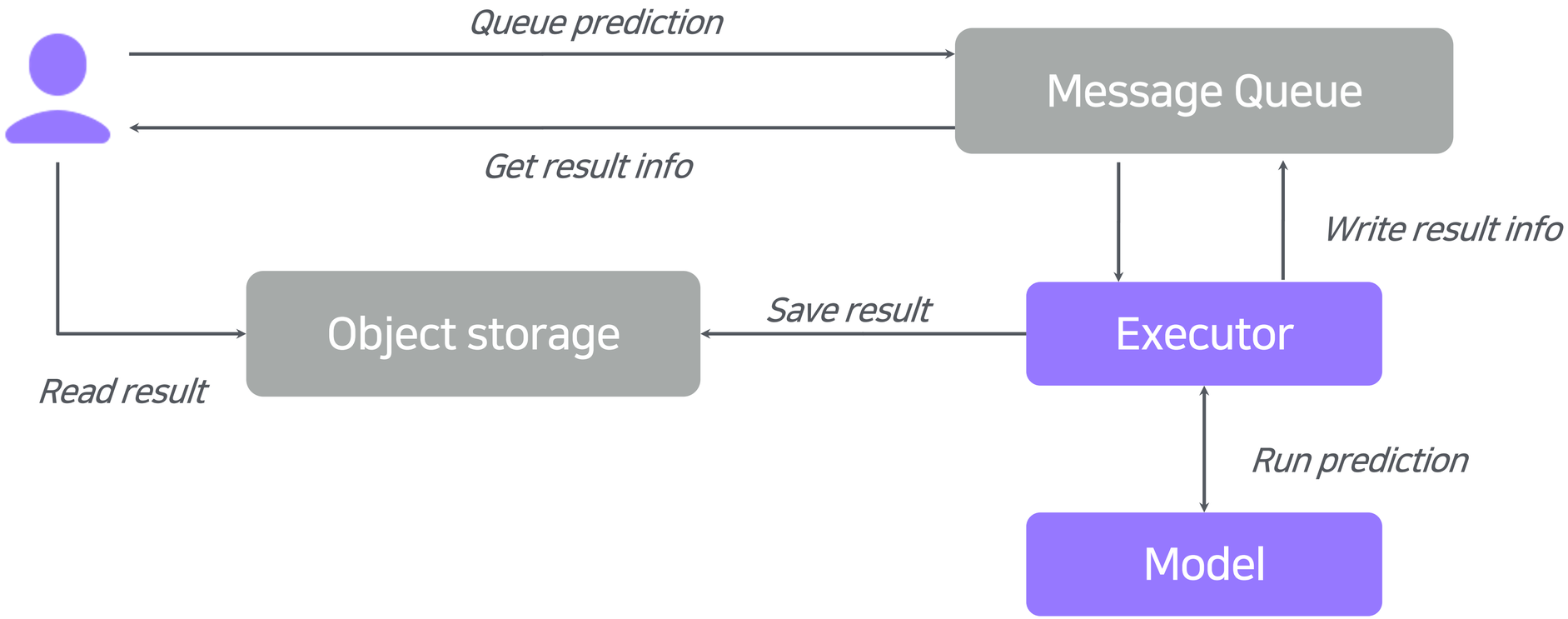
- Store prediction payload if request body is large



2.8 Synchronous/Asynchronous - Detail

DEVIEW 2021

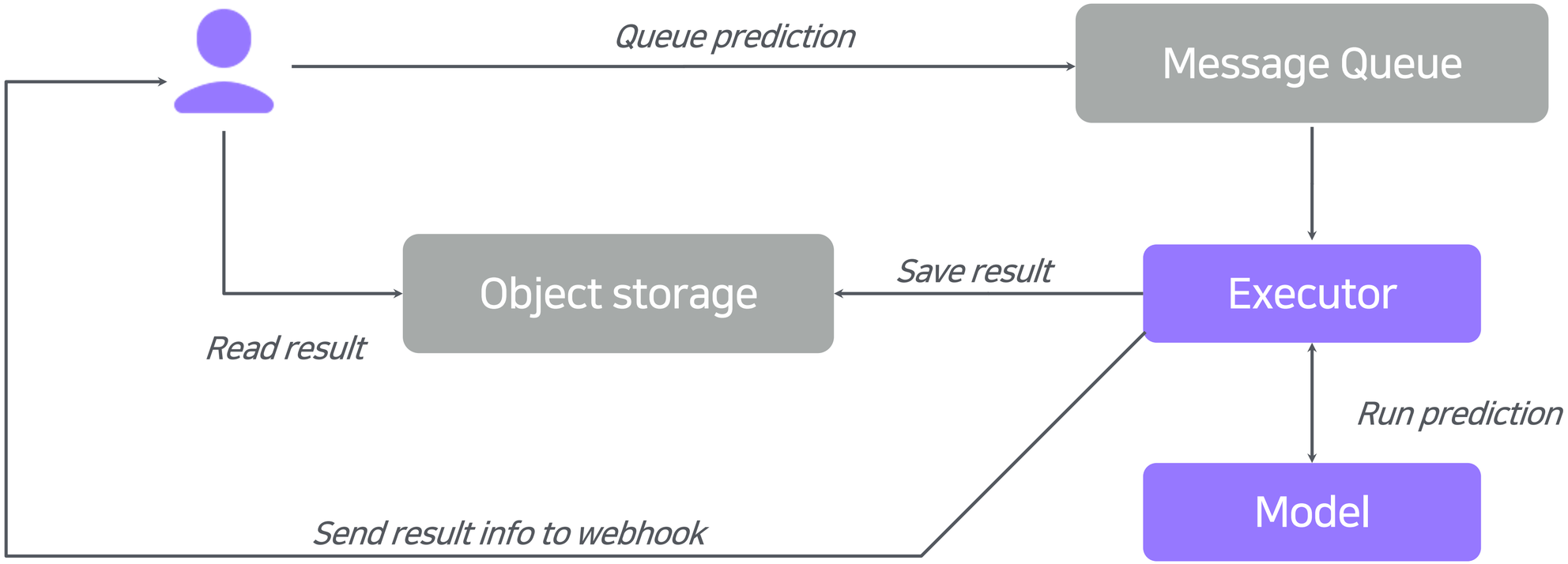
- Get result information via message queue



2.8 Synchronous/Asynchronous - Detail

DEVIEW 2021

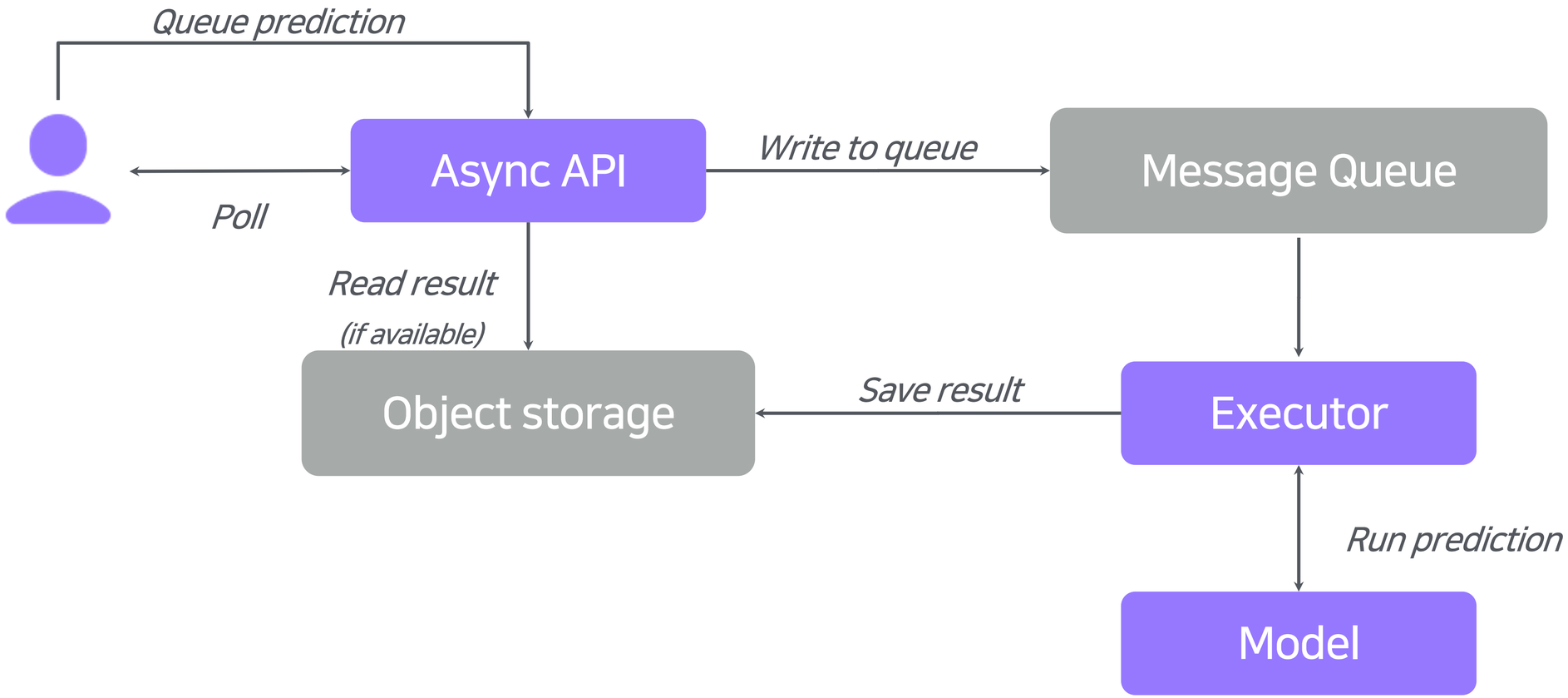
- Get result information via webhook



2.8 Synchronous/Asynchronous - Detail

DEVIEW 2021

- Using Async API server



2.9 Provider / Consumer - Why

DEVIEW 2021

Provider

- Used to represent a model service
- Ability to control request access modes and rate limits for endpoints
- Optionally includes a description and usage guide

Consumer

- Allows us to identify requesting clients
- Linked to organizations or individual users
- Enables auditing and access control per client
- Contains API tokens

2.9 Provider / Consumer - Detail

DEVIEW 2021

Provider access modes

Private
- Only certain authorized clients can access model API
- Provider can choose exactly which consumers have access

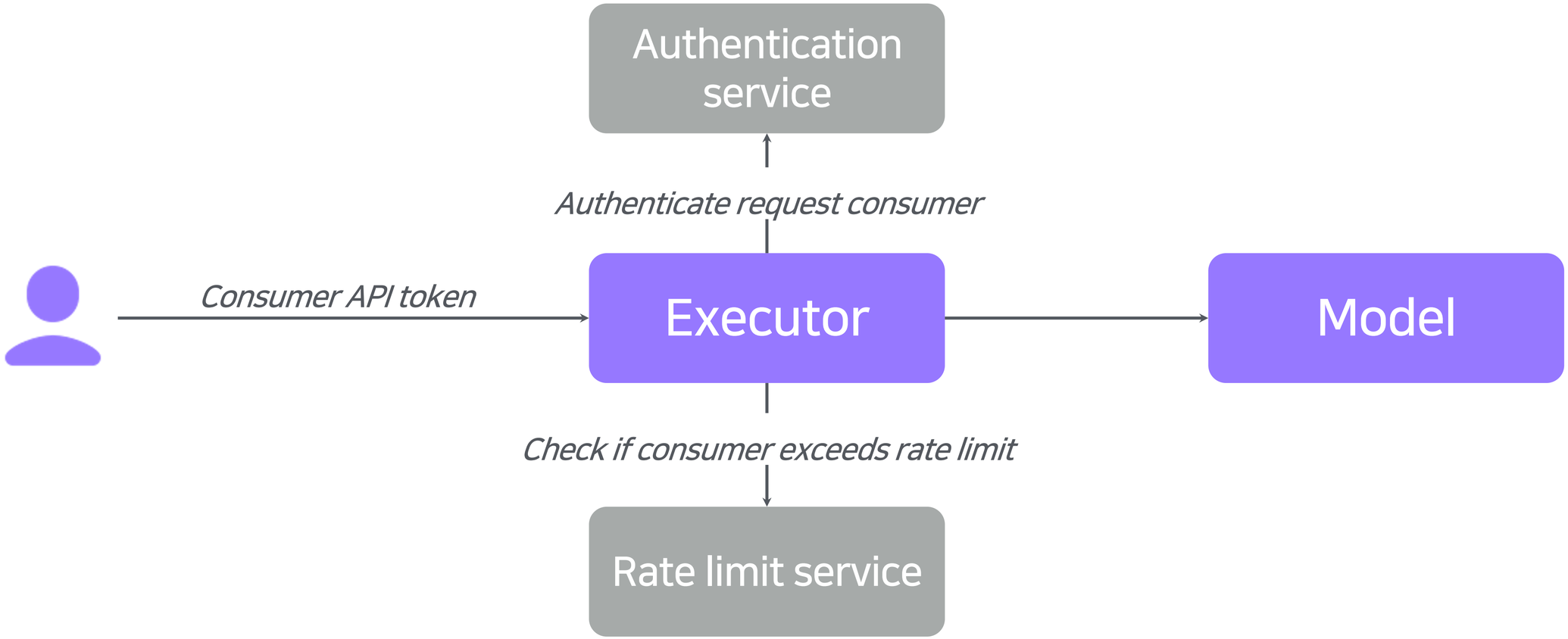
Permissive
- Any request with an API token allowed
- Allows provider owner to audit all requests

Public
- All requests are allowed
- Still authenticates API tokens for auditing if possible

2.9 Provider / Consumer - Detail

DEVIEW 2021

Consumer authentication via API token



2.9 Provider / Consumer - TIP

DEVIEW 2021

Try to utilize service mesh capabilities

- If using a service mesh, try to see if built-in mechanisms fits your use case

Build service catalogs

- With list of providers it is possible to create a view of all service providers
- Gives clear view to model consumers of which services are available

3.Improvements

3.1 What improved?

DEVIEW 2021

Allocating server resources

- *(Before)* PM/VM equipment ordering and installation
- *(Now)* Various machine type pools that can be assigned as needed

Deploying models

- *(Before)* Install environment and implement specific middleware
- *(Now)* Runtime container image and YAML configuration

3.1 What improved?

DEVIEW 2021

Model logging/monitoring management

- *(Before)* Implement logging and monitoring forwarding
- *(Now)* Simply set forwarding information in YAML

Model networking

- *(Before)* API Gateway, discovery based infra with routes in DB
- *(Now)* Networking configured directly through YAML resources

3.1 What improved?

DEVIEW 2021

Model scalability

- *(Before)* LB in front model, manually add machines to scale out
- *(Now)* Conditional in/out scaling for models

Model registry

- *(Before)* Models spread out in different storages
- *(Now)* Upload/download via CLI/SDK, central store with metadata tracking

3.1 What improved?

DEVIEW 2021

Model operation management

- (Before) Manage server access rights, individual operations by admin
- (Now) Management authority for models and load controllable by user and linked to in-house authentication

4.Next Challenges

4.1 Next Challenges

DEVIEW 2021

Pipelining

- Data collection + Model training + Model inference
- MLOps Level 2

Model graph

- Define flow for complex predictions involving multiple models

Model interface SDK

- Provide abstract model interface to improve model compatibility and reusability

WE ARE HIRING

https://naver-career.gitbook.io/kr/service/clova/mlops

clova-jobs@navercorp.com

Thank You