



OpenShift ACM





Open Cluster Manager

TRAINING

CHOI GOOK HYUN

Freelancer Instructor

DAY 1

- ▶ About the OpenShift Advanced Cluster

THE FEDERATION

THE FEDERATION

with kubernetes

"The Federation" API는 올해 2022년도 10월에 더 이상 개발을 하지 않기로 결정하였음. 다만, API사양은 계속 개발하며, 프로그램 부분은 OCM, KARMADA와 같은 커뮤니티에 개발을 맡기기로 결정.

공식적으로 Federation API는 v1, v2기반으로 되어 있으며, 쿠버네티스는 API정의 애플리케이션 개발은 외부 그룹에서 맡기로 하였음.

현재는 다음과 같이 두 그룹이 주요 리더로 끌고 가고 있음.

- ▶ Open Cluster Manager
- ▶ Karmada

THE FEDERATION

with kubernetes

Follow-up: discussion on archiving Kubefed 조회수 177회



pmo...@gmail.com

받는사람 kubernetes-sig-multiclustar

Everyone-

In recent Kubecon SIG updates we have previously communicated that SIG leadership has been considering archiving the Kubefed[1] repository. This topic was given discussion offline. This communication is a follow-up on that action item.

TLDR: Our current favored option is to archive the kubefed project but we are soliciting community input before making a final decision. Continue reading for rationale.

In our discussions, Jeremy and I considered the following questions:

KARMADA

KARMADA

with kubernetes

KARMADA는 쿠버네티스 federation v1, v2프로토콜 기반으로 작성이 되어 있음. 쿠버네티스 기반으로 동작하며, 동작 방식은 OCM과 매우 흡사하나, 중간에 "Karmada API Server"중심으로 "etcd", "scheduler"와 통신하며 중간에 "Karmada controller"를 통해서 정책 배포 및 관리를 한다.

현재 KARMADA는 "CLOUD NATIVE"의 샌드박스(Sandbox) 프로젝트로 선언이 되어 있다. CNCF에서는 좀 더 KARMADA를 지원하고 있는 분위기.

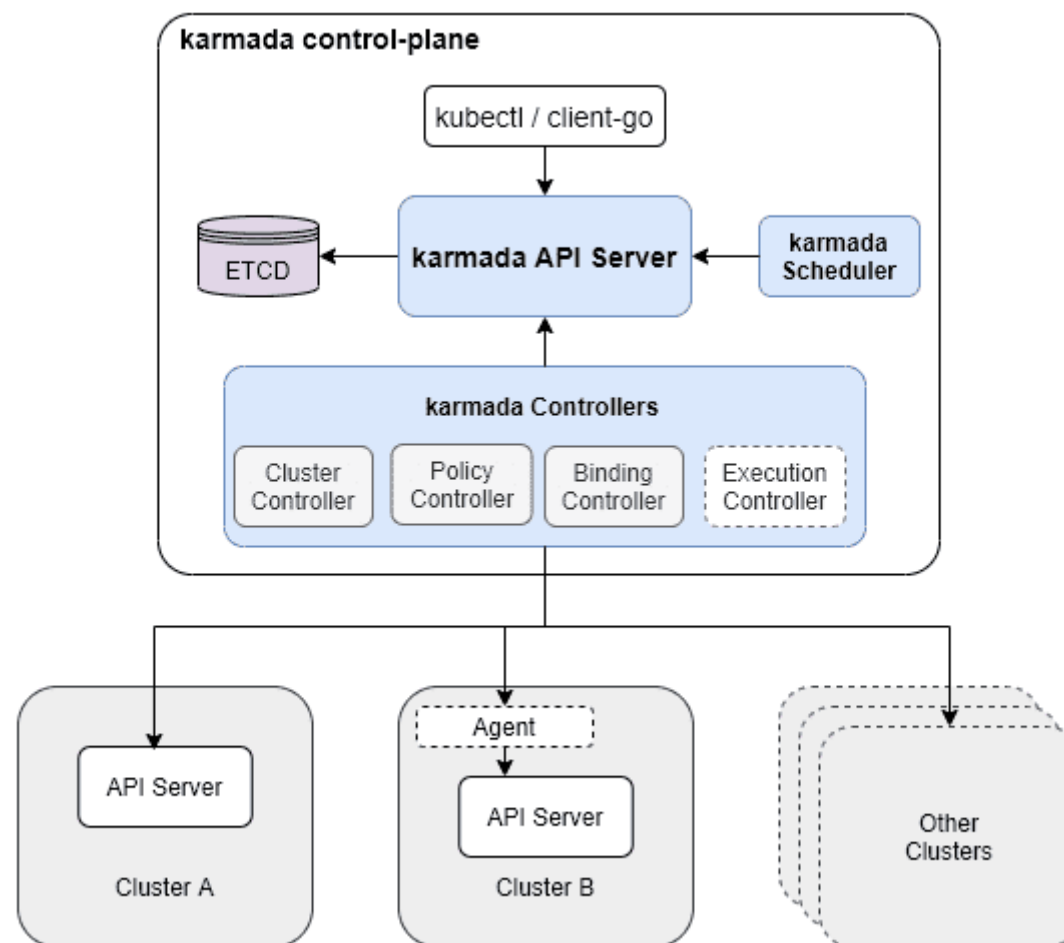
KARMADA

with kubernetes

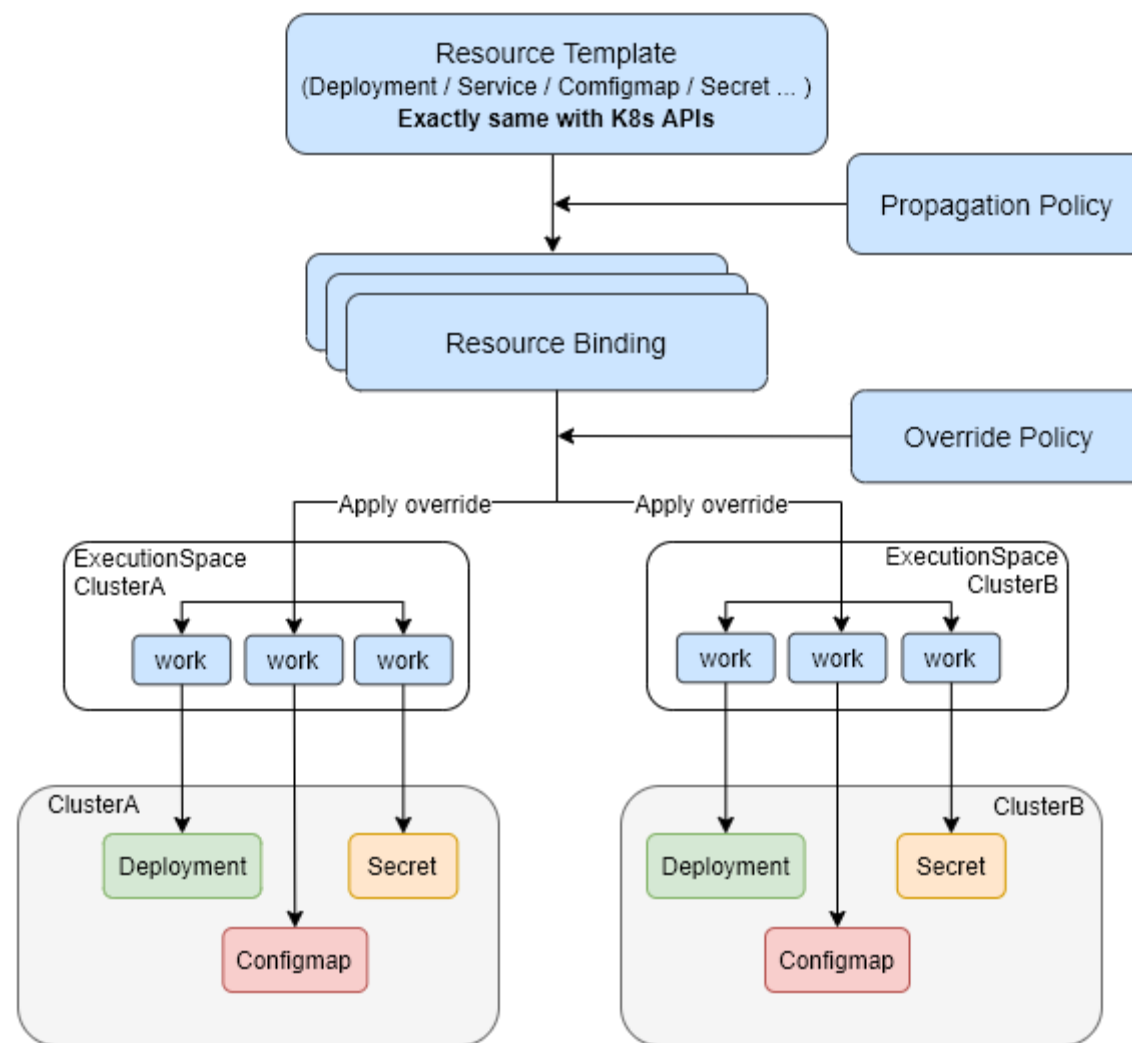
주요 목적은 다음과 같다.

- ▶ KIS native API compatible
- ▶ Out of the Box
- ▶ Avoid Vendor Lock-in
- ▶ Centralized Management
- ▶ Fruitful Multi-Cluster Scheduling Policies
- ▶ Open and Neutral

KARMADA



Karmada Concepts



OCM

OCM

with OpenShift

OpenShift는 Kubernetes기반으로 작성이 되어 있음. API는 두 가지 API로 구별이 됨.

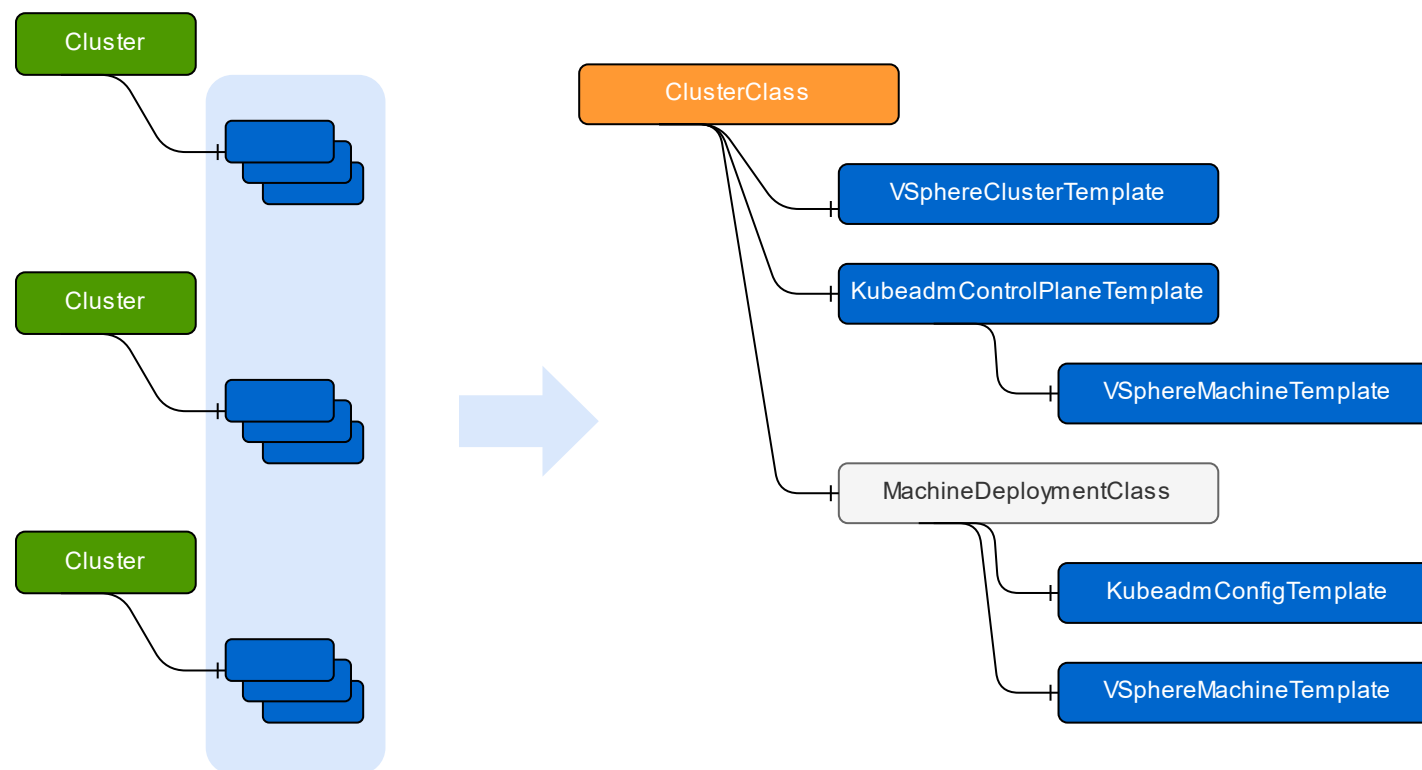
- ▶ Cluster API
- ▶ Federation API

Cluster API는 Kubernetes SIG에서 맨 처음 릴리즈 한 API. Federation API는 다중 클러스터 운영을 위해서 SIG에서 개발을 시작. OCM에는 현재 레드햇이 개발을 참여하고 있으며, 이 프로젝트도 "CLOUD NATIVE COMPUTING FOUNDATION"의 일부분.

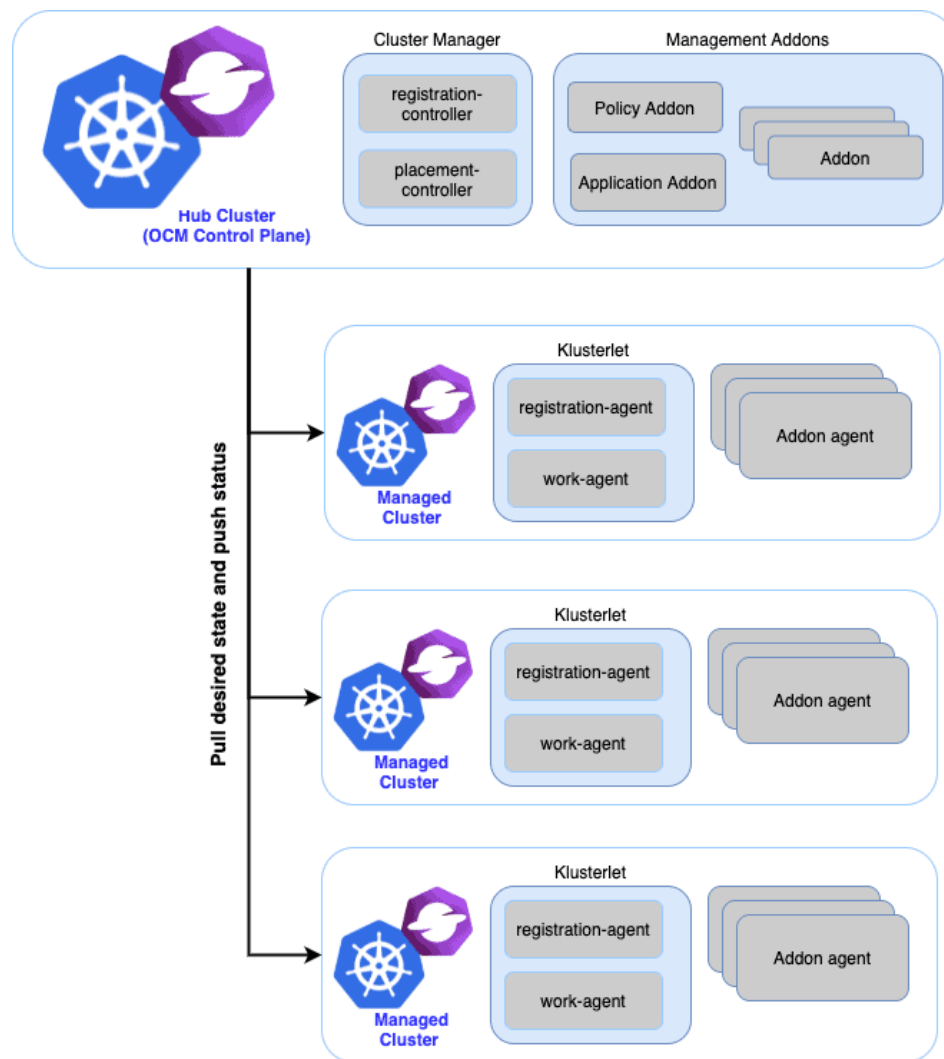
프로젝트 주소

<https://github.com/stolostron>

Federation



OCM Arch



OCM

MultiCluster API

클러스터 API는 다음과 같은 기능을 기본적으로 다루고 있다.

- ▶ 자원을 라이프 사이클(LifeCycle) 관리
- ▶ 설정 및 정책(Deployment, Policy) 적용 및 배포
- ▶ Pod 및 컨테이너 복제
- ▶ 네임스페이스 관리

OCM

Federation V1

Fede v1는 SIG에서 멀티 클러스터를 관리하기 위해서 프로젝트를 시작하였다. 쿠버네티스 SIG에서 제일 큰 프로젝트 중 하나이었으며, 앞서 말한 것처럼 멀티 클러스터 관리를 위해서 v1를 설계를 하였다. 주요 목적은 다음과 같다.

- Difficulties in re-implementing the Kubernetes API at the cluster level, as federation-specific extensions were stored in annotations.
- Limited flexibility in federated types, placement and reconciliation, due to 1:1 emulation of the Kubernetes API.
- No settled path to GA, and general confusion on API maturity; for example, Deployments are GA in Kubernetes but not even Beta in Federation v1.

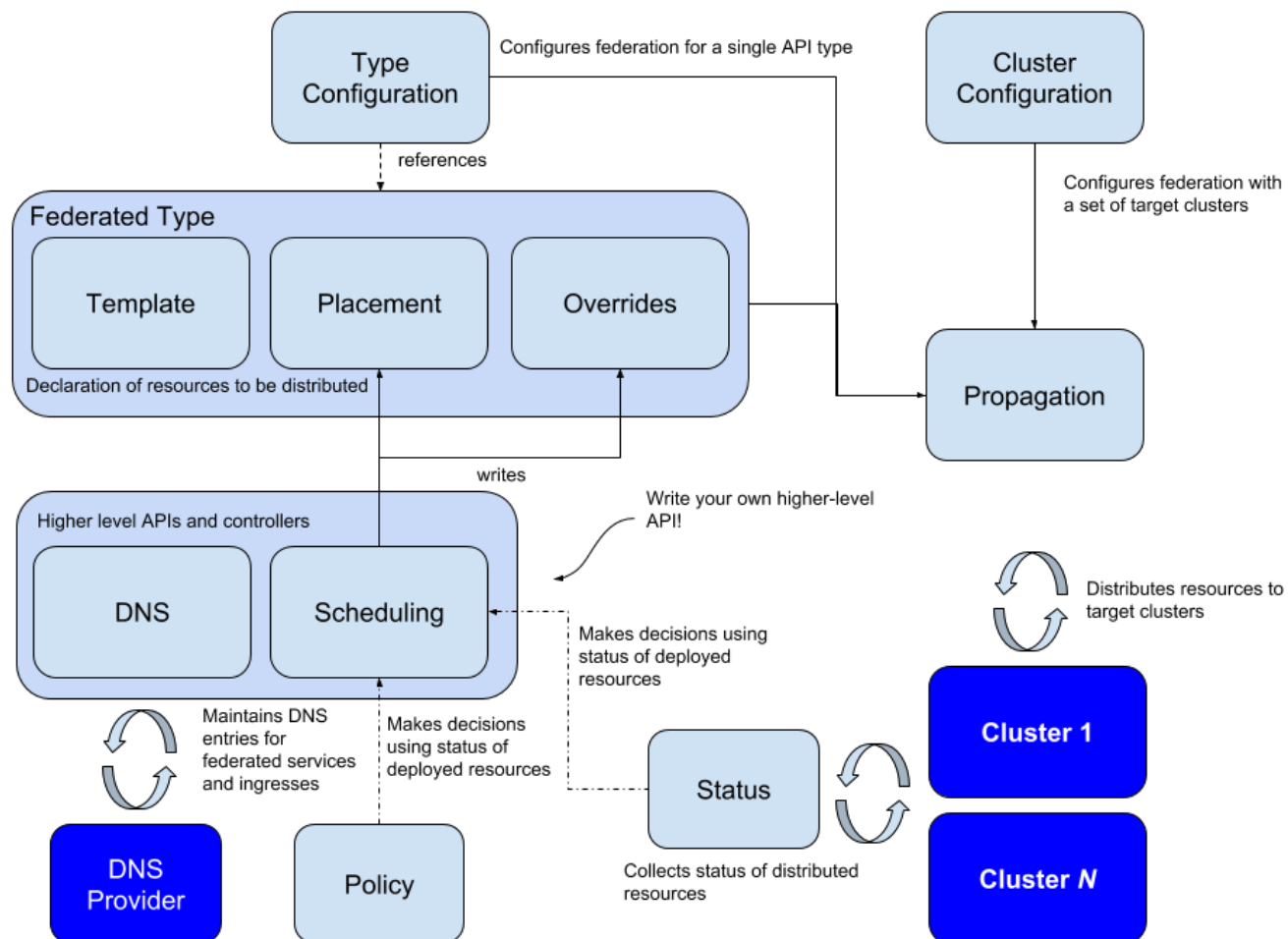
OCM

Federation V1

Federation은 v1기반으로 v2로 진화가 되었으며, 기본적인 컨셉은 v1에서 가져와서 v2에서 기능 추가를 시작하였다.

Federation은 "Hub and Spoke(pull and push)"구조를 가지고 있고 다음과 같이 동작한다.

Native Federation Diagram



OCM

Federation

연합의 주요 목표는 쿠버네티스 자원을 연합 그리고, "API to API"를 그룹으로 정의 하는게 주요 목표이자 목적이다.

쿠버네티스의 기능을 확장하기 위해서 **CRD(CustomResourceDefinitions)**를 통해서 구현이 된다. Federation API를 통해서 요청이 들어오면, 단일 클러스터 API를 배포하는 메커니즘이다.

Kubernetes

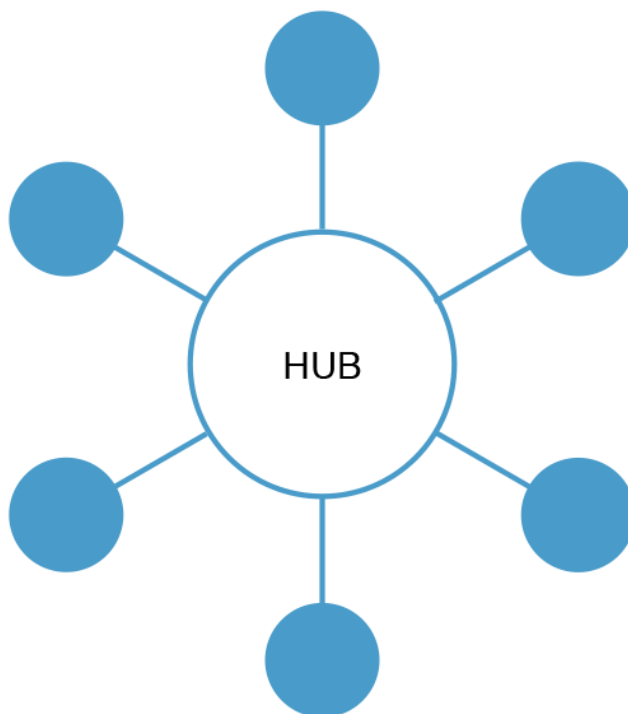
Federation

During the journey of defining building blocks of the federation APIs, one of the near term goals also evolved as 'to be able to create a simple federation a.k.a. simple propagation of any Kubernetes resource or a CRD, writing almost zero code'. What ensued further was a core API group defining the building blocks as a Template resource, a Placement resource and an Override resource per given Kubernetes resource, a TypeConfig to specify sync or no sync for the given resource and associated controller(s) to carry out the sync.

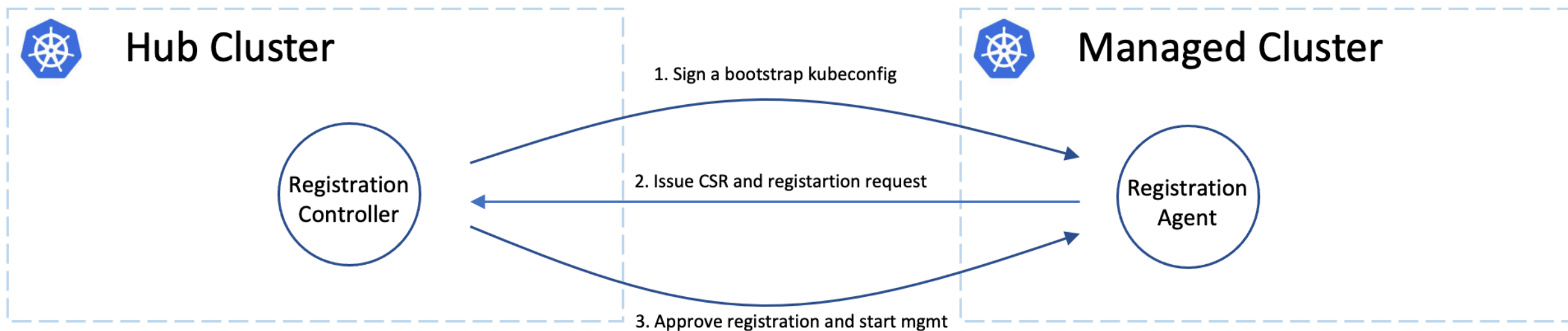
간단한 요약

CRD기반으로 별도의 코드 작성이 필요 없이 CORE API에게 정의하여 자원을 배포하는 형식. 이때 템플릿 자원들로 빌드 블록을 만들어서 Placement자원을 각각 쿠버네티스 자원에 오버라이드 하여 동기화 혹은 비동기화 여부를 컨트롤러를 통해서 결정

HUB and SPOKE



HUB and SPOKE



Template

Federation

Federation에서 주요 자원 지시자는 다음과 같다.

- ▶ Template
 - A Template type holds the base specification of the resource – for example, a type called **FederatedReplicaSet** holds the base specification of a **ReplicaSet** that should be distributed to the targeted clusters

Placement

Federation

- ▶ Placement
 - A Placement type holds the specification of the clusters the resource should be distributed to – for example, a type called **FederatedReplicaSetPlacement** holds information about which clusters **FederatedReplicaSets** should be distributed to

Override

Federation

- ▶ Override
 - An optional Overrides type holds the specification of how the Template resource should be varied in some clusters - for example, a type called **FederatedReplicaSetOverrides** holds information about how a **FederatedReplicaSet** should be varied in certain clusters.

TypeConfig

Federation

- ▶ TypeConfig
 - specify sync or no sync for the given resource and associated controller(s) to carry out the sync.

Federation v2

Higher-level behaviour

API v2에서 핵심 기능은 API에서 "템플릿", "배치" 및 "오버라이드"를 통해서 컨트롤러(Controller)통해서 상위 API를 재구성(CRD) 합니다.

클러스터를 가로지르면서 자원을 생성 시 "ReplicaSchedulingPreference"를 통해서 Deployment 및 ReplicaSet를 통해서 연합 클러스터에 자원을 생성 및 구성이 가능하다. 역시 기존 클러스터에서 사용하였던 기능 "limits", "quote"같은 자원 관리도 API를 통해서 배포가 가능하다.

OpenShift vs OCM

Components

오픈 시프트의 ACM은 쿠버네티스의 Federation기반으로 되어 있으며, 레드햇은 OCM기반으로 ACM를 개발하기 시작 하였다. 레드햇 ACM은 다음과 같은 부분에 OCM과 다르다.

- ▶ 손쉬운 설치 및 오픈 시프트와 통합된 환경
- ▶ 클라우드 환경에서 손쉬운 확장
- ▶ Quay.io, Ceph, InSight(OpenScape)와 같은 에코 환경 구성
- ▶ 강화된 보안모델 제공

OpenShift ACM

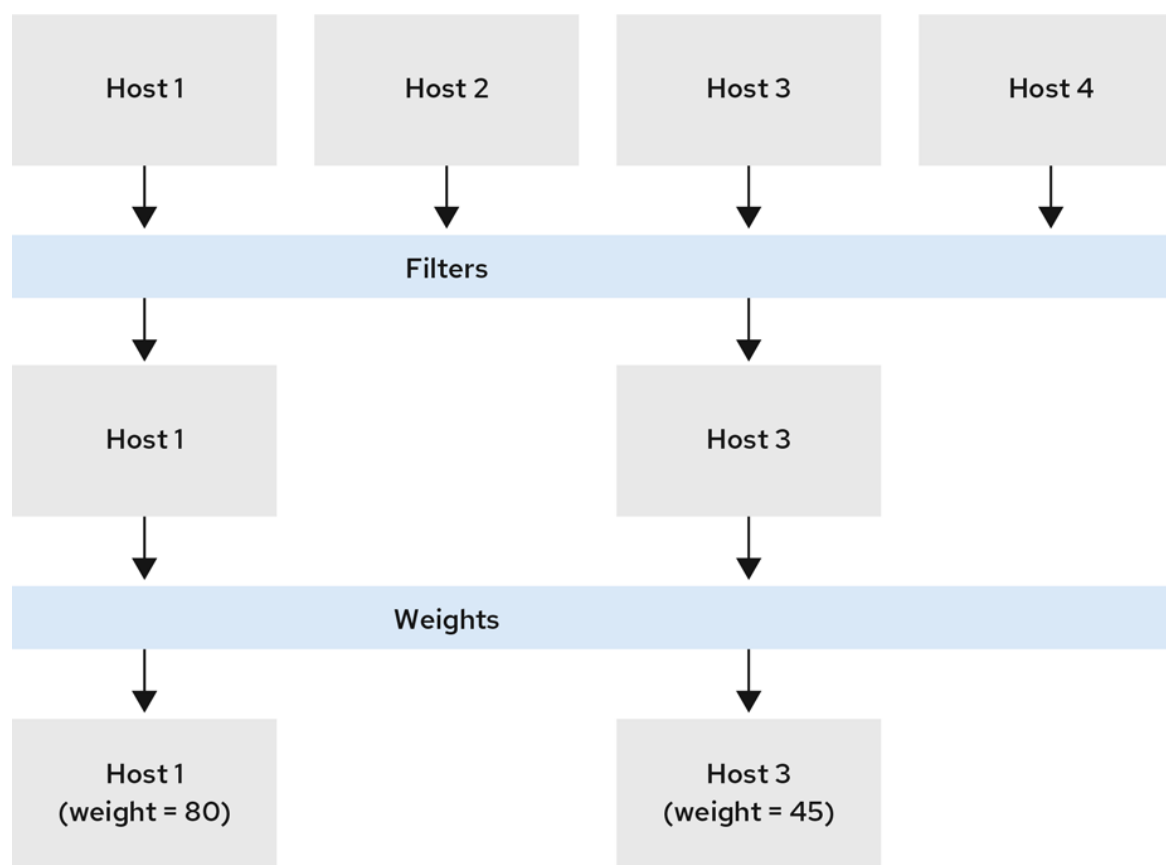
Components

ACM은 오픈시프트와 동일하게 통합된 CI/CD환경을 여러 클러스터에 제공한다. 이때 사용하는 개념은 Hub 및 Subscription기반으로 동작하게 된다. ACM는 Fleets라는 개념기반(앞서 이야기한 Placement)으로 클러스터를 관리한다.

오픈스택의 Placement와 비슷하지만 조금 다른 개념이다.

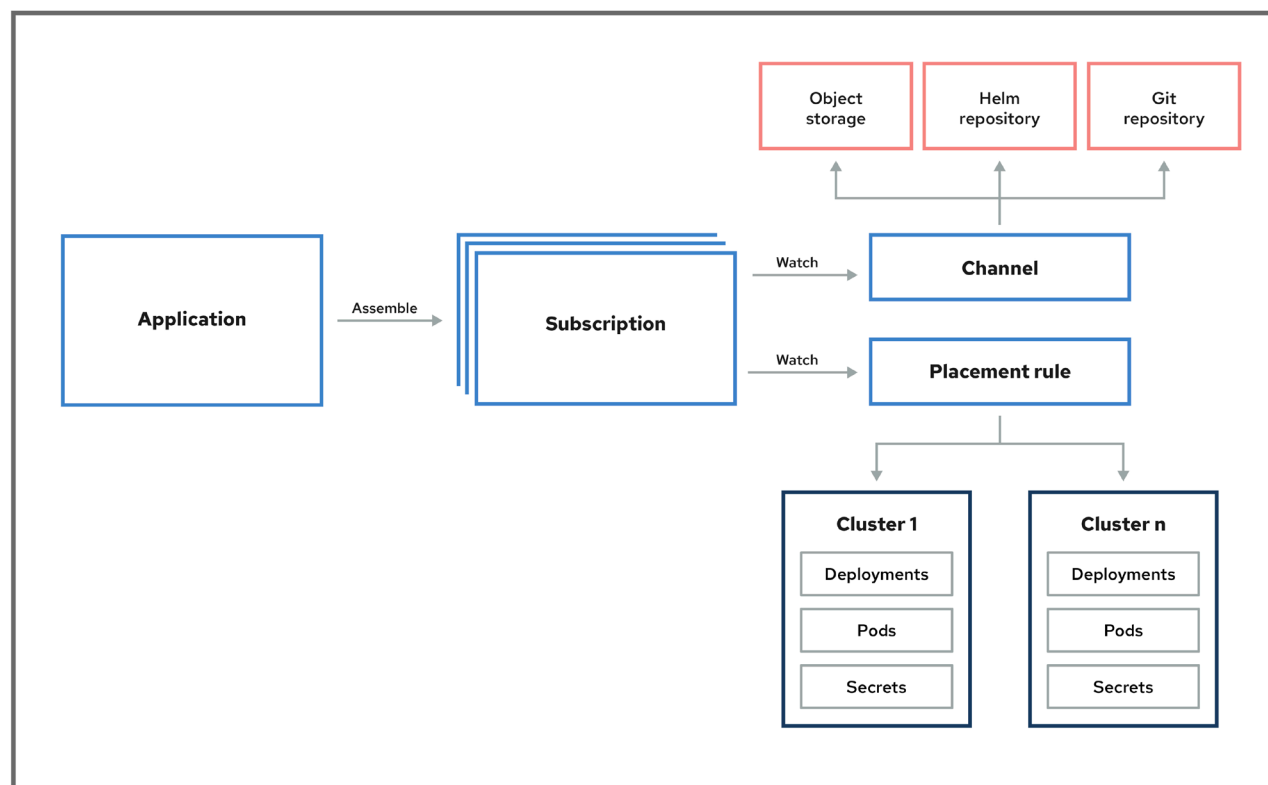
진행 전, 미리 한번 비교 해본다.

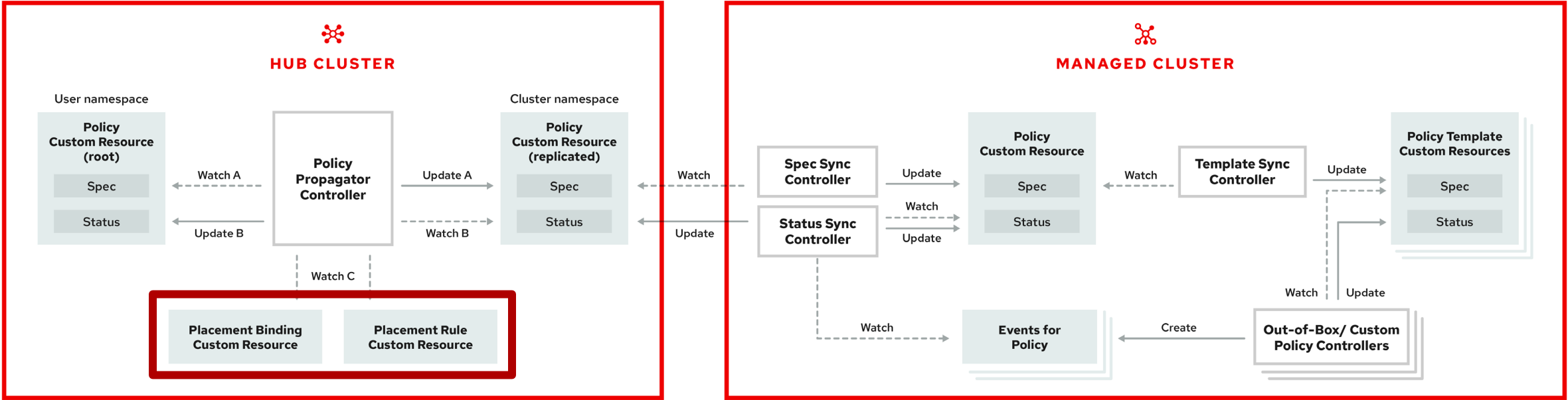
OpenStack Placement



OCM/ACM Placement

APPLICATION SUBSCRIPTION MODEL





Klusterlet

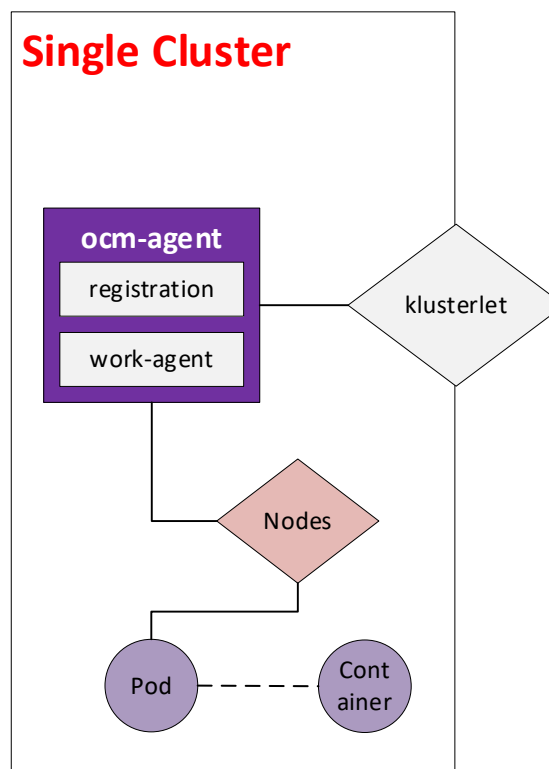
Components

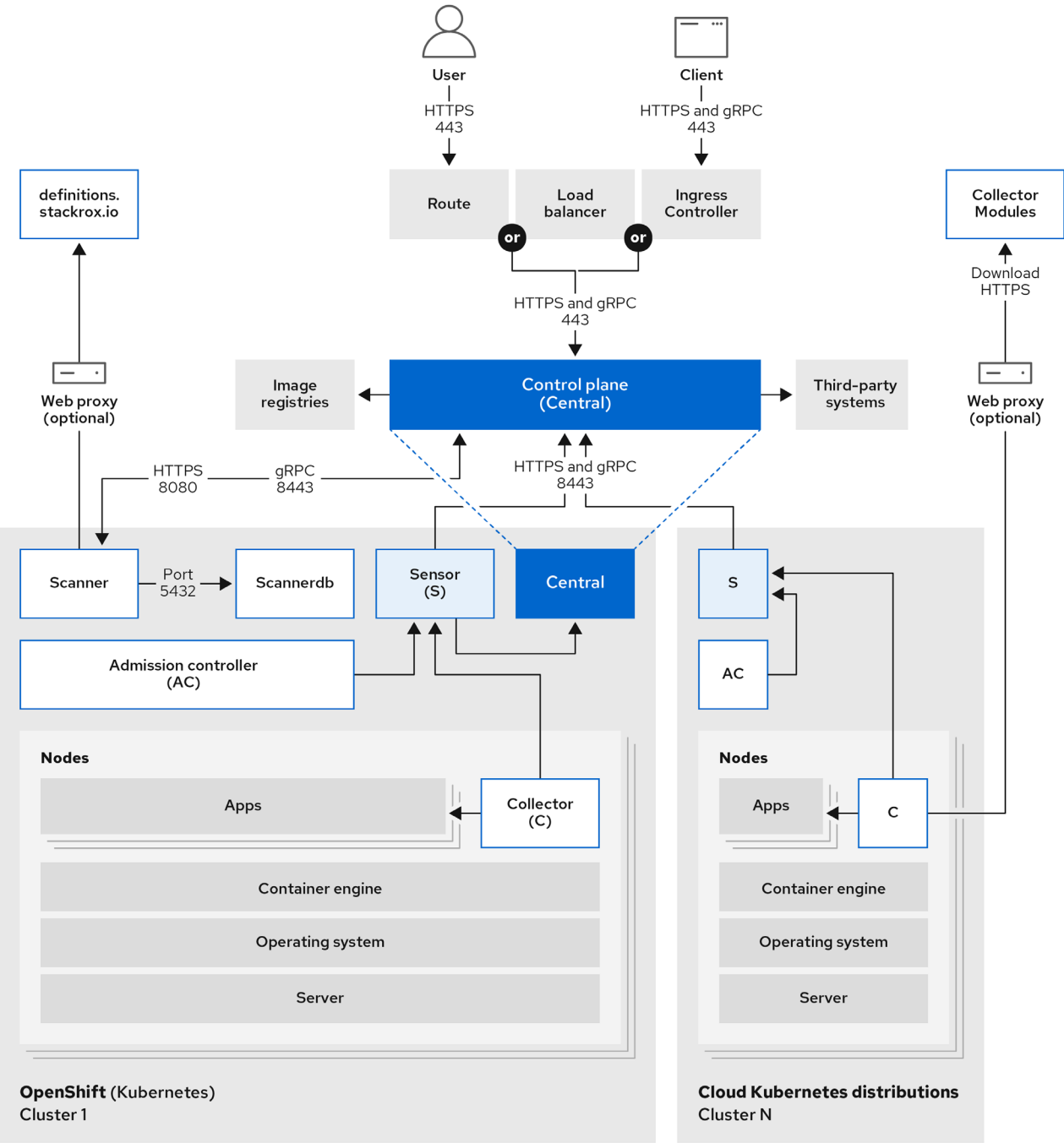
OCM 그리고 ACM에서는 Klusterlet은 매우 중요하다.

Klusterlet은 기존 쿠버네티스나 혹은 오픈 시프트에서 사용하는 kubelet하고 비슷한 기능을 한다. 다만, 차이점이라고 하면, "Node to Node"에서 "Cluster to Cluster"로 변경이 되었다.

Klusterlet

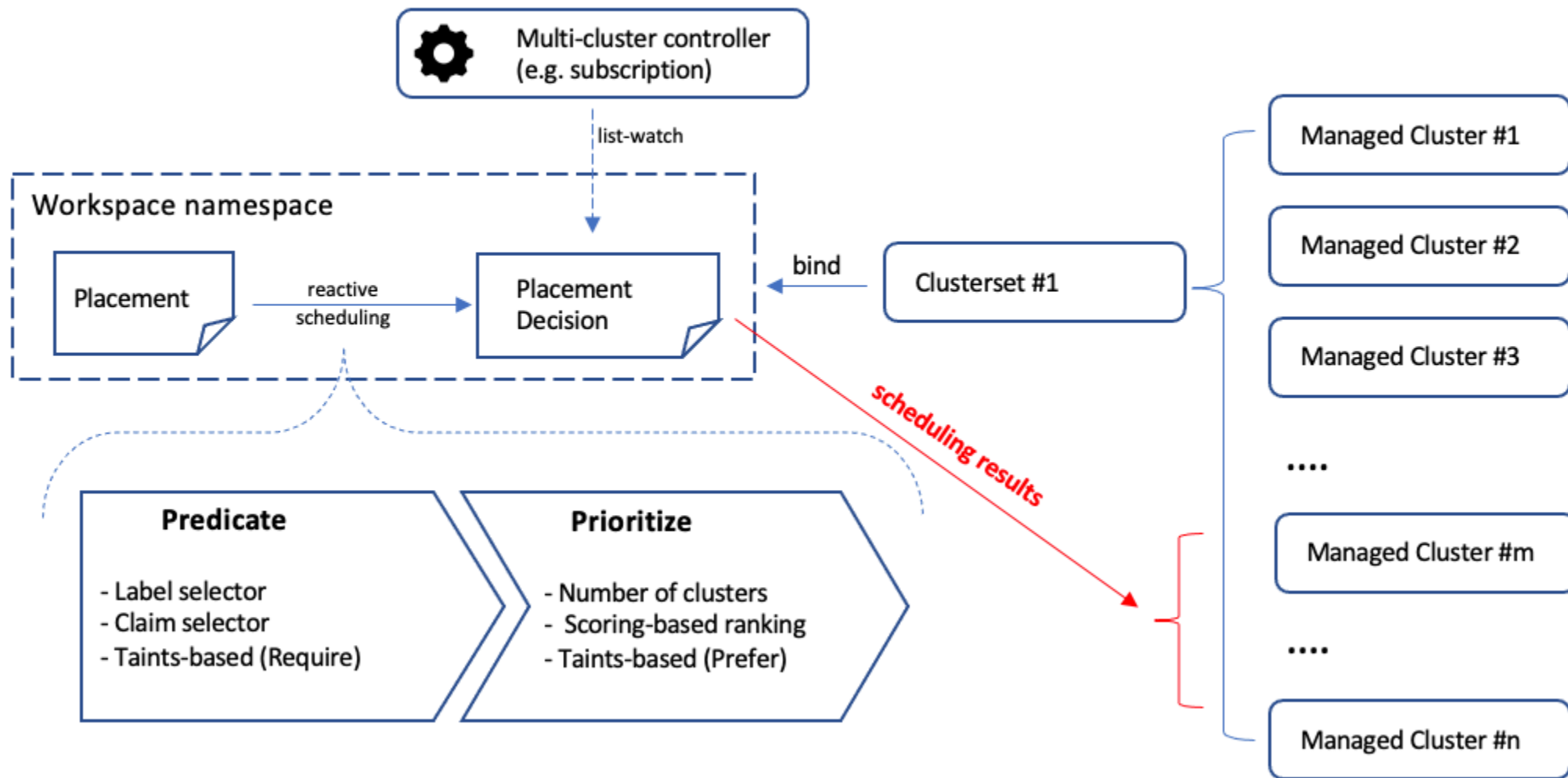
Components







Hub Cluster



Placement

Components

apiVersion: cluster.open-cluster-management.io/v1beta1

kind: **Placement**

metadata:

name: **placement1**

namespace: **default**

spec:

numberOfClusters: 3

clusterSets:

- **prod**

Placement

Components

predicates:

- **requiredClusterSelector:**

- labelSelector:

- matchLabels:

- purpose: test

- **claimSelector:**

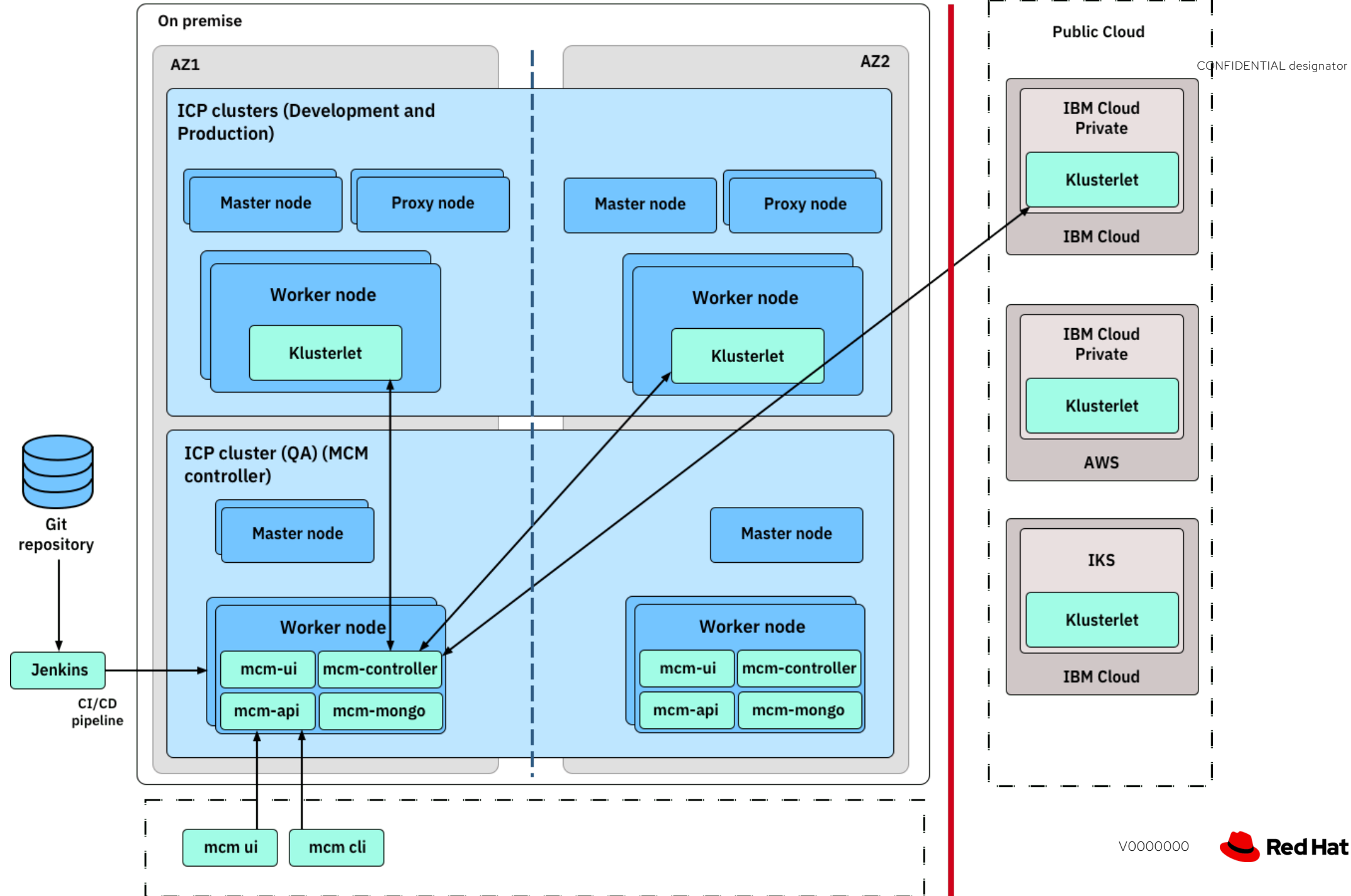
- matchExpressions:

- key: platform.open-cluster-management.io

- operator: In

- values:

- aws



DAY 2

- ▶ Role based Access Control

ROLE BASED

ROLE-BASED ACCESS CONTROL

ROLE-BASED ACCESS CONTROL

open-source Identity and Access Management (IAM)

Red Hat Identity Management (IdM)

- **Identity** (machine, user, virtual machines, groups, authentication credentials)
- **Policy** (host based access control)
- **Audit** (this component is deferred)

Keycloak

Propagator/Propaganda

Propagator

watch and update

<https://github.com/open-cluster-management-io/governance-policy-propagator>

propagation.go

update

<https://github.com/open-cluster-management-io/governance-policy-propagator/blob/main/controllers/propagator/propagation.go>

```
54  const (  
55      startDelim      = "{{hub"  
56      stopDelim       = "hub}}"  
57      TriggerUpdateAnnotation = "policy.open-cluster-management.io/trigger-update"  
58  )
```


propagation.go

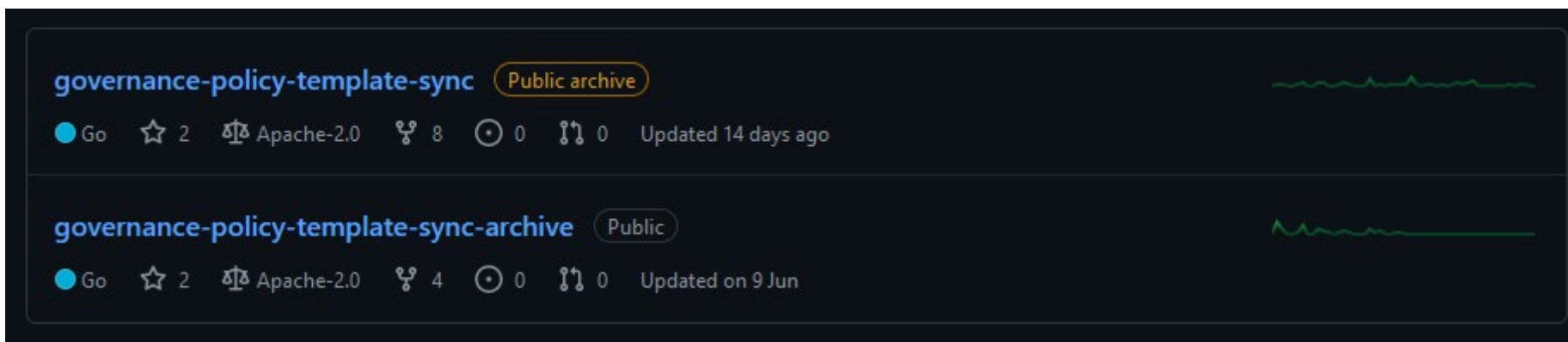
watch

<https://github.com/open-cluster-management-io/governance-policy-propagator/blob/main/controllers/propagator/propagation.go>

```
243 // decisionResult contains the result of handling a placement decision of a policy. It is intended
244 // to be sent in a channel by handleDecisionWrapper for the calling Go routine to determine if the
245 // processing was successful. Identifier is the PlacementDecision, with the ClusterNamespace and
246 // the ClusterName. TemplateRefObjs is a set of identifiers of objects accessed by hub policy
247 // templates. Err is the error associated with handling the decision. This can be nil to denote success.
248 type decisionResult struct {
249     Identifier      appsv1.PlacementDecision
250     TemplateRefObjs map[k8sdepwatches.ObjectIdentifier]bool
251     Err             error
252 }
```

template-sync

the archived



template-sync

<https://github.com/open-cluster-management-io/governance-policy-template-sync-archive>

```
38  var log = ctrl.Log.WithName(ControllerName)
39
40  //+kubebuilder:rbac:groups=policy.open-cluster-management.io,resources=*,verbs=get;list;watch;create;update;patch;delete
41  //+kubebuilder:rbac:groups=core,resources=events,verbs=get;list;watch;create;update;patch;delete
```

policy controller

NIST-800-53

<https://github.com/open-cluster-management-io/governance-policy-addon-controller>

policy controller

NIST-800-53

<https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final>

<https://csf.tools/reference/nist-sp-800-53/r5/cm/cm-2/>

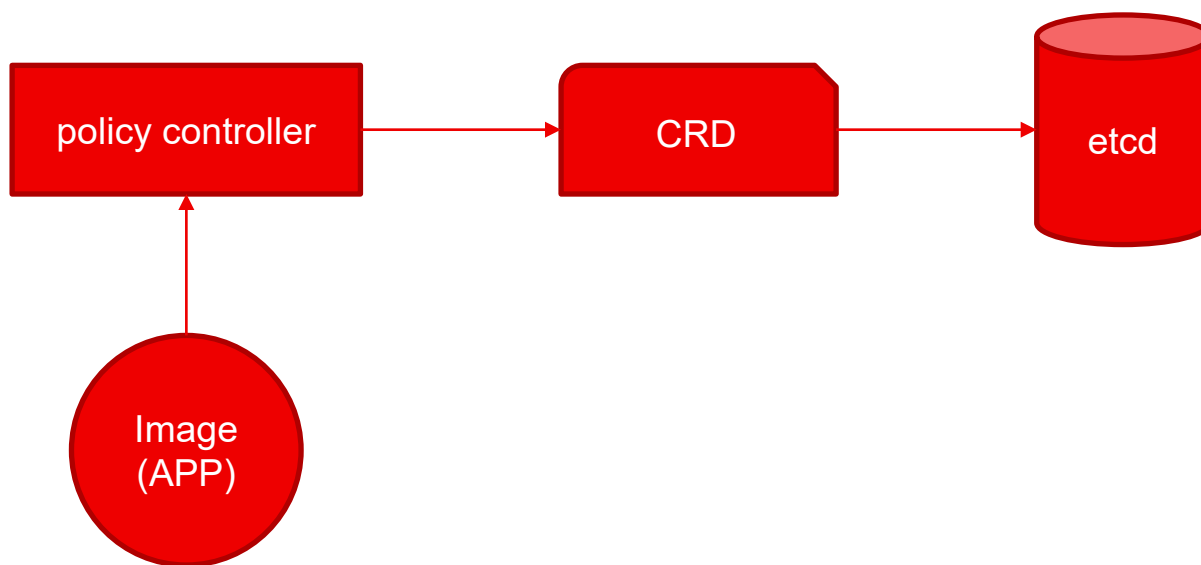
policy controller

image

```
image: quay.io/open-cluster-management/governance-policy-propagator:latest
imagePullPolicy: Always
name: governance-policy-propagator
ports:
- containerPort: 8383
  name: http
  protocol: TCP
serviceAccountName: governance-policy-propagator
```

policy controller

image



IAM Policy controller

<https://github.com/stolostron/iam-policy-controller>

IAM Policy

사용자화

<https://github.com/stolostron/governance-policy-framework>

subscription

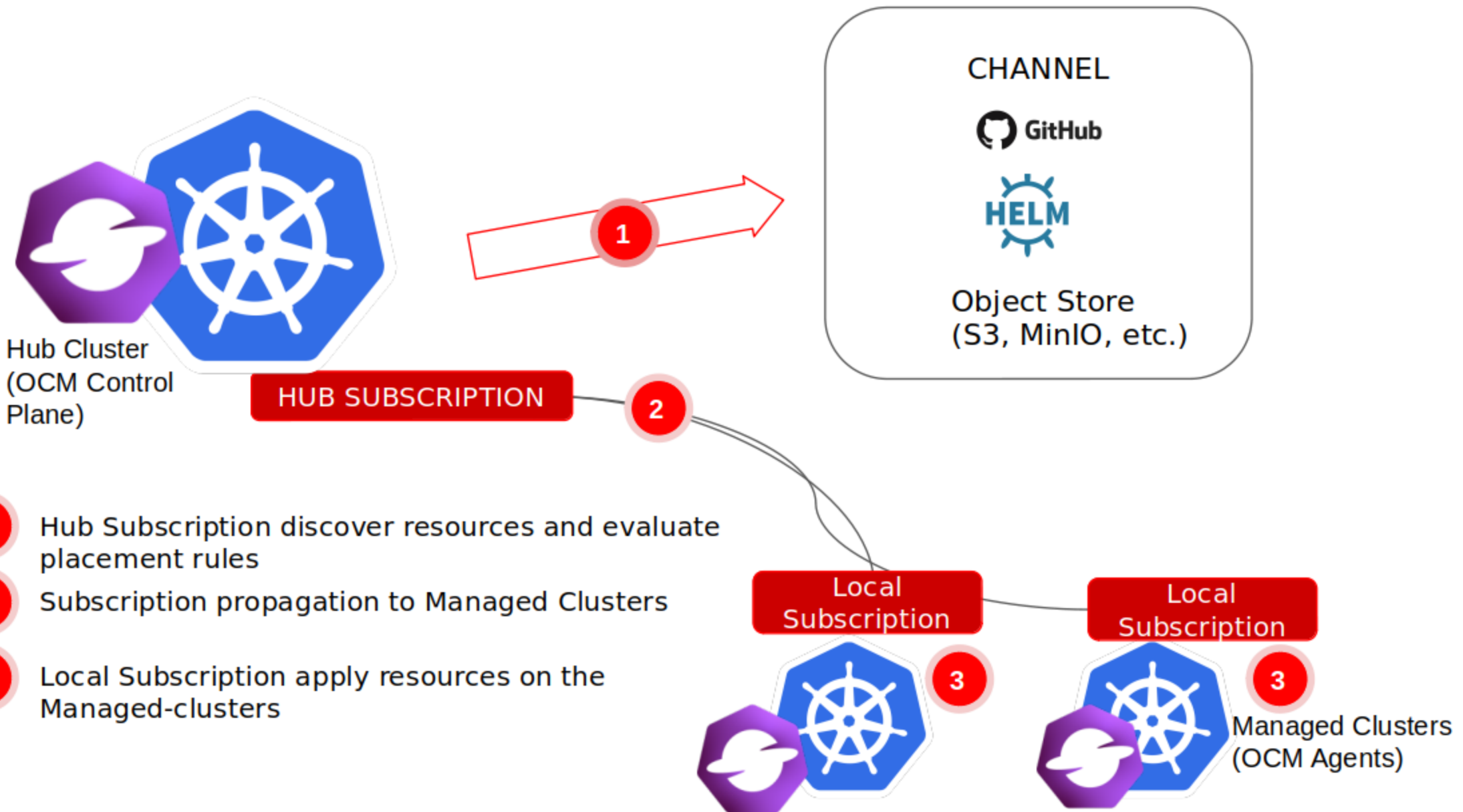
사용자화

<https://github.com/open-cluster-management-io/multicloud-operators-subscription>

subscription

사용자화

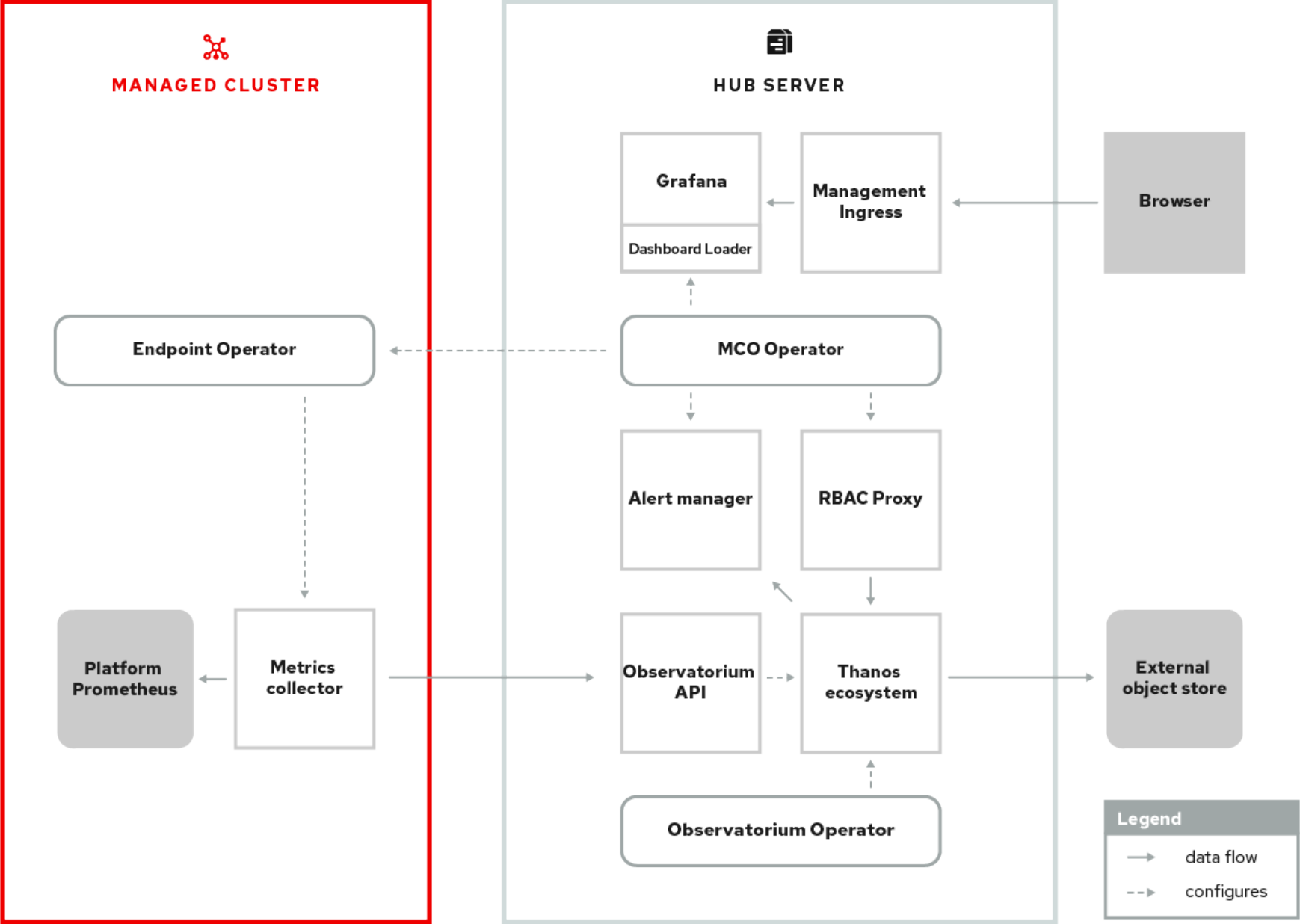
<https://github.com/open-cluster-management-io/multicloud-operators-subscription>



Observability

Observability

모니터링 및 분석



Thanos Project

사용자화



Thanos Project

모니터링

이 프로젝트의 기반은 **Prometheus**이며, 컨테이너 기반의 pv,pvc에서 오랜 기간 데이터 저장 및 분석하는 도구.

프로젝트 홈페이지는 다음과 같다.

<https://thanos.io/>

Thanos Project

조건

- Put Prometheus in the same failure domain. This means same network, same datacenter as monitoring services.
- Use persistent disk to persist data across Prometheus restarts.
- Use local compaction for longer retentions.
- Do not change min TSDB block durations.
- Do not scale out Prometheus unless necessary. Single Prometheus is highly efficient (:

Thanos Project

조건

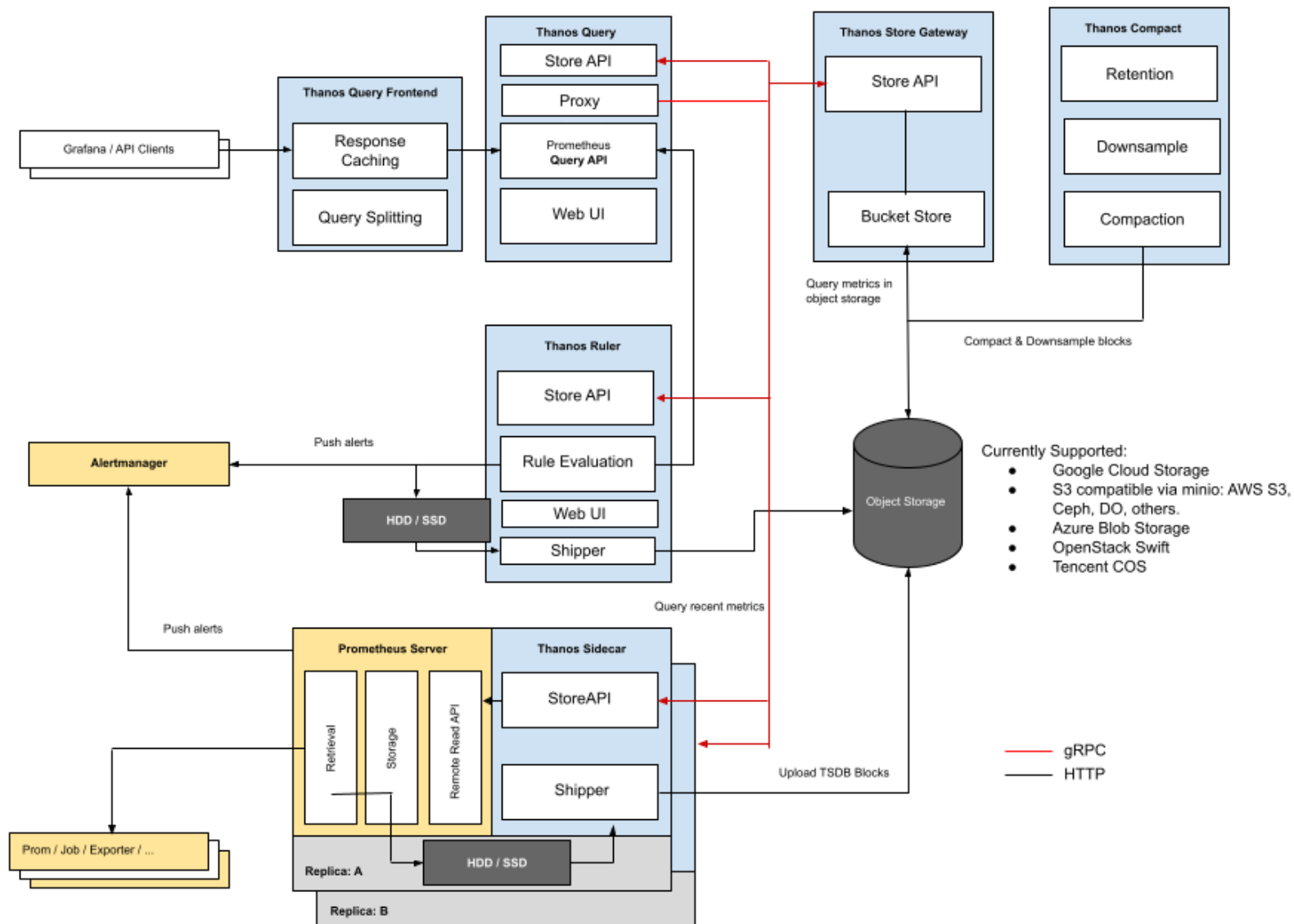
- Sidecar: connects to Prometheus, reads its data for query and/or uploads it to cloud storage.
- Store Gateway: serves metrics inside of a cloud storage bucket.
- Compactor: compacts, downsamples and applies retention on the data stored in cloud storage bucket.
- Receiver: receives data from Prometheus's remote-write WAL, exposes it and/or upload it to cloud storage.
- Ruler/Rule: evaluates recording and alerting rules against data in Thanos for exposition and/or upload.
- Querier/Query: implements Prometheus's v1 API to aggregate data from the underlying components.
- Query Frontend: implements Prometheus's v1 API proxies it to Query while caching the response and optional splitting by queries day.

Thanos Project

what is the Sidecar?

- ▶ Thanos integrates with existing Prometheus servers through a Sidecar process, which runs on the same machine or in the same pod as the Prometheus server.
- ▶ The purpose of the Sidecar is to backup Prometheus data into an Object Storage bucket, and give other Thanos components access to the Prometheus metrics via a gRPC API.

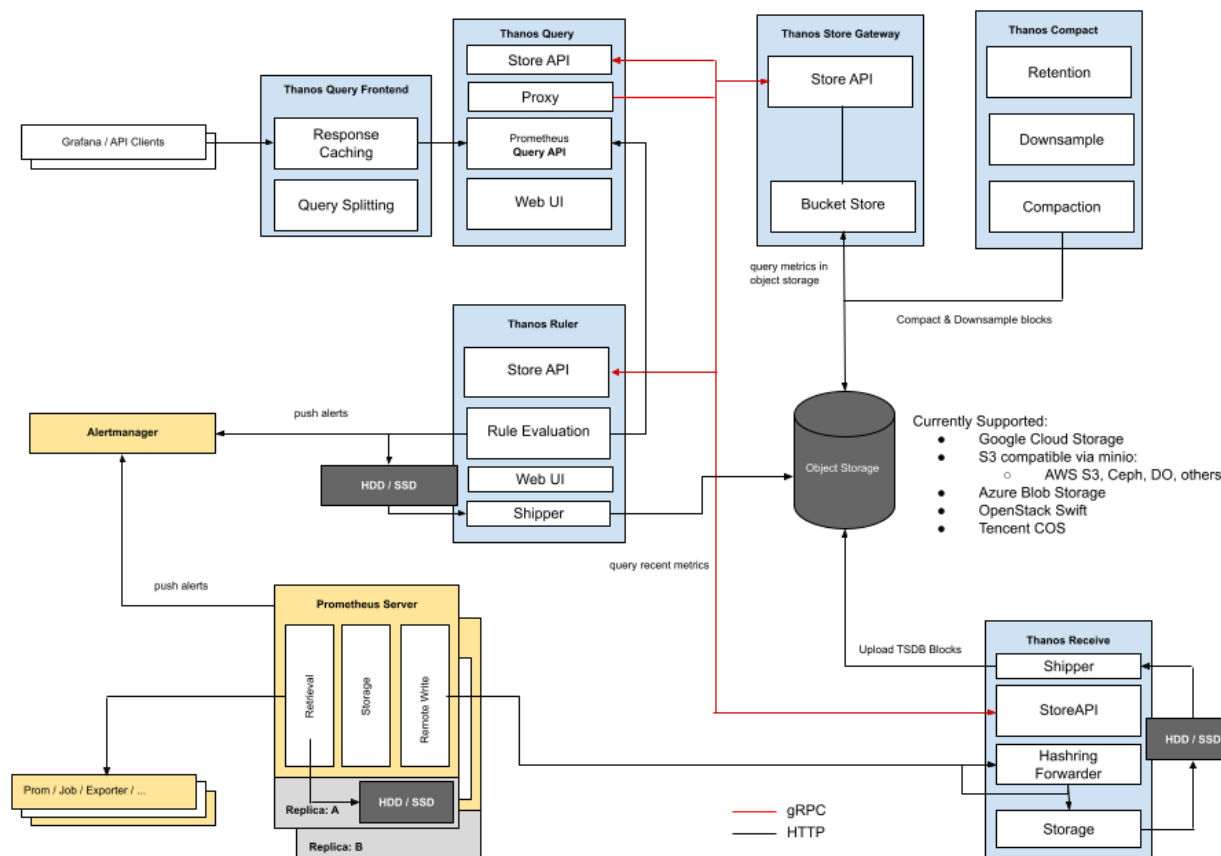
사이드카



배포

Thanos Project

사용자화



gatekeeper

사용자화

<https://github.com/open-policy-agent/gatekeeper>

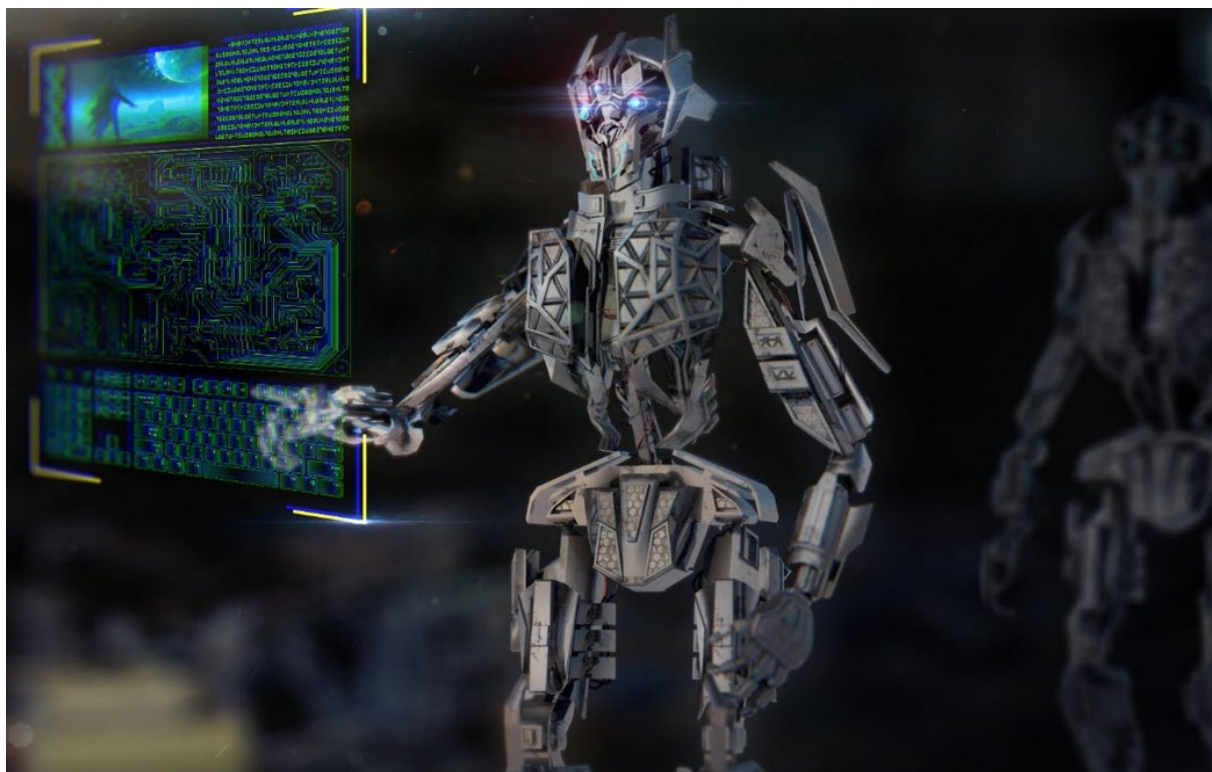
DAY 3

- ▶ CI/CD

CI/CD

CI/CD

ACM



CI/CD

ACM

- Convert any Kubernetes manifest files to RHACM configuration policies.
- Patch the input Kubernetes manifests before they are inserted into a generated RHACM policy.
- Generate additional configuration policies to be able to report on Gatekeeper and Kyverno policy violations through RHACM.
- Be run locally so that generated RHACM policies can be stored directly in Git and applied using GitOps.
- Be run by GitOps tools so that only the manifests and the policy generator configuration need to be stored in Git.

CI/CD

ACM

- The policy generator is implemented as a Kustomize generator plugin. This allows for any Kustomize configuration (e.g. patches, common labels) to be applied to the generated policies. This also means that the policy generator can be integrated into existing GitOps processes that support Kustomize, such as RHACM Application Lifecycle Management and OpenShift GitOps (i.e. ArgoCD).

CI/CD

ACM

Field	Description
spec.remediationAction	Optional. Specifies the remediation of your policy. The parameter values are enforce and inform . If specified, the spec.remediationAction value that is defined overrides the remediationAction parameter defined in the child policy, from the policy-templates section. For example, if spec.remediationAction value section is set to enforce , then the remediationAction in the policy-templates section is set to enforce during runtime. Important: Some policies might not support the enforce feature.

CI/CD

ACM

```
27     spec:
28       remediationAction: enforce # the policy-template spec.remediationAction is overridden by the preceding parameter value for spec.remediationAction.
29       severity: low
30       namespaceSelector:
31         exclude: ["kube-*"]
32         include: ["*"]
33       object-templates:
34         - complianceType: musthave
35           objectDefinition:
36             apiVersion: v1
37             kind: Namespace
38             metadata:
39               name: falco-operator
```

CI/CD

The Policy Template

적용 부분

<https://www.sans.org/information-security-policy/>

이론 부분

https://policylibrary.gatech.edu/Policy_Development_-_Policy_Template

소스코드

<https://github.com/stolostron/policy-collection/blob/main/community/README.md>

NIST Spec.

[NIST.SP.800-53r4.](#)

CI/CD

ACM

ACM현재 다음과 같은 CI/CD 도구를 지원하고 있다.

REGISTRY

- [Amazon Elastic Container Registry \(ECR\)](#)
- [Docker Hub](#)
- [Google Container Registry \(GCR\)](#)
- [Google Artifact Registry](#)
- [IBM Cloud Container Registry \(ICR\)](#)
- [JFrog Artifactory](#)
- [Microsoft Azure Container Registry \(ACR\)](#)
- [Red Hat Quay](#)
- [Red Hat container registries](#)
- [Sonatype Nexus](#)
- Any other registry that uses the [Docker Registry HTTP API](#)

CI/CD

ACM

CI(Continuous Integration)

CI도구는 일반적으로 git, svn같은 기능을 이야기한다.

CD(Continuous Delivery)

CD도구는 일반적으로 Tekton, Jenkins 그리고 ArgoCD같은 도구이다.

CI/CD

ACM

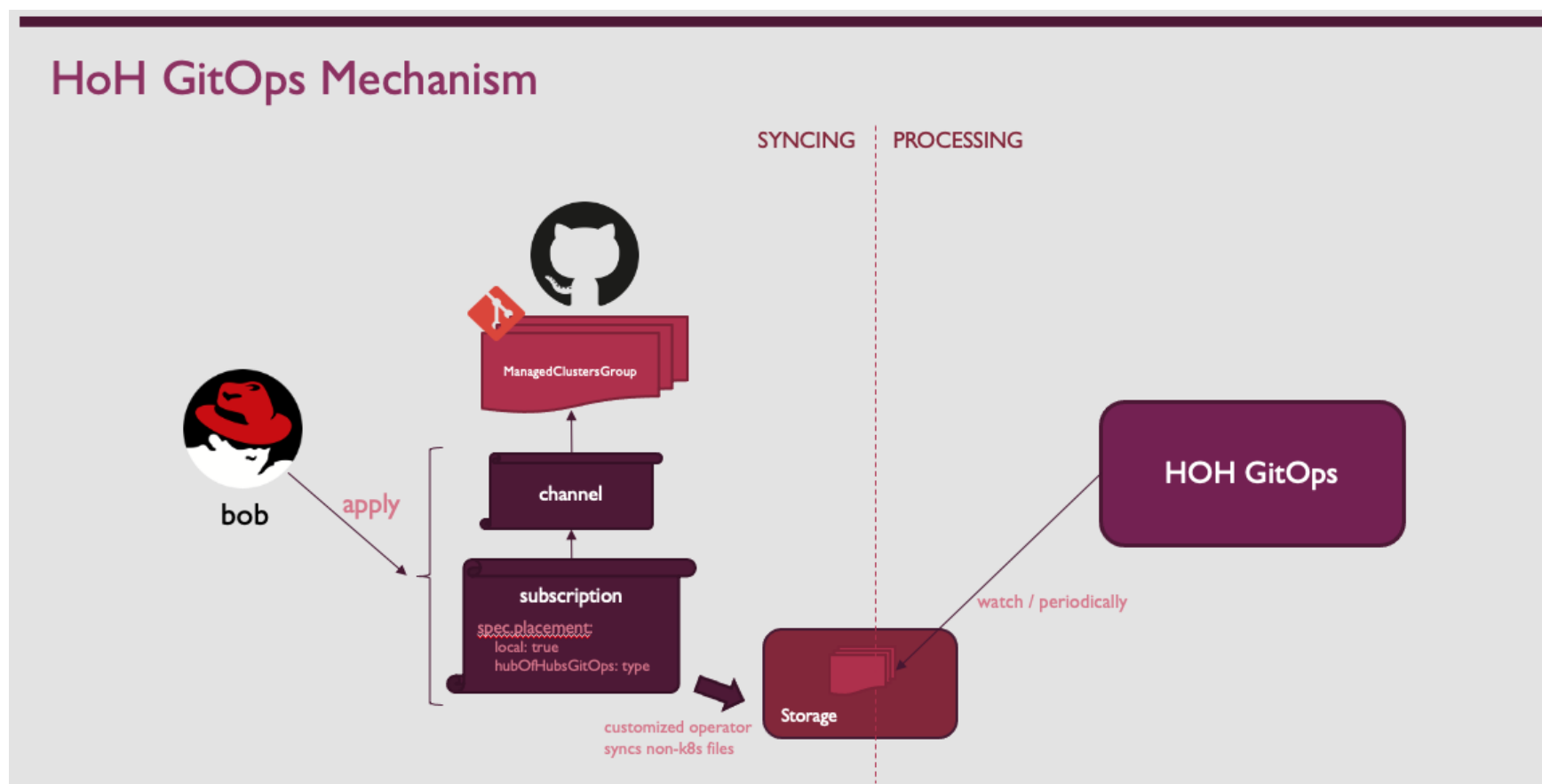
GitOps

The core idea of GitOps is having a Git repository that always contains declarative descriptions of the infrastructure currently desired in the production environment and an automated process to make the production environment match the described state in the repository. If you want to deploy a new application or update an existing one, you only need to update the repository - the automated process handles everything else. It's like having cruise control for managing your applications in production.

Git Operation, versioned CI/CD on top of declarative infrastructure. Stop scripting and start shipping.

CI/CD

ACM



CI/CD

ACM

<https://github.com/stolostron/ci-cluster-config>

<https://github.com/stolostron/hub-of-hubs-gitops>

<https://github.com/stolostron/acm-aap-aas-operations>

<https://github.com/stolostron/multicluster-observability-operator>

Quay

저장소

Quay는 다음과 같은 기능을 가지고 있다.

clair Vulnerability Static Analysis for Containers ● Go ☆ 9.1k 🍷 1.1k Public	quay Build, Store, and Distribute your Applications and Containers ● Python ☆ 2.1k 🍷 221 Public
jwtproxy An HTTP-Proxy that adds AuthN through JWTs ● Go ☆ 394 🍷 43 Public	quay-operator Operator for deploying and managing Quay ● Go ☆ 106 🍷 70 Public
claircore foundation modules for scanning container packages and reporting vulnerabilities ● Go ☆ 98 🍷 56 Public	quayctl quayctl is a command-line client for Quay ● Go ☆ 90 🍷 17 Public

Quay

저장소

- High availability
- Geo-replication
- Repository mirroring
- Docker v2, schema 2 (multiarch) support
- Continuous integration
- Security scanning with Clair
- Custom log rotation
- Zero downtime garbage collection
- 24/7 support

Quay

저장소

- ▶ Project Quay provides support for:
 - Multiple authentication and access methods
 - Multiple storage backends
 - Custom certificates for Quay, Clair, and storage backends
 - Application registries
 - Different container image types

Quay

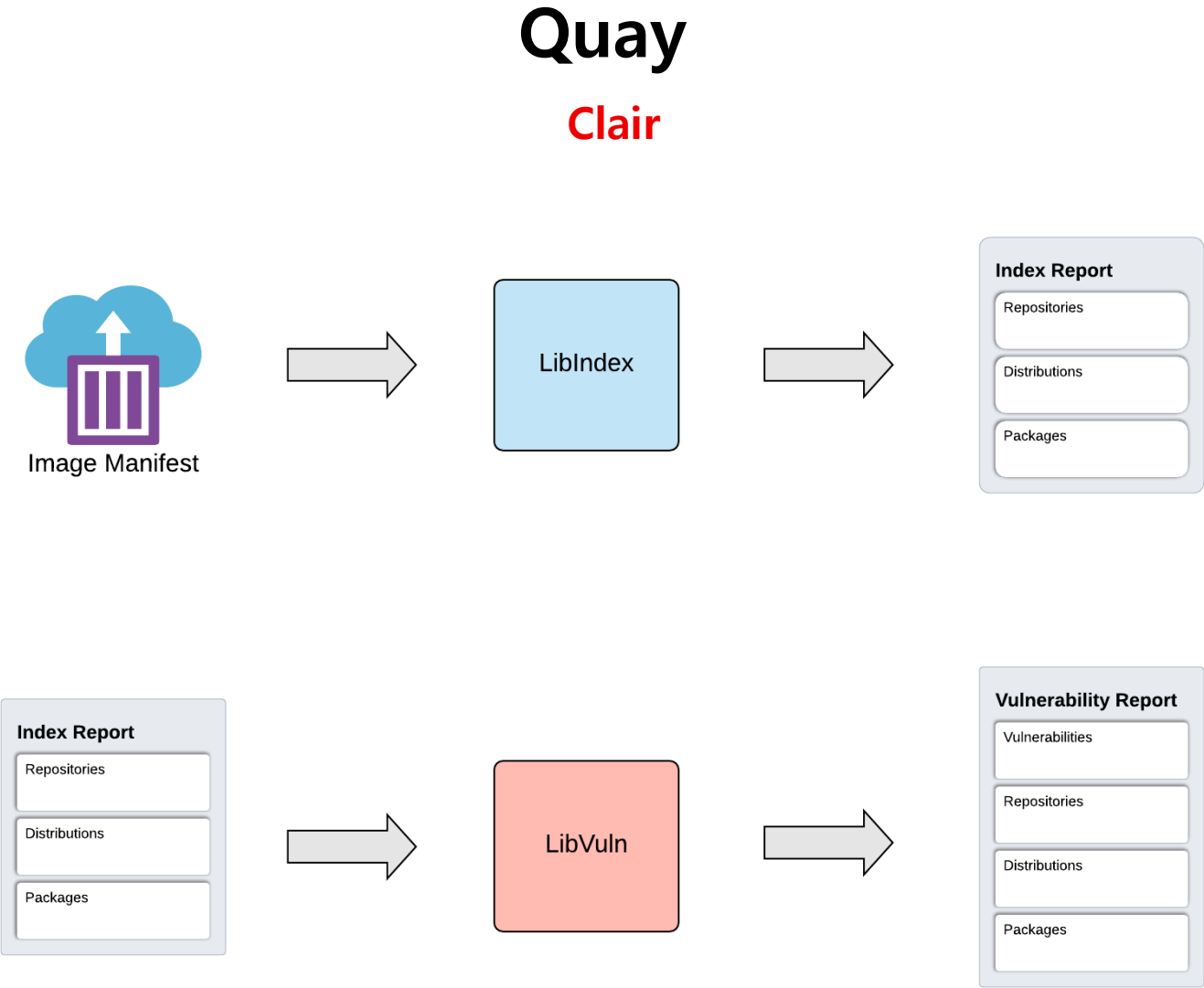
저장소

본래 호스티드 형태로 제작이 되었다가, 오픈 시프트가 릴리즈 되면서 컨테이너드 형태로 제작이 되었음.

오픈소스 버전과 상용 버전에 많은 차이가 있으며, Quay기존의 git기능에 다중 접근 및 저장이 가능하도록 프론트 및 백엔드가 작성이 되어있다. 또한, H/A , Scape를 통해서 보안이 강화가 되었다.

https://quay.github.io/claircore/what_is.html

https://docs.projectquay.io/deploy_quay.html



Kustomize ACM

YAML

<https://kustomize.io/>

Kustomize introduces a template-free way to customize application configuration that simplifies the use of off-the-shelf applications. Now, built into kubectl as `apply -k`.

<https://cloud.redhat.com/blog/generating-governance-policies-using-kustomize-and-gitops>