Application을 넘어 Infrastructure와 Kubernetes도 GitOps로 관리하기

SK텔레콤 엄주관

Cluster API 기술을 활용하여 GitOps로 Kubernetes 클러스터를 구축하고 운영하는 방법을 소개합니다.

Kubernetes 클러스터를 구성하기 위한 방법

		What	How	
1	Infrastructure Provisioning	노드 (서버 or VM) 네트워크 스토리지 OS	사람 IaC (Terraform, …)	Managed Kubernetes Service - EKS, AKS, GKE, NKS, … with IaC (Terraform, …)
2	Kubernetes Cluster Bootstrapping	Kubernetes 설치	kubeadm Kubespray	
3	Addons	CNI CSI Ingress 모니터링 어플리케이션 CICD …	Kubespray Helm Kustomize …	

실제 사용했던 방법은 다음과 같았습니다

Kubespray를 중심으로 Ansible 플레이북을 추가하고 통합 https://github.com/openinfradev/tacoplay

- 필요한 소프트웨어들을 한 번에 설치, 제거, 확장할 수 있는 통합 playbook 제공
- 각 소프트웨어의 정상 동작에 필요한 OS 설정 자동화
- 각 소프트웨어들 간의 통합 연동 설정
- 인터넷 접근이 불가능한 환경에서의 구축을 위한 준비/설치 간소화

장점이라면,

➔ Ansible inventory와 변수 파일을 통해 각 사이트/클러스터 별 설정 관리
➔ Git을 이용하여 변경 사항에 대한 비교 및 이력 관리

Challenges

“일단 급하니까 수동으로 해놓자“ (작업 내역 관리는 없이)
“Git에 있는 코드 적용을 언제 했지? 했던가?"
(결국 다음 번 설정 적용 시 충돌이 발생, 잘 되던 게 안되기 시작하니) "그냥 전부 수동으로 설정하는 게 낫지 않아요?“

On-Presmise 물리 서버 환경, VM 환경에서는 “그럭저럭” 잘 동작

그러나...

- ❏ 플레이북 실행 시점에만 적용되기 때문에 운영 편의 등의 이유로 수동으로 환경 수정한 부분과 이후 재 반영 시 충돌이 발생
- ❏ 현재 Git 저장소 내용과 실제 환경에 적용된 내역과의 비교/추적 및 조치의 어려움

퍼블릭 클라우드로 확장한다면?

- ❏ 기존 방법을 재사용하는 것도 가능하지만 비효율적
- ❏ 퍼블릭 클라우드를 위한 방법을 추가 도입 → 환경 별로 서로 다른 방법을 유지하는 부담 증가

잠깐, 어플리케이션 배포 방법은 어떻게 개선했을까요?

Decapod (https://openinfradev.github.io/decapod-docs) 을 활용하여 다양한 어플리케이션, 서비스들을 GitOps 형태로 배포/관리

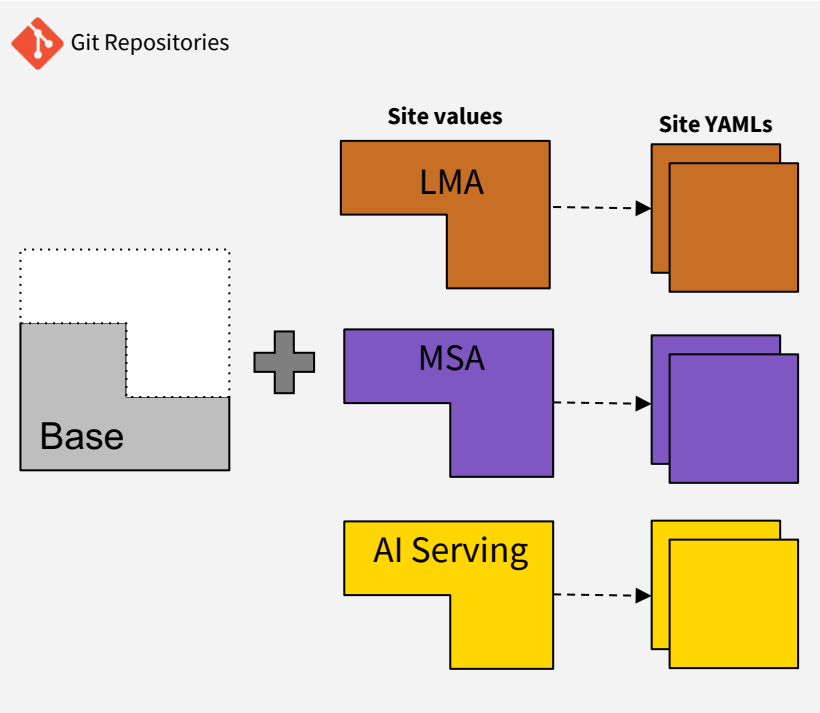
- LMA(Logging, Monitoring and Alarm), Service Mesh 등의 pre-defined App Group 제공
- 사용자의 Kubernetes 환경에 따른 손쉬운 Yaml Customization 제공
- Application간 Dependency를 고려한 순차적/단계적인 Deployment 지원 via Argo Workflow
- Git을 통한 버전 관리 및 자동 반영 via Argo CD

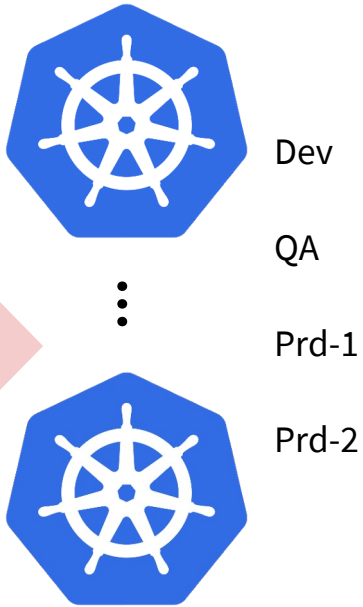
GitOps 장점

➔ Git 내용과 실제 환경의 내용이 동일

➔ Pull/Merge Request 기반의 리뷰/반영 절차

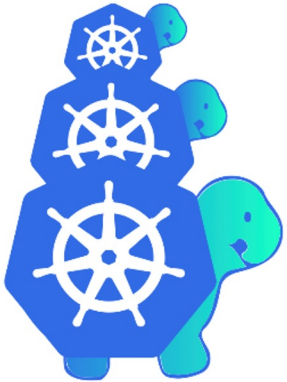
Decapod https://openinfradev.github.io/decapod-docs

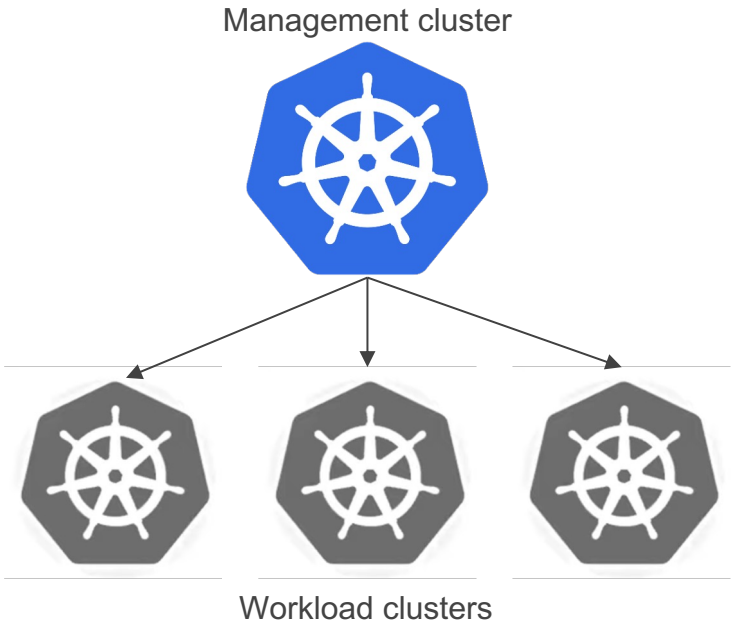




Cluster API

Cluster API는 Kubernetes 클러스터를 프로비저닝, 업그레이드 및 운영하기 위한 **선언적 API** 및 도구를 제공하는 Kubernetes 하위 프로젝트입니다.





Goal

Decapod

Cluster API Concepts

Cluster

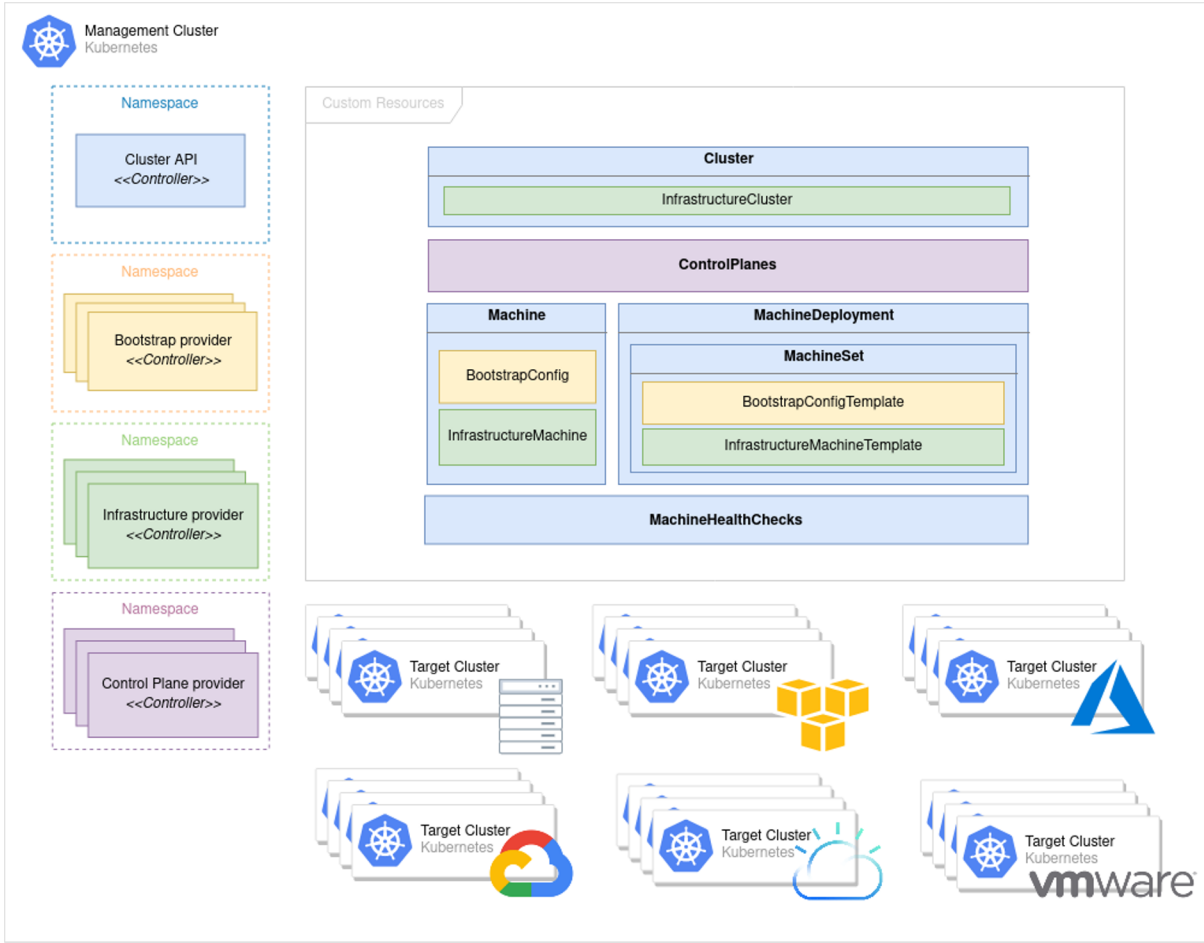
- Management cluster
- Workload cluster

Provider

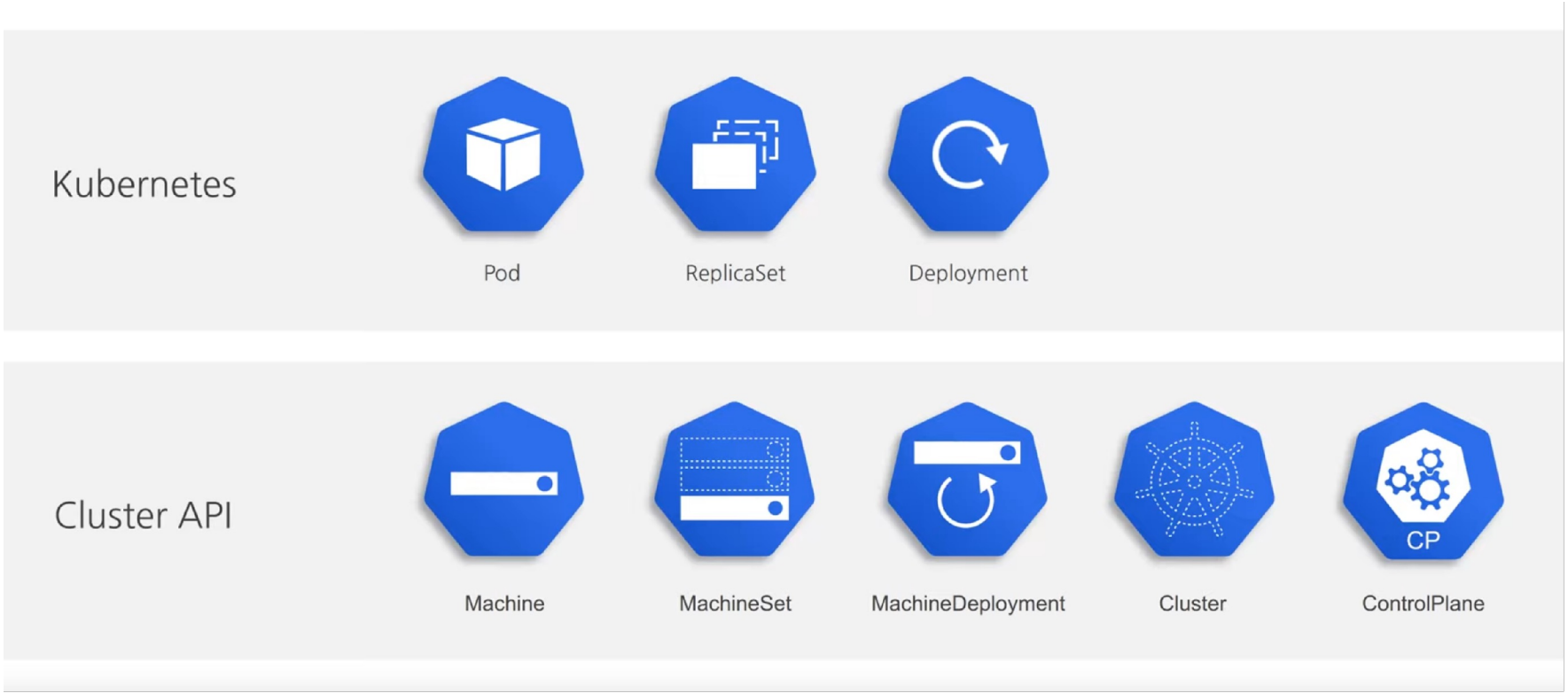
- Infrastructure Provider
- Bootstrap Provider

Control Plane

- Self-provisioned
- Mananged: EKS, GKE, AKS, …



Cluster API CRDs



https://youtu.be/A2BBuKx1Yhk?t=1447

Multi Infra Providers

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: aws-cluster
spec:
  infrastructureRef:
    apiVersion:
infrastructure.cluster.x-k8s.io/v1beta1
    kind: AWSCluster
    kind: GCPCluster
    kind: AzureCluster
    name: infra-cluster
  controlPlaneRef:
…
```

```
apiVersion: infrastructure.cluster.x-
k8s.io/v1beta1
kind: AWSCluster
metadata:
  name: infra-cluster
spec:
…
```

```
apiVersion: infrastructure.cluster.x-
k8s.io/v1beta1
kind: GCPCluster
metadata:
  name: infra-cluster
spec:…
```

```
apiVersion: infrastructure.cluster.x-
k8s.io/v1beta1
kind: AzureCluster
metadata:
  name: infra-cluster
spec:
…
```

Cluster API Provider AWS

Control Plane

Self-provisioned

AWSCluster ⇒ VPC, Subnet, ELF for K8S API, Bastion Node, …

KubeadControlPlane → AWSMachineTemplate → **Machine** → **AWSMachine** ⇒ EC2 Instance

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: aws-cluster-self
spec:
  controlPlaneRef:
    apiVersion: controlplane.cluster.x-
k8s.io/v1beta1
    kind: KubeadmControlPlane
    name: aws-cluster-self-control-plane
  infrastructureRef:
    apiVersion: infrastructure.cluster.x-
k8s.io/v1beta1
    kind: AWSCluster
    name: aws-cluster-self
```

Mananged

AWSManagedControlPlane ⇒ EKS Cluster

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: aws-cluster-eks
spec:
  controlPlaneRef:
    apiVersion: controlplane.cluster.x-
k8s.io/v1beta1
    kind: AWSManagedControlPlane
    name: aws-cluster-eks-control-plane
  infrastructureRef:
    apiVersion: controlplane.cluster.x-
k8s.io/v1beta1
    kind: AWSManagedControlPlane
    name: aws-cluster-eks-control-plane
```

Worker Nodes

MachineDeployment

KubeadmConfigTemplate + AWSMachineTemplate → **Machine** → **AWSMachine** ⇒ EC2 Instance

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachineDeployment
metadata:
  name: aws-md-0
spec:
  clusterName: aws-cluster
  replicas: 3
  selector:
    matchLabels: null
  template:
    spec:
      bootstrap:
        configRef:
          apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
          kind: KubeadmConfigTemplate
          name: aws-md-0
      clusterName: eom-test-default
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
        kind: AWSMachineTemplate
        name: aws-md-0
      version: v1.25.0
```

MachinePool

AWSMachinePool + KubeadmConfig ⇒ EC2 Auto Scaling group

AWSManagedMachinePool ⇒ EKS Node group

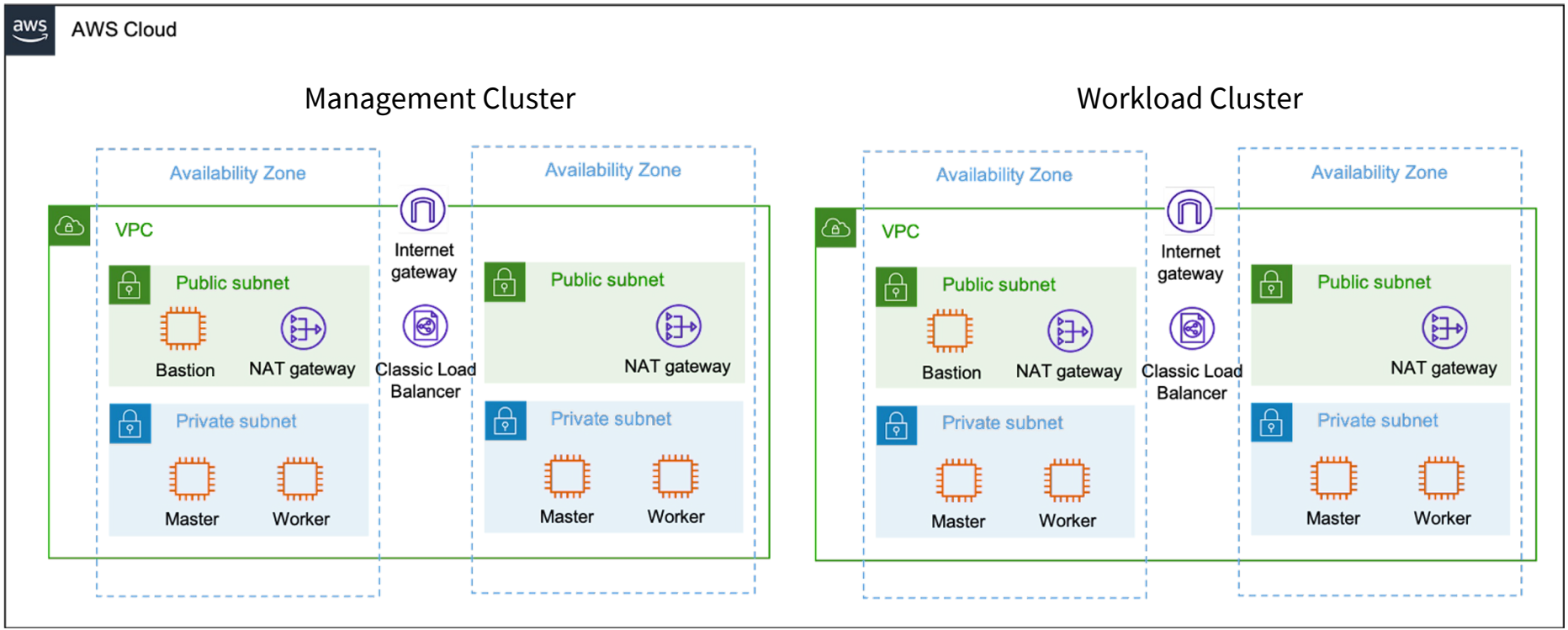
```
apiVersion: cluster.x-k8s.io/v1beta1
kind: MachinePool
metadata:
  name: aws-mp-0
  namespace: default
spec:
  clusterName: aws-cluster
  replicas: 3
  template:
    spec:
      bootstrap:
        configRef:
          apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
          kind: KubeadmConfig
          name: aws-mp-0
      clusterName: ${CLUSTER_NAME}
      infrastructureRef:
        apiVersion: infrastructure.cluster.x-k8s.io/v1beta2
        kind: AWSMachinePool / AWSManagedMachinePool
        name: aws-mp-0
      version: v1.25.0
```

Worker Nodes Multi AZ

Self-Provisioned 쿠버네티스 클러스터 배포용 자원

	Cluster API	Cluster API Provider AWS
Cluster	**Cluster** ● Cluster Network ● Service Domain	**AWSCluster** ● region ● ssh key name ● AWS 자원: VPC, Subnets, Bastion Node, …
Control Plane Node	**KubeadmControlPlane** ● replicas ● Kubeadm config ● Kubernetes version	**AWSMachineTemplate** ● EC2 instance type ● ssh key name ● root volume size spot instance 지원
Worker Node	**MachineDeployment** ● replicas ● kubernetes version **KubeadmConfigTemplate** ● kubeadm config	**AWSMachineTemplate** ● EC2 instance type ● ssh key name ● root volume size ● Subnet spot instance 지원

On AWS : Cluster per VPC



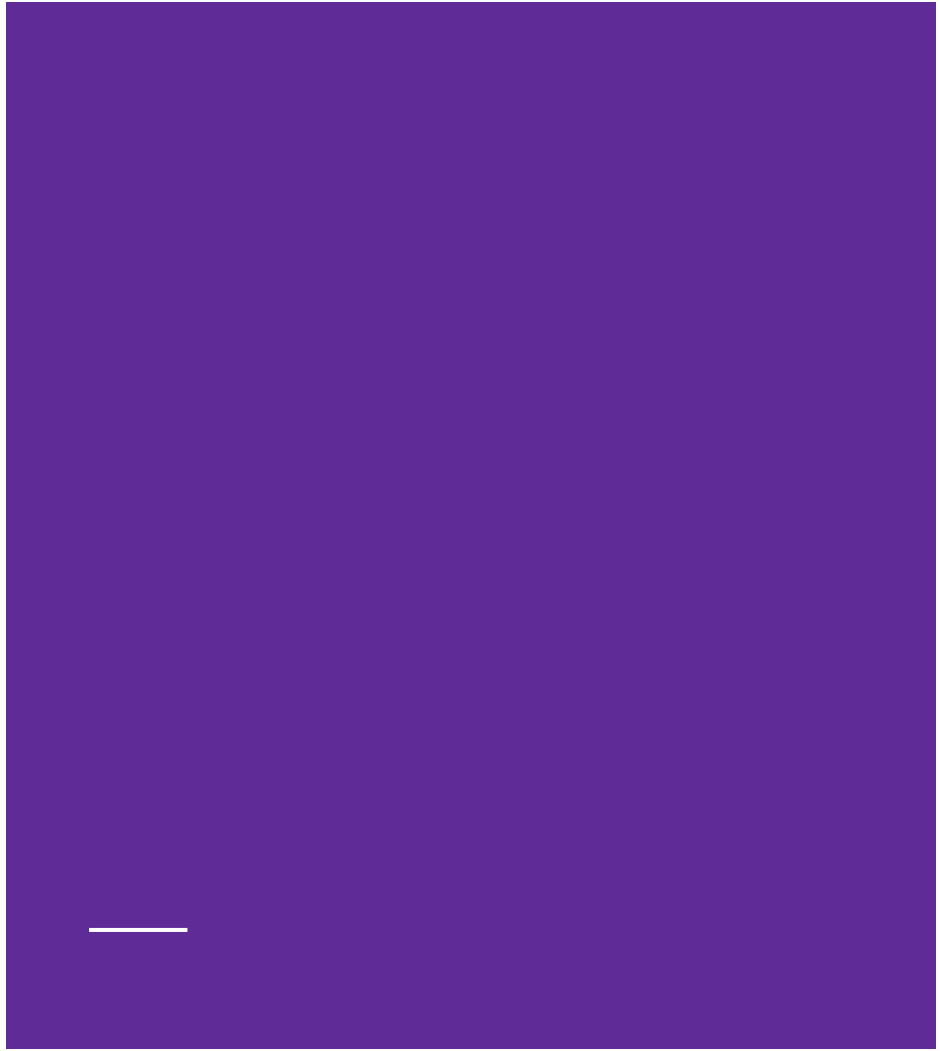
Cluster API Provider AWS

Spot 인스턴스 지원

- 저렴한 인스턴스 사용의 요구사항
- Batch process 혹은 실시간을 요하지 않는 workload에 적합
- MachineDeployment를 사용하고 최대 지불비용 지정 (MachinePool은 미지원)
- Bastion 노드에도 적용가능

Cluster API Provider BYOH

BYOH (Bring Your Own Host)



Cluster API Provider BYOH

https://github.com/vmware-tanzu/cluster-api-provider-bringyourownhost/

기존 인프라 프로바이더와는 달리 인프라 프로비저닝 같은 역할을 하지 않고 이미 리눅스가 실행되고 있는 호스트를 대상 (Bring Your Own Host)

왜?

- On-Premise 환경에서도 동일하게 Cluster API를 활용한 일관된 방법을 사용할 수 있다
- 인프라 프로바이더가 지원되지 않는 혹은 여러 "어른들의 사정"으로 사용할 수 없는 프라이빗, 퍼블릭 클라우드에서도 동일하게 사용할 수 있다

각 호스트에 에이전트를 실행하여 가용한 노드를 등록하고 클러스터가 배포될 때 관련 도구 설치 작업 및 부트스트래핑 등의 역할을 수행하도록 함

Helm Charts 구현 결과물

Cluster API Provider AWS

- https://github.com/openinfradev/helm-charts/tree/main/cluster-api-aws

Cluster API Provider BYOH

- https://github.com/openinfradev/helm-charts/tree/main/cluster-api-byoh

Cluster API 효과와 한계점

모든 환경에 대해 인프라 프로바이더가 존재하지도 않고 일부 프로바이더는 상용 수준이 아님

상위 수준에서 일관된 API로 추상화하였지만 각 인프라 프로바이더 자원의 내역 및 구현, 동작에 대해서는 이해 및 검증이 필요

Kubernetes 클러스터 구성뿐만 아니라 가상 머신, 네트워크,로드 밸런서 및 VPC와 같은 기반 인프라 구성 모두 Kubernetes에서 어플리케이션을 배포하고 관리하는 방식으로 정의하며 베어메탈, 프라이빗/퍼블릭 클라우드 등 다양한 인프라 환경에서 일관된 방식으로 클러스터 배포가 가능

GitOps와 결합하여 전체시스템의 버전 및 상태 관리를 일원화할 수 있고 이력을 확인하거나 롤백 등의 작업도 매우 수월하게 수행

풀 리퀘스트 및 리뷰를 통해 최종 변경 사항이 반영되기 때문에 운영상의 오류를 검토하고 바로 잡을 수 있는 기회도 제공

다루지 않은 주제들

Upgrading

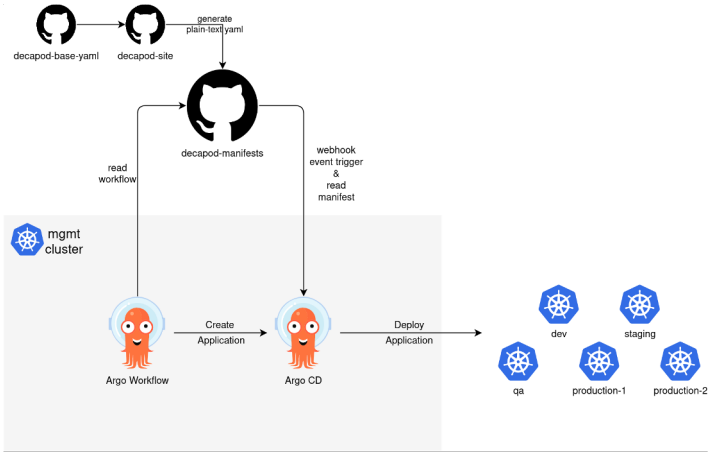
Cluster scaling / autoscaling

ClusterResourceSet

ClusterClass

…

Kubernetes Anywhere, Everything on Kubernetes



Decapod를 기반으로 Cloud Infra Resource, Kubernetes Cluster, 다양한 서비스들을 GitOps 형태로 배포하고 관리

감사합니다