Argo CD 운영

민감한 데이터 암호화

- ❖ Sealed Secrets
  - ✓ 개요
    - Sealed Secrets은 Kubernetes 시크릿을 SealdSecret 커스텀 리소스로 암호화하는 Bitnami의 오픈소스 프로젝트로 이 리소스는 Git에 저장해도 안전한 암호화된 오브젝트
    - Sealed Secrets은 공개 키 암호화(public-key cryptography) 기법을 사용하며 두 가지 주요 구성 요소로 이루어져 있음
      - ◆ 시크릿을 복호화 및 암호화하는 사설 키와 공개 키를 알 뿐 아니라 그 재조정 과정을 총괄하는 Kubernetes 컨트롤러가 있는데 이 컨트롤러는 시크릿의 재암호화를 강제하기 위한 개인 키 자동 로테이션(rotation)과 키 만료(expiration) 관리도 담당
      - ◆ 개발자가 Git 저장소에 커밋하기 전에 시크릿을 암호화하는데 사용하는 CLI kubeseal
    - SealedSecrets 오브젝트는 오직 대상 Kubernetes 클러스터에서 실행 중인 SealedSecret 컨트롤러만 암호화하고 복호화할 수 있는데 해당 컴포넌트만 독점적으로 수행하는 작업이므로 다른 누구도 해당 오브젝트를 해독할 수 없음
    - 개발자는 kubeseal CLI를 사용하여 일반 Kubernetes 시크릿 리소스를 SealedSecret 리소스로 변환할 수 있음

민감한 데이터 암호화

- Sealed Secrets
  - ✓ 설치: https://github.com/bitnami-labs/sealed-secrets/releases/
    - Windows: choco install kubeseal
    - Mac OS: brew install kubeseal
    - Ubuntu
      - # 최신 버전 확인 및 다운로드

```
KUBESEAL_VERSION=$(curl -s https://api.github.com/repos/bitnami-labs/sealed-secrets/releases/latest | grep tag_name | cut -d '"' -f 4 | sed 's/v//')

curl -OL "https://github.com/bitnami-labs/sealed-secrets/releases/download/v${KUBESEAL_VERSION}/kubeseal-${KUBESEAL_VERSION}-linux-amd64.tar.gz"
```

      - 압축 해제 및 이동

```
tar -xvzf kubeseal-${KUBESEAL_VERSION}-linux-amd64.tar.gz

sudo install -m 755 kubeseal /usr/local/bin/kubeseal
```

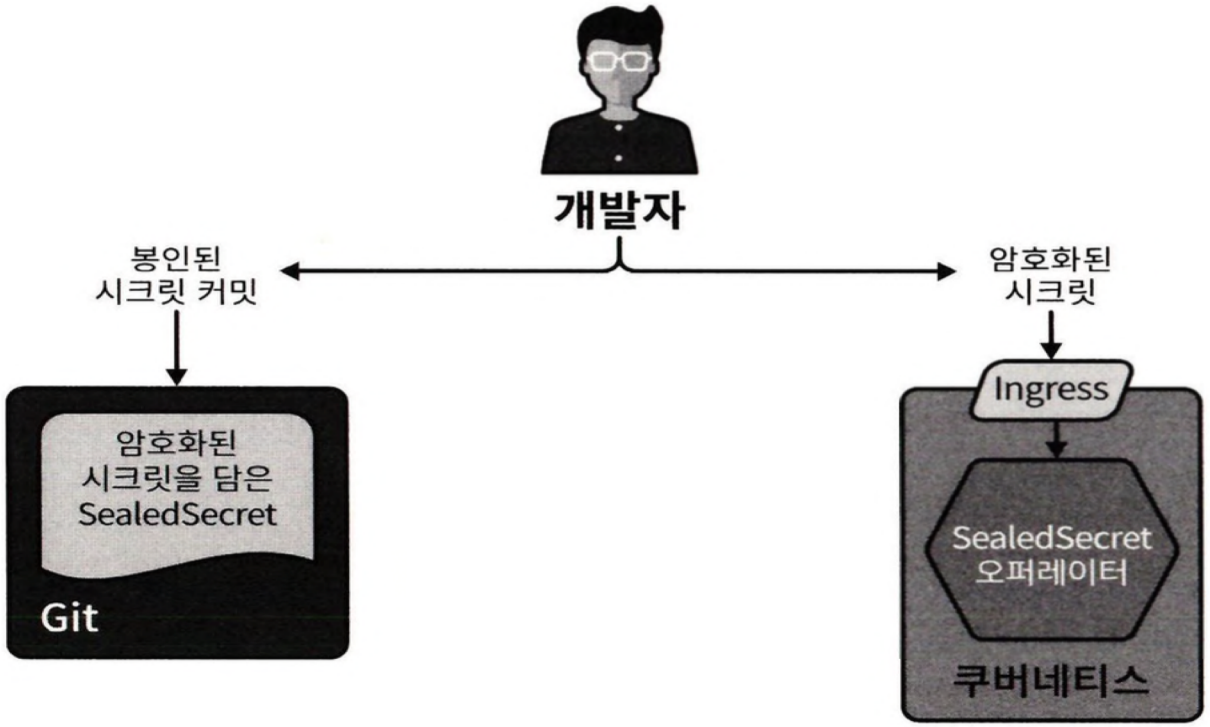
  - ✓ 설치 확인: kubeseal --version

민감한 데이터 암호화

- Sealed Secrets
  - ✓ Controller 설치: https://github.com/bitnami-labs/sealed-secrets/releases/
    - kubectl apply -f https://github.com/bitnami-labs/sealed-secrets/releases/download/v0.36.1/controller.yaml
  - ✓ 변환 프로세스
    - Input: kubectl이 생성한 일반 Secret(Base64 인코딩만 된 평문 상태)이 kubeseal로 전달
    - Encryption: kubeseal은 클러스터에 있는 Sealed Secrets Controller로부터 공개키(Public Key)를 가져와 데이터를 암호화
    - Output: 암호화된 SealedSecret 커스텀 리소스가 YAML 형태로 생성되는데 이 파일은 이제 GitHub에 올려도 안전
  - ✓ 주의사항
    - 네임스페이스(Namespace): kubeseal은 기본적으로 Secret이 생성될 네임스페이스와 이름을 기반으로 암호화하는데 default 네임스페이스에서 생성한 SealedSecret을 다른 네임스페이스에 배포하면 복호화되지 않음 (범위를 넓히려면 --scope 옵션을 사용)
    - 컨트롤러 연결: kubeseal 실행 시 클러스터에 접근 권한이 있어야 공개키를 가져올 수 있는데 오프라인 상태라면 미리 공개키를 파일로 저장(kubeseal --fetch-cert > pub-cert.pem)해서 사용

민감한 데이터 암호화

- Sealed Secrets
  - ✓ GitOps 와 Sealed Secrets



민감한 데이터 암호화

- ❖ Sealed Secrets
  - ✓ 변환 방법
    - ● yaml 파일의 경우
      - ◆ secret.yaml을 SealedSecret인 sealed-secret.yaml로 변환

```
cat secret.yaml | kubeseal --format yaml > sealed-secret.yaml
```

      - ◆ 명령어로 즉석 생성 및 변환: 파일을 따로 만들지 않고 kubectl로 Secret을 생성하는 동시에 kubeseal로 넘겨서 암호화된 파일을 만들 수도 있는데 이 방식이 중간에 암호화되지 않은 Secret 파일이 남지않아 더 안전

```
kubectl create secret generic my-secret ₩
  --from-literal=username=admin ₩
  --from-literal=password=password123 ₩
  --dry-run=client -o yaml | ₩
  kubeseal --format yaml > sealed-secret.yaml
```

민감한 데이터 암호화

- ❖ Sealed Secrets
  - ✓ secret 생성
    - kubectl create namespace pacman
    - kubectl create secret generic pacman-secret -n pacman --from-literal=user=pacman --from-literal=pass=pacman

      secret/pacman-secret created
    - yaml 확인: kubectl get secret pacman-secret -n pacman -o yaml

```
apiVersion: v1
data:
  pass: cGFjbWFu
  user: cGFjbWFu
kind: Secret
metadata:
  creationTimestamp: "2026-03-26T23:13:06Z"
  name: pacman-secret
  namespace: pacman
  resourceVersion: "437202"
  uid: 698a9ba5-858c-4b2f-a61e-173af855b4de
type: Opaque
```

민감한 데이터 암호화

- Sealed Secrets
  - ✓ SealedSecret 변환
    - 변환: kubectl get secret pacman-secret -n pacman -o yaml | kubeseal -o yaml --scope=namespace-wide > pacman-sealedsecret.yaml
    - 확인: nano pacman-sealedsecret.yaml

```
---
apiVersion: bitnami.com/v1alpha1
kind: SealedSecret
metadata:
  annotations:
    sealedsecrets.bitnami.com/namespace-wide: "true"
  name: pacman-secret
  namespace: pacman
spec:
  encryptedData:
    pass: AgCCJ+umh77V8zdlZKc+VzkX2xGcaFtKdwnCqAjLm+TjLxK+bclMrm+LXHBAE1rNEjS53MiYnq+2j7opk4EXZoo2ib0qTEDrKpBjoFnCb>
    user: AgCC+HidA8yDyoXNtQ7qlXt/l+8wCoy6J4w8V4PY2IdtxnPewsA/KK/LKPFiDMvsqh4ZTvBudFkV321BjeJShER+CzM3ilu8D6iQQORdn>
  template:
    metadata:
      annotations:
        sealedsecrets.bitnami.com/namespace-wide: "true"
                                                [ Read 19 lines ]
```

민감한 데이터 암호화

- Sealed Secrets
  - ✓ SealedSecret 변환
    - 파일을 로컬로 전송
      - 권한 변경: chmod 400 <pem 파일 경로>
      - 파일을 로컬의 현재 디렉토리에 전송: scp -i <pem 파일 경로> <계정>@<Public IP>:파일경로 ./
  - ✓ 기존 Secret 대신에 이 파일을 Git Hub에 Push
    - 봉인된 시크릿은 실제로 시스템에 반영되는 순간에 원본 시크릿 pacman-secret 으로 복호화되므로 봉인된 시크릿을 저장소에 푸시한 이후에는 네임스페이스 pacman에 만들었던 시크릿은 삭제해야 함
    - 시크릿 삭제: kubectl delete secret pacman-secret -n pacman
    - 변환된 파일을 git hub에 push

민감한 데이터 암호화

- ❖ Sealed Secrets
  - ✓ Argo CD 적용
    - ● sealedsecret.yaml 파일 생성

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: pacman-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: 'https://github.com/itstudy001/bgd.git'
    targetRevision: 'main'
    path: 'sealedsecrets'
  destination:
    server: 'https://kubernetes.default.svc'
    namespace: pacman
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
      - CreateNamespace=true # 네임스페이스가 없으면 자동 생성
```

민감한 데이터 암호화

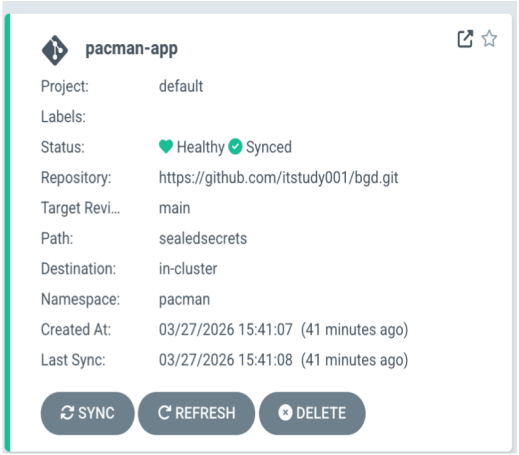
- ❖ Sealed Secrets
  - ✓ Argo CD 적용
    - sealedsecret.yaml 파일 적용: kubectl apply -f sealedsecrets.yaml
    - 확인
      - ◆ argocd app list

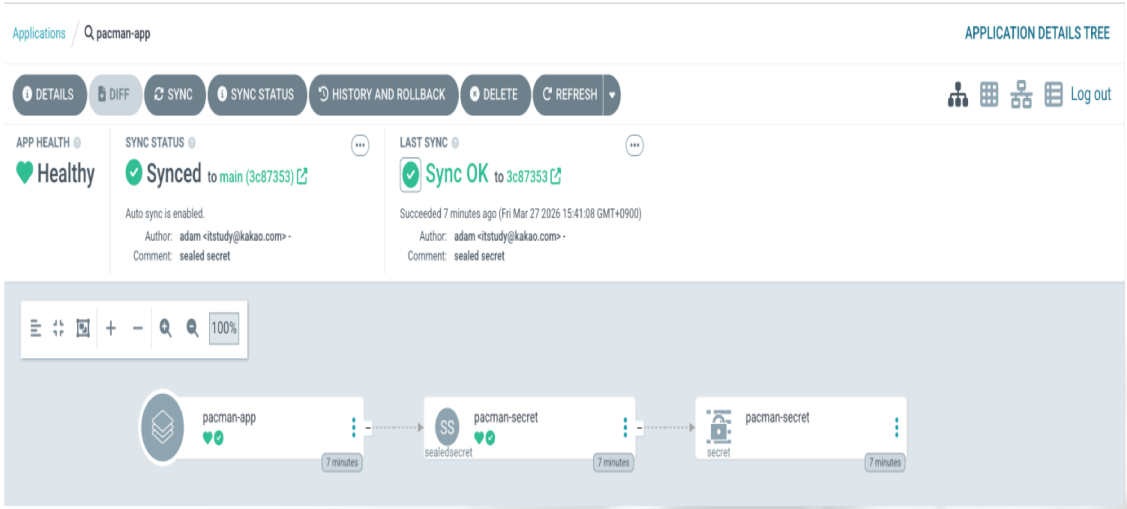
```
NAME                CLUSTER                         NAMESPACE  PROJECT
STATUS  HEALTH   SYNCPOLICY  CONDITIONS  REPO
PATH          TARGET

argocd/pacman-app  https://kubernetes.default.svc  pacman     default
Synced  Healthy  Auto-Prune  <none>
https://github.com/itstudy001/bgd.git  sealedsecrets  main
```

민감한 데이터 암호화

- ❖ Sealed Secrets
  - ✓ Argo CD 적용
    - ● 확인
      - ◆ Web UI





민감한 데이터 암호화

- ❖ Sealed Secrets
  - ✓ Argo CD 적용
    - ● 확인
      - ◆ kubectl get secrets -n pacman -o yaml

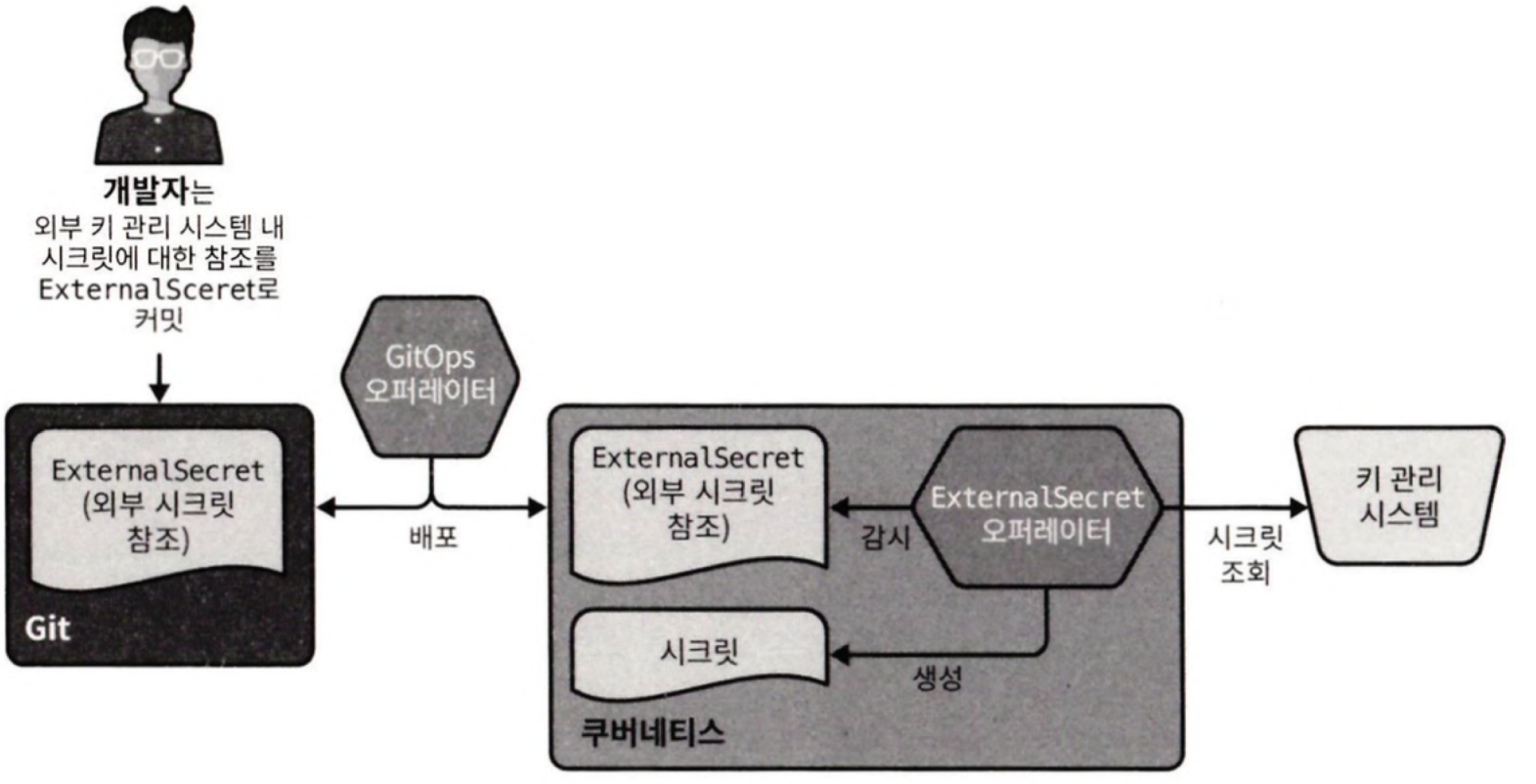
```
apiVersion: v1
items:
- apiVersion: v1
  data:
    pass: cGFjbWFu
    user: cGFjbWFu
  kind: Secret
  metadata:
    annotations:
      sealedsecrets.bitnami.com/namespace-wide: "true"
    creationTimestamp: "2026-03-27T06:41:07Z"
    name: pacman-secret
    namespace: pacman
```

민감한 데이터 암호화

- ❖ Argo CD + HashiCorp Vault + External Secret
  - ✓ 개요
    - 암호화된 자격 증명마저도 저장하고 싶지 않다면 외부 시크릿(External Secret)을 이용할 수 있는데 이 프로젝트는 외부 서비스 또는 다른 공급업체의 볼트에 시크릿을 저장한 다음 해당 시크릿에 대한 참조만 Git에 저장하는 고대 디(GoDaddy) 에서 자체 개발한 오픈 소스 프로젝트
    - 현재 외부 시크릿 저장소로는 AWS 시크릿 관리자(Secrets Manager), 하시코프 볼트(Hashcorp Vault), 구글 시크릿 관리자(Secret Manager), 애저 키 볼트(Azure Key Vault) 등을 지원
    - 시크릿을 저장하고 그 수명 주기를 관리하는 외부 API에 대한 사용자 친화적 추상화 계층을 제공하는 것이 핵심 아이디어
    - 외부 시크릿(External Secret)은 외부 키 관리 시스템에 저장된 시크릿에 대한 참조를 클러스터 내 시크릿으로 변환하는 쿠버네티스 컨트롤러
    - 사용자 지정 리소스 Secret store는 기밀 데이터가 저장된 백엔드 및 해당 기밀 데이터를 시크릿으로 변환하는 템플릿을 정의하는데 외부 시크릿 매니저에 연결하는 데 필요한 구성 정보도 포함하는 것으로 ExternalSecrets 객체는 외부 서비스가 관리하는 자격 증명 정보에 대한 참조만 포함하므로(실제 자격 증명은 포함하지 않음) Git에 안전하게 저장할 수 있음

민감한 데이터 암호화

❖ Argo CD + HashiCorp Vault + External Secret



Application Set

- 개요
  - ✓ Argo CD는 Applicationset 리소스를 통해 Argo CD Application 리소스를 템플릿(template)처럼 쓸 수 있도록 함
  - ✓ 자주 사용하는 중요한 기능
    - 같은 쿠버네티스 매니페스트를 다른 여러 클러스터에 배포
    - 하나 또는 여러 Git 저장소에 분산된 여러 애플리케이션을 함께 배포
    - Applcationset는 런타임에 실제 값들로 대체될 자리(place holders)가 마련되어 있는 템플릿 파일이므로 배포 전에 값을 바꿔 넣는 과정이 필요한데 이를 위해 Applcationset은 생성기(generator)라는 개념을 지원하는데 생성기는 인자 생성을 담당하고 생성된 인자는 템플릿의 빈자리를 대체하여 최종적으로 유효한 Argo CD Applcation을 만들어 냄

Application Set

- 여러 개의 Application 배포
  - ✓ prod 디렉토리에 배포 파일 생성: bgd.application.yaml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: bgd
  name: bgd
```

Application Set

- 여러 개의 Application 배포
  - ✓ prod 디렉토리에 배포 파일 생성: bgd.application.yaml

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bgd
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: bgd
    spec:
      containers:
      - image: ggnagpae1/bgd:1.0.0
        name: bgd
        env:
        - name: COLOR
          value: "green"
        resources: {}
---
```

Application Set

- 여러 개의 Application 배포
  - ✓ staging 디렉토리에 배포 파일 생성: bgd.application.yaml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: bgd
  name: bgd
```

Application Set

- 여러 개의 Application 배포
  - ✓ staging 디렉토리에 배포 파일 생성: bgd.application.yaml

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bgd
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: bgd
    spec:
      containers:
      - image: ggnagpae1/bgd:1.0.0
        name: bgd
        env:
        - name: COLOR
          value: "blue"
        resources: {}
---
```

Application Set

- 여러 개의 Application 배포
  - ✓ Github에 Push
  - ✓ 배포 파일 작성 - bgd-application-set.yaml

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: bgd-app
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - cluster: staging
        url: https://kubernetes.default.svc
        location: default
      - cluster: prod
        url: https://kubernetes.default.svc
        location: app
```

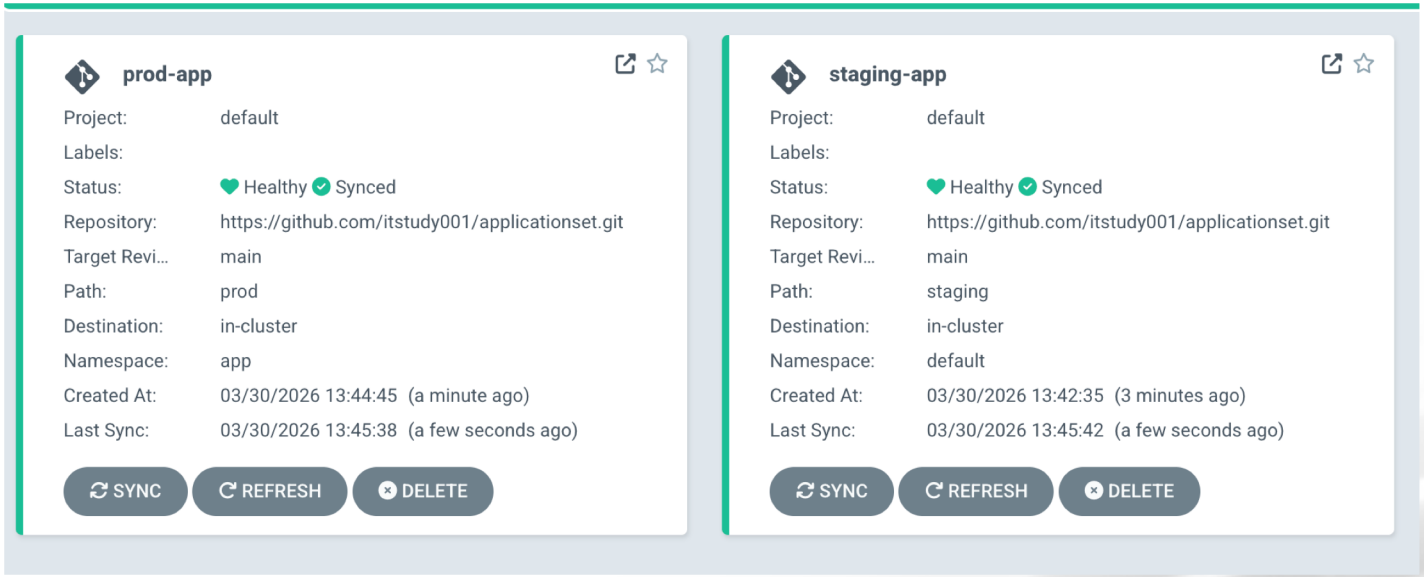
Application Set

- 여러 개의 Application 배포
  - ✓ 배포 파일 작성 - bgd-application-set.yaml

```
template:
  metadata:
    name: '{{cluster}}-app'
  spec:
    project: default
    source:
      repoURL: https://github.com/itstudy001/applicationset.git
      targetRevision: main
      path: '{{cluster}}'
    destination:
      server: '{{url}}'
      namespace: '{{location}}'
    syncPolicy:
      syncOptions:
      - CreateNamespace=true
```

Application Set

- 여러 개의 Application 배포
  - ✓ 배포: kubectl apply -f bgd-application-set.yaml
  - ✓ 확인



Application Set

- 여러 개의 Application 배포
  - ✓ 확인: argocd app list

```
NAME                CLUSTER                         NAMESPACE  PROJECT  STATUS
    HEALTH   SYNCPOLICY  CONDITIONS  REPO
    PATH     TARGET
argocd/prod-app     https://kubernetes.default.svc  app        default  Synced  Healthy
    Manual       <none>       https://github.com/itstudy001/applicationset.git  prod
    main
argocd/staging-app  https://kubernetes.default.svc  default    default  Synced  Healthy
    Manual       <none>       https://github.com/itstudy001/applicationset.git  staging
    main
```

  - ✓ 삭제: kubectl delete -f bgd-application-set.yaml

Application Set

- 생성기 종류
  - ✓ List: 고정된 클러스터 목록에서 Application 정의를 생성
  - ✓ Cluster: List와 유사하지만 Argo CD에 정의된 클러스터 목록을 기반으로 생성
  - ✓ Git: Git 저장소에 보관된 JSON/YAML 속성 파일(properties file)이나 디렉토리 구조에서 Application 정의를 자동 생성
  - ✓ SCM Provider: 조직 내 코드 저장소에서 Application 정의를 생성
  - ✓ Pull Request (PR): 새로 열린 PR에서 Application 정의를 생성
  - ✓ Cluster Decision Resource: 덕 타이핑을 통해 Application 정의를 생성
  - ✓ Matrix: 두 개의 개별 생성기가 만들어 낸 값을 조합
  - ✓ Merge: 두 개 이상의 생성기가 만들어 낸 값을 병합

Application Set

- Git 생성기를 이용한 애플리케이션 배포
  - ✓ 현재의 디렉토리 구조를 수정: bgd-gen 디렉토리를 생성해서 2개의 디렉토리를 bgd-gen 안으로 이동
  - ✓ 배포 매니페스트 파일

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: bgd-app
  namespace: argocd
spec:
  generators:
  - git:
      repoURL: https://github.com/itstudy001/applicationset.git
      revision: main
      directories:
      - path: bgd-gen/*
```

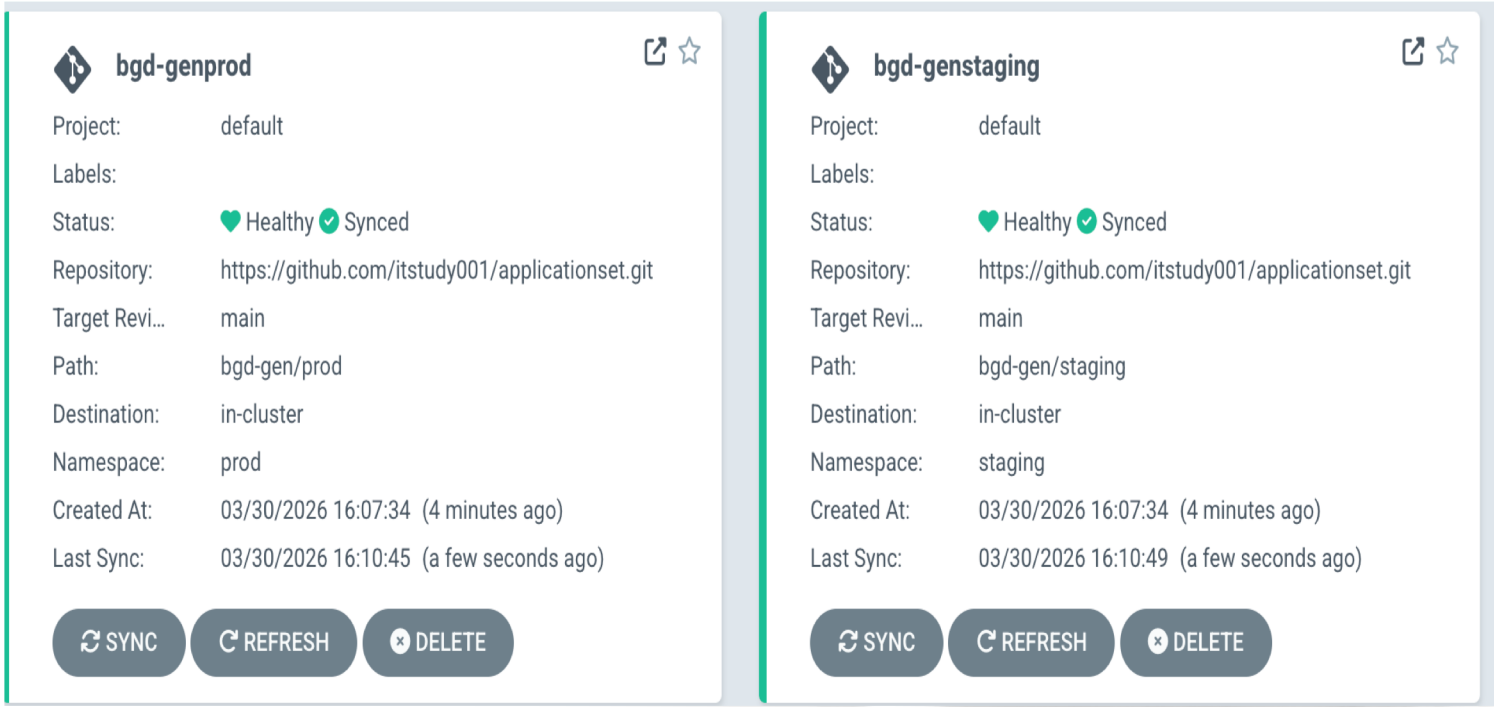
Application Set

- Git 생성기를 이용한 애플리케이션 배포
  - ✓ 배포 매니페스트 파일

```
template:
  metadata:
    name: '{{path[0]}}{{path[1]}}'
  spec:
    project: default
    source:
      repoURL: https://github.com/itstudy001/applicationset.git
      targetRevision: main
      path: '{{path}}'
    destination:
      server: https://kubernetes.default.svc
      namespace: '{{path.basename}}'
    syncPolicy:
      automated:       # 자동 동기화 활성화 (선택 사항이나 권장)
        prune: true
        selfHeal: true
      syncOptions:
      - CreateNamespace=true
```

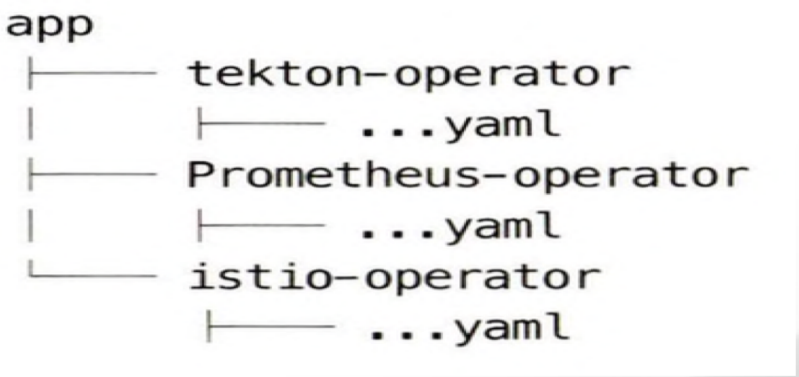
Application Set

- Git 생성기를 이용한 애플리케이션 배포
  - ✓ 배포



Application Set

- Git 생성기를 이용한 애플리케이션 배포
  - ✓ 장점
    - 애플리케이션이 많은 구성 요소(서비스, 데이터베이스, 분산 캐시, 이메일 서버 등)로 이루어져 있으며 그 각각의 배포 파일이 다른 디렉토리에 배치된 경우 유용
    - 모든 쿠버네티스 오퍼레이터가 담긴 저장소를 클러스터에 한 번에 배포해야 할 때도 좋음



PR 배포

❖ 개요

- ✓ pullRequest 생성기를 사용하면 저장소에 열린 PR에 기반하여 Application 객체를 자동으로 만들어 낼 수 있음
- ✓ 이전 Application 삭제: kubectl delete applicationset bgd-app -n argocd
- ✓ bgd-pr 이라는 디렉토리를 만들고 deployment 파일을 생성

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: bgd
  name: bgd
```

PR 배포

❖ 개요

✓ bgd-pr 이라는 디렉토리를 만들고 deployment 파일을 생성

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bgd
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: bgd
    spec:
      containers:
      - image: ggnagpae1/bgd:1.0.0
        name: bgd
        env:
        - name: COLOR
          value: "blue"
        resources: {}
---
```

PR 배포

❖ 개요

✓ bgd-pr 이라는 디렉토리를 만들고 deployment 파일을 생성

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bgd
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: bgd
    spec:
      containers:
      - image: ggnagpae1/bgd:1.0.0
        name: bgd
        env:
        - name: COLOR
          value: "blue"
        resources: {}
---
```

PR 배포

❖ 개요

✓ Git Hub에 Push

✓ 배포 파일 생성

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: bgd-app-set
  namespace: argocd
spec:
  generators:
  - pullRequest:
      github:
        owner: itstudy001
        repo: applicationset
        labels:
        - preview
      requeueAfterSeconds: 60
```

PR 배포

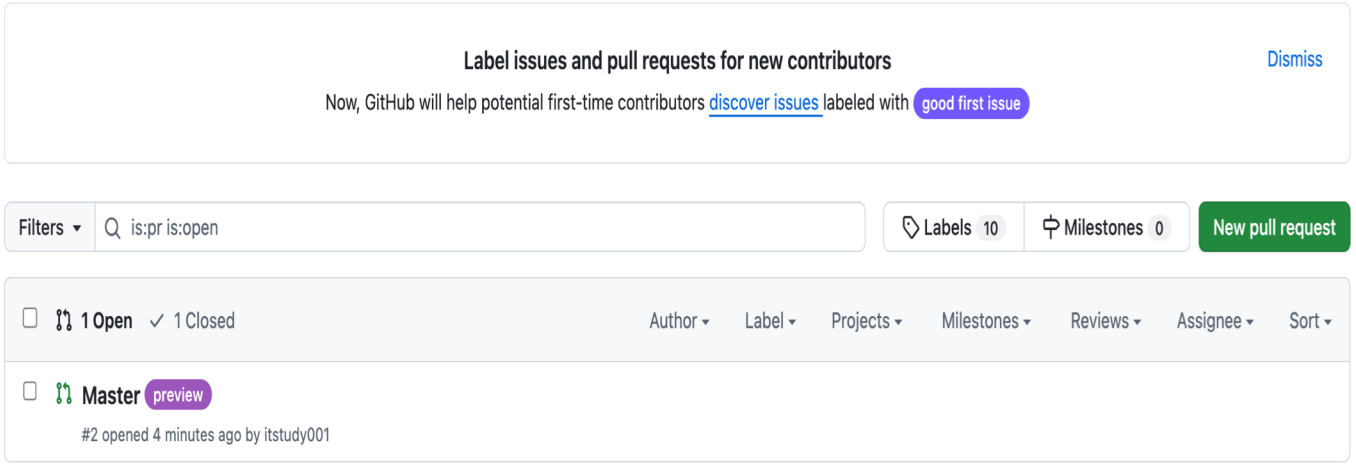
- 개요
  - ✓ 배포 파일 생성

```
template:
  metadata:
    name: 'bgd-{{branch}}-{{number}}'
    namespace: argocd
  spec:
    project: default
    source:
      repoURL: https://github.com/itstudy001/applicationset.git
      targetRevision: '{{head_sha}}'
      path: bgd-pr
    destination:
      server: https://kubernetes.default.svc
      namespace: '{{branch}}-{{number}}'
    syncPolicy:
      automated: {}
      syncOptions:
      - CreateNamespace=true
```

PR 배포

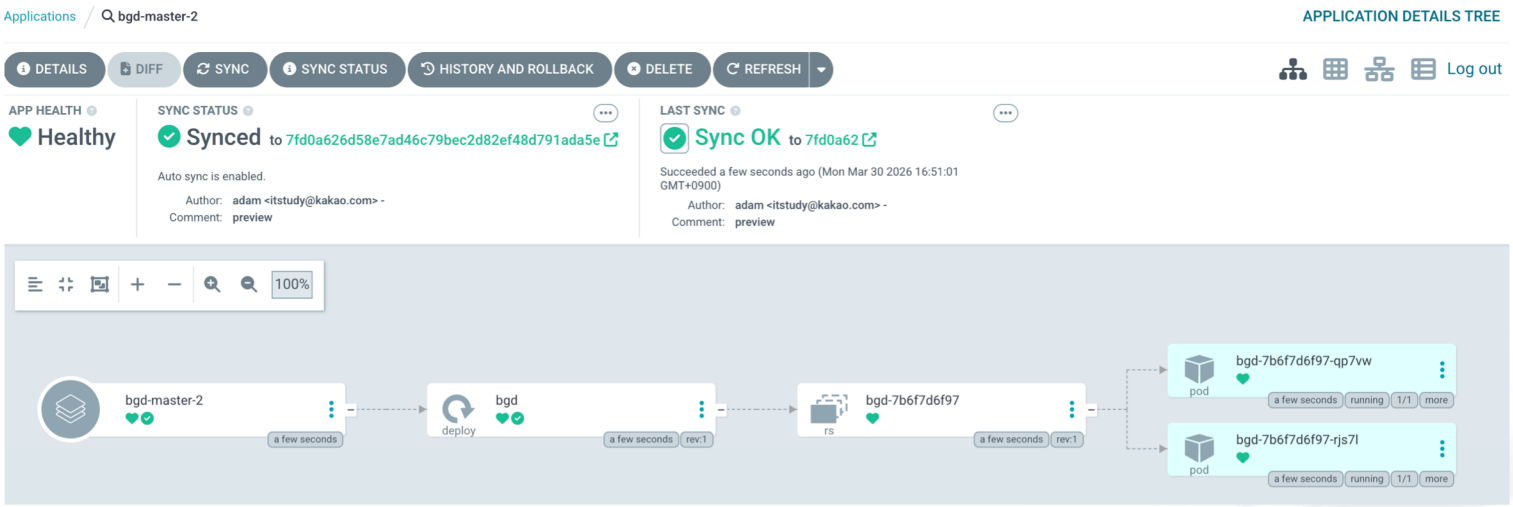
❖ 개요

- ✓ ApplicationSet 배포
- ✓ 확인: argocd app list(아직 Application이 생성 안됨)
- ✓ 브랜치를 생성하고 코드를 수정해서 배포
  - git checkout -b master
  - git add .
  - git commit –m
  - git push origin master
- ✓ Git hub 사이트에서 PR을 생성하는데 레이블을 preview로 설정



PR 배포

- 개요
  - ✓ 확인


Applications
bgd-master-2
APPLICATION DETAILS TREE
DETAILS
DIFF
SYNC
SYNC STATUS
HISTORY AND ROLLBACK
DELETE
REFRESH
Log out
APP HEALTH
Healthy
SYNC STATUS
Synced to 7fd0a626d58e7ad46c79bec2d82ef48d791ada5e
Auto sync is enabled.
Author: adam <itstudy@kakao.com> -
Comment: preview
LAST SYNC
Sync OK to 7fd0a62
Succeeded a few seconds ago (Mon Mar 30 2026 16:51:01 GMT+0900)
Author: adam <itstudy@kakao.com> -
Comment: preview
100%
bgd-master-2
a few seconds
deploy
bgd
a few seconds
rev:1
rs
bgd-7b6f7d6f97
a few seconds
rev:1
pod
bgd-7b6f7d6f97-qp7vw
a few seconds
running
1/1
more
pod
bgd-7b6f7d6f97-rjs7l
a few seconds
running
1/1
more

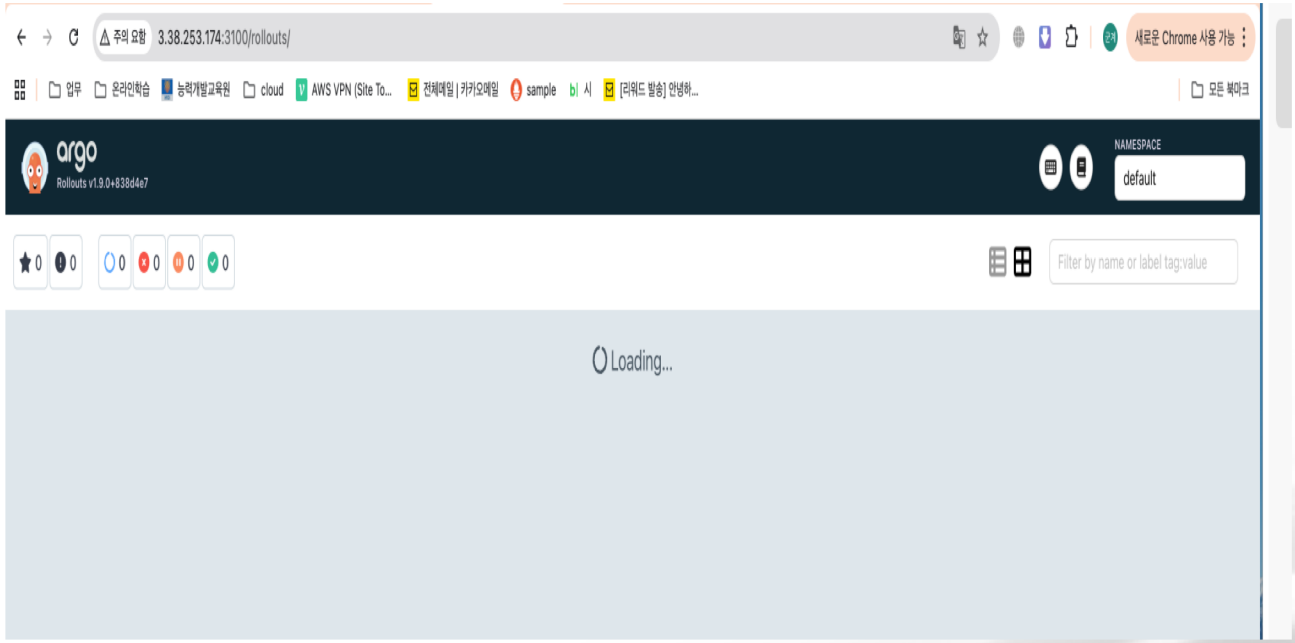
Argo Rollout

❖ 개요

- ✓ Argo 롤아웃은 블루-그린, 카나리, 미러링(mirroring), 다크 카나리(dark canary), 트래픽 분석(traffic analysis) 등의 고급 배포 기술을 쿠버네티스에 제공하는 컨트롤러
- ✓ 트래픽 관리를 위한 앰배서더(Ambassador), 이스티오(Istio), AWS 로드밸런서 컨트롤러(Load Balancer Controller), NGNI, SMI 또는 트래픽(Traefik) 등의 쿠버네티스 프로젝트와 연동되며 프로메테우스(Prometheus), 데이터독(Datadog), 뉴 렐릭(New Relic) 등의 제품과 연동하여 지속적 배포 상태를 모니터 링 할 수 있도록 함
- ✓ Argo rollouts Controller 설치
  - kubectl create namespace argo-rollouts
  - kubectl apply -n argo-rollouts -f https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml
  - kubectl get pods -n argo-rollouts
- ✓ Argo rollouts kubectl Plugin 설치
  - curl -LO https://github.com/argoproj/argo-rollouts/releases/latest/download/kubectl-argo-rollouts-linux-amd64
  - chmod +x ./kubectl-argo-rollouts-linux-amd64
  - sudo mv ./kubectl-argo-rollouts-linux-amd64 /usr/local/bin/kubectl-argo-rollouts
  - kubectl argo rollouts version

Argo Rollout

- 개요
  - ✓ 대시보드 실행: kubectl argo rollouts dashboard



Argo Rollout

- Canary Update 실습
  - ✓ Argo Rollout은 표준 쿠버네티스 Deployment가 아닌 Rollout이라는 새로운 리소스를 사용 Deployment 객체의 모든 옵션을 지원할 뿐 아니라 롤링 업데이트를 구성하는 필드들을 추가로 지원
  - ✓ 개요
    - 트래픽의 20%를 새 버전으로 전달
    - 운영자가 프로세스를 계속 진행하기로 결정할 때까지 기다림
    - 트래픽의 40%, 60%, 80%를 자동으로 새 버전으로 전달하고 트래픽을 늘릴 때마다 10초씩 대기

Argo Rollout

- Canary Update 실습
  - ✓ rollout.yaml

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: rollouts-demo
spec:
  replicas: 5
  strategy:
    canary:
      # 단계별 배포 설정
      steps:
      - setWeight: 20    # 1단계: 새 버전에 트래픽 20% 전달
      - pause: {}        # 2단계: 무기한 중단 (관리자가 승인할 때까지 대기)
      - setWeight: 40    # 3단계: 승인 후 40%로 확대
      - pause: {duration: 10s} # 4단계: 10초 대기 후 자동 진행
      - setWeight: 60
      - pause: {duration: 10s}
      - setWeight: 80
      - pause: {duration: 10s}
```

Argo Rollout

- Canary Update 실습
  - ✓ rollout.yaml

```
revisionHistoryLimit: 2
selector:
  matchLabels:
    app: rollouts-demo
template:
  metadata:
    labels:
      app: rollouts-demo
  spec:
    containers:
    - name: rollouts-demo
      image: argoproj/rollouts-demo:blue # 처음에는 blue 버전으로 시작
      ports:
      - name: http
        containerPort: 8080
        protocol: TCP
```

Argo Rollout

- Canary Update 실습
  - ✓ service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: rollouts-demo
spec:
  ports:
  - port: 80
    targetPort: http
    protocol: TCP
    name: http
  selector:
    app: rollouts-demo
```

Argo Rollout

- Canary Update 실습
  - ✓ 초기 배포
    - kubectl apply -f rollout.yaml
    - kubectl apply -f service.yaml
    - 현재 상태: kubectl get pods

```
NAME                              READY   STATUS    RESTARTS   AGE
rollouts-demo-759566c557-2gr4k    1/1     Running   0          19s
rollouts-demo-759566c557-4hnxt    1/1     Running   0          19s
rollouts-demo-759566c557-9swqq    1/1     Running   0          19s
rollouts-demo-759566c557-lqgfp    1/1     Running   0          19s
rollouts-demo-759566c557-s998z    1/1     Running   0          20s
```

Argo Rollout

- ❖ Canary Update 실습
  - ✓ 이미지 업데이트
    - kubectl argo rollouts set image rollouts-demo rollouts-demo=argoproj/rollouts-demo:yellow
    - 현재 상태: kubectl get pods

```
NAME                                READY   STATUS    RESTARTS   AGE
rollouts-demo-759566c557-2gr4k      1/1     Running   0          51s
rollouts-demo-759566c557-4hnxt      1/1     Running   0          51s
rollouts-demo-759566c557-lqgfp      1/1     Running   0          51s
rollouts-demo-759566c557-s998z      1/1     Running   0          52s
rollouts-demo-85b6995845-k5fts      1/1     Running   0          15s
```

Argo Rollout

- ❖ Canary Update 실습
  - ✓ 상태 모니터링
    - kubectl argo rollouts get rollout rollouts-demo --watch

```
Name:            rollouts-demo
Namespace:       default
Status:          Healthy
Name:            rollouts-demo
Namespace:       default
Status:          Healthy
Strategy:        Canary
  Step:          8/8
  SetWeight:     100
  ActualWeight:  100
Images:          argoproj/rollouts-demo:yellow (stable)
Replicas:
  Desired:       5
  Current:       5
  Updated:       5
  Ready:         5
  Available:     5
```

Argo Rollout

- ❖ Canary Update 실습
  - ✓ 상태 모니터링
    - kubectl argo rollouts get rollout rollouts-demo --watch

```
NAME                                      KIND        STATUS        AGE    INFO
⟳ rollouts-demo                           Rollout     Healthy       9m49s
├──# revision:2
│  └──⧉ rollouts-demo-85b6995845          ReplicaSet  Healthy       9m12s  stable
│     ├──□ rollouts-demo-85b6995845-k5fts  Pod         Running       9m12s  ready:1/1
│     ├──□ rollouts-demo-85b6995845-mddl6  Pod         Running       8m16s  ready:1/1
│     ├──□ rollouts-demo-85b6995845-dkvnb  Pod         Running       8m2s   ready:1/1
│     ├──□ rollouts-demo-85b6995845-25slq  Pod         Running       7m51s  ready:1/1
│     └──□ rollouts-demo-85b6995845-fhw8h  Pod         Running       7m40s  ready:1/1
└──# revision:1
```

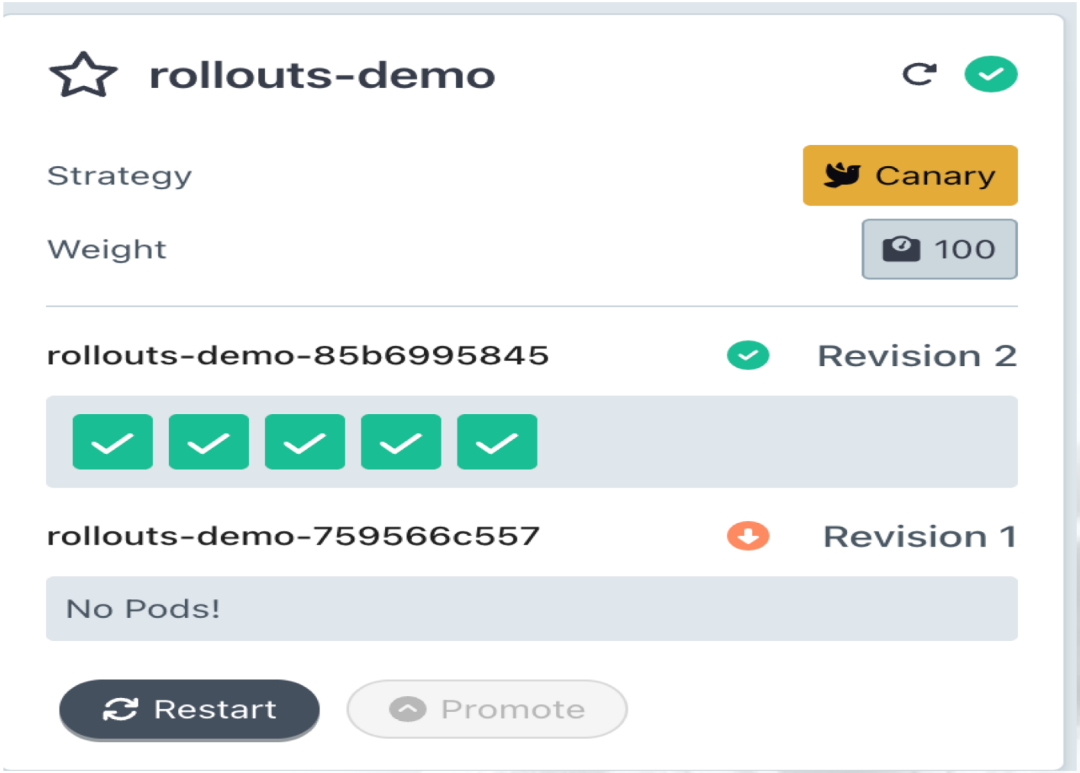
Argo Rollout

- Canary Update 실습
  - ✓ 다음 단계로 진행
    - kubectl argo rollouts promote rollouts-demo
    - 현재 상태: kubectl get pods

```
NAME                              READY   STATUS    RESTARTS   AGE
rollouts-demo-85b6995845-25slq    1/1     Running   0          9m8s
rollouts-demo-85b6995845-dkvnb    1/1     Running   0          9m19s
rollouts-demo-85b6995845-fhw8h    1/1     Running   0          8m57s
rollouts-demo-85b6995845-k5fts    1/1     Running   0          10m
rollouts-demo-85b6995845-mddl6    1/1     Running   0          9m33s
```

Argo Rollout

- ❖ Canary Update 실습
  - ✓ 대시보드로 확인
    - kubectl argo rollouts dashboard



Argo Rollout

- Canary Update 실습
  - ✓ bgd-rollout.yaml

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: bgd-rollouts
spec:
  replicas: 5
  strategy:
    canary:
      steps:
      - setWeight: 20
      - pause: {}
      - setWeight: 40
      - pause: {duration: 30s}
      - setWeight: 60
      - pause: {duration: 30s}
      - setWeight: 80
      - pause: {duration: 30s}
```

Argo Rollout

- Canary Update 실습
  - ✓ bgd-rollout.yaml

```
revisionHistoryLimit: 2
selector:
  matchLabels:
    app: bgd-rollouts
template:
  metadata:
    labels:
      app: bgd-rollouts
  spec:
    containers:
    - image: ggnagpae1/bgd:1.0.0
      name: bgd
      env:
      - name: COLOR
        value: "blue"
      resources: {}
```

Argo Rollout

- Canary Update 실습
  - ✓ 배포: kubectl apply -f bgd-rollout.yaml
  - ✓ 확인: kubectl get pods

```
NAME                                READY   STATUS    RESTARTS   AGE
bgd-rollouts-65bf89656c-8wkmz       1/1     Running   0          5m
bgd-rollouts-65bf89656c-ksc75       1/1     Running   0          5m
bgd-rollouts-65bf89656c-vk5v7       1/1     Running   0          5m
bgd-rollouts-65bf89656c-xlkfm       1/1     Running   0          5m
bgd-rollouts-69995dd557-z4nh2       1/1     Running   0          5m
```

  - ✓ 확인: kubectl argo rollouts get rollout bgd-rollouts

Argo Rollout

- Canary Update 실습
  - ✓ bgd-rollout2.yaml

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: bgd-rollouts
spec:
  replicas: 5
  strategy:
    canary:
      steps:
      - setWeight: 20
      - pause: {}
      - setWeight: 40
      - pause: {duration: 30s}
      - setWeight: 60
      - pause: {duration: 30s}
      - setWeight: 80
      - pause: {duration: 30s}
```

Argo Rollout

- Canary Update 실습
  - ✓ bgd-rollout2.yaml

```
revisionHistoryLimit: 2
 selector:
   matchLabels:
     app: bgd-rollouts
 template:
   metadata:
     labels:
       app: bgd-rollouts
   spec:
     containers:
     - image: ggnagpae1/bgd:1.0.0
       name: bgd
       env:
       - name: COLOR
         value: "green"
       resources: {}
```

Argo Rollout

- Canary Update 실습
  - ✓ 배포: kubectl apply -f bgd-rollout2.yaml
  - ✓ 확인: kubectl get pods

```
AME                                READY   STATUS    RESTARTS   AGE
bgd-rollouts-65bf89656c-8wkmz      1/1     Running   0          5m
bgd-rollouts-65bf89656c-ksc75      1/1     Running   0          5m
bgd-rollouts-65bf89656c-vk5v7      1/1     Running   0          5m
bgd-rollouts-65bf89656c-xlkfm      1/1     Running   0          5m
bgd-rollouts-69995dd557-z4nh2      1/1     Running   0          29s
```

Argo Rollout

- ❖ Canary Update 실습
  - ✓ 확인: kubectl argo rollouts get rollout bgd-rollouts

```
NAME                                      KIND        STATUS     AGE    INFO
⟳ bgd-rollouts                            Rollout     ॥ Paused   5m15s
├──# revision:2
│  └──⧉ bgd-rollouts-69995dd557           ReplicaSet  Healthy    43s    canary
│     └──□ bgd-rollouts-69995dd557-z4nh2  Pod         Running    43s    ready:1/1
└──# revision:1
   └──⧉ bgd-rollouts-65bf89656c           ReplicaSet  Healthy    5m15s  stable
      ├──□ bgd-rollouts-65bf89656c-8wkmz  Pod         Running    5m14s  ready:1/1
      ├──□ bgd-rollouts-65bf89656c-ksc75  Pod         Running    5m14s  ready:1/1
      ├──□ bgd-rollouts-65bf89656c-vk5v7  Pod         Running    5m14s  ready:1/1
      └──□ bgd-rollouts-65bf89656c-xlkfm  Pod         Running    5m14s  ready:1/1
```

Argo Rollout

- Canary Update 실습
  - ✓ 업데이트: kubectl argo rollouts promote bgd-rollouts
  - ✓ 확인: kubectl get pods

```
NAME                              READY   STATUS    RESTARTS   AGE
bgd-rollouts-65bf89656c-8wkmz     1/1     Running   0          13m
bgd-rollouts-65bf89656c-xlkfm     1/1     Running   0          13m
bgd-rollouts-69995dd557-6nmhs     1/1     Running   0          35s
bgd-rollouts-69995dd557-xk9tm     1/1     Running   0          4s
bgd-rollouts-69995dd557-z4nh2     1/1     Running   0          8m37s
```

Argo Rollout

❖ Canary Update 실습

✓ 확인: kubectl argo rollouts get rollout bgd-rollouts

```
NAME                                      KIND        STATUS        AGE    INFO
⟳ bgd-rollouts                            Rollout     Healthy       13m
 ├──# revision:2
 │  └──⧉ bgd-rollouts-69995dd557          ReplicaSet  Healthy       9m24s  stable
 │     ├──□ bgd-rollouts-69995dd557-z4nh2  Pod         Running       9m24s  ready:1/1
 │     ├──□ bgd-rollouts-69995dd557-6nmhs  Pod         Running       82s    ready:1/1
 │     ├──□ bgd-rollouts-69995dd557-xk9tm  Pod         Running       51s    ready:1/1
 │     ├──□ bgd-rollouts-69995dd557-w69v6  Pod         Running       21s    ready:1/1
 │     └──□ bgd-rollouts-69995dd557-nn2js  Pod         Running       10s    ready:1/1
 └──# revision:1
    └──⧉ bgd-rollouts-65bf89656c           ReplicaSet  • ScaledDown  13m
```

Argo Rollout

- ❖ Canary Update 실습
  - ✓ 대시보드로 확인
    - kubectl argo rollouts dashboard