Argo CD

GitOps

- GitOps
  - ✓ 개요
    - GitOps는 Git 저장소를 단일 소스로 사용하여 인프라를 코드로 제공하는 방법론 및 관행
    - DevOps 문화가 이룩한 핵심 성과와 접근법을 실현하는 프레임워크를 제공
    - DevOps, 플랫폼 엔지니어링(platform engineering), SRE(Site Reliability Engineering – 사이트 안정성 엔지니어링) 등을 개선하고 구현하는 방법으로 널리 채택되고 있기 때문에 DevOps와 밀접한 관계를 갖음
    - GitOps는 구현 방식에 열린(agnostic) 접근법으로 GitOps 프레임워크는 Git, Kubernetes 및 CI/CD 솔루션 등으로 구축할 수 있음

GitOps

- GitOps
  - ✓ GitOps 원칙

    개발자 → Git 저장소(SSOT) → GitOps Controller(ArgoCD/Flux) → 클러스터/인프라

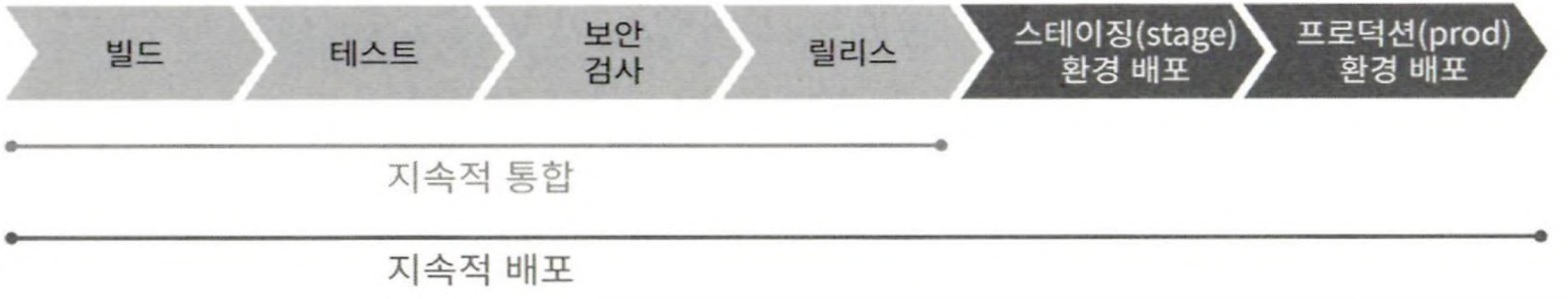
    - 선언적 구성
      - GitOps로 관리되는 시스템은 원하는 상태를 선언적으로 표현
      - 컨테이너 3개를 만들어라 와 같이 명령적인 방식이 아니라 이 애플리케이션에 3개의 컨테이너를 사용하겠다고 선언하면 에이전트가 3이라는 숫자를 맞춰주는 방식으로 지금 컨테이너가 5개 동작 중이라면 에이전트가 2개의 컨테이너를 종료 시킴
    - 버전이 제어되는 불변의 저장소: 깃은 버전이 제어되는 불변의 저장소로 현재 가장 많이 사용되는 소스 제어 시스템인데 유일한 것은 아니기 때문에 다른 소스 제어 시스템도 GitOps를 구현할 수 있음
    - 자동 반영(automatic pull) 또는 자동화된 배포(automated delivery): 배포 환경에 설치된 소프트웨어 에이전트가 원하는 상태에 대한 선언적 표현을 Git 저장소에서 자동으로 끌어와야 함
    - 지속적 조정(continuous reconciliation): 설정이 업데이트 된 후 소프트웨어 에이전트는 실제 시스템 상태를 계속 관찰하고 원하는 상태에 맞도록 변경

GitOps

- GitOps를 사용하는 이유
  - ✓ GitOps는 개발자에게 익숙한 보편적 Git 기반 워크플로를 사용하여 애플리케이션 개발부터 배포, 앱 수명 주기(life cycle) 관리 그리고 인프라 구성에 이르는 기존 프로세스를 확장
  - ✓ 애플리케이션 수명 주기 동안의 모든 변경 사항은 Git 저장소 기록을 통해 추적하고 감사(audit)하는데 이 접근법은 개발팀과 운영팀 모두에게 좋은데 문제를 신속하게 추적하고 재현할 수 있어 보안이 전반적으로 개선되기 때문
  - ✓ 핵심은 원치 않는 변경(drift) 때문에 발생하는 위험을 줄이고 프로덕션에 배포되기 전에 수정하는 것
  - ✓ GitOps 도입의 이점
    - 표준 워크플로: 애플리케이션 개발팀에서 익숙한 도구 와 Git 워크플로를 사용
    - 강화된 보안: 변경 사항을 미리 검토하고 예기치 않은 구성 변동을 감지하여 조치를 취함
    - 가시성 및 감사: Git 저장소의 변경 이력을 통해 클러스터의 모든 변경 사항을 캡처하고 추적
    - 멀티클러스터 일관성: 많은 환경과 Kubernetes 클러스터들에 대한 배포를 안정적이고 일관되게 구성

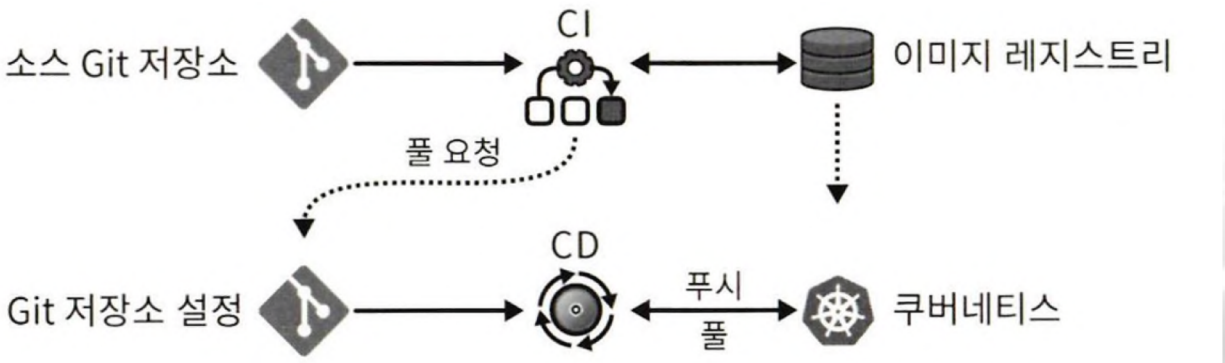
GitOps

- Kubernetes CI/CD
  - ✓ CI/CD
    - CI/CD 파이프라인의 경우 CI 프로세스는 제출된 코드를 확인하고 CD 프로세스는 보안, IaC 또는 애플리케이션 프레임워크에 설정된 경계(boundary) 등에 대한 요구 사항을 확인하고 적용
    - 모든 코드 변경 사항을 추적하므로 업데이트가 쉬워지며 롤백이 필요한 경우에 대비해 버전 관리 기능도 제공
    - CD는 GitOps 도메인이며 CI와 연동하여 다양한 환경에 앱을 배포



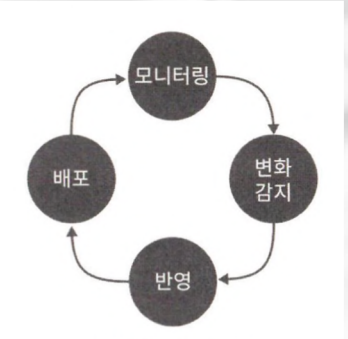
GitOps

- ❖ Kubernetes CI/CD
  - ✓ Kubernetes CI/CD
    - Kubernetes를 사용하면 클러스터 내부에 CI/CD 파이프라인을 쉽게 구현 가능
    - CI 소프트웨어는 애플리케이션을 나타내는 컨테이너 이미지를 생성하여 컨테이너 이미지 레지스트리에 저장
    - pull request 등의 Git 워크플로를 통해 배포할 앱의 명세서 격 인 manifest를 변경한 후 CD 동기화 루프를 개시



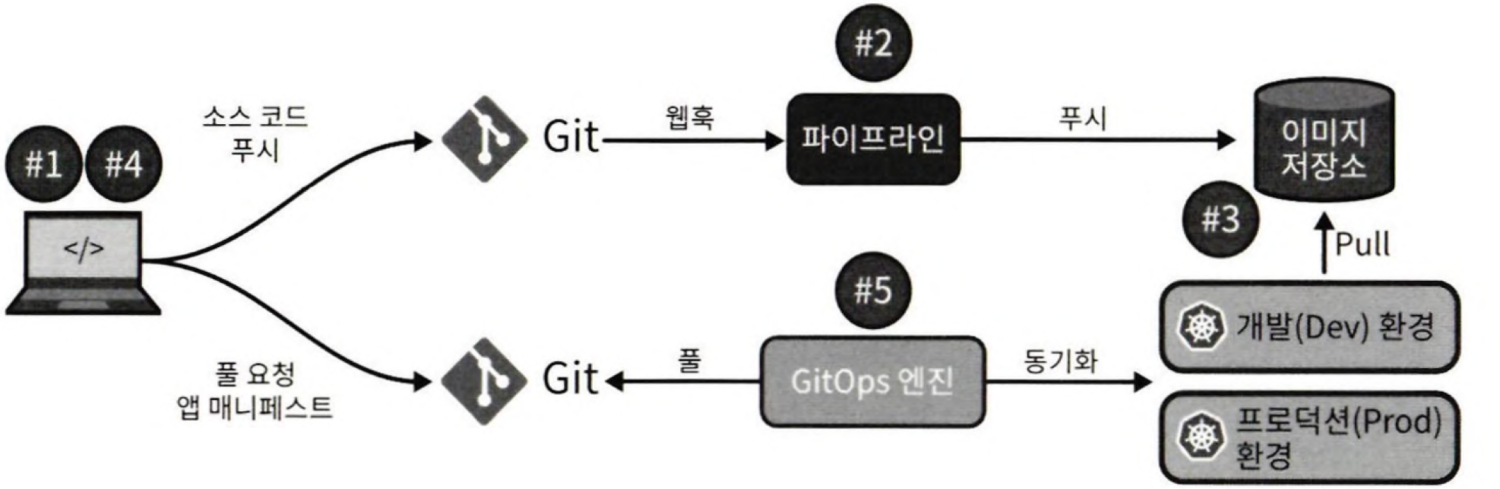
GitOps

- Kubernetes에 GitOps를 접목한 앱 배포 방법
  - ✓ GitOps는 플랫폼에 구애받지 않는 열린 접근법이므로 그 애플리케이션 배포 모델은 Kubernetes 클러스터 내부(in-cluster)에 구축될 수도 있고 여러 클러스터(multi-cluster)에 걸친 공통 플랫폼 형태로 구축될 수도 있음
  - ✓ Kubernetes 외부에 GitOps 도구를 구축하면 Kubernetes는 앱 배포를 위한 대상 플랫폼으로만 이용되고 클러스터 내부에 GitOps 도구를 구축하면 GitOps 엔진은 Kubernetes 클러스터 안에서 실행되어 해당 클러스터에 앱을 스스로 동기화
  - ✓ GitOps 생명 주기
    - 배포: Git에 저장된 manifest를 배포
    - 모니터링: Git 저장소나 클러스터 상태를 모니터링
    - 변화(drift) 감지: Git에 설명된 내용과 클러스터의 실제 구성 사이의 차이를 감지
    - 반영
      - Git에 있는 내용을 클러스터에 반영하는 작업(롤백 또는 3-way diff)을 실행
      - Git 저장소의 내용만을 신뢰 가능한 정보로 간주
      - 모든 변경은 Git 워크플로를 통해 수행



GitOps

- Kubernetes에 GitOps를 접목한 앱 배포 방법
  - ✓ Kubernetes에서 GitOps 접근법을 사용하여 애플리케이션을 배포하려면 최소 두 개의 Git 저장소가 필요한데 하나는 애플리케이션 소스 코드를 보관하기 위해 사용하고 다른 하나는 앱의 배포 형상을 기술하는 Kubernetes manifest를 보관하기 위해 사용 (Deployment, Service 등)
  - ✓ Kubernetes에 구축한 GitOps 프로젝트의 구조



GitOps

- Kubernetes에 GitOps를 접목한 앱 배포 방법
  - ✓ 중요 항목
    - 앱 소스 코드 저장소
    - 컨테이너 이미지를 만드는 CI 파이프라인
    - 컨테이너 이미지 저장소
    - Kubernetes manifest 저장소
    - manifest를 하나 이상의 클러스터에 동기화하고 변화를 감지하는 GitOps 엔진

GitOps

- DevOps 및 기민성
  - ✓ GitOps는 지속적 배포 및 인프라 운영을 위한 개발자 중심적 접근법이며 프로세스 자동화를 위해 Git을 사용하는 개발자 워크플로
  - ✓ DevOps가 애자일 소프트웨어 개발 프로세스를 보완한다면 GitOps는 인프라 자동화 및 애플리케이션 수명 주기 관리 측면에서 DevOps를 보완
  - ✓ GitOps는 운영 자동화를 위한 개발자 워크플로
  - ✓ 애자일 방법론의 가장 중요한 점 가운데 하나는 납기(lead time)를 줄인다는 것
  - ✓ 납기를 추상적으로 설명하자면 요구 사항을 식별하고 이를 충족하는 데까지 걸리는 시간
  - ✓ 납기를 줄이려면 IT 조직의 문화는 근본적으로 변화되어야 하는데 애플리케이션을 실시간으로 확인할 수 있으면 개발자는 그 피드백 루프를 통해 코드를 재설계하고 개선하여 프로젝트를 성공으로 이끌 수 있음
  - ✓ 기민성은 DevOps 철학과 마찬가지로 GitOps 역시 사업 프로세스에 문화로 스며들어야 함
  - ✓ 애플리케이션 배포나 인프라 변경 같은 모든 작업은 Git 워크플로를 통해서만 할 수 있어야 하는데 그러려면 개발 문화 자체가 달라져야 함
  - ✓ 전통적 환경이 GitOps 도구로 구동되는 최신 환경으로 바뀌려면 많은 단계를 거쳐야 함
  - ✓ 일부 조직은 드물게 처음부터 다시 시작할 기회를 누리기도 하지만 많은 기업은 코끼리를 우아한 발레리나로 훈련하는 어려운 숙제를 풀어야 함

Argo CD

- 개요
  - ✓ 예전에는 애플리케이션은 개발, 테스트, staging, production으로 환경을 구분해서 동작했는데 Kubernetes에서는 이렇게 환경을 분리하고자 할 때 환경 별로 별도의 Kubernetes 클러스터를 사용하던가 하나의 클러스터 안에 namespace로 구분
  - ✓ namespace를 분리하는 방식
    - namespace를 새로 만들고 애플리케이션을 구성하는데 필요한 모든 환경(ConfigMap, Secret, Ingress 등)을 추가하는 방식
    - 이 방식은 시간이 지나면서 구성 드리프트가 발생하는데 개발 환경 클러스터 namespace에만 최신 버전의 애플리케이션이 배포되거나 네트워크 정책과 같은 리소스를 수정하게 되면 나머지 다른 환경을 개발 환경과 동일하게 맞추기 위해서 다른 부분을 확인해서 수동으로 변경을 해야하는데 이 부분을 단순화하기 위해서 Helm, Kustomize 와 같은 패키지 매니저를 사용해서 애플리케이션 리소스를 재활용하고 하나의 단일 지점처럼 관리
    - Helm을 사용하면 여러 버전을 만들고 각 환경에 맞는 원하는 버전을 배포할 수 있지만 이력을 추적하기 어렵고 관리의 복잡성이 증가
    - GitOps 접근 방식을 따르면 Git Repository에 pull request 및 모든 변경인 원천 소스를 보관하고 컨트롤러를 이용해서 Git Repository의 모든 구성을 자동으로 적용하는 방식을 해서 단점을 보완

Argo CD

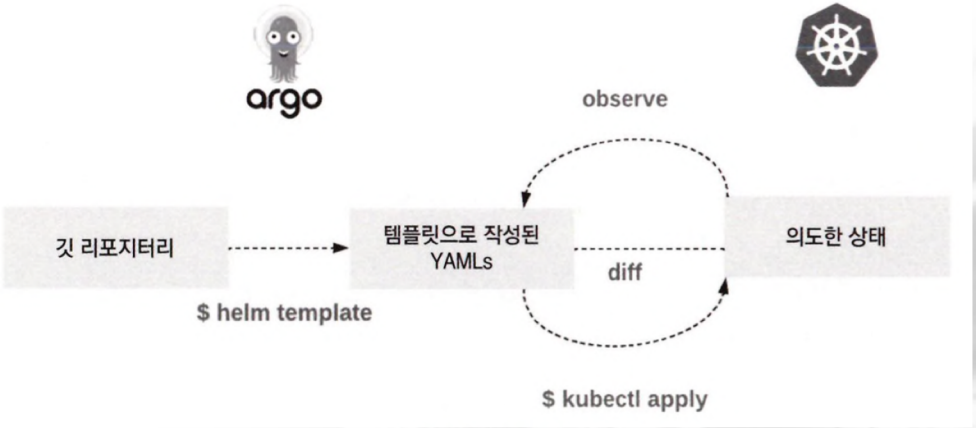
- 개요
  - ✓ 선언적인 Kubernetes의 GitOps CD 도구
  - ✓ ArgoCD 안에 있는 애플리케이션 컨트롤러가 운영 중인 애플리케이션을 지속적으로 관찰하고 현재 애플리케이션 상태 와 원천 소스인 Git Repository에 작성한 의도한 상태를 비교해서 변경 내용을 적용
  - ✓ 사용 사례
    - 배포 자동화: Git Commit 또는 CI 파이프라인이 동작하고 수동 동기화를 트리거 한 후 Argo CD 컨트롤러는 자동으로 클러스터를 Git Repository에 의도한 상태로 푸시를 하게 되는데 Argo 이벤트 자동화 프레임워크 나 수동적인 요청을 통해서 가능
    - 관찰 가능성: Argo CD는 애플리케이션 상태가 깃에서 의도한 상태와 동기화 되어 있는지 식별할 수 있는 UI 와 CLI를 제공하고 Argo CD Notification 엔진을 제공
    - 멀티 테넌시: 인증을 위한 RBAC 정책을 이용해서 여러 클러스터를 관리하고 배포 가능

Argo CD

- 도구
  - ✓ Argo CD: GitOps Tool
  - ✓ Argo Rollouts: Progressive Delivery(점진적 배포) 컨트롤러
  - ✓ Argo Events:
    - Kubernetes 네이티브 이벤트 기반 워크플로우 트리거러(Triggering Engine)
    - 특정 이벤트가 발생했을 때 → 자동으로 Argo Workflows, Argo Rollouts, Kubernetes Job/Service 등을 실행
    - GitOps, CI/CD, MLOps 파이프라인에서 이벤트 드리븐 자동화를 가능하게 함
  - ✓ Argo Workflows
    - Kubernetes 네이티브 워크플로우 엔진 & 오케스트레이션 도구
    - 복잡한 작업(Job)들을 DAG(Directed Acyclic Graph) 또는 Step 기반으로 정의하고 실행
    - 데이터 파이프라인, CI/CD 빌드 파이프라인, MLOps 워크플로우 등에서 많이 활용

Argo CD

- ❖ 핵심 개념
  - ✓ 조정
    - Git Repository에 담긴 의도한 상태를 현재 상태의 클러스터 와 일치시켜야 하며 필요한 환경에 올바르게 전달하는 것
    - Git Repository에 있는 Helm 차트를 Kubernetes yaml로 랜더링을 하고 클러스터의 의도한 상태와 비교하는데 이를 동기화 상태(sync status)라고 하는데 이 때 다르다고 판단되면 자동 혹은 수동으로 kubectl apply 사용해서 적용
    - Argo CD는 helm install을 사용하지 않고 kubectl apply 명령을 사용



Argo CD

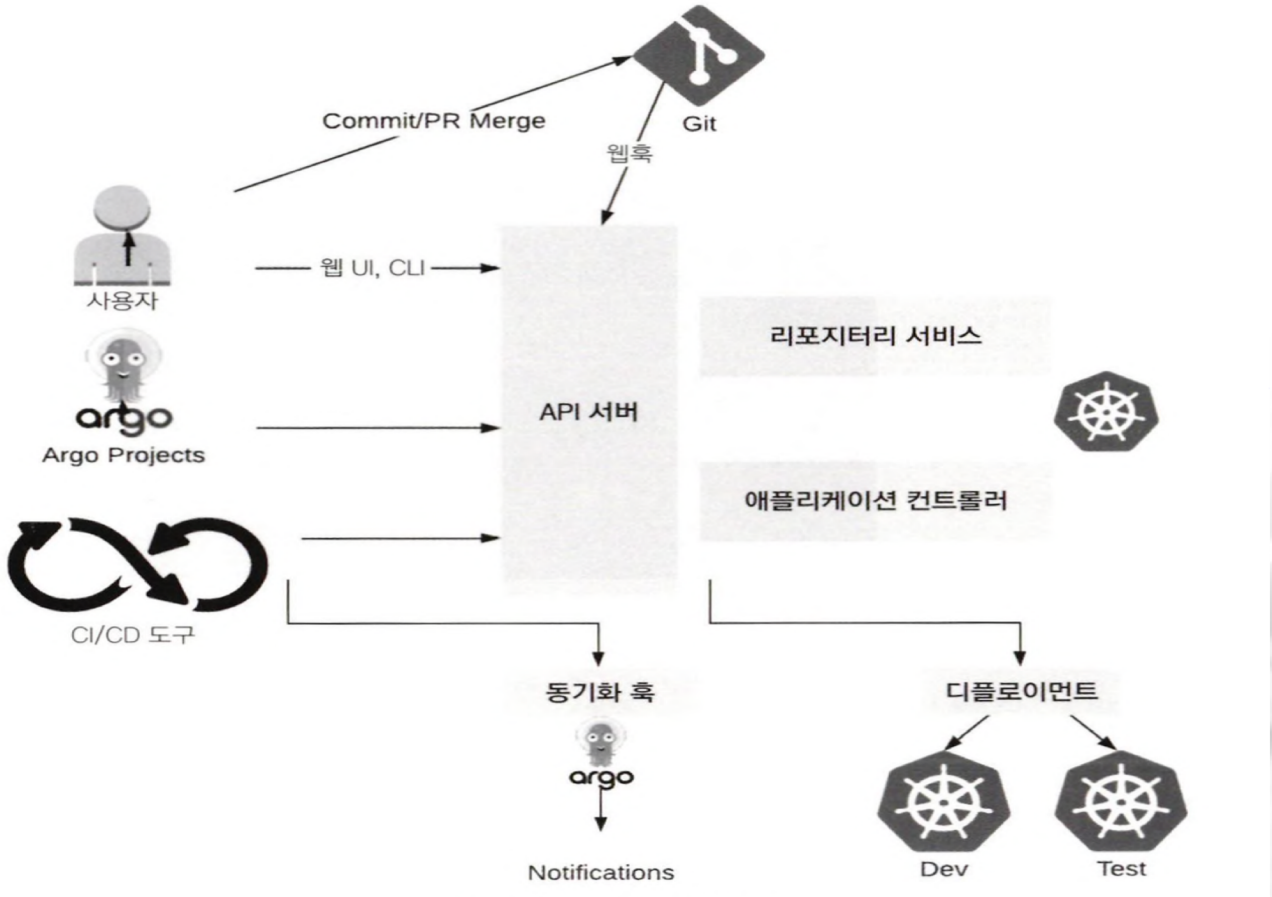
- ❖ 핵심 개념
  - ✓ 조정
    - Argo CD는 바라보는 Git Repository에 있는 Helm 차트를 Kubernetes yaml로 렌더링하고 클러스터를 의도한 상태와 비교하는데 이 상태가 동기화 상태(sync status)
    - Arg0 CD가 다르다고 판단하면 자동 혹은 수동으로 kubectl apply를 사용해 템플릿 화된 파일을 적용시키고 Kubernetes를 의도한 상태로 변경
    - Argo CD는 운영 중인 Kubernetes 리소스와 Kubernetes의 의도한 상태를 비교하면서 애플리케이션의 상태 체크를 진행
    - Argo CD는 helm install을 사용하지 않고 kubectl apply를 사용하는데 Argo CD는 여러 템플릿 도구를 지원하고 있는데 이런 도구들을 래퍼로 사용하는 것이 아니라 GitOps 원칙에 맞는 선언적 도구로 사용해 원하는 상태를 배포하는 데 목적이 있기 때문

Argo CD

- 용어
  - ✓ Application: Kubernetes 리소스 그룹
  - ✓ Application Source Type: Helm, Kustomize 등
  - ✓ Target State: 애플리케이션이 의도한 상태를 의미하며 원천 소스인 Git Repository를 의미
  - ✓ Live State: 애플리케이션의 현재 상태로 클러스터에 배포된 상태
  - ✓ Sync State: 현재 상태와 타깃 상태가 일치하는지 확인
  - ✓ Sync: Kubernetes 클러스터에 변화를 적용해 애플리케이션을 타깃 상태로 변경
  - ✓ 동기화 동작 상태: 동기화 단계에서 작업이 실패인지 성공인지 여부를 확인
  - ✓ 새로 고침: Git Repository의 최신 코드 와 현재 상태의 차이점을 비교
  - ✓ 서비스 상태: 애플리케이션이 요청을 받을 수 있고 운영 중인 상태 인지

Argo CD

❖ Architecture



Argo CD

- 핵심 구성 요소
  - ✓ API Server: 웹 UI, CLI, Argo Event, CI/CD 시스템 같은 다른 시스템과 상호 작용을 위한 역할을 수행
    - 애플리케이션 관리 및 상태 보고
    - 애플리케이션 트리거 작업
    - Git Repository 와 Kubernetes 클러스터 관리
    - 인증과 SSO 지원
    - RBAC 정책 강화
  - ✓ Repository Server
    - 애플리케이션 manifest를 보관하는 git repository 로컬 캐시를 유지
    - 요청에 필요한 매개변수
    - Repository URL
      - Git Version
      - 애플리케이션 경로
      - 템플릿 세부 설정: 매개변수, ksonnet(초창기 설정 관리 도구)의 environment, Helm의 values.yaml

Argo CD

- 핵심 구성 요소
  - ✓ Application Controller
    - 애플리케이션의 상태를 지속적으로 확인하고 의도한 상태와 비교
    - 사용자가 생성한 훅을 생명 주기 동안 실행
  - ✓ Argo 웹 UI, CLI와 CI/CD 도구들이 Argo CD의 API 서버와 직접 소통
    - Argo CD는 Repository를 확인해 원하는 상태를 불러오고 기본적으로 3분마다 Git Repository를 확인하기 때문에 즉시 반영되지 않으므로 지연을 피하기 위해서는 동기화 단계를 즉시 트리거 할 수 있는 몇 가지 다른 옵션을 제공
    - Argo CD에는 동기화 단계를 수동으로 시작할 수 있는 UI를 포함
    - Argo CD에는 API 서버와 상호 작용하기 위해 사용하는 CLI도 제공

Argo CD

- helm을 이용한 Argo CD설치
  - ✓ Helm Repository 추가: helm repo add argo https://argoproj.github.io/argo-helm
  - ✓ Repository 업데이트: helm repo update
  - ✓ namespace 생성: kubectl create namespace argocd
  - ✓ 설치:
    - helm install argocd argo/argo-cd -n argocd
    - helm install argocd argo/argo-cd -n argocd --version <차트버전>
  - ✓ 설치 확인: kubectl get all -n argocd
  - ✓ 초기 비밀번호 확인: kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 -d

Argo CD

- Argo CD
  - ✓ Web UI 접속
    - 서비스 확인: kubectl get svc -n argocd

```
NAME                                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
argocd-applicationset-controller        ClusterIP   10.96.193.45    <none>
7000/TCP                    40h
argocd-dex-server                       ClusterIP   10.111.86.236   <none>
5556/TCP,5557/TCP           40h
argocd-redis                            ClusterIP   10.96.168.209   <none>        6379/TCP
40h
argocd-repo-server                      ClusterIP   10.98.194.122   <none>
8081/TCP                    40h
argocd-server                           ClusterIP   10.109.102.33   <none>
80/TCP,443/TCP    40h
```

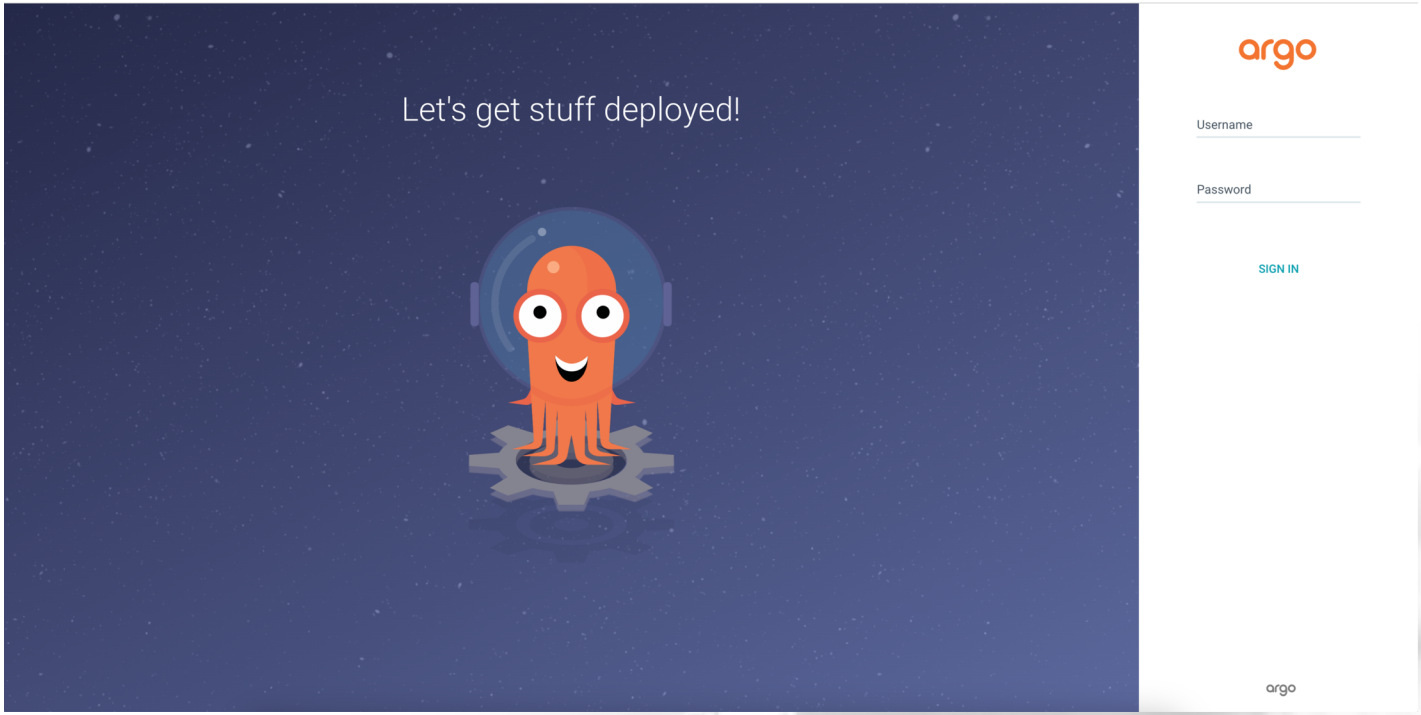
Argo CD

- Argo CD
  - ✓ Web UI 접속
    - 서비스 타입 변경: kubectl edit svc argocd-server -n argocd

```
  - name: http
    port: 80
    protocol: TCP
    targetPort: 8080
    nodePort: 30080
  - name: https
    port: 443
    protocol: TCP
    targetPort: 8080
    nodePort: 30443
  selector:
    app.kubernetes.io/instance: argocd
    app.kubernetes.io/name: argocd-server
  sessionAffinity: None
  type: NodePort
```

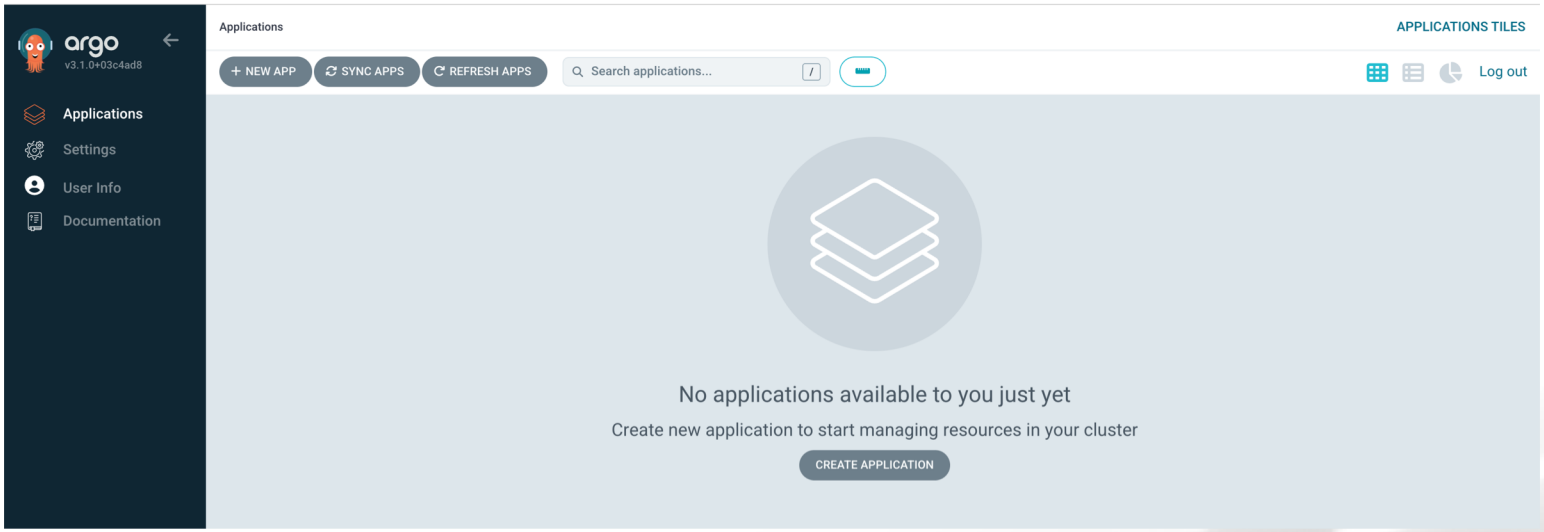
Argo CD

- Argo CD
  - ✓ Web UI 접속
    - 접속: 기본 아이디는 admin



Argo CD

- Argo CD
  - ✓ Web UI 접속
    - 접속: 기본 아이디는 admin



Argo CD

- ❖ Argo CD
  - ✓ Application 배포
    - ● 리소스 파일 생성 – argocd/application.yaml

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: nginx-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://charts.bitnami.com/bitnami
    chart: nginx
    targetRevision: 22.6.5
    helm:
      parameters:
        - name: "service.type"
          value: "NodePort" # VirtualBox 환경 접속을 위해 NodePort 설정
        - name: "replicaCount"
          value: "2"
```

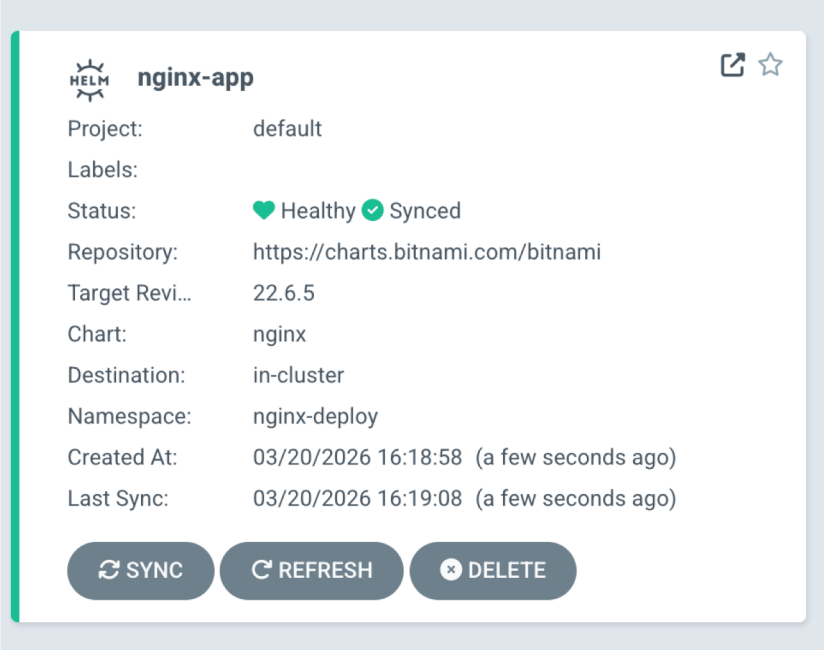
Argo CD

- Argo CD
  - ✓ Application 배포
    - 리소스 파일 생성 – argocd/application.yaml

```
destination:
  server: https://kubernetes.default.svc
  namespace: nginx-deploy
syncPolicy:
  automated:
    prune: true
    selfHeal: true
  syncOptions:
    - CreateNamespace=true
```

Argo CD

- Argo CD
  - ✓ Application 배포
    - manifest 적용: kubectl apply -f application.yaml
    - 확인: 웹 브라우저에서 확인



Argo CD

- Argo CD
  - ✓ Application 배포
    - namespace 확인: kubectl get ns

```
NAME              STATUS   AGE
argocd            Active   4h24m
default           Active   24h
kube-flannel      Active   24h
kube-node-lease   Active   24h
kube-public       Active   24h
kube-system       Active   24h
nginx-deploy      Active   6h15m
```

    - pod 확인: kubectl get pods -n nginx-deploy

```
NAME                         READY   STATUS    RESTARTS   AGE
nginx-app-5c8754d48f-kjbbc   1/1     Running   0          116s
nginx-app-5c8754d48f-mjjh9   1/1     Running   0          105s
```

Argo CD

- ❖ Argo CD
  - ✓ Argo CD CLI를 활용한 애플리케이션 배포
    - ● Argo CD CLI 설치
      - ◆ 다운로드
        - ▪ curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
        - ▪ Mac M 시리즈: curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-arm64
      - ◆ 실행 권한 부여: chmod +x argocd-linux-amd64
      - ◆ 파일 이동: sudo mv argocd-linux-amd64 /usr/local/bin/argocd
      - ◆ 버전 확인: argocd version --client

```
argocd: v3.3.4+34ccdfc
  BuildDate: 2026-03-16T11:24:58Z
  GitCommit: 34ccdfc3d5235b0184eb910b8ba4edcd81ef8f03
  GitTreeState: clean
  GoVersion: go1.25.5
  Compiler: gc
  Platform: linux/arm64
```

Argo CD

- ❖ Argo CD
  - ✓ Argo CD CLI를 활용한 애플리케이션 배포
    - ● Argo CD CLI를 이용한 배포
      - ◆ 로그인: argocd login localhost:30443(자신이 설정한 노드포트)
      - ◆ 이전 애플리케이션 삭제: argocd app delete nginx
      - ◆ 배포

```
argocd app create nginx ₩
  --repo https://charts.bitnami.com/bitnami ₩
  --helm-chart nginx ₩
  --revision 22.6.5 ₩
  --dest-server https://kubernetes.default.svc ₩
  --dest-namespace nginx
```

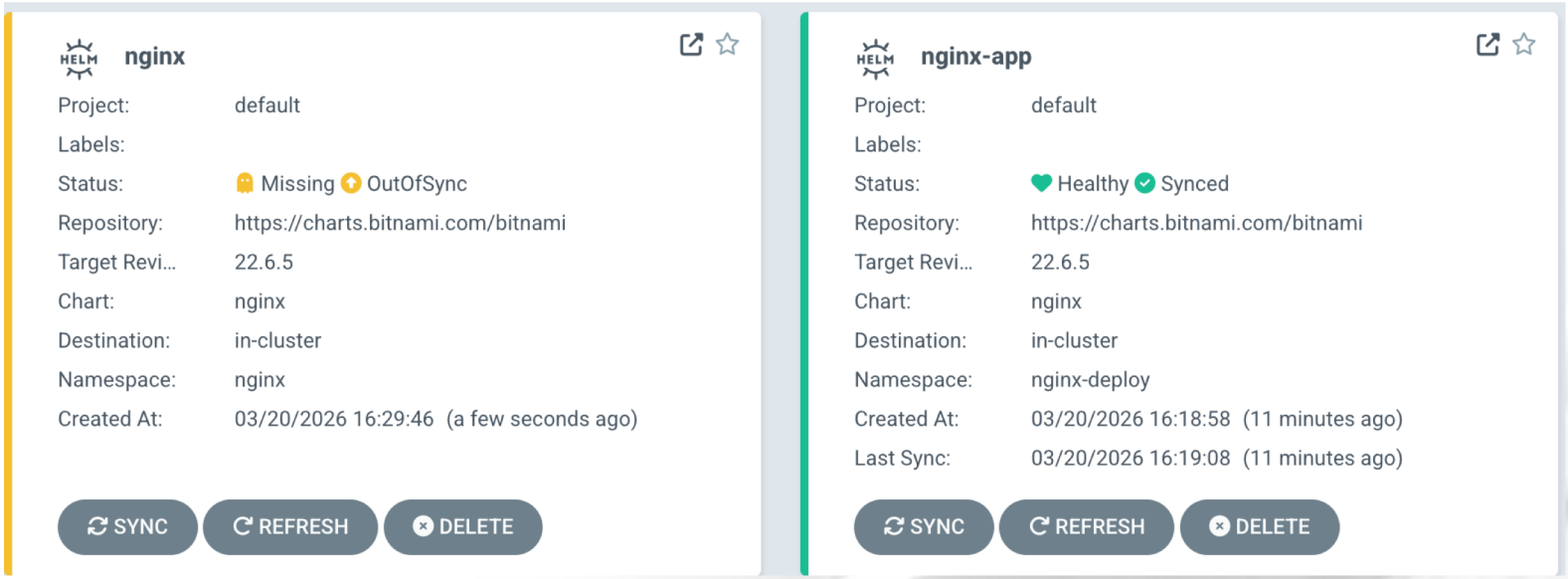
Argo CD

- Argo CD
  - ✓ Argo CD CLI를 활용한 애플리케이션 배포
    - Argo CD CLI를 이용한 배포
      - ◆ 동기화: argocd app sync nginx

```
TIMESTAMP                  GROUP        KIND   NAMESPACE
NAME    STATUS    HEALTH        HOOK  MESSAGE
2025-08-25T05:41:01+00:00            Service        nginx                  nginx
OutOfSync
2025-08-25T05:41:01+00:00   apps  Deployment        nginx
nginx   OutOfSync
2025-08-25T05:41:01+00:00            Service        nginx                  nginx
OutOfSync                  service/nginx created
2025-08-25T05:41:01+00:00   apps  Deployment        nginx
nginx   OutOfSync                  deployment.apps/nginx created
2025-08-25T05:41:01+00:00            Service        nginx                  nginx
Synced  Progressing          service/nginx created
2025-08-25T05:41:01+00:00   apps  Deployment        nginx
nginx    Synced  Progressing          deployment.apps/nginx created
```

Argo CD

- Argo CD
  - ✓ Argo CD CLI를 활용한 애플리케이션 배포
    - Argo CD CLI를 이용한 배포
      - 결과



Argo CD

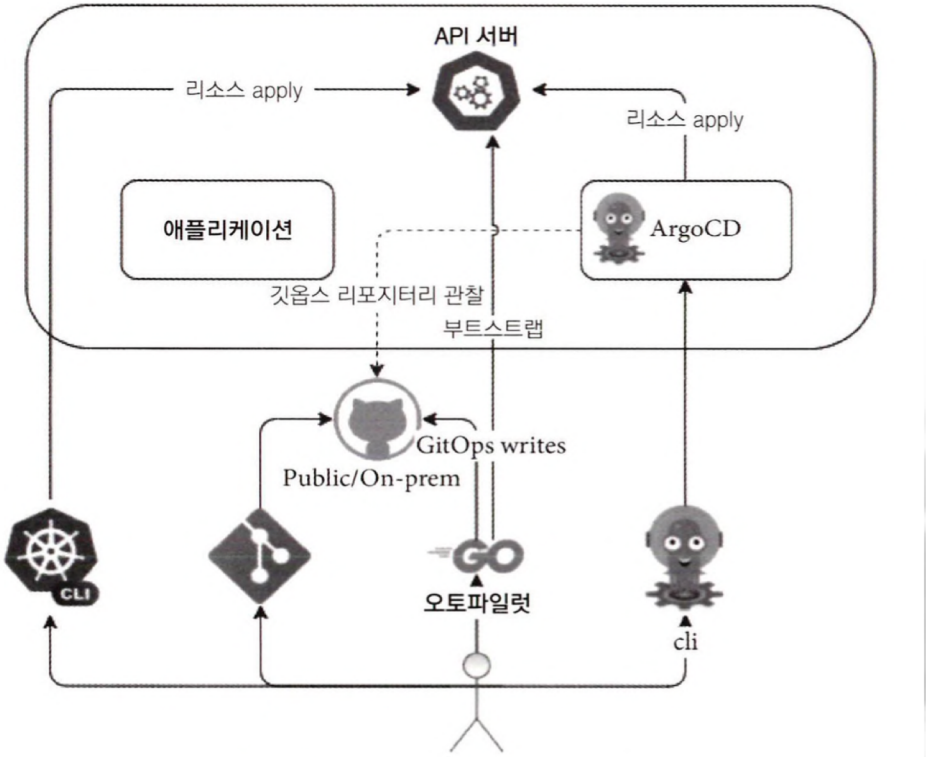
- Argo CD
  - ✓ Argo CD Autopilot
    - 개요
      - Argo CD를 GitOps 방식으로 설치/운영하기 쉽게 만들어주는 툴
      - 일반적으로 Argo CD를 수동으로 설치하고 Application을 하나 하나 정의하지만 Autopilot은 Argo CD 자체 설치 + 애플리케이션 관리 구조를 Git Repository 중심으로 자동화 함
    - 기능
      - GitOps를 사용해 부트스트랩 Argo CD 애플리케이션을 생성하고 관리할 수 있음
      - GitHub Repository를 짜여진 구조로 세팅해 새로운 서비스를 추가하고 Argo CD의 수명 주기에 적용
      - 각각 다른 환경에서 애플리케이션을 업데이트하고 승격(promote)할 수 있게 함
      - 재해 복구를 대비할 수 있고 필요한 모든 유틸리티와 애플리케이션에 대한 장애 조치(failover) 클러스터를 부트스트랩
      - Argo CD 애플리케이션에 시크릿에 대한 암호화도 지원

Argo CD

- Argo CD
  - ✓ Argo CD Autopilot
    - Architecture
      - 부트스트랩 단계에서 Argo CD를 배포할 때 오토파일럿은 Kubernetes 클러스터와 통신하고 그 후에는 더 이상 Kubernetes 클러스터에 접근할 일이 없음
      - 접근이 필요한 건 GitOps Repository 뿐
      - 새로운 Argo CRD를 Kubernetes 클러스터에 추가할 때 오토파일럿은 Argo CD 서버에 접근함

Argo CD

- Argo CD
  - ✓ Argo CD Autopilot
    - Architecture



Argo CD

- Argo CD
  - ✓ Argo CD Autopilot
    - 설치: https://argocd-autopilot.readthedocs.io/en/stable/Installation-Guide/
      - ◆ 최신 버전 값을 환경 변수에 저장

        VERSION=$(curl --silent "https://api.github.com/repos/argoproj-labs/argocd-autopilot/releases/latest" | grep '"tag_name"' | sed -E 's/.*"([^"]+)".*/₩1/ ')
      - 바이너리 다운로드

        curl -L --output - https://github.com/argoproj-labs/argocd-autopilot/releases/download/"$VERSION"/argocd-autopilot-linux-amd64.tar.gz | tar zx
      - Path로 이동

        mv ./argocd-autopilot-* /usr/local/bin/argocd-autopilot
      - 버전 확인

        argocd-autopilot version

Argo CD

- Argo CD
  - ✓ Argo CD Autopilot
    - Git Hub Repository 등록
      - Git Hub Access Token 발급
      - Git Hub Repository 생성
      - export GIT_TOKEN=Token값
      - export GIT_REPO=git 주소

Argo CD

- Argo CD
  - ✓ Argo CD Autopilot
    - Argo-CD 설치
      - argocd-autopilot repo bootstrap
      - 서비스 수정
      - 프로젝트 생성: argocd-autopilot project create testing
      - 앱 생성: argocd-autopilot app create hello-world --app github.com/argoproj-labs/argocd-autopilot/examples/demo-app/ -p testing --wait-timeout 2m

Argo CD

- ❖ Argo CD
  - ✓ Argo CD Autopilot
    - ● nginx 배포
      - ◆ 디렉토리 구조

```
gitops-repo/
├── projects/
│   └── default/                    # 기본 프로젝트 (autopilot init 시 생성됨)
└── apps/
    └── nginx/                      # nginx 앱 배포용 디렉토리
        ├── kustomization.yaml      # Helm 차트를 포함하는 Kustomize 매니페스트
        └── values.yaml             # Nginx Helm 차트 설정값 (선택)
```

Argo CD

- Argo CD
  - ✓ Argo CD Autopilot
    - nginx 배포
      - kustomization.yaml

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

helmCharts:
  - name: nginx
    repo: https://charts.bitnami.com/bitnami
    version: 13.2.10
    releaseName: nginx
    namespace: nginx
    valuesFile: values.yaml
```

Argo CD

- ❖ Argo CD
  - ✓ Argo CD Autopilot
    - ● nginx 배포
      - ◆ values.yaml

```
replicaCount: 2
service:
  type: NodePort
  nodePorts:
    http: 30080
    https: 30443
resources:
  limits:
    cpu: 200m
    memory: 256Mi
  requests:
    cpu: 100m
    memory: 128Mi
```

Argo CD

- Argo CD
  - ✓ Argo CD Autopilot
    - nginx 배포
      - ◆ 배포 명령

argocd-autopilot app create nginx --app github.com/itstudy001/autopilot_nginx/apps/nginx?ref=main --project testing --repo https://github.com/itstudy001/autopilot_nginx.git --git-token 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