k8s를 이용한 GitOps 기반 CI/CD 자동화

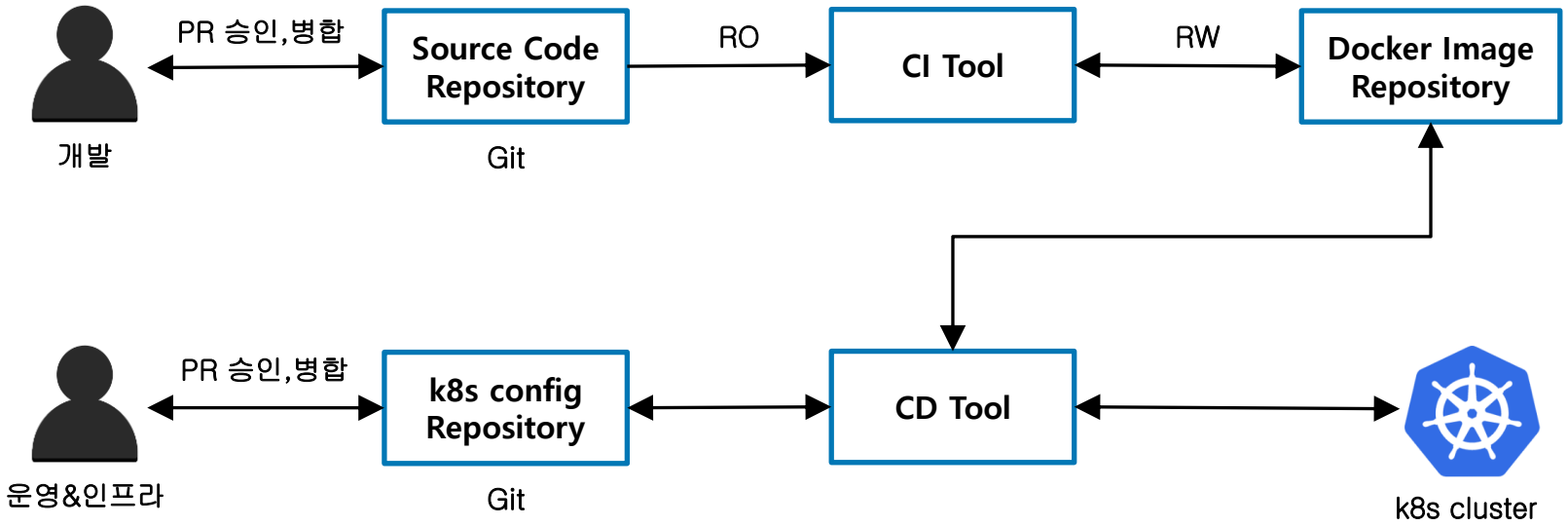
3. GitOps 개요

1. GitOps란?

❖GitOps

- Weaveworks Inc. 에서 처음 사용한 용어로써 Devops를 적용하는 방법 중 하나
- Cloud Native 애플리케이션을 대상으로 지속적 배포에 집중

❖GitOps 아키텍처



1. GitOps란?

❖ GitOps 원칙

- Git가 단일 정보 소스임(Single Source Of Truth, SSOT)
  - Git에 저장된 k8s 설정(manifest)을 기준으로 시스템에 배포됨
  - 이전버전으로 되돌리려면? --〉 git revert !!
- 승인된 변경 사항은 자동으로 시스템에 적용함
  - k8s 설정, 이미지에 변경이 발생하면 자동으로 시스템에 적용
  - 자동 복구 및 자동화된 배포를 촉진하는 도구 사용(예: fluxcd, argocd)
- 명령형이 아닌 선언적인 방법으로 정의되어야 함
  - 명령 집합(X), 원하는 상태의 집합(O)
  - Desired State(k8s 설정 리포) <----> 동기화 <-----> Current State(시스템)
  - Push가 아닌 Pull 방식
- 자동화된 배포 실패시 사용자에게 경고할 수 있어야 함

1. GitOps란?

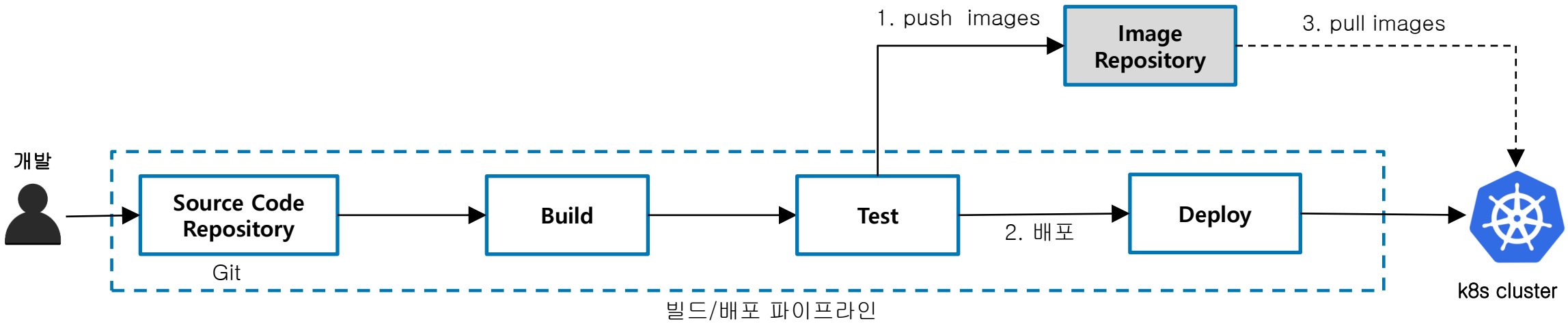
❖GitOps의 장점

- Git를 이용해 전체 변경 내역을 추적할 수 있음 --〉 단일 정보 소스
  - 롤백, 롤아웃도 Git을 통해서 관리됨
  - 안정성, 신뢰성
- Git는 개발자, 운영자라면 누구나 알고 있음
  - 생산성, 편의성
  - Application Code, Infra 모두를 Git으로 제어할 수 있음
    - Git과 k8s 클러스터, 리소스, 애플리케이션은 동기화됨
- GitOps는 CI와 CD를 분리하여 관리함
  - 보안성
  - 보안 자격증명을 클러스터 외부로 공개하지 않고 배포할 수 있음
    - 보안 자격증명은 클러스터 내부에 Secret으로 관리

2. Push VS Pull

❖ Push 방식

- Git 저장소의 매니페스트가 변경되었을 때 파이프라인을 트리거하는 구조
  - 간단한 아키텍처
  - 배포 환경 구성이 비교적 단순하고 배포 환경의 갯수에 영향을 받지 않음
- 대표적인 도구 : Jenkins, CircleCI
- 단점
  - 배포 환경과의 연결 상태에 따라서는 배포가 실패할 수 있음
  - 배포 환경이 손상되었을 때 배포 파이프라인을 다시 트리거할 수 있는 별도의 모니터링 시스템이 필요함



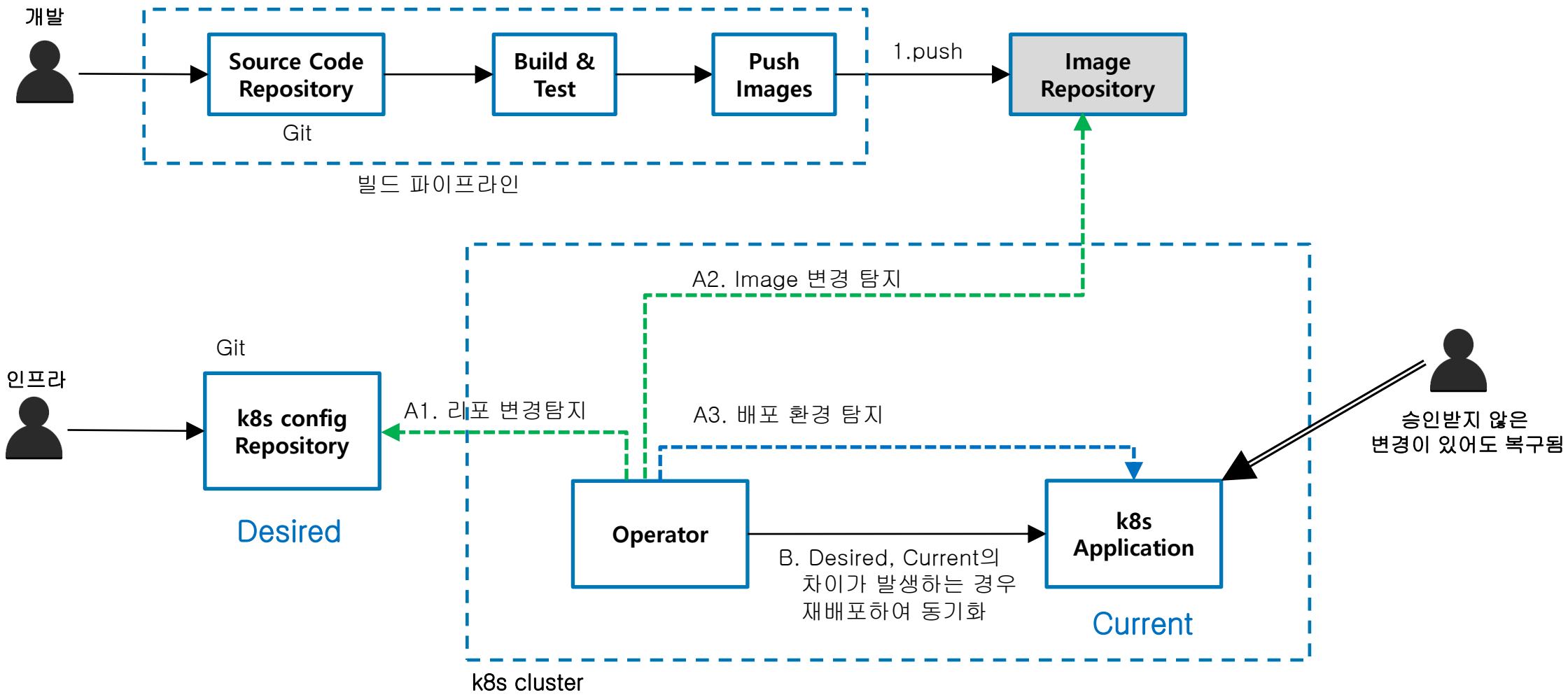
2. Push VS Pull

❖Pull 방식

- 배포 대상 환경의 별도의 Operator가 배포 파이프 라인을 대신함
  - 빌드 파이프라인은 Push 방식과 동일함
- 작동
  - k8s config 리포지토리의 매니페스트와 배포환경을 지속적으로 비교
  - 비교 도중 차이(Diff)가 발생하면 매니페스트를 기준으로 배포 환경을 동기화함
- 장점
  - 승인받지 않은 배포 환경의 변경이 발생하더라도 동기화 과정을 거쳐서 배포 환경이 재배포됨
  - 배포환경이 리포지토리를 접근하는 방식이므로 배포 환경의 자격증명이 외부로 노출되지 않음
- 대표적인 도구
  - ArgoCD, FluxCD

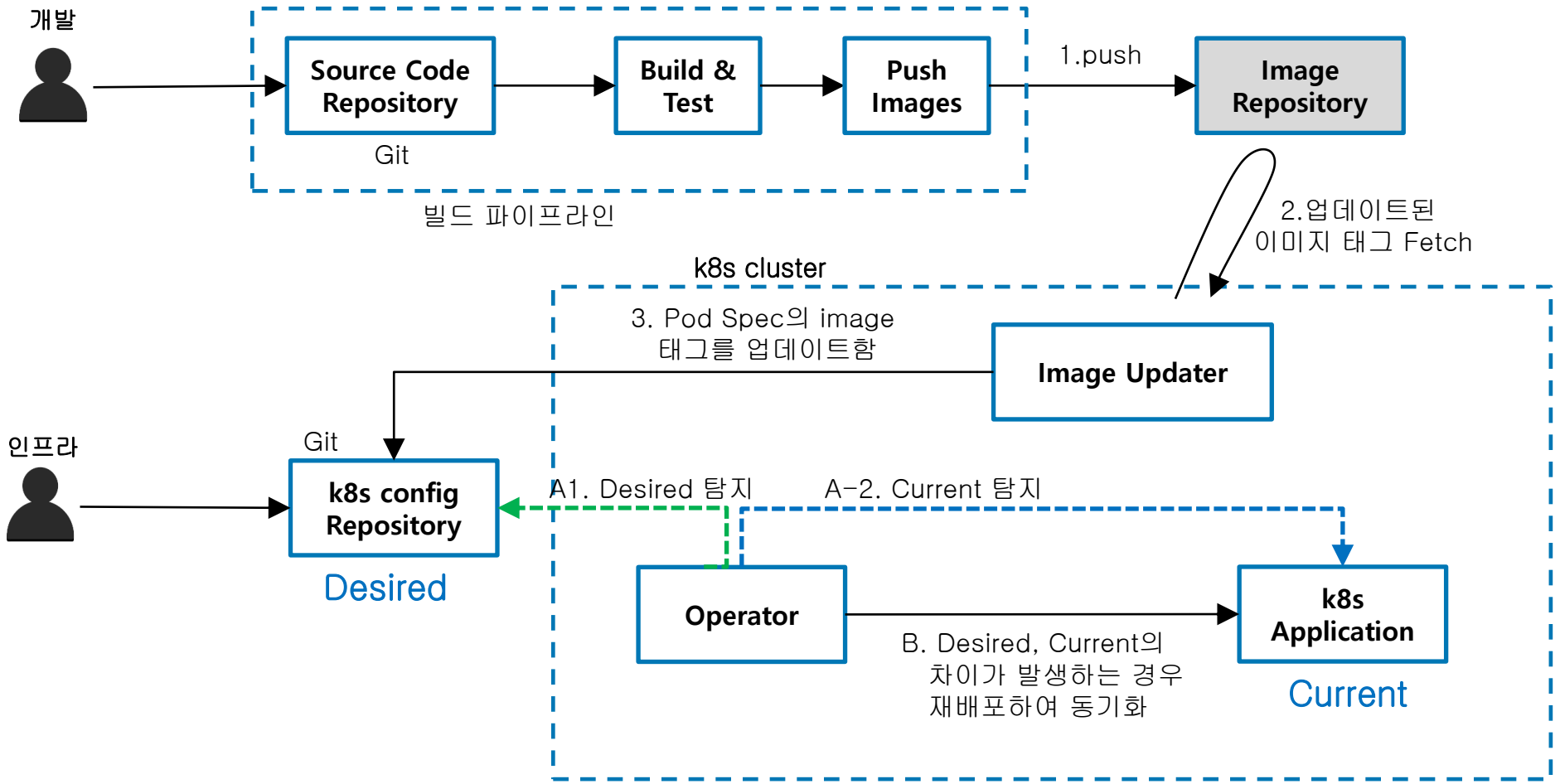
2. Push VS Pull

❖Pull 방식 아키텍처1



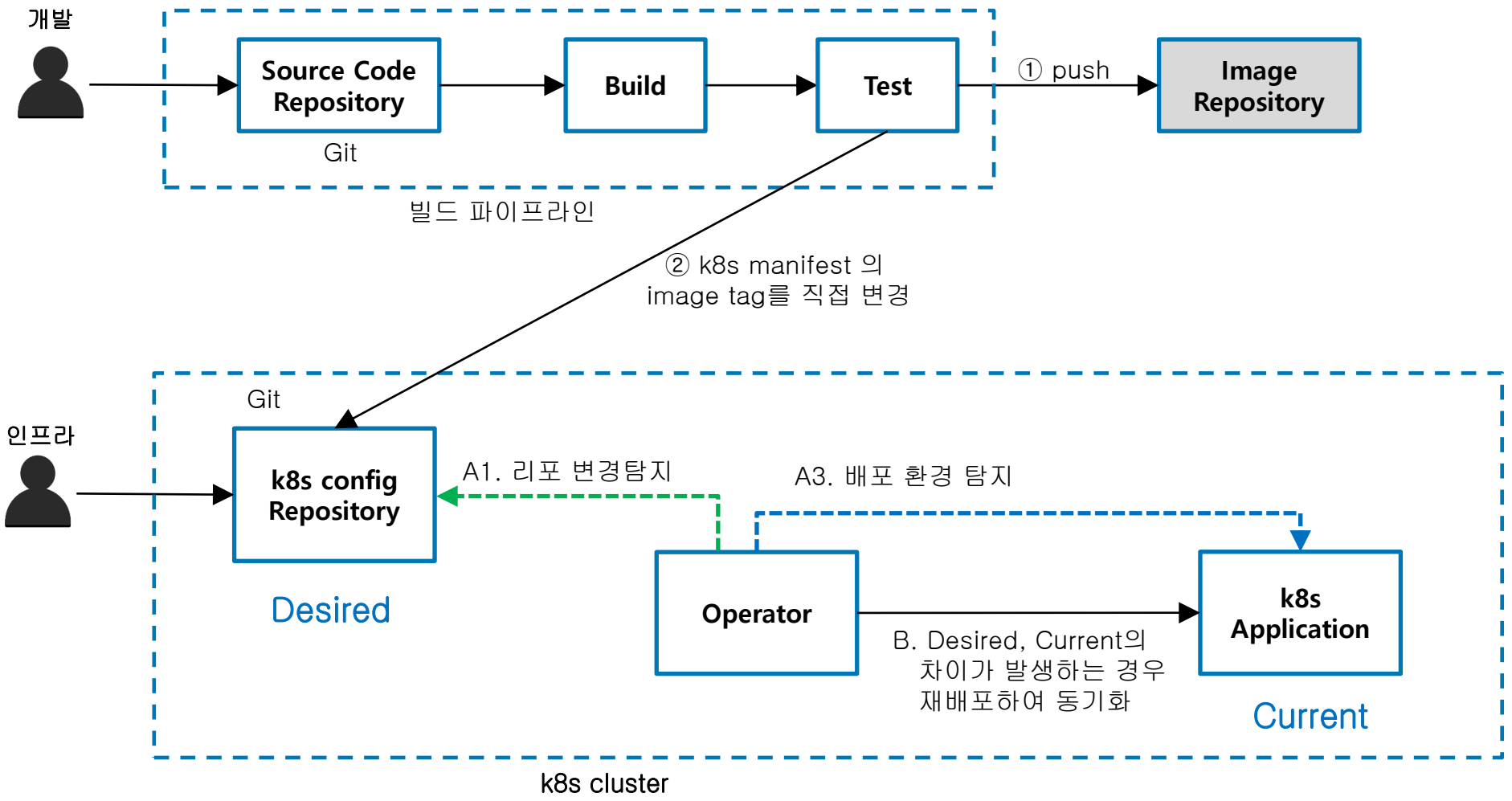
2. Push VS Pull

❖Pull 방식 아키텍처2



2. Push VS Pull

❖Pull 방식 아키텍처3



3. Pull 방식 시나리오

❖시나리오 1

- 인프라 관리자가 k8s config를 변경함
  - 예) Deployment의 spec.replicas 값을 변경

```
apiVersion: apps/v1
kind: Deployment
……
spec:
  replicas: 3
  ……
```

- Operator가 Desired vs Current 차이를 탐지
  - 애플리케이션을 재배포

3. Pull 방식 시나리오

❖시나리오 2

- 애플리케이션 코드가 변경
- CI 단계 진행
  - Application Code를 테스트, 빌드하여 Docker Image 생성(최신 버전 태깅)
  - 빌드된 이미지를 Image 리포지토리에 Push
- CD 도구의 Image Updater가 작동
  - Image Updater가 Image 리포지토리에서 최신 버전의 이미지 태그를 Fetching
  - k8s config 리포지토리의 Pod Spec 상의 Image 태그를 변경
    - image: stepanowon/nodeapp:1.0.0 ---〉 image: stepanowon/nodeapp:2.0.0
  - 단일 정보 소스인 k8s config 가 변경된 상태
- Operator가 Desired vs Current의 차이를 탐지한 후 애플리케이션 재배포

3. Pull 방식 시나리오

❖시나리오 3

- Git의 변경에 의하지 않은 변경
  - 승인받지 않은 사용자의 k8s 클러스터에 대한 직접적인 변경
  - 관리자의 부주의로 인한 애플리케이션 직접 변경, 삭제
- Operator가 Desired vs Current의 차이를 탐지한 후 애플리케이션 재배포
  - 손상된 애플리케이션을 복구

3. GitOps 도구

❖Argo CD
- 가장 대표적인 GitOps 배포 도구
- GUI 지원 : 수동 Sync, 모니터링 가능

❖Flux CD
- GitOps를 처음으로 제안한 weaveworks 사에서 만든 도구
- 경량이지만 Argo CD에 비해서 불편
- GUI를 사용하려면 별도의 도구를 설치해야 함
  - Flux UI : https://fluxcd.io/blog/2023/04/how-to-use-weave-gitops-as-your-flux-ui/

❖JenkinsX
- 다른 도구와는 달리 CI/CD 전체에 대한 파이프라인 구축 가능
  - Argo CD, Flux CD는 CD만 지원함
- 아키텍처가 복잡함

5. GitOps 학습내용

❖Pull 방식 아키텍처 2

❖단계

- Source 단계
  - 버전 제어, 형상 관리
  - Github, 관리형 Git 서비스
- CI 단계
  - 도커 이미지 빌드 후 이미지 리포지토리에 Push
  - Jenkins, AWS Code Pipeline
- CD 단계
  - k8s config, CD Tool을 이용한 배포
  - Flux CD, Argo CD