CI Pipeline을 Vault로 "더" 안전하게

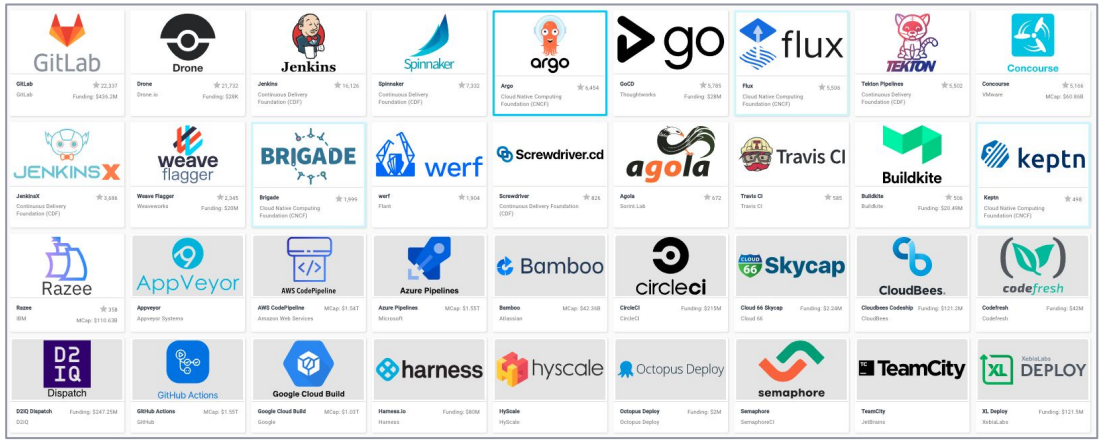
박준상

Senior Solutions Engineer at HashiCorp

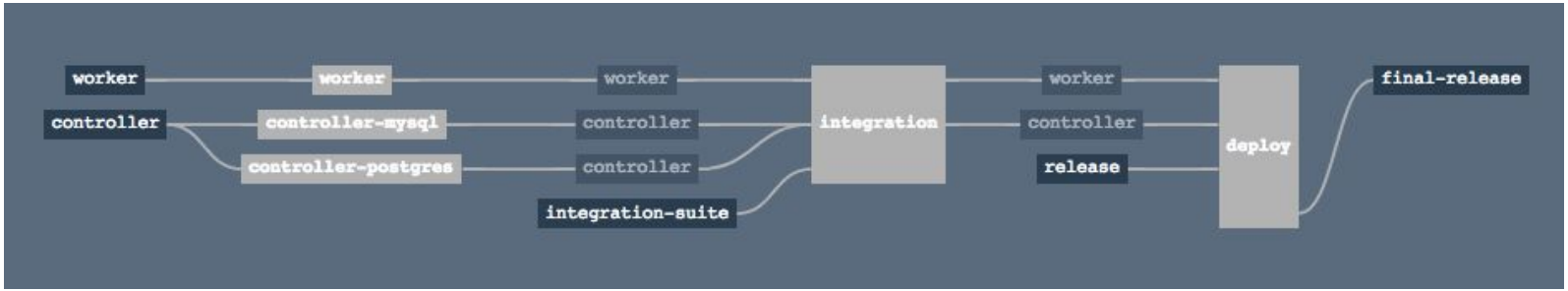
jsp@hashicorp.com

요즘 파이프 라인은...

- 애플리케이션 지속 배포
- 설치 자동화
- 인프라 구성 변경
- Day 2 operations



Refer: https://landscape.cncf.io/

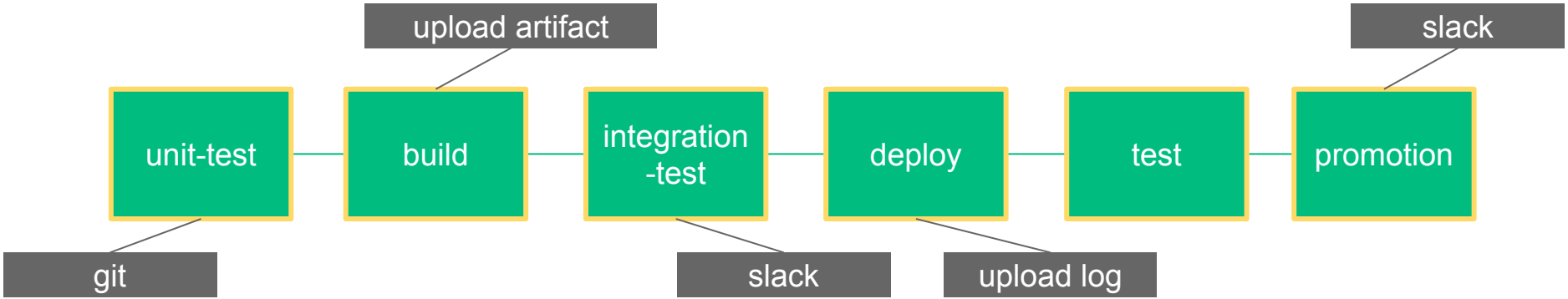


Refer: https://concoursetutorial-ja.site.lkj.io/

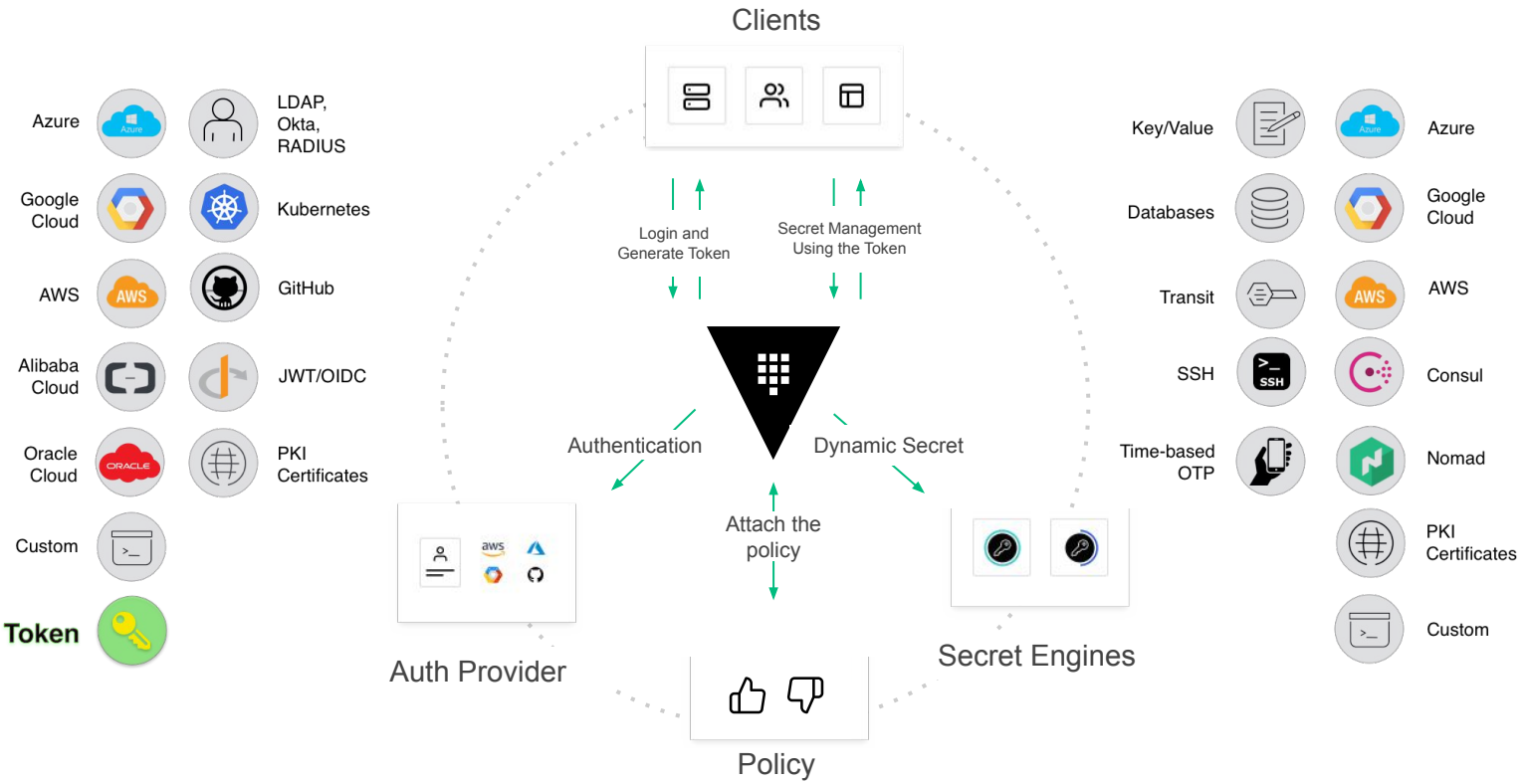
각종 시크릿을 안전하게 관리할 방법이 필요!

*** 시크릿 : 시스템 접속 시 인증및 인가를 위해 사용되는 모든 것**

- 파이프 라인 상의 시크릿 ...
  - 클라우드 접속 정보
  - Git 과 Slack 토큰
  - TLS 인증서 또는 SSH 계정 및 비밀번호 등



HashiCorp Vault, 시크릿을 위한 전자 금고



Vault Secret Engines : 시크릿 저장소

- **Identities**
  - 클라우드 서비스 접속을 위한 키
  - Active Directory / Openldap
- **각종 시스템 접속 정보**
  - Database and other middleware
  - 인증서 (TLS / PKI)
  - SSH Singed CA & OTP
- **기존 시크릿 저장을 위한 Key Value Store**
- **Vault 외부에 있는 주요 시크릿들**
  - 신용 카드 정보, 여권 번호, 주민 번호 등

Dynamic Secrets

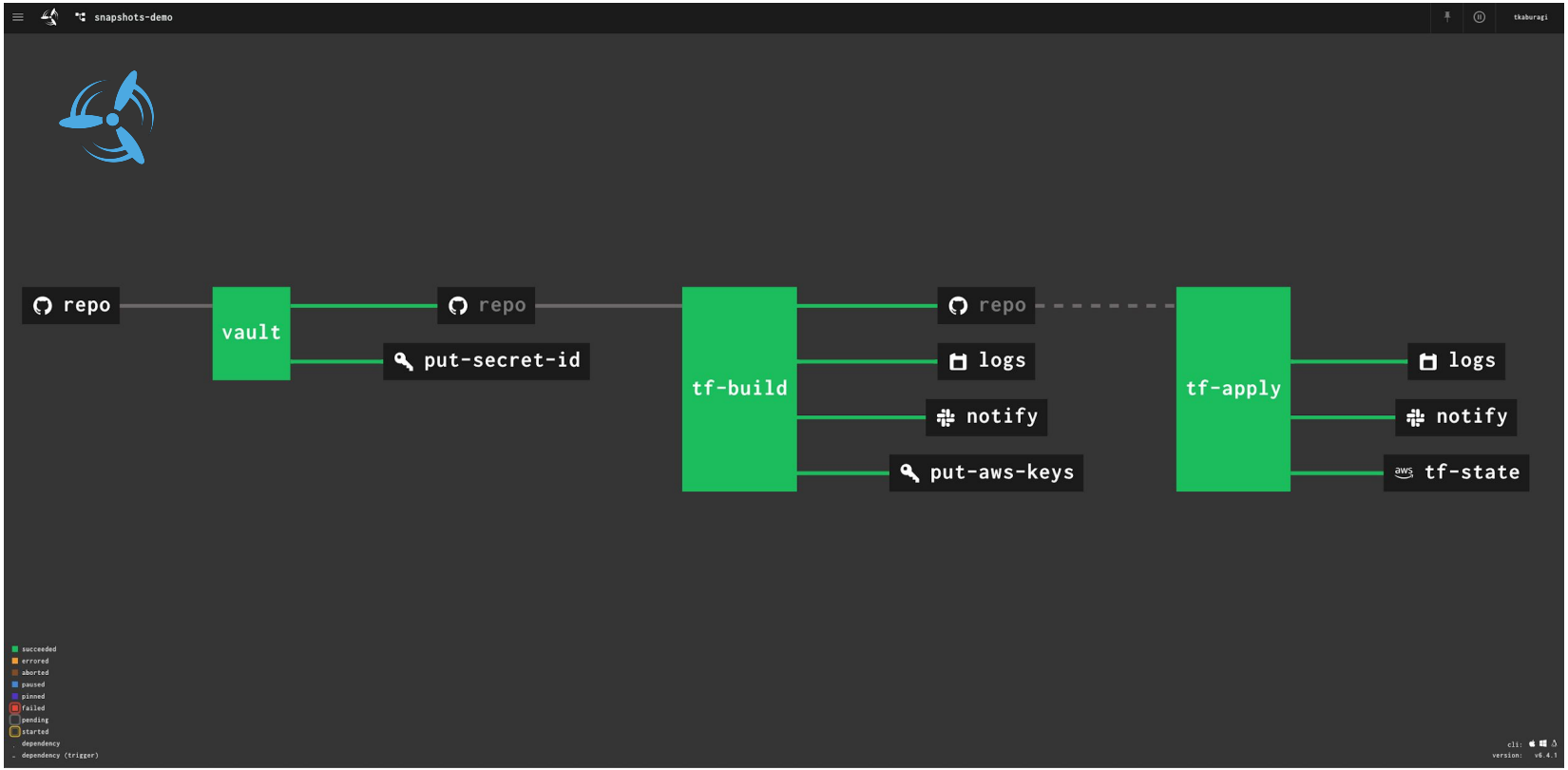
Static Secrets

Encryption

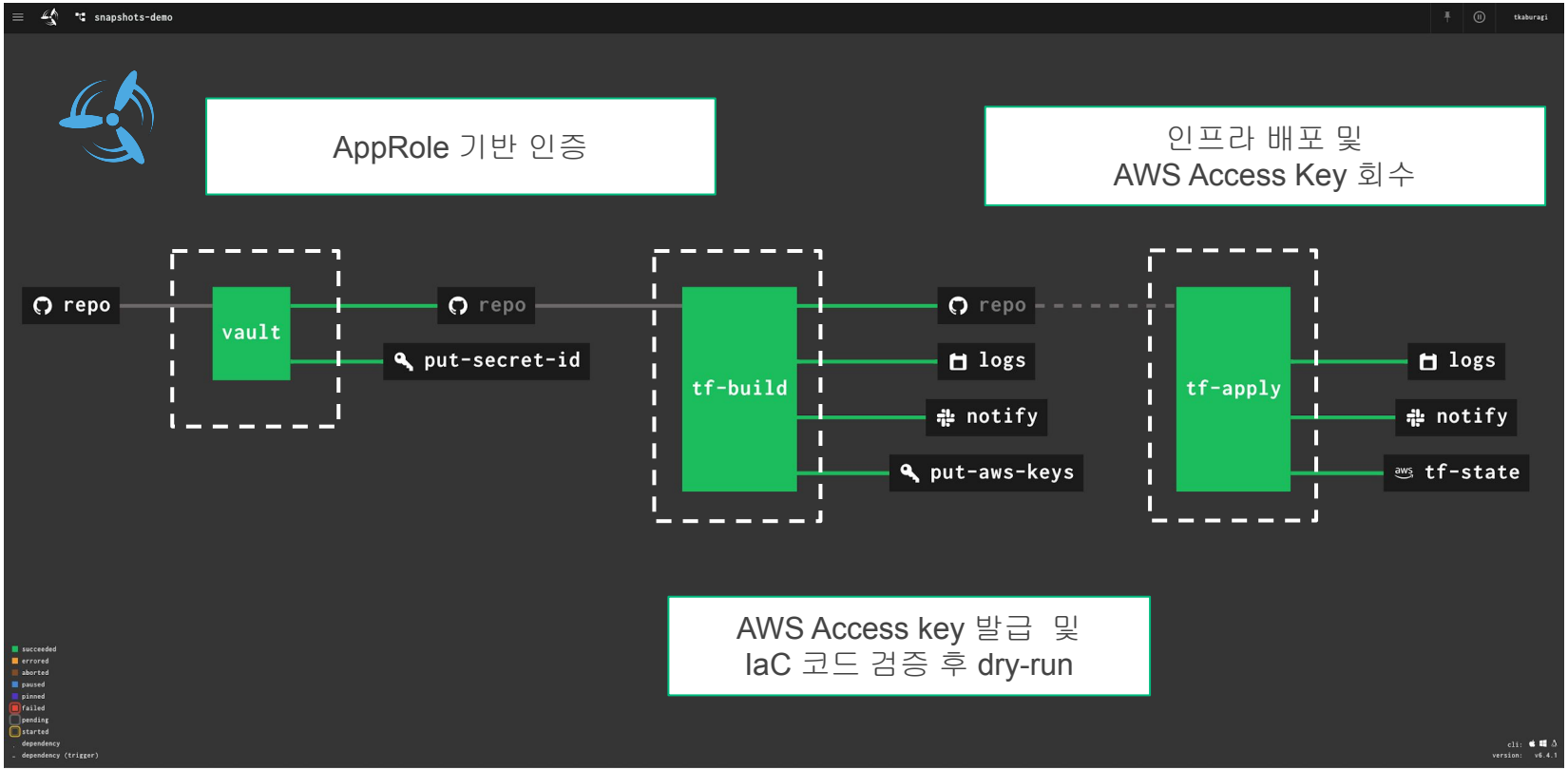
Demo Scenario

https://github.com/jsp-hashicorp/vault-secure-ci-pipeline

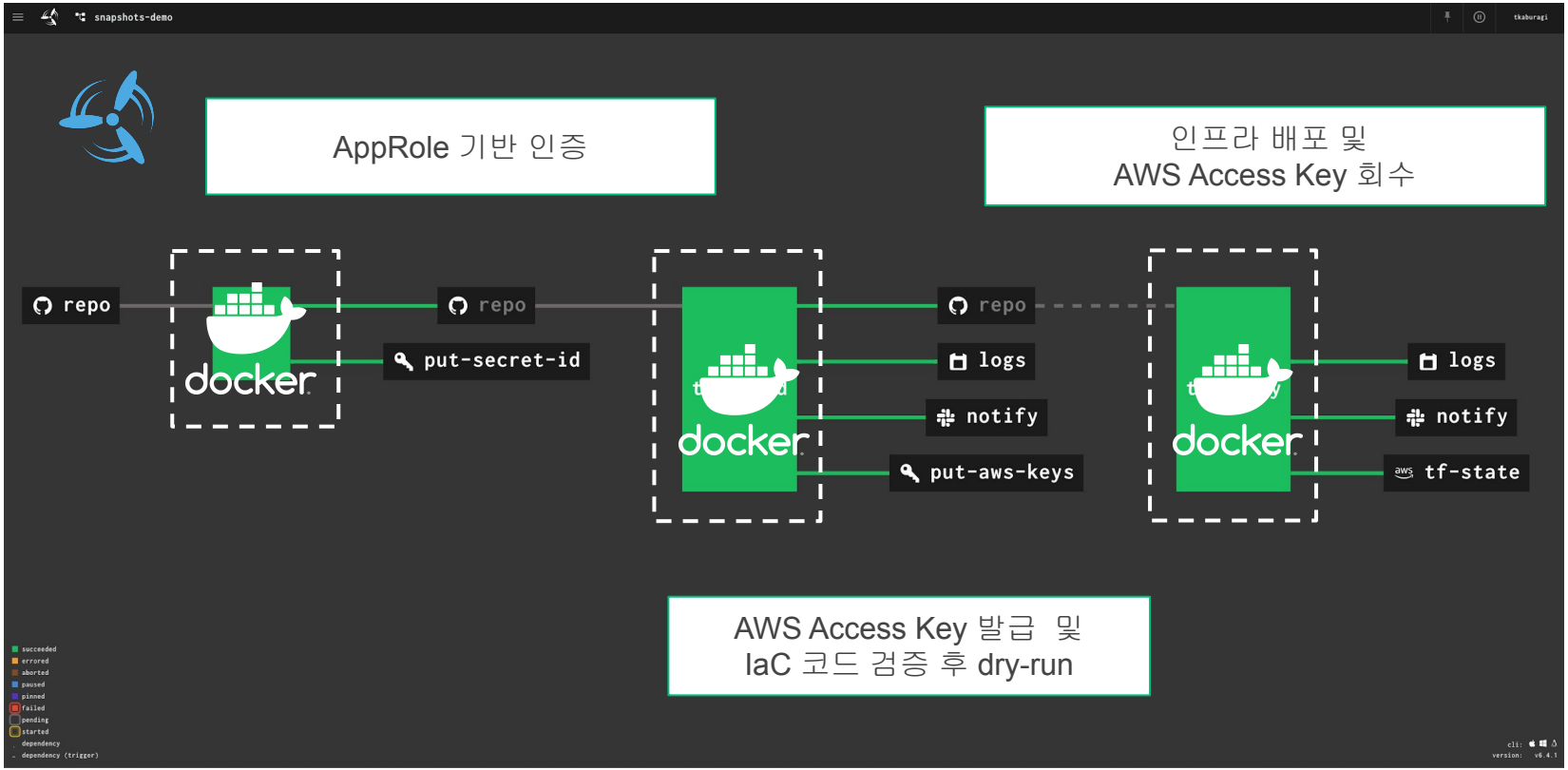
The demo pipeline:



The demo pipeline:

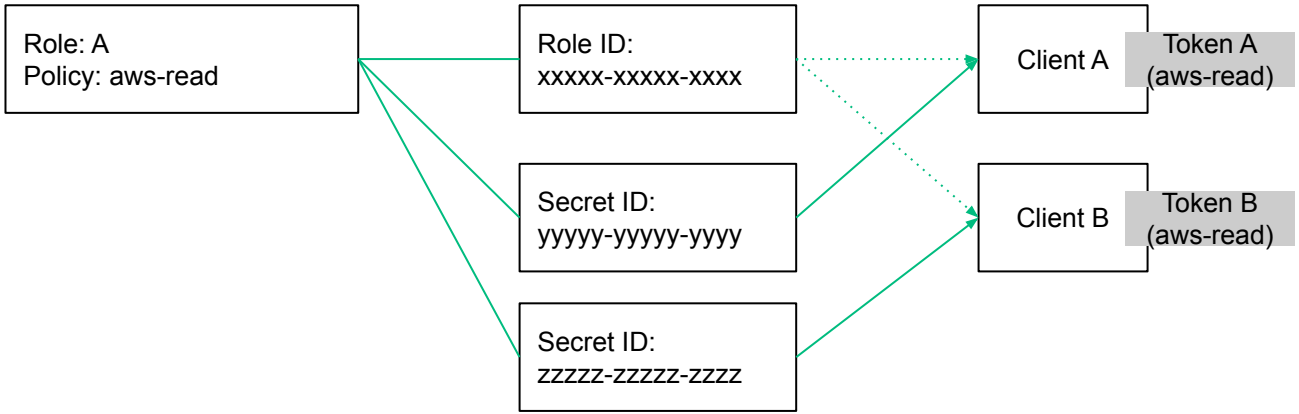


The demo pipeline:

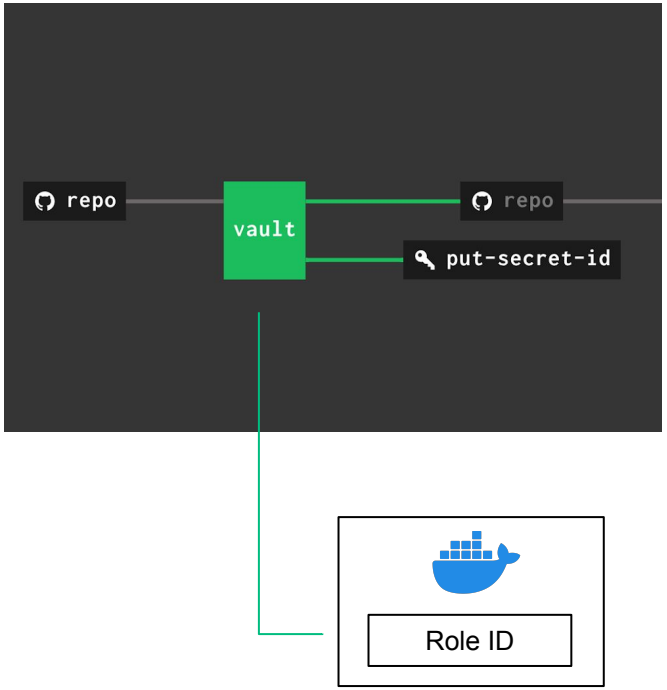


1. AppRole 기반 인증

- **AppRole**은 서비스, 애플리케이션 혹은 **VM**처럼 사람이 아닌 클라이언트 인증에 특화
- **RoleID**와 **SecretID** 조합으로 클라이언트를 인증
  - 서로 다른 채널을 통해 생성.
  - **SoD (Separation of Duty)**를 통한 토큰 별 **RoleID** 및 **SecretID** 생성.
- **AppRole** 구성 시 정의한 역할**(Role)** 기반으로 정책이 할당.

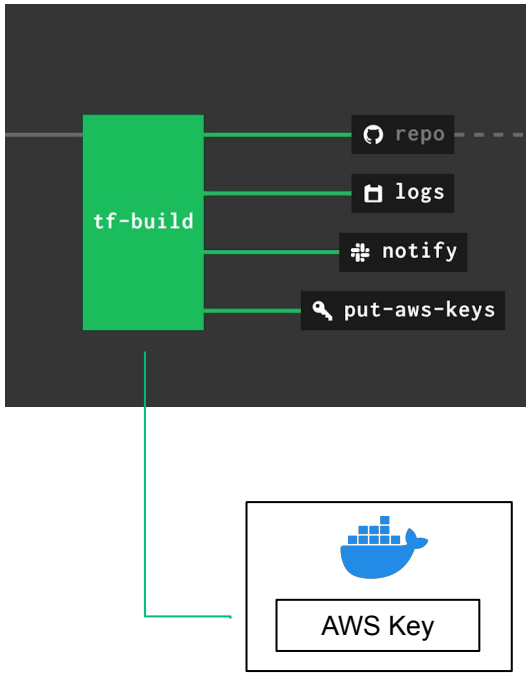


1. AppRole 기반 인증



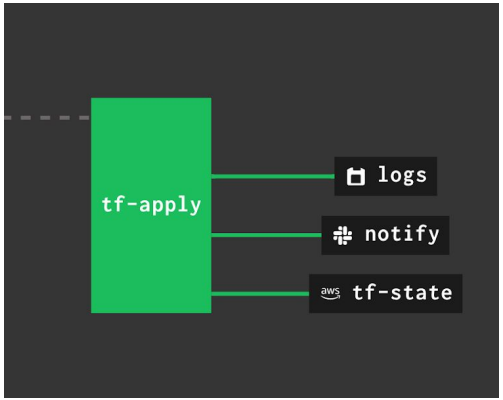
- RoleID는 $ docker build. 명령어를 통해 삽입되어, 도커 컨테이너에 내장.
- 해당 작업은 Worker로 동작할 도커 컨테이너를 내려받음.
- 도커 컨테이너를 내려받는 동안 SecretID가 생성됨. SecretID는 해당 작업 수행 시 매번 새롭게 생성.
- SecretID는 Vault에 대한 로그인이 진행되면, 폐기됨
- 생성된 SecretID를 KV 시크릿 엔진에 저장.

2. AWS Key 발급 및 IaC 코드 검증 후 dry-run



- RoleID와 SecretID를 사용하여 Vault에 로그인하여 토큰을 생성.
- 생성된 토큰을 이용, Access Key와 Secret Access Key를 부여받음.
- 이 정보를 terraform에서 사용. Validation 과정 후 *$ terraform plan* 수행.
- 해당 토큰은 사용 이후 폐기됨.

3. 인프라 배포 및 AWS Access Key 회수

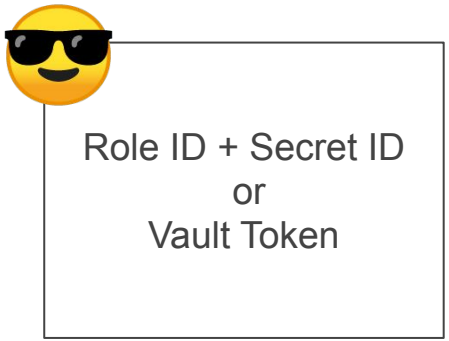


- $ terraform apply 명령어를 사용하여 인프라 배포 수행
- 파이프라인 종료 시
  - Secret ID : 폐기
  - Vault 토큰 : 폐기
  - AWS key : 폐기
- 사용되는 시크릿은 실행 과정 중에만 존재하게 됨.

Advanced Topic

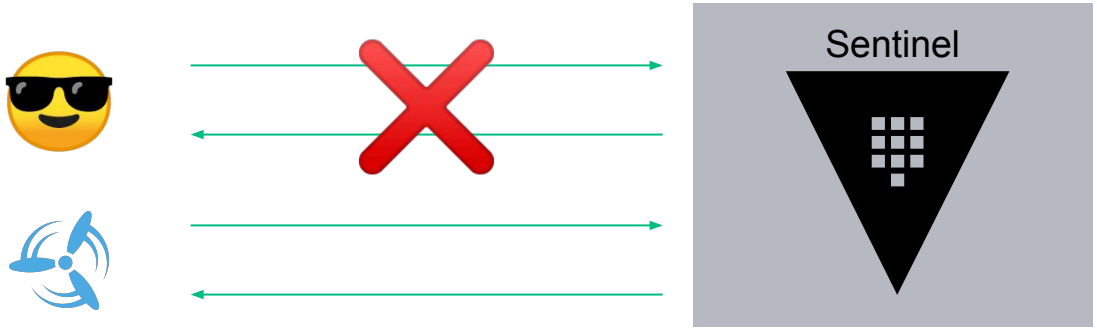
- Sentinel (The Enterprise Only Feature)

최악의 시나리오, 시크릿 유출!!!



- 해당 정보를 이용하여 Vault 로그인 후 토큰 생성 가능.
- 이후, AWS 접속 정보 획득 가능

HashiCorp Sentinel for Vault (Enterprise Feature)



Vault 접근 CIDR 체크

```
import "sockaddr"
import "strings"

# We expect logins to come only from our private IP range
cidrcheck = rule {
    sockaddr.is_contained("10.20.0.0/16",
request.connection.remote_addr)
}

main = rule when strings.has_prefix(request.path, "auth/ldap/login") {
    cidrcheck
}
```

기 생성된 토큰에 대한 즉각적인 접근 거부

```
import "time"

main = rule when not request.unauthenticated {
    time.load(token.creation_time).unix >
      time.load("2017-09-17T13:25:29Z").unix
}
```

Thank You

jsp@hashicorp.com

learn.hashicorp.com

hashicorp.com/events/#snapshots