



Amazon EKS CI/CD Pipeline

신 정 섭 (ssupp@amazon.com)

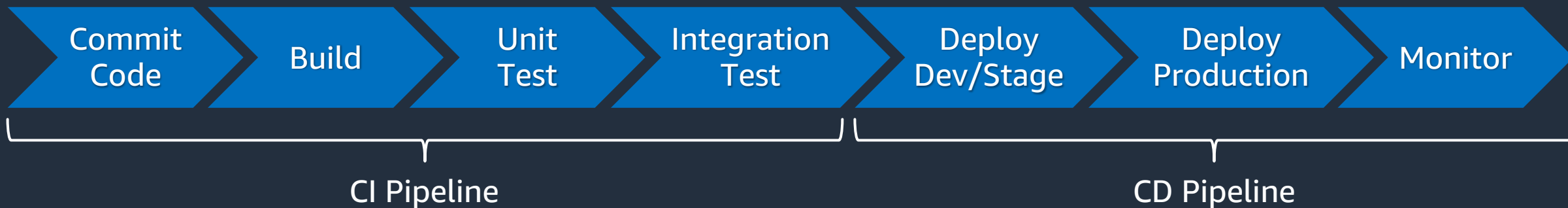
AWS Solutions Architect

Agenda

- CI/CD Pipeline with Amazon EKS
- Continuous Integration (CI) with Container
- Continuous Deployment (CD), GitOps with Amazon EKS Cluster

CI/CD Pipeline with Amazon EKS

CI/CD Pipeline with Amazon EKS

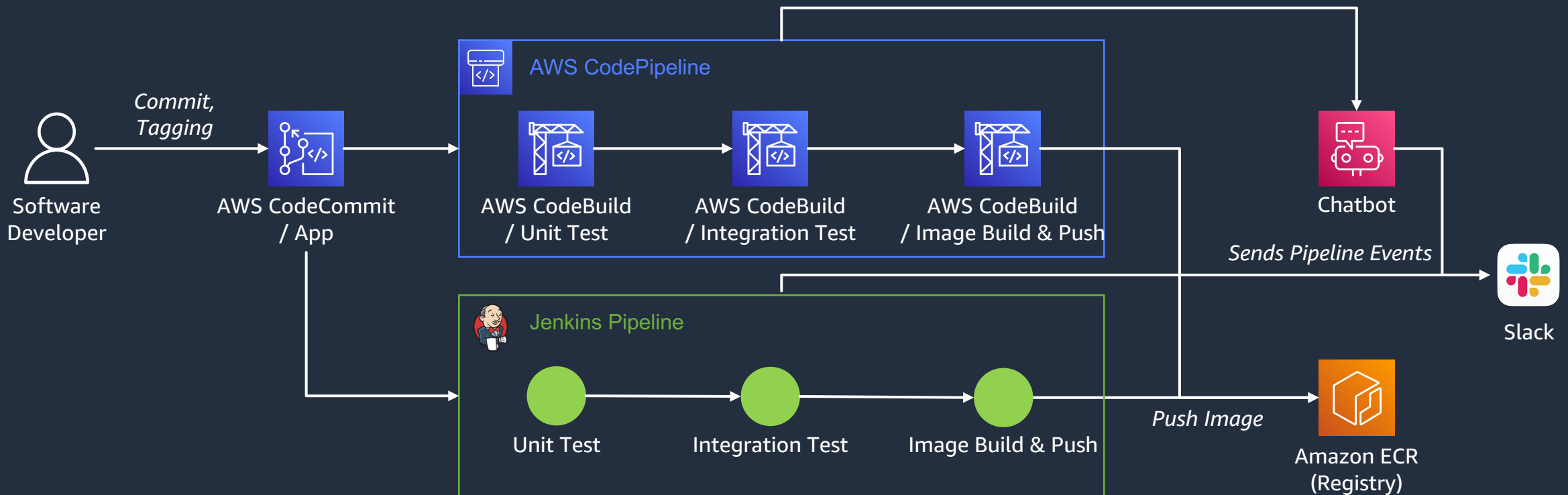


- Code Commit 이후 Build, Test 자동 수행
- 다양한 Platform, Service 이용하여 구성 가능
- CI 수행 결과 Test된 Container Image 생성
- Dockerfile 작성 필요

- 생성된 Container를 EKS Cluster에 배포
- 일반적으로 GitOps를 이용하여 구성
- 일반적으로 Dev/Stage 환경은 자동 배포 수행, Production 환경은 수동 배포 수행
- ArgoCD, Flux와 같은 GitOps 기반의 OpenSource 이용하여 구성

Continuous Integration (CI) with Container

CI Pipeline / Jenkins & AWS CodePipeline



- 다양한 Platform, Service 이용하여 구성 가능
- Image Build & Push 단계에서 Source Code, Dockerfile, Docker CLI를 이용
- CI Pipeline 수행 시간이 오래걸리기 때문에 Slack을 통해서 Pipeline Event 수신 권장

Container Image Tag Policy

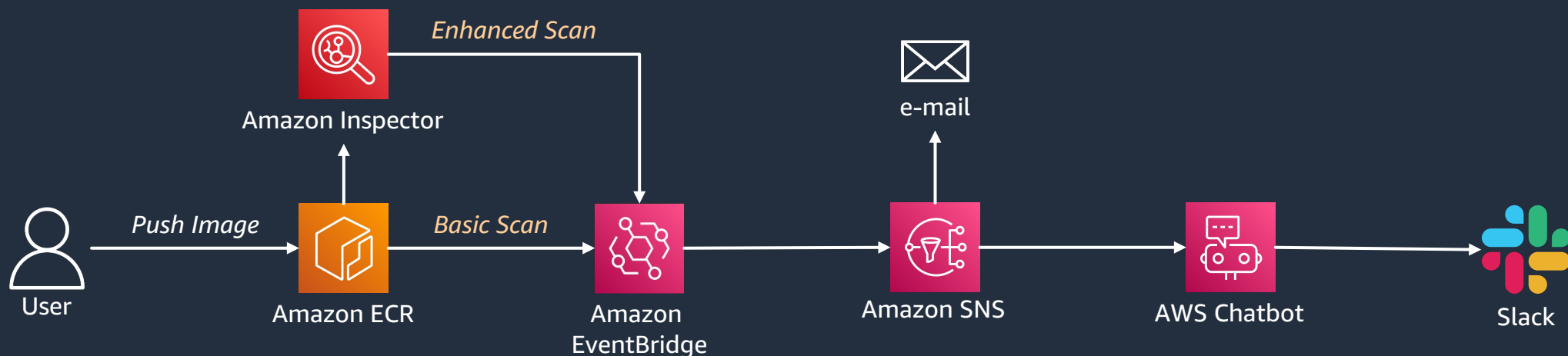
- Commit, Pull Request 승인 시

- Commit ID로 Tagging
- web-server:[Commit ID]
- Ex) web-server:57576f7

- Tagging, Release 수행 시

- Tag, Release Version으로 Tagging
- web-server:[Tag Version]
- Ex) web-server:2.1.2

Image Scan with Amazon ECR



- Amazon ECR에서는 Image Scan 기능을 통해서 Image 취약점 파악 가능
- Image Scan 결과는 Amazon EventBridge를 통해서 User에게 전달 가능
- Basic Scan : “Open-source Clair Project”에서 관리하는 OS 취약점 탐지 수행
- Enhanced Scan : “Amazon Inspector”와 연동하여 OS 및 취약점 및 Program Language 취약점 탐지 수행

Container Image Build for Multi Architecture

- `docker buildx` 명령어를 활용하여 동일한 Repo, Tag를 갖는 Multi Architecture Image 구성 가능
- AWS EC2 (x86), AWS EC2 Graviton (ARM)을 위한 Container Image Build시 수행

```
(host)$ docker buildx create --platform linux/amd64,linux/arm64...
```

- Kubernetes의 kubelet은 자신이 동작하는 Worker Node와 동일한 Architecture의 Container Image 이용
 - Hybrid Architecture Cluster 환경에서 Architecture에 따른 별도의 배포 관리 불필요

Dockerfile Tip / Bundle Commands

- Image Layer를 줄이기 위해서 다수의 명령어를 묶어서 실행

```
FROM golang:1.18 as build
RUN apt update
RUN apt-get --no-install-recommends install -y gcc
...
```

Single Stage : Image Layer 3개

```
FROM golang:1.18 as build
RUN apt update && apt-get --no-install-
recommends install -y gcc
...
```

Multi-stage : Image Layer 2개

Dockerfile Tip / Multi-stage Build

- Image Layer를 줄이기 위해서 Build를 별도의 Stage에서 진행

```
FROM golang:1.18 as build
WORKDIR /go/src/app
COPY . .

RUN go mod download
RUN CGO_ENABLED=0 go build -o /app
CMD ["/app"]
```

Single Stage : Image Layer 6개

```
# Build Stage
FROM golang:1.18 as build
WORKDIR /go/src/app
COPY . .

RUN go mod download
RUN CGO_ENABLED=0 go build -o /go/bin/app

# Copy Stage
FROM alpine:3.18.3
COPY --from=build /app /
CMD ["/app"]
```

Multi-stage : Image Layer 3개

Dockerfile Tip / Container Init Process

- 좀비 Process 방지 : Container의 Init Process는 좀비 Process 처리 역할 수행 필요

```
# Build Stage
FROM golang:1.18 as build
WORKDIR /go/src/app
COPY . .

RUN go mod download
RUN CGO_ENABLED=0 go build -o /go/bin/app

# Copy Stage
FROM alpine:3.18.3
COPY --from=build /app /
CMD ["/app"]
```

```
# Build Stage
FROM golang:1.18 as build
WORKDIR /go/src/app
COPY . .

RUN go mod download
RUN CGO_ENABLED=0 go build -o /go/bin/app

# Copy Stage
FROM alpine:3.18.3
RUN apk add --no-cache tini
COPY --from=build /app /
ENTRYPOINT ["/sbin/tini", "--"]
CMD ["/app"]
```

CD, GitOps with Amazon EKS Cluster

GitOps



- Continous Deployment 방법론 중 하나
- Git으로 배포 형상 관리
- SSOT (Single Source Of Truth) 원칙 기반
- Declarative (선언형) Model에 적합
- with Kubernetes
 - Declartive API를 이용하는 Kubernetes에 적합
 - Git으로 Kubernetes Manifest 관리

Argo CD



- Kubernetes Native GitOps Platform
- Web Console 제공


Application (CR, Custom Resource)

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    server: https://kubernetes.default.svc
    namespace: guestbook
```

Application Manifest

- 하나의 Manifest Group을 의미
- 일반적으로 하나의 Workload와 Mapping
- Application에 명시되어 있는 Git Repository에 Workload Manifest 존재
- 사용자는 Application Manifest를 직접 작성하거나, Web Console을 통해서 생성, 제어 가능

Argo CD Web Console / Application List


v2.4.124

Applications

APPLICATIONS TILES

+ NEW APP

↻ SYNC APPS

↻ REFRESH APPS

🔍 Search applications...

/

🔍 FILTERS

☐ ★ Favorites Only

SYNC STATUS

☐ ⚪ Unknown 0

☐ ✅ Synced 1

☐ ⚠️ OutOfSync 0

HEALTH STATUS

☐ ❓ Unknown 0

☐ 🔄 Progressing 0

☐ ⏸️ Suspended 0

☐ ❤️ Healthy 1

☐ ❤️ Degraded 0

☐ 🐛 Missing 0

aks-nodejs-argocd

Project: default

Labels:

Status: ❤️ Healthy ✅ Synced

Reposi... https://github.com/CIRCLECI-GWP...

Target ... circleci-project-setup

Path: manifests

Destin... in-cluster

Names... nodejs

↻ SYNC


↻





✖

Items per page: 10 ▼

Log out

Argo CD Web Console / Application Status


v2.4.124

Applications / aks-nodejs-argocd

APPLICATION DETAILS TREE

APP DETAILSAPP DIFFSYNCSYNC STATUSHISTORY AND ROLLBACKDELETEREFRESH

APP HEALTH
Healthy

CURRENT SYNC STATUS
Synced
To circleci-project-setup (074a470)
Author: CircleCI User <yemiwebby@gmail.com> -
Comment: Bumps docker tag [skip ci]

LAST SYNC RESULT
Sync OK
Succeeded 8 minutes ago (Thu Oct 06 2022 17:16:41 GMT+0100)
Author: CircleCI User <yemiwebby@gmail.com> -
Comment: Bumps docker tag [skip ci]

FILTERS

NAME
NAME

KINDS
KINDS

SYNC STATUS
☐ Synced 3
☐ OutOfSync 0

HEALTH STATUS
☐ Healthy 6
☐ Progressing 0
☐ Degraded 0
☐ Suspended 0
☐ Missing 0
☐ Unknown 0

aks-nodejs-argocd

nodejs ns

nodejs svc

nodejs deploy

nodejs ep

nodejs-xj9cs endpointslice

nodejs-849f48d55 rs

nodejs-849f48d55-hh8kj pod

nodejs-849f48d55-pkl5s pod

nodejs-849f48d55-sh2hs pod

Sync Policy

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  ...
  syncPolicy:
    automated: {}
```

Automated Sync

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  ...
  syncPolicy:
    automated:
      prune: true
```

Automated Sync with prune

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  ...
  syncPolicy:
    automated:
      selfHeal: true
```

Automated Sync

- Automated Sync
 - Polling을 통해서 Latest Commit (Manifest)를 찾아내고 자동 배포 수행
 - Default Polling 주기 : 3분
 - Argo CD ConfigMap에 설정 가능
- Prune
 - Git Repository에서 Manifest가 제거될 경우, Kubernetes Cluster에서도 제거
- Self Heal
 - Git Repository의 Manifest와 Kubernetes Cluster의 상태가 다른 경우 복구 수행

Manifest with Kustomize

- Manifest File 모듈화
- ArgoCD에서 지원

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
labels:
  app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
          env:
            - name: env
              value : dev
```

Dev Manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
labels:
  app: nginx
spec:
  replicas: 30
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
          env:
            - name: env
              value : prod
```

Prod Manifest

Kustomize X

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
labels:
  app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

Base Manifest

Kustomize O

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    spec:
      containers:
        - name: nginx
          env:
            - name: env
              value : dev
```

Dev Overlay
Manifest

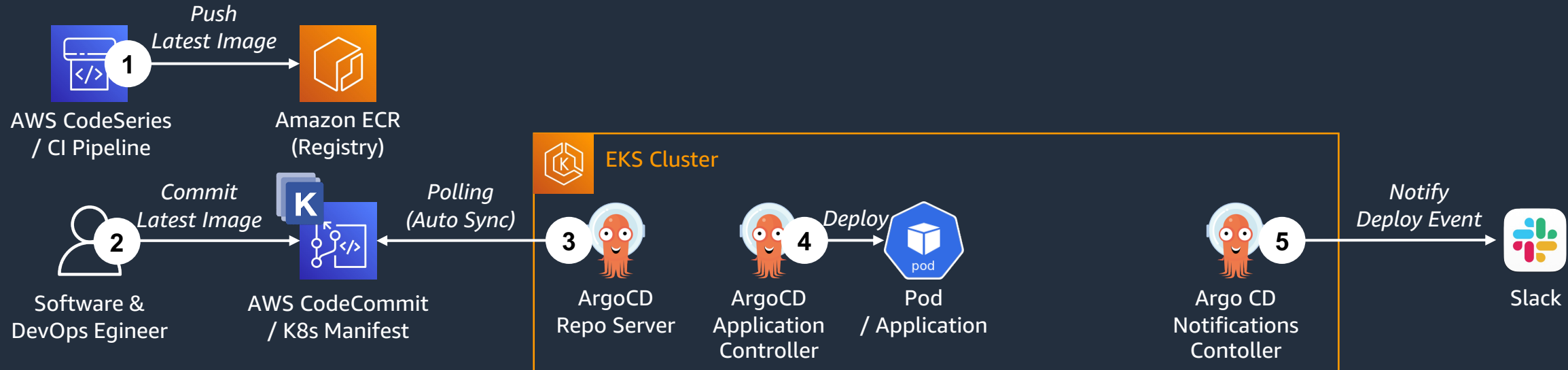
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 30
  template:
    spec:
      containers:
        - name: nginx
          env:
            - name: env
              value : prod
```

Prod Overlay
Manifest

Argo CD Components

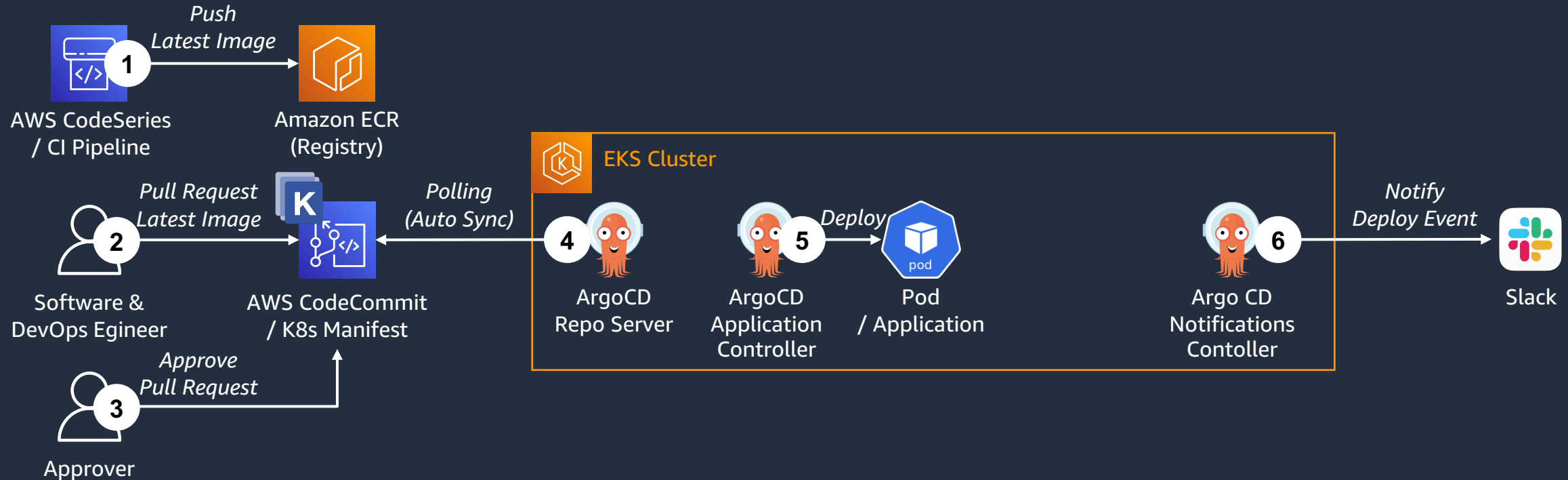
Name	Role
Application Controller	Application CR 관리, Workload 배포 수행
ApplicationSet Controller	ApplicationSet CR 관리, Application CR 생성
Dex Server	SSO 인증 연동
Notifications Controller	Slack, Email 등으로 배포 Event 전달
Redis	Redis Server
Repo Server	Github Repository의 K8s Manifest Caching, 변경
Server	Web Console 제공, argo CLI Backend
argo CLI	Application 생성, 조회, 삭제
Image Updater	최신 Container Image 정보를 받아 자동 Commit 수행

CD Pipeline / Manual Deployment



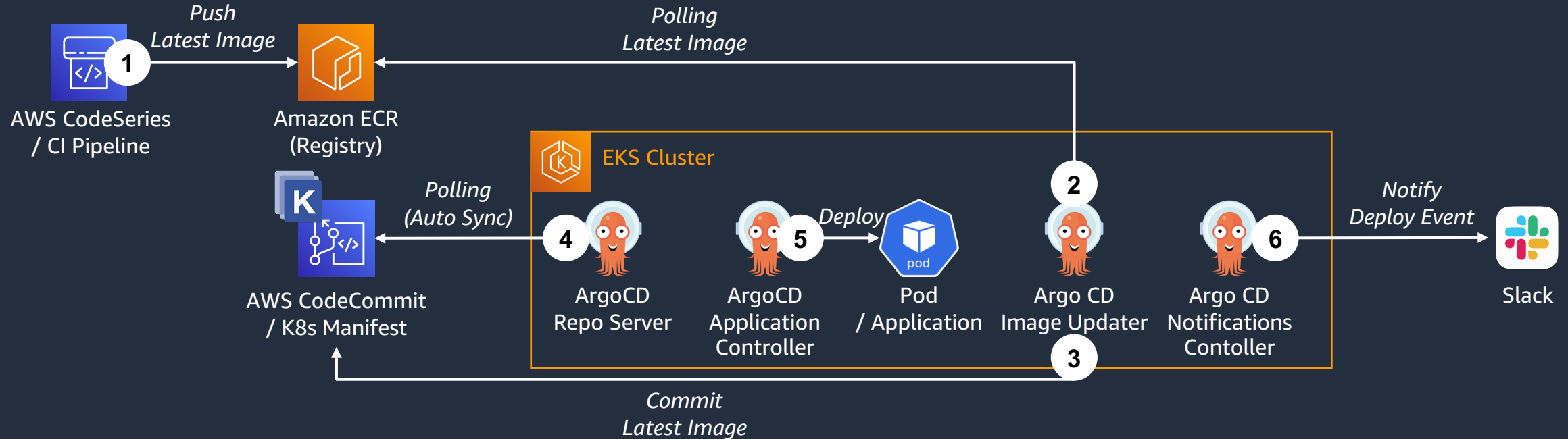
- User가 직접 최신 Container Image Commit 수행
- Argo CD Notifications를 통해서 배포 Event를 Slack으로 전달

CD Pipeline / Manual Deployment with Approval



- Git Repository의 Pull Request 기능을 활용하여 배포 요청 및 배포 승인 Process 구현
- 승인 요청, 승인 History 관리 가능

CD Pipeline / Auto Deployment



- 최신 Container Image가 Container Registry에 등록되면 자동으로 EKS Cluster에 배포 수행

Q & A