from Beginner to Master

DevOps를 위한 Docker 가상화

Section 10. Capstone Project

프로필

Dowon Lee

수강생 19K 강의평점 4.8(1K)

멘토링 ON

멘토링 설정

- 홈
- 강의
- 로드맵
- 수강평
- 게시글
- 블로그

수정하기

이 되었던 때가 있었습니
"IT 엔지니어"라고 적고

을 가지고 있습니다. 누구
람입니다. 개발자로써, 강
지만, 그래도, 남들보다 조

tecture, Blockchain,

강의 전체 6

[개정판] 웹 애플리케이션 개발을 위한 IntelliJ IDEA 설정

학습중

0강 / 25강 (0%)

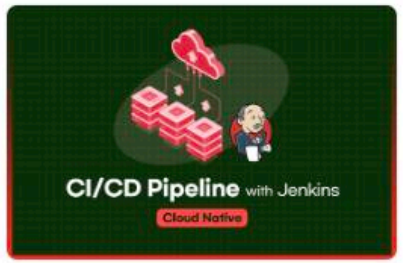
818명

멀티OS 사용을 위한 가상화 환경 구축 가이드 (Docker + Kubernetes)

학습중

0강 / 16강 (0%)

924명



Jenkins를 이용한 CI/CD Pipeline 구축

학습중

81강 / 83강 (98%)

독점 2709명

Spring Cloud로 개발하는 마이크로서비스 애플리케이션(MSA)

학습중

156강 / 156강 (100%)

독점 5531명

Spring Boot를 이용한 RESTful Web Services 개발

학습중

3강 / 48강 (6%)

독점 3862명

[구버전] 웹 애플리케이션 개발을 위한 IntelliJ IDEA 설정 (2020 ver.)

학습중

14강 / 14강 (100%)

4813명

목차

Section 10.

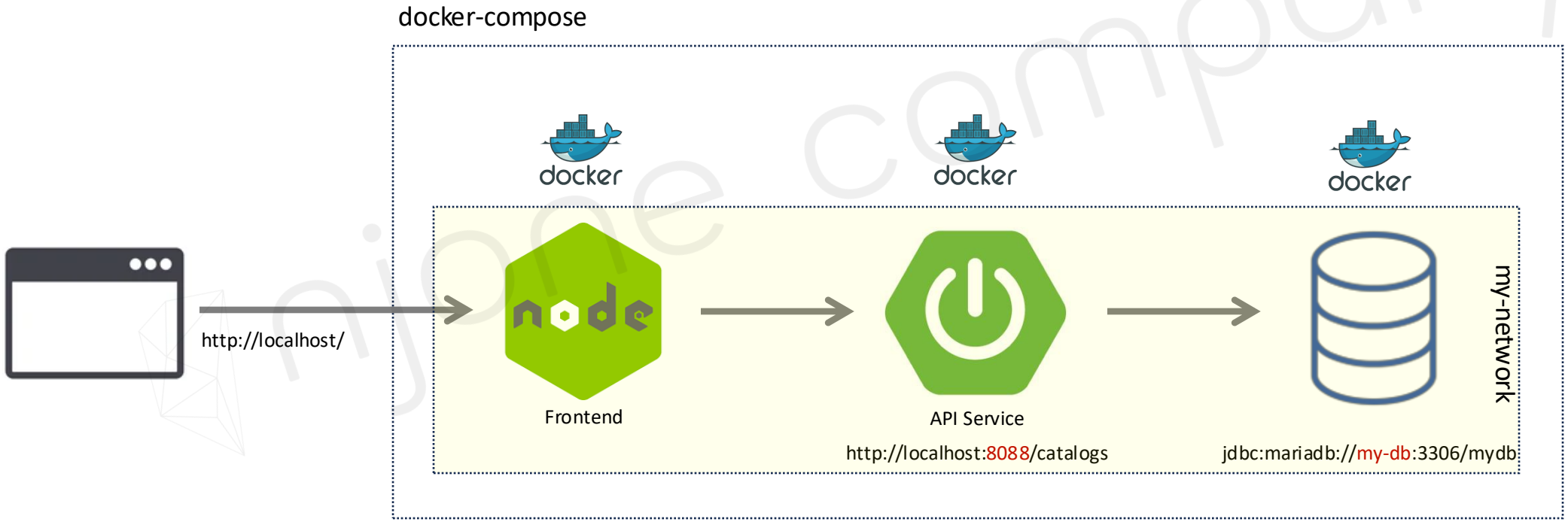
Capstone Project

- Deploy Web Application
- Docker를 활용한 자동화 빌드 시스템 구축

Deploy Web Application

Node + Spring Boot + MariaDB

- Docker Compose
  - Docker Image, Container의 활용
  - MSA의 이해, MSA Architecture의 장점
  - CI/CD Pipeline 구축
  - Docker Compose를 활용한 멀티 컨테이너 운영



Deploy Web Application

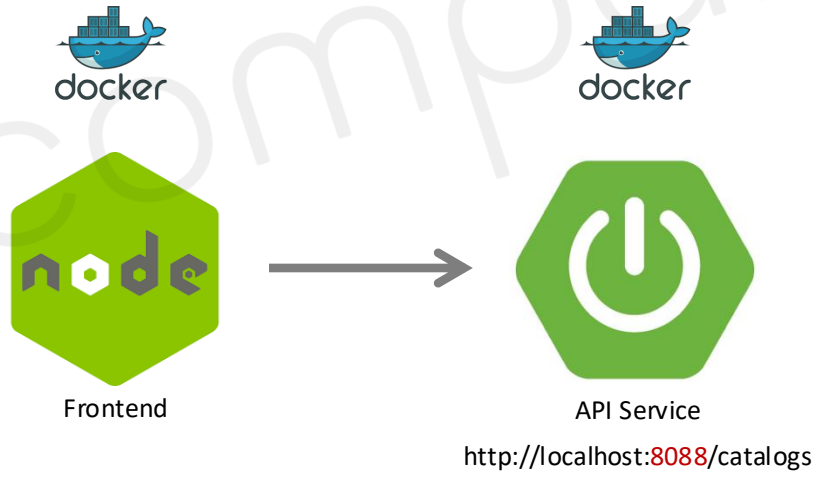
Node + Spring Boot + MariaDB

- Dockerfile + docker-compose.yml
  - Frontend + API Service를 위한 Dockerfile 작성
  - docker-compose.yml 파일을 정의하여 각 서비스들에 대한 Orchestrate
  - Network, Volume mount를 사용한 Container 관리

Deploy Web Application

Node + Spring Boot + MariaDB

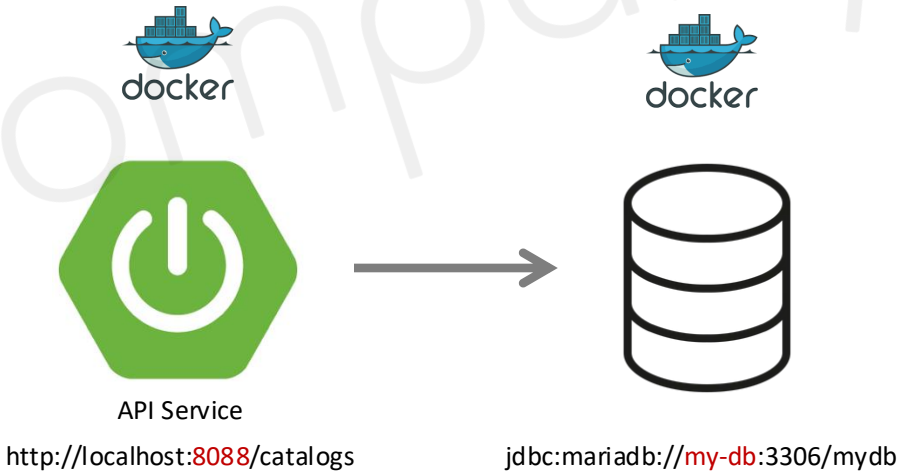
- Frontend Service
  - React로 구현 된 Web Application
  - API Service를 호출하여 UI에 결과 출력



Deploy Web Application

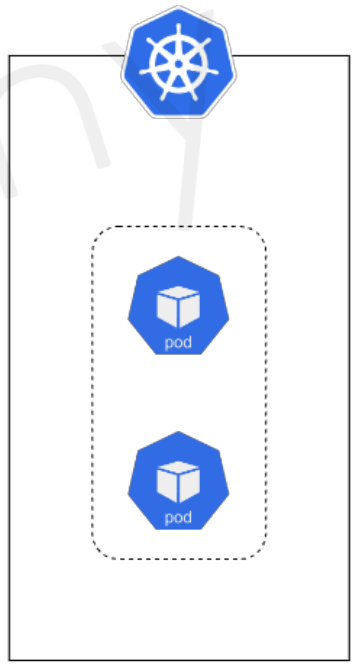
Node + Spring Boot + MariaDB

- API Service
  - Spring Boot + Spring framework로 구현 된 RESTful API 서비스
  - API Endpoint에 대한 CRUD 처리
  - 외부 RDB와의 연동



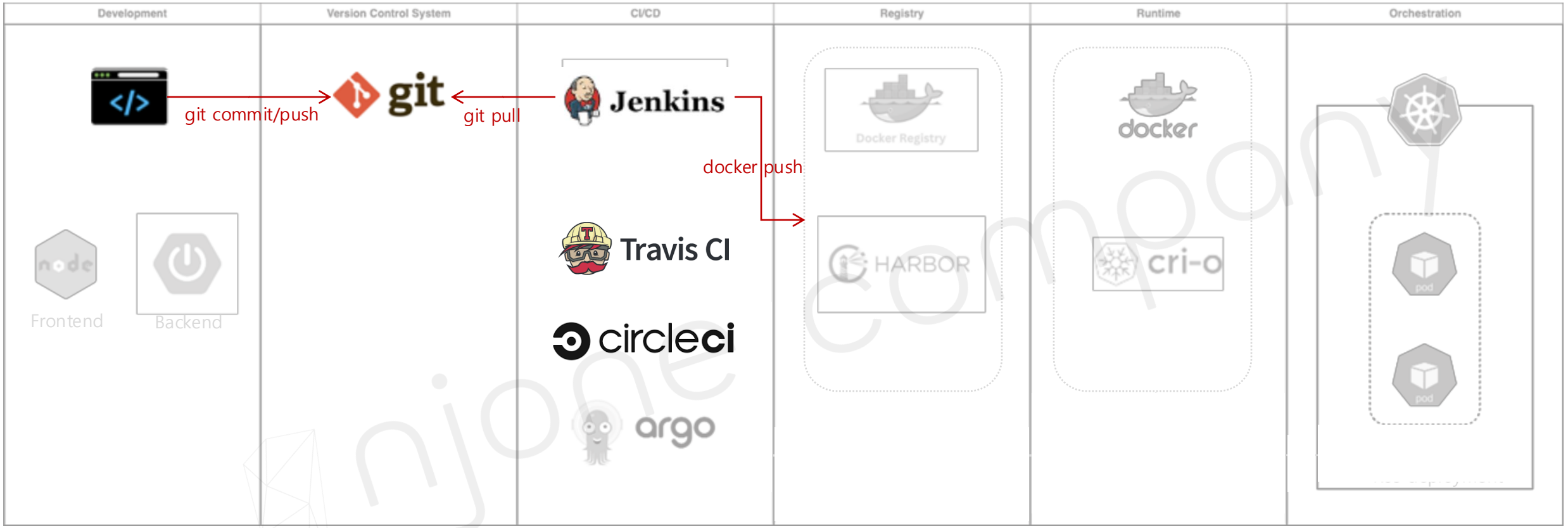
Docker를 활용한 자동화 빌드 시스템 구축

CI/CD

Development	Version Control System	CI/CD	Registry	Runtime	Orchestration
  Frontend  Backend		   	 	  	

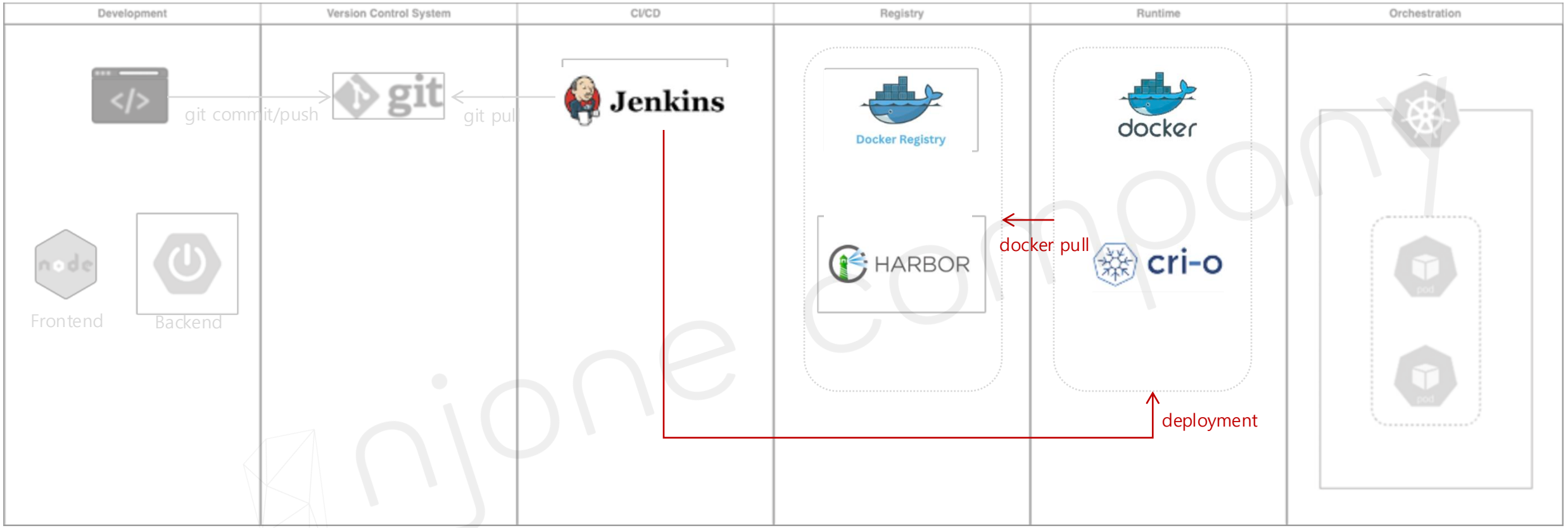
Docker를 활용한 자동화 빌드 시스템 구축

CI(Continuous Integration)



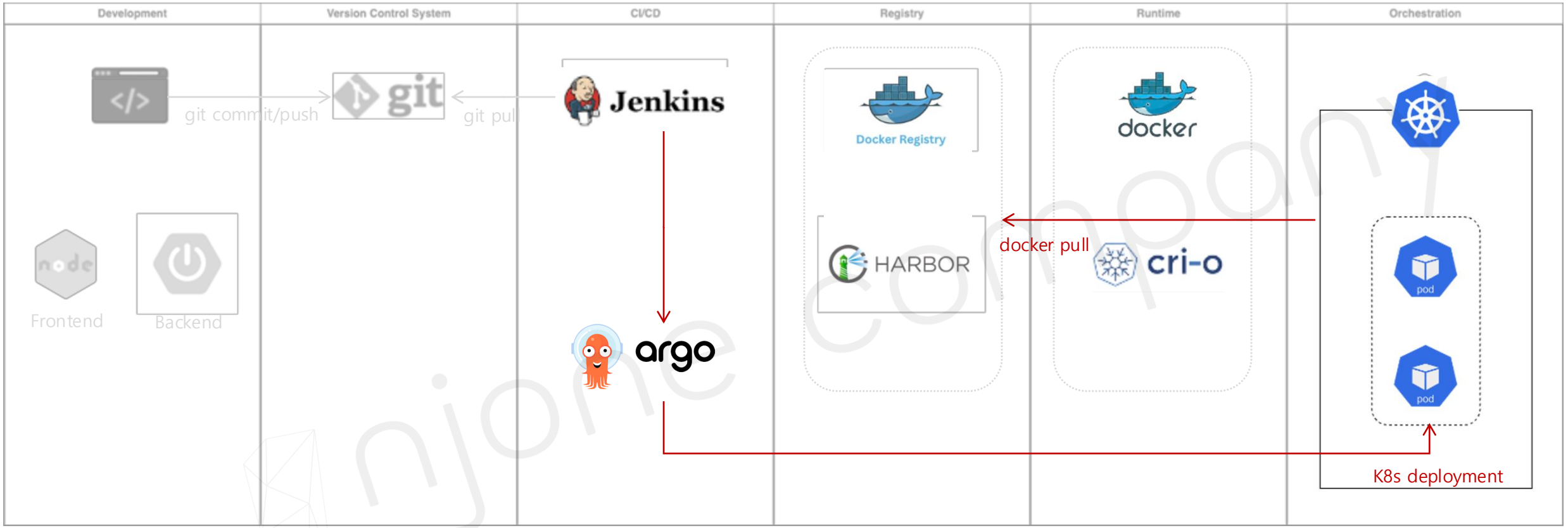
Docker를 활용한 자동화 빌드 시스템 구축

CD(Continuous Deployment) → Container runtime



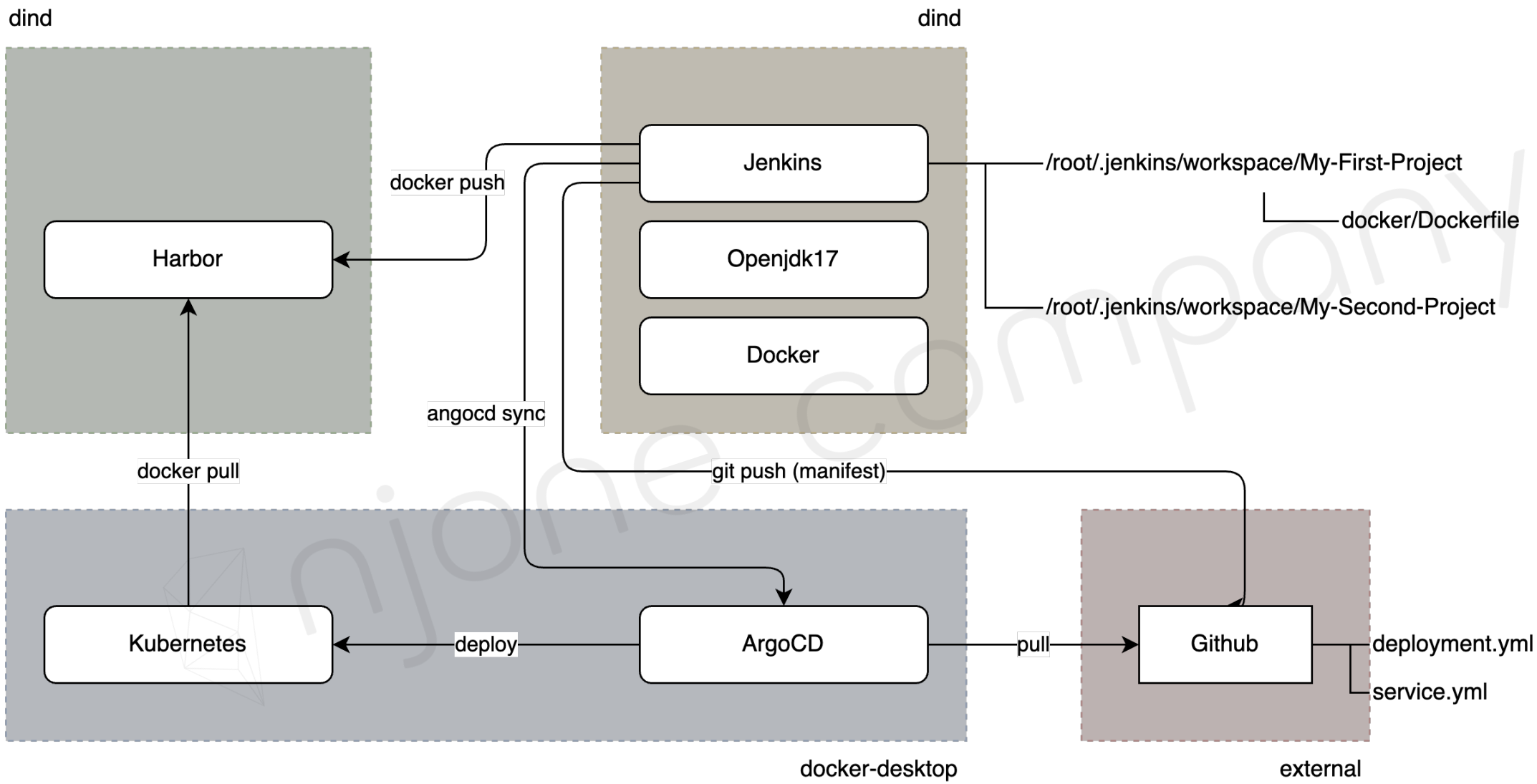
Docker를 활용한 자동화 빌드 시스템 구축

CD(Continuous Deployment) → Kubernetes + Argocd



Docker를 활용한 자동화 빌드 시스템 구축

CI/CD Pipeline



Jenkins

- 설치
  - https://www.jenkins.io
  - Docker dind 이미지에서 설치 (Jenkins + Docker 사용)
  - Port mapping 추가 –p 8080:8080 등
  - Private registry 사용을 위한 /etc/docker/daemon.json 수정 → inseure-registries 추가
  - Openjdk 설치

```
$ yum list java*-openjdk-devel
$ yum install java*-openjdk-devel
$ .bashrb에 JAVA_HOME 등록
```

```
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-17.0.1.0.12-2.el8_5.x86_64
export PATH=$JAVA_HOME/bin:$PATH
```

```
$ source ~/.bashrc
```

Jenkins

- 설치
  - 추가 SW 설치

```
$ yum install -y wget git
$ wget https://mirrors.tuna.tsinghua.edu.cn/jenkins/war-stable/2.452.1/jenkins.war
```

- Jenkins 실행
  - 실행

```
$ java -jar jenkins.war
```

  - 초기 Plugins 설치, 계정 추가

Jenkins

- 설정

  - Plugins 추가 설치

    - Docker API Plugin, Docker, Docker Commons Plugin, Docker Pipeline, Docker Plugin
    - docker-build-step
    - Maven Integration, Publish over SSH, Stage View 등

  - Tools

    - JDK (Openjdk 17.0.1m JAVA_HOME), Maven 3.9.5

  - System

    - Docker Build → unix:///var/run/docker.sock

  - Credentials

    - Harbor 계정 추가 → user1

Jenkins – My-First-Project

- Git → https://github.com/joneconsulting/cicd-web-project
- Maven Build → clean compile package –DskipTests=true
- Post Steps
  - Execute Docker command → Create/build image
    - Build context folder → $WORKSPACE
    - Tag of the resulting docker image → *192.168.0.41*/devops/helloworld:$BUILD_NUMBER
  - Execute Docker command → Push image
    - Name of the image to push (repository/image) → *192.168.0.41*/devops/helloworld
    - Tag → $BUILD_NUMBER
    - Registry → (Empty)
    - Docker registry URL → *192.168.0.41*
    - Registry credentials → user1/*****

Jenkins – My-Second-Project

- Pipeline item

```
pipeline {
    agent none
    tools {
        maven "Maven3.9.5"
    }

    stages {

        // ①

        // ②

        // ③

    }
}
```

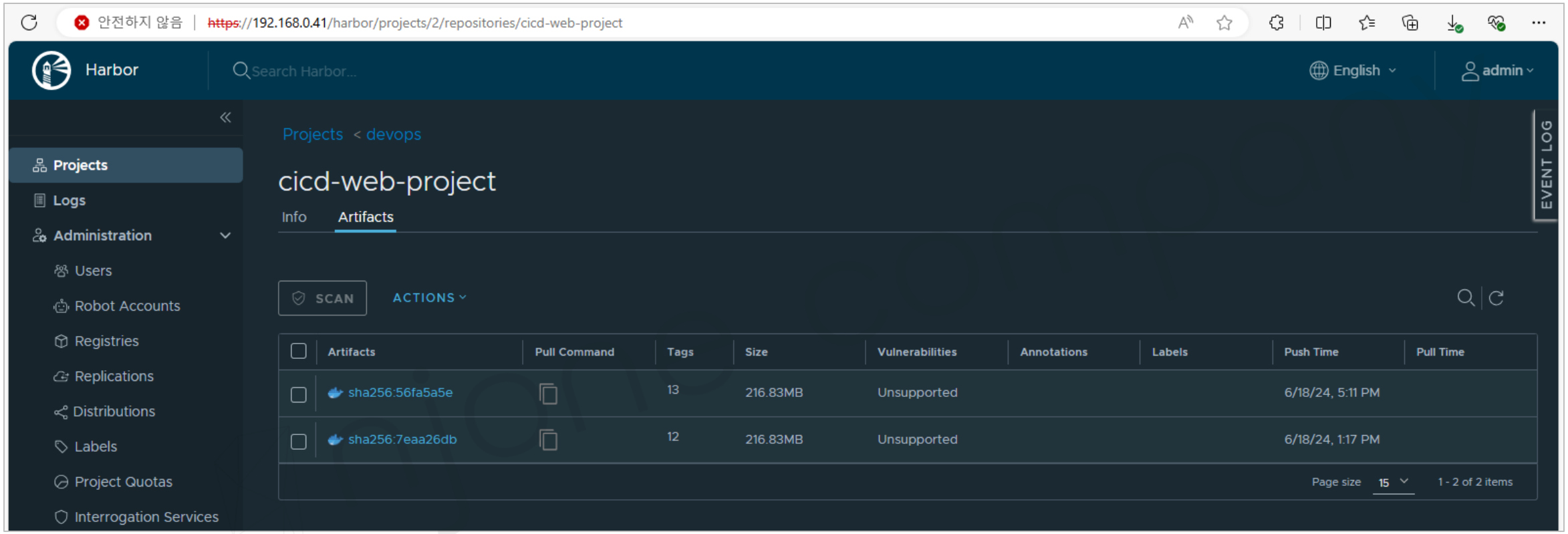
```
stage('Maven Install') {
            agent any
            steps {
                git branch: 'main', url: 'https://github.com/joneconsulting/cicd-web-project'
                sh 'mvn clean compile package -DskipTests=true'
            }
}
```

```
stage('Docker Build') {
            agent any
            steps {
                sh 'docker build -t 192.168.0.41/devops/cicd-web-project:$BUILD_NUMBER .'
            }
}
```

```
stage('Docker Push') {
            agent any
            steps {
                withDockerRegistry(credentialsId: 'harbor-user', url: 'https://192.168.0.41') {
                    sh 'docker push 192.168.0.41/devops/cicd-web-project:$BUILD_NUMBER'
                }
            }
}
```

Jenkins – My-Second-Project

- Harbor registry



K8s + ArgoCD 연동

- K8s
  - Docker desktop에 포함된 K8s 사용
- ArgoCD 설치
  - kubectl create namespace argocd
  - kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.3.3/manifests/install.yaml
  - kubectl get pod -n argocd
  - kubectl get service -n argocd
  - kubectl edit svc argocd-server -n argocd → NodePort 변경

K8s + ArgoCD 연동

- ArgoCD Cli 설치
  - Windows) Powershell 사용

```
$version = (Invoke-RestMethod https://api.github.com/repos/argoproj/argo-cd/releases/latest).tag_name
$url = "https://github.com/argoproj/argo-cd/releases/download/" + $version + "/argocd-windows-amd64.exe"
$output = "argocd.exe"
Invoke-WebRequest -Uri $url -OutFile $output
```

  - MacOS) homebrew 사용

```
$ brew install argocd
```

  - Terminal(CMD)에서 argocd 설치 확인

```
$ argocd version
```

K8s + ArgoCD 연동

- ArgoCD Login
  - admin 로그인 암호 확인 (방법1, MacOS)

```
$ kubectl get secret argocd-initial-admin-secret -n argocd -o “jsonpath={.data.password}” | base64 -d
```

  - admin 로그인 암호 확인 (방법2, Windows)

```
$ kubectl get secret argocd-initial-admin-secret -n argocd -o “jsonpath={.data.password}”
ABCSEDFGHKKSKQL… → encode.b64로 저장
$ certutil -decode encode.b64 decode.txt
```

  - admin 로그인 암호 확인 (방법3)

```
$ argocd admin initial-password -n argocd
```

Docker를 활용한 자동화 빌드 시스템 구축

K8s + ArgoCD 연동 (Windows)

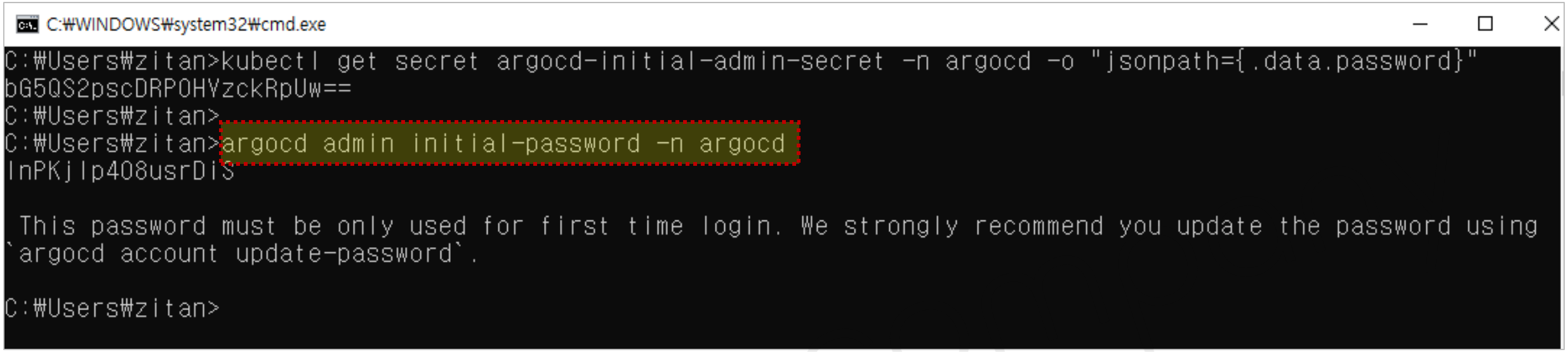
Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:₩Users₩zitan> $version = (Invoke-RestMethod https://api.github.com/repos/argoproj/argo-cd/releases/latest).tag_name
PS C:₩Users₩zitan> $url = "https://github.com/argoproj/argo-cd/releases/download/" + $version + "/argocd-windows-amd64.exe"
PS C:₩Users₩zitan> $output = "argocd.exe"
PS C:₩Users₩zitan> Invoke-WebRequest -Uri $url -OutFile $output
PS C:₩Users₩zitan>
```

C:₩WINDOWS₩system32₩cmd.exe

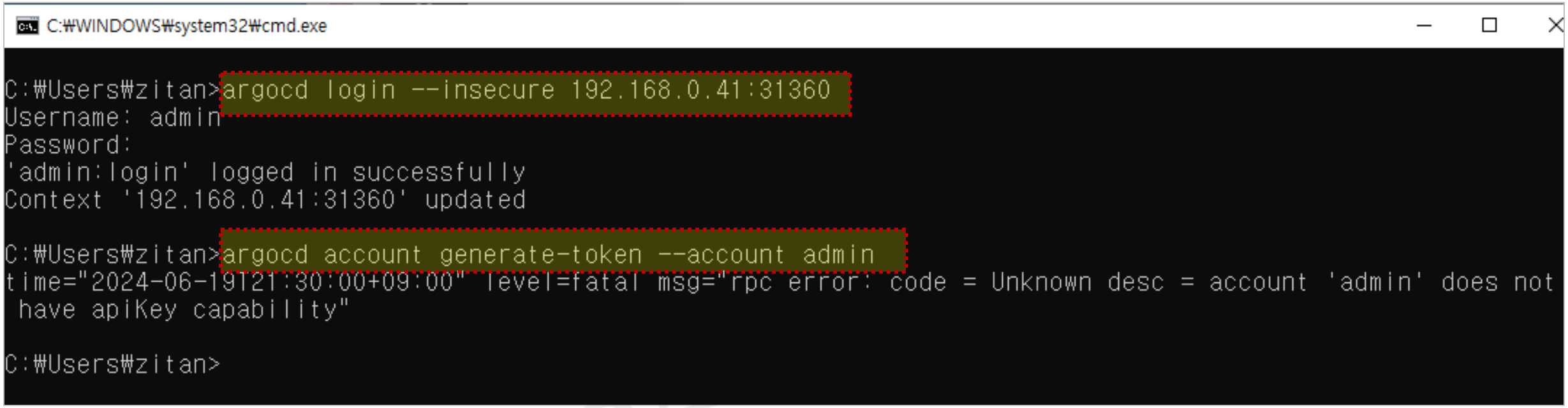
```
C:₩Users₩zitan>kubectl get secret argocd-initial-admin-secret -n argocd -o "jsonpath={.data.password}"
bG5QS2pscDRPOHVzckRpUw==
C:₩Users₩zitan>
C:₩Users₩zitan>argocd admin initial-password -n argocd
lnPKjlp408usrDiS

 This password must be only used for first time login. We strongly recommend you update the password using
`argocd account update-password`.

C:₩Users₩zitan>
```

K8s + ArgoCD 연동 (Windows 실행 명령어)



Docker를 활용한 자동화 빌드 시스템 구축

K8s + ArgoCD 연동 (Windows 실행 명령어)



K8s + ArgoCD 연동 (Windows 실행 명령어)

C:₩WINDOWS₩system32₩cmd.exe

```
C:₩Users₩zitan>argocd login --insecure 192.168.0.41:31360
Username: admin
Password:
'admin:login' logged in successfully
Context '192.168.0.41:31360' updated

C:₩Users₩zitan>argocd account generate-token --account admin
time="2024-06-19T21:30:00+09:00" level=fatal msg="rpc error: code = Unknown desc = account 'admin' does not
 have apiKey capability"

C:₩Users₩zitan>kubectl edit cm argocd-cm -n argocd
configmap/argocd-cm edited
```

K8s + ArgoCD 연동 (Windows 실행 명령어)

*kubectl-edit-1506214661.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"ConfigMap","metadata":{"annotations":{},"labels":{"app.kubernetes.io/name":"argocd-cm","app.kubernetes.io/part-of":"argc
  creationTimestamp: "2024-06-19T12:23:49Z"
  labels:
    app.kubernetes.io/name: argocd-cm
    app.kubernetes.io/part-of: argocd
  name: argocd-cm
  namespace: argocd
  resourceVersion: "191116"
  uid: ff83e6ea-8339-4c99-88ae-c0f6822bfa5e
data:
  accounts.admin: apiKey
```

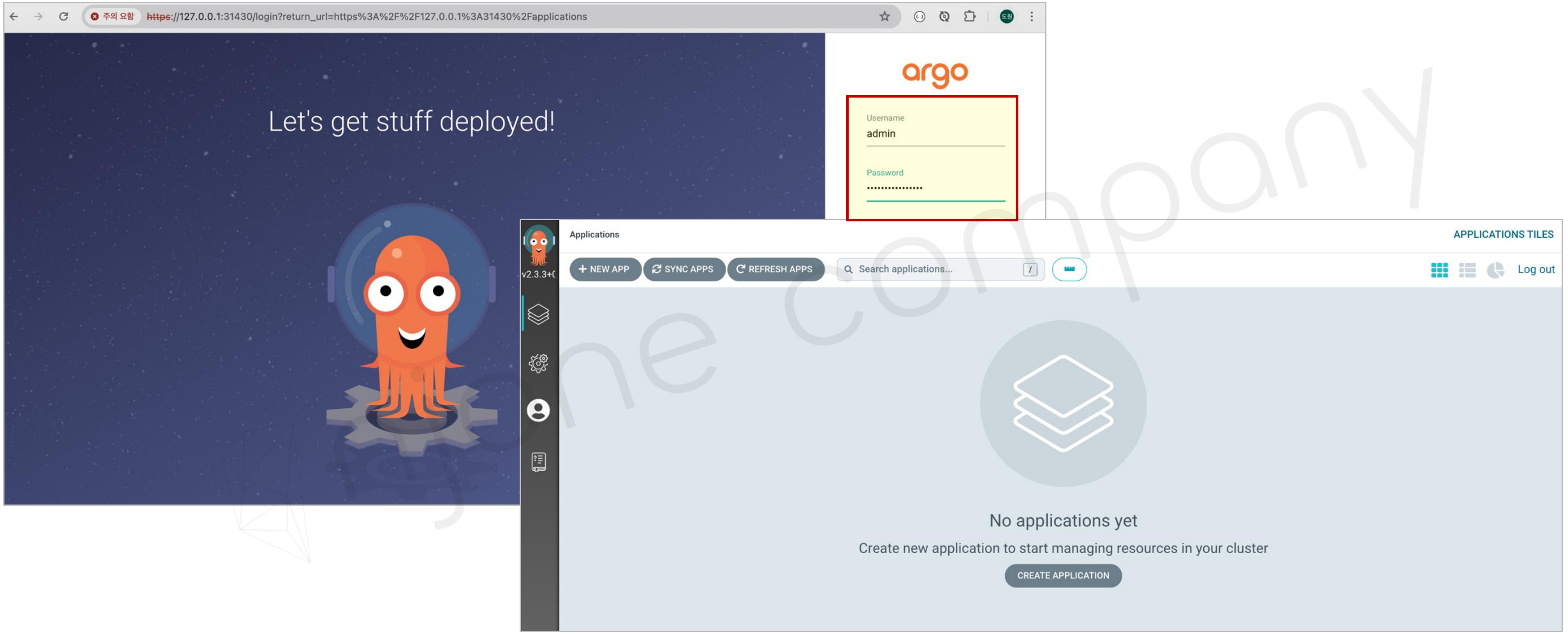
Docker를 활용한 자동화 빌드 시스템 구축

K8s + ArgoCD 연동 (Windows 실행 명령어)

```
C:₩Users₩zitan>argocd account generate-token --account admin
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJhcmdvY2QiLCJzdWIiOiJhZG1pbjphcGlLZXkiLCJuYmYiOjE3MTg4MDAyODcsImlhdCI6MTcxODgwMDI4NywianRpIjoiZDQ4NDFhYWMtYTY5NyOONGUyLTIjNTYtMDAyODE3OWY3N2IxIn0.snPsPw8PsaJwKSJ9OWIPNBKe9xJKjvnTPWQQpq9aQXU

C:₩Users₩zitan>curl -k -L -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJhcmdvY2QiLCJzdWIiOiJhZG1pbjphcGlLZXkiLCJuYmYiOjE3MTg4MDAyODcsImlhdCI6MTcxODgwMDI4NywianRpIjoiZDQ4NDFhYWMtYTY5NyOONGUyLTIjNTYtMDAyODE3OWY3N2IxIn0.snPsPw8PsaJwKSJ9OWIPNBKe9xJKjvnTPWQQpq9aQXU" https://192.168.0.41:31360/api/v1/applications
{"metadata":{"resourceVersion":"191848"},"items":null}
C:₩Users₩zitan>
```

K8s + ArgoCD 연동

- ArgoCD Login



K8s + ArgoCD 연동

- ArgoCD Login
  - argocd login --insecure ***192.168.0.41:30764*** ← NodePort
- ArgoCD Token 생성
  - argocd account generate-token --account admin

Error) account 'admin' does not have apiKey capability 발생 시
- $ kubectl edit cm argocd-cm -n argocd → 아래 항목 추가

```
data:
    accounts.admin: apiKey
```

  - curl 명령어로 applications 확인

```
$ curl -k -L -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJhcmdvY2QiLCJ..." \
        https://192.168.0.41:30674/api/v1/applications
```

Docker를 활용한 자동화 빌드 시스템 구축

K8s + ArgoCD 연동 (Windows)

```
C:₩WINDOWS₩system32₩cmd.exe

C:₩Users₩zitan>argocd login --insecure 192.168.0.41:31360
Username: admin
Password:
'admin:login' logged in successfully
Context '192.168.0.41:31360' updated

C:₩Users₩zitan>argocd account generate-token --account admin
time="2024-06-19T21:30:00+09:00" level=fatal msg="rpc error: code = Unknown desc = account 'admin' does not
 have apiKey capability"

C:₩Users₩zitan>kubectl edit cm argocd-cm -n argocd
configmap/argocd-cm edited

C:₩Users₩zitan>argocd account generate-token --account admin
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJhcmdvY2QiLCJzdWIiOiJhZG1pbjphcGlLZXkiLCJuYmYiOjE3MTg4MDAyOD
csImlhdCI6MTcxODgwMDI4NywianRpIjoiZDQ4NDFhYWMtYTY5NyOONGUyLTIjNTYtMDAyODE3OWY3N2IxIn0.snPsPw8PsaJwKSJ9OWIPN
BKe9xJKjvnTPWQQpq9aQXU

C:₩Users₩zitan>curl -k -L -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJhcmdvY2
QiLCJzdWIiOiJhZG1pbjphcGlLZXkiLCJuYmYiOjE3MTg4MDAyODcsImlhdCI6MTcxODgwMDI4NywianRpIjoiZDQ4NDFhYWMtYTY5NyOON
GUyLTIjNTYtMDAyODE3OWY3N2IxIn0.snPsPw8PsaJwKSJ9OWIPNBKe9xJKjvnTPWQQpq9aQXU" https://192.168.0.41:31360/api/
v1/applications
{"metadata":{"resourceVersion":"191848"},"items":null}
C:₩Users₩zitan>
```

*kubectl-edit-1506214661.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# Please edit the object below. Lines beginning with a '#' w
# and an empty file will abort the edit. If an error occurs wh
# reopened with the relevant failures.
#
apiVersion: v1
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"ConfigMap","metadata":{"annot
  creationTimestamp: "2024-06-19T12:23:49Z"
  labels:
    app.kubernetes.io/name: argocd-cm
    app.kubernetes.io/part-of: argocd
  name: argocd-cm
  namespace: argocd
  resourceVersion: "191116"
  uid: ff83e6ea-8339-4c99-88ae-c0f6822bfa5e
data:
  accounts.admin: apiKey
```

K8s + ArgoCD 연동

- ArgoCD New Project
  - Application Name: helloworld
  - Project: default
  - Cluster: in-cluster (https://Kubernetes.default.svc)
  - Namespace: default
  - Repo URL: https://github.com/joneconsulting/argocd-sample
  - Target revision: main
  - Path: base
  - Retry options: Retry disabled

- Sample → https://github.com/joneconsulting/argocd-sample
  - base/deployment.yml
  - base/service.yml

K8s + ArgoCD 연동

- K8s에서 docker-registry에 대한 인증 정보 등록

```
$ kubectl create secret docker-registry regcred \
--docker-server=192.168.0.41 \
--docker-username=user1 \
--docker-password=Harbor12345 \
--docker-email=
```

- MacOS) kubectl get secret regcred --output=yaml
- MacOS) kubectl get secret regcred --output="jsonpath={.data.\.dockerconfigjson}" | base64 --decode

```
{"auths":{"127.0.0.1":{"username":"user1","password":"Harbor12345","auth":"dXNlcjE6SGFyYm9yMTIzNDU="}}
```

- Windows) certutil -decode dockerregistry.encode dockerregistry.txt 등으로 내용 확인 가능

```
$ kubectl get secret regcred -n argocd -o “jsonpath={.data.password}”
```

K8s + ArgoCD 연동

- deployment.yml 파일에 imagePullSecrets 항목 추가

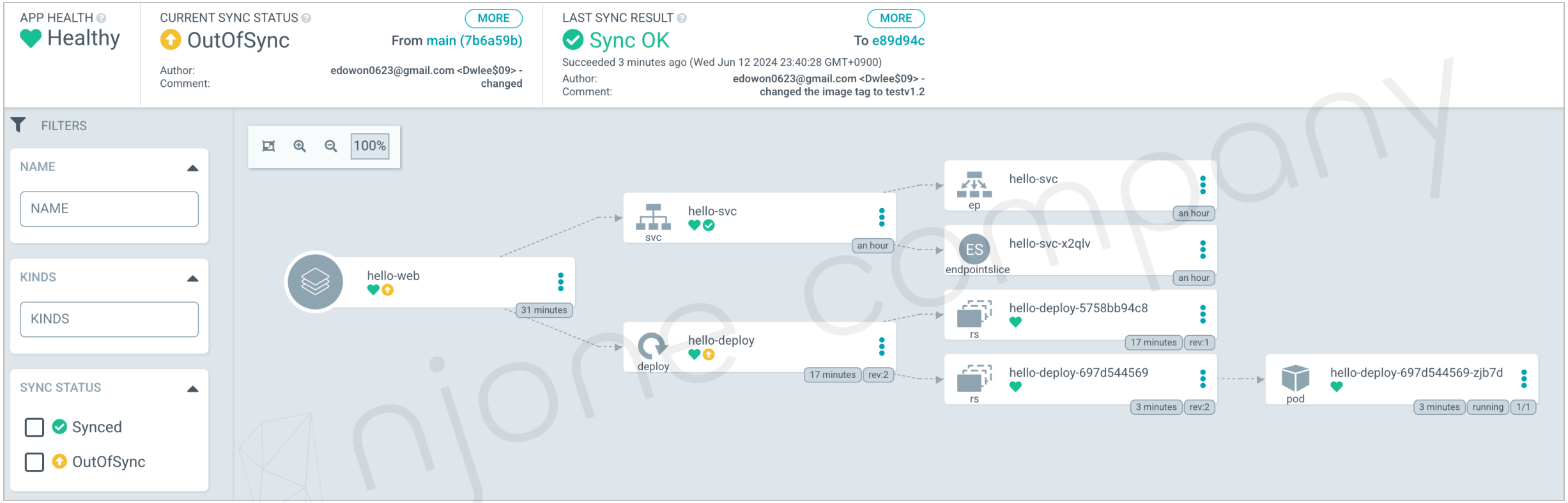
```yaml
spec:
      containers:
      - name: hello-kubernetes
        image: 192.168.0.41/devops/cicd-web-project:10
        imagePullPolicy: IfNotPresent
        ports:
        - name: http
          containerPort: 8080
      imagePullSecrets:
      - name: regcred
```

- ArgoCD sync
  - deployment.yml 변경 사항을 Git에 push 후 ArgoCD Sync 실행

```
$ curl -k -L -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJhcmdvY2QiLCJ..." \
        -X POST https://192.168.0.41:30674/api/v1/applications/hello-web/sync
```

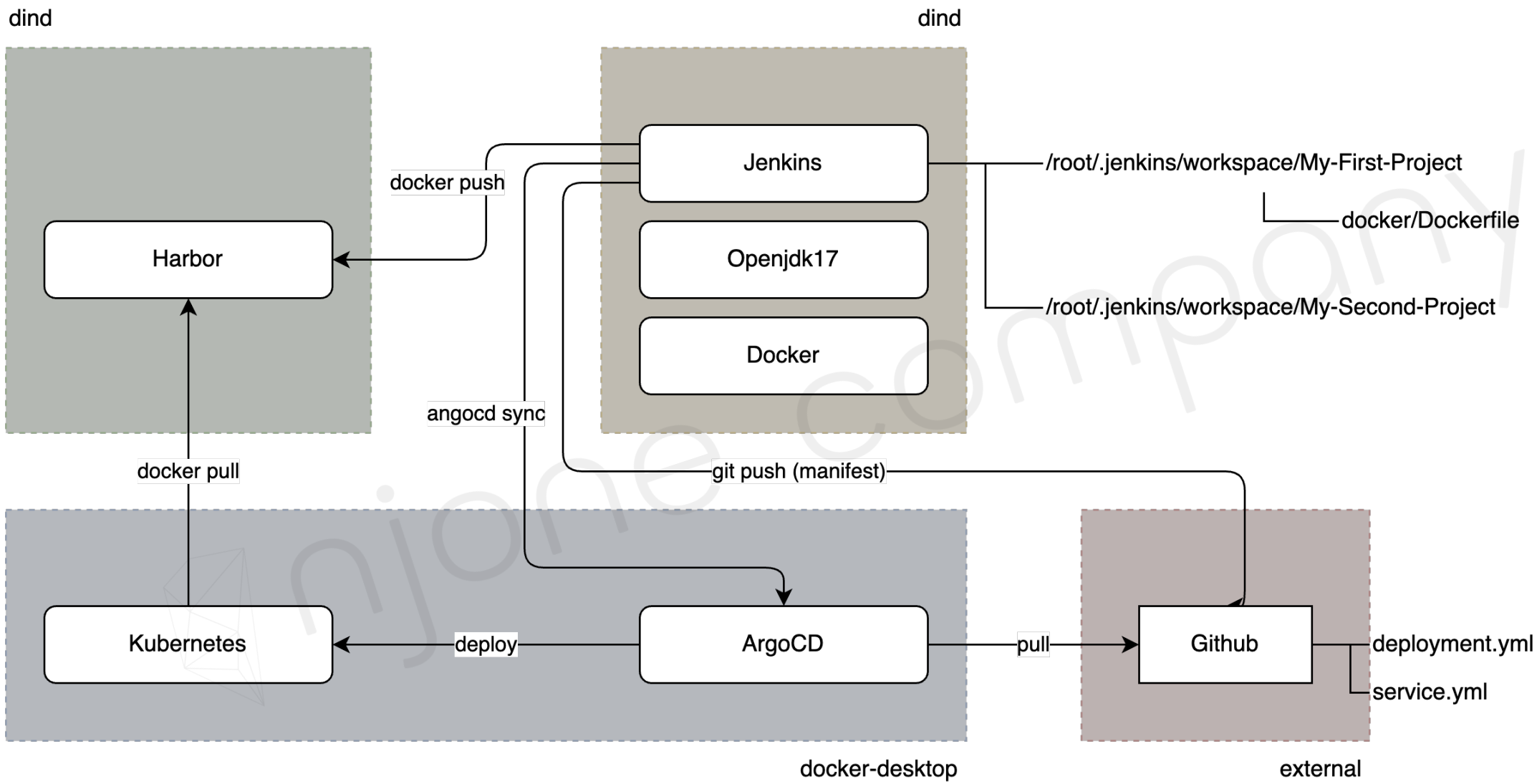
K8s + ArgoCD 연동

- ArgoCD sync



Docker를 활용한 자동화 빌드 시스템 구축

CI/CD Pipeline



Appendix

K8s + Kustomize 연동

- Kustomize → Manifest 파일 변경 자동 감지
  - https://kubernetes.io/ko/docs/tasks/manage-kubernetes-objects/kustomization/

- Kustomize CLI 설치
  - 설치 → https://github.com/kubernetes-sigs/kustomize/releases
  - Windows) choco install kustomize
  - MacOS) brew install kustomize

Appendix

K8s + Kustomize 연동

- Kustomize를 위한 폴더 구성
  - appcd-sample

./base
- deployment.yuml
- *kustomization.yml*
- service.yml

./kustomize1
- application.properties
- *kustomization.yml*

./kustomize2
- deployment.yml
- *kustomization.yml*

./kustomize3
- *kustomization.yml*

./overlay/dev
- *kustomization.yml*

```yaml
resources:
- deployment.yml
- service.yml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
```

```yaml
resources:
- deployment.yml
configMapGenerator:
- name: example-configmap-1
  files:
  - application.properties
```

K8s + Kustomize 연동

- Kustomize 설정 확인
  - kubectl kustomize ./kustomize1 (or ./kustomize)

```
kubectl kustomize ./kustomize1
apiVersion: v1
data:
  application.properties: FOO=Bar
kind: ConfigMap
metadata:
  name: example-configmap-1-tcgd99d22m
```

  - kustomize build .

- Kustomize 적용 (Deployment에 적용)
  - kustomize edit set image 192.168.0.41/devops/cicd-web-project:newest
  - kubectl apply -k ./kustomize1 (or ./kustomize2 or ./kustomize3)

Appendix

K8s + Kustomize + ArgoCD 연동

- Kustomize에서 변경 된 결과를 Git에 반영
  - kubectl kustomize ./kustomize1 (or ./kustomize)

Thank you

- **Section 0: Introduction to Cloud Native Technologies**
- **Section 1: Docker Essentials - Container**
- **Section 2: Docker Essentials - Image**
- **Section 3: Docker Network and Storage**
- **Section 4: Building and Managing Containerized Application**
- **Section 5: Container Orchestration**
- **Section 6: Continuous Integration and Continuous Deployment**
- **Section 7: Docker Security**
- **Section 8: Logging and Monitoring**
- **Section 9: Advanced Docker Usage**
- **Section 10: Capstone Project**

Thank you