from Beginner to Master

DevOps를 위한 DocKer 가상화

Section 5.

CI/CD

- Deploy Web Application
- Docker를 활용한 자동화 빌드 시스템 구축

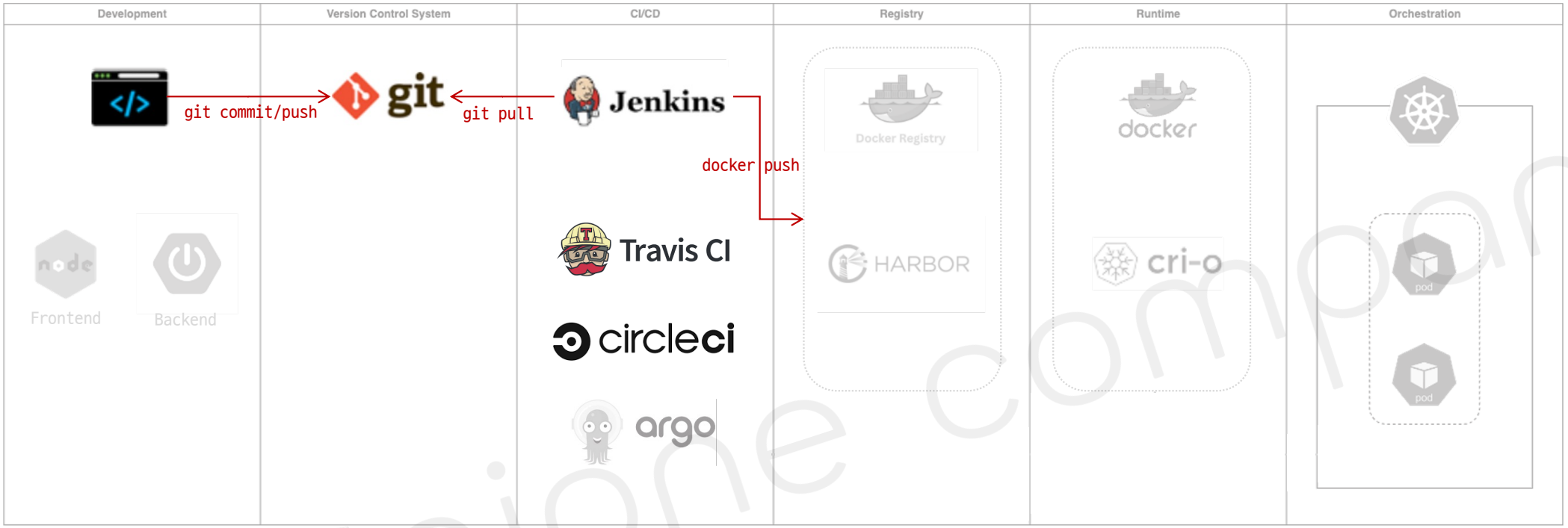
Docker를 활용한 자동화 빌드 시스템 구축

CI/CD

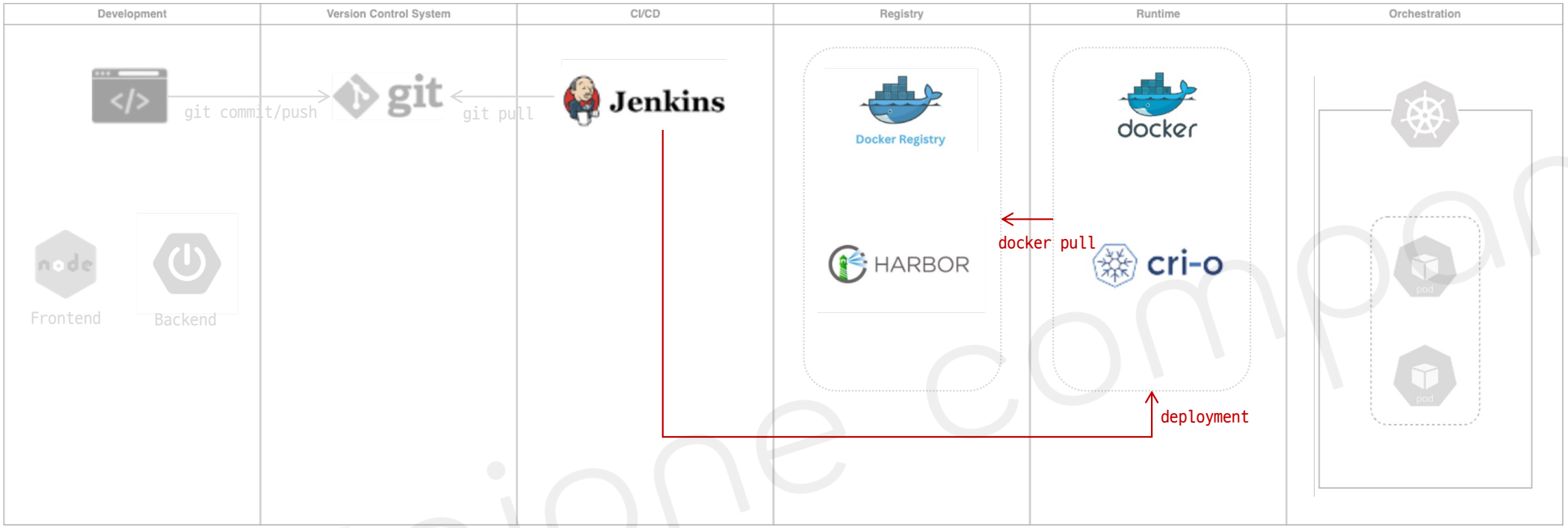
Development	Version Control System	CI/CD	Registry	Runtime	Orchestration
Frontend Backend	git	Jenkins Travis CI circleci argo	Docker Registry HARBOR	docker cri-o podman	pod pod

Docker를 활용한 자동화 빌드 시스템 구축

CI(Continuous Integration)

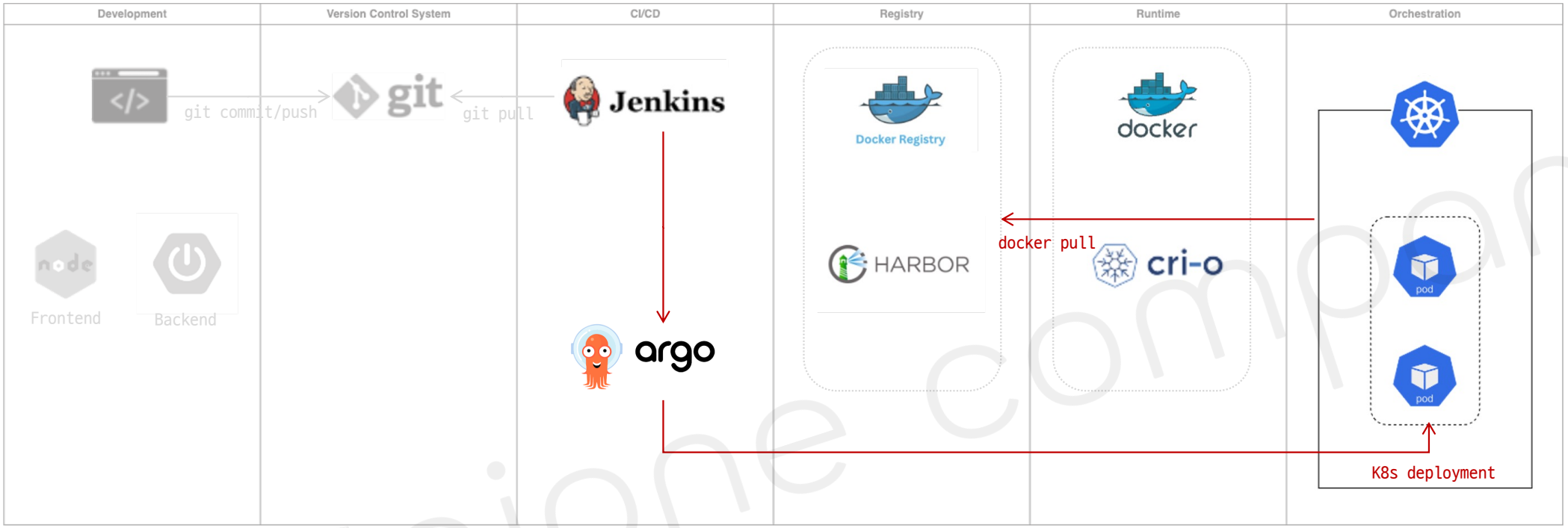


CD(Continuous Deployment) → Container runtime



Docker를 활용한 자동화 빌드 시스템 구축

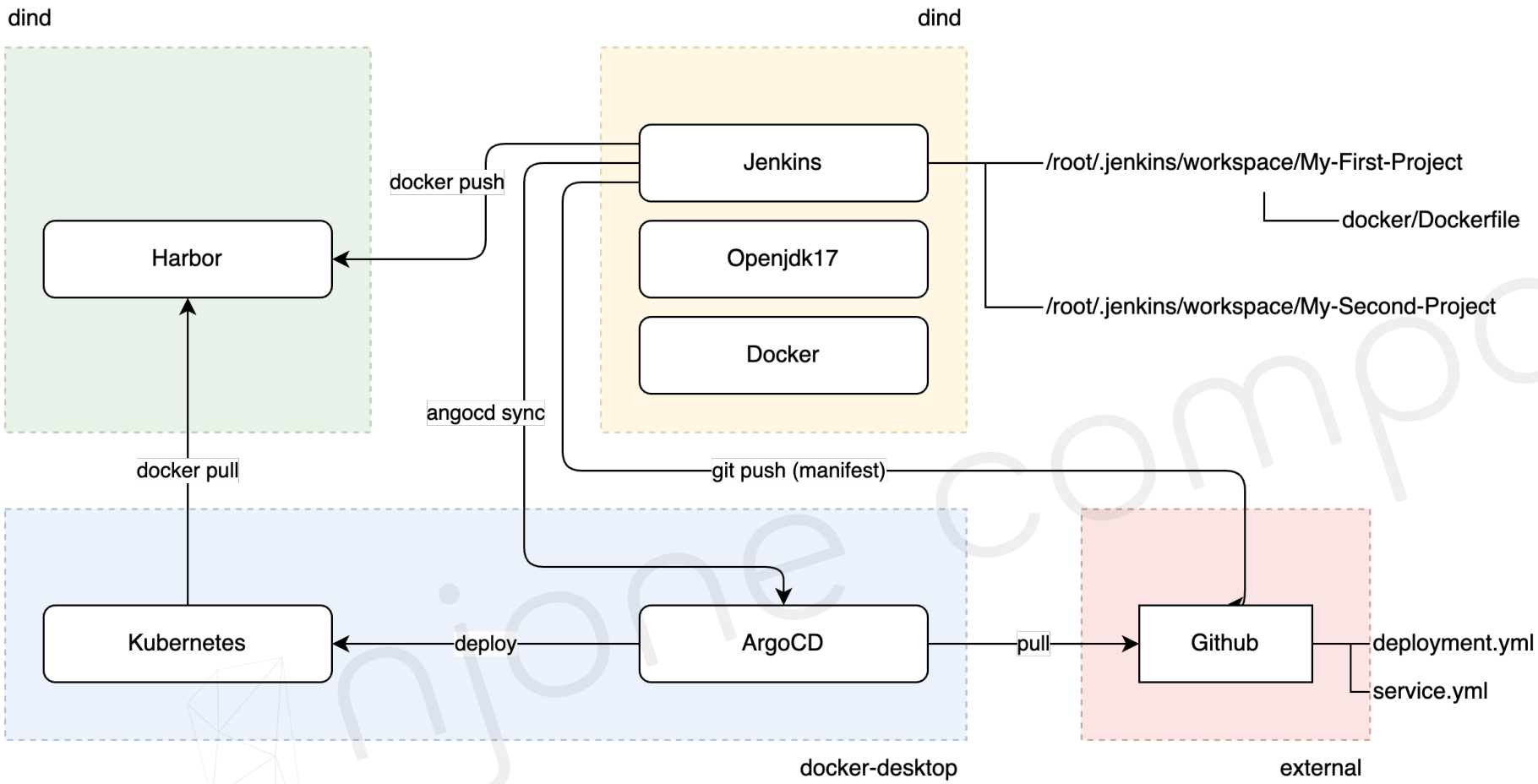
CD(Continuous Deployment) → Kubernetes + Argocd



Docker를 활용한 자동화 빌드 시스템 구축

njone company

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’
                }
            }
}
```

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)

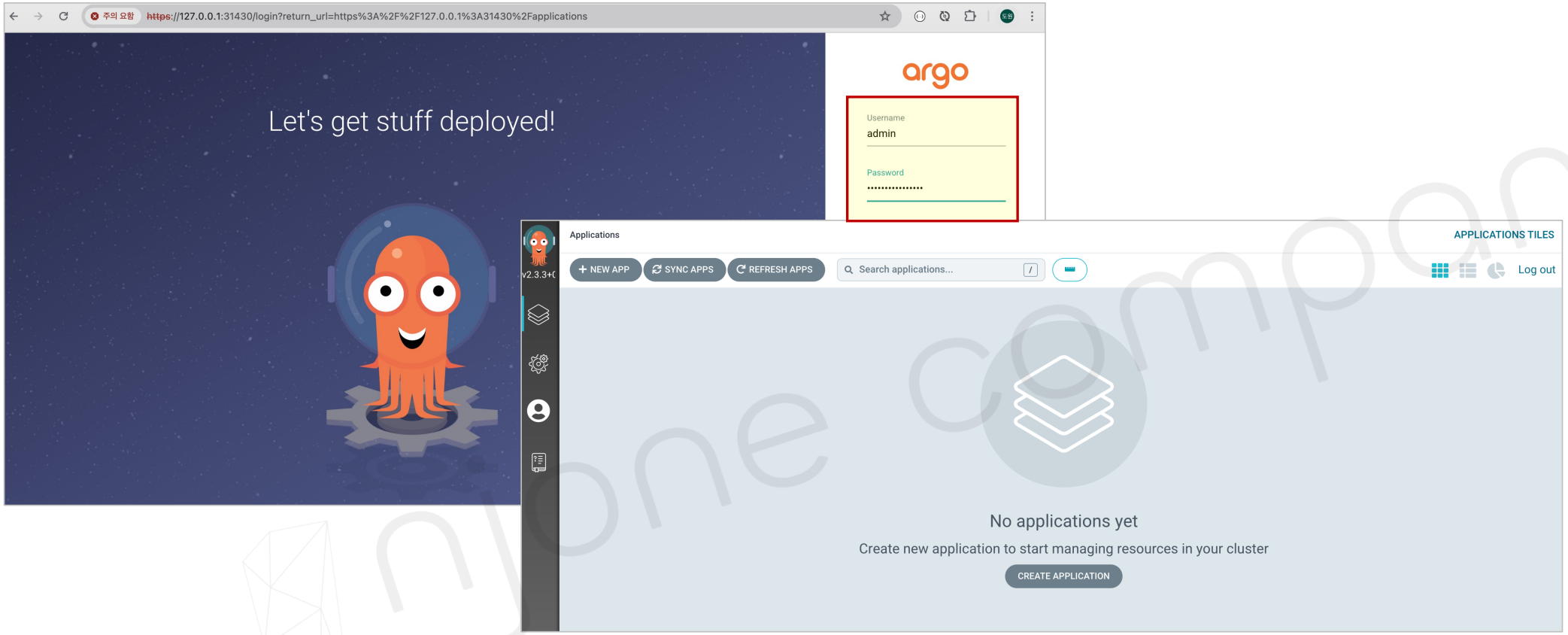
```
$ kubectl get secret argocd-initial-admin-secret -n argocd -o “jsonpath={.data.password}”
ABCSEDFGHKKSKQL… → encode.b64로 저장
$ certutil -decode encode.b64 decode.txt
```

  - admin 로그인 암호 확인 (방법3)

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

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

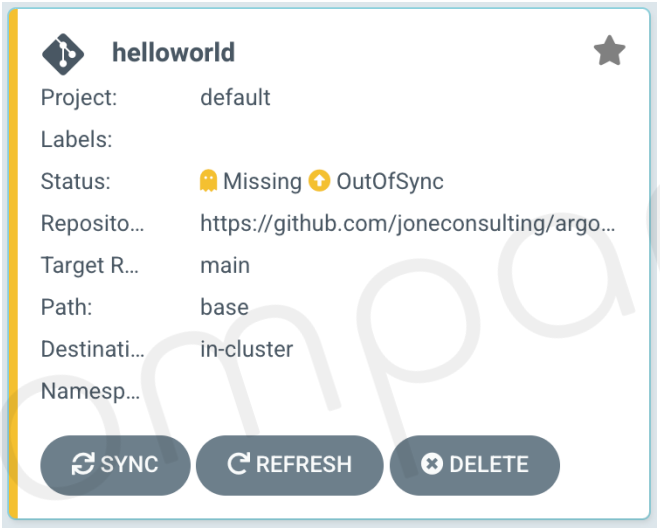
```
data:
    accounts.admin: apiKey
```

  - curl 명령어로 applications 확인

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

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
```

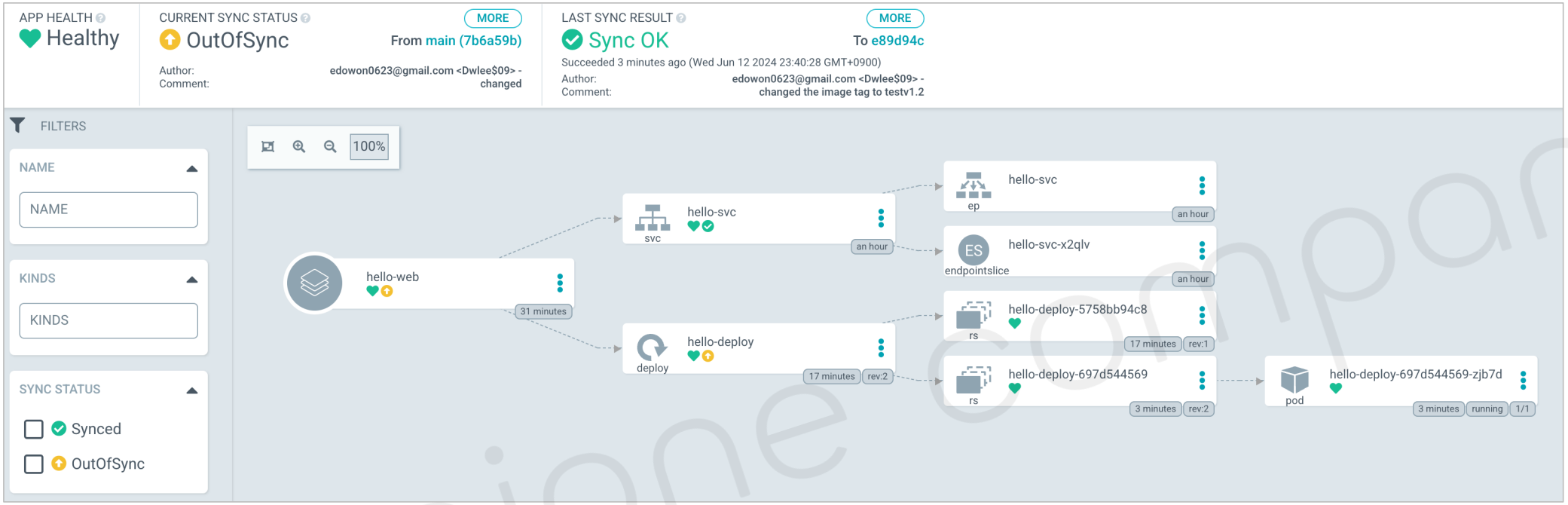
- ArgoCD sync

  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

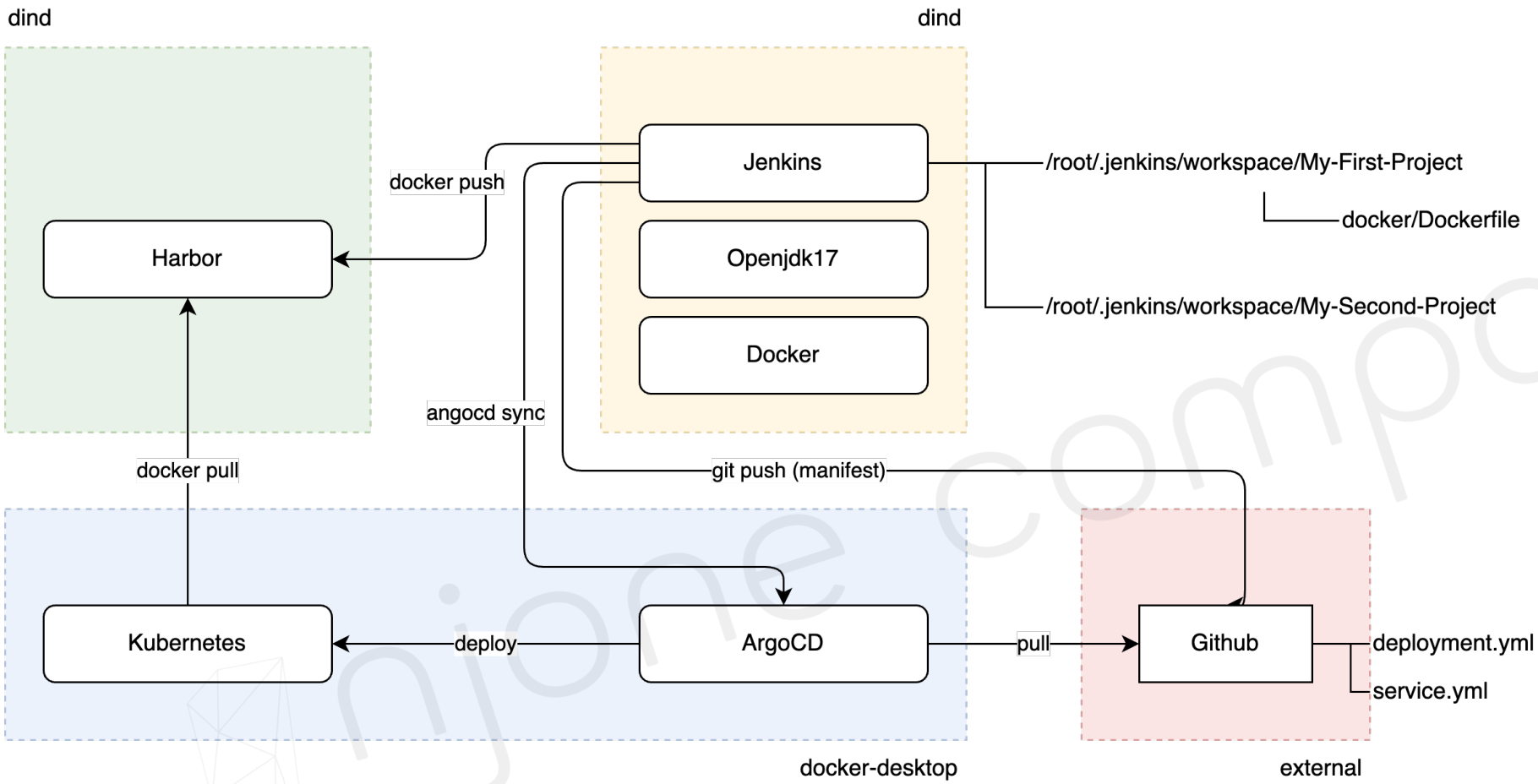
- ArgoCD sync



Docker를 활용한 자동화 빌드 시스템 구축

njone company

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

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)

K8s + Kustomize + ArgoCD 연동

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