K8s + ArgoCD 연동

- ArgoCD 설치
  - argocd cli 설치 (Windows 기준, 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
```

  - argocd cli 설치 (MacOS 기준, homebrew 사용)

```
$ brew install argocd
```

K8s + ArgoCD 연동

- ArgoCD 설치
  - CMD에서 argocd 설치 확인
  - admin 로그인 암호 확인 (방법1)

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

  - admin 로그인 암호 확인 (방법2, Windows)

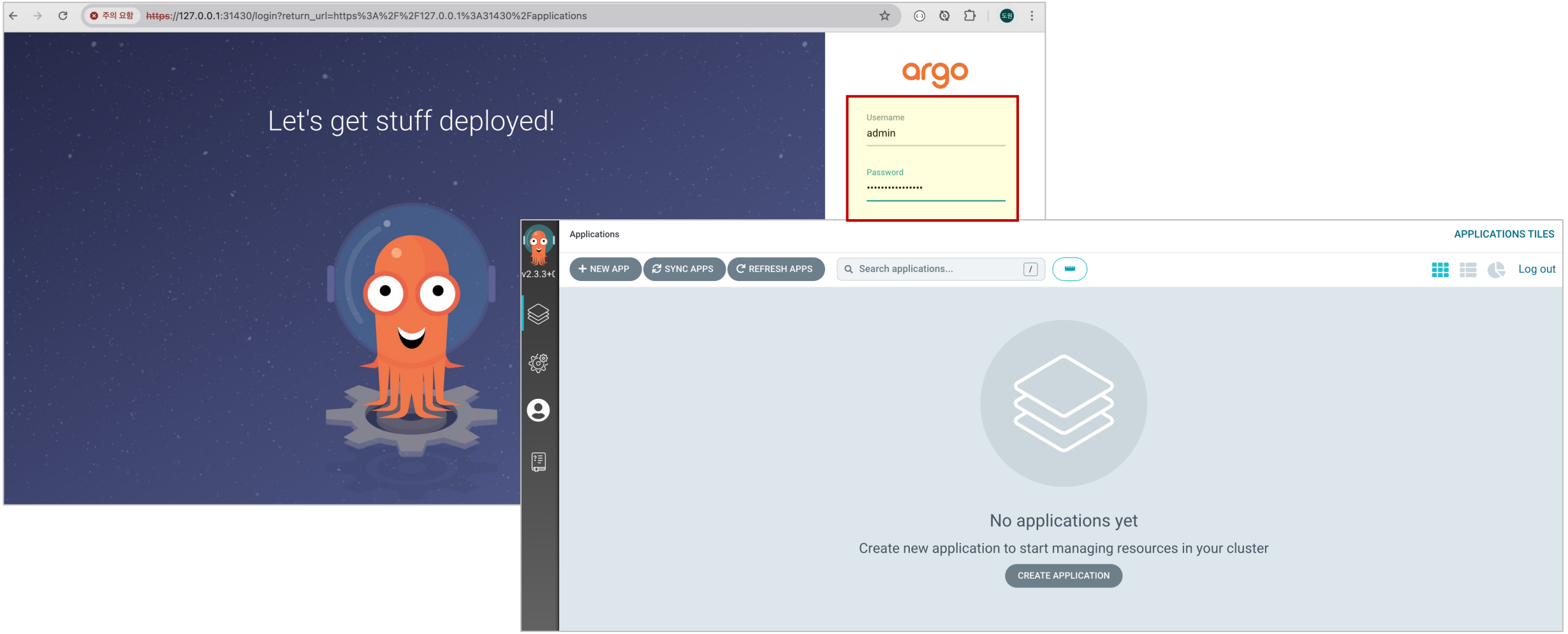
```
$ kubectl get secret argocd-initial-admin-secret –n argocd –o jsonpath="{.data.password}" → encode.b64로 저장
$ certutil -decode encode.b64 decode.txt
```

  - admin 로그인 암호 확인 (방법3, MacOS)

```
$ kubectl get secret argocd-initial-admin-secret –n argocd –o jsonpath="{.data.password}" |
```

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..." ₩
```

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

K8s + ArgoCD 연동

- deployment.yml 파일에 imagePullSecrets 항목 추가

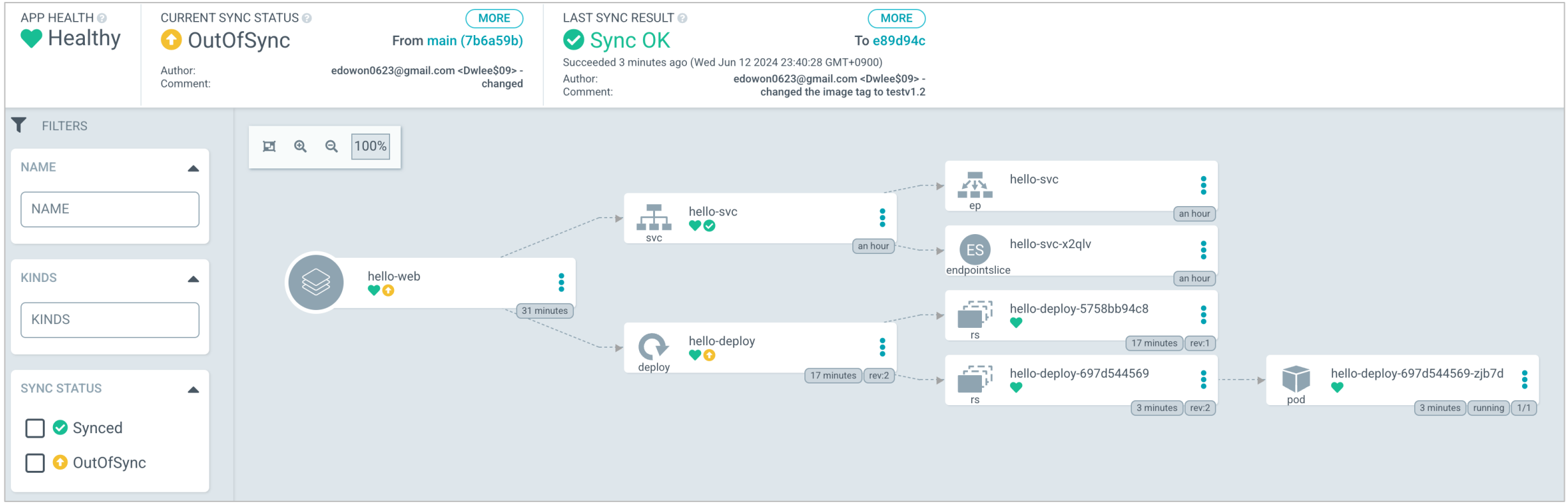
```
spec:
      containers:
      - name: hello-kubernetes
        image: 192.168.0.41/devops/cicd-web-project:10
        imagePullPolicy: IfNotPresent
        ports:
        - name: http
          containerPort: 8080
      imagePullSecrets:
      - name: regcred
```

- ArgoCD sync
  - deployment.yml 변경 사항을 git에 PUSH 후 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



K8s + Kustomize 연동

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

```
./base
        deployment.yuml
        kustomization.yml
        service.yml
./kustomize1
        application.properties
        kustomization.yml
./kustomize2
        deployment.yml
        kustomization.yml
./kustomize3
        kustomization.yml
```

K8s + Kustomize 연동

- Kustomize 설정 확인
  - kubectl kustomize ./kustomize1 (or ./kustomize2 or ./kustomize3)

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

- Kustomize 적용
  - kubectl apply -k ./kustomize1 (or ./kustomize2 or ./kustomize3)

K8s + Kustomize 연동

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

- Kustomize 사용
  - 이미지 태그 적용

```
$ cd ./kustomize3
$ kustomize edit set image 192.168.0.41/devops/cicd-web-project:newest
$ kubectl apply -k .
```