

# 10. Go를 활용한 Kubernetes CLI 개발

## 01 Go를 활용한 Kubernetes CLI 개발 소개

# 챕터 소개

## 01. Go를 활용한 Kubernetes CLI 개발 소개

1. Go를 활용한 Kubernetes CLI 개발 소개
2. Go언어 필수 문법 소개
3. Kubernetes Custom CLI 설계
4. [실습] Kubernetes API 활용 모듈 구현
5. [실습] Kubernetes CLI 빌드 및 실행

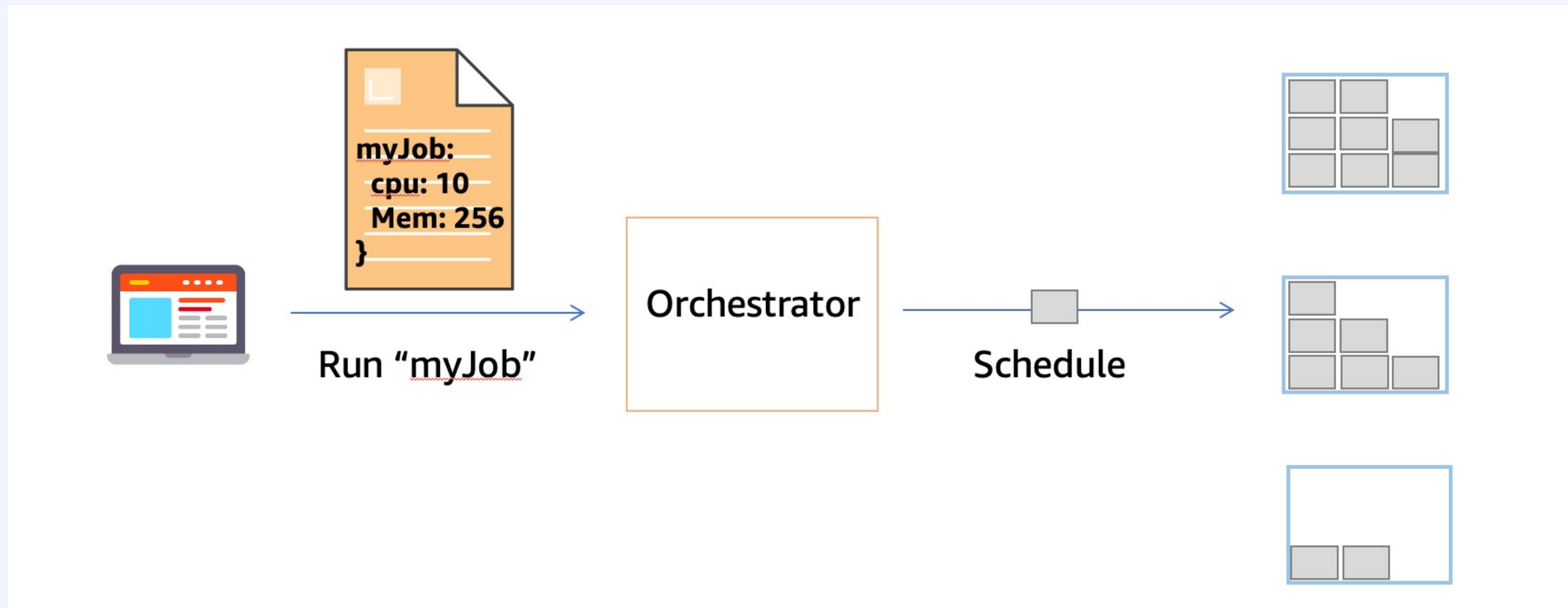
# 소개 순서

## 01. Go를 활용한 Kubernetes CLI 개발 소개

1. Kubernetes **Life Cycle** 적용 방법
2. **YAML**방식의 장점과 단점
3. Go를 활용한 Kubernetes **CLI** 개발 흐름
4. **CDK8s** 소개
5. CDK8s **특징**
6. CDK8s **워크플로우**

# 1. Kubernetes Life Cycle 적용 방법

- 선언적 프로그래밍(명세)가 가능한 YAML 파일로 작성 후 적용
- 적용하고자 하는 **Desired State**(원하는 상태)를 명시해 적용
- kubectl 이나 helm, kustomize 등 **Kubernetes CLI**를 통해 적용



출처 - <https://aws.amazon.com/ko/blogs/korea/using-cdk8s-for-kubernetes-applications/>

## 2. YAML 방식의 장점과 단점

01.  
Go를 활용한  
Kubernetes  
CLI 개발 소개

### 장점

- 사람이 읽기 쉬운 형태
- 어디서든 사용가능
- 선언적 프로그래밍
- 원하는 상태로 배포/관리
- Static한 파일 명세

YAML  
(K8s Manifest,  
Helm Charts,  
Kustomize)

### 단점

- 다수 자원, 환경일 경우  
중복 부분 처리 필요
- 별도 툴을 이용해 관리
- 코드가 기존 언어와 상이
- 코드 업데이트가 복잡함

### 3. Go를 활용한 Kubernetes CLI 개발 흐름

**01.**  
Go를 활용한  
Kubernetes  
CLI 개발 소개

#### 개발자 요구사항

- Go 개발자에게 친숙하고 특화
- 동적인 Go 프로그래밍 언어 지원
- CLI 필요시 단순한 명령어만 사용
- 개발 도구/워크플로우를 그대로 사용

#### 요구사항 충족 툴 (Kubernetes CLI)

#### CDK8s

- AWS EKS 전용 CLI
- CLI 개발후 빌드/컴파일 불필요
- EKS에 배포할 자원을 go로 작성
- EKS에 적용할 API를 go로 작성

## 4. CDK8s 소개 #1

01.  
Go를 활용한  
Kubernetes  
CLI 개발 소개

- CDK8s란, 개발자 친화적인 **프로그래밍 언어**와 풍부한 **객체지향 API**를 이용하여 Kubernetes 애플리케이션과 재사용 가능한 추상화를 정의할 수 있는 소프트웨어 개발 **프레임워크**
- CDK8s를 통해 각 언어에서 구성하고자 하는 Kubernetes 리소스를 **작성**하고, **배포**할 수 있는 **YAML 파일 생성**가능
- 중앙에서 모든 **Addon** 및 Kubernetes Cluster **자원 버전/설정** 관리
- **Git**을 통한 **버전** 관리 및 **확장** 가능



## 4. CDK8s 소개 #2

# 01.

Go를 활용한  
Kubernetes  
CLI 개발 소개

The screenshot shows an IDE with two files open. The left file, `main.go`, contains Go code for creating a CDK8s chart and deploying it. The right file, `web-text-box.k8s.yaml`, is the Kubernetes manifest generated by the code.

```

13
14 func NewMyChart(scope constructs.Construct, id string, props *MyChartProps) cdk8s.Chart {
15     var cprops cdk8s.ChartProps
16     if props != nil {
17         cprops = props.ChartProps
18     }
19     chart := cdk8s.NewChart(scope, jsii.String(id), &cprops)
20
21     label := map[string]*string{"app": jsii.String("web-text-box")}
22
23     k8s.NewKubeService(chart, jsii.String("service"), &k8s.KubeServiceProps{
24         Spec: &k8s.ServiceSpec{
25             Type: jsii.String("LoadBalancer"),
26             Ports: &[]*k8s.ServicePort{{
27                 Port: jsii.Number(80),
28                 TargetPort: k8s.IntOrString_FromNumber(jsii.Number(8080)),
29             }},
30             Selector: &label,
31         },
32     })
33
34     k8s.NewKubeDeployment(chart, jsii.String("deployment"), &k8s.KubeDeploymentProps{
35         Spec: &k8s.DeploymentSpec{
36             Replicas: jsii.Number(2),
37             Selector: &k8s.LabelSelector{
38                 MatchLabels: &label,
39             },
40             Template: &k8s.PodTemplateSpec{
41                 Metadata: &k8s.ObjectMeta{
42                     Labels: &label,
43                 },
44                 Spec: &k8s.PodSpec{
45                     Containers: &[]*k8s.Container{{
46                         Name: jsii.String("web-text-box"),
47                         Image: jsii.String("347880001135.dkr.ecr.ap-northeast-2.amazonaws.com/web-text-box:1.0"),
48                         Ports: &[]*k8s.ContainerPort{{ContainerPort: jsii.Number(8080)}},
49                     }},
50                 },
51             },
52         },
53     })
54
55

```

```

1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: web-text-box-service
5 spec:
6   ports:
7     - port: 80
8     targetPort: 8080
9   selector:
10     app: web-text-box
11   type: LoadBalancer
12 ---
13 apiVersion: apps/v1
14 kind: Deployment
15 metadata:
16   name: web-text-box-deployment
17 spec:
18   replicas: 2
19   selector:
20     matchLabels:
21       app: web-text-box
22   template:
23     metadata:
24       labels:
25         app: web-text-box
26     spec:
27       containers:
28         - image: 347880001135.dkr.ecr.ap-northeast-2.amazonaws.com/web-text-box:1.0
29         name: web-text-box
30         ports:
31         - containerPort: 8080
32

```



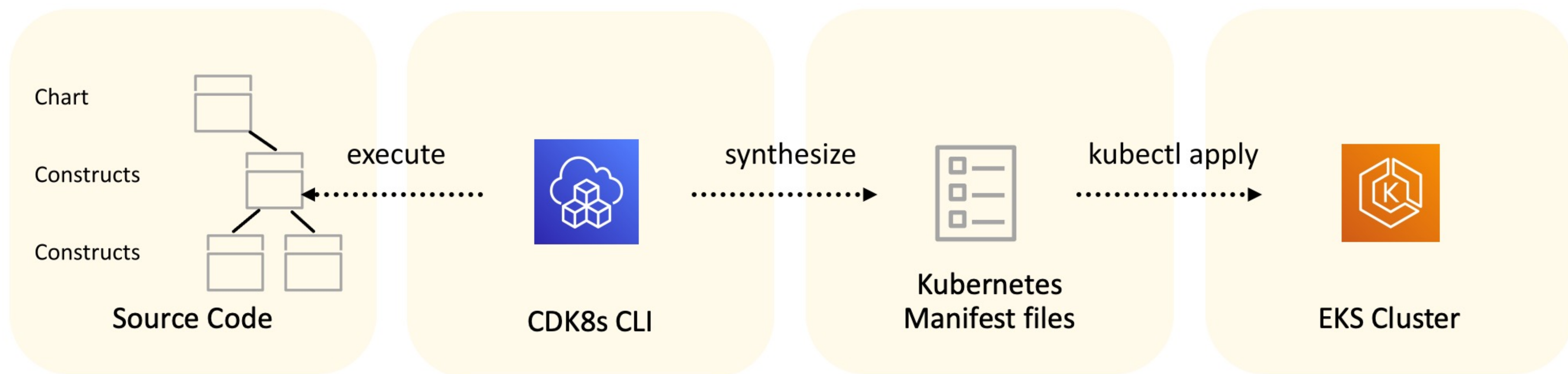
## 5. CDK8s 특징

**01.**  
Go를 활용한  
Kubernetes  
CLI 개발 소개

특징	상세내용
K8s App 정의 용이	<ul style="list-style-type: none"> <li>익숙한 <b>프로그래밍 언어</b>로 Kubernetes 애플리케이션을 <b>정의</b></li> <li><b>Go</b>, Java, Python, Typescript</li> </ul>
모든 K8s 환경에서 실행	<ul style="list-style-type: none"> <li><b>로컬 환경</b>에서 실행 가능 및 <b>YAML 파일 생성</b>후 K8s 클러스터에 배포 가능</li> <li>Public 및 On-Premise 클라우드에서도 <b>동일한 코드로 표준화</b> 가능</li> </ul>
코드 라이브러리 작성 및 공유	<ul style="list-style-type: none"> <li>템플릿보다 관리가 용이한 <b>라이브러리 형태</b>로 작성 및 공유 가능</li> <li>K8s 애플리케이션 <b>정의</b>를 위한 라이브러리를 <b>표준화 및 재사용</b> 가능</li> </ul>
App 워크플로우 단순화	<ul style="list-style-type: none"> <li>애플리케이션을 개발하는 것과 <b>동일한 툴</b>을 사용해 <b>워크플로우</b> 정의</li> <li>작성한 코드를 <b>CI/CD Pipeline</b>을 통해 K8s에 자동화된 <b>배포</b> 가능</li> </ul>

## 6. CDK8s 워크플로우

- 개발자는 자기가 원하는 언어로 **코드를 작성**하고, **CDK8s CLI**를 통해서 코드를 쿠버네티스 매니페스트 **YAML 파일로 전환**이 가능
- 전환된 **YAML**은 **kubectl CLI**를 통해서 kubernetes Cluster에 **배포**하거나, **GitOps Repository**를 활용해 **버전/형상 관리** 및 배포 가능



출처 - <https://aws.amazon.com/ko/blogs/korea/using-cdk8s-for-kubernetes-applications/>