10. Go를 활용한 Kubernetes CLI 개발

02 Go언어 필수 문법 소개

소개 내용

1. Go언어 소개
2. Go언어 설치
3. Go언어 필수 문법1 - 함수
4. Go언어 필수 문법2 - 구조체
5. Go언어 필수 문법3 – 패키지
6. Go언어 필수 문법4 – 메서드
7. go.mod 및 go.sum 파일 소개

1. Go언어 소개

Go는 2009년 구글에서 일하는 로버트 그리즈머, 롭 파이크, 켄 톰프슨이 개발한 프로그래밍 언어이다.
가비지 컬렉션 기능이 있고, 병행성(concurrent)을 잘 지원하는 컴파일 언어다. 구문이 C와 비슷하지만 메모리 보안, 가비지 컬렉션, 구조 타이핑, CSP 스타일 병행성을 제공한다.
이 언어는 golang.org이라는 도메인 이름 때문에 종종 고랭(Golang)으로도 호칭되지만 정확한 명칭은 Go이다.
2개의 주요 구현체가 있다. 하나는 구글의 셀프 호스팅 컴파일러 툴체인으로서 여러 운영 체제, 모바일 장치, 웹어셈블리를 대상으로 한다. 나머지 하나는 GCC 프론트엔드인 gccgo가 있다.
서드파티 트랜스파일러 GopherJS는 프론트엔드 웹 개발을 위해 Go를 자바스크립트로 컴파일한다.

출처 - https://ko.wikipedia.org/wiki/Go_(%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D_%EC%96%B8%EC%96%B4)

2. Go언어 설치 #1

(1) 설치 파일 다운로드 URL

https://go.dev/doc/install

(2) MacOS 및 Windows 설치 방법

- 설치파일 다운로드후 실행해서 설치

2. Go언어 설치 #1

(3) Linux설치 방법

- 다운로드된 파일을 /usr/local/go에 압축을 품

$ tar -C /usr/local -xzf <다운로드 받은 go 압축파일>

- PATH 환경 변수 추가 방법

  > /etc/profile 혹은 $HOME/.bash_profile에 다음의 구문을 추가

  PATH=$PATH:/usr/local/go/bin

3. Go언어 필수 문법1 – 함수 #1

- 함수(func)는 여러 문장을 묶어서 실행하는 코드 블럭의 단위
- Go에서 함수는 func 키워드를 사용하여 정의
- func 뒤에 함수명을 적고, 괄호 ( ) 안에 그 함수에 전달하는 파라미터를 명시
- 함수 파라미터는 0개 이상 사용할 수 있는데, 각 파라미터는 파라미터명 뒤에 int, string 등의 파라미터 타입을 적어서 정의
- 함수의 리턴 타입은 파라미터 괄호 ( ) 뒤에 적게 되는데, 이는 C와 같은 다른 언어에서 리턴 타입을 함수명 앞에 쓰는 것과는 다른 형태임
- 함수는 패키지 안에 정의되며, 호출되는 함수가 호출하는 함수의 반드시 앞에 위치해야 할 필요는 없음

3. Go언어 필수 문법1 – 함수 #2

```
func NewMyChart(scope constructs.Construct, id string, props *MyChartProps) cdk8s.Chart {
  var cprops cdk8s.ChartProps
  if props != nil {
    cprops = props.ChartProps
  }
  chart := cdk8s.NewChart(scope, jsii.String(id), &cprops)

.. 중략 ..

  return chart
}


func main() {
  app := cdk8s.NewApp(nil)
  NewMyChart(app, "hello", nil)
  app.Synth()
}
```

4. Go언어 필수 문법2 - 구조체 #1

- Go에서 구조체(struct)는 Custom Data Type을 표현하는데 사용되며, 필드들의 집합체로 사용
- 구조체는 필드 데이타만을 가지며, 행위를 표현하는 메서드를 갖지 않음
- Go 언어는 객체지향 프로그래밍(Object Oriented Programming, OOP)을 고유의 방식으로 지원하기 때문에 클래스, 객체, 상속 개념이 없음
- OOP의 클래스(class)는 Go 언어에서 Custom 타입을 정의하는 struct로 표현
- 전통적인 OOP의 클래스가 필드와 메서드를 함께 갖는 것과 달리 Go 언어의 구조체는 필드만을 가지며, 메서드는 별도로 분리하여 정의

4. Go언어 필수 문법2 - 구조체 #2

```
type MyChartProps struct {
  cdk8s.ChartProps
}
```

5. Go언어 필수 문법3 – 패키지 #1

- Go는 패키지(Package)를 통해 코드의 모듈화, 코드의 재사용 기능을 제공
- Go는 패키지를 사용해서 작은 단위의 컴포넌트를 작성하고, 이러한 작은 패키지들을 활용해서 프로그램을 작성할 것을 권장
- Go는 실제 프로그램 개발에 필요한 많은 패키지들을 표준 라이브러리로 제공
- main 패키지인 경우, 컴파일러는 해당 패키지를 공유 라이브러리가 아닌 실행(executable) 프로그램으로 만듬
- main 패키지 안의 main() 함수가 프로그램의 시작점(EntryPoint)가 된다. 패키지를 공유 라이브러리로 만들 때에는, main 패키지나 main 함수를 사용해서는 안됨
- import는 다른 패키지를 프로그램에서 사용하기 위해서는 패키지를 포함시키는 방법

5. Go언어 필수 문법3 – 패키지 #2

```
package main


import (
  "example.com/hello-k8s/imports/k8s"
  "github.com/aws/constructs-go/constructs/v3"
  "github.com/aws/jsii-runtime-go"
  "github.com/cdk8s-team/cdk8s-core-go/cdk8s"
)

.. 중략 ..

func main() {
  app := cdk8s.NewApp(nil)
  NewMyChart(app, "hello", nil)
  app.Synth()
}
```

6. Go언어 필수 문법4 – 메서드 #1

- 타 언어의 OOP의 클래스가 필드와 메서드를 함께 갖는 것과 달리 Go 언어에서는 struct가 필드만을 가지며, 메서드는 별도로 분리되어 정의
- Go 메서드는 특별한 형태의 func 함수임
- 메서드는 함수 정의에서 func 키워드와 함수명 사이에 "그 함수가 어떤 struct를 위한 메서드인지"를 표시하며, 흔히 receiver로 불리우는 이 부분은 메서드가 속한 struct 타입과 struct 변수명을 지정
- struct 변수명은 함수 내에서 마치 입력 파라미터처럼 사용

6. Go언어 필수 문법4 – 메서드 #2


```
import (
  "example.com/hello-k8s/imports/k8s"
  "github.com/aws/constructs-go/constructs/v3"
  "github.com/aws/jsii-runtime-go"
  "github.com/cdk8s-team/cdk8s-core-go/cdk8s"
)

type MyChartProps struct {
  cdk8s.ChartProps
}

func NewMyChart(scope constructs.Construct, id string, props *MyChartProps) cdk8s.Chart {
  var cprops cdk8s.ChartProps
  if props != nil {
    cprops = props.ChartProps
  }
  chart := cdk8s.NewChart(scope, jsii.String(id), &cprops)

  label := map[string]*string{"app": jsii.String("hello-k8s")}

  k8s.NewKubeDeployment(chart, jsii.String("deployment"), &k8s.KubeDeploymentProps{
    Spec: &k8s.DeploymentSpec{
      Replicas: jsii.Number(2),
      Selector: &k8s.LabelSelector{
        MatchLabels: &label,
      },
      Template: &k8s.PodTemplateSpec{
        Metadata: &k8s.ObjectMeta{
          Labels: &label,
        },
        Spec: &k8s.PodSpec{
          Containers: &[]*k8s.Container{{
            Name:  jsii.String("hello-kubernetes"),
            Image: jsii.String("paulbouwer/hello-kubernetes:1.7"),
            Ports: &[]*k8s.ContainerPort{{ContainerPort: jsii.Number(8080)}},
          }},
        },
      },
    },
  })
  return chart
}
```

7. go.mod 및 go.sum 파일 소개

- go.mod 파일은 모듈 즉, 패키지(Package)의 모음으로, 한 개의 모듈은 다수의 패키지를 포함
- go.mod 내 명시된 모듈을 통해 Go언어는 패키지들의 종속성(Dependency)를 관리할 수 있으며, 모듈은 패키지 관리 기반으로 활용
- 모듈은 패키지를 트리 형식으로 관리하며, Go프로젝트 root 디렉토리에 go.mod 파일을 생성해 모듈을 정의하고, 종속성 정보를 관리
- go.sum 파일은 go.mod로 받은 패키지의 무결성(중복 방지)를 위해 자동 생성
- go.sum으로 Go프로젝트 재실행시 모든 패키지를 다시 설치하지 않음