



# Piscine Golang - Day04

🕒 Created @October 24, 2021 11:28 PM

요약: 이 문서는 Golang의 Concurrency, Synchronization 내용을 담고 있습니다.

서브젝트 오류 문의 : bigpel66@icloud.com

## Contents

- III*      Exercise 00 : Hello Server!
- IV*      Exercise 01 : My First Server
- V*      Exercise 02 : 42 Chatting

# Chapter I - General Rules

- 모든 Exercise들은 Golang 1.17로 진행하며 테스트 코드와 함께 작성되어야 합니다. 작성된 테스트 코드들은 Exercise와 함께 제출되어야 합니다.
- 제출된 테스트 코드 내에 테스트 케이스는 반드시 존재해야 합니다.
- 적극적인 테스트 케이스는 권장되며, 특히 예외 케이스에 신경써야 합니다.
- 평가자는 자신이 생각했던 테스트 케이스가 존재하지 않는 경우, 피평가자에게 테스트 케이스를 요청할 수 있습니다. 이 때 피평가자의 코드가 동작하지 않으면 평가는 종료됩니다.
- 테스트 진행은 "testing" 패키지를 이용해야 하며, "testing"과 "[github.com/stretchr/testify/assert](https://github.com/stretchr/testify/assert)" 둘 중 하나를 일관성 있게 사용해야 합니다.
- 테스트 진행 시 testing 객체를 이용한 벤치마킹 역시 적극 권장됩니다.
- Exercise의 제출은 동료 평가 및 Moulinette 채점으로 **PASS** 혹은 **FAIL**이 결정됩니다.
- Exercise들은 gofmt를 이용하여 포맷팅이 되어 있어야 합니다. 동료 평가 진행 시, 제출된 Exercise가 포맷팅이 되어 있지 않다면 **FAIL**입니다.
- Exercise들은 golangci-lint를 이용했을 때, 별다른 문제가 없다는 것이 검증되어야 합니다. golangci-lint에서 문제가 있는 부분이 발견되면 **FAIL**입니다.
- 동료 평가 진행을 위한 주석은 적극 허용됩니다. 각 함수 및 변수들에게 적용된 주석은 Golang에서 정식으로 지원하는 godoc을 이용하여 Document로 만들 수 있습니다.
- 컴파일은 **go build**를 이용하며, 이를 위해 **go mod**를 먼저 이해해야 합니다. **go mod**를 이용하지 않고 제출된 Exercise 역시 **FAIL**입니다.
- 제공된 서브젝트를 철저히 파악하여 Golang의 기초 지식을 모두 얻을 수 있도록 하십시오.
- 서브젝트에서 요구하는 내용들이 짧더라도, 이 내용들을 이해하고 예상한대로 작동되도록 시간을 쓰는 것은 굉장히 가치 있는 행동입니다. 다양한 시도를 권장합니다.
- Go ? Ahead ! Write in Go.



go mod를 이용하기 전에 GOPATH, GOROOT를 알아보십시오.

## Chapter II - Preamble

---

구글에서는 15년 이상 대규모 리눅스 컨테이너 환경에서 서비스를 운영하고 있었는데, 이를 관리하기 위한 "컨테이너 오케스트레이션 시스템"으로 구글은 'borg'라는 시스템을 개발해 사용하고 있었다.

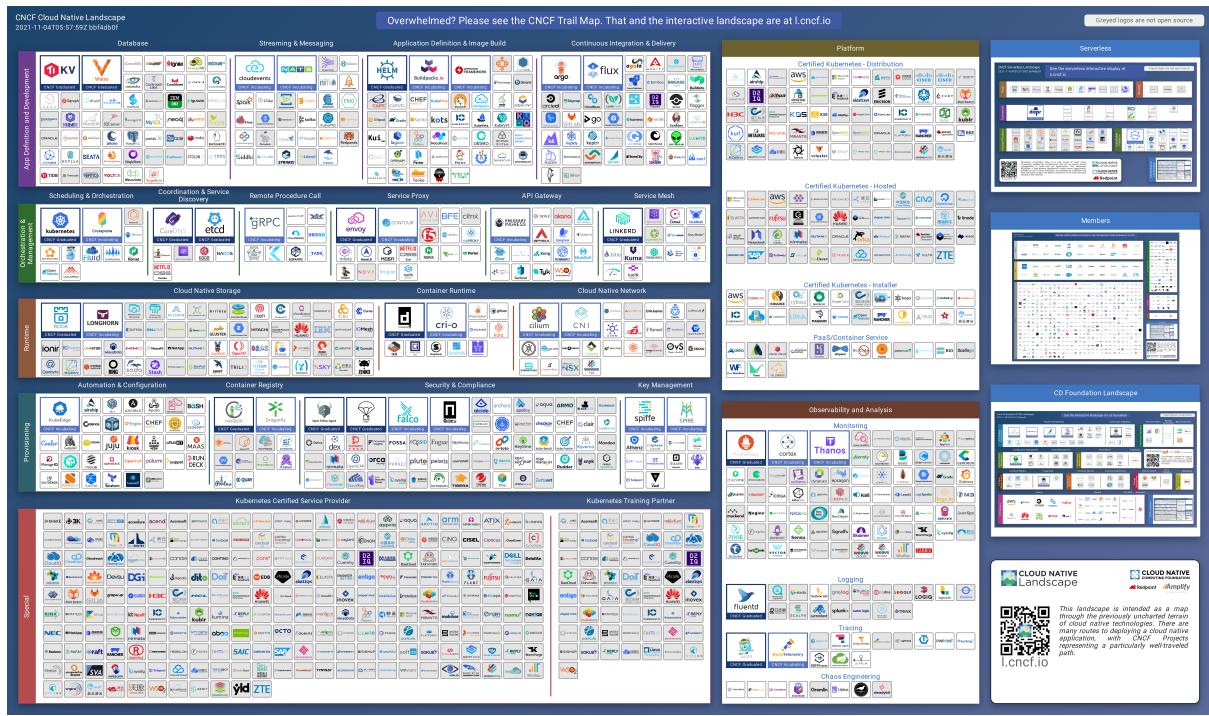
Docker가 출시되고 많은 개발자들이 컨테이너 환경에 익숙해지기 시작하자, 구글은 내부적으로만 사용중이던 borg가 더 많은, 다양한 개발자들이 사용할 수 있을 것이라고 생각했다.

시간이 흘러 2015년 7월 21일, 구글은 borg 기반의 새로운 프로젝트를 출시하였고, 이 프로젝트의 이름을 "쿠버네티스"로 지었다. 출시와 더불어 구글은 더 많은 사용자가 쿠버네티스를 사용하고, 기여할 수 있게 하기 위해서 쿠버네티스를 오픈소스화하기로 결정했다.

구글은 클라우드 네이티브 컴퓨팅 재단(Cloud Native Computing Foundation, CNCF)을 리눅스 재단과 파트너십을 맺으며 설립하였고, 쿠버네티스를 CNCF의 시드 테크놀로지로 기부하게 되었다.

2021년 현재, 많은 개발자들이 컨테이너 환경으로 눈길을 돌리고 있으며, 이에 따라 쿠버네티스는 오픈소스 역사상 리눅스 커널 이후 가장 커다란 오픈소스 프로젝트로 인정받고 있다.

또한 쿠버네티스의 성공 이후 CNCF에서는 containerd, HELM, Prometheus 등의 수많은 클라우드 네이티브 오픈소스 프로젝트들을 적극적으로 지원하여 "클라우드 네이티브 생태계"의 발전을 견고하게 지키고 있다.



CNCF에서 지원하는 클라우드 네이티브 오픈소스 프로젝트들의 landscape

# Chapter III

## Exercise 00 : Hello, Server!

Aa Name	≡ Tags
<u>Turn-in directory</u>	ex00
<u>Files to turn in</u>	server/server.go
<u>Module name</u>	None
<u>Allowed packages</u>	"io" "bufio" "net" "log" "fmt"

서버에 클라이언트 요청이 들어오면 1초마다 42를 반환해주는 서버를 만드십시오. 서버는 8000번 포트를 사용합니다. 이에 따라 아래 그림처럼 결과를 만들어야 합니다. main 함수가 작성되어야 합니다.

```
▶ nc localhost 8000
42
42
42
42
42
42
42
^C
```



nc 명령어에 대해서 알아보십시오.

# Chapter IV

---

## Exercise 01: My 42 Server

Aa Name	Tags
<u>Turn-in directory</u>	ex01
<u>Files to turn in</u>	server/server.go , client/client.go
<u>Module name</u>	None
<u>Allowed packages</u>	"io" "bufio" "fmt" "net" "time" "log" "flag"

이전 문제에서 우리는 처음으로 서버를 만들었습니다. 하지만 서버에는 하나의 요청만 처리할 수 있고, 특정 요청에 반응하는 할 수 없는 문제가 있습니다. 이번에는 서버를 조금 더 업그레이드 해봅시다.

국가코드를 입력받으면 코드에 해당하는 언어로 42를 5번 출력하는 서버 프로그램과 국가코드를 서버에 전달하기 위해 서버와 연결하는 클라이언트 프로그램을 만드십시오. 각 프로그램의 main 함수가 작성되어야 합니다. Error는 적절히 처리하십시오.

최소한 다음 국가들은 포함해야 합니다.

- kr(한국) : `사십이`
- us(미국) : `forty-two`
- jp(일본) : `よんじゅうに`
- cn(중국) : `四十二`

출력 메시지는 1초 간격으로 출력되어야 하며, 서버는 여러 개의 클라이언트가 동시에 접속하더라도 각 클라이언트마다 요청을 잘 처리해야 합니다. 만일 알수 없는 국가코드가 입력될 경우 Unknown code 라는 메시지를 출력해야 합니다.

```
▶ ./client
fr
Unknown code
Unknown code
Unknown code
Unknown code
Unknown code
```

클라이언트는 출력 중에도 입력이 가능해야 합니다. 입력 시 이전 출력과 함께 출력되어야 합니다. 이전 출력이 끝나고나서 입력이 출력되면 안 됩니다. 다음은 출력 예시입니다.

```
▶ ./client
kr
사십이
사십이
사십이
사십이
사십이
kr
사십이
cn
四十二
사십이
四十二
사십이
四十二
사십이
四十二
사십이
四十二
```

서버와 클라이언트는 실행할 때 -p 플래그를 추가해 포트 번호를 지정할 수 있습니다. 예를 들어 -p 8080을 지정하면 8080번 포트를 사용합니다. 플래그가 주어지지 않을 경우 8000번 포트를 기본으로 사용합니다.

```
▶ ./server -p 8080
The server is running
```

```
▶ lsof -i :8080 | grep LISTEN
server 64032 yongckim 5u IPv4 0xa67fd0ab03fe6e07 0t0 TCP localhost:http-alt (LISTEN)
```

# Chapter V

## Exercise 02: 42 chatting

Aa Name	Tags
<u>Turn-in directory</u>	ex02
<u>Files to turn in</u>	chat/server.go , chat/client.go
<u>Module name</u>	None
<u>Allowed packages</u>	"flag", "fmt", "io", "log", "net", "os", "bufio" "strconv"

42학생들이 모여서 익명으로 대화할 수 있는 프로그램을 만들려고 합니다. 우선, 채팅을 하는 클라이언트와 연결된 모든 클라이언트가 입력한 채팅을 전송하는 서버를 만드십시오. 각 프로그램의 main 함수가 작성되어야 합니다. Error는 적절히 처리하십시오.

채팅서버는 8000번 포트를 사용하여 연결하여 실행시 다음과 같이 작동합니다.

```
▶ ./server
Chatting Server running...
```

서버는 클라이언트가 접속하거나 종료시 다음과 같이 로그를 남겨야 합니다.

```
▶ ./server
Chatting Server running...
2021/11/03 20:34:49 login : 127.0.0.1:49414(anonymous 0)
2021/11/03 20:35:43 logout : 127.0.0.1:49414(anonymous 0)
```

클라이언트는 8000번 포트로 연결하여 실행시 연결에 성공했다는 메시지와 함께 자신의 아이디를 출력해주며 종료시 메시지를 출력해주어야 합니다.

```
▶ ./client
Connection Success, If you want to end, input EOF.
login : anonymous 0
bye
```

채팅서버에 새로운 클라이언트가 들어오면 다음과 같은 메시지가 출력되어야 합니다.

```
▶ ./client
Connection Success, If you want to end, input EOF.
login : anonymous 0
anonymous 1 participated.
```

다음과 같이 채팅입력시 자기자신을 포함한 모든 클라이언트에게 채팅 메시지가 보여야 합니다.

```
▶ ./client
Connection Success, If you want to end, input EOF.
login : anonymous 0
anonymous 1 participated.
hello
anonymous 0: hello
```

```
▶ ./client
Connection Success, If you want to end, input EOF.
login : anonymous 1
anonymous 0: hello
```