**GO web server**

web.go…..

package webstuff

import (

"net/http"

"github.com/labstack/echo/v4"

)

func HelloWebCall() {

e := echo.New()

e.GET("/", func(c echo.Context) error {

return c.String(http.StatusOK, "Hello, World!")

})

e.Logger.Fatal(e.Start(":1323"))

}

func IDWebcall() {

e := echo.New()

e.GET("/users/:id", func(c echo.Context) error {

// User ID from path `users/:id`

id := c.Param("id")

return c.String(http.StatusOK, "Hello "+id)

})

e.GET("/", func(c echo.Context) error {

return c.String(http.StatusOK, "server running")

})

e.Logger.Fatal(e.Start(":8080"))

}

Main.go….

package main

import (

greet "myPOC/myPackage"

webstuff "myPOC/myWebPackage"

)

func main() {

// az aks get-credentials --resource-group K8SRG --name aksbetatest

// kubectl apply -f deploy.yml

// curl http://20.108.239.131:30475/users/bob

// kubectl logs web-65f9f75fdf-dccct

//

greet.Hello()

webstuff.IDWebcall()

}

In visual code, build a dockerfile. Note the extra syntax around copying packages and exposing ports:

# syntax=docker/dockerfile:1

FROM golang:1.18-alpine

WORKDIR /app

COPY go.mod ./

COPY go.sum ./

RUN go mod download

COPY \*.go ./

COPY ./myPackage/\*.go ./myPackage/

COPY ./myWebPackage/\*.go ./myWebPackage/

RUN go build -o /docker-mypoc

EXPOSE 8080

CMD [ "/docker-mypoc" ]

In visual code, build the container, then run it locally:

// docker build --tag alpha .

// docker image ls

// docker run --publish 8080:8080 beta

// http://localhost:8080/users/bob