From a golang http handler to kubernetes

Go meetup Paris - 4 Apr 2016

Adrien Delorme
Rive Paris

Go in prod ?

Go in prod !

- Über - How we built uber engineering's highest query per second service using go (https://eng.uber.com/go-geofence/)
- Google - dl.google.com: Powered by Go (https://talks.golang.org/2013/oscon-dl.slide)

And others !

- https://github.com/golang/go/wiki/GoUsers (https://github.com/golang/go/wiki/GoUsers)

HTTP handling in go

HTTP handling in go - Basics

```
package main

import (
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}
```

Run

HTTP handling in go - Real life

HTTP handling in go - Real life

```
import (
    "encoding/json"
    "net/http"
    "strconv"
)

func init() { http.HandleFunc("/user", UserGetHandler) }

func UserGetHandler(w http.ResponseWriter, r *http.Request) {
    id, err := strconv.ParseUint(r.URL.Query().Get("id"), 10, 64)
    if err != nil {
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    user, found, err := DB.UserGet(id)
    if err != nil {
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    if !found {
        w.WriteHeader(http.StatusNotFound)
    }
    json.NewEncoder(w).Encode(&user)
}
```

Run

Can't this be simple ?

HTTP handling in go - Code generation

Create User

But what does core job really look like ?

```
func UserCreate(u UserCreateRequest) (status int, err error) {
    return http.StatusCreated, DB.CreateUser(u)
}
```

... And how do I get the request ?

```
type UserCreateRequest struct {
    Name string `validate:"min=3,max=40,regexp=^[a-zA-Z]*$"`
}


func HTTPUserCreateRequest(r *http.Request) (uc UserCreateRequest, err error) {
    err = json.NewDecoder(r.Body).Decode(&uc)
    if err != nil {
        return
    }
    err = validator.Validate(uc)
    return
}
```

HTTP handling in go - Gode Generation

```
//go:generate varhandler -func UserCreate
func UserCreate(u UserCreateRequest) (status int, err error) {
```

HTTP handling in go - Gode Generation

The jobs:

```
//go:generate varhandler -func UserCreate
func UserCreate(u UserCreateRequest) (status int, err error) {
```

```
//go:generate varhandler -func UserGet
func UserGet(un UserName) (u User, status int, err error) {
```

Run "go generate"

```
func init() {
    http.HandleFunc("/user/create", UserCreateHandler)
    http.HandleFunc("/user/get", UserGetHandler)
}
```

Code Generated - show time

Any question so far ?

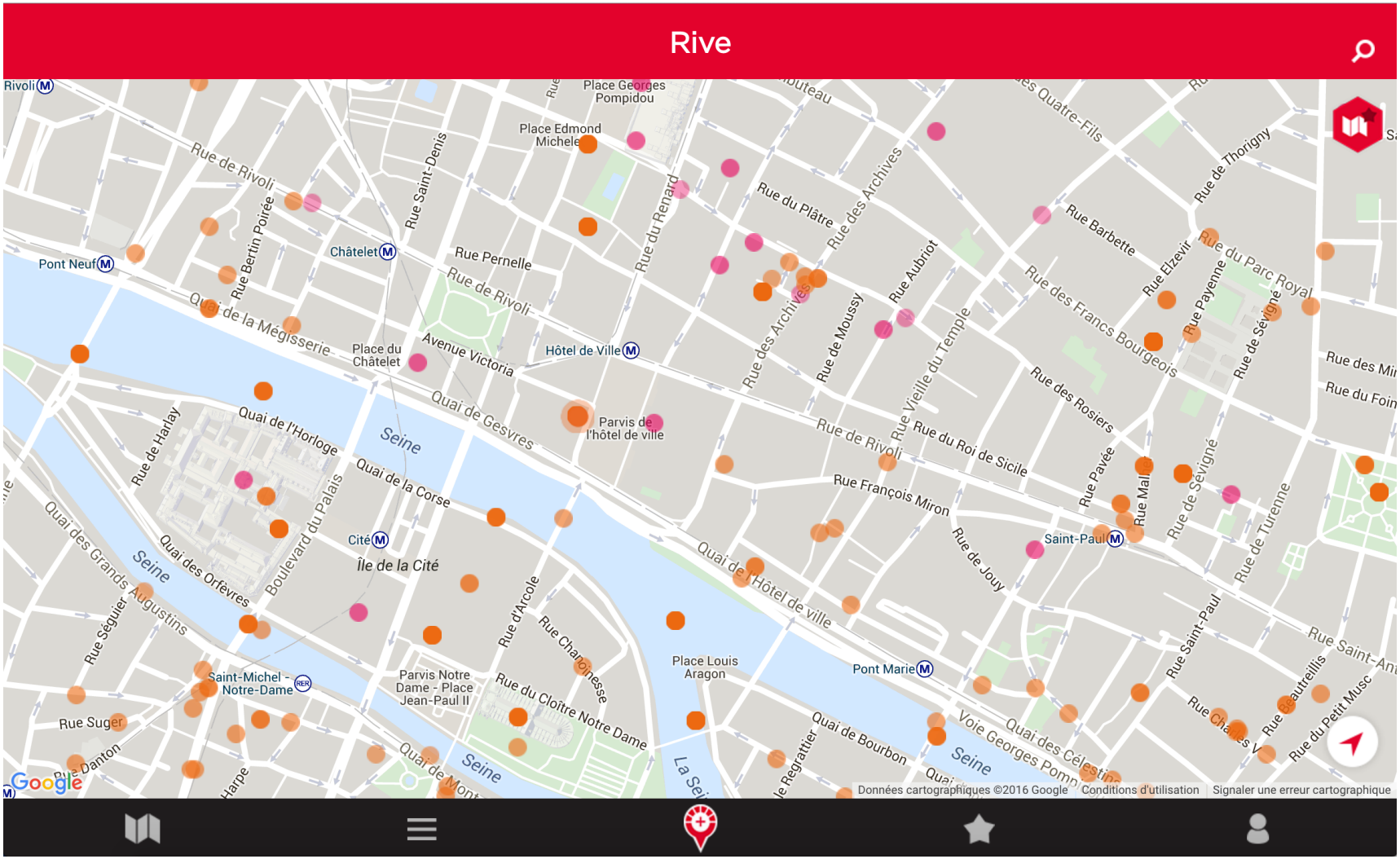
Where to run those handlers ?

When it's a line that calls a backend service

- Appengine
- Heroku
- Others

But sometimes it's not that simple

Other cases - Rive



We chose Kubernetes for this

Kubernetes

κυβερνήτης: *Greek for "pilot" or "helmsman of a ship"*
the open source cluster manager from Google

A.K.A **k8s**

How we use it ?

Git flow

- Code your exotic program or feature
- Submit code review

--> For every git push tests are run

When code is merged

- re run tests
- build executable

Deploy

Deploy - init

- dockerise executable
- push image to private repository

Deploy - create the replication controller

```
#!/bin/bash
cat <<EOF
apiVersion: v1
kind: ReplicationController
metadata:
  name: rtpush-${GIT_SHA}
spec:
  replicas: ${NB_REPLICAS}









EOF
```

Deploy - define the template of a pod

```
#!/bin/bash
cat <<EOF
apiVersion: v1
kind: ReplicationController
metadata:
  name: rtpush-${GIT_SHA}
spec:
  replicas: ${NB_REPLICAS}
  selector:
    app: rtpush
    deployment: ${GIT_SHA}
  template:
    metadata:
      name: rtpush-${GIT_SHA}
      labels:
        app: rtpush
        deployment: ${GIT_SHA}
    spec:
      containers: #pods of the rc
      - name: rtpush
        image: gcr.io/rive/rtpush:${GIT_SHA}
        ports:
        - containerPort: 8080
EOF
```

Deploy - define limits of a pod

- number of cpu needed
- max mem use per pod

Deploy - rolling update

```
$ kubectl rolling-update ${PREVIOUS_RC_NAME} -f rc-rtpush.json
```

Deploy - rolling update output

```
$ kubectl rolling-update rtpush-b185186 -f rc-rtpush.json
Created rtpush-new
Scaling up rtpush-new from 0 to 1, scaling down rtpush-b185186  from 1 to 0 (keep 1 pods
available, don't exceed 2 pods)

Scaling rtpush-new up to 1

Scaling rtpush-b185186 down to 0

Update succeeded. Deleting rtpush-b185186 replicationcontroller "rtpush-b185186"
rolling updated to "rtpush-new"
```

It's live !

Deploy - create the service

```
---
  kind: "Service"
  apiVersion: "v1"
  metadata:
    name: "rtpush"
    labels:
      app: "rtpush"
  spec:
    type: "LoadBalancer"
    loadBalancerIP: "SOME_IP"
    ports:
      -
        name: "http"
        port: 80
        targetPort: 8080
      -
        name: "https"
        port: 443
        targetPort: 4443
    selector:
      app: "rtpush"
```

It's reachable !

Et voilà

Thank you

Go meetup Paris - 4 Apr 2016

Adrien Delorme
Rive Paris

http://www.rive.world/ (http://www.rive.world/)

http://github.com/azr (http://github.com/azr)

http://github.com/azr/generators (http://github.com/azr/generators)

http://github.com/azr/talks (http://github.com/azr/talks)

@gratator (http://twitter.com/gratator)