
Creating a Dynamic
Reverse Proxy with Go
Kincaid Savoie
Hoodoo

Things we'll cover

- My team's (slightly unusual) reverse proxy problem
- Why we used Go to solve it
- How you can do the same

A little about me

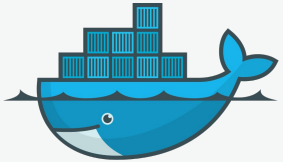
- Graduated from University of Utah
- Competed with supercomputing team
- Previously at Hoodoo Digital
- Fair amount of AWS experience

OUR PROBLEM

Our app

- Helps manage monolithic Java apps
- Uses Docker to provide developer environments
- Kubernetes starts environments for QA and stakeholders to evaluate
- Big idea: eliminate differences between environments & promote CI

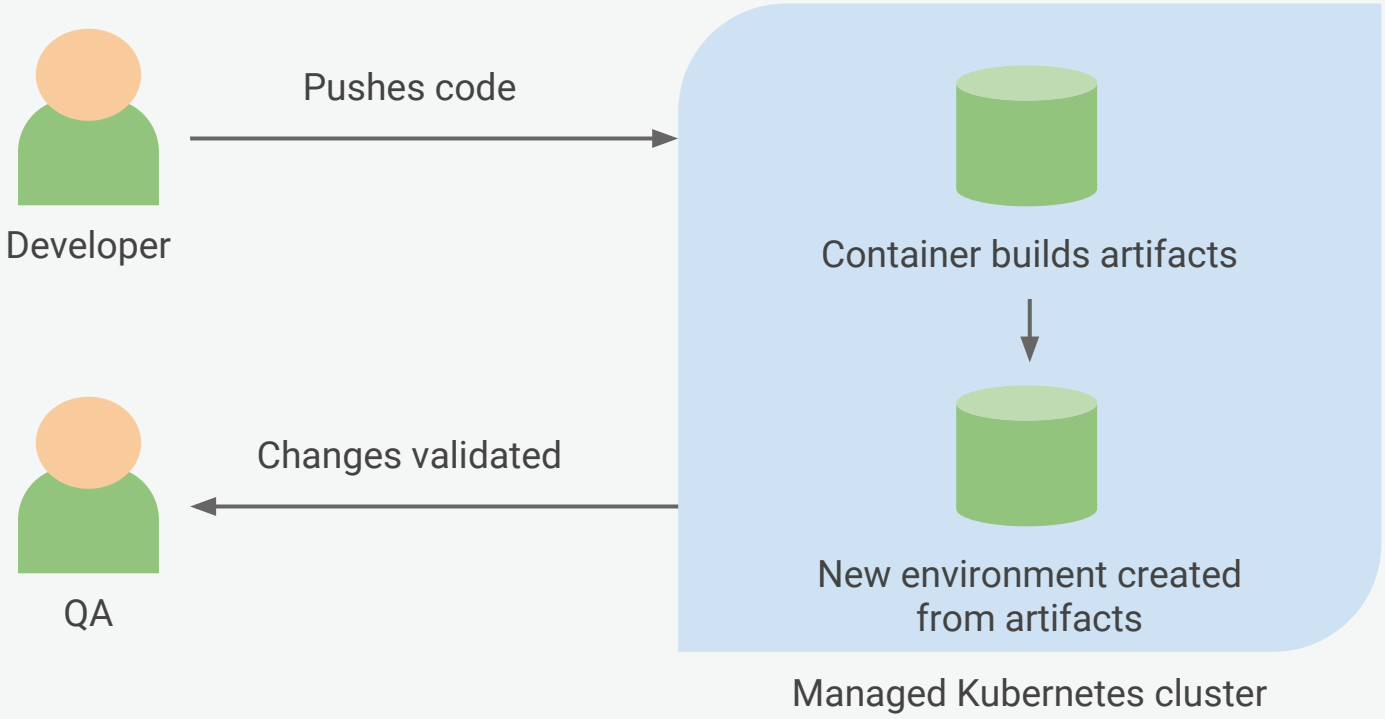
Adobe Experience Manager



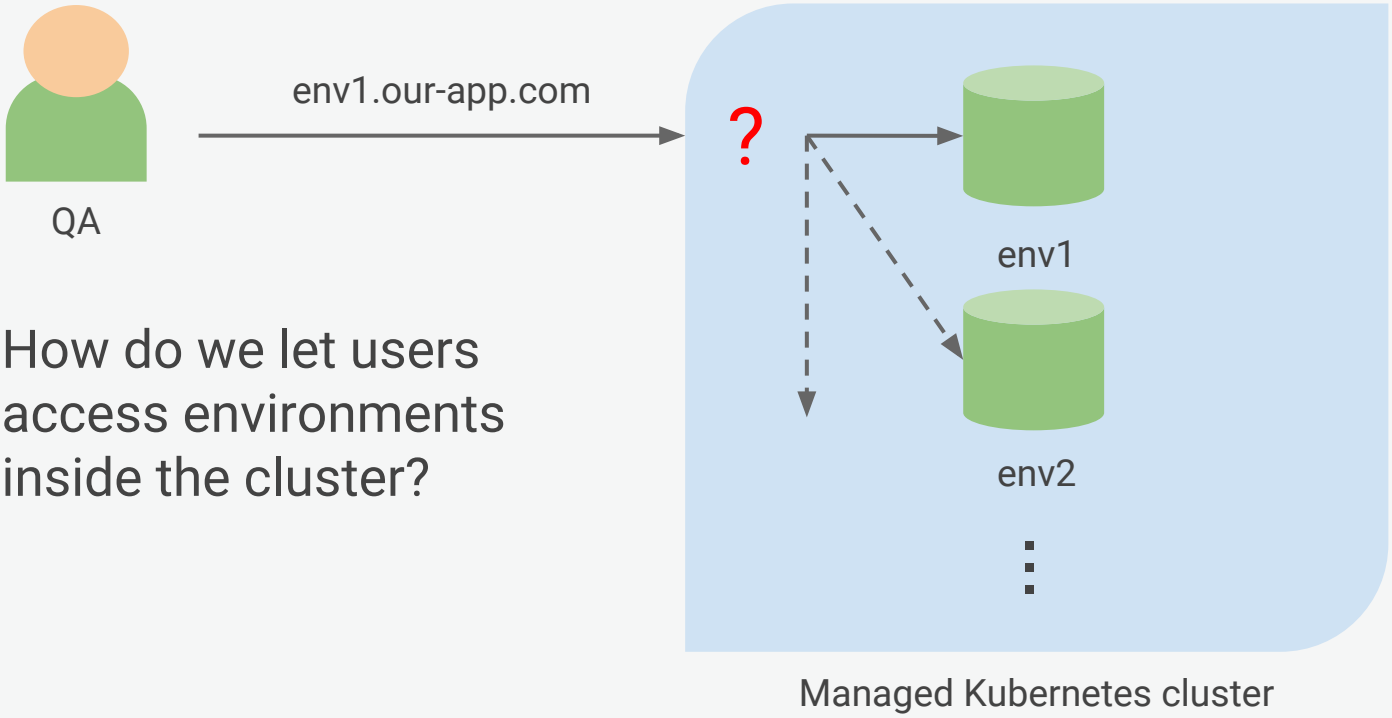
docker

kubernetes

Workflow overview



The problem



How do we let users access environments inside the cluster?

Extra goals

- Report environment access times to management server
- Collect performance metrics on pages accessed
- Start environments if they are accessed
- Don’t modify environments themselves

The solution

- Reverse proxy based on URL
- Perform environment lookups with Kubernetes API
- Report performance, access, etc. to management server
- Redirect to loading, environment missing, etc. pages dynamically

OUR IMPLEMENTATION

Off the shelf web servers

- Apache, Nginx, etc.
- Very fast, stable
- Lots of features
- Not very dynamic
- Difficult to add custom features

™

NGINX

Writing our own with Go

- Language is fast (enough)
- Easy to Dockerize (compiled binaries)
- Can leverage Apache/Nginx features by placing our proxy behind theirs
- Helpful standard library, which includes...
- a built-in reverse proxy framework!

WHIRLWIND TOUR OF GO

Variable declarations and assignments

```
var x string = "long-hand declaration"

y := "inferred type declaration"

y = "assignment"
```

Conditionals

```
if x < 100 {
   doSomething()
}
```

Loops

```
for i, item := range someList {
   fmt.Println(i, item)
}

sum := 1
for sum < 100 {
   sum += 1
}
```

Function declarations

```
func add(x int, y int) int {
   return x + y
}
```

Multiple return values

```
func multReturn(x int) (int, error) {
   if x < 1 {
      return x, errors.New("negative x")
   } else {
      return x, nil
   }
}
```

Structs

```
type rectangle struct {
   length int
   width int
}

func (r rectangle) getArea() int {
   return r.length * r.width
}
```

Pointers

```
mySquare := newRectangle(5,5)

var sqrPointer *rectangle = &mySquare

dereferencedSquare := *sqrPointer
```

Goroutines

```
mySquare := newRectangle(5,5)

go mySquare.getArea()
```

Channels

```
someChannel := make(channel string)
go func() {
   for {
      someChannel <- "hi"
   }
}()
```

REVERSE PROXY IMPLEMENTATION

https://github.com/Caid11/creating-a-dynamic-reverse-proxy-with-go


Thanks!