



Airy Journey on Adopting Kubernetes in AWS & Intro to Telepresence

Kurnianto Trilaksono - Infrastructure / DevSecOps Team - Airy

About

Kurnianto Trilaksono (Anto)

@antooooks | t.me/kurniantotrilaksono

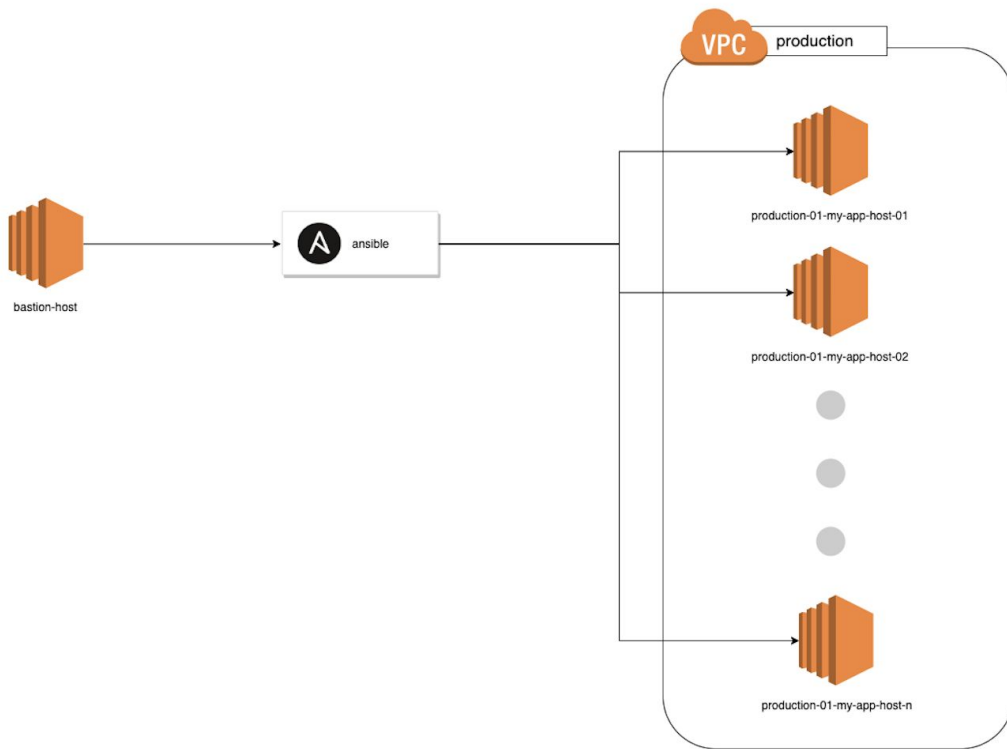
- Quality Engineer of Airy for Product (2017-2018)
- Member of Airy Infrastructure / DevSecOps Team (2018-present)

Contents

- Kubernetes at Airy
- Develop with Telepresence
- Demo
- Lessons Learned

Kubernetes at Airy

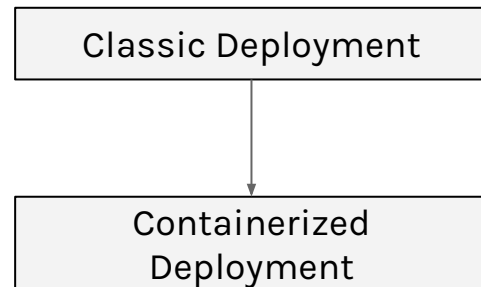
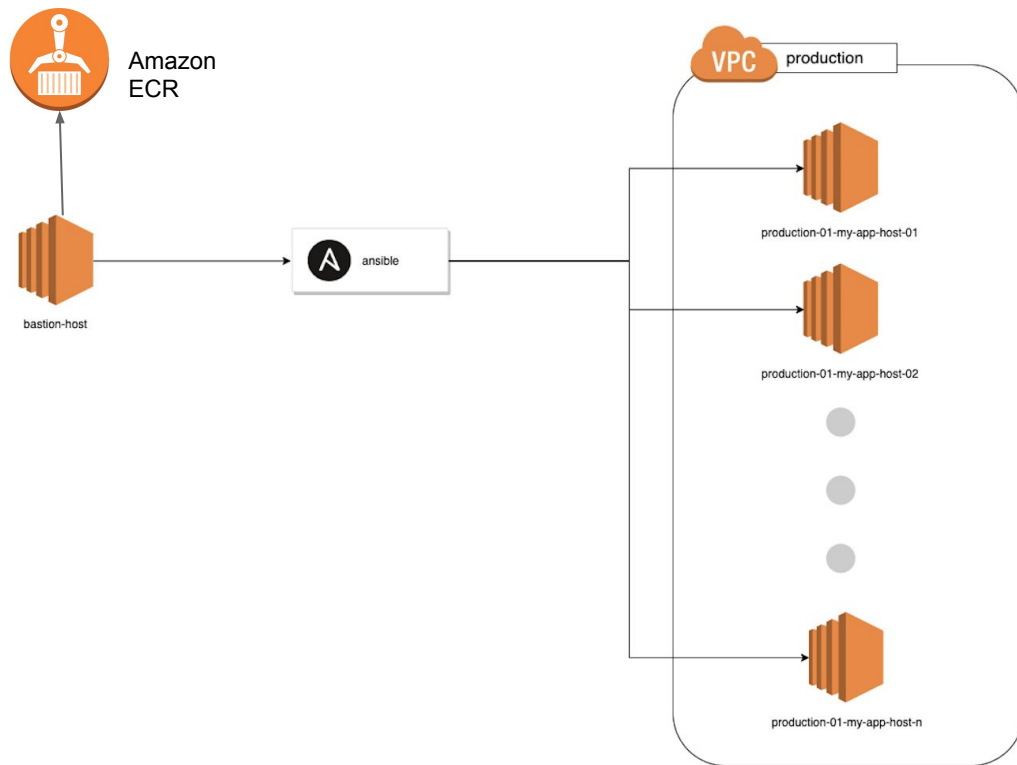
Before the Age of Kubernetes



Classic Deployment

Source:

Before the Age of Kubernetes



Journey to Kubernetes

- Born on **AWS** cloud provider
- **Microservice** in mind
- Multiple **stacks**
- All using **AWS EC2** Instances
- Services scale so does **infrastructure cost**
- Focusing on **reliability, scalability, and operational excellence**

Journey to Kubernetes

Another one:



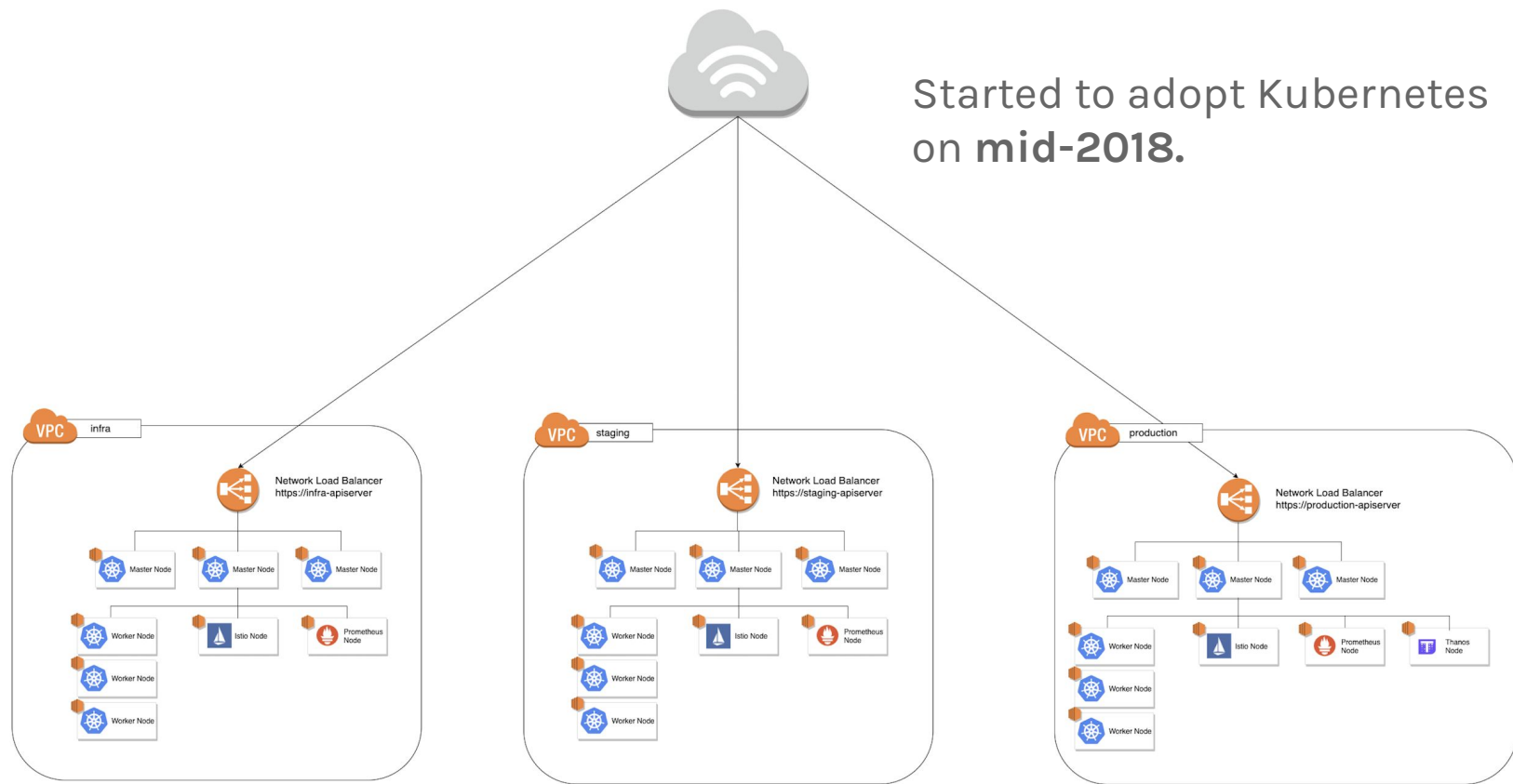
PET

VS

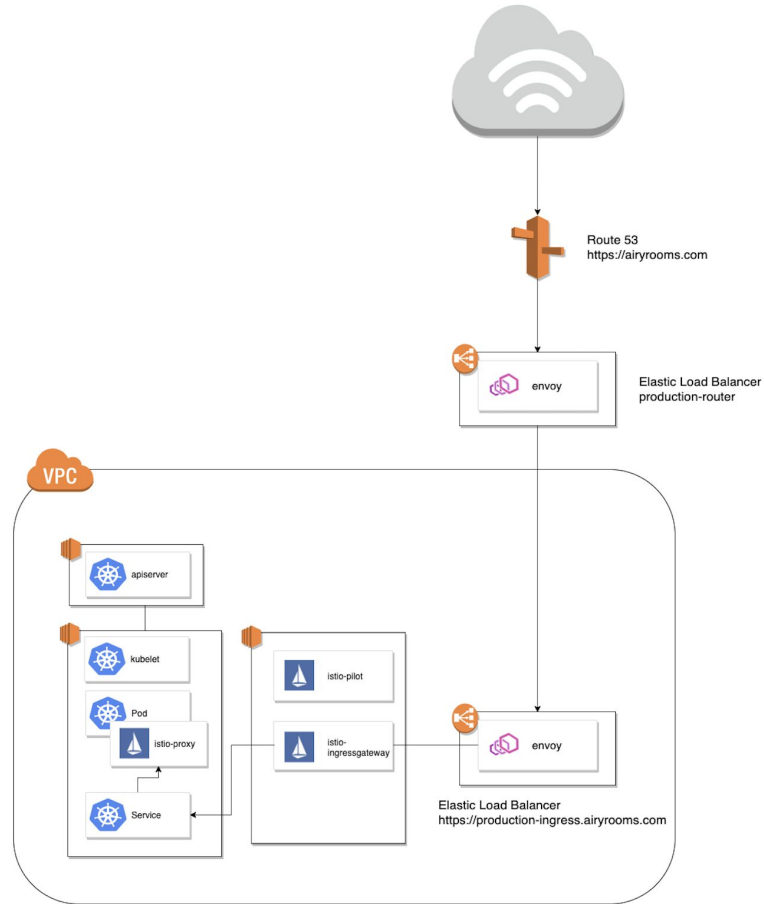


CATTLE

Architecture Overview



Architecture Overview (Cont'd)



3 Environments

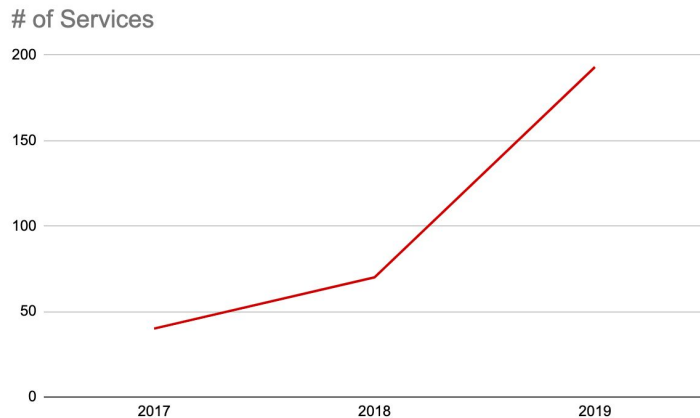
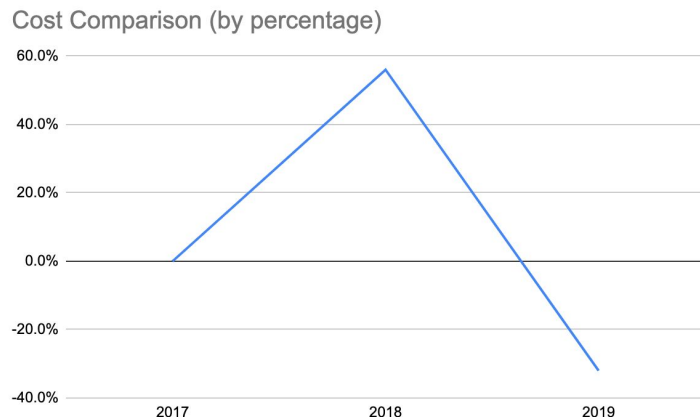
80 Nodes

190+ Services

1100+ Live Pods

Source:

Life after Kubernetes



... Also more complexity for developers :(

A Developer Perspective

- Debugging is harder (e.g. some debugging cannot be done in local environment)
- Unreliable VPN connection
- Inability to respond quickly to alerts

Develop with Telepresence

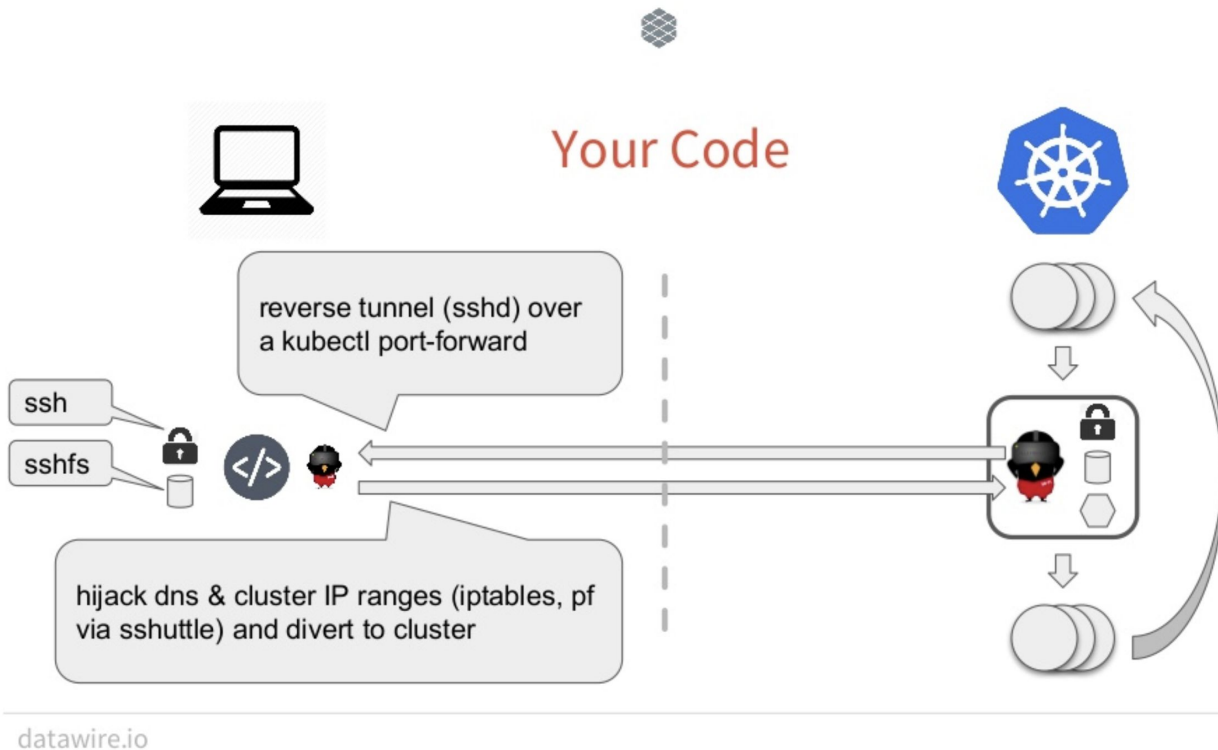
Enter Telepresence



Telepresence is an open source tool that lets you run a single service locally, while connecting that service to a remote Kubernetes cluster.



How Does it Work?



Source: <https://www.slideshare.net/datawire/kubecon-na-2018-t-elepresence-deep-dive-session-rafael-schloming-luke-shumaker>

Use Case 1: VPN to Deployment Environment

Sometimes developer has to work remotely which makes them depend heavily on our VPN.

- ▶ VPN is a third-party service
- ▶ You have to login to the login UI first before VPN creates tunnel for you
- ▶ Most of the time VPN experience bottlenecks and broken pipe
- ▶ Our VPN is practically unreliable

Use Case 1: VPN to Deployment Environment

```
telepresence --context="my-cluster-01"  
--new-deployment="telepresence-$(date +%s)" \  
--namespace="demo" \  
--logfile="/tmp/telepresence_$(date +%m-%d-%Y_%H:%M:%S).log" \  
--run-shell
```

NAME	READY	STATUS	RESTARTS	AGE
client-app-987f796c6-bqq7x	2/2	Running	0	23h
server-app-6f46fc8866-7wvgb	2/2	Running	0	23h
telepresence-1576384103-78d6ff89f6-qh4sk	1/1	Running	0	19s

Source:

Use Case 2: Development & Live Debugging

- Your QA find bugs on staging. You tried to reproduce the bug using local environment, and you can't seem to find it happens on local environment.
- You introduced a hotfix, you have to run build scripts, build the container, then deploy your code back to deployment environment. After your application is up and running, it turns out that the bug still happening or another bug is introduced afterwards.

Use Case 2: Development & Live Debugging

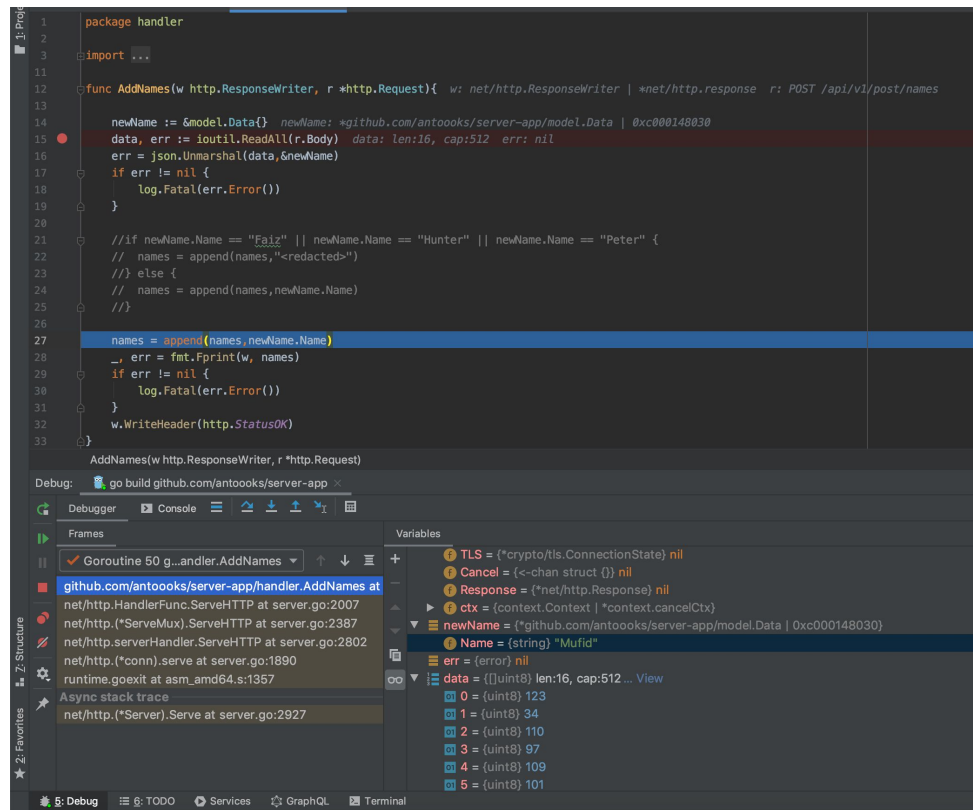
```
telepresence --context="my-cluster-01"  
--swap-deployment="server-app" \  
--namespace="demo" \  
--logfile="/tmp/telepresence_$(date +%m-%d-%Y_%H:%M:%S).log" \  
--run-shell
```

NAME	READY	STATUS	RESTARTS	AGE
client-app-987f796c6-bqq7x	2/2	Running	0	24h
server-app-2fdd893a031d49798fa07bccf559c25d-7579d86799-gr44k	2/2	Running	0	3s
server-app-6f46fc8866-gvg9t	2/2	Terminating	0	39s

Source:

Use Case 2: Development & Live Debugging

you can use it with your favorite IDE !



The screenshot shows an IDE with a Go source file and its debug state. The source code is as follows:

```
1 package handler
2
3 import ...
4
11 func AddNames(w http.ResponseWriter, r *http.Request){ w: net/http.ResponseWriter | *net/http.response r: POST /api/v1/post/names
12
13
14     newName := &model.Data{ newName: *github.com/antiooks/server-app/model.Data | 0xc000148030
15     data, err := ioutil.ReadAll(r.Body) data: len:16, cap:512 err: nil
16     err = json.Unmarshal(data, &newName)
17     if err != nil {
18         log.Fatal(err.Error())
19     }
20
21     //if newName.Name == "Faiz" || newName.Name == "Hunter" || newName.Name == "Peter" {
22     // names = append(names, "<redacted>")
23     //} else {
24     // names = append(names, newName.Name)
25     //}
26
27     names = append(names, newName.Name)
28     _, err = fmt.Fprintf(w, names)
29     if err != nil {
30         log.Fatal(err.Error())
31     }
32     w.WriteHeader(http.StatusOK)
33 }
```

The debug state shows the following:

- Debugger:** go build github.com/antiooks/server-app <
- Frames:**
 - Goroutine 50 g...andler.AddNames
 - github.com/antiooks/server-app/handler.AddNames at net/http.HandlerFunc.ServeHTTP at server.go:2007
 - net/http.(*ServeMux).ServeHTTP at server.go:2387
 - net/http.serverHandler.ServeHTTP at server.go:2802
 - net/http.(*conn).serve at server.go:1890
 - runtime.goexit at asm_amd64.s:1357
 - Async stack trace
 - net/http.(*Server).Serve at server.go:2927
- Variables:**
 - TLS = (*crypto/tls.ConnectionState) nil
 - Cancel = (<-chan struct {}) nil
 - Response = (*net/http.Response) nil
 - ctx = (context.Context | *context.cancelCtx)
 - newName = (*github.com/antiooks/server-app/model.Data | 0xc000148030)
 - Name = (string) "Mufid"
 - err = (error) nil
 - data = ([[]uint8] len:16, cap:512 ... View
 - 0 = (uint8) 123
 - 1 = (uint8) 34
 - 2 = (uint8) 110
 - 3 = (uint8) 97
 - 4 = (uint8) 109
 - 5 = (uint8) 101

Source:

Limitations

- Does not support Windows OS natively, or Ubuntu older than 16.04
- Cannot auto-mounting other Kubernetes resources such as ConfigMap and Secrets
- Cannot use other VPN while using telepresence
- Cannot be more than one telepresence session at the same time
- Cloud resources outside Kubernetes are not resolved automatically, you have to use **--also-proxy** and add your resource **VPCs** (e.g. **--also-proxy="192.10.0.0/16"**)
- Bug when resolving Kubernetes FQDN, **Linux** and **vpn-tcp method** only (e.g. server-app.demo.svc.cluster.local)

<https://github.com/telepresenceio/telepresence/issues/161>

Source:

Demo

Demo

- server-app -- holds a list of names
- client-app -- a job to register new names to server-app

Lessons Learned

Lessons Learned

- Using Kubernetes on EC2 will absolutely cost less than it should be, given amount of services deployed.
- Adopting Kubernetes introduce new challenges on various aspects.
- Telepresence enables to bring 'your presence' to the deployment environment.
- You can get feedback faster by using Telepresence, therefore development is faster.



Thank You !





is hiring for
Infrastructure / DevSecOps Eng.

airy.com/careers