KubeCon | CloudNativeCon

North America 2017

Moving from Mesos to Kubernetes *without anyone noticing**

Anubhav Mishra

Anubhav Mishra

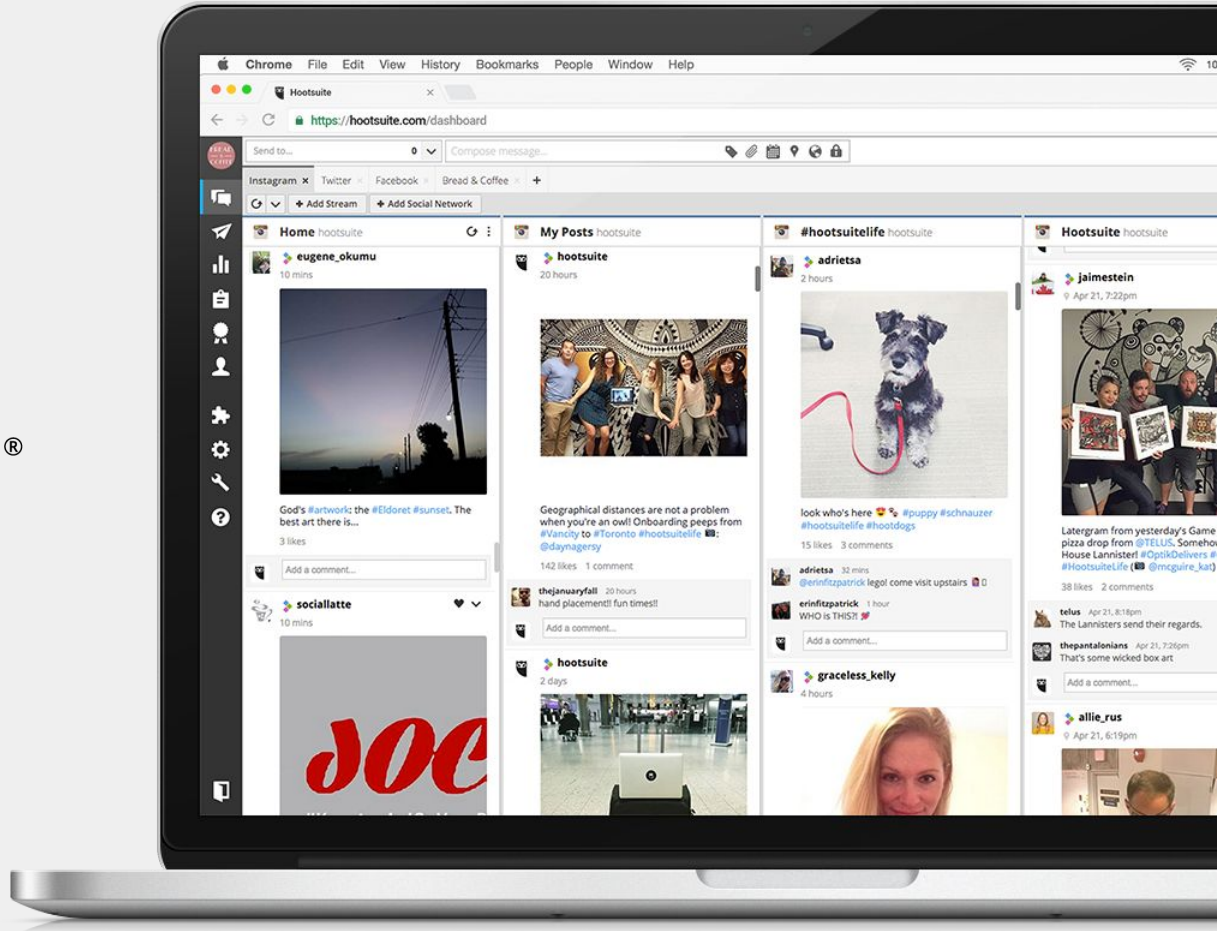
@anubhavm

Hootsuite®

Anubhav Mishra

@anubhavm

Hootsuite®


Chrome File Edit View History Bookmarks People Window Help
Hootsuite
https://hootsuite.com/dashboard
Send to...
Compose message...
Instagram
Twitter
Facebook
Bread & Coffee
+ Add Stream
+ Add Social Network
Home hootsuite
eugene_okumu
10 mins
God's #artwork: the #Eldoret #sunset. The best art there is...
3 likes
Add a comment...
sociallatte
10 mins
My Posts hootsuite
hootsuite
20 hours
Geographical distances are not a problem when you're an owl! Onboarding peeps from #Vancity to #Toronto #hootsuitelife :: @daynagersy
142 likes 1 comment
thejanuaryfall 20 hours
hand placement!! fun times!!
Add a comment...
hootsuite
2 days
#hootsuitelife hootsuite
adrietsa
2 hours
look who's here #puppy #schnauzer #hootsuitelife #hootdogs
15 likes 3 comments
adrietsa 32 mins
@erinfitzpatrick lego! come visit upstairs
erinfitzpatrick 1 hour
WHO is THIS?!
Add a comment...
graceless_kelly
4 hours
Hootsuite hootsuite
jaimestein
Apr 21, 7:22pm
Latergram from yesterday's Game
pizza drop from @TELUS. Someho
House Lannister! #OptikDelivers
#HootsuiteLife (@mcguire_kat)
38 likes 2 comments
telus Apr 21, 8:18pm
The Lannisters send their regards.
thepantalonians Apr 21, 7:26pm
That's some wicked box art
Add a comment...
allie_rus
Apr 21, 6:19pm

Hootsuite®

Anubhav Mishra

@anubhavm

Hootsuite®

Atlantis

Anubhav Mishra

@anubhavm

Atlantis

Anubhav Mishra

@anubhavm

HashiCorp
Terraform

Atlantis

Anubhav Mishra

@anubhavm

HashiCorp
Terraform

Hootsuite®

Atlantis

Anubhav Mishra

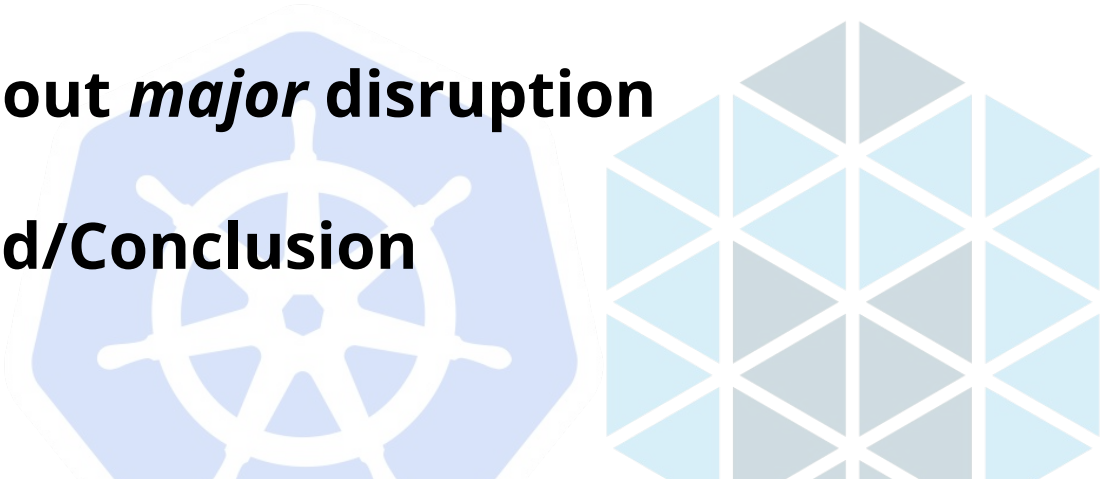
@anubhavm

HashiCorp Terraform

VS

Agenda

- Hootsuite's Journey from Mesos to Kubernetes
- Microservices pipeline
  - Mesos and Marathon
  - Kubernetes
- Migration without *major* disruption
- Live demo! 🙏
- Lessons learned/Conclusion



Hootsuite Now

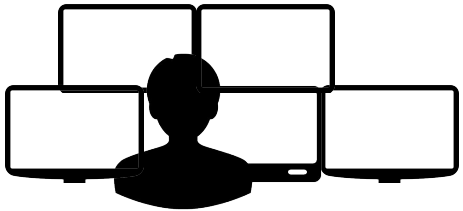
Numbers

- **120+** developers
- **50+** microservices
- **2** cluster schedulers
- **1500+** servers on **AWS**

2014

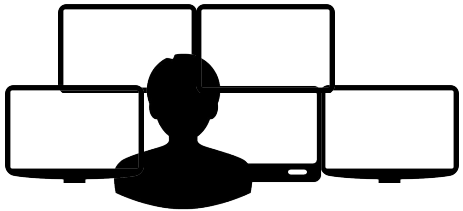
I want to build a microservice

I want to build a microservice



I want to build a microservice

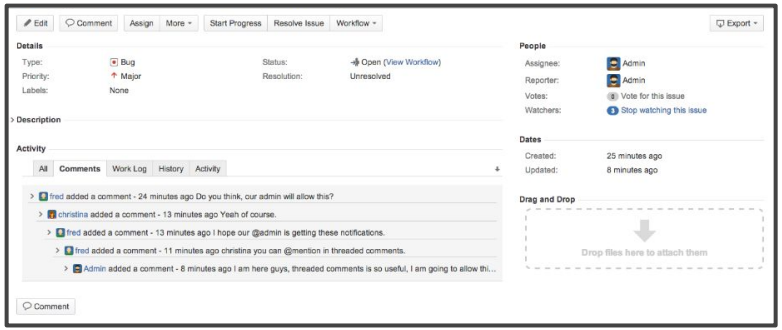
Oh! A “microservice”? Hmm.. seems to be the new thing huh. Yep, just create a JIRA ticket.

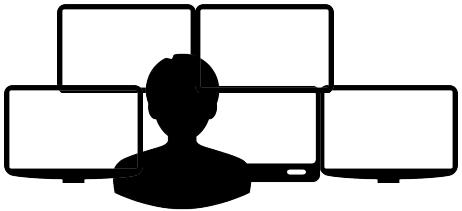


Minutes later….

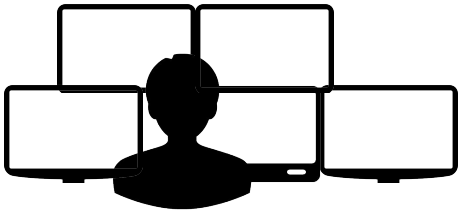

JIRA


JIRA

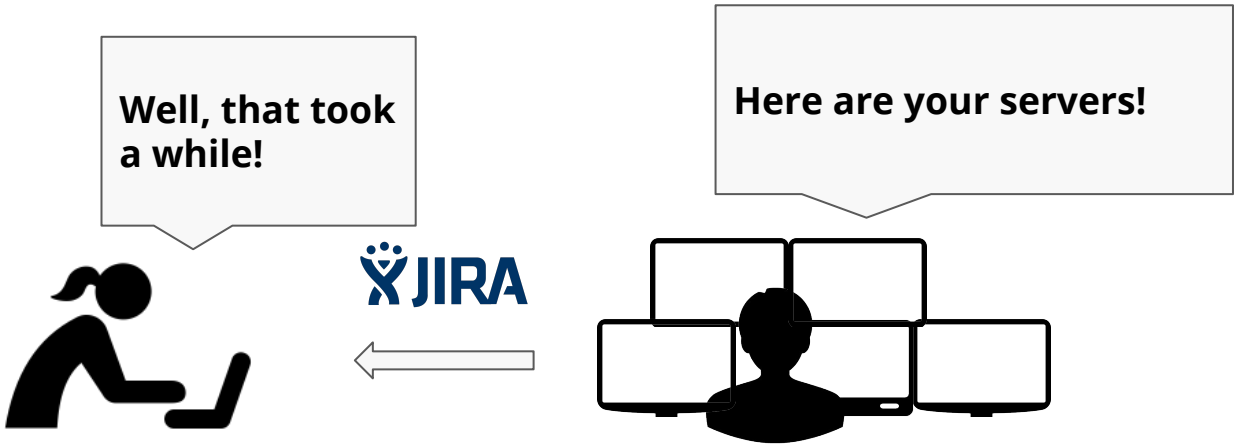

Edit
Comment
Assign
More
Start Progress
Resolve Issue
Workflow
Export
Details
Type: Bug
Priority: Major
Labels: None
Status: Open (View Workflow)
Resolution: Unresolved
People
Assignee: Admin
Reporter: Admin
Votes: Vote for this issue
Watchers: Stop watching this issue
Description
Activity
All
Comments
Work Log
History
Activity
fred added a comment - 24 minutes ago Do you think, our admin will allow this?
christina added a comment - 13 minutes ago Yeah of course.
fred added a comment - 13 minutes ago I hope our @admin is getting these notifications.
fred added a comment - 11 minutes ago christina you can @mention in threaded comments.
Admin added a comment - 8 minutes ago I am here guys, threaded comments is so useful, I am going to allow thi...
Comment
Dates
Created: 25 minutes ago
Updated: 8 minutes ago
Drag and Drop
Drop files here to attach them

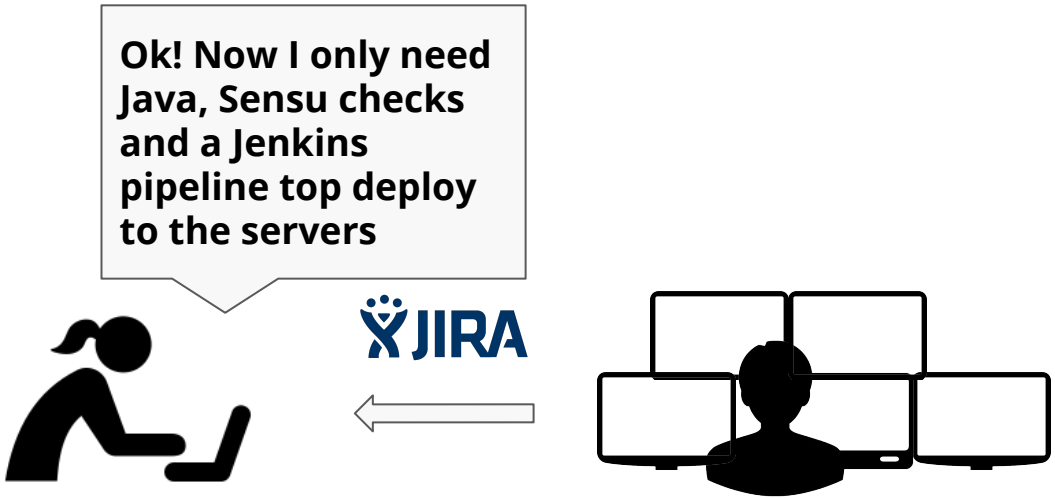



JIRA



Weeks later….


Well, that took a while!
JIRA
Here are your servers!


Ok! Now I only need Java, Sensu checks and a Jenkins pipeline top deploy to the servers
JIRA

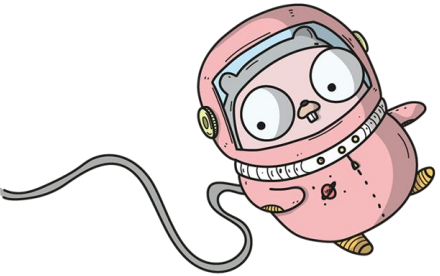
2016-2017

I want to build a microservice

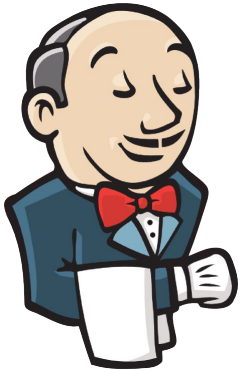
5 minutes later....


I just deployed a microservice to production!

Microservice Pipeline



`./project-generator`



Pipeline as Code

Mesos

Marathon

Project Skeleton

```
################################################
# Modify these vars

# Name of your service, lowercase, dash separated. ie. [a-z\-]
# !!! SHOULD NOT have "service" as prefix or suffix. We're trying to get rid of that.
# Will be used for Skyline and Kubernetes names as well as statsd prefixes, etc.
#name: my-awesome-service
name: new-service


# Human readable title. Used for docs and such
#nameNice: My Awesome Service
nameNice: My New Service

# Long form description
#description: Skyline sample service generated from the skeleton based off of Play! framework, dockerized, and
description: Hello World

#maintainers:
#  - name: Firstname Lastname
#    email: firstname.lastname@hootsuite.com
#  - name: Joe Blough
#    email: joe.blough@hootsuite.com
maintainers:
  - name: Anubhav Mishra
    email: anubhav.mishra@hootsuite.com

# Pick a project type. Can either be: scala, go, go-grpc, idl etc.
# This can be set with the command line arg `--project-type` which will allow you to
# create multiple project types with the same config. This is useful for projects like `go-grpc` + `idl`
# projectType: go
projectType: go
```

Project Skeleton

```
################################################
# Modify these vars

# Name of your service, lowercase, dash separated. ie. [a-z\-]
# !!! SHOULD NOT have "service" as prefix or suffix. We're trying to get rid of that.
# Will be used for Skyline and Kubernetes names as well as statsd prefixes, etc.
#name: my-awesome-service
name: new-service


# Human readable title. Used for docs and such
#nameNice: My Awesome Service
nameNice: My New Service

# Long [illegible]
#description: Skyline-ready service generated from the skeleton based on[illegible] of Play! framework, dockerized, and
description: Hello World

#maintainers:
#  - name: Firstname Lastname
#    email: firstname.lastname@hootsuite.com
#  - name: Joe Blough
#    email: joe.blough@hootsuite.com
maintainers:
  - name: Anubhav Mishra
    email: anubhav.mishra@hootsuite.com

# Pick a project type. Can either be: scala, go, go-grpc, idl etc.
# This can be set with the command line arg `--project-type` which will allow you to
# create multiple project types with the same config. This is useful for projects like `go-grpc` + `idl`
# projectType: go
projectType: go
```

./project-generator =

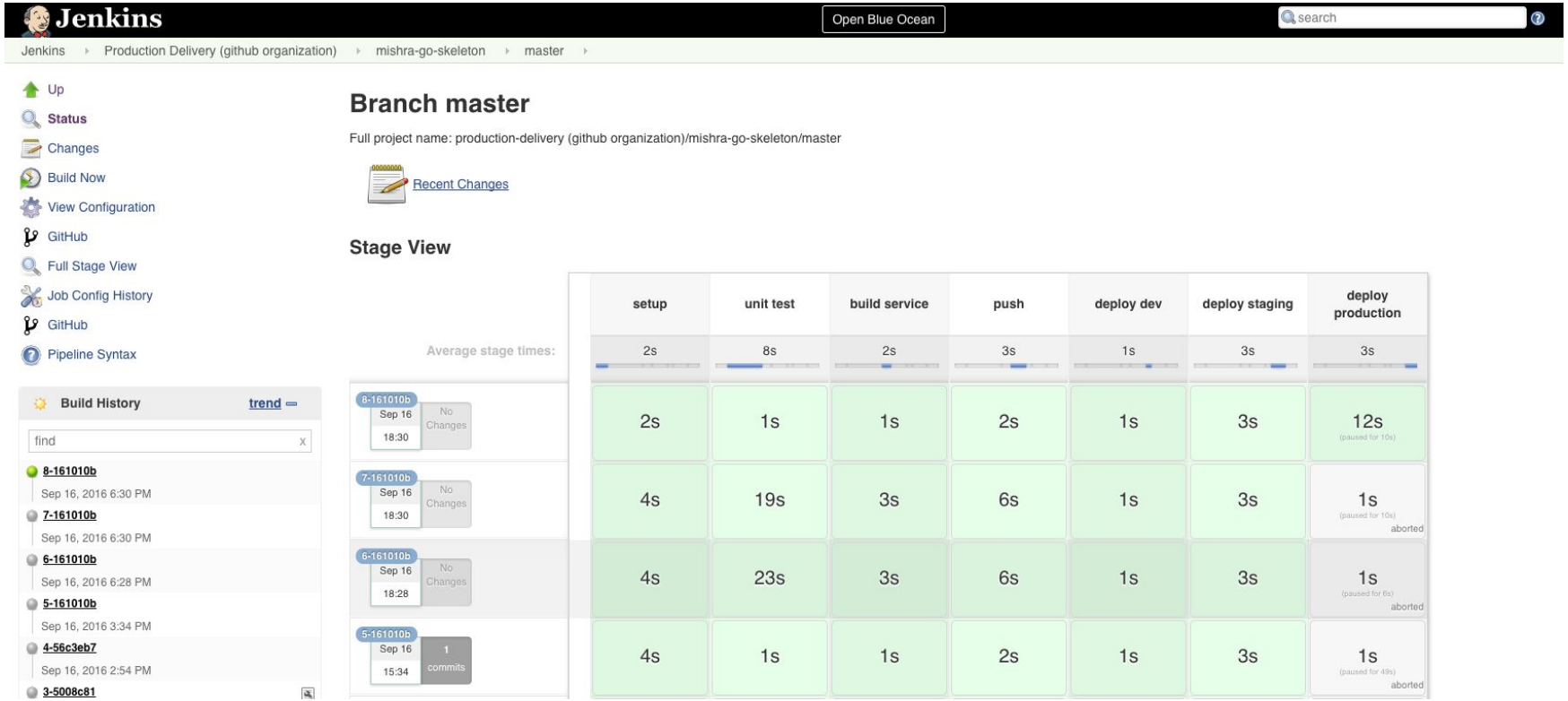
```
.
├── Dockerfile
├── Dockerfile-dev
├── Jenkinsfile
├── Makefile
├── Makefile.base.mk
├── README.md
├── bin
│   ├── coverage.sh
│   ├── envtpl
│   └── tdd.sh
├── conf
│   ├── about.json
│   ├── default.yml
│   ├── dev.yml
│   ├── production.yml
│   ├── staging.yml
│   └── vagrant.yml
├── deploy
│   ├── application.yml
│   ├── dev.yml
│   ├── production.yml
│   ├── staging.yml
│   └── vagrant.yml
├── deploy-test.py
├── docs
│   └── todo
├── index.go
├── main.go
├── main_test.go
├── middleware.go
├── server.go
├── skeleton-vars.yml
├── status.go
└── status_test.go
```

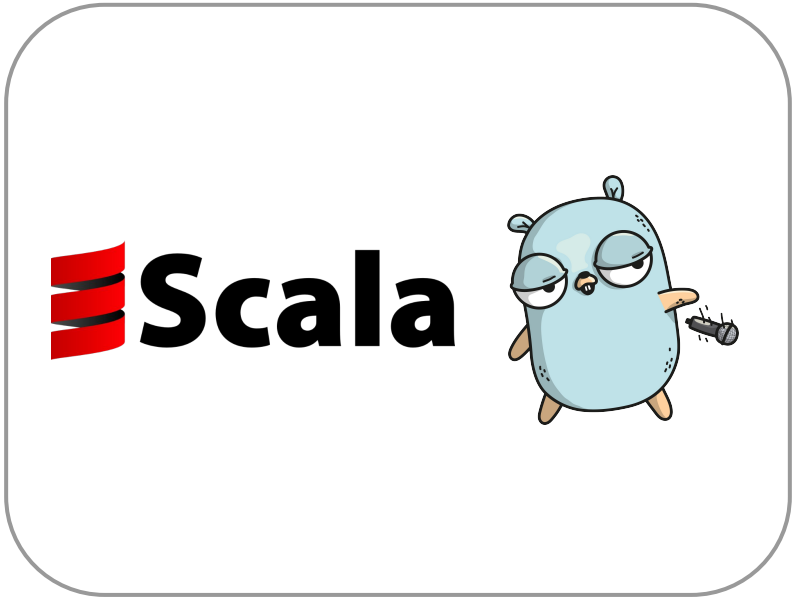
Pipeline as Code

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                sh 'make'
            }
        }
        stage('Test'){
            steps {
                sh 'make test'
                junit 'reports/**/*.xml'
            }
        }
        stage('Deploy') {
            steps {
                sh 'make deploy'
            }
        }
    }
}
```

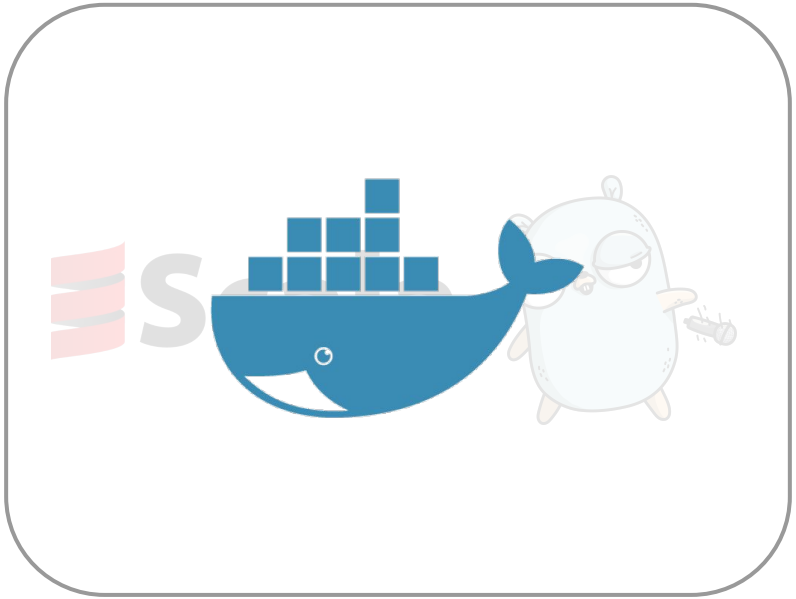
```
.
├── Dockerfile
├── Dockerfile-dev
├── Jenkinsfile
├── Makefile
├── Makefile.base.mk
├── README.md
├── bin
│   ├── coverage.sh
│   ├── envtpl
│   └── tdd.sh
├── conf
│   ├── about.json
│   ├── default.yml
│   ├── dev.yml
│   ├── production.yml
│   ├── staging.yml
│   └── vagrant.yml
├── deploy
│   ├── application.yml
│   ├── dev.yml
│   ├── production.yml
│   ├── staging.yml
│   └── vagrant.yml
├── deploy-test.py
├── docs
│   └── todo
├── index.go
├── main.go
├── main_test.go
├── middleware.go
├── server.go
├── skeleton-vars.yml
├── status.go
└── status_test.go
```

Pipeline as Code



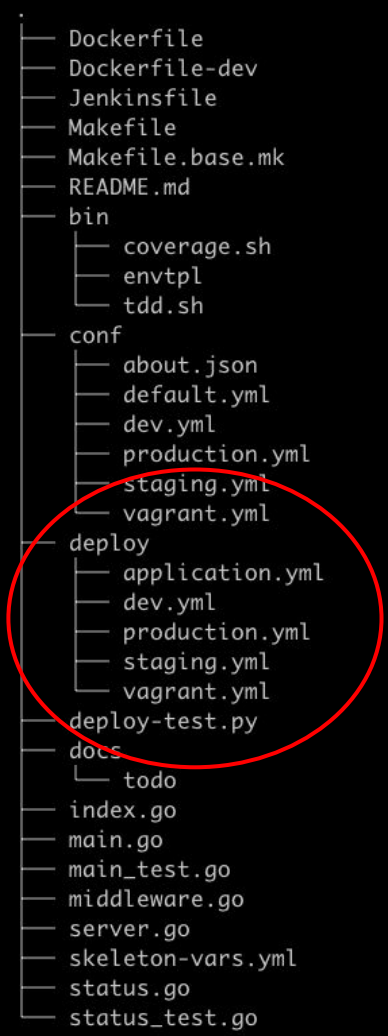

Scala

Packaging



Deployment Files

```
replicas: 1
resources:
   cpu: 2
   memory: 200M
healthChecks:
...
```

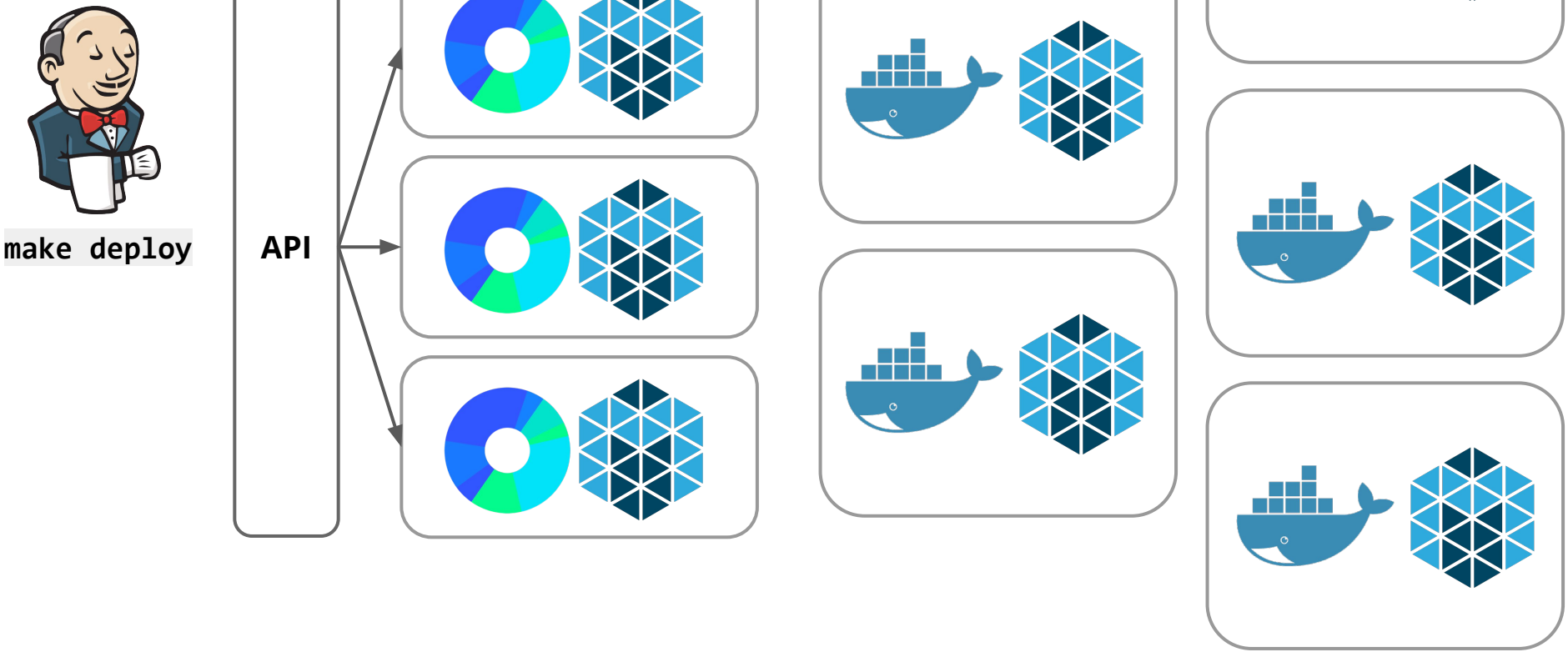


Makefile

- `make deploy-dev`
- `make deploy-staging`
- `make deploy-production`

Mesos

Marathon

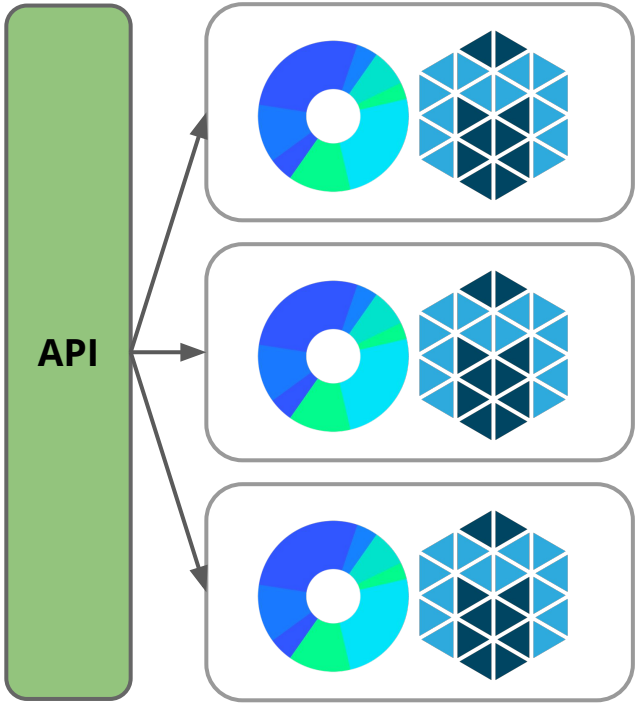

make deploy
API

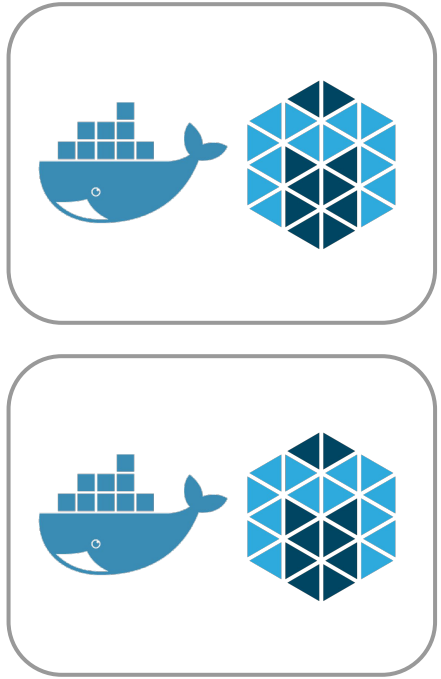
`make deploy`

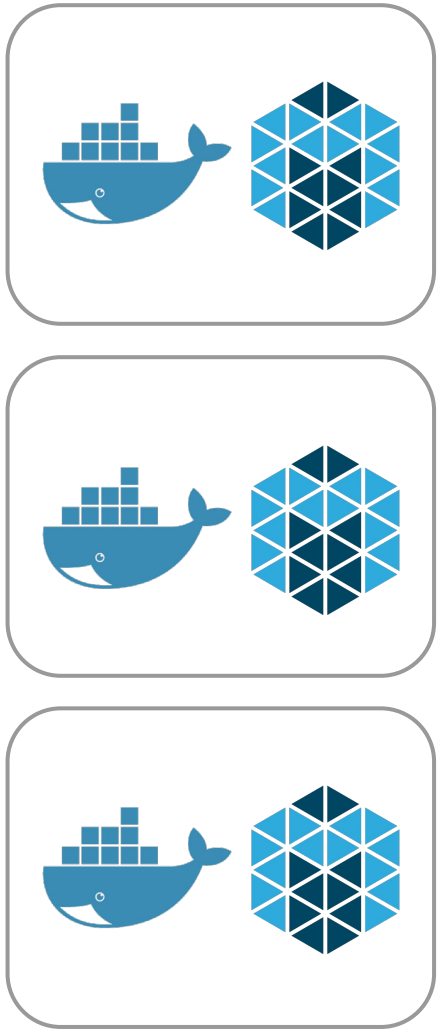
POST

```
{

"id":"service-1",
"cpus": 0.1,
"mem": 10.0,
"instances": 1

}
```

API







`make deploy`

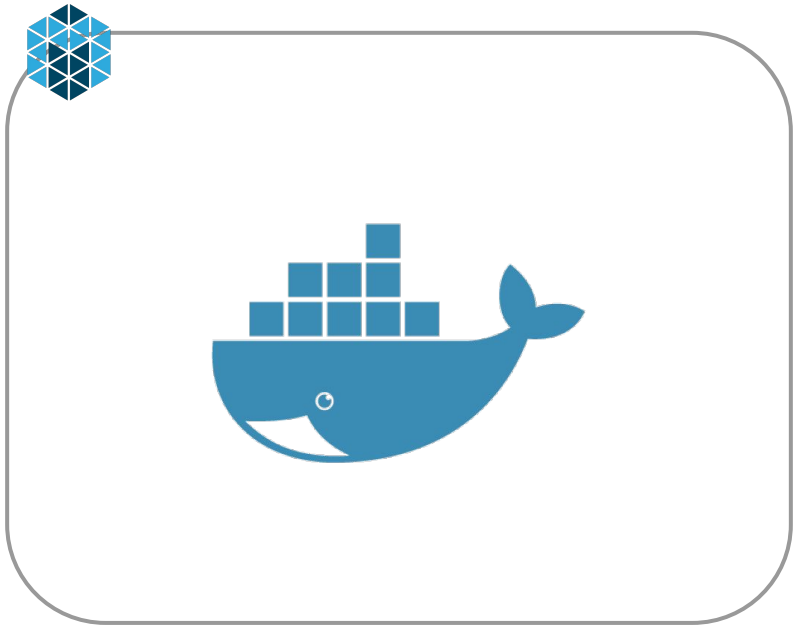
POST

```
{

"id":"service-1",
"cpus": 0.1,
"mem": 10.0,
"instances": 1

}
```

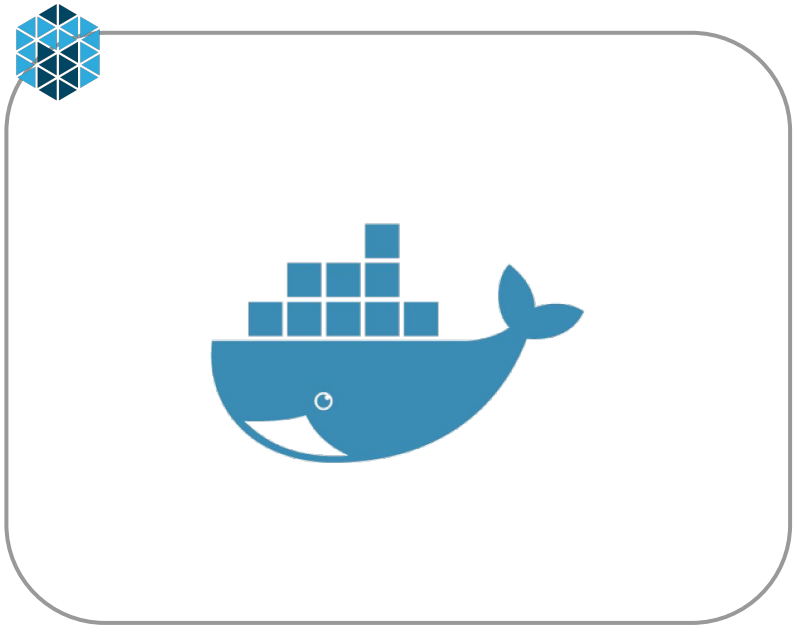
API

service-1

Routing

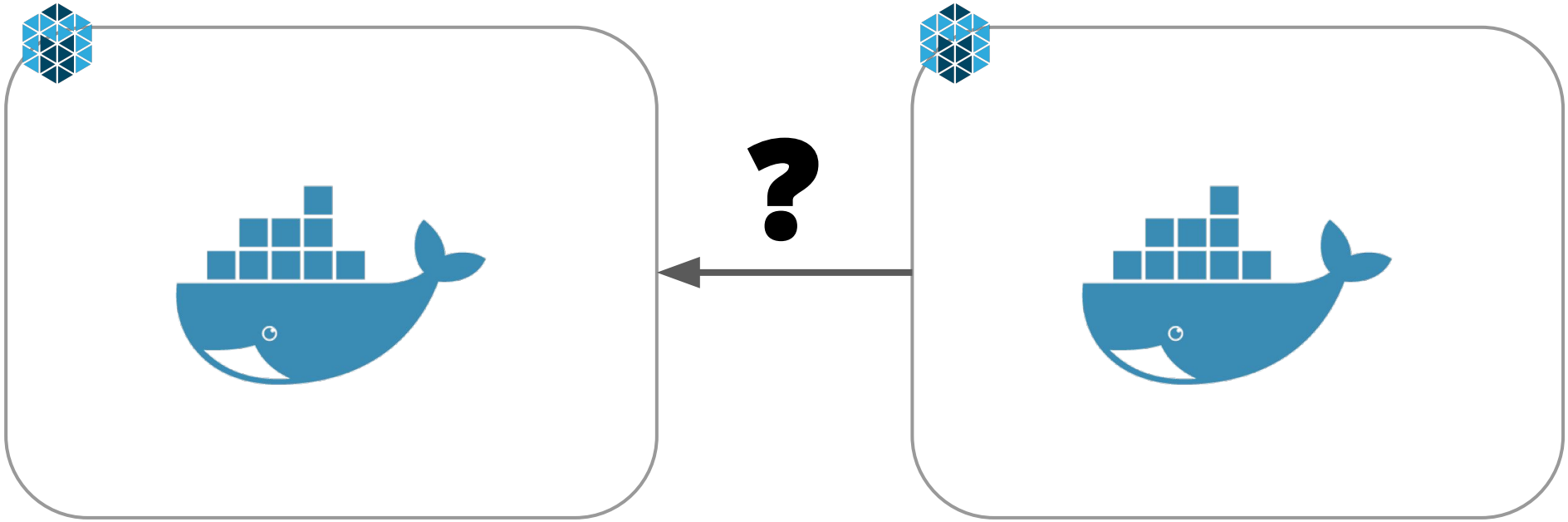


service-1
10.0.10.1



service-2
10.0.10.2

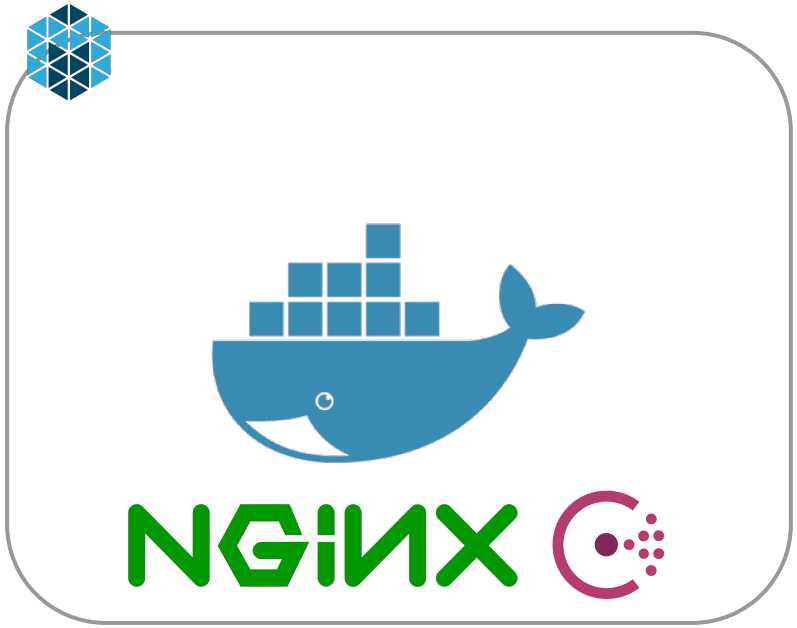
Routing



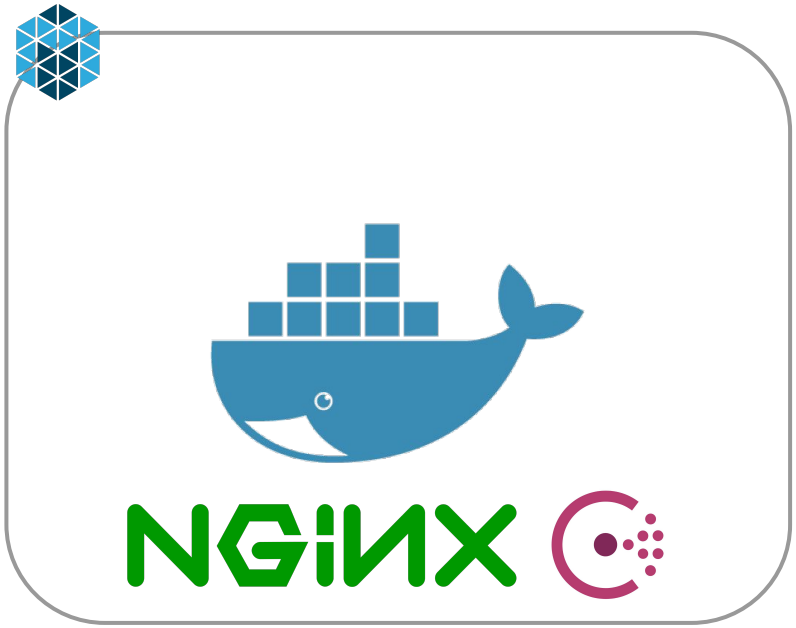
service-1
10.0.10.1

service-2
10.0.10.2

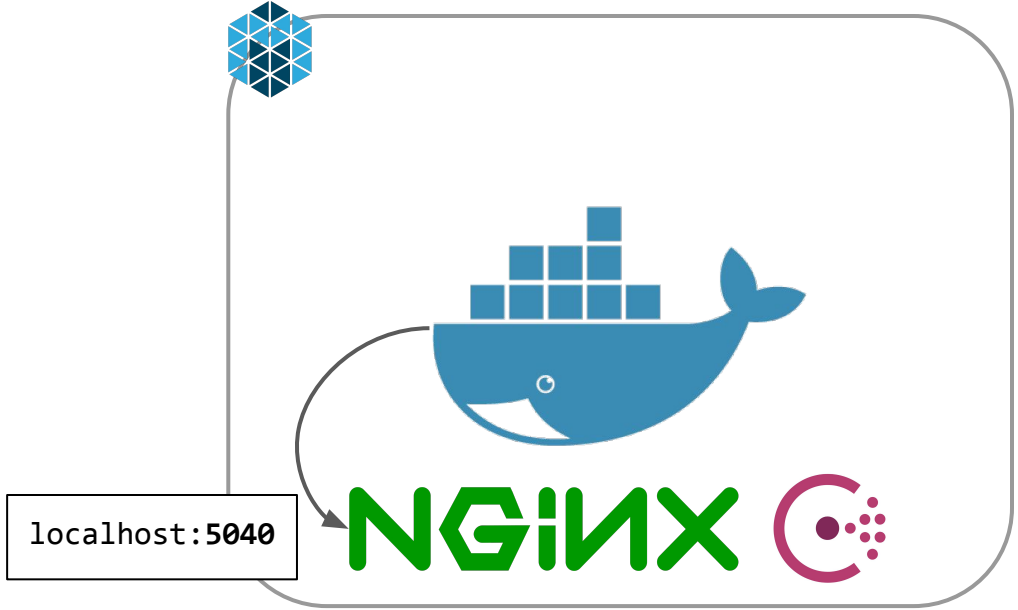
Routing - Fat Middleware



service-1
10.0.10.1



service-2
10.0.10.2

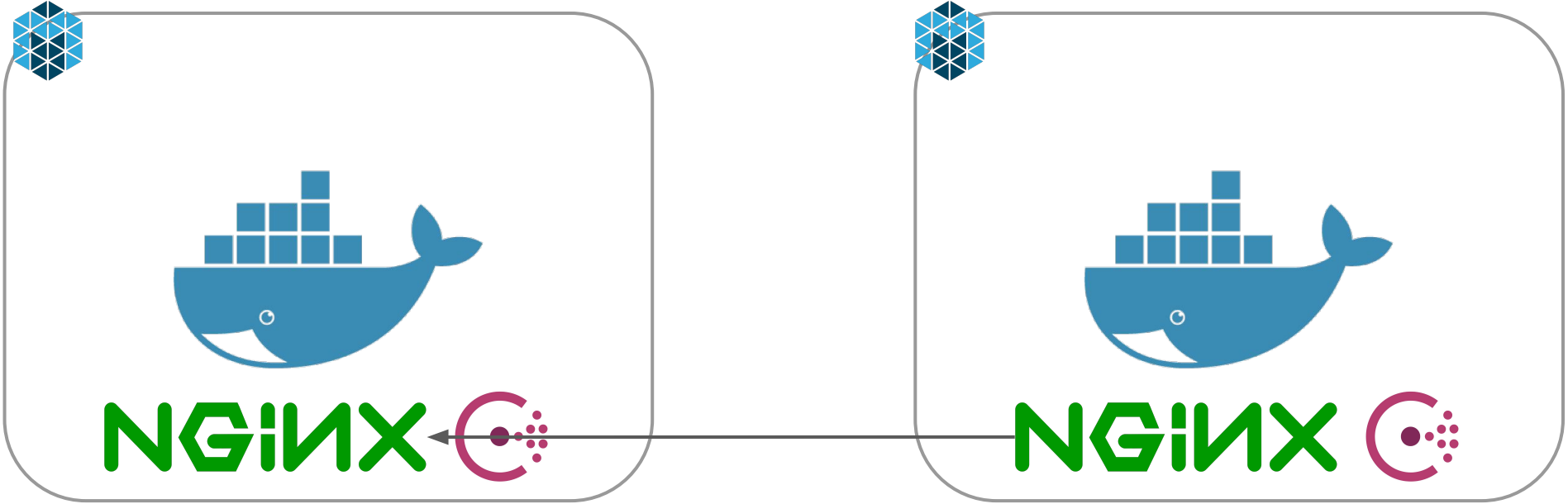


service-2

10.0.10.2

```
{
    upstream service-1 {
       server 10.0.10.1:5041;
       ....
     }
}
```

```
curl http://localhost:5040/service/service-1/endpoint
```

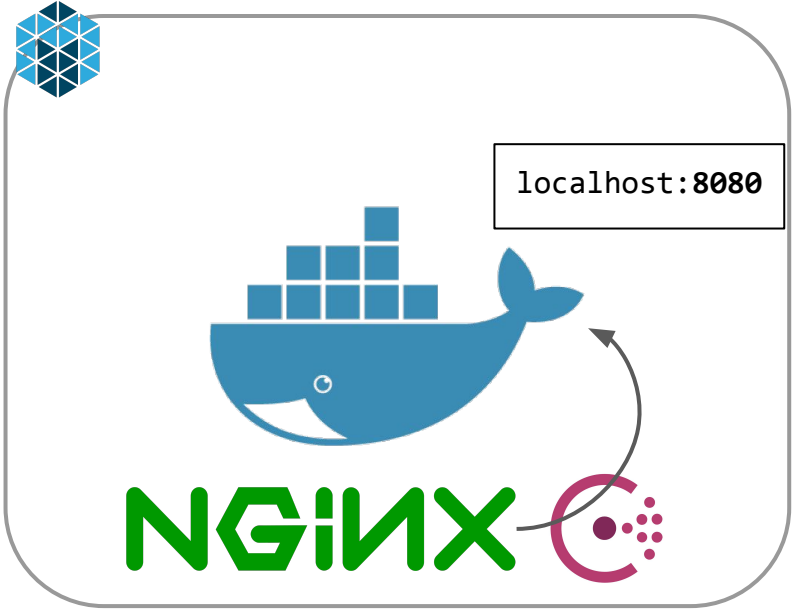


service-1
10.0.10.1

service-2
10.0.10.2

```
curl https://10.0.10.1:5041/service/service-1/endpoint
```

```
{
    upstream service-1 {
       server 10.0.10.1:5041;
       ....
     }
}
```



service-1

10.0.10.1

service-2

10.0.10.2

```
{
    upstream service-1 {
       server 10.0.10.1:5041;
       ....
     }
}
```


30
20
10
0
4/1/2016
7/1/2016
10/1/2016
1/1/2017
4/1/2017


30
20
10
0
4/1/2016
7/1/2016
10/1/2016
1/1/2017
4/1/2017

Why Kubernetes?

Hootsuite Development

Co-ops High School Students Development

Go to Hootsuite.com

Careers

Migrating Container Orchestrators – Mesos, Kubernetes, or Nomad?

Hootsuite's recent transition from Monolith to Microservices, like any large scale change, has met many challenging issues. As we move towards a SOA (Service Oriented Architecture), we need to build new infrastructure, tools, and pipelines. This new architecture requires our web applications to be made up of many small components, which can be done via containerization. Containers then need to be orchestrated in order to truly run as a distributed system. An issue arose when our initial container orchestrator choice slowly became outclassed by other alternatives. Looking at where we wanted to be in the near future, we decided to migrate to a new platform. This blog will describe our technical decisions that drove this change.

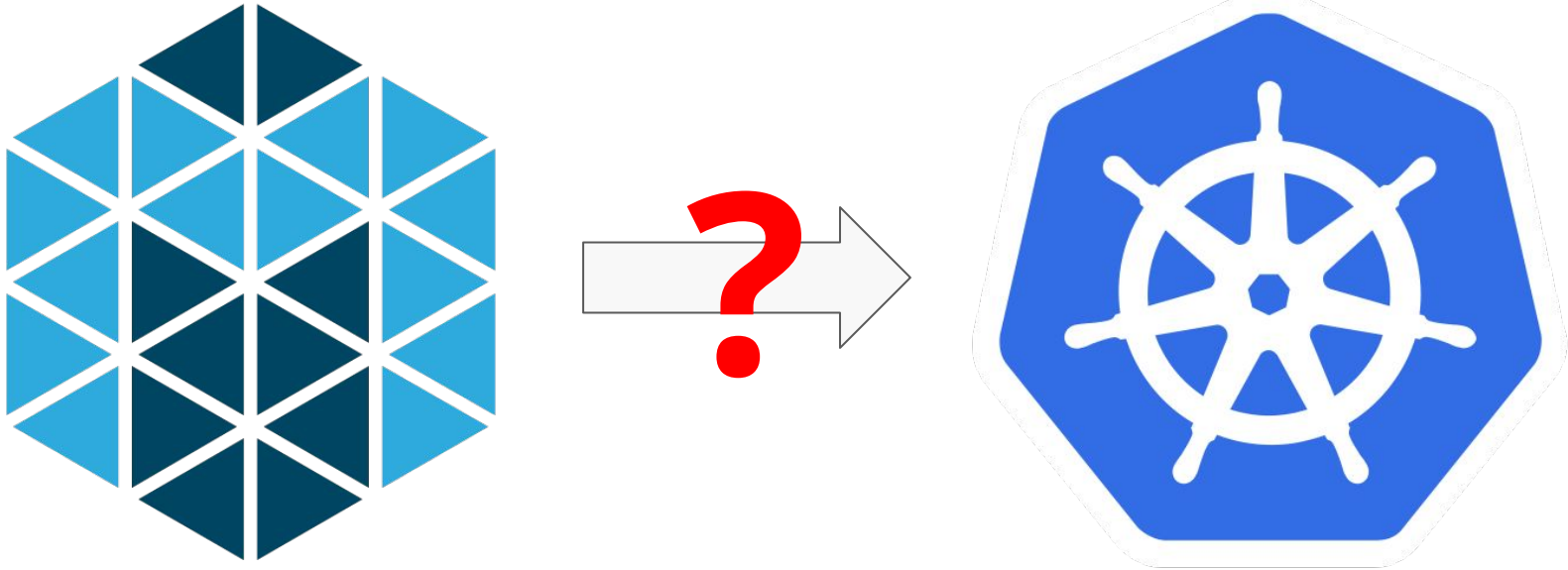
What is container orchestration/scheduling? Why should you care?

Consider a web developer who just finished building a simple PHP application. He has made sure every component functions properly on his localhost server. He uploads his code and assets to a host on the internet for the public, but is he guaranteed that everything will function the same as it did locally? The answer is 'no'. The system environment could be very different between his local server and the external internet server. There might be missing dependencies, or the operating system could be entirely different. This is where a containerization tool called Docker comes in. Docker solves the inconsistency issue by stuffing the *entire* environment, along with the web app, into a "Docker image". Any server with Docker installed can then run this image, regardless of the environment.

4 months
x




K8S DANCE!



Microservices on Mesos and Marathon

- Project Skeleton
  - Golang or Scala
- Pipeline as Code
  - Jenkinsfile
  - Makefile
- Docker images for packaging
- API on top of Marathon
- Dynamic service discovery
- Fat middleware using Consul and NGINX

Microservices on Kubernetes

- Project Skeleton
  - Golang or Scala
- Pipeline as Code
  - Jenkinsfile
  - Makefile
- Docker images for packaging
- *API on top of Marathon*
- Dynamic service discovery
- Fat middleware using Consul and NGINX
- Documentation for getting started
- ./mesos2k8s

./mesos2k8s

```
->./mesos2k8s
🏃 mesos2k8s 🏃
▶️ Skytrain Deployment Directory: deploy
▶️ Kubernetes Deployment Directory: kubernetes
```

```
.
├── Dockerfile
├── Dockerfile-dev
├── Jenkinsfile
├── Makefile
├── Makefile.base.mk
├── README.md
├── bin
│   ├── coverage.sh
│   ├── envtpl
│   └── tdd.sh
├── conf
│   ├── about.json
│   ├── default.yml
│   ├── dev.yml
│   ├── production.yml
│   ├── staging.yml
│   └── vagrant.yml
├── deploy
│   ├── application.yml
│   ├── dev.yml
│   ├── production.yml
│   ├── staging.yml
│   └── vagrant.yml
├── deploy-test.py
├── docs
│   └── todo
├── index.go
├── main.go
├── main_test.go
├── middleware.go
├── server.go
├── skeleton-vars.yml
├── status.go
└── status_test.go
```

```
.
├── Dockerfile
├── Dockerfile-dev
├── Jenkinsfile
├── Makefile
├── Makefile.base.mk
├── README.md
├── bin
│   ├── coverage.sh
│   └── tdd.sh
├── conf
│   ├── about.json
│   ├── default.yml
│   ├── dev.yml
│   ├── local.yml
│   ├── production.yml
│   └── staging.yml
├── docs
│   ├── CONTRIBUTING.md
│   └── todo
├── index.go
├── kubernetes
│   ├── application.yml
│   ├── dev.yml
│   ├── minikube.yml
│   ├── production.yml
│   └── staging.yml
├── main.go
├── main_test.go
├── middleware.go
├── policy.hcl
├── server.go
├── skeleton-vars.yml
├── status.go
└── status_test.go
```

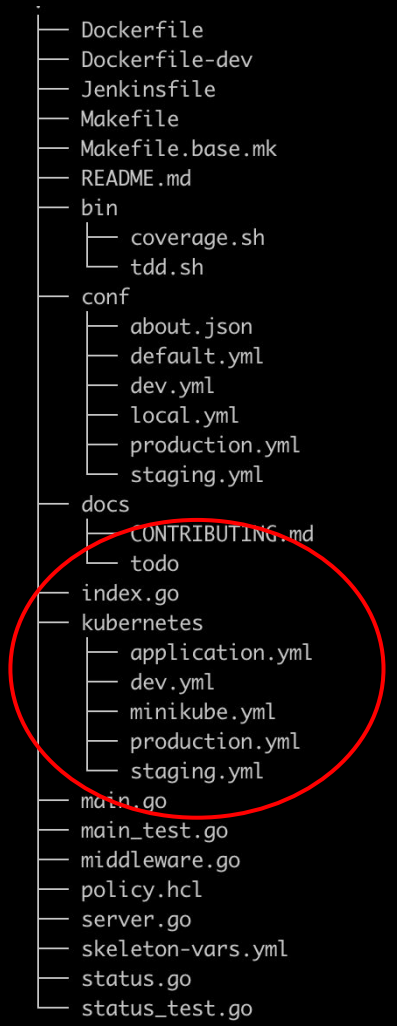
Project Skeleton

```
################################################
# Modify these vars

# Name of your service, lowercase, dash separated. ie. [a-z\-]
# !!! SHOULD NOT have "service" as prefix or suffix. We're trying to get rid of that.
# Will be used for Skyline and Kubernetes names as well as statsd prefixes, etc.
#name: my-awesome-service
name: new-service


# Human readable title. Used for docs and such
#nameNice: My Awesome Service
nameNice: My New Service

# Long form description
#description: Skyline sample service generated from the skeleton based off of Play! framework, dockerized, and
description: Hello World

#maintainers:
#  - name: Firstname Lastname
#    email: firstname.lastname@hootsuite.com
#  - name: Joe Blough
#    email: joe.blough@hootsuite.com
maintainers:
  - name: Anubhav Mishra
    email: anubhav.mishra@hootsuite.com

# Pick a project type. Can either be: scala, go, go-grpc, idl etc.
# This can be set with the command line arg `--project-type` which will allow you to
# create multiple project types with the same config. This is useful for projects like `go-grpc` + `idl`
# projectType: go
projectType: go
```

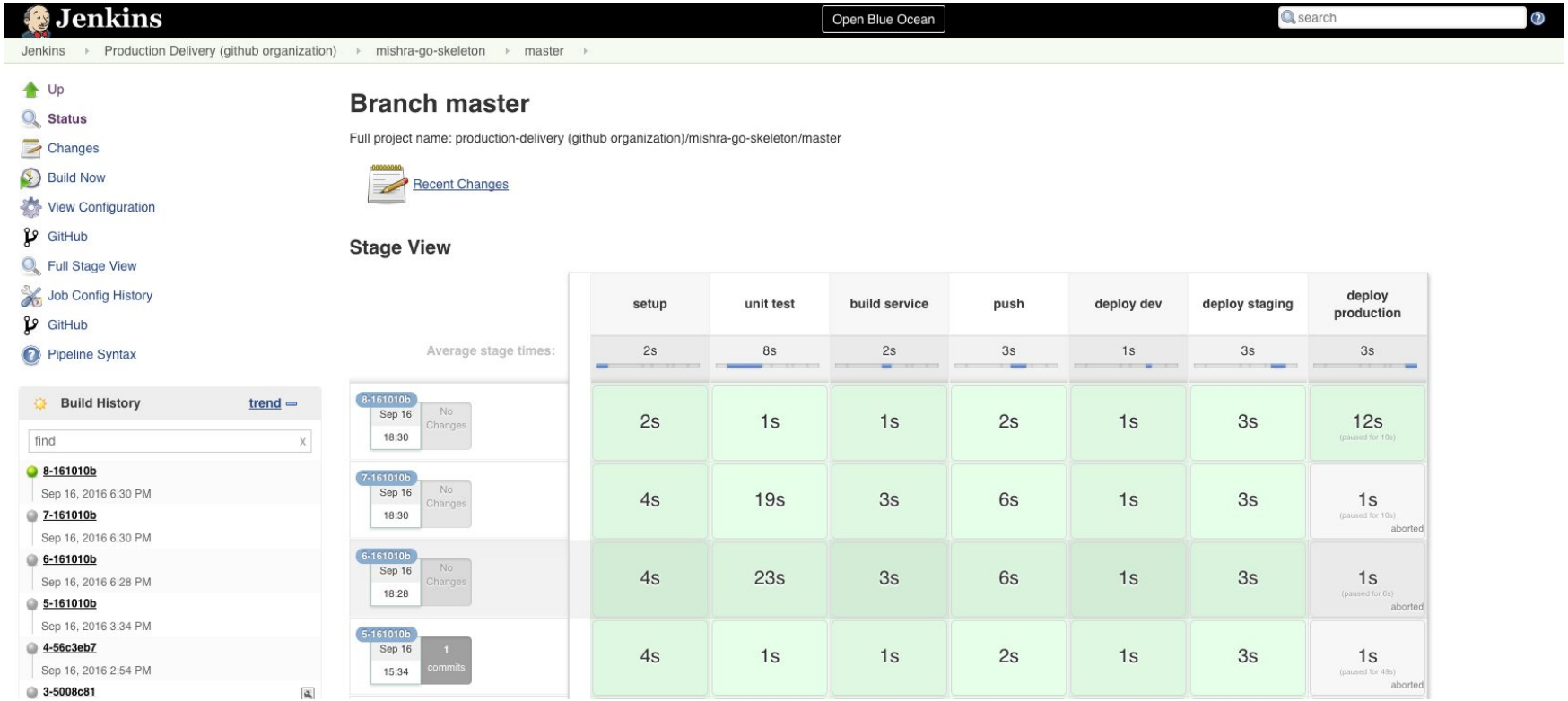
```
.
├── Dockerfile
├── Dockerfile-dev
├── Jenkinsfile
├── Makefile
├── Makefile.base.mk
├── README.md
├── bin
│   ├── coverage.sh
│   └── tdd.sh
├── conf
│   ├── about.json
│   ├── default.yml
│   ├── dev.yml
│   ├── local.yml
│   ├── production.yml
│   └── staging.yml
├── docs
│   ├── CONTRIBUTING.md
│   └── todo
├── index.go
├── kubernetes
│   ├── application.yml
│   ├── dev.yml
│   ├── minikube.yml
│   ├── production.yml
│   └── staging.yml
├── main.go
├── main_test.go
├── middleware.go
├── policy.hcl
├── server.go
├── skeleton-vars.yml
├── status.go
└── status_test.go
```

Deployment Files

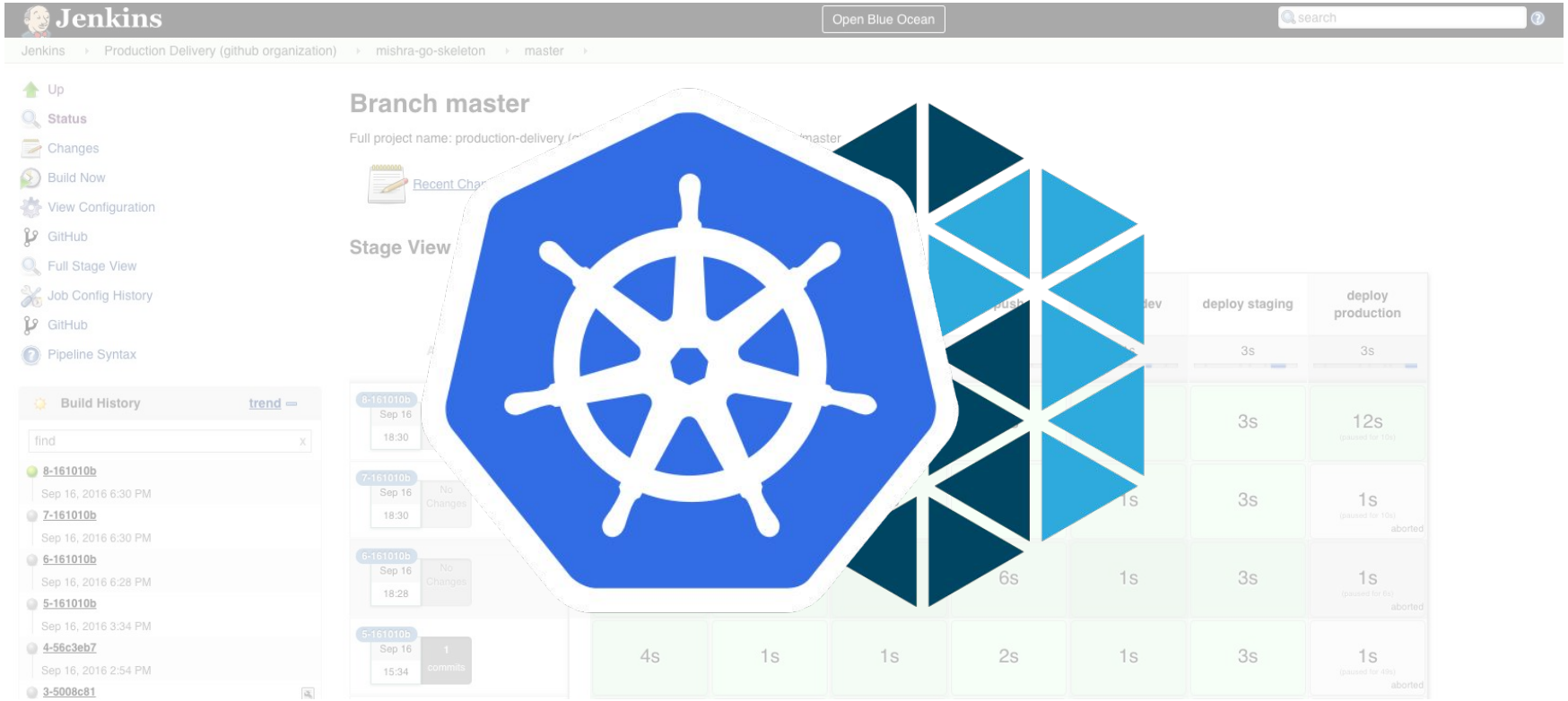
- `make deploy-k8s-dev`
- `make deploy-k8s-staging`
- `make deploy-k8s-production`



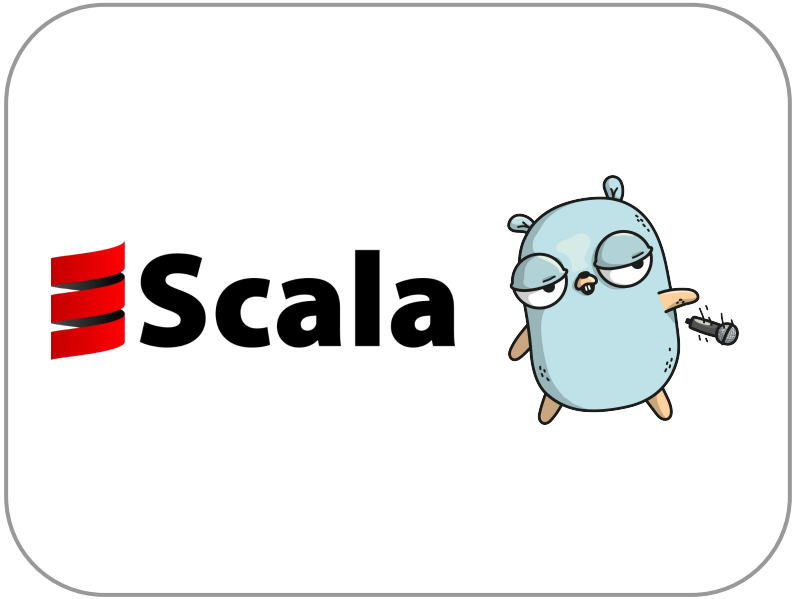
Pipeline as Code



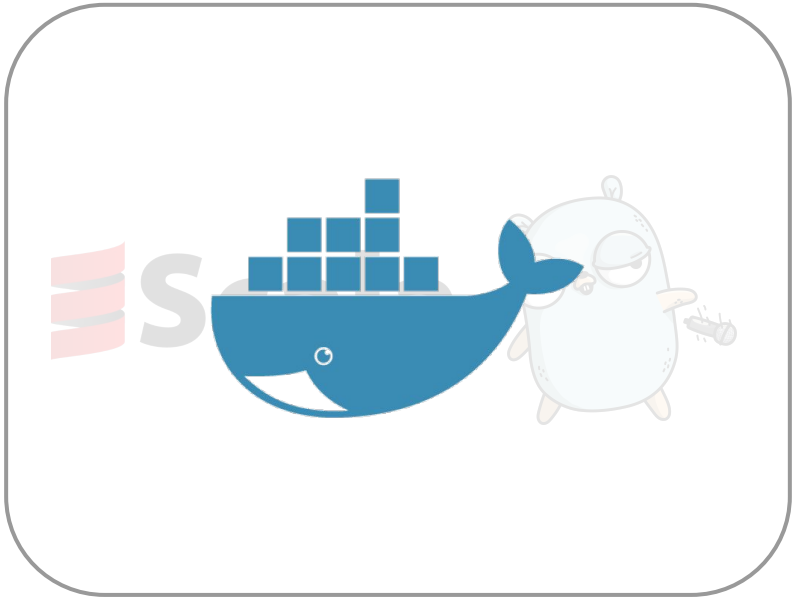
Pipeline as Code



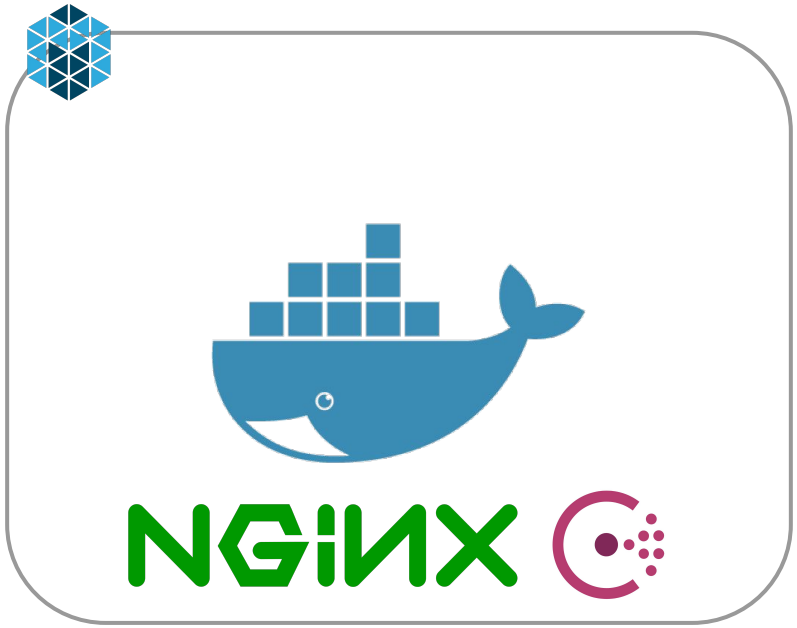
Packaging



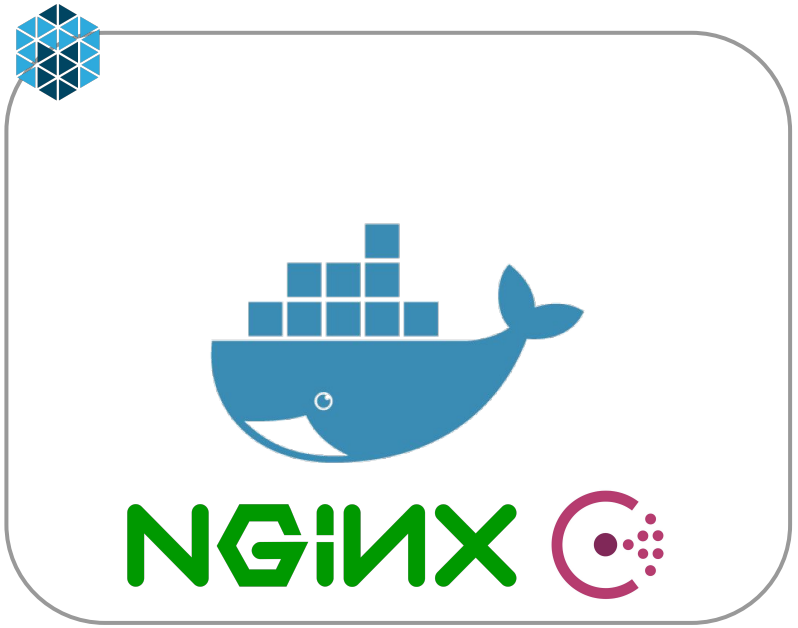
Packaging



Routing in Mesos

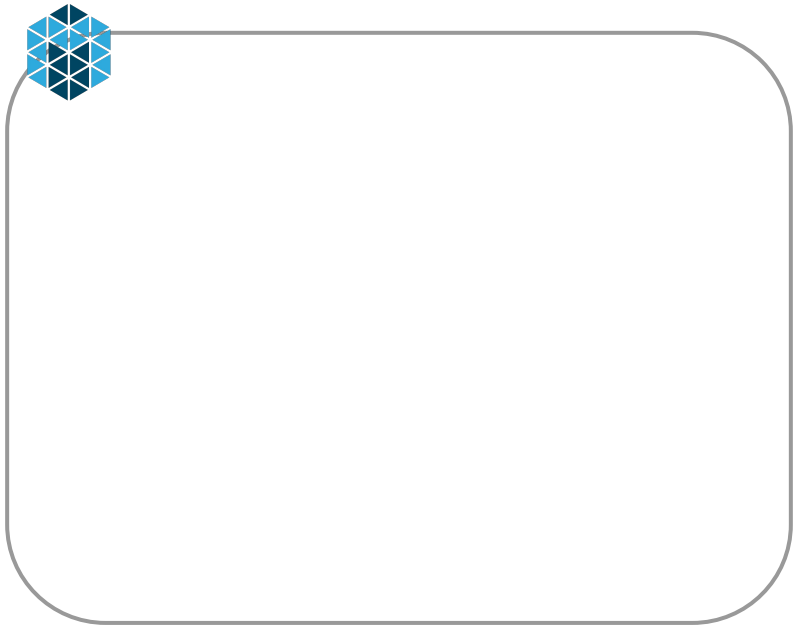


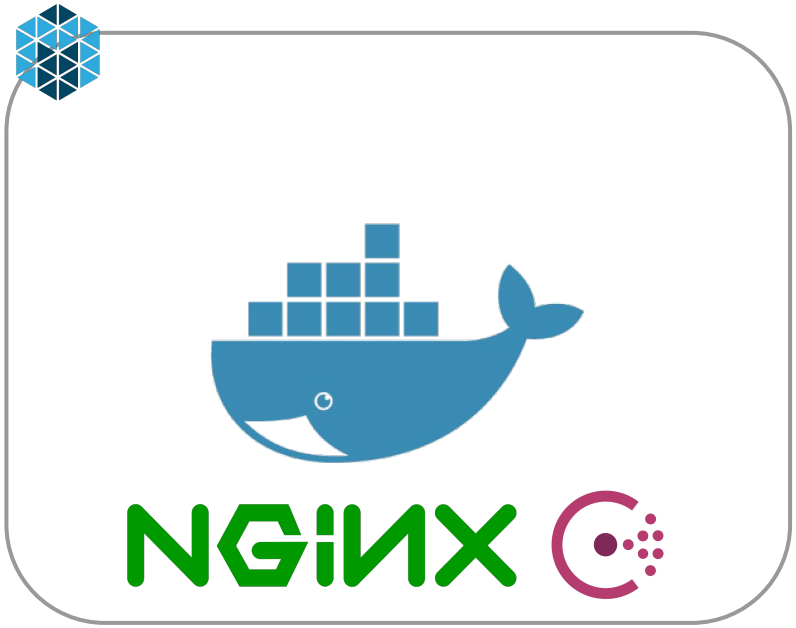
service-1
10.0.10.1



service-2
10.0.10.2

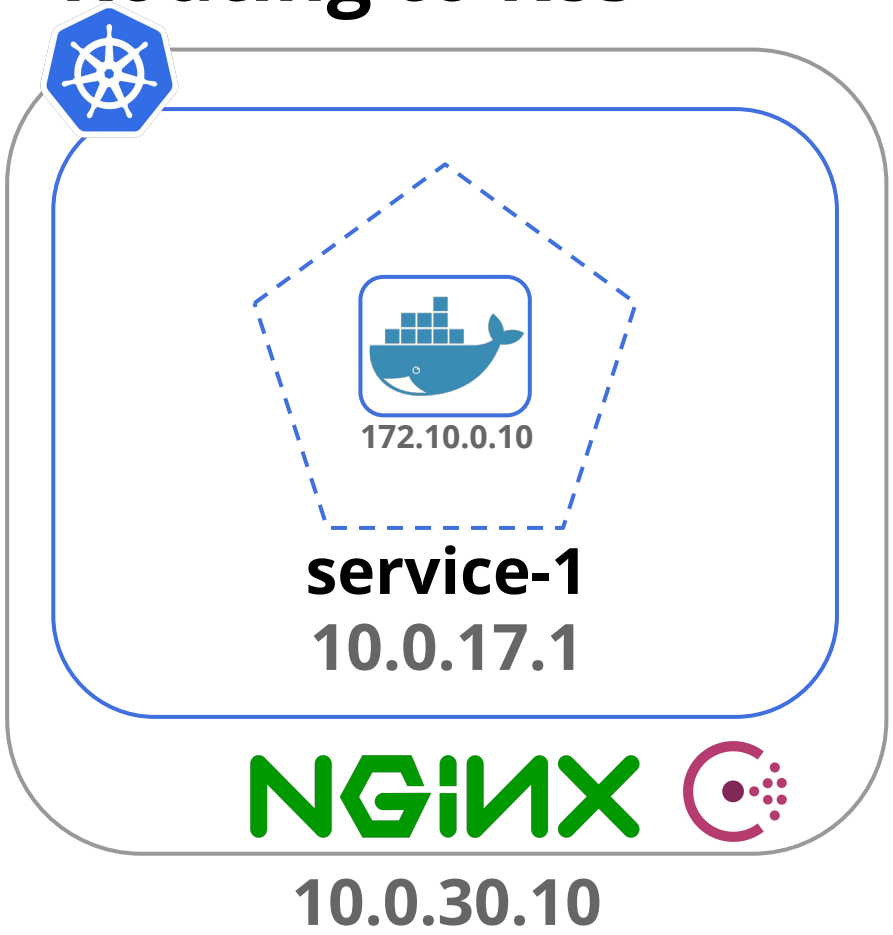
Routing in Mesos

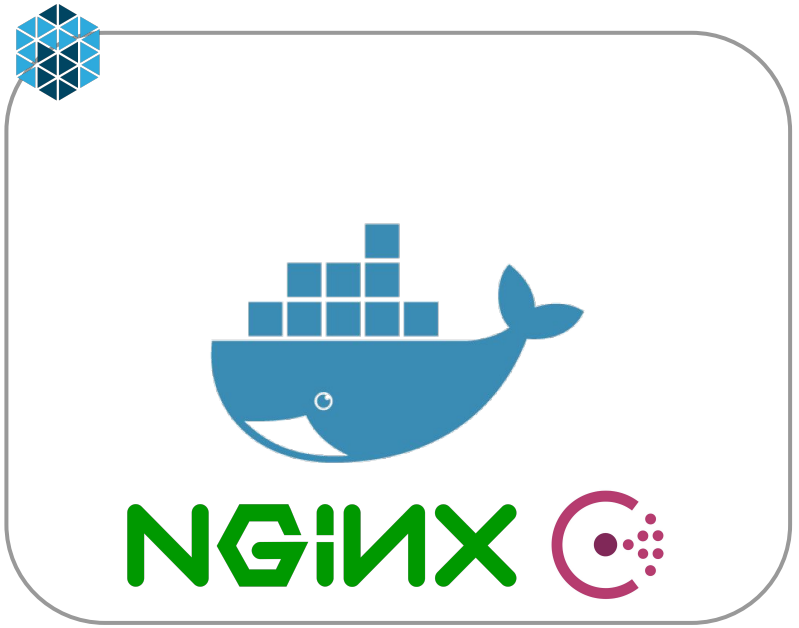




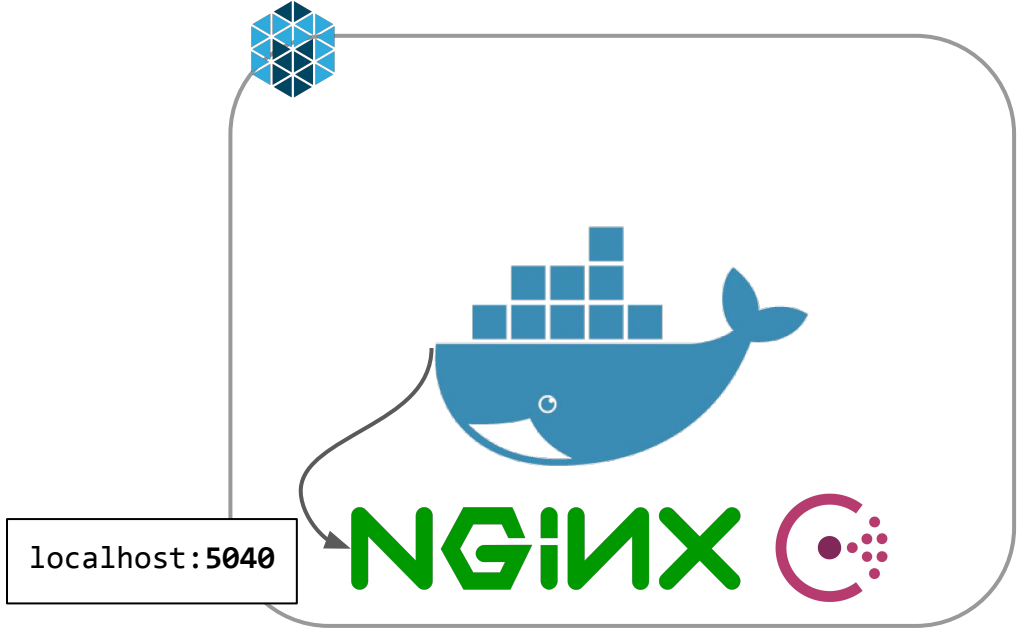
service-2
10.0.10.2

Routing to K8s





service-2
10.0.10.2

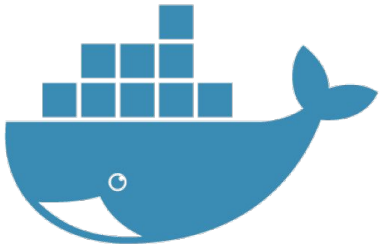


service-2

```
{
    upstream service-1 {
    }
    upstream bridge-1 {
       server 10.0.20.1:5041;
       ....
     }
}
```

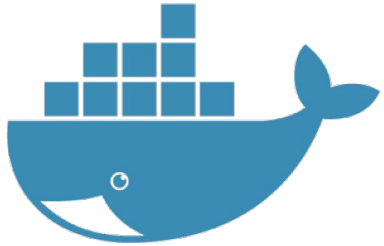
```
curl http://localhost:5040/service/service-1/endpoint
```

bridge-1
(multi-dc aware)



service-2

```
{
    upstream service-1 {
    }
    upstream bridge-1 {
       server 10.0.20.1:5041;
       ....
     }
}
```

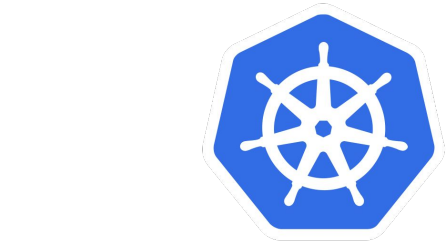


bridge-1
(multi-dc aware)

```
curl https://10.0.20.1:5041/service/service-1/endpoint
```

service-2

```
{
    upstream service-1 {
    }
    upstream bridge-1 {
       server 10.0.20.1:5041;
       ....
     }
}
```



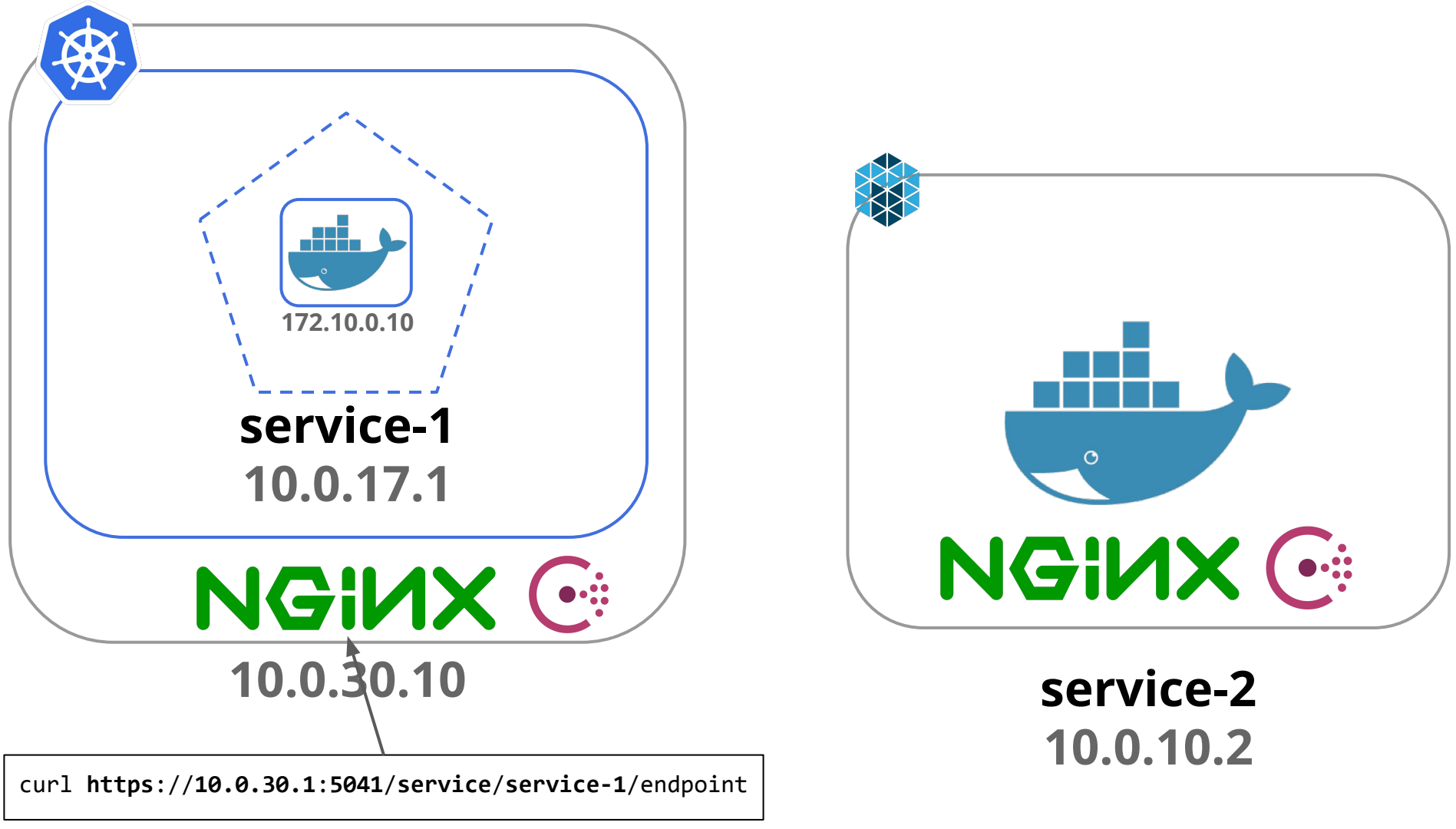
bridge-1

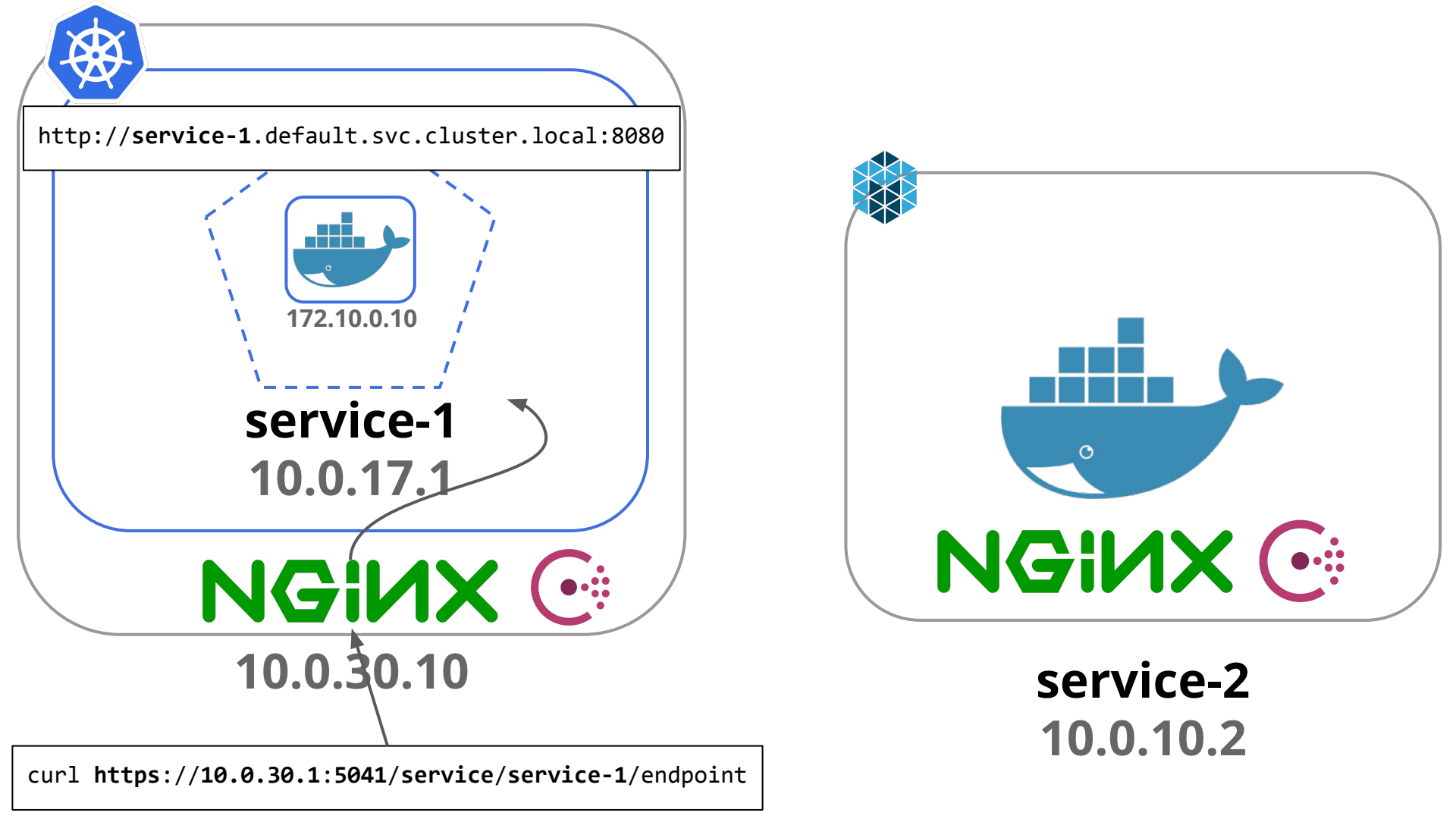
(multi-dc aware)

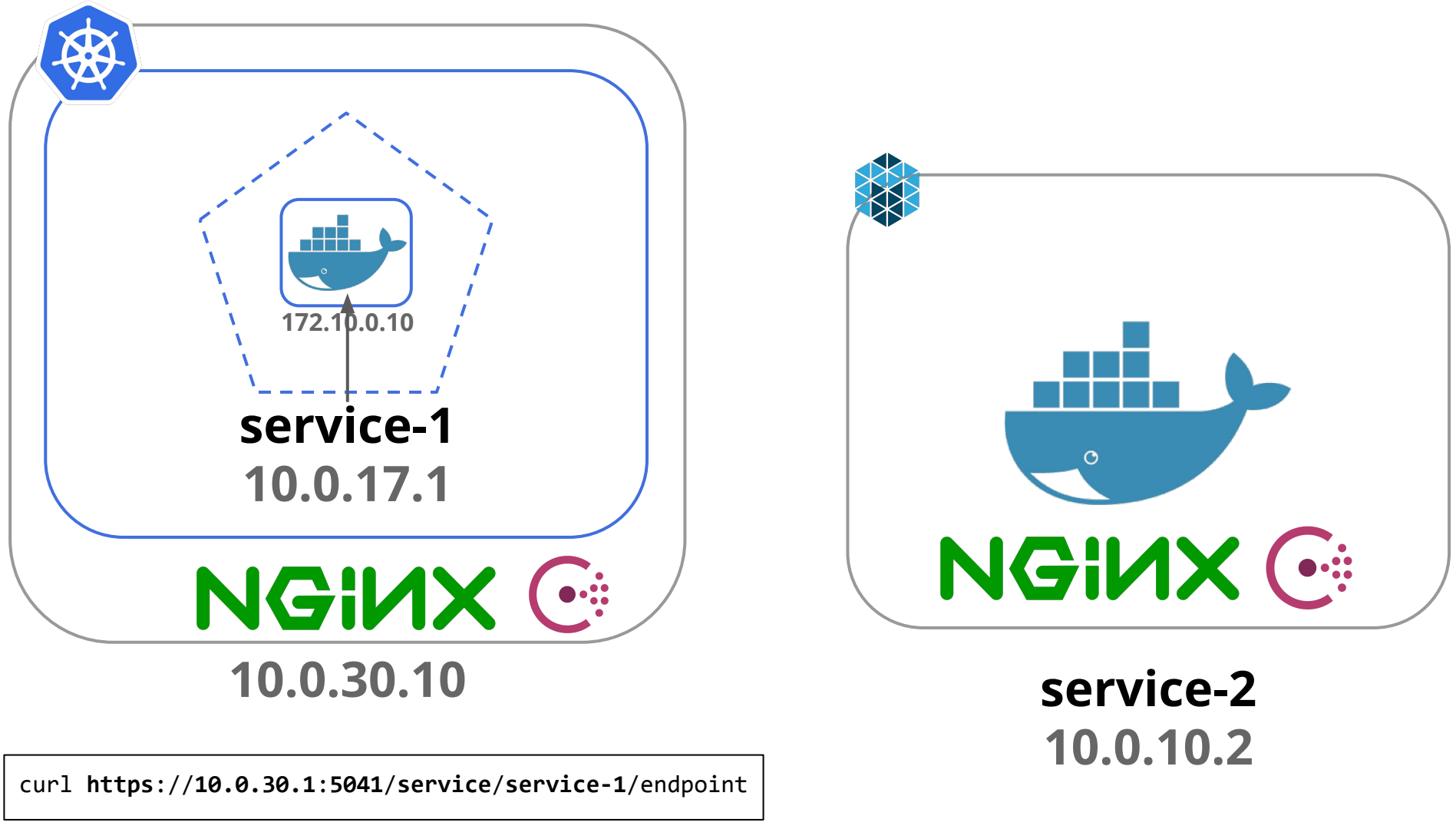
```
curl https://10.0.30.1:5041/service/service-1/endpoint
```

service-2

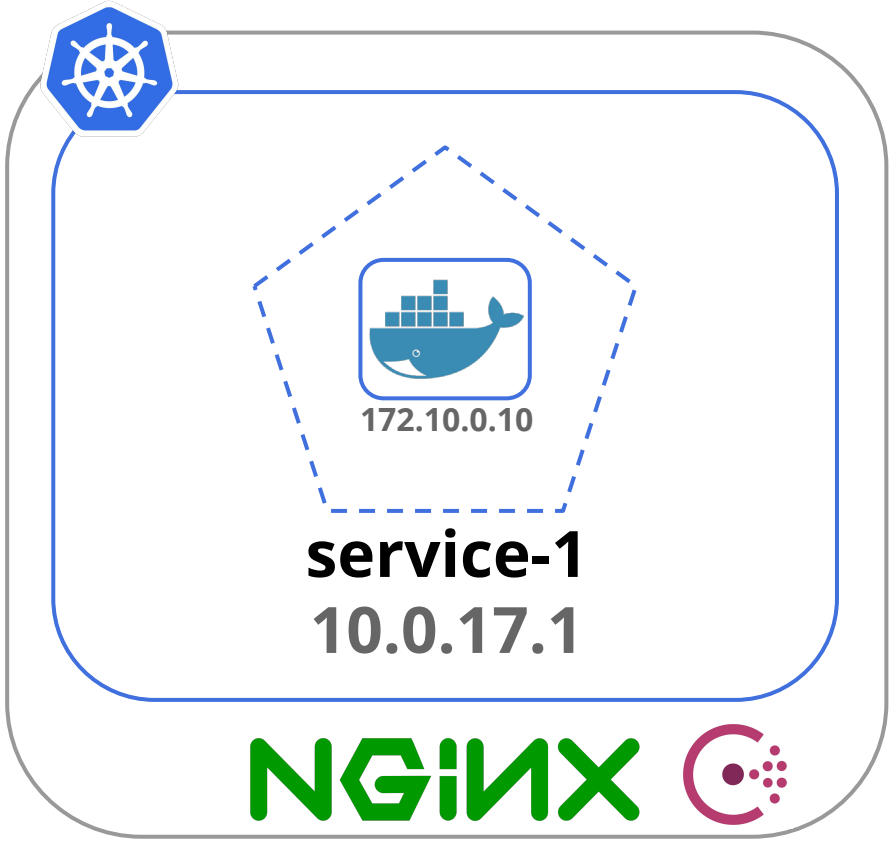
```
{
    upstream service-1 {
    }
    upstream bridge-1 {
       server 10.0.20.1:5041;
       ....
     }
}
```


172.10.0.10
service-1
10.0.17.1
10.0.30.10
service-2
10.0.10.2
curl https://10.0.30.1:5041/service/service-1/endpoint


http://service-1.default.svc.cluster.local:8080
172.10.0.10
service-1
10.0.17.1
NGINX
10.0.30.10
curl https://10.0.30.1:5041/service/service-1/endpoint
NGINX
service-2
10.0.10.2



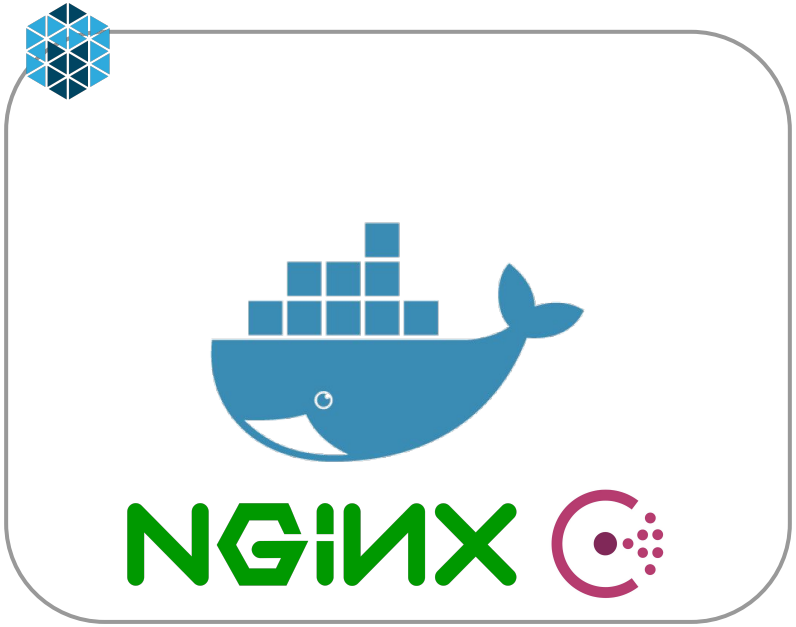

```
curl https://10.0.30.1:5041/service/service-1/endpoint
```



service-2
10.0.10.2

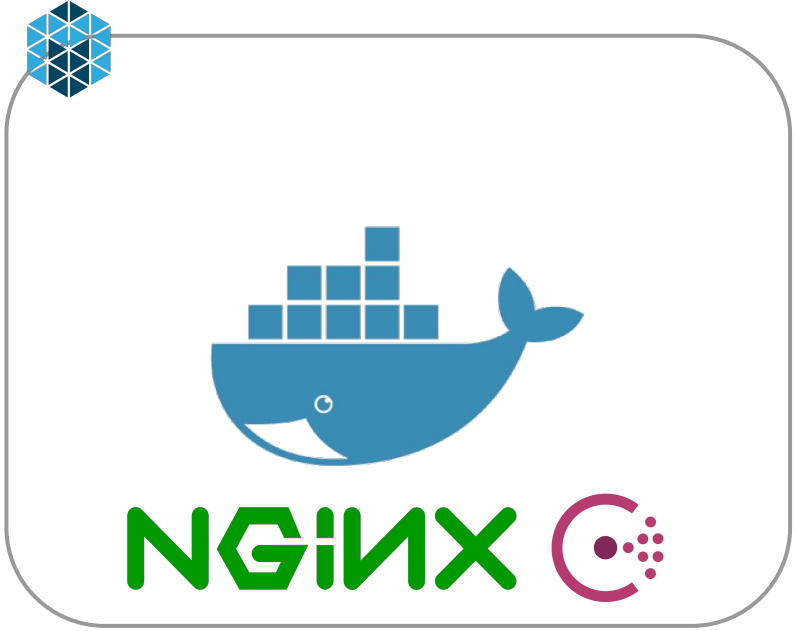
```
curl https://10.0.30.1:5041/service/service-1/endpoint
```

Rollback



service-1

10.0.10.1

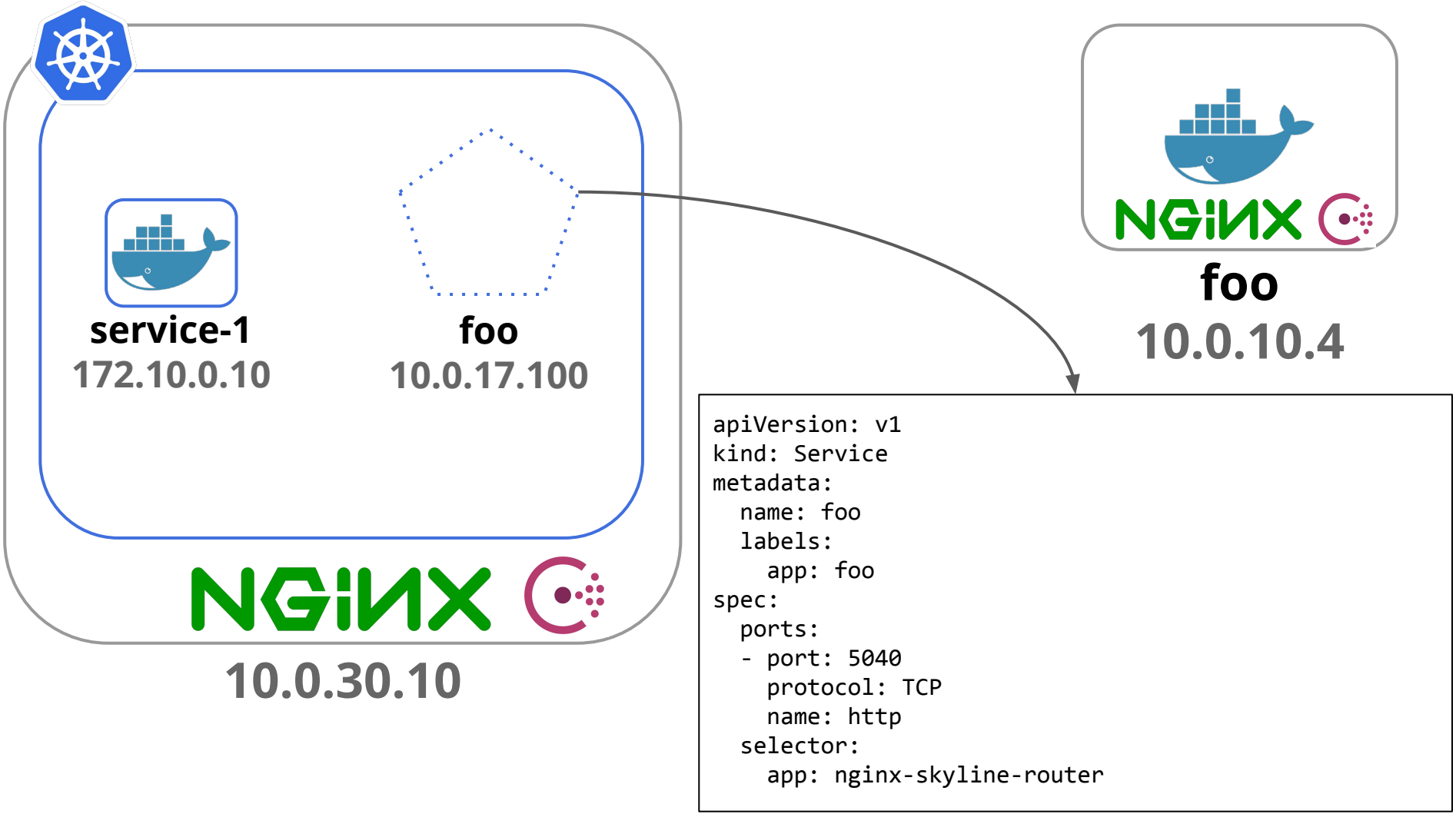


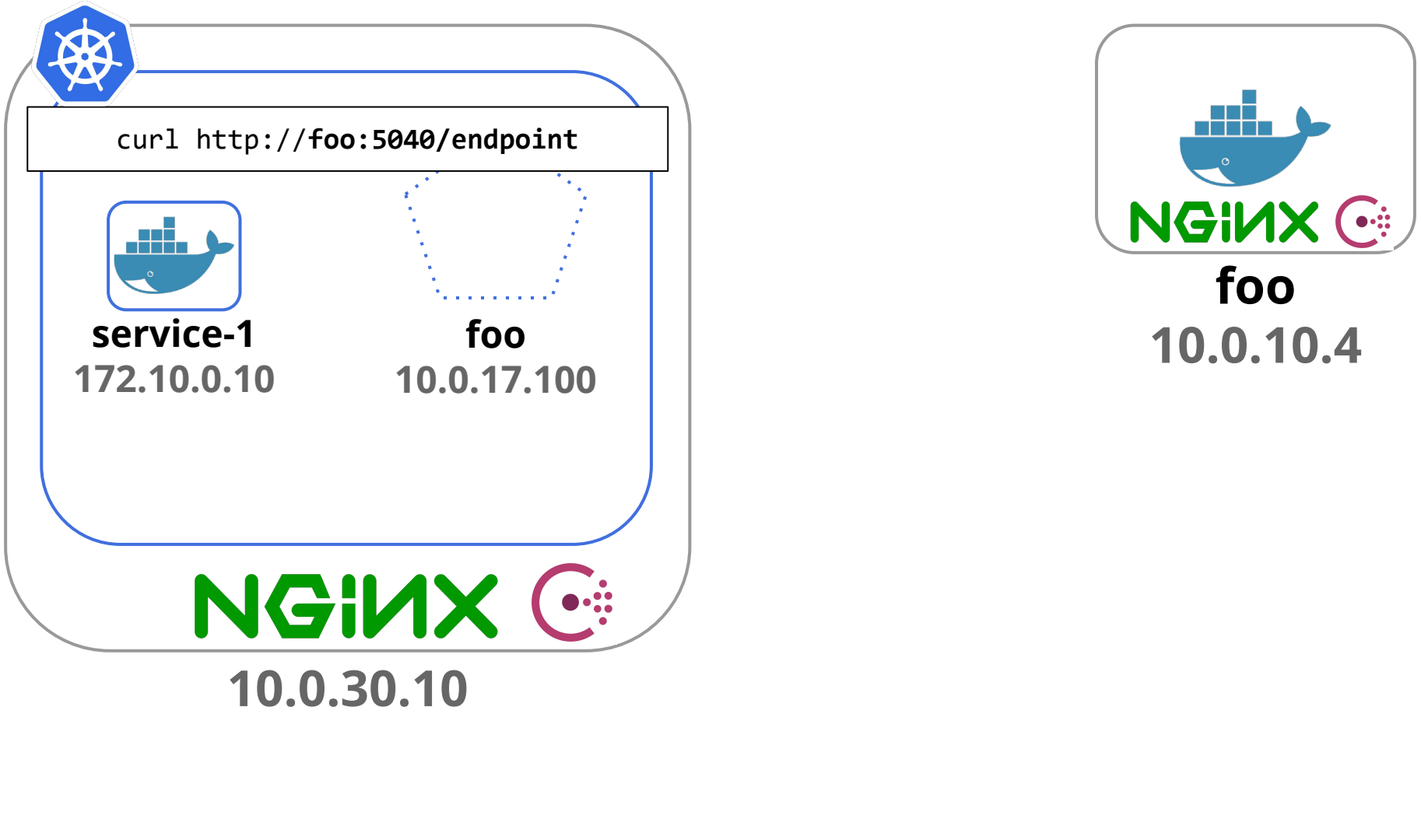
service-2

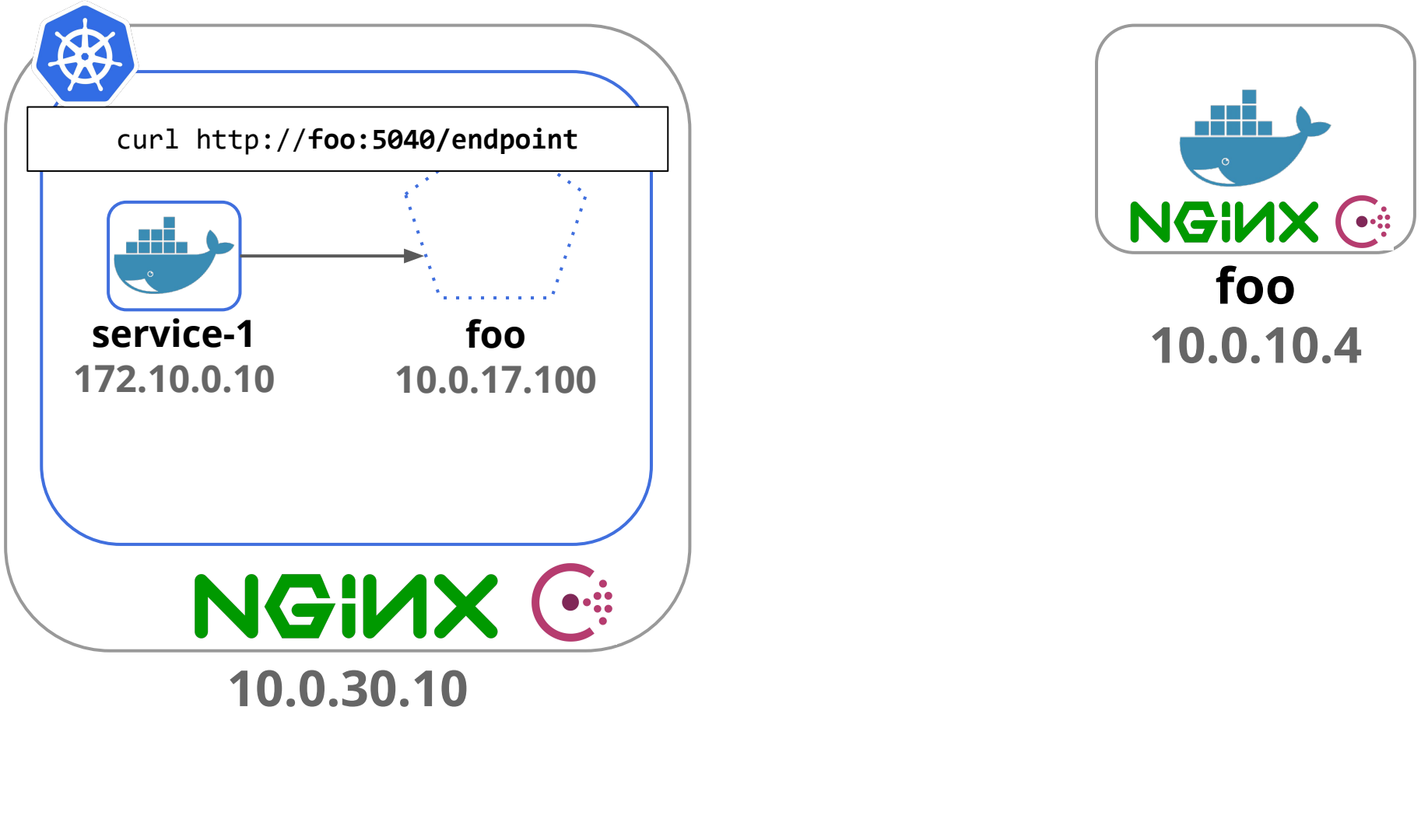
```
{
    upstream service-1 {
       server 10.0.10.1:5041;
    }
    upstream bridge-1 {
       server 10.0.20.1:5041;
       ....
     }
```

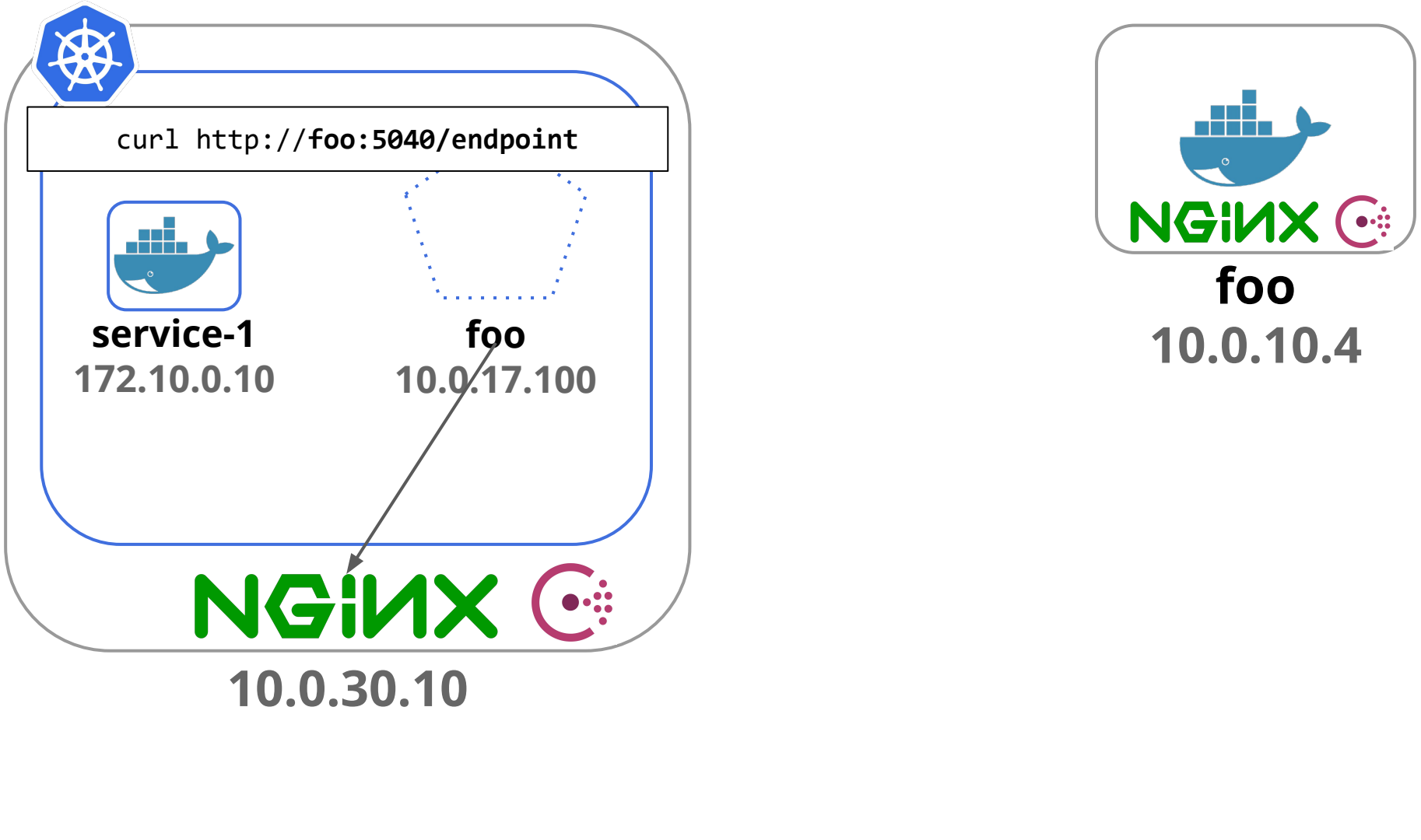

service-1
172.10.0.10
10.0.30.10

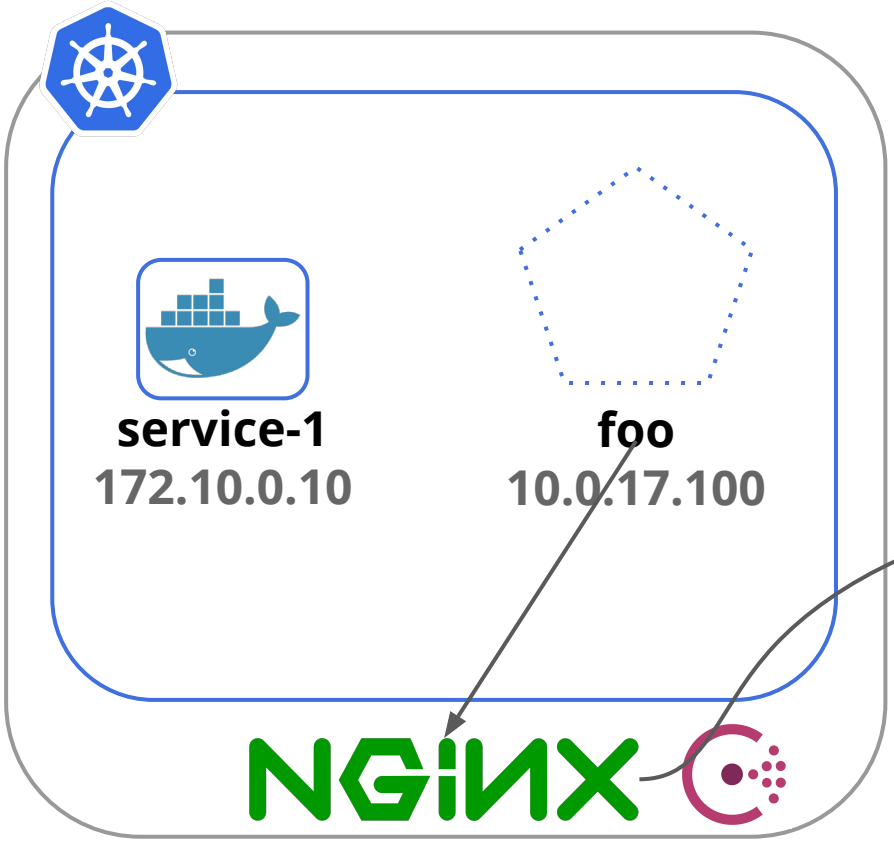

foo
10.0.10.4


service-1
172.10.0.10
foo
10.0.17.100
NGINX
10.0.30.10
NGINX
foo
10.0.10.4
apiVersion: v1
kind: Service
metadata:
name: foo
labels:
app: foo
spec:
ports:
- port: 5040
protocol: TCP
name: http
selector:
app: nginx-skyline-router

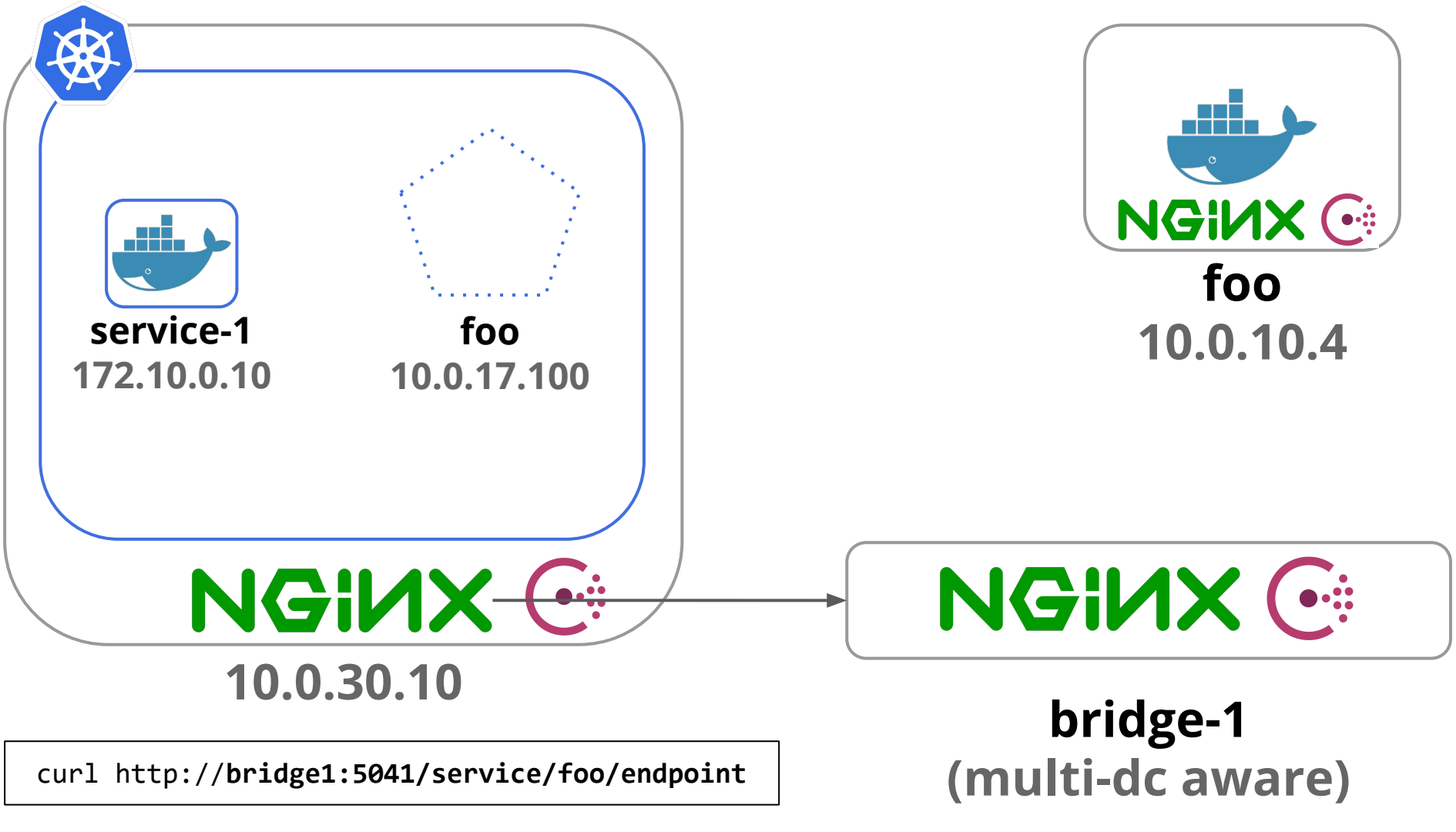

curl http://foo:5040/endpoint
service-1
172.10.0.10
foo
10.0.17.100
10.0.30.10
foo
10.0.10.4

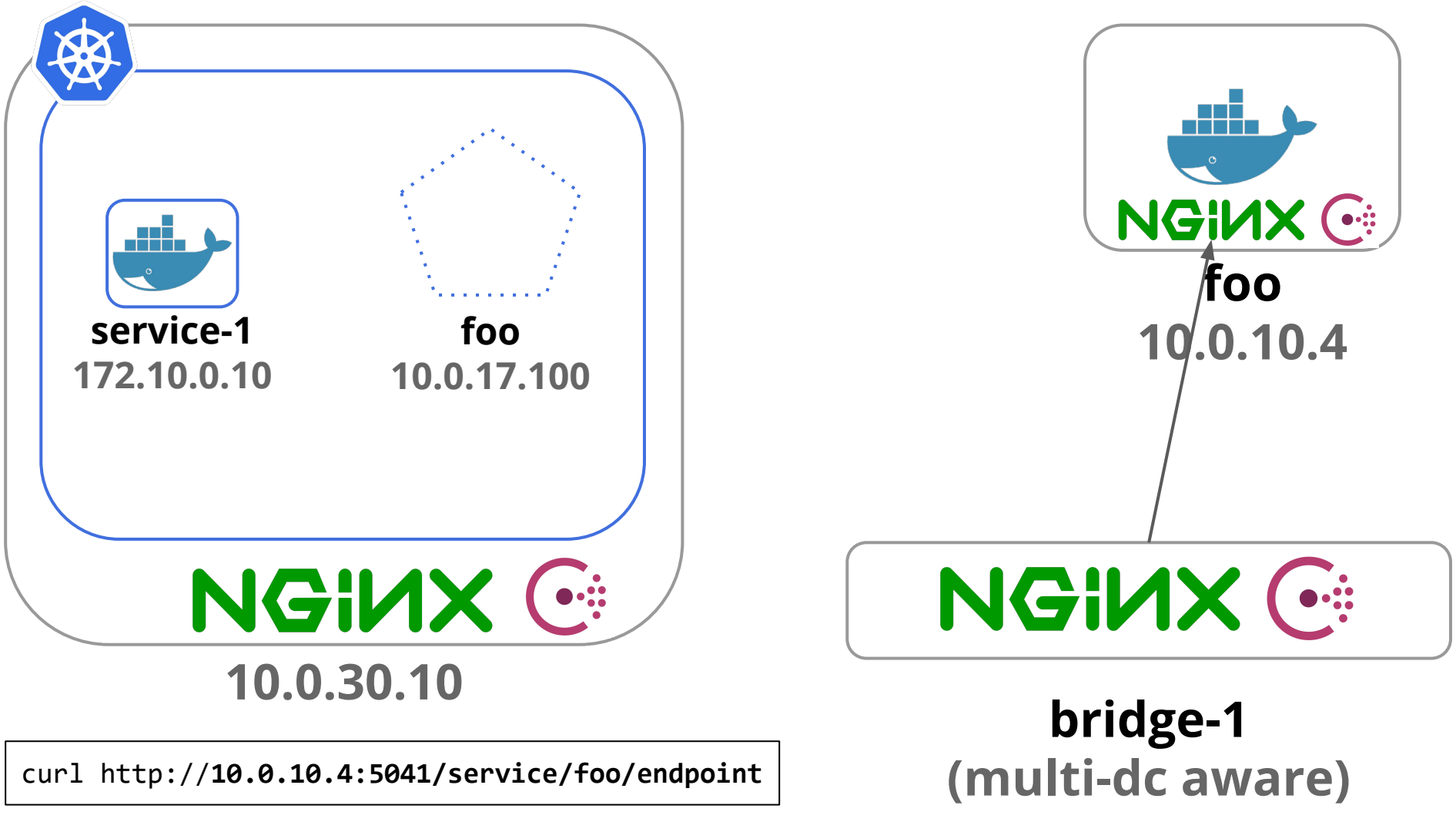

curl http://foo:5040/endpoint
service-1
172.10.0.10
foo
10.0.17.100
10.0.30.10
foo
10.0.10.4


curl http://foo:5040/endpoint
service-1
172.10.0.10
foo
10.0.17.100
10.0.30.10
foo
10.0.10.4



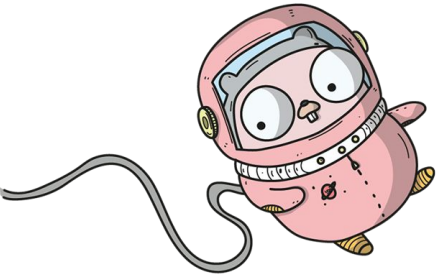

```
# match on:
# service.namespace.svc.cluster.local
# service.namespace
# service
server_name REGEX
....
location / {
     rewrite ^/(.*)$ /service/$service/$1 break;
     proxy_pass https://egress_bridge;
}
```

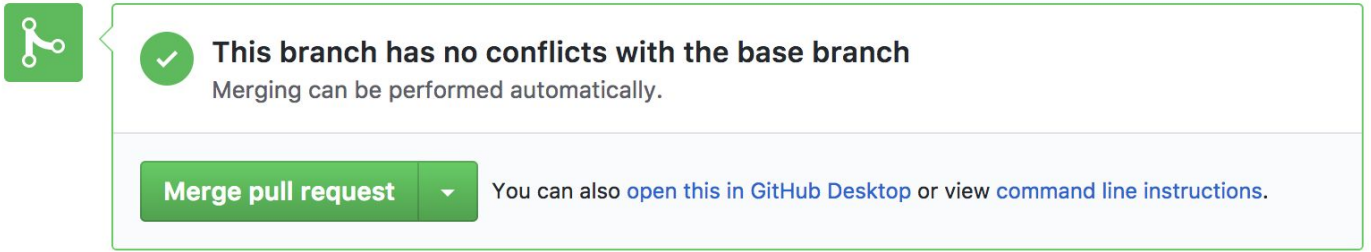

service-1
172.10.0.10
foo
10.0.17.100
NGINX
10.0.30.10
foo
10.0.10.4
NGINX
bridge-1
(multi-dc aware)
curl http://bridge1:5041/service/foo/endpoint




```
curl http://10.0.10.4:5041/service/foo/endpoint
```

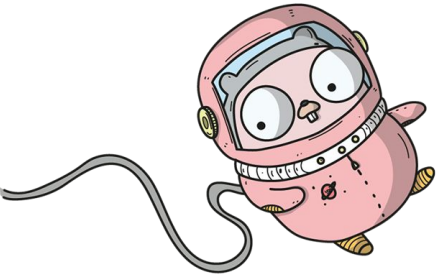
Ship it!



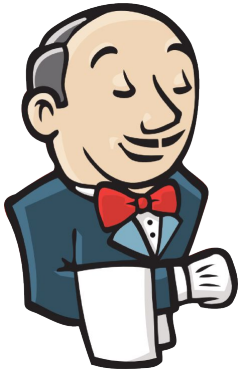



```
./project-generator
```

Microservice Pipeline



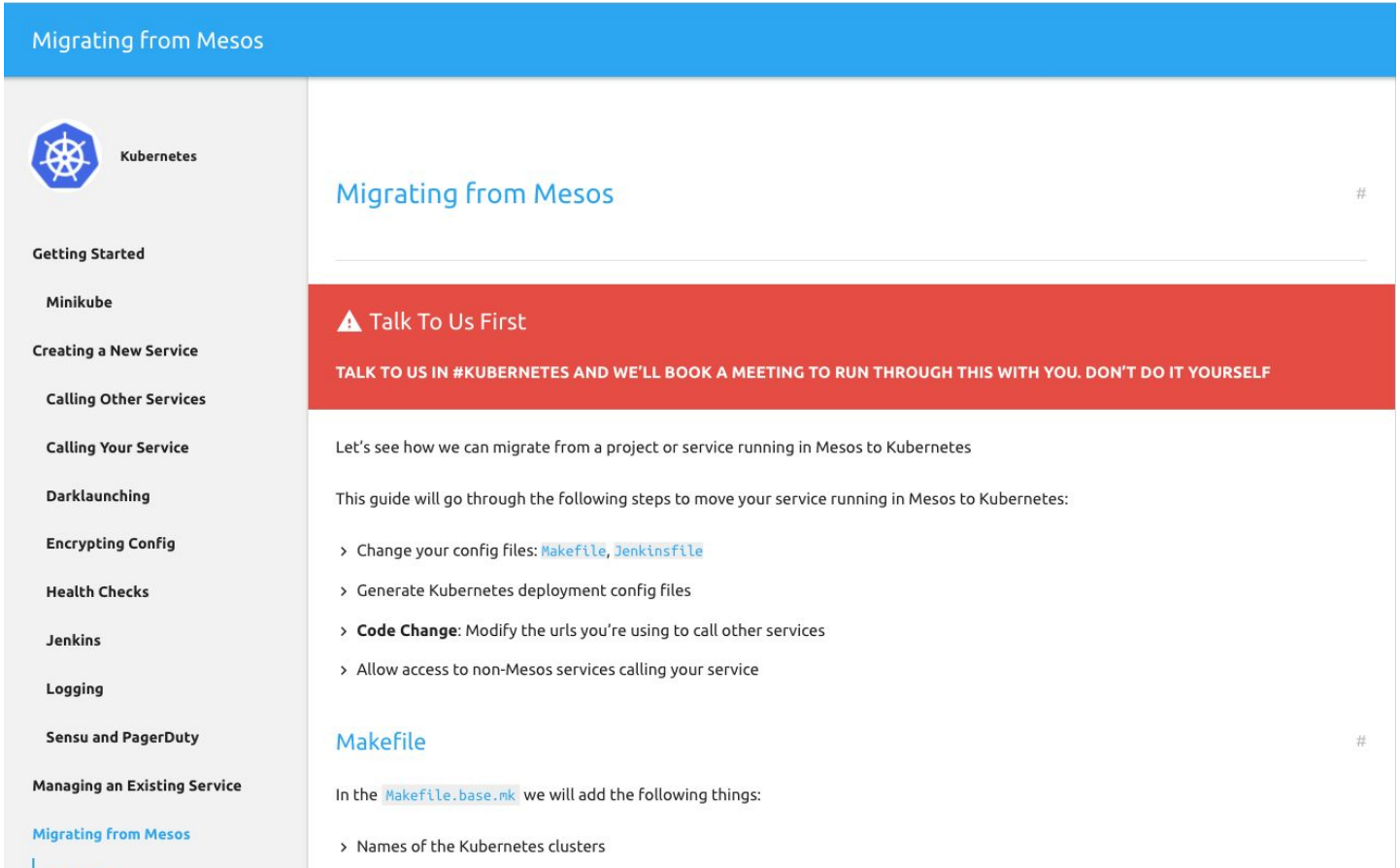
`./project-generator`



Pipeline as Code

Kuberenetes

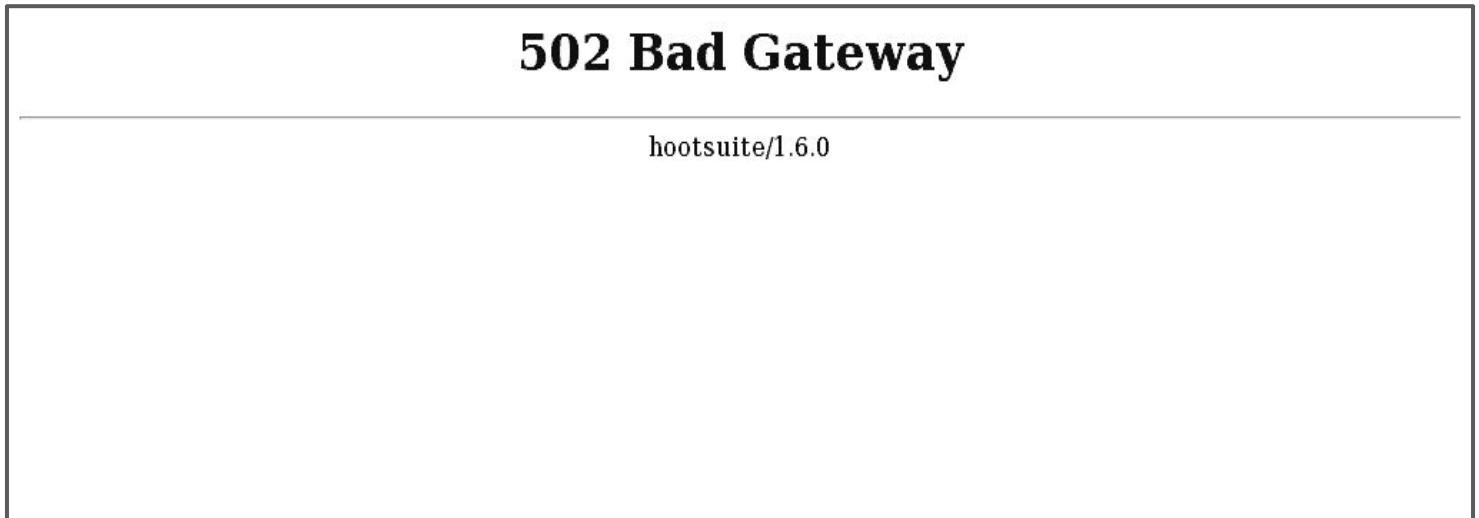
Documentation



Live Demo

Things fail 😔, Let's talk about it....

- "The bad config outage"
- "The classic security group fail"



Lessons Learned/Conclusion

Choose the least important service


Have a rollback plan

Write down what your deployment pipeline looks like


Documentation should be written for humans to read

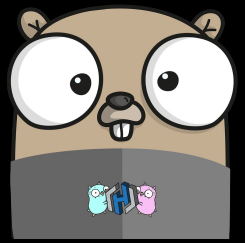
Pragmatic


Minimizing disruption = Great Adoption

Links

- Migrating Container Schedulers: http://code.hootsuite.com/migrating-container-orchestrators-mesos-kubernetes-nomad/
- Abstracting Marathon Deployment Details from Microservices: http://code.hootsuite.com/abstracting-marathon-deployment-details-from-microservices/

Thank you!



Anubhav Mishra

@anubhavm

Hootsuite®