



KubeCon



CloudNativeCon

North America 2017



# Moving from Mesos to Kubernetes *without anyone noticing\**

Anubhav Mishra



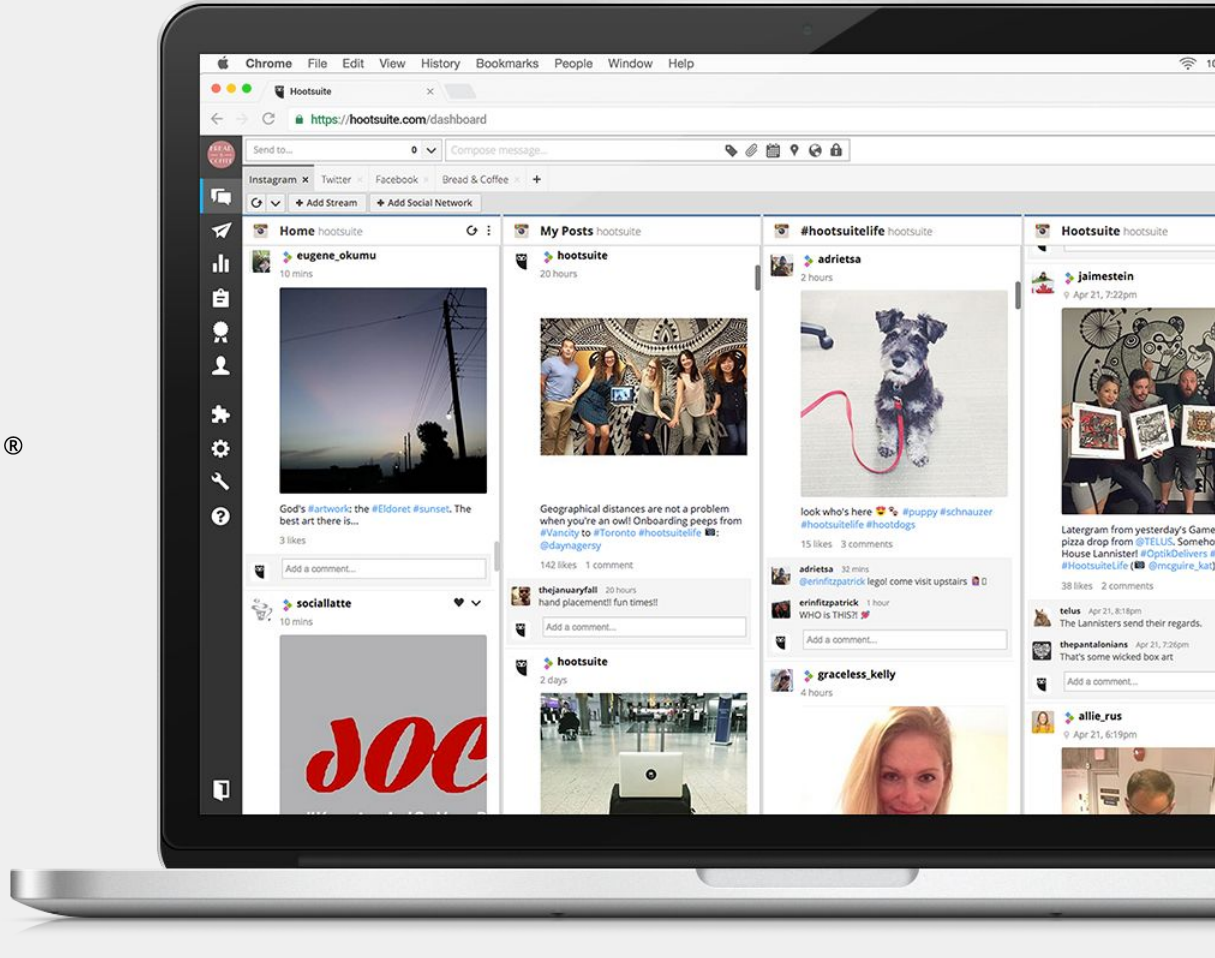
**Anubhav Mishra**

@anubhavm



**Anubhav Mishra**

@anubhavm





**Anubhav Mishra**

@anubhavm



**Hootsuite**®



**Atlantis**



**Anubhav Mishra**

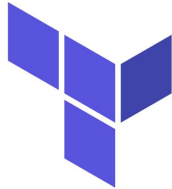
@anubhavm



**Hootsuite**®



**Atlantis**



HashiCorp

**Terraform**



**Anubhav Mishra**

@anubhavm



**Hootsuite**®



**Atlantis**



HashiCorp

**Terraform**



**Anubhav Mishra**

@anubhavm



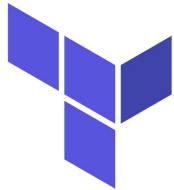




**Hootsuite**®



**Atlantis**



HashiCorp

**Terraform**



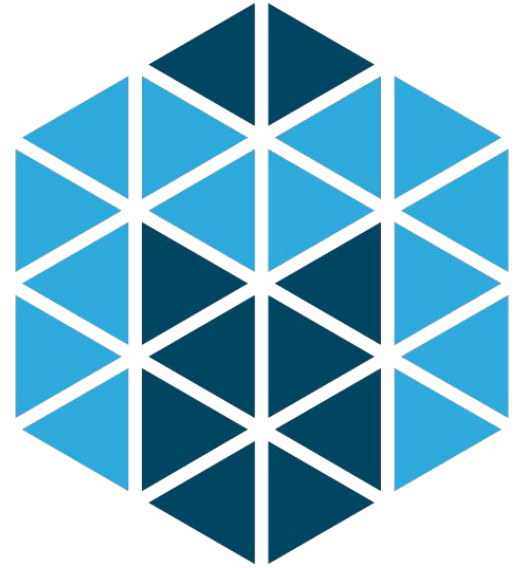
**Anubhav Mishra**

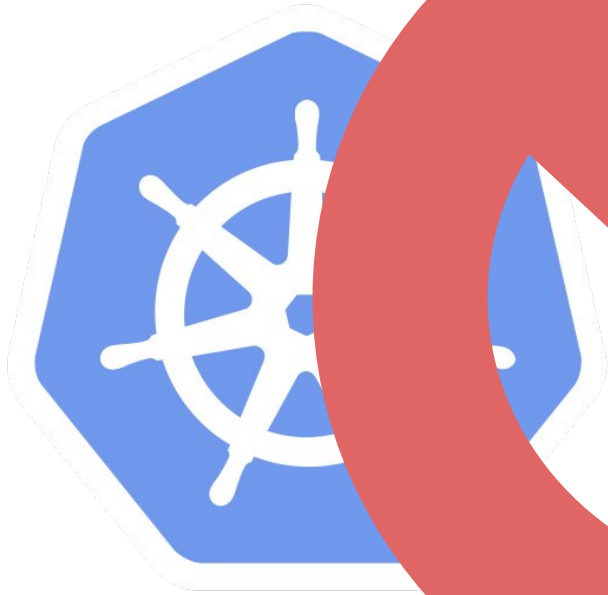
@anubhavm





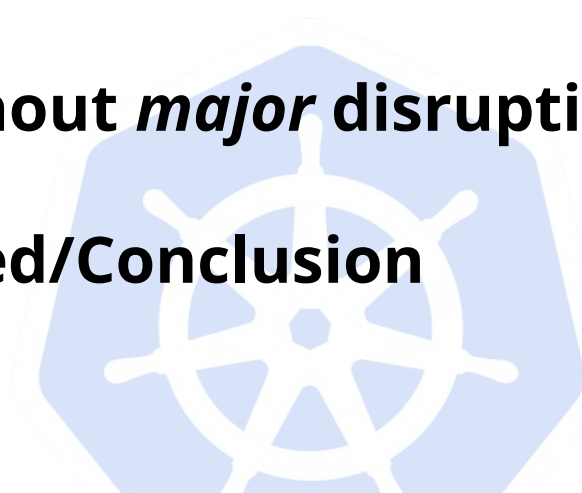
**VS**





# 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**

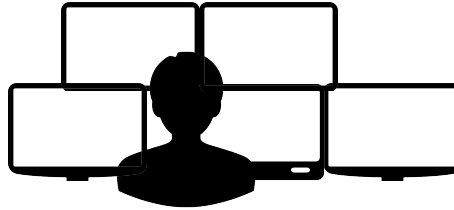




**I want to build  
a microservice**



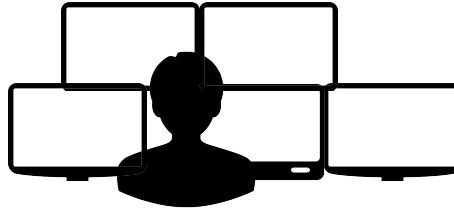
**I want to build  
a microservice**



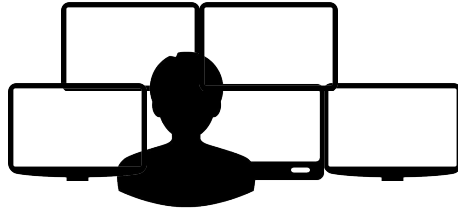
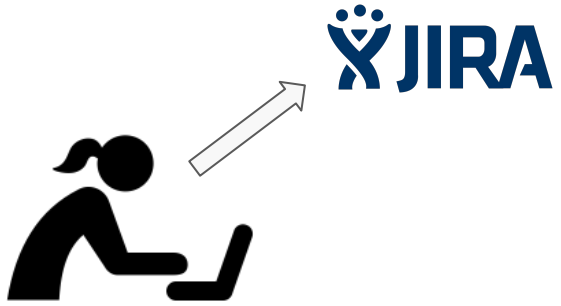
**I want to build  
a microservice**

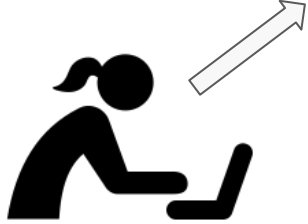


**Oh! A “microservice”?  
Hmm.. seems to be the  
new thing huh. Yep, just  
create a JIRA ticket.**

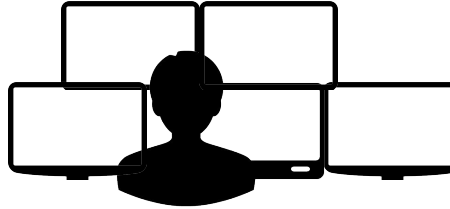


**Minutes later...**









✎ Edit | Comment | Assign | More - | Start Progress | Resolve Issue | Workflow - | Export -

**Details**

Type: 🔴 Bug Status: ➡ Open (View Workflow)  
Priority: 🔴 Major Resolution: Unresolved  
Labels: None

**People**

Assignee: 👤 Admin  
Reporter: 👤 Admin  
Votes: 👍 Vote for this issue  
Watchers: 👤 Stop watching this issue

**Dates**

Created: 25 minutes ago  
Updated: 8 minutes ago

**Drag and Drop**

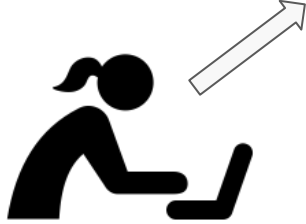
Drop files here to attach them

**> Description**

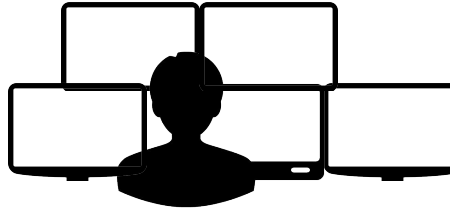
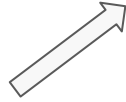
**Activity**

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...



The JIRA logo, consisting of a blue stylized figure above the word "JIRA" in a bold, blue, sans-serif font.



Three screenshots of the Atlassian Confluence interface. The top-left screenshot shows the 'Dashboard' with a 'Welcome to Confluence' message and 'Upcoming Events'. The top-right screenshot shows the 'People Directory' with a list of users. The bottom screenshot shows the 'Space Directory' with a table of spaces and a user profile for 'Matt Hodges'.

**Dashboard**

Welcome to Confluence

Upcoming Events

**Space Directory**

Name	Description	Category
Development	We build all Good stuff	DEV
Marketing	We build all Bad stuff	MARKETING
Product Research	We look at what you really think	PRD
Internal Systems	Small Chat Phones, We build it all	INTERNAL
Development Team Lead		DEV
Feature Leader		DEVELOPMENT
Jerry		PRD
James		MARKETING
Management		DEVELOPMENT
Marketing		MARKETING
Matt		MARKETING
Team		MARKETING

**People Directory**

All People

**Matt Hodges**

Full Name: Matt Hodges  
Email: matt.hodges@atlassian.com  
Phone: (415) 414-1234  
Website: http://www.atlassian.com/people/matt

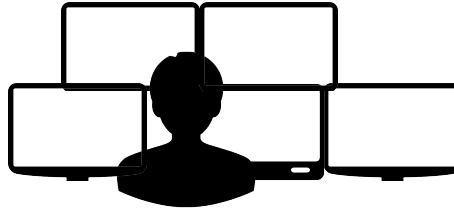
**Weeks later...**



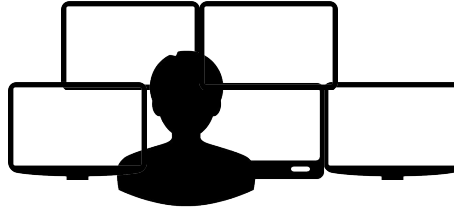
Well, that took a while!



Here are your servers!



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





**2016-2017**



**I want to build  
a microservice**



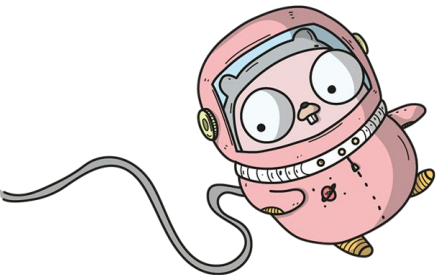
**5 minutes later...**

**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 form name  
#description: Skyline-compatible 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-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

- Up
- Status
- Changes
- Build Now
- View Configuration
- GitHub
- Full Stage View
- Job Config History
- GitHub
- Pipeline Syntax

## Branch master

Full project name: production-delivery (github organization)/mishra-go-skeleton/master



## Stage View

Average stage times:

	setup	unit test	build service	push	deploy dev	deploy staging	deploy production
Average	2s	8s	2s	3s	1s	3s	3s
0-161010b Sep 16 18:30 No Changes	2s	1s	1s	2s	1s	3s	12s (paused for 10s)
7-161010b Sep 16 18:30 No Changes	4s	19s	3s	6s	1s	3s	1s (paused for 10s) aborted
6-161010b Sep 16 18:28 No Changes	4s	23s	3s	6s	1s	3s	1s (paused for 6s) aborted
5-161010b Sep 16 15:34 1 commits	4s	1s	1s	2s	1s	3s	1s (paused for 40s) aborted

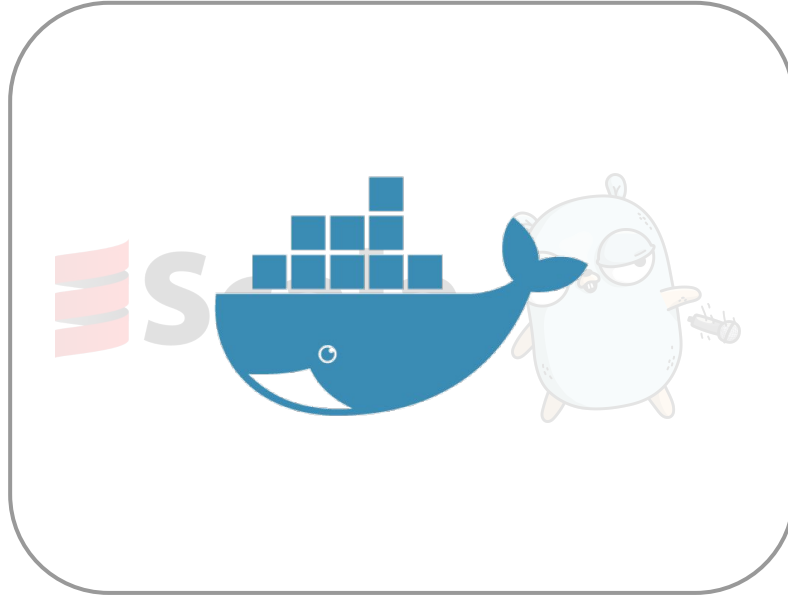
### Build History trend

- 8-161010b  
Sep 16, 2016 6:30 PM
- 7-161010b  
Sep 16, 2016 6:30 PM
- 6-161010b  
Sep 16, 2016 6:28 PM
- 5-161010b  
Sep 16, 2016 3:34 PM
- 4-56c3eb7  
Sep 16, 2016 2:54 PM
- 3-5008c81

 **Scala**

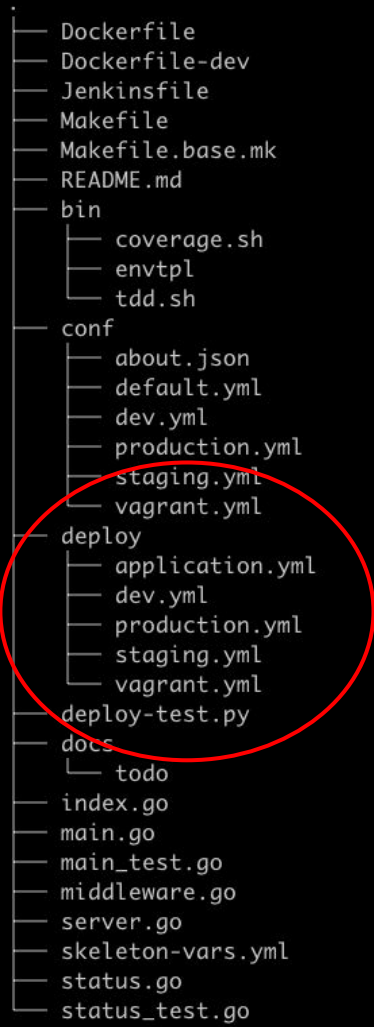


# Packaging



# Deployment Files

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





# Makefile

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

```
├── 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
```



**Mesos**

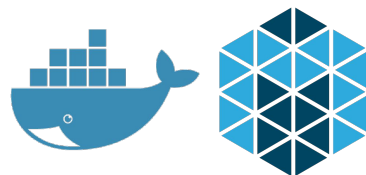
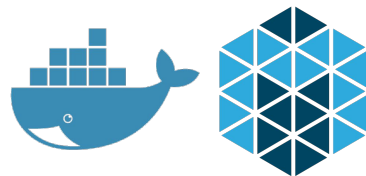
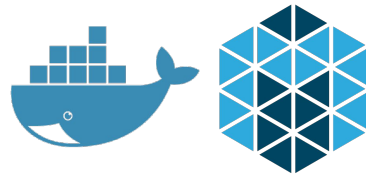
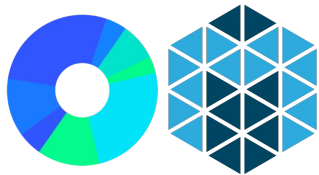
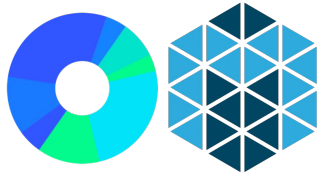
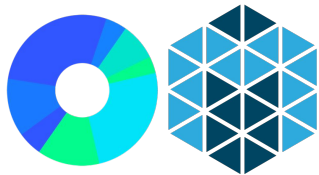


**Marathon**



make deploy

API





make deploy

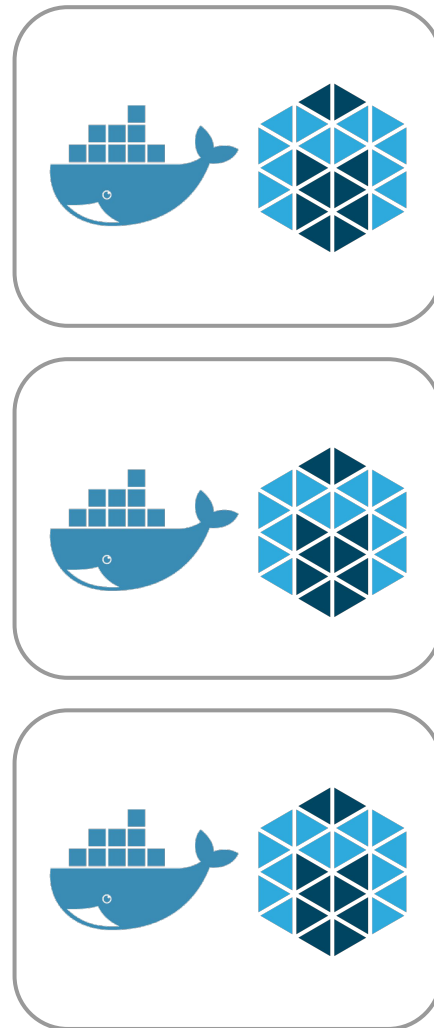
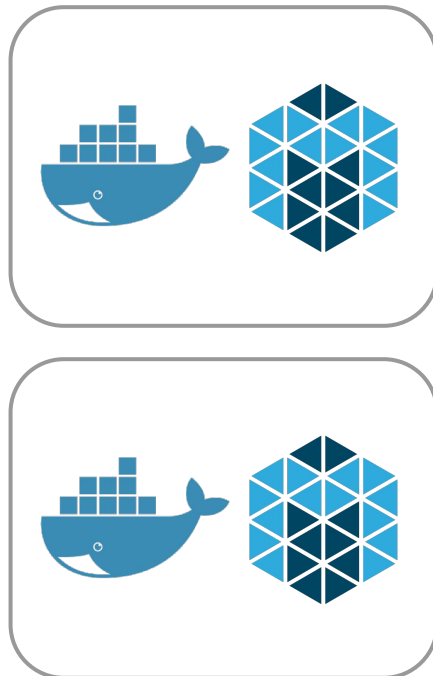
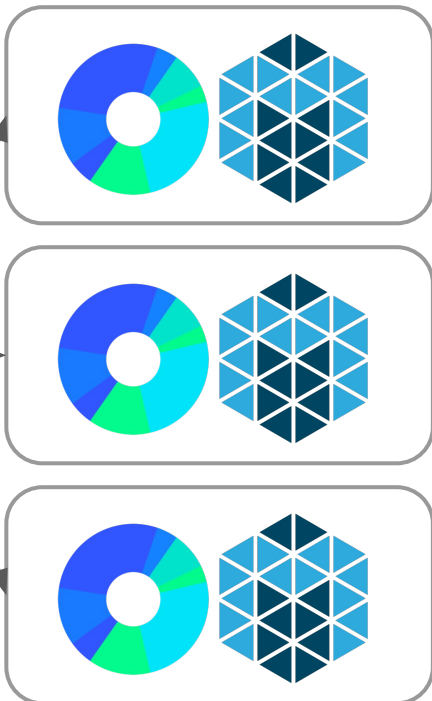
```
POST  
{
```

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

```
}
```



API





make deploy

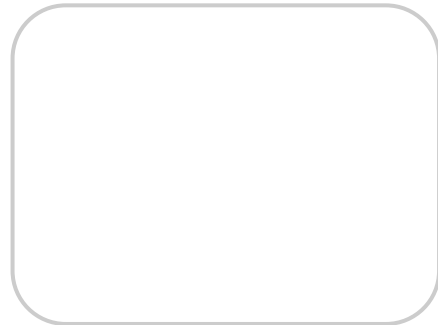
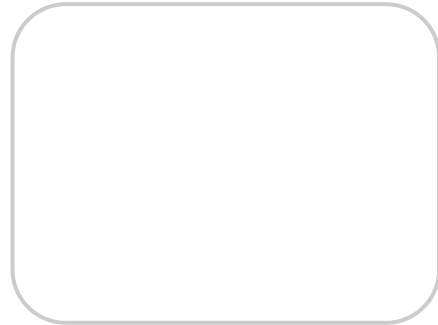
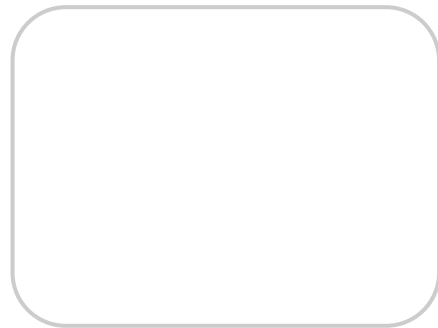
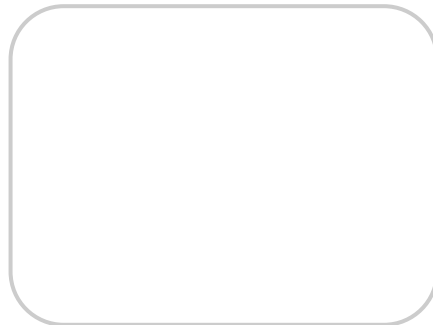
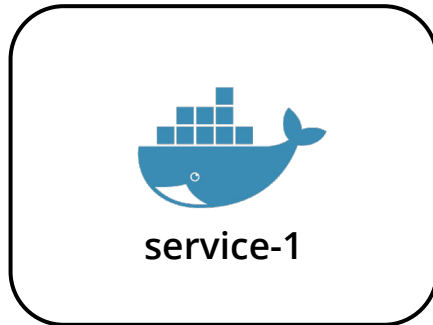
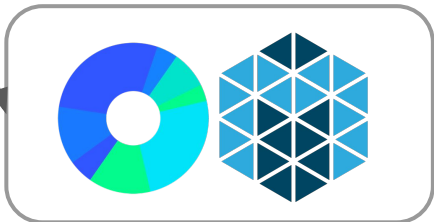
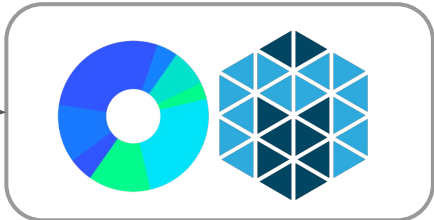
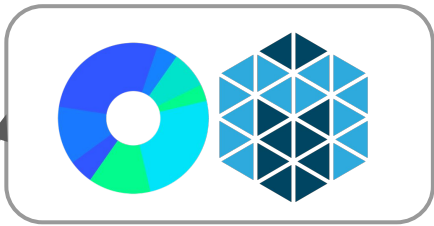
```
POST  
{
```

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

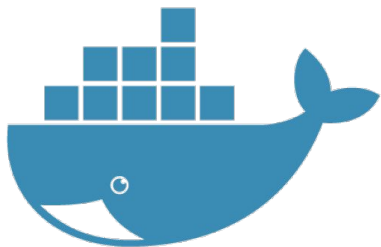
```
}
```



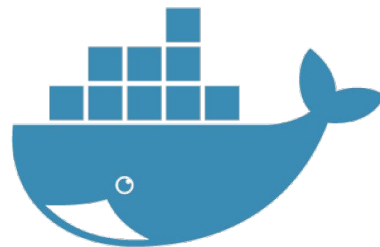
API



# Routing

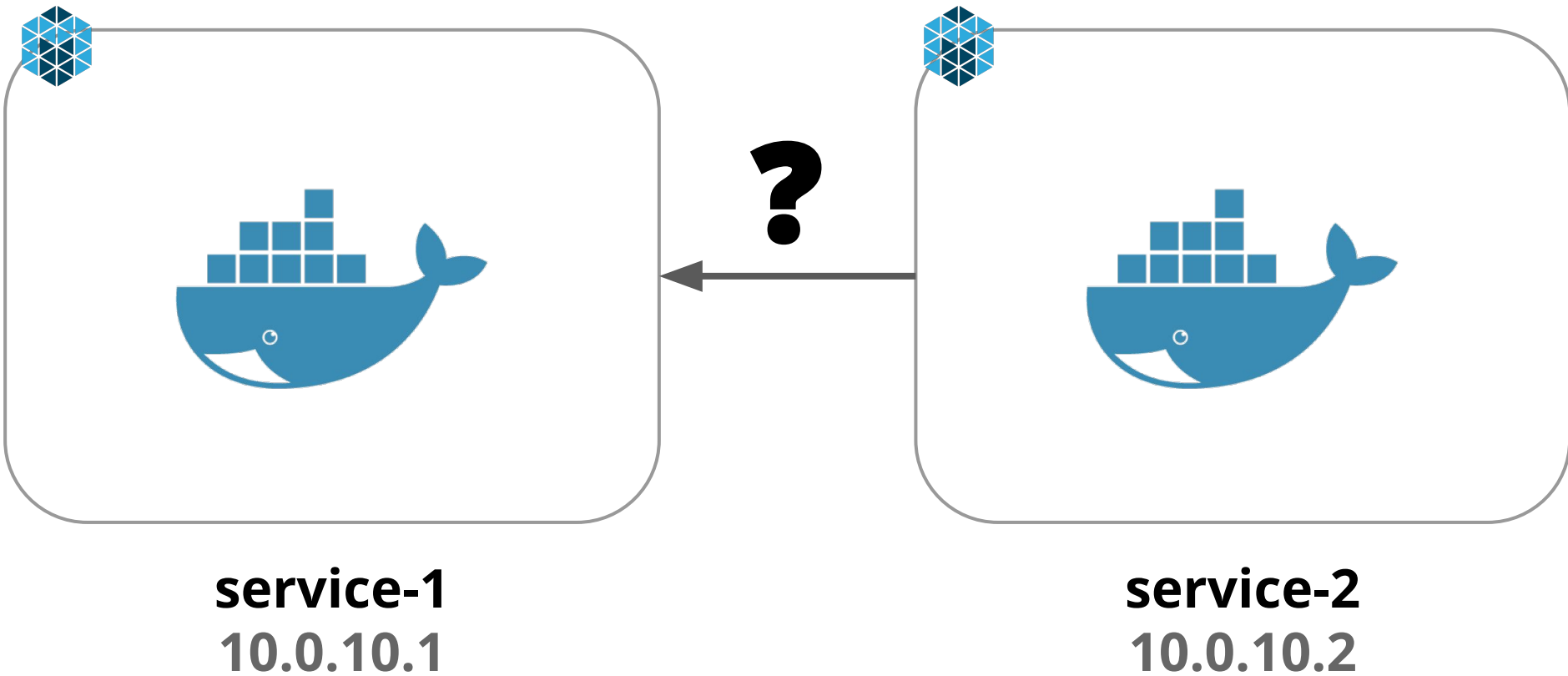


**service-1**  
10.0.10.1

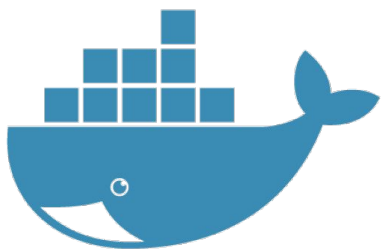


**service-2**  
10.0.10.2

# Routing

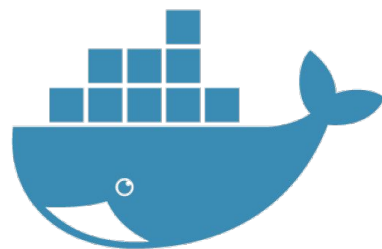


# Routing - Fat Middleware



**NGINX** 

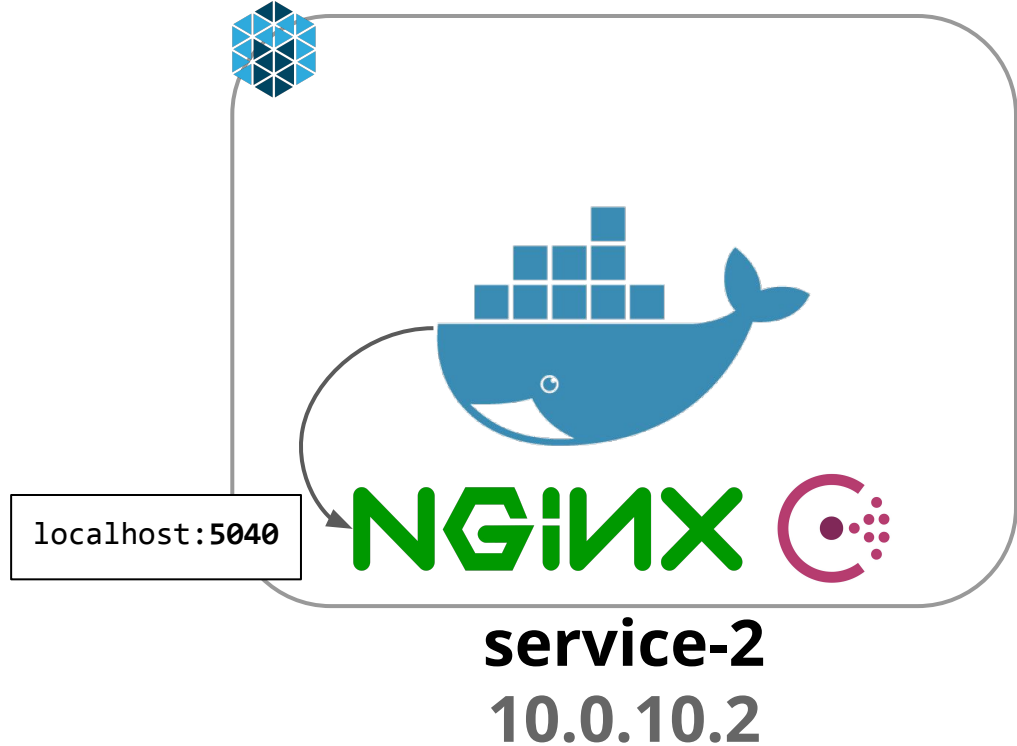
**service-1**  
10.0.10.1



**NGINX** 

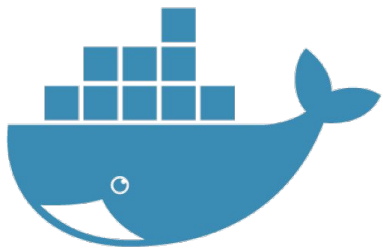
**service-2**  
10.0.10.2





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

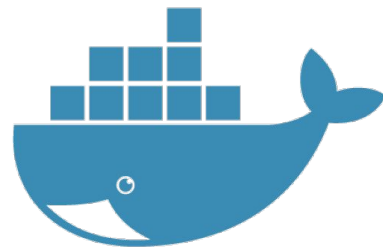
```
{  
  upstream service-1 {  
    server 10.0.10.1:5041;  
    ....  
  }  
}
```



**NGINX** 

**service-1**  
10.0.10.1

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



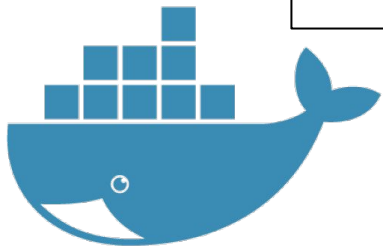
**NGINX** 

**service-2**  
10.0.10.2

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

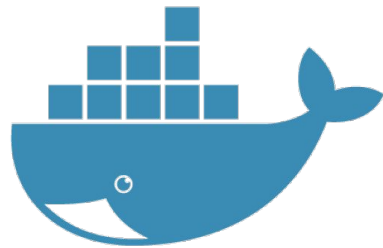


localhost:8080



NGINX 

**service-1**  
10.0.10.1

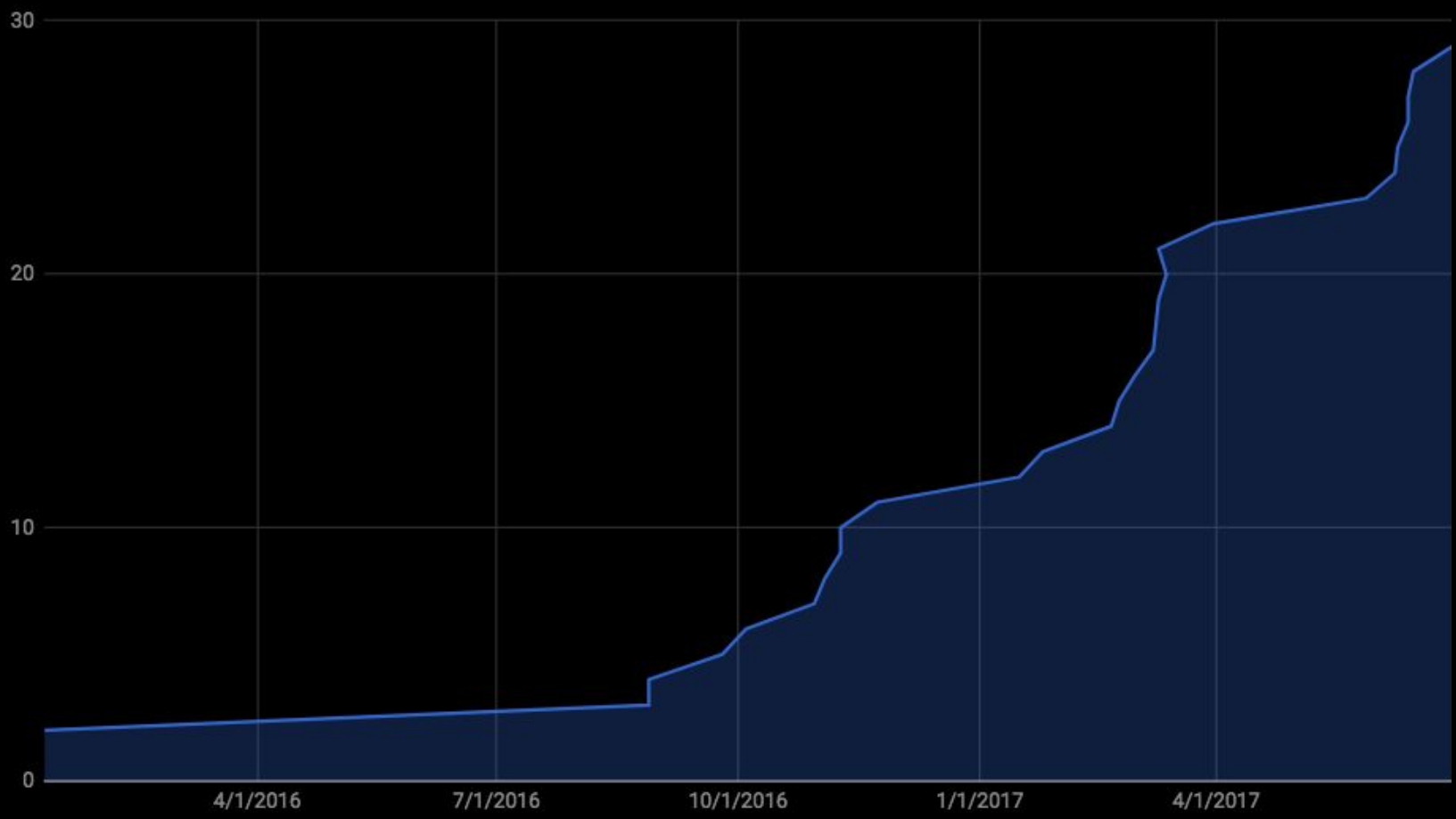


NGINX 

**service-2**  
10.0.10.2

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









# Why Kubernetes?

## 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.



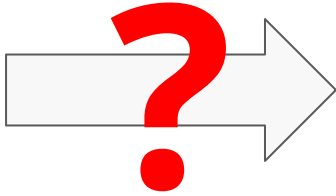
**4 months**

**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

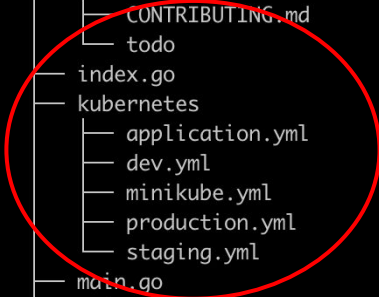
```
#####  
# Modify these vars  
  
# Name of your service, lowercase, dash separated. ie. [a-z\-]  
# !!! SHOULD NOT have "service" as prefix or suffix. We're trying to get rid of that.  
# Will be used for Skyline and Kubernetes names as well as statsd prefixes, etc.  
#name: my-awesome-service  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

```
— 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
```





# Pipeline as Code

- Up
- Status
- Changes
- Build Now
- View Configuration
- GitHub
- Full Stage View
- Job Config History
- GitHub
- Pipeline Syntax

## Branch master

Full project name: production-delivery (github organization)/mishra-go-skeleton/master



## Stage View

Average stage times:

	setup	unit test	build service	push	deploy dev	deploy staging	deploy production
Average	2s	8s	2s	3s	1s	3s	3s
0-161010b Sep 16 18:30 No Changes	2s	1s	1s	2s	1s	3s	12s <small>(paused for 10s)</small>
7-161010b Sep 16 18:30 No Changes	4s	19s	3s	6s	1s	3s	1s <small>(paused for 10s)</small> aborted
6-161010b Sep 16 18:28 No Changes	4s	23s	3s	6s	1s	3s	1s <small>(paused for 6s)</small> aborted
5-161010b Sep 16 15:34 1 commits	4s	1s	1s	2s	1s	3s	1s <small>(paused for 40s)</small> aborted

### Build History trend

- 8-161010b  
Sep 16, 2016 6:30 PM
- 7-161010b  
Sep 16, 2016 6:30 PM
- 6-161010b  
Sep 16, 2016 6:28 PM
- 5-161010b  
Sep 16, 2016 3:34 PM
- 4-56c3eb7  
Sep 16, 2016 2:54 PM
- 3-5008c81

# Pipeline as Code

The screenshot displays the Jenkins web interface for a pipeline. At the top, the Jenkins logo and 'Open Blue Ocean' button are visible. The breadcrumb navigation shows the path: Jenkins > Production Delivery (github organization) > mishra-go-skeleton > master > .

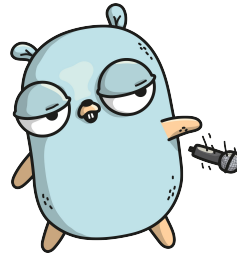
The main content area is titled 'Branch master' and shows a 'Stage View' of a pipeline. A large blue hexagonal logo with a white ship's wheel is overlaid on the pipeline graph. The pipeline consists of several stages, with the final stage being 'deploy production'. The stages and their durations are as follows:

Build ID	Time	Changes	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
8-161010b	Sep 16 18:30	No Changes	4s	1s	1s	2s	1s	3s	12s (paused for 10s)
7-161010b	Sep 16 18:30	No Changes	4s	1s	1s	2s	1s	3s	1s (paused for 10s) aborted
6-161010b	Sep 16 18:28	No Changes	4s	1s	1s	2s	1s	3s	1s (paused for 10s) aborted
5-161010b	Sep 16 15:34	1 commits	4s	1s	1s	2s	1s	3s	1s (paused for 10s) aborted

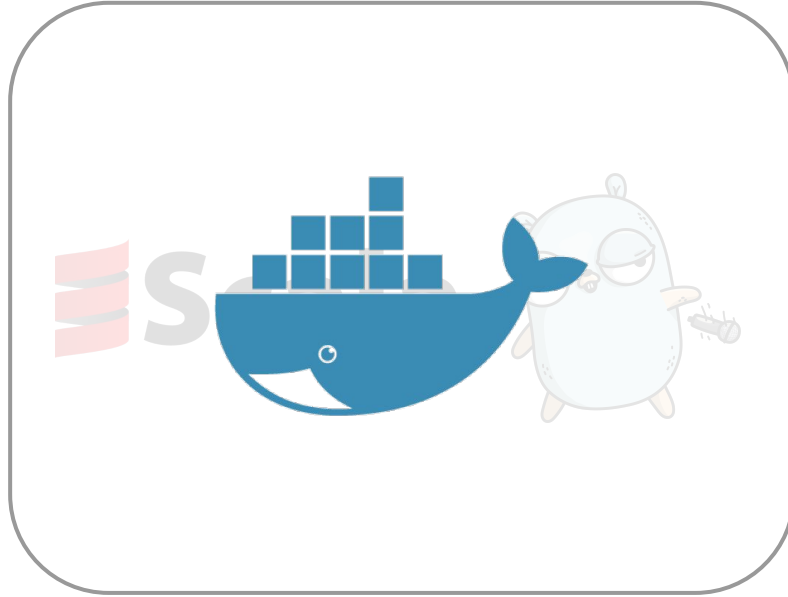
On the left side, there is a 'Build History' section with a search bar and a list of recent builds:

- 8-161010b (Sep 16, 2016 6:30 PM)
- 7-161010b (Sep 16, 2016 6:30 PM)
- 6-161010b (Sep 16, 2016 6:28 PM)
- 5-161010b (Sep 16, 2016 3:34 PM)
- 4-56c3eb7 (Sep 16, 2016 2:54 PM)
- 3-5008c81

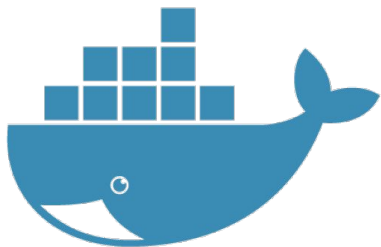
# Packaging

The Scala logo consists of a red icon on the left, which is a stylized 'S' composed of three horizontal bars of varying lengths, followed by the word "Scala" in a bold, black, sans-serif font.

# Packaging

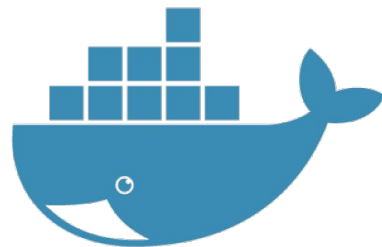


# Routing in Mesos



**NGINX** 

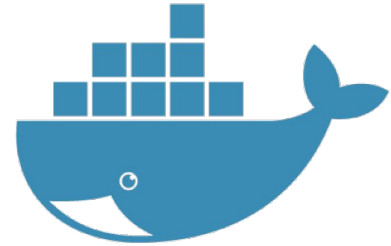
**service-1**  
10.0.10.1



**NGINX** 

**service-2**  
10.0.10.2

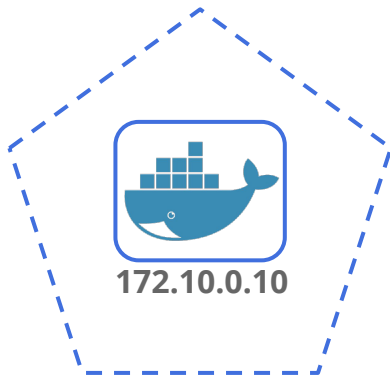
# Routing in Mesos



**NGINX** 

**service-2**  
10.0.10.2

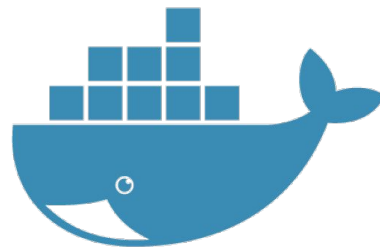
# Routing to K8s



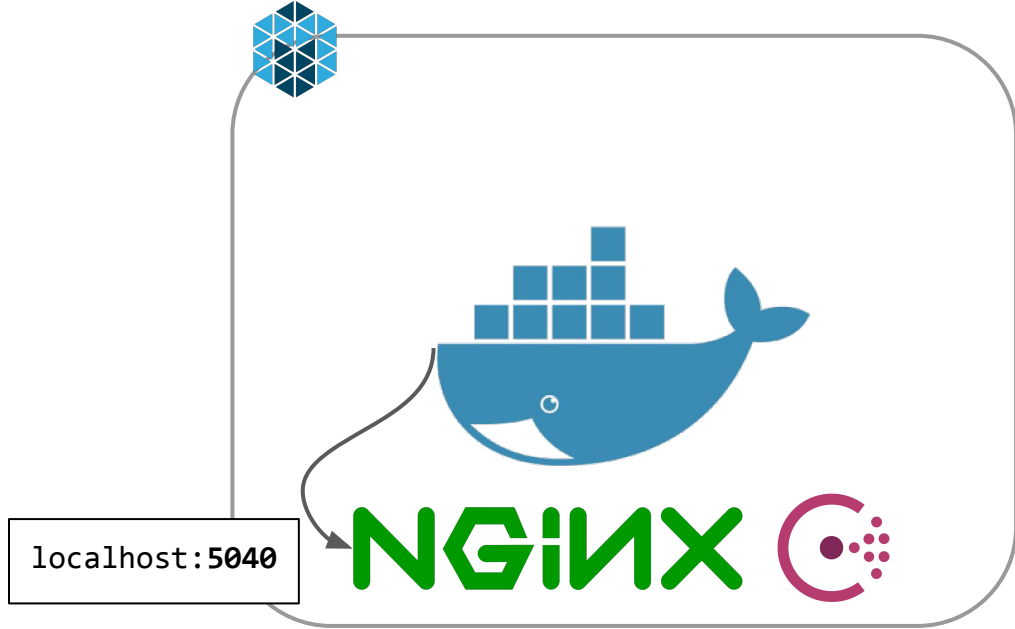
**service-1**  
10.0.17.1



10.0.30.10



**service-2**  
10.0.10.2



localhost:5040

NGINX 

## service-2

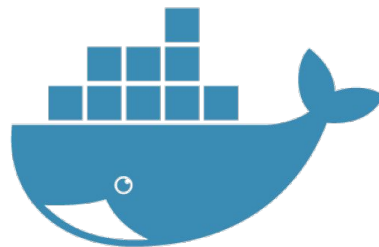
```
{
  upstream service-1 {
  }
  upstream bridge-1 {
    server 10.0.20.1:5041;
    ....
  }
}
```

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



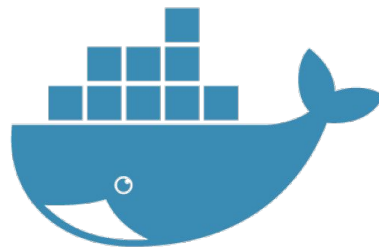


**bridge-1**  
(multi-dc aware)



**service-2**

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



NGINX 

NGINX 

**bridge-1**  
(multi-dc aware)

**service-2**

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

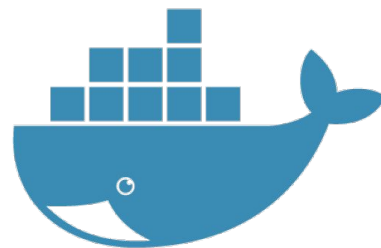
```
{  
  upstream service-1 {  
  }  
  upstream bridge-1 {  
    server 10.0.20.1:5041;  
    ....  
  }  
}
```



NGINX 

**bridge-1**  
(multi-dc aware)

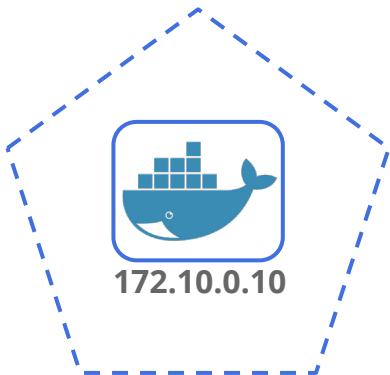
```
curl https://10.0.30.1:5041/service/service-1/endpoint
```



NGINX 

**service-2**

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



172.10.0.10

**service-1**

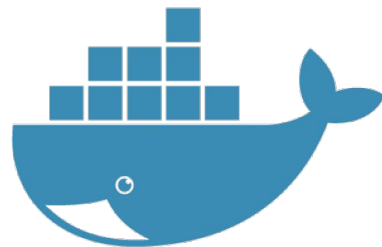
10.0.17.1

**NGINX**



10.0.30.10

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



**NGINX**



**service-2**

10.0.10.2



```
http://service-1.default.svc.cluster.local:8080
```



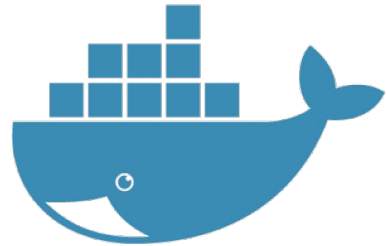
172.10.0.10

**service-1**  
10.0.17.1

**NGINX**

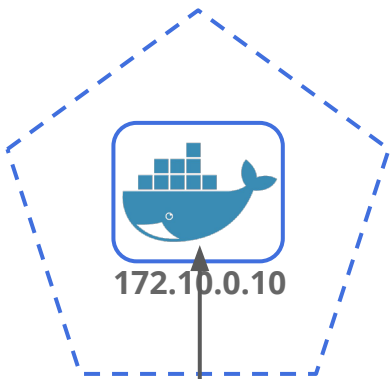
10.0.30.10

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



**NGINX**

**service-2**  
10.0.10.2



172.10.0.10

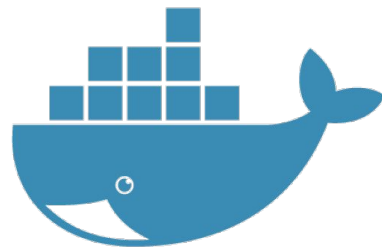
**service-1**

10.0.17.1

**NGINX**



10.0.30.10



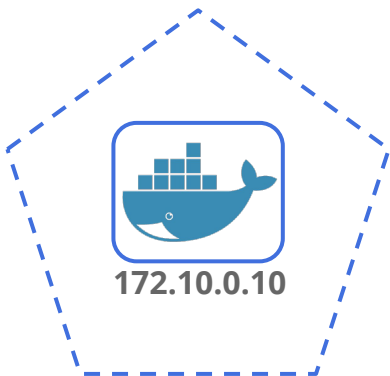
**NGINX**



**service-2**

10.0.10.2

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



172.10.0.10

**service-1**

10.0.17.1

**NGINX**

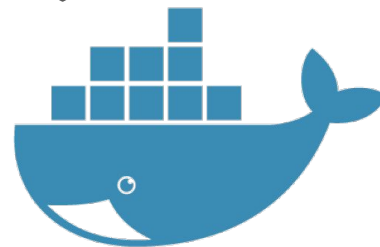


10.0.30.10

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



Love getting those  
**OK** responses! 😊



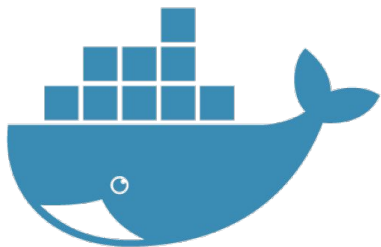
**NGINX**



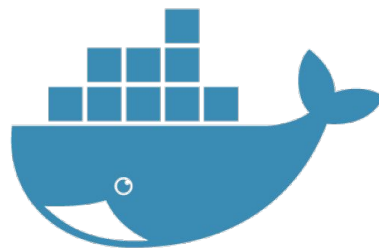
**service-2**

10.0.10.2

# 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

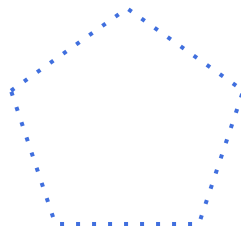


**foo**  
10.0.10.4



**service-1**

172.10.0.10



**foo**

10.0.17.100



10.0.30.10



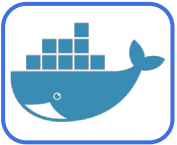
**foo**

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
```

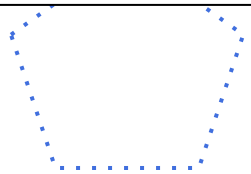


```
curl http://foo:5040/endpoint
```



**service-1**

172.10.0.10



**foo**

10.0.17.100

**NGINX** 

10.0.30.10



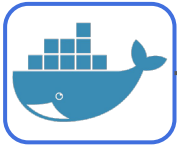
**NGINX** 

**foo**

10.0.10.4



```
curl http://foo:5040/endpoint
```



**service-1**

172.10.0.10



**foo**

10.0.17.100

**NGINX**

10.0.30.10



**NGINX**

**foo**

10.0.10.4

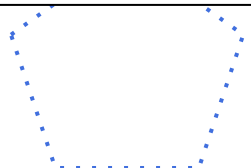


```
curl http://foo:5040/endpoint
```



**service-1**

172.10.0.10



**foo**

10.0.17.100



10.0.30.10



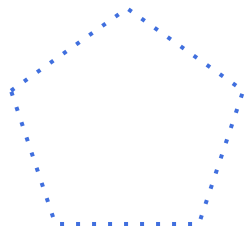
**foo**

10.0.10.4



**service-1**

172.10.0.10



**foo**

10.0.17.100



10.0.30.10



**foo**

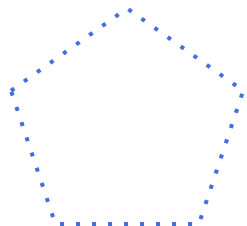
10.0.10.4

```
# 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



10.0.30.10



**NGINX**

**foo**

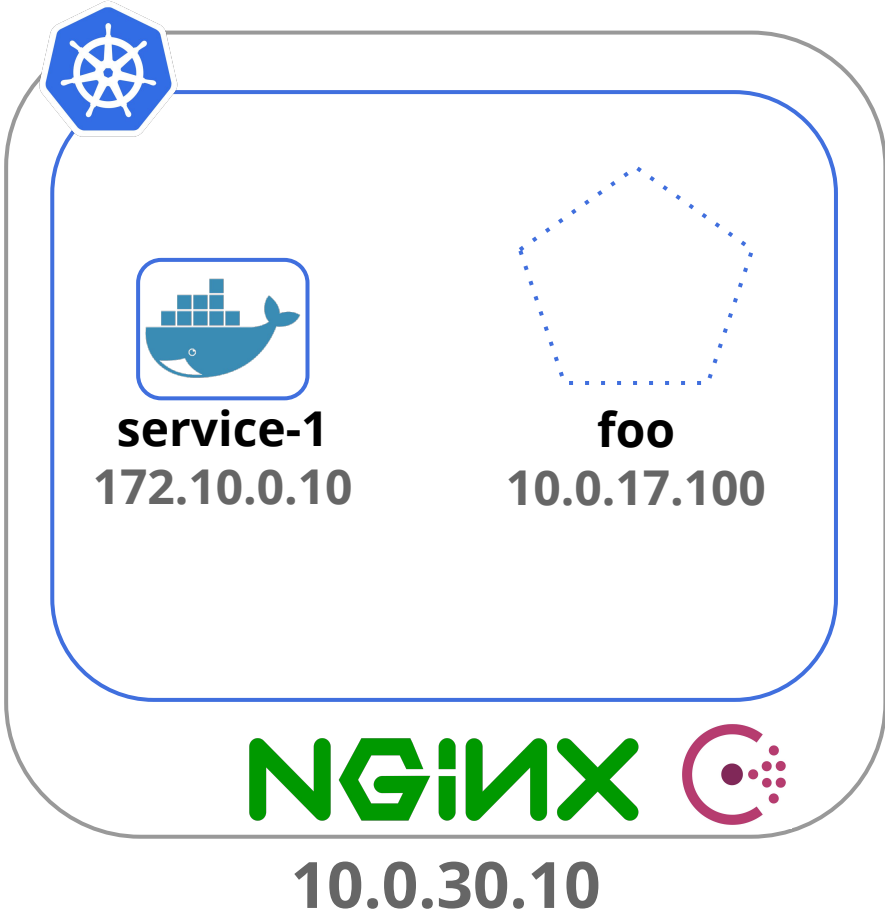
10.0.10.4



**bridge-1**

(multi-dc aware)

```
curl http://bridge1:5041/service/foo/endpoint
```

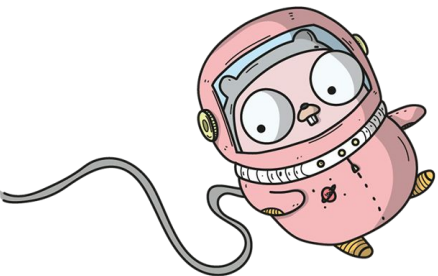


```
curl http://10.0.10.4:5041/service/foo/endpoint
```





# Ship it!



`./project-generator`



**This branch has no conflicts with the base branch**

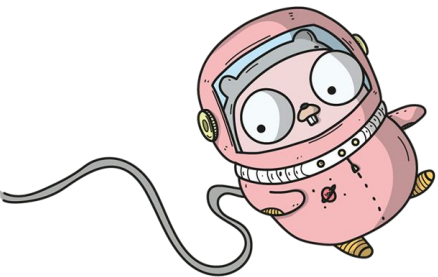
Merging can be performed automatically.

**Merge pull request**



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# Microservice Pipeline



`./project-generator`




Pipeline as Code



Kubernetes

# Documentation

## Migrating from Mesos



Kubernetes

- Getting Started
  - Minikube
- Creating a New Service
  - Calling Other Services
  - Calling Your Service
- Darklaunching
- Encrypting Config
- Health Checks
- Jenkins
- Logging
- Sensu and PagerDuty
- Managing an Existing Service
- Migrating from Mesos**

## Migrating from Mesos

---

**⚠ Talk To Us First**

TALK TO US IN #KUBERNETES AND WE'LL BOOK A MEETING TO RUN THROUGH THIS WITH YOU. DON'T DO IT YOURSELF

Let's see how we can migrate from a project or service running in Mesos to Kubernetes

This guide will go through the following steps to move your service running in Mesos to Kubernetes:

- > Change your config files: [Makefile](#), [Jenkinsfile](#)
- > Generate Kubernetes deployment config files
- > **Code Change:** Modify the urls you're using to call other services
- > Allow access to non-Mesos services calling your service

## Makefile

In the [Makefile.base.mk](#) we will add the following things:

- > Names of the Kubernetes clusters

# Live Demo





# Things fail 🙄, Let's talk about it....

- “The bad config outage”
- “The classic security group fail”

**502 Bad Gateway**

---

hootsuite/1.6.0



**Lessons Learned/Conclusion**



**Choose the least important  
service**



A man with a beard and sunglasses, wearing a black long-sleeved shirt and black pants, is performing a wheelie on a green bicycle. He is wearing large white headphones. The background shows a construction site with a chain-link fence, wooden scaffolding, and several white bags of Tyvek material. A red and white striped traffic barrel is visible behind the fence. The scene is brightly lit, suggesting a sunny day.

**Have a rollback plan**



A man with short, dark, curly hair is sitting at a wooden desk in an office. He is wearing a dark blue t-shirt and is focused on writing in a spiral-bound notebook with a blue pen. To his left is a large computer monitor and a keyboard. On the desk, there are also some papers, a pair of glasses, and a blue object. The background shows a modern office environment with a hallway and some blurred lights. A dark grey horizontal bar is overlaid across the middle of the image, containing white text.

**Write down what your  
deployment pipeline looks like**



**Documentation should be written for humans to read**

A close-up photograph of the back of a white electronic device, showing a black panel with three RCA connectors. The left connector is labeled 'VIDEO' and has a blue cable plugged into it. The middle connector is labeled 'AUDIO' and has a black cable plugged into it. The right connector is labeled 'AUDIO' and has a red cable plugged into it. The device has a white perforated grille above the connectors. A semi-transparent dark grey horizontal bar is overlaid across the middle of the image, containing the word 'Pragmatic' in white text.

**Pragmatic**



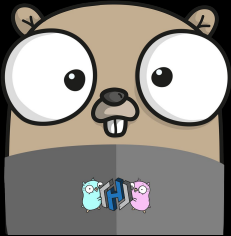
**Minimizing disruption = Great Adoption**



# 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

