ROME 18-19 MARCH 2016

{codemotion}

Paas with Docker and K8s

Massimiliano Dessì @desmax74

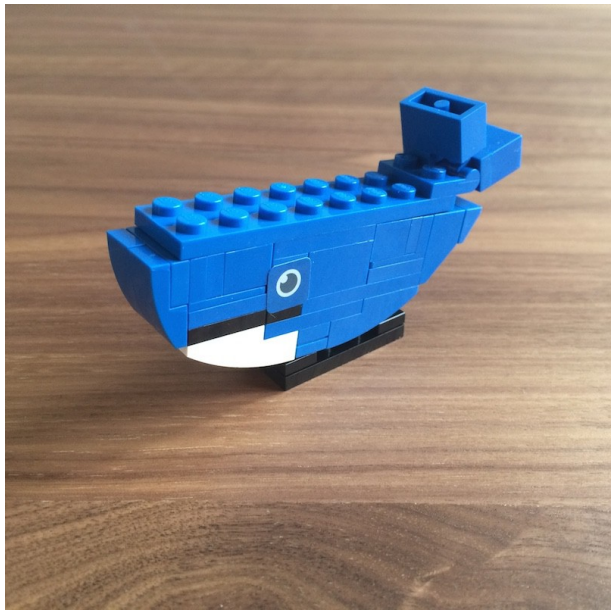
Speaker

Massimiliano Dessì has more than 15 years of experience in programming.
Manager of GDG Sardegna, Founder of SpringFramework IT,
co-founder of Jug Sardegna.
Author of Spring 2.5 AOP.
He works in the CloudComputing and DevOps area.

GDG Sardegna

A lot of hype around Docker

```
FROM ubuntu:latest
WORKDIR /app
ADD src/github.com/<mycompany>/service1 /app/service1
EXPOSE 8080
ENTRYPOINT ["/app/service1"]
```



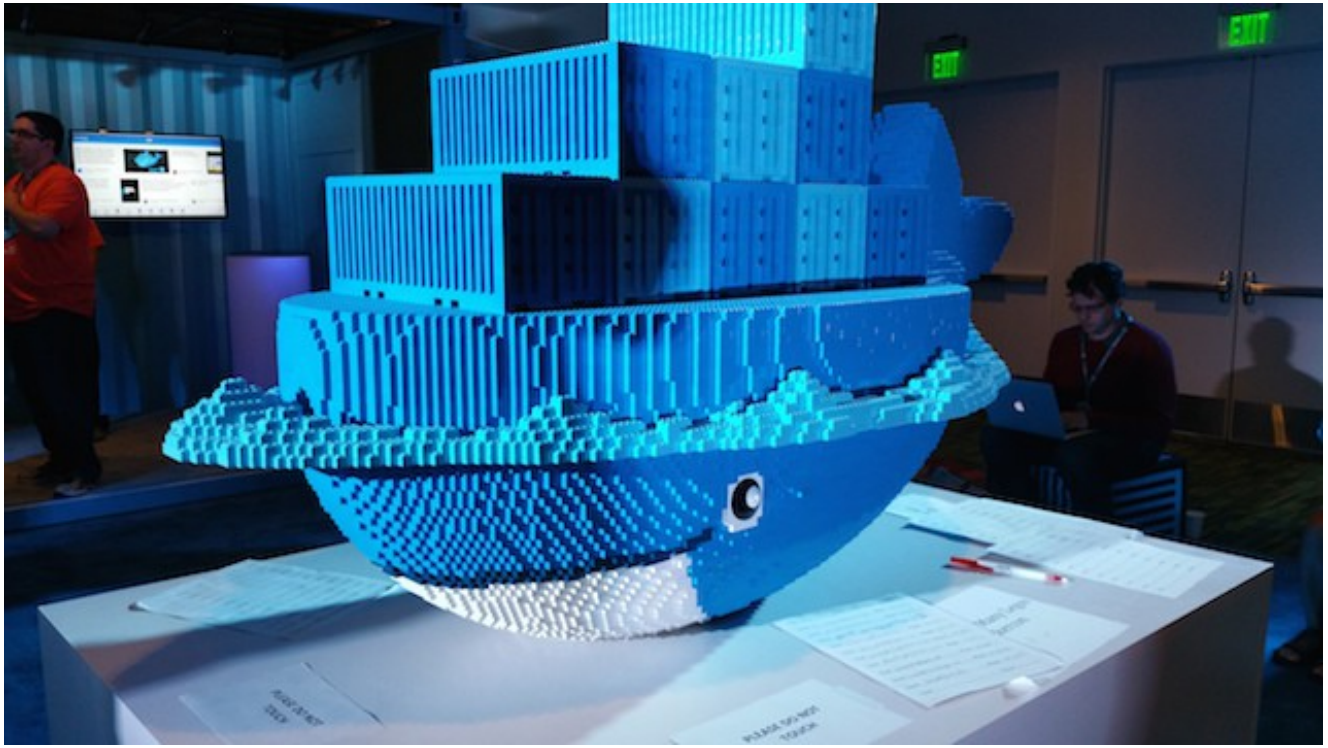
Layered and immutable images

Copy on write model

each container has its own file system and ports

...

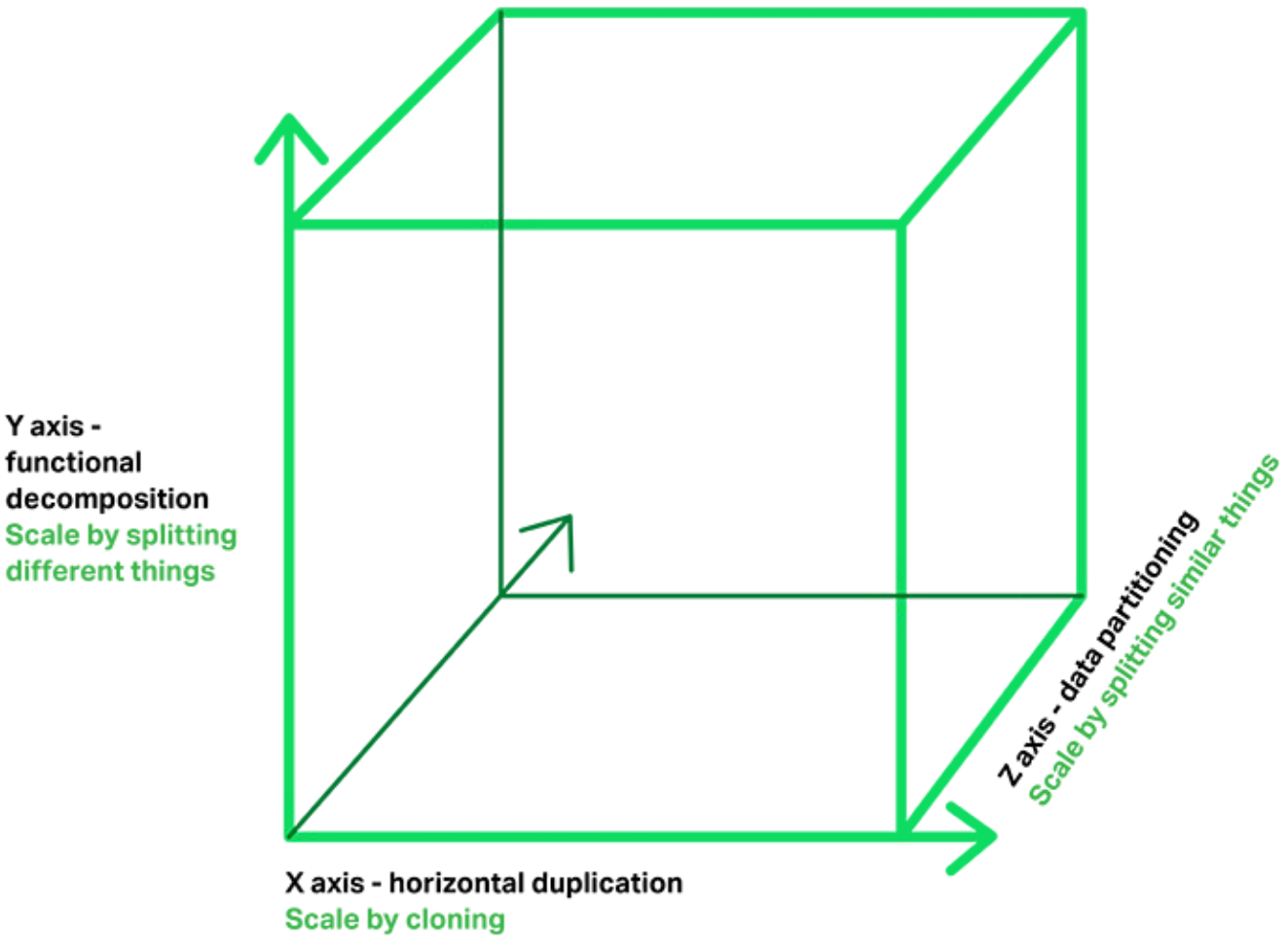
But the Rock & roll is when the system grows



the rationale behind docker

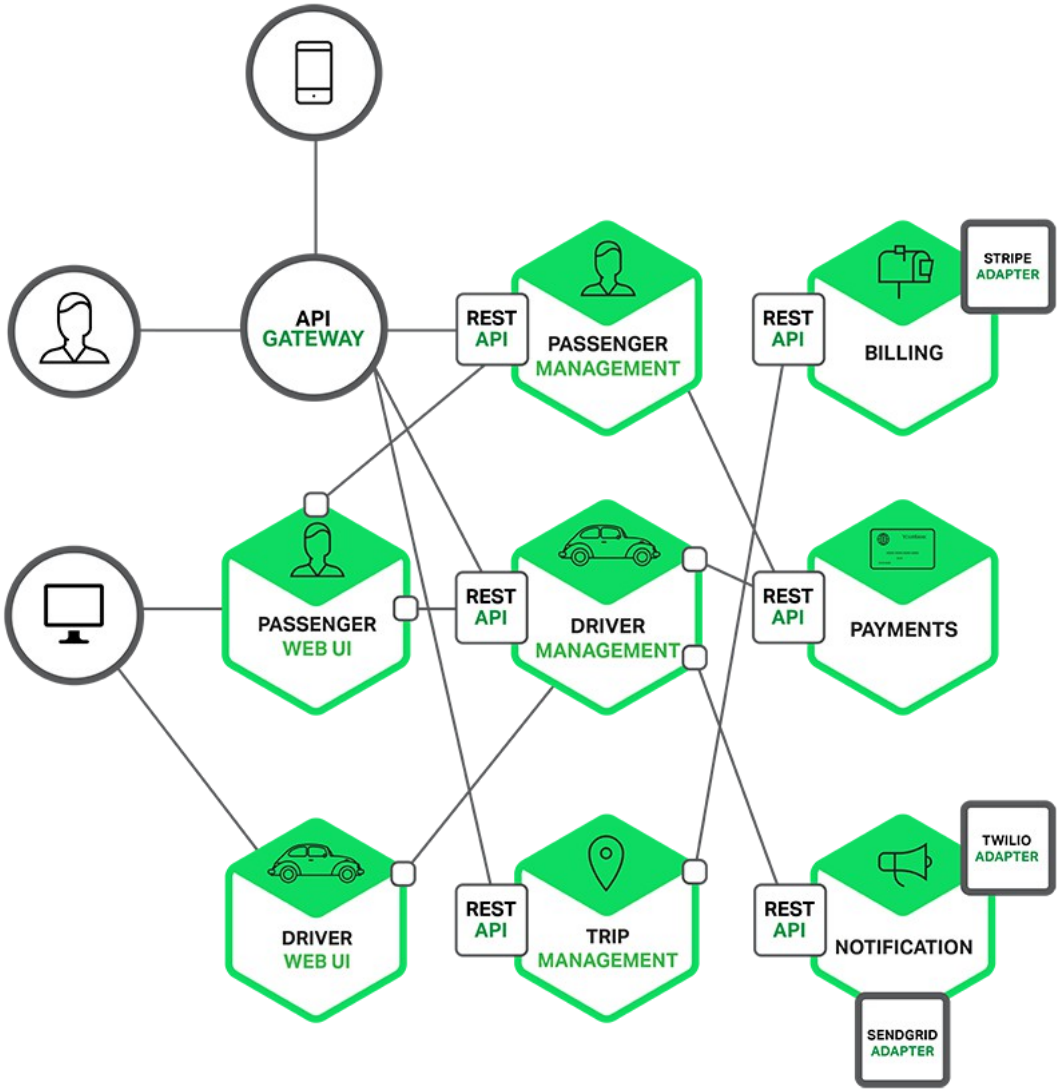
http://martinfowler.com/articles/microservices.html

Modern requirement: Scale



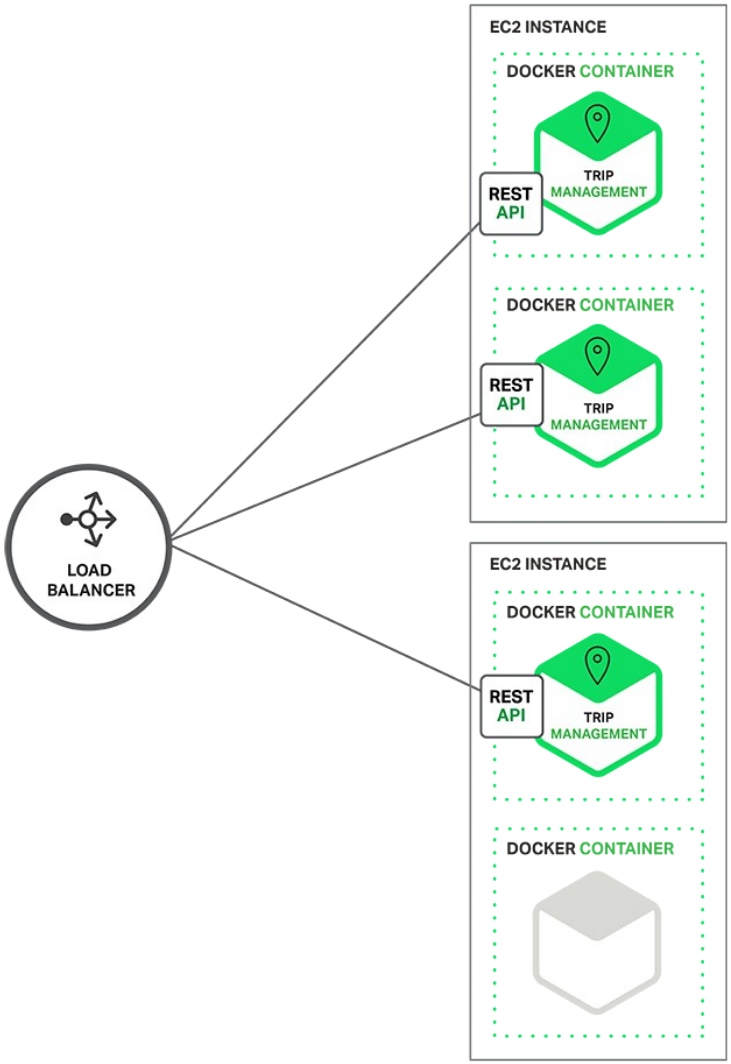
https://www.nginx.com/blog/introduction-to-microservices/

Y Axis (Functional Decomposition)



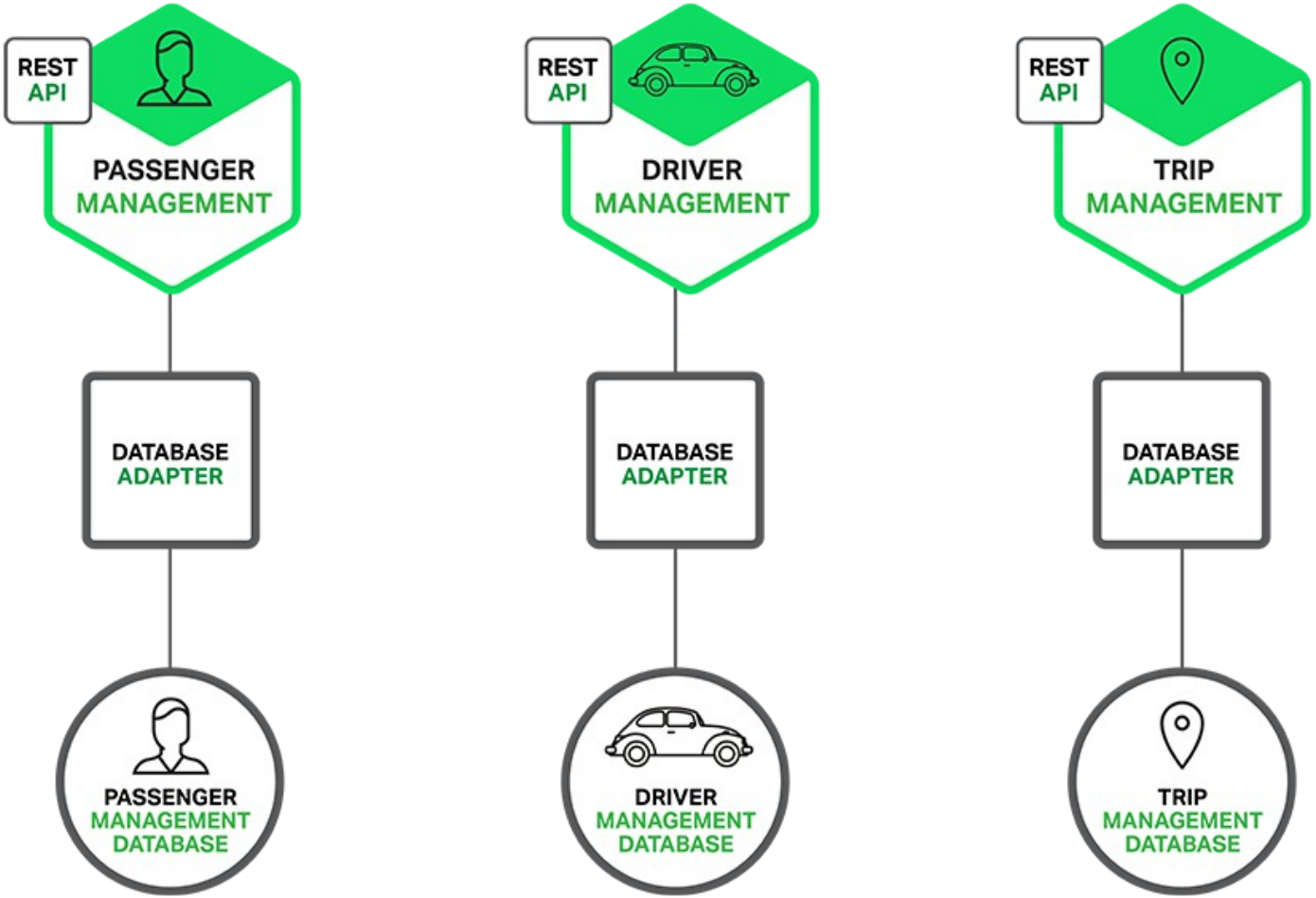
https://www.nginx.com/blog/introduction-to-microservices/

X Axis (Horizontal duplication)



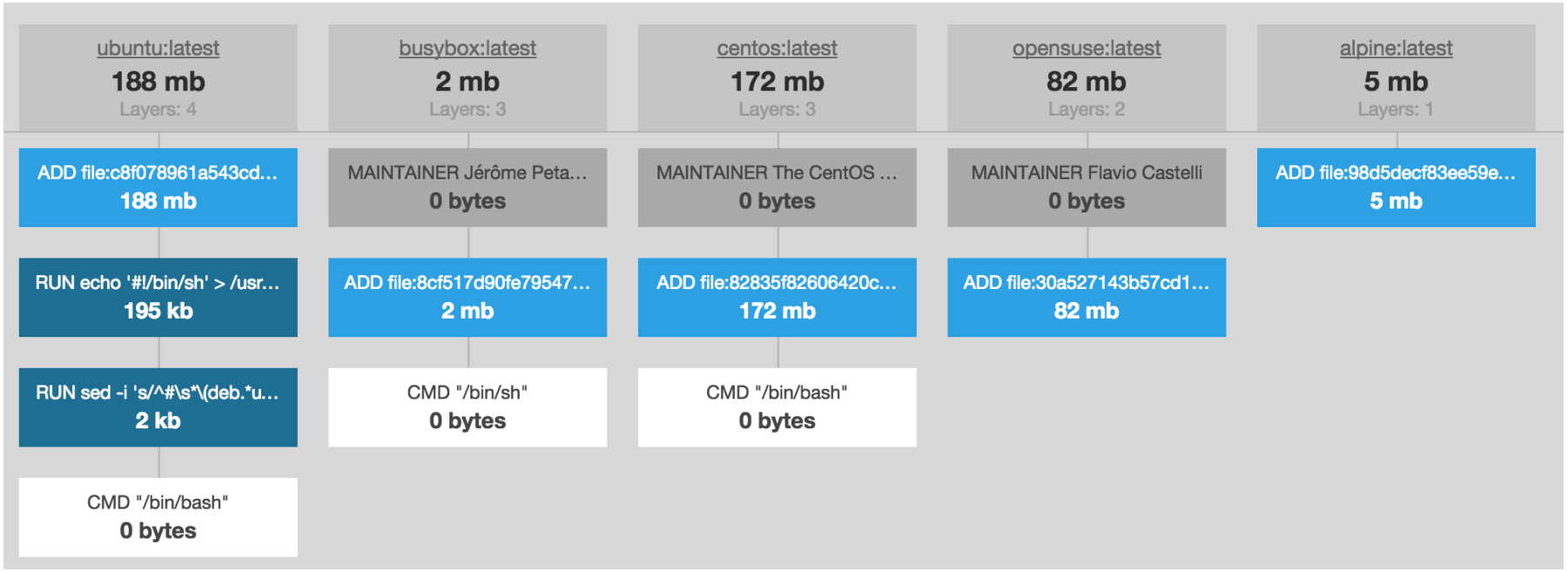
https://www.nginx.com/blog/introduction-to-microservices/

Z Axis (Data Partitioning)



https://www.nginx.com/blog/introduction-to-microservices/

Docker OS images size



https://imagelayers.io

Microservices

Each microservice exposes a service and is independent, i.e. each microservices includes the “server” to expose its API to other microservices or to the outside world, this leads us to create images with small footprint as possible to optimize resources needed to scale.

Micro didn't means micro endopint of a docker image with a size of n-Gigabyte.

Docker best practice

“In almost all cases, you should only run a single process in a single container. Decoupling applications into multiple containers makes it much easier to scale horizontally and reuse containers. If that service depends on another service, make use of container linking.”

https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/

We can use Docker to run microservices inside containers

lightweight services on Lightweight process

Each microservice has its own release cycle

reduce complexity per microservice

smaller independent releases

Continous Integration

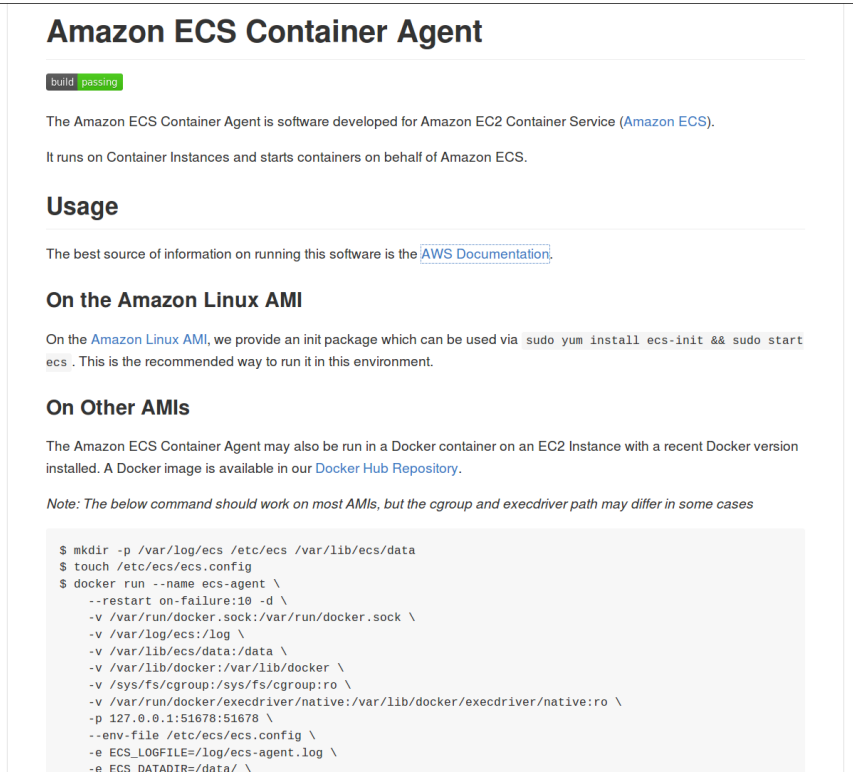
Continous Delivery / Deployment

Let's see
different approaches
to deploy our microservices
as a docker containers
to understand pro and cons

With plain docker
we can use
AWS EC2 Container Service
and
compose our microservices

ECS Container instances are EC2 instances with an Agent installed (GO)

https://github.com/aws/amazon-ecs-agent



Amazon ECS Container Agent

build passing

The Amazon ECS Container Agent is software developed for Amazon EC2 Container Service (Amazon ECS).

It runs on Container Instances and starts containers on behalf of Amazon ECS.

Usage

The best source of information on running this software is the AWS Documentation.

On the Amazon Linux AMI

On the Amazon Linux AMI, we provide an init package which can be used via `sudo yum install ecs-init && sudo start ecs`. This is the recommended way to run it in this environment.

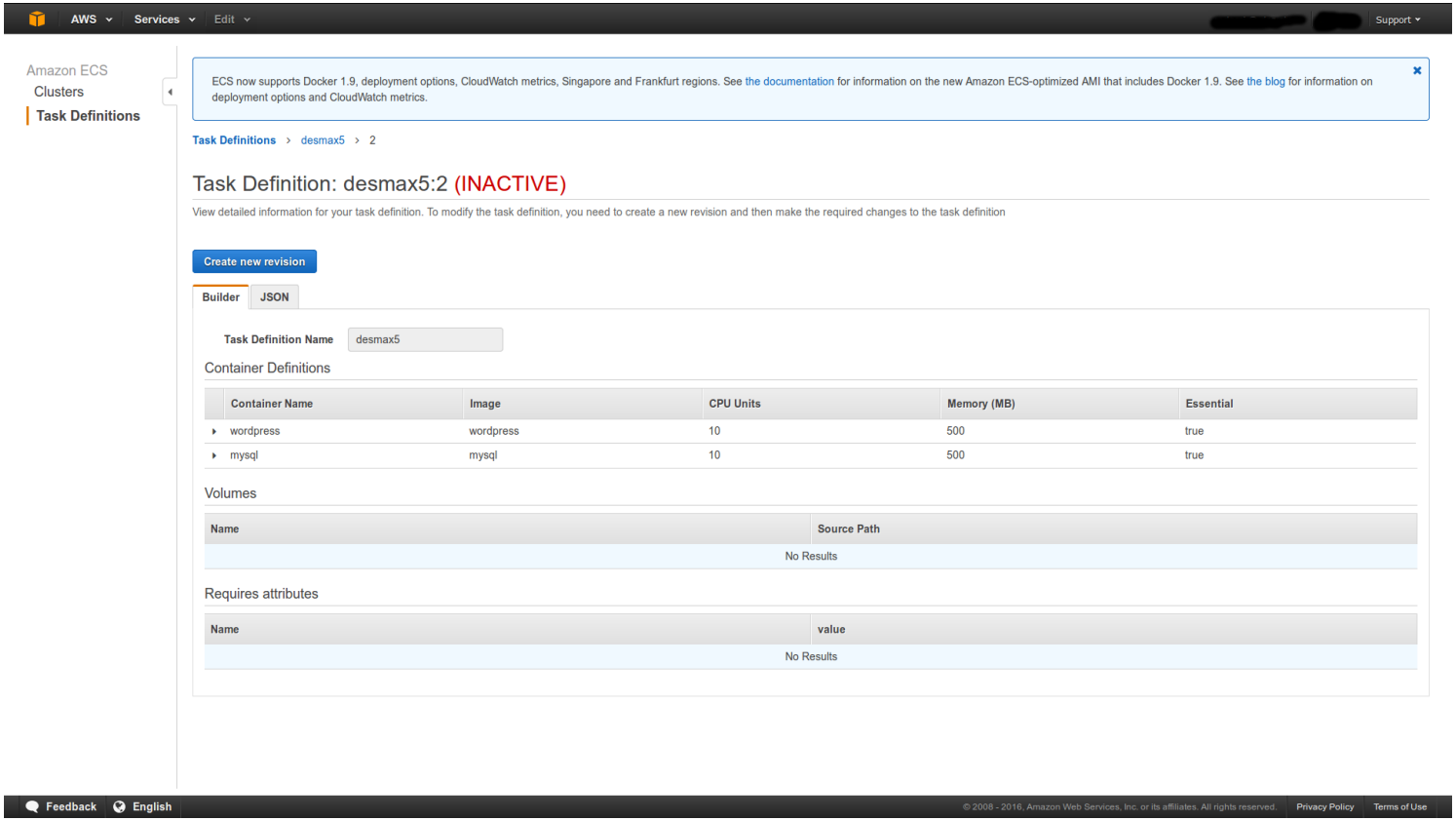
On Other AMIs

The Amazon ECS Container Agent may also be run in a Docker container on an EC2 Instance with a recent Docker version installed. A Docker image is available in our Docker Hub Repository.

Note: The below command should work on most AMIs, but the cgroup and execdriver path may differ in some cases

```
$ mkdir -p /var/log/ecs /etc/ecs /var/lib/ecs/data
$ touch /etc/ecs/ecs.config
$ docker run --name ecs-agent \
    --restart on-failure:10 -d \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v /var/log/ecs:/log \
    -v /var/lib/ecs/data:/data \
    -v /var/lib/docker:/var/lib/docker \
    -v /sys/fs/cgroup:/sys/fs/cgroup:ro \
    -v /var/run/docker/execdriver/native:/var/lib/docker/execdriver/native:ro \
    -p 127.0.0.1:51678:51678 \
    --env-file /etc/ecs/ecs.config \
    -e ECS_LOGFILE=/log/ecs-agent.log \
    -e ECS_DATADIR=/data/ \
```

You can compose microservices (like docker compose)



Docker on AWS ECS

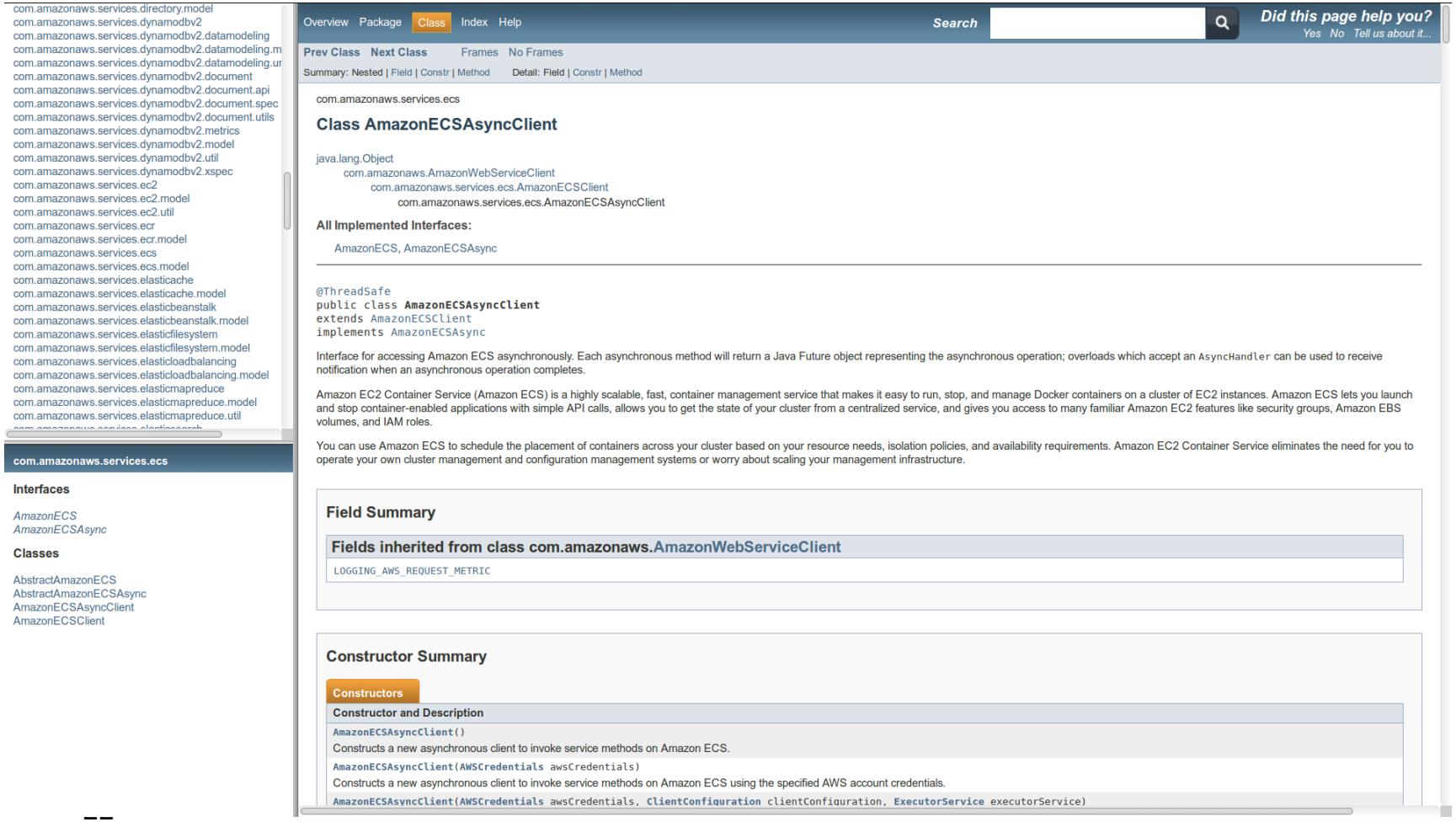

```
{
  "requiresAttributes": [],
  "taskDefinitionArn": "arn:aws:ecs:eu-west-1:073930265103:task-definition/console-sample-app-
static:7",
  "status": "ACTIVE",
  "revision": 7,
  "containerDefinitions": [
    {
      "volumesFrom": [],
      "memory": 300,
      "extraHosts": null,
      "dnsServers": null,
      "disableNetworking": null,
      "dnsSearchDomains": null,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "hostname": null,
      "essential": true,
      "entryPoint": [
        "sh",
        "-c"
      ],
```


As usual on AWS
every parts
is configurable
but not portable
outside AWS

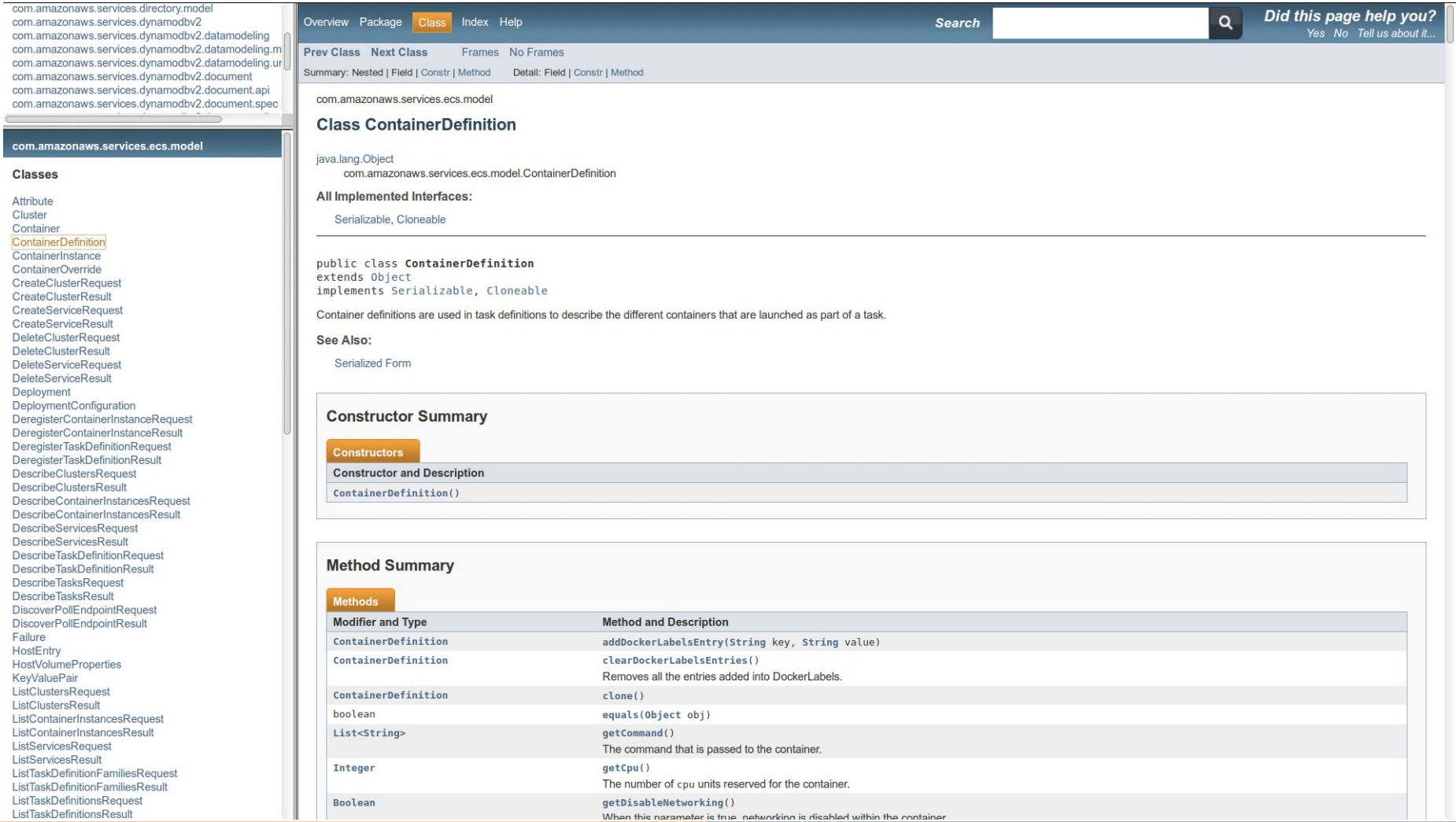
Docker on AWS ECS

```
      "mountPoints": [],
      "name": "simple-app",
      "ulimits": null,
      "dockerSecurityOptions": null,
      "environment": [],
      "links": [],
      "workingDirectory": null,
      "readonlyRootFilesystem": null,
      "image": "httpd:2.4",
      "command": [
        "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample App</title> <style>body
{margin-top: 40px; background-color: #333;} </style> </head><body> <div style=color:white;text-
align:center> <h1>Amazon ECS Sample App</h1> <h2>Congratulations!</h2> <p>Your application is
now running on a container in Amazon ECS.</p> </div></body></html>' >
/usr/local/apache2/htdocs/index.html && httpd-foreground\""
      ],
      "user": null,
      "dockerLabels": null,
      "logConfiguration": null,
      "cpu": 10,
      "privileged": null,
      "expanded": true
    }
  ],
  "volumes": [],
  "family": "console-sample-app-static"
}
```

Approchable through sync/async API (various languages sdk available)



Fully configurable through API



com.amazonaws.services.directory.model
com.amazonaws.services.dynamodbv2
com.amazonaws.services.dynamodbv2.datamodeling
com.amazonaws.services.dynamodbv2.datamodeling.m
com.amazonaws.services.dynamodbv2.datamodeling.ur
com.amazonaws.services.dynamodbv2.document
com.amazonaws.services.dynamodbv2.document.api
com.amazonaws.services.dynamodbv2.document.spec

com.amazonaws.services.ecs.model

Classes

Attribute
Cluster
Container
ContainerDefinition
ContainerInstance
ContainerOverride
CreateClusterRequest
CreateClusterResult
CreateServiceRequest
CreateServiceResult
DeleteClusterRequest
DeleteClusterResult
DeleteServiceRequest
DeleteServiceResult
Deployment
DeploymentConfiguration
DeregisterContainerInstanceRequest
DeregisterContainerInstanceResult
DeregisterTaskDefinitionRequest
DeregisterTaskDefinitionResult
DescribeClustersRequest
DescribeClustersResult
DescribeContainerInstancesRequest
DescribeContainerInstancesResult
DescribeServicesRequest
DescribeServicesResult
DescribeTaskDefinitionRequest
DescribeTaskDefinitionResult
DescribeTasksRequest
DescribeTasksResult
DiscoverPollEndpointRequest
DiscoverPollEndpointResult
Failure
HostEntry
HostVolumeProperties
KeyValuePair
ListClustersRequest
ListClustersResult
ListContainerInstancesRequest
ListContainerInstancesResult
ListServicesRequest
ListServicesResult
ListTaskDefinitionFamiliesRequest
ListTaskDefinitionFamiliesResult
ListTaskDefinitionsRequest
ListTaskDefinitionsResult

Overview Package Class Index Help Search Did this page help you? Yes No Tell us about it...

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

com.amazonaws.services.ecs.model

Class ContainerDefinition

java.lang.Object
com.amazonaws.services.ecs.model.ContainerDefinition

All Implemented Interfaces:

Serializable, Cloneable

```
public class ContainerDefinition
extends Object
implements Serializable, Cloneable
```

Container definitions are used in task definitions to describe the different containers that are launched as part of a task.

See Also:

Serialized Form

Constructor Summary

Constructors

Constructor and Description
`ContainerDefinition()`

Method Summary

Methods

Modifier and Type	Method and Description
`ContainerDefinition`	`addDockerLabelsEntry(String key, String value)`
`ContainerDefinition`	`clearDockerLabelsEntries()` Removes all the entries added into DockerLabels.
`ContainerDefinition`	`clone()`
`boolean`	`equals(Object obj)`
`List<String>`	`getCommand()` The command that is passed to the container.
`Integer`	`getCpu()` The number of `cpu` units reserved for the container.
`Boolean`	`getDisableNetworking()` When this parameter is true, networking is disabled within the container.

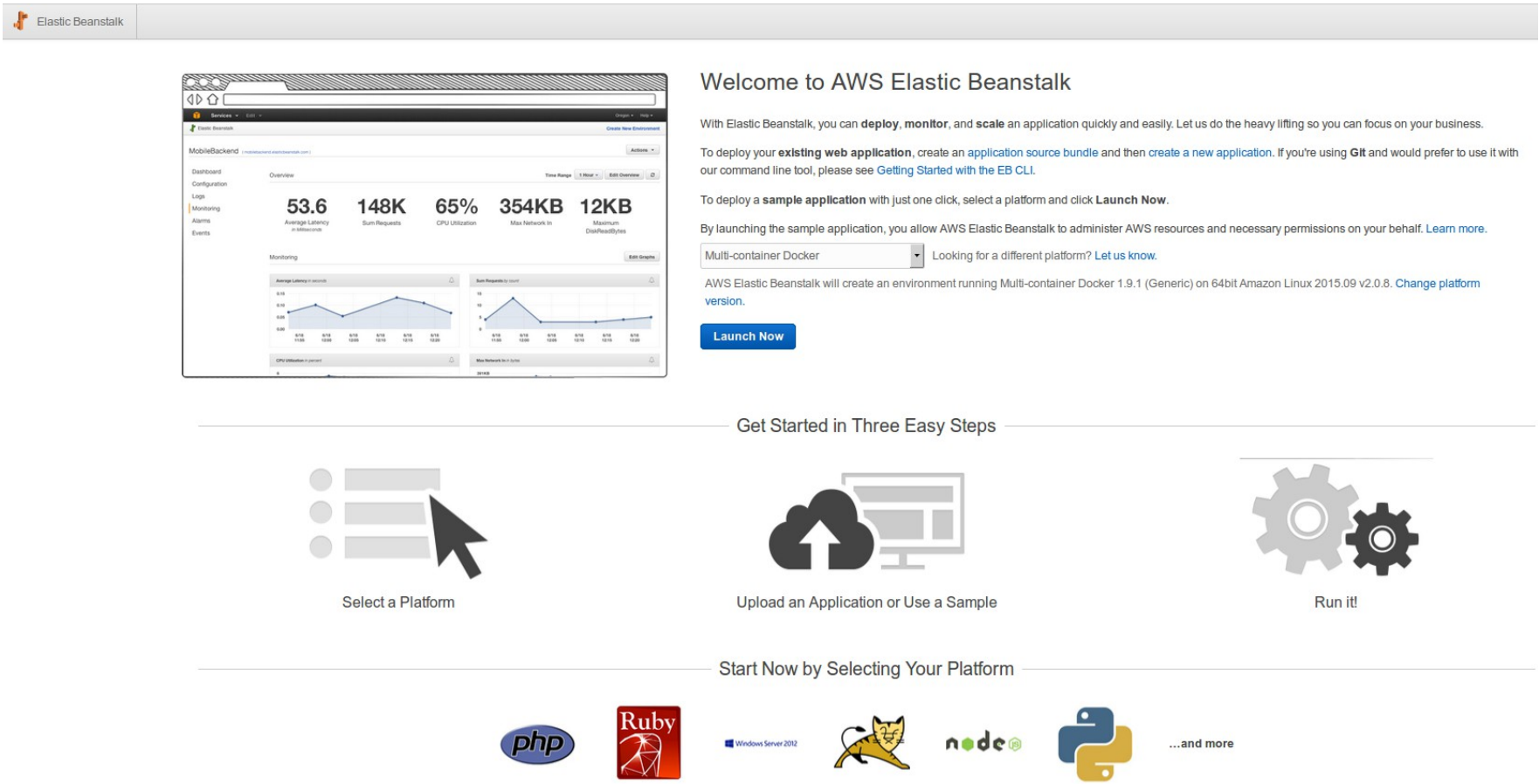
Pro of ECS:

- Fast way to deploy docker
- Cluster

Cons of ECS:

- Docker is a simple agent in the EC2 instance you haven't full control
- Conf not portable
- Test only on AWS no Vagrant/Vbox to simulate two or more instances togheter outside AWS

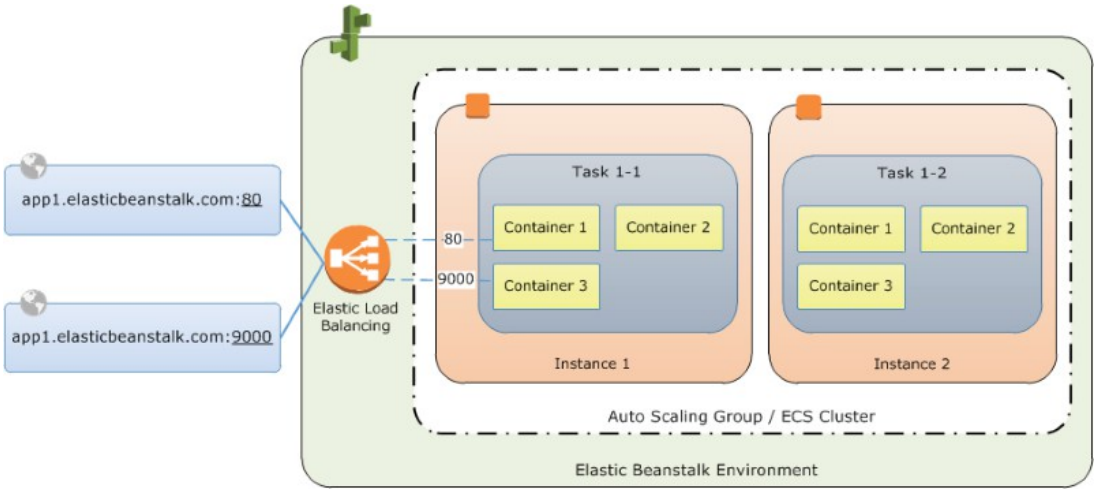
Docker on AWS Elastic bean stalk



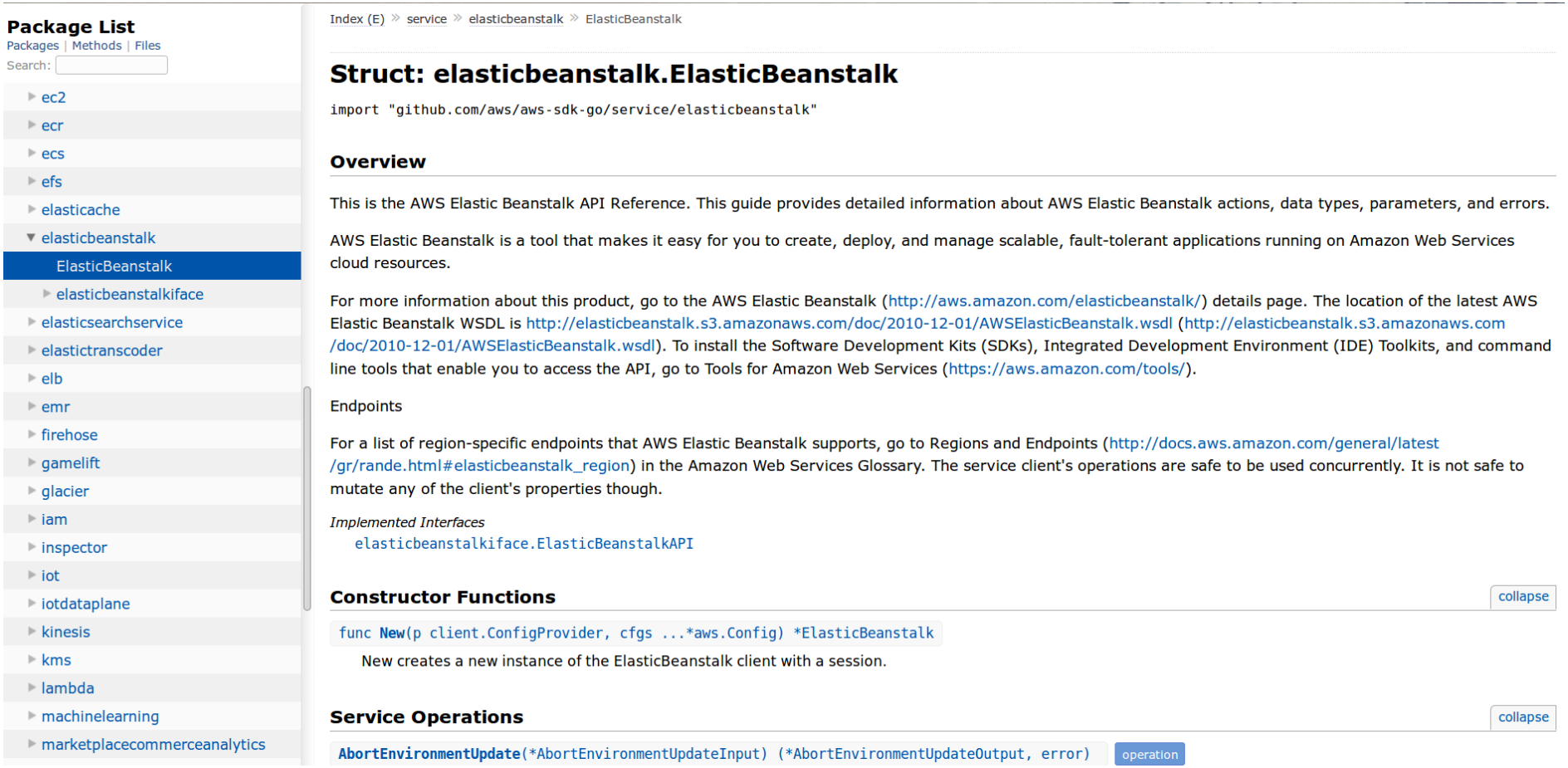
Docker on AWS Elastic bean stalk

Dockerrun.aws.json

```
{
  "AWSEBDockerrunVersion": 2,
  "volumes": [
    {
      "name": "php-app",
      "host": {
        "sourcePath": "/var/app/current/php-app"
      }
    },
    {
      "name": "nginx-proxy-conf",
      "host": {
        "sourcePath": "/var/app/current/proxy/conf.d"
      }
    }
  ],
  "containerDefinitions": [
    {
      "name": "php-app",
      "image": "php:fpm",
      "environment": [
        {
          "name": "Container",
          "value": "PHP"
        }
      ],
      "essential": true,
      "memory": 128,
      "mountPoints": [
        {
          "sourceVolume": "php-app",
          "containerPath": "/var/www/html",
          "readOnly": true
        }
      ]
    },
    {
      "name": "nginx-proxy",
      "image": "nginx",
      "essential": true,
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
        }
      ],
      "links": [
        "php-app"
      ],
      "mountPoints": [
        ….
      ]
    }
  ]
}
```



Docker on AWS Elastic bean stalk



Pro of EBS:

- Fast way to deploy your stack with docker
- Windows APP Support (not related with Docker)
- Cluster

Cons of EBS:

- Conf not portable
- Test only on AWS no Vagrant/Vbox to simulate two or more instances togheter outside AWS
- Not simple/clear cleanup of the resources used

Houston we have a problem

Who manages, controls, monitors and orchestrate and scales our microservices inside docker container ?

AWS provides cluster group and loadbalancer but is a coarse grained solution.

Some problems to solve

Manage: Who decides the number of containers with the same image control ?
Who decides the dedicated resources for each container ?
Who provide HA ?

Monitor: Who checks the health of the containers

Scale: Who acts as load balancer in front of the containers

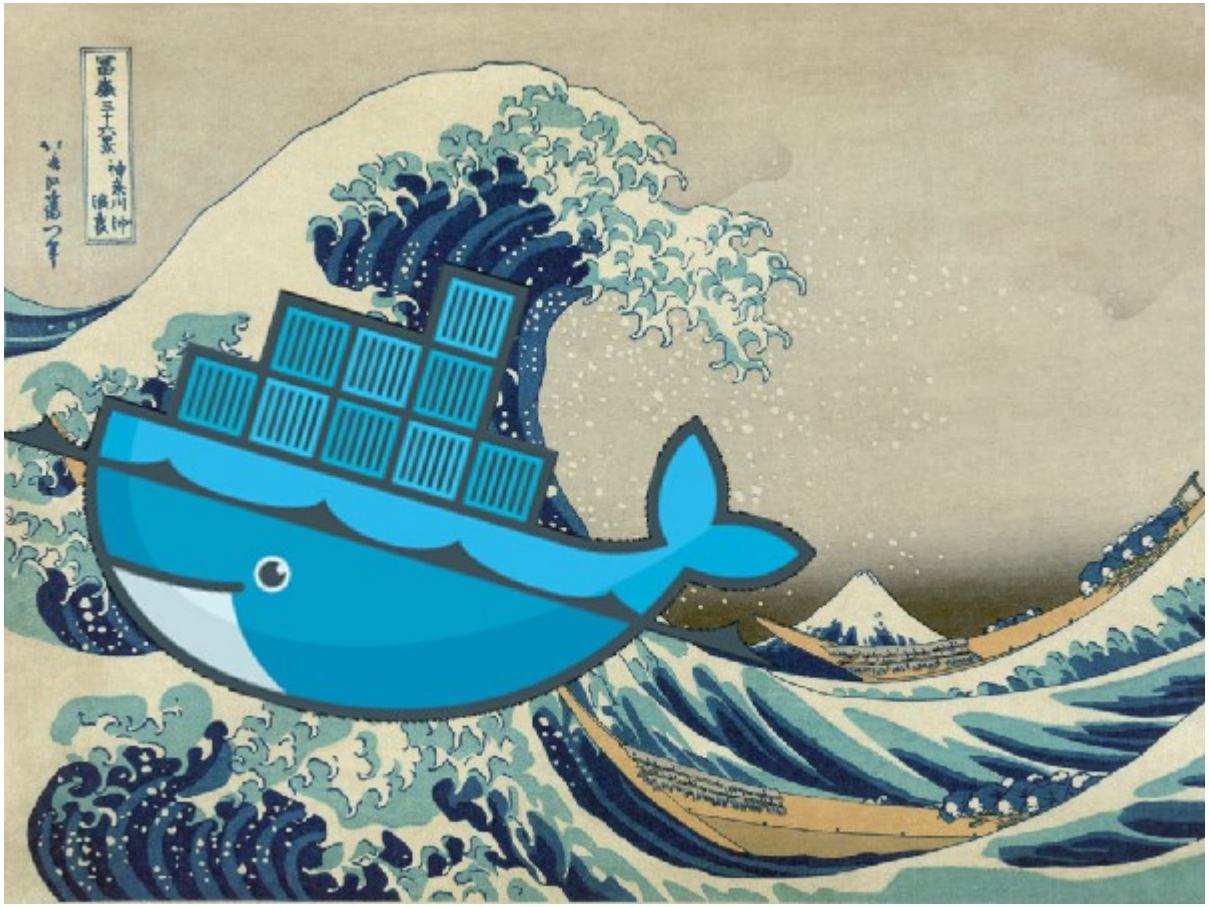
Some problems to solve

Orchestrate: Who puts the related container near each other ?

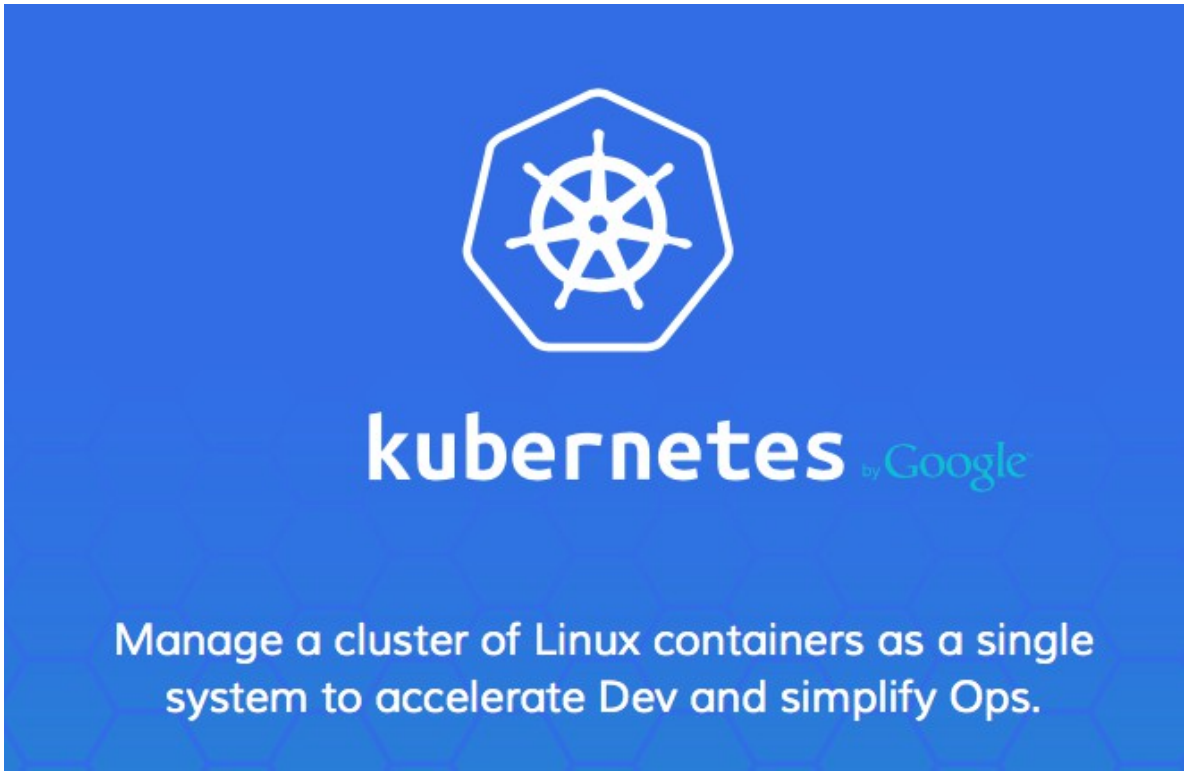
Who provides the notion of app stack with containers ?

Who provides endpoints ?

Now we start to play the Rock & Roll



K8s

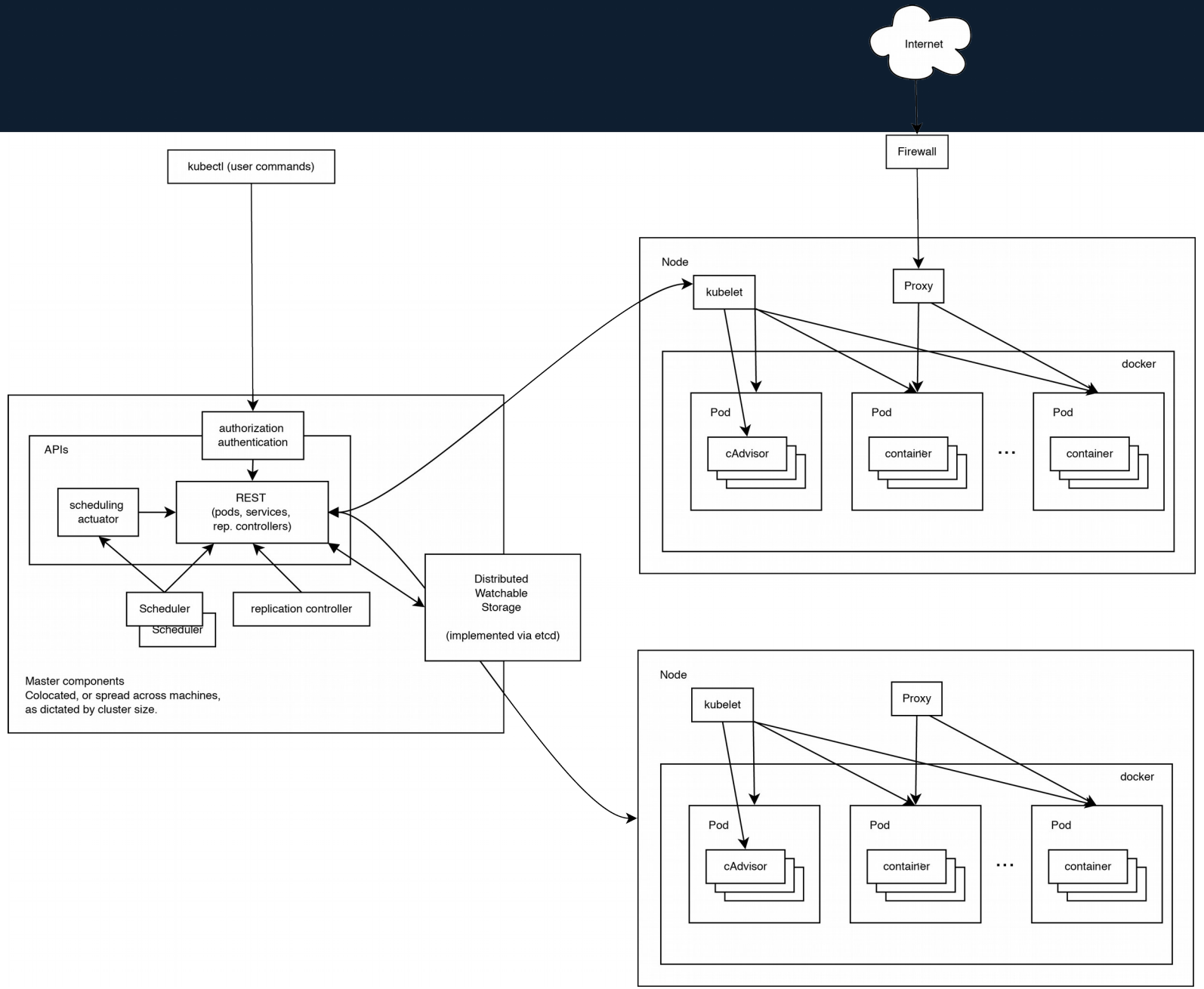


Companies

We are working with a broad group of companies to make sure that Kubernetes works well for everyone, from individual developers to the largest companies in the cloud space.

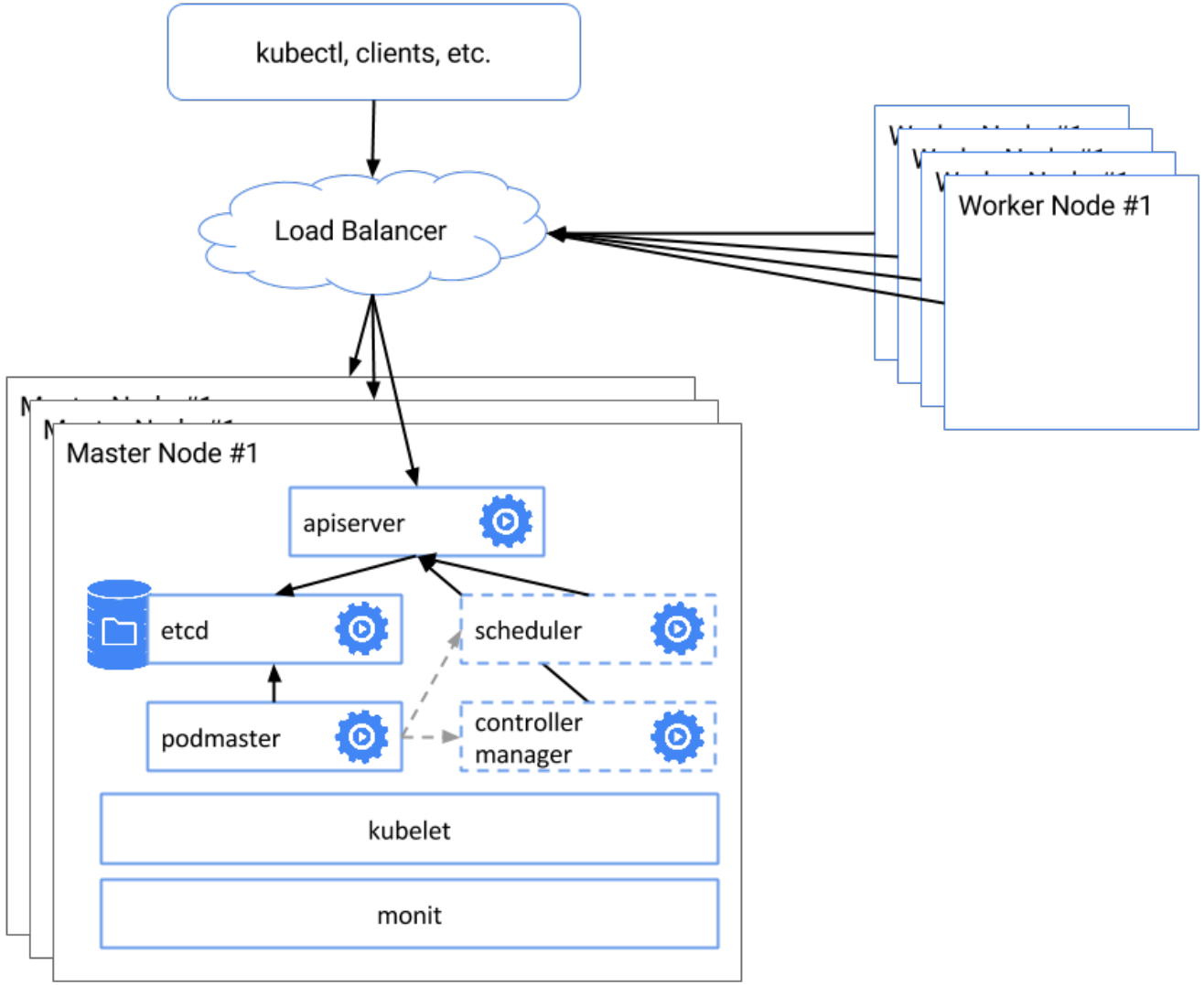
redhat intel CoreOS puppet labs sysdig

https://github.com/kubernetes



http://kubernetes.io/v1.1/docs/design/architecture.html

K8s HA



http://kubernetes.io/v1.1/docs/admin/high-availability.html

Note

Docker and Kubernetes are written in Go
and are released with
Open Source license,
you can learn, hack and contribute
every single line of code

Names the pieces

Master: Is the core of the cluster, with restful commands define our desidered cluster, it contains the API Server

Scheduler: Schedule workloads in terms of Pods on the nodes. It spread pods across the cluster for matching pods replicas.

Etcd: Distributed configuration store, contains the kubernetes state

Node-n

kubelet: Interact with the API Server

Kube-proxy: provides load balancing and directs the traffic to specific service to ther proper pod

Pod: is the entity to aggregate related containers and the data, close in terms of network and HW.
Pods allow us to logically group containers.
Eaxch pods has its unique ip Address

Service:
Provides a reliable endpoint for the pods.
A service can be implemented internally using podSelector or externally with Endpoints.
Each service has its unique ip address.
Services are visible internally and can be exposed externally

ReplicationController: Manage the number of nodes of the pods and Container included to satisfy the desidered number in the conf.

We interact with kubernetes throught UI or with kubectl

```
kubectl create -f mysql-pod.yaml

kubectl create -f mysql-service.yaml

kubectl create -f wildfly-rc.yaml

kubectl get pods

kubectl get services

kubectl get replicationcontrollers
…
```

http://kubernetes.io/docs/user-guide/kubectl/kubectl/

Javaee yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
  labels:
    name: mysql-pod
    context: docker-k8s-lab
spec:
  ports:
    # the port that this service should
serve on
    - port: 3306
  # label keys and values that must
match in order to receive traffic for
this service
  selector:
    name: mysql-pod
    context: docker-k8s-lab
```

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: wildfly-rc
  labels:
    name: wildfly
    context: docker-k8s-lab
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: wildfly
    spec:
      containers:
      - name: wildfly-rc-pod
        image: arungupta/wildfly-mysql-
javaee7:k8s
        ports:
        - containerPort: 8080
```

nginx-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: mysql-pod
  labels:
    name: mysql-pod
    context: docker-k8s-lab
spec:
  containers:
    -
      name: mysql
      image: mysql:latest
      env:
        -
          name: "MYSQL_USER"
          value: "mysql"
        -
          name: "MYSQL_PASSWORD"
          value: "mysql"
        -
          name: "MYSQL_DATABASE"
          value: "sample"
        -
          name: "MYSQL_ROOT_PASSWORD"
          value: "supersecret"
      ports:
        -
          containerPort: 3306
```

Ephemereal

Docker container are ephemeral,
You can use a persistent volume, but if a POD is removed the persistent volume will be deleted.
On AWS we can use Elastic Block Store

Volumes

```
apiVersion:  v1
kind:    Pod
metadata:
        name:    test-aws
Spec:
    containers:
        -    image:   nginx:latest
             ports:
        -    containerPort:   80
             name:    test-aws
             volumeMounts:
        -    mountPath:   /usr/share/nginx/html
             name:    aws-pd
     volumes:
        -    name:    aws-pd
             awsElasticBlockStore:
             volumeID:    aws://<availability-zone>/<volume-id>
             fsType:  ext4
```

Volume types available :

EmptyDir, hostPath, gcePersistentDisk, awsElasticBlockStore, nfs, iscsi, flocker, glusterfs, rbd,

GitRepo, secret, persistentVolumeClaim

Multitenancy

By default the namespaces created in kubernetes are

`kube-system` for system level containers

`default` for the containers created by the user

We can create other namespaces with

```
apiVersion: v1

kind: Namespace

metadata:

      name: test
```

Now we can use this fresh namespace for new pods

```
apiVersion:  v1
kind:    Pod
metadata:
         name:    utility
         namespace:   test
spec:
         containers:
         -    image:   debian:latest
                  command:
                           -    sleep
                           -    "3600"
                  name:    utility
```

The pod can access services in other namespace but with using this naming

```
<service-name>.<namespace-name>.cluster.local
```

Scaling

When your application needs more resources you can

Add resources, not deleting Replication controller and related pods, but adding other pods.

First we ask number of the pods

```
kubectl get pods -l  name=<pod_name>
```

And then increase

```
kubectl scale –replicas=<desidered-number> rc/<pod-name>
```

Dev on K8s

Write code

Put code/artifact into docker image

Write kubernetes metadata

Apply the metadata

Choose size and updates when you need

K8s distribution examples

Pro of Kubernetes:

- Open Source (Golang)/Community
- Tested (former Google Borg Project)
- Deployable on bare metal or public Cloud public (AWS, GCE, Azure) or private Cloud or virtualized env with VMWare vSphere

Pro of Kubernetes (continuation):

- Testable locally with Vagrant and virtual machines
- Portable configurations (pods/service/controllers)
- Dashboard, Grafana, Metrics

Pro of Kubernetes:

Under control of two foundations
i.e standard

(containers)

(orchestration)

Open Container initiative (OCI)

The Open Container Initiative is a lightweight, open governance structure, formed under the auspices of the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtime.

...

Docker is donating its container format and runtime, runC, to the OCI to serve as the cornerstone of this new effort. It is available now at https://github.com/opencontainers/runc.

https://www.opencontainers.org/

https://github.com/opencontainers/specs

Cloud native Initiative Foundation (CNCF)

The Cloud Native Computing Foundation's mission is to create and drive the adoption of a new computing paradigm that is optimized for modern distributed systems environments.

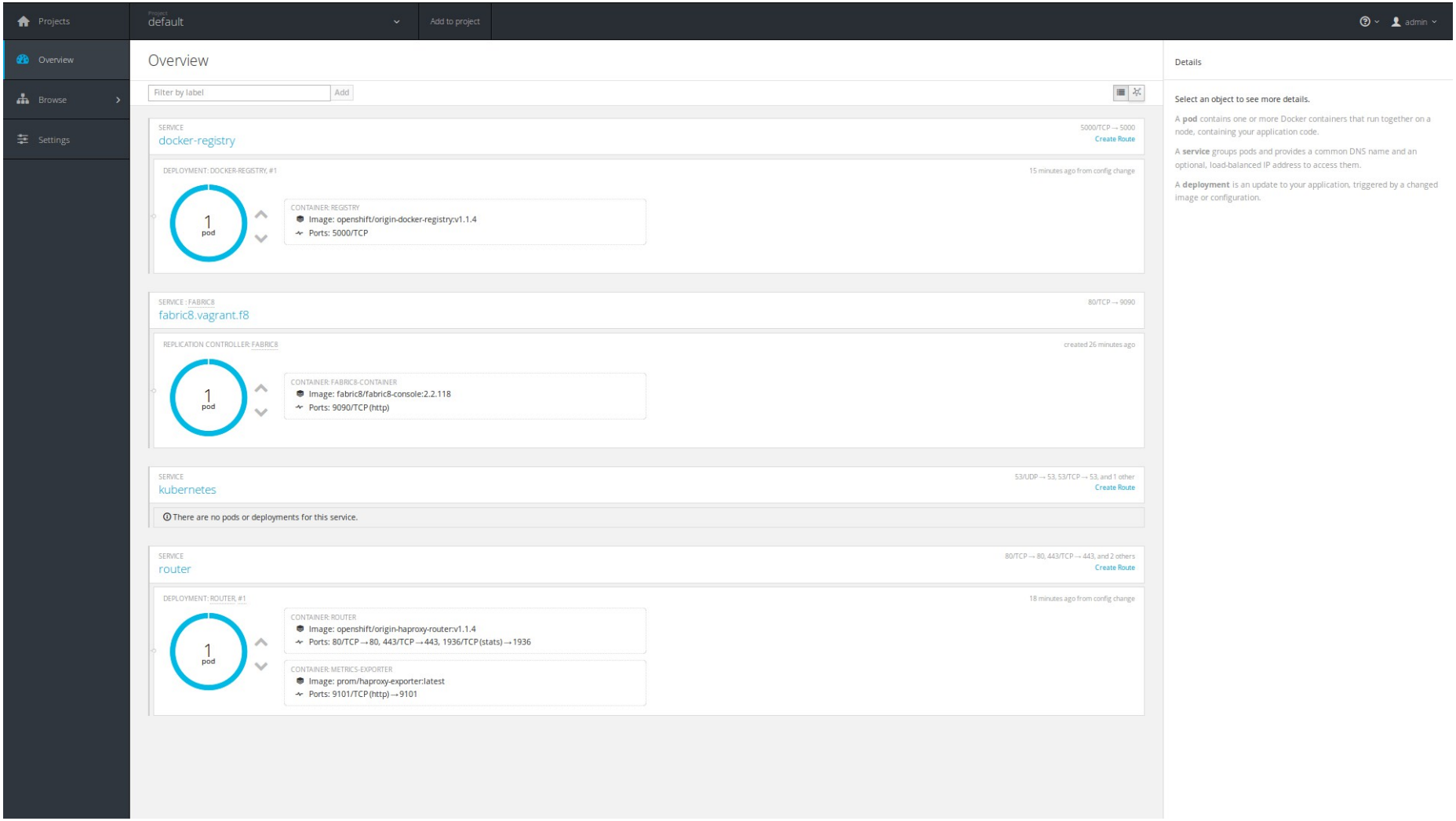
https://cncf.io

Ok I like
Docker and K8s
but I want more

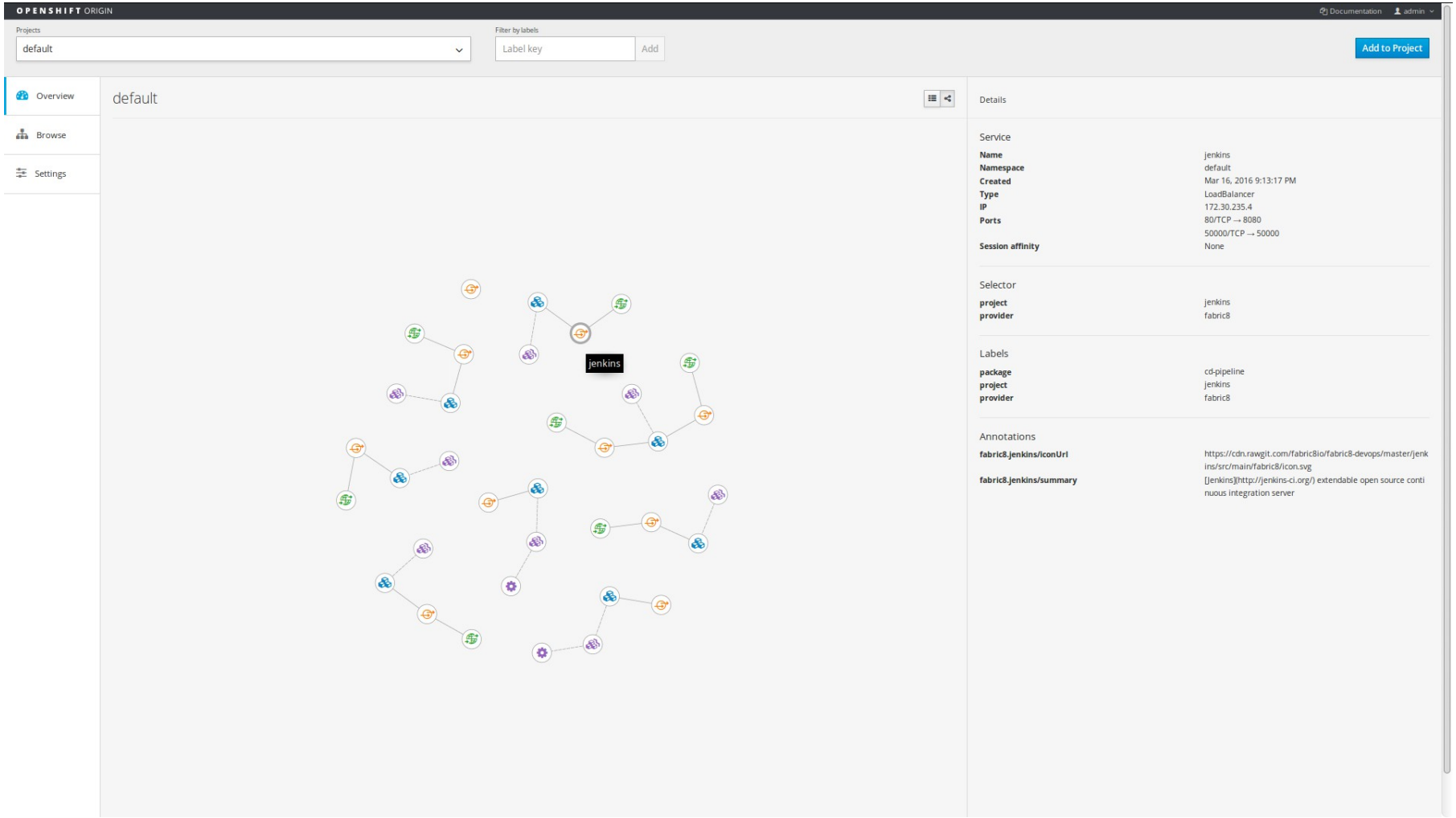
Docker & K8s with Devops tool, deployable everywhere, included notebook with Vagrant and Virtualbox
(included a commercial version)

OPENSHIFT
by Red Hat

Openshift


Projects
default
Add to project
admin
Overview
Browse
Settings
Overview
Filter by label
Add
SERVICE
docker-registry
5000/TCP → 5000
Create Route
DEPLOYMENT: DOCKER-REGISTRY, #1
15 minutes ago from config change
1
pod
CONTAINER: REGISTRY
Image: openshift/origin-docker-registry:v1.1.4
Ports: 5000/TCP
SERVICE : FABRIC8
fabric8.vagrant.f8
80/TCP → 9090
REPLICATION CONTROLLER: FABRIC8
created 26 minutes ago
1
pod
CONTAINER: FABRIC8-CONTAINER
Image: fabric8/fabric8-console:2.2.118
Ports: 9090/TCP (http)
SERVICE
kubernetes
53/UDP → 53, 53/TCP → 53, and 1 other
Create Route
There are no pods or deployments for this service.
SERVICE
router
80/TCP → 80, 443/TCP → 443, and 2 others
Create Route
DEPLOYMENT: ROUTER, #1
18 minutes ago from config change
1
pod
CONTAINER: ROUTER
Image: openshift/origin-haproxy-router:v1.1.4
Ports: 80/TCP → 80, 443/TCP → 443, 1936/TCP (stats) → 1936
CONTAINER: METRICS-EXPORTER
Image: prom/haproxy-exporter:latest
Ports: 9101/TCP (http) → 9101
Details
Select an object to see more details.
A pod contains one or more Docker containers that run together on a node, containing your application code.
A service groups pods and provides a common DNS name and an optional, load-balanced IP address to access them.
A deployment is an update to your application, triggered by a changed image or configuration.

Openshift


OPENSHIFT ORIGIN
Documentation
admin
Projects
default
Filter by labels
Label key
Add
Add to Project
Overview
Browse
Settings
default
jenkins
Details
Service
Name jenkins
Namespace default
Created Mar 16, 2016 9:13:17 PM
Type LoadBalancer
IP 172.30.235.4
Ports 80/TCP → 8080
50000/TCP → 50000
Session affinity None
Selector
project jenkins
provider fabric8
Labels
package cd-pipeline
project jenkins
provider fabric8
Annotations
fabric8.jenkins/iconUrl https://cdn.rawgit.com/fabric8io/fabric8-devops/master/jenkins/src/main/fabric8/icon.svg
fabric8.jenkins/summary [Jenkins](http://jenkins-ci.org/) extendable open source continuous integration server

Deployable on your local machine

Openshift commons

“Commons builds connections and collaboration across OpenShift communities, projects and stakeholders. In doing so we'll enable the success of customers, users, partners, and contributors as we deepen our knowledge and experiences together.

Our goals go beyond code contributions. Commons is a place for companies using OpenShift to accelerate its success and adoption. To do this we'll act as resources for each other, share best practices and provide a forum for peer-to-peer communication.”

http://commons.openshift.org/index.html#colleagues

Openshift commons 1

http://commons.openshift.org/index.html#colleagues

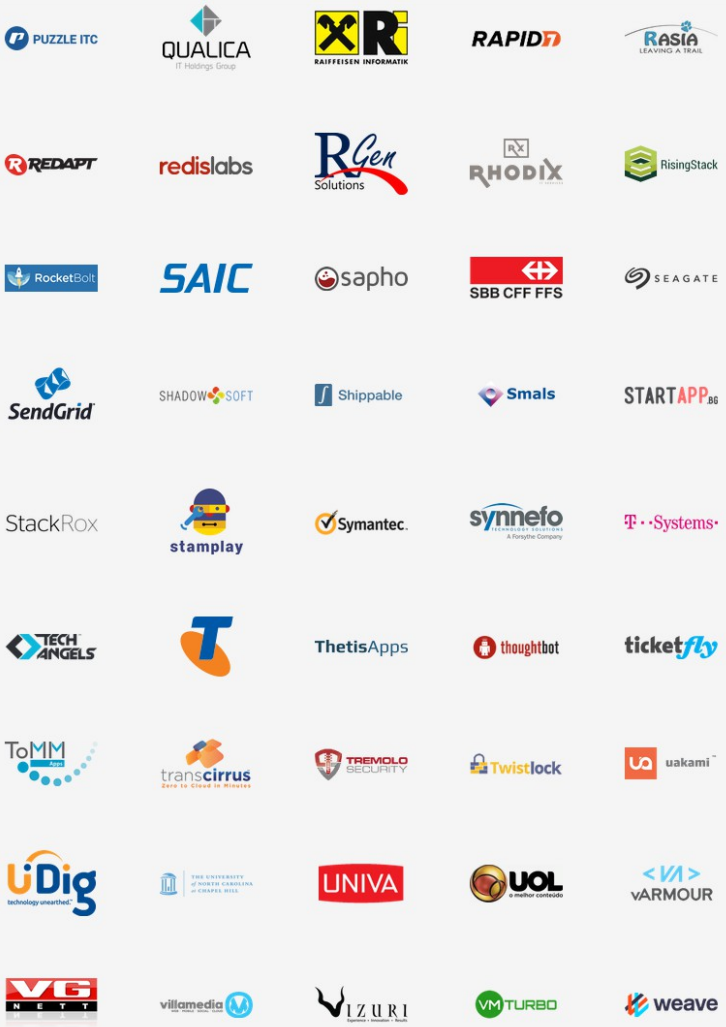
Openshift commons 2

http://commons.openshift.org/index.html#colleagues

Openshift commons 3

http://commons.openshift.org/index.html#colleagues

Openshift commons 4



http://commons.openshift.org/index.html#colleagues

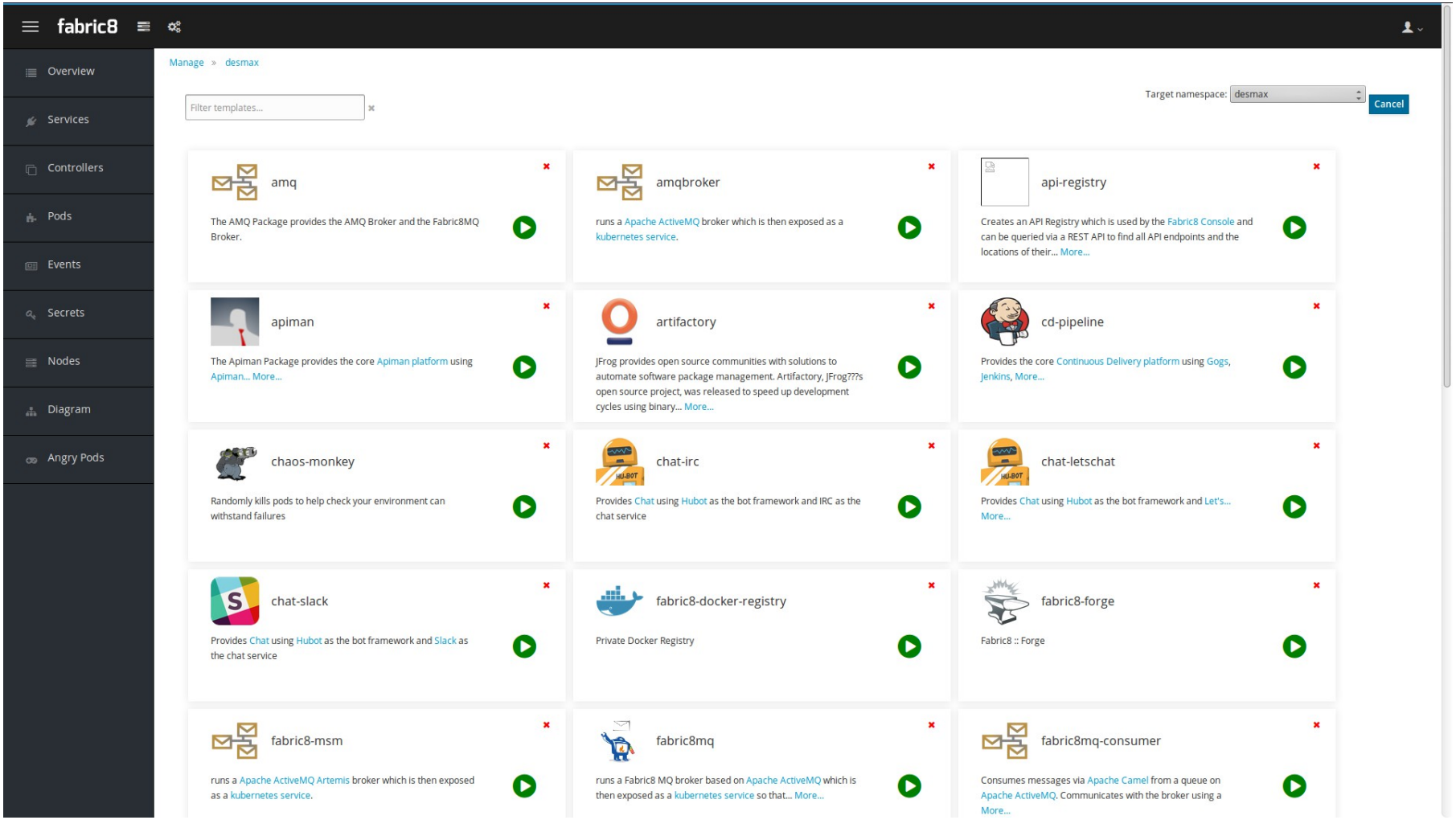
Openshift commons 5

http://commons.openshift.org/index.html#colleagues

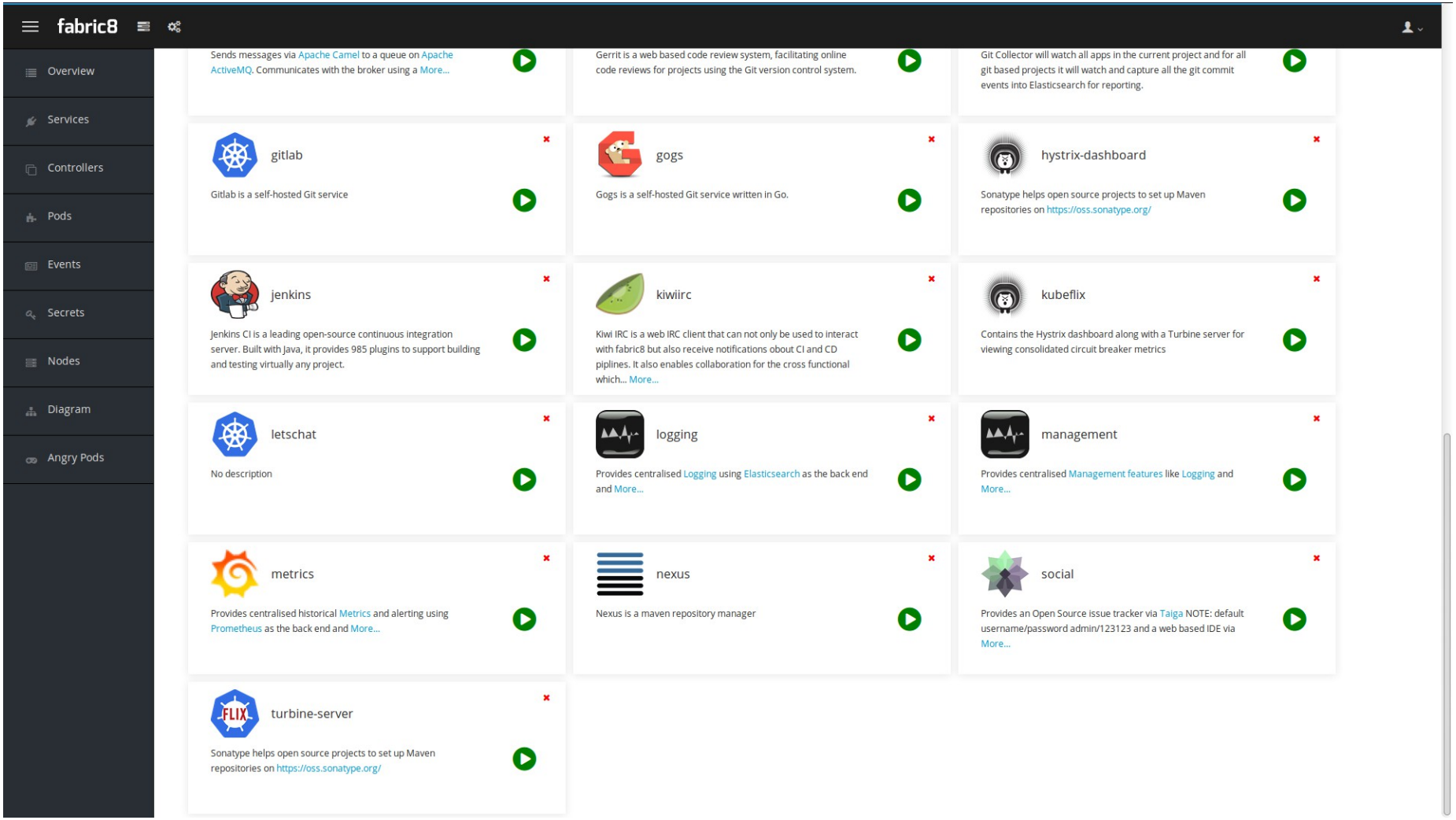
Ok I like
Docker and K8s
and Openshift but…
I want a ready Paas DevOps

fabric8

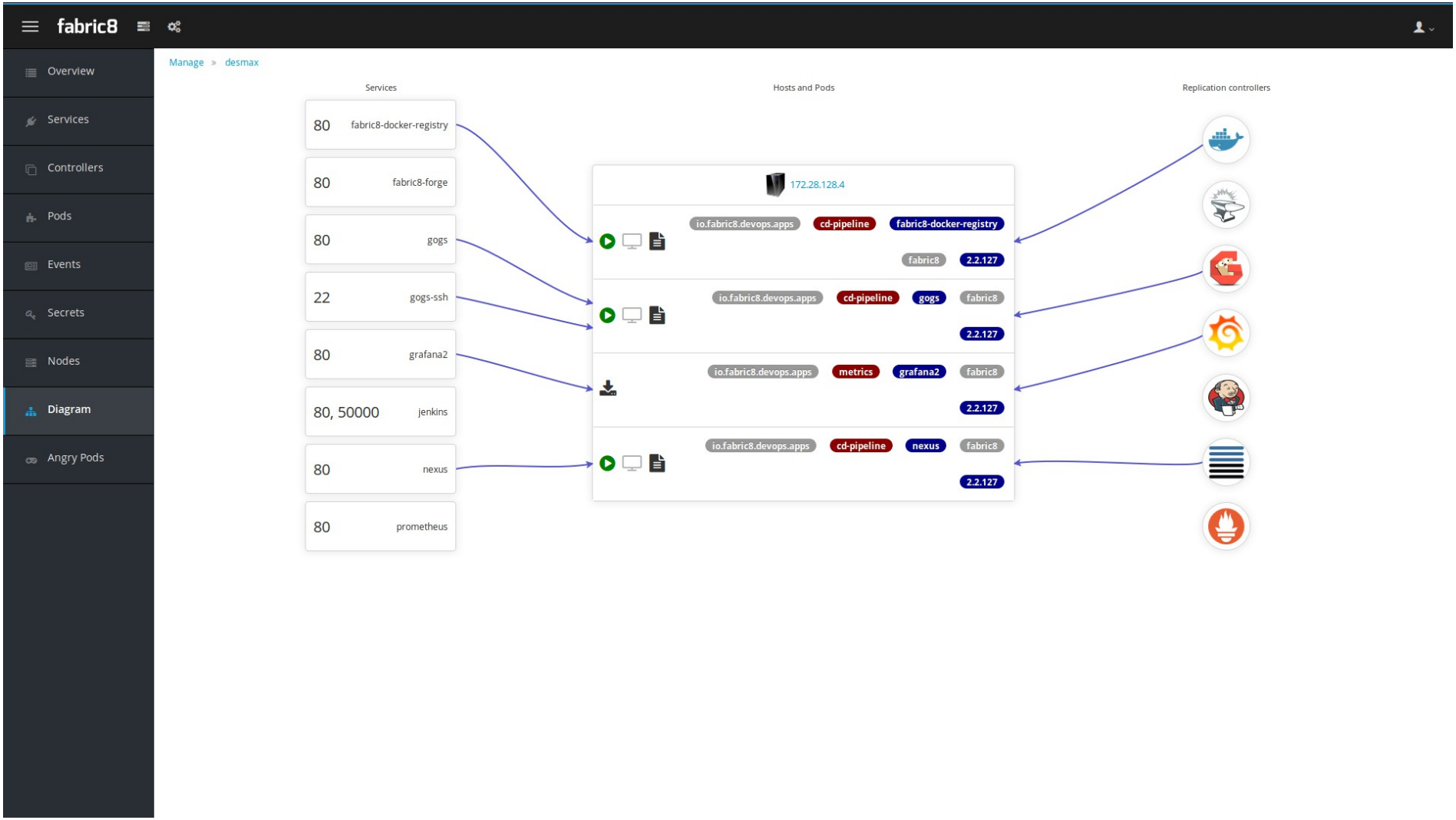
Fabric8


fabric8
Overview
Services
Controllers
Pods
Events
Secrets
Nodes
Diagram
Angry Pods
Manage » desmax
Filter templates...
Target namespace: desmax
Cancel
amq
The AMQ Package provides the AMQ Broker and the Fabric8MQ Broker.
amqbroker
runs a Apache ActiveMQ broker which is then exposed as a kubernetes service.
api-registry
Creates an API Registry which is used by the Fabric8 Console and can be queried via a REST API to find all API endpoints and the locations of their... More...
apiman
The Apiman Package provides the core Apiman platform using Apiman... More...
artifactory
JFrog provides open source communities with solutions to automate software package management. Artifactory, JFrog???s open source project, was released to speed up development cycles using binary... More...
cd-pipeline
Provides the core Continuous Delivery platform using Gogs, Jenkins, More...
chaos-monkey
Randomly kills pods to help check your environment can withstand failures
chat-irc
Provides Chat using Hubot as the bot framework and IRC as the chat service
chat-letschat
Provides Chat using Hubot as the bot framework and Let's... More...
chat-slack
Provides Chat using Hubot as the bot framework and Slack as the chat service
fabric8-docker-registry
Private Docker Registry
fabric8-forge
Fabric8 :: Forge
fabric8-msm
runs a Apache ActiveMQ Artemis broker which is then exposed as a kubernetes service.
fabric8mq
runs a Fabric8 MQ broker based on Apache ActiveMQ which is then exposed as a kubernetes service so that... More...
fabric8mq-consumer
Consumes messages via Apache Camel from a queue on Apache ActiveMQ. Communicates with the broker using a More...

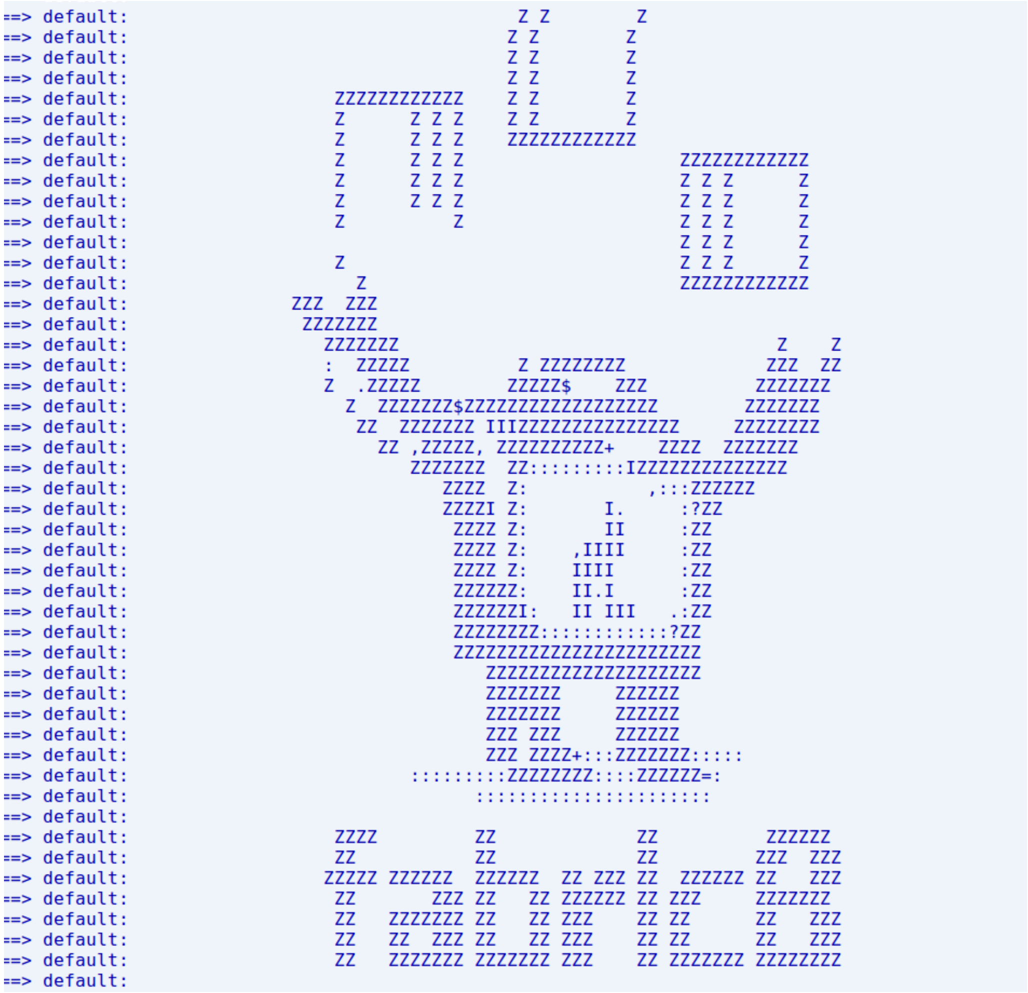
Fabric8


fabric8
Overview
Services
Controllers
Pods
Events
Secrets
Nodes
Diagram
Angry Pods
Sends messages via Apache Camel to a queue on Apache ActiveMQ. Communicates with the broker using a More...
Gerrit is a web based code review system, facilitating online code reviews for projects using the Git version control system.
Git Collector will watch all apps in the current project and for all git based projects it will watch and capture all the git commit events into Elasticsearch for reporting.
gitlab
Gitlab is a self-hosted Git service
gogs
Gogs is a self-hosted Git service written in Go.
hystrix-dashboard
Sonatype helps open source projects to set up Maven repositories on https://oss.sonatype.org/
jenkins
Jenkins CI is a leading open-source continuous integration server. Built with Java, it provides 985 plugins to support building and testing virtually any project.
kiwiirc
Kiwi IRC is a web IRC client that can not only be used to interact with fabric8 but also receive notifications about CI and CD piplines. It also enables collaboration for the cross functional which... More...
kubeflix
Contains the Hystrix dashboard along with a Turbine server for viewing consolidated circuit breaker metrics
letschat
No description
logging
Provides centralised Logging using Elasticsearch as the back end and More...
management
Provides centralised Management features like Logging and More...
metrics
Provides centralised historical Metrics and alerting using Prometheus as the back end and More...
nexus
Nexus is a maven repository manager
social
Provides an Open Source issue tracker via Taiga NOTE: default username/password admin/123123 and a web based IDE via More...
turbine-server
Sonatype helps open source projects to set up Maven repositories on https://oss.sonatype.org/

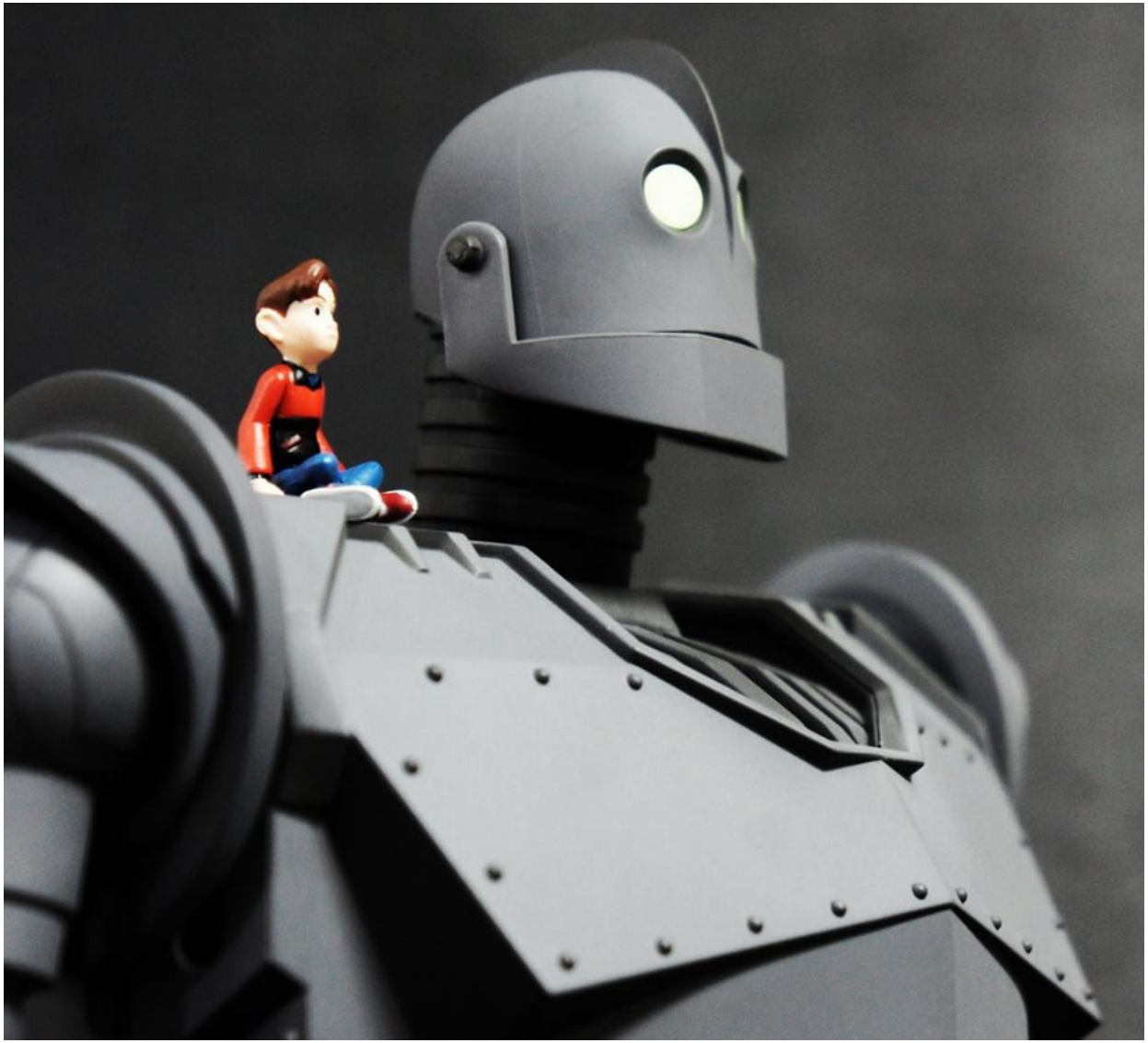
Fabric8


fabric8
Overview
Services
Controllers
Pods
Events
Secrets
Nodes
Diagram
Angry Pods
Manage » desmax
Services
Hosts and Pods
Replication controllers
80 fabric8-docker-registry
80 fabric8-forge
80 gogs
22 gogs-ssh
80 grafana2
80, 50000 jenkins
80 nexus
80 prometheus
172.28.128.4
io.fabric8.devops.apps
cd-pipeline
fabric8-docker-registry
fabric8
2.2.127
io.fabric8.devops.apps
cd-pipeline
gogs
fabric8
2.2.127
io.fabric8.devops.apps
metrics
grafana2
fabric8
2.2.127
io.fabric8.devops.apps
cd-pipeline
nexus
fabric8
2.2.127

Fabric8



Q & A



Resources

http://www.docker.com/

http://kubernetes.io/

http://aws.amazon.com/documentation/ecs/

https://aws.amazon.com/documentation/elastic-beanstalk/

https://www.openshift.org/

http://fabric8.io/

{ Thanks!

@desmax74
http://slideshare.net/desmax74