# Lighthouse

Senior Design Team May15-17

Iowa State University - Ames, IA

May 1st, 2015

## About the Team

| | |
|---|---|
| Caleb Brose | Team Lead |
| Chris Fogerty | Communication Lead |
| Zach Taylor | Key Concept Holder |
| Rob Sheehy | Key Concept Holder |
| Nick Miller | Web Designer |

Thanks to:

- Dr. Mitra, CS Dept. - Advisor
- Dave Tucker, Workiva - Client

## Topics

1. Docker
2. Lighthouse
3. Design
4. Testing and Quality Process

## What is Docker?

“Docker is an open platform for developers and sysadmins to build, ship and run distributed applications”

- docker.com

Still confused? So were we.

## Why use Docker?

### Situation

- As a Release Manager for a distributed application, I want to ensure
  - Consistency - Developer computers aren’t production servers
  - Stability - Nodes will fail in distributed applications
  - Modularity - Program dependencies and environments need to be isolated

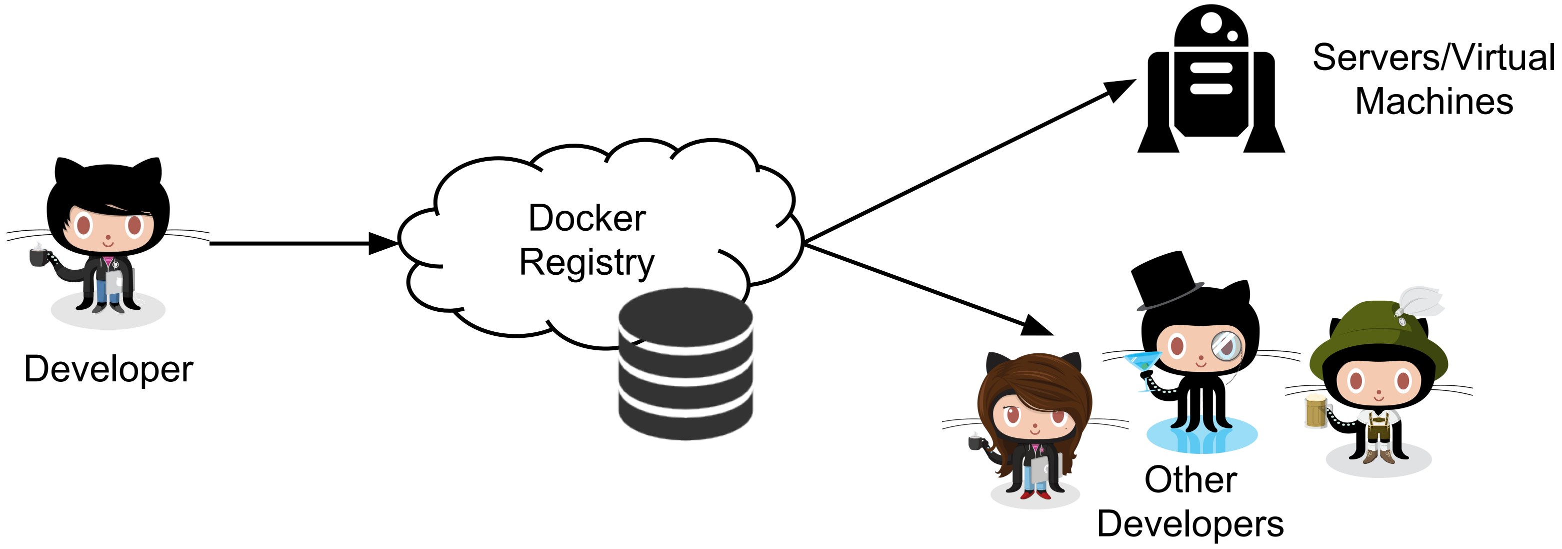
### The Solution

- Docker
  - Images ensure consistent environments and files
  - Containers ensure stability & runtime isolation

## Docker Images

### What

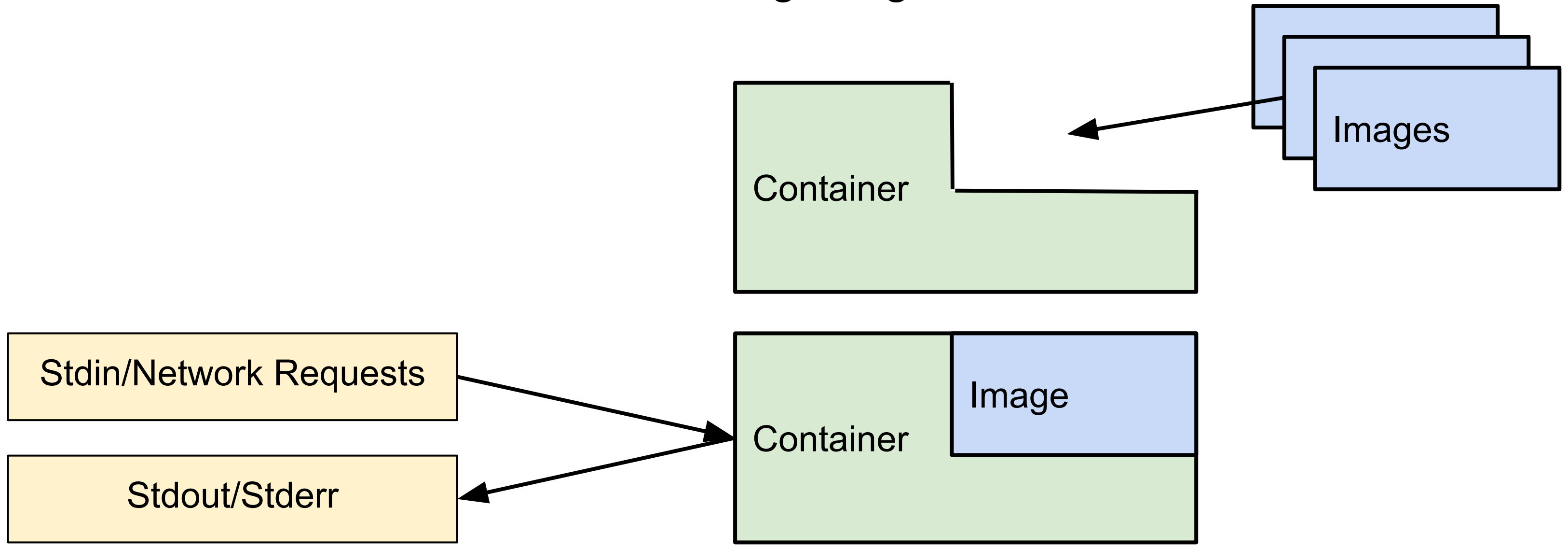
- An immutable snapshot of an operating system to share with other machines running docker.



# Docker Containers

## What

- A sandboxed instance of a running image.



# Hello World, Docker

```
> docker pull debian
 /* wait a sec to download 84.89 MBs */
> docker images
 /* list of images */
> docker run -ti debian echo 'hello world'
   hello world
> docker ps -a
 /* list of containers */
```

## Why use Lighthouse?

Docker is great, **but** I want to:

- Manage my application as a whole, not as individual Docker instances
- Utilize cloud providers like GCE or AWS to run my application
- Control who can change my application
- Do all of this through a user interface

The Solution

- A web-based tool for administrating hundreds of Docker programs across multiple networks

## Lighthouse

## Functional Requirements

Docker core functionality

- Container/image management

Lighthouse custom functionality

- Application level control - deployments, new versions, rollbacks
- Cloud provider interfacing

Application analysis

- Logs, history, usages, etc

User management

## Non-Functional Requirements

Security

- Authenticated requests
- Authorized users

Extensibility and documentation

- Additional functionality should be straightforward
- Users should be able to create their own frontend

Low latency

- Early error detection in the pipeline

## Market Survey

### Related products

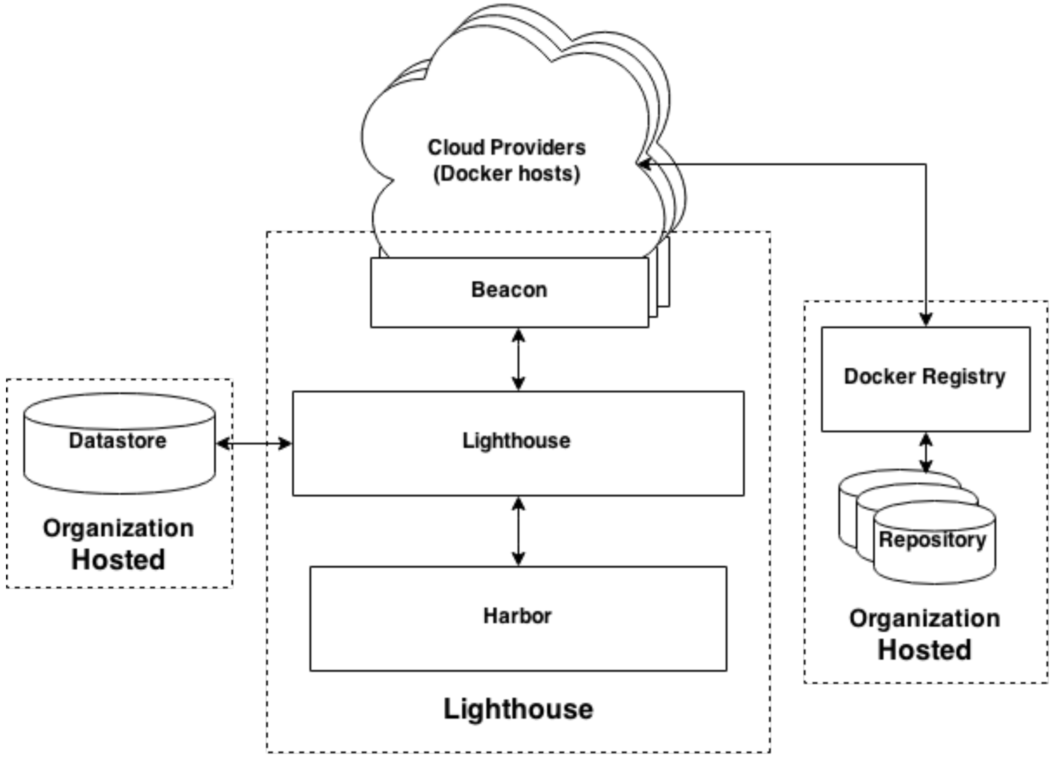
- Panamax - existed at conception
- Google Container Engine - November 4
- Rocket - December 1
- Docker Swarm - December 4

### How Lighthouse sets itself apart

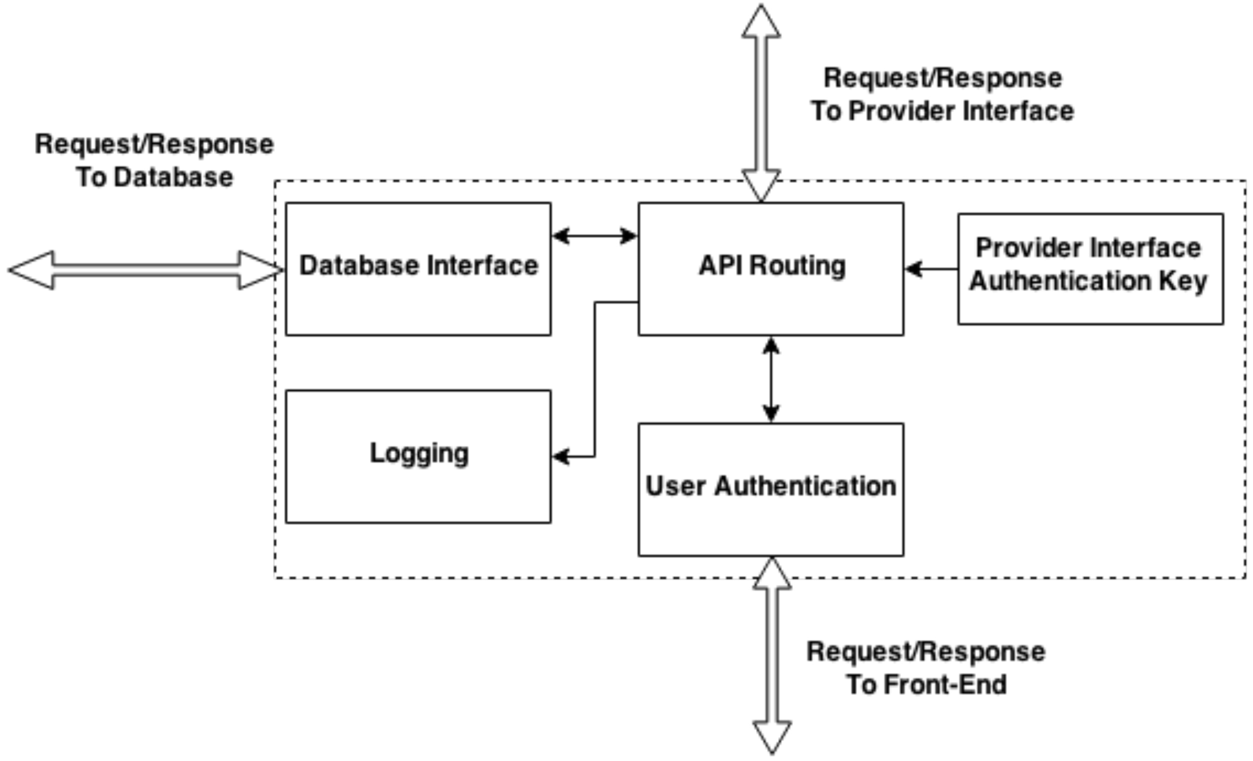
- More user control
- Enterprise-ready
- Self-hosted

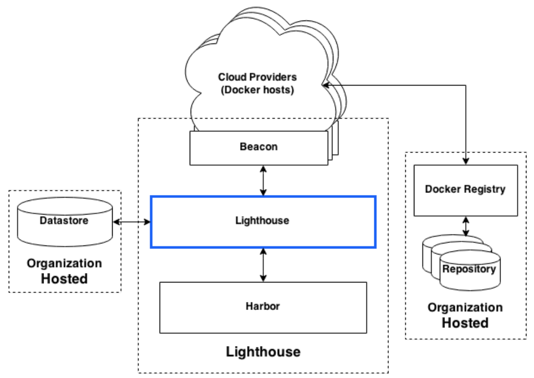
## Design

## Design - System Diagram

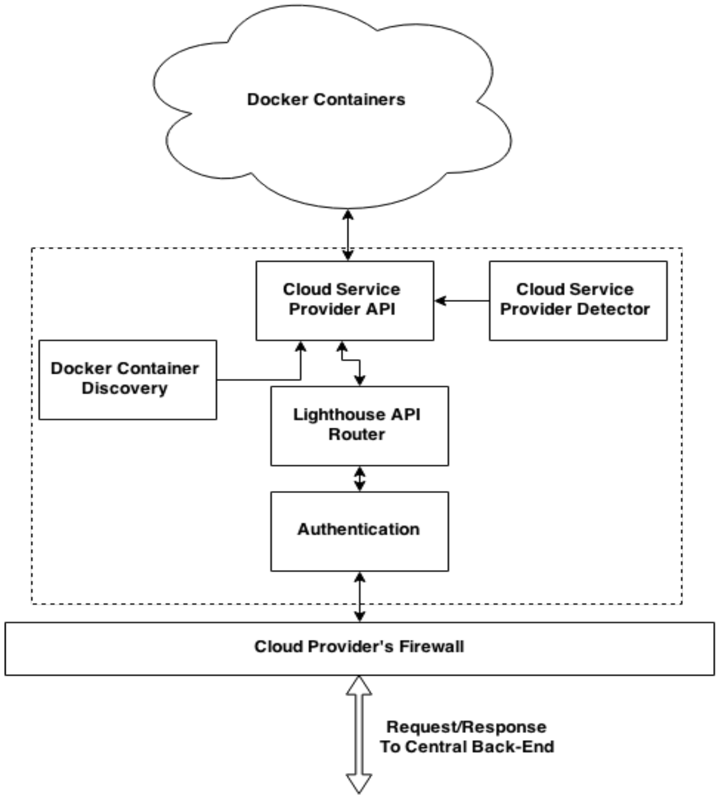


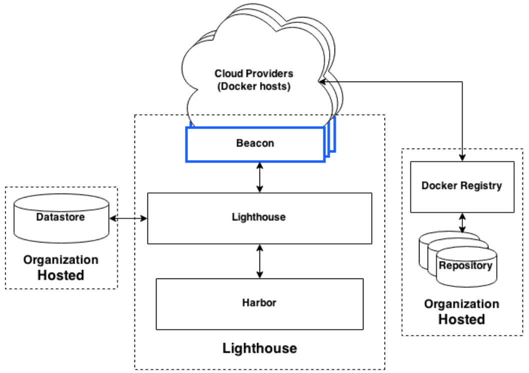
# Design - Lighthouse


Request/Response To Provider Interface
Request/Response To Database
Database Interface
API Routing
Provider Interface Authentication Key
Logging
User Authentication
Request/Response To Front-End



# Design - Beacon


Docker Containers
Cloud Service Provider API
Cloud Service Provider Detector
Docker Container Discovery
Lighthouse API Router
Authentication
Cloud Provider's Firewall
Request/Response To Central Back-End



## Technical Issue - Why Beacons Exist

### Situation

- As a member of IT/DevOps I want to quickly and efficiently expose my cluster of Docker VMs

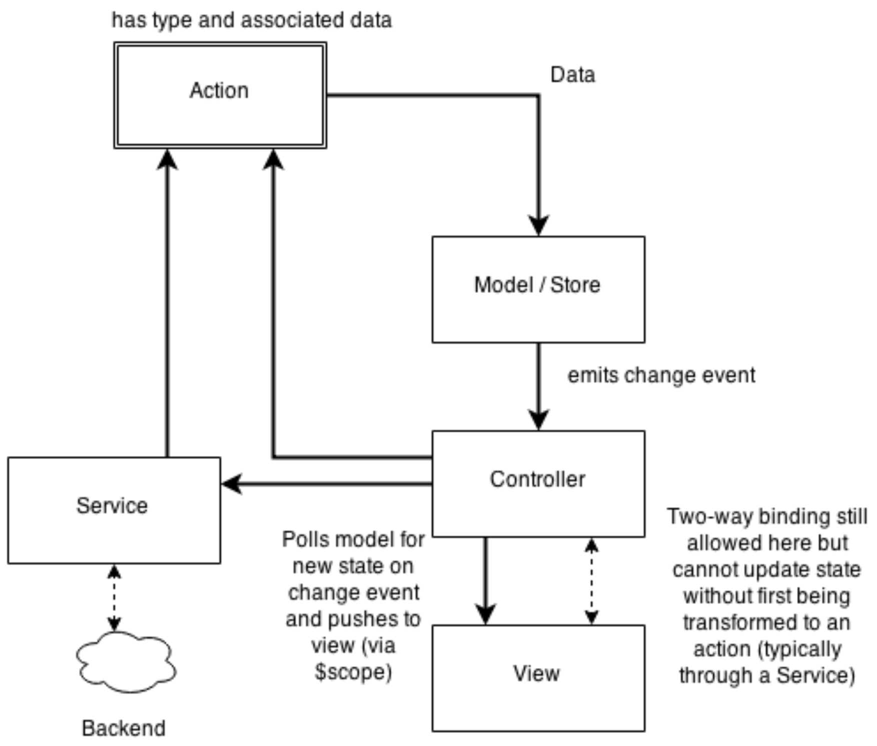
### Challenge

- Need a tool to discover VMs inside a cloud environment with little/no prior knowledge of Provider

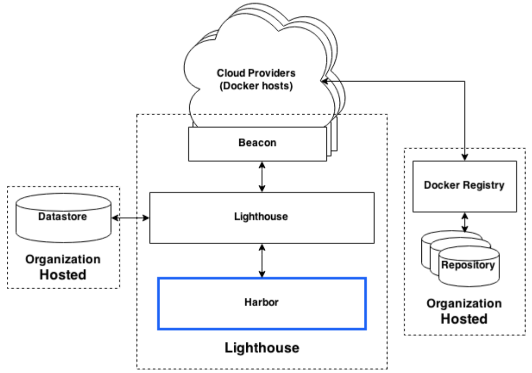
### Solution

- Create software “drivers” for detecting cloud environments
  - Probe Providers for existing/owned VMs with Docker
  - Establish basic API for Lighthouse
  - Create pluggable interface for future cloud providers

# Design - Harbor



has type and associated data
Action
Data
Model / Store
emits change event
Controller
Service
Polls model for new state on change event and pushes to view (via $scope)
Two-way binding still allowed here but cannot update state without first being transformed to an action (typically through a Service)
View
Backend



# Technical Issue - Authentication

## Situation

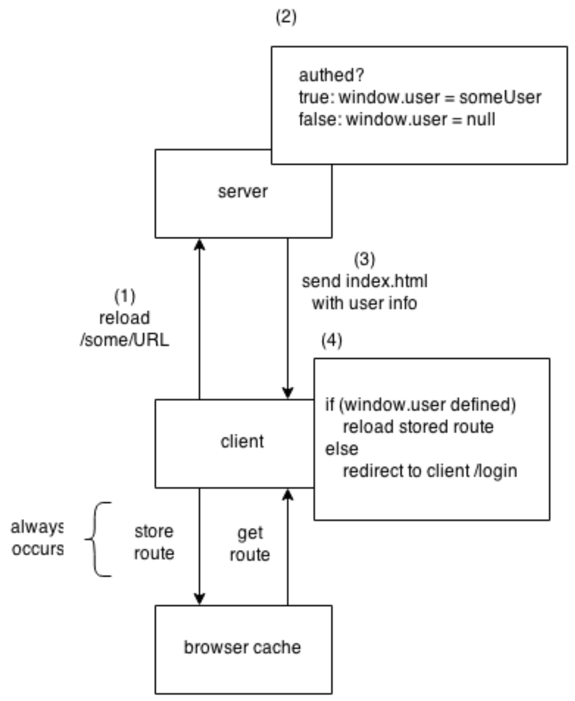
- As a user of Harbor I need to refresh my page or restart Lighthouse.

## Challenge

- Harbor is a single-page application and uses client-side template rendering
  - Causes a problem with routing and authentication

## Solution

- Generate a notification on the server to automatically inform the client of its auth status



## Technical Issue - Streaming

### Situation

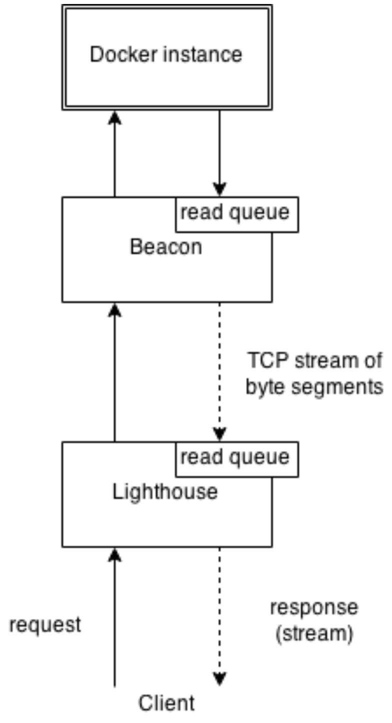
- As a user of Harbor, I want to be able to view stats, progress, and logs without manually refreshing.

### Challenge:

- HTTP responses can be streamed, but there are 3 hops from Docker to Harbor which requires a lot of coordination

### Solution:

- Stream all Docker responses from Beacon to Lighthouse and from Lighthouse to Harbor



## Technical Issue - Application Streams

### Situation

- As a release manager creating or updating a large application I want real time updates of the deployment status and concise errors.

### Challenge

- Deployments perform many operations on potentially hundreds of instances
  - Need to consistent way to report statuses and operation updates

### Solution

- A stream of well-defined status update objects
  - Operation updates wrap statuses of individual instances
  - Instances report successes, failures, warnings, etc.

## Testing and Quality Process

# Test-Driven Development in GitHub

GitHub tracks commits and discussion

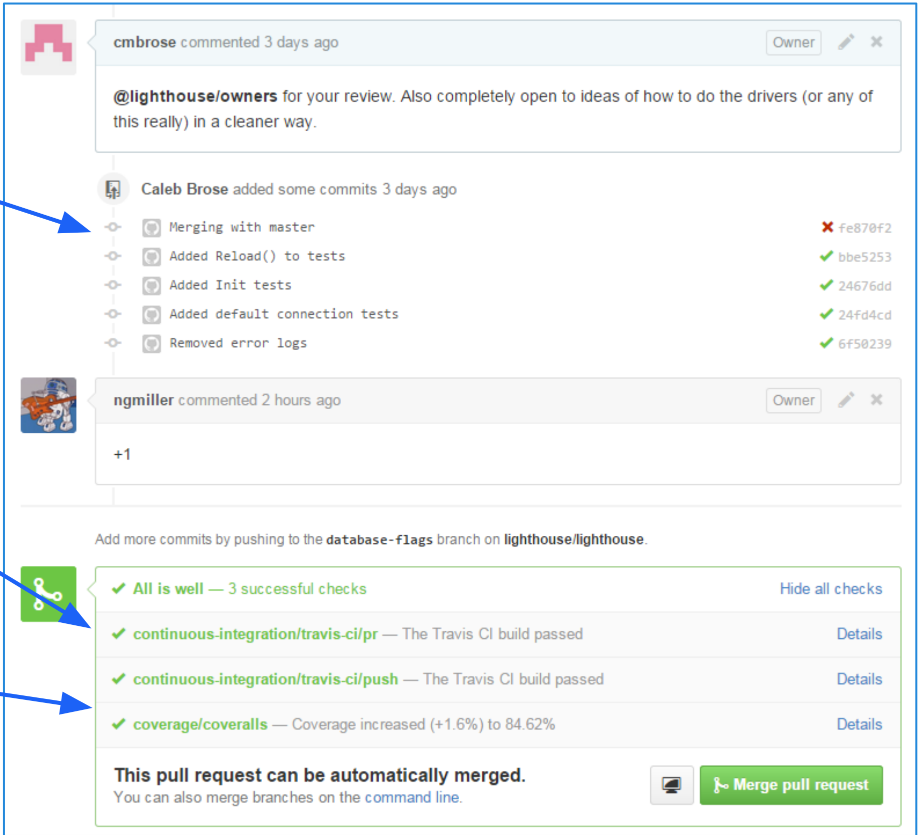
- All pull requests go through code review

Travis-CI runs unit tests

- Build fails if any tests fail

Coveralls reports code coverage

- Build fails if coverage is too low



cmbrose commented 3 days ago Owner

@lighthouse/owners for your review. Also completely open to ideas of how to do the drivers (or any of this really) in a cleaner way.

Caleb Brose added some commits 3 days ago

```
Merging with master               ✗ fe870f2
Added Reload() to tests           ✓ bbe5253
Added Init tests                  ✓ 24676dd
Added default connection tests    ✓ 24fd4cd
Removed error logs                ✓ 6f50239
```

ngmiller commented 2 hours ago Owner

+1

Add more commits by pushing to the `database-flags` branch on **lighthouse/lighthouse**.

✓ All is well — 3 successful checks — Hide all checks

✓ continuous-integration/travis-ci/pr — The Travis CI build passed — Details

✓ continuous-integration/travis-ci/push — The Travis CI build passed — Details

✓ coverage/coveralls — Coverage increased (+1.6%) to 84.62% — Details

**This pull request can be automatically merged.**
You can also merge branches on the command line.

# Testing Environment

## Lighthouse, Beacon

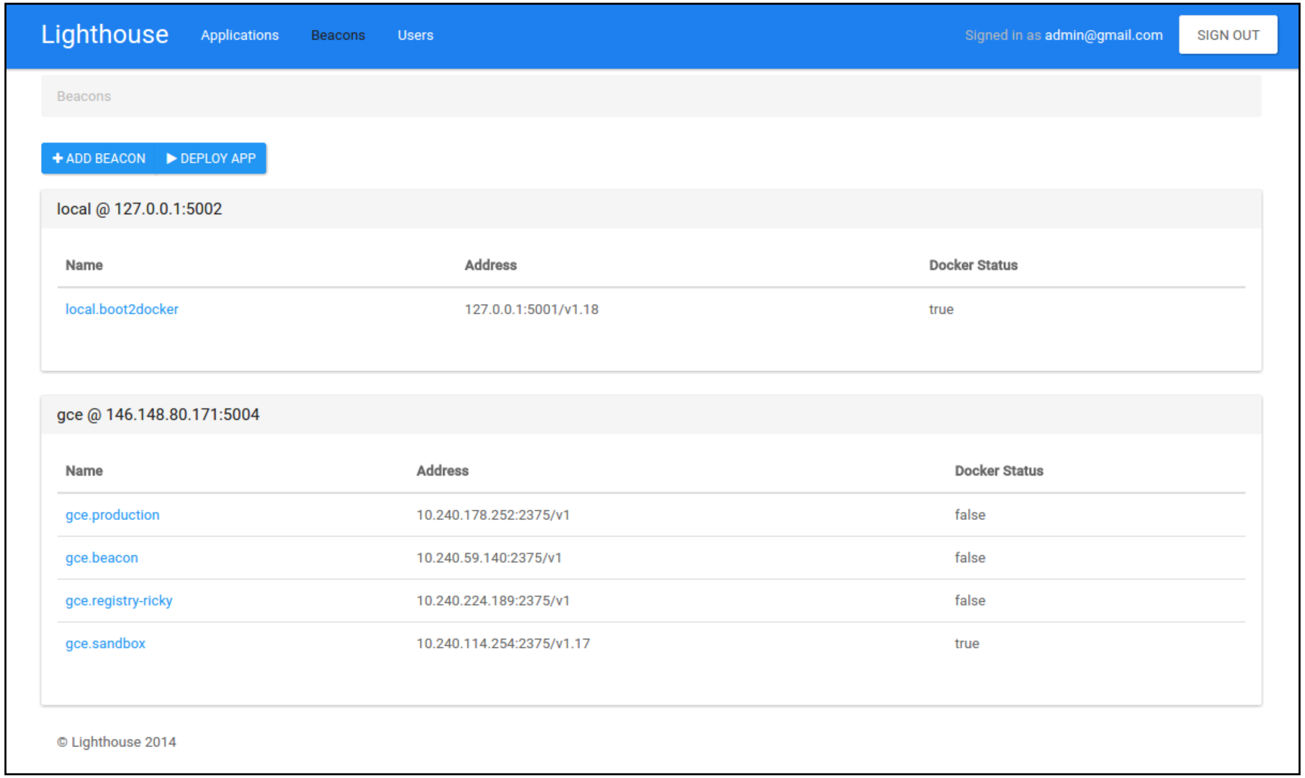
- Golang packages
  - testing, testify
- Go fmt

## Harbor

- PhantomJS “headless” browser
- Jasmine behavioral testing framework
- JSHint

## Thank you

## Beacon Management



local @ 127.0.0.1:5002

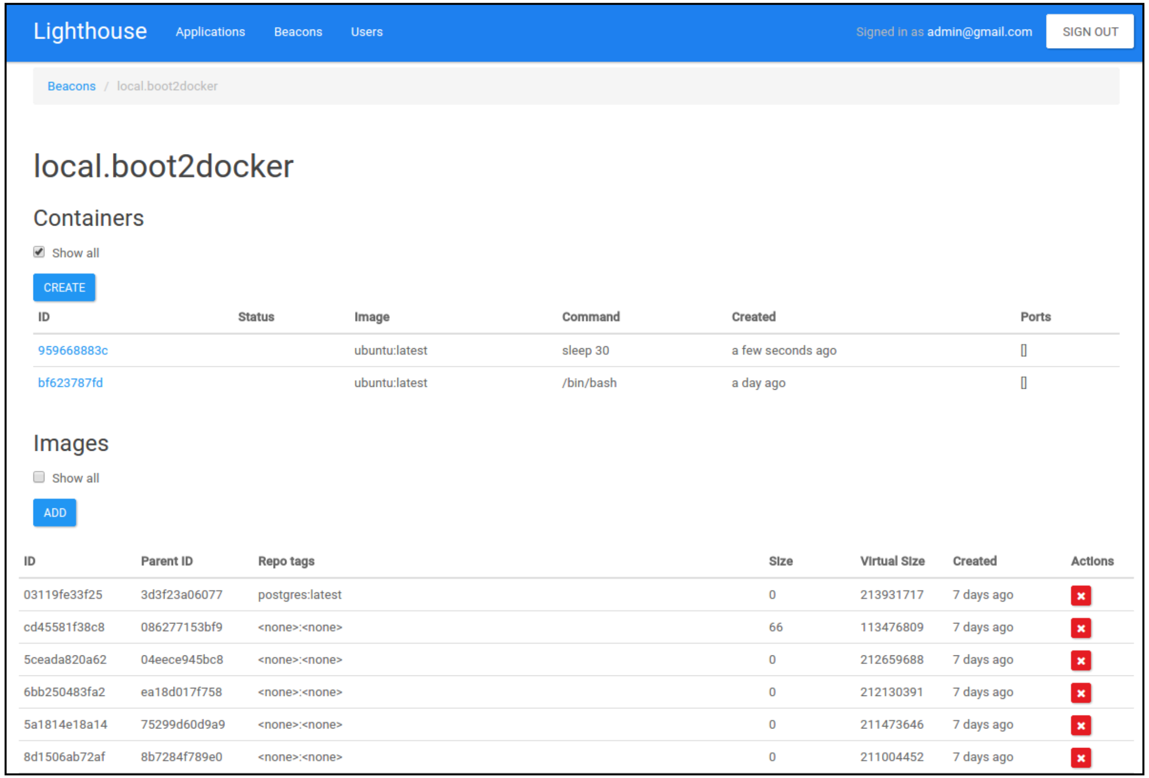
| Name | Address | Docker Status |
|---|---|---|
| local.boot2docker | 127.0.0.1:5001/v1.18 | true |

gce @ 146.148.80.171:5004

| Name | Address | Docker Status |
|---|---|---|
| gce.production | 10.240.178.252:2375/v1 | false |
| gce.beacon | 10.240.59.140:2375/v1 | false |
| gce.registry-ricky | 10.240.224.189:2375/v1 | false |
| gce.sandbox | 10.240.114.254:2375/v1.17 | true |

© Lighthouse 2014

# Instance Management



local.boot2docker

Containers

[✓] Show all

CREATE

| ID | Status | Image | Command | Created | Ports |
|---|---|---|---|---|---|
| 959668883c | | ubuntu:latest | sleep 30 | a few seconds ago | [] |
| bf623787fd | | ubuntu:latest | /bin/bash | a day ago | [] |

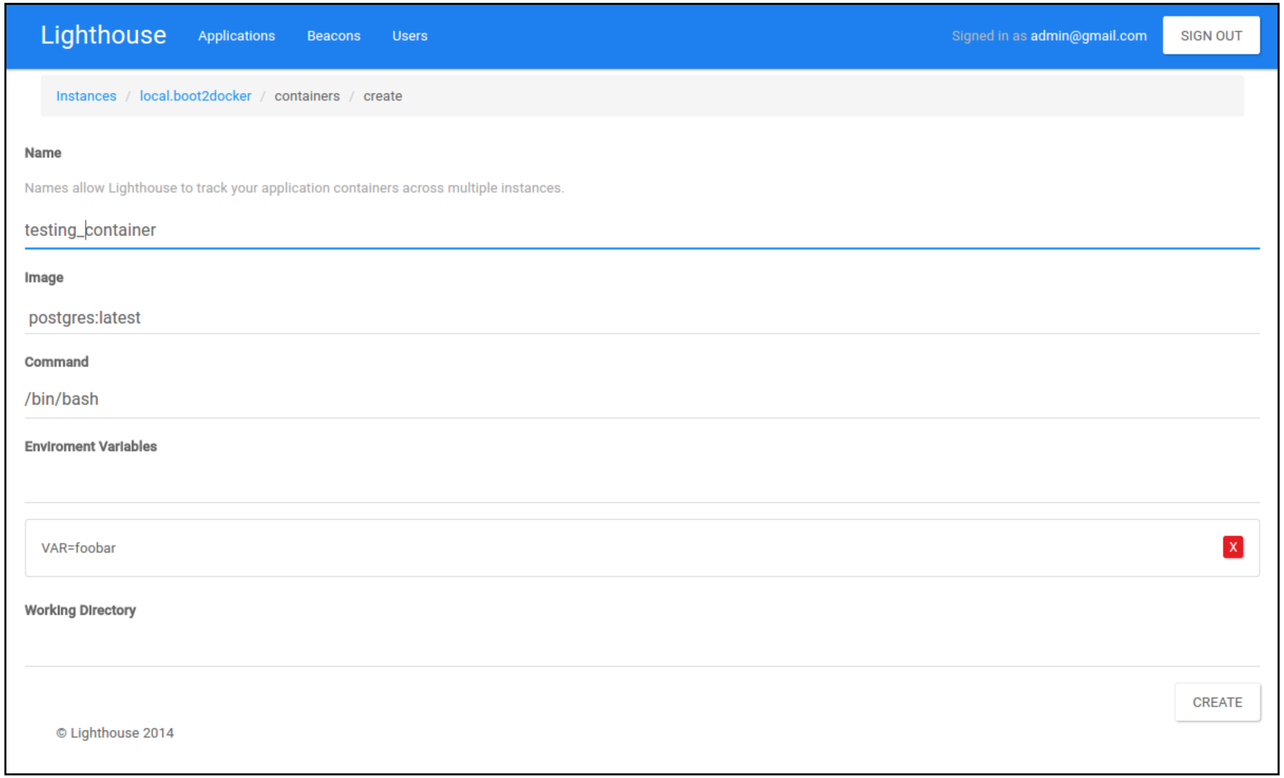
Images

[ ] Show all

ADD

| ID | Parent ID | Repo tags | Size | Virtual Size | Created | Actions |
|---|---|---|---|---|---|---|
| 03119fe33f25 | 3d3f23a06077 | postgres:latest | 0 | 213931717 | 7 days ago | ✖ |
| cd45581f38c8 | 086277153bf9 | <none>:<none> | 66 | 113476809 | 7 days ago | ✖ |
| 5ceada820a62 | 04eece945bc8 | <none>:<none> | 0 | 212659688 | 7 days ago | ✖ |
| 6bb250483fa2 | ea18d017f758 | <none>:<none> | 0 | 212130391 | 7 days ago | ✖ |
| 5a1814e18a14 | 75299d60d9a9 | <none>:<none> | 0 | 211473646 | 7 days ago | ✖ |
| 8d1506ab72af | 8b7284f789e0 | <none>:<none> | 0 | 211004452 | 7 days ago | ✖ |

# Container Creation



Lighthouse
Applications
Beacons
Users
Signed in as admin@gmail.com
SIGN OUT
Instances / local.boot2docker / containers / create

Name

Names allow Lighthouse to track your application containers across multiple instances.

testing_container

Image

postgres:latest

Command

/bin/bash

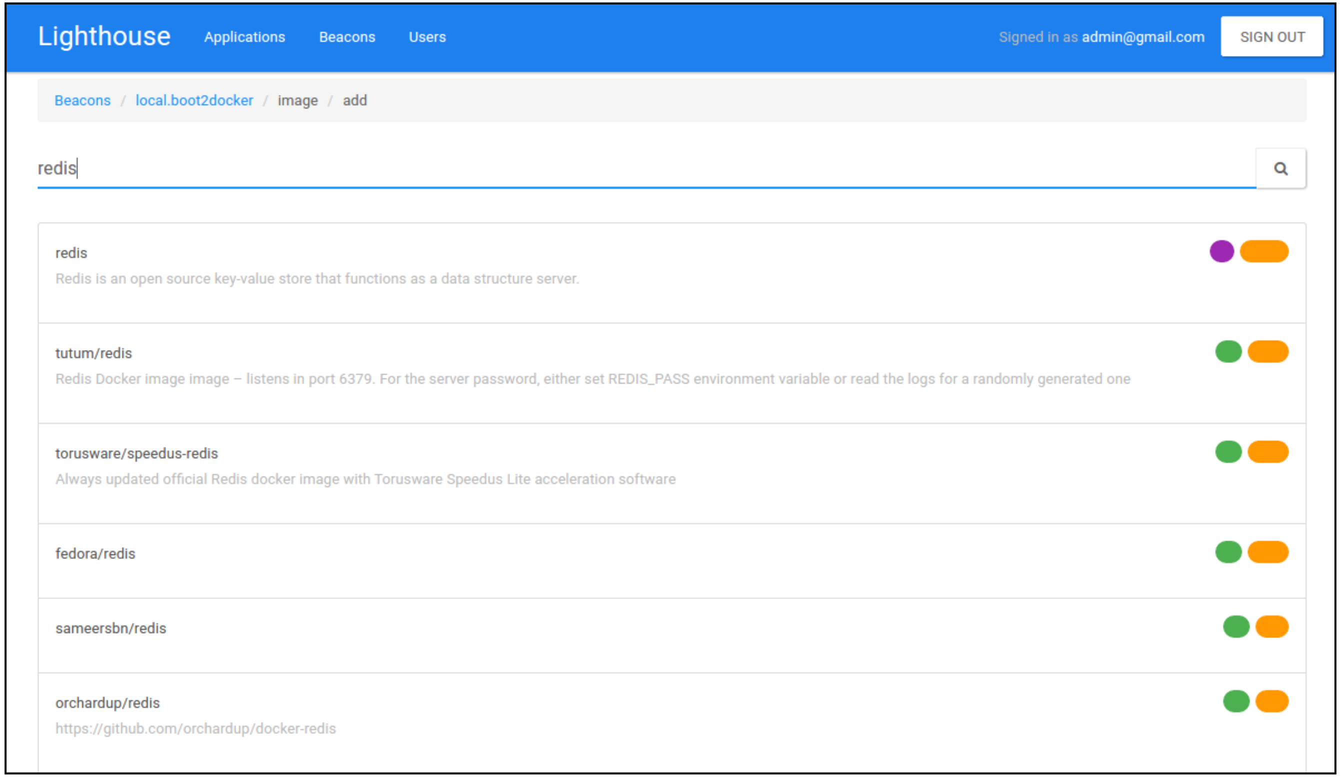
Enviroment Variables

VAR=foobar

Working Directory

X
CREATE
© Lighthouse 2014

# Pulling Images


Lighthouse
Applications
Beacons
Users
Signed in as admin@gmail.com
SIGN OUT
Beacons / local.boot2docker / image / add

redis

redis
Redis is an open source key-value store that functions as a data structure server.

tutum/redis
Redis Docker image image – listens in port 6379. For the server password, either set REDIS_PASS environment variable or read the logs for a randomly generated one

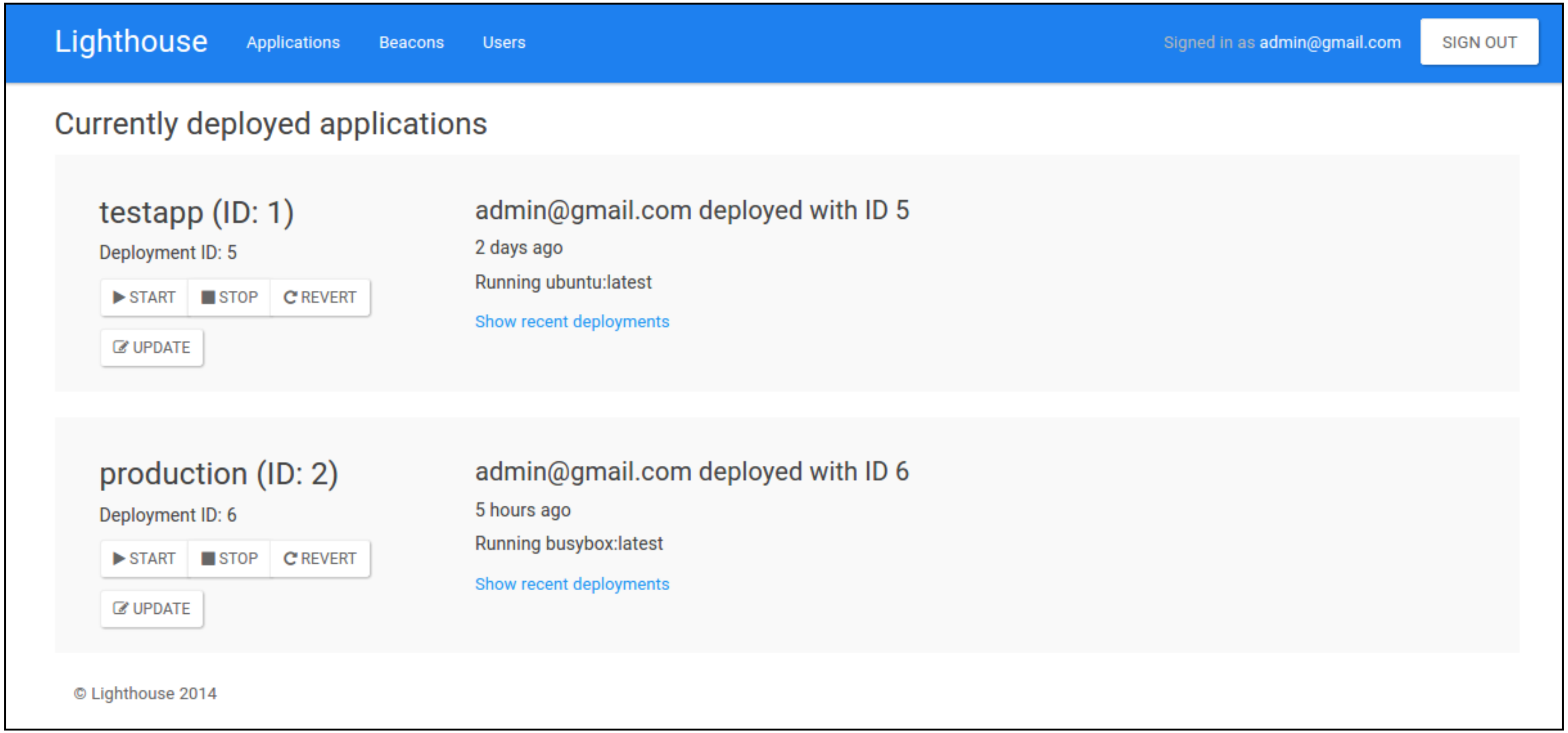
torusware/speedus-redis
Always updated official Redis docker image with Torusware Speedus Lite acceleration software

fedora/redis

sameersbn/redis

orchardup/redis
https://github.com/orchardup/docker-redis

# Application Management



Currently deployed applications

testapp (ID: 1)

Deployment ID: 5

admin@gmail.com deployed with ID 5

2 days ago

Running ubuntu:latest

Show recent deployments

production (ID: 2)

Deployment ID: 6

admin@gmail.com deployed with ID 6

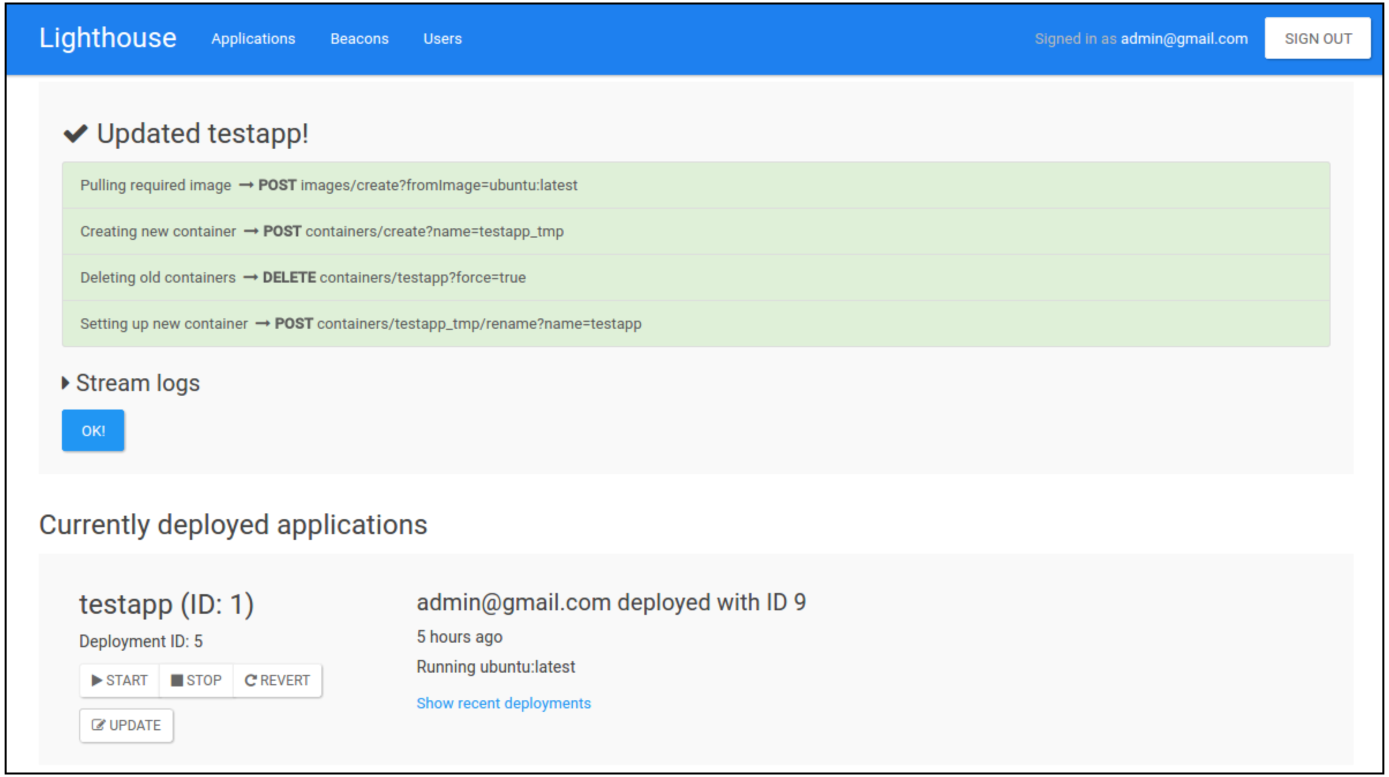
5 hours ago

Running busybox:latest

Show recent deployments

© Lighthouse 2014

# Deployment Updates



✔ Updated testapp!

Pulling required image ⟶ **POST** images/create?fromImage=ubuntu:latest

Creating new container ⟶ **POST** containers/create?name=testapp_tmp

Deleting old containers ⟶ **DELETE** containers/testapp?force=true

Setting up new container ⟶ **POST** containers/testapp_tmp/rename?name=testapp

▸ Stream logs

Currently deployed applications

testapp (ID: 1)

Deployment ID: 5

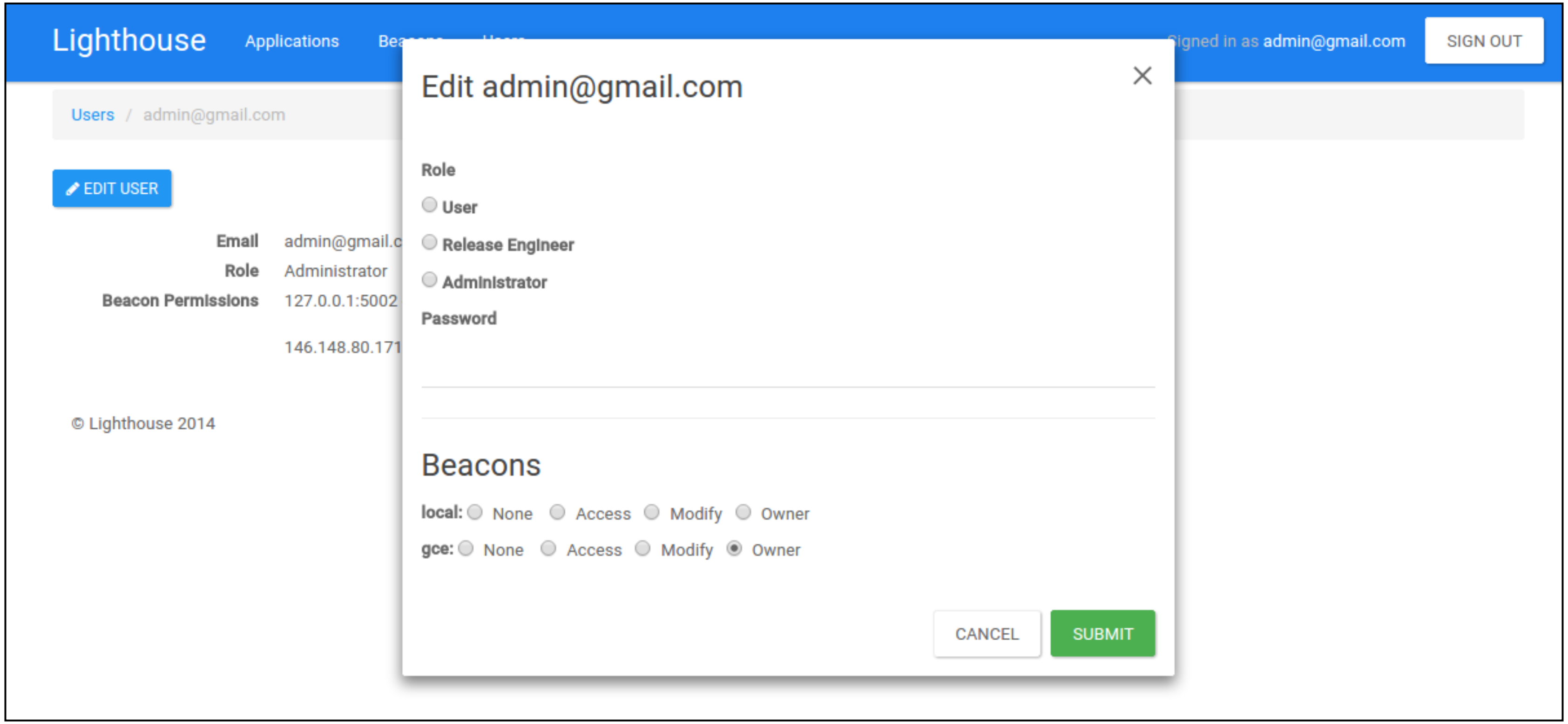
admin@gmail.com deployed with ID 9

5 hours ago

Running ubuntu:latest

Show recent deployments

# User Management



| | |
|---|---|
| Email | admin@gmail.c |
| Role | Administrator |
| Beacon Permissions | 127.0.0.1:5002 |
| | 146.148.80.171 |

© Lighthouse 2014

Edit admin@gmail.com

Role

User

Release Engineer

Administrator

Password

Beacons

local: None Access Modify Owner

gce: None Access Modify [✓ Owner]

## Complex Usage

```
    > docker run -ti debian:jessie /bin/bash
root@:12345/# apt-get update && apt-get install python
root@:12345/# echo “while True: print ‘foo’” > test.py
    > docker commit 12345 foo
    > docker run -d --name finn foo python test.py
    > docker logs finn
      /* a whole lot of foo */
    > docker kill finn
    > docker push foo
```

## Technical Issue - Testing

### Situation

- I’m a user of Lighthouse who runs important applications with Docker which use databases shared with Lighthouse.

### Challenge

- Need to be confident Lighthouse works precisely as expected
  - Testing code that needs external services is difficult

### Solution

- Services like external servers or databases can be mocked
  - Go has packages specifically for mocking servers
  - Databases have varying levels of abstraction

## Milestones - Fall

August

- Start of project

September

- Created the Lighthouse organization on Github

October

- Proof of concept presented to Workiva

November

- Basic Docker functionality

## Milestones - Spring

January

- Finalized architectures

February

- Authentication, Beacon integration, Container control

March

- Streaming, Container creation

April

- Application management, user management, container logs

## Desired Additions

Support for multiple database drivers

Full-stack HTTPS support

Real-time network and resource usage