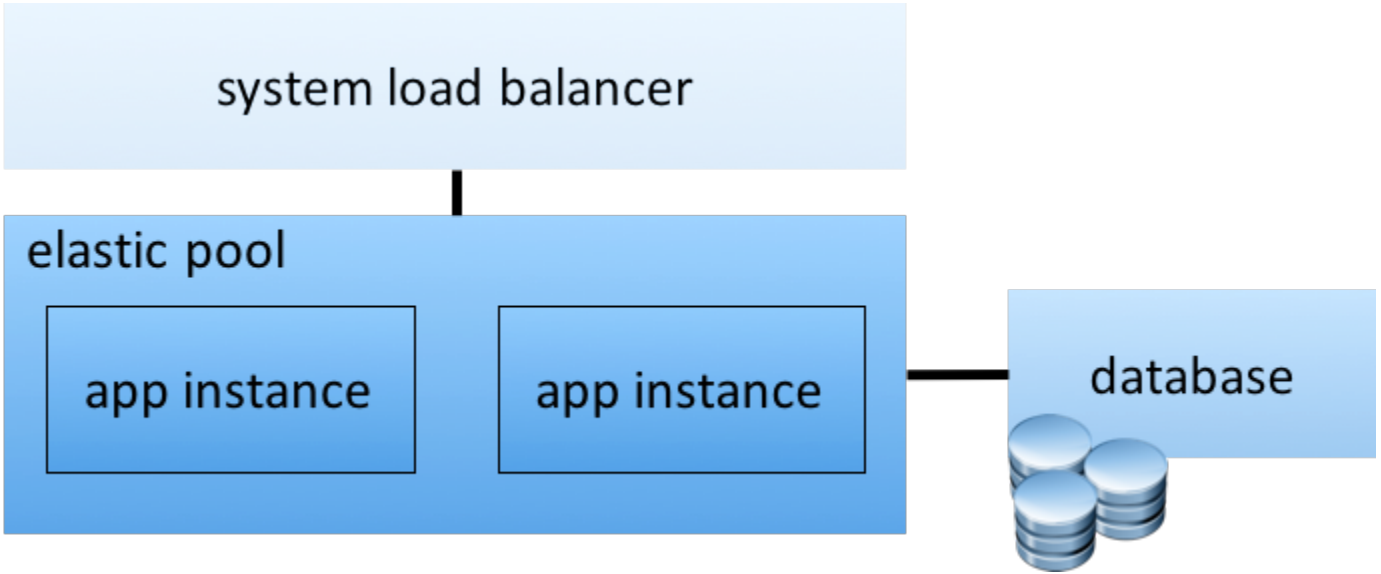
Deploy Your First App on Cloud Foundry

Hands-on Workshop

Objectives

Gaining first hand experience with the workflow of deploying applications into a Cloud Foundry cluster

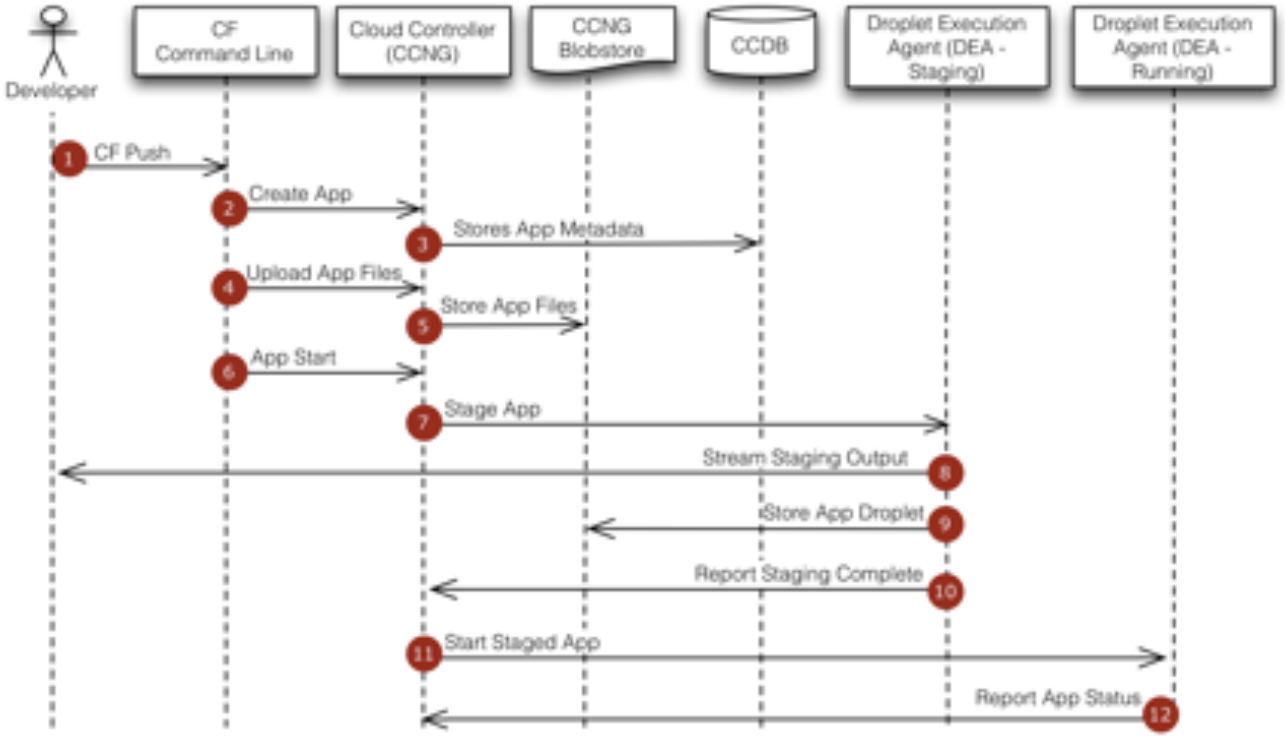
Typical Web Application



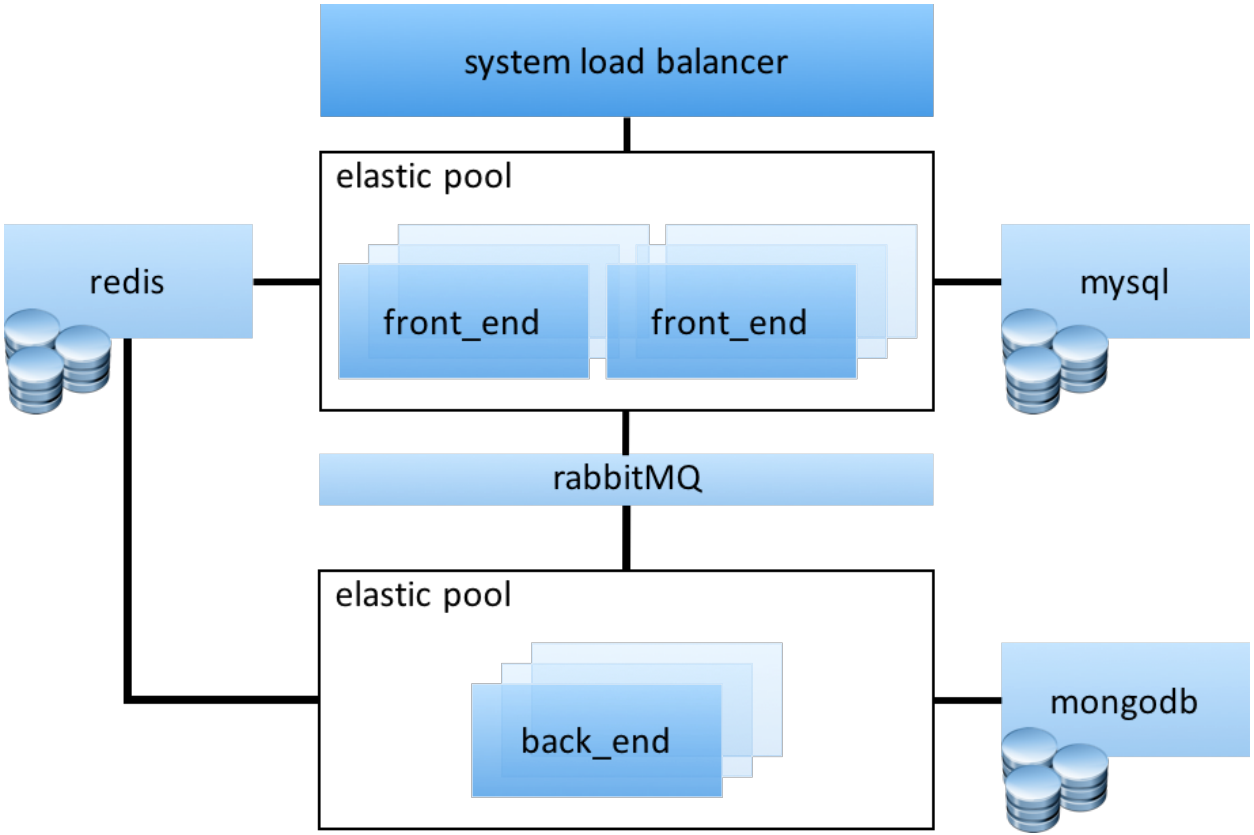
Typical Workflow deploying

```
# to target and login to cloud foundry
helion target https://hcfXXX.helion-dev.com
helion login

# to create and boot the app for the first time
helion push myapp --instances 2 --mem 64M -path ../code

# to create the database and bind it to the app
helion create-service mysql --name mydb --bind myapp

# update live app with new code
helion update myapp --path ../code
```

Cloud Foundry App Deployment



More Complicated



More Complicated, More of the Same!

```
# create the front end and backend apps
# front end is small but multi-instance
helion push fe --instances 8 --mem 64M --path ../fe_code
helion push be --instances 2 --mem 256M --path ../be_code
# create the services and bind per spec
helion create-service mysql --name mysql --bind fe
helion create-service mongodb --name mongo --bind be
helion create-service rabbit --name rabbit --bind fe
helion create-service redis --name redis --bind fe
helion bind-service redis be
helion bind-service rabbit be
# to perform an update of code
helion update fe --path ../new_fe_code
helion update be --path ../new_be_code
```

Prerequisites

Laptop running Windows, Linux, MacOS

GitHub client installed

https://help.github.com/articles/set-up-git/

Helion Cloud Foundry

40x micro clusters, containing:

Controller, router, DEA, HM, MySQL, Redis

Running on single OpenStack VM in hpcloud.com

Naming convention:

https://hcfXXX.helion-dev.com

XXX == your cluster id, 3 digits [001-040]

Credentials:

UID == dev PWD == password1234!

Getting Ready

Install the helion CLI for your platform:

Overview: http://docs.hpcloud.com/helion/devplatform/1.1/als/client/download

Linux64: http://clients.als.hpcloud.com/helion-1.1.0-linux-glibc2.3-x86_64.zip
Linux32: http://clients.als.hpcloud.com/helion-1.1.0-linux-glibc2.3-ix86.zip
MacOS: http://clients.als.hpcloud.com/helion-1.1.0-macosx10.5-i386-x86_64.zip
Windows: http://clients.als.hpcloud.com/helion-1.1.0-win32-ix86.zip

Validating Your Setup

```
# set target to your cluster endpoint
helion target https://hcfXXX.helion-dev.com

# login using uid:dev pwd:password1234!
helion login

# list cluster info
helion info
# list existing applications
helion apps
```

Your first application

```
# create a working directory
mkdir ~/workshop
cd ~/workshop

# clone repo as starting point
git clone https://github.com/hcf-workshop/cf-node-app.git

# Very simple node.js application:
# server.js
# package.json
```

To deploy your first application...

```
# Create a manifest.yml file using your favorite editor
# manifest.yml content

---
# This is an extremely basic manifest file.
# Note that the name is always required
# while other fields are optional.
applications:
- name: cf-node-app
  mem: 128M
```

To deploy your first application...

```
# cd ~/workshop/cf-node-app

# -n is no questions asked :)
helion push -n

# when done, open your application
https://cf-node-app.hcfXXX.helion-dev.com

# check the portal and look at the log stream and files
https://hcfXXX.helion-dev.com
```

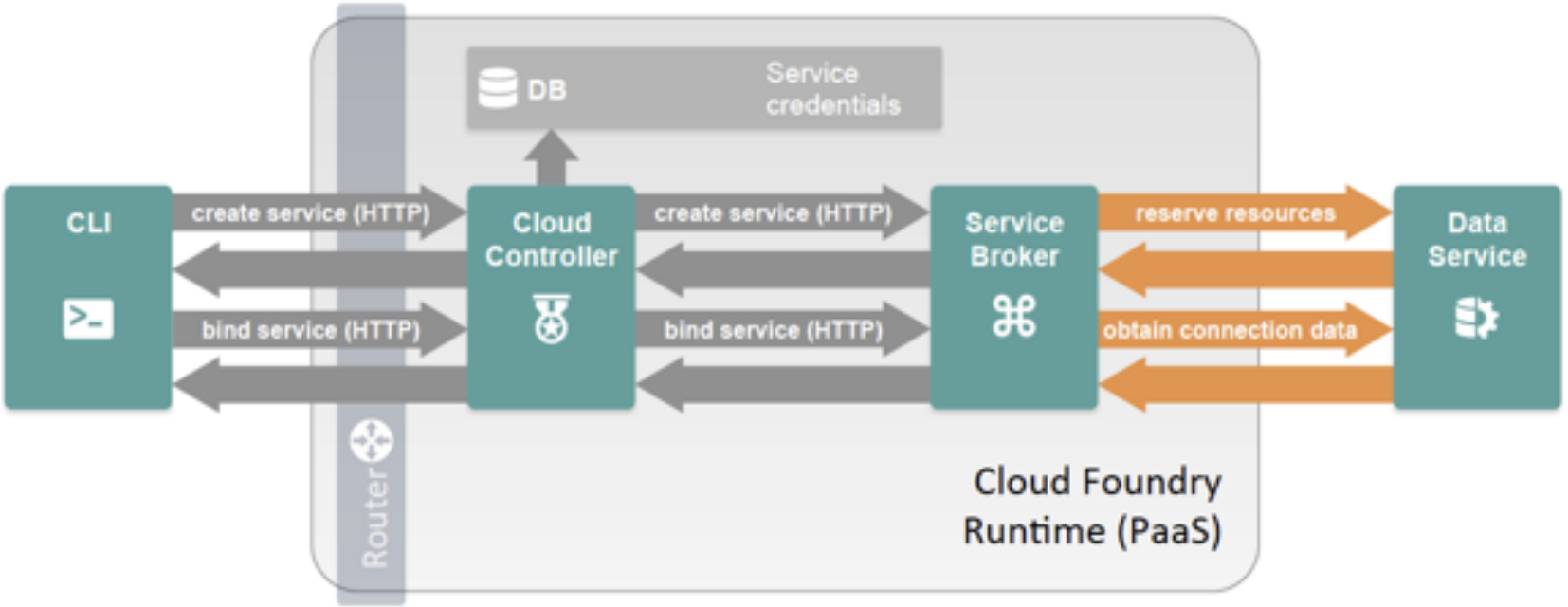
Ready for your second app?

This application will bind to a MySQL database

To get started:

```
cd ~/workshop
git clone https://github.com/hcf-workshop/cf-node-mysql-app.git
cd ~/workshop/cf-node-mysql-app
```

Creating and Binding a Service



Deploy app, bind it with a service

```
# create manifest.yml file, first iteration
---
applications:
- name: cf-node-mysql-app
  mem: 128M

# deploy but do not start
helion push -n --no-start
```

How the app uses the database?

```
var services = process.env.VCAP_SERVICES;
services = JSON.parse(services);

var credentials = services.mysql[0].credentials;

var dbname = credentials.name;
var hostname = credentials.hostname;
var user = credentials.user;
var password = credentials.password;
var port = credentials.port;
```

Create & Bind the Service to the App

```
# list available services
helion services

# create the service
helion create-service mysql cf-node-mysql-app-db
# bind the the service to the application
helion bind-service cf-node-mysql-app-db cf-node-mysql-app

# start the application
helion start cf-node-mysql-app
```

Look at your app...

```
# dump the environment variables of the app
helion env cf-node-mysql-app

# dump the logs of your application from the command line
helion logs cf-node-mysql-app
```

Now do it again using a manifest

```
# delete app
helion delete cf-node-mysql-app
# update manifest.yml to be like this
---
applications:
- name: cf-node-mysql-app
  mem: 128M
  services:
    ${name}-db:
      type: mysql
```

Rinse and repeat...

```
# deploy again
cd ~/workshop/cf-node-mysql-app
helion push -n
```

If you have time...

Feel free to play around!

Suggestion:

https://github.com/cloudfoundry-samples/fib-cpu

```
# scale to 2 instances
helion scale <app-name> --instances 2
# list instances
helion instances <app-name>
```

Thank you!

I hope you had fun
and
learned something too!