Spring into the Cloud

Josh Long
@starbuxman

josh.long@springsource.com

Chris Richardson
@crichardson

chris.richardson@springsource.com

About this session

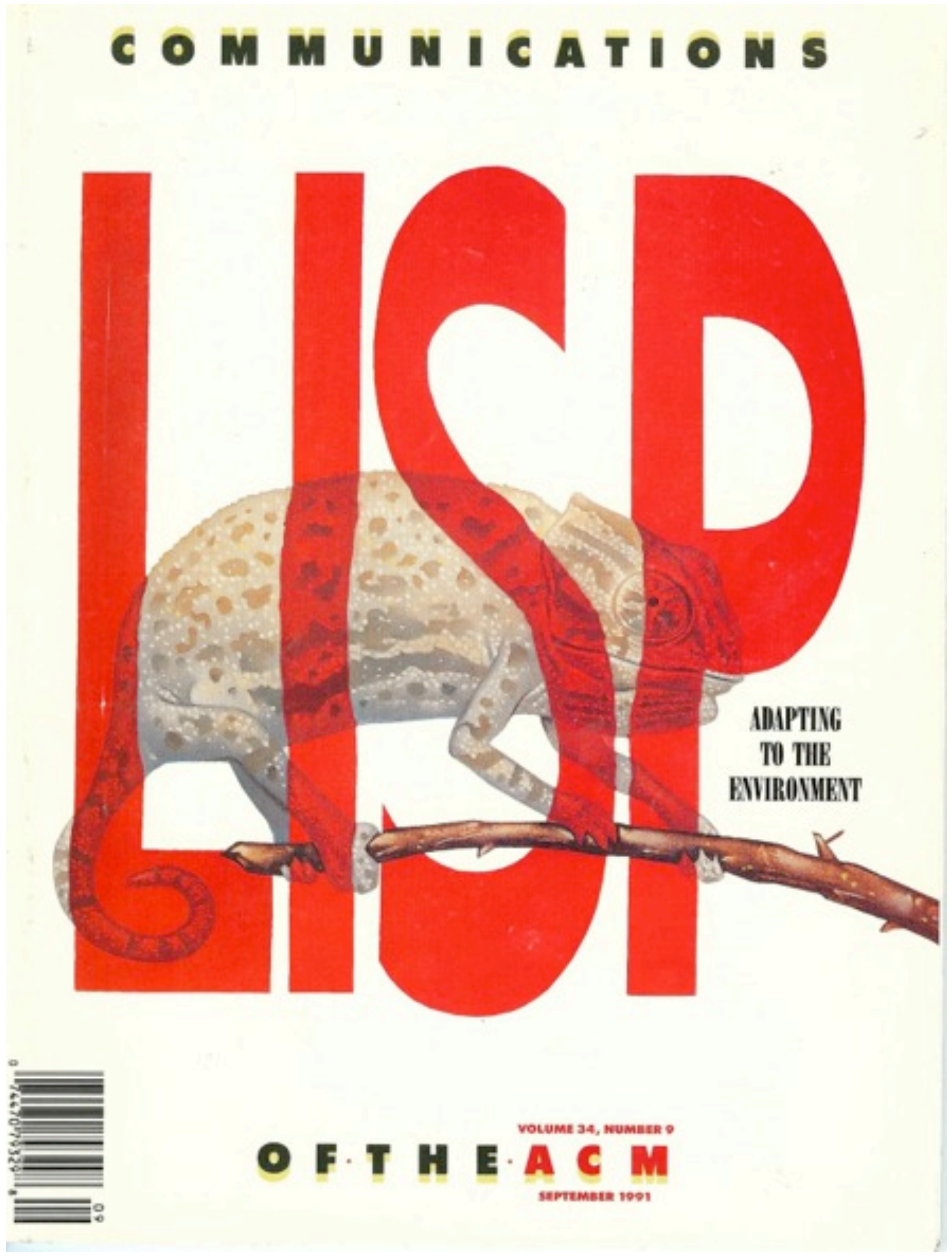
Spring and **Cloud Foundry**: a match made in heaven

About Chris

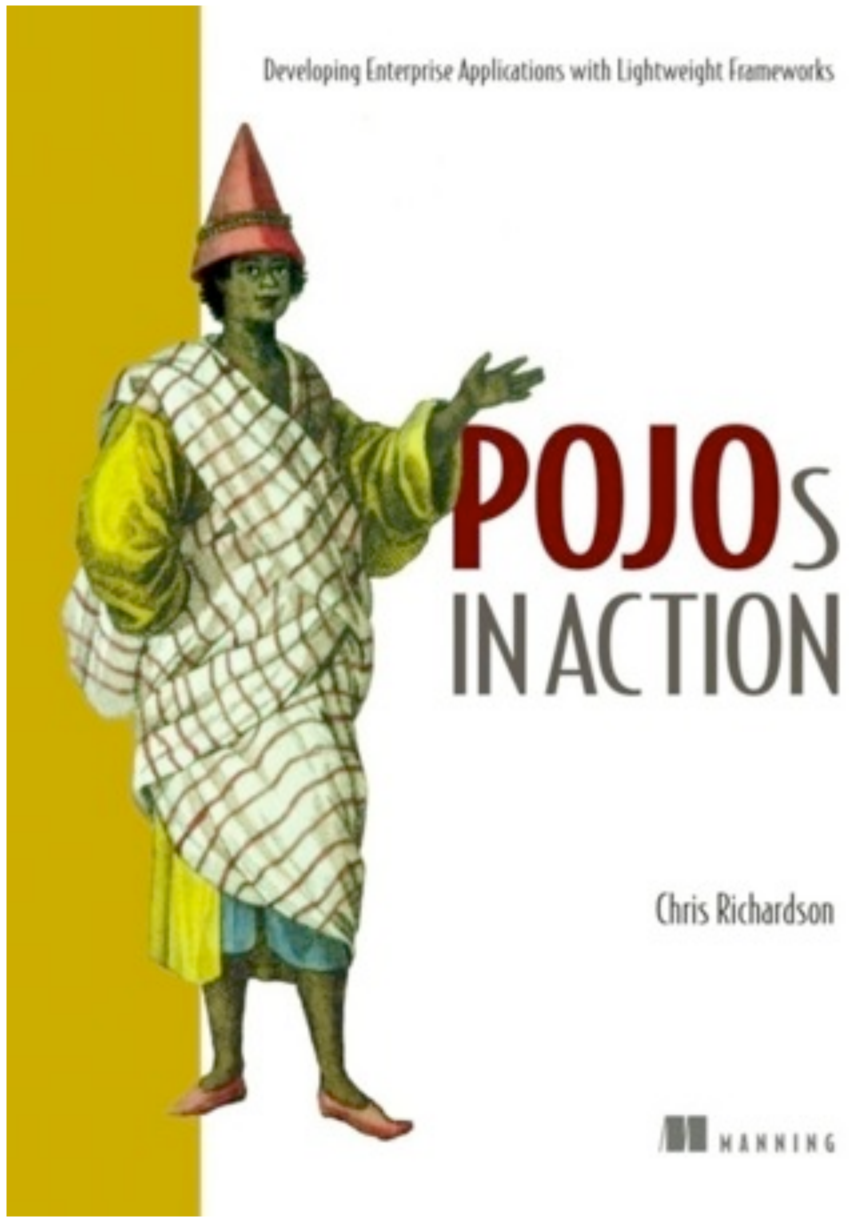
Aberdeen
Dundee
Glasgow
Londonderry / Derry
Belfast
United Kingdom
Hartlepool
Isle of Man
Leeds
Preston
Hull
Dublin
Ireland
Manchester
Leicester
Norwich
Birmingham
Ipswich
Northampton
Swansea
Basildon
Barnet
Cardiff
Bristol
Exeter
Portsmouth
Eastbourne
Plymouth
A

Medford
Klamath Falls
Yreka
Eureka
Redding
Elko
Chico
Reno
Nevada
Yuba City
Carson City
Santa Rosa
Sacramento
San Francisco
Stockton
San Jose
California
Fresno
Visalia
Las Vegas
Santa Maria
Bakersfield
Lompoc
Palmdale
Santa Clarita
Riverside
Los Angeles
Hemet
Oceanside
Yuma
Tijuana
Mexicali
Ensenada
Puerto Peñasco
A

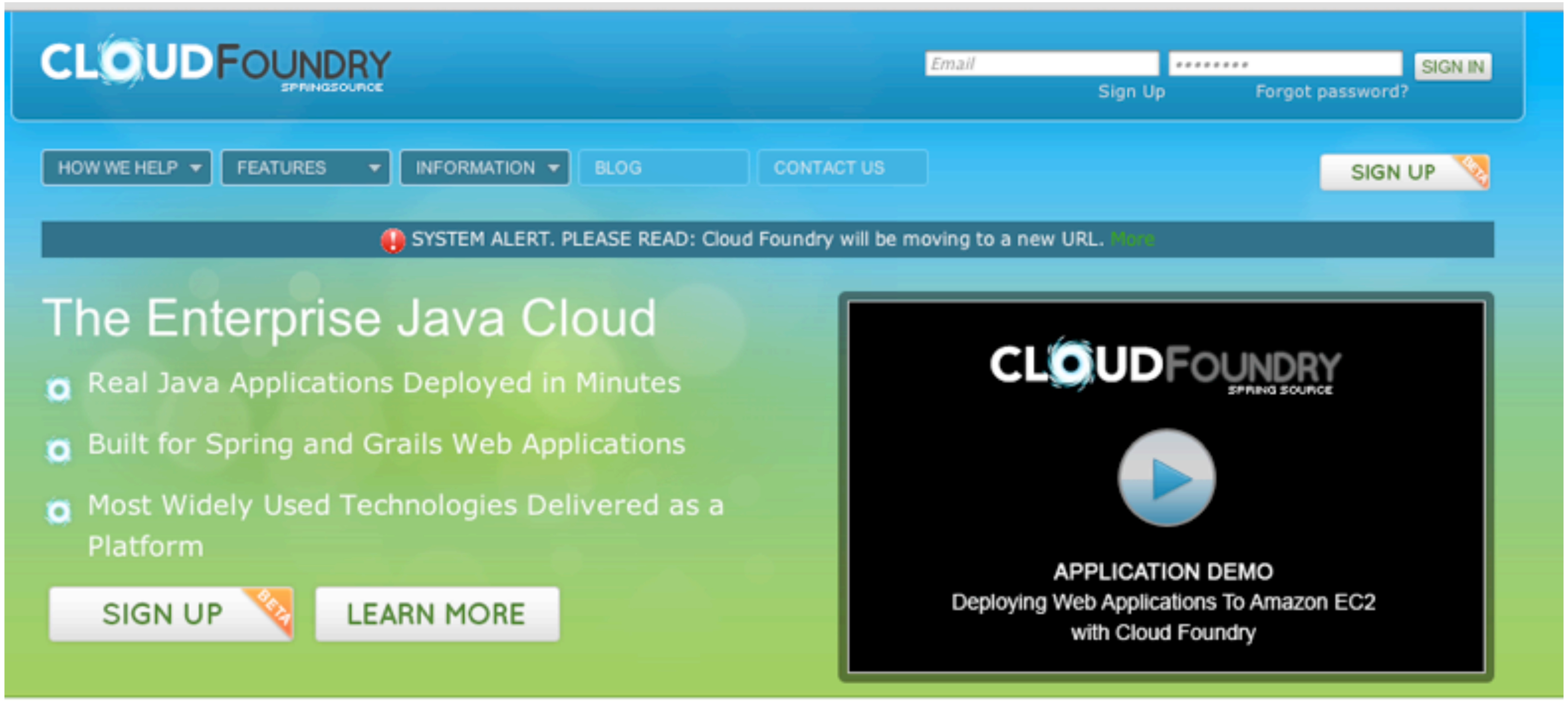
(About Chris)


COMMUNICATIONS
LISP
ADAPTING TO THE ENVIRONMENT
VOLUME 34, NUMBER 9
OF THE ACM
SEPTEMBER 1991

About Chris()


Developing Enterprise Applications with Lightweight Frameworks
POJOs IN ACTION
Chris Richardson
MANNING

About Chris



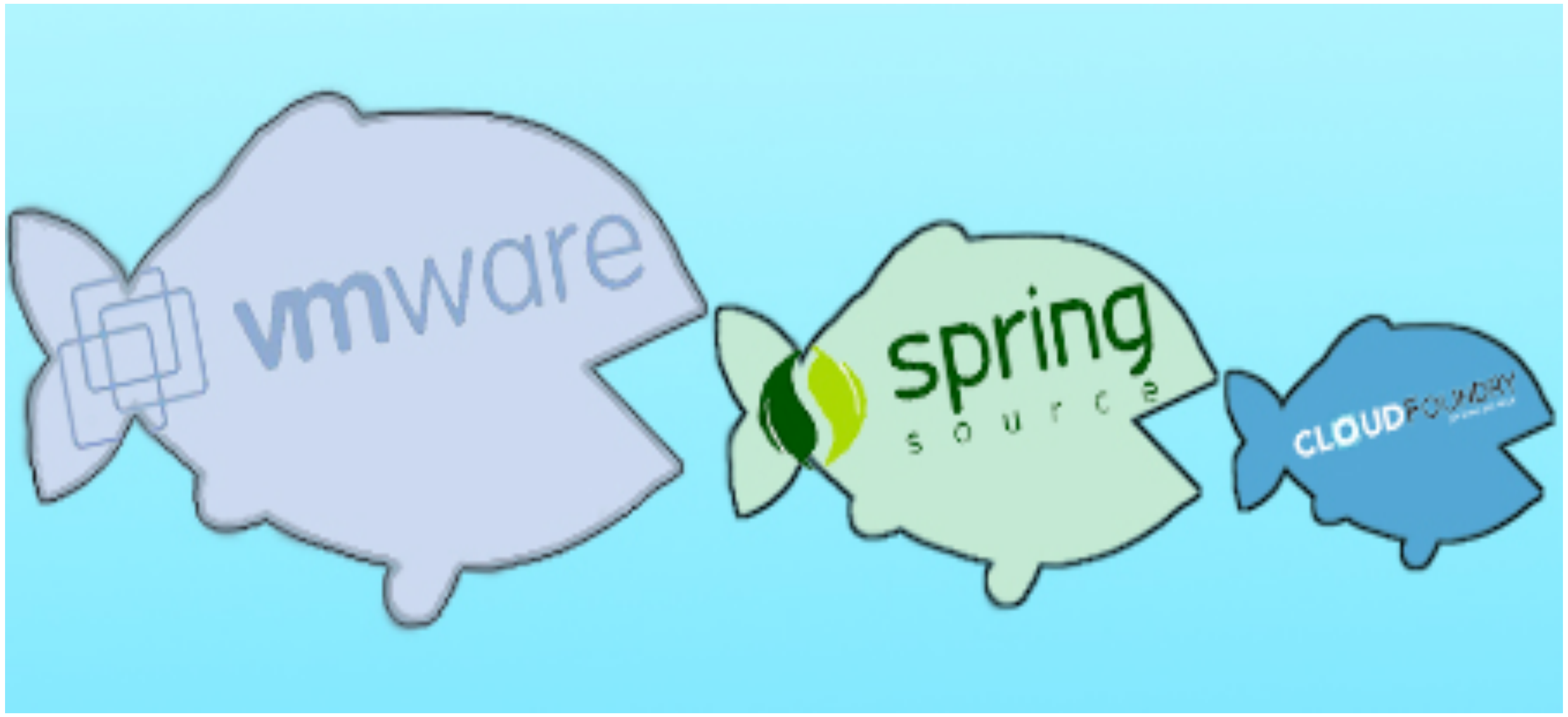
SYSTEM ALERT. PLEASE READ: Cloud Foundry will be moving to a new URL. More

The Enterprise Java Cloud

- Real Java Applications Deployed in Minutes
- Built for Spring and Grails Web Applications
- Most Widely Used Technologies Delivered as a Platform

APPLICATION DEMO
Deploying Web Applications To Amazon EC2 with Cloud Foundry

About Chris



http://www.theregister.co.uk/2009/08/19/springsource_cloud_foundry/

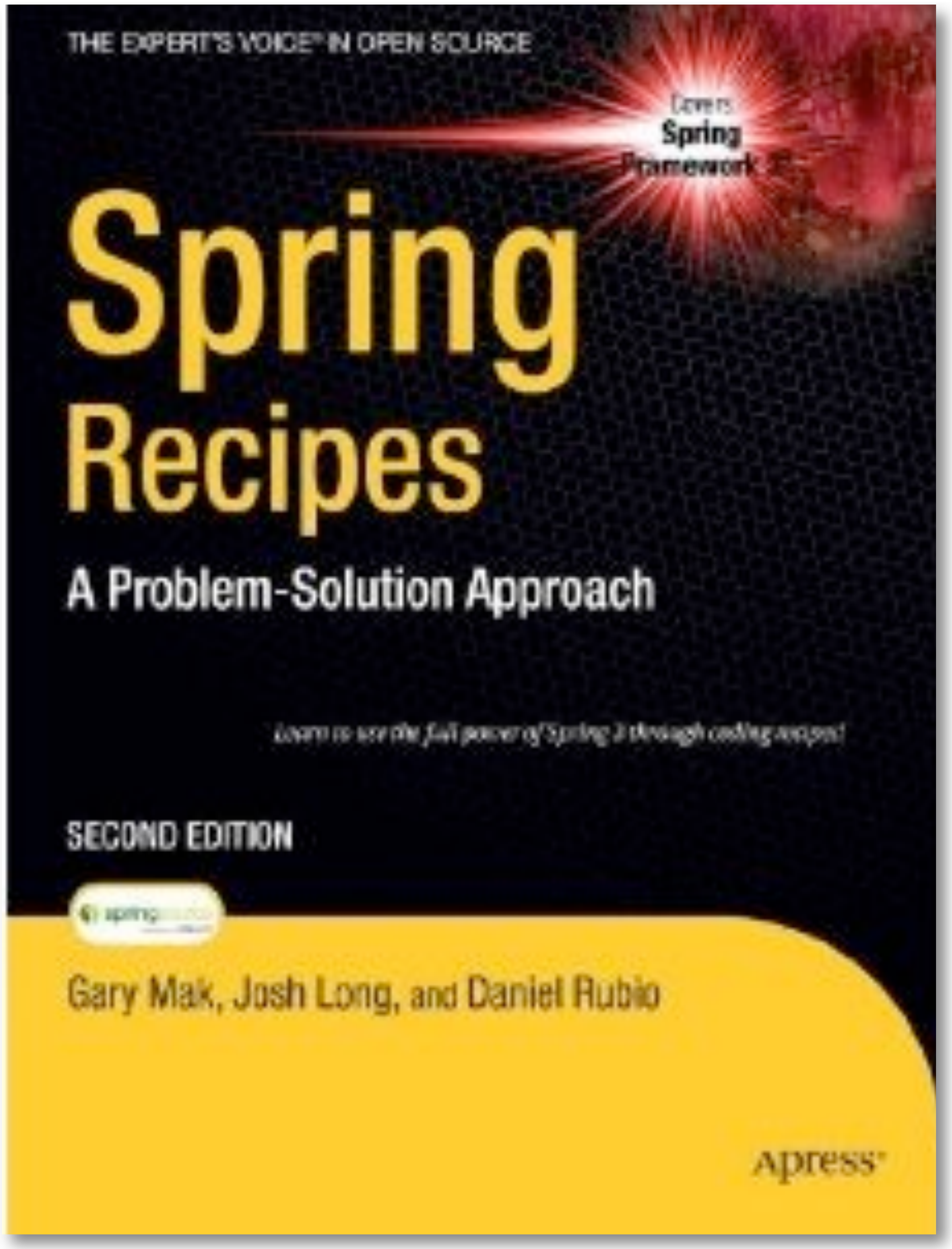
About Chris

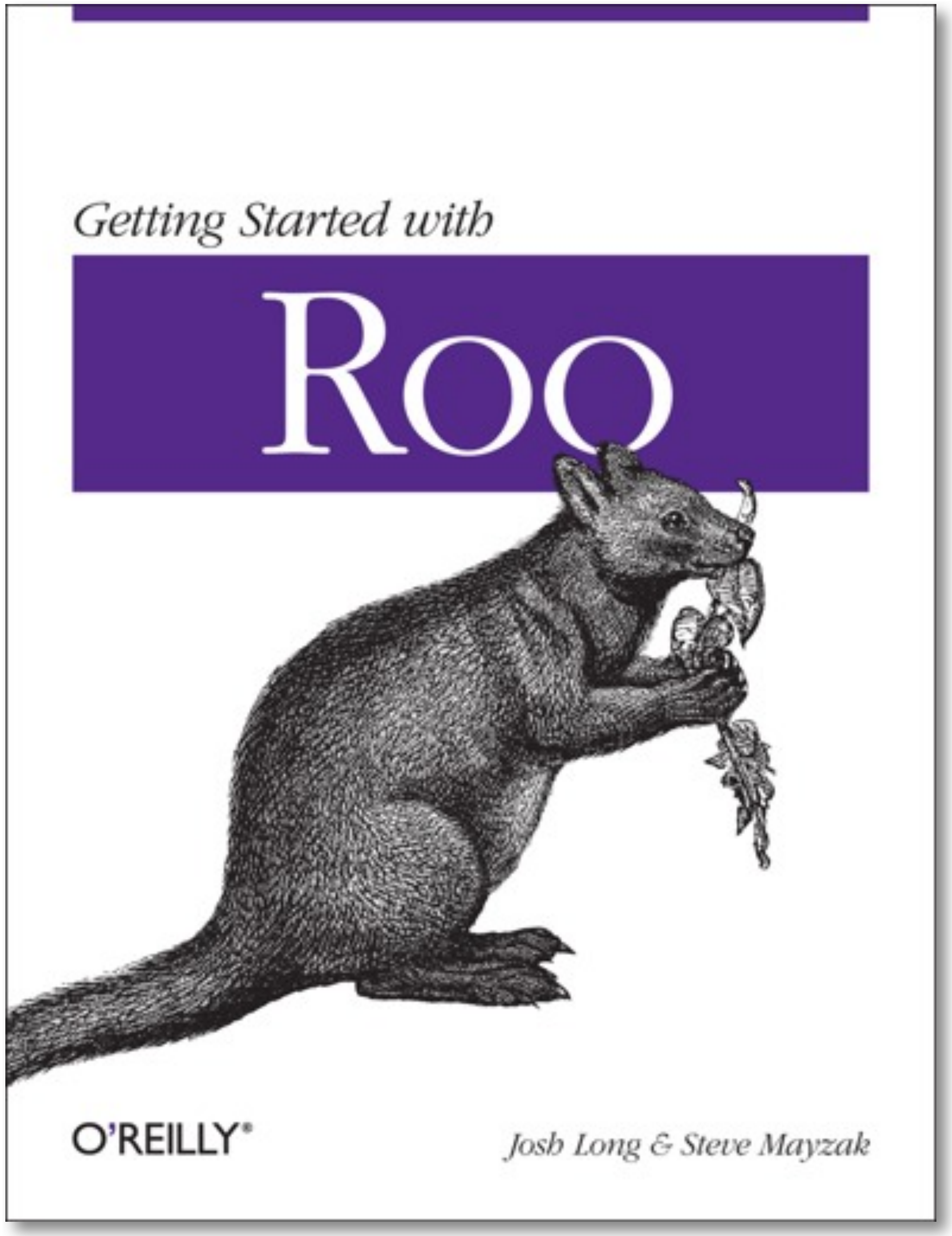
Developer Advocate for

BETA
CLOUD FOUNDRY™
DEPLOY & SCALE YOUR APPLICATIONS IN SECONDS

About Josh Long

Spring Developer Advocate
twitter: @starbuxman
josh.long@springsource.com


THE EXPERT'S VOICE® IN OPEN SOURCE
Covers Spring Framework 3
Spring Recipes
A Problem-Solution Approach
SECOND EDITION
Gary Mak, Josh Long, and Daniel Rubio
Apress®



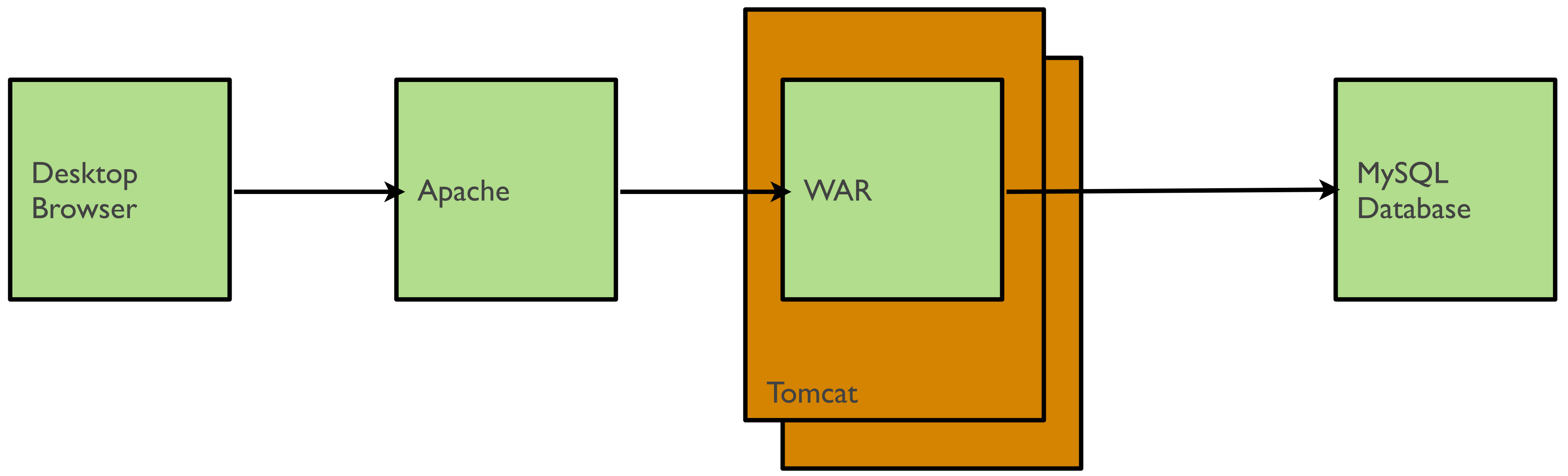
By The Way

Promo Code:
JFokus

Agenda

- **Why Cloud? Why PaaS?**
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Developing NoSQL applications for Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

Traditional web application architecture



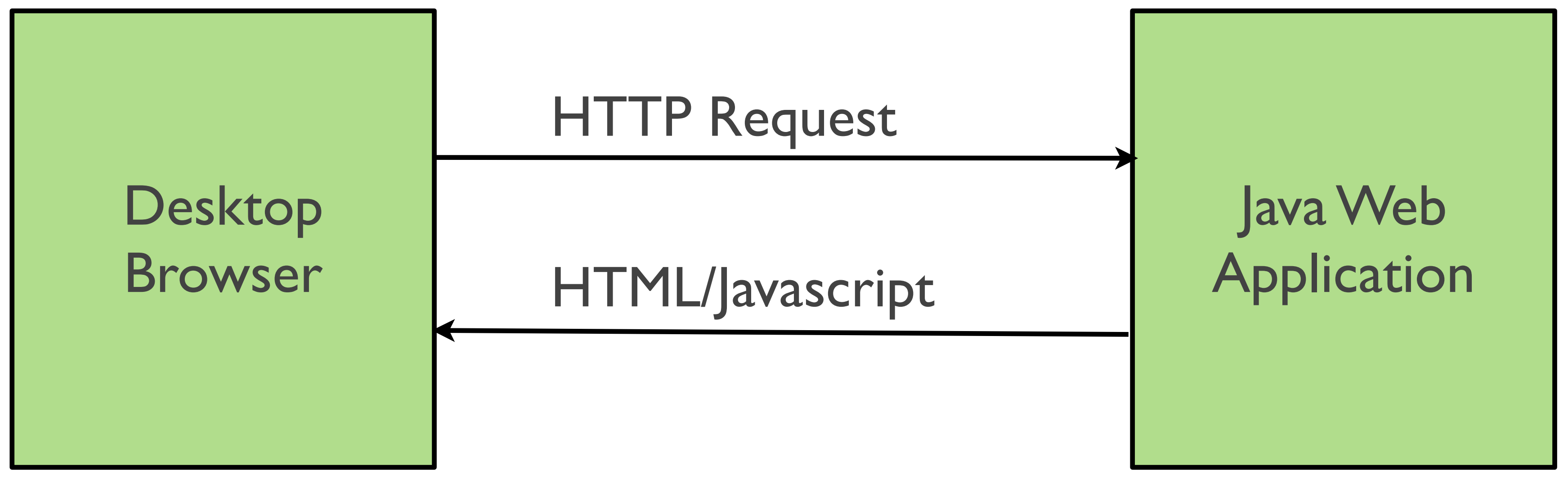
Simple to develop
Simple to test
Simple to deploy
Simple to scale: just add Apache + more Tomcats

But things are changing: this simple architecture is inadequate

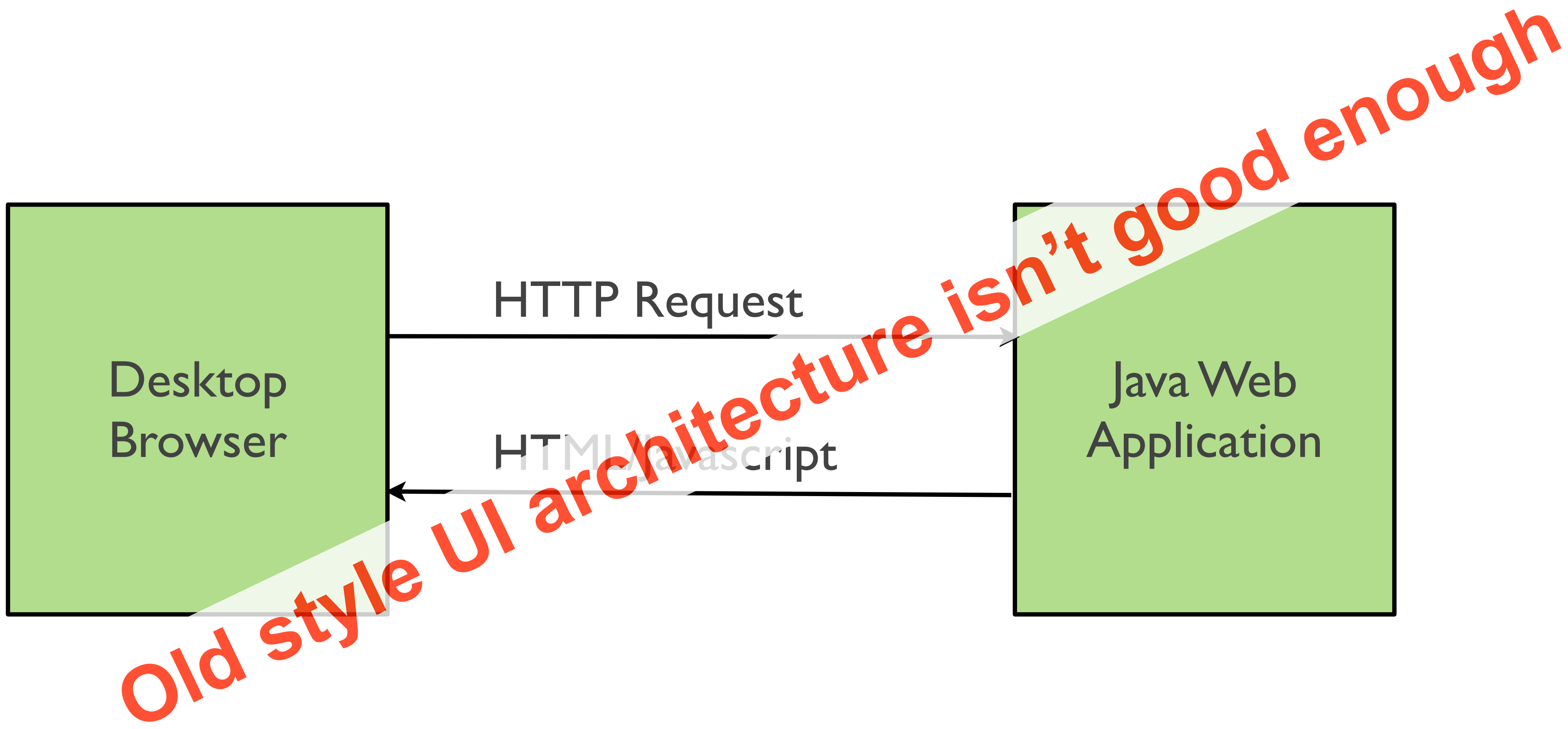
New kinds of clients

Smart phones overtake PCs in Q4 2010

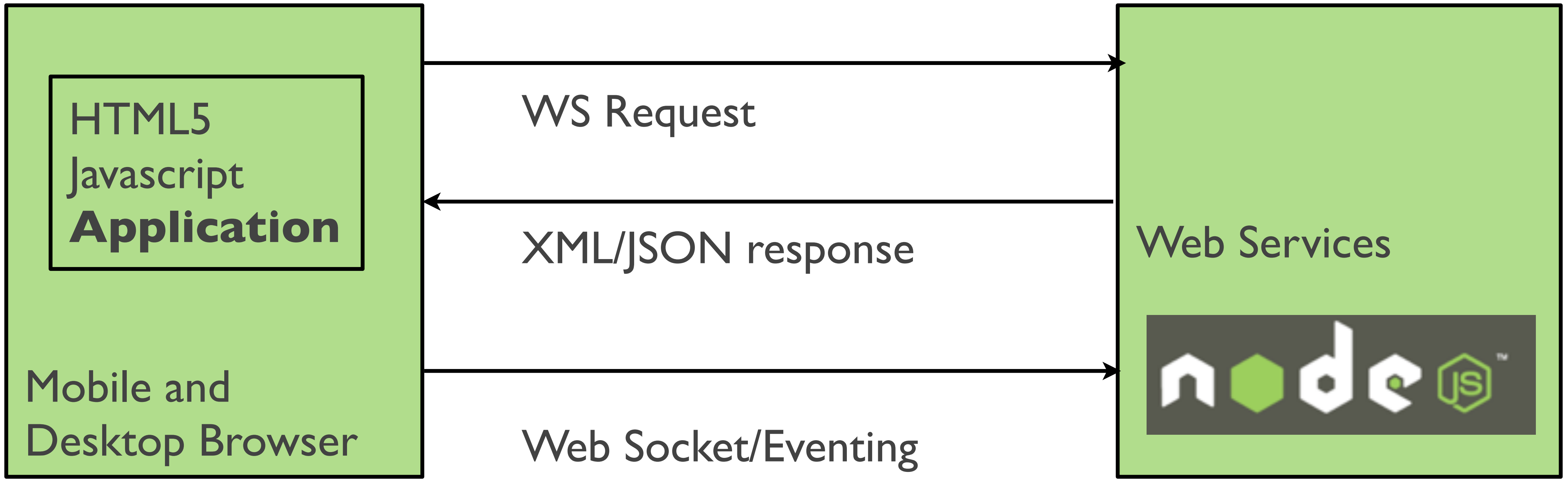
Users expect a rich, dynamic and interactive experience on mobile devices and desktop



Users expect a rich, dynamic and interactive experience on mobile devices and desktop



Users expect a rich, dynamic and interactive experience on mobile devices and desktop



Finally we can have a 1980s UIs :-)

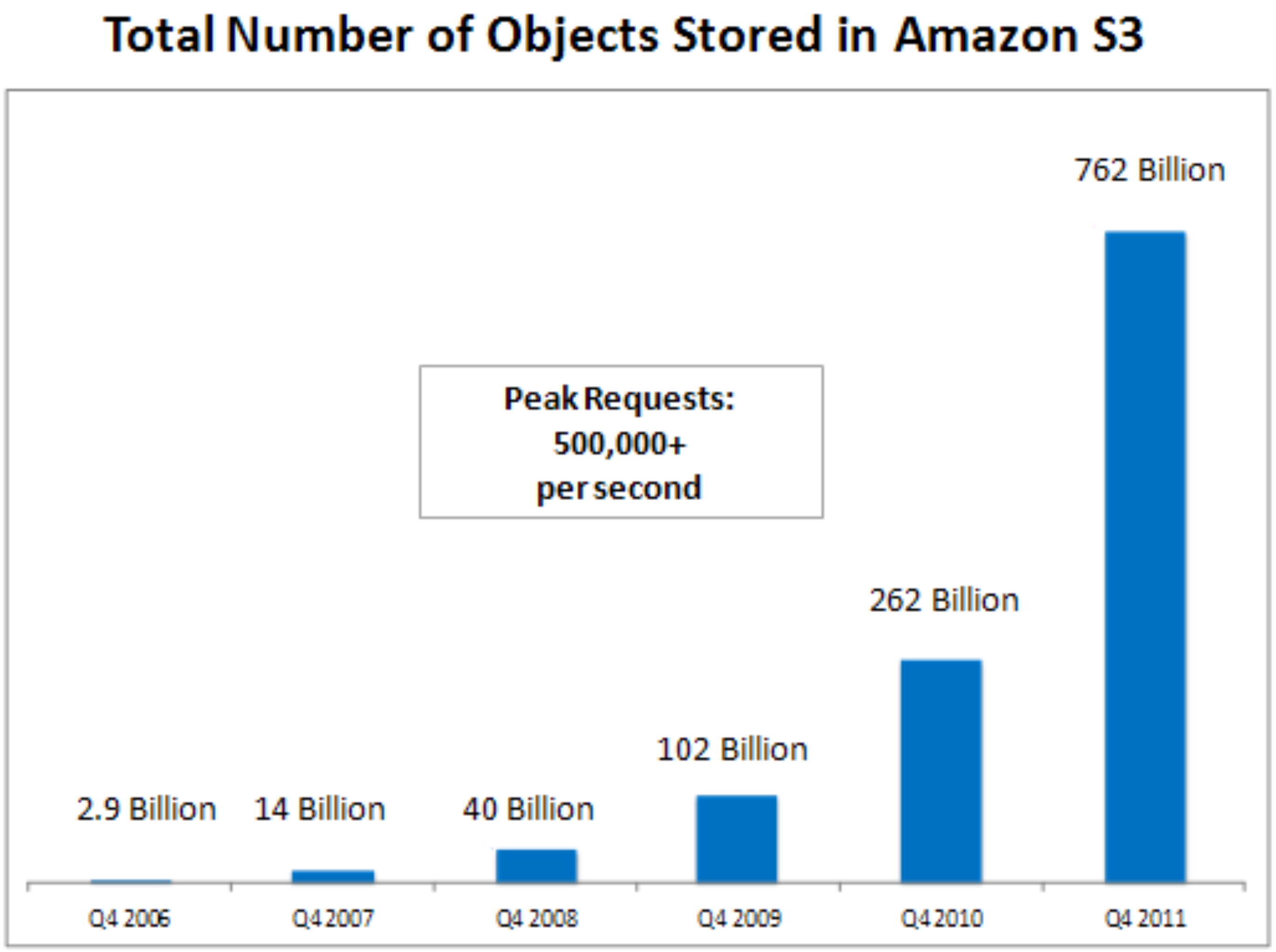
Popular social networks

- **Applications need to integrate with them**
  - Application integration problem
  - Scaling graphs is challenging
- **Application go viral through social networks**
  - Very rapid growth
  - Capacity planning nightmare

Need scalable architectures to handle massive loads

- **Application tier:**
  - Replicated/clustered servers
  - Modular so that components can be scaled differently
  - Asynchronous architecture - communication via a message broker
- **Database tier:**
  - Replication
  - Sharding
  - Polyglot persistence: Relational, NoSQL, NewSQL databases

Data Explosion: Data Volumes increasing at 60% per year

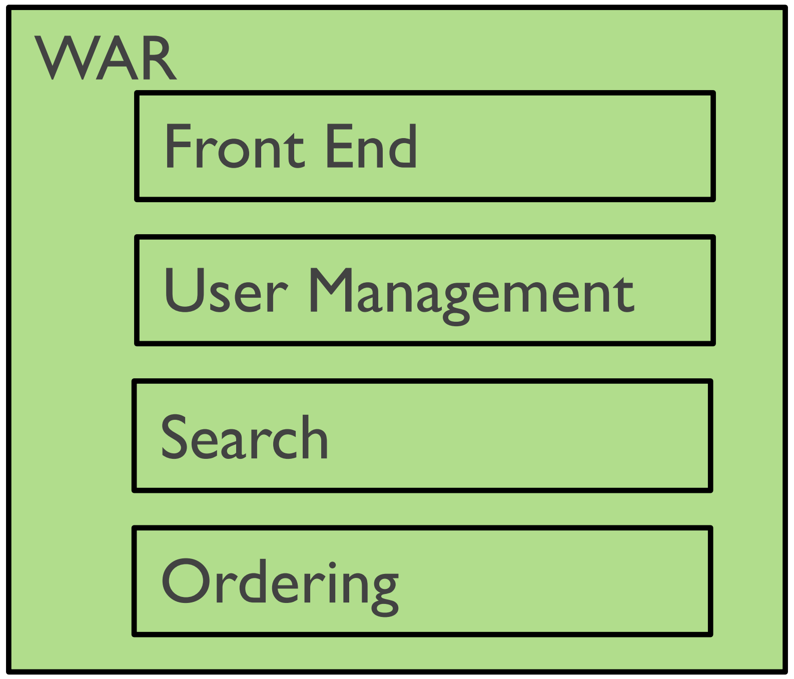


amazon web services

Horizontally scalable, distributed NoSQL Databases

Eventual consistency rather than ACID

Scaling development

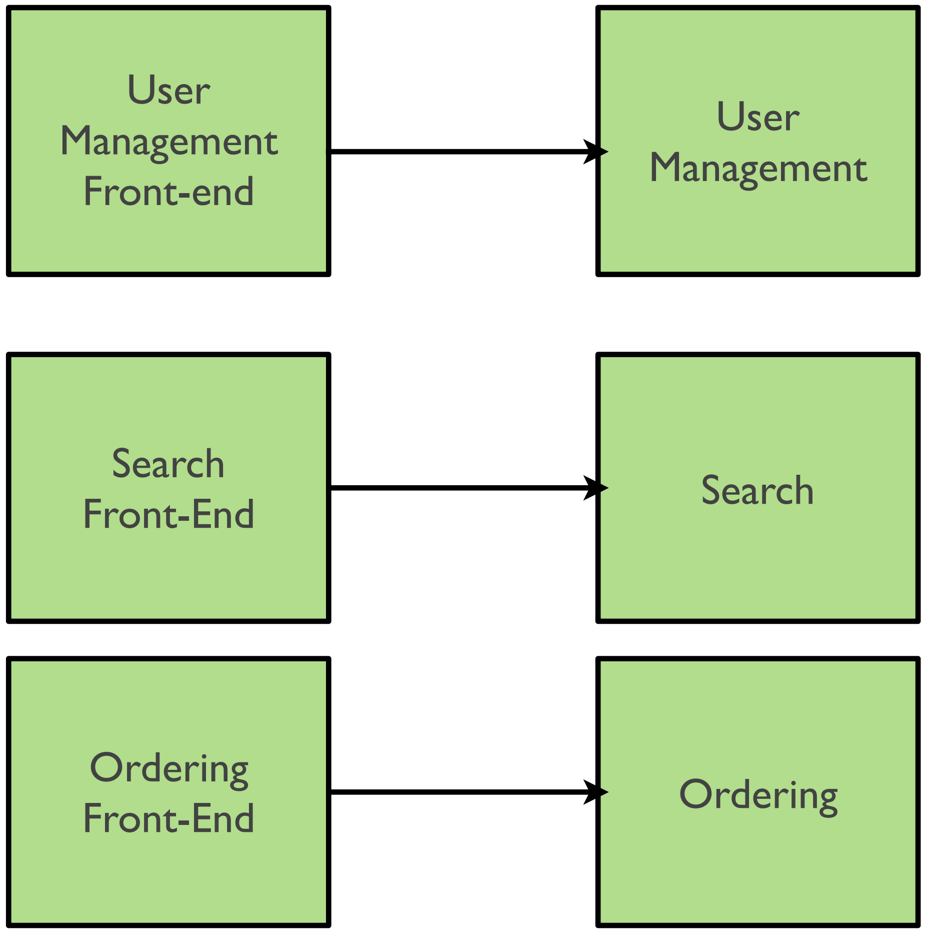


!= Scalable development

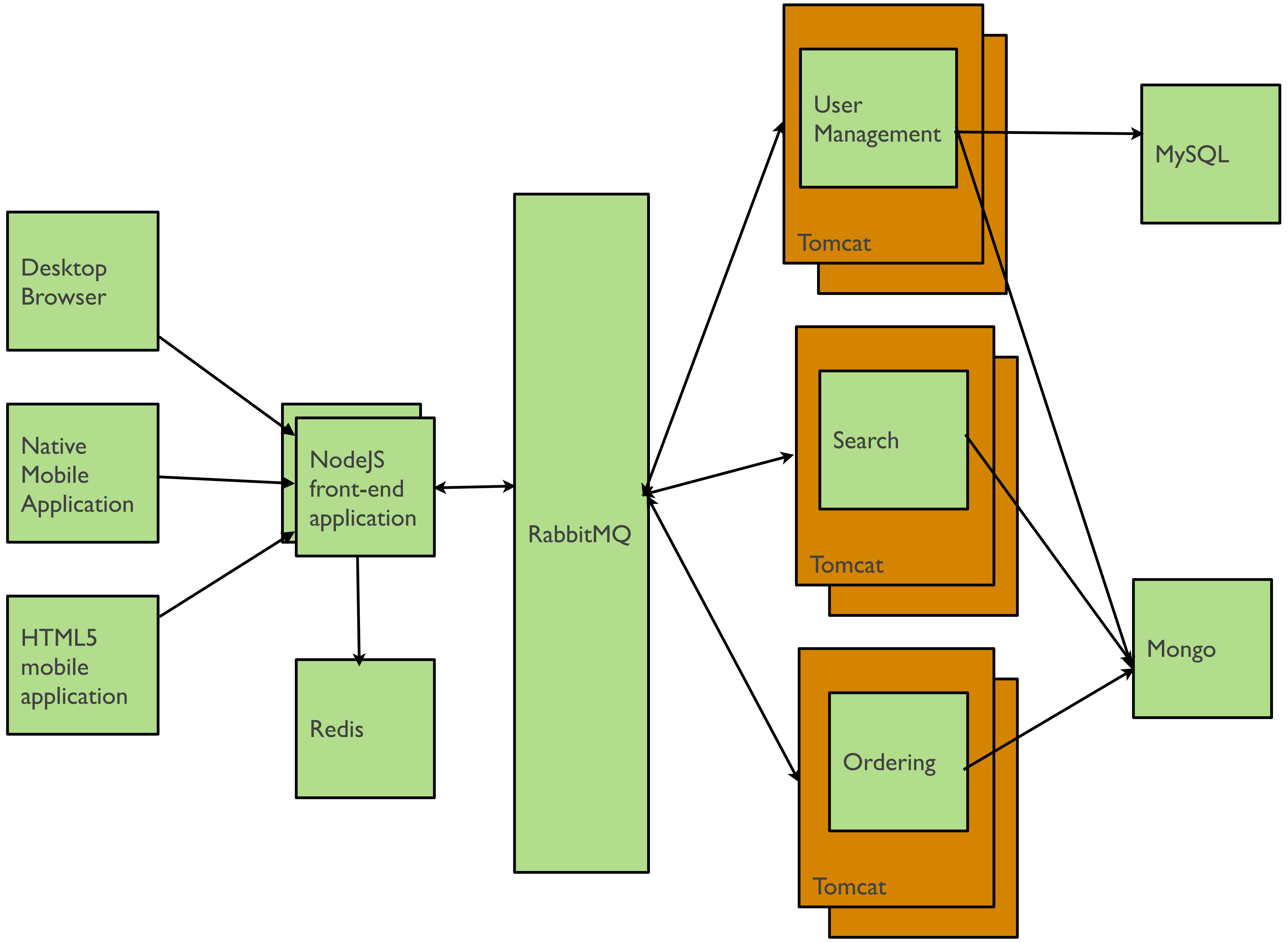
- Forces multiple developers/teams to synchronize development efforts
- Obstacle to frequent, independent deployments
- Increases risk of failure - need to redeploy everything to change one thing

Scaling development

- **Need “SOA” approach**
  - Partition application into set of services
  - Partition by noun or by verb
- **Each team is responsible for a service and manages their own release schedule.**
  - New code updates frequently
  - Mature services upgrade infrequently



Modern application architecture



Developing and testing these applications is challenging

Let’s imagine...

You are fixing a bug and want to run some JUnit integration tests

Who is going to install and configure your sandbox: MySQL, RabbitMQ, MongoDB, ...?

Let’s imagine...

You have fixed a bug and want to run some functional tests

How long to purchase the servers?

Who is going to set up the servers?

Who is going to install and configure MySQL, RabbitMQ, MongoDB, ...?

Let's imagine…

You want to deploy that application in production

How many servers do you need?

How quickly can you scale up?

Who is going to manage those servers?

http://www.oaklandzoo.org/site/zoo-info/animal-management/about-zookeeping

Who is going to carry the pager and answer that 3am call?

Cloud Computing is the solution

Cloud computing defined

IT delivered as a service
Over the internet
Self-service
Pay per use

Three layers of Cloud Computing

SaaS
Software as a Service

salesforce.com

Google Apps

Zimbra

workday

PaaS
Platform as a Service

CLOUD FOUNDRY

Windows Azure

heroku

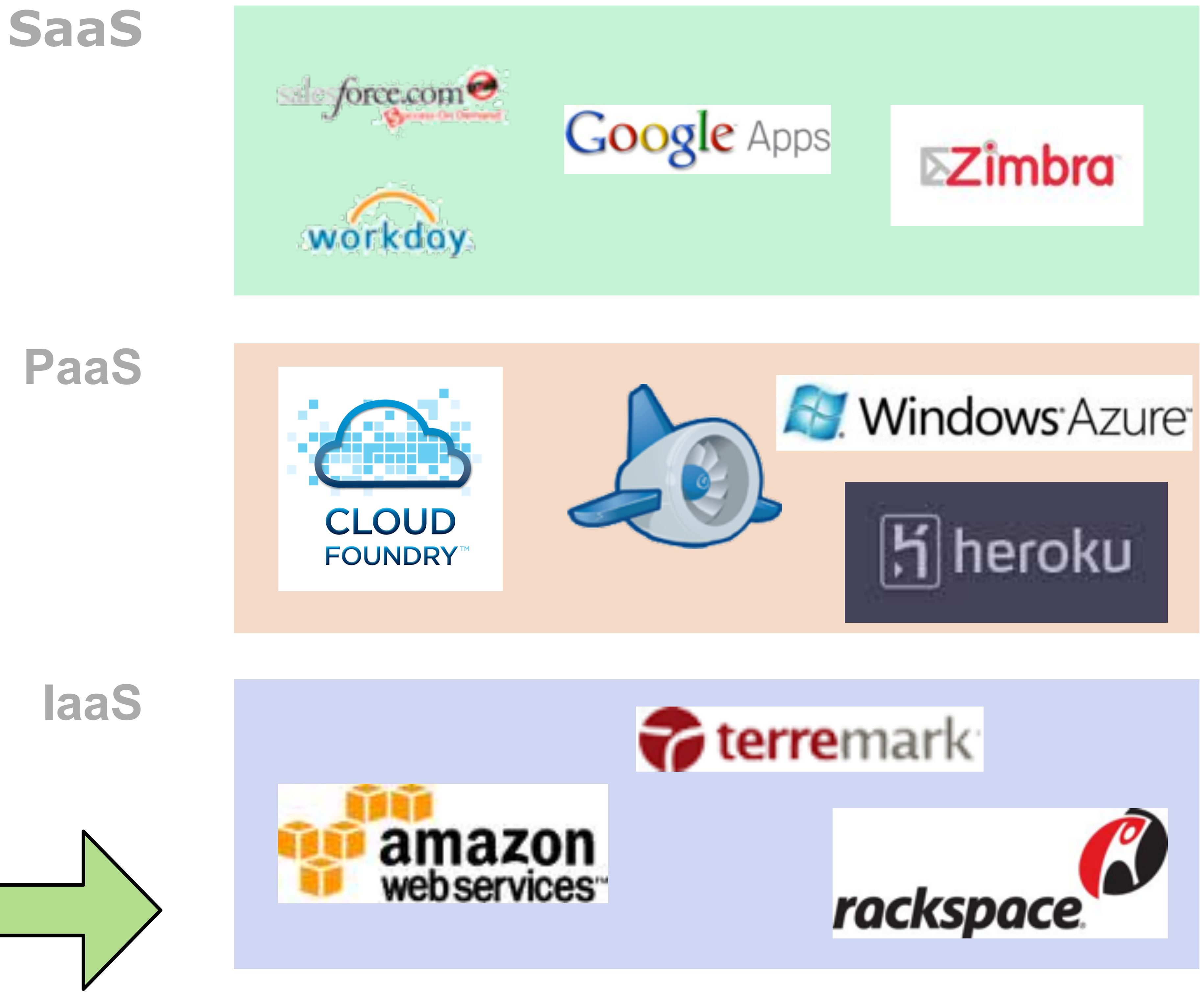
IaaS
Infrastructure as a Service

terremark

amazon web services

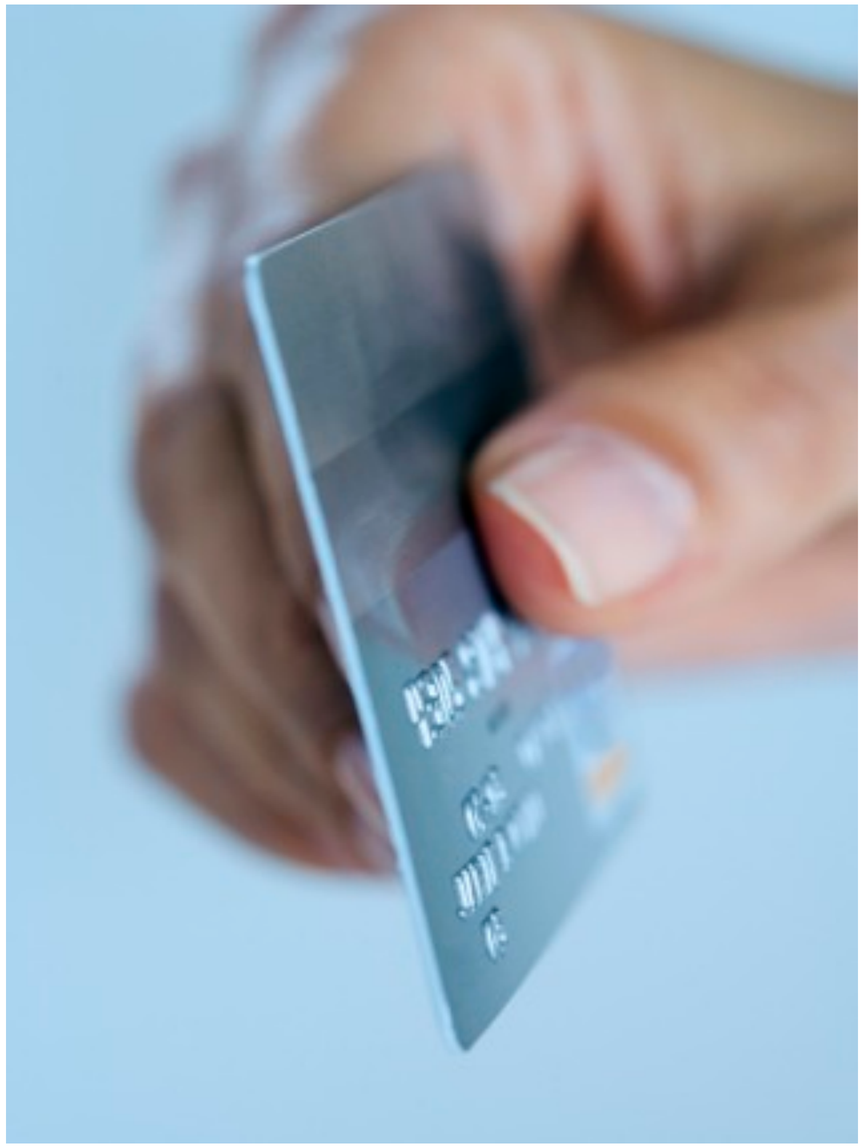
rackspace

Amazon EC2 = IaaS



rent a
server by
the hour

Sign up and deploy your application a few minutes later



- Login using your existing Amazon account
- Select the web services you want to use
- Only takes a few minutes

Benefits of IaaS for small companies

- **Get up and running quickly**
- **Validate your business idea without:**
  - Upfront costs
  - Long-term financial commitment
- **Leverage operational expertise of others**
- **Easily identify the right hardware for your application**
- **Scale up/down with load**
- **Reduces the risk of a success catastrophe**

Benefits of IaaS for enterprises

- **Increased agility - no need to wait for corporate IT**
  - In some companies it can take 2 months to acquire hardware
  - Requires a long-term financial commitment, upfront costs
- **Use for short-term projects, e.g.**
  - Websites for marketing campaigns
  - New York Times style projects
- **Reduce costs - use for applications that have fluctuating loads, e.g. heavily used once a week, once a month**

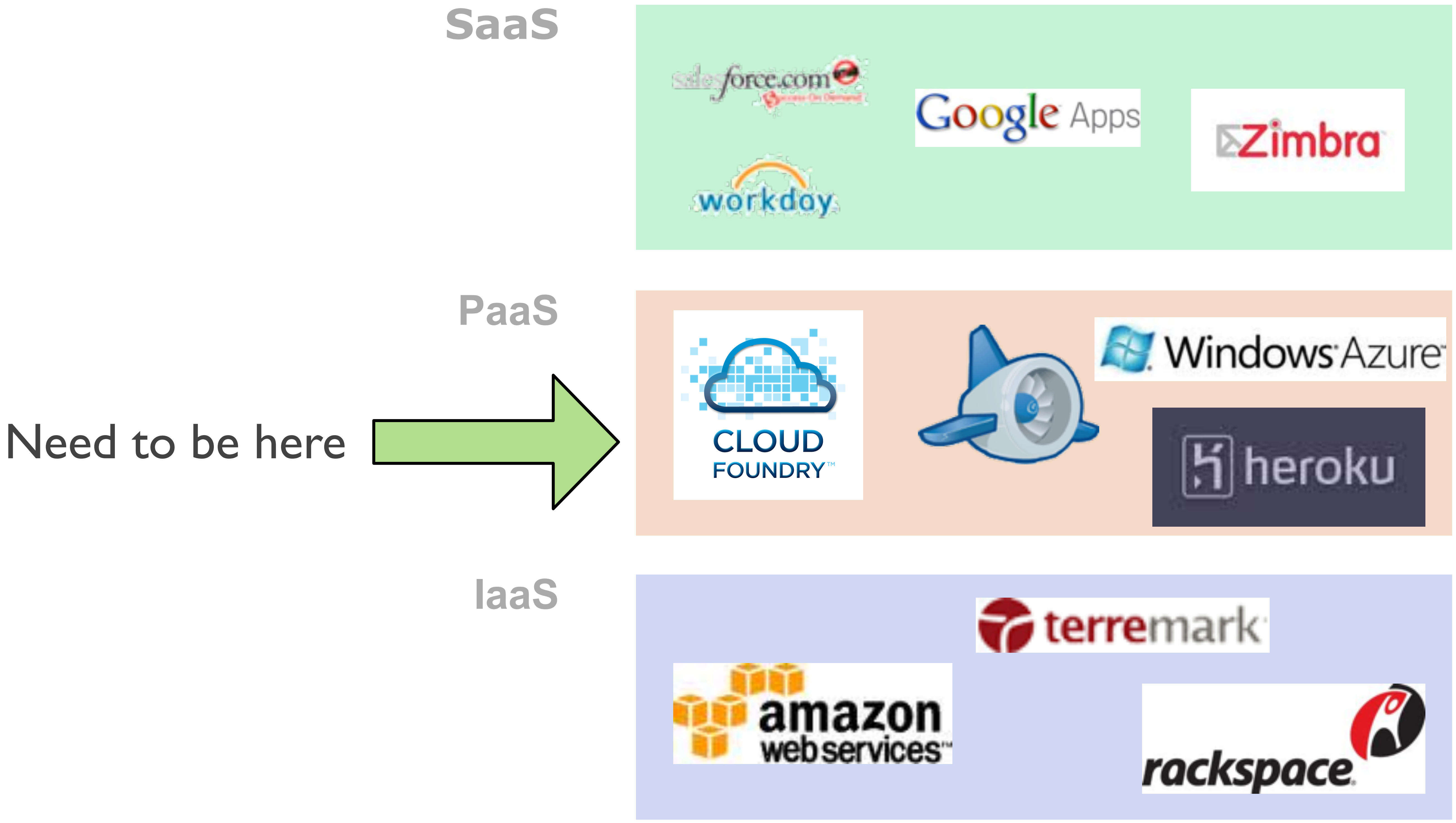
IaaS: Lots of flexibility BUT

You all you get

```
$ ssh ...
root@ec2-67-202-41-150.compute-1.amazonaws.com
Last login: Sun Dec 30 18:54:43 2007 from 71.131.29.181
[root@domU-12-31-36-00-38-23:~]
```

Everything else is your responsibility

We need to move up the stack



What you need is PaaS =

Easy deployment

Application management

Easy scaling up and down

+

Services:
Database
Blob storage
Messaging
...

Developers no longer need to be the janitor

Imagine if architects had to be the janitor for every building they designed. This is how the development team felt prior to moving to Windows Azure.

Duncan Mackenzie Nov 07, 2011
http://www.infoq.com/articles/Channel-9-Azure

PaaS Today

Run your web apps on Google's infrastructure
Easy to build, easy to maintain, easy to scale

heroku

OPENSHIFT™
PaaS by Red Hat Cloud

amazon web services

Windows Azure

CLOUD FOUNDRY™
No Obstacles: Deploy and Scale Your Applications in Seconds

The need for private PaaS

- Public PaaS is great

BUT

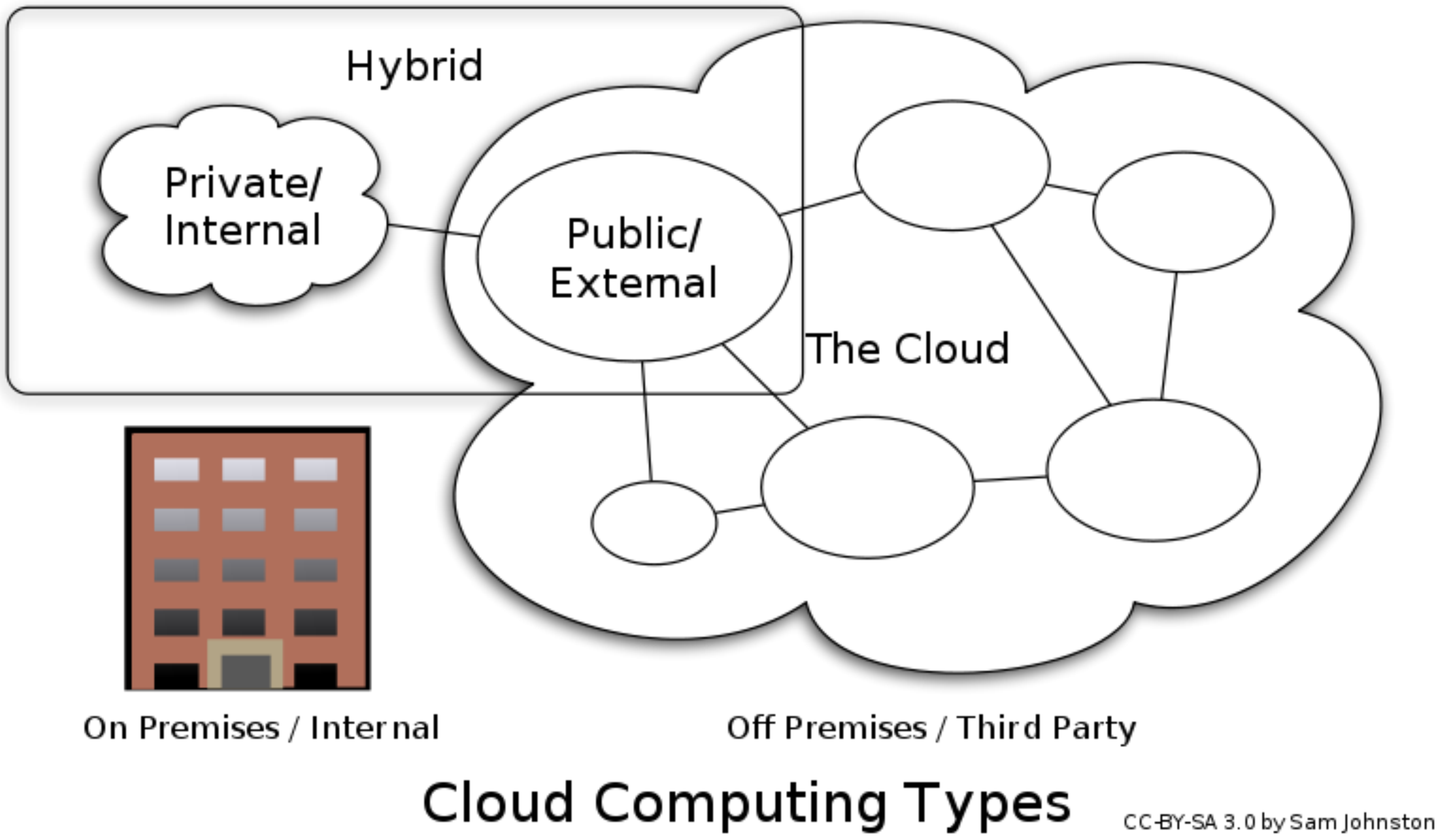
- Trust
- The need to feel in control
- Investment in existing data centers
- Compliance with regulations
- ...

THEREFORE

- Run a PaaS in your own datacenter

And why not have your own very private PaaS on your desktop?

Cloud Deployment models



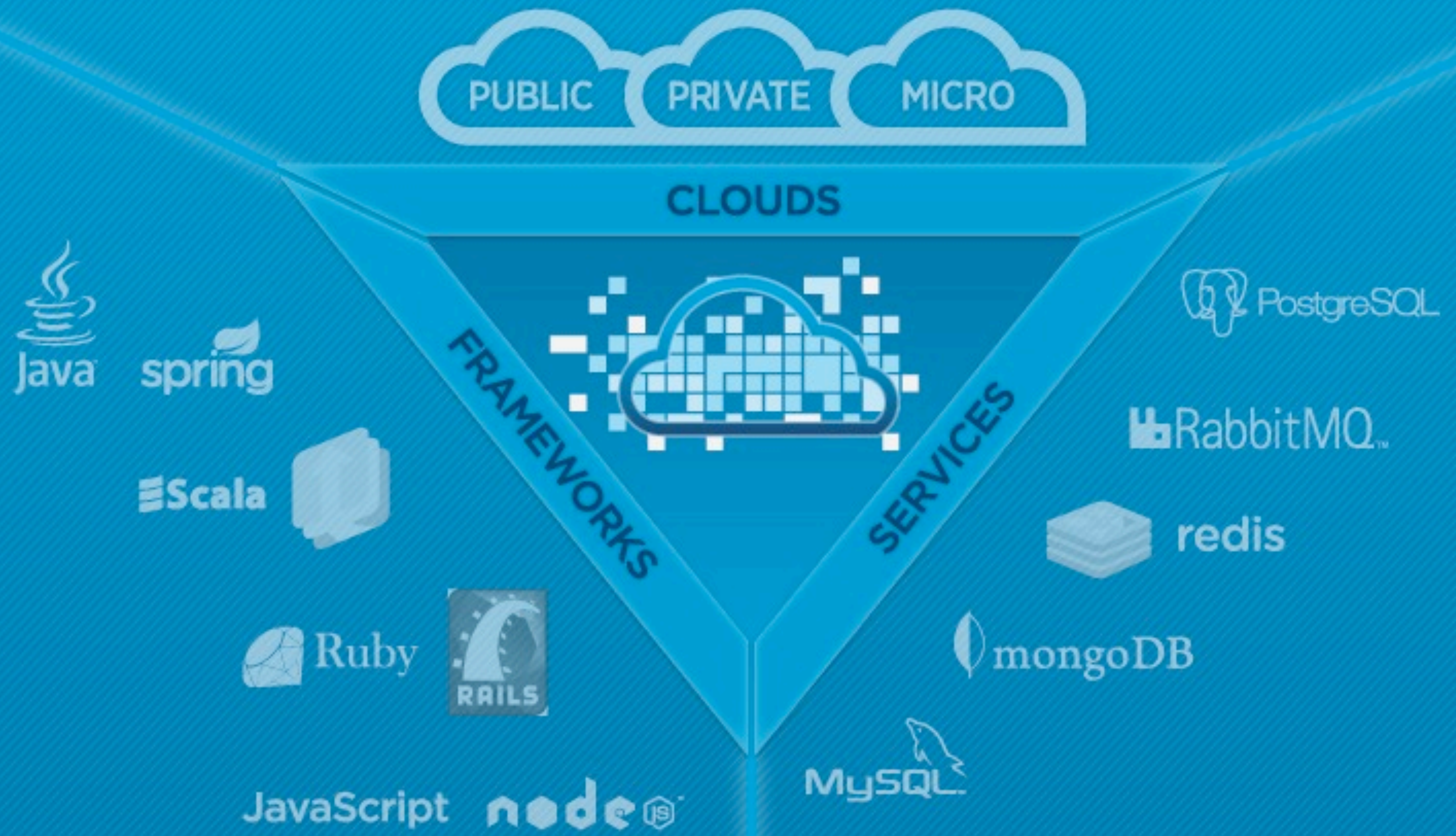
http://en.wikipedia.org/wiki/File:Cloud_computing_types.svg

Ideally: Public and Private PaaS use the same technology

Agenda

- Why Cloud? Why PaaS?
- **Introducing Cloud Foundry**
- Cloud Foundry for Spring developers
- Developing NoSQL applications for Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

Cloud Foundry: Services, Frameworks and Clouds



Monday, February 13, 12

Cloud Developer Bill of Rights (www.developerrights.org)

- **The Right to Code**
  use the best tools for the job
- **The Right to Build Applications (and Only Applications) :**
  devs != admins
- **The Right to Cloud Portability :**
  write once, run anywhere (really!)
- **The Right to a Choice of Frameworks**
  I say "potato," you say "Node.js"
- **The Right to a Choice of Application Services**
  MySQL, Redis, Mongo, All? More?
- **The Right to Platform Transparency**
  simple != opaque; I need logs damnit!
- **The Right to Emigrate**
  it's your code, your data, always. you can take it and leave.
- **The Right of Ownership**
  it's your code, your data, always. you own access rights.
- **The Right to Be Left Alone**
  even applications need personal space, respect!
- **The Right to Open Source**
  lots of clouds during spring - both Apache2 licensed!

Flexible Administration

Application Lifecycle API

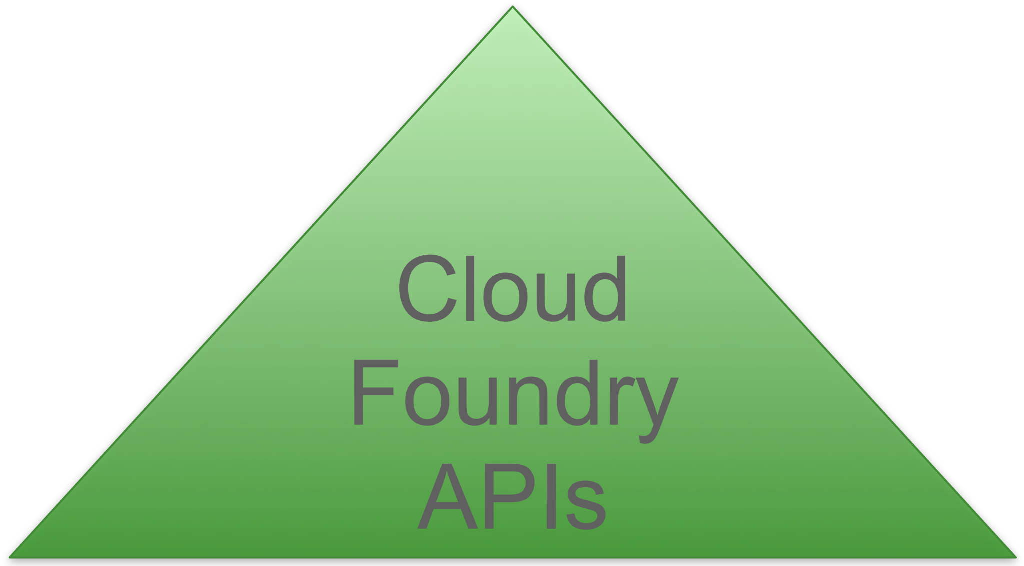
- Create, start, stop, update
- Set URL(s), instance count, memory
- Get stats, logs, crashes, files

Services API

Enumerate system services

Select and create service instance

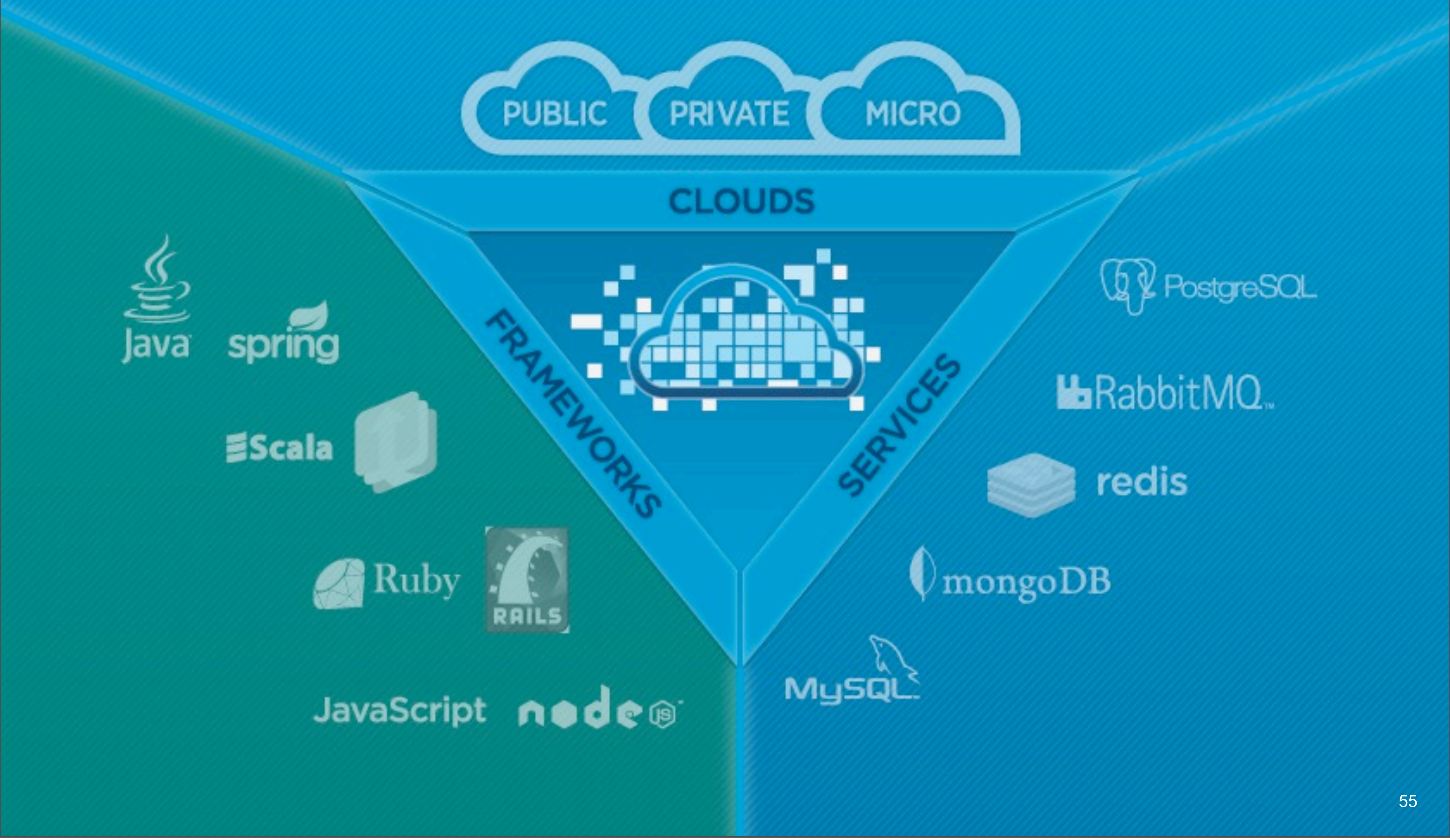
Bind and unbind service & app(s)



Also includes

- account spacing
- clients: STS, VMC
- Info API for both system and account space

Cloud Foundry: Choice of Runtimes



Monday, February 13, 12

Frameworks and Runtimes Supported

- **Out of the Box**
  - **Java** (.WAR files, on Tomcat. Spring's an ideal choice here, of course..)
  - **Scala** (Lift, Play!)
  - **Ruby** (Rails, Sinatra, etc.)
  - **Node.js**
- **Other**
  - **Python (Stackato)**
  - **PHP (AppFog)**
  - **Haskell (1)**
  - **Erlang (2)**

1) http://www.cakesolutions.net/teamblogs/2011/11/25/haskell-happstack-on-cloudfoundry/
2) https://github.com/cloudfoundry/vcap/pull/20

Deploying an Application

CLI

Application name

Dir containing application (or, **`.WAR`** for Java)

```
$ vmc push cf1 --path target \
      --url cer-cf1.cloudfoundry.com
```

Application URL

Deploying an Application

```
$ vmc push cf1 --path target \
      --url cer-cf1.cloudfoundry.com
Detected a Java Web Application, is this correct?
   [Yn]:
```

Deploying an Application

```
$ vmc push cf1 --path target \
      --url cer-cf1.cloudfoundry.com
Detected a Java Web Application, is this correct?
   [Yn]:

Memory Reservation [Default:512M] (64M, 128M, 256M,
   512M, 1G or 2G)
```

Deploying an Application

```
$ vmc push cf1 --path target \
      --url cer-cf1.cloudfoundry.com
Detected a Java Web Application, is this correct?
   [Yn]:

Memory Reservation [Default:512M] (64M, 128M, 256M,
   512M, 1G or 2G)
Creating Application: OK
Would you like to bind any services to 'cf1'? [yN]:
```

Deploying an Application

```
$ vmc push cf1 --path target \
      --url cer-cf1.cloudfoundry.com
Detected a Java Web Application, is this correct?
   [Yn]:

Memory Reservation [Default:512M] (64M, 128M, 256M,
   512M, 1G or 2G)

Creating Application: OK
Would you like to bind any services to 'cf1'? [yN]:

Uploading Application:
  Checking for available resources: OK
  Packing application: OK
  Uploading (2K): OK
Push Status: OK
Starting Application: OK
```

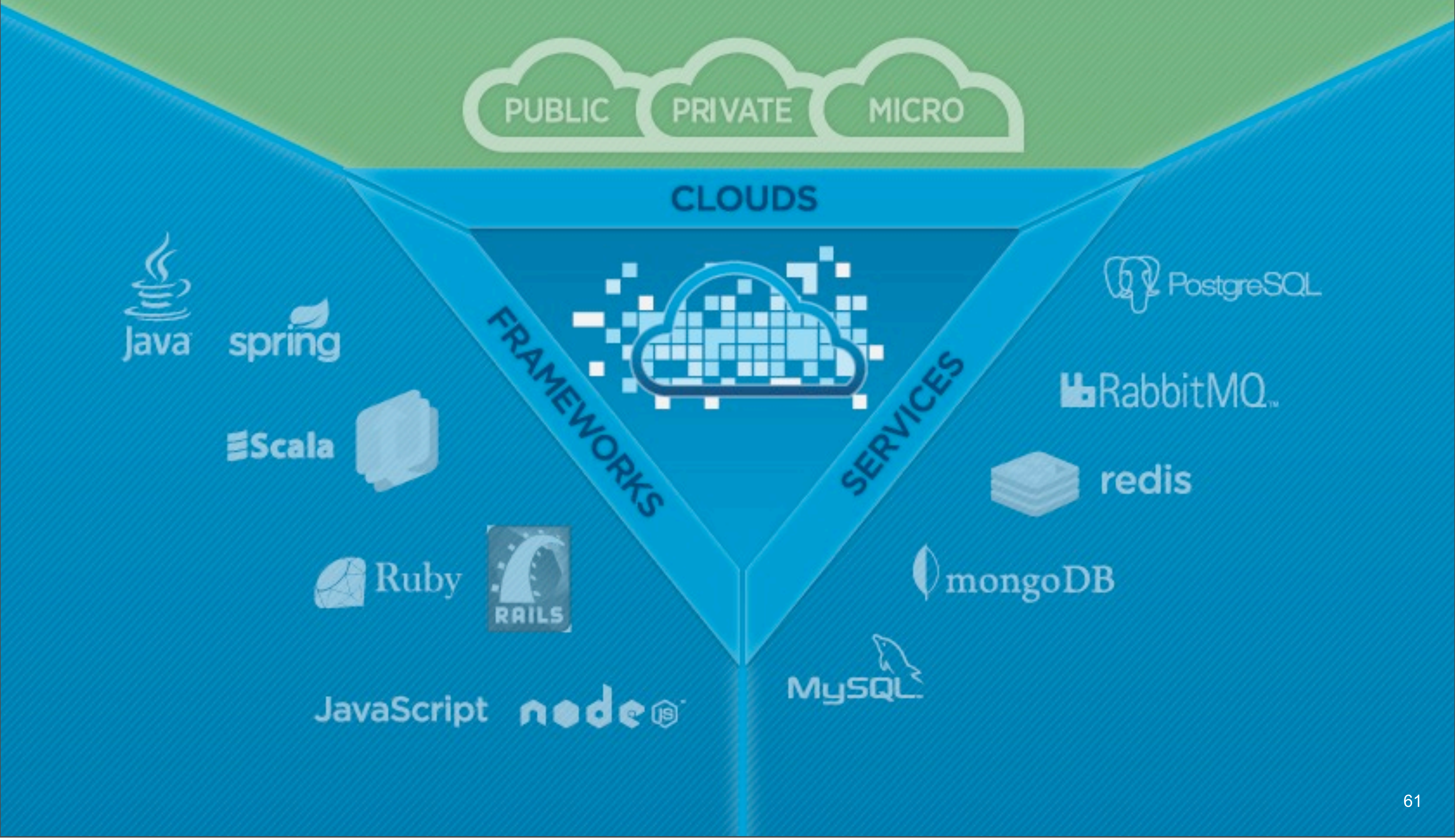
Deploying an Application (with a Manifest)

```
$ vmc push
Would you like to deploy from the current directory?
   [Yn]: y
Pushing application 'html5expenses'...
Creating Application: OK
Creating Service [expenses-mongo]: OK
Binding Service [expenses-mongo]: OK
Creating Service [expenses-postgresql]: OK
Binding Service [expenses-postgresql]: OK
Uploading Application:
  Checking for available resources: OK
  Processing resources: OK
  Packing application: OK
  Uploading (6K): OK
Push Status: OK
```

Deploying an Application (with a `manifest.yml`)

```
---
applications:
  target:
    name: html5expenses
    url: ${name}.${target-base}
    framework:
      name: spring
      info:
        mem: 512M
        description: Java SpringSource Spring Application
        exec:
    mem: 512M
    instances: 1
    services:
      expenses-mongo:
        type: :mongodb
      expenses-postgresql:
        type: :postgresql
```

Cloud Foundry: Choice of Clouds



Main Risk: Lock In

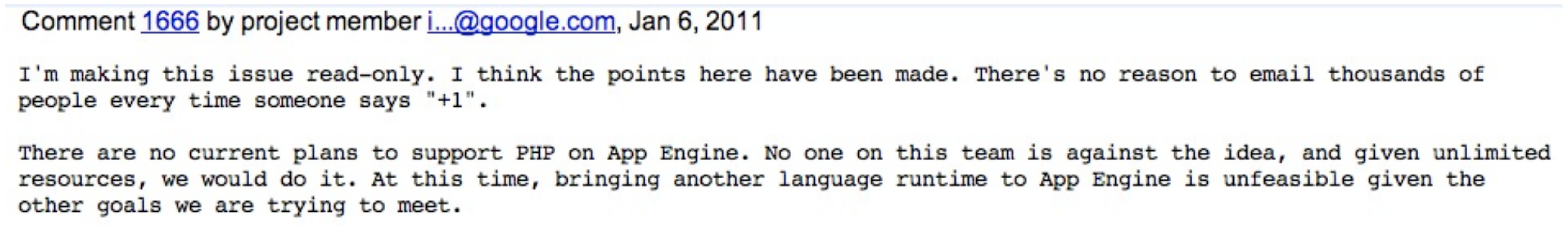
Welcome to the hotel california
Such a lovely place
Such a lovely face
Plenty of room at the hotel california
Any time of year, you can find it here

Last thing I remember, I was
Running for the door
I had to find the passage back
To the place I was before
'relax,' said the night man,
We are programmed to receive.
You can checkout any time you like,
But you can never leave!

-the Eagles

Open Source Advantage

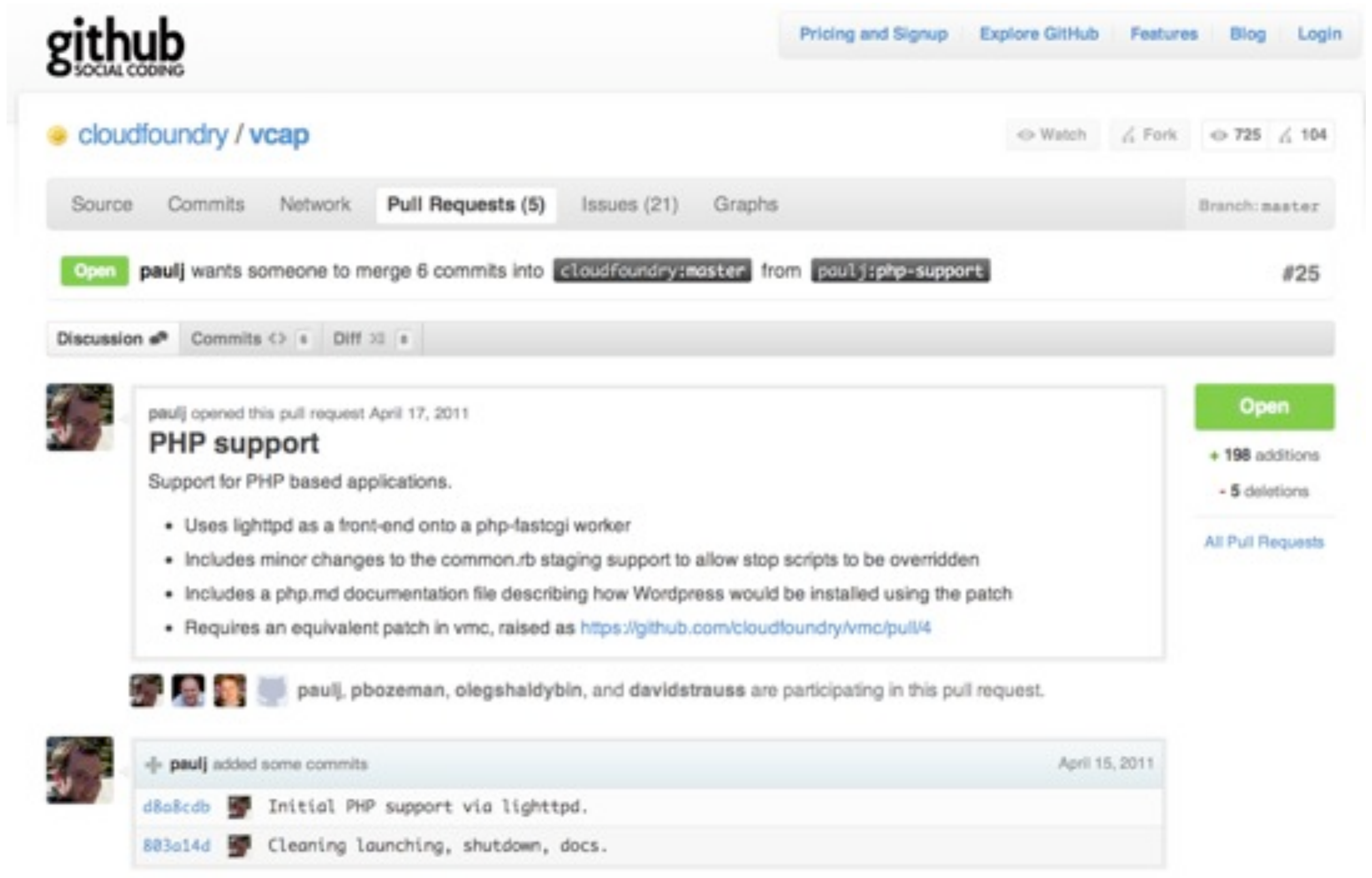
- http://code.google.com/p/googleappengine/issues/detail?id=13



Comment 1666 by project member i...@google.com, Jan 6, 2011

```
I'm making this issue read-only. I think the points here have been made. There's no reason to email thousands of
people every time someone says "+1".

There are no current plans to support PHP on App Engine. No one on this team is against the idea, and given unlimited
resources, we would do it. At this time, bringing another language runtime to App Engine is unfeasible given the
other goals we are trying to meet.
```

- https://github.com/cloudfoundry/vcap/pull/25



paulj wants someone to merge 6 commits into cloudfoundry:master from paulj:php-support #25

paulj opened this pull request April 17, 2011

PHP support

Support for PHP based applications.

- Uses lighttpd as a front-end onto a php-fastcgi worker
- Includes minor changes to the common.rb staging support to allow stop scripts to be overridden
- Includes a php.md documentation file describing how Wordpress would be installed using the patch
- Requires an equivalent patch in vmc, raised as https://github.com/cloudfoundry/vmc/pull/4

+ 198 additions
- 5 deletions

paulj, pbozeman, olegshaldybin, and davidstrauss are participating in this pull request.

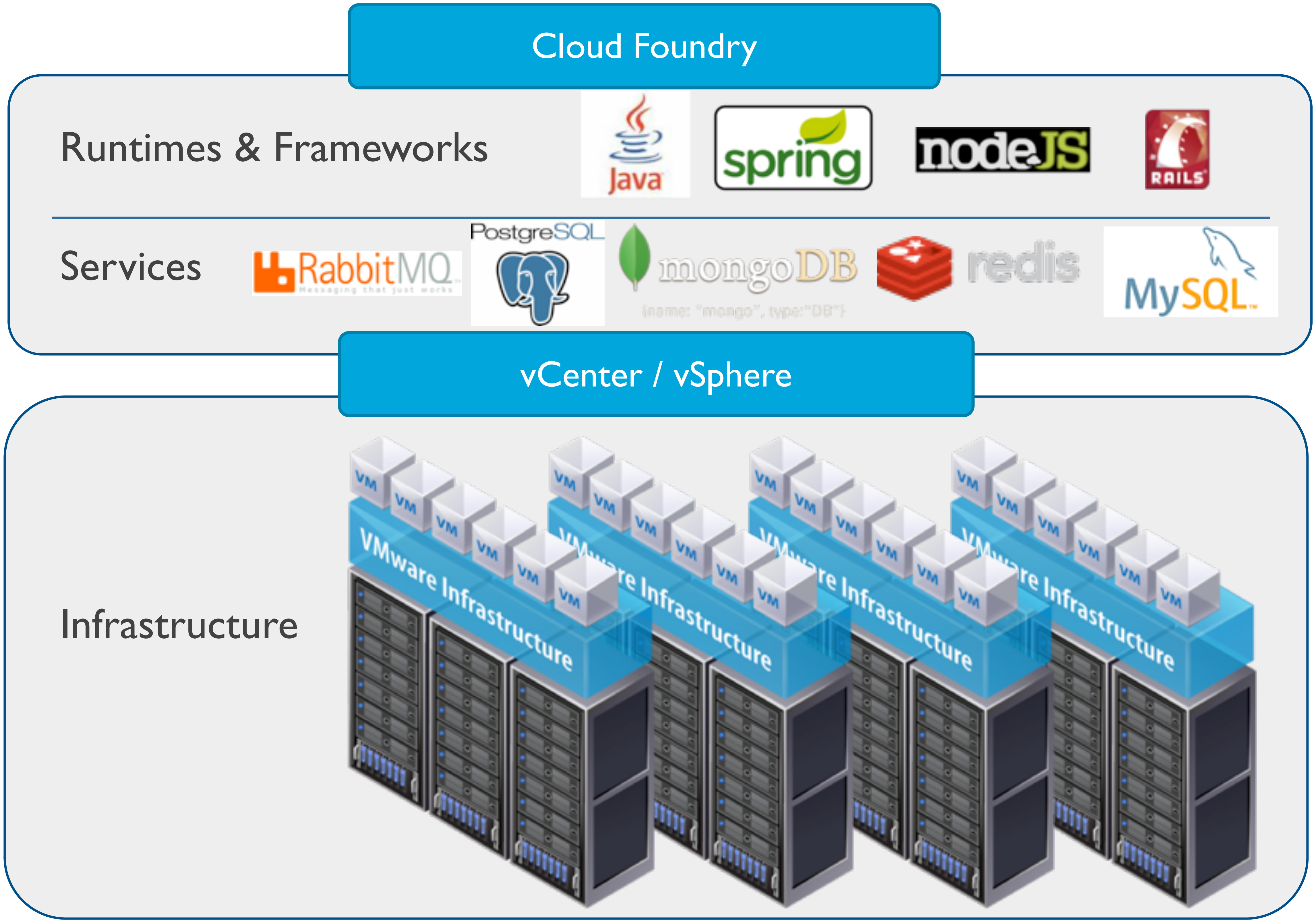
paulj added some commits April 15, 2011

```
d8a8cdb  Initial PHP support via lighttpd.
803a14d  Cleaning launching, shutdown, docs.
```

Cloud Foundry: Clouds

- **AppFog.com**
  - community lead for PHP
  - PaaS for PHP
- **Joyent**
  - community lead for Node.js
- **ActiveState**
  - community lead for Python, Perl
  - Providers of Stackato private PaaS

Cloud Foundry.com



Cloud Foundry.org

Cloud Foundry

The Source Code to Compile & Build Cloud Foundry

vCenter / vSphere

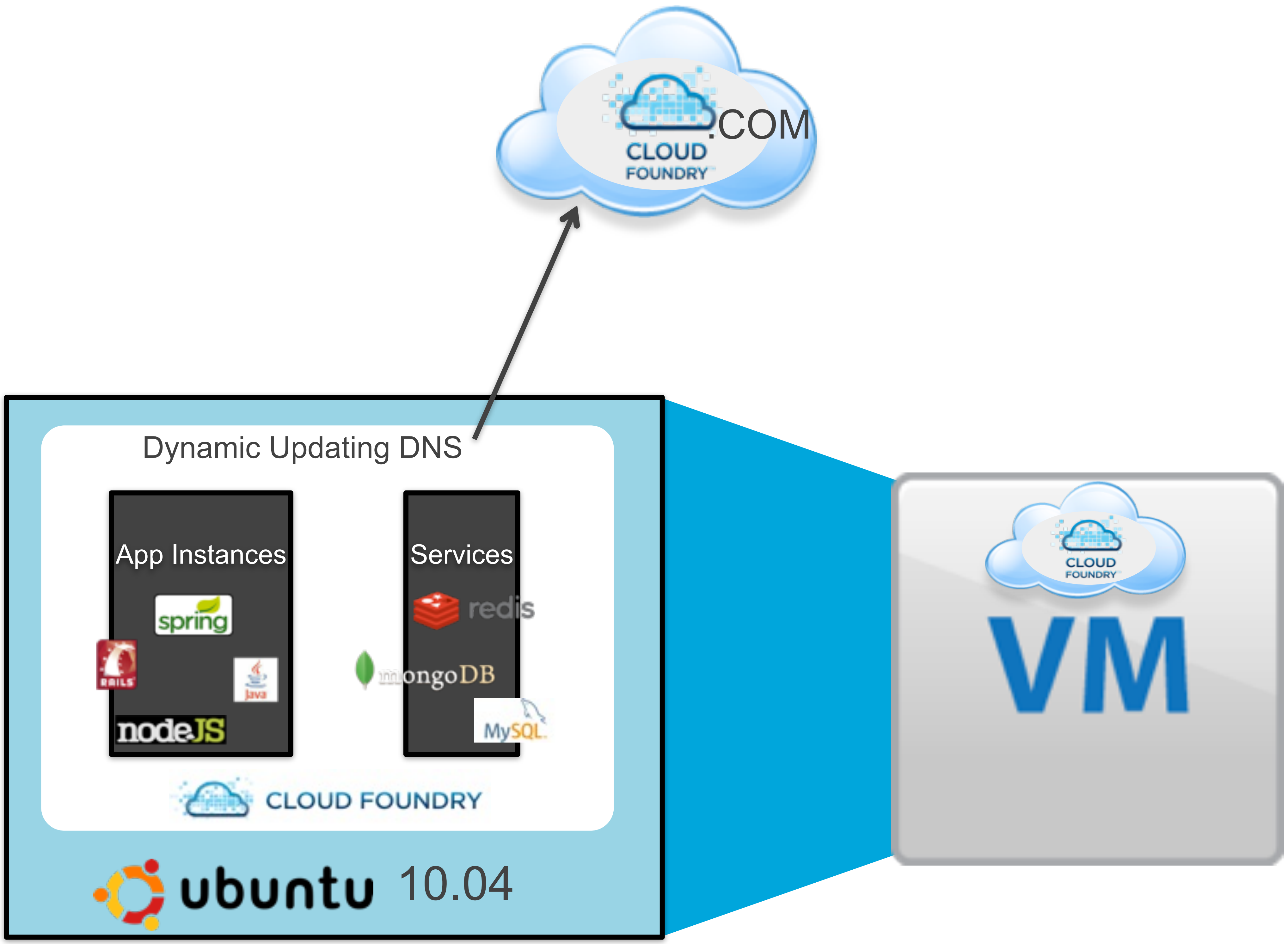
Download Code

Setup Environment

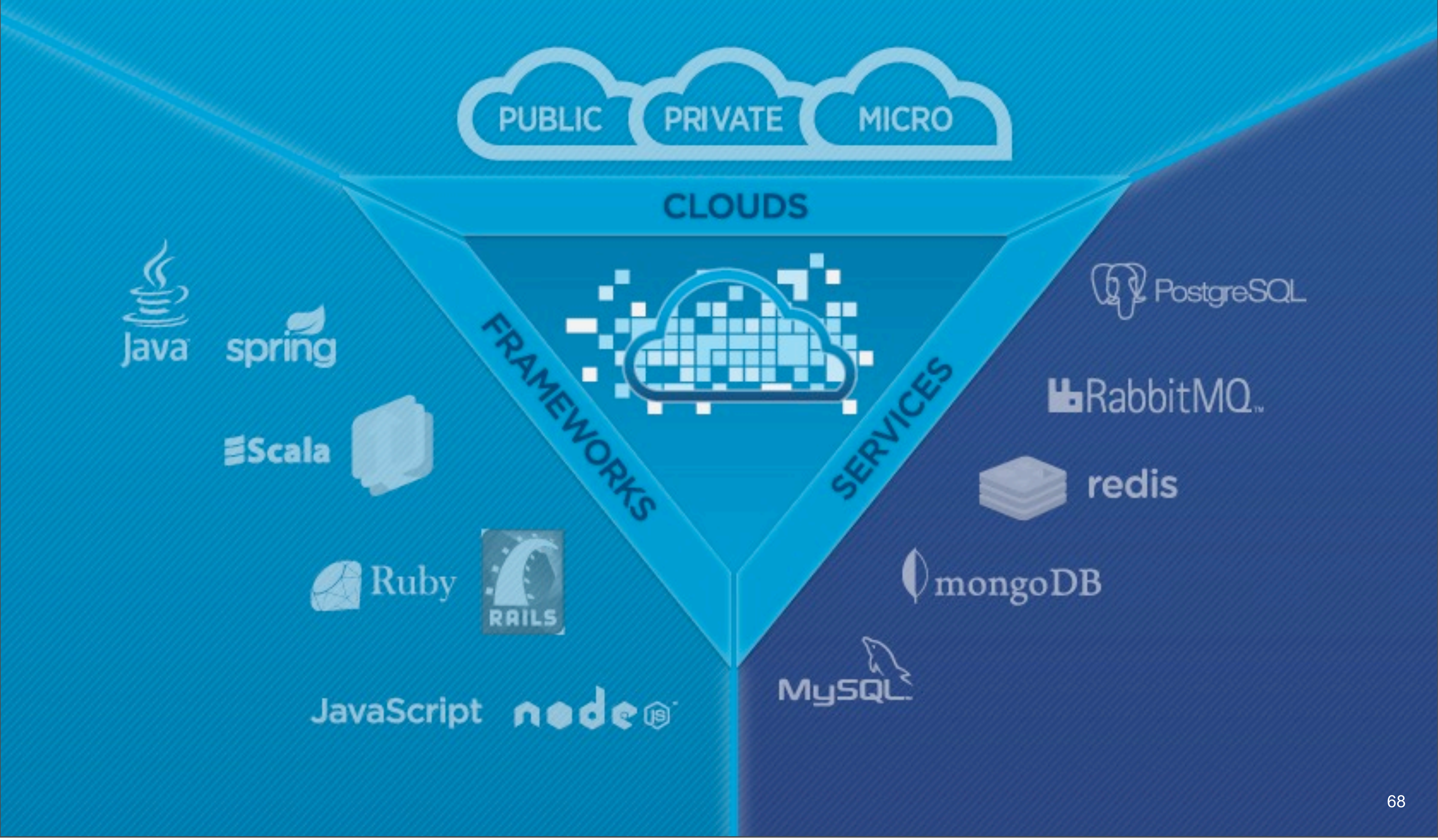
Setup Scripts

Deploy Behind Firewall

Micro Cloud Foundry (beta)



Cloud Foundry: Services



Cloud Foundry: Services

- **Services are one of the extensibility planes in Cloud Foundry**
  - there are more services being contributed by the community daily!
- **MySQL, Redis, MongoDB, RabbitMQ, PostgreSQL**
- **Services may be shared across applications**
- **Cloud Foundry abstracts the provisioning aspect of services through a uniform API hosted in the cloud controller**
- **It’s very easy to take an app and add a service to the app in a uniform way**
  - Cassandra? COBOL / CICS, Oracle

Cloud Foundry: Services

```
$ vmc create-service mysql --name mysql1
Creating Service: OK

$ vmc services

============== System Services ==============
+------------+---------+---------------------------------------+
| Service    | Version | Description                           |
+------------+---------+---------------------------------------+
| mongodb    | 1.8     | MongoDB NoSQL store                   |
| mysql      | 5.1     | MySQL database service                |
| postgresql | 9.0     | PostgreSQL database service (vFabric) |
| rabbitmq   | 2.4     | RabbitMQ messaging service            |
| redis      | 2.2     | Redis key-value store service         |
+------------+---------+---------------------------------------+

=========== Provisioned Services ============

+-------------+---------+
| Name        | Service |
+-------------+---------+
| mysql1      | mysql   |
+-------------+---------+
```

Cloud Foundry: Services Creation and Binding

```
$VCAP_SERVICES:
{"redis-2.2":
[{"name":"redis_sample","label":"redis-2.2","plan":"free",
"tags":["redis","redis-2.2","key-value","nosql"],
"credentials":
{"hostname":"172.30.48.40",
"host":"172.30.48.40",
"port":5023,
"password":"8e9a901f-987d-4544-9a9e-ab0c143b5142",
"name":"de82c4bb-bd08-46c0-a850-af6534f71ca3"}
}],
"mongodb-1.8":[{"name":"mongodb-
e7d29","label":"mongodb-1.8","plan":"free","tags”:.....................
```

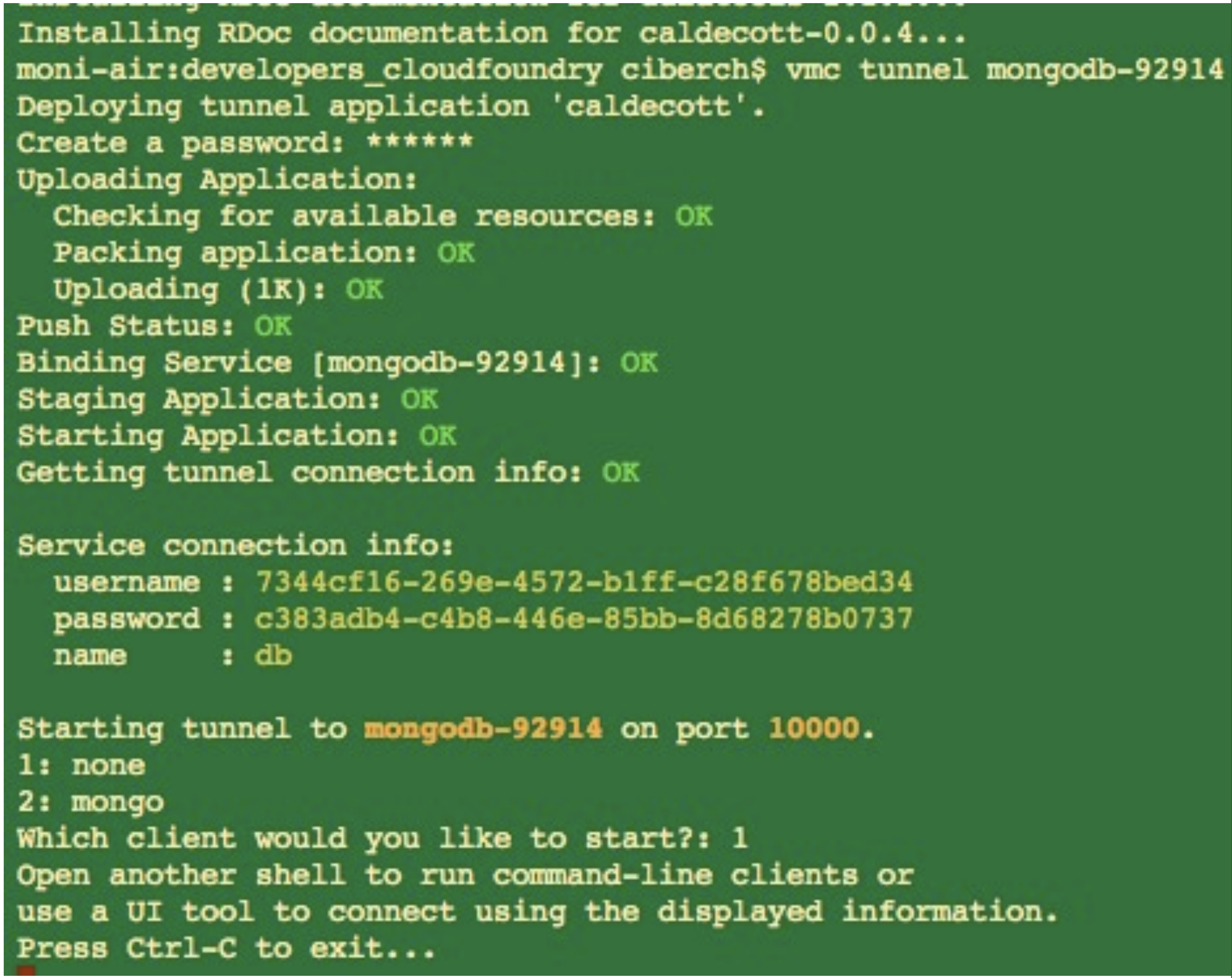
Accessing Your Services

- **Debugging and accessing the data locally**
  - Caldecott --> Service tunneling. Access your Cloud Foundry service as if it was local.

Tunneling

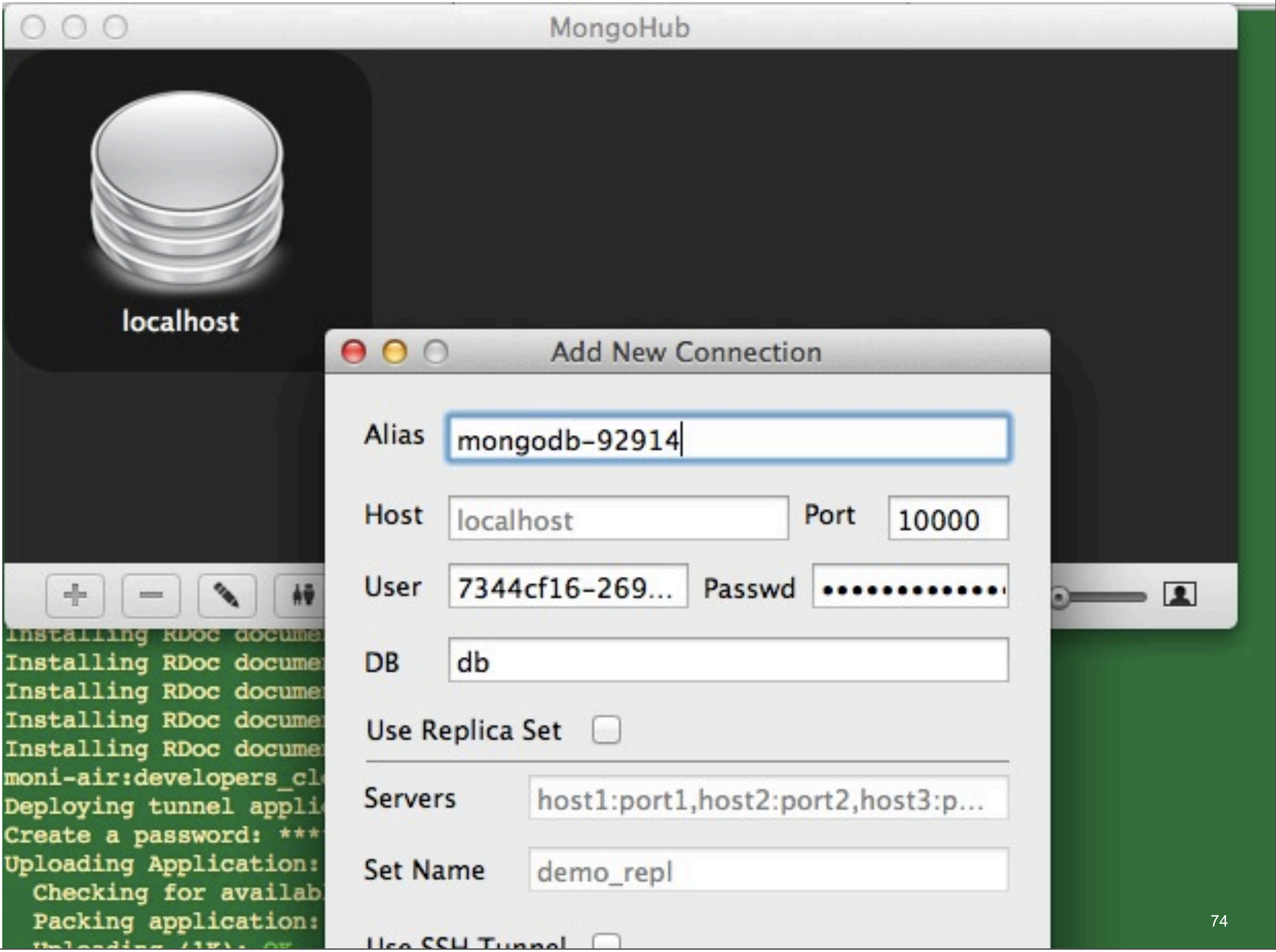
`gem install caldecott`

`vmc tunnel <mongodb>`



```
Installing RDoc documentation for caldecott-0.0.4...
moni-air:developers_cloudfoundry ciberch$ vmc tunnel mongodb-92914
Deploying tunnel application 'caldecott'.
Create a password: ******
Uploading Application:
  Checking for available resources: OK
  Packing application: OK
  Uploading (1K): OK
Push Status: OK
Binding Service [mongodb-92914]: OK
Staging Application: OK
Starting Application: OK
Getting tunnel connection info: OK

Service connection info:
  username : 7344cf16-269e-4572-b1ff-c28f678bed34
  password : c383adb4-c4b8-446e-85bb-8d68278b0737
  name     : db

Starting tunnel to mongodb-92914 on port 10000.
1: none
2: mongo
Which client would you like to start?: 1
Open another shell to run command-line clients or
use a UI tool to connect using the displayed information.
Press Ctrl-C to exit...
```

Using your favorite tools



Alias mongodb-92914

Host localhost Port 10000

User 7344cf16-269... Passwd ••••••••••••••

DB db

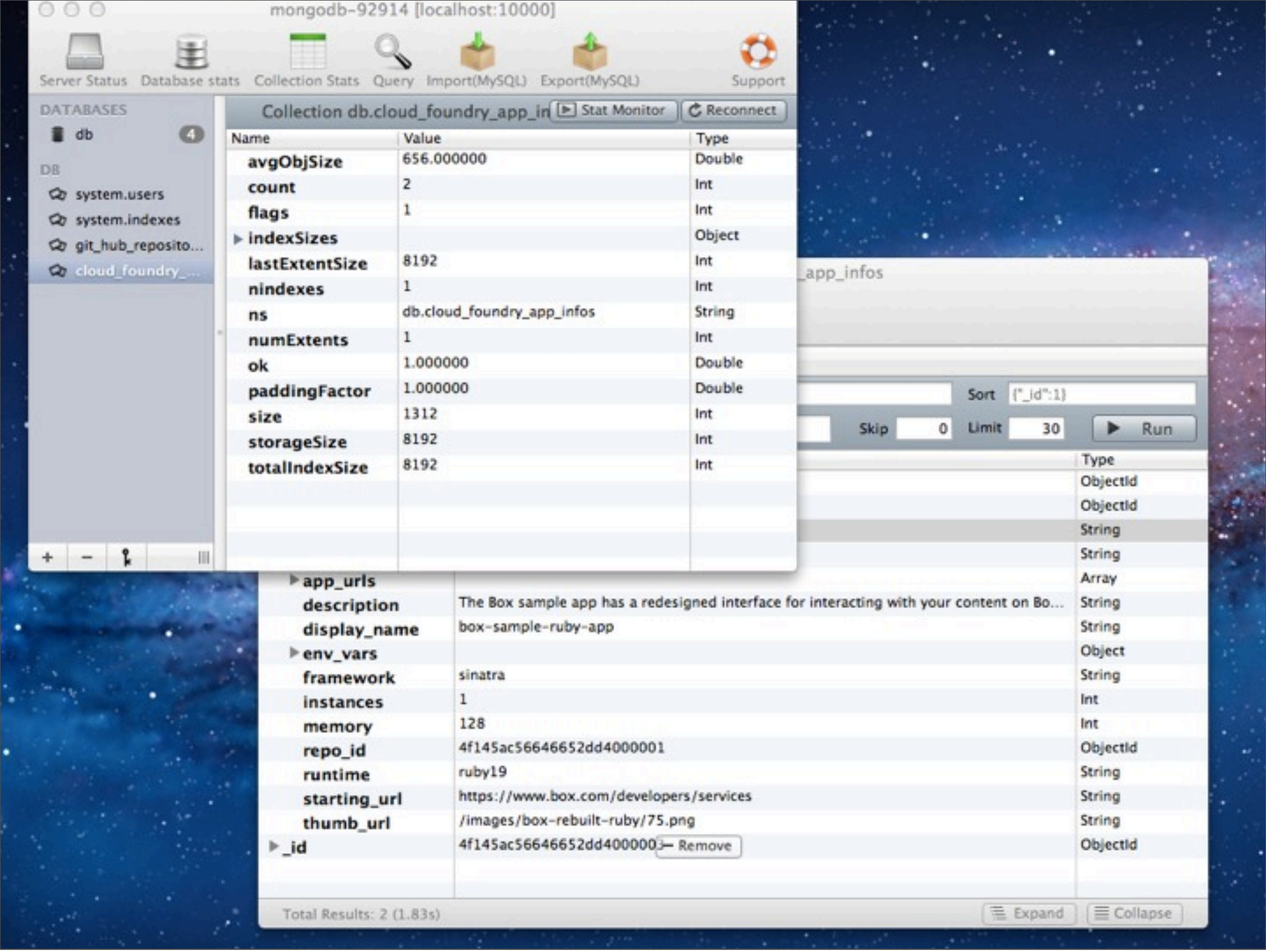
Use Replica Set ☐

Servers host1:port1,host2:port2,host3:p...

Set Name demo_repl

Use SSH Tunnel ☐

```
Installing RDoc docume
Installing RDoc docume
Installing RDoc docume
Installing RDoc docume
Installing RDoc docume
moni-air:developers_cl
Deploying tunnel appli
Create a password: ***
Uploading Application:
  Checking for availab
  Packing application:
  Uploading (1K): OK
```



Name	Value	Type
avgObjSize	656.000000	Double
count	2	Int
flags	1	Int
indexSizes		Object
lastExtentSize	8192	Int
nindexes	1	Int
ns	db.cloud_foundry_app_infos	String
numExtents	1	Int
ok	1.000000	Double
paddingFactor	1.000000	Double
size	1312	Int
storageSize	8192	Int
totalIndexSize	8192	Int

		Type
		ObjectId
		ObjectId
		String
		String
app_urls		Array
description	The Box sample app has a redesigned interface for interacting with your content on Bo...	String
display_name	box-sample-ruby-app	String
env_vars		Object
framework	sinatra	String
instances	1	Int
memory	128	Int
repo_id	4f145ac56646652dd4000001	ObjectId
runtime	ruby19	String
starting_url	https://www.box.com/developers/services	String
thumb_url	/images/box-rebuilt-ruby/75.png	String
_id	4f145ac56646652dd400000 — Remove	ObjectId

Total Results: 2 (1.83s)

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- **Cloud Foundry for Spring Developers**
- Developing NoSQL applications for Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

The Spring framework

- de-facto standard programming model for enterprise Java
- Two million+ developers
- Rapid evolution
  - Spring 1.0 – March 2004
  - Spring 2.0 – October 2006
  - Spring 2.5 – December 2007
  - Spring 3.0 – November 2009
  - Spring 3.1 - December 2011
- **Complete backward compatibility**

Spring's aim:

bring simplicity to java development

web tier & RIA

service tier

batch processing

integration & messaging

data access / NoSQL / Big Data

mobile

The Spring framework

the cloud:

CloudFoundry
Google App Engine
Amazon Web Services

lightweight

tc Server
Tomcat
Jetty

traditional

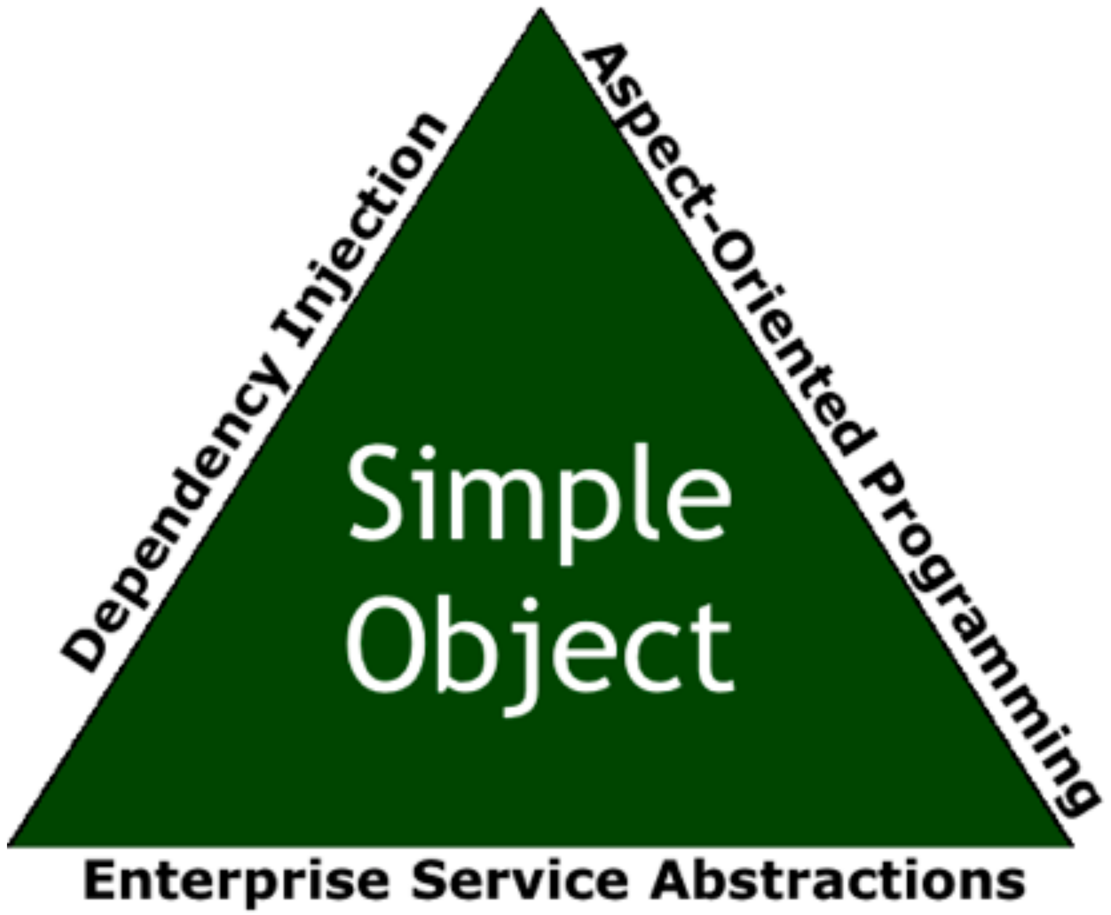
WebSphere
JBoss AS
WebLogic
(on legacy versions, too!)

The Spring Framework

Framework	Description
Spring Core	The foundation
Spring @MVC	the web leading framework (comes with the core framework)
Spring Security	Extensible framework providing authentication, authorization
Spring Webflow	An excellent web framework for building multi-page flows
Spring Web Services	Contract-first, document–centric SOAP and XML web services
Spring Batch	Powerful batch processing framework
Spring Integration	Implements enterprise integration patterns
Spring BlazeDS	Support for Adobe BlazeDS
Spring AMQP	interface with AMQP message brokers, like RabbitMQ
Spring Data	NoSQL options: HBase, MongoDB, Redis, Riak, CouchDB, Neo4J, etc.
Spring Social	integrate Twitter, Facebook, Tripit, MySpace, LinkedIn, etc.
Spring Hadoop	Provides a POJO-centric approach to building Hadoop applications
Spring Mobile, Spring Android	provides first-class support for service creation and consumption for iPhone, Android
Spring GemFire	Provides the easiest interface for the GemFire enterprise data grid technology

At its core, the Spring Framework...

- **Provide comprehensive infrastructural support for developing enterprise Java™ applications**
  - Spring deals with the plumbing
  - So you can focus on solving the domain problem



Spring Has Only One Type of Component: a POJO

- **POJO: Plain ‘Ol Java Object**
  - standard objects
  - objects have dependencies

```
public class CustomerRepository {

      // ‘depends’ on a database connection
      private javax.sql.DataSource dataSource;

}
```

  - objects have lifecycles:

```
Connection conn = ... ;
conn.open() ;
...
conn.close();
```

beginning and
end of the lifecycle

The Spring ApplicationContext

▪ Spring Beans are Managed by An ApplicationContext

- whether you're in an application server, a web server, in regular Java SE application, in the cloud, Spring is initialized through an **ApplicationContext**
- In a Java SE application:

```
ApplicationContext ctx =
   new GenericAnnotationApplicationContext( "com.foo.bar.my.package");
```

- In a web application, you will configure an application context in your web.xml

```
<servlet>
   <servlet-name>Spring Dispatcher Servlet</servlet-name>
   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
   <init-param>
     <param-name>contextConfigLocation</param-name>
     <param-value>/WEB-INF/spring/myAppContext*.xml</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
 </servlet>
```

Auto-Reconfiguration: Getting Started

- Deploy Spring apps to the cloud without changing a single line of code
- Cloud Foundry automatically re-configures bean definitions to bind to cloud services
- Works with Spring and Grails

Auto-Reconfiguration: Relational DB

- **Detects beans of type javax.sql.DataSource**
- **Connects to MySQL or PostgreSQL services**
  - Specifies driver, url, username, password, validation query
- **Creates Commons DBCP or Tomcat DataSource**
- **Replaces existing DataSource**

```
import org.apache.commons.dbcp.BasicDataSource;
...
@Bean(destroyMethod = "close")
public BasicDataSource dataSource(){

   BasicDataSource  bds = new BasicDataSource();
   bds.setUrl( "jdbc:h2:mem");
   bds.setPassword("");
   bds.setUsername("sa");
   bds.setDriverClass( Driver.class);
   return bds;
}
```

Auto-Reconfiguration: ORM

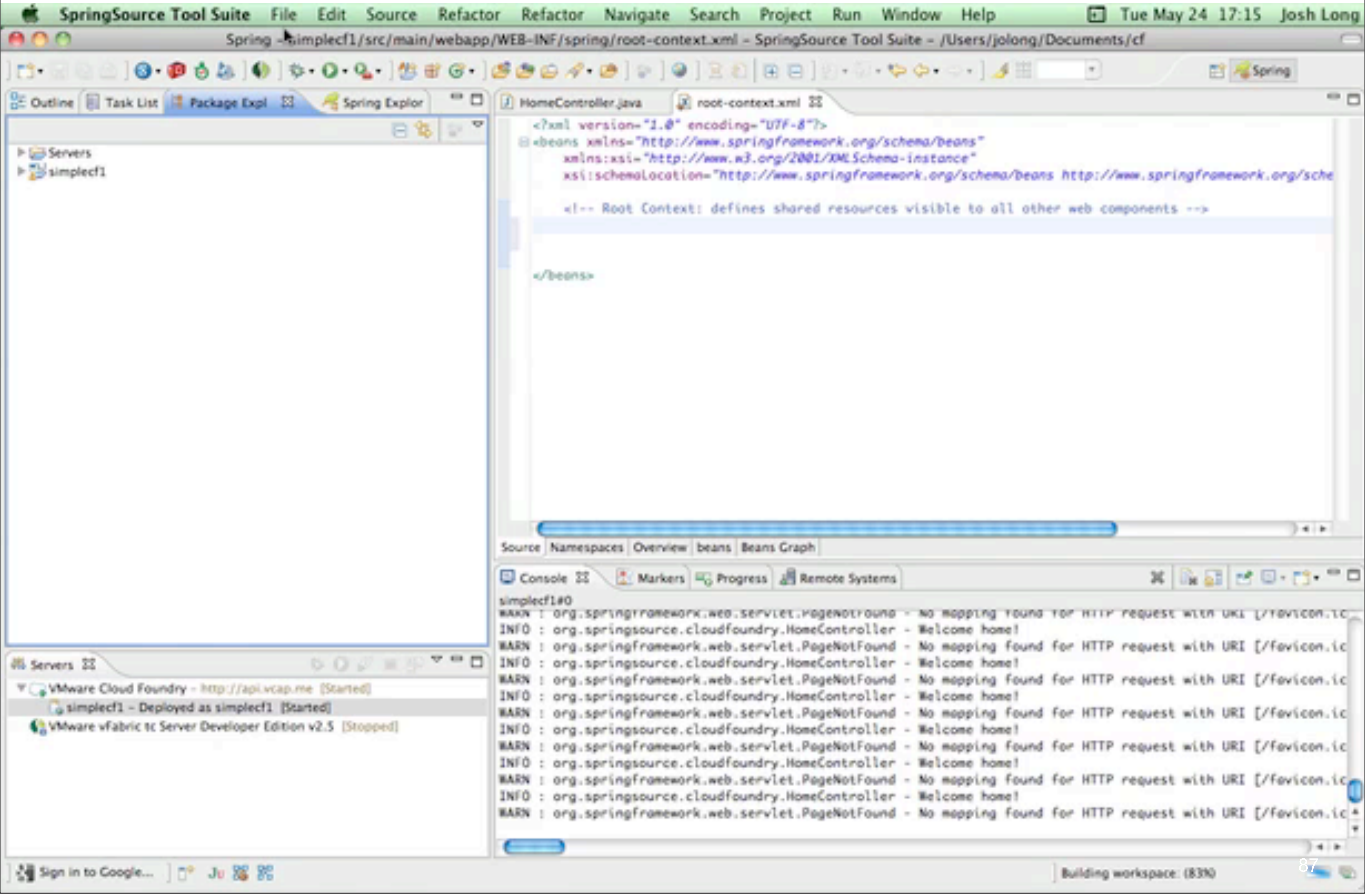
- ***Adjusts* Hibernate Dialect**
- **Changes hibernate.dialect property to MySQLDialect (MyISAM) or PostgreSQLDialect**
  - org.springframework.orm.jpa.AbstractEntityManagerFactoryBean
  - org.springframework.orm.hibernate3.AbstractSessionFactoryBean (Spring 2.5 and 3.0)
  - org.springframework.orm.hibernate3.SessionFactoryBuilderSupport (Spring 3.1)

```
@Bean
public LocalContainerEntityManagerFactoryBean entityManager(){
 LocalContainerEntityManagerFactoryBean lcem =
        new LocalContainerEntityManagerFactoryBean();
 lcem.setDataSource( dataSource() );
 return lcem;
}
```

Auto-Reconfiguration: How It Works

- **Cloud Foundry installs a `BeanFactoryPostProcessor` in your application context during staging**
  - Adds jar to your application
  - Modifies `web.xml` to load BFPP
    - Adds context file to contextConfigLocation
      - web-app `context-param`
      - Spring MVC DispatcherServlet `init-param`
- **Adds PostgreSQL and MySQL driver jars as needed for `DataSource` reconfiguration**

The Spring Developer's Perspective: Auto Reconfiguration



SpringSource Tool Suite File Edit Source Refactor Refactor Navigate Search Project Run Window Help Tue May 24 17:15 Josh Long
Spring - simplecf1/src/main/webapp/WEB-INF/spring/root-context.xml - SpringSource Tool Suite - /Users/jolong/Documents/cf
Outline
Task List
Package Expl
Spring Explor
Servers
simplecf1
HomeController.java
root-context.xml
Source Namespaces Overview beans Beans Graph
Console
Markers
Progress
Remote Systems
Servers
VMware Cloud Foundry - http://api.vcap.me [Started]
simplecf1 - Deployed as simplecf1 [Started]
VMware vFabric tc Server Developer Edition v2.5 [Stopped]
Sign in to Google...
Building workspace: (83%)

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/sche

    <!-- Root Context: defines shared resources visible to all other web components -->



</beans>
```

```
simplecf1#0
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HTTP request with URI [/favicon.ic
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HTTP request with URI [/favicon.ic
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HTTP request with URI [/favicon.ic
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HTTP request with URI [/favicon.ic
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HTTP request with URI [/favicon.ic
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HTTP request with URI [/favicon.ic
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HTTP request with URI [/favicon.ic
```

The Environment

- **Asking Questions**
  - You can introspect the environment variables (`System.getenv(“VCAP_SERVICES”)`), or...
  - import the CloudFoundry runtime API from Java!
    - *(much simpler)*

```
<dependency>
   <groupId>org.cloudfoundry</groupId>
    <artifactId>cloudfoundry-runtime</artifactId>
    <version>0.8.0</version>
  </dependency>
```

The Spring Developer’s Perspective: The Environment

```
@Controller
public class HomeController {

  @RequestMapping(value = "/", method = RequestMethod.GET)
   public String home(Map<String, Object> model) {
     CloudEnvironment cloudEnvironment = new CloudEnvironment();
     if (cloudEnvironment.getCloudApiUri() != null) {
      model.put("host", cloudEnvironment.getInstanceInfo().getHost());
      model.put("port", cloudEnvironment.getInstanceInfo().getPort());
     }
     return "home";
    }

}
```

Giving Your Application Clues with the `env` command

env <appname>

List application environment variables

env-add <appname> <variable[=]value>

Add an environment variable to an application

env-del <appname> <variable>

Delete an environment variable to an application

```
$ env-add html5expenses PAYMENT_GATEWAY=http://blah.com
```

is the same as..

```
$ export PAYMENT_GATEWAY=http://blah.com
```

Introducing... the Cloud Namespace

- **<cloud:> namespace for use in Spring app contexts**
- **Provides application-level control of bean service bindings**
- **Recommended for development of new cloud apps**
- **Use when:**
  - You have multiple services of the same type
  - You have multiple connecting beans of the same type
    - e.g. **DataSource**, **MongoDBFactory**
  - You have custom bean configuration
    - e.g. DataSource pool size, connection properties

<cloud:data-source>

- **Configures a DataSource bean**
  - Commons DBCP or Tomcat DataSource
- **Basic attributes:**
  - id: defaults to service name
  - service-name: only needed if you have multiple relational database services bound to the app

```
<cloud:data-source id="dataSource"/>

<bean class="org.sf.orm.jpa.LocalContainerEntityManagerFactoryBean"
id="entityManagerFactory">
        <property name="dataSource" ref="dataSource"/>
</bean>
```

<cloud:data-source> Example

```
<cloud:data-source id="dataSource" service-name="mySQLSvc">
    <cloud:pool pool-size="1-5"/>
    <cloud:connection properties="charset=utf-8"/>
</cloud:data-source>


...

@Autowired
private DataSource dataSource ;
```

<cloud:properties>

- Exposes basic information about services that can be consumed with Spring's property placeholder support
- Basic attributes:
  - id: the name of the properties bean
- Properties automatically available when deploying Spring 3.1 applications

```
<cloud:properties id="cloudProperties" />
<context:property-placeholder properties-ref="cloudProperties"/>

@Autowired private Environment environment;

@Bean
public ComboPooledDataSource dataSource() throws Exception {
  String user = this.environment.getProperty
        ("cloud.services.mysql.connection.username");
  ComboPooledDataSource cpds  = new  ComboPooledDataSource();
  cpds.setUser(user);
  return cpds;
}
```

Spring 3.1 Environment Abstraction

- **Bean definitions for a specific environment (Profiles)**
  - e.g. development, testing, production
  - Possibly different deployment environments
  - Activate profiles by name
    - spring.profiles.active system property
    - Other means outside deployment unit
    - “default” profile activates if no other profiles specified
- **Custom resolution of placeholders**
  - Dependent on the actual environment
  - Ordered property sources
- **Requires Spring 3.1 (or later)**

Isolating Cloud Foundry Configuration

- **Switch between local, testing and Cloud Foundry deployments with Profiles**
- **“cloud” profile automatically activates on Cloud Foundry**
  - usage of the cloud namespace should occur within the cloud profile block

Isolating Cloud Foundry Configuration

```
<bean class="org.sf.orm.jpa.LocalContainerEntityManagerFactoryBean">
         <property name="dataSource" ref="dataSource"/>
</bean>

<beans profile="cloud">
        <cloud:data-source id="dataSource" />
</beans>

<beans profile="default">
        <bean class="org.a.commons.dbcp.BasicDataSource" id="dataSource">
                <property name="url" value="jdbc:mysql://localhost/my_db" />
        </bean>
</beans>
```

Profile Support: How It Works

- **Cloud Foundry installs a custom ApplicationContextInitializer in your app during staging**
  - Modifies web.xml
    - Adds to `contextInitializerClasses` context-param
- **Adds “`cloud`” as an active profile**
- **Adds a `PropertySource` to the `Environment`**

Java Configuration

- **Alternative to <cloud:*> namespace**
  - Spring Java Configuration
  - Non-Spring apps
- **Programmatic creation of service connection factories**
  - Using `ServiceCreator` and `ServiceInfo` classes
- **Works well with `CloudEnvironment`**
- **Included in `cloudfoundry-runtime` lib**

Java Configuration with Profiles

```
@Configuration
@Profile("local")
public class LocalDataSourceConfiguration {

 @Bean public javax.sql.DataSource dataSource()  { ... }

}


@Configuration
@Profile("cloud")
public class CloudDataSourceConfiguration {

 @Bean public javax.sql.DataSource dataSource()  { ... }

}
```

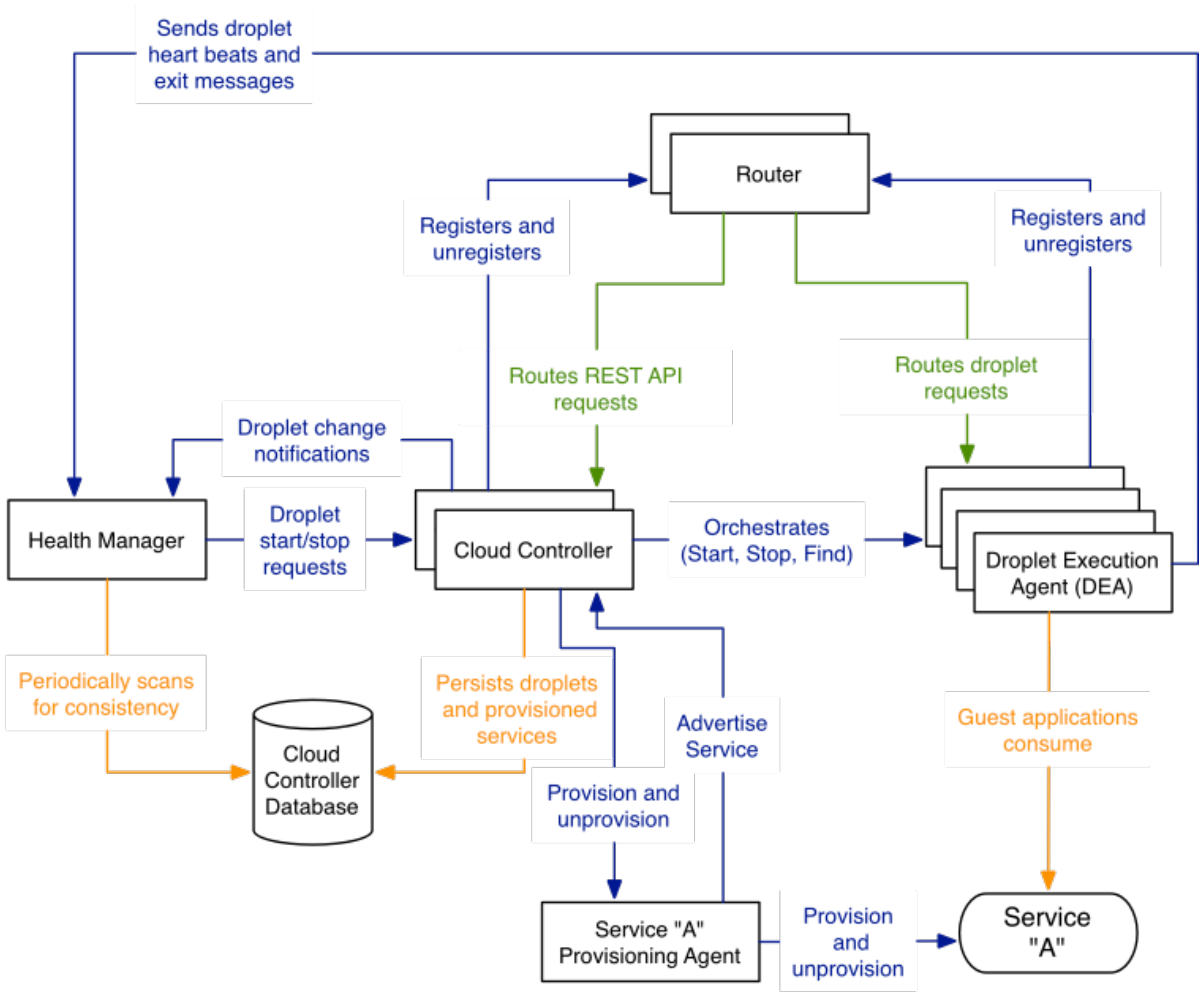
Using ServiceCreator

```
//Provides access to CF service and application env info
CloudEnvironment environment = new CloudEnvironment();

//Retrieve env info for bound service named "mysqlService"
RdbmsServiceInfo mysqlSvc =
        environment.getServiceInfo("mysqlService", RdbmsServiceInfo.class);

//create a DataSource bound to the service
RdbmsServiceCreator dataSourceCreator = new RdbmsServiceCreator();

DataSource dataSource = dataSourceCreator.createService(mysqlSvc);
```

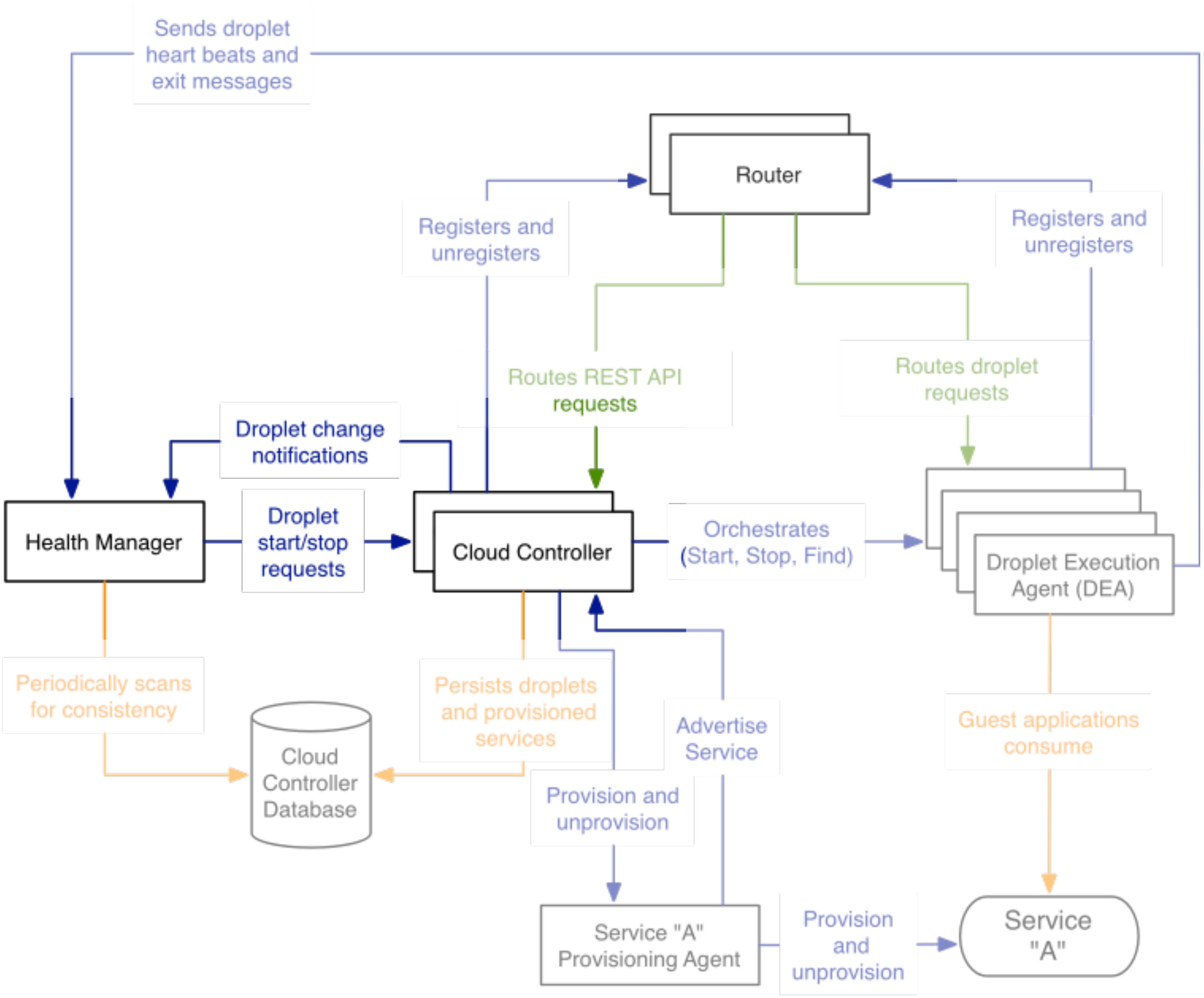
Using ServiceInfo

```
//Provides access to CF service and application env info
CloudEnvironment environment = new CloudEnvironment();

//Retrieve env info for bound service named "mongoService"
MongoServiceInfo mongoSvc =
        environment.getServiceInfo("mongoService", MongoServiceInfo.class);

//create a Mongo DB bound to the service
Mongo mongoDB = new Mongo(mongoSvc.getHost(), mongoSvc.getPort());
```

Cloud Foundry Internal view


Sends droplet heart beats and exit messages
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Droplet change notifications
Health Manager
Droplet start/stop requests
Cloud Controller
Orchestrates (Start, Stop, Find)
Droplet Execution Agent (DEA)
Periodically scans for consistency
Persists droplets and provisioned services
Advertise Service
Guest applications consume
Cloud Controller Database
Provision and unprovision
Service "A" Provisioning Agent
Provision and unprovision
Service "A"

Cloud Foundry Internal view


Sends droplet heart beats and exit messages
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Droplet change notifications
Health Manager
Droplet start/stop requests
Cloud Controller
Orchestrates (Start, Stop, Find)
Droplet Execution Agent (DEA)
Periodically scans for consistency
Cloud Controller Database
Persists droplets and provisioned services
Advertise Service
Provision and unprovision
Guest applications consume
Service "A" Provisioning Agent
Provision and unprovision
Service "A"

Cloud Foundry: Cloud Controller

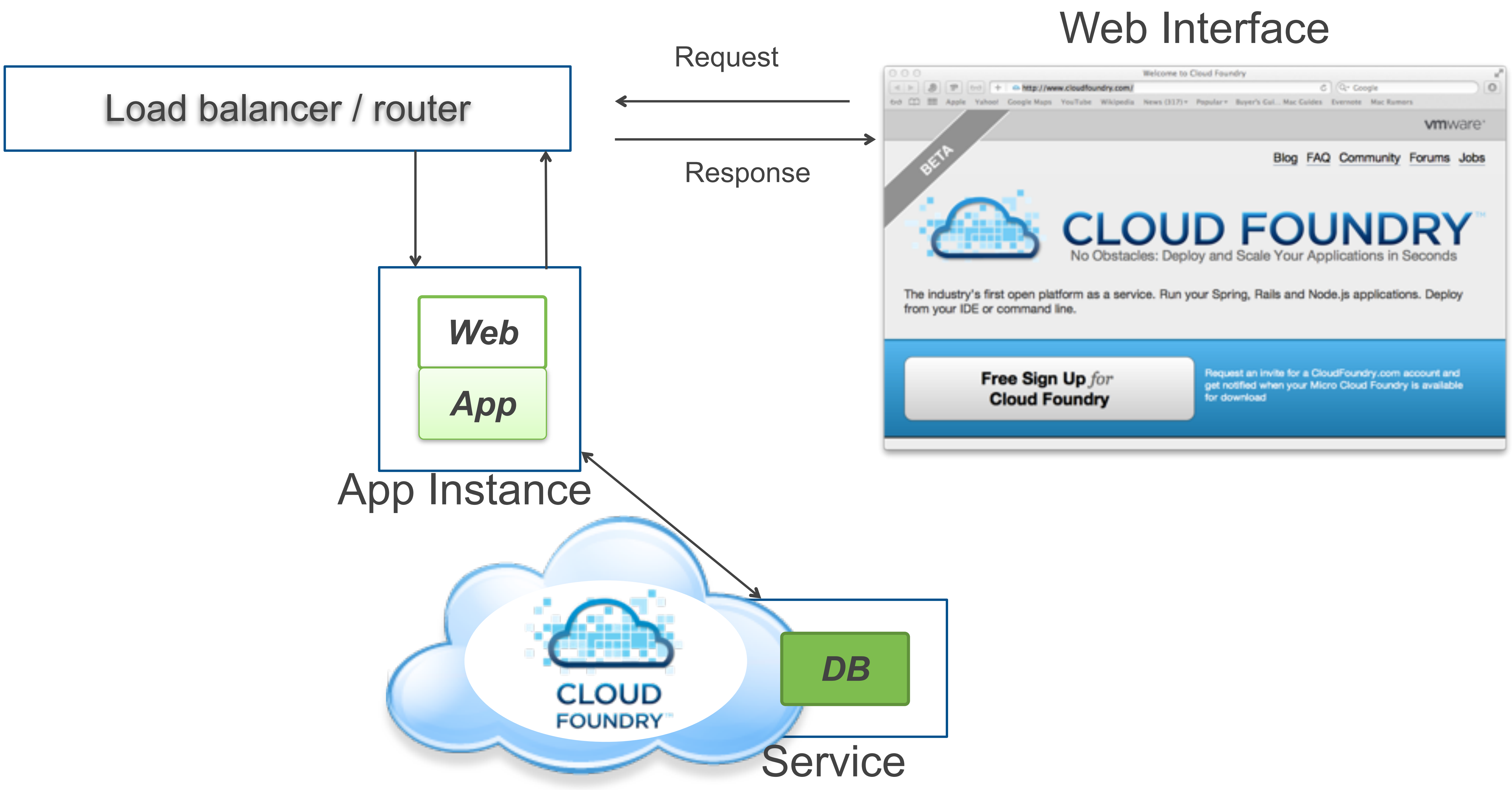
- **It is responsible for all state changes in the system**
  - Ensuring all dependencies are available
  - Binding the application to services
- **Anything that effects users, apps, or services is controlled by the Cloud Controllers**
  - Examples : `vmc push`, `vmc instances`, `vmc create-service`, etc. are driven by the Cloud Controller
- **Once staged, the Cloud Controller is responsible for connecting the application to a DEA execution unit**

Cloud Foundry: Health Manager

- Health manager reconciles world view of cloud controller
- puts “sick” or inconsistent parts of cloud into “flapping” state

Cloud Foundry: Router

- routes requests to REST API to a cloud controller
- route from URIs to applications
- load balancer



No Obstacles: Deploy and Scale Your Applications in Seconds

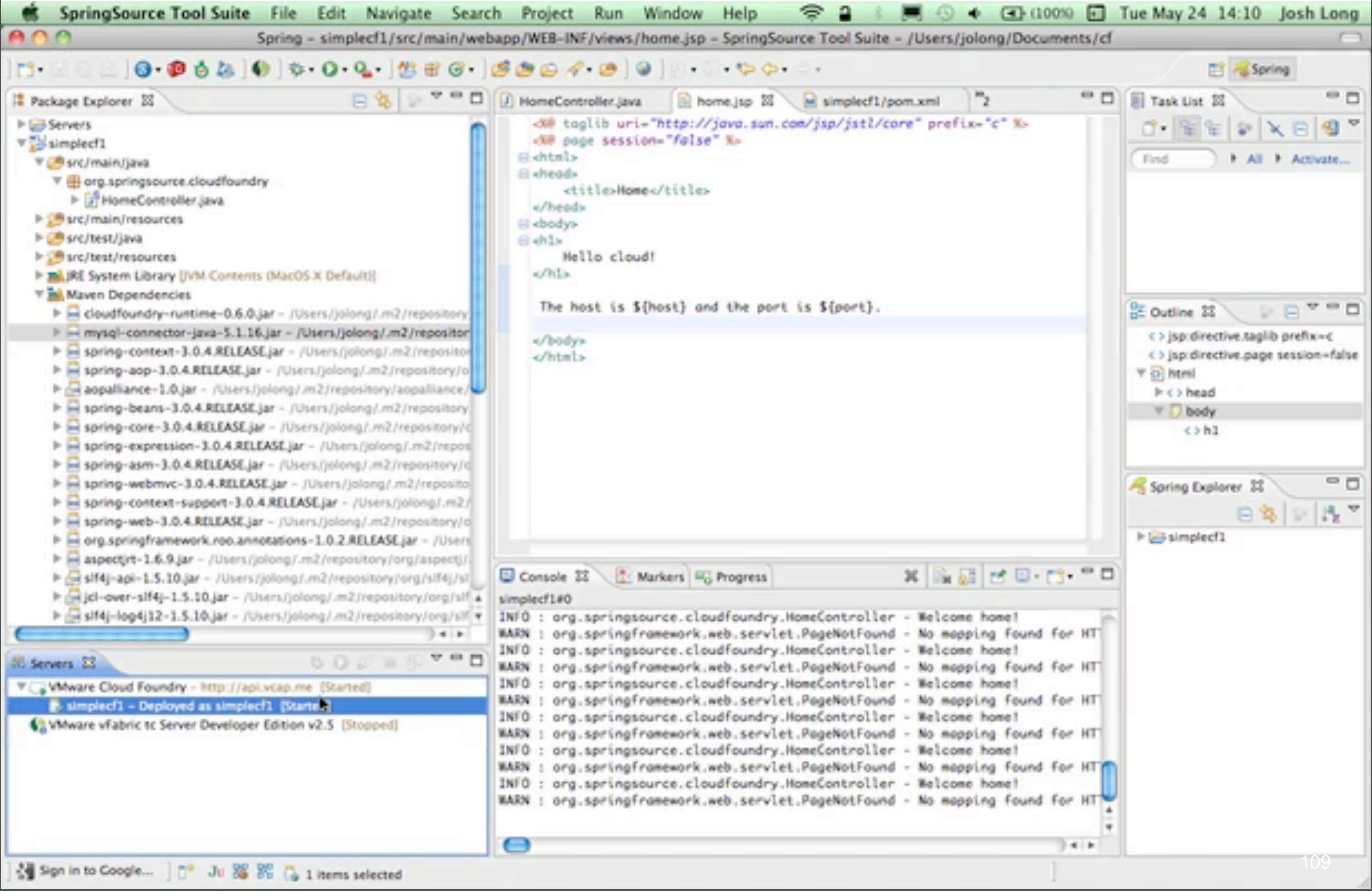
The industry's first open platform as a service. Run your Spring, Rails and Node.js applications. Deploy from your IDE or command line.

Request an invite for a CloudFoundry.com account and get notified when your Micro Cloud Foundry is available for download

Cloud Foundry: Router

- **Divides work across configured application instances (round robin)**
- **Features session affinity, or “sticky sessions”**
  - a request to a web endpoint that uses a session will be pinned to the original server of the request on subsequent requests
- **there is NO session state failover**
  - don’t put business data in the session
  - promote critical process state to a fast in-RAM store like Redis
    - (which Cloud Foundry supports!)

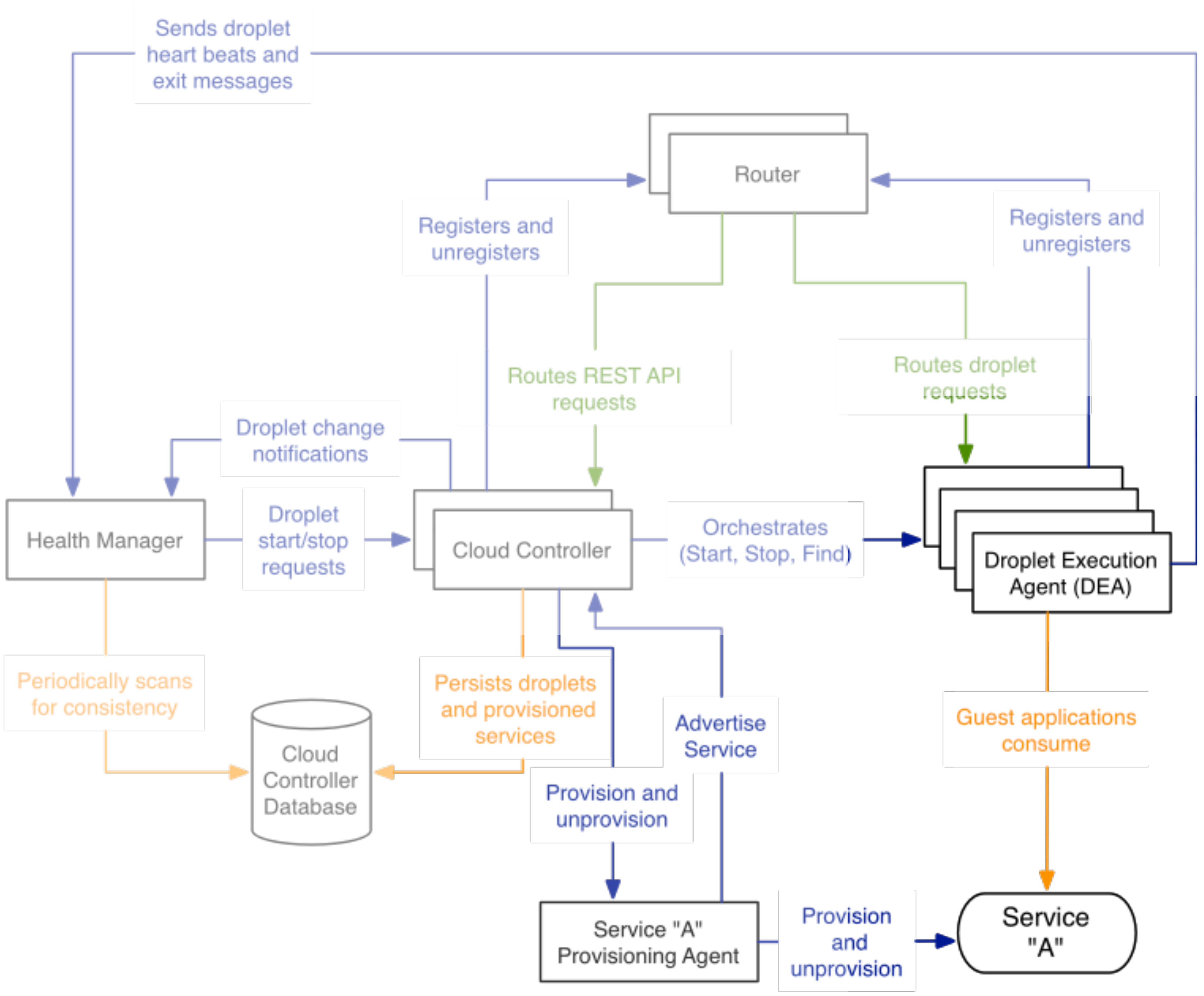
Cloud Foundry: Scaling Up and Down with the Router


SpringSource Tool Suite File Edit Navigate Search Project Run Window Help (100%) Tue May 24 14:10 Josh Long
Spring - simplecf1/src/main/webapp/WEB-INF/views/home.jsp - SpringSource Tool Suite - /Users/jolong/Documents/cf
Spring
Package Explorer
Servers
simplecf1
src/main/java
org.springsource.cloudfoundry
HomeController.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JVM Contents (MacOS X Default)]
Maven Dependencies
cloudfoundry-runtime-0.6.0.jar - /Users/jolong/.m2/repository
mysql-connector-java-5.1.16.jar - /Users/jolong/.m2/reposito
spring-context-3.0.4.RELEASE.jar - /Users/jolong/.m2/reposito
spring-aop-3.0.4.RELEASE.jar - /Users/jolong/.m2/repository/o
aopalliance-1.0.jar - /Users/jolong/.m2/repository/aopalliance/
spring-beans-3.0.4.RELEASE.jar - /Users/jolong/.m2/repository
spring-core-3.0.4.RELEASE.jar - /Users/jolong/.m2/repository/
spring-expression-3.0.4.RELEASE.jar - /Users/jolong/.m2/repos
spring-asm-3.0.4.RELEASE.jar - /Users/jolong/.m2/repository/
spring-webmvc-3.0.4.RELEASE.jar - /Users/jolong/.m2/reposito
spring-context-support-3.0.4.RELEASE.jar - /Users/jolong/.m2/
spring-web-3.0.4.RELEASE.jar - /Users/jolong/.m2/repository/
org.springframework.roo.annotations-1.0.2.RELEASE.jar - /Users
aspectjrt-1.6.9.jar - /Users/jolong/.m2/repository/org/aspectj/
slf4j-api-1.5.10.jar - /Users/jolong/.m2/repository/org/slf4j/sl
jcl-over-slf4j-1.5.10.jar - /Users/jolong/.m2/repository/org/slf
slf4j-log4j12-1.5.10.jar - /Users/jolong/.m2/repository/org/slf
Servers
VMware Cloud Foundry - http://api.vcap.me [Started]
simplecf1 - Deployed as simplecf1 [Starte
VMware vFabric tc Server Developer Edition v2.5 [Stopped]
HomeController.java
home.jsp
simplecf1/pom.xml
Task List
Find
All
Activate...
Outline
jsp:directive.taglib prefix=c
jsp:directive.page session=false
html
head
body
h1
Spring Explorer
simplecf1
Console
Markers
Progress
Sign in to Google...
1 items selected

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
    <title>Home</title>
</head>
<body>
<h1>
    Hello cloud!
</h1>

 The host is ${host} and the port is ${port}.

</body>
</html>
```

```
simplecf1#0
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HT
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HT
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HT
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HT
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HT
INFO : org.springsource.cloudfoundry.HomeController - Welcome home!
WARN : org.springframework.web.servlet.PageNotFound - No mapping found for HT
```

Cloud Foundry Internal view


Sends droplet heart beats and exit messages
Router
Registers and unregisters
Registers and unregisters
Routes REST API requests
Routes droplet requests
Droplet change notifications
Health Manager
Droplet start/stop requests
Cloud Controller
Orchestrates (Start, Stop, Find)
Droplet Execution Agent (DEA)
Periodically scans for consistency
Cloud Controller Database
Persists droplets and provisioned services
Advertise Service
Provision and unprovision
Guest applications consume
Service "A" Provisioning Agent
Provision and unprovision
Service "A"

Cloud Foundry: DEA

- The system maintains a pool of standby DEAs and these act as the VM-level container for an application
- DEAs support both single and multi-tenant operation (1 app per DEA VM, or n apps per DEA VM)
- DEAs provide a secure/constrained OS environment running the application’s app-server and the application code

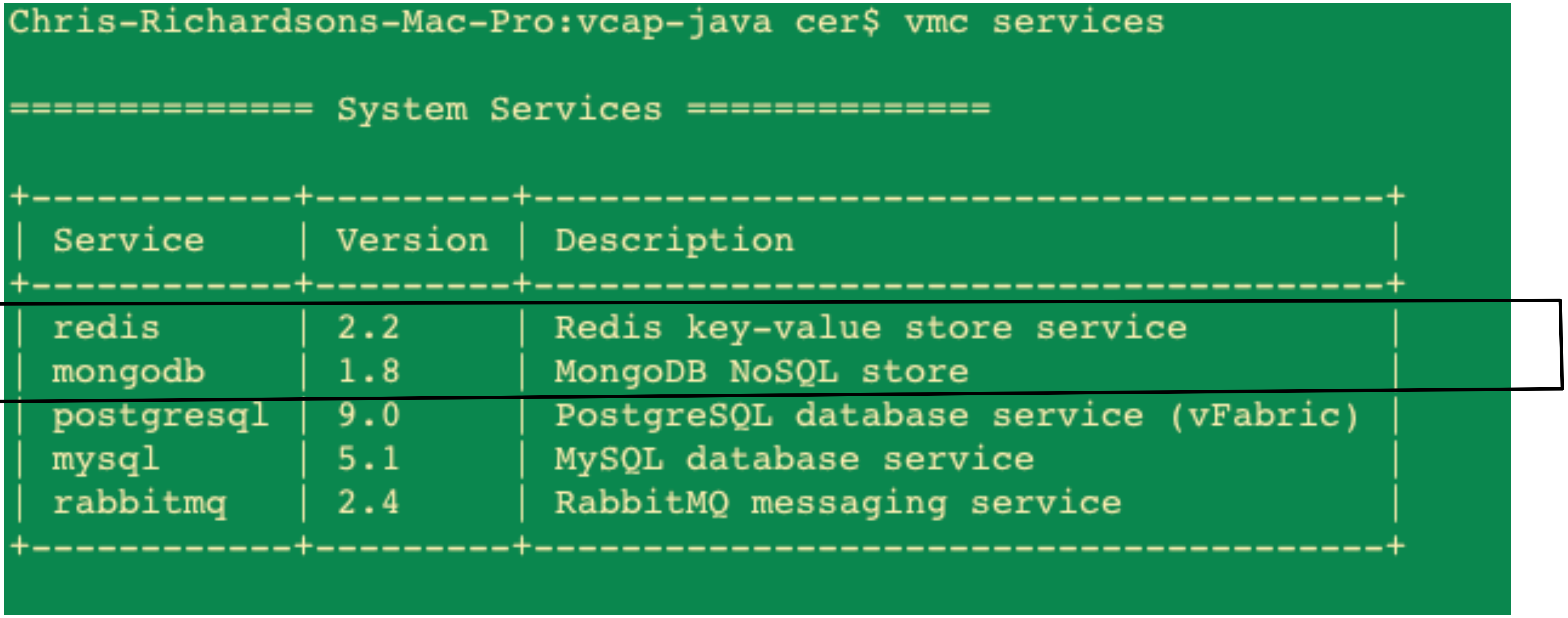
Cloud Foundry: DEA

- **If an application instance crashes**
  - DEA detects unexpected exit, DEA broadcasts message
  - Routers remove instance from routing
  - Health manager notifies cloud controller
- **If a DEA VM crashes**
  - Application instances become unavailable
  - Health manager notices the missing instances and notifies the cloud controller
  - cloud controller requests application instances to be started
  - existing DEA will reply and start the applications

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- **Developing NoSQL applications for Cloud Foundry**
  - **Why NoSQL?**
  - Overview of NoSQL databases
  - Introduction to Spring Data
  - Using Spring Data for Redis
  - Using Spring Data for Mongo
  - Deploying on Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

Cloud Foundry provides NoSQL-aaS



```
Chris-Richardsons-Mac-Pro:vcap-java cer$ vmc services

============== System Services ==============

+------------+---------+---------------------------------------+
| Service    | Version | Description                           |
+------------+---------+---------------------------------------+
| redis      | 2.2     | Redis key-value store service         |
| mongodb    | 1.8     | MongoDB NoSQL store                   |
| postgresql | 9.0     | PostgreSQL database service (vFabric) |
| mysql      | 5.1     | MySQL database service                |
| rabbitmq   | 2.4     | RabbitMQ messaging service            |
+------------+---------+---------------------------------------+
```

But what's a NoSQL database?

Why would you want to use it?

How do you use it?

Relational databases are great...

- **SQL**
  - High-level
  - Sorting
  - Aggregation
- **ACID semantics**
- **Well supported**
  - JDBC
  - Hibernate/JPA
  - Spring
- **Well understood**
  - Developers
  - Operators

... but they have limitations

- Object/relational impedance mismatch
- Complicated to map rich domain model to relational schema
- Difficult to handle semi-structured data, e.g. varying attributes
- Schema changes
- Extremely difficult/impossible to scale
- Poor performance for some use cases

Solution: Spend Money

http://upload.wikimedia.org/wikipedia/commons/e/e5/Rising_Sun_Yacht.JPG

OR

http://www.trekbikes.com/us/en/bikes/road/race_performance/madone_5_series/madone_5_2/#

Solution: Use NoSQL

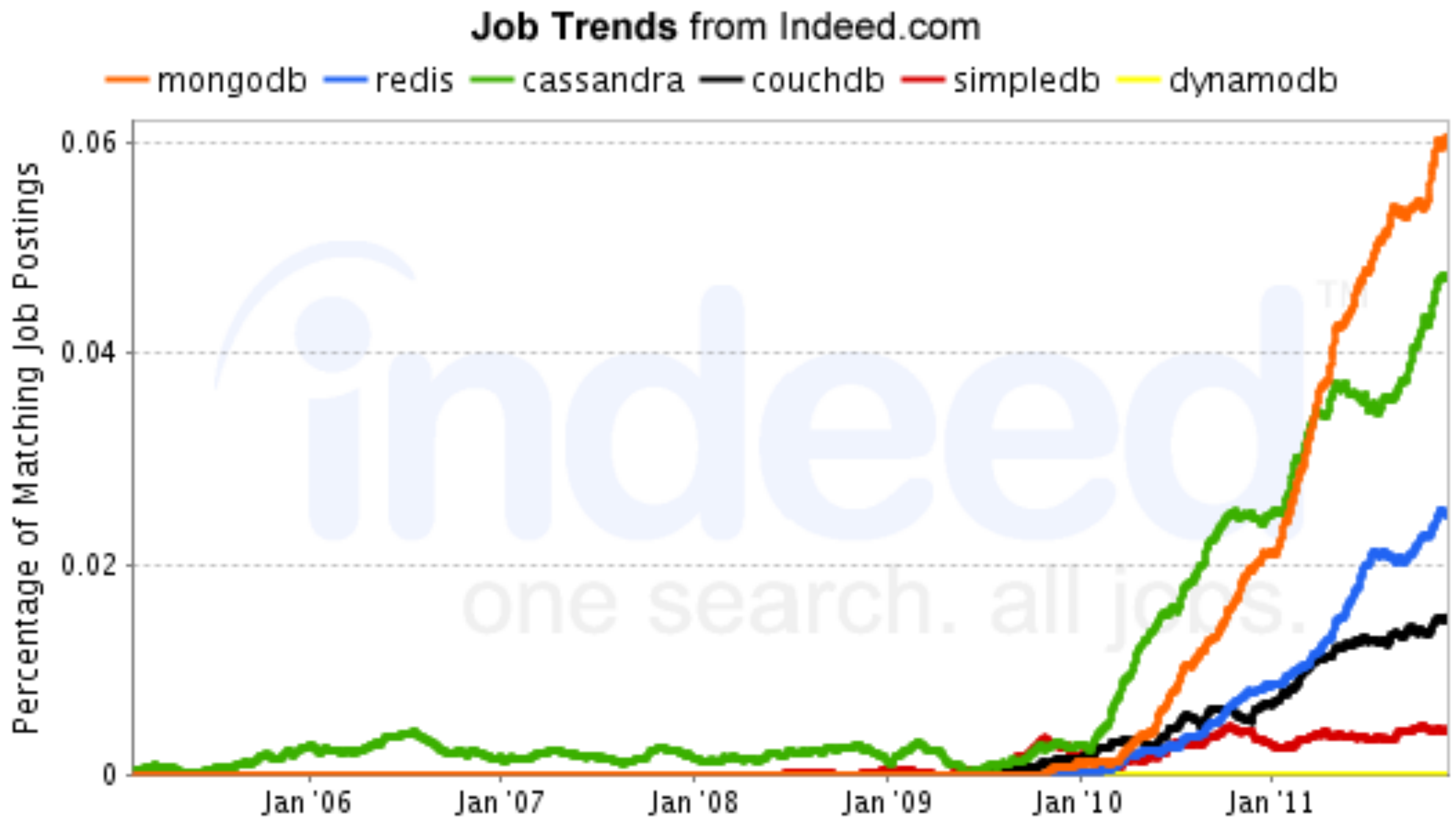
Benefits

- Higher performance
- Higher scalability
- Richer data-model
- Schema-less

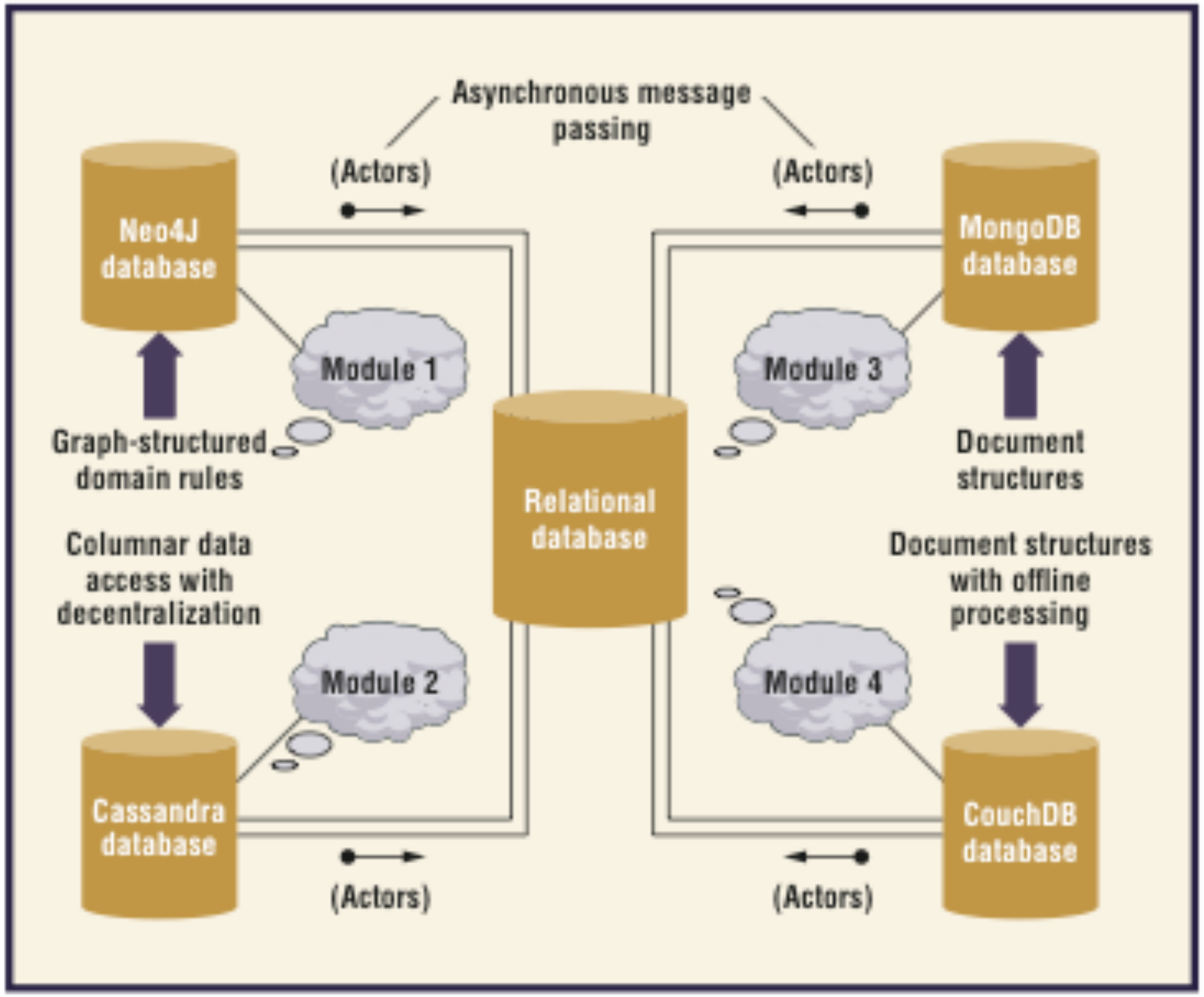
Drawbacks

- Limited transactions
- Relaxed consistency
- Unconstrained data

Growing in popularity…



Future = multi-paradigm data storage for enterprise applications



IEEE Software Sept/October 2010 - Debasish Ghosh / Twitter @debasishg

e.g. Netflix
- RDBMS
- SimpleDB
- Cassandra
- Hadoop/Hbase

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- **Developing NoSQL applications for Cloud Foundry**
  - Why NoSQL?
  - **Overview of NoSQL databases**
  - Introduction to Spring Data
  - Using Spring Data for Redis
  - Using Spring Data for Mongo
  - Deploying on Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

Redis

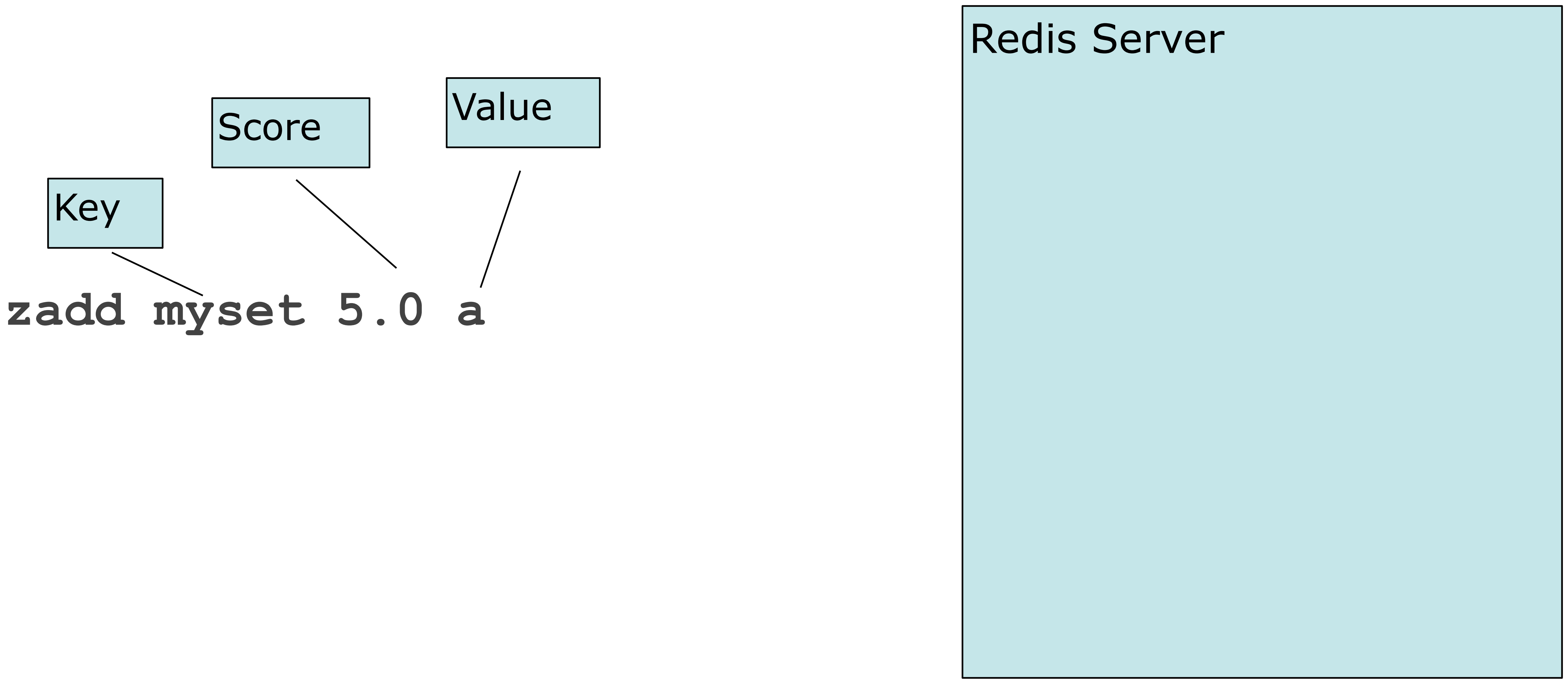
- Advanced key-value store
- Very fast
- Optional persistence
- Transactions with optimistic locking
- Master-slave replication
- Sharding using client-side consistent hashing

Redis

- Advanced key-value store
- Very fast
- Optional persistence
- Transactions with optimistic locking
- Master-slave replication
- Sharding using client-side consistent hashing

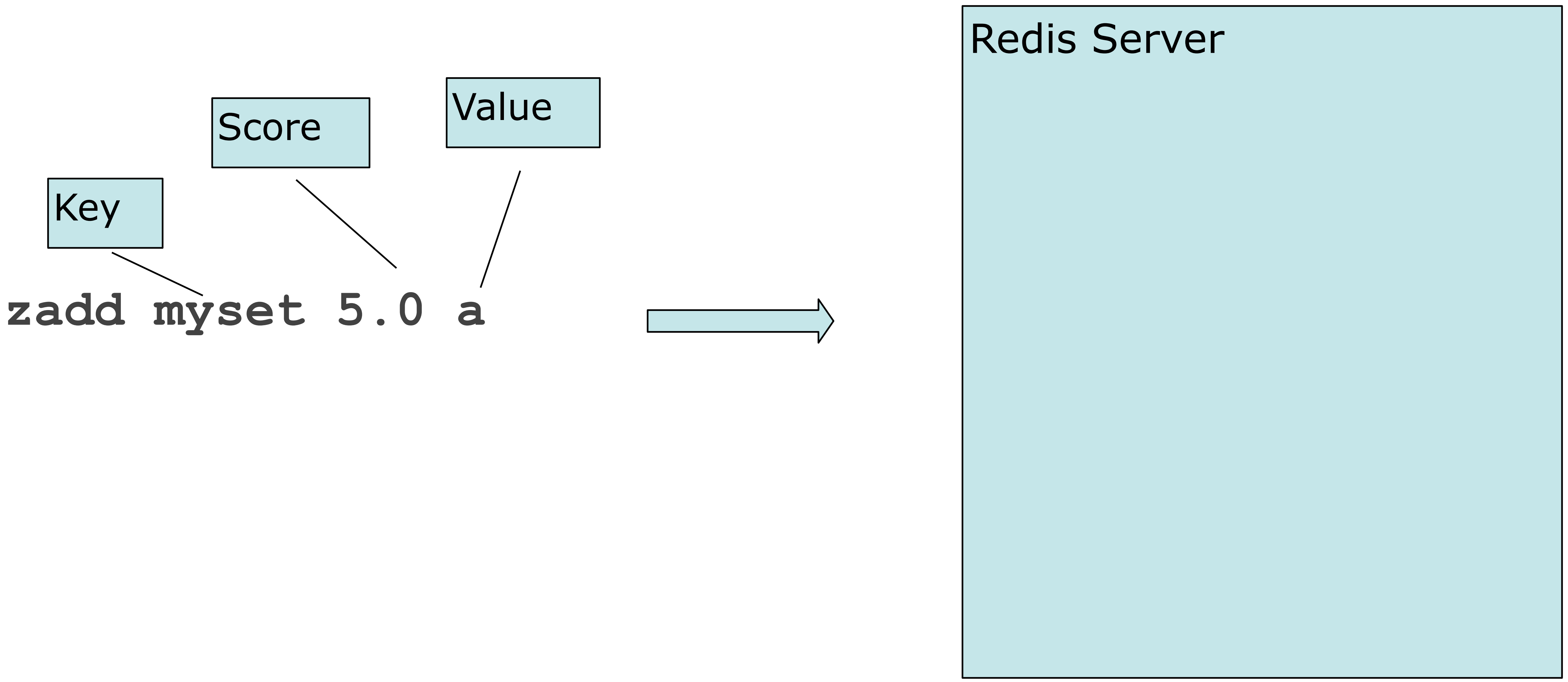
K1	V1
K2	V2
K3	V2

Adding members to a sorted set



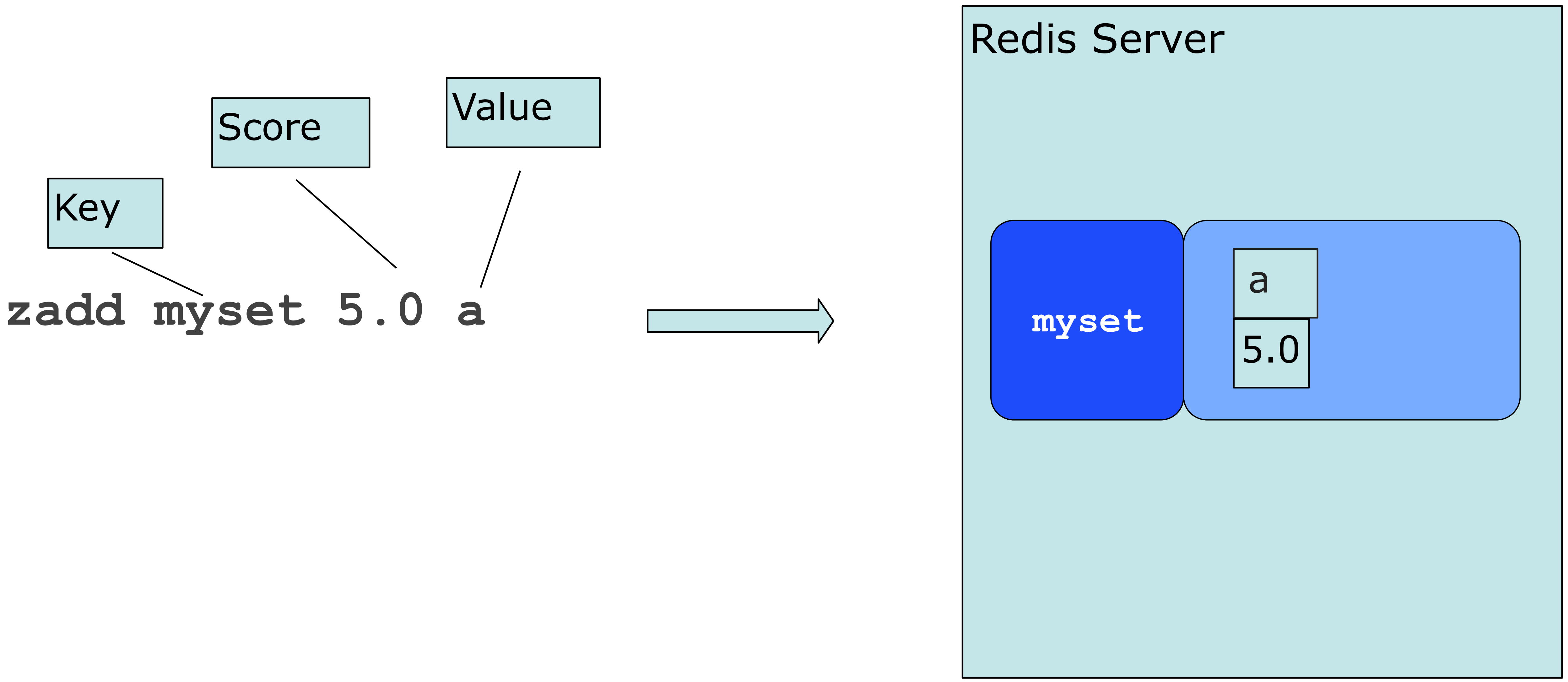
```
zadd myset 5.0 a
```

Adding members to a sorted set



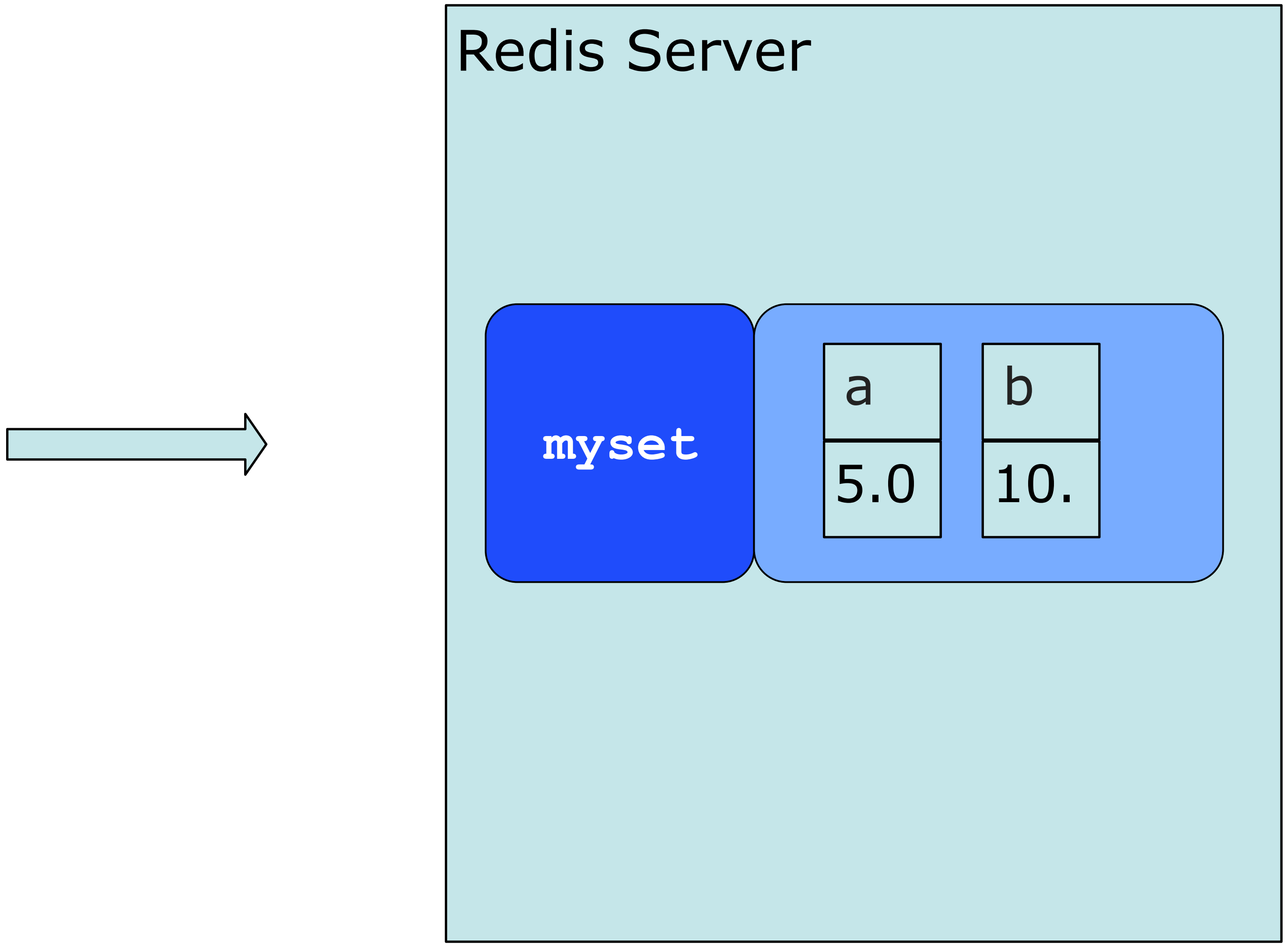
```
zadd myset 5.0 a
```

Adding members to a sorted set



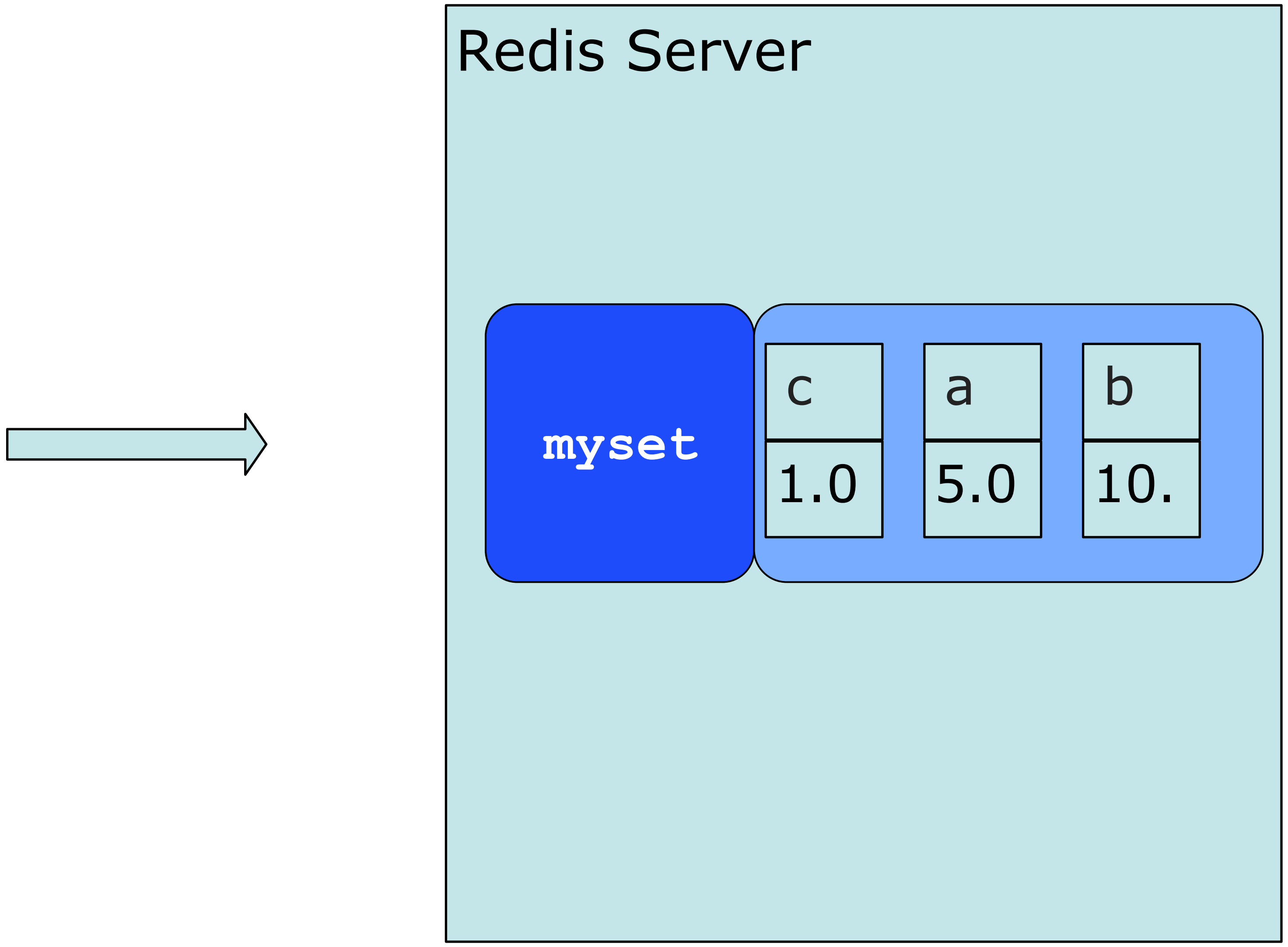
Adding members to a sorted set

```
zadd myset 10.0 b
```

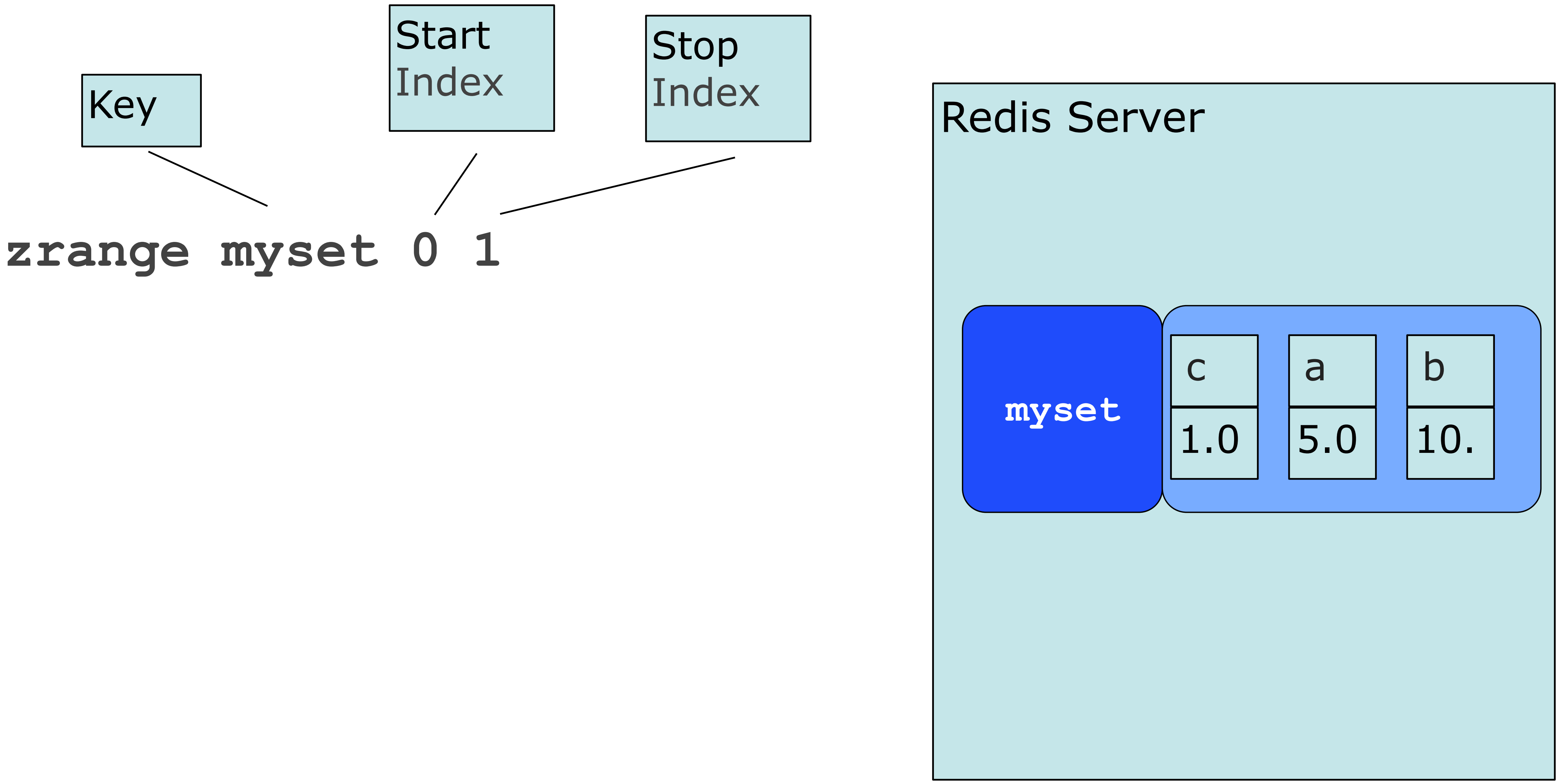


Adding members to a sorted set

```
zadd myset 1.0 c
```

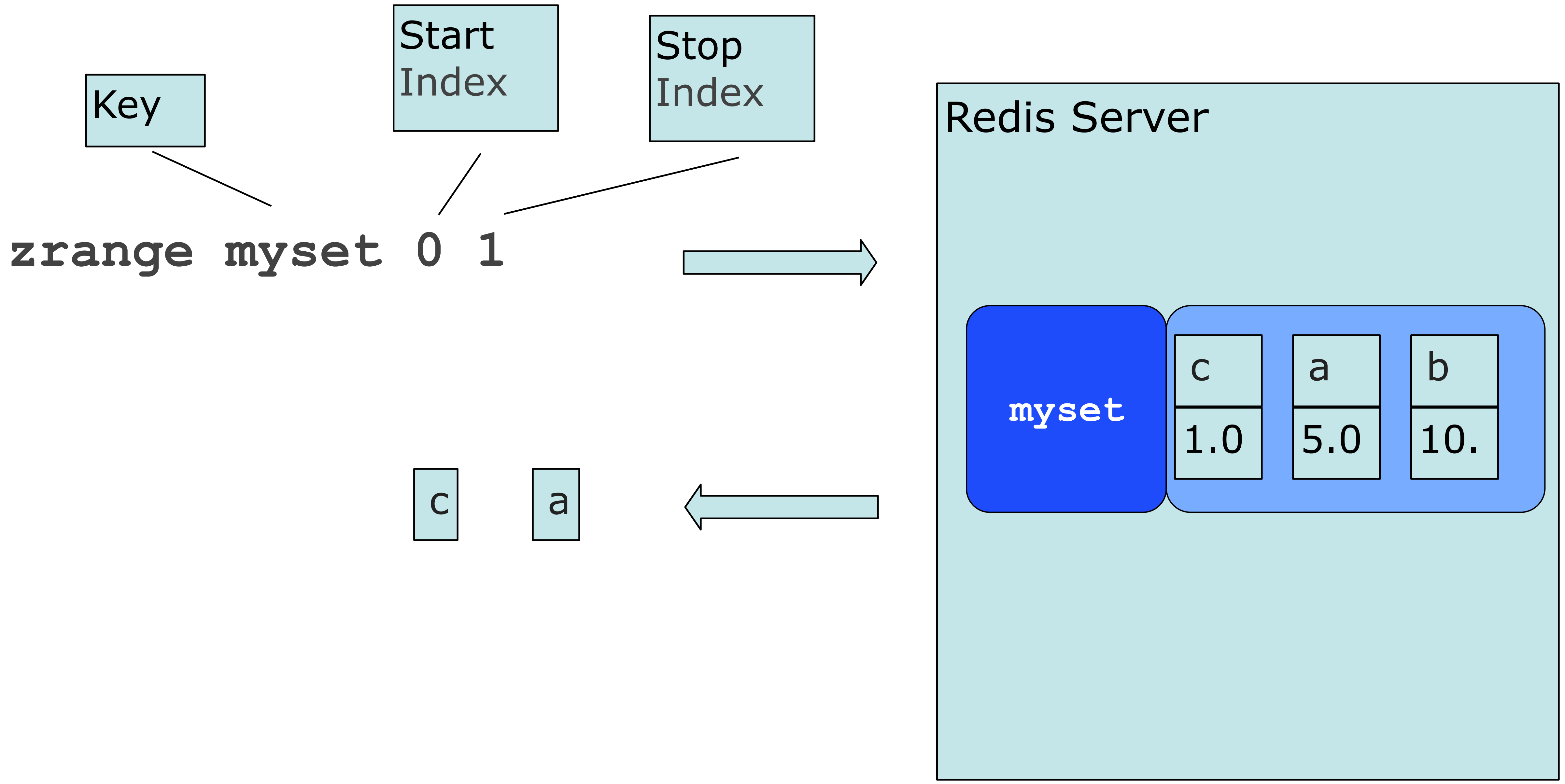


Retrieving members by index range

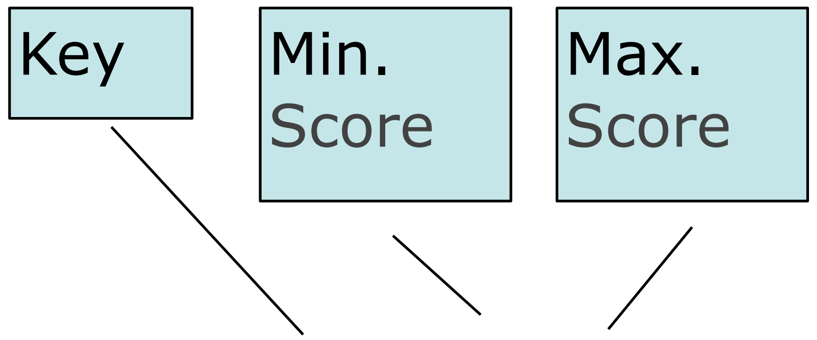


```
zrange myset 0 1
```

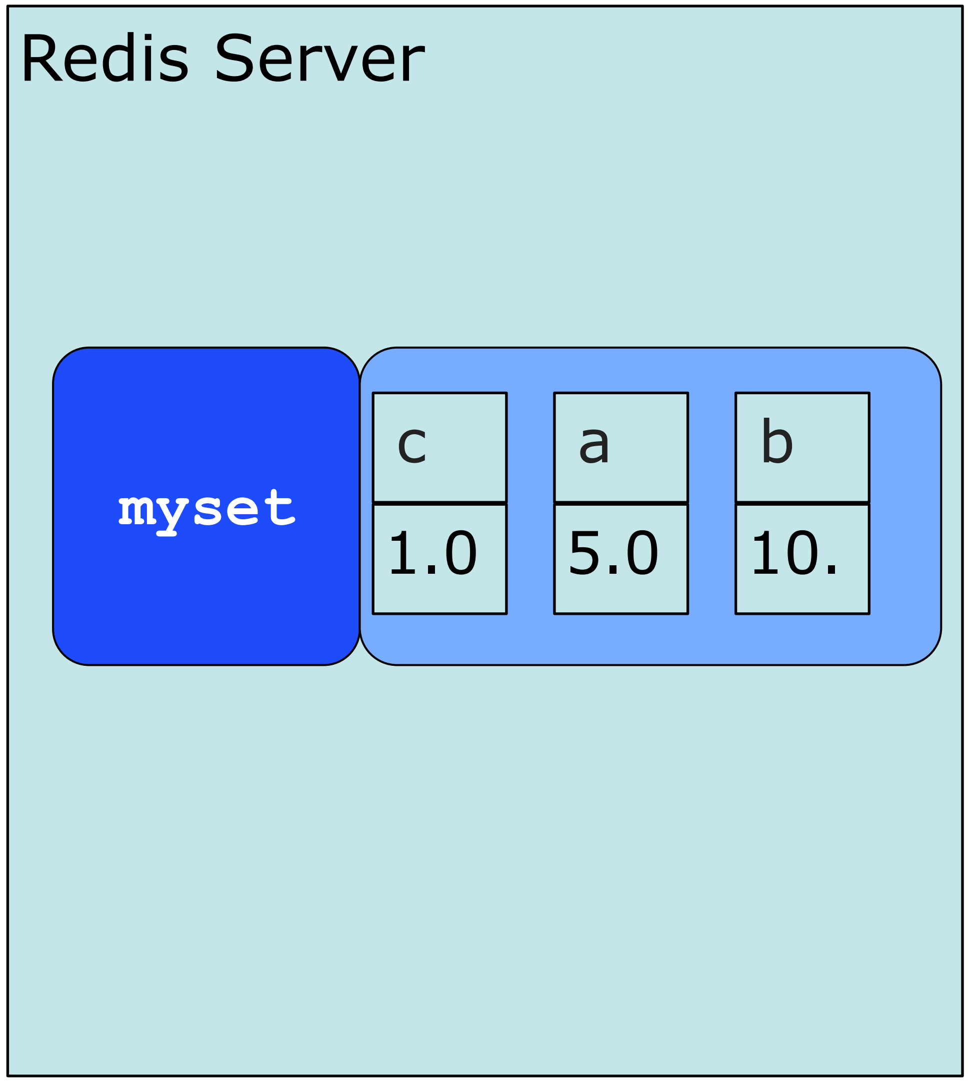
Retrieving members by index range



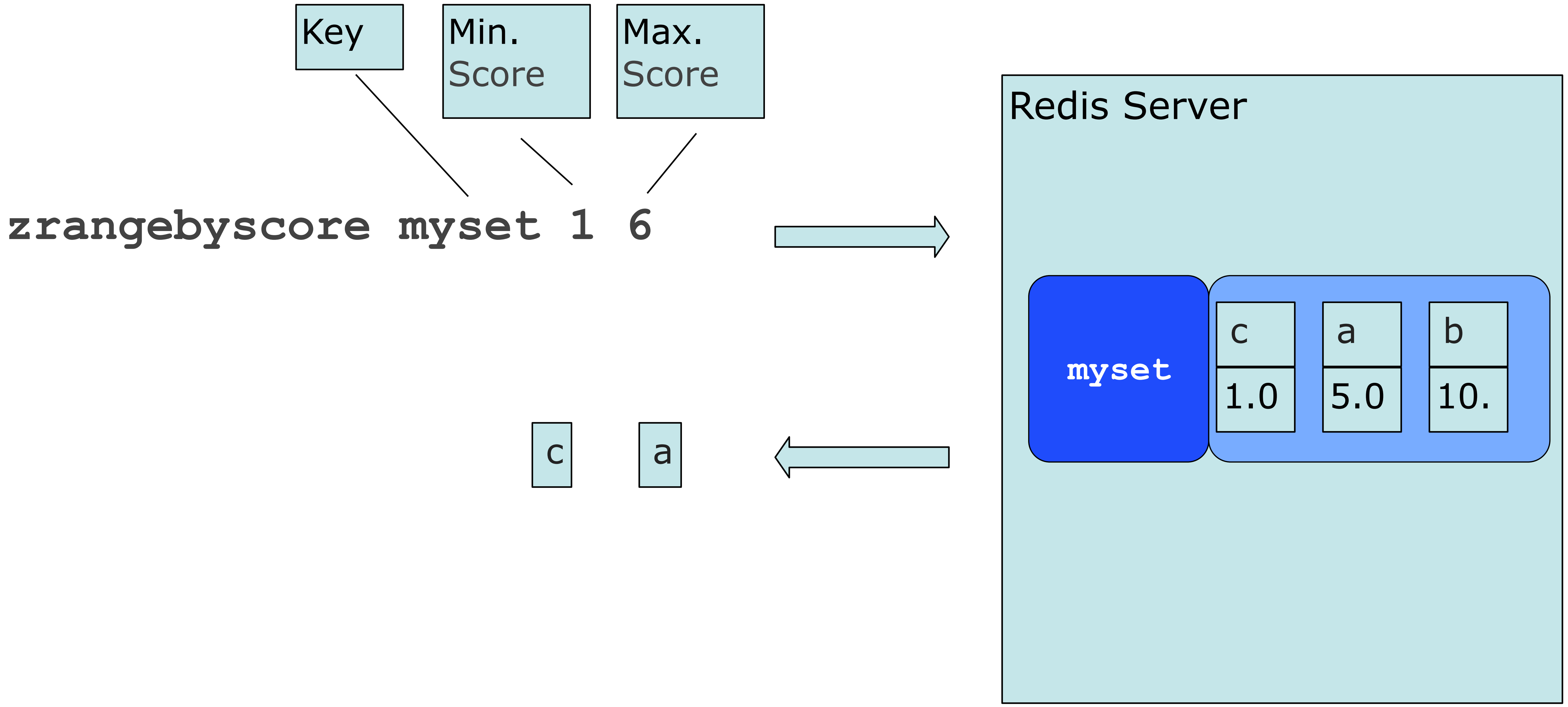
Retrieving members by score



```
zrangebyscore myset 1 6
```



Retrieving members by score



```
zrangebyscore myset 1 6
```

Redis use cases

- **Drop-in replacement for Memcached**
  - Session state
  - Cache of data retrieved from SOR
- **Replica of SOR for queries needing high-performance**
- **Handling tasks that overload an RDBMS**
  - Hit counts - INCR
  - Most recent N items - LPUSH and LTRIM
  - Randomly selecting an item – SRANDMEMBER
  - Queuing – Lists with LPOP, RPUSH, ….
  - High score tables – Sorted sets and ZINCRBY
  - …

- **Notable users: github, guardian.co.uk, ….**

MongoDB

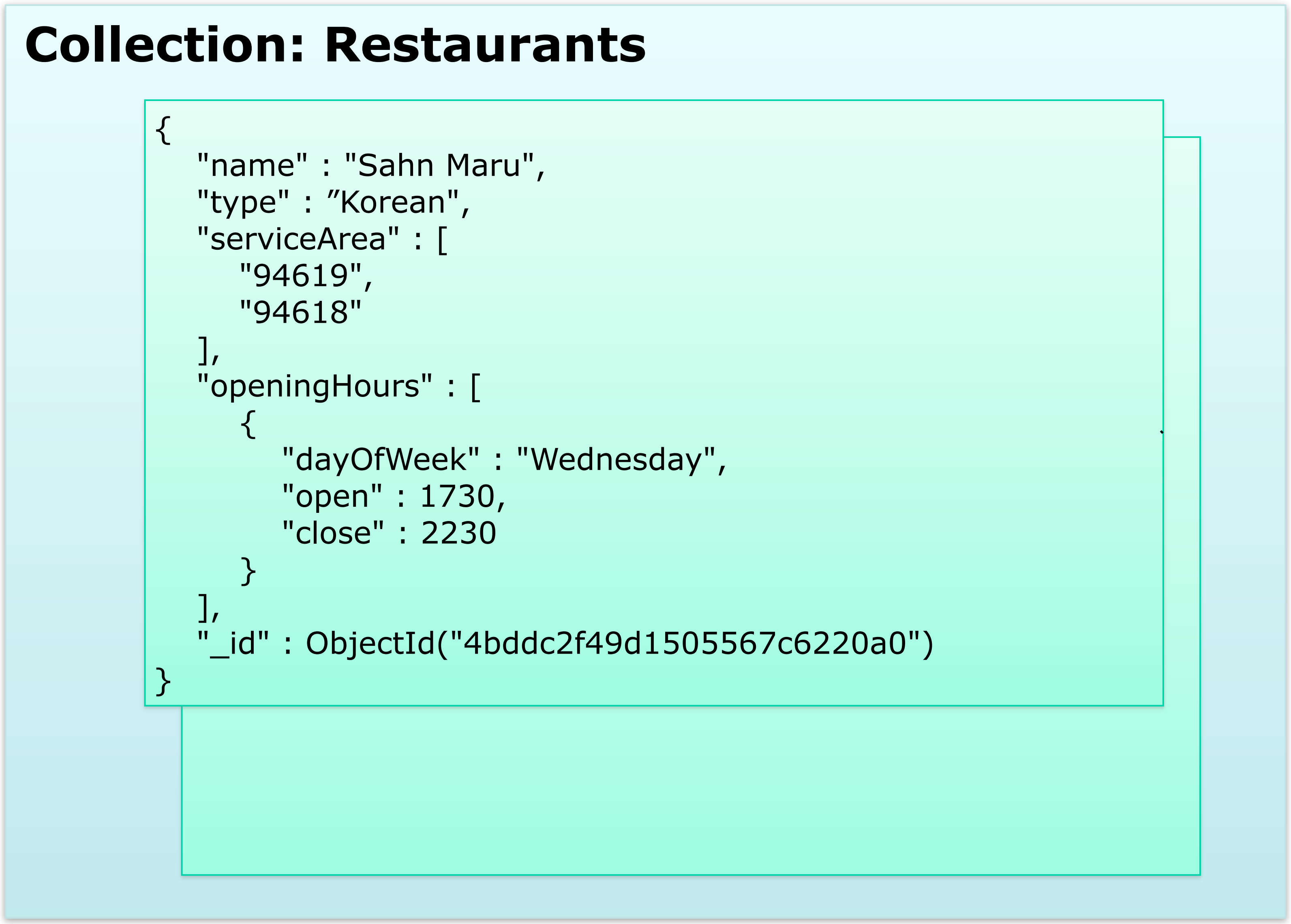
- **Document-oriented database**
  - JSON-style documents: Lists, Maps, primitives
  - Schema-less
- **Transaction = update of a single document**
- **Rich query language for dynamic queries**
- **Very fast**
- **Writes are asynchronous!**
- **Highly scalable and available**

Data model = Binary JSON documents

```
{
   "name" : "Sahn Maru",
   "type" : ”Korean",
   "serviceArea" : [
      "94619",
      "94618"
   ],
   "openingHours" : [
      {
         "dayOfWeek" : "Wednesday",
         "open" : 1730,
         "close" : 2230
      }
   ],
   "_id" : ObjectId("4bddc2f49d1505567c6220a0")
}
```

Sequence of bytes on disk ➔ fast i/o

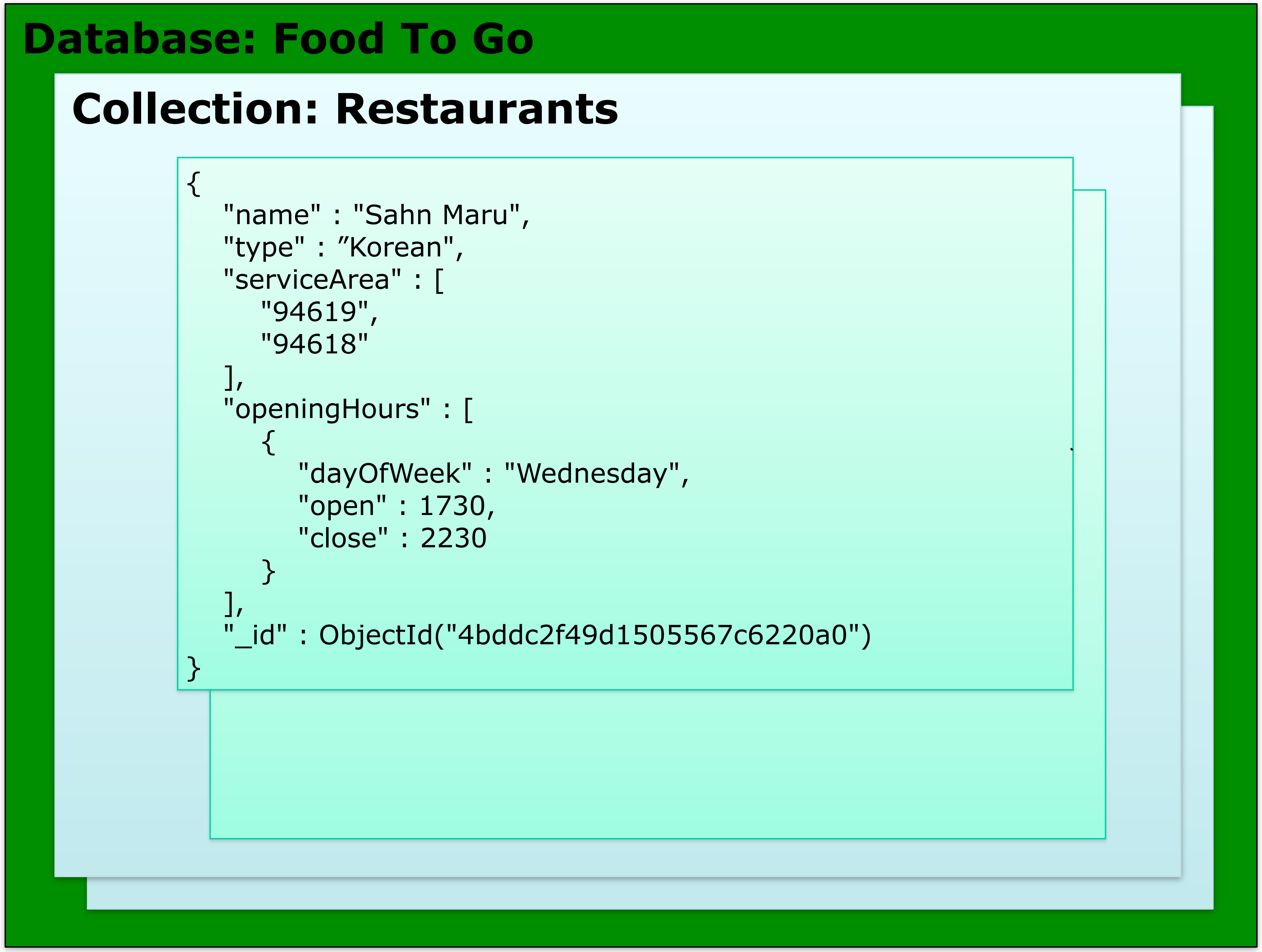
Data model = Binary JSON documents



```
{
   "name" : "Sahn Maru",
   "type" : ”Korean",
   "serviceArea" : [
      "94619",
      "94618"
   ],
   "openingHours" : [
      {
         "dayOfWeek" : "Wednesday",
         "open" : 1730,
         "close" : 2230
      }
   ],
   "_id" : ObjectId("4bddc2f49d1505567c6220a0")
}
```

Sequence of bytes on disk ➔ fast i/o

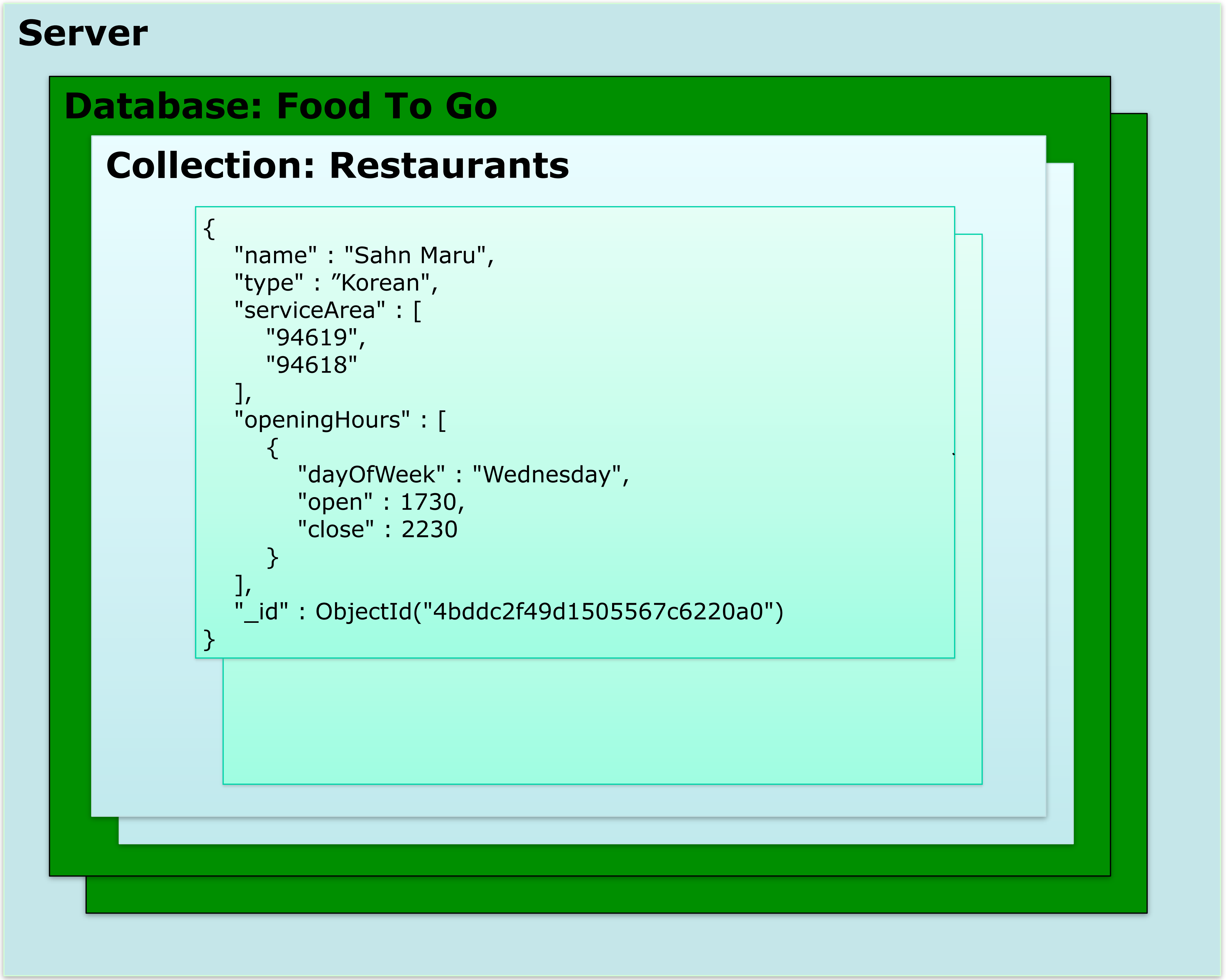
Data model = Binary JSON documents



```
{
   "name" : "Sahn Maru",
   "type" : ”Korean",
   "serviceArea" : [
      "94619",
      "94618"
   ],
   "openingHours" : [
      {
         "dayOfWeek" : "Wednesday",
         "open" : 1730,
         "close" : 2230
      }
   ],
   "_id" : ObjectId("4bddc2f49d1505567c6220a0")
}
```

Sequence of bytes on disk ➔ fast i/o

Data model = Binary JSON documents



```
{
    "name" : "Sahn Maru",
    "type" : ”Korean",
    "serviceArea" : [
        "94619",
        "94618"
    ],
    "openingHours" : [
        {
            "dayOfWeek" : "Wednesday",
            "open" : 1730,
            "close" : 2230
        }
    ],
    "_id" : ObjectId("4bddc2f49d1505567c6220a0")
}
```

Sequence of bytes on disk ➔ fast i/o

MongoDB CLI

```
> r = {name: 'Ajanta'}
> db.restaurants.save(r)
> r
{ "_id" : ObjectId("4e555dd9646e338dca11710c"), "name" : "Ajanta" }
> r = db.restaurants.findOne({name:"Ajanta"})
{ "_id" : ObjectId("4e555dd9646e338dca11710c"), "name" : "Ajanta" }
> r.type= "Indian”
> db.restaurants.save(r)
> db.restaurants.update({name:"Ajanta"},
                  {$set: {name:"Ajanta Restaurant"},
                   $push: { menuItems: {name: "Chicken Vindaloo"}}})
> db.restaurants.find()
{ "_id" : ObjectId("4e555dd9646e338dca11710c"), "menuItems" : [ { "name" : "Chicken
   Vindaloo" } ], "name" : "Ajanta Restaurant", "type" : "Indian" }
> db.restaurants.remove(r.id)
```

MongoDB query by example

```
{
  serviceArea:"94619",
  openingHours: {
    $elemMatch :  {
            "dayOfWeek" : "Monday",
            "open": {$lte: 1800},
            "close": {$gte: 1800}
     }
  }
}
```

Find a restaurant that serves the 94619 zip code and is open at 6pm on a Monday

```
DBCursor cursor = collection.find(qbeObject);
while (cursor.hasNext()) {
    DBObject o = cursor.next();
    …
 }
```

MongoDB use cases

- **Use cases**
  - High volume writes
  - Complex data
  - Semi-structured data
- **Who is using it?**
  - Shutterfly, Foursquare
  - Bit.ly Intuit
  - SourceForge, NY Times
  - GILT Groupe, Evite,
  - SugarCRM

Other NoSQL databases

Type	Examples
Extensible columns/Column-oriented	**Hbase** **SimpleDB** **DynamoDB**
Graph	**Neo4j**
Key-value	**Membase** **Voldemort**
Document	**CouchDb**

http://nosql-database.org/ lists 122+ NoSQL databases

Other NoSQL databases

Type	Examples
Extensible columns/Column-oriented	**Hbase** **SimpleDB** **DynamoDB**
Graph	**Neo4j**
Key-value	**Membase** **Voldemort**
Document	**CouchDb**

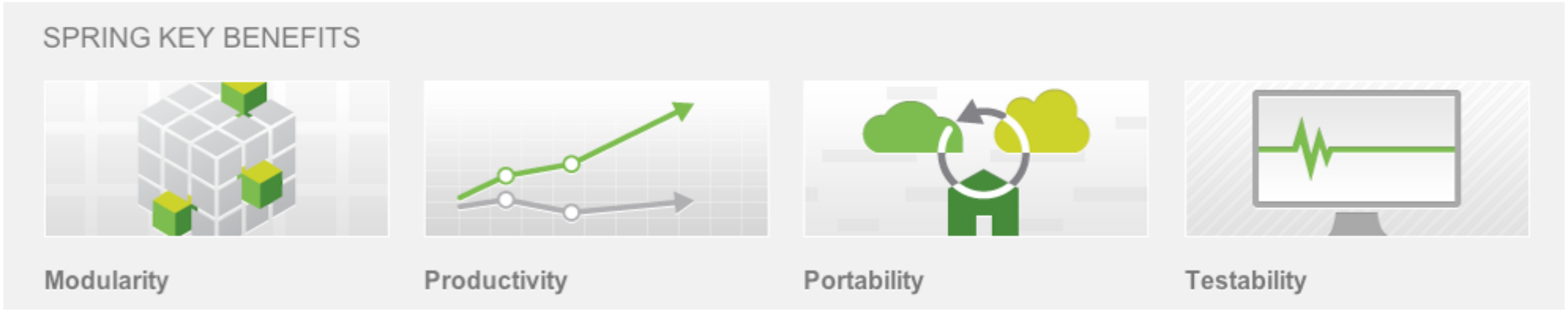
Sorry if I left out your favorite

http://nosql-database.org/ lists 122+ NoSQL databases

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- **Developing NoSQL applications for Cloud Foundry**
  - Why NoSQL?
  - Overview of NoSQL databases
  - **Introduction to Spring Data**
  - Using Spring Data for Redis
  - Using Spring Data for Mongo
  - Deploying on Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

Spring Data is here to help



For

NoSQL databases

http://www.springsource.org/spring-data

Spring Data sub-projects

- SQL: Spring Data JPA, JDBC extensions
- Commons: Polyglot persistence
- Key-Value: Redis, Riak
- Document: MongoDB
- Graph: Neo4j
- GORM for NoSQL

What you get

- Template classes that hide the boilerplate code
- Auto-generated (generic) repositories
- Java ⇔ NoSQL mapping
- Cross Store Persistence
- Support in Roo and Grails

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- **Developing NoSQL applications for Cloud Foundry**
  - Why NoSQL?
  - Overview of NoSQL databases
  - Introduction to Spring Data
  - **Using Spring Data for Redis**
  - Using Spring Data for Mongo
  - Deploying on Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

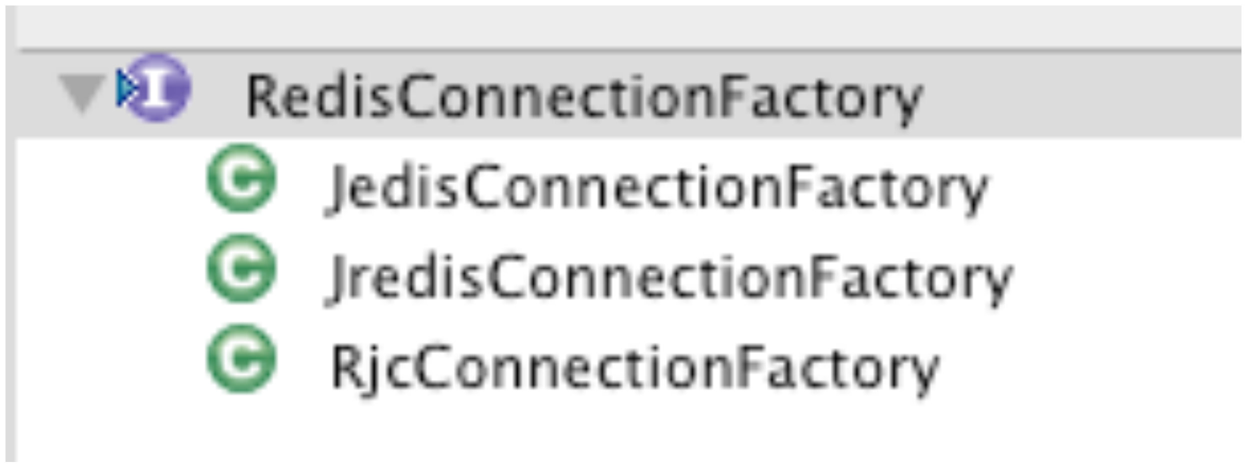
Redis challenges

- **Connection management**
  - Need to get and close connections
- **Data mapping**
  - Redis = binary/strings
  - Application = objects
- **Multiple client libraries**
  - Gratuitously different APIs

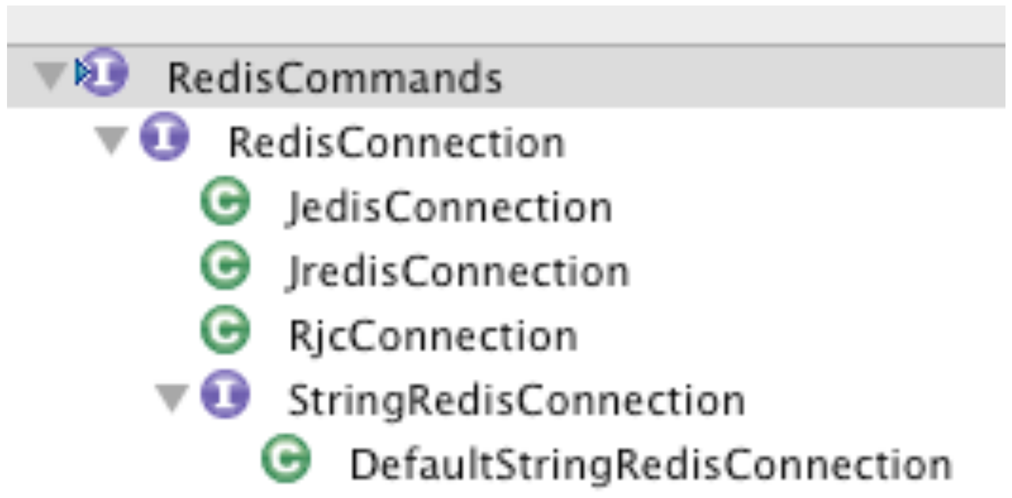
Spring Data for Redis

- **Low-level - RedisConnection(Factory)**
  - Supports Jedis, Jredis and Rjc
  - Insulates client code from underlying library
- **High-level - RedisTemplate**
  - Builds on RedisConnection(Factory)
  - Connection management
  - Pluggable Java ⇔ binary conversion
- **Support classes:**
  - Collections-backed by RedisTemplate
  - Atomic Counters

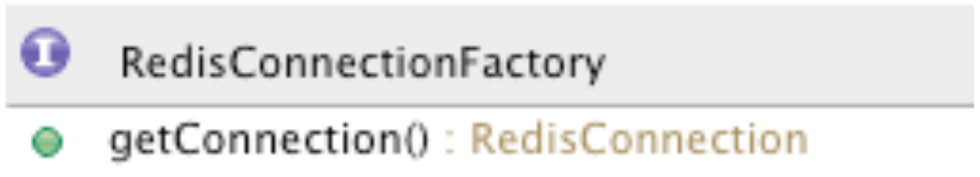
Low-level API = RedisConnection(Factory)



RedisConnectionFactory
JedisConnectionFactory
JredisConnectionFactory
RjcConnectionFactory



RedisCommands
RedisConnection
JedisConnection
JredisConnection
RjcConnection
StringRedisConnection
DefaultStringRedisConnection



RedisConnectionFactory
getConnection() : RedisConnection

Using RedisConnectionFactory

```
public class LowLevelRedisTest {

 @Autowired private RedisConnectionFactory redisConnectionFactory;

 @Test
 public void testLowLevel() {
  RedisConnection con = null;
  try {
   con = redisConnectionFactory.getConnection();

   byte[] key = "foo".getBytes();
   byte[] value = "bar".getBytes();
   con.set(key, value);

   byte[] retrievedValue = con.get(key);

   Assert.assertArrayEquals(value, retrievedValue);

  } finally {
   if (con != null) { con.close();  }
  }
 }
```

Library independent code ☺

Ugly byte arrays ☹

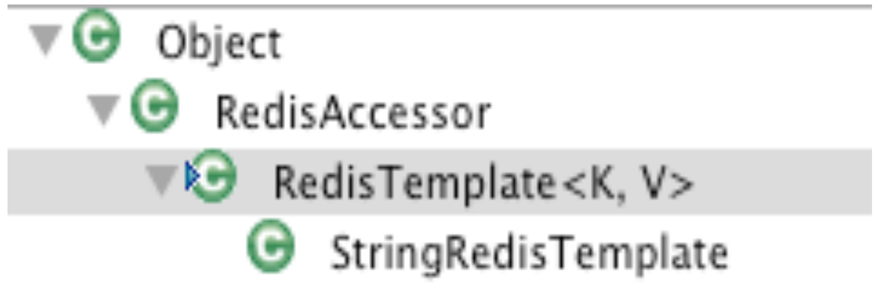
Need to clean up ☹

Configuring RedisConnectionFactory

```
@Configuration
public class RedisConfiguration {

  @Value("${databaseHostName}")
  protected String databaseHostName;

  @Bean
  public RedisConnectionFactory jedisConnectionFactory() {
    JedisConnectionFactory factory = new JedisConnectionFactory();
    factory.setHostName(databaseHostName);
    factory.setPort(6379);
    factory.setUsePool(true);
    return factory;
  }

}
```

High-level API = RedisTemplate

- **Builds on RedisConnection(Factory)**
- **Analogous to JdbcTemplate**
- **Parameterized type**
  - K - Key type
  - V – Value type
- **Handles Java Key/Value ⇔ Redis byte[]**
- **Maps Redis exceptions ⇒ DataAccessException**
- **StringRedisTemplate**
  - Extends RedisTemplate<String, String>
  - Keys and values are Strings



```
▼ Object
  ▼ RedisAccessor
    ▼ RedisTemplate<K, V>
        StringRedisTemplate
```

Using StringRedisTemplate

```
public class RedisTemplateTest {

  @Autowired private StringRedisTemplate stringRedisTemplate;

 @Test
 public void testGetAndSet() {
  stringRedisTemplate.opsForValue().set("foo", "bar");
  assertEquals("bar", stringRedisTemplate.opsForValue().get("foo"));
 }

 @Test
 public void testHashOps() {
  stringRedisTemplate.opsForHash().put("myHash", "myKey", "value");

  assertEquals("value",
      stringRedisTemplate.opsForHash().get("myHash", "myKey"));

  assertEquals(Collections.singleton("myKey"),
      stringRedisTemplate.opsForHash().keys("myHash"));

  assertEquals(Collections.singletonMap("myKey", "value"),
      stringRedisTemplate.opsForHash().entries("myHash"));
 }
```

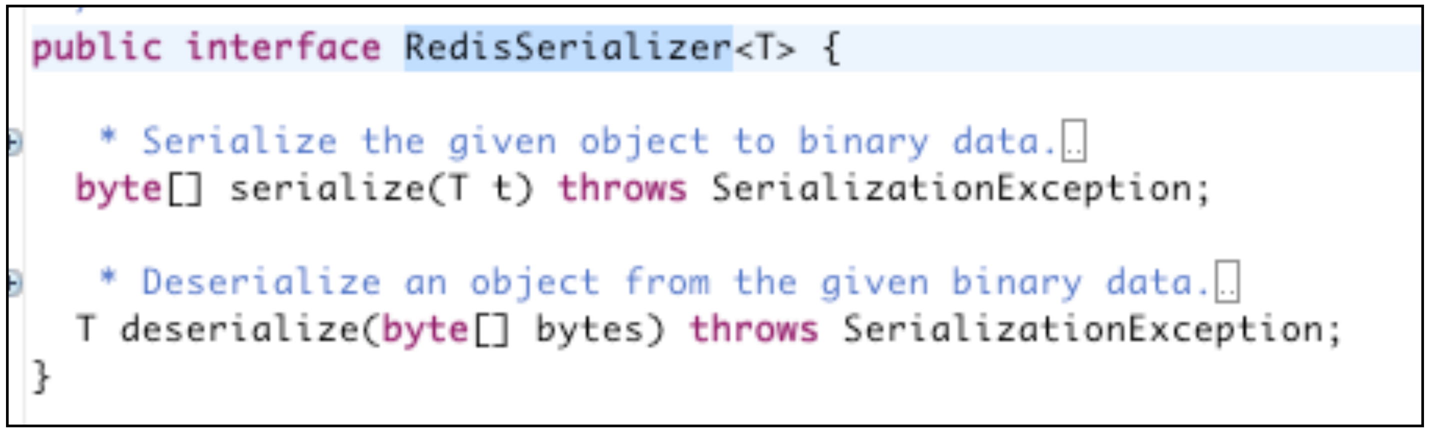
Returns KV type specific interface

Converts between Strings and byte[]

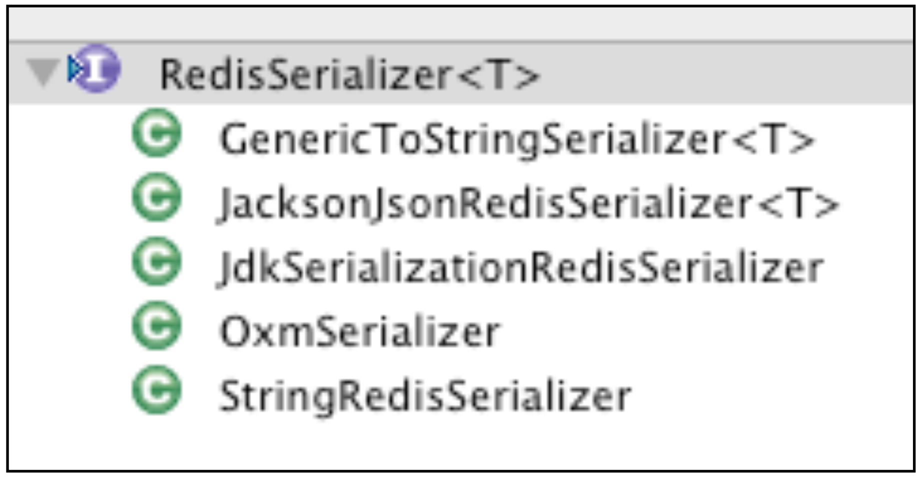
Configuring StringRedisTemplate

```
@Configuration
public class RedisConfiguration {

  @Bean
  public RedisConnectionFactory jedisConnectionFactory() {
    ...
  }

 @Bean
  public StringRedisTemplate stringRedisTemplate(RedisConnectionFactory
                                                     factory) {
    StringRedisTemplate template = new StringRedisTemplate();
    template.setConnectionFactory(factory);
    return template;
  }
}
```

RedisTemplate: Java objects ⇔ binary data



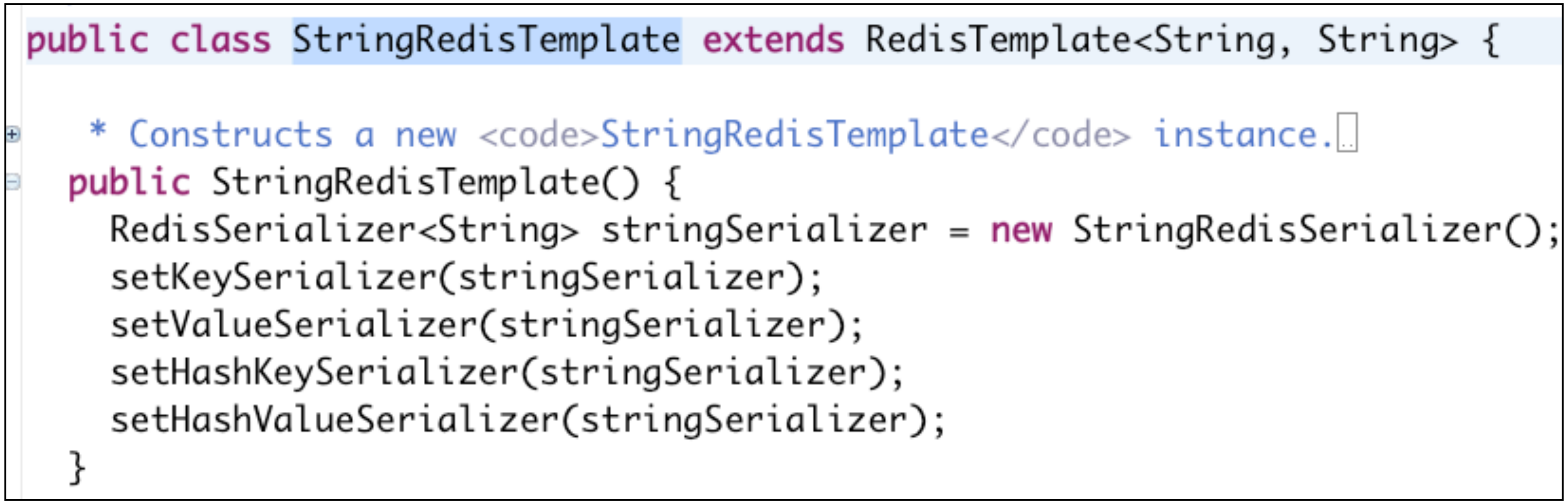
```
public interface RedisSerializer<T> {

    * Serialize the given object to binary data.
  byte[] serialize(T t) throws SerializationException;

    * Deserialize an object from the given binary data.
  T deserialize(byte[] bytes) throws SerializationException;
}
```



```
RedisSerializer<T>
    GenericToStringSerializer<T>
    JacksonJsonRedisSerializer<T>
    JdkSerializationRedisSerializer
    OxmSerializer
    StringRedisSerializer
```

- **RedisTemplate has multiple Serializers:**
  - DefaultSerializer - defaults to JdkSerializationRedisSerializer
  - KeySerializer
  - ValueSerializer
  - HashKeySerializer
  - HashValueSerializer

StringRedisTemplate uses StringRedisSerializer



```java
public class StringRedisTemplate extends RedisTemplate<String, String> {

   * Constructs a new <code>StringRedisTemplate</code> instance.
  public StringRedisTemplate() {
    RedisSerializer<String> stringSerializer = new StringRedisSerializer();
    setKeySerializer(stringSerializer);
    setValueSerializer(stringSerializer);
    setHashKeySerializer(stringSerializer);
    setHashValueSerializer(stringSerializer);
  }
```

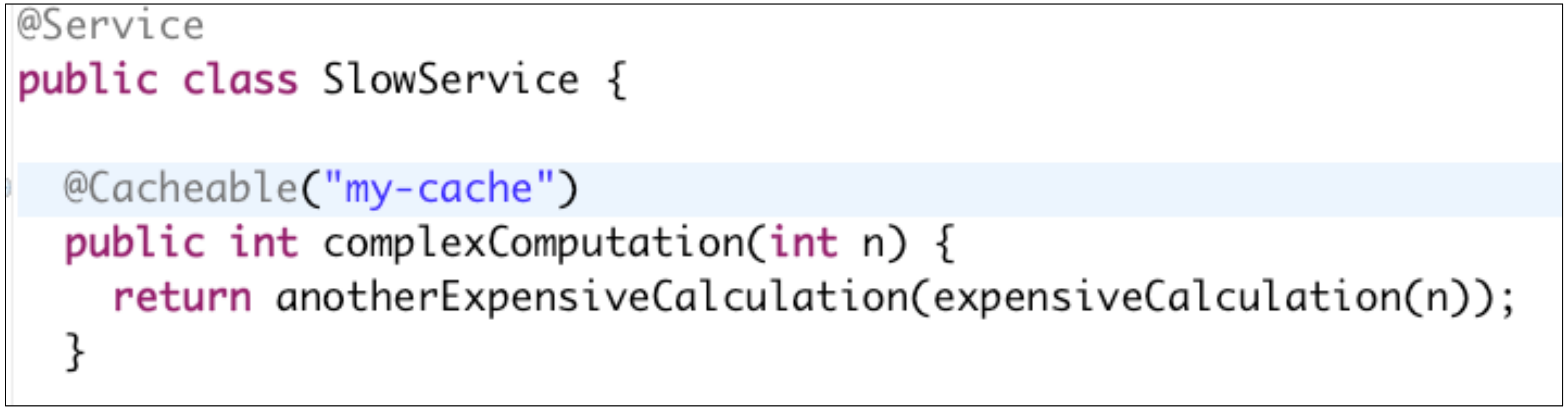
Register serializers to override the default behavior

```
@Bean
@Qualifier("Restaurant")
public RedisTemplate<String, Restaurant> restaurantTemplate(RedisConnectionFactory factory) {
  RedisTemplate<String, Restaurant> template = new RedisTemplate<String, Restaurant>();
  template.setConnectionFactory(factory);
  template.setDefaultSerializer(new StringRedisSerializer());
  JacksonJsonRedisSerializer<Restaurant> jsonSerializer = makeRestaurantJsonSerializer();
  template.setValueSerializer(jsonSerializer);
  return template;
}
```

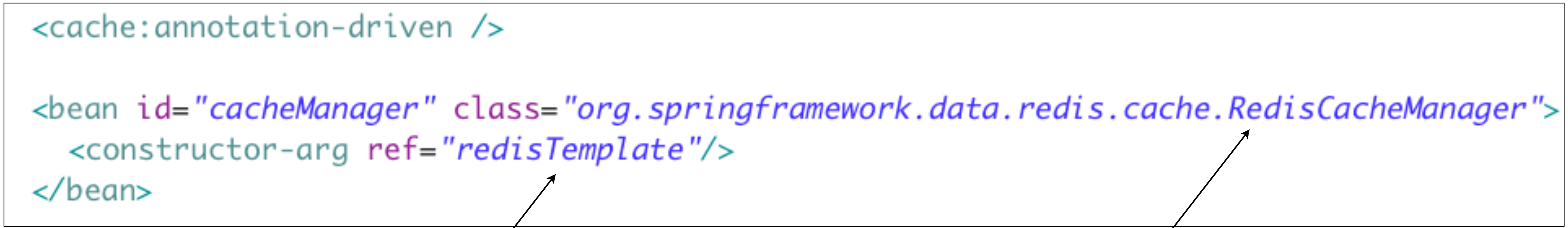
```
@Override
public void addRestaurantDetails(Restaurant restaurant) {
  restaurantTemplate.opsForValue().set(keyFormatter.key(restaurant.getId()), restaurant);
}
```

Converted to JSON by RedisTemplate

Redis caching support



```
@Service
public class SlowService {

  @Cacheable("my-cache")
  public int complexComputation(int n) {
    return anotherExpensiveCalculation(expensiveCalculation(n));
  }
```



```
<cache:annotation-driven />

<bean id="cacheManager" class="org.springframework.data.redis.cache.RedisCacheManager">
  <constructor-arg ref="redisTemplate"/>
</bean>
```

Template needs to (de)serialize K and V

KVs = <prefix + K,V>
Sorted set of all keys for clear()

Other Spring data for Redis features

- Redis-backed collections
- Atomic counters
- Support for Redis Pub/sub

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- **Developing NoSQL applications for Cloud Foundry**
  - Why NoSQL?
  - Overview of NoSQL databases
  - Introduction to Spring Data
  - Using Spring Data for Redis
  - **Using Spring Data for Mongo**
  - Deploying on Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

MongoDB API usage patterns

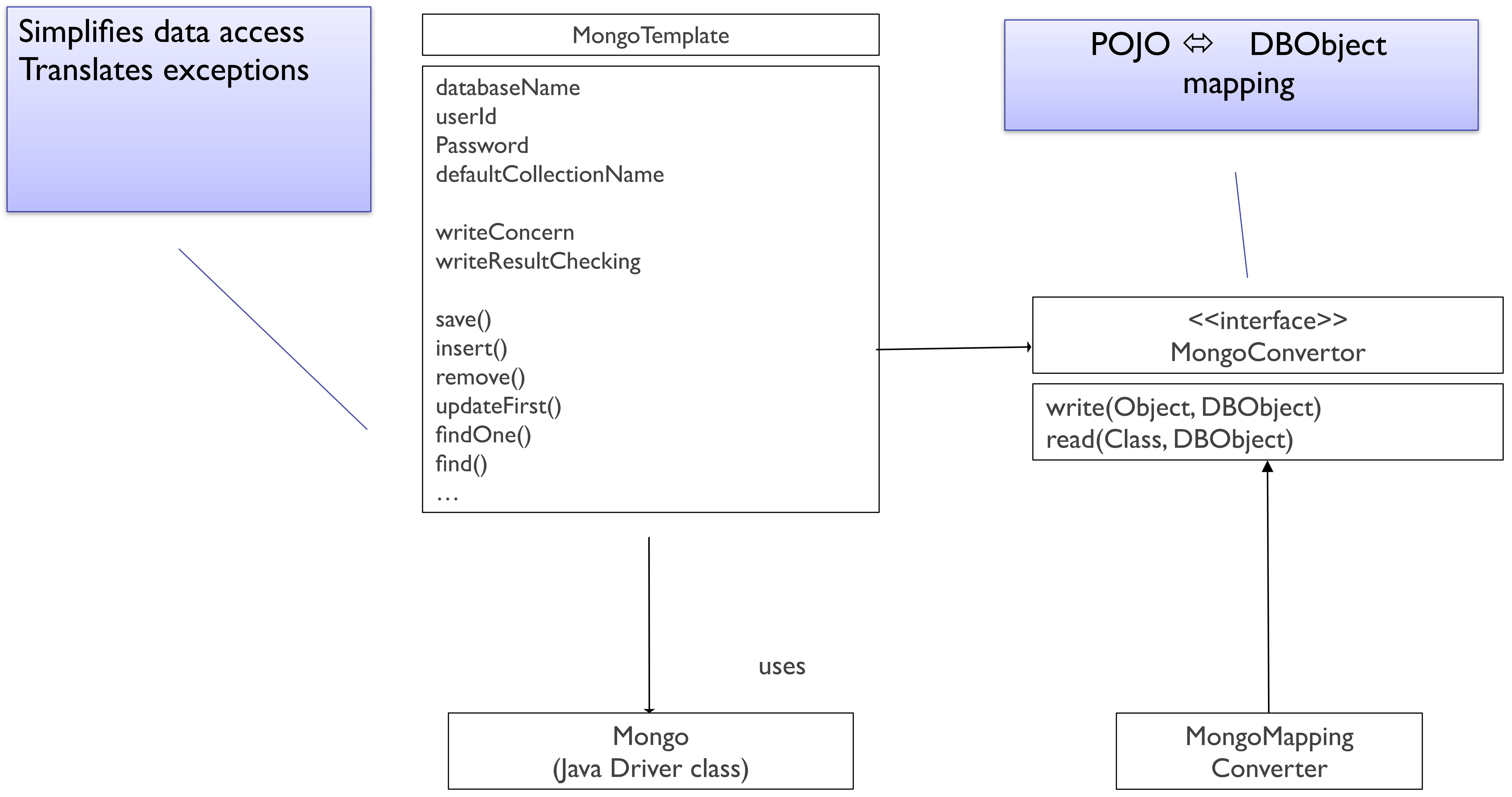
- **Create and store Mongo singleton**
- **Externalized server host, port etc.**
- **Inserts/Updates**
  - Map application POJO ⇒ DBObject
  - mongo.getDatabase(…).getCollection(…)
  - Partial document updates
  - Configure asynchronous vs. synchronous writes
- **Queries**
  - Construct query object
  - mongo.getDatabase(…).getCollection(…)
  - Iterate through Cursor
  - Map DBObject ⇒ application POJO

⇒ Higher-level than JDBC but still repetitive, …

Spring Data - MongoDB

- MongoTemplate
- Generic repositories
- Querydsl integration
- Cross-store persistence

MongoTemplate



Example entity

```
public class Restaurant {
 private String id;
 private String name;
 private List<MenuItem> menuItems;

 public Restaurant() {
 }

 public Restaurant(String name) {
   this.name = name;
   ...
 }

public String getName() { return name; }

 public void setName(String name) {
    this.name = name;
 }

 ...getters and setters...
```

```
public class MenuItem {
 private String name;
 private double price;

 public MenuItem() {
 }

  public MenuItem(String name,
                         double price) {
   this.name = name;
   this.price = price;
  }

 ...getters and setters...
```

Example data access code

```
@Repository
public class RestaurantRepository {

 @Autowired
 private MongoTemplate mongoTemplate;

 public static final String RESTAURANTS_COLLECTION = "restaurants";


 public void add(Restaurant restaurant) {
  mongoTemplate.save(RESTAURANTS_COLLECTION, restaurant);
 }

 public List<Restaurant> findRestaurantsByName(String restaurantName) {
  return mongoTemplate.find(RESTAURANTS_COLLECTION,
            new Query(where("name").is(restaurantName)),
            Restaurant.class);
 }
```

Mongo document

```
{
 "_id" : ObjectId("4d977f55d3fe3119c904e026"),
 "menuItems" : [
          {
                    "name" : "Tandoori Portobello Mushrooms",
                    "price" : 5.5
          },
          {
                    "name" : "Duck Curry Kerala",
                    "price" : 15
          }
     ],
  "name" : "Ajanta"
}
```

Spring MongoDB Example - Config 1

```
@Configuration public class MongoDbExampleConfig {
  private @Value("#{mongoDbProperties.databaseName}") String mongoDbDatabase;
  private @Value("#{mongoDbProperties.host}") String mongoDbHost;

 @Bean public Mongo mongo()  throws Exception {
   return new Mongo(mongoDbHost);
  }

  @Bean public MongoTemplate mongoTemplate(Mongo mongo)  {
   MongoTemplate mongoTemplate = new MongoTemplate(mongo, mongoDbDatabase);
   mongoTemplate.setWriteConcern(WriteConcern.SAFE);
   mongoTemplate.setWriteResultChecking(WriteResultChecking.EXCEPTION);
   return mongoTemplate;
  }
...
```

```
<beans>
 <context:annotation-config/>

 <context:component-scan
     base-package="net.chrisrichardson.mongodb.example"/>

<util:properties id="mongoDbProperties"
    location="mongodb.properties"/>

</beans>
```

External Config

mongodb.properties:

```
databaseName=demo1
host=192.168.253.150
```

Spring MongoDB Example - Config 2

```
<bean id="mongoTemplate"
    class="org.springframework.data.mongodb.core.MongoTemplate">
     <constructor-arg ref="mongoFactory"/>
</bean>

<mongo:db-factory id="mongoFactory"
   host= "#{mongoDbProperties.host}"
   dbname="#{mongoDbProperties.databaseName}" />


<util:properties
   id="mongoDbProperties"
   location="mongodb.properties"/>
```

Summarize other features

- In-place updates
- Callbacks
- Generic repositories
- Annotation-driven mapping
- Support for QueryDSL
- Cross-store persistence

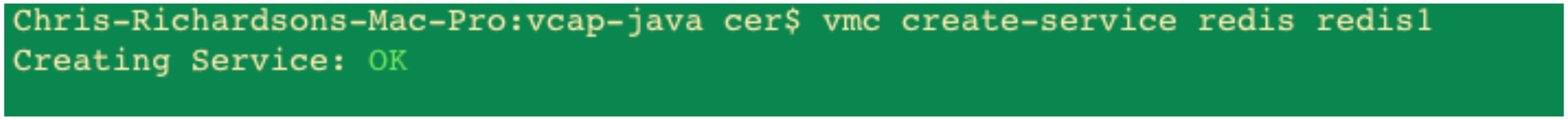
Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- **Developing NoSQL applications for Cloud Foundry**
  - Why NoSQL?
  - Overview of NoSQL databases
  - Introduction to Spring Data
  - Using Spring Data for Redis
  - Using Spring Data for Mongo
  - **Deploying on Cloud Foundry**
- Application integration with RabbitMQ and Spring AMQP
- Wrap up

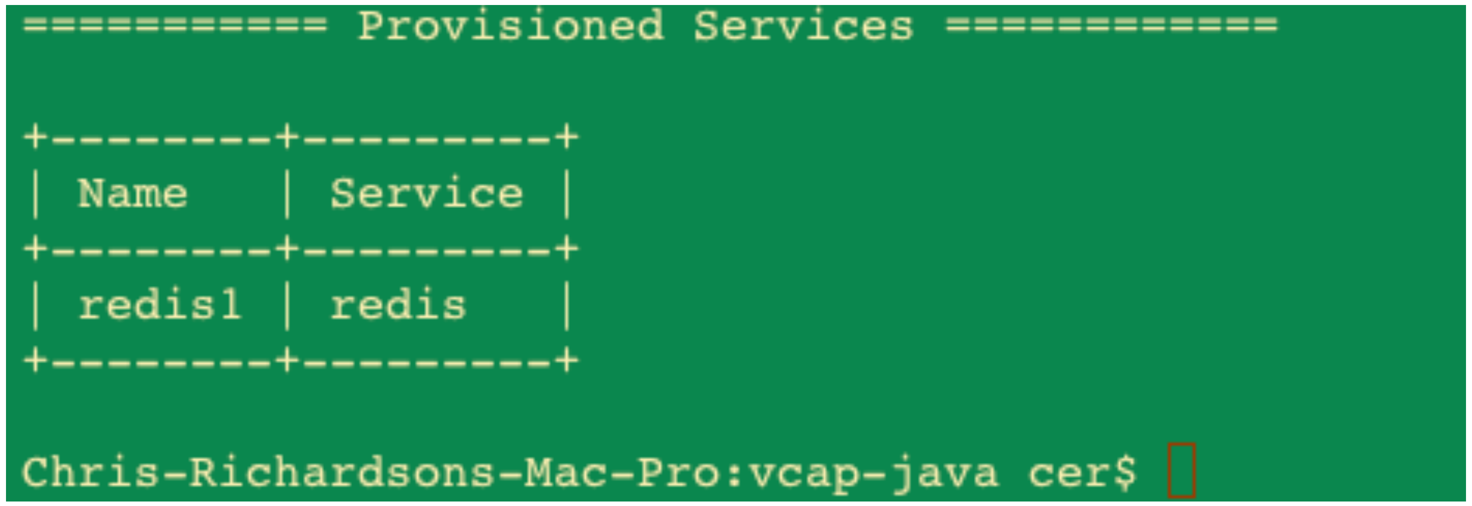
Using Mongo and Redis with Cloud Foundry

- **Create a service - Mongo or Redis**
- **Bind it to your application**
- **Use <cloud:*/> namespace to access the bound service**
  - when cloud profile is active

Creating a Redis Server

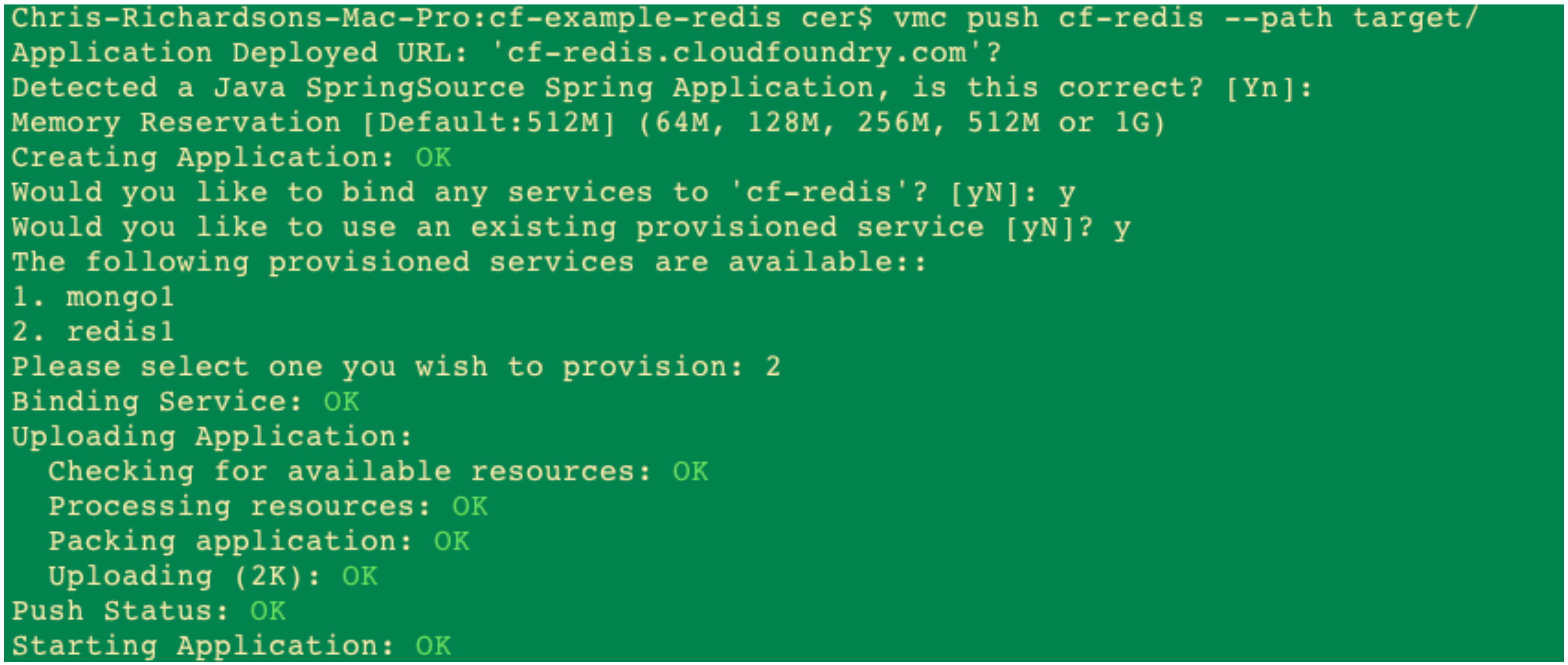


```
Chris-Richardsons-Mac-Pro:vcap-java cer$ vmc create-service redis redis1
Creating Service: OK
```



```
=========== Provisioned Services ============

+--------+---------+
| Name   | Service |
+--------+---------+
| redis1 | redis   |
+--------+---------+

Chris-Richardsons-Mac-Pro:vcap-java cer$
```

Deploying a Redis application



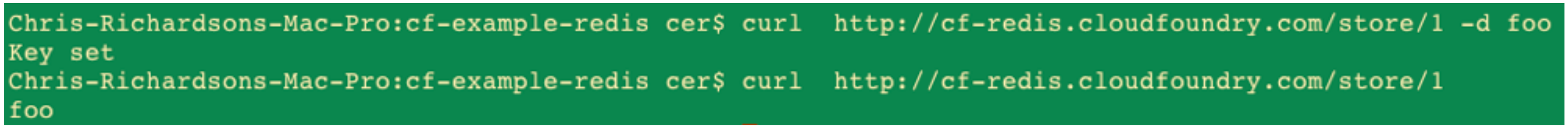
```
Chris-Richardsons-Mac-Pro:cf-example-redis cer$ vmc push cf-redis --path target/
Application Deployed URL: 'cf-redis.cloudfoundry.com'?
Detected a Java SpringSource Spring Application, is this correct? [Yn]:
Memory Reservation [Default:512M] (64M, 128M, 256M, 512M or 1G)
Creating Application: OK
Would you like to bind any services to 'cf-redis'? [yN]: y
Would you like to use an existing provisioned service [yN]? y
The following provisioned services are available::
1. mongo1
2. redis1
Please select one you wish to provision: 2
Binding Service: OK
Uploading Application:
  Checking for available resources: OK
  Processing resources: OK
  Packing application: OK
  Uploading (2K): OK
Push Status: OK
Starting Application: OK
```

Redis bean definitions

```xml
<beans profile="default">
    <bean id="redisConnectionFactory"
        class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory" />
</beans>

<beans profile="cloud">
    <cloud:redis-connection-factory id="redisConnectionFactory" />
</beans>
```

```xml
<bean id="redisTemplate"
    class="org.springframework.data.redis.core.StringRedisTemplate"
    p:connectionFactory-ref="redisConnectionFactory" />
```

Using the application



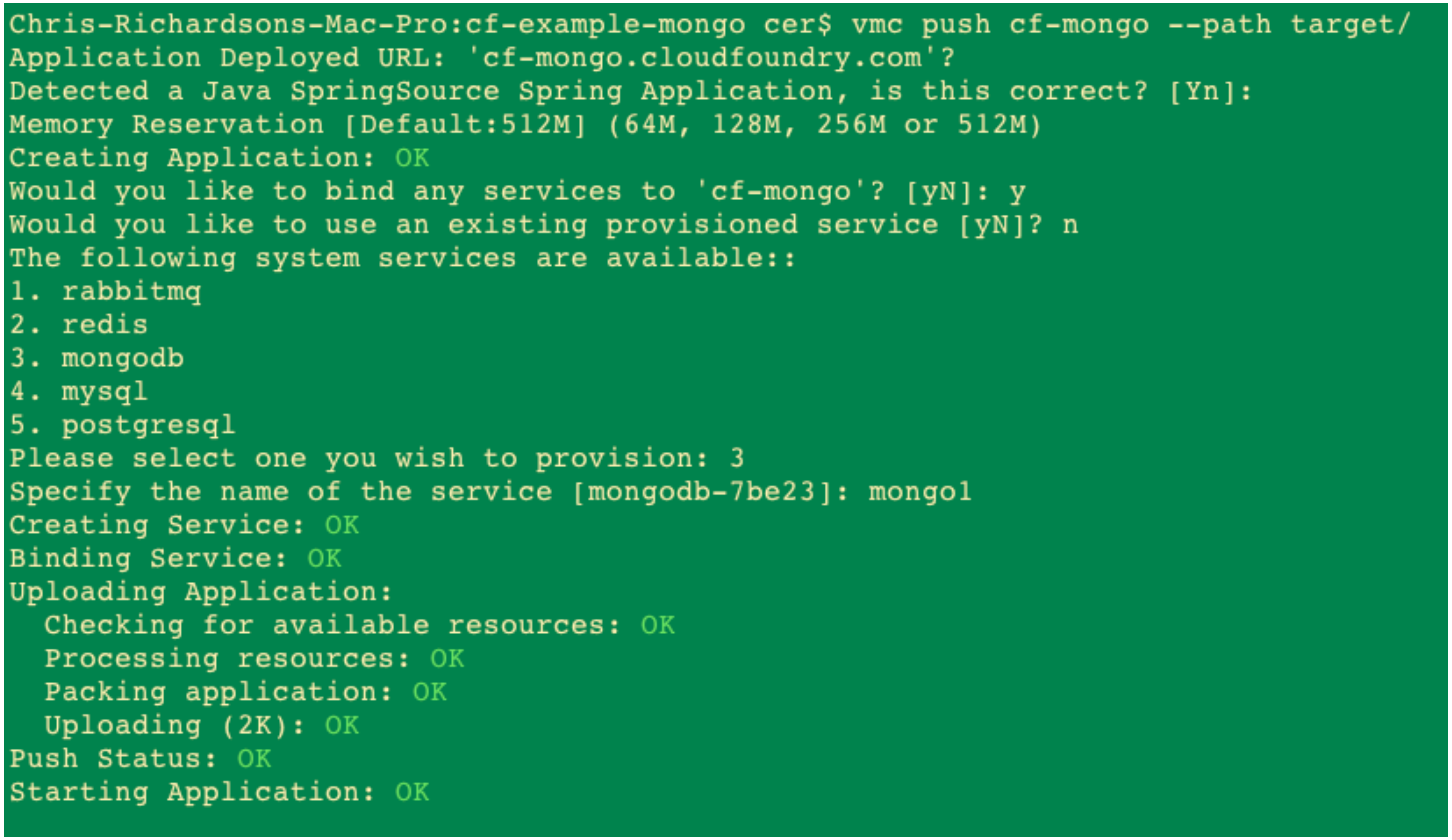
```
Chris-Richardsons-Mac-Pro:cf-example-redis cer$ curl  http://cf-redis.cloudfoundry.com/store/1 -d foo
Key set
Chris-Richardsons-Mac-Pro:cf-example-redis cer$ curl  http://cf-redis.cloudfoundry.com/store/1
foo
```

About <cloud:redis-connection-factory/>

```
<cloud:redis-connection-factory
          id="redisConnectionFactory"
         [ service-name="redis1" ]
    />
```

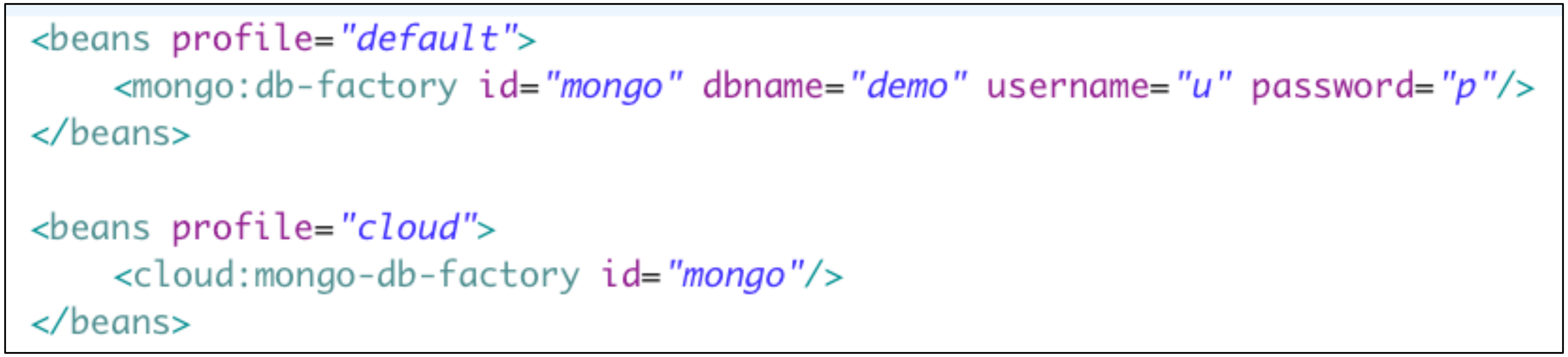
Use when multiple services are bound

Deploying a Mongo application

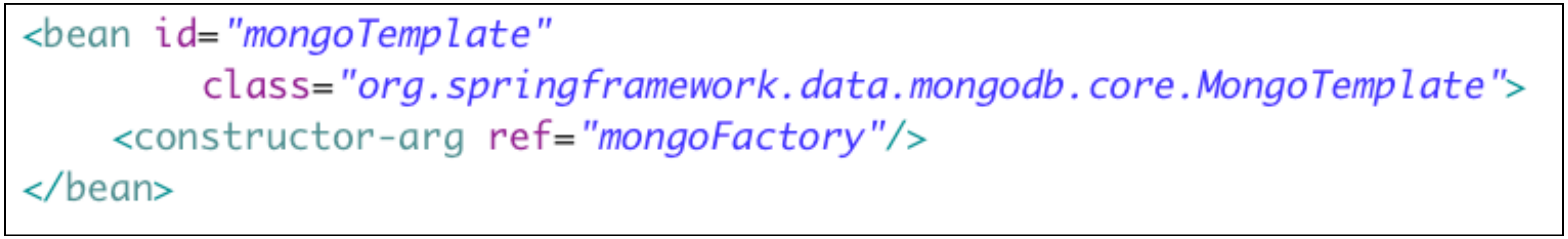


```
Chris-Richardsons-Mac-Pro:cf-example-mongo cer$ vmc push cf-mongo --path target/
Application Deployed URL: 'cf-mongo.cloudfoundry.com'?
Detected a Java SpringSource Spring Application, is this correct? [Yn]:
Memory Reservation [Default:512M] (64M, 128M, 256M or 512M)
Creating Application: OK
Would you like to bind any services to 'cf-mongo'? [yN]: y
Would you like to use an existing provisioned service [yN]? n
The following system services are available::
1. rabbitmq
2. redis
3. mongodb
4. mysql
5. postgresql
Please select one you wish to provision: 3
Specify the name of the service [mongodb-7be23]: mongo1
Creating Service: OK
Binding Service: OK
Uploading Application:
  Checking for available resources: OK
  Processing resources: OK
  Packing application: OK
  Uploading (2K): OK
Push Status: OK
Starting Application: OK
```

MongoDB bean definitions

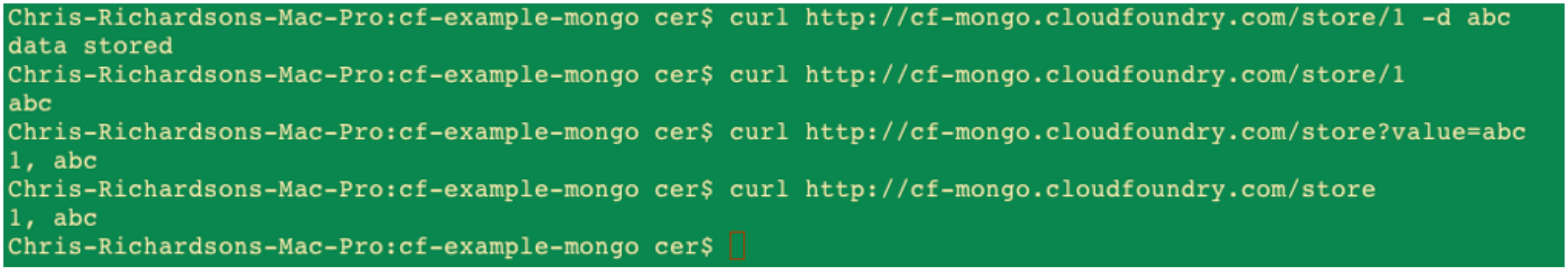


```xml
<beans profile="default">
    <mongo:db-factory id="mongo" dbname="demo" username="u" password="p"/>
</beans>

<beans profile="cloud">
    <cloud:mongo-db-factory id="mongo"/>
</beans>
```



```xml
<bean id="mongoTemplate"
        class="org.springframework.data.mongodb.core.MongoTemplate">
    <constructor-arg ref="mongoFactory"/>
</bean>
```

Using the Mongo Application



```
Chris-Richardsons-Mac-Pro:cf-example-mongo cer$ curl http://cf-mongo.cloudfoundry.com/store/1 -d abc
data stored
Chris-Richardsons-Mac-Pro:cf-example-mongo cer$ curl http://cf-mongo.cloudfoundry.com/store/1
abc
Chris-Richardsons-Mac-Pro:cf-example-mongo cer$ curl http://cf-mongo.cloudfoundry.com/store?value=abc
1, abc
Chris-Richardsons-Mac-Pro:cf-example-mongo cer$ curl http://cf-mongo.cloudfoundry.com/store
1, abc
Chris-Richardsons-Mac-Pro:cf-example-mongo cer$
```

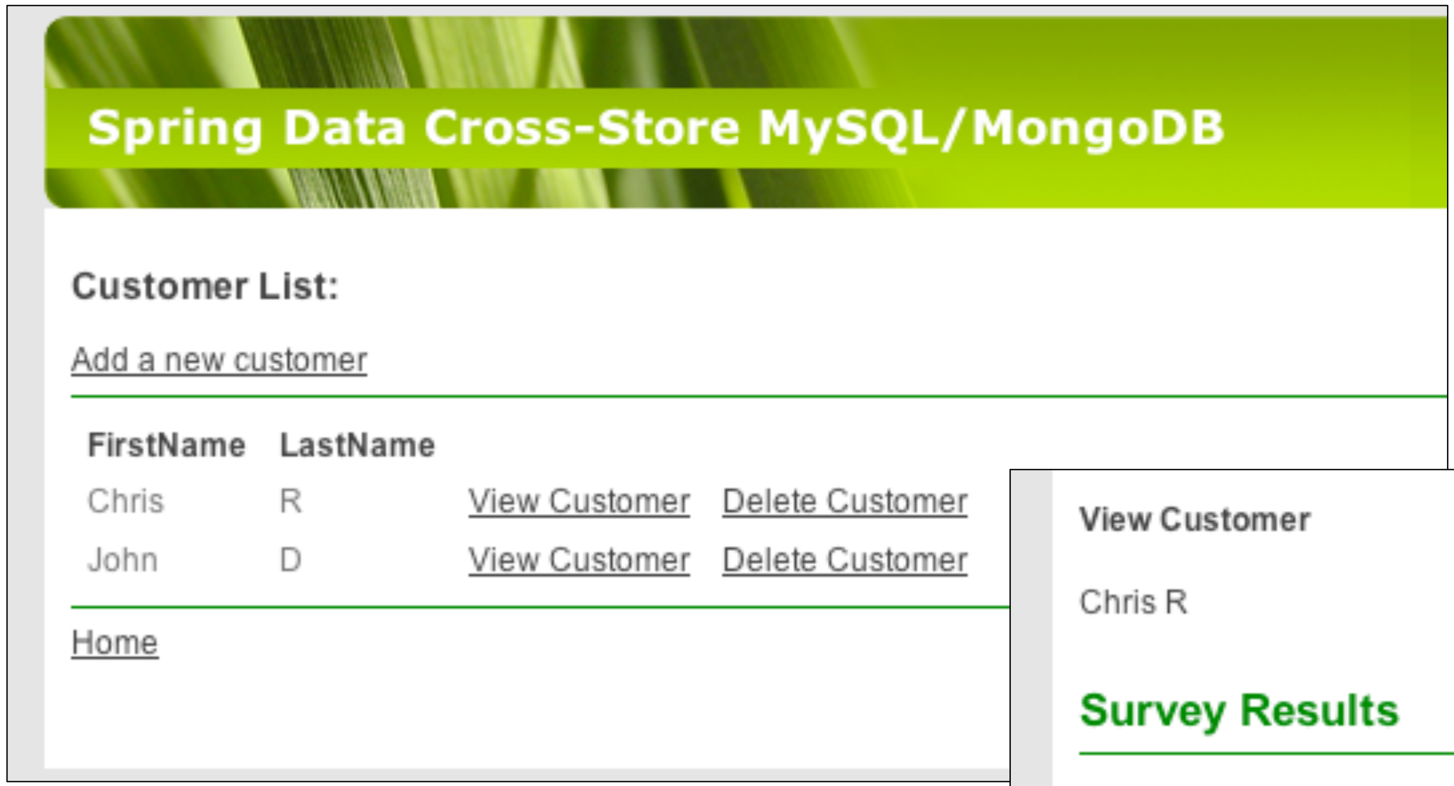
About <cloud:mongo-db-factory/>

```
<cloud:mongo-db-factory
          id="mongoFactory"
         [ service-name="mongo1" ]
         [ write-concern="SAFE"  ]
    >
   [ <cloud:mongo-options
          connections-per-host="..."
          max-wait-time="..."
    /> ]
</cloud:mongo-db-factory>
```

Use when multiple services are bound

Whether to wait for writes to complete

Cross store persistence example



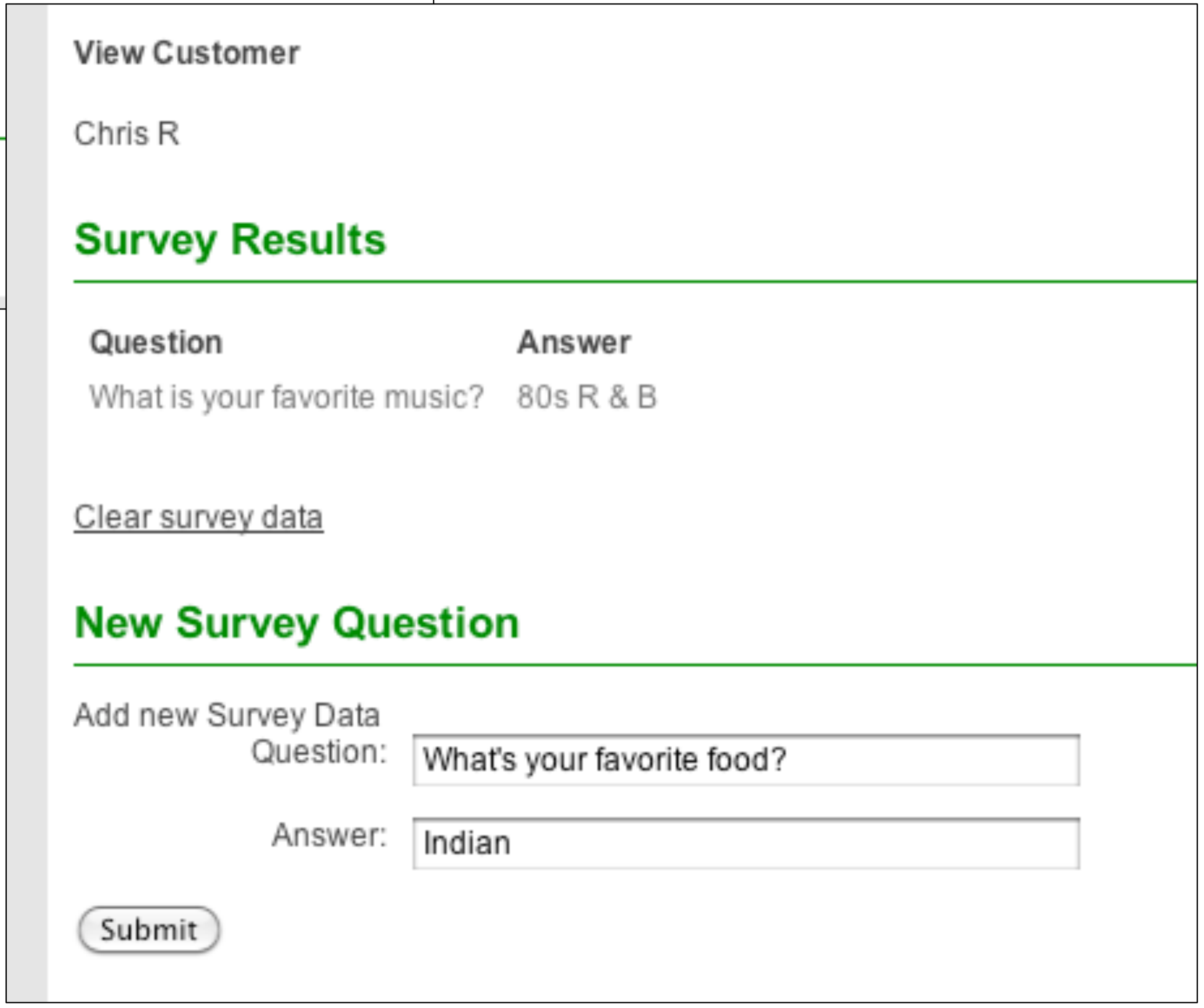
Spring Data Cross-Store MySQL/MongoDB

Customer List:

Add a new customer

FirstName	LastName		
Chris	R	View Customer	Delete Customer
John	D	View Customer	Delete Customer

Home



View Customer

Chris R

Survey Results

Question	Answer
What is your favorite music?	80s R & B

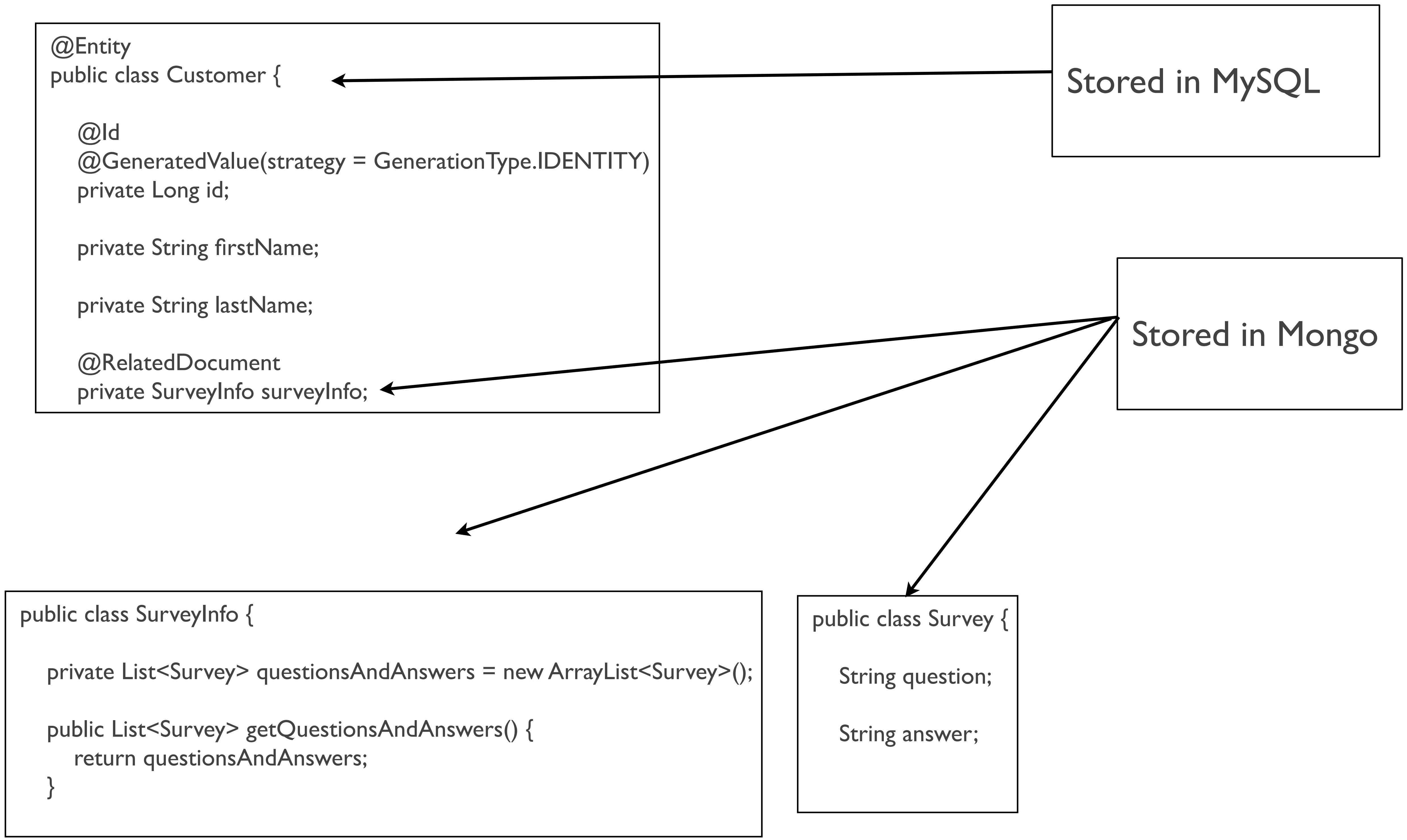
Clear survey data

New Survey Question

Add new Survey Data

Question:	What's your favorite food?
Answer:	Indian

Uses MySQL and MongoDB



```
@Entity
public class Customer {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String firstName;

    private String lastName;

    @RelatedDocument
    private SurveyInfo surveyInfo;
```

```
public class SurveyInfo {

    private List<Survey> questionsAndAnswers = new ArrayList<Survey>();

    public List<Survey> getQuestionsAndAnswers() {
        return questionsAndAnswers;
    }
```

```
public class Survey {

    String question;

    String answer;
```

Cross store configuration

```
@Configuration
@ComponentScan(basePackageClasses = CrossStoreCustomerRepository.class)
@EnableTransactionManagement(mode = AdviceMode.ASPECTJ)
public class ServicesConfiguration {

    private String mongoDatabaseServiceName = "survey-mongo";
    private String mysqlDatabaseServiceName = "survey-mysql";

    @Bean
    public CloudEnvironment cloudEnvironment() {
        return new CloudEnvironment();
    }

    @Bean
    public MongoServiceInfo mongoServiceInfo() {
        return cloudEnvironment().getServiceInfo(mongoDatabaseServiceName, MongoServiceInfo.class);
    }

    @Bean
    public MongoDbFactory mongoDbFactory() {
        MongoServiceCreator mongoServiceCreator = new MongoServiceCreator();
        return mongoServiceCreator.createService(mongoServiceInfo());
    }

    @Bean
    public DataSource dataSource() {
        RdbmsServiceInfo rdbmsServiceInfo = cloudEnvironment().getServiceInfo(mysqlDatabaseServiceName, RdbmsServiceInfo.class);
        RdbmsServiceCreator rdbmsServiceCreator = new RdbmsServiceCreator();
        DataSource dataSource = rdbmsServiceCreator.createService(rdbmsServiceInfo);
        return dataSource;
    }
...
```

Manifest for Cloud Foundry deployment

```
---
applications:
  target:
    name: xs-survey
    url: ${name}.${target-base}
    framework:
      name: spring
      info:
        mem: 512M
        description: Java SpringSource Spring Application
        exec:
    mem: 512M
    instances: 1
    services:
      survey-mongo:
        type: :mongodb
      survey-mysql:
        type: :mysql
```

```
Chris-Richardsons-Mac-Pro:cross-store cer$ vmc apps

+-------------+----+---------+-------------------------------------------------------+----------------------------+
| Application | #  | Health  | URLS                                                  | Services                   |
+-------------+----+---------+-------------------------------------------------------+----------------------------+
| xs-survey   | 1  | RUNNING | xs-survey.cloudfoundry.com                            | survey-mysql, survey-mongo |
+-------------+----+---------+-------------------------------------------------------+----------------------------+
```

NoSQL and Caldecott

- **Caldecott let’s you tunnel to a NoSQL service**
- **Use Redis CLI**
  - redis-cli
  - Explore database, adhoc operations
  - ...
- **Use Mongo CLI etc**
  - Explore database, adhoc operations
  - Mongo dump/restore
  - ...

NoSQL wrap up

- **Cloud Foundry supports Mongo and Redis**
- **For some use cases, NoSQL databases offer some combination of:**
  - Higher scalability
  - Higher performance
  - Richer data models
  - Schema less
- **Spring Data simplifies the development of NoSQL applications**

Cloud Foundry

+

Spring Data

=

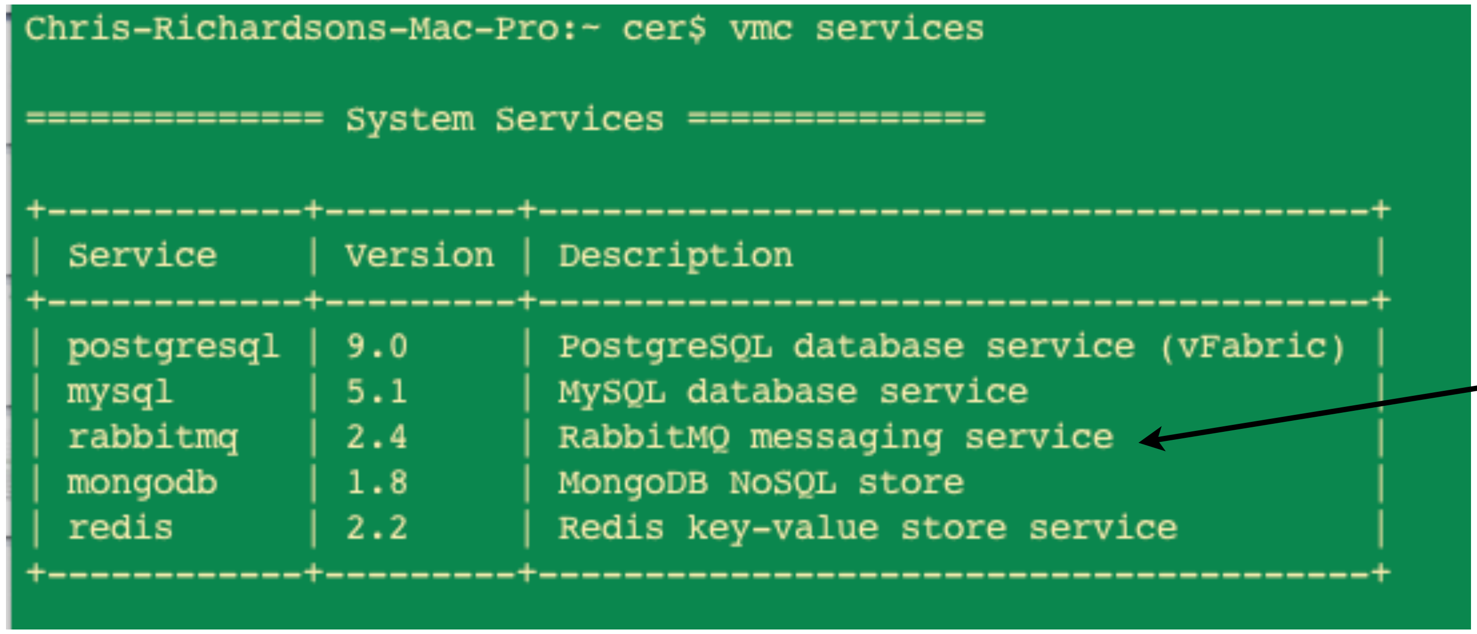
Easy development and deployment of NoSQL applications

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- Developing NoSQL applications for Cloud Foundry
- **Application integration with RabbitMQ and Spring AMQP**
  - Why messaging?
  - Messaging with RabbitMQ and AMQP
  - Using Spring Integration
  - Cloud Foundry and RabbitMQ
- Wrap up

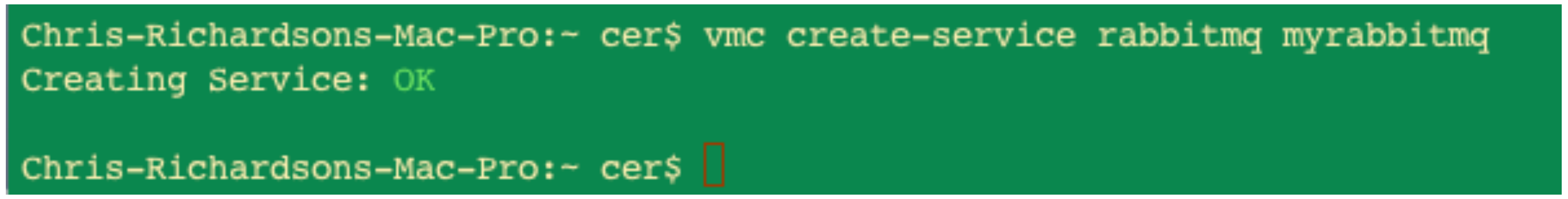
Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- Developing NoSQL applications for Cloud Foundry
- **Application integration with RabbitMQ and Spring AMQP**
  - **Why messaging?**
  - Messaging with RabbitMQ and AMQP
  - Using Spring Integration
  - Cloud Foundry and RabbitMQ
- Wrap up



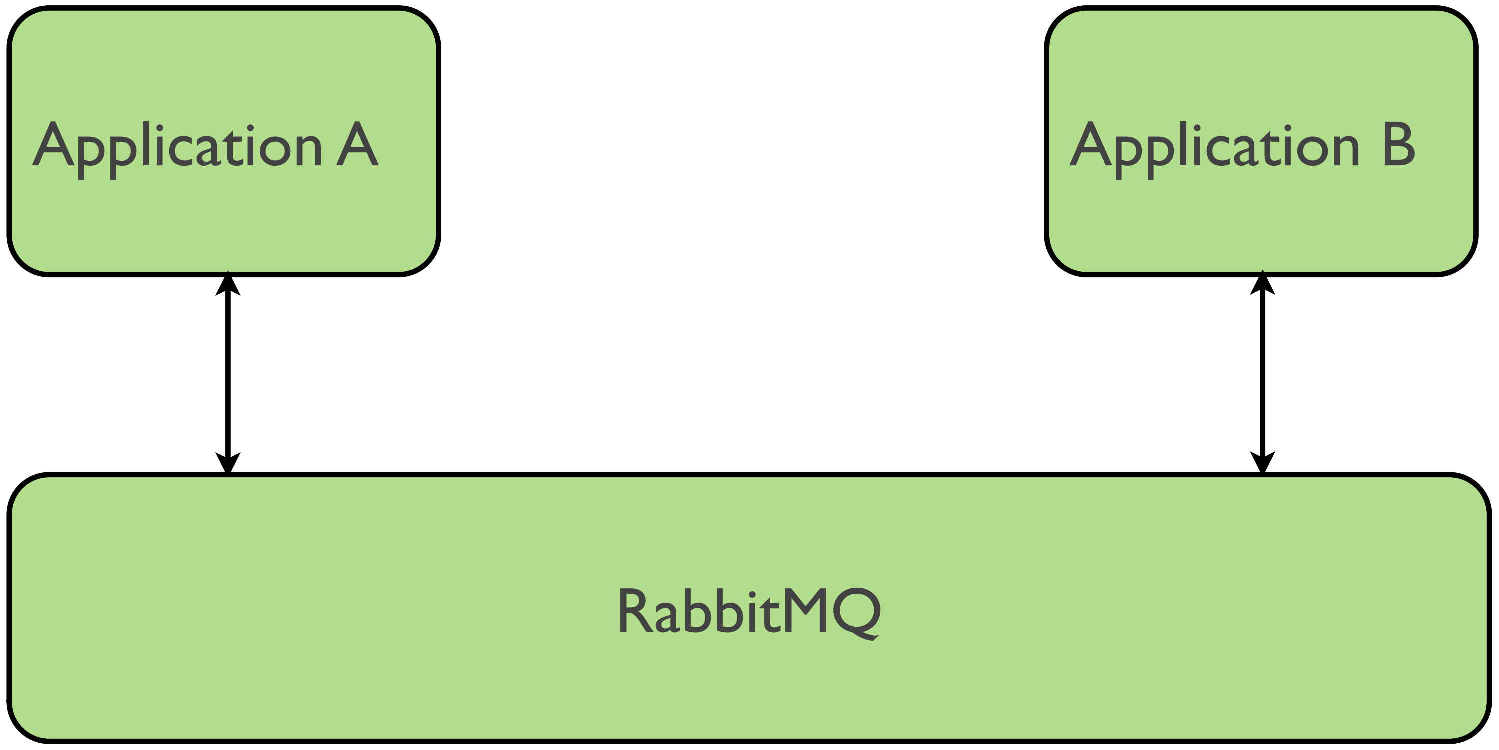
```
Chris-Richardsons-Mac-Pro:~ cer$ vmc services

============== System Services ==============

+------------+---------+---------------------------------------+
| Service    | Version | Description                           |
+------------+---------+---------------------------------------+
| postgresql | 9.0     | PostgreSQL database service (vFabric) |
| mysql      | 5.1     | MySQL database service                |
| rabbitmq   | 2.4     | RabbitMQ messaging service            |
| mongodb    | 1.8     | MongoDB NoSQL store                   |
| redis      | 2.2     | Redis key-value store service         |
+------------+---------+---------------------------------------+
```

Cloud Foundry provides RabbitMQ - aaS



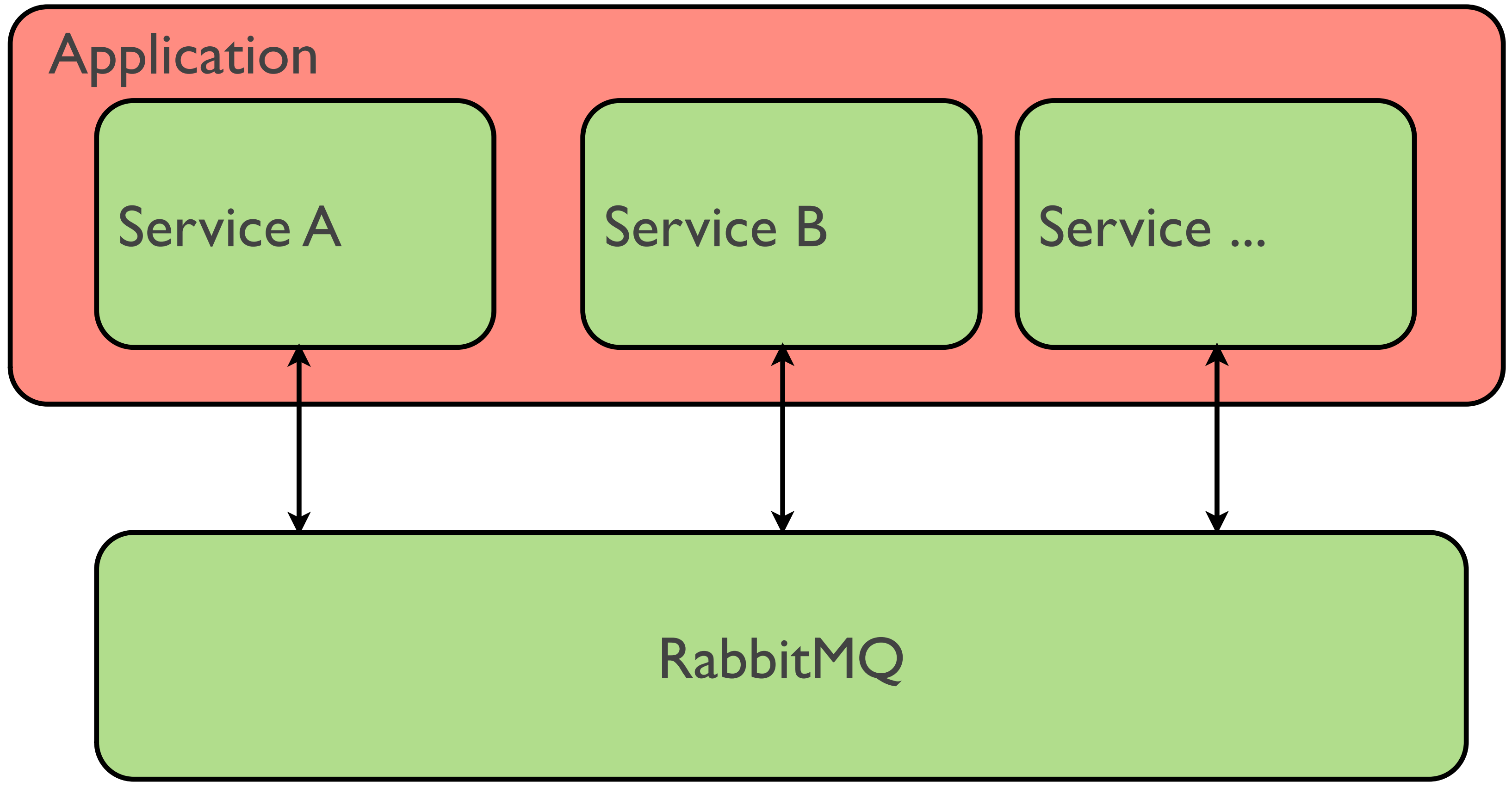
```
Chris-Richardsons-Mac-Pro:~ cer$ vmc create-service rabbitmq myrabbitmq
Creating Service: OK

Chris-Richardsons-Mac-Pro:~ cer$
```

But why messaging? Why RabbitMQ?



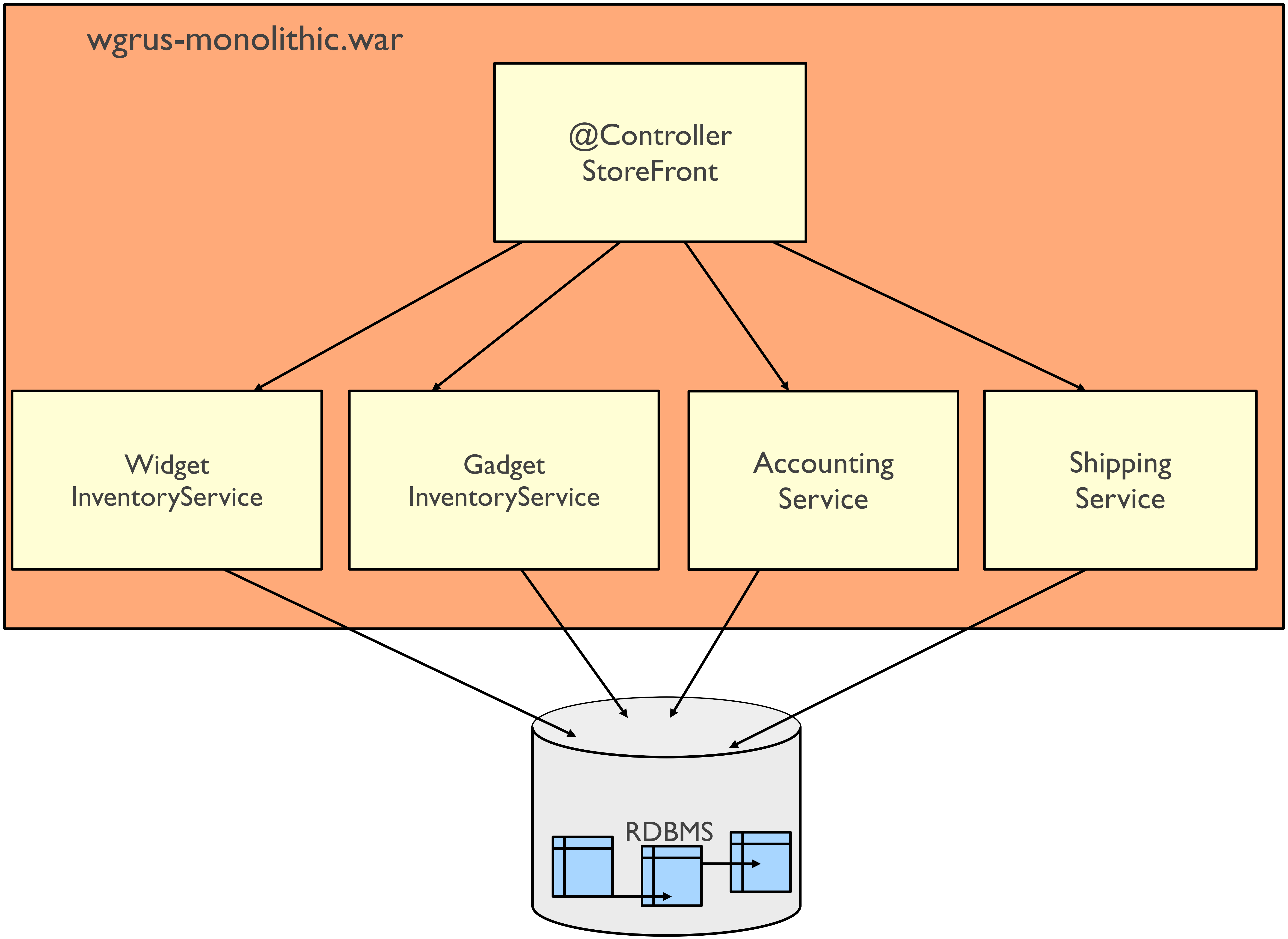
Traditional application integration

But why messaging? Why RabbitMQ?



- Essential component of our new scalable and fault tolerant architecture
- Integration mechanism for the services
- Enables services to discover each other

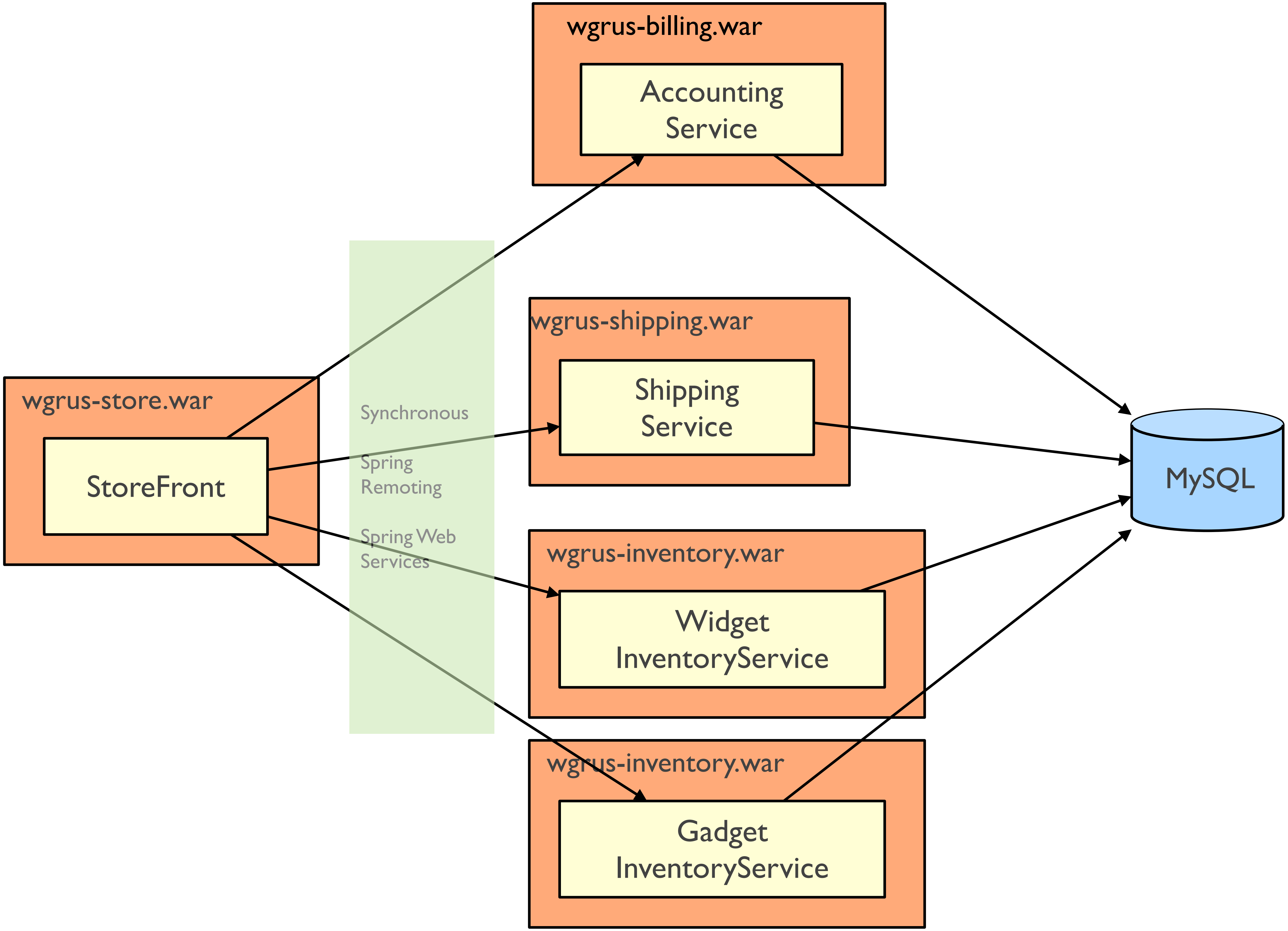
Let’s imagine you are building an e-commerce application


wgrus-monolithic.war
@Controller StoreFront
Widget InventoryService
Gadget InventoryService
Accounting Service
Shipping Service
RDBMS

It’s simple to develop but ....

- **Lack of scalability**
  - Scale through replication
  - Non-replicable component => nothing can be replicated
  - Can’t scale different parts of the application differently
- **Lack of deployability**
  - Deploy it all in one go
  - Increased risk of something breaking
- **Applications are brittle**
  - Store can’t accept orders unless all services are available
  - Failure (e.g. memory leak) in one component can take down every other
- **Monolingual**
  - Can’t use non-JVM server-side technologies: NodeJS, Rails,

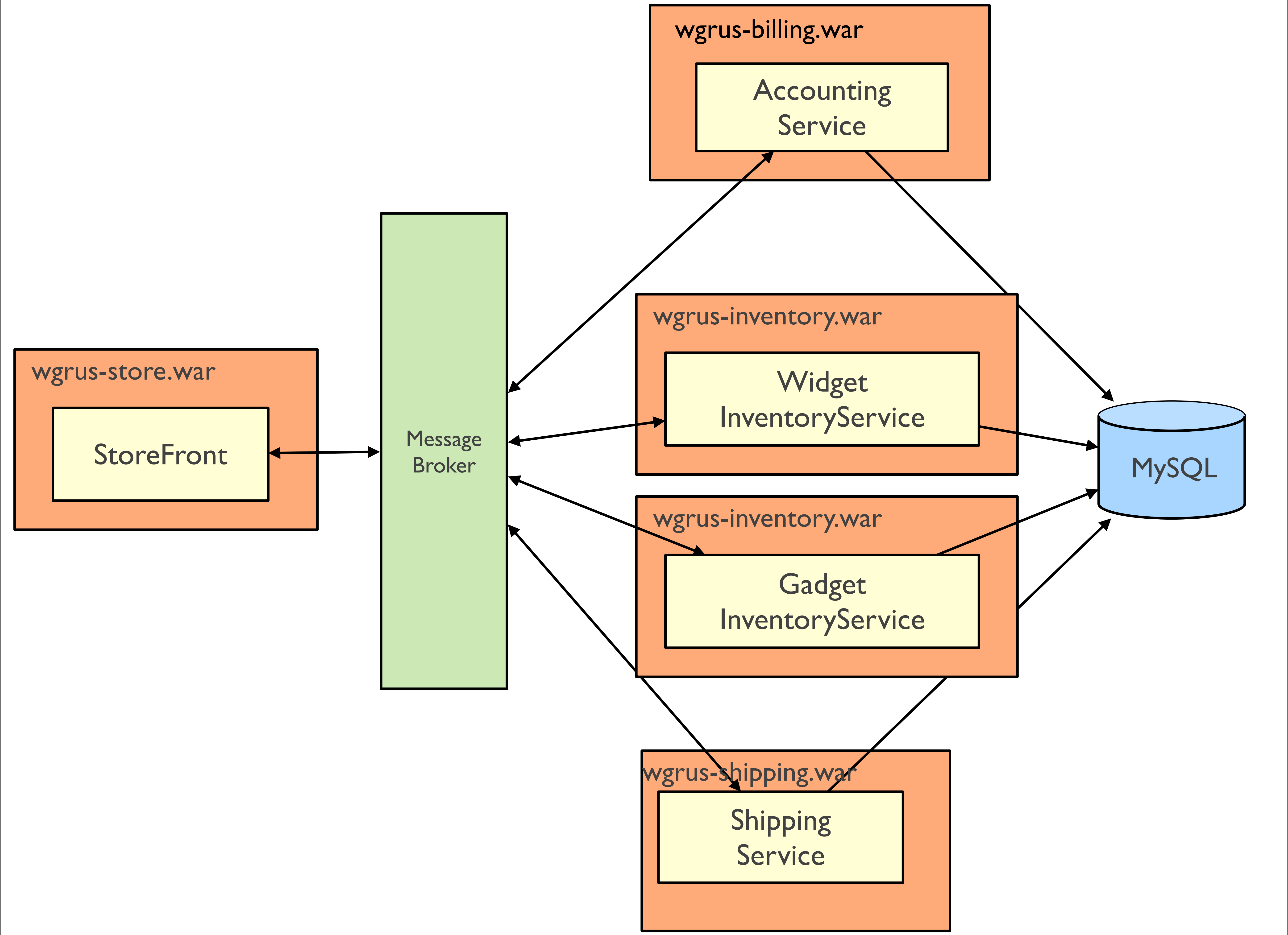
Decompose application into services
By noun or by verbs


wgrus-billing.war
Accounting Service
wgrus-shipping.war
Shipping Service
wgrus-store.war
StoreFront
Synchronous
Spring Remoting
Spring Web Services
wgrus-inventory.war
Widget InventoryService
wgrus-inventory.war
Gadget InventoryService
MySQL

Benefits and Drawbacks

- **Benefits:**
  - Scale each service independently
  - Deploy each service independently
  - Mix JVM and non-JVM languages
- **Drawbacks**
  - Application is still brittle
    - Store can’t accept orders unless all services are available
    - Failure (e.g. memory leak) in one component can take down every other

Solution:
Asynchronous Architecture


wgrus-billing.war
Accounting Service
wgrus-inventory.war
Widget InventoryService
wgrus-store.war
StoreFront
Message Broker
MySQL
wgrus-inventory.war
Gadget InventoryService
wgrus-shipping.war
Shipping Service

Benefits and Drawbacks

- **Benefits:**
  - Scale each service independently
  - Deploy each service independently
  - Mix JVM and non-JVM languages
  - **Improved availability**
    - **Front-end keeps working even when backend services are down**
    - **Messaging broker can buffer traffic and smooth out spikes**
- **Drawbacks**
  - Yet another moving part
  - Sometimes synchronous RPC is a better fit

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- Developing NoSQL applications for Cloud Foundry
- **Application integration with RabbitMQ and Spring AMQP**
  - Why messaging?
  - **Messaging with RabbitMQ and AMQP**
  - Using Spring Integration
  - Cloud Foundry and RabbitMQ
- Wrap up

RabbitMQ – Messaging that Just Works

Robust
High-performance
Easy to use
AMQP LEADER

Why AMQP?

A Protocol, not an API

•A **defined set of messaging capabilities** called the AMQ model

•A **network wire-level protocol,** AMQP

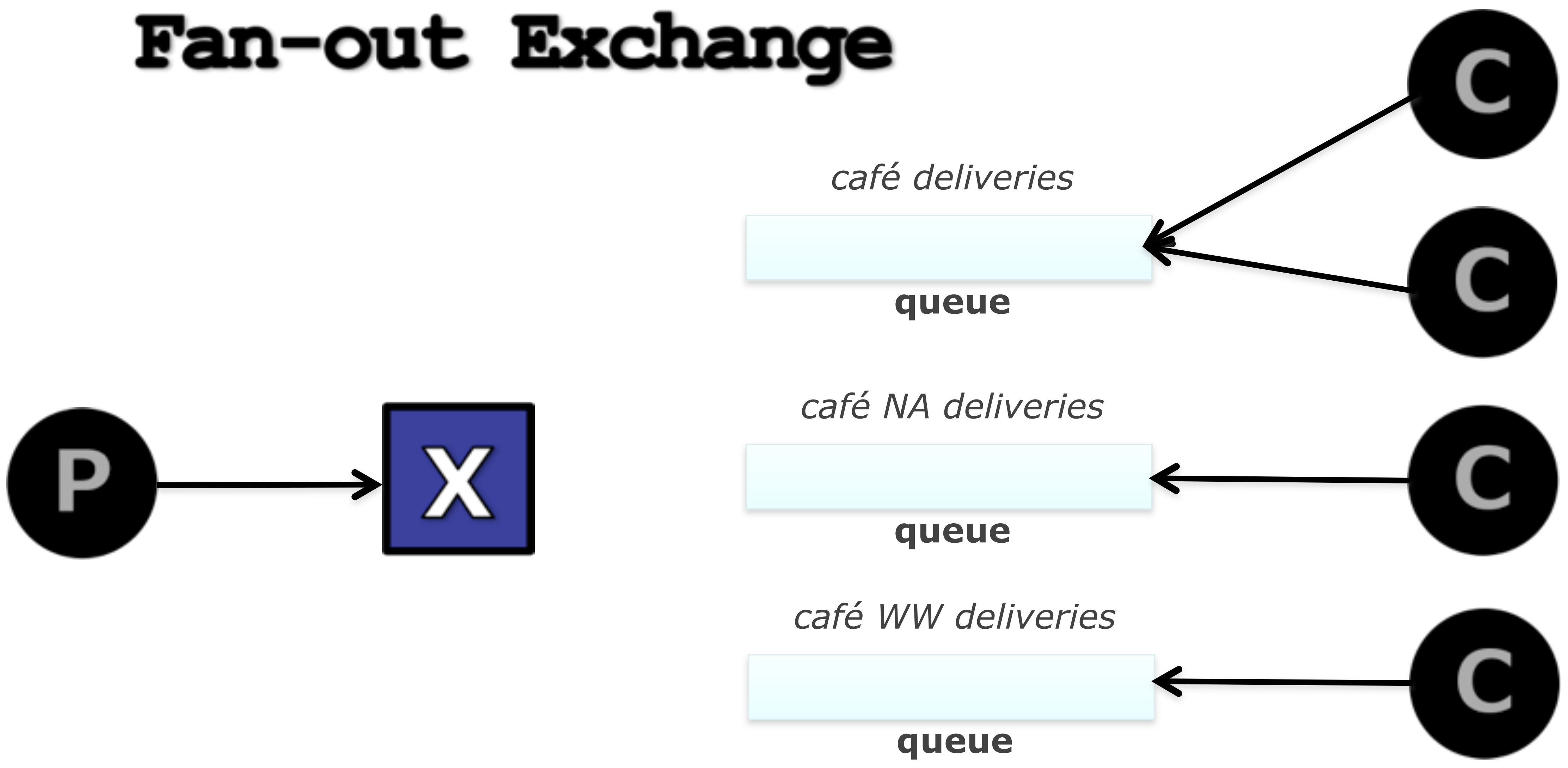
On commodity hardware

•10-25 thousand messages per second is routine *

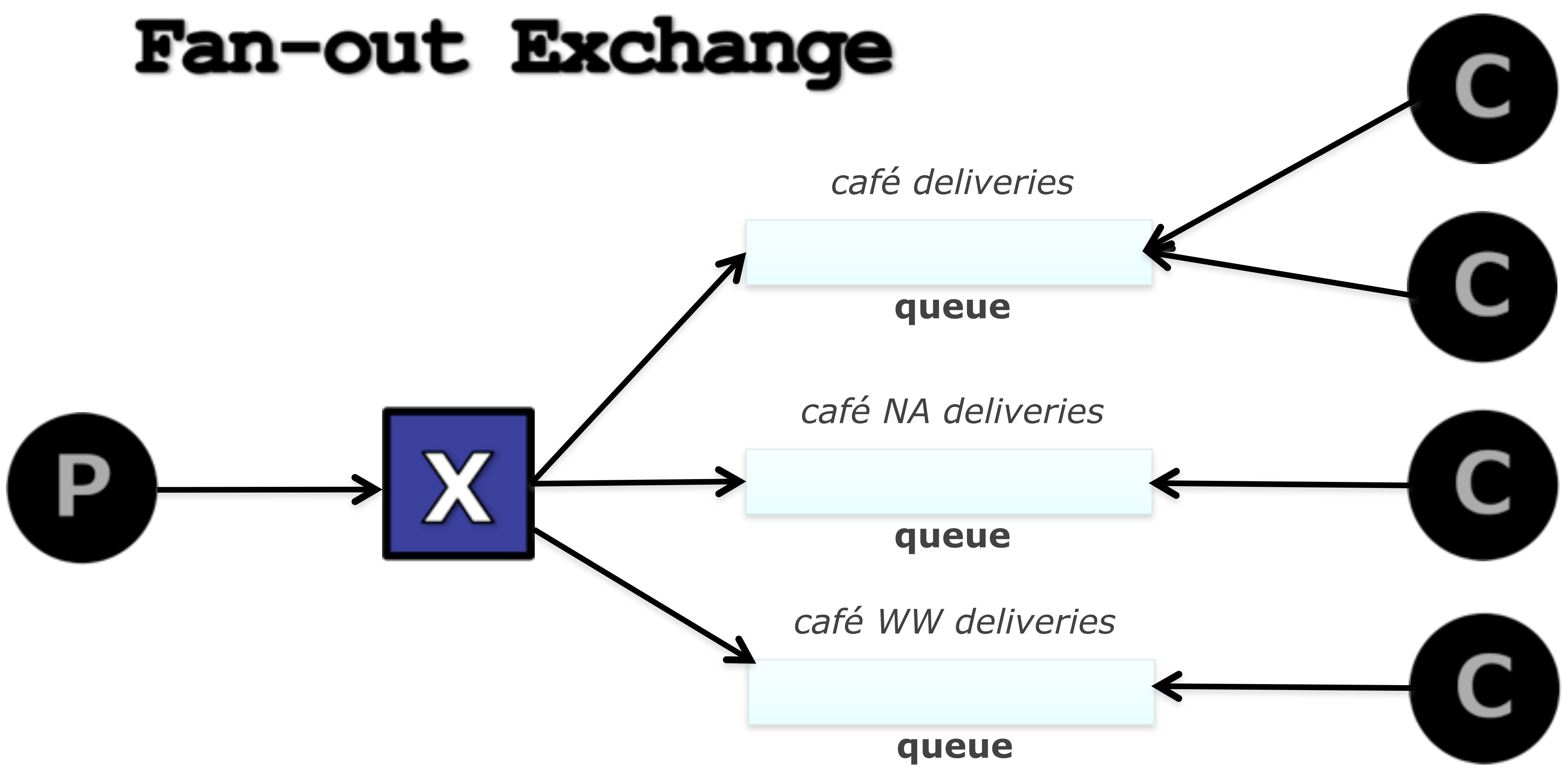
•The NIC is usually the bottleneck

* Non-persistent messages

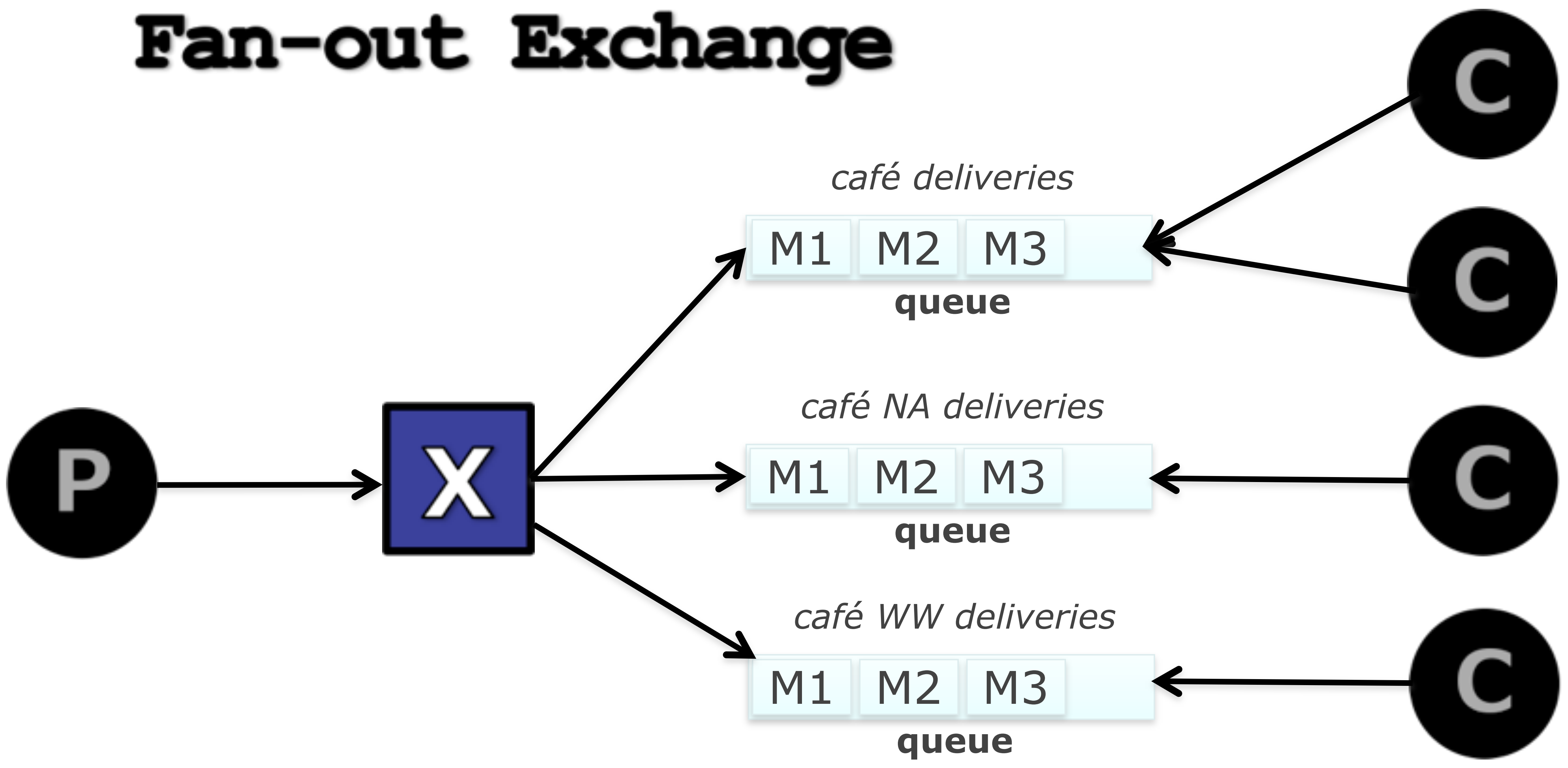
AMQP Architecture



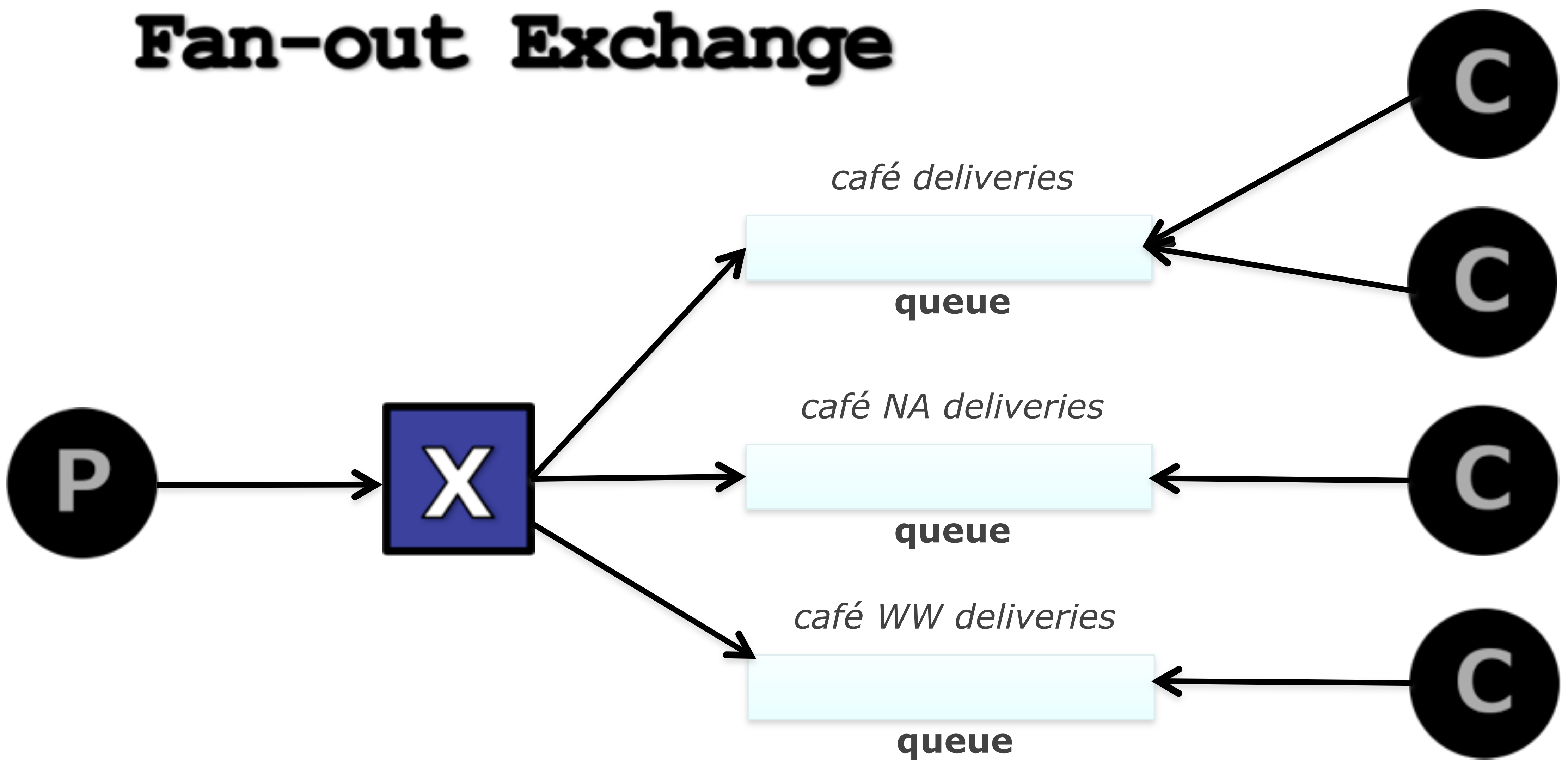
AMQP Architecture



AMQP Architecture



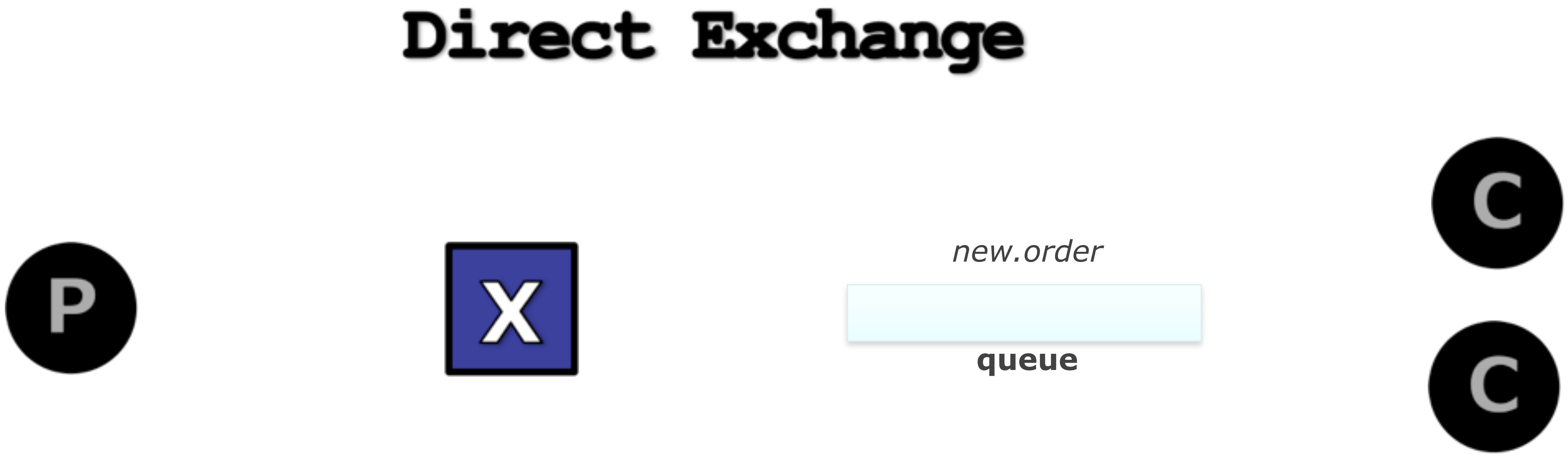
AMQP Architecture



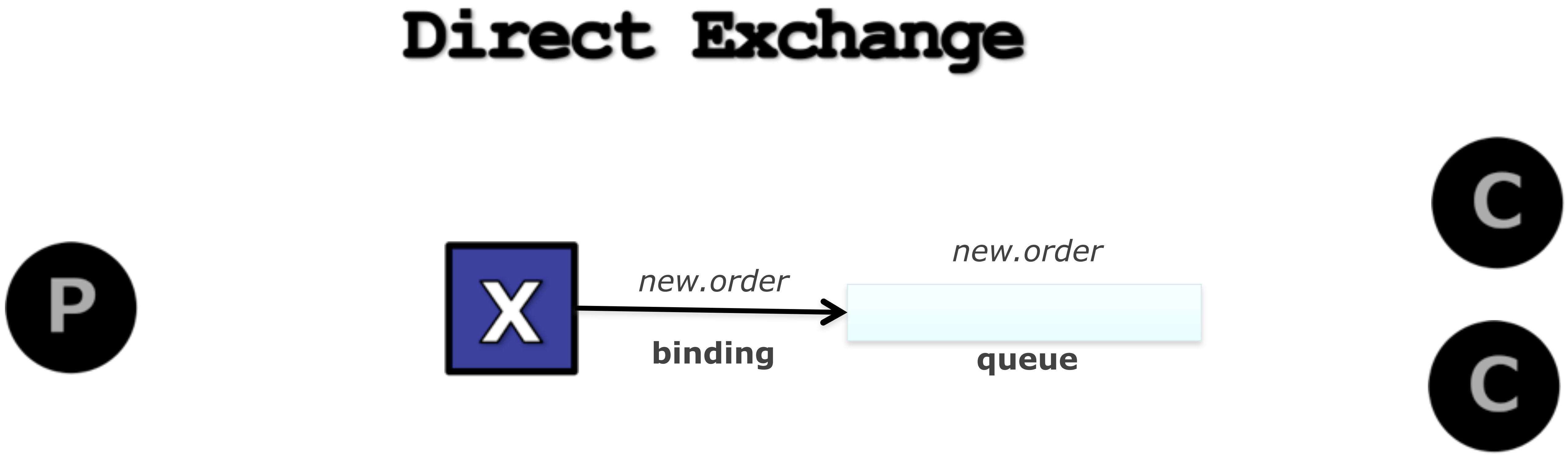
AMQP Architecture

Direct Exchange

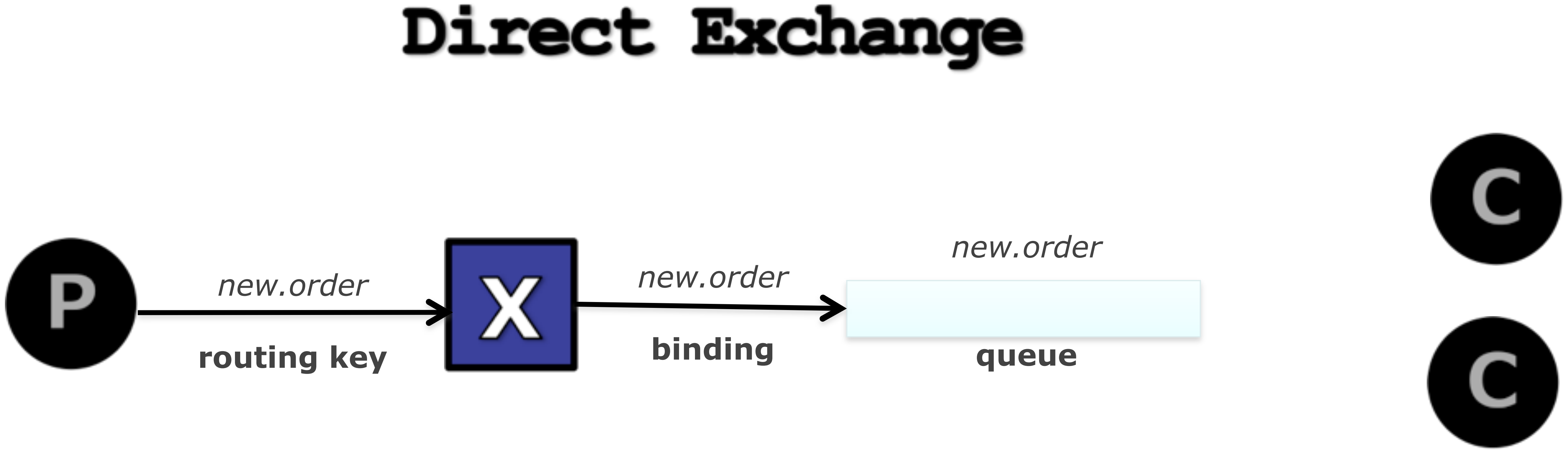
AMQP Architecture



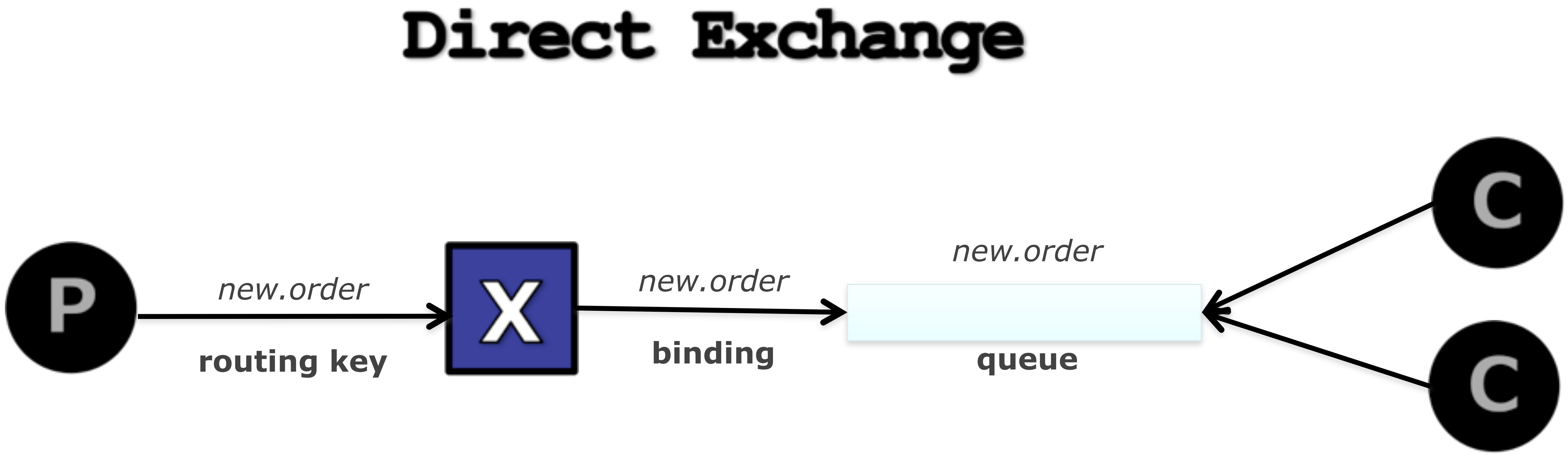
AMQP Architecture



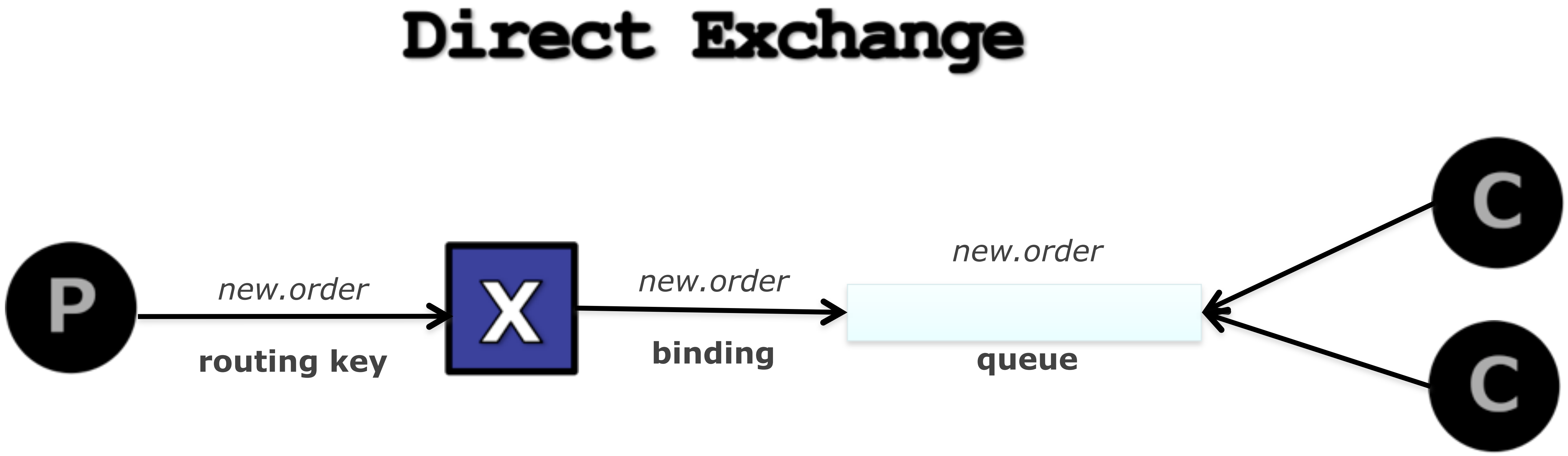
AMQP Architecture



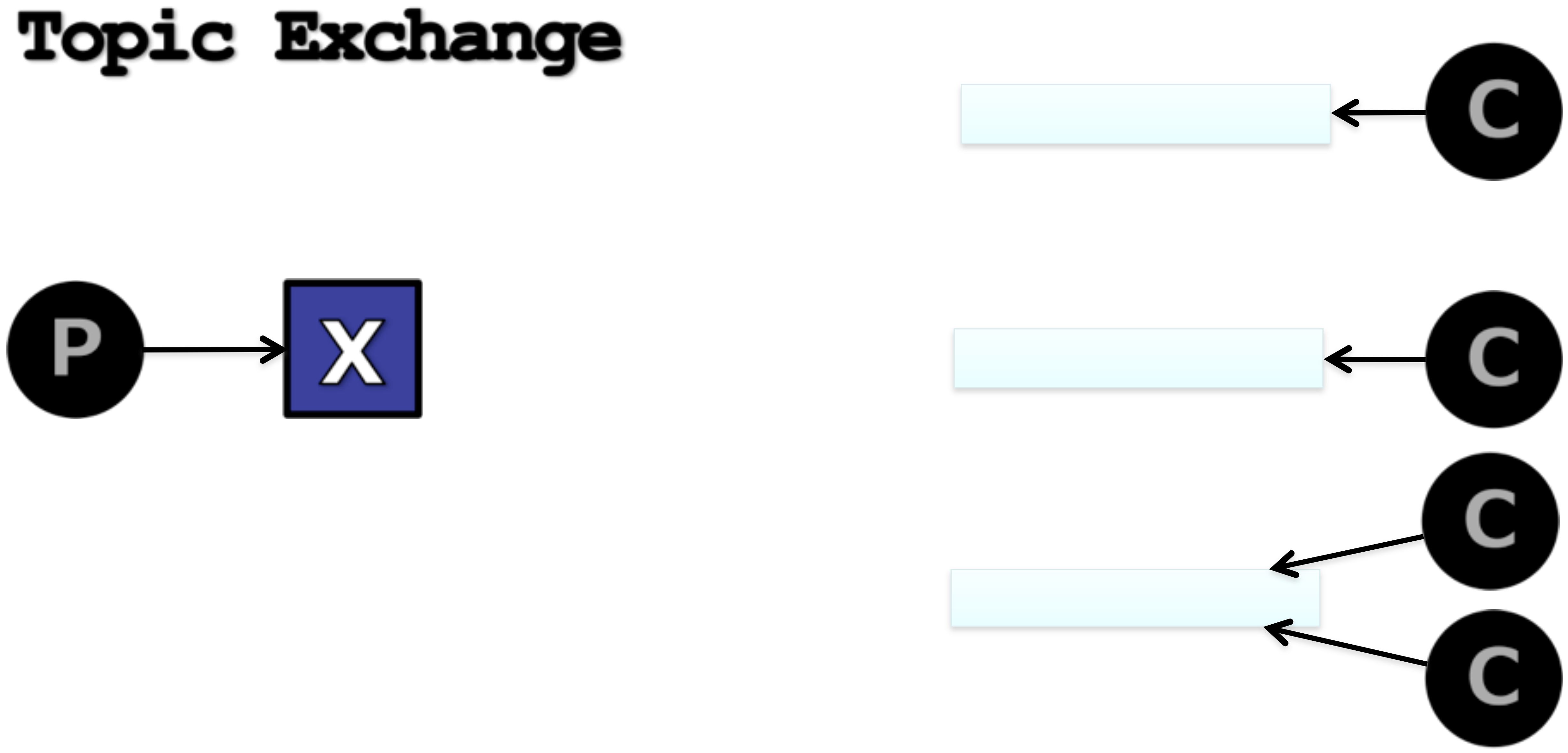
AMQP Architecture



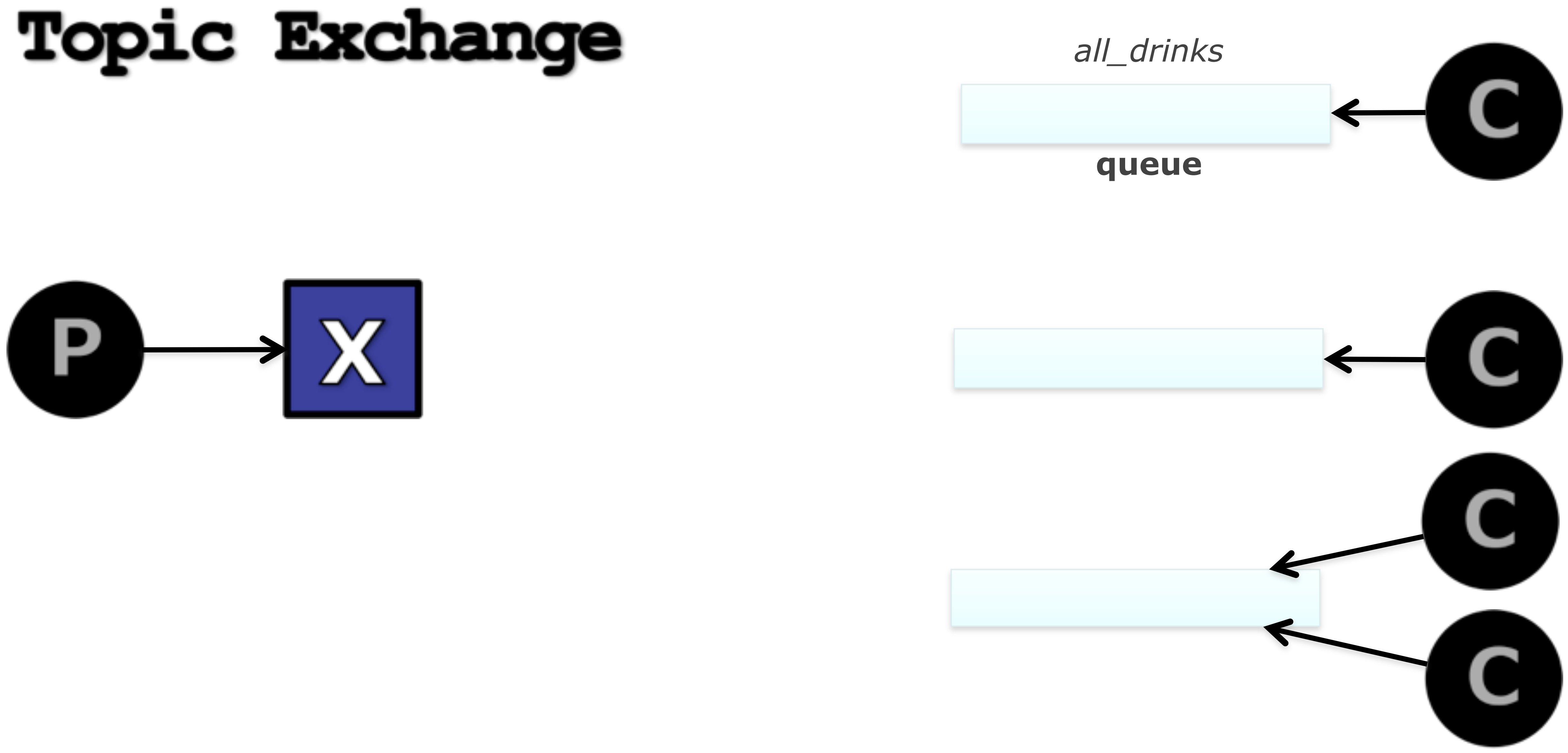
AMQP Architecture



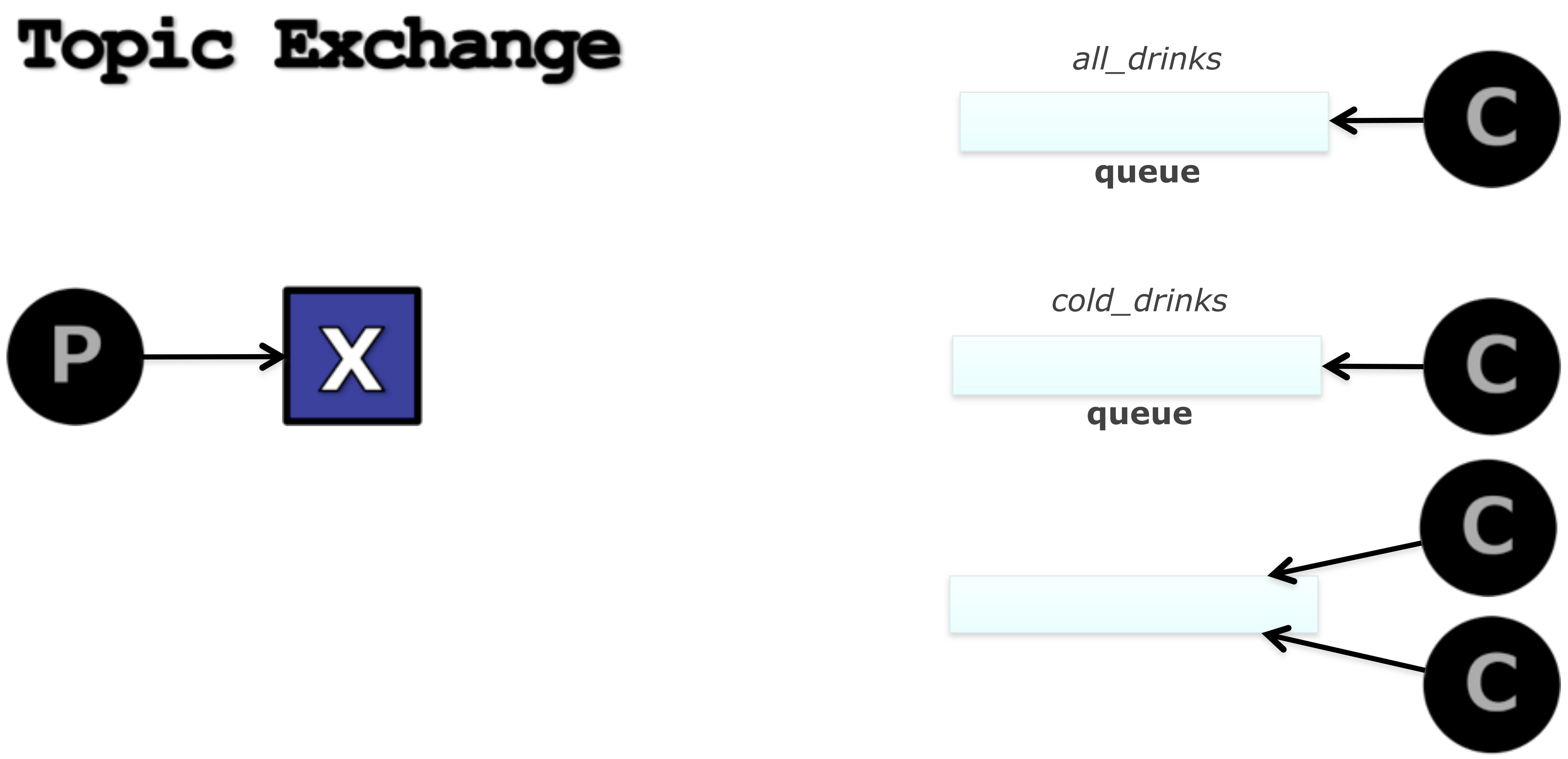
AMQP Architecture



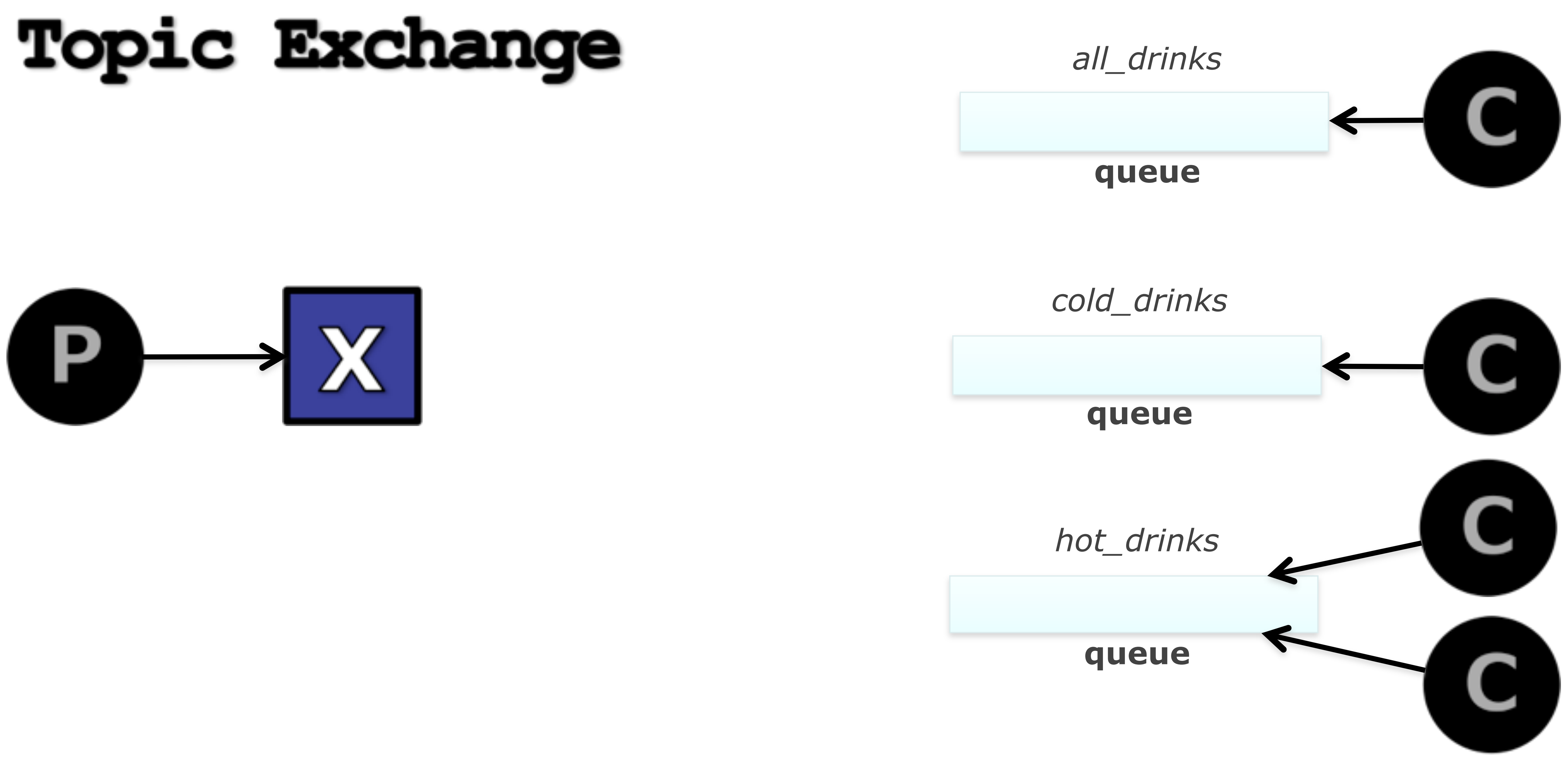
AMQP Architecture



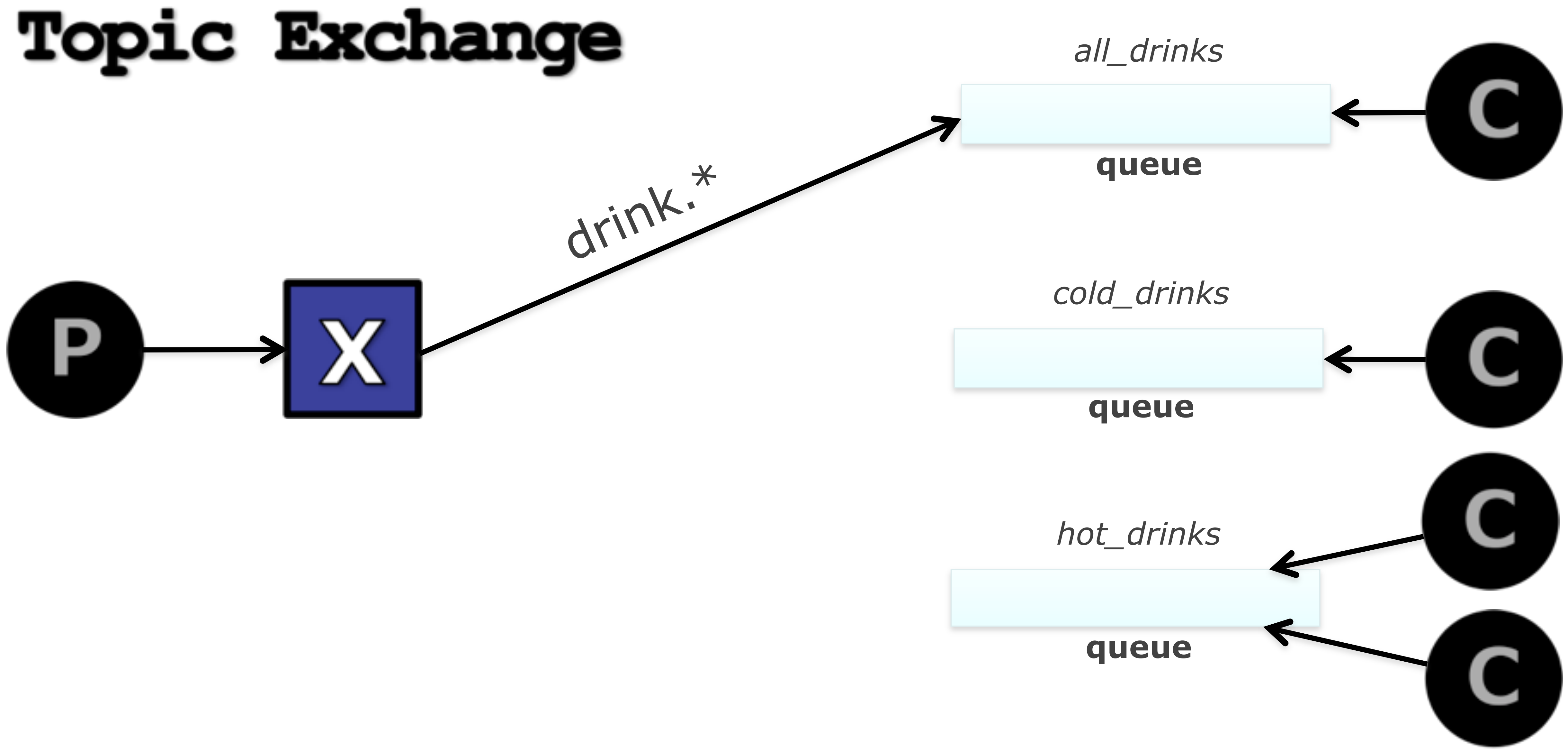
AMQP Architecture



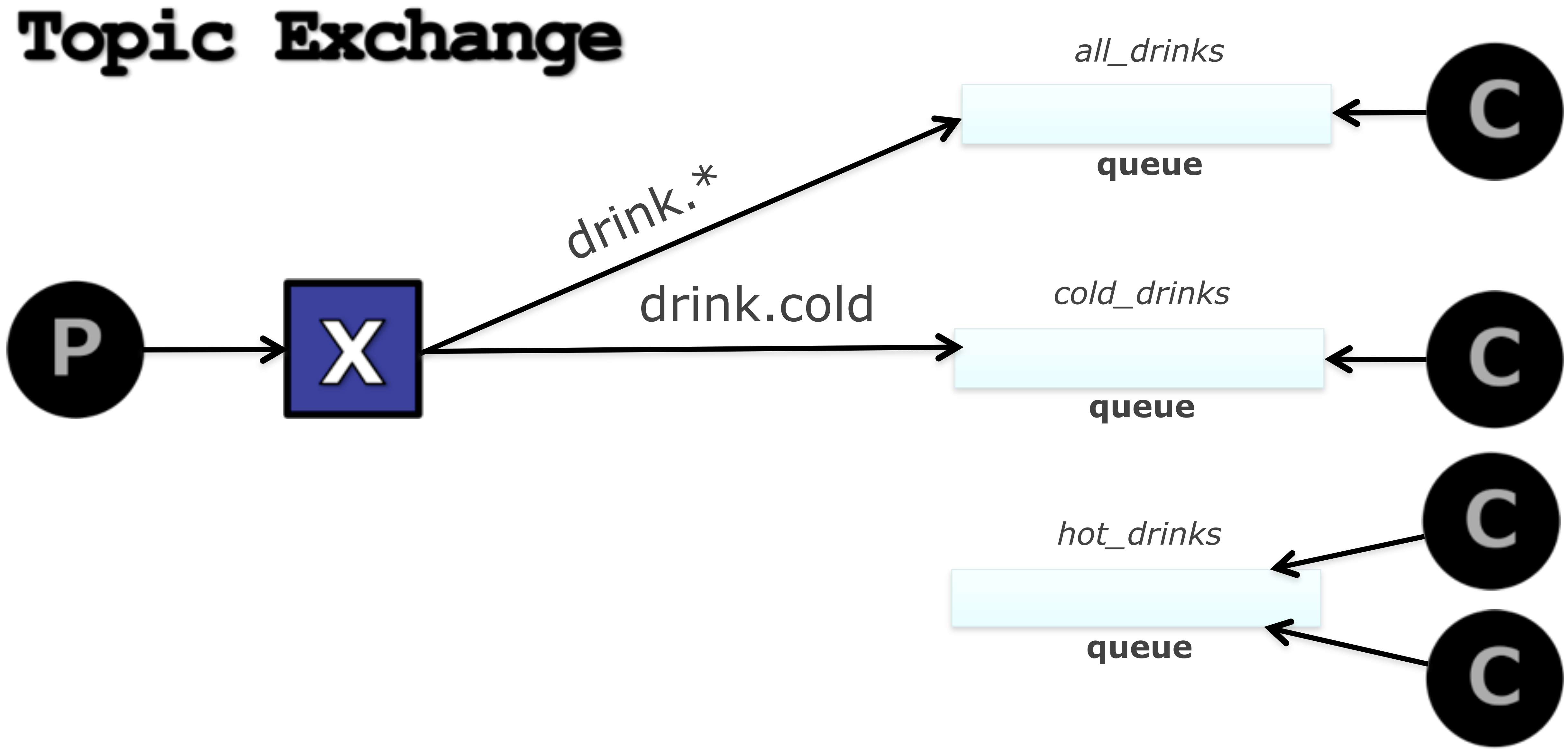
AMQP Architecture



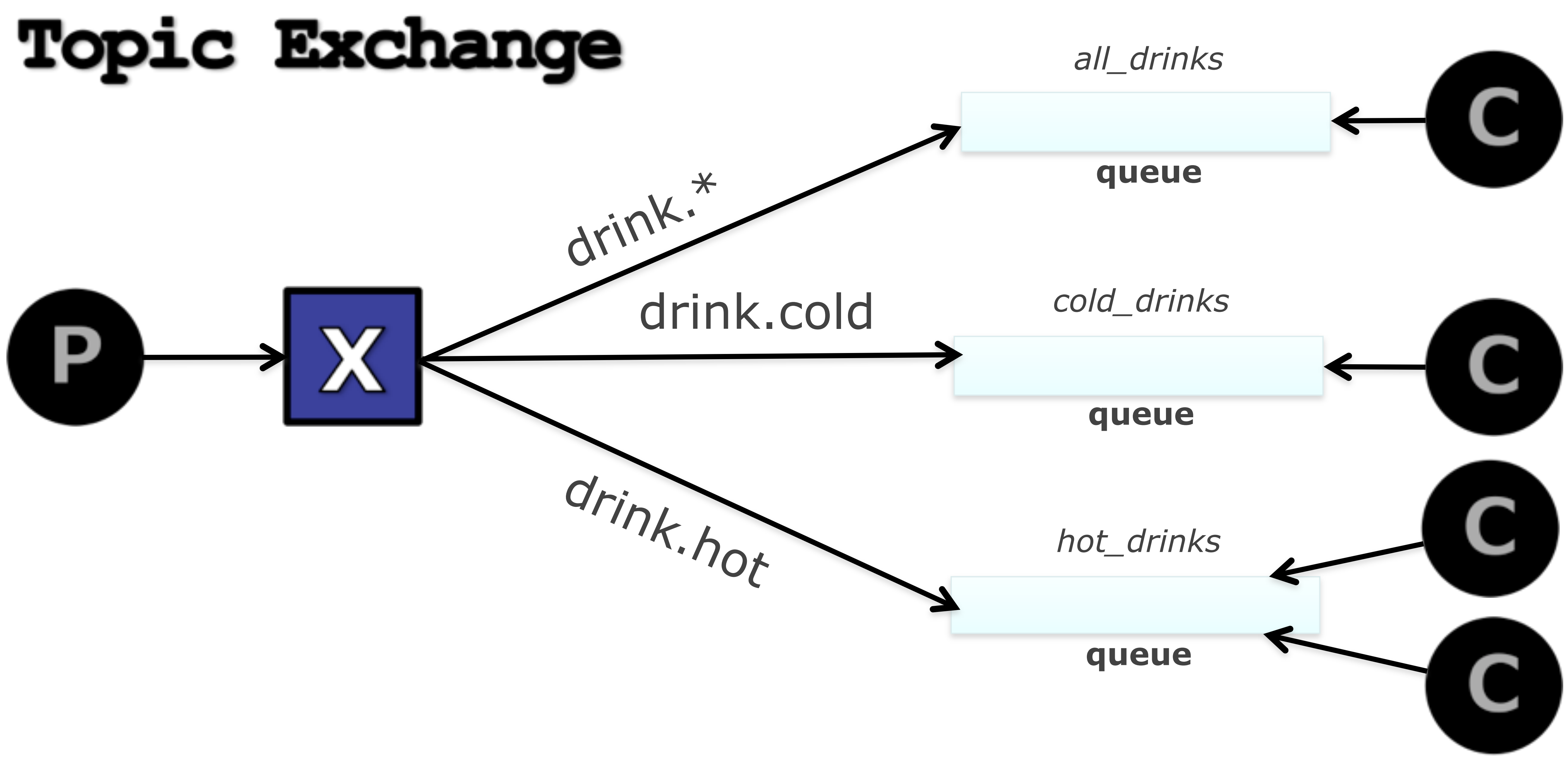
AMQP Architecture



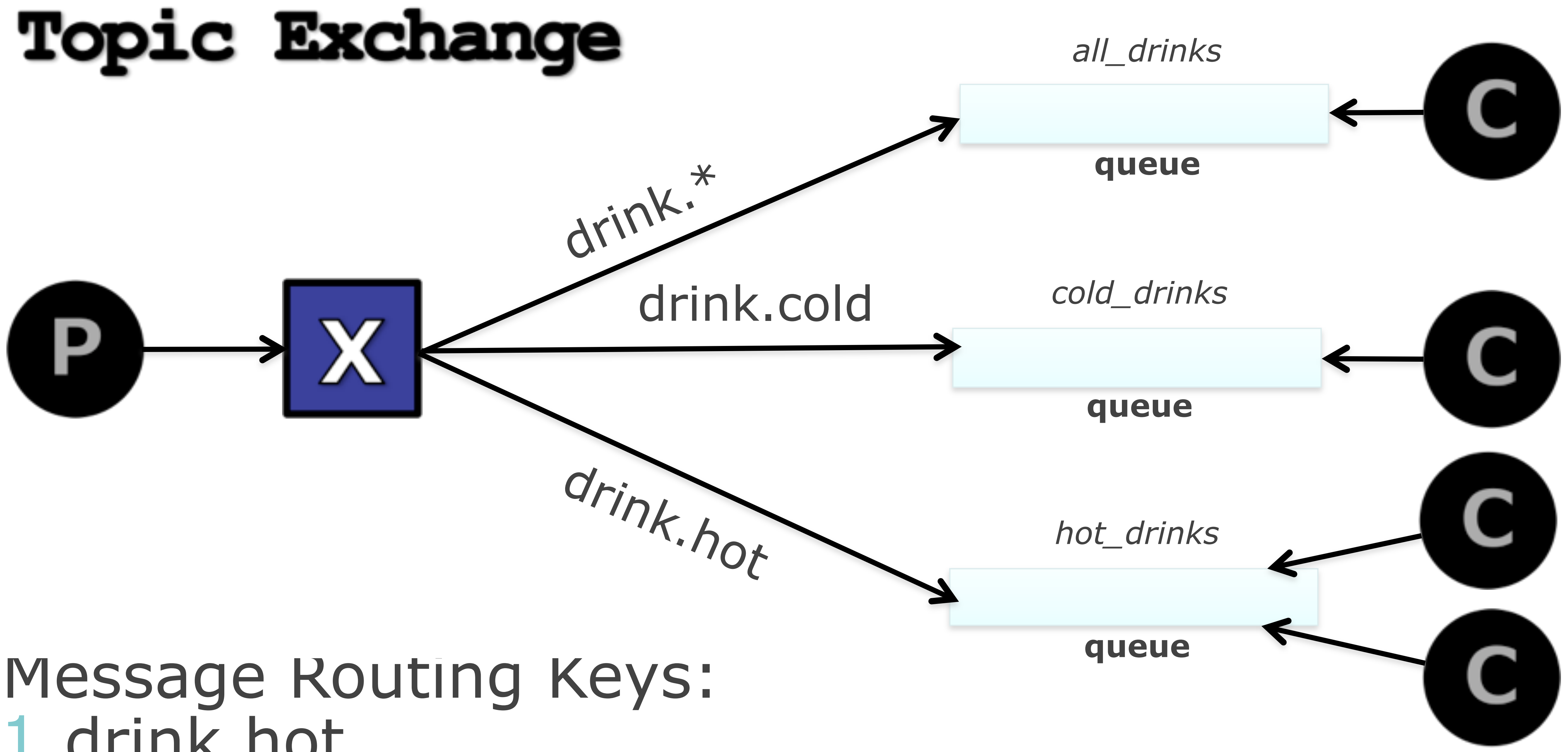
AMQP Architecture



AMQP Architecture

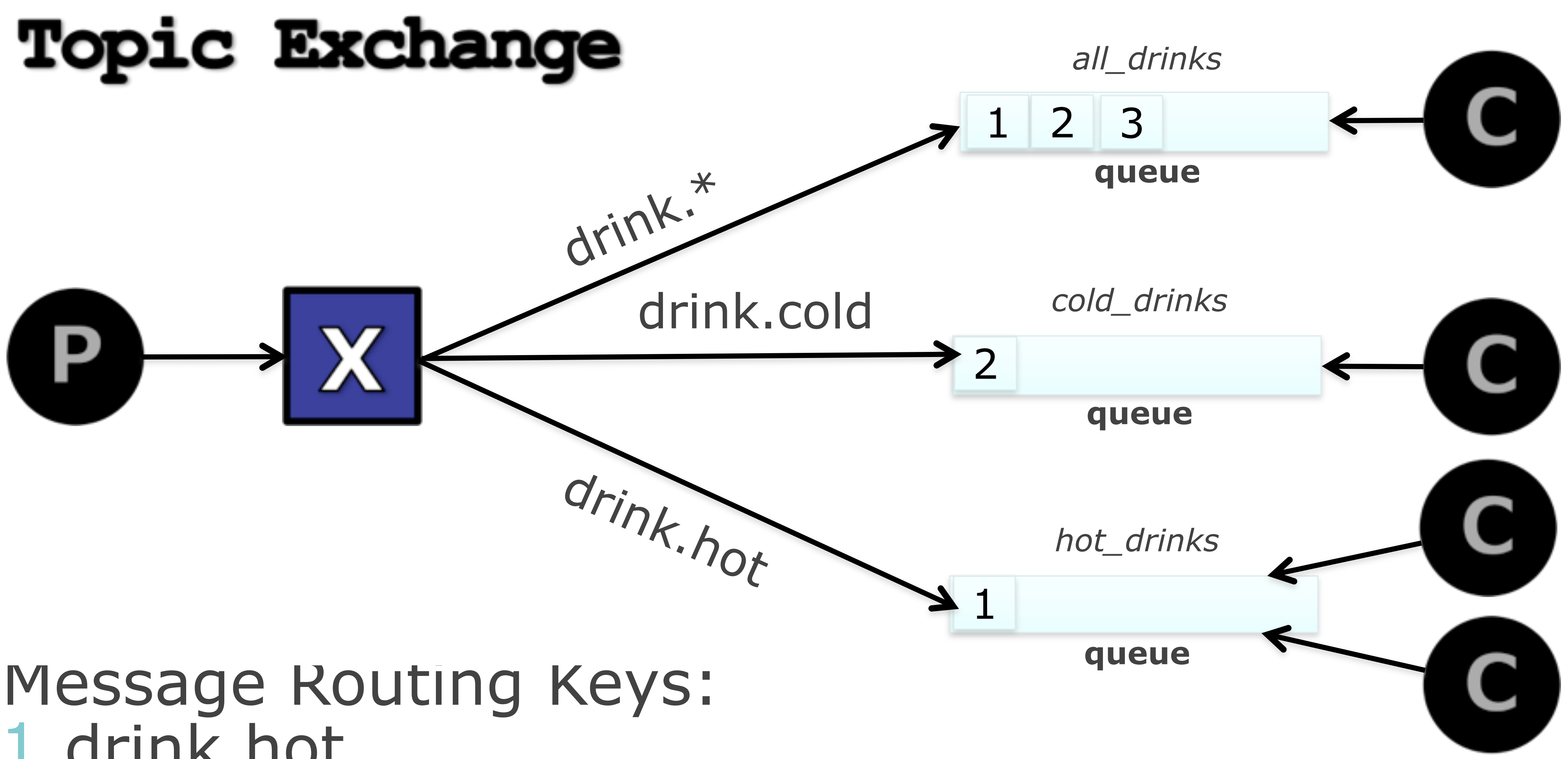


AMQP Architecture



Message Routing Keys:
1. drink.hot
2. drink.cold
3. drink.warm

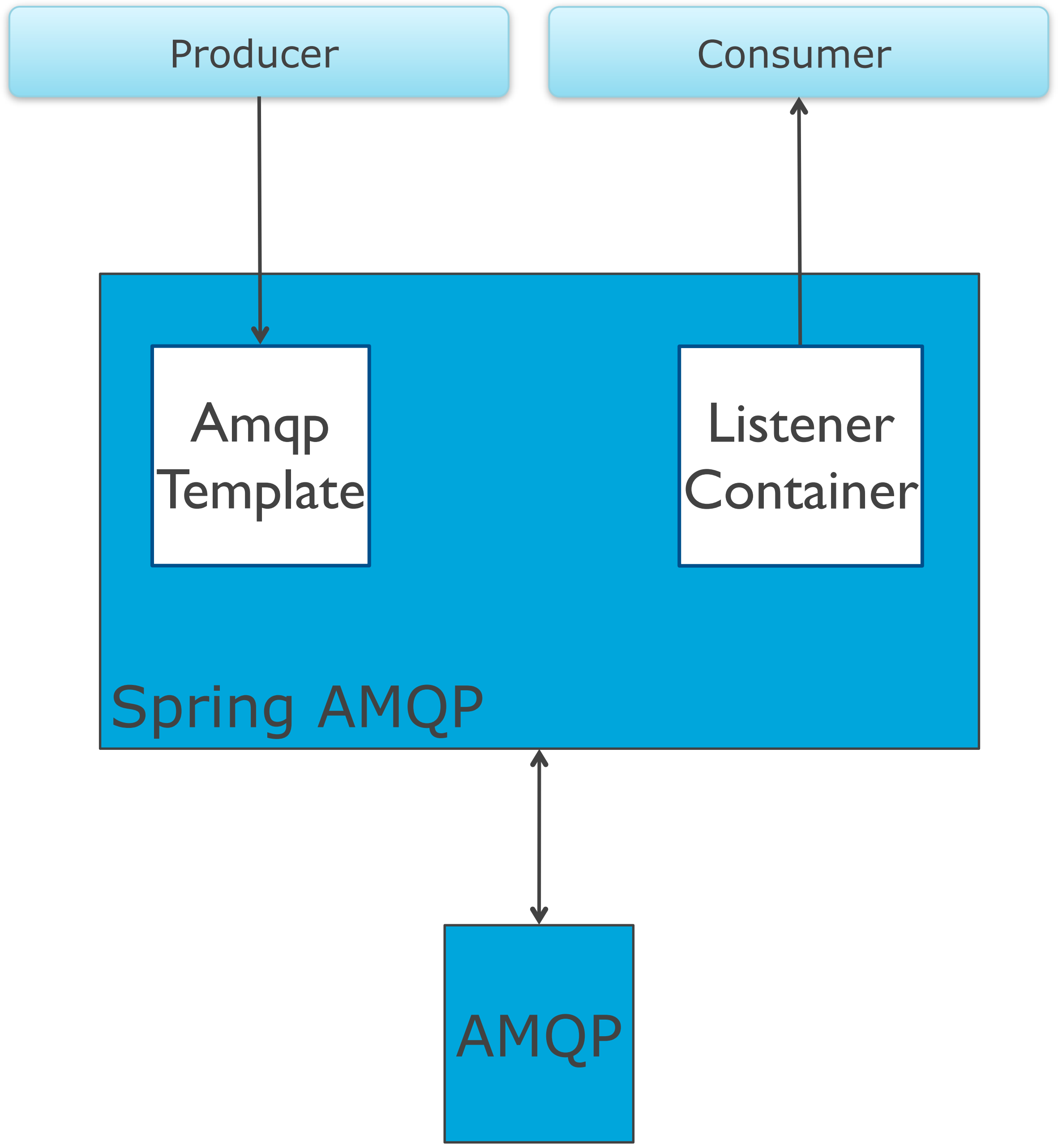
AMQP Architecture



Message Routing Keys:
1. drink.hot
2. drink.cold
3. drink.warm

Spring AMQP

- **Encapsulates low-level details**
- **Simplifies sending and receiving of messages**



Sending AMQP messages

```
@Component public class MessageSender {

  @Autowired
  private volatile AmqpTemplate amqpTemplate;

  public void send(String message) {
    this.amqpTemplate.convertAndSend(
            "myExchange", "some.routing.key", message);
  }

}
```

Receiving AMQP messages

```
public class MyComponent {

   @Autowired
   private AmqpTemplate amqpTemplate;

    public void read() throws Exception {
         ...
        String value = amqpTemplate.receiveAndConvert("myQueueName");
        ...
    }

}
```

Spring AMQP: SimpleMessageListenerContainer

- Asynchronous message receiver
- POJO handlers
- Handles re-connection and listener failure (rollback, redelivery)
- Message conversion and error handling strategies

```
<listener-container connection-factory="rabbitConnectionFactory">
 <listener ref="handler" method="handle" queue-names="my.queue">
</listener-container>
```

Spring configuration

```
<rabbit:template id="rabbitTemplate"
   connection-factory="rabbitConnectionFactory"/>

<rabbit:connection-factory
   id="rabbitConnectionFactory"/>
```

Spring AMQP is flexible and dynamic

BUT

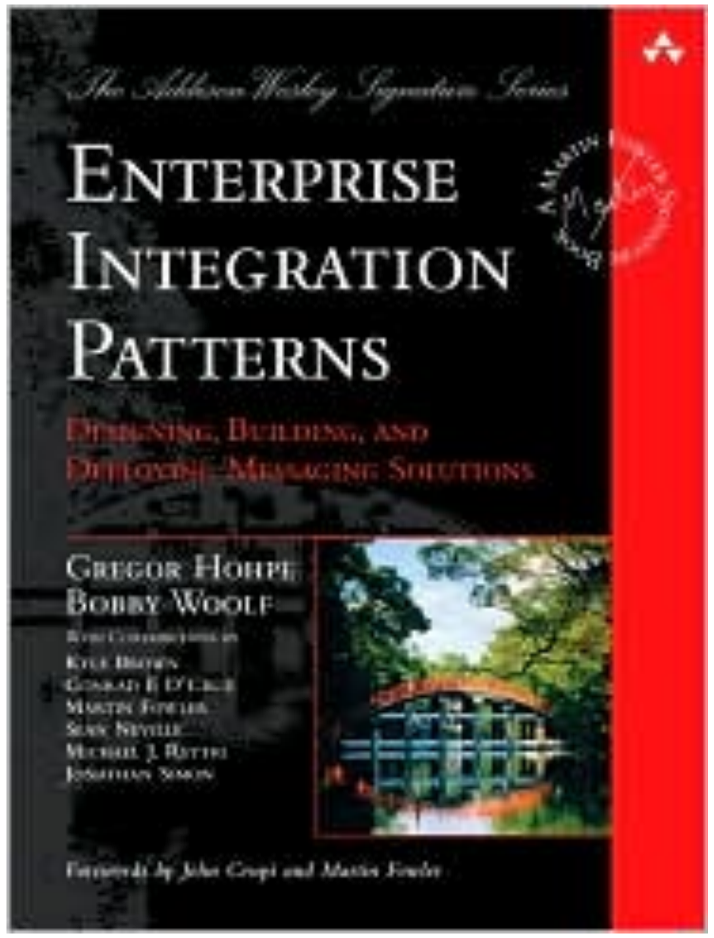
It’s very low level

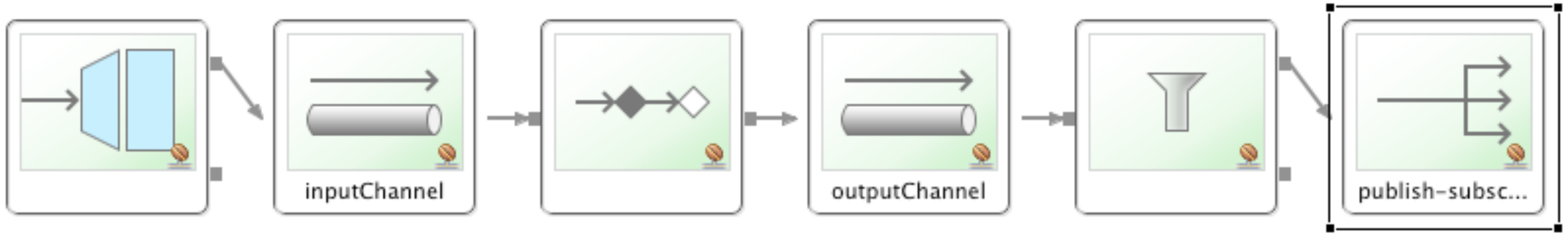
Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- Developing NoSQL applications for Cloud Foundry
- **Application integration with RabbitMQ and Spring AMQP**
  - Why messaging?
  - Messaging with RabbitMQ and AMQP
  - **Using Spring Integration**
  - Cloud Foundry and RabbitMQ
- Wrap up

Spring Integration

- Builds on Spring framework
- High-level of abstraction for building message based applications
- Implements EAI patterns
- Provides plumbing for exchanging messages between application components
- Promotes loosely coupled components
- Integrates with external messaging infrastructure: JMS, AMQP, HTTP, Email, File transfer


The Addison-Wesley Signature Series
Enterprise Integration Patterns
Designing, Building, and Deploying Messaging Solutions
Gregor Hohpe
Bobby Woolf
Forewords by John Crupi and Martin Fowler



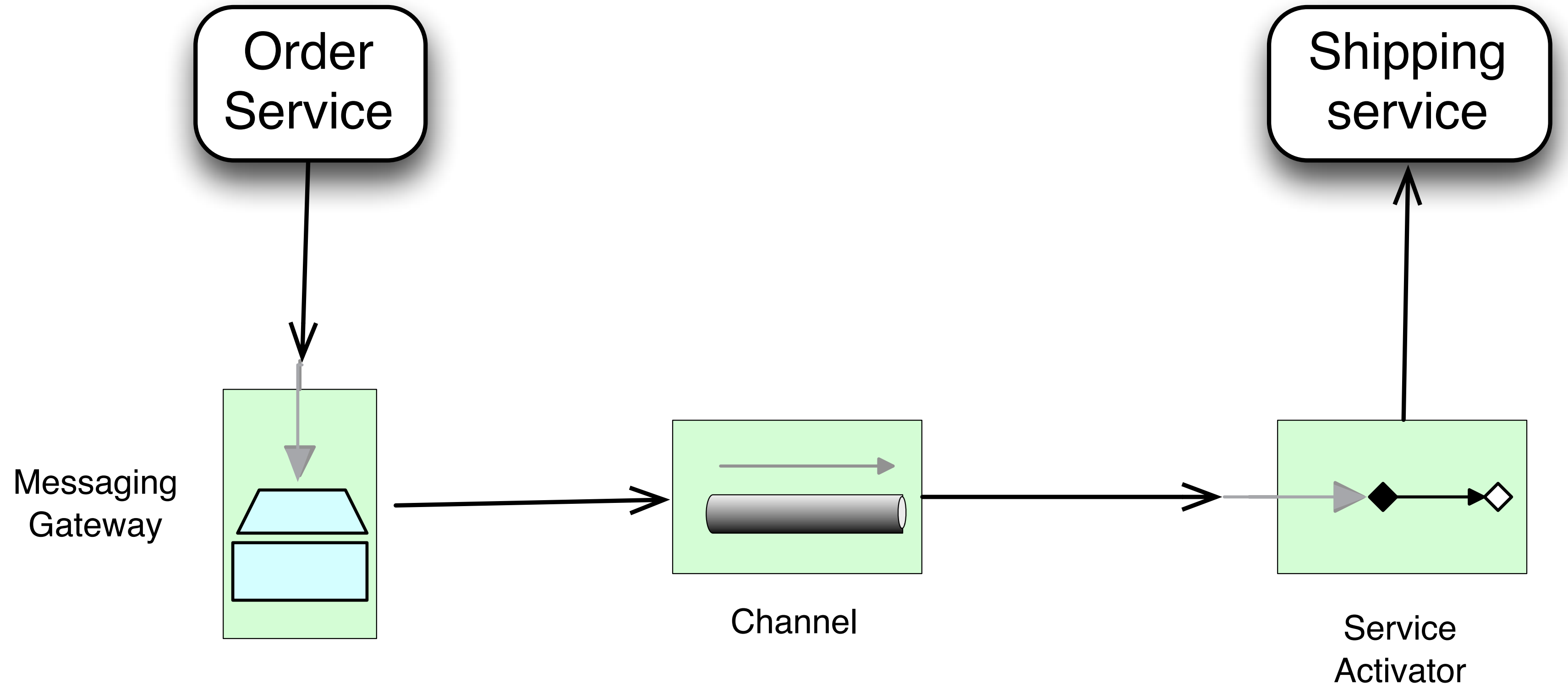
Spring Integration concepts

- **Message channel**
  - Virtual pipe connecting producer and consumer
- **Message endpoints**
  - The filter of a pipes-and-filter architecture
  - Read from and/or write to channel
- **Endpoint types:**
  - Transformer
  - Filter
  - Router
  - Splitter
  - Aggregator
  - ServiceActivator
  - Inbound channel adapter - read from external source, writes to channel
  - Outbound channel adapter - read from channel write to external destination

Example of reconfigurability - local

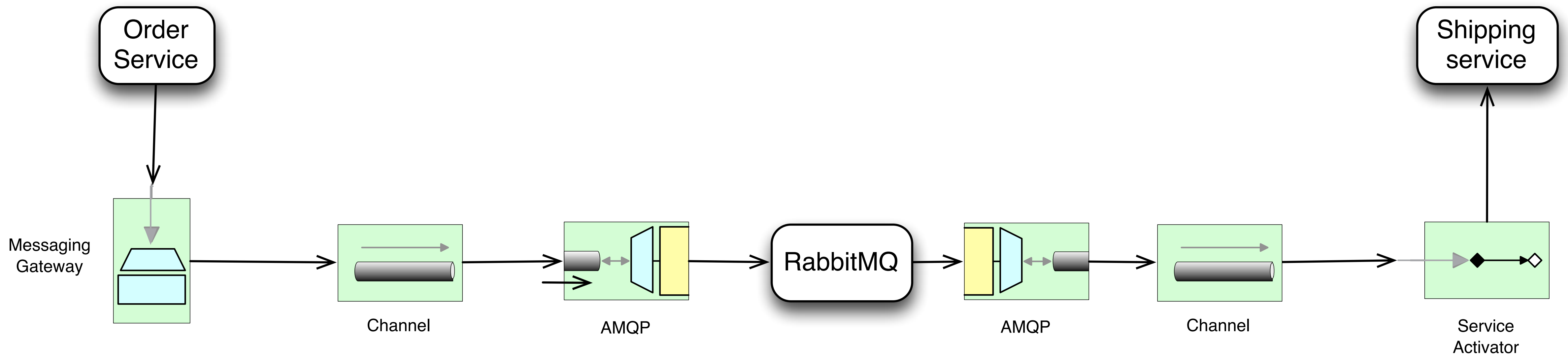
```
@Service
public class OrderServiceImpl {

@Autowired
private ShippingService shippingService;

public void placeOrder() {
String orderId = generateOrderId();
...
shippingService.shipOrder(orderId);
}

}
```

```
@Service
public class ShippingServiceImpl  {

public void shipOrder(String orderId) {
  System.out.println("shipped order: " +
                    orderId);
}

}
```

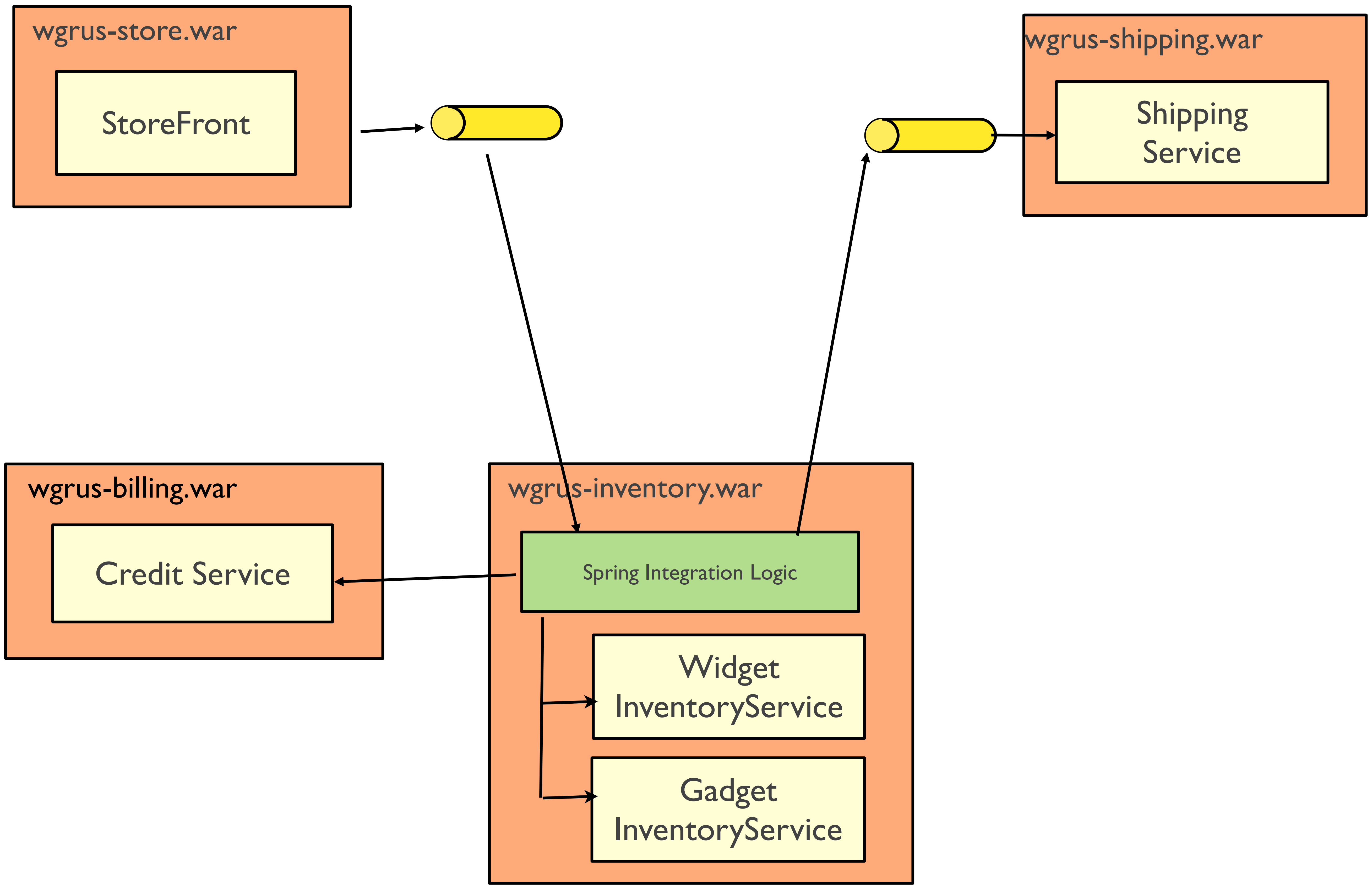


Example of reconfigurability - distributed

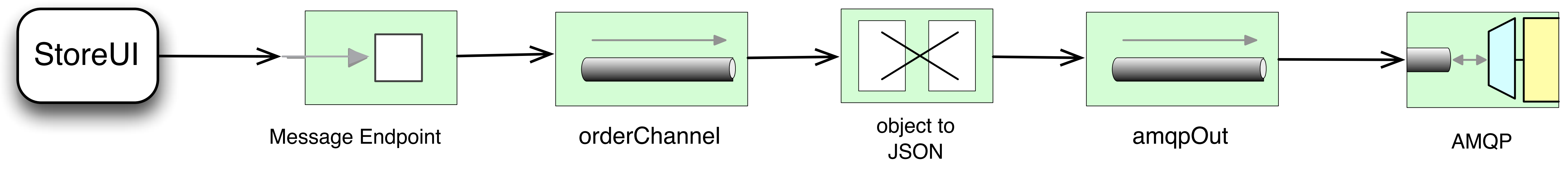
Code unchanged in new deployment



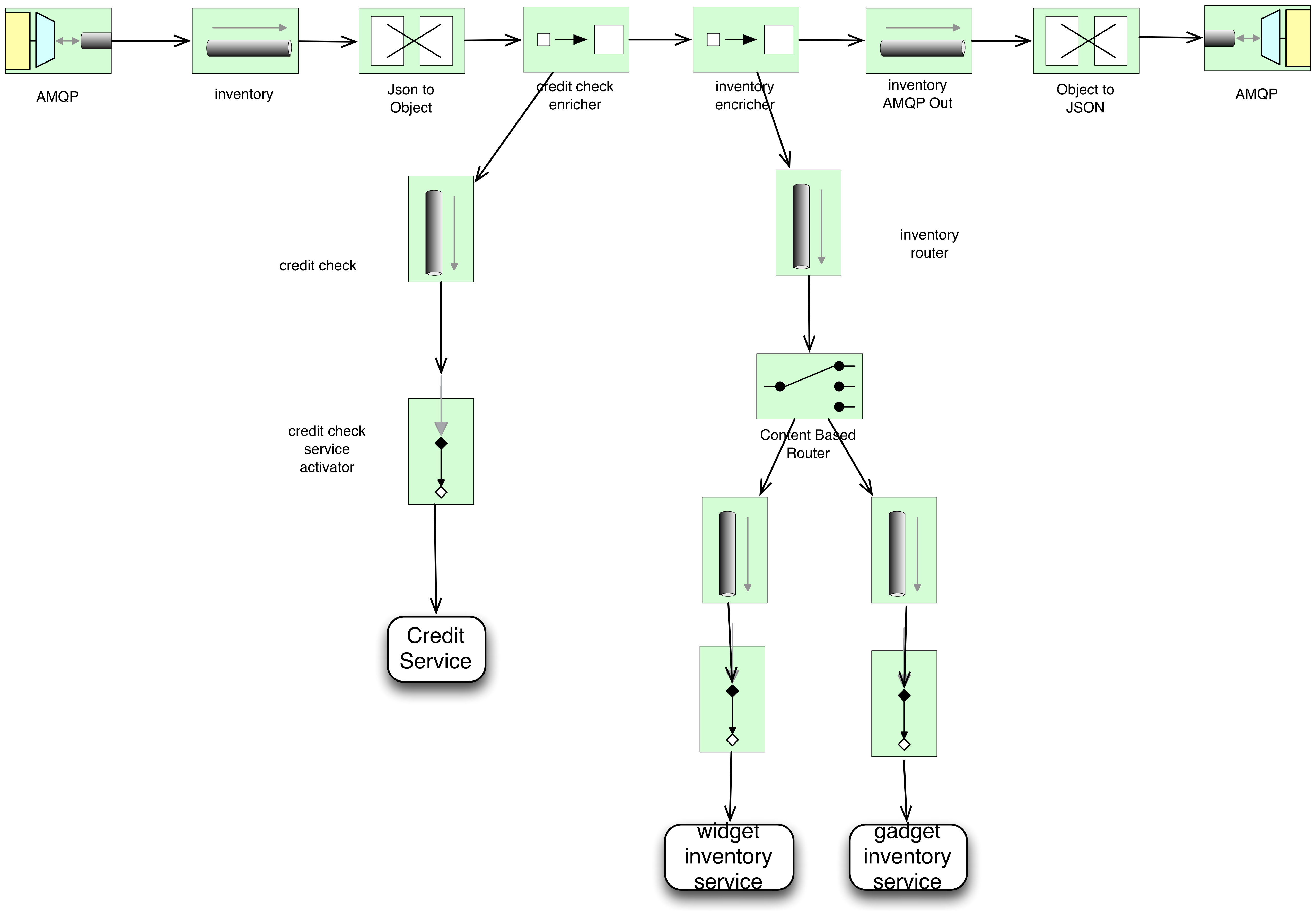
Using Spring Integration with the web store application


wgrus-store.war
StoreFront
wgrus-shipping.war
Shipping Service
wgrus-billing.war
Credit Service
wgrus-inventory.war
Spring Integration Logic
Widget InventoryService
Gadget InventoryService

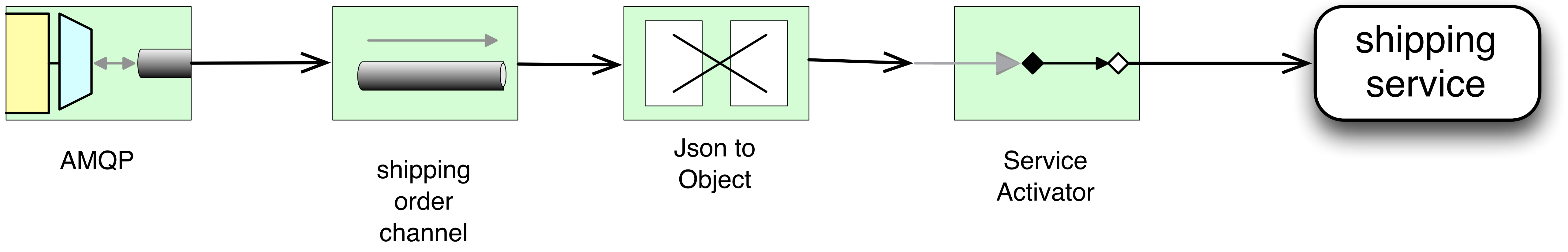
Store front flow



Inventory flow



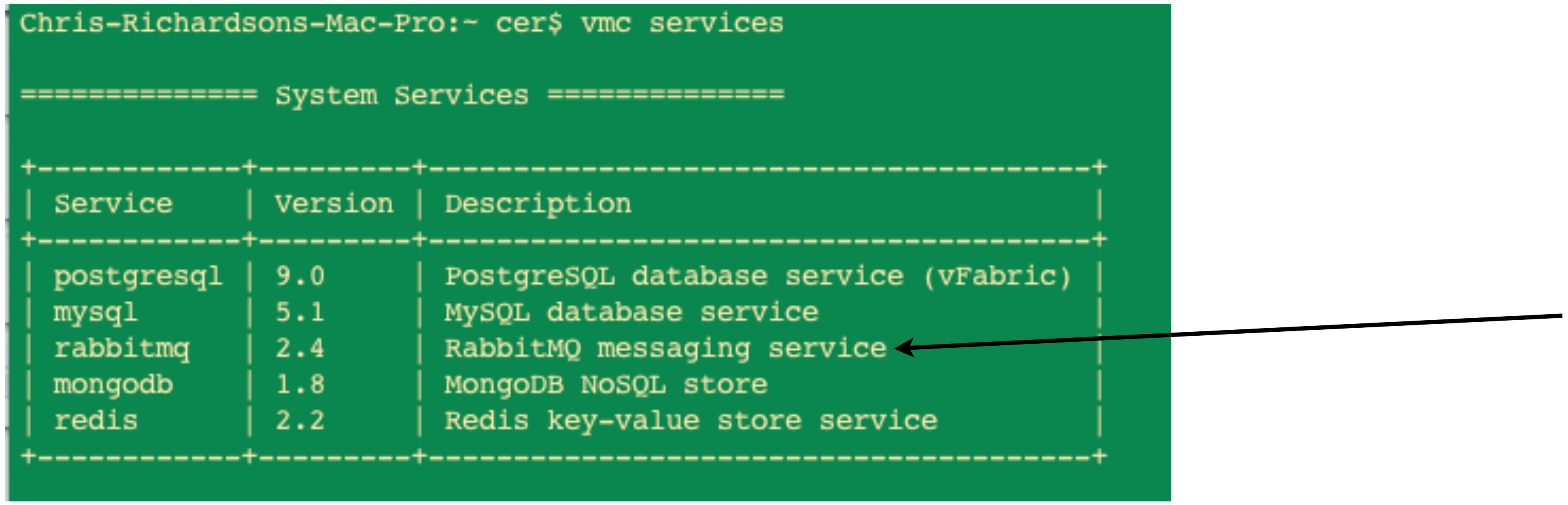
Shipping flow



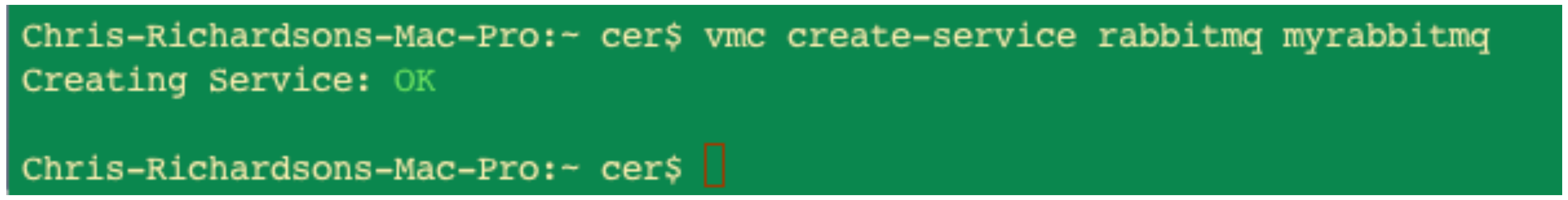
Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Building Java applications on Cloud Foundry
- Moving Spring applications to the Cloud
- Developing NoSQL applications for Cloud Foundry
- **Application integration with RabbitMQ and Spring AMQP**
  - Why messaging?
  - Messaging with RabbitMQ and AMQP
  - Using Spring Integration
  - **Cloud Foundry and RabbitMQ**
- Wrap up

Rabbit on Cloud Foundry



```
Chris-Richardsons-Mac-Pro:~ cer$ vmc services

============== System Services ==============

+------------+---------+---------------------------------------+
| Service    | Version | Description                           |
+------------+---------+---------------------------------------+
| postgresql | 9.0     | PostgreSQL database service (vFabric) |
| mysql      | 5.1     | MySQL database service                |
| rabbitmq   | 2.4     | RabbitMQ messaging service            |
| mongodb    | 1.8     | MongoDB NoSQL store                   |
| redis      | 2.2     | Redis key-value store service         |
+------------+---------+---------------------------------------+
```



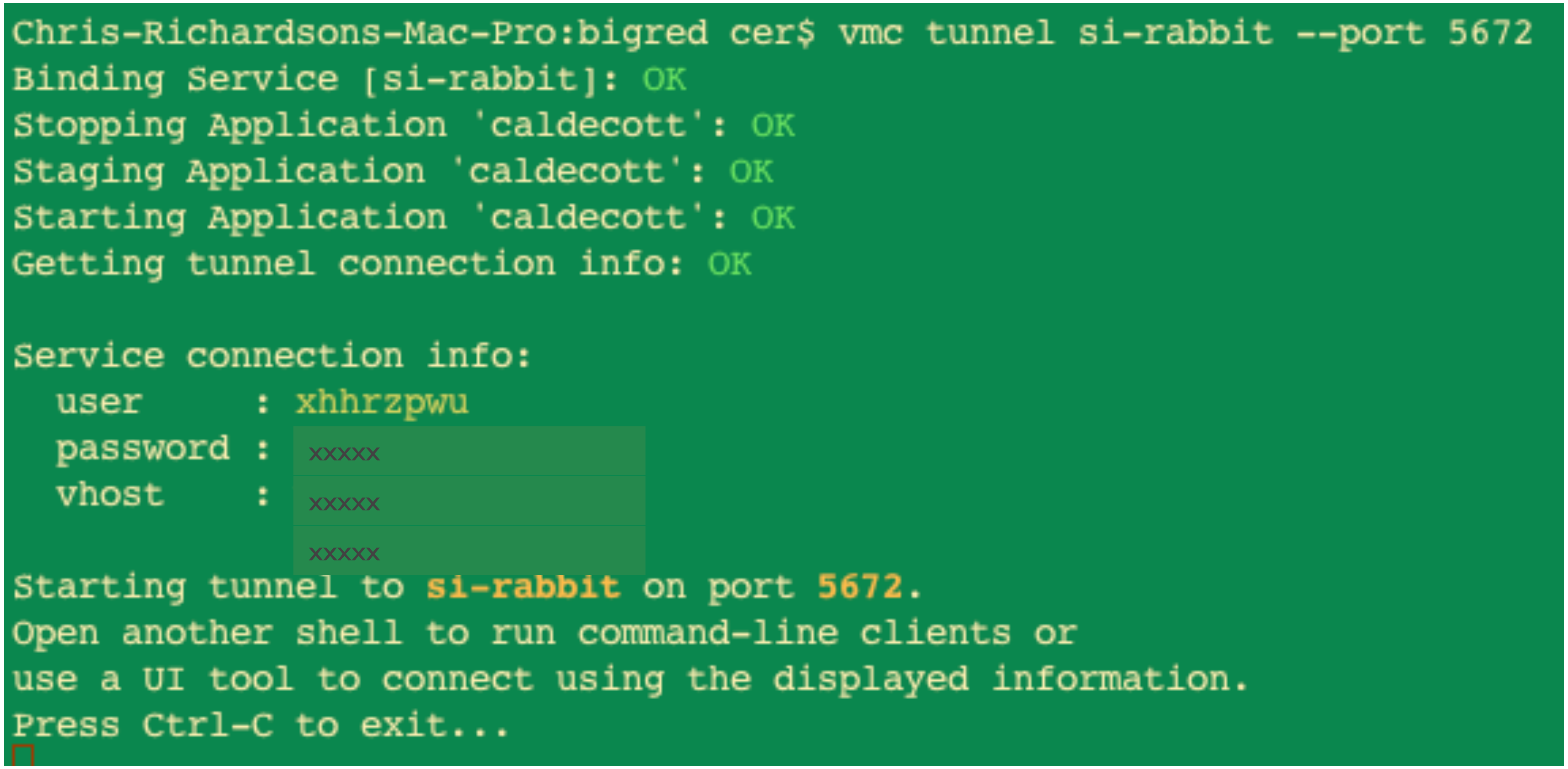
```
Chris-Richardsons-Mac-Pro:~ cer$ vmc create-service rabbitmq myrabbitmq
Creating Service: OK

Chris-Richardsons-Mac-Pro:~ cer$
```

Configuring a ConnectionFactory

```
<rabbit:template id="rabbitTemplate"
     connection-factory="rabbitConnectionFactory"/>

<beans profile="default">
  ...
  <rabbit:connection-factory id="rabbitConnectionFactory"/>
</beans>

<beans profile="cloud">
  ...
 <cloud:rabbit-connection-factory id="rabbitConnectionFactory"/>
</beans>
```

Using Caldecott with RabbitMQ

- Use for JUnit/Integration tests
- Run RabbitMQ tools



```
Chris-Richardsons-Mac-Pro:bigred cer$ vmc tunnel si-rabbit --port 5672
Binding Service [si-rabbit]: OK
Stopping Application 'caldecott': OK
Staging Application 'caldecott': OK
Starting Application 'caldecott': OK
Getting tunnel connection info: OK

Service connection info:
  user     : xhhrzpwu
  password : xxxxx
  vhost    : xxxxx
             xxxxx
Starting tunnel to si-rabbit on port 5672.
Open another shell to run command-line clients or
use a UI tool to connect using the displayed information.
Press Ctrl-C to exit...
```

Summary

- Modern applications need to have message-based architecture
- Spring Integration abstracts away the low-level aspects of messaging
- Cloud Foundry simplifies the development and deployment of RabbitMQ-based applications

Agenda

- Why Cloud? Why PaaS?
- Introducing Cloud Foundry
- Cloud Foundry for Spring developers
- Developing NoSQL applications for Cloud Foundry
- Application integration with RabbitMQ and Spring AMQP
- **Wrap Up**

Summary

- **Cloud? Good.**
- **Cloud Foundry? Good.**
- **Spring? Good.**
- **Cloud Foundry and Spring is a match made in heaven**

- **Home work:**
  - Learn Spring: **http://www.springframework.org**
  - Learn Spring Data **http://www.springframework.org/spring-data**
  - sign up for (free) Cloud Foundry at **http://www.cloudfoundry.com** or Download the **Cloud Foundry Micro Cloud**

By The Way

Promo Code:

JFokus

Stop by VMware booth

Questions?