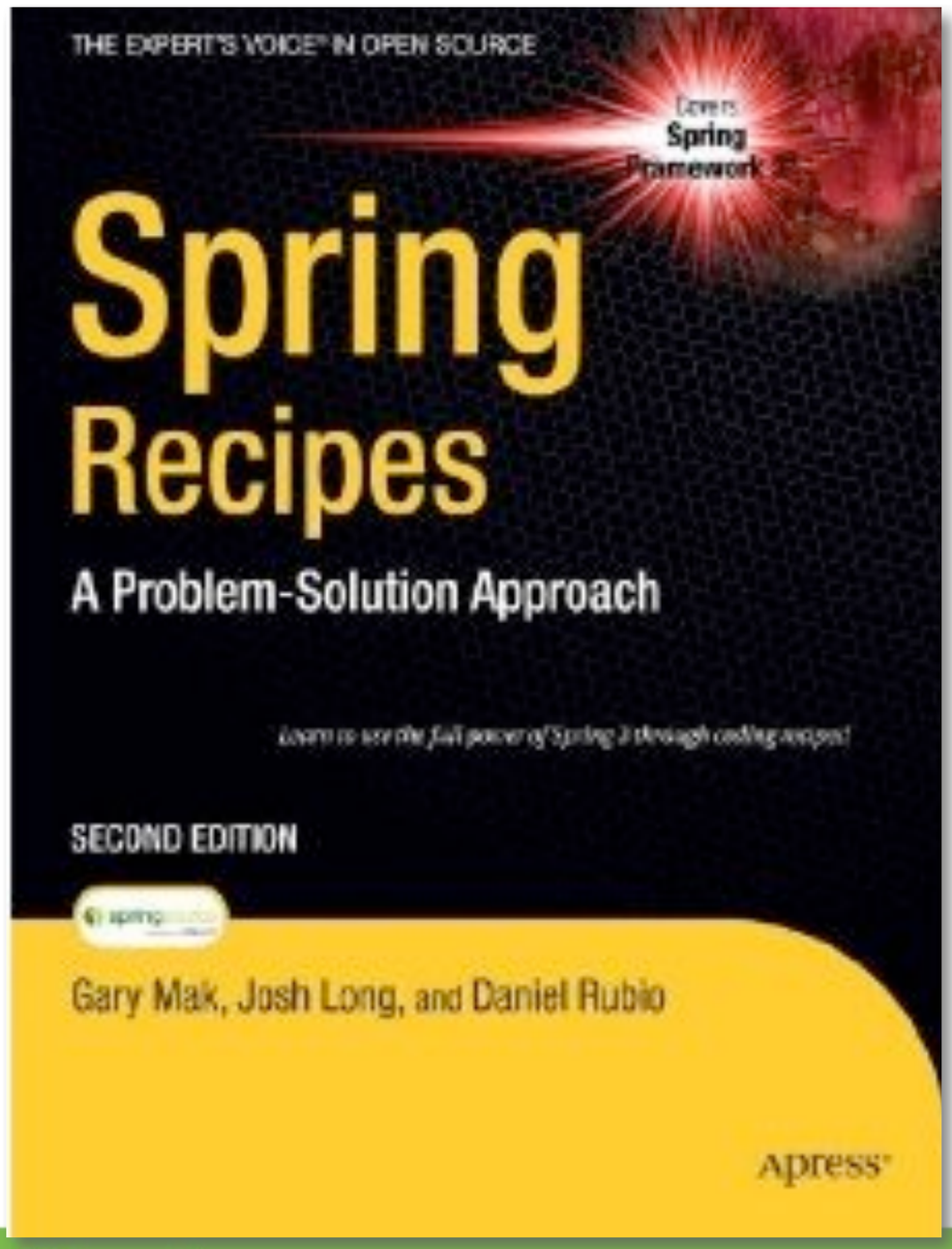
Spring and Cloud Foundry: a Marriage Made in Heaven

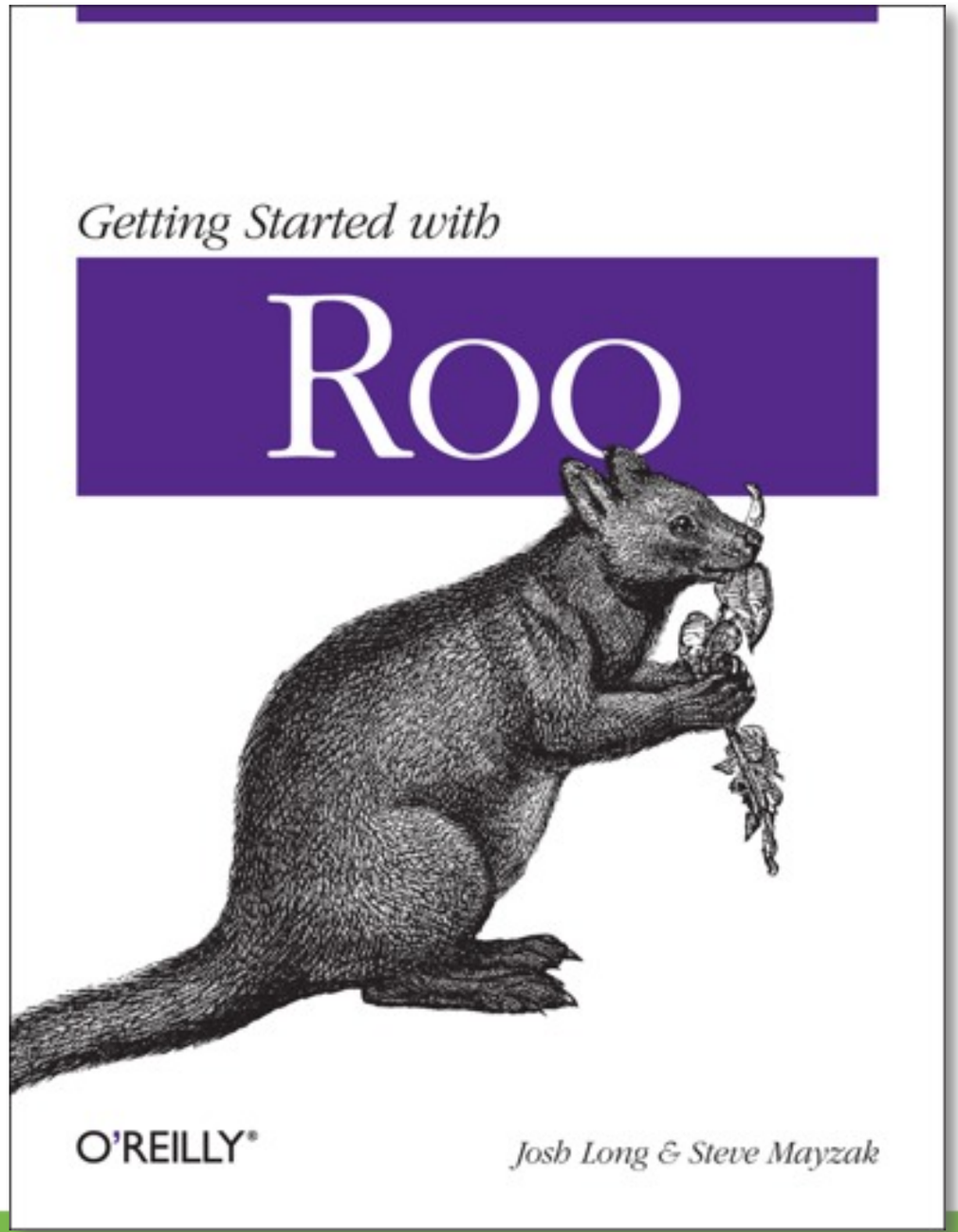
Josh Long, Spring Developer Advocate
SpringSource, a division of VMware

Twitter: @starbuxman
Email: josh.long@springsource.com

About Josh Long

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





Spring Makes Building Applications Easy...

Tell Spring About Your Objects

```
package the.package.with.beans.in.it;

@Service
public class CustomerService {

    public Customer createCustomer(String firstName,
                                    String lastName,
                                    Date signupDate)    {

    }

        ...

}
```

I want Database Access ... with Hibernate 4 Support

```
package the.package.with.beans.in.it;
...
@Service
public class CustomerService {

    @Inject
    private SessionFactory sessionFactory;

    public Customer createCustomer(String firstName,
                                   String lastName,
                                   Date signupDate)    {
        Customer customer = new Customer();
        customer.setFirstName(firstName);
        customer.setLastName(lastName);
        customer.setSignupDate(signupDate);

        sessionFactory.getCurrentSession().save(customer);
        return customer;
    }

...
```

I want Declarative Transaction Management...

```
package the.package.with.beans.in.it;
...
@Service
public class CustomerService {

    @Inject
    private SessionFactory sessionFactory;

    @Transactional
    public Customer createCustomer(String firstName,
                                   String lastName,
                                   Date signupDate)    {
        Customer customer = new Customer();
        customer.setFirstName(firstName);
        customer.setLastName(lastName);
        customer.setSignupDate(signupDate);

        sessionFactory.getCurrentSession().save(customer);
        return customer;
    }
    ...
}
```

I want Declarative Cache Management...

```
package the.package.with.beans.in.it;
...
@Service
public class CustomerService {

    @Inject
    private SessionFactory sessionFactory;

    @Transactional
    @Cacheable("customers")
    public Customer createCustomer(String firstName,
                                   String lastName,
                                   Date signupDate)    {
        Customer customer = new Customer();
        customer.setFirstName(firstName);
        customer.setLastName(lastName);
        customer.setSignupDate(signupDate);

        sessionFactory.getCurrentSession().save(customer);
        return customer;
    }
}
```

I want a RESTful Endpoint...

```
package org.springsource.examples.spring31.web;
..

@Controller
public class CustomerController {

   @Inject
   private CustomerService customerService;

   @RequestMapping(value = "/customer/{id}",
                           produces = MediaType.APPLICATION_JSON_VALUE)
   public @ResponseBody Customer customerById(@PathVariable("id") Integer id) {
      return customerService.getCustomerById(id);
   }
    ...
}
```

Spring's aim:

bring simplicity to Java development

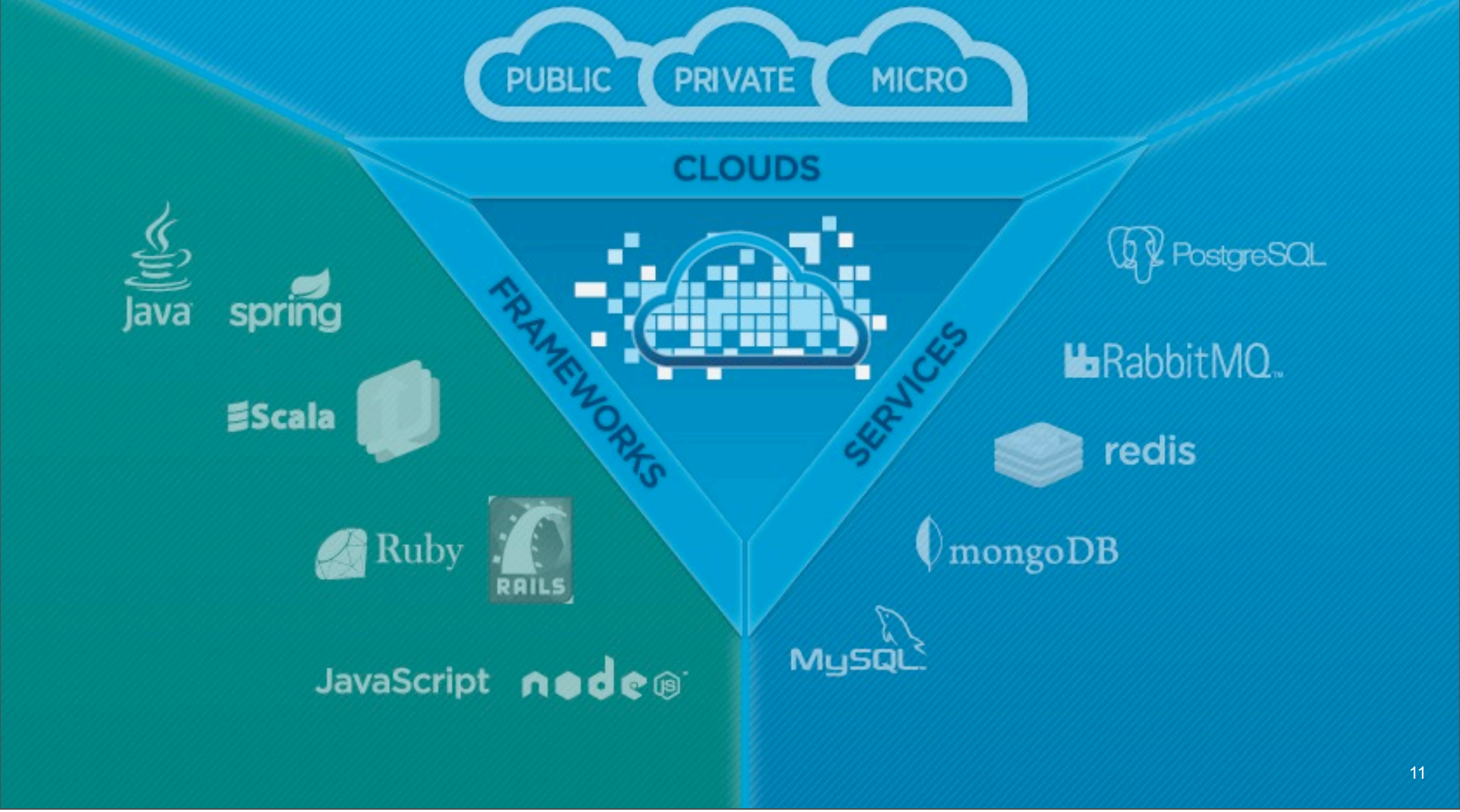
web tier & RIA	service tier	batch processing	integration & messaging	data access / NoSQL / Big Data	mobile

The Spring framework

the cloud:	lightweight	traditional
CloudFoundry Google App Engine Amazon Web Services	tc Server Tomcat Jetty	WebSphere JBoss AS WebLogic (on legacy versions, too!)

Now we just need a platform...

Cloud Foundry: Choice of Runtimes



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**

- **Ecosystem Partners**
  - **.NET (Uhuru, Tier3)**
    - **both from Seattle-native partners!**
    - **Runs standard .NET (no need to port your application)**
  - **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

Manifests

```
---
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
```

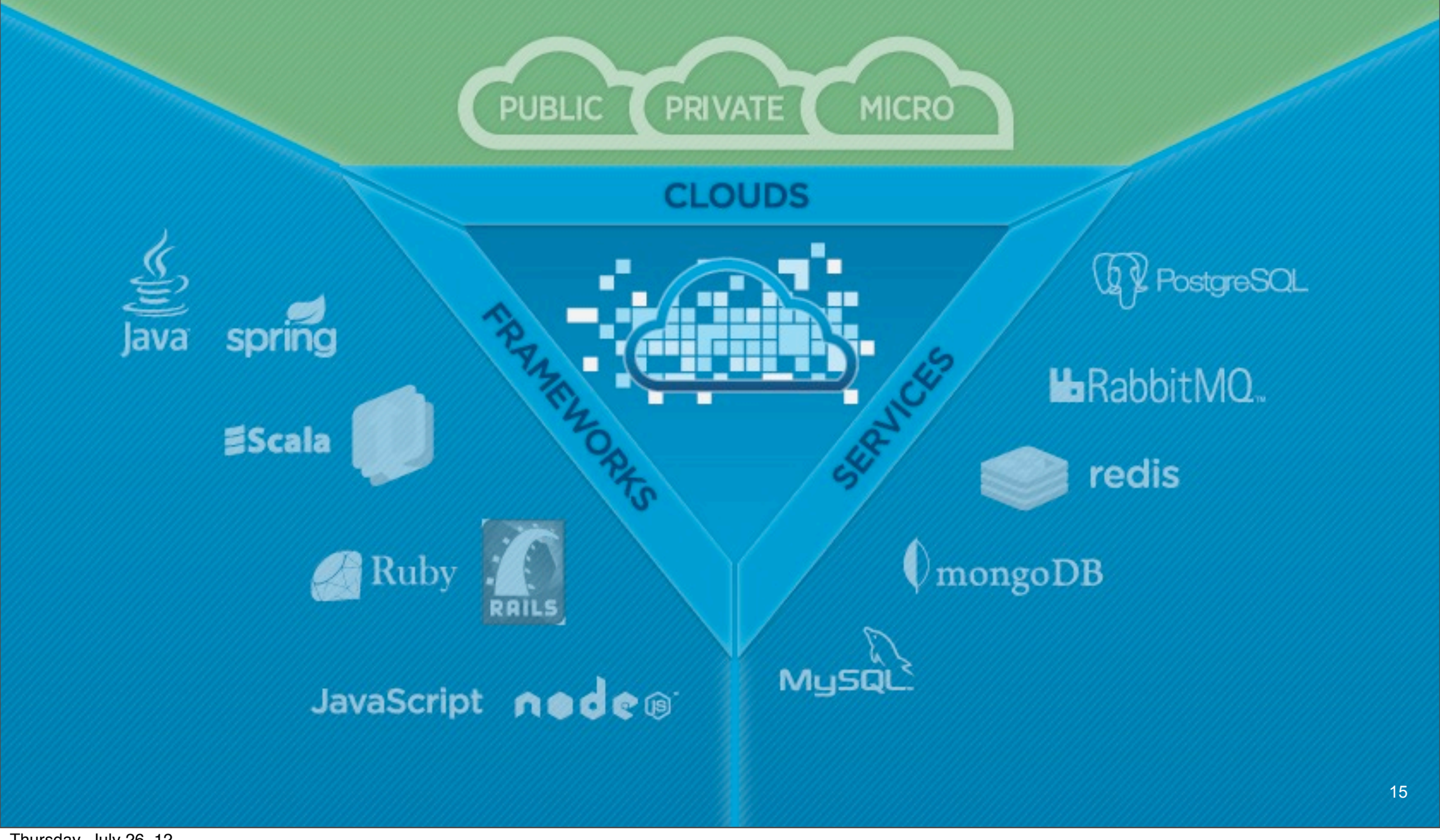
Demo: **Deploying an Application**

...from vmc

with *and* without **manifest.yml**

...from STS

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

- added April 8, 2008
- 3198 people starred it
- eventually closed in January, 2011
- never added

Issue 13 - googleappengine

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

googleappengine
Google App Engine

Project Home | Downloads | Wiki | **Issues** | Source

New issue | Search | Open issues | for

Issue 13: PHP support is a must
3198 people starred this issue.
Comments by non-members will not trigger notification emails to user

Status: Acknowledged
Owner: ----
Type-Feature
Priority-Medium
Component-Languages
Restrict-AddIssu...t-Commit

Reported by mirnazim, Apr 8, 2008

```
PHP is one of the most popular language for web. SO I think PHP support is
very important, I guess
```

Showing comments 420 - 919 of 919 Older ›

Comment 1163 by jordan.d...@gmail.com, Jul 23, 2009

```
PHP = More Developers, More projects
Python = Less Devs, less projects
```

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

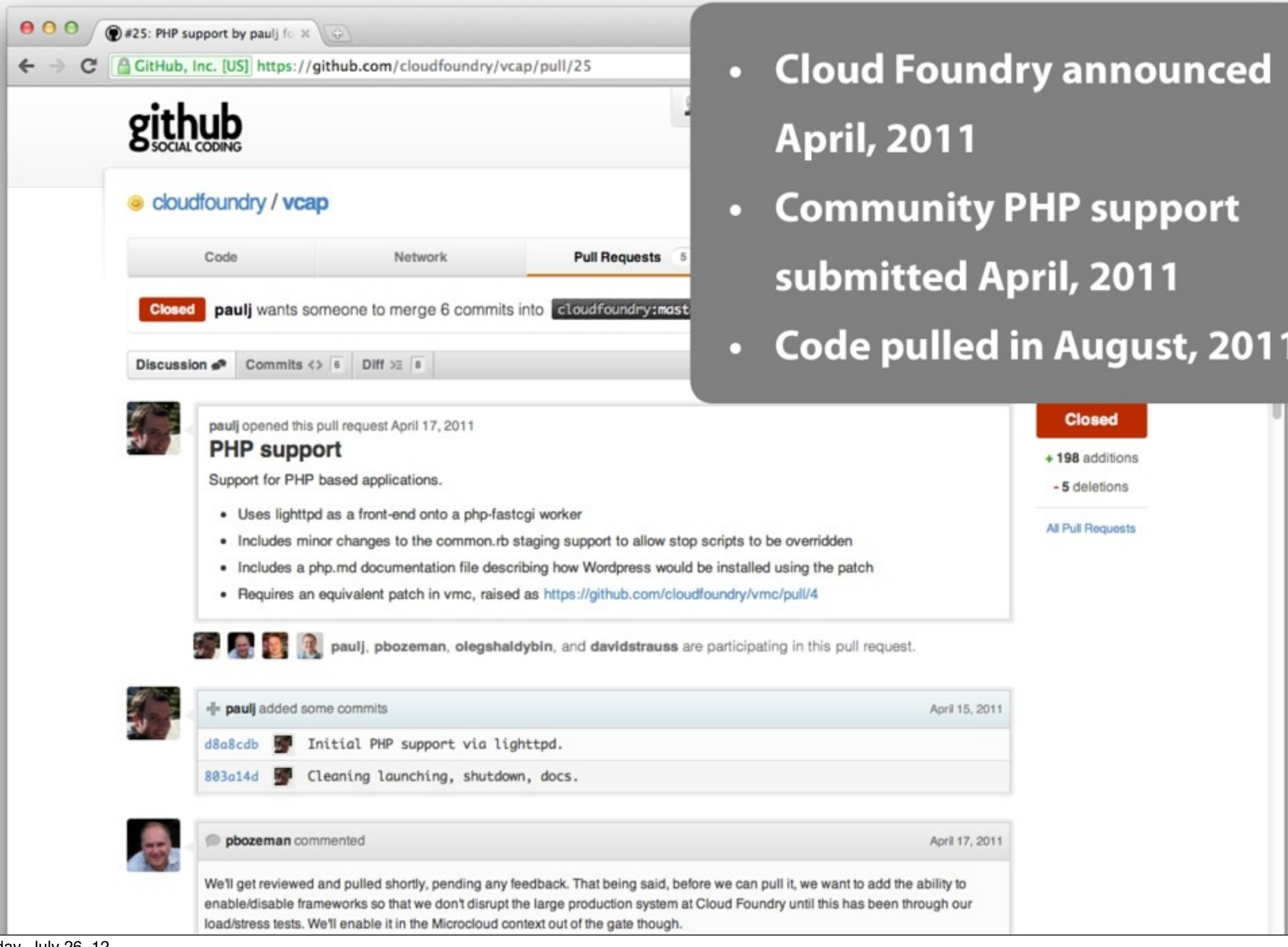
```
I'm making this issue read-only. I think the points here have been made. There's no reason to email thousands of
people every time someone says "+1".

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.
```

```
please support php.
```

Comment 1167 by tapsboy, Jul 28, 2009

Open Source Advantage

- Cloud Foundry announced April, 2011
- Community PHP support submitted April, 2011
- Code pulled in August, 2011



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 Community

cloudfoundry (The open p

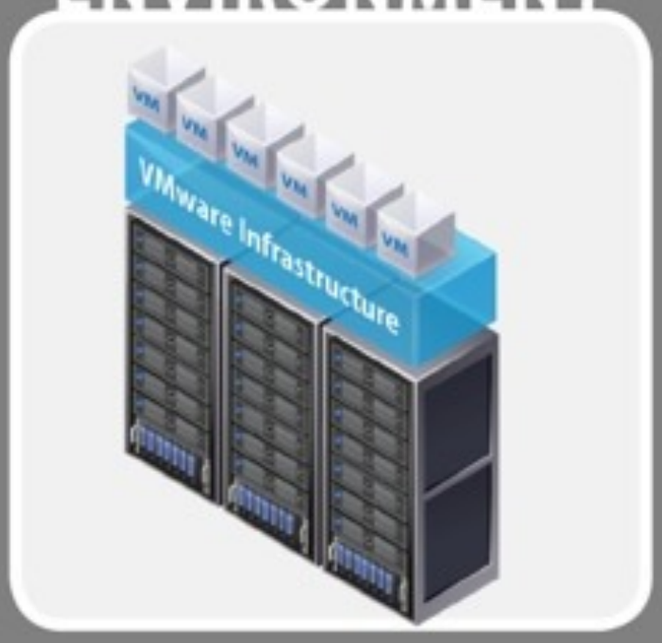
Name	The open platform-as-a-service project
Email	support@cloudfoundry.org
Website/Blog	http://www.cloudfoundry.org
Member Since	Feb 16, 2011

Public Repositories (7)

vCENTER / vSPHERE

GITHUB

ENVIRONMENT



SCRIPTS

DEPLOY

vcap-services Ruby 250

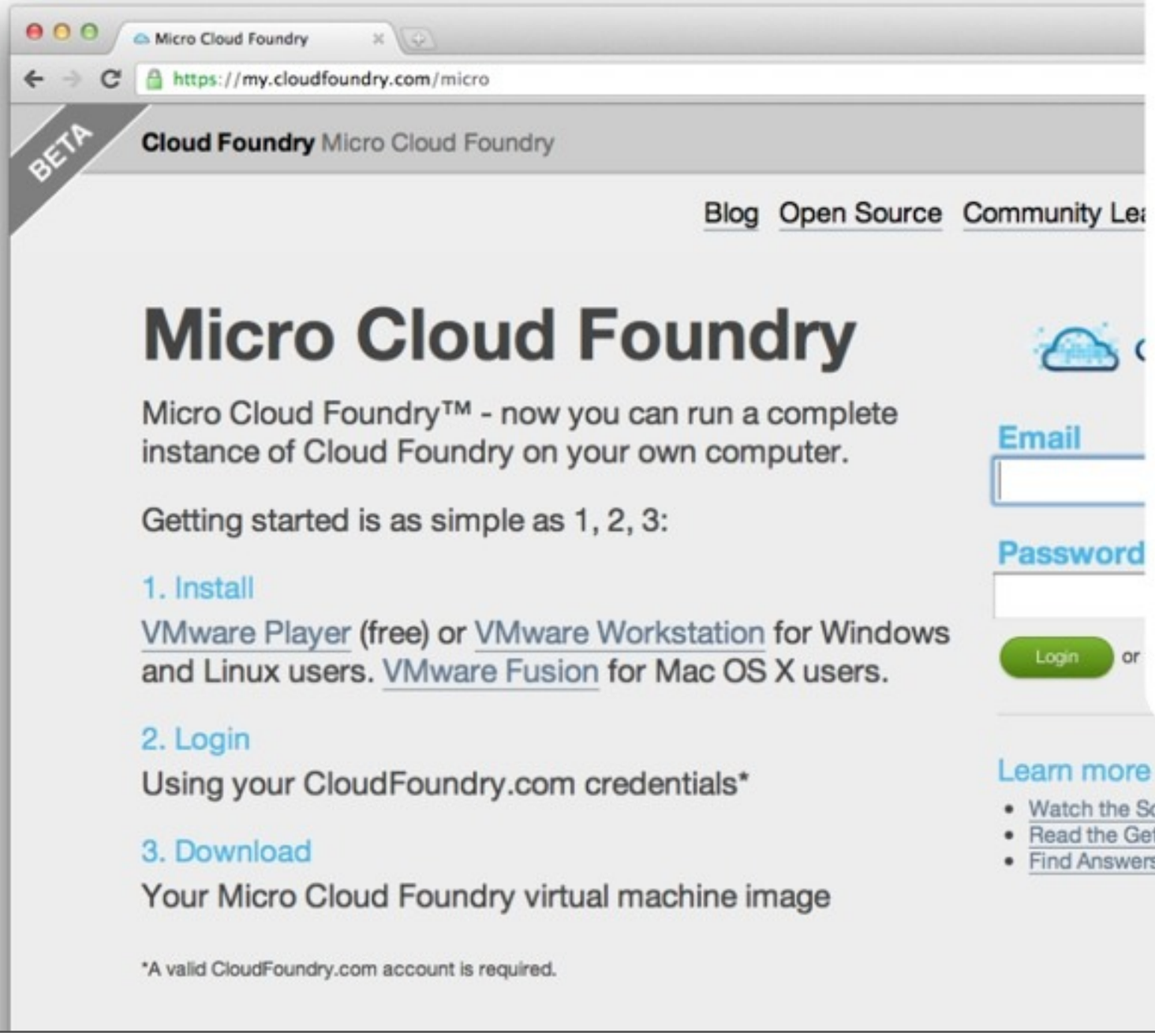
Cloud Foundry - the open platform as a service project

Last updated about 9 hours ago

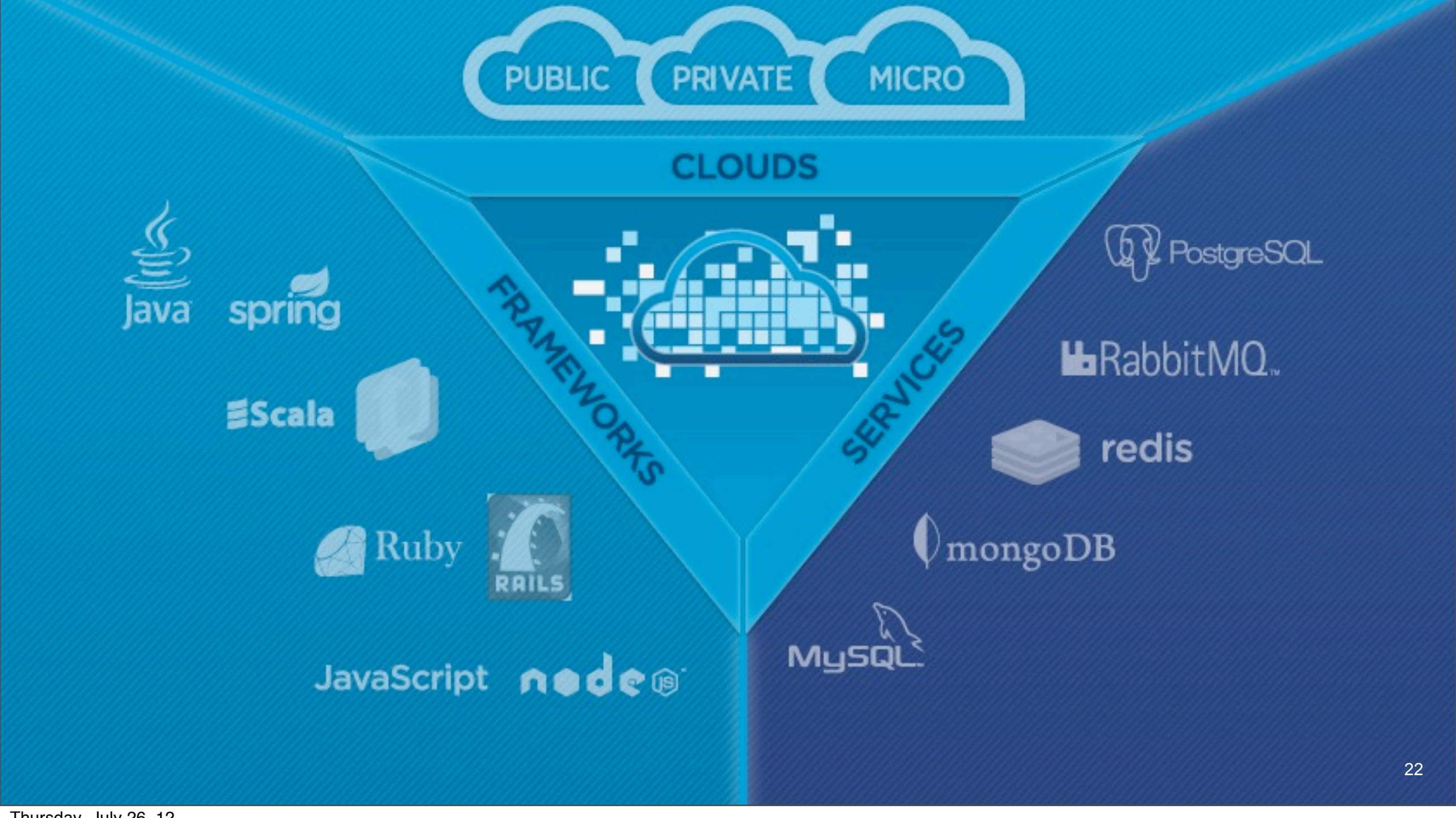
all commits commits by owner 52 week partic

Micro Cloud Foundry

- micro.cloudfoundry.com
- works on OS X, Windows, Linux



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

- **Take Advantage of Services**
  - they cost *nothing* to setup
  - they deliver value

- **They Encourage Better Architectures**
  - Need a fast read-write cache? **Redis** is ready to go!
  - Need to store long-tail documents? Give **MongoDB** a try
  - Need to decouple what applications do from when they do it? Use messaging and **RabbitMQ**

Spring's the Best Toolkit for Your Services

- **Spring Data**
  - supports advanced JPA, MongoDB, Redis connectivity
- **Spring AMQP, Spring Integration**
  - Supports messaging, and event-driven architectures
- **Spring core**
  - Has best-of-breed support for RDBMS access, be it through JPA, JDBC, JDO, Hibernate, etc.

Let’s Talk About Spring, Baby...

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
 ....
<beans>

 <tx:annotation-driven transaction-manager = "txManager"  />

 <context:component-scan base-package = "org.springsource.examples.spring31.services" />

 <context:property-placeholder properties = "config.properties" />

 <bean id = "txManager"
        class = "org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name = "sessionFactory" ref = "mySessionFactory" />
 </bean>

  <bean id = "sessionFactory"
        class = "org.springframework.orm.hibernate4.LocalSessionFactoryBean">
   ...
  </bean>

</beans>
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
 ....
<beans>

  <tx:annotation-driven transaction-manager = "txManager"  />

  <context:component-scan base-package = "org.springsource.examples.spring31.services" />

  <context:property-placeholder properties = "config.properties" />

  <bean id = "txManager"
          class = "org.springframework.orm.hibernate3.HibernateTransactionManager">
      <property name = "sessionFactory" ref = "mySessionFactory" />
  </bean>

   <bean id = "sessionFactory"
           class = "org.springframework.orm.hibernate4.LocalSessionFactoryBean">
    ...
   </bean>

</beans>
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
@Configuration
@PropertySource("/config.properties")
@EnableTransactionManagement
@ComponentScan(basePackageClasses = {CustomerService.class})
public class ServicesConfiguration {

    @Bean
    public PlatformTransactionManager txManager() throws Exception {
        return new HibernateTransactionManager(this.sessionFactory());
    }

    @Bean
    public SessionFactory sessionFactory()  { ... }

}
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
 ....
<beans>

 <tx:annotation-driven transaction-manager = "txManager"  />

 <context:component-scan base-package = "org.springsource.examples.spring31.services" />

 <context:property-placeholder properties = "config.properties" />

 <bean id = "txManager"
        class = "org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name = "sessionFactory" ref = "mySessionFactory" />
 </bean>

  <bean id = "sessionFactory"
        class = "org.springframework.orm.hibernate4.LocalSessionFactoryBean">
   ...
  </bean>

</beans>
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
@Configuration
@PropertySource("/config.properties")
@EnableTransactionManagement
@ComponentScan(basePackageClasses = {CustomerService.class})
public class ServicesConfiguration {

    @Bean
    public PlatformTransactionManager txManager() throws Exception {
        return new HibernateTransactionManager(this.sessionFactory());
    }

    @Bean
    public SessionFactory sessionFactory()  { ... }

}
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
 ....
<beans>

 <tx:annotation-driven transaction-manager = "txManager"  />

 <context:component-scan base-package = "org.springsource.examples.spring31.services" />

 <context:property-placeholder properties = "config.properties" />

 <bean id = "txManager"
        class = "org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name = "sessionFactory" ref = "mySessionFactory" />
 </bean>

  <bean id = "sessionFactory"
        class = "org.springframework.orm.hibernate4.LocalSessionFactoryBean">
   ...
  </bean>

</beans>
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
@Configuration
@PropertySource("/config.properties")
@EnableTransactionManagement
@ComponentScan(basePackageClasses = {CustomerService.class})
public class ServicesConfiguration {

    @Bean
    public PlatformTransactionManager txManager() throws Exception {
        return new HibernateTransactionManager(this.sessionFactory());
    }

    @Bean
    public SessionFactory sessionFactory()  { ... }

}
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
 ....
<beans>

 <tx:annotation-driven transaction-manager = "txManager"  />

 <context:component-scan base-package = "org.springsource.examples.spring31.services" />

 <context:property-placeholder properties = "config.properties" />

 <bean id = "txManager"
        class = "org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name = "sessionFactory" ref = "mySessionFactory" />
 </bean>

 <bean id = "sessionFactory"
        class = "org.springframework.orm.hibernate4.LocalSessionFactoryBean">
  ...
 </bean>

</beans>
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
@Configuration
@PropertySource("/config.properties")
@EnableTransactionManagement
@ComponentScan(basePackageClasses = {CustomerService.class})
public class ServicesConfiguration {

    @Bean
    public PlatformTransactionManager txManager() throws Exception {
        return new HibernateTransactionManager(this.sessionFactory());
    }

    @Bean
    public SessionFactory sessionFactory()  { ... }

}
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
 ....
<beans>

 <tx:annotation-driven transaction-manager = "txManager"  />

 <context:component-scan base-package = "org.springsource.examples.spring31.services" />

 <context:property-placeholder properties = "config.properties" />

 <bean id = "txManager"
        class = "org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name = "sessionFactory" ref = "mySessionFactory" />
 </bean>

  <bean id = "sessionFactory"
        class = "org.springframework.orm.hibernate4.LocalSessionFactoryBean">
   ...
  </bean>

</beans>
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
@Configuration
@PropertySource("/config.properties")
@EnableTransactionManagement
@ComponentScan(basePackageClasses = {CustomerService.class})
public class ServicesConfiguration {

    @Bean
    public PlatformTransactionManager txManager() throws Exception {
        return new HibernateTransactionManager(this.sessionFactory());
    }

    @Bean
    public SessionFactory sessionFactory()  { ... }

}
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
 ....
<beans>

 <tx:annotation-driven transaction-manager = "txManager"  />

 <context:component-scan base-package = "org.springsource.examples.spring31.services" />

 <context:property-placeholder properties = "config.properties" />

 <bean id = "txManager"
        class = "org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name = "sessionFactory" ref = "mySessionFactory" />
 </bean>

  <bean id = "sessionFactory"
        class = "org.springframework.orm.hibernate4.LocalSessionFactoryBean">
   ...
  </bean>

</beans>
```

Tangent: Quick Primer on Java Configuration in Spring 3.1

```
@Configuration
@PropertySource("/config.properties")
@EnableTransactionManagement
@ComponentScan(basePackageClasses = {CustomerService.class})
public class ServicesConfiguration {

    @Bean
    public PlatformTransactionManager txManager() throws Exception {
        return new HibernateTransactionManager(this.sessionFactory());
    }

    @Bean
    public SessionFactory sessionFactory()  { ... }

}
```

The 80% Case

Auto-Reconfiguration

- **Solves the 80% Case**
- **Application works on Tomcat, connects to an RDBMS, uses Spring**
- **Cloud Foundry Supports This out of the Box**
  - Auto reconfiguration
- **Supported by many technologies (Spring, Ruby, Grails, Lift, etc.)**

Auto-Reconfiguration

- **Detects common Java beans:**
  - DataSource
  - RedisConnectionFactory
  - RabbitConnectionFactory
  - Mongo

- **hibernate.dialect property becomes MySQLDialect, 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)
- **Cloud Foundry installs a BeanFactoryPostProcessor**

The cloud namespace

the cloud Namespace

- **<cloud:> namespace for use in Spring app contexts**
- **Provides application-level control of bean service bindings**
- **Use when you have multiple...**
  - services of the same type
  - beans of the same type, e.g.: **DataSource**, **MongoDBFactory**
- **Use when you have custom bean configuration**
  - data source 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>

- **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>

```
<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;
}
```

Using the CloudEnvironment API

The Spring Developer's Perspective: The Environment

$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":.....

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
```

The Spring Developer's Perspective: The Environment

```
<dependency>
    <groupId>org.cloudfoundry</groupId>
     <artifactId>cloudfoundry-runtime</artifactId>
     <version>0.8.1</version>
</dependency>
```

The Spring Developer's Perspective: The Environment

```
CloudEnvironment e = new CloudEnvironment();

Iterable<RdbmsServiceInfo> dsServiceInformation =
    e.getServiceInfos( RdbmsServiceInfo.class) ;

Iterable<RedisServiceInfo> redisServiceInformation =
    e.getServiceInfos( RedisServiceInfo.class) ;

Iterable<RabbitServiceInfo> rabbitServiceInformation =
    e.getServiceInfos( RabbitServiceInfo.class) ;
```

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());
```

Demo: **Scaling with `vmc instances` and `CloudEnvironment`**

One Binary to Rule Them All

Spring 3.1 Profiles

- In Development

```
<ds:embedded-database id= "dataSource" type = “H2” />
```

- In Production

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

Spring 3.1 Profiles

```
...
<beans ....>

 <beans profile = “dev”>
  <ds:embedded-database id= "dataSource" type = “H2” />
 </beans>

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

</beans>
```

Spring 3.1 Profiles

- **“cloud” profile automatically activates on Cloud Foundry**
  - usage of the cloud namespace should occur within the cloud profile block

Demo: **Look at a Multi-Env Application with H2 and PostgreSQL**

A Running Tour of Some Useful Spring's APIs

Using MongoDB from Cloud Foundry

```
<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>
```

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

Using MongoDB from Cloud Foundry

```
<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>
```

for a sophisticated example, check out
http://www.github.com/SpringSource/cloudfoundry-samples/cross-store

Demo: **Cross Store Persistence with MongoDB on Cloud Foundry**


the ORIGINAL “Cloud Scale Messaging”

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

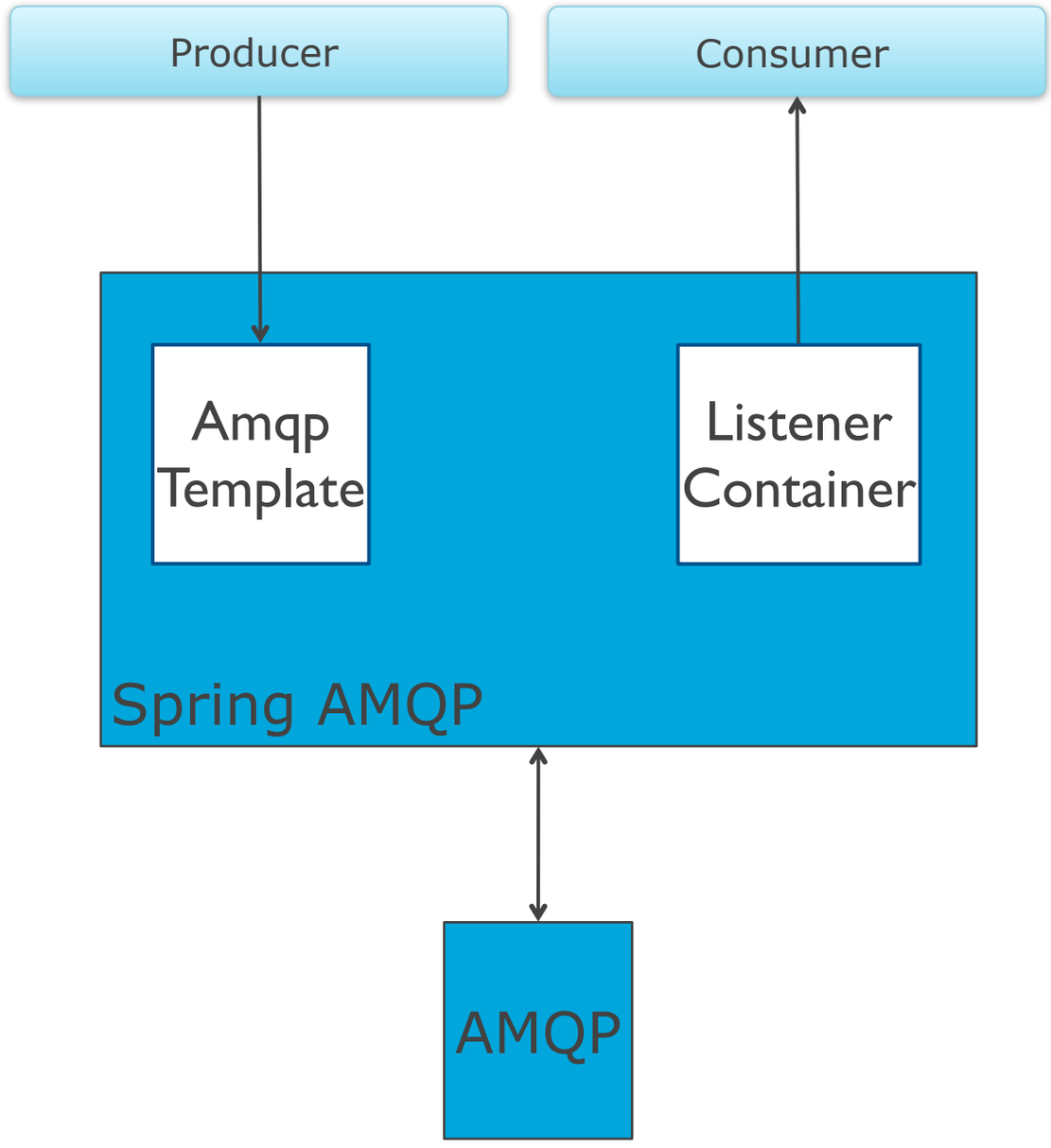
On commodity hardware

- 10-25 thousand messages per second is routine *
- The NIC is usually the bottleneck

* Non-persistent messages

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>
```

Using Redis from Cloud Foundry as a CacheManager

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

```
@Bean
public CacheManager cacheManager() throws Exception {
    return new RedisCacheManager(redisTemplate());
}
```

```
<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>
```

Demo: **Implementing Caching in the Cloud**
add data, remove some data, request data

$> vmc tunnel redis
keys *

CloudFoundry.com signup promo code: **cloudtalk**

Questions?

Say hi on Twitter:

@starbuxman @springsource @cloudfoundry