2013大数据全球技术峰会
Big Data Global Summit 2013

Cloud Foundry的弹性设计

喻　勇（Frank Yu）
yuf@vmware.com

@cloudfoundry
http://www.cloudfoundry.cn

VMware上海研发中心

内容提要

- Cloud Foundry产品概述
- Cloud Foundry架构剖析
- Cloud Foundry的NATS模块
- Warden Container
- 数据及服务的整合
- Q&A

Cloud Foundry产品概述

云计算的三个层次

SaaS
Software as a Service

PaaS
Platform as a Service

IaaS
Infrastructure as a Service

IaaS: 硬件的自动化管理，人与机器的解耦合　　获得效率/牺牲性能

PaaS: 应用的自动化管理，应用与OS的解耦合　　获得弹性/牺牲控制

Cloud Foundry的元素

OSS
community

vFabric
Postgres

vFabric
RabbitMQ™

Application Service Interface

Cloud Provider Interface

CLOUD
FOUNDRY™

Data
Services

Msg Services

Other
Services

Private
Clouds

Public
Clouds

Micro
Clouds

Additional partners
services …

Apache2 License

演　示

vmc 工具

```
Create app, update app, control app
vmc push [appname] [--path] [--url] [--instances N] [--mem] [--no-start]
vmc update <appname> [--path PATH]
vmc stop <appname>
vmc start <appname>
vmc target [url]

Update app settings, get app information
vmc mem <appname> [memsize]
vmc map <appname> <url>
vmc instances <appname> <num | delta>
vmc {crashes, crashlogs, logs} <appname>
vmc files <appname> [path]

Deal with services, users, and information
vmc create-service <service> [--name servicename] [--bind appname]
vmc bind-service <servicename> <appname>
vmc unbind-service  <servicename> <appname>
vmc delete-service <servicename>

vmc user, vmc passwd, vmc login, vmc logout, vmc add-user
vmc services, vmc apps, vmc info
```

应用平台支持: 不断增多

- **Java平台**
  - Grails模仿Rails的Java平台实现
  - Java_web 普通Java web程序
  - Lift基于Scale的web框架
  - Spring 流行的Java框架
  - Play
- **Ruby平台**
  - Rack 最小化的Ruby Web框架
  - Rails3 一站式的Ruby Web框架
  - Sinatra 极简主义的Ruby Web框架
- **Python平台**
  - Django 最流行的PythonWeb框架
  - Wsgi Python的CGI
- **其他平台**
  - node.js 异步Web框架
  - Erlang
  - Php
  - Microsoft .net
  - Standalone 独立的程序

服务平台支持：不断增多

- **主流服务**
  - Mongodb
    最流行的Nosql数据库
  - MySQL
    传统开源关系数据库
  - Postgresql
    MySQL的有力竞争者
  - Redis
    极快的内存KV数据库
  - Neo4j
    图数据库

More on github

- **存储**
  - Blob
    - Amazon S3
    - Atmos
    - NFS
    - Local Disk
  - FileSystem 远程NFS支持

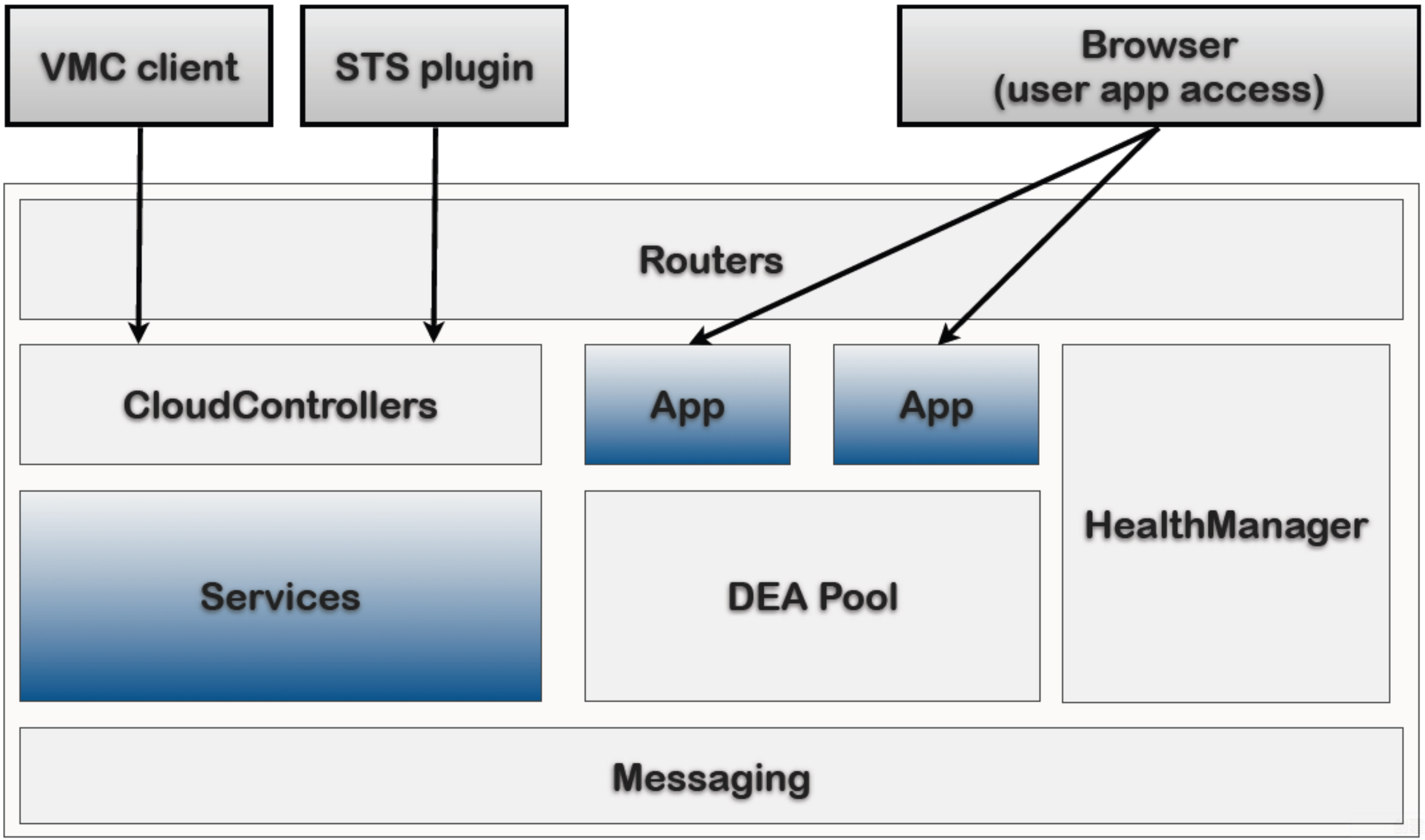
- **消息**
  - RabbitMQ 出色的Erlang队列系统

- **大数据**
  - Hadoop集成
  - Project Serengeti

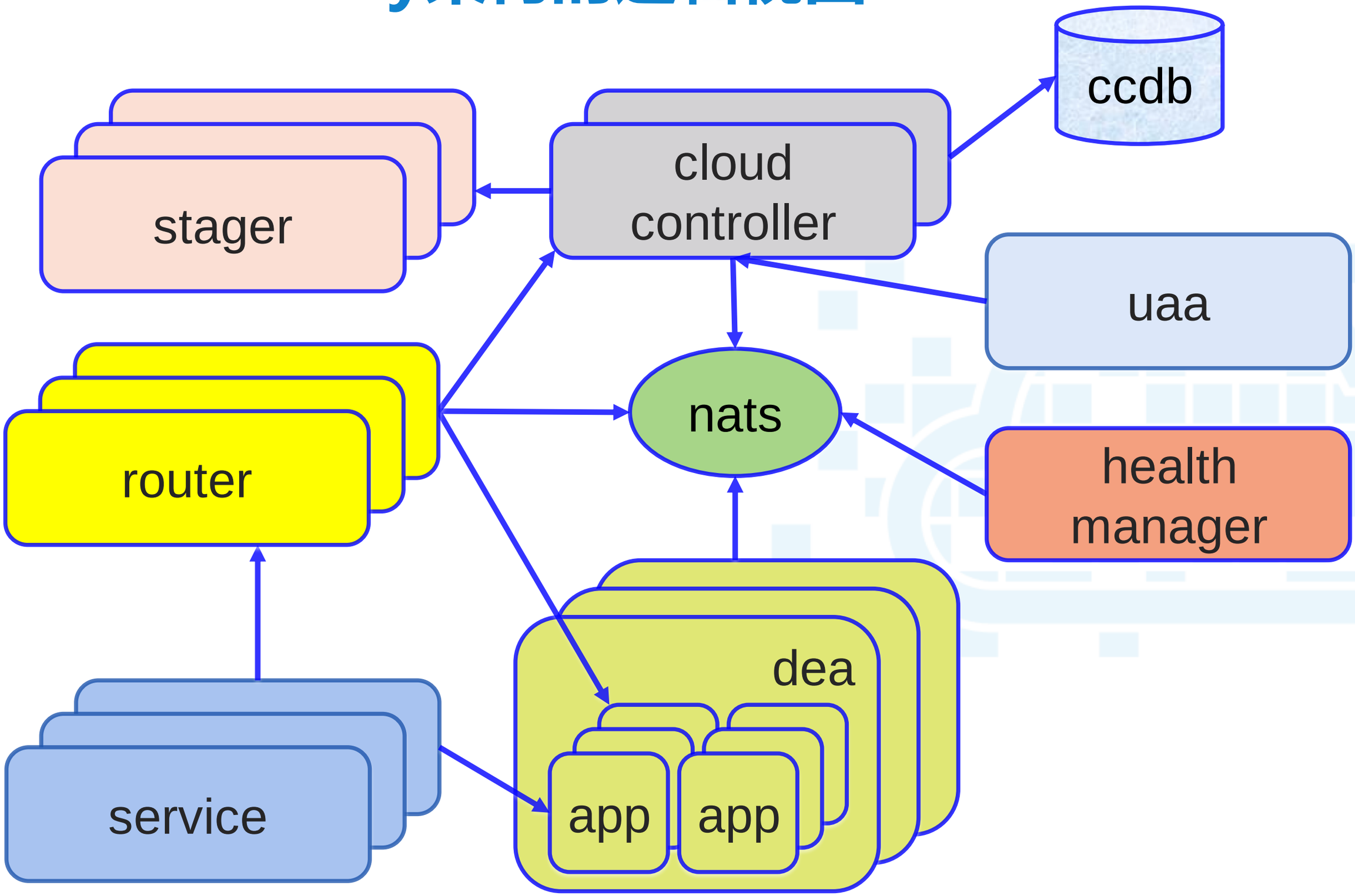
http://serengeti.cloudfoundry.com

Cloud Foundry架构剖析

Cloud Foundry架构的逻辑视图 - 1



Cloud Foundry架构的逻辑视图 - 2

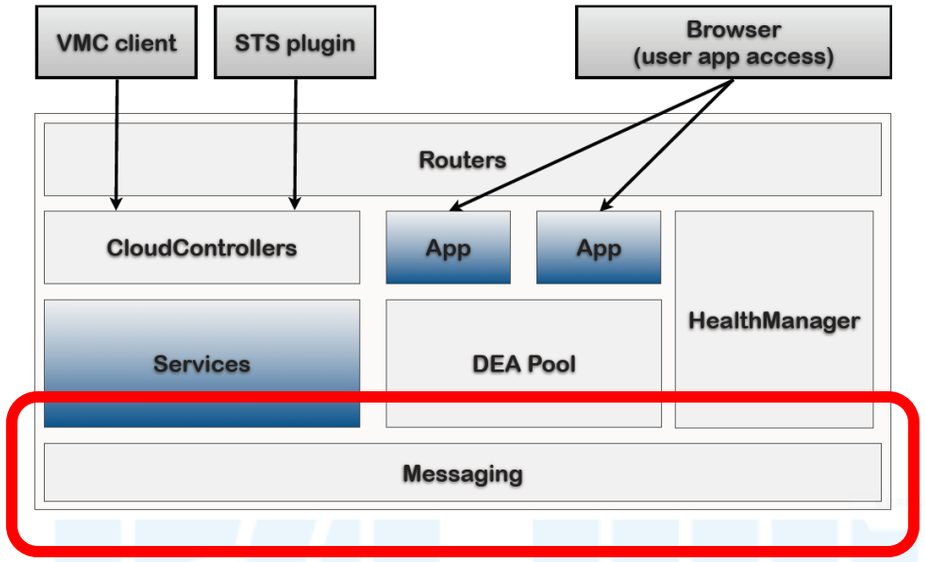


Cloud Foundry的设计理念

- **前提**
  - 假设失败
  - 为MTTR优化，而不是MTBF
  - 快速失效，自我修复
  - 横向扩展的组件
  - 分布式状态，没有单点故障
  - 极度简单
- **模式**
  - 事件(消息)驱动
  - 异步
  - 非堵塞
  - 独立
  - 消息传递
  - 最终一致性
- **设计**
  - 组件的松耦合
    - 更少的类，更多的实例
  - 消息是基础
    - 寻址和组件发现
    - 命令和控制
  - JSON
    - 数据通过HTTP或File/Blob传递
- **内核组件的特点**
  - 动态发现
  - 无依赖性和启动顺序
  - 通过HTTP/JSON监控
  - 位置独立性

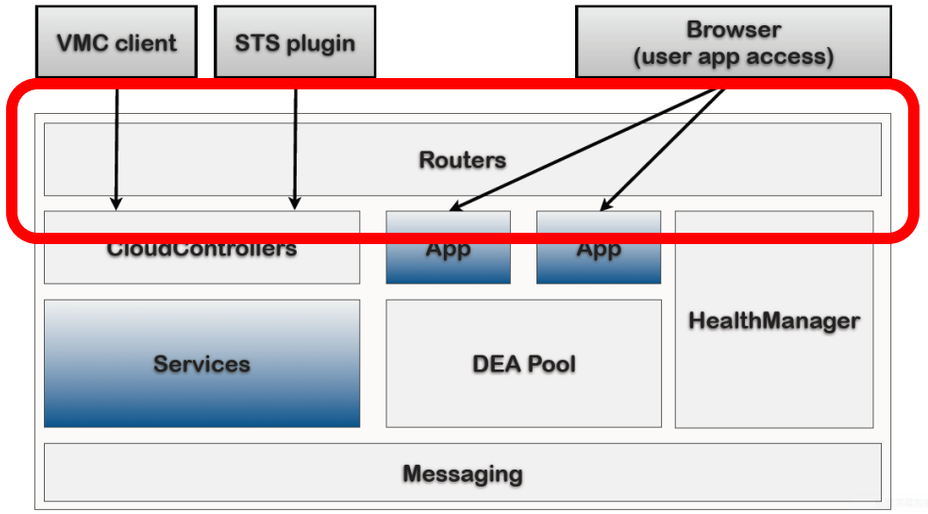
消息总线

- **寻址和发现**
  - 不需要静态IP和DNS查询
  - Layer 4协议
- **命令和控制**
- **中央通讯系统**
- **Dial tone, fire and forget**
- **Protects *itself* at all costs**
- **Idempotent Semantics**



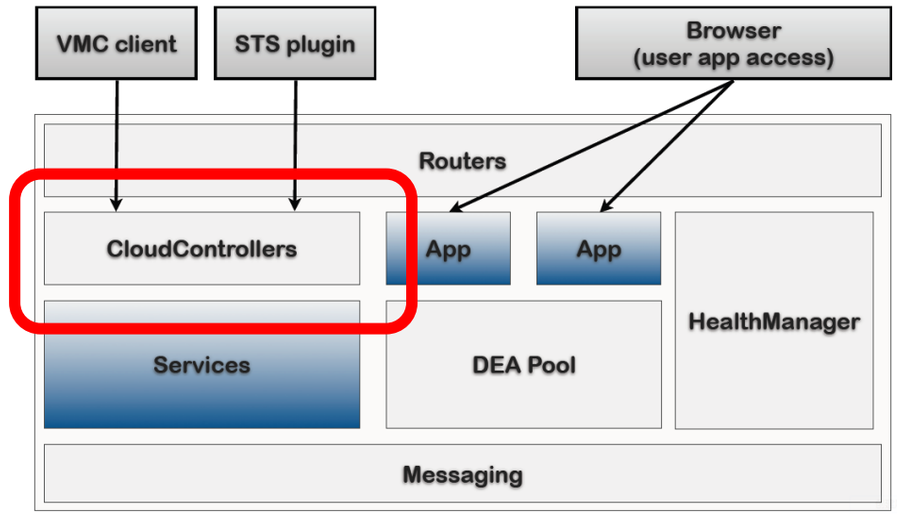
路由引擎

- 处理所有HTTP流量
- 从DEAs获得信息并实时更新路由表
- 维护分布式路由状态
- 将对URL的访问路由至具体的应用
- 在应用实例之间分发流量（均衡负载）



大管家：Cloud Controller

- 处理所有的状态（state）变化
- 控制用户、应用和服务
- 对应用进行打包和预处理*
- 将服务绑定到应用
- 对外提供REST API接口
  - 可在Github中查看
  - https://github.com/cloudfoundry/cloud_controller/blob/master/cloud_controller/config/routes.rb

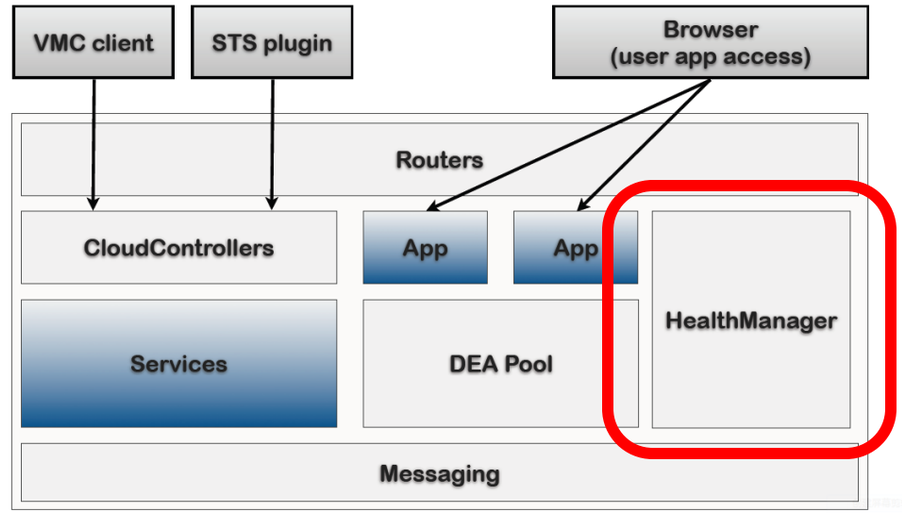



```
# The priority is based upon order of creation:
# first created -> highest priority.
# Routes with asterisks should go at the end of the file if they are ambiguous.
CloudController::Application.routes.draw do
  get    'info'                    => 'default#info',          :as => :cloud_info
  get    'info/services'           => 'default#service_info',  :as => :cloud_service_info
  get    'info/runtimes'           => 'default#runtime_info',  :as => :cloud_runtime_info
  get    'users'                   => 'users#list',            :as => :list_users
  post   'users'                   => 'users#create',          :as => :create_user
  get    'users/*email'            => 'users#info',            :as => :user_info
  delete 'users/*email'            => 'users#delete',          :as => :delete_user
  put    'users/*email'            => 'users#update',          :as => :update_user
  post   'users/*email/tokens'     => 'user_tokens#create',    :as => :create_token
  post   'apps'                    => 'apps#create',           :as => :app_create
  get    'apps'                    => 'apps#list',             :as => :list_apps
  get    'apps/:name'              => 'apps#get',              :as => :app_get
  put    'apps/:name'              => 'apps#update',           :as => :app_update
  delete 'apps/:name'              => 'apps#delete',           :as => :app_delete

  put    'apps/:name/application'  => 'apps#upload',           :as => :app_upload
  get    'apps/:name/crashes'      => 'apps#crashes',          :as => :app_crashes
  post   'resources'               => 'resource_pool#match',   :as => :resource_match
  get    'apps/:name/application'  => 'apps#download',         :as => :app_download
  get    'staged_droplets/:id/:hash' => 'apps#download_staged', :as => :app_download_staged
  get    'apps/:name/instances'    => 'apps#instances',        :as => :app_instances
  get    'apps/:name/stats'        => 'apps#stats',            :as => :app_stats
  get    'apps/:name/update'       => 'apps#check_update'
  put    'apps/:name/update'       => 'apps#start_update'
```

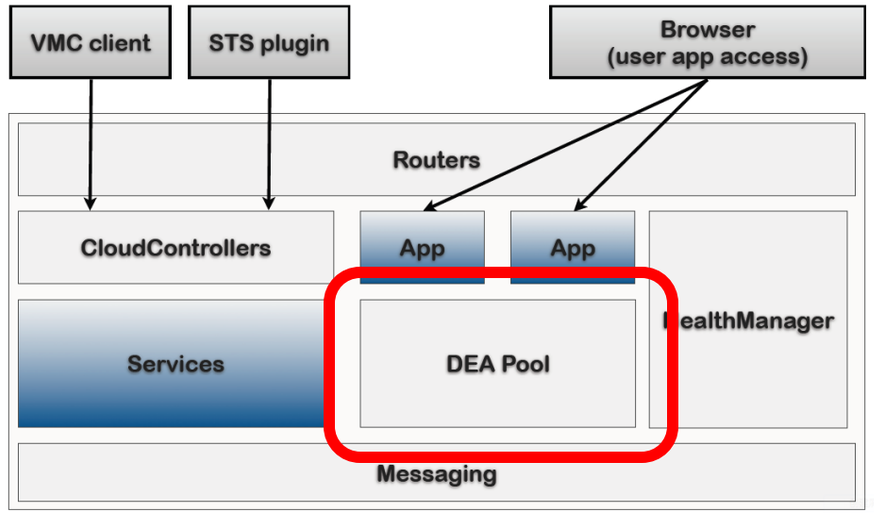
小护士：HealthManager

- 负责监控应用和服务的状态
- 如果出现状态异常，通知CloudController
- 没有改变应用或服务状态的权力（只读）
- Initial value with real-time delta updates to intended vs. real
- Determines drift



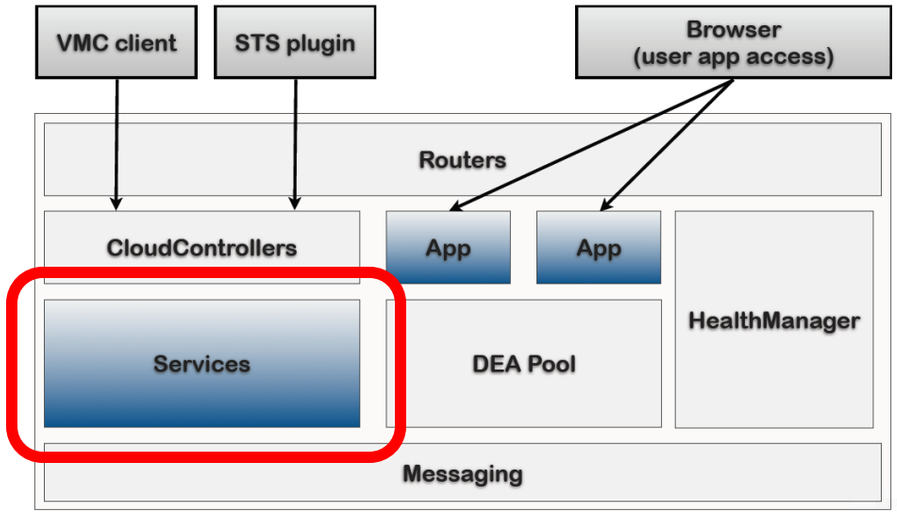
DEA

- **Droplet 和 Droplet Execution Agent**
- **负责运行所有的应用**
- **监控应用的具体运行参数**
  - CPU，内存，IO，线程，磁盘，等等
- **对于DEA来说，所有应用都是一样的**
- **“concept” of ability and desire to run an application**
  - runtimes, options, cluster avoidance, memory/cpu
- **Alerts on any change in state of applications**
- **Provides secure/constrained OS runtime**
  - Hypervisor, Unix File and User, Linux Containers
  - Single or Multi-Tenant

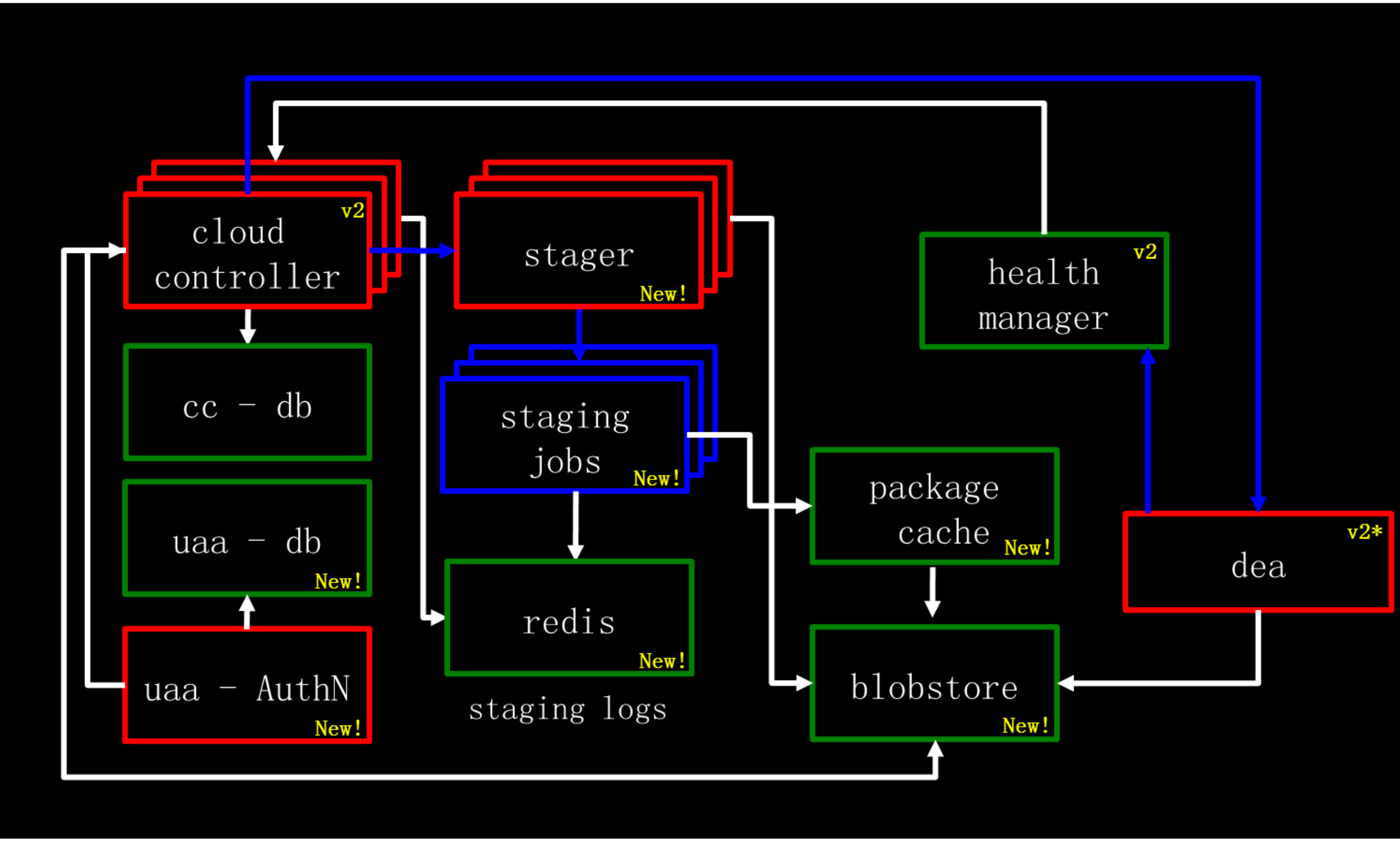


服务

- 一个可扩展/扩充的层
- Service Gateway和Service Node
- 绑定到应用
- 可以被共享
- 服务API用来发现、列表和provision
- Provision之后，应用对服务的访问是直接的
- 很容易将各类应用和各类服务自动绑定



内核重构的路线图


cloud controller
v2
stager
New!
health manager
v2
cc - db
staging jobs
New!
package cache
New!
v2*
dea
uaa - db
New!
redis
New!
staging logs
uaa - AuthN
New!
blobstore
New!

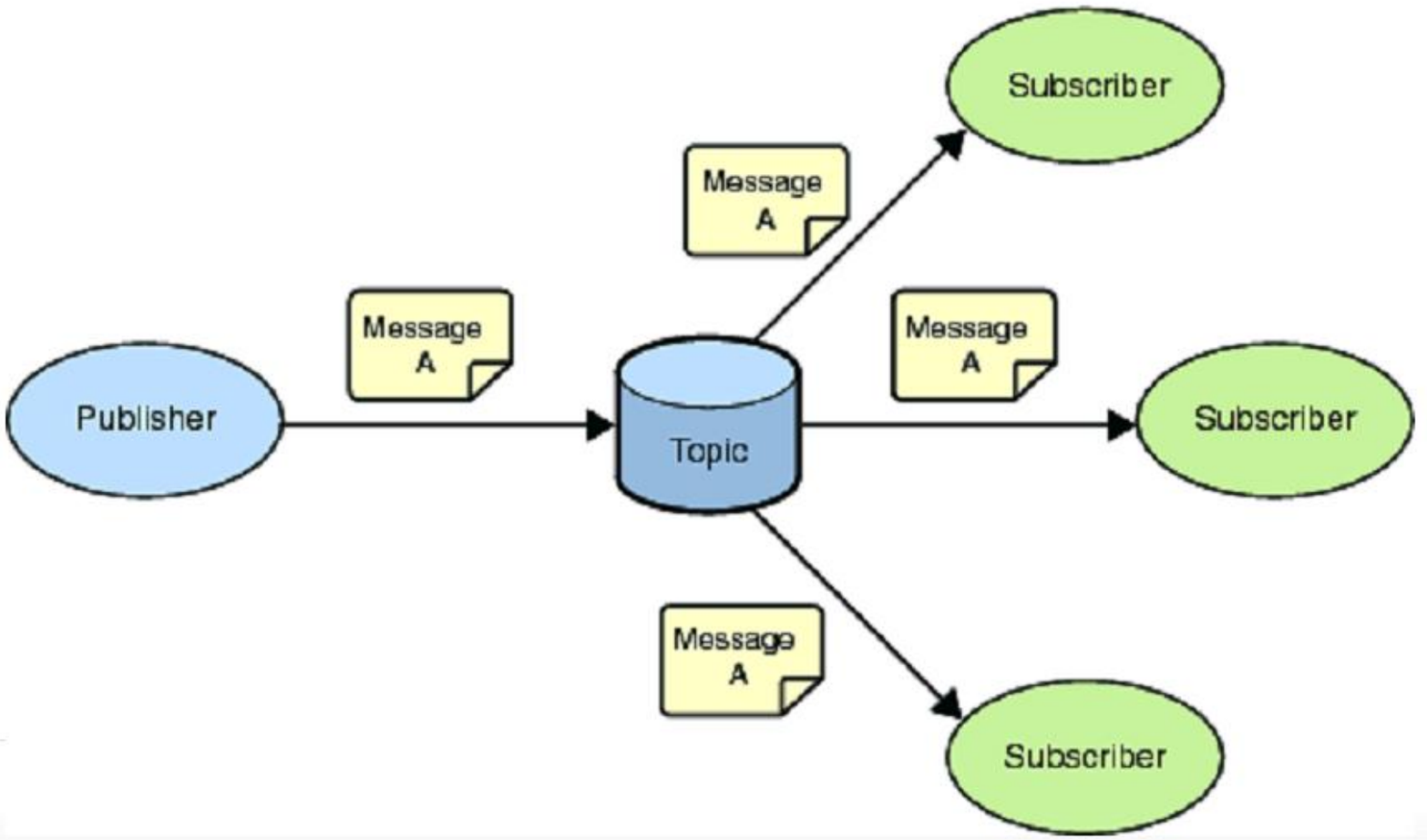
Cloud Foundry的NATS模块

什么是NATS

- **一个基于事件驱动的轻量级支持消息发布、订阅的系统**
- **使用Ruby实现**
- **依赖以下模块构建**
  - eventmachine
  - json_pure
  - daemons
  - Thin
- **https://github.com/derekcollison/nats/**

消息的发布和订阅

- 基于Topic
- 发布者以Topic发送消息
- 订阅者订阅特定Topic并收到



Cloud Foundry下的NATS用例

- **DEA启动和处理应用运行**
  - DEA: NATS.publish('dea.start', @hello_message_json)
    - @hello_message_json for UUID, IP, Port, Version of the DEA

- **Cloud Controller需要启动一个Droplet的实例**

  - DEA: NATS.subscribe('dea.#{UUID}.start') {|msg|process_dea_start(msg)}
    - Send message with the topic of "dea.#{uuid}.start" to start this DEA
    - 对自己UUID的消息主题进行订阅

- Cloud Controller: NATS.publish('dea.#{dea_id}.start',json)
  - DEA get the msg, call process_dea_start(msg)
  - msg = droplet_id, instance_index, service, runtime

NATS的源代码

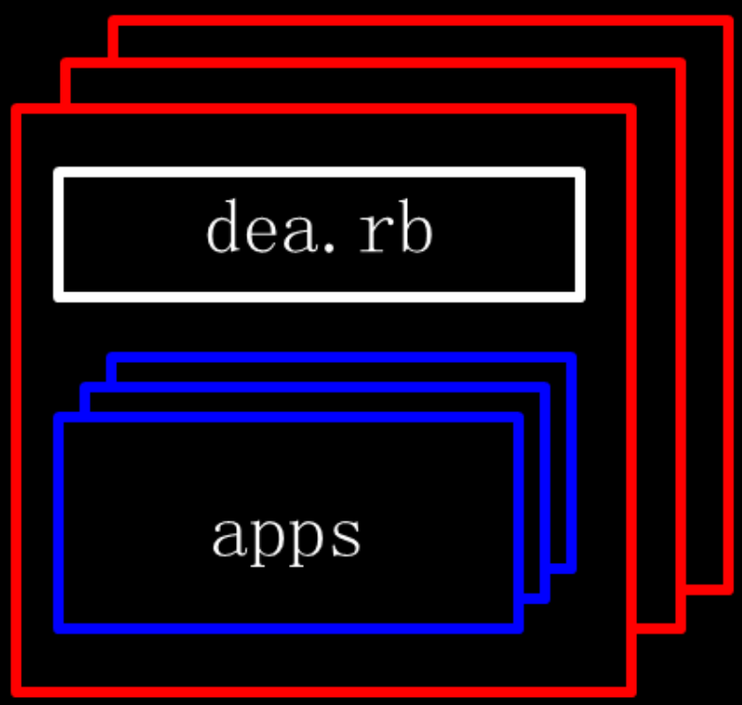
- **lib**
  - nats/client.rb #NATS client
  - nats/server.rb #NATS server wrapper
  - nats/server/server.rb #NATS server
  - nats/server/connection.rb #Connection processing
  - nats/server/sublist.rb #Subscriber manager
  - nats/server/options.rb
  - nats/ext
- **bin #commands to start client and server**
- **spec #test cases**
- **examples**
- **benchmark**

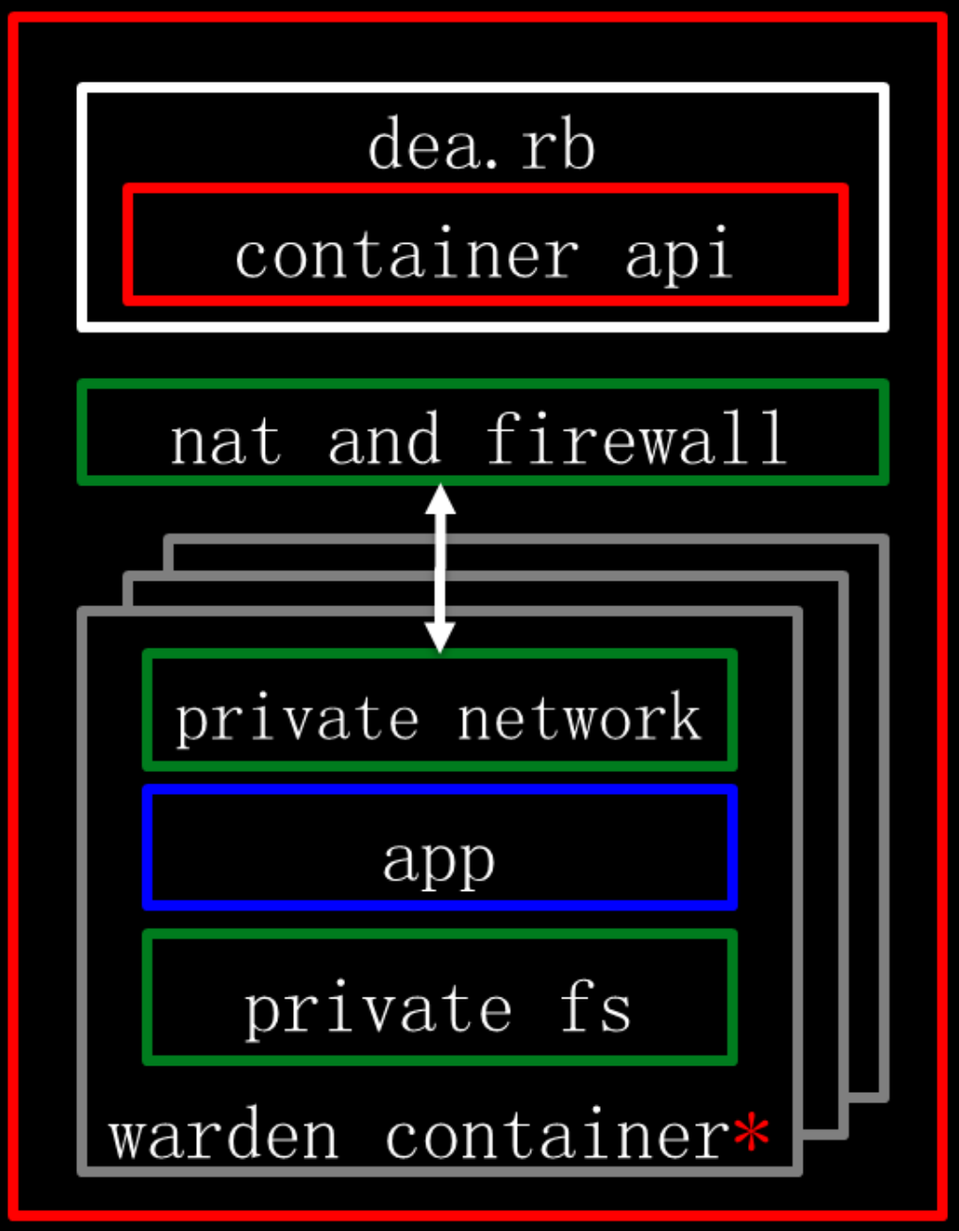
幕后英雄：EventMachine

- **高速可伸缩的事件驱动I/O框架**
- **Reactor Pattern（反应堆模式）的Ruby实现**
  - 将网络逻辑层从应用逻辑层中抽离。无需操心底层的网络连接和 Socket 逻辑处理
- **性能优先**
  - 引入了一种避免使用线程的编程模型
  - 轻量级的并且支持系统底层网络基础操作
  - 关注高性能的部分都是由 C/C++ 实现
  - 操作系统的最优特性（比如Linux 下的 epoll）
- **http://rubyeventmachine.com/**
- **http://www.infoq.com/cn/news/2008/08/eventmachine**
- **Research on EventMachine**
  - http://blog.csdn.net/resouer/article/details/7975550
  - http://www.igvita.com/2008/05/27/ruby-eventmachine-the-speed-demon/

Warden Container

DEA的重构和增强





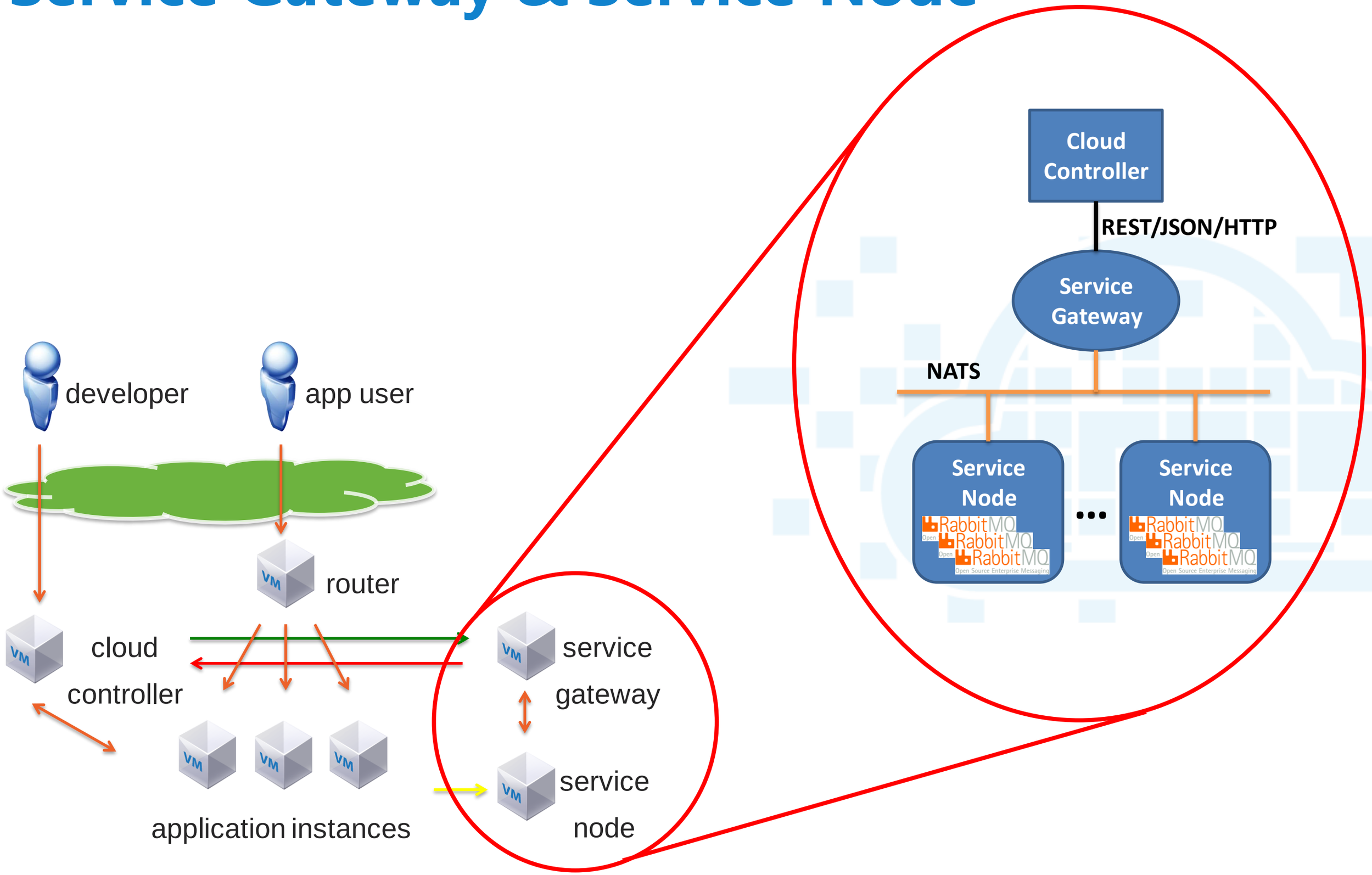
* also running in our Jenkins workers

Warden Container

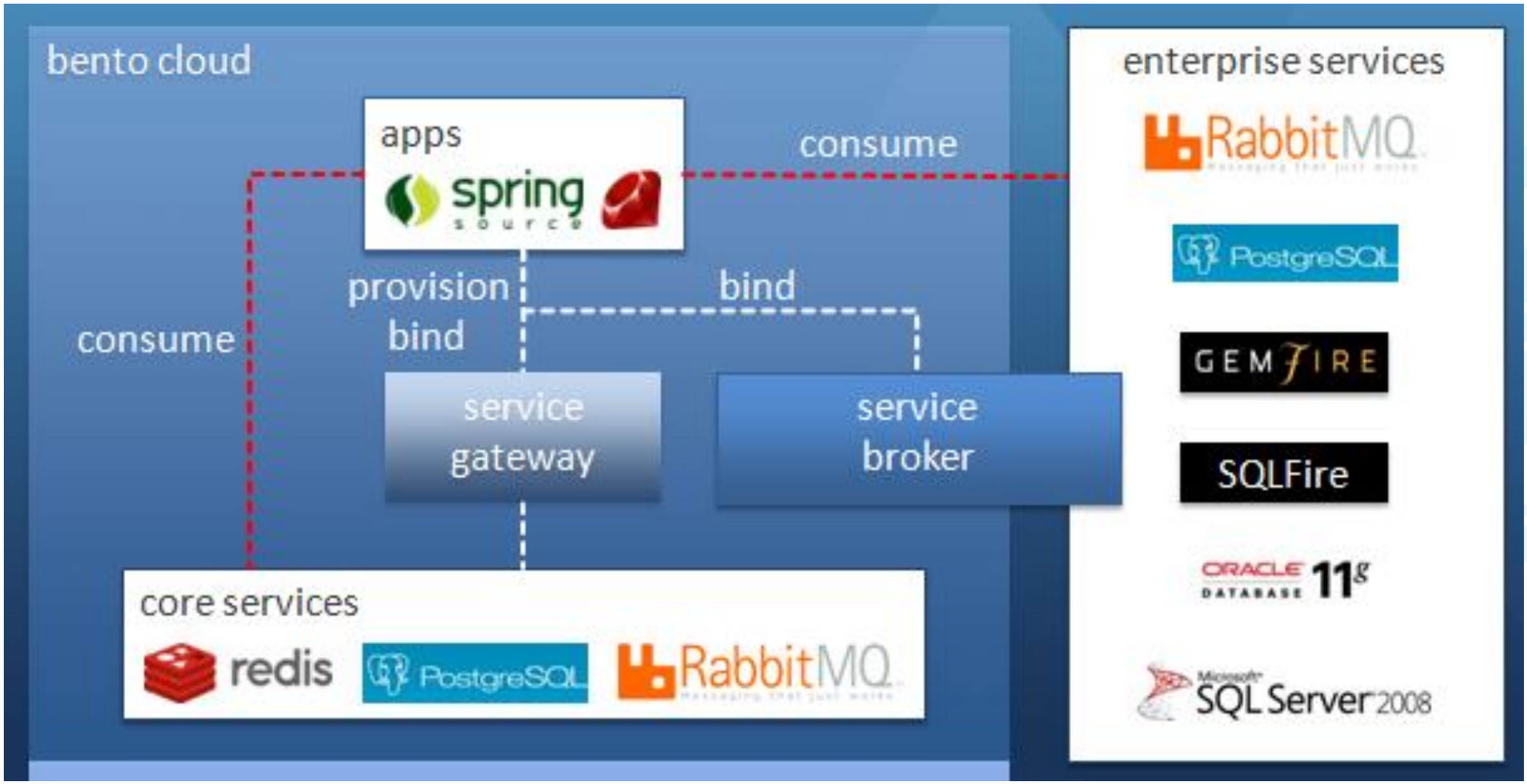
- **源自于Cgroup**
- **非硬件类Hypervisor的一种隔离技术**
- **起源于Google**
  - KVM虚拟机是Linux的一个进程，因此Google采用了Cgroups机制来进行资源管理
  - 用户态的虚拟化技术
  - Cgroup控制KVM虚拟机可以使用的物理资源，包括CPU占用率，CPU核心，内存等
- **Warden是一个孤立的环境，Droplet只可以获得受限的CPU,内存，磁盘访问权限，网络权限**
- **目前在GCE，OpenShift等IaaS/PaaS平台中广泛使用**
- **满足应用隔离要求的同时，获得高性能和伸缩性**
- **不直接使用Cgroup，以确保Cloud Foundry DEA对操作系统无依赖性**

数据及服务的整合

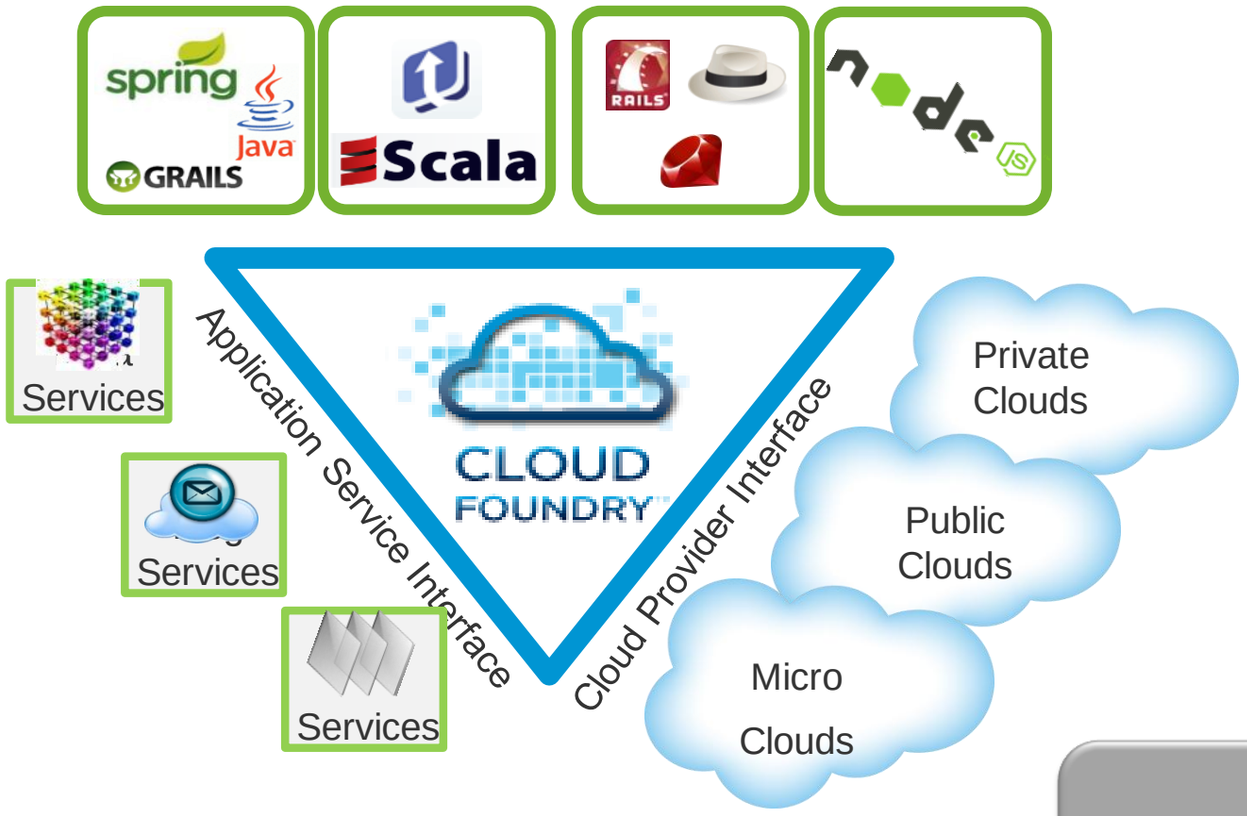
Service Gateway & Service Node

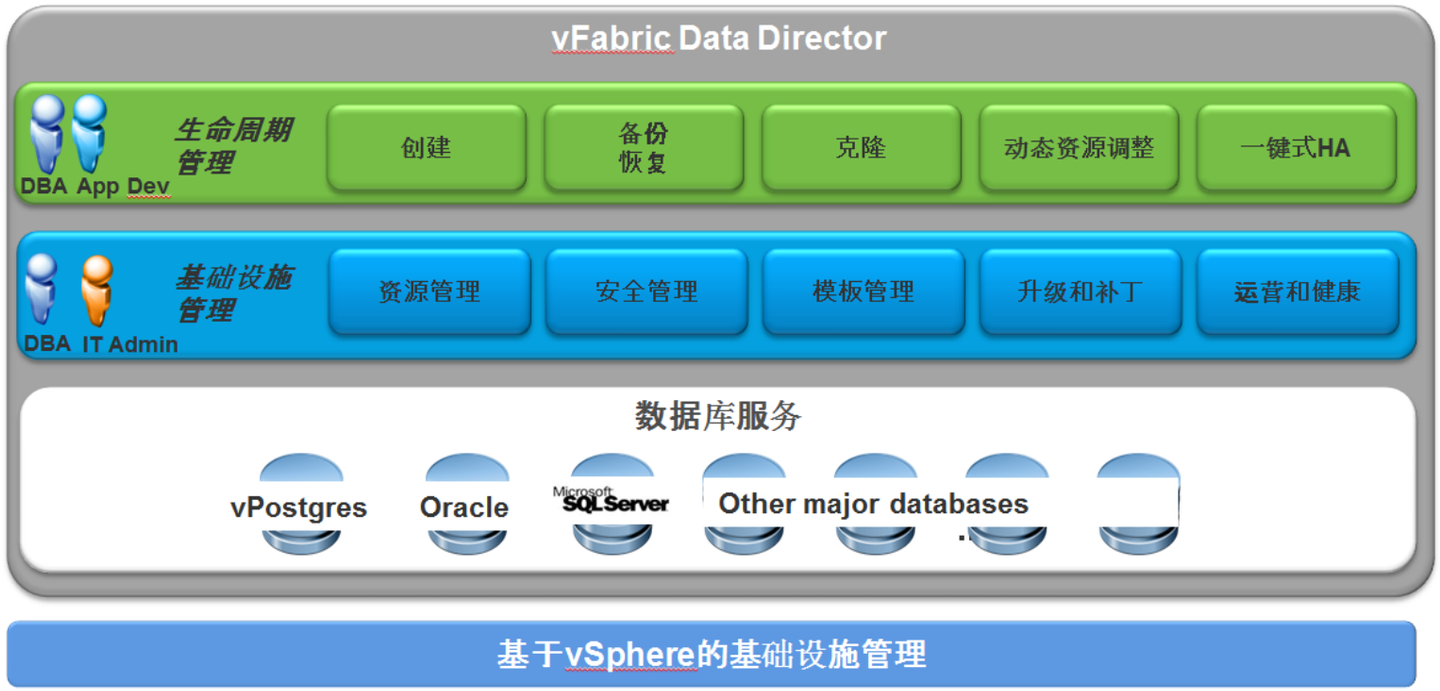


Service Broker

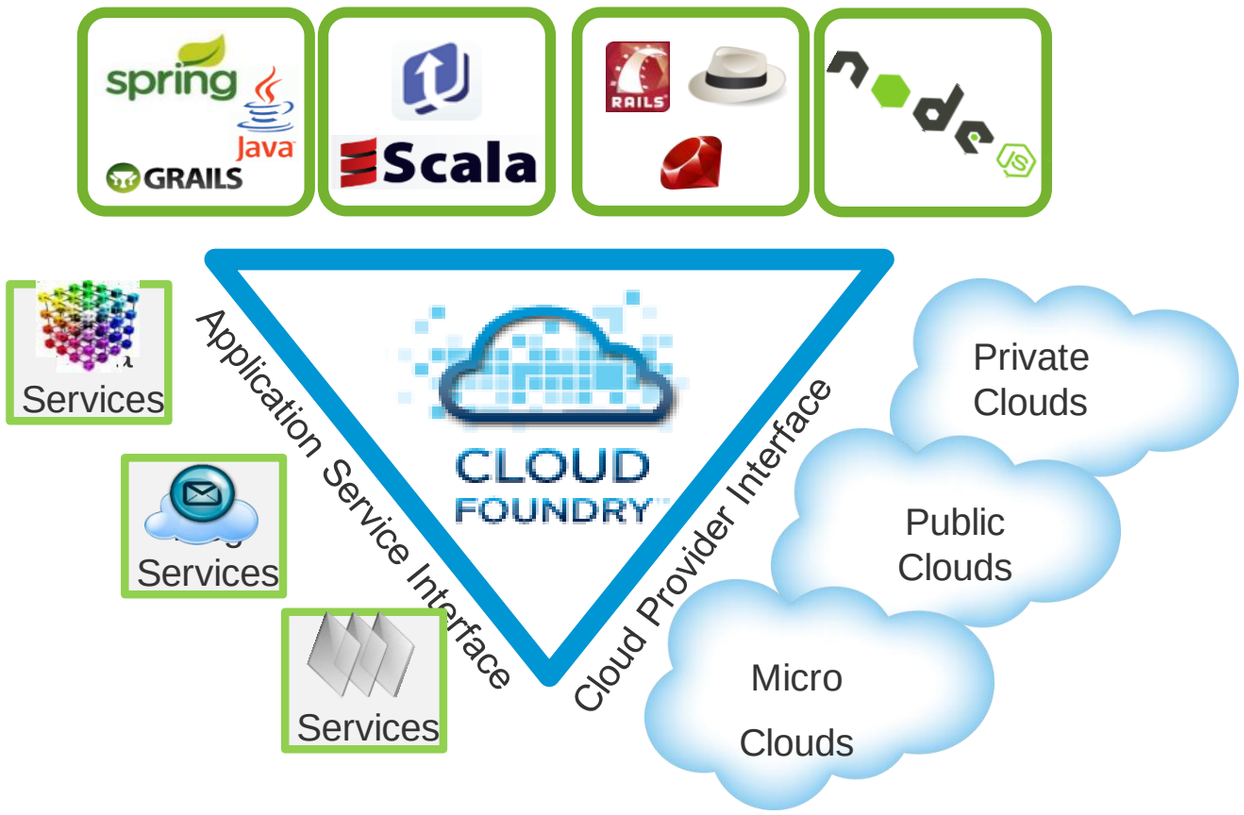


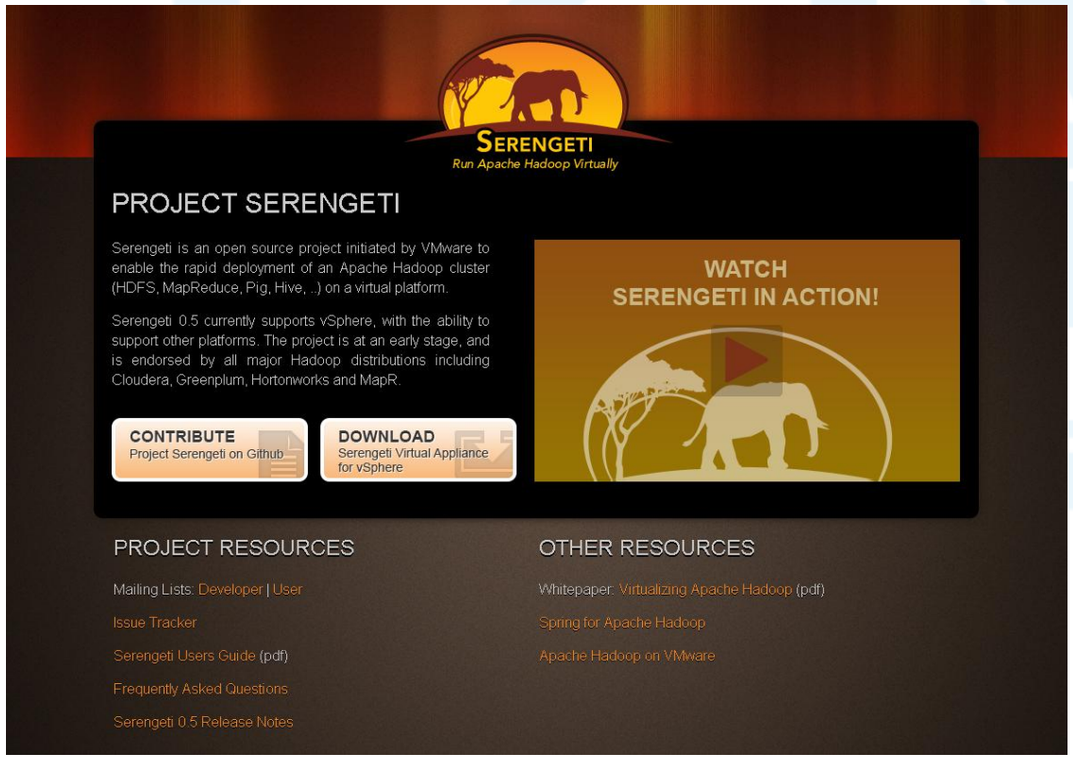
Cloud Foundry + Virtualized Database





Cloud Foundry + Virtualized Hadoop





Open source project that leverage virtualization to simplify Hadoop deployment and operations

http://serengeti.cloudfoundry.com

小结

- 选择高效的编程语言
- 用消息订阅代替模块之间的接口
- 来自架构的弹性
- 细节之处的优化
- 利用现有的技术
- 与IaaS层的紧密结合，获得平台级的弹性

Q&A

更多内容，请关注官方微博：

@CloudFoundry