# cloudfoundry说明

## 简介

Cloud Foundry 是一个现代应用程序平台，专为简化云时代应用程序的端到端开发、部署和运行而构建。 Cloud Foundry 将在多种框架中构建的不同种类的应用服务和应用程序精心编排在一起，并在不同的云基础架构中自动部署应用程序及其基础架构。

CloudFoundry使用Ruby语言进行编写和实现。主要包含Router、CloudController、NATS、DEA、Stager、HealthManager、blobstore、ServiceGateway、ServiceNode等组件，消息为通信核心，协调各个组件模块的运行。扩展Nginx，实现网络请求负载均衡。通过服务网关模式管理cloudfoundry内部的MySQL、Redis、Rabbit等扩展服务。支持Ruby、Java等多语言的应用，目前包含了JDK 5、JDK7等运行环境，内嵌了支持Java的 Web服务器 Tomcat等。

## 架构图





CloudFoundry使用Ruby语言进行编写和实现。CloudFoundry是一套基于消息的分布式系统，面向消息的架构是它节点横向扩展，组件自发现等云特性的基础。

基于消息的多组件架构，消息可以使集群节点间解耦，使自注册，自发现这些在大规模数据中心中很重要的功能得到实现；

适当的抽象层，模板模式的使用，方便第三方可以方便在CloudFoundry开发扩展功能。CloudFoundry在DEA及 Service层都做了抽象层处理，相对应地使开发者可以容易地为CloudFoundry开发Runtime和Service。例如，在 CloudFoundry刚推出的时候，只支持Node.js,Java, Ruby，但第三方提供商、开源社区快速跟进，为CloudFoundry添加了PHP,Python的支持。

## CloudFoundry运行分析

启动各个模块：Router、NATS、CloudController、DEA、ServiceGateway、ServiceNode。

### NATS

消息服务处理中心，基于EventMachine开源项目和Reactor模式。

### Router启动

获取IP参数

启动Thin server

连接NATS

订阅消息

向系统注册Router自己

设置vars和sweeper

定期发布一个Router已启动的消息

### CloudController启动

CloudController是一个Ruby on Rails程序，使用MVC架构。

### DEA启动

Droplet Execution Agency

初始化状态

订阅消息

|  |
| --- |
| dea.status |
| droplet.status |
| dea.discover |
| dea.find.droplet |
| dea.update |
| dea.stop |
| dea.#{uuid}.start |
| router.start |
| healthmanager.start |
| dea.locate |

恢复已存在应用的状态

定期发送状态消息：heartbeat等

发布已启动DEA的消息

# 配置工具

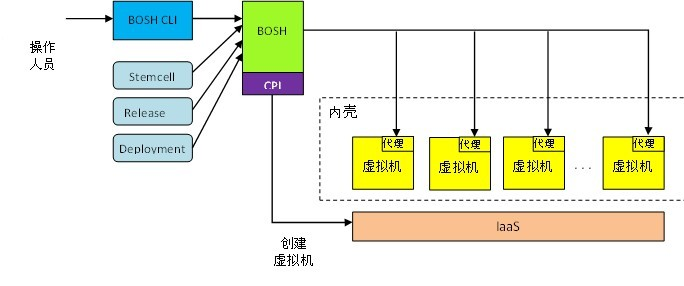
## VMC

通过VMC进行的操作请求通过http方式发送cloudfoundry的cloudcontroller上进行处理。

VMC和STS与CloudFoundry通信采用的是 restful接口，另一方面CloudController是一个典型的Rubyon Rails项目，从VMC或者STS接到JSON格式的协议，然后写入CloudController Database，并发消息到各模快去控制管理整个云。VMC 与 CC 连接的桥梁：[Gem rest\_client](https://github.com/adamwiggins/rest-client" \t "_blank)。

# BOSH

## 结构图



BOSH安装的基本内容：

一个作为虚拟机模板的stemcell、一个作为待部署软件的 Release，以及一个用来定义部署配置的部署清单文件。

BOSH安装基本步骤：

下载、上传虚拟机模板的stemcell

创建、上传一个作为待部署软件的 Release

准备部署清单文件

根据部署清单文件进行发布

通过BOSH安装多节点、分布式的cloudfoundry时，需要有如下条件：

充足的IP地址

充足、稳定的内存、硬盘、CPU等硬件

稳定的网络

需要操作的内容：

使用vCenter分配硬件资源

设置NFS Server

网关路由(可选)

## BOSH的机制

BOSH 视作一台负责协调分布式系统部署过程的服务器或机器人。有个Ruby 工具可以与 BOSH 命令行界面 (CLI) 进行交互。BOSH 需要以下三个必备项才能开始部署系统：一个stemcell、一个Release（要安装的软件）和一个部署清单(Deployment Manifest)。

Stemcell：在云平台中，虚拟机通常是从模板克隆而来的。一个stemcell就是一个包含标准 Ubuntu Linux的虚拟机模板。该模板中还嵌入了一个 BOSH 代理，以便 BOSH 可以控制从该stemcell克隆出来的虚拟机

Release：Release包含若干组将要安装到目标系统上的软件代码和配置。每个虚拟机上都要部署一组软件，这组软件称作一个作业（job）。

部署：部署就是使静态的Release变成虚拟机上可运行的软件的过程。部署清单定义了部署所需的实际参数值。在部署过程中，BOSH 会替换掉Release中的参数，从而使软件按照我们规划的配置来运行。

当上述 3 项内容都准备好后，BOSH CLI 工具会将它们上传到 BOSH。接着，用BOSH 来安装分布式系统包括以下主要步骤：

1) 如果Release中的某些包需要编译，BOSH 首先会创建几个临时虚拟机（worker，工作者虚拟机）来编译它们。编译完后，BOSH 便会销毁这些工作者虚拟机，将所产生的二进制代码存储在其内部blobstore中。

2) BOSH 创建一个虚拟机池，池中的虚拟机将成为该Release要部署到的节点。这些虚拟机是从装有 BOSH 代理的stemcell克隆而来的。BOSH使用CPI接口调用vSphere的虚拟机创建操作API，来自动化的完成虚拟机的创建和配置工作。CPI接口同样适合于OpenStack、AWS等其他IaaS管理平台。

3) 对于该Release的每个作业，BOSH 会从该池中选取一个虚拟机，然后根据部署清单更新该虚拟机的配置。具体的配置可以包括 IP 地址、持久磁盘的大小等。

4) 重新配置完该虚拟机后，BOSH 会向每个虚拟机内的代理发送命令。这些命令通知该代理安装软件包。在安装期间，该代理可能会从 BOSH 下载包并安装它们。安装完毕后，该代理会运行启动脚本来启动该虚拟机的作业。

5) BOSH 重复执行第 3 步至第 4 步，直至所有作业都部署完毕并启动为止。这些作业可以同时部署，也可以按顺序部署。清单文件中的“max\_in\_flight”值用于控制并行部署的作业数。当该值为 1 时，表示这些作业逐一按顺序部署。对于较慢的系统，该值有助于避免因资源拥塞而造成的超时。该值大于 1 时，表示这些作业可以并行部署。

## 部署要求

### 硬件环境

6 台服务器，每台服务器有 8 核 CPU和 32GB RAM。同时需要考虑有比较高吞吐量的网卡和存储设备。存储最好应有 500 GB 或更大的可用空间，以便保存所有虚拟机的映像。在生产系统中，建议采用快速的共享存储。NFS 是用来在Hypervisor间共享存储的最常用协议。在试验环境中，可以使用基于 Linux 的 NFS 服务器来代替专用存储。

保证充足的硬盘容量（创建cloudfoundry release时需要大容量的硬盘空间）。

### 网络环境

有较高吞吐量的网卡；稳定可靠的交换机、路由器。

### 软件环境

1) 64 位 Ubuntu 10.04 LTS，最好是 ISO 格式。

2) vSphere V5.x（本文采用vSphere 作为hypervisor）

3) vSphere Client

4) vCenter（安装在 Windows 2008 R2 64 位或 Windows 2003 服务器上，物理机或虚拟机皆可）

所有对部署和安装的时间根据实际的硬件环境和网络环境而不同。

### 部署内容

#### 基本配置

|  |  |  |  |
| --- | --- | --- | --- |
| **组件** | **节点数目** | **操作系统** | **备注** |
| vCenter  vSphere Client | 1 | win 2008 | 可以安装在一起，也可划分成两个节点 |
| BOSH CLI | 1 | Ubuntu 10.04 LTS | 可以在虚拟机或物理机上 |
| Micro BOSH | 1 | Ubuntu 10.04 LTS | 只能在虚拟机上 |
| BOSH | 7 | Ubuntu 10.04 LTS | 只能在虚拟机上 |
| Cloudfoundry | 15 | Ubuntu 10.04 LTS | 只能在虚拟机上 |

节点总数：25.

**BOSH节点部署清单**

IP网段为：192.168.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **组件** | **节点数** | **IP** | **内存** | **硬盘(基本)** | **硬盘(持久化)** | **job** |
| nats | 1 | 31 | 512M | 2048M |  | nats |
| postgres | 1 | 32 | 512M | 2048M | 2048M | postgres |
| redis | 1 | 33 | 512M | 2048M |  | redis |
| director | 1 | 34 | 1048M | 8192M | 2048M | director |
| blobstore | 1 | 35 | 512M | 2048M | 20480M | blobstore |
| health\_monitor | 1 | 36 | 512M | 2048M |  | health\_monitor |
| powerdns | 1 | 37 | 512M | 2048M |  | powerdns |

**Cloudfoundry节点部署清单**

IP网段为：192.168.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **组件** | **节点数** | **IP** | **内存** | **硬盘(基本)** | **硬盘(持久化)** | **job** |
| nfs\_syslog\_nats\_redis | 1 | 81 | 1024M | 8096M | 8192M | - debian\_nfs\_server  - syslog\_aggregator  - nats  - vcap\_redis |
| router | 1 | 84 | 1024M | 8096M |  | - router |
| cc\_ccdb | 1 | 82 | 1024M | 8096M | 1024M | - cloud\_controller  - postgres |
| uaa\_uaadb | 1 | 83 | 1024M | 8096M | 1024M | - uaa  - postgres |
| stager\_dea | 2 | 90 | 2536M | 10096M |  | - stager  - dea |
| 91 | 2536M | 10096M |  |
| hm | 1 | 97 | 1024M | 8096M | 1024M | health\_manager\_next |
| hbase\_slave | 1 | 99 | 1024M | 8096M | 2048M | hbase\_slave |
| hbase\_master | 1 | 98 | 1024M | 8096M | 2048M | hbase\_master |
| opentsdb | 1 | 86 | 1024M | 8096M |  | opentsdb |
| collector | 1 | 87 | 1024M | 8096M |  | collector |
| dashboard | 1 | 95 | 1024M | 8096M |  | dashboard |
| login | 1 | 96 | 1024M | 8096M |  | login |
| mysql\_gateway\_node\_100 | 1 | 85 | 900M | 8096M | 1315M | - mysql\_gateway  - mysql\_node\_ng |
| rabbit\_gateway\_node\_100 | 1 | 88 | 900M | 8096M | 2584M | - rabbit\_gateway  - rabbit\_node\_ng |

说明：

对于dashboard节点，它的关联job是存在依赖关系的，所以要保证他们在yml里的描述定义按照如下顺序：hbase\_slave，hbase\_master，opentsdb，collector，dashboard。

# Cloudfoundry的特性

## 支持多框架

Cloud Foundry 支持下列应用程序开发框架：

* Spring
* Ruby on Rails
* Ruby 和 Sinatra
* Node.js
* Grails

支持包括一个运行环境，通过它您的应用程序可以在 Cloud Foundry 上执行，以及运行可以检测框架并自动配置和部署到 Cloud Foundry 的部署工具（vmc 和 STS）。您可能不需要做任何特殊操作就可以部署到 Cloud Foundry。如果应用程序需要数据库等服务，而且您遵循框架的惯例，vmc 或 STS 在部署应用程序时可以处理需要的配置变化。

## 支持多种服务

* [MySQL](http://www.mysql.com/)，开源关系数据库。
* [vFabric Postgres](http://www.vmware.com/products/datacenter-virtualization/vfabric-data-director/features.html)，基于 PostgreSQL 的关系数据库。
* [MongoDB](http://www.mongodb.org/)，基于文档的可扩展开源数据库。
* [Redis](http://redis.io/)，开源键值数据结构服务器。
* [RabbitMQ](http://www.rabbitmq.com/)，用于您应用程序的可靠、可扩展和便携消息传送。

## 添加框架

## 添加运行时

## 添加系统服务

可以用您想用的任何语言编写您自己的服务， 并可以用 Cloud Foundry 支持的任何语言来编写作为使用方的应用程序。 每项服务都有一个服务节点、一个置备器和一个服务网关。服务节点是 Cloud Foundry 服务的实现。 置备器是在置备或取消置备该服务时执行特定于域的操作的代理。 例如，在置备标准的 MySQL 服务时， 该服务会创建一个新用户和架构。 服务网关是用来与服务置备器 进行交互的 REST 接口。

# 应用的部署、维护

## 设定目标站点平台

|  |
| --- |
| $ vmc target api.cf.local |

## 登陆目标站点平台

|  |
| --- |
| $ vmc login |

如果是cloudfoundry安装、部署后，使用“vmc login”第一次登陆目标站点平台，系统将要求您为自己的帐户设置密码。登录后，您可以获得自己 Cloud Foundry 实例的信息。

## 获取平台信息

|  |
| --- |
| $ vmc info |

## 为平台添加用户

输入“vmc register”命令，根据用户注册向导注册用户。

## 发布一个Java web

1. 通过windows启动菜单中“Start Command Prompt with Ruby”，进入命令行模式；
2. 设定目标平台；

|  |
| --- |
| vmc target api.local.do2013.cloudfoundry.me |

1. 登陆平台

|  |
| --- |
| vmc login --email [test@do2013.com](mailto:test@do2013.com) --password test |

1. 进入包含java web应用的war包文件的目录；
2. 输入“vmc push”，根据vmc发布应用的流程提示将当前项目发布到cloudfoundry站点。

## 发布standalone模式的java-web应用

以tomcat 做为web server。

步骤1：进入命令行模式窗口

步骤2：进入tomcat的根目录

步骤3：发布应用

|  |
| --- |
| $ vmc push mytomcat7 |

执行命令后，出现如下的应用发布向导，根据提示进行操作。

|  |
| --- |
| Would you like to deploy from the current directory? [Yn]:  Detected a Standalone Application, is thiscorrect? [Yn]:  1: java  2: node  3: node06  4: ruby18  5: ruby19  Select Runtime 1: 1  Selected java  Start Command: bin/startup.sh  Application Deployed URL [None]: mytomcat7.cloudfoundry.com  Memory reservation (128M, 256M, 512M, 1G, 2G) [512M]: 256M  How many instances? [1]:  Bind existing services to 'mytomcat7'? [yN]:  Create services to bind to 'mytomcat7'? [yN]:  Would you like to save thisconfiguration? [yN]: y  Manifest written to manifest.yml.  Creating Application: OK  Uploading Application:    Checking foravailable resources: OK    Processing resources: OK    Packing application: OK    Uploading (23K): OK  Push Status: OK  Staging Application 'mytomcat7': OK  Starting Application 'mytomcat7': OK |

## 查看应用状态

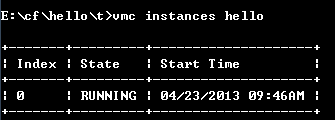
查看已部署的应用列表

|  |
| --- |
| $ vmc apps |



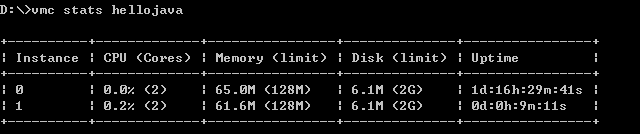
查看应用的实例个数

|  |
| --- |
| $ vmc instances appName |



查看应用实例状态

|  |
| --- |
| vmc status appName |

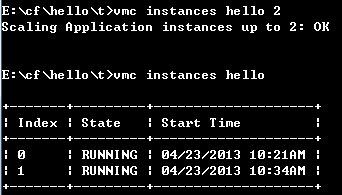


## 多实例故障迁移

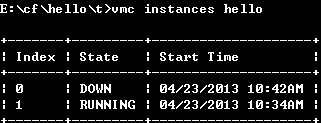
动态创建应用的多个实例

|  |
| --- |
| $ vmc instances hello 2 |

该命令会根据服务器的运行状况，自动将实例部署在空闲资源较多的服务器上。



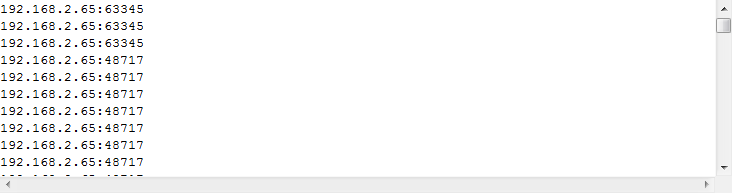
关掉其中一台服务器，查看此时应用的实例状态，hello实例只有一个在运行



通过浏览器可以正常访问hello.cf.offline，并且可以清楚的看到访问的是未停止的实例。

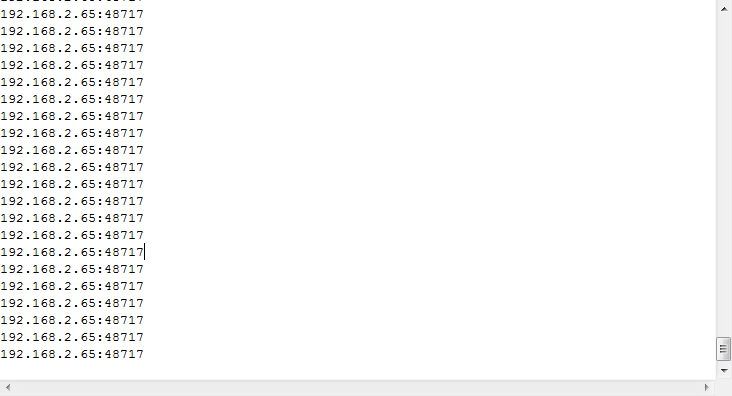
## 负载均衡测试

部署两个应用实例（有时两个实例部署在了同一dea上），开启100个线程访问http://tom7.cf.local/index.jsp，基本可以平均的访问（51:49）



## Session粘性测试

如果测试的所有线程中设置同一session，所有的请求都是发送到了同一应用实例



## 为dashboard添加用户

**步骤1：设定目标站点**

|  |
| --- |
| $ uaac target http://uaa.<YOURtarget>.com |

**步骤2：获取特定客户端的令牌**

|  |
| --- |
| $ uaac token client get dashboard --secret <copy-paste the secret from UAA clients> |

**步骤3：为uaa增加用户**

在执行uaac member add之前需要先往uaa中增加用户，命令如下。

|  |
| --- |
| $ uaac user add test@do2013.com --given\_name test@do2013.com --e  mails [test@do2013.com](mailto:test@do2013.com) --family\_name test@do2013.com |

执行uaac user add命令时，一定要带上--given、--emails、--family\_name 等参数，否则在创建用户时会提示email必输项无值。如果不带--given、--family\_name等参数，则在访问dashboard页面时会提示服务器端有错误信息。

通过uaac命令新增加的用户会出现uaadb中。

**步骤4： 添加成员**

|  |
| --- |
| $ uaac member add dashboard.user <username1>... <usernameN> |

上面的username1是通过vmc注册到cloudfoundry中的用户。

**步骤5： 清除令牌**

|  |
| --- |
| $ uaac token delete |

# 参考文档

CloudFoundry参考手册.docx

BOSH参考手册.docx

CloudFoundry整体分析.docx