# 介绍

CloudFoundry 是VMware主导使用Ruby开发的一款开源PaaS云计算平台

Email: [tchen.data@gmail.com](mailto:tchen.data@gmail.com)  
Temporary password: nwtRSPsX treat-medallion-lack

[test@do2013.com](mailto:test@do2013.com) test

Ubuntu: do2013 do

**域名：**

do2013.cloudfoundry.me' domain has been reserved

Please use the token below to configure your Micro Cloud Foundry.

The configuration token is: spoke-garbage-lace

**单机域名**：

offsite: local.do2013.cloudfoundry.me

vmc target api.local.do2013.cloudfoundry.me

local.do2013.cf.me

Micro Iron Foundry: pass/2013

# MCF虚拟机操作步骤：

## 注意事项：

安装Ruby时，Ruby的安装目录不要带有空格，否则会影响DevKit的使用。

## 安装软件

|  |  |
| --- | --- |
| 1. 虚拟机 [VMware Player](http://www.vmware.com/products/player/overview.html) |  |
| 1. Ruby and RubyGems | Download and install [Ruby Installer for Windows](http://www.rubyinstaller.org/). The installer already includes RubyGems |
| 1. vmc | 通过启动菜单中“Start Command Prompt with Ruby”，进入命令行模式，输入‘gem install vmc’安装vmc |
| 1. Micro Cloud Foundry | [Micro Cloud Foundry](https://cloudfoundry.com/micro) |
| 注册一个cloud foundry的用户，使用这个用户下载Micro Cloud Foundry，并在下载Micro Cloud Foundry时，选择一个域名，得到一个token，在后面设置Micro Cloud Foundry时会用到。 |
| 启动和设置Micro Cloud Foundry Virtual Machine  启动之前，为了确保VMPlayer能正常启动Micro Cloud Foundry，需要电脑开机时按F1进入BIOS设置，确认以下内容：  a）如果BIOS中CPU设置有VT-x功能，就将它打开  b）如果BIOS中Security设置里，有虚拟机的相关设置，也要将它们打开。 |
| 启动Micro Cloud Foundry Virtual Machine  a）解压MCF压缩包  b）启动VMPlayer，打开MCF解压缩目录micro里的micro.vmx文件。通过VMPlayer是打开已存在VMX文件来启动虚拟机，而不是新建一个虚拟机。  c）通过VMPlayer启动（Power on）这个MCF。 |
| 设置Micro Cloud Foundry Virtual Machine  根据提示设置MCF。  设置网络时，为了方便设置MCF选择DHCP。  设置域名时，为了单机模式输入自定义的域名而不输入token。 |

## MCF单机模式设置

Offline mode is only supported with the VM network adapter set to NAT. To share your Micro Cloud Foundry with others, you must set the network adapter to Bridged mode and run Micro Cloud Foundry in online mode.

**Step 1**. VMPlayer的设置：In the VM’s Virtual Machine Settings, select Network Adapter and make sure that NAT is selected. If you have to change the setting, restart the virtual machine.

**Step 2**. In the Micro Cloud Foundry console menu, select option 6 to toggle to offline mode.

在MCF主菜单的第4项 reconfigure domain中，重新设置domain，使用自定义的域名，不使用在cloudfoundry.com/网站上注册用户时得到的token。

**Step 3**. Configure your host computer to route DNS requests to the Micro Cloud Foundry VM. This is accomplished in differing ways depending on the OS and whether you use DHCP or a static IP address. In the instructions that follow, replace the IP number 172.16.52.136 with the IP number shown on the Micro Cloud Foundry console.

Windows：

* Open the Network and Sharing control panel.
* Choose Change adapter settings.
* Right-click VMware Virtual Ethernet Adapter for VMnet8, and choose Properties.
* Set the preferred DNS server to 172.16.52.136.

**Step 4：**重启MCF

## 注册MCF用户（使用vmc）

注册之前，设置MCF为单机模式

Registering a user creates a user account on the Micro Cloud Foundry virtual machine. You log in with this account to publish and manage applications.

Target your Micro Cloud Foundry. In a shell, enter the following command:

$ vmc target api.appname.cloudfoundry.me

Create a new account using the vmc register command:

$ vmc register

Enter your email address. Enter a password and confirm it when requested.

You are now ready to log in with vmc or set up Spring Tool Suite to deploy your applications to Micro Cloud Foundry.

## 发布一个Ruby 的demo

1. 创建一个sinatra例子

通过启动菜单中“Start Command Prompt with Ruby”，进入命令行模式，输入‘gem install sinatra’安装sinatra

Create the directory in which the new application will live. For example:

prompt$ mkdir /usr/bob/sample-apps/hello

Using your favorite text editor, create a file called hello.rb in this new directory with the following contents:

require 'sinatra'

get '/' **do**

"Hello from Cloud Foundry"

**end**

1. 发布一个sinatra例子

Open a terminal window (Linux) or command prompt (Windows) and change the directory that contains your application.

prompt$ cd /usr/bob/sample-apps/hello

Deploy your application using the vmc push command, which interactively prompts for deployment information:

prompt$ vmc push

在发布程序的向导中，在“Application Name”一项中输入应用程序的名字 。

Verify that your application is available by executing the vmc apps command:

$ vmc apps

+--------------+----+--------+-------------------------------+----------+

| Application | # | Health | URLS | Services |

+--------------+----+--------+-------------------------------+----------+

| hello | 1 | RUNNING| hello-bob.cloudfoundry.com | |

+--------------+----+--------+-------------------------------+----------+

Run your application in your browser by going to the URL you provided to the vmc push command, which in the example above is hello-bob.cloudfoundry.com.

## 发布一个Java web

1. 通过启动菜单中“Start Command Prompt with Ruby”，进入命令行模式；
2. 输入“vmc target cloudfoundry站点”，绑定cloudfoundry发布站点；

(vmc target api.local.do2013.cloudfoundry.me)；

1. 输入“vmc login 登陆用户”，绑定cloudfoundry发布站点；

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

1. 进入D:\reference\PAAS\code\cloudfoundry-samples-master\hello-java的target子目录；
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。

## 发布一个使用数据库的Java web

1. 通过启动菜单中“Start Command Prompt with Ruby”，进入命令行模式；
2. 输入“vmc target cloudfoundry站点”，绑定cloudfoundry发布站点；

(vmc target api.local.do2013.cloudfoundry.me)；

1. 输入“vmc login 登陆用户”，绑定cloudfoundry发布站点；

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

1. 进入D:\reference\PAAS\code\cloudfoundry-samples-master\hello-spring-mysql的target子目录；
2. 输入“vmc push”，根据发布流程的提示将当前项目发布到cloudfoundry站点。
   1. 在发布流程提示中，选择“Runtime”时要选择“1：java”。如果选择“2：java7”，无法通过“Checking ….”的验证，根据检查cloudcontroller和dea的日志可以发现“cf 不支持java7 的runtime”。查看日志内容的命令如下：

tail -f cloudfoundry/.deployments/devbox/log/dea.log

* 1. 在发布流程提示中，选择“128M”的Memory Limit。

### 其他：

将数据库连接信息改成Oracle，把oracle jdbc jar放入WEB-INF/lib目录中，同样可以通过发布到CF中的应用去正常访问CF之外的Oracle。

需要注意：如果不关闭主机中的防火墙，虚拟机可能无法访问主机的服务（比如：Oracle）。

查看日志：

tail -f /var/vcap.local/dea/apps/hello\_sql-0-b33c8a067f5b6e68c5f8abf2c36e210/logs/stderr.log

## 通过SSH登陆MCF的ubuntu系统

1. 在MCF的主控制菜单中，选择第三个菜单项，重新设置ubuntu的root用户密码；
2. 使用SecureCRT登陆远程的ubuntu系统，设置远程机器的IP，选择SSH2协议，使用root用户和密码。

# CF单节点的基于源码安装：

[test@do2013.com](mailto:test@do2013.com) test; [t1@test.com](mailto:t1@test.com) t1

## 步骤

1. 安装VM player，Ubuntu 10.0.4
2. 在CF主机上，
   1. 安装SSH：sudo apt-get install openssh-server
3. 在客户端通过SSH登陆CF主机，完成如下操作：
   1. **下载CF的vcap源代码：git clone** <https://github.com/cloudfoundry/vcap.git>
   2. **下载、安装完整CF代码：v**cap/dev\_setup/bin/vcap\_dev\_setup

执行完命令后，会下载**完整CF代码到**$HOME/cloudfoundry/vcap，会建 立一个默认的发布目录：$HOME/cloudfoundry/.deployments/devbox。网络正常的情况下，会安装成功。

* 1. 启动CF：vcap/dev\_setup/bin/vcap\_dev start
  2. 安装vmc： sudo gem install vmc -v 0.3.18 //否则默认是高版本会有错误

1. 在客户端使用VMC访问CF
   1. 使用ifconfig命令在CF主机上查看CF主机的IP
   2. 修改客户端的hosts文件，将api.vcap.me域名映射到CF主机IP地址上。
   3. 在客户端的Ruby环境下，输入：vmc target api.vcap.me 访问CF。
   4. 为了使用vmc注册用户、发布应用等，
      1. 安装SSH
      2. 建立客户端与CF主机通道：ssh -L 80:192.168.150.131:80 do2013@192.168.150.131 -N

当需要在客户端的浏览器上访问发布在CF主机上的域名应用时，需要用此命令建立SSH通道。

1. 离线访问客户端发布到CF上的新域名应用
   1. vmc push发布新的应用和域名
   2. 在客户端的hosts文件中加入新域名与127.0.0.1的映射。例如：

127.0.0.1 api.vcap.me helloruby.vcap.me helloruby35.vcap.me

* 1. vmc push发布到CF上的应用位于CF主机上的/var/vcap.local/dea/apps/目录中。

## 参考资料

1. vcap dev\_setup at master · cloudfoundry vcap · GitHub
2. Single Multi Node VCAP Deployment using dev\_setup CloudFoundry.com Support
3. Multi-node CloudFoundry setup CloudFoundry.com Support
4. ubuntu 12.04上部署cloud foundry

## 备注：

192.168.150.131 api.do2013.me

使用git clone下来vcap包。下载后，会发现dea，router，uaa，acm，cloudcontroller目录下是空的，手动cd到该目录下，执行git clone https://github.com/cloudfoundry/uaa.git（以uaa为例子，其余的同理）。或者cd 到vcap下，执行git submodule update --recursive --init  
然后执行安装脚本： ../vcap/dev\_setup/bin/vcap\_dev\_setup

# CF多节点安装：

常用命令

rm –dfr DIR\_NAME

git clone git://github.com/cloudfoundry/uaa.git

tar

|  |
| --- |
|  |

安装步骤：

My setup is composed of 4 nodes   
- A router node with nginx proxy frontend.   
- A Cloud controller and Health Manager node.   
- A DEA node.   
- A Service node with MongoDB, Redis and MySQL.

The goal of this post is to show you how to intall all the components of Cloud Foundry on multiple nodes.

Step #1 : Create an archive with common files   
- git clone [git@github.com](mailto:git@github.com):cloudfoundry/vcap.git   
- cd vcap   
- git submodule update --init   
- tar -vzcf vcap\_common.tar.gz bin/vcap common/lib/ rakelib/ Rakefile .rvmrc

Step #2 : Create archive for all components :   
- tar -vzcf vcap\_router.tar.gz bin/router router/   
- tar -vzcf vcap\_dea.tar.gz bin/dea dea   
- tar -vzcf vcap\_cloud\_controller.tar.gz bin/cloud\_controller cloud\_controller   
- tar -vzcf vcap\_health\_manager.tar.gz bin/healh\_manager health\_manager   
- tar -vzcf vcap\_services.tar.gz bin/services/ services/

Step #3 : On all nodes, install the common files, rvm and packages :   
- mkdir /home/cloudfoundry/   
- groupadd cloudfoundry   
- useradd -G cloudfoundry -m -b /home/cloudfoundry cloudfoundry   
- cd /home/cloudfoundry   
- apt-get install curl git-core build-essential binutils-doc autoconf flex bison libreadline5-dev zlib1g-dev libssl-dev libxml2-dev libxslt1-dev   
- bash < <( curl -L <http://rvm.beginrescueend.com/releases/rvm-install-head> )   
- rvm update --head ; rvm reload   
- rvm install 1.9.2   
- rvm install 1.8.7   
- rvm --default use 1.9.2   
- mkdir /var/vcap   
- mkdir /var/vcap/sys   
- mkdir /var/vcap/sys/log   
- mkdir /var/vcap/shared   
- mkdir /var/vcap/services   
- chmod -R 777 /var/vcap   
- mkdir "/var/vcap.local"   
- chmod 777 "/var/vcap.local"   
- cd /home/cloudfoundry/   
- copy vcap\_common.tar.gz to /home/cloudfoundry/   
- tar -vzxf vcap\_common.tar.gz   
- apt-get install curl libcurl3 zlib1g-dev libssl-dev libreadline5-dev libxml2 libxml2-dev libxslt1.1 libxslt1-dev sqlite3 libsqlite3-ruby libsqlite3-dev unzip zip rake   
- gem install bundler vmc rack rake thin sinatra eventmachine nats

Step #4 : Deploy Cloud Controller and Health Manager   
- ssh cloud\_controller.vcap   
- copy vcap\_cloud\_controller.tar.gz to /home/cloudfoundry   
- cd /home/cloudfoundry/   
- tar -vzxf vcap\_cloud\_controller.tar.gz   
- apt-get install postgresql postgresql-server-dev-8.4 libcurl3 libcurl3-gnutls libcurl4-openssl-dev  
- gem install pg   
- rake bundler:install   
- edit /home/cloudfoundry/cloud\_controller/config/cloud\_controller.yml and change local\_route to ip address of your node, change the domain name of your installation and mbus configuration:   
local\_route: 10.30.1.4   
external\_uri : api.paasform.com   
mbus: nats://10.30.1.4:4222   
- cd bin/   
- ./vcap start   
- copy vcap\_health\_manager.tar.gz to /home/cloudfoundry   
- cd /home/cloudfoundry   
- tar -vzxf vcap\_health\_manager.tar.gz   
- rake bundler:install   
- edit /home/cloudfoundry/health\_manager/config/health\_manager.yml and change local\_route to ip address of your node and change mbus to ip address of your cloud controller node :   
mbus: nats://10.30.1.4:4222   
local\_route: 10.30.1.4   
- cd bin/   
- ./vcap start

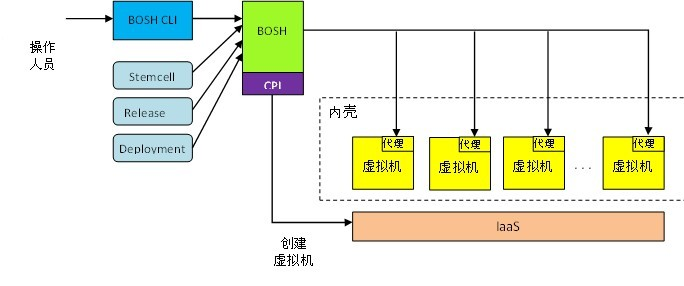
Step #5 : Deploy Cloud Foundry Router on router node :   
- ssh router.vcap   
- apt-get install nginx   
- copy vcap\_router.tar.gz to /home/cloudfoundry/   
- cd /home/cloudfoundry   
- tar -vzxf vcap\_router.tar.gz   
- copy nginx.simple.conf to /etc/nginx/nginx.conf   
- restart nginx   
- rake bundler:install   
- Modify /home/cloudfoundry/router/config/router.yml and change mbus to ip address of your cloud controller node :   
mbus: nats://10.30.1.4:4222   
- cd bin/   
- ./vcap start

Step #6 : Deploy Cloud Foundry DEA on DEA node :   
- ssh dea.vcap   
- copy vcap\_dea.tar.gz to /home/cloudfoundry/   
- cd /home/cloudfoundry   
- tar -vzxf vcap\_dea.tar.gz   
- apt-get install python-software-properties   
- add-apt-repository "deb <http://archive.canonical.com/> lucid partner"   
- apt-get -qqy update   
- apt-get install lsof psmisc librmagick-ruby curl ruby-dev libmysql-ruby libmysqlclient-dev libpq-dev postgresql-client   
- apt-get install sun-java6-bin  
- apt-get install sun-java6-jdk   
- apt-get install sun-java6-jre   
- Install node with the following commands :   
cd /tmp   
export NODE\_VERSION=0.4.5   
wget <http://nodejs.org/dist/node-v>$NODE\_VERSION.tar.gz   
tar -zxf node-v$NODE\_VERSION.tar.gz   
cd node-v$NODE\_VERSION   
./configure   
make   
make install   
cd ..   
rm node-v$NODE\_VERSION.tar.gz   
rm -fr node-v$NODE\_VERSION   
- cd /home/coudfoundry   
- rake bundler:install   
- Modify /home/cloudfoundry/dea/config/dea.yml and change mbus to ip address of your cloud controller node and local\_router to ip address of your dea node :   
mbus: nats://10.30.1.4:4222   
local\_router : 10.30.1.3   
- Adapt path to ruby 1.8.7 and 1.9.2 executable path   
- cd /home/coudfoundry/bin/   
- ./vcap start

Step #7 : Deploy Cloud Foundry Services on Service node :   
- ssh service.vcap   
- copy vcap\_services.tar.gz to /home/cloudfoundry   
- tar -vzxf vcap\_services.tar.gz   
- apt-get install ruby-dev libmysql-ruby libmysqlclient-dev   
- apt-get install mysql-server   
- gem install mysql   
- Install Redis   
cd /tmp   
export REDIS\_VERSION=2.2.4   
wget --no-check-certificate -O redis\_master.tgz <http://redis.googlecode.com/files/redis->$REDIS\_VERSION.tar.gz   
ar -zxf redis\_master.tgz   
cd redis-$REDIS\_VERSION/src   
make   
cp redis-server /usr/bin   
cp redis-cli /usr/bin   
cd ../..   
rm redis\_master.tgz   
rm -fr redis-$REDIS\_VERSION   
- Install RabbitMQ   
export RABBITMQ\_VERSION=2.4.0   
export RABBITMQ\_VERSION\_FULL=2.4.0-1\_all   
wget <http://www.rabbitmq.com/releases/rabbitmq-server/v>$RABBITMQ\_VERSION/rabbitmq- server\_$RABBITMQ\_VERSION\_FULL.deb   
dpkg -i rabbitmq-server\_$RABBITMQ\_VERSION\_FULL.deb   
rm rabbitmq-server\_$RABBITMQ\_VERSION\_FULL.deb   
- Install MongoDB   
export MONGODB\_VERSION=1.8.1   
export mongodb="mongodb-linux-x86\_64-$MONGODB\_VERSION"   
wget <http://fastdl.mongodb.org/linux/>$mongodb.tgz   
tar -zxvf $mongodb.tgz   
cp $mongodb/bin/\* /usr/bin   
rm $mongodb.tgz   
rm -fr $mongodb   
- cd /home/cloudfoundry/   
- rake bundler:install   
- For each file on /home/cloudfoundry/services/[mongodb|mysql|redis|rabbitmq]/config/[mongodb|mysql|redis|rabbitmq]\_gateway.yml, change [mongodb|redis|rabbitmq|mysql]\_mbus, service\_mbus to ip address of your cloud controller. Also change cloud\_controller\_uri to your domain name installation   
mysql\_mbus: nats://10.30.1.4:4222   
service\_mbus: nats://10.30.1.4:4222   
cloud\_controller\_uri : api.passform.com   
- For each file on /home/cloudfoundry/services/[mongodb|mysql|redis|rabbitmq]/config/[mongodb|mysql|redis|rabbitmq]\_node.yml, change mbus to ip address of your cloud controller. Don't forget to change your MySQL root password on mysql\_node.yml. And also change cloud\_controller\_uri to your domain name installation   
cloud\_controller\_uri : api.passform.com   
mbus: nats://10.30.1.4:4222   
- cd /home/cloudfoundry/bin/   
- ./vcap start

You can also change memory allocation on services node to fit your system and think to verify your setup with ./vcap tail on all node.   
If your Cloud Foundry installation is very slow, check your /etc/hosts, add entries for all your node and restart all the components.

# BOSH



参见 “BOSH参考手册.docx”。

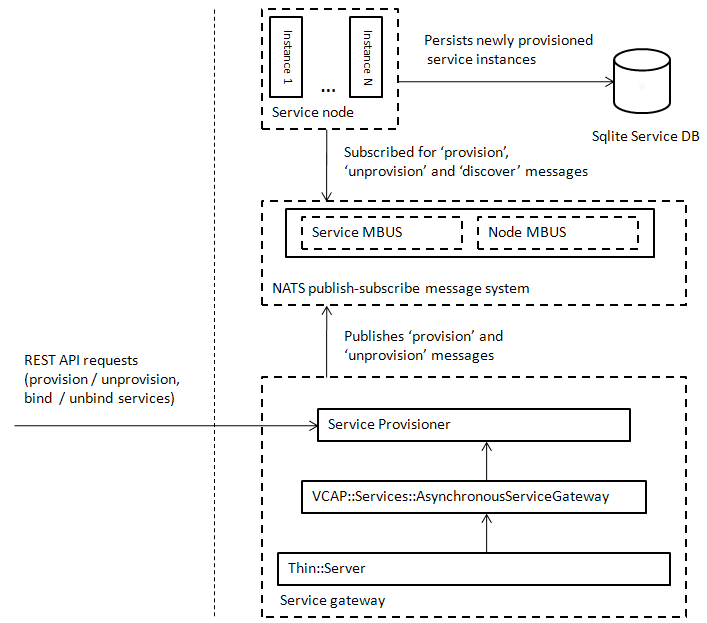
# CF新服务的建立：

## 说明：

1. 用Ruby基于CF 服务系统的结构写Node和Gateway，并用source进行联编。
2. org.cloudfoundry.services.EchoService

使用 new ServerSocket(port);来创建ServerSocket

## CF服务系统结构



## 步骤

### Services in Cloud Foundry

In Cloud Foundry you have two basic states of the services - System services and Provisioned services. System services are all types of services available to the system. Services of these types can be provisioned and bound to applications. When you provision a service you give it a name. This name is later used by applications to lookup metadata about the provisioned services. You can list both system and provisioned services by logging in to vmc and typing vmc services

### Adding the echo service to the system services

After completing this section our echo service will appear as a system service in the table printed by vmc services. This guide can be used for both single-machine and distributed setup of Cloud Foundry. **Note that by ellipsis (...) we mean the directory where your Cloud Foundry installation resides.** Here are the steps you need to execute:

1) On the file .../cloudfoundry/.deployments/devbox/config/vcap\_components.json add echo\_node and echo\_gateway to the list :

{"components":["health\_manager","cloud\_controller","redis\_node","redis\_gateway","neo4j\_node","mysql\_gateway","dea","router","mongodb\_gateway","mongodb\_node","neo4j\_gateway","mysql\_node","echo\_node","echo\_gateway"]}

2) Add service token configuration at .../cloudfoundry/.deployments/devbox/config/cloud\_controller.yml

# Services we provide, and their tokens. Avoids bootstrapping DB.  
 builtin\_services:  
 redis:  
 token: changeredistoken  
 mongodb:  
 token: changemongodbtoken  
 mysql:  
 token: changemysqltoken  
 neo4j:  
 token: changeneo4jtoken  
  **echo:**  
 **token: changeechotoken**

3) On the services host go to .../cloudfoundry/vcap/services/tools/misc/bin/nuke\_service.rb and add the path to the echo service configuration

default\_configs = {  
 :mongodb => File.expand\_path("../../mongodb/config/mongodb\_gateway.yml", \_\_FILE\_\_),  
 :redis => File.expand\_path("../../redis/config/redis\_gateway.yml", \_\_FILE\_\_),  
 :mysql => File.expand\_path("../../mysql/config/mysql\_gateway.yml", \_\_FILE\_\_),  
 :neo4j => File.expand\_path("../../neo4j/config/neo4j\_gateway.yml", \_\_FILE\_\_),  
 **:echo => File.expand\_path("../../echo/config/echo\_gateway.yml", \_\_FILE\_\_),**  
 }

4) Create start scripts for '\_node' and '\_gateway' on the services host at .../cloudfoundry/vcap/bin/services/

4.1) Content of 'echo\_node':

#!/usr/bin/env ruby  
 exec(File.expand\_path("../../../services/echo/bin/echo\_node", \_\_FILE\_\_), \*ARGV)

4.2) Content of 'echo\_gateway':

#!/usr/bin/env ruby  
 exec(File.expand\_path("../../../services/echo/bin/echo\_gateway", \_\_FILE\_\_), \*ARGV)

These scripts just delegate to the real scripts provided with the service.

4.3) Make them executable:

$ chmod +x .../cloudfoundry/vcap/bin/services/echo\_{node,gateway}

5) Add the echo service provisioner implementation on the services host  
Download and extract the implementation of the echo service provisioner from the attachments ([echo\_sp.zip](http://support.cloudfoundry.com/attachments/token/ijhmdt4oy2wygkh/?name=echo_sp.zip)) and copy the extracted directory

cp -r echo .../cloudfoundry/vcap/services/.

You should now have the **echo** directory service inside the services.

Copy the **.../cloudfoundry/vcap/services/echo/config/echo\_gateway.yml** and **.../cloudfoundry/vcap/services/echo/config/echo\_node.yml** files over to **../cloudfoundry/.deployments/devbox/config/**

Your echo\_gateway and echo\_node files should look like the following:

(一定要在echo\_gateway.yml文件中的service里加上必须的参数timeout)

.../cloudfoundry/.deployments/devbox/config/echo\_gateway.yml

---  
 cloud\_controller\_uri: api.vcap.me  
 service:

**timeout: 15**   
 name: echo  
 version: "1.0"  
 description: 'Echo key-value store service'  
 plans: ['free']  
 tags: ['echo', 'echo-1.0', 'key-value', 'echobased']  
 index: 0  
 token: **changeechotoken**  
 logging:  
 level: debug  
 mbus: nats://**nats:nats@<nats\_host>:<nats\_port>**  
 pid: /var/vcap/sys/run/echo\_service.pid   
 node\_timeout: 2

.../cloudfoundry/.deployments/devbox/config/echo\_node.yml

---  
local\_db: sqlite3:/var/vcap/services/echo/echo\_node.db  
mbus: nats://**nats:nats@<nats\_host>:<nats\_port>**   
index: 0  
base\_dir: /var/vcap/services/echo/  
ip\_route: <services\_host\_ip>  
logging:  
 level: debug  
pid: /var/vcap/sys/run/echo\_node.pid  
available\_memory: 4096  
node\_id: echo\_node\_1  
port: **<echo\_service\_port>** # port where echo service listens  
host: **<echo\_service\_host>** # host where echo service resides. May be different from services host

**Prefer using real IP addresses over localhost as some of these variables may become part of environment on other hosts!**

6) cd .../cloudfoundry/vcap/services/echo

7) source .../.cloudfoundry\_deployment\_profile && bundle package

8) Restart cloud controller and services node: Execute .../cloudfoundry/vcap/dev\_setup/bin/vcap\_dev restart . This should reveal "echo\_node" and "echo\_gateway" running. cd .../cloudfoundry/.deployments/devbox/log && tail -f \*.log will reveal the echo\_node and echo\_gateway logs.  
Now execute the command vmc services. Our new echo service should be available in the upper table. Congratulations! You have just provided your first Cloud Foundry service! Now, let's do something with it!

## 参考资料

1. How to Add a System Service to OSS Cloud Foundry - Step by Step guide CloudFoundry\_com Support
2. https://groups.google.com/a/cloudfoundry.org/forum/?fromgroups=#!searchin/vcap-dev/echo$20nats/vcap-dev/nfw-6oP37rs/iRCRq1iW9\_kJ

## 备注：

**1. 以DHCP方式配置网卡**

编辑文件/etc/network/interfaces:  
sudo vi /etc/network/interfaces

并用下面的行来替换有关eth0的行:  
# The primary network interface - use DHCP to find our address  
auto eth0  
iface eth0 inet dhcp

用下面的命令使网络设置生效:  
sudo /etc/init.d/networking restart  
也可以在命令行下直接输入下面的命令来获取地址

sudo dhclient eth0  
 **2. 为网卡配置静态IP地址**

编辑文件/etc/network/interfaces:

sudo vi /etc/network/interfaces

并用下面的行来替换有关eth0的行:# The primary network interface  
auto eth0  
iface eth0 inet static  
address 192.168.3.90  
gateway 192.168.3.1  
netmask 255.255.255.0  
#network 192.168.3.0  
#broadcast 192.168.3.255

将上面的ip地址等信息换成你自己就可以了.用下面的命令使网络设置生效:  
sudo /etc/init.d/networking restart

二、配置dns服务器

         ubuntu 的dns服务器信息，放在 /etc/resolv.conf中,

         添加dns服务器地址，如202.112.125.53,则在上述文件中加入

             nameserver  202.112.125.53

# 实现机制分析

## 架构图





## CloudFoundry启动过程

分析Ruby代码时，注意Ruby特殊语法。

运行如下命令：vcap-master\dev\_setup\bin\vcap\_dev start

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

### Router启动

获取IP参数

|  |
| --- |
| Router.inet = inet || VCAP.local\_ip(config['local\_route'])  Router.port = port |

启动Thin server

|  |
| --- |
| *# TCP/IP Socket*  Router.server = Thin::Server.new(inet, port, RouterULSServer, :signals => false) if inet && port  Router.local\_server = Thin::Server.new(fn, RouterULSServer, :signals => false) if fn  Router.server.start if Router.server  Router.local\_server.start if Router.local\_server |

连接NATS

|  |
| --- |
| NATS.start(:uri => config['mbus']) |

订阅消息

|  |
| --- |
| Router.setup\_listeners  def setup\_listeners  NATS.subscribe('router.register') { |msg|  msg\_hash = Yajl::Parser.parse(msg, :symbolize\_keys => true)  return unless uris = msg\_hash[:uris]  uris.each { |uri| register\_droplet(uri, msg\_hash[:host], msg\_hash[:port],  msg\_hash[:tags], msg\_hash[:app]) }  }  NATS.subscribe('router.unregister') { |msg|  msg\_hash = Yajl::Parser.parse(msg, :symbolize\_keys => true)  return unless uris = msg\_hash[:uris]  uris.each { |uri| unregister\_droplet(uri, msg\_hash[:host], msg\_hash[:port]) }  }  end |

向系统注册Router自己

|  |
| --- |
| *# Register ourselves with the system*  status\_config = config['status'] || {}  VCAP::Component.register |

设置vars和sweeper

|  |
| --- |
| *# Setup some of our varzs..*  VCAP::Component.varz  *# This will check on the state of the registered urls, do maintenance, etc..*  Router.setup\_sweepers |

定期发布一个Router已启动的消息

|  |
| --- |
| NATS.publish('router.start', @hello\_message) |

### CloudController启动

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

设置目录

|  |
| --- |
| tmp\_dir = CloudController.tmp\_dir  uploads\_dir = CloudController.uploads\_dir |

### DEA启动

Droplet Execution Agency

dea.rb 中生成 agent 对象，调用agent run方法进行初始化。agent.rb中run方法处理内容如下：

初始化状态

订阅消息

|  |
| --- |
| *# Setup our listeners..*  NATS.subscribe('dea.status') { |msg, reply| process\_dea\_status(msg, reply) }  NATS.subscribe('droplet.status') { |msg, reply| process\_droplet\_status(msg, reply) }  NATS.subscribe('dea.discover') { |msg, reply| process\_dea\_discover(msg, reply) }  NATS.subscribe('dea.find.droplet') { |msg, reply| process\_dea\_find\_droplet(msg, reply) }  NATS.subscribe('dea.update') { |msg| process\_dea\_update(msg) }  NATS.subscribe('dea.stop') { |msg| process\_dea\_stop(msg) }  NATS.subscribe("dea.#{uuid}.start") { |msg| process\_dea\_start(msg) }  NATS.subscribe('router.start') { |msg| process\_router\_start(msg) }  NATS.subscribe('healthmanager.start') { |msg| process\_healthmanager\_start(msg) }  NATS.subscribe('dea.locate') { |msg| process\_dea\_locate(msg) } |

恢复已存在应用的状态

|  |
| --- |
| *# Recover existing application state.*  recover\_existing\_droplets  delete\_untracked\_instance\_dirs |

定期发送状态消息

|  |
| --- |
| EM.add\_periodic\_timer(@heartbeat\_interval) { send\_heartbeat }  EM.add\_periodic\_timer(@advertise\_interval) { send\_advertise }  EM.add\_timer(MONITOR\_INTERVAL) { monitor\_apps }  EM.add\_periodic\_timer(CRASHES\_REAPER\_INTERVAL) { crashes\_reaper }  EM.add\_periodic\_timer(VARZ\_UPDATE\_INTERVAL) { snapshot\_varz }  EM.add\_periodic\_timer(DROPLET\_FS\_PERCENT\_USED\_UPDATE\_INTERVAL) { update\_droplet\_fs\_usage } |

发布已启动DEA的消息

|  |
| --- |
| NATS.publish('dea.start', @hello\_message\_json)  send\_advertise |

### ruby调试

gem install ruby-debug-ide --platform=ruby --pre

## 应用发布过程

以vmc push为例

1.vmc 会调用lib\mothership.rb的start方法。

mothership.rb位于D:\reference\PAAS\tools\Ruby193\lib\ruby\gems\1.9.1\gems\mothership-0.3.5\lib

4.2. 触发lib\vmc\cli\app\push.rb里upload\_app方法

|  |
| --- |
| def upload\_app(app, path)  with\_progress("Uploading #{c(app.name, :name)}") do  app.upload(path) //触发包含在App里的upload\_helpers.rb upload方法  end  rescue  err "Upload failed. Try again with 'vmc push'."  raise  end |

4.2.1触发包含在App里的upload\_helpers.rb upload方法

D:\reference\PAAS\tools\Ruby193\lib\ruby\gems\1.9.1\gems\cfoundry-0.4.19\lib\cfoundry**\**

**upload\_helpers.rb 🡪**upload

|  |
| --- |
| @client.base.upload\_app(@guid, packed && zipfile, resources || []) |

**v1\base.rb 🡪 upload\_app ；该方法会调用父类baseclient.rb post方法**

## 调用应用过程

## 应用查询过程

vmc apps

代码根目录：D:/reference/PAAS/tools/Ruby193/lib/ruby/gems/1.9.1/gems/vmc-0.3.18/

经过bin\vmc、lib\cli\runner.rb的引导，进入lib\cli\commands\apps.rb程序文件中，由于在apps.rb中的alias :apps :list， 使apps方法与list方法具有同样功能，apps方法直接指向list方法。

apps.rb 文件中的list方法逻辑：

先调用lib\cli\commands\base.rb的client(cli=nil)方法获取client对象（该对象的定义位于lib\vmc\client.rb，用来操作http请求），然后调用client.rb的apps方法。

client.rb的apps方法

json\_get(url) 🡪http\_get(path, content\_type=nil)

## 服务发布过程

## 应用实例的manifest.yml

## 模块结构

CloudFoundry使用Ruby语言进行编写和实现。

### NATS：

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

EventMachine is an event-driven I/O and lightweight concurrency library for Ruby. It provides event-driven I/O using the [Reactor pattern](http://en.wikipedia.org/wiki/Reactor_pattern), much like [JBoss Netty](http://www.jboss.org/netty), [Apache MINA](http://mina.apache.org/), Python's [Twisted](http://twistedmatrix.com), [Node.js](http://nodejs.org), libevent and libev.

EventMachine is designed to simultaneously meet two key needs:

* Extremely high scalability, performance and stability for the most demanding production environments.
* An API that eliminates the complexities of high-performance threaded network programming, allowing engineers to concentrate on their application logic.

This unique combination makes EventMachine a premier choice for designers of critical networked applications, including Web servers and proxies, email and IM production systems, authentication/authorization processors, and many more.

EventMachine has been around since the early 2000s and is a mature and battle tested library.

## MCF目录结构

/var/vcap/sys/log：存放日志信息

/var/vcap/data/packages/：存放运行CloudFoundry的源代码

/var/vcap/data/jobs/：

/var/vcap/data/jobs/micro/83.1-dev/templates/cloud\_controller

/var/vcap/data/jobs/micro/83.1-dev/packages： 目录链接，指向/var/vcap/data/packages/

/var/vcap/packages/cloud\_controller：目录链接，指向/var/vcap/data/packages

## VMC

通过VMC进行的操作实际上到最后都是CC在处理。VMC 与 CC 连接的桥梁：[Gem rest\_client](https://github.com/adamwiggins/rest-client)

# 常见问题：

windows下安装或升级时gem经常会碰到

|  |  |  |
| --- | --- | --- |
| 1 | Please update your PATH to include build tools or download the DevKit | |
| 2 | from '<http://rubyinstaller.org/downloads>' and follow the instructions |

|  |  |
| --- | --- |
| 3 | at '<http://github.com/oneclick/rubyinstaller/wiki/Development-Kit>' |

通过安装DevKit解决。DevKit 是windows平台下编译和使用本地C/C++扩展包的工具。它就是用来模拟Linux平台下的make, gcc, sh来进行编译。注：这个方法目前仅支持通过RubyInstaller安装的Ruby，如果不是建议你重新安装。安装方法：双击下载的7z文件，指定解压路径，路径中不能有空格。如C:\DevKit，这个路径就是<DEVKIT\_INSTALL\_DIR>。

参见：https://github.com/oneclick/rubyinstaller/wiki/development-kit

连接MCF数据库服务

可以通过 Cloud Foundry，借助[**Caldecott**](http://en.wikipedia.org/wiki/Caldecott_Tunnel)通过本地端口打开通向任意 Cloud Foundry 数据服务的通道了。您可以使用熟悉的客户端应用程序，直接分析、操作或传输绑定到 Cloud Foundry 应用程序的数据服务中包含的数据。[**Caldecott**](http://en.wikipedia.org/wiki/Caldecott_Tunnel)的底层是一个简单的 Ruby 应用程序。

1. gem install eventmachine --pre
2. 如果vmc是在windows操作系统下操作，需要下载安装DevKit

2.1下载：<http://cdn.rubyinstaller.org/archives/devkits/DevKit-tdm-32-4.5.2-20110712-1620-sfx.exe>

2.2 解压DevKit

2.3 以命令行方式进入DevKit根目录

> cd <DEVKIT\_INSTALL\_DIR>

> ruby dk.rb init

> ruby dk.rb review

> ruby dk.rb install

> gem install rdiscount --platform=ruby #验证DevKit是否安装成功。

1. gem install caldecott
2. gem install tunnel-vmc-plugin
3. vmc services #查看服务
4. vmc tunnel <service name> #连接服务

使用dev\_setup进行单节点Cloud Foundry安装

sudo vi /ect/hosts

## 问题：

用离线包来部署cloudfoundry；一个节点部署cloudfoundry的多个模块；

设置DNS；

设置loadbalance

研究一下vmc发布程序过程，为下一步使用界面形式发布程序做准备

多节点的cloudfoundry如何公布出服务信息；

vmc如何获取这些cloudfoundry上的服务信息；

---------------------

1. 作为消息中心的NATS在稳定性、容错、灾难恢复上的机制
2. 如何在发布Java Web应用时，可以同时指定JVM的系统参数、内存分配参数
3. 在Cloudfoundry加入统一用户验证功能（是否可以通过UAA、系统服务、发布的应用等几种方式）
4. 如何将Memcached作为系统服务添加到系统中
5. .Net应用在cloudfoundry上的发布