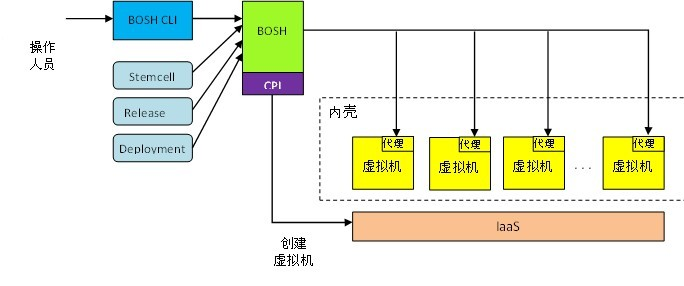
**BOSH 参考**手册

# 版本变更

|  |  |  |
| --- | --- | --- |
| 变更内容 | 时间 | 操作者 |
|  |  |  |

# BOSH介绍



## BOSH创建release过程

1. 解析config/blobs.yml文件，下载指定的文件到.blobs中（文件名以blobs.yml文件中的sha的值命名），并在blobs文件夹里对应的项目下建此文件链接，该连接指向.blobs中的对应文件（链接名正常文件名）
2. 把package的spec文件中的files指向的所有文件拷贝到临时目录中（分别从blobs目录和src目录下查找对应的文件进行拷贝），同时把packaging和pre\_packaging（若有）也一同拷到临时目录
3. 执行package的pre\_packaging脚本(若有)。在pre\_packaging脚本中一般会执行pkg\_utils/vender.sh \*\*\*脚本，vender.sh会执行两个ruby脚本transform\_git\_source和fetch\_gems，transform\_git\_source脚本会下载依赖的git源码并打包成gem，修改gemfile和gemfile.lock文件的内容。fetch\_gems脚本用于下载gem包，下载的gem包拷到临时目录下的vender/cache目录下，同时在.cache/fetch\_gems下放一个备份，需要下载的gem包在src/\*\*\*/Gemfile.lock中指定，如果在.cache中已存在此gem包，则直接使用cache中的。
4. 把临时目录打包成由具体的package所生成的packageName.tgz包
5. jobs下的文件在执行完prepare脚本后直接生成tgz包，生成时spec文件重命名为job.MF
6. 所有的packages和jobs生成最终的releaseName-version.yml文件，里面包含了package和job的名字，版本，指纹，sha等信息，同时包含了release的name和版本等信息
7. releaseName-version.yml和所有的packages、jobs打包成releaseName-verion.tgz,其中releaseName-version.yml重命名为release.MF

## deploy过程

1. 执行相应packages下tgz包中的packaging脚本
2. 执行jobs模板文件中的属性替换并把生成的文件放到具体的位置，具体位置由job的spec文件的templates部分的映射指定
3. 启动服务

## BOSH release的目录结构

.blobs ----打包时下载的各种二进制包的存放目录

.cache----打包时下载的gem包的存放目录

.dev\_builds----生成dev版本时各种tgz文件的临时存放目录

.final\_builds----生成final版本时各种tgz文件的临时存放目录

Blobs---大型源代码捆绑包，里面定义了各种包的链接，指向.blobs中的具体文件

Dir/file

Config---发行版配置文件

Blobs.yml定义了需要的各种源代码包

Jobs----jobs定义，job是package的实现，即运行一个或多个进程

Prepare脚本----在job打包前执行

Templates---各种作业模板

Spec---job规范文件

Name—job名

Templates—模板文件与最终位置的映射关系

Packages—依赖的package

Monit---监视文件，通常包含一段脚本，可通过调用此脚本来启动/停止该进程

with pidfile—该进程将其pid文件存放在何处

start program—monit 应如何启动该进程

stop program—monit 应如何停止该进程

Packages----package定义

Pre\_packaging—创建release时执行，

Spec—最终包中包含的文件列表

Name—包的名称

Dependencies—可选，该包依赖的其他包列表

Files—该包包含的文件列表，\* 与任何文件匹配，\*\*递归方式匹配目录，具体定义的文件从blobs或src下查找

Packaging—部署时执行的脚本，完成package的部署

Releases---最终发行版，包含已经创建好的final发行版，可以直接使用，避免自己创建release

Src----package的源代码

## 服务管理

为了获取CF中的服务信息，需要发布一个service gateway节点，CC通过service gateway获取服务信息。

## 各功能的流程

### 用户

vmc register的用户会增加到ccdb中。

uaac user add新增加的用户会出现uaadb中。

### 监控

There are 2 components which put metrics into the tsdb, one is collector, the other is bosh health\_monitor. Collector collects metrics based on the varz mechanism. Bosh health\_monitor collects metrics based on the heartbeat message.  
  
The general philosophy is that, collector is responsible for the cf component metrics, while bosh health\_monitor is designed to collect os-level metrics, such as load average, memory usage, IO stats.  
  
For collector, please refer to <https://github.com/cloudfoundry/vcap-tools/tree/master/collector>.  
For bosh health\_monitor, please refer to <https://github.com/cloudfoundry/bosh/tree/master/health_monitor>.

The colloector collects the various metrics fropm varz and pushes to the tsdb. The dashboard service gets the data from the tsdb is it?

调试：

curl <http://192.168.2.86:4242>

curlhttp://192.168.2.86:4242/q?start=2013/05/07-16:10:00&end=2013/05/07-17:00:00&m=sum:frameworks.running\_apps&ascii

*TSDB client*  
tsdb query <start timestamp> sum rate frameworks.running\_apps  
  
timestamp, such as 2012/04/13-09:10:00  
  
*HTTP client*  
curl http://<host ip>:<port>/q?start=<timestamp>\&end=<timestamp>\&m=sum:frameworks.running\_apps\&ascii

# 资源说明

## IP分配

192.168

2-10：vSphere 物理机，ssh:root/password

11-20：

11, (BOSH CLI, gateway: user:boshcli/boshcli);

12, nfs-server (user:nfsserver/nfsserver)；

21 : Micro BOSH (director 25555)

31-40： BOSH (6)

192.168.1.51 - 192.168.1.110： cf

103：vcenter，administrator，dz+1

无线路由器

IP: 10.68.199.5

子网掩码：255.255.255.0

网关：10.68.199.253

DNS： 10.67.12.130

10.68.199.5 用户：administrator 密码：pass2013/

## 虚拟机资源

ssh登陆micro bosh（bosh director 25555）的用户和密码：vcap/c1oudc0w

## 硬件资源

3台台式机：12G内存，500G硬盘，Intel i5-2400 3.1 GHz

2台台式机：10G内存，500G硬盘，Intel i5-2400 3.1 GHz

## 部署内容

部署BOSH：192.168.2.11，boshcli用户主目录下 bosh\_manifest.yml

部署cloudfoundry：192.168.2.11，boshcli用户主目录下cf-small.yml

cloudfoudry：

offline of code, 192.168.2.11,/home/boshcli/tempestL/tempest/cf-release/dev\_releases/

BOSH: code,192.168.2.11, /home/boshcli/bosh/release

.net win2008: password: pass2013/ pass/2013

# 部署步骤

## 设置NFS server

设置NFS时，需要将一个Ubuntu设置为一个NFS server。可以将NFS server设在SH CLI所在的Ubuntu虚拟机操作系统上。通过增加Ubuntu的硬盘，扩充NFS的容量。参见附录“设置NFS“。

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

## IAAS准备

按照cloudfoundry的“准备IaaS环境“文档进行设置。

## 设置BOSH CLI

在ubuntu 10.04上安装BOSH CLI （可以是物理机或虚拟机）

### 设置ubuntu 的ip、DNS

编辑文件/etc/network/interfaces:

|  |
| --- |
| $ sudo vi /etc/network/interfaces |

内容如下：

|  |
| --- |
| auto eth0 iface eth0 inet static address 192.168.2.15 gateway 192.168.2.1 netmask 255.255.255.0 #network 192.168.2.0 #broadcast 192.168.2.255 |

使网络设置生效:

|  |
| --- |
| $ sudo /etc/init.d/networking restart |

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 192.168.2.1  nameserver  202.112.125.53 |

可以考虑将BOSH CLI所在的虚拟机设置为网关，构建一个局域网环境。网关设置参见“附录🡪BOSH CLI的网络”。

### 安装openssh

安装openssh是为了方便远程操作虚拟机操作系统。

|  |
| --- |
| $sudo apt-get install openssh-server |

如果不成功，先执行sudo apt-get install update 更新源或者增加镜像。

### 安装BOSH CLI

我们现在开始在该虚拟机上安装 BOSH CLI。为此，请登录到 Ubuntu，然后遵照以下步骤操作。（请注意，这些步骤大都摘自 BOSH 官方文档。为了您方便起见，在此将它们列出来。）

#### 安装前更新Ubuntu的apt-get：

|  |  |
| --- | --- |
| $ sudo apt-get upgrade | #(takes long time, 200M+ files to download, optional for bosh install) |
| $ sudo apt-get dist-upgrade | #(200M+ to download, update the kernel and image, optional) |
| $ sudo reboot |  |

#### 安装Ruby

安装BOSH CLI需要完整的Ruby环境，所以接下来我们分别介绍如何使用rbenv和rvm这两种Ruby版本管理器来安装Ruby和BOSH CLI。其中使用rvm安装的方法相对简单且易于维护，建议大家使用。参见附录“安装Ruby“

#### 安装 BOSH CLI

## 安装bosh\_deployer

在 BOSH CLI 虚拟机中，安装 BOSH 部署器bosh\_deployer，这一部分大约需要5分钟。

|  |
| --- |
| $ gem install bosh\_deployer |

根据提示安装其他依赖组件。

### Q&A:

如果安装过程中有异常信息；或安装后，输入bosh help命令无法显示micro的子命令；可以尝试下面步骤：

|  |
| --- |
| gem install net-ssh -v 2.2.2  gem install net-scp -v 1.0.4  gem install fog -v "~> 1.9.0"  gem install bosh\_deployer |

## 设置Micro BOSH

在192.168.2.15上进行Micro BOSH的安装操作。

将Micro BOSH视作袖珍版的 BOSH。尽管标准的 BOSH各个组件分布在 6 个虚拟机上，但Micro BOSH却恰恰相反，它在单个虚拟机中包含了所有组件。它可以轻易的设置，通常用于部署小型的Release，如 BOSH。

### 创建文件夹

创建用来存放虚拟机和模板的文件夹

在 vCenter中的“主页”(Home) ->“清单”(Inventory) ->“虚拟机和模板”(VMs and Templates) 视图下，确保用来存放虚拟机和模板的文件夹已经创建（见“IAAS准备”章节）。这些文件夹将在部署配置中使用。

创建boshdeployer

在vCenter中，从“主页”(Home) ->“清单”(Inventory) ->“数据存储”(Datastores) 视图中，选择我们创建的NFSdatastore数据存储，并浏览该存储。

右键单击根文件夹，创建一个用来存储虚拟机的子文件夹。在本例中，我们将此子文件夹命名为“boshdeployer”。此文件夹名称将成为我们部署清单中的“disk\_path”参数的值。

### 下载公共 stemcell

$ mkdir -p ~/stemcells

$ cd stemcells

$ bosh public stemcells

根据输出列表选择适当的Micro BOSH stemcell：

使用下面的命令下载Micro BOSH的 stemcell：（耗时较长）

$ bosh download public stemcell micro-bosh-stemcell-vsphere-0.6.4.tgz

### 设置部署文件

**建立目录**

|  |
| --- |
| $ cd ~  $ mkdir deployments  $ cd deployments  $ mkdir micro01 |

**创建部署 (.yml)文件**

将创建好的yml文件保存在micro1文件夹下，该文件夹名称与 .yml文件中的name属性值保持一致。

**yml文件说明**

文件内容示例参见“附录🡪micro\_bosh.yml”。

在 yml 文件中，有一节内容是关于 vCenter 的。请在此节中输入我们在“设置Micro BOSH🡪创建文件夹“章节中创建的文件夹的名称。“disk\_path”应为我们刚刚在数据存储 (NFSdatastore) 中创建的文件夹。datastore\_pattern 和persistent\_datastore\_pattern 的值是共享数据存储的名称 (NFSdatastore)。如果您采用本地磁盘，此值可以是像“localstore\*”这样的通配字符串。

|  |
| --- |
| datacenters:  - name:vDataCenter  vm\_folder:vm\_folder  template\_folder:template  disk\_path:boshdeployer  datastore\_pattern:NFSdatastore  persistent\_datastore\_pattern:NFSdatastore allow\_mixed\_datastores:true |

下面是Micro BOSH的一个示例yml文件的链接：

<https://github.com/vmware-china-se/bosh_doc/blob/master/micro.yml>

#### Q&A

micro.yml文件说明

|  |
| --- |
| 1.文件内容中不要出现 tab 的缩进符，否则会影响部署。  2.micro bosh的ip地址不要设成和micro cli一样的IP地址，否则会出现wait agent问题。  3.在micro.yml文件中设置vcenter ip地址时，有时候要加上9443端口号，否则会连不上。  4.micro.yml配置文件中的clusters参数值一定要加上。 |

### Micro BOSH部署

注意：所有 bosh micro 命令必须先cd到这个Micro BOSH的部署目录中才能执行。

|  |
| --- |
| $ cd deployments  $ bosh micro deployment micro01  Deployment set to “~/deployments/micro01/micro\_bosh.yml”  $ bosh micro deploy ~/stemcells/micro-bosh-stemcell-vsphere-0.6.4.tgz |

如果一切都运行顺利，Micro BOSH将在几分钟内部署完毕。您可以通过下面的命令查看部署状态：

|  |
| --- |
| $ bosh micro deployments |

您将会看到您的Micro BOSH部署已列出。

#### Q&A

1.部署时会创建虚拟机。如果中途失败，在vCenter删掉虚拟机，并删掉deployments目录中的deploy文件，然后重新运行bosh micro deploy…命令。

2.在执行bosh deploy后提示Deployment name not found in the deployment manifest

---用16进制模式打开部署文件，开头有3个字节的不可见字符EF BB BF，删除后重新部署

## 安装多节点的BOSH

在192.168.2.11上进行多节点BOSH的安装。stemcell、release使用tempestL/tempest/目录里的内容。

Micro BOSH准备就绪后，我们就可用它来部署 BOSH。BOSH 是一个包含 6 个虚机的分布式系统。正如上一节所提到的那样，我们需要有三项内容：一个作为虚拟机模板的stemcell、一个作为待部署软件的 BOSH Release，以及一个用来定义部署配置的部署清单文件。我们来逐一准备。

### stemcell

1)首先，我们将 BOSH CLI 的目标设为Micro BOSH的Director。可以将 BOSH Director视作 BOSH 的控制者或协调者。所有 BOSH CLI 命令均发往该Director加以执行。该Director的 IP 地址在我们用来创建Micro BOSH的 yml 文件中定义。BOSH Director的默认用户/密码为 admin/admin。在我们的示例中，我们使用下面的命令来设定Micro BOSH的目标和进行身份验证：

|  |
| --- |
| $ bosh target 192.168.1.21:25555  $ bosh login |

2)接下来，我们下载 BOSH stemcell并将其上传到Micro BOSH。这一步与下载Micro BOSH的 stemcell 类似。唯一的差别在于，我们选择的是 BOSH 而非Micro BOSH的stemcell。

|  |
| --- |
| $ cd ~/stemcells  $ bosh public stemcells |

随即便会显示一列stemcell；请选择最新的stemcell（0.6.7）进行下载和上传：

|  |
| --- |
| $ bosh download public stemcell bosh-stemcell-vsphere-0.6.7.tgz  $ bosh upload stemcell bosh-stemcell-vsphere-0.6.7.tgz |

### release

实验环境中的bosh release代码在192.168.2.15用户主目录下的boshdeploy/bosh目录中。

#### 设置git账户

4)设置您的 ssh 公钥（接受所有默认值）

|  |
| --- |
| $ ssh-keygen -t rsa |

5)将您的密钥内容从 ~/.ssh/id\_rsa.pub 完整复制到您的Git帐户配置的SSH public key部分中

6)设置您的姓名和电子邮件

|  |
| --- |
| $ git config --global user.name FirstnameLastname  $ git config --global user.email your\_email@youremail.com |

7)如果前面已经安装好了git，在终端输入git -v检查是否能打印出版本号

8)使用Git从 Cloud Foundry 代码库中克隆Release代码。以下命令分别获取 BOSH 和 Cloud Foundry 的代码。

|  |
| --- |
| $ git clone https://github.com/cloudfoundry/bosh.git  $ git clone https://github.com/cloudfoundry/cf-release.git |

#### 创建 BOSH Release

|  |
| --- |
| $ cd bosh/release  $ git submodule update –-init --recursive  $ bosh create release --with-tarball |

下载所有依赖包，并把整个BOSH打包成一个tar包

这个过程需要自己输入deployment的名字，BOSH会为最后的tar包和yml生成形如x.x-dev这样的版本号来命名。如果存在本地代码冲突，您可以添加“–force”选项：

|  |
| --- |
| $ bosh create release --with-tarball --force |

这里会要求您输入deployment的名字，您可以自己定义一个易于识别的名字比如bosh1。

这一步可能需要一些时间才能完成，具体取决于您的网络速度。它首先会从一个 Blob 服务器下载二进制文件。然后它会构建包并生成清单文件。该命令的输出大致如下：

|  |
| --- |
| Syncing blobs…  ...  Building DEV release  Please enter development release name:bosh-dev1  ---------------------------------  Building packages  …  Generating manifest...  …  Copying jobs... |

最后，Release创建完毕后，您将看到大致如下的内容。请注意，最后两行指出了清单文件和Release文件。

|  |
| --- |
| Generated /home/boshcli/bosh-release/dev\_releases/bosh-dev1-6.1-dev.tgz  Release summary  ……  Release version:6.1-dev  Release manifest:/home/boshcli/bosh-release/dev\_releases/bosh-dev1-6.1-dev.yml  Release tarball (88.8M):/home/boshcli/bosh-release/dev\_releases/bosh-dev1-6.1-dev.tgz |

最后两句就是您create release生成出的产物。

#### 上传release

9)将创建好的Release tar包上传到Micro BOSH的director。

|  |
| --- |
| $ bosh upload release dev\_releases/bosh-dev1-6.1-dev.tgz |

使用bosh releases就可以看到我们上传好的releases。

#### Q&A

**解决创建BOSH release时，对“需要BOSH CLI 1.5.0”的提示提供的解决方案1：**

修改bosh/release/config/final.yml文件中的min\_cli\_version属性值，将其值改为1.0.3

**解决创建BOSH release时，对“需要BOSH CLI 1.5.0”的提示提供的解决方案2：**

you need to run bosh from the bosh repo until we release a newbosh\_cli gem, e.g.临时运行如下命令：

|  |
| --- |
| cd bosh  bundle install --local  bundle exec bosh ... |

**创建release的方案2：**

说明：这一步可以不创建release，通过上传bosh达到下一步部署BOSH目的。

|  |
| --- |
| $ bosh upload release releases/bosh-11.yml |

该命令在tempestL/tempest/bosh-release/ 或bosh/release目录下操作。

**执行bosh upload release时，release在上传完后提示 Error 100：execute expire**

----此问题是由于网络原因导致director没法访问网络，把网络配置正确或等网络空闲后重新上传

**执行create release时，如果出现下载gem失败导致不能创建release**，

需要找到失败的gem包，然后从.cache/getch\_gems中删除此gem包再重新执行create release，如果遇到多次执行都不能下载的包，自己去rubygems.org网上下载对应的版本，然后拷到.cache/fetch\_gems下再执行

### BOSH 部署清单

10)配置 BOSH 部署清单。

我们至少需要在这个配置文件中更改bosh director的UUID，release的名称和版本

首先，我们通过执行以下命令获取并记录下该director的 UUID 信息：

|  |
| --- |
| $ bosh status  Updating director data... done  Target micro01 (http://10.60.98.124:25555) Ver: 0.4 (00000000)  UUID 7d72eb71-9a98-4081-9857-ad7c7ff4ee33  User admin  Deployment /home/boshcli/bosh-dev1.yml |

利用bosh status命令结果里的UUID值替换部署文件里的director\_uuid对应的值。

在NFSdatastore数据存储中创建一个用来存储虚拟机的子文件夹（名字是boshdeployer\_bosh）。

现在我们将进入此安装过程中最为复杂的环节：修改部署清单文件。由于大多数 BOSH 部署错误都是因为清单文件中的设置不正确而造成的，因此我们详细地对此环节进行说明。

首先，我们从以下位置获取清单模板：<https://github.com/cloudfoundry/oss-docs/blob/master/bosh/tutorial/examples/bosh_manifest.yml>

由于 BOSH 官方文档提供了该清单文件的规范，因此我们假定您在阅读本篇文章前，已经通读了此文档。我们将不会对该文件进行全面的介绍；而是讨论该清单文件中的一些重要项目。

清单文件的条目内容介绍参见“附录🡪bosh清单条目介绍”

清单文件内容示例参见“附录🡪bosh\_manifest.yml”

#### 组件配置清单

|  |  |  |
| --- | --- | --- |
| **作业** | **资源池** | **IP** |
| nats | small | 192.168.2.31 |
| postgres | small | 192.168.2.32 |
| redis | small | 192.168.2.33 |
| director | director | 192.168.2.34 |
| blob\_store | small | 192.168.2.35 |
| health\_monitor | small | 192.168.2.36 |

#### 实际部署

安装过程中保持网络稳定。

11)更新完部署清单文件后，我们便可以通过运行以下命令开始实际部署：

|  |
| --- |
| $ bosh deployment bosh\_manifest.yml  $ bosh deploy |

此过程可能需要等待一段时间才能完成，具体取决于您的网络条件以及可用的硬件资源。您也可以在 vCenter 控制台查看创建、配置并销毁虚拟机的过程。操作信息输出如下：

|  |
| --- |
| Preparing deployment  …..  Compiling packages  ……  Binding instance VMs  postgres/0 (00:00:01)  director/0 (00:00:01)  redis/0 (00:00:01)  blobstore/0 (00:00:01)  nats/0 (00:00:01)  health\_monitor/0 (00:00:01)  Done 6/6 00:00:01  Updating job nats  nats/0 (canary) (00:01:14)  Done 1/1 00:01:14  ……  Updating job director  director/0 (canary) (00:01:10)  Done 1/1 00:01:10  …… |

如果一切运行顺利，您最终将会看到大致如下的结果：

|  |
| --- |
| Task 14 done  Started 2012-08-12 03:32:24 UTC  Finished 2012-08-12 03:52:24 UTC  Duration 00:20:00  Deployed `bosh-dev1.yml` to `micro01` |

这表示您已成功部署 BOSH。您可以通过执行下面的命令来查看您的部署：

$ bosh deployments

您可以通过执行下面的命令来查看所有虚拟机的状态：

$ bosh vms

如果一切都未出问题的话，您将看到大致如下的虚拟机状态：

|  |
| --- |
| +------------------+---------+---------------+--------------+  | Job/index | State | Resource Pool | IPs |  +------------------+---------+---------------+--------------+  | blobstore/0 | running | medium | 10.60.98.116 |  | director/0 | running | director | 10.60.98.117 |  | health\_monitor/0 | running | small | 10.60.98.115 |  | nats/0 | running | small | 10.60.98.120 |  | postgres/0 | running | medium | 10.60.98.119 |  | redis/0 | running | small | 10.60.98.118 |  +------------------+---------+---------------+--------------+  VMs total: 6 |

#### Q&A

1.如果多节点的BOSH安装失败，先删除已存在的deployment

查看已安装的deployment

bosh deployments

删除deployment

bosh delete deployment <已部署的bosh 名称> --force

2.在执行bosh deploy后提示Deployment name not found in the deployment manifest

---用16进制模式打开部署文件，开头有3个字节的不可见字符EF BB BF，删除后重新部署

3.Error 400007: ‘job/index’ is not running after update

在执行bosh deploy时偶尔会出现这个错误，重新执行即可

## 部署Cloud Foundry

在192.168.2.11上进行多节点CloudFoundry的安装。stemcell、release使用tempestL/tempest/目录里的内容。

### stemcell

设置target

|  |
| --- |
| $ bosh target 192.168.2.34  $ bosh login |

将BOSH stemcell上传到BOSH director(多节点下的BOSH，ip 192.168.2.34)，使用与多节点的BOSH相同的bosh stemcell：

|  |
| --- |
| $ bosh upload stemcell ~/tempestL/tempest/stemcells/bosh-stemcell-vsphere-0.6.7.tgz |

### release

实验环境中的cloudfoundry release代码在192.168.2.15用户主目录下的deployments/cf-release目录中。

#### 创建cloudfoundry release

如果需要在无SSL情况下也可以使用Cloudfoundry，需要修改几处文件：参见“附录🡪无SSL证书的设置”

前提条件：你已设好git的公共密钥。参见“安装BOSH🡪release🡪设置git账户”

|  |
| --- |
| # 获取较为稳定的版本  $ git clone -b release-candidate <https://github.com/cloudfoundry/cf-release.git>  $ cd cf-release/release  $ git submodule update –-init --recursive  $ bosh create release --with-tarball |

下载所有依赖包，并把整个BOSH打包成一个tar包。这个过程需要自己输入deployment的名字，BOSH会为最后的tar包和yml生成形如x.x-dev这样的版本号来命名。

如果存在本地代码冲突，您可以添加“–force”选项：

|  |
| --- |
| $ bosh create release --with-tarball -–force |

这里会要求您输入deployment的名字，您可以自己定义一个易于识别的名字比如bosh1。

这一步可能需要一些时间才能完成，具体取决于您的网络速度。它首先会从一个 Blob 服务器下载二进制文件。然后它会构建包并生成清单文件。

#### 上传release

将创建好的Release tar包上传到BOSH的director。

|  |
| --- |
| $ bosh upload release dev\_releases/cloudfoundry-dev128.3.tgz |

使用bosh releases就可以看到我们上传好的releases。

#### Q&A

说明：如果遇到“You should use CLI >= 1.5.0.pre.3 with this release, you have 1.0.3”的提示，需要修改cf-release/config/final.yml文件中的min\_cli\_version值，将值改为1.0.3

**创建release的方案1：**

直接用清单文件来上传

$ cd cf-release

$ bosh upload release releases/appcloud-129.yml # use the highest number available - inspecting the files in this directory

**创建release的方案2：**

说明：这一步可以不创建release，通过直接用清单文件来上传，可以达到下一步部署cloudfoundry目的。

|  |
| --- |
| $ bosh upload release releases/appcloud-128.yml |

该命令在cf-release/release目录下操作。

**bosh create release过程中的问题1：**

|  |
| --- |
| Building daylimit...  Final version: Package `daylimit' has a glob that resolves to an empty file list: services/govendor/\*\*/\* |

解决办法：

|  |
| --- |
| cd src/services  git submodule update –init |

**执行bosh upload release时，release在上传完后提示 Error 100：execute expire**

----此问题是由于网络原因导致director没法访问网络，把网络配置正确或等网络空闲后重新上传

**执行create release时，如果出现下载gem失败导致不能创建release**，

需要找到失败的gem包，然后从.cache/getch\_gems中删除此gem包再重新执行create release，如果遇到多次执行都不能下载的包，自己去rubygems.org网上下载对应的版本，然后拷到.cache/fetch\_gems下再执行

**注意：**

1.如果您编辑了 CF Release中的代码，那么您可能需要在命令 bosh create release 中添加 –force 选项。

2.在运行此命令时系统一定要直接连接Internet。

3.如果您的网络速度慢或者您未与 Internet 建立直接连接，那么您可能需要在一个更好的环境中完成创建Release的操作。您可以在有良好 Internet 连接的机器上使用–with-tarball选项创建此Release。然后，您需要将所生成的 tar 包复制到所需系统。

### Cloudfoundry部署清单

如果实际job节点数少于资源池中的节点个数，资源池中的多余节点的ip不会使用network 里static定义的ip，会使用reserved以外的ip。

#### 部署计划

IP分配：192.168.2.51 - 192.168.2.80

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

| Job/index | State | Resource Pool | IPs |

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

| cc\_ccdb/0 | running | infrastructure | 192.168.2.82 |

| dashboard/0 | running | infrastructure | 192.168.2.95 |

| hbase\_master/0 | running | infrastructure | 192.168.2.98 |

| hbase\_slave/0 | running | infrastructure | 192.168.2.99 |

| hm/0 | running | infrastructure | 192.168.2.97 |

| login/0 | running | infrastructure | 192.168.2.96 |

| mysql\_node\_100/0 | running | services\_3gb\_small | 192.168.2.85 |

| nfs\_syslog\_nats\_redis/0 | running | infrastructure | 192.168.2.81 |

| opentsdb\_collector/0 | running | infrastructure | 192.168.2.94 |

| router/0 | running | infrastructure | 192.168.2.84 |

| stager\_dea/0 | running | deas | 192.168.2.90 |

| stager\_dea/1 | running | deas | 192.168.2.91 |

| stager\_dea/2 | running | deas | 192.168.2.92 |

| uaa\_uaadb/0 | running | infrastructure | 192.168.2.83 |

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

计划0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **作业** | **实例** | **IP** | **资源池** | **磁盘M** | **模板** |
| nfs\_syslog\_nats\_redis | 1 | 51 | infrastructure | 8000 | - debian\_nfs\_server  - syslog\_aggregator  - nats  - vcap\_redis |
| cc\_ccdb | 1 | 52 | infrastructure |  | - cloud\_controller  - postgres |
| uaa\_uaadb | 1 | 53 | infrastructure |  | - uaa  - postgres |
| router | 1 | 54 | infrastructure |  | router |
| stager\_dea | 2 | 55  56 | deas |  | - stager  - dea |
| service\_gateways | 1 | 60 | infrastructure |  | - mysql\_gateway  - postgresql\_gateway |
| mysql\_node\_100 | 1 | 61 | infrastructure |  | mysql\_node\_ng |
| postgresql\_node\_100 | 1 | 62 | infrastructure |  | postgresql\_node\_ng |
| health\_manager | 1 | 63 | small |  | health\_manager\_next |
|  |  |  |  |  |  |
| 增加dashboard |  |  |  |  |  |
| hbase\_slave\_master\_opentsdb\_collector | 1 | 64 | infrastructure |  | - hbase\_slave  - hbase\_master  - opentsdb  - collector |
| dashboard | 1 | 65 | infrastructure |  |  |

计划1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **作业** | **实例** | **IP** | **内存M** | **磁盘M** | **模板** |
| nats | 1 | 51 | 1000 | 8000 | nats |
| cloud\_controller\_ccdb | 1 | 52 |  |  | cloud\_controller  postgres |
| uaa\_uaadb | 1 | 53 | 1000 | 2048 | uaa  postgres |
| router | 1 | 54 |  |  | router |
| health\_manager | 1 | 55 |  |  | health\_manager\_next |
| stager\_dea | 2 | 56  57 |  |  | - stager  - dea |
| nfs\_syslog\_redis | 1 | 58 | 1000 | 8000 | debian\_nfs\_server  syslog\_aggregator  vcap\_redis |
| services\_nfs | 1 | 59 | 1000 | 102400 | debian\_nfs\_server |
| hbase\_slave\_master\_opentsdb\_collector | 1 | 60 | 1000 | 10048 | hbase\_slave  hbase\_master  opentsdb  collector |
| dashboard | 1 | 61 | 1000 | 8000 | dashboard |
| services\_redis | 1 | 62 | 1000 | 2048 | vcap\_redis |
| backup\_manager\_service\_utilities |  | 63 |  |  | - backup\_manager  - service\_utilities |

计划2

**必需节点**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **作业** | **实例数** | **IP** | **内存** | **磁盘 (GB)** |  |
| debian\_nfs\_server |  | 51 |  |  |  |
| nats |  | 52 |  |  |  |
| ccdb\_postgres |  | 53 |  |  |  |
| uaadb |  | 54 |  |  |  |
| vcap\_redis |  | 55 |  |  |  |
| uaa |  | 56 |  |  |  |
| acmdb |  | 57 |  |  |  |
| acm（不是必须，无该模版） |  |  |  |  |  |
| cloud\_controller |  | 59 |  |  |  |
| stager |  | 60 |  |  |  |
| router |  | 61、62 |  |  |  |
| health\_manager |  | 63 |  |  |  |
| dea |  | 65、66 |  |  |  |

**dashboard节点**

这些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。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **作业** | **实例数** | **IP** | **内存** | **磁盘 (GB)** |  |
| hbase\_slave |  | 73 |  |  | - |
| hbase\_master |  | 74 |  |  | - |
| opentsdb |  | 58 |  |  |  |
| collector |  | 67 |  |  |  |
| dashboard |  | 68 |  |  |  |

**可选节点**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **作业** | **实例数** | **IP** | **内存** | **磁盘 (GB)** |  |
| vblob\_node |  | 77 |  |  | - |
| vblob\_gateway |  | 78 |  |  | - |
| serialization\_data\_server |  |  |  |  |  |
| services\_redis |  | 69 |  |  |  |
| services\_nfs |  | 64 |  |  |  |
| syslog\_aggregator |  |  |  |  |  |
| login |  | 79 |  |  |  |
| mysql\_node |  |  |  |  |  |
| mysql\_gateway |  |  |  |  |  |
| redis\_node |  | 71 |  |  | - |
| redis\_gateway |  | 72 |  |  | - |
| rabbit\_node |  |  |  |  |  |
| rabbit\_gateway |  |  |  |  |  |
| postgresql\_node |  | 75 |  |  |  |
| postgresql\_gateway |  | 76 |  |  |  |
| service\_utilities |  | 80 |  |  | - |

离线包的部署计划：

nfs\_syslog\_nats\_redis 81

cc\_ccdb\_hm 82

uaa\_uaadb 83

router 84

mysql\_node\_100 85

mongodb\_node\_100 86

redis\_node\_100 87

rabbit\_node\_100 88

postgresql\_node\_100 89

stager\_dea 90

service\_gateways 91

#### 部署文件说明

cloudfoundry部署清单示例文件内容参见“附录🡪cf-lab-1.yml”。

#### 实际部署

|  |
| --- |
| $ bosh deployment cf-lab-1.yml  $ bosh deploy |

#### 验证

使用bosh deployments 和bosh vms

#### Q&A

1.通过使用“bosh cloudcheck”命令进行环境问题的检查、修改；修正虚拟机的错误状态（如job与虚拟机未挂接上、虚拟机未正常创建等）。

2.如果cf.yml中compilation（关于工作机）的设置过高，将无法创建虚拟工作机进行cloudfoundry的部署，需要调低设置。

3.在执行bosh deploy后提示Deployment name not found in the deployment manifest

---用16进制模式打开部署文件，开头有3个字节的不可见字符EF BB BF，删除后重新部署

4.Error 400007: ‘job/index’ is not running after update

在执行bosh deploy时偶尔会出现这个错误，重新执行即可

# 维护

## vCenter故障排除

**问题：**

新虚拟机IP与已有虚拟机IP（如NFS Server的IP）冲突后，造成已有虚拟机、BOSH、Cloudfoundry不能正常工作。

**解决方案：**

1. 使用ESXi主机上的“Restart Management Agent”；可以恢复虚拟机状态
2. 重启vCenter服务
3. 使用“主机—>配置🡪存储适配器、存储器”上的“全部重新扫描”功能刷新存储。（可选）
4. 重启Micro BOSH虚拟机（Micro BOSH的模板虚拟机虽然未启动，但需要在第一步的操作使他恢复正常状态，否则即使重启了Micro BOSH虚拟机，也无法访问。）。

通过以下一组命令确认micro bosh已恢复正常。

|  |
| --- |
| $ bosh target <micro bosh 的ip>  $ bosh login  $ bosh deployments  $ bosh releases  $ bosh vms |

如果以上命令能正常运行并返回预期的信息，则micro bosh恢复正常。

1. 在micro bosh恢复正常后，运行“bosh cck”检查多节点的bosh的运行状态。根据“bosh cck”的提示操作，先选择“reboot vm”进行恢复。如果无法恢复，则选择“bosh cck”命令提示中的“recreate vm 。。。”进行恢复。
2. 在多节点的bosh恢复正常后，运行“bosh cck”检查多节点的cloudfoundry的运行状态。根据“bosh cck”的提示操作，先选择“reboot vm”进行恢复。如果无法恢复，则选择“bosh cck”命令提示中的“recreate vm 。。。”进行恢复。

cloudfoundry恢复后，各节点中的原有数据和设置会随着节点的重建而恢复。

## 主机增加

往vCenter中增加一个新的vSphere主机后，需要在vCenter中为新的vSphere主机增加NFS 存储（NFS datastore）、网络（CF Network、Services Network），使得BOSH部署CloudFoundry时能够识别出网络中新增加的vSphere主机。

## powerDNS节点

第一次部署多节点BOSH时，如果powerDNS在Job列表中最后一个时，最后一个执行该dns job，则DNS这个节点上的数据库会没有表。因为powerDNS节点上的表有director job来创建。因此最好将powerDNS这个节点放在director之前。或者多节点BOSH部署完后，再重新部署一下director这个job。

## 修改‘最大打开文件数’

1. 修改DEA节点上的agent.rb

agent.rb: /var/vcap/packages/dea/lib/dea/agent.rb

搜索 ‘fds’,在def process\_dea\_start方法中修改num\_fds值为9000.

1. 使用root权限修改以下文件内容

/etc/security/limits.conf，增加以下内容，

|  |
| --- |
| \* soft nproc 10240  \* hard nproc 10240  vcap soft nofile 9000  vcap hard nofile 65000  root soft nofile 9000  root soft nofile 65000  \* soft nofile 10240 # 控制进程的打开最大文件数  \* hard nofile 10240 # 控制进程的打开最大文件数 |

/etc/pam.d/common-session，增加以下内容，

|  |
| --- |
| session required pam\_limits.so |

/etc/profile，增加以下内容，（该操作可选）

|  |
| --- |
| ulimit -HSn 10240 |

需要重启电脑，否则更改不会立即生效。

检查：

ulimit -a 这个是列出当前系统的所有内容限制

ulimit -n 这个是列出当前系统能够打开最大的文件句柄数。

重启电脑之后，就可以使用root方式进行修改/etc/profile中的ulimit -HSn 后的数字，再执行source /etc/profile，就可以马上看到效果了。（该操作可选）

查看进程的limits

1. ps -ea

查看进程信息。

1. cat /proc/进程id号/limits

查看指定进程下的操作限制内容。

## 修改cloud controller的nginx上传文件大小限制

当上传大文件到cloudfoundry时，文件大小超过256M，此时cloudfounry会返回413 的http 响应代码（上传文件大小超过nginx最大值）。

为了能够上传大文件，修改两个节点上的nginx.conf文件。

1、cloud controller节点上的nginx的“client\_max\_body\_size 556M;”参数值。

文件：/var/vcap/data/jobs/cloud\_controller/56.8-dev/config/nginx.conf，在http上下文中修改client\_max\_body\_size。

|  |
| --- |
| http {  ...  client\_max\_body\_size 556M;  } |

2、router节点上的nginx的“client\_max\_body\_size 556M;”参数值。

文件：/var/vcap/data/jobs/router/18.5-dev/config/nginx.conf，在http上下文中修改client\_max\_body\_size。

|  |
| --- |
| http {  ...  client\_max\_body\_size 556M;  } |

## 修改DEA节点的系统时间

因为MemcacheD要求各个客户端的时间要保持一致，所以统一DEA节点上的时间。bios和系统时间都使用本地时间。

使用ROOT用户登录进行操作。

1. 设置时区为UTC：dpkg-reconfigure tzdata，第一步中选择‘Etc’，第二步中选择‘UTC’
2. 查看系统时间和bios时间：

date; hwclock --show

1. 设置硬件时间

sudo hwclock --set --date ‘07/10/13 12:12:00’

1. 将系统时间设成硬件时间

sudo hwclock --hctosys

1. 再次用 date; hwclock --show 查看当前系统时间和硬件时间
2. 修改 .profile；加入如下内容

|  |
| --- |
| #hwclock --hctosys  TZ='Asia/Shanghai'  export TZ |

1. 重启后，再用date查看系统日期，达到修改日期和时区的目的。

备注：

1. 修改 ROOT用户的.profile，增加如下内容

TZ=’Asia/Shanghai’

export TZ

1. 修改时间： sudo date -s ‘20130710 10:15:00’

以上的步骤可以重复使用，最终达到效果。

## 使用部署文件更新多节点BOSH

如果在执行‘bosh deploy’命令时，出现各种错误提示，可以使用重启Micro BOSH节点上的虚拟机，或者在vCenter中选择Micro BOSH、多节点BOSH所在的vSphere主机 并选择右键菜单中的‘重新引导’功能进行重启，使用这两种方法可以修复错误提示。

## 修改CloudFoundry用户的内存分配量

根据cloud controller的部署文件模板（文件路径如下），

tempest\cf-release\jobs\cloud\_controller\templates\cloud\_controller.yml.erb

修改cloud foundry部署文件中关于cc（cloudcontroller）中的default\_account\_capacity值。内容如下：

|  |
| --- |
| cc:  srv\_api\_uri: http://api.cf.offline  password: zbTuuLaap44jadlas2l312lk  token: 267f4eaa8c1758f66d5cb7adcb24adb9d7  use\_nginx: true  new\_stager\_percent: 100  new\_stager\_email\_regexp: '.\*@vmware.com'  staging\_upload\_user: zxsfhgjg  staging\_upload\_password: ZNVfdase9  allow\_registration: true  **default\_account\_capacity:**  **memory: 8048**  uaa: |

参考：

tempest\cf-release\src\cloud\_controller\cloud\_controller\config\appconfig.rb

tempest\cf-release\src\cloud\_controller\cloud\_controller\config\cloud\_controller.yml

## 修复hbase\_master

问题：hbase\_master节点处于‘failing’状态。

查看该节点上的日志文件 /var/vcap/sys/log/hbase\_master/hbase\_master\_start.stdout.log，发现两个错误信息，分别处理如下：

错误1：没有权限修改/var/vcap/sys/log/hbase\_master/hbase\_master.log,此文件只有root有权修改。

解决办法：改成所有人都有权修改； chmod 666 hbase\_msater.log

错误2：错误的hbase.rootdir地址信息

解决办法1：修改源代码中的hbase.rootdir。在源码中，找到jobs下对应的hbase.hbase-site.xml.erb文件，把hbase.rootdir的值改成下面的形式

|  |
| --- |
| <property>  <name>hbase.rootdir</name>  <value>hdfs://<%= `nslookup #{properties.hbase\_master.address}`.gsub(/.\*name\s=\s/m,'').gsub(/\.\n\n/,'') %>:<%= properties.hbase\_master.hadoop\_namenode.port %>/hbase</value>  </property> |

解决办法2：修改hbase\_master节点上配置文件中的hbase.rootdir。

被修改的文件位置：

/var/vcap/data/jobs/hbase\_master/1.1-dev/config/hbase/hbase-site.xml

|  |
| --- |
| <property>  <name>hbase.rootdir</name>  <value>hdfs://192.168.2.98:9000/hbase</value>  </property> |

说明：192.168.2.98是hbase\_master节点上的主机IP地址 。

## 修改cloudcontroller中stage操作时间

查看tempest\cf-release\jobs\cloud\_controller\templates\cloud\_controller.yml.erb，可以看到staging下的 max\_staging\_runtime参数控制cloud controller的stage时间，这个时间可以在cloudfoundry的部署文件中进行配置；或者修改cloud controller节点上的/var/vcap/data/jobs/cloud\_controller/56.8-dev/config/cloud\_controller.yml文件中的该参数值。

## 更新login模块

**问题：**

login模块中的**cloudfoundry-login-server-1.2.0.war代码**存在问题，在spring中，转换属性类型时，会有Map和Properties类型不匹配问题。

**方案：**

步骤1：修改war包中env-context.xml文件，去掉org.cloudfoundry.identity.uaa.config.EnvironmentMapFactoryBean，替换成org.cloudfoundry.identity.uaa.config.EnvironmentPropertiesFactoryBean。

步骤2：使用命令sha1sum 文件名重新生成war包的SHA签名值。

步骤3：在cloudfoundry release目录中的config/blobs.yml中使用新值替换cloudfoundry-login-server-1.2.0.war对应的sha值，改一下war对应的文件大小属性值。

步骤4：把cloudfoundry-login-server-1.2.0.war放入.blobs目录中，改名成sha值。

建立blobs目录中cloudfoundry-login-server-1.2.0.war与.blobs目录中对应sha值为文件名的文件链接。

步骤5：cloudfoundry release目录里dev\_release目录内容不变，重新生成、上传、部署一个新的cloudfoundry release。

## 无SSL证书的设置

注意：如果您没有SSL证书的话，使用新版vmc是无法进行登陆我们当前的部署环境的。这时我们需要对配置文件和cf-release中的代码做如下两个修改：

1、在Cloud Foundry的部署文件cf-dev.yml中修改login的协议为HTTP：

**.**.**.**

properties:

**.**.**.**

login:

protocol:http

**.**.**.**

2、修改Cloud Controller的配置文件，文件位置：../cf-release/jobs/cloud\_controller/templates/cloud\_controller.yml.erb

**<**% if cc\_props**.**uaa**&&**properties**.**uaa**.**cc%>

uaa:

...

url: https://uaa.<%= properties.domain %>

**.**.**.**

**<**% else %>

uaa:

...

url: https://uaa.<%= properties.domain %>

**.**.**.**

**<**% end **%>**

把上述两个https改成http。另外：如果您的网络环境比较特殊的话，还可能会出现CloudController在内网中不能通过URL直接解析到UAA的IP，那么您还需要修改下面的源文件和hosts文件：

../cf-release/src/cloud\_controller/cloud\_controller/app/models/uaa\_token.rb

把这个文件中所有的 AppConfig[:uaa][:url] 替换为 "http://uaa.<yourdomain>:8080" ，要记得带双引号。

然后在部署完成后CC的虚拟机hosts中添加uaa的URL来帮助内网中进行正确的解析，这一部分会在后面有说明。

前面提到在缺少SSL证书且网络环境比较特殊的情况下还需要修改Cloud Controller的hosts文件来使uaa的URL能够正确解析：

登录到所有CloudController的虚拟机中，在他们的hosts中添加这样一行即可：

<IP\_of\_uaa> uaa.<yourdomain>

到这里，Cloud Foundry 已经完全安装好了。如果您迫不及待地想要验证此安装，您可以使用vmc命令将其中一个router的 IP 地址设为目标，然后在该目标上部署一个测试用的 Web 应用程序（见后一节）。由于此时没有配置 DNS 服务，因此，在vmc客户端以及运行浏览器来测试的机器的 hosts 文件需包含至少以下两行内容：

<router的 IP 地址> api.yourdomain.com

<router的 IP 地址><youapp>.yourdomain.com

如果上述测试顺利通过，则说明您的 Cloud Foundry 实例正常工作。最后要做的是部署负载均衡器和 DNS服务。它们不属于 Cloud Foundry 的组件，但在生产环境中往往需要正确地设置它们。我们简要地介绍一下如何设置。

## 增加dashboard模块

监控实例和cloudfoundry运维

dashboard的job，它可以让管理员监控Cloud Foundry的运行状态，这是日常运维中不可或缺的工具。在Dashboard中使用了OpenTSDB作为其数据存储。在我们部署Cloud Foundry的yml中已经给出了Dashboard这个job的配置方法。Dashboard需要使用OpenTSDB和collector，而OpenTSDB的正常运行则依赖于HBase Master和HBase Slave。因此，如果我们想要使用Dashboard，首先需要确认这些有关的job都在Cloud Foundry的yml文件中正确地配置并部署了。这里需要注意的是这些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。由于Dashboard的用户认证采用了UAA模块，因此我们在使用之前还要配置管理员用户和赋予其使用Dashboard的管理权限。配置的过程如下：

首先我们安装一个叫做uaac的命令行工具，登录到BOSH CLI这台虚拟机上安装必要的依赖：

### 安装uaac依赖包

for GEM in bundler rake rspec simplecov simplecov-rcov ci\_reporter highline rest-client yajl-ruby eventmachine launchy em-http-request ; do

gem install $GEM

done

将上面的执行脚本放入可执行文件中运行。ubuntu下执行脚本文件的创建参见“附录🡪shell 脚本文件”.

### 安装uaac

下载uaa的源码并使用源码安装uaac：

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

cd uaa/gem

bundle install

gem install cf-uaac

uaac -v

### 添加用户

（删除、添加用户均不需要特殊的用户权限，但需要‘target’、‘token client get dashboard’操作。）

uaac 安装成功以后我们需要使用他来完成添加用户的操作：

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

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

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

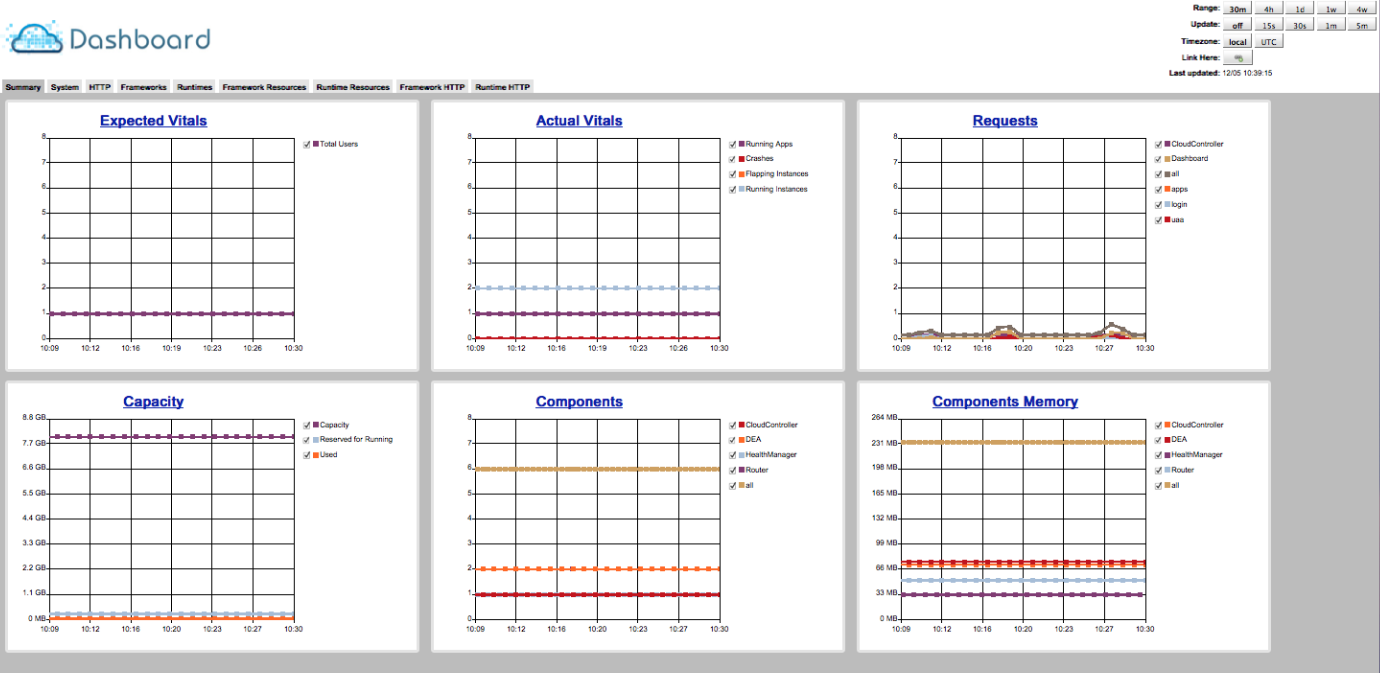
|  |
| --- |
| $ uaac user add test@do2013.com --given\_name test@do2013.com --e  mails [test@do2013.com](mailto:test@do2013.com) --family\_name [test@do2013.com](mailto:test@do2013.com) |

$ uaac member add dashboard.user <username1><username2> ... <usernameN>

# 上面的username1是通过vmc注册到cf中的用户邮箱

$ uaac token delete

如果在上面的操作中有错误提示group不存在，可参照uaac帮助信息来执行uaac group add [name]就可以了。命令中的secret就是在Cloud Foundry的yml文件中定义的属性值uaa.clients.dashboard.secret，dashboard.user是赋予用户使用dashboard的权限，username1等用户名就是我们配置过的Cloud Foundry管理员用户邮箱（比如：admin@vmware.com）。配置完成后，我们就可以登录Dashboard了。登录过程会跳转到UAA的页面，登录成功后就可以看到Dashboard的页面了。如果这一步登录后只显示登录成功没有跳转到Dashboard，手动访问一下Dashboard的URL(dashboard.yourdomain.com)即可。



我们可以看到在这个页面上有多个标签页，每一个标签页代表了一种我们的监控项。现在我们切换到System标签页，有可能发现这一页上是没有数据的。System标签页负责监控当前环境中所有Cloud Foundry虚拟机的资源状况，这些数据的来源需要BOSH的HealthMonitor组件和Cloud Foundry的OpenTSDB一起来负责。所以查看我们的BOSH部署文件bosh.yml中HealthMonitor的properties部分，我们会看到tsdb部分被注释掉了：

hm:

...

smtp:

...

# tsdb\_enabled: true

# tsdb:

# address: 10.40.97.85

# port: 4242

之所以注释掉的原因是这里HM依赖的OpenTSDB是Cloud Foundry的一个job，也就是说我们在部署BOSH的时候OpenTSDB还没有安装，所以需要暂时注释掉了这个部分。当我们按照文章第三部分部署过Cloud Foundry之后，OpenTSDB就存在了，所以我们只需要把这部分注释去掉（注意tsdb要和上面的smtp处于同一级别），然后重新deploy一次BOSH就可以了。由于BOSH的部署是增量式的，所以这次更新BOSH的操作只需要很短的时间就能完成。

$ bosh target http://<ip\_of\_micro\_bosh>:25555

$ bosh deployment bosh.yml

$ bosh deploy

部署完成后，直接访问Dashboard并查看System标签页，我们就能看到System的数据了。

## 实例运行环境动态伸缩

## 实例的负载均衡

## CF的负载均衡

## 粘性session

## 添加已部署过job的实例

1. 在部署文件中找到原实例对应的resource，增加size的值
2. 在部署文件中对应的job增加instances的值
3. 如果使用固定ip，在static\_ips下输入新分配的ip

添加新job

# 使用

部署应用到cloudfoundry之前，需要使用vmc工具先登录到cloudfoundry site上。

## 部署工具介绍（适用于v1 cloudfoundry）

通过浏览器登录<http://login.cf.local可以正常登录，通过0.4版本以上的vmc>操作 login无法登录，提示请求内容格式错误。

VMC 是使用Cloud Foundry的命令行工具。它可以执行 Cloud Foundry 上的大多数操作，例如配置应用程序、将应用部署到 Cloud Foundry 以及监控应用程序的状态。要安装 VMC，需要先安装 Ruby 和RubyGems（ Ruby Gem管理器）。目前支持 Ruby 1.8.7 和 1.9.2。接着，您可以通过下面的命令安装 VMC（有关 VMC 安装的更多信息，请参见<http://docs.cloudfoundry.com/tools/vmc/installing-vmc.html>）：

$ gem install vmc –v 0.3.18

现在，可指定该 Cloud Foundry 实例的目标，相应的 URL 应形如 api.yourdomain.com，例如：

$ vmc target api.cf.local

使用管理员用户和密码（部署清单中指定了该凭据）登录：

$ vmc login

起初，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：

$ vmc info

## 部署standalone 应用（tomcat）

[**新提供的独立框架支持**](http://blog.cloudfoundry.com/2012/05/01/cloud-foundry-improves-support-for-background-processing/)大大增加了可以在 Cloud Foundry 上运行的各种非 Web 应用程序的类型数目，包括应用程序服务器。将应用程序安装到 Tomcat 7 实例中，对配置稍作改动，然后将 Tomcat 7 及您的应用程序的全部内容作为独立应用程序推送到 Cloud Foundry。

**第1步 - 下载 Apache Tomcat**

[**将 Apache Tomcat 7 下载**](http://tomcat.apache.org/download-70.cgi)（Apache 7.0.27）到您将使用 **vmc** 命令行工具的位置。其文件名为 apache-tomcat-7.0.27.zip。部署过程中，自始至终的所有命令都将假定当前工作目录为 Tomcat 7 基目录。

**第2步 -解压缩 Tomcat 并更新权限**

将 tomcat zip 文件解压缩到一个本地目录。如果在window环境下，不需要更改tomcat中sh文件的权限。

**第3步 -编辑启动脚本**

**bin/startup.sh**

Tomcat 通常是使用 bin/startup.sh 脚本启动的。为了让 Tomcat 使用调用 startup.sh 的那个 shell 而不是生成一个新 shell，请将 startup.sh 最后一行中的执行参数从“start”更改为“run”：

|  |
| --- |
| exec "$PRGDIR"/"$EXECUTABLE"start "$@" |

改为：

|  |
| --- |
| exec "$PRGDIR"/"$EXECUTABLE"run "$@" |

**bin/catalina.sh**

我们希望 Tomcat 7 使用的不是预定义的静态端口，而是由 Cloud Foundry 分配的端口，在部署时此端口将存储在 VCAP\_APP\_PORT 环境变量中。将下面的 Bash 代码放在 catalina.sh 顶部附近的初始注释后面。这样做只是为了使我们无需修改代码也可在本地运行此端口，如果动态端口不可用作环境变量，此代码将分配一个静态端口号 8080。

|  |
| --- |
| # USE VCAP PORT IF IT EXISTS, OTHERWISE DEFAULT TO 8080    if[ -z ${VCAP\_APP\_PORT} ]; then    export VCAP\_APP\_PORT=8080    fi    export JAVA\_OPTS="-Dport.http.nonssl=$VCAP\_APP\_PORT $JAVA\_OPTS" |

第 4 步 – 编辑 Tomcat 配置

**conf/server.xml**

将 Server 元素的 port 属性设置为 -1，这将禁用 Tomcat 关闭端口。Cloud Foundry 不使用关闭端口，因为它通过发出“kill -9 PID”命令来停止任何独立应用程序实例。我们希望避免与在同一 Droplet Execution Agent (DEA) 上运行的其他应用程序产生任何潜在的端口冲突，因此目前建议对在 Cloud Foundry 上运行的独立应用程序仅使用单个 http 端口。

|  |
| --- |
| <Server port=”-1” command=”SHUTDOWN”> |

[**查看原始内容**](https://gist.github.com/raw/2886710/a984869ddbe2a61f1810bb393ddcc7a1c8433bee/Server.xml) [**Server.xml**](https://gist.github.com/2886710#file_server.xml)。[**此 Gist**](https://gist.github.com/2886710) 由 [**GitHub**](http://github.com/) 惠供。

由于 Cloud Foundry 无需使用 AJP 连接器即可为您处理负载平衡，您应通过注释掉下面显示的节来禁用 AJP 连接器，以确保我们不会遇到端口冲突。

|  |
| --- |
| <!-- Define an AJP 1.3Connector on port 8009  <Connector port="8009"protocol="AJP/1.3"redirectPort="8443"/>  --> |

Connector 元素应使用在 JAVA\_OPTS 环境变量中提供的端口，我们之前已在 catalina.sh 脚本中设置了此变量。

|  |
| --- |
| <Connector port="${port.http.nonssl}"protocol="HTTP/1.1"  connectionTimeout="20000"  redirectPort="8443"/> |

[**查看原始内容**](https://gist.github.com/raw/2886705/ae9eaaa41f4b60550654421839dfc3f3fefaa38a/Connector.xml) [**Connector.xml**](https://gist.github.com/2886705#file_connector.xml)。[**此 Gist**](https://gist.github.com/2886705) 由 [**GitHub**](http://github.com/) 惠供。

如果是在linux环境下发步应用，建议在本地服务器上试着应用一下所做的编辑，看看 Tomcat 7 是否像预期的那样启动。为了测试采用的是否为 VCAP\_APP\_PORT，我建议使用一个命令 shell 指定一个示例端口，如 8082。

|  |
| --- |
| jbayer$ export VCAP\_APP\_PORT=8082  jbayer$ bin/startup.sh |

控制台不应该返回（当 Tomcat 运行时它应该阻塞），控制台输出中的最后几行中应有一行为：

|  |
| --- |
| INFO: Starting ProtocolHandler ["http-bio-8082"] |

如果确实如此，那么您应该能够访问 [**http://localhost:8082**](http://localhost:8082/) 来看看是否出现欢迎页面。此时，您可能希望备份（压缩成 zip）包含了您到目前为止所进行的各种定制的整个 Tomcat 7 目录，以便您今后可以对其他应用程序重用该目录。

第 5 步 – 安装您的应用程序

Cloud Foundry 将 ROOT Web 应用程序与 Tomcat 6 一起使用，那我们就对 Tomcat 7 也采用同样的做法。

首先，删除现有的 ROOT 应用程序。

|  |
| --- |
| jbayer$ rm –r webapps/ROOT |

现在该您的 Web 应用程序上台了，请将它解压缩到 webapps/ROOT 位置。如果您有 .war 文件，那么解压缩命令大致如下：

|  |
| --- |
| jbayer$ unzip –d webapps/ROOT ~/dev/mytomcat7.war |

在本地检查您的应用程序，看看它是否正确运行。

第 6 步 – 将此应用程序推送到 Cloud Foundry

从 Tomcat 7 基目录执行 vmc 命令，执行时对于很多默认选择请予以采纳。请注意，vmc 会自动检测到它是“独立应用程序”，因此您只需提供启动脚本路径并映射一个唯一 URL 即可。还请注意，如果此 URL 不是唯一的，将会出错。

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

在推送过程结束时，您将有机会选择将此配置写入到一个清单文件中。下面是生成的 manifest.mf 文件，该文件是从我们的部署中写入到 Tomcat 7 基目录的。请注意，该文件包含启动命令 bin/startup.sh。如果根目录中存在该文件，那么下次您推送此应用程序时 vmc 将从该文件中读取内容并跳过上述交互问题。

|  |
| --- |
| ---  applications:    .:      url: mytomcat7.cloudfoundry.com      command: bin/startup.sh      runtime: java      framework:        info:          exec:          description: Standalone Application          mem: 64M        name: standalone      name: mytomcat7      instances: 1      mem: 256M |

**总结**

诸如 Jetty 等其他容器所遵循的模式与上述模式类似。大多数应用程序都应能够使用 Cloud Foundry 提供的现有框架。如需定制或引入您自己的容器，可以考虑使用Cloud Foundry 独立应用程序支持。

# .Net

## win 2008 core server:

11、启用远程管理 (启用后，可以使用远程桌面连接服务器)

需要在防火墙上打开3389端口，在客户端可以用mstsc命令进行远程连接。

netsh firewall add portopening TCP 3389 name=”Remote Desktop”

cscript scregedit.wsf /ar 0

cscript scregedit.wsf /cs 0

设置通过远程MMC管理

首先要打开135端口，然后用MMC进行远程连接。

**其他**

使用 Net Use 命令映射或断开驱动器

使用 net use 命令对批处理文件和脚本可能很有用。要使用 net use 命令映射或断开驱动器，请按照下列步骤操作： · 要映射网络驱动器，请使用 net use x:\\计算机名称\共享名称 命令，其中 x: 是要分配给共享资源的驱动器号。 如果此映射要一起生效，则要加上参数，如：net use t: \\192.168.2.112\c$

要断开映射的驱动器，请使用 net use x: /delete 命令，其中 x: 是共享资源的驱动器号。

## Iron Foundrys

c:\Users\MyUserName\start /w ocsetup IIS-WebServerRole

c:\Users\MyUserName\start /w ocsetup WAS-NetFxEnvironment

c:\Users\MyUserName\start /w ocsetup IIS-ISAPIExtensions

c:\Users\MyUserName\start /w ocsetup IIS-ISAPIFilter

c:\Users\MyUserName\start /w ocsetup IIS-NetFxExtensibility

c:\Users\MyUserName\start /w ocsetup IIS-ASPNET

c:\Users\MyUserName\start /w ocsetup MicrosoftWindowsPowerShell

c:\Users\MyUserName\start /w ocsetup ServerCore-WOW64

c:\Users\MyUserName\start /w ocsetup NetFx2-ServerCore

c:\Users\MyUserName\start /w ocsetup NetFx2-ServerCore-WOW64

# 注意事项

1. 部署文件在BOSH发布cloudfoundry中起着重要的配置作用，调节各个模块的设置、安装、运行阶段期间的各种参数。

在cf-release中jobs目录下各个子模块templates模块中有一个形如 \*.yml.erb的默认参数配置模板，依据该模板，接收部署文件中参数，并对其进行合并处理，最终形成一个BOSH发布、运行cloudfoundry时所需要的参数值。

有时，cf-release不同版本会对应不同版本的各模块templates内容。

1. dashboard和login都是java web类型、运行在tomcat上的cf模块。如果部署在同一个节点虚拟机上的话，容易引起端口冲突，导致这两个模块都无法运行。
2. 在部署多节点的BOSH，要把powerDNS节点部署上。否则在多节点的cloudfoundry上，在dashboard、cloudcontrol等节点上需要单独设置login、uaa等域名映射；同时会使其他的节点出现类似的域名解析异常。

# 问题

## dashboard：executing action "FetchChartData", java.lang.NumberFormatException: For input string:

Error making TSDB request. (Connection to http://192.168.2.86:4242 refused)

A：重新发布dashboard的相关节点模块后，现象消失。

## failed to get updated incarnation from monit

A：不做其他修改，重新运行“bosh deploy”命令后，该错误不再出现。

## Could not get JDBC Connection; nested exception is java.sql.SQLException: Driver:org.hsqldb.jdbcDriver@3c6b5360 returned null for <URL:jdbc:://192.168.2.83:2544/uaa>

A: 参考cf-release/jobs/uaa/templates/uaa.yml.erb的默认参数配置模板，依据该模板，发现cf-small.yml部署文件里uaadb缺少 db\_scheme: postgresql属性，将该属性加上可解决问题。

## bosh cck时出现：binding existing deployment: VM `vm-3f77e352-1d89-495f-b9af-dea369516bb1' is out of sync: expected to be a part of deployment

A: 先关掉出问题的虚拟机电源，使用bosh cck进行检查，选择“recreate vm。。。”选项进行虚拟机重建。

## 使用“vmc register”注册用户时，出现“Creating New User: Error 100: Bad request”错误提示。

A：与cc相关联的数据库的user数据表里存在与要注册的用户重名的数据，将user数据表里的重名用户删掉即可。

|  |
| --- |
| <http://support.cloudfoundry.com/entries/20465077-what-are-involved-in-vmc-add-user-I-am-getting-error-100->   We will get 'Bad request' if the database backing cloud controller (postgres in local vcap install) is not available for  some reason.  It led me to check on the database side of the log and I found that this particular user is already exist.  What I did to resolve this (since I do not know the password now) is to delete the user from the sqlite3 database and restart CC. It allows me to create this user now. |

## 使用cloudfoundry的standalone方式部署框架应用时，出现错误信息。

错误信息如下：

|  |
| --- |
| 2013-05-06 09:00:32,871 [com.mchange.v2.async.ThreadPoolAsynchronousRunner]-[WARN] com.mchange.v2.async.ThreadPoolAsynchronousRunner$DeadlockDetector@5554ea79 -- APPARENT DEADLOCK!!! Creating emergency threads for unassigned pending tasks! |

该错误信息出现在/var/vcap/data/dea/apps/tom58-0-4190d7f1266526216d4b594cd7ecfb5b/logs/stdout.log，这个会使应用无法正常启动，启动的信息如下：

|  |
| --- |
| May 6, 2013 9:17:29 AM org.apache.catalina.core.StandardEngine start  INFO: Starting Servlet Engine: Apache Tomcat/5.5.35  May 6, 2013 9:17:29 AM org.apache.catalina.core.StandardHost start  INFO: XML validation disabled |

该启动信息来自于/var/vcap/data/dea/apps/tom58-0-4190d7f1266526216d4b594cd7ecfb5b/logs/stderr.log。

stdout.log和stderr.log这两个文件所在路径中的tom58-0-4190d7f1266526216d4b594cd7ecfb5b路径名字会根据vmc对应用的发布操作（比如：重新启动应用等）而变化。

A: 因为数据库服务器所在的机器上的防火墙将自己的1524端口设为其他机器不可访问，使得其他机器上应用无法Oracle数据库。解决办法：修改他的防火墙设置，在防火墙中把Oracle的1524端口开放出来。

## hbase\_mater中的错误信息

A:修改文件权限 /var/vcap/sys/log/hbase\_master/hbase\_master.log,此文件只有root有权修改，改成所有人都有权修改 chmod 666 hbase\_msater.log

修改hbase.rootdir,

在源码中，找到jobs下对应的hbase.hbase-site.xml.erb文件，把hbase.rootdir的值改成下面的形式

|  |
| --- |
| <property>  <name>hbase.rootdir</name>  <value>hdfs://<%= `nslookup #{properties.hbase\_master.address}`.gsub(/.\*name\s=\s/m,'').gsub(/\.\n\n/,'') %>:<%= properties.hbase\_master.hadoop\_namenode.port %>/hbase</value>  </property> |

## Cannot find file `tempestL/tempest/cf-release/dev\_releases/appcloud-128.7-dev.yml'

A: 引起该错误的原因：没有在cf-release的根目录下执行“bosh upload release”命令。

## Error 100：release\_id and name and version unique

A:在“bosh upload release”命令中加上“--rebase”的参数，可以消除错误信息。

## 在dashboard登录时，页面显示内部错误，登录到uaa上查看日志，有java.lang.IllegalArgumentException: familyName is required，givenname is required等信息

使用postgreSQL 客户端工具连接到uaadb，为users表的familyName 和givenName字段赋上相应的值即可

## uaac member add dashboard.user 时的错误信息：CF::UAA::BadResponse: invalid status response: 500

uaa节点上的日志文件/var/vcap/sys/log/uaa/uaa.log的错误信息摘要如下：

|  |
| --- |
| org.springframework.http.converter.HttpMessageNotWritableException: Could not write JSON: No enum const class org.cloudfoundry.identity.uaa.scim.groups.ScimGroup$Authority.MEMBER (through reference chain: org.cloudfoundry.identity.uaa.scim.groups.ScimGroup["members"]); nested exception is org.codehaus.jackson.map.JsonMappingException: No enum const class org.cloudfoundry.identity.uaa.scim.groups.ScimGroup$Authority.MEMBER  …  …  …  Caused by: java.lang.IllegalArgumentException: No enum const class org.cloudfoundry.identity.uaa.scim.groups.ScimGroup$Authority.MEMBER  at java.lang.Enum.valueOf(Enum.java:196)  **at org.cloudfoundry.identity.uaa.scim.groups.ScimGroup$Authority.valueOf(ScimGroup.java:15)**  **at org.cloudfoundry.identity.uaa.scim.groups.JdbcScimGroupMembershipManager$ScimGroupMemberRowMapper.getAuthorities(JdbcScimGroupMembershipManager.java:293)**  **at org.cloudfoundry.identity.uaa.scim.groups.JdbcScimGroupMembershipManager$ScimGroupMemberRowMapper.mapRow(JdbcScimGroupMembershipManager.java:287)**  at org.cloudfoundry.identity.uaa.scim.groups.JdbcScimGroupMembershipManager$ScimGroupMemberRowMapper.mapRow(JdbcScimGroupMembershipManager.java:280)  at org.springframework.jdbc.core.RowMapperResultSetExtractor.extractData(RowMapperResultSetExtractor.java:92)  at org.springframework.jdbc.core.RowMapperResultSetExtractor.extractData(RowMapperResultSetExtractor.java:1)  at org.springframework.jdbc.core.JdbcTemplate$1.doInPreparedStatement(JdbcTemplate.java:649)  at org.springframework.jdbc.core.JdbcTemplate.execute(JdbcTemplate.java:587) |

A：参考标红色的日志信息提示，将ScimGroupMemberRowMapper.mapRow所使用的 UAA数据库group\_membership表authorities列数据和ScimGroup$Authority枚举类型常量值进行比较发现，数据库中的authorities列数据是“MEMBER”，而ScimGroup$Authority枚举类型常量值是“READ”和“WRITE”，数据库中的数据不符合ScimGroup$Authority枚举类型常量值。将数据库中的authorities列数据改成“WRITE”后，重新运行“uaac member add dashboard.user <username>”命令后，用户被成功加入到uaa中。group\_membership表authorities列数据也恢复正常。

## HTTP exception: RestClient::RequestTimeout:Request Timeout

A1：

One option here is to try to push the application like so:

1. vmc push <j2ee-app> --path <path-to-app> --no-start

2. If the above fails, then you can try a variation of the above like so:

    vmc update <j2ee-app> --path <path-to-app>

You may have to do the update a few times. CF fingerprints / caches application artifacts and you may see the upload size reported by the "Uploading (XX) .." to slowly whittle down till the app is uploaded successfully. At that point you can start the app with "vmc start ,j2ee-app>".

A2: 增加vmc 的http request的timeout参数值。

<http://grokbase.com/t/cloudfoundry.org/vcap-dev/12aptmw989/deploying-app-on-cloudfoundry>

D:/reference/PAAS/tools/Ruby193/lib/ruby/gems/1.9.1/gems/vmc-0.3.18/lib/vmc/client.rb

390th line

|  |
| --- |
| req = {  :method => method, :url => "#{@target}/#{path}",  :payload => payload, :headers => headers, :multipart => true,  :timeout => 600  } |

A3：

from the dist folder of the project run  
**vmc push projName --no-start**  
wait for the upload to complete (nice thing is that my war is 24MB, but upload was about 100K)

then **vmc start projName**

## Why is my app returning "504 Gateway Time-out" errors?

A1：

The error page is generated by the CloudFoundry.com HTTP Router and not your application. Application requests taking longer than 30 seconds are terminated by the Router. The 30-second timeout cannot be adjusted and applies to all apps running on CloudFoundry.com. To troubleshoot this issue, please investigate what is causing the delayed response from your app and ensure one is returned in less than 30 seconds.

A2：

<http://support.cloudfoundry.com/entries/20499693-size-of-the-application-to-upload-to-cf-com>

Uploading large applications (100+ megs) results in failure is a known issue.  Please consider

  reducing the size of the application.

  One known workaround for this issue, are the initial push ( which results in failure), repeat with

  'vmc update <appname> --no-start' (this step you may have to do many times) till you get the

  application uploaded successfully.

  We have an internal issue tracking this, I will add one more vote for this issue to be fixed pointing

  to this forum thread.

A3：

Even when using the vmc command line client, I have gotten 504's pushing an app/WAR file to CF. Generally this is with large apps and usually deleting the app and trying push again works. There's not much more that I can really suggest at this time but to try again. Sorry.

In WM 6.5, we've improved the archiver we use to package up the  
app to send to CF. This can't eliminate 504's, but will help.

I should also note that when running apps ON cloudfoundry, the nginx server has a 30 second timeout. Any request exceeding 30 seconds will return a 504. For Studio running on CF, we've implemented long polling to handle this. Micro CF has increased this timeout with version 1.2, but cloudfoundry.com is currently sill at 30 sec.

## Error 402: App packaging failed: 'Failed synchronizing resource pool'

A：vmc push appName --no-resource

## vmc instances appName number扩展应用实例个数时，重启正在运行的实例

对于正在运行的实例，这会影响访问它的session。

## error "Operation not permitted" appears when to delete VMC User

A:

It sounds like the user that is executing the 'vmc delete-user' is not an  
admin user.  
You can make a user admin by adding 'admins: [ema[...@...](http://grokbase.com/t/cloudfoundry.org/bosh-users/12c763zbc0/how-to-delete-vmc-user)com]' to the  
cloud controller yaml and restarting the cloud controller.

## vmc create-service出现的错误

|  |
| --- |
| status:502;  body:{"code":503,"description":"Unexpected response from service gateway"}; |

|  |
| --- |
| [2013-05-23 16:40:35.891621] cc - pid=20555 tid=1c27 fid=c7a3 ERROR -- Error talking to gateway: Reponse status:400,error:[{:code=>30003, :description=>"Unknown plan free"}]  [2013-05-23 16:40:35.892206] cc exception pid=20555 tid=1c27 fid=c7a3 ERROR -- VCAP::Services::Api::ServiceGatewayClient::ErrorResponse<<Reponse status:400,error:[{:code=>30003, :description=>"Unknown plan free"}]:/var/vcap/packages/cloud\_controller/cloud\_controller/vendor/bundle/ruby/1.9.1/gems/vcap\_common-2.0.7/lib/services/api/clients/service\_gateway\_client.rb:180:in `perform\_request',/var/vcap/packages/cloud\_controller/cloud\_controller/vendor/bundle/ruby/1.9.1/gems/vcap\_common-2.0.7/lib/services/api/clients/service\_gateway\_client.rb:77: |

# 附录

## Ubuntu

重启：　shutdown -r now

### 文件句柄数

在Ubuntu下启动java应用时，在控制台输出会抛出打开太多的文错误信息。

open too many file

是因为应用打开的文件句柄数超出了系统的限制。使用命令：

ulimit -n

输出1024

root用户是可以的？普通用户又会遇到这样的问题？

看一下/etc/security/limits.conf大概就会明白。

linux对用户有默认的ulimit限制，而这个文件可以配置用户的硬配置和软配置，硬配置是个上限。

超出上限的修改就会出“不允许的操作”这样的错误。

于是我用root权限修改了/etc/security/limits.conf文件，增加以下内容：

\* soft nproc 10240

\* hard nproc 10240

\* soft nofile 10240

\* hard nofile 10240

\*代表domain，影响到的域，我这里设置了全部。

soft/hard ：代表软性或者硬性限制

nproc：max number of processes 设置最大的进程数

nofile：max number of open files 设置最大的文件句柄数

再编辑/etc/pam.d/common-session,加入一行，之所以加入这行，是因为需要在下次重启的时候，能够让用户直接修改

/etc/profile中的文件句柄数生效。http://www.ccs.neu.edu/home/battista/articles/winbind/pam.html （具体这个文件是做什么用的，我在下次再讨论下。）

session required pam\_limits.so

再编辑/etc/profile文件，加入刚才的内容：

ulimit -HSn 10240

此时执行：

source /etc/profile 仍然会提示：

bash: ulimit: open files: 无法修改 limit 值: 不允许的操作

这是因为需要重启电脑，否则更改不会立即生效。

检查：

ulimit -a 这个是列出当前系统的所有内容限制

ulimit -n 这个是列出当前系统能够打开最大的文件句柄数。

重启电脑之后，就可以使用root方式进行修改/etc/profile中的ulimit -HSn 后的数字，再执行source /etc/profile，就可以马上看到效果了。

## 无线路由器设置

对于无线路由器控制的局域网以外的机器，如果它需要通过无线路由器的wan口访问无线路由器控制的局域网内的某台机器，可以设置无线路由器的DMZ主机。

步骤如下：

1. 登录无线路由器：<http://192.168.2.1> admin admin
2. 进入‘转发规则🡪DMZ主机’，界面如下：



2.1激活DMZ的‘启用’状态。

2.2 在‘DMZ主机IP地址‘中输入无线局域网内某台机器的IP地址，该IP地址对应的机器能被外网能通过无线路由器WAN端口IP地址直接访问。

比如：

10.68.199.5 是无线路由器WAN口上的IP地址；

192.168.2.103是一台windows操作系统，且已经设置了远程桌面控制；

无线局域网以外的机器可以通过远程桌面输入10.68.199.5，直接访问192.168.2.103机器上的操作系统。

DMZ设置说明：

路由器类似于某单位的电话交换机。

每个分机都可以打出电话；但是如果别人想打进来，只知道总机号是不行的，不知道分机号就没办法转到特定的某部分机。通常我们听到的提示就是“请拨分机号，查号请拨○”，不知道别的分机号码的时候，我们就只好拨○了，拨○就转到特定的分机了，这部分机就相当于路由器的DMZ主机。

　　常见的家用上网的路由器就相当于这么个功能不太完善的电话交换机。默认情况下只支持拨出，就是浏览网页、发电子邮件等没有问题。但是如果想建设个网页，让别人来访问、或者为外网提供其他服务、或者想通过NetMeeting被外网的其他人呼叫就不行啦，因为外网的访问信息到了路由器以后就被阻挡住了，外网的计算机联络不到内网的计算机。默认的状态下只能单向通信。

　　当有这种需求的时候最简单的方法就是启用DMZ主机。启用后，外网的机器发往路由器外网地址（路由器外网地址是路由器WAN口上设置的IP地址）的所有信息都会被路由器转发给指定的DMZ主机。

当然，更好的办法是根据需要设置相应的转发规则。

## Nginx

#设置保留链接超时时间为75秒 设置header超时时间为20秒  
keepalive\_timeout 75 20;

## 搭建DNS Server

### PowerDNS

编辑powerdns 配置文件 /etc/powerdns/pdns.conf ,使其使用mysql 数据库,找到如下字段,修改如下, recursor 表示如果查询的域名不在本机上,则向上级域名服务器查询,例如，查询的地址为 192.168.2.11

### bind9

1、在ubuntu上安装bind9

|  |
| --- |
| $ sudo apt-get install bind9 |

2、 Configuring BIND9（配置 BIND9）

BIND9 配置文件被保存在 /etc/bind/ 目录里。

主配置文件被保存在下列文件中

/etc/bind/named.conf

/etc/bind/named.conf.options

/etc/bind/named.conf.local

3、将Bind9设成Master Server（主服务器）

3.1 要添加 DNS 域到 BIND9，让 BIND9 成为主服务器，只需要编辑文件/etc/bind/named.conf.local

|  |
| --- |
| //forward  # This is the zone definition. replace example.com with your domain name  zone "cf.offline" {  type master;  file "/etc/bind/db.cf.offline";  };  //backward  # This is the zone definition for reverse DNS. replace 0.168.192 with your network address in  #reverse notation - e.g my network address is 192.168.0  zone "2.168.192.in-addr.arpa" {  type master;  notify no;  file "/etc/bind/db.2.168.192";  }; |

3.2 编辑 域定义文件 /etc/bind/db.cf.offline

|  |
| --- |
| ;  ; BIND data file for local loopback interface  ;  $TTL 604800  // replace example.com with your domain name. do not forget the . after the domain name!  // Also, replace ns1 with the name of your DNS server  @ IN SOA ns.cf.offline. root.cf.offline. (  // Do not modify the following lines!  2 ; Serial  604800 ; Refresh  86400 ; Retry  2419200 ; Expire  604800 ) ; Negative Cache TTL  ;  @ IN NS ns.cf.offline.  @ IN A 127.0.0.1  @ IN AAAA ::1  ns IN A 192.168.2.84  www IN A 192.168.2.84  uaa IN CNAME www  api IN CNAME www  login IN CNAME www  dashboard IN CNAME www  user IN CNAME www  platform IN CNAME WWW |

* 1. 编辑 反转DNS域名 文件 /etc/bind/db.2.168.192

|  |
| --- |
| ;  ; BIND reverse data file for local loopback interface  ;  $TTL 604800  //replace example.com with yoour domain name, ns1 with your DNS server name.  // The number before IN PTR example.com is the machine address of the DNS server. in my  // case, it's 1, as my IP address is 192.168.0.1.  @ IN SOA ns.cf.offline. root.cf.offline. (  1 ; Serial  604800 ; Refresh  86400 ; Retry  2419200 ; Expire  604800 ) ; Negative Cache TTL  ;  @ IN NS ns.  84 IN PTR ns.cf.offline.  84 IN PTR www.cf.offline.  84 IN PTR uaa.cf.offline.  84 IN PTR api.cf.offline.  84 IN PTR login.cf.offline.  84 IN PTR dashboard.cf.offline.  84 IN PTR user.cf.offline  84 IN PTR platform.cf.offline |

4、 重启 bind:

|  |
| --- |
| sudo /etc/init.d/bind9 restart |

5、测试DNS server

在ubuntu客户机上，修改 /etc/resolv.conf

|  |
| --- |
| // replace example.com with your domain name, and 192.168.0.1  // with the address of your new DNS server.  search example.com  nameserver 192.168.0.1 |

测试域名映射：

|  |
| --- |
| dig example.com |

**备注：**

**DNS Record Types（DNS 记录类型）**

DNS 记录类型是有很多不同的，但对于阅读本文档的人来说，您只需要处理以下这些记录类型

**Address Records（地址记录）**

最常用的记录类型

|  |
| --- |
| www IN A 1.2.3.4 |

**Alias Records（别名记录）**

常用于为一个已有的 A 记录创建别名。您不能创建一个CNAME记录指向另一个CNAME记录。

|  |
| --- |
| mail IN CNAME www  www IN A 1.2.3.4 |

**Mail Exchange Records（邮件交换记录）**

常用于定义邮件发往何处。必须指向一个 A 记录，不能是 CNAME。

|  |
| --- |
| IN MX mail.example.com.  [...]  mail IN A 1.2.3.4 |

## 设置NFS

在Ubuntu下安装、配置Ubuntu nfs服务的步骤如下：

1. 安装Ubuntu nfs  
   Ubuntu上默认是没有安装Ubuntu nfs服务器的，因此我们首先安装Ubuntu nfs服务器端：

$sudo apt-get install nfs-kernel-server  
在一些文档中，提出还需要使用apt-get来手动安装nfs的客户端nfs-common，以及端口映射器portmap，但其实这是没有必要的，因为在安装nfs-kernel-server时，apt会自动为我们把它们安装好。

1. 配置/etc/exports  
   Ubuntu nfs允许挂载的目录及权限在文件/etc/exports中进行了定义。  
   例如，我们要将根目录下的rootfs目录共享出来，那么我们需要在/etc/exports文件末尾添加如下一行：/rootfs \*(rw,sync,no\_root\_squash)  
   其中：/rootfs是要共享的目录，\*代表允许所有的网络段访问，rw是可读写权限,sync是资料同步写入内存和硬盘，no\_root\_squash是Ubuntu nfs客户端分享目录使用者的权限，如果客户端使用的是root用户，那么对于该共享目录而言，该客户端就具有root权限。
2. Ubuntu nfs重启服务  
   $sudo /etc/init.d/portmap restart  
   $sudo /etc/init.d/nfs-kernel-server restart
3. 测试Ubuntu nfs  
   此时可以运行以下命令来显示一下共享出来的目录：  
   $showmount -e  
   或者可以使用以下命令把它挂载在本地磁盘上，例如将/rootfs挂载到/mnt下：  
   $ sudo mount -t nfs localhost:/rootfs /mnt  
   可以运行df命令查看是否挂载成功。查看后可以使用以下命令卸载：  
   $ sudo umount /mnt

## 增加硬盘

1. 在VMware里点击菜单VM——>Settings

2. 在Hardware里面单击底下的Add按钮，然后选择Hard Disk——>Next——>Next——>Disk size(GB)是新增硬盘的大小——>Next——>Finish——>Ok. 这样在VMware的Devices里就多了一个Hard Disk 2。

3，接下来就是进入ubuntu，把新的硬盘mount进去了。

:~# fdisk -l

可以看到多了一个 /dev/sdb

下面是格式化成ext3文件系统

[root@sparkle-desktop](mailto:root@sparkle-desktop):~# mkfs -t ext3 /dev/sdb (需要使用超级用户)

[root@sparkle-desktop](mailto:root@sparkle-desktop):~# cd /home

[root@sparkle-desktop:/home](mailto:root@sparkle-desktop:/home)# mkdir work

[root@sparkle-desktop:/home](mailto:root@sparkle-desktop:/home)# mount /dev/sdb /datastore //把硬盘mount到work目录。

[root@sparkle-desktop:/home](mailto:root@sparkle-desktop:/home)# cd work

[root@sparkle-desktop:/home/work](mailto:root@sparkle-desktop:/home/work)# ls

lost+found

用df命令查看一下挂载情况

[root@sparkle-desktop](mailto:root@sparkle-desktop):~# df

实现启动自动挂载。

[root@sparkle-desktop](mailto:root@sparkle-desktop):~# gedit /etc/fstab

根据下面的格式

# <file system><mount point><type><options><dump><pass>

我们自己添加：

/dev/sdb /home/work ext3 defaults 0 0

重启，这样就完成了增加硬盘的全部步骤了。

## 安装Ruby

### 方法一：通过rbenv安装Ruby和bosh cli

1.安装 Ruby 的依赖项：

$ sudo apt-get install git-core build-essential libsqlite3-devcurl libmysqlclient-dev libxml2-dev libxslt-dev libpq-dev genisoimage

2.获取最新版本的rbenv

$ git clone git://github.com/sstephenson/rbenv.git .rbenv

3.将 ~/.rbenv/bin 添加到您的 $PATH 以便能够访问rbenv命令行实用程序

$ echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bash\_profile

4.将rbenvinit添加到您的 shell 以启用填充程序 (Shim) 和自动完成

$ echo 'eval "$(rbenv init -)"' >> ~/.bash\_profile

5.下载 Ruby 1.9.2注意：您也可以使用适用于 rbenv 的 ruby-build 插件来构建 ruby。请参见https://github.com/sstephenson/ruby-build

$ wget<http://ftp.ruby-lang.org/pub/ruby/1.9/ruby-1.9.2-p290.tar.gz>

$ wget<http://ftp.ruby-lang.org/pub/ruby/1.9/ruby-1.9.3-p392.tar.gz>

6.将 Ruby 解包并安装 (使用 1.9.2-p290版本的ruby)

$ tar xvfz ruby-1.9.2-p290.tar.gz

$ cd ruby-1.9.2-p290

$ ./configure --prefix=$HOME/.rbenv/versions/1.9.3-p392

$ make

$ make install

7.重新启动您的 shell 以使路径更改生效

$ source ~/.bash\_profile

8.将您的默认 Ruby 设置为 1.9.2 版本

$ rbenv global 1.9.3-p392

安装libyaml

$ wget http://pyyaml.org/download/libyaml/yaml-0.1.4.tar.gz

$ tar xzvf yaml-0.1.4.tar.gz

$ cd yaml-0.1.4

$ ./configure --prefix=/usr/local

$ make

$ make install

cd ext/psych (搜psych所在的ruby目录)

ruby extconf.rb

make

make install

注意：使用此方法时可能需要重新安装 rake 0.8.7 gem$ gem pristine rake

9.更新rubygem并安装捆绑包。注意：安装 gem（gem install 或 bundle install）后，请运行rbenv rehash 以添加新的填充程序

$ rbenv rehash

$ gem update –-system

$ gem install bundler

$ rbenv rehash

1.安装 gerrit-cli gem：

$ gem install gerrit-cli

2.安装 BOSH CLI：

$ gem install bosh\_cli

$ rbenv rehash

$ bosh –version

如果一切运行顺利，最后一个命令将会显示您刚刚创建的 BOSH 版本。这表明 BOSH CLI 已安装成功。

### 方法二：使用rvm安装Ruby和bosh cli：

$ sudo apt-get install curl

\curl -L https://get.rvm.io | bash -s

在.bashrc中添加rvm的环境变量，否则退出终端后再进入，rvm不会被正常加载：

[[ -s "$HOME/.rvm/scripts/rvm" ]] && source "$HOME/.rvm/scripts/rvm"

$ source .bashrc

$ source ~/.bash-profile

输入下面指令来验证rvm是否安装成功：

$ type rvm |head -1

正确的返回结果：

rvm is a function

然后重启终端rvm就可以工作了。

执行$ rvm requirements

rvm的依赖包可以用过这条命令列出，经过测试，只要把这些依赖装好，之后的cli和bosh deploy gem的依赖，都已经自动加入了

对于那些未安装的依赖包，使用sudo apt-get install 方式安装。

执行rvm install 1.9.2 这个ruby的安装非常耗费时间，一般在半个小时左右

执行ruby -v 检测ruby版本

安装一些bosh cli所需的gem包：

$ gem install bundler

$ gem install rake

$ gem install gerrit-cli

$ gem install bosh\_cli

执行$ bosh --version看到版本号，说明bosh cli安装成功。

## BOSH 故障排除

### BOSH SSH

要对正在运行的作业运行 ssh，首先请查出该作业的名称和索引。使用 bosh vms 可显示正在运行的虚拟机的列表以及每个虚拟机上的作业。To ssh to it, run bosh ssh <job\_name><index>.密码是在 Stemcell 中设置的任意密码。对于默认 Stemcell，密码为 cloudc0w。

### BOSH 日志

对 BOSH 或 BOSH 部署进行故障排除时，务必要阅读日志文件以便可以缩小问题范围。有以下三种类型的日志。

1. 可通过 bosh task <task\_number> 查看的 BOSH 控制器日志

这些日志包含 BOSH 控制器上运行 BOSH 命令时产生的输出。如果在运行 BOSH 命令时出现问题，您应该先从这些日志着手排查。例如，如果您运行 bosh deploy，且此命令失败，那么 BOSH 控制器将有记录了出错位置的日志。要访问这些日志，请通过运行 bosh tasks recent 找到失败的命令的任务编号。然后，运行 bosh task <task\_number>。控制器的日志记录程序会将相关信息写入到这些日志中。

1. 可在 /var/vcap/bosh/log 中找到或通过 bosh logs 查看的代理日志

这些日志包含代理产生的输出。当怀疑虚拟机设置存在问题时，这些日志便很有用。它们将显示代理的操作，如设置网络、磁盘和运行作业。如果 bosh deploy 因其中一个虚拟机出现问题而失败，您需要使用 BOSH 控制器日志才能查出是哪个虚拟机。Then, either ssh and access /var/vcap/bosh/log or use bosh logs <job\_name><index> --agent.

1. 服务日志

这些日志由虚拟机上运行的实际作业产生，可能是由 Redis、Web 服务器等产生的日志。这些日志间会有差别，因为它们输出到的位置由部署来配置。通常，具体的输出路径在release/jobs/<job\_name>/templates/<config\_file> 中的配置文件中定义。对于 Cloud Foundry，我们的作业通常配置为记录到 /var/vcap/sys/log/<job\_name>/<job\_name>.log。These logs can also be accessed via bosh logs <job\_name><index>.

### BOSH 云检查

BOSH 云检查是一项 BOSH 命令行实用程序，可自动检查已经部署的虚拟机和作业中是否存在问题。它会检查是否存在虚拟机不响应/不同步、磁盘未绑定等情况。要使用它，请运行 bosh cck，如果发现了任何问题，它会提示您需采取哪些操作。

运行bosh cloudcheck具有和运行bosh cck相同的效果。

## BOSH CLI的网络

### 修改网卡名称

新网卡的名称是eth1，因为eth0被原有的网卡占用。如果想将新网卡设置为eth0，则需要进行以下步骤。

1）ifconfig eth0和ifconfig eth1查看网卡的MAC地址并记录，MAC地址主要用来区分两块网卡；

2）sudo vi /etc/udev/rules.d/70-persistent-net.rules

3） 文件内容：其中，NAME="eth0"就是网卡的名称，该名称与MAC地址"00:1a:a0:9d:f2:58"对应。只要更改eth0为eth1，就可以将MAC地址为"00:1a:a0:9d:f2:58"的网卡从eth0变成eth1.

4）更改完后重启机器。

如果无法分清百兆网卡和千兆网卡所对应的eth\*，可以使用ethtool eth0和ethtool eth1查看。

对网络进行设置，可以编辑vi /etc/network/interfaces,类似与下

设置完后重启网络

/etc/init.d/networking restart

### 设置双网卡

设置内网的ip地址时，去掉网关设置。

### 网关

涉及三个文件：

/etc/network/interfaces

/proc/sys/net/ipv4/ip\_forward

/etc/rc.local

/proc/sys/net/ipv4/ip\_forward

起到一个开关的作用，一般用“echo 1 > /proc/sys/net/ipv4/ip\_forward”修改它，这条命令的意思是把ip\_forward的内容设为“1”，表示开启路由转功能，若为“0”，则表示关闭路由转发功能。若把Ubuntu做为网关，则需要开启路由转发功能。

/etc/rc.local

开机启动脚本，可以把linux命令写到它里边，表示开机时自动执行该命令。

开始把Ubuntu配置成路由器：

1、正确配置IP，若你有两张网卡，则网卡代号分别是eth0、eth1。把对外的那张网卡的获得IP方式配置成static方式；对内的那张网卡的 IP设置成static方式，并给它分配一个IP，不填网关地址。

2、启动Ubuntu的路由转发功能，终端模式下输入：

echo 1 > /proc/sys/net/ipv4/ip\_forward（用下面命令代替）

sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip\_forward"

3、设置NAT规则，终端模式下：

sudo/sbin/iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j MASQUERADE

注意：这里的“192.168.1.0/24”是你的对内网卡所在的网络号。“eth0”是你的对外的网卡的网卡代号。根据自己配置的修改。

这样之后其他使用改网关的机器应该能上网了，然后再设置开机自动启动，把

|  |
| --- |
| sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip\_forward"  /sbin/iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j MASQUERADE |

这两条命令添加进/etc/rc.local文件中

下面是客户机的配置：

ip设为内网ip，也就是BOSH CLI机器上对内的那张网卡的ip所在的网段，网关设为BOSH CLI机器上对内的那张网卡的ip。以现在配置为例，就是：

IP地址：192.168.1.11

子网掩码：255.255.255.0

网关：192.168.1.11

首选DNS服务器：202.102.154.3

## shell脚本文件

我们可以使用任意一种文字编辑器，比如gedit、kedit、emacs、vi等来编写shell脚本，它必须以如下行开始（必须放在文件的第一行）：

#!/bin/bash...

注意：最好使用“!/bin/bash”而不是“!/bin/sh”，如果使用tc shell改为tcsh，其他类似。符号#!用来告诉系统执行该脚本的程序，本例使用/bin/bash。编辑结束并保存后，如果要执行该脚本，必须先使其可执行：

chmod +x filename

此后在该脚本所在目录下，输入 ./filename 即可执行该脚本。

## Github

>Permission denied (publickey).

>fatal: The remote end hung up unexpectedly

由于没有在github上提供公钥，执行命令ssh –vT [git@github.com](mailto:git@github.com)最后提示

debug1: No more authentication methods to try.

Permission denied (publickey).

解决方式：

生成密钥：ssh-keygen–t rsa –C “登录git的email”

添加密钥：使用git账号登录到git，选择account setting🡪ssh keys,把.ssh/id\_rsa.pub中的内容原样复制到git中保存即可

## Ubuntu

### 修改用户密码

I forgot my Ubuntu password so I booted into recovery and dropped into a root shell prompt and this is what happened:

root@username-PC:~# passwd username

Enter new UNIX password:

Retype new UNIX password:

passwd: Authentication token manipulation error

passwd: password unchanged

Also make sure you are mounting the file system read/write.

After immediately selecting 'Drop into root shell prompt' I found the filesystem was mounted read only, which prevents resetting the password.

Choosing the option to remount / as read/write and going back into the root shell prompt enabled the password change.

The command to run prior to changing the password is: mount -rw -o remount /

### 增加apt-get的镜像资源服务器

以Precise(12.04)为例, 编辑/etc/apt/sources.list文件, 在文件最前面添加以下条目(操作前请做好相应备份)

deb http://mirrors.163.com/ubuntu/ precise main restricted universe multiverse

deb http://mirrors.163.com/ubuntu/ precise-security main restricted universe multiverse

deb http://mirrors.163.com/ubuntu/ precise-updates main restricted universe multiverse

deb http://mirrors.163.com/ubuntu/ precise-proposed main restricted universe multiverse

deb http://mirrors.163.com/ubuntu/ precise-backports main restricted universe multiverse

deb-src http://mirrors.163.com/ubuntu/ precise main restricted universe multiverse

deb-src http://mirrors.163.com/ubuntu/ precise-security main restricted universe multiverse

deb-src http://mirrors.163.com/ubuntu/ precise-updates main restricted universe multiverse

deb-src http://mirrors.163.com/ubuntu/ precise-proposed main restricted universe multiverse

deb-src http://mirrors.163.com/ubuntu/ precise-backports main restricted universe multiverse

### 安装JDK

http://www.mkyong.com/java/how-to-install-java-jdk-on-ubuntu-linux/

**1. Find Openjdk**

Ubuntu is using OpenJDK, in terminal, issue this command opt-cache search openjdk to search all the available Openjdk version.

**2. Install Openjdk**

Issue command apt-get install openjdk-7-jdk to install JDK7. Ubuntu will auto download JDK and start the installation, wait a few minutes for the downloading process.

**3. Verification**

Ubuntu installs JDK at /usr/lib/jvm/jdk-folder, for example /usr/lib/jvm/java-7-openjdk-amd64/. In additional, Ubuntu also puts the JDK bin folder in the system path, via symbolic link. For example, /usr/bin/java.

To verify if JDK is installed properly, type java -version.

**4. Post-Installation Setup**

This step is optional. To configured JAVA\_HOME in system path each time the terminal is started, you can append the export JAVA\_HOME to /etc/profile or .bashrc.

|  |
| --- |
| echo$JAVA\_HOME  //show nothing    $sudoecho"export JAVA\_HOME=\"/usr/lib/jvm/java-7-openjdk-amd64/\"">> ~/.bashrc  //restart terminal    echo$JAVA\_HOME  /usr/lib/jvm/java-7-openjdk-amd64/ |

## vCenter序列号

安装vCenter时，序列号的生成器选项里选‘vCenter Server 5 Standard’

## 配置vSphere嵌套虚拟化

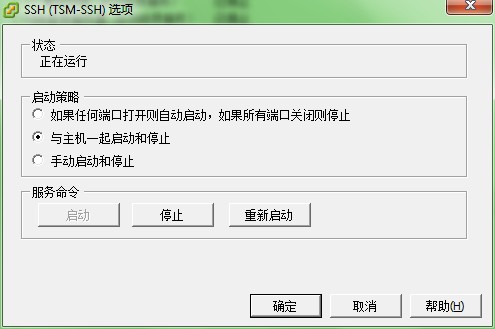
在一个vSphere服务器测试另一个服务器虚拟化产品，即将hypervisor部署到一个已经安装VMware ESXi的环境上，这里就涉及到为vSphere开启虚拟化嵌套的问题；

这个问题对应不同的hypervisor也有不同的表现，我曾经在一个已有的vSphere的服务器测试Oracle VM Server（OVS），实际上安装和配置OVS都不会有问题，但是如果在OVS上再创建一个Guest VM就会失败。

解决这个问题就需要为ESXi配置嵌套虚拟化，这里我们使用的是vSphere Client工具，具体操作如下：

1 . 启用ESXi主机的SSH功能

这一步是前提，在vSphere Client主界面中依次找到：配置 => 安全配置文件 => 服务 => 属性，修改SSH选项为“与主机一起启动或停止”

[](https://zziuew.bn1.livefilestore.com/y1pep2O4EjTANYKOwcOw08jJ4wLi-pZQ7nwwYxIjPojUsSNZS4JsU43H5Q_Osig-GBBPIg9SrORTec/ssh%E9%80%89%E9%A1%B9.jpg)

2 . 修改ESXi配置文件

SSH登录ESXi主机，执行操作：

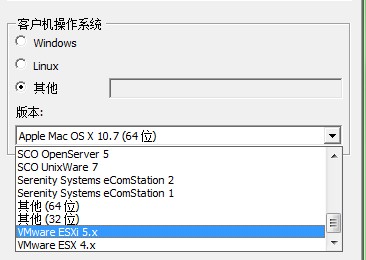
ESXi 5.0

|  |
| --- |
| echo vhv.allow= \"TRUE\" >> /etc/vmware/config |

ESXi 5.1

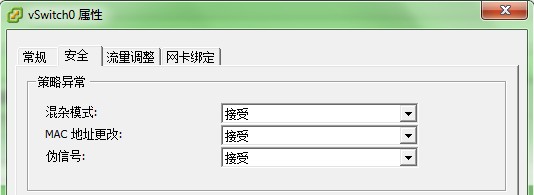
|  |
| --- |
| echo vhv.enable = \"TRUE\" >> /etc/vmware/config |

3 . 修改hypervisor宿主机操作系统类型



这样就可以保证在ESXi上可以成功安装hypervisor了。

4.虚拟机中创建的虚拟机如果要和外面通信，需要把虚拟交换机开通混杂模式，即promiscous模式，如下，把“拒绝”改成“接受”(Accept)；这样在这些虚拟机上的虚拟机才能通信



## Maven

D:\Server\apache-maven-3.0.3-bin\bin\mvn install:install-file -Dfile=gchart-2.7.jar -DgroupId=com.googlecode.gchart -DartifactId=gchart -Dversion=2.7 -Dpackaging=jar -DgeneratePom=true -DcreateChecksum=true

D:\Server\apache-maven-3.0.3-bin\bin\mvn install:install-file -Dfile=cloudfoundry-identity-common-1.2.0.jar -DgroupId=org.cloudfoundry.identity -DartifactId=cloudfoundry-identity-common -Dversion=1.2 -Dpackaging=jar -DgeneratePom=true -DcreateChecksum=true

## Ruby

sudo gem install vmc -v 0.3.18 0.4.0

RUBY GEM如何打包

RubyGems是一个库和程序的标准化打包以及安装框架，它使定位、安装、升级和卸载Ruby包变的很容易。rails以及它的大部分插件都是以gem形式发布的。下面简单描述创建ruby Gems的过程的可行性。

第一步，写Gems规范文件

如：编写xx..gemspec ：

第二步：添加测试和文档(非必要)

第三步：生成gem包

gem build xx..gemspec

将生成一个gem文件，将这个文件共享只要下载这个文件，执行：

gem install xx.0.01.gem

列出所有已安装的gem命令：

gem query --local

安装方法：下载.gem文件，用cd到.gem的所在目录，然后执行下面的命令：

gem install --local filename.gem

或者执行命令的时候带有.gem文件的路径名。

gem install --local F:/ruby/gem/filename.gem

**3，增加其它下载资源：**

先通过gem env 查看gem运行环境

|  |
| --- |
| gem env |

增加两个更新的源地址：

|  |
| --- |
| gem sources -a http://gems.github.com  gem sources -a [http://gems.rubyforge.org](http://gems.rubyforge.org/)  $ gem sources --remove https://rubygems.org/  $ gem sources -a http://ruby.taobao.org/  $ gem sources -l  保持gem 的source中只有一个源服务器。 |

## micro\_bosh.yml

micro.yml文件说明

文件内容中不要出现 tab 的缩进符，否则会影响部署。

micro bosh的ip地址不要设成和micro cli一样的IP地址，否则会出现wait agent问题。

在micro.yml文件中设置vcenter ip地址时，有时要加上9443端口号，否则会连不上。

micro.yml配置文件中的clusters参数值一定要加上。

|  |
| --- |
| ---  name: micro01 #the same name as your folder name  #XXX using micro01  network:  ip: 192.168.2.21 #define ip for micro BOSH  netmask: 255.255.255.0  gateway: 192.168.2.1  dns:  - 192.168.2.1  - 202.102.154.3  cloud\_properties:  name: VM Network #this should be the same as the “Network” of all your virtual machines  resources:  persistent\_disk: 8000  cloud\_properties:  ram: 528  disk: 8000  cpu: 1  cloud:  plugin: vsphere  properties:  agent:  ntp:  - time.windows.com  - ntp02.las01.emcatmos.com  vcenters:  - host: 192.168.2.103 # ip address of vCenter  user: Administrator #user of vCenter  password: dz+1  datacenters:  - name: NCF #name of the datacenter you created  vm\_folder: vm\_folder #folder for vms  template\_folder: template\_folder #folder for templates  disk\_path: boshdeployer #disk path where VMs are stored.  datastore\_pattern: NFSdatastore #name of a datastore or some regular expression matching the datastore name.  persistent\_datastore\_pattern: NFSdatastore #name of a datastore or some regular expression matching the persistent datastore  allow\_mixed\_datastores: true #true if two datastore patterns are the same  clusters:  - CF #name of your cluster in your datacenter  #resource\_pool: <resource\_pool\_name> # Resource pool is optional you can run without a resource pool. If so, remove this line.  apply\_spec:  agent:  blobstore:  address: 192.168.2.21 #change to micro bosh ip defined above  nats:  address: 192.168.2.21 #change to micro bosh ip defined above  properties: {} |

## bosh\_manifest.yml

|  |
| --- |
| ---  name: dev124  director\_uuid: 516f1338-9fb7-45fd-a163-adee0edc4107 # Change this to the UUID of the BOSH director.  release: # Find this with > bosh status.  name: bosh # Change this to reflect the name of your release. This is specified in your release manifest.  version: 11 # Specify the version of your release. This is also specified in your release manifest.  networks:  - name: default # An internal name for the network in your manifest file  subnets:  - reserved:  # IP addresses which you don't want allocated by BOSH  - 192.168.2.2 - 192.168.2.10  - 192.168.2.12 - 192.168.2.30  - 192.168.2.51 - 192.168.2.254  static:  # A static IP range that can be used for deploying BOSH  - 192.168.2.31 - 192.168.2.36  range: 192.168.2.0/24 # The range of IP addresses that can be accessible by BOSH  gateway: 192.168.2.1  dns:  - 192.168.2.1  - 202.102.154.3  cloud\_properties:  name: "CF Network" # The name of the virtual machine network in vCenter  #  # You shouldn't have to change any of the resource pool parameters  #  resource\_pools:  - name: small  stemcell:  name: bosh-stemcell  version: 0.6.7  network: default  size: 5  cloud\_properties:  ram: 512  disk: 2048  cpu: 1  - name: director  stemcell:  name: bosh-stemcell  version: 0.6.7  network: default  size: 1  cloud\_properties:  ram: 1048  disk: 8192  cpu: 2  compilation:  workers: 4  network: default  cloud\_properties:  ram: 1048  disk: 4048  cpu: 4  #  # If you have errors with canary instances during a deployment, you can try  # increasing the canary\_watch\_time and update\_watch\_time. The values here are in  # milliseconds. Divide by 1000 to get the value in seconds.  #  #  update:  canaries: 1  canary\_watch\_time: 60000  update\_watch\_time: 60000  max\_in\_flight: 1  max\_errors: 2  jobs:  - name: nats  template: nats  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.31 # Assign a static IP  - name: postgres  template: postgres  instances: 1  resource\_pool: small  persistent\_disk: 2048  networks:  - name: default  static\_ips:  - 192.168.2.32 # Assign a static IP from the static range specified above  - name: redis  template: redis  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.33 # Assign a static IP from the static range specified above  - name: director  template: director  instances: 1  resource\_pool: director  persistent\_disk: 2048  networks:  - name: default  static\_ips:  - 192.168.2.34 # Assign a static IP from the static range specified above  - name: blobstore  template: blobstore  instances: 1  resource\_pool: small  persistent\_disk: 20480  networks:  - name: default  static\_ips:  - 192.168.2.35 # Assign a static IP from the static range specified above  - name: health\_monitor  template: health\_monitor  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.36 # Assign a static IP from the static range specified above  properties:  env:  blobstore:  address: 192.168.2.35 # This should be the same IP specified for the job above  port: 25251  backend\_port: 25552  agent:  user: agent  password: 38fh20  director:  user: director  password: oe9U37dbd  networks:  apps: default  management: default  nats:  user: nats  password: 38dhj389ddck378gcE  address: 192.168.2.31 # This should be the same IP specified for the job above  port: 4222  postgres:  user: bosh  password: 29fhcuebJDldhv265kB  address: 192.168.2.32 # This should be the same IP specified for the job above  port: 5432  database: bosh  redis:  address: 192.168.2.33 # This should be the same IP specified for the job above  port: 25255  password: wicjew93hf2  director:  name: dev124  address: 192.168.2.34 # This should be the same IP specified for the job above  port: 25555  hm:  http:  port: 25923  user: admin  password: 8wufn  director\_account:  user: admin  password: 8wufn  intervals:  poll\_director: 60  poll\_grace\_period: 30  log\_stats: 300  analyze\_agents: 60  agent\_timeout: 180  rogue\_agent\_alert: 180  loglevel: info  email\_notifications: false  email\_recipients:  - noemail@yourdomain.com  smtp:  from: bhm@localhost.localdomain  host: smtp.yourdomain.com  port: 25  auth: plain  user: appcloud  password: 38fhsoeY  domain: localdomain  tsdb\_enabled: false # If this option is set to false, comment out its params.  #tsdb:  # address: 10.1.4.228  # port: 4242  vcenter:  address: 192.168.2.103 # The IP of your vCenter server  user: Administrator # Your vCenter server user  password: dz+1 # You vCenter user password  datacenters:  - name: NCF # The datacenter name in vCenter  vm\_folder: vm\_folder\_bosh #folder to save vm, created in vCenter.  template\_folder: template\_folder\_bosh  disk\_path: boshdeployer\_bosh #disk path, created in the datastore in vCenter  datastore\_pattern: NFSdatastore  persistent\_datastore\_pattern: NFSdatastore  allow\_mixed\_datastores: true  clusters:  - CF |

## cf-lab-1.yml

|  |
| --- |
| ---  name: cf.local  director\_uuid: 80c1db30-93b8-4059-9ca9-3380ef3df06f  release:  name: cloudfoundry  version: 130.3-dev  compilation:  workers: 2  reuse\_compilation\_vms: true  network: default  cloud\_properties:  ram: 2096  disk: 8096  cpu: 2  update:  canaries: 1  canary\_watch\_time: 3000-90000  update\_watch\_time: 3000-90000  max\_in\_flight: 1  max\_errors: 1  networks:  - name: default  subnets:  - range: 192.168.2.0/24  reserved:  - 192.168.2.2 - 192.168.2.10  - 192.168.2.21 - 192.168.2.50  - 192.168.2.160 - 192.168.2.254  static:  - 192.168.2.51 - 192.168.2.100  gateway: 192.168.2.1  dns:  - 192.168.2.1  - 202.102.154.3  cloud\_properties:  name: "CF Network"  resource\_pools:  - name: small  network: default  size: 18  stemcell:  name: bosh-stemcell  version: 0.6.7  cloud\_properties:  ram: 624  disk: 8192  cpu: 1  env:  bosh:  password: $6$gZBmUQH4T$MY9zOP4rRDD/.62MOfLy/ZILHdUpyx1DnMqKDP429Qhbo5O9Nfj18Z2gk8sCFnJDKRMEEzgCnXFXamODQvH4x0  # c1oudc0w - password generated using mkpasswd -m sha-512  - name: medium  network: default  size: 4  stemcell:  name: bosh-stemcell  version: 0.6.7  cloud\_properties:  ram: 1048  disk: 16384  cpu: 2  env:  bosh:  password: $6$gZBmUQH4T$MY9zOP4rRDD/.62MOfLy/ZILHdUpyx1DnMqKDP429Qhbo5O9Nfj18Z2gk8sCFnJDKRMEEzgCnXFXamODQvH4x0  # c1oudc0w - password generated using mkpasswd -m sha-512  - name: router  network: default  size: 2  stemcell:  name: bosh-stemcell  version: 0.6.7  cloud\_properties:  ram: 512  disk: 8192  cpu: 1  env:  bosh:  password: $6$gZBmUQH4T$MY9zOP4rRDD/.62MOfLy/ZILHdUpyx1DnMqKDP429Qhbo5O9Nfj18Z2gk8sCFnJDKRMEEzgCnXFXamODQvH4x0  # c1oudc0w - password generated using mkpasswd -m sha-512  jobs:  - name: debian\_nfs\_server  template: debian\_nfs\_server  instances: 1  resource\_pool: medium  persistent\_disk: 16384  networks:  - name: default  static\_ips:  - 192.168.2.51  - name: services\_nfs  template: debian\_nfs\_server  instances: 1  resource\_pool: small  persistent\_disk: 8192  properties:  debian\_nfs\_server:  no\_root\_squash: true  networks:  - name: default  static\_ips:  - 192.168.2.64 # Enter a static IP  - name: nats  template: nats  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.52  - name: ccdb  template: postgres  properties:  db: ccdb  instances: 1  resource\_pool: small  persistent\_disk: 8192  networks:  - name: default  static\_ips:  - 192.168.2.53  - name: login  template: login  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.79  - name: uaadb  template: postgres  properties:  db: uaadb  instances: 1  resource\_pool: small  persistent\_disk: 8192  networks:  - name: default  static\_ips:  - 192.168.2.54  - name: vcap\_redis  template: vcap\_redis  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.55  - name: services\_redis  template: vcap\_redis  instances: 1  resource\_pool: small  persistent\_disk: 2048  networks:  - name: default  static\_ips:  - 192.168.2.69 # Enter a static IP.  properties:  vcap\_redis:  port: 3456  password: passwordpassword01  maxmemory: 2000000000  persistence:  dir: /var/vcap/store/vcap\_redis  - name: uaa  template: uaa  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.56  - name: acmdb  template: postgres  properties:  db: acmdb  instances: 1  resource\_pool: small  persistent\_disk: 8192  networks:  - name: default  static\_ips:  - 192.168.2.57 # Enter a static IP.  - name: cloud\_controller  template: cloud\_controller  instances: 1  resource\_pool: medium  networks:  - name: default  static\_ips:  - 192.168.2.59  - name: stager  template: stager  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.60  - name: router  template: router  instances: 2  resource\_pool: router  networks:  - name: default  static\_ips:  - 192.168.2.61  - 192.168.2.62  default: [dns, gateway]  #- name: lb  # static\_ips:  # - 192.168.0.2 - 192.168.0.2  - name: health\_manager  template: health\_manager\_next  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.63  - name: dea  template: dea  instances: 2  resource\_pool: medium  networks:  - name: default  static\_ips:  - 192.168.2.65  - 192.168.2.66  - name: redis\_node  template: redis\_node  instances: 1  resource\_pool: small  persistent\_disk: 2048  networks:  - name: default  static\_ips:  - 192.168.2.71 # Enter a static IP for each instance.  - name: redis\_gateway  template: redis\_gateway  instances: 1  resource\_pool: small  networks:  - name: default # No static IP required here  static\_ips:  - 192.168.2.72  - name: postgresql\_node  template: postgresql\_node  instances: 1  resource\_pool: small  persistent\_disk: 2048  networks:  - name: default  static\_ips:  - 192.168.2.75 # Enter a static IP for each instance.  - name: postgresql\_gateway  template: postgresql\_gateway  instances: 1  resource\_pool: small  networks:  - name: default # No static IP required here  static\_ips:  - 192.168.2.76  - name: vblob\_node  template: vblob\_node  instances: 1  resource\_pool: small  persistent\_disk: 1024  networks:  - name: default  static\_ips:  - 192.168.2.77 # Enter a static IP.  - name: vblob\_gateway  template: vblob\_gateway  instances: 1  resource\_pool: small  networks:  - name: default # No static IP required here  static\_ips:  - 192.168.2.78  - name: service\_utilities  template: service\_utilities  instances: 1  resource\_pool: small  networks:  - name: default  static\_ips:  - 192.168.2.80 # Enter a static IP  properties:  domain: cf.local  env:  networks:  apps: default  management: default  nats:  user: root  password: passwordpassword01  address: 192.168.2.52  port: 4222  ccdb:  address: 192.168.2.53  port: 5524  pool\_size: 10  roles:  - tag: admin  name: root  password: passwordpassword01  databases:  - tag: cc  name: appcloud  uaadb:  address: 192.168.2.54  port: 2544  roles:  - tag: admin  name: root  password: passwordpassword01  databases:  - tag: uaa  name: uaa  acmdb:  address: 192.168.2.57  port: 2544  roles:  - tag: admin  name: root  password: passwordpassword01  databases:  - tag: acm  name: acm  cc:  srv\_api\_uri: http://api.cf.local  password: passwordpassword01  token: aaaf4eaa8c1758f66d5cb7adcb24adb9d7  use\_nginx: true  new\_stager\_percent: 100  new\_stager\_email\_regexp: '.\*@vmware.com'  staging\_upload\_user: root  staging\_upload\_password: passwordpassword01  allow\_registration: true  logging\_level: debug  uaa:  enabled: true  resource\_id: cloud\_controller  token\_creation\_email\_filter: vmware.com  admins:  - admin1@vmware.com  - admin2@vmware.com  vcap\_redis:  address: 192.168.2.55  port: 5454  password: passwordpassword01  maxmemory: 2000000000 # 2GB  router:  status:  port: 8080  user: root  password: passwordpassword01  dea:  max\_memory: 3096  dea\_next:  memory\_mb: 4192  memory\_overcommit\_factor: 4  disk\_mb: 16384  disk\_overcommit\_factor: 4  num\_instances: 256  nfs\_server:  address: 192.168.2.64  network: 192.168.2.0/24  service\_plans:  mysql:  free:  job\_management:  high\_water: 1400  low\_water: 100  configuration:  allow\_over\_provisioning: true  capacity: 200  max\_db\_size: 128  max\_long\_query: 3  max\_long\_tx: 0  max\_clients: 20  postgresql:  free:  job\_management:  high\_water: 1400  low\_water: 100  configuration:  capacity: 200  max\_db\_size: 128  max\_long\_query: 3  max\_long\_tx: 30  max\_clients: 20  redis:  free:  job\_management:  high\_water: 1400  low\_water: 100  configuration:  capacity: 200  max\_memory: 16  max\_swap: 32  max\_clients: 500  rabbit:  free:  job\_management:  low\_water: 100  high\_water: 1400  configuration:  max\_memory\_factor: 0.5  max\_clients: 512  capacity: 200  vblob:  free:  job\_management:  low\_water: 100  high\_water: 1400  configuration:  capacity: 200  service\_lifecycle:  download\_url: service-serialization.cf.local # Modify this url for your environment  mount\_point: /var/vcap/service\_lifecycle  tmp\_dir: /var/vcap/service\_lifecycle/tmp\_dir  resque:  host: 192.168.2.69 # Change this IP address  port: 3456  password: passwordpassword01 # Create a secure password  nfs\_server:  address: 192.168.2.64 # Change this IP address  export\_dir: /var/vcap/store/shared  redis\_gateway:  token: aaaf4eaa8c1758f66d5cb7adcb24adb9d7  check\_orphan\_interval: 7200  supported\_versions: ["2.2", "2.4", "2.6"]  version\_aliases:  current: "2.2"  next: "2.6"  redis\_node:  command\_rename\_prefix: sample  supported\_versions: ["2.2", "2.4", "2.6"]  default\_version: "2.2"  rabbit\_gateway:  token: aaaf4eaa8c1758f66d5cb7adcb24adb9d7  check\_orphan\_interval: 7200  supported\_versions: ["2.4"]  version\_aliases:  current: "2.4"  rabbit\_node:  supported\_versions: ["2.4"]  default\_version: "2.4"  postgresql\_gateway:  check\_orphan\_interval: 7200  token: aaaf4eaa8c1758f66d5cb7adcb24adb9d7  supported\_versions: ["9.0"]  version\_aliases:  current: "9.0"  postgresql\_node:  production: true  supported\_versions: ["9.0"]  default\_version: "9.0"  vblob\_gateway:  token: aaaf4eaa8c1758f66d5cb7adcb24adb9d7 # Change this to your own token string  acls:  wildcards: [\*@cf.local]  supported\_versions: ["0.51"]  version\_aliases:  current: "0.51"  vblob\_node:  vblobd\_auth: s3  nfs\_server:  address: 192.168.2.64 # Change this IP address. It should reference your services\_nfs server.  export\_dir: /var/vcap/store/shared  supported\_versions: ["0.51"]  default\_version: "0.51"  stager:  max\_staging\_duration: 120  max\_active\_tasks: 20  queues:  - staging  uaa:  no\_ssl: true  catalina\_opts: -Xmx768m -XX:MaxPermSize=256m  url: http://uaa.cf.local  resource\_id: account\_manager  client\_secret: somesecret  token\_secret: tokensecret  cc:  token\_secret: WGvbxaiC371JM  client\_secret: fOZF5DMNDZIfCb9A  admin:  client\_secret: MeM5fER8evgEisjj  login:  protocol: http  client\_secret: HZtd2FyZS5jb20iL  batch:  username: do0H2AKay2jAc  password: FSCTiplI5Q83n  client:  override: true  autoapprove:  - vmc  - login  clients:  scim:  users: []  login:  protocol: http  links:  home: http://portal.cf.local  passwd: http://portal.cf.local/password\_resets/new  signup: http://portal.cf.local/register |

## bosh清单条目介绍

**网络（networks）**

下面是网络一节的一个示例。

networks: #定义网络

- name:default

subnets:

- reserved: #您不希望分配的 IP

- 10.60.98.121 - 10.60.98.254

static: #您将使用的 IP

- 10.60.98.115 - 10.60.98.120

range: 10.60.98.0/24

gateway: 10.60.98.1

dns:

- 10.40.62.11

- 10.135.12.101

cloud\_properties:#与所有其他虚拟机都相同的网络。

name:VM Network

static：包含 BOSH 虚拟机的 IP 地址。

reserved：BOSH不应使用的 IP 地址。请务必要排除所有已经分配给同一网络中其他设备（例如存储设备、网络设备、Micro BOSH和 vCenter 主机）的 IP 地址。在安装期间，Micro BOSH可能会创建一些临时虚拟机（工作者虚拟机）来执行编译。如果我们不指定预留的地址，这些临时虚拟机可能会与现有设备或主机存在 IP 地址冲突。

cloud\_properties：name是我们在 vSphere 中定义的网络名称（见第 II 部分）。

**资源池（resource\_pools）**

此节定义作业使用的虚拟机配置（CPU、内存、磁盘和网络）。通常，应用程序的各个作业在资源使用方面各异。例如，有些作业需要较多的内存量，而有些作业则需要更多的vCPU来执行计算密集型任务。根据实际需要，我们应创建一个或多个资源池。需要注意的是，所有池的总规模应等于在清单文件中定义的作业实例的总数。部署 BOSH 时，由于总共有 6 个虚拟机（6 个作业），因此所有池的规模加起来应等于 6。

在我们的清单文件中，我们有 3 个资源池：

|  |  |  |  |
| --- | --- | --- | --- |
| **Resource Pool** | **Size** | **VM Configuration** | **Job** |
| small | 3 | RAM：512 MB；CPU：1 个；磁盘：2 GB | nats、redis、health\_monitor |
| medium | 2 | RAM：1 GB；CPU：1 个；磁盘：8 GB | postgres、blobstore |
| director | 1 | RAM：2 GB；CPU：2 个；磁盘：8 GB | director |

**编译（compilation）**

此节定义为编译包而创建的工作者虚拟机。在资源有限的系统中，我们应减少并发工作者虚拟机的数目，以确保编译成功。在我们的示例中，我们定义 4 个工作者虚拟机。

**更新（update）**

此节包含一个非常有用的参数：max\_in\_flight。此参数用于向 BOSH 告知最多可以并行安装的作业数。在运行速度缓慢的系统中，请设法减小此数目。如果将此数目设置为 1，则表示作业将按顺序部署。对于 BOSH 部署，我们建议将此数目设置为 1，以确保 BOSH 可以成功安装。

**作业（jobs）**

在 BOSH Release中有六个作业。每个作业占用一个虚拟机。根据作业的性质和资源使用情况，我们将作业分配给各个资源池。需要注意的一点是，我们需要向以下三个作业分配持久磁盘：postgres、director 和blobstore。若无持久磁盘，这些作业将无法正常运行，因为它们的本地磁盘很快就会被占满。

最好填写一张像下面这样的电子表格来对您的部署进行规划。您可以根据该电子表格修改部署清单。

|  |  |  |
| --- | --- | --- |
| **作业** | **资源池** | **IP** |
| nats | small | 10.60.98.120 |
| postgres | medium | 10.60.98.119 |
| redis | small | 10.60.98.118 |
| director | director | 10.60.98.117 |
| blob\_store | medium | 10.60.98.116 |
| health\_monitor | small | 10.60.98.115 |

我们基于上表创建了一个示例部署清单，您可以从这里下载：

<https://github.com/vmware-china-se/bosh_doc/blob/master/bosh.yml>

我们至少需要在这个配置文件中更改bosh director的UUID，release的名称和版本