

Deployment Engine Documentation

Version 1.0

Contents

Deployment Engine 特性.....	4
Deployment Engine 节点系统要求.....	5
Deployment Engine 安装与激活.....	6
Deployment Engine 安装.....	6
Deployment Engine 激活.....	6
Deployment Engine 场景描述.....	7
1、集群部署.....	7
2、组件状态监控.....	8
3、组件重启.....	8
4、集群扩展.....	9
5、停止组件.....	10
Deployment Engine API.....	11
获取配置文件.....	11
提交配置文件.....	14
执行部署方案.....	15
组件状态监控.....	16
请求节点硬件配置.....	18
获取日志文件列表.....	19
下载日志文件.....	20
重启节点.....	21

停止节点	22
Deployment engine 部署 cloud foundry	23
Trouble Shoot	26

Deployment Engine 特性

- 快速自动化部署一个完整的 cloudfoundry 集群，各组件并发安装
- 自动化部署单个 cloudfoundry 组件
- 可离线化的内网部署。各组件节点内核版本为 3.2.0-61-virtual，即可离线，只需要 Deployment Engine 控制机联网
- 物理机、虚拟机或容器皆可部署，剥离对 Iaas 层的依赖
- 支持高可用部署，无单点
- 方便管理和运维集群的所有虚拟机节点
- 对各虚拟机节点实时监控，包括各节点的硬件配置（虚拟机核数 vCPU、内存 RAM、硬盘 DISK）、资源使用量（CPU 使用率、内存占用率、磁盘使用率）、组件运行状况
- 提供集群运维日志的显示与下载
- 提供各节点运行日志的下载
- 兼容 cf-release 版本：v160、v170、v194

Deployment Engine 节点系统要求

- 操作系统: Ubuntu10.04 或 Ubuntu12.04 64 位
- Ruby 版本: Ruby 1.9.3p448
- Rubygem 版本: 1.8.17
- Yaml 版本: 0.1.4
- 监控软件: monit-5.2.4
- vCPU 核数: 1 个以上
- 内存大小: 1024MB 以上
- 磁盘大小: 5120MB 以上

Deployment Engine 安装与激活

Deployment Engine 安装

Step 1: 解压 Deployment-Engine.tgz 文件到任意目录下

Step 2: 解压文件后，运行 ./local_build.sh 进行 Deployment Engine 本地初始化，主要完成以下工作：

架设 linux 本地源：

部署 Apache 服务器

使缓存安装包可访问

修改本地源

构建本地运行时环境：

从 linux 本地源安装 linux 依赖

安装 ruby、rubygems 等文件

安装缓存的 ruby gems 包

Deployment Engine 激活

初次使用 Deployment Engine 需要输入购买的序列号进行激活操作。执行 ./DeploymentEngineServer，输入产品序列号，确保 Deployment Engine 所在的这台部署机联网，否则无法进行激活或者部署集群。

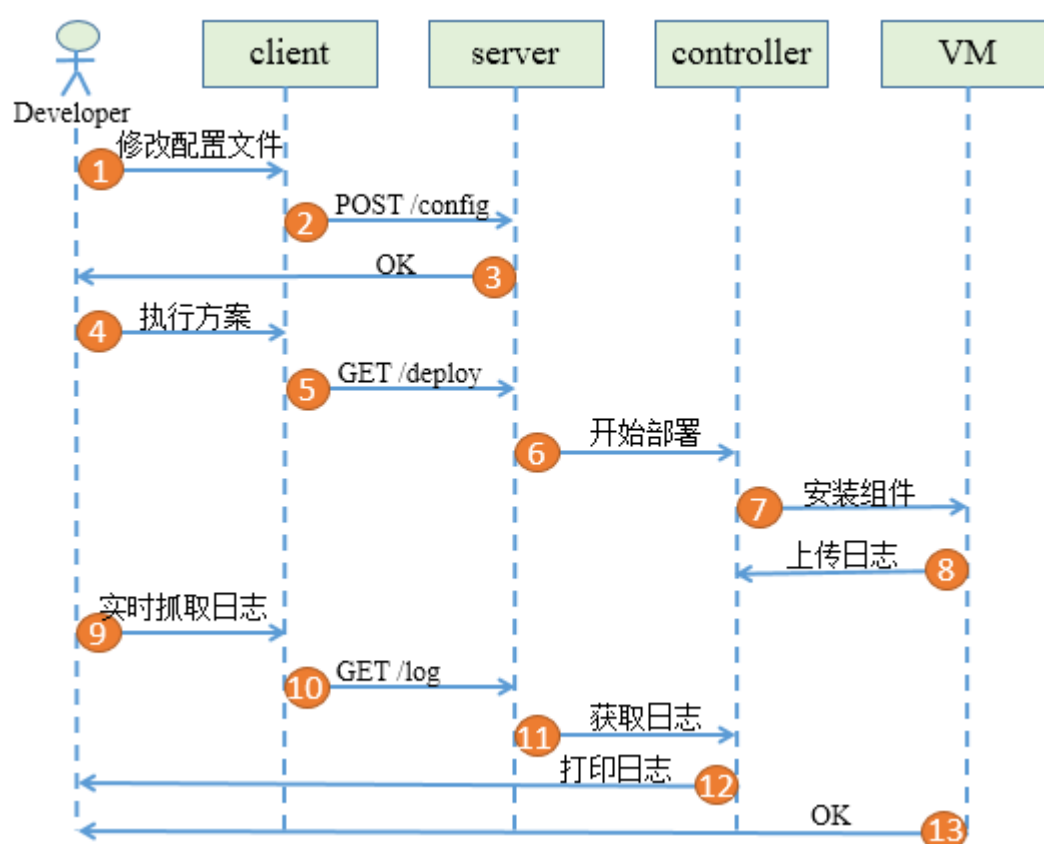
！注意：不要随意修改 Deployment Engine 下的文件，文件损坏可能会导致 Deployment Engine 无法工作；不要把 Deployment Engine 这台部署机复制到别的 VLAN 中进行集群部署，否则会导致两台部署机都无法工作。

Deployment Engine 场景描述

Deployment engine 场景描述

Deployment engine 主要包括集群部署、组件状态监控、组件重启、集群规模扩展、停止组件 5 大场景，每部分场景具体描述如下：

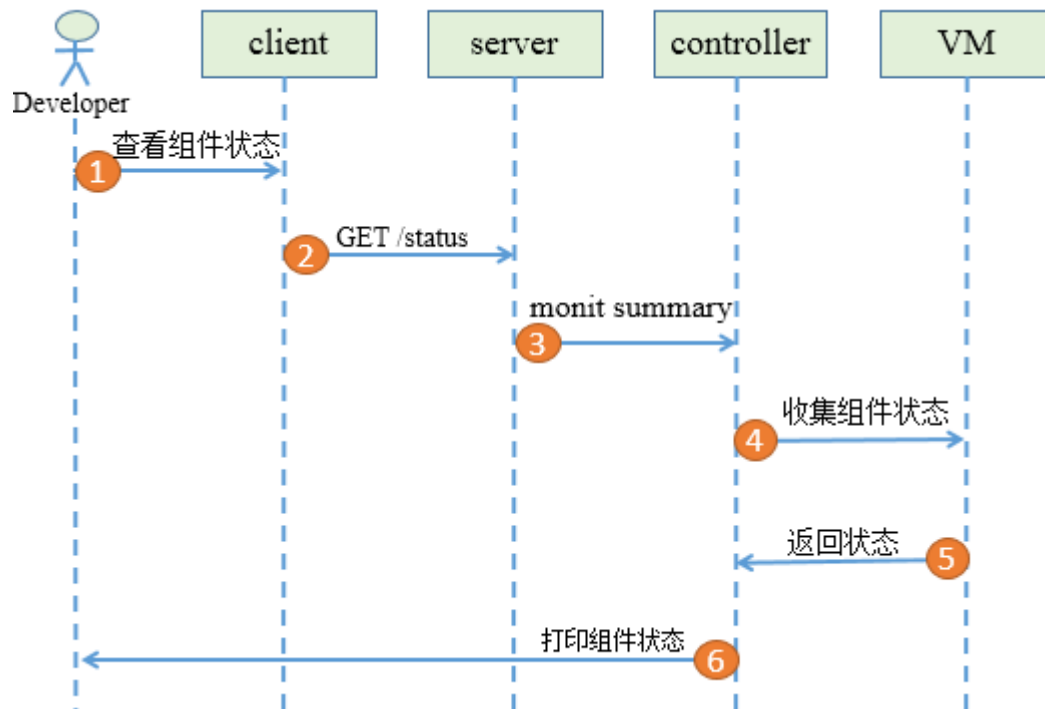
1、集群部署



- 1) 用户在客户端修改配置文件，包括集群域名、虚拟机密码、每个组件所在虚拟机的 IP 组等信息。
- 2) 客户端向 deployment engine 的服务端发送请求，发送配置信息。
- 3) 服务端检测配置信息的有效性，创建配置文件，写入配置信息，返回 OK 状态。
- 4) 集群配置完毕，执行部署方案。
- 5) 客户端向 deployment engine 发送部署请求。
- 6) Deployment engine 服务端收到部署请求，调用 deployment engine 核心模块 controller 执行部署任务。
- 7) Controller 模块根据配置文件向各个虚拟机安装组件。
- 8) Controller 收集各个虚拟机安装组件过程中的日志。

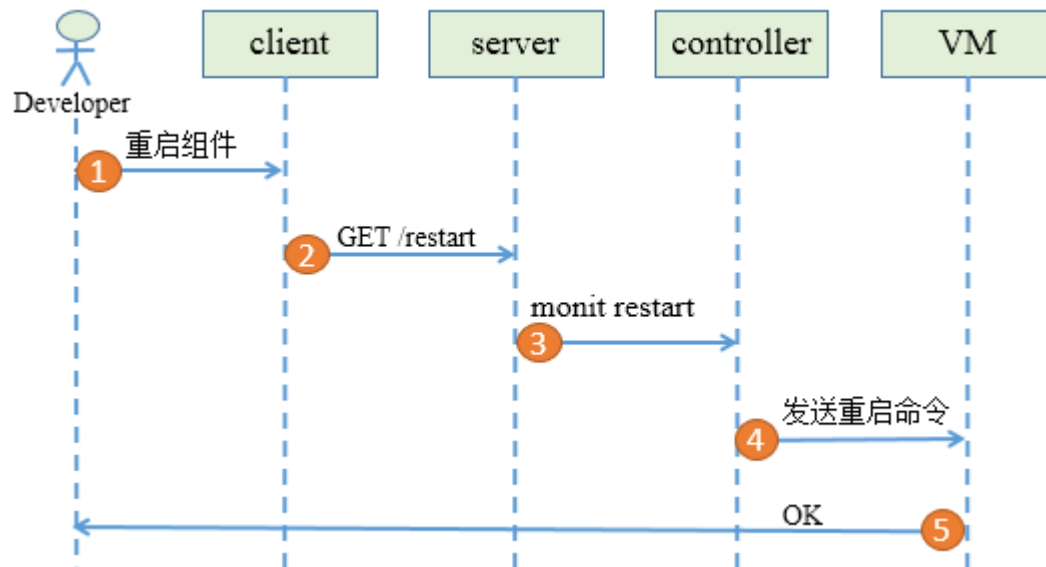
- 9) 用户需要实时看到部署过程中的日志信息。
- 10) 客户端发送获取日志请求。
- 11) 服务端通知 controller 上传收集到的日志。
- 12) 返回日志信息。
- 13) 集群部署完毕，返回 OK 状态。

2、组件状态监控



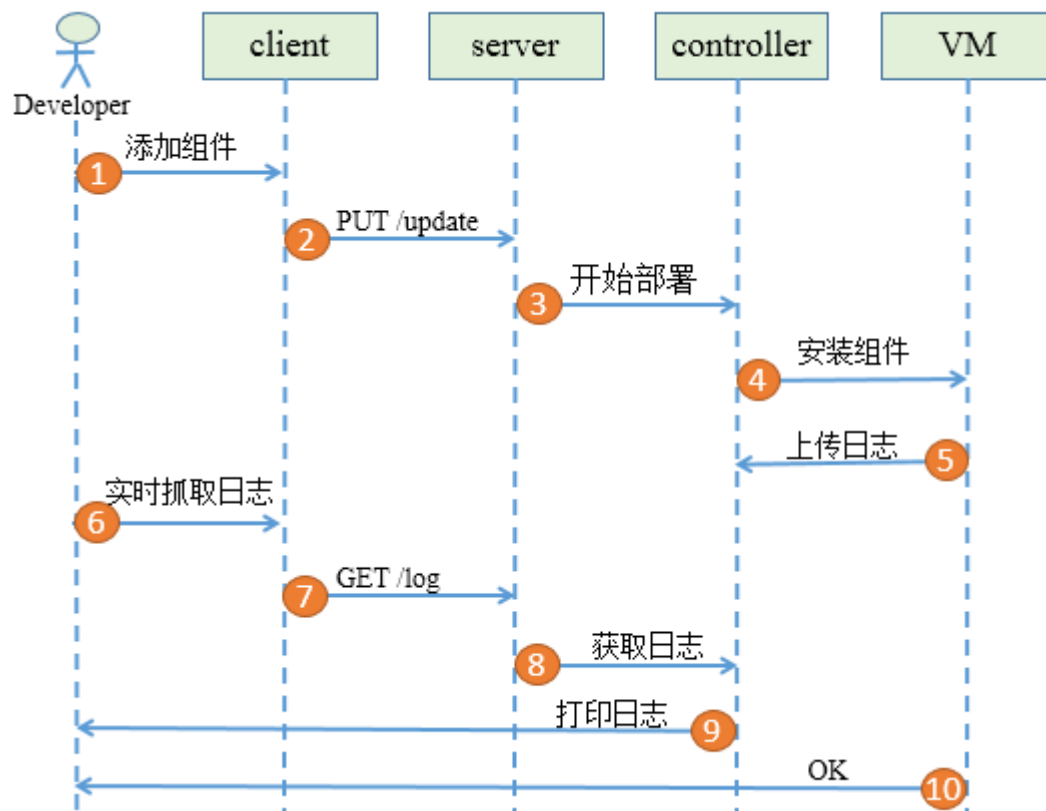
- 1) 用户在客户端点击按钮查看组件状态。
- 2) 客户端向 deployment engine 发送请求获取组件状态。
- 3) 服务端通知 controller 模块使用 monit summary 命令查询组件状态。
- 4) Controller 接受命令后收集所有组件状态。
- 5) 虚拟机返回组件状态告知 controller。
- 6) Deployment engine 向客户端请求进行响应，返回组件状态。

3、组件重启



- 1) 用户在客户端针对某个组件点击重启组件。
- 2) 客户端向 deployment engine 发送重启组件请求，包含重启组件的名称及 IP 等信息。
- 3) 服务端向 controller 发送 monit restart 命令。
- 4) Controller 根据组件名称及 IP 重启组件。
- 5) 返回 OK 状态。

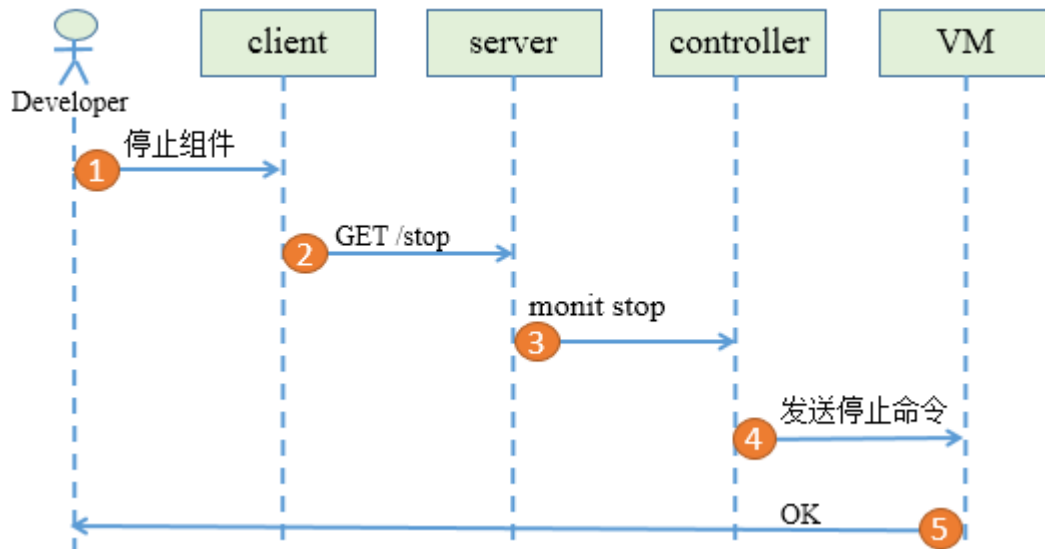
4、集群扩展



- 1) 用户在客户端填写要添加的组件名称和对应 IP。

- 2) 客户端发送扩展集群请求，包括组件名称及 IP 等数据。
- 3) 服务端通知 controller 部署新的节点。
- 4) Controller 按照组件及 IP 安装相对应的组件。
- 5) ~10)同集群部署中的 8) ~13)，这里不再累赘。

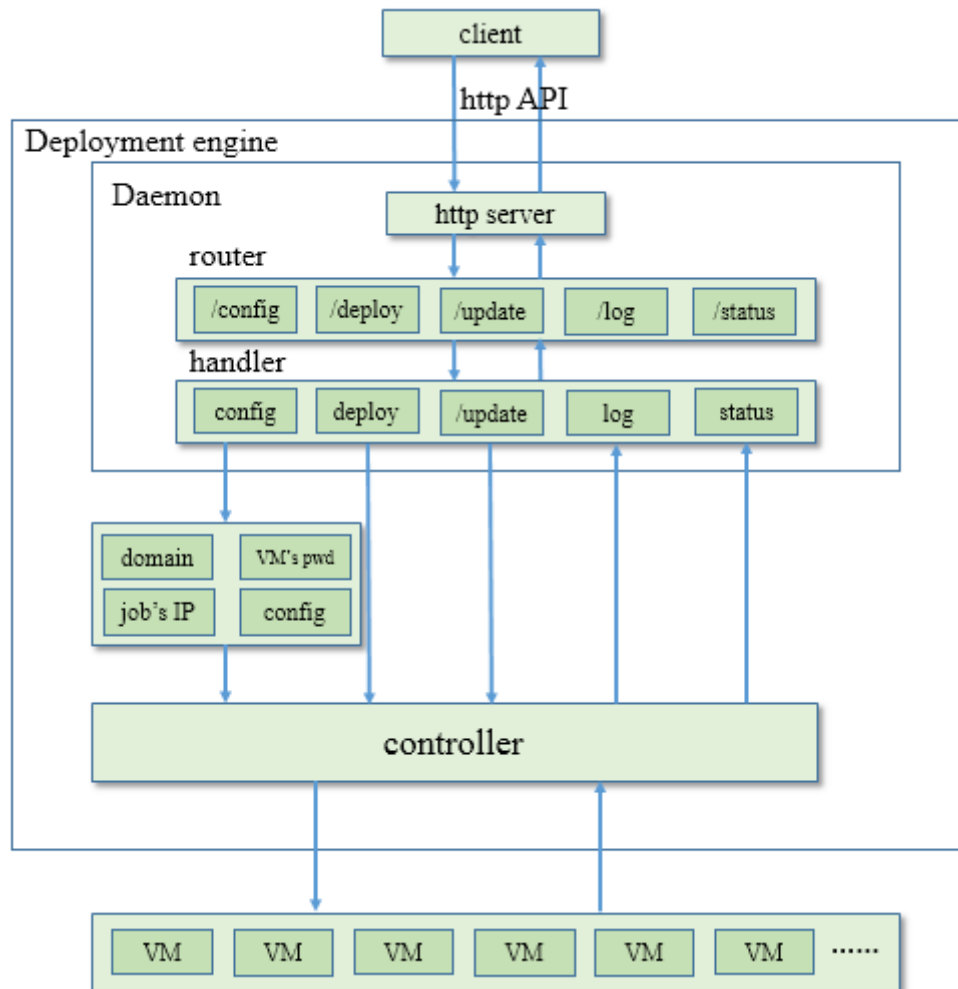
5、停止组件



- 1) 用户在客户端针对某个组件点击停止组件。
- 2) 客户端向 deployment engine 发送停止组件请求，包含组件的名称及 IP 等信息。
- 3) 服务端向 controller 发送 monit stop 命令。
- 4) Controller 根据组件名称及 IP 停止组件。
- 5) 返回 OK 状态。

Deployment Engine API

Deployment Engine 整体架构图如下所示：



Deployment engine 架构图

（注：API 较多，未一一列出）。

由上图所示，Deployment Engine 主要提供以下 API：

获取配置文件

GET /deployment-engine/config

Request

Route

```
GET /deployment-engine/config
```

cURL

```
curl http://IP:50000/deployment-engine/config -X GET
```

(注：IP 为 Deployment Engine 所在控制机 IP)

Response

Status

200 OK

Body

部署方案的配置，数据项有：

	Name	Descriptions	Valid values
Properties:	domain	集群的域名	字符串
	home	控制机 ip	字符串
	disksize	节点最低配置（GB）	字符串
	checkdisk	是否检测磁盘	字符串，1 是；0 否
	adduser	是否新建用户	字符串，1 是；0 否
	user	新建用户名称	字符串
	password	新建用户密码	字符串
	root	虚拟机的 root 密码	字符串，集群所有虚拟机 root 密码要求一致
Components:	haproxy	负载均衡器	Ip 地址，一个
	nats	消息通信总线	Ip 地址，可以多个
	nfs_server	共享存储	Ip 地址，一个，CPU>=1 RAM>=1024 DISK>=20480
	database	数据库	Ip 地址，一个，CPU>=1 RAM>=1024 DISK>=13312
	cloud_controller_ng	云控制器	Ip 地址，可以多个，CPU>=1 RAM>=4096 DISK>=13312
	etcd	键值存储	Ip 地址，一个
	uua	用户信息认证	Ip 地址，可以多个
	login	Web 登录	Ip 地址，可以多个
	hm9000	组件健康状态监控	Ip 地址，可以多个
	loggregator	应用日志收集	Ip 地址，一个，CPU>=1 RAM>=2048 DISK>=10240

	collector	组件日志收集	Ip 地址，可以多个
	syslog_aggregator	系统日志收集	Ip 地址，一个
	gorouter	路由器	Ip 地址，可以多个，CPU>=1 RAM>=2048 DISK>=10240
	dea	应用服务器	Ip 地址，可以多个，CPU>=2 RAM>=4096 DISK>=35840

（注：内存、硬盘单位均以 MB 表示，默认有效 IP 所在的虚拟机配置为 vCPU>=1 RAM>=1024 DISK>=5120）

```
{
  "components":{
    "haproxy":["10.10.101.181"],
    "nats":["10.10.101.182", "10.10.101.183"],
    "nfs_server":["10.10.101.184"],
    "database":["10.10.101.185"],
    "cloud_controller_ng":["10.10.101.186", "10.10.101.187"],
    "etcd":["10.10.101.188"],
    "uaa":["10.10.101.189", "10.10.101.190", "10.10.101.191"],
    "login":["10.10.101.192"],
    "hm9000":["10.10.101.193", "10.10.101.194"],
    "loggregator":["10.10.101.195"],
    "collector":["10.10.101.196"],
    "syslog_aggregator":["10.10.101.197"],
    "gorouter":["10.10.101.198"],
    "dea":["10.10.101.199"]
  },
  "properties":{
    "domain":"cf.local",
    "disksize":15,
    "checkdisk":1,
    "adduser":1,
    "user":"vcap",
    "password":"password",
    "root":"123456",
    "home":"10.10.101.180"
  }
}
```

Headers

Content-Type: application/json;charset=utf-8

提交配置文件

POST /deployment-engine/config

Request

Route

```
POST /deployment-engine/config
```

Body

同获取配置文件 Body 相同

Headers

```
Content-Type: application/json;charset=utf-8
```

cURL

```
curl http://IP:50000/deployment-engine/config -X POST -d
'{
  "components":{
    "haproxy":["10.10.101.181"],
    "nats":["10.10.101.182", "10.10.101.183"],
    "nfs_server":["10.10.101.184"],
    "database":["10.10.101.185"],
    "cloud_controller_ng":["10.10.101.186","10.10.101.187"],
    "etcd":["10.10.101.188"],
    "uaa":["10.10.101.189","10.10.101.190","10.10.101.191"],
    "login":["10.10.101.192"],
    "hm9000":["10.10.101.193","10.10.101.194"],
    "loggregator":["10.10.101.195"],
    "collector":["10.10.101.196"],
    "syslog_aggregator":["10.10.101.197"],
    "gorouter":["10.10.101.198"],
    "dea":["10.10.101.199"]
  },
  "properties":{
    "domain":"cf.local",
    "disksize":15,
    "checkdisk":1,
```

```
        "adduser":1,  
        "user":"vcap",  
        "password":"password",  
        "root":"123456",  
        "home":"10.10.101.180"  
    }  
}'
```

Response

Status

201 Created

400 Bad Request 发送的 json 数据格式不对

执行部署方案

GET /deployment-engine/deploy

Request

Route

GET /deployment-engine/deploy

cURL

```
curl http://IP:50000/deployment-engine/deploy -X GET
```

Response

Status

202 Accepted

409 Conflict 网络中有两台相同的控制机
500 Internal Server Error 控制机文件受损
502 Bad Gateway 网络异常

组件状态监控

GET /deployment-engine/status

Request

Route

GET /deployment-engine/status

cURL

curl http://IP:50000/deployment-engine/status -X GET

Response

Status

200 OK
403 Forbidden 没有提交配置方案

Headers

Content-Type: application/json;charset=utf-8

Body

Key	Value
"Comp"	组件名称
"Ip"	对应 IP

"Cpu"	CPU 使用率
"Mem"	内存占用率
"Disk"	硬盘使用率
"Status"	节点运行状态

```
[
{"Comp": "login", "Ip": "10.10.101.185", "Cpu": "1.18844%", "Mem": "81.74%", "Disk": "8%", "Status": "DOWN"},
{"Comp": "nfs_server", "Ip": "10.10.101.181", "Cpu": "1.16847%", "Mem": "82.78%", "Disk": "10%", "Status": "DOWN"},
{"Comp": "syslog_aggregator", "Ip": "10.10.101.181", "Cpu": "1.16972%", "Mem": "82.78%", "Disk": "10%", "Status": "DOWN"},
{"Comp": "uaa", "Ip": "10.10.101.185", "Cpu": "1.18958%", "Mem": "81.74%", "Disk": "8%", "Status": "RUNNING"},
{"Comp": "gorouter", "Ip": "10.10.101.185", "Cpu": "1.19083%", "Mem": "81.74%", "Disk": "8%", "Status": "RUNNING"},
{"Comp": "loggregator", "Ip": "10.10.101.181", "Cpu": "1.17088%", "Mem": "82.78%", "Disk": "10%", "Status": "DOWN"},
{"Comp": "dea", "Ip": "10.10.101.181", "Cpu": "1.1721%", "Mem": "82.78%", "Disk": "10%", "Status": "DOWN"},
{"Comp": "cloud_controller_ng", "Ip": "10.10.101.185", "Cpu": "1.19196%", "Mem": "81.74%", "Disk": "8%", "Status": "DOWN"},
{"Comp": "database", "Ip": "10.10.101.181", "Cpu": "1.17334%", "Mem": "82.78%", "Disk": "10%", "Status": "DOWN"},
{"Comp": "collector", "Ip": "10.10.101.181", "Cpu": "1.17454%", "Mem": "82.78%", "Disk": "10%", "Status": "DOWN"},
{"Comp": "hm9000", "Ip": "10.10.101.185", "Cpu": "1.19313%", "Mem": "81.74%", "Disk": "8%", "Status": "RUNNING"},
{"Comp": "nats", "Ip": "10.10.101.181", "Cpu": "1.17558%", "Mem": "82.78%", "Disk": "10%", "Status": "DOWN"},
{"Comp": "etcd", "Ip": "10.10.101.185", "Cpu": "1.19434%", "Mem": "81.74%", "Disk": "8%", "Status": "RUNNING"},
{"Comp": "haproxy", "Ip": "10.10.101.181", "Cpu": "1.1768%", "Mem": "82.78%", "Disk": "10%", "Status": "DOWN"}
]
```

请求节点硬件配置

GET /deployment-engine/hardware

Request

Route

```
GET /deployment-engine/hardware
```

Headers

```
Content-Type: application/json;charset=utf-8
```

Body

Key	Value
"Comp"	组件名称
"Ip"	对应 IP

(注：当 Comp=="all" && Ip=="0.0.0.0"时表示获取所有节点的硬件配置)

```
{ "Comp": "nats", "Ip": "10.10.101.183" }
```

cURL

```
curl http://IP:50000/deployment-engine/hardware -X GET -d '{ "Comp": "nats", "Ip": "10.10.101.183" }'
```

Response

Status

200 OK

400 Bad Request 组件和 IP 不匹配

403 Forbidden 没有提交配置方案

Headers

```
Content-Type: application/json;charset=utf-8
```

Body

Key	Value
"Comp"	组件名称
"Ip"	对应 IP
"CPU"	vCPU 核数
"RAM"	内存大小
"DISK"	硬盘大小

(注：当 Comp=="all" && Ip=="0.0.0.0"时表示获取所有节点的硬件配置，此时 json 文件中会以数组形式传输数据)

```
{"Comp": "haproxy", "IP": "10.10.101.181", "CPU": 2, "RAM": 4049952, "DISK": 54988.8}
```

获取日志文件列表

GET /deployment-engine/download

Request

Route

```
GET /deployment-engine/download
```

cURL

```
curl http://IP:50000/deployment-engine/download -X GET
```

Response

Status

200 OK

Headers

Content-Type: html/text; charset=utf-8

Body

字符串数组

```
["local.log", "nats_0.log", "nats_1.log"]
```

下载日志文件

PUT /deployment-engine/download

Request

Route

PUT /deployment-engine/download

Headers

Content-Type: html/text; charset=utf-8

Body

字符串，日志文件名称

```
{ "local.log" }
```

cURL

```
curl http://IP:50000/deployment-engine/download -X PUT -d 'local.log'
```

Response

Status

200 OK
404 Not Found 请求的日志文件不存在

Headers

Content-Type: html/text;charset=utf-8

Body

“start es.....\nstart success.”

重启节点

PUT /deployment-engine/restart

Request

Route

PUT /deployment-engine/restart

Headers

Content-Type: application/json;charset=utf-8

Body

Key	Value
"name"	组件名称
"ip"	对应 IP

(注：当 Comp=="all" && Ip=="0.0.0.0"时表示重启整个集群)

```
{ "name": "nats", "Ip": "10.10.101.183" }
```

cURL

```
curl http://IP:50000/deployment-engine/restart -X PUT -d '{ "Comp":"nats", "Ip":"10.10.101.183" }'
```

Response

Status

200 OK
400 Bad Request 请求的 json 数据格式不对
403 Forbidden 没有提交配置方案

停止节点

PUT /deployment-engine/stop

Request

Route

PUT /deployment-engine/stop

Headers

Content-Type: application/json;charset=utf-8

Body

Key	Value
"name"	组件名称
"ip"	对应 IP

(注：当 Comp=="all" && Ip=="0.0.0.0"时表示停止整个集群)

```
{ "name":"nats", "Ip":"10.10.101.183" }
```

cURL

```
curl http://IP:50000/deployment-engine/stop -X PUT -d '{ "Comp": "nats", "Ip": "10.10.101.183" }'
```

Response

Status

200 OK

400 Bad Request 请求的 json 数据格式不对

403 Forbidden 没有提交配置方案

Deployment engine 部署 cloud foundry

1、启动 deployment engine 只需要将 deployment-engine.ova 虚拟机

模板导入到 vcenter 上启动即可。记录虚拟机 IP，则客户端可通过接口 IP:50000/deployment-engine/config 向 deployment engine 发送创建配置文件请求。

虚拟机启动以后，deployment engine 会进行初始化，首先启动 deployment engine server 监听客户端请求，监听端口为 50000；

架设 linux 本地源：

1.1 部署 Apache 服务器

1.2 使缓存安装包可访问

1.3 修改本地源

构建本地运行时环境：

1.4 从 linux 本地源安装 linux 依赖

1.5 安装 ruby、rubygems 等文件

1.6 安装缓存的 ruby gems 包

2、用户在客户端填写配置文件，包括集群域名、虚拟机 root 密码、各组件对应的虚拟机节点 IP 组等。填写完成后，保存，客户端会将配置数据发送到 deployment engine 服务端，server 端会检测各组件下的节点是否满足最低配置要求，不满足会返回错误原因，否则返回保存成功，同时会将各节点资源数据（CPU、RAM、DISK）返回，具体流程如下：

部署前安全性检查：

2.1 检查组件网络是否通畅

2.2 检查系统内存、硬盘、CPU 是否满足基本要求

2.3 检查用户是否存在，否则添加用户

配置文件解析

2.4 解析运行参数

2.5 解析配置文件（组件、IP、域名等）

2.6 根据读入的组件信息，修改 cf 标准配置文件

2.7 根据部署的组件生成相应的自动化部署脚本

3、完成配置文件的创建后，用户需要在 DNS 服务器上添加域名解析，比如配置集群域名为 cf.local，则需在 DNS 添加对 *.cf.local ⇔ haproxy 组件 IP 的解析。

4、用户点击按钮执行部署方案，deployment engine 接收到命令后，根据配置文件启动部署，具体流程如下：

文件传输：

4.1 根据部署的文件进行拓扑排序，不同时对同一个安装介质（虚拟机或容器）安装，不传输冗余文件

4.2 传输生成的部署脚本及配置文件

4.3 解析组件所需要的 release 文件中必要的内容传输

远程自动化部署：

4.4 替换 linux/apt 源，换成本地源并更新

4.5 安装运行时依赖包

4.6 源码安装 ruby、gemsd 等运行必须文件

4.7 使用 nisebosh 部署组件

4.8 特殊组件安装

4.9 启动组件，并加入 linux 启动项

错误处理：

4.10 收集部署过程中的日志

4.11 对已经检测出的配置错误等反馈给用户进行修改

4.12 对出错的组件安装，文件传输遗漏等错误进行重新安装

5、部署完成以后，点击 status 按钮，可以查看所有节点的组件运行状态，选中某个组件可以进行重启或者停止操作。

6、如果需要扩大集群规模，用户在客户端中点击 update，选择添加的组件名称，再输入对应节点 IP，最后点击 apply，即可完成集群扩展。

Trouble Shoot

- scp 传输不完整

Exception:

```
scp not finish completely
```

-

重新运行部署脚本。

- 组件经过改动后需要重新登录

```
"FAILED Server error, status code: 404, error code: 40004, message: The app space could not be found"
```

```
"FAILED Server error, status code: 404, error code: 40004, message: The Authentication failed"
```

-

重新登陆集群: `cf logout`。

- monit 资源不够不能使用

```
monit: cannot read status from the monit daemon
```

-

解决方案:这个是由于 CPU 资源不够了, 需要重启虚拟机。

```
reboot
```

- 域名服务器没有添加导致的各类错误。

解决方案：

DNS 服务器添加对域名到 haproxy IP 的解析