Flynn 文档中文版 VPreview

说明

简介

Flynn(http://www.oschina.net/p/flynn) 是一个开源的 PaaS 系统，由 Docker 开发。采用 Go 语言编写。支持数据库包括 Postgres、Redis 和 MongoDB. Flynn 使用完全组件化模块化的设计，任何一个组件和模块都可以独立的进行替换。

《Flynn 文档中文版》原文出自《 Flynn Docs(https://flynn.io/docs)》，由多位网友在开源中国众包平台协作翻译完成，其中：

• Using Flynn + Installation + Language 由 @Andy(http://my.oschina.net/hand) 翻译，@国栋(http://my.oschina.net/goldenshaw) 校对；

• Databases + Stability + Security + Contributing + Development 由 @shellmode(http://my.oschina.net/shellmode) 翻译，@静怡芸香(http://my.oschina.net/u/2485518) 校对；

• FAQ + API + Trademark Guidelines 由 @shellmode(http://my.oschina.net/shellmode) 翻译。

反馈

对《Flynn 文档中文版》有任何反馈，欢迎在以下网址评论：

http://www.oschina.net/news/69536/flynn-doc-cn(http://www.oschina.net/news/69536/flynn-doc-cn)

版权

《Flynn 文档中文版》由开源中国组织翻译，转载请注明出处，未经许可不得为其它商业目的使用。

开始使用

开始使用

开始使用

阅读本文档需要先安装好 Flynn 集群环境，并配置好命令行工具。如果 Flynn 还没安装好，请参考安装指南(https://flynn.io/docs/installation)，先进行环境配置。

本文档假定你的 Flynn 集群使用demo.localflynn.com这个默认域名（如果你安装的是演示环境）。如果你使用自定义的域名，请在 Flynn 启动时，设置CLUSTER_DOMAIN参数的值，替换掉默认的`demo.localflynn.com`。

部署应用

下面我们将部署一个使用 Node.js 编写的演示程序，它自带了一个最简单功能的 HTTP 服务器。

首先使用 Git 将代码 clone 到本地：

```
$ git clone https://github.com/flynn/nodejs-flynn-example.git
```

进入本地的代码目录，创建一个 Flynn 应用：

```
$ cd nodejs-flynn-example
$ flynn create example
Created example
```

上面的命令会增加一个 Flynn 的 Git 远程分支。

```
$ git remote -v
flynn   https://git.demo.localflynn.com/example.git (push)
flynn   https://git.demo.localflynn.com/example.git (fetch)
origin  https://github.com/flynn/nodejs-flynn-example.git (fetch)
origin  https://github.com/flynn/nodejs-flynn-example.git (push)
```

同时还会增加一个默认路由，将example.demo.localflynn.com指向example-web服务。

```
$ flynn route
ROUTE                       SERVICE     ID
http:example.demo.localflynn.com  example-web
http/1ba949d1654e711d03b5f1e471426512
```

推送 Flynn 远程分支，来部署应用：

```
$ git push flynn master
...
-----> Building example...
-----> Node.js app detected
...
-----> Creating release...
=====> Application deployed
=====> Added default web=1 formation
```

```
To https://git.demo.localflynn.com/example.git
* [new branch]      master -> master
```

现在应用已经部署成功，可以使用上面创建的域名来访问：

```
$ curl http://example.demo.localflynn.com
Hello from Flynn on port 55006 from container d55c7a2d5ef542c186e0feac5b94a0b0
```

应用扩展

Flynn 里部署的应用会在根目录下生成一个Procfile配置文件，在此文件中定义该应用的类型。比如上文中部署的演示程序，它的配置中声明这是一个web应用，入口可执行程序是web.js。

```
$ cat Procfile
web: node web.js
```

web类型的应用默认的配置是启动一个web进程，这可以在 Flynn 里用ps命令查看：

```
$ flynn ps
ID                                TYPE
flynn-d55c7a2d5ef542c186e0feac5b94a0b0  web
```

可以在 Flynn 里使用scale命令来启动更多的web进程：

```
$ flynn scale web=3
```

使用ps命令可以看到现在启动了三个进程：

```
$ flynn ps
ID                                TYPE
flynn-7c540dffaa7e434db3849280ed5ba020  web
flynn-3e8572dd4e5f4136a6a2243eadca5e02  web
flynn-d55c7a2d5ef542c186e0feac5b94a0b0  web
```

重复访问部署的 web 服务，可以看到启动的三个进程会自动进行负载均衡：

```
$ curl http://example.demo.localflynn.com
Hello from Flynn on port 55006 from container d55c7a2d5ef542c186e0feac5b94a0b0
```

```
$ curl http://example.demo.localflynn.com
Hello from Flynn on port 55007 from container 3e8572dd4e5f4136a6a2243eadca5e02

$ curl http://example.demo.localflynn.com
Hello from Flynn on port 55008 from container 7c540dffaa7e434db3849280ed5ba020

$ curl http://example.demo.localflynn.com
Hello from Flynn on port 55007 from container 3e8572dd4e5f4136a6a2243eadca5e02
```

查看应用日志

可以使用log命令来查看任务日志（例如：stdout/stderr）：

```
$ flynn log flynn-d55c7a2d5ef542c186e0feac5b94a0b0
Listening on 55006
```

关于flynn log命令的详细介绍可以访问这里(https://flynn.io/docs/cli#log)。

应用发布

在 git 里提交更新，同时推送到 Flynn 里就可以创建应用的新发布。

在web.js里最上面增加下面一行：

```
console.log("I've made a change!")
```

在 git 里提交更新到 Flynn：

```
$ git add web.js
$ git commit -m "Add log message"
$ git push flynn master
```

一旦更新成功，你会看到三个新的进程：

```
$ flynn ps
ID                                TYPE
flynn-cf834b6db8bb4514a34372c8b0020b1e  web
flynn-16f2725f165343fca22a65eebab4e230  web
flynn-d7893da39a8847f395ce47f024479145  web
```

这些进程的日志里能看到上面增加的信息：

```
$ flynn log flynn-cf834b6db8bb4514a34372c8b0020b1e
I've made a change!
Listening on 55007
```

应用路由

创建应用时，应用的web进程会生成一个默认的HTTP路由，是默认域名的子域名（例如：example.demo.localflynn.com）。如果你想使用不同的域名，则需要增加新的路由配置。

假如你有一个example.com域名，它指向你的 Flynn 集群（例如：它是example.demo.localflynn.com的CNAME别名）。

为这个域名增加一个路由：

```
$ flynn route add http example.com
http/5ababd603b22780302dd8d83498e5172
```

现在你的web应用有了两个路由：

```
$ flynn route
ROUTE                          SERVICE      ID
http:example.com                    example-web
http/5ababd603b22780302dd8d83498e5172
http:example.demo.localflynn.com  example-web
http/1ba949d1654e711d03b5f1e471426512
```

现在访问example.com的请求会被路由到对应的web进程：

```
$ curl http://example.com
Hello from Flynn on port 55007 from container cf834b6db8bb4514a34372c8b0020b1e
```

现在你可以修改应用通过 HTTP 头信息（这里是example.demo.localflynn.com 或 example.com）来识别不同的访问并且返回不同的内容。

注意：HTTP 路由只能增加到默认的web应用和以-web结尾的应用类型里。例如：对于api-web类型的应用，你可以增加如下路由：

```
$ flynn route add http -s example-api-web api.example.com
http/9cfb5f1b-b174-476c-b869-71f1e03ef4b
```

多进程

到目前为止，我们的应用只有一种进程类型（例如：web进程），但在 Flynn 里部署的应用可以同时拥有多种进程类型，并且可以独立进行配置扩展。

让我们增加一个简单的clock服务，每秒钟输出当前的时间。

在clock.js里增加下面的代码：

```
setInterval(function() {
  console.log(new Date().toTimeString());
}, 1000);
```

在应用的Procfile里指定clock的进程类型：

```
clock: node clock.js
```

发布更新：

```
$ git add clock.js Procfile
$ git commit -m "Add clock service"
$ git push flynn master
```

创建一个clock进程，并且查看输出：

```
$ flynn scale clock=1

$ flynn ps
ID                                TYPE
flynn-aff3ae71d0c149b185ec64ea2885075f  clock
flynn-cf834b6db8bb4514a34372c8b0020b1e  web
flynn-16f2725f165343fca22a65eebab4e230  web
flynn-d7893da39a8847f395ce47f024479145  web

$ flynn log flynn-aff3ae71d0c149b185ec64ea2885075f
18:40:42 GMT+0000 (UTC)
18:40:43 GMT+0000 (UTC)
18:40:44 GMT+0000 (UTC)
18:40:45 GMT+0000 (UTC)
18:40:46 GMT+0000 (UTC)
...
```

运行进程

单次运行的交互式进程可以在容器里用命令创建：

```
$ flynn run bash
```

关于flynn run命令的详细介绍可以访问这里(https://flynn.io/docs/cli#run)。

安装

云安装

系统安装

正式开始安装之前，需要先配置Flynn的命令行环境。

在 OS X 和 Linux 上，运行如下命令进行安装：

```
L=/usr/local/bin/flynn && curl -sSL -A "`uname -sp`" https://dl.flynn.io/cli | zcat >$L && chmod +x $L
```

在 Windows 上，启动 PowerShell，运行：

```
(New-Object Net.WebClient).DownloadString('https://dl.flynn.io/cli.ps1') | iex
```

CLI 命令行包含一个基于浏览器的本地安装程序，它可通过 SSH 启动和配置一个在Amazon Web Services，DigitalOcean，Azure 或是你自己服务器上的 Flynn 集群。

运行flynn install命令，来启动安装程序。

如果想在你本地的机器上运行 Flynn，可以安装基于Vagrant的测试程序包(https://flynn.io/docs/installation/vagrant)，这是最简单的办法。

如果你想手动安装 Flynn，请参见手动安装文档(https://flynn.io/docs/installation/manual)。

Vagrant 演示环境下安装

Vagrant 演示环境下安装

Vagrant 是一个虚拟机管理程序，可以运行在本地机器上。我们提供了一个配置好的 Vggrant/VirtualBox 虚拟机镜像文件。你不用进行安装，可以直接用来做本地测试。

系统要求

电脑里需要先配置好 Vagrant 和 VirtualBox，如果没有安装，请参见下面的网址进行安装：

- VirtualBox(https://www.virtualbox.org/wiki/Downloads)
- Vagrant 1.6 或以上(https://www.vagrantup.com/downloads.html)

安装

使用 git 将 Flynn 代码 clone 到本地：

```
$ git clone https://github.com/flynn/flynn
```

如果你系统里安装了make，我们提供了一个Makefile文件，对常用的 Flynn 命令进行了封装，可以方便使用。当然，如果没有make，你也可以手动运行这些命令。

进入代码树里的demo目录，启动 Vagrant：

```
$ cd flynn/demo

# Provision the VM and bootstrap a flynn cluster
# Init should only be called once
$ make init
# Print login token and open dashboard in browser
$ make dashboard

# OR

$ vagrant up
# Follow the instructions output by vagrant up, then...
$ flynn -a dashboard env get LOGIN_TOKEN
# Copy the login token
# Open the dashboard in a browser
$ open http://dashboard.demo.localflynn.com
```

其他有用的make命令：

```
# Halt the VM
$ make down

# Bring up the VM and flynn
$ make up

# SSH into the VM
$ make ssh

# Destroy the VM
$ make destroy

# See Makefile for other useful commands
```

```

```

如果虚拟机没有成功启动，可以使用下面的命令重启：

```
# Stop and restart the VM
$ make down
$ make up

# OR, destroy and rebuild the VM
$ make reset
```

安装成功之后，虚拟机里运行一个单节点的 Flynn 集群。

现在，你已经成功的安装并运行 Flynn。可以参见上面的使用文档(https://flynn.io/docs)来部署你的应用。

手动安装

手动安装

在 Ubuntu 14.04 amd64 上可以使用我们提供的安装脚本进行安装。

我们推荐在一个新安装的 Ubuntu 电脑上进行安装，需要的最小硬件配置为：1GB 内存，20GB 硬盘剩余空间，双核心 CPU。

还需要注意的是，如果你计划安装多节点的 Flynn 集群，为了保证集群的健壮性，至少需要三个 Flynn 节点。

*注意：

如果你使用Linode的 VPS 进行安装，必须使用原生的 Linux 内核（不能使用 Linode 的内核）才能支持AUFS文件系统。参见这里(https://www.linode.com/docs/tools-reference/custom-kernels-distros/run-a-distributionsupplied-kernel-with-pvgrub)，了解如何切换内核。*

安装

下载并运行安装脚本：

```
$ sudo bash < <(curl -fsSL https://dl.flynn.io/install-flynn)
```

此脚本需要以 root 权限运行，如果在运行之前，你想先看看脚本的内容，可以将上述命令分成两步：

```
$ curl -fsSL -o /tmp/install-flynn https://dl.flynn.io/install-flynn
... take a look at the contents of /tmp/install-flynn ...
$ sudo bash /tmp/install-flynn
```

```

```

此安装脚本将会执行如下任务：

1.安装 Flynn 运行时依赖的环境。

2.下载，校验并且安装flynn-host程序。

3.下载，校验每个Flynn组件的文件系统映像。

4.在系统里安装Upstart作业，来控制flynn-host服务。

有些文件系统映像非常大（几百MB的数量级），因此上面第三步的运行会需要一定的时间。

多节点安装

在集群环境下，需要在每个节点上重复上述的安装步骤。

启动

启动之前先确认 Flynn 集群各节点之间的网络通信是否正常（所有的 TCP 和 UDP通信），还需要在集群的防火墙上放行如下端口：

- 80 (HTTP)
- 443 (HTTPS)
- 3000 到 3500（用户自定义的 TCP 服务，可选）

注意：在 Flynn 系统中，基于安全考虑，需要配置防火墙来防止外界访问内部的管理 API。

接下来需要在所有的节点上启动 flynn-host 服务来配置集群 Layer 0。此进程在各节点间使用Raft 协议来进行 Leader 的选举，需要各节点的服务正确运行。

如果启动了多个节点，集群会使用一个token来同步s配置各节点。使用flynn-host init 命令生成和使用token。

在第一个节点上使用 --init-discovery参数生成一个新的token，集群环境最少需要三个节点，在单节点环境下，不需要运行这个命令。

```
$ sudo flynn-host init --init-discovery
https://discovery.flynn.io/clusters/53e8402e-030f-4861-95ba-d5b5a91b5902
```

在其余节点上使用上面生成的token 进行配置：

```
$ sudo flynn-host init --discovery https://discovery.flynn.io/clusters/53e8402e-030f-
4861-95ba-d5b5a91b5902
```

接下来启动 flynn-host 服务：

```
$ sudo start flynn-host
```

可以使用下面的命令查看服务运行状态：

```
$ sudo status flynn-host
flynn-host start/running, process 4090
```

如果服务状态是 stop/waiting，表示服务的启动不成功。可以打开日志文件(/var/log/upstart/flynn-host.log)查找具体原因，并尝试重新启动。

Bootstrap Flynn

flynn-host成功运行后，就可以使用flynn-host bootstrap启动 Flynn。现在你需要在 DNS 上为每个节点的 IP 配置对应的 A 记录，同时使用通配符配置集群域 CNAME 别名的泛解析。

例如：

```
demo.localflynn.com.    A      192.168.84.42
demo.localflynn.com.    A      192.168.84.43
demo.localflynn.com.    A      192.168.84.44
*.demo.localflynn.com.  CNAME  demo.localflynn.com.
```

如果你使用单节点 Flynn，不想进行 DNS 的配置，可以使用xip.io(http://xip.io/)提供的域名泛解析服务。

将集群域名设置为CLUSTER_DOMAIN，作为参数启动 Flynn，同时在参数里还要提供节点数量和上面创建的 Token。

```
$ sudo \
    CLUSTER_DOMAIN=demo.localflynn.com \
    flynn-host bootstrap \
    --min-hosts 3 \
    --discovery https://discovery.flynn.io/clusters/53e8402e-030f-4861-95ba-
d5b5a91b5902
```

注意：在集群环境下，只需要在一个节点上执行上面的命令即可。在需要的情况下会自动同步到各节点。

Flynn Layer 1 的启动过程中会打开所有依赖 Layer 0 API 的服务。日志的最后包含使用命令行接口(https://flynn.io/docs/cli)的配置。

如果在上面的步骤中出现了问题，可以在网上提交或者寻找社区帮助。

现在，你已经成功的安装并运行 Flynn。可以参见上面的使用文档(https://flynn.io/docs)来部署你的应用。

编程语言支持

如何部署 Go 应用

编程语言支持

如何部署 Go 应用

Flynn 使用 Go buildpack(https://github.com/kr/heroku-buildpack-go) 来支持 Go 语言。

检测

当检测到部署的应用里包含以.go 为后缀的文件时，Flynn 会使用 Go buildpack。

环境依赖

Go buildpack 提供两个方法安装依赖的软件包，godep 和 go get，推荐使用 godep(https://github.com/tools/godep)，这个命令可以将依赖的软件包存储在 git 仓库里，可以在重复性部署的时候自动解决依赖关系。

godep 命令

使用 godep(https://github.com/tools/godep) 命令保存依赖的软件包，进入应用的目录，运行 godep save命令，然后提交 Godeps 目录。当在 Flynn 里部署应用的时候，Godeps目录里的软件包会被自动安装。

go get 命令

如果应用的代码仓库里没有Godeps目录，buildpack 会自动下载 Mercurial(http://mercurial.selenic.com/) 和 Bazaar(http://bazaar.canonical.com/en/)，并运行 go get 命令安装所导入软件包的最新版本。这种方式比较慢，并且不能重复构建的过程，所以不可靠。

如果不使用godep，应用的根目录里需要有一个.godir文件。这个文件里包含应用软件包的完整路径，Flynn 读取这个参数来定位软件包并且给应用命名。例如：一个应用根目录下的.godir文件包含github.com/flynn/flynn，表示这个应用的名称为：flynn。

Go 版本

使用godep命令时，可以通过Godeps/Godeps.json里的GoVersion来指定 Go 的版本。

如果不使用 godep ，系统默认使用构建包里的最新版本。

二进制程序

应用目录下的所有主软件包都要被编译成二进制格式，并存放在/app/bin目录里，该路径包含在系统的PATH环境变量里。程序以包含它们的目录来命名。

如果应用的根目录下包含一个主软件包（ main package ），最终应用的名称会根据主软件包的路

径得出。如果你使用godep，系统从Godeps/Godeps.json里读取ImportPath参数得到这个路径，如果你不使用godep，系统会读取.godir文件。

应用类型

在应用根目录下的 Procfile里声明应用支持的类型，一种应用类型占一行，格式：TYPE: COMMAND。

例如：应用的根目录下有一个主软件包，软件包路径为：github.com/flynn/myserver，那么，最终应用会被命名为：myserver，Procfile里的定义如下：

```
web: myserver
```

web类型的应用会有默认的 HTTP 路由，会在环境变量里定义服务器监听的通信端口（PORT）。

如何部署 Java 应用

如何部署 Java 应用

Flynn 需要使用 JAVA(https://github.com/heroku/heroku-buildpack-java) 或 Gradle(https://github.com/heroku/heroku-buildpack-gradle) 构建包， 利用 Maven(http://maven.apache.org/) 或者 Gradle(http://www.gradle.org/) 来完成 JAVA 应用的自动构建和部署。

Flynn 使用 OpenJDK(http://openjdk.java.net/) 来运行 JAVA 应用。

应用检测

Flynn 使用下面的规则检测 JAVA 应用：

• 在应用的根目录下的 pom.xml 文件，表明此 JAVA 应用使用 Maven 构建工具。

• 在应用的根目录下的gradlew，build.gradle，或settings.gradle文件，表明此 JAVA 应用使用 Gradle 构建工具。

环境依赖

Maven

当使用 Maven 的时候，将所依赖的软件包加入pom.xml文件的<dependencies>配置项里。

例如，下面的pom.xml配置文件说明编译环境需要log4j的支持：

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
  <modelVersion>4.0.0</modelVersion>

  <groupId>io.flynn.example</groupId>
  <artifactId>flynn-example</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>flynn-example</name>

  <dependencies>
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
      <scope>compile</scope>
    </dependency>
  </dependencies>
</project>
```

当包含此配置文件的应用部署到 Flynn 时，系统会自动执行如下的 Maven 命令：

```
mvn -B -DskipTests=true clean install
```

上述命令会下载编译依赖的软件包，并将编译结果输出到target目录。

Flynn 默认使用最新版本的 Maven。如果想使用特定版本的 Maven，需要在根目录下创建system.properties文件，在此文件指定 Maven 的版本：

```
maven.version=3.2.3
```

使用 Maven 时，关于环境依赖的更多信息，可以参见 Maven 环境依赖 (http://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html)文档。

Gradle

使用 Gradle 时，需要使用它的 Java 插件 (http://www.gradle.org/docs/current/userguide/java_plugin.html) ，按插件要求的格式在根目录下的build.gradle文件里配置所依赖的软件包。

Flynn 在编译和部署应用的过程中，会运行一系列预定义的stage任务。

例如，下面是一个build.gradle文件，里面声明了编译期间依赖log4j，并且定义了一个stage任务，完成编译清理工作：

```
apply plugin: "java"

repositories {
  mavenCentral()
}

dependencies {
  compile group: "log4j", name: "log4j", version: "1.2.17"
}

task stage (dependsOn: ["clean", "jar"])
```

当包含上述配置文件的应用部署到 Flynn 时，系统自动执行下面的命令：

```
./gradlew stage
```

会下载所依赖的软件，执行编译，并将编译结果放在build目录中。

注意：建议在应用里包含gradlew脚本，这个脚本能检测使用的 Gradle 版本。如果系统中没有，Flynn 会自动安装。但此版本是不确定的。参见 Gradle Wrapper 页面 (https://docs.gradle.org/current/userguide/gradle_wrapper.html)获取更多信息

使用 Gradle 时，关于环境依赖的更多信息，可以参见 Gradle 环境依赖 (http://www.gradle.org/docs/current/userguide/artifact_dependencies_tutorial.html)文档。

JAVA 运行环境

运行 JAVA 应用时，Flynn 默认使用 OpenJDK 8。OpenJDK 6 和 7 也是可用的。可以修改根目录下的system.properties文件，修改其中的java.runtime.version来更换版本：

```
! OpenJDK 6
java.runtime.version=1.6

! OpenJDK 7
java.runtime.version=1.7
```

应用类型

可以在应用根目录下的 Procfile 里声明应用的类型，声明的格式是：TYPE: COMMAND。

web

一般情况下 WEB 类型的应用会包含 HTTP 路由，通信端口(port)等环境变量，会为应用启动一个 HTTP 服务器。

内嵌 Jetty

内嵌 Jetty 服务器的应用会使用环境变量中定义的端口(port)来启动应用，例如：

```
import javax.servlet.http.HttpServlet;
import org.eclipse.jetty.server.Server;

public class MyServlet extends HttpServlet
{
  // handler definitions

  public static void main(String[] args) throws Exception
  {
      Server server = new Server(Integer.valueOf(System.getenv("PORT")));
      // code to set handlers and start the server
  }
}
```

想了解更多关于 Jetty 嵌入的信息，请参见嵌入 Jetty 页面 (https://wiki.eclipse.org/Jetty/Tutorial/Embedding_Jetty)。

假如一个应用使用 Gradle 构建工具，通过 Gradle 应用插件创建了名为example的应用，应用的类型在 Procfile 里按如下格式定义：

```
web: build/install/example/bin/example
```

外部 Jetty + WAR 包

打包成 WAR 格式的 JAVA 应用需要使用外部的 Servlet 容器来运行。

也可以让 Marven Dependency 插件(http://maven.apache.org/plugins/maven-dependency-plugin/) 自动拷贝一个 jetty-runner JAR 文件的副本到target/dependency 目录，这需要在 pom.xml里增加如下配置：

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>2.3</version>
      <executions>
```

```
      <execution>
        <phase>package</phase>
        <goals><goal>copy</goal></goals>
        <configuration>
          <artifactItems>
            <artifactItem>
              <groupId>org.mortbay.jetty</groupId>
              <artifactId>jetty-runner</artifactId>
              <version>7.5.4.v20111024</version>
              <destFileName>jetty-runner.jar</destFileName>
            </artifactItem>
          </artifactItems>
        </configuration>
      </execution>
    </executions>
  </plugin>
 </plugins>
</build>
```

将应用打包成 WAR 格式，存储到 target 目录，在 Procfile 文件里定义应用类型：

```
web: java $JAVA_OPTS -jar target/dependency/jetty-runner.jar --port $PORT
target/*.war
```

如何部署 Node.js 应用

如何部署 Node.js 应用

Flynn 使用 Heroku Node.js 构建包(https://github.com/heroku/heroku-buildpack-nodejs)来支持 Node.js 应用。

应用检测

当应用的目录里包含package.json文件时，Flynn 就会使用 Node.js 构建包。

环境依赖

Flynn 使用npm管理依赖。npm读取package.json文件的dependencies属性

(https://www.npmjs.org/doc/files/package.json.html#dependencies)里的定义，它是一个简单的键值对象，软件包的名称是键名，对应可用版本的范围。

使用特定版本的 Node.js

可以在package.json文件的engines节（section(https://www.npmjs.org/doc/files/package.json.html#engines)）里定义 Node.js 版本。它使用 semver.io(http://semver.io/) 来解析 Node.js 的版本，支持诸如：0.8.x，>0.4，>=0.8.5 <-0.8.14之类格式的查询。Node.js 的构建包支持 0.8.5 之后的版本，包含开发版。

示例 package.json

```
{
  "name": "node-example",
  "version": "0.0.1",
  "dependencies": {
    "express": "4.10.0",
    "stylus": "0.49.2"
  },
  "devDependencies": {
    "grunt": "0.4.5"
  },
  "engines": {
    "node": "0.10.x",
    "npm": "1.2.x"
  }
}
```

自定义构建

有些应用在部署前需要额外的处理步骤，这时可以增加一个 npm postinstall脚本。它会在npm install --production后执行，生产环境下也是可用的。注意，构建工具不会自动安装devDependencies里的软件包，如果你需要安装其中的软件，就把它移到dependencies里。

默认应用类型

Node.js 应用在部署时可以没有Procfile文件。如果没有Procfile，构建工具会查找package.json里的scripts.start定义的脚本，然后以默认的web应用类型使用npm start启动此脚本。

运行任务

另外，npm，node等都在系统的path环境变量里，可以直接通过flynn run运行：

```
$ flynn run node -v
v0.10.32
```

如何部署 PHP 应用

如何部署 PHP 应用

Flynn 使用 PHP 或者 HHVM(http://hhvm.com/) 来支持 PHP 应用，使用的 web 服务器是 Apache2(http://httpd.apache.org/) 或 Nginx(http://wiki.nginx.org/Main)。

Flynn 使用 Heroku PHP 构建包(https://github.com/heroku/heroku-buildpack-php)来完成 PHP 应用的检测，编译和部署。

应用检测

Flynn 通过应用根目录下的composer.json来检测是否为 PHP 应用。使用 Composer(https://getcomposer.org/) 来管理依赖关系，可以自动下载安装依赖的软件包。

即使应用没有外部依赖的软件包需要用 Composer 管理，也需要在应用的根目录下创建一个composer.json空文件，Flynn 据此判断应用类型。

软件依赖

软件包

composer.json文件的主要用途是用来声明软件包的依赖关系。

下面composer.json的例子，声明了应用对 monolog(https://github.com/Seldaek/monolog) 的依赖：

```
{
  "require": {
    "monolog/monolog": "1.11.*"
  }
}
```

运行composer install命令会自动下载安装所依赖的软件，并创建一个composer.lock文件，其中包含所有已安装的软件版本的快照。当应用部署时，必须包含一个composer.lock文件，Flynn 据此检测软件包安装的版本。

Composer同时创建vendor/autoload.php文件，可以在 PHP 程序中包含进去，能在程序里自动加载所依赖的软件包。例如，刚才依赖的monolog，在程序里按如下方式使用：

```
require 'vendor/autoload.php';

// you can now reference Monolog
use Monolog\Logger;

$log = new Logger('my-application');
...
```

关于composer.json文件的详细信息，参见：Composer JSON 模式说明页面(https://getcomposer.org/doc/04-schema.md)。

环境配置

可能有些软件包只需要在本地使用，部署到生产系统时不需要安装，这可以通过运行composer install时，增加--no-dev参数实现。这样在部署时，系统会忽略composer.json文件里require-dev中定义的内容。

例如：本地环境可能会使用 phpunit，但在部署后很少会需要使用，你可以这样配置require-dev：

```
 {
  "require": {
    "monolog/monolog": "1.11.*"
  },
  "require-dev": {
    "phpunit/phpunit": "4.3.*"
  }
}
```

针对上面配置，完整运行的 Composer 命令如下：

```
composer install \
  --no-dev \
  --prefer-dist \
  --optimize-autoloader \
  --no-interaction
```

关于composer install的详细信息，请参加composer install 文档(https://getcomposer.org/doc/03-cli.md#install)。

PHP运行环境

Flynn 默认使用最新的稳定版 PHP，也可以在依赖关系里配置特定的 PHP 或 HHVM 版本。

例如，使用 PHP 5.6.x：

```
{
  "require": {
    "php": "~5.6.0"
  }
}
```

使用 HHVM 3.2.x：

```
{
  "require": {
    "hhvm": "~3.2.0"
  }
}
```

建议在版本号前增加~操作符号，这样应用会使用该大版本下最新的稳定小版本。关于~的详细介绍，参见 Composer 文档(https://getcomposer.org/doc/01-basic-usage.md#next-significant-release-tilde-operator-)。

应用类型

在应用根目录下的Procfile里声明应用的类型。格式：TYPE: COMMAND。

web

web类型的应用包含 HTTP 路由，配置了通信端口等环境变量，一般来说还会启动一个 HTTP 服务器。

Flynn 内置支持两种 web 服务器。

Apache2

系统使用heroku-php-apache2脚本启动 Apache2（包括 PHP-FPM）：

```
web: vendor/bin/heroku-php-apache2
```

如果使用 HHVM 运行环境，则使用heroku-hhvm-apache2脚本：

```
web: vendor/bin/heroku-hhvm-apache2
```

Nginx

启动 Nginx（和 PHP-FPM ），使用 heroku-php-nginx脚本：

```
web: vendor/bin/heroku-php-nginx
```

如果使用 HHVM 运行环境，则使用heroku-hhvm-nginx脚本：

```
web: vendor/bin/heroku-hhvm-nginx
```

默认

如果应用没有配置Procfile文件，系统会默认使用 Apache 作为 web 服务器，并使用composer.json里定义的运行环境（vendor/bin/heroku-php-apache2 或vendor/bin/heroku-hhvm-apache2 ）。

如何部署 Python 应用

如何部署 Python 应用

Flynn 使用 Heroku 构建包(https://github.com/heroku/heroku-buildpack-python)来支持 Python。

应用检测

当应用的根目录下包含requirements.txt 时，Flynn 会使用 Python 的构建包。当应用里包含manage.py时，Flynn 会认为这是个 Django 应用。当 Flynn 检测到 Django 应用时，会在编译过程中运行manage.py collectstatic命令。

环境依赖

Flynn 使用 pip 来解决 Python 应用的依赖问题。配置文件为requirements.txt，例如：

```
Flask==0.9
```

指定 Python 版本

Flynn 官方支持最新的 python-2.7 和 python-3.4,但理论上 2.4.4 到 3.4.1 之间的版本都支持，包含 PyPy。参见 构建包的 Github 页面(https://github.com/heroku/heroku-buildpack-python/tree/master/builds/runtimes)获取所支持版本的完整列表。

默认应用类型

这个构建包没有默认的应用类型，所以必须在应用根目录下的Procfile里指定。例如，部署 Gunicorn(http://gunicorn.org/) 应用时，Procfile里的配置如下:

```
web: gunicorn hello:app --log-file -
```

如何部署 Ruby 应用

如何部署 Ruby 应用

Flynn 支持多种 Ruby 解释器来实现对 Ruby，Rack，Rails 等类型应用的支持，如：MRI，JRuby(http://www.jruby.org/)，Rubinius(http://rubini.us/)。

Flynn 使用 Heroku Ruby 构建包(https://github.com/heroku/heroku-buildpack-ruby)来实现 Ruby 应用的检测，编译和发布。

应用检测

Flynn 通过检测根目录下的Gemfile文件来确定这是 Ruby 应用。通过 Bundler(https://bundler.io/)(一个 Ruby 的软件包管理工具)来检测，下载，安装应用所依赖的软件包。

软件依赖

Gems

Gemfile文件的主要用途是用来管理软件包的依赖关系。

下面是Gemfile文件的一个示例，其中部署的 Ruby 应用依赖 rack：

```
source "https://rubygems.org"

gem "rack"
```

在应用中附上上述 Gemfile后，应该运行bundle install来安装所需的软件包，这样会保存所有软件包内容和版本的快照到本地的Gemfile.lock文件。

在 Flynn 中部署 Ruby 应用时，必须通过 Gemfile.lock文件来确定哪些依赖的软件包需要安装。**如果没有这个文件，部署将会失败。**

关于Gemfile文件格式的详细信息，请参见：Gemfile 页面(http://bundler.io/gemfile.html)。

具体环境

可能有些 gems 只会在本地开发的时候用到，当部署到生产系统里时不需要。 这可以在Flynn 调用 bundle install时，使用--withoutdevelopment：test参数，这样在部署时就会跳过Gemfile里的development和test组的内容。

例如，你可能在本地开发时用到debugger和rspc，但在真正部署时很少需要，你可以将其放到上述组中：

```
group :development do
  gem "debugger"
end
```

```
group :test do
  gem "rspec"
end
```

针对上面的配置，下面是 Flynn 将会运行的完整命令：

```
bundle install \
  --without development:test \
  --path vendor/bundle \
  --binstubs vendor/bundle/bin \
  -j4 \
  --deployment \
  --no-clean
```

关于bundle install的详细信息，请参见 bundle 安装页(http://bundler.io/bundle_install.html)。

Ruby 解释器

如果应用需要特定的 Ruby 解释器，可以在Gemfile里指定。

使用 MRI v2.1.2：

```
ruby "2.1.2"
```

使用 JRuby 1.7.16 ，带 Ruby 2.0 支持：

```
ruby "2.0.0", engine: "jruby", engine_version: "1.7.16"
```

使用 Rubinius 2.2.10，带 Ruby 2.1 支持：

```
ruby "2.1.0", engine: "rbx", engine_version: "2.2.10"
```

原生扩展库

应用构建及运行时所在的容器包含了一些预安装的本地库，可用于编译 Ruby 应用的原生扩展，例如：

- libssl-dev
- libmysqlclient-dev
- libxml2-dev
- libxslt-dev

库文件的完整列表，参见：Dockfile 基础软件包页面 (https://github.com/flynn/flynn/blob/master/util/cedarish/Dockerfile)。

应用类型

在应用根目录下的 Procfile 里声明应用的类型，格式：TYPE:COMMAND。

如果没有Procfile，Flynn 会指定一个默认的应用类型(参见下文开发框架检测 (https://flynn.io/docs/how-to-deploy-ruby#framework-detection)获取更多信息)。

下面是一些常见的应用类型：

web

web类型的应用会包含 HTTP 路由和通信的端口设置，通常会启动一个 HTTP 服务器，常见的配置：

Thin

```
web: bundle exec thin start -p $PORT -e $RACK_ENV
```

Unicorn

```
web: bundle exec -p $PORT -c config/unicorn.rb
```

Puma

```
web: bundle exec puma -C config/puma.rb
```

注意：如果使用 Puma，确认在config/puma.rb里配置了ENV[PORT]端口变量。

worker

worker类型的应用通常是在后台运行的进程，用来处理队列任务，例如：

Resque

```
worker: QUEUE=* bundle exec rake resque:work
```

Sidekiq

```
worker: bundle exec sidekiq
```

Delayed::Job

```
worker: bundle exec delayed_job start
```

clock

clock类型的应用，通常用来按特定的周期执行定时（cron-like）任务。

Clockwork(https://github.com/tomykaira/clockwork)软件包提供了一个好用的 DSL ，可以按如下声明其应用类型：

```
clock: bundle exec clockwork lib/clock.rb
```

其中lib/clock.rb里包含 Clockwork 声明。

本地测试

如果想在本地测试应用的类型，需要安装Foreman(https://github.com/ddollar/foreman)，同时在根目录下的.env文件里设置必要的环境变量（例如：PORT=5000），启动 Foreman：

```
$ foreman start
14:25:33 web.1    | started with pid 42868
14:25:33 worker.1 | started with pid 42869
14:25:33 clock.1  | started with pid 42870
14:25:34 web.1    | == Sinatra/1.4.5 has taken the stage on 5000 for development with backup from Thin
14:25:34 web.1    | Thin web server (v1.6.3 codename Protein Powder)
14:25:34 web.1    | Maximum connections set to 1024
14:25:34 web.1    | Listening on localhost:5000, CTRL+C to stop
14:25:34 clock.1  | I, [2014-10-24T14:25:34.729860 #42870]  INFO -- : Starting clock for 1 events: [ frequent.job ]
14:25:34 clock.1  | I, [2014-10-24T14:25:34.729999 #42870]  INFO -- : Triggering 'frequent.job'
14:25:34 clock.1  | Running frequent.job
...
```

开发框架检测

不同的 Ruby 开发框架会有不同的配置，Flynn 可以检测在用的框架，并按需配置。

下面是 Flynn 检测常见框架的规则，并设置默认的应用类型（当Procfile文件没有声明应用类型时）：

Ruby

应用根目录下的Gemfile表明这是一个 Ruby 应用。

配置默认应用类型：

```
rake: bundle exec rake
console: bundle exec irb
```

Rack

在Gemfile.lock里包含rack包，表明这是一个 Rack 应用。

配置默认应用类型：

```
web: bundle exec rackup config.ru -p $PORT
rake: bundle exec rake
console: bundle exec irb
```

Rails 2

在Gemfile.lock里包含rails，版本大于等于2.0.0，并且小于 3.0.0，表明这是一个 Rail 2 应用。

配置默认应用类型：

```
web: bundle exec ruby script/server -p $PORT
worker: bundle exec rake jobs:work
rake: bundle exec rake
console: bundle exec script/console
```

Rails 3

在Gemfile.lock里包含rails，版本大于等于3.0.0，并且小于 4.0.0，表明这是一个 Rail 3 应用。

配置默认应用类型：

```
web: bundle exec rails server -p $PORT
worker: bundle exec rake jobs:work
rake: bundle exec rake
console: bundle exec rails console
```

Rails 4

在Gemfile.lock里包含rails，版本大于等于4.0.0，并且小于 5.0.0，表明这是一个 Rail 4 应用。

配置默认应用类型：

```
web: bin/rails server -p $PORT -e $RAILS_ENV
```

```
worker: bundle exec rake jobs:work
rake: bundle exec rake
console: bin/rails console
```

Assets

Flynn 会在应用编译的最后阶段运行自定义的assets:precompile任务。

如果 Flynn 检测到 Rails 3 或 4 的应用，但Gemfile里没有包含rails_12factor，系统会自动安装 `rails3*serve*static*assets*`*并且配置*`*config.serve*static_assets = true`。这样你的应用才能在 `public`目录里，对外发布这些资源。

执行任务

如果需要运行 Rake 任务，或者在应用里打开 Rails 终端，使用Flynn run。例如：

```
$ flynn run rake db:migrate
$ flynn run rails console
```

关于flynn run的详细信息，请参见命令行文档(https://flynn.io/docs/cli#run)。

数据库

PostgreSQL

PostgreSQL

Flynn Postgres 应用以一种自动精简配置的方法提供高度可配置的 PostgreSQL 9.4。Flynn 保证，在主服务器宕机时，故障自动迁移并保证数据不丢失

使用手册

APP 新增数据库

只要成功安装Flynn，Postgres 服务就手到擒来。 App 创建成功后，执行如下操作，为 App 新增数据库

```
flynn resource add postgres
```

执行成功后，Postgres 集群就为 App 新增了一个数据库，并且将你的 APP 连接到那个数据库

数据库连接

数据库创建成功后，Flynn 会为你的应用程序版本新增几个环境变量，比如 PGDATABASE、PGUSER、PGPASSWORD、PGHOST，Postgres 的客户端会通过这些参数，获取数据库连接详情。

同时，Flynn 提供 DATABASE_URL`` 环境变量给某些框架配置数据库连接时使用。

控制台连接

数据库如何连接 psql 控制台？本地无需安装 Postgres 客户端，无需修改防火墙/安全，只需执行

```
flynn pg psql
```

它是在 Flynn 集群中的容器内运行的。

备份和恢复

Flynn 命令行接口支持数据库导入、导出和故障修复功能。

$ flynn pg dump -f latest.dump

60.34 MB 8.77 MB/s

该文件既可用于数据修复，如执行```flynn pg restore```，也可用于导入到本地不在 Flynn 管理的其他 Postgres 数据库中，如执行 ```pg_restore```

$ pg_restore --clean --no-acl --no-owner -d mydb latest.dump

```flynn pg restore``` 命令将本地数据库文件装载到 Flynn Postgres 数据库中。Flynn 新建数据库/表对象之前，会先删掉数据库/表对象，避免数据库/表重建。

$ flynn pg restore -f latest.dump

62.29 MB / 62.29 MB [====================] 100.00 % 4.96 MB/s

WARNING: errors ignored on restore: 4

上面执行会产生一些告警信息，但无关紧要，可忽略。

恢复命令也可以用于从一个非 Flynn Postgres 的备份来恢复数据库，通过 ```pg_dump``` 来创建备份文件：

$ pg_dump --format=custom --no-acl --no-owner mydb > mydb.dump

####扩展
```

Flynn Postgres 提供多款扩展，包括：hstore，PostGIS，以及 PLV8 等等。
可以通过 ```CREATE EXTENSION``` 来使用扩展：

```
$ flynn pg psql
psql (9.4.1)
Type "help" for help.
bbabc090024fcdd118b04c50a0fb0d8c=> CREATE EXTENSION hstore;
CREATE EXTENSION
bbabc090024fcdd118b04c50a0fb0d8c=>
```

下面给出Flynn Postgres 完整扩展库列表：

名称	版本	描述
btree_gin	1.0	支持检索 GIN 中通用数据类型
btree_gist	1.0	支持检索 GIST 中通用数据类型
chkpass	1.0	用于自动加密密码的数据类型
citext	1.0	用于忽略大小写字符串的数据类型
cube	1.0	用于多维数据库的数据类型
dblink	1.1	用于在数据库内连接到其他 PostgreSQL 数据库
dict_int	1.0	整数的文本检索字典模板
earthdistance	1.0	计算地球表面上两点最短连线的距离
fuzzystrmatch	1.0	判断字符串之间的相似性和差异性
hstore	1.3	用于存储键值对(key-value)的数据结构

intarray	1.0	支持线性数组的函数、操作符和索引
isn	1.0	支持国际产品标准编号的数据结构
ltree	1.0	类层次树的数据结构
pg_prewarm	1.0	缓存关系数据
pg_stat_statements	1.2	回溯执行过的SQL语句的所有数据
pg_trgm	1.1	文本相似性检测以及基于三字母的检索
pgcrypto	1.1	加密函数
pgrouting	2.0	pgRouting 插件
pgrowlocks	1.1	显示行锁信息
pgstattuple	1.2	显示列数据
plpgsql	1.0	PL/pgSQL 过程式语言
plv8	1.4.2	PL/JavaScript (v8) 可信的过程式语言
postgis	2.1.5	PostGIS 几何、地理、栅格空间类型和函数
postgis_topology	2.1.5	PostGIS 拓扑空间类型和函数
postgres_fdw	1.0	为远程 PostgreSQL 服务器提供外键数据的存储
tablefunc	1.0	可操控所有表的函数，包括交叉表
unaccent	1.0	无 accents 的文本检索字典
uuid-ossp	1.0	生成UUID

此外，默认安装如下的检索词典扩展：

名称	版本
danish_stem	丹麦语的词干分析器
dutch_stem	荷兰语的词干分析器
english_stem	英语的词干分析器
finnish_stem	芬兰语的词干分析器
french_stem	法语的词干分析器
german_stem	德语的词干分析器
hungarian_stem	匈牙利语的词干分析器
italian_stem	意大利语的词干分析器
norwegian_stem	挪威语的词干分析器
portuguese_stem	葡萄牙语的词干分析器
romanian_stem	罗马尼亚语的词干分析器
russian_stem	俄语的词干分析器
simple	simple 字典：将大写转小写，检查 stop word('and', 'the'...)
spanish_stem	西班牙语的词干分析器
swedish_stem	瑞典语的词干分析器
turkish_stem	土耳其语的词干分析器

设计

Flynn Postgres 的设计理念是：

1. 写操作不得丢失，必须保证数据一致性。

2. 网络分区必须具有容错性，且没有脏数据。也不该出现 split-brain，以及数据修改的失败。

3. 一旦出现错误，在确保操作可安全执行前提下，Flynn Postgres 可自动转换配置，而无需人工干预

基于CAP(https://en.wikipedia.org/wiki/CAP_theorem)理论，该结果系统可称为 "CP" 系统。
Flynn Postgres 由三个或以上的 Postgres 实例组成：

• 集群中有一个是主实例，提供持续的读写操作。

• 主实例会将副本同步到一个名叫 *sync* 的实例中。客户端提交的写操作会先不会立马执行，除非他们加到了 sync 的操作日志(transaction log)中。

• 一个或者多个实例以异步的方式从 sync 中同步增量数据，这一个或者多个实例，就构成了一个实例链，在链中从它们上游的链接处异步复制数据。

• 当服务故障时，系统通过主动重置保证服务正常运行最大化，确保数据不丢失。

当服务故障或者维护时，集群可能无法处理写请求，和保证读请求一致性，但时间很短。但通过 sync 和 async 实例，最终会确保读请求的一致性。

如果主实例服务不可用， sync 实例会知晓，并让自己成为主实例，让 async 实例从它这里复制数据并成为新的 sync 实例。在新的 sync 实例生成前，无法进行写操作。因为在设计中考虑了许多安全方面的情况，所以转换过程中不会造成写操作的丢失和split-brain。

Postgres 内置了状态机功能，保证数据流正常执行，如先流式写，后日志同步；这些状态信息，在主实例中维护，并存储在discoverd。通过访问 discoverd DNS 和 HTTP API，可以获取当前主实例信息。

此设计很大程度基于先前 Joyent 在 Manatee state machine(https://github.com/joyent/manatee-state-machine) 中做的工作。

Flynn 集群默认配置了三个实例。如果一个实例不可用了，调度程序会创建一个新的实例，并且由主实例自动重新配置，无需人为干预。当用户运行命令flynn resouce add postgres时，会在默认的集群上创建一个新的用户和数据库。

稳定性

稳定性

稳定性

每个用户对『稳定性』需求都不尽相同。

对术语『production ready』的理解，则是仁者见仁智者见智。这里分享一下我们对它的理解。

我们认为软件应用只有当它真的可以应变各种生产环境时，才能被称为“production read”。比如：

软件在生产环境中对 SLA 没有负面的影响。软件发布后，不仅需要观察其稳定性，还需要考察软件生命周期内更新、监控、调试、备份和恢复的能力，并且这些操作都不能对线上产生负面影响。软件只有具备这些能力，我们才称它为『production ready』。

Flynn 的用户很多，由于他们的需求和评估方式不同，使用 Flynn 的方式也不尽相同。评估 Flynn 是否符合你的需求的最好办法就是试用它。

稳定性和功用性间该如何平衡，相信用户能做出正确的选择。我们不提供全套解决方案，因此你完全可以根据指定的需求，自主选择使用 Flynn 哪种功能，及如何使用。

当前我们提供 2 种发布渠道，分别是：**夜间版**和**稳定版**

夜间版包括部分经过边缘测试后合并到 Flynn 中代码。当然，这些修正都经过代码评审，并在持续集成系统中跑通，只差在“真实世界”测试过。

稳定版则是按周发布，以便我们有更多的时间来保证稳定性。虽然我们无法保证稳定版中没有任何 bug，或者非预期操作，但是我们将一如既往以高标准来完成它们。用户信任我们团队，信任

Flynn 系统，对我们来说尤其重要，也是我们奋斗的目标。可能会受到主流 Web 浏览器使用的发布系统模型的影响，发布频率可能会随着时间改变。

Flynn 当前存在一些安全隐患(https://flynn.io/docs/security)，评估时请慎重考虑。。

鉴于 Flynn 内置的 Postgres 数据库服务性能还未达到最优，对于数据频繁写、大数据场景，我们不推荐使用。

安全

安全

安全性

安全对 Flynn 来说尤其重要。

Flynn 是一个用于控制端到端的集成平台，因而我们可以内置了许多最佳实践，用户可直接使用，无需用户自定义实现这些较难的技术。

当然，我们觉得稳定性是第一位的。我们将很快完成一个足够稳定的 Flynn 版本，它能满足大多数用户对生产环境需求。详细请看 stability(http://https://flynn.io/docs/stability) 页面。当该项工作告一段落，我们将会为 Flynn 添加更多安全方面功能。

在那之前，我们认为解释一下Flynn 当前具备哪些安全特性是非常必要的。

分发的安全性

我们提供的所有的二进制文件，包括flynn-host 和 flynn命令行工具，以及 container 镜像，都是通过The Update Framework(http://theupdateframework.com/) 保障安全的分发。TUF 包含了一个鲁棒的、基于角色的签名系统，它可以防御许多攻击，包括降级攻击、CDN compromise 等。除了 TUF 之外，只能通过 HTTPS 来访问所需内容。

我们的 Vagrant 虚拟机镜像通过 HTTPS 提供服务，但并没有被署名，所以 Vagrant 不支持签名。

内部通信

Flynn 使用多个端口来进行内部通信，当前对于内部通信是没有认证的，所以绝对不可以将这些访问的端口暴露在外网上。主机必须配置防火墙，使得唯有 80 和 443 端口 Flynn 的端口可以访问，以防止被攻击。访问这些内部的 Flynn 端口需要 root 权限，所以请务必小心。

可以在 443 端口上通过 HTTPS 来访问控制器和仪表板，并生成一个 bearer token 进行认证。TLS 用于通信的证书在安装时就被生成了。证书的 hash 值作为命令行配置的一部分，以防止中间人攻击。

CA 证书仪表盘

在 Flynn 安装过程中，CA 证书会自动安装，并同时完成控制器和仪表板显示证书签署。CA 证书对应的私钥将立马被弃用，以防被滥用。CA 证书的作用是，让浏览器知道如何去处理多个用自己签署的证书连接到服务器的链接。CA 证书是仪表板提供给浏览器的，用来允许浏览器和仪表板、控制器之间的TLS通信加密。如果使用了 Flynn 安装器，CA 证书会在安装时通过 SSH 安全传输，防

止在访问仪表板时被攻击。如果没有使用安装器，证书可能会在第一次访问仪表板时通过不安全的链接进行传输，请不要在不安全的链接上安装证书。未来我们将使用Let's Encrypt(https://letsencrypt.org/)，这样既无需生成证书，也不用安装证书。

应用

在 Flynn 中运行的应用并非全是沙盒，这些应用可通过访问内部 Flynn API 来获取服务器的 root 权限，所以请不要在 Flynn 环境下运行不可信的代码。

Flynn 中可能存在一些其他未知的安全漏洞。若你的服务包含敏感数据，暂时我们不推荐通过 Flynn 托管。

未来几个月内将推出重要的更新，来提高 Flynn 的基础安全性。在那之后，可用性会被限制。未来 Flynn 将打造业内领先的安全策略。

报告问题

如果在使用 Flynn 过程中发现了一个没有被明确确认的安全漏洞，请立即邮件(https://flynn.io/docs/security@flynn.io)告知我们。

如果你可以使用 PGP，将你的邮件用下面的公钥加密：

```
pub   4096R/6913B2EF 2015-11-04 [expires: 2016-11-03]
      Key fingerprint = C334 DB91 6744 BD00 B347  0A86 0281 AD75 6913 B2EF
uid                  Flynn Security Team <security@flynn.io>
sub   4096R/38F74B09 2015-11-04 [expires: 2016-11-03]

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFY6QowBEADgRoqA7rwYc54npNAweozzylx4jIIFf6JcwxaCzc+zeHw9iAk+
PFMw7IdkWIA8A+2/sa11vPufCAa7OVzsk/YOYaZUPWv0khm1fO/CR0dmWoGB56jH
IWPUjtJpcEXp5j76qNp6U9VRcP/pfE5kTpwa8dFvOmtiwF1mDMAiMYedlz1eYfNW
hZbwJ6etMTd8SaAU/AP0rM+tDoRXtli6LOpkxRT3Bi4ykTwnhY1e2WYEvKPGqBvL
4YvxP1X+ZON7mgxbRwX01KyrhkLdks6nXmXltEewTy9uutz2oLiFWQyY+JC5C4PO
pLgTiaY+bXwow5SF52Ztc1bETdfYUbAfLWQ3bONjniGRGNSm3zT5mnc4x4eUXAf1
9XGvX7N3mWXzA+fHZF+WSuyDYGK8n8TsT9/rOaZryNSFWtjLxwXft8I4V5Sfm2hF
ljc/50fHAf39jAwTZwF8aAFqFSsNt5o0TWMt5fbYaOZA53zBEdOGT0+6HyoK7wRE
wdcRDLSHkoCSkzr8jTMs29ln/Dk79xFOyte3Pp5jDlm5biLeoF6dYYCUs/P2QiCE
j1ZSfvIdRd1Nl7OdtNblMSVboiJfGf0UTnqc6CWz3WnNNuL6D1GiyZgY0eenbzik
pVIqq+/EdarIu5RgP25ONkaZf8IVMI6Xwhx6HQTDlQCPpY39IQUS7StUywARAQAB
tCdGbHlubiBTZWN1cml0eSBUZWFtIDxzZWN1cml0eUBmbHlubi5pbz6JAj4EEwEC
ACgFAlY6QowCGwMFCQHhM4AGCwkIBwMCBhUIAgkKCwQWAgMBAh4BAheAAAoJE
AKB
rXVpE7LvRUEQAK4cX9lTSABXvlA9Ur7EUy5QnUOXgLjA3g125P4daLWQLjKGIjYF
Qengl+T/HED8QSiNF6au4Om3KbYutSqOEe3eyi1krnIILhLXp2p7eNVSmWic8Img
J2GDCfDxNxzaHKiuIy9cRmVLD9+nQBF7c3IHj5cvXHSROWKq4wgJUljuOs9s9VDN
```

GjCExWeeG5pF5J9HnmS1F+N21BK8E2KTzouRhVmJ8fpaxJy7Ofr9N6WyxbZCwwwD
QMamo1mczo0p/+cUeQik92D2tz1Pn+RiF/Ooq16RKsls8N8poR9ffQ0rkE2hksjB
3+OIaZ3Q0vKUBowUu6pmPSfkorq29zy45V3nku9Ly8K0B/t1Xl/wUQaY/OBm9Iiy
eJ6p2NBc8U8Xcl33Xe7t5r9nCVrhzEDoggcbCo6YjcGQJwzgwhgbwtO0HDIcax3z
y2j7pfmABwJifmyMAD6UxAqftXcBxKXT9+jxa6skqFAid2zceqn1V/M83tQ28gLa
/IssVSJ0DbjPpoPOSjFV+b0lKzre12EcsTVvDsm9u0jUSH+XVJA5o8cZFU8WCAs7
h9x6yhrE+n3AX3ZP8zUSv8mSH1Eh1EAolsnqBBA3qiC+b6l/KcVbYjo3bv2jw0IZ
5xSG8hOdUShKpKkPc4CiY5Z5c75TibA4+UaAvDELXUsOkU0eBoD/fbUJuQINBFY6
QowBEADMK3ES9lEjDyhV43WB8oNizzA0fbf3k3giyOneAZz7VgP8BPy8xvI5zkjU
tjDJfCKc5Yk+3pDJRgi+7u2O0KdF7VVJ+YnHRfPib4YB033fbhDTa15qa7Mr3uDr
EyTyUzZ0tVuDAnvSz8To6Z2HWOynBFB7N31plhN/xubMCfqH20l6ZtgKszdZ6pVR
xD0BZ47JhD9JcZF1Xa1tgASAQ286XCVkOwxRnSGmnNCc+HjAqepbKMbJgtTCOLRK
Zx1I9jIikAakwxbzXtv7gFzYWrsGWPEKEtKCdmhD2V6Zvl5nwBkr7nS/JQQgzWXM
/ltcG3Np0qJRkXA2ZSlhh21bOgfjHvfijRuxAPWlJv21qzN0nzBpLXtu3XntnzFP
BR1u7HW5hfGXIbRUbmkiJ/5j29QpbXn+beYmGUH0ukvGpAqIyHDIgiTqLYMErePZ
aY97tx/XTVsBKJmnGnsYUe2TIdIEpcKe0JSijOC8AVPBjm3A8Mna8169q3s7ARxC
NfpBOelExfdxWcBprTUkqwK13vX3X2FgOHXH3LPqMzwuuh9QMtr0tvyG+g/8KsAP
1+va1Zi4gBUxB2PeTdxkSbue4apctKEOsEhbMjvBKWQ/Ip3hSUAvFiUwBK0IsP1k
sv8T1vIpYBInHtkUUSDgKwn2X7KQ/khTKtwmNjHFGbnyCOdUVwARAQABiQIlBBgB
AgAPBQJWOkKMAhsMBQkB4TOAAAoJEAKBrXVpE7Lvlq8P/2RwDa47yQO31eP0cYyf
l8SdJFkTvPx8hy6IOgm5p1xOOGXHkVWioG8R8KQMNPLAdu7g6hTmboE69XNuLox/
9T9YyTN5TkUxh/9uJmupahN/hbS9aomZsznIBIan6I6QzSk17UNXm8rY5hIKB9qb
4JLKZq8tSMTGhUAyKbqeLbClvRM/LTFrq+J+FKMOrBdW5BjfdNTYLf7w1yAVLThI
TDsu6epdKpV2kG1/cp0QJQssbePxe6xvZ9PeWI5axGN0A74pIKWSn5K5tP9DpDf1
FfJcx9obzIzOAffO6ID6mn1Rc20Yu3NSW0cvB3TOv97jdviSoq3eP1v7pgqfJtc2
ZWXa9hIlIYv6bx0ukkSYuERHoi3SVFZMiVeOTddOPKAKy2vWzQRt9S/mIDh32PmD
oNvPfTIdRGivYzKqTzIkjB73Vq4Jn2BflQoyAoEu8BzI9/oYATke2TpYmcsYgh9r
03zbex98lF/rIhySMuJDp20/FHsJUMZMnfxv/NgN/A2wotSA5/idvrBUwVkdNsKU
MpVfhFfhxSvkMXofbcSSAsRX1+r8S3BAQrnqV2fDzBnJqmAQ8CUTYQheZ8iMDdzJ
47wRWTZBgqCCedNOBN6TSmQGiwGhZKVKxMfIORp+1FgLEl/2FiJVoi3736SagKGG
aOmKnAD2rS4Lu4+Ez2pTZFz9
=yreI
-----END PGP PUBLIC KEY BLOCK-----

贡献

贡献指南

贡献指南

我们欢迎和鼓励社区朋友为 Flynn 共享自己的力量。

由于该项目尚未稳定，所以后续开发过程会朝着某些侧重点依次展开。若 Flynn 还未具备 "production ready" 特性，那么某些还未纳入研发排期的需求将不予接受。

请在贡献代码前熟悉本贡献指南和项目线路图。

除了贡献代码外，当然还有很多方法可以共建 Flynn：

• 修复 bug 或 提出问题

• 改进文档(https://github.com/flynn/flynn.io)

贡献代码

除非你是在修复一个已知的 bug，否则我们强烈建议，在你开始提交代码前，通过 GitHub issue、IRC(irc://irc.freenode.net/flynn) 或者email(mailto:contact@flynn.io)和 Flynn 核心团队探讨，以便确定你的工作，同 Flynn 技术路线以及架构是否一致。

所有的贡献通过 pull request 来完成。注意，所有贡献者修正的补丁都会经过严格审查。当一个 pull request 完成后，其他贡献者可以进行反馈，如果补丁通过审查，那么维护者就会接受该补丁并进行标注。若 pull request 测试未通过，那么作者需要更新 pull request 处理错误直到测试通过并且 pull request 合并成功。

所有的补丁都至少被一个维护者审查过（即使补丁的作者是维护者）

审查者通过补丁的时候应当标记 “LGTM” 。如果补丁是由一个维护者提交并且有了写权限，pull request 应该在审查通过后由提交者进行合并。

代码风格

编程式，请遵循下列规则：

• Go 语言的代码应该匹配 gofmt -s 的输出

• Shell 脚本应当遵循 Google shell 编程规范(https://google-styleguide.googlecode.com/svn/trunk/shell.xml)

开发者的源证书

所有贡献必须接受 DCO

Developer Certificate of Origin
Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.
660 York Street, Suite 102,
San Francisco, CA 94110 USA

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
have the right to submit it under the open source license
indicated in the file; or

(b) The contribution is based upon previous work that, to the best
of my knowledge, is covered under an appropriate open source
license and I have the right under that license to submit that
work with modifications, whether created in whole or in part
by me, under the same open source license (unless I am
permitted to submit under a different license), as indicated
in the file; or

(c) The contribution was provided directly to me by some other
person who certified (a), (b) or (c) and I have not modified
it.

(d) I understand and agree that this project and the contribution
are public and that a record of the contribution (including all
personal information I submit with it, including my sign-off) is
maintained indefinitely and may be redistributed consistent with
this project or the open source license(s) involved.

每次提交前，只需你的姓名和邮箱地址后，添加下面的信息就可以接受 DCO （使用 git commit -s ）

```
Signed-off-by: Jane Example <jane@example.com>
```

因为法律原因，不接受匿名和假名的贡献（如果这是个问题请与我们联系(contact@flynn.io)）

Pull request 程序

在申请 pull request 前，请确保你有 GitHub 账户，如果你对这些不了解，请看 GitHub forking(https://help.github.com/articles/fork-a-repo) 和 pull request(https://help.github.com/articles/using-pull-requests) 文档。通俗的讲，Pull request 就是所谓的主干开发。新建 pull request 之前，请检查下列事项：

1. 基于master创建一个特定分支，这样就不会搞乱了。
2. 将本地代码合并到master上，并将本地重置(http://git-scm.com/book/en/Git-Branching-Rebasing)为新的基线版本。
3. 用命令go test ./...进行全项目的测试，并确保通过。
4. 运行gofmt -s（如果项目是用 Go 语言写的）。

5. 在所有提交中接受开发者的源证书。

6. 确认每个提交都有一个子系统前缀（例如：controller: ）

Pull requests 被看作是 “review requests “，维护者会对补丁的风格和实质进行反馈。

通常，所有的 pull request 必须功能测试通过才行。偶尔也会出现某个新功能，很难测试，这种情况下，请在提交消息中标记处并说明缘由。

讨论

我们在 Freenode 上使用 #flynn IRC(irc://chat.freenode.net/flynn) 进行讨论。欢迎大家加入，提问，讨论bug。你可以通过BotBot.me(https://botbot.me/freenode/flynn/) 登录，若你本地未安装 IRC 客户端，也可通过webchat(https://webchat.freenode.net/?channels=flynn)登录。

管理

不管你是 Flynn 老朋友，还是新朋友，我们专注于将这个社区打造为一个安全的社区并得到你的青睐。

• 我们致力于向所有人提供一个友好、安全、包容的环境，无论性别、性取向、残疾与否、种族、宗教信仰或类似的个人特征。

• 请不要使用昵称，这会破坏大家社区的友好、安全和包容的环境。

• 请保持善良和谦虚美德，不要变的那么尖酸刻薄和粗鲁。

• 如果你对别人进行辱骂和骚扰，我们将对你禁言。特别的，我们不会容忍排挤社会中边缘化的人的行为。

• 对别人的骚扰也是不可容忍的。无论你是谁，只要你觉得自己被社区成员骚扰，请立即联系该频道的 ops 或者 Flynn 核心团队的成员。

• 同样的，任何发送垃圾邮件、钓鱼等意在窃取的行为都是禁止的。

我们欢迎来讨论如何为社区提供一个友好的、安全的生产环境。如果你有任何问题、反馈或者建议，请联系(contact@flynn.io)我们。

开发

开发指南

开发

本指南将会按照如下流程，教你如何运用 Flynn 开发：

• Flynn 源码修改

• Flynn 构建和执行

• Flynn 调试

• 新建 Flynn 发布版本

开发环境

开发任务通常在 Vagrant(https://www.vagrantup.com/) 管理的 VirtualBox(https://www.virtualbox.org/) 虚拟机内完成的，Flynn 使用 Vagrantfile 自动创建虚拟机。

运行开发虚拟机

如果你还没安装 VirtualBox 和 Vagrant，请按照官网安装文档，自主安装。

把 Flynn 的源码复制到本地：

```
$ git clone https://github.com/flynn/flynn.git
```

接着在 flynn 目录下，启动虚拟机：

```
$ vagrant up
```

若你是第一次创建虚拟机，Vagrant 会下载 VirtualBox 下的文件，大约 1G 大小，会占用你部分时间，时间长短，由网络带宽决定。

一旦 vagrant up 命令执行成功，你就可通过 SSH 方式连接虚拟机：

```
$ vagrant ssh
```

从现在开始，若无特别说明，接下来所有命令都会在虚拟机中运行。

修改代码

开发虚拟环境通过挂载物理机/vagrant目录实现 Flynn 源码共享。这意味着你可以在本地物理机修改文件，并且这些变化在虚拟机内也能看见。

因为 Flynn 主要由 Go 写成，源代码需要一个可用的 Go 工作空间。开发虚拟机有一个$HOME/go 的GOPATH，Flynn 的源代码是从 /vagrant 到 $GOPATH/src/github.com/flynn/flynn 的一个软链。

如果你没有特殊的问题需要修复，但又对贡献这个社区很感兴趣，你可以先看一下 GitHub 被标记为 easy(https://github.com/flynn/flynn/labels/easy) 的问题。

Flynn 搭建

我们在 tup(http://gittup.org/tup/) 构建系统里执行命令即可完成 Flynn 多组件安装。

搭建 Flynn 只需运行make：

```
$ make
```

该命令会构建 GO 二进制文件，创建个Docker镜像等工作。若你对某个环节感兴趣，可查看子目录下的Tupfiles。

若构建失败，tup 将会输出错误并退出。修复后，重新运行make即可。

如果你想重新编译 Go 二进制文件，运行make clean。

一旦 tup 成功运行，你会有许许多多的 Go 二进制文件和 Docker 镜像，这些对运行 Flynn 都是有用的。

Flynn 运行

一旦 Flynn 系统依赖组件搭建完成，执行如下脚本即可创建Flynn 集群节点：

```
$ script/bootstrap-flynn
```

这个命令将会做如下几件事：

- 停止flynn-host守护进程，以及所有正在运行的 Flynn 服务
- 开启flynn-host守护进程
- 运行 Flynn 加载程序，该程序会开启所有 Flynn 服务

本脚本还提供了其他选项，供用户使用他方式开启 Flynn 后台服务，比如使用其他外部IP的eth0设备。请通过script/bootstrap-flynn -h查看所有功能选项。

一旦 Flynn 提供服务，使用bootstrap 输出添加集群flynn CLI工具，请尝试修改(比如遵循指南(https://github.com/flynn/flynn#trying-it-out))。

调试

如果没有按预期运行，使用如下命令工具进行调试：

查看 flynn-host 守护进程日志

```
$ less /tmp/flynn-host.log
```

查看运行任务列表

```
$ flynn-host ps
ID                                STATE    STARTED          CONTROLLER APP
CONTROLLER TYPE
flynn-66f3ca0c60374a1abb172e3a73b50e21              running  About a minute ago
example         web
flynn-9d716860f69f4f63bfb4074bcc7f4419              running  4 minutes ago
gitreceive      app
flynn-7eff6d37af3c4d909565ca0ab3b077ad              running  4 minutes ago
router          app
flynn-b8f3ecd48bb343dab96744a17c96b95d              running  4 minutes ago
blobstore       web
...
```

查看所有的任务（运行中 + 停止）

```
$ flynn-host ps -a
ID                                  STATE    STARTED          CONTROLLER APP
CONTROLLER TYPE
flynn-7fd8c48542e442349c0217e7cb52dec9          running  15 seconds ago
example         web
flynn-66f3ca0c60374a1abb172e3a73b50e21          running  About a minute ago
example         web
flynn-9868a539703145a1886bc2557f6f6441          done     2 minutes ago
example         web
flynn-100f36e9d18849658e11188a8b85e79f          done     3 minutes ago
example         web
flynn-f737b5ece2694f81b3d5efdc2cb8dc56          done     4 minutes ago
flynn-9d716860f69f4f63bfb4074bcc7f4419          running  5 minutes ago
gitreceive      app
flynn-7eff6d37af3c4d909565ca0ab3b077ad          running  5 minutes ago
router          app
flynn-b8f3ecd48bb343dab96744a17c96b95d          running  5 minutes ago
blobstore       web
...
```

查看任务输出

```
$ flynn-host log $JOBID
Listening on 55006
```

检查任务详情

```
$ flynn-host inspect $JOBID
ID                      flynn-075c21e8b79a41d89713352f04f94a71
Status                  running
StartedAt               2014-10-14 14:34:11.726864147 +0000 UTC
EndedAt                 0001-01-01 00:00:00 +0000 UTC
ExitStatus              0
IP Address              192.168.200.24
flynn-controller.release    954b7ee40ef24a1798807499d5eb8297
flynn-controller.type       web
flynn-controller.app        e568286366d443c49dc18e7a99f40fc1
flynn-controller.app_name
```

停止任务

```
$ flynn-host stop $JOBID
```

停止所有任务

```
$ flynn-host ps -a | xargs flynn-host stop
```

停止某个APP所有任务

假设APP名字叫 example：

```
$ flynn-host ps | awk -F " {2,}" '$4=="example" {print $1}' | xargs flynn-host stop
```

向 GitHub 的 gist 上传日志和系统信息

如果你想帮助诊断你系统上的问题，运行下面命令来上传一些有用的信息到一个匿名的 GitHub gist

```
$ flynn-host collect-debug-info
INFO[03-11|19:25:29] uploading logs and debug information to a private, anonymous gist
INFO[03-11|19:25:29] this may take a while depending on the size of your logs
INFO[03-11|19:25:29] getting flynn-host logs
INFO[03-11|19:25:29] getting job logs
INFO[03-11|19:25:29] getting system information
INFO[03-11|19:25:30] creating anonymous gist
789.50 KB / 789.50 KB
[==========================================================]
100.00 % 93.39 KB/s 8s
INFO[03-11|19:25:38] debug information uploaded to:
https://gist.github.com/47379bd4604442cac820
```

接着你可以在#flynn IRC 聊天室内附上 gist 信息 ，便于得到聊天室内其他人帮助。

如果你不想用 GitHub gist 服务，或者你的 log 文件太大不能放在一个单独的 gist 中，你可以通过 --tarball 选项创建信息的 tar 包：

```
$ flynn-host collect-debug-info --tarball
INFO[03-11|19:28:58] creating a tarball containing logs and debug information
INFO[03-11|19:28:58] this may take a while depending on the size of your logs
INFO[03-11|19:28:58] getting flynn-host logs
INFO[03-11|19:28:58] getting job logs
INFO[03-11|19:28:58] getting system information
INFO[03-11|19:28:59] created tarball containing debug information at /tmp/flynn-
```

```
host-debug407848418/flynn-host-debug.tar.gz
```

同 IRC 内交谈的时候，可以附上这些信息。

运行测试

Flynn 有两种测试方式

- "单元"测试，运行go test
- "集成"测试，运行在启动了的 Flynn 集群上

运行单元测试

运行所有单元测试

```
$ go test ./...
```

给一个单独的组建运行单元测试（例如：router）

```
$ go test ./router
```

给一个组建以及它的子组建运行单元测试（例如：controller）

```
$ go test ./controller/...
```

运行集成测试

集成测试在tests目录中，在测试可以运行前，需要一个正在运行的 Flynn 集群

运行所有的集成测试

```
$ script/run-integration-tests
```

它会做下列的事情：

- 运行make建立 Flynn
- 通过script/bootstrap-flynn来启动 Flynn 集群节点
- 运行集成测试（bin/flynn-test）

运行一个单独的集成测试（例如：TestEnvDir）

```
$ script/run-integration-tests -f TestEnvDir
```

Pull request

一旦你修改了 Flynn 的源代码，并且测试了你的新特性，你可以在 GitHub 上开一个 pull request，这样我们就可以审查你的新特性，并将它们合并到 Flynn 中。

请看 贡献指南(https://flynn.io/docs/contributing)

Flynn 发布

一旦 Flynn 在你的开发虚拟机内搭建并测试通过，你就能创建一个发行版本并且在其他主机上安装这些组件。

一个 Flynn 发行版本是一系列组件的集合，包括二进制文件、配置文件、文件系统镜像，这些必须按顺序安装，才能保证 Flynn 正常运行。

更新框架 (TUF)

Flynn 使用更新框架(http://theupdateframework.com/)（即TUF） 安全分发所有 Flynn 组件。

Flynn 新版本发布，使用 go-tuf(https://github.com/flynn/go-tuf) 库自动生成一个 TUP 仓库

参考安装指引(https://github.com/flynn/go-tuf#install)内的“Create signed root manifest“(https://github.com/flynn/go-tuf#examples)例子。保证代码目录结构如下所示：

```
.
├── keys
│   ├── snapshot.json
│   ├── targets.json
│   └── timestamp.json
├── repository
└── staged
    ├── root.json
    └── targets
```

TUF 根密钥需要编译进 Flynn 的发布版本中，镜像 URL 必须和 TUF 仓库将上传的文件服务器相关。可以通过在tup.config中设置参数CONFIG*TUF*ROOT_KEYS``和```CONFIG*IMAGE*REPOSITORY```并再次运行```make```来完成。例如：

```
CONFIG_IMAGE_REPOSITORY=https://s3.amazonaws.com/my-flynn-repo/tuf
CONFIG_TUF_ROOT_KEYS=[{"keytype":"ed25519","keyval":{"public":"31351ecc833417968faabf98e004d1ef48ecfd996f971aeed399a7dc735d2c8c"}}]
```

导出组件

将 Flynn 组件导出到 TUF 仓库（这需要提示输入 TUF 密钥的 passphrase）

```
$ script/export-components /path/to/tuf-repo
```

上传组件

上传 TUF 仓库的repository目录到CONFIG*IMAGE*REPOSITORY指示的文件服务器。

例如，如果使用 S3，并且CONFIG*IMAGE*REPOSITORY设置为https://s3.amazonaws.com/my-flynn-repo/tuf，使用s3cmd来同步文件：

```
$ s3cmd sync --acl-public /path/to/tuf-repo/repository s3://my-flynn-repo/tuf
```

现在你可以分发script/install-flynn，并且用一个明确的仓库 URL 来安装用户建立的 Flynn 组件：

```
install-flynn -r https://s3.amazonaws.com/my-flynn-repo
```

常见问题

如果部署失败，该如何调试？

如果部署失败，该如何调试？

你可以在 Flynn 的控制板里，查看你所有的应用日志记录，包括部署日志。打开你的应用控制板，点击 *查看日志（Show Logs）*，然后选择 *部署日志（Deploy logs）* 下的最新条目。

图片地址：https://static.oschina.net/uploads/img/201512/02205949_jgam.png

如何更新 Flynn？

如何更新 Flynn ？

有两种更新 Flynn 的方法：原地更新和备份/还原。因为原地更新是一种比较新的方式，并不完全稳定。所以，最安全的更新方式还是备份/还原。

备份/还原

备份/还原这种更新方法是：先将集群进行完整备份，再用新版的 Flynn 将备份还原到一个新的集群。更新步骤：

1. 用 flynn cluster backup --file backup.tar 命令对集群进行完整备份。

2. 按照手动安装说明(https://flynn.io/docs/installation/manual)在新的集群上安装新版 Flynn，但不要执行启动那一步。

3. 在运行 flynn-host bootstrap 命令时，添加选项来指向集群的备份文件：flynn-host bootstrap --from-backup backup.tar

4. 更新指向旧群集的 DNS 记录，使其指向新集群。

原地更新

原地更新是一种新的更新方法，操作过程中可能会出现无法预料的问题，我们建议在进行更新操作之前先用 flynn cluster backup 命令对集群进行完整备份。集群进行更新时几乎可以实现零停机，但 Postgres 集群在更新时，可能会有几秒钟不可用。

要原地更新整个集群，运行 flynn-host update 命令。

如果部署故障，该如何处理？

如果部署故障，该如何处理？

Flynn 能够自动检测出部署故障。这时，你可以部署一个新的修正版本，或者更改原来的应用环境设置，然后重新部署。

Flynn 中如何查看日志？

Flynn 中如何查看日志？

Flynn 会收集应用进程的所有日志。在控制板界面，你可以实时查看每个进程的日志，同时，CLI 会为你提供更多功能：

```
# 查看所有进程的日志
$ flynn log
# 实时跟踪日志
$ flynn log -f
# 单独查看web日志
$ flynn log -t web
```

Flynn 中如何使用自定义域名？

Flynn 中如何使用自定义域名？

默认情况下，Flynn 集群下所有的主机名（包括应用程序）都会使用 flynnhub.com 域，比如：wordpress-master-master.evoa.flynnhub.com 。要使用你自己的域名，最简单的方法是为每个应用程序添加一个别名（CNAME）。例如：

```
wordpress.mydomain.com. IN CNAME wordpress-master-master.evoa.flynnhub.com
```

同时，你还需要在控制板界面或者通过 CLI，为你域名中的应用添加路由。

```
# 要添加路由，可使用 `flynn route` 命令
$ flynn route add http wordpress.mydomain.com
```

图片地址：https://static.oschina.net/uploads/img/201512/02211930_0Djf.png

如果你要完全弃用 flynnhub.com 域，你需要在你自己的域中，为 Flynn 集群中的每台主机添加一个 A 记录。例如：

```
flynn.mydomain.com. IN A 10.0.0.250
flynn.mydomain.com. IN A 10.0.0.251
```

```
flynn.mydomain.com. IN A 10.0.0.252
```

然后，你可以将你的 CNAME 指向这个主机名，例如：flynn.mydomain.com。注意：无论你在 Flynn 集群中添加或者删除一台主机，你要确保同步更新你的 DNS 映射。

集群故障调试

集群出现故障，该如何调试？

Flynn 服务分布在你整个集群主机上，它们大多数在容器中运行。收集必要的日志和主机信息最简单的方法是 SSH 到任一台 Flynn 群集主机，以 root 用户运行 flynn-host collect-debug-info 命令。

```
$ sudo flynn-host collect-debug-info
```

这个命令，会将你所有的日志上传到一个匿名节点上，如果你熟悉的话，也可以上传到你自己指定的节点；或者发布到我们的 IRC 频道上与 Flynn 的开发者共享，这是我们在 Freenode 上的 #flynn 频道(https://webchat.freenode.net/?channels=flynn) 。

如何使用 CLI 管理多个 Flynn 集群？

如何使用 CLI 管理多个 Flynn 集群？

Flynn 的命令行工具默认支持多集群管理。要管理某个特定的集群，使用 -c 标记，并在后面加集群的名字。

```
# 列出 production 集群上的所有应用
$ flynn -c production apps
```

如何重启单个组件？

如何重启单个组件？

首先通过 flynn ps 命令得到应用的独立进程列表，其中也包括 Flynn 的内部服务进程。然后，将得到的进程 ID 传递给 flynn kill 命令，杀掉这个进程。这时，Flynn 将自动重启这个被杀掉的进程。

```
# 获取进程列表
$ flynn -a myapp ps
ID                                   TYPE    RELEASE
host-28a16c12-6136-4e06-93b1-2b014147de79  web   ace81d3d-93f5-4df3-b364-
55f05cb908c3
# 杀掉一个进程
$ flynn -a myapp kill host-28a16c12-6136-4e06-93b1-2b014147de79
```

```
Job host-28a16c12-6136-4e06-93b1-2b014147de79 killed.
```

如何将日志传送到日志聚合管理工具，比如 logstash？

如何将日志传送到日志聚合管理工具，比如 logstash？

Flynn 中的应用程序可以使用系统或客户端提供的库，将日志发送给远端日志（syslog）服务器或日志聚合服务器，比如 Logstash。大多数流行的编程语言都内置提供了支持远程日志记录的功能，例如 Python 的 SysLogHandler。

如何使用自定义构建包（buildpacks）？

如何使用自定义构建包（buildpacks）？

Flynn 使用 Heroku buildpacks 来准备和生成应用程序。Flynn 自动为支持的语言选择一个标准的构建包。当自动选择不可行，或选择的标准构建包功能不合适时，你也可以手工指定一个构建包。要自定义构建包，只需在启动或者部署新版本之前，在控制板界面设置好环境变量 BUILDPACK_URL。之后，Flynn 会自动下载指定的构建包并用它来构建你的应用程序。

图片地址：https://static.oschina.net/uploads/img/201512/03155020_eM53.png

如何得到控制板界面的登陆令牌（login token）？

如何得到控制板界面的登陆令牌（login token）？

Flynn 安装程序在运行成功后会为你提供一个登录令牌。如果令牌遗失，你可以通过 CLI 工具将它找回。 注意，本操作仅限于已用 flynn cluster add 命令配置好的集群。

```
$ flynn -a dashboard env get LOGIN_TOKEN
0ff8d3b563d24c0d02fd25394eb86136
```

如何在 Flynn 中使用其它类型的数据存储？

如何在 Flynn 中使用 redis/memcache/mysql 等其它类型的数据存储？

Flynn 应用程序可以与部署在 Flynn 集群以外的服务器，包括 MySQL, Redis, 和 Memcached 进行正常的通信。也可以直接在 Flynn 中通过 Docker 镜像来跑一个数据库或者其他类型的数据存储，但目前 Flynn 只支持 Docker 镜像的临时存储卷，因而数据将无法持久化。虽然这种用法适合跑 Memcached，但我们还是建议通过外部服务，如亚马逊的 RDS 或自托管服务，来支持数据存储。

容器之间如何通信？

容器之间如何通信？

Flynn 为其运行的所有应用程序提供 DNS 解析。对于内部的和跨容器的通信，Flynn 提供本地的 TLD .Discovered。

对于 Web 进程，Flynn 使用 <app name>>-web.discoverd 模式，比如：blog-web.discoverd。

对于数据存储，Flynn 使用集群 leader 选举算法，该模式是 <app name>.discovered，如要与当前集群 leader 通信，使用 leader.<app name>.discoverd。

如何用 CLI 工具注册 Flynn 集群？

如何用命令行工具注册 Flynn 集群？

当一个集群引导启动后，它会创建一个自签署证书。Flynn 会为该证书提供一个指纹，该指纹须传给命令行工具，用来添加该集群。虽然在集群创建时，同时也生成控制器密匙，但它可以被改写或另外再添加。

```
flynn cluster add -g <git host> -p <tls pin> <cluster name> <controller URL>
<controller key>
```

- -g <git host>　指定控制器的 URL。
- -p <pin>　证书的 pin 值
- -cluster name　可为任意名
- -controller url　格式为：controller.[你使用的域]
- -controller key　控制器认证密钥。

例如：

```
flynn cluster add -g dev.localflynn.com:2222 -p
KGCENkp53YF5OvOKkZIry71+czFRkSw2ZdMszZ/0ljs= default
https://controller.dev.localflynn.come09dc5301d72be755a3d666f617c4600
Cluster "default" added and set as default.
```

在 Flynn 上如何部署 Docker 镜像？

如何将 Docker 镜像部署到 Flynn？

Flynn 支持从 Docker 注册库运行 Docker 镜像。但是请注意，当前在 Docker 镜像上申明的容量都是暂时的，一旦实例被重启或者缩小，容量就会被清除。

首先给应用程序创建一个配置文件，然后运行命令来指定任务类型、 初始化命令以及开放端口。

创建一个名为 config.json 的文件，并将该文件通过 -f 选项传给 flynn release add 命令。

```
{
     "processes": {
      "server": {
        "cmd": ["memcached", "-u", "nobody", "-l", "0.0.0.0", "-p", "11211", "-v"],
        "data": true,
        "ports": [{
          "port": 11211,
          "proto": "tcp",
          "service": {
            "name": "memcached",
            "create": true,
            "check": { "type": "tcp" }
          }
        }]
      }
    }
  }
```

Flynn 需要得到 Docker 镜像的注册地址。得到该注册地址最简单的方法是从 Docker 注册表中拉出镜像，然后用 Docker CLI 打印镜像ID。

`

$ docker pull memcached

$ docker images --no-trunc	grep memcachedmemcached latest 8937d6d027a4f52b877aacd9faa68b8a89652d858845e1ea2cf7bc6a8306db00 13 days ago 132.4 MB

现在可以通过 Flynn CLI 创建应用程序，并用 Docker 注册地址创建发布，然后通过扩展启动服务。

创建应用程序

$ flynn create --remote "" memcached

使用注册地址创建发布

$ flynn -a memcached release add -f config.json "https://registry.hub.docker.com?name=memcached&id=8937d6d027a4f52b877aacd9faa68b8a89652d858845e1ea2cf7bc6a8306db00"

通过扩展启动服务。

$ flynn -a memcached scale server=1

`

注意，因为 Docker 镜像启动前，Flynn 必须在下载 Docker 镜像，所以最后一个命令也许会超时，但是 Flynn 最终会启动。你可以通过运行 flynn -a memcached ps 命令，在启动 Docker 镜像的同时，打开进程列表来查看它的启动情况。

现在， Flynn 应用程序可以通过连接到 memcached.discoverd:11211 与新的 memcached 服务器通信。应用程序的主机名基于 service.name 密钥，该密钥在发布提供的配置文件中。

如何 SSH 到 Flynn 集群的主机？

如何 SSH 到 Flynn 的集群主机？

Flynn 安装程序会生成一个 SSH 密钥对，并在所有主机的 Root 账户中配置这个公钥。该公钥的位置在 ~/.flynn/installer/keys 中。密钥默认名是 flynn。

要登陆到 Flynn 服务器，你可以运行：

```
$ ssh -l root -i ~/.flynn/installer/keys/flynn <ip>
```

如何将配置传递给应用程序？

如何将配置传递给应用程序？

推荐通过设置环境变量的方式将配置传递到你的应用上去，这个跟平常在 Herok 或 Docker 中的处理类似。

Flynn 直接从 git 仓库中提取代码进行应用部署，没有类似于 Chef 或 Puppet 这样的配置管理系统来处理配置文件和存储配置。在 Git 中并不鼓励这么做。

这种方式带来的好处是可支持多种部署环境，并且无需为每一个环境设置专门的配置文件。如果更改配置，也无需经历代码更改、提交、代码审查和部署的这样的周期。

你可以用 CLI 工具的 flynn env set NAME=value 命令来设置一个配置变量，例如：

```
# 在 myapp 中设置环境变量 SECRET
flynn -a myapp env set SECRET=thisismysecret
```

在 Flynn 上设置环境变量将会创建一个新的应用版本，这将重启该应用的所有进程。

API 扩展

命令行工具

命令行工具

输入 flynn help 列出所有可用的命令。

安装

OS X 和 Linux 系统中，在终端运行以下命令：

`

L=/usr/local/bin/flynn && curl -sSL -A "`uname -sp`" https://dl.flynn.io/cli	zcat >$L && chmod +x $L
Windows 系统，在 PowerShell 中运行以下命令：	
(New-Object Net.WebClient).DownloadString('https://dl.flynn.io/cli.ps1')	iex
## 应用程序管理 ### 用法：	

flynn apps

列出所有应用程序。 ### 例子

$ flynn apps

ID	NAME
f1e85f5392454a329929e3f27f7a5644	gitreceive
4c6325c1f13547059e5496c91a6a97dd	router
8cfd94d040b14bd8aecc086c8f5f5e0d	blobstore

f488cfb478f54edea497bf6347c2eb80 postgres

9d5be7be873c41b9898032c08aa87597 controller

```
## 集群管理

### 用法：
```

flynn cluster

flynn cluster add [-f] [-d] [-g <githost>] [--git-url <giturl>] [--no-git] [-p <tlspin>] <cluster-name> <domain> <key>

flynn cluster remove <cluster-name>

flynn cluster default [<cluster-name>]

flynn cluster migrate-domain <domain>

flynn cluster backup [--file <file>]

```
### 命令参数

如果不带任何命令参数，将显示配置的集群列表。

- add 增加一个群集到 ~/.flynnrc 配置文件中
```

选项:

```
-f, --force 强制添加集群
-d, --default 设置为默认集群
-g, --git-host=<githost> git 主机 (只支持传统的 SSH)
--git-url=<giturl> git URL
--no-git 跳过 git 配置
-p, --tls-pin=<tlspin> 集群 TLS 证书的 SHA256 哈希值
```

```
- remove 从 ~/.flynnrc 配置文件中删除集群
- default 不带参数，则显示默认的集群；后面带集群名，则将它设置为默认集群
- migrate-domain 将集群当前的基本域迁移到指定的域。迁移过程中，将为控制器/控制板生成一个新证书，同时会为每个应用添加 `<app-name>.<domain>` 模式的路由。
- backup 备份集群
```

当通过 'flynn-host bootstrap --from-backup' 命令创建一个新集群时，该备份可以被恢复。

选项:

```
--file=<backup-file> 指定备份文件的名称(默认为输出到标准输出)
```

例子

$ flynn cluster add -p KGCENkp53YF5OvOKkZIry71+czFRkSw2ZdMszZ/0ljs= default dev.localflynn.com e09dc5301d72be755a3d666f617c4600

Cluster "default" added.

$ flynn cluster migrate-domain new.example.com

Migrate cluster domain from "example.com" to "new.example.com"? (yes/no): yes

Migrating cluster domain (this can take up to 2m0s)...

Changed cluster domain from "example.com" to "new.example.com"

创建

用法：

flynn create [-r <remote>] [-y] [<name>]

`

在 flynn 中创建应用程序。如果没有提供 [<name>] 参数，将生成一个随机名。

如果使用 Git 仓库来创建一个应用，那么名为 ‘flynn’ 的远程仓库连接将被创建或替换，同样允许 git 用这个远程仓库来部署应用。

选项

选项	描述
-r --remote=<remote>	要创建的 Git 远程仓库名称，空字符串代表使用默认值。 [默认：flynn]
-y --yes	如果该仓库已经存在，跳过确认提示

例子

$ flynn create
Created turkeys-stupefy-perry

删除

用法：

```
flynn delete [-y] [-r <remote>]
```

删除一个应用。

假如该应用使用了名为 'flynn' 的远程 git 仓库连接，该连接同样也会被删除。

选项

标志	描述
-r --remote=<remote>	要删除的 Git 远程仓库名称，没有空字符串。 [默认：flynn]
-y --yes	跳过确认提示。

例子

```
$ flynn -a turkeys-stupefy-perry delete
Are you sure you want to delete the app "turkeys-stupefy-perry"? (yes/no): yes
Deleted turkeys-stupefy-perry
```

部署

用法：

```
flynn deployment
```

管理应用的部署

命令参数

不带任何参数，显示已部署列表。

例子

```
$ flynn deployment
ID                               STATUS    CREATED             FINISHED
37a63fb05fe946f18f11f741aed74d60  running   4 seconds ago
51cbf2bba1204e94b1d847ae0122c647  complete  16 seconds ago      14 seconds ago
12875d153f5c4c6cb64e263c4b422e8c  failed    About a minute ago  About a minute
ago
```

```
21d4a8174a4240a0b1dcb6303f40cad5  complete  4 minutes ago       4 minutes ago
```

环境

用法

```
flynn env [-t <proc>]
flynn env set [-t <proc>] <var>=<val>...
flynn env unset [-t <proc>] <var>...
flynn env get [-t <proc>] <var>
```

管理应用程序的环境变量。

选项

标志	描述
-t --process-type=<proc>	设置或读取指定进程类型的环境变量

命令参数

不带任何参数，显示所有环境变量列表。

- set　设置一个或多个环境变量
- unset　删除一个或多个环境变量
- get　返回环境变量的值

例子

```
$ flynn env set FOO=bar BAZ=foobar
Created release 5058ae7964f74c399a240bdd6e7d1bcb.
```

```
$ flynn env
BAZ=foobar
FOO=bar
```

```
$ flynn env get -t web FOO
bar
```

```
$ flynn env unset FOO
Created release b1bbd9bc76d6436ea2fd245300bce72e.
```

导出

用法

```
flynn export [options]
```

导出应用配置和数据。

应用的元数据（metadata）、部署策略（deploy strategy）、版本配置（release configuration）、slug、结构（formation）、Postgres 数据库将被导出到一个 tar 压缩文件。

选项

标志	描述
-f --file=<file>	存储导出数据的文件名(默认为标准输出)
-q --quiet	不显示进度

git 授权（git-credentials）

用法：

```
flynn git-credentials <operation>
```

导入

用法：

```
flynn import [options]
```

使用已导出的配置和数据来创建一个新应用。

将使用所提供导出文件中的应用的元数据（metadata）、部署策略（deploy strategy）、版本配置（release configuration）、slug、结构（formation）、Postgres 数据库，来构建新的应用。

选项

标志	描述
-f --file=<file>	数据来源文件名(默认为标准输入)

-n --name=<name>	待创建的应用名称(默认为导出文件的原应用名)
-q --quiet	不显示进度
-r --routes	导入路由

信息

显示应用的信息

```
flynn info
```

例子

```
$ flynn info
=== example
Git URL:  https://git.dev.localflynn.com/example.git
Web URL:  http://example.dev.localflynn.com
```

```
$ flynn -a example info
=== example
Git URL:  https://git.dev.localflynn.com/example.git
Web URL:  http://example.dev.localflynn.com
```

安装

启动安装器的 web 界面服务。

```
flynn install
```

例子

```
$ flynn install
```

停止任务（Kill）

```
flynn kill <job>
```

杀掉一个任务进程

资源限制

用法：

```
flynn limit [-t <proc>]flynn limit set <proc> <var>=<val>...
```

管理应用程序的资源配置。

选项

标志	描述
-t --process-type=<proc>	设置或读取指定进程类型的限制

命令

不带任何参数，即显示资源限制列表。

- set　设置一个或多个资源限制

例子

```
$ flynn limit
web:     max_fd=10000  memory=1GB
worker:  max_fd=10000  memory=1GB
```

```
$ flynn limit set web memory=512MB max_fd=12000
Created release 5058ae7964f74c399a240bdd6e7d1bcb
```

```
$ flynn limit
web:     max_fd=12000  memory=512MB
worker:  max_fd=10000  memory=1GB
```

```
$ flynn limit set web memory=256MB
Created release b39fe25d0ea344b6b2af5cf4d6542a80
```

```
$ flynn limit
web:     max_fd=12000  memory=256MB
worker:  max_fd=10000  memory=1GB
```

日志

应用程序的日志流。

```
flynn log [-f] [-j <id>] [-n <lines>] [-r] [-s] [-t <type>]
```

选项

标志	描述
-f --follow	新的日志流
-j --job=<id>	过滤出特定任务的 ID 的日志
-n --number=<lines>	从日志缓冲区中提取至多 N 行日志
-r --raw-output	输出没有前缀的原始日志信息
-s --split-stderr	将标准错误流发送到 stderr
-t --process-type=<type>	过滤出特定进程类型的日志

元数据

管理应用元数据

```
flynn meta
flynn meta set <var>=<val>...
flynn meta unset <var>...
```

例子

```
$ flynn meta
KEY  VALUE
foo  bar
```

```
$ flynn meta set foo=baz bar=qux
```

```
$ flynn meta
KEY  VALUE
foo  baz
bar  qux
```

```
$ flynn meta unset foo
```

```
$ flynn meta
```

```
KEY  VALUE
bar  qux
```

pg

```
flynn pg psql [--] [<argument>...]
flynn pg dump [-q] [-f <file>]
flynn pg restore [-q] [-f <file>]
```

选项

标志	描述
-f --file=<file>dump	备份文件的名称
-q --quiet	不显示进度

命令

• psql　打开一个 flynn的 Postgres 数据库控制台。可以使用任何有效的 PSQL 命令参数。

• dump　备份 Postgres 数据库。如果没有指定文件，备份将输出到标准输出。

• restore　从数据库备份中还原 Postgres 数据库。如果没有指定文件，将通过标准输入获取还原备份。

例子

```
$ flynn pg psql
$ flynn pg psql -- -c "CREATE EXTENSION hstore"
$ flynn pg dump -f db.dump
$ flynn pg restore -f db.dump
```

提供程序

管理与控制器相关的资源提供程序。

```
flynn provider
flynn provider add <name> <url>
```

命令

不带任何参数，即显示当前提供程序

• add 添加新提供程序

ps

列出所有 flynn 进程。

```
flynn ps
```

例子

```
$ flynn ps
ID                                TYPE    RELEASE
flynn-bb97c7dac2fa455dad73459056fabac2  web
b69d7fb5308a4684a09b160b82d267ec
flynn-c59e02b3e6ad49809424848809d4749a  web
b69d7fb5308a4684a09b160b82d267ec
flynn-46f0d715a9684e4c822e248e84a5a418  web
b69d7fb5308a4684a09b160b82d267ec
```

发布

管理应用程序的发布。

```
flynn release
flynn release add [-t <type>] [-f <file>] <uri>
flynn release show [<id>]
```

选项

标志	描述
-q --quiet	只打印发布ID
-t <type>	发布类型。目前只支持 “docker”。[默认为：docker]
-f --file=<file>	配置文件
--json	打印json格式的配置信息

命令

如果不带任何参数，将列出应用相关的所有版本。

• add 添加一个新发布

从 Docker 镜像中创建一个新的版本。

还可添加文件参数，它是 JSON 格式的配置文件的文件路径。这主要用于指定版本环境和流程（类似于Procfile）。可以使用任何控制版本类型可使用的参数。

• show显示版本信息

如果省略ID，则显示当前发布信息。

例子

比如以 flynn/slugbuilder 镜像为基础发布一个 echo 服务器，用来运行 socat。

```
$ cat config.json
{
   "env": {"MY_VAR": "Hello World, this will be available in all process types."},
   "processes": {
      "echo": {
         "cmd": ["socat -v tcp-l:$PORT,fork exec:/bin/cat"],
         "entrypoint": ["sh", "-c"],
         "env": {"ECHO": "This var is specific to the echo process type."},
         "ports": [{"proto": "tcp"}]
      }
   }
}

$ flynn release add -f config.json
https://registry.hub.docker.com?name=flynn/slugbuilder&id=15d72b7f573b
Created release 427537e78be4417fae2e24d11bc993eb.
```

```
$ flynn release
ID                              Created
427537e78be4417fae2e24d11bc993eb  11 seconds ago
```

```
$ flynn release show
ID:           427537e78be4417fae2e24d11bc993eb
Artifact:
docker+https://registry.hub.docker.com?name=flynn/slugbuilder&id=15d72b7f573b
Process Types:  echo
Created At:     2015-05-06 21:58:12.751741 +0000 UTC
ENV[MY_VAR]:    Hello World, this will be available in all process types.
```

仓库

创建一个可以通过 git 来部署应用的仓库。如果没有提供仓库名称，默认使用'flynn'

```
flynn remote add [<remote>] [-y]
```

选项

标志	描述
-y --yes	假如该仓库存在，则跳过确认提示

例子

```
$ flynn -a turkeys-stupefy-perry remote add
Created remote flynn with url https://git.dev.localflynn.com/turkeys-stupefy-perry.git
```

```
$ flynn -a turkeys-stupefy-perry remote add staging
Created remote staging with url https://git.dev.localflynn.com/turkeys-stupefy-perry.git
```

资源

管理应用程序资源。

```
flynn resource
flynn resource add <provider>
flynn resource remove <provider> <resource>
```

命令

不带任何参数，即显示资源列表。

- add 通过 <provide> 为使用应用程序添加一个新资源。
- remove 通过 <provider> 删除已存在的资源

路由

管理应用程序的路由。

```
flynn route
flynn route add http [-s <service>] [-c <tls-cert> -k <tls-key>] [--sticky] <domain>
flynn route add tcp [-s <service>] [-p <port>]
flynn route update <id> [-s <service>] [-c <tls-cert> -k <tls-key>] [--sticky] [--no-sticky]
```

```
flynn route remove <id>
```

选项

标志	描述
-s --service=<service>	路由域的服务名称（默认为APPNAME-web）
-c --tls-cert=<tls-cert>	TLS 的 PEM 编码证书路径， - 为标准输入（只支持HTTP）
-k --tls-key=<tls-key	TLS 的 PEM 编码私钥路径， - 为标准输入（只支持HTTP）
--sticky	支持基于 Cookie 的粘性路由（只支持HTTP）
--no-sticky	不支持基于 Cookie 的粘性路由（只支持更新HTTP）
-p --port=<port>	通信端口（只支持TCP）

命令

不带任何参数，即显示路由的列表。

- add　给应用程序增加路由
- remove　删除路由

例子

```
$ flynn route add http example.com
$ flynn route add tcp
```

运行

运行任务。

```
flynn run [-d] [-r <release>] [-e <entrypoint>] [--] <command> [<argument>...]
```

选项

选项	描述
-d --detached	不连接 IO 流运行任务
-r <release>	以版本 ID 来运行（默认为当前应用程序版本）
-e <entrypoint>	覆盖版本镜像的默认入口点
-l --enable-log	把输出传入日志流

scale

scale 改变一个版本中每种进程的任务数量。省略参数即显示当前 scale。

```
flynn scale [options] [<type>=<qty>...]
```

选项

标志	描述
-n --no-wait	不等待扩展事件发生
-r --release=<release>	待扩展的版本ID（默认为当前应用程序的版本）
-a --all	显示所有版本的结构

例子

```
$ flynn scale
web=4 worker=2
```

```
$ flynn scale --all
496d6e74-9db9-4cff-bcce-a3b44015907a (current)
web=1 worker=2
```

632cd907-85ab-4e53-90d0-84635650ec9a web=2

```
$ flynn scale web=2 worker=5
scaling web: 4=>2, worker: 2=>5
```

02:28:34.333 ==> web flynn-3f656af6f1e44092aa7037046236b203 down

02:28:34.466 ==> web flynn-ee83def0b8e4455793a43c8c70f5b34e down

02:28:35.479 ==> worker flynn-84f70ca18c9641ef83a178a19db867a3 up

02:28:36.508 ==> worker flynn-a3de8c326cc542aa89235e53ba304260 up

02:28:37.601 ==> worker flynn-e24760c511af4733b01ed5b98aa54647 up

scale completed in 3.944629056s

版本

显示flynn版本信息。

```
flynn version
```

控制器 API 接口

控制器

所有请求都通过 basic auth 进行验证，其中用户名为空，密码为控制器密钥。可以为事件流（在浏览器中允许通过JS来使用）选择把控制器密码作为密钥 URL 参数，你可以通过运行以下命令获取控制器密钥:

```
flynn -a controller env get AUTH_KEY
```

API 根路径是控制器的域，默认为：

```
https://controller.$CLUSTER_DOMAIN
```

应用

应用是进程的组成及其依赖和元数据的命名空间。

https://flynn.io/schema/controller/app#(https://flynn.io/schema/controller/app#)

属性	类型	描述
id	uuid string	唯一标示
name	string matching ^[a-z\d]+(-[a-z\d]+)*$	应用名称
protected	boolean	假如为true，当删除或归零时应用会被保护
meta	object	客户端指定的元数据
strategy	string	

release	uuid string	唯一标示
deploy_timeout	integer	部署超时时间（默认30秒）
created_at	date-time string	对象创建的时间
updated_at	date-time string	对象最新更新时间

创建App

实例：

————————————————————————请求
————————————————————————

```
POST /apps HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "name": "my-app-1422557606845347930",
  "protected": false
}
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "7406a4d71a0c43d3ac4b39f006cb0342",
  "name": "my-app-1422557606845347930",
  "protected": false,
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:26.845896Z",
  "updated_at": "2015-01-29T18:53:26.845896Z"
}
```

获取App

实例：

————————————————————————请求
————————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————————响应
————————————————————

```
Content-Type: application/json
```


```
{
  "id": "7406a4d71a0c43d3ac4b39f006cb0342",
  "name": "my-app-1422557606845347930",
  "protected": false,
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:26.845896Z",
  "updated_at": "2015-01-29T18:53:26.845896Z"
}
```


获取App列表

实例：

————————————————————————————请求
————————————————————

```
GET /apps HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————————响应
————————————————————

```
Content-Type: application/json
```


```
{
[
  {
    "id": "7406a4d71a0c43d3ac4b39f006cb0342",
    "name": "my-app-1422557606845347930",
    "protected": false,
    "strategy": "all-at-once",
    "created_at": "2015-01-29T18:53:26.845896Z",
    "updated_at": "2015-01-29T18:53:26.845896Z"
  },
```

```
{
  "id": "79b56ad29bac4cefb83c37965ae4ce1c",
  "name": "dashboard",
  "protected": true,
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:24.997991Z",
  "updated_at": "2015-01-29T18:53:25.015618Z"
},
{
  "id": "b6f62b8fda484f6e825484a668e161b7",
  "name": "taffy",
  "protected": true,
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:24.981366Z",
  "updated_at": "2015-01-29T18:53:24.996193Z"
},
{
  "id": "48a30977ae9c4b64ba33b773f1808915",
  "name": "gitreceive",
  "protected": true,
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:22.46581Z",
  "updated_at": "2015-01-29T18:53:22.480289Z"
},
{
  "id": "b1061d0a149a4a7498a70367c42d4013",
  "name": "router",
  "protected": true,
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:21.72534Z",
  "updated_at": "2015-01-29T18:53:21.7391Z"
},
{
  "id": "3a6f75c62c68489c82c91922d45429ad",
  "name": "blobstore",
  "protected": true,
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:21.415441Z",
  "updated_at": "2015-01-29T18:53:21.722021Z"
},
{
  "id": "eb11600643d24b5f8e18c0a42afe7f91",
```

```
    "name": "postgres",
    "protected": true,
    "strategy": "all-at-once",
    "created_at": "2015-01-29T18:53:20.053148Z",
    "updated_at": "2015-01-29T18:53:20.06459Z"
  },
  {
    "id": "a78f01dc04fb4fe3bccd85c05ae3fd73",
    "name": "controller",
    "protected": false,
    "strategy": "all-at-once",
    "created_at": "2015-01-29T18:53:20.013969Z",
    "updated_at": "2015-01-29T18:53:20.049305Z"
  }
]}
```

更新App

实例：

————————————————————————————请求
————————————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "id": "7406a4d71a0c43d3ac4b39f006cb0342",
  "protected": false,
  "meta": {
    "bread": "with hemp"
  }
}
```

————————————————————————————响应
————————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "7406a4d71a0c43d3ac4b39f006cb0342",
  "name": "my-app-1422557606845347930",
```

```
  "protected": false,
  "meta": {
    "bread": "with hemp"
  },
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:26.845896Z",
  "updated_at": "2015-01-29T18:53:26.845896Z"
}
```

删除App

实例：

————————————————————————————请求
————————————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————————响应
————————————————————————————

```
Content-Type: text/plain; charset=utf-8
```

获取应用日志

属性	类型	描述
follow	boolean	当新日志产生时将它们发送出去
job_id	uuid string	唯一标识
lines	integer	从缓冲区要返回的行数
process_type	string	只返回给定进程类型的log

以 JSON 格式来返回每个日志的每行条目

实例：

————————————————————————————请求
————————————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————————响应
————————————————————

```
Content-Type: text/plain
```


```
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-
ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request
completed\" component=controller req_id=344d4b61-d437-437c-a3c4-e4aa015647cd
status=200
duration=6.151899ms","process_type":"web","source":"app","stream":"stdout","timesta
mp":"2015-12-16T02:20:56.663589985Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-
ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\"
component=controller req_id=312895c5-4b45-4e65-b906-a46ee9bf3599
method=GET path=/apps/gitreceive/release
client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timest
amp":"2015-12-16T02:20:56.664482277Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-
ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request
completed\" component=controller req_id=312895c5-4b45-4e65-b906-a46ee9bf3599
status=200
duration=1.274184ms","process_type":"web","source":"app","stream":"stdout","timesta
mp":"2015-12-16T02:20:56.66574836Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-
ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\"
component=controller req_id=cea0a133-93c9-4ff6-b532-cd719c888eb6 method=GET
path=/apps/adcccdb4-b1a4-4209-a03a-762f4e021632
client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timest
amp":"2015-12-16T02:20:56.667514227Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-
ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request
completed\" component=controller req_id=cea0a133-93c9-4ff6-b532-cd719c888eb6
status=200
duration=1.132266ms","process_type":"web","source":"app","stream":"stdout","timesta
mp":"2015-12-16T02:20:56.66782991Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-
ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\"
req_id=7cce8772-bdad-485e-ac35-320465d70b6e component=controller
```

method=GET path=/apps
client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.668616602Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request completed\" req_id=7cce8772-bdad-485e-ac35-320465d70b6e component=controller status=200
duration=750.095µs","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.669398265Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\" component=controller req_id=99c6ab06-b4be-4996-8321-9f41847ac50c method=GET path=/apps/controller
client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.670213173Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request completed\" component=controller req_id=99c6ab06-b4be-4996-8321-9f41847ac50c status=200
duration=492.8µs","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.670729161Z"}
{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\" component=controller req_id=0738164d-2c09-41ae-a391-b550c0710fae method=GET path=/apps/f7064b9f-c968-4f16-be0e-f2efd1b2c7b7/log
client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.671577593Z"}

应用的日志流

属性	类型	描述
follow	boolean	当新日志产生时将它们发送出去
job_id	uuid string	唯一标识
lines	integer	从缓冲区要返回的行数
process_type	string	只返回给定进程类型的 log

实例：

————————————————————————请求
————————————————————

```
GET /apps/f7064b9f-c968-4f16-be0e-f2efd1b2c7b7/log?lines=10 HTTP/1.1
Accept: text/event-stream
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————

```
Content-Type: text/plain
```

```
data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request completed\" component=controller req_id=312895c5-4b45-4e65-b906-a46ee9bf3599 status=200 duration=1.274184ms","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.66574836Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\" component=controller req_id=cea0a133-93c9-4ff6-b532-cd719c888eb6 method=GET path=/apps/adcccdb4-b1a4-4209-a03a-762f4e021632 client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.667514227Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request completed\" component=controller req_id=cea0a133-93c9-4ff6-b532-cd719c888eb6 status=200 duration=1.132266ms","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.66782991Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\" req_id=7cce8772-bdad-485e-ac35-320465d70b6e component=controller method=GET path=/apps client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.668616602Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-
```

91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request completed\" req_id=7cce8772-bdad-485e-ac35-320465d70b6e component=controller status=200 duration=750.095µs","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.669398265Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\" component=controller req_id=99c6ab06-b4be-4996-8321-9f41847ac50c method=GET path=/apps/controller client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.670213173Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request completed\" component=controller req_id=99c6ab06-b4be-4996-8321-9f41847ac50c status=200 duration=492.8µs","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.670729161Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\" component=controller req_id=0738164d-2c09-41ae-a391-b550c0710fae method=GET path=/apps/f7064b9f-c968-4f16-be0e-f2efd1b2c7b7/log client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.671577593Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request completed\" component=controller req_id=0738164d-2c09-41ae-a391-b550c0710fae status=200 duration=7.132192ms","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.678703917Z"}}

data: {"event":"message","data":{"host_id":"host","job_id":"host-9869a97a-ef83-4c39-91a2-ca3110f91079","msg":"t=2015-12-16T02:20:56+0000 lvl=info msg=\"request started\" req_id=2809d7e8-dcc6-4457-971e-307ebb5919e5 component=controller method=GET path=/apps/f7064b9f-c968-4f16-be0e-f2efd1b2c7b7/log client_ip=100.100.45.12","process_type":"web","source":"app","stream":"stdout","timestamp":"2015-12-16T02:20:56.679319917Z"}}

data: {"event":"eof"}

中间件

中间件是进程所使用的镜像的固定引用。

https://flynn.io/schema/controller/artifact#(https://flynn.io/schema/controller/artifact#)

属性	类型	描述
id	uuid string	唯一标示
type	string	中间件类型
uri	uri string	用于获取中间件的uri
created_at	date-time string	对象的创建时间

创建中间件

实例：

————————————————————————请求
————————————————————

```
POST /artifacts HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "type": "docker",
  "uri":
"https://registry.hub.docker.com/flynn/slugrunner?id=039222f9884412eb0667988933caeb0ed06da28f71e3dff540a61d13d0d85ae0"
}
```

————————————————————————响应
————————————————————

```
Content-Type: application/json
```

```
{
  "id": "4d365e6cdb1d47b8b98c9cd9a7b047fd",
  "type": "docker",
  "uri":
"https://registry.hub.docker.com/flynn/slugrunner?id=039222f9884412eb0667988933caeb0ed06da28f71e3dff540a61d13d0d85ae0",
  "created_at": "2015-01-29T18:53:26.889875Z"
}
```

获取中间件列表

实例：

——————————————————————————请求
————————————————————————

```
POST /artifacts HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

——————————————————————————响应
————————————————————————

```
Content-Type: application/json
```


```
[
 {
  "id": "4d365e6cdb1d47b8b98c9cd9a7b047fd",
  "type": "docker",
  "uri":
"https://registry.hub.docker.com/flynn/slugrunner?id=039222f9884412eb0667988933c
aeb0ed06da28f71e3dff540a61d13d0d85ae0",
  "created_at": "2015-01-29T18:53:26.889875Z"
 },
 {
  "id": "8b6548e25bc8477e905deee8a33eab2a",
  "type": "docker",
  "uri":
"https://registry.hub.docker.com/flynn/dashboard?id=04b11c64060562a786281b7348
1049c7867658fc979d62c102e9a095fd73aadd",
  "created_at": "2015-01-29T18:53:25.000437Z"
 },
 {
  "id": "03f9d97aeab94f2b9bb23f4ade0eb270",
  "type": "docker",
  "uri":
"https://registry.hub.docker.com/flynn/taffy?id=f9fec4547fcec06160a32da164cad7954
1b1e2edd96c4a55dee7fc64814a8b26",
  "created_at": "2015-01-29T18:53:24.983953Z"
 },
 {
  "id": "74010365c551463386bae39cf4ad0a47",
  "type": "docker",
  "uri":
```

```
"https://registry.hub.docker.com/flynn/receiver?id=c2cfc8c7715c8300f3d0d1c318eb7a2fbeba6fd678a5a7d7ab8bd0f1659fad2f",
    "created_at": "2015-01-29T18:53:22.469009Z"
  },
  {
    "id": "73d37c1bdacd432f9e5c3cbe121c365e",
    "type": "docker",
    "uri":
"https://registry.hub.docker.com/flynn/router?id=9ff4d42de0758dfe3278dbc0877e08b596c5f726cbb06819ff99fe4fdb448aa5",
    "created_at": "2015-01-29T18:53:21.728047Z"
  },
  {
    "id": "afe339ec9a0f46b28cfa5dff383e5179",
    "type": "docker",
    "uri":
"https://registry.hub.docker.com/flynn/blobstore?id=75231ad60a567fa118cc003462612abc37a4646e5688ced2e7308f0951392f87",
    "created_at": "2015-01-29T18:53:21.711605Z"
  },
  {
    "id": "61b804fd0f9c42359ac8d8e3131d16bf",
    "type": "docker",
    "uri":
"https://registry.hub.docker.com/flynn/postgresql?id=5cb8db666db41d1f86091c06d709f282295ded77e854241bef6ebee25fda09d0",
    "created_at": "2015-01-29T18:53:20.055663Z"
  },
  {
    "id": "1b7c8dcfd0b14755a452367e5f6723b6",
    "type": "docker",
    "uri":
"https://registry.hub.docker.com/flynn/controller?id=e65f0f3d5e5853f2fc1f60758fb134d8ba7b746da0d8f4fdda473ff5a57fee6c",
    "created_at": "2015-01-29T18:53:20.024582Z"
  }
]
```


Formation

Formation 是一个正在运行的版本。

https://flynn.io/schema/controller/formation#(https://flynn.io/schema/controller/formatio

n#)

属性	类型	描述
app	uuid string	唯一标示
release	uuid string	唯一标示
processes	object	每种进程所运行的进程数
created_at	date-time string	对象的创建时间
updated_at	date-time string	对象最新更新时间

更新 Formation

实例：

————————————————————————请求
————————————————————————

```
PUT
/apps/7406a4d71a0c43d3ac4b39f006cb0342/formations/689ce5b9ad1541ab975d51cba1e051d0 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "app": "7406a4d71a0c43d3ac4b39f006cb0342",
  "release": "689ce5b9ad1541ab975d51cba1e051d0",
  "processes": {
    "foo": 1
  }
}
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "app": "7406a4d71a0c43d3ac4b39f006cb0342",
  "release": "689ce5b9ad1541ab975d51cba1e051d0",
  "processes": {
    "foo": 1
```

```
  },
  "created_at": "2015-01-29T18:53:26.907195Z",
  "updated_at": "2015-01-29T18:53:26.907195Z"
}
```

获取 Formation

实例：

——————————————————————————请求
————————————————————————

```
GET
/apps/7406a4d71a0c43d3ac4b39f006cb0342/formations/689ce5b9ad1541ab975d51c
ba1e051d0 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

——————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "app": "7406a4d71a0c43d3ac4b39f006cb0342",
  "release": "689ce5b9ad1541ab975d51cba1e051d0",
  "processes": {
    "foo": 1
  },
  "created_at": "2015-01-29T18:53:26.907195Z",
  "updated_at": "2015-01-29T18:53:26.907195Z"
}
```

获取 Formation 列表

实例：

——————————————————————————请求
————————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342/formations HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

——————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
[
  {
    "app": "7406a4d71a0c43d3ac4b39f006cb0342",
    "release": "689ce5b9ad1541ab975d51cba1e051d0",
    "processes": {
      "foo": 1
    },
    "created_at": "2015-01-29T18:53:26.907195Z",
    "updated_at": "2015-01-29T18:53:26.907195Z"
  }
]
```

删除 Formation

实例：

————————————————————————请求
————————————————————————

```
DELETE
/apps/7406a4d71a0c43d3ac4b39f006cb0342/formations/40202cf3b0e946a4b2c4db42a0c14194 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: text/plain; charset=utf-8
```

扩展的 Expanded

Formation 是一个正在运行的版本。

https://flynn.io/schema/controller/expanded_formation#(https://flynn.io/schema/controller/expanded_formation)

属性	类型	描述
app	object	应用是进程的组成及其依赖和元数据的命名空间
app.id	uuid string	唯一标示

app.name	string matching ^[a-z\d]+(-[a-z\d]+)*$	应用名称
app.protected	boolean	若为真，则应用可以受保护不被删除
app.meta	object	用户自定义元数据
app.strategy	string	all-at-once one-by-one postgres discoverd-meta
app.release	uuid string	唯一标示
app.deploy_timeout	integer	部署超时时间（默认30秒）
app.created_at	date-time string	对象创建时间
app.updated_at	date-time string	对象更新时间
release	object	
release.id	uuid string	唯一标示
release.artifact	uuid string	唯一标示
release.meta	object	用户自定义元数据
release.env	object	环境变量
release.processes	object	
release.created_at	date-time string	对象创建时间
artifact	object	中间件是进程所使用的镜像的固定引用
artifact.id	uuid string	唯一标示
artifact.type	string	docker
artifact.uri	uri string	唯一标示
artifact.created_at	date-time string	对象的创建时间
processes	object	每种进程所运行的进程数

updated_at	date-time string	对象最新更新时间

获取扩展的 formation

实例：

——————————————————————————请求
————————————————————————

```
GET /apps/adcccdb4-b1a4-4209-a03a-762f4e021632/formations/47154f8c-a604-469d-ae6a-e431990ddee8?expand=true HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

——————————————————————————响应
————————————————————————

```
Content-Type: application/json
```


```
{
  "app": {
    "id": "adcccdb4-b1a4-4209-a03a-762f4e021632",
    "name": "my-app-1450232456657062340",
    "meta": null
  },
  "release": {
    "id": "47154f8c-a604-469d-ae6a-e431990ddee8",
    "artifact": "c1889f55-c244-43ce-af70-ead357daa6ec",
    "env": {
      "some": "info"
    },
    "processes": {
      "foo": {
        "cmd": [
          "ls",
          "-l"
        ],
        "env": {
          "BAR": "baz"
        },
        "resources": {
          "max_fd": {
            "request": 10000,
            "limit": 10000
```

```
        },
        "memory": {
          "request": 1073741824,
          "limit": 1073741824
        }
       }
     }
    }
  },
  "artifact": {
    "id": "c1889f55-c244-43ce-af70-ead357daa6ec",
    "type": "docker",
    "uri":
"https://dl.flynn.io/tuf?name=flynn/slugrunner&id=dc1dadc66c31a1957bad101a22c4
b607b2dd4f8f42ae5593ec28f3eef45e7b97"
  },
  "processes": {
    "foo": 1
  },
  "updated_at": "2015-12-16T02:21:06.748757Z"
}
```


获取运行的 formation 列表

实例：

————————————————————————请求
————————————————————————

```
GET /formations?active=true HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```


```
[
  {
    "app": {
      "id": "adcccdb4-b1a4-4209-a03a-762f4e021632",
      "name": "my-app-1450232456657062340",
      "meta": null
```

```
    },
    "release": {
      "id": "47154f8c-a604-469d-ae6a-e431990ddee8",
      "artifact": "c1889f55-c244-43ce-af70-ead357daa6ec",
      "env": {
        "some": "info"
      },
      "processes": {
        "foo": {
          "cmd": [
            "ls",
            "-l"
          ],
          "env": {
            "BAR": "baz"
          },
          "resources": {
            "max_fd": {
              "request": 10000,
              "limit": 10000
            },
            "memory": {
              "request": 1073741824,
              "limit": 1073741824
            }
          }
        }
      }
    },
    "artifact": {
      "id": "c1889f55-c244-43ce-af70-ead357daa6ec",
      "type": "docker",
      "uri":
"https://dl.flynn.io/tuf?name=flynn/slugrunner&id=dc1dadc66c31a1957bad101a22c4
b607b2dd4f8f42ae5593ec28f3eef45e7b97"
    },
    "processes": {
      "foo": 1
    },
    "updated_at": "2015-12-16T02:21:06.748757Z"
  },
  {
```

```
  "app": {
    "id": "28a8aed1-a50d-446e-b4ad-38ca3e95fb8f",
    "name": "status",
    "meta": null
  },
  "release": {
    "id": "fff9dd4b-9f38-4f71-b3c5-5cd5fc19d31a",
    "artifact": "e88714df-27b5-40aa-900e-717384fddcdd",
    "env": {
      "AUTH_KEY": "44335767c99fb93b23c9a1500bf1ebeb"
    },
    "processes": {
      "web": {
        "ports": [
          {
            "port": 80,
            "proto": "tcp",
            "service": {
              "name": "status-web",
              "create": true
            }
          }
        ],
        "resources": {
          "max_fd": {
            "request": 10000,
            "limit": 10000
          },
          "memory": {
            "request": 1073741824,
            "limit": 1073741824
          }
        }
      }
    }
  },
  "artifact": {
    "id": "e88714df-27b5-40aa-900e-717384fddcdd",
    "type": "docker",
    "uri":
"https://dl.flynn.io/tuf?name=flynn/status&id=33c003b37b2919a841d6e14a6335ff440
607957efeb2e9022dce05a36f86203f"
```

```
    },
    "processes": {
      "web": 1
    },
    "updated_at": "2015-12-16T02:20:50.513318Z"
  },
  {
    "app": {
      "id": "52e4cac9-3e4a-42ab-8b87-b714403aafc6",
      "name": "dashboard",
      "meta": null
    },
    "release": {
      "id": "93f1f03e-30c4-4c1f-8c4b-7c0c7e295421",
      "artifact": "ec791440-fa5b-4d8e-903a-94999f634ea1",
      "env": {
        "APP_NAME": "dashboard",
        "CA_CERT": "-----BEGIN CERTIFICATE-----
\nMIIDBDCCAe6gAwIBAgIRAP6BP9TQKrq0l1CCcSI4gb4wCwYJKoZIhvcNAQELMC0x\n
DjAMBgNVBAoTBUZseW5uMRswGQYDVQQLExJGbHlubiBFcGhlbWVyYWwgQ0EwHhc
N\nMTUxMjE2MDIyMDQxWhcNMjAxMjE0MDIyMDQxWjAtMQ4wDAYDVQQKEwVGbH
lubjEb\nMBkGA1UECxMSRmx5bm4gRXBoZW1lcmFsIENBMIIBIjANBgkqhkiG9w0BAQEF
AAOC\nAQ8AMIIBCgKCAQEAn+KUccUSVeWoJZTlOxqaiXacGECGAPETxfmnL9ep1rA/4
DWi\n2OTpZoeSFdyL1yV5KJxPkq6XMNyl/BR72SOomCmoFRnZzDih//+gfuJAFVGaF1j9
\nFOIX+VyO8jyFzqk2vZg9R+ncRyFgjEgpHn0gDdt4WLWDxPpfFMClEZU6Mpu02ngj\n1
kmlNgzajl71qhhKtrv3W7ACKh6O3fMSBg4n/ZiVnVQjdkejHskV5R43FH4bpZzP\nhb82rI
6FNBNOCKqwbQ/AdGGVvJSzXPypvzCMkGghhx8pckoZw7DONXhSojfIO6Kt\nlxUYO1O
dmnb+nyJjxGyidSperhfm+CuSFPn2BwIDAQABoyMwITAOBgNVHQ8BAf8E\nBAMCAAY
wDwYDVR0TAQH/BAUwAwEB/zALBgkqhkiG9w0BAQsDggEBAGvEqq1ko+pE\nS5RBH
mTWW6mJom9rA+aii9n88lMhbdjikeVFkQkH6Qi8zR5O4hCdDUC4qYUp+rIF\nZ4XZ9O
gl4CcR1gEPGu1KX2EnjVKloS1LThugCuP5YJgQ5qoMsQs681s1ZpPd/0gh\ntjASQy1Y72
wY1SxVWqJzkuzf8mMDg52I4wwD9Eif2zq1/+sJa9mdilhvd0Wp+hDV\n6qN6xbH65V2V
KwDrkD2p7CIn3NmknYhmw+obezqRkSl6N5Is/QPVg59bOu0evWfS\nDyaOehm1Emm
RVehNj8IR0noFwUMbIASnpTescLd4Nz9pEjbgHQ+m6Swe8hnaZ3dd\n1TvxABm/4VY=\
n-----END CERTIFICATE-----\n",
        "CONTROLLER_DOMAIN": "controller.dev.localflynn.com",
        "CONTROLLER_KEY": "s3cr3t",
        "DEFAULT_ROUTE_DOMAIN": "dev.localflynn.com",
        "LOGIN_TOKEN": "c8414bceb41bd679535c76f87472a8c2",
        "SECURE_COOKIES": "true",
        "SESSION_SECRET": "f180d52702a70f23a5eb6b1d87d5ab3a",
        "URL": "https://dashboard.dev.localflynn.com"
```

```
    },
    "processes": {
      "web": {
        "ports": [
          {
            "port": 80,
            "proto": "tcp",
            "service": {
              "name": "dashboard-web",
              "create": true,
              "check": {
                "type": "http",
                "status.omitempty": 0
              }
            }
          }
        ],
        "resources": {
          "max_fd": {
            "request": 10000,
            "limit": 10000
          },
          "memory": {
            "request": 1073741824,
            "limit": 1073741824
          }
        }
      }
    }
  },
  "artifact": {
    "id": "ec791440-fa5b-4d8e-903a-94999f634ea1",
    "type": "docker",
    "uri":
"https://dl.flynn.io/tuf?name=flynn/dashboard&id=d602235982d4d3e4a75253aae15f5
ced25d2a6b3585518b644788ac5f0f943ec"
  },
  "processes": {
    "web": 1
  },
  "updated_at": "2015-12-16T02:20:50.47223Z"
},
```

```
{
  "app": {
    "id": "3ccdfb02-1dfb-4148-9ade-caa9ac918947",
    "name": "logaggregator",
    "meta": null
  },
  "release": {
    "id": "d755489a-89ce-4c8c-b717-73d5589bab41",
    "artifact": "08d14942-65c0-456e-b1ef-e7d46717dcd4",
    "processes": {
      "app": {
        "cmd": [
          "-logaddr",
          ":514",
          "-apiaddr",
          ":80"
        ],
        "ports": [
          {
            "port": 80,
            "proto": "tcp"
          },
          {
            "port": 514,
            "proto": "tcp"
          }
        ],
        "resources": {
          "max_fd": {
            "request": 10000,
            "limit": 10000
          },
          "memory": {
            "request": 1073741824,
            "limit": 1073741824
          }
        }
      }
    }
  },
  "artifact": {
    "id": "08d14942-65c0-456e-b1ef-e7d46717dcd4",
```

```
      "type": "docker",
      "uri":
"https://dl.flynn.io/tuf?name=flynn/logaggregator&id=cd30beb1606288f61eae7a4fe8
3ad3652689e644d9bb3e8bc10ca22d8ca0652b"
    },
    "processes": {
      "app": 1
    },
    "updated_at": "2015-12-16T02:20:50.392915Z"
  },
  {
    "app": {
      "id": "86750e3a-927f-4146-8651-4c8b7cc2de01",
      "name": "gitreceive",
      "meta": null
    },
    "release": {
      "id": "ea333a70-5971-467a-ba92-1d2e724db07f",
      "artifact": "1a5bf802-3337-48b3-b812-b994f1cde371",
      "env": {
        "CONTROLLER_KEY": "s3cr3t",
        "SLUGBUILDER_IMAGE_URI":
"https://dl.flynn.io/tuf?name=flynn/slugbuilder&id=304a939ca74764b7e356133d7559
cc245207deca8c65e782d4a043f14a566cae",
        "SLUGRUNNER_IMAGE_URI":
"https://dl.flynn.io/tuf?name=flynn/slugrunner&id=dc1dadc66c31a1957bad101a22c4
b607b2dd4f8f42ae5593ec28f3eef45e7b97"
      },
      "processes": {
        "app": {
          "ports": [
            {
              "port": 0,
              "proto": "tcp",
              "service": {
                "name": "gitreceive",
                "create": true
              }
            }
          ],
          "resources": {
            "max_fd": {
```

```
            "request": 10000,
            "limit": 10000
          },
          "memory": {
            "request": 1073741824,
            "limit": 1073741824
          }
        }
      }
    },
    "artifact": {
      "id": "1a5bf802-3337-48b3-b812-b994f1cde371",
      "type": "docker",
      "uri":
"https://dl.flynn.io/tuf?name=flynn/gitreceive&id=426e13b19a91af29a373f59d7edae7
e54fe9d035961738d342e4b2c001027cde"
    },
    "processes": {
      "app": 1
    },
    "updated_at": "2015-12-16T02:20:47.071423Z"
  },
  {
    "app": {
      "id": "aaae4e4b-dc79-4753-bc99-71e40c7d24dd",
      "name": "router",
      "meta": null
    },
    "release": {
      "id": "c8268dcf-8f16-48e5-9c40-797a399dac99",
      "artifact": "48953db6-1c0a-4af2-b687-7fe6971fcf84",
      "env": {
        "COOKIE_KEY": "7AaXi7KZDeWu4FotDN+7wi6RbHWywW6ClN2qap78cSg=",
        "DATABASE_URL":
"postgres://975487bc5a45fc80ade5765db668d873:46aabd4df6f39a1a23e63d9a76dece
78@leader.postgres.discoverd:5432/6e63ce03061c8788f766994d9a9be92a",
        "FLYNN_POSTGRES": "postgres",
        "PGDATABASE": "6e63ce03061c8788f766994d9a9be92a",
        "PGHOST": "leader.postgres.discoverd",
        "PGPASSWORD": "46aabd4df6f39a1a23e63d9a76dece78",
        "PGUSER": "975487bc5a45fc80ade5765db668d873",
```

"TLSCERT": "-----BEGIN CERTIFICATE-----
\nMIIDSjCCAjSgAwIBAgIQDH8zACle7B+pqJB3QfwthDALBgkqhkiG9w0BAQswLTEO\nM
AwGA1UEChMFRmx5bm4xGzAZBgNVBAsTEkZseW5uIEVwaGVtZXJhbCBDQTAeFw0x\n
NTEyMTYwMjIwNDJaFw0yMDEyMTQwMjIwNDJaMC0xDjAMBgNVBAoTBUZseW5uMRs
w\nGQYDVQQDExJkZXYubG9jYWxmbHlubi5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4I
B\nDwAwggEKAoIBAQC3NsEp+fYsbgr94cOBVnSK3ZcHBobwcCdzNvNqh6y1V9wn7x4
C\nS6Rk0pn1Zz/FS1F+uA8KMVUzwWCCSl/jW4BNAqbLbVi4I5oap8mCDQDt+MDPZyn
5\noOCn6vj5OCJXva3RmhJRocF88oXcH4lXWUyfKMOb+0/+tRC7lg5gqBxnvkiJrW7D\n
yq0+FoZViBgrN3JcxAgWFx9FK8sITzWpgPqJvJiS4Ry54o8hBTQCKyFCHTwdoufd\nKyBw
m6tNn/KLhUXcAEsBD3qvdI0AEX9rssGxVErCualgPaBAvDCFz2H+wwvKnHpK\nCcwIflE0
4YYOjCHphq9j7FIzd5/+fp0Ng53pAgMBAAGjajBoMA4GA1UdDwEB/wQE\nAwIAoDATB
gNVHSUEDDAKBggrBgEFBQcDATAMBgNVHRMBAf8EAjAAMDMGA1UdEQQs\nMCqCE
mRldi5sb2NhbGZseW5uLmNvbYIUKi5kZXYubG9jYWxmbHlubi5jb20wCwYJ\nKoZIhvcN
AQELA4IBAQAIR7k3teJZ5c2eLUlcfLSZiigz+RaZeqxHHt6fZXA9P9XS\n+m8NSIUffdfOXa
XjokYTYc91RqGhxVGUUopEA7P64QkEJRpu4qcfq6CXxDoExyf5\ncumrzrBZmTvExqlGBb
yviz7T4c7IZAe0U3zYCoaVd1Ss84mY8DA37oqM2Y1pH7Tc\nzwBprXDqAfwt70xoYFWQ
ZoK6SbS+qCQkc+0K9HW0ZzGRwqMTbAoPOTn9D2c9f9o5\nuM9KkiTuvY3T1Kimb848
IVvhBfEofeHXZspc9HWqfLJM8wIpMJCaDwcDBN+yFku6\nqqaRa9SXkZLNrB+eCqsVOU
zBsmK+sgUL6jmM+Pfc\n-----END CERTIFICATE-----\n",
"TLSKEY": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEogIBAAKCAQEAtzbBKfn2LG4K/eHDgVZ0it2XBwaG8HAnczbzaoestVfcJ+8e\nAku
kZNKZ9Wc/xUtRfrgPCjFVM8Fggkpf41uATQKmy21YuCOaGqfJgg0A7fjAz2cp\n+aDgp+
r4+TgiV72t0ZoSUaHBfPKF3B+JV1lMnyjDm/tP/rUQu5YOYKgcZ75Iia1u\nw8qtPhaGVYg
YKzdyXMQIFhcfRSvLCE81qYD6ibyYkuEcueKPIQU0AishQh08HaLn\n3SsgcJurTZ/yi4VF3
ABLAQ96r3SNABF/a7LBsVRKwrmpYD2gQLwwhc9h/sMLypx6\nSgnMCH5RNOGGDowh
6YavY+xSM3ef/n6dDYOd6QIDAQABAoIBAGEZTsRdal9frsmg\n1gl89WUHTVx21BsnhX
DlZZpG73xoZRBxBPl4d1bL2raZks2QPk+nYqknPh2g1fIX\nUrxfKR8nUZisigGFNM39sKk
RFTYJyL3i0fcvkQDNYkhKvMC9kRptylmjHbWHnvPr\nbtFxTU9ovkqzpVC7S3Pvcg0961f7
NgUqM5bfzWlgZYDATtNTpngc6/8bUgG8xhdL\nLsfd84FJcC/0k09Gxu5GPuwvcws1Iag3
9+Rkz2G9KmOEnN8dywLLedWPQ8A6fIjP\ngSSYTPWRBEF7V6/yRb7a0z1M+SJUrA0vD
YJsa2P2ry0YslJi1O9ipOeBmmCddVjN\n21SywoECgYEA45mlj46ySdKhB7iRDaD+M1Ffo
9vaCniGjZxEH+sPWXet3gr/hTGN\nmsQngU66v/Gj6clTSOlkOImQHT7Q0/GeHTDEAVEX
JNTX8NllElJe6LjOUyqc7kUN\n/zASp3YOHWVC8y3cNy8ScydUhO5hkaVJcpqO0RWxnW
nqV6m2OOtcyvkCgYEAzhNA\ntxEpP/7qMz2/Kx7LflRXtMAIfecIlLmq4mw0pnMCebCCX
Op7/Lw7VOLIz1JfzGtW\n2b6BTspzbTW16rjedA65+Pj/EdmXxtL7KIfDlnmab7cSQ3Vi/T
mbza4Z+etZ8Yy2\nRwA2Ni8IPS2y8NukcAJH+fGjIAbS/FqgR6KJtnECgYB955scpRWfnP
MTsFgdr/ev\nL862eIJP1Iiqgc110rS1C22VLYRYjCcHfIdWEtVgJS5Fv+dgqCuW0xJz+zHO
bRTI\nlRd4mwEwkMW8KKau0f5KwyDwBNy3OmAZ0OAg3z/Dpya4G6B8rn+IUDODLdO
vnD3b\nNgDXTSbqK5NFV07Egf7dEQKBgHdthvrRckOJW3dY2JqH3U0YWkABbqpwo3N
16sgQ\nB69TReL6XYK1zidZe1UlnVUxmx1gM9Q7aLsd7ykikIw4mYtPIY5d140juqdajH4f\
nuEVTsiH+ShJYTIcOxPG9bYZ+VAD+GIEraT+boD28Z1DZKgJUABylYnFkIxBnAS6i\nzwax
AoGAa3TWYsK0VUrn1EwlI9zzCLW8myjzyPLgpEQ6264ETNSw4K42M2fthcS4\nzlVMrPJ5

```
yhftxCkxbXNFp74lt5Nhryv9K3RWdnL6aXoxR4juhfXBPUBPK+bk/g6F\nohfHj/PeDJFBe0
MWC1Q4x7jbeHgqXQ9q7mun+9xH8ZziMdknHc4=\n-----END RSA PRIVATE KEY-----
\n"
      },
      "processes": {
        "app": {
          "cmd": [
            "-http-port",
            "80",
            "-https-port",
            "443",
            "-tcp-range-start",
            "3000",
            "-tcp-range-end",
            "3500"
          ],
          "omni": true,
          "host_network": true,
          "resources": {
            "max_fd": {
              "request": 10000,
              "limit": 10000
            },
            "memory": {
              "request": 1073741824,
              "limit": 1073741824
            }
          }
        }
      }
    },
    "artifact": {
      "id": "48953db6-1c0a-4af2-b687-7fe6971fcf84",
      "type": "docker",
      "uri":
"https://dl.flynn.io/tuf?name=flynn/router&id=35dd76cf940ed67fdb4f373f79fc79dda3
27a0fb36d3dffb3894cb5f981b4a21"
    },
    "processes": {
      "app": 1
    },
    "updated_at": "2015-12-16T02:20:47.037485Z"
```

```
  },
  {
    "app": {
      "id": "0b5fb9b4-af2b-462e-aaca-cc64591573e5",
      "name": "blobstore",
      "meta": null
    },
    "release": {
      "id": "53815de9-8a78-4ab1-909e-a465242f57e7",
      "artifact": "a436a3d6-a2c5-430e-961d-96518dacddff",
      "env": {
        "DATABASE_URL":
"postgres://8a4906b3587c5faabef321d34936ac3a:6ac0c933a4abba2a2f8ec0e87ccb8f6
8@leader.postgres.discoverd:5432/d2f1b01c72f997dc2dbc0e1bc6c4c6ec",
        "FLYNN_POSTGRES": "postgres",
        "PGDATABASE": "d2f1b01c72f997dc2dbc0e1bc6c4c6ec",
        "PGHOST": "leader.postgres.discoverd",
        "PGPASSWORD": "6ac0c933a4abba2a2f8ec0e87ccb8f68",
        "PGUSER": "8a4906b3587c5faabef321d34936ac3a"
      },
      "processes": {
        "web": {
          "ports": [
            {
              "port": 80,
              "proto": "tcp"
            }
          ],
          "resources": {
            "max_fd": {
              "request": 10000,
              "limit": 10000
            },
            "memory": {
              "request": 1073741824,
              "limit": 1073741824
            }
          }
        }
      }
    },
    "artifact": {
```

```
      "id": "a436a3d6-a2c5-430e-961d-96518dacddff",
      "type": "docker",
      "uri":
"https://dl.flynn.io/tuf?name=flynn/blobstore&id=8b27756a551d5dd5e29181209b842
c71288494d2fd7aded04fe7608c74bf4f9c"
    },
    "processes": {
      "web": 1
    },
    "updated_at": "2015-12-16T02:20:46.608041Z"
  },
  {
    "app": {
      "id": "f7064b9f-c968-4f16-be0e-f2efd1b2c7b7",
      "name": "controller",
      "meta": null
    },
    "release": {
      "id": "9afcffd7-6278-47c9-a2f0-1326b42b4a03",
      "artifact": "c8400666-26d9-493f-9630-3d34045f8498",
      "env": {
        "AUTH_KEY": "s3cr3t",
        "CA_CERT": "-----BEGIN CERTIFICATE-----
\nMIIDBDCCAe6gAwIBAgIRAP6BP9TQKrq0l1CCcSI4gb4wCwYJKoZIhvcNAQELMC0x\n
DjAMBgNVBAoTBUZseW5uMRswGQYDVQQLExJGbHlubiBFcGhlbWVyYWwgQ0EwHhc
N\nMTUxMjE2MDIyMDQxWhcNMjAxMjE0MDIyMDQxWjAtMQ4wDAYDVQQKEwVGbH
lubjEb\nMBkGA1UECxMSRmx5bm4gRXBoZW1lcmFsIENBMIIBIjANBgkqhkiG9w0BAQEF
AAOC\nAQ8AMIIBCgKCAQEAn+KUccUSVeWoJZTlOxqaiXacGECGAPETxfmnL9ep1rA/4
DWi\n2OTpZoeSFdyL1yV5KJxPkq6XMNyl/BR72SOomCmoFRnZzDih//+gfuJAFVGaF1j9
\nFOIX+VyO8jyFzqk2vZg9R+ncRyFgjEgpHn0gDdt4WLWDxPpfFMClEZU6Mpu02ngj\n1
kmlNgzajl71qhhKtrv3W7ACKh6O3fMSBg4n/ZiVnVQjdkejHskV5R43FH4bpZzP\nhb82rI
6FNBNOCKqwbQ/AdGGVvJSzXPypvzCMkGghhx8pckoZw7DONXhSojfIO6Kt\nlxUYO1O
dmnb+nyJjxGyidSperhfm+CuSFPn2BwIDAQABoyMwITAOBgNVHQ8BAf8E\nBAMCAAY
wDwYDVR0TAQH/BAUwAwEB/zALBgkqhkiG9w0BAQsDggEBAGvEqq1ko+pE\nS5RBH
mTWW6mJom9rA+aii9n88lMhbdjikeVFkQkH6Qi8zR5O4hCdDUC4qYUp+rIF\nZ4XZ9O
gl4CcR1gEPGu1KX2EnjVKloS1LThugCuP5YJgQ5qoMsQs681s1ZpPd/0gh\ntjASQy1Y72
wY1SxVWqJzkuzf8mMDg52I4wwD9Eif2zq1/+sJa9mdilhvd0Wp+hDV\n6qN6xbH65V2V
KwDrkD2p7CIn3NmknYhmw+obezqRkSl6N5Is/QPVg59bOu0evWfS\nDyaOehm1Emm
RVehNj8IR0noFwUMbIASnpTescLd4Nz9pEjbgHQ+m6Swe8hnaZ3dd\n1TvxABm/4VY=\
n-----END CERTIFICATE-----\n",
        "DATABASE_URL":
"postgres://df60a79e03b92c3157aaad47dab10278:623f1831ed89cd811ea1da0fb985b
```

```
d81@leader.postgres.discoverd:5432/33c5b220c9eed7307d65a2b367236d10",
      "DEFAULT_ROUTE_DOMAIN": "dev.localflynn.com",
      "FLYNN_POSTGRES": "postgres",
      "NAME_SEED": "e53b1785be43a8569707",
      "PGDATABASE": "33c5b220c9eed7307d65a2b367236d10",
      "PGHOST": "leader.postgres.discoverd",
      "PGPASSWORD": "623f1831ed89cd811ea1da0fb985bd81",
      "PGUSER": "df60a79e03b92c3157aaad47dab10278"
    },
    "processes": {
      "scheduler": {
        "cmd": [
          "scheduler"
        ],
        "ports": [
          {
            "port": 0,
            "proto": "tcp"
          }
        ],
        "omni": true,
        "service": "controller-scheduler",
        "resurrect": true,
        "resources": {
          "max_fd": {
            "request": 10000,
            "limit": 10000
          },
          "memory": {
            "request": 1073741824,
            "limit": 1073741824
          }
        }
      },
      "web": {
        "cmd": [
          "controller"
        ],
        "ports": [
          {
            "port": 80,
            "proto": "tcp"
```

```
          }
         ],
         "resurrect": true,
         "resources": {
          "max_fd": {
           "request": 10000,
           "limit": 10000
          },
          "memory": {
           "request": 1073741824,
           "limit": 1073741824
          }
         }
        },
        "worker": {
         "cmd": [
          "worker"
         ],
         "ports": [
          {
           "port": 0,
           "proto": "tcp"
          }
         ],
         "resources": {
          "max_fd": {
           "request": 10000,
           "limit": 10000
          },
          "memory": {
           "request": 1073741824,
           "limit": 1073741824
          }
         }
        }
       }
      },
      "artifact": {
       "id": "c8400666-26d9-493f-9630-3d34045f8498",
       "type": "docker",
       "uri":
"https://dl.flynn.io/tuf?name=flynn/controller&id=38242063cc75f2f6e3cfc15ce44899d
```

```
103191c0be9b11403e44ee4bf4749db72"
    },
    "processes": {
      "scheduler": 1,
      "web": 1,
      "worker": 1
    },
    "updated_at": "2015-12-16T02:20:44.205169Z"
  },
  {
    "app": {
      "id": "4940a4fe-3acc-4be7-aa97-7c3f982ba552",
      "name": "discoverd",
      "meta": null
    },
    "release": {
      "id": "dd6a3a63-a920-4954-8a0b-7f6e9e4e1b79",
      "artifact": "b2b4c646-aef0-4b3b-a501-cd282c31299e",
      "env": {
        "DISCOVERD": "none",
        "DISCOVERD_PEERS": "10.0.2.15:1111,"
      },
      "processes": {
        "app": {
          "ports": [
            {
              "port": 1111,
              "proto": "tcp"
            },
            {
              "port": 53,
              "proto": "tcp"
            }
          ],
          "data": true,
          "omni": true,
          "host_network": true,
          "service": "discoverd",
          "resurrect": true,
          "resources": {
            "max_fd": {
              "request": 10000,
```

```
            "limit": 10000
          },
          "memory": {
            "request": 1073741824,
            "limit": 1073741824
          }
        }
      }
    }
  },
  "artifact": {
    "id": "b2b4c646-aef0-4b3b-a501-cd282c31299e",
    "type": "docker",
    "uri":
"https://dl.flynn.io/tuf?name=flynn/discoverd&id=da0bd1401a6df09efb904a852992cd
3a19b657223347bad6b427350e52c254a4"
  },
  "processes": {
    "app": 1
  },
  "updated_at": "2015-12-16T02:20:44.193653Z"
 },
 {
  "app": {
    "id": "291b591e-ea67-4393-9dd8-f8b8f79ea34c",
    "name": "flannel",
    "meta": null
  },
  "release": {
    "id": "ce17ef61-be09-4f60-b12d-d8aa87eb0a82",
    "artifact": "172410eb-4e75-4597-a366-efe596b0af27",
    "env": {
      "DISCOVERD": "none"
    },
    "processes": {
      "app": {
        "ports": [
          {
            "port": 5002,
            "proto": "tcp"
          }
        ],
```

```
        "omni": true,
        "host_network": true,
        "resurrect": true,
        "resources": {
          "max_fd": {
            "request": 10000,
            "limit": 10000
          },
          "memory": {
            "request": 1073741824,
            "limit": 1073741824
          }
        }
      }
    }
  },
  "artifact": {
    "id": "172410eb-4e75-4597-a366-efe596b0af27",
    "type": "docker",
    "uri":
"https://dl.flynn.io/tuf?name=flynn/flannel&id=7d6c9acf4dc854210cc753c6a8c889899
200b35fd45f50fb35d1bb9189c92510"
  },
  "processes": {
    "app": 1
  },
  "updated_at": "2015-12-16T02:20:44.170687Z"
 },
 {
  "app": {
    "id": "2b18fac0-cb75-4243-ba0e-1a96d5e2ddda",
    "name": "postgres",
    "meta": null
  },
  "release": {
    "id": "6244ff18-44a3-4db1-a0d4-ceab0abded24",
    "artifact": "e3e69c33-dd0c-4381-b9a9-30cf0c417ce7",
    "env": {
      "FLYNN_POSTGRES": "postgres",
      "PGDATABASE": "postgres",
      "PGHOST": "leader.postgres.discoverd",
      "PGPASSWORD": "2b12bf3d5a0bfae0ed5723947e21af7b",
```

```
    "PGUSER": "flynn"
  },
  "processes": {
    "postgres": {
      "cmd": [
        "postgres"
      ],
      "env": {
        "SINGLETON": "true"
      },
      "ports": [
        {
          "port": 5432,
          "proto": "tcp"
        }
      ],
      "data": true,
      "service": "postgres",
      "resurrect": true,
      "resources": {
        "max_fd": {
          "request": 10000,
          "limit": 10000
        },
        "memory": {
          "request": 1073741824,
          "limit": 1073741824
        }
      }
    },
    "web": {
      "cmd": [
        "api"
      ],
      "ports": [
        {
          "port": 80,
          "proto": "tcp"
        }
      ],
      "resources": {
        "max_fd": {
```

```
            "request": 10000,
            "limit": 10000
          },
          "memory": {
            "request": 1073741824,
            "limit": 1073741824
          }
        }
      }
    }
  },
  "artifact": {
    "id": "e3e69c33-dd0c-4381-b9a9-30cf0c417ce7",
    "type": "docker",
    "uri":
"https://dl.flynn.io/tuf?name=flynn/postgresql&id=7447010d0a300f45d6785bb6cef48
58500821d7afcb8febaabefbe55bf03fd0f"
  },
  "processes": {
    "postgres": 1,
    "web": 1
  },
  "updated_at": "2015-12-16T02:20:44.152134Z"
 }
]
```


部署

https://flynn.io/schema/controller/deployment#(https://flynn.io/schema/controller/deployment#)

属性	类型	描述	
id	uuid string	唯一标示	
app	uuid string	唯一标示	
old_release	object	唯一标示	
new_release	date-time string	唯一标示	
status	date-time string	pending running complete failed	

strategy	uuid string	all-at-once one-by-one postgres discoverd-meta	
processes	uuid string	每种进程运行的进程数	
deploy_timeout	integer	部署超时时间（默认30秒）	
created_at	date-time string	对象的创建时间	
finished_at	date-time string		
name	string	null	
url	url string	null	

创建部署

实例：

————————————————————————请求
————————————————————————

```
POST /apps/7406a4d71a0c43d3ac4b39f006cb0342/deploy HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "id": "40202cf3b0e946a4b2c4db42a0c14194"
}
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "ccef998b56374bca8d4c402c655b4534",
  "app": "7406a4d71a0c43d3ac4b39f006cb0342",
  "old_release": "689ce5b9ad1541ab975d51cba1e051d0",
  "new_release": "40202cf3b0e946a4b2c4db42a0c14194",
  "strategy": "all-at-once",
  "created_at": "2015-01-29T18:53:26.927187Z"
```

```
}
```

获取部署

实例：

————————————————————————请求
————————————————————————

```
GET /deployments/aab1ee14-776d-4ba4-979b-1b4bda2d9b35 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "aab1ee14-776d-4ba4-979b-1b4bda2d9b35",
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "old_release": "47154f8c-a604-469d-ae6a-e431990ddee8",
  "new_release": "77e9e956-ecf9-427f-a031-222c2f394fb8",
  "strategy": "all-at-once",
  "status": "pending",
  "processes": {
    "foo": 1
  },
  "deploy_timeout": 30,
  "created_at": "2015-12-16T02:21:16.782263Z"
}
```

列出所有部署

实例：

————————————————————————请求
————————————————————————

```
GET /apps/adcccdb4-b1a4-4209-a03a-762f4e021632/deployments HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
[
  {
    "id": "aab1ee14-776d-4ba4-979b-1b4bda2d9b35",
    "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
    "old_release": "47154f8c-a604-469d-ae6a-e431990ddee8",
    "new_release": "77e9e956-ecf9-427f-a031-222c2f394fb8",
    "strategy": "all-at-once",
    "status": "pending",
    "processes": {
      "foo": 1
    },
    "deploy_timeout": 30,
    "created_at": "2015-12-16T02:21:16.782263Z"
  }
]
```

新任务

新任务描述了一个所需流程配置

https://flynn.io/schema/controller/new_job#(https://flynn.io/schema/controller/new_job#)

属性	类型	描述	
release	uuid string	唯一标示	
cmd	array of strings	shell命令	
entrypoint	array of strings		
env	object	环境变量	
meta	object	客户端指定元数据	
tty	boolean	初始化tty会话	
tty_columns	integer	tty中的列数	
tty_lines	integer	tty中的行数	
release_env	boolean	包括版本环境	

disable_log	boolean	不拷贝标准输入/输出到日志流中	
resources	object	资源请求和限制	

运行任务

实例：

————————————————————请求
————————————————————

```
POST /apps/7406a4d71a0c43d3ac4b39f006cb0342/jobs HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "release": "40202cf3b0e946a4b2c4db42a0c14194",
  "cmd": [
    "echo",
    "$BODY"
  ],
  "env": {
    "BODY": "Hello!"
  }
}
```

————————————————————响应
————————————————————

```
Content-Type: application/json
```

```
{
  "id": "flynn-cef74685c83b47889c69fa95451e75b3",
  "release": "40202cf3b0e946a4b2c4db42a0c14194",
  "cmd": [
    "echo",
    "$BODY"
  ]
}
```

任务

一个任务就是容器中的一个单进程。

https://flynn.io/schema/controller/job#(https://flynn.io/schema/controller/job#)

属性	类型	描述
id	uuid string	唯一标示
app	uuid string	唯一标示
release	uuid string	唯一标示
type	string	进程类型名字
state	string	客户端指定元数据
cmd	array of strings	shell命令
meta	object	客户端指定元数据
exit_status	integer	任务退出状态
created_at	date-time string	对象创建时间
updated_at	date-time string	对象最新更新时间

获取任务列表

实例：

————————————————————————请求
————————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342/jobs HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/jsonb
```

```
[
  {
    "id": "flynn-68940e0fe9664ce0b369baaeb10b4cc4",
    "app": "7406a4d71a0c43d3ac4b39f006cb0342",
    "release": "40202cf3b0e946a4b2c4db42a0c14194",
    "type": "foo",
    "state": "starting",
```

```
    "created_at": "2015-01-29T18:53:28.595924Z",
    "updated_at": "2015-01-29T18:53:28.595924Z"
  },
  {
    "id": "flynn-cef74685c83b47889c69fa95451e75b3",
    "app": "7406a4d71a0c43d3ac4b39f006cb0342",
    "release": "40202cf3b0e946a4b2c4db42a0c14194",
    "state": "starting",
    "created_at": "2015-01-29T18:53:27.81058Z",
    "updated_at": "2015-01-29T18:53:27.81058Z"
  },
  {
    "id": "flynn-c96be7a9527047e4b11e3c19d3bef904",
    "app": "7406a4d71a0c43d3ac4b39f006cb0342",
    "release": "689ce5b9ad1541ab975d51cba1e051d0",
    "type": "foo",
    "state": "crashed",
    "created_at": "2015-01-29T18:53:26.933346Z",
    "updated_at": "2015-01-29T18:53:27.954494Z"
  }
]
```


更新任务

实例：

————————————————————————请求
————————————————————————

```
PUT /apps/7406a4d71a0c43d3ac4b39f006cb0342/jobs/flynn-
cef74685c83b47889c69fa95451e75b3 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/
```


```
{
  "id": "flynn-cef74685c83b47889c69fa95451e75b3",
  "app": "7406a4d71a0c43d3ac4b39f006cb0342",
  "release": "40202cf3b0e946a4b2c4db42a0c14194",
  "state": "down"
}
```


————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "flynn-cef74685c83b47889c69fa95451e75b3",
  "app": "7406a4d71a0c43d3ac4b39f006cb0342",
  "release": "40202cf3b0e946a4b2c4db42a0c14194",
  "state": "down",
  "created_at": "2015-01-29T18:53:27.81058Z",
  "updated_at": "2015-01-29T18:53:28.723854Z"
}
```

获取任务

实例：

————————————————————————请求
————————————————————————

```
GET /apps/adcccdb4-b1a4-4209-a03a-762f4e021632/jobs/host-40cc2d07-7a48-4fda-9790-ba9768a3f616 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "host-40cc2d07-7a48-4fda-9790-ba9768a3f616",
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "release": "77e9e956-ecf9-427f-a031-222c2f394fb8",
  "state": "down",
  "created_at": "2015-12-16T02:21:16.799294Z",
  "updated_at": "2015-12-16T02:21:16.811094Z"
}
```

删除任务

实例：

————————————————————————请求
————————————————————————

```
DELETE /apps/7406a4d71a0c43d3ac4b39f006cb0342/jobs/flynn-
```

```
cef74685c83b47889c69fa95451e75b3 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————————响应
————————————————————

```
Content-Type: text/plain; charset=utf-8
```

端口

https://flynn.io/schema/controller/port#(https://flynn.io/schema/controller/port#)

属性	类型	描述
port	integer	null
proto	string	tcp udp

进程类型

https://flynn.io/schema/controller/process_type#(https://flynn.io/schema/controller/process_type#)

属性	类型	描述
cmd	array of strings	shell命令
env	object	环境变量
entrypoint	array of strings	null
ports	array of objects	null
ports[].port	integer	null
ports[].proto	string	null
data	boolean	null
omni	boolean	null

提供程序

https://flynn.io/schema/controller/provider#(https://flynn.io/schema/controller/provider#)

属性	类型	描述
id	uuid string	唯一标识符
created_at	date-time string	对象的创建时间戳
updated_at	date-time string	对象最后更新的时间戳
name	string	null
url	uri string	null

创建提供程序

实例：

————————————————————————请求
————————————————————————

```
POST /providers HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "url": "http://example-provider-
1422557608732167310.discoverd:12345/providers/1422557608732167310",
  "name": "example-provider-1422557608732167310"
}
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "7277db2655e44b4ba7128ff5ff01a0ff",
  "url": "http://example-provider-
1422557608732167310.discoverd:12345/providers/1422557608732167310",
  "name": "example-provider-1422557608732167310",
  "created_at": "2015-01-29T18:53:28.733018Z",
  "updated_at": "2015-01-29T18:53:28.733018Z"
}
```

获取提供程序

实例：

————————————————————————请求
————————————————————

```
GET /providers/7277db2655e44b4ba7128ff5ff01a0ff HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————

```
Content-Type: application/json
```

```
{
  "id": "7277db2655e44b4ba7128ff5ff01a0ff",
  "url": "http://example-provider-
1422557608732167310.discoverd:12345/providers/1422557608732167310",
  "name": "example-provider-1422557608732167310",
  "created_at": "2015-01-29T18:53:28.733018Z",
  "updated_at": "2015-01-29T18:53:28.733018Z"
}
```

获取提供程序列表

实例：

————————————————————————请求
————————————————————

```
GET /providers HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————

```
Content-Type: application/json
```

```
[
  {
    "id": "7277db2655e44b4ba7128ff5ff01a0ff",
    "url": "http://example-provider-
1422557608732167310.discoverd:12345/providers/1422557608732167310",
    "name": "example-provider-1422557608732167310",
    "created_at": "2015-01-29T18:53:28.733018Z",
```

```
    "updated_at": "2015-01-29T18:53:28.733018Z"
  },
  {
    "id": "d9ba709dc3e7413db3cba9b86e7cf325",
    "url": "http://pg-api.discoverd/databases",
    "name": "postgres",
    "created_at": "2015-01-29T18:53:20.031975Z",
    "updated_at": "2015-01-29T18:53:20.031975Z"
  }
]
```

版本

https://flynn.io/schema/controller/release#(https://flynn.io/schema/controller/release#)

属性	类型	描述
id	uuid string	唯一标识
artifact	uuid string	唯一标识
meta	object	客户指定元数据
env	object	环境变量
processes	object	
created_at	object	对象的创建时间戳

获取应用初始化版本

实例：

————————————————————————请求
————————————————————————

```
GET /apps/gitreceive/release HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "8a9760deeba7486c987267196edd9032",
  "artifact": "74010365c551463386bae39cf4ad0a47",
  "processes": {
    "app": {
      "env": {
        "CONTROLLER_AUTH_KEY": "s3cr3t",
        "SLUGBUILDER_IMAGE_URI":
"https://registry.hub.docker.com/flynn/slugbuilder?id=a4a8a9489eacbad331ced4ad5b
4b63238c7592d42d30b04058aac914912344f3",
        "SLUGRUNNER_IMAGE_URI":
"https://registry.hub.docker.com/flynn/slugrunner?id=039222f9884412eb0667988933c
aeb0ed06da28f71e3dff540a61d13d0d85ae0",
        "SSH_PRIVATE_KEYS": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEpAIBAAKCAQEAvVM5o2jcAO9qN/Bza+omOXxqJlPySHkxGEmpBBGOJ/thjF63\njr
x4nsDAYB8x5fkVYNR4QPqIEfAAvvRAK1walOPrvPcrtp/TAD1csYYADIj11gs7\nD5VcNo4
6H9gc8Bw7ZQN+vB20/ZfhjdJHVVS3oi1co74rBIwEQq3FEQUTMEsLESJb\nzr4JqllKq4Vs
DHUqxEcIUVllnt0l/3Y1ou9kp2s0EGIzbGL8W0ku8V+PrG3/3iI9\n8eEQ043hawSrQTYY8z
CQxQvss3GlKpMQDT4y14gS6gnD6F9Xf26ZOqVvkezshDtR\n8hrsSnaAB2JO3WCvu410
1nE2Ahqm5SlogrL8rwIDAQABAoIBAG5b5wtEAophBK+a\n8A0FrkZwKYgyAtcQHfgkZ+t
BOHZB6HjkdWc0obMYdIsTZAjECxwoffiSkWKzXhvx\nUL51r+D+gtnh0o9f1qUbH78zdy/
XXkXowgKxc9ExUtxhI20rZ8vYH7YUMv5n3EjJ\nPF+XRHu6qXehUH4UCB++H1Nue+L3u
lk9GViGO3/PXSbF0awVl3c2VKvgrGvMkgKP\nT+JH7KZj318m3DM+w4xGBpvhNAcTHf7
+WXO+IToGect/be5aQ4dDycnp28Sb0vdR\nvgmMEMrP+AqkHiaduGWyXjWpLImdJox
qTZT9Y/XxIT4u9e4PEVcuC2QL0PinyY5C\nDSq20iECgYEA6dJaAcpeKMRcr4cVxogC6JUG
jxIBK8YTZoowsfWjYMDsG92SP8Bg\nNT0QHFSDIzC+UB8SG705bRes+T2woqeTiwudik
OKsEl4ey1NbAPc5o1JFqKv5M3V\nOeSeyOOviZMCKM2WksOE/T1P8vGcnHJyu7fc1/FR
nJn/GiO1Negu+DkCgYEAz0ho\nxmDaq+ekkoYNkIQyVPgrT9v7u9V8A9fZ3O0wCcFQiT
pQ7YhrwPkItfZZsm26BqkR\nOsj+78PQlX9iGLmIDvkmRV2FilG3tZhLFiLpOSiq9pQegbz
DdThrWYpPNgDINkDs\nmOjK67Eb1anywVPpLCtCDWT87wgdQCI1cBosjCcCgYEAvcqs
nH9y5oxc320606db\nFQbCCCeJI8rnvHe3Bvxn3QU7oXK8X1r7vLsP9K70sv/Buq89t55km
pR5JuZYFe8H\n5/lJqw0f6e/5B6LCEUcha1KLXQA43fPle9SCvnVx6q2VNyGJcx+ZNyUpW
/2aLbMI\nRi7755xNrluhocoSMOuCFKECgYAA0OYoscTqnFTXhtOS2BJipz90ZwdMrAB6
+2SO\nLBjeUPaCM8qRxNG3xsM0BC5CN5Bd6lh5BMWMKBcwH3pBYqmYdX6jE0UtUD
kK6/iP\ns797TB+wLNEZ7aZPVdOGXLcHHWWsYQENcQ0rdF0JsEuWW0A1vk3aQ6WDQ
2LqF7hi\nIpg+TQKBgQDalWyZwwbIYI8K6BB2wA8PkZV46egnLQCqaGmhPdJcdE7tBjm
dTGgj\nVZ0d77Cxik08hsQofyXjB6zufih/zan4VV37pBgQ7SZFBgOEu8ScqMz1EpeVCFC3
\nFvOUZ0ed0/mql57JqsYKJ3ZroKMAJZs0wLKYOLZ8N88fVNJdmWI16g==\n-----END
RSA PRIVATE KEY-----\n-----BEGIN EC PRIVATE KEY-----
\nMHcCAQEEIFGhKRLx3WqbFxYc9RMwIaJK/xa7WWcj90p2q/3DTNdPoAoGCCqGSM49
\nAwEHoUQDQgAE7m4eZSDcX7hJ0+iyv8m1t0f/BkQW6tBO3McnTYWjoU07JN/mxYlk\
nAfiET9Yqq5viMFr2ydOKXGlVU5khmjiOnA==\n-----END EC PRIVATE KEY-----\n"
```

```
      },
      "ports": [
        {
          "port": 0,
          "proto": "tcp",
          "range_end": 0
        }
      ]
    }
  },
  "created_at": "2015-01-29T18:53:22.471286Z"
}
```

设置应用版本

实例：

————————————————————请求
————————————————————

```
PUT /apps/7406a4d71a0c43d3ac4b39f006cb0342/release HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "id": "689ce5b9ad1541ab975d51cba1e051d0"
}
```

————————————————————响应
————————————————————

```
Content-Type: application/json
```

获取应用版本

实例：

————————————————————请求
————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342/release HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————响应
————————————————————

```
Content-Type: application/json
```


```
{
  "id": "689ce5b9ad1541ab975d51cba1e051d0",
  "artifact": "4d365e6cdb1d47b8b98c9cd9a7b047fd",
  "env": {
    "some": "info"
  },
  "processes": {
    "foo": {
      "cmd": [
        "ls",
        "-l"
      ],
      "env": {
        "BAR": "baz"
      }
    }
  },
  "created_at": "2015-01-29T18:53:26.89224Z"
}
```


创建版本

实例：

————————————————————————————请求
————————————————————————————

```
POST /releases HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```


```
{
  "artifact": "4d365e6cdb1d47b8b98c9cd9a7b047fd",
  "env": {
    "some": "info"
  },
  "processes": {
    "foo": {
      "cmd": [
        "ls",
        "-l"
```

```
      ],
      "env": {
        "BAR": "baz"
      }
    }
  }
}
```

——————————————————————————响应
————————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "689ce5b9ad1541ab975d51cba1e051d0",
  "artifact": "4d365e6cdb1d47b8b98c9cd9a7b047fd",
  "env": {
    "some": "info"
  },
  "processes": {
    "foo": {
      "cmd": [
        "ls",
        "-l"
      ],
      "env": {
        "BAR": "baz"
      }
    }
  },
  "created_at": "2015-01-29T18:53:26.89224Z"
}
```

获取版本列表

实例：

——————————————————————————请求
————————————————————————————

```
GET /releases HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

——————————————————————————响应
————————————————————————————

Content-Type: application/json


```
[
  {
    "id": "689ce5b9ad1541ab975d51cba1e051d0",
    "artifact": "4d365e6cdb1d47b8b98c9cd9a7b047fd",
    "env": {
      "some": "info"
    },
    "processes": {
      "foo": {
        "cmd": [
          "ls",
          "-l"
        ],
        "env": {
          "BAR": "baz"
        }
      }
    },
    "created_at": "2015-01-29T18:53:26.89224Z"
  },
  {
    "id": "258c9a2f43124c76b85ba24fa127dc3b",
    "artifact": "8b6548e25bc8477e905deee8a33eab2a",
    "env": {
      "APP_NAME": "dashboard",
      "CA_CERT": "-----BEGIN CERTIFICATE-----
\nMIIC5DCCAc6gAwIBAgIRAL1sRMV4diGUf9Lztaw9/OkwCwYJKoZIhvcNAQELMB0x\n
DjAMBgNVBAoTBUZseW5uMQswCQYDVQQLEwJDQTAeFw0xNTAxMjkxODUzMjNaFw
0y\nMDAxMjgxODUzMjNaMB0xDjAMBgNVBAoTBUZseW5uMQswCQYDVQQLEwJDQT
CCASIw\nDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJeMZ5RC7Aan+JImyIDEN4
dfikjO\nr323YWPdYiLM7bi/o4x6Lp2+LOeeV+1Y7xky3gybuekBtNveEp6wHg3pqXCvAfx
V\nQ4EGDXs0/9RyXNCgHsiDmnftB1OtmjPTZjtbgDWVhkJZcq1X6hlcTgD9rs1HOVmg\n
7m/gPCa0avI30QTajb2/DBQbKIQovY7vcQLaPbhqknVK1WZoCnxDomT0ueFllD8h\n6p/
sJ5qk/ZQF2WL8gL9elQOUm/K7jUjkfLbHN3hzx2QQgNqlTDKMgPApMri5UjUa\nnxdLoY
OPiUj0QZmMyZkaYwG0cyGp/8MuEzY6F/dgSEt1gemS5lGI7s+TDNECAwEA\nAaMjMCE
wDgYDVR0PAQH/BAQDAgAGMA8GA1UdEwEB/wQFMAMBAf8wCwYJKoZIhvcN\nAQE
LA4IBAQAjiAj10RbsCxzp0wp79g/BLRaYcypFWnjenn6aWUD2pvc7oK2n8XX1\nbhSym0
```

```
O4aUlTbx1lc+N4R+MI/Hx83MPFEMXNoLGMnwZ+6IxYKuoXnqt6rFdm2kh4\nOUABwk
nyNx4jLFniAEIlPa+fcbHacgJtkxm85OpdW9kVrZU3Af7NaXl9u1TDJ0ww\nwtOlMRGBMB
yuKxU5zwXBz28rgmBIZ2F3w5guZxnNUDKuk8z9WIowJumB+DymDYxh\nSSAfL2jHwdD
BeZK5XMmXRobWpf4oSm2z6Nx+0RWHms8tKTQXUDOlwLdMMho1jj/l\nJAvqe8tmpQ
Bg90lc3bCCLmaeJG3nkfSA\n-----END CERTIFICATE-----\n",
    "CONTROLLER_DOMAIN": "controller.dev.localflynn.com",
    "CONTROLLER_KEY": "s3cr3t",
    "DEFAULT_ROUTE_DOMAIN": "dev.localflynn.com",
    "LOGIN_TOKEN": "a9b51e131d3f444074aa35425eb8fdc3",
    "SECURE_COOKIES": "true",
    "SESSION_SECRET": "4fd6d9c447b02a5e3b8d03ab818652d9",
    "STATIC_PATH": "/app",
    "URL": "https://dashboard.dev.localflynn.com"
  },
  "processes": {
    "web": {
      "ports": [
        {
          "port": 80,
          "proto": "tcp",
          "range_end": 0
        }
      ]
    }
  },
  "created_at": "2015-01-29T18:53:25.003718Z"
 },
 {
  "id": "b9aa14dd9fbe4ea6a06acd88da3b3c80",
  "artifact": "03f9d97aeab94f2b9bb23f4ade0eb270",
  "env": {
    "CONTROLLER_AUTH_KEY": "s3cr3t",
    "SLUGBUILDER_IMAGE_URI":
"https://registry.hub.docker.com/flynn/slugbuilder?id=a4a8a9489eacbad331ced4ad5b
4b63238c7592d42d30b04058aac914912344f3",
    "SLUGRUNNER_IMAGE_URI":
"https://registry.hub.docker.com/flynn/slugrunner?id=039222f9884412eb0667988933c
aeb0ed06da28f71e3dff540a61d13d0d85ae0"
  },
  "created_at": "2015-01-29T18:53:24.986924Z"
 },
 {
```

```
    "id": "8a9760deeba7486c987267196edd9032",
    "artifact": "74010365c551463386bae39cf4ad0a47",
    "processes": {
      "app": {
        "env": {
          "CONTROLLER_AUTH_KEY": "s3cr3t",
          "SLUGBUILDER_IMAGE_URI":
"https://registry.hub.docker.com/flynn/slugbuilder?id=a4a8a9489eacbad331ced4ad5b
4b63238c7592d42d30b04058aac914912344f3",
          "SLUGRUNNER_IMAGE_URI":
"https://registry.hub.docker.com/flynn/slugrunner?id=039222f9884412eb0667988933c
aeb0ed06da28f71e3dff540a61d13d0d85ae0",
          "SSH_PRIVATE_KEYS": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEpAIBAAKCAQEAvVM5o2jcAO9qN/Bza+omOXxqJlPySHkxGEmpBBGOJ/thjF63\njr
x4nsDAYB8x5fkVYNR4QPqIEfAAvvRAK1walOPrvPcrtp/TAD1csYYADIj11gs7\nD5VcNo4
6H9gc8Bw7ZQN+vB20/ZfhjdJHVVS3oi1co74rBIwEQq3FEQUTMEsLESJb\nzr4JqllKq4Vs
DHUqxEcIUVllnt0l/3Y1ou9kp2s0EGIzbGL8W0ku8V+PrG3/3iI9\n8eEQ043hawSrQTYY8z
CQxQvss3GlKpMQDT4y14gS6gnD6F9Xf26ZOqVvkezshDtR\n8hrsSnaAB2JO3WCvu410
1nE2Ahqm5SlogrL8rwIDAQABAoIBAG5b5wtEAophBK+a\n8A0FrkZwKYgyAtcQHfgkZ+t
BOHZB6HjkdWc0obMYdIsTZAjECxwoffiSkWKzXhvx\nUL51r+D+gtnh0o9f1qUbH78zdy/
XXkXowgKxc9ExUtxhI20rZ8vYH7YUMv5n3EjJ\nPF+XRHu6qXehUH4UCB++H1Nue+L3u
lk9GViGO3/PXSbF0awVl3c2VKvgrGvMkgKP\nT+JH7KZj318m3DM+w4xGBpvhNAcTHf7
+WXO+IToGect/be5aQ4dDycnp28Sb0vdR\nvgmMEMrP+AqkHiaduGWyXjWpLImdJox
qTZT9Y/XxIT4u9e4PEVcuC2QL0PinyY5C\nDSq20iECgYEA6dJaAcpeKMRcr4cVxogC6JUG
jxIBK8YTZoowsfWjYMDsG92SP8Bg\nNT0QHFSDIzC+UB8SG705bRes+T2woqeTiwudik
OKsEl4ey1NbAPc5o1JFqKv5M3V\nOeSeyOOviZMCKM2WksOE/T1P8vGcnHJyu7fc1/FR
nJn/GiO1Negu+DkCgYEAz0ho\nxmDaq+ekkoYNkIQyVPgrT9v7u9V8A9fZ3O0wCcFQiT
pQ7YhrwPkItfZZsm26BqkR\nOsj+78PQlX9iGLmIDvkmRV2FilG3tZhLFiLpOSiq9pQegbz
DdThrWYpPNgDINkDs\nmOjK67Eb1anywVPpLCtCDWT87wgdQCI1cBosjCcCgYEAvcqs
nH9y5oxc320606db\nFQbCCCeJI8rnvHe3Bvxn3QU7oXK8X1r7vLsP9K70sv/Buq89t55km
pR5JuZYFe8H\n5/lJqw0f6e/5B6LCEUcha1KLXQA43fPle9SCvnVx6q2VNyGJcx+ZNyUpW
/2aLbMI\nRi7755xNrluhocoSMOuCFKECgYAA0OYoscTqnFTXhtOS2BJipz90ZwdMrAB6
+2SO\nLBjeUPaCM8qRxNG3xsM0BC5CN5Bd6lh5BMWMKBcwH3pBYqmYdX6jE0UtUD
kK6/iP\ns797TB+wLNEZ7aZPVdOGXLcHHWWsYQENcQ0rdF0JsEuWW0A1vk3aQ6WDQ
2LqF7hi\nIpg+TQKBgQDalWyZwwbIYI8K6BB2wA8PkZV46egnLQCqaGmhPdJcdE7tBjm
dTGgj\nVZ0d77Cxik08hsQofyXjB6zufih/zan4VV37pBgQ7SZFBgOEu8ScqMz1EpeVCFC3
\nFvOUZ0ed0/mql57JqsYKJ3ZroKMAJZs0wLKYOLZ8N88fVNJdmWI16g==\n-----END
RSA PRIVATE KEY-----\n-----BEGIN EC PRIVATE KEY-----
\nMHcCAQEEIFGhKRLx3WqbFxYc9RMwIaJK/xa7WWcj90p2q/3DTNdPoAoGCCqGSM49
\nAwEHoUQDQgAE7m4eZSDcX7hJ0+iyv8m1t0f/BkQW6tBO3McnTYWjoU07JN/mxYlk\
nAfiET9Yqq5viMFr2ydOKXGlVU5khmjiOnA==\n-----END EC PRIVATE KEY-----\n"
        },
```

```
      "ports": [
       {
        "port": 0,
        "proto": "tcp",
        "range_end": 0
       }
      ]
     }
    },
    "created_at": "2015-01-29T18:53:22.471286Z"
  },
  {
    "id": "2222dd828450483d96b425b8b1294040",
    "artifact": "73d37c1bdacd432f9e5c3cbe121c365e",
    "processes": {
     "app": {
      "cmd": [
       "-httpaddr",
       ":80",
       "-httpsaddr",
       ":443",
       "-tcp-range-start",
       "3000",
       "-tcp-range-end",
       "3500"
      ],
      "omni": true,
      "host_network": true
     }
    },
    "created_at": "2015-01-29T18:53:21.730018Z"
  },
  {
    "id": "e48208857b6e42c1baaba738e766ec7e",
    "artifact": "afe339ec9a0f46b28cfa5dff383e5179",
    "env": {
     "FLYNN_POSTGRES": "pg",
     "PGDATABASE": "c69edca77f27f07c70f10521dc1b9271",
     "PGPASSWORD": "5883fcfe1603a1bc54883cab0e835b6e",
     "PGUSER": "af3dd154197bb764a5fb377fa6d39f31"
    },
    "processes": {
```

```
    "web": {
      "ports": [
        {
          "port": 80,
          "proto": "tcp",
          "range_end": 0
        }
      ]
    }
  },
  "created_at": "2015-01-29T18:53:21.713992Z"
},
{
  "id": "fdaba774be0a4332bfb5be1d440cdc60",
  "artifact": "61b804fd0f9c42359ac8d8e3131d16bf",
  "processes": {
    "postgres": {
      "cmd": [
        "postgres"
      ],
      "ports": [
        {
          "port": 5432,
          "proto": "tcp",
          "range_end": 0
        }
      ],
      "data": true
    },
    "web": {
      "cmd": [
        "api"
      ],
      "ports": [
        {
          "port": 80,
          "proto": "tcp",
          "range_end": 0
        }
      ]
    }
  },
```

```
  "created_at": "2015-01-29T18:53:20.058243Z"
 },
 {
  "id": "a665aa801cbb45f4a41831b66416f4b9",
  "artifact": "1b7c8dcfd0b14755a452367e5f6723b6",
  "env": {
   "AUTH_KEY": "s3cr3t",
   "BACKOFF_PERIOD": "",
   "DEFAULT_ROUTE_DOMAIN": "dev.localflynn.com",
   "FLYNN_POSTGRES": "pg",
   "NAME_SEED": "f09d8d597f0a39acf991",
   "PGDATABASE": "51b7cf0edd53ca6921147823f40737eb",
   "PGPASSWORD": "4bbb7b6bdff41444d876e02e9bfc7133",
   "PGUSER": "f331aec77952871eeec5f7b5f16253bd"
  },
  "processes": {
   "deployer": {
    "cmd": [
     "deployer"
    ]
   },
   "scheduler": {
    "cmd": [
     "scheduler"
    ],
    "omni": true
   },
   "web": {
    "cmd": [
     "controller"
    ],
    "ports": [
     {
      "port": 80,
      "proto": "tcp",
      "range_end": 0
     }
    ]
   }
  },
  "created_at": "2015-01-29T18:53:20.02833Z"
 }
```

```
]
```

资源请求

https://flynn.io/schema/controller/resource_req#(https://flynn.io/schema/controller/resource_req#)

属性	类型	描述
apps	array of uuid string	
config	object	

创建资源

实例：

————————————————————————————请求
————————————————————————————

```
POST /providers/7277db2655e44b4ba7128ff5ff01a0ff/resources HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "config": {
  }
}
```

————————————————————————————响应
————————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "5489d2758d9b452783de686c0ca5fe9d",
  "provider": "7277db2655e44b4ba7128ff5ff01a0ff",
  "env": {
    "some": "data"
  },
  "created_at": "2015-01-29T18:53:28.757571Z"
}
```

资源

https://flynn.io/schema/controller/resource#(https://flynn.io/schema/controller/resource#)

属性	类型	描述
id	uuid string	唯一标示
provider	uuid string	唯一标示
external_id	uuid string	唯一标示
env	object	环境变量
apps	array of uuid strings	
created_at	date-time string	对象创建时间

添加资源

实例：

————————————————————————请求
————————————————————————

```
PUT /providers/0952f692-2667-4be0-a159-9d68382a262c/resources/cc9f3342-bed0-4ed3-840e-c462e05808c6 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "id": "cc9f3342-bed0-4ed3-840e-c462e05808c6",
  "provider": "0952f692-2667-4be0-a159-9d68382a262c",
  "external_id": "/foo/bar",
  "env": {
    "FOO": "BAR"
  },
  "apps": [
    "adcccdb4-b1a4-4209-a03a-762f4e021632"
  ]
}
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
```

```
  "id": "cc9f3342-bed0-4ed3-840e-c462e05808c6",
  "provider": "0952f692-2667-4be0-a159-9d68382a262c",
  "external_id": "/foo/bar",
  "env": {
    "FOO": "BAR"
  },
  "apps": [
    "adcccdb4-b1a4-4209-a03a-762f4e021632"
  ],
  "created_at": "2015-12-16T02:21:16.838613Z"
}
```

列出应用资源

实例：

——————————————————————————请求
————————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342/resources HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

——————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
[
  {
    "id": "cc9f3342-bed0-4ed3-840e-c462e05808c6",
    "provider": "0952f692-2667-4be0-a159-9d68382a262c",
    "external_id": "/foo/bar",
    "env": {
      "FOO": "BAR"
    },
    "apps": [
      "adcccdb4-b1a4-4209-a03a-762f4e021632"
    ],
    "created_at": "2015-12-16T02:21:16.838613Z"
  }
]
```

获取资源

实例：

——————————————————————————请求
————————————————————————

```
GET
/providers/7277db2655e44b4ba7128ff5ff01a0ff/resources/5489d2758d9b452783de68
6c0ca5fe9d HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

——————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "5489d2758d9b452783de686c0ca5fe9d",
  "provider": "7277db2655e44b4ba7128ff5ff01a0ff",
  "env": {
    "some": "data"
  },
  "created_at": "2015-01-29T18:53:28.757571Z"
}
```

获取资源列表

实例：

——————————————————————————请求
————————————————————————

```
GET /providers/7277db2655e44b4ba7128ff5ff01a0ff/resources HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

——————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
[
  {
    "id": "5489d2758d9b452783de686c0ca5fe9d",
    "provider": "7277db2655e44b4ba7128ff5ff01a0ff",
```

```
    "env": {
      "some": "data"
    },
    "created_at": "2015-01-29T18:53:28.757571Z"
  }
]
```

删除资源

实例：

————————————————————————请求
————————————————————————

```
DELETE /providers/0952f692-2667-4be0-a159-9d68382a262c/resources/ HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: text/plain; charset=utf-8
```

路由

https://flynn.io/schema/controller/route#(https://flynn.io/schema/controller/route#)

属性	类型	描述
id	uuid string	唯一标示
parent_ref	uuid string	唯一标示
type	string	
service	uuid string	唯一标示
domain	string	路由域名。仅适用于HTTP路由。
tls_cert	string	路由可选的TLS公证书。仅用于HTTP路由。
tls_key	string	路由可选的TLS私钥。仅用于HTTP路由。

sticky	boolean	无论路由是否使用粘性会话。仅用于HTTP路由。
port	integer	TCP端口上监听TCP路由。
created_at	date-time string	对象创建的时间
updated_at	date-time string	对象最新更新时间

创建路由

实例：

————————————————————————————请求
————————————————————————————

```
POST /apps/7406a4d71a0c43d3ac4b39f006cb0342/routes HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "type": "http",
  "config": {
    "domain": "http://example.com",
    "service": "my-app-1422557606845347930-web"
  }
}
```

————————————————————————————响应
————————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "http/a9b9f04336ce0181a08e774e01113b31",
  "parent_ref": "controller/apps/7406a4d71a0c43d3ac4b39f006cb0342",
  "type": "http",
  "config": {
    "domain": "http://example.com",
    "service": "my-app-1422557606845347930-web"
  },
  "created_at": "2015-01-29T18:53:26.870533465Z",
  "updated_at": "2015-01-29T18:53:26.870533465Z"
```

```
}
```

获取路由

实例：

————————————————————————请求
————————————————————————

```
GET
/apps/7406a4d71a0c43d3ac4b39f006cb0342/routes/http/a9b9f04336ce0181a08e774e
01113b31 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": "http/a9b9f04336ce0181a08e774e01113b31",
  "parent_ref": "controller/apps/7406a4d71a0c43d3ac4b39f006cb0342",
  "type": "http",
  "config": {
    "domain": "http://example.com",
    "service": "my-app-1422557606845347930-web"
  },
  "created_at": "2015-01-29T18:53:26.870533465Z",
  "updated_at": "2015-01-29T18:53:26.870533465Z"
}
```

更新路由

实例：

————————————————————————请求
————————————————————————

```
PUT /apps/adcccdb4-b1a4-4209-a03a-762f4e021632/routes/http/5bfb9c8b-ae1f-
4a5a-af0c-94fa2996d543 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
```

```
  "type": "http",
  "id": "5bfb9c8b-ae1f-4a5a-af0c-94fa2996d543",
  "parent_ref": "controller/apps/adcccdb4-b1a4-4209-a03a-762f4e021632",
  "service": "my-app-1450232456657062340-other",
  "created_at": "2015-12-16T02:21:06.704111Z",
  "updated_at": "2015-12-16T02:21:06.704111Z",
  "domain": "http://example.com",
  "sticky": true,
  "path": "/"
}
```

————————————————————————响应
————————————————————

```
Content-Type: application/json
```

```
{
  "type": "http",
  "id": "5bfb9c8b-ae1f-4a5a-af0c-94fa2996d543",
  "parent_ref": "controller/apps/adcccdb4-b1a4-4209-a03a-762f4e021632",
  "service": "my-app-1450232456657062340-other",
  "created_at": "2015-12-16T02:21:06.704111Z",
  "updated_at": "2015-12-16T02:21:06.71424Z",
  "domain": "http://example.com",
  "sticky": true,
  "path": "/"
}
```

获取路由列表

实例：

————————————————————————请求
————————————————————

```
GET /apps/7406a4d71a0c43d3ac4b39f006cb0342/routes HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————

```
Content-Type: application/json
```

```
[
  {
    "id": "http/a9b9f04336ce0181a08e774e01113b31",
    "parent_ref": "controller/apps/7406a4d71a0c43d3ac4b39f006cb0342",
    "type": "http",
    "config": {
      "domain": "http://example.com",
      "service": "my-app-1422557606845347930-web"
    },
    "created_at": "2015-01-29T18:53:26.870533465Z",
    "updated_at": "2015-01-29T18:53:26.870533465Z"
  },
  {
    "id": "http/55b43353e8ae0779b3579ef7661bee91",
    "parent_ref": "controller/apps/7406a4d71a0c43d3ac4b39f006cb0342",
    "type": "http",
    "config": {
      "domain": "my-app-1422557606845347930.dev.localflynn.com",
      "service": "my-app-1422557606845347930-web"
    },
    "created_at": "2015-01-29T18:53:26.847692447Z",
    "updated_at": "2015-01-29T18:53:26.847692447Z"
  }
]
```


删除路由

实例：

————————————————————————请求
————————————————————————

```
DELETE
/apps/7406a4d71a0c43d3ac4b39f006cb0342/routes/http/a9b9f04336ce0181a08e774e
01113b31 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: text/plain; charset=utf-8
```

错误

https://flynn.io/schema/controller/error#(https://flynn.io/schema/controller/error#)

属性	类型	描述
code	string	not_found object_not_found object_exists syntax_error validation_error unknown_error
message	string	
detail	boolean	

创建无效应用

实例：

————————————————————————请求
————————————————————————

```
POST /apps HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "name": "this is not valid",
  "protected": false
}
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "code": "validation_error",
  "message": "name String must match the pattern: \"^[a-z\\d]+(-[a-z\\d]+)*$\".",
  "detail": {
    "field": "name"
  }
}
```

事件

https://flynn.io/schema/controller/event#(https://flynn.io/schema/controller/event)

属性	类型	描述
id	integer	
app	uuid string	唯一标志符
object_type	string	app app_deletion app_release deployment job scale release artifact provider resource resource_deletion key key_deletion route route_deletion domain_migration
object_id	uuid string	唯一标志符
data	object	
created_at	date-time string	对象创建时间戳

事件流

属性	类型	描述
app_id	uuid string	唯一标志符
object_types	string	app app_deletion app_release deployment job scale release artifact provider resource resource_deletion key key_deletion route route_deletion domain_migration
past	boolean	返回之前运行的事件
count	integer	限制返回的之前运行事件的数量

实例：

————————————————————————请求
————————————————————

```
GET /events?count=10&past=true HTTP/1.1
Accept: text/event-stream
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
Last-Event-Id: 0
```

————————————————————————响应
————————————————————

```
Content-Type: text/event-stream; charset=utf-8
```


```
data: {"id":102,"app":"adcccdb4-b1a4-4209-a03a-
762f4e021632","object_type":"job","object_id":"host-26c0ea34-e655-40d2-8d5e-
c5694470af6c","data":{"id":"host-26c0ea34-e655-40d2-8d5e-
c5694470af6c","app":"adcccdb4-b1a4-4209-a03a-
762f4e021632","meta":{"bread":"with
hemp"},"type":"foo","state":"up","release":"47154f8c-a604-469d-ae6a-
e431990ddee8","created_at":"2015-12-16T02:21:06.765744Z","updated_at":"2015-12-
16T02:21:07.689399Z"},"created_at":"2015-12-16T02:21:07.691418Z"}

data: {"id":103,"app":"adcccdb4-b1a4-4209-a03a-
762f4e021632","object_type":"job","object_id":"host-26c0ea34-e655-40d2-8d5e-
c5694470af6c","data":{"id":"host-26c0ea34-e655-40d2-8d5e-
c5694470af6c","app":"adcccdb4-b1a4-4209-a03a-
762f4e021632","meta":{"bread":"with
hemp"},"type":"foo","state":"down","release":"47154f8c-a604-469d-ae6a-
e431990ddee8","created_at":"2015-12-16T02:21:06.765744Z","updated_at":"2015-12-
16T02:21:07.704057Z","exit_status":2},"created_at":"2015-12-16T02:21:07.707269Z"}

data: {"id":104,"app":"adcccdb4-b1a4-4209-a03a-
762f4e021632","object_type":"job","object_id":"host-e6582097-9aee-4b6d-94fc-
83c0a3d1b51c","data":{"id":"host-e6582097-9aee-4b6d-94fc-
83c0a3d1b51c","app":"adcccdb4-b1a4-4209-a03a-
762f4e021632","meta":{"bread":"with
hemp"},"type":"foo","state":"starting","release":"47154f8c-a604-469d-ae6a-
e431990ddee8","created_at":"2015-12-16T02:21:07.710312Z","updated_at":"2015-12-
16T02:21:07.710312Z"},"created_at":"2015-12-16T02:21:07.711479Z"}

data: {"id":105,"app":"adcccdb4-b1a4-4209-a03a-
```

762f4e021632","object_type":"job","object_id":"host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c","data":{"id":"host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c","app":"adcccdb4-b1a4-4209-a03a-762f4e021632","meta":{"bread":"with hemp"},"type":"foo","state":"up","release":"47154f8c-a604-469d-ae6a-e431990ddee8","created_at":"2015-12-16T02:21:07.710312Z","updated_at":"2015-12-16T02:21:08.529747Z"},"created_at":"2015-12-16T02:21:08.531191Z"}

data: {"id":106,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"job","object_id":"host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c","data":{"id":"host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c","app":"adcccdb4-b1a4-4209-a03a-762f4e021632","meta":{"bread":"with hemp"},"type":"foo","state":"down","release":"47154f8c-a604-469d-ae6a-e431990ddee8","created_at":"2015-12-16T02:21:07.710312Z","updated_at":"2015-12-16T02:21:08.534753Z","exit_status":2},"created_at":"2015-12-16T02:21:08.5359Z"}

data: {"id":107,"object_type":"release","object_id":"77e9e956-ecf9-427f-a031-222c2f394fb8","data":{"id":"77e9e956-ecf9-427f-a031-222c2f394fb8","env":{"some":"info"},"artifact":"c1889f55-c244-43ce-af70-ead357daa6ec","processes":{"foo":{"cmd":["ls","-l"],"env":{"BAR":"baz"},"resources":{"max_fd":{"limit":10000,"request":10000},"memory":{"limit":1073741824,"request":1073741824}}}},"created_at":"2015-12-16T02:21:16.775714Z"},"created_at":"2015-12-16T02:21:16.775714Z"}

data: {"id":108,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"deployment","object_id":"aab1ee14-776d-4ba4-979b-1b4bda2d9b35","data":{"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","status":"pending","release":"77e9e956-ecf9-427f-a031-222c2f394fb8","deployment":"aab1ee14-776d-4ba4-979b-1b4bda2d9b35"},"created_at":"2015-12-16T02:21:16.784374Z"}

data: {"id":109,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"job","object_id":"host-40cc2d07-7a48-4fda-9790-ba9768a3f616","data":{"id":"host-40cc2d07-7a48-4fda-9790-ba9768a3f616","app":"adcccdb4-b1a4-4209-a03a-762f4e021632","state":"starting","release":"77e9e956-ecf9-427f-a031-222c2f394fb8","created_at":"2015-12-16T02:21:16.799294Z","updated_at":"2015-12-16T02:21:16.799294Z"},"created_at":"2015-12-16T02:21:16.806296Z"}

data: {"id":110,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"job","object_id":"host-40cc2d07-7a48-4fda-9790-

ba9768a3f616","data":{"id":"host-40cc2d07-7a48-4fda-9790-ba9768a3f616","app":"adcccdb4-b1a4-4209-a03a-762f4e021632","state":"down","release":"77e9e956-ecf9-427f-a031-222c2f394fb8","created_at":"2015-12-16T02:21:16.799294Z","updated_at":"2015-12-16T02:21:16.811094Z"},"created_at":"2015-12-16T02:21:16.812493Z"}

data: {"id":111,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"resource","object_id":"cc9f3342-bed0-4ed3-840e-c462e05808c6","data":{"id":"cc9f3342-bed0-4ed3-840e-c462e05808c6","env":{"FOO":"BAR"},"apps":["adcccdb4-b1a4-4209-a03a-762f4e021632"],"provider":"0952f692-2667-4be0-a159-9d68382a262c","created_at":"2015-12-16T02:21:16.838613Z","external_id":"/foo/bar"},"created_at":"2015-12-16T02:21:16.838613Z"}

data: {"id":112,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"deployment","object_id":"aab1ee14-776d-4ba4-979b-1b4bda2d9b35","data":{"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","status":"running","release":"77e9e956-ecf9-427f-a031-222c2f394fb8","job_type":"foo","job_state":"starting","deployment":"aab1ee14-776d-4ba4-979b-1b4bda2d9b35"},"created_at":"2015-12-16T02:21:17.017287Z"}

data: {"id":113,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"scale","object_id":"adcccdb4-b1a4-4209-a03a-762f4e021632:77e9e956-ecf9-427f-a031-222c2f394fb8","data":{"release":"77e9e956-ecf9-427f-a031-222c2f394fb8","processes":{"foo":1}},"created_at":"2015-12-16T02:21:17.018787Z"}

data: {"id":114,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"job","object_id":"host-8ba33370-e3b5-4b09-8eaa-629279403202","data":{"id":"host-8ba33370-e3b5-4b09-8eaa-629279403202","app":"adcccdb4-b1a4-4209-a03a-762f4e021632","meta":{"bread":"with hemp"},"type":"foo","state":"starting","release":"77e9e956-ecf9-427f-a031-222c2f394fb8","created_at":"2015-12-16T02:21:17.062294Z","updated_at":"2015-12-16T02:21:17.062294Z"},"created_at":"2015-12-16T02:21:17.064426Z"}

data: {"id":115,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"route_deletion","object_id":"32cd199d-9e83-48e6-96c2-05ab7167eab8","data":{"id":"32cd199d-9e83-48e6-96c2-05ab7167eab8","path":"/","type":"http","domain":"my-app-1450232456657062340.dev.localflynn.com","service":"my-app-1450232456657062340-

web","created_at":"2015-12-16T02:20:56.661301Z","parent_ref":"controller/apps/adcccdb4-b1a4-4209-a03a-762f4e021632","updated_at":"2015-12-16T02:20:56.661301Z"},"created_at":"2015-12-16T02:21:17.532255Z"}

data: {"id":116,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"resource_deletion","object_id":"cc9f3342-bed0-4ed3-840e-c462e05808c6","data":{"id":"cc9f3342-bed0-4ed3-840e-c462e05808c6","env":{"FOO":"BAR"},"apps":["adcccdb4-b1a4-4209-a03a-762f4e021632"],"provider":"0952f692-2667-4be0-a159-9d68382a262c","created_at":"2015-12-16T02:21:16.838613Z","external_id":"/foo/bar"},"created_at":"2015-12-16T02:21:17.537413Z"}

data: {"id":117,"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","object_type":"app_deletion","object_id":"adcccdb4-b1a4-4209-a03a-762f4e021632","data":{"error":"","app_deletion":{"app":"adcccdb4-b1a4-4209-a03a-762f4e021632","deleted_routes":[{"id":"32cd199d-9e83-48e6-96c2-05ab7167eab8","path":"/","type":"http","domain":"my-app-1450232456657062340.dev.localflynn.com","service":"my-app-1450232456657062340-web","created_at":"2015-12-16T02:20:56.661301Z","parent_ref":"controller/apps/adcccdb4-b1a4-4209-a03a-762f4e021632","updated_at":"2015-12-16T02:20:56.661301Z"}],"deleted_resources":[{"id":"cc9f3342-bed0-4ed3-840e-c462e05808c6","env":{"FOO":"BAR"},"apps":["adcccdb4-b1a4-4209-a03a-762f4e021632"],"provider":"0952f692-2667-4be0-a159-9d68382a262c","created_at":"2015-12-16T02:21:16.838613Z","external_id":"/foo/bar"}]}},"created_at":"2015-12-16T02:21:17.564196Z"}

列出事件

属性	类型	描述
app_id	uuid string	唯一标志符

object_types	string	app app_deletion app_release deployment job scale release artifact provider resource resource_deletion key key_deletion route route_deletion domain_migration
before_id	integer	只返回发生在指定 ID 事件之前的事件
since_id	integer	只返回发生在指定 ID 事件之后的事件
count	integer	限制返回的之前运行事件的数量

实例：

————————————————————————请求
————————————————————————

```
GET /events?count=10 HTTP/1.1
Accept: application/json
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: text/event-stream; charset=utf-8
```

```
[
 {
  "id": 111,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "resource",
  "object_id": "cc9f3342-bed0-4ed3-840e-c462e05808c6",
  "data": {
   "id": "cc9f3342-bed0-4ed3-840e-c462e05808c6",
   "env": {
    "FOO": "BAR"
   },
   "apps": [
```

```
      "adcccdb4-b1a4-4209-a03a-762f4e021632"
    ],
    "provider": "0952f692-2667-4be0-a159-9d68382a262c",
    "created_at": "2015-12-16T02:21:16.838613Z",
    "external_id": "/foo/bar"
  },
  "created_at": "2015-12-16T02:21:16.838613Z"
},
{
  "id": 110,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "job",
  "object_id": "host-40cc2d07-7a48-4fda-9790-ba9768a3f616",
  "data": {
    "id": "host-40cc2d07-7a48-4fda-9790-ba9768a3f616",
    "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
    "state": "down",
    "release": "77e9e956-ecf9-427f-a031-222c2f394fb8",
    "created_at": "2015-12-16T02:21:16.799294Z",
    "updated_at": "2015-12-16T02:21:16.811094Z"
  },
  "created_at": "2015-12-16T02:21:16.812493Z"
},
{
  "id": 109,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "job",
  "object_id": "host-40cc2d07-7a48-4fda-9790-ba9768a3f616",
  "data": {
    "id": "host-40cc2d07-7a48-4fda-9790-ba9768a3f616",
    "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
    "state": "starting",
    "release": "77e9e956-ecf9-427f-a031-222c2f394fb8",
    "created_at": "2015-12-16T02:21:16.799294Z",
    "updated_at": "2015-12-16T02:21:16.799294Z"
  },
  "created_at": "2015-12-16T02:21:16.806296Z"
},
{
  "id": 108,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "deployment",
```

```
  "object_id": "aab1ee14-776d-4ba4-979b-1b4bda2d9b35",
  "data": {
    "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
    "status": "pending",
    "release": "77e9e956-ecf9-427f-a031-222c2f394fb8",
    "deployment": "aab1ee14-776d-4ba4-979b-1b4bda2d9b35"
  },
  "created_at": "2015-12-16T02:21:16.784374Z"
},
{
  "id": 107,
  "object_type": "release",
  "object_id": "77e9e956-ecf9-427f-a031-222c2f394fb8",
  "data": {
    "id": "77e9e956-ecf9-427f-a031-222c2f394fb8",
    "env": {
      "some": "info"
    },
    "artifact": "c1889f55-c244-43ce-af70-ead357daa6ec",
    "processes": {
      "foo": {
        "cmd": [
          "ls",
          "-l"
        ],
        "env": {
          "BAR": "baz"
        },
        "resources": {
          "max_fd": {
            "limit": 10000,
            "request": 10000
          },
          "memory": {
            "limit": 1073741824,
            "request": 1073741824
          }
        }
      }
    },
    "created_at": "2015-12-16T02:21:16.775714Z"
  },
```

```
  "created_at": "2015-12-16T02:21:16.775714Z"
 },
 {
  "id": 106,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "job",
  "object_id": "host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c",
  "data": {
   "id": "host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c",
   "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
   "meta": {
    "bread": "with hemp"
   },
   "type": "foo",
   "state": "down",
   "release": "47154f8c-a604-469d-ae6a-e431990ddee8",
   "created_at": "2015-12-16T02:21:07.710312Z",
   "updated_at": "2015-12-16T02:21:08.534753Z",
   "exit_status": 2
  },
  "created_at": "2015-12-16T02:21:08.5359Z"
 },
 {
  "id": 105,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "job",
  "object_id": "host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c",
  "data": {
   "id": "host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c",
   "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
   "meta": {
    "bread": "with hemp"
   },
   "type": "foo",
   "state": "up",
   "release": "47154f8c-a604-469d-ae6a-e431990ddee8",
   "created_at": "2015-12-16T02:21:07.710312Z",
   "updated_at": "2015-12-16T02:21:08.529747Z"
  },
  "created_at": "2015-12-16T02:21:08.531191Z"
 },
 {
```

```
  "id": 104,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "job",
  "object_id": "host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c",
  "data": {
    "id": "host-e6582097-9aee-4b6d-94fc-83c0a3d1b51c",
    "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
    "meta": {
      "bread": "with hemp"
    },
    "type": "foo",
    "state": "starting",
    "release": "47154f8c-a604-469d-ae6a-e431990ddee8",
    "created_at": "2015-12-16T02:21:07.710312Z",
    "updated_at": "2015-12-16T02:21:07.710312Z"
  },
  "created_at": "2015-12-16T02:21:07.711479Z"
},
{
  "id": 103,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "job",
  "object_id": "host-26c0ea34-e655-40d2-8d5e-c5694470af6c",
  "data": {
    "id": "host-26c0ea34-e655-40d2-8d5e-c5694470af6c",
    "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
    "meta": {
      "bread": "with hemp"
    },
    "type": "foo",
    "state": "down",
    "release": "47154f8c-a604-469d-ae6a-e431990ddee8",
    "created_at": "2015-12-16T02:21:06.765744Z",
    "updated_at": "2015-12-16T02:21:07.704057Z",
    "exit_status": 2
  },
  "created_at": "2015-12-16T02:21:07.707269Z"
},
{
  "id": 102,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "job",
```

```
    "object_id": "host-26c0ea34-e655-40d2-8d5e-c5694470af6c",
    "data": {
      "id": "host-26c0ea34-e655-40d2-8d5e-c5694470af6c",
      "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
      "meta": {
        "bread": "with hemp"
      },
      "type": "foo",
      "state": "up",
      "release": "47154f8c-a604-469d-ae6a-e431990ddee8",
      "created_at": "2015-12-16T02:21:06.765744Z",
      "updated_at": "2015-12-16T02:21:07.689399Z"
    },
    "created_at": "2015-12-16T02:21:07.691418Z"
  }
]
```

获取事件

实例：

————————————————————————请求
————————————————————————

```
GET /events/111 HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/json
```

```
{
  "id": 111,
  "app": "adcccdb4-b1a4-4209-a03a-762f4e021632",
  "object_type": "resource",
  "object_id": "cc9f3342-bed0-4ed3-840e-c462e05808c6",
  "data": {
    "id": "cc9f3342-bed0-4ed3-840e-c462e05808c6",
    "env": {
      "FOO": "BAR"
    },
    "apps": [
```

```
      "adcccdb4-b1a4-4209-a03a-762f4e021632"
    ],
    "provider": "0952f692-2667-4be0-a159-9d68382a262c",
    "created_at": "2015-12-16T02:21:16.838613Z",
    "external_id": "/foo/bar"
  },
  "created_at": "2015-12-16T02:21:16.838613Z"
}
```

CA 证书

获取 CA 证书

实例：

————————————————————————请求
————————————————————————

```
GET /ca-cert HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————响应
————————————————————————

```
Content-Type: application/x-x509-ca-cert
```

```
-----BEGIN CERTIFICATE-----
MIIDBDCCAe6gAwIBAgIRAP6BP9TQKrq0l1CCcSI4gb4wCwYJKoZIhvcNAQELMC0x
DjAMBgNVBAoTBUZseW5uMRswGQYDVQQLExJGbHlubiBFcGhlbWVyYWwgQ0EwHhc
N
MTUxMjE2MDIyMDQxWhcNMjAxMjE0MDIyMDQxWjAtMQ4wDAYDVQQKEwVGbHlubj
Eb
MBkGA1UECxMSRmx5bm4gRXBoZW1lcmFsIENBMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAn+KUccUSVeWoJZTlOxqaiXacGECGAPETxfmnL9ep1rA/4DWi
2OTpZoeSFdyL1yV5KJxPkq6XMNyl/BR72SOomCmoFRnZzDih//+gfuJAFVGaF1j9
FOIX+VyO8jyFzqk2vZg9R+ncRyFgjEgpHn0gDdt4WLWDxPpfFMClEZU6Mpu02ngj
1kmlNgzajl71qhhKtrv3W7ACKh6O3fMSBg4n/ZiVnVQjdkejHskV5R43FH4bpZzP
hb82rI6FNBNOCKqwbQ/AdGGVvJSzXPypvzCMkGghhx8pckoZw7DONXhSojfIO6Kt
lxUYO1Odmnb+nyJjxGyidSperhfm+CuSFPn2BwIDAQABoyMwITAOBgNVHQ8BAf8E
BAMCAAYwDwYDVR0TAQH/BAUwAwEB/zALBgkqhkiG9w0BAQsDggEBAGvEqq1ko+pE
S5RBHmTWW6mJom9rA+aii9n88lMhbdjikeVFkQkH6Qi8zR5O4hCdDUC4qYUp+rIF
Z4XZ9Ogl4CcR1gEPGu1KX2EnjVKloS1LThugCuP5YJgQ5qoMsQs681s1ZpPd/0gh
```

```
tjASQy1Y72wY1SxVWqJzkuzf8mMDg52I4wwD9Eif2zq1/+sJa9mdilhvd0Wp+hDV
6qN6xbH65V2VKwDrkD2p7CIn3NmknYhmw+obezqRkSl6N5Is/QPVg59bOu0evWfS
DyaOehm1EmmRVehNj8IR0noFwUMbIASnpTescLd4Nz9pEjbgHQ+m6Swe8hnaZ3dd
1TvxABm/4VY=
-----END CERTIFICATE-----
```

备份

获取集群备份

下载集群的全备份

实例：

————————————————————————————请求

```
GET /backup HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

————————————————————————————响应

```
Content-Disposition: attachment; filename="flynn-backup-2015-12-16_022126.tar"
Content-Type: application/tar
```

域名迁移

https://flynn.io/schema/controller/domain_migration#(https://flynn.io/schema/controller/domain_migration)

属性	类型	描述
id	uuid string	唯一标志符
old_tls_cert	string	旧域名的 TLS 证书，迁移任务时会设置它
tls_cert	string	域名的 TLS 证书，若没有会生成一个
old_domain	uri string	旧的迁出的域名
domain	uri string	新的迁入的域名

created_at	date-time string	对象创建时间戳
finished_at	date-time string	迁入完成时间

迁移集群域名

将集群域名迁移的任务排队。从 domain_migration 事件流来看什么时候完成集群域名迁移。

实例：

————————————————————————请求
————————————————————

```
PUT /domain HTTP/1.1
Authorization: Basic OnMzY3IzdA==
Content-Type: application/json
```

```
{
  "id": "",
  "old_domain": "dev.localflynn.com",
  "domain": "127.0.0.1.xip.io"
}
```

————————————————————————响应
————————————————————

```
Content-Type: application/json
```

```
{
  "id": "5d4c321b-fe6a-42df-867e-ea1a55282396",
  "old_tls_cert": {
    "ca_cert": "",
    "cert": "-----BEGIN CERTIFICATE-----
\nMIIDSjCCAjSgAwIBAgIQDH8zACle7B+pqJB3QfwthDALBgkqhkiG9w0BAQswLTEO\nM
AwGA1UEChMFRmx5bm4xGzAZBgNVBAsTEkZseW5uIEVwaGVtZXJhbCBDQTAeFw0x\n
NTEyMTYwMjIwNDJaFw0yMDEyMTQwMjIwNDJaMC0xDjAMBgNVBAoTBUZseW5uMRs
w\nGQYDVQQDExJkZXYubG9jYWxmbHlubi5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4I
B\nDwAwggEKAoIBAQC3NsEp+fYsbgr94cOBVnSK3ZcHBobwcCdzNvNqh6y1V9wn7x4
C\nS6Rk0pn1Zz/FS1F+uA8KMVUzwWCCSl/jW4BNAqbLbVi4I5oap8mCDQDt+MDPZyn
5\noOCn6vj5OCJXva3RmhJRocF88oXcH4lXWUyfKMOb+0/+tRC7lg5gqBxnvkiJrW7D\n
yq0+FoZViBgrN3JcxAgWFx9FK8sITzWpgPqJvJiS4Ry54o8hBTQCKyFCHTwdoufd\nKyBw
m6tNn/KLhUXcAEsBD3qvdI0AEX9rssGxVErCualgPaBAvDCFz2H+wwvKnHpK\nCcwIflE0
4YYOjCHphq9j7FIzd5/+fp0Ng53pAgMBAAGjajBoMA4GA1UdDwEB/wQE\nAwIAoDATB
gNVHSUEDDAKBggrBgEFBQcDATAMBgNVHRMBAf8EAjAAMDMGA1UdEQQs\nMCqCE
mRldi5sb2NhbGZseW5uLmNvbYIUKi5kZXYubG9jYWxmbHlubi5jb20wCwYJ\nKoZIhvcN
```

```
AQELA4IBAQAIR7k3teJZ5c2eLUlcfLSZiigz+RaZeqxHHt6fZXA9P9XS\n+m8NSIUffdfOXa
XjokYTYc91RqGhxVGUUopEA7P64QkEJRpu4qcfq6CXxDoExyf5\ncumrzrBZmTvExqlGBb
yviz7T4c7IZAe0U3zYCoaVd1Ss84mY8DA37oqM2Y1pH7Tc\nzwBprXDqAfwt70xoYFWQ
ZoK6SbS+qCQkc+0K9HW0ZzGRwqMTbAoPOTn9D2c9f9o5\nuM9KkiTuvY3T1Kimb848
IVvhBfEofeHXZspc9HWqfLJM8wIpMJCaDwcDBN+yFku6\nqqaRa9SXkZLNrB+eCqsVOU
zBsmK+sgUL6jmM+Pfc\n-----END CERTIFICATE-----\n",
    "pin": "",
    "key": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEogIBAAKCAQEAtzbBKfn2LG4K/eHDgVZ0it2XBwaG8HAnczbzaoestVfcJ+8e\nAku
kZNKZ9Wc/xUtRfrgPCjFVM8Fggkpf41uATQKmy21YuCOaGqfJgg0A7fjAz2cp\n+aDgp+
r4+TgiV72t0ZoSUaHBfPKF3B+JV1lMnyjDm/tP/rUQu5YOYKgcZ75Iia1u\nw8qtPhaGVYg
YKzdyXMQIFhcfRSvLCE81qYD6ibyYkuEcueKPIQU0AishQh08HaLn\n3SsgcJurTZ/yi4VF3
ABLAQ96r3SNABF/a7LBsVRKwrmpYD2gQLwwhc9h/sMLypx6\nSgnMCH5RNOGGDowh
6YavY+xSM3ef/n6dDYOd6QIDAQABAoIBAGEZTsRdal9frsmg\n1gl89WUHTVx21BsnhX
DlZZpG73xoZRBxBPl4d1bL2raZks2QPk+nYqknPh2g1fIX\nUrxfKR8nUZisigGFNM39sKk
RFTYJyL3i0fcvkQDNYkhKvMC9kRptylmjHbWHnvPr\nbtFxTU9ovkqzpVC7S3Pvcg0961f7
NgUqM5bfzWlgZYDATtNTpngc6/8bUgG8xhdL\nLsfd84FJcC/0k09Gxu5GPuwvcws1Iag3
9+Rkz2G9KmOEnN8dywLLedWPQ8A6fIjP\ngSSYTPWRBEF7V6/yRb7a0z1M+SJUrA0vD
YJsa2P2ry0YslJi1O9ipOeBmmCddVjN\n21SywoECgYEA45mlj46ySdKhB7iRDaD+M1Ffo
9vaCniGjZxEH+sPWXet3gr/hTGN\nmsQngU66v/Gj6clTSOlkOImQHT7Q0/GeHTDEAVEX
JNTX8NllElJe6LjOUyqc7kUN\n/zASp3YOHWVC8y3cNy8ScydUhO5hkaVJcpqO0RWxnW
nqV6m2OOtcyvkCgYEAzhNA\ntxEpP/7qMz2/Kx7LflRXtMAIfecIlLmq4mw0pnMCebCCX
Op7/Lw7VOLIz1JfzGtW\n2b6BTspzbTW16rjedA65+Pj/EdmXxtL7KIfDlnmab7cSQ3Vi/T
mbza4Z+etZ8Yy2\nRwA2Ni8IPS2y8NukcAJH+fGjIAbS/FqgR6KJtnECgYB955scpRWfnP
MTsFgdr/ev\nL862eIJP1Iiqgc110rS1C22VLYRYjCcHfIdWEtVgJS5Fv+dgqCuW0xJz+zHO
bRTI\nlRd4mwEwkMW8KKau0f5KwyDwBNy3OmAZ0OAg3z/Dpya4G6B8rn+IUDODLdO
vnD3b\nNgDXTSbqK5NFV07Egf7dEQKBgHdthvrRckOJW3dY2JqH3U0YWkABbqpwo3N
16sgQ\nB69TReL6XYK1zidZe1UlnVUxmx1gM9Q7aLsd7ykikIw4mYtPIY5d140juqdajH4f\
nuEVTsiH+ShJYTIcOxPG9bYZ+VAD+GIEraT+boD28Z1DZKgJUABylYnFkIxBnAS6i\nzwax
AoGAa3TWYsK0VUrn1EwlI9zzCLW8myjzyPLgpEQ6264ETNSw4K42M2fthcS4\nzlVMrPJ5
yhftxCkxbXNFp74lt5Nhryv9K3RWdnL6aXoxR4juhfXBPUBPK+bk/g6F\nohfHj/PeDJFBe0
MWC1Q4x7jbeHgqXQ9q7mun+9xH8ZziMdknHc4=\n-----END RSA PRIVATE KEY-----
\n"
  },
  "old_domain": "dev.localflynn.com",
  "domain": "127.0.0.1.xip.io",
  "created_at": "2015-12-16T02:21:26.911192Z"
}
```


路由

路由

https://flynn.io/schema/router/route#(https://flynn.io/schema/router/route#)

属性	类型	描述
id	uuid string	唯一标示
parent_ref	uuid string	唯一标示
type	string	http tcp
service	uuid string	唯一标示
domain	string	路由域名。仅适用于 HTTP 路由。
tls_cert	string	路由可选的 TLS 证书。仅用于 HTTP 路由。
tls_key	string	路由可选的 TLS 私钥。仅用于 HTTP 路由。
path	string	路由到这个服务的可选路径。仅用于 HTTP 路由，且不与 TLS 选项公用
sticky	boolean	路由是否使用粘性会话。仅用于 HTTP 路由。
port	integer	TCP 端口上监听 TCP 路由。
created_at	date-time string	对象创建的时间戳
updated_at	date-time string	对象最新更新的时间戳

创建路由

实例：

————————————————————————请求————————————————————————

```
POST /routes HTTP/1.1
Content-Type: application/json
```

```
{
```

```
  "type": "http",
  "service": "foo-web",
  "created_at": "0001-01-01T00:00:00Z",
  "updated_at": "0001-01-01T00:00:00Z",
  "domain": "http://example.com"
}
```

————————————————————响应————————————————————

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

```
{
  "type": "http",
  "id": "e8080915-1d48-4374-ac4f-c54a88bdd210",
  "service": "foo-web",
  "created_at": "2015-02-16T21:19:04.116018Z",
  "updated_at": "2015-02-16T21:19:04.116018Z",
  "domain": "http://example.com"
}
```

更新路由

实例：

————————————————————请求————————————————————

```
PUT /routes/http/e8080915-1d48-4374-ac4f-c54a88bdd210 HTTP/1.1
Content-Type: application/json
```

```
{
  "type": "http",
  "id": "e8080915-1d48-4374-ac4f-c54a88bdd210",
  "service": "bar-web",
  "created_at": "0001-01-01T00:00:00Z",
  "updated_at": "0001-01-01T00:00:00Z",
  "domain": "http://example.com"
}
```

————————————————————响应————————————————————

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

```
{
  "type": "http",
```

```
  "id": "e8080915-1d48-4374-ac4f-c54a88bdd210",
  "service": "bar-web",
  "created_at": "2015-02-16T21:19:04.116018Z",
  "updated_at": "2015-02-16T21:19:04.11883Z",
  "domain": "http://example.com"
}
```

获取路由

实例：

————————————————————请求————————————————————

```
GET /routes/http/e8080915-1d48-4374-ac4f-c54a88bdd210 HTTP/1.1
Content-Type: application/json
```

————————————————————响应————————————————————

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

```
{
  "type": "http",
  "id": "e8080915-1d48-4374-ac4f-c54a88bdd210",
  "service": "bar-web",
  "created_at": "2015-02-16T21:19:04.116018Z",
  "updated_at": "2015-02-16T21:19:04.11883Z",
  "domain": "http://example.com"
}
```

获取路由列表

实例：

————————————————————请求————————————————————

```
GET /routes HTTP/1.1
Content-Type: application/json
```

————————————————————响应————————————————————

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

```
[
  {
    "type": "http",
```

```
    "id": "e8080915-1d48-4374-ac4f-c54a88bdd210",
    "service": "bar-web",
    "created_at": "2015-02-16T21:19:04.116018Z",
    "updated_at": "2015-02-16T21:19:04.11883Z",
    "domain": "http://example.com"
  },
  {
    "type": "http",
    "id": "6391ee42-7db1-4387-95da-75cfee2e80e1",
    "parent_ref": "controller/apps/61c01ac5e89140d39410395cdfcf38fa",
    "service": "dashboard-web",
    "created_at": "2015-02-16T21:19:02.094033Z",
    "updated_at": "2015-02-16T21:19:02.094033Z",
    "domain": "dashboard.dev.localflynn.com",
    "tls_cert": "-----BEGIN CERTIFICATE-----
\nMIIDOjCCAiSgAwIBAgIQLcnOaDV3iECxwsxhMPbvSTALBgkqhkiG9w0BAQswHTEO\n
MAwGA1UEChMFRmx5bm4xCzAJBgNVBAsTAkNBMB4XDTE1MDIxNjIxMTkwMVoXDTI
w\nMDIxNTIxMTkwMVowLTEOMAwGA1UEChMFRmx5bm4xGzAZBgNVBAMTEmRldi5s
b2Nh\nbGZseW5uLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MZc97RD\nC66GKEpKRmtCazziQl+M4M8P0OfWyN73IANpCiu6pbgdXuCVsQFjC/kLu
WwINxp9\ntvXvjWk5UYyU5EX0ZMUu6uaN7DQ8tIUPV89lpFx1bIbH+vtx/KvhUtUc0eNT
0leG\n2Jd6PgO468oFYAT8I+QPe3ngpbNvLWofZe59nOTYWvxBfHoE2kacFUCaTYC9fgj
q\nnh6Pn9xcBM3pIJQ4m9kLjW6EPGQrEnFd/ryLo6a8UGh+OdGh/3LuTZBkch5zetbu\na
5bL3MX0WEkDoRdRXOI7+nh8d0LIcvaZqD4kiN2YMmtU87J0N47pHu4YyoArwILA\nTrK
N7+gjX8EvWwkCAwEAAaNqMGgwDgYDVR0PAQH/BAQDAgCgMBMGA1UdJQQMMA
oG\nCCsGAQUFBwMBMAwGA1UdEwEB/wQCMAAwMwYDVR0RBCwwKoISZGV2LmxvY
2FsZmx5\nbm4uY29tghQqLmRldi5sb2NhbGZseW5uLmNvbTALBgkqhkiG9w0BAQsDgg
EBACtZ\ndCxpaBJXHlRHz8co8LReHeyCez1GKlqD1bCD1qYxToEZPuWmi39xTtdCG2ZO
A7SO\nhVZbG4vT4Ra5ANXWtoYco71UZeNn/viJXfF5FxKH8u1rNAsTH8uLdi4PO4Shy4/
d\n2DBueWG1E4BB6LKRfkxSxqEaFeLuV8Z+IEfPaibvEG8G2cvMtK9frCjSSn+iPDtC\neS
WnOvgoWmI8GEj9tdFpxwVjDuzK1wcjxkOprjigMV1Oh19D4u+rnjsZKzFNpgfK\nI768GF
sL0eA42rDjyN6Q20oZ+4AJPCN7dHBszRZM1G5bfQE9e6C5JIYgbMEBDLJ\nbNGj5eq9
CKY9WAVz6bw=\n-----END CERTIFICATE-----\n",
    "tls_key": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEpAIBAAKCAQEAxlz3tEMLroYoSkpGa0JrPOJCX4zgzw/Q59bI3vcgA2kKK7ql\nuB1e
4JWxAWML+Qu5bAg3Gn229e+NaTlRjJTkRfRkxS7q5o3sNDy0hQ9Xz2WkXHVs\nhsf6+
3H8q+FS1RzR41PSV4bYl3o+A7jrygVgBPwj5A97eeCls28tah9l7n2c5Nha\n/EF8egTaRpw
VQJpNgL1+COqeHo+f3FwEzekglDib2QuNboQ8ZCsScV3+vIujprxQ\naH450aH/cu5Nk
GRyHnN61u5rlsvcxfRYSQOhF1Fc4jv6eHx3Qshy9pmoPiSI3Zgy\na1TzsnQ3juke7hjKgCv
AgsBOso3v6CNfwS9bCQIDAQABAoIBAEvPhh/Pq+xEe6uA\n3Y7qsH9xXbqU5epc+hNR
BC4qtoJ3J5r015cZKoSc0SxdhVrmPzRpo12thDpUFnCL\npn4a/W866zRtLPt2bDq+pKNh
7MZn6zwm6JZOYb6tnsq17+lg/VprU6197NNftcnI\nZ90q1rqw/qqPvDolQE7mg63WWC
```

egD724BhVRBo8FUmcrn91uqt2plfCMnnzZPTbL\nB2Hk575ww+vjAysdihavJAeWtAVO7
BpAFcQoDXjbFZXUxWaZbFd1r15PE55uJ/uh\nx0VdQ9GMGahF+yRyOqaKRnovEp0twKj
e0BgGFNwloU90VBL2HipuWvK0tM+pImCf\nX5rHcF0CgYEA09wbQZe7EIt2uWSyM5P0
aMtrReWXHtSJaXUU5u4e1cF/J9KeeZfS\nMNXzIAcsJdZfF2jCyrQs5s+gE5d7yRWHl3L+q
ng18/yiLX9JXSVGtxiLIFdrFpqd\nGrgxMwyf8eGAUdFJnBE1nu4CKytVFTksMdHz0lZTDge
1xyQ8zCvyqrMCgYEA77D+\n4dbX7GV8R1P6Lu7IAeACPuccHEPyGO0lrCiUZtrvzbyKa7n
TI1ELeBCHzty4akGX\nR6lnWpUW0X0LZcsZAQzMfVYezwPwpgIJqtf529TEjEfvUsc9aYOP
RvtoDtnSZ4dz\nz9ChbsRG5tt+Abipva80XqHYv+G4ZyKMNlOzcVMCgYB8/PX3lJSABkdc
yNN5Kmif\nCqOW8QRe4/TNio1yUDlQ0n8590AXbEJNHWv6EUEycW5YWS3SeMmkwg
kmhdA5qcH8\nz5L+7zylXP4w6U9W7I7zpj8uLWS4SS82fERKYcBa6zbRDi/nKx//S1tsLHx
dsQUp\nvzuhc9w8AsBzZCWaLrNTkQKBgQC6hGry5rJDnH1nK+wFVX6CJwTYWhM/04e
y6edk\n3ncMytXMrSixx3qLgud/2K01xgAjf0LvWt24WUfWYI3b+n1Mmd8OsVNsGE7Nf
8xy\nC3q9HAaOevO64JvjXaLCw6Qn2kymj3sJffoBLe9WefrF8CL2Hj6MXTlDp6xNZU7i\nf
6OlTQKBgQCYa/mTIT4NZuyoQitUIl8GyqeR3GHyaSfxCaPlynUG3iHdlr2NOfL1\nGlx5tG
wcawJ1DUe/eKzA3Cj7IM+UcDTdIJM2t3dbQOPh/qEPY54NKTh5yHZ+qeuO\nrpSlpImS
AUx46q7rnnm8KlEYKg6xVEwACHJFk55WxilDmee+/ZACag==\n-----END RSA PRIVATE
KEY-----\n"
  },
  {
    "type": "http",
    "id": "4cf50c90-4f33-43a6-af4b-f1695fd17ee4",
    "parent_ref": "controller/apps/0fa0d7a91e634d58be2a758e9e109ee8",
    "service": "flynn-controller",
    "created_at": "2015-02-16T21:19:01.537271Z",
    "updated_at": "2015-02-16T21:19:01.537271Z",
    "domain": "controller.dev.localflynn.com",
    "tls_cert": "-----BEGIN CERTIFICATE-----
\nMIIDOjCCAiSgAwIBAgIQLcnOaDV3iECxwsxhMPbvSTALBgkqhkiG9w0BAQswHTEO\n
MAwGA1UEChMFRmx5bm4xCzAJBgNVBAsTAkNBMB4XDTE1MDIxNjIxMTkwMVoXDTI
w\nMDIxNTIxMTkwMVowLTEOMAwGA1UEChMFRmx5bm4xGzAZBgNVBAMTEmRldi5s
b2Nh\nbGZseW5uLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBA
MZc97RD\nC66GKEpKRmtCazziQl+M4M8P0OfWyN73IANpCiu6pbgdXuCVsQFjC/kLu
WwINxp9\ntvXvjWk5UYyU5EX0ZMUu6uaN7DQ8tIUPV89lpFx1bIbH+vtx/KvhUtUc0eNT
0leG\n2Jd6PgO468oFYAT8I+QPe3ngpbNvLWofZe59nOTYWvxBfHoE2kacFUCaTYC9fgj
q\nnh6Pn9xcBM3pIJQ4m9kLjW6EPGQrEnFd/ryLo6a8UGh+OdGh/3LuTZBkch5zetbu\na
5bL3MX0WEkDoRdRXOI7+nh8d0LIcvaZqD4kiN2YMmtU87J0N47pHu4YyoArwILA\nTrK
N7+gjX8EvWwkCAwEAAaNqMGgwDgYDVR0PAQH/BAQDAgCgMBMGA1UdJQQMMA
oG\nCCsGAQUFBwMBMAwGA1UdEwEB/wQCMAAwMwYDVR0RBCwwKoISZGV2LmxvY
2FsZmx5\nbm4uY29tghQqLmRldi5sb2NhbGZseW5uLmNvbTALBgkqhkiG9w0BAQsDgg
EBACtZ\ndCxpaBJXHlRHz8co8LReHeyCez1GKlqD1bCD1qYxToEZPuWmi39xTtdCG2ZO
A7SO\nhVZbG4vT4Ra5ANXWtoYco71UZeNn/viJXfF5FxKH8u1rNAsTH8uLdi4PO4Shy4/
d\n2DBueWG1E4BB6LKRfkxSxqEaFeLuV8Z+IEfPaibvEG8G2cvMtK9frCjSSn+iPDtC\neS

```
WnOvgoWmI8GEj9tdFpxwVjDuzK1wcjxkOprjigMV1Oh19D4u+rnjsZKzFNpgfK\nI768GF
sL0eA42rDjyN6Q20oZ++4AJPCN7dHBszRZM1G5bfQE9e6C5JIYgbMEBDLJ\nbNGj5eq9
CKY9WAVz6bw=\n-----END CERTIFICATE-----\n",
    "tls_key": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEpAIBAAKCAQEAxlz3tEMLroYoSkpGa0JrPOJCX4zgzw/Q59bI3vcgA2kKK7ql\nuB1e
4JWxAWML+Qu5bAg3Gn229e+NaTlRjJTkRfRkxS7q5o3sNDy0hQ9Xz2WkXHVs\nhsf6+
3H8q+FS1RzR41PSV4bYl3o+A7jrygVgBPwj5A97eeCls28tah9l7n2c5Nha\n/EF8egTaRpw
VQJpNgL1+COqeHo+f3FwEzekglDib2QuNboQ8ZCsScV3+vIujprxQ\naH450aH/cu5Nk
GRyHnN61u5rlsvcxfRYSQOhF1Fc4jv6eHx3Qshy9pmoPiSI3Zgy\na1TzsnQ3juke7hjKgCv
AgsBOso3v6CNfwS9bCQIDAQABAoIBAEvPhh/Pq+xEe6uA\n3Y7qsH9xXbqU5epc+hNR
BC4qtoJ3J5r015cZKoSc0SxdhVrmPzRpo12thDpUFnCL\npn4a/W866zRtLPt2bDq+pKNh
7MZn6zwm6JZOYb6tnsq17+lg/VprU6197NNftcnI\nZ90q1rqw/qqPvDolQE7mg63WWC
egD724BhVRBo8FUmcrn91uqt2plfCMnnzZPTbL\nB2Hk575ww+vjAysdihavJAeWtAVO7
BpAFcQoDXjbFZXUxWaZbFd1r15PE55uJ/uh\nx0VdQ9GMGahF+yRyOqaKRnovEp0twKj
e0BgGFNwloU90VBL2HipuWvK0tM+pImCf\nX5rHcF0CgYEA09wbQZe7EIt2uWSyM5P0
aMtrReWXHtSJaXUU5u4e1cF/J9KeeZfS\nMNXzIAcsJdZfF2jCyrQs5s+gE5d7yRWHl3L+q
ng18/yiLX9JXSVGtxiLIFdrFpqd\nGrgxMwyf8eGAUdFJnBE1nu4CKytVFTksMdHz0lZTDge
1xyQ8zCvyqrMCgYEA77D+\n4dbX7GV8R1P6Lu7IAeACPuccHEPyGO0lrCiUZtrvzbyKa7n
TI1ELeBCHzty4akGX\nR6lnWpUW0X0LZcsZAQzMfVYezwPwpgIJqtf529TEjEfvUsc9aYOP
RvtoDtnSZ4dz\nz9ChbsRG5tt+Abipva80XqHYv+G4ZyKMNlOzcVMCgYB8/PX3lJSABkdc
yNN5Kmif\nCqOW8QRe4/TNio1yUDlQ0n8590AXbEJNHWv6EUEycW5YWS3SeMmkwg
kmhdA5qcH8\nz5L+7zylXP4w6U9W7I7zpj8uLWS4SS82fERKYcBa6zbRDi/nKx//S1tsLHx
dsQUp\nvzuhc9w8AsBzZCWaLrNTkQKBgQC6hGry5rJDnH1nK+wFVX6CJwTYWhM/04e
y6edk\n3ncMytXMrSixx3qLgud/2K01xgAjf0LvWt24WUfWYI3b+n1Mmd8OsVNsGE7Nf
8xy\nC3q9HAaOevO64JvjXaLCw6Qn2kymj3sJffoBLe9WefrF8CL2Hj6MXTlDp6xNZU7i\nf
6OlTQKBgQCYa/mTIT4NZuyoQitUIl8GyqeR3GHyaSfxCaPlynUG3iHdlr2NOfL1\nGlx5tG
wcawJ1DUe/eKzA3Cj7IM+UcDTdIJM2t3dbQOPh/qEPY54NKTh5yHZ+qeuO\nrpSlpImS
AUx46q7rnnm8KlEYKg6xVEwACHJFk55WxilDmee+/ZACag==\n-----END RSA PRIVATE
KEY-----\n"
  },
  {
    "type": "tcp",
    "id": "733348c4-f2e0-490f-bd90-67c9551aa3a6",
    "parent_ref": "controller/apps/a1f78fa3a47243cea6a0b4a4e0b40beb",
    "service": "gitreceive",
    "created_at": "2015-02-16T21:19:01.521562Z",
    "updated_at": "2015-02-16T21:19:01.521562Z",
    "port": 2222
  }
]
```

删除路由

实例：

——————————————————————请求——————————————————————

```
DELETE /routes/http/e8080915-1d48-4374-ac4f-c54a88bdd210 HTTP/1.1
Content-Type: application/json
```

——————————————————————响应——————————————————————

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

错误

https://flynn.io/schema/router/error#(https://flynn.io/schema/router/error#)

属性	类型	描述
code	string	未发现、未发现对象、对象已存在、语法错误、校验错误、未知错误
message	string	
detail	object	

商标准则

商标指南

Prime Directive公司（简称“Flynn”）商标及 logo（“our Marks”）是 Flynn 所保护的资产。为帮助您正确使用我们的商标，我们发布了这些商标的使用指南（“Guidelines”）。我们商标的保护强度是部分取决于商标使用的一致性和适当性。我们希望您按照这些准则正确使用并信任我们的商标。我们保留随时自主更改这些准则权利。

企业标志

我们的企业标志包括注册的、注册中的商标、logo：

Flynn，Flynn logo，商标和(或)在特定许可协议或其他协议中提到的其他商标。

如有变更，恕不另行通知。

禁止创建复合商标

我们的标记不能作为任何其他标记的一部分（无论词语或徽标）。未经事先书面批准，不得以任何形式用我们的商标为您的产品做设计、包装、网页或者是宣传品。如果您的商标与我们的商标用在

一起，使用时必须区分两个商标。若因商标的变化导致消费者对商品或服务来源混淆，这种使用若侵犯我们的商标权，我们会根据适用的法律采取行动。

禁止商标用途

Flynn不允许以如下方式使用我们的任何商标：作为自己商标的一部分;标识非 Flynn的 产品、服务或技术;容易造成 Flynn 的产品、服务或技术与其他实体的混淆;在您的活动、产品、服务或技术上假借 Flynn 赞助、认可或者其他名号;以任何方式诋毁 Flynn 或者其产品、服务或技术;经判断，可能会损害商标商誉的相关产品、服务或者活动;非法活动。

网站和域名用途

允许在网站上正确使用我们的商标来命名或准确地描述 Flynn的 产品、服务或技术。使用我们的商标不能造成误导，该网站是由 Flynn 赞助、隶属于 Flynn 或者该产品、服务或技术是否由 Flynn 提供。任何主体或二级域名不能与我们的任何商标相同或类似。网站拥有者不能含有我们注册商标的任何域名，并且不能在域名中索取商标或类似的专有权利。网站应显示法律声明或包含以下文字说明的链接：

“在美国和(或)其他国家，Flynn和Flynn标志是Prime Directive公司的商标或者注册商标，Prime Directive公司和其他当事人拥有本文中使用的其它术语的商标所有权。”

此说明可以修改为仅包含我们的商标。网站也应该在其他方面遵守域名注册政策和有关商标侵权和调整适用的法律。

请求权限

您可以合理地用我们的商标来命名或引用我们的产品、服务或技术，例如，未经我们许可，只要您遵循这些原则即可在一篇杂志文章中引用我们的商标。除非得到明确的书面许可，否则所有其他用途都是被禁止的，您需要事先提出请求，然后根据我们的判断授予或拒绝。若想请求权限，请通过电子邮件legal@flynn.io联系Flynn的法律部门，Flynn将尽快评估请求。Flynn保留拒绝以任何理由使用我们商标的权利。

使用限制之前需要商标许可

Flynn 代码可在BSD3级许可证上使用。尽管BSD许可证允许第三方根据许可的条款复制和重新发布底层软件，但是BSD许可证不提供任何使用Flynn商标的许可或权利。因此，许可人可根据BSD许可证的条款重新分配 Flynn 软件，但不能在未明确得到Prime Directive公司的书面许可或者本商标政策的明确允许下使用Flynn商标。

Flynn 审议

Flynn 保留商标外用用途的审查权，并保有对其使用进行定期抽查的权利。向Flynn提出请求时，您必须提供所有带Flynn标志的产品、包装、截图或其他材料的复印件。您必须在使用商标时纠正所有可能的缺陷。拒绝纠正这些缺陷可能导致撤销使用该商标的许可。

相关问题

若想得到更多的信息或有关正确使用我们商标的指导，请联系Flynn的法律部门。

基于https://www.docker.com/legal/trademark_guidelines/。经许可使用。

原文地址