Raspberry Pi 与 Docker 构建 Serverless 集群

Phodal Huang

September 8, 2017

目录

原文链接：https://www.wandianshenme.com/play/raspberry-pi-docker-swarm-build-faas-cluster

本博客将向您展示，如何使用 Docker 和 OpenFaaS 框架创建自己的 Serverless 框架的 Raspberry Pi 集群。人们经常问我们，应该怎么做他们的集群，同时这个应用程序是完美的信用卡大小的设备 - 想要更多的计算能力？只需要通过添加更多的 RPi 设置就能扩展。

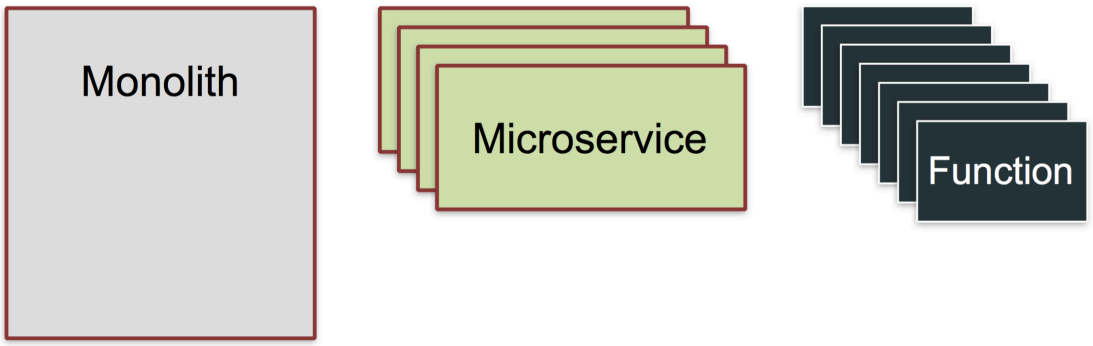
> “Serverless” 是事件驱动架构的设计模式，就像 “桥”，“装饰器”，“工厂” 和 “云” 也是抽象的概念 - 因此它是 “Serverless”。

这是我的博客文章的集群 - 用黄铜支架来分隔每个设备。

> 我在 @docker 和 @Raspberry_Pi 上写博客文章 - 创建集群并部署代码可能并不容易 □。pic.twitter.com/KD2MIIrAx9 — Alex Ellis (@alexellisuk) August 19, 2017

步骤 **0**：什么是 **Serverless**，为什么它对你很重要？

作为一个行业，我们对 “Serverless” 这个术语的意思做了一些解释。为了这个博客文章，让我们假设它是一个用于事件驱动架构的新架构，它允许您以任何您喜欢的语言编写小而可重用的函数。（更多内容可以阅读：Introducing Functions as a Service (FaaS)）



Serverless 函数可以做任何事情，但通常可以在给定的输入中工作 - 例如来自 GitHub，Twitter，PayPal，Slack，Jenkins CI 管道中的事件。或者在 Raspberry Pi 的情况下，可能是真实世界的传感器输入，例如 PIR 运动传感器，激光绊网或甚至温度计。

我们还假定 Serverless 函数，倾向于使用第三方后端服务来使其大于其部件的总和。

步骤 **1**：概要

我们将使用 OpenFaaS，您可以将任何单个主机或群集转换为后端来运行 Serverless 函数。并能使用 Docker 部署任何二进制程序、脚本或编程语言，它们都可以在 OpenFaaS 上运行，您可以在速度和灵活性之间进行选择。还有一个好消息是，它提供了一个 UI，并且内置了度量。

这是我们要做的步骤：

- 在一个或多个 Raspberry Pi 上部署 Docker
- 在 Docker 群中加入他们
- 部署 OpenFaaS
- 使用 Python 编写你的第一个函数

Docker Swarm

Docker 是一种用于打包和部署应用程序的技术，它还具有内置的集群，默认情况下是安全的，只需一行即可。OpenFaaS 使用 Docker 和 Swarm 在您所有可用的 RPi 中传递 Serverless 函数。

PS：我建议使用 Raspberry Pi 2 或 3 用于此项目，它们带有以太网交换机和强大的 USB 多适配器。

步骤 **2**：准备 **Raspbian**

烧录 Raspbian Jessie Lite 到 SD 卡上，8GB 的空间就够了，但是推荐 16GB。

注意：不要下载 Raspbian Stretch

> 社区正在帮助 Docker 团队准备支持 Raspbian Stretch，但还不能无缝的工作。

我建议使用 Etcher.io 来烧录镜像。

> 在启动你的 Raspberry Pi 之前，你需要在你的 SD 卡的 boot 分区，创建一个名为 `ssh` 的空文件。它可以由 Raspbian 系统识别，以允许远程登录。

启动 **Raspberry Pi**，并更改主机名

现在打开 Raspberry Pi，并使用 ssh 连接

```
1 $ ssh pi@raspberrypi.local
```

密码是 raspberry

接着运行 raspi-config 来修改 hostname 为 swarm-1，或者类似的名字，然后重启设备。

在这里，您还可以将 GPU（图形）和系统之间的内存分配更改为 16mb。

安装 **Docker**

我们可以使用一个实用程序脚本：

```
1 $ curl -sSL https://get.docker.com | sh
```

此安装方式，可能会在将来更改。如上所述，您需要运行 Jessie，因此我们有一个已知的配置。

您可能会看到这样的警告，但您可以忽略它，你最终使用的应该是 Docker CE 17.05：

```
1 WARNING: raspbian is no longer updated @ https://get.docker.com/
2 Installing the legacy docker-engine package...
```

之后，确保您的用户帐户可以使用此命令访问 Docker 客户端：

```
1 $ usermod pi -aG docker
```

如果您的用户名不是 pi，那么用你的用户名替换 pi。

更改默认密码

输入 $sudo passwd pi，并输入新密码，请不要跳过这一步！

重复

为每个 RPi，重复上面的每一个步骤。

步骤 **3**：创建您的 **Swarm** 群集

登录第一个 RPi，并键入以下内容：

```
1 $ docker swarm init
2 Swarm initialized: current node (3ra7i5ldijsffjnmubmsfh767) is now a
    manager.
```

```
3
4 To add a worker to this swarm, run the following command:
5
6 docker swarm join \    --token
        SWMTKN-1-496mv9itb7584pzcddzj4zvzzfltgud8k75rvujopw15n3ehzu-af445b08359golnzhncbdj9o3
7     192.168.0.79:2377
```

您将看到带有您的连接令牌的输入，在其他 RPi 中输入这些内容。因此使用 ssh 登录到每个 Raspberry Pi，并粘贴在命令中。

给这几秒钟让他们连接，然后在第一个 RPi 中检查您的所有节点列出：

```
1 $ docker node ls
2 ID                            HOSTNAME            STATUS
    AVAILABILITY        MANAGER STATUS
3 3ra7i5ldijsffjnmubmsfh767 *   swarm1              Ready
    Active              Leader
4 k9mom28s2kqxocfq1fo6ywu63     swarm3              Ready               Active
5 y2p089bs174vmrlx30gc77h4o     swarm4              Ready               Activ
```

恭喜！你有一个 Raspberry Pi 集群！

- 更多集群

您可以看到我的三台主机正在运行。其中，只有一个是管理机（manager）。如果我们的管理机要下台，那么我们将处于不可恢复的状态。这样做的方法是，通过向管理者提供更多的节点来增加冗余 - 除非您专门设置服务，否则他们仍然会运行工作负载。

要将一个进程机（worker）升级到管理机（manager），只需要在你的一个管理机上执行：`docker node promote <node_name>`

> 注意：群集（Swarm）命令，如 docker 服务 ls 或 docker node ls 只能在管理器上完成。

深入理解 Swarm 的更多管理信息，请查阅官方的文档：[Docker Swarm admin guide](https://docs.docker.com/engine/swarm/admin_guide/)

步骤 4：OpenFaaS

现在，让我们继续部署一个真正的应用程序，使 Serverless 函数能在在我们的集群上运行。OpenFaaS 是 Docker 下的一个框架，可以让任何进程或容器成为 Serverless 函数 - 大规模和任何硬件或云端。由于 Docker 和 Golang 的便携性，它在 Raspberry Pi 上运行得很好。

登录第一个 RPi（我们运行 `docker swarm init` 的机器）并克隆/部署项目：

```
$ git clone https://github.com/alexellis/faas/
$ cd faas
$ ./deploy_stack.armhf.sh
Creating network func_functions
Creating service func_gateway
Creating service func_prometheus
Creating service func_alertmanager
Creating service func_nodeinfo
Creating service func_markdown
Creating service func_wordcount
Creating service func_echoit
```

Docker Swarm 现在将要求您的其他 RPi，从互联网上拉出 Docker 镜像并将其解压缩到 SD 卡。这项工作将分布在所有的 RPIS 中，以免他们都不会过度劳累。

这可能需要几分钟的时间，因此您可以通过输入下面的内容，来查看进度：

```
$ watch 'docker service ls'
ID                  NAME                MODE                REPLICAS
                IMAGE                                       PORTS
57ine9c10xhp        func_wordcount      replicated          1/1
                    functions/alpine:latest-armhf
d979zipx1gld        func_prometheus     replicated          1/1
                    alexellis2/prometheus-armhf:1.5.2       *:9090->9090/tcp
f9yvm0dddn47        func_echoit         replicated          1/1
                    functions/alpine:latest-armhf
lhbk1fc2lobq        func_markdown       replicated          1/1
                    functions/markdownrender:latest-armhf
pj814yluzyyo        func_alertmanager   replicated          1/1
                    alexellis2/alertmanager-armhf:0.5.1     *:9093->9093/tcp
```

```
8 q4bet4xs10pk        func_gateway        replicated          1/1
                      functions/gateway-armhf:0.6.0           *:8080->8080/tcp
9 v9vsvx73pszz        func_nodeinfo       replicated          1/1
                      functions/nodeinfo:latest-armhf
```

我们希望看到我们所有服务中列出的 **1/1**。

给定任何服务名称，您可以键入以下内容，来查看哪个 RPi 将安装它：

```
1 $ docker service ps func_markdown
2 ID                  IMAGE                                   NODE    STATE
3 func_markdown.1     functions/markdownrender:latest-armhf   swarm4  Running
```

其状态应该是 **Running**，如果是 **Pending** 则说明镜像仍然可以从互联网上下来。

此时，请查找您的 RPi 的 IP 地址，并在 Web 浏览器中打开该 IP 地址的 8080 端口：

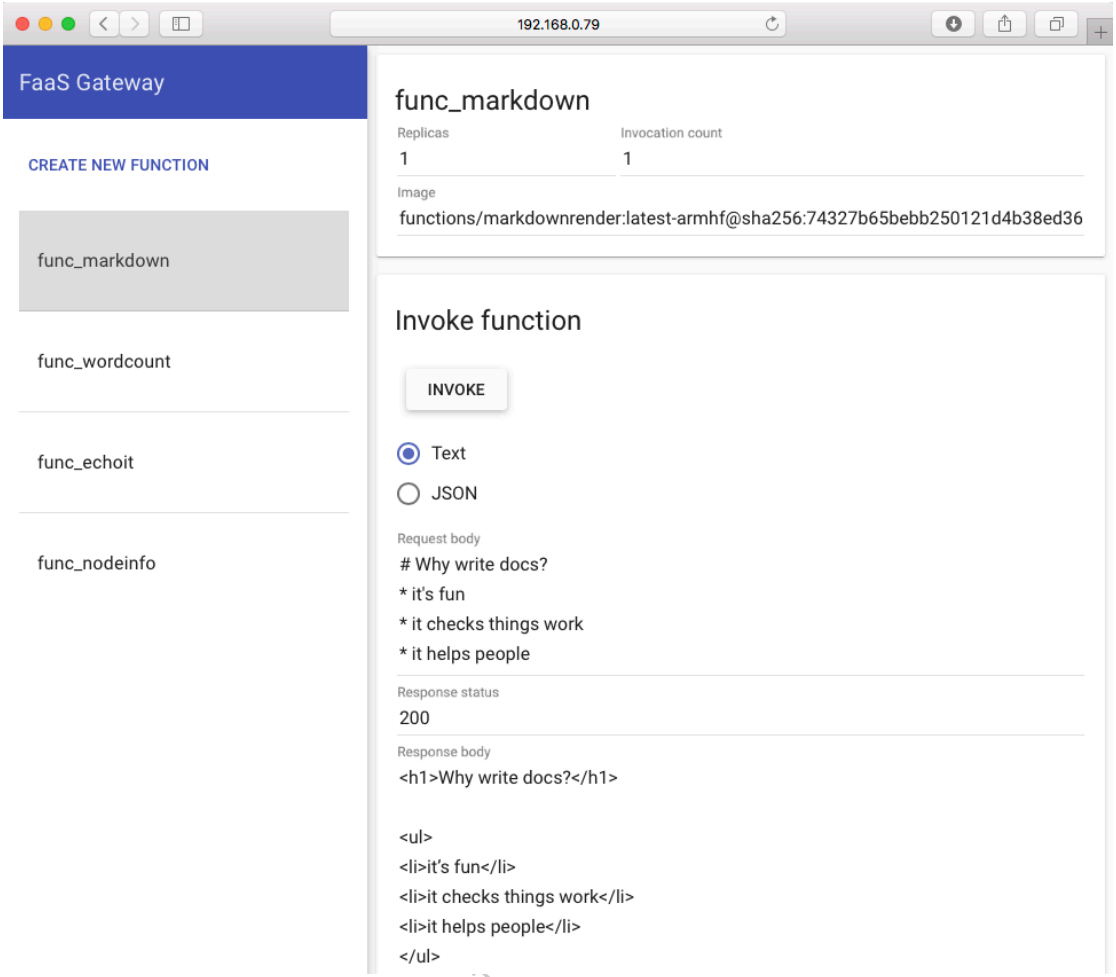
```
1 $ ifconfig
```

比如你的 IP 是 `192.168.0.100`，那么就打开 http://192.168.0.100:8080

此时，您应该会看到 FaaS 的 UI 界面，也称为 API 网关。这是您可以定义、测试和调用您的函数的地方。

点击名为 `func_markdown` 的 Markdown 转换函数，然后输入一些 Markdown 文本。

然后点击 invoke。您将看到 invoke 计数上升，屏幕的下半部分显示您的功能的结果，如下图所示：



步骤 **5**：部署您的第一个 **Serverless** 函数

已经有了这个部分的教程，但是我们需要首先使用几个自定义步骤来设置 RPi。

1. 获取 FaaS-CLI

```
$ curl -sSL cli.openfaas.com | sudo sh
armv7l
Getting package
    https://github.com/alexellis/faas-cli/releases/download/0.4.5-b/faas-cli-armhf
```

2.Clone 示例：

```
$ git clone https://github.com/alexellis/faas-cli
$ cd faas-cli
```

3. 为 Raspberry Pi 打补丁

我们将暂时更新我们的模板，以便他们能在 Raspberry Pi 上工作：

```
$ cp template/node-armhf/Dockerfile template/node/
$ cp template/python-armhf/Dockerfile template/python/
```

这样做的原因是，Raspberry Pi 与我们每天互动的大多数计算机具有不同的处理器。

现在，您可以按照下面的 PC，笔记本电脑和云端编写的相同教程，但是我们将首先为 Raspberry Pi 运行几个命令。

Your first serverless Python function with OpenFaaS

在教程中步骤 3 的时候：

- 在 `~/functions/hello-python` 替换成你的函数，并将它们放在我们刚从 GitHub 克隆的 faas-cli 文件夹中。
- 然后在 `stack.yml` 中将 localhost 替换成你的第一个 Raspberry Pi 的 IP 地址

请注意，Raspberry Pi 可能需要几分钟才能将 Serverless 函数下载到相关的 RPi 中。您可以检查您的服务，以确保您使用此命令显示 1/1 副本：

```
$ watch 'docker service ls'
pv27thj5lftz        hello-python        replicated          1/1
                    alexellis2/faas-hello-python-armhf:latest
```

相关阅读资料：Your first serverless Python function with OpenFaaS

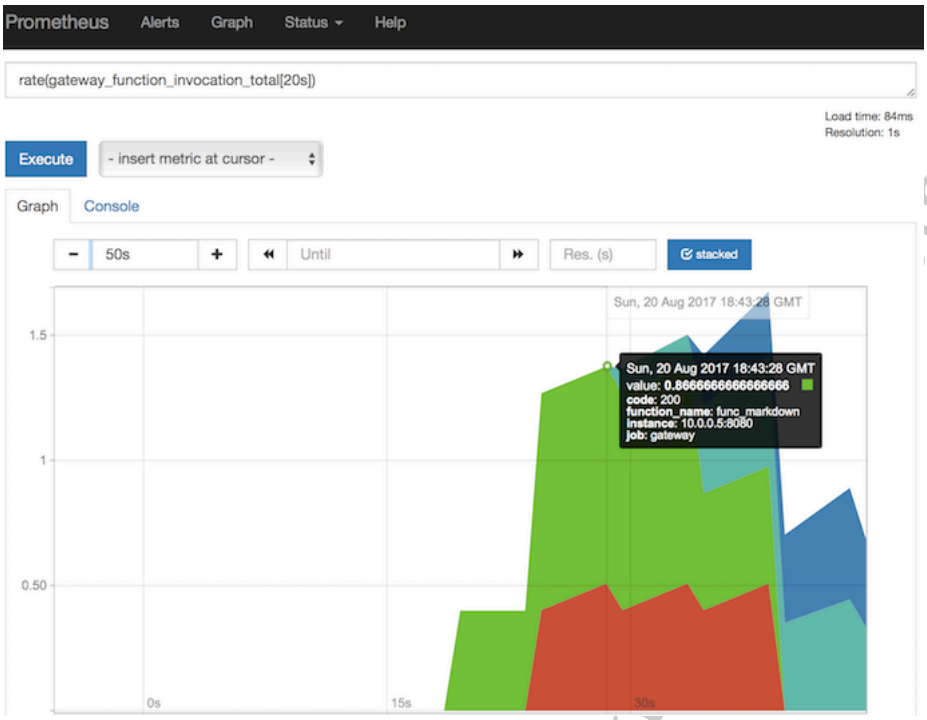
步骤 6：检查您的功能指标

通过 Serverless 的体验，您不需要花费所有时间来管理您的函数。幸运的是，Prometheus 指标内置于 OpenFaaS 中，这意味着您可以跟踪每个功能需要多长时间运行以及调用频率。

指标驱动自伸缩

如果您在任何功能上产生足够的负载时，OpenFaaS 将自动调整您的函数，当需求消失时，您将再次返回到单个副本。

以下是可以粘贴到 Safari，Chrome 等的示例查询：



PS：只需将 IP 地址更改为您自己的 IP 地址。

```
1 http://192.168.0.25:9090/graph?g0.range_input=15m&g0.stacked=1&g0.expr=rate(gateway_function_
```

这个请求是用 PromQL 编写的。PromQL 是 Prometheus 的查询语言。第一个查询，显示了函数调用的频率：

```
1 rate(gateway_function_invocation_total[20s])
```

第二个查询显示了我们对每个函数有多少副本，开始时应该只有一个：

```
1 gateway_service_count
```

如果要触发自动缩放，可以在 RPi 上尝试以下操作：

```
1 $ while [ true ]; do curl -4 localhost:8080/function/func_echoit --data
    "hello world" ; done
```

检查 Prometheus 的『alert』页面，看看是否生成足够的负载来触发自动缩放，如果您不是在几个附加的终端窗口中运行该命令。

减少负载后，第二个图表中显示的副本计数和 gateway_service_count 度量将再次返回 1。

步骤 7：**Done**

我们现在已经设置了 Docker、Swarm 并运行 OpenFaaS，让我们将 Raspberry Pi 像一台巨型计算机一样处理，并准备好通过代码。

原文链接：https://blog.alexellis.io/your-serverless-raspberry-pi-cluster/

原文链接：https://www.wandianshenme.com/play/raspberry-pi-docker-swarm-build-faas-cluster