Docker入门教程（四）Docker Registry

【编者的话】DockerOne组织翻译了Flux7的Docker入门教程，本文是系列入门教程的第四篇，介绍了Docker Registry，它是Docker中的重要组件。本文通过情景演绎的方式对其进行了介绍，图文并茂，强烈推荐读者阅读。

在Docker系列教程的上一篇文章中，我们讨论了Dockerfile的重要性并提供了一系列Dockerfile的命令，使镜像的自动构建更加容易。在这篇文章中，我们将介绍Docker的一个重要组件：Docker Registry。它是所有仓库（包括共有和私有）以及工作流的中央Registry。在深入Docker Registry之前，让我们先去看看一些常见的术语和与仓库相关的概念。

1. Repositories（仓库）可以被标记为喜欢或者像书签一样标记起来
2. 用户可以在仓库下评论。
3. 私有仓库和共有仓库类似，不同之处在于前者不会在搜索结果中显示，也没有访问它的权限。只有用户设置为合作者才能访问私有仓库。
4. 成功推送之后配置webhooks。

Docker Registry有三个角色，分别是index、registry和registry client。

角色 1 -- Index

`index` 负责并维护有关用户帐户、镜像的校验以及公共命名空间的信息。它使用以下组件维护这些信息：

- Web UI
- 元数据存储
- 认证服务
- 符号化

这也分解了较长的URL，以方便使用和验证用户存储库。

角色 2 --Registry

`registry`是镜像和图表的仓库。然而，它没有一个本地数据库，也不提供用户的身份认证，由S3、云文件和本地文件系统提供数据库支持。此外，通过Index Auth service的Token方式进行身份认证。Registries可以有不同的类型。现在让我们来分析其中的几种类型：

1. Sponsor Registry：第三方的registry，供客户和Docker社区使用。
2. Mirror Registry：第三方的registry，只让客户使用。
3. Vendor Registry：由发布Docker镜像的供应商提供的registry。
4. Private Registry：通过设有防火墙和额外的安全层的私有实体提供的registry。

角色 3 --Registry Client

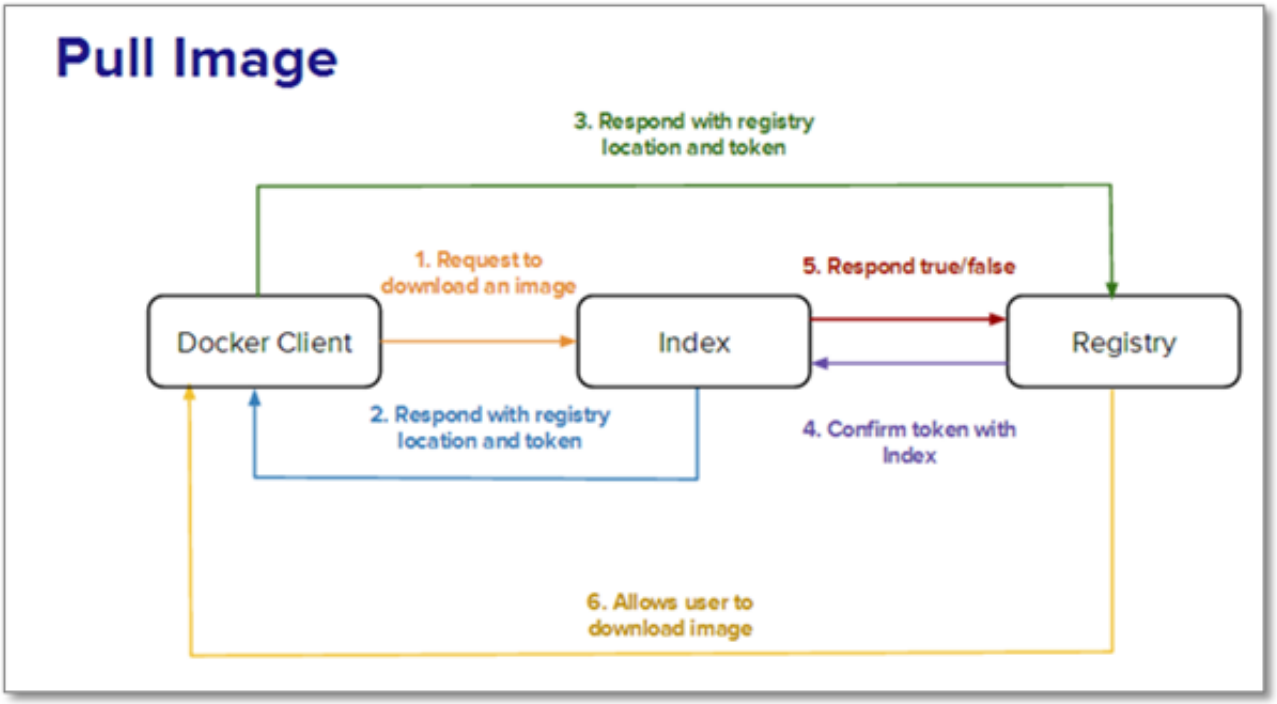
Docker充当registry客户端来负责维护推送和拉取的任务，以及客户端的授权。

Docker Registry工作流程详解

现在，让我们讨论五种情景模式，以便更好地理解Docker Registry。

情景A：用户要获取并下载镜像。所涉及的步骤如下：

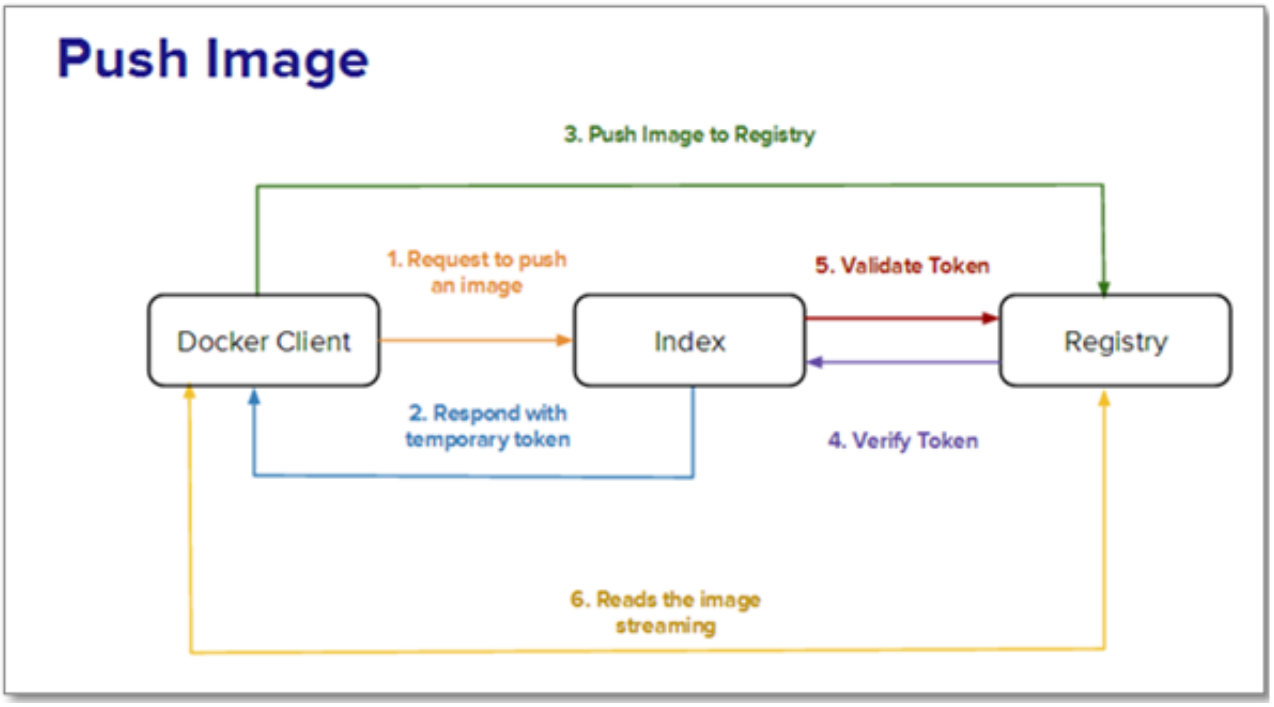
1. 用户发送请求到index来下载镜像。
2. index 发出响应，返回三个相关部分信息：
   - 该镜像所处的registry
   - 该镜像包括所有层的校验
   - 以授权为目的的Token > 注意：当请求header里有X-Docker-Token时才会返回Token。而私人仓库需要基本的身份验证，对于公有仓库这一点不是强制性的。
3. 用户通过响应后返回的Token和registry沟通，registry全权负责镜像，它用来存储基本的镜像和继承的层。
4. registry现在要与index证实该token是被授权的。
5. index会发送“true” 或者 “false”给registry，由此判定是否允许用户下载所需要的镜像。



情景B：用户想要将镜像推送到registry中。其中涉及的步骤如下：

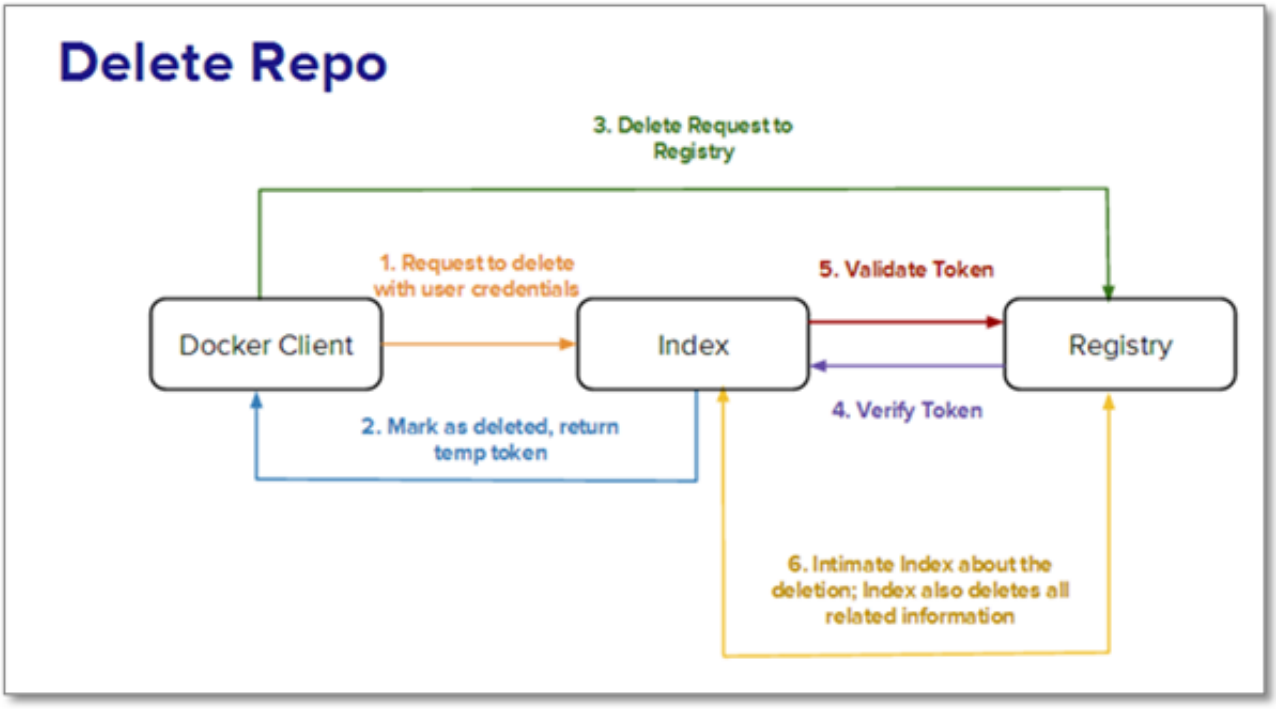
1. 用户发送附带证书的请求到index要求分配库名。
2. 在认证成功，命名空间可用之后，库名也被分配。index发出响应返回临时的token。
3. 镜像连带token，一起被推送到registry中。
4. registry与index证实token被授权，然后在index验证之后开始读取推送流。

5. 该index由Docker校验的镜像更新。



情景C：用户想要从index或registry中删除镜像：

1. index接收来自Docker一个删除库的信号。
2. 如果index对库验证成功，它将删除该库，并返回一个临时的token。
3. registry现在接收到带有该token的删除信号。
4. registry与index核实该token，然后删除库以及所有与其相关的信息。
5. Docker现在通知有关删除的index，然后index移除库的所有记录。



情景D：用户希望在没有index的独立模式中使用registry。
使用没有index的registry，这完全由Docker控制，它最适合于在私有网络中存储镜像。registry运行在一个特殊的模式里，此模式限制了registry与Docker index的通信。所有有关安全性和身份验证的信息需要用户自己注意。

情景E：用户想要在有index的独立模式中使用registry。
在这种情况下，一个自定义的index会被创建在私有网络里来存储和访问镜像的问题。然而，通知Docker有关定制的index是耗时的。 Docker提供一个有趣的概念chaining registries，从而，实现负载均衡和为具体请求而指定的registry分配。在接下来的Docker教程系列中，我们将讨论如何在上述每个情景中使用Docker Registry API ，以及深入了解Docker Security。

原文链接：Part 4: Registry & Workflows（翻译：田浩浩 审校：李颖杰）

= =

译者介绍

田浩浩，悉尼大学USYD硕士研究生，目前在珠海从事Android应用开发工作。业余时间专注Docker的学习与研究，希望通过DockerOne把最新最优秀的译文贡献给大家，与读者一起畅游Docker的海洋。
