链滴

docker-compose: 部署 Nexus3

作者：liumapp
原文链接：https://ld246.com/article/1528787422835
来源网站：链滴

<blockquote>
利用 docker-compose 在 docker 下部署 Maven 私服 Nexus
</blockquote>
<h2 id="前言">前言</h2>
先上项目源代码 liumapp/nexus-in-docke 和用于测试发布的 liumapp/convert html-to-pdf
nexus-in-docker 项目用于在系统的 Docker 上部署 Nexus3，convert-html-to-pdf 用于测试布到该 Nexus 私服上。
<h2 id="Nexus安装运行">Nexus 安装运行</h2>
使用命令
<pre><code class="highlight-chroma">docker-compose up -d
</code></pre>
会自动去 docker hub 拉取 Nexus:3.12.1 版本的镜像并生成容器运行。
不过因为设置了 volumes 与容器内的/nexus-data 目录建立关联，所以在运行之前，需要确保 nxus-in-docker 的根目录具备写权限。
运行成功后（不会有提示信息，可以使用 <code>docker logs -t -f --tail 100 nexus</code> 查看启动日志），在浏览器内访问 http://localhost:8081 便可以打开 Nexus 的管理 web 页面。</p

初始的管理员帐号密码为 admin/admin123。
<h2 id="Deploy项目">Deploy 项目</h2>
<h3 id="创建Repository">创建 Repository</h3>
使用 admin 登陆后，我们可以在设置一栏里创建一个 Repository，需要注意的地方只有三个：

Recipe 要选择 Maven2(hosted)。
可能有小伙伴会问，为什么不是 Maven2(group)或者 Maven2(proxy)呢？
首先要搞清楚 proxy,hosted,group 三者的关系：

<code>proxy</code> 远程仓库的代理，比如说 <code>nexus</code> 配置了一个 <code>entral repository</code> 的 <code>proxy</code>,当用户向这个 <code>proxy</code> 请求个 <code>artifact</code> 的时候，会现在本地查找，如果找不到，则会从远程仓库下载，然后返给用户。

<code>hosted</code> 宿主仓库，用户可以把自己的一些仓库 <code>deploy</code> 到个仓库中

<code>group</code> 仓库组，是 nexus 特有的概念，目的是将多个仓库整合，对用户暴露一的地址，这样就不需要配置多个仓库地址。

所以我们的项目是要发布到本地的 Nexus 私服，自然就要选择 hosted。

Version policy 要选择 Mixed
因为我们的 Maven 项目在打版本的时候，有时候是 Release 版本，有时候就是一个 v1.0.0 版，像我就喜欢用后者，选择 Mixed 可以让私服支持不同类型的项目版本，如果您选择的是 Release

者 Snapshot，那么 deploy 过来的项目就必须是这两种类型的版本，不然就会报错。

Deployment policy 要选择 Allow redeploy
一个项目总不可能不更新迭代了吧，除非已经放弃治疗删库跑路了。

<h3 id="添加Nexus用户">添加 Nexus 用户</h3>
添加一个 Nexus 用户，用于后面的 Deploy，只需要注意两点：

User 的 ID 将会在后面的 Maven 配置项中作为 username 来使用，所以很多情况都会发现 id 跟 username 相同的情况。

User 的 status 请注意设置为 active，虽然这是一个显而易见的事情，但还是会存在很多粗心大的情况。

<h3 id="本地Deploy">本地 Deploy</h3>
接下来轮到我们的另一个测试项目 liumapp/convert-html-to-pdf 上场。
首先修改本地 maven 的配置文件 settings.xml，把新添加的 maven 私服用户写入 server 下。
<pre><code class="highlight-chroma"><servers>
  <server>
    <id>liumap</id>
    <username&t;liumapp</username>
    <password&t;liumapp</password>
  </server>
</servers>
</code></pre>
在要发布到该私服下的 maven 项目中，添加：
<pre><code class="highlight-chroma"><distributionManagement>
    <repository
gt;
        <id>liuapp</id>
        <url>http://127.0.0.1:8081/repository/liumapp/</url>
    </repository
gt;
</distributionMnagement>
</code></pre>
这里 repository/liumapp 的 liumapp 代表您刚刚创建的 repository 名称，并不是用户名。</
>
然后在 build 下添加以下插件

<pre><code class="highlight-chroma"> <!--发布代码Jar插件-->
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-deploy-plugin</artifactId>
    <version>2.7</version>
  </plugin>
  <!--发布源码件-->
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-source-plugin</artifactId>
    <version>2.2.1</version>
    <execution>
      <execution>
        <phase>package</phase>
        <goal>
          <goal>jar</goal>
        </goal>
      </execution>
    </execution>
  </plugin>
</code></pre>
最后在 console 下输入命令
<pre><code class="highlight-chroma">mvn deploy
</code></pre>
具体的配置可以直接在 convert-html-to-pdf 下查看 pom.xml 文件。
如果您要发布的是一个 Jar 包，那么直接使用命令：
<pre><code class="highlight-chroma">mvn deploy:deploy-file -DgroupId=com.aspose.words -DartifactId=aspose-words -Dversion=15.8.0 -Dpackaging=jar -Dfile=./aspose-words-15.8.0-jdk16.jar -Durl=http://127.0.0.1:8081/repository/liumapp/ -DrepositoryId=liumapp
</code></pre>
即可。
<h2 id="使用私服">使用私服</h2>
在要从该私服下载依赖的项目中，配置 pom.xml 文件：
<pre><code class="highlight-chroma"><repositories>
  <repository&g

;
    <id>test&l
;/id>
    <url>http:
/127.0.0.1:8081/repository/liumapp/</url>
  </repository&
t;
</repositories
gt;
</code></pre>
即可