 **IBM Bluemix** 点击按钮，开始云上的开发！

开始您的试用

developerWorks 中国 技术主题 Cloud computing 文档库

使用 IBM DevOps Services for Bluemix 自动化滚动部署

2015 年 2 月 05 日



Daniel Berg

IBM 杰出工程师，CTO
DevOps 工具和战略
在 [G+](#) 上关注我

吸引新用户并留住现有用户的最佳方式是，创建从不中断的富有吸引力的应用程序。用户会对宕机（无论是否是计划性的）的站点具有很低的容忍度。而且随着团队交付变更的速度增加，避免服务中断变得越来越重要。

应用程序的新版本可通过滚动部署 流程部署到环境中，无需为用户中断服务。如果想要扩大用户群，则必须 实现滚动部署。

[IBM® Bluemix™](#) 提供了一个容易使用的平台来实现滚动部署。我将介绍如何使用 IBM DevOps Services 和 [交付管道服务](#) 轻松地自动化向 Bluemix 空间的滚动部署。



在 **IBM Bluemix** 云平台上开发并部署您的下一个应用。

开始您的试用

实现您的应用程序的前提条件

[Bluemix](#) 和 [IBM DevOps Services](#) 帐户。

对 IBM 交付管道功能拥有基本的理解，比如如何创建一个默认 Builder 和 Deployer。如果有必要的话，请查阅 [通过 Java 开始使用 IBM Bluemix 和 DevOps Services](#) 教程。

Bash 脚本知识。

查阅管道

获取示例代码

滚动部署是任何生产级应用程序避免服务中断的必备要素。

阅读： [通过 Java 开始使用 IBM Bluemix 和 DevOps Services](#)

观看： [BlueGreenDeployment](#)

我为一个简单的 Hello World 示例设置了一个自动化的滚动部署管道。在阅读过程中可以查阅该代码。该管道可公开使用，您可以查阅部署自动化配置和部署日

志。

我将演示在 DevOps Services 中建立自动化的滚动部署的步骤。这些步骤仅用作示例，您可以调整这些步骤和脚本，使之满足您的要求，或者在我的基本 **bash** 脚本能力基础上对它们进行改进。

滚动部署通常被称为蓝-绿部署、红-黑部署或零宕机时间部署。尽管这些部署流程存在细微的差别，但它们共同的核心原则都是为用户避免服务中断。

第 1 步：分解示例项目

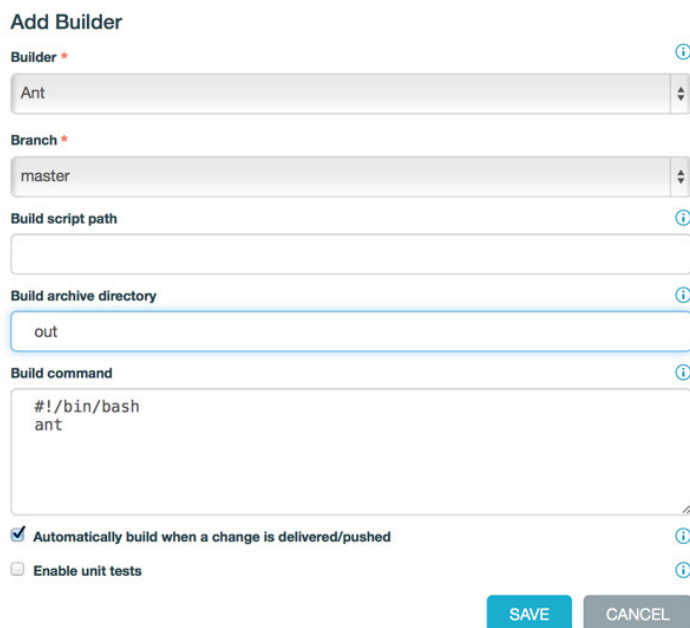
首先分解示例项目，让您拥有自己的具有必要源代码的 DevOps Services 项目：

1. 向上滚动并单击本教程的获取示例代码按钮。
2. 在 danberg | Pipeline_hello 项目的概述页面中，单击 **EDIT CODE** 按钮（如果尚未登录，请输入您的 DevOps Services 凭据）并单击菜单上的 **FORK** 按钮。
3. 为新项目输入一个名称，选择 **Deploy to Bluemix** 复选框，并选择一个组织和空间用于结算用途。
4. 单击 **Save** 使用该源代码创建新项目。

第 2 步：添加一个新构建阶段

项目创建后，向交付管道添加一个新构建阶段，以便开始设置该管道：

1. 单击页面顶部的 **BUILD & DEPLOY** 按钮。
2. 默认情况下没有启用交付流程。单击 **ADVANCED** 按钮打开高级交付管道特性。
3. 单击 **add a builder** 按钮为项目配置一个构建器。保留默认设置（Ant 构建器）并在 **Build archive directory** 文本域中输入 **out**：



[点击查看大图](#)

4. 单击 **SAVE** 保存构建器设置。
5. 单击 **add a stage** 按钮配置一个部署器。单击 **SAVE** 保存默认设置。我们将


在后续步骤中针对滚动发布来调整部署器。

第 3 步：调整交付管道以使用部署脚本

因为滚动部署比其他部署类型稍微复杂一点，所以最好在 **SCM** 内管理部署脚本。您需要调整构建器和部署器来使用部署脚本：

1. 单击您项目的 **EDIT CODE** 按钮。
2. 选择 **build.xml** 文件，您可以注意到，它包含以下用于将 **deploy.sh** 文件（用于滚动部署的脚本）复制到归档目录的代码段。需要使用以下代码才能确保 **deploy.sh** 文件在部署阶段可用：

```
1 <echo message="Copy files to ${artifact_dir} for deployment"/>
2 <copy todir="${artifact_dir}">
3   <fileset file="manifest.yml" />
4   <fileset file="deploy.sh" />
5 </copy>
```

3. 单击 **BUILD & DEPLOY** 按钮打开该管道。
4. 单击 **Deployer** 右上角的配置图标来编辑配置。
5. 将 **Application name** 设置为一个对您的项目有意义的值。请注意，此名称将用于在部署过程中计算一个新应用程序名称。
6. 将 **Script** 节的内容替换为：

```
1 # Use a script from the build result
2 source deploy.sh
```

使用脚本正文中的 **source**，这是调用您的构建工件中包含的 **bash** 脚本的正确方式：

Script ⓘ

```
# Use a script from the build result.
source deploy.sh
```

SAVE CANCEL DELETE

[点击查看大图](#)

7. 单击 **SAVE**。

第 4 步：检查滚动部署脚本

请查阅示例 **deploy.sh** 脚本文件中，使用 **bash** 脚本自动化滚动部署所需的主要代码段。（在这一步不需要执行任何更改。）

1. 在脚本顶部，定义要在以后使用的变量：

```

1 #####
2 # Colors #
3 #####
4 green='\e[0;32m'
5 red='\e[0;31m'
6 label_color='\e[0;33m'
7 no_color='\e[0m'
8
9 # Used to create a public route within the space for accessing the deployed
10 # application. The reserved CF_SPACE variable (configured in the deployer) is
11 # used to unique the route name per space.
12 if [ "${CF_SPACE}" == "prod" ]; then
13     PUBLIC_HOST=${CF_APP}
14 else
15     PUBLIC_HOST="${CF_SPACE}-${CF_APP}"
16 fi
17
18 # Extract the selected build number from the reserved BUILD_SELECTOR variable.
19 SELECTED_BUILD=$(grep -Eo '[0-9]{1,100}' <<< "${BUILD_SELECTOR}")
20
21 # Compute a unique app name using the reserved CF_APP name (configured in the
22 # deployer or from the manifest.yml file), the build number, and a
23 # timestamp (allowing multiple deploys for the same build).
24 NEW_APP_NAME="${CF_APP}-${SELECTED_BUILD}$(date +%s)"
25
26 # Domain can be customized if needed.
27 DOMAIN=mybluemix.net
28 # Used to define a temporary route to test the newly deployed app
29 TEST_HOST=$NEW_APP_NAME
30 # Java artifact being deployed. Needs to match output from
31 # the build process.
32 WAR=Pipeline_hello1.war
33 MEMORY=256M

```

NEW_APP_NAME 很重要，因为它展示了如何根据构建版本编号和时间戳为每次部署计算一个新应用程序名称。颜色变量用于让日志文件更容易读取。

Deployer 配置中指定的应用程序名称在脚本中使用 CF_APP 环境变量引用，目标空间使用 CF_SPACE 变量引用。

- 在空间（稍后将删除）中找到已部署的应用程序名称。我在空间中搜索所有与 PUBLIC_HOST 有关联的应用程序，在每次匹配中忽略计算的构建版本编号和时间戳：

```

1 rm -f apps.txt
2 echo -e "${label_color}Find all existing deployments: ${no_color}"
3 cf apps | { grep $PUBLIC_HOST || true; } | sed -r 's/\x1B[([0-9]{1,2};[0-9]{1,2})?}[mGK]//g'
4 &> apps.txt
5 cat apps.txt
6 # Store just the deployed application names
7 awk '{ print $1 }' apps.txt > app_names.txt
8 rm -f apps.txt

```

- 使用计算的名称和路由从选定的构建版本推送更新：

```

1 echo -e "${label_color}Pushing new deployment - ${green}$NEW_APP_NAME${no_color}"
2 cf push $NEW_APP_NAME -p $WAR -d $DOMAIN -n $TEST_HOST -m $MEMORY
3 DEPLOY_RESULT=$?
4 if [ $DEPLOY_RESULT -ne 0 ]; then
5     echo -e "${red}Deployment of $NEW_APP_NAME failed!!"
6     cf logs $NEW_APP_NAME --recent
7     return $DEPLOY_RESULT
8 fi
9 echo -e "${label_color}APPS:${no_color}"
10 cf apps | grep ${CF_APP}

```

我使用了一个基本的 push 命令。在对未来的应用程序应用此技术时，可以调整 push 命令来满足您的需求，包括根据需要创建服务。

- 测试以确保新应用程序能正常工作。这里，我使用一个基本的 cURL 命令来确保应用程序能够响应。此刻您可能想要执行更多的测试，以确定新应用程序是否已准备好处理负载：

```

1 echo -e "${label_color}Testing new app - ${green}$NEW_APP_NAME${no_color}"
2 curl "http://$TEST_HOST.$DOMAIN"
3 TEST_RESULT=$?
4 if [ $TEST_RESULT -ne 0 ]; then
5   echo -e "${red}New app did not deploy properly - ${green}$NEW_APP_NAME${no_color}"
6   return $TEST_RESULT
7 else
8   echo -e "${green}Test PASSED!!!!${no_color}"
9 fi

```

5. 通过将现有的和新的应用程序绑定到公共路由，将流量映射到新应用程序。

Bluemix 自动将流量路由到两个实例：

```

1 echo -e "${label_color}Map public space route to new app - ${green}$NEW_APP_NAME${no_color}"
2 cf map-route $NEW_APP_NAME $DOMAIN -n $PUBLIC_HOST
3
4 echo -e "${label_color}Public route bindings:${no_color}"
5 cf routes | { grep $PUBLIC_HOST || true; }

```

6. 删除临时测试路由，我们不再需要它。这一步不是绝对必要的，但它对清理不再使用的路由很有帮助：

```

1 echo -e "${label_color}Remove and delete test route - ${green}$TEST_HOST${no_color}"
2 cf unmap-route $NEW_APP_NAME $DOMAIN -n $TEST_HOST
3 if [ $? -ne 0 ]; then
4   echo -e "${label_color}Test route isn't mapped and doesn't need to be removed.${no_color}"
5 fi
6 cf delete-route $DOMAIN -n $TEST_HOST -f

```

7. 现在删除旧应用程序版本，仅保留新版本来接收来自公共路由的流量：

```

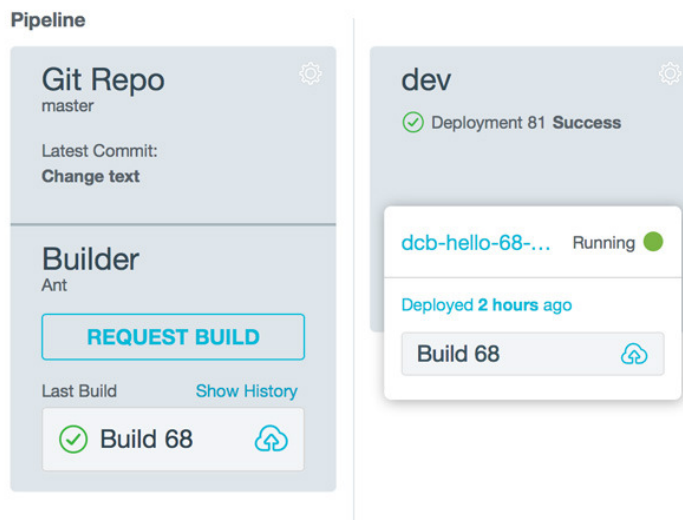
1 while read name
2 do
3   if [ "$name" != "$NEW_APP_NAME" ]; then
4     echo -e "${label_color}Deleting old deployed application - $name${no_color}"
5     cf delete $name -f
6   fi
7 done < app_names.txt
8 rm -f app_names.txt

```

第 5 步：检查部署请求的日志

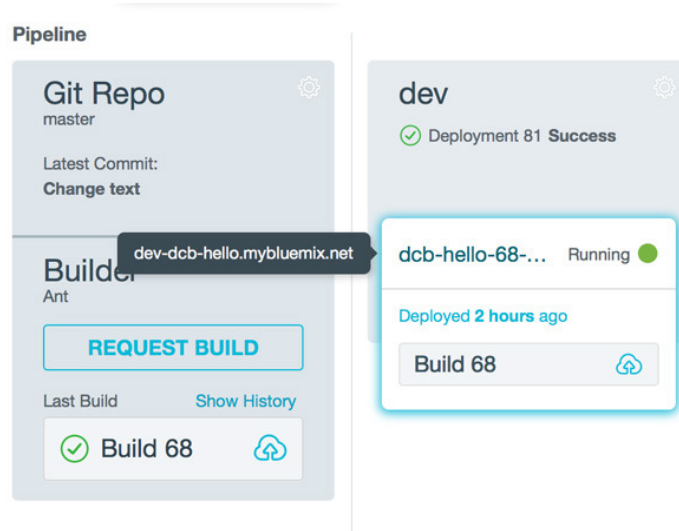
在设置好管道后，就可以请求一次新构建并部署它：

1. 单击 **REQUEST BUILD** 按钮构建项目并自动触发一次部署。
2. 单击 **REQUEST BUILD** 按钮请求另一次构建。在这个示例中，请求在构建后将 Build 68 部署到 dev 空间中。您的构建版本编号将会有所不同，但流程是相同的：



[点击查看大图](#)

3. 从悬停信息（将鼠标指针放在运行的应用程序名称上方）中可以看到用于访问 dev 空间的应用程序的路由。单击部署请求（在本例中为 **Deployment 81**）来查看部署日志：



[点击查看大图](#)

4. 可以看到分配的变量和已从 **Build 68** 部署的应用程序。请注意，计算的 **NEW_APP_NAME** 使用构建版本编号和时间戳作为后缀名：

```

1 Variables:
2 PUBLIC_HOST=dev-dcb-hello
3 DOMAIN=mybluemix.net
4 NEW_APP_NAME=dcb-hello-68-1408373927
5 TEST_HOST=dcb-hello-68-1408373927
6 WAR=Pipeline_hello1.war
7 MEMORY=256M
8 Find all existing deployments:
9 dcb-hello-67-1408373457 started 1/1 256M 1G dev-dcb-hello.mybluemix.net

```

5. **Build 68** 的内容使用计算的名称被推送到 Bluemix：

```

1 Pushing new deployment - dcb-hello-68-1408373927
2 Creating app dcb-hello-68-1408373927 in org danberg@us.ibm.com / space dev as danberg@
3 OK
4
5 Creating route dcb-hello-68-1408373927.mybluemix.net...
6 OK
7
8 Binding dcb-hello-68-1408373927.mybluemix.net to dcb-hello-68-1408373927...
9 OK
10
11 Uploading dcb-hello-68-1408373927...
12 Uploading from: /home/jenkins/workspace/b3a8fa2f-27f8-5b88-4007-76445be089ce/
13 0b7a50d5-2d8f-4dac-ac7b-c5b9aa3e6f27/out/Pipeline_hello1.war
14 2.5K, 3 files
15 OK
16
17 Starting app dcb-hello-68-1408373927 in org danberg@us.ibm.com / space dev as danberg@
18 OK
19 —> Downloaded app package (4.0K)
20 —> Uploading droplet (104M)
21
22 0 of 1 instances running, 1 starting
23 0 of 1 instances running, 1 starting
24 0 of 1 instances running, 1 starting
25 0 of 1 instances running, 1 starting
26 0 of 1 instances running, 1 starting
27 1 of 1 instances running
28
29 App started
30

```

可以看到，新应用程序和旧应用程序都已部署，但使用了不同的路由。旧 **Build 67** 使用了公共路由，而新部署使用了私有路由：

```

1 APPS:
2 dcb-hello-67-1408373457 started 1/1 256M 1G dev-dcb-hello.mybluemix.net
3 dcb-hello-68-1408373927 started 1/1 256M 1G dcb-hello-68-1408373927.mybluemix.net

```

6. 现在可使用测试路由，测试来自 Build 68 的新应用程序：

```

1 Testing new app - dcb-hello-68-1408373927
2 % Total % Received % Xferd Average Speed Time Time Time Current
3 Dload Upload Total Spent Left Speed
4
5 0 0 0 0 0 0 0 0 0 0 0 0 0
6 100 46 0 46 0 0 76 0 0 0 0 0 89
7 100 46 0 46 0 0 76 0 0 0 0 0 89
8 Rolling deployments made easy!!!0.0.0.0:63645
9 Test PASSED!!!!

```

7. 在成功测试后，将新应用程序绑定到公共主机 (dev-dcb-hello.mybluemix.net)。请注意，新和旧应用程序现在都被绑定到公共路由：

```

1 Map public space route to new app - dcb-hello-68-1408373927
2 Creating route dev-dcb-hello.mybluemix.net for org
3 danberg@us.ibm.com / space dev as danberg@us.ibm.com...
4 OK
5 Route dev-dcb-hello.mybluemix.net already exists
6 Adding route dev-dcb-hello.mybluemix.net to
7 app dcb-hello-68-1408373927 in org danberg@us.ibm.com /
8 space dev as danberg@us.ibm.com...
9 OK
10 Public route bindings:
11 dev-dcb-hello mybluemix.net dcb-hello-68-1408373927, dcb-hello-67-1408373457

```

8. 删除测试路由：

```

1 Remove and delete test route - dcb-hello-68-1408373927
2 Removing route dcb-hello-68-1408373927.mybluemix.net from app dcb-hello-68-1408373927 in
3 danberg@us.ibm.com / space dev as danberg@us.ibm.com...
4 OK
5 Deleting route dcb-hello-68-1408373927.mybluemix.net...
6 OK

```

9. 最后，删除旧应用程序，仅保留新应用程序来接收客户流量：

```

1 Deleting old deployed application - dcb-hello-67-1408373457
2 Deleting app dcb-hello-67-1408373457 in org danberg@us.ibm.com / space dev as danberg
3 OK
4 Sending deployment success of dcb-hello-67-1408373457 to IBM DevOps Services...
5 IBM DevOps Services notified successfully.
6 Deployed Applications:
7 dcb-hello-68-1408373927 started 1/1 256M 1G dev-dcb-hello.mybluemix.
8 Public route bindings:
9 dev-dcb-hello mybluemix.net dcb-hello-68-1408373927
10 You have successfully executed a rolling deployment of dcb-hello-68-1408373927.

```

结束语

滚动部署是任何生产级应用程序避免服务中断和保持用户满意的必备要素。您可以在 **Bluemix** 中手动执行应用程序的滚动发布，但这么做既麻烦又容易出错。最好自动化滚动部署，实现高度可重复且可靠的部署。

我展示了一种使用来自 **DevOps Services** 的 [交付管道服务](#) 实现自动化的滚动部署的方法。我使用了一个普通的应用程序，重点介绍了滚动部署脚本；实际上，您还需要自动化服务的设置（惟一或共享）。

借助来自本教程和示例的信息，您应该能够对任何 **Bluemix** 应用程序应用滚动部署。祝您在自己的滚动部署上取得成功，请将您可能遇到的任何问题发表到我们的 [论坛](#) 上。

本教程中用到的 **BLUEMIX** 服务：[交付管道服务](#) 能够自动构建和部署应用程序，同时也可以执行测试和构建脚本配置。

[注册免费的 Bluemix 试用版](#)[连接 Bluemix 开发者](#)

相关主题: [DevOps](#) [bash](#)