# 一、Docker 介绍

Docker

号称是下一代的虚拟机，它在启动和创建速度、性能、移植性等方面均优于传统虚拟机。Docker 是 PaaS 提供商 dotCloud开源的一个基于 LXC 的高级容器引擎。它能够让开发者打包他们的应用

以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux机器上，也可以实现虚拟化。目前，通过 Boot2Docker 已能使 Docker 运行在 Windows 和 OS X 中。Docker容器完全使用沙箱机

制，相互之间没有任何接口。Docker 几乎没有性能开销，可以很容易地在机器和数据中心中运行。最重要的是, 他们不依赖于任何语言、框架或系统。Docker 在其网站上明确地提到的 Docker 的

典型应用进行自动打包和部署 创建轻量、私有的 PAAS 环境.自动化测试和持续整合与部署.部署和扩展Web 应用、数据库和后端服务在自动化测试领域，Docker 大有取代传统虚拟机技术的趋势，

原因如下： 自动化测试依赖的是测试所需要的应用，而并非整个操作系统。因此，传统的虚拟机技术存在资源浪费。

Docker 构建于 LXC 之上。借助 LXC 轻量级的特性，相比于目前的虚拟机技术，Docker 启动更快、资源占用更小。

Docker 比虚拟机轻量、灵活，便于管理。

Docker 标准化的构建方法 (Dockerfile) 和良好的 REST API，使得自动测试和持续集成/部署能够很好的集成进来。

# 二、Sahi 介绍

Sahi 是一个开源的 Web UI 自动化测试框架。尽管它的知名度不及 Selenium，用户群也不及 Selenium 庞大，但它确有它独特的魅力，例如：基于上下文的页面元素识别机制。隐式页面加载响应

等待机制。 良好的浏览器支持。 优秀的跨浏览器录制回放调试工具 Sahi Controller。这些特性都大大地加快了自动化测试脚本的开发速度，并降低了维护成本。通过一个用 Java 编写的代理服

务器，将用户的脚本转换为 JavaScript 后注入往返的 HTTP请求及响应中，从而驱动浏览器事件。编程语言方面，除 Sahi 脚本（其本质是一个 JavaScript 库）以外，还支持 Java 和Ruby。

Sahi 有开源和收费两个版本。虽然收费版本提供了很多开源版本不具备的高级特性，不过开源版本已经能够满足大部分的功能要求。点此查看开源版本与收费版本的差异。点此查看Sahi 与其他

web 自动化测试框架的对比。

# 三、Jenkins 介绍

Jenkins

是一种开源的基于 Java 开发的持续集成工具，前身称作 Hudson。Jenkins提供了用于监控持续重复工作的软件平台。它支持丰富的插件，用户可以按照需求进行选择安装和配置，以实现生成执

行状态表格，自动部署、更新自动化测试包等高级功能。本文将要介绍 Jenkins 的 Docker 插件，它能够动态地创建 Docker 容器作为 Jenkins Slave节点，并在执行任务后，自动关闭容器。另

外，它还支持一些额外功能，比如当构建任务成功完成后自动将容器保存为镜像、自动将镜像上传到资源库等。

# 四、实例演示

该实例演示如何制作一个运行 Sahi 的镜像以及如何在 Jenkins 上配置 Docker 插件以运行 Sahi 测试脚本。

准备 Docker 镜像

本实例需要准备三个镜像：一个运行 Jenkins，一个运行 Subversion，另外一个运行 Sahi。

首先，制作运行 Sahi 的镜像。Docker Hub 是一个用于分享 Docker 镜像的资源库。目前，该资源库还没有运行 Sahi 的镜像，所以必须自己创建 Dockerfile 来构建镜像。下图是用来制作 Sahi

镜像的目录。

图 1. 制作 Sahi 镜像的目录

下图是 Dockerfile 的文件内容。

图 2. Dockerfile文件内容

整个过程大体分为八个步骤：

指定基础镜像

我们使用的是 evarga/jenkins-slave。该镜像基于 Ubuntu，安装了 SSH 服务，并创建了用户 Jenkins。

替换系统默认更新源。把系统更新源替换为 163 的，后面的软件安装速度会比较快一些。

安装必要的软件（不包括 Oracle Java 和

Sahi）：Firefox, Unzip 以及 Xvfb。删除 Open JDK 是为了后面安装 Oracle Java 做准备。Firefox 不用多说，因为该镜像用于 Web UI 自动化测试，所以安装了最新版本的 Firefox。安装

Unzip 是因为后面安装 Sahi需要用到它。Xvfb（X virtual framebuffer）是一个 X11显示服务器的实现。它不是将图形在屏幕上显示出来，而是在内存中运行所有的图形操作。对客户端来说，除

了无法看到真正的图形显示，没有其他分别。一般称这种运行方式为 headless。

安装 Oracle Java 8。理论上 Sahi 应当也支持 Open JDK，所以安装 Oracle Java 不是必须的。

安装 Sahi。上传了 zip 文件后，用 Unzip 解压，Sahi 是基于 Java 的，所以解压后即已安装好 Sahi。之后，替换了几个文件，它们的作用如下：

userdata.properties 中有个属性叫

proxy\_alert.disabled，默认值为 false。用户第一次启动 Sahi Dashboard的时候，会弹出一个如下图所示的对话框。用户勾选了“Do no show this message again”并点击 Continue按钮之后

，Sahi 修改该属性值为 true。之后就不会在跳出这个对话框了。由于用 headless 的方式运行 Sahi无法点击该对话框，所以必须事先用一个已将该属性设置为 true 的文件替换 Sahi 默认的

图 3. Sahi 代理问题对话框

sahi\_example\_com

这是一个证书文件。用户第一次在 Sahi Dashboard

中打开 Firefox 时的页面如下图所示。用户需要点击 SSL Manager 链接手动接受 SSL 证书。此时，文件

sahi\_example\_com 被生成到 userdata/certs 目录下。之后，就不需要再进行该操作了。所以，复制的

sahi\_example\_com 目的也是为了自动完成这步需要在图形界面下才能完成的操作。

图 4. Sahi 启动页

图 5. SSLManager 的非信任连接页面

browser\_types.xml

替换该文件是为了使 Sahi 以 headless

的方式工作。在该文件中，事先添加了一个名字为 firefox-xvfb 的浏览器配置信息。之后，测试脚本就可以指向该浏览器运行。具体配置 Sahi 和 Xvfb 的步骤参考。

添加 init.sh 文件

init.sh 文件用于启动 Xvfb 和 Sahi。其内容如下。

图 6. init.sh 文件内容

“sleep 5”是因为 Sahi 启动需要一点时间，若立即运行测试脚本会导致失败。

修改权限

最终运行的容器是作为 Jenkins Slave 节点用

Jenkins 用户运行，因此将整个/usr/local 目录及子目录的所有人修改为 Jenkins，并给 Shell

文件添加执行权限。Jenkins 用户是在基础镜像 evarga/jenkins-slave 中创建的，所以在这个 Dockerfile

里没有创建 Jenkins 用户的语句。

指定默认执行的命令

CMD ["/usr/sbin/sshd","-D"] – 该语句令镜像默认启动 SSH 服务。事实上，这条语句也可以不添加，因为 evarga/jenkins-slave 的 Dockerfile 中已包含该语句。

一

切就绪之后，在该目录中执行“docker build -t shenrui/sahi –rm=true”。命令成功执行后，通过“docker

images”应当可以查看到名为“shenrui/sahi”的镜像。该镜像已经上传到 Docker Hub，有需要的读者可以自行拉取。

Sahi

镜像构建好之后，接下来准备 Jenkins 容器。Docker Hub 上已有官方的 Jenkins 镜像，直接用命令“docker pull jenkins”拉取。在 Docker 主机上，创建一个目录（例

如，/data/jenkins\_home），并修改权限（chmod 777）以便 Jenkins 容器能读写该目录，然后用命令“docker run -d -t --name myjenkins -p 8080:8080 -v

/data/jenkins\_home:/var/jenkins\_home jenkins”启动。此时，打开http://<Docker Host IP>:8080 应当可以看到 Jenkins 的管理页面。同样地，直接拉取 Docker Hub 的 Subversion

的镜像（docker pull bsboiko/subversion）。不过，还需要做进一步的配置。

步骤如下： 用命令“docker run -i -t bsboiko/subversion /bin/bash”以交互的方式启动 Subversion 容器。

在容器中创建目录 /var/svn/repos（mkdir -p /var/svn/repos），并基于该目录创建 Subversion 资源库（svnadmin create /var/svn/repos）。

在资源库目录下的 svnserve.conf 文件中删除“password-db = passwd”前面的注释符 #，并在 passwd 文件中添加一行“shenrui = password”。

用命令“docker commit <container id> mysvn”把上面所做的修改保存下来创建一个新镜像 mysvn。

最后，用命令“docker run -t -d -p 3690:3690 --name mysvn mysvn svnserve -d --foreground -r /var/svn/repos”启动 mysvn 容器。

启动时，添加容器到主机的端口映射“-p 3690:3690”以便之后直接通过主机 IP 提交测试脚本。

至此，Docker 上有两个正在运行的容器（docker ps），名字分别是 myjenkins 和 mysvn。