

基础知识准备

创建镜像

创建容器

编码实现UI测试



基础知识准备

创建镜像

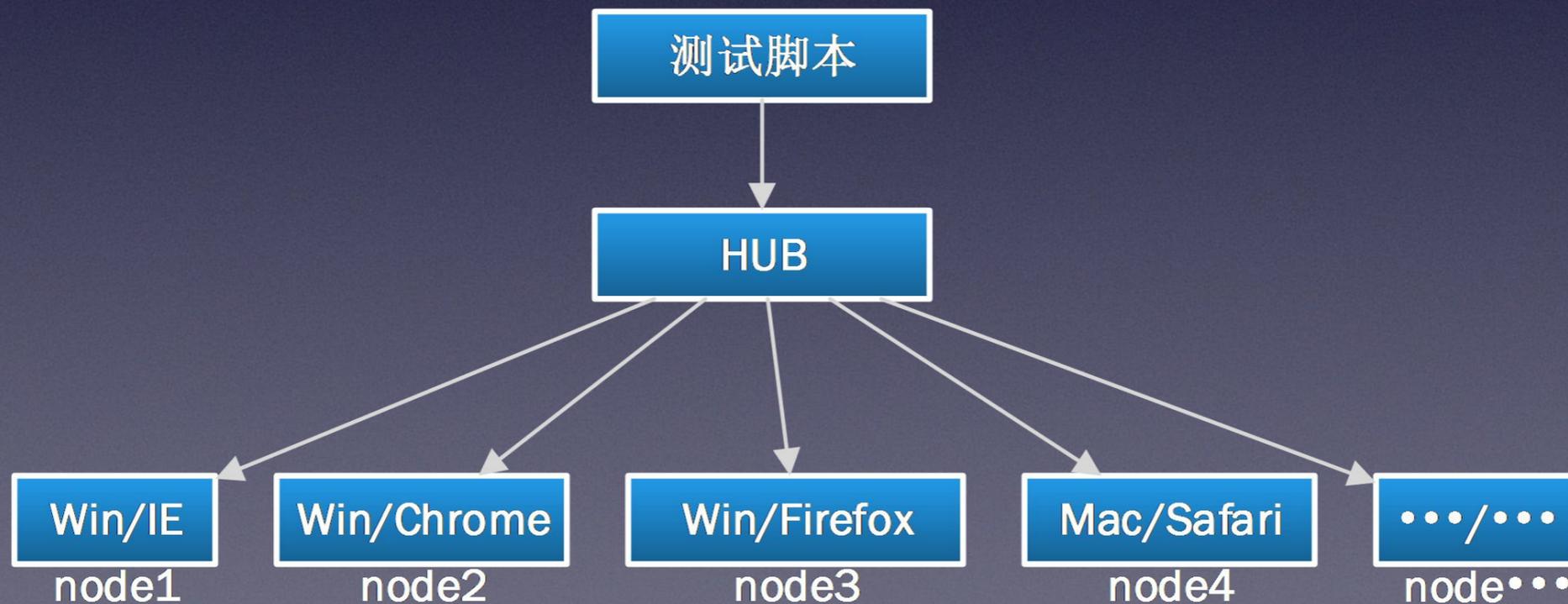
创建容器

编码实现UI测试

Selenium基础

Selenium 是针对Web应用的自动化测试框架和工具集合，支持多种浏览器和编程语言。测试用例直接运行在浏览器中，并模拟用户的操作。

Selenium Grid 是一种可以进行分布式自动化测试的辅助工具，该架构中包含两个主要角色：**Hub**是中心点控制节点，而**Node**是Selenium的工作节点，它们注册到Hub上，并会操作浏览器执行由Hub下发的自动测试用例。



Selenium Grid架构图

Docker基础

Docker是开源的应用容器引擎，让开发者可以打包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。主要有三个组件：

容器 container

容器是镜像创建的实例，是应用实际运行的位置

镜像 image

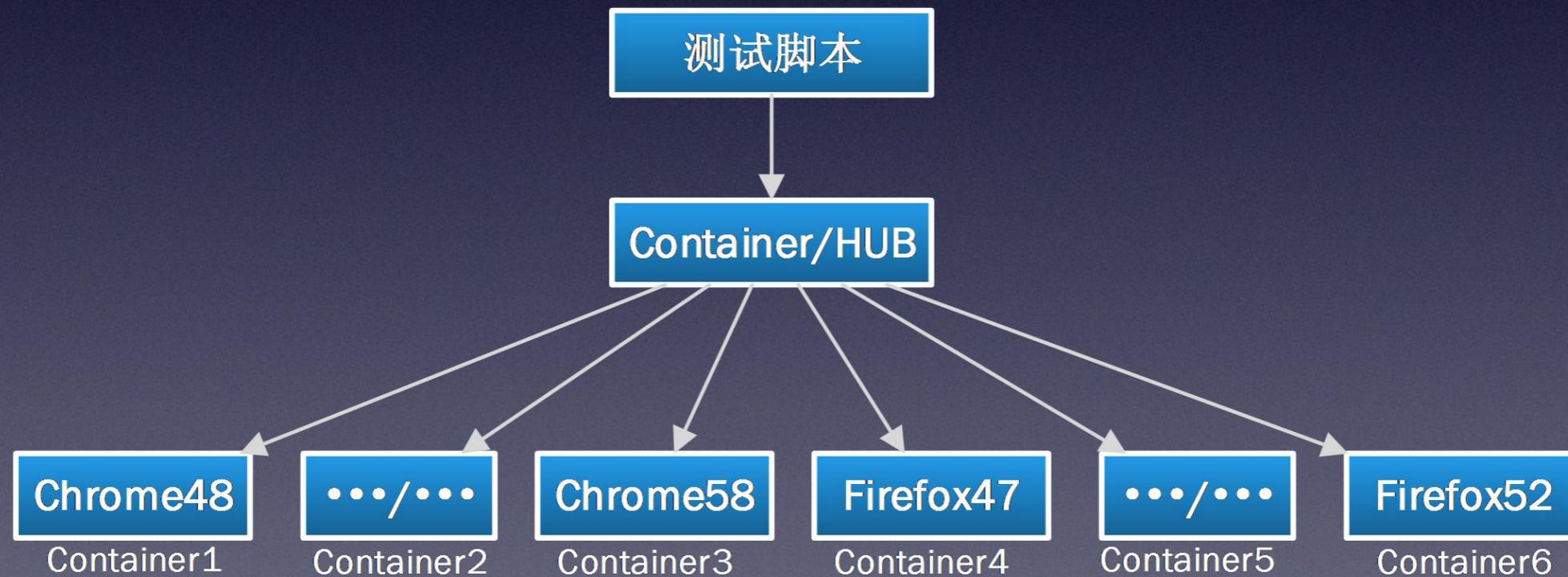
一个镜像可以包含一个完整的操作系统环境和用户需要的其它应用程序。**docker**的镜像是只可读的，一个镜像可以创建多个容器

仓库 repository

仓库是集中存放镜像的系统。hub.docker.com是**docker**官方的公开仓库，存放了大量的镜像供用户下载

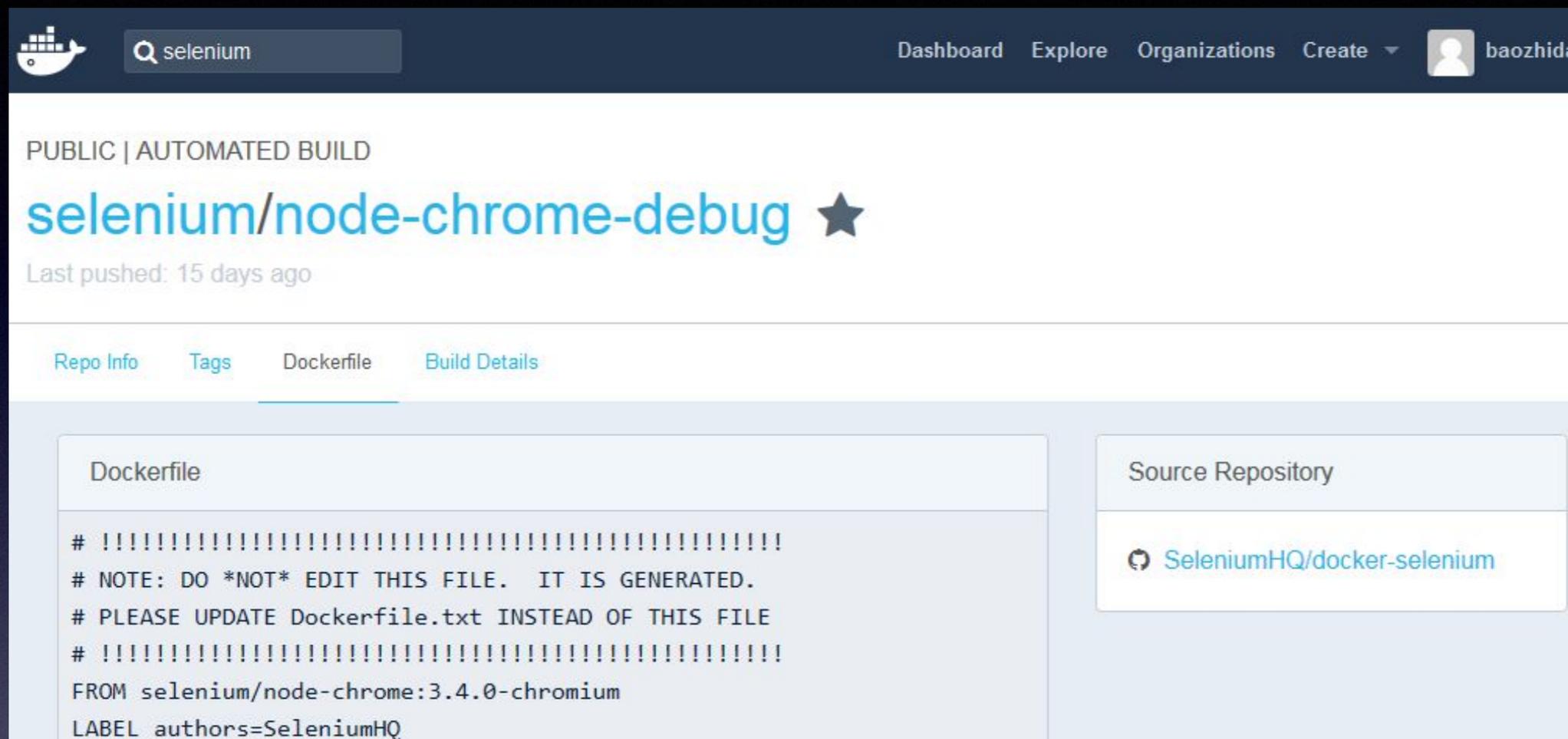
Selenium和Docker怎么结合使用？

目前可以把版本最多，更新最频繁的Chrome浏览器环境和Firefox浏览器环境部署在Docker容器中



Docker 集群中的Selenium Grid架构图

selenium官方镜像



官方镜像仓库地址:

<https://hub.docker.com/r/selenium/>

Docker仓库允许关联Github仓库自动构建镜像。

Dockerfile来自Github:

<https://github.com/SeleniumHQ/docker-selenium>

SeleniumHQ / docker-selenium

Watch 160 Star 1,521 Fork 612

Code Issues 101 Pull requests 15 Projects 1 Wiki Pulse Graphs

Docker images for Selenium Standalone Server <https://hub.docker.com/r/selenium/>

selenium selenium-grid selenium-node selenium-server webdriver docker docker-selenium

373 commits 17 branches 28 releases 50 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

ddavison committed on GitHub Merge pull request #475 from METAJJI/patch-2 Latest commit fe860cb 14 days ago

.github	be more specific about what "docker version" we need	15 days ago
Base	generic namespacing (allow others to easily customize image names)	15 days ago
Hub	More human readable generate_config	14 days ago
NodeBase	3.4.0 chromium update	15 days ago
NodeChrome	More human readable generate_config	14 days ago
NodeChromeDebug	3.4.0 chromium update	15 days ago
NodeDebug	3.4.0 chromium update	15 days ago
NodeFirefox	More human readable generate_config	14 days ago
NodeFirefoxDebug	3.4.0 chromium update	15 days ago

官方镜像存在两点问题

1. 系统字体不支持中文，中文网站乱码
2. 官方的镜像默认使用最新的稳定版，chrome现在是V58，没有老版本的镜像

基础知识准备

创建镜像

创建容器

编码实现UI测试

修改Dockerfile

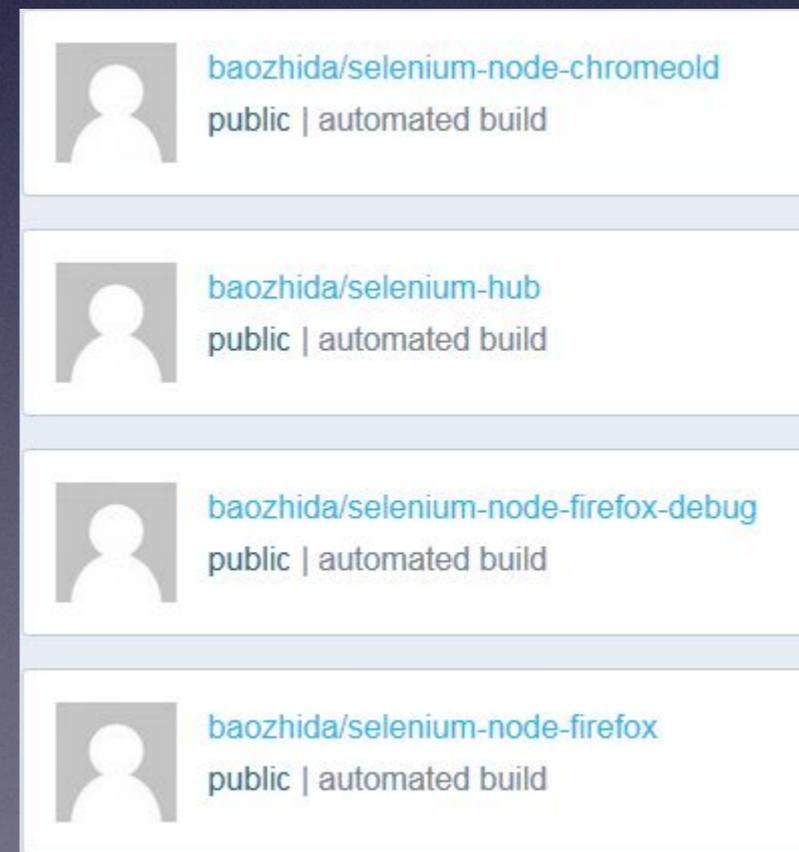
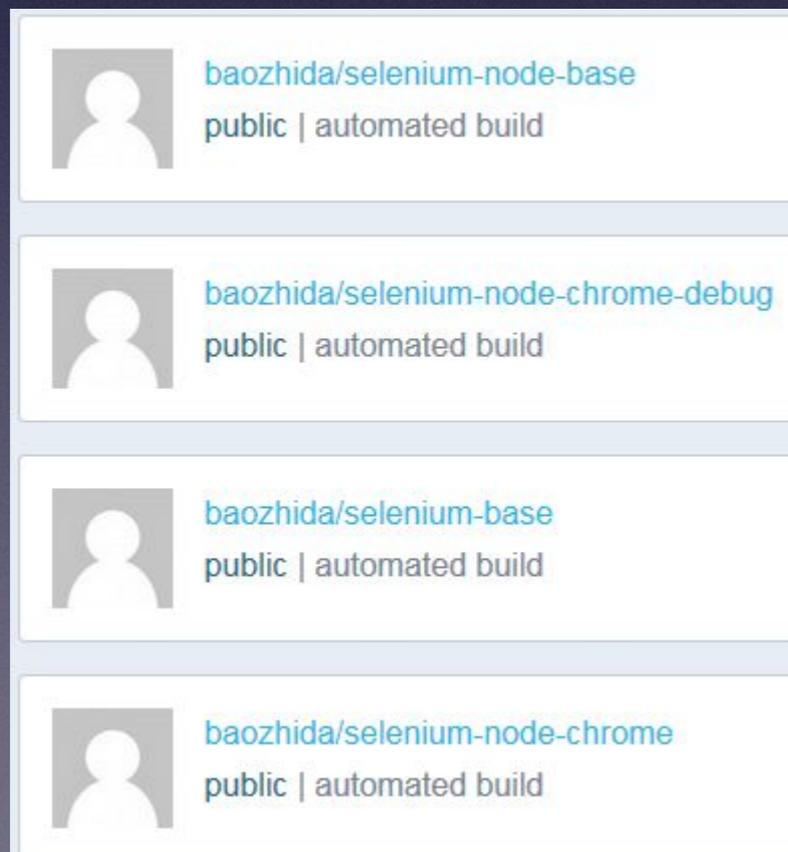
改版之后的DockerFile地址

<https://github.com/baozhida/docker-selenium>

Dockerfile安装的主要部件:

ubuntu16.04、java8、selenium jar包、browser(chrome和firefox)、driver(chromedriver和geckodriver)、VNCServer、一些可执行脚本等

仓库地址: <https://hub.docker.com/u/baozhida/>
新增的镜像:



Chrome浏览器镜像 V48-V58

PUBLIC | AUTOMATED BUILD

baozhida/selenium-node-chrome-debug

Last pushed: 13 minutes ago

Repo Info Tags Dockerfile Build Details Build Settings Collaborators Webhooks Settings

Status	Actions	Tag	Created	Last Updated
✓ Success		58		
✓ Success		48		
✓ Success		49		
✓ Success		50		
✓ Success		51		

Firefox浏览器镜像 V47-V52

PUBLIC | AUTOMATED BUILD

baozhida/selenium-node-firefox-debug

Last pushed: 3 days ago

Repo Info Tags Dockerfile Build Details Build Settings Collaborators Webhooks Settings

Status	Actions	Tag	Created	Last Updated
✓ Success		41	3 days ago	3 days ago
✓ Success		41	3 days ago	3 days ago
✓ Success		47	3 days ago	3 days ago
✓ Success		48	3 days ago	3 days ago
✓ Success		49	3 days ago	3 days ago
✓ Success		50	3 days ago	3 days ago

基础知识准备

创建镜像

创建容器

编码实现UI测试

安装使用Docker Toolbox

由于Docker不能在Windows7不能直接安装使用，但可以借助Docker Tolbox来完成。

主要包含以下一些内容：

-DockerClient客户端

用来运行docker引擎创建镜像和容器

-DockerMachine

在windows的命令行中运行docker引擎命令

-DockerCompose

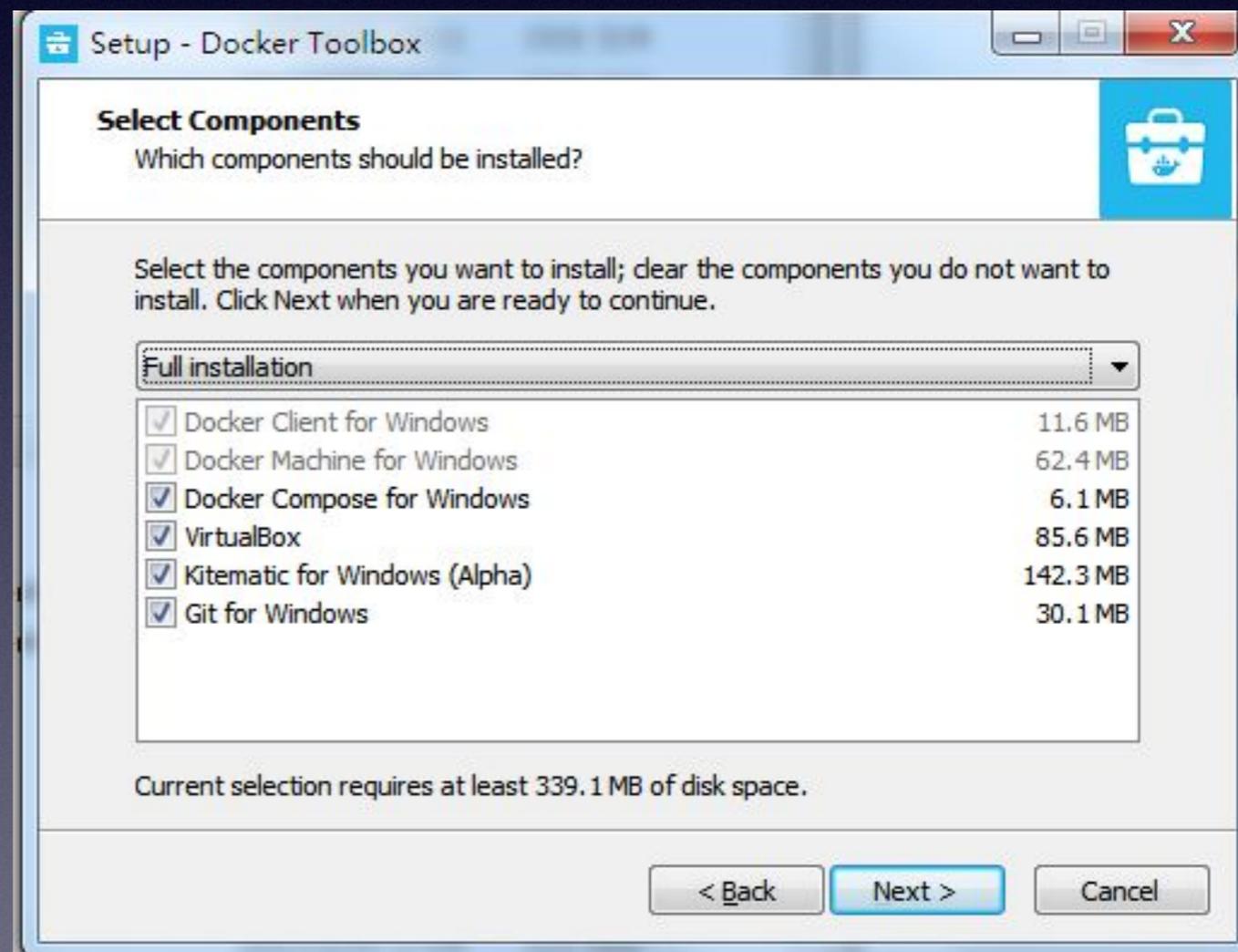
用来运行docker-compose命令

-Virtualbox

-Kitematic

这是Docker的GUI版本

-Git



下载镜像

命令 docker pull: 从docker hub中下载镜像

```
Docker Pull Command
```

```
docker pull baozhida/selenium-hub
```

PUBLIC | AUTOMATED BUILD

baozhida/selenium-hub

Last pushed: 10 days ago

[Repo Info](#) [Tags](#) [Dockerfile](#) [Build Details](#)

Tag Name	Compressed Size
3.3.1	189 MB

```
docker@default:~$ docker pull baozhida/selenium-hub:3.3.1
$ docker pull baozhida/selenium-node-chrome-debug:48
$ docker pull baozhida/selenium-node-chrome-debug:58
$ docker pull baozhida/selenium-node-firefox-debug:52
$ docker pull baozhida/selenium-node-firefox-debug:47
```

创建容器

创建selenium hub容器

```
docker run -d -p 4444:4444 --name selehub baozhida/selenium-hub:3.3.1
```

创建chrome node容器

```
docker run -d -p 5911:5900 --name node48 --link selehub:hub --shm-size=512m  
baozhida/selenium-node-chrome-debug:48
```

```
docker run -d -p 5901:5900 --name node58 --link selehub:hub --shm-size=512m  
baozhida/selenium-node-chrome-debug:58
```

创建firefox node容器

```
docker run -d -p 5917:5900 --name ff47 --link selehub:hub --shm-size=512m  
baozhida/selenium-node-firefox-debug:47
```

```
docker run -d -p 5912:5900 --name ff52 --link selehub:hub --shm-size=512m  
baozhida/selenium-node-firefox-debug:52
```

说明：-d参数：后台模式运行；--name参数：别名

-p参数：将容器的5900端口映射到docker的5901端口，访问Docker的5901端口即可访问到node容器；

--shm-size参数：docker默认的共享内存/dev/shm只有64m，有时导致chrome崩溃，该参数增加共享内存大小到512m

查看镜像和容器

命令 `docker images`: 查看本地的镜像

```
docker@default:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
baozhida/selenium-node-chrome-debug	58	36863bdc2bbf	2 days ago	923MB
baozhida/selenium-node-firefox-debug	41	8082e1d37462	5 days ago	719MB
baozhida/selenium-node-firefox-debug	47	2792d74ddc5c	5 days ago	729MB
baozhida/selenium-node-firefox-debug	48	84cfd60be18a	5 days ago	732MB
baozhida/selenium-node-firefox-debug	49	6b767b6914b6	5 days ago	732MB
baozhida/selenium-node-firefox-debug	50	6a400d0dd9bc	5 days ago	736MB
baozhida/selenium-node-firefox-debug	51	5e1f32dfde4a	5 days ago	735MB
baozhida/selenium-node-firefox-debug	52	6952015b6de6	5 days ago	736MB
baozhida/selenium-hub	3.3.1	8bbca693c0c0	9 days ago	394MB
baozhida/selenium-node-chrome-debug	48	d11acf8b035b	10 days ago	976MB
baozhida/selenium-node-chrome-debug	49	71da79535876	10 days ago	979MB
baozhida/selenium-node-chrome-debug	50	9ef3f1cdcd07	10 days ago	979MB
baozhida/selenium-node-chrome-debug	51	37d470f893c5	10 days ago	985MB
baozhida/selenium-node-chrome-debug	52	229910d50a3e	10 days ago	986MB
baozhida/selenium-node-chrome-debug	53	0efcafd4177d	10 days ago	988MB
baozhida/selenium-node-chrome-debug	54	7d5f9c1238fb	13 days ago	971MB
baozhida/selenium-node-chrome-debug	55	778cc5983791	2 weeks ago	974MB
baozhida/selenium-node-chrome-debug	56	006b6f8b925e	2 weeks ago	973MB
baozhida/selenium-node-chrome-debug	57	c182e392b972	2 weeks ago	977MB

docker ps -a 查看正在运行的容器

```
docker@default:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND
1d7461ef7204	baozhida/selenium-node-firefox-debug:52	"/opt/bin/entry_po..."
5cc579c2da7b	baozhida/selenium-node-firefox-debug:47	"/opt/bin/entry_po..."
7cd671ef819e	baozhida/selenium-node-chrome-debug:58	"/opt/bin/entry_po..."
a45747c77293	baozhida/selenium-node-chrome-debug:48	"/opt/bin/entry_po..."
48da09743f0a	baozhida/selenium-hub:3.3.1	"/opt/bin/entry_po..."

CREATED	STATUS	PORTS	NAMES
About a minute ago	Up 58 seconds	0.0.0.0:5912->5900/tcp	ff52
About a minute ago	Up About a minute	0.0.0.0:5917->5900/tcp	ff47
About a minute ago	Up About a minute	0.0.0.0:5901->5900/tcp	node58
About a minute ago	Up About a minute	0.0.0.0:5911->5900/tcp	node48
About a minute ago	Up About a minute	0.0.0.0:4444->4444/tcp	selehub

浏览器看Grid控制台

http://192.168.99.100:4444/grid/console

The screenshot displays the Selenium Grid Console interface. At the top left is the Selenium logo (Se) and the title "Grid Console v.3.3.1". A "Help" link is located at the top right. The main area is divided into four panels, each representing a node:

- Node 1 (Top Left):** DefaultRemoteProxy (version : 3.3.1), id : http://172.17.0.3:5555, OS : LINUX. It has tabs for "Browsers" and "Configuration". Under "Browsers", it shows "WebDriver v:48.0.2564.109" with a Chrome icon.
- Node 2 (Top Right):** DefaultRemoteProxy (version : 3.3.1), id : http://172.17.0.5:5555, OS : LINUX. It has tabs for "Browsers" and "Configuration". Under "Browsers", it shows "WebDriver v:47.0" with a Firefox icon.
- Node 3 (Bottom Left):** DefaultRemoteProxy (version : 3.3.1), id : http://172.17.0.4:5555, OS : LINUX. It has tabs for "Browsers" and "Configuration". Under "Browsers", it shows "WebDriver v:58.0.3029.81" with a Chrome icon.
- Node 4 (Bottom Right):** DefaultRemoteProxy (version : 3.3.1), id : http://172.17.0.6:5555, OS : LINUX. It has tabs for "Browsers" and "Configuration". Under "Browsers", it shows "WebDriver v:52.0" with a Firefox icon.

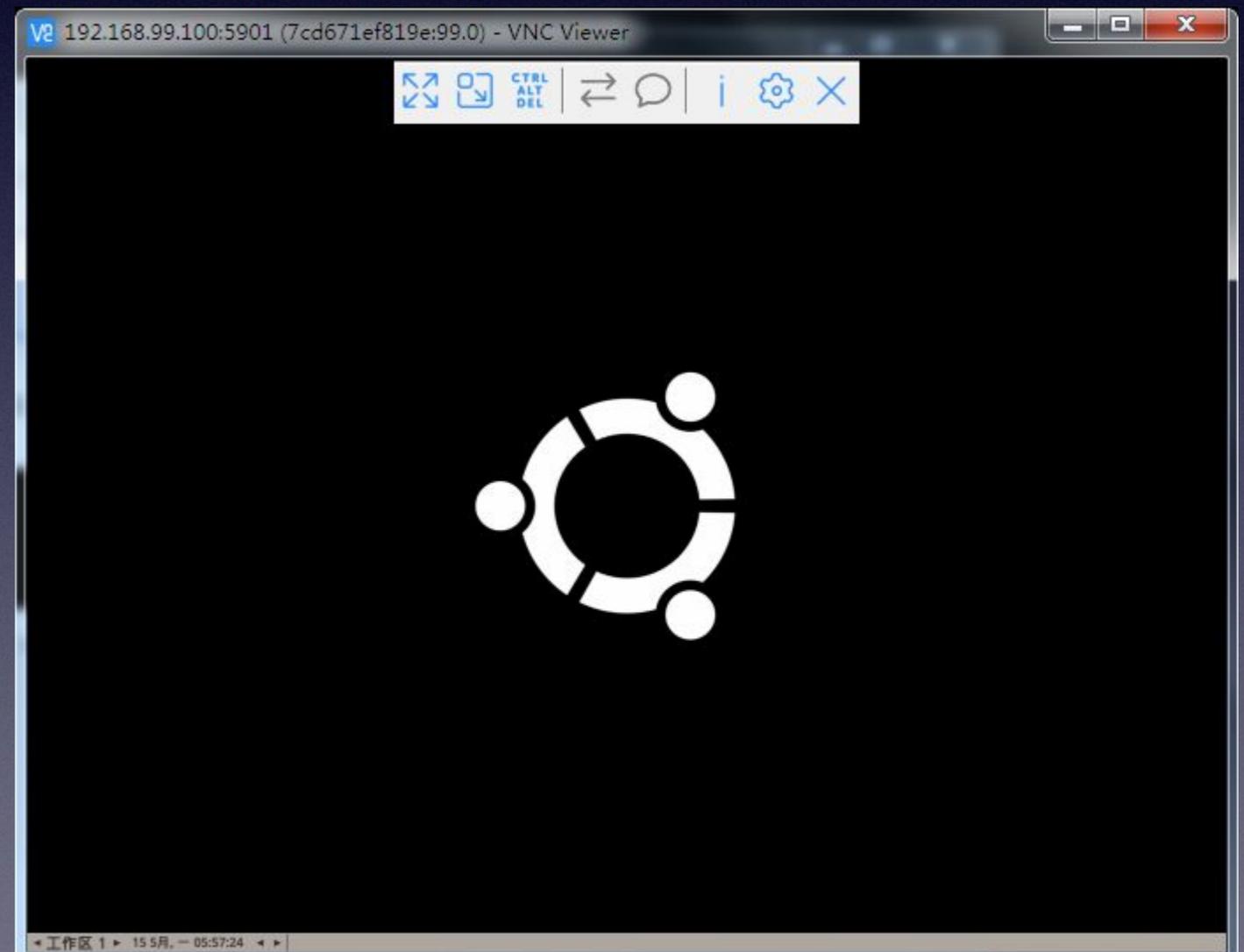
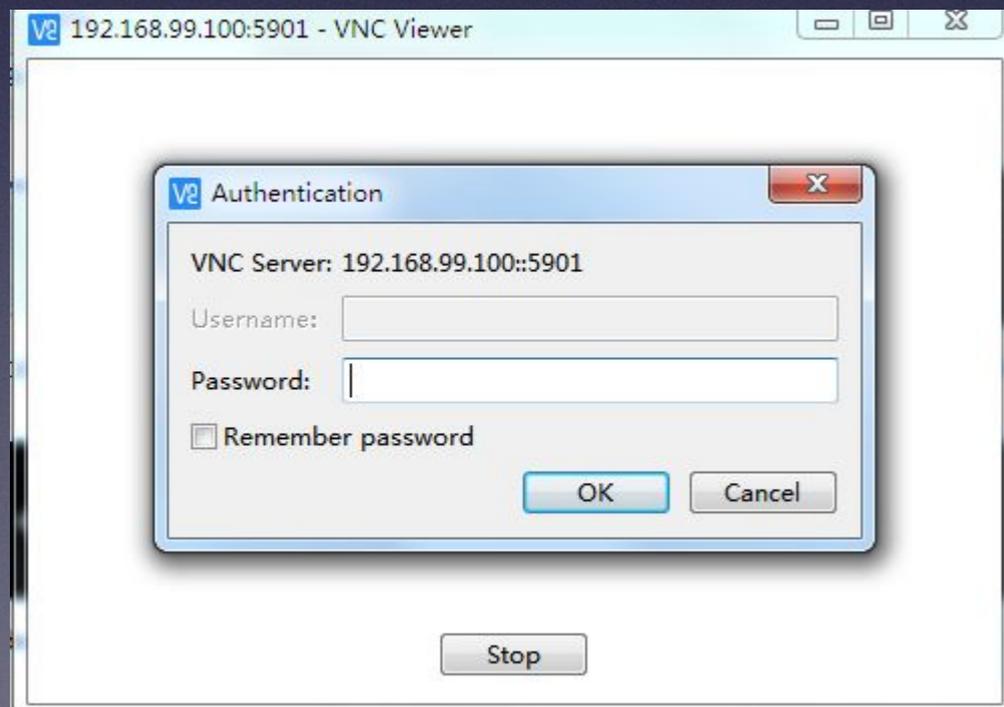
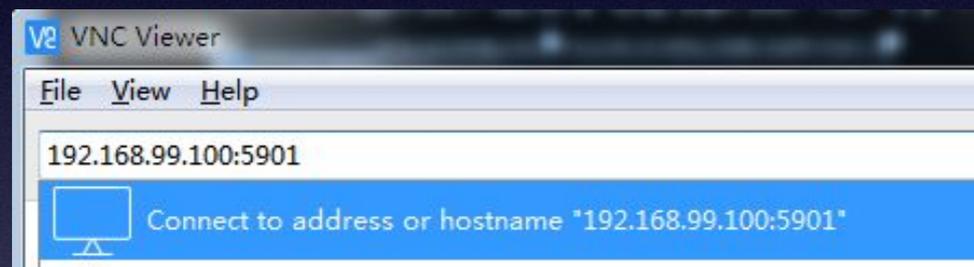
At the bottom left of the console, there is a link labeled "view config".

VNC远程浏览器环境

debug结尾的镜像都带有VNC服务端，本机安装VNC客户端，即可远程连接

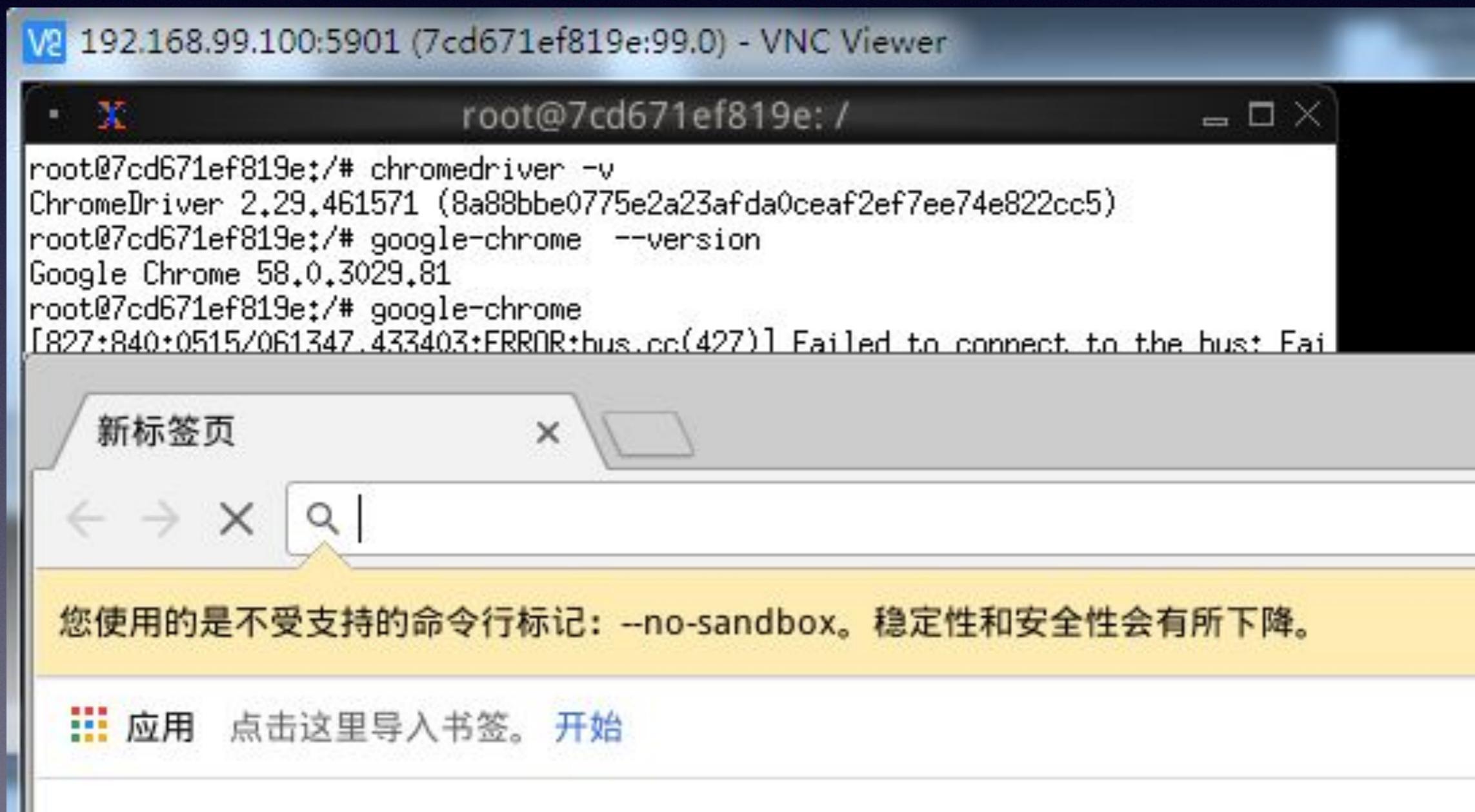
针对chrome58的容器

输入192.168.99.100:5901-->回车-->输入密码：secret-->确认-->进入容器桌面



远程桌面检查

`chromedriver -v` :查看启动的版本
`google-chrome --version` :查看浏览器版本
`google-chrome` : 直接启动浏览器



基础知识准备

创建镜像

创建容器

编码实现UI测试

Driver初始化(JAVA)

1.指定chrome，版本随机

```
WebDriver driver = new RemoteWebDriver(new URL("http://192.168.99.100:4444/wd/hub/"),  
DesiredCapabilities.chrome());
```

2.指定chrome版本

```
DesiredCapabilities desiredCaps= new DesiredCapabilities("chrome", "48.0.2564.109",  
Platform.LINUX);  
WebDriver driver = new RemoteWebDriver(new URL("http://192.168.99.100:4444/wd/hub/"),  
desiredCaps);
```

3.指定firefox，版本随机

```
WebDriver driver = new RemoteWebDriver(new URL("http://192.168.99.100:4444/wd/hub/"),  
DesiredCapabilities.firefox());
```

4.指定firefox版本

```
DesiredCapabilities desiredCaps= new DesiredCapabilities("firefox", "47.0", Platform.LINUX);  
WebDriver driver = new RemoteWebDriver(new URL("http://192.168.99.100:4444/wd/hub/"),  
desiredCaps);
```

TAP 自动化工具修改关键字可以使用docker容器。

主要修改初始化driver两处代码：

org\tn\qa\automation\component\web\Chrome.java

org\tn\qa\automation\component\web\Firefox.java

以Chrome.java为例：

```
public class Chrome extends WebBrowser {
    public Chrome(String chrome_version, String remoteip) {
        super();
        if (chrome_version == null || chrome_version == "" || remoteip == null || remoteip == "") {
            System.setProperty("webdriver.chrome.driver", Path.getChromeDriverPath());
            ChromeOptions options = new ChromeOptions();
            options.addArguments("--no-sandbox");
            options.setExperimentalOption("forceDevToolsScreenshot", true);
            driver = new ChromeDriver(options);
            mType = BrowserType.Chrome;
        } else {
            DesiredCapabilities desiredCaps = new DesiredCapabilities("chrome",
chrome_version, Platform.LINUX);
            try {
                driver = new RemoteWebDriver(new URL("http://" + remoteip + ":4444/wd/hub/"),
desiredCaps);
            } catch (MalformedURLException e) {e.printStackTrace();}
        }
    }
}
```

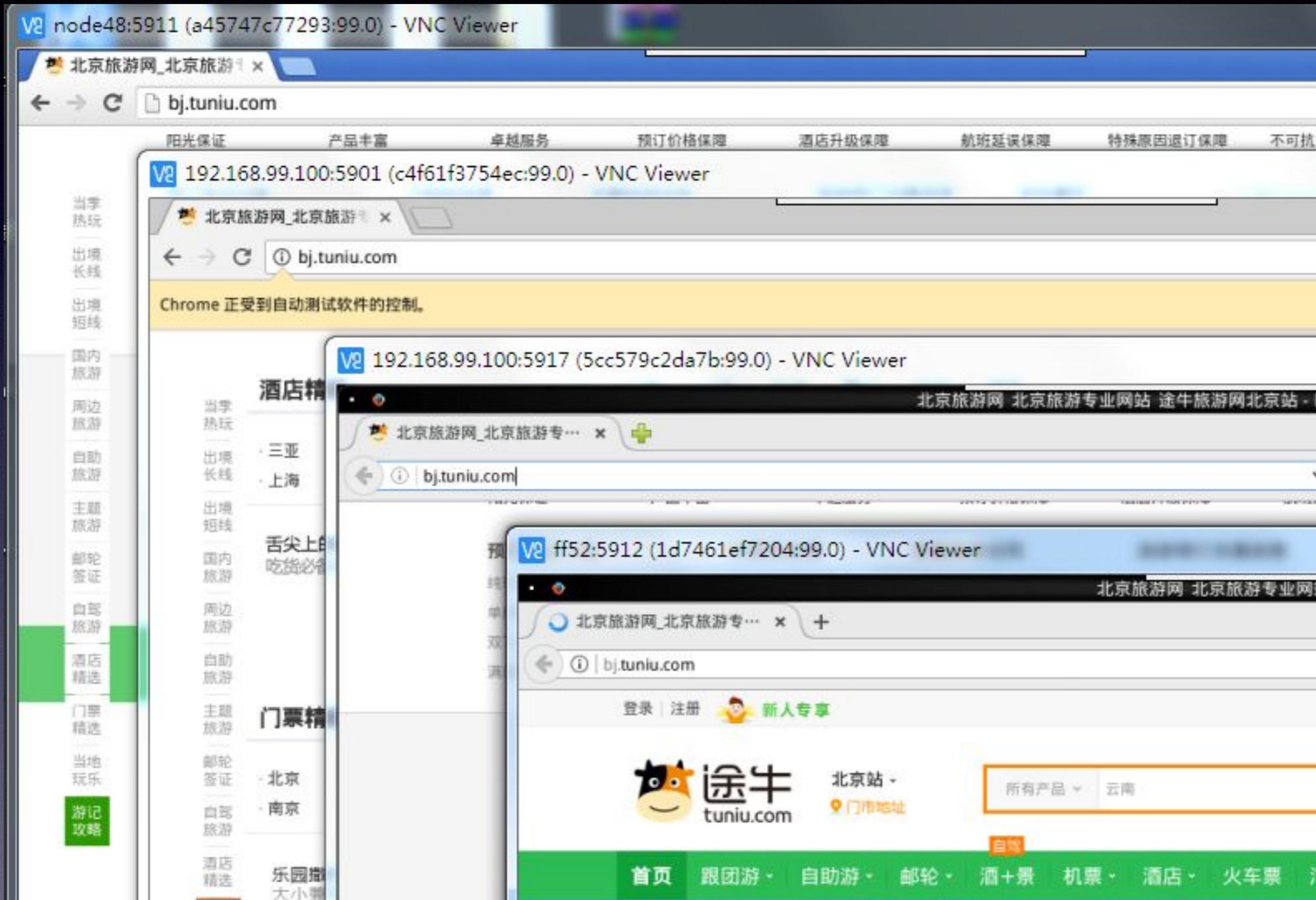
TestNG

是支持多线程测试框架，对分布式的UI自动化测试有着较好的支持。和ant和Jenkins结合可以搭建持续集成环境。

Demo项目：<https://github.com/baozhida/uiauto>
配置文件：testng.xml

```
testng.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="WEBUIAUTO" parallel="tests" thread-count="4">
4   <test name="chrome48">
5     <parameter name="browsertype" value="chrome" />
6     <parameter name="browserversion" value="48.0.2564.109" />
7     <parameter name="remoteIP" value="192.168.99.100" />
8     <classes>
9       <class name="com.tn.automation.testngcase.RunCase" />
10    </classes>
11  </test>
12
13 <test name="chrome58">
14   <parameter name="browsertype" value="chrome" />
15   <parameter name="browserversion" value="58.0.3029.81" />
16   <parameter name="remoteIP" value="192.168.99.100" />
17   <classes>
18     <class name="com.tn.automation.testngcase.RunCase" />
19   </classes>
20 </test>
```

并发执行



测试报告

UIAuto TestRep

[Overview](#)

- WEBUIAUTO
- [ff52](#) ✘
- [ff47](#) ✔
- [chrome58](#) ✔
- [chrome48](#) ✔

Generated by [TestNG](#) with [ReportNG](#) at 10:44 CST on 星期五 19 五月 2017
baozhida@TN-00-50000181 / Java 1.8.0_101 (Oracle Corporation) / Windows 7 6.1 (amd64)

UIAuto TestReport

WEBUIAUTO	Duration	Passed	Skipped	Failed	Pass Rate
ff52	103.789s	1	0	1	50%
ff47	110.525s	2	0	0	100%
chrome58	120.368s	2	0	0	100%
chrome48	122.936s	2	0	0	100%
Total		7	0	1	87%

UIAuto TestReport

[Overview](#)

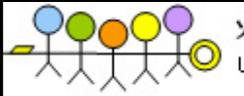
- WEBUIAUTO
- [ff52](#) ✘
- [ff47](#) ✔
- [chrome58](#) ✔
- [chrome48](#) ✔

chrome58

Test duration: 120.368s

Passed Tests

com.tn.automation.testngcase.RunWEB	
case_001检查北京首页css_js引用连接	21.646s
case_002检查北京首页图片正常连接	85.741s



Q&A