Running UI tests in a local VM

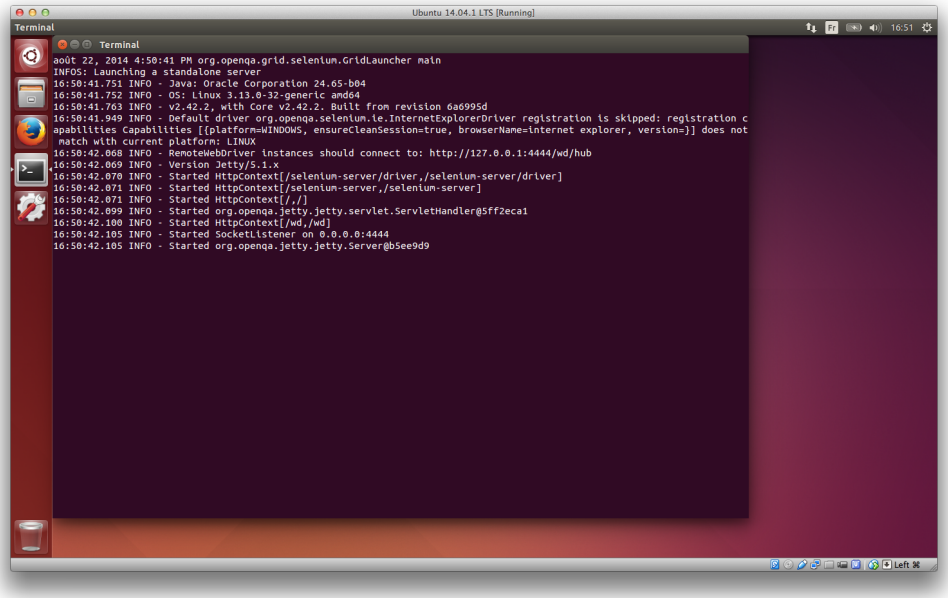
This page *probably* is outdated partially.

For a more up-to-date documentation regarding UI tests for Magnolia 6.1 and higher, please also check the documentation on bitbucket.

UI tests typically require browser/machine focus (especially when entering text in input fields). To avoid being stuck while UI tests proceed, it's handy to run them — the Selenium part — inside a VM.

Outlook

- The **Magnolia webapp** runs on the host as usual
  - with manual-tests profile
  - e.g. `mvn clean verify -P jetty9-standalone,manual-tests`
  - free hint #1: run this build from another location on your machine than the one you typically work at
  - free hint #2: run it in offline mode (`mvn -o ...`) if you just installed one of the modules you want to put under test
- A **Selenium standalone server** runs on the guest VM
- **Test execution** is done from the IDE, i.e. also on the host

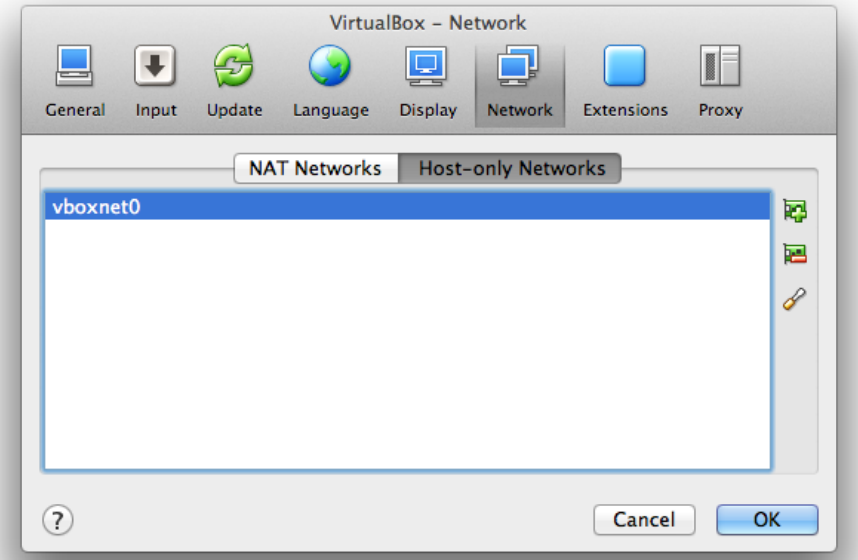


Instructions

1a. VM Setup manually

1. Setup your VM and install Ubuntu for example
   a. mount iso image in optical drive
2. In **VirtualBox** File > Host Network Manager..., click the add icon
   - creates a new virtual network interface e.g. `vboxnet0`

- (used to be Preferences > Network in former vbox versions, as illustrated below)



3. In **VM Settings** > Network
    - 1st slot: select "Host-only Adapter", then choose the one you just created — that `vboxnet0`
    - 2nd slot: select NAT as is usually the default (to access the interwebs through the host)
4. Install guest additions
    - When **VirtualBox VM** is running
    - Devices > Insert Guest Additions CD image...

1b. VM Setup with Vagrant

1. Download and install *vagrant* from https://www.vagrantup.com or use the *vagrant manager* http://vagrantmanager.com
2. Clone the repro https://git.magnolia-cms.com/users/mmuehlebach/repos/uitestmachine
3. Go to the cloned directory and run `vagrant up`
4. Start selenium server with `./selenium-server.sh`

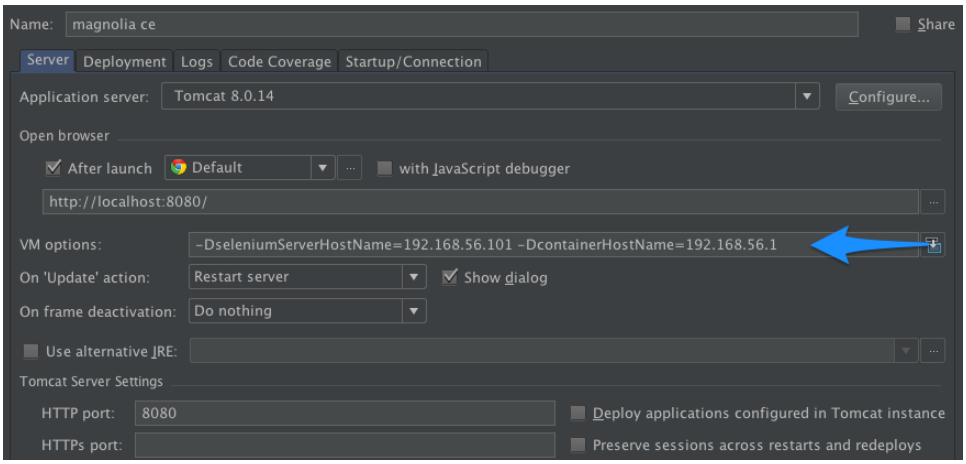
2. Selenium server setup

1. In the VM, download **Selenium Server (formerly the Selenium RC Server)**
    - http://docs.seleniumhq.org/download/
        - in case of problems with the only linked version (i.e. 3.0.0-beta3), links to older packages can be found here: http://selenium-release.storage.googleapis.com/, e.g. http://selenium-release.storage.googleapis.com/2.52/selenium-server-standalone-2.52.jar
    - should be something like `selenium-server-standalone-2.42.2.jar`
    - and then run it, e.g. `java -jar selenium-server-standalone-2.42.2.jar`
        - For Selenium 3+, you'll need to get Geckodriver first and provide it's location to Selenium through the webdriver.gecko.driver (Java) system property.
          To do so, either
            - run it as `java -jar -Dwebdriver.gecko.driver=/path/to/geckodriver selenium-server-standalone-3.3.1.jar`
            - or once set it globally in `/etc/environment`, e.g. through `echo "_JAVA_OPTIONS='-Dwebdriver.gecko.driver=/path/to/geckodriver" | sudo tee -a /etc/environment`, then log out and back in for the changes to take effect
2. On your host run the UI Tests with the variables `seleniumServerHostName` which is the address of your newly created VM and `containerHostName` which is the address used of your VM to access your host system (this is probably not equal to your machines address in the magnolia network)

```
$ mvn -U clean install -Pjetty9-standalone,ui-tests -DseleniumServerHostName=192.168.56.101 -
DcontainerHostName=192.168.56.1
```

If necessary replace the IP addresses.

3. In your run configuration in *IntelliJ* you have to add these two variables `seleniumServerHostName` and `containerHostName` as well.



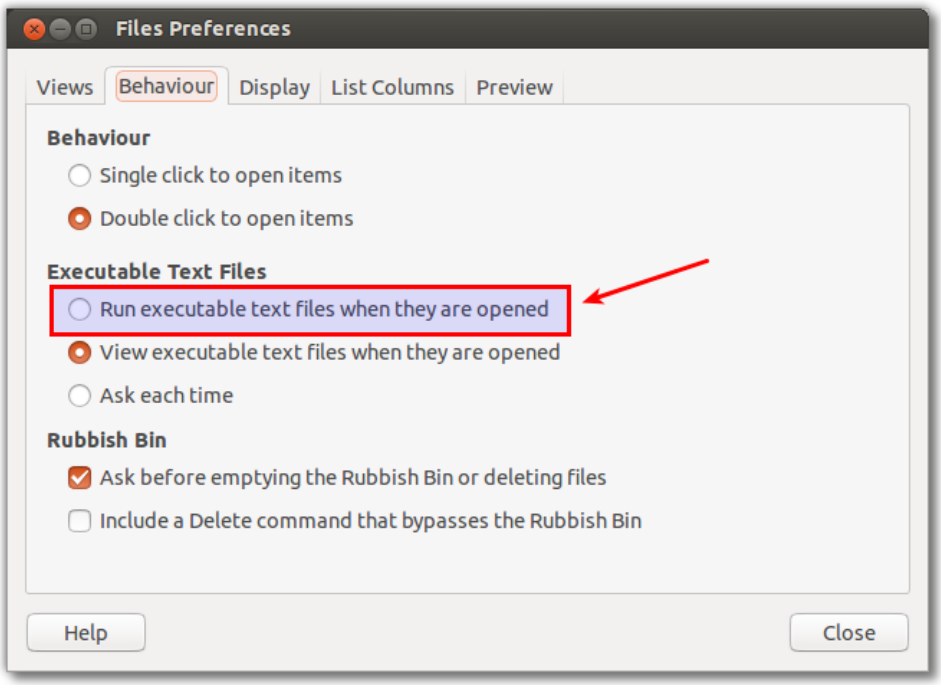
3. Fancy shell script + desktop launcher for the Selenium server

1. Create a new script, e.g. `selenium-server` on the desktop as follows
   - (provided your selenium server jar is on the desktop too)
   - **selenium-server.sh**

```
#!/bin/bash
gnome-terminal -e "java -jar selenium-server-standalone-2.42.2.jar"
```

2. Make it executable
   - `chmod +x your-script`
   - Go to Nautilus (eq. Finder) preferences
     - Edit > Preferences, "Behavior" tab

- Tick "Run executable text files when they are opened"



3. The fancy icon
   - Download it http://www.seleniumhq.org/images/selenium-logo.png and save it locally to e.g. `/home/{user}/.icons/`
   - Select your script, right-click and go to the file Properties > Click the icon and select yours

Double-click your script and you're good to go! 😁

Tips for simulating low-end hardware (like on Jenkins test runs)

When tests seem to fail randomly in a non-reproducible manner on from Jenkins, it's sometimes related to slower hardware compared to local execution. To be more consistent and robust, it can be helpful to mimic such a low-end environment.

Use low screen resolution

Some failures (like hidden dialog commit buttons) only occur on screens with low resolution. With VirtualBox guest extensions properly installed, you should be able to simply resize the VM window and the guest's virtual screen should automatically adjust. As a reference for size, screenshots from failed Jenkins build may be used; or just roughly 1000 x 700 px.

Throttle Magnolia

Other test failures are caused by racing conditions only surfacing on slow systems, where certain elements take longer to load than usual. For Linux (now OS X too), there's a command line tool called cpulimit to set an upper CPU time percentage limit for certain processes.

- On Ubuntu, install with `sudo apt-get install cpulimit` or simply click here (apt-url)
- On OS X, clone it from GitHub and check the README for build instructions or with brew → cpulimit with brew

Setting the limit to 5% for the process with id 1234 is done by

```
cpulimit -l 5 -i 1234
```