Rootid: Setting up Behat Testing on a Mac with Mink and Selenium

Ok, you've got a choice. You can either get this stuff set up and running in your local environment, or you can get this stuff set up and running using Docker containers. Or both, if that's how you roll... Whichever you choose, there's also a certain amount of setup you have to do for each site.

Setting up Behat on your local

Use homebrew to install chromedriver (so selenium can drive chrome), geckodriver (so selenium can drive firefox), phantomjs (which is what my team actually uses for our CircleCI testing) and the selenium server (so behat's selenium can talk to the drivers):

```
brew install geckodriver selenium-server-standalone
brew cask install chromedriver phantomjs
```

> Full Disclosure Note: I haven't been able to get Firefox to properly run tests. It opens and attempts the tests, but Mink's built-in "I should see" is failing (when it passes on Chrome and PhantomJS and the element is, in fact, visible), which is a rather bad sign... This is both with my local Firefox and the docker container Firefox, so I'm thinking there might be something wrong with geckodriver? Also, PhantomJS has officially been discontinued, and at some point is going to stop working :(The official recommendation is to use Chrome in headless mode.

Then use brew services to start up the selenium server and set it to restart every time the computer reboots:

```
sudo brew services start selenium-server-standalone
```

You don't need to start chromedriver or geckodriver -- since brew installed them in /usr/local/bin, they're in your PATH and selenium knows where to find them.

If you want to run tests using PhantomJS, you need to launch PhantomJS in a separate terminal tab using the command `phantomjs --webdriver=8643` before running your tests.

> Note that the number after `--webdriver=` is the port that PhantomJS will be listening on. If it's conflicting with something else on your computer, you can change that number, but you'll need to make sure to open up your behat.yml and change the wd_host port under your phantomjs definition to match. Also note that PhantomJS is headless, so while you can tell it to take screenshots there's nothing to watch.

If you'd like to set up docker containers instead or in addition, those instructions are next. Otherwise, jump down to the per site setup.

Setting up Behat in Docker containers

You can also use docker to run standalone Chrome, Firefox, and PhantomJS nodes. If you use the debug versions of Chrome and Firefox, you can even watch them run the tests using VNC. (PhantomJS is headless, so while you can tell it to take screenshots there's nothing to watch.)

First, download and install Docker. Make sure to launch the Docker app -- Docker commands only work when the app is running.

Then, copy and paste the (rather long but all-one-line) terminal commands:

```
docker run -d --name chrometest -p 4445:4444 -p 5901:5900 -v
/dev/shm:/dev/shm selenium/standalone-chrome-debug
```

and/or

```
docker run -d --name firefoxtest -p 4446:4444 -p 5902:5900 -v
/dev/shm:/dev/shm selenium/standalone-firefox-debug
```

and/or

```
docker run -d --name phantomtest -p 8910:8910 wernight/phantomjs phantomjs
--webdriver=8910
```

Each of those commands is telling docker to build and run a container in detached mode. Next you give each container a reasonable name (otherwise Docker will make up a weird name for you). The -p is telling docker that you want to map some_port_on_your_local:some_port_inside_the_docker_container. So you'll use port 4445 on your local to hook up to port 4444 inside the chrometest container, for example. 5900 is the VNC port, so you'll VNC into 5901 to watch your chrometest container and 5902 to watch your firefoxtest container. The -v business is dealing with memory for your container. The last bit is the name of the image Docker will use to build your container. If you haven't already downloaded the image, Docker will automatically do so when you try to build a container based on an image you don't have.

Now you can start and stop these containers from the terminal using:

```
docker start <container_name> <maybe_you_want_to_start_another_one_too>
```

and

```
docker stop <same_deal>
```

eg: `docker start chrometest` or `docker stop firefoxtest phantomtest`
Just make sure you've started a container before you try to use it! (Note that containers automatically stop when you shutdown/restart your computer.)

The stuff below goes in your behat.yml file (discussed in the per site setup below). As you can see, the selenium wd_host port should match the port you set up your docker containers to listen on.

```
dockerchrome:
    extensions:
        Behat\MinkExtension:
            browser_name: chrome
            selenium2:
                wd_host: 'http://localhost:4445/wd/hub'

dockerfirefox:
    extensions:
        Behat\MinkExtension:
            browser_name: firefox
            selenium2:
                wd_host: 'http://localhost:4446/wd/hub'

dockerphantomjs:
    extensions:
        Behat\MinkExtension:
            browser_name: phantomjs
            selenium2:
                wd_host: 'http://localhost:8910/wd/hub'
```

As long as your container is running, you can use it to run your behat tests using the terminal command `behat <optional_single_feature_filepath> -p <profile_name>` eg: `behat -p dockerfirefox` or `behat features/desktop-menu.feature -p dockerphantomjs`. The profile name should match the name in your behat.yml file. If you want to use a specific profile all the time, you can change the default values in the behat.yml.

Setup for each site

Make a new folder someplace logical for the site you want to test, and navigate to it in terminal.

Install behat, mink, goutte, and selenium2:

```
composer require behat/mink-extension behat/mink-goutte-driver
behat/mink-selenium2-driver --dev
```

Create a symlink so you can just type "behat" to run tests:

```
ln -s vendor/behat/behat/bin/behat behat
```

Initialize behat to create a couple of files and folders:

```
behat --init
```

Create a behat.yml file in the root of your site's test directory and paste in the following:

```
# Don't forget to change the base url!

# call specific browsers from the terminal using their profile. eg: behat
-p firefox or behat features/test.feature -p chrome
# Note that the default is chrome, but you can replace it with one of the
options and change the default


default:
    extensions:
        Behat\MinkExtension:
            base_url: https://site_you_want_to.test
            javascript_session: selenium2
            goutte: ~
            browser_name: chrome
            # browser_name: phantomjs
            # browser_name: firefox
            selenium2:
                wd_host: "http://localhost:4444/wd/hub"
                capabilities:
                    # marionette: null is required by
                    # mink-extension 2.3!!!!
                    # Without it, firefox launches
```

```
                    # (even if you specified chrome)
                    # but does nothing
                    marionette: null
    suites:
        default:
            contexts:
                - FeatureContext

chrome:
    extensions:
        Behat\MinkExtension:
            browser_name: chrome

firefox:
    extensions:
        Behat\MinkExtension:
            browser_name: firefox
            selenium2:
                capabilities:
                    marionette: true
phantomjs:
    extensions:
        Behat\MinkExtension:
            browser_name: phantomjs

dockerchrome:
    extensions:
        Behat\MinkExtension:
            browser_name: chrome
            selenium2:
                wd_host: 'http://localhost:4445/wd/hub'

dockerfirefox:
    extensions:
        Behat\MinkExtension:
            browser_name: firefox
            selenium2:
                wd_host: 'http://localhost:4446/wd/hub'

dockerphantomjs:
    extensions:
        Behat\MinkExtension:
```

```
            browser_name: phantomjs
            selenium2:
                wd_host: 'http://localhost:8910/wd/hub'
```

Edit the features/bootstrap/FeatureContext.php, so it contains the following:

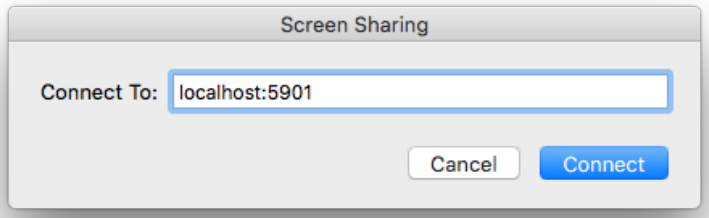
> Note: You especially want to get the MinkContext stuff, as it comes with all sorts of useful built-in definitions! To see what MinkContext adds, before you edit the FeatureContext.php file, type `behat -dl` in the terminal. (That's the command that lists all the definitions Behat knows about). The results will be... underwhelming. Do it again after you've added the MinkContext stuff 😄

```
<?php

use Behat\Behat\Context\Context;
use Behat\Gherkin\Node\PyStringNode;
use Behat\Gherkin\Node\TableNode;
use Behat\Behat\Context\SnippetAcceptingContext;
use Behat\MinkExtension\Context\MinkContext;

/**
 * Defines application features from the specific context.
 */
class FeatureContext extends MinkContext implements Context,
SnippetAcceptingContext
{
    /**
     * Initializes context.
     *
     * Every scenario gets its own context instance.
     * You can also pass arbitrary arguments to the
     * context constructor through behat.yml.
     */
    public function __construct()
    {
    }
}
```

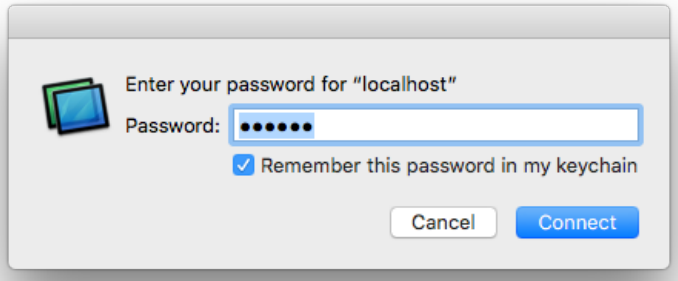
Assuming that at some point you're going to use git to sync these tests with a repository somewhere, create /.gitignore and paste in:

```
# Don't sync composer's vendor folder!
vendor
```

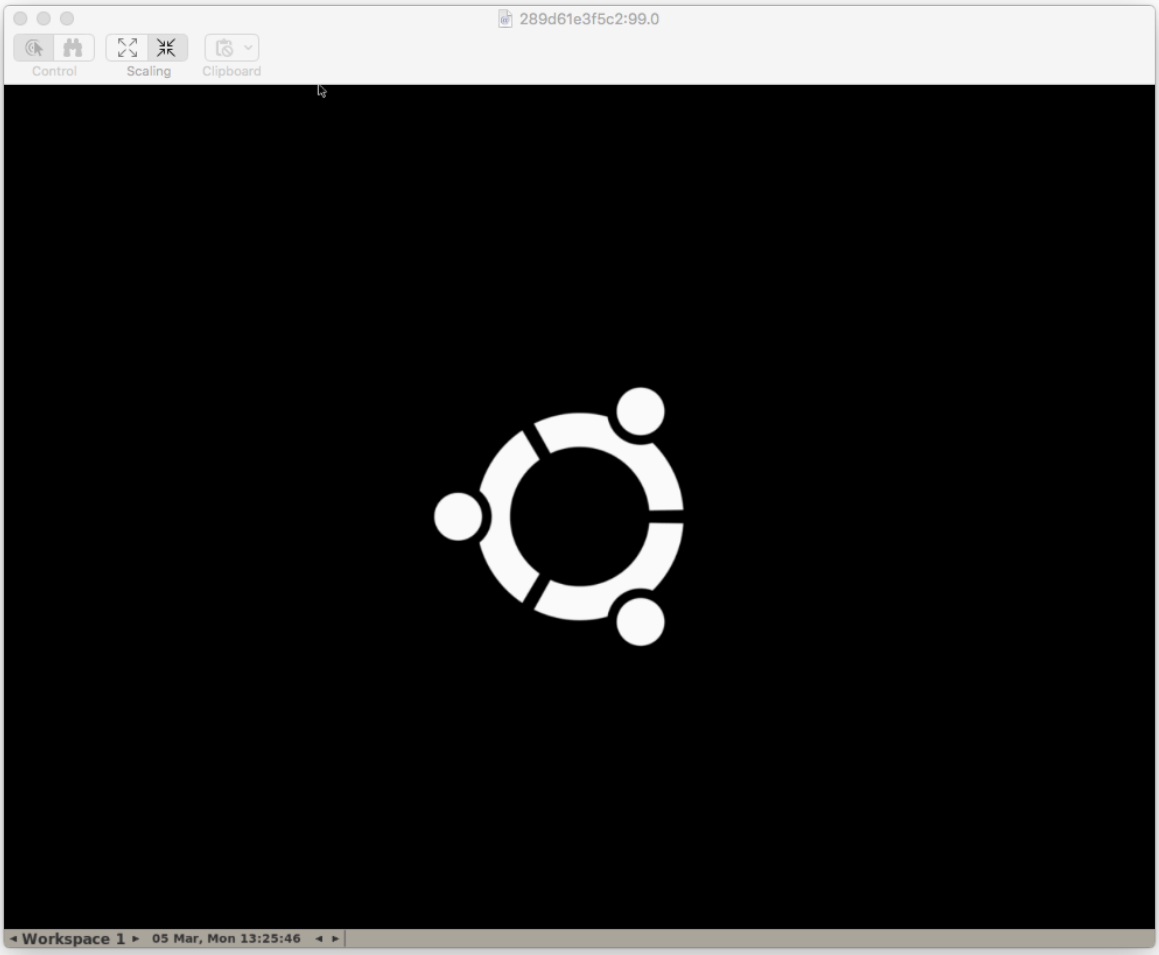
If you use your local firefox or chrome to run tests, you'll see a new browser window open and you can watch the browser run through the tests. If you'd like to do the same thing with the docker containers, you can use the VNC port you set up. Macs have a VNC client built in, called Screen Sharing. (On my Mac it's hidden in System/Library/CoreServices/Applications.) The easiest way to launch it is to search for it in spotlight. Once you've got the Screen Sharing app open, type the address into the popup box. Use localhost:5901 to connect to the chrometest docker container and localhost.5902 to connect to the firefoxtest container.



It will then ask you for the password to connect to the docker container. The Selenium folks, who made the containers, set them up with the password "secret".



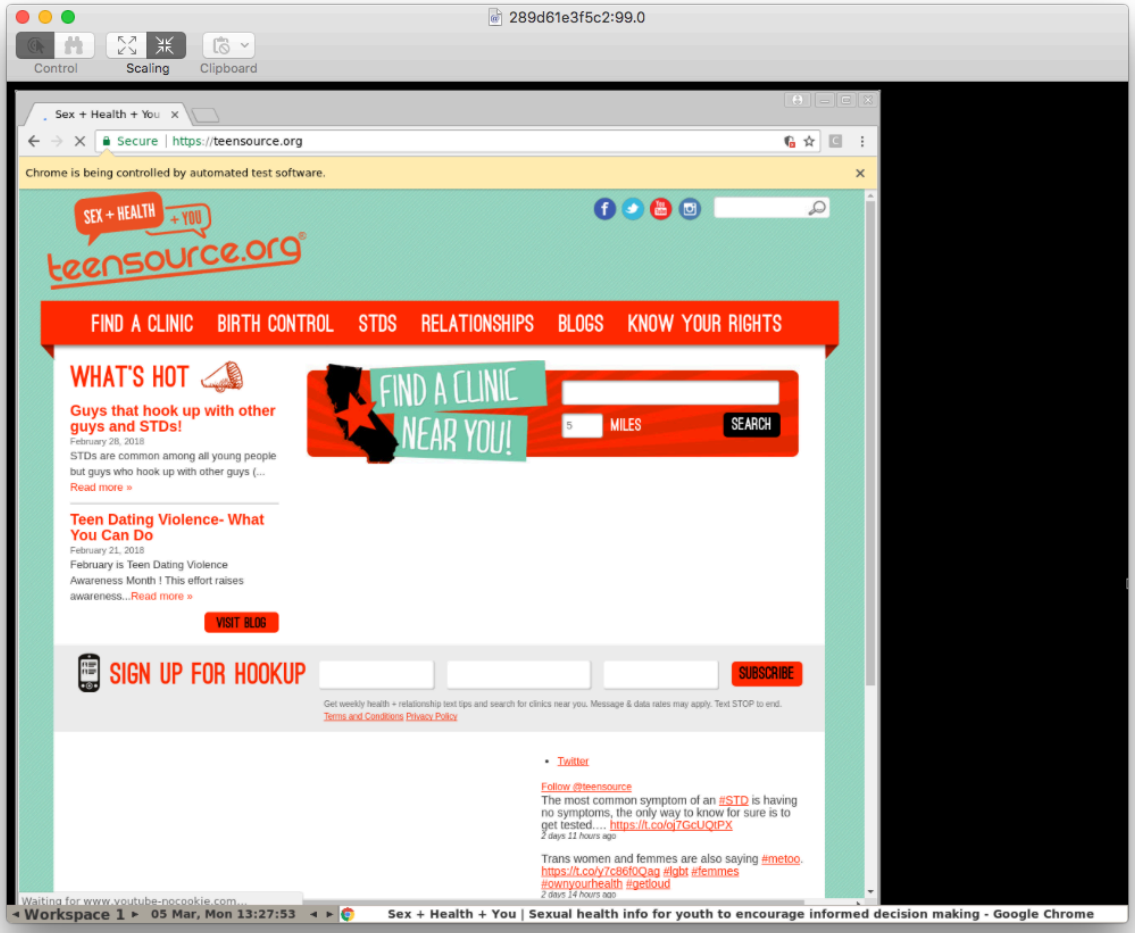
Then you'll be greeted with this exciting screen:



And it'll just sit there, doing nothing...

In order to make things happen, you need to hop over to your terminal and run a test using that docker container. eg `behat -p dockerchrome`.

Then, just like on your local, a browser window will pop up and you'll be able to watch your tests run:



Now that you've got your test environment set up, you can focus on actually *writing* your tests 😛