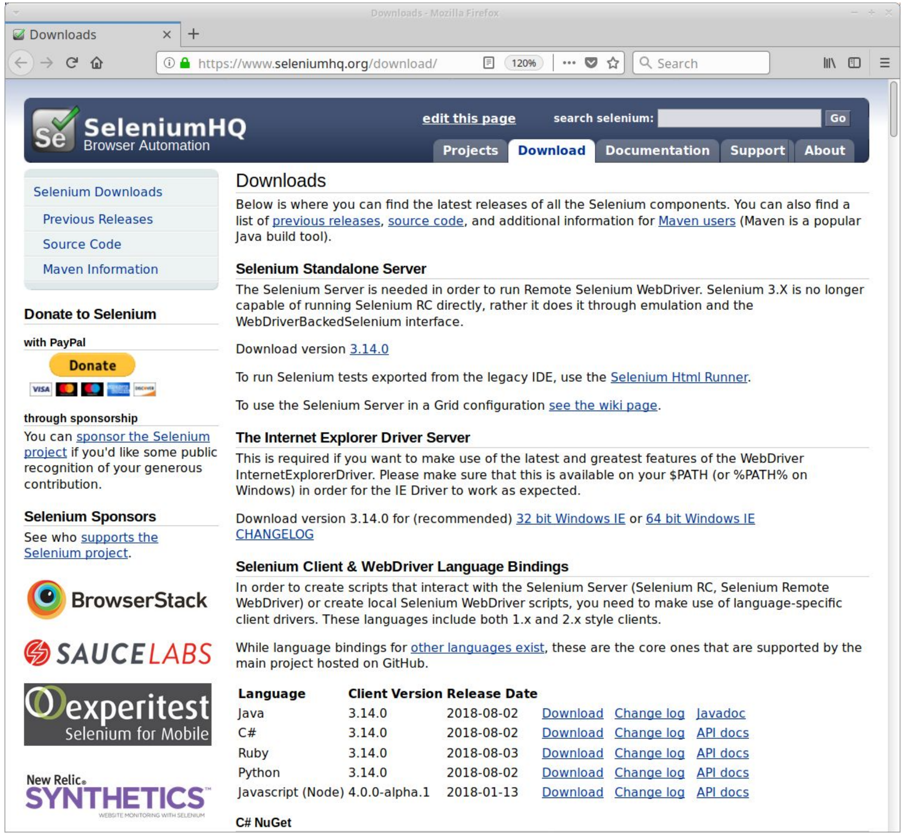
```
$ make test-env-up

$ make test

$ make test-env-down
$ make test-env-up

$ make test

$ make test-env-down
$ make test-env-up

$ make test

$ make test-env-down
```



Downloads - Mozilla Firefox

Downloads

https://www.seleniumhq.org/download/ 120% Search

SeleniumHQ
Browser Automation

edit this page search selenium: Go

Projects Download Documentation Support About

- Selenium Downloads
  - Previous Releases
  - Source Code
  - Maven Information

Donate to Selenium

with PayPal

Donate

through sponsorship

You can sponsor the Selenium project if you'd like some public recognition of your generous contribution.

Selenium Sponsors

See who supports the Selenium project.

BrowserStack

SAUCELABS

experitest Selenium for Mobile

New Relic. SYNTHETICS WEBSITE MONITORING WITH SELENIUM

Downloads

Below is where you can find the latest releases of all the Selenium components. You can also find a list of previous releases, source code, and additional information for Maven users (Maven is a popular Java build tool).

Selenium Standalone Server

The Selenium Server is needed in order to run Remote Selenium WebDriver. Selenium 3.X is no longer capable of running Selenium RC directly, rather it does it through emulation and the WebDriverBackedSelenium interface.

Download version 3.14.0

To run Selenium tests exported from the legacy IDE, use the Selenium Html Runner.

To use the Selenium Server in a Grid configuration see the wiki page.

The Internet Explorer Driver Server

This is required if you want to make use of the latest and greatest features of the WebDriver InternetExplorerDriver. Please make sure that this is available on your $PATH (or %PATH% on Windows) in order for the IE Driver to work as expected.

Download version 3.14.0 for (recommended) 32 bit Windows IE or 64 bit Windows IE
CHANGELOG

Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

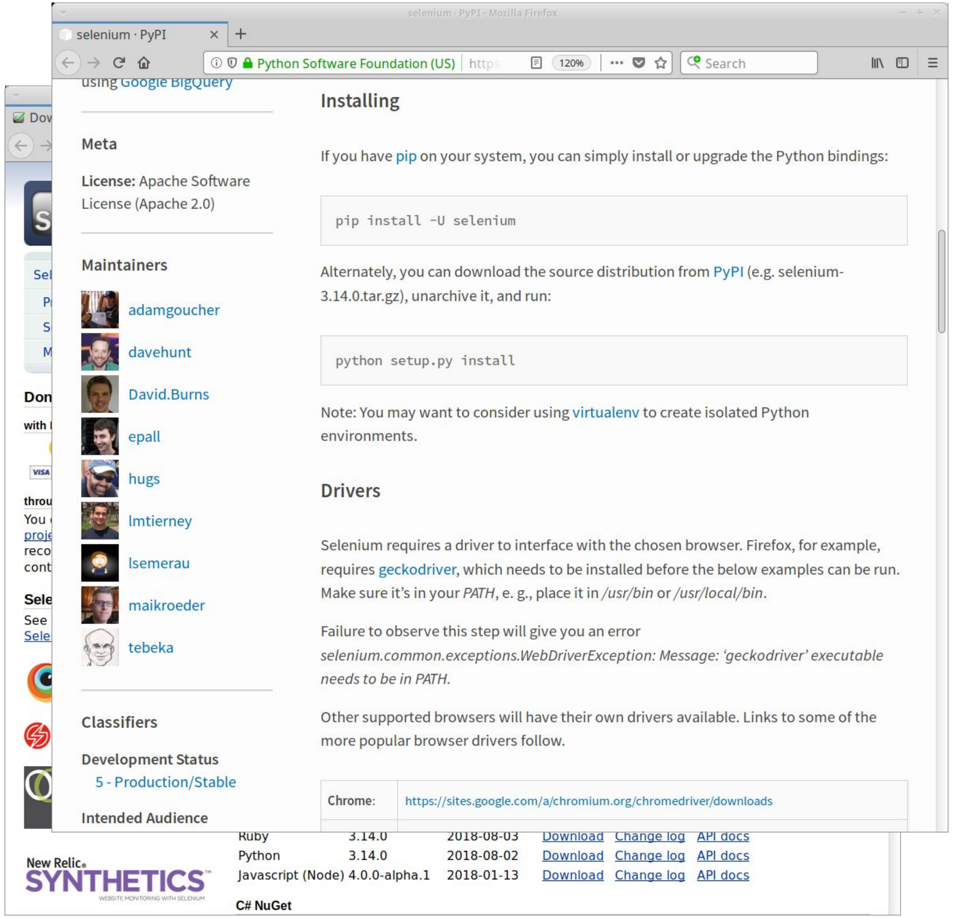
While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on GitHub.

Language	Client Version	Release Date			
Java	3.14.0	2018-08-02	Download	Change log	Javadoc
C#	3.14.0	2018-08-02	Download	Change log	API docs
Ruby	3.14.0	2018-08-03	Download	Change log	API docs
Python	3.14.0	2018-08-02	Download	Change log	API docs
Javascript (Node)	4.0.0-alpha.1	2018-01-13	Download	Change log	API docs

C# NuGet

https://www.seleniumhq.org/ 18 Oct, 2018

```
$ make test-env-up

$ make test

$ make test-env-down
```


selenium · PyPI - Mozilla Firefox
selenium · PyPI
Python Software Foundation (US) https
120%
Search

using Google BigQuery

Meta

License: Apache Software License (Apache 2.0)

Maintainers

adamgoucher
davehunt
David.Burns
epall
hugs
lmtierney
lsemerau
maikroeder
tebeka

Classifiers

Development Status
5 - Production/Stable

Intended Audience

Installing

If you have pip on your system, you can simply install or upgrade the Python bindings:

```
pip install -U selenium
```

Alternately, you can download the source distribution from PyPI (e.g. selenium-3.14.0.tar.gz), unarchive it, and run:

```
python setup.py install
```

Note: You may want to consider using virtualenv to create isolated Python environments.

Drivers

Selenium requires a driver to interface with the chosen browser. Firefox, for example, requires geckodriver, which needs to be installed before the below examples can be run. Make sure it's in your *PATH*, e. g., place it in */usr/bin* or */usr/local/bin*.

Failure to observe this step will give you an error *selenium.common.exceptions.WebDriverException: Message: 'geckodriver' executable needs to be in PATH.*

Other supported browsers will have their own drivers available. Links to some of the more popular browser drivers follow.

Chrome:	https://sites.google.com/a/chromium.org/chromedriver/downloads

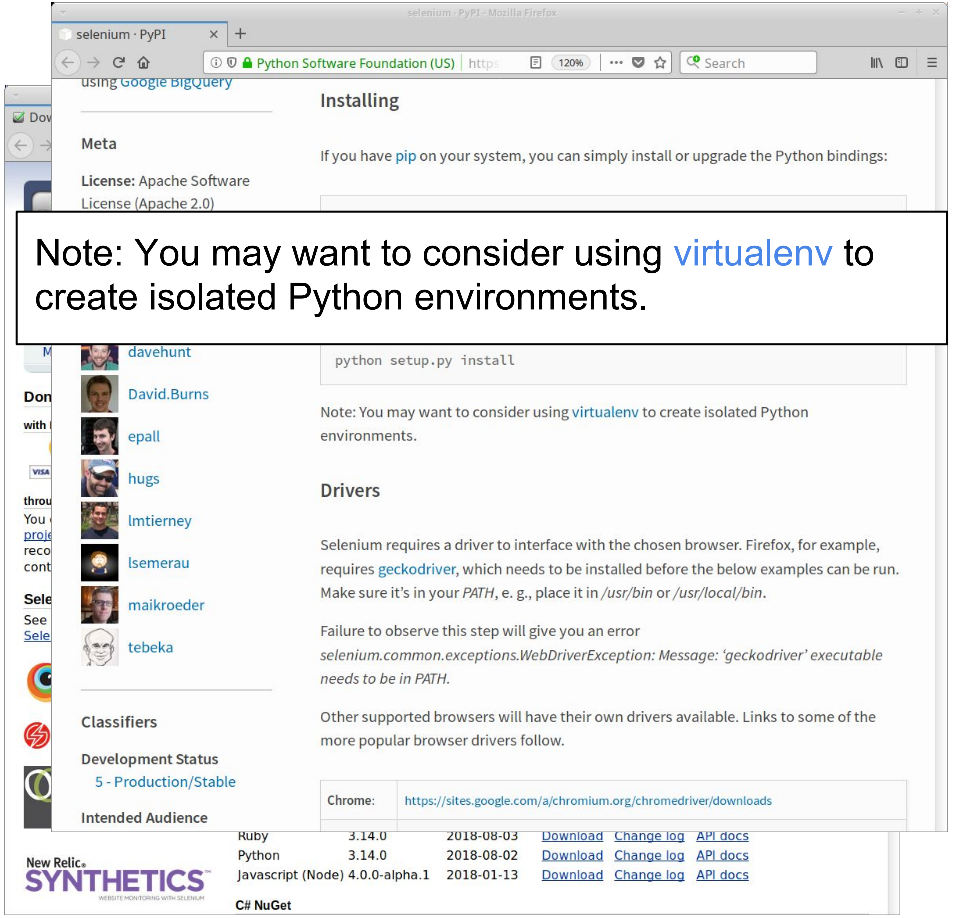
Ruby	3.14.0	2018-08-03	Download	Change log	API docs
Python	3.14.0	2018-08-02	Download	Change log	API docs
Javascript (Node)	4.0.0-alpha.1	2018-01-13	Download	Change log	API docs

C# NuGet

New Relic SYNTHETICS

https://pypi.org/project/selenium/ 18 Oct, 2018

```
$ make test-env-up

$ make test

$ make test-env-down
```



selenium · PyPI — Mozilla Firefox

Python Software Foundation (US) https 120% Search

using Google BigQuery

Meta

License: Apache Software License (Apache 2.0)

davehunt
David.Burns
epall
hugs
lmtierney
lsemerau
maikroeder
tebeka

Classifiers

Development Status
5 - Production/Stable

Intended Audience

Installing

If you have pip on your system, you can simply install or upgrade the Python bindings:

```
python setup.py install
```

Note: You may want to consider using virtualenv to create isolated Python environments.

Drivers

Selenium requires a driver to interface with the chosen browser. Firefox, for example, requires geckodriver, which needs to be installed before the below examples can be run. Make sure it's in your *PATH*, e. g., place it in */usr/bin* or */usr/local/bin*.

Failure to observe this step will give you an error *selenium.common.exceptions.WebDriverException: Message: 'geckodriver' executable needs to be in PATH.*

Other supported browsers will have their own drivers available. Links to some of the more popular browser drivers follow.

Chrome:	https://sites.google.com/a/chromium.org/chromedriver/downloads

Ruby	3.14.0	2018-08-03	Download	Change log	API docs
Python	3.14.0	2018-08-02	Download	Change log	API docs
Javascript (Node)	4.0.0-alpha.1	2018-01-13	Download	Change log	API docs

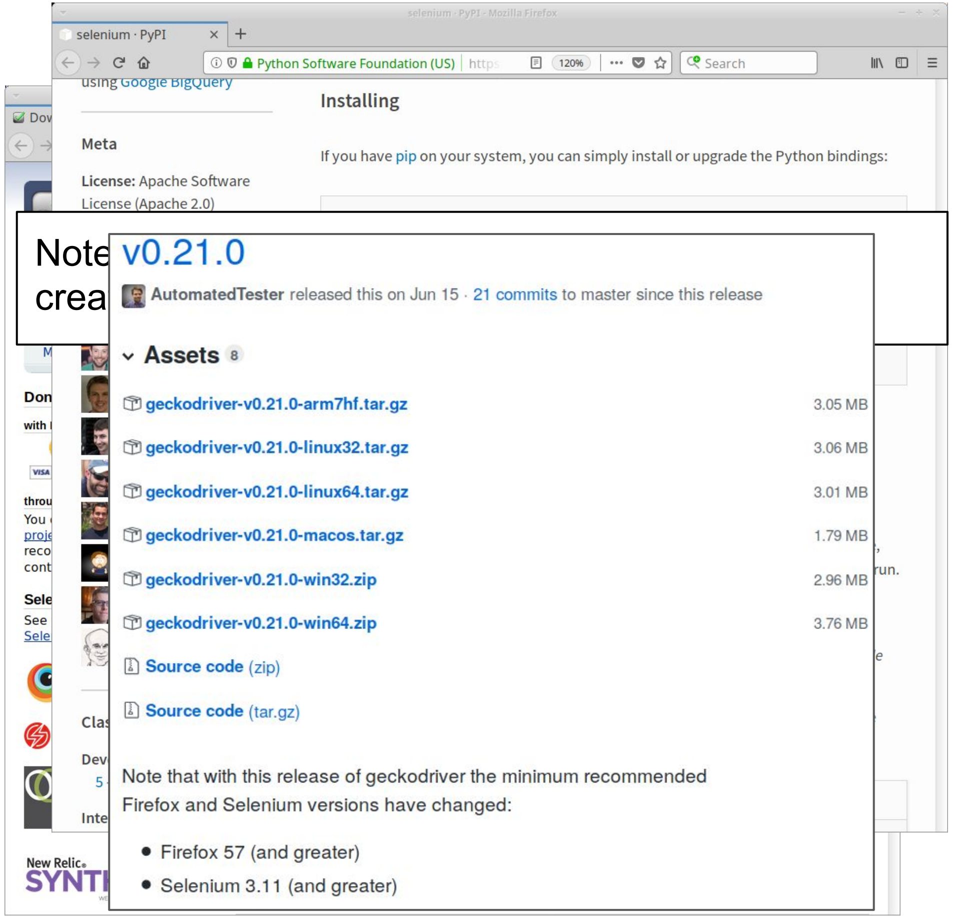
C# NuGet

New Relic. SYNTHETICS

Note: You may want to consider using virtualenv to create isolated Python environments.

```
$ make test-env-up

$ make test

$ make test-env-down
```

https://www.seleniumhq.org/ 18 Oct, 2018



using Google BigQuery

If you have pip on your system, you can simply install or upgrade the Python bindings:

License: Apache Software License (Apache 2.0)

Note
crea

v0.21.0

AutomatedTester released this on Jun 15 · 21 commits to master since this release

Assets 8

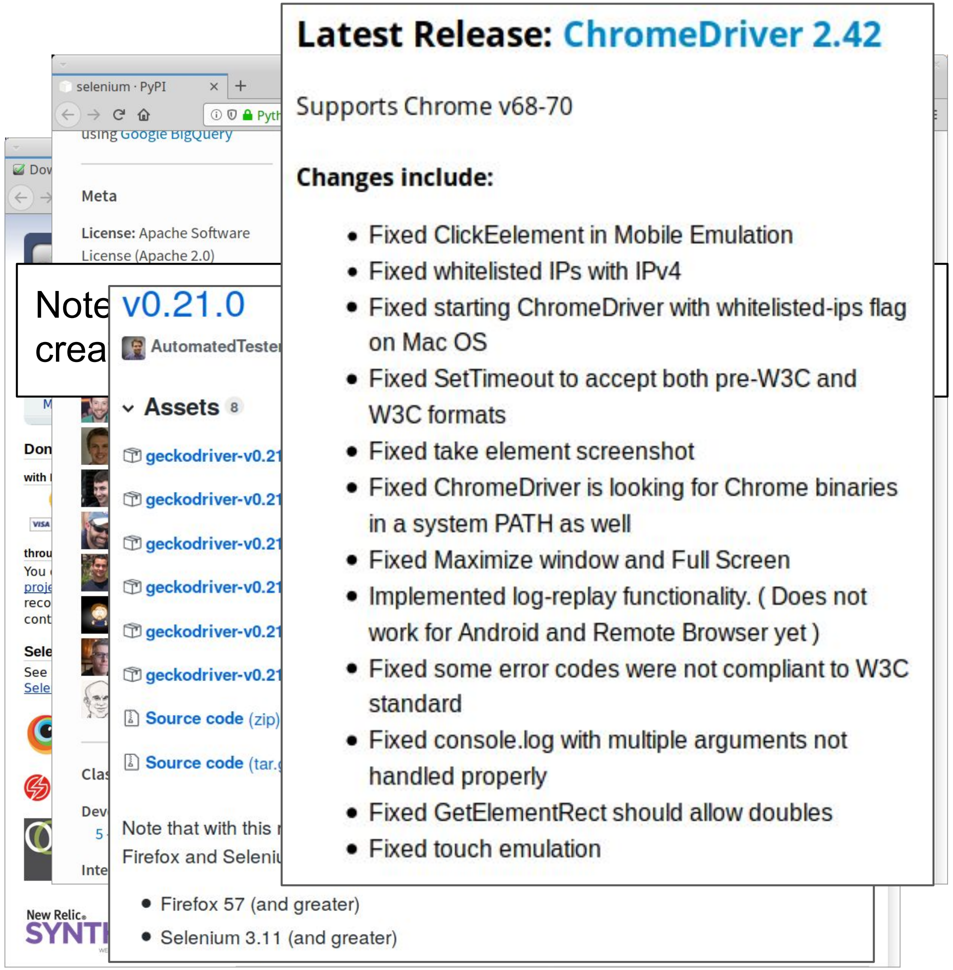
geckodriver-v0.21.0-arm7hf.tar.gz	3.05 MB
geckodriver-v0.21.0-linux32.tar.gz	3.06 MB
geckodriver-v0.21.0-linux64.tar.gz	3.01 MB
geckodriver-v0.21.0-macos.tar.gz	1.79 MB
geckodriver-v0.21.0-win32.zip	2.96 MB
geckodriver-v0.21.0-win64.zip	3.76 MB
Source code (zip)	
Source code (tar.gz)	

Note that with this release of geckodriver the minimum recommended Firefox and Selenium versions have changed:

- Firefox 57 (and greater)
- Selenium 3.11 (and greater)

https://github.com/mozilla/geckodriver/releases / 18 Oct, 2018

```
$ make test-env-up

$ make test

$ make test-env-down
```



selenium · PyPI

using Google BigQuery

Meta

License: Apache Software License (Apache 2.0)

Note
crea

v0.21.0

AutomatedTester

Assets 8

geckodriver-v0.21

geckodriver-v0.21

geckodriver-v0.21

geckodriver-v0.21

geckodriver-v0.21

geckodriver-v0.21

Source code (zip)

Source code (tar.

Note that with this r
Firefox and Seleniu

- Firefox 57 (and greater)
- Selenium 3.11 (and greater)

Latest Release: ChromeDriver 2.42

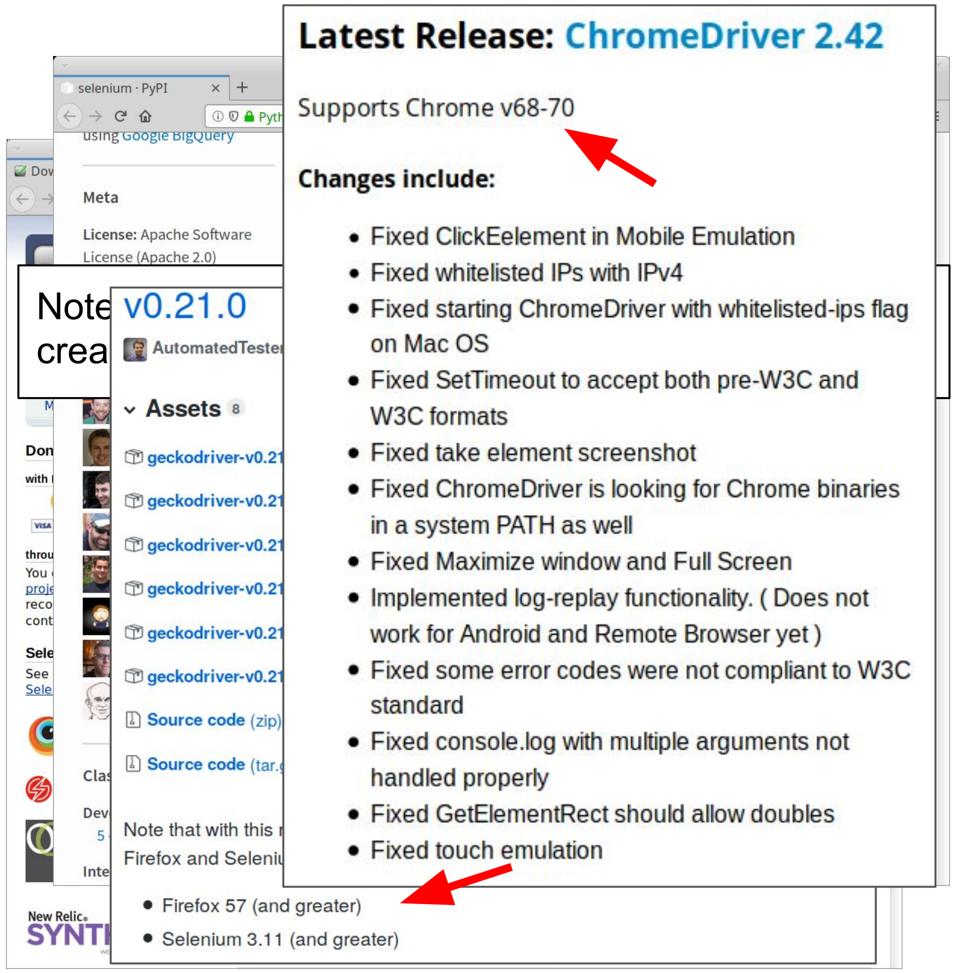
Supports Chrome v68-70

Changes include:

- Fixed ClickEelement in Mobile Emulation
- Fixed whitelisted IPs with IPv4
- Fixed starting ChromeDriver with whitelisted-ips flag on Mac OS
- Fixed SetTimeout to accept both pre-W3C and W3C formats
- Fixed take element screenshot
- Fixed ChromeDriver is looking for Chrome binaries in a system PATH as well
- Fixed Maximize window and Full Screen
- Implemented log-replay functionality. (Does not work for Android and Remote Browser yet)
- Fixed some error codes were not compliant to W3C standard
- Fixed console.log with multiple arguments not handled properly
- Fixed GetElementRect should allow doubles
- Fixed touch emulation

http://chromedriver.chromium.org/ 18 Oct, 2018

```
$ make test-env-up

$ make test

$ make test-env-down
```



selenium · PyPI

using Google BigQuery

Meta

License: Apache Software License (Apache 2.0)

Note
crea

v0.21.0

AutomatedTester

Assets 8

geckodriver-v0.21
geckodriver-v0.21
geckodriver-v0.21
geckodriver-v0.21
geckodriver-v0.21
geckodriver-v0.21
Source code (zip)
Source code (tar.

Note that with this
Firefox and Seleni

- Firefox 57 (and greater)
- Selenium 3.11 (and greater)

Latest Release: ChromeDriver 2.42

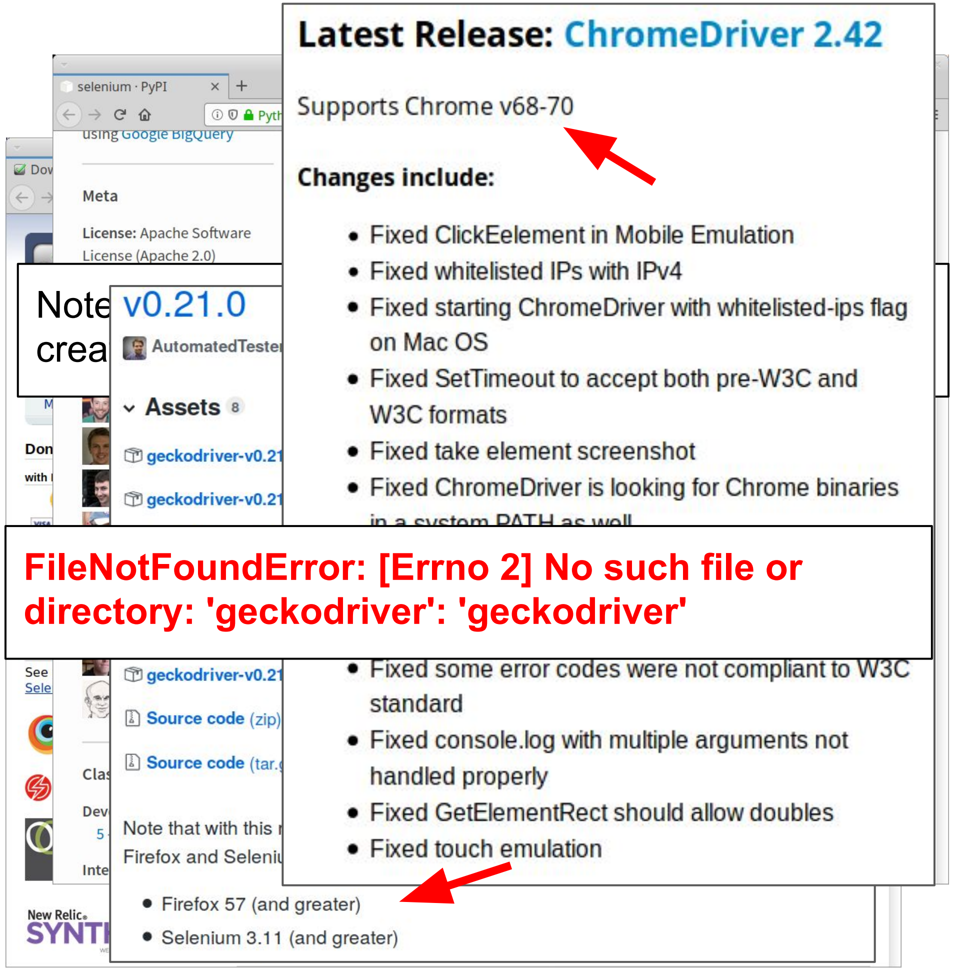
Supports Chrome v68-70

Changes include:

- Fixed ClickEelement in Mobile Emulation
- Fixed whitelisted IPs with IPv4
- Fixed starting ChromeDriver with whitelisted-ips flag on Mac OS
- Fixed SetTimeout to accept both pre-W3C and W3C formats
- Fixed take element screenshot
- Fixed ChromeDriver is looking for Chrome binaries in a system PATH as well
- Fixed Maximize window and Full Screen
- Implemented log-replay functionality. (Does not work for Android and Remote Browser yet)
- Fixed some error codes were not compliant to W3C standard
- Fixed console.log with multiple arguments not handled properly
- Fixed GetElementRect should allow doubles
- Fixed touch emulation

```
$ make test-env-up

$ make test

$ make test-env-down
```

http://chromedriver.chromium.org/ 18 Oct, 2018



Latest Release: ChromeDriver 2.42

Supports Chrome v68-70

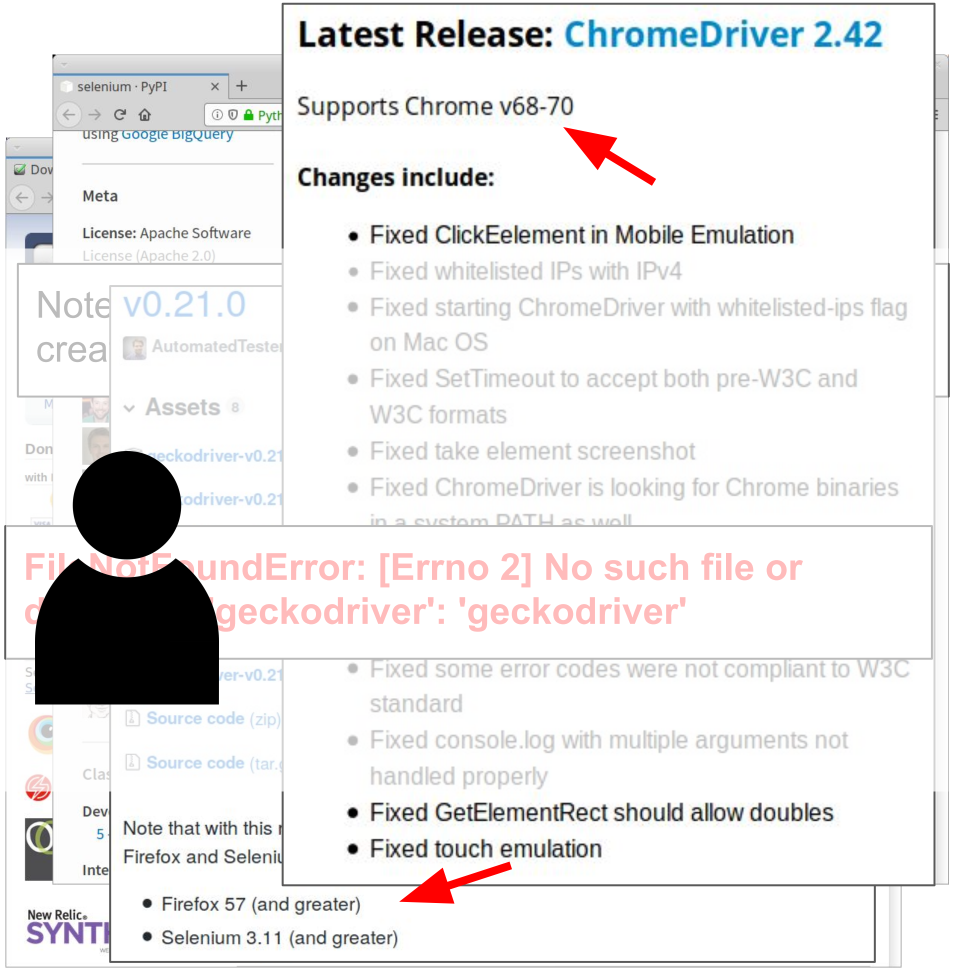
Changes include:

- Fixed ClickEelement in Mobile Emulation
- Fixed whitelisted IPs with IPv4
- Fixed starting ChromeDriver with whitelisted-ips flag on Mac OS
- Fixed SetTimeout to accept both pre-W3C and W3C formats
- Fixed take element screenshot
- Fixed ChromeDriver is looking for Chrome binaries in a system PATH as well
- Fixed some error codes were not compliant to W3C standard
- Fixed console.log with multiple arguments not handled properly
- Fixed GetElementRect should allow doubles
- Fixed touch emulation

- Firefox 57 (and greater)
- Selenium 3.11 (and greater)

FileNotFoundError: [Errno 2] No such file or directory: 'geckodriver': 'geckodriver'

```
$ make test-env-up

$ make test

$ make test-env-down
```



Latest Release: ChromeDriver 2.42

Supports Chrome v68-70

Changes include:

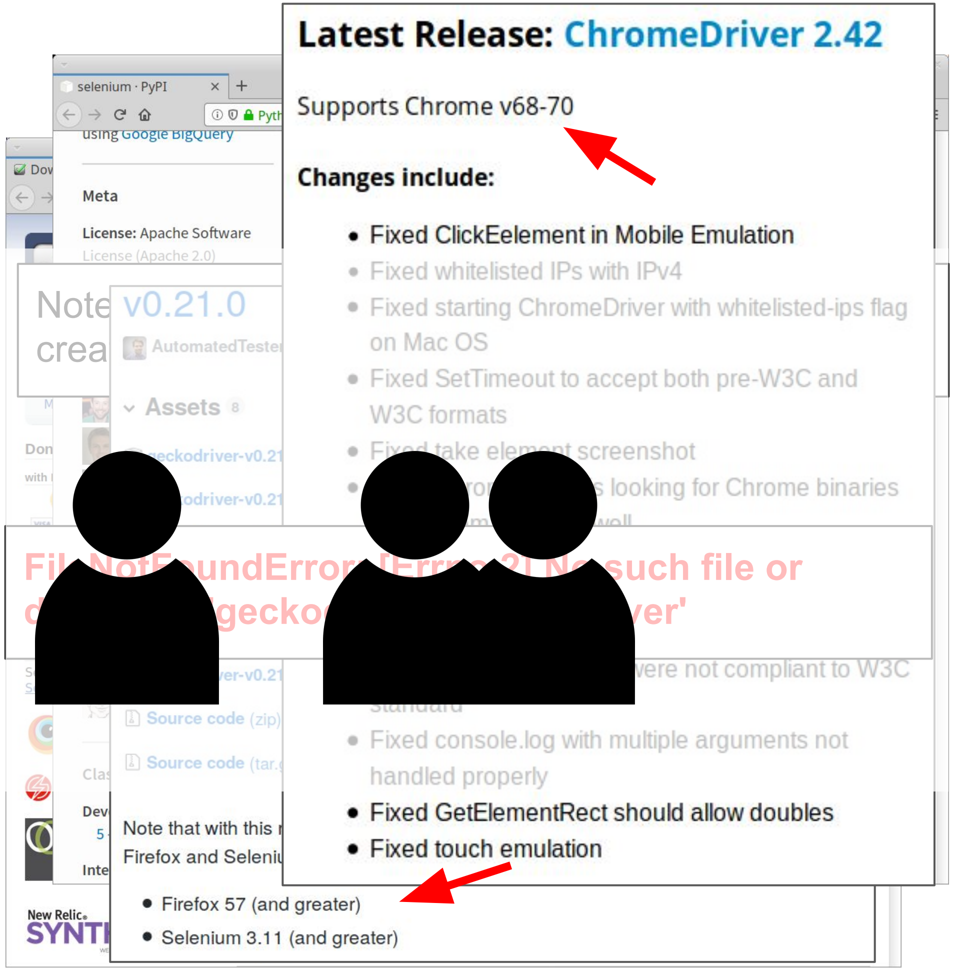
- Fixed ClickEelement in Mobile Emulation
- Fixed whitelisted IPs with IPv4
- Fixed starting ChromeDriver with whitelisted-ips flag on Mac OS
- Fixed SetTimeout to accept both pre-W3C and W3C formats
- Fixed take element screenshot
- Fixed ChromeDriver is looking for Chrome binaries in a system PATH as well
- Fixed some error codes were not compliant to W3C standard
- Fixed console.log with multiple arguments not handled properly
- Fixed GetElementRect should allow doubles
- Fixed touch emulation

```
FileNotFoundError: [Errno 2] No such file or directory: 'geckodriver': 'geckodriver'
```

Note that with this r
Firefox and Seleniu

- Firefox 57 (and greater)
- Selenium 3.11 (and greater)

```
$ make test-env-up

$ make test

$ make test-env-down
```



Latest Release: ChromeDriver 2.42

Supports Chrome v68-70

Changes include:

- Fixed ClickEelement in Mobile Emulation
- Fixed whitelisted IPs with IPv4
- Fixed starting ChromeDriver with whitelisted-ips flag on Mac OS
- Fixed SetTimeout to accept both pre-W3C and W3C formats
- Fixed take element screenshot
- s looking for Chrome binaries
- were not compliant to W3C standard
- Fixed console.log with multiple arguments not handled properly
- Fixed GetElementRect should allow doubles
- Fixed touch emulation

selenium · PyPI

using Google BigQuery

Meta

License: Apache Software License (Apache 2.0)

v0.21.0

AutomatedTeste

Assets 8

Source code (zip)

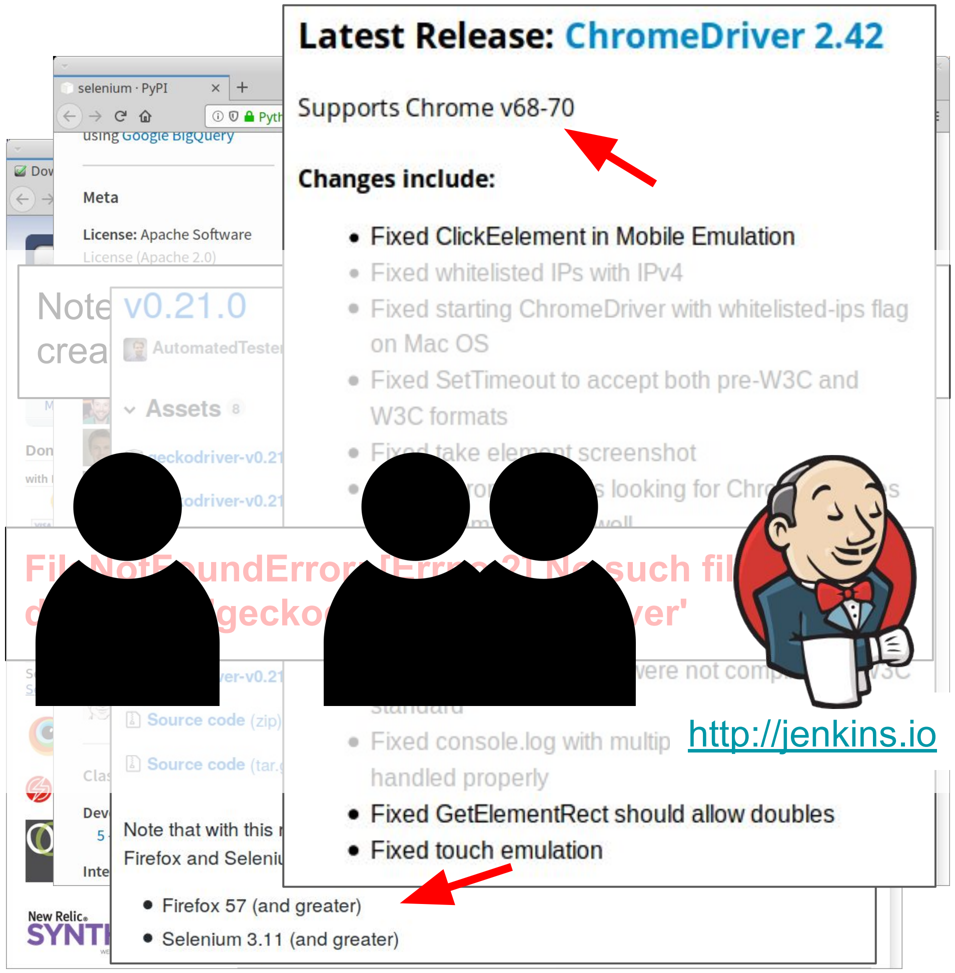
Source code (tar.

Note that with this r
Firefox and Seleni

- Firefox 57 (and greater)
- Selenium 3.11 (and greater)

FileNotFoundError: [Errno 2] No such file or d geckod er'

```
$ make test-env-up

$ make test

$ make test-env-down
```



Latest Release: ChromeDriver 2.42

Supports Chrome v68-70

Changes include:

- Fixed ClickEelement in Mobile Emulation
- Fixed whitelisted IPs with IPv4
- Fixed starting ChromeDriver with whitelisted-ips flag on Mac OS
- Fixed SetTimeout to accept both pre-W3C and W3C formats
- Fixed take element screenshot
- Fixed console.log with multip handled properly
- Fixed GetElementRect should allow doubles
- Fixed touch emulation

selenium · PyPI

using Google BigQuery

Meta

License: Apache Software License (Apache 2.0)

v0.21.0

AutomatedTester

Assets 8

Source code (zip)

Source code (tar.gz)

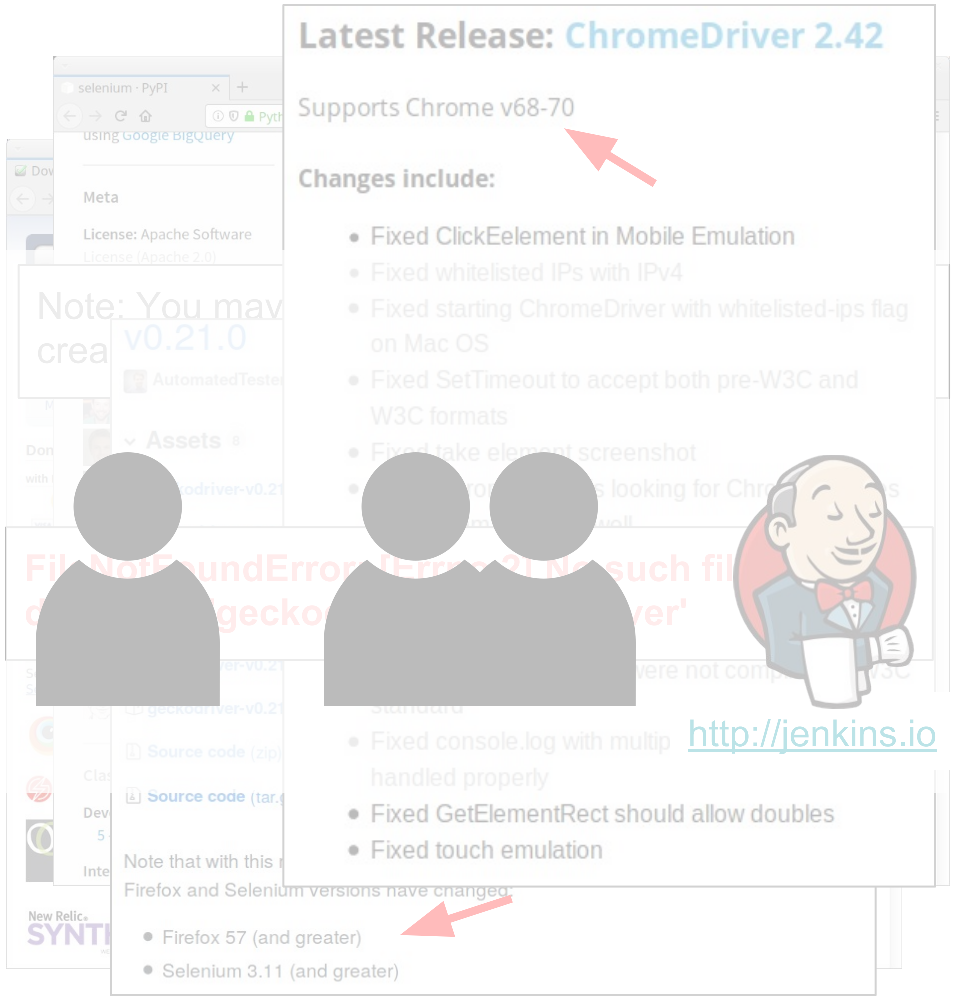
Note that with this r Firefox and Seleni

- Firefox 57 (and greater)
- Selenium 3.11 (and greater)

FileNotFoundError: [Errno 2] No such file or directory: 'geckodriver'

http://jenkins.io

```
$ make test-env-up

$ make test

$ make test-env-down
```



Latest Release: ChromeDriver 2.42

Supports Chrome v68-70

Changes include:

- Fixed ClickEelement in Mobile Emulation
- Fixed whitelisted IPs with IPv4
- Fixed starting ChromeDriver with whitelisted-ips flag on Mac OS
- Fixed SetTimeout to accept both pre-W3C and W3C formats
- Fixed take element screenshot
- Fixed GetElementRect should allow doubles
- Fixed touch emulation

FileNotFoundError: [Errno 2] No such file or directory: 'geckodriver'

v0.21.0

Note that with this release the minimum recommended Firefox and Selenium versions have changed:

- Firefox 57 (and greater)
- Selenium 3.11 (and greater)

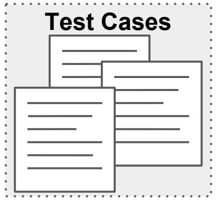
http://jenkins.io

```
$ make test-env-up

$ make test

$ make test-env-down
```

Jump Starting Your Testing with Selenium Grid Docker Containers, Selene, and pytest

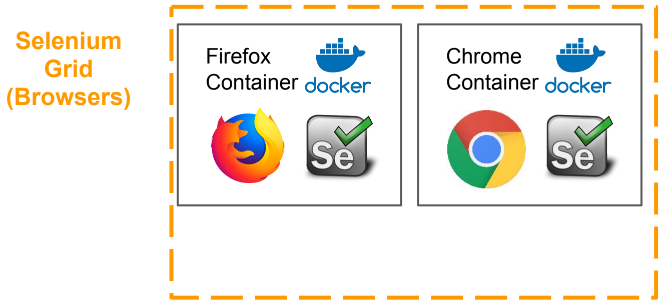
Derrick Kearney
tellldsk@gmail.com

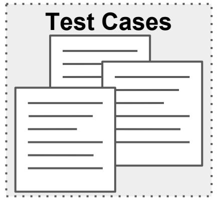
https://github.com/dskard/seleniumconf2018



Command Line

```
$
```

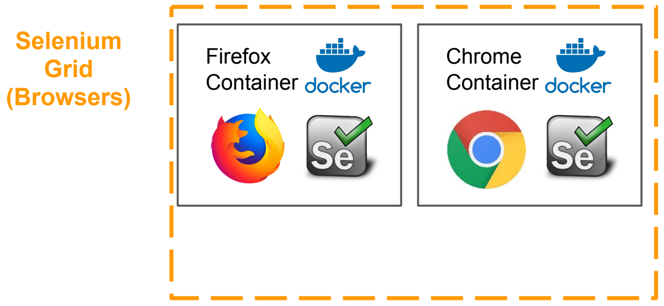


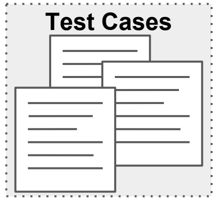


Command Line

```
$ make test-env-up
```

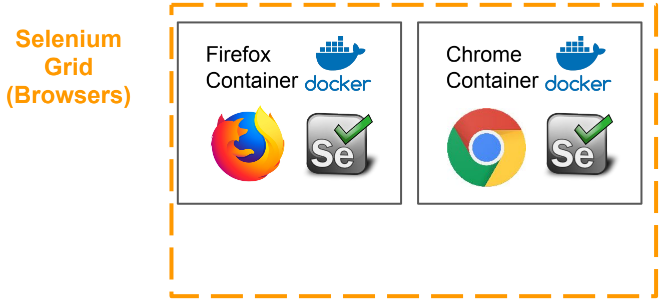
More on Docker:
http://docker.com/resources/what-container

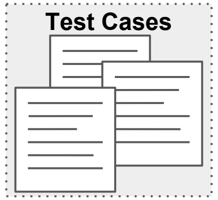




Command Line

```
$ make test-env-up
```





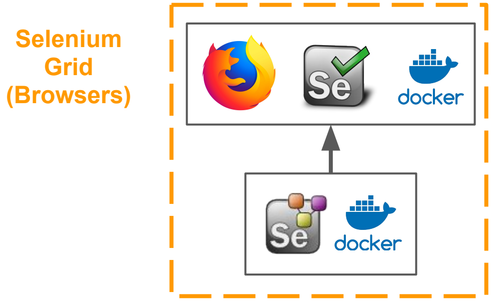
More on Docker:
http://docker.com/resources/what-container

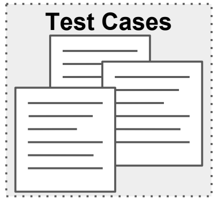
SeleniumHQ docker-selenium:
http://github.com/SeleniumHQ/docker-selenium

- Selenium Grid standalone images
- Selenium Grid hub & node images

Command Line

```
$ make test-env-up
```





More on Docker:
http://docker.com/resources/what-container

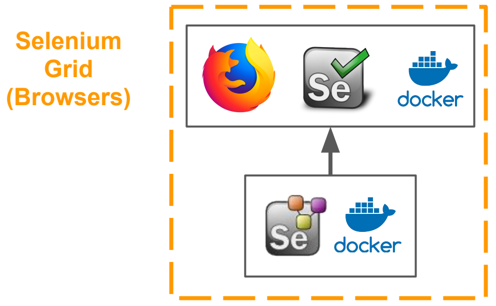
SeleniumHQ docker-selenium:
http://github.com/SeleniumHQ/docker-selenium

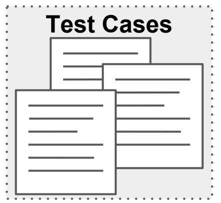
- Selenium Grid standalone images
- Selenium Grid hub & node images

Command Line

```
$ make test-env-up
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.





More on Docker:
http://docker.com/resources/what-container

SeleniumHQ docker-selenium:
http://github.com/SeleniumHQ/docker-selenium
- Selenium Grid standalone images
- Selenium Grid hub & node images

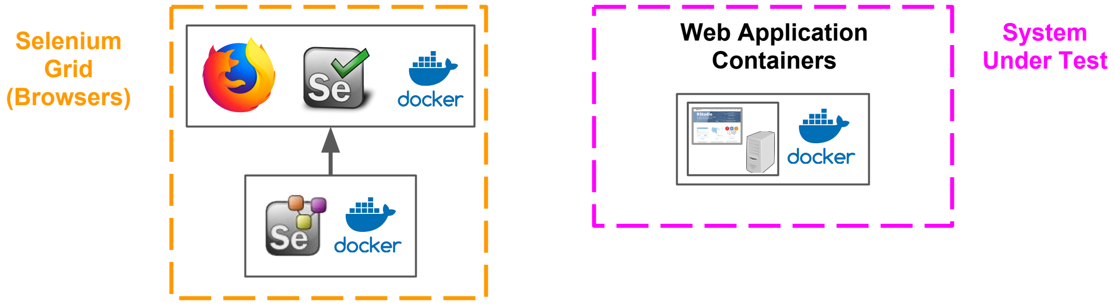
Store the setup in a Docker Compose YML

docker/grid/docker-compose.yml

Command Line

```
$ make test-env-up
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



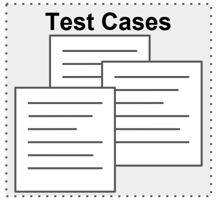
More on Docker:
http://docker.com/resources/what-container

SeleniumHQ docker-selenium:
http://github.com/SeleniumHQ/docker-selenium

- Selenium Grid standalone images
- Selenium Grid hub & node images

Store the setup in a Docker Compose YML

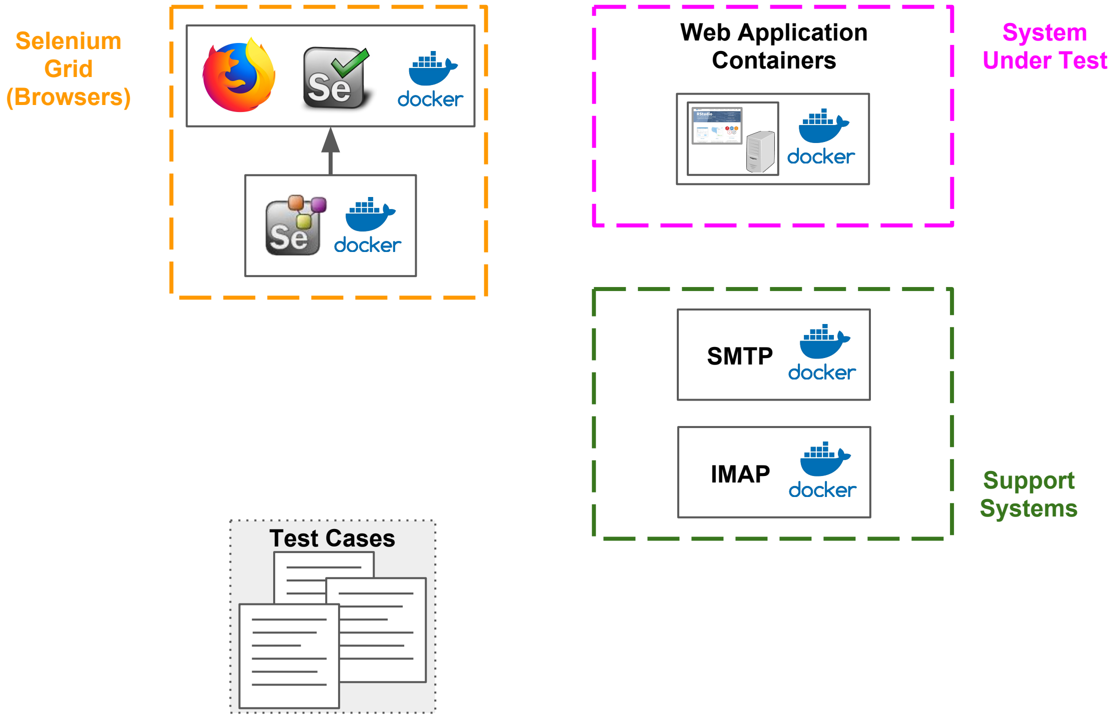
docker/grid/docker-compose.yml



Command Line

```
$ make test-env-up
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



More on Docker:
http://docker.com/resources/what-container

SeleniumHQ docker-selenium:
http://github.com/SeleniumHQ/docker-selenium

- Selenium Grid standalone images
- Selenium Grid hub & node images

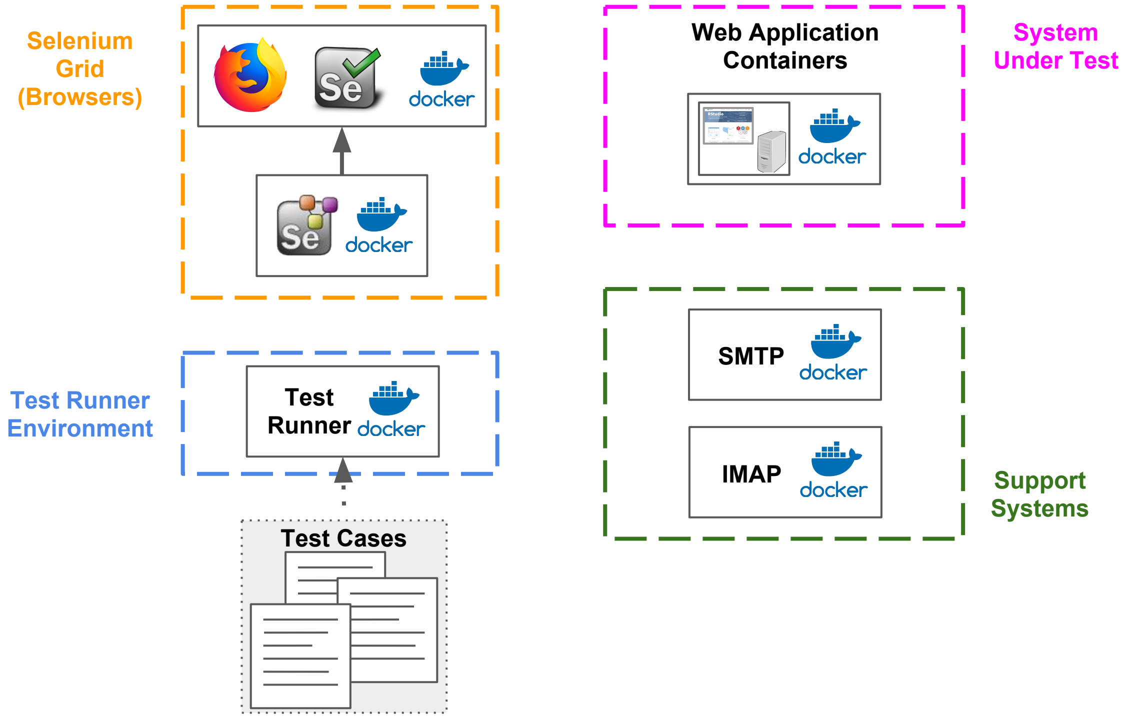
Store the setup in a Docker Compose YML

docker/grid/docker-compose.yml

Command Line

```
$ make test-env-up
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.

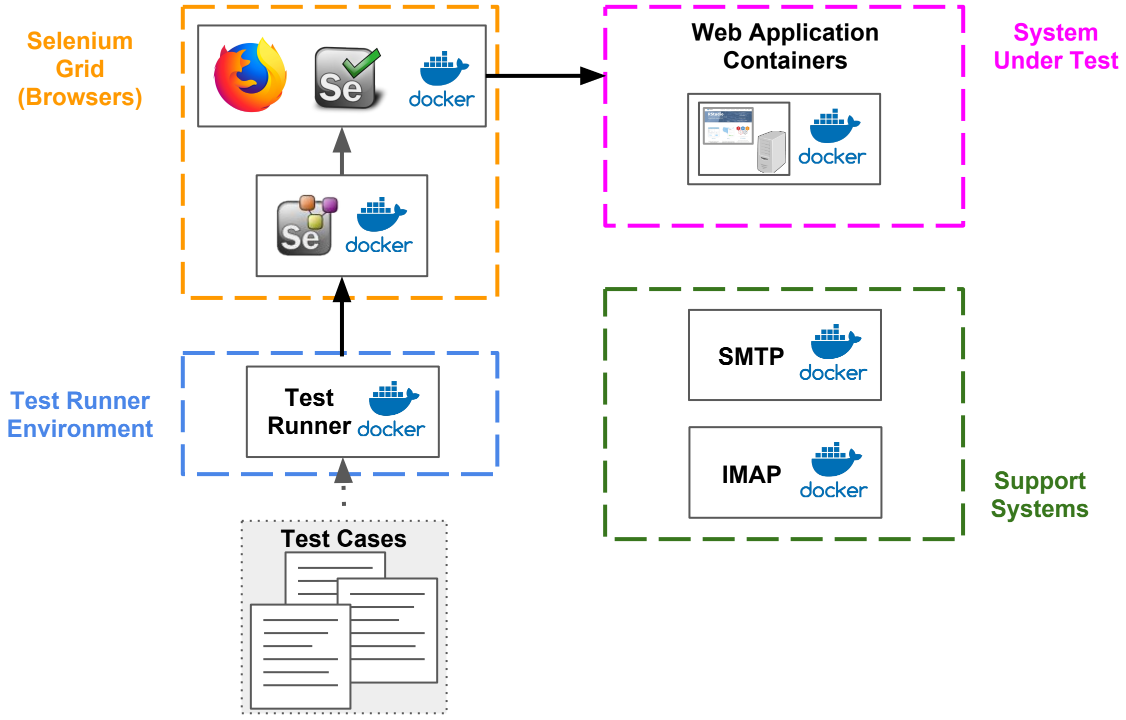


Test Runner Environment:
dskard/tew:0.1.0

- Runs on same Docker network as Test Environment
- Access test cases via shared mount
- Contains software to run test cases
  - Python3 interpreter & debugger
  - Selenium client libraries
  - *pytest* test runner
  - bash, curl, wget

Command Line

```
$ make test-env-up
$ make test
```



Test Runner Environment:
dskard/tew:0.1.0

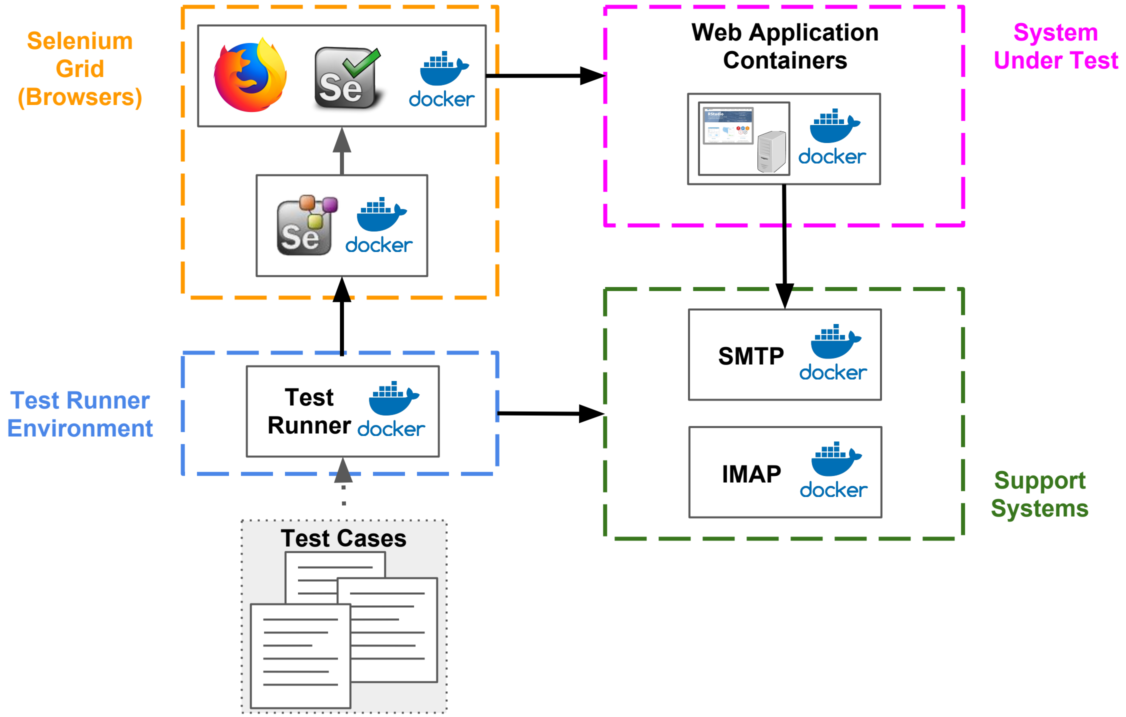
- Runs on same Docker network as Test Environment
- Access test cases via shared mount
- Contains software to run test cases
  - Python3 interpreter & debugger
  - Selenium client libraries
  - *pytest* test runner
  - bash, curl, wget

Test cases use web browsers from remote Selenium Grid.

Command Line

```
$ make test-env-up
$ make test
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



Test Runner Environment:
dskard/tew:0.1.0

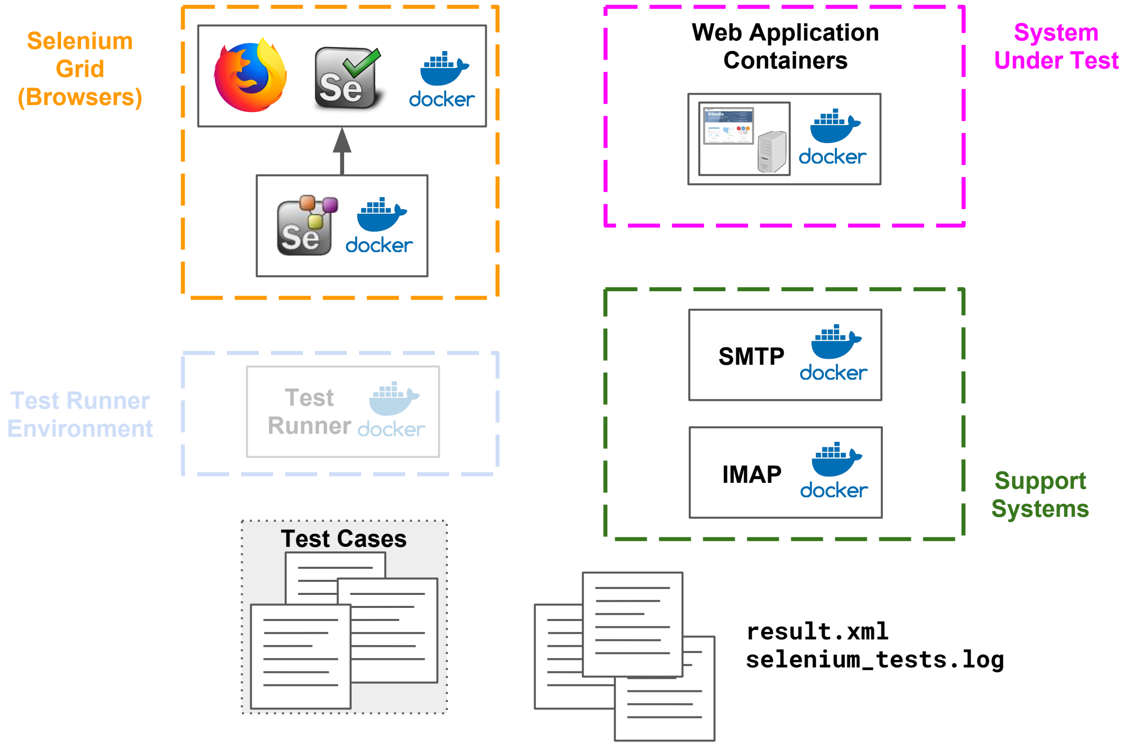
- Runs on same Docker network as Test Environment
- Access test cases via shared mount
- Contains software to run test cases
  - Python3 interpreter & debugger
  - Selenium client libraries
  - *pytest* test runner
  - bash, curl, wget

Test cases use web browsers from remote Selenium Grid.

Command Line

```
$ make test-env-up
$ make test
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



Test Runner Environment:
dskard/tew:0.1.0

- Runs on same Docker network as Test Environment
- Access test cases via shared mount
- Contains software to run test cases
  - Python3 interpreter & debugger
  - Selenium client libraries
  - *pytest* test runner
  - bash, curl, wget

Test cases use web browsers from remote Selenium Grid.

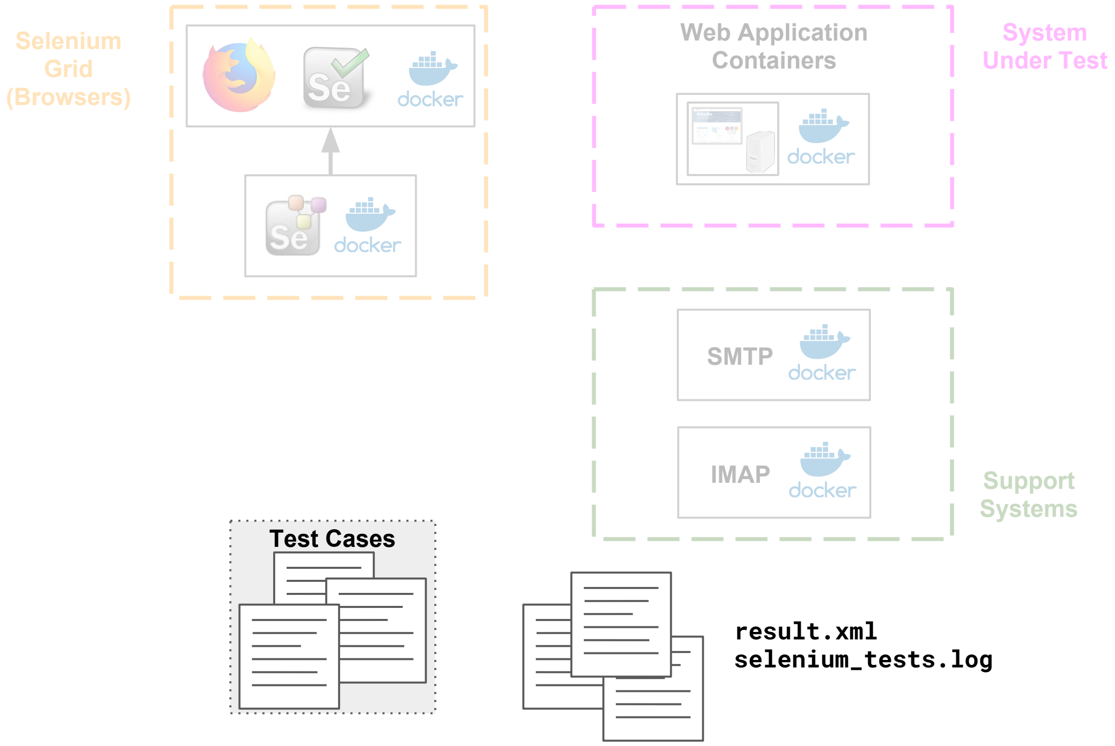
Container shuts down after test runner process exits.

Command Line

```
$ make test-env-up
$ make test
```

Shut down the remaining containers:

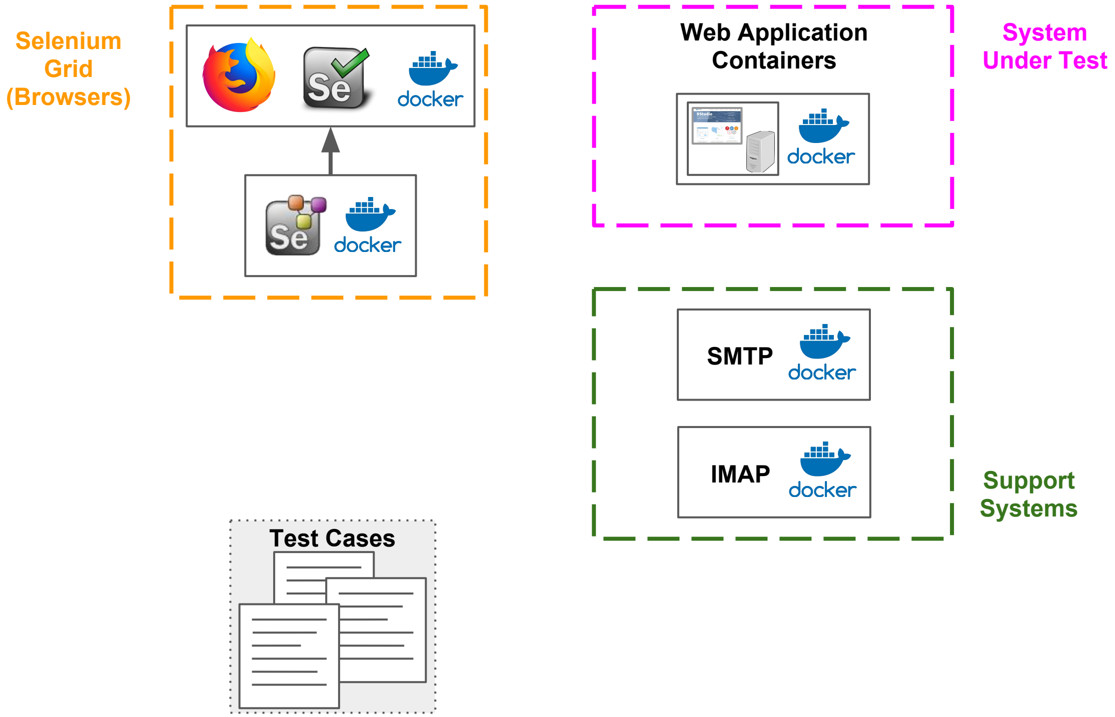
- Selenium Grid
- System Under Test
- Support Systems



Command Line

```
$ make test-env-up
$ make test
$ make test-env-down
```

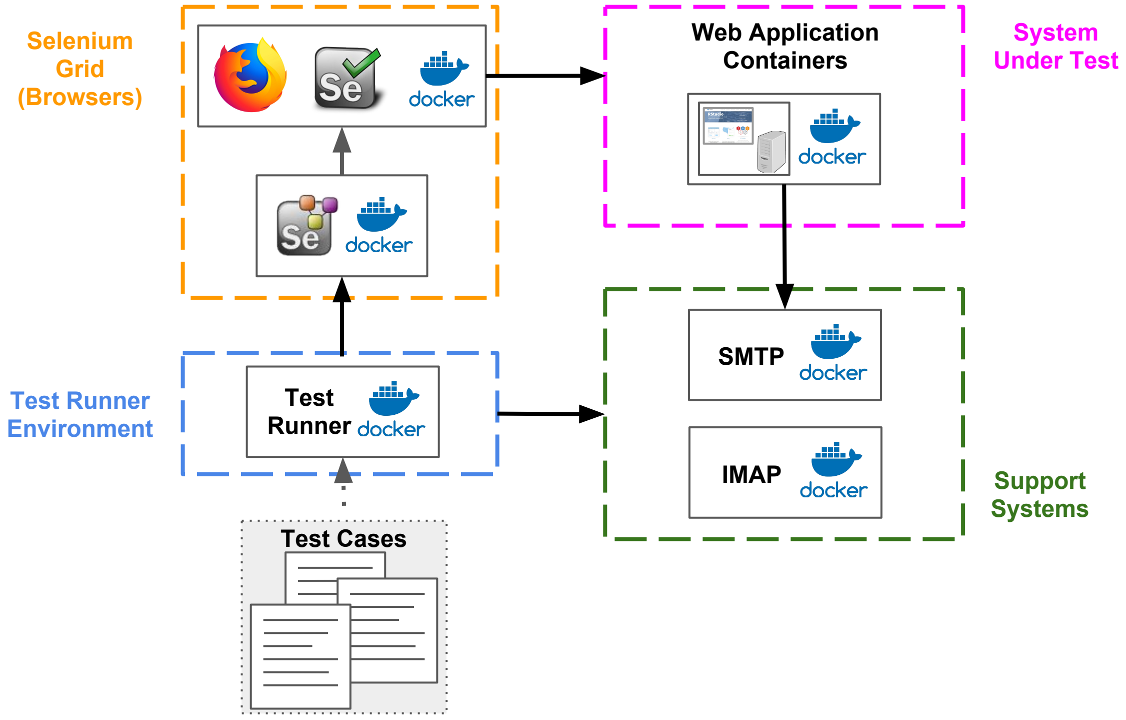
The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



Command Line

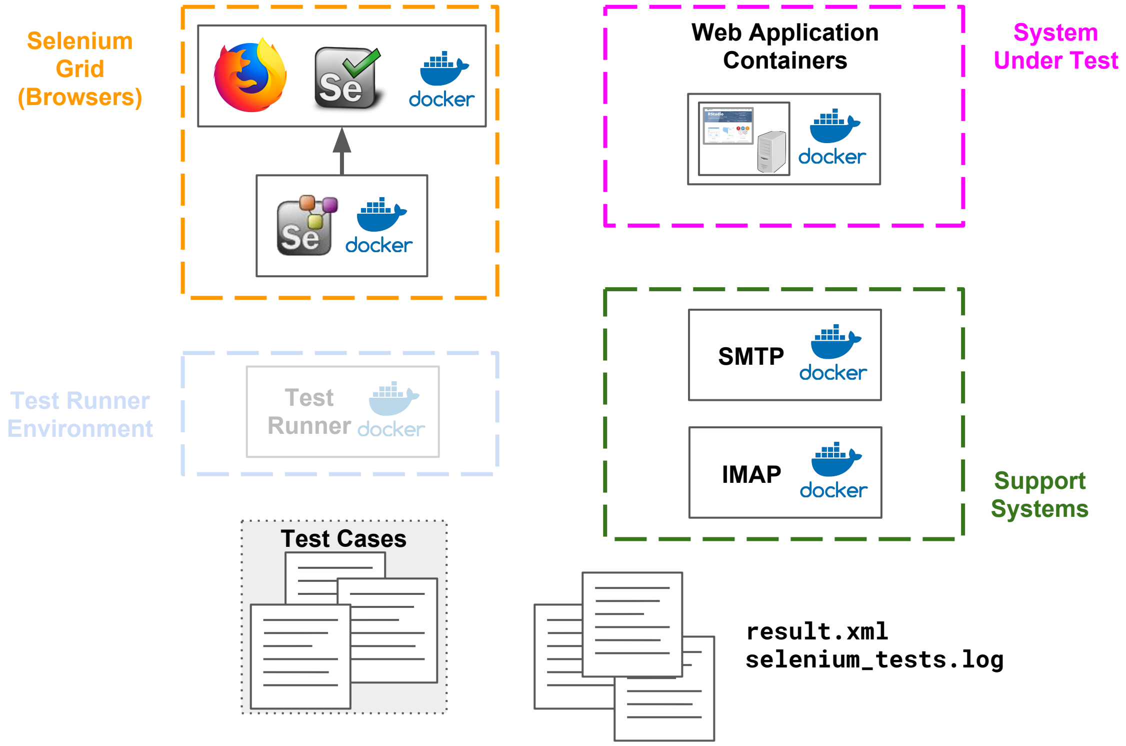
```
$ make test-env-up
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



Command Line

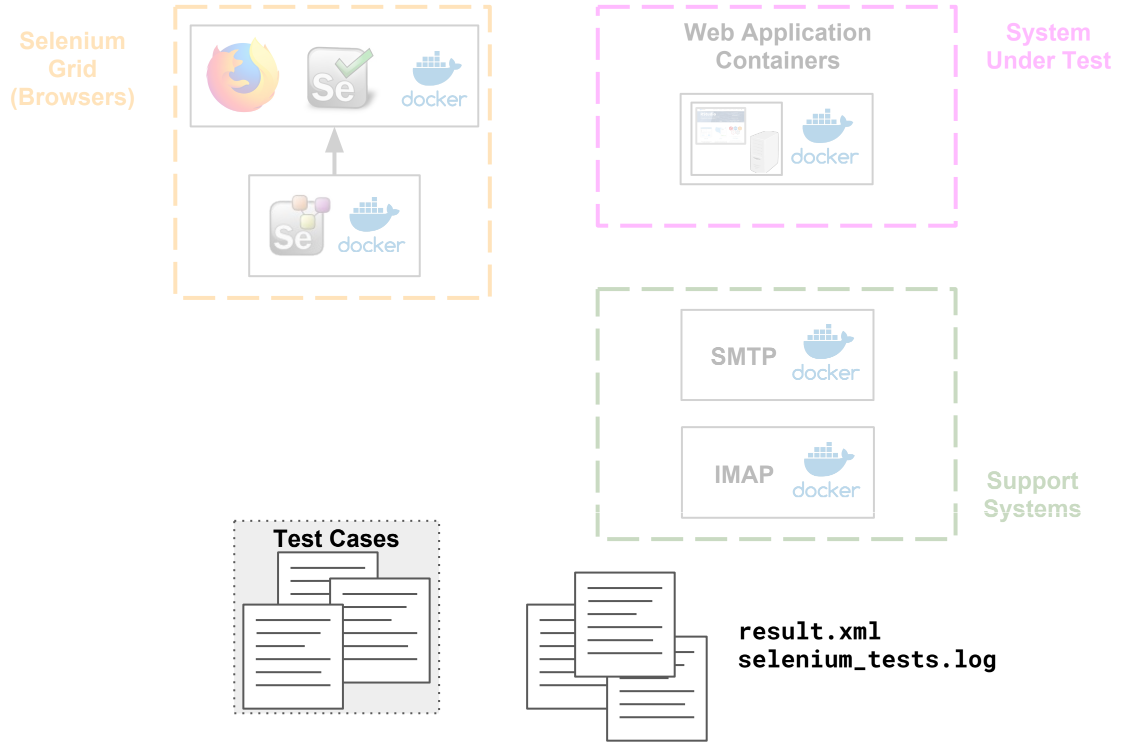
```
$ make test-env-up
$ make test
```



Command Line

```
$ make test-env-up
$ make test
$
```

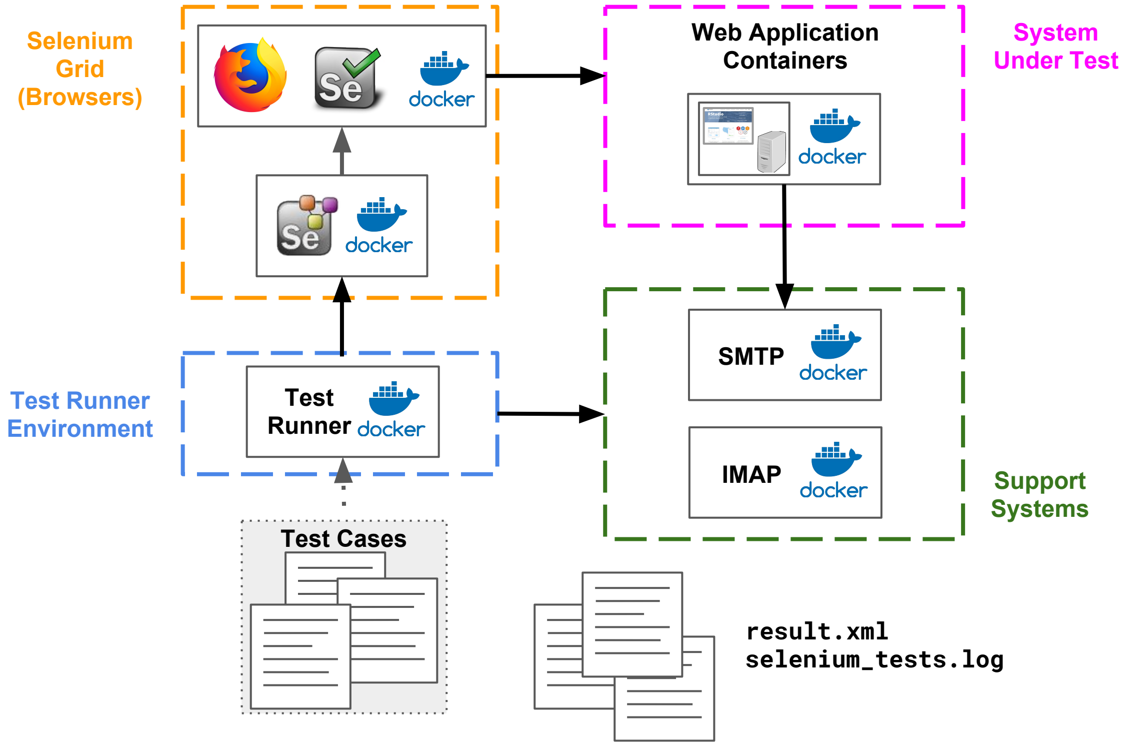
The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



Command Line

```
$ make test-env-up
$ make test
$ make test-env-down
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



Grid Docker images are maintained by SeleniumHQ project.

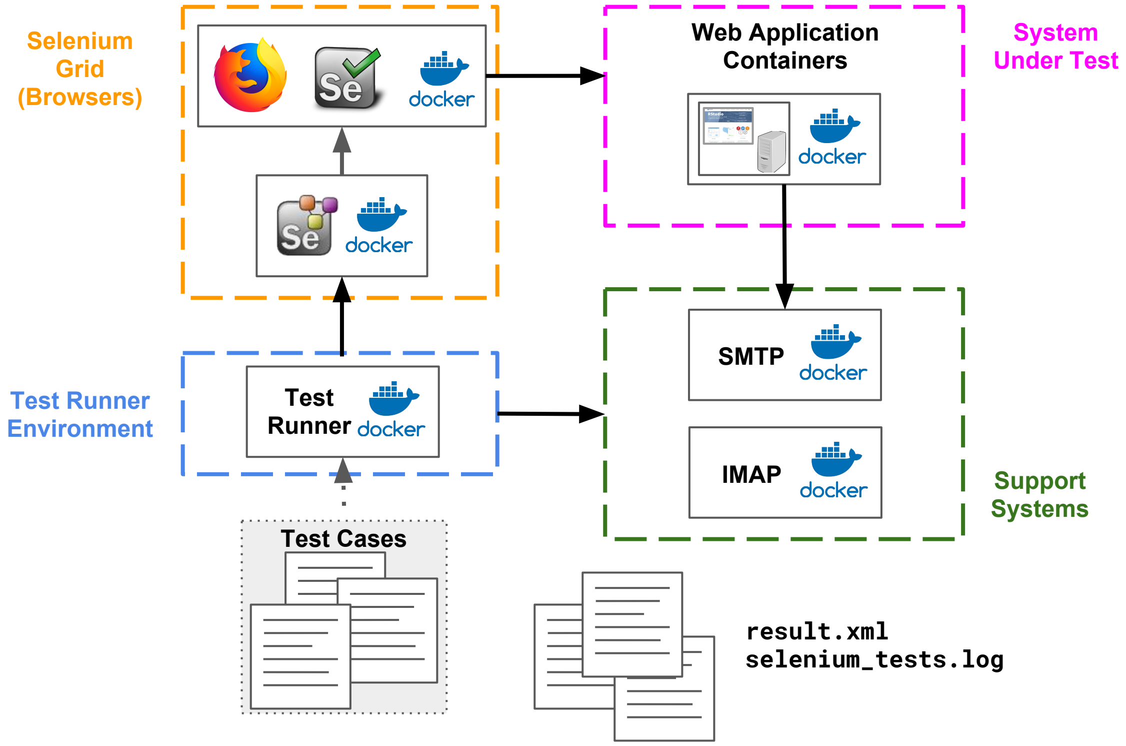
Tools used under the hood:

- bash, coreutils, make, …
- Docker, Docker Compose

Command Line

```
$ make test-env-up
$ make test
$ make test-env-down
```

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.



Grid Docker images are maintained by SeleniumHQ project.

Tools used under the hood:

- bash, coreutils, make, …
- Docker, Docker Compose

Command Line

```
$ make test-env-up
$ make test
$ make test-env-down
```

Makefile

```
COMMAND             = bash
SELENIUM_VERSION    = 3.8.1-dubnium
TRE_IMAGE           = dskard/tew:0.1.0
DOCKER_RUN_COMMAND  = docker run …
TEST_RUNNER_COMMAND = pytest …
...


test: wait-for-systems-up prepare-logging
    ${DOCKER_RUN_COMMAND} ${TEST_RUNNER_COMMAND} \
        > ${TMP_PIPE} || EXITCODE=$$?; \
    rm -f ${TMP_PIPE}; \
    exit $$EXITCODE

run:
    @${DOCKER_RUN_COMMAND} ${COMMAND}

test-env-up: grid-up

test-env-down: network-down

grid-up: network-up
    NETWORK=${NETWORK} \
    GRID_TIMEOUT=${GRID_TIMEOUT} \
    SELENIUM_VERSION=${SELENIUM_VERSION} \
    docker-compose -f ${DCYML_GRID} -p ${PROJECT} up \
        -d \
        --scale firefox=${SCALE_FIREFOX} \
        --scale chrome=${SCALE_CHROME}
...
```

A word about Makefiles…

- Variables declared a top
- Rules tell how to build targets
- *test-env-up*, *test*, and *test-env-down* are targets

Command Line

```
$ make test-env-up
$ make test
$ make test-env-down
```

Makefile

```
COMMAND                = bash
SELENIUM_VERSION     = 3.8.1-dubnium
TRE_IMAGE             = dskard/tew:0.1.0
DOCKER_RUN_COMMAND  = docker run …
TEST_RUNNER_COMMAND = pytest …
...


test: wait-for-systems-up prepare-logging
    ${DOCKER_RUN_COMMAND} ${TEST_RUNNER_COMMAND} \
        > ${TMP_PIPE} || EXITCODE=$$?; \
    rm -f ${TMP_PIPE}; \
    exit $$EXITCODE

run:
    @${DOCKER_RUN_COMMAND} ${COMMAND}

test-env-up: grid-up

test-env-down: network-down

grid-up: network-up
    NETWORK=${NETWORK} \
    GRID_TIMEOUT=${GRID_TIMEOUT} \
    SELENIUM_VERSION=${SELENIUM_VERSION} \
    docker-compose -f ${DCYML_GRID} -p ${PROJECT} up \
        -d \
        --scale firefox=${SCALE_FIREFOX} \
        --scale chrome=${SCALE_CHROME}
...
```

A word about Makefiles…

- Variables declared a top
- Rules tell how to build targets
- *test-env-up*, *test*, and *test-env-down* are targets

Command Line

```
$ make test-env-up
$ make test
$ make test-env-down
```

Makefile

```
COMMAND             = bash
SELENIUM_VERSION    = 3.8.1-dubnium
TRE_IMAGE           = dskard/tew:0.1.0
DOCKER_RUN_COMMAND  = docker run …
TEST_RUNNER_COMMAND = pytest …
...


test: wait-for-systems-up prepare-logging
    ${DOCKER_RUN_COMMAND} ${TEST_RUNNER_COMMAND} \
        > ${TMP_PIPE} || EXITCODE=$$?; \
    rm -f ${TMP_PIPE}; \
    exit $$EXITCODE

run:
    @${DOCKER_RUN_COMMAND} ${COMMAND}

test-env-up: grid-up

test-env-down: network-down

grid-up: network-up
    NETWORK=${NETWORK} \
    GRID_TIMEOUT=${GRID_TIMEOUT} \
    SELENIUM_VERSION=${SELENIUM_VERSION} \
    docker-compose -f ${DCYML_GRID} -p ${PROJECT} up \
        -d \
        --scale firefox=${SCALE_FIREFOX} \
        --scale chrome=${SCALE_CHROME}
...
```

A word about Makefiles…

- Variables declared a top
- Rules tell how to build targets
- *test-env-up*, *test*, and *test-env-down* are targets

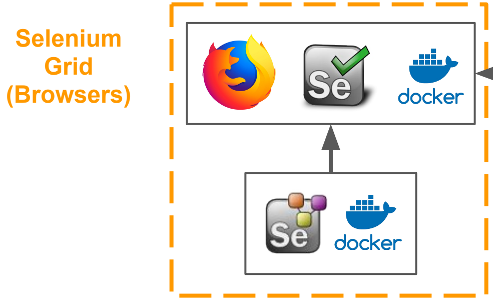
Command Line

```
$ make test-env-up
$ make test
$ make test-env-down
```

Q

How do you view the browsers as the tests run?

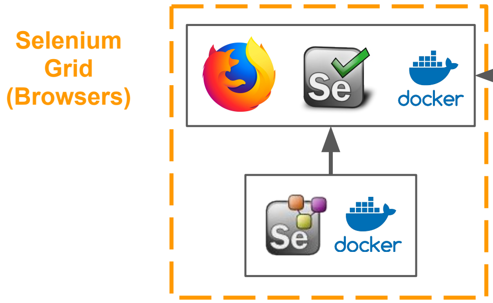
Q How do you view the browsers as the tests run?



selenium/node-chrome**-debug**
or
selenium/node-firefox**-debug**

VNC server on port 5900.

Q How do you view the browsers as the tests run?

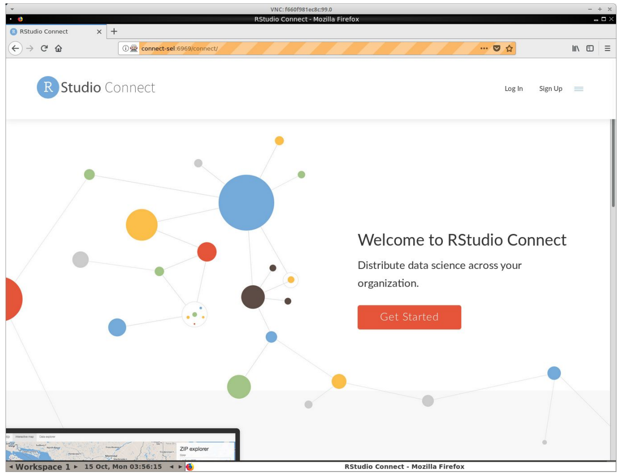


selenium/node-chrome-**debug**
or
selenium/node-firefox-**debug**

VNC server on port 5900.

Command Line (The Easy Way)

```
$ make test-env-up
$ ./shownode
```


VNC: f660f981ec8c:99.0
RStudio Connect - Mozilla Firefox
RStudio Connect
connect-sel:6969/connect/
RStudio Connect
Log In
Sign Up

Welcome to RStudio Connect

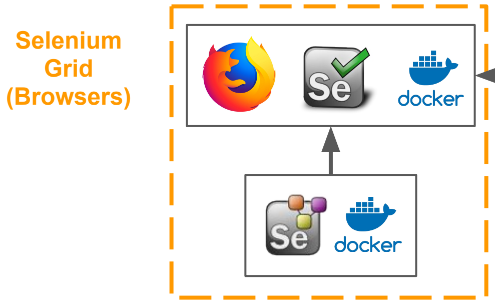
Distribute data science across your organization.

Get Started

Workspace 1 15 Oct, Mon 03:56:15 RStudio Connect - Mozilla Firefox

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.

Q How do you view the browsers as the tests run?

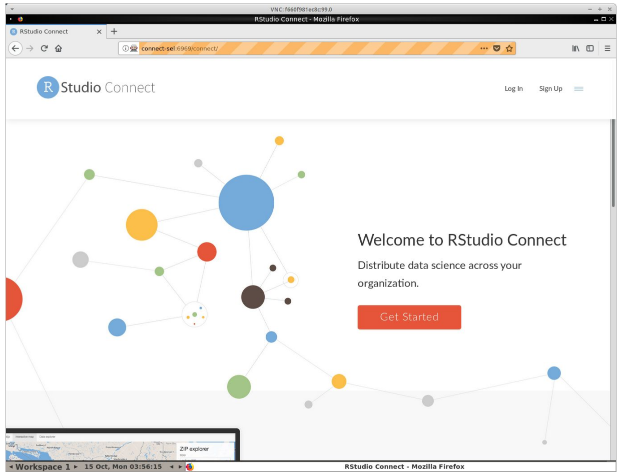


selenium/node-chrome**-debug**
or
selenium/node-firefox**-debug**

VNC server on port 5900.

Command Line (The Hard Way)

```
$ make test-env-up
$ docker ps
IMAGE                      PORTS
selenium/node-firefox...   0.0.0.0:33002->5900/tcp

$ vncviewer localhost:33002
$ # open vnc://:secret@localhost:33002
$ # password is “secret”
```


RStudio Connect - Mozilla Firefox
RStudio Connect
connect-sel:6969/connect/
R Studio Connect
Log In
Sign Up
Get Started
Workspace 1
RStudio Connect - Mozilla Firefox

Welcome to RStudio Connect

Distribute data science across your organization.

The Firefox logo is a trademark of the Mozilla Foundation in the U.S. and other countries.
Docker Marks are a trademark of Docker, Inc.

Q

How do you debug test cases?

Q

How do you debug test cases?

Test Case

```
def test_valid_login(self):
    menu = HeaderMenuFrontPage()


    menu.login.click()
    …
```

Q

How do you debug test cases?

Test Case

```
def test_valid_login(self):
    menu = HeaderMenuFrontPage()

    import pdb; pdb.set_trace()
    menu.login.click()
    …
```

Q

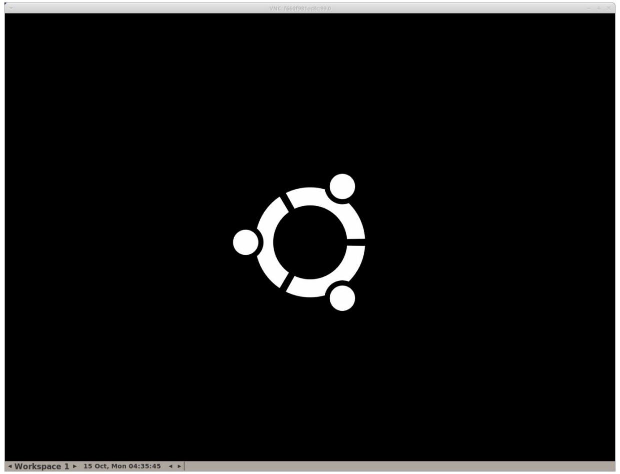
How do you debug test cases?

Test Case

```
def test_valid_login(self):
    menu = HeaderMenuFrontPage()

    import pdb; pdb.set_trace()
    menu.login.click()
    …
```

Command Line

```
$ ./shownode
$
```



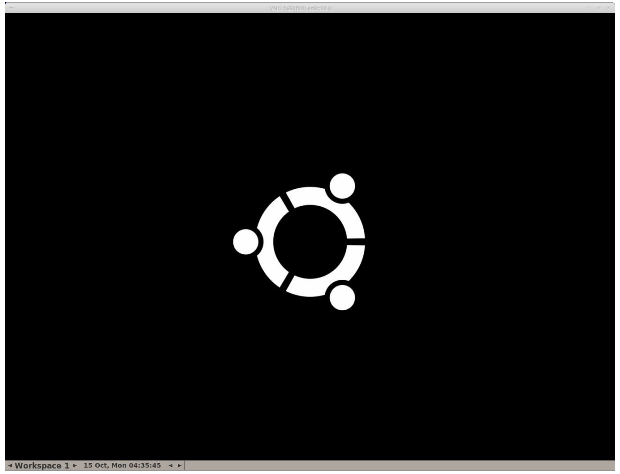
Q How do you debug test cases?

Test Case

```
def test_valid_login(self):
    menu = HeaderMenuFrontPage()

    import pdb; pdb.set_trace()
    menu.login.click()
    …
```

Command Line

```
$ ./shownode
$ make test PYTESTOPTS="-k test_valid_login"
```



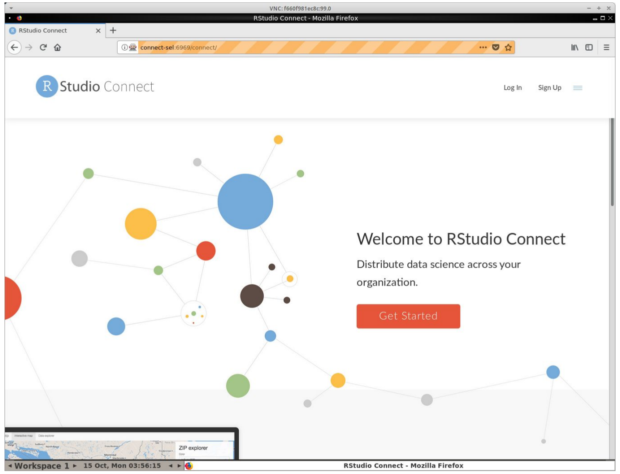
Q How do you debug test cases?

Test Case

```
def test_valid_login(self):
    menu = HeaderMenuFrontPage()

    import pdb; pdb.set_trace()
    menu.login.click()
    …
```

Command Line

```
$ ./shownode
$ make test PYTESTOPTS=“-k test_valid_login”
…
[30] > /opt/…/test_login.py(175)test_valid_login()
-> menu.login.click()
(Pdb++)
```


RStudio Connect - Mozilla Firefox
RStudio Connect
connect-sel:6969/connect/
RStudio Connect
Log In
Sign Up
Welcome to RStudio Connect
Distribute data science across your organization.
Get Started
Workspace 1
15 Oct, Mon 03:56:15
RStudio Connect - Mozilla Firefox

Q How do you write new test cases?

Q How do you write new test cases?

Command Line

```
$ make test-env-up DEBUG=1
```

- Set **DEBUG=1** so browser doesn’t timeout.

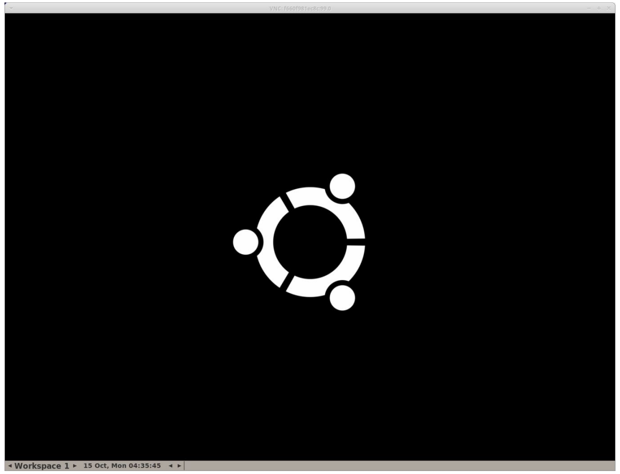
Q

How do you write new test cases?

- Set **DEBUG=1** so browser doesn’t timeout.

Command Line

```
$ make test-env-up DEBUG=1
$ ./shownode
```

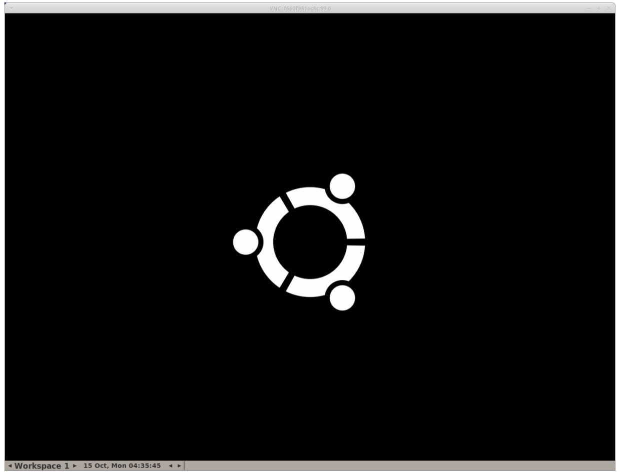


Q How do you write new test cases?

Command Line

```
$ make test-env-up DEBUG=1
$ ./shownode
$ make run COMMAND=ipython3
```

- Set **DEBUG=1** so browser doesn’t timeout.
- Set **COMMAND=ipython3** launches interpreter in container

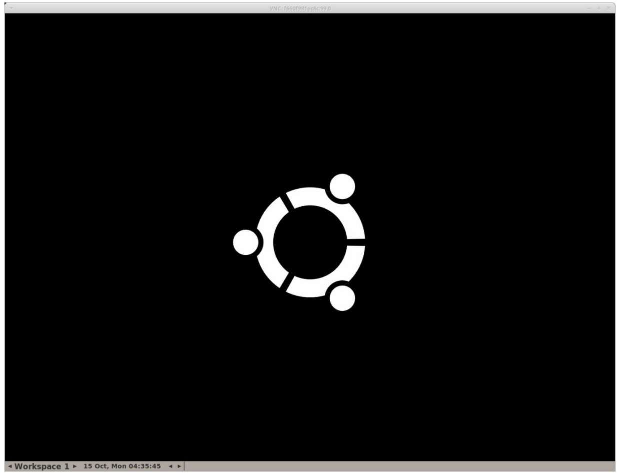


Q How do you write new test cases?

Command Line

```
$ make test-env-up DEBUG=1
$ ./shownode
$ make run COMMAND=ipython3
…
In [1]: from selenium import webdriver
```

- Set **DEBUG=1** so browser doesn't timeout.
- Set **COMMAND=ipython3** launches interpreter in container

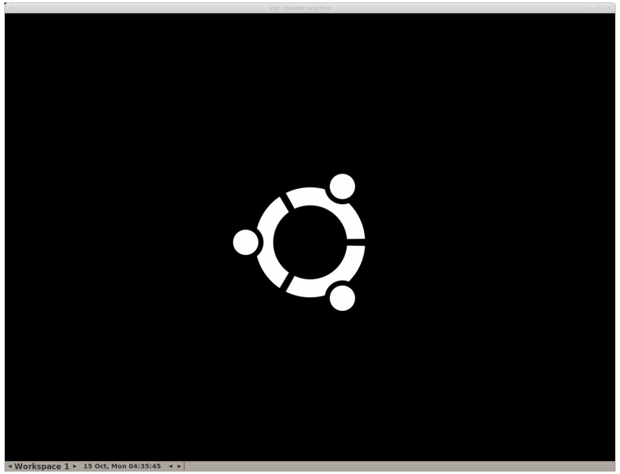


Q How do you write new test cases?

Command Line

```
$ make test-env-up DEBUG=1
$ ./shownode
$ make run COMMAND=ipython3
…
In [1]: from selenium import webdriver
In [2]: from selene.api import browser, s, be
```

- Set **DEBUG=1** so browser doesn’t timeout.
- Set **COMMAND=ipython3** launches interpreter in container
- Use Selene library to wrap Selenium commands.

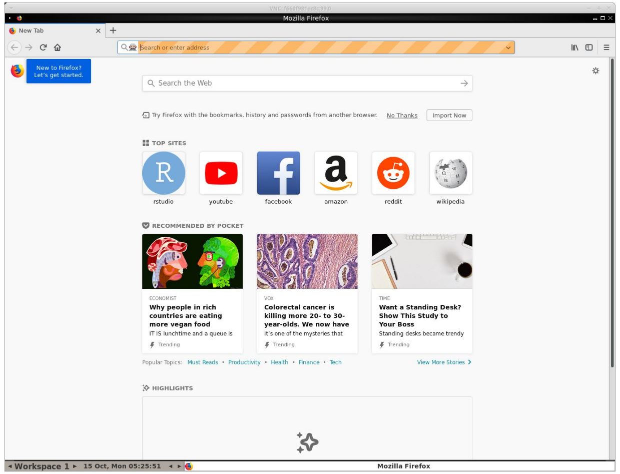


Q How do you write new test cases?

Command Line

```
$ make test-env-up DEBUG=1
$ ./shownode
$ make run COMMAND=ipython3
…
In [1]: from selenium import webdriver
In [2]: from selene.api import browser, s, be
In [3]: driver = webdriver.Remote(
            "http://selenium-hub:4444/wd/hub",
            webdriver.DesiredCapabilities\
                .FIREFOX.copy())
```

- Set **DEBUG=1** so browser doesn't timeout.
- Set **COMMAND=ipython3** launches interpreter in container
- Use Selene library to wrap Selenium commands.



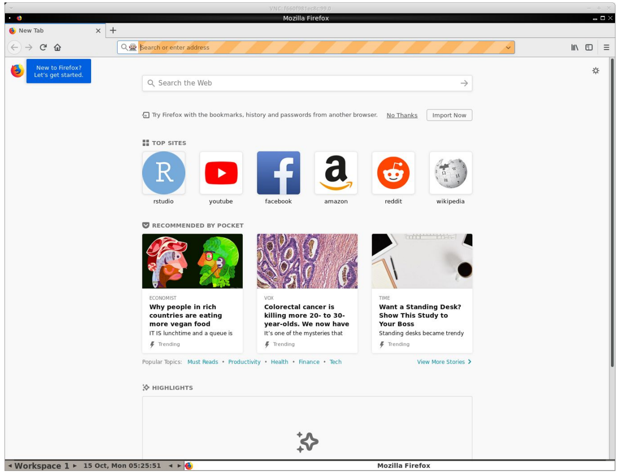
Mozilla Firefox
New Tab
Search or enter address
New to Firefox? Let's get started.
Search the Web
Try Firefox with the bookmarks, history and passwords from another browser.
No Thanks
Import Now
TOP SITES
rstudio
youtube
facebook
amazon
reddit
wikipedia
RECOMMENDED BY POCKET
Why people in rich countries are eating more vegan food
Colorectal cancer is killing more 20- to 30-year-olds. We now have
Want a Standing Desk? Show This Study to Your Boss
Popular Topics: Must Reads • Productivity • Health • Finance • Tech
View More Stories
HIGHLIGHTS
Workspace 1
15 Oct, Mon 05:25:51

Q How do you write new test cases?

Command Line

```
$ make test-env-up DEBUG=1
$ ./shownode
$ make run COMMAND=ipython3
…
In [1]: from selenium import webdriver
In [2]: from selene.api import browser, s, be
In [3]: driver = webdriver.Remote(
              "http://selenium-hub:4444/wd/hub",
              webdriver.DesiredCapabilities\
                  .FIREFOX.copy())
In [4]: browser.set_driver(driver)
```

- Set **DEBUG=1** so browser doesn't timeout.
- Set **COMMAND=ipython3** launches interpreter in container
- Use Selene library to wrap Selenium commands.

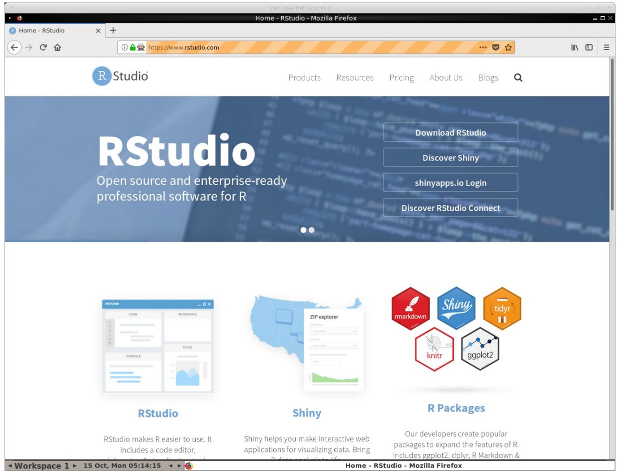

VNC: f660f981ec8c:99.0
Mozilla Firefox
New Tab
Search or enter address
New to Firefox? Let's get started.
Search the Web
Try Firefox with the bookmarks, history and passwords from another browser.
No Thanks
Import Now
TOP SITES
rstudio
youtube
facebook
amazon
reddit
wikipedia
RECOMMENDED BY POCKET
ECONOMIST
Why people in rich countries are eating more vegan food
IT IS lunchtime and a queue is
Trending
VOX
Colorectal cancer is killing more 20- to 30-year-olds. We now have
It's one of the mysteries that
Trending
TIME
Want a Standing Desk? Show This Study to Your Boss
Standing desks became trendy
Trending
Popular Topics: Must Reads • Productivity • Health • Finance • Tech
View More Stories
HIGHLIGHTS
Workspace 1
15 Oct, Mon 05:25:51
Mozilla Firefox

Q How do you write new test cases?

Command Line

```
$ make test-env-up DEBUG=1
$ ./shownode
$ make run COMMAND=ipython3
…
In [1]: from selenium import webdriver
In [2]: from selene.api import browser, s, be
In [3]: driver = webdriver.Remote(
              "http://selenium-hub:4444/wd/hub",
              webdriver.DesiredCapabilities\
                  .FIREFOX.copy())
In [4]: browser.set_driver(driver)
In [5]: browser.open_url('https://rstudio.com')
```

- Set **DEBUG=1** so browser doesn't timeout.
- Set **COMMAND=ipython3** launches interpreter in container
- Use Selene library to wrap Selenium commands.



Home - RStudio - Mozilla Firefox
Home - RStudio
https://www.rstudio.com
RStudio
Products
Resources
Pricing
About Us
Blogs
RStudio
Open source and enterprise-ready professional software for R
Download RStudio
Discover Shiny
shinyapps.io Login
Discover RStudio Connect
rmarkdown
Shiny
tidyr
knitr
ggplot2
RStudio
Shiny
R Packages
RStudio makes R easier to use. It includes a code editor,
Shiny helps you make interactive web applications for visualizing data. Bring
Our developers create popular packages to expand the features of R. Includes ggplot2, dplyr, R Markdown &
Workspace 1
15 Oct, Mon 05:14:15
Home - RStudio - Mozilla Firefox

Q How do you write new test cases?

Test Script

```
import pytest

from selene.api import browser, s, be

…
```

Load the Selene library modules.

Q How do you write new test cases?

Test Script

```
…
def test_valid_login(url, driver):
    """submit form with valid account info"""

    # navigate to the login page
    browser.open_url(url + ‘/login’)

    # login with local user credentials
    s(‘[data-auto=“username”]’).set(‘username’)
    s(‘[data-auto=“password”]’).set(‘password’)
    s(‘[data-auto=“submit”]’).click()

    # check that no error messages are shown
    s(‘[data-auto=“err”]’).should_not(be.visible)
```

Q

How do you write new test cases?

Test Script

```
…
def test_valid_login(url, driver):
    """submit form with valid account info"""

    # navigate to the login page
    browser.open_url(url + ‘/login’)

    # login with local user credentials
    s(‘[data-auto=“username”]’).set(‘username’)
    s(‘[data-auto=“password”]’).set(‘password’)
    s(‘[data-auto=“submit”]’).click()

    # check that no error messages are shown
    s(‘[data-auto=“err”]’).should_not(be.visible)
```

driver fixture comes from *pytest-selenium* plugin.

Takes care of starting web browser. No need to call ‘webdriver.Remote(…)’.

Q How do you write new test cases?

Test Script

```
…
def test_valid_login(url, driver):
    """submit form with valid account info"""

    # navigate to the login page
    browser.open_url(url + '/login')

    # login with local user credentials
    s('[data-auto="username"]').set('username')
    s('[data-auto="password"]').set('password')
    s('[data-auto="submit"]').click()

    # check that no error messages are shown
    s('[data-auto="err"]').should_not(be.visible)
```

Selene's **s()** method accepts a locator and returns an object that lazily represents an element.

Q How do you write new test cases?

Test Script

```
…
def test_valid_login(url, driver):
    """submit form with valid account info"""

    # navigate to the login page
    browser.open_url(url + ‘/login’)

    # login with local user credentials
    s(‘[data-auto=“username”]’).set(‘username’)
    s(‘[data-auto=“password”]’).set(‘password’)
    s(‘[data-auto=“submit”]’).click()

    # check that no error messages are shown
    s(‘[data-auto=“err”]’).should_not(be.visible)
```

Selene’s s() method accepts a locator and returns an object that lazily represents an element.

The search occurs when the **action** is performed on the element.

Q How do you write new test cases?

Test Script

```
…
def test_valid_login(url, driver):
    """submit form with valid account info"""

    # navigate to the login page
    browser.open_url(url + ‘/login’)

    # login with local user credentials
    s(‘[data-auto=“username”]’).set(‘username’)
    s(‘[data-auto=“password”]’).set(‘password’)
    s(‘[data-auto=“submit”]’).click()

    # check that no error messages are shown
    s(‘[data-auto=“err”]’).should_not(be.visible)
```

Selene’s s() method accepts a locator and returns an object that lazily represents an element.

The search occurs when the action is performed on the element.

should() and **should_not()** functions perform assertion of element’s condition and take screenshots on failure.

Q

How do I hook this up to Continuous Integration?

Q How do I hook this up to Continuous Integration?

Jenkinsfile

```
…
try {

    sh ‘make test-env-up’

    try {

        sh ‘make test’

    } finally {
        archiveArtifacts ‘*.png, *.xml, *.log’
        junit ‘*.xml’
    }

} finally {
    sh ‘make test-env-down’
}
```

Use standard commands to

- launch environment
- run tests
- clean environment

Q How do I hook this up to Continuous Integration?

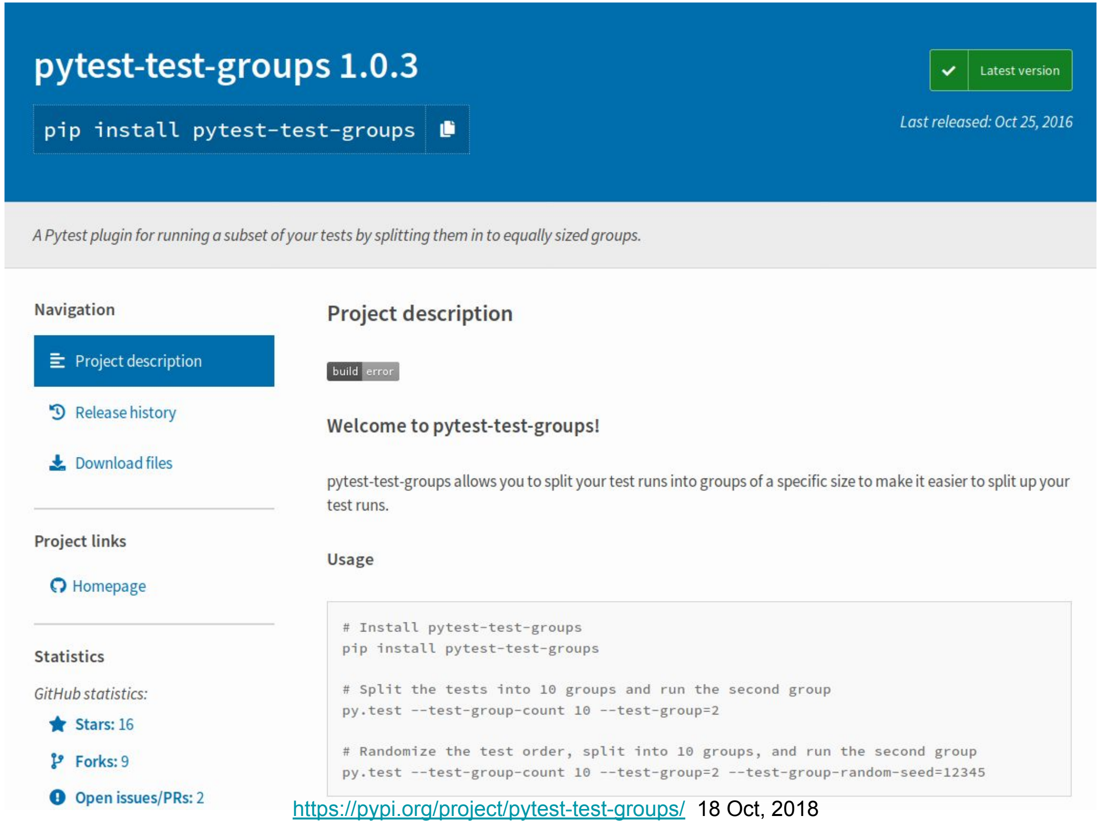
Jenkinsfile

```
…
try {

    sh ‘make test-env-up’

    try {

        sh ‘make test’

    } finally {
        archiveArtifacts ‘*.png, *.xml, *.log’
        junit ‘*.xml’
    }

} finally {
    sh ‘make test-env-down’
}
```

Use standard commands to
- launch environment
- run tests
- clean environment

Store:
- screenshots from failed tests (*.png)
- junit result (*.xml)
- saved stdout (*.log)

Q How do I hook this up to Continuous Integration?



pytest-test-groups 1.0.3

```
pip install pytest-test-groups
```

Last released: Oct 25, 2016

A Pytest plugin for running a subset of your tests by splitting them in to equally sized groups.

Project description

Welcome to pytest-test-groups!

pytest-test-groups allows you to split your test runs into groups of a specific size to make it easier to split up your test runs.

Usage

```
# Install pytest-test-groups
pip install pytest-test-groups

# Split the tests into 10 groups and run the second group
py.test --test-group-count 10 --test-group=2

# Randomize the test order, split into 10 groups, and run the second group
py.test --test-group-count 10 --test-group=2 --test-group-random-seed=12345
```

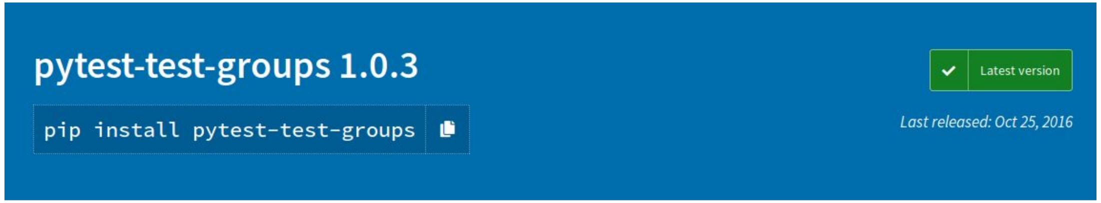
https://pypi.org/project/pytest-test-groups/ 18 Oct, 2018

Use standard commands to
- launch environment
- run tests
- clean environment

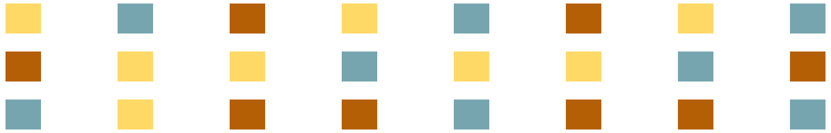
Store:
- screenshots from failed tests (*.png)
- junit result (*.xml)
- saved stdout (*.log)

Test Groups:
- pytest-test-groups plugin

Q How do I hook this up to Continuous Integration?


pytest-test-groups 1.0.3

```
pip install pytest-test-groups
```

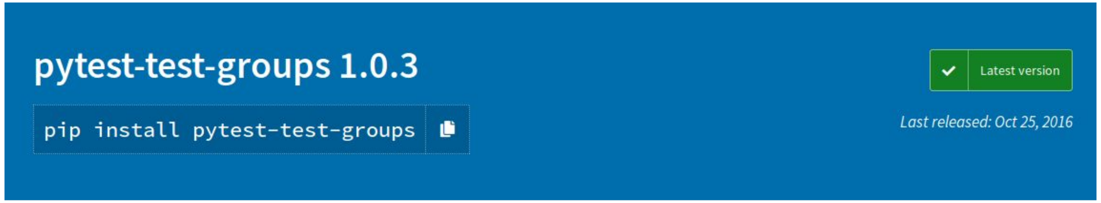


Use standard commands to
- launch environment
- run tests
- clean environment

Store:
- screenshots from failed tests (*.png)
- junit result (*.xml)
- saved stdout (*.log)

Test Groups:
- pytest-test-groups plugin

Q How do I hook this up to Continuous Integration?



pytest-test-groups 1.0.3

```
pip install pytest-test-groups
```

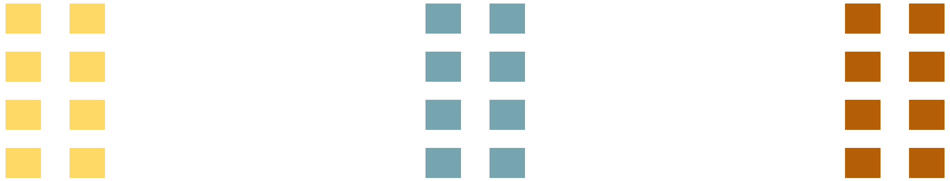
Last released: Oct 25, 2016

Use standard commands to

- launch environment
- run tests
- clean environment

Store:

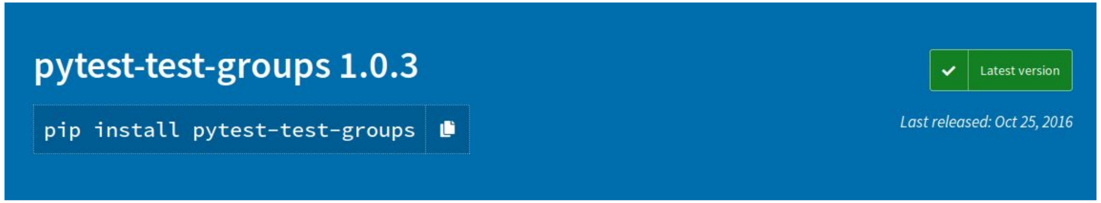
- screenshots from failed tests (*.png)
- junit result (*.xml)
- saved stdout (*.log)



Test Groups:

- pytest-test-groups plugin

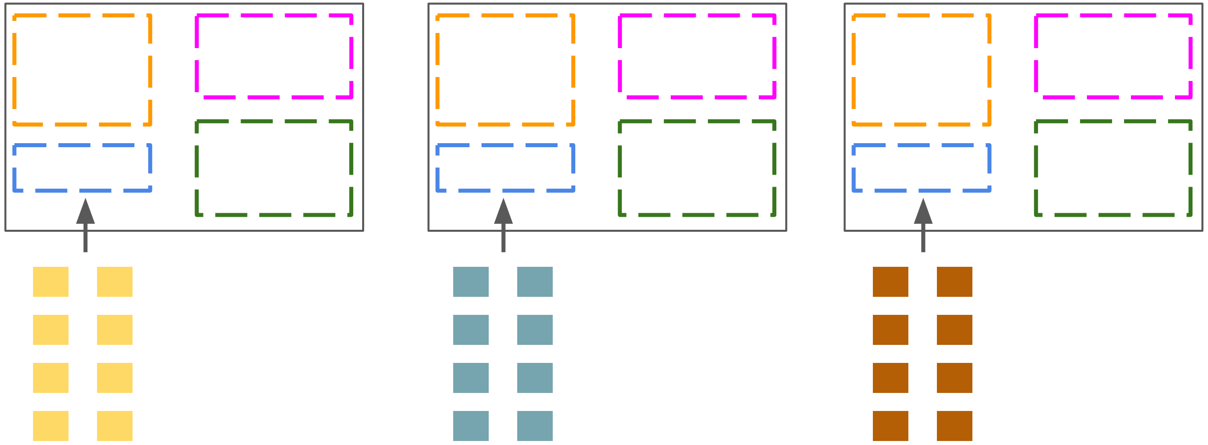
Q How do I hook this up to Continuous Integration?



pytest-test-groups 1.0.3

```
pip install pytest-test-groups
```

Last released: Oct 25, 2016



Use standard commands to
- launch environment
- run tests
- clean environment

Store:
- screenshots from failed tests (*.png)
- junit result (*.xml)
- saved stdout (*.log)

Test Groups:
- pytest-test-groups plugin

Q How do I hook this up to Continuous Integration?

Jenkinsfile

```
…
try {

    sh ‘make test-env-up’

    try {

        sh ‘make test PYTESTOPTS=“...”’

    } finally {
        archiveArtifacts ‘*.png, *.xml, *.log’
        junit ‘*.xml’
    }

} finally {
    sh ‘make test-env-down’
}
```

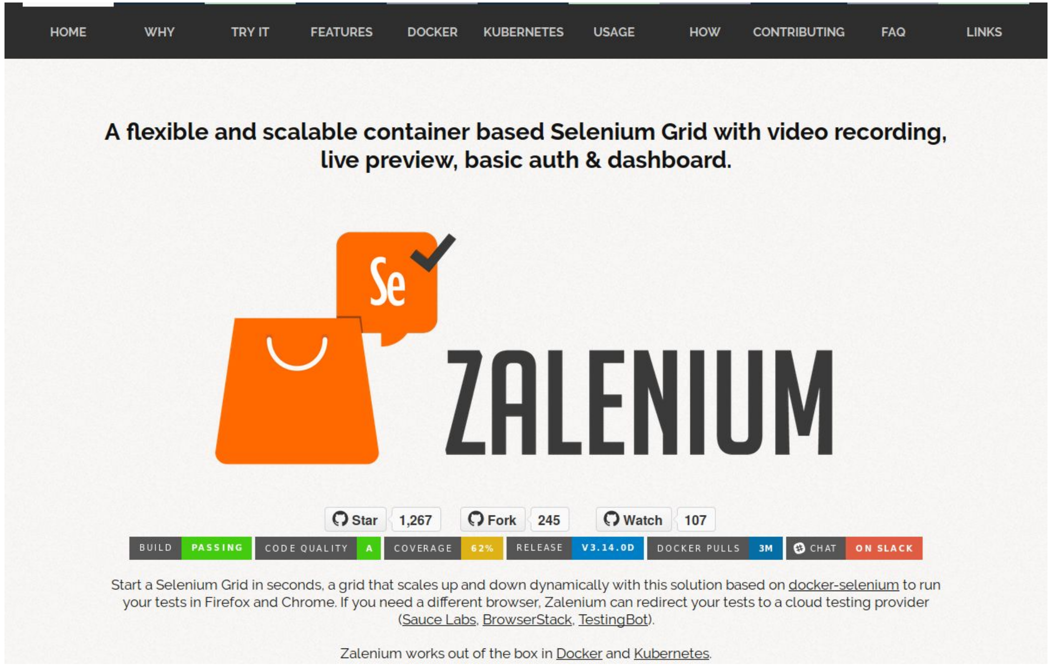
Use standard commands to
- launch environment
- run tests
- clean environment

Store:
- screenshots from failed tests (*.png)
- junit result (*.xml)
- saved stdout (*.log)

Test Groups:
- pytest-test-groups plugin

Q How do I send my tests to other Selenium Grids?

Q How do I send my tests to other Selenium Grids?



A flexible and scalable container based Selenium Grid with video recording, live preview, basic auth & dashboard.

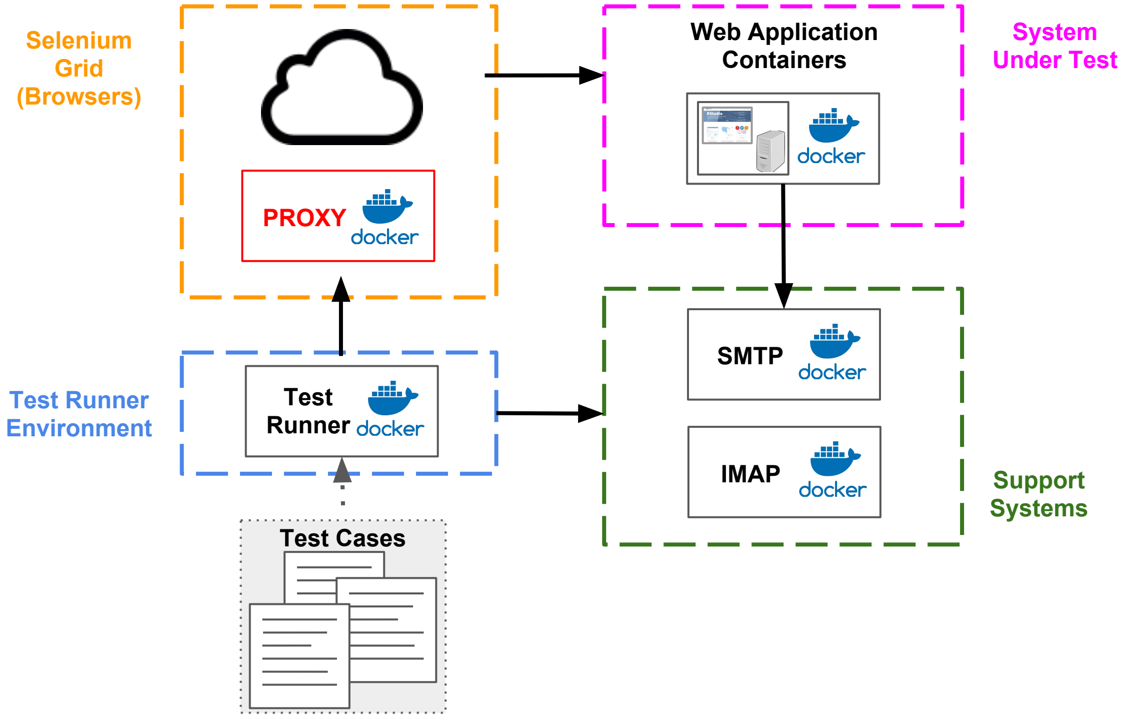
Start a Selenium Grid in seconds, a grid that scales up and down dynamically with this solution based on docker-selenium to run your tests in Firefox and Chrome. If you need a different browser, Zalenium can redirect your tests to a cloud testing provider (Sauce Labs, BrowserStack, TestingBot).

Zalenium works out of the box in Docker and Kubernetes.

https://opensource.zalando.com/zalenium/ 18 Oct, 2018

Zalenium:
- Dynamic Selenium Grid system from Zalando
- Selenium Conference Austin 2017 https://youtu.be/W5qMsVrob6I

Q How do I send my tests to other Selenium Grids?



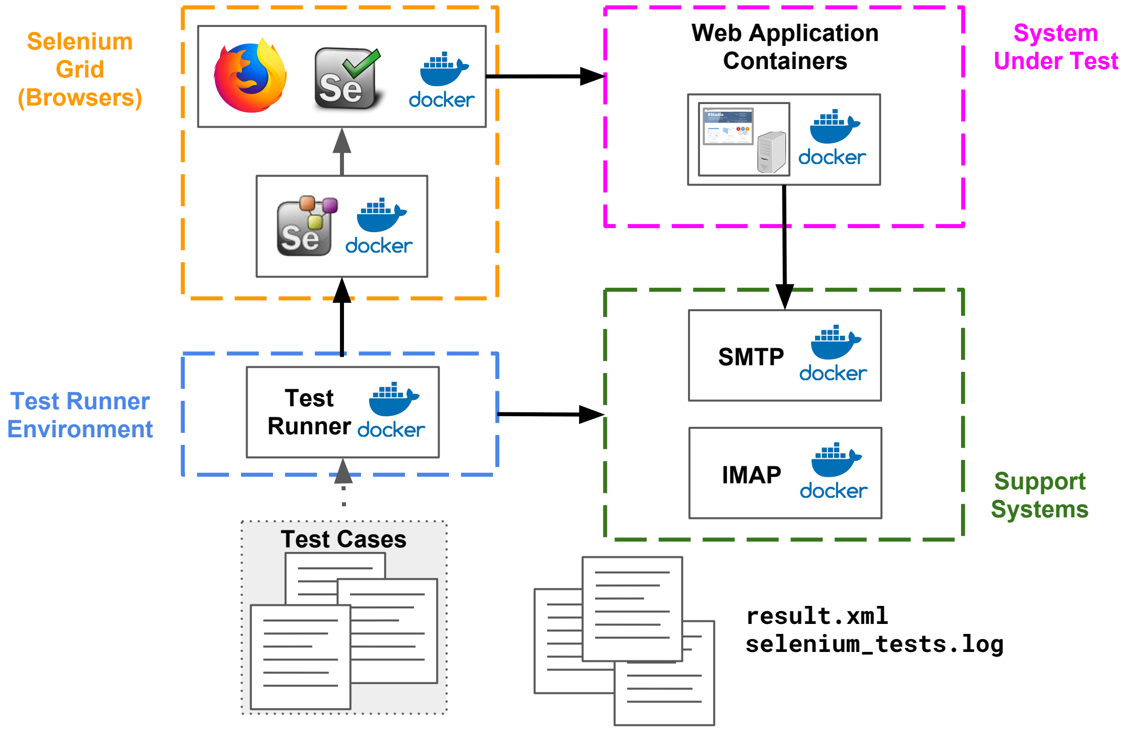
Zalenium:

- Dynamic Selenium Grid system from Zalando
- Selenium Conference Austin 2017 https://youtu.be/W5qMsVrob6I

Point your tests at an external grids through a proxy:

- Sauce Labs
- BrowserStack
- TestingBot

Docker Marks are a trademark of Docker, Inc.



Command Line

```
$ make test-env-up
$ make test
$ make test-env-down
```

Let's Talk!

office hours: 1pm

Slides and examples:

https://github.com/dskard/seleniumconf2018