



SeConf 13 Workshop

<http://appium.io/seconf.pdf>

Jonathan Lipps | @jlipps | Sauce Labs

@TheDanCuellar | @maudineormsby

appium is the cross-platform
solution for native and hybrid
mobile automation



appium introduction



iOS

calabash-ios
Frank
UIAutomation
ios-driver
KeepItFunctional

Android

calabash-android
MonkeyTalk
Robotium
UiAutomator
selendroid



Philosophy

R1. Test the same app you submit to the marketplace

R2. Write your tests in any language, using any framework

R3. Use a standard automation specification and API

R4. Build a large and thriving open-source community effort



Framework	R1	R2	R3	R4
calabash-ios				
Frank				
UIAutomation				
ios-driver	●	●	●	
KeepItFunctional	●			
calabash-android				
MonkeyTalk				
Robotium			●	
UiAutomator				
selendroid		●	●	
appium	●	●	●	●



Platforms

- Real devices (iOS, Android)
- Simulators (iOS, Android, FirefoxOS)
- Hybrid apps (iOS, Android, FirefoxOS)
- Safari on iOS
- Chrome on Android
- Robot-controlled devices



Architecture

- Apple Instruments & UIAutomation for iOS
- Google UiAutomator for Android (4.2.1 up)
- Selendroid for older Android & hybrid
- Selenium WebDriver interface



Selenium WebDriver?

- this is SeConf, isn't it?



appium.app



Appium.app

- GUI for launching Appium server
 - Monitor status
 - Set preferences



Appium.app

- Inspector for probing your app
 - Create hooks for UI elements in app
 - Try out actions
 - Record / playback actions
 - Convert UIAutomation JS to Appium code

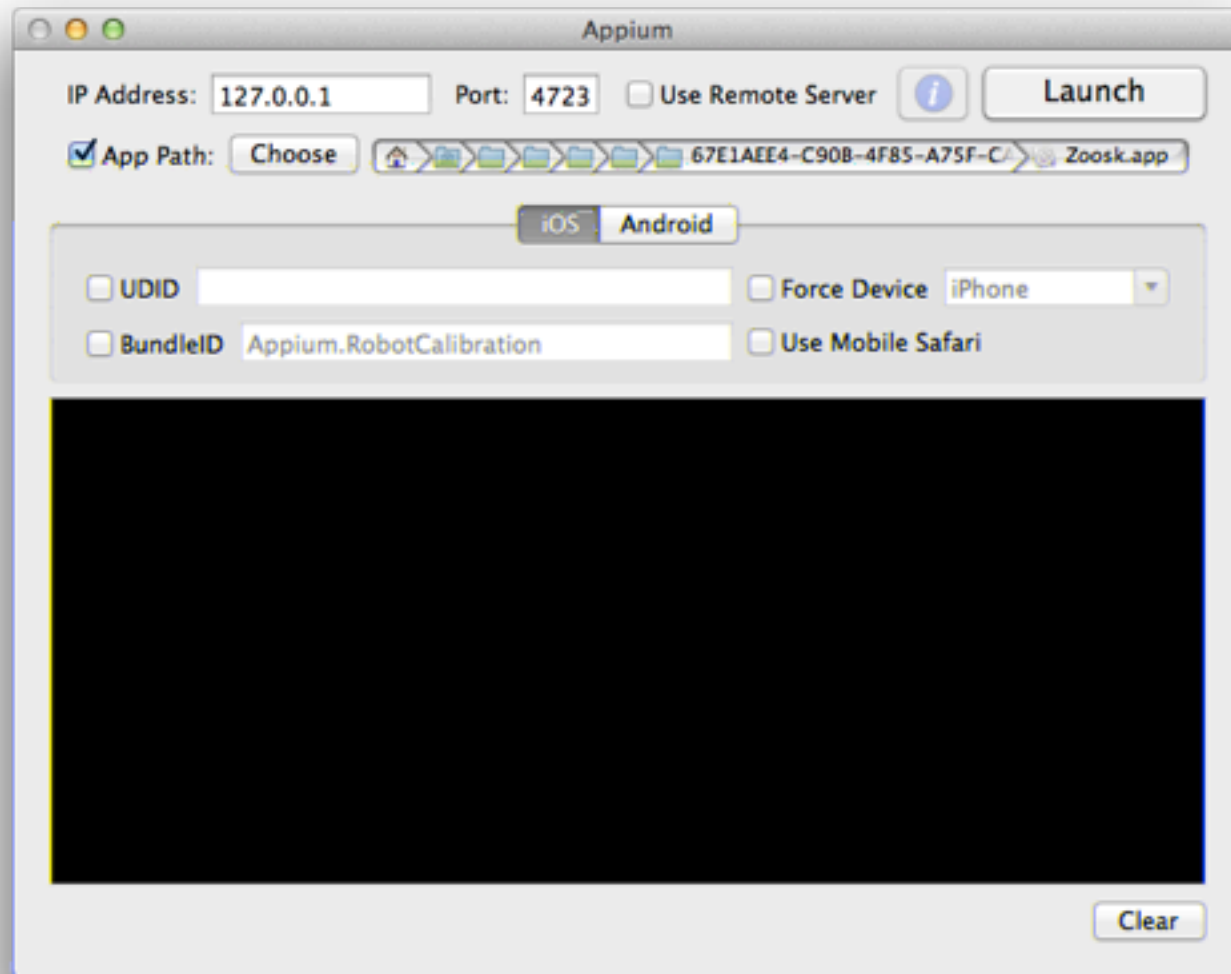


Appium.app

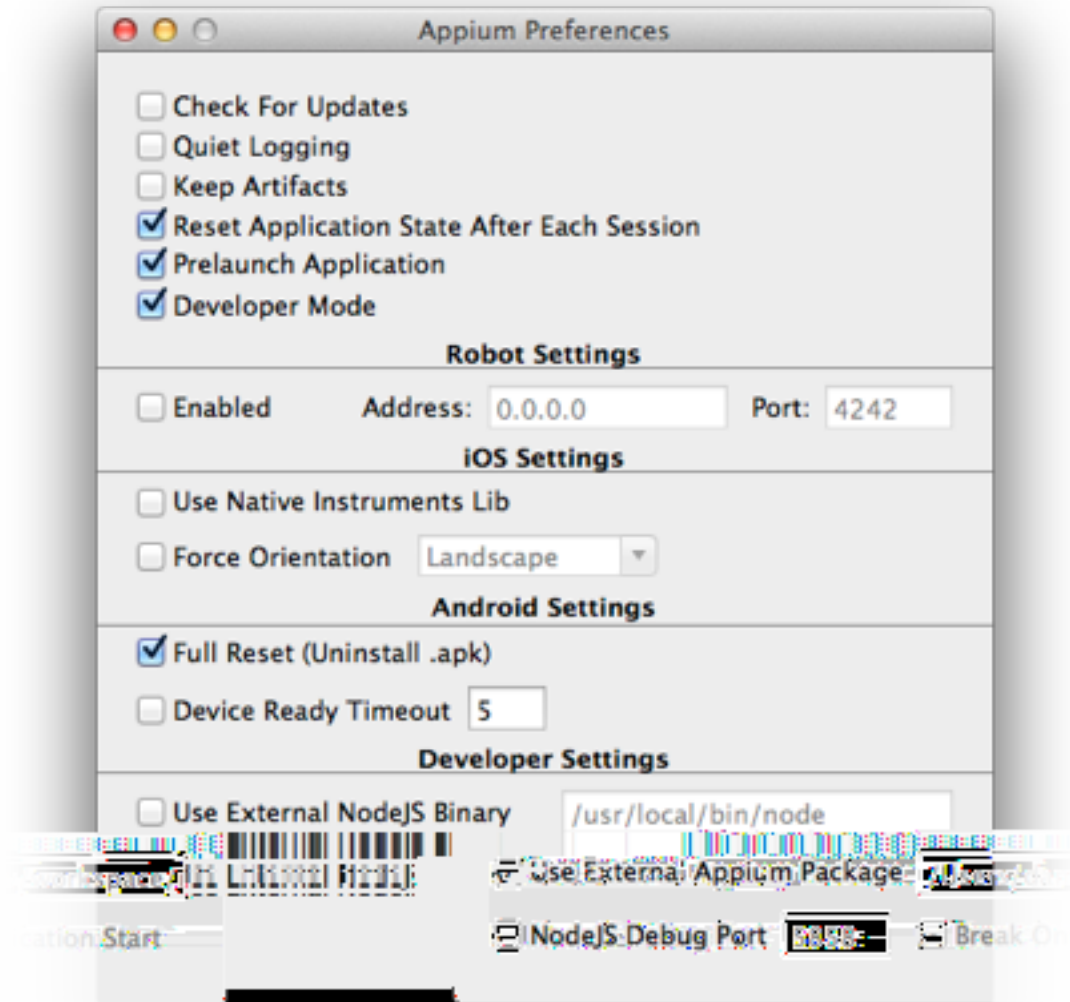
- Mac: stable
- Windows: under development



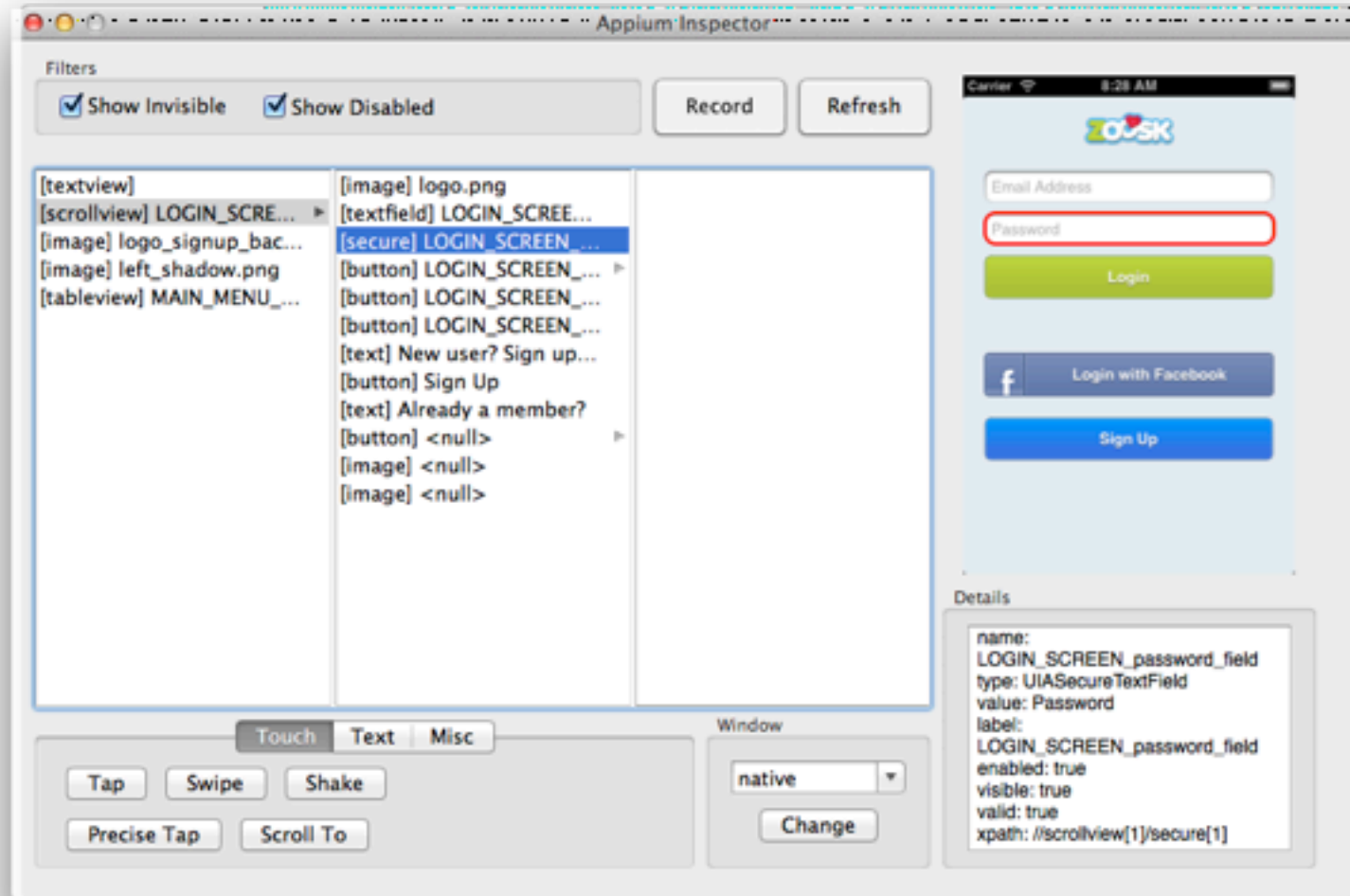
Monitor



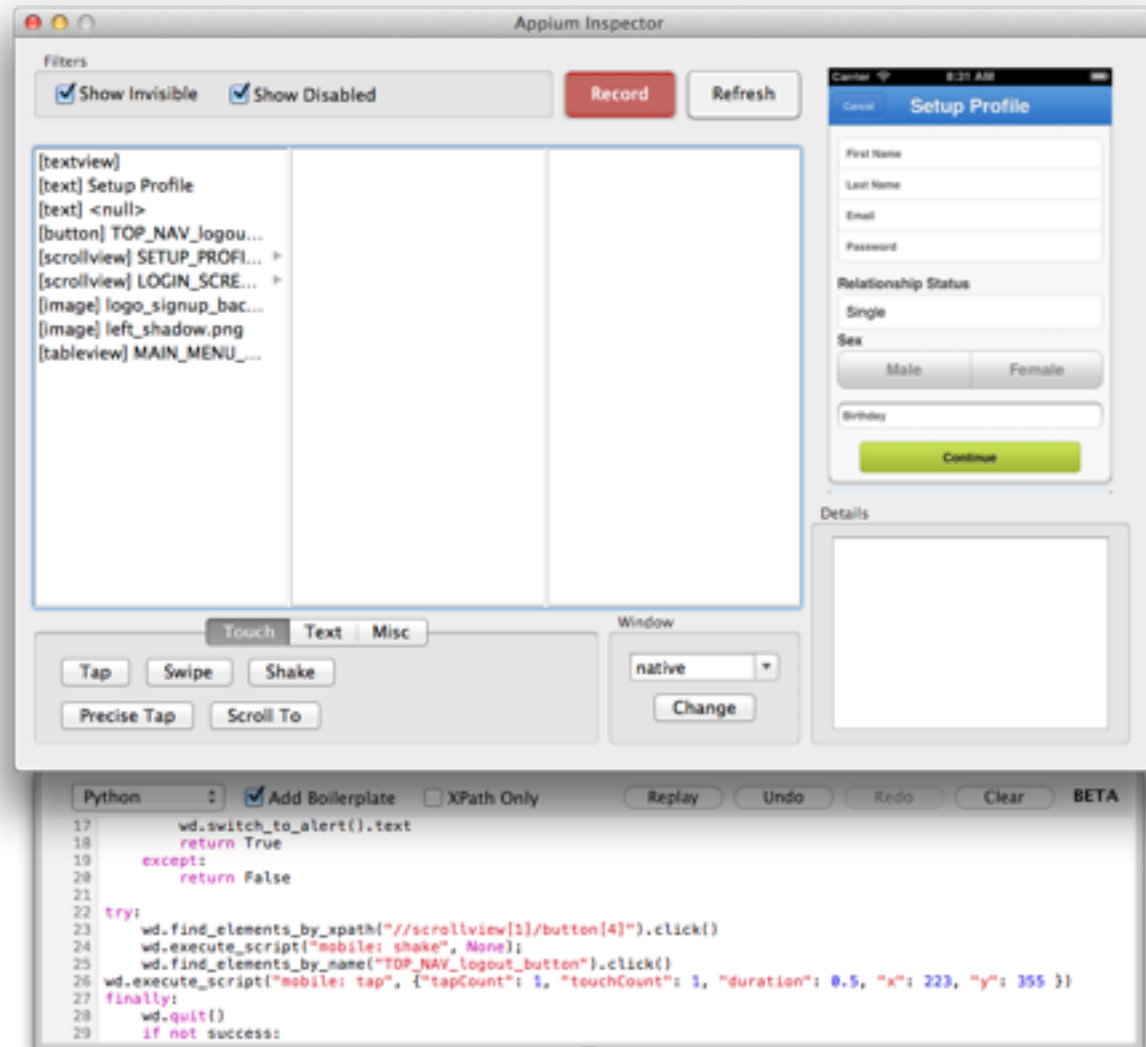
Preferences



Inspector



Recorder



Robot support

- Bitbeambot Delta-2
 - <http://www.bitbeam.org>
- Tapster
 - <https://www.tindie.com/products/hugs/robot-that-plays-angry-birds/>



Robot support

- Redirects touch actions to robots
 - tap
 - swipe
 - etc



Robot support

- Moving beyond UIAutomation for iOS
 - home button
 - task switching
 - etc



Robot support

- Check out the showroom in the back!



appium setup



Requirements (1/3)

- Mac (10.8 preferred)
 - If you're not on Mac, talk to us, we can still get you going for Android
- Python ≥ 2.7
- Xcode 4.6 with CLI tools and iOS 6.1
- Download Appium.app
 - <http://appium.io>



Requirements (2/3)

- Android Developer Tools ≥ 21
 - <http://developer.android.com/sdk/index.html>
 - mv to /usr/local/adt
 - export ANDROID_HOME=/usr/local/adt/sdk
 - add (.bashrc, .zshrc, etc):
\$ANDROID_HOME/tools
\$ANDROID_HOME/platform-tools
to \$PATH



Requirements (3/3)

- Set up Sauce Labs demo account credentials
- ```
export SAUCE_USERNAME="AppiumUser"
export SAUCE_PASSWORD="appiumrocks"
export SAUCE_ACCESS_KEY="e4c82f68-cf00-40af-a8bc-5c78df0511a9"
```
- This is for automating the test app; not an Appium requirement



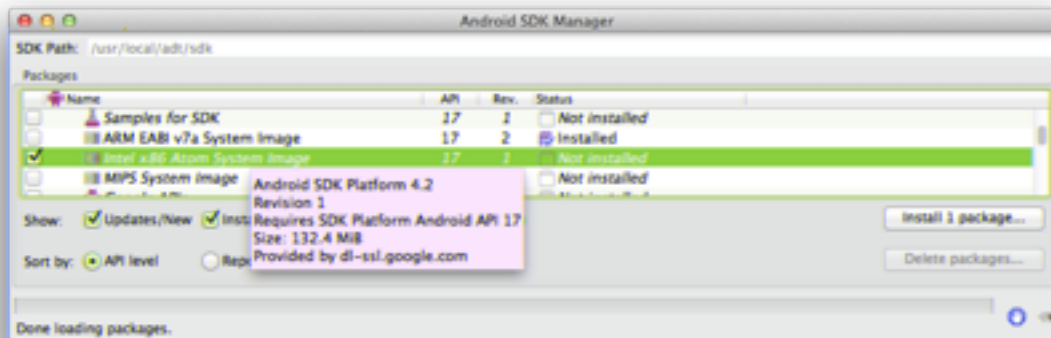
# Install HAXM for Android Speed!

- open `/usr/local/adt/sdk/extras/intel/  
Hardware_Accelerated_Execution_Manager/  
IntelHAXM.dmg`



# Make an Android Device

- android
- Check 'Intel x86 Atom System Image' - Android (4.2.2)
- Click 'Install 1 package...'



- Tools > Manage AVDs
- New...



# Create the Image

- AVD Name: workshop
- Device: Nexus S
- Target: Android 4.2.2
- CPU: Intel/Atom
- Host GPU



# Launch AVD

- In a new terminal window:
- `emulator @workshop -netfast`
- Go through the new device tour

```
emulator @workshop -netfast
HAX is working and emulator runs in fast virt mode
emulator: emulator window was out of view and was recentered
```

- `$ANDROID_HOME/sdk/tools/emulator @workshop -netfast (without env)`



# Get the workshop code

- `git clone https://github.com/appium/workshop.git appium-workshop`
- `cd appium-workshop`

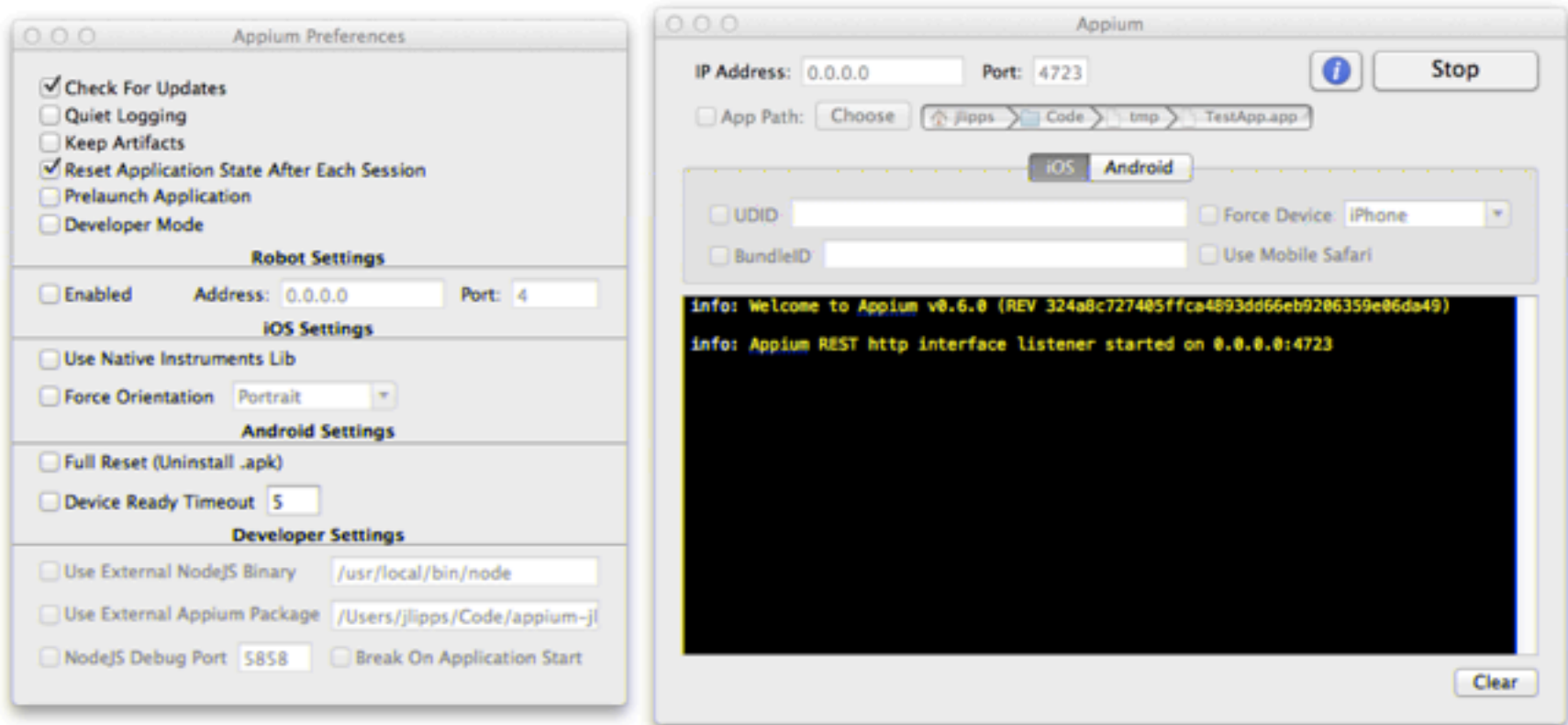


# Install dependencies

- `mkvirtualenv workshop # optional`
  - `deactivate && workon workshop`
- `pip install -r requirements.txt`
- `apps/download.sh`



# Launch Appium



- open /Applications/Appium.app





# Moment of truth...

- `cd python`
- `nosetests ios/step1.py`
- `nosetests android/step1.py`



# appium test model



# Start/stop a session

```
class AppiumTestCase(unittest.TestCase):

 def setUp(self):
 appium = "http://localhost:%s/wd/hub" % 4723
 self.desired_caps = {
 device: 'iPhone Simulator',
 app: '/abs/path/to/my.app',
 }
 self.driver = webdriver.Remote(appium, self.desired_caps)

 def tearDown(self):
 self.driver.quit()
```



# Find elements

```
by accessibility label
```

```
el = self.driver.find_element_by_name("Sign In")
```

```
by element type
```

```
el = self.driver.find_element_by_tag_name("button")
```

```
by hierarchy
```

```
el = self.driver.find_element_by_xpath("//button[@text='Hi ']")
```

```
by Android ID
```

```
el = self.driver.find_element_by_id("myId")
```



# Interact with elements

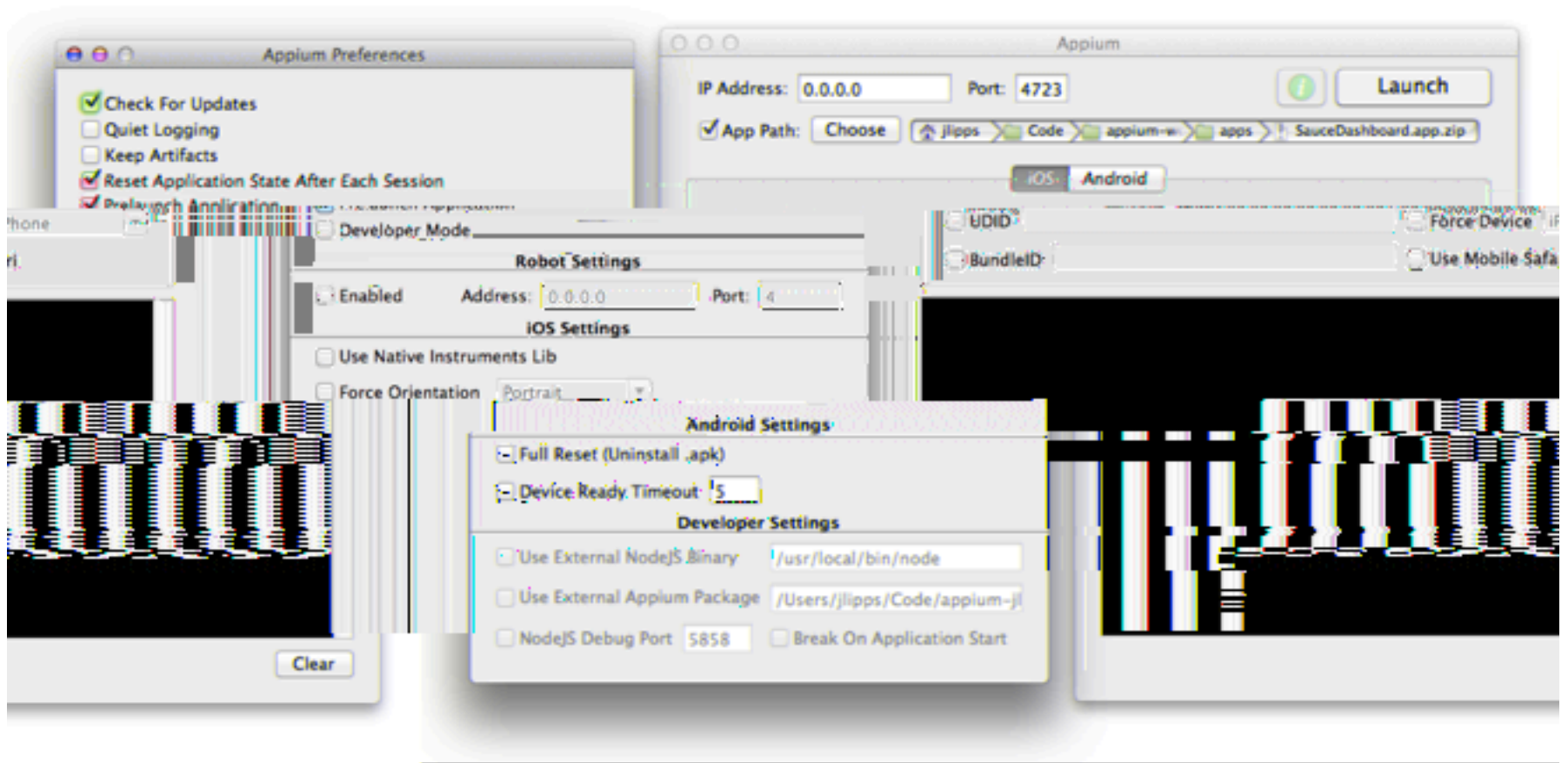
```
el = self.driver.find_element_by_tag_name("button")
print el.text # text of button
el.click() # click button
```



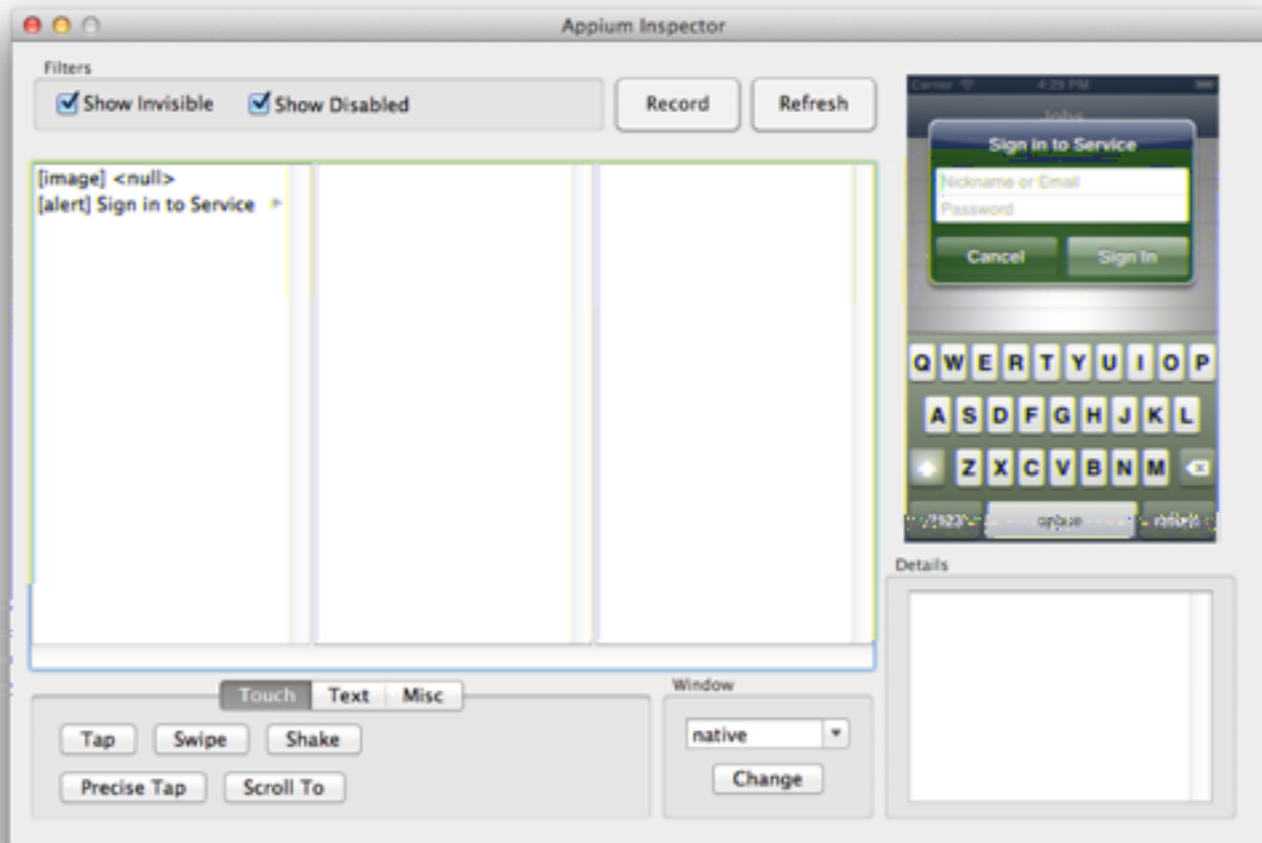
# appium test building



# Pre-launch app to inspect



# Inspector helps!





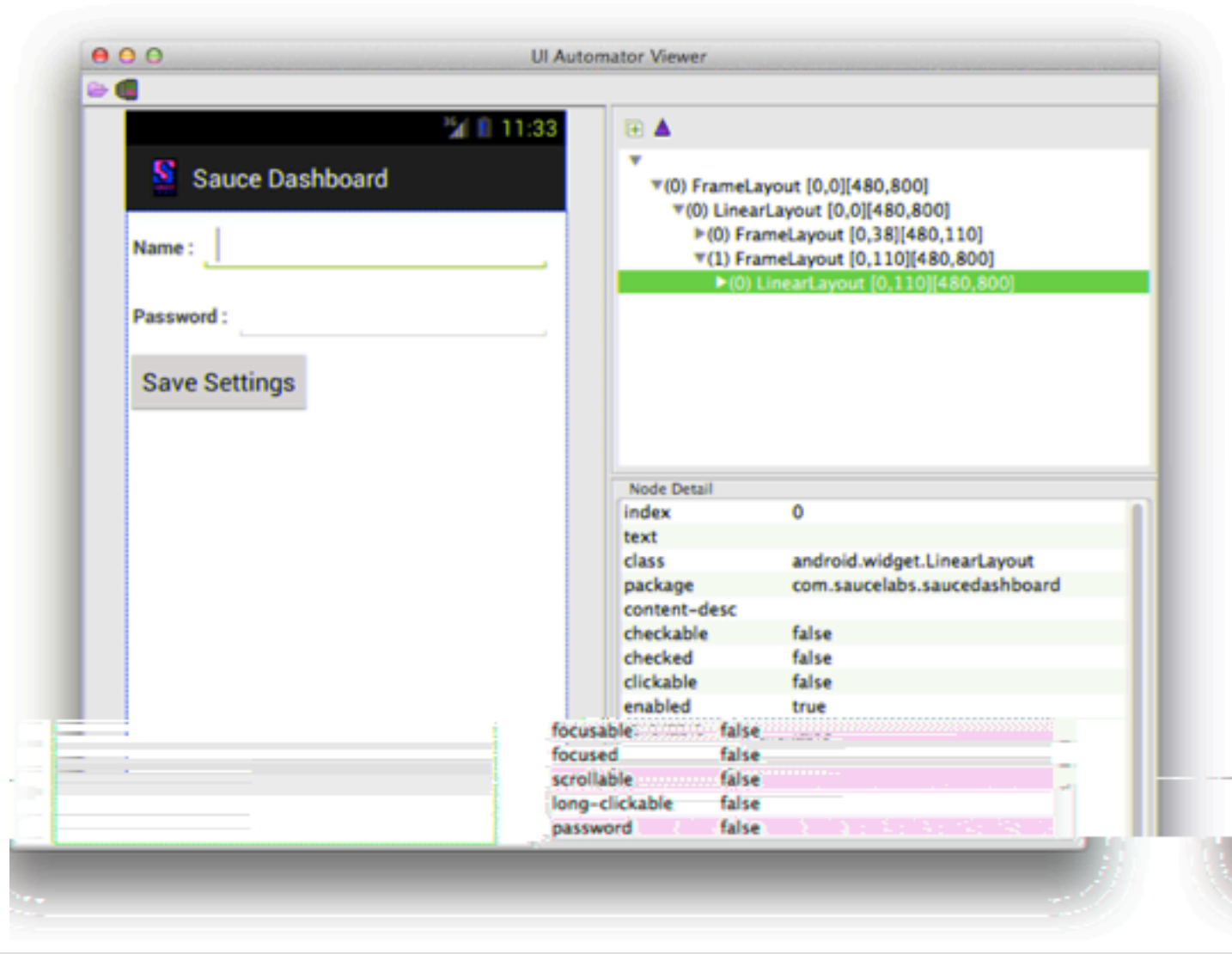
**edit** python/common.py



**edit** python/ios/step1.py



# uiautomatorviewer for Android



**edit** python/android/step1.py



# appium cross-platform



```
vim python/both.py
nosetests python/both.py
```



# appium real devices



# Android

- Device must be in developer mode
- USB debugging must be enabled





# IOS

- Choose the UDID option in Appium.app
- Can also bundle ID of app installed on device



appium scale



**appium is great for local** test development, but has limitations when scaling up for use in CI



**Sauce Labs is great for scale**  
when you need to run a lot of  
**appium** tests in your build



 [saucelabs.com/appium](https://saucelabs.com/appium)



# Run tests on Sauce

```
LOCAL
def setUp(self):
 appium = "http://localhost:%s/wd/hub" % 4723

 self.desired_caps = {
 device: 'iPhone Simulator',
 app: '/abs/path/to/my.app',
 }

red_caps) self.driver = webdriver.Remote(appium, self.desi

#SAUCE
def setUp(self):
 appium = "http://%s:%s@ondemand.saucelabs.com"
) = appium = appium % (SAUCE_USERNAME, SAUCE_PASSWORD)
 self.desired_caps = {
 device: 'iPhone Simulator',
 app: '/abs/path/to/my.app',
 }

red_caps) self.driver = webdriver.Remote(appium, self.desi
```



# appium mobile web



# appium committers





# We need you...

- Node.js devs (for Appium server)
- Obj-c devs (for Appium.app)
- C#.Net devs (for Appium.exe)
- Java devs (for Appium's Android bootstrap)



# Thanks!



appium.io

[github.com/appium/appium](https://github.com/appium/appium)

@AppiumDevs | @jlipps | @saucelabs

with @TheDanCuellar and @maudineormsby