-: Workshop :-

Accelerate Mobile QA & Deployment with Automated Testing & Continuous Integration

About Me: Akhilkumar Patel

- Architect & Strategist @ InfoStretch
  (akhilkumar.patel@infostretch.com)
  - Carrying over 13 years of experience at different level as developer, application architect, test automation architect, and Director of Quality
  - Demonstrated in-depth knowledge in designing & implementing solutions, quality & development processes and test automation frameworks using cutting edge technologies
  - Always keep evaluating and exploring latest and future technologies like NextGen Automation, Node.JS, DevOPs, BigData and NOSQL etc.

Agenda

Time always flies.. Let’s bind it.

Welcome

First Half

- Reminder
  - Pre-requisites
- Mobile Automation Solutions Overview
- Appium Overview
  - By Isaac Murchie
- Short Break
- Appium Walkthrough

Second Half

- Hands on
  - KAYAK Use Case
- How to run in Cloud
  - Saucelabs
- Continuous Integration
  - Jenkins
- Quick Recap
- Q & A

Workshop Pre-Requisites

Get Prepared for amazing ride of Mobile Automation

STPCON2015 Github REPO URL: https://github.com/patelakhil/STPCon2015.git

Pre-Requisites – Appium, Maven,JAVA Editor

REFER "Workshop Pre-Requisites.pdf" (In STPCON2015 github repo) for more details

Android Platform (Mac & Windows)

- Java
- Android SDK
- Maven
- Intellij IDEA latest version
- Node / NPM
- Appium
- Android Emulator

iOS Platform (Only Mac)

- XCode Application
- Java
- Maven
- Intellij IDEA latest version
- Node / NPM
- Appium
- *iOS Simulators are installed automatically
  - Go to Xcode->Windows->Devices for adding more simulators

Which of following you worked with?

1. Appium Scripting w/ CI
2. Selenium Scripting w/ CI
3. Appium Scripting w/o CI
4. Selenium Scripting w/o CI
5. Only Appium UI and no scripting
6. Only Selenium IDE / Other UI based tools and no scripting
7. Only CI
8. Any Other Mobile/Web Automation

What you wanted to learn today?

How to automate Mobile testing?

Mobile Automation tools and technologies?

Parallel Test Execution?

Automation scripting?

Mobile Cloud and Automation?

Continuous Integration?

Why do you think Mobile Test Automation is important?

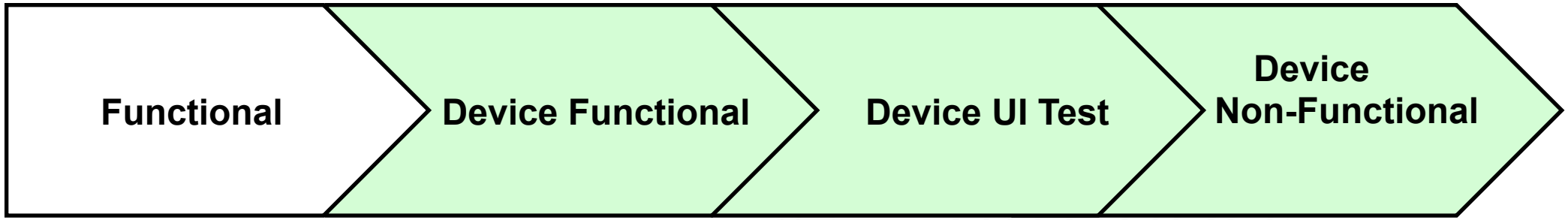
- Rapid Delivery?
  Does your organization has aggressive release plans for releasing new features?
- Manual interaction with small screens repeatedly?
- More complex mobile applications?
- Continuous Integration?
- Evidence collection for bug/issue is not easy?
- Exploding number of devices?

Mobile Automation Solutions Overview

2000 million users… Surpass desktop usage in 2014

What kind of mobile application?

Different Tools for Different Tests



	Functional	Device Functional	Device UI Test	Device Non-Functional
Description:	Test business function Business rules validation Data flows Business requirements	Test device functions Check capture Location Push notification	Test device UI Label positioning Error messages Image rendering Landscape/portrait	Test Device Limits High memory usage High processor usage Interruption test
Specific to Mobile?	**NO**	**YES**	**YES**	**YES**
Test Approach	**Simulators and Automation**	**Devices and Selective Automation**	**Devices and Selective Automation**	**Devices and Manual**

Industry Trends on Mobile Testing Solutions

Device Cloud Offerings

- Almost all automation tool vendors bundle device cloud offerings
- Services offerings are pretty similar
- Mostly support only iOS and Android Platform
- Device Functionality Testing – still an open issue device cloud solutions

Automation Tools

- Custom Scripting Solutions have not gotten the market approval
- Selenium Support as become as the de-facto standard for Mobile Automation
- Object Based Automation Support – with and with out software instrumentation

Shift Left & Integrations

- Continuous Integration (CI) Solutions support has become a MUST feature
- CI and Continuous Deployment (CD) is being demanded from many enterprises
- Exposure of Internal systems to Device Cloud is the BIGGEST challenge

Device Cloud - Key Players

Sauce Labs,
Perfecto Mobile

Mobile Labs,
Keynote Device
Anywhere

HP Mobility
Center

Appurify (Google)

NTT Docomo

Xamarin

TestDroid

Automation Tools – Technology Options

Selenium Support - Key For Success

Perfecto Mobile, Appium, Calabash, TestDroid, Appurify – Support Selenium

Non-Selenium

- Xamarin – C#
- Keynote – Custom Scripting, UFT (Recently Selenium)
- Calabash – Ruby
- Soasta – Custom Scripting

Appium – Popular Open Source Solution

Instrumentation Option (Source Code) still a requirement for some of the solutions

Mobile QA Solutions

What Does it Offers?

ON-DEMAND CLOUD

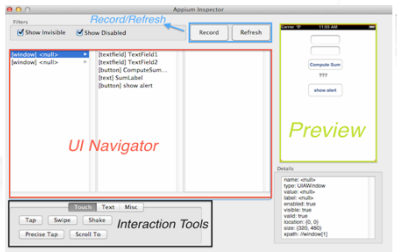
AUTOMATION SUPPORT

INTEGRATION WITH SUPPORT SYSTEMS

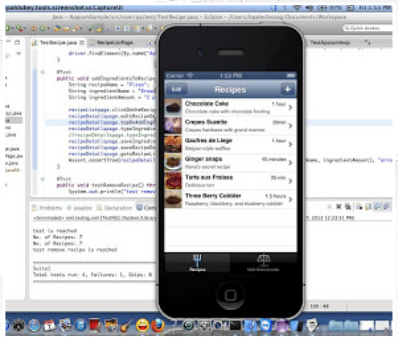
Mobile Automation Tool

What Do You Look For

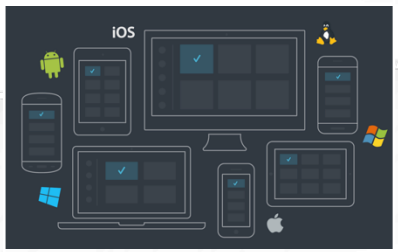
OBJECT RECOGNITION



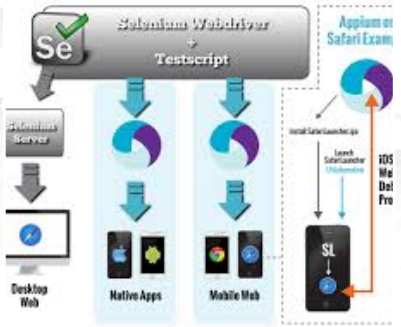
IDE Support



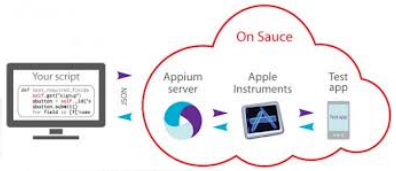
CROSS PLATFORM



EMULATOR / DEVICE



SCALABILITY SUPPORT



Appium Overview

Tool of the day!

Appium is an open source test automation framework which **automates, native, hybrid and mobile apps**.

PLEASE FIND THE Isaac Murchie's PRESENTATION HERE:
https://imurchie.github.io/stpcon-presentation/#/

Thanks to Isaac to spare some time to cover this topic during workshop and being present to support.

Appium – Hands On

Lets' try it.

Make sure you have and setup?

- JAVA setup
  - JAVA_HOME
    - Mac: ex: export JAVA_HOME=`/usr/libexec/java_home -v 1.8`
- Android SDK Home
  - ANDROID_HOME - eg: /opt/android-sdk
- App
  - For Android: ContactManager.apk
  - For iOS: UICatalog.zip
  - For Apps,
    - Copy SampleProject from USB stick and go to src/test/resources/app folder
      - (UPDATE: now FinalProject from github STPCon2015 repo)

Verify – Appium, Maven,JAVA Editor

Android Platform (Mac & Windows)

- Java -version
- Mac:
  - $ANDROID_HOME/tools/android
- Windows:
  - "%ANDROID_HOME%\tools\android"
- Mvn -v
- Intellij IDEA latest version
- Npm version
- appium -v
- Real device:
  - Adb devices
- Emulator:
  - Mac: $ANDROID_HOME/tools/emulator -avd STPCon2015
  - Windows: %ANDROID_HOME%\tools\emulator -avd STPCon2015
  - Adb devices

iOS Platform (Only Mac)

- XCode Application
- Java -version
- Mvn -v
- Intellij IDEA latest version
- Npm version
- Appium –v

Appium Configuration – Android Settings

Contact Manager App

Android Real Device

http://developer.android.com/tools/help/adb.html

In order to use adb with a device connected over USB, you must enable **USB debugging** in the device system settings, under **Developer options**.
On Android 4.2 and higher, the Developer options screen is hidden by default. To make it visible, go to **Settings > About phone** and tap **Build number** seven times. Return to the previous screen to find **Developer options** at the bottom.
On some devices, the Developer options screen may be located or named differently.

NOTE: Better to add ~android_sdk/tools in PATH system var

~android sdk/tools/adb devices – to see the list of devices

```
TechnocratMacPro:app TheTechnocrat$ adb devices
List of devices attached
a100c80b        device
```

Android Virtual Device (AVD) / Emulator

Start an emulator:

Mac: $ANDROID_HOME/tools/emulator -avd STPCon2015

Windows: %ANDROID_HOME%/tools/emulator -avd STPCon2015

NOTE: Better to add ~android_sdk/tools in PATH system var

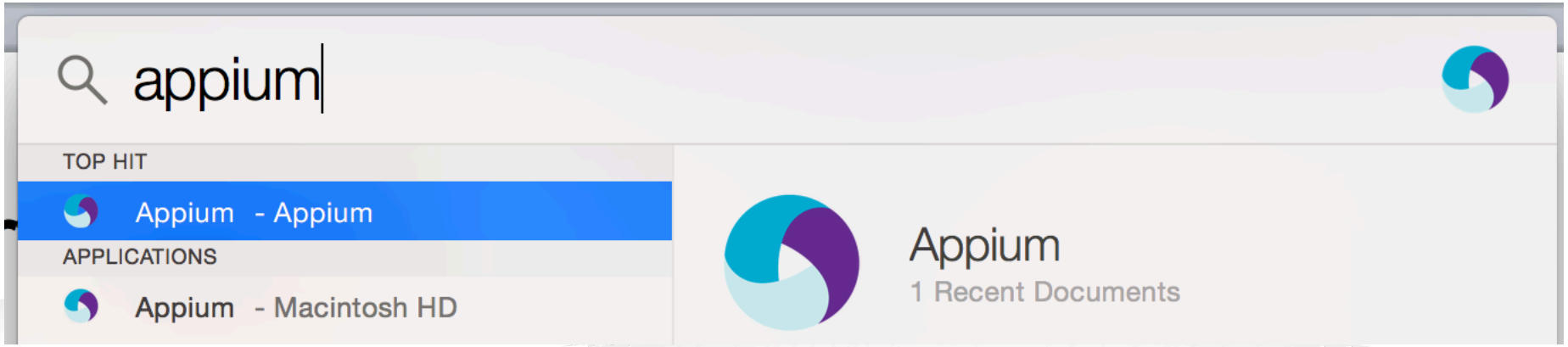
~android sdk/tools/adb devices – to see the list of devices

~android sdk/tools/android avd – android virtual device manager

```
TechnocratMacPro:~ TheTechnocrat$ adb devices
List of devices attached
emulator-5554   device
```

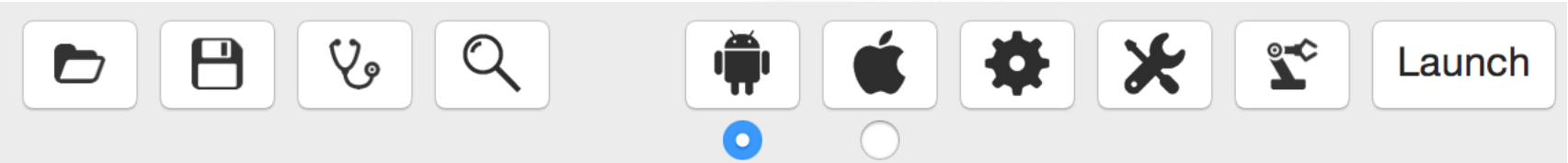
Start Appium

UI mode

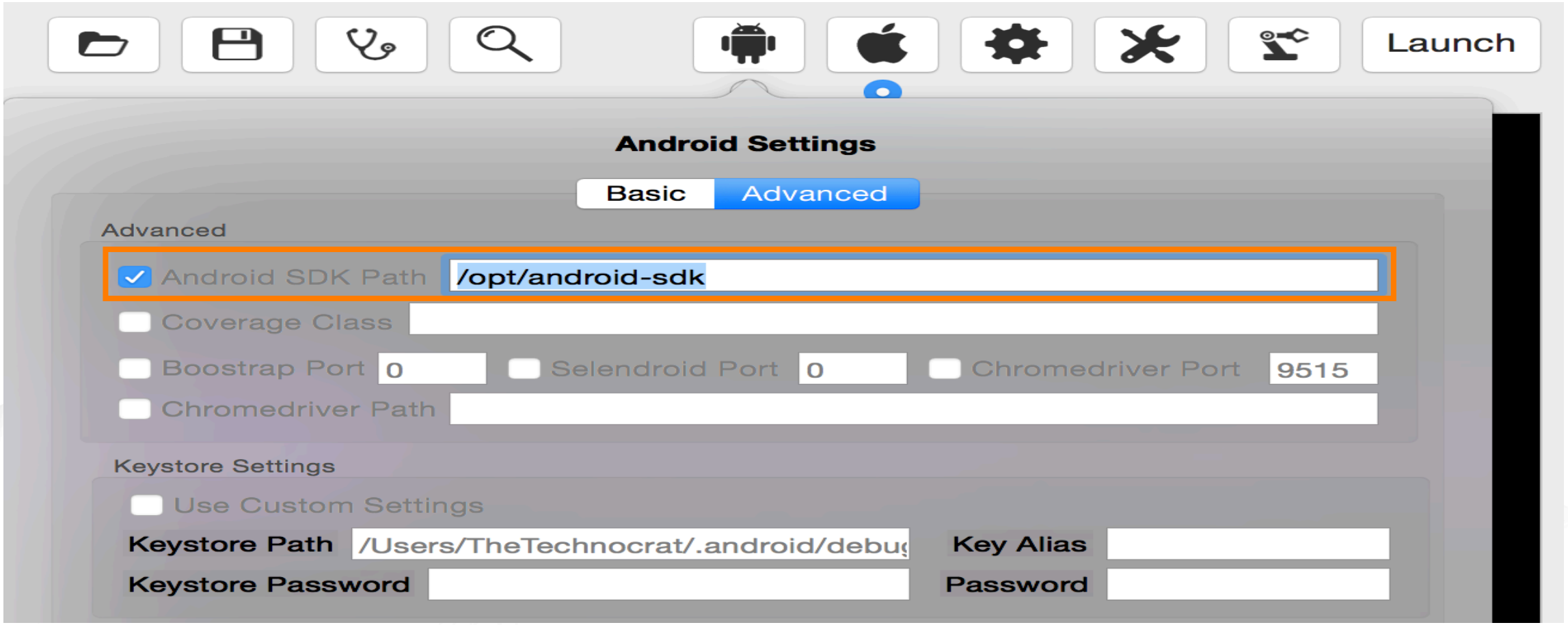


Check the installations…

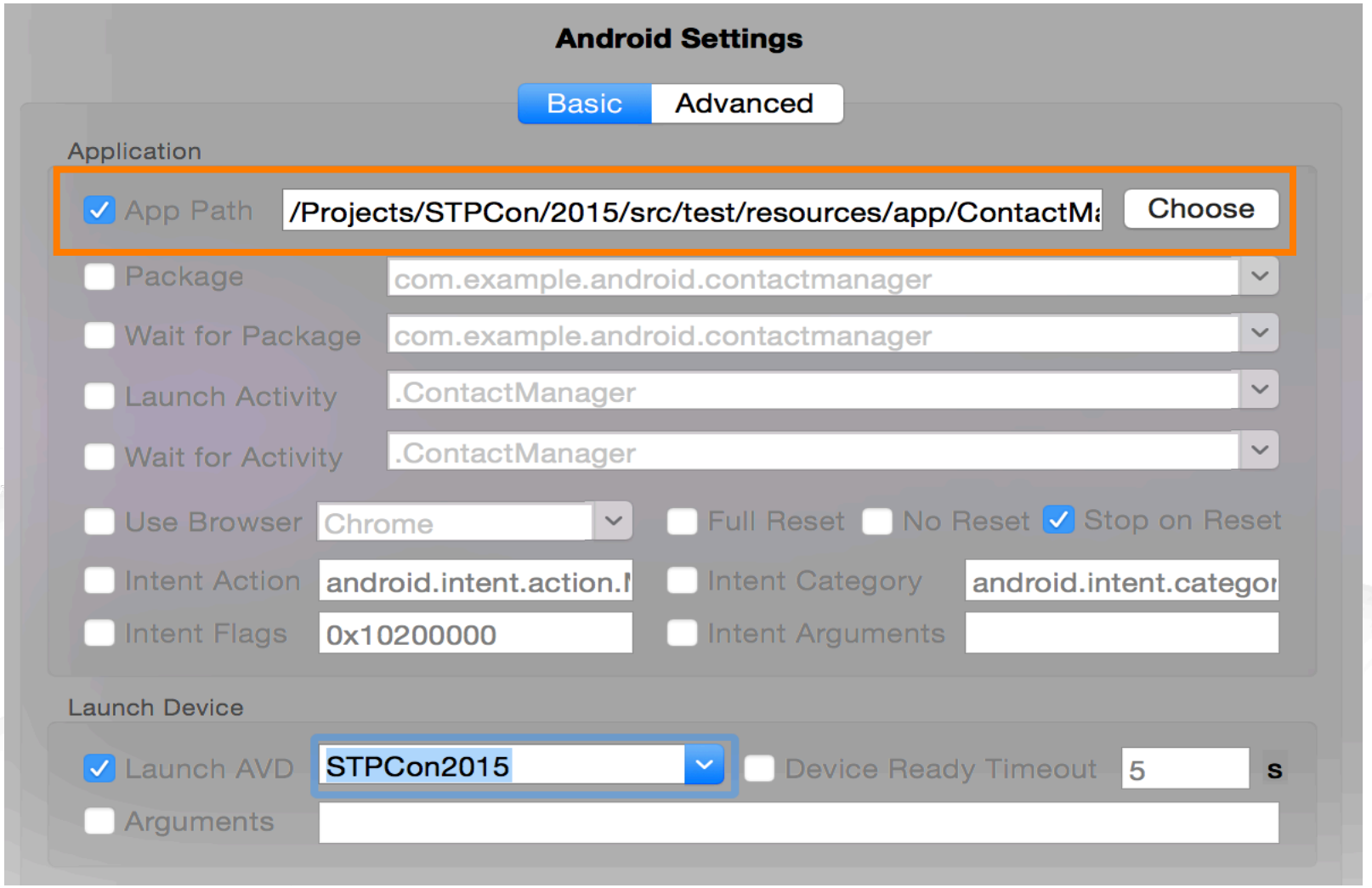
Quick look of tool bar



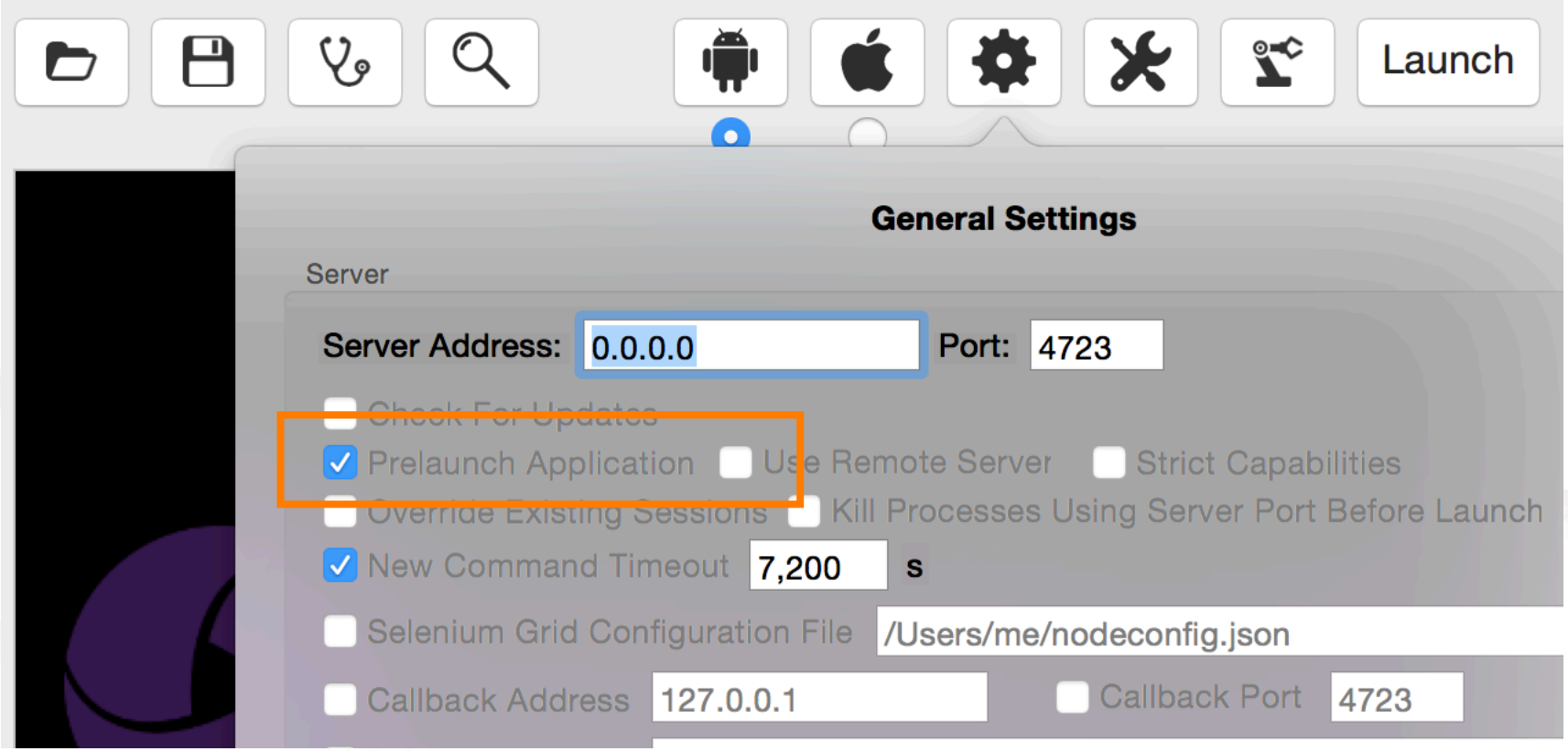
Android SDK & Advanced Settings



Android Settings Tab



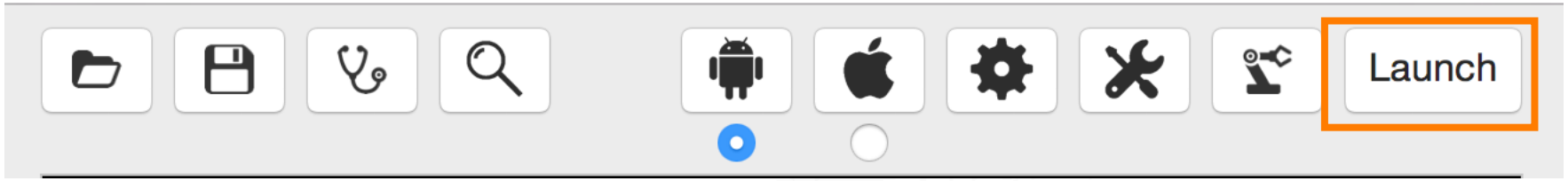
General Settings



Appium GUI Options

- http://appium.io/slate/en/master/?java#parameter-guide

Launch Appium



Verify
the application is loaded in devices/
already running emulator/
after launching emulator automatically

Appium Topics

Appium Locator Strategy

Appium Locator Strategy

id	• driver.findElement(By.id("buttonTest"));
name	• driver.findElement(By.name("buttonTestCD"));
Class Name	• driver.findElementsByClassName(“UIAStaticText”))
xpath	• driver.findElement(By.*xpath("// android.widget.Button[1]")*

Appium – Advanced Locator Strategy

- **Android uiautomator:** a string corresponding to a recursive element search using the - UiAutomator Api
- https://developer.android.com/tools/testing-support-library/index.html#uia-apis

Appium Desired Capabilities

Desired Capabilities

- Desired capabilities are a set of keys and values (i.e., a map or hash) sent to the Appium server to tell the server what kind of automation session

- http://appium.io/slate/en/master/?java#appium-server-capabilities

Desired Capabilities - Android

capabilities.setCapability(MobileCapabilityType.***AUTOMATION_NAME, "Appium");***

capabilities.setCapability(MobileCapabilityType.***PLATFORM_NAME, "Android Emulator");***

capabilities.setCapability(MobileCapabilityType.***PLATFORM_VERSION, "5.0");***

capabilities.setCapability(MobileCapabilityType.***APP,"<<APP URL>>");***

capabilities.setCapability(MobileCapabilityType.***APP_PACKAGE, "com.kayak.android");***

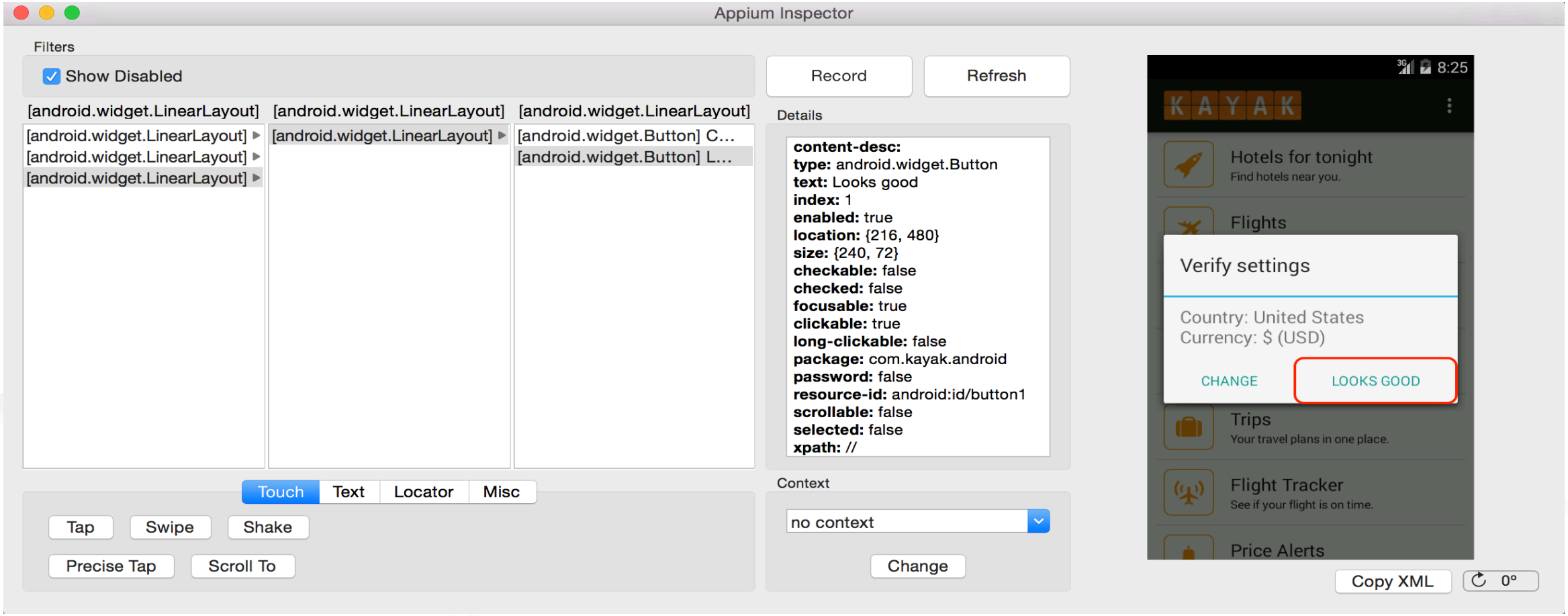
capabilities.setCapability(MobileCapabilityType.***APP_ACTIVITY, "com.kayak.android.Splash");***

Desired Capabilities Generator

- https://docs.saucelabs.com/reference/platforms-configurator/#/

Appium Inspector UI

Appium Inspector – Android KAYAK Application



Appium Device Interactions

Appium: Device Interactions

https://github.com/appium/java-client

Activity Management	• Start Activity • Current Activity
Device Actions	• Swipe, Zoom, Pinch, Shake • Touch Action, Multi Touch Action
App Management	• Install / Reset / Remove App

Appium: Device Interactions

- startActivity() / currentActivity()
- performTouchAction() / performMultiTouchAction()
- tap() / swipe()
- pinch() / zoom()
- installApp() / removeApp() / launchApp()
- isLocked() / lockScreen()
- shake()

Appium: Hybrid Applications

- Switching between native and Web view

```
Set<String> contextNames = driver.getContextHandles();
    for (String contextName : contextNames) {
    System.out.println(contextName);
    if (contextName.contains("WEBVIEW")){
            driver.context(contextName);
    }
 }
```

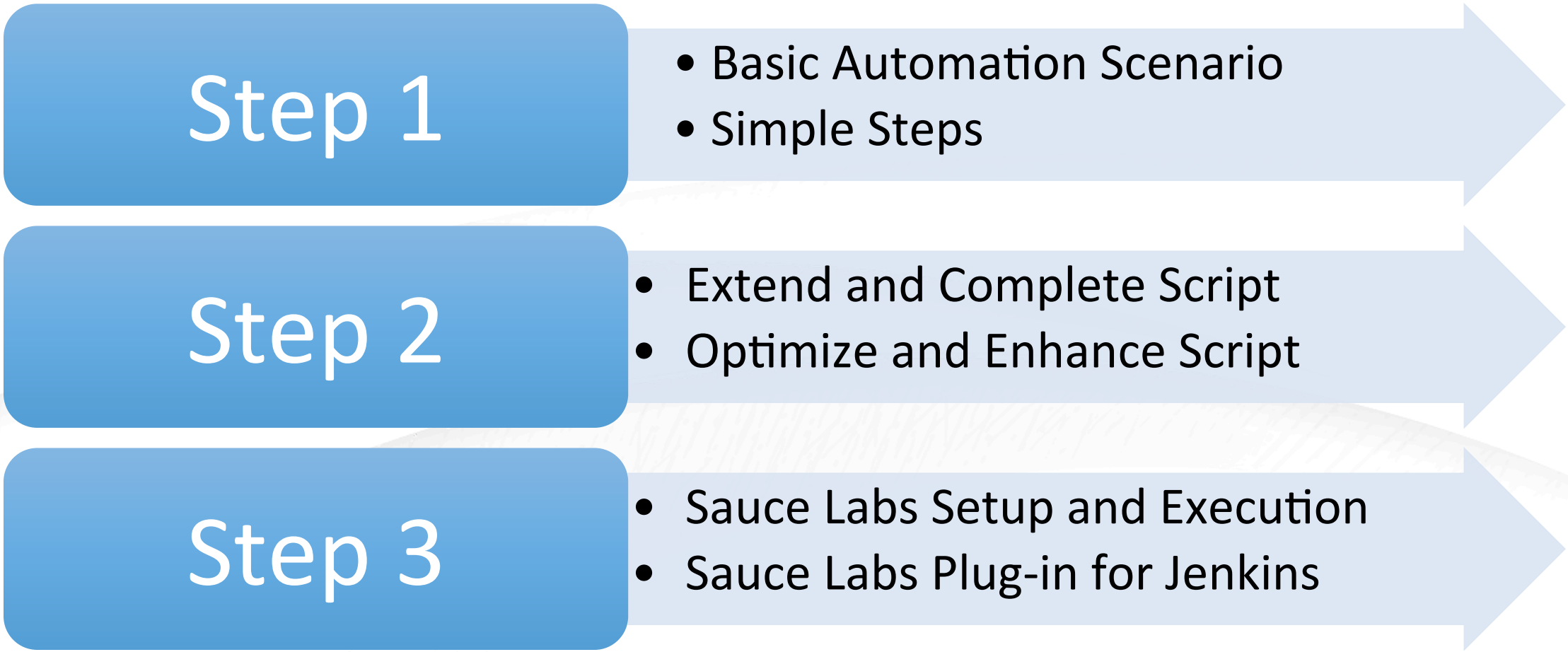
Kayak App Use Case

Android Platform

Kayak App Use Case

1. Launch Kayak App
2. Wait for Main Screen to load
3. Select Flights
4. Select One Way
5. Enter Search Details
    1. From and To Destination
    2. No. of Travelers, Cabin Class
6. Search For Flights
7. Select a Flight

Approach & Steps



Hands-on Scripting

Basic Scenario Scripting

1. Desired Capabilities
2. Load Application via Automation Script
3. Verify the application loading via Assert Statements
4. Introduce Intelligent Wait
5. Make Code / Function Re-usable
6. Capture Screenshot

Sample Project Setup

1. Extract the SampleProject.zip
    1. (UPDATE: now clone the STPCont2015 github repo and go to FinalProject)
2. Open this (SampleProject[Update:FinalProject]) project from IntelliJ IDEA by selecting pom.xml
3. Start Appium Server:
    1. For Androd: *appium -p 2500 --full-reset*
    2. For iOS (Only Mac Users): *appium -p 2600 --full-reset*

Optimize & Enhance Script

Optimization

- Centralized Locators
- Make Code Readable

Enhancement

- Advanced Features
- Swipe & Touch
- Device Operations

Example for iOS App

Sample App: UICatalog.zip

Sauce Labs Cloud & Jenkins Configuration

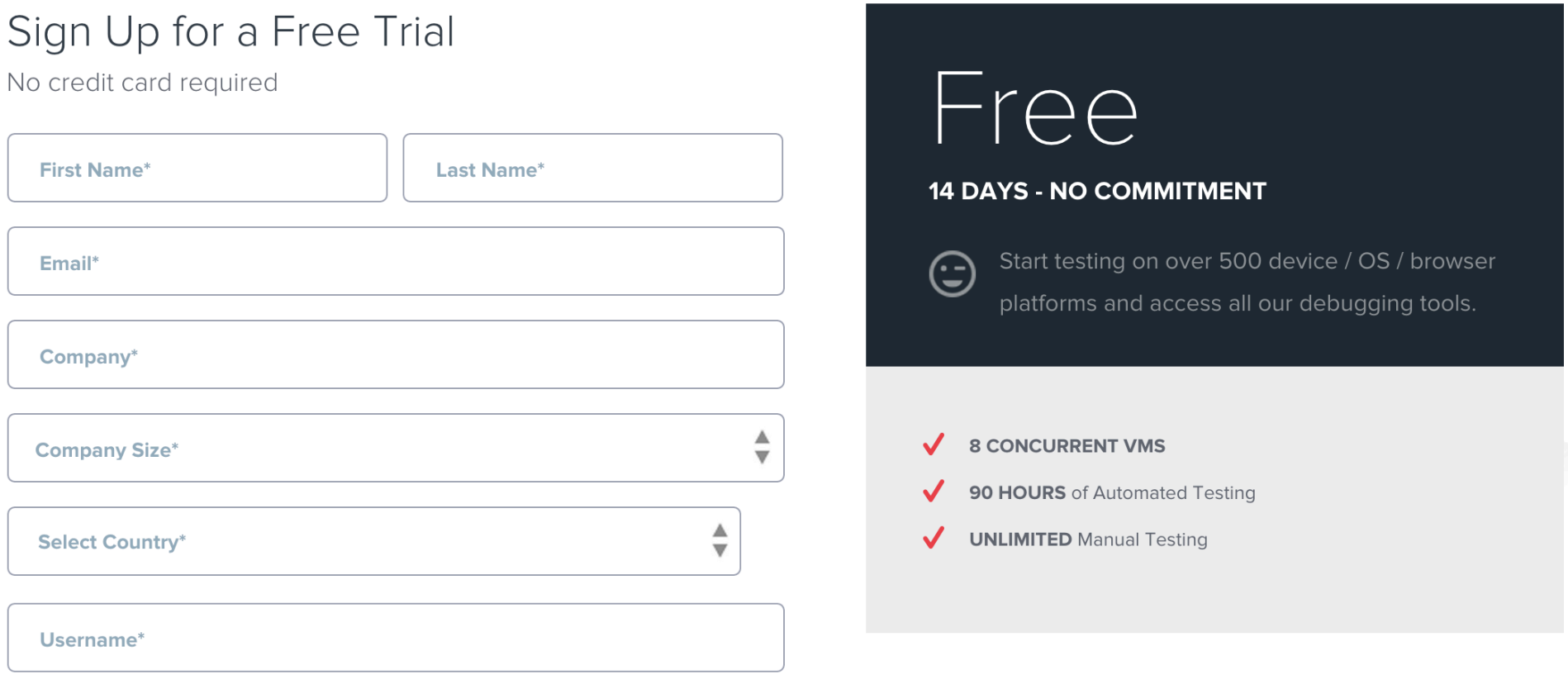
Sauce Labs Configuration

- https://docs.saucelabs.com/tutorials/appium/

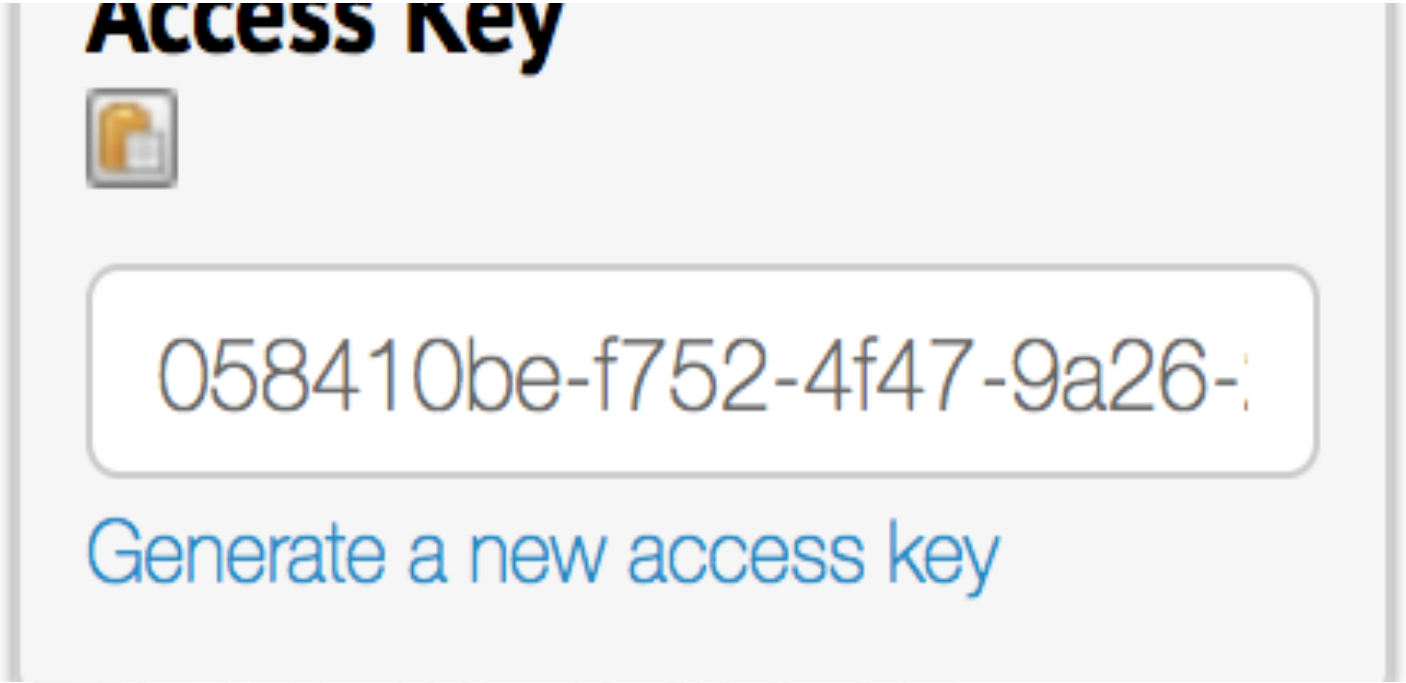
http://SAUCE_USERNAME:SAUCE_ACCESS_KEY@ondemand.saucelabs.com:80/wd/hub

- Mobile App:
  - Sauce Temporary Storage:
    - curl -u akhilkumar:058410be-f752-4f47-9a26-261482fbb7b3 \ -X POST \ -H "Content-Type: application/octet-stream" \ https://saucelabs.com/rest/v1/storage/akhilkumar/KAYAK-com.kayak.android-1126-v14.1.apk?overwrite=true \ --data-binary @KAYAK-com.kayak.android-1126-v14.1.apk
  - Git hub repository:

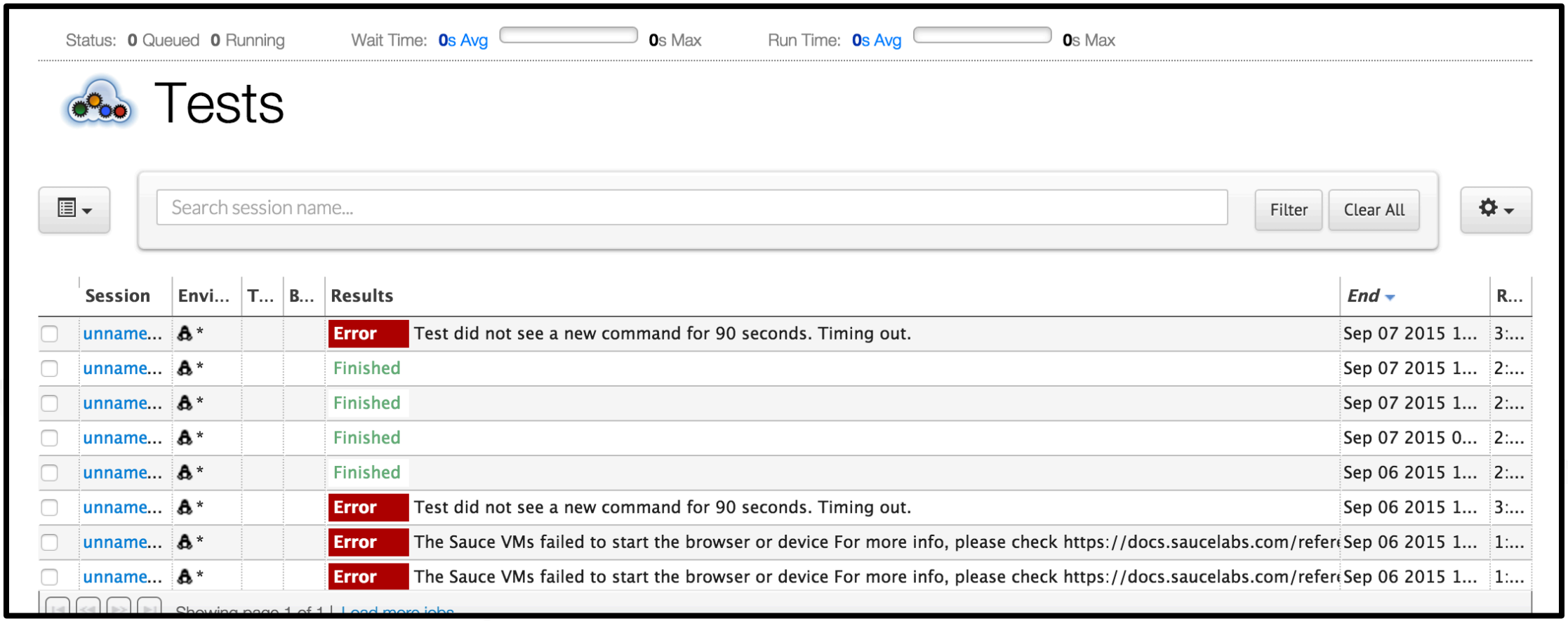
Sauce Labs: Trial Signup



Sauce Labs: API Access Key



Sauce Labs Execution



Parallel Execution

- Using TestNG “Parallel” Features

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="run tests" parallel="tests" verbose="1" configfailurepolicy="continue" thread-count="2">
  <test name="Android Test">
    <classes>
      <class name="com.stpcon.workshop.accelerated.mobile.qa.android.testrunner" />
    </classes>
  </test>
  <test name="iOS Test">
    <classes>
      <class name="com.stpcon.workshop.accelerated.mobile.qa.ios.testrunner" />
    </classes>
  </test>
</suite>
```

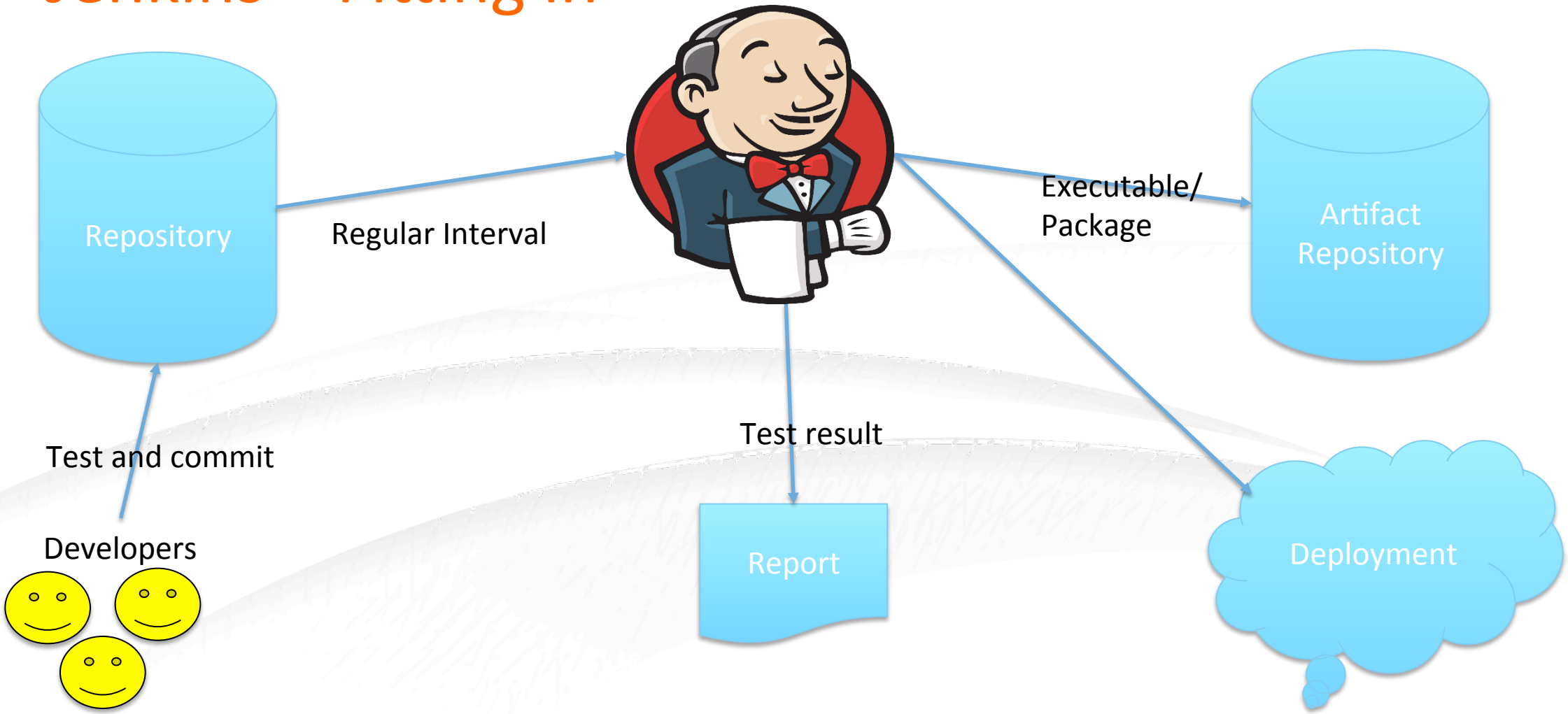
Jenkins Job Setup & Execution

CI – What does it really mean?

At a regular frequency (ideally at every commit), the system is:

- **Integrated** All changes up until that point are combined into the project
- **Built** The code is compiled into an executable or package
- **Tested** Automated test suites are run
- **Archived** Versioned and stored so it can be distributed as is, if desired
- **Deployed** Loaded onto a system where the developers can interact with it

Jenkins – Fitting in



How Jenkins works - Setup

- When setting up a project in Jenkins, out of the the following general options:
    - Associating with a version control server
    - Triggering builds
    - Polling, Periodic, Building based on other projects
- Execution of shell scripts, bash scripts, Ant targets, and Maven targets
    - Artifact archival
    - Publish JUnit test results and Javadocs
    - Email notifications

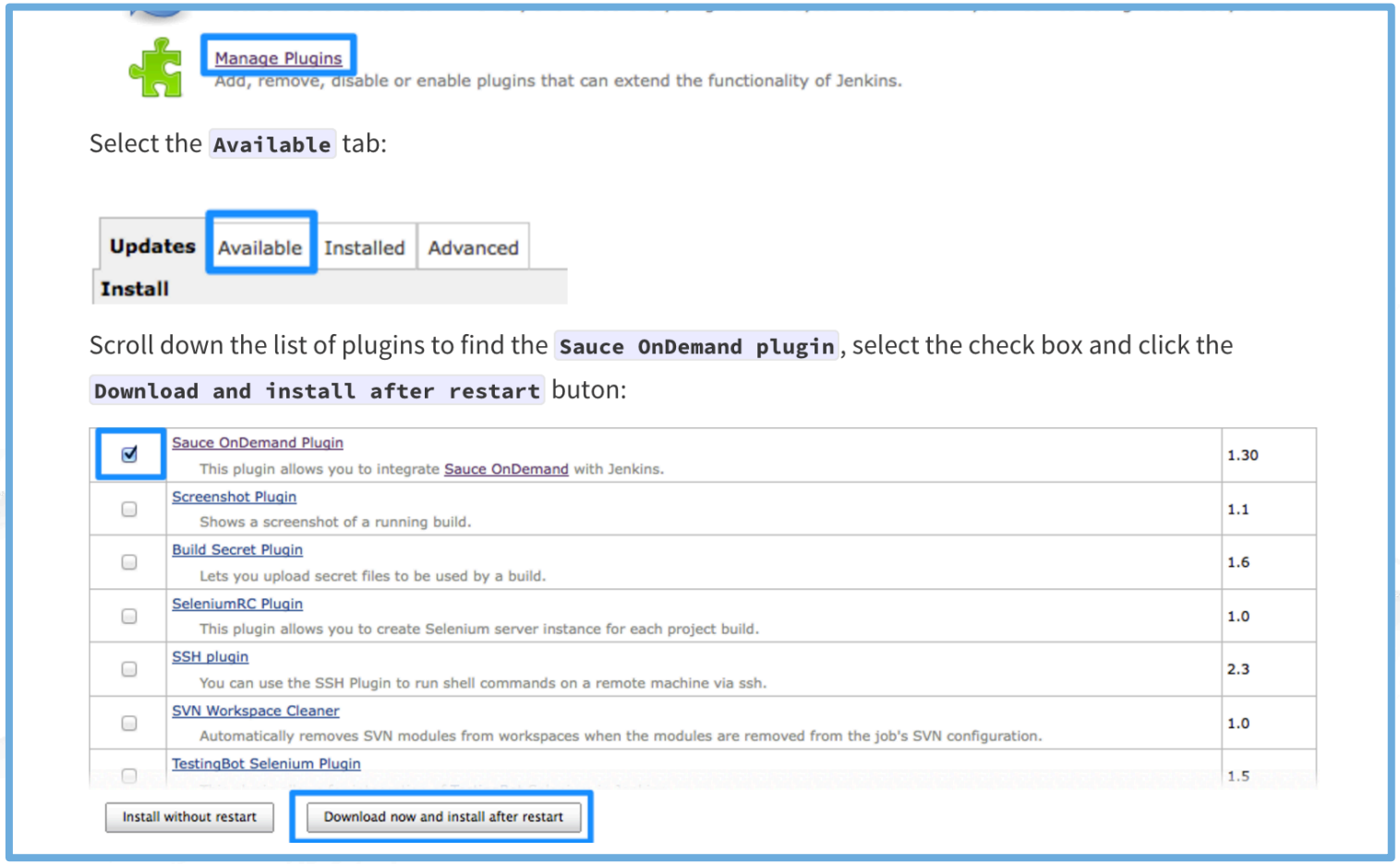
Jenkins Job Setup

- Java –jar jenkins.war
  - By default 8080 port
  - Localhost:8080
- Create New Job
  - New Item
  - Maven style Project
  - Provide the gloals and you can pass testng.suite.xml parameter

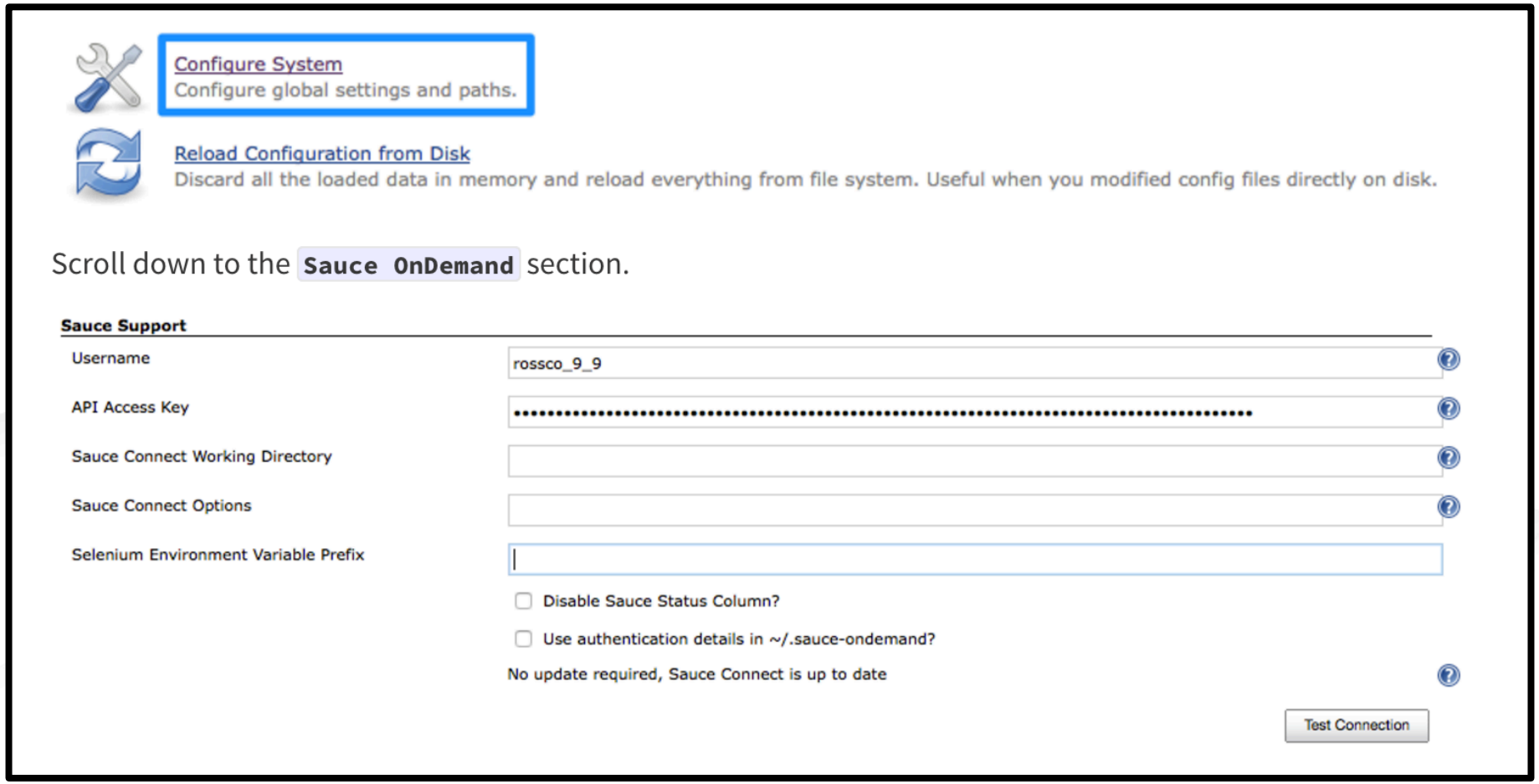
Jenkins & Sauce Labs

- Sauce Labs Plug-in (https://docs.saucelabs.com/ci-integrations/jenkins/)

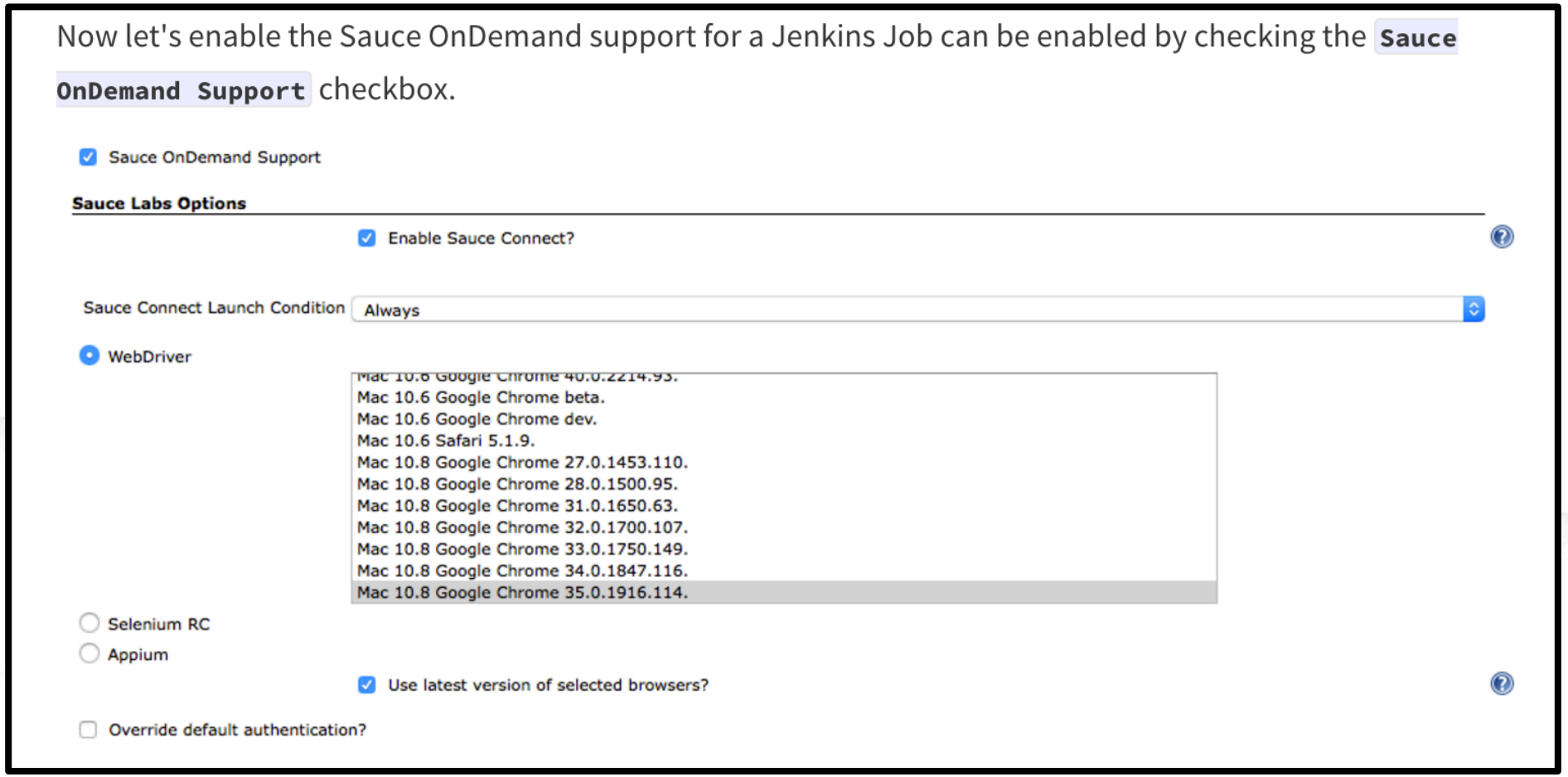
Sauce Labs Add-on Installation



Sauce Labs Add-on Configuration
- One-Time Setup



Configuring Jobs to run-against Sauce Labs



Configuring Jobs – Appium Desired Capabilities

The plugin will set a series of environment variables based on the information provided on the Job Configuration. These environment variables can either be explicitly referenced by your unit tests, or through the use of the [selenium-client-factory] library.

- `SELENIUM_HOST` - The hostname of the Selenium server
- `SELENIUM_PORT` - The port of the Selenium server
- `SELENIUM_PLATFORM` - The operating system of the selected browser
- `SELENIUM_VERSION` - The version number of the selected browser
- `SELENIUM_BROWSER` - The browser name of the selected browser.
- `SELENIUM_DEVICE` - The device name of the selected browser (only available for mobile browsers)
- `SELENIUM_DEVICE_TYPE` - The device type of the selected browser (only available for Appium browsers)
- `SELENIUM_DRIVER` - Contains the operating system, version and browser name of the selected browser, in a format designed for use by the Selenium Client Factory
- `SAUCE_ONDEMAND_BROWSERS` - A JSON-formatted string representing the selected browsers
- `SELENIUM_URL` - The initial URL to load when the test begins
- `SAUCE_USER_NAME` - The user name used to invoke Sauce OnDemand
- `SAUCE_API_KEY` - The access key for the user used to invoke Sauce OnDemand
- `SELENIUM_STARTING_URL` - The value of the `Starting URL` field

Takeaways

How to setup System for Mobile Automation

How easy to work with Appium?

How to write the scripts?

Why SauceLabs Cloud is helpful?

How quick to setup CI (atleast for tests).

Q & A

Please fill out the feedback form!

Thanks