Test Tool Strategies For Lone Testers
Test Leadership Congress 2019

Presented by Joshua Gorospe

Introduction

- I have been a tester for roughly 12 years.
  - I have worked in agency and product companies.
- I originally came from a background of working with very formal and traditional test teams.
  - Formal specifications were given then etc.
- During the last 3 years I became an omega tester (not by choice, just happened). I started studying and teaching myself Rapid Software Testing.
  - When it was available I would read through pieces of this every day. James Bach took it down after his recent website redesign...
    - http://www.satisfice.com/rst.pdf
  - I would often read through these too...
    - https://www.satisfice.com/download/a-context-driven-approach-to-automation-in-testing
    - https://www.satisfice.com/download/session-based-test-management
  - I just started reading "Lessons Learned in Software Testing: A Context-Driven Approach". It has become my favorite book for self-study.
- I enjoy experimenting with test tools.
- I like considering myself as simply a tester, I am proud of it.
- I personally believe that automation is only a part of a tester's toolbox, it's not everything.
- Testing, designing test strategies, and finding bugs makes me feel like a detective scientist.
  - Funny fact: I Googled "detective scientist" and found this... https://www.nasa.gov/audience/forstudents/nasaandyou/home/detective_en-index.html

Before we dive into it. This interactive presentation is…

- **NOT** an exhaustive list of all possible test strategies that can be applied to any project or product.
- **NOT** trying to give you silver bullets.
    - Your project, product, or situation may be unique
    - Your tester journey, experience, or career path may be completely different
    - "Only you know the context of your own work." (Lessons Learned in Software Testing: A Context-Driven Approach)

- An evolving and ongoing experiment of mine. I enjoy learning new things.
- Going to attempt to try to give you some new idea(s) in 45 minutes (maybe less?).
- Going to frequently mention some tools that the presenter is familiar with, and enjoys experimenting on.
    - Docker
        - https://www.docker.com/why-docker
    - Robot Framework
        - https://robotframework.org
        - Generic test framework that uses natural language syntax
        - Currently, one of the few free and open source RPA (robotic process automation) solutions
            - https://en.wikipedia.org/wiki/Robotic_process_automation
            - http://robotframework.org/rpa/
    - **Many of these examples can be implemented using any other test framework, approach, etc.**
        - Maybe you like to write all of the code yourself (kudos to you)

Maybe this seems familiar to some of you.
Ever encounter simultaneous projects that feel like a combination of...

...this?

...and this?

Before the lone tester (you) shows up…

The projects…

- Are **fast paced**
- Are **frequently changing**
  - Requirements
  - Scope
  - Client wants more features
- Have a client who…
  - Thinks that “Agile” means going to production as soon as possible…
  - Thinks software development is “just build it once and you’re done”
  - Is furious when they notice one bug or issue slip by to production
- Have too many meetings
- Have possibly more than just your team working on it
  - Different companies or agencies are also working on critical parts
  - Waiting for that other company or agency to get their part done

Your development team…

- Is probably spread out across several projects at once
- Is several months into development
- Probably has a **“wild west” or use-anything-you-want approach to testing**
- Uses different types of tools for testing
- Testing is performed only in a Dev environment
  - Or maybe only on a developer’s machine

After you jump into the biggest one out of those projects...

You will probably notice quickly.

- "Anything you do means something else won't get done."
    - "You feel too busy to plan and prepare."
        - "You feel too busy to mind your infrastructure."
            - ^ These are the challenges that you, "Omega Tester" (a lone tester on several projects), will face.

"Omega Tester: Testing with a Team of One", James Bach, http://www.satisfice.com/articles/omega_tester.pdf

Some suggestions from James Bach's "Omega Tester: Testing with a Team of One"

- "… **Think test activities and the testing story.**"
  - "… Configuring the product, operating it, observing it, and evaluating it-- that's testing. Because you're overloaded, I suggest **thinking about your strategy in terms of what you need to do, or what you need to create in order to do it**. A detailed documented test procedure is usually neither of those things. **Focus on the activity of testing…**"
- "Test in short, sharp, uninterrupted **sessions.**"
  - "**Session-based testing** means that you set aside a block of time-- I prefer 90-minute blocks-- put a sign up at your desk that says "Gone Bug Finding" or something like that, **pick a risk or an area of the product**, and test it. Record your testing charter (a sentence or two about the objective of the session), too. This is important because as a lone tester you need to be more definite about the progress you make"
- "**Use unexpected lulls for infrastructure, preparation**, or re-assessment."
  - "**…**You need to take those opportunities to look over your strategy, try to think ahead…"
- "**Use recording tools to shorten bug investigation time.**"
  - "They say that scripted testing is better than exploratory testing when it comes to bug investigation. This is completely untrue. Having a list of steps that you produced two months ago is no more helpful than a list you made as you went along, except **you are two months wiser, today, and the notes you record today are fresh in your mind**…"

"Omega Tester: Testing with a Team of One", James Bach, http://www.satisfice.com/articles/omega_tester.pdf

Before we move on to the examples… FYI to Windows users

All of the examples in this presentation use Bash. Here are some possible options that may help you, **but I have not tested them myself**…

- https://gitforwindows.org/
- https://www.cygwin.com/
- http://babun.github.io/

Presentation Examples

Are organized into the following parts...

- Part One - Tool strategy examples for **rapidly testing** and **adapting to frequently changing projects**

- Part Two - Tool strategy examples for **enhancing existing tests** and **taking your test process further**

- Part Three - Combined results **visualizing your testing story**

The examples in Part One are building up to demonstrations in Part Two, and these examples will encourage you towards "**mixing scripting and exploration**". This idea comes from Michael Bolton and James Bach's "**Rapid Software Testing**" course, slide 97, **http://www.satisfice.com/rst.pdf**

Presentation Outline

Here is a high-level outline of the examples that will be covered, and what we will do when we get through all of the parts...

- Part One - Tool strategy examples for **rapidly testing** **and** **adapting to frequently changing projects**
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-One-Postman-Newman-Workshop-Examples
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-One-cURL-Workshop-Examples
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-One-Requests-Library-Workshop-Examples
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-One-Python-Library-Workshop-Examples
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-One-Run-All-Docker-Workshop-Examples

- Part Two - Tool strategy examples for **enhancing existing tests** **and** **taking your test process further**
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-Two-Postman-Newman-Workshop-Examples
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-Two-Python-Library-Workshop-Examples
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-Two-Requests-Library-Workshop-Examples
  - ./start-specific-docker-example-workflows-for-workshop.sh Part-Two-cURL-Workshop-Examples
  - ./start-specific-docker-example-workflows-for-workshop.sh Long-Graphwalker-Run-Workshop-Example
  - ./start-specific-docker-example-workflows-for-workshop.sh Short-Graphwalker-Run-Workshop-Example
  - ./start-specific-docker-example-workflows-for-workshop.sh Previous-Long-Graphwalker-Run-Workshop-Example
  - ./start-specific-docker-example-workflows-for-workshop.sh Previous-Short-Graphwalker-Run-Workshop-Example
  - ./start-specific-docker-example-workflows-for-workshop.sh Remote-Test-Process-Triggered-By-A-Webhook-Docker-Example
  - ./start-specific-local-machine-example-workflows-for-workshop.sh Hybrid-Tool-Desktop-Workshop-Example
  - ./start-specific-local-machine-example-workflows-for-workshop.sh Hybrid-Tool-Desktop-Charles-Proxy-Workshop-Example
  - Using Appium Desktop to automatically generate scripts that can be used for automation, while using the Hybrid-Tool-Desktop-Workshop-Example on an iOS simulator

- Part Three - Combined results **visualizing your testing story**
  - ./start-specific-result-gathering-example-workflows-for-workshop.sh Combined-Results-Dashboard-Workshop-Example
  - ./start-specific-result-gathering-example-workflows-for-workshop.sh Metrics-Dashboard-Workshop-Example
  - I will also demonstrate how to manually deploy a new dashboard for a Long-Graphwalker-Run-Workshop-Example using Heroku

- Answer questions etc.

Part One - Tool strategy examples for **rapidly testing** and **adapting to frequently changing projects**

The examples in Part One deal with a hypothetical project(s) scenario introduced in the earlier slides. Developers working on projects could decide to use tools they are most comfortable and confident with. As a lone tester trying to learn a project as quickly as possible while also deeply testing Jira Tasks or Pull Requests, it may be helpful to reuse the test approaches that developers are already using and be able to easily run them locally or in CI pipelines (using Docker). Robot Framework makes it easy to achieve this while allowing us to prepare for interesting and useful enhancements that we will see in Part Two.

- Postman Newman tool is an example of a popular API testing tool with convenient code generation, API documentation, test running features, it even has API monitoring mode, and other useful features.
    - https://learning.getpostman.com/docs/postman/launching_postman/installation_and_updates/

    - Here is an example of Robot Framework triggering specific tests using a Postman Newman collection with the "--folder" option while capturing the results.
        - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Postman-Newman.robot

- cURL is an example of a very popular and powerful general purpose command line tool for transferring data using various protocols. Manually exploring can be as simple as running various cURL one-liners.
    - https://curl.haxx.se/docs/

    - Here are examples of Robot Framework triggering tests in serial and parallel using cURL while capturing the response.
        - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Curl1.robot

        - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Curl2.robot

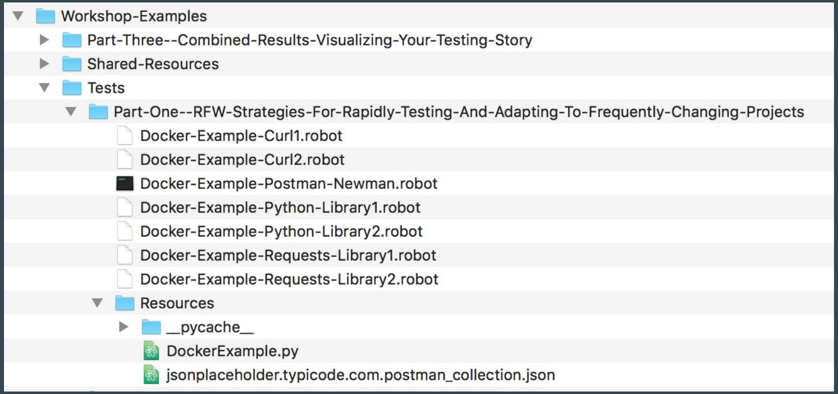
Part One - Tool strategy examples for rapidly testing and adapting to frequently changing projects

As projects are progressing, a lone tester may be able to convince the team to assist with some of the testing activities. For example, a tester could convince developers to help with creating more regression tests. The syntax of Robot Framework may cause certain developers to get confused or not understand it. Luckily Robot Framework offers several options to utilize user-created libraries to reduce that confusion (natively and through Remote Interface). Also having an easy to understand folder structure can help someone quickly learn how the tests work and jump in to help you more quickly. Of course, a lone tester who is also a savvy programmer can probably do these user-created library activities with very little assistance.

- Here is an example utilizing a user-created Python library with Robot Framework that is located in the same folder as the .robot file, in a folder named "Resources". This example runs in serial and in parallel.
    - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Python-Library1.robot

    - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Python-Library2.robot

- FYI - there is also a Java option for user-created libraries, but it is not covered in today's workshop. There are several examples of using Robot Framework with Java in the Robot Framework User Guide and Thomas Jaspers wrote an excellent article about that as well, https://blog.codecentric.de/en/2012/06/robot-framework-tutorial-writing-keyword-libraries-in-java/



```
▼ Workshop-Examples
    ▶ Part-Three--Combined-Results-Visualizing-Your-Testing-Story
    ▶ Shared-Resources
    ▼ Tests
        ▼ Part-One--RFW-Strategies-For-Rapidly-Testing-And-Adapting-To-Frequently-Changing-Projects
              Docker-Example-Curl1.robot
              Docker-Example-Curl2.robot
              Docker-Example-Postman-Newman.robot
              Docker-Example-Python-Library1.robot
              Docker-Example-Python-Library2.robot
              Docker-Example-Requests-Library1.robot
              Docker-Example-Requests-Library2.robot
            ▼ Resources
                ▶ __pycache__
                  DockerExample.py
                  jsonplaceholder.typicode.com.postman_collection.json
```

Part One - Tool strategy examples for **rapidly testing** and **adapting to frequently changing projects**

The opposite of the previous example could also happen. A lone tester may not be able to convince the team to assist with any of the testing activities. In this case, a tester could be completely comfortable with the keyword driven syntax of Robot Framework and may be able to convince the team to use this approach. This could result in a single .robot test file containing all of the keywords, or as the amount of keywords grows over time they can be kept in the "Resources" folder in the "Tests" folder using either .robot or .resource file extensions.

- Here is an example utilizing the robotframework-requests library, that is provided by the Robot Framework community. This example runs in serial and in parallel.
  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Requests-Library1.robot
  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Requests-Library2.robot

Part One - Tool strategy examples for **rapidly testing** and **adapting to frequently changing projects**

Now lets get hands on...

- Please proceed to https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019

- Please feel free to try the examples on your machines (if you have not yet done so already).
  - **Mac and Linux users**: you should set up a .env file on your machines
  - **Windows users:** you can hard-code those values in the docker-compose file

- I am going to run and explain all of the following Part One examples as they are running.

  - **./start-specific-docker-example-workflows-for-workshop.sh Part-One-Postman-Newman-Workshop-Examples**

  - **./start-specific-docker-example-workflows-for-workshop.sh Part-One-cURL-Workshop-Examples**

  - **./start-specific-docker-example-workflows-for-workshop.sh Part-One-Requests-Library-Workshop-Examples**

  - **./start-specific-docker-example-workflows-for-workshop.sh Part-One-Python-Library-Workshop-Examples**

  - **./start-specific-docker-example-workflows-for-workshop.sh Part-One-Run-All-Docker-Workshop-Examples**

Part Two - Tool strategy examples for **enhancing existing tests** and **taking your test process further**

The first group of examples in Part Two make a few enhancements to the examples in Part One. Hopefully they will give you ideas to experiment with as well...

- Part Two of the Postman Newman example has been enhanced by using the SharedKeywordsAndListeners.robot resource file to add 2 types of Robot Framework listeners and a team notification keyword (for now it only uses Slack). Also added the "--randomize tests" option in the commands-running-inside-a-docker-container.sh script, and will demonstrate the use of the "--rerunfailedsuites".
  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Postman-Newman-Enhanced-Version.robot

- Part Two of the Python Library example has been enhanced by using the SharedKeywordsAndListeners.robot resource file to add 2 types of Robot Framework listeners and a team notification keyword (for now it only uses Slack). Also added the "--randomize all" option in the commands-running-inside-a-docker-container.sh script, and will demonstrate the use of the "--rerunfailedsuites".
  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Python-Library1-Enhanced-Version.robot

  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Python-Library2-Enhanced-Version.robot

- Part Two of the cURL examples has been enhanced by using the SharedKeywordsAndListeners.robot resource file to add randomly generated malformed data to a test through the "Create Random Malformed Test Data With Radamsa" keyword. This keyword turns the cURL examples into an API fuzz testing tool that can run in serial or in parallel.
  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Curl1-Enhanced-Version.robot

  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Curl2-Enhanced-Version.robot

- Part Two of the robotframework-requests library examples has been enhanced by using the SharedKeywordsAndListeners.robot resource file to add randomly generated data to a test through the "Create Various Random Test Data With The String Library" keyword. This keyword turns the robotframework-requests library examples into an API fuzz testing tool that can run in serial or in parallel.
  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Requests-Library1-Enhanced-Version.robot

  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Requests-Library2-Enhanced-Version.robot

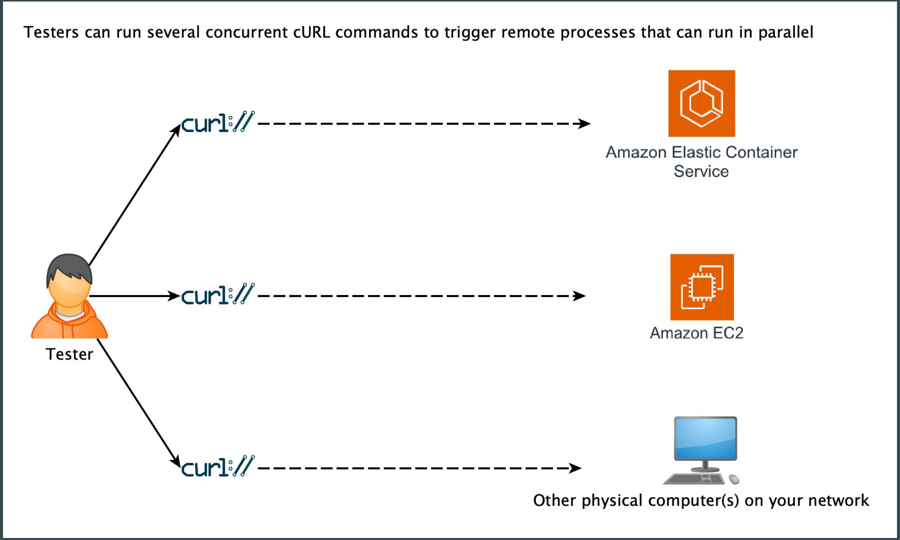
Part Two - Tool strategy examples for **enhancing existing tests** and **taking your test process further**

The following takes one of the examples in Part Two and adds the capability of running tests as a webhook triggered remote process.

- This example can be run remotely from...
  - AWS Elastic Container Service (ECS)
  - Remote environment or Amazon Elastic Compute Cloud (Amazon EC2)
  - Other computer(s) located on the same network in your office

- Any of the Docker examples in Part One or Two can also have this capability. These are the components used to set up and run this demonstration.
  - https://github.com/adnanh/webhook
  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/multiple-webhooks.json
  - https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Dockerfile.RemoteTestWebhook
  - ./Start-specific-docker-example-workflows-for-workshop Remote-Test-Process-Triggered-By-A-Webhook-Docker-Example

- After the Docker container starts running, the following cURL command can be used to trigger the remote test process...
  - curl -i http://0.0.0.0:9080/hooks/robot-framework-remote-test-process-webhook


Testers can run several concurrent cURL commands to trigger remote processes that can run in parallel
Tester
curl://
curl://
curl://
Amazon Elastic Container Service
Amazon EC2
Other physical computer(s) on your network

Part Two - Tool strategy examples for **enhancing existing tests** and **taking your test process further**

The second group of examples in Part Two has a combination of approaches (random test data generation, Slack notifications, 2 types of Robot Framework listeners) from the enhanced Part One examples and also uses model-based testing. This combination of approaches is placed into one .robot file. The keywords are triggered by a test path (for this workshop, it's a .csv file) generated by a model-based test tool called Graphwalker.

- If this is your first time hearing about Graphwalker, you can find more information about it here, http://graphwalker.github.io/

- Graphwalker has its own DSL and uses various model-based testing techniques to generate test sequences.
  - Here is more information about the types of path generators, http://graphwalker.github.io/generators_and_stop_conditions/

Here is Robot Framework + Graphwalker Docker example...

- https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Robot-Framework-Graphwalker.robot

- There are 4 types of Robot Framework + Graphwalker Docker demonstrations in the repo...
  - **[1]** Do a long run of Graphwalker + Robot Framework while using a slow internet proxy (https://github.com/bcoe/crapify) in a Docker container
    - By running the following...
    - **./start-specific-docker-example-workflows-for-workshop.sh Long-Graphwalker-Run-Workshop-Example**

  - **[2]** Do a re-run using the same Graphwalker path generated in **[1]**
    - By running the following...
    - **./start-specific-docker-example-workflows-for-workshop.sh Previous-Long-Graphwalker-Run-Workshop-Example**

  - **[3]** Do a short run of Graphwalker + Robot Framework with more vertices than edges, and running without the slow internet proxy in a Docker container
    - By running the following...
    - **./start-specific-docker-example-workflows-for-workshop.sh Short-Graphwalker-Run-Workshop-Example**

  - **[4]** Do a re-run using the same Graphwalker path generated in **[3]**
    - By running the following...
    - **./start-specific-docker-example-workflows-for-workshop.sh Previous-Short-Graphwalker-Run-Workshop-Example**

Part Two - Tool strategy examples for **enhancing existing tests** and **taking your test process further**

The third type of example in Part Two uses a hybrid tool approach inspired by both Session-based Test Management and James Bach and Michael Bolton's "A Context-Driven Approach to Automation in Testing". If you have no idea what any of that is, I encourage you to read the following...

- http://www.satisfice.com/sbtm/
- http://www.satisfice.com/articles/cdt-automation.pdf

The following is the hybrid tool example.

- https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Hyrid-Session-Based-Test-Example-For-Desktop.robot

Before running these examples here are some suggestions and ideas that I like to use, but feel free to explore using your own approaches. That is part of the gist of Session-based Test Management. YOU are in full control of the testing session.

- One thing I like to do to set up for a session is run some git log commands such as the following...
  - **git log --since=X.weeks --grep='bug' --grep='fix' --abbrev-commit**
  - **git log --since=X.days --grep='bug' --grep='fix' --abbrev-commit**
  - Here is an article that describes that approach in more detail, http://google-engtools.blogspot.com/2011/12/bug-prediction-at-google.html

- Another thing I like to do is use a tool called GitRisky. It uses binary classification machine learning to assign bug risk scores to any git commit that you give it. I combine GitRisky with those git log commands in the example below. FYI, the **GitRisky training step may take a very long time** on your machine if you are feeding it a lot of git log data. Also, **your git commit messages must have at least the word "fix"** if the commit is meant to fix something.
  - *First step, install it on a machine or Docker container running Python 3.5->* https://github.com/hinnefe2/gitrisky
    - **WARNING: You will get Python deprecation and package errors if you do not use the correct Python version for GitRisky.**
  - *Second step, run these commands. Worst case is it could take an hour. ->* **cd /path/to/your-repo-name && gitrisky train**
  - *Third step, run these commands. ->* **for i in $(git log --since=40.weeks --grep='bug' --grep='fix' --pretty=format:%h | uniq -c | awk '{print $2}'); do echo && gitrisky predict -c $i; done**
  - **A convenient Shell script has been provided in the Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019 GitHub repo. It will run that Shell one-liner with a prompt, it can be run from your Terminal (or command line) like this...**
    - **./gitrisky-oneline-example.sh**
    - **We will use this Mule Community Edition repo as an example for GitRisky to analyze,** https://github.com/mulesoft/mule

If you are testing a website. Here are some interesting Chrome Extension tools to use concurrently in your testing session as you are running these examples.

- https://github.com/olsh/monkey-testing
- Gojko Adzic's https://bugmagnet.org/

Part Two - Tool strategy examples for **enhancing existing tests** and **taking your test process further**

Let's quickly walk through Charles Proxy and combining its use with the upcoming Hybrid-Tool-Desktop-Workshop-Example demonstration...

- To find more information about Charles Proxy please check out the following.
  - https://www.charlesproxy.com/
  - https://www.charlesproxy.com/documentation/configuration/browser-and-system-configuration/
  - https://www.charlesproxy.com/documentation/faqs/using-charles-from-an-iphone/
  - https://medium.com/@hackupstate/using-charles-proxy-to-debug-android-ssl-traffic-e61fc38760f7

- I am going to run Charles Proxy while interacting with an iPhone application on a real device.
  - We will observe and filter out the logs generated after the device is connected through Charles Proxy.
  - One of its main features is providing MITM (man-in-the-middle) HTTPS proxying, and basically proving live monitoring for all HTTP traffic coming from the application.
  - There are 15 other useful features of Charles Proxy listed here.

    https://www.charlesproxy.com/documentation/tools/

- As I am simulating a testing session, I will use the following hybrid tool to capture information about my session (screenshots, notes, logs from Charles Proxy etc.)
  - **./start-specific-local-machine-example-workflows-for-workshop.sh Hybrid-Tool-Desktop-Workshop-Example**

Part Two - Tool strategy examples for **enhancing existing tests** and **taking your test process further**

Let’s quickly walk through Charles Proxy, Appium Desktop, and combining its use with the upcoming Hybrid-Tool-Desktop-Workshop-Example demonstration…

- To find more information about Appium Desktop please check out the following.
  - http://appium.io/downloads.html
  - https://saucelabs.com/resources/webinars/an-introduction-to-appium-desktop
  - https://bitbar.com/blog/use-appium-desktop-to-boost-your-appium-efficiency/
  - https://medium.com/@eliasnogueira/inspect-an-app-with-the-new-appium-desktop-8ce4dc9aa95c

- I am going to run Charles Proxy while interacting with an iPhone application on a simulator while manually inspecting it with Appium Desktop.
  - We will observe and filter out the logs generated after the device is connected through Charles Proxy same as before with a real device.
  - I will also demonstrate how Appium Desktop can help you automatically generate code that can be used for automation using its recording feature.

- As I am simulating a testing session, I will use the following hybrid tool to capture information about my session (screenshots, notes, logs from Charles Proxy etc.)

  - **./start-specific-local-machine-example-workflows-for-workshop.sh Hybrid-Tool-Desktop-Charles-Proxy-Workshop-Example**

Part Two - Tool strategy examples for **enhancing existing tests** and **taking your test process further**

Now lets get hands on...

- Please proceed to https://github.com/jg8481/Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019
- Please feel free to try the examples on your machines (if you have not yet done so already).
  - **Mac and Linux users**: you should set up a .env file on your machines
  - **Windows users:** you can hard-code those values in the docker-compose file for the Docker examples. For the hybrid desktop test you can hardcode the values using the robot runner variable command line option
    - Here is an example, -**-variable TEST_ENVIRONMENT:QA**
- I am going to run and explain all of the following for Part Two.
  - **./start-specific-docker-example-workflows-for-workshop.sh Part-Two-Postman-Newman-Workshop-Examples**
  - **./start-specific-docker-example-workflows-for-workshop.sh Part-Two-Python-Library-Workshop-Examples**
  - **./start-specific-docker-example-workflows-for-workshop.sh Part-Two-Requests-Library-Workshop-Examples**
  - **./start-specific-docker-example-workflows-for-workshop.sh Part-Two-cURL-Workshop-Examples**
  - **./start-specific-docker-example-workflows-for-workshop.sh Remote-Test-Process-Triggered-By-A-Webhook-Docker-Example**

  - **./start-specific-docker-example-workflows-for-workshop.sh Long-Graphwalker-Run-Workshop-Example**
  - **./start-specific-docker-example-workflows-for-workshop.sh Previous-Long-Graphwalker-Run-Workshop-Example**
  - **./start-specific-docker-example-workflows-for-workshop.sh Short-Graphwalker-Run-Workshop-Example**
  - **./start-specific-docker-example-workflows-for-workshop.sh Previous-Short-Graphwalker-Run-Workshop-Example**

  - **./start-specific-local-machine-example-workflows-for-workshop.sh Hybrid-Tool-Desktop-Workshop-Example**
  - **./start-specific-local-machine-example-workflows-for-workshop.sh Hybrid-Tool-Desktop-Charles-Proxy-Workshop-Example**
  - **Using Appium Desktop to automatically generate scripts that can be used for automation, while using the Hybrid-Tool-Desktop-Workshop-Example on an iOS simulator**

Part Three - Combined results <u>**visualizing your testing story**</u>

Robot Framework has excellent builtin test logging and reporting mechanisms. The community has also developed other types of excellent reports as well...

- Specifically for this workshop, Shiva Prasad Adirala's excellent RobotFramework-Metrics project which I use to demonstrate a way to help show Session-based Test Management metrics from the hybrid desktop tests in Part Two.
- https://github.com/adiralashiva8/robotframework-metrics

The Part 3 examples will show some useful Rebot commands combined with git commands to create a way to automatically deploy Robot Framework .html files to Heroku. These are the dashboards we will focus on sending test logs and dashboard files to...

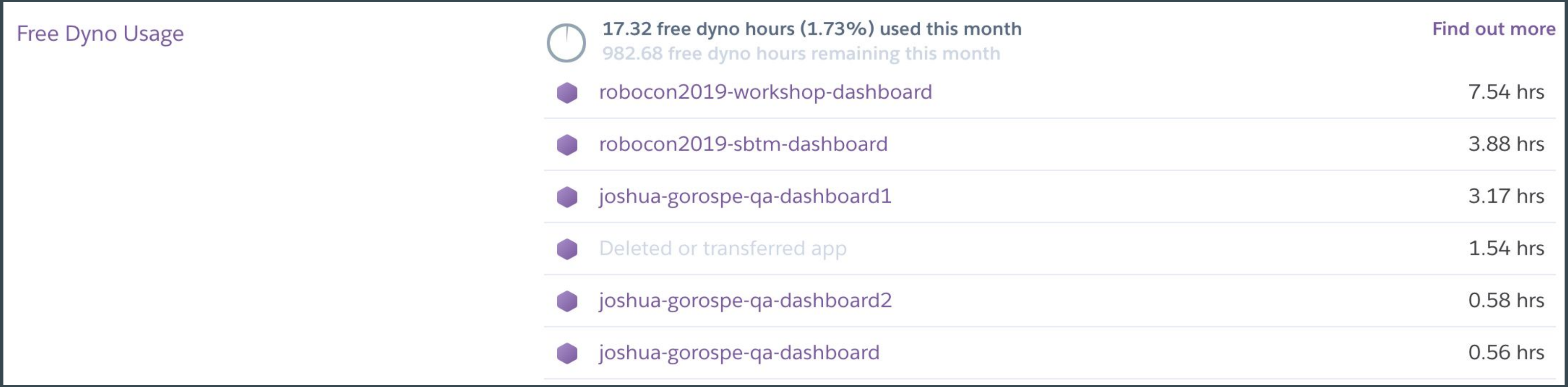
- https://robocon2019-workshop-dashboard.herokuapp.com/

- https://robocon2019-sbtm-dashboard.herokuapp.com/

I will also give quick explanation on how to set up Heroku.

Part Three - Combined results **visualizing your testing story**

Creating a Heroku account is free. Heroku has very flexible service plans and a very generous "Free Tier" that gives you a lot of free Dyno hours for every app you deploy. I have barely put a dent in the 1000 hours of free usage for this month.



Free Dyno Usage

17.32 free dyno hours (1.73%) used this month
982.68 free dyno hours remaining this month

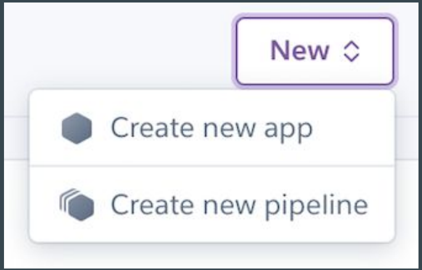
Find out more

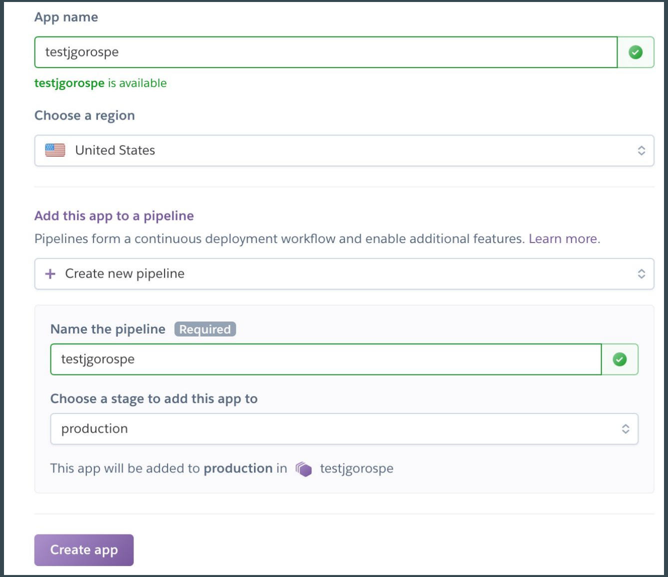
robocon2019-workshop-dashboard	7.54 hrs
robocon2019-sbtm-dashboard	3.88 hrs
joshua-gorospe-qa-dashboard1	3.17 hrs
Deleted or transferred app	1.54 hrs
joshua-gorospe-qa-dashboard2	0.58 hrs
joshua-gorospe-qa-dashboard	0.56 hrs

Part Three - Combined results **visualizing your testing story**

If you decide to set up a Heroku account, setting up a new app that turns your Robot Framework .html outputs into an online dashboard is simple.

- Set up an empty GitHub repo from your account.
- Go to, https://dashboard.heroku.com/apps
- Select “Create new app”.
- Name your app, and “Create a new pipeline” for it. Select “production”.
- Click “Create app”.
- From https://dashboard.heroku.com/apps click on the app you created.




App name

testjgorospe

testjgorospe is available

Choose a region

United States

Add this app to a pipeline

Pipelines form a continuous deployment workflow and enable additional features. Learn more.

Create new pipeline

Name the pipeline Required

testjgorospe

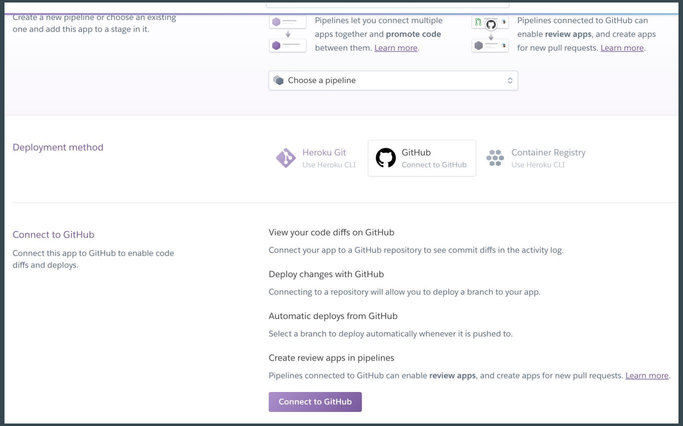
Choose a stage to add this app to

production

This app will be added to production in testjgorospe

Part Three - Combined results **visualizing your testing story**

- Click on the “Deploy” tab.
- An “Authorize Heroku Dashboard” window pop-up.
- Click the “Authorize heroku” button.
- Select the repo you set up.


Create a new pipeline or choose an existing one and add this app to a stage in it.

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Deployment method

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

View your code diffs on GitHub

Connect your app to a GitHub repository to see commit diffs in the activity log.

Deploy changes with GitHub

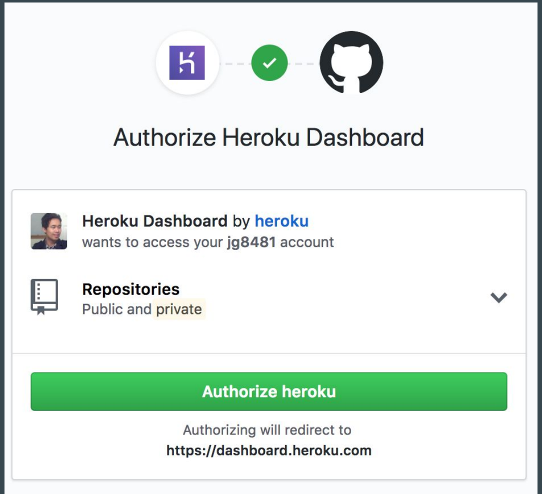
Connecting to a repository will allow you to deploy a branch to your app.

Automatic deploys from GitHub

Select a branch to deploy automatically whenever it is pushed to.

Create review apps in pipelines

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.


Authorize Heroku Dashboard

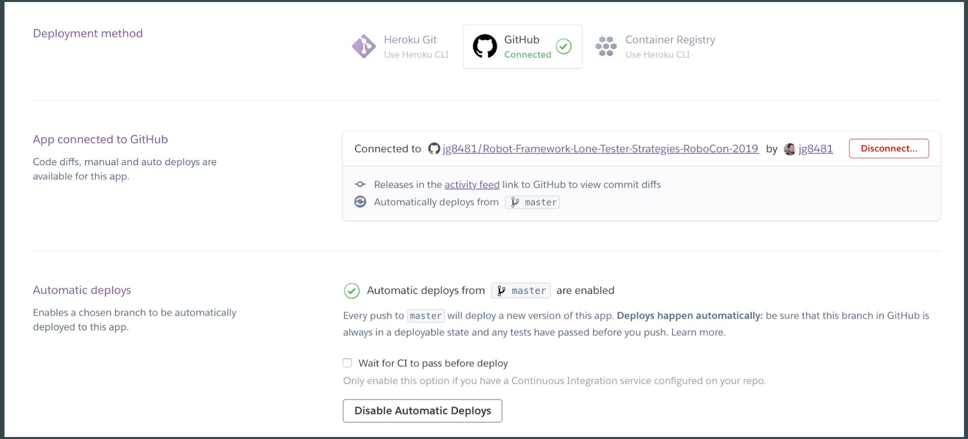
Heroku Dashboard by heroku
wants to access your **jg8481** account

Repositories
Public and private

Authorizing will redirect to
https://dashboard.heroku.com

Part Three - Combined results visualizing your testing story

- The page should now say “GitHub Connected”.
- You should “Enable Automatic Deploys”.
- In the Tool-Strategies-Lone-Testers-Test-Leadership-Congress-2019 repo you need to copy the “composer.json” and “index.php” files and put them into your repo. Also copy your version of the “combined-results.html” to your repo.
- Push the files to your repo. If you selected “Enable Automatic Deploys”, click “Open App” and it should display the combined-results.html in a new browser window. I will use a Graphwalker html log file and demonstrate this.
- You can also click on “Manual deploy” too as well.


Deployment method

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019 by jg8481

Releases in the activity feed link to GitHub to view commit diffs

Automatically deploys from master

Automatic deploys

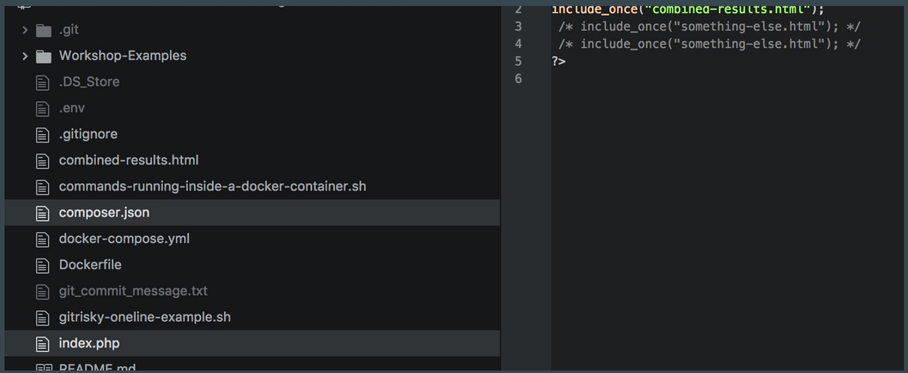
Enables a chosen branch to be automatically deployed to this app.

Automatic deploys from master are enabled

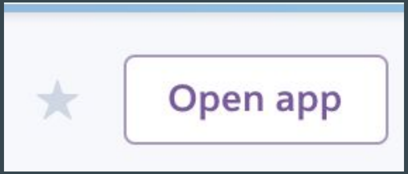
Every push to master will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. Learn more.

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.


```
.git
Workshop-Examples
.DS_Store
.env
.gitignore
combined-results.html
commands-running-inside-a-docker-container.sh
composer.json
docker-compose.yml
Dockerfile
git_commit_message.txt
gitrisky-oneline-example.sh
index.php
README.md
```

```
include_once("combined-results.html");
 /* include_once("something-else.html"); */
 /* include_once("something-else.html"); */
?>
```



Part Three - Combined results **visualizing your testing story**

I will now run the following examples and explain what is happening...

- **./start-specific-result-gathering-example-workflows-for-workshop.sh Combined-Results-Dashboard-Workshop-Example**
- **./start-specific-result-gathering-example-workflows-for-workshop.sh Metrics-Dashboard-Workshop-Example**

Eventually (a few seconds) the following dashboards will be updated...

- https://robocon2019-workshop-dashboard.herokuapp.com/
- https://robocon2019-sbtm-dashboard.herokuapp.com/

Questions?

The End...

:-) ...for now.

References

- “Robot Framework Documentation”, https://robotframework.org/#documentation
- “Lessons Learned in Software Testing: A Context-Driven Approach” by Cem Kaner, James Bach, Bret Pettichord
- “Omega Tester: Testing with a Team of One”, by James Bach, http://www.satisfice.com/articles/omega_tester.pdf
- “Rapid Software Testing”, by James Bach and James Bach, slide 97, http://www.satisfice.com/rst.pdf
- “Session-Based Test Management”, by Jonathan Bach and James Bach, http://www.satisfice.com/sbtm/
- “A Context-Driven Approach to Automation in Testing”, by James Bach and Michael Bolton, http://www.satisfice.com/articles/cdt-automation.pdf