# Robot Framework strategies for the lone tester

## RoboCon 2019 - Workshop

Presented by Joshua Gorospe

# Before we begin...

Just want to again, thank the following people and groups.

- Pekka Klarck
- Ed Manlove
- Antti Karjalainen
- James Bach
- Jonathon Bach
- Michael Bolton
- Cem Kaner
- Kristian Karl
- Mikko Korpela
- Bryan Oakley
- Shiva Prasad Adirala
- The Docker development team
- The Graphwalker development team
- The entire Robot Framework community and its contributors.

# Introduction

Some information...

- I work at Intersection, https://www.intersection.com/
- I am a senior test engineer working in the Intersection Connected Communities team.
- I enjoy experimenting with test tools.
- I like considering myself as simply a tester, I am proud of it.
- Testing, designing test strategies, and finding bugs makes me feel like a detective scientist.
    - Funny fact: After I wrote this slide, I Googled “detective scientist” and found this... https://www.nasa.gov/audience/forstudents/nasaandyou/home/detective_en-index.html
- Funny fact: I also enjoy exciting Dungeons & Dragons sessions with my buddies from college (we are just casual players).

# Introduction

**Intersection?**

Delivers products such as...

- Link Wifi
  - New York
  - London... and more
- MTA On-The-Go Kiosks

In the past, we provided client services for...

- Large telecom companies
- Hudson Yards
- Large retail companies
- Large real estate companies

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**..."

***^ For each of these suggestions, especially the underlined parts, Robot Framework can be used to assist us.***

**"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 workshop 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/

# Workshop Examples

Are organized into the following parts...

- Part One - Robot Framework strategies for **rapidly testing** and **adapting to frequently changing projects**

- Part Two - Robot Framework strategies 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 to "**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**

# Workshop 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 - Robot Framework strategies 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
- Short Break: For answering questions etc.

- Part Two - Robot Framework strategies 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-local-machine-example-workflows-for-workshop.sh Hybrid-Tool-Desktop-Workshop-Example
  - ./start-specific-local-machine-example-workflows-for-workshop.sh Hybrid-Tool-Appium-Workshop-Example
- Short Break: For answering questions etc.

- 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

- (the fun stuff) Break out into teams and lets experiment...

# Part One - Robot Framework strategies 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/ig8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-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/

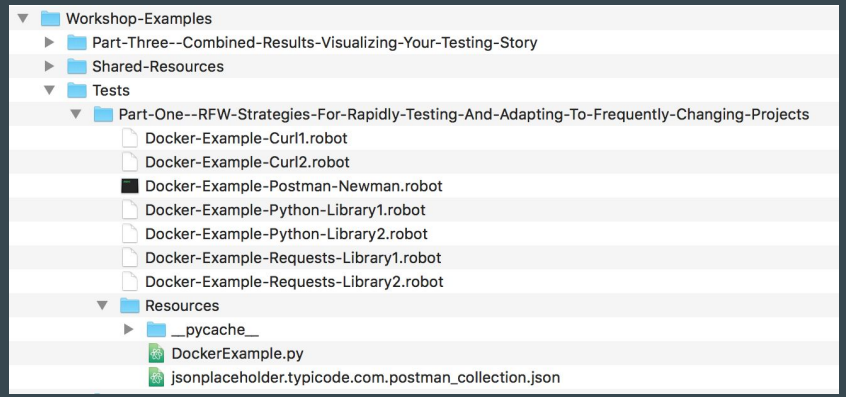
  - Here are examples of Robot Framework triggering tests in serial and parallel using cURL while capturing the response.
    - https://github.com/ig8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Curl1.robot

    - https://github.com/ig8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Curl2.robot

# Part One - Robot Framework strategies 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 with Robot Framework. The keyword driven syntax of Robot Framework may cause certain developers to get confused or not understand it. Luckily the option to utilize user-created libraries can help reduce that confusion. Also having an easy to understand test folder structure can help someone quickly learn how Robot Framework works 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 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/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Python-Library1.robot
  - https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-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/



# Part One - Robot Framework strategies 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/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Requests-Library1.robot
  - https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-One/Docker-Example-Requests-Library2.robot

# Part One - Robot Framework strategies for **rapidly testing** and **adapting to frequently changing projects**

Now lets get hands on...

- Please proceed to https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-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**

Questions about any of the Part One examples?

# Part Two - Robot Framework strategies 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/Robot-Framework-Lone-Tester-Strategies-RoboCon-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/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Python-Library1-Enhanced-Version.robot
  - https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-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/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Curl1-Enhanced-Version.robot
  - https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-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/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Requests-Library1-Enhanced-Version.robot
  - https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Docker-Example-Requests-Library2-Enhanced-Version.robot

# Part Two - Robot Framework strategies 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/Robot-Framework-Lone-Tester-Strategies-RoboCon-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 - Robot Framework strategies for **enhancing existing tests** and **taking your test process further**

The third group of examples in Part Two use 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 are the hybrid tool examples.

- https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-2019/blob/master/Workshop-Examples/Tests/Workshop-Part-Two/Hyrid-Session-Based-Test-Example-For-Android.robot
- https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-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.
  - https://github.com/hinnefe2/gitrisky
  - **cd /path/to/your-repo-name && gitrisky train && time 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;**

# Part Two - Robot Framework strategies for **enhancing existing tests** and **taking your test process further**

Now lets get hands on...

- Please proceed to https://github.com/jg8481/Robot-Framework-Lone-Tester-Strategies-RoboCon-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 Part Two examples as they are running.
  - **./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 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-Appium-Workshop-Example**

Questions about any of the Part Two examples?

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

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 a quick explanation on how to set up Heroku.

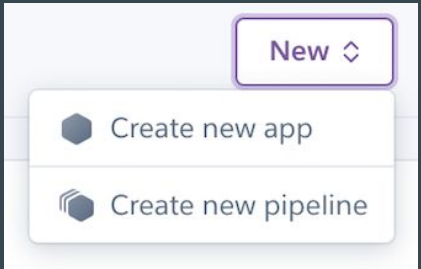
# Part Three - Combined results **visualizing your testing story**

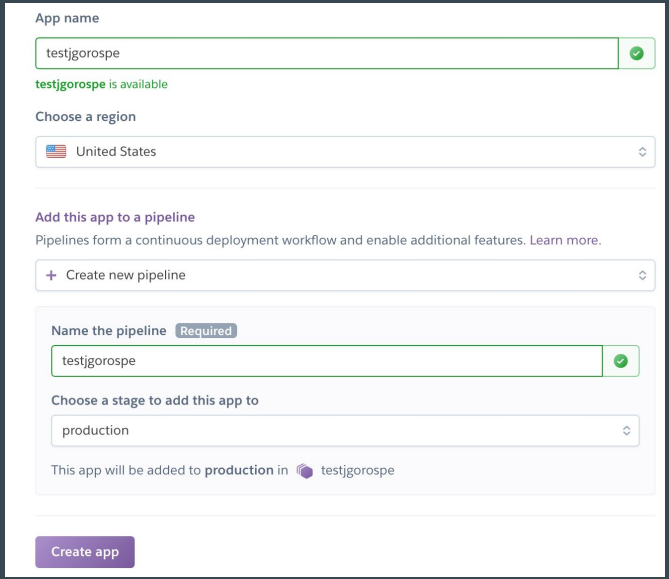
Creating a Heroku account is free. Heroku has very flexible service plans and a very generous “Free Tier” that gives you a lot of free Dyno hours for every app you deploy. I have barely put a dent in the 1000 hours of free usage for this month.

# 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.

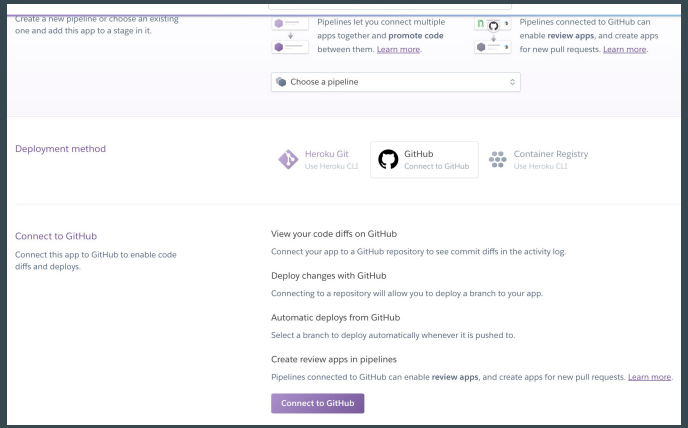
- Set up an empty GitHub repo from your account.
- Go to, https://dashboard.heroku.com/apps
- Select “Create new app”.
- Name your app, and “Create a new pipeline” for it. Select “production”.
- Click “Create app”.
- From https://dashboard.heroku.com/apps click on the app you created.

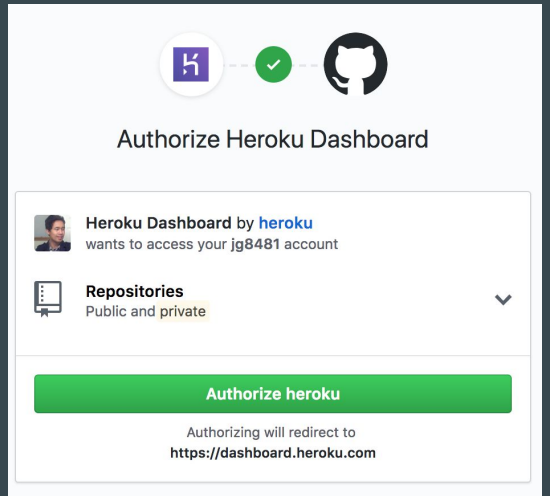


# Part Three - Combined results visualizing your testing story

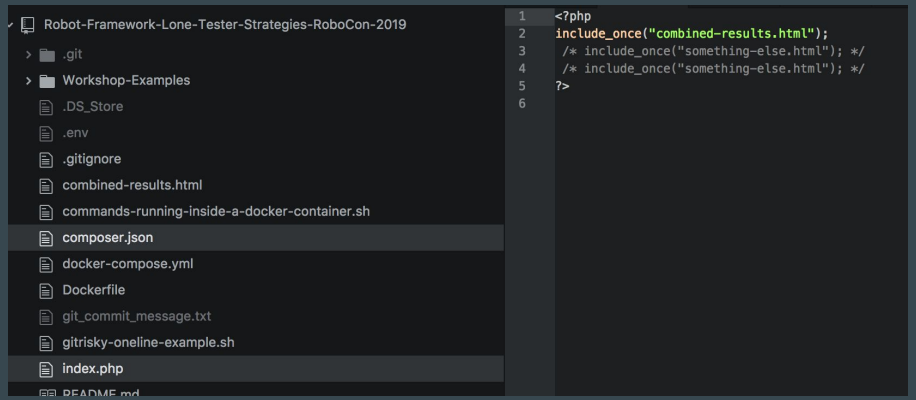
- Click on the “Deploy” tab.
- An “Authorize Heroku Dashboard” window pop-up.
- Click the “Authorize heroku” button.
- Select the repo you set up.

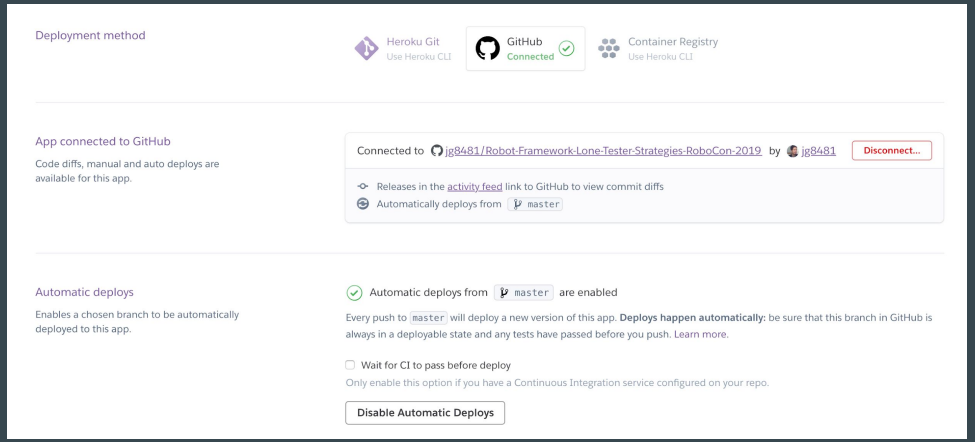


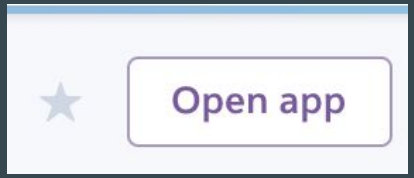



# Part Three - Combined results <u>visualizing your testing story</u>

- The page should now say “GitHub Connected”.
- You should “Enable Automatic Deploys”.
- In the Robot-Framework-Lone-Tester-Strategies-RoboCon-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.



# 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 about any of the Part Three examples?

# Please break out into teams...

- Let's experiment.

- Let's experiment with different workflows.

    - For example, run the hybrid SBTM desktop test example (try the Google Bug Hot Spot approach + Gitrisky + any of your personal favorite approaches during testing) while running any of the Docker examples.
        - My personal favorite is the Robot Framework + Graphwalker + Docker examples...
    - Experiment with the results post-processing with Rebot.

- Let's try different things with the examples.

Some ideas to probably get your creativity flowing...

- Try this out, http://eradman.com/entrproject/
- Or this, https://github.com/ianmiell/cheapci
- Or maybe this, https://github.com/alexylem/jarvis
- Or maybe this one, https://github.com/kalliope-project/kalliope

I may have some personal assistant tool ideas to share with you.

# The End...

:-)  ...for now.

# References


- “Robot Framework Documentation”, https://robotframework.org/#documentation
- “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