

DataRobot



Swift Mode

Dragging pytest-testmon up
to DataRobot scale

Carson Gee

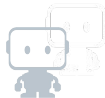
DataRobot

Confidential. ©2020 DataRobot, Inc. – All rights reserved



Agenda

1. Overview
2. Why use testmon?
3. Scaling testmon to Swift Mode
4. Results and next steps



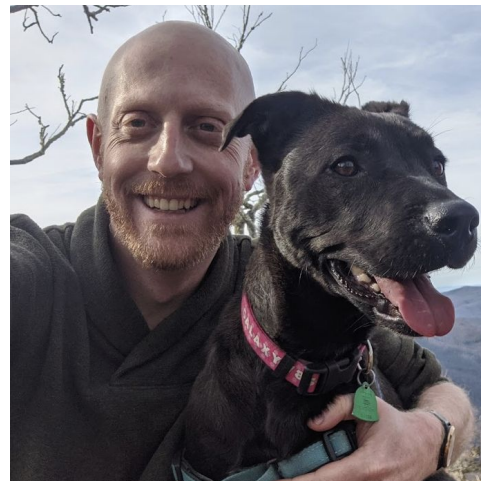
Overview

1. Who am I?
2. What is pytest-testmon?

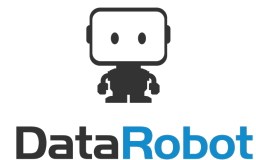
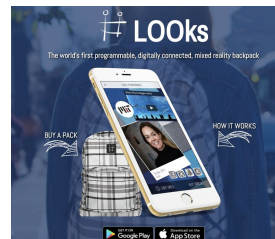
Who am I? Carson Gee



- **Really Full Stack (i.e. Easily distracted)**
- **Worked on Cool Stuff**
 - Helped NASA map out what is under the ice in Antarctica, Greenland, and Mars
 - Served on Architects Council for Open edX
 - Brought the Copenhagen Wheel to market (e-bike conversion kit)
 - Built an AR enabled social network out of backpacks, hats, and even duct tape
 - Helped DataRobot run ~70% less unit/integration tests
- **Loves** 🦩, 🚲, 🐶, and 🍺



OPENedX[™]
Over 40 million
learners reached

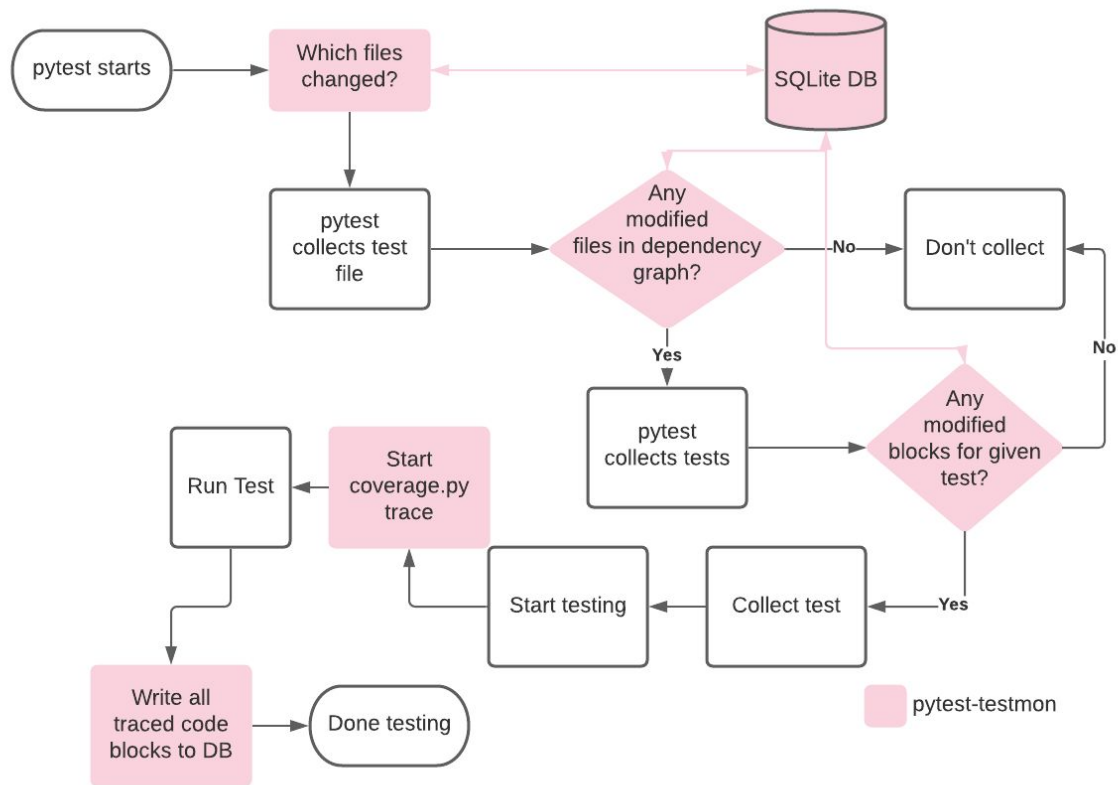


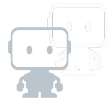
What is pytest-testmon?



- Open source plugin for the Python test runner [pytest](#)
- [Testmon](#) selects tests affected by changed files and methods
- Uses [coverage.py](#) to record the code a test exercises
- Uses recordings from previous runs to determine what changed and what tests should run

Pytest with testmon flow





Why testmon?

1. Wait, how many tests?!
2. Why the usual isn't enough

Wait, how many tests?!

No one wants to wait four days to test a twenty line change

Why four days you ask? Oh hi there monorepo!

- 12,604,434 lines of code (~2,000,000 Python)
- >100,000 commits, ~70,000 PRs
- 150,000+ tests run on main branch merge
- Run same tests in multiple configurations and environments (Py2/3, Kubernetes/Hadoop, etc.)



Why the usual isn't enough



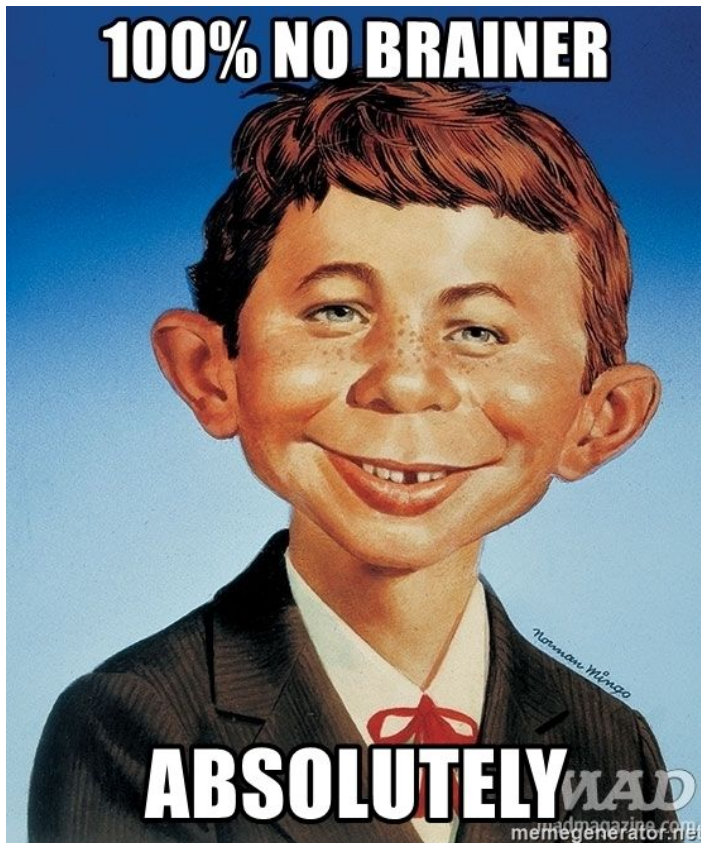
- **Parallelization**
 - Only so many cores, shared services
 - More servers, more problems
 - Setup time starts to dominate testing time
 - \$\$\$\$\$
- **Run most tests nightly instead of per PR**
 - Too much change in a given day, git bisect can take >6 hours for one failed test and doesn't work with flaky tests
 - Staging/CI system broken more frequently
- **Reduce tests run per change with highly cohesive tests and code**
 - Challenging to enforce clean dependency graphs
 - Less cross dependency testing = more broken things

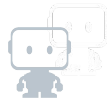
Let's Use testmon!



- ✓ Reduces number of tests for a given PR based on changes
- ✓ Less tests = less runtime
- ✓ Selecting the right tests means irrelevant broken tests are not run

Let's Use testmon!





Scaling testmon

1. Why it doesn't work
2. How we scaled it
3. ...How we scaled it
4. ...Really, how we scaled it


Why it doesn't work



- **Designed to speed local development, not CI level**
- **Requires you to run all tests to populate DB**
- **Database is SQLite and needs to get distributed to build nodes**
- **Once distributed to build node, SQLite is single writer**
- **Non-code file changes break tests**
- **Not ready for millions of lines of code**

How we scaled it: Making Swift Mode

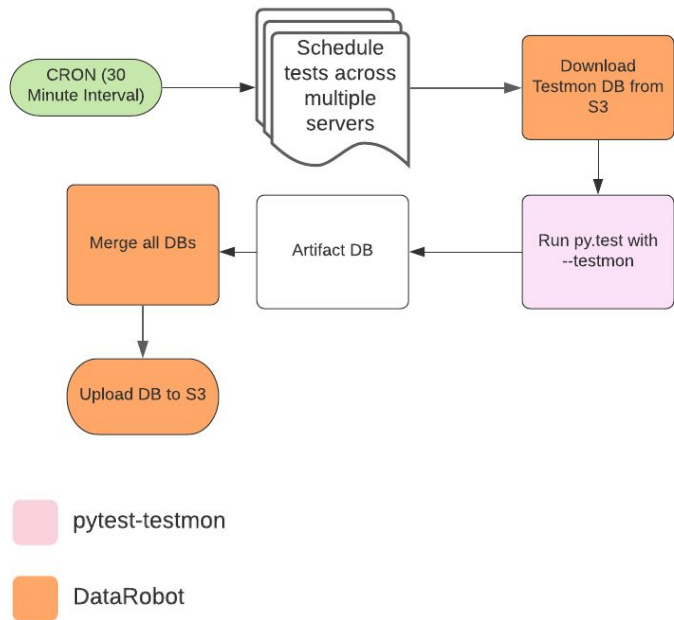


- **Created Script for managing DBs**
 - S3 Storage of DB
 - Merging of executor build DBs
- **Created dedicated Jenkins jobs for updating DB**
- **Made Pytest plugin to export selected tests**
 - Enable xdist and Cythonization
- **Created  Smart Test Entropy Rule Engine Override **
 - Pytest-plugin that lets you override testmon's selections and select them anyway
 - Plugin based architecture, one plugin: file-glob
- **Enable configuration in PR Templates to enable/disable testmon for a PR**

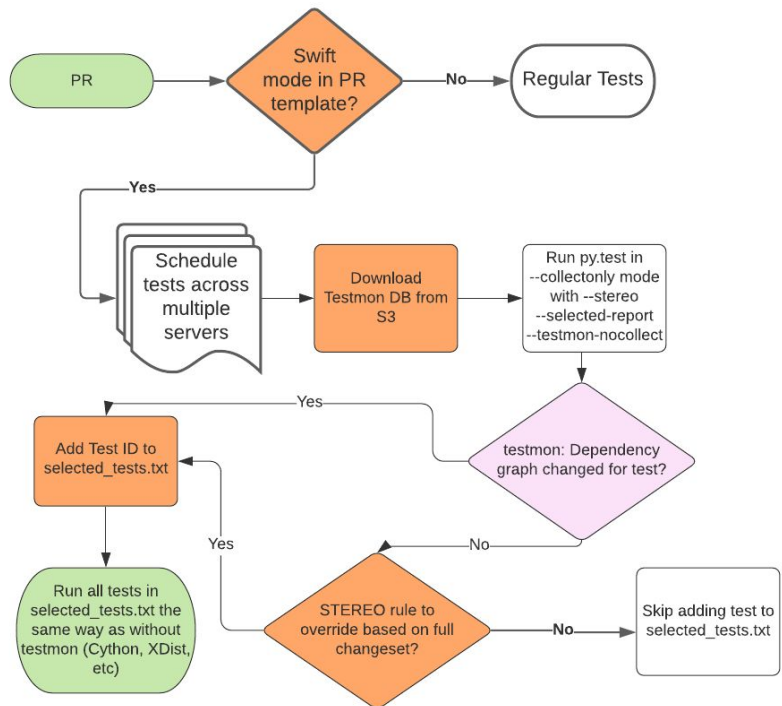
How we scaled it - Swift Mode v0



Update DB Process



Run Tests



How we scaled it - Swift Mode v0



Update DB Process



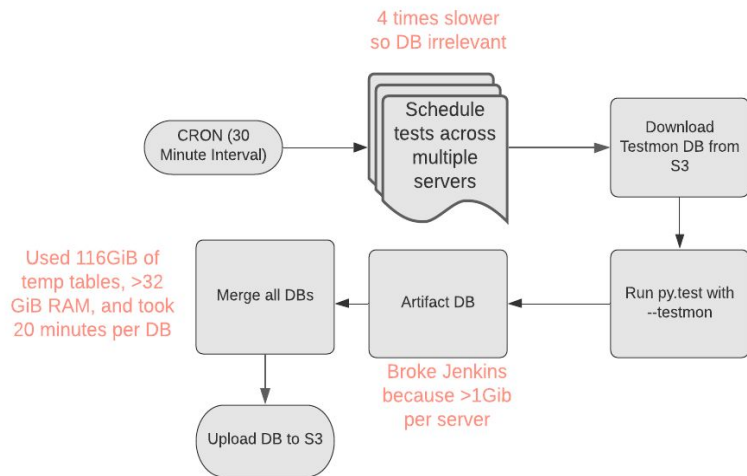
Run Tests



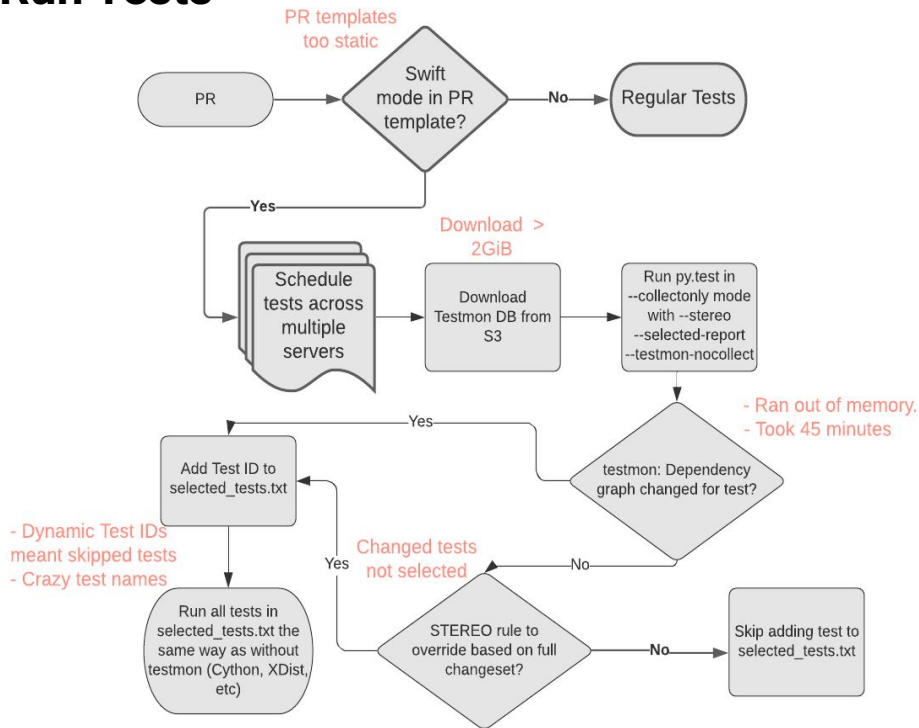
...How we scaled it



Update DB Process



Run Tests



...Really, how we scaled it - Swift Mode v1



Forked: <https://github.com/datarobot/pytest-testmon/>

- **Significant DB schema and pragma changes, i.e. page through the 10 million+ dependency graph edges instead of all in one**
- **Code block fingerprinting performance improvements**
 - Turns out finding comment and blanks lines 35% faster using strip over regex on 2 million lines really matters
- **Stopped sorting tests by runtime (took >5 minutes to sort 34,000 tests)**
- **[ALL the memory optimizations](#)**

...Really, how we scaled it - Swift Mode v1

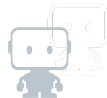


Updates:

- Scaled update server count
- SQL optimizations for merging executor DBs
- XZ compression of DBs for transit speedup (Thanks DevInfra!)

PRs/DTS:

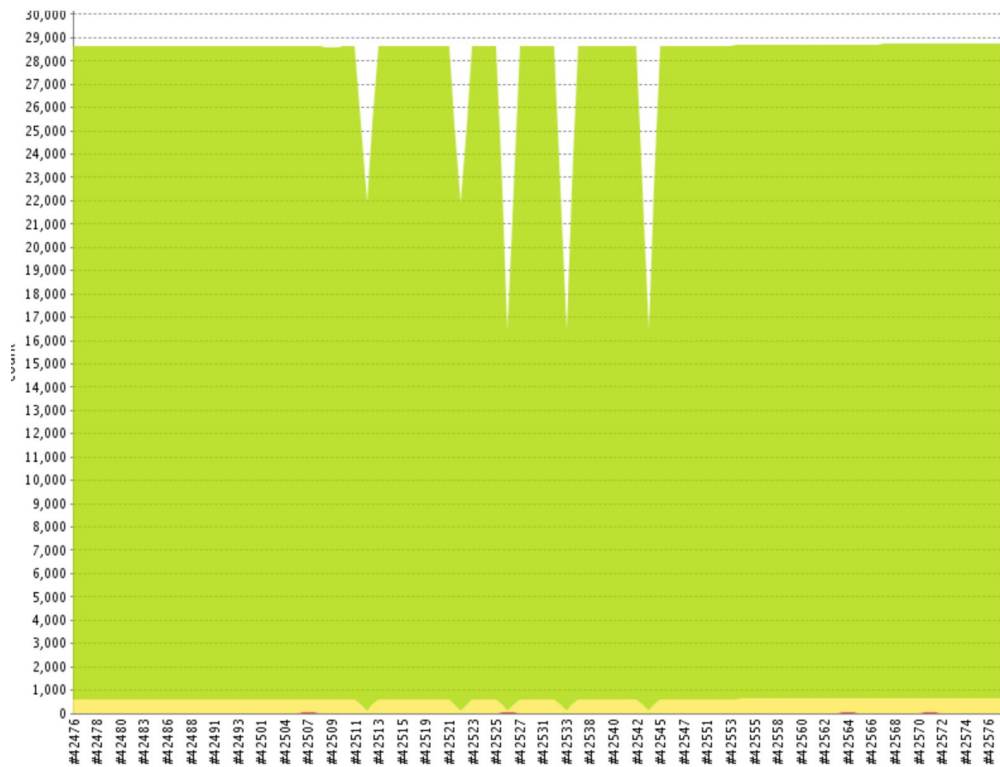
- Created EngProd Service for fast degradation
- Randomized test ID detection, URL encoded test names
- Added changed-tests plugin to pytest-STEREO
- AutoFCS and EngProd App detection of Swift Mode skipped tests (Thanks EngProd Squad!)



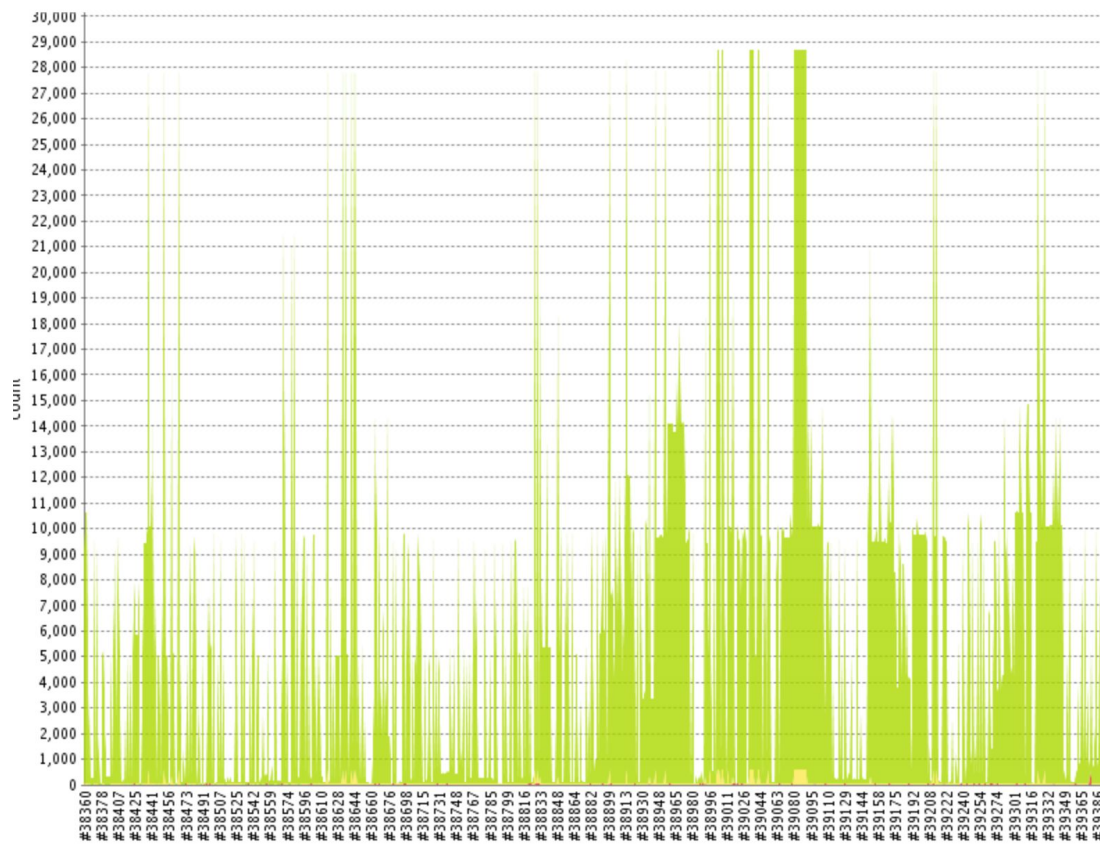
Results and Next Steps

1. Was it worth it?
2. Next steps...?

Was it Worth it?



Number of Tests Run on PRs - Before



Number of Tests Run on PRs - After

Was it worth it?



Yes (probably)!

- **We run about 70% less tests**
- **Runtime down about 30% on average**
- **Saving ~\$600 USD/day**

...

- **Coupled tests exposed!**
- **Master breaks more often!**

Next Steps...?



Speed up PRs:

- Utilize dependency graph in fan out test scheduling
- Faster update cycles to reduce drift from master
- Experiment with using a graph database

Break CI Pipelines Less less:

- Automate correcting dependency graph selection failures
- Resolve coverage.py/Python caching issues

Questions?

