Integrating

Yegor Bugayenko

Lecture #6 out of 8
80 minutes

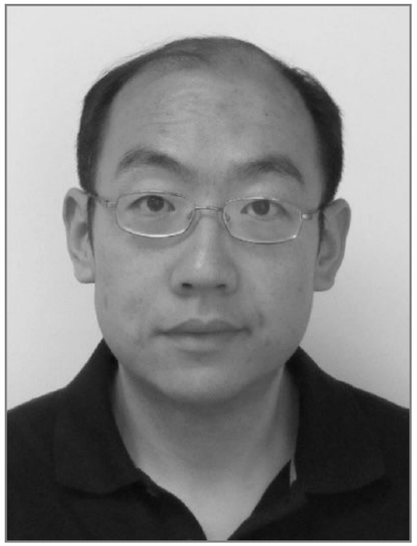
The slidedeck was presented by the author in this YouTube Video

1. Setup continuous integration in order to prove that your product works

Bogdan Vasilescu

"Teams using CI are significantly more effective at merging pull requests submitted by core members. Availability of CI is also associated with external contributors having fewer pull requests rejected."

— Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*, pages 805–816, 2015. doi:10.1145/2786805.2786850



Pei Liu

"We start by collecting a set of 84,475 open-source Android apps from the most popular three online code hosting sites, namely Github, GitLab, and Bitbucket. We then look into those apps and find that only around 10% of apps have leveraged CI/CD services, i.e., the majority of open-source Android apps are developed without accessing CI/CD services."

— Pei Liu, Xiaoyu Sun, Yanjie Zhao, Yonghui Liu, John Grundy, and Li Li. A First Look at CI/CD Adoptions in Open-Source Android Apps. In *Proceedings of the 37th International Conference on Automated Software Engineering*, pages 1–6, 2022. doi:10.1145/3551349.3561341

Mehdi Golzadeh

"Together with Travis, GHA covers more than 80% of all usages. Moreover, in only 18 months GHA has overtaken all other CIs in popularity."

— Mehdi Golzadeh, Alexandre Decan, and Tom Mens. On the Rise and Fall of CI Services in GitHub. In *Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 662–672. IEEE, 2022. doi:10.1109/SANER53432.2022.00084

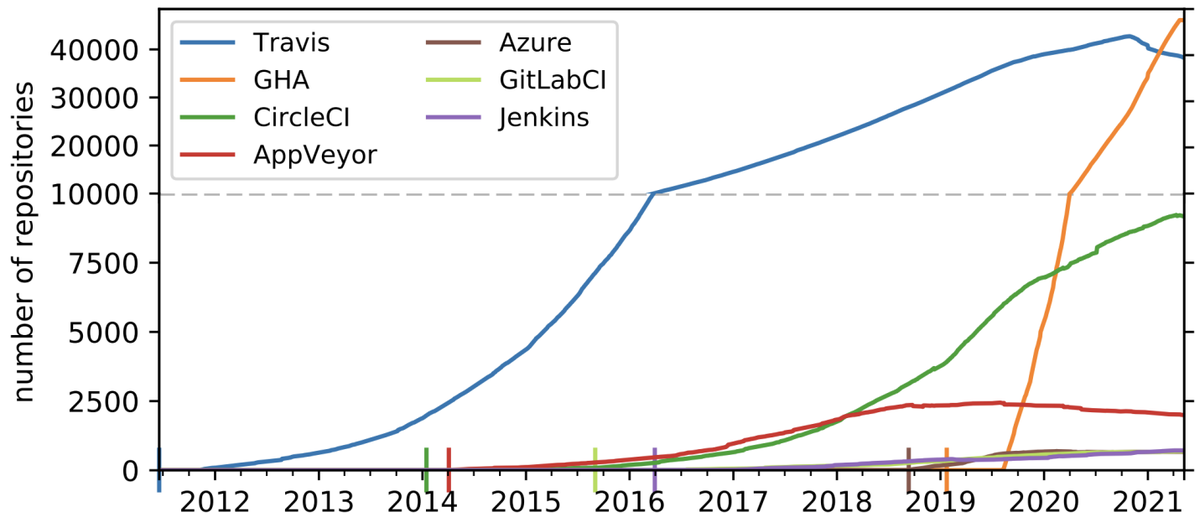


Fig. 2: Number of repositories using a specific CI.

Source: Mehdi Golzadeh, Alexandre Decan, and Tom Mens. On the Rise and Fall of CI Services in GitHub. In *Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 662–672. IEEE, 2022. doi:10.1109/SANER53432.2022.00084

2. Use fixed versions of dependencies

3. Use build matrix, with fixed versions

```yaml
jobs:
  example_matrix:
    strategy:
      matrix:
        os: [ubuntu-22.04, ubuntu-20.04]
        version: [10, 12, 14]
    runs-on: ${{ matrix.os }}
    steps:
      - uses: actions/setup-node@v4
        with:
          node-version: ${{ matrix.version }}
```

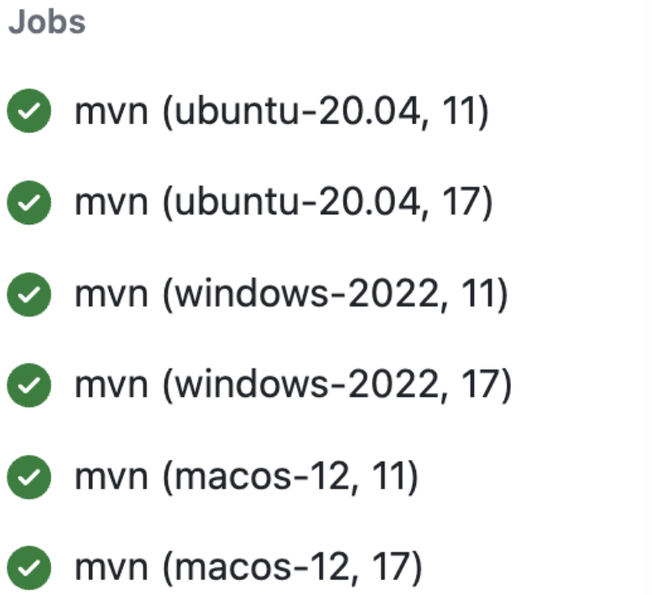
“A matrix strategy lets you use variables in a single job definition to automatically create multiple job runs that are based on the combinations of the variables. For example, you can use a matrix strategy to test your code in multiple versions of a language or on multiple operating systems.”

— GitHub

Moritz Beller

"The use of multiple integration environments leads to 10% more failures being caught at build time."

— Moritz Beller, Georgios Gousios, and Andy Zaidman. Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI With GitHub. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR)*, pages 356–367. IEEE, 2017. doi:10.1109/MSR.2017.62

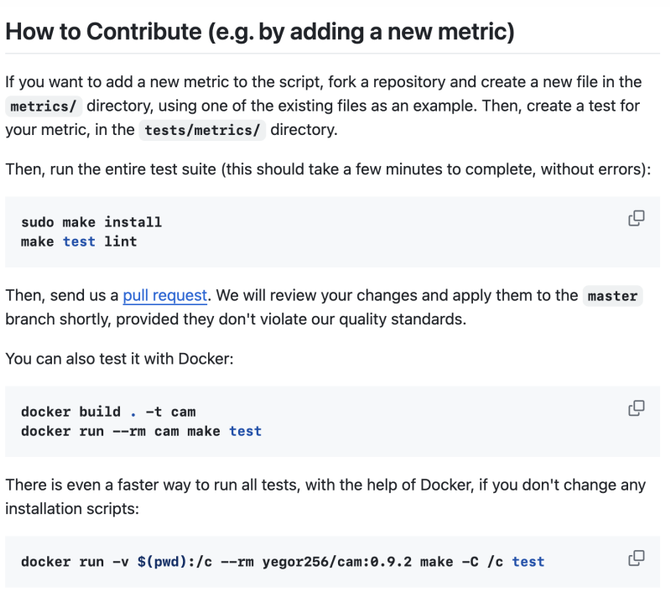


GitHub Actions in yegor256/xembly

"One might argue that it therefore only makes sense to do continuous integration in several environments when their execution leads to different results, capturing errors that would not have been caught with one single environment."

Source: Moritz Beller, Georgios Gousios, and Andy Zaidman. Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI With GitHub. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR)*, pages 356–367. IEEE, 2017. doi:10.1109/MSR.2017.62

4. Provide `Dockerfile`



How to Contribute (e.g. by adding a new metric)

If you want to add a new metric to the script, fork a repository and create a new file in the `metrics/` directory, using one of the existing files as an example. Then, create a test for your metric, in the `tests/metrics/` directory.

Then, run the entire test suite (this should take a few minutes to complete, without errors):

```
sudo make install
make test lint
```

Then, send us a pull request. We will review your changes and apply them to the `master` branch shortly, provided they don't violate our quality standards.

You can also test it with Docker:

```
docker build . -t cam
docker run --rm cam make test
```

There is even a faster way to run all tests, with the help of Docker, if you don't change any installation scripts:

```
docker run -v $(pwd):/c --rm yegor256/cam:0.9.2 make -C /c test
```

https://github.com/yegor256/cam

"One of the largest benefits about Dockerfiles is that they can be completely self contained. Your CI vendor of choice starts to matter less and less because the Dockerfiles themselves are portable and predictable."

Source: Aaron Batilo. The 4 Ways to Run Your Unit Tests in CI With Dockerfiles. https://jttu.net/batilo2022ways, 11 2022. [Online; accessed 03-04-2024]

5. Be aware of flaky tests

Thomas Durieux

"We observe that developers restart at least 1.72% of builds, amounting to 56,522 restarted builds in our Travis CI dataset. We observe that more mature and more complex projects are more likely to include restarted builds. The restarted builds are mostly builds that are initially failing due to a test, network problem, or a Travis CI limitations such as execution timeout."

— Thomas Durieux, Claire Le Goues, Michael Hilton, and Rui Abreu. Empirical Study of Restarted and Flaky Builds on Travis CI. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pages 254–264, 2020. doi:10.1145/3379597.3387460

6. Enable @renovate or @dependabot

7. Discriminate tests as fast and slow [Bugayenko, 2023].

8. Use caching in GitHub Action

Islem Bouzenia

"The majority of the used resources is consumed by testing and building (91.2%), which is triggered by pull requests (50.7%), pushes (30.9%), and regularly scheduled workflows (15.5%). While existing optimizations, such as caching (adopted by 32.9% of paid-tier repositories), demonstrate a positive impact, they overall remain underutilized."

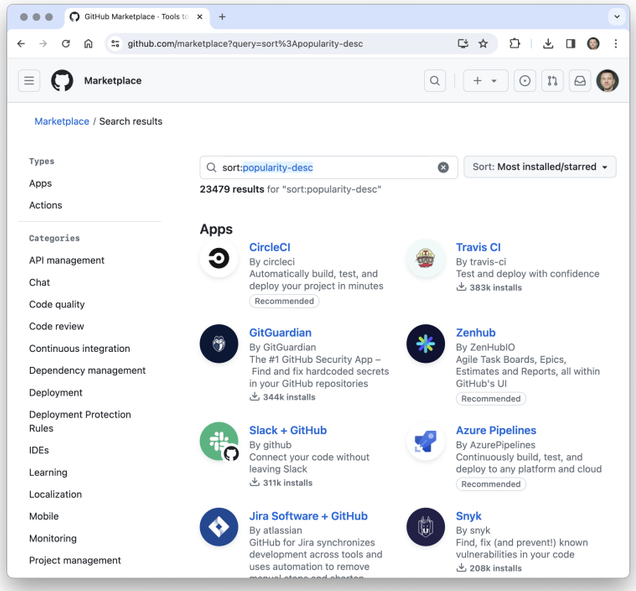
— Islem Bouzenia and Michael Pradel. Resource Usage and Optimization Opportunities in Workflows of GitHub Actions. In *Proceedings of the 46th International Conference on Software Engineering*, pages 1–12, 2024. doi:10.1145/3597503.3623303

9. Implement your own GitHub Actions

Sk Golam Saroar

"We found that developers find the composition of YAML files, which are essential for GitHub Action integration, challenging and error-prone."

— Sk Golam Saroar and Maleknaz Nayebi. Developers' Perception of GitHub Actions: A Survey Analysis. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pages 121–130, 2023. doi:10.1145/3593434.3593475



https://github.com/marketplace

"While there are over 15,000 Actions in existence, an increasing number of Actions are published every day on the Marketplace. This creates the question as to why developers prefer to develop new Actions rather than reusing the existing ones."

Source: Sk Golam Saroar and Maleknaz Nayebi. Developers' Perception of GitHub Actions: A Survey Analysis. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pages 121–130, 2023. doi:10.1145/3593434.3593475

Table 3: Motivations and challenges for creating Actions (P_{44})

Reason for creating Action	% of participants
No existing Action was available.	56.82
Existing actions are limited in functionality.	25
Existing actions are not performant enough.	11.36
Not sure how to find and reuse Actions.	4.55
Existing actions were too complex.	2.27

Source: Sk Golam Saroar and Maleknaz Nayebi. Developers' Perception of GitHub Actions: A Survey Analysis. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pages 121–130, 2023. doi:10.1145/3593434.3593475

Bibliography

Aaron Batilo. The 4 Ways to Run Your Unit Tests in CI With Dockerfiles. `https://jttu.net/batilo2022ways`, 11 2022. [Online; accessed 03-04-2024].

Moritz Beller, Georgios Gousios, and Andy Zaidman. Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI With GitHub. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR)*, pages 356–367. IEEE, 2017. doi:10.1109/MSR.2017.62.

Islem Bouzenia and Michael Pradel. Resource Usage and Optimization Opportunities in Workflows of GitHub Actions. In *Proceedings of the 46th International Conference on Software Engineering*, pages 1–12, 2024. doi:10.1145/3597503.3623303.

Yegor Bugayenko. Fast Tests Help Humans, Deep Tests Help Servers. `https://www.yegor256.com/230822.html`, 8 2023. [Online; accessed 05-03-2024].

Thomas Durieux, Claire Le Goues, Michael Hilton, and Rui Abreu. Empirical Study of Restarted and Flaky Builds on Travis CI. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pages 254–264, 2020. doi:10.1145/3379597.3387460.

Mehdi Golzadeh, Alexandre Decan, and Tom Mens. On the Rise and Fall of CI Services in GitHub. In *Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 662–672. IEEE, 2022. doi:10.1109/SANER53432.2022.00084.

Pei Liu, Xiaoyu Sun, Yanjie Zhao, Yonghui Liu, John Grundy, and Li Li. A First Look at CI/CD Adoptions in Open-Source Android Apps. In *Proceedings of the 37th International Conference on Automated Software Engineering*, pages 1–6, 2022. doi:10.1145/3551349.3561341.

Sk Golam Saroar and Maleknaz Nayebi. Developers' Perception of GitHub Actions: A Survey Analysis. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pages 121–130, 2023. doi:10.1145/3593434.3593475.

Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*, pages 805–816, 2015. doi:10.1145/2786805.2786850.