

DevOps & Software Delivery in a Global Pandemic







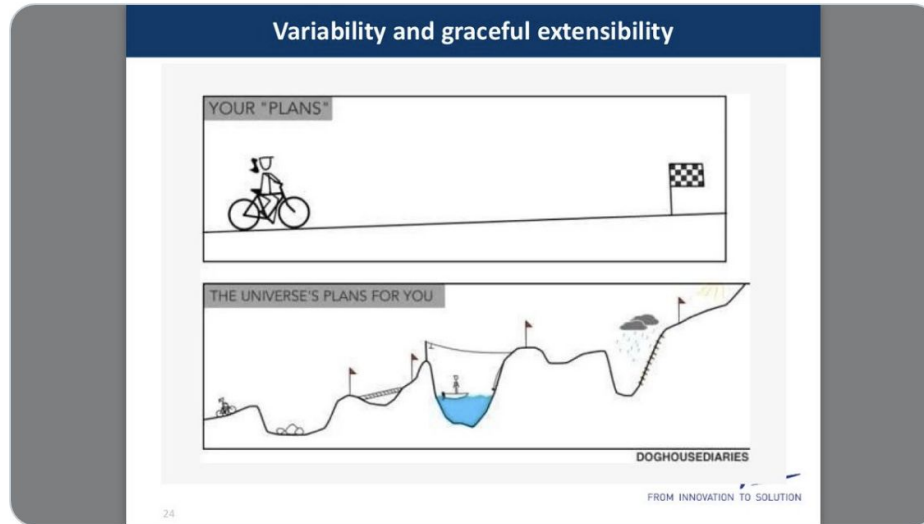
John Allspaw

@allspaw

Normal 0%



Work-as-imagined versus work-as-done



3:00 AM · Apr 28, 2016 · Twitter for iPhone

performance described
VS
performance derived



Jeremy Meiss
Director, DevRel & Community



 @IAmJerdog

2 million

jobs/day

44,000+

orgs

** 40k in 2019*

160,000+

projects

** 150k in 2019*

1,000x

Larger than surveys

Four classic metrics

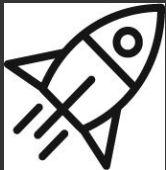
Deployment
frequency

Lead time
to change

Change failure
rate

Recovery
from failure
time

CI/CD Benchmarks for high performance



Throughput

At will



Duration

<10 minutes



Success Rate

> 90%



Mean Time to Recovery

<1 hour

The Data



Photo by: Matthew Henry

Throughput

Percentile	2020 Value	2019 Value
5p	0.03	0.03
50p	0.70	0.80
90p	16.03	13.00
95p	32.125	25.47
Mean	8.22	5.76

***Most teams are not
deploying dozens of times
per day***

Image by Pawan Kolhe from Pixabay



Duration

Percentile	2020 Value	2019 Value
5p	12 sec	10 sec
50p	3.96 min	3.38 min
90p	21.35 min	19.18 min
95p	34.01 min	31.73 min
Mean	24.6 min	26.76 min



Photo by Lukas from Pexels

Success Rate

Percentile	2020 Value	2019 Value
5p	0%	0%
50p	61%	60%
90p	100%	100%
95p	100%	100%
Mean	54%	54%



Photo by Brett Sayles from Pexels

Recovery Time

Percentile	2020 Value	2019 Value
5p	2.06 min	2.83 min
50p	55.11 min	52.5 min
90p	39 hours	47 hours
95p	3.4 days	3.93 days
Mean	14.85 hours	16.61 hours

Recovery Time

Percentile	2020 Value	2019 Value
5p	2.06 min	2.83 min
50p	55.11 min	52.5 min
90p	39 hours	47 hours
95p	3.4 days	3.93 days
Mean	14.85 hours	16.61 hours

Recovery Time

Percentile	2020 Value	2019 Value
5p	2.06 min	2.83 min
50p	55.11 min	52.5 min
90p	39 hours	47 hours
95p	3.4 days	3.93 days
Mean	14.85 hours	16.61 hours

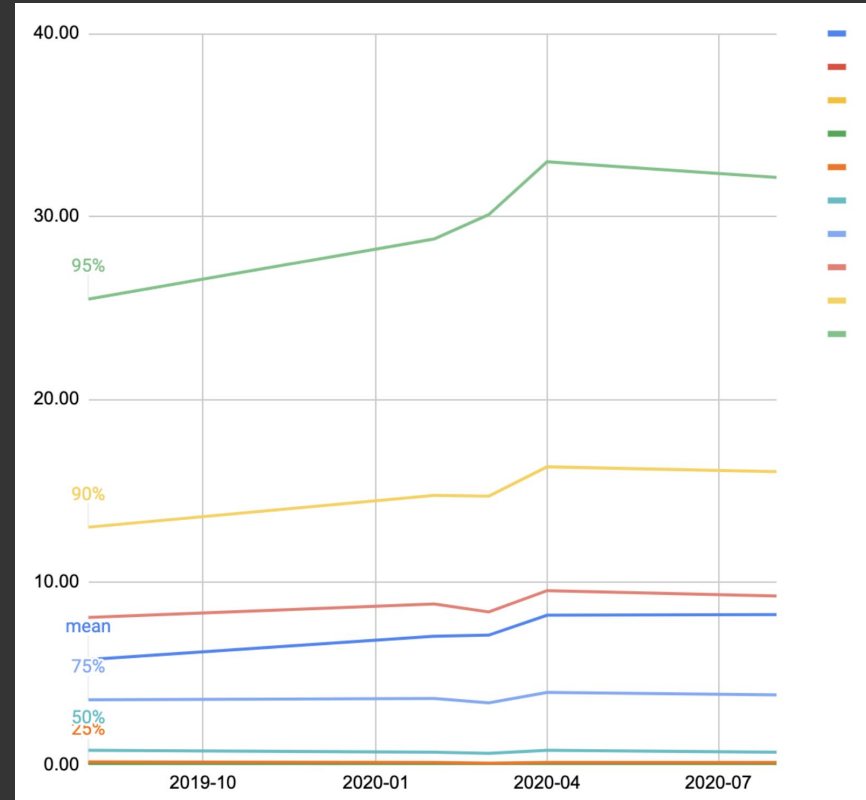
The Insight



2020 has been
a year.

Throughput

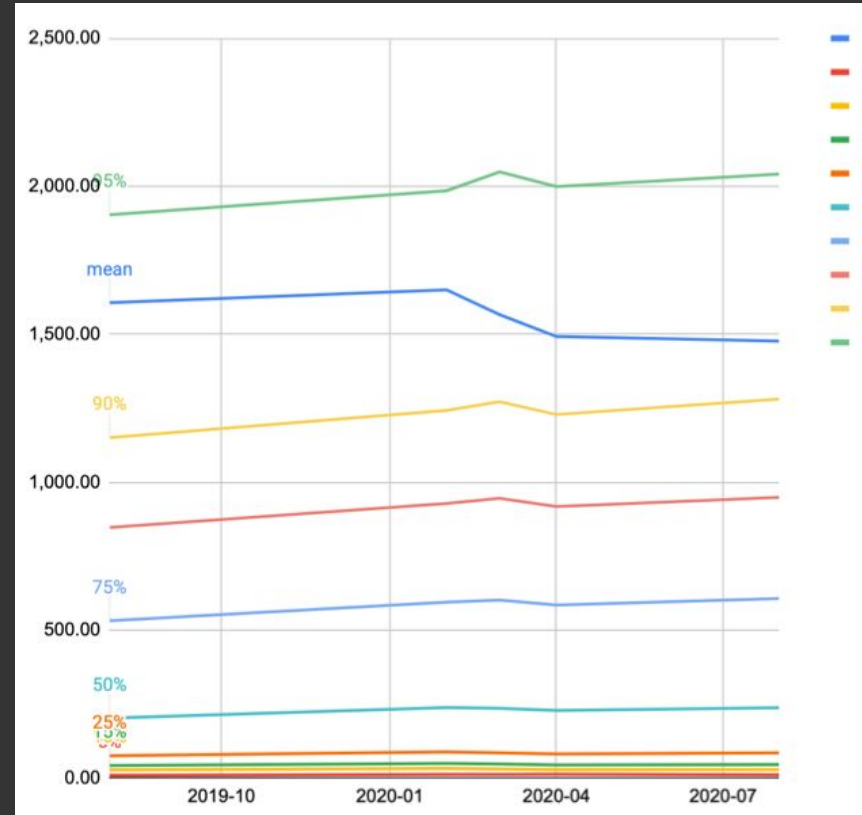
Throughput in a global pandemic



Peak Throughput was in April 2020

Duration

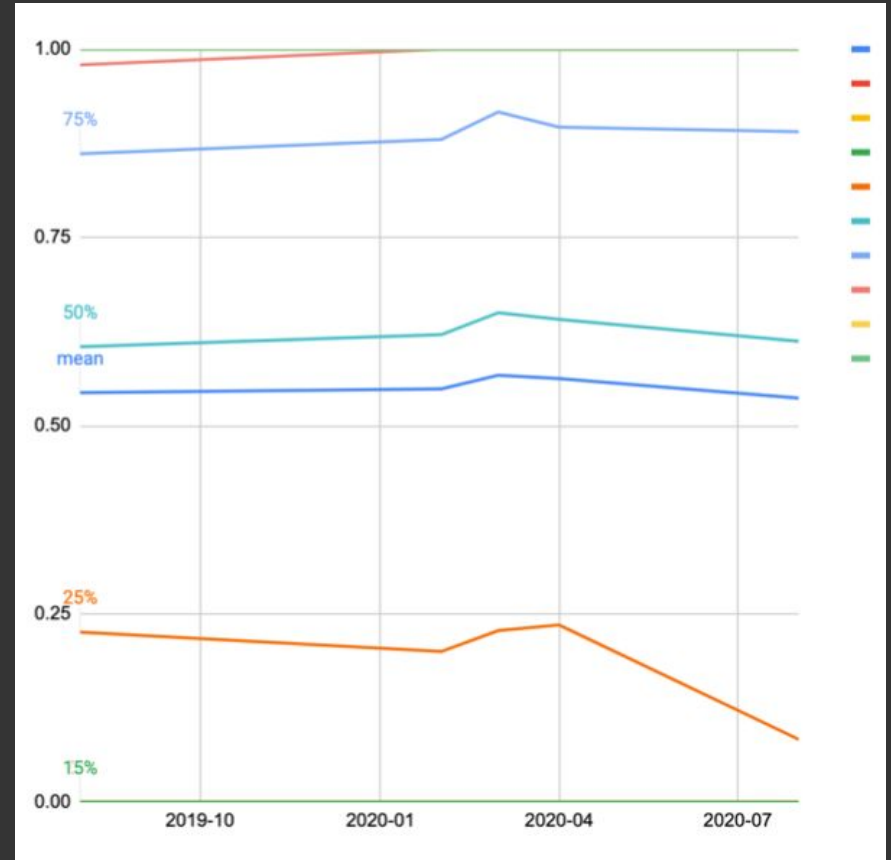
Duration in a global pandemic



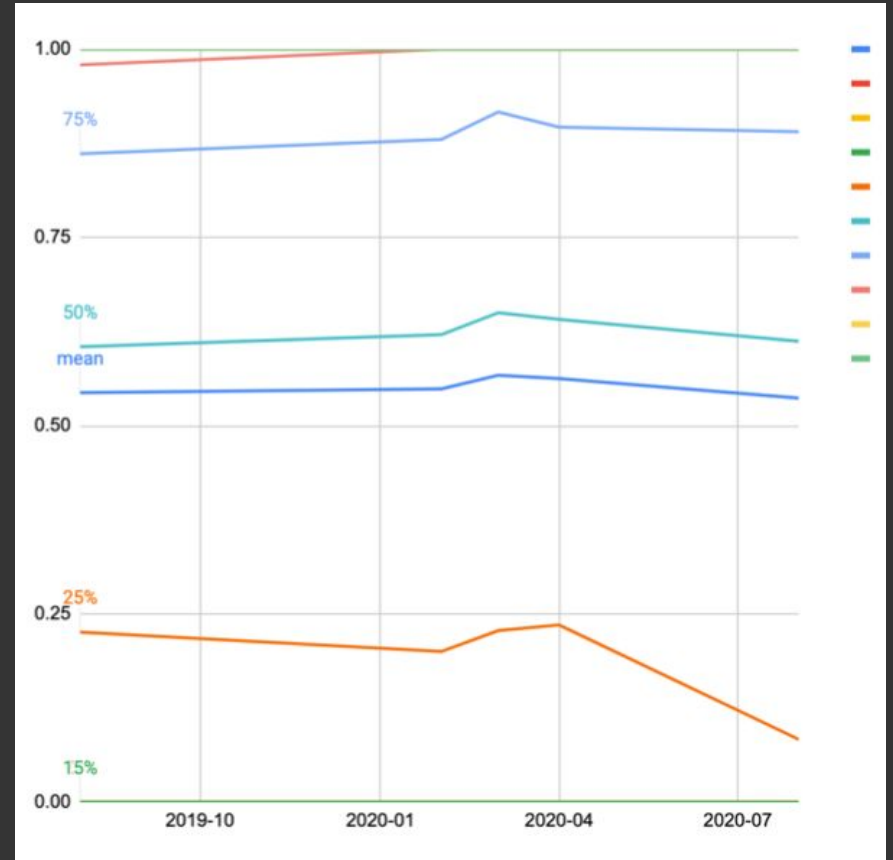
Hypothesis: more tests written in March, driving up Duration. In April, a concerted effort on optimization

Success rate

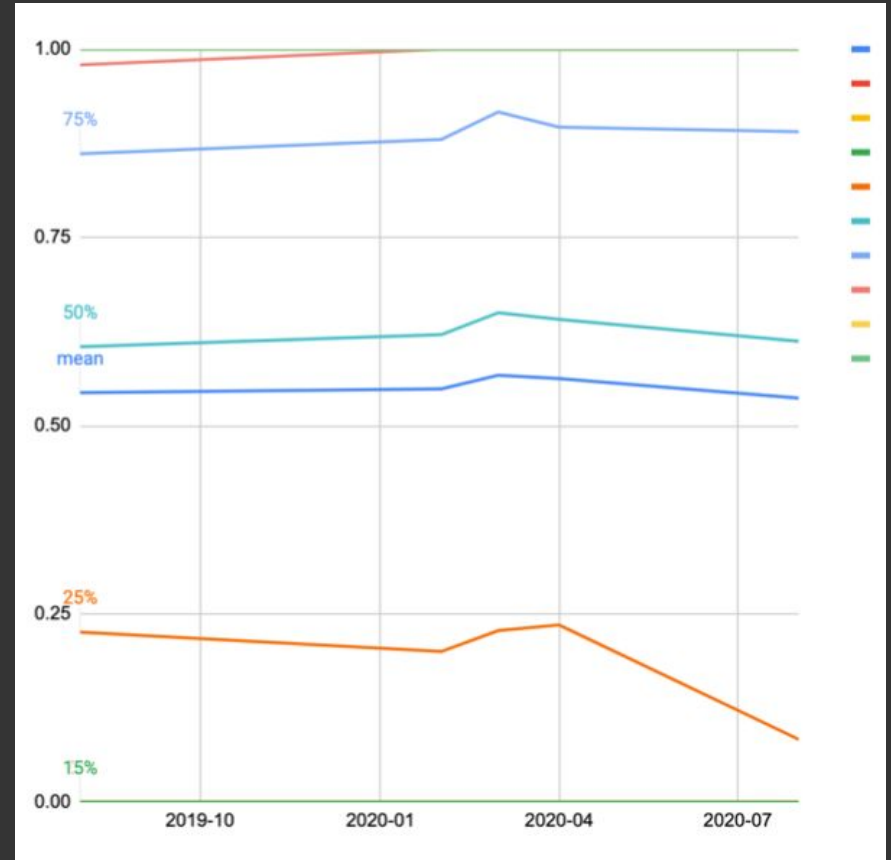
Success rate in a global pandemic



Success rate in a global pandemic



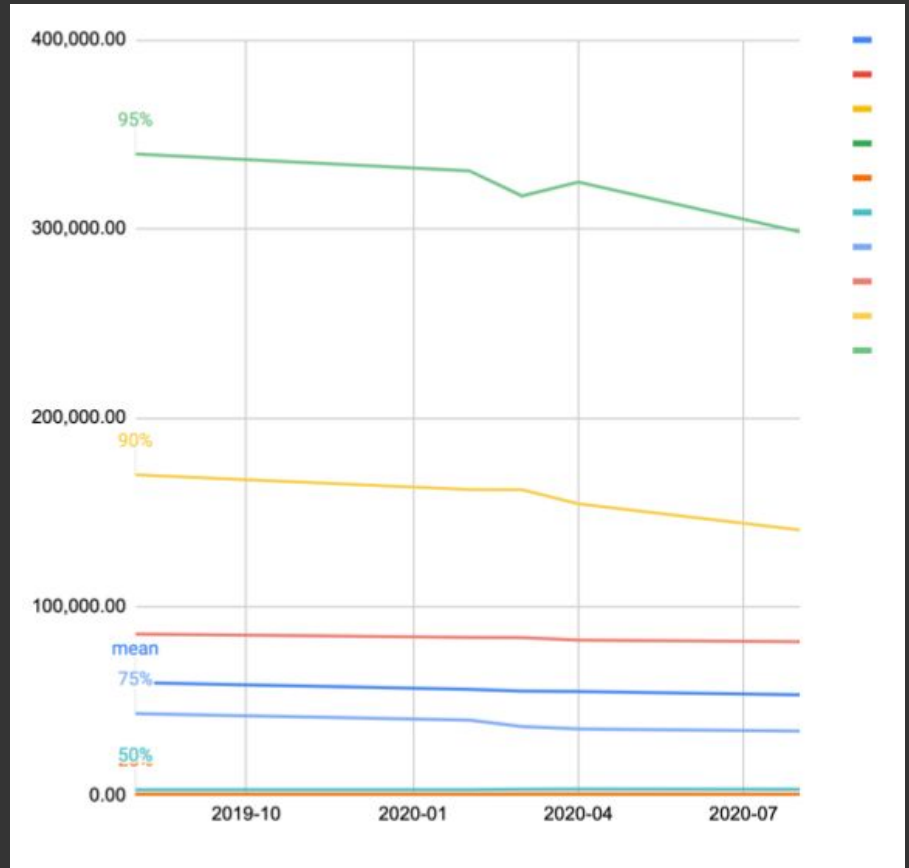
Success rate in a global pandemic



**Hypothesis: people working hard
on core business stability**

Recovery Time

Recovery time in a global pandemic



**Hypothesis: few distractions*
working at home**

Important to set targets

	Median CircleCI Developer	Suggested Benchmarks
Throughput The average number of workflow runs per day	0.7 times/day	Merge on any pull request
Duration The average length of time for a workflow to run	< 4 minutes	5-10 minutes
Mean time to recovery The average time between failures & their next success	< 56 minutes	Under 1 hour
Success rate The number of successful runs / the total number of runs over a period of time	80% for default branch	90% or better on default branch

Things that make you go 🤔

Success Rate on default branch
higher than on non-default

Duration on default branches *faster*
at every percentile

Recovery Time lower on default
branches at every percentile

**What development
practices definitively work?**

Success Rate does not correlate
with company size

Duration is longest
for teams of one

Recovery Time decreases with
increased team size (up to 200)

Performance is better
with >1 contributor

**Software is
collaborative**

Language by Throughput

- | | |
|---------------|----------------------|
| 1. Ruby | 11. PHP |
| 2. TypeScript | 12. Java |
| 3. Go | 13. C# |
| 4. Python | 14. Jupyter Notebook |
| 5. Kotlin | 15. Shell |
| 6. Elixir | 16. Vue |
| 7. Swift | 17. C++ |
| 8. HCL | 18. HTML |
| 9. JavaScript | 19. CSS |
| 10. TSQL | 20. Dockerfile |

Language by Success Rate

- | | |
|----------------|----------------------|
| 1. Vue | 11. Elixir |
| 2. CSS | 12. PHP |
| 3. Shell | 13. Jupyter Notebook |
| 4. Dockerfile | 14. Python |
| 5. TSQL | 15. Ruby |
| 6. HTML | 16. Java |
| 7. HCL | 17. Kotlin |
| 8. Go | 18. C# |
| 9. TypeScript | 19. C++ |
| 10. JavaScript | 20. Swift |

Language by fastest TTR

- | | |
|---------------|----------------------|
| 1. Go | 11. Vue |
| 2. JavaScript | 12. Jupyter Notebook |
| 3. Elixir | 13. Kotlin |
| 4. HCL | 14. Java |
| 5. Shell | 15. Scala |
| 6. Python | 16. Ruby |
| 7. TypeScript | 17. PHP |
| 8. CSS | 18. TSQL |
| 9. C# | 19. Swift |
| 10. HTML | 20. C++ |

Language by shortest duration

- | | |
|---------------------|----------------|
| 1. Shell | 11. PHP |
| 2. HCL | 12. TypeScript |
| 3. CSS | 13. Java |
| 4. HTML | 14. Elixir |
| 5. Gherkin | 15. TSQL |
| 6. JavaScript | 16. Kotlin |
| 7. Vue | 17. Scala |
| 8. Go | 18. Ruby |
| 9. Jupyter Notebook | 19. C++ |
| 10. Python | 20. Swift |

**“Don’t deploy on Friday” is
not a thing.**

“Don’t Deploy on Friday” is not a thing

- 70% less **Throughput** on weekends
- 11% less **Throughput** on Friday (UTC)
- 9% less **Throughput** on Monday (UTC)

2021/22 Sneak Peek

1. Workflows with 0 tests increase YoY, but decrease as total of all workflows
2. More deployments YoY
3. Change validation

2021/22 Sneak Peek

Software delivery performance metric	Elite	High	Medium	Low
 Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
 Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
 Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
 Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

50th percentile on CircleCI fit into the “Elite performer” category on the 2021 State of DevOps report

Full Report



<https://circle.ci/ssd2020>

Thank you.

For feedback and swag: circle.ci/jeremy



Timeline.jerdog.me



IAmJerdog



jerdog



/in/jeremymeiss

 @IAmJerdog