DevOps & Software Delivery in a Global Pandemic

ConFoo.CA 20

@IAmJerdog


2019
State of DevOps Report
puppet
circleci
splunk>
2017
State of DevOps Report
Presented by:
puppet + DORA
DEVOPS RESEARCH & ASSESSMENT
Sponsored by:
Hewlett Packard Enterprise
splunk>
amazon web services
Atlassian
WAVEFRONT by vmware
Electric Cloud
Deloitte.
2018
State of DevOps Report
puppet · splunk>
aws
CYBERARK
Cognizant
2016
State of DevOps Report
2015
STATE
REPORT

@IAmJerdog


John Allspaw
@allspaw
Normal 0%
Work-as-imagined versus work-as-done
Variability and graceful extensibility
YOUR "PLANS"
THE UNIVERSE'S PLANS FOR YOU
DOGHOUSEDIARIES
FROM INNOVATION TO SOLUTION
24
3:00 AM · Apr 28, 2016 · Twitter for iPhone

@IAmJerdog

performance described

vs

performance derived

@IAmJerdog

Jeremy Meiss

Director, DevRel & Community

circleci

ConFoo.CA 20

@IAmJerdog

2+ million
jobs/day

43,000+
orgs

** 40k in 2019*

290,000+
projects

** 150k in 2019*

1,000x
Larger than surveys

Four classic metrics

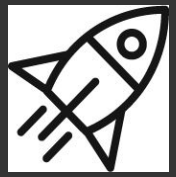
Deployment frequency

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

@IAmJerdog

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

@IAmJerdog

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

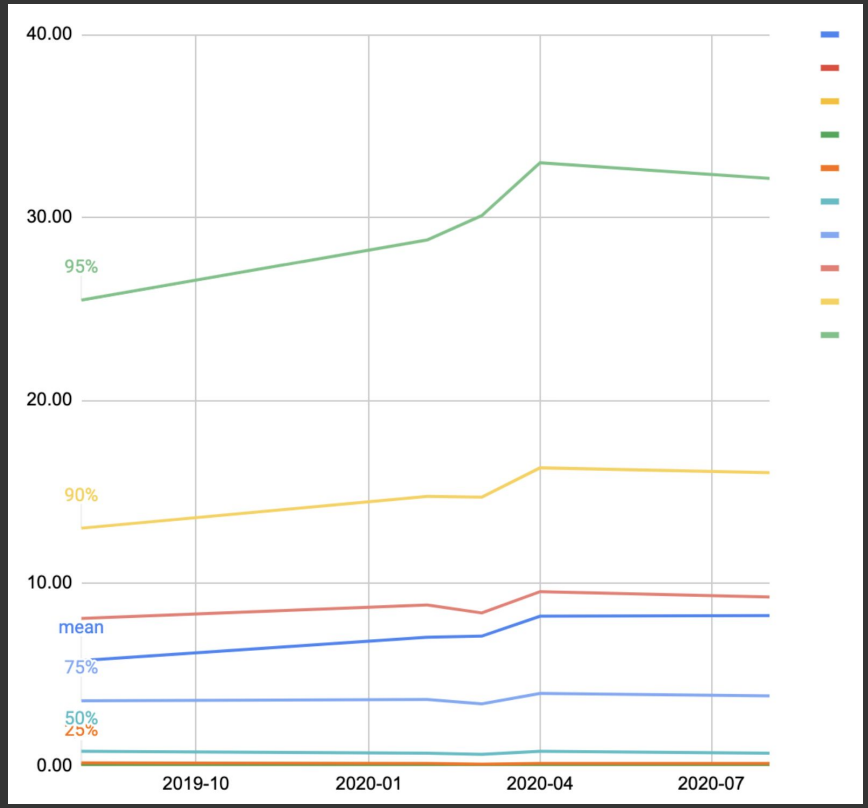
The Insight

2020/21 was quite a timeline.

@IAmJerdog

Throughput

Throughput in a global pandemic

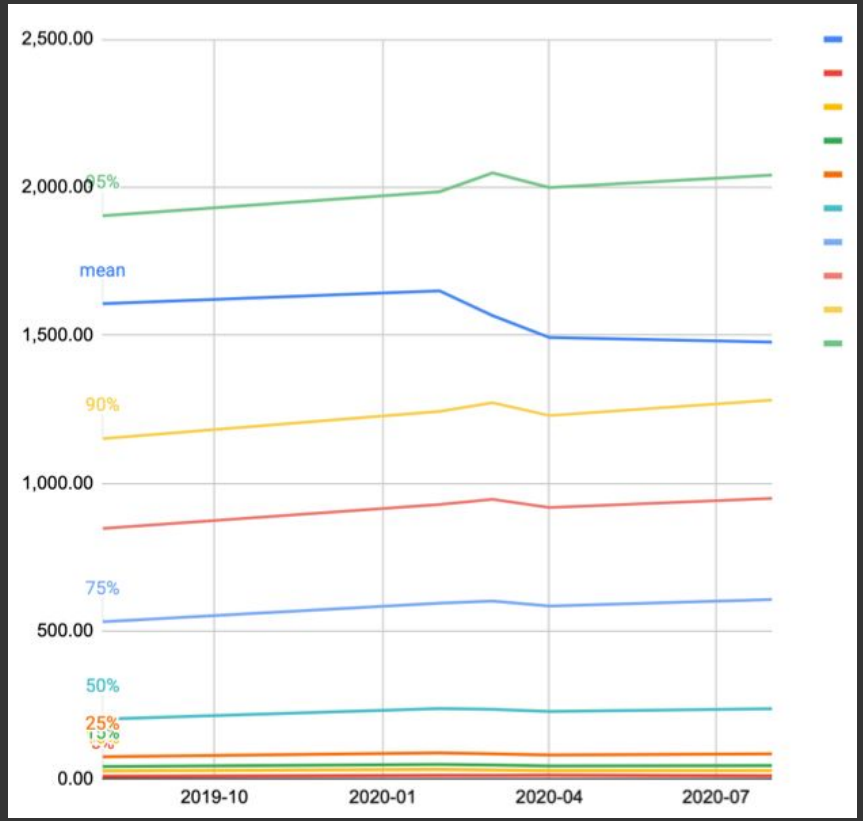


@IAmJerdog

Peak Throughput was in April 2020

Duration

Duration in a global pandemic

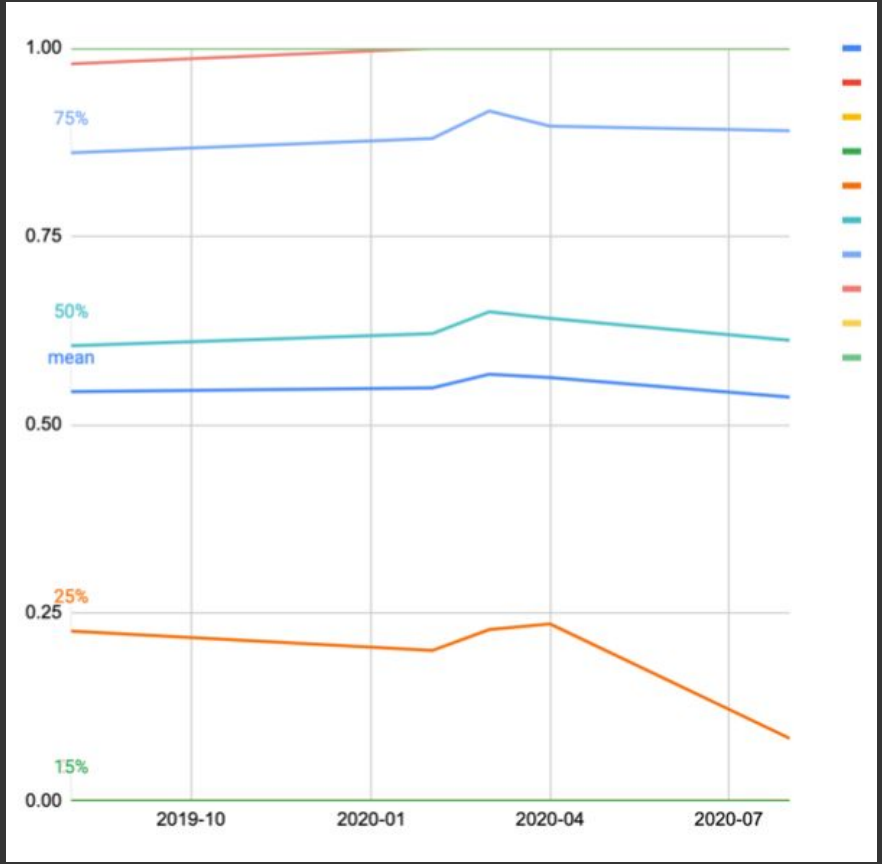


@IAmJerdog

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

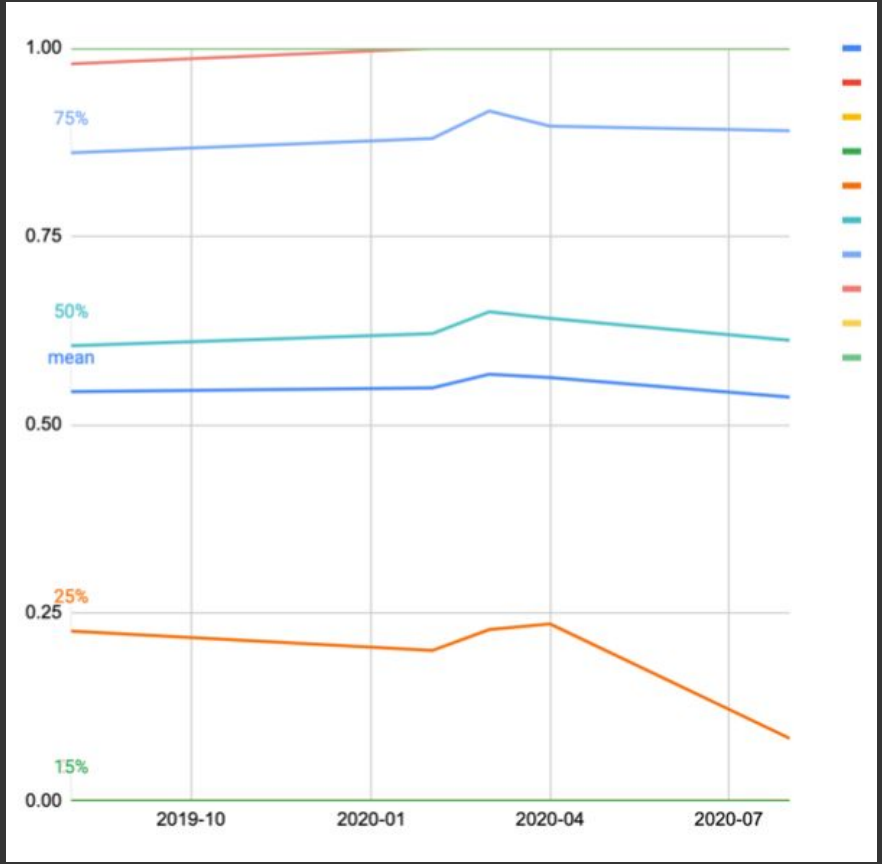
Success rate

Success rate in a global pandemic



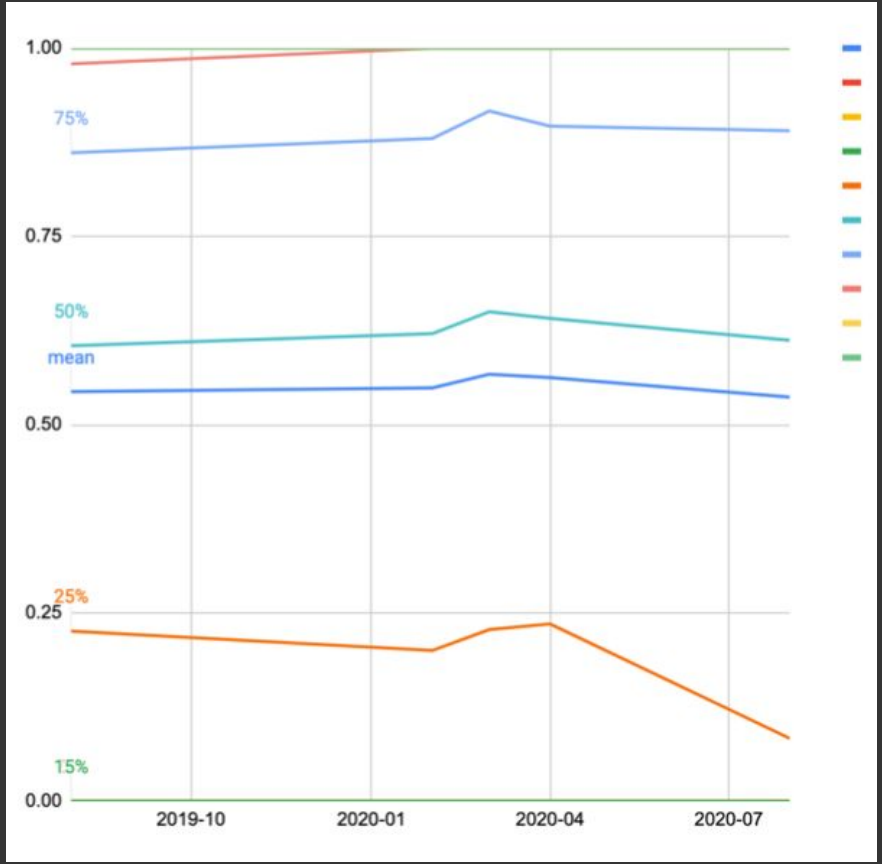
@IAmJerdog

Success rate in a global pandemic



@IAmJerdog

Success rate in a global pandemic

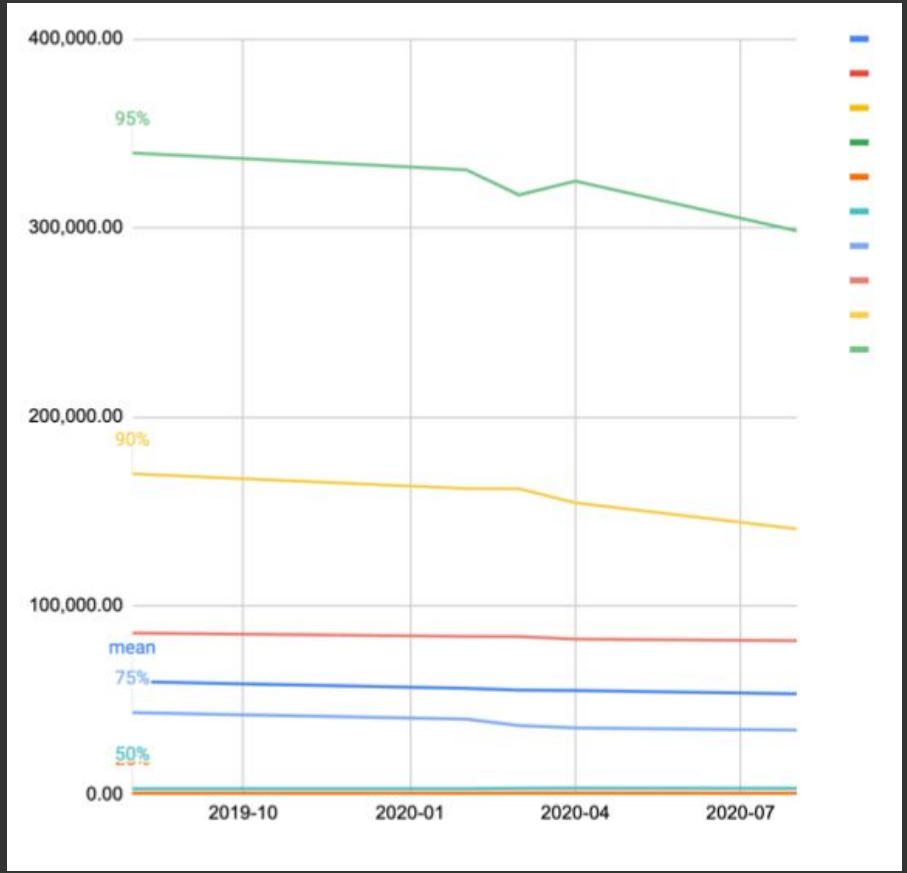


@IAmJerdog

Hypothesis: people working hard on core business stability

Recovery Time

Recovery time in a global pandemic



@IAmJerdog

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

@IAmJerdog

Language by Success Rate

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

@IAmJerdog

Language by fastest MTTR

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

Language by shortest duration

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

@IAmJerdog

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

@IAmJerdog

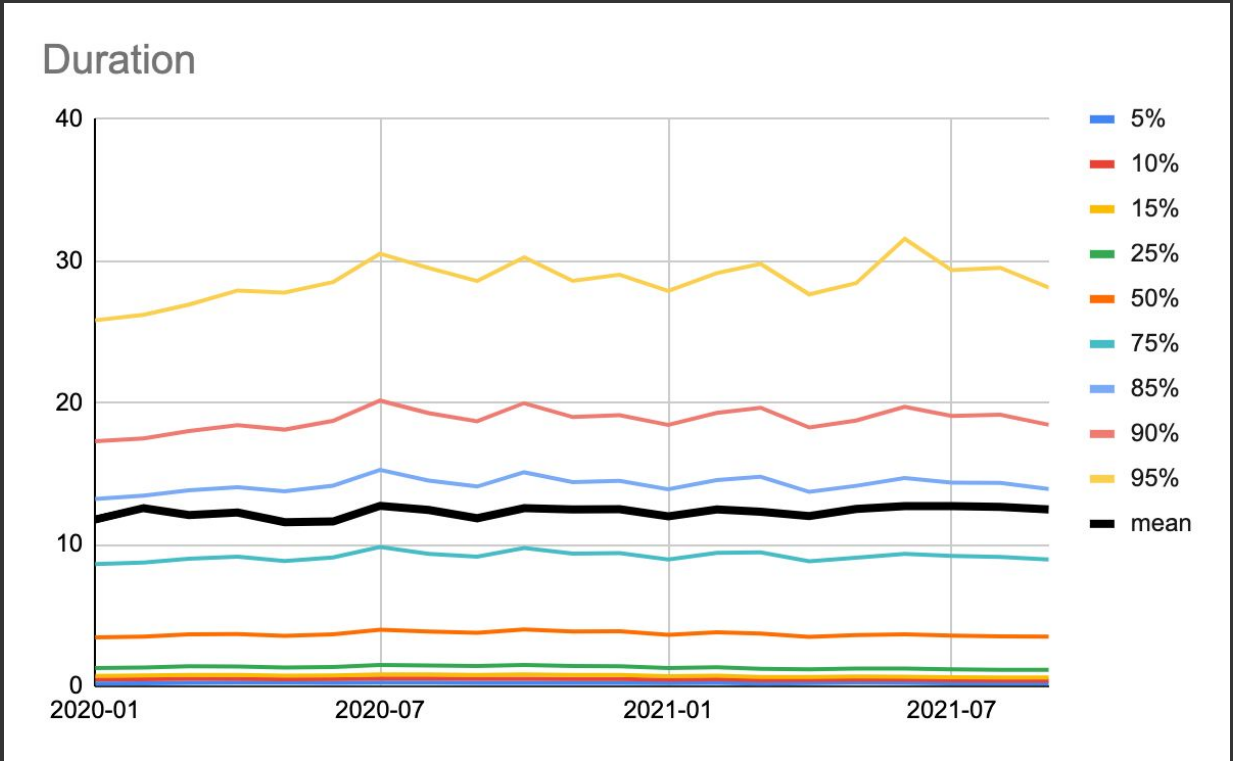
“Don’t Deploy on Friday” is not a thing

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

2021/22 Sneak Peek

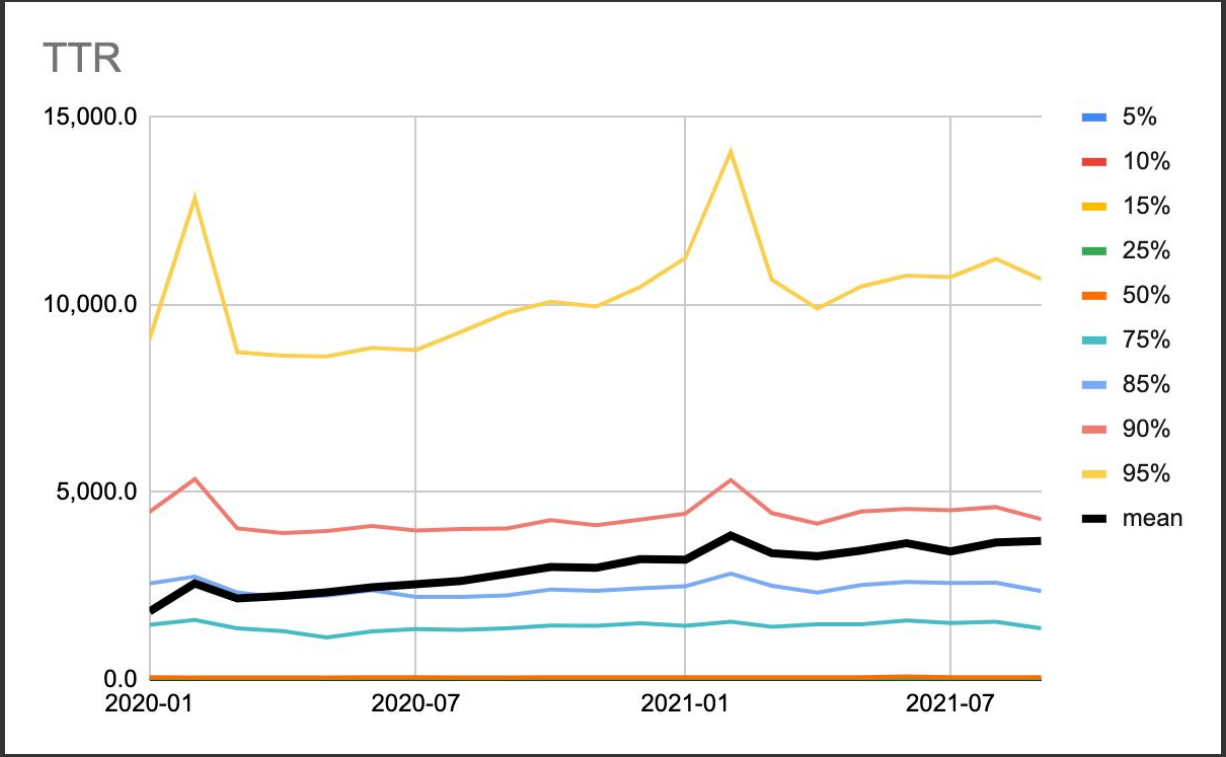
1. Workflows with 0 tests increase YoY, but decrease as total of all workflows
2. More deployments YoY
3. Change validation

2021/22 Sneak Peek



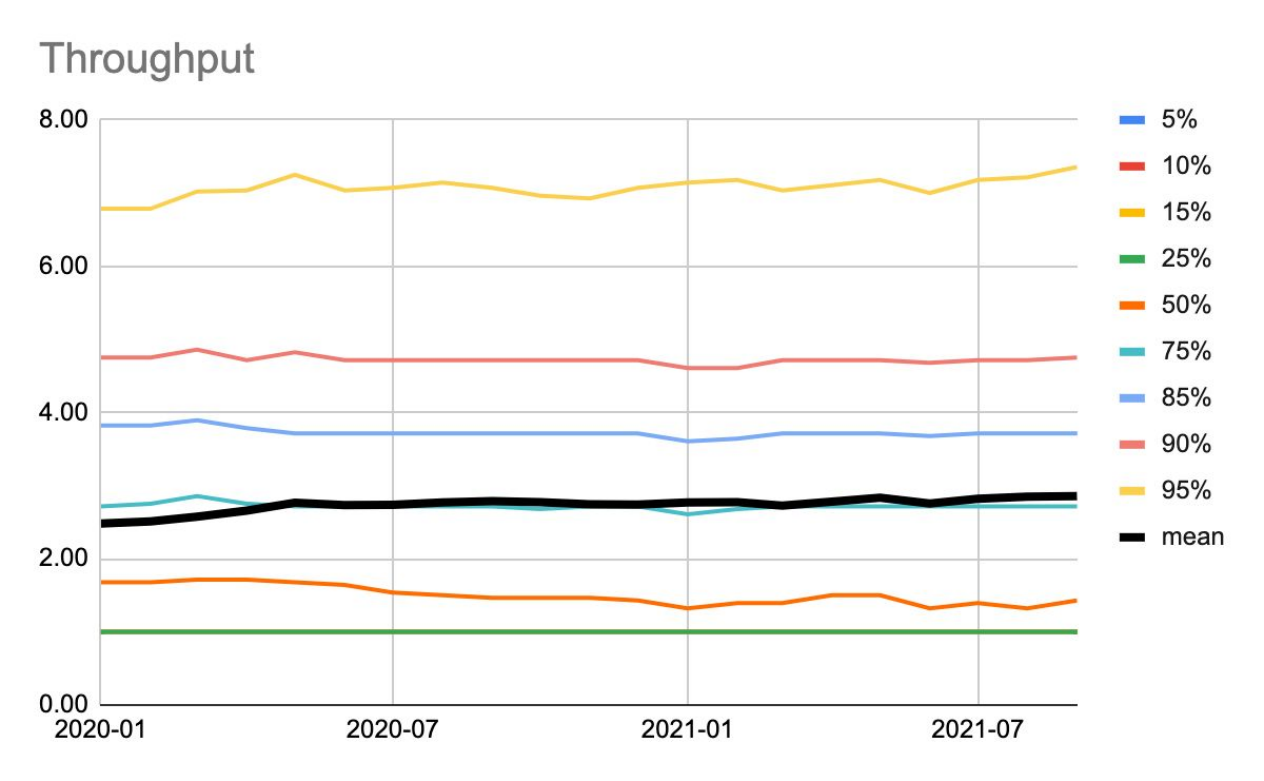
@IAmJerdog

2021/22 Sneak Peek



@IAmJerdog

2021/22 Sneak Peek



@IAmJerdog

2021/22 Sneak Peek
Top Languages by # of workflows

	Language	Workflows			
1	TypeScript	2,141,524	14	Elixir	133,194
2	JavaScript	1,989,404	15	Jupyter Notebook	130,424
3	Ruby	1,712,578	16	Vue	125,126
4	Python	1,610,022	17	C#	88,364
5	Go	684,239	18	C++	80,022
6	Java	568,671	19	Gherkin	53,844
7	PHP	475,190	20	CSS	48,955
8	Kotlin	293,032	21	Clojure	47,281
9	HCL	260,143	22	Apex	32,073
10	HTML	256,976	23	Rust	28,144
11	Shell	221,042	24	C	26,607
12	Swift	206,635	25	Dart	23,604
13	Scala	152,340			

@IAmJerdog

2021/22 Sneak Peek
Shortest Duration by Language

	Language		
1	Batchfile	14	Lua
2	SaltStack	15	Liquid
3	Makefile	16	VCL
4	Smarty	17	Open Policy Agent
5	Jsonnet	18	Groovy
6	Shell	19	Go
7	Mustache	20	Starlark
8	HCL	21	API Blueprint
9	FreeMarker	22	Roff
10	Dockerfile	23	HTML
11	PLSQL	24	R
12	Jinja	25	Python
13	Elm		

@IAmJerdog

2021/22 Sneak Peek
Shortest MTTR by Language

	Language		
1	Gherkin	14	Kotlin
2	HCL	15	Elixir
3	JavaScript	16	HTML
4	Go	17	Scala
5	Clojure	18	Jupyter Notebook
6	C#	19	Java
7	Vue	20	Swift
8	TypeScript	21	Apex
9	Ruby	22	CSS
10	Python	23	C++
11	PHP	24	Rust
12	Perl	25	C
13	Shell		

@IAmJerdog

2021/22 Sneak Peek
Throughput by Language

	Language
1	Hack
2	Slim
3	Elm
4	Mustache
5	Haskell
6	Jinja
7	Gherkin
8	Jsonnet
9	Jupyter Notebook
10	Apex
11	TypeScript
12	Swift
13	Ruby
14	Dart
15	Elixir
16	Go
17	C#
18	Kotlin
19	Blade
20	Scala
21	Python
22	LookML
23	Lua
24	CoffeeScript
25	Clojure

@IAmJerdog

2021/22 Sneak Peek
Success Rate by Language

	Language
1	Dockerfile
2	Vue
3	Shell
4	Go
5	SCSS
6	HTML
7	TypeScript
8	PHP
9	Python
10	C#
11	HCL
12	JavaScript
13	Elixir
14	Clojure
15	Jupyter Notebook
16	Java
17	Scala
18	CSS
19	PLpgSQL
20	Kotlin
21	Ruby
22	Makefile
23	Groovy
24	TSQL
25	Gherkin

@IAmJerdog

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

@IAmJerdog

Full Report

https://circle.ci/ssd2020

Thank you.

For feedback and swag: **circle.ci/jeremy**


Timeline.jerdog.me


IAmJerdog

jerdog

/in/jeremymeiss