DevOps from a Different Data Set

What 30 million workflows reveal about high performing teams

Michael Stahnke
VP Platform Engineering
@stahnma

The Setup

What are you talking about and how does this work?

The Data

Here's what the data tell us.

The Insights

Using the data, what can we apply to current industry trends and influences?

The Setup

What are you talking about
and how does this work?

201
STATE
DevO
REPO
IT R
In Partnership with
Sponsored

2016 State of DevOps Report
Presented by:
pup

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
Deloitte.
ElectricCloud

2018 State of DevOps Report
Presented by:
puppet + splunk>
Sponsored by:
aws
cloudability
CYBERARK
Cognizant
diaxion
ROOT

2019 State of DevOps Report
Presented by
puppet
circleci
splunk>
Sponsored by
ANITIAN
servicenow

2016 State of DevOps Report

2018 State of DevOps Report

Presented by: puppet + splunk>

Sponsored by: aws cloudability CYBERARK Cognizant diaxion

State of DevOps Report

DORA DEVOPS RESEARCH & ASSESSMENT

amazon web services Atlassian Electric Cloud Deloitte. WAVEFRONT by vmware

2019 State of DevOps Report

Presented by puppet circleci splunk>

Sponsored by ANITIAN servicenow

Performance derived vs performance described

44,000 orgs

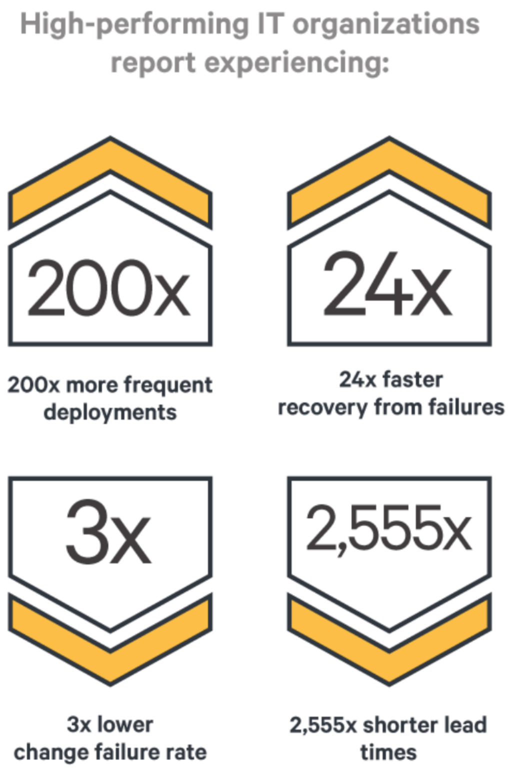
160,000 projects

1000x larger than all State of DevOps Surveys

What’s changed year over year?

Second Year for Analysis

Year	2019	2020
Days in set	30	30
Orgs	>40,000	>44,000
Projects	>150,000	>160,000



High-performing IT organizations report experiencing:

200x

200x more frequent deployments

24x

24x faster recovery from failures

3x

3x lower change failure rate

2,555x

2,555x shorter lead times

High-performing organizations are decisively outperforming their lower-performing peers in terms of throughput.

Mapping Metrics

State of DevOps Report Metrics	Description when mapping to CI	Metric
Deployment Frequency	How often you initiate a pipeline	Throughput
Lead time to Change	Pipeline duration	Duration
Change Failure Rate	Pipeline failure rate	Success Rate
MTTR	Time from red to green	Recovery Time

The Setup

What are you talking about and how does this work?

The Data

Here's what the data tell us.

The Insights

Using the data, what can we apply to current industry trends and influences?

The Data

Here's what the data tell us.

Throughput

How often do you push code that triggers CI?

Most projects configured to run per push to git server

Throughput

Percentile	2020 Value
5p	0.03
50p	0.70
90p	16.03
95p	32.125
Mean	8.22

Most projects are not deploying dozens of times per day

Why is this different from survey data?

“Primary application or service you work on”

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

Those leveraging CI well, are doing so even more

There are fewer developers worldwide pushing code

Duration

How long does it take to get results?

5% of builds finish in < 12 seconds

* That’s roughly 500,000 builds in this sample

Duration

Percentile	2020 Value
5p	12 sec
50p	3.96 min
90p	21.35 min
95p	34.01 min
Mean	24.6 min

Half of all builds finish in under 4 minutes

Duration delta in a year

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

All pipelines are running longer

Duration delta in a year

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

Success Rate

How often does your pipeline complete with a green status?

Success Rate

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

Some of our sample dabbles with CI, but doesn’t get a working build

Some of our sample saw no failures within a month

Success Rate

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

Success Rate

Percentile	2020 Value	2019 Value
50p	61%	60%
75p	89%	86%
85p	100%	98%

Recovery Time

Time a pipeline sits in a failure state

Recovery Time

Percentile	2020 Value
5p	2.06 min
50p	55.11 min
90p	39 hours
95p	3.4 days
Mean	14.85 hours

Quick **Recovery Time** can be from multiple contributors running in parallel

The gap between 50th and 75th percentiles looks like it represents waiting until tomorrow to fix a failed build
(from 55 min at 50p to 9.5 hours at 75p)

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

Fastest **Recovery Times** have improved (10th and percentile and lower) year over year

The Setup

What are you talking about and how does this work?

The Data

Here's what the data tell us.

The Insights

Using the data, what can we apply to current industry trends and influences?

The Insights

Using the data, what can we apply to current industry trends and influences?

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)

Longest **Recovery Times** are from teams of one.

Performance is better with more than one contributor as shown by multiple indicators

Software is collaborative

Is “Don’t Deploy on Friday” a real thing?

70% less **Throughput** on weekends

11% less **Throughput** on Friday (UTC).

9% less **Throughput** on Monday (UTC).

Conclusion: About the same amount of work happens Monday or Friday. So people not holding back on pushing code on Fridays.

What Language Trends emerge?

circleci

Languages in our sample

21.73%	JavaScript	2.44%	Vue
11.36%	TypeScript	2.12%	Kotlin
9.56%	Python	1.70%	HCL
9.04%	Ruby	1.59%	Swift
6.16%	HTML	1.26%	C++
5.37%	Java	1.21%	Dockerfile
4.92%	PHP	1.08%	C#
3.89%	Go	1.00%	TSQL
3.17%	CSS	0.96%	Jupyter Notebook
2.99%	Shell	0.83%	Elixir

Language **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 **Success Rate** at 50p

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

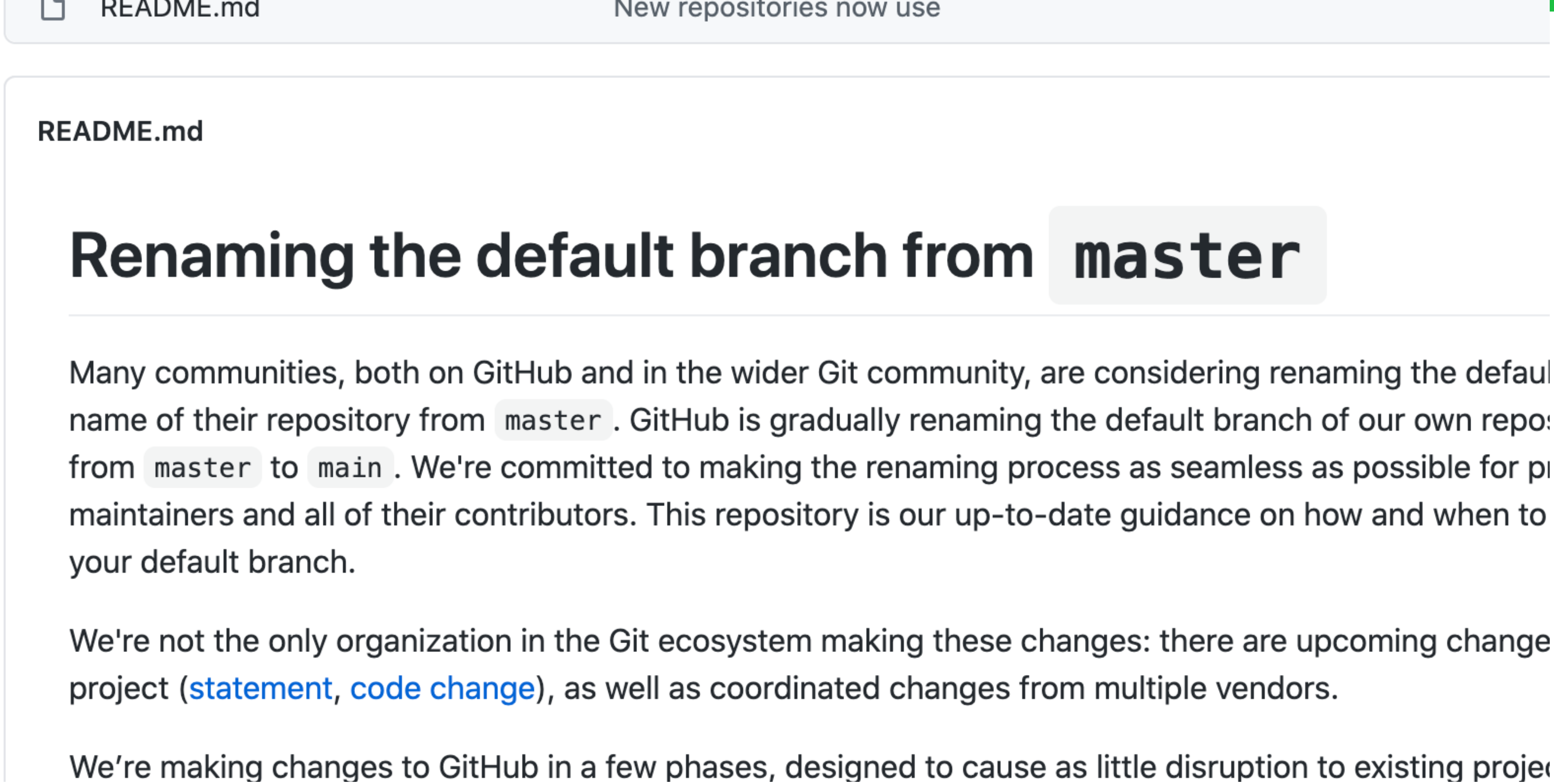
Language **Recovery Time** at 50p

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 **Duration** at 50p

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

Branch Information


README.md New repositories now use

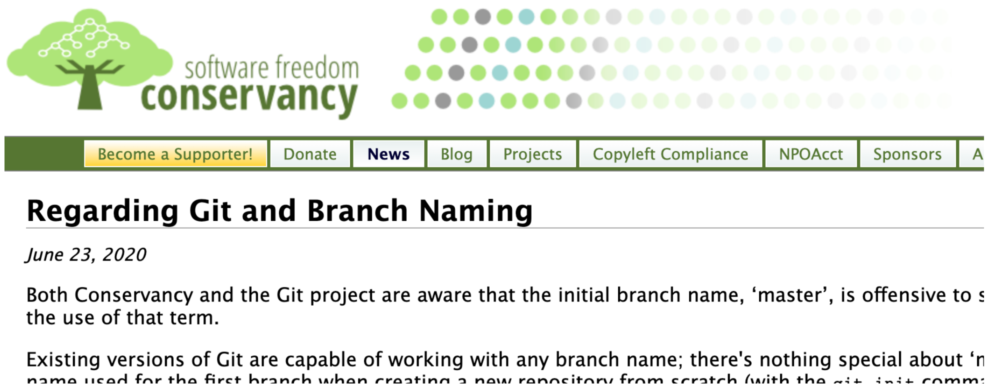
README.md

Renaming the default branch from `master`

Many communities, both on GitHub and in the wider Git community, are considering renaming the defaul
name of their repository from `master`. GitHub is gradually renaming the default branch of our own repo
from `master` to `main`. We're committed to making the renaming process as seamless as possible for p
maintainers and all of their contributors. This repository is our up-to-date guidance on how and when to
your default branch.

We're not the only organization in the Git ecosystem making these changes: there are upcoming change
project (statement, code change), as well as coordinated changes from multiple vendors.

We're making changes to GitHub in a few phases, designed to cause as little disruption to existing proje



Regarding Git and Branch Naming

June 23, 2020

Both Conservancy and the Git project are aware that the initial branch name, 'master', is offensive to s the use of that term.

Existing versions of Git are capable of working with any branch name; there's nothing special about 'n name used for the first branch when creating a new repository from scratch (with the `git init` comm

Did the use of `master` branch decrease?

Not in any significant way….yet.

Teams are innovating and experimenting on feature branches

Success Rate on default branch higher than on non-default branches

Success Rate is 80% on the default branch at 50th percentile and 100% for 75th percentile and above

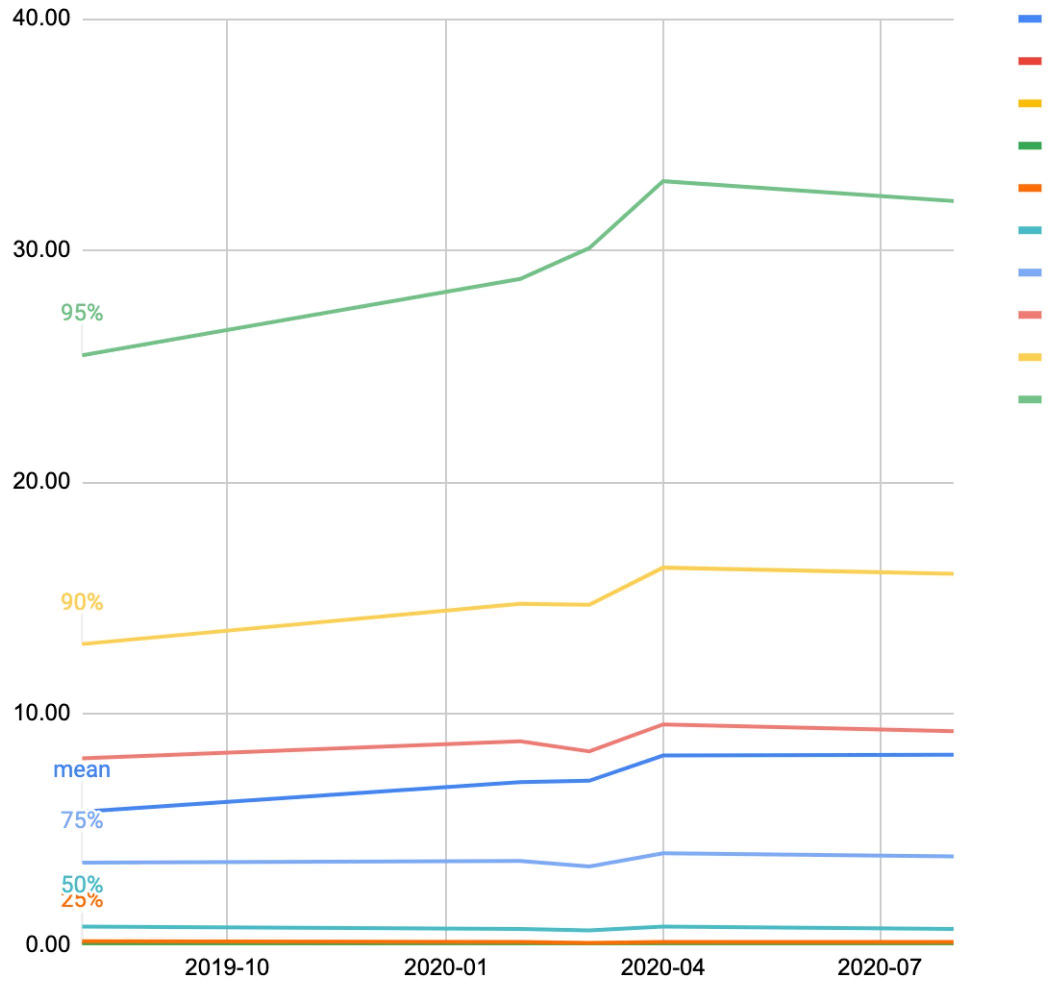
Success Rate at 50p is 80% for default and 58% for non-default branches

Duration on default branches are faster at every percentile.

Recovery Time is lower on default branch at every percentile.

How has the global pandemic impacted team performance?

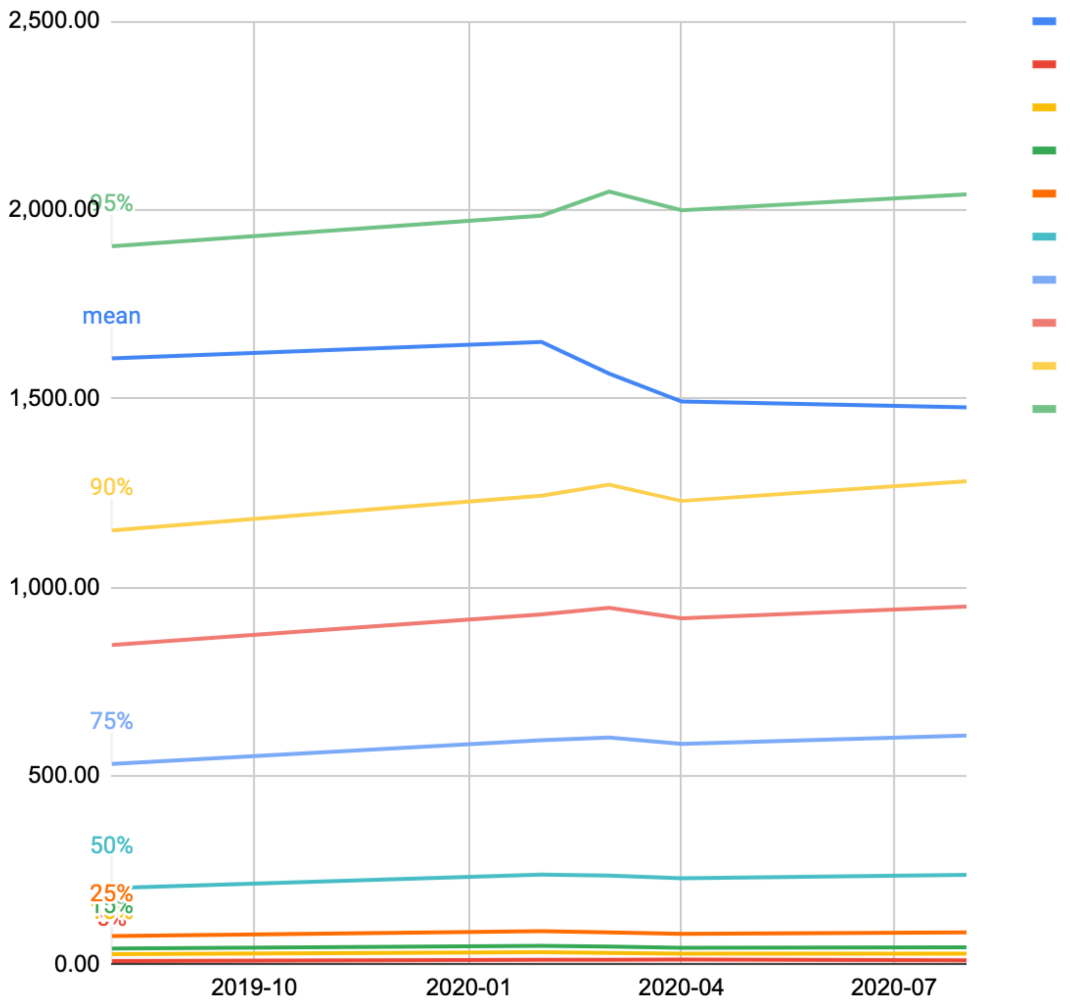
Throughput


40.00
30.00
20.00
10.00
0.00
95%
90%
mean
75%
50%
25%
2019-10
2020-01
2020-04
2020-07

Peak **Throughput** was April 2020

After April,
Throughput falls a bit

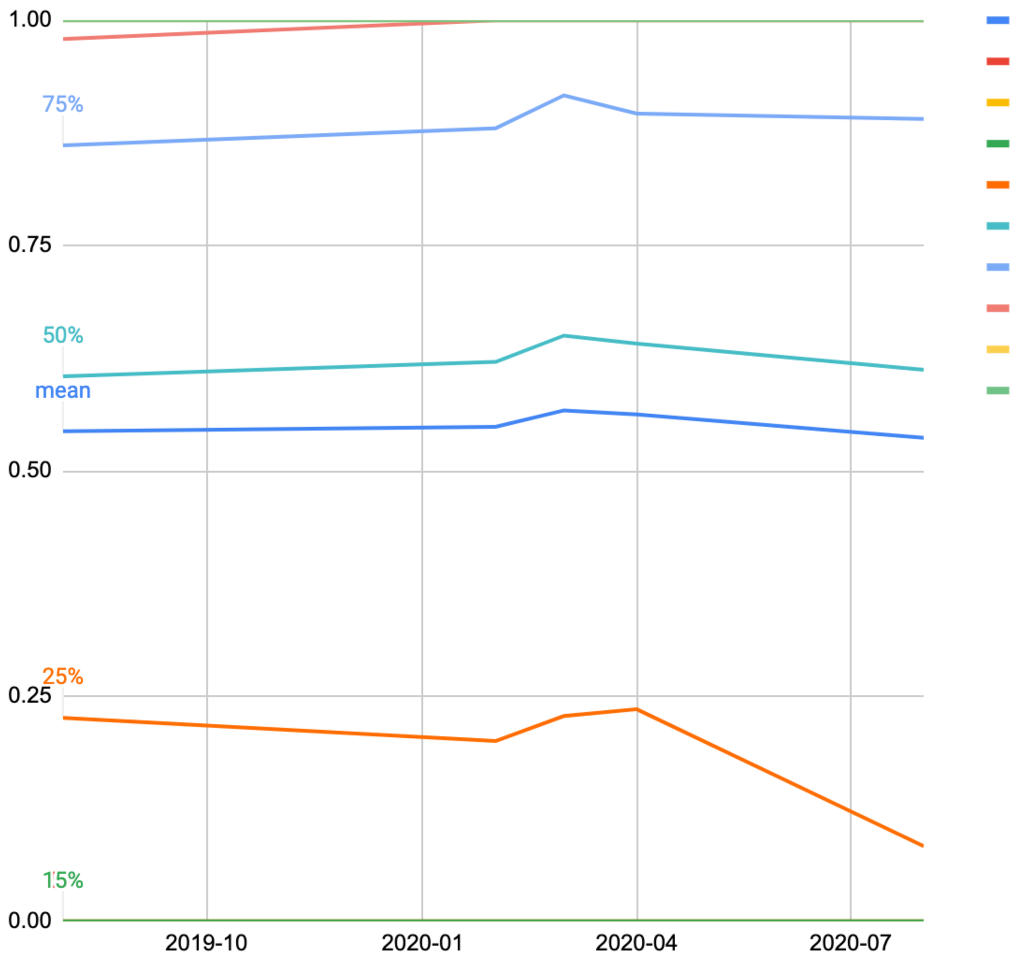
Duration


2,500.00
2,000.00
1,500.00
1,000.00
500.00
0.00
95%
mean
90%
75%
50%
25%
2019-10
2020-01
2020-04
2020-07

For 75th percentile and above, **Duration** increased in Feb, the increase accelerated in March, decreased in April, and increased again in August to longest **Duration**

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

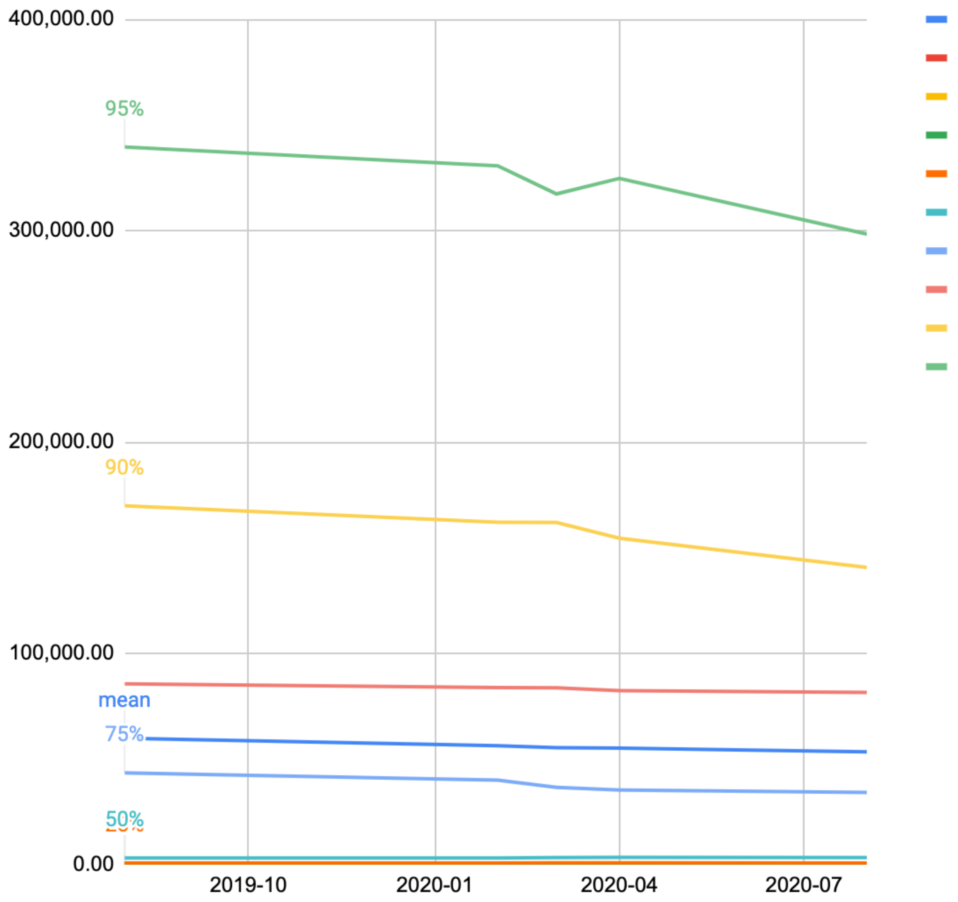
Success Rate



Success Rates were the highest on record in April 2020

Hypothesis: people working hard on core business stability

Recovery Time



Since April, **Recovery Time** has been improving

Orgs with the longest **Recovery Times** (75th percentile and above) have improved significantly

Hypothesis: Fewer distractions* working at home

*For some values of distraction.

Final Thoughts

When mapped against survey surveying data, CI users at 50p show up between medium and high performers at an org level (vs project level).

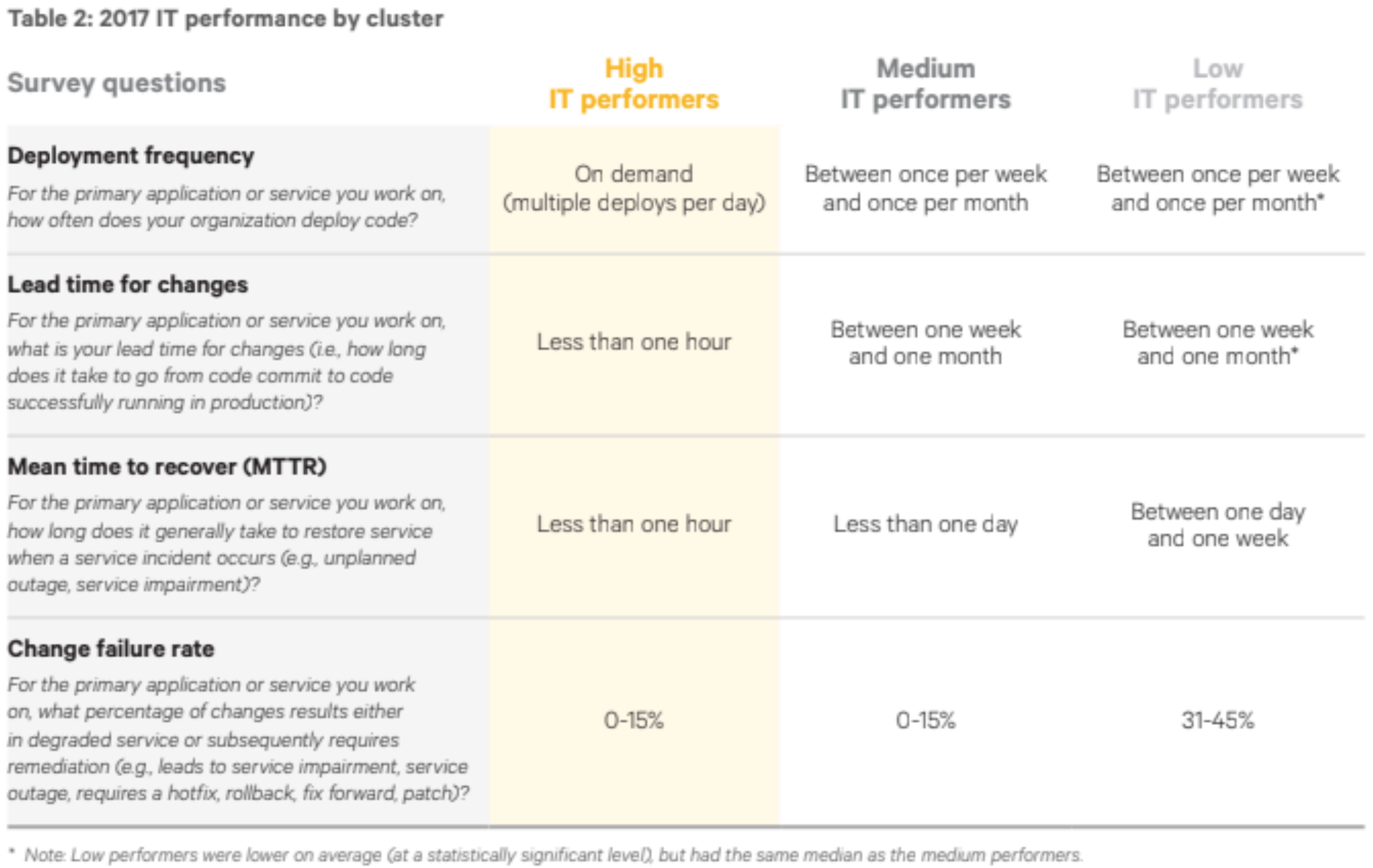


Table 2: 2017 IT performance by cluster

Survey questions	High IT performers	Medium IT performers	Low IT performers
Deployment frequency *For the primary application or service you work on, how often does your organization deploy code?*	On demand (multiple deploys per day)	Between once per week and once per month	Between once per week and once per month*
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 commit to code successfully running in production)?*	Less than one hour	Between one week and one month	Between one week and one month*
Mean time to recover (MTTR) *For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?*	Less than one hour	Less than one day	Between one day and one week
Change failure rate *For the primary application or service you work on, what percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?*	0-15%	0-15%	31-45%

* *Note: Low performers were lower on average (at a statistically significant level), but had the same median as the medium performers.*

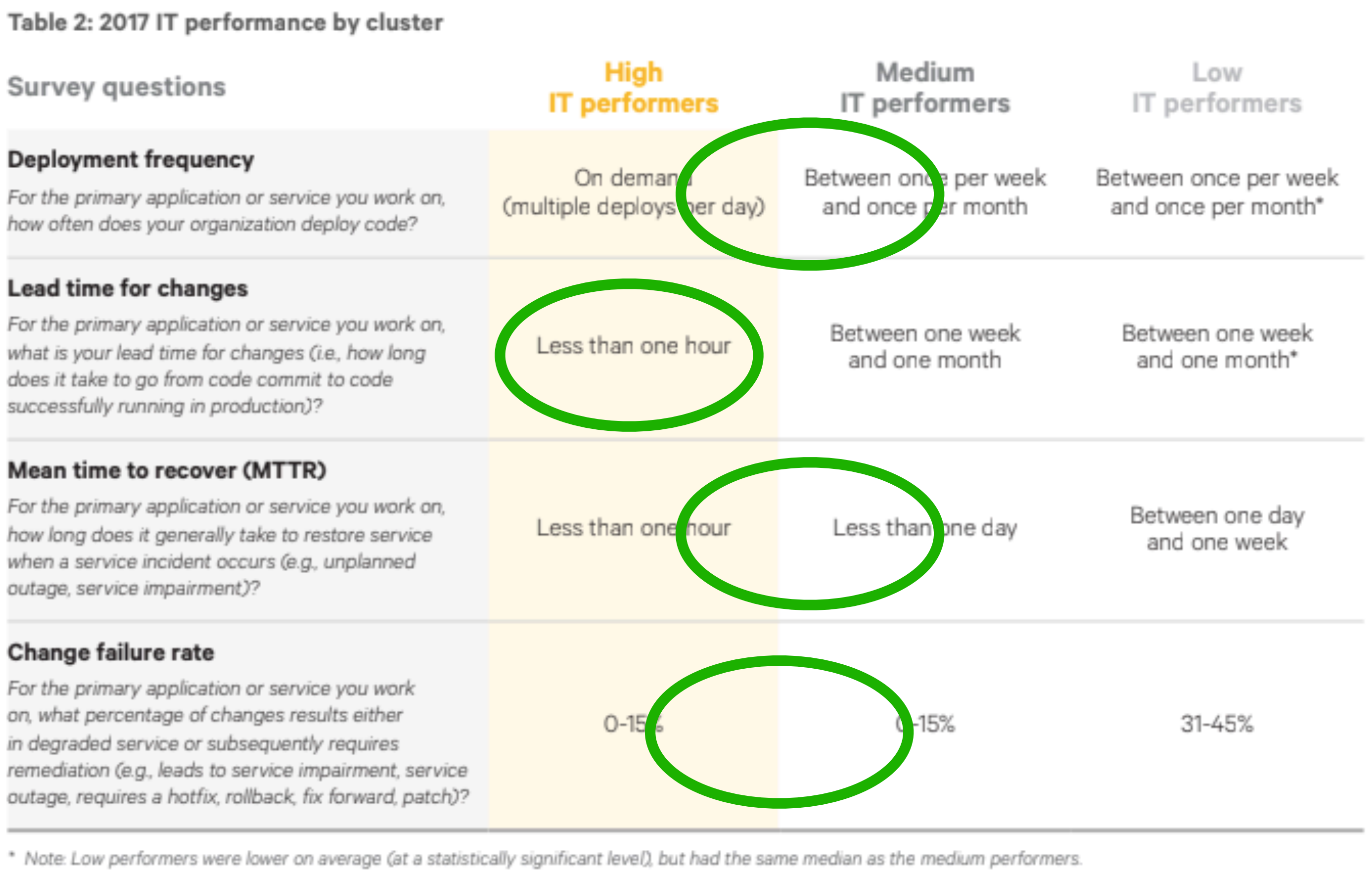


Table 2: 2017 IT performance by cluster

Survey questions	High IT performers	Medium IT performers	Low IT performers
Deployment frequency *For the primary application or service you work on, how often does your organization deploy code?*	On demand (multiple deploys per day)	Between once per week and once per month	Between once per week and once per month*
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 commit to code successfully running in production)?*	Less than one hour	Between one week and one month	Between one week and one month*
Mean time to recover (MTTR) *For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?*	Less than one hour	Less than one day	Between one day and one week
Change failure rate *For the primary application or service you work on, what percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?*	0-15%	0-15%	31-45%

* *Note: Low performers were lower on average (at a statistically significant level), but had the same median as the medium performers.*

If you are average at using a CI platform, you'll be right on the line between medium and high performer.

Our most frequent CI users have better outcomes on our four critical metrics

More collaborators means better outcomes

We're hiring.

circleci.com/careers

Thank you

Michael Stahnke @stahnma

Special thanks to Ron Powell and Melissa Santos who gathered this data and were able to answer my questions about it.