



How we use telemetry to improve Octopus Deploy



Terence Wong

October 13, 2021 • 4 mins

Over the years, Octopus Deploy has grown to include many new features and areas within the app. Customers use these to view the dashboard, create or deploy releases, author projects, or run runbooks against their infrastructure. To serve our customers better, we want to know how fast and responsive the experience is.

Like many software companies, we collect telemetry to measure how customers use and experience the product. Some of the telemetry we collect is the timing of API calls and database operations, which we call performance telemetry. We surface this telemetry in our custom-built engineering dashboard, code-named Crow's Nest.

Crow's Nest provides a high-level overview of how users are experiencing the product. A web request triggers when a customer wants to visit a page or poll an endpoint, and Crow's Nest measures how long the request takes.

We also track these requests over several versions to ensure Octopus Deploy's responsiveness continues to improve.

Apdex

We calculate an Apdex (Application Performance Index) score, which aims to convert measurements into insights about user satisfaction. The formula is:

$$\text{Apdex} = (\text{SatisfiedCount} + \text{ToleratingCount} * 0.5) / \text{TotalCount}$$

Apdex uses API (web) requests that return a 2xx response. Apdex excludes certain requests that are called often and are cached.

- Requests that are returned in less than or equal to 50ms are considered to be within the satisfied threshold.
- Requests that are greater than 50ms and less than 200ms are considered to be within a tolerated threshold.

Apdex gives a uniform scale to test the customer experience. A higher number indicates a more positive user experience. We can vary thresholds to experiment with Apdex scores given a specific appetite for tolerance.

The examples in this post display the default threshold values. These are configurable in real-time in the application to view Apdex scores against different criteria.

Only web requests with a 2xx status are included in the calculations. A 2xx status indicates a successful web request. Versions with less than 50 instances sending telemetry on any given data get filtered out to remove outliers. The response times of these web requests estimate how satisfied a customer is with their service.

Visualizing Apdex and Octopus Deploy

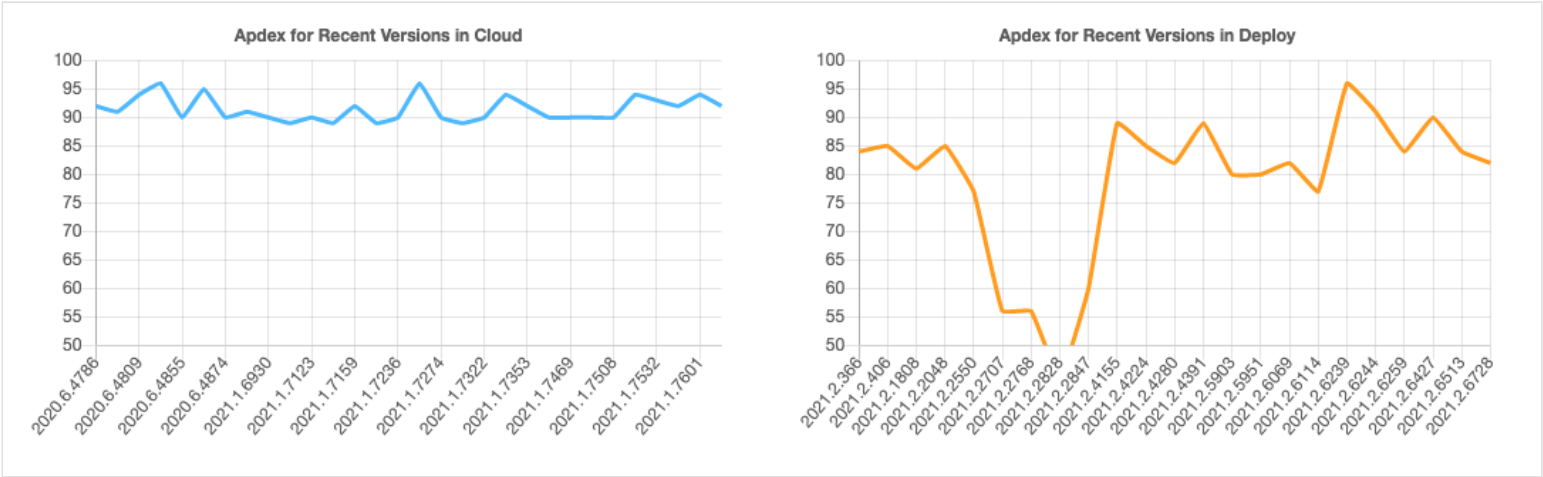
There are several ways to visualize Apdex. The following graphs show how we use Crow's Nest to display Apdex and gain valuable insights.

Apdex for Cloud and Deploy

The blue line indicates the Apdex performance of recent versions in Octopus Cloud and the deployment server. The Cloud Apdex performance has been consistent at around 90 for this period. The deployment server is the internal Octopus instance not released to customers.

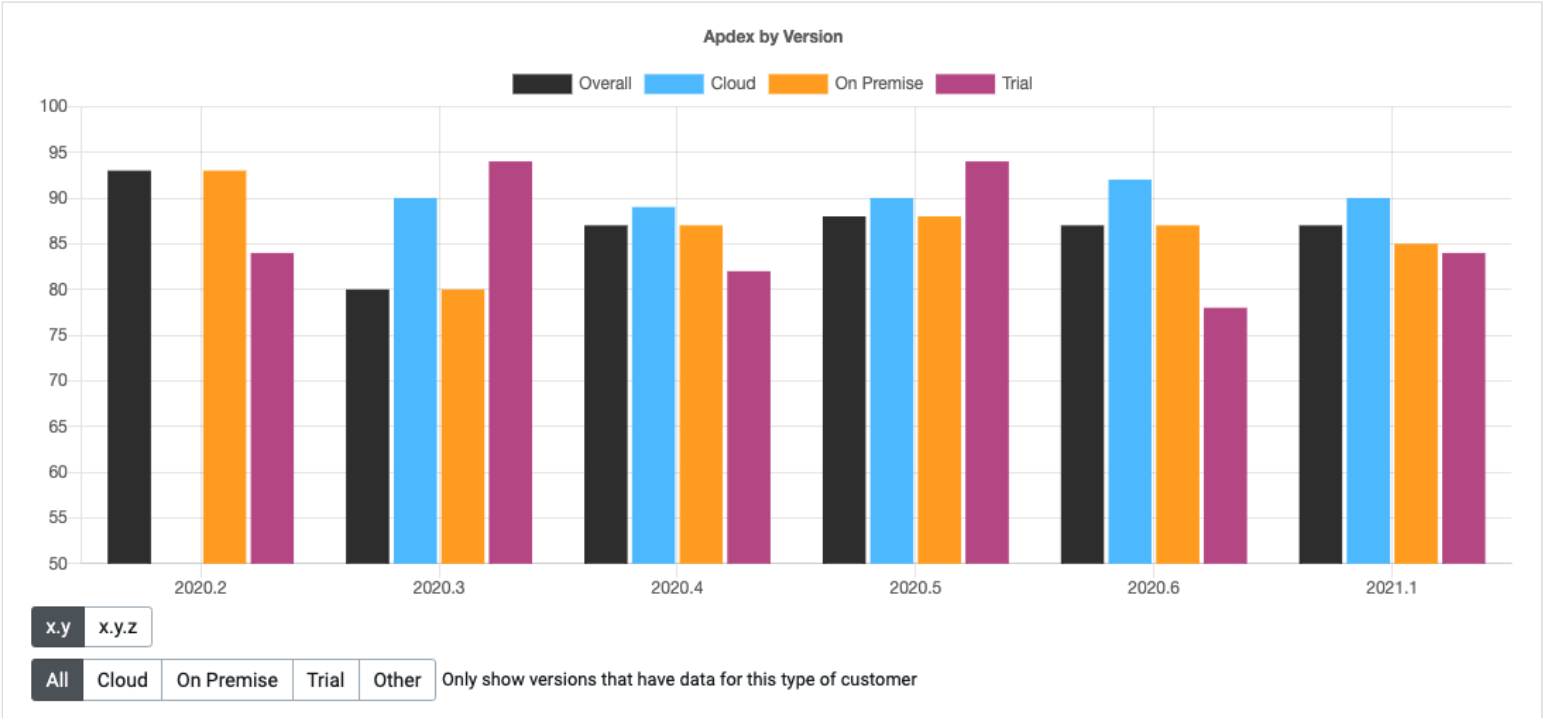
The orange graph indicates a significant dip in Apdex from 2021.2.2048, where it recovered in 2021.2.4155 as we fixed the regressions.

It's helpful to look back and see how different versions affected the user experience. If there's a significant dip in performance, we can conduct a root cause analysis to identify and address causes.



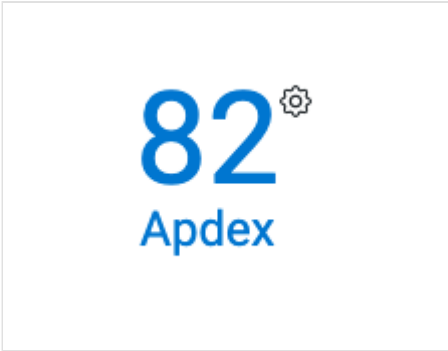
Apdex by version and license

We can compare the Apdex score for different versions and licenses. The licenses are Cloud, On Premise, Trial, and Overall. There's a lot of variability in these numbers, as they're representative of all customers.



Apdex overall score

Each customer has an overall Apdex score, which indicates the responsiveness of the customer experience.



Apdex routes

The performance of individual routes are in the customer view. We time web requests and group the results on the route. From these timings we show the mean, median and 95th percentile time.

By bucketing the requests by route, we avoid transmitting large amounts of data. We can use these metrics to identify the worst-performing routes in a later version.

Route	Apdex	Median	Mean	95th	Count	Count/day	Duration/day
{spaceid}/projects/{id}/metadata	100	10	6	20	29,032	558	3,727
{spaceid}/projectgroups/{id}	100	10	9	30	924,589	23,707	228,049
projects/{id}/metadata	100	10	5	10	3	2	8
projects/{id}/summary	100	10	15	30	2	1	15
projects/{id}/releases/{version}	99	20	20	40	128,044	4,001	81,331
projects/{id}/logo	99	10	8	30	2,309	1,154	9,602
{spaceid}/projectgroups	98	20	17	50	65,600	2,116	37,497
{spaceid}/projecttriggers/{id}	97	10	14	50	8,287	224	3,349
{spaceid}/projects/{id}/releases/{version}	96	20	25	60	1,047,017	19,389	486,466
{spaceid}/projects/{id}OrSlug	96	10	16	70	10,495	2,624	43,338
{spaceid}/projects/all	95	20	52	100	12,906,339	253,065	13,338,930
{spaceid}/projects/{id}/triggers	95	20	34	100	492,507	9,471	323,578
{spaceid}/projects/{id}/runbooks	94	20	36	200	373,786	7,188	261,209
{spaceid}/projects/{id}/summary	93	20	44	90	1,105,643	19,744	872,300
{spaceid}/projects/{id}/versioncontrolcompatibility/report	93	40	33	60	42	3	93
projectgroups	93	40	39	80	2,007,077	118,063	4,701,361
{spaceid}/projectgroups/all	92	20	40	200	1,305,964	22,912	926,284
projectgroups/all	92	30	32	80	3,557	148	4,849

Apdex route difference

We can use Crow's Nest to view the differences in routes across different versions. The project routes improved between 2020.6 and 2021.1, as denoted by the green difference indicators for the Apdex score.

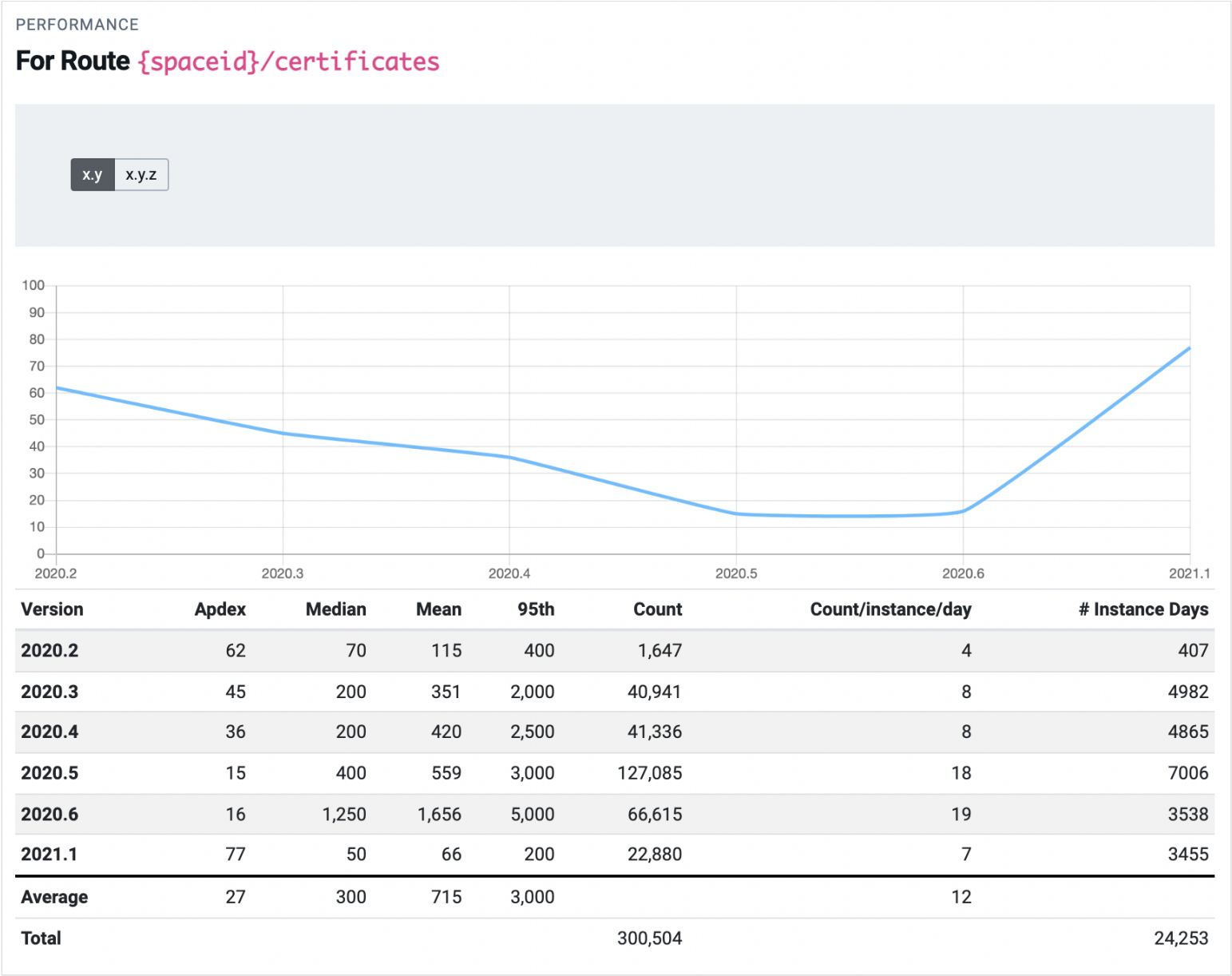
Route	Apdex <small>come val diff</small>	Median <small>come val diff</small>	Mean <small>come val diff</small>	95th <small>come val diff</small>
projects/{id}/runbookSnapshots	50	30	137	300
projects	12 61 +49	900 60 -840	803 176 -627	1,500 700 -800
{spaceid}/projects/{id}/versioncontrol/branches/{name}	46 64 +18	70 70 +0	263 174 -89	1,250 1,250 +0
projects/{id}/summary	83 100 +17	50 10 -40	51 15 -36	100 30 -70
{spaceid}/projects/{idOrSlug}	85 96 +11	40 10 -30	44 16 -28	100 70 -30
{spaceid}/projects	60 69 +9	60 40 -20	355 265 -90	1,750 1,250 -500
{spaceid}/projects/experimental/summaries	60 68 +8	70 50 -20	201 123 -78	800 500 -300
projects/all	13 19 +6	1,000 1,250 +250	2,860 1,791 -1,069	9,000 6,000 -3,000

Apdex route view

Historical performance is visible on every route. This allows us to see how the performance of a route has changed in each release.

For example, the performance of the certificates route above degraded from 2020.2 to 2020.6. The Apdex score decreased from 62 to 16.

In 2021.1, we identified the cause of this decrease in the Apdex score and resolved the issue, leading to an improved Apdex score of 77.



Future improvements

Crow's Nest helps us plan what to work on next. It highlights areas needing the most attention, that should be addressed first. It also gives us another signal when monitoring our overall application health.

Conclusion


Telemetry is a powerful tool that offers businesses a complete picture of user experiences.

Telemetry and Apdex give full visibility of each user and the performance of each route. We can compare performance across different versions and licenses. Comparing versions quantifies the effect of each update. Poor-performing routes are visible on a per-user basis or across the Octopus Deploy platform.

Our Crow's Nest tool helps to improve the customer experience, leading to more responsive interactions, and happier deployments.


Tagged with: Engineering

Related posts



Engineering

Variable use in Octopus Deploy



Stephen Heise
October 12, 2022 • 5 mins

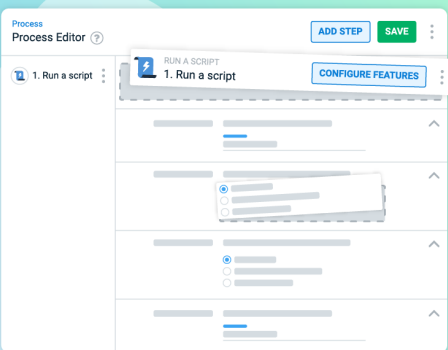


Engineering

Sharing Workers across spaces




Shawn Sesna
October 10, 2022 • 3 mins



Engineering

Improving delivery of your deployment steps



Shaun Hevey
August 8, 2022 • 13 mins

Newsletter

Logged in as Terence Wong (terence.wong@octopus.com)

Join ~48,000 DevOps professionals and sign up for the latest Octopus news, events, and opinions. No spam. Unsubscribe at any time.

Subscribe

Your privacy is important to us. Read more in our [Privacy Policy](#).

