



**Aubrey Chipman &
Roberto Perez Alcolea**

NETFLIX



Netflix's Journey to Confident Automated Changes



Brought to you by



Gradle, Inc

Netflix engineering ecosystem

N

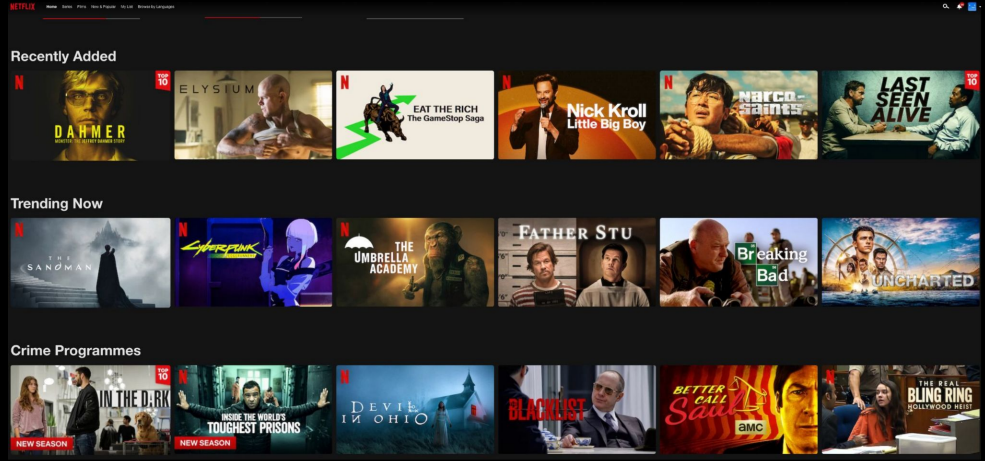


From...



Image from <https://www.wired.com/story/why-are-2-million-people-still-getting-netflix-dvds-by-mail/>

To...



NETFLIX



VS



VS



Images from <https://about.netflix.com/>

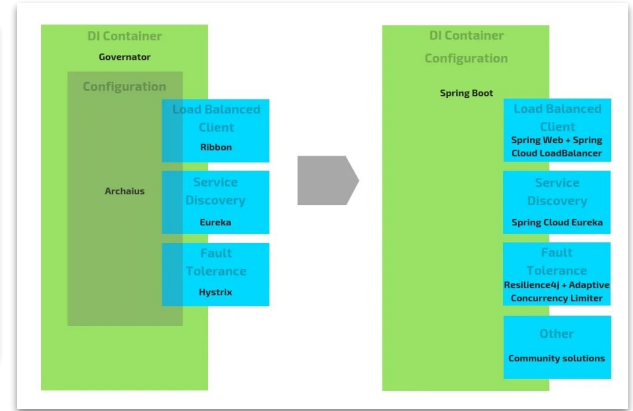
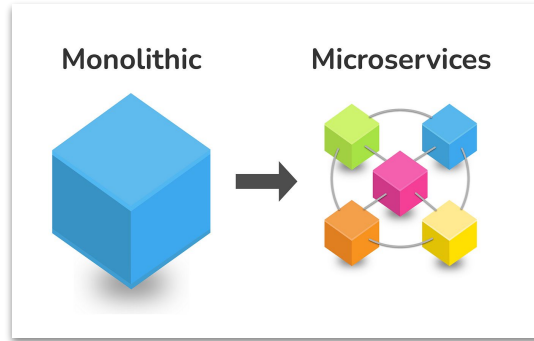
• LIVE

CHRISTMAS DAY

N

**Let's focus
on JVM backend
Landscape! How
did we get here?**

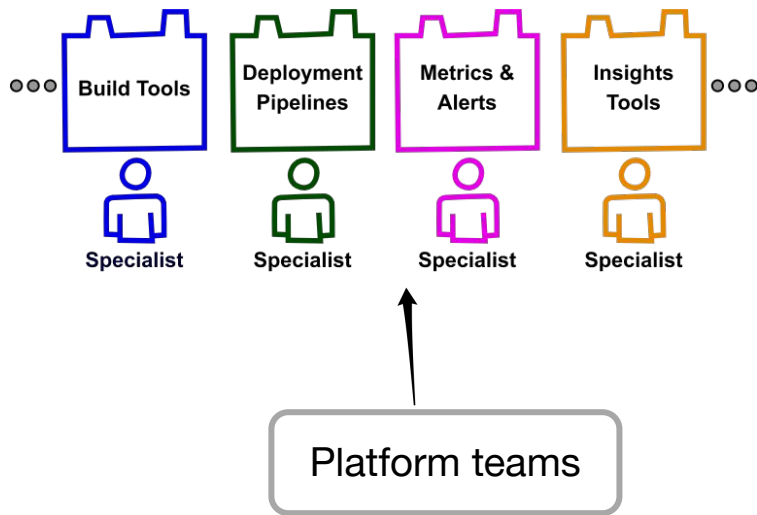
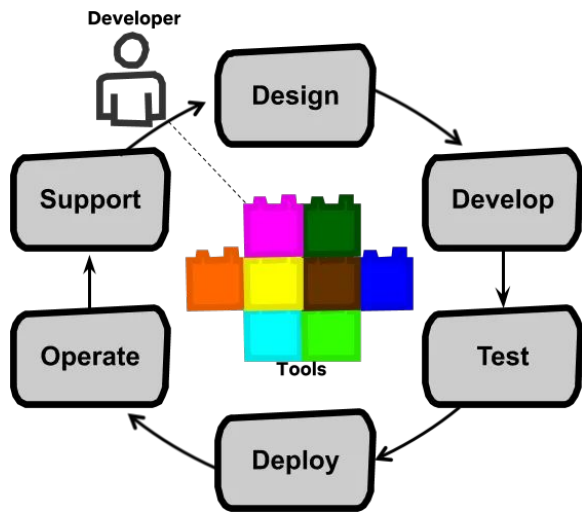
Architecture evolution



NETFLIX | OSS

Full cycle developers

(operate what you build)



We have the paved path



Image from https://media.wired.com/photos/5926a050cefa457b079b6b6/master/w_2400%2Cc_limit/Getty/images-166135659.jpg

JVM Backend landscape

- ~4k git repositories
- ~2.8k applications
- ~1.5k (internal) libraries



DEV/ELLOCITY

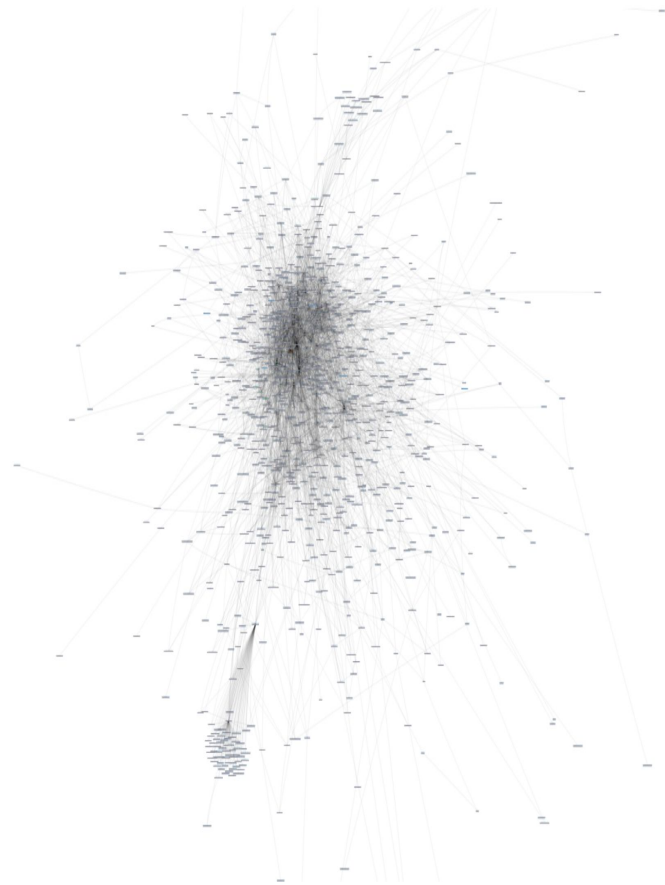
NETFLIX | OSS

HØ11ØW



EVcache

Eventually consistent distributed monolith



JVM and OSS libraries evolve faster

 **Spring Boot** helps you to create Spring-powered, production-grade applications and services with absolute minimum fuss. It takes an opinionated view of the Spring platform so that new and existing users can quickly get to the bits they need.

Release	Released	OSS support	Commercial Support	Latest
3.3	3 months and 3 weeks ago (23 May 2024)	Ends in 8 months (23 May 2025)	Ends in 1 year and 11 months (23 Aug 2026)	3.3.3 (22 Aug 2024)
3.2	9 months and 4 weeks ago (23 Nov 2023)	Ends in 2 months and 1 week (23 Nov 2024)	Ends in 1 year and 5 months (23 Feb 2026)	3.2.9 (22 Aug 2024)
3.1	1 year and 4 months ago (18 May 2023)	Ended 4 months ago (18 May 2024)	Ends in 11 months (18 Aug 2025)	3.1.12 (29 May 2024)
3.0	1 year and 9 months ago (24 Nov 2022)	Ended 9 months ago (24 Nov 2023)	Ends in 5 months (24 Feb 2025)	3.0.13 (29 Nov 2023)
2.7	2 years and 4 months ago (19 May 2022)	Ended 9 months ago (24 Nov 2023)	Ends in 11 months (24 Aug 2025)	2.7.18 (29 Nov 2023)
2.6	2 years and 10 months ago (17 Nov 2021)	Ended 1 year and 9 months ago (24 Nov 2022)	Ended 6 months and 3 weeks ago (24 Feb 2024)	2.6.15 (18 May 2023)

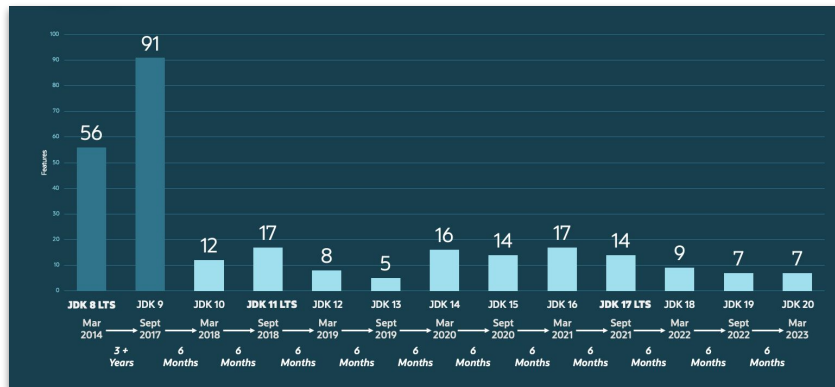


FIGURE 1.2. OPEN SOURCE NEW PROJECT GROWTH RATE OVER THE PAST 4 YEARS

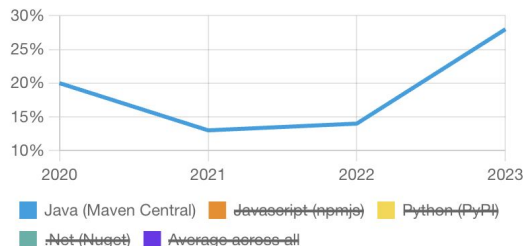
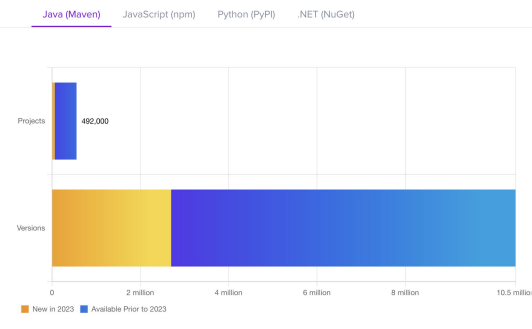


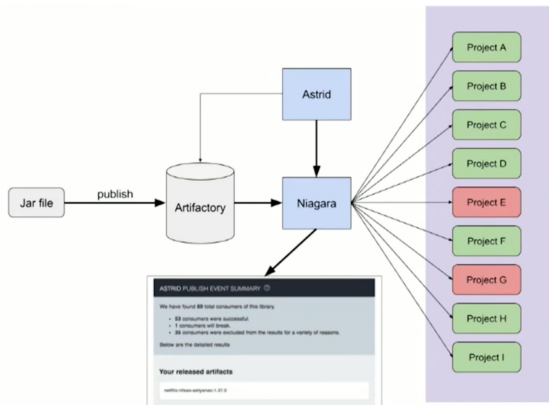
FIGURE 1.3. OPEN SOURCE PROJECTS AND VERSIONS GROWTH, 2023



**We had to invest
on delivering
change at scale**



The diagram illustrates the dependency resolution workflow. It starts with a 'Jar file' being published to an 'Artifact' repository. 'Astrid' is shown interacting with the 'Artifact' repository and 'Niagara'. 'Niagara' then resolves dependencies for a set of projects (Project A through Project I). The projects are listed in a vertical column on the right, with 'Project E' and 'Project G' highlighted in red, indicating they are the focus of the resolution process.



#dep-talk-2017

Managed Source Dependency Lock Update #250

 **Merged** **arcas** merged 1 commit into [corp:main](#) from [corp:msrc-DEPENDENCY_LOCK-NEBULA-for-main](#)  yesterday

 Conversation  **Commits**  Checks  Files changed



arcas  commented yesterday

Updated locked dependencies.

Detailed lock with dependency path based differences for dependency update debugging can be found in [http://go/nebula-path-aware-lock-diff/35f7560d-fd0d-4364-b873-86312ca22ca2](#)
[How to understand dependency path based differences](#)

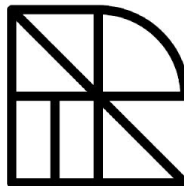
Re-generate this PR or manage automated Nebula updates at [go/enroll/corp/nebula-netflix-gradle-bootstrap](#)

root project

updated

- `netflix.nebula.resolutionrules:resolution-rules:1.312.1 -> 1.314.0` [\[changes\]](#)

 [jvm-ecosystem](#)
Created about 1 month ago by [@penezaos](#)

[illegible]

jvm/ecosystem-team / add-gradle-daemon-jvm-toolchain-properties-file-to-codeowners

Created about 1 month ago by perezakola · Updated about 1 month ago

[Edit](#) [Close](#)

[OPEN](#) 39% complete +371 -2,131 changesets 0 unpublished 0 draft 414 open 5 closed 280 merged 957 archived 22 others

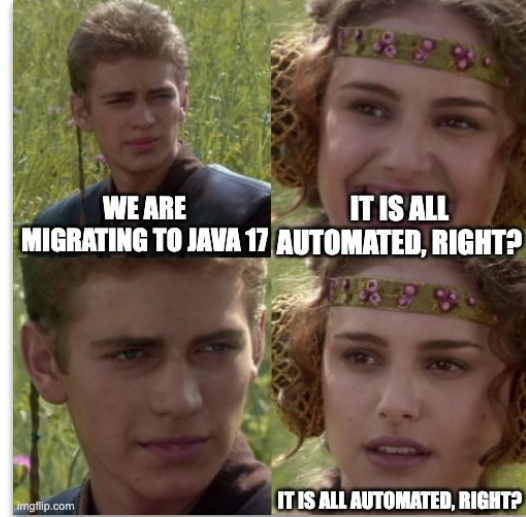
Add gradle daemon jvm toolchain properties file to CODEOWNERS

[Changesets](#) [Burndown chart](#) [Executions](#) [Archived](#) [Bulk operations](#)

Search your repository name Status Check state Review state

	STATUS	CHANGESET INFORMATION	CHECK STATE	REVIEW STATE	CHANGES
Merged	glinba-netflix:netflix:jvm-toolchain-properties-file-to-codeowners jbatch-changes-user:platform-change/netflix-add-gradle-daemon-jvm-properties-file-to-codeowners Last synced about 1 hours ago	Passed	Pending	+1 -1	
Open	glinba-netflix:netflix:jvm-toolchain-properties-file-to-codeowners jbatch-changes-user:platform-change/netflix-add-gradle-daemon-jvm-properties-file-to-codeowners Last synced about 4 hours ago	Passed	Pending	+1 -1	
Open	glinba-netflix:netflix:jvm-toolchain-properties-file-to-codeowners jbatch-changes-user:platform-change/netflix-add-gradle-daemon-jvm-properties-file-to-codeowners Last synced about 4 hours ago	Passed	Pending	+1 -1	

**And now everything
is perfect and we
can modernize our
stack without
engineers
interaction!**



**Turns out
delivering code
changes is hard!**



Making code changes at scale

N



Changing code

“

Changing code is hard.

Making code changes across tens of thousands of git repositories is even harder.

Netflix has been able to make progress to addressing this issue, but we need to take this further.

”



From a recent Netflix job posting

A response

“

What the heck are you doing that requires changing code in so many repositories? It doesn't sound hard so much as weird. The only scenario that makes sense is if you are changing third-party code. Maybe auto-generating and applying security patches? That could be interesting but the hard part would be getting permission to apply the patches.

”



A person's reply to a recent Netflix job posting

**Technology is not
the only aspect
for a successful
code change**



**Let's look at
a real world
example**



Updating application JDKs at scale

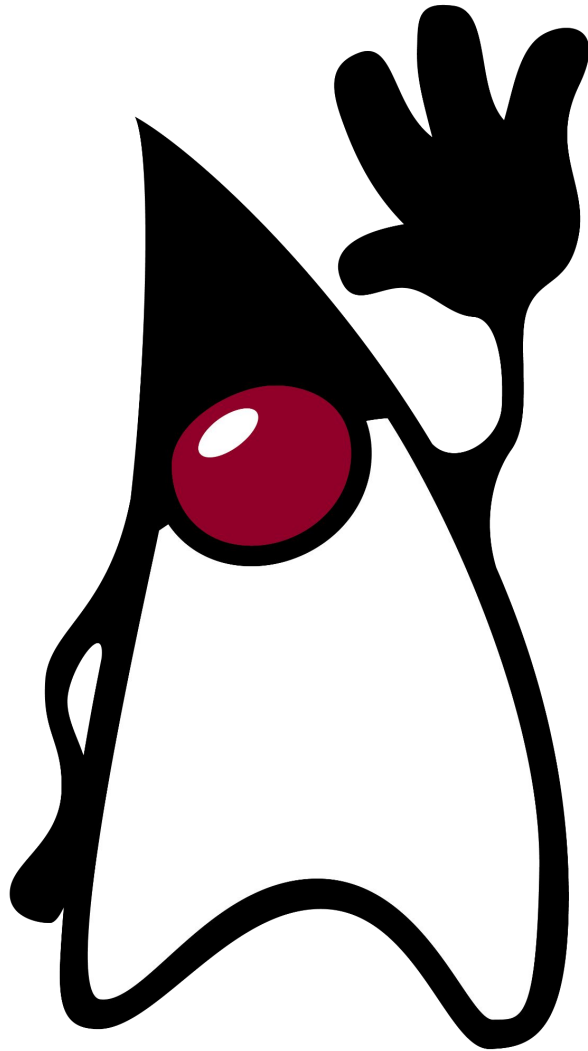


Image from <https://wiki.openjdk.org/display/duke/Gallery>

A very small code change

```
1 java {  
2     toolchain {  
3         languageVersion = JavaLanguageVersion.of(17)  
4     }  
5 }
```

to upgrade application runtime JDK

**And we have an
internal campaign
tool to plan and
follow up in this
migrations 🎉**

Campaign plan

PLANNING

1

Plan your work for this target

STEPS

1

Pull Request
Merge bulk generated PR

2

Canary
Canary your code

3

Deployment
Going live with JDK 17

I cannot or do not need to do this

Plan your migration

If you need more context for what is required to complete this effort, please click around the additional sections in the left-hand navigation. Or you can check out the [JDK 17 Upgrade for Applications landing page](#) to learn more about why the campaign exists.

Minimum required fields

Estimated Date Work Will Complete ⓘ

When do you believe the work will complete? If you fill this out we won't nag you until you approach this date. All times in PST.

This campaign is past due, its end date was Jul 31, 2024. Please plan to complete your target as soon as possible.

Assignee

The individual accountable for driving this campaign for this resource to completion, not necessarily executing all changes themselves

>

Additional fields

Update Plan

Campaign stages

Stages

1

Pull Request

Merge bulk generated PR

2

Canary

Canary your code

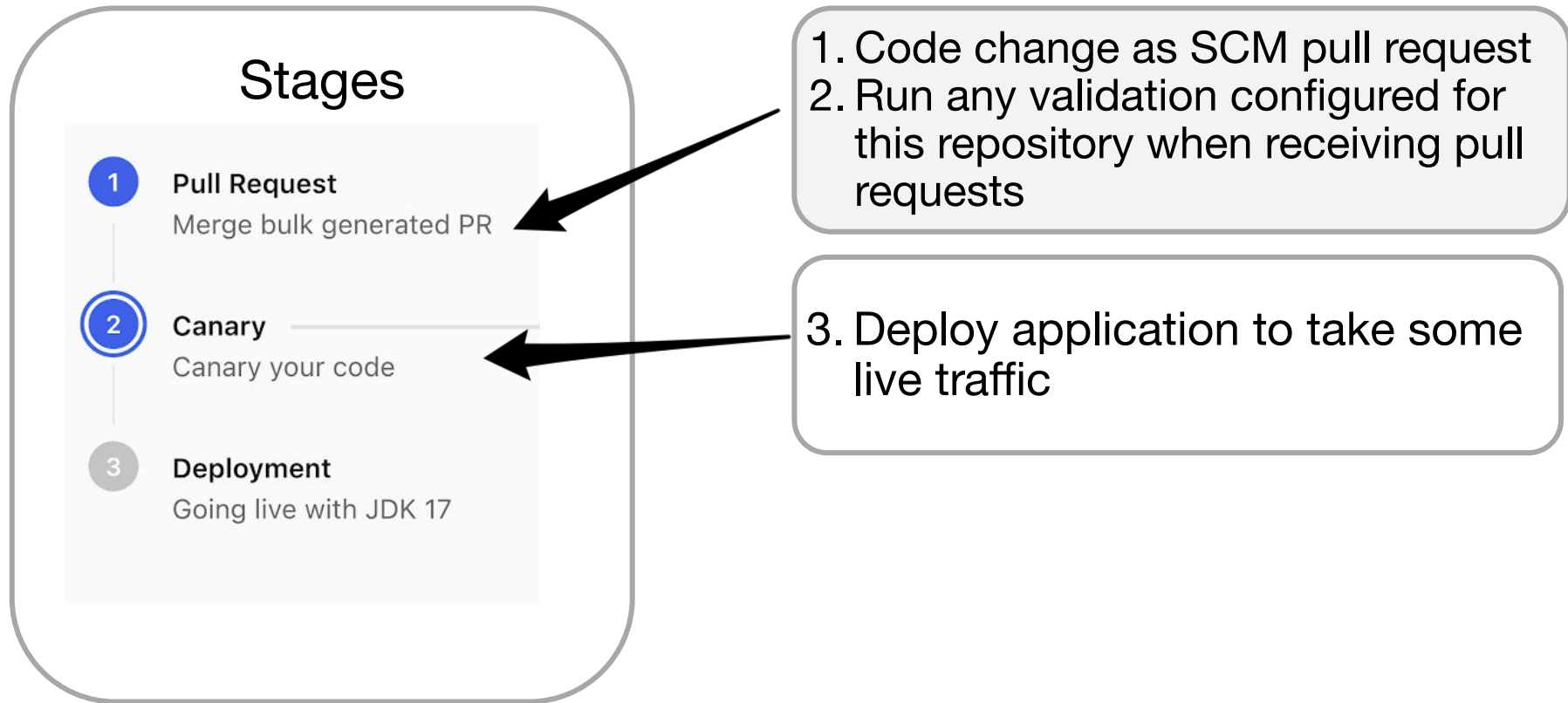
3

Deployment

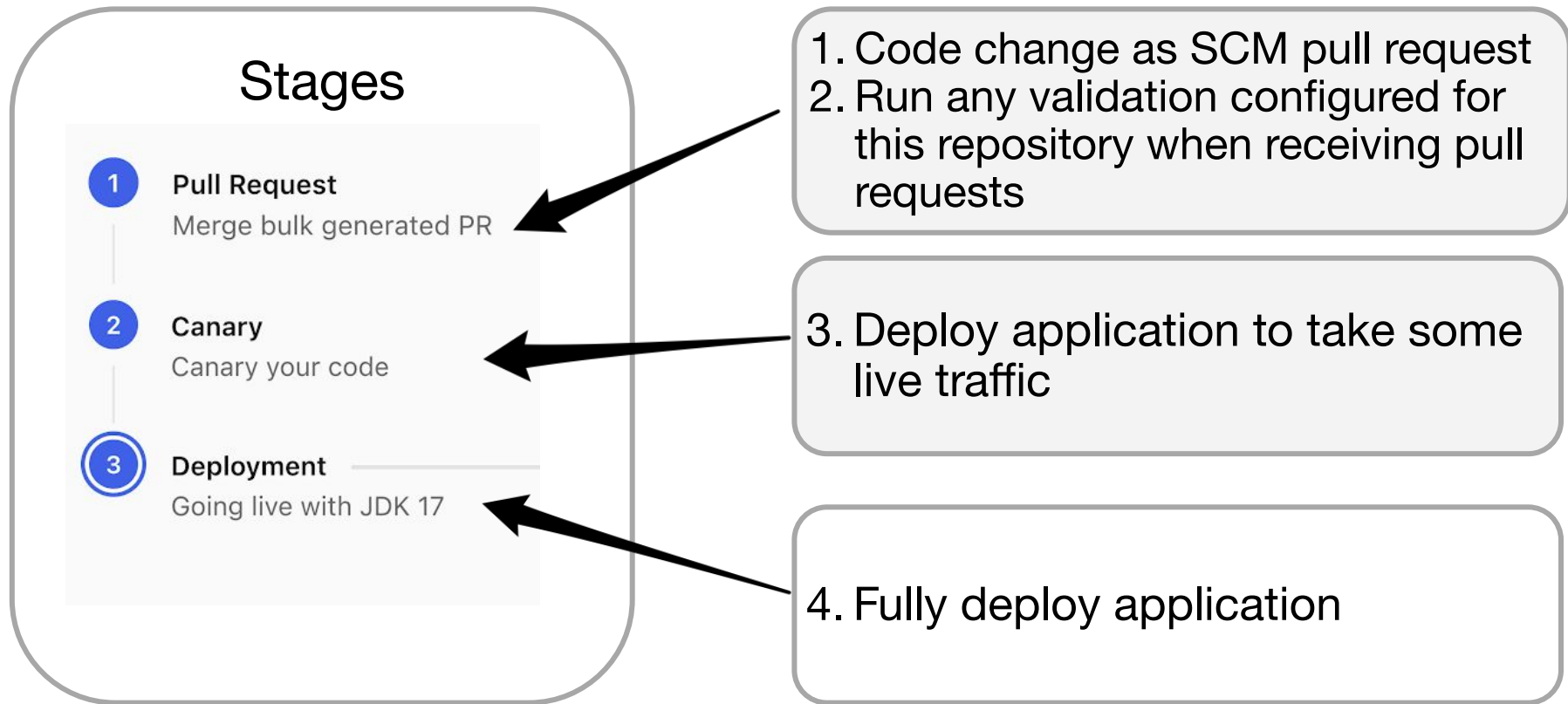
Going live with JDK 17

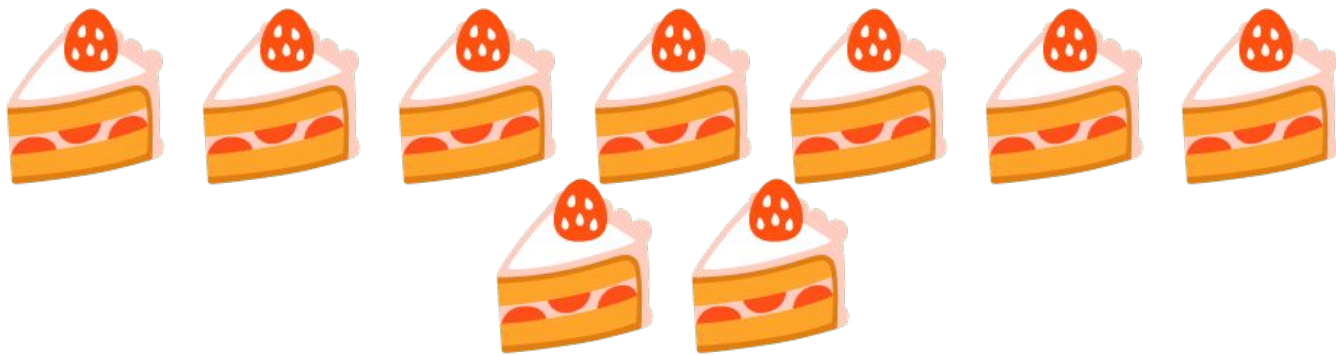
1. Code change as SCM pull request
2. Run any validation configured for this repository when receiving pull requests

Campaign stages



Campaign stages





**Very easy...
right?**



**Unfortunately,
IT IS NOT 🙄...
let's look at
some learnings**



Existing verification might no be sufficient

 Pinned by incident

 **incident** APP 2:27 PM

JDK17 upgrade preventing batch send for [REDACTED]

Problem: The upgrade to JDK17 is preventing the batch send process for [REDACTED].

Impact: Unknown

Causes: Unknown

Steps to resolve: We pinned to a previous version of [REDACTED] in order to proceed with the revert, then re-ran the segment.

 Page a team

 Set fields

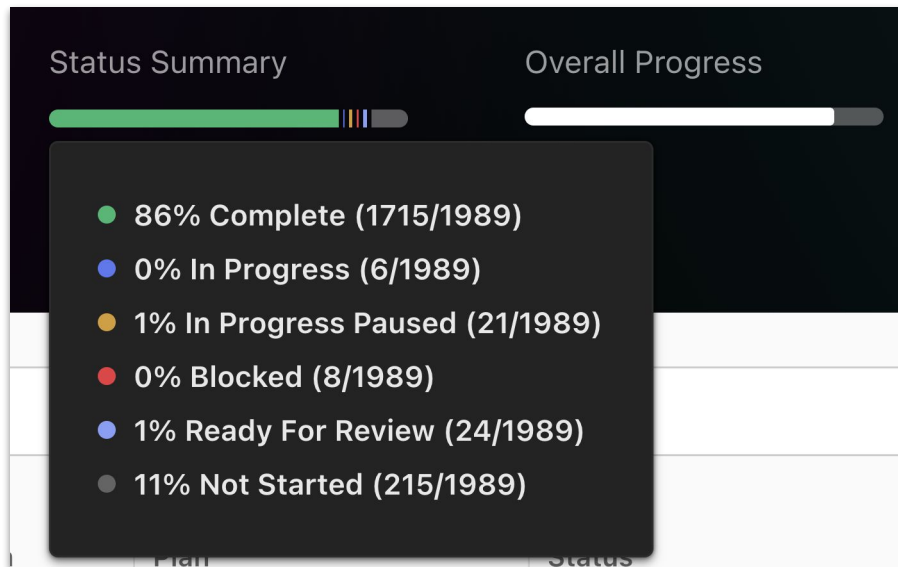
 View all commands

 Overview

And human-watched or human-implemented migrations take time



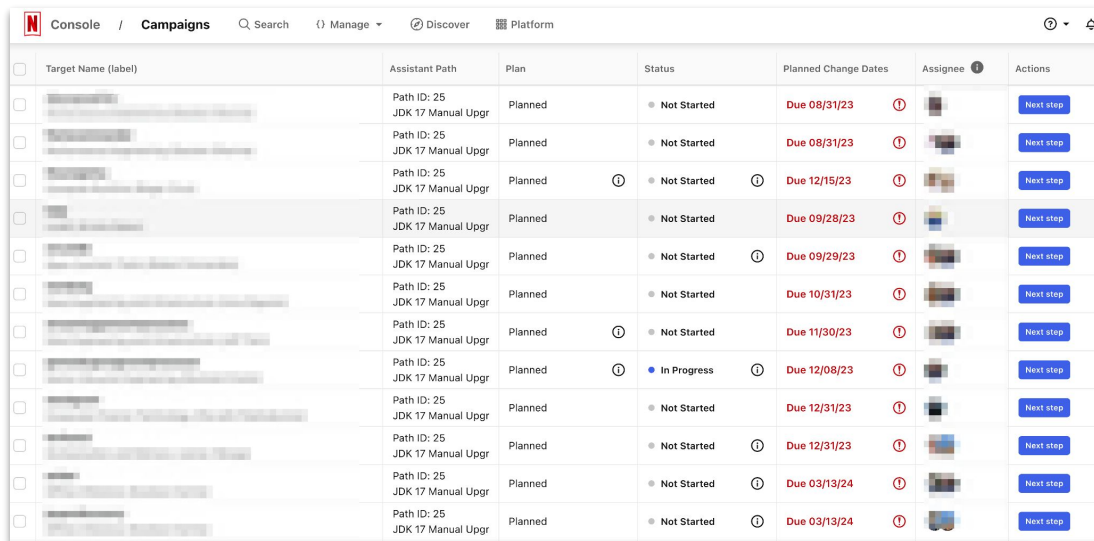
The campaign targets (size)



Thousands of targeted applications and each one of them has 1 or more running instances

Requires owner engagement

1. Teams have their own priorities
1. Providing automation is great but they still wanted to have control of the change
1. Requires scaling for providing great support and customer experience



☐	Target Name (label)	Assistant Path	Plan	Status	Planned Change Dates	Assignee	Actions
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 08/31/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 08/31/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 12/15/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 09/28/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 09/29/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 10/31/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 11/30/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● In Progress	Due 12/08/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 12/31/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 12/31/23	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 03/13/24	[Redacted]	Next step
☐	[Redacted]	Path ID: 25 JDK 17 Manual Upgr	Planned	● Not Started	Due 03/13/24	[Redacted]	Next step

**And this is just our
JDK 17
campaign...**

**What about
more and
simultaneous
campaigns?** 🤔



This could be the campaigns for an engineer

Campaigns: **Roberta Perez Alcala**

 Include Child Organizations

 Directly Owned

 Filter results



 Sorted by priority

Increasing Confidence and Frequency of Employment ⓘ

07/08/24 - 12/13/24 ● In Progress

1 Targets require action

[View Targets](#)



Team Progress

 4/5 completed (80%)

Action required by

 10/31/24

Increasing Self-Confidence ⓘ

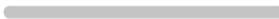
08/22/24 - 10/02/24 ● In Progress

3 Targets require action

[View Targets](#)



Team Progress

 0/3 completed (0%)

Action required by

 10/02/24

Build Confidence in Self ⓘ

01/22/24 - 12/31/24 ● In Progress

2 Targets require action

[View Targets](#)



Team Progress

 0/2 completed (0%)

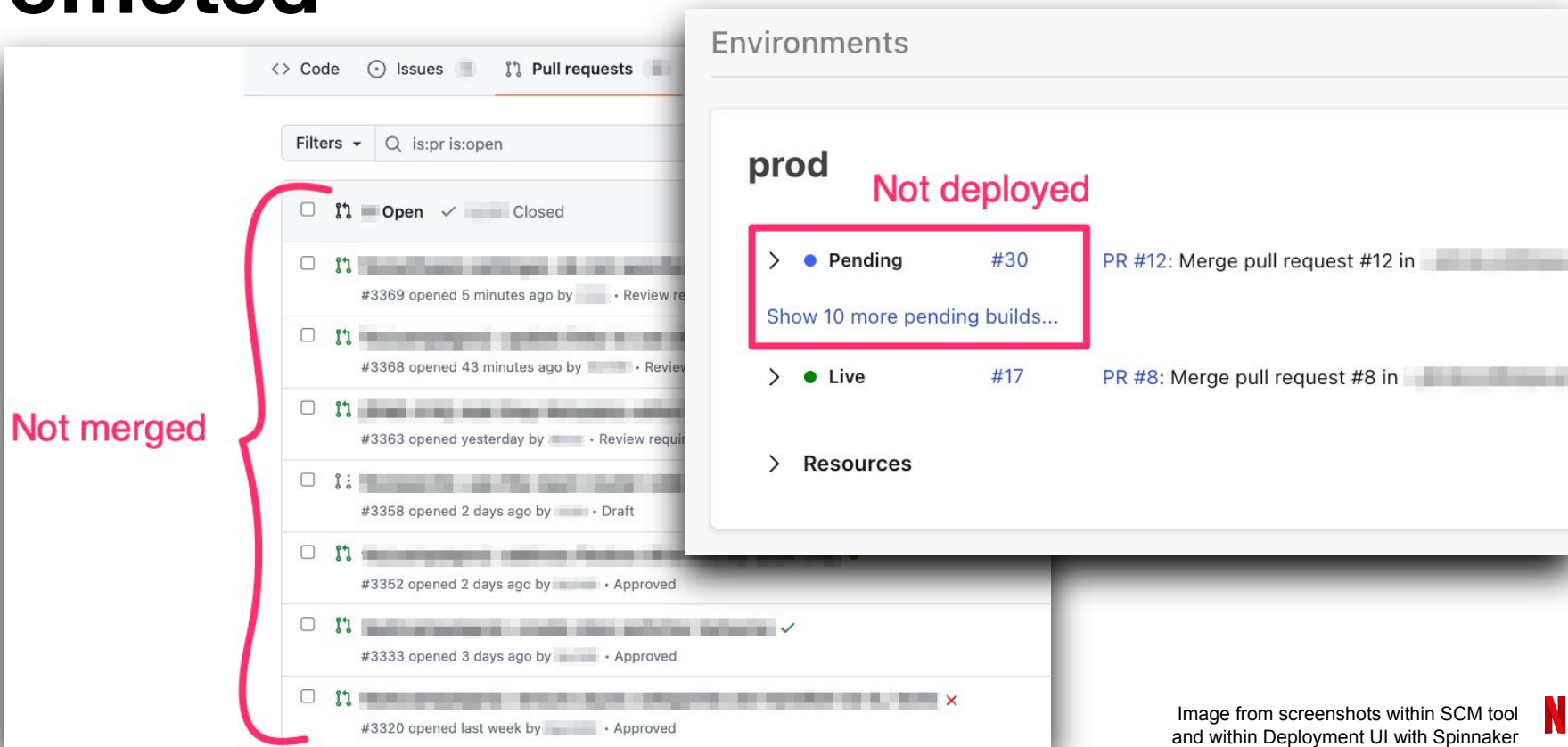
Action required by

 10/31/24

A screenshot of the GitHub Pull Requests interface. The top navigation bar includes links for Code, Issues, Pull requests (highlighted), Actions, Security, and Insights. Below the navigation bar, there's a search bar with the text "is:pr is:open". A red bracket on the left side of the page groups the first five pull request entries, with the text "Not merged" written in red next to it. The pull requests listed are:

- #3369: Open, Review required (5 minutes ago)
- #3368: Open, Review required (43 minutes ago)
- #3363: Open, Review required (yesterday)
- #3358: Draft (2 days ago)
- #3352: Approved (2 days ago)
- #3333: Approved (3 days ago)
- #3320: Approved (last week)

Observations: Deployments are not promoted



The image consists of two overlapping screenshots from a software development tool.

The left screenshot shows a 'Pull requests' tab. It has a search bar with 'is:pr is:open' and a 'Filters' dropdown. Below, there's a list of pull requests. A red bracket on the left side of the list, spanning the first four items, is labeled 'Not merged' in red text. The items are: #3369 (opened 5 minutes ago), #3368 (opened 43 minutes ago), #3363 (opened yesterday), and #3358 (opened 2 days ago). Below these are #3352 (opened 2 days ago, Approved), #3333 (opened 3 days ago, Approved), and #3320 (opened last week, Approved).

The right screenshot shows an 'Environments' tab. It displays a 'prod' environment. A red box highlights the 'Pending' build #30, with the text 'Not deployed' in red next to it. Below the box is a link 'Show 10 more pending builds...'. Other builds shown are #17 (Live) and #12 (Merge pull request #12 in ...). There is also a 'Resources' section.

Image from screenshots within SCM tool
and within Deployment UI with Spinnaker

Is that enough?



Is that enough?

**It is NOT
enough**



Observation

Although we provided changes at scale,

Our users were not fully taking advantage of this productivity boost.

We probably became good at creating and scheduling changes.

We were still missing something.

What was missing?

N



Why are people not merging and deploying changes?

The image consists of two overlapping screenshots. The background screenshot shows a GitHub 'Pull requests' page with a list of open pull requests. A red bracket on the left side of the list is labeled 'Not merged'. The foreground screenshot shows the 'Environments' section of a Spinnaker deployment tool. Under the 'prod' environment, which is labeled 'Not deployed', there is a 'Pending' build (#30) highlighted with a red box. The text 'PR #12: Merge pull request #12 in' is visible next to the pending build. Below the pending build, there is a link 'Show 10 more pending builds...'. Further down, there is a 'Live' build (#17) with the text 'PR #8: Merge pull request #8 in'.

Not merged

Environments

prod Not deployed

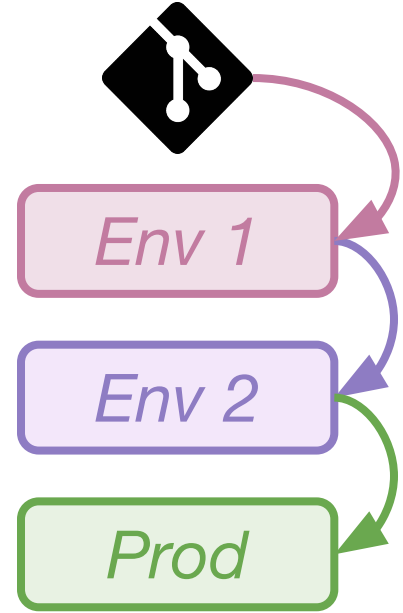
> Pending #30 PR #12: Merge pull request #12 in

Show 10 more pending builds...

> Live #17 PR #8: Merge pull request #8 in

> Resources

Can we get to zero-touch deploys for these campaigns?



We sent a survey



Image from <https://lordicon.com/icons/wired/outline/967-questionnaire>

Confidence questions

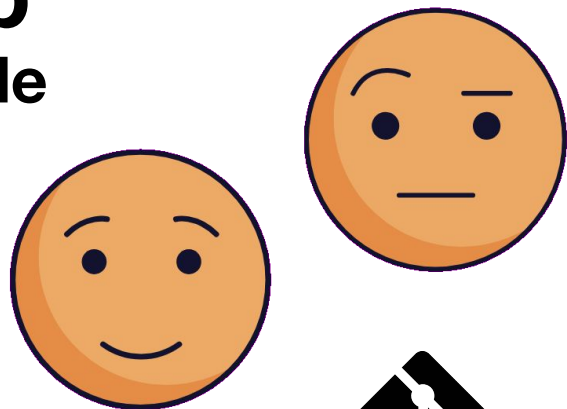
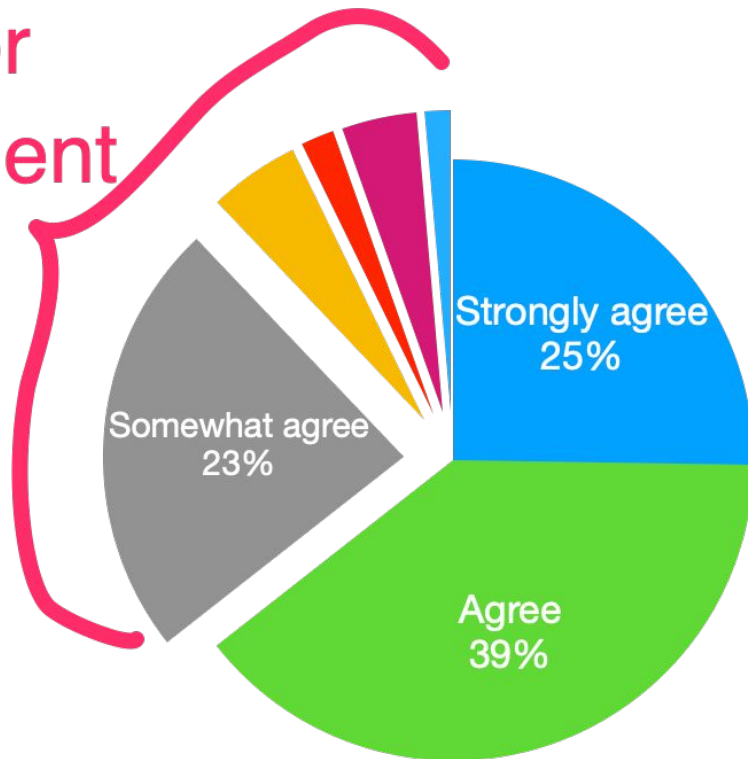
Automatic commit to deployment workflows with the following types of changes:

1. Application code changes made by your own team?
1. Application code changes made from engineers outside your team?
1. Changes outside of the application code?



Confidence Rating per app for changes outside of application code

Targets for
improvement



Env 1

Env 2

Prod

N

Deployment Frequency

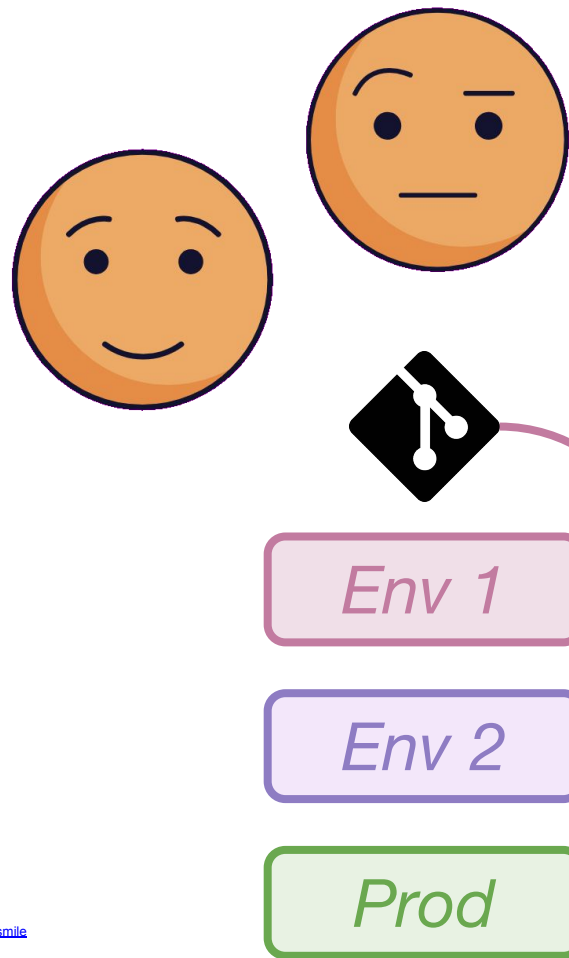
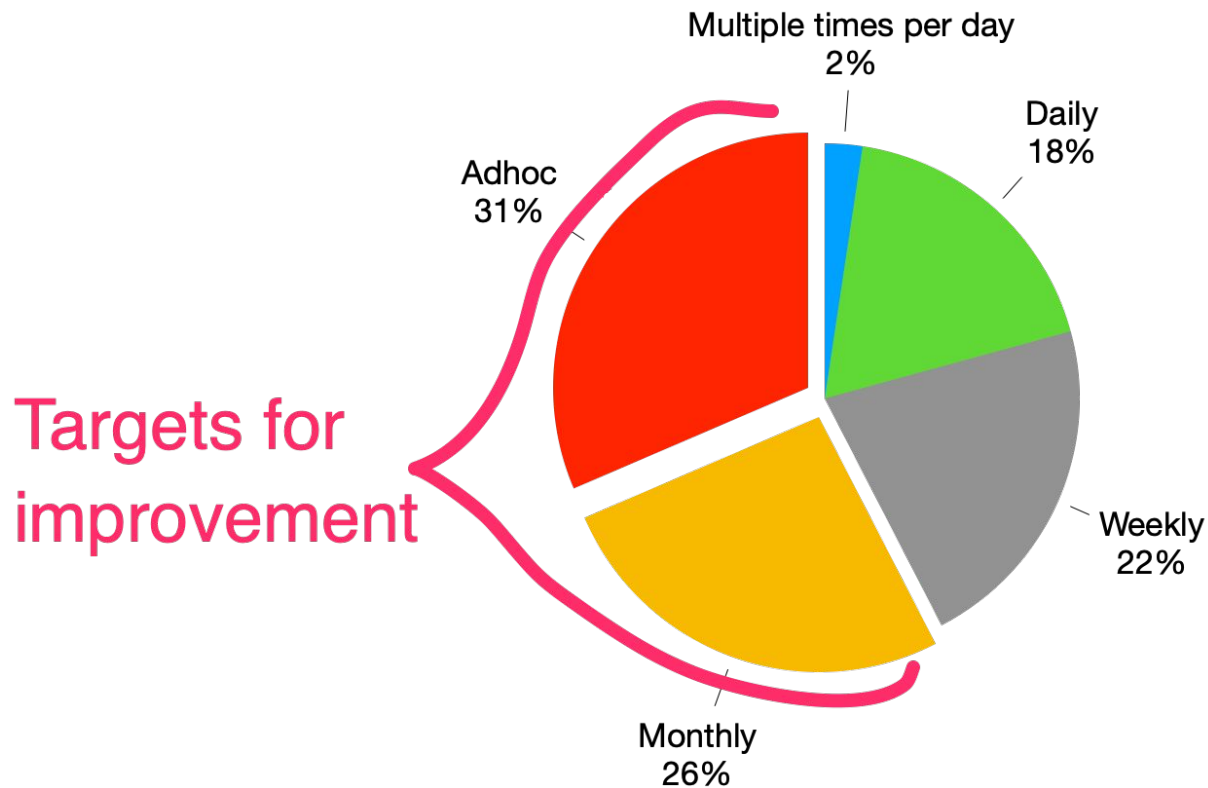
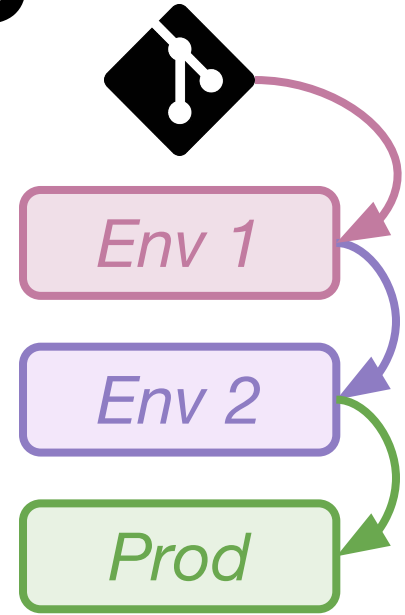


Image from internal data of applicable applications

Images from <https://back.gil-scm.com/downloads/logos>, from <https://lordicon.com/icons/wired/lineal/2340-thinking-skeptic-person-avatar>, and from <https://lordicon.com/icons/wired/lineal/261-emoji-smile>

What is preventing people from achieving these outcomes?



Maybe the paved path had some gravel sections





Image generated with ChatGPT

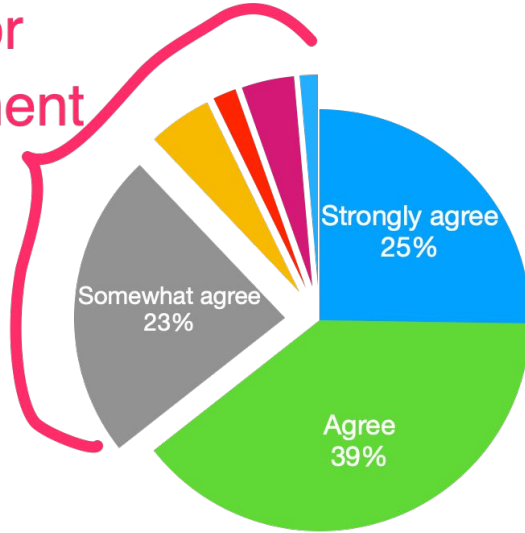
Virtual team

23 passionate individuals from:

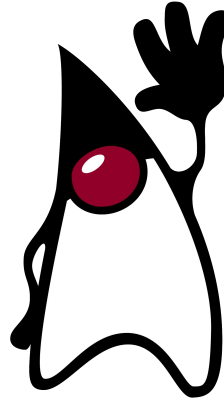
- Platform Engineering teams
- Product Engineering teams
- Product Management
- Subject Matter Experts

Scope

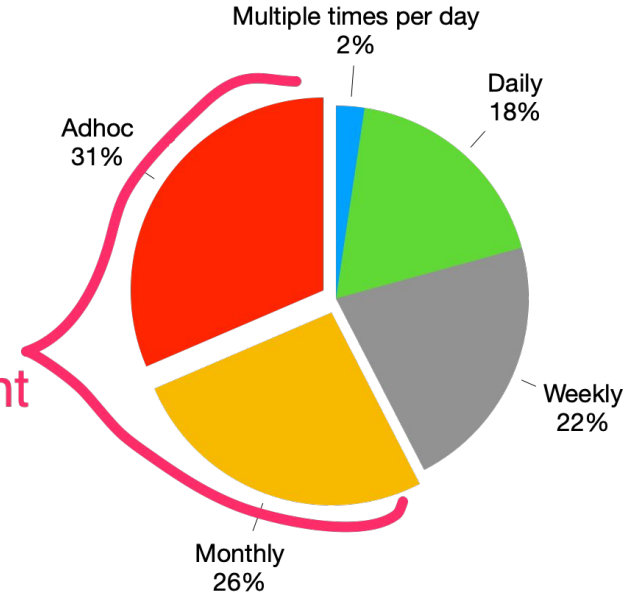
Targets for improvement



Confidence



Targets for improvement



Frequency

Virtual team focus areas

- Clean up abandoned services
- Dependency updates
- Confidence by default
- Frequency
- Delivery
- Monitoring
- Validations
- And more!



Engineering requirements



Image generated with ChatGPT

How long did this whole campaign take for the virtual team?

*Some folks worked on this one quarter.
Some folks are still working on this 3
quarters later.*



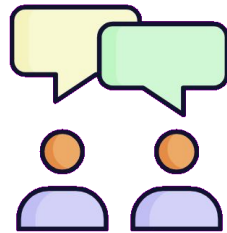
N

**We validated
these
requirements**



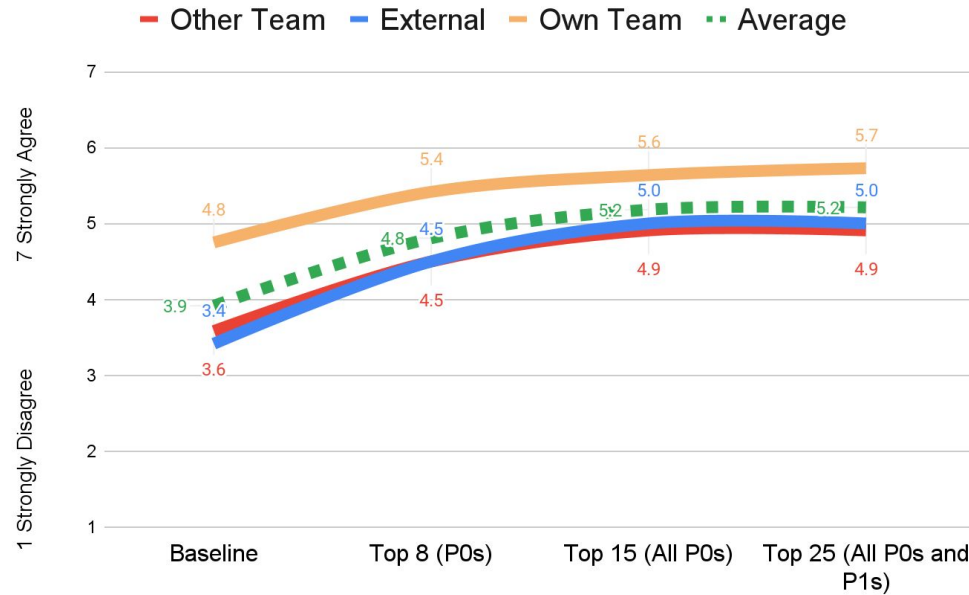
Validation Guided Interviews

**Imagine your same
production application *now*
conformed to all of those
requirements**



Requirements Validation

Average Confidence Scores



↑ 23% *average* change potential

↑ 46% *external to the application* change potential



What were the requirements?



How did we make them actionable?



Engineering requirements

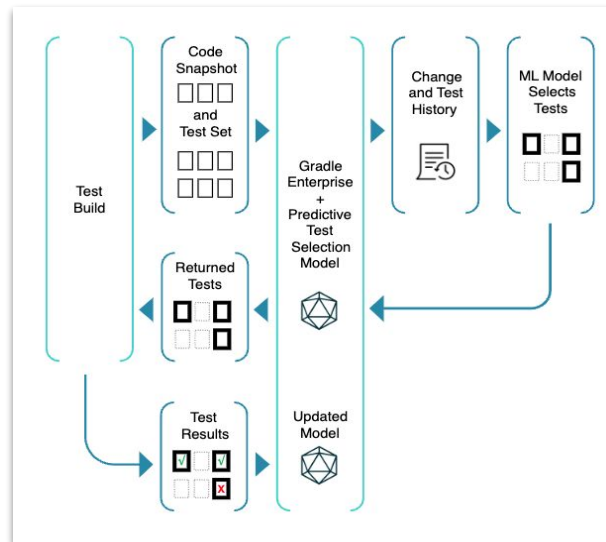
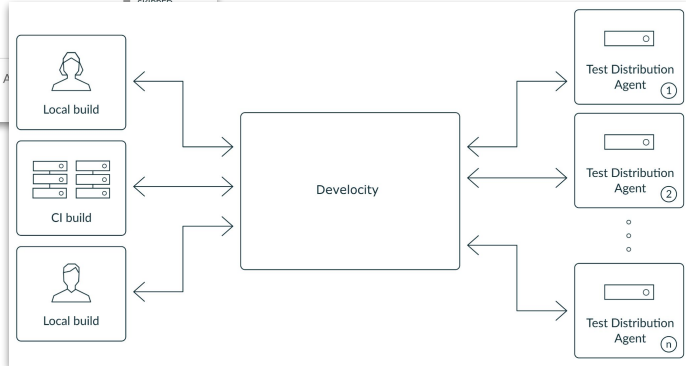
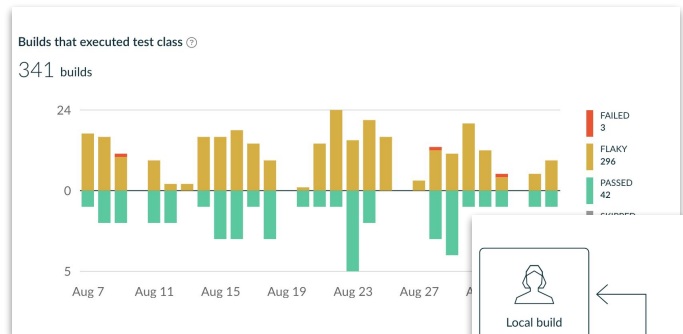
Require improvements in different categories:

- Application metadata
- Build and Inner Dev Loop Testing infrastructure
- Delivery infrastructure
- Security
- Source Code quality
- Resiliency and Scalability Testing
- Observability
- And more!



Engineering requirements

Java Ecosystem was more mature because of previous investments in the testing experience!





📍 San Francisco, CA
📅 September 20-21

How Improving the Testing Experience Goes Beyond Quality: A Dev Productivity Point of View

Roberto Perez Alcolea & Aubrey Chipman

Developer Productivity Team
Netflix



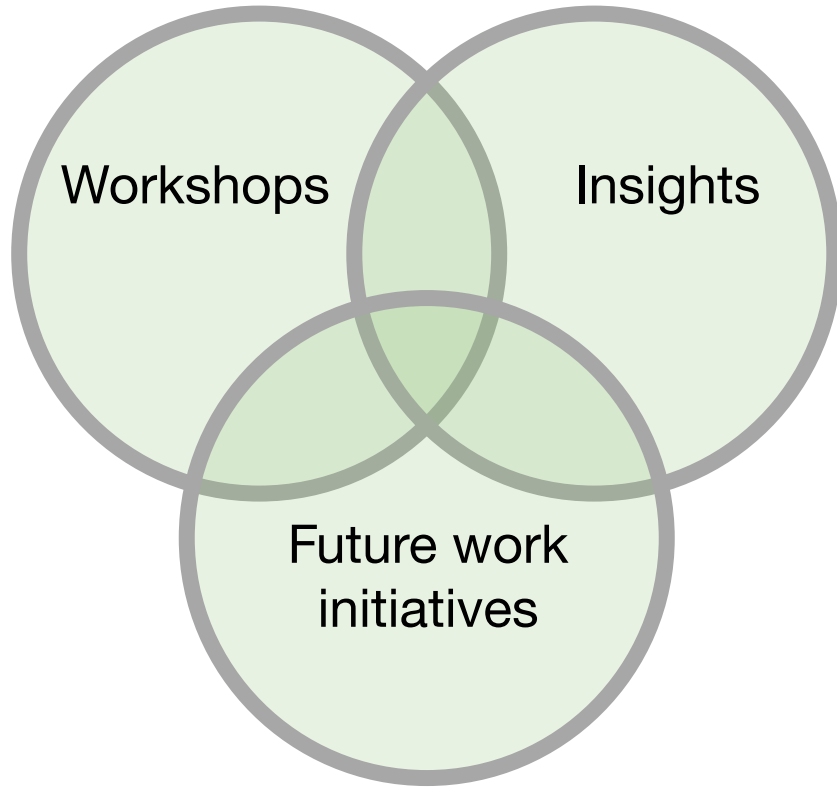
Major areas of investment

- Testing developer experience
- Language agnostic tooling
- Language maturity models
- Continuous Integration
- Production Always Ready initiative
- Tech debt as innovation



Key focus area:
The importance of
adapting to changes

Engineering investments



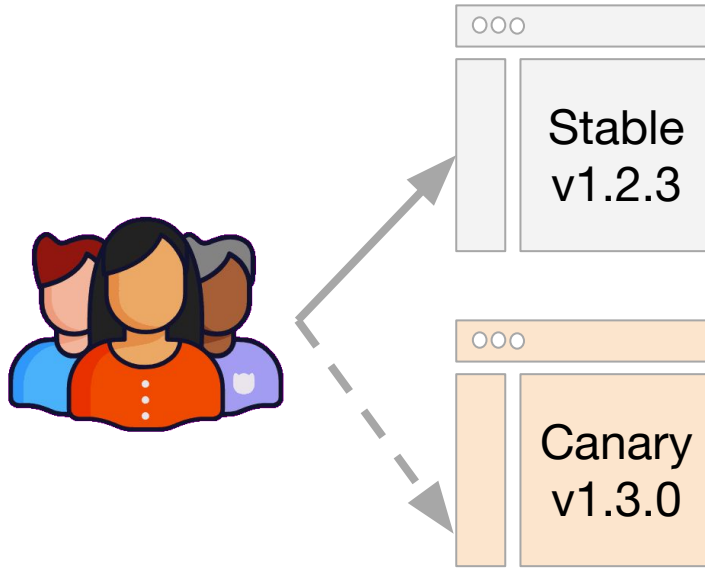
Shifting the company culture

Workshops

Testing



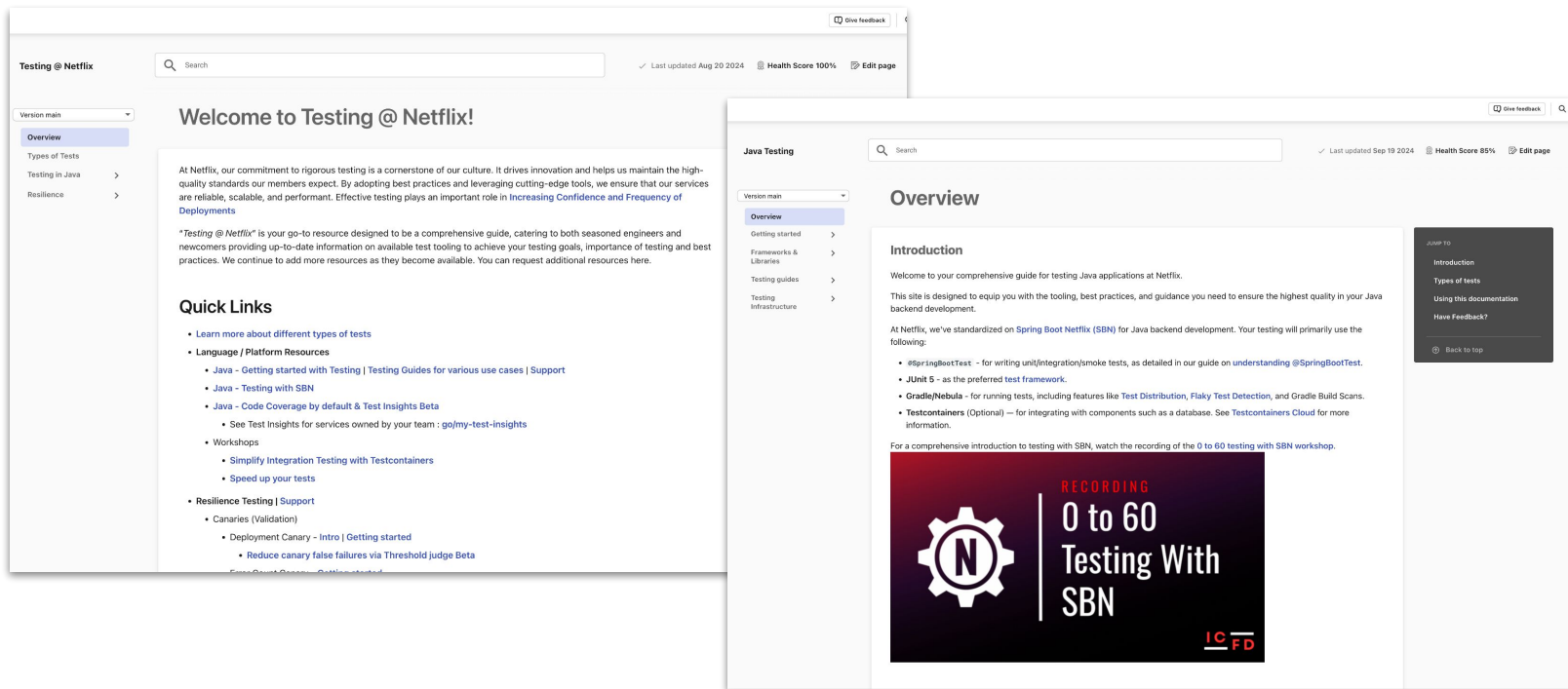
Auto-deployment configuration



Monitoring



Documentation



Improving documentation increased our CSAT considerably!



Testing insights

69% Code Coverage

↑ +38%



[Documentation](#) [Code Coverage Report](#)

93% Test Success Rate

↓ -7%



[Documentation](#) [Unstable Tests Report](#)

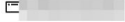
12m 12s Test Execution Time

↑ +5345%



[Documentation](#) [Slowest Tests Report](#)

Testing Insights

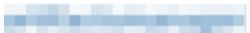
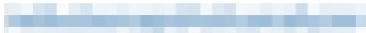
Test Insights Beta									
Search names		Lifecycle state		All Software Lifecycles		Directly Owned Resources		All Resources	
Name		Code coverage	90-d... Δ	Test success rate	90-day Δ	Test execution time	90-day Δ	# of tests	90-day Δ
		15%	$\uparrow +5\%$	100%	$\pm 0\%$	43s	$\uparrow +15\%$	2	$\pm 0\%$
		90%	$\pm 0\%$	100%	$\pm 0\%$	3m 28s	$\uparrow +53\%$	377	$\uparrow +2\%$
		79%	$\pm 0\%$	100%	$\pm 0\%$	3m 48s	$\uparrow +12\%$	9	$\pm 0\%$
		27%	$\pm 0\%$	100%	$\pm 0\%$	37s	$\pm 0\%$	1	$\pm 0\%$
		83%	$\pm 0\%$	100%	$\pm 0\%$	1m 23s	$\uparrow +5\%$	76	$\pm 0\%$
		22%	$\pm 0\%$	100%	$\pm 0\%$	19s	$\downarrow -13\%$	3	$\pm 0\%$
		90%	$\pm 0\%$	100%	$\pm 0\%$	4m 2s	$\downarrow -5\%$	384	$\uparrow +2\%$
		85%	$\pm 0\%$	100%	$\pm 0\%$	2m 28s	$\uparrow +7\%$	155	$\pm 0\%$
		84%	$\pm 0\%$	100%	$\pm 0\%$	1m 52s	$\uparrow +9\%$	92	$\pm 0\%$
		42%	$\pm 0\%$	100%	$\pm 0\%$	47s	$\uparrow +4\%$	4	$\pm 0\%$
		87%	$\pm 0\%$	100%	$\pm 0\%$	1m 56s	$\uparrow +1\%$	51	$\pm 0\%$
		57%	$\pm 0\%$	100%	$\pm 0\%$	9m 49s	$\downarrow -1\%$	149	$\pm 0\%$
		87%	$\downarrow -3\%$	100%	$\pm 0\%$	1m 43s	$\uparrow +11\%$	96	$\uparrow +14\%$
viewing 13 software components									

Delivery notifications

  APP 9:30 AM

Deployment Frequency Summary for **10 applications** (

 —  — 8 days ago (2 pending artifacts)

 —  |  |  |
 |  |  |  |
 | 

 — Deployed to production within last 14 day(s).

 — Deployed to production within last 7 day(s).

 **3 replies** Last reply today at 9:31 AM

Adopting new practices and tools



Our strategy

- Instrumentation for free!
- Don't reinvent the wheel
- Do more for customers (managed)
- Heavy focus on cultural change, learning and education
- Don't pursue major architectural changes
- Leverage our campaign tooling to increase confidence and frequency of deployments



The ICFD Campaign

Console / Campaigns

Search

Manage

Discover

Platform

← All campaigns

Increasing Confidence and Frequency of Deployments

Status

Priority

Duration

Contact

Status Summary

Overall Progress

In progress

High

07/08/24 - 12/13/24

icfd-increasing-confidence-and-freq-of-deploys

Details

Targets

Context

A key engineering goal for 2024 is to increase the **confidence** and **frequency** of commit → prod delivery workflows for spinnaker based services with the goal of unlocking the ability to drive more fully automated changes with less effort from teams. As such, [Increasing Confidence and Frequency of Deployments](#) is focused on achieving the following 2 outcomes:

1. Deployment frequency of at least weekly (i.e. 1 deploy per week)

2. Confidence scores of AGREE or STRONGLY AGREE regarding confidence in accepting changes that sit outside main application code (dependencies, baseOS, updates, sidecars, etc.)

Completion criteria

This campaign assistant will guide you through recommended actions to improve your application's deployment posture. Please implement the ones that make sense for your service and your specific use case.

The steps are divided into several sections, each covering a relevant aspect of the SDLC.

Each section starts with an overview of what we are asking for and why – and then proceeds to steps with actions you should review, address, and acknowledge when you've completed them.

For some use cases, you may need to invest in additional protections not covered by this guide -- please do what is necessary and don't feel limited by this list.

Work involved

Applications must achieve a confidence score when deploying external dependencies of AGREE or STRONGLY AGREE AND deploy at least WEEKLY. Confidence scores can be updated in the campaign itself (see the "Confidence" step of the campaign) or in Console.

NOTE: The system will automatically scan for this criteria to be met and mark your target as COMPLETE within 4 hours of detection.

Expected benefits

About

Priority

Targets owned by your team

Launch Date

Plans Due Date

Completion Date

Campaign Owner

Campaign Team

Resources

High

5

Jul 8, 2024

Aug 15, 2024

Dec 13, 2024

[Increasing Confidence and Frequency of Deployments](#)

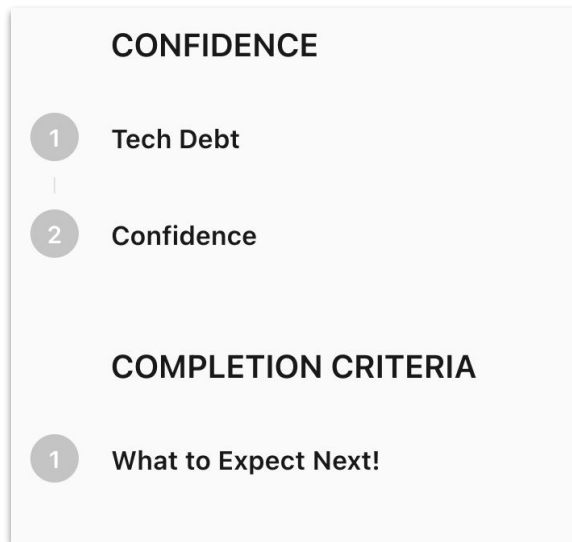
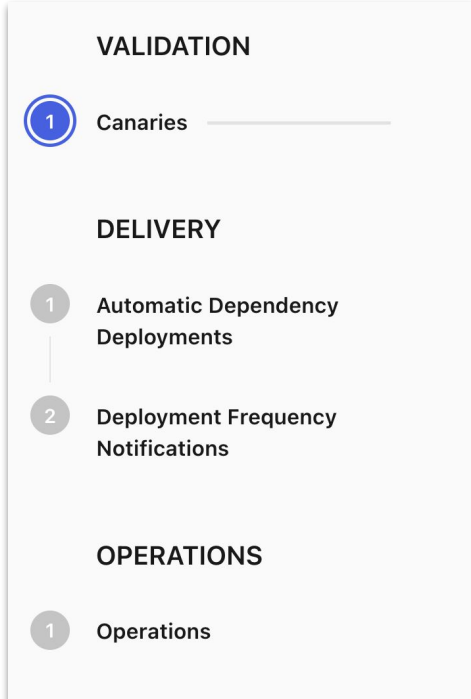
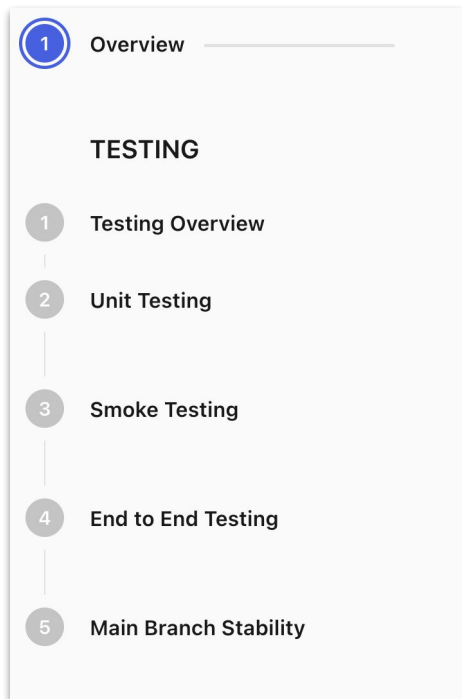
[Historical Burndown](#)

[Status](#)

[Progress Analytics](#)

Image from internal tool for campaign lifecycle

The ICFD Campaign



Guiding engineers through potential improvements
across a variety of areas

The ICFD Campaign

As the final step of the campaign, please update your confidence score(s)!

I am confident accepting changes that sit outside main application code (dependencies, baseOS, updates, sidecars, etc.)



6 - AGREE

I am confident accepting changes to my code made by my own team



7 - STRONGLY AGREE

I am confident accepting changes to my code from engineers outside my team



6 - AGREE

Asking them to self-score after their work and provide insight into tech debt

The ICFD Campaign

☐ **Architecture Debt**

This is when the software's architecture does not support current or future requirements well. Architecture Debt exists in tightly or improperly coupled systems, and areas with fragmented or unreconciled duplication of capability

☐ **Maintenance Debt**

This debt indicates difficulty in maintaining, updating, and generally working with a piece of software. The complexity of code, configuration, and tooling slows the ability to understand and modify the system. This debt stems from failing to keep software best practices up to date such as not deprecating code/unused code artifacts or not performing migrations.

☐ **Deployability Debt**

Software that is difficult or not capable of being updated or configured suffers from deployability debt. The software itself may be in good standing, but there are barriers to deploying changes to production. These could stem from manual or complex steps in the pipeline to not having a deployment route to the client (e.g. old devices) and prevent the ability to experiment or make critical fixes.

☐ **Operational Debt**

This debt increases the difficulty of supporting, triaging, and operating a piece of software. Deferred work on testing, observability, resiliency, and availability as well as the processes to track issues and drive improvements to the software can be considered operational debt.

☐ **Security Debt**

Security Debt accumulates when known security issues or vulnerabilities are not addressed promptly, or when decisions are made that compromise the security posture of the system. This includes using outdated and vulnerable software, not adhering to the security paved road or secure coding practices, and failing to consider security as an integral part of SDLC.

☐ **Extensibility Debt**

The system was purpose-built and may be well-architected, but is no longer the right solution for changing business or product use cases. This debt refers to limitations in the system's ability to handle new features or variations in the inputs to the system based on those architectural choices.

☐ **Knowledge Debt**

Knowledge debt is the gap in understanding or documentation about the system among the development team. This type of debt often arises when team members leave, when there's poor documentation, when system ownership is transferred, or when new technology is adopted without sufficient training and mutual understanding.

☐ **Compound Debt**

Technical debt in one system often prevents technical debt from another system from being paid down. The impact of the debt and the work required to reduce the debt may reside in different teams, creating an imbalance of impact and prioritization. When debt is transitively prevented from being resolved, the debt is compounding and the impact of the debt has a far larger blast radius.

Save Tech Debts

Focusing on these Tech Debt types

- Architecture
- Maintenance
- Deployability
- Operational
- Security
- Extensibility
- Knowledge
- Compound

Provide insight into tech debt

Our progress

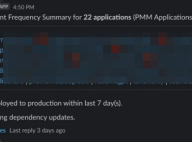
Engineers are excited!

Jul 18th at 4:44 PM

Hi team! Wanted to give some positive feedback 🙌 The Deployment Frequency Notification has been a great help for our team. In just a few weeks, we were able to get all our production BE applications deploying in the last 7 days (see the before and after pics of our notification). It has helped us avoid forgetting about locked environments in Spinnaker, and it generally feels better when we go to deploy a change and not see 20+ msrc pending changes.

The ability to suppress the notification if everything is green allowed us to point the notification directly at our support channel (so that oncall only gets pinged if there is action to take), so kudos to that feature in particular.

2 files ▾



👏 12 🙌 2 🗨️

9:44 AM

I am aware that often when I get asked for feedback on something, I will point out the one thing that's off. So to counter my point above, 🤖 the insights look great! I'm looking at the dashboards for 4 apps that my team supports (1 2 3 4) and these are great high level testing insights. Already interesting to see how things vary for different services. Our people will love it as well, I'm sure 😊

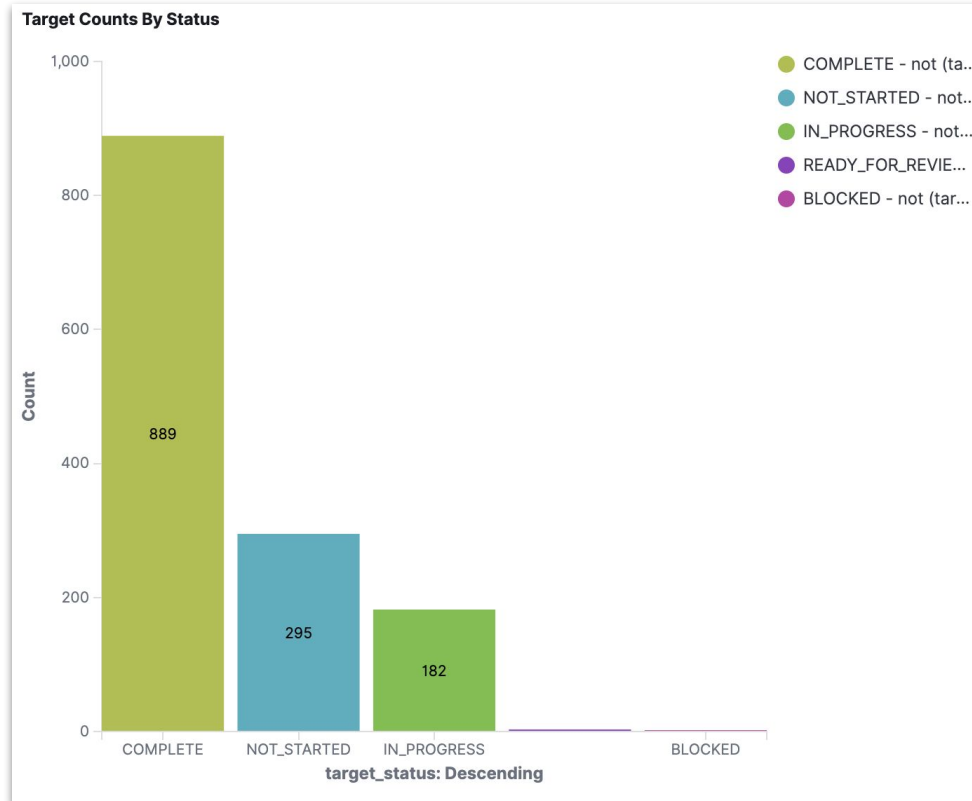
❤️ 7 🙌 6 🗨️

May 22nd at 9:00 AM

A shout out to [redacted]  !! The new dependency update notification is working like charm. The slack notification was discussed during our backend sync where we decided to kill 3 apps, turn on automatic deploys on 1 and actually fix the long standing build failure on the other 🙌🙌

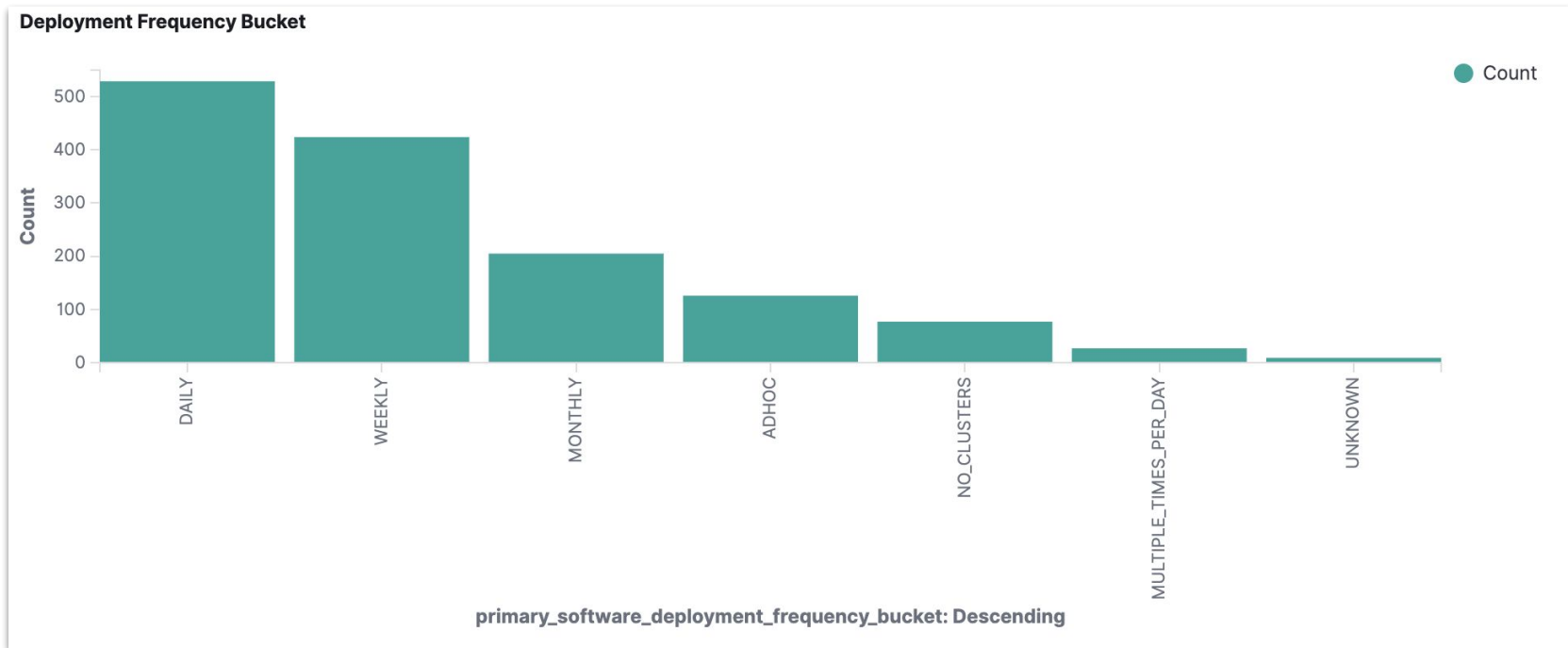
👤 8 🙌 8 🗨️ 3 🦖 3 🗨️

The ICFD Campaign



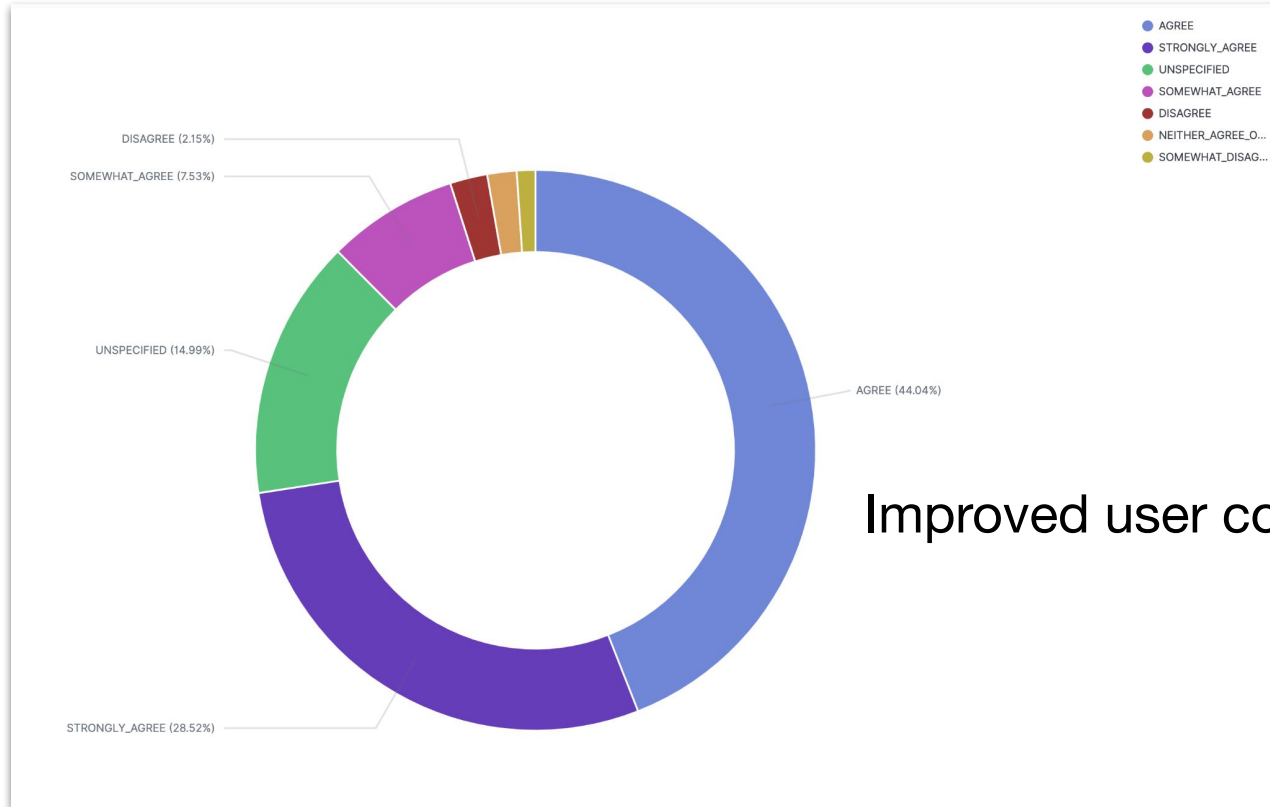
Target applications by status

The ICFD Campaign



Deployment frequency configuration

The ICFD Campaign

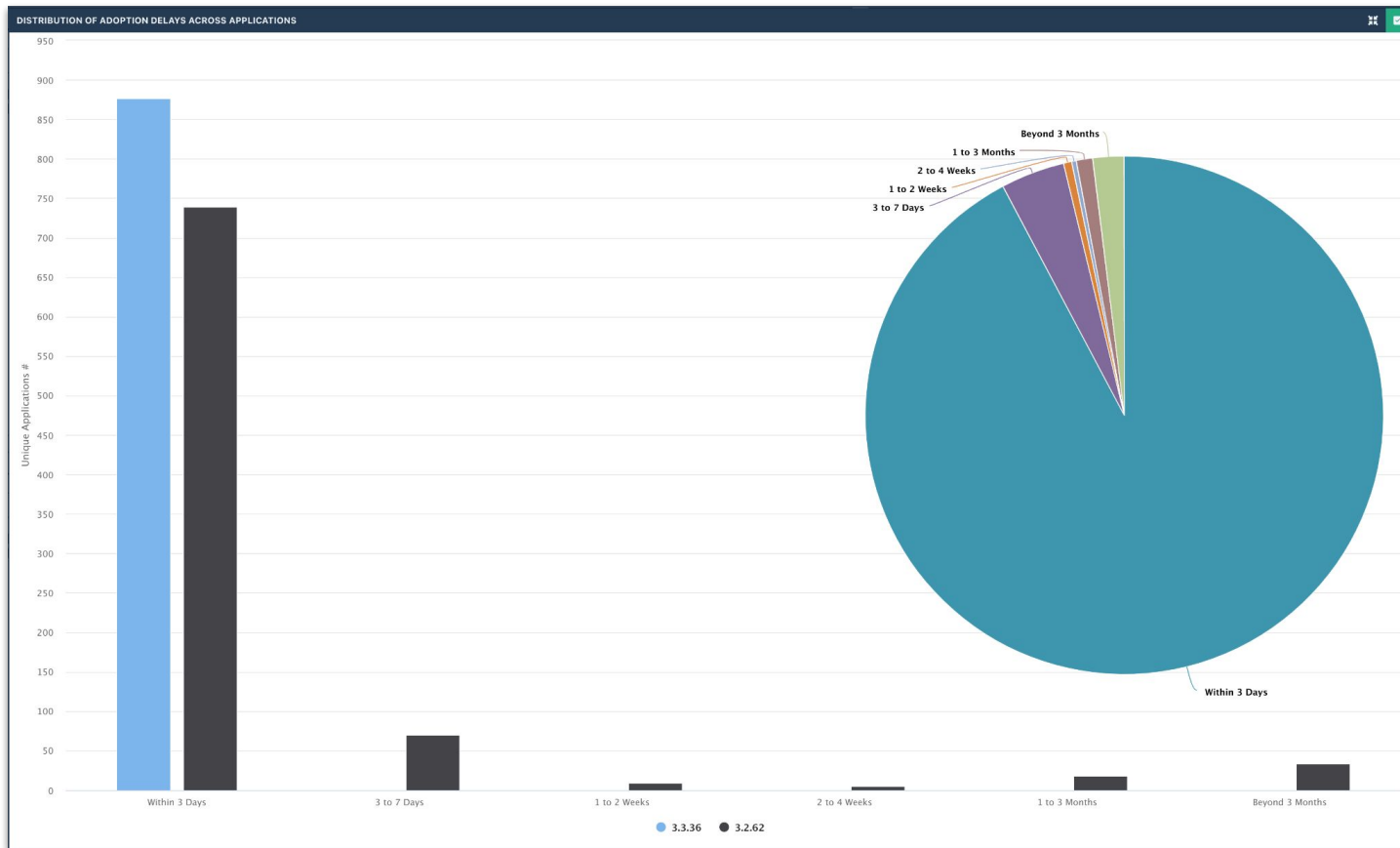


Improved user confidence!

Confidence score distribution

It's paying off

Adoption of platform-provided libraries has become faster

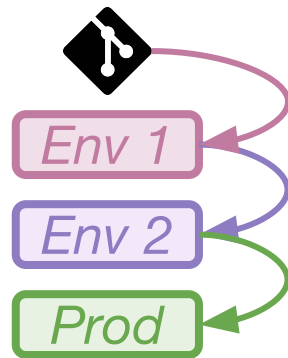


Confidence chimp campaign

For those who are feeling confident and are fully automated

What is it?

- “Throw a switch” to build and deploy latest commit
- A platform-led exercise that is run periodically to identify gaps and/or friction in Netflix's SDLC workflows



Expected benefits

To our company:

- Better prepare for automatic remediation of incidents and automating code changes for migrations and/or upgrades.

To our engineers:

- Follow-ups will contribute to improvements in CI/CD, which leads to increased reliability and stability for service owners.

Confidence chimp campaign

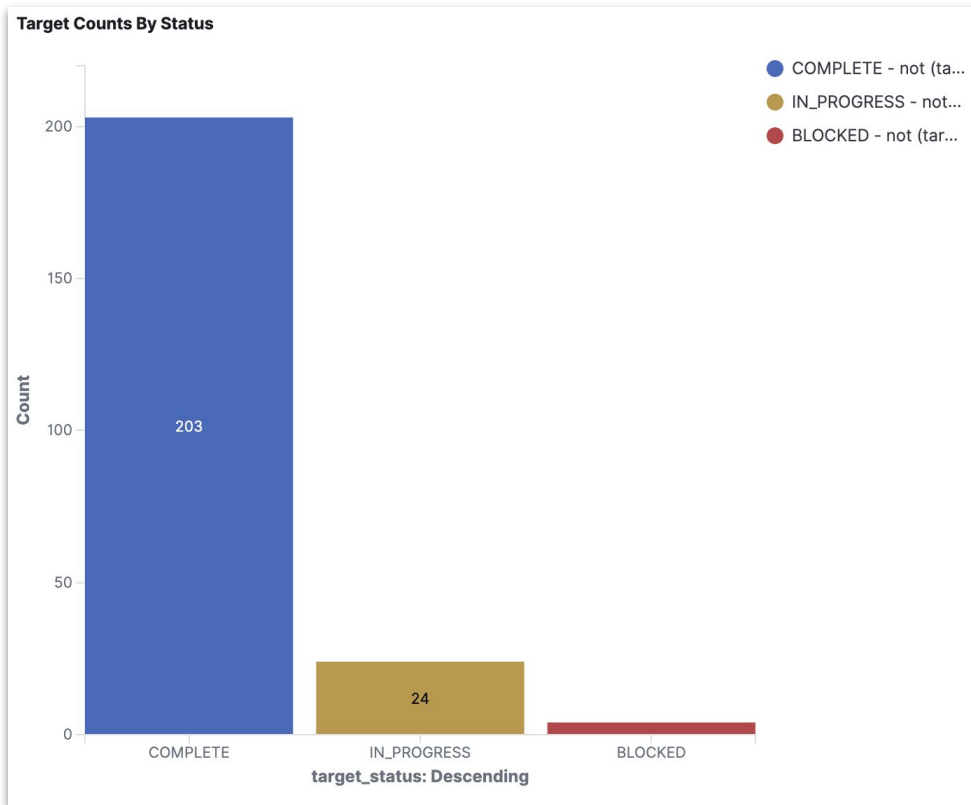
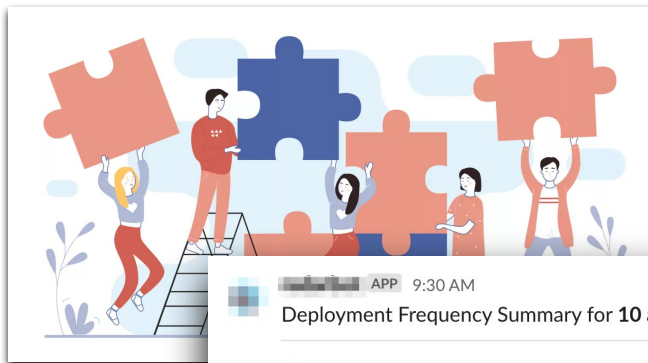


Image from <https://netflix.github.io/chaosmonkey/>

Target applications by status

Recap



APP 9:30 AM

Deployment Frequency Summary for 10 applications

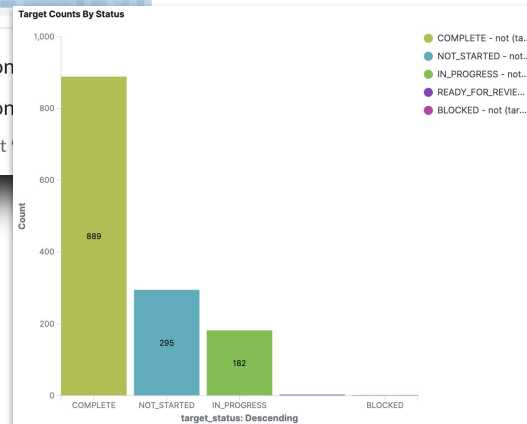
⚠️ — 8 days ago (2 pending artifacts)

✅				

⚠️ — Deployed to production

✅ — Deployed to production

🗨️ 3 replies Last reply today at



Takeaways

Code changes are a part of life.

We must be adaptable and flexible to these changes.

And why do we care about doing this well?

Doing this badly costs toil, friction, and money

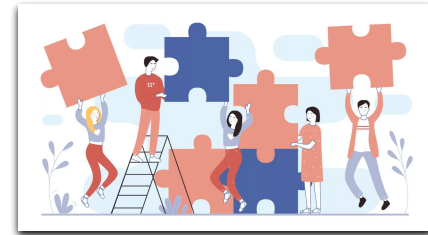
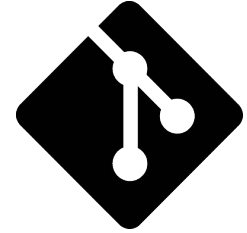


2

Takeaways

Implementing
code changes
goes beyond
transformations
and
scheduling.

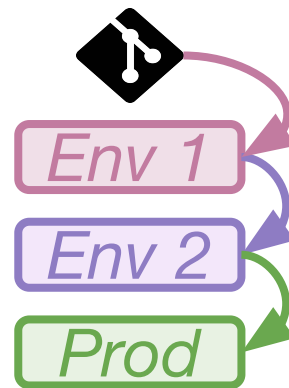
*There are great
companies solving and
tackling this aspect
though*



3

Takeaways

Making and releasing fully automated code changes within the organization has to be a priority with a *cultural* focus.



Takeaways

4

This takes a highly cross-functional team and effort and a company priority



Questions?



Roberto Perez Alcolea
rperezalcolea@netflix.com

Aubrey Chipman
achipman@netflix.com



Thank you!



Roberto Perez Alcolea
rperezalcolea@netflix.com

Aubrey Chipman
achipman@netflix.com



N

