Thomas the Deployer

Author: Brandon Novick

Claps: 77

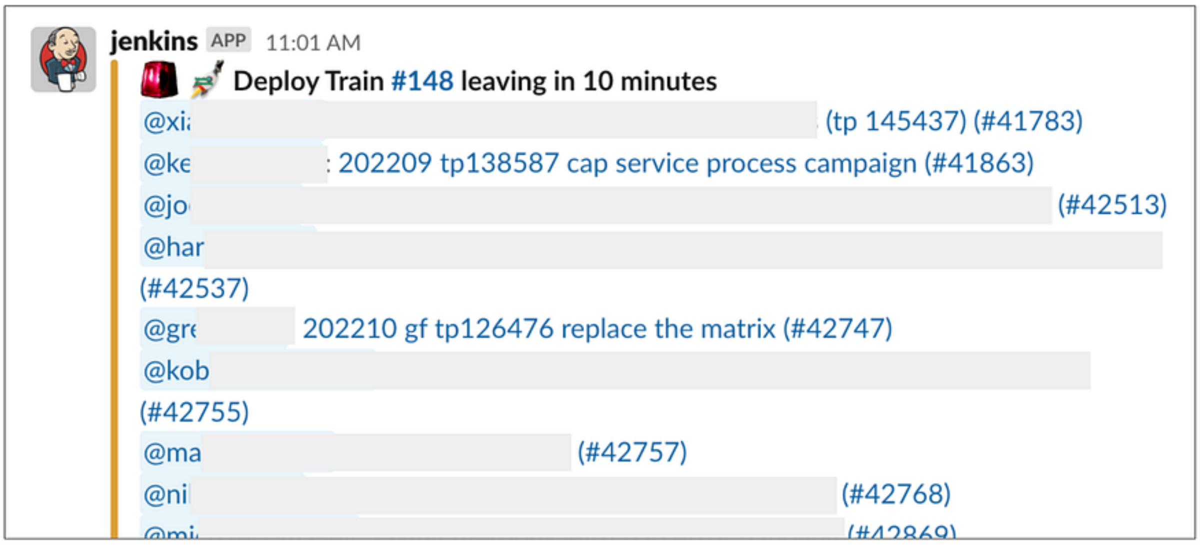
Date: Nov 2, 2022

In this post Iâ€™m going to describe our teamâ€™s recent rollout of **deployment trains**, which allows for the scheduled pooling and automatic deployment of multiple releases. The work builds on years of making our deployments faster and more robust.

Deployment trains: user perspective

Hereâ€™s how **deployment trains** look from the perspective of an engineer at Klaviyo:

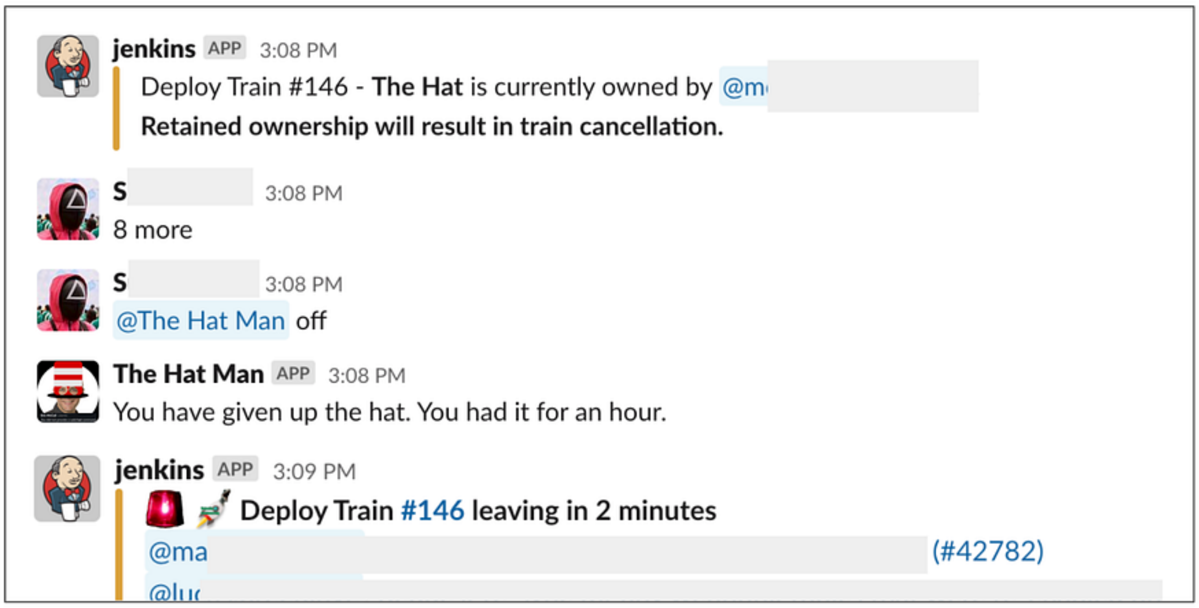
- An engineer creates a PR, waits for CI checks to pass, and receives code review approval.
- The engineer tags the PR as **ready-to-merge** in Github.
- Ten minutes before the â€œtrain departs,â€� all engineers with PRs on the train are notified in our #deployments slack channel. Example:



jenkins APP 11:01 AM
Deploy Train #148 leaving in 10 minutes
@xi[redacted] (tp 145437) (#41783)
@ke[redacted]: 202209 tp138587 cap service process campaign (#41863)
@jo[redacted] (#42513)
@har[redacted]
(#42537)
@gr[redacted] 202210 gf tp126476 replace the matrix (#42747)
@kob[redacted]
(#42755)
@ma[redacted] (#42757)
@ni[redacted] (#42768)
@mi[redacted] (#42869)

Automated post to #deployments channel with usernames and PR names partially redacted. In this example there were a total of 38 PRs.

- At T-6 and T-2 minutes, additional notifications are sent to make sure everyone is aware their code will soon be deployed.
- At â€œdeparture time,â€� the process grabs the hat, which is our lock on merges and deployment. If unable to grab the hat, it notifies the current holder in our #deployments channel and tries again. For example:



jenkins APP 3:08 PM
Deploy Train #146 - **The Hat** is currently owned by @m[redacted]
Retained ownership will result in train cancellation.

S[redacted] 3:08 PM
8 more

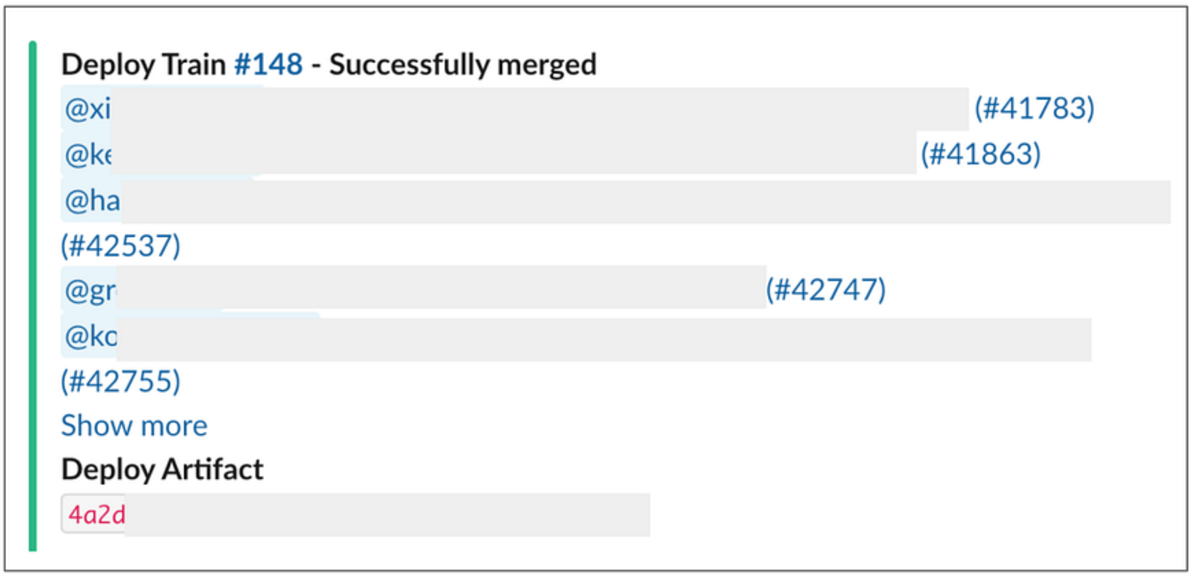
S[redacted] 3:08 PM
@The Hat Man off

The Hat Man APP 3:08 PM
You have given up the hat. You had it for an hour.

jenkins APP 3:09 PM
Deploy Train #146 leaving in 2 minutes
@ma[redacted] (#42782)
@lu[redacted]

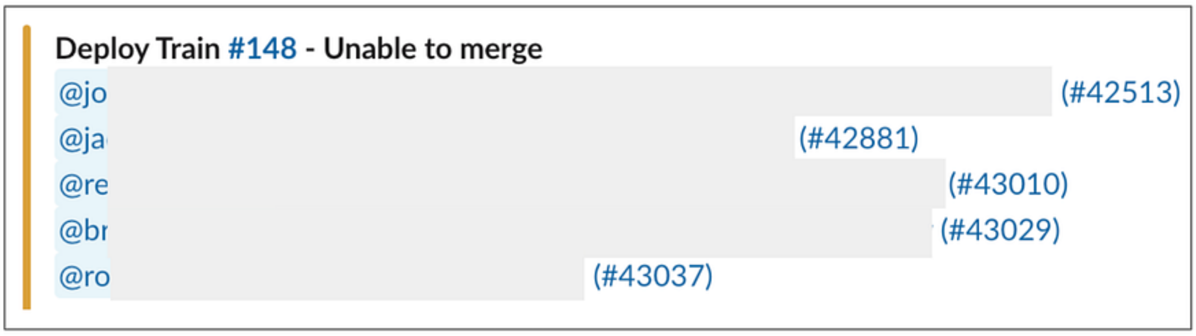
The hat was held by an engineer. Once they give it up the train is allowed to depart.

- A Jenkins pipeline merges the ready-to-merge PRs, one by one. Any that canâ€™t be merged due to merge conflicts and/or unexpected CI failures are tagged **could-not-merge**. More messages are posted to slack:



Deploy Train #148 - Successfully merged
@xi[redacted] (#41783)
@ke[redacted] (#41863)
@ha[redacted]
(#42537)
@gr[redacted] (#42747)
@ko[redacted]
(#42755)
Show more
Deploy Artifact
4a2d[redacted]

Message posted to #deployments listing PRs that were merged

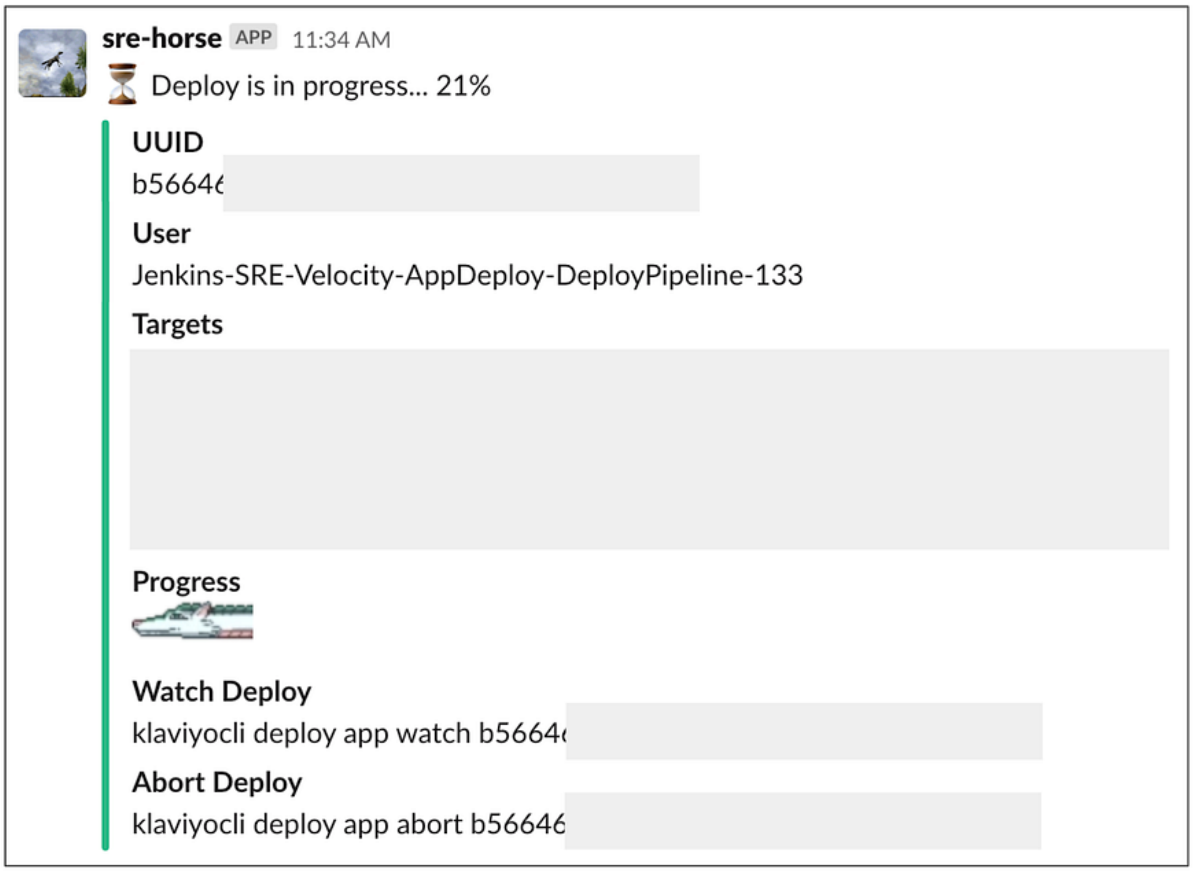


Deploy Train #148 - Unable to merge
@jo[redacted] (#42513)
@ja[redacted] (#42881)
@re[redacted] (#43010)
@br[redacted] (#43029)
@ro[redacted] (#43037)

Message posted to #deployments listing PRs that could not be merged

- Once all of the train-bound PRs are merged, Jenkins builds the deployment artifact.

- Once the deployment artifact is available, Jenkins submits a request to our internal deployment tooling to kick off a deploy.


sre-horse APP 11:34 AM
⏳ Deploy is in progress... 21%

UUID
b5664[redacted]

User
Jenkins-SRE-Velocity-AppDeploy-DeployPipeline-133

Targets
[redacted]

Progress

Watch Deploy
klaviyocli deploy app watch b5664[redacted]

Abort Deploy
klaviyocli deploy app abort b56646[redacted]

Message posted to #deployments showing deployment to clusters in progress

- Deploy is done!

How does this make life easier for our engineers?

- Engineers no longer need to manually deploy all of their releases (but they still can when needed).
- When the â€œhatâ€� is not available, engineers no longer need to enter the queue and wait for their turn to merge and deploy.

Safeguards

Itâ€™s easy to prevent a train from going out. For example, in the event of an incident, someone can simply hold the hat, using our hat man tool. This will prevent a train from departing.

Why not CD?

We do full CD (continuous delivery) on some of our smaller repos. However, our primary application repo has a long CI duration. If we were to move to an automated deploy on merge-to-main strategy, there wouldnâ€™t be enough hours in the day to keep up with the pace of releases. We considered using a waiting period to address the naive queueing problem of one merge is one deploy, but ultimately settled on the scheduled approach because it was simpler to implement and simpler for our team to adopt. We do view automatic pooling and deployment via deployment trains as a soft exploration of how to enable full CD in the future.

How did we build this?

The train is implemented through three Jenkins pipelines: **merge, deploy,** and **revert.**

Hereâ€™s a snippet from the merge pipeline:

We also have a few tricks up our sleeve, for example, avoiding releases on company holidays:

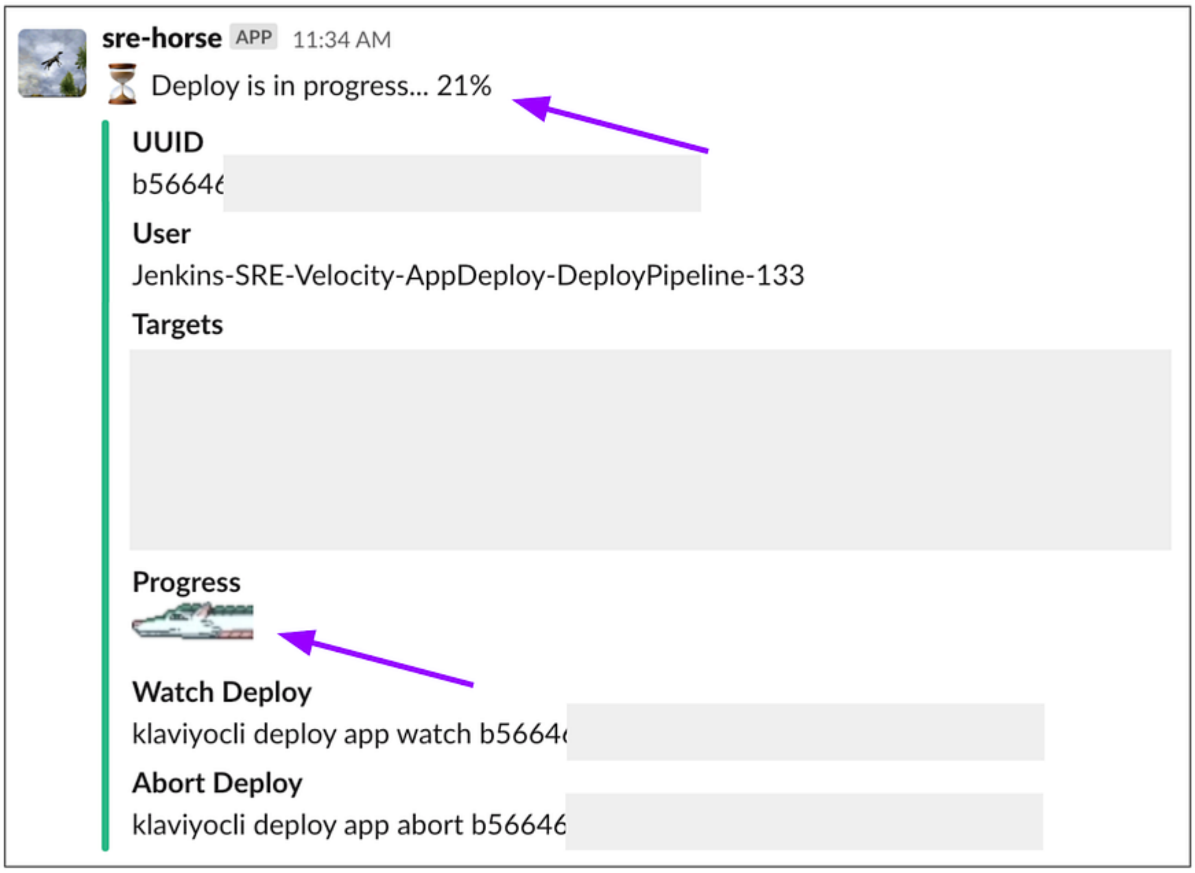
Hereâ€™s the mergePullRequests() function called above. This shows how we use the GitHub API to automatically merge pull requests:

Fun username challenges

Sometimes the little things become a pain. As you can see in the screenshots, we notify our engineers in slack. This means we needed to connect github usernames to slack usernames. Because of the various ways our engineers might be registered with github, and because of what the github API returns, this wasnâ€™t trivial. We ended up writing a script that looks at multiple sources and updates a nightly json file that maps from github to slack usernames.

The Wyvern progress bar!

Before trains, a developer always initiated a deploy on the command line and therefore would watch it in their terminal, know exactly when it finished, and could then confirm things worked as expected in production. With trains, we wanted to make it easy to monitor progress over slack. Our solution was to post and then periodically edit a message in our #deployments channel to display progress. For example:



sre-horse APP 11:34 AM

⏳ Deploy is in progress... 21%

UUID

b5664[redacted]

User

Jenkins-SRE-Velocity-AppDeploy-DeployPipeline-133

Targets

[redacted]

Progress

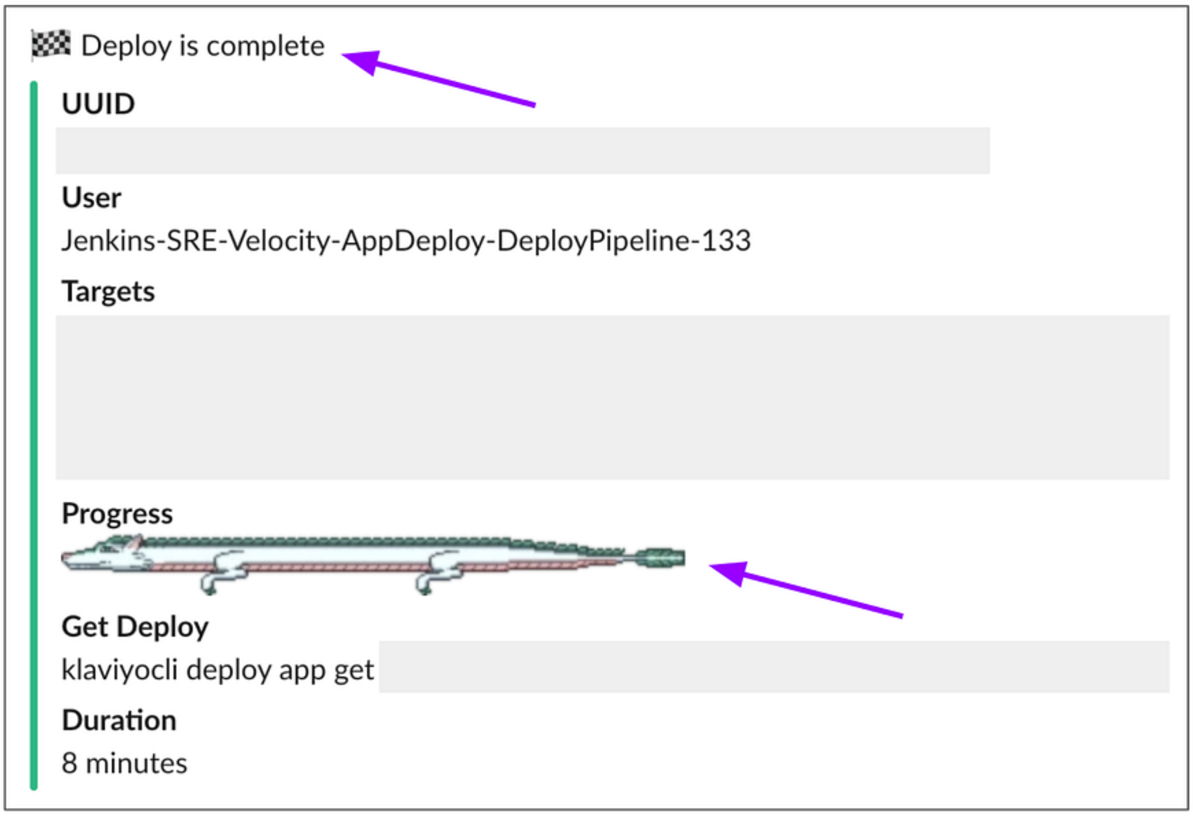
Watch Deploy

klaviyocli deploy app watch b5664[redacted]

Abort Deploy

klaviyocli deploy app abort b56646[redacted]

Message posted to our #deployments channel updated to show that the deploy is 21% complete



🏁 Deploy is complete

UUID

[redacted]

User

Jenkins-SRE-Velocity-AppDeploy-DeployPipeline-133

Targets

[redacted]

Progress

Get Deploy

klaviyocli deploy app get [redacted]

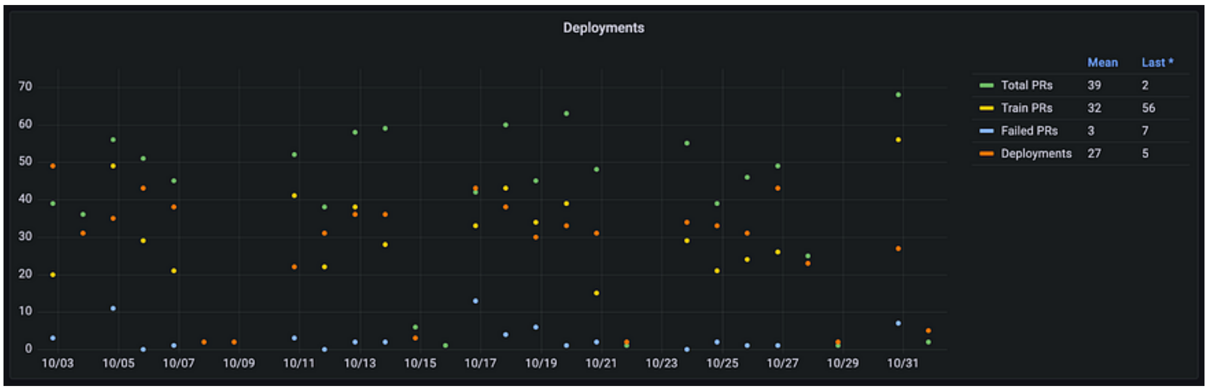
Duration

8 minutes

Final update to message when deployment is complete

Metrics

In our Jenkins pipelines, we ship metrics to Graphite, which is readable via Grafana. Example pane from our deployments dashboard:



Pane from a Grafana dashboard showing PRs deployed each day

Reaction from our engineers

Very positive! Some quotes:

- â€œI know Iâ€™m late to the party but I had my first train deployment this morning. I love this feature, kudos!â€� â€" Bob Long
- â€œDeployment trains have streamlined the process and has saved the company time. It is just what was needed especially as we scale.â€� â€" Prisca Joseph
- â€œWeâ€™ve reduced the story point minimum for tickets that require an app deploy. Thereâ€™s just so much less friction now!â€� â€" Travis Hansen
- â€œItâ€™s really nice not having to queue up with the hat man and wait in line to manually merge in and deploy code. Especially when the changes are super straightforward and donâ€™t need to be deployed right away.â€� â€" Jon Ross
- â€œDeploy trains have meant I donâ€™t have to spend hours waiting to deploy and also donâ€™t have to wrangle tons of tickets together for a larger co-deploy.â€� â€" Will Fergus

Useful links

- GitHub API for Java
- The Hat Man

All aboard the deployment train!