Your* Friend, Datadog

(I** couldn’t think of a better title)

* The new CMS Team

** Nate Douglas

Expectations Unclear

Introduction

Nathan Douglas

- many aliases
- soon-to-be-former-Tech-Lead
- born at a young age
- suffers

who I am

Other Introduction

Datadog

- makes more than I do
- won’t hire me
- most of our DevOps data flows through there and whatever doesn’t probably should too

what I’ll be doing today

What Am I Doing Here?

- The CMS Team is ultimately responsible for two large, intricate products that have seen a lot of turnover.
- Datadog is the core of the many-tendriled apparatus that provides most of our insight into our product delivery processes and outcomes.
- Understanding the peripheral systems that feed into and from Datadog is absolutely essential to success (but that's covered in the document in Confluence).

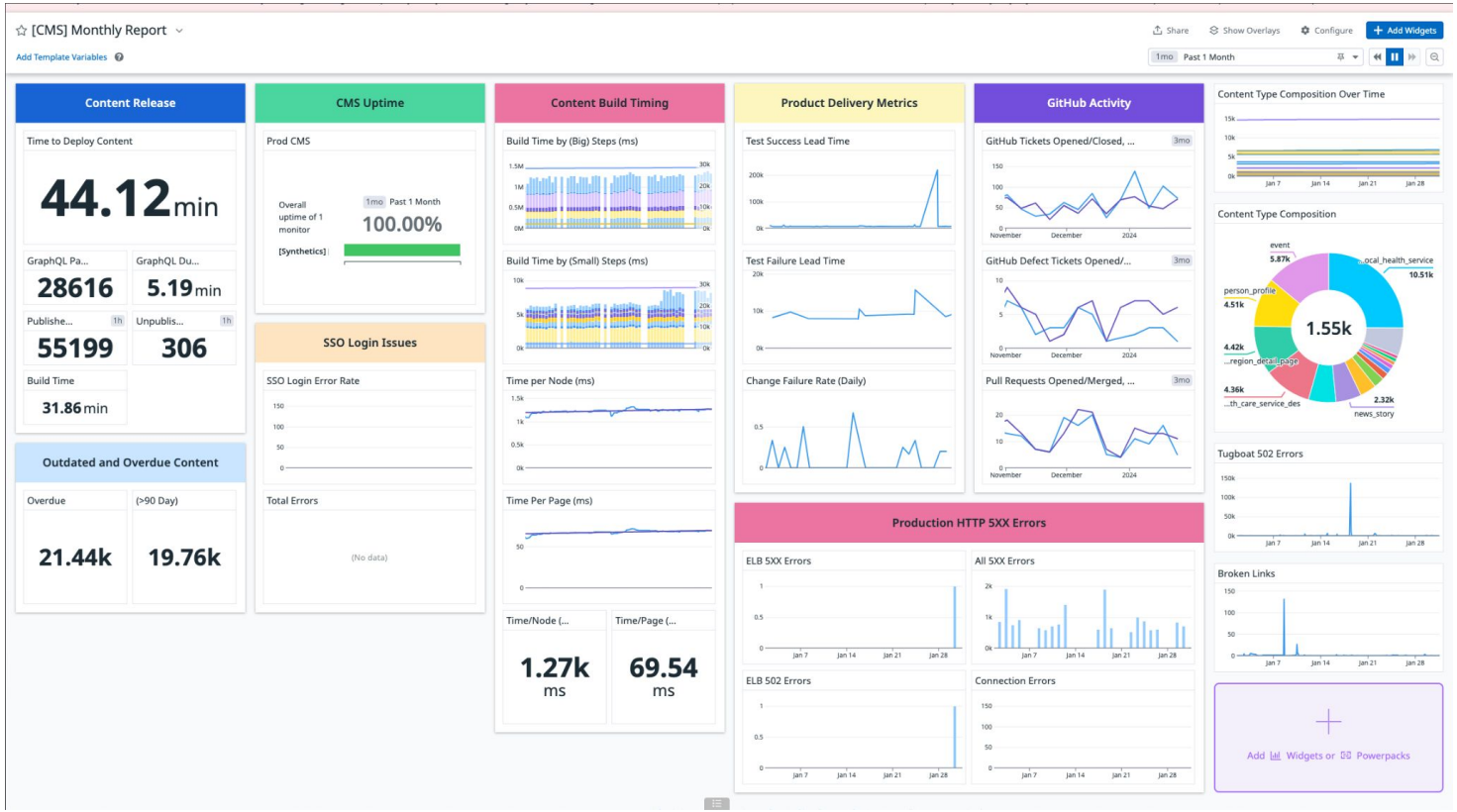
Responsibilities

Below are the CMS Team's key responsibilities pertinent to the contents of this document:

- timely, reliable, frequent, quick, and correct product delivery of two major products:
  - the Drupal CMS (in `va.gov-cms`) that hosts the content of the unauthenticated experience on VA.gov
  - the content release application (in `content-build`) that retrieves that content and renders it for the public
- the infrastructure that hosts the Drupal CMS, including:
  - a `staging` and `prod` environment in both primary (`cms`) and "test" (`cms-test`) configurations
  - Single Sign-On functionality to all of the above (using PIV cards via the VA IAM team and SAML technology)
  - the Tugboat platform that hosts persistent "demo" environments for editors and developers on other teams to test content and functionality changes, and supports certain automated tests
- the continuous integration and continuous delivery pipelines for both products, including:
  - the Jenkins "BRD" system that Builds, Releases, and Deploys the CMS into each of its four environments
  - the execution of certain automated tests on Tugboat, namely those that *require* a fully bootstrapped, fully functional CMS and production content:
    - content release query verification via `content-build-gql`
    - functional API and code tests via PHPUnit
    - behavioral browser-based tests via Cypress
    - accessibility tests via Cypress
    - general system integrity via `status-error`
  - the execution of certain automated tests in GitHub Actions workflows, namely those that do *not* require a fully functional CMS:
    - code style linting via PHP, PHP_CodeSniffer, ESLint, StyleLint,
    - static analysis via PHPStan
    - unit tests via PHPUnit
    - some minor content model consistency checks via `check-fields`
    - Composer internal consistency via `composer-validate`
    - code quality and compliance via CodeQL
- various bits of functionality that don't fall neatly into the above categories:
  - automated S3 backups of the prod CMS filesystem executed from GitHub Actions workflows
  - regularly-scheduled jobs performed on the prod and staging CMS instances invoked from Jenkins
  - the CMS contains a state machine that controls and dispatches content releases

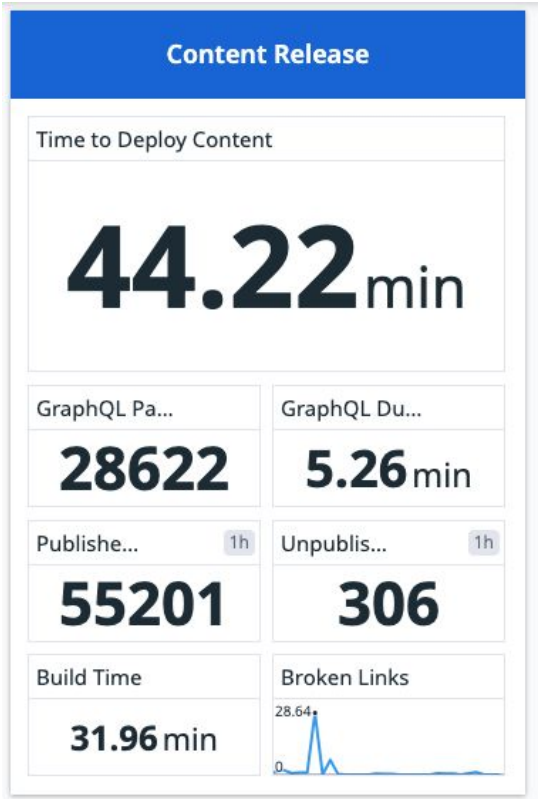
I'm sure I'm forgetting some; and, to be clear, this is not a comprehensive list of all of the CMS Team's responsibilities, merely those that are relevant in this specific context.

The Monthly Report

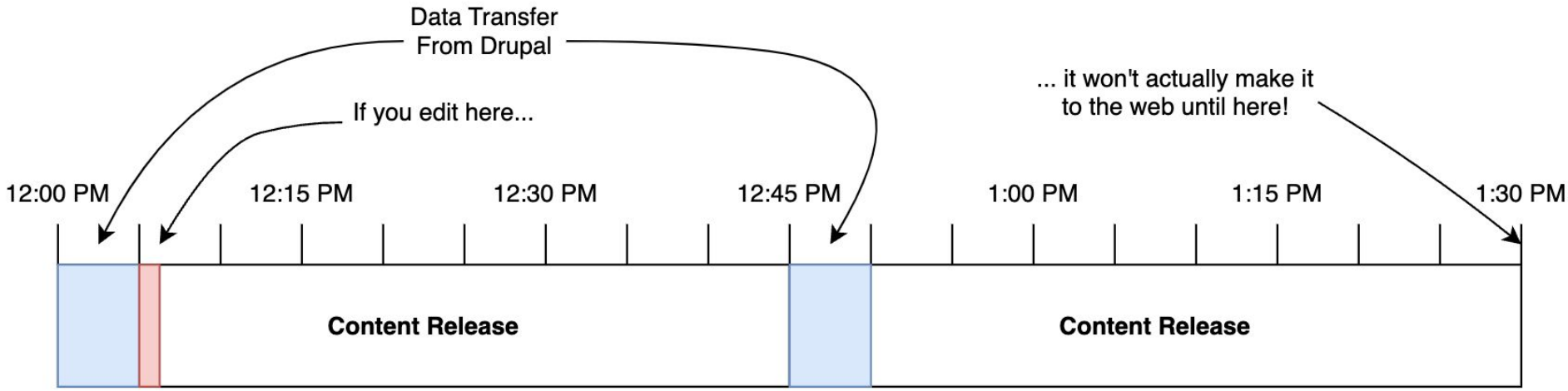


Content Release

- Releases take ~44 minutes.
- Of that, the build takes ~32 minutes and the deployment takes ~12 minutes.
- Communication between the frontend script and Drupal only lasts ~5 minutes.
- Beware! It can take up to ~2 hours for a content change to make it to VA.gov.

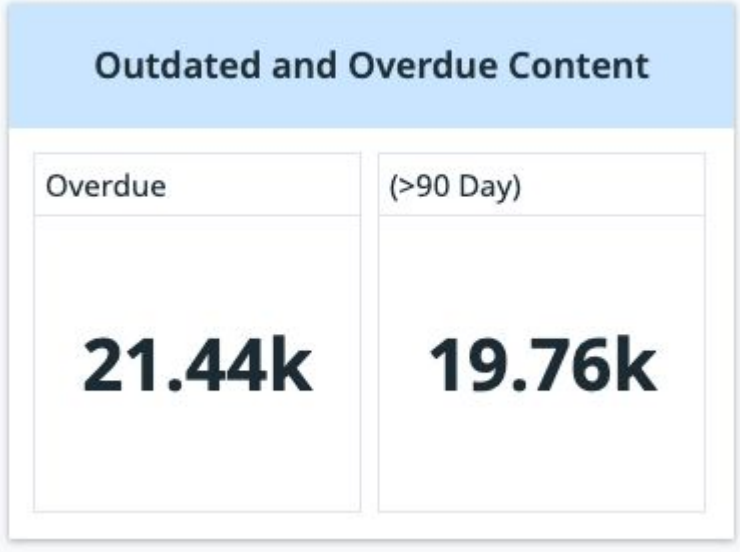


Unfortunate Editor Experience



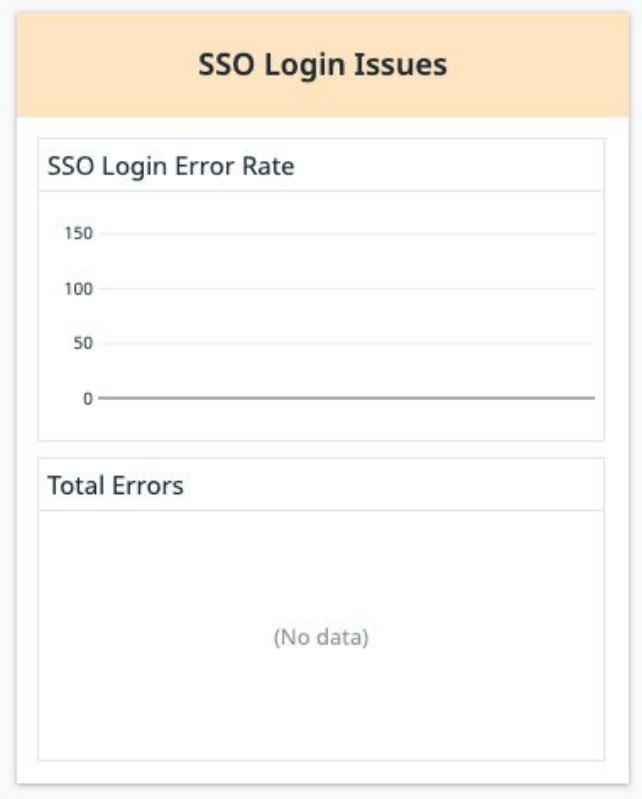
Outdated and Overdue Content

- VA Directive 6102 requires that websites keep content from becoming stale (longer than a year since human review).
- Our responsibility here is to track that outdated content and, once a month, email notifications to the owners of outdated content.
- The notification system is implemented in custom code in Drupal.



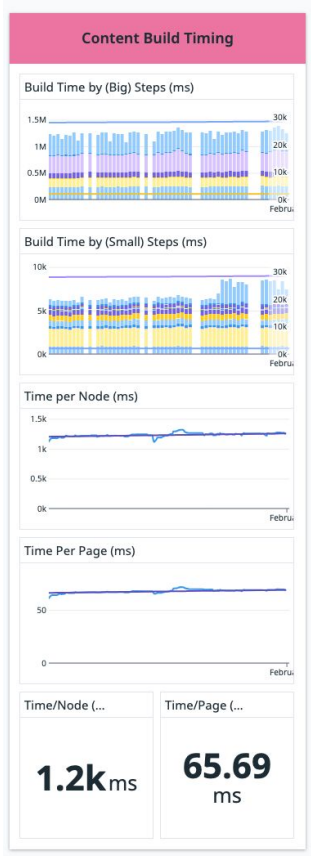
Single Sign-On Login Issues

- This displays errors occurring when an authenticated user relays the cryptographic record of their successful authentication to Drupal and something fails catastrophically.
- 1 Error = Non-Critical PagerDuty Incident.
- 2 Errors = Critical PagerDuty Incident.
- SSO problems can occur and not be visible on this graph; but usually that means it’s outside of our control.



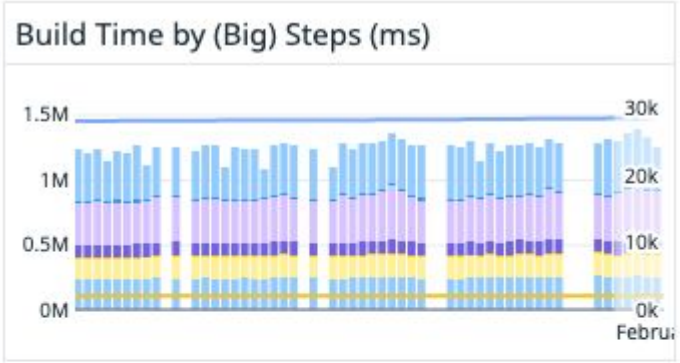
Content Build Timing

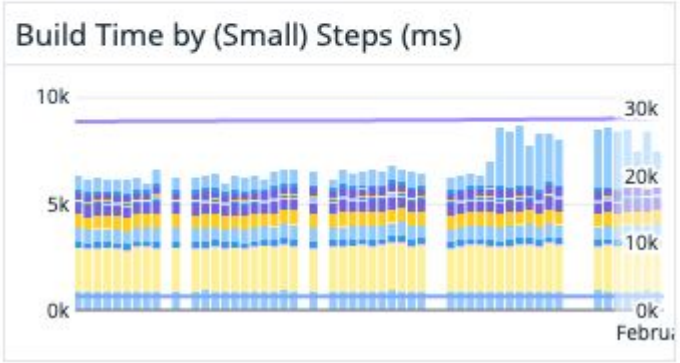
- This displays a time-based breakdown of the “build” portion of a content release.
- The steps are broken into two graphs because of the multiple-order-of-magnitude difference in duration between some of the steps.
- Generally pretty stable, might help in troubleshooting some weird regressions.
- Page = Frontend, Node = Backend; not always 1:1



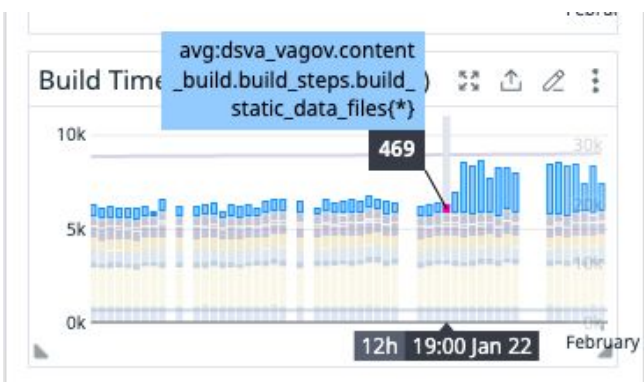
Content Build Timing (2)

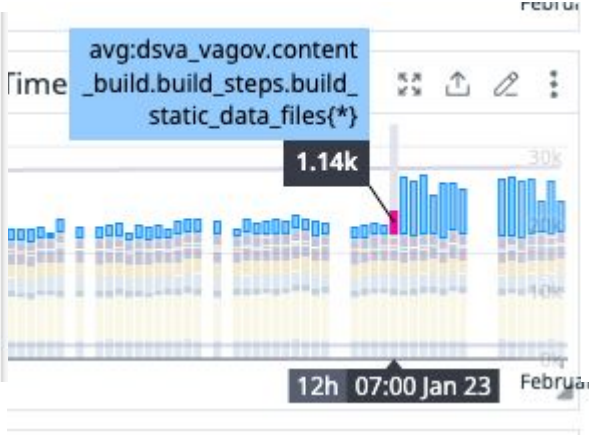
- Again, this is mostly stable.
- Be careful with units when looking at these graphs.

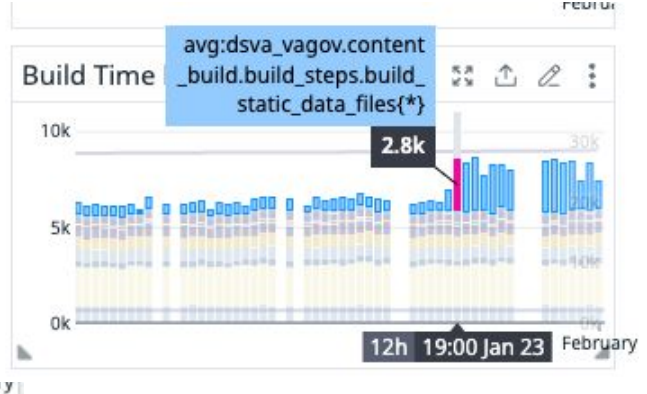




Content Build Timing (3)

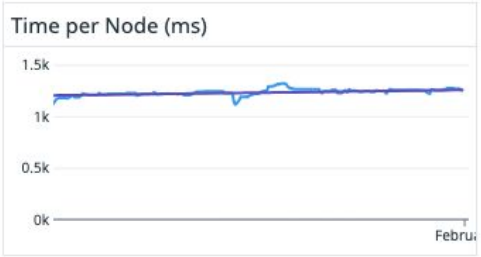




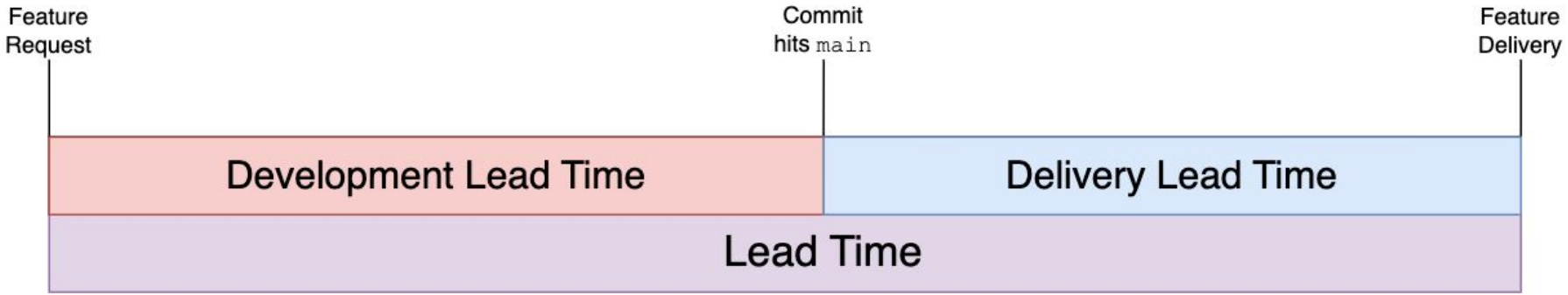


Content Build Timing (4)

- Very stable
- Unlikely to be a source of concern



DevOps Noises



Development vs. Delivery

Development Lead Time

- More variable
- Every feature is a new and distinct variety of nightmare
  - Definition
  - Prioritization
  - Meetings
  - Design
  - Suffering
  - StackOverflow
  - More Suffering
  - Code Review
  - Continuous Integration tests
- Hard to quantify
- Can’t automate core components

Delivery Lead Time

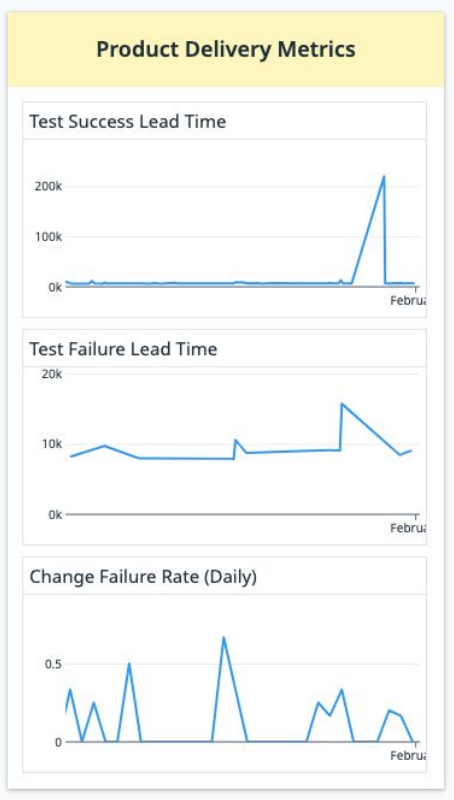
- Less Variable
- The same thing every day
- Easy to quantify
- Highly automatable

Common Product Delivery Metrics

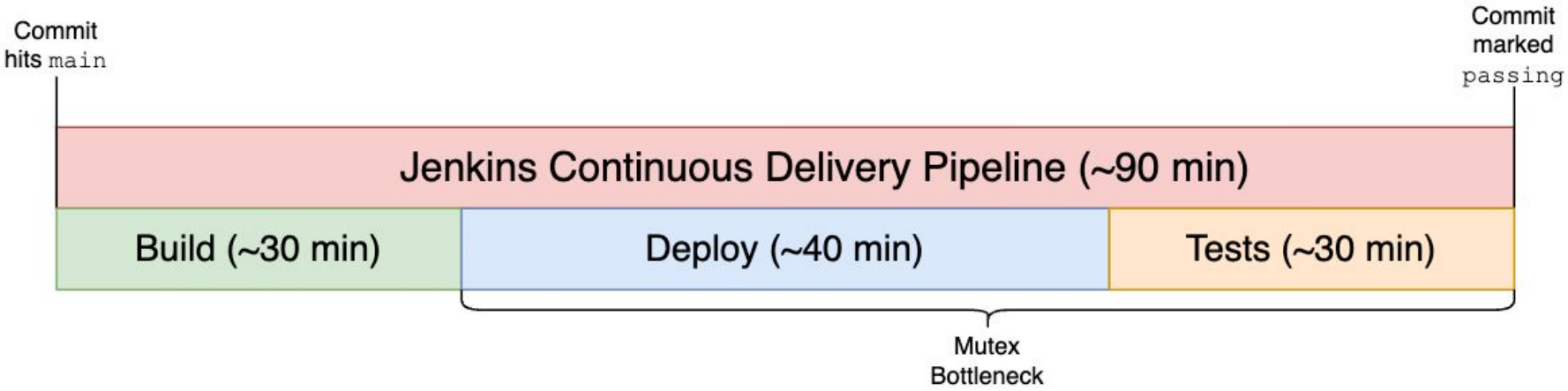
- Product Delivery Lead Time: Time between commit and delivery
- Change Failure Rate: Percentage of changes that cause a failed delivery or degraded service
- Mean Time to Recover: Average time needed to recover from a failed delivery
- Deployment Frequency: How often we actually deploy (Continuous Delivery does not imply Continuous Deployment)
  - CMS deploys once per weekday
    - current architecture (and to some degree, the nature of Drupal) imposes a 10-15 minute “dead zone” where no one can interact with the site
    - Used to be once per week IIRC
  - Creation/release of new `content-build` versions happens once per weekday
    - Until this happens, all content releases use the previous day’s version

Product Delivery Metrics

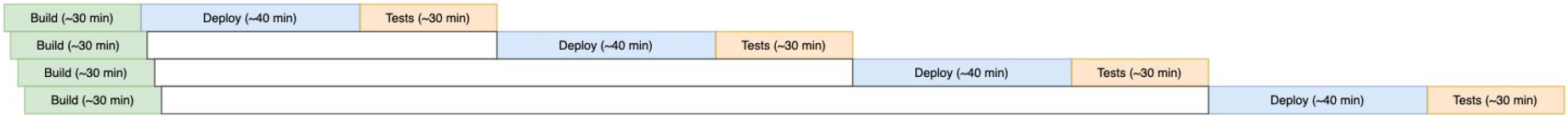
- Test Success Lead Time (proxy for Product Delivery Lead Time)
- Test Failure Lead Time (as above, but could be reduced)
- Change Failure Rate (test failures)
- Deployment Frequency = 1
- Mean Time to Recover is absent



CMS Continuous Delivery Pipeline

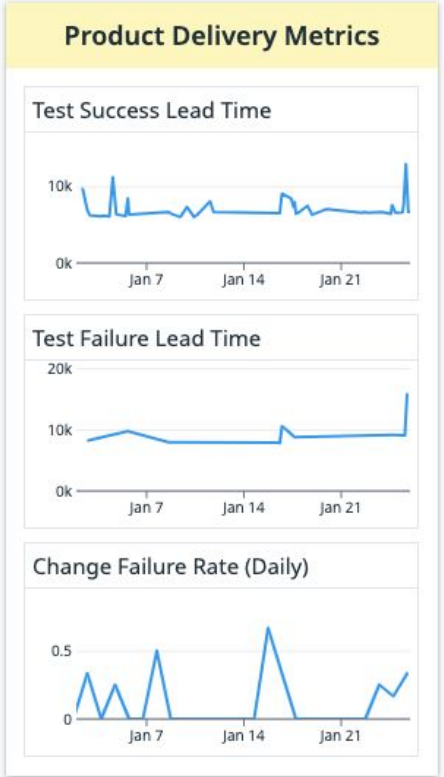


CMS Continuous Delivery Pipeline (2)



Product Delivery Metrics Revisited

- Parallelizing Continuous Delivery pipeline would dramatically shorten the general Lead Time.
- Failing early could improve Test Failure Lead Time.
- Change Failure Rate generally reflects the flakiness of the test base, though sometimes infrastructure issues will be the cause.

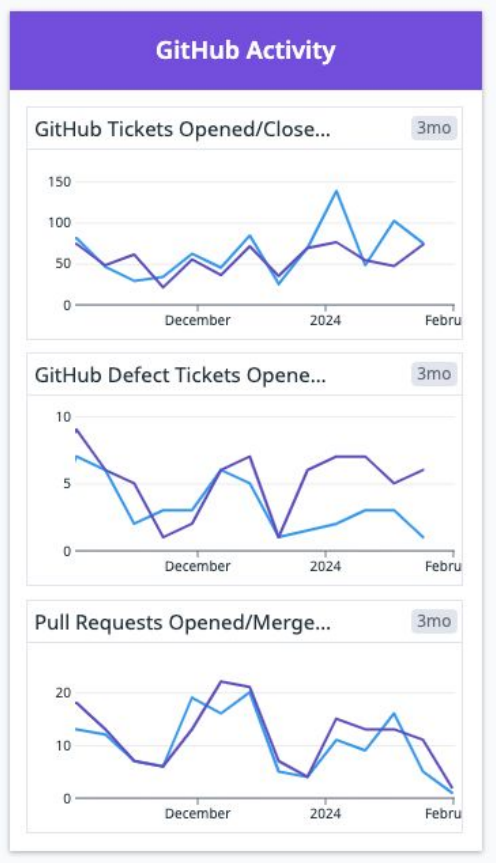


Production HTTP 5XX Errors

- Mostly ignored, can be a sign of a misconfiguration, bug in an upstream package, etc.
- We use Sentry for alerting on user-facing errors.
- ELB 502s can be caused by mismatches between LB and backend idle timeout, segfaults on the backend, etc.
- Causes of other 5XX errors might be programming errors, etc.

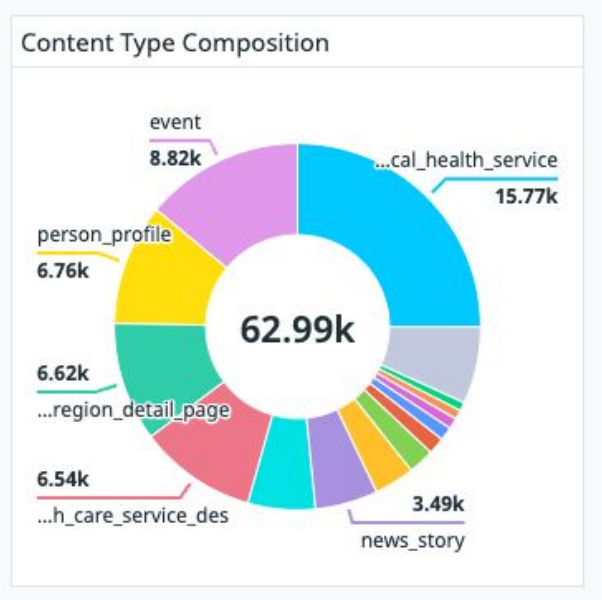
GitHub Activity

- This shows metrics about tickets in the `va.gov-cms` repository.
- The ratios of opened:closed tickets and opened:merged pull requests can give a rough picture of a team's balance and code quality.



Content Type Composition

- This provides insight into how many nodes exist of each type.
- This will change very slowly over time.



Links

- Deep technical document about Datadog and its tendrils https://vfs.atlassian.net/wiki/spaces/PCMS/pages/2929919119/Metrics+Monitoring+and+Incidents
- Datadog monthly report dashboard https://vagov.ddog-gov.com/dashboard/4j8-8ev-m2u/cms-monthly-report
- CMS Prod & Staging dashboard https://vagov.ddog-gov.com/dashboard/vnk-g4s-fru/cms-prod--staging
- HTTP 502-focused dashboard https://vagov.ddog-gov.com/dashboard/f87-xne-z8d/cms-502s-prod-staging-tugboat
- TIC latency dashboard https://vagov.ddog-gov.com/dashboard/qcx-jbg-bqu/cms-tic-monitoring-dashboard

Conclusion

- How to contact me:
  - Before February 14:
    - Slack: @Nathan Douglas
    - Email: nathan.douglas@agile6.com
    - Phone: 1 (330) 998-8049
    - LinkedIn: nug-doug
  - On/After February 14:
    - Don’t
- Donations to SCFEND can be made directly to me
- I’m so, so sorry