Collecting and visualizing Continuous Delivery Indicators

February 7th, 2021
About me

• Vincent Behar
• Engineer at Dailymotion
• https://github.com/vbehar
• https://twitter.com/vbehar
Collecting and visualizing Continuous Delivery Indicators

In a Kubernetes-based CI/CD platform, using Jenkins X, Lighthouse, Tekton, PostgreSQL and Grafana.

• Why
• What
• How
Why?

- It’s not «just» CI anymore...
- CD defines how fast you bring value to the business

Jaana Dogan ヤナ ドガン
@rakyll

Your CI/CD pipelines are a part of your production story. Observability for CI/CD is observability for a critical prod component.

1:24 AM · Dec 22, 2020 · Twitter Web App
Why?

- What’s the performance of your CD platform?
- How do you know if it’s slow or broken?
- “If you can’t measure it, you can’t improve it”

Rick Branson
@rbranson

Observability is putting temperature probes in the turkey *before* you put it in the oven, monitoring is checking it every 20 minutes with your fingers.

10:40 PM · Dec 25, 2020 from Yucca Valley, CA · Twitter for iPhone
What?

- What should we measure?
- Devs
  - Build duration
  - Build failure rate
- Ops
  - Deployment failure rate
  - Time to recover
- Product
  - How long before it’s in prod?
Standards
Throughput

• **Mean Lead Time**
  • How long for a commit to go to production
  • 1 day, 1 week, 1 month, 6 month

• **Deployment Frequency**
  • How often you deploy code to production
  • On demand, 1 per week, per month, 6 months
Stability

• Mean Time To Recover
  • How long to restore the service
  • 1 hour, 1 day, 1 week/month

• Change Failure Rate
  • % of change resulting in service failure
Standards

• Why use them?
  • DORA « DevOps reports »
  • Elite, high, medium, low performance teams

• How should we use them?
  • Automatic collection of metrics
  • Requires a very precise definition
  • Use them to improve your
    • Process / workflow
    • Tools
How?

• Define your CD Indicators
• Collect them
• Build visualizations
• Setup alerting
Define your CD indicators

• Focus on actionnable indicators
• Based on your:
  • Process / workflow
  • Code review?
  • QA / manual validation?
  • Continuous Deployment?
  • Feature flags?
• Tools
Our own indicators

- Wiki
- Based on our GitOps workflow
- For each indicator:
  - Clear definition
    - Start / end
  - Dimensions
    - Application, Environment, ...
  - Actions
    - If exceed a threshold
  - How to collect
    - System/event to watch
Our own indicators
Our own indicators

• Mean Lead Time:
  • PR ready for review => deployed in prod
  • Use GitHub events
  • Actions: ?
• Time to review
  • PR ready for review => PR approved
• Time to release
  • Commit to master => Release created
• Time to promote
  • Release created => GitOps PR ready
Our own indicators

- Build Failure Rate
- Release Failure Rate
  - Release Failure Recovery Time
- Deployment Failure Rate
  - Deployment Failure Recovery Time
- ...
Our stack
Architecture

Collector

Kubernetes API

PostgreSQL

Lighthouse

GitHub

Grafana
Collecting from Kubernetes

- Custom Resource Definition - CRD
- Tekton
  - PipelineRun
- Jenkins X
  - PipelineActivity
  - Release
- Kubernetes client-go
Collecting from GitHub

- GitOps
- GitHub WebHooks
  - Lighthouse Plugin
  - [github.com/jenkins-x/lighthouse](https://github.com/jenkins-x/lighthouse)
  - Fork of Kubernetes’ Prow
- Pull Request events
- Deployment events
  - DeploymentStatus
  - Pending, in progress, success, failure, inactive
Storing Data

- PostgreSQL
- Easy, powerful
- Integration with Grafana
- Raw data
  - Pipelines
    - Start, end, status, app, type, …
  - Pull Requests
    - Repo, created, ready for review, merged…
  - Releases
    - Repo, commit time, release time, …
  - Deployments
    - App, env, start, end, status, …
Visualization

- Grafana
  - Time series
  - Time range, global variables, ...
- Dashboards packaged in Helm charts
- 1 dashboard per team / app / repo
Alerting

- Make it useful!
  - Define indicators thresholds
  - Take action
- Grafana
  - Per-panel (graph) rules
  - Slack
  - Webhook
Conclusion

• Invest in your CD platform
• It should be an enabler
• Do it early
• Continuous improvement
• Take action
• Modern workflows / tools
• Open-source
Thank you!