DDDevOps

Implementing **and** operating a domain

Christian Folie

christian.folie@outlook.com

Daniel Weller

daniel.weller@trustbit.tech

From last talk 😸

Christian Folie

christian.folie@outlook.com

Daniel Weller

daniel.weller@trustbit.tech

Trustbit

From last talk 😸

Strategic DDD?

DDDevOps?

DDDevOps

Implementing **and** operating a domain

Strategic DDD?

We don't talk about how we discovered & evolved the domain

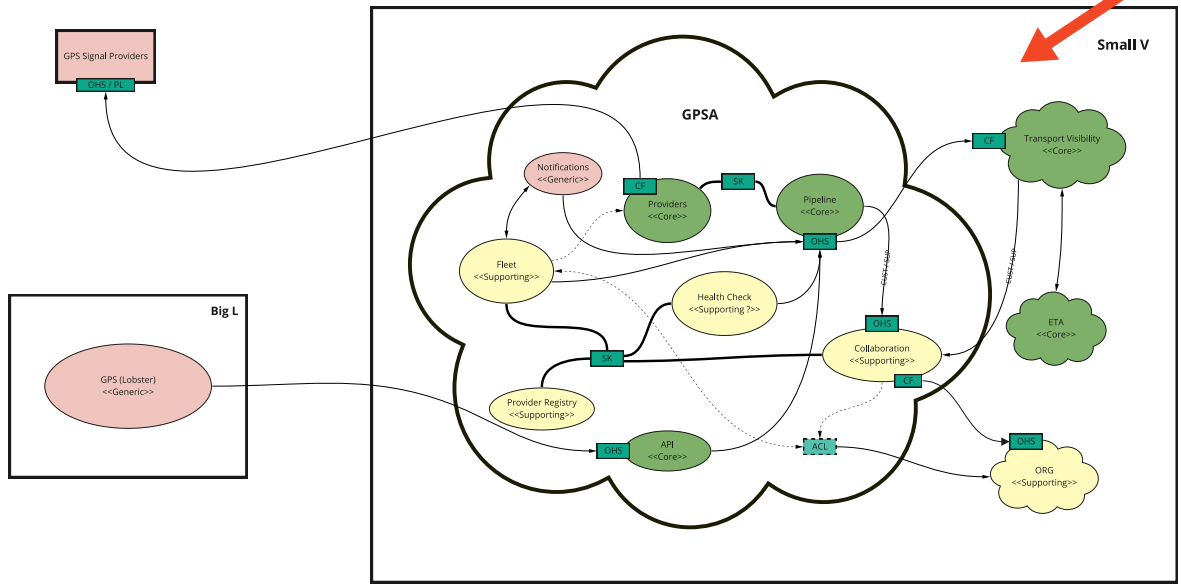
We don't dive into concrete implementation patterns "Tactical DDD"

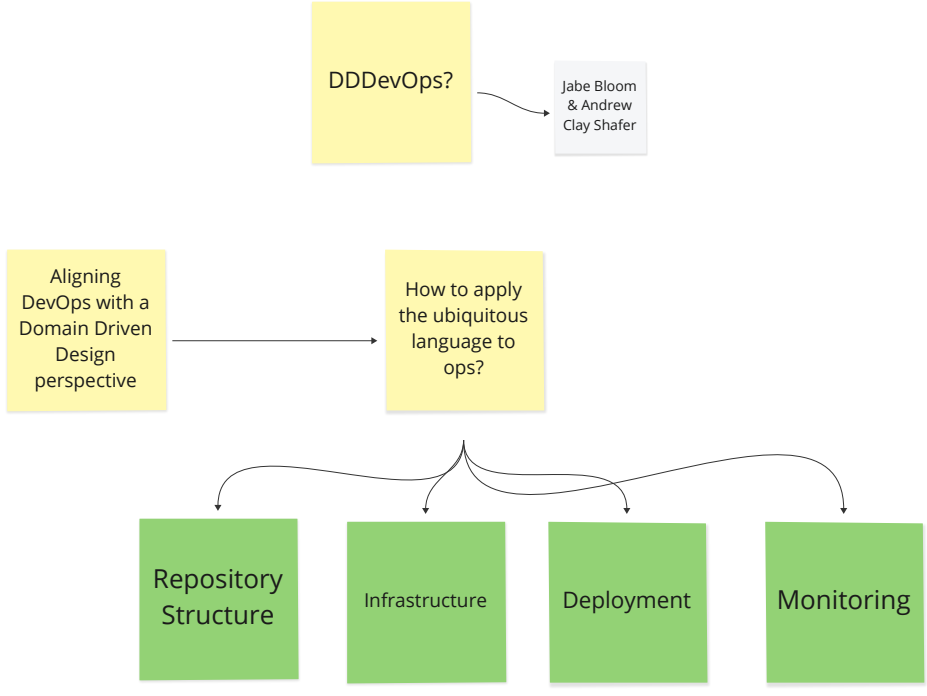
We focus on the development and integration of various Bounded Contexts in a Domain

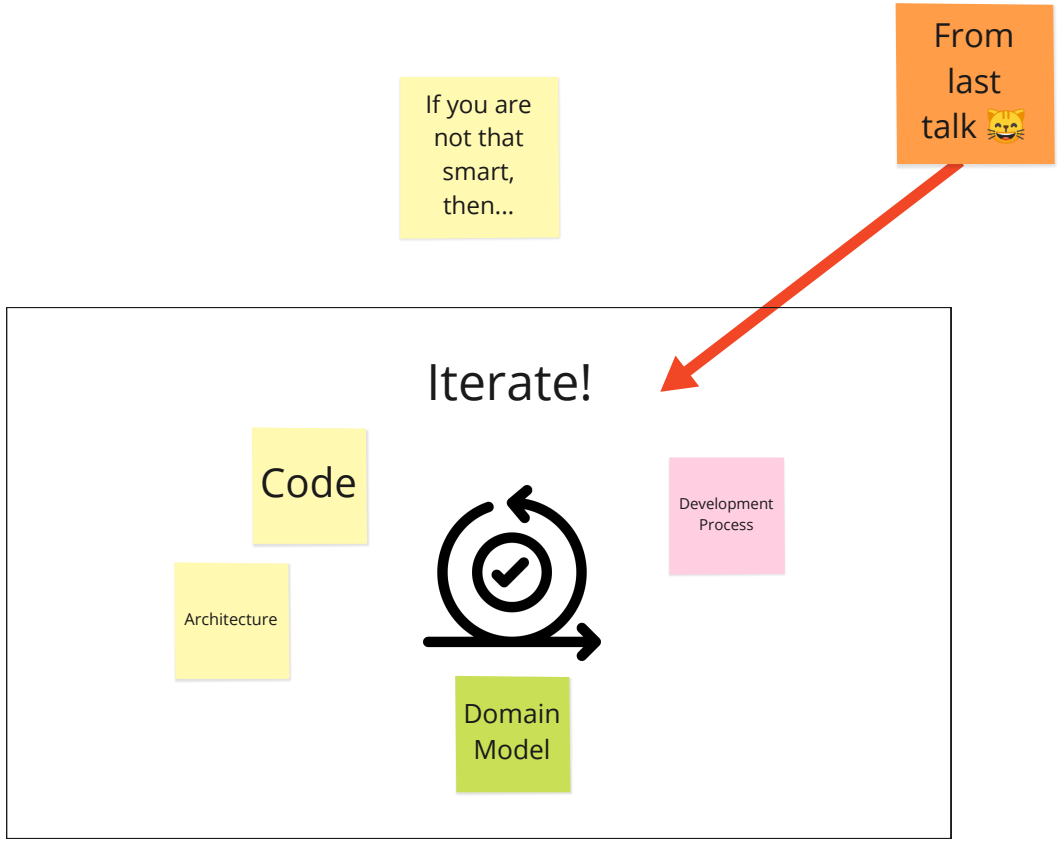
Look up our other talk for this

Strategic DDD?

From last talk 😸




DDDevOps?
Jabe Bloom & Andrew Clay Shafer
Aligning DevOps with a Domain Driven Design perspective
How to apply the ubiquitous language to ops?
Repository Structure
Infrastructure
Deployment
Monitoring


From last talk 😸
If you are not that smart, then...
Iterate!
Code
Development Process
Architecture
Domain Model

What we will discuss today

Evolutionary, *product-driven* approach to DevOps

One team handling all aspects of the product lifecycle

Important *technical* decisions and mistakes

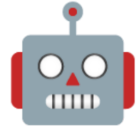
Topics we will discuss today

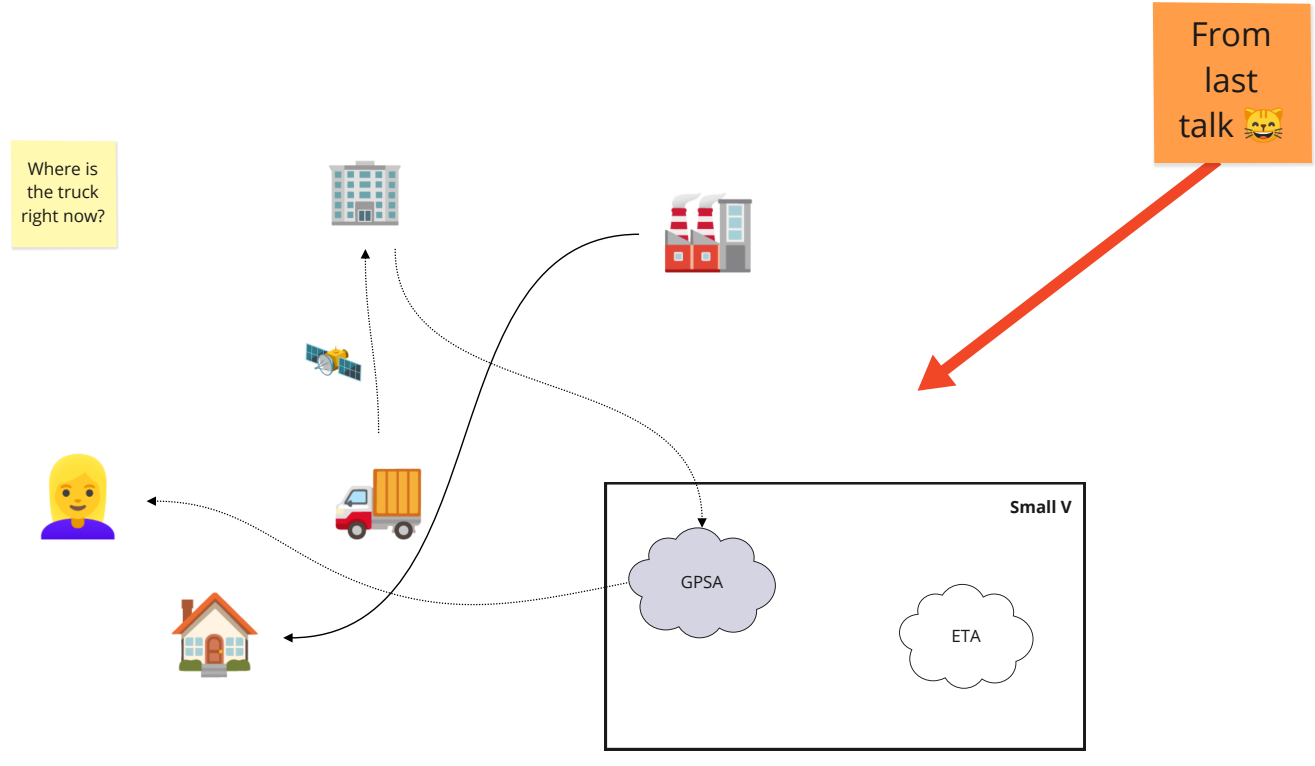
Dev Experience

Architecture

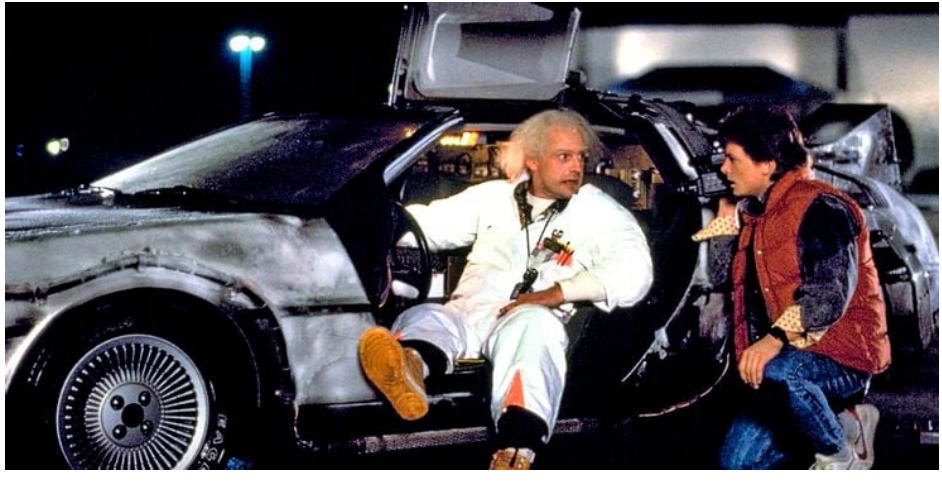
Deployment

Monitoring




From last talk 😸
Where is the truck right now?
Small V
GPSA
ETA

BACK TO THE FUTURE



Early

Green field project

Product

Friendly first customer

New product

Vision vague and unclear

Tech

Google Cloud Platform

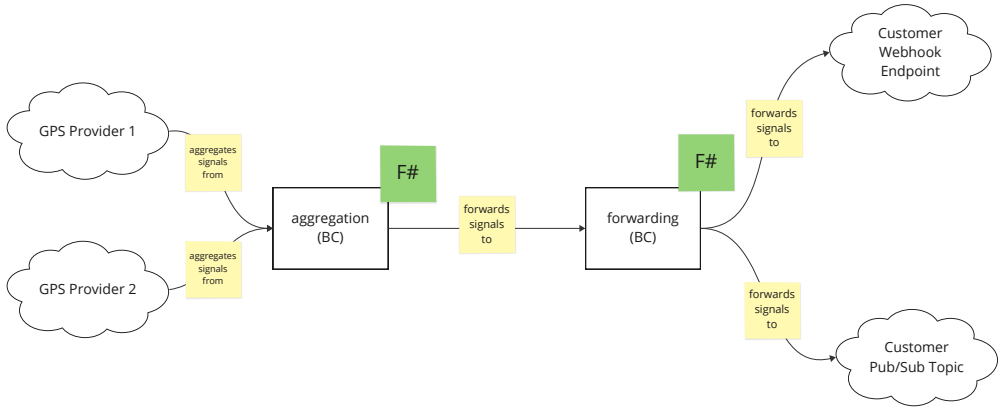
Team develops **and** operates

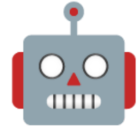
Team

Senior Devs only

Small Team

Signals Domain





Kanban

ASYNC Communication

Readme

IAP for the rescue

Default is good enough

Decoupled

In-Memory

ADR

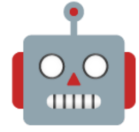
CLI

Feature iteration is fast

Team collaboration is smooth

Convention over Configuration

Important technical decisions are discussed & documented



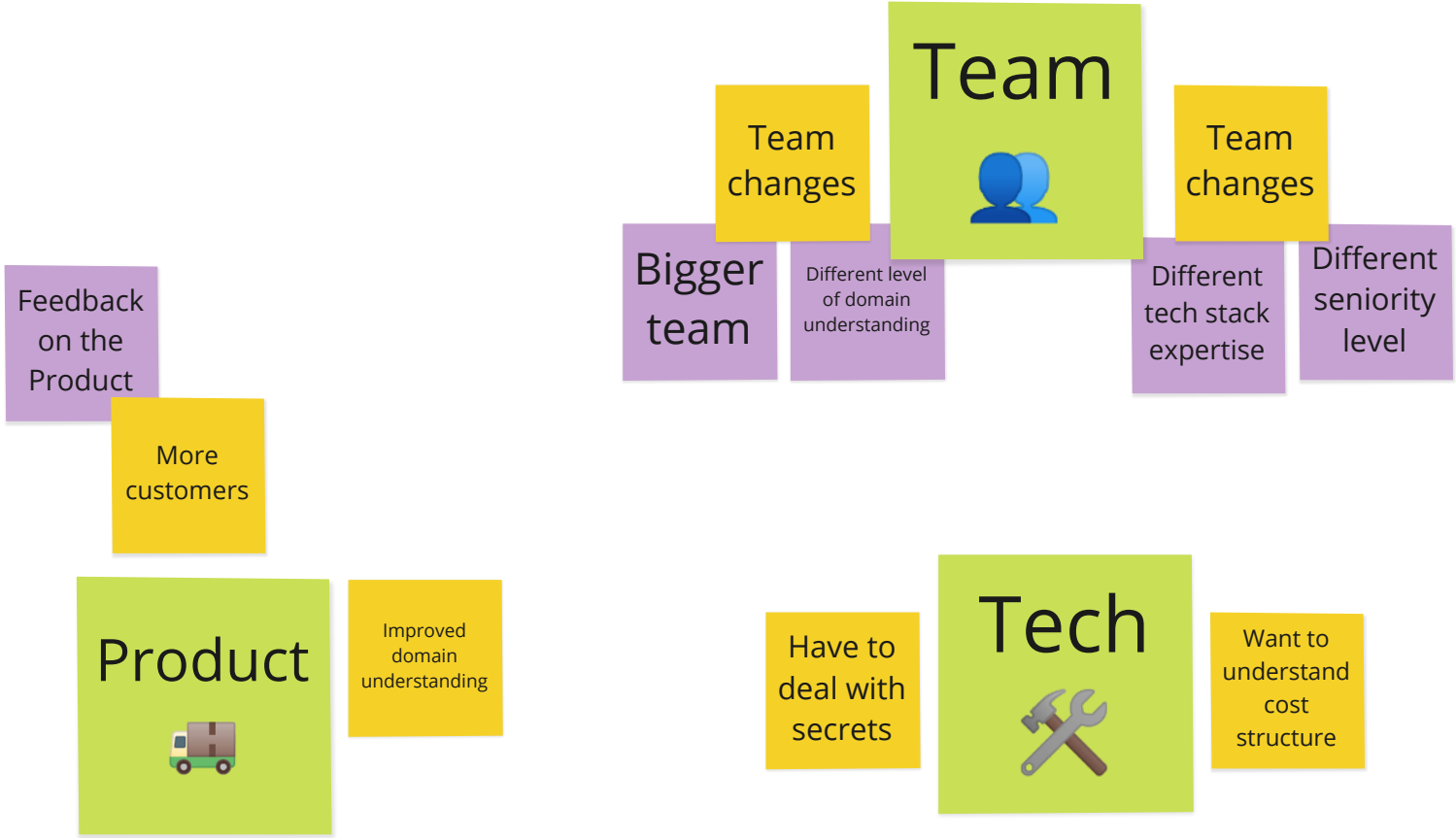
DEV/PROD stages are up

Manual deployment cumbersome

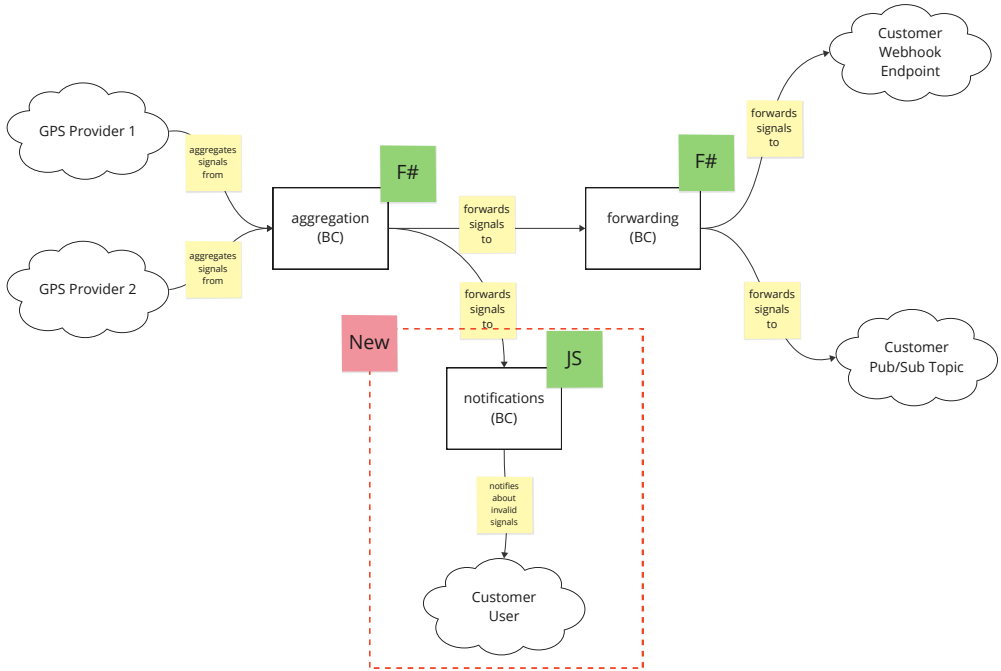
Deployment instructions complicated

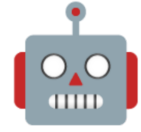
No Domain Monitoring

Mid


Team
Team changes
Team changes
Bigger team
Different level of domain understanding
Different tech stack expertise
Different seniority level
Feedback on the Product
More customers
Product
Improved domain understanding
Tech
Have to deal with secrets
Want to understand cost structure

Signals Domain





Docker-Compose

Application Configuration

Team Agreements

Linting & Formatting

Contract changes

JS

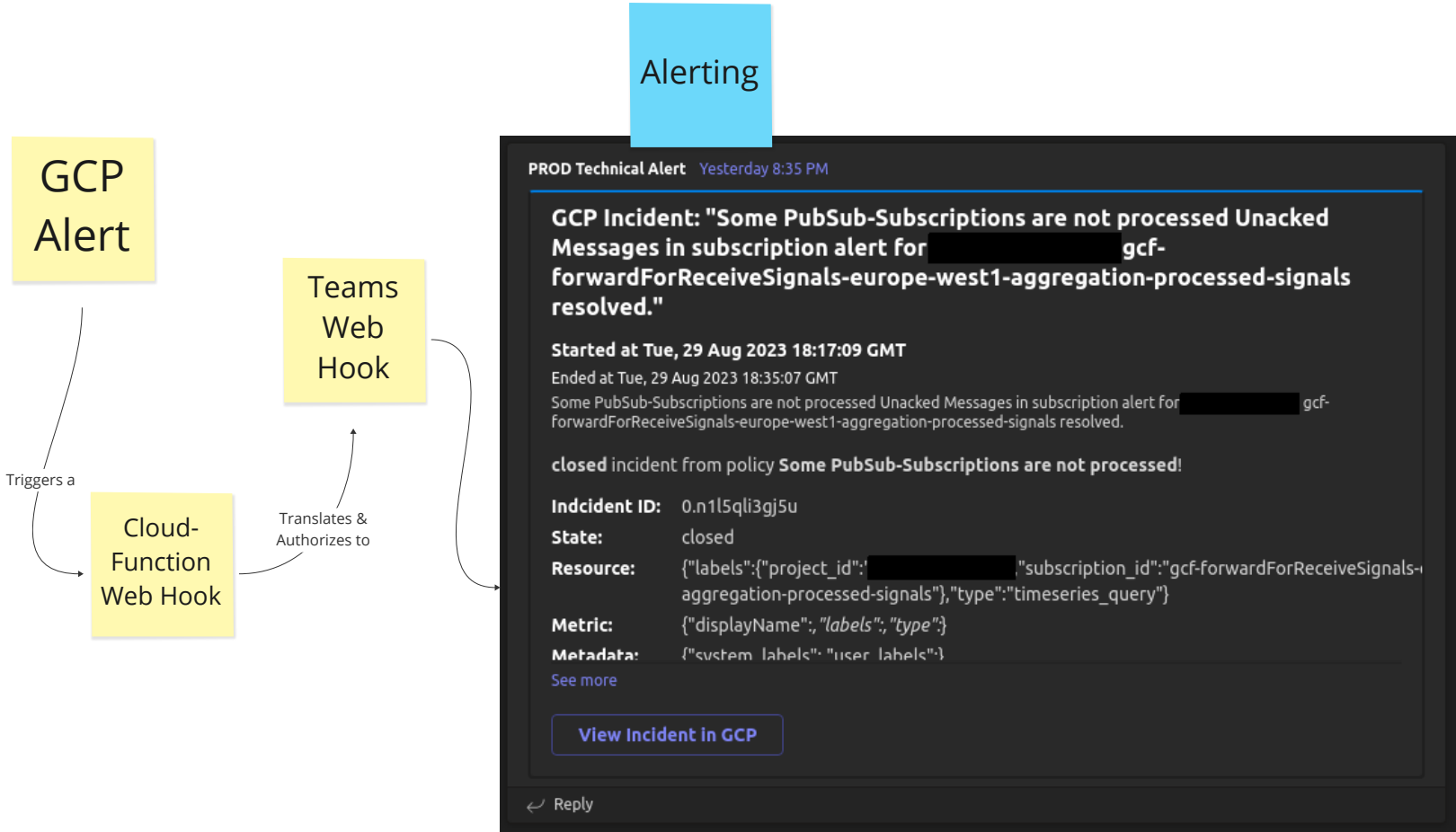
IaC

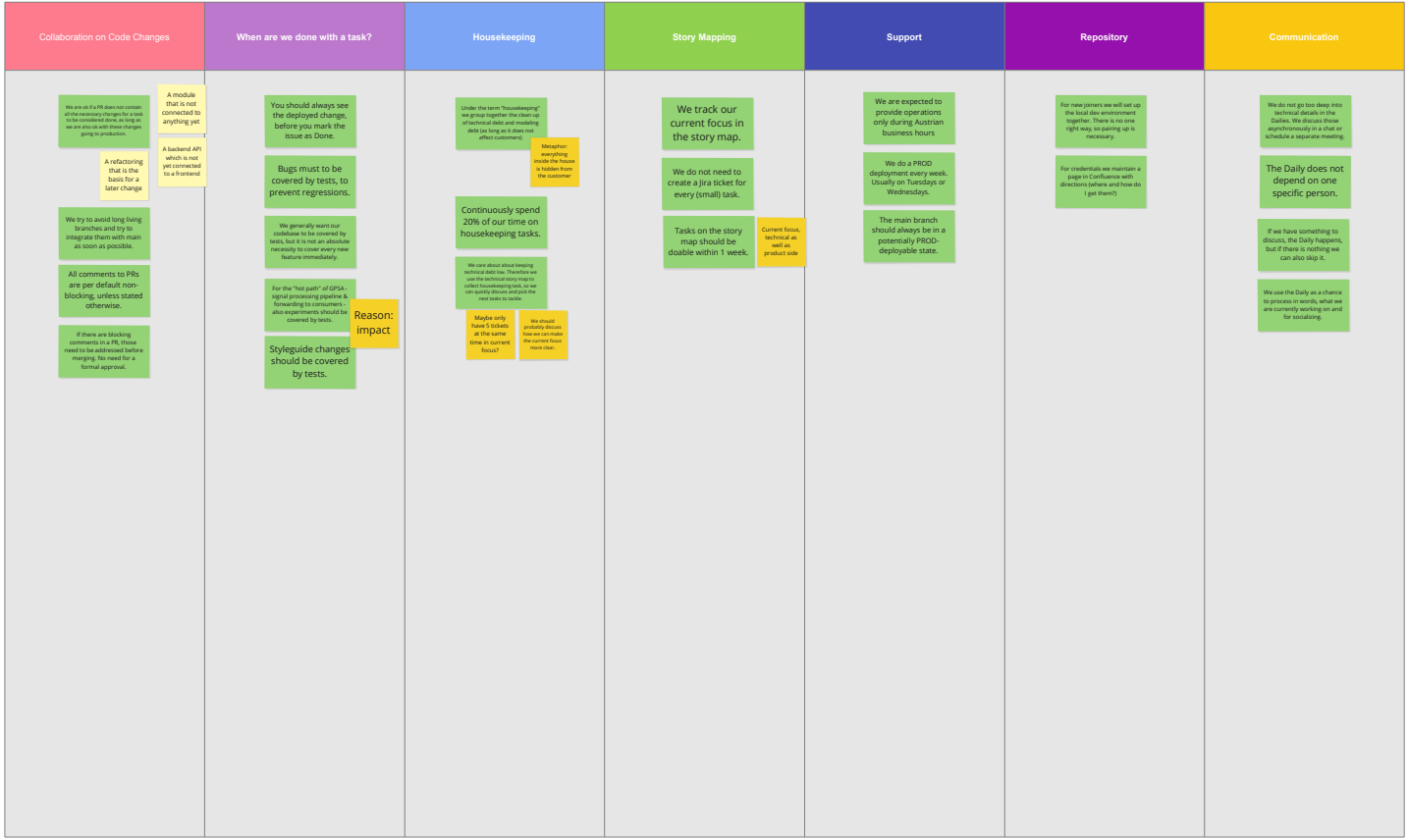
Automated Deployment

Domain Monitoring

Cost Labeling

Alerting

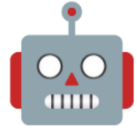

Alerting
GCP Alert
Teams Web Hook
Triggers a
Cloud-Function Web Hook
Translates & Authorizes to
PROD Technical Alert Yesterday 8:35 PM
GCP Incident: "Some PubSub-Subscriptions are not processed Unacked Messages in subscription alert for [redacted] gcf-forwardForReceiveSignals-europe-west1-aggregation-processed-signals resolved."
Started at Tue, 29 Aug 2023 18:17:09 GMT
Ended at Tue, 29 Aug 2023 18:35:07 GMT
Some PubSub-Subscriptions are not processed Unacked Messages in subscription alert for [redacted] gcf-forwardForReceiveSignals-europe-west1-aggregation-processed-signals resolved.
closed incident from policy Some PubSub-Subscriptions are not processed!
Indcident ID: 0.n1l5qli3gj5u
State: closed
Resource: {"labels":{"project_id":"[redacted]"subscription_id":"gcf-forwardForReceiveSignals-
aggregation-processed-signals"},"type":"timeseries_query"}
Metric: {"displayName":, "labels":, "type":}
Metadata: {"system_labels": "user_labels":}
See more
View Incident in GCP
Reply


Collaboration on Code Changes
We are ok if a PR does not contain all the necessary changes for a task to be considered done, as long as we are also ok with these changes going to production.
A module that is not connected to anything yet
A refactoring that is the basis for a later change
A backend API which is not yet connected to a frontend
We try to avoid long living branches and try to integrate them with main as soon as possible.
All comments to PRs are per default non-blocking, unless stated otherwise.
If there are blocking comments in a PR, those need to be addressed before merging. No need for a formal approval.
When are we done with a task?
You should always see the deployed change, before you mark the issue as Done.
Bugs must to be covered by tests, to prevent regressions.
We generally want our codebase to be covered by tests, but it is not an absolute necessity to cover every new feature immediately.
For the "hot path" of GPSA - signal processing pipeline & forwarding to consumers - also experiments should be covered by tests.
Reason: impact
Styleguide changes should be covered by tests.
Housekeeping
Under the term "housekeeping" we group together the clean up of technical debt and modeling debt (as long as it does not affect customers)
Metaphor: everything inside the house is hidden from the customer
Continuously spend 20% of our time on housekeeping tasks.
We care about about keeping technical debt low. Therefore we use the technical story map to collect housekeeping task, so we can quickly discuss and pick the next tasks to tackle.
Maybe only have 5 tickets at the same time in current focus?
We should probably discuss how we can make the current focus more clear.
Story Mapping
We track our current focus in the story map.
We do not need to create a Jira ticket for every (small) task.
Tasks on the story map should be doable within 1 week.
Current focus, technical as well as product side
Support
We are expected to provide operations only during Austrian business hours
We do a PROD deployment every week. Usually on Tuesdays or Wednesdays.
The main branch should always be in a potentially PROD-deployable state.
Repository
For new joiners we will set up the local dev environment together. There is no one right way, so pairing up is necessary.
For credentials we maintain a page in Confluence with directions (where and how do I get them?)
Communication
We do not go too deep into technical details in the Dailies. We discuss those asynchronously in a chat or schedule a separate meeting.
The Daily does not depend on one specific person.
If we have something to discuss, the Daily happens, but if there is nothing we can also skip it.
We use the Daily as a chance to process in words, what we are currently working on and for socializing.

Local development easy for new team members

Way of collaboration more explicit

Contract versioning is hard



IaC instead of deployment docs

Hardcoding of inter-BC infra references

Less worrying about PROD due to alerting/domain monitoring

Manual budget checks are a pain

Late

Product

Even more customers

Automate Budget Controlling

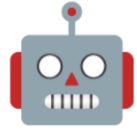
Tech

GCP discourages secrets via env vars

Desire for SLAs

Team

Growing complexity of BCs



Dependencies as Packages

Custom Contract Serialization

Improve IaC structure

Budget Alerts

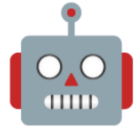
Feature Toggles

SLIs/SLOs

Easier contract consumption in JS

Decoupled library update / library use

Contract-first is the sane way



IaC reflects BC structure

Alerts give overview & safety

Easier and faster PR merging

SLI tracks main domain feature

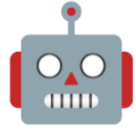
Maybe move away from monorepo?

Packaged Libraries

Split domain due to size?

Move to Kubernetes? 💀

Keep >1 backend languages?



Move to a build-once strategy?

Use tooling (e.g. release please) for releases

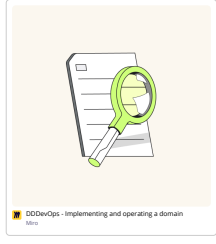
Start working with error budgets?


Domain
Context matters!
B2C vs B2B
Green field vs legacy
Team composition
Compliance requirements

References

This presentation

Juggling 300 integrations



GitHub Repository

Google SRE Book