Why you, as a tester, should take note of platform engineering

Abby Bangser
@a_bangser
she/her

Quality is part of a larger ecosystem.
Platform engineering is just a newer addition.

Some other reasons you may be interested in...

1. The selfish reason
   - How platforms came to be
   - Examples of platform use cases for QAs
2. The opportunistic reason
   - Inner-sourced platforms giving everyone a voice
   - Service offerings and service configurations
3. The career path reason
   - Developing (and testing) a platform

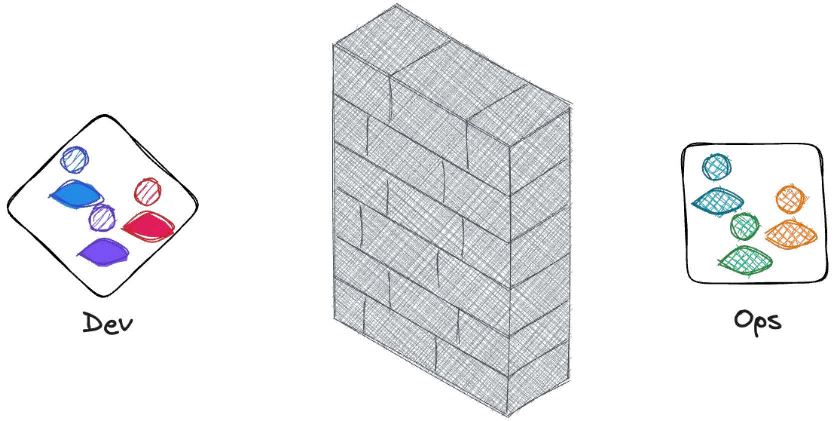
@a_bangser

The selfish reason

Internal platforms determine ease of access to necessary tooling

@a_bangser

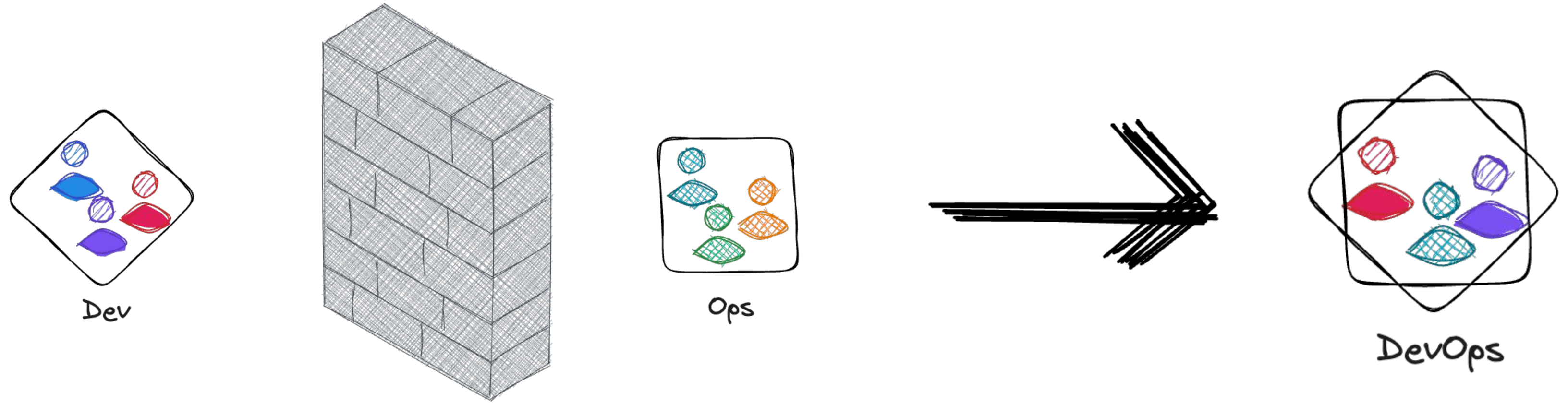
The history of Dev/Ops



Want a new tool or environment?

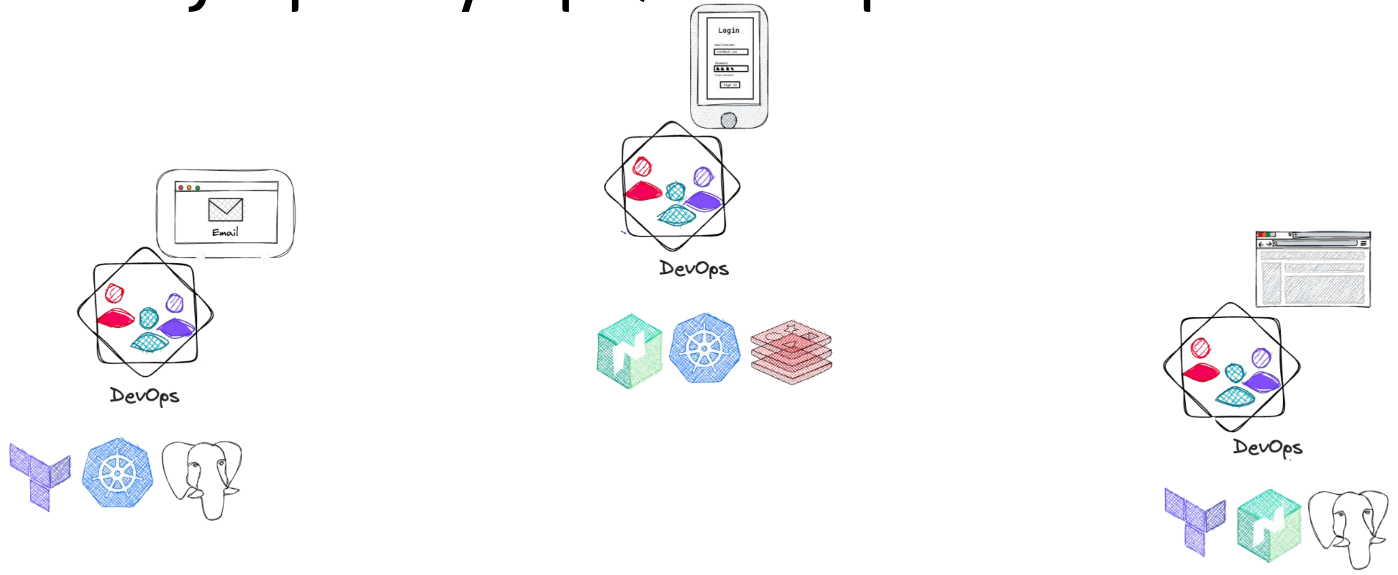
Make a request to a backlog with an indeterminate amount of time to resolve

The history of Dev/Ops, DevOps



@a_bangser

The history of Dev/Ops, DevOps

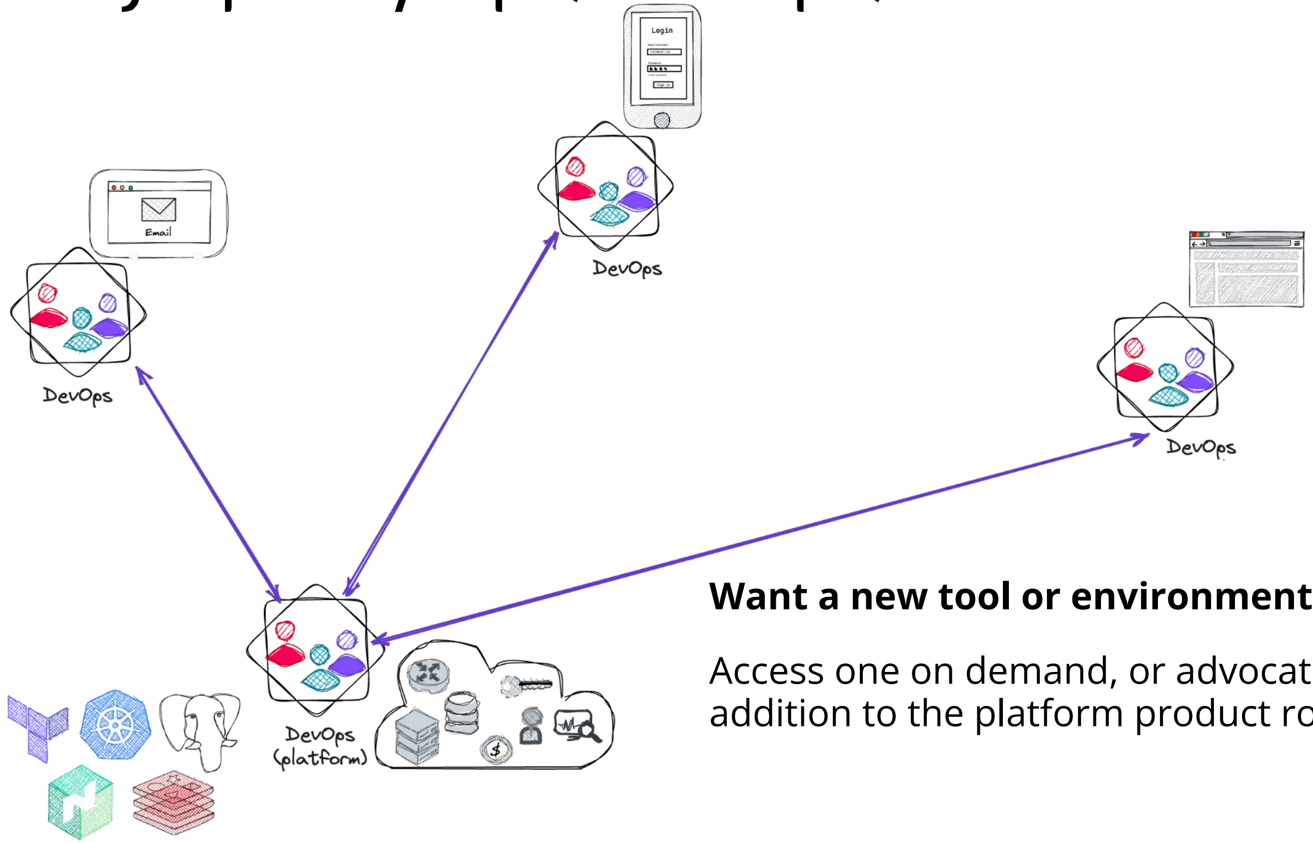


Want a new tool or environment?

Climb a learning curve for building it yourself, or do without

@a_bangser

The history of Dev/Ops, DevOps, and now Platforms



Want a new tool or environment?

Access one on demand, or advocate for an addition to the platform product roadmap

@a_bangser

A platform definition

A platform is a product that serves or enables other products or services.

Platforms (in the context of digital business) exist at many levels. They range from high-level platforms that enable a platform business model to ***low-level platforms that provide a collection of business and/or technology capabilities that other products or services consume to deliver their own business capabilities.***

- Gartner

https://www.gartner.com/en/information-technology/glossary/platform-digital-business

My experience building internal platforms

- I have been on a team specifically called "platform" since 2018
- Typical work would be:
    - Upgrading the server versions
    - Creating databases using Infrastructure-as-Code (IaC)
    - Debugging and extending delivery pipelines
    - Researching new tools (e.g. DynamoDB on Amazon)

User experience with the Platforms I built

- Some actions were on demand, e.g. local test environments
- Many actions were self-service code, e.g. terraform modules for a DB
- Most actions were completely manual, e.g. secrets rotation
- None of the requests were optional

Revisiting the platform definition

A platform is a product that serves or enables other products or services.

Platforms (in the context of digital business) exist at many levels. They range from high-level platforms that enable a platform business model to ***low-level platforms that provide a collection of business and/or technology capabilities that other products or services consume to deliver their own business capabilities.***

- Gartner

https://www.gartner.com/en/information-technology/glossary/platform-digital-business

@a_bangser

Extending to a platform *engineering* definition

A platform is a product that serves or enables other products or services.

Platforms (in the context of digital business) exist at many levels. They range from high-level platforms that enable a platform business model to low-level platforms that provide a collection of business and/or technology capabilities that other products or services consume to deliver their own business capabilities.

- Gartner

https://www.gartner.com/en/information-technology/glossary/platform-digital-business

The goal is a ***frictionless, self-service developer experience*** that offers the right capabilities to enable developers and others to produce valuable software with as little overhead as possible.

The platform should **increase developer productivity, along with reducing the cognitive load**.

- also Gartner

https://www.gartner.com/en/articles/what-is-platform-engineering

@a_bangser

Arguably, I was building a “platform”.
You should expect a “platforms **experience**”.

Central Ops

- Team is focused on **reducing their own toil**
- Requests for an existing service are done via **a queue (ticket or pull request)**
- Requests for a new offering are treated as **a backlog item**
- Success is **stable infrastructure**

Platform Engineering

- Team is focused on **improving application developer workflows**
- Requests for an existing service are done via **a self-serve interface**
- Requests for a new offering are treated as **a feature request**
- Success is **increased business value**

@a_bangser

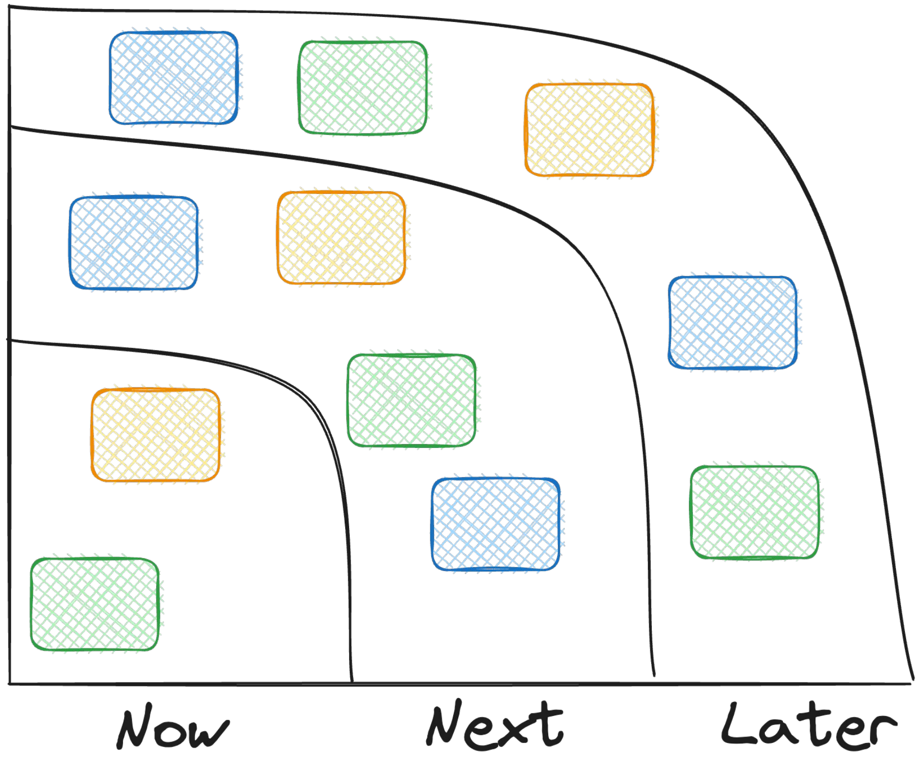
Platforms are software, they need engineering

- User needs are explored and prioritised (Platform-as-a-Product!)
- The team must build and run their own software (apply DevOps!)
- Clear boundaries supports a clear ownership model (domain modeling)

Be selfish, ask for what you need.
Demand a better **experience**.
Expect **self-service** and on-demand offerings.

Things you can ask for

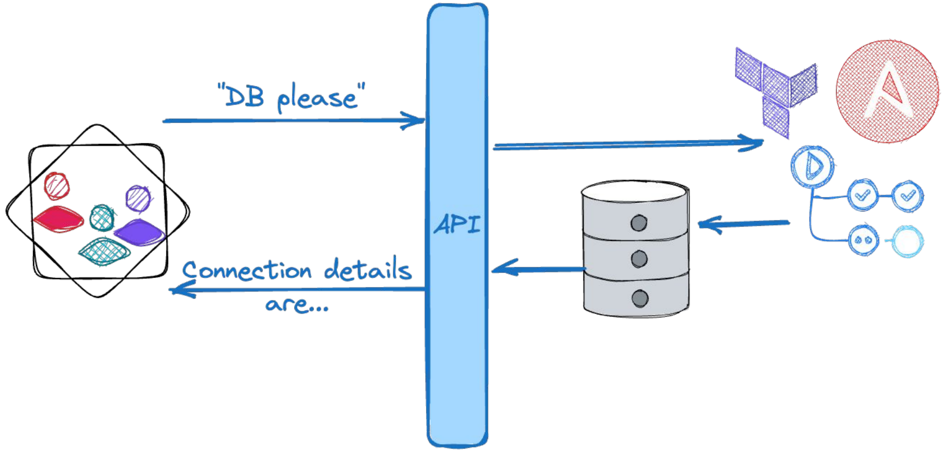
- Visibility into the platform roadmap



@a_bangser

Things you can ask for

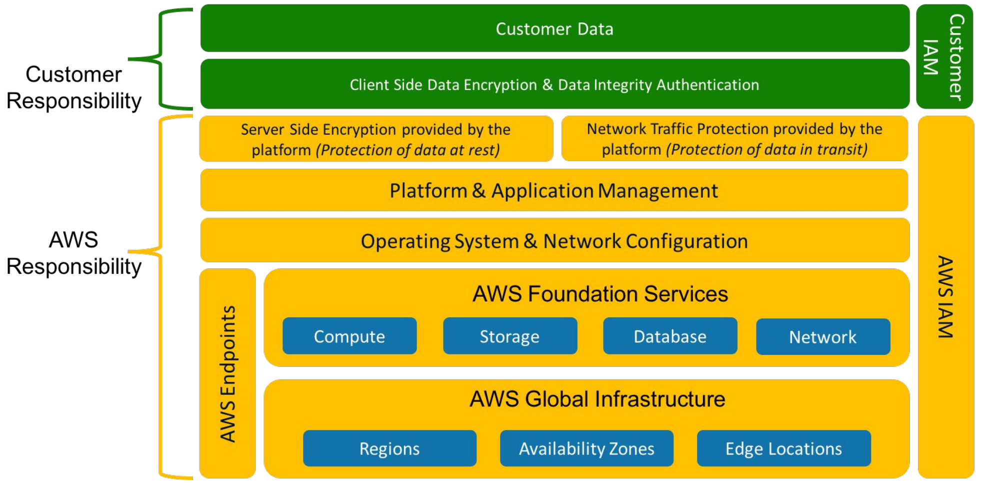
- Visibility into the platform roadmap
- Common requests should be made available self-service and on-demand



@a_bangser

Things you can ask for

- Visibility into the platform roadmap
- Common requests should be made available self-service and on-demand
- Shared responsibility model should be clearly defined for all offerings



Customer Responsibility
AWS Responsibility
Customer Data
Client Side Data Encryption & Data Integrity Authentication
Customer IAM
Server Side Encryption provided by the platform *(Protection of data at rest)*
Network Traffic Protection provided by the platform *(Protection of data in transit)*
Platform & Application Management
Operating System & Network Configuration
AWS IAM
AWS Endpoints
AWS Foundation Services
Compute
Storage
Database
Network
AWS Global Infrastructure
Regions
Availability Zones
Edge Locations

https://cloudacademy.com/blog/aws-shared-responsibility-model-security/

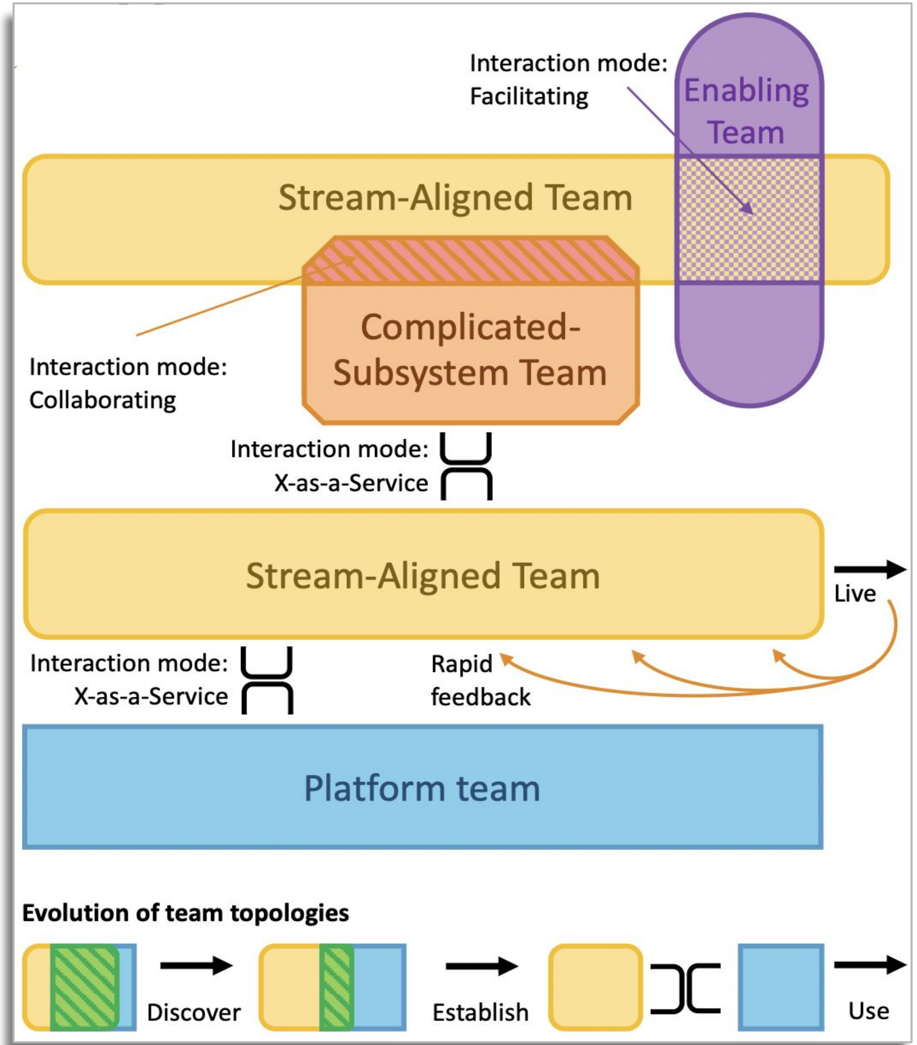
@a_bangser

The opportunistic reason

Platforms enable dissemination of quality principles, practices, and tools

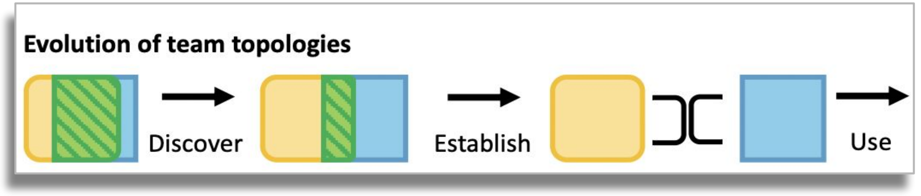
@a_bangser

Team Topologies has provided a spotlight on platforms at least in part due to the focus on **interaction models**



@a_bangser

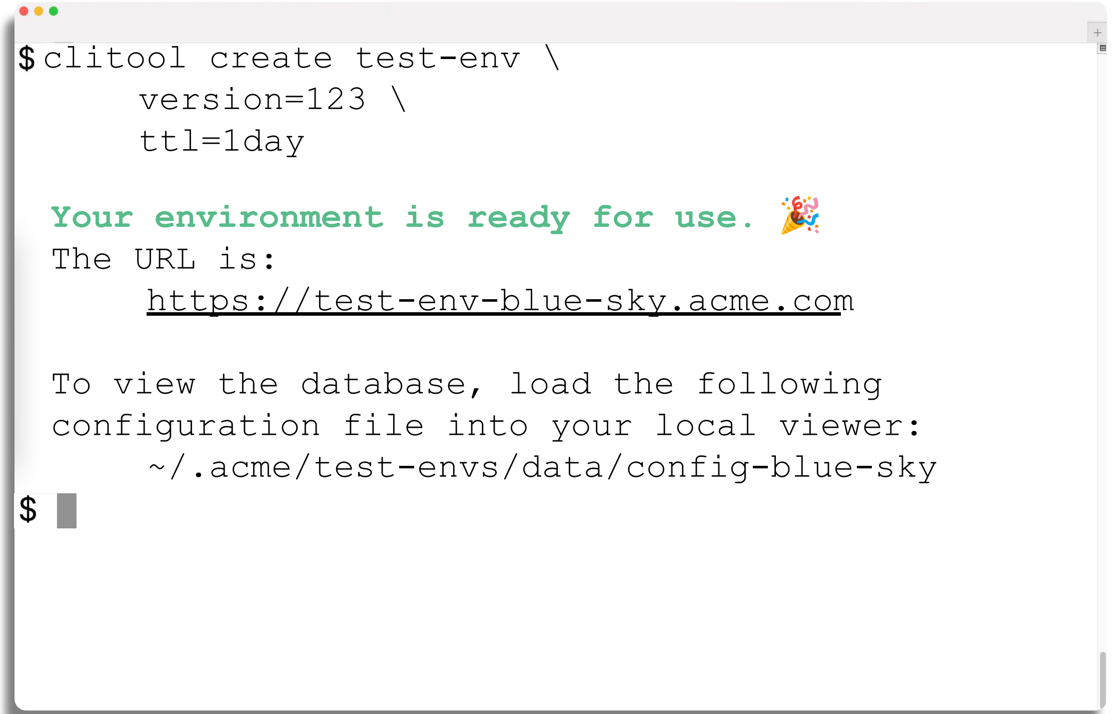
Quality has been on this evolution for years



- Do the testing yourself, find and understand the risk
- Identify the opportunities for improving quality and design a goal
- Work with the team to execute a plan to reach that goal
- Build tools and processes to support the quality process independent of doing everything yourself

@a_bangser

Example platform offering:
Test environment-as-a-Service



```
$ clitool create test-env \
      version=123 \
      ttl=1day

  Your environment is ready for use. 🎉
  The URL is:
       https://test-env-blue-sky.acme.com

  To view the database, load the following
  configuration file into your local viewer:
       ~/.acme/test-envs/data/config-blue-sky
$
```

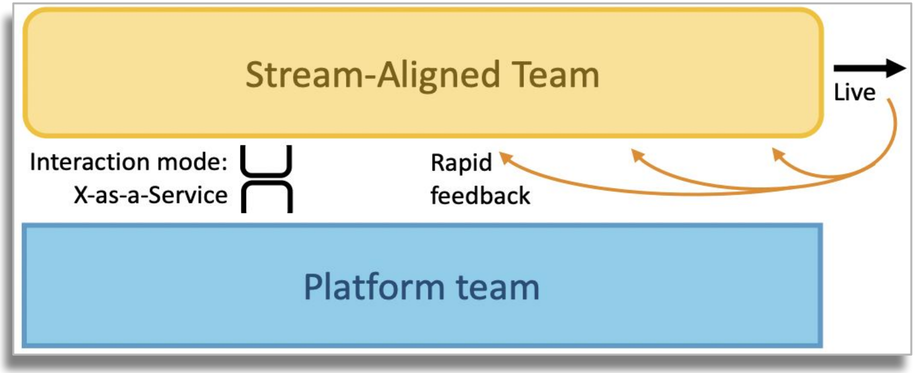
@a_bangser

Be **opportunistic**, disseminate your message.
Codify aspects of **quality in platform offerings**.
Offer tooling that will **enable quality**.

@a_bangser

Offer as-a-Service model for quality

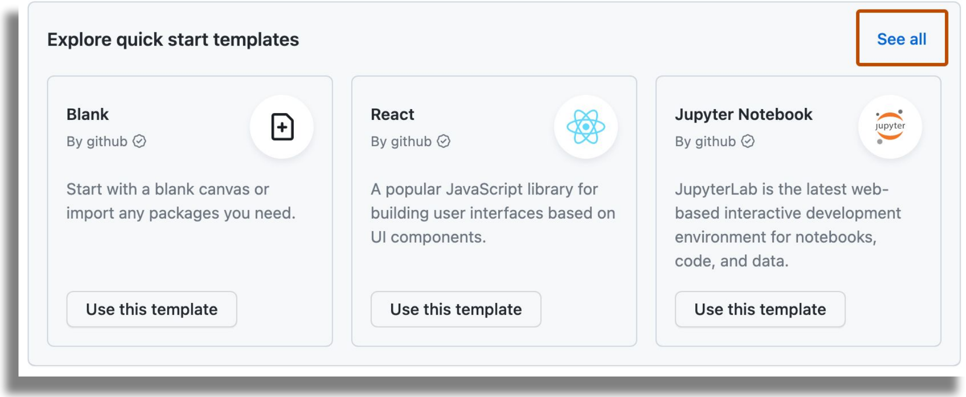
- Linting and other code quality tools
- Mutation testing
- Chaos engineering
- Observability
- Test environments



@a_bangser

But it doesn't have to be complete services

Reviewing template configurations can have an outsized impact


Explore quick start templates

See all

Blank
By github

Start with a blank canvas or import any packages you need.

Use this template

React
By github

A popular JavaScript library for building user interfaces based on UI components.

Use this template

Jupyter Notebook
By github

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data.

Use this template

@a_bangser

And evolve offerings, e.g. test env-as-a-Service

- The data sanitation strategy
- Making sure telemetry is provided with all test environments
- Providing a way to quickly raise issues for a specific environment
- ...

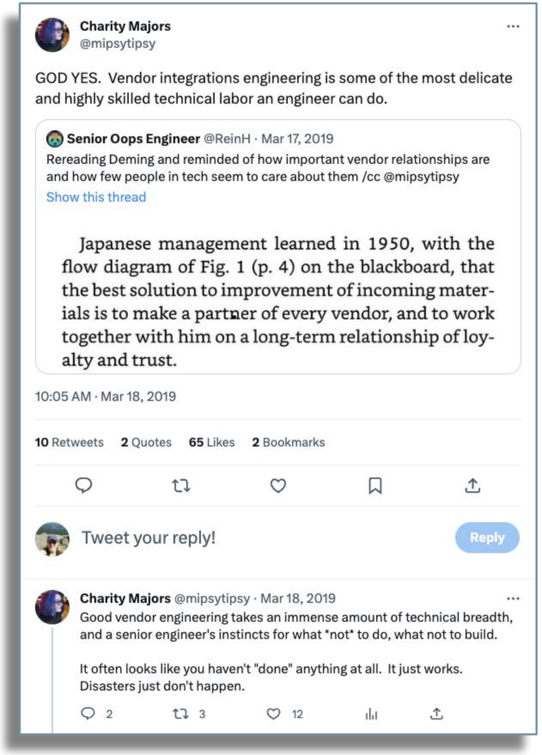
@a_bangser

The career path reason

Platforms are a growing domain and they require testing

@a_bangser

Platform engineering may be vendor engineering, but that doesn't reduce its complexity or value



Charity Majors
@mipsytipsy

GOD YES. Vendor integrations engineering is some of the most delicate and highly skilled technical labor an engineer can do.

Senior Oops Engineer @ReinH · Mar 17, 2019
Rereading Deming and reminded of how important vendor relationships are and how few people in tech seem to care about them /cc @mipsytipsy
Show this thread

Japanese management learned in 1950, with the flow diagram of Fig. 1 (p. 4) on the blackboard, that the best solution to improvement of incoming materials is to make a partner of every vendor, and to work together with him on a long-term relationship of loyalty and trust.

10:05 AM · Mar 18, 2019

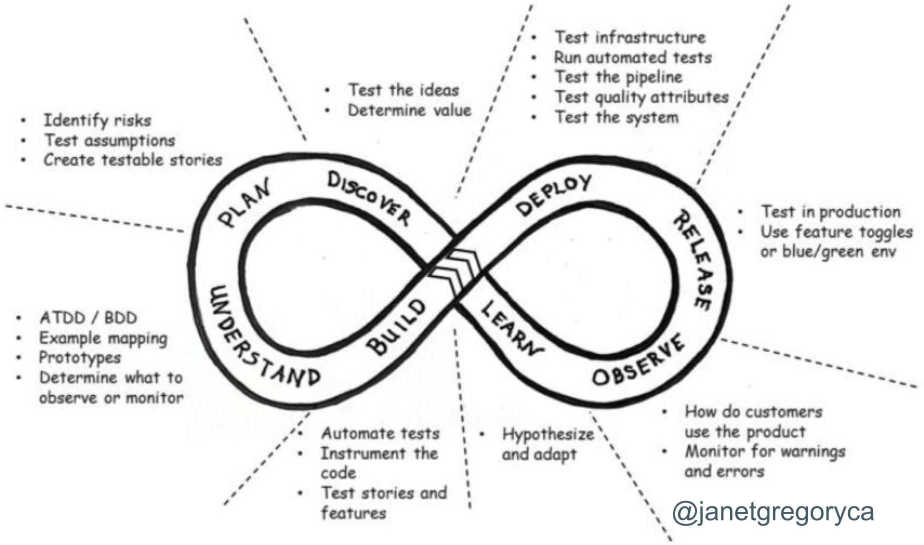
10 Retweets 2 Quotes 65 Likes 2 Bookmarks

Charity Majors @mipsytipsy · Mar 18, 2019
Good vendor engineering takes an immense amount of technical breadth, and a senior engineer's instincts for what *not* to do, what not to build.

It often looks like you haven't "done" anything at all. It just works. Disasters just don't happen.

2 3 12

The same delivery cycle



requires the same holistic testing

@a_bangser

Platforms APIs require testing

- User research to define next steps for the platform
- Unit testing of the business logic behind the API
- UAT testing of the API experience
- Production testing of the final product

Broaden your career options.
Pioneer in a **relatively unexplored domain**.
Define what quality means for internal platforms.

@a_bangser

Upskill in platforms

- Explore infrastructure test automation
- Get comfortable on the command line
- Join communities that are talking about this
- Build your own helper tools

Focus on your platform

- Do (even informal) code reviews of platform code
- Think like a user researcher for internal platforms
- Explore the backlog and other team charters

Summary

Platform engineering is not testing.

But quality is part of a bigger ecosystem and depends on platforms.

This is "just another hype cycle".

But with every hype cycle there are nuggets of greatness.
You can (and should) build on the progression platforms provide.

Platforms are an unexplored and (historically) underinvested in domain.

But quality professionals are made to explore and the domain is growing.

Thank you!

Reach out to learn more about building platforms with Kratix.io or Platform Engineering in general

abby@syntasso.io
@a_bangser