Reliability Tests All the Way Down!

Paul Balogh, @javaducky

cloud gnome

Disclaimer

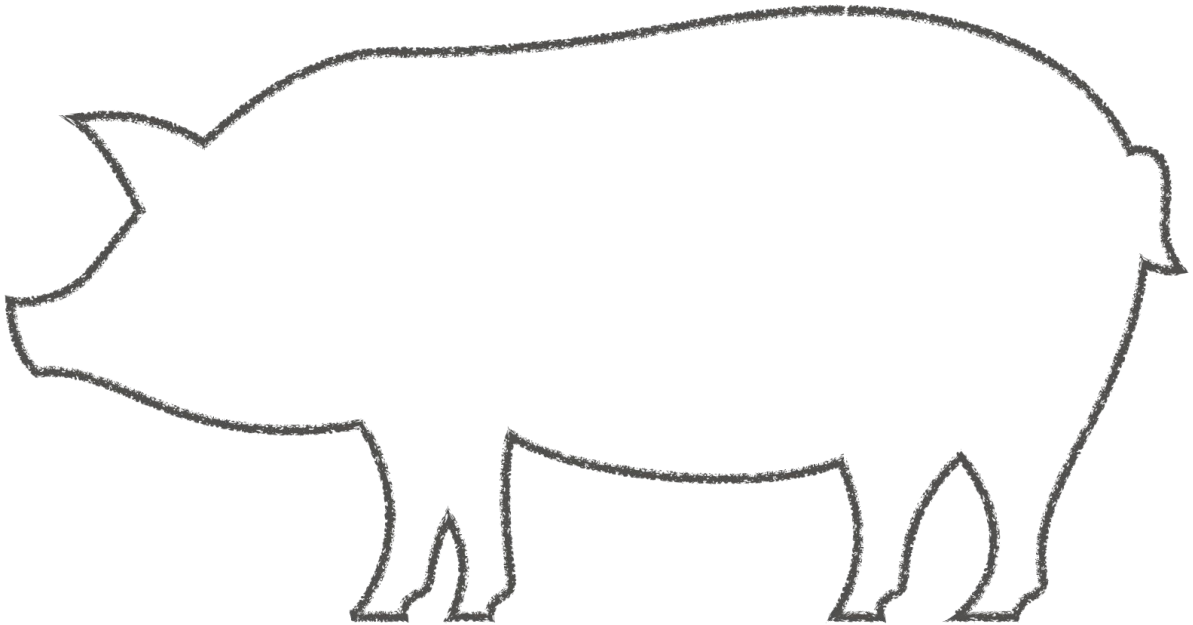
Paul is **not** here to sway you to one product over another, nor am I here to persuade you into a specific paradigm over another.

He is simply interested in **Continuous Improvement** of the software we create.

Act I

When things were simple...or were they?

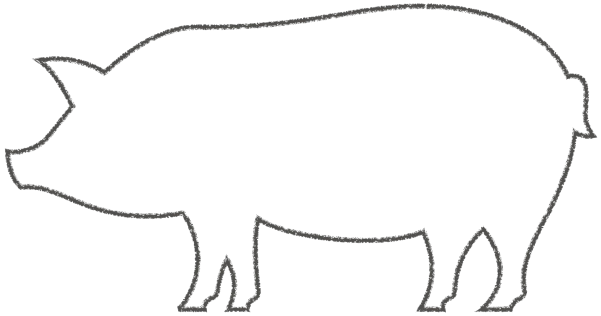
The Monolith



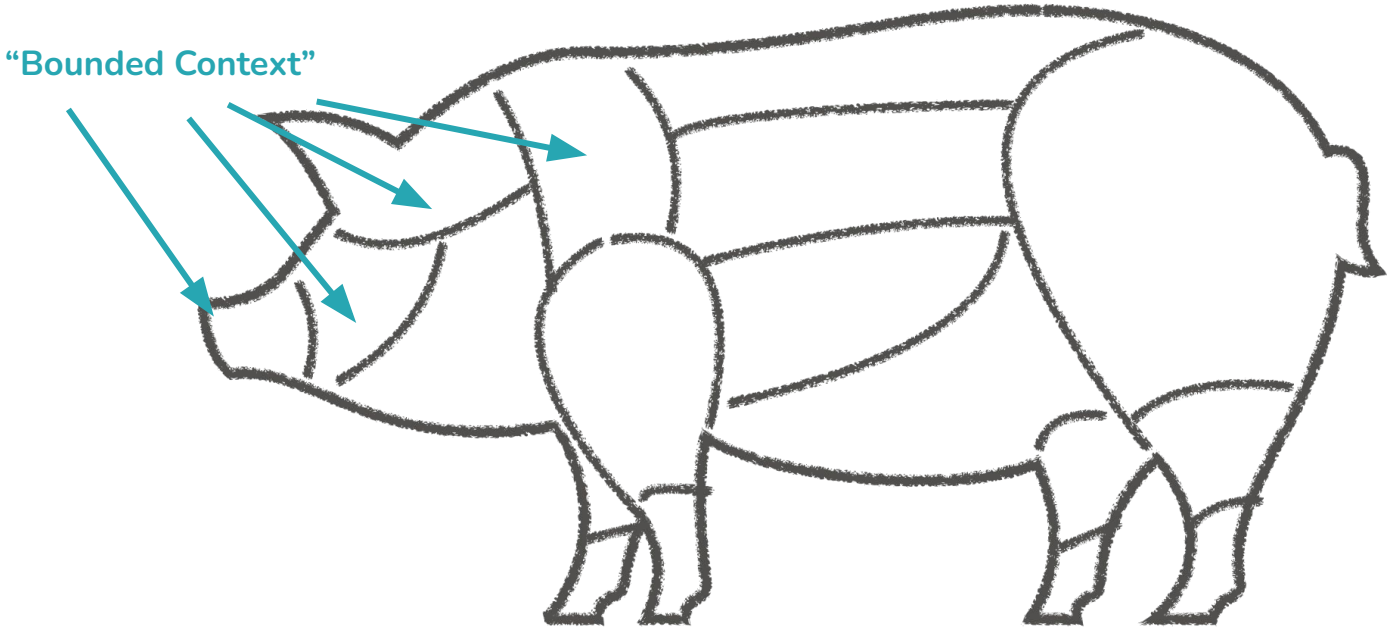
"My Cool App"

The Monolith

- Development handed-off “finished” work to QA
- Developers are introduced to Test Driven Development (TDD)
- Deployments were infrequent, unless there were bugs
- Scaling was done vertically
- We loved our **servers**!



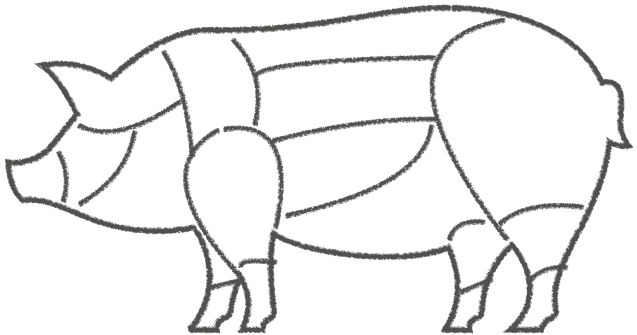
Domain-Driven Development



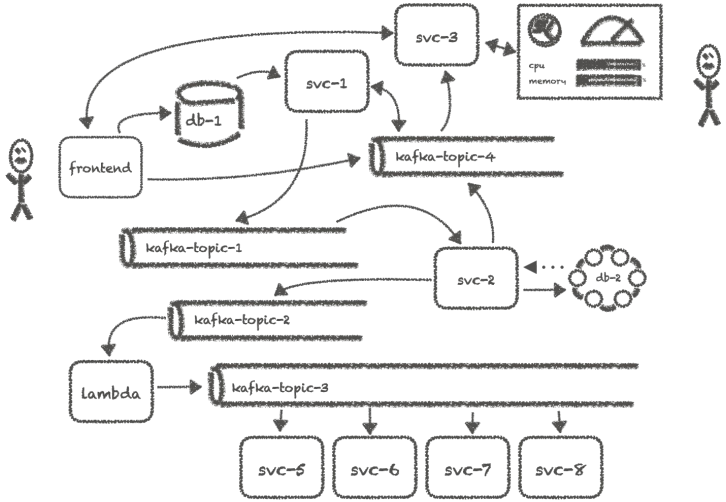
“My Cool(er?) App”

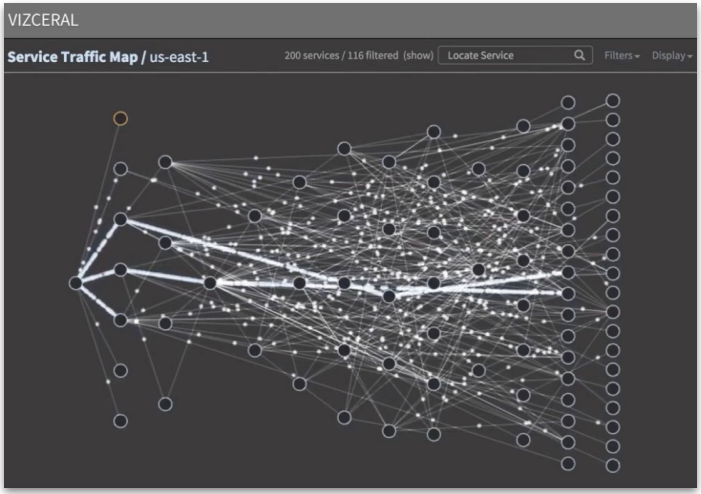
Domain-Driven Development

- Monoliths became complex and inflexible
- Breaking up things made us more agile
- Our testing practices stayed the same
- More things to test; fortunately we now have tools like *Selenium*, *JMeter*, and *Postman*
- APIs are becoming important
- We learn to love **VMs** and **Cloud**!



Microservices

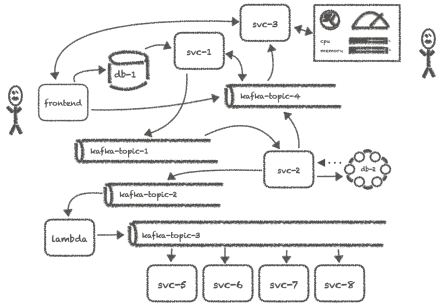




"A Netflix of our Own"

Microservices

- Finer-grained services and even serverless functions
- More teams + more services + more APIs = **more complex**
- Hyper scalability based upon metrics and requests
- Best practices outlined by the 12-factor app
- Shift-left is necessary, testers are overwhelmed
- We love our **containers** and **Kubernetes**!



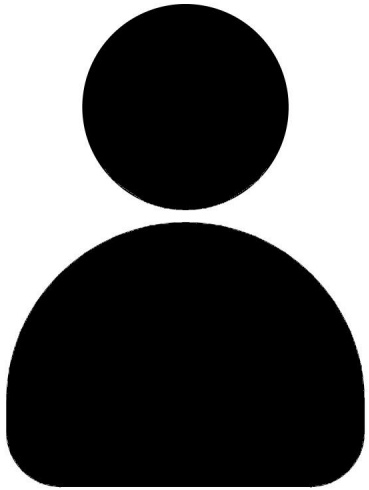
“ ”

To those of you here at **NISC**…I’m sorry.

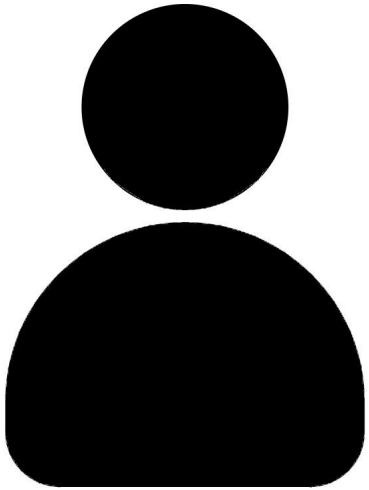
Act II

A continuous process to deal...

Our Actors

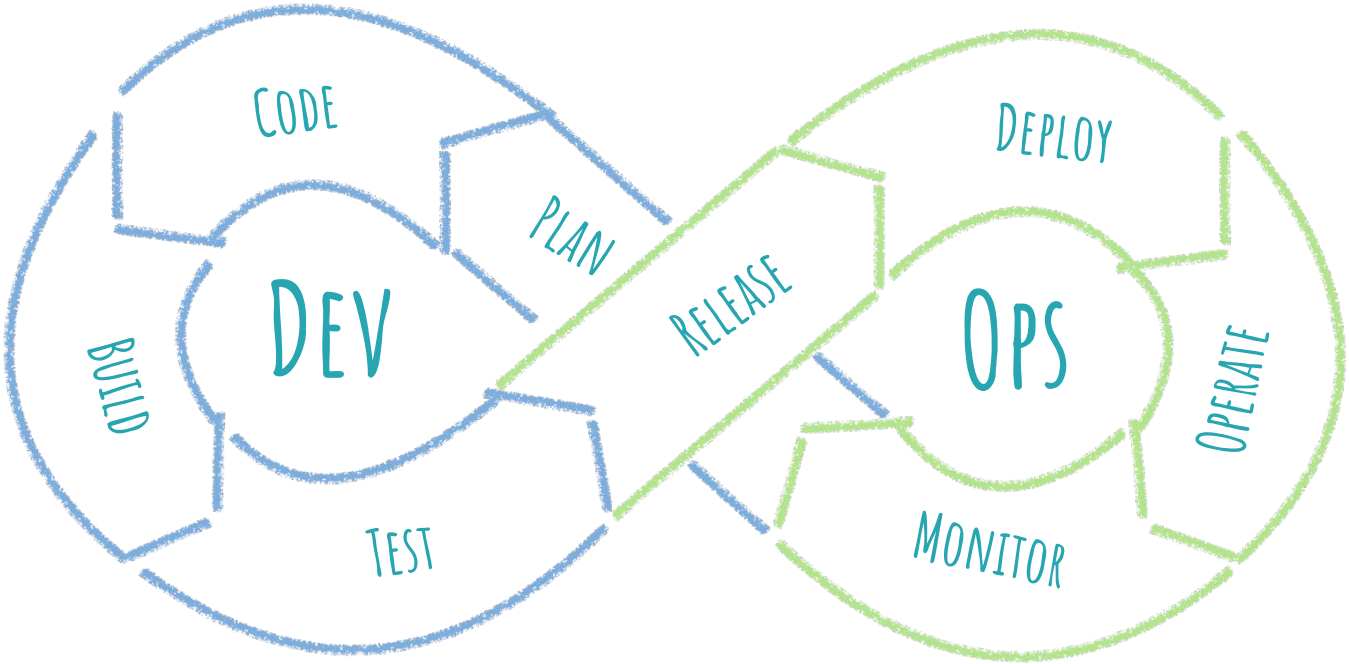


Developer

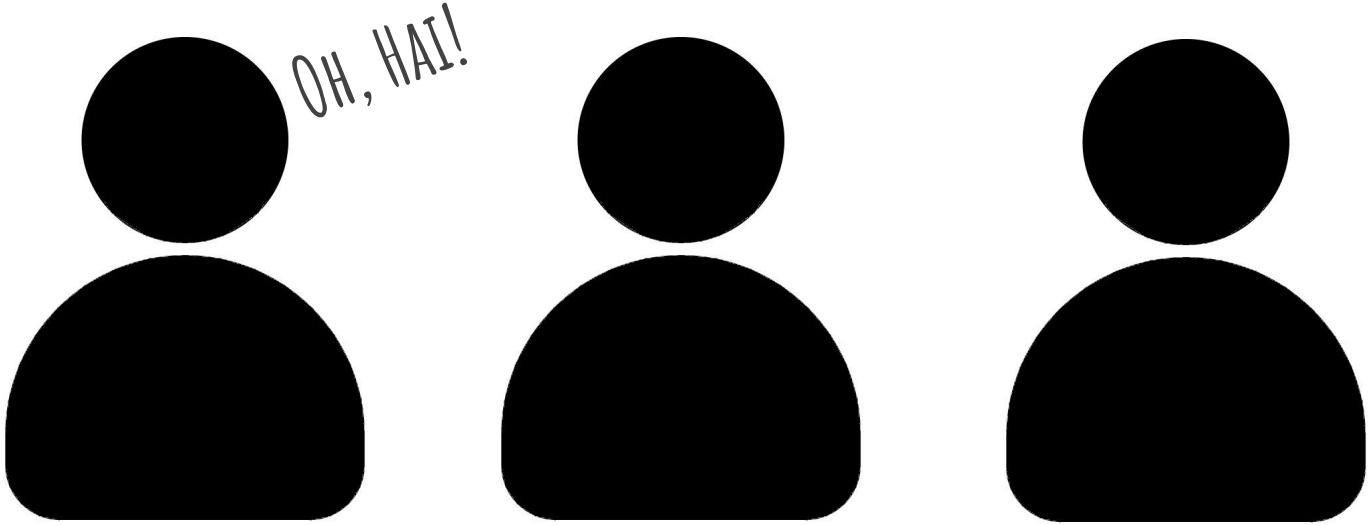


Tester

A Continuous Lifecycle



Our Actors



Site Reliability Engineer (SRE)

Developer

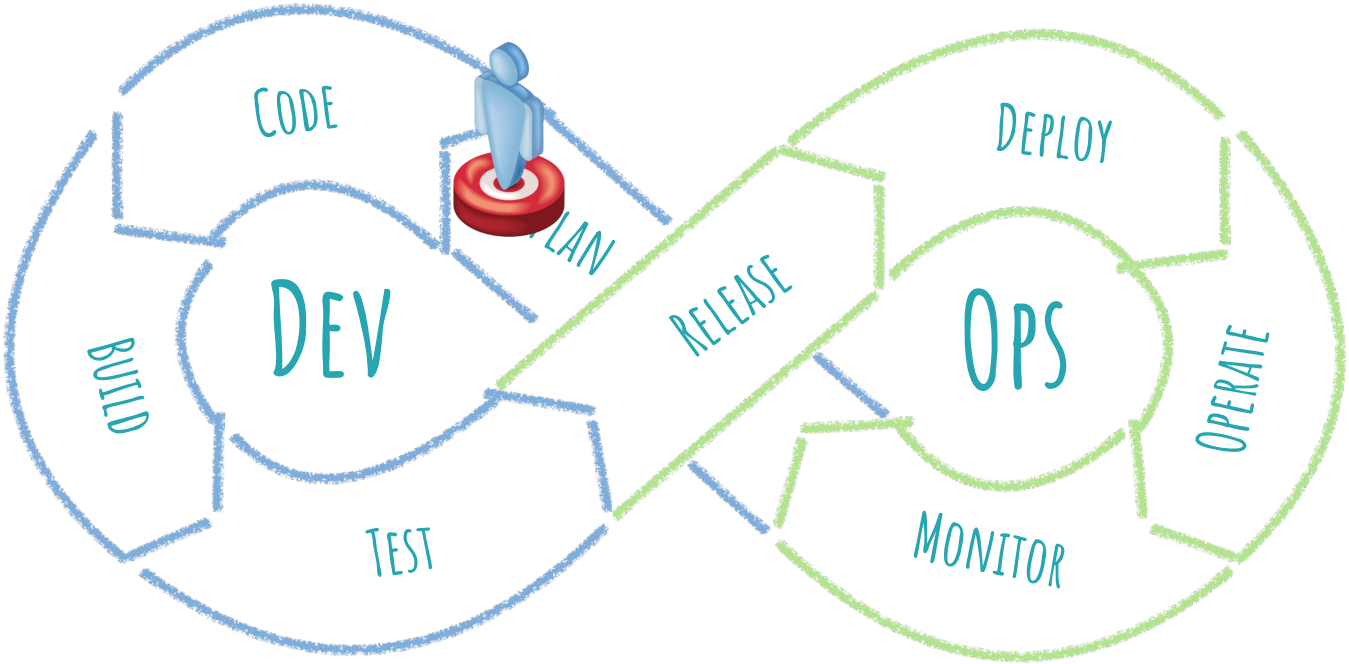
Tester

Disclaimer

I’m using somewhat generic terms for ease. In reality, these roles have many names and a single person may assume the role of multiple roles at any given time.

- SRE: anyone actively working on upkeep of infrastructure and watching for alerts.
- Developer: anyone actively working on the creation of code to be deployed as an application.
- Tester: anyone actively creating automation or directly ensuring functionality.

Plan phase



Plan phase

The beginning.

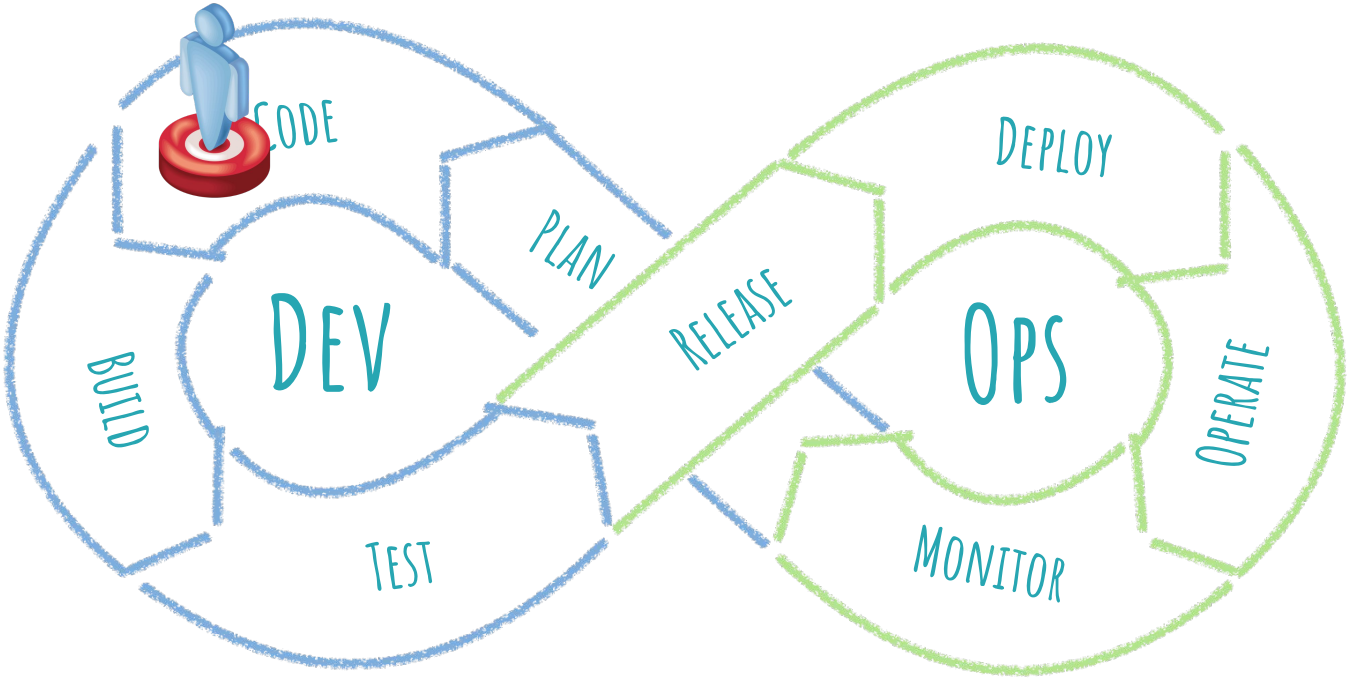
- Establish performance baselines given past experience.
- SREs and Developers agree upon stack and features.

Plan phase

The beginning.

- Establish performance base given p
  experience.
- SREs and Developers agr st k ar
  features.

Code phase


CODE
PLAN
DEV
BUILD
TEST
RELEASE
DEPLOY
OPS
OPERATE
MONITOR

Code phase

Develops develop. Testers...prepare.

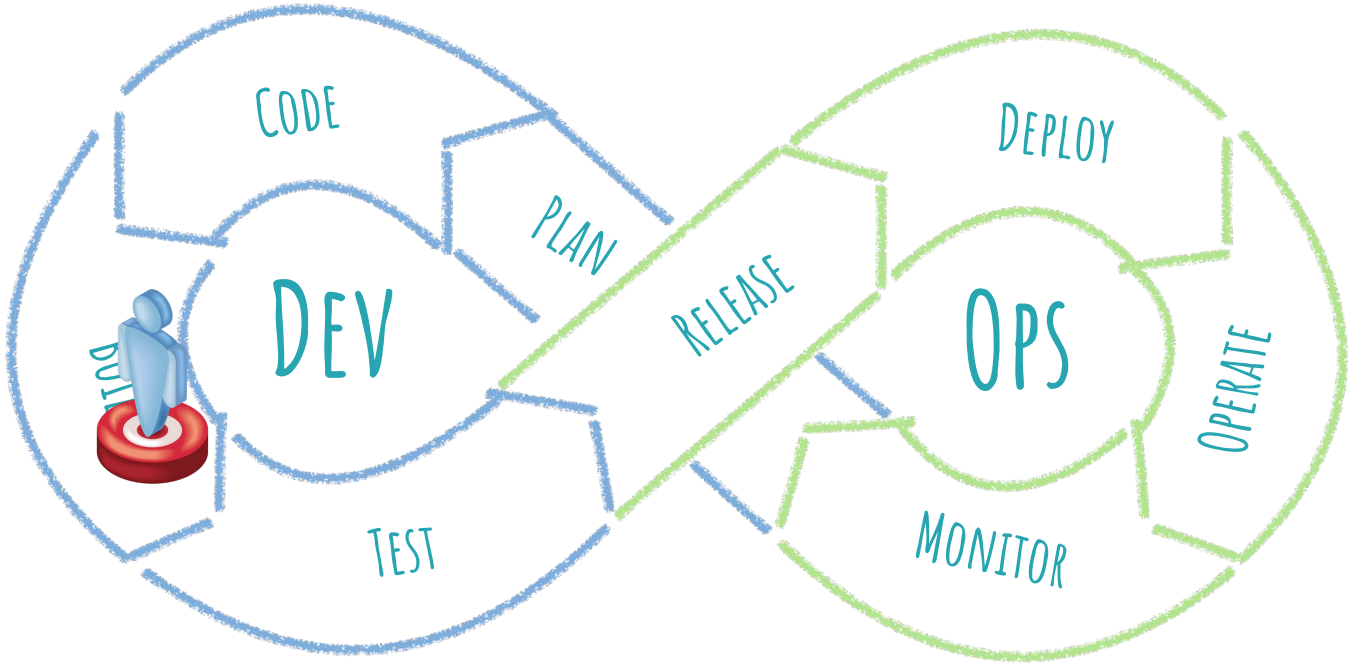
- **Developers** begin creating the features, with unit tests and integration tests.
- **Developers** include telemetry data for use in later phases.
- **Developers** ensure performant code using profiling tools.

Code phase (cont'd)

Develops develop. Testers...prepare.

- **Testers** create automation tests as Developers are creating based upon agreed specifications. These can be Contract tests and even Load tests.

Build phase

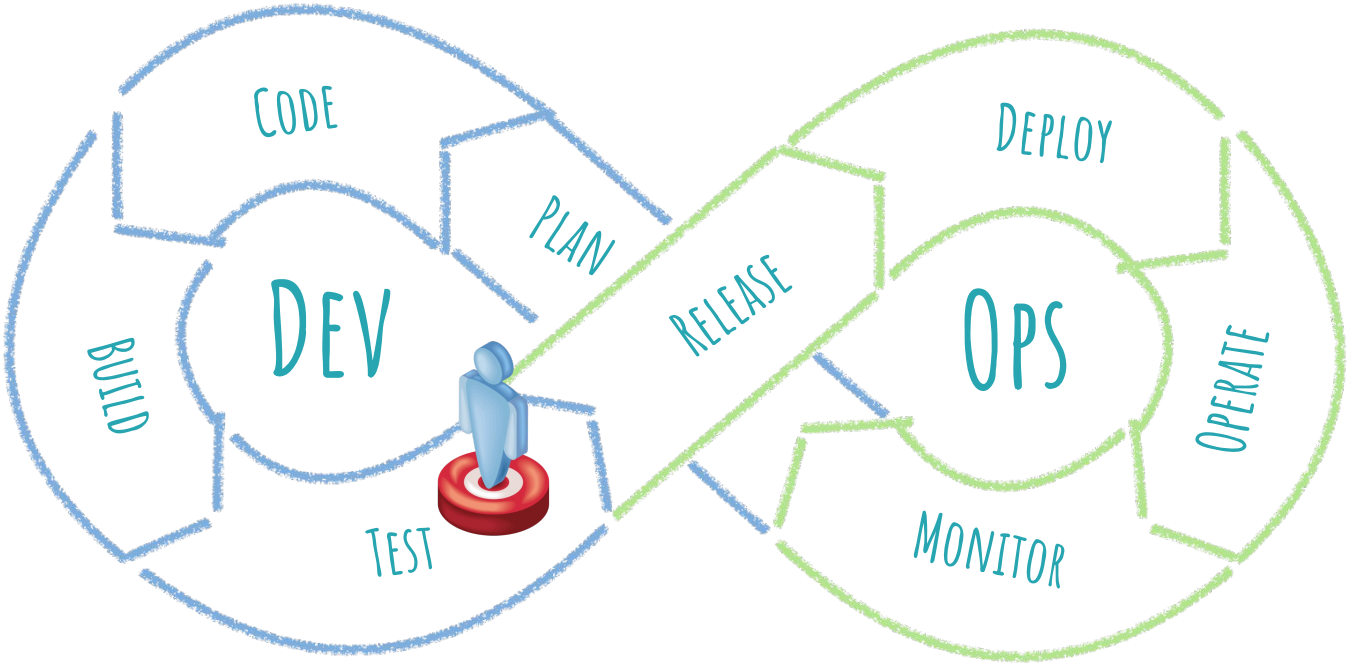

CODE
PLAN
DEV
BUILD
TEST
RELEASE
DEPLOY
OPS
OPERATE
MONITOR

Build phase

SREs buildout infrastructure.

- **SREs** create resources using tools like *Terraform* while creating tests to validate IaC.
- If new tooling was agreed upon, they build expertise in operating such resources.

Test phase

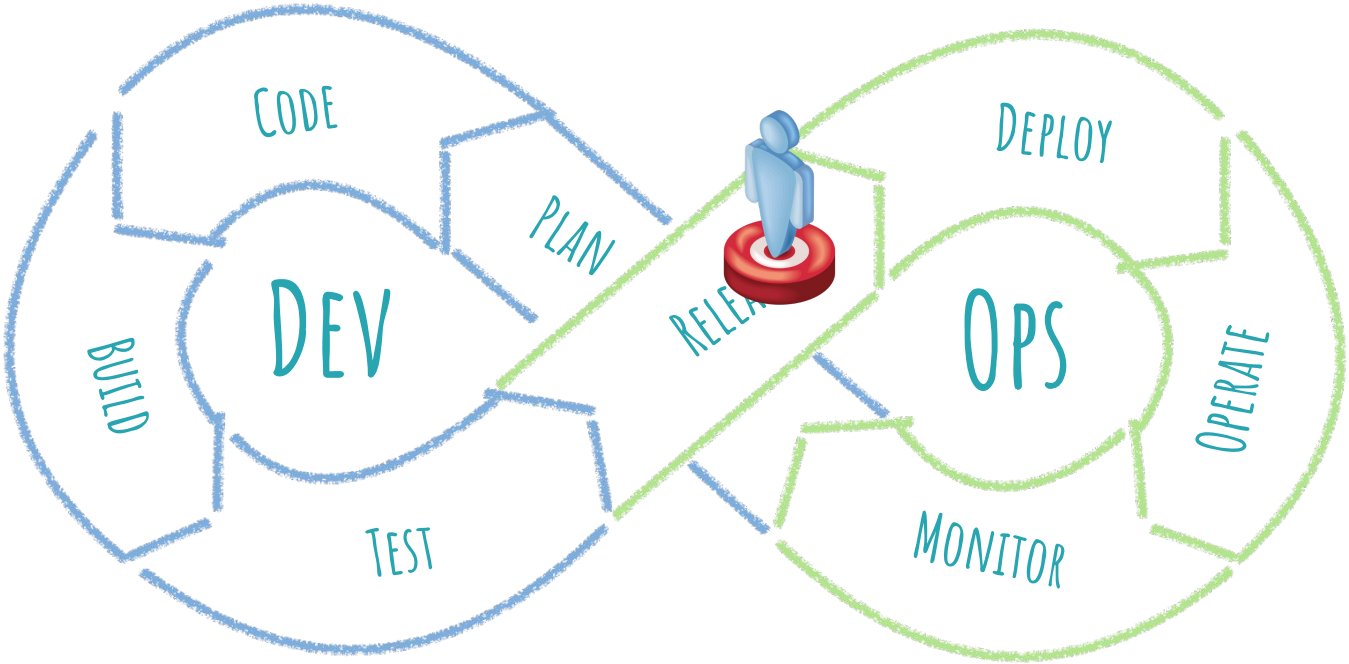

CODE
PLAN
BUILD
DEV
TEST
RELEASE
DEPLOY
OPS
OPERATE
MONITOR

Test phase

Testers test.

- Create hybrid-testing with chaos-style disruptions to dependent systems.
- Validate user experience.
- Attempt to raise security issues.
- Verify observability metrics are available.

Release phase

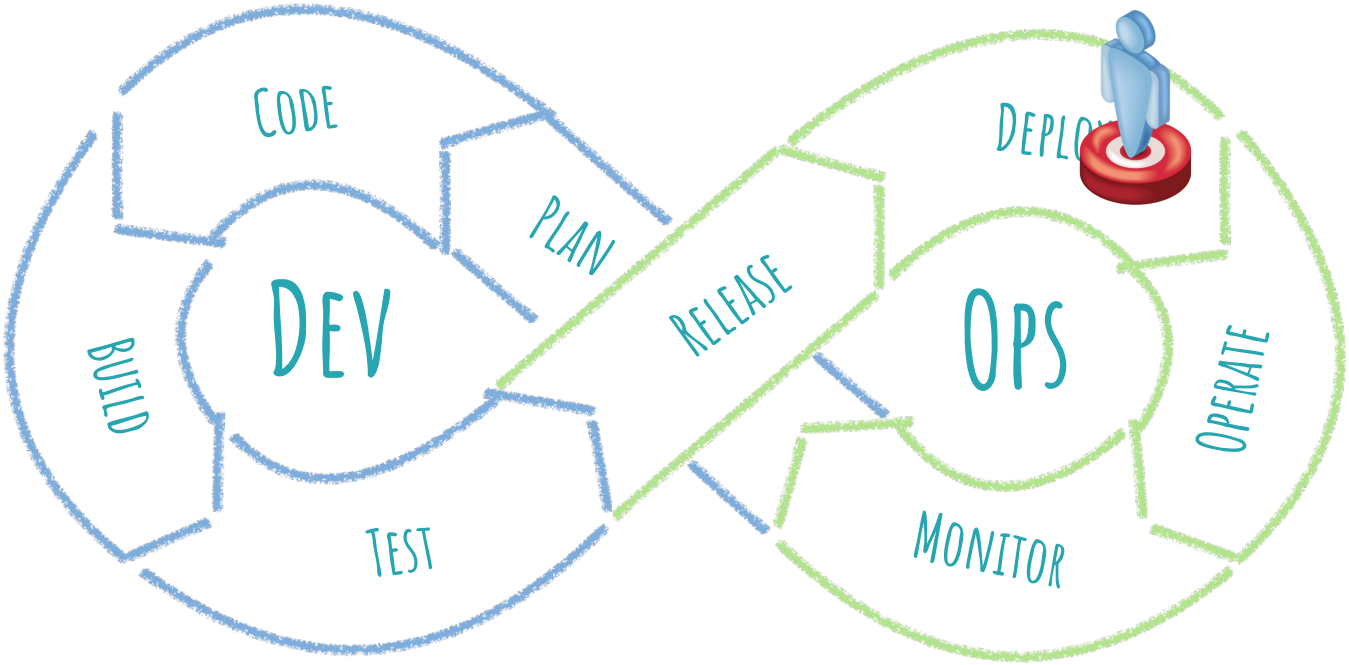


Release phase

Check quality gates and readiness.

- **Developers**, **Testers**, and **SREs** agree on ready state for application.
- Testers should confirm End-to-End testing (E2E).
- Perform load testing.

Deploy phase

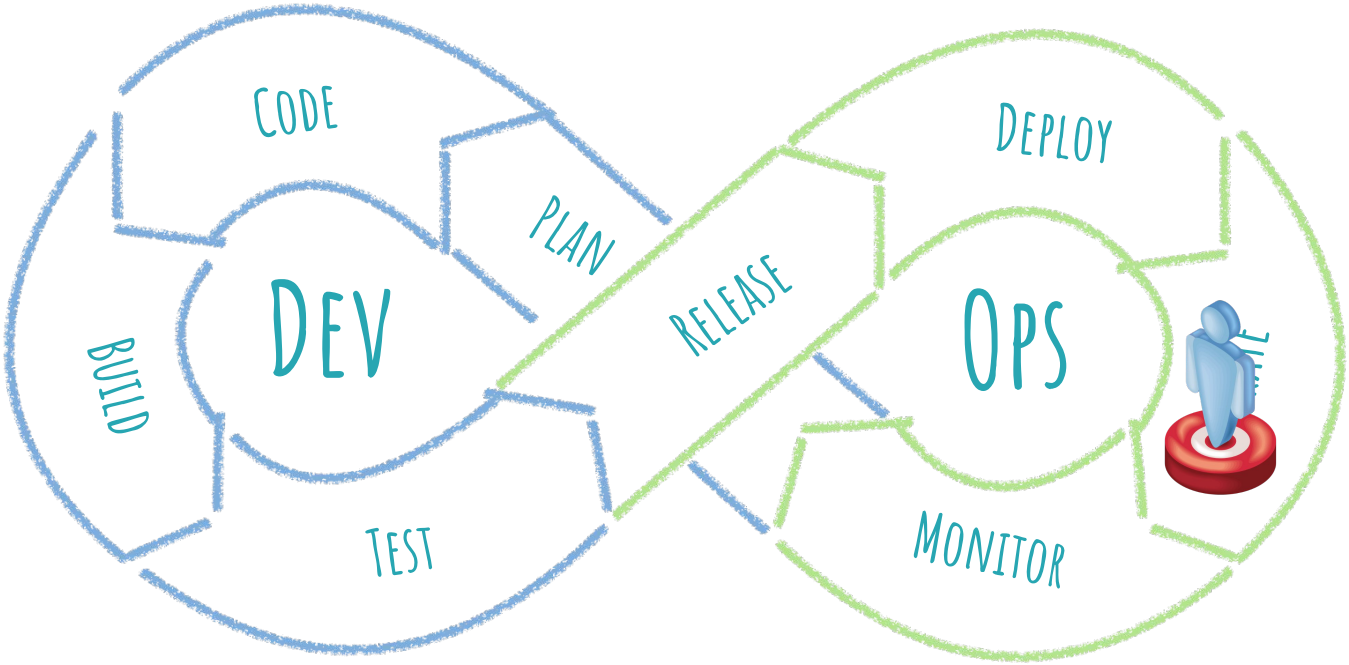


Deploy phase

Thrusters engaged.

- **SREs** utilize metrics-based quality gates with Canary deployments, or use Blue/Green deployment.

Operate phase

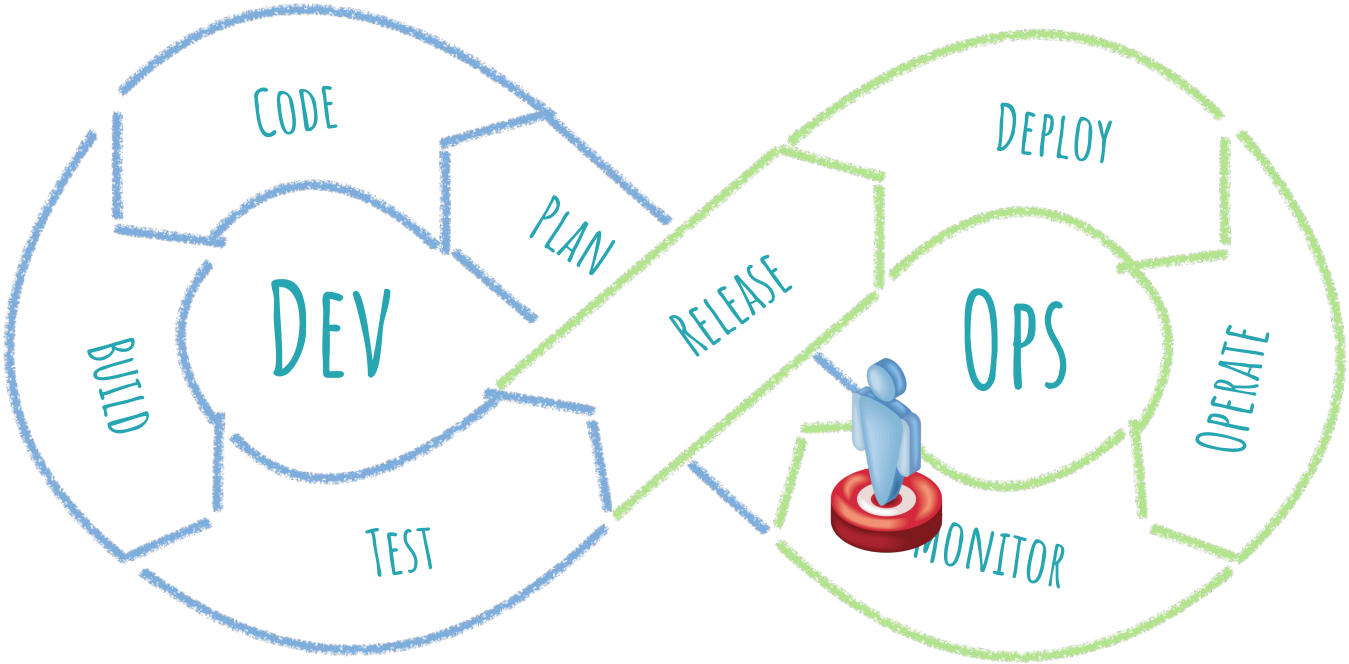


Operate phase

Keeping the wheels moving.

- **SREs** and **Testers** conduct Chaos experiments.
- **SREs** watch costs for compute.

Monitor phase

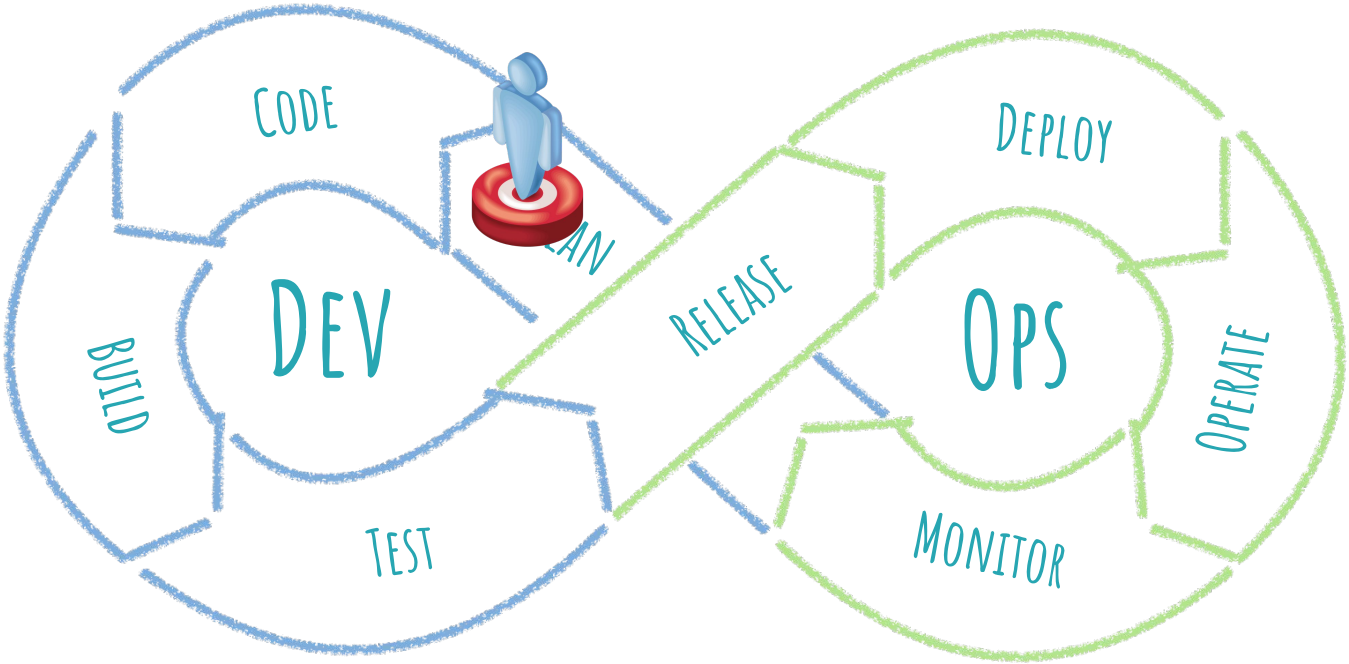


Monitor phase

Keeping the peace.

- **SREs** create alerts and on-call schedules.
- **SREs** provide continuous profiling for applications.
- **SREs** install kernel-monitoring tools like *eBPF*

Plan phase…again



Plan phase

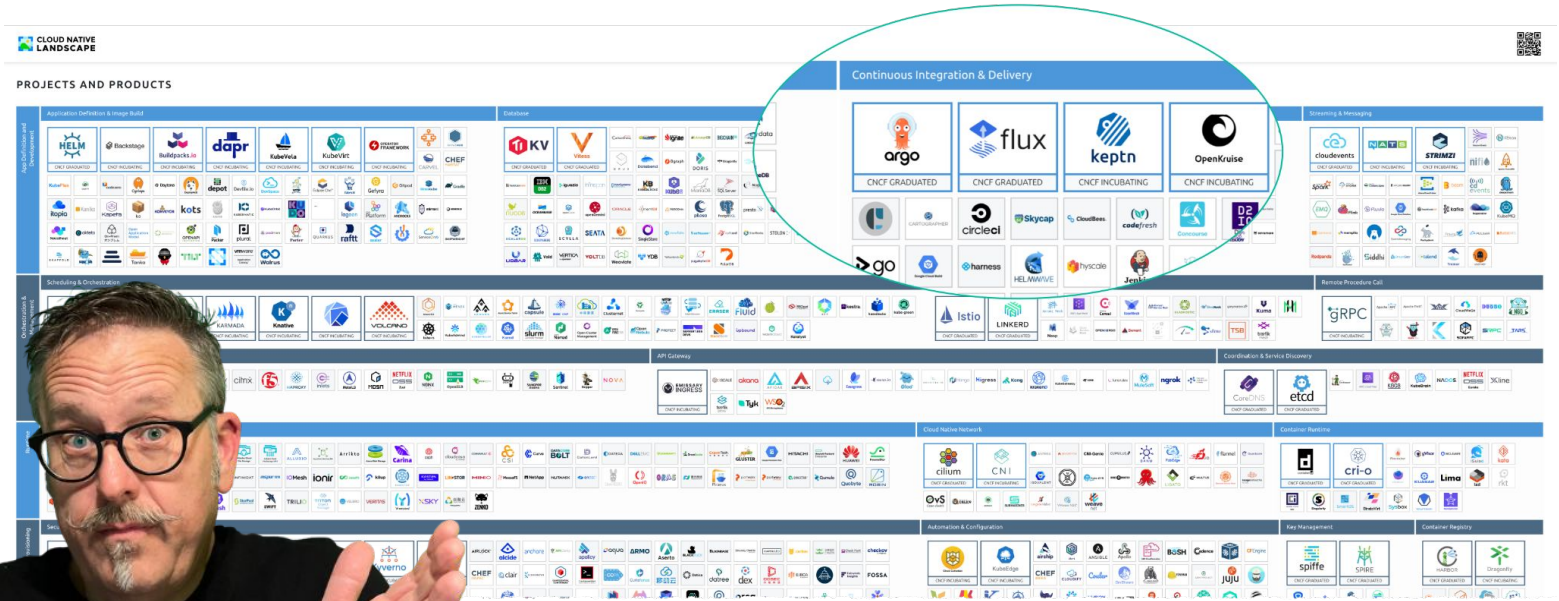
Back to the beginning.

- **SREs** provide feedback to refine baselines.

Act III

Choosing the right tools...

Navigating the CNCF



To be continued…

Thank you!

Connect with Paul as
@javaducky or **linkedin/in/pabalogh**

THIS SLIDE INTENTIONALLY LEFT BLANK