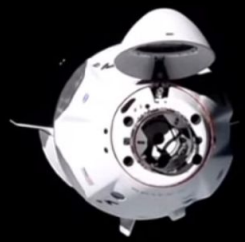
Continuous Automation



The question

Can we use various automated testing approaches to understand how reliable and resilient our applications and systems might be?

Agenda

The distributed system
- A 1000 ft view

Performance
- Shift left and moving right

Chaos
- Cultural change to when things fail

Automated checks
- Testing using real world examples

Security
- Surfacing potential vulnerabilities

Combining them all
- Tests that play well together stay together

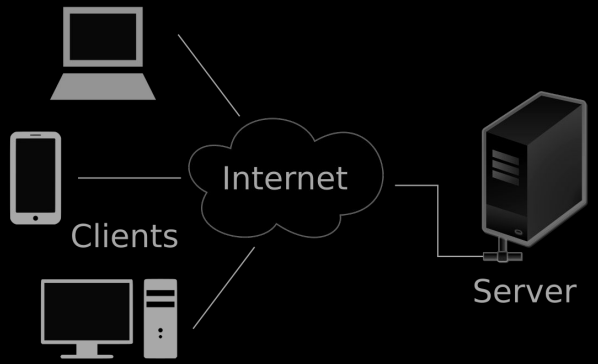
Observability
- Knowing what's going on at any point

The Goal of CA

The unification of test tooling is to provide the most reward from the least amount of effort in determining system reliability.

And that collectively provides useful information as to system state which surfaces problems and that can lead to predicting potential future issues

The distributed system

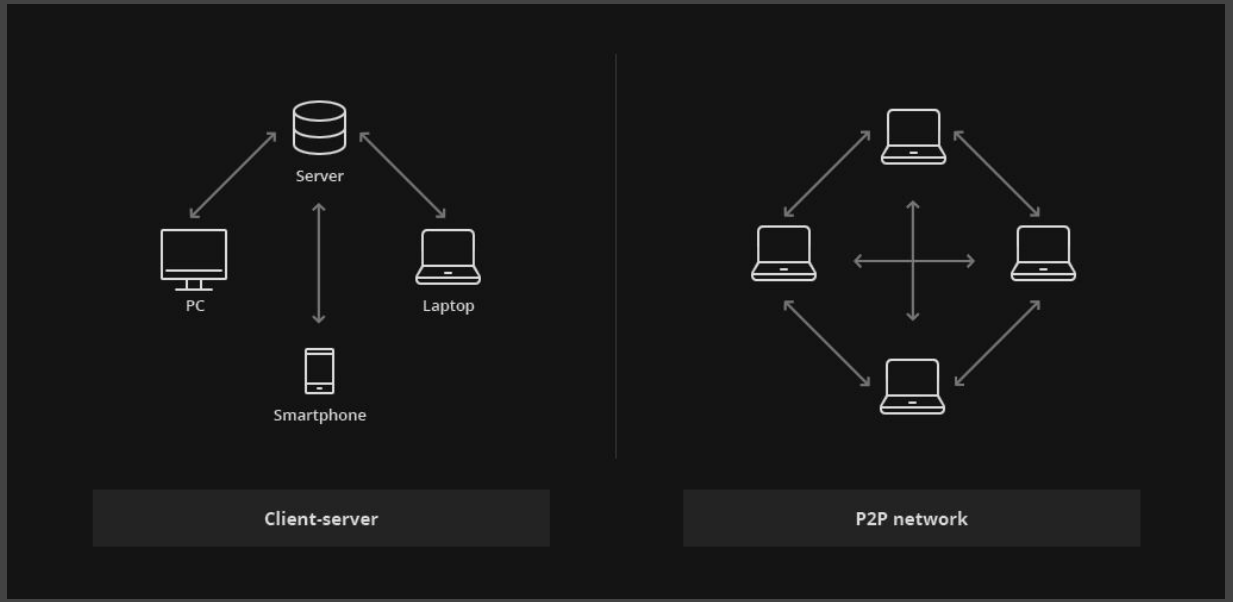


Distributed system simplified

A group of machines that pass information to each other in order to achieve a common goal

Client server & P2P

- No guarantees regarding path
- Server location could be anywhere
- Latency can vary
- Transfer rates can depend on other user



Distributed system fallacies

The Dream!

The network is reliable;
Latency is zero;
Bandwidth is infinite;
The network is secure;
Topology doesn't change;

Down on the farm...

The Thundering herd problem

The problem happens when a large number of processes or threads waiting for an event are awoken when that event occurs, but only one process is able to handle the event.

When (4) worlds collide

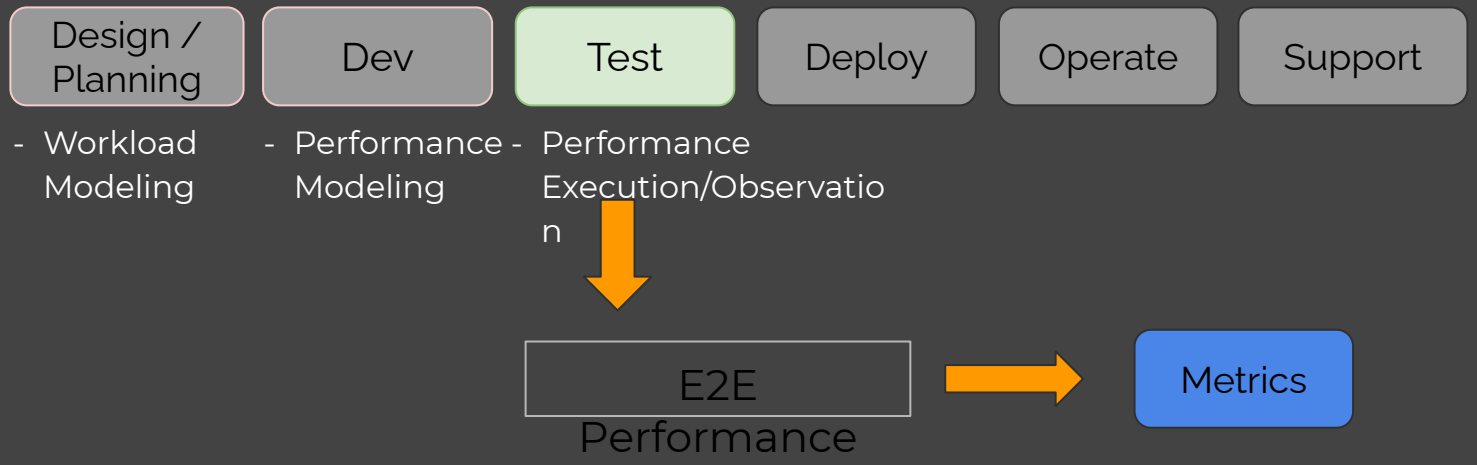
Chaos, Performance, Test Automation, Security

Performance Engineering

Planning and building the application
with performance in mind

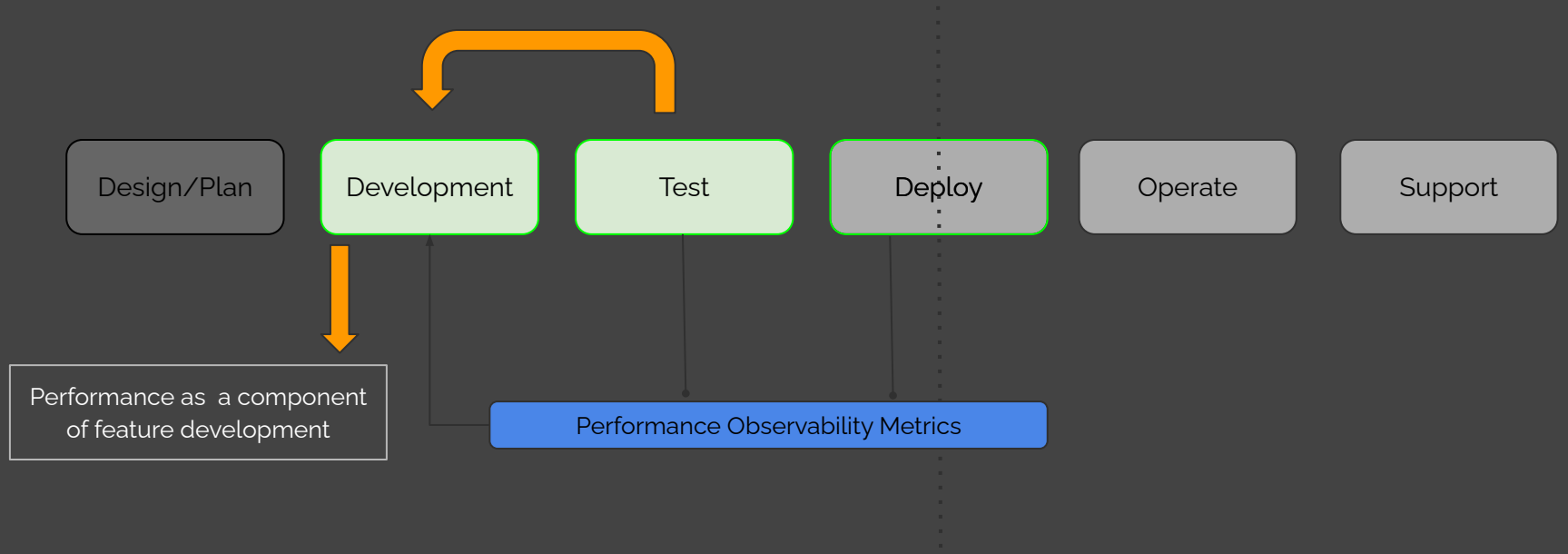
Traditional Performance testing

- Very waterfall orientated
- Testing late in the cycle leads to a larger feedback loop and in turn increasing cost, resources, energy and delivery time



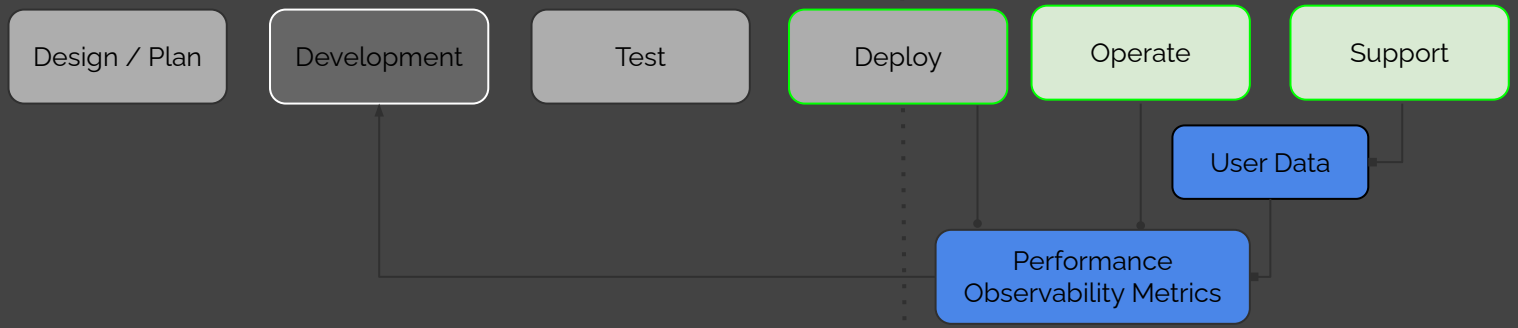
Performance Engineering
Shift Left

- Moving part of the performance testing process into the development phase



Performance Engineering
Move Right

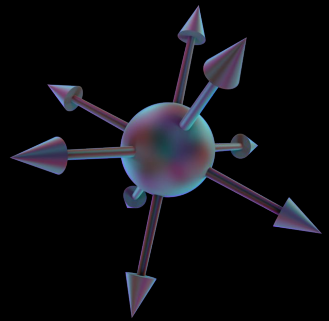
- Using production data to verify performance assumptions
- Using reported and collected user data in conjunction with operational metrics to drive development priorities



Performance Benefits

- Understand performance early
- Isolated testing
- Enhanced Traceability
- Reduced fix time
- Transparency

Chaos Engineering(CE)



Chaos Engineering is NOT...

About breaking the system

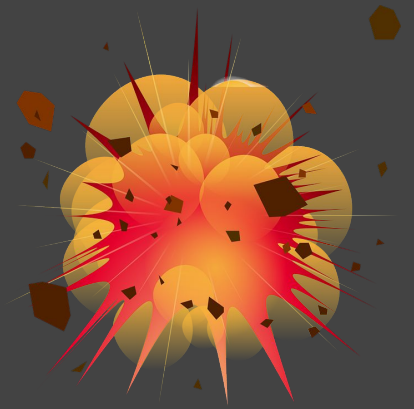
Chaos is about building a culture of resilience in the presence of unexpected system outcomes

It’s all about understanding the end user experience and understanding how we are building tolerant systems

Chaos is about experimentation

The Approach

- ➔ Start By Defining the Baseline (Steady-State)
- ➔ Hypothesize the Steady-State Will Endure
- ➔ Introduce Variables/Experiments
- ➔ Try to Disprove the Hypothesis



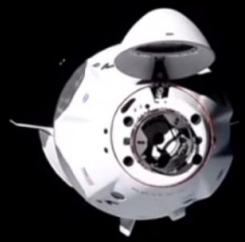
The Pillars

- ➔ Adequate coverage
- ➔ Run often and in Prod (or similar)
- ➔ Minimising the blast radius

Chaos Benefits

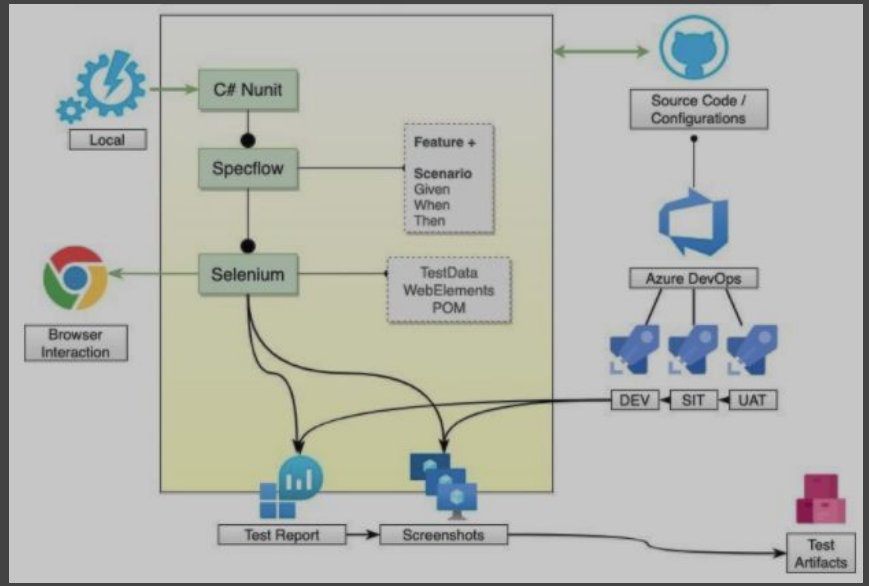
- Exposing system weaknesses
- Determining application/environment behaviour under varying conditions
- Cultural shift to what happens when things fail
- Improves how infrastructure is built
- Builds internal trust and empathy

Automation

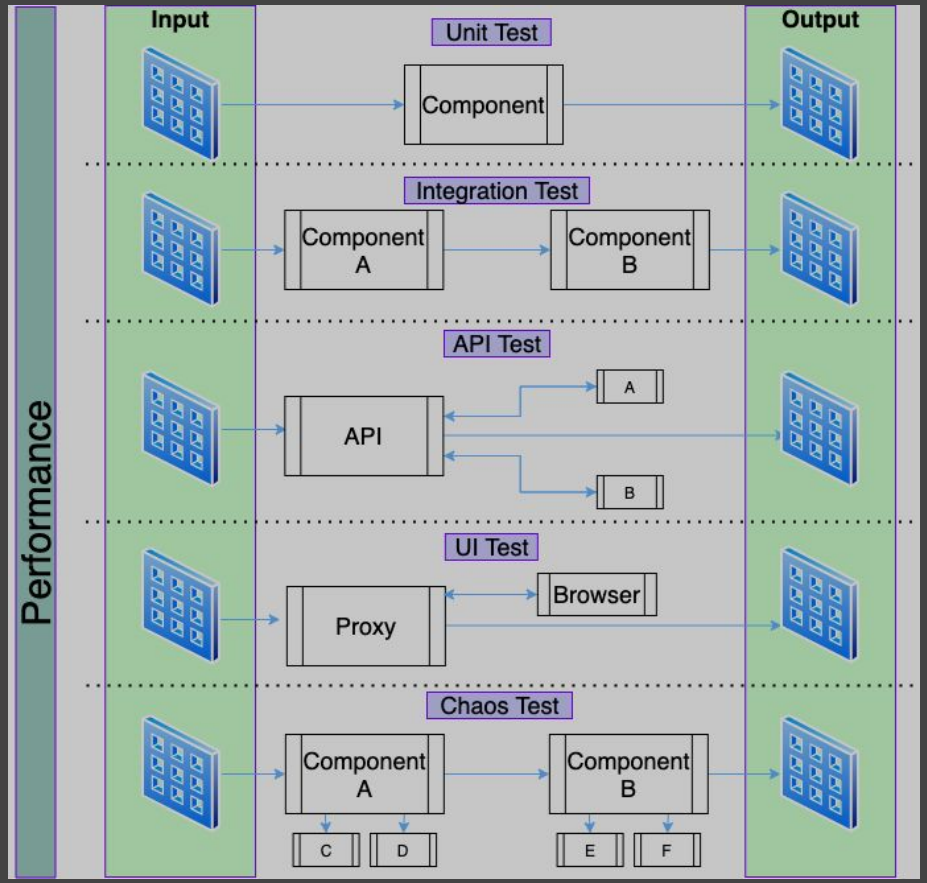


Automation in context

We are looking to confirm a question we expect we know the answer to



The comparison



We take inputs and produce outputs we then compare them to what's expected in the way of assertions

Performance spans all boundaries as each approach can and should have a performance consideration

Automation Benefits

- Confirming expectations
- Understanding logical flow
- Enhanced communication and socialisation
- Fast feedback cycles
- Reduced double handling
- Repeatability


Security

Vulnerability testing

SAST

- Inside out approach
- Can be run early on(feature branches)
- Cheaper to fix
- Can be run against all code bases (app, services, apis)
- Easily automatable

DAST

- Outside in approach
- Used later on in the SDLC
- Only used for web app and services
- Uses fault injection techniques

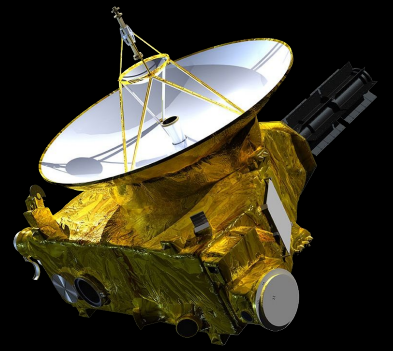
IAST

- Scalable
- Reduced false positives

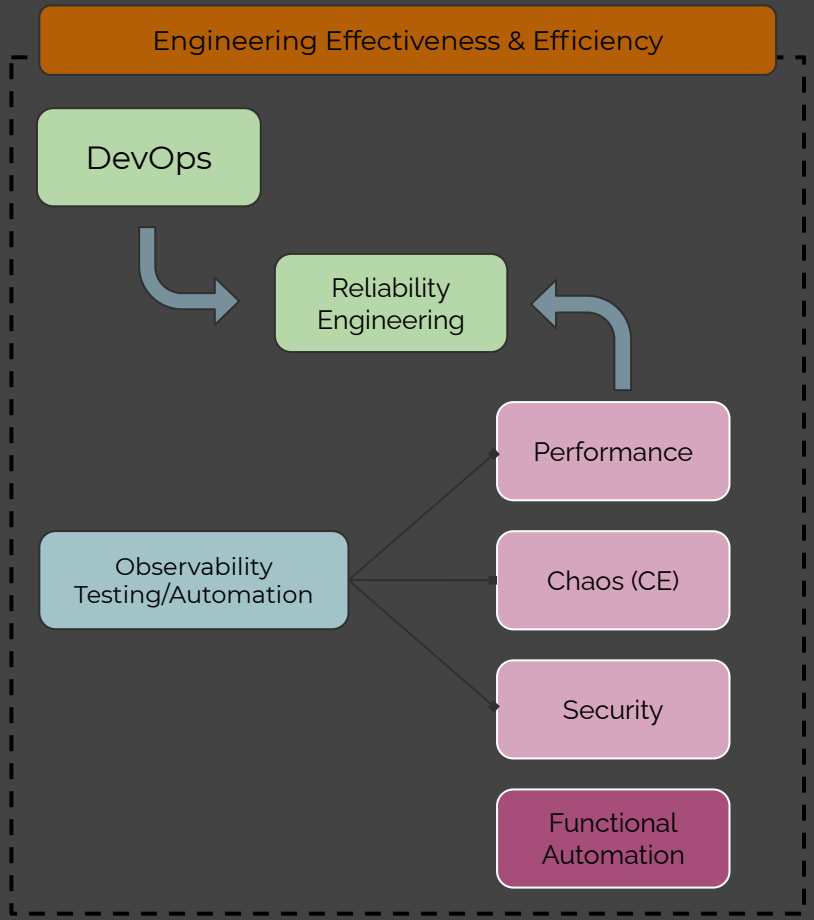
Security Benefits

- Improved reliability / predictability
- Reduced chance of error
- Unbiased feedback
- Lowered costs

Test Automation and SRE



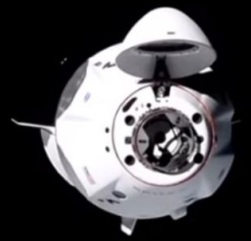
Measurements using SRE



- Define the problem space (SLO/SLI/SLA's)
- Abide to DevOps principles
- Consume and act on observability metrics

- Build a hypothesis and validate using implicit metrics
- Use of Performance component based models
- Actions/triggers based around observability data
- Use simple but measurable security scans

Observability

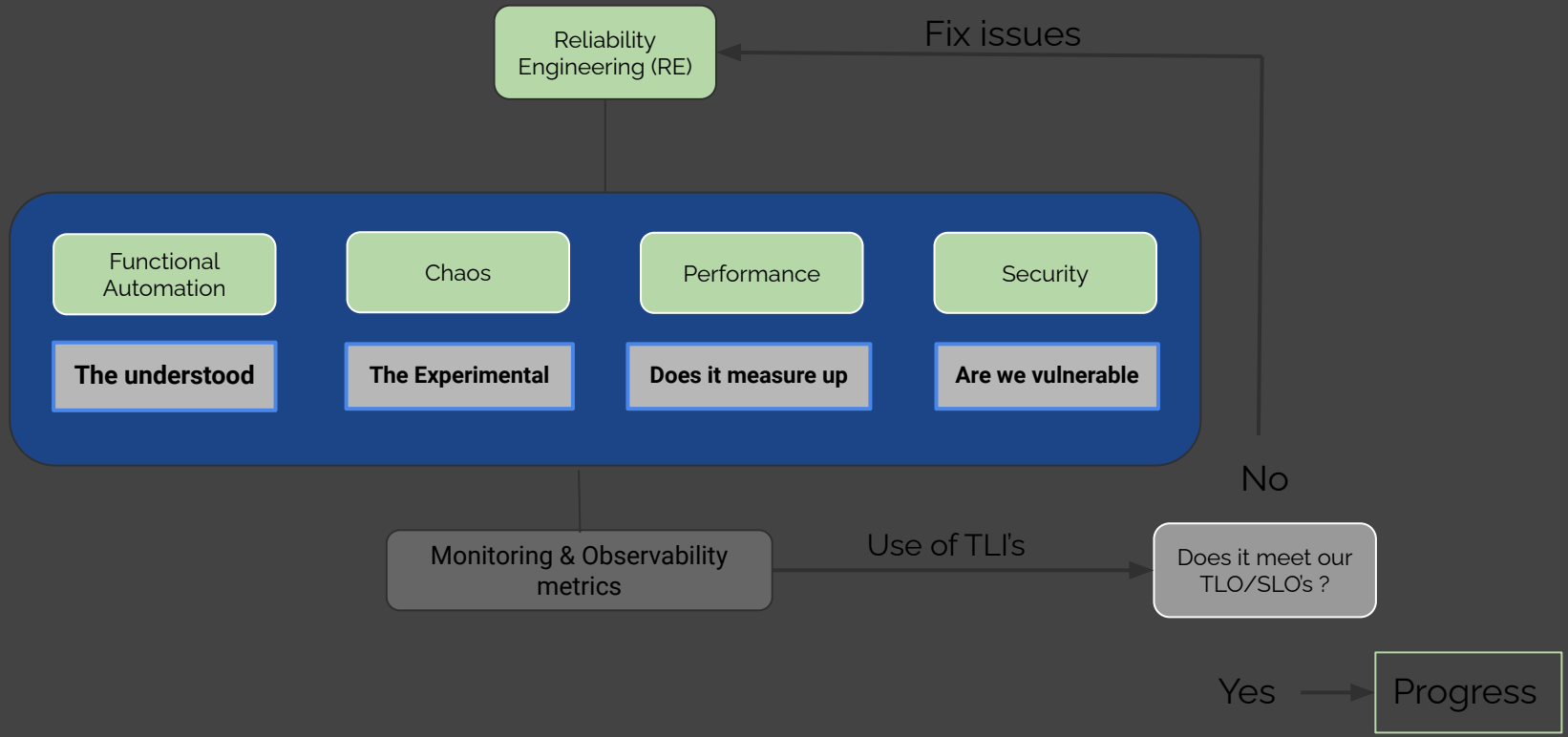


Understanding test state

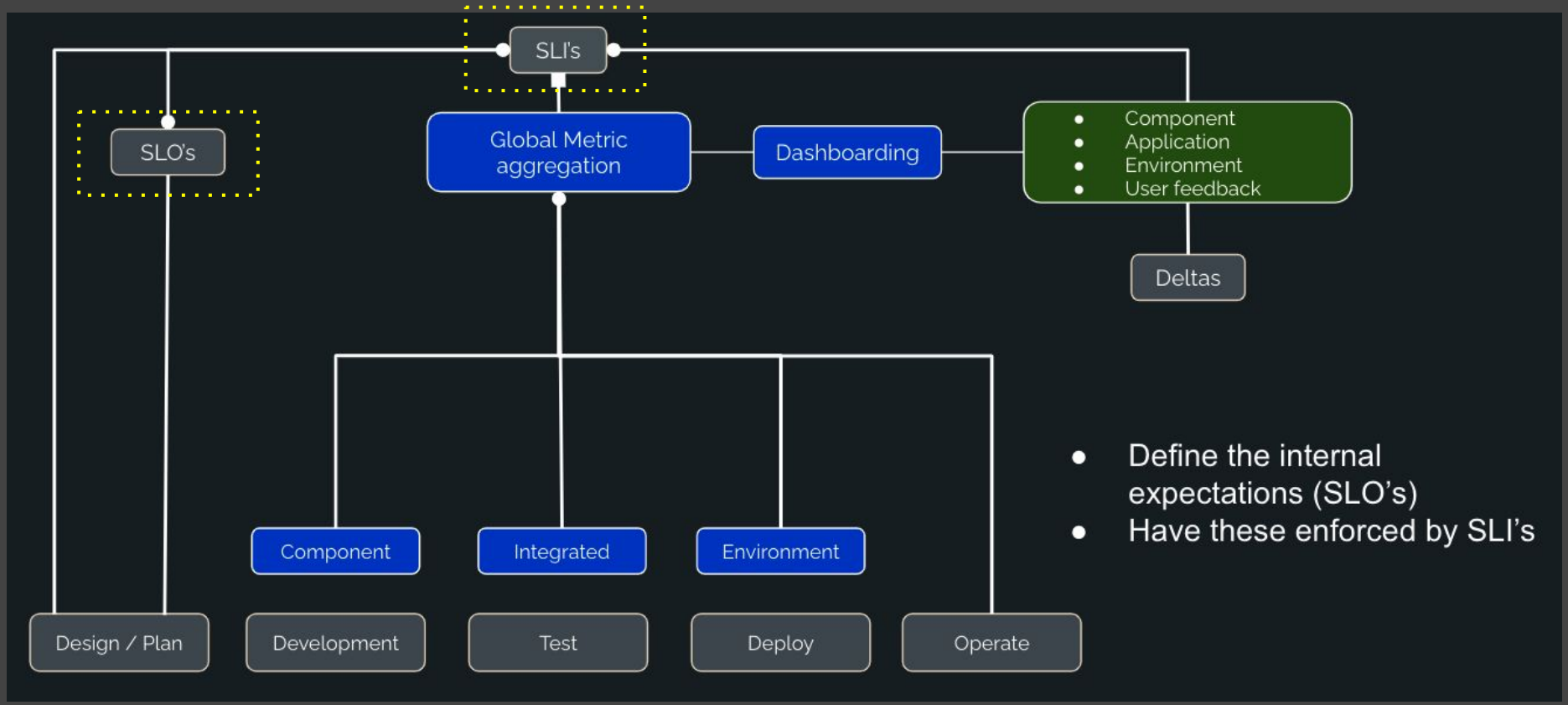
We use tests to verify load expectations, look for vulnerabilities, induce erratic behaviour and confirm the understood.

We need a way to make sure that the results that are generated have a level of accountability

Collecting the data

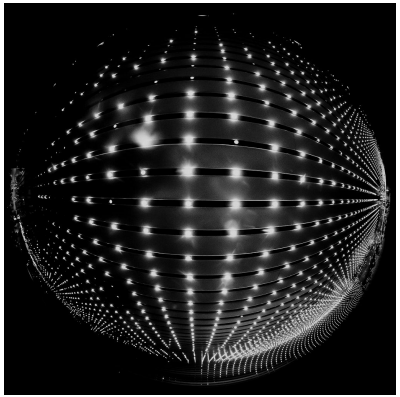


Doing the comparisons



Bringing it all together

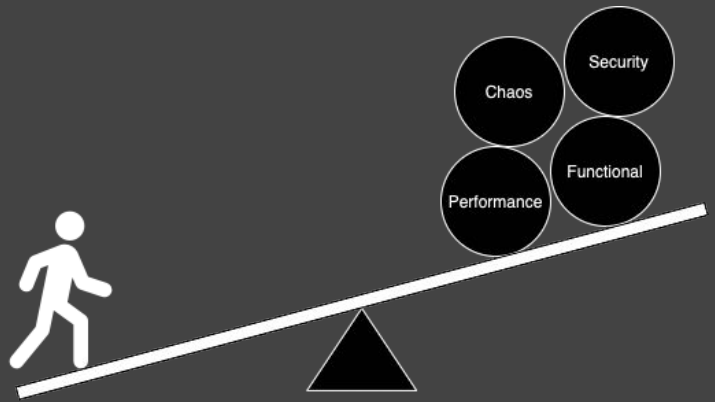
Chaos, Performance, Test Automation, Security



Finding Balance

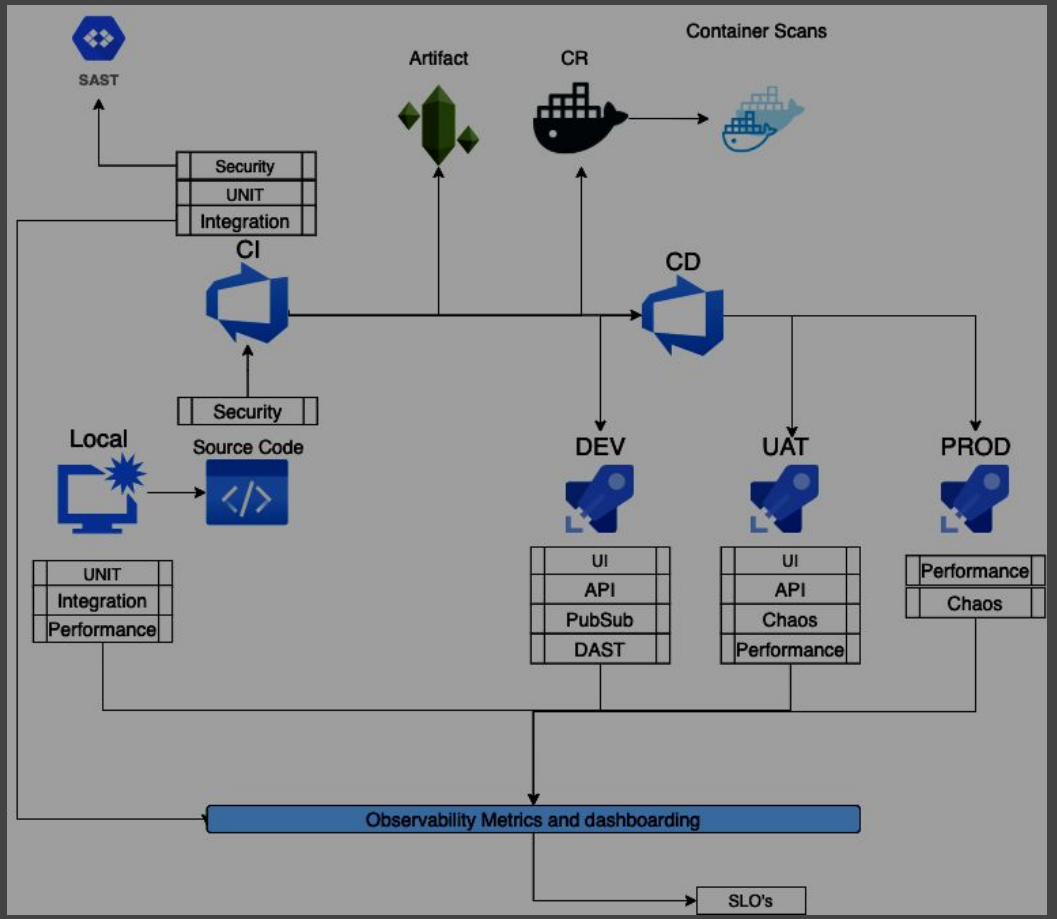
What's the right combination of tools and at what point in the development chain will they return the most benefit

- The right tool
- For the right domain
- At the right time
- Aligns to the teams maturity
- Open source contribution models



Example approach

- Aim for good distribution of various tests at each pivotal point
- Use the faster running tests early on
- Provide coverage appropriate to your domain
- Measure and expose all tests in a common/unified dashboarding solution



Getting traction - Lean Canvas

What are we trying to solve / Enable?

Problem
- We only deploy once a month

Solution
- Include automation mechanisms to build trust and confidence within the team and we can release faster

The approach

Process
- Build and share this lean canvas

UI
- Dev / Sit / UAT automation
- CI/CD integration (Cloud)

API assessment & implementation
- Evaluate API automation in use and present implementation options

Chaos
- Look at 1 small chaos experiment

Performance
- The ability for the developers to run these locally

Additional
- Look into container vulnerability and SAST scans

Why are we doing this
Benefit / Outcome

- Alignment to enterprise QE/automation processes
- Reducing manual testing effort

Objective / Deliverables
Metrics

- Standardised automation

Who is the audience
Customer / Users

- Team / Business stakeholders

How is it done now
Current state

- Manual tests
- Writing the automation tests in BDD format

Questions / Assumptions / Risks
Problem / Opportunity

- Uplifting the automation coverage percentage is critical

Risk
- The performance of the application
- Lack of resources

Re-Cap

The distributed system
- A 1000ft view

Performance
- Shift left and moving right

Chaos
- Cultural change to when things fail

Automated checks
- Testing using real world examples

Security
- Surfacing potential vulnerabilities

Combining them all
- Tests that play well together stay together

Observability
- Knowing what's going on at any point

Useful links

Nicholas Taleb's : The Black Swan: The Impact of the Highly Improbable
Principles of chaos engineering : https://principlesofchaos.org/
Let a 1000 flowers bloom : measuring engineering effectiveness

Contact

Web : https://scott.griffiths.me
Linkedin : https://www.linkedin.com/in/scgriffiths/
Twitter : https://twitter.com/__ScottyG_