



LINUXCON + CONTAINERCON + CLOUDOPEN CHINA

June 19 - 20, 2017 - Beijing, China

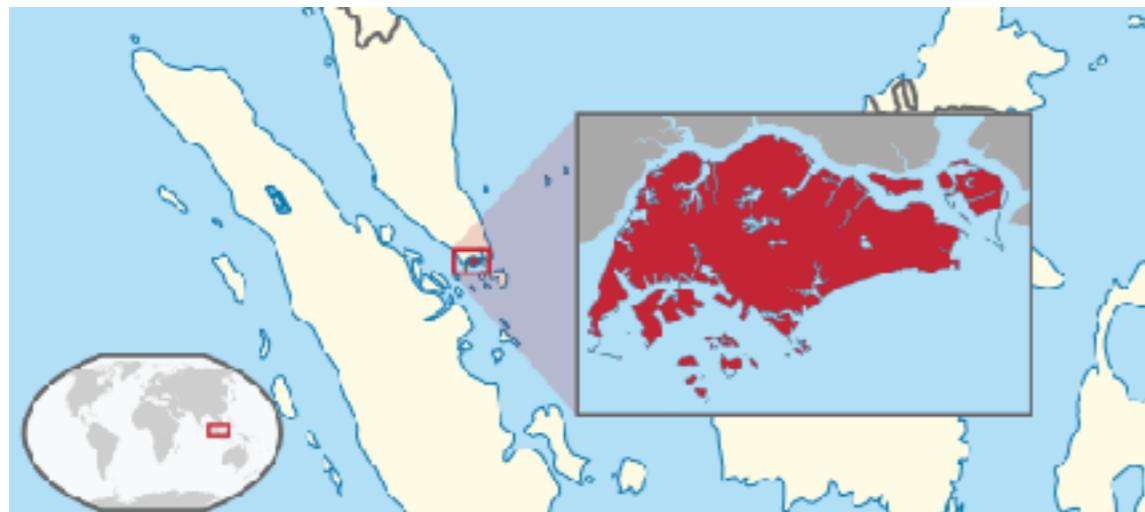
From Resilient to Antifragile Chaos Engineering Primer

By @Sergiu_Bodiu
Solution Architect

From Resilient to Antifragile

Chaos Engineering Primer

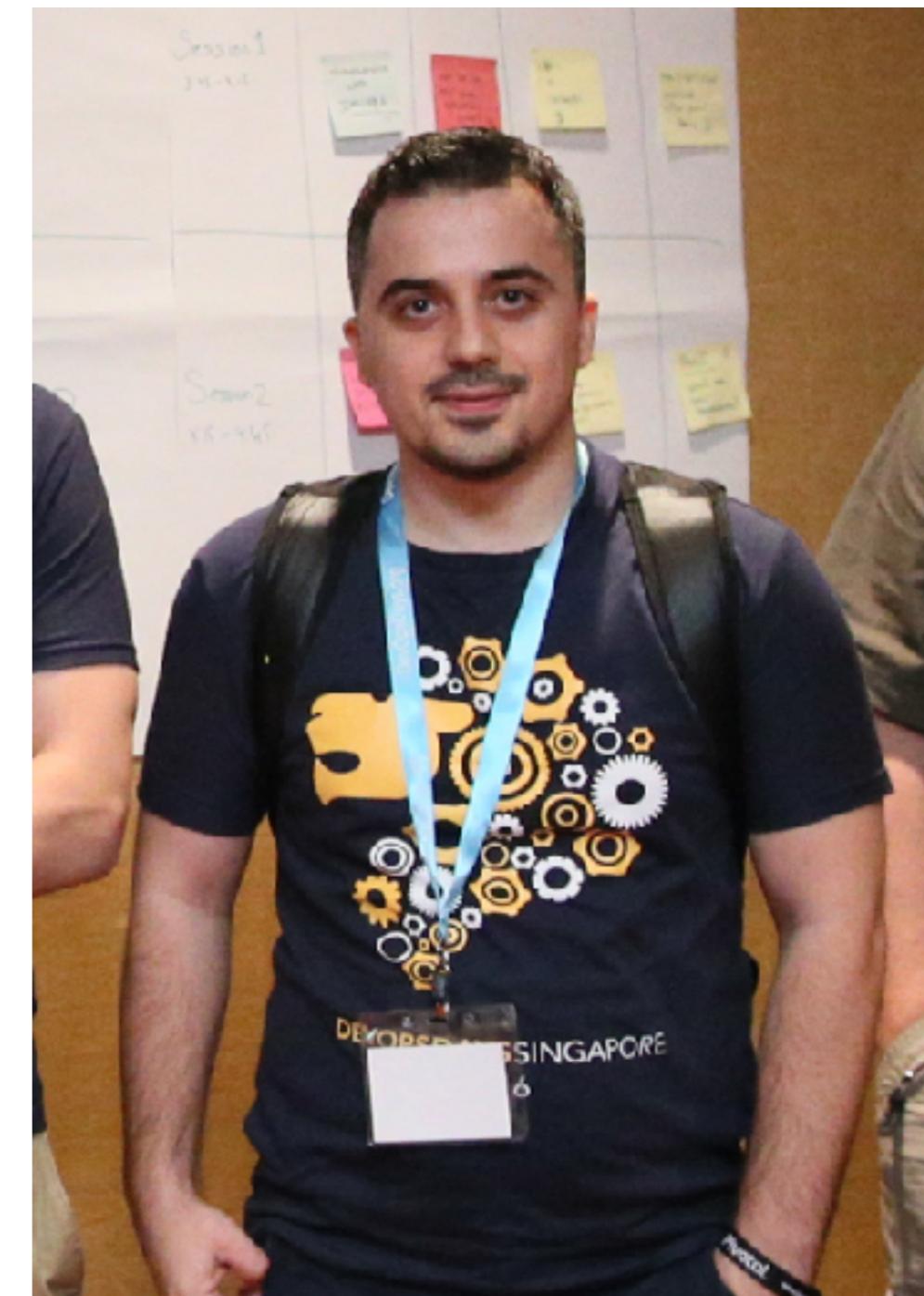
By @Sergiu_Bodiu
Solution Architect



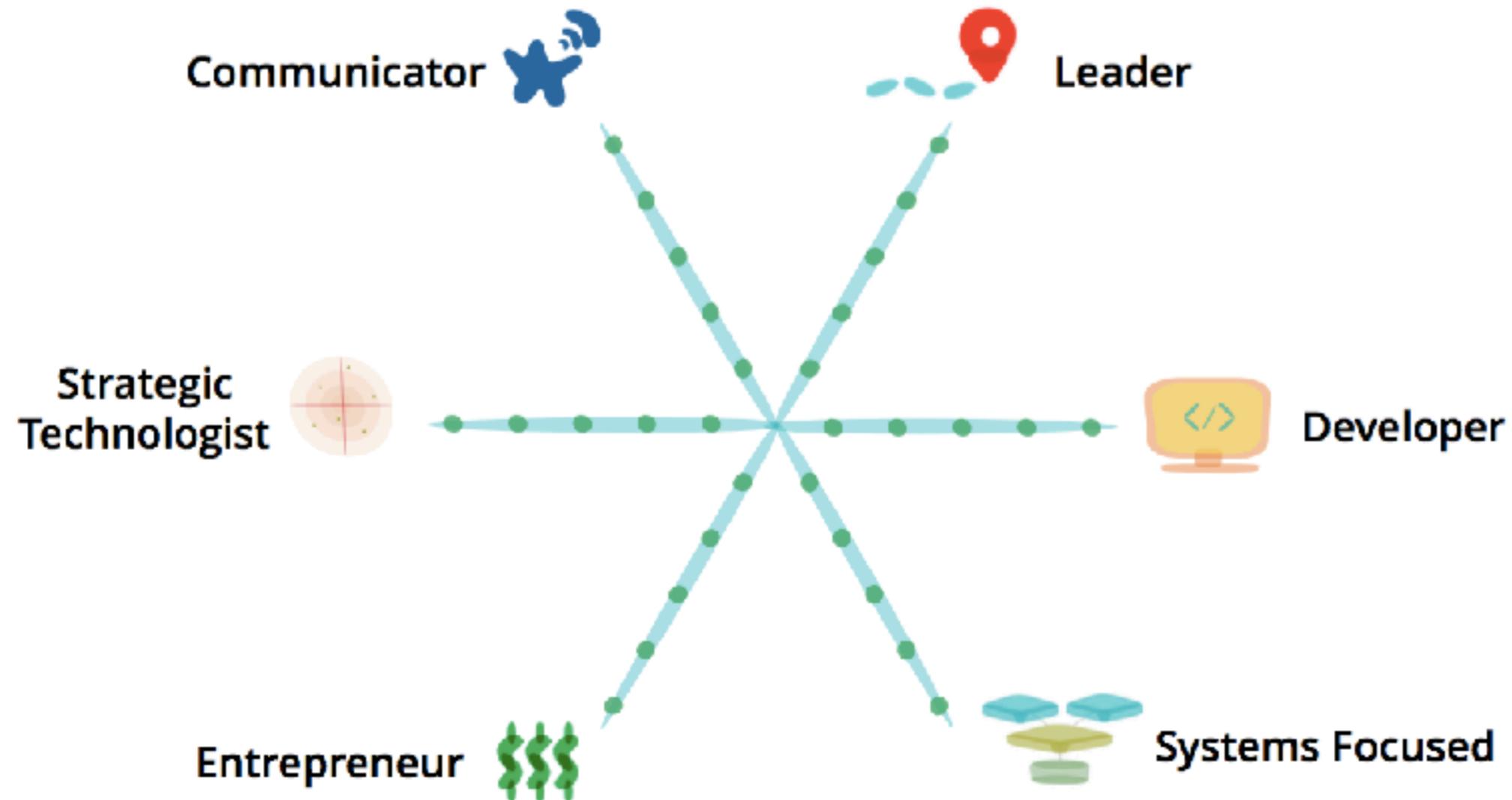
Singapore Spring
User Group



DevOpsDays
Singapore
Conference



what is an ARCHITECT



Risk management

The new normal:

from RESILIENT
to ANTIFRAGILE

A new way to look at organizations

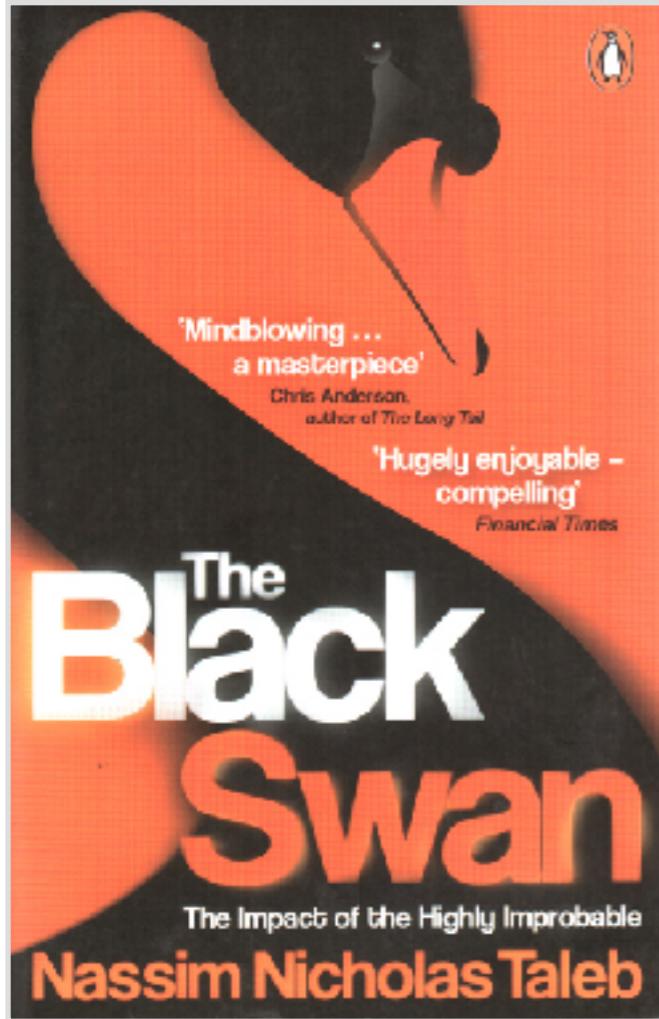
Fragile: At risk of total failure / financial ruin

Resilient: Takes damage, avoids total failure, recovers

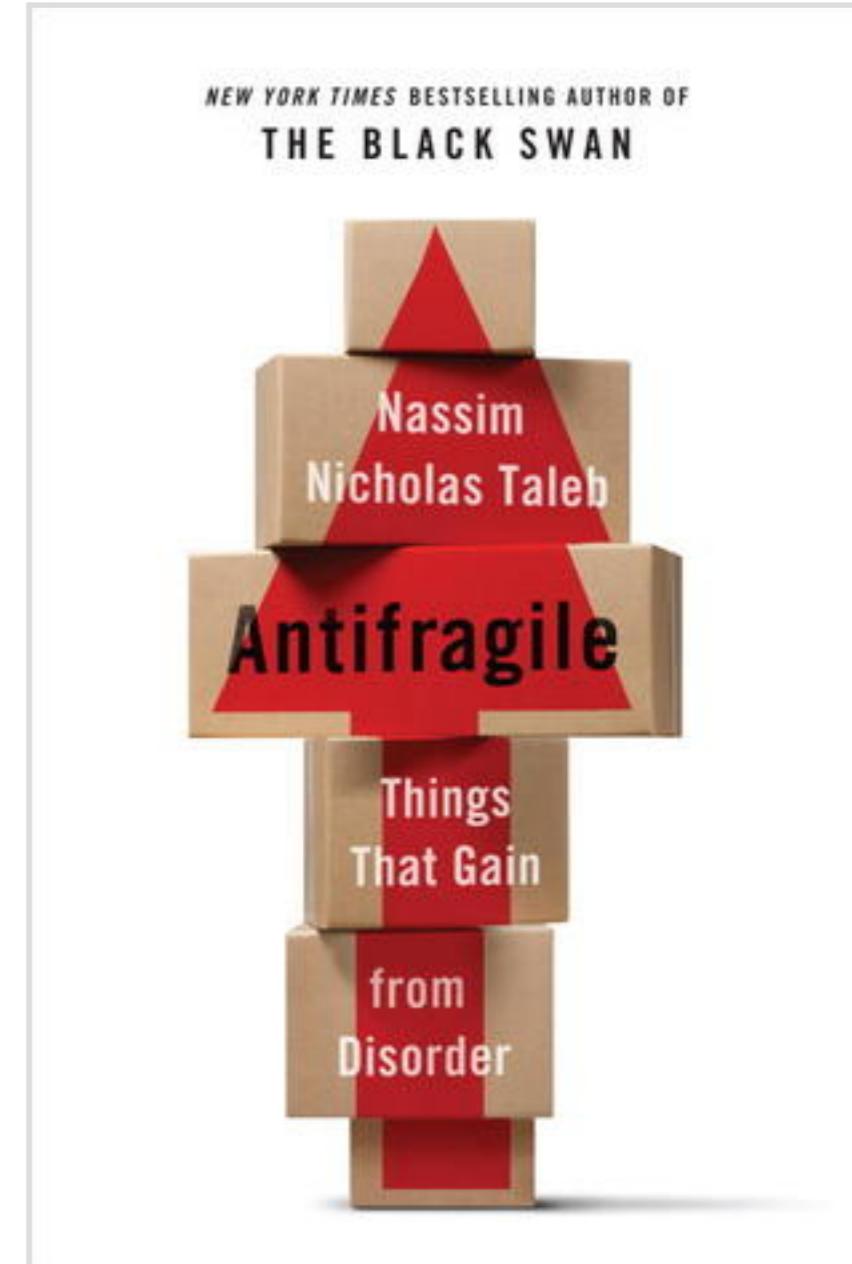
Robust: Absorbs uncertainty, repels blows, avoids damage

Antifragile: Responds to stress by mutating, maintains fitness for purpose. Identity Change.

Blueprint for living in a Black Swan world.



Antifragile
and only
the
Antifragile,
will Make it.



https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

1. The **network** is reliable.
2. **Latency** is zero.
3. **Bandwidth** is infinite.
4. The network is **secure**.
5. **Topology** doesn't change.
6. There is one **administrator**.
7. Transport cost is zero.
8. The network is homogeneous.

Software is Single Point of Failure

Root Cause Analysis: While component failures such as NETWORK, STORAGE, SERVER, HARDWARE, and POWER failures are anticipated and thus guarded with extra redundancies.

| Root Causes and Implications | | | | | | |
|------------------------------|------------|--------|-------------|------|-------|----------|
| | FullOutage | OpFail | Performance | Loss | Stale | Security |
| Hardware | 5 | 1 | 2 | . | . | . |
| Server | 7 | 4 | 1 | . | . | . |
| NatDis | 9 | . | 4 | . | . | . |
| Security | 12 | 1 | 2 | 1 | . | . |
| Human | 12 | . | 4 | 1 | . | . |
| Power | 9 | 2 | 3 | . | . | 6 |
| Storage | 21 | . | 4 | . | . | . |
| Cross | 28 | 2 | 4 | 1 | . | . |
| Config | 30 | 3 | 6 | 3 | . | . |
| Load | 22 | 4 | 15 | 1 | . | . |
| Upgrade | 44 | 6 | 4 | 2 | 2 | . |
| Network | 43 | 7 | 10 | 2 | . | . |
| Bugs | 36 | 9 | 19 | 2 | 2 | 1 |
| Unknown | 206 | 114 | 48 | 5 | 5 | . |

| Root Causes and Fixes | | | | | | | | | |
|-----------------------|----------|--------|--------|----------|-------|-------|---------|-------|------|
| | Hardware | Server | NatDis | Security | Power | Human | Storage | Cross | Load |
| Hardware | 1 | 4 | 1 | . | . | . | . | . | . |
| Server | 7 | 2 | 1 | . | . | 1 | . | . | . |
| NatDis | . | 9 | . | 3 | . | . | . | . | . |
| Security | 10 | 1 | . | 3 | . | . | . | . | 1 |
| Power | 7 | . | 7 | 3 | . | . | . | . | 1 |
| Human | 5 | 1 | 2 | 4 | 4 | 1 | . | 1 | 1 |
| Storage | 2 | 18 | . | 3 | . | . | 1 | . | . |
| Cross | 8 | . | 1 | . | . | . | 20 | . | . |
| Load | 8 | 1 | 7 | 5 | 11 | 5 | . | 1 | 4 |
| Config | 9 | 1 | 4 | 5 | 10 | 12 | . | 1 | 2 |
| Network | 28 | 11 | 7 | . | 4 | 1 | 3 | 1 | . |
| Upgrade | 27 | 3 | 9 | 6 | 3 | 2 | . | 10 | 1 |
| Bugs | 17 | 4 | 12 | 10 | 7 | 4 | . | 7 | 1 |
| Unknown | 345 | 1 | 3 | 3 | . | . | . | . | . |

Distributed Systems Complexity



OK SO WHAT ARE THE SOLUTIONS TO THIS
“TECHNICAL DEBT”?

CAN WE GET TECHNICAL REFINANCING?

MAYBE A TECHNICAL BAILOUT?

OPTIONS PEOPLE

RETWEETS LIKES
1,195 955

6:31 AM - 3 Sep 2015

43 1.2K 955

Complexity is
like
Addiction...

Case study: How
complexity creeps in

- @jasonfried



Cannot connect to the Netflix Service

Please Try Again

Chaos Engineering

Discipline of **experimenting** on a distributed system in order to **build confidence** in the system's capability to withstand turbulent conditions in production.

NETFLIX

Some outages in the Region

SingTel fined a record \$6m for Bukit Panjang exchange fire;

Telstra goes down again, people can't drink beer or catch Ubers

Amazon Web Services outage causes Australian website chaos

Backups

"Backups always succeed.
It's the restores that fail.

Test your backups by practicing
restores!"

Netflix Simian Army

Suite of tools for
keeping your
cloud operating
in top form.

[https://github.com/Netflix/
SimianArmy](https://github.com/Netflix/SimianArmy)



Chaos Monkey

1. Active during normal working hours
2. Break things in production
3. Design better software services
4. Embracing failure

https://github.com/Netflix/security_monkey

Monitor AWS and GCP accounts for policy changes and alerts on insecure configurations.

Security Monkey can be extended with custom account types, custom watchers, custom auditors, and custom alerters.



Other Monkeys

- Latency Monkey
- Janitor Monkey
- Conformity Monkey
- Doctor Monkey





PRINCIPLES > TOOLS

why do we do >
What we do

In Progress

The image is a composite of several screenshots from different project management tools:

- Top Left:** A Jira board view showing multiple projects like "Customer", "Support", and "Marketing".
- Top Center:** A Trello board titled "In Progress" with cards for "iOS v4.23 Release", "Partner Portal UI update", and "Android".
- Top Right:** A "New" section showing three issues: #3 (Bug Alpha 2), #2 (Feature 0.11.x), and #5 (Feature REST API implementation).
- Middle Left:** An "Incredible Universe" Program Read Map showing releases across weeks.
- Middle Center:** A Kanban board titled "Project Backlog" with columns for "Product Backlog", "1.7.1", "1.8.0", and "1.9.0".
- Middle Right:** A detailed Kanban board with many cards and columns, likely a custom backlog tool.
- Bottom Left:** A Jira backlog interface with a search bar and filter options.
- Bottom Center:** A "Sprint 1" backlog view showing stories and tasks.
- Bottom Right:** A "Sprint Backlog (20 stories)" view with a grid of stories and tasks.

MOST OF US USE SOME SORT OF BACKLOG MANAGEMENT TOOLS...

In Progress

This screenshot shows a specialized backlog management application:

- Top Bar:** Includes "Board", "Details", "List", "Roadmap", "Q", "Options", and "Change assignee".
- Left Sidebar:** Shows a tree view of categories: "Customer", "Support", "Marketing", "Product Backlog", "1.7.1", "1.8.0", and "1.9.0".
- Central Area:**
 - Product Backlog:** A Kanban board with columns: "Backlog", "1.7.1", "1.8.0", and "1.9.0".
 - Story View:** A detailed view of a story titled "As an user, I would like to be able to add an attachment to a product instead of a file". It includes sections for "Description", "Acceptance Criteria", "Notes", and "Attachments".
 - Task View:** A list of tasks associated with the story, including "TX-2250", "TX-3493", "US-1850", "US-1960", and "US-2250".
- Bottom:** A footer with links for "Help & View", "All Tracker's", and "Logout".

Principles of Chaos

1. Build a Hypothesis around Steady State Behavior
2. Vary Real-world Events
3. Run Experiments in Production
4. Automate Experiments to Run Continuously

Chaos Engineering Whitepaper 2016

Hypothesize

> sudo watch

- Start with steady state behavior.
- Monitor metrics that are visible
- Capture an interaction between the users and the system.

TIP: Utilisation is Virtually Useless as a Metric!

Vary Events

> sudo halt

- Terminate virtual machine instances
- Inject latency into requests between services
- Fail requests between services
- Fail an internal microservice
- Make an entire region unavailable

TIP: Select only a subset of users

Experiment

- End to end TESTING (Expensive)
 - Process is slow
 - Configuration Drift from Production
- 92% ERRORS could be prevented (Simple)

TIP: Customers don't behave as your JMeter script.

Automate

> sudo while (1)

- Distributed systems changes continuously over time.
- Engineers modify the behavior of existing services, add new services.
- Engineers are changing runtime configuration parameters, upgrading and patching systems

TIP: Depending on the context, **change** the rate of each experiment.

Principles of Chaos

1. Build a Hypothesis around Steady State Behavior
2. Vary Real-world Events
3. Run Experiments in Production
4. Automate Experiments to Run Continuously

TIP: Intentionally break things, compare measured with expected impact, and correct any problems uncovered this way.

Chaos Engineering Whitepaper 2016

Reference Architecture for Cloud Native Platform

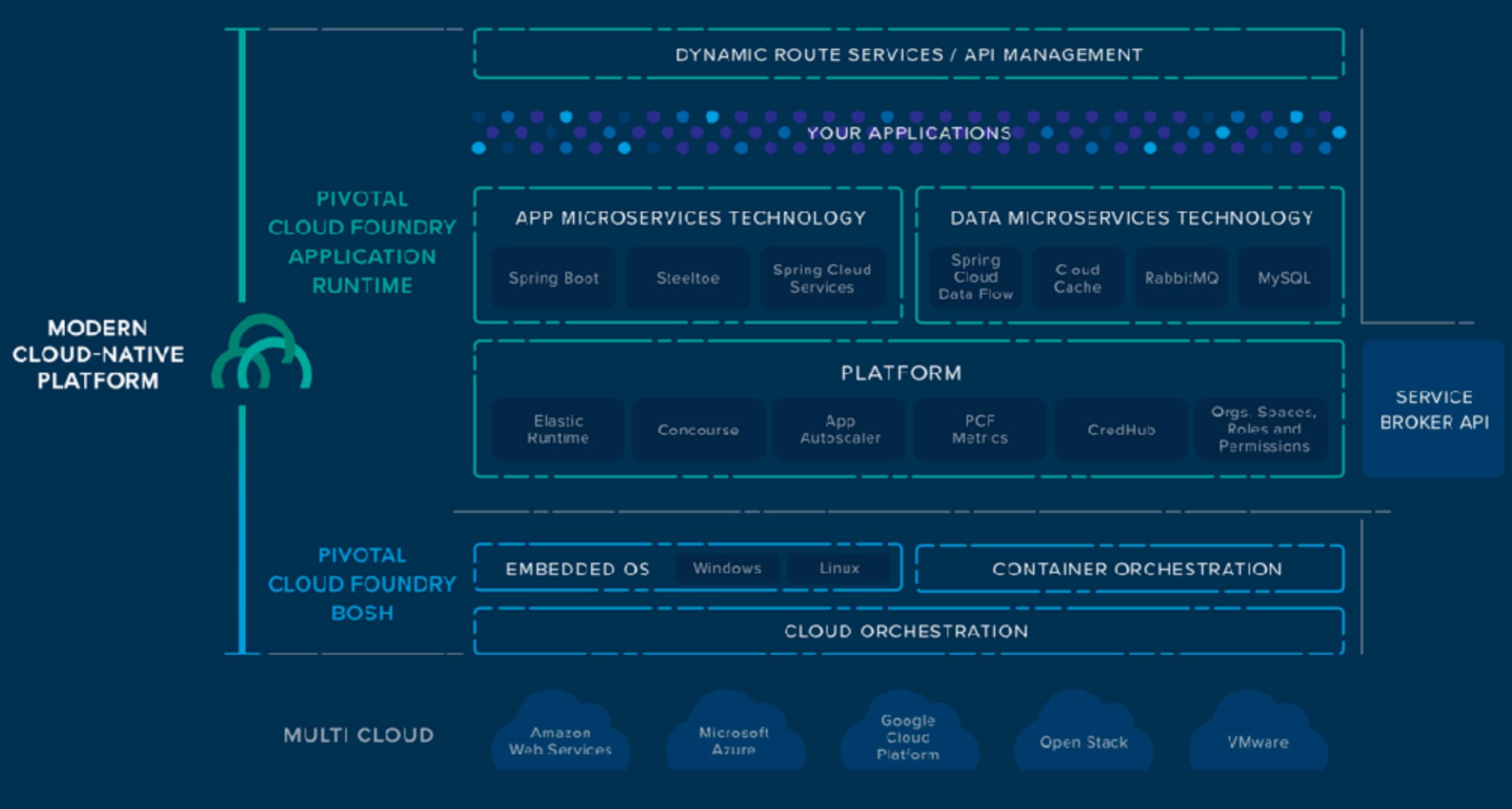
| Infrastructure | Operations | Deployment | Runtime & Data | Security |
|------------------------------|--|---|------------------------|--------------------------------------|
| Container Orchestration | Service Monitoring and Dependency Management | Lifecycle Management Deploy Patch Upgrade Retire | HTTP / Reverse Proxy | Control Plane Audit & Compliance |
| Infrastructure Orchestration | Inventory, Capacity, and Management | Release Packaging, Management & Deployment | Application Runtime | Security Event & Incident Management |
| Service Discovery | Event Management and Routing | CI Orchestration | In-Memory Object Cache | Secrets Management |
| Configuration Management | Persistent Team Chat | TDD Frameworks | Search | Certificate Management |
| Core IaaS | Metrics & Logging Analytics & Visualization | Artifical Repository | Messaging | Identity Management |
| NAT | DNS | Standard Builds & Configurations | NoSQL Docuemnt Store | Threat & Vulnerability Scanning |
| SDN | IPAM | Source Control Management | NoSQL Key/Value Store | Network Security |
| Firewalls | WAN & VPN | | | |
| Storage | Load Balancers | | | |
| Compute | Network | | | |

Figure 2 - The 5 domains of a cloud-native platform.

Pivotal

Pivotal Cloud Foundry

Pivotal



Chaos Lemur demo

Chaos Lemur =
Chaos Monkey + PCF

Locust demo

Locust is an open-source Python load testing framework.

- Define user behaviour in code
- Can execute end-to-end user test with sessions and cookies.
- Expands to multiple slaves to increase load capacity
- Allows for distributed user paths based on percentages



Gatling is an open-source Scala load testing framework

- High performance
- Ready-to-present HTML reports
- Scenario recorder and developer-friendly DSL



Lessons Learned

- **Systematic approach** to Chaos Testing
 - This is incredible hard under pressure.
- **Don't wait** so long to start load testing.
 - The conversations drive new requirements.
 - Changing architecture last minute is extremely dangerous.
- **Join the community**
 - Build relation with Networking Team, Database Team, Third Party Partners, Vendors etc..
 - Make everything Asynchronous (Embrace Failure, Background Tasks, Retry, Idempotence)

The importance of reliability

Don't trust claims systems make about themselves & their dependencies.

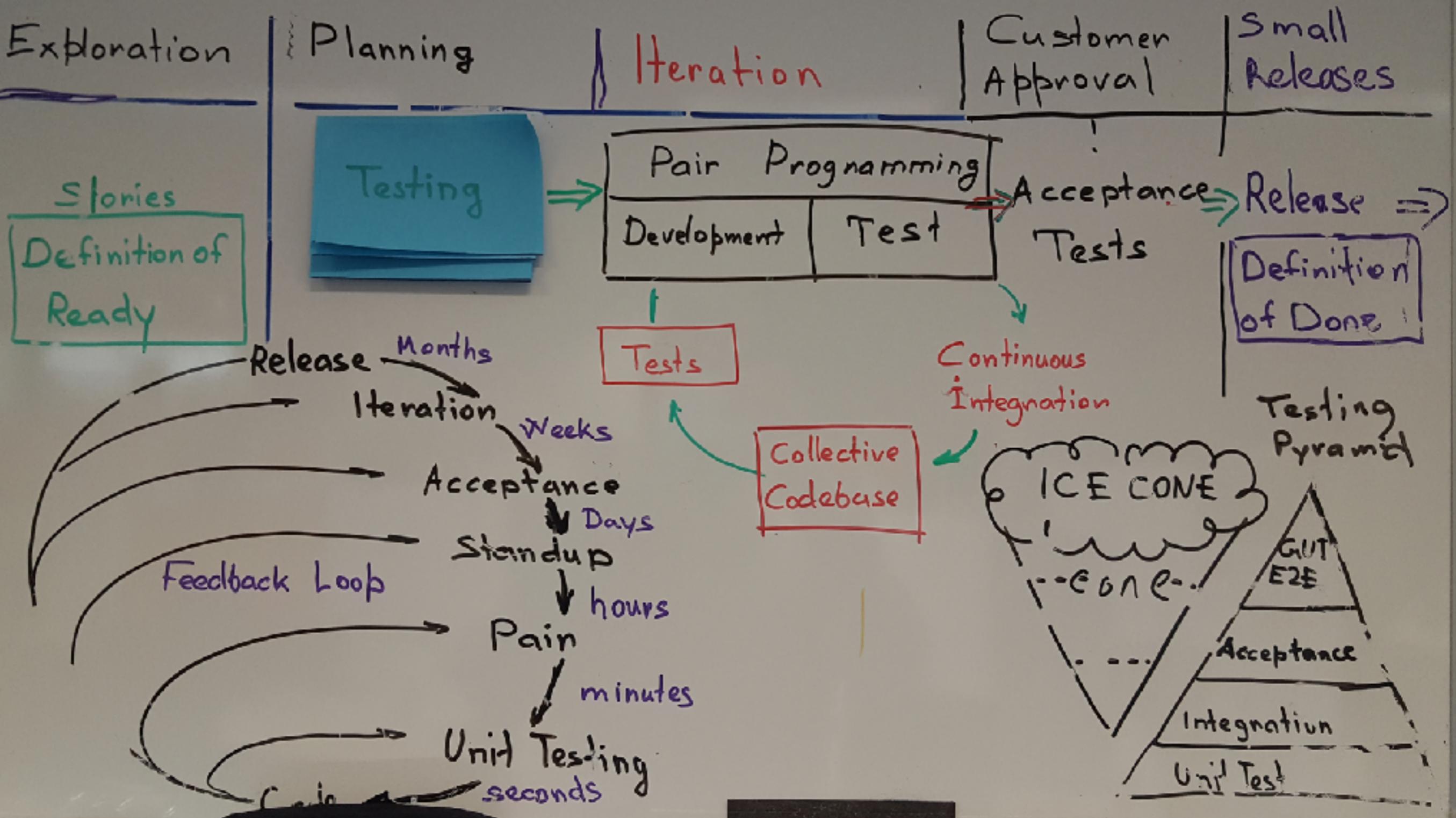
Verify by breaking.

Clean your process

**Culture > Principles >
Tools**

> sudo halt
Incident Start > Post Mortem
Impact

Testing Pyramid



Further Reading

<https://www.infoq.com.br/presentations/exercising-failure-at-netflix>

<https://www.infoq.com/podcasts/failure-as-a-service>

<https://www.infoq.com/articles/chaos-engineering>

@Ops_Engineering <https://www.youtube.com/watch?v=CZ3wluvmHeM>

@caseyrosenthal <https://www.youtube.com/watch?v=Q4nniyAarbs>

[Peter Alvaro: Orchestrated Chaos: Applying Failure Testing Research at Scale](#)

[Adrian Colyer Simple Testing Can Prevent Most Critical Failures](#)

Questions

Thank You

@sergiu_bodiu



Principles

Any developer building applications which run as a service. Ops engineers who deploy or manage such applications.

[https://12factor.net:](https://12factor.net)

Anyone working in software that writes tests or maintains continuous integration pipelines.

<http://www.10factor.ci>