

DEVOPS 101

ESIR3 DLC, 2022-2023

BENOIT COMBEMALE

FULL PROFESSOR, UNIV. RENNES 1 & INRIA, FRANCE

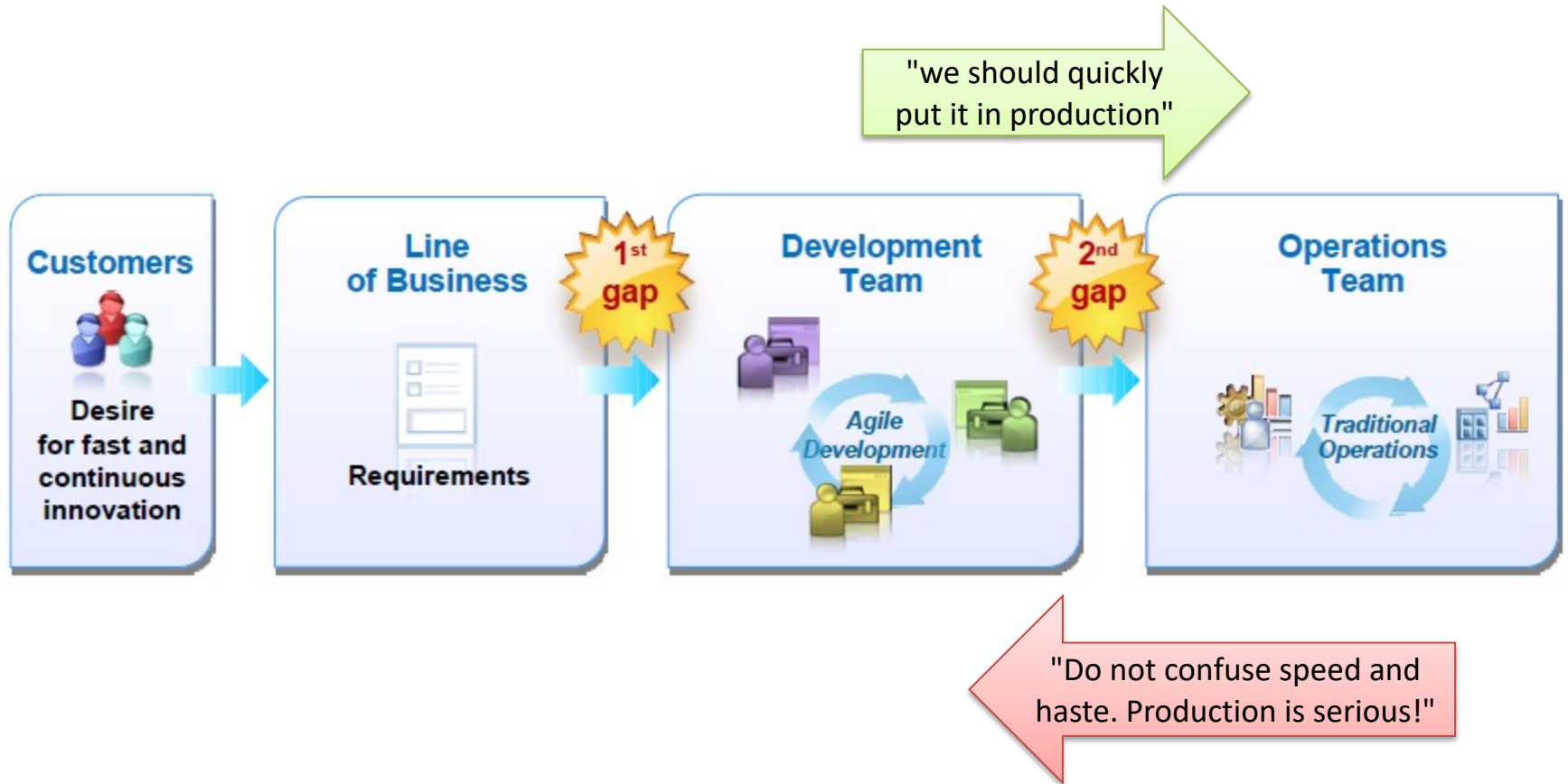
[HTTP://COMBEMALE.FR](http://combemale.fr)
[BENOIT.COMBEMALE@IRISA.FR](mailto:benoit.combemale@irisa.fr)
[@BCOMBEMALE](https://twitter.com/bcombemale)



Traditional Software Development Model



Traditional Software Development Model



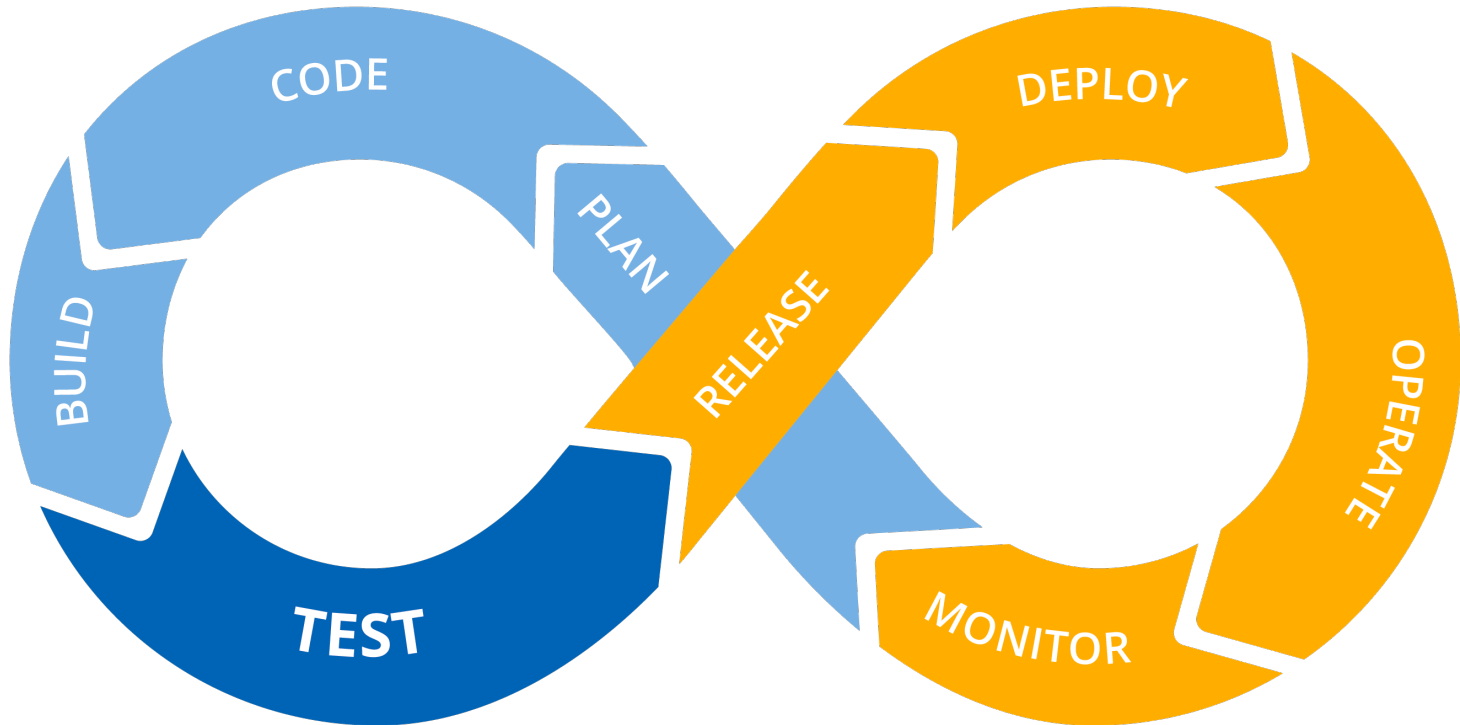


What's DevOps?

***“A software engineering practice
that aims at unifying
software development (Dev)
and software operation (Ops).”***

DevOps has been initially coined by Patrick Debois in 2009

What's DevOps?



Motivations

- Reduce the release cycle (time to market, lead time between fixes...)
- More fragmented approach (small increments vs. bigbang)
- Seamless updates
- Shared responsibilities (all in the same boat)
- Continuous improvement

Typical Stories

- Story 0: Dev and Ops collaborate to develop environment definitions
 - *Value: Ensures that Dev understands and deals with production-like environments; avoids architectural miscommunications*
- Story 1: Dev continuously delivers application changes to a realistic environment for testing
 - *Value: Shared technology ensures testable environments and script reuse for repeatable delivery; Test org always has known good builds, properly deployed*
- Story 2: Release Applications from Test /Staging to production
 - *Value: Shared technology and automation ensures no gratuitous differences between dev/test and prod*
- Story 3: Collaborative incident management
 - *Value: ensures an integrated process for reproducing and resolving defects and issues between dev, test, and ops*
- Story 4: Dev and Ops use the same analysis and instrumentation in dev, test, and ops
 - *Value: Ensures a common understanding of quality and performance (and no fingerpointing)*
- Story 5: Manage the entire delivery pipeline with end-to-end visibility and dashboards
 - *Value: Enables end-to-end delivery metrics and visibility into bottlenecks*

Responsibilities

Pre-DevOps

- Developers produce the source code
 - *Do not care about the impact on the overall system in production*
- IT teams operate the system and ensure the quality of service
 - *Do not care about the performance of the code*



@rahuldighe

Post-DevOps

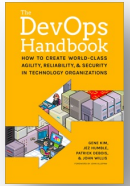
- Shared responsibilities with all stakeholders in the same boat
- *"You build it, You run it."* - Walter Vogels, Amazon CTO



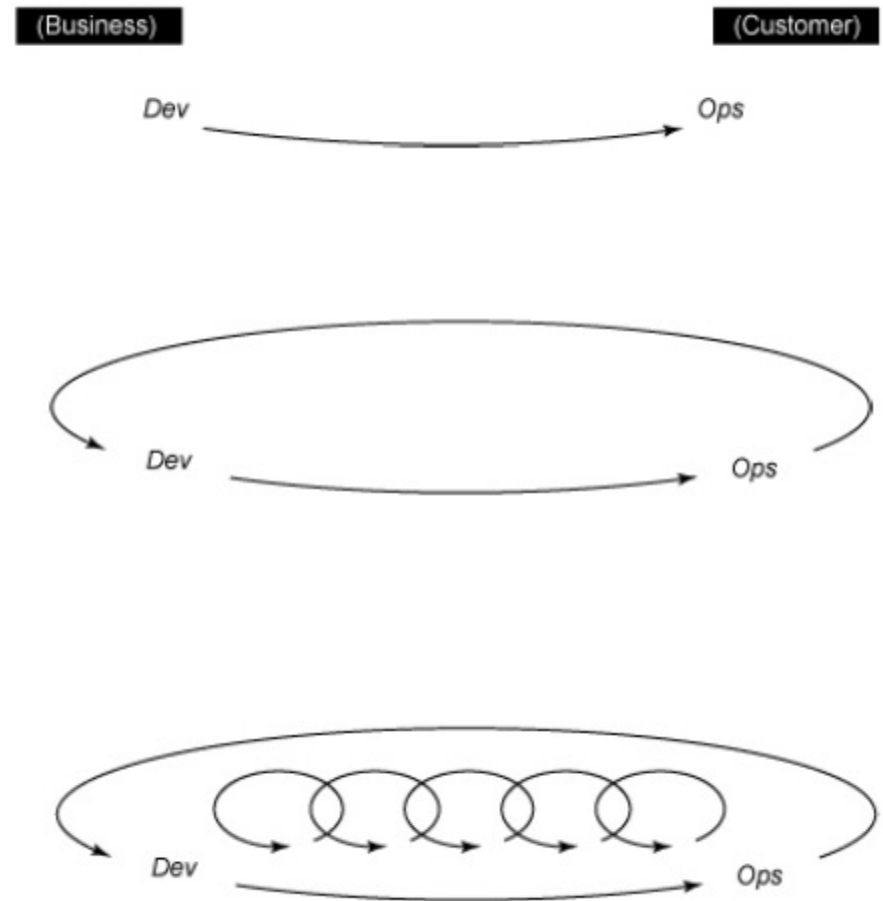
Expected Benefits

- Faster time-to-market/delivery times that improves ROI
- Engaged, empowered cross-discipline teams
- Stable/reliable operating environments
- Early detection and faster correction of defects
- Improved quality

DevOps: 3 Basic Principles

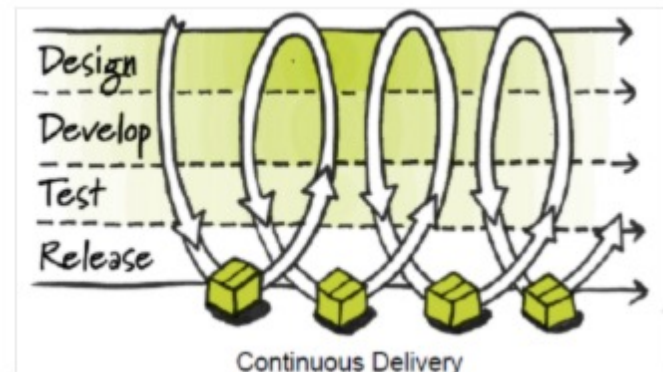
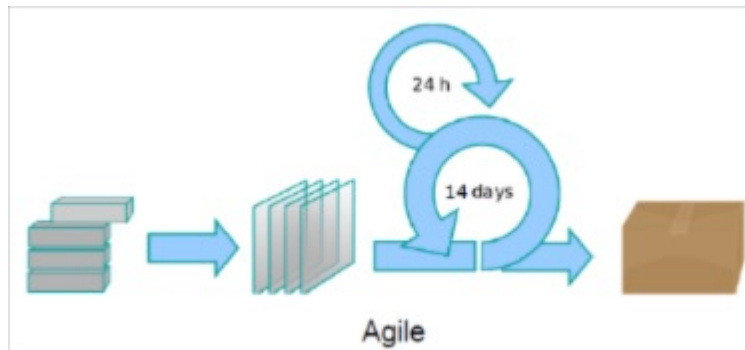


- System thinking
- Amplify feedback loops
- Culture of continual experiment and learning



DevOps vs. Agile

- DevOps is especially complementary to the Agile software development process.
 - extends and completes the continuous integration and release process
- DevOps enables a far more continuous flow of work into IT Operations
 - Avoid situation where development delivers code every two weeks but it's deployed only every two months



Some DevOps Principles

- Observability
- Stateless architecture
- Reproducibility and replicability
- Accountability
- Software lifecycle automation

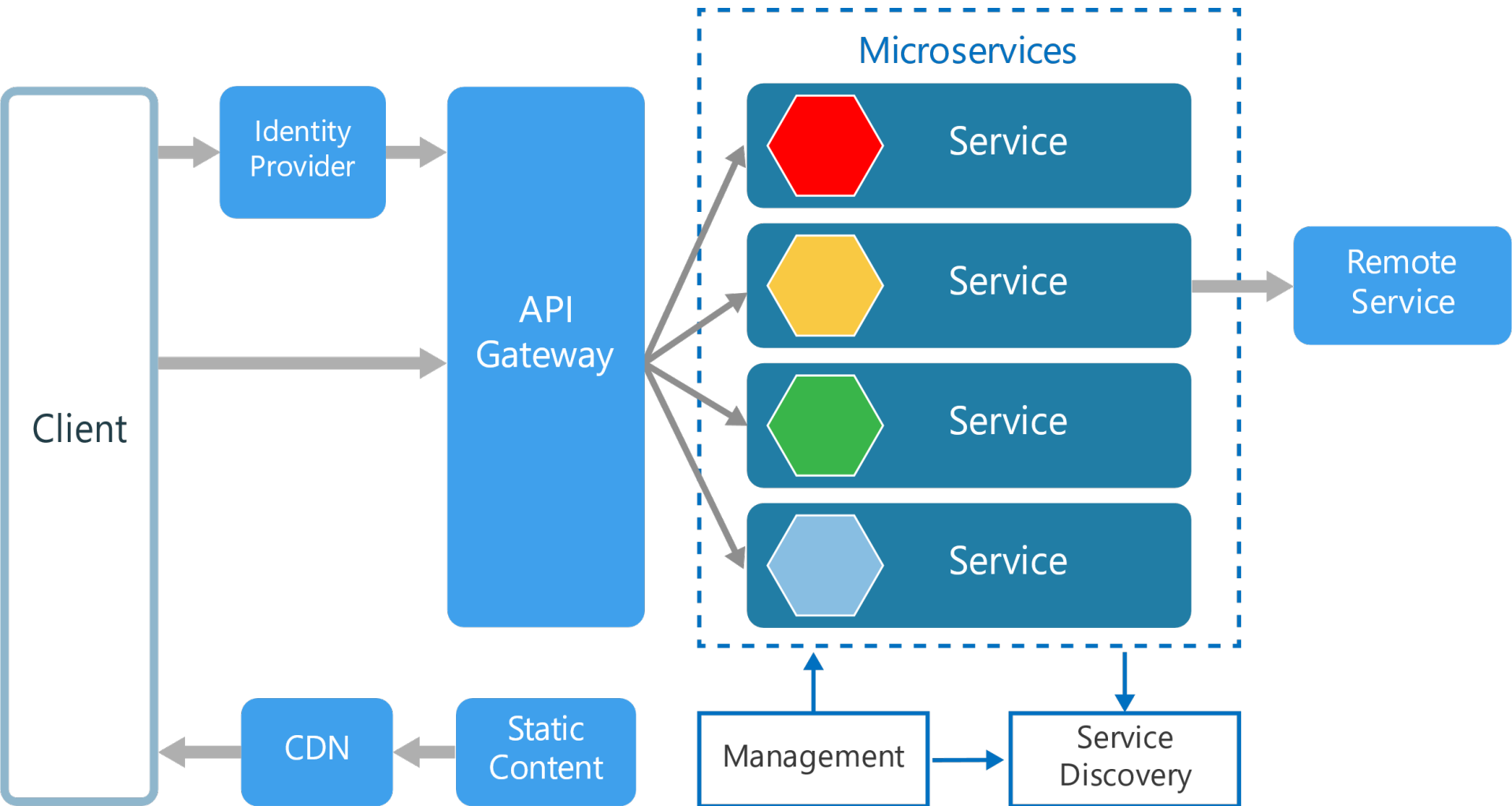
Some DevOps-Friendly SE Principles

- Cloud-native applications and microservice architecture (*e.g., quarkus*)
- Static analysis and test automation
- Collaborative development (*e.g., gitlab*)
- CI/CD pipelines (*e.g., gitlab CI*)
- Container-based release engineering (*e.g., docker*)
- Delivery through orchestration systems & Infrastructure as Code (*e.g., Kubernetes + Ansible + GitOps*)
- Resilience engineering (*A/B Canary testing, Chaos eng.*)
- Monitoring (e.g., performance, availability...)
- Change management, hypothesis driven dev.

Common Attributes of Successful Cultures

- **Infrastructure As Code**
 - Full Stack Automation
 - Commodity Hardware and/or Cloud infra
 - Reliability in software stack
 - Datacenter or Cloud Infrastructure APIs
 - Core Infra Services
- **Application As Services**
 - Service Orientation
 - Lightweight Protocols
 - Versioned APIs
 - Software Resiliency (Design for Failure)
 - Database/Storage Abstraction
- **Dev/Ops/All As Teams**
 - Shared Metrics/Monitoring
 - Incident Management
 - Service Owners On-call
 - Tight integration
 - Continuous Integration
 - Continuous Deployment
 - GameDay

Microservice Architecture



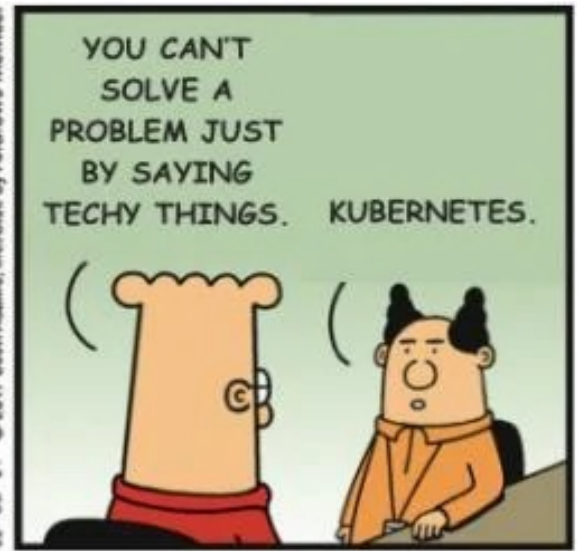
Towards Cloud-Native Applications



Dilbert.com @ScottAdamsSays



11-08-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel



Definition

- Cloud-native is an approach to building and running applications that exploits the advantages of the cloud computing delivery model
- Cloud-native is about how applications are created and deployed, not where
- Not related to public or private cloud
- Benefit: when companies build and operate applications in a cloud-native fashion, they bring new ideas to market faster and respond sooner to customer demands





In Practice

- **Microservices:** Architectural principle of distributed apps
- **Containers:** Shipping, and OS mechanism
- **DevOps:** Objectives, team mindset, continuous feedback loop
- **Continuous delivery:** mechanics

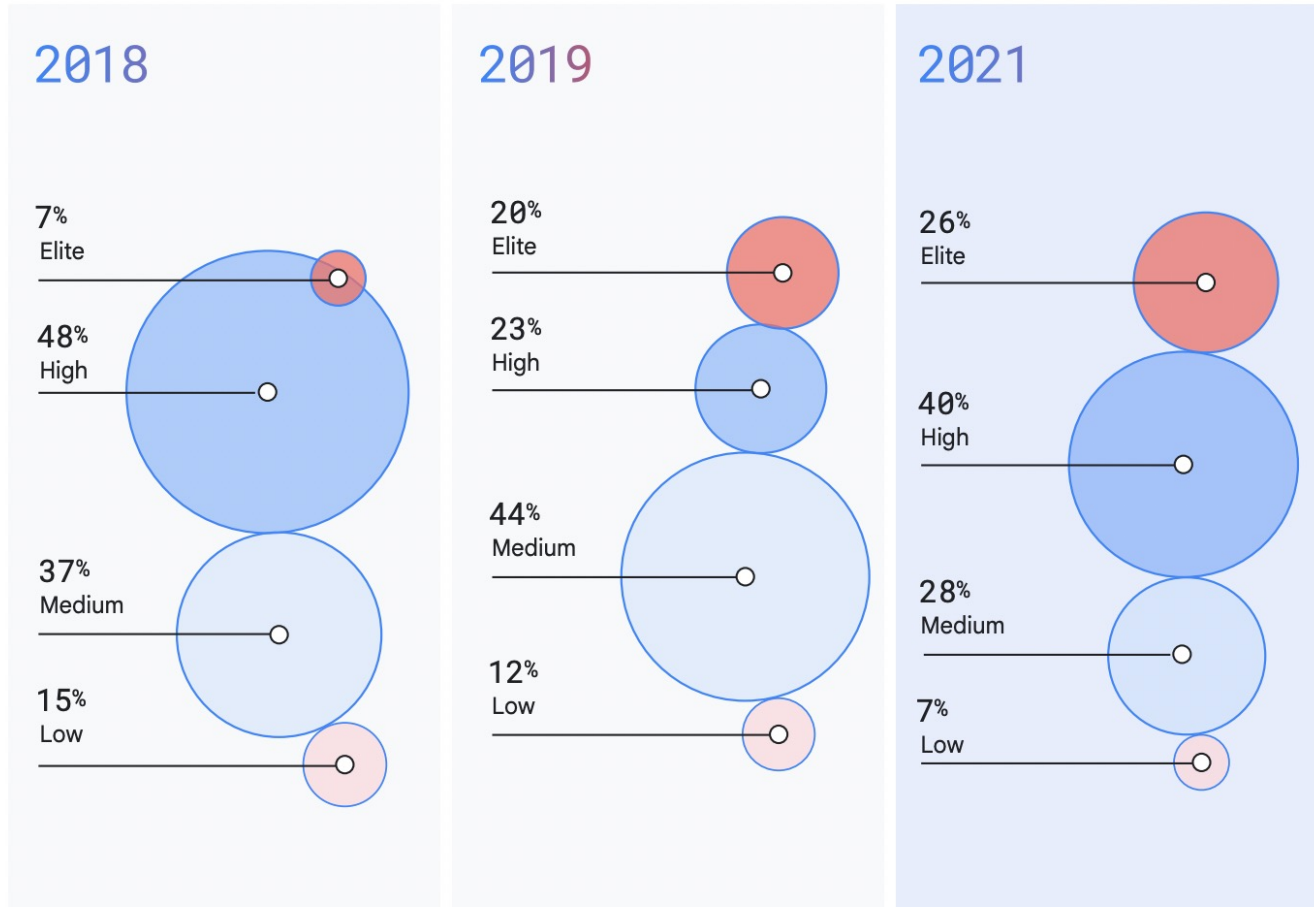
Purpose of the Configuration, Release and Deployment Pipeline

- **Visibility:** All aspects of the delivery system are visible to all team members promoting collaboration
- **Feedback:** Team members learn of problems as soon as they occur so that issues are fixed as soon as possible
- **Continually Deploy:** Through a fully automated process, you can deploy and release any version of the software to any environment

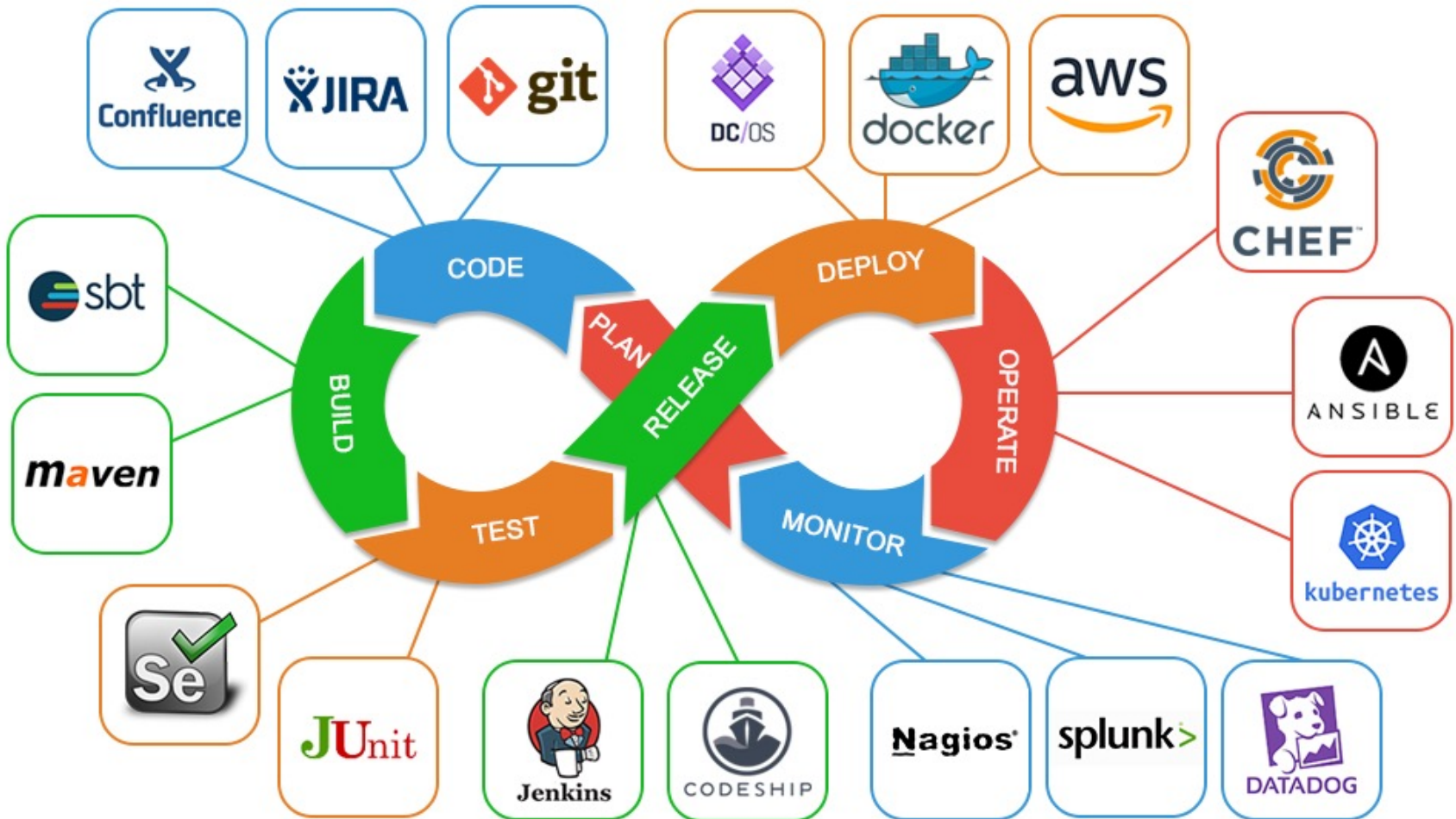
Performance Metrics

| Software delivery performance metric | Elite | High | Medium | Low |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|------------------------------------------|------------------------------------------------|--------------------------------|
| <p> Deployment frequency</p> <p>For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?</p> | On-demand (multiple deploys per day) | Between once per week and once per month | Between once per month and once every 6 months | Fewer than once per six months |
| <p> Lead time for changes</p> <p>For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?</p> | Less than one hour | Between one day and one week | Between one month and six months | More than six months |
| <p> Time to restore service</p> <p>For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?</p> | Less than one hour | Less than one day | Between one day and one week | More than six months |
| <p> Change failure rate</p> <p>For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?</p> | 0%-15% | 16%-30% | 16%-30% | 16%-30% |

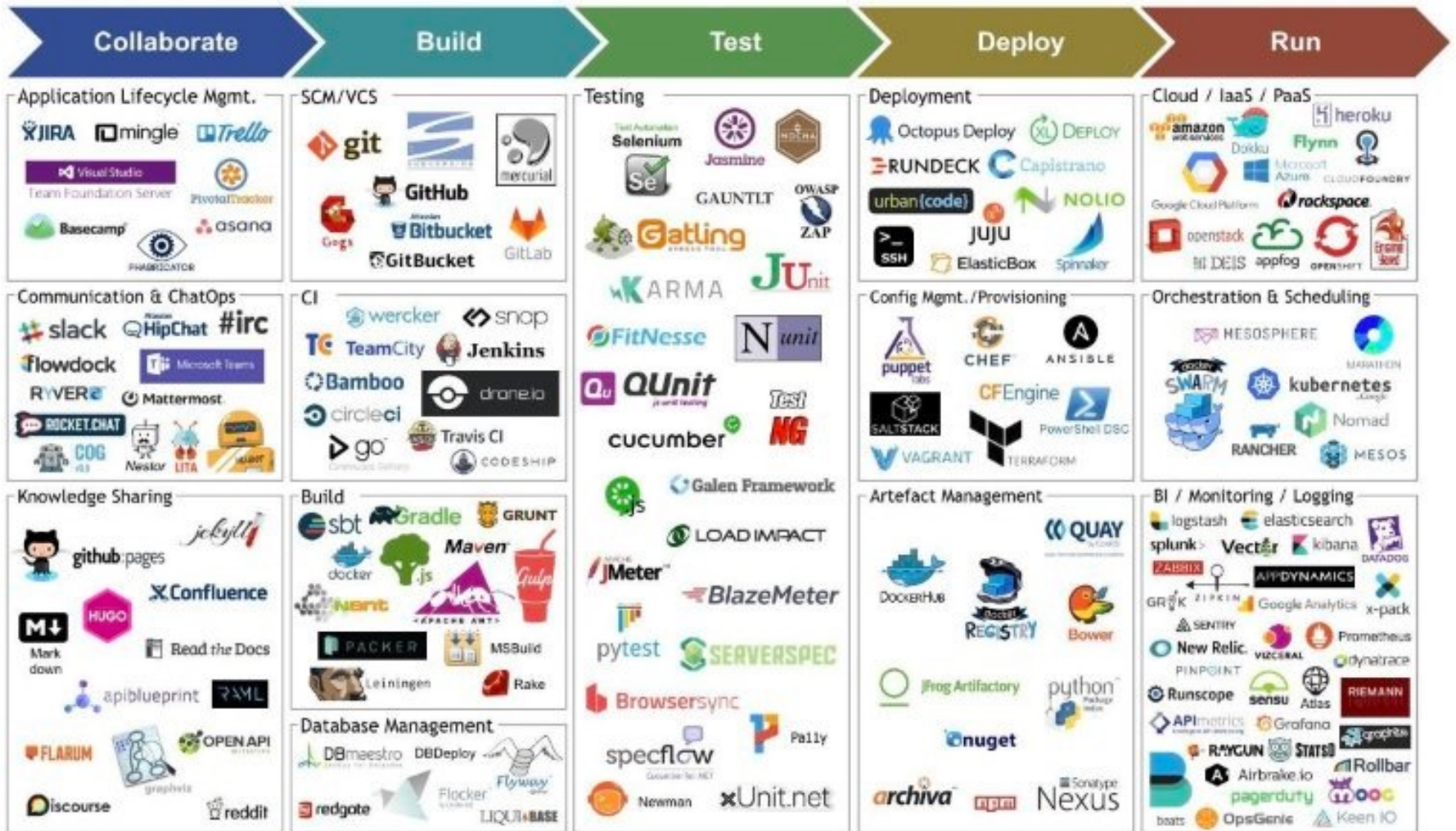
Performance Metrics



DevOps Tools



DevOps Tools



The adoption of DevOps is being driven by factors

- Use of **agile** and other development processes and methodologies
- Demand for an **increased rate of production releases** from application and business unit stakeholders
- Wide availability of **virtualized and cloud infrastructure** from internal and external providers
- Increased usage of data center **automation and configuration management tools**
- Increased focus on **test automation and continuous integration** methods;

The CALMS Conceptual Framework

- **Culture:** There is nothing fluffy about culture.
 - **Automation:** Automation is the idea that you should program everything.
 - **Lean:** Running lean means keeping everything to a minimum.
 - **Measurement:** If a team does not have visibility into everything, something will eventually go horribly wrong.
 - **Sharing:** Sharing is not just reporting facts, it is regular exchanging of ideas across teams.
-
- ✓ Often used as a maturity model
 - ✓ Proposed by Jez Humble



Key indicators of performers

Elite performers

Comparing the elite group against the low performers, we find that elite performers have...

973x

more frequent
code deployments

6570x

faster lead time
from commit to deploy

Yes, you read
correctly.
This is not an
editorial error.

3x

lower change failure rate
(changes are 1/3 less likely to fail)

6570x

faster time to recover
from incidents

Security Practice

| | |
|---------------------------------------------|-----|
| Test for security | 58% |
| Integrate security reviews into every phase | 54% |
| Security reviews | 60% |
| Build pre-approved code | 49% |
| Invite InfoSec early and often | 63% |

Comparing highest to lowest performers.

from <https://services.google.com/fh/files/misc/state-of-devops-2021.pdf>

Anti-patterns

- Management just saying we're doing DevOps
- Just changing job titles to DevOps
- Just merging dev and ops teams or creating a separate DevOps team
- Committing is done
- My responsibility ends here
- Devs blaming Ops; Ops blaming Devs
- Ops not involved early
- DevOps means Developers Managing Production
- It's not just automation or a tool (or set of tools)

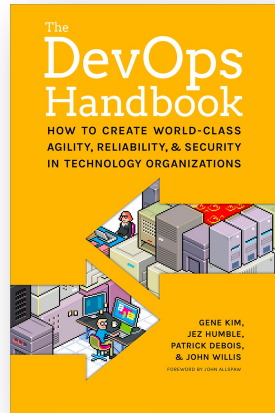
Conclusion

Take away principles

- observability
- stateless architecture
- reproducibility and replicability
- accountability
- software lifecycle automation

Going further

- Book:



- Initiative: <https://teachdevops.github.io>

- Research topics, e.g., pipeline management, continual learning, live analytics ops to dev., etc.

- Introduction
 - Some Facts on Modern Developments
 - State of the Practice (large-scale, polyglot, short term delivery...)
 - DevOps: current scope, concepts and principles
- Get ready for DevOps!
 - Execution platform (e.g., virtual machines, containerization and clouds),
 - Software architecture (microservice, stateless),
 - organizational concerns (gitflow and branching, continuous improvement...)
- Test automation, incl. flaky test, code and test coverage, mutation analysis, fuzzing.
- Build Management, Configuration Management, Release Management
 - Software Build (e.g., Maven)
 - Software Delivery (e.g., Docker, Docker Compose)
 - Software Deployment (e.g., Kubernetes)
 - Continuous Integration, Delivery and deployment (e.g., Jenkins)
 - Infrastructure as code
- Measurement: Logging, Tracing and Monitoring (e.g., LogStash, OpenTracing, Sonar, Kibana)
- A/B and Canary Testing
- Resilience engineering / testing (e.g., Chaos Engineering)

<http://combemale.fr/course/devops/>

Further Material

- Books



...

- Conferences

- [DevOpsCon](#)
- [DevOpsDays](#)
- [DevSecDays](#)
- [KubeCon + CloudNativeCon](#)

- A lot of high-quality posts (e.g., medium)