Hybrid DevOps

Combine mainframe and distributed in one DevOps pipeline

IBM

—

Agnes ten Brink

IBM Benelux - Technical Sales DevOps on z

agnes.ten.brink@nl.ibm.com

So what is DevOps ?

DevOps is a set of **practices**

intended to **reduce the time** to move change to production

while ensuring **high quality**

Why DevOps ?

- Shorten **time to value** – *“from concept to cash”*

- Increased **capacity to innovate**

- Enhanced **customer experience**

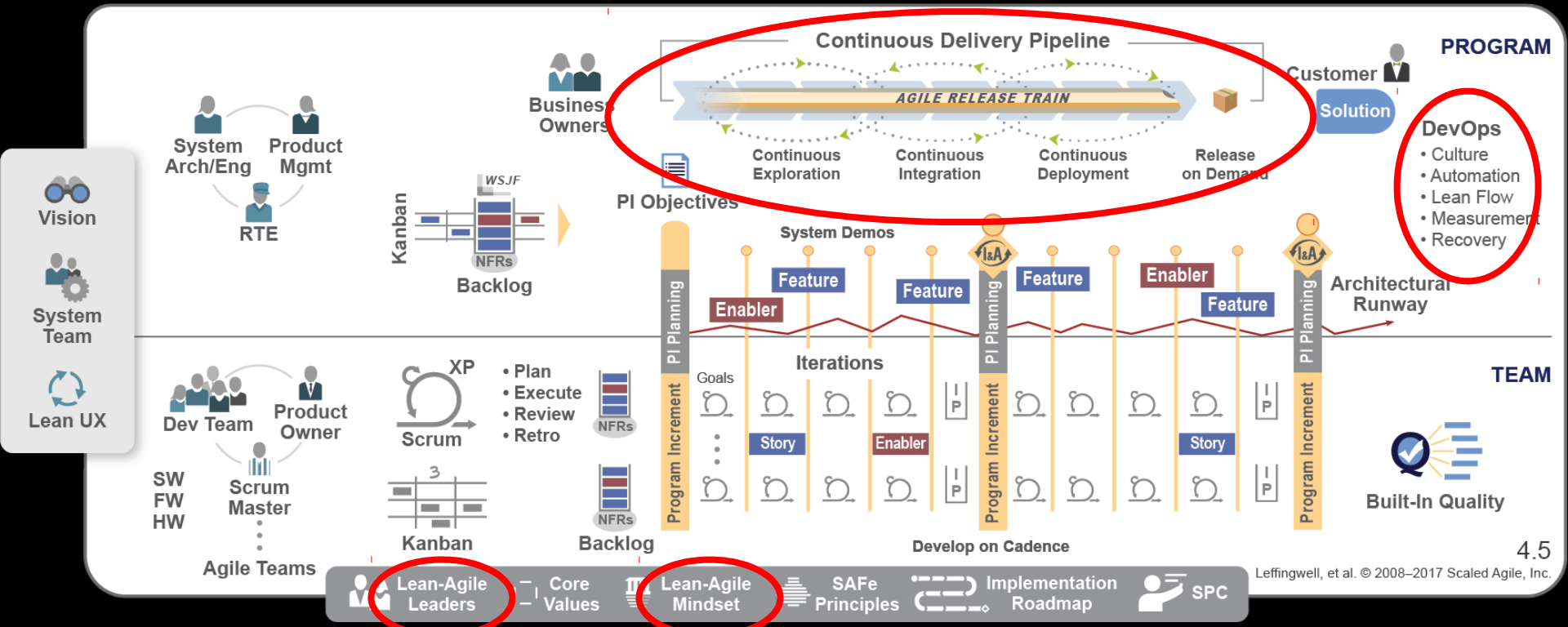
Why DevOps ?

Rise of Devops influenced by:

- Spread of **Agile/LEAN** development
- Introduction of (private) **Cloud**
- Move to a **Microservices** architecture

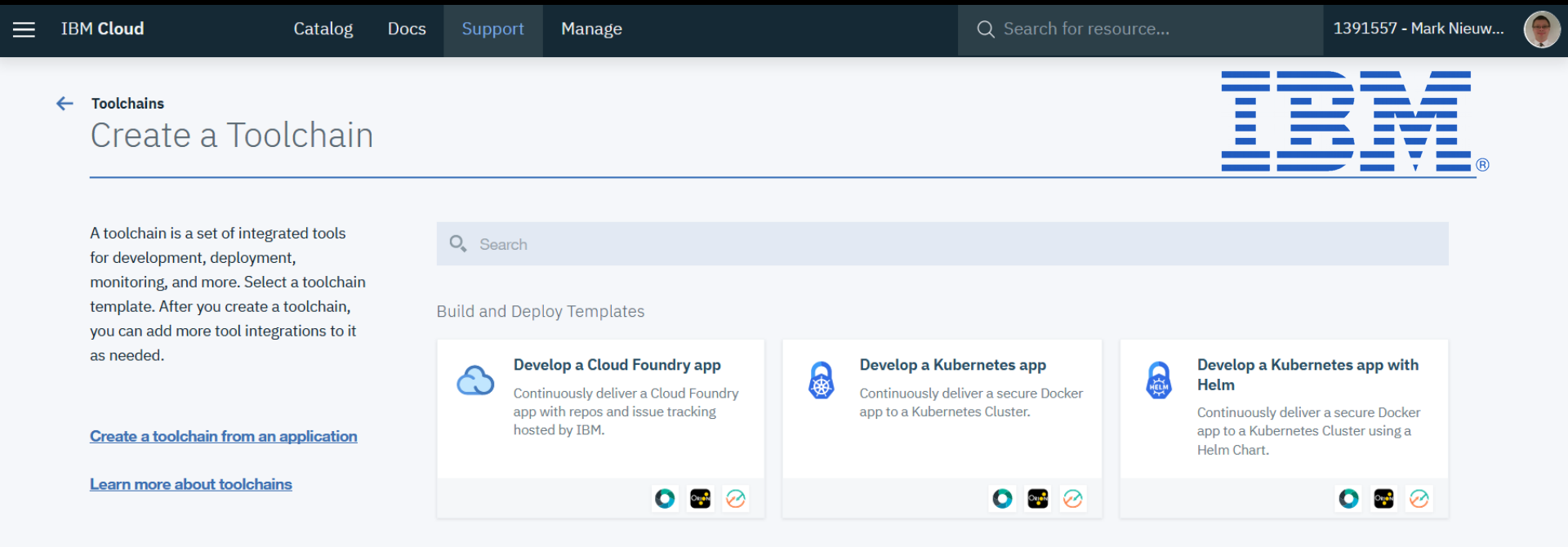
Agile at scale leads to demand velocity in infrastructure

SAFe (Scaled Agile Framework) includes DevOps



Cloud simplifies automatically provisioning (test) environments

PaaS solutions allow you to start quickly



Azure DevOps

ORACLE Cloud

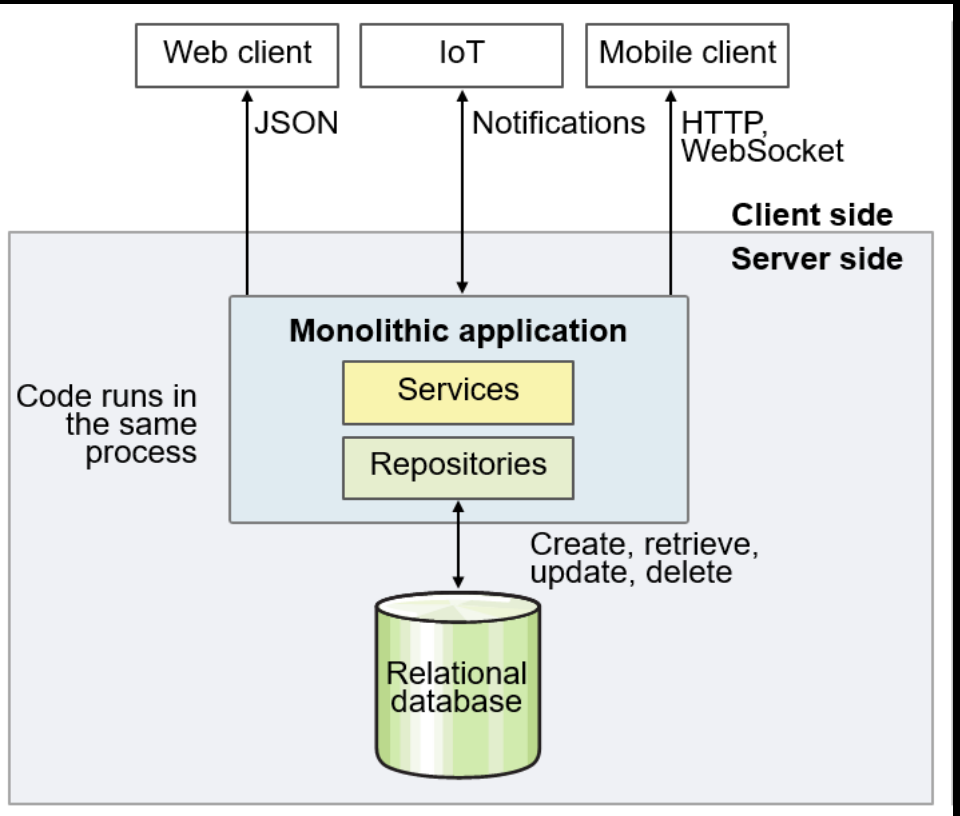
aws

Google Cloud

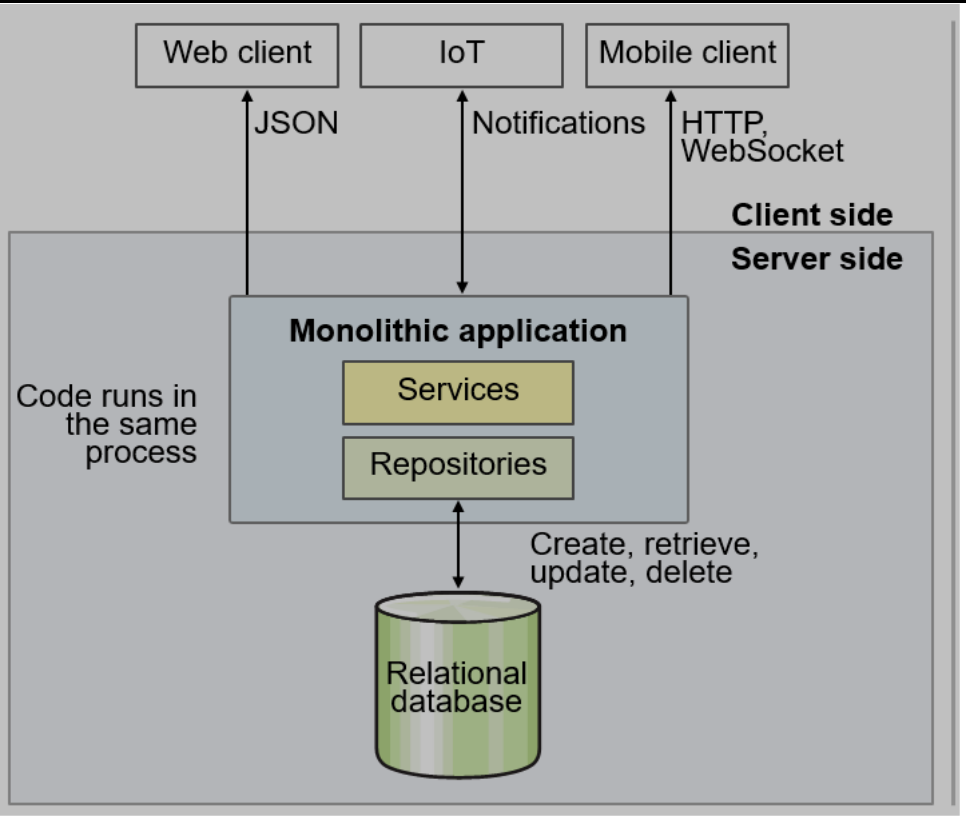
IBM Z cloud

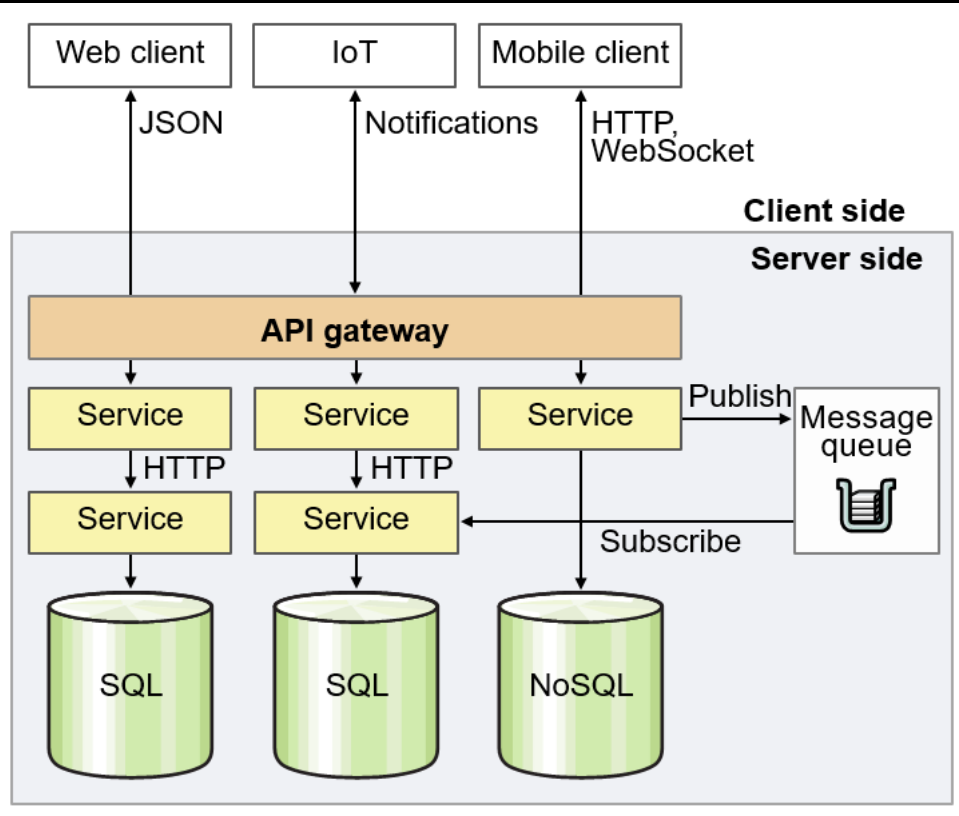
IBM Cloud Private: z/OS Cloud Broker

MicroServices break up applications in manageable parts to improve business agility, and leading to many microservices



MicroServices break up applications in manageable parts to improve business agility, and leading to many microservices





DevOps is about **Culture**...

It´s about

- **collaboration** across roles
- focus on **business**

  not departments
- **learning** by experimenting

It´s all about **people**

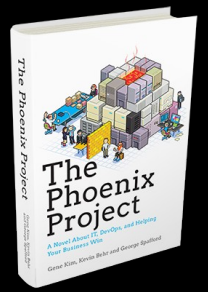
DevOps is about **“the three ways”**

1st way **Systems Thinking**

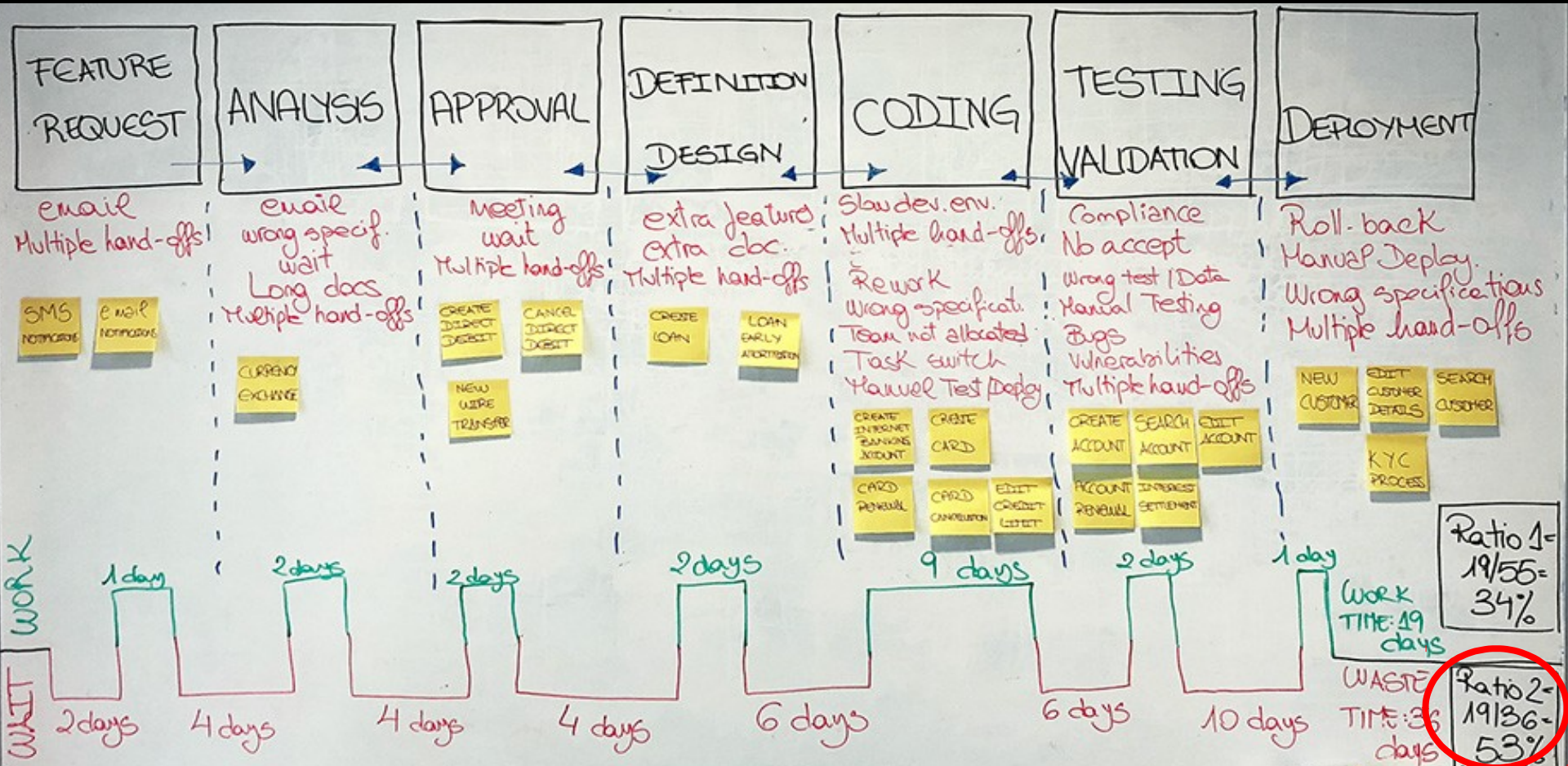
2nd way amplify **Feedback Loops**

3rd way **Continuous Experimentation & Learning**

… this is explained in book “The Phoenix Project”



Systems Thinking: Value Stream Mapping



Systems Thinking: Value Stream Mapping

*… optimize using **TIM WOOD** (LEAN manufacturing)*

Transport - *transfers, handoffs*

Inventory - *backlog , work in progress (reqs, code)*

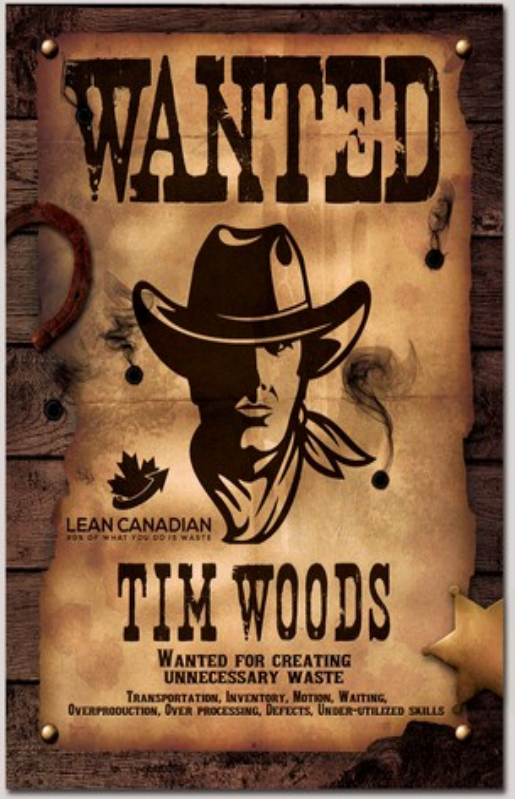
Motion - *task switching, interruptions*

Waiting - *waiting, approvals, decision making*

Overprocessing – *relearning, forgotten knowledge*

Overproduction - *extra features, goldplating*

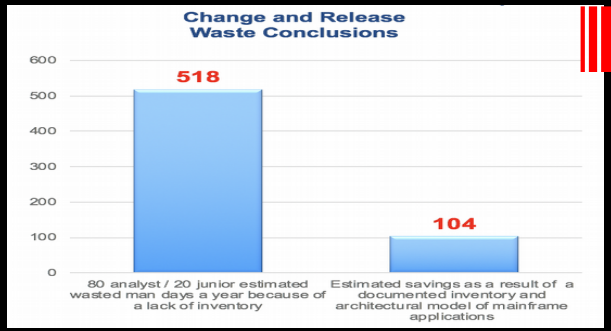
Defect – *defects, rework, re-clarification*



Systems Thinking: Value Stream Mapping: The Workshop

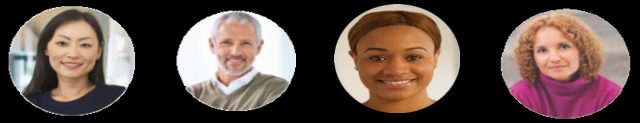
Day 1

Point in Time View of Practices, Waste and Where to Start

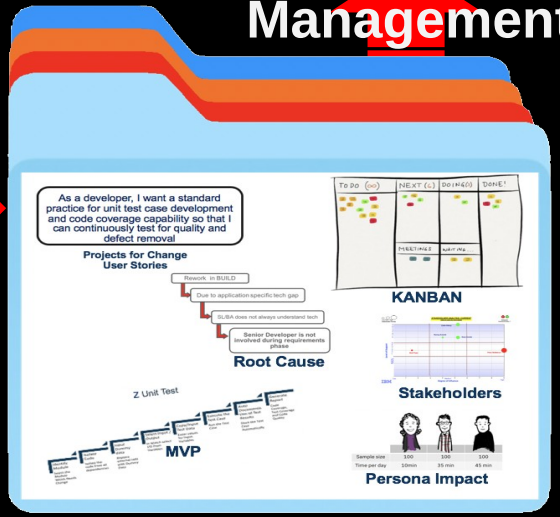


Quantification of Waste

Day 2



Team Presentations to Senior Management



Now have up to 6 projects to begin Dev/OPS change

Systems Thinking: e.g. Automated Testing

Automatically testing a release

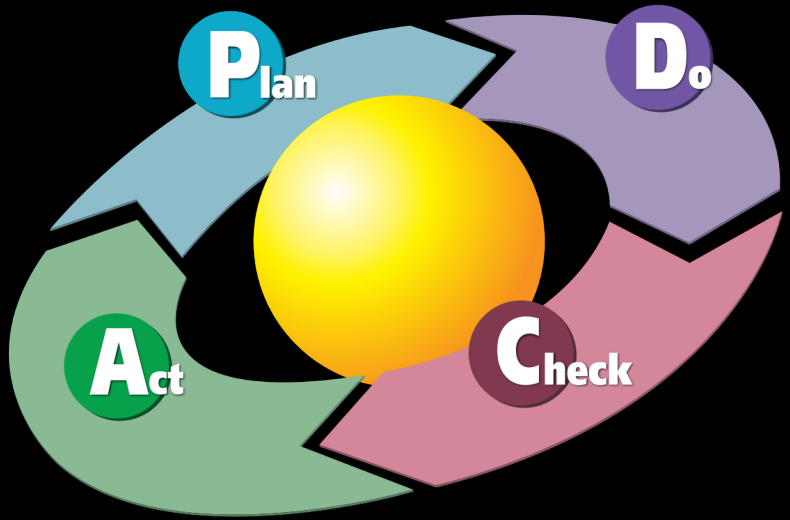
- removes a handover
- reduces / eliminates queueing

Testing may include:

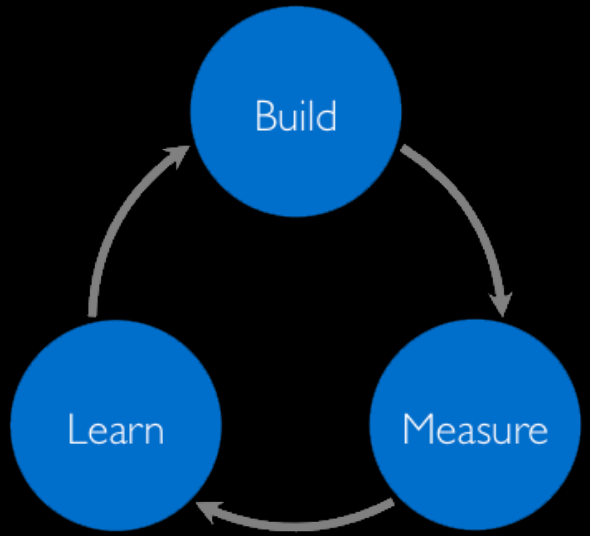
- functional testing
- performance testing
- pentesting
- ...

IBM

Feedback loops



Deming cycle



Lean Startup

Feedback loops (and Shift Left testing)

"80% of development costs are spent identifying and correcting defects" **

During the Coding or Unit Testing phases

$80/defect

During the BUILD phase

$240/defect

During Quality Assurance or the System Test phases

$960/defect

Once released into production

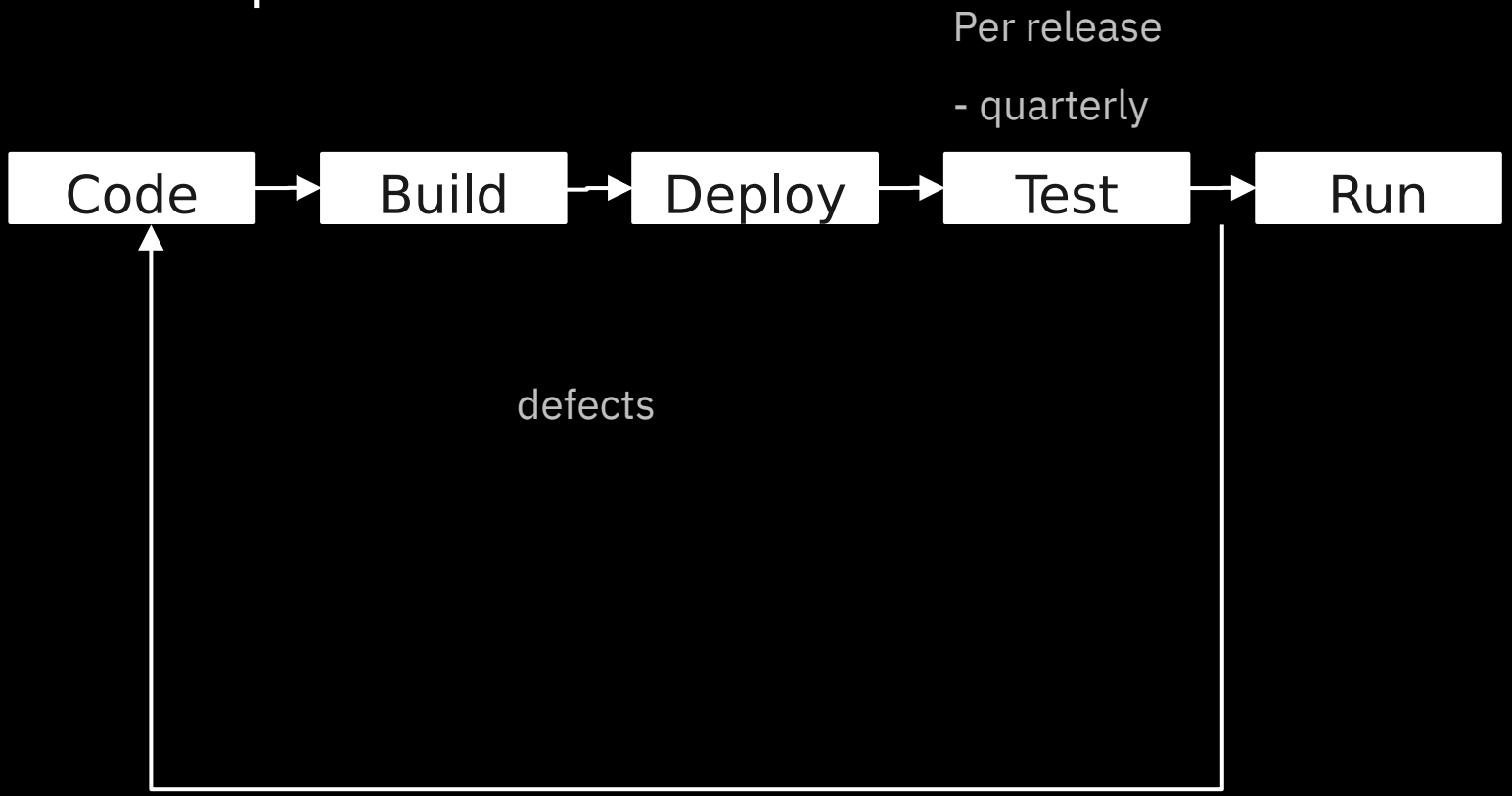
$7,600/defect
+
Law suits, loss of customer trust, damage to brand

**National Institute of Standards & Technology

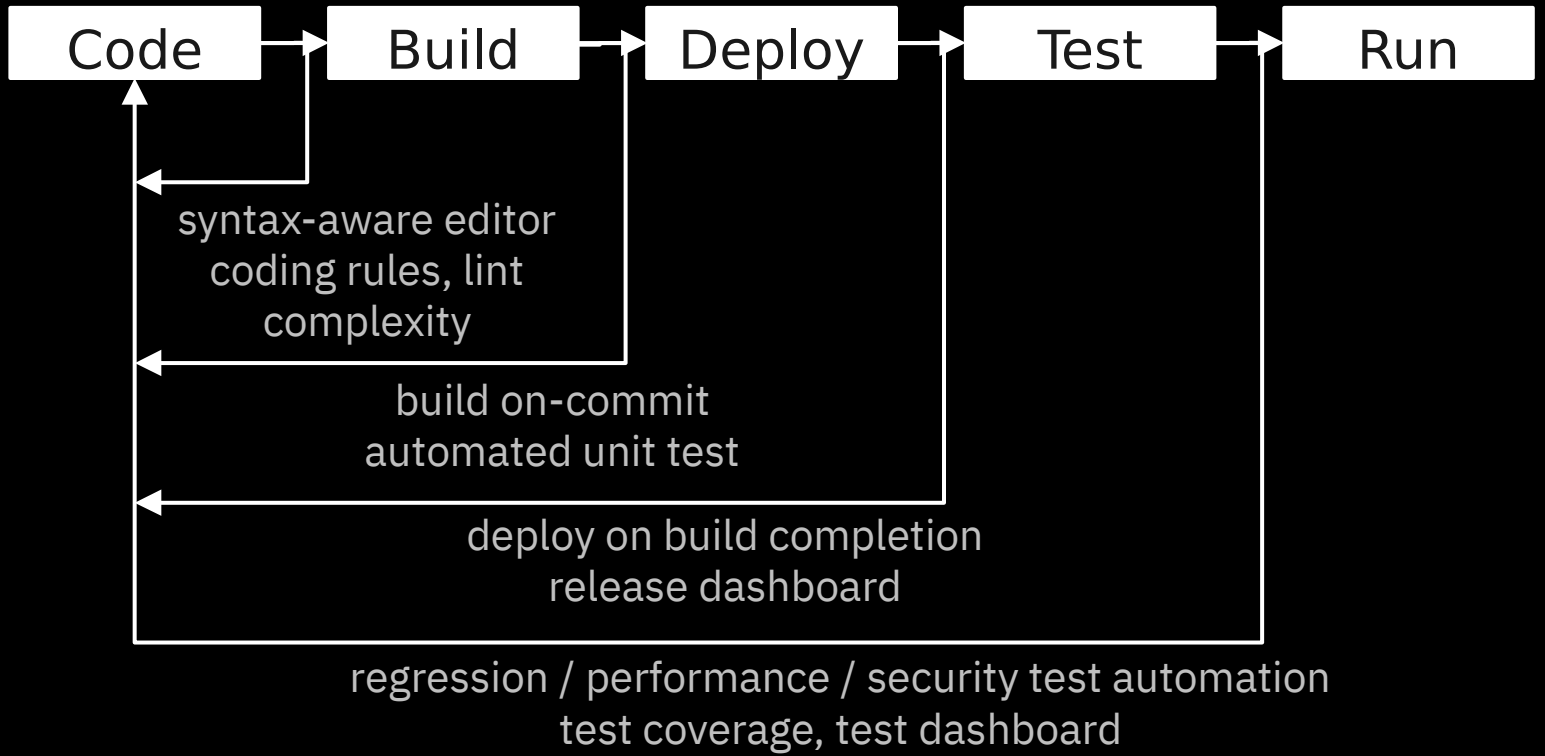
Source: GBS Industry standard study

Defect cost derived in assuming it takes 8 hours to find, fix and repair a defect when found in code and unit test.
Defect FFR cost for other phases calculated by using the multiplier on a blended rate of $80/hr.

Feedback loops

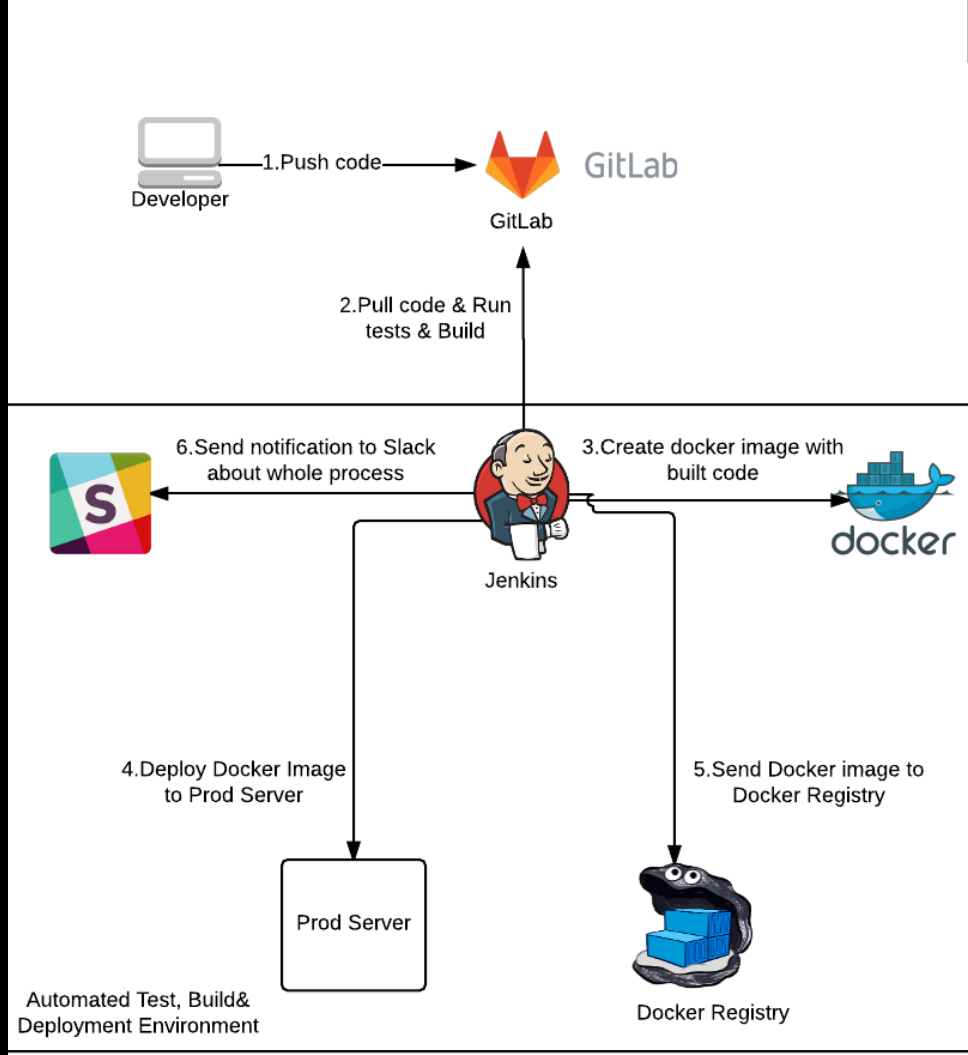


Feedback loops

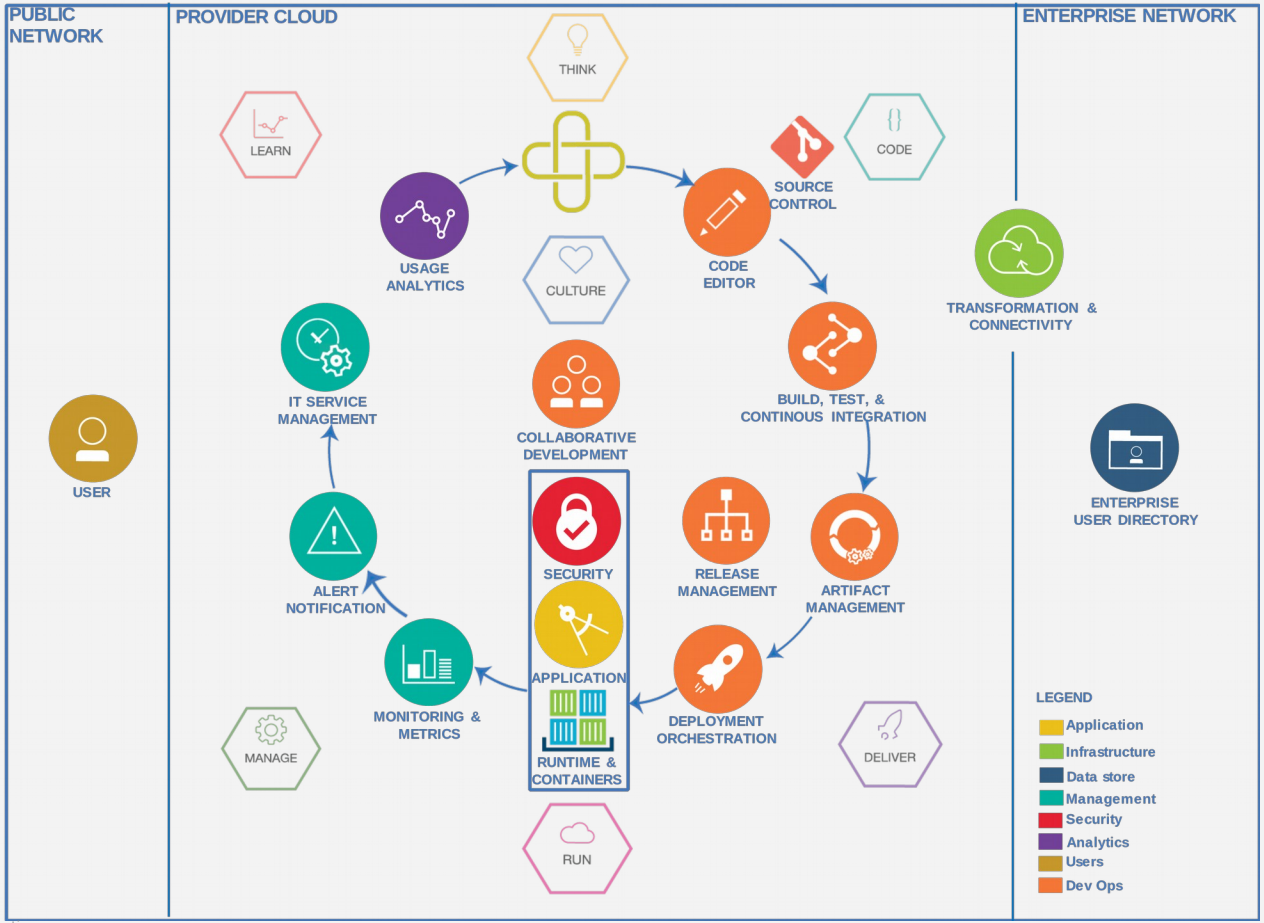


Feedback Loops: inform team using Slack

Slack provides build feedback to team



Cont. experiment and learn: DevOps Reference Architecture

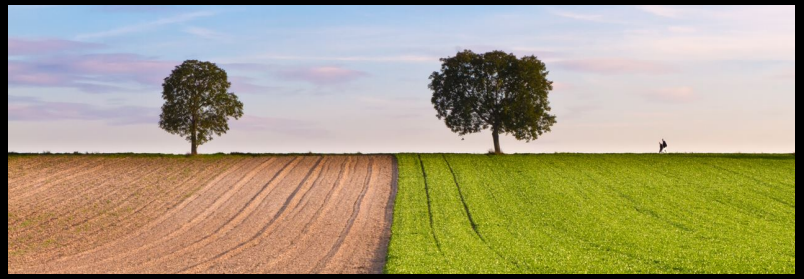


To design a toolchain

- Clarify **requirements**
- Make **design decisions**

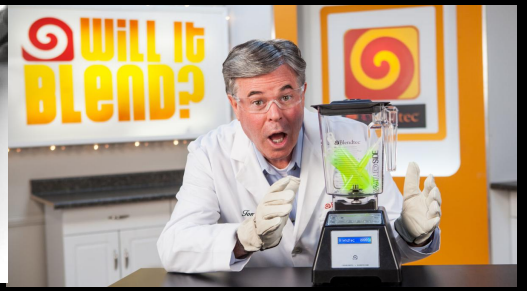
#architecture

Build for **brownfield** or **greenfield** ?



What **existing tools** do you have ?

Will the toolchain **integrate** ?

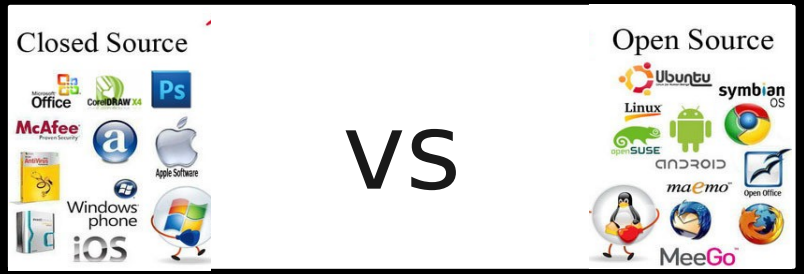


Do we prefer **open source** ?

Or **vendor based** tools?

What do **other companies** do ?

Do we have the right **skills** ?



So what does this look like in a hybrid application landscape ?

... consisting of applications on mainframe and on distributed

DevOps Reference Architecture

Requirements and design

Typical tools include:

- **Application Discovery**
- Doors Next Gen
- ....

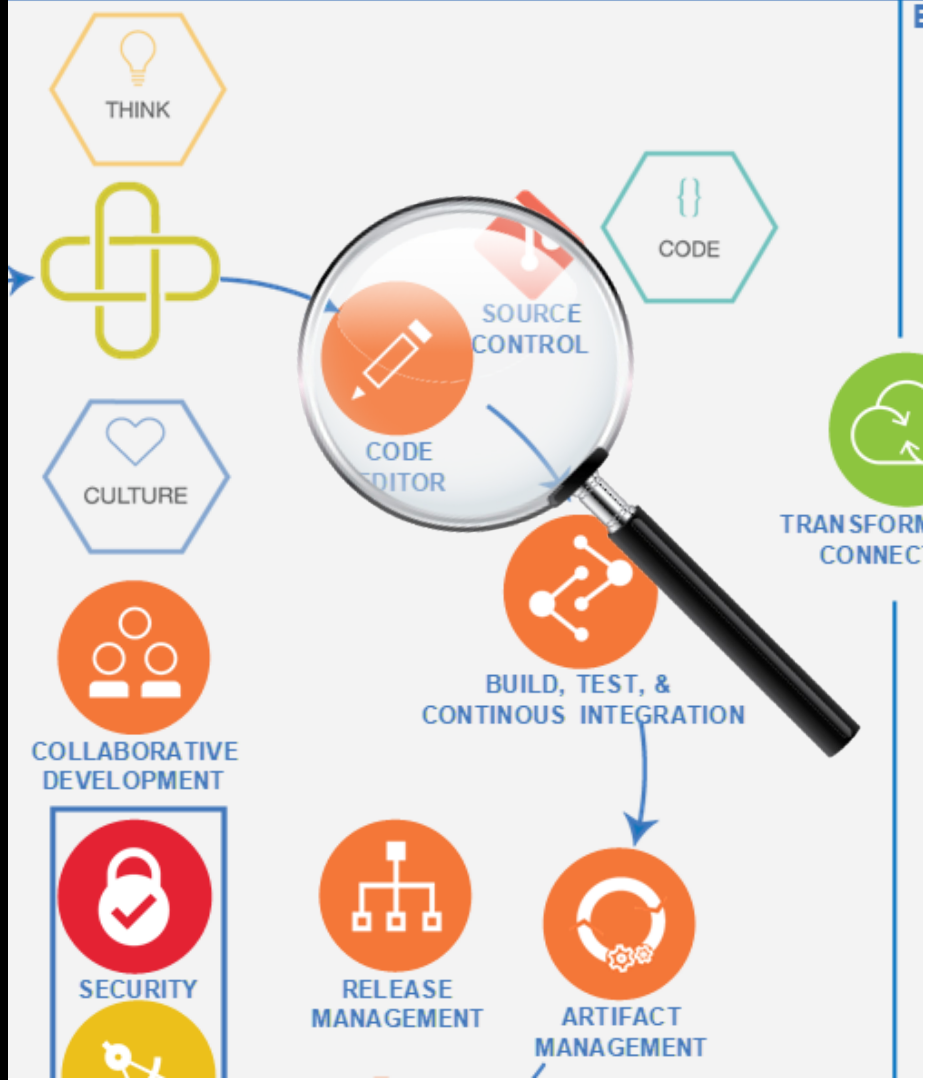
Source control / code editor

Code Editors

- IDE (eclipse based or otherwise)
  - **IDz, Topaz, ...**
  - **jUnit, zUnit**
  - **Application Discovery**
  - **Sonarqube**

Version Control

- **RTC, Git**, BitBucket, SubVersion



Build, test & continuous integration

Build

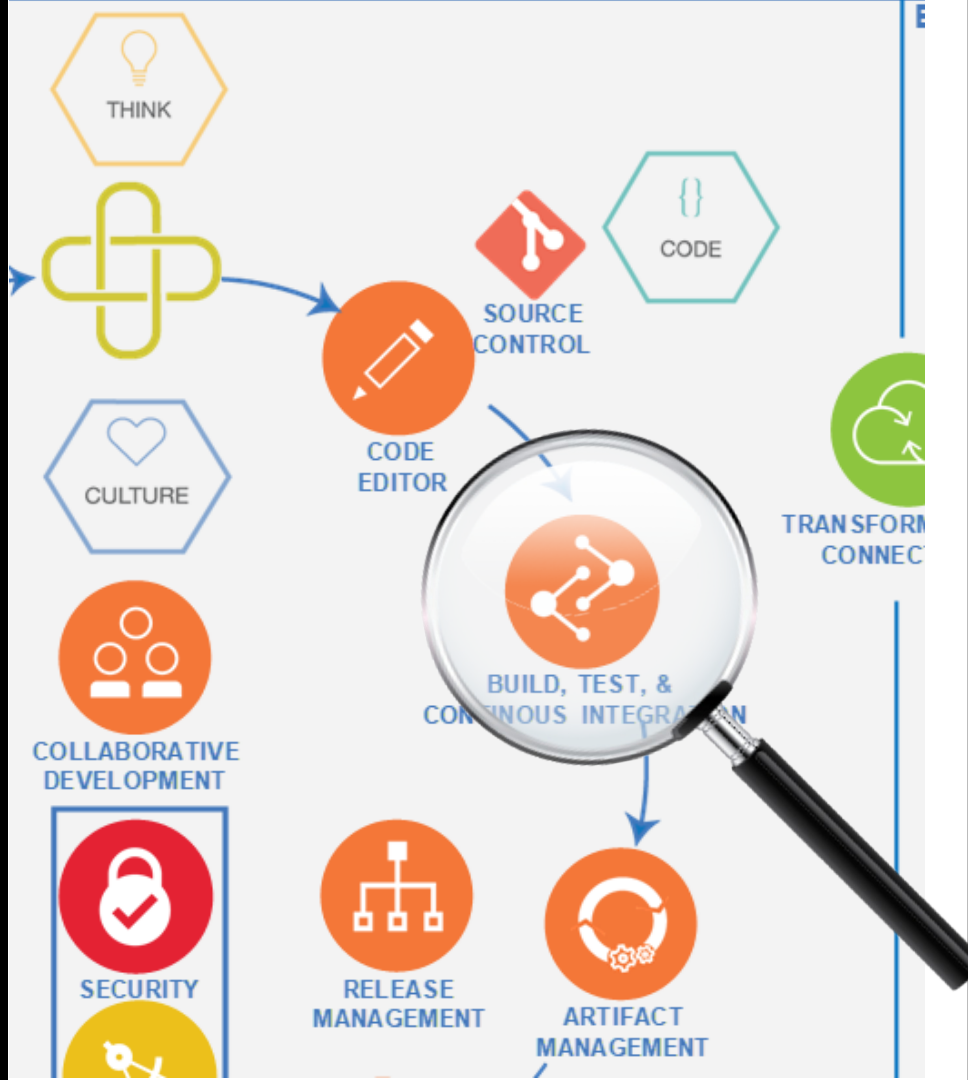
- **IDE (eclipse based or otherwise)**
- **Dependency Based Build**
- Maven, Gradle, …

Test

- **Rational Test Workbench**
- **zUnit**

Continuous Integration

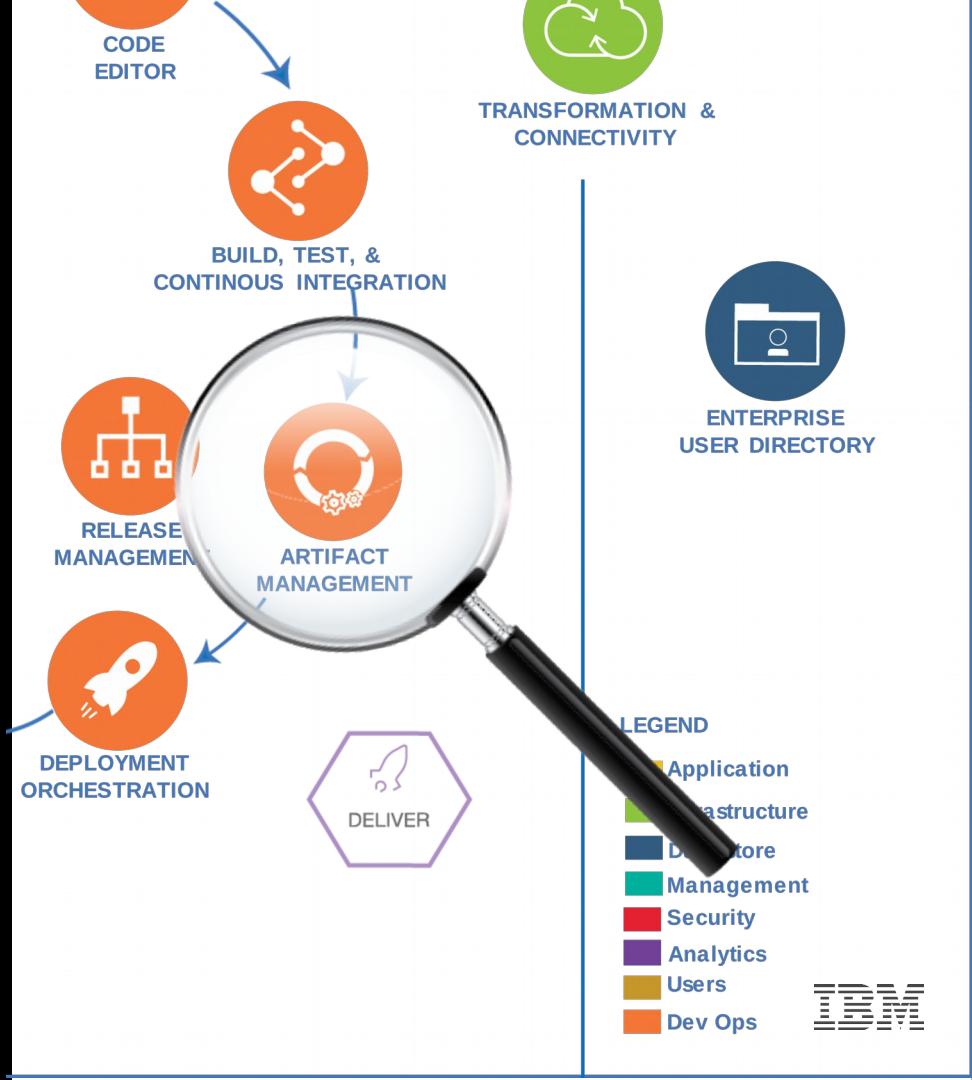
- **Jenkins**, Travis, CircleCI,



Artifact Management

Typical tools

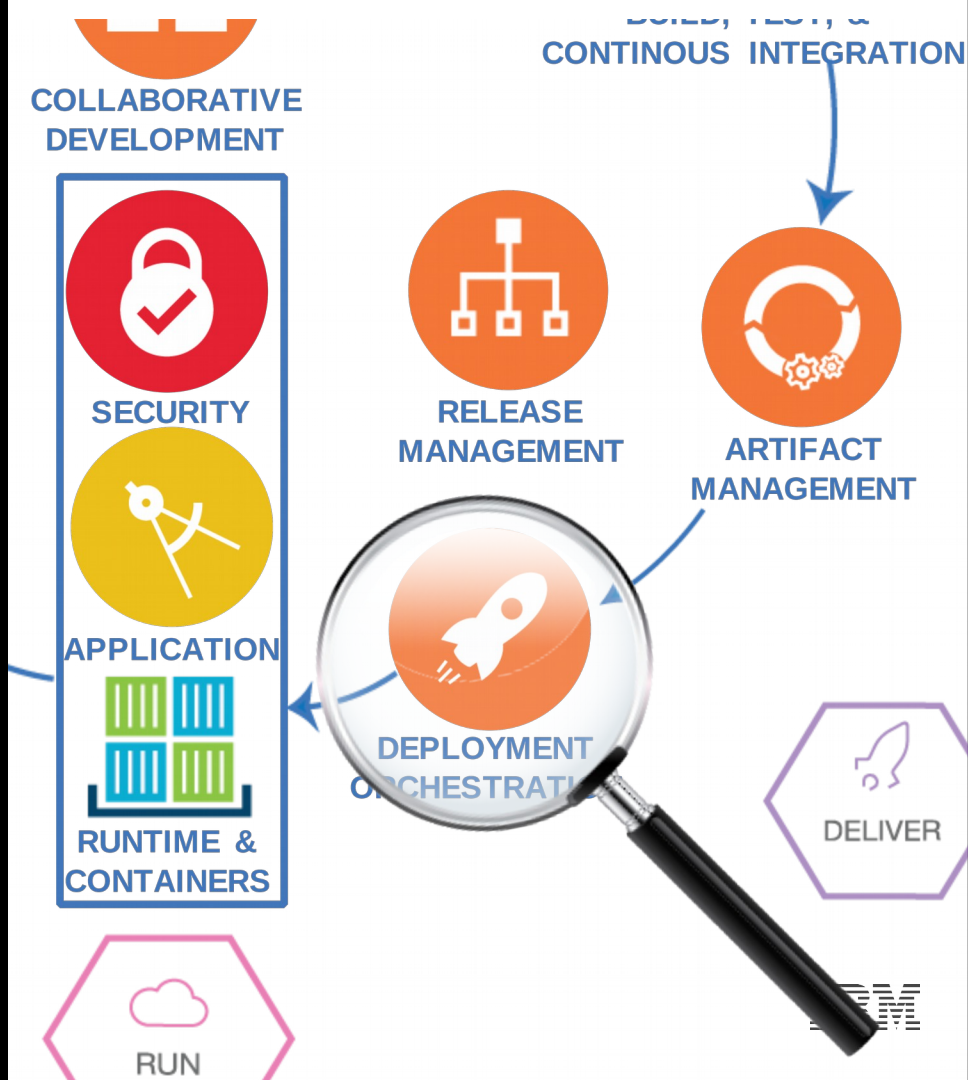
- **Rational Team Concert**
- (Jfrog) Artifactory
- (Sonatype) Nexus



Deployment orchestration

Typical tools include:

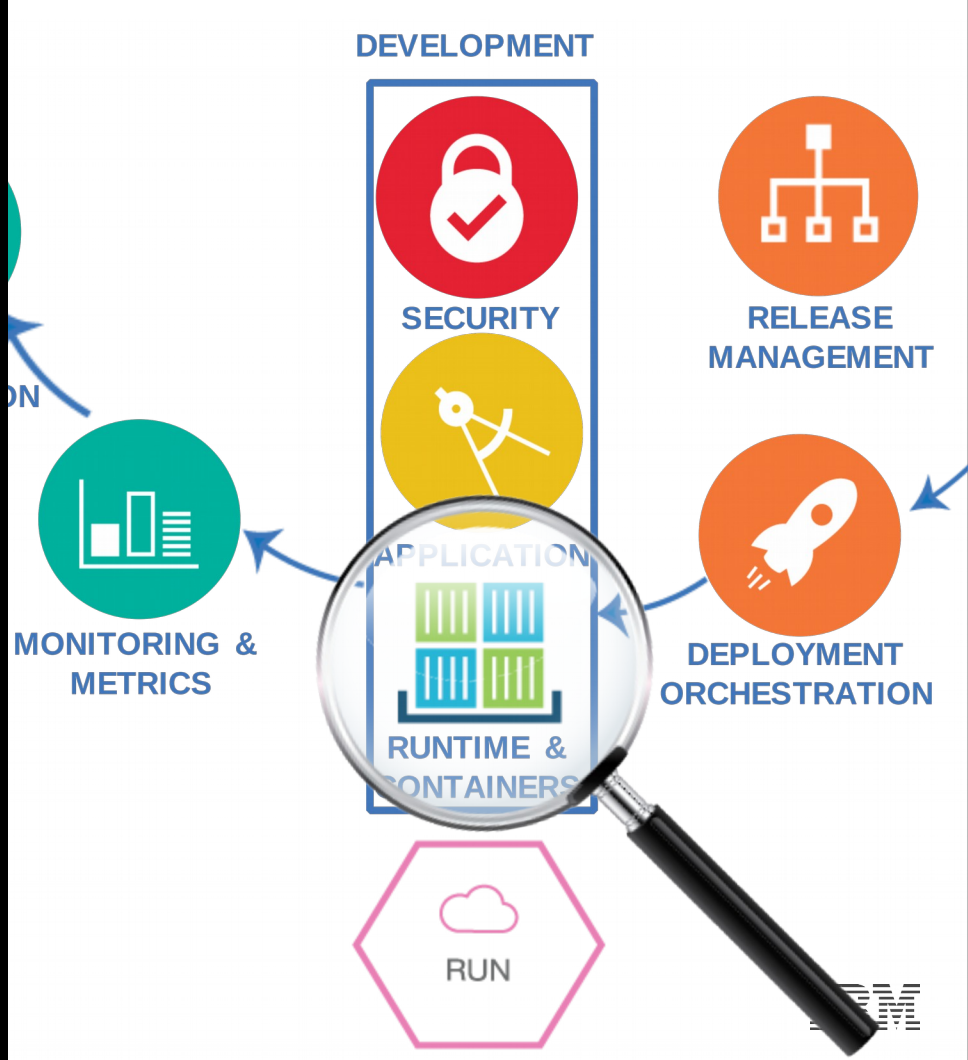
- **UrbanCode Deploy**
- XL Deploy
- Bamboo



Runtime & Containers

Typical target environments include:

- Virtualized (VM) environment
  - Vmware (Windows, Linux)
  - **ZD&T (z/OS)**
- Containerized environment
  - **Docker** (soon...)
  - Kubernetes
- ... Cloud ...



Conclusions & recommendations

- Defining a toolchain is essentially **architecture work**
  - *… so use NFRs, architecture decisions, …*

- Creating a mainframe+distributed toolchain is **feasible**
  - *… tools supporting all required technologies help*
  - *… it requires co-operation across technologies*

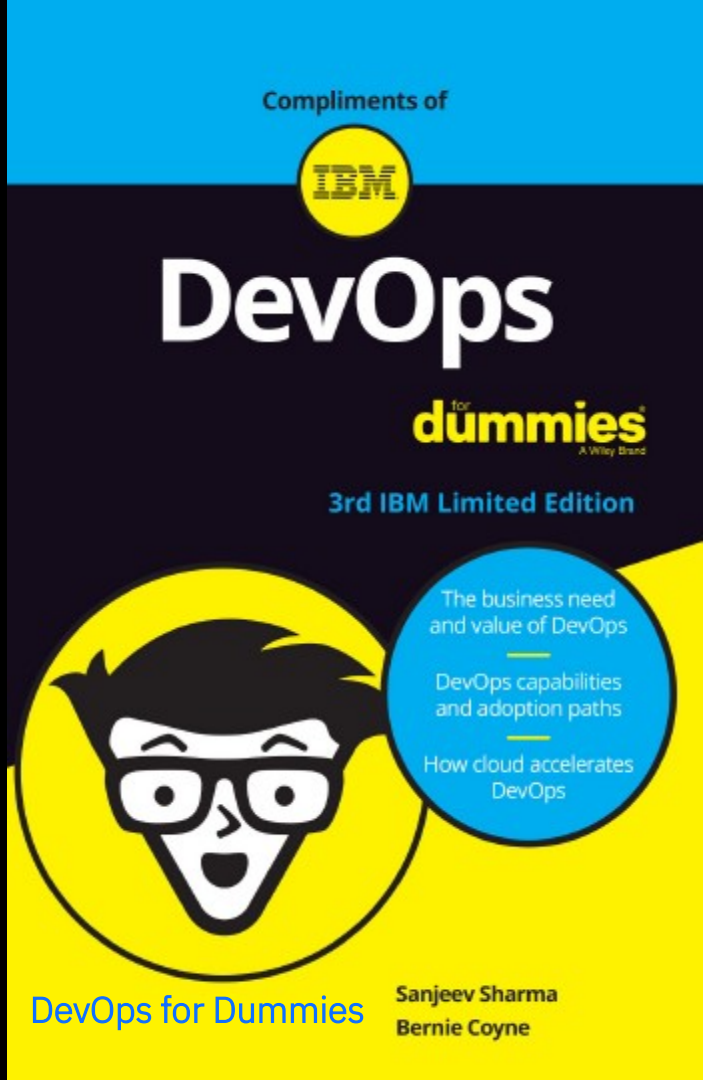
Q&A

Questions ?

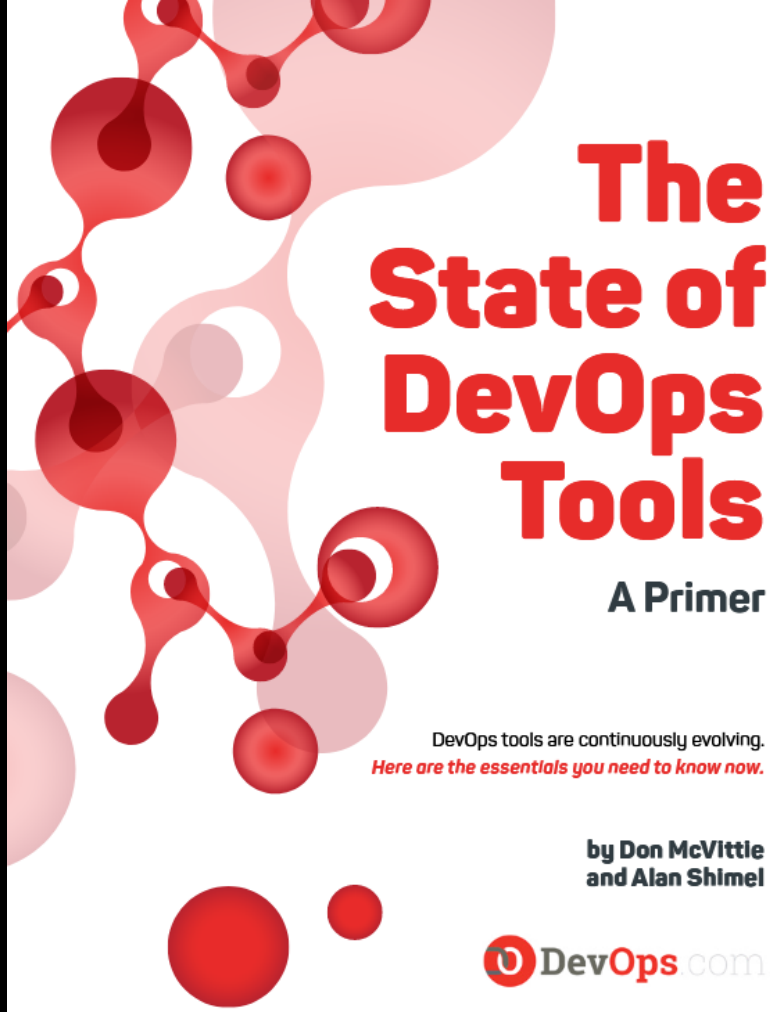
Any requests ? Tell us !

agnes.ten.brink@nl.ibm.com

References:
DevOps for Dummies

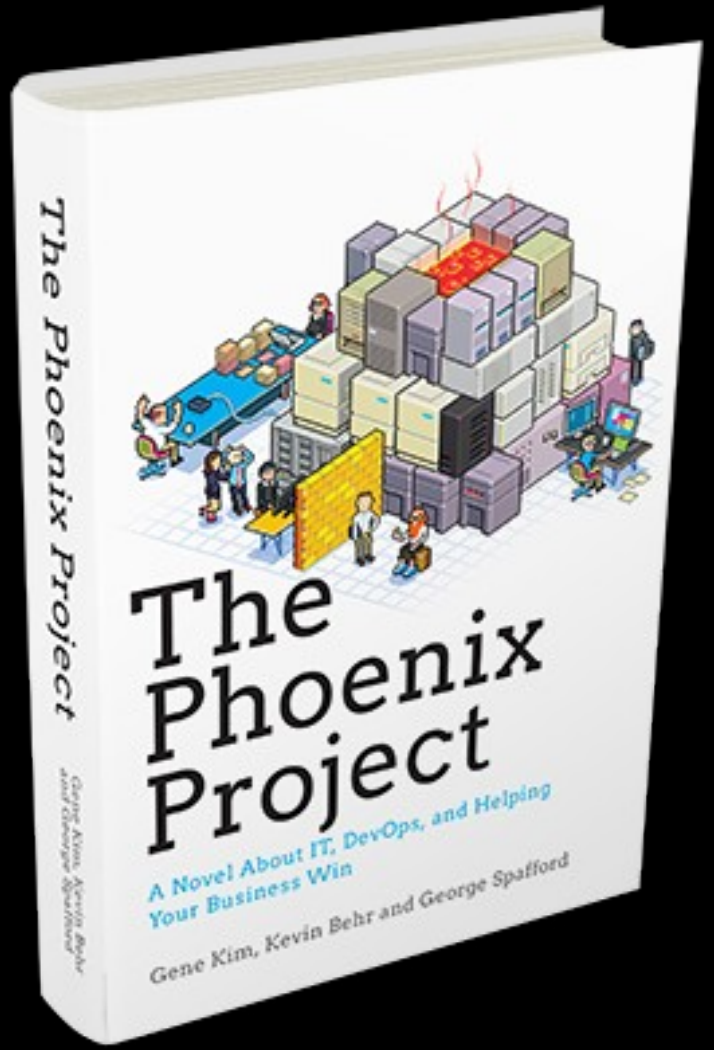


References
The State of DevOps Tools



State of DevOps

References
The Phoenix Project



References
The Goal – Theory of Constraints