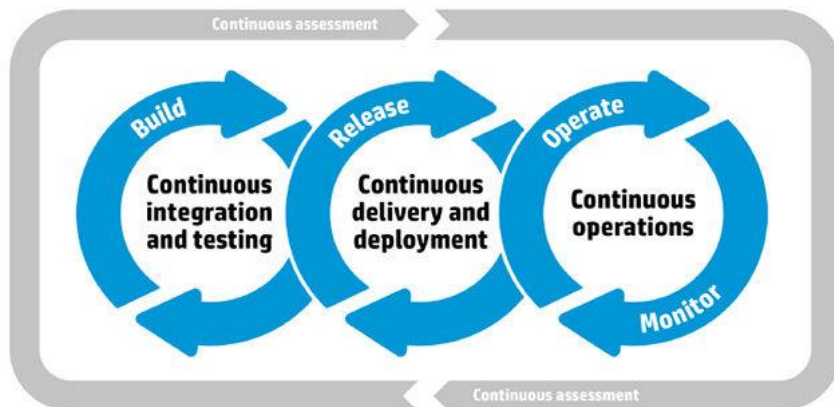




DevOps and Cloud Powered Software Development (DevOps)

Lecture 1 - Introduction

DevOps Foundation and Concepts:
Continuous Integration, Continuous Deployment,
Agile, Scrum, Sprint, others



Dr. Yuri Demchenko
University of Amsterdam



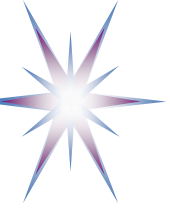
Outline

- DevOps History and trends
 - Value for organisations
- What is DevOps?
 - DevOps and Development lifecycle
 - DevOps and Agile
- Kaizen, Kata, Lean
- Agile, Scrum, Sprint
- Other DevOps component technologies
 - Continuous Delivery, Continuous Integration
 - DevOps maturity assessment
- DevOps tools and trends
- Summary and take away



DevOps History

- The origins of the DevOps movement took place around 2009:
 - “10+ Deploys Per Day: Dev and Ops Cooperation at Flickr” - presentation by John Allspaw and Paul Hammond from Flickr at O'Reilly Velocity Conference
- “Infrastructure as code” – Connected and empowered by Cloud Computing (since 2011)
 - Programmable networks SDN/NFV full infrastructure virtualisation
- The Lean Startup and Lean Thinking – Management approach/technology by Toyota (2012)
- Continuous Integration and Continuous Delivery
- Cloud and Platform as a Service (PaaS) technologies
- DevOps and containers



2015 Puppet “State of DevOps” Report

Executive Summary

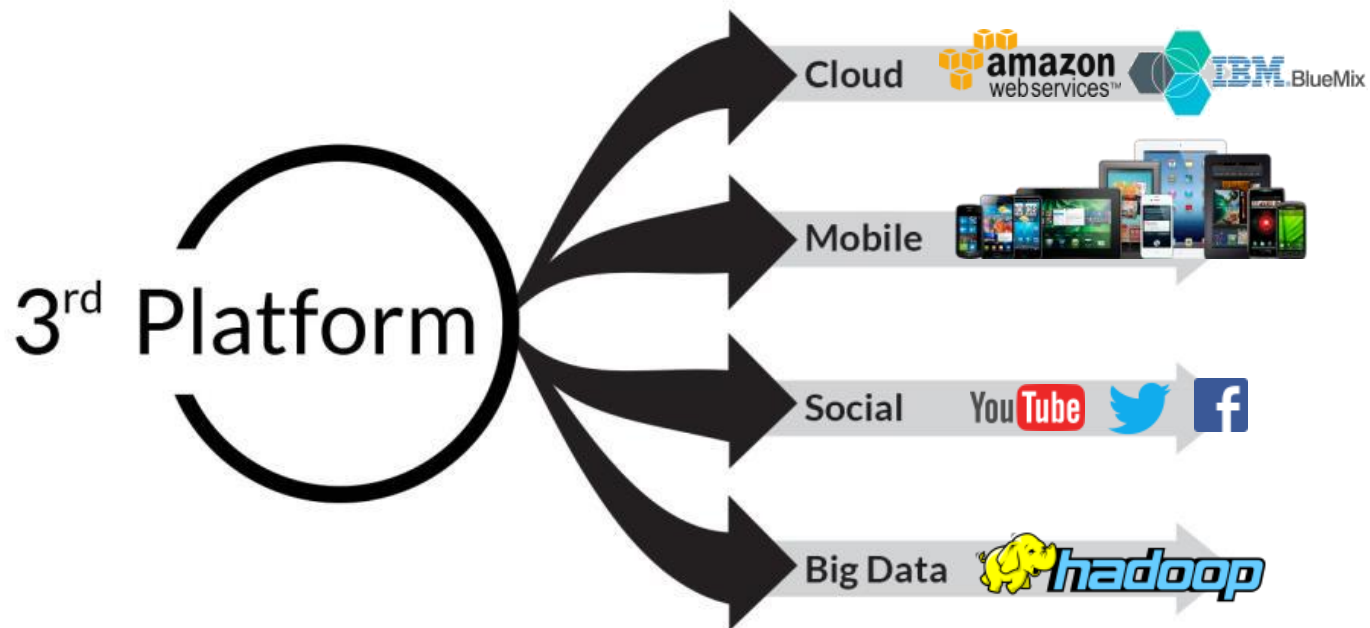
- **High-performing IT organizations: 1) deploy 30x more frequently 2) have 200x shorter lead times 3) have 60x fewer failures and recover 168x faster.**
- Lean management and continuous delivery practices create conditions to deliver value faster, sustainably.
- High performance is achievable whether your applications are greenfield, brownfield or legacy.
- IT managers play a critical role in any DevOps transformation.
- Diversity matters.
- Deployment pain is a key indicator of IT performance.
- Burnout can be prevented, and DevOps can help.
- <https://puppet.com/resources/white-paper/2015-state-devops-report>
- <https://puppet.com/resources/white-paper/2016-state-devops-report>

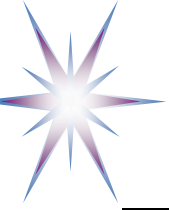


Look from 2016

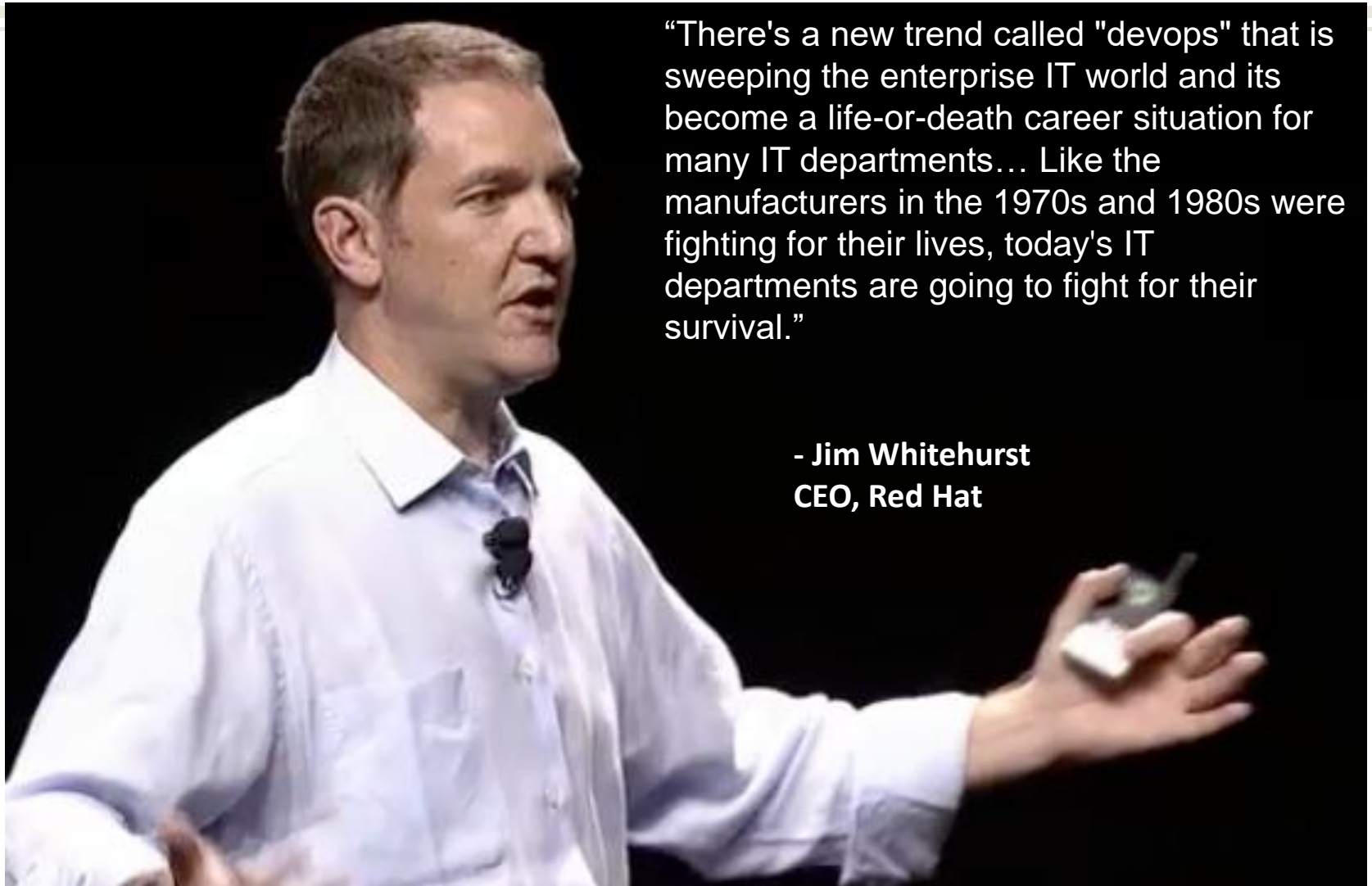


“By 2018, *one-third* of the *top 20* market share leaders in most industries will be *significantly disrupted* by competitors that use the *3rd Platform* to create new services and business models.”



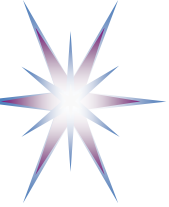


Voice from 2016

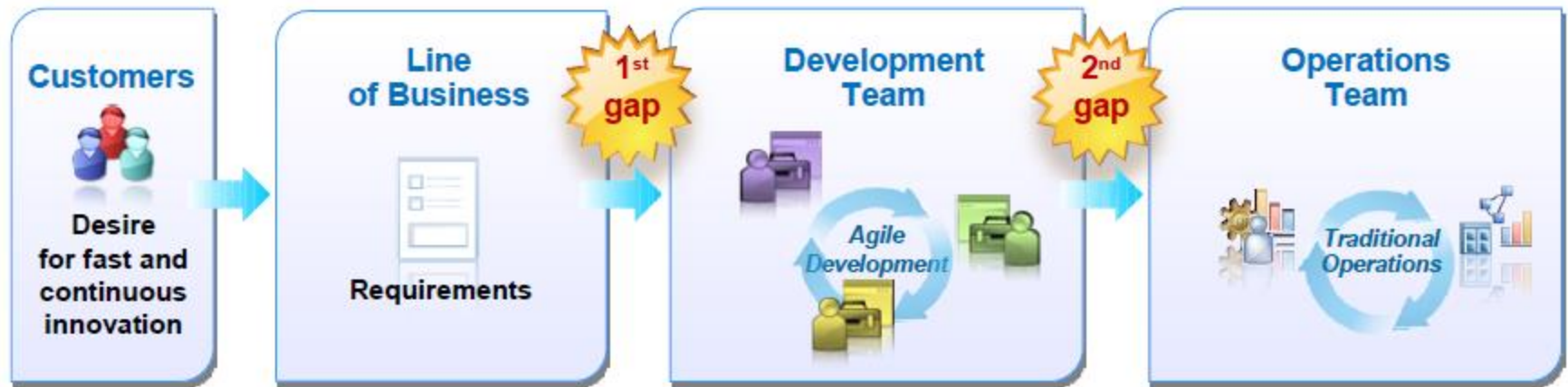


“There's a new trend called "devops" that is sweeping the enterprise IT world and its become a life-or-death career situation for many IT departments... Like the manufacturers in the 1970s and 1980s were fighting for their lives, today's IT departments are going to fight for their survival.”

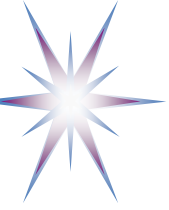
**- Jim Whitehurst
CEO, Red Hat**



Delivery Challenges

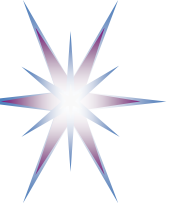


- Today's business and technical needs are pushing traditional delivery approaches to the breaking point
- Technical Challenges: **Scale - Complexity - Time – Market Pressures**
- Technical Trends (since 2015): SoLoMo (Social – Local - Mobile)
 - Increased demand that software and apps work across SoLoMo environment: Location aware linked to Social Network
 - Data collection and processing across SoLoMo



DevOps to Reduce Delivery Gaps

- Design and Deployment Planning
 - Integrate and automate deployment planning processes across development and operations
 - Ensure asset and configuration details are shared and synchronized across asset stores
- Environment Setup, Testing, Deployment and Monitoring
 - Leverage integrated tools for discovery and accelerating provisioning of test lab and production environments.
 - Improving test performance by replicating “real world” environments - faster testing & problem resolution
- Issue Identification and Resolution Management
 - Resolving problems quicker by sharing problem and ticket information
 - Ensuring tracking tools for production problems and application fixes remain synchronized
- Easy with Cloud based Infrastructure as Code



DevOps Typical Stories

Software Delivery Lifecycle (Integrated Development and Operations Lifecycles)

- Story 1: Dev and Ops collaborate to develop environment definitions
 - Value: Ensures that Dev understands and deals with production-like environments; avoids architectural miscommunications
- Story 2: 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 3: Release Applications from Test/Staging to production
 - Value: Shared technology and automation ensures no gratuitous differences between dev/test and prod.
- Story 4: Collaborative incident management
 - Value: ensures an integrated process for reproducing and resolving defects and issues between dev, test, and ops.
- Story 5: 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 6: Manage the entire delivery pipeline with end-to-end visibility and dashboards
 - Value: Enables end-to-end delivery metrics and visibility into bottlenecks.



What is DevOps

Collaborative Development

Foster productive collaboration with deeper lifecycle integrations
“No hassle” collaborative development capabilities on the cloud for continuous delivery



Continuous Testing

Enhanced integrations and capabilities to synchronize software testing with deployment and operations



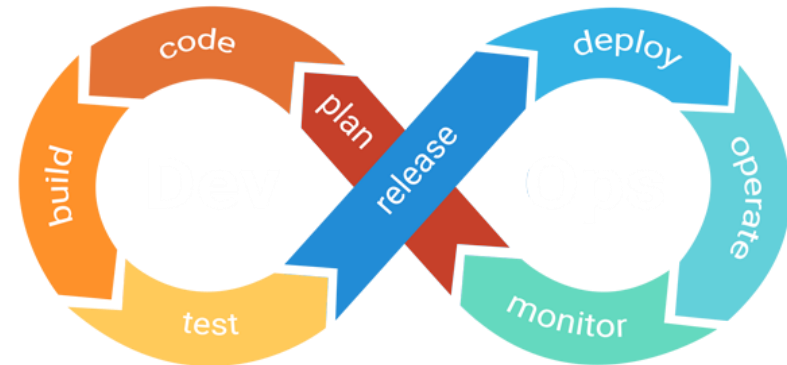
Continuous Delivery: Release and Deployment

Greater delivery speed and frequency for complex applications



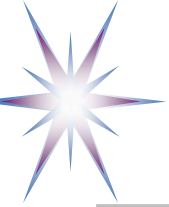
Continuous Monitoring

Capabilities to improve service quality by monitoring application performance

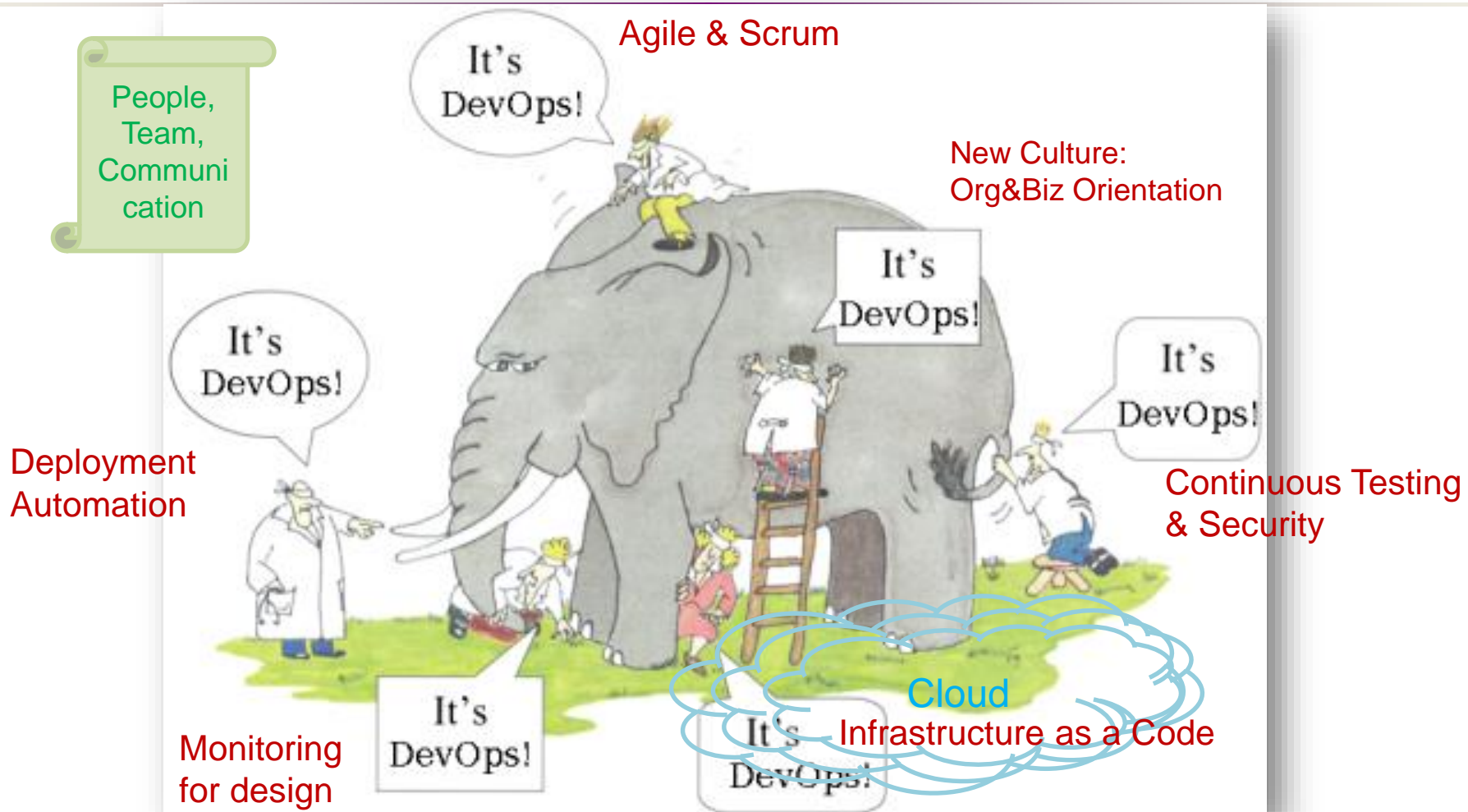


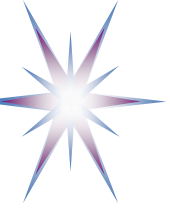
Agile development approach

- Spans the entire lifecycle, includes business planning and creation to delivery and feedback.
- Enable continuous delivery of software-driven processes and innovation



What is DevOps Really?





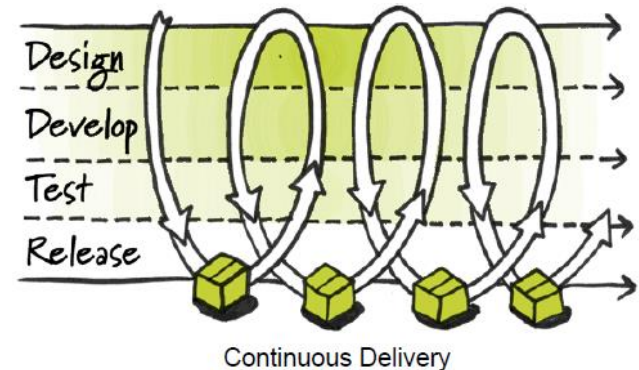
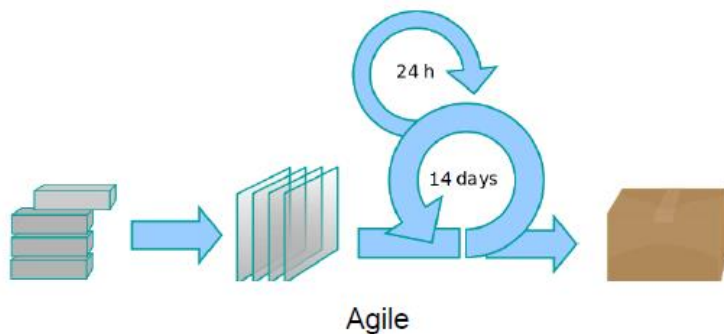
DevOps adoption: Driving and enabling factors

- 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**
- **Programmable network (SDN/NFV)** that made possible infrastructure virtualisation



DevOps vs/and Agile

- DevOps is especially complementary to the Agile software development process.
 - extends and completes the continuous integration and release process by ensuring that code is production ready and will provide value to the customer
- DevOps enables far more continuous flow of work into IT Operations.
 - If development delivers code every two weeks but it's deployed only every two months, customers don't get value and the deployments often result in chaos and disruption.





DevOps 3 Basic Principles

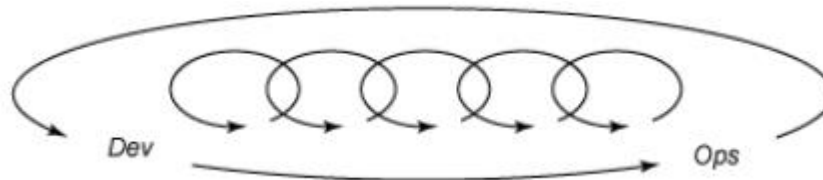
- System thinking

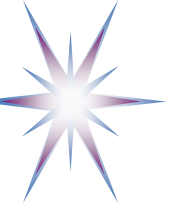


- Amplify feedback loops



- Culture of continuous experimenting and learning





DevOps 3 Basic Principles – Compare with the Research Methods

- System thinking

Compare with the research process

- Design Experiment
- Collect Data
- Analyse Data
- Identify Patterns
- Hypothesis Explanation
- Test Hypothesis

Dev

Think so-called 21st Century Skills

- Design/System Thinking
- Critical thinking and problem solving
- Innovation and Creativity
- Cognitive flexibility
- Cross-disciplinary ability

Dev

Collect Data

Analyse Data

Identify Patterns

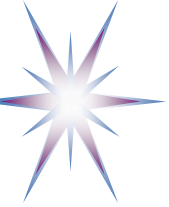
Hypothesise Explanation

Test Hypothesis

Design Experiment

Experiment and learning

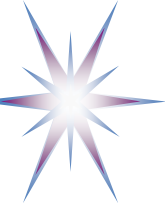
Ops



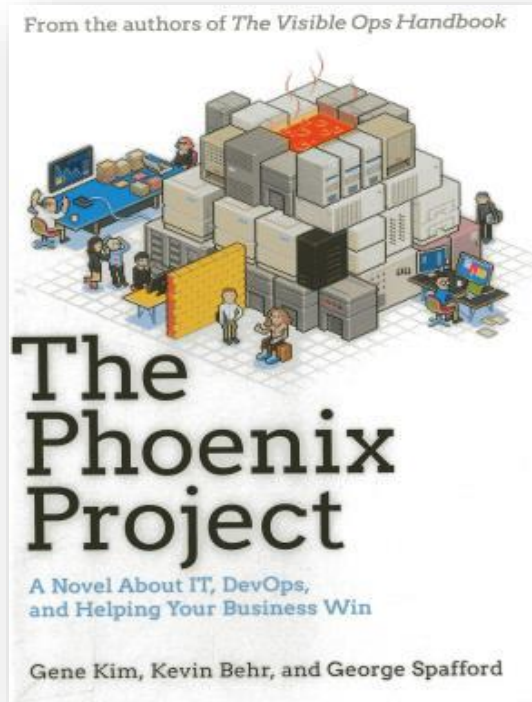
Applying DevOps Principles to Organisation

Common goals of an enterprise DevOps practice

- Increased deployment frequency
- Reduced lead time for changes
- Faster recovery when problems occur
- More robust and better integrated security
- A “shift left” (to origin) in quality – quality of code, testing, architecture, “deployability” and culture
- Fast feedback loops and effective communication between teams and departments



2013: *The Phoenix Project*: A Novel About IT, DevOps, and Helping Your Business Win



Gene Kim



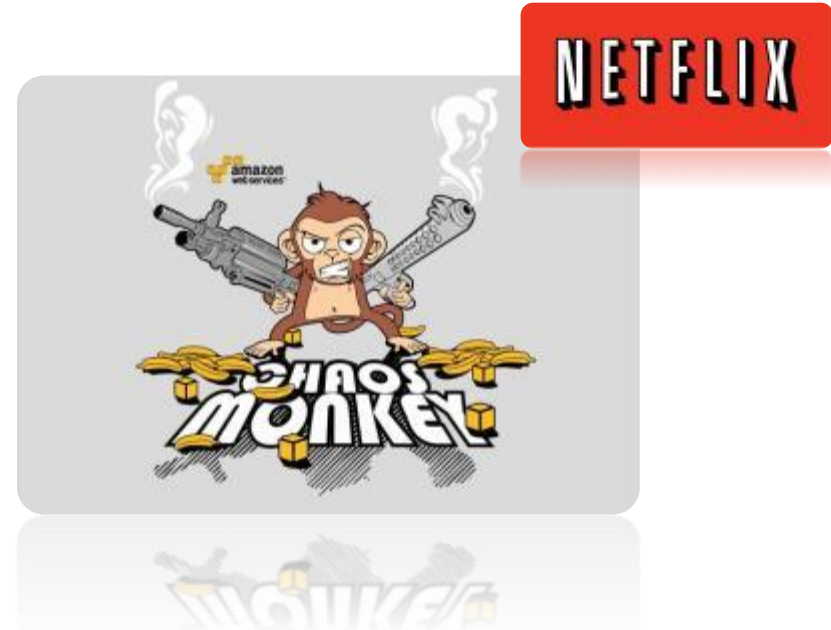
**George
Spafford**



Kevin Behr

- The Phoenix Project is an allegorical tale of DevOps—the needs, the transition, and the business impact—as told through the main character, Bill Palmer, who has recently been named VP of IT Operations, much against his will.
- DevOps is the application of Lean manufacturing methods, architecture, and ideology to the world of software development and operations.

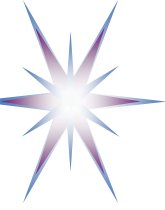
One of Important DevOps Tool - Failure



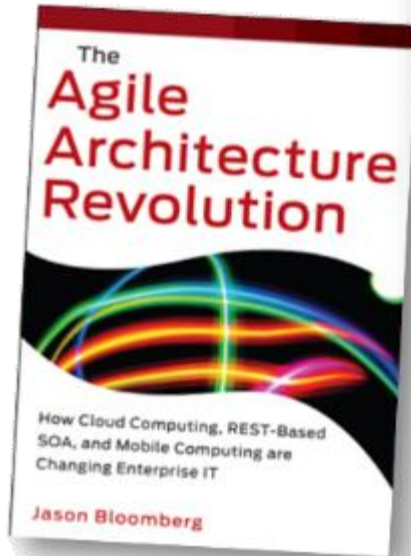
Chaos Monkey is a failure generation tools at Netflix

<https://netflix.github.io/chaosmonkey/>

- Failure is not a cause for blame, it is a vehicle for change, learning, and improvement.

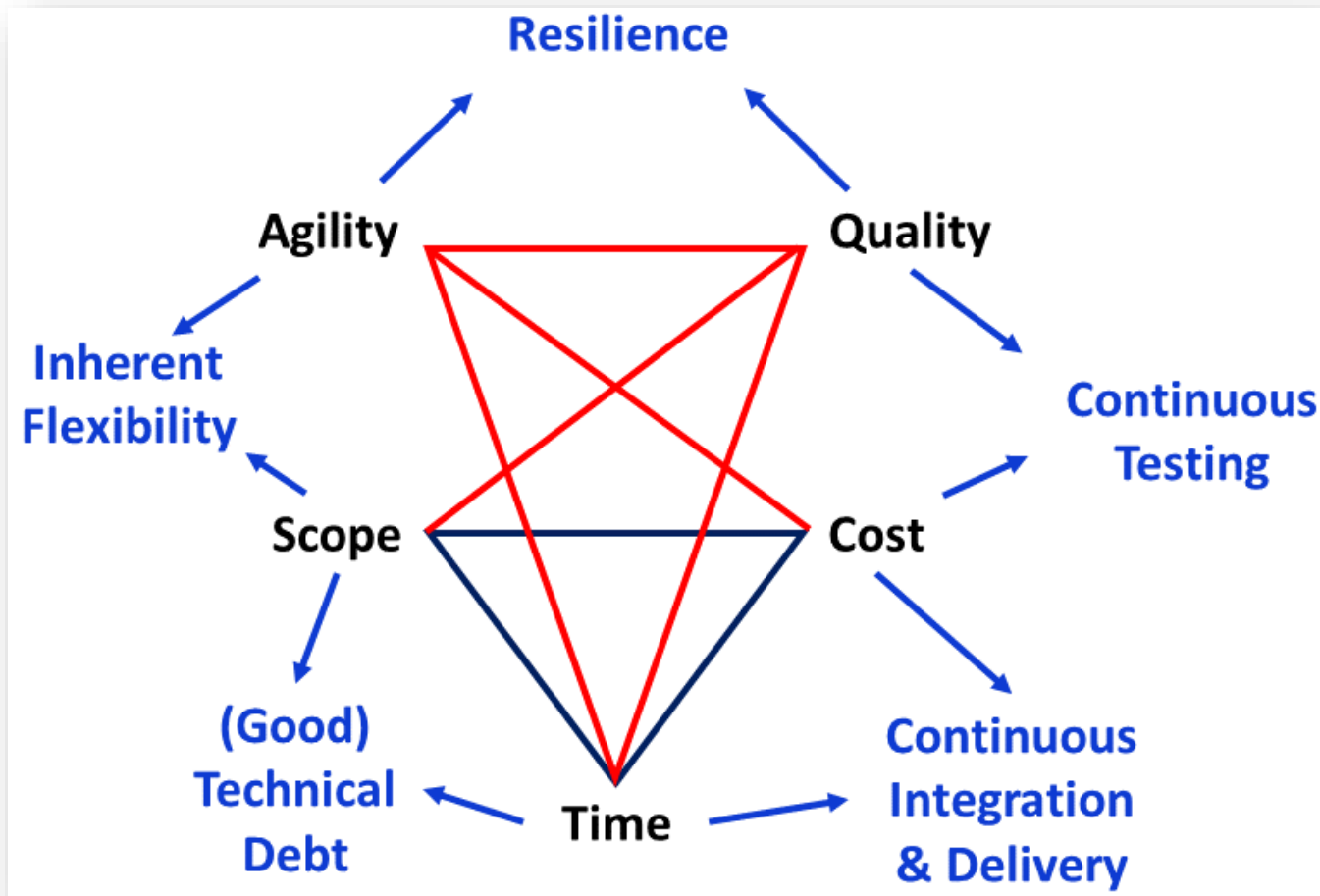


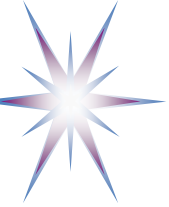
Person and book to know: Jason Bloomberg, Intellyx (<https://intellyx.com/author/jbloomborg/>)



The Agile Architecture Revolution: How Cloud Computing, REST-Based SOA, and Mobile Computing Are Changing Enterprise IT (2013)

Jason Bloomberg's Agile Architecture "Quality Star"





How DevOps influences software engineering & architect roles

- Manage projects effectively through open, standards-based platforms
- Address requirements for the organizations, vendors and teams (not just for components)
- Increase project visibility through traceability
- Common reporting and analytics across the lifecycle
- Improve quality and reduce development costs with collaboration
- Establish assets reuse across organizations, vendors and teams



DevOps and Cloud



Reduce cost and speed up delivery with an integrated Cloud solution

- Integrated IaaS, PaaS, Application Lifecycle Management tooling, Service Management and Monitoring provide an instant platform for DevOps
- Control operating costs with Token licensing



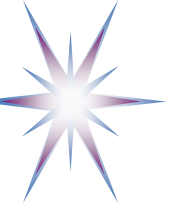
Streamline software delivery process and reduce time to value

- Rapid provisioning of virtual private/hybrid clouds environments with Patterns
- Continuous deployment and release across environments and SDLC stages
- High-quality achieved with early and continuous testing
- Collaboration across the enterprise (SoE, SoR teams)



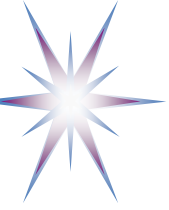
An integrated set of DevOps services in the cloud

- Shrink development/test/deploy/learn cycle time, but consistently deliver software with speed, quality, accuracy with progressive rollouts



DevOps: Closer look at components technologies

- DevOps definition and relation to Agile
- Organisation processes
- CAMS
- Kaizen
- Kata
- Lean
- Agile, Scrum, Sprint, User Stories



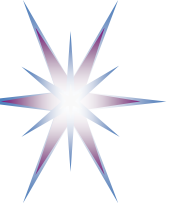
DevOps – Formal Definition

DevOps - is the practice of operations and development engineers participating together through the entire service lifecycle; from the design and development process all the way to production support. DevOps is also characterized by operations staff making use of many of the same techniques as developers for their systems work.

- Extended “live” definition <https://theagileadmin.com/what-is-devops/>
- **DevOps has strong affinities with Agile and Lean approaches.**
- DevOps can be interpreted as an outgrowth of Agile – agile software development
 - prescribes close collaboration of customers, product management, developers, and (sometimes) QA to fill in the gaps and rapidly iterate towards a better product
- ***DevOps is simply extending Agile principles beyond the boundaries of “the code” to the entire delivered service***

DevOps Essentials

- Better Software, Faster time to market
- Infrastructure as Code and cloud based virtualisation
- Synergy of Development and Operations
- Covers the *entire* Application Life Cycle



Multiple aspects of DevOps

- DevOps means a lot of different things to different people because the discussion around it covers a lot of ground. People talk about DevOps being
 - “developer and operations collaboration”
 - “treating your code as infrastructure”
 - “using automation”
 - “using kanban” (continuous improvement)
 - “a toolchain approach”
 - “culture” or a variety of seemingly loosely related items.
- The best way to define is to link DevOps to similarly complex term, agile development.



Relationship to Agile and Continuous Delivery

Agile and DevOps are similar, but, while agile software development represents a change in thinking and practice (that should lead to organizational change), DevOps places more emphasis on implementing organizational change to achieve its goals

- The need for DevOps was born from the increasing popularity of agile software development, as that tends to lead to an increased number of releases.
- One goal of DevOps is to establish an environment where releasing more reliable applications, faster and more frequently, can occur.

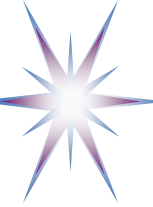
Continuous delivery and **DevOps** are similar in their meanings but they are two different concepts.

DevOps has a broader scope, and centers around:

- Organizational change: specifically, to support greater collaboration between the various types of workers involved in software delivery:
- Automating the processes in software delivery

Continuous delivery, on the other hand, is an approach to automate the delivery aspect, and focuses on:

- Bringing together different processes and executing them more quickly and more frequently.



DevOps and all Organisational processes

- DevOps is not a plan, it's a reaction. It was originated from practical experience.
 - A trend emerged from Agile based web operating (WebOps) companies.
 - These new companies were building operational organizations that were running their operations very similar to the way they ran their agile development process.
 - Cloud and virtualisation facilitated change in thinking and culture
- DevOps and change management
 - Different part can be provisioned at different speed: that require approval and not



DevOps is about CAMS

Culture

- People and process first. If you don't have culture, all automation attempts will be fruitless.

Automation

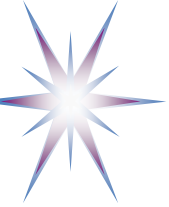
- This is one of the places you start once you understand your culture. At this point, the tools can start to stitch together an automation fabric for Devops. Tools for release management, provisioning, configuration management, systems integration, monitoring and control, and orchestration become important pieces in building a Devops fabric.

Measurement

- Measuring helps improving. A successful DevOps implementation will measure everything it can as often as it can: performance metrics, process metrics, and even people metrics.

Sharing

- Sharing is the loopback in the CAMS cycle. Creating a culture where people share ideas and problems is critical.
- Focus communication between Dev and Ops on the problem, not each other.
- Another interesting motivation in the DevOps movement is the way sharing DevOps success stories helps others.
 - First, it attracts talent, and second, there is a belief that by exposing ideas you can create a great open feedback that in the end helps them improve.
- **CAMS + Lean => CALMS**

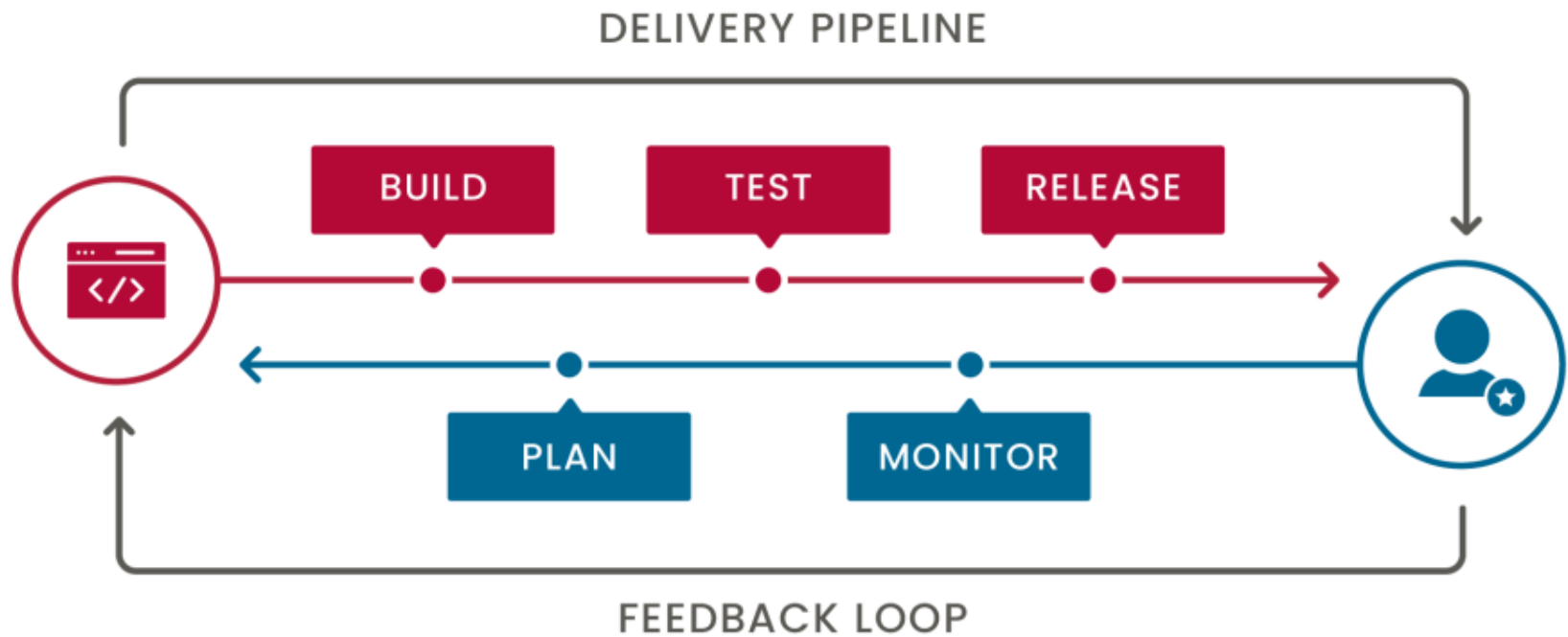


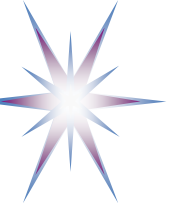
Waterfall model



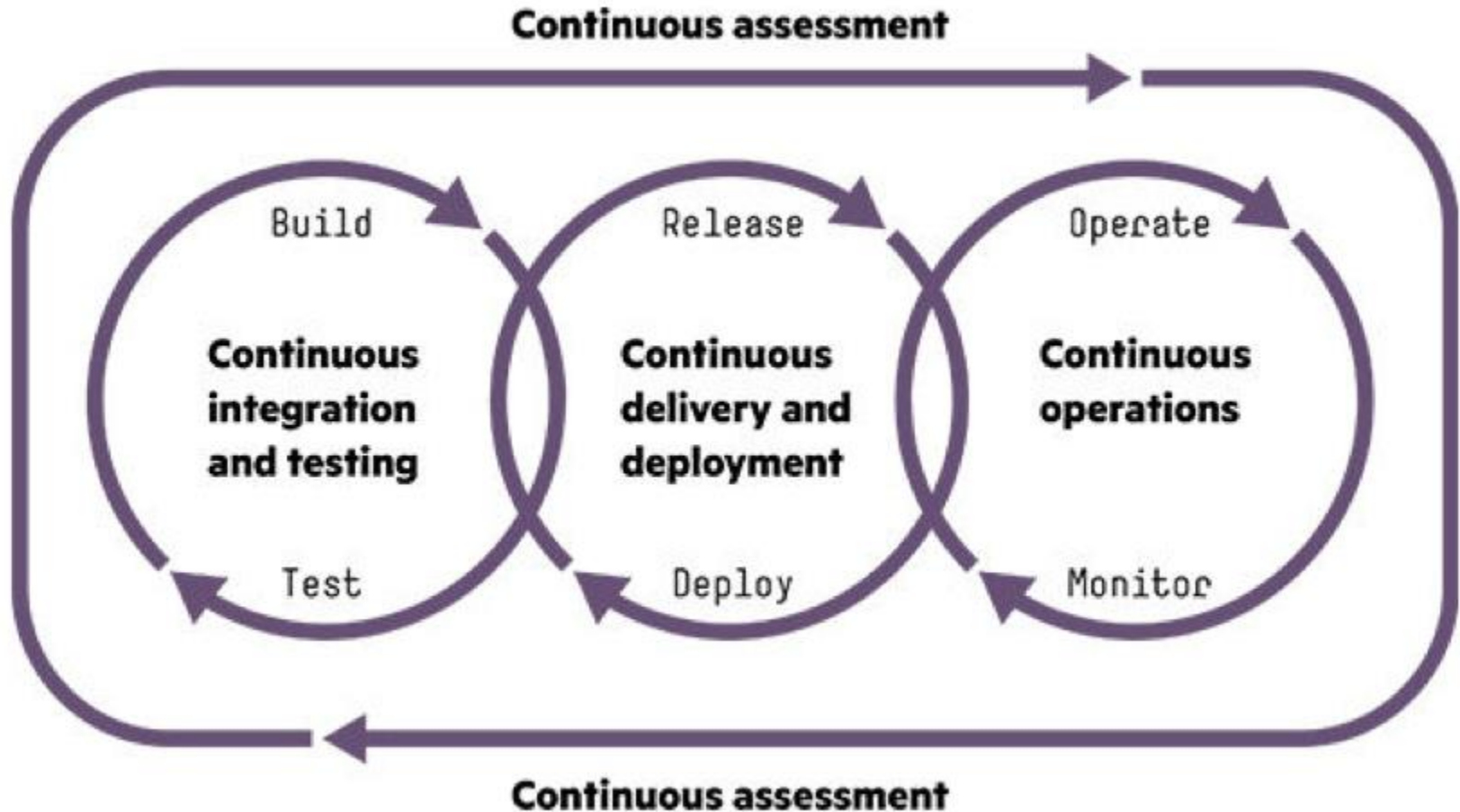


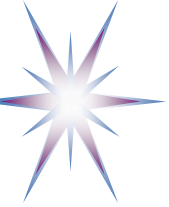
Delivery Pipeline



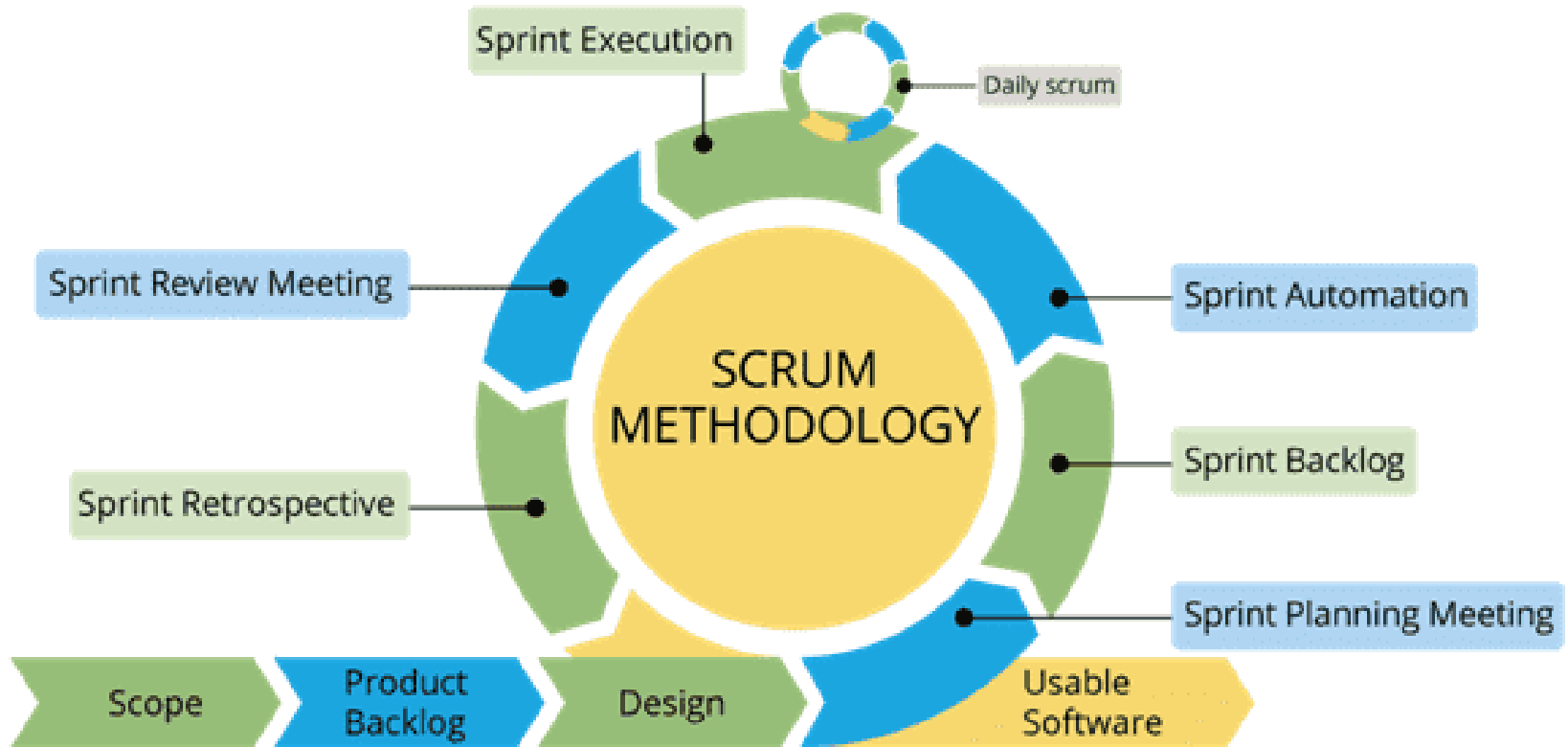


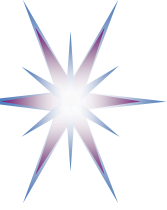
3 Cycles in the DevOps process



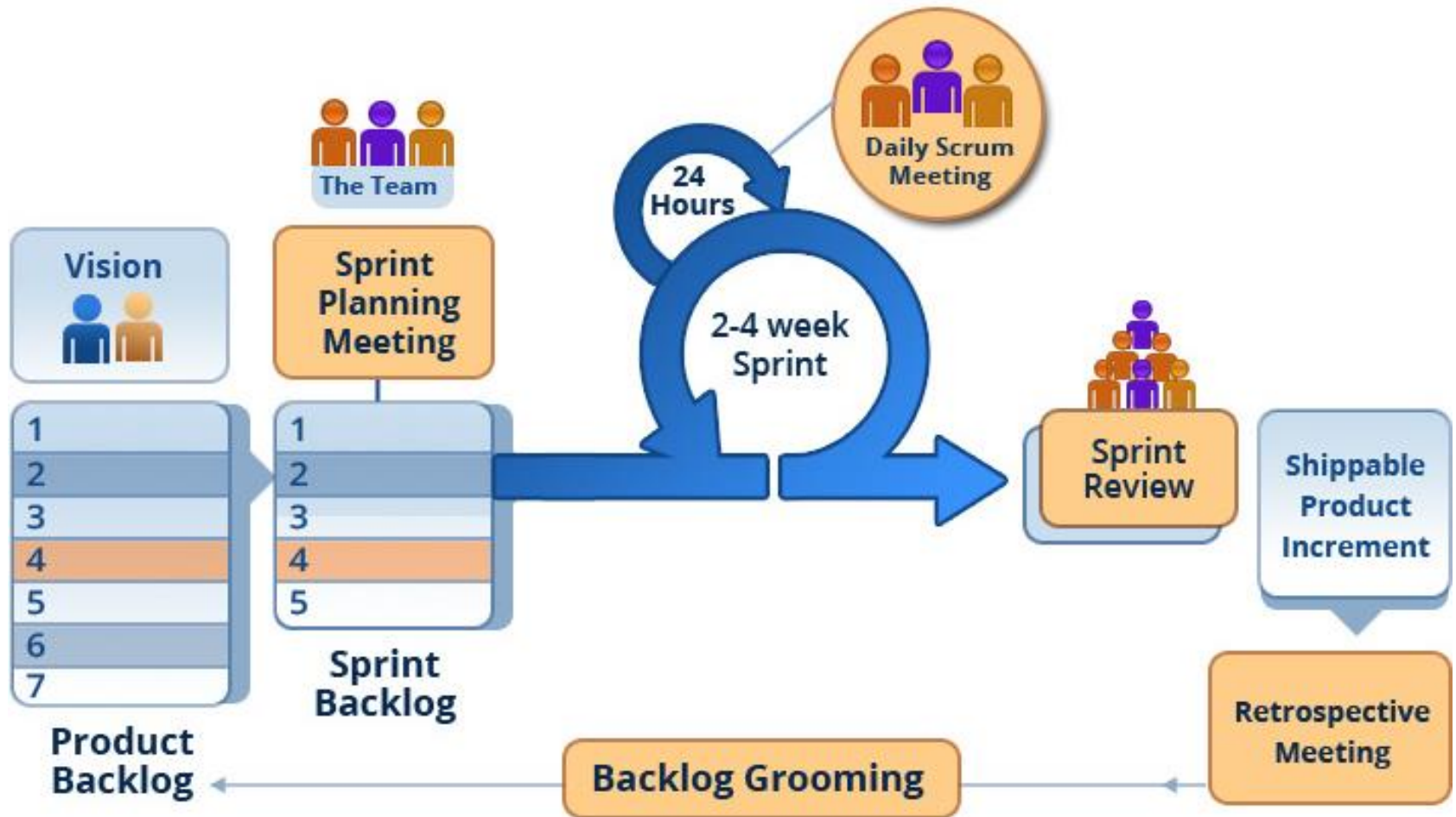


Scrum methods – Primary focused on the Design stage

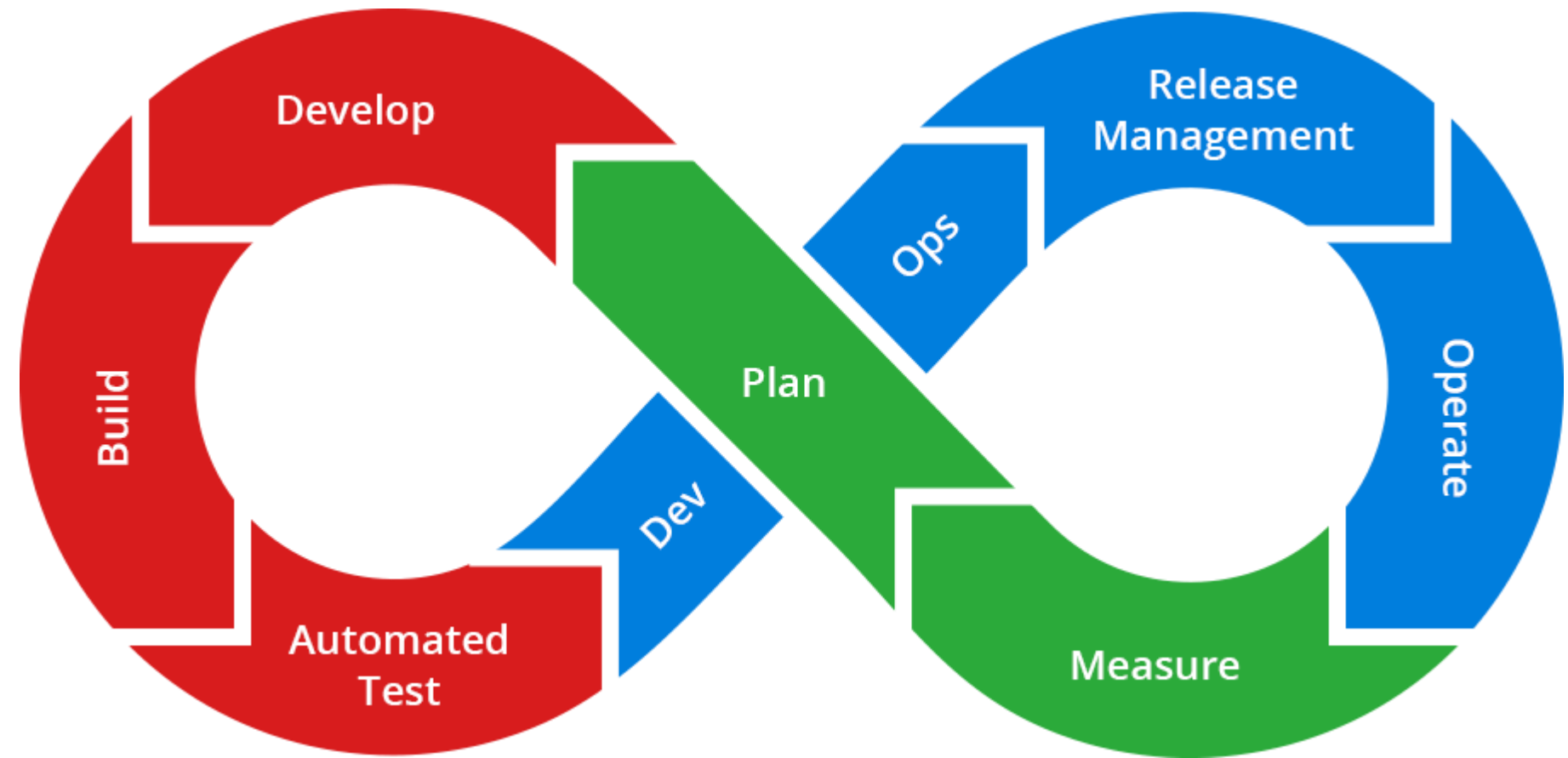




Scrum methods: Typical practices



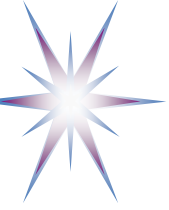
Continuous Evolution in Product Development





5 Key DevOps Methodologies

- People over Process over Tools
- Continuous Delivery
- Lean Management
- Visible Ops style Change Control
- Infrastructure as Code

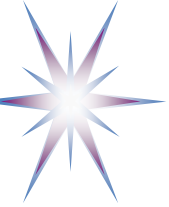


Kaizen: Continuous Improvement

Kaizen - change for the better

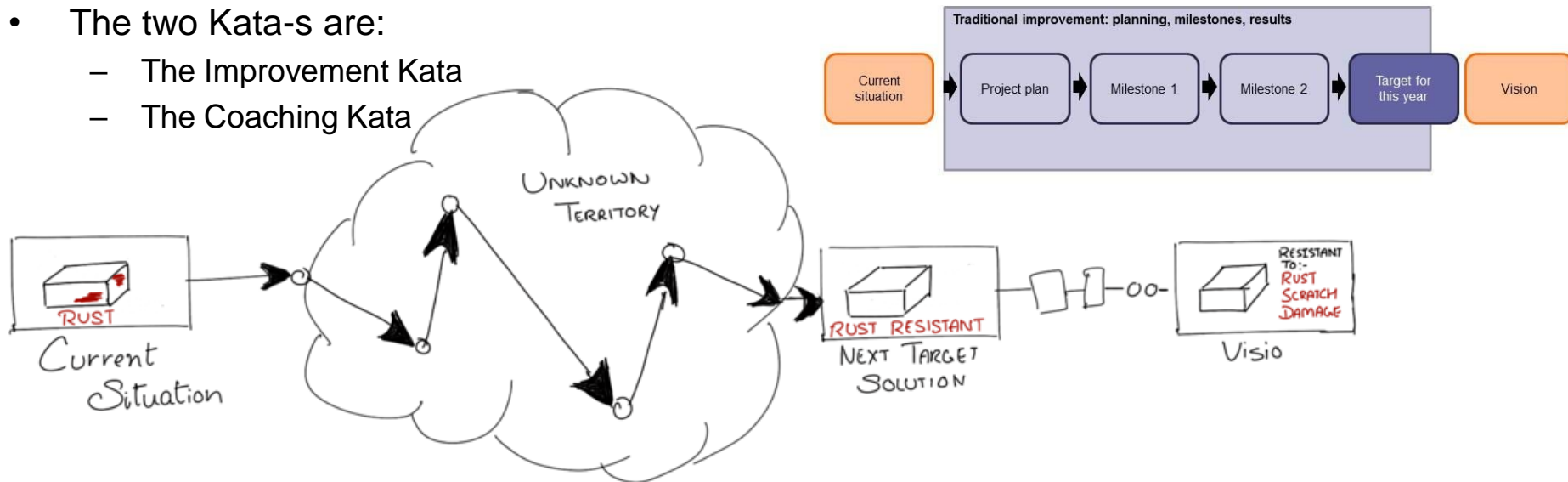
Kaizen's Guiding Principles

- Good processes bring good results
- Go see for yourself to grasp the current situation (gemba)
- Speak with data, manage by facts
- Take action to contain and correct root causes of problems
- Work as a team
- Kaizen is everybody's business



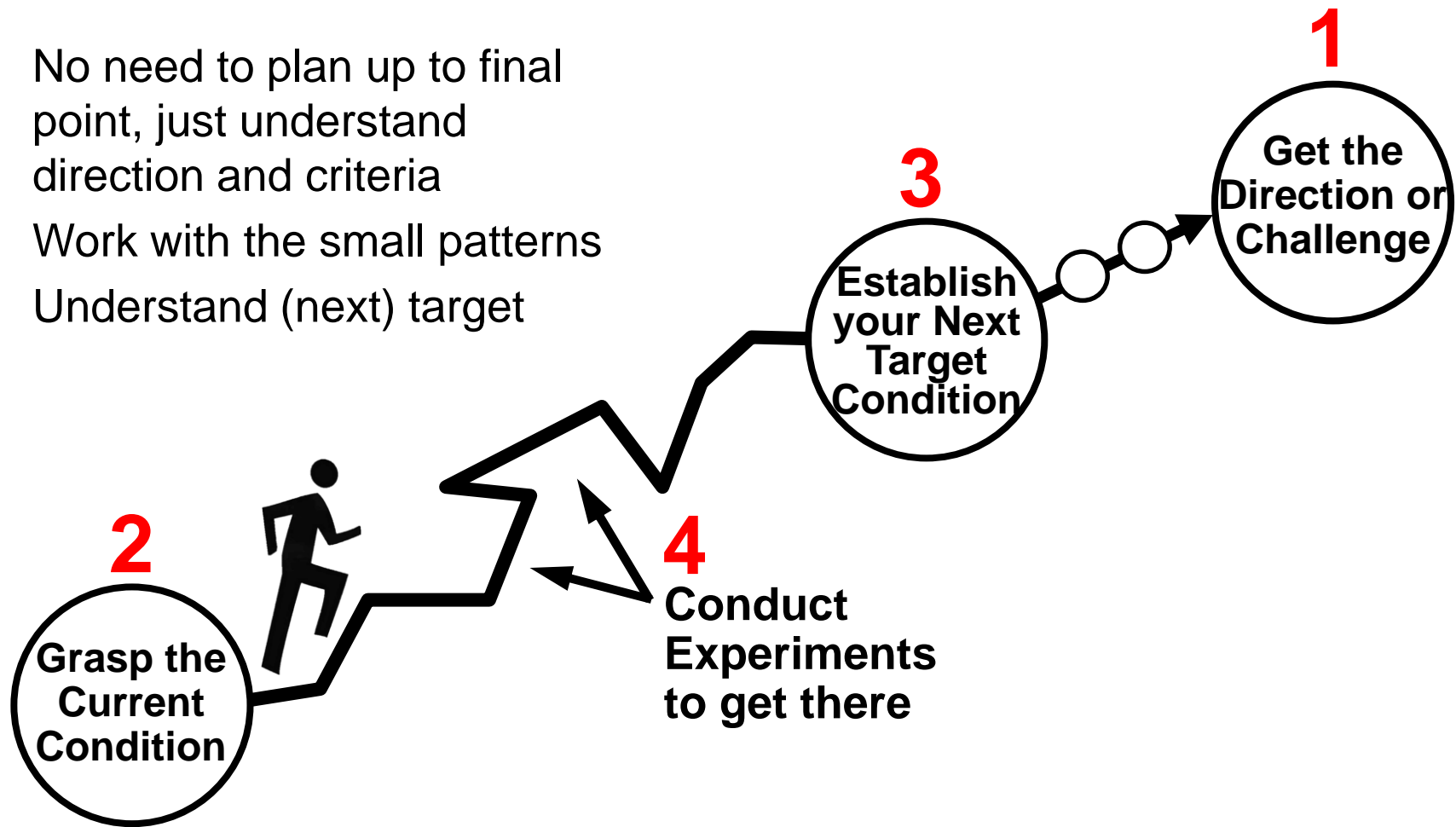
Toyota Kata

- Kata means pattern, routine, habits or way of doing things. Kata is about creating a fast “muscle memory” of how to take action instantaneously in a situation without having to go through a slower logical procedure.
- Follow kata to experiment with the quality, instead of first, taking a sprint or two to figure out the best possible solution to use for achieving the best quality possible, before delivering anything useful.
- Similar to the Agile way of working, Kata improves in small steps and doesn’t plan the whole path to the desired improvement. The desired end state or ‘definition of done’ is known. But only the first achievable target condition is determined in advance. No further milestones.
- The two Kata-s are:
 - The Improvement Kata
 - The Coaching Kata



4 Steps of Improvement Kata Approach

- No need to plan up to final point, just understand direction and criteria
- Work with the small patterns
- Understand (next) target



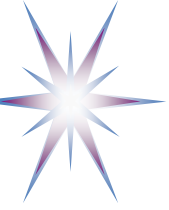
Toyota Kata, by Mike Rother

<http://www-personal.umich.edu/~mrother/Homepage.html>



Improvement Kata – 4 steps summary

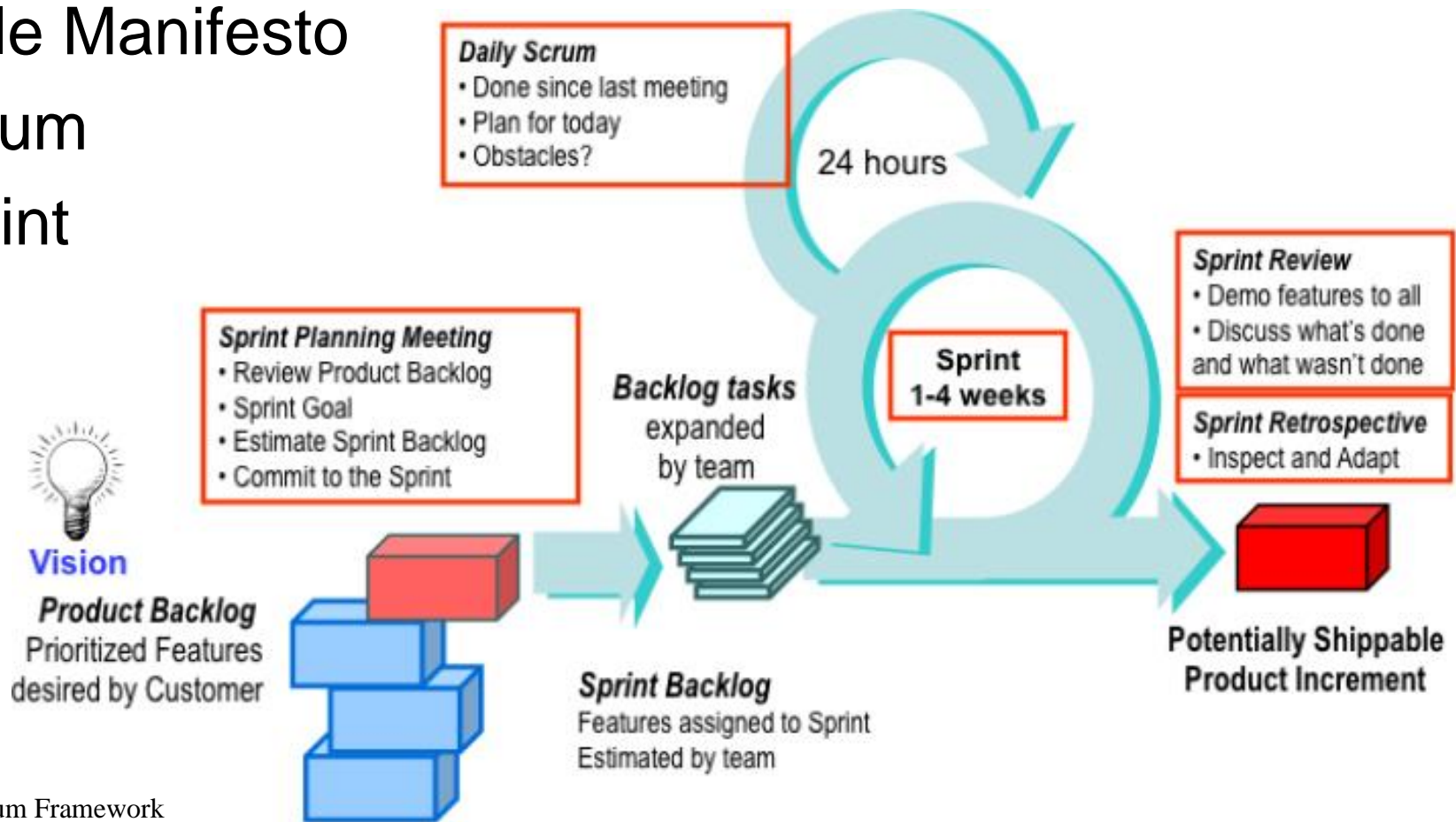
- The Improvement Kata contains four steps
 - Understand the Direction, so you know where you are going
 - Grasp the Current Condition, to get your reference point
 - Establish the Next Target Condition, that is your next step on the path towards the vision
 - Plan-Do-Check-Act Toward the Target Condition
- The team consists of developers, testers, operations and business representatives
 - **Collocation is important** (as well as commitment)



DevOps Building Blocks: Agile

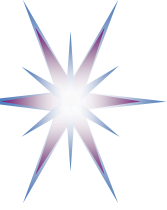
Agile is umbrella for multiple methods and frameworks

- Agile Manifesto
- Scrum
- Sprint



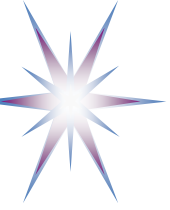
The Original Scrum Framework

<https://www.pmi.org/learning/library/agile-project-management-scrum-6269>



Agile: Primarily focused on development/code

- **Agile Values** – Top level philosophy, usually agreed to be embodied in the [Agile Manifesto](#). These are the core values that inform agile.
- **Agile Principles** – Generally agreed upon strategic approaches that support Agile values.
 - The Agile Manifesto cites [a dozen of specific principles](#): Use as guidance, not law.
- **Agile Methods** – More specific process implementations of the principles (not mandatory, just possible implementation).
 - XP, Scrum, organisation's homebrew process: “how we intend to do this in real life”
- **Agile Practices** – Highly specific tactical techniques that tend to be used in conjunction with agile implementations.
 - Standups, planning poker, backlogs, CI, - all the specific artifacts a developer uses to perform their work.
 - None are required to be agile but many agile implementations have seen value from adopting them.
- **Agile Tools** – Specific technical implementations of these practices used by teams to facilitate doing their work according to these methods.
 - JIRA Agile, planningpoker.com, et al.



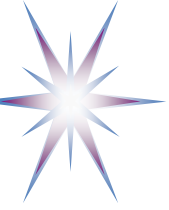
12 Principles of Agile - <http://agilemanifesto.org/>

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.



Scrum

- Scrum is a process framework that has been used to manage work on complex products since the early 1990s.
 - Scrum is not a process, technique, or definitive method. Rather, it is a framework within which you can employ various processes and techniques.
- Scrum relies to team of 7+-2 members
 - Scrum master
 - Product owner
 - Team members, preferable T-shaped, part-time is possible
- Delivery in limits of budget and time
 - Delivery of 2 to 4 weeks
- Daily standup meetings
- Retrospective (scrum ceremony to assess achievements and improve)



Scrum Team: Product Owner (PO)

- Acts as the full-time business representative
- Reviews the team work
- Ensures highest value is delivered
- + Interact with stakeholders
- Formerly was represented by requirements that never have changed
 - Acts as “living” voice of customer
- Target can be changed along progress



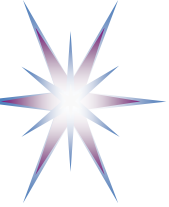
Scrum Team: Scrum Master

- Scrum master is most visible spoke person for team
- Remove blocks for team
- Focus on how the team does work
- Holds the team accountable to the product owner



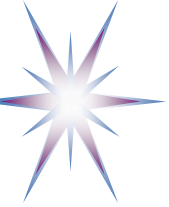
Scrum Team Norms

- Working relationships
- Conflict resolution
- Consensus options
 - Agree to disagree but go forward
- Standup meeting

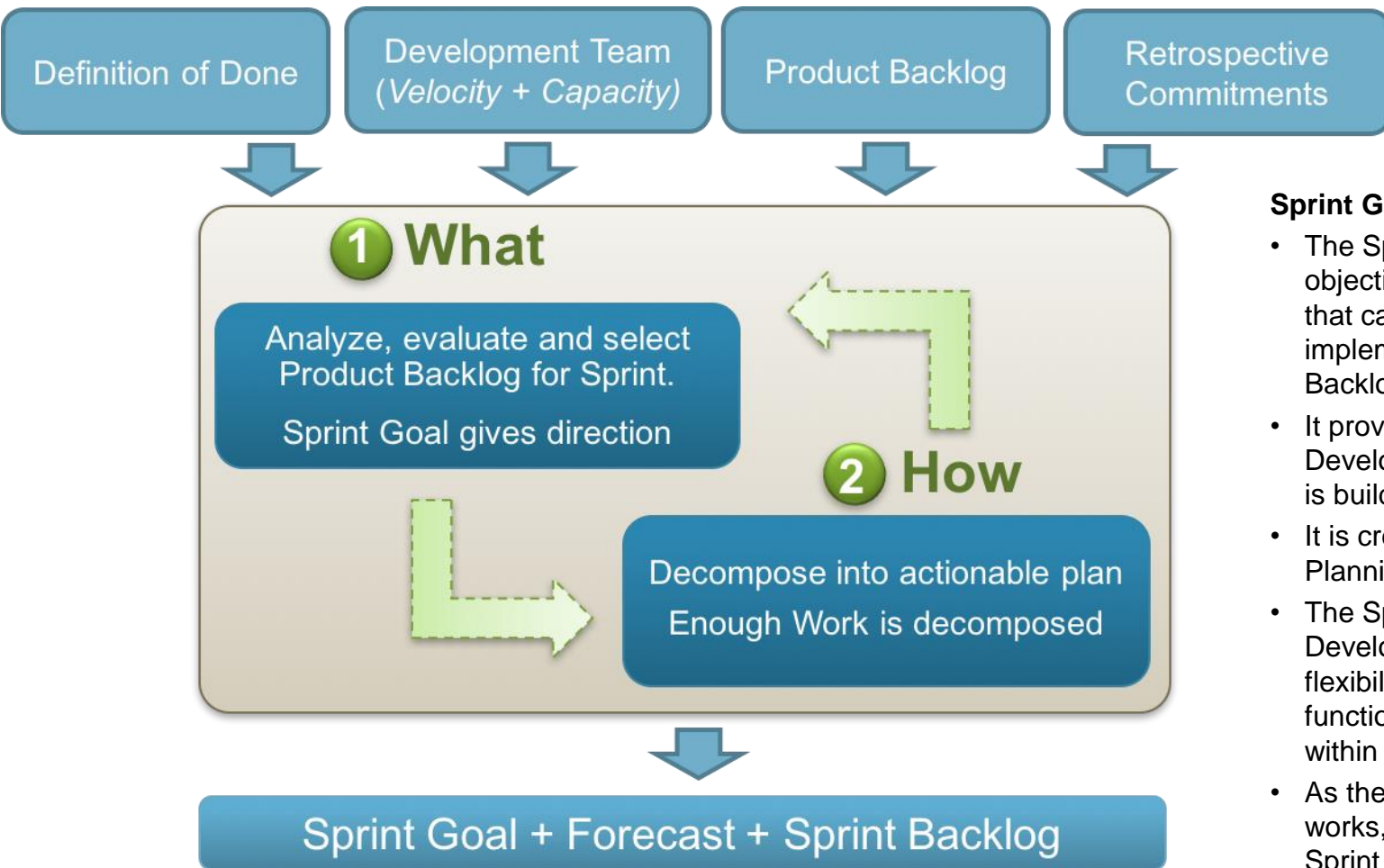


Sprint planning

- **Sprint planning** is a collaborative effort involving a Scrum Master, who facilitates the meeting, a Product Owner, who clarifies the details of the product backlog items and their respective acceptance criteria, and the Entire **Agile** Team, who define the work and effort necessary to meet their **sprint** commitment.
- Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter.
- The [Scrum Master](#) ensures that the event takes place and that attendants understand its purpose. The Scrum Master teaches the Scrum Team to keep it within the time-box.
- Sprint Planning answers the following:
 - What can be delivered in the Increment resulting from the upcoming [Sprint](#)?
 - How will the work needed to deliver the Increment be achieved?
- Work is selected from the [Product Backlog](#) and pulled into the [Sprint Backlog](#).
 - Remember that the work in the Sprint Backlog is not a commitment, it is a forecast. The only container of a Sprint is its **time box**, not the work planned for the Sprint.

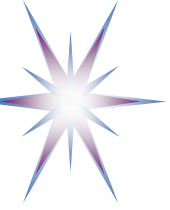


Sprint Planning



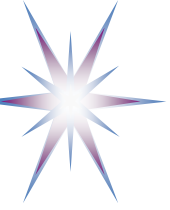
Sprint Goal

- The Sprint Goal is an objective set for the Sprint that can be met through the implementation of Product Backlog.
- It provides guidance to the Development Team on why it is building the Increment.
- It is created during the Sprint Planning meeting.
- The Sprint Goal gives the Development Team some flexibility regarding the functionality implemented within the Sprint.
- As the Development Team works, it does so with the Sprint Goal always in mind.



Scrum process – Prepare Sprint

- Product Vision
 - Serves as a guide for the team
 - Focused on final value for users
 - Established by PO
- Minimum Viable Product (MVP)
 - The product developed enough to get user feedback
 - Allows for a fast feedback
 - Reduces scope creep
- Break vision onto features
 - E.g., profile, login, security, user interface browser/mobile



Scrum: User stories

- Detailed piece of work the team can deliver
- Comply with INVEST
 - Independent
 - Negotiable
 - Valuable, meaningful for all
 - Estimable
 - Small, enough to complete in one sprint
 - Testable
- Diverting tasks to be excluded: e.g., upgrade database – find way around



DevOps Building Blocks: Lean

7 Principles of Lean Software Development

- **Eliminate waste**
- Amplify learning
- Decide as later as possible
- Deliver as fast as possible
- Empower the team
- Build integrity in
- See the whole

The Seven Wastes (Muda) of Lean Software

- Waste #1 - Partially Done Work
- Waste #2 - Extra Features
- Waste #3 – Relearning
- Waste #4 – Handoffs
- Waste #5 – Delays
- Waste #6 - Task Switching
- Waste #7 - Defects

Build-Measure-Learn

- Build – Minimum Viable Product
- Measure – The outcome and internal metrics
- Learn – About your problem and your solution
- Repeat – Go deeper where it is needed



Kanban vs Scrum

SCRUM

KANBAN

A Scrum Master, Product Owner and Team Members make up a Scrum Team.

Team Roles

No set roles are defined. Roles are not required to be cross functional.

Columns are labeled to reflect periods in the work flow from beginning through the team's definition of done.

Work Boards

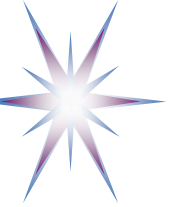
Columns are likewise labeled to show work flow states, but also publish the max number of stories allowed in column at once.

Scrum processes place heavy emphasis on schedule with a prioritized list of story points. This iterative process enables accurate estimations of work flow and effective management of multiple projects.

Scheduling/ Cadence

There are no required time boxes or iterations. While the Kanban method is iterative in nature, the continual improvement is expected to occur in an evolutionary fashion as work is continually completed.

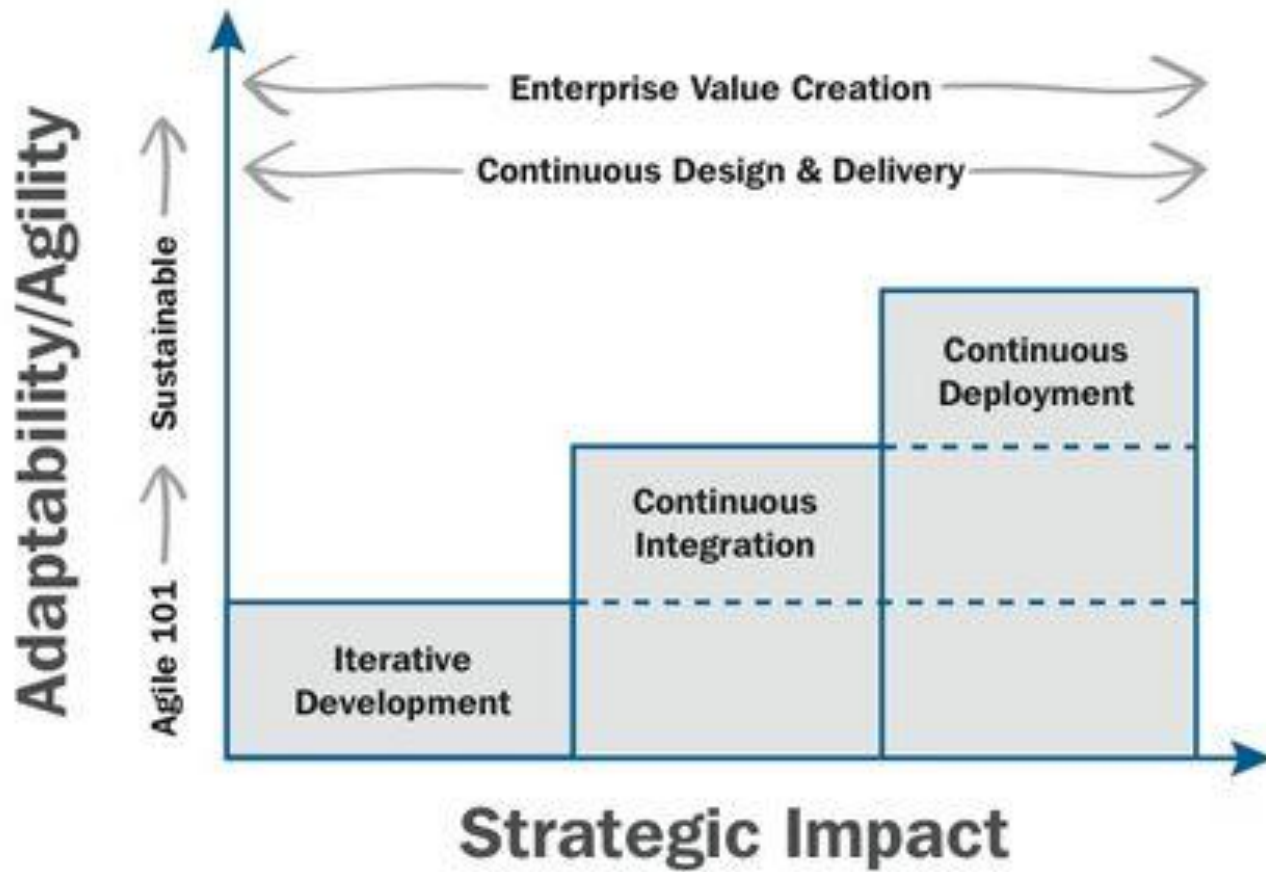
- <https://www.atlassian.com/agile/kanban/kanban-vs-scrum>
- <https://www.cprime.com/2015/02/3-differences-between-scrum-and-kanban-you-need-to-know/>

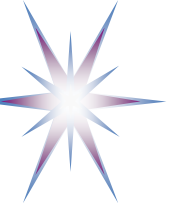


Kanban vs Scrum vs Agility: Which framework to choose?

- Regardless, choose one and stick with it, at least for a little while.
- We recommend running a whole sprint cycle if using scrum, or for as long as it takes to ship a few cards with your kanban board. Then, hold a retrospective and ask what went well and what went poorly.
- Choose scrum if your development team can devote 100% of their time to your project and it's important to ship value to customers on a regular basis.
 - Don't choose scrum if it's not the best possible fit for your team.
- Choose kanban if you value flexibility more than you value predictability and if your work maps nicely to a workflow.
 - Don't choose kanban if it's not the best possible fit for your team.
- Choose Agility if you value the agile principles but neither scrum nor kanban is a perfect fit.

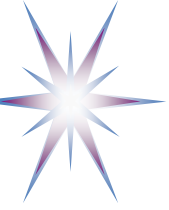
DevOps and Continuous Delivery/Deployment





Keys to Continuous Delivery

- Process should be automated
- Reduces the number of features introduced per release, minimizing shock to users
- Will reduce the standard release cycle
- Changes approach to releasing software from an event to a non-event
- Helps to avoid off-hour, high risk, expensive deployments
- Know your rollback plan (do you rollback or roll forward only)



The Continuous Delivery Pipeline

To successfully perform Continuous Delivery:

- Only build artifacts once
- Artifacts should be immutable
- Deployment should go to a copy of production before going into production
- Stop deploys if a previous step fails
- Deployments should be idempotent (i.e. if successful can be repeated and in different conditions)



DevOps: Deployment Improvement Practices

Some common things that DevOps teams do to improve software releases:

- Increase efficiency – less waste
- Decrease time to commit software changes
- Automate tests
- Identify defects/issues quickly
- Automate the build process
- Simplify the deployment process
- Make deployments reproducible
- Automate as much as possible



Continuous Delivery Benefits

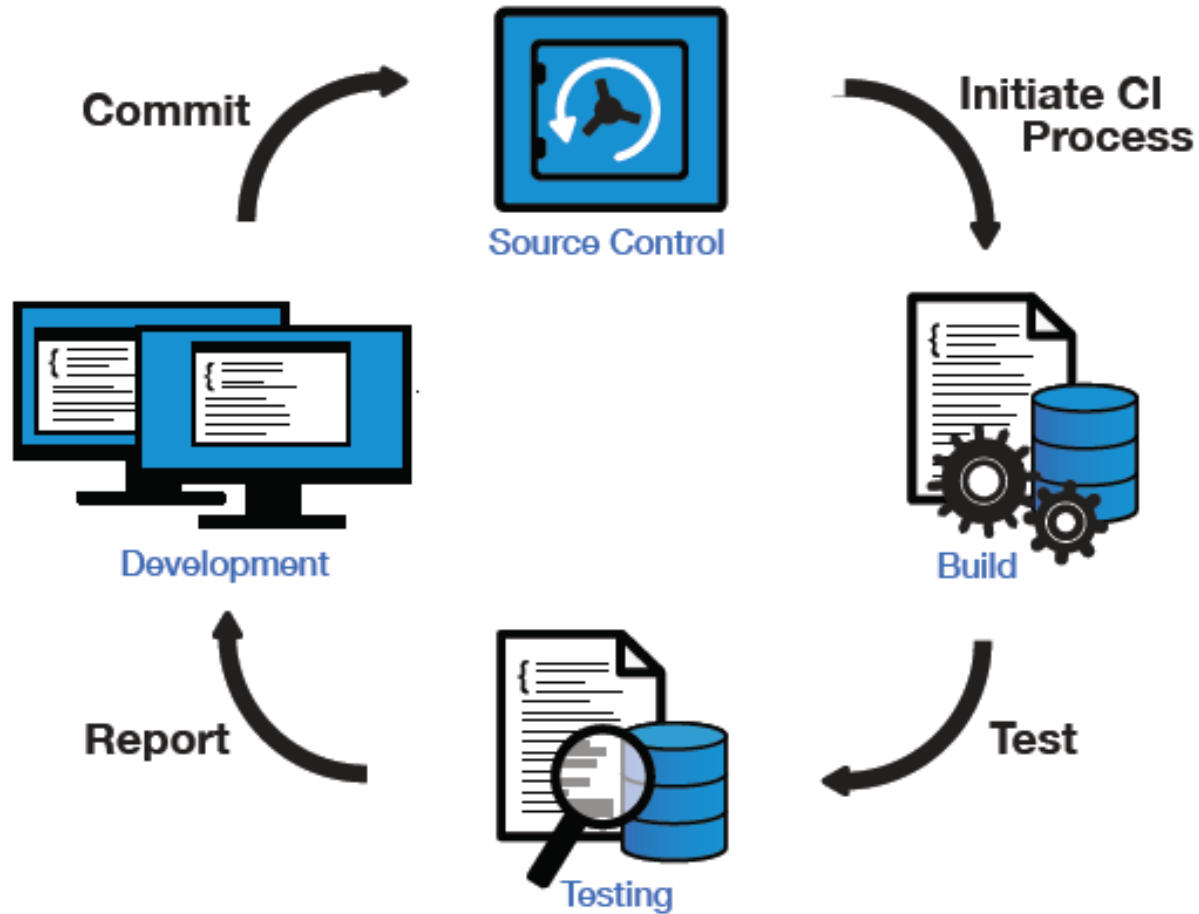
Small + Fast = Better

Continuous Delivery has the following benefits:

- Time to market goes down
- Quality increases not decreases
- Limits your Work In Progress
- Shortens lead times for changes
- Improves Mean Time To Recover



Continuous Integration





Principles of Continuous Integration

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Facilitate automated deployments



Continuous Integration Practices

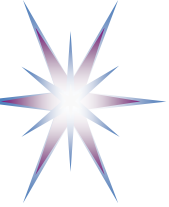
To successfully perform Continuous Integration:

- Builds should pass the coffee test (< 5 minutes)
- Commit really small bits
- Don't leave the build broken
- Use a trunk based development flow
- Don't allow flaky tests, fix them!
- The build should return a status, a log, and an artifact



How CI Improves Efficiency

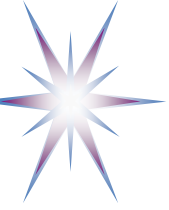
- Simplify Merges
- Rapid Feedback
 - Identify problems early
 - Makes bugs easier to find
- Reduce bug accumulation
- Visibility (team and stakeholders)
- Builds Automated
 - minimizes manual intervention
 - plug-ins (i.e. for static code analysis, gathering metrics)
- Precursor to Continuous Delivery & Deployment



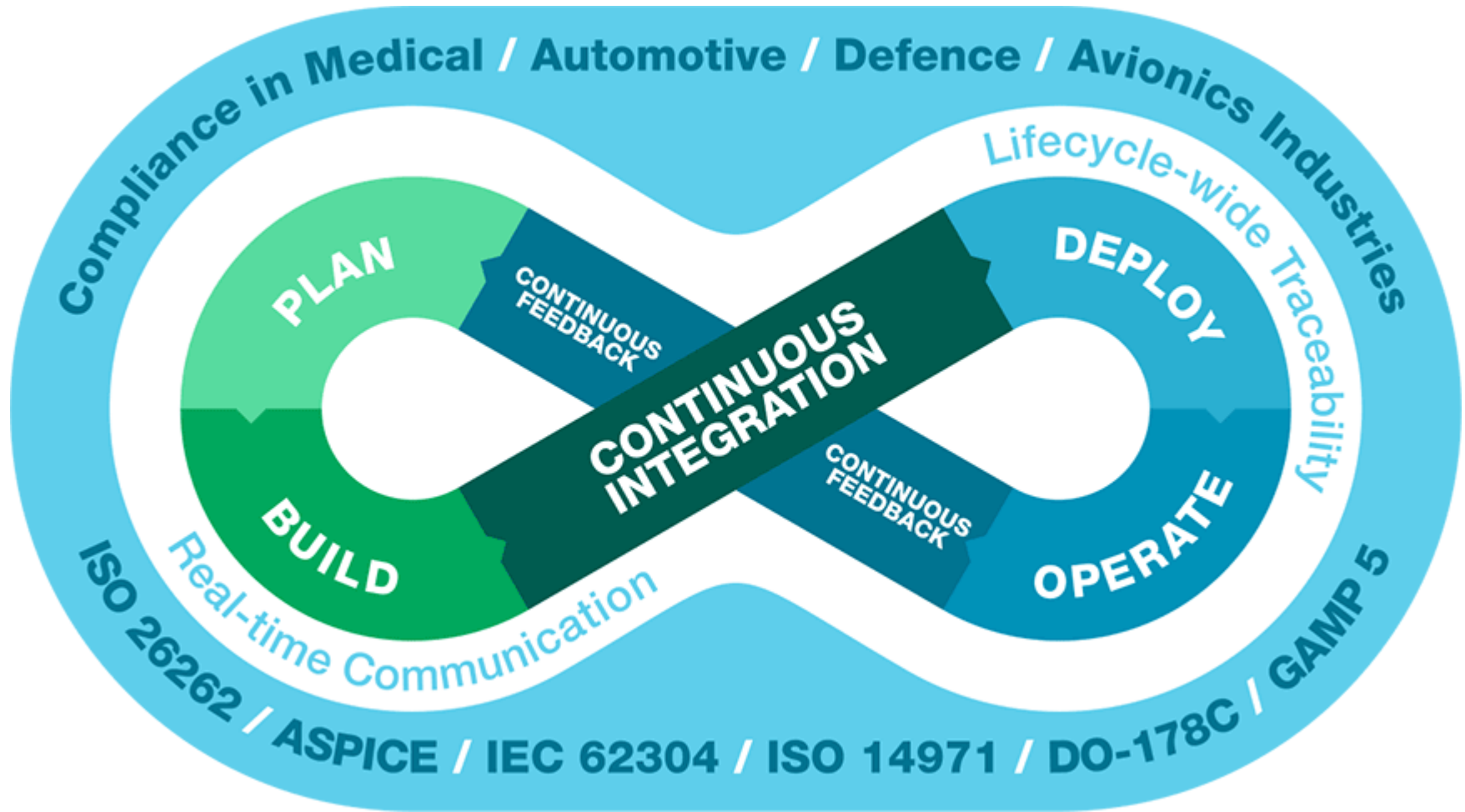
Operate For Design: Metrics and Monitoring

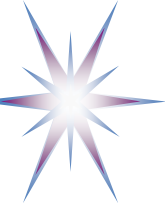
The 6 Monitoring Areas

- Service Performance and Uptime
- Software Component Metrics
- System Metrics
- App Metrics
- Performance
- Security

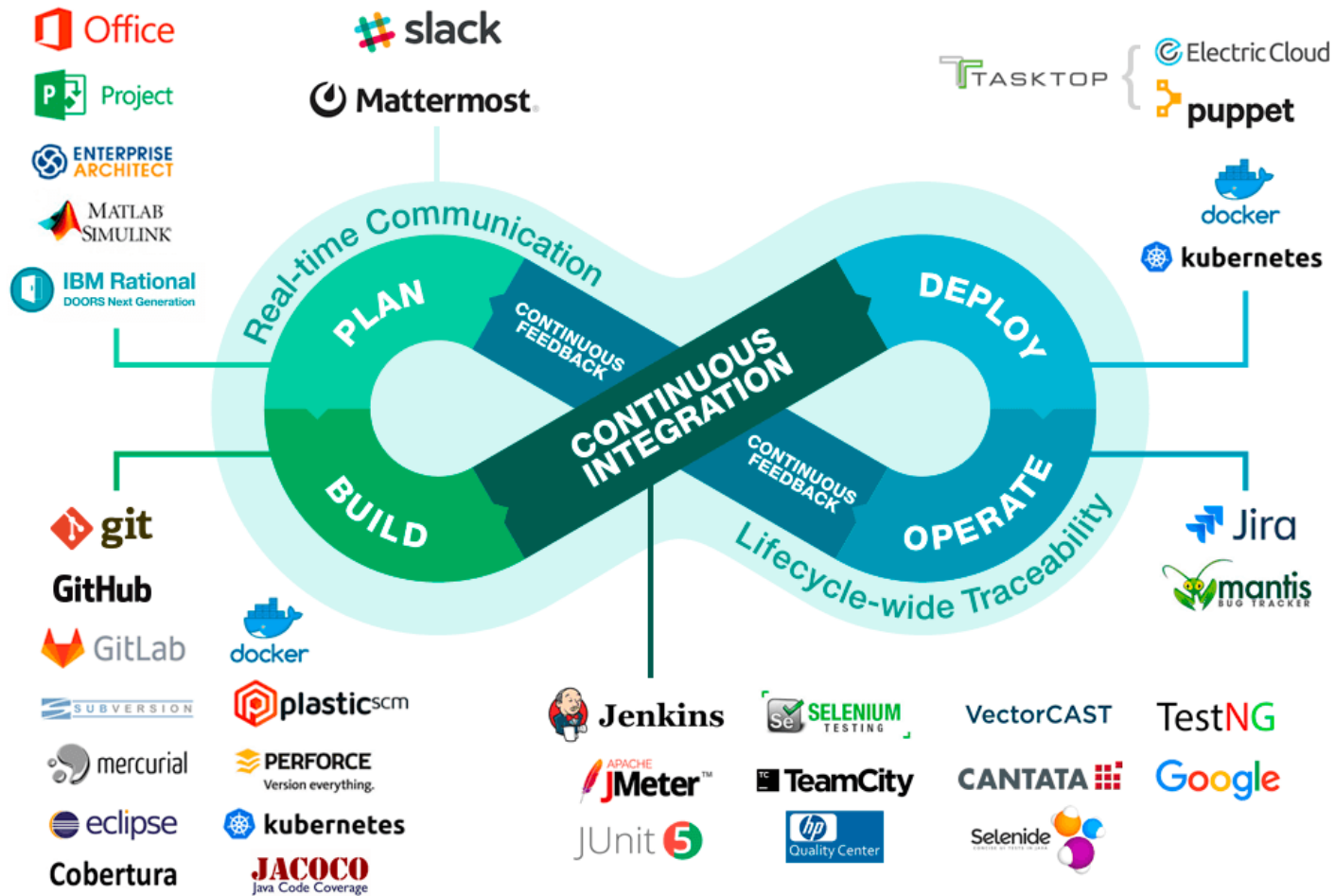


DevOps ecosystem and standards

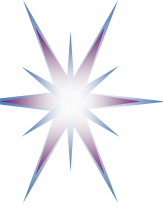




DevOps Tools



<https://intland.com/devops-it-operations/>



DevOps Toolchain – Classical, enabled by clouds

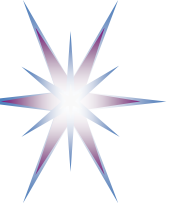
DevOps is a cultural shift and collaboration between development, operations and testing, enabled by DevOps toolchain

- **Code** — Code development and review, version control tools, code merging
- **Build** — Continuous integration tools, build status
- **Test** — Test and results determine performance
- **Package** — Artifact repository, application pre-deployment staging
- **Release** — Change management, release approvals, release automation
- **Configure** — Infrastructure configuration and management, Infrastructure as Code tools
- **Monitor** — Applications performance monitoring, end–user experience



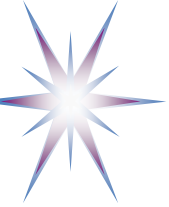
The DevOps Trends

- DevOps of everything – DataOps, DevBizOps, etc
- Cloud based DevOps environment
- From VM to Container to Serverless
- DevOps for Security: DevSecOps
 - Mind Vulnerabilities of Open Source Software
 - Code inspection, Compliance checking
 - Automated testing for known Security Vulnerabilities



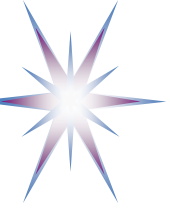
Acceptance of Agile and DevOps by US Military (2018)

- Recognizing this, the US defense community asked several advisory groups—including the Defense Science Board (DSB), the Defense Innovation Board (DIB) and the Intelligence and National Security Alliance (INSA) - to analyze the problems and recommend solutions.
 - These advisory groups strongly engaged commercial high tech industries such as Google, Facebook, IBM, Microsoft, and others. The advice from the advisory groups is very similar.
- They all recognized that software development is different from hardware procurement, and that software is never finished. They strongly recommend replacing the traditional software acquisition process with variants of disciplined agile methods that emphasize
 - requirements based on “goals and outcomes and use cases” rather than long lists of detailed, rigid specifications; and
 - small development teams (~10–15 professionals), who develop and deliver workable software iteratively in short sessions (2–4 weeks), and continual or continuous software builds and tests.



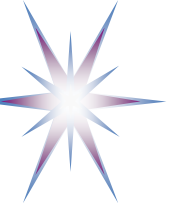
Wrap up and Take away

- DevOps is a result of all technologies development and integration to increase organisations efficiency by improving and optimizing IT infrastructure management
- DevOps pipeline and processes are enabled by cloud based technologies to automate CD/CI
- Agile is essential part of DevOps extended to the whole organisational product or service
- Other technologies next to Agile can be considered: Kanban, Kaizen, Kata, Lean – to select, adopt, combine
- DevOps and Agile require new mind set for developers of the 21st Century and culture change for companies



Reflection and discussion

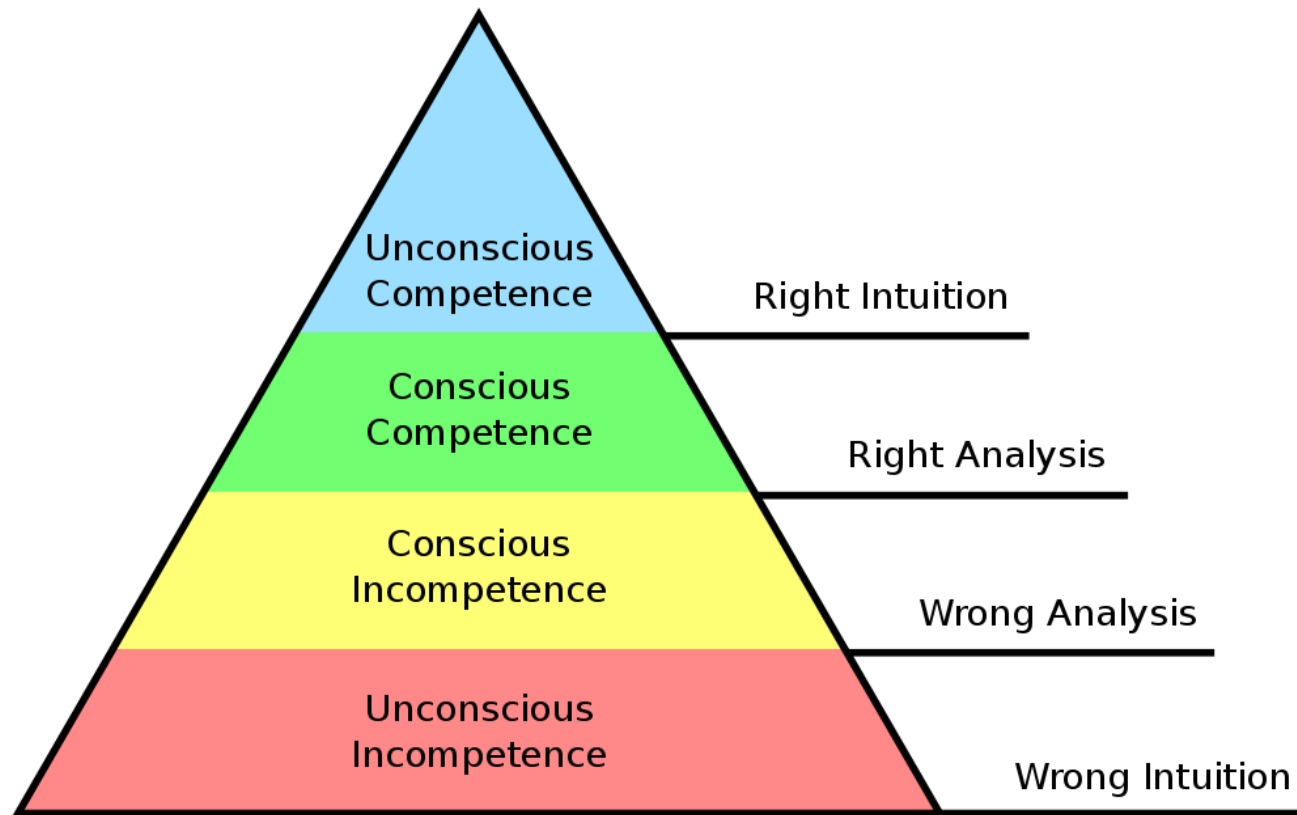
- Can you identify additional use cases for DevOps benefits?
- DevOps looks like closer to management than programming. How it can improve quality of SE?
- Why my company is (not/not enough) using DevOps?
- Do I/my team need to learn clouds to implement DevOps?
- What motivates companies to change to DevOps?
- What are limitations for DevOps and Agile?



References and further reading

- Literature list by topics on Canvas

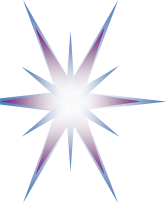
Learning DevOps: Competence Stages



Hierarchy of Competence

[ref] Competence Stages – Wikipedia

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



Applying the model to the Cloud - Observation

<https://www.cloudtp.com/doppler/four-stages-cloud-competence/>

This model has many parallels to how large organizations adapt to Big Data Infrastructure Technologies (BDIT) typically cloud based. The following stages of cloud adoption are based on observations how organisations progress through the learning curve.

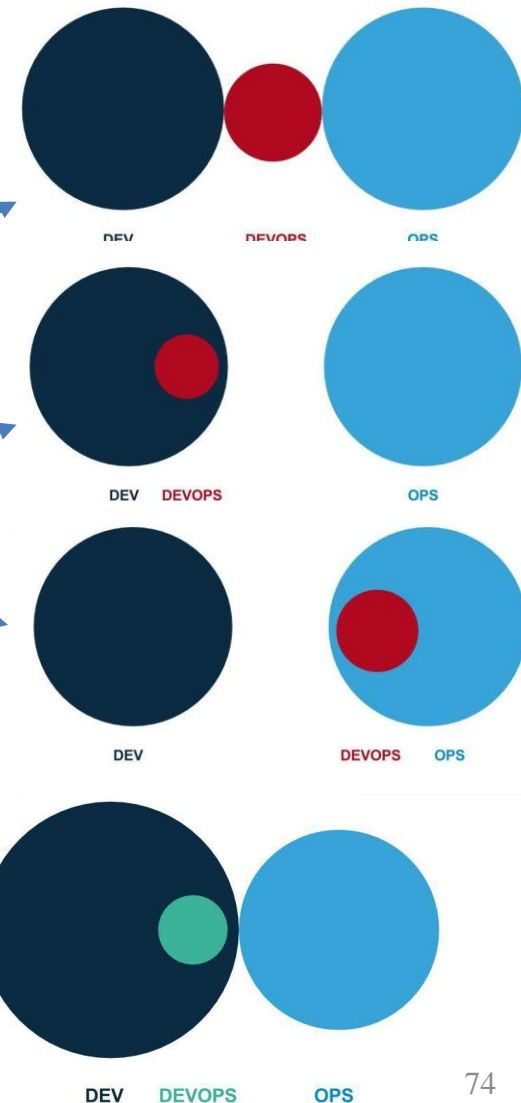
- **Stage 1 – Cloud Denial**
 - At the unconscious incompetence stage, the lack of knowledge of the underlying technology, organizational impact, and potential business value causes organizations to deny the usefulness of cloud computing. Organizations in this stage dispute the benefits of cloud and use things like security, compliance, and outages as justification for continuing to run IT with a legacy data center mentality.
- **Stage 2 – Do-It-Yourself (DIY) Cloud**
 - At the conscious incompetence stage, many either see tangible value in the cloud or have a mandate from the C-suite to go cloud. However, these organizations don't necessarily trust cloud providers, especially public cloud vendors, and they continue to apply their legacy data center thinking to the cloud architectures that they build. These organizations are turning their companies into infrastructure companies, instead of turning them into software companies.
- **Stage 3 – Cloud based Transformation**
 - Those at the conscious competence stage have a year or two of hands on experience with the cloud and a solid understanding of IaaS. Now that those driving the change understand the underlying technologies, the organizational impacts, and the potential business value, they often start looking for ways to accelerate their cloud adoption programs. These same companies also understand that they must transform the way they build software and start looking to DevOps/DataOps as a way to become more agile in the cloud. In this stage, cloud adoption is intentional and rapid.
- **Stage 4 – “All in” the Cloud**
 - At the unconscious competence stage, building solutions in the cloud becomes natural. These companies transformed the way software is built and delivered and have created great value for their business. They also start embracing PaaS, they understand that there is little to no business value in infrastructure and the application stacks. The value received from cloud computing is derived from delivering on business requirements that drive more revenue, creating higher level of services for customers, and getting to market before their competitors.
 - **Companies in stage 4 are disrupting industries.**



The Four Stages of DevOps Maturity

<https://www.forbes.com/sites/mikekavis/2017/11/17/the-four-stages-of-devops-maturity/#116ea7e2f625>

- **Stage 1 – DevOps Denial and Misinterpretation**
- **Stage 2 – Automation for the Sake of Automation**
 - The *Conscious Incompetence* stage is usually present in the first 12 to 18 months of an organization's DevOps journey.
 - Anti-pattern 1 - The DevOps Silo
 - Anti-pattern 2 - Dev Don't Need No Stinkin' Op
 - Anti-pattern 3 - Rebranding Sysadmins as DevOps Engineers
- **Stage 3 – Collaboration and Reorganization**
- **Stage 4 – A High Performing Organization**





Continuous Delivery Maturity Matrix

