DevOpsPM

A Crash Course in DevOps and Continuous Delivery for Project Managers

Andy Dingfelder

Planit

PMI NEW ZEALAND CHAPTER

DevOps Crash Course

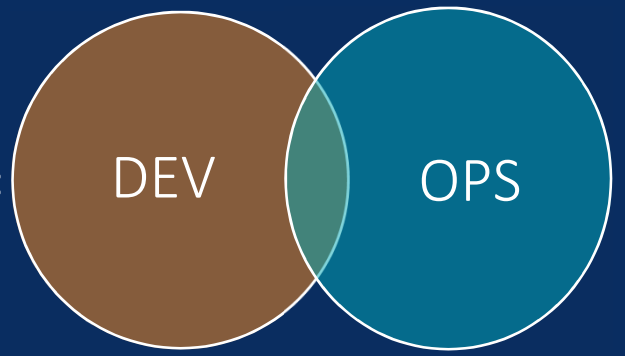
- DevOps Overview
- DevOps Culture
- Key Processes
- Practices
- Tools
- PM

What is DevOps?

The simple definition many use is a combination of **DEV**elopment & **OP**eration**S**

DevOps = Dev + Ops

A more complete answer is:

DevOps = 

[Collaboration] + Automation

A more complete picture → A **set of practices** that emphasizes the **collaboration** and **communication** of both software developers and other information-technology (IT) professionals while **automating** the process of software delivery and infrastructure changes.

Overview

What is DevOps?

Wikipedia defines DevOps as:

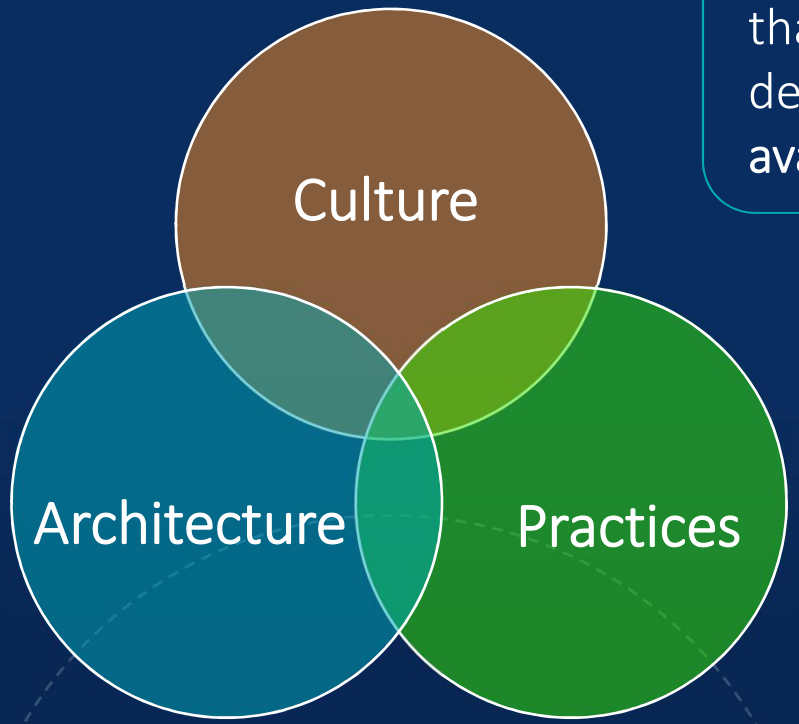
This definition is more complete as it adds business value and frequency

Overview

Culture, Technical Practices and Architecture

Another viewpoint is presented by **Gene Kim**, author of The Phoenix Project, a must read for anyone interested in learning about DevOps.

He defines DevOps as:



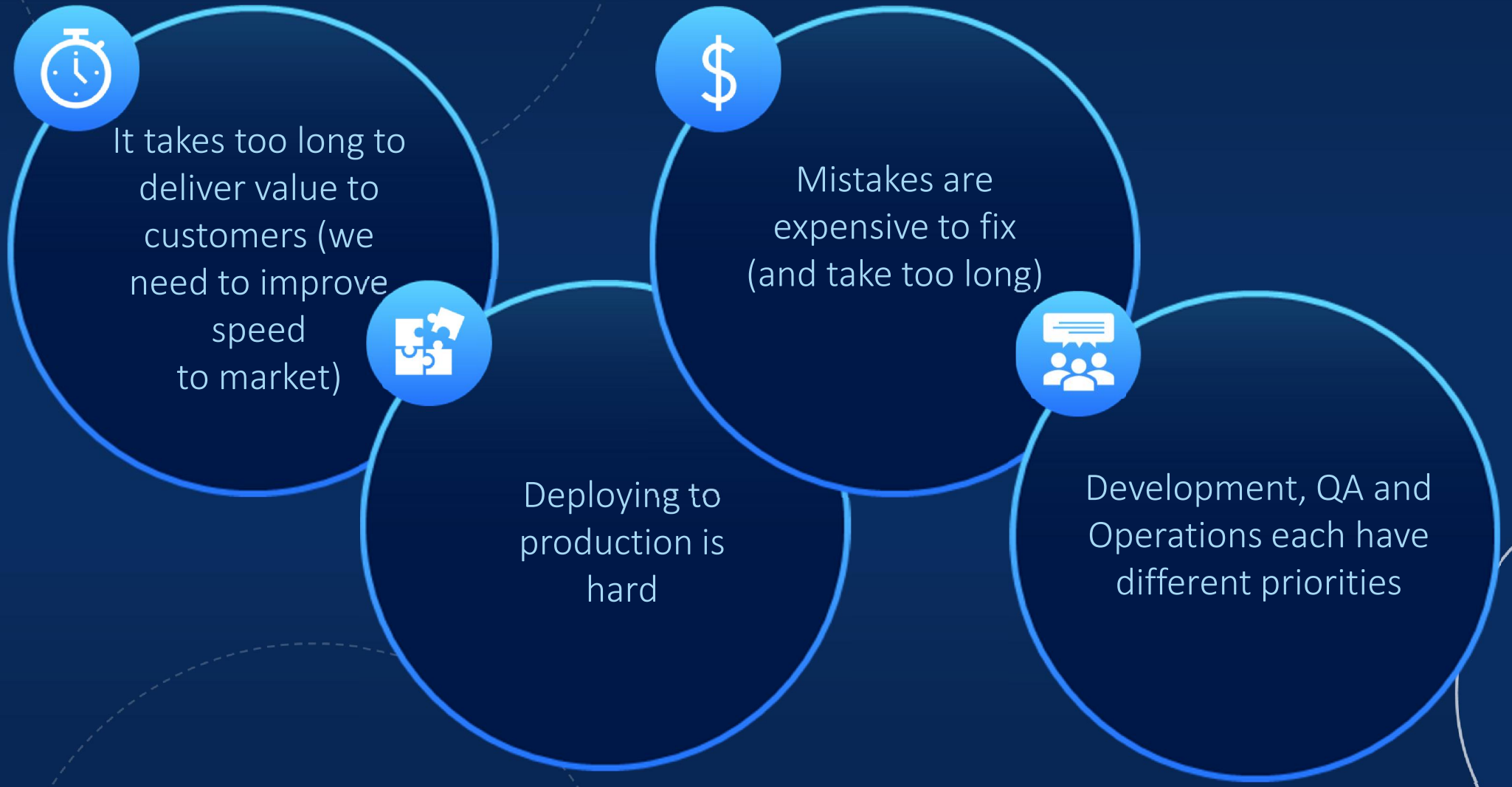
"The set of **cultural** norms, **technical practices** and **architecture** that enables organizations to have both a **fast flow of work** from development to deployment, as well as world-class **reliability, availability and security** [of information systems and IT services]."

< FAST >

Overview

Why Do We Need DevOps?

Common business complaints are:



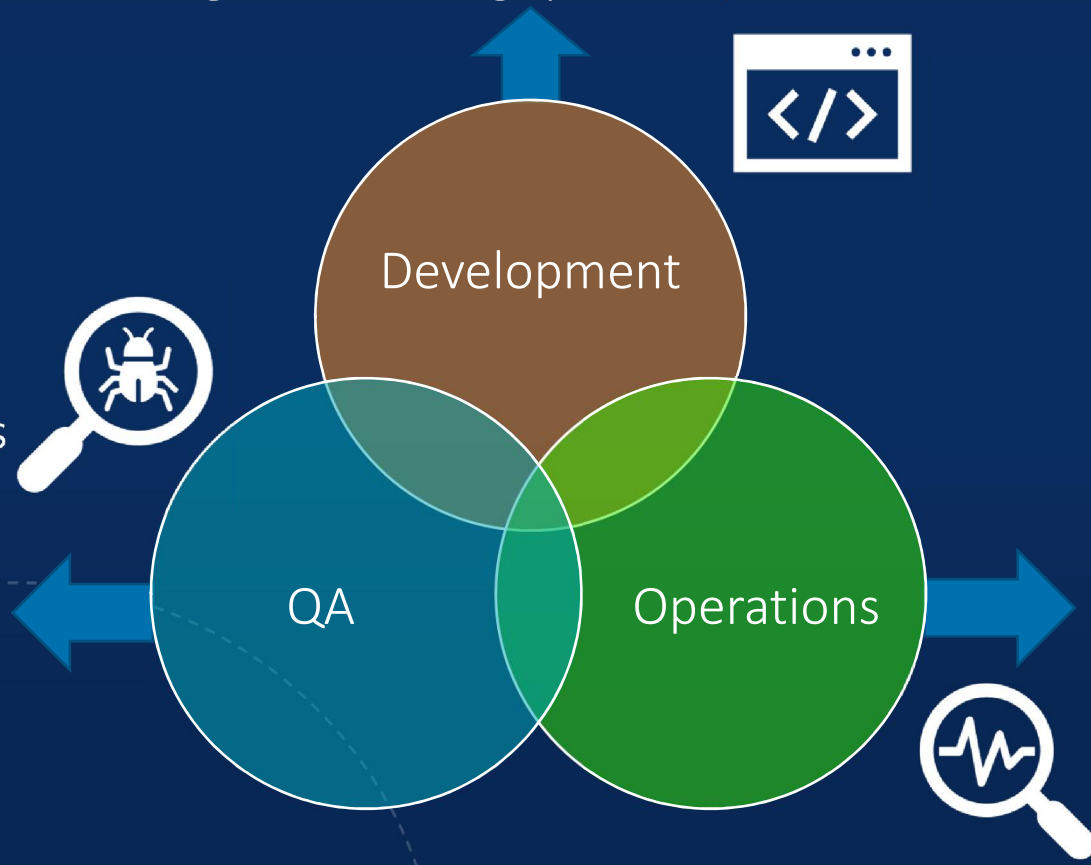
DevOps seeks to solve these and other business problems

Overview

Conflicting Priorities

Development, QA and Operations have **very different (conflicting) priorities**

Developers focus on delivering business value, and providing a powerful customer experience. They are concerned with product quality while sustaining or increasing speed of production delivery



Testers want to identify defects as early as possible in the lifecycle (shifting left) to minimize customer impact in production

Ops want to minimize production changes to decrease risk. They are rewarded for system stability (keep the ship floating and limit chaos in production)

Overview

High Performing Organisations

The following performance metrics can tell us a lot about successful organisations

Deployment Frequency:

how often do we deploy?

Lead Time:

how long does it take from code committed to code successfully running in production?

Measure of software delivery performance tempo

Mean Time to Restore (MTTR):

how quickly can a service be restored?

Change Fail Rate:

what percentage of changes to production fail?

Measures of reliability

From the 2019 DORA State of DevOps report

Overview

High Performing Organisations

These performance metrics can tell us a lot about successful organisations

Aspect of Software Delivery Performance	Elite	High	Medium	Low
Deployment frequency How often does your organization deploy code?	On-demand deploys (multiple per day)	Between once per hour and once per day	Between once per week and once per month	Between once per week and once per month
Lead time for changes What is your lead time for changes (i.e., how long does it take to go from code commit to production)?	Less than one hour	Between one day and one week	Between one week and one month	Between one month and six months
Time to restore service How long does it generally take to restore service when a service incident occurs?	Less than one hour	Less than one day	Less than one day	Between one week and one month
Change failure rate What percentage of changes result either in degraded service or subsequently requires remediation?	0-15%	0-15%	0-15%	46-60%

From the 2019 DORA State of DevOps report

Overview

What are high performing organisations doing?

TOOL USAGE BY PERFORMANCE PROFILE

	Low	Medium	High	Elite
A mix of proprietary tools, open source, and commercial off-the-shelf (COTS) software	30%	34%	32%	33%
Mainly open source and COTS, heavily customized	17%	8%	7%	10%
Mainly open source and COTS, with little customization	14%	21%	18%	20%
Primarily COTS packaged software	8%	12%	8%	4%
Primarily developed in-house and proprietary to my organization	20%	6%	5%	6%
Primarily open source, heavily customized	6%	7%	5%	12%
Primarily open source, with little customization	5%	12%	24%	15%

Note what low performing organisations are doing (or not doing)

AUTOMATION AND INTEGRATION BY PERFORMANCE PROFILE

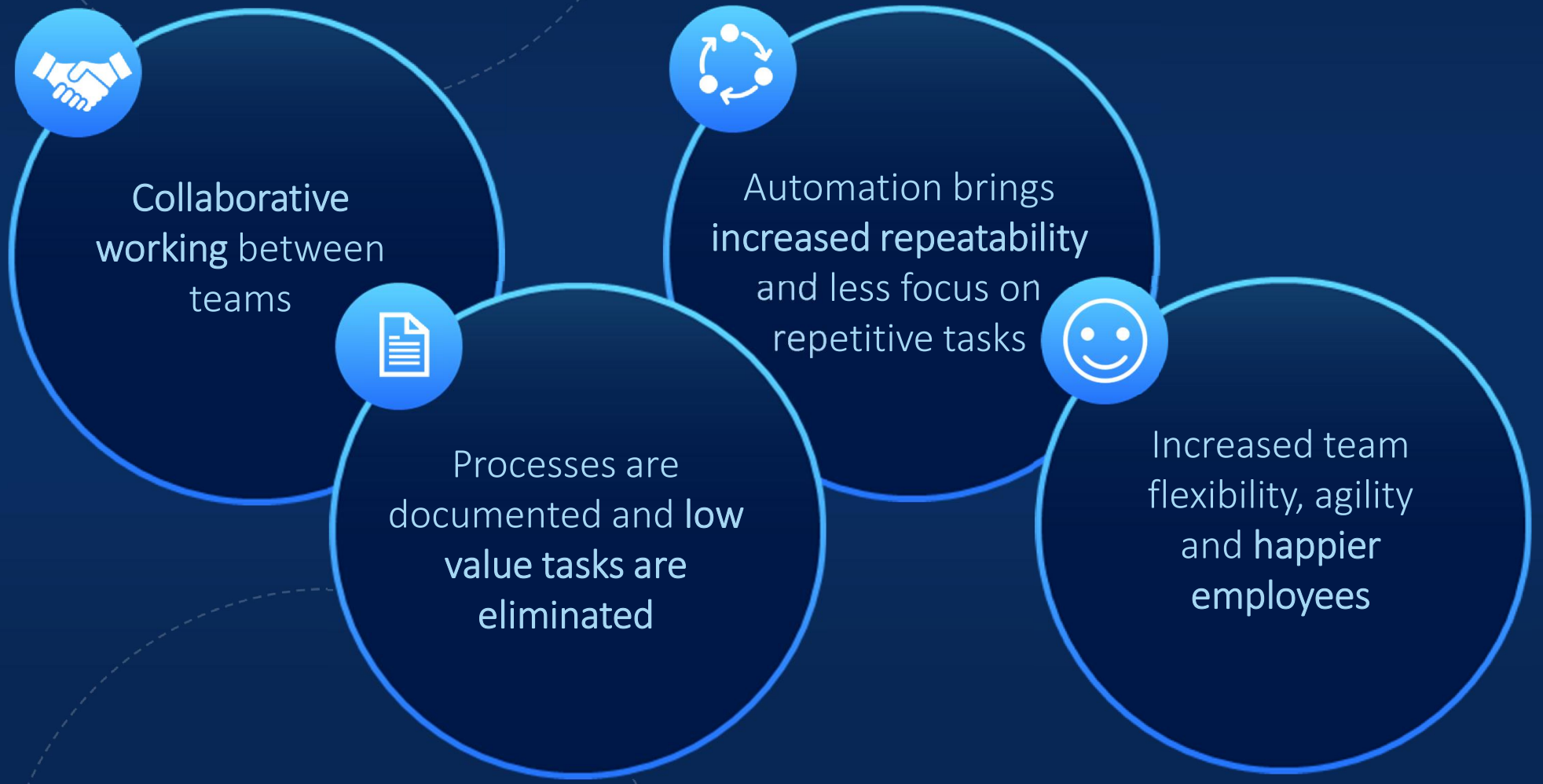
	Low	Medium	High	Elite
Automated build	64%	81%	91%	92%
Automated unit tests	57%	66%	84%	87%
Automated acceptance tests	28%	38%	48%	58%
Automated performance tests	18%	23%	18%	28%
Automated security tests	15%	28%	25%	31%
Automated provisioning and deployment to testing environments	39%	54%	68%	72%
Automated deployment to production	17%	38%	60%	69%
Integration with chatbots / Slack	29%	33%	24%	69%
Integration with production monitoring and observability tools	13%	23%	41%	57%

Puppet 2019 State of DevOps report

Overview

Benefits of DevOps

High performing DevOps organisations have common trends:

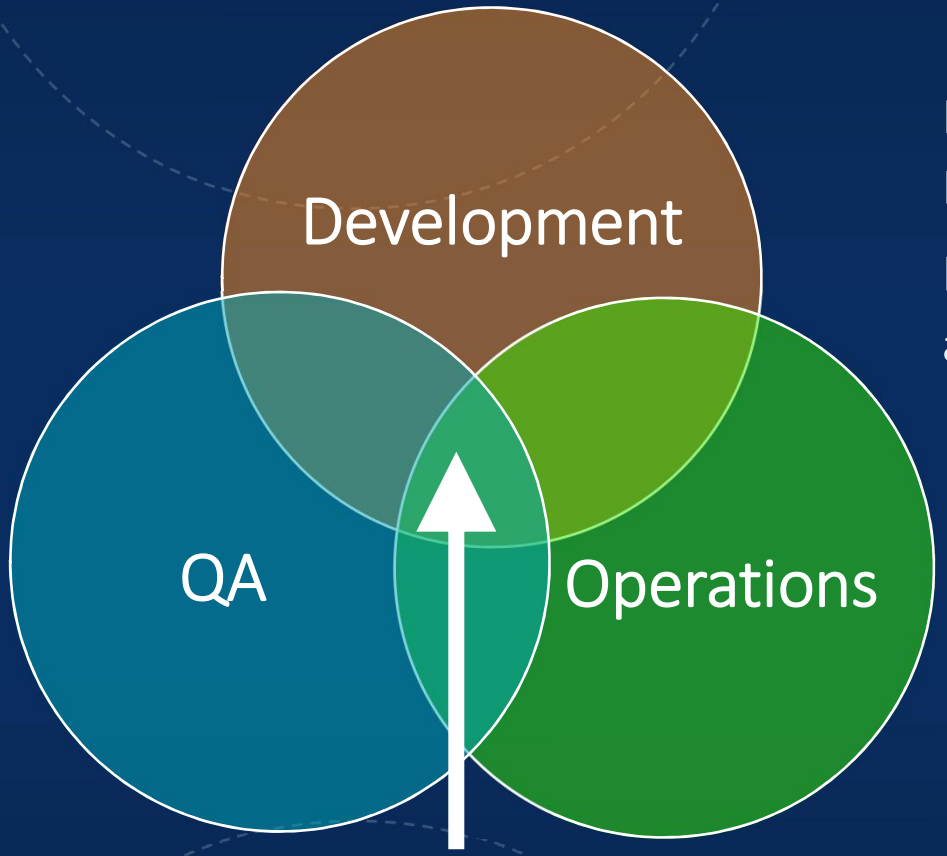


Overview

Benefits of DevOps - 2

Additional benefits of DevOps include:

A Cultural Shift Relentless Pursuit of Quality at All Stages



Relentless Pursuit of Quality

DevOps is more though than just collaboration between Development, QA and Operations

It is a **cultural shift** for all 3 groups to integrate quality across the entire SDLC, using:

- Continuous Integration
- Continuous Testing
- Continuous Delivery
- Continuous Monitoring
- Continuous Feedback
- Continuous Improvement

Overview

DevOps Culture

DevOps Overview

DevOps Culture

Key Principles

Practices

Tools

PM

Team Responsibility and Quality Focus

- **Product Quality & Customer Focus** - everyone on the team is responsible
- Highly **Collaborative** Culture
- **Automate Everything**
- **Relentless** Continuous Improvement
- **Blameless** Culture

Martin Fowler

Team Responsibility Highly Collaborative Culture

Everyone in the team has shared responsibility

Developers are responsible for their application support in production.

Everyone works collectively to build, ship and support the product

Everyone is equally responsible for the E2E health of the entire value stream. **You built it you support it**

Teams are Cross-Functional

Customer / Quality Focus **Everyone is responsible for quality**, not just the tester

Good Quality Engineering practices and cross-functional disciplines are required in each delivery team

Culture

Continuous Improvement

DevOps teams continuously strive to improve through:

Blameless Culture

In a blameless culture, everyone feels safe and no one is afraid to make mistakes.

Developers feel confident enough to express their ideas, take chances, and feel able to speak up about problems and risks.

Dev and Ops collaborate well, and everyone is aligned on the problem.

You've heard the saying: **Failure is not an option**.

In a blameless environment, failure is **always** an option, because it means that systems are always improving and innovation is always happening.

We won't learn and improve if we fear making mistakes.

Culture

Anti – Fragility

To build resilience we need to apply stress to the areas we need to improve on.

The **more frequently we apply pressure** in these areas **the more we improve**.

Reducing deployment lead times - **go faster to production**

Increase test coverage and decrease execution times **Test more, faster!**

Test early, test often, use automation to execute the right tests faster

Perform **Game-Days** to rehearse full-scale outages

An extreme example of this is Netflix injecting faults into production – Chaos Monkey / Simian Army

Culture

Key DevOps Principles

DevOps Overview

DevOps Culture

Key Principles

Practices

Tools

Lean

- Deliver Maximum Customer Value with Minimum Resource Waste
- Theory of Constraints

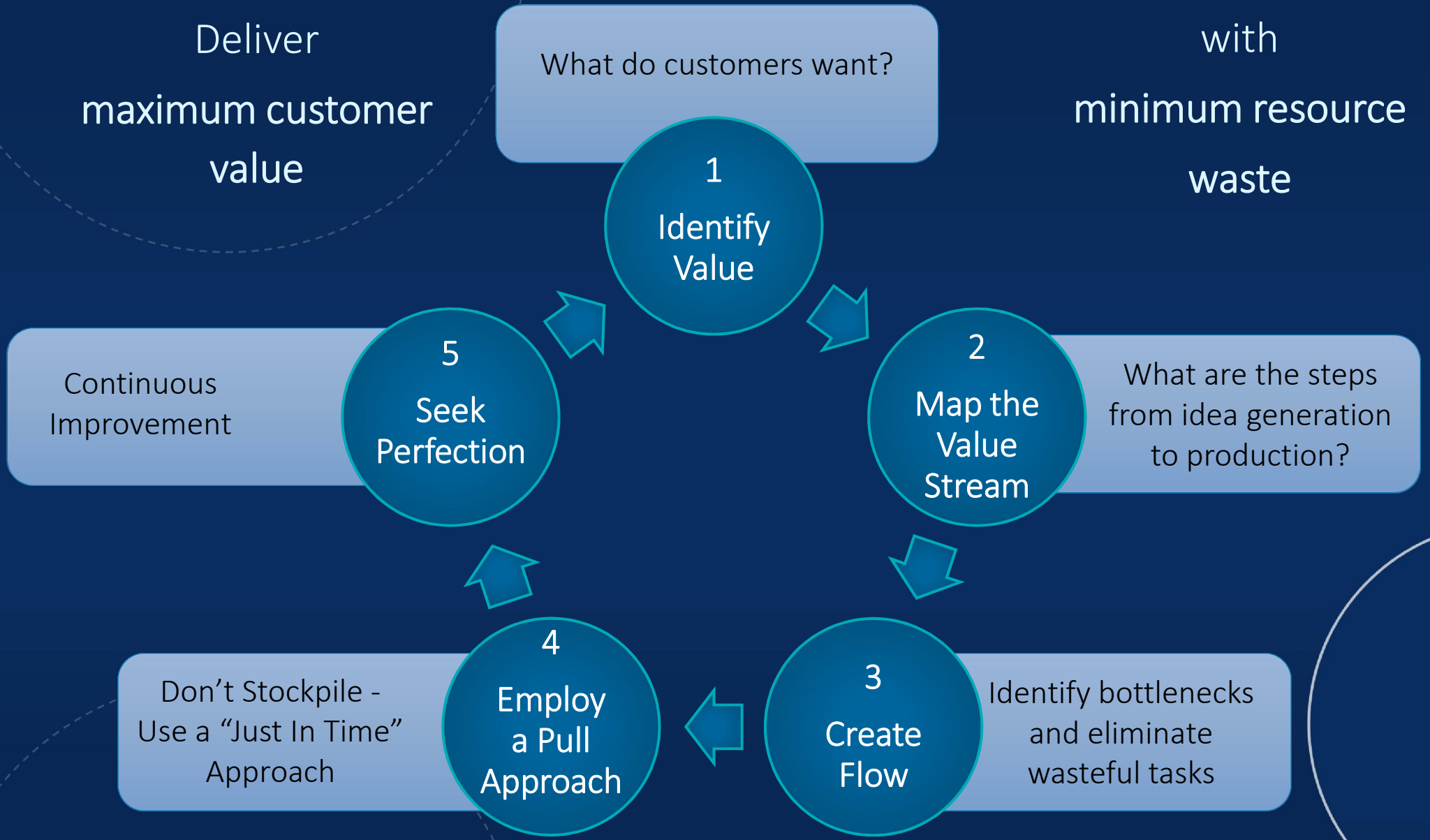
The Three Ways

- First Way: Principle of Flow
- Second Way: Principle of Feedback
- Third Way: Continual Learning and Experimentation

Lean

Deliver maximum customer value

with minimum resource waste

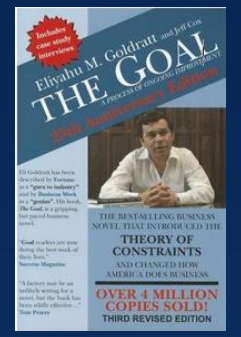


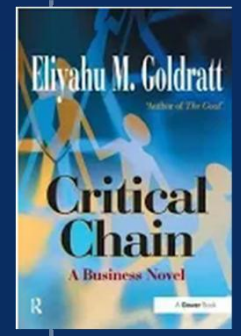
Key Principles

Theory of Constraints

The Theory of Constraints (ToC) was introduced by Eliyahu Goldratt

as an approach to identify and eliminate bottlenecks within technology value streams.





Reduce batch sizes

keep project work small

Limit the # of handoffs

shorten # of steps to deploy

Decrease WIP

restrict the # of projects or tasks in progress

Shorten & amplify feedback loops

improve the flow of work down the value stream

Key Principles

The 3 Ways

The 3 ways were described in the book **The Phoenix Project** by Gene Kim.

Highly recommended for anyone wanting to know more about DevOps

The 1st Way

Systems Thinking
&
Principle of Flow

The 2nd Way

Amplify
Feedback Loops
&
Feedback

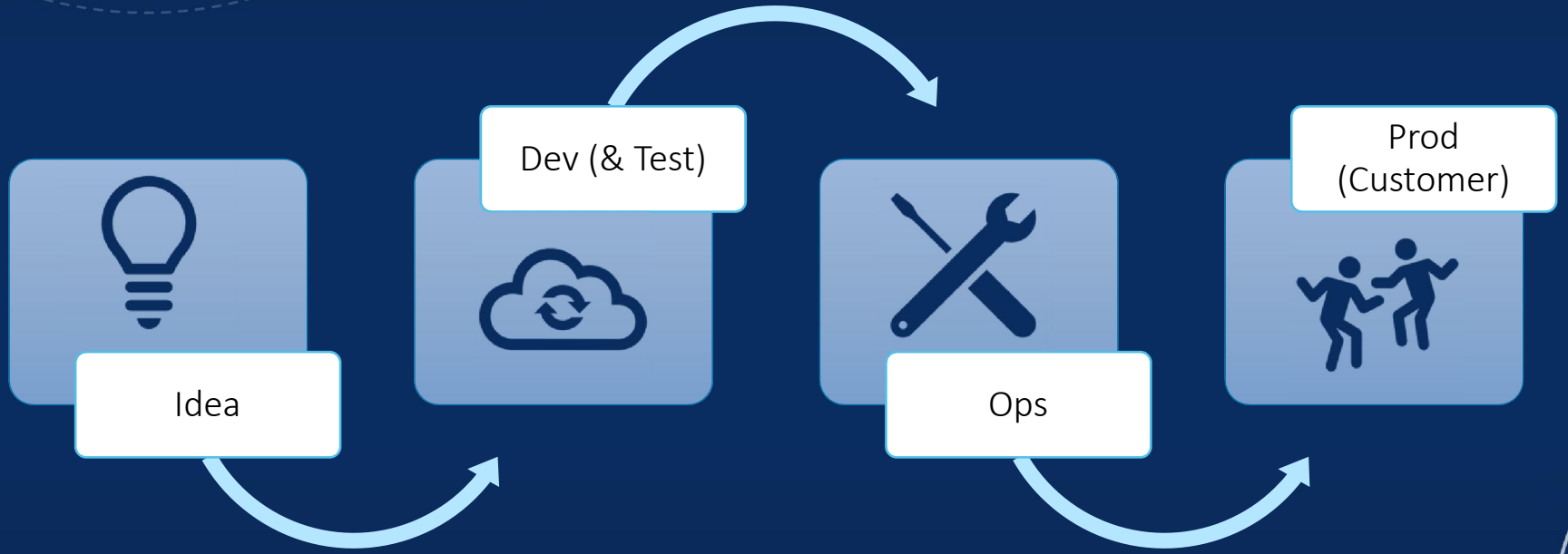
The 3rd Way

Continual
Learning
&
Experimentation

Key
Principles

The First Way

The First Way according to Gene Kim emphasizes the **performance of the entire system**, as opposed to the performance of a specific silo of work or department



The focus is on **value streams** enabled by IT, beginning with requirements, built in Development, and then transitioned into IT Operations, where the value is then delivered to the customer as a form of a service – using Agile and newer ways of working, including SAFe

Key Principles

The First Way

Never pass a known defect to downstream work centers

Never allow local optimization to create global degradation

Always seek to increase flow

Always seek to achieve a profound understanding of the system

Key Principles

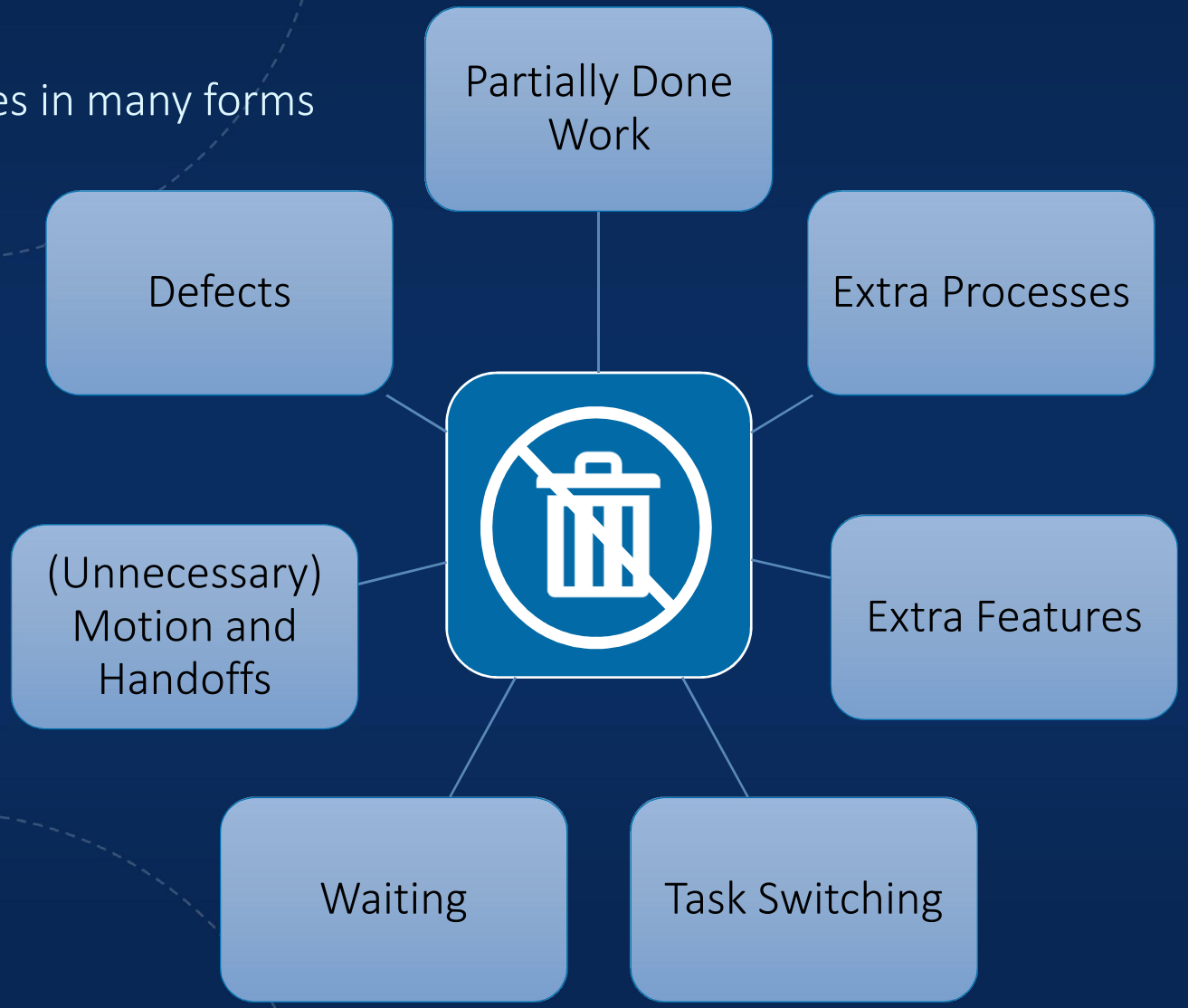
The First Way – Key Principles

- Make Work Visible
- Limit Work In Progress
- Reduce Batch Sizes
- Reduce Handoffs
- Identify and Elevate Constraints
- Eliminate Waste

Key Principles

Eliminating Waste – one of the Frist Way principles

Waste comes in many forms



Key Principles

The Second Way

The Second Way is about creating **Right to Left feedback loops.**

The goal of many process improvement initiatives is to shorten and amplify feedback loops, so necessary corrections can be continually made.

Understanding and responding to all customers, internal and external

Embedding knowledge where we need it.

Shortening and amplifying all feedback loops

Key Principles

The Second Way – Key Principles

- Establish an upstream feedback loop
- Shorten the feedback loop
- Amplify the feedback loop (Self Reinforcing Loops)

DevOps requires constant feedback,
with a goal of finding issues quickly
so that corrections can continually be made.

Key Principles

The Third Way

Experimentation and taking risks allow us to keep pushing to improve.

We need to learn from our mistakes and keep striving to improve

Shorten & amplify feedback loops, so necessary corrections can be continually made.

Allocate time for the improvement of daily work

Create rituals that reward the team for taking risks

Key Principles

The Third Way – Key Principles

- Promote experimentation
- Learn from success and failure
- Constant improvement
- Seek to achieve mastery through practice

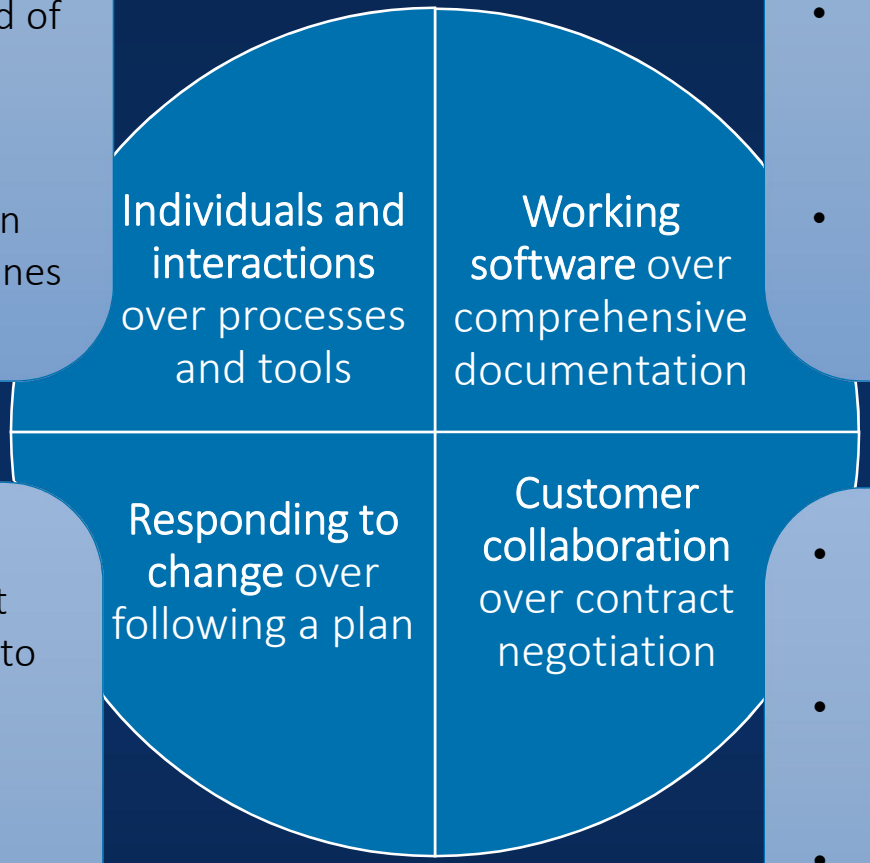
Don't work in a silo –

Include Organizational Learning

Examples include Netflix intentionally introducing faults 100s of times a day into the system to increase resilience.

Key Principles

Agile Manifesto



- Business people and developers must work together daily throughout the project.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

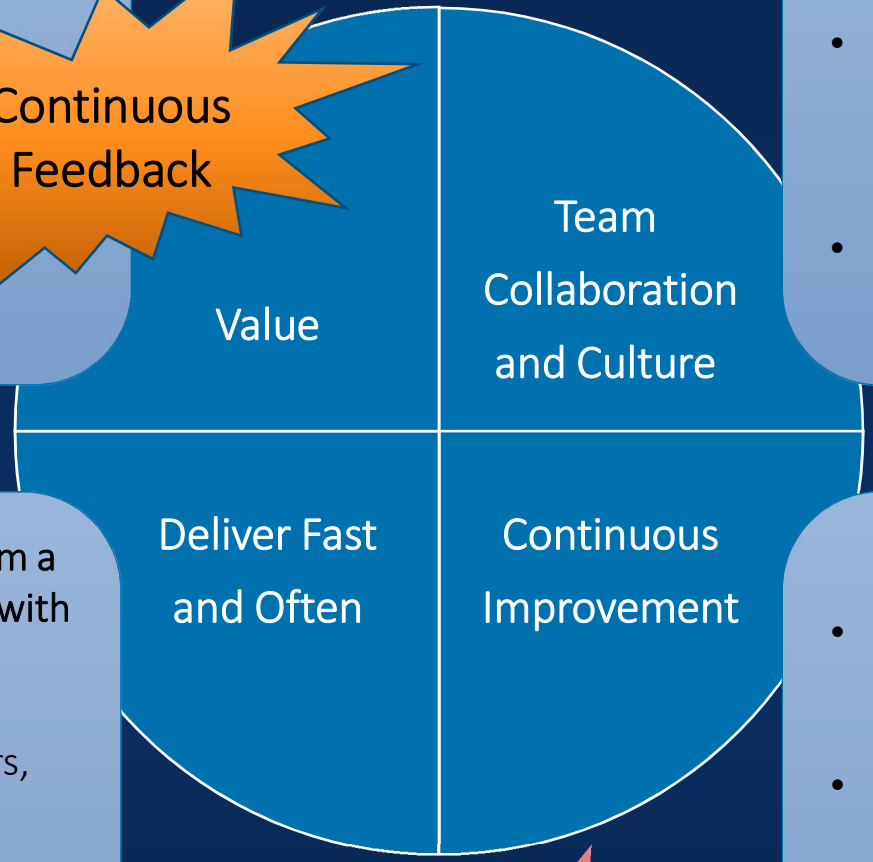
- Simplicity--the art of maximizing the amount of work not done--is essential.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Working software is the primary measure of progress.

12 Agile Principles

- Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
- Continuous attention to technical excellence and good design enhances agility.
- The best architectures, requirements, and designs emerge from self-organizing teams.

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

How do DevOps Principles relate to the 12 Agile Principles?



- Working software is the primary measure of progress.
- Welcome changing requireme in development. Agile process change for the customer's com advantage.
- Business people and develope together daily throughout the

Continuous Feedback

Lean

- [illegible] ftware frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Our highest priority is to customer through early delivery of valuable soft

Automation

Flow

- Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
- The most efficient conveying informa development team conversation.
- The best architectu designs emerge fro[illegible]

Blameless Culture

- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
- Continuous attention to technical excellence and good design en
- Simplicity – the art of work not done –

Relentless Improvement

DevOps Practices

DevOps Overview

DevOps Culture

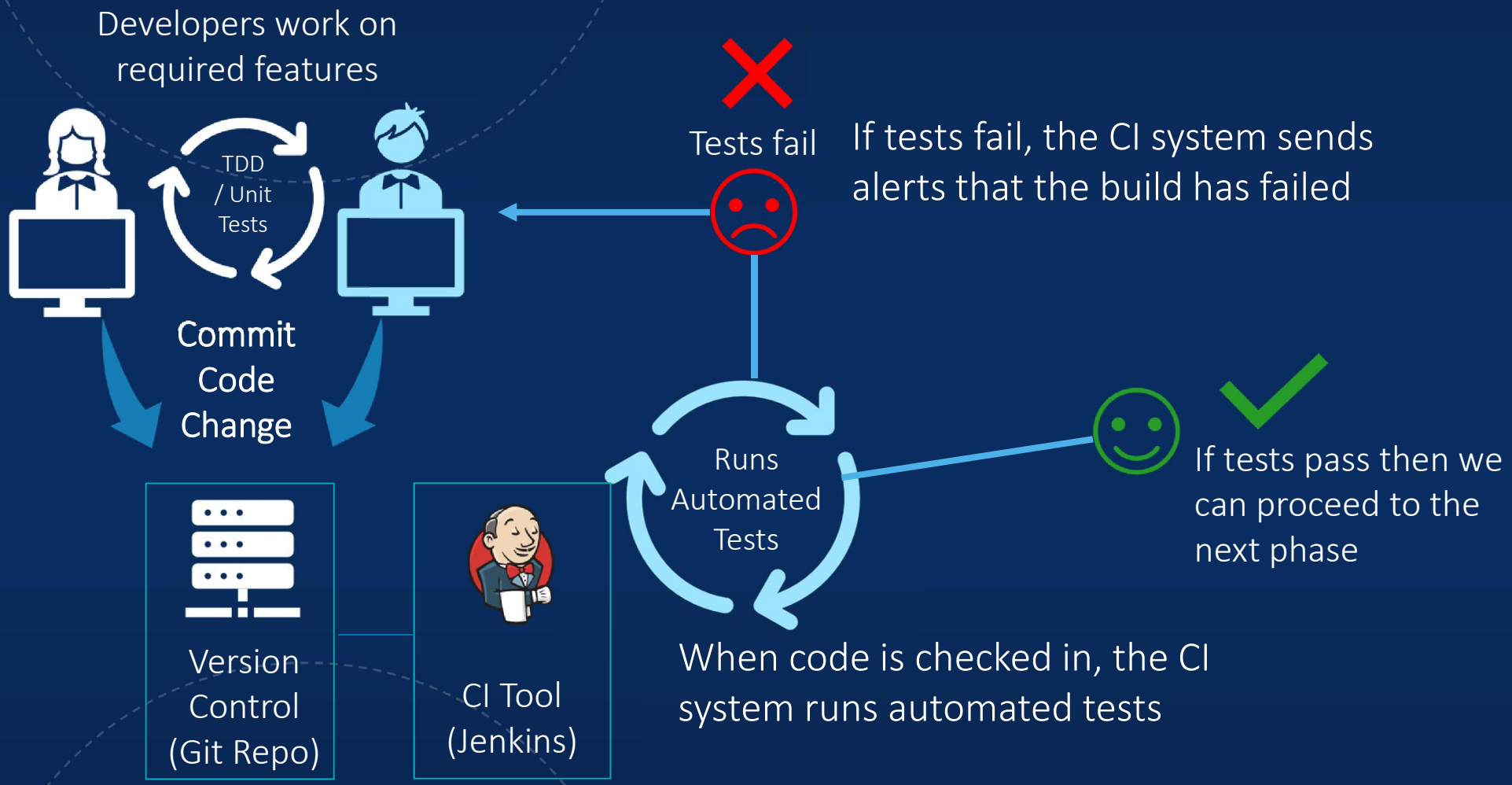
Key Processes

Practices

Tools

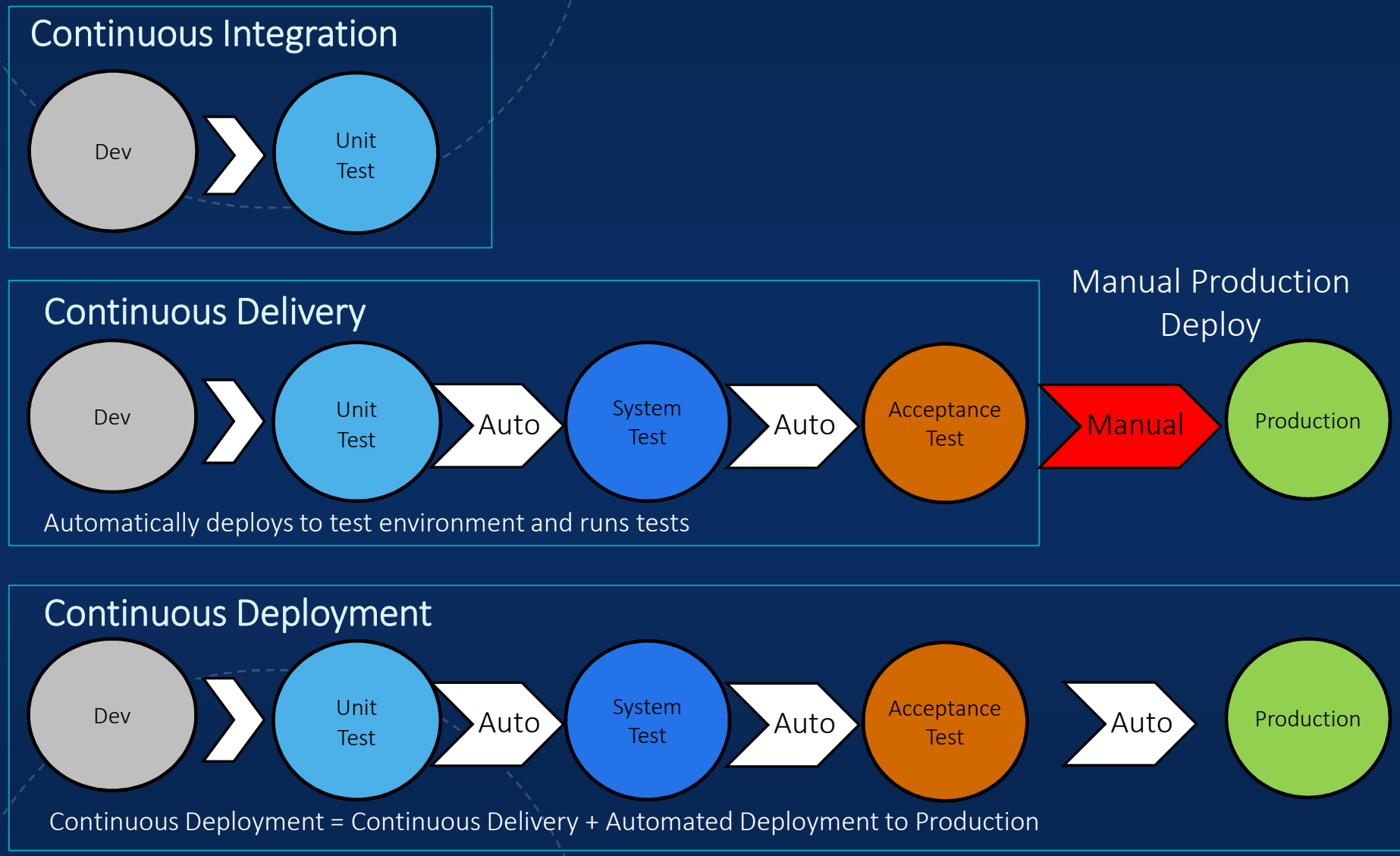
PM

Continuous Integration



DevOps Practices

Continuous Integration, Deployment & Delivery



DevOps Practices

Continuous Testing

The process of executing **automated tests** as part of the software delivery pipeline, in order to Identify and assess business risks associated with a software release candidate as **rapidly** as possible.

If there are business risks with our release, we want to find them quick

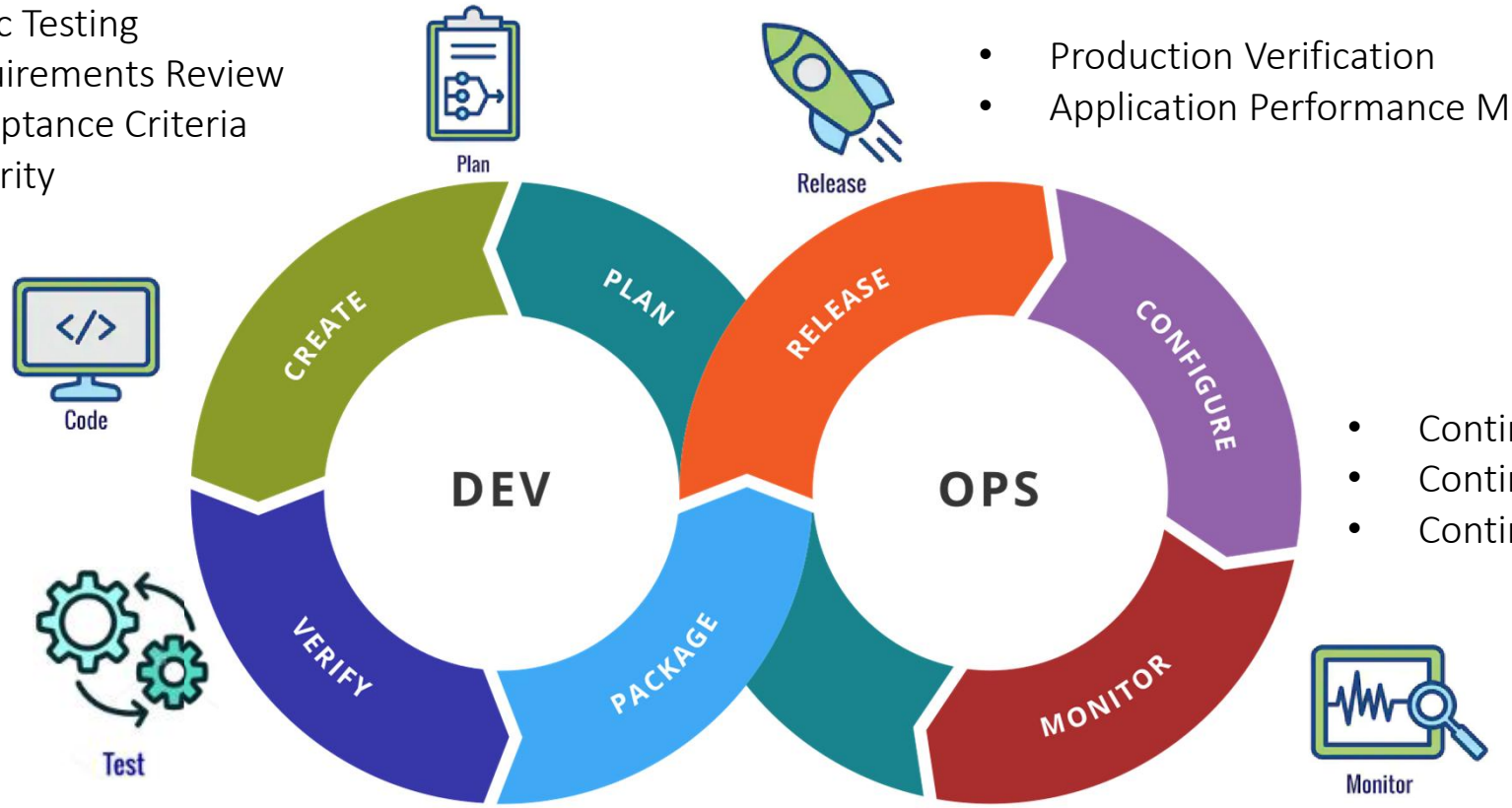
We want to ensure quality across all phases of the SDLC

DevOps Practices

Continuous Testing – Complete Quality Coverage

- Static Testing
- Requirements Review
- Acceptance Criteria
- Security

- Production Verification
- Application Performance Management



Functional

- Unit
- System
- System Integration
- Acceptance
- TDD, BDD, SBE

- Continuous Monitoring
- Continuous Feedback
- Continuous Improvement

Non-Functional

- Performance
- DevSecOps (Security, Penetration, Compliance, etc)
- Compatibility, Usability, Accessibility, etc

- Infrastructure as Code
- Continuous Integration & Delivery
- Service Virtualisation
- Automate Everything
- Environment Management

DevOps Tools

DevOps Overview

DevOps Culture

Key Processes

Practices

Tools

PM

Each DevOps practice area has its own toolset

Teams need to have the flexibility to choose the right tool for each task

Tool Examples

DevOps Tools

Where does Project Management fit in?

DevOps Overview

DevOps Culture

Key Processes

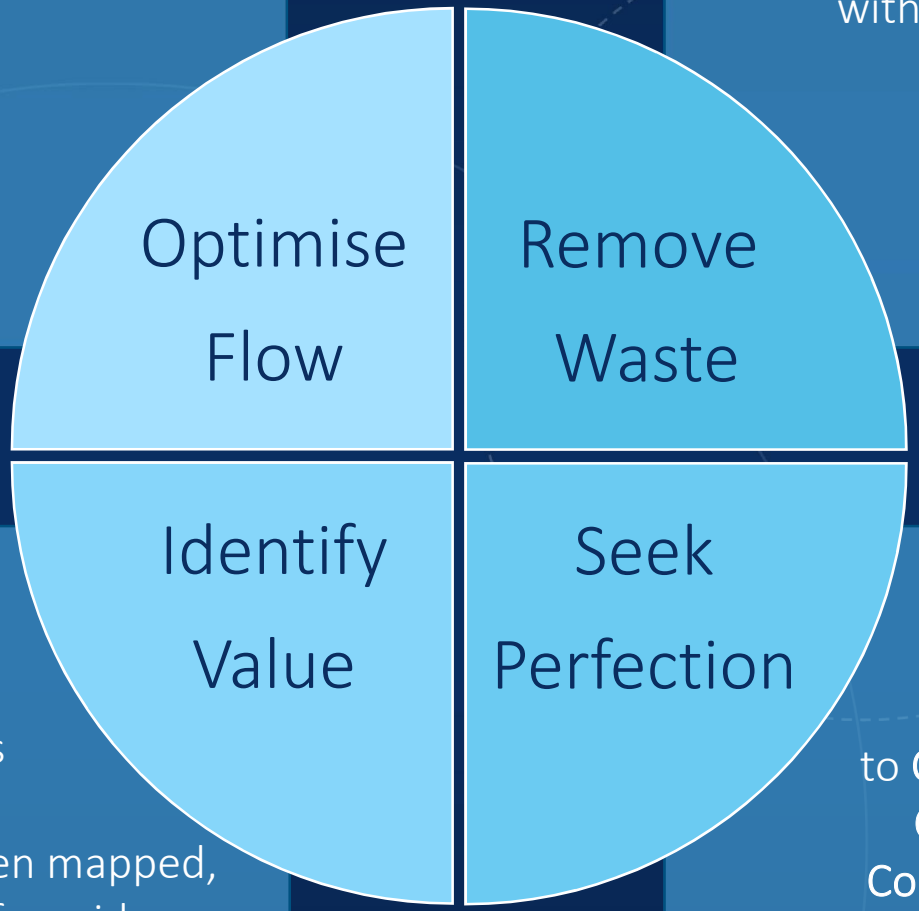
Practices

Tools

PM

PMs help the team to ensure that tasks flow and processes run smoothly.

Lean PMs ensure that activities within the organization flow properly, without interruption, delays or bottlenecks.



Find out what your product will do & how much customers will pay for it.
Value streams are then mapped, following every step from idea generation to production.

Lean management demands a dedication to **Continuous Monitoring, Continuous Feedback & Continuous Improvement**.
Help the team seek perfection

DevOps Tips

- Start with small projects and set your team up to succeed
- Focus on the Minimum Viable Product (MVP)
- Collaboration, communication, the removal of silos
- Enable productivity by reducing overhead
- Allow your teams to pick the right tools for the job
- Look at Kanban vs Scrum for CI/CD/DevOps

PM for DevOps

DevOps Tips

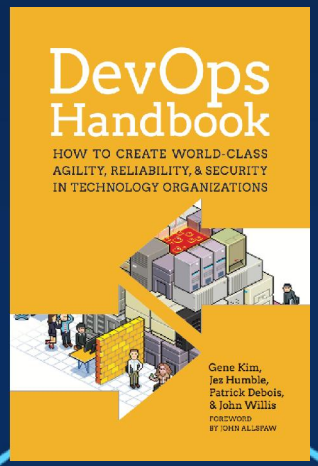
- Monitoring & Feedback are critical
- Emphasise creating real-time project visibility
- Focus on flow and integration
- Eliminate waste wherever possible & reduce handoffs
- Help your team to manage change collaboratively
- Budgeting challenge of ongoing Programme vs Project

PM for DevOps

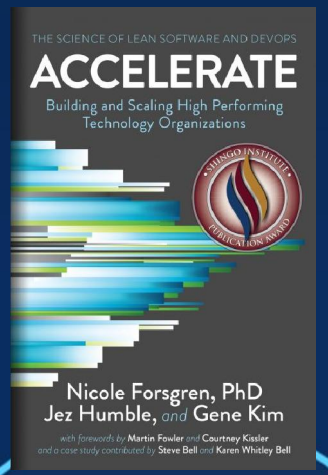
Next Steps

The **Phoenix Project** is a must read for anyone seeking to understand DevOps. The **DevOps Handbook** and **Accelerate** are great follow ups for anyone working in a DevOps team

The Phoenix Project
By Gene Kim,
George Spafford
and
Kevin Behr

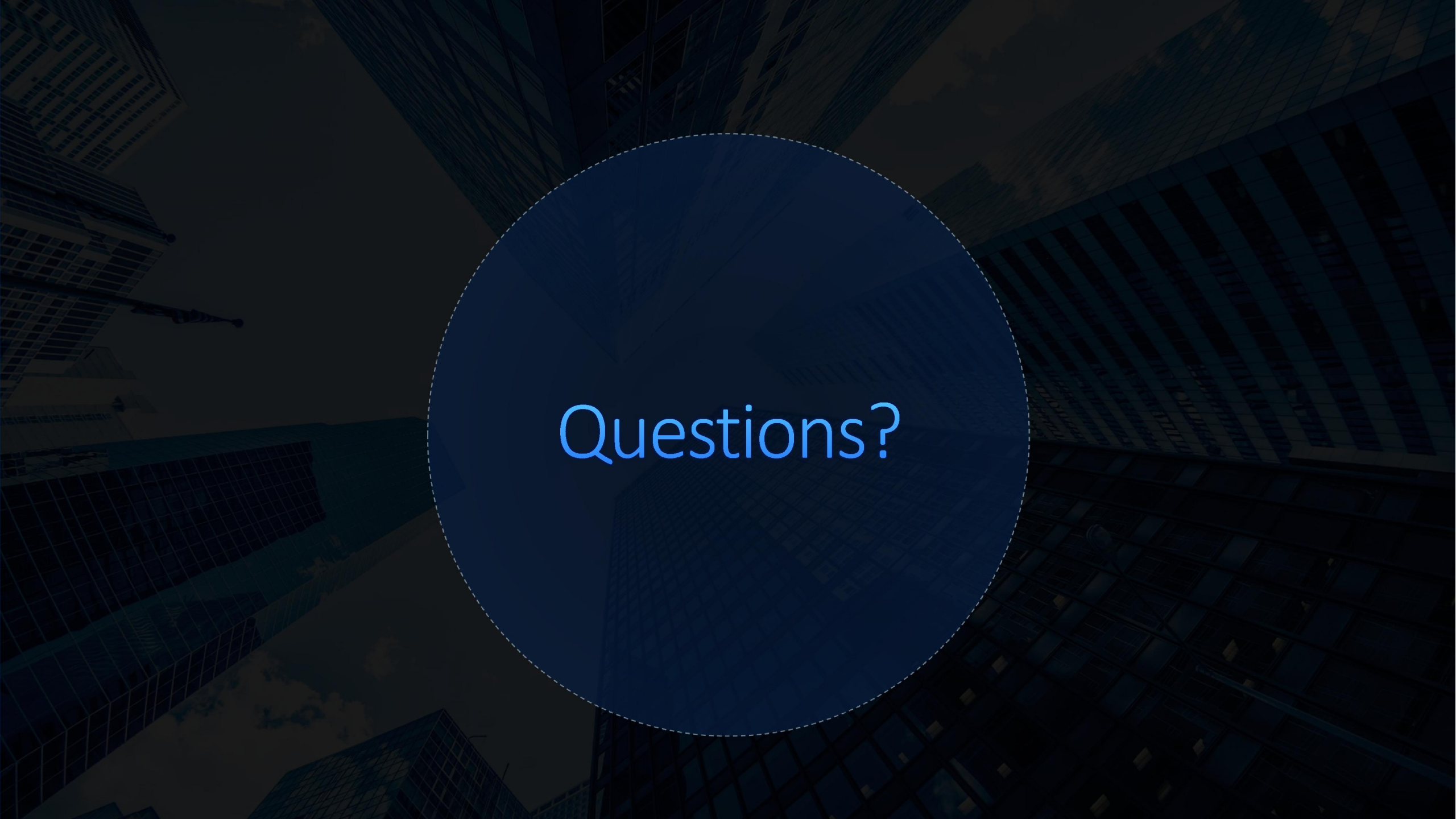


The DevOps Handbook
By Gene Kim
Jez Humble
John Willis,
and
Patrick Debois



Accelerate
By Gene Kim,
Jez Humble
and
Nicole Forsgren

Wrapping it up


Questions?