DevOps and ITSM Training

Course Book

DevOps and ITSM Training

Copyright and Disclaimer

DevOps and ITSM Training | r1.0.0

Copyright

Disclaimer

Information provided about the course, modules, topics and any services for courses including simulations or handouts, are an expression of intent only and are not to be taken as a firm offer or undertaking. The Publisher reserves the right to discontinue or vary or maintain such course, modules, topics, or services at any time without notice and to impose limitations on enrolment in any course.

The course materials provided may have hypertext links to a number of other web sites as a reference to users. This service does not mean that the publisher endorses those sites or material on them in any way. The publisher is not responsible for the use of a hypertext link for which a commercial charge applies. Individual users are responsible for any charges that their use may incur.

The information in this course is written using a blend of British and American English. Although every effort has been made regarding the usage of correct spelling, punctuation, vocabulary, and grammar with regard to the Standard English, ITpreneurs accepts no responsibility for any loss or inconvenience caused due to the regional differences in the usage of the English language.

Contents

My Notes

Acknowledgements

We would like to sincerely thank the experts who have contributed to the development of the DevOps and ITSM Training.

Lead Author

Marcel Foederer, Master Trainer, ITpreneurs Technology Private Limited, Nederland

As an IT Service Management trainer, consultant and line manager with over twenty-five years of experience in IT, Marcel has performed strategic and tactical assignments in a wide variety of areas. He also excels in DevOps in combination with ITIL. His experience includes project and program management including process design, product management, requirements analysis, and training delivery related to the IT Service Management international best practice in both the private and public sectors on a global scale. His area of consulting expertise is in advising organizations on IT Service Management, based on ITIL® (IT Infrastructure Library) best practices, and in the management of these initiatives to improve organizational and operational efficiencies and service delivery quality. He is an experienced facilitator, trainer, and lecturer. He is committed to the successful delivery of total solutions to his client base, achieved through respect for the management of change issues involved in the resulting integration of people, process, and technology.

Harriette Blaauboer, Senior Consultant, Quint Wellington Redwood

Harriette has extensive experience as the Change Manager and Project Manager with the (process) design, management, and development of IT Governance, and performance of IT services. As the Service Delivery Manager, she had the central role between contract management, information managers, project leaders, technicians, and suppliers, where it unites the different interests to an effective solution for users. She has extensive experience in change management, processes and governance, and management of internal or outsourced services. She is an APMG accredited Lean IT trainer. Currently, within Quint Wellington Redwood (Quint), she is involved in product development on Agile Governance and DevOps transformation and the development of course materials. Quint is an independent consulting firm with a portfolio focused on providing consultancy and training aimed at optimizing IT-intensive processes.

Claudine Koers, Senior Consultant, Quint Wellington Redwood

Claudine has a Master degree in Management and Organization at the Rijks University of Groningen with Enterprise Strategy and Management of Changes as a specialization. As a consultant, coach, or project leader, she has been involved in many projects around the world in IT Best Practices related to improvements in IT Governance, IT Service Management, Agile, and DevOps amongst others. She is a Service Management Expert, with more than 20 certificates in IT Best Practices, opinion leader, writes articles, gives lectures at universities, and develops course material. She strongly believes in "Continuous learning is key because learning is growing: being meaningful, this becomes valuable and profitable not only for an individual but also for a team, a department or an organization". Claudine is responsible for the IT Service Management portfolio at Quint Wellington Redwood (Quint), an independent consulting firm with a portfolio focused on providing consultancy and training aimed at optimizing IT-intensive processes.

Rik Farenhorst, Business Unit Manager at Xebia

Rik has a PhD in software engineering and more than 10 years of experience in the business-IT domain. He has worked as Enterprise/IT Architect, Consultant, Trainer and Coach, and has later grown into sr. management positions. He guides organizations and individuals in raising their bar by making IT simpler, better, and of higher quality. Rik works as Business Unit Manager at Xebia, a specialized international IT consultancy, solutions and training company specialized in high-performance IT, digital transformation, and thought leader in DevOps. Rik is a member of the editorial board of the DevOps Agile Skills Association (DASA).

DevOps and ITSM Training - Introduction

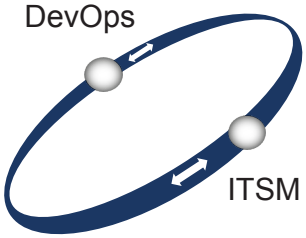
LET'S GET TO KNOW EACH OTHER

Introduce yourself in the following format:

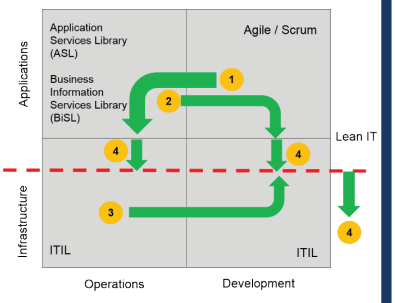
- Name
- Company
- Role and background
- Familiarity with DevOps concepts and their practice
- Experience in application development, infrastructure development, and/or operations
- Expectations from this course

TOPICS COVERED

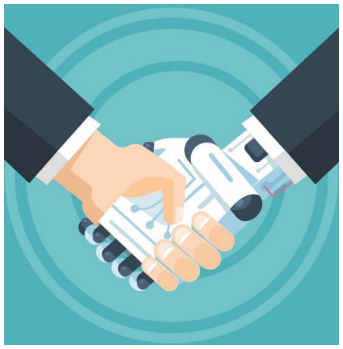
Training Introduction



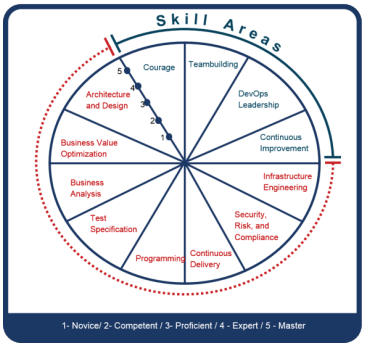
DevOps and ITSM (ITIL®)



DevOps Journey Examples



Contribution of ITIL® and DevOps



DevOps Agile Skills Association (DASA)

TRAINING INTRODUCTION

Overview

This 1-day training provides you with the basic knowledge required for the effective utilization of ITSM skill sets in any DevOps organization. The training focuses on the essential knowledge of the two different worlds, ITSM and DevOps, and how they complement each other. It helps you know how DevOps movement fits perfectly with ITSM. The various group discussions and the case study are the primary ingredients of this course that help you understand a DevOps movement in an organization.

DevOps is gaining momentum, therefore, think of this course before looking elsewhere for the DevOps movement in your organization. However, it does not provide any one-size-fits-all solutions or formulae that ensure a successful DevOps implementation.

Training Objectives

At the end of this training, you will be able to:

- Relate and complement DevOps and ITSM.
- Describe DevOps.
- Discuss the value of ITSM in a DevOps environment.
- Explain how to utilize ITSM processes in DevOps teams.

Course Agenda

Topic	Subject	Start	End	Total Time (in hours)
01	Training Introduction	09:00	09:30	00:30
02	DevOps and ITSM (ITIL®)	09:30	10:00	00:30
	Break	*10:00*	*10:15*	*00:15*
02	DevOps and ITSM (ITIL®)	10:15	11:15	01:00
03	DevOps Journey Examples	11:15	11:30	00:15
	Activity Time: DevOps Impact on EJ Airways	11:30	12:00	00:30
	Lunch	*12:00*	*01:00*	*01:00*
04	Contribution of ITIL® and DevOps	01:00	01:30	00:30
	Activity Time: ITSM Providing a Solid Base for DevOps in Operation	01:30	03:30	02:00
	Break	*03:30*	*03:45*	*00:15*
	Activity Time: ITSM Providing a Solid Base for DevOps in Design and Transition (Optional)			
05	DevOps Agile Skills Association (DASA)	03:45	04:15	00:30
06	Training Closure (Q&A)	04:15	05:00	00:45
	Total (Less Lunch and Break)			**08:00**
	Total			**06:30**

Type of Activities

Group Discussions

The training contains group discussions with the intent to enhance participants' understanding, add context to the content, broaden participants perspective, reinforce knowledge, and build confidence.

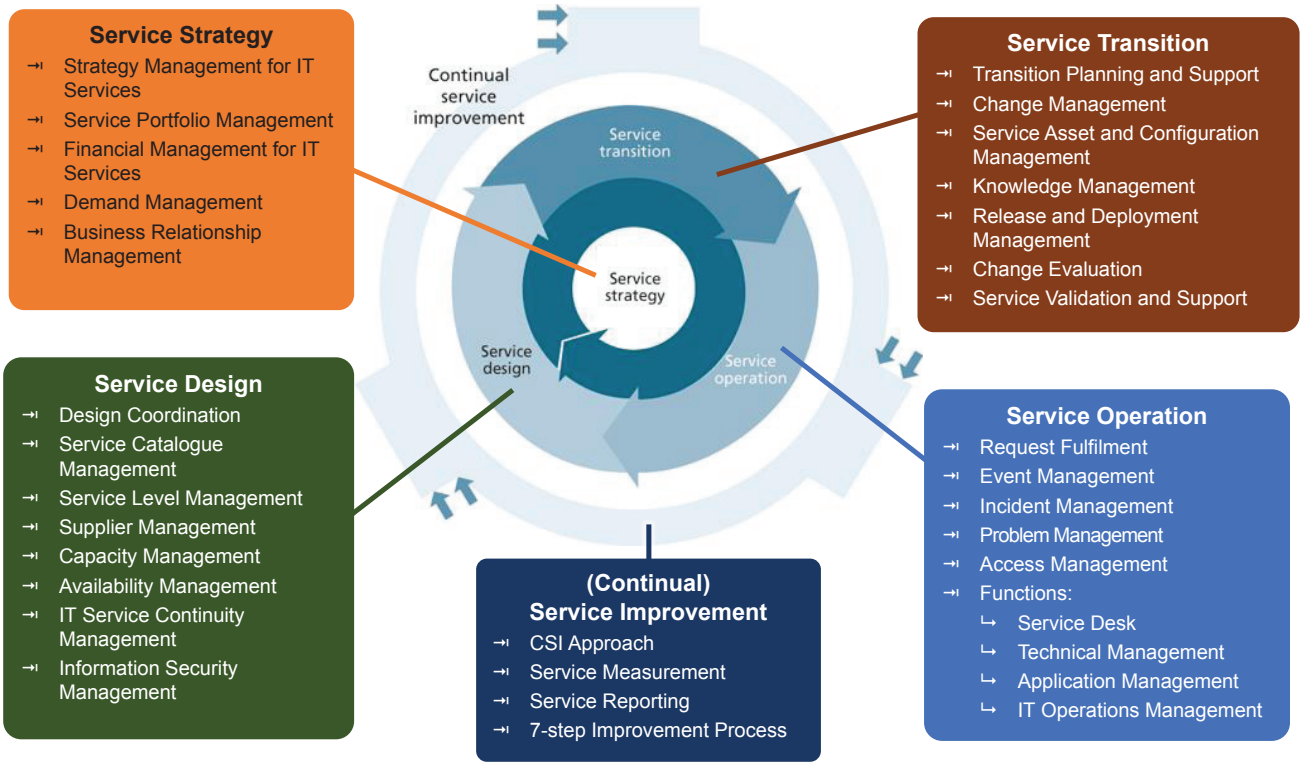
By interacting among themselves and responding to the varying viewpoints, participants tend to learn continually. These discussions allow participants to come across the thoughts of their peers, which help them know about each other's past experience, perspectives, and opinions in the context of the topic of discussion.

Group Activity

Write down your expectations from this training on a sticky and attach it to a wall.

DEVOPS AND ITSM (ITIL®)

ITIL Refresh



Information Technology Infrastructure Library (ITIL) is a set of practices for IT Service Management (ITSM) that focuses on aligning IT services with the needs of business. ITIL is published as a series of five topics that cover the five different ITSM lifecycle stages. ITIL describes processes, activities, roles, Key Performance Indicators (KPIs), and Critical Success Factors (CSFs) for establishing integration with the organization's strategy, delivering value, and maintaining this value (the product).

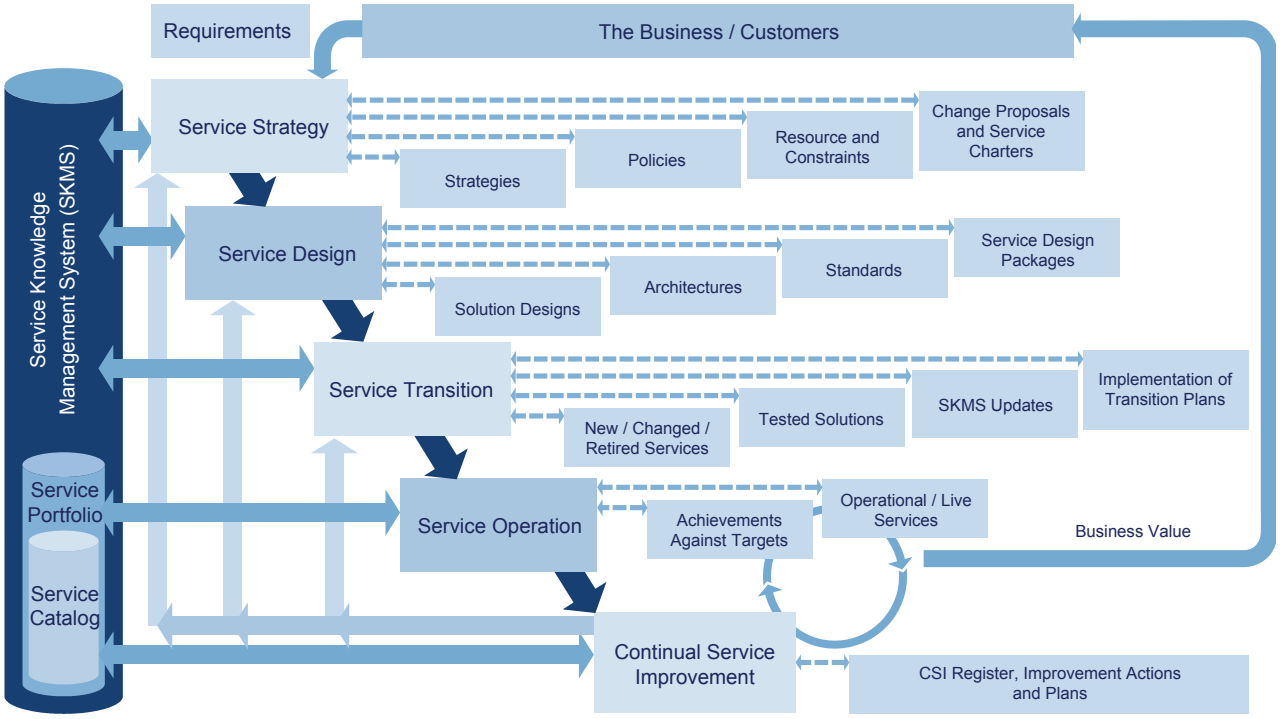
The ITIL Service Lifecycle can be viewed as a phased lifecycle, where phases are:

- **Service Strategy**: Define the strategy for the ITSM.
- **Service Design**: Design the services to support the strategy.
- **Service Transition**: Implement the services to meet the designed requirements.
- **Service Operation**: Support the services, managing the operational activities.
- **Continual Service Improvement**: The interaction between phases are managed through the Continual Service Improvement (CSI) approach, which is responsible for measuring and improving service and process maturity level.

After the completion of all phases, a lifecycle is concluded, and another lifecycle begins.

- At the beginning (**Service Strategy phase**), the IT Service Provider starts to set strategy by managing the business requirements (Demand Management), translating it into a strategy to deliver service (Service Strategy), validating the sustainable costs (Financial Management), and introducing the service in the service portfolio (Service Portfolio Management). In this phase, IT is required to use resources (costs) in consultancy projects at a strategic level. At this phase, IT does not provide a tangible product value to the business which is related to this specific phase.
- When a strategy is complete, IT starts to design the service (**Service Design phase**), by setting the service level requirements for the service (Service Level Management), analyzing the required availability and capacity (Availability and Capacity Management), selecting the suppliers which will support service (Supplier Management), defining the adequate service continuity arrangements (Service Continuity Management), validating and designing the security requirements and introducing the service in the service catalogue (Service Catalogue Management).
- In the third phase (**Service Transition phase**), the service is ready to be implemented in the live environment. The Service Provider defines the transition plan (Transition Planning and Support) and assesses, approves, implements, and plans the change (Change Management). After the change implementation, the service is tested (Service Validation and Support) in a “pre-live” environment. If the test is successful, the service is documented (Knowledge Management) and its components are introduced in the Asset and Configuration Databases (Service Asset and Configuration Management). The last activity is to release the service into the live environment (Release and Deployment Management) and after the “go-live”, a post-implementation review will be made (Change Evaluation).
- In the fourth phase (**Service Operation phase**), the service starts to be managed and supported in order to reach the agreed service level by managing the users’ support requests (Request Fulfillment), monitoring the service event and alerts (Event Management), restoring the service after disruptions (Incident Management), avoiding the incident causes and reduce the incident duration (Problem Management), managing in a secure manner the utilization of services (Access Management), maintaining the software (Application Management), executing the day-to-day activities (IT Operations Management), and supporting the infrastructure (Technical Management).
- The **Continual Service Improvement phase** is involved during all phases of the service lifecycle. It is responsible to measure the service and the processes (Service Measurement), and to document the results (Service Reporting) in order to improve the service quality and the processes maturity (7-step Improvement Process).
- These improvements will be implemented in the next period of service lifecycle, starting again with Service Strategy. After Service Design and Service Transition, the Service Operation phase continues to manage operations during all service periods.

Overview of ITIL® Key Lifecycle Phases and Outputs



Integration Across the Service Lifecycle

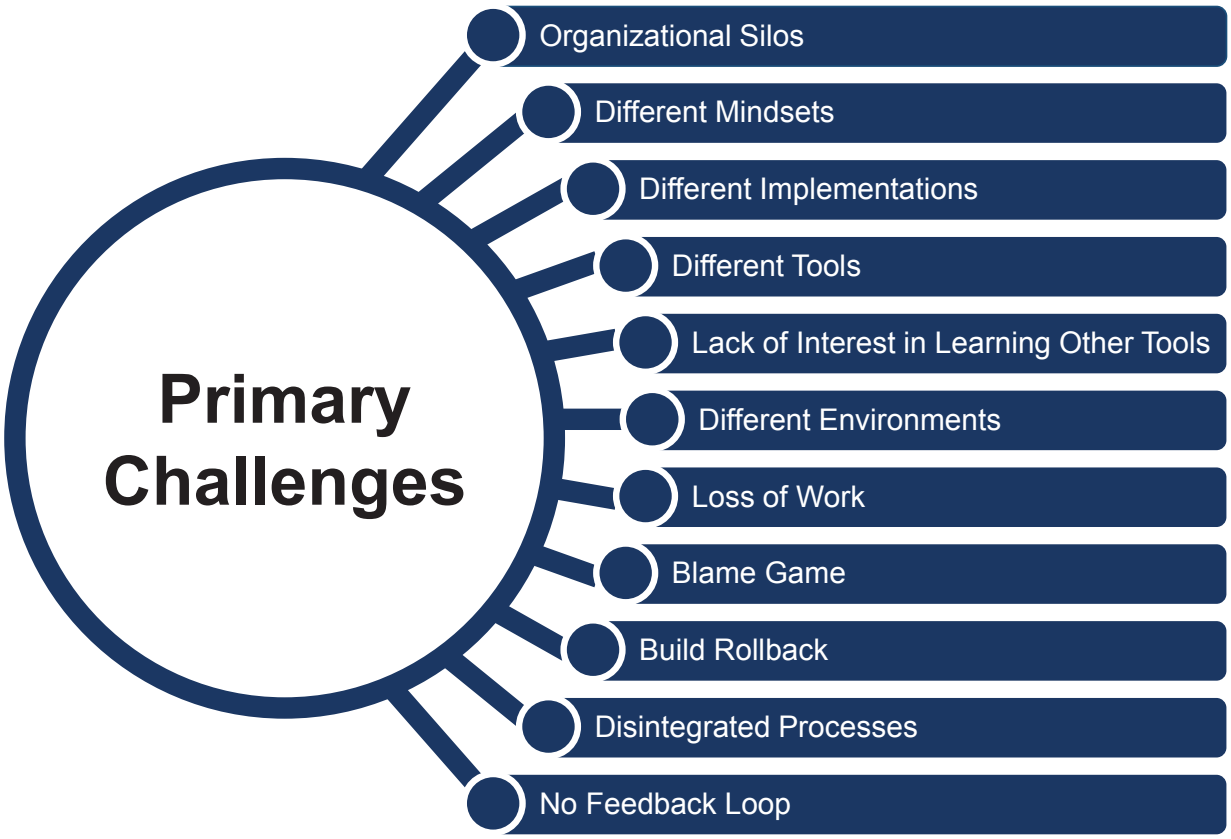
What is DevOps?

DevOps is a **CULTURAL** and **OPERATIONAL** model that fosters **COLLABORATION** to **ENABLE** high-performance IT to **ACHIEVE** business goals.

To survive, organizations need to adopt DevOps practices to create a culture of high-performance IT. The cultural and operational model underlying the core vision and principles of DevOps is the key to arriving at this high-performance level. The focus on culture means people, like you, need to help investigate what works and what does not. Therefore, people need to bring the right mentality and drive to continuously improve the state of the practice, together with colleagues and peers. Unfortunately, there is no silver bullet for DevOps, but the core principles of DevOps can help individuals, teams and IT organizations get started in the right direction.

Challenges Leading to DevOps

Some of the challenges with the traditional teams of Development and Operations are:



- **Organizational Silos**: In many IT organizations, the Development and Operations teams tend to work in isolation from one another to avoid confrontations until the critical moments of the delivery of IT products and services. The stress of these moments tends to lead to irritation rather than understanding.
- **Different Mindsets**: The Development team aims to incorporate new techniques or features to do their work efficiently. On the other hand, these changes to the IT service and the underlying components result in instability and make the work of the Operations team difficult. Therefore, the Operations team sees these changes as a challenge in maintaining the integrity of the Production environment.
- **Different Implementations**: The different implementations to perform the same work by the two teams result in incompatibility and lead to various bugs in the QA and the Production environments.
- **Different Tools**: The different tools used by the two teams lead to various errors and bugs in the Production environment. As an example, Development might deploy to a Test environment using a dependency management tool, while Operations might use a self-made script for the process.
- **Lack of Interest in Learning Other Tools**: The teams do not want to learn a new tool. They consider their style of working the best along with the tools that they are using.
- **Different Environments**: The different environments, such as Development, Test, and Production, are one of the largest sources of errors and bugs raised by the different teams.

- **Loss of Work**: The various errors and bugs result in loss of valuable efforts.
- **Blame Game**: For each error, bug, or resulting incident, each team tries to ensure they are not identified as the cause of the issue. Therefore, they tend to pass on the blame for delayed delivery or errors to each other, leading to further irritation, lack of understanding, and intensifying of the wall of confusion.
- **Build Rollback**: Many times the build rollbacks are required as a result of a variety of causes, such as incorrect client requirements, an incorrect database in the QA or Production environment, incompatible tools and others.
- **Disintegrated Processes**: The traditional structure of the various processes of the two teams are generally based on different frameworks, such as ITIL, ASL, COBIT, and Scrum. Therefore, development processes do not integrate well with operations processes leading to disjointed processes and different vocabulary, again intensifying the wall of confusion.
- **No Feedback Loop**: In many IT organizations, there is a feedback loop. It is often negative and strongly affected by the blame game. Lack of a feedback loop that is positive and continuous between the development and operational processes causes the lack of understanding to increase.

The problems enhance the wall of confusion to an extent that a truly concerted effort is required to bring it down. DevOps encompasses a series of ideas, concepts, and concrete actions that focus on removing these problems.

A Brief History of DevOps

The problems due to the wall of confusion have been encountered by many IT organizations. DevOps took birth due to the wall of confusion between Development and Operations.

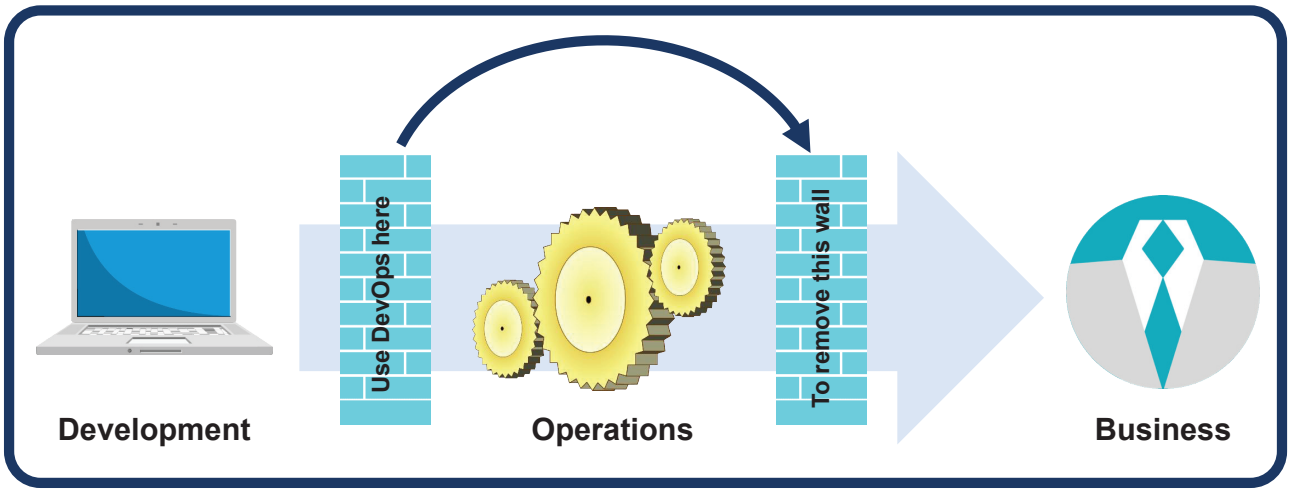
According to Damon Edwards, co-founder of DTO Solutions, the DevOps movement was germinated in Belgium back in 2007. Patrick Debois, an IT consultant, was frustrated by struggle, lack of communication, and disconnection between Development and Operations departments. He found himself straddling between the two teams while working on a huge data center migration project for the Belgium Government Ministry. Patrick was performing the dual role of firefighting in the unpredictable world of IT operations and working on the Agile development. He was confident that there was a better way of working which would allow bridging the substantial gap between the two teams.

At the Agile 2008 Conference in Toronto, Canada, Andrew Shafer (a partner at Reductive Labs) proposed a discussion topic for an ad hoc session entitled Agile Infrastructure. However, the session got canceled due to lack of interest and feedback. Patrick Debois was only to follow the discussion and eventually tracked down Andrew at the conference, and then they had an in-depth discussion about their mutual frustrations. The discussion gave rise to the Agile System Administration group on Google Groups by Andrew Shafer and Patrick Debois. The group was not overly popular, but it leads to some fascinating discussion. Moving forward to Velocity Conference, San Jose, the famous talk was given by John Allspaw and Paul Hammond of Flickr in 2009.

DevOpsDays was born, and the first event was conducted from October 30th to 31st, 2009, Ghent, Belgium. The event attracted administrators, developers, and managers from all around the world. The success of the event also inspired other DevOpsDays events in different countries. These events acted as a catalyst for the conversation and a grassroots movement. Most vendors, analysts, and traditional enterprise IT shops ignored the movement, but it ignited the passions of those who were concerned. As a result, the movement began to flow and spawned tools, such as Vagrant, Puppet, Chef, and FPM. The movement also started running circles around legacy enterprise IT systems.

How DevOps make it easy for a business to work with IT?

Traditional Development and Operations teams create the wall of confusion that limits business performance. Using DevOps, the teams can enhance the performance, as depicted in the following figure.



How do break down the wall?

It requires breaking down the internal wall that pushes the customers to talk to different people. With DevOps, they can talk to the person or the team responsible for the complete health and development of the IT service supporting their business process.

Does it mean creating many smaller IT departments?

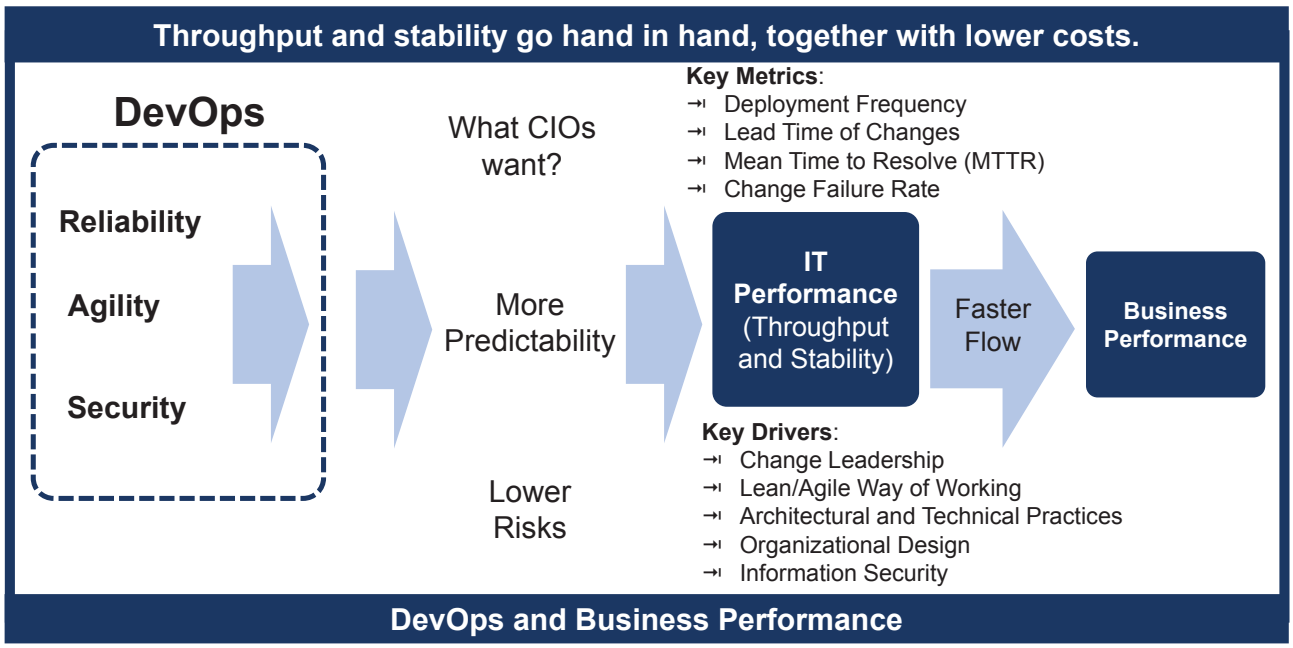
No. A great value lies in reuse, collaboration, and standardization.

Does it mean creating new silos?

Yes, but these silos do not limit performance towards the customers. They represent the problem that Development and Operations should be managing, the optimization and standardization of IT services.

The role of architecture is vital. Architecture guidelines are used better in the DevOps situation than the traditional business. Therefore, DevOps makes it easy for customers to do business with the organizations as IT.

How DevOps drives business performance?



In the well known and highly regarded 2014 State-of-DevOps Report by Puppet Labs, IT performance is measured in terms of throughput and stability. The two attributes are essential in achieving high-performing IT.

Performance metrics for throughput:

- Deployment frequency
- Lead Time for Changes

Performance metrics for stability:

- MTTR
- Change Fail Rate

Performance Predictors of IT:

- Performance Predictors of IT
- Peer-reviewed change approval process
- Version control everything
- Proactive monitoring
- High-trust organizational culture
- Dev vs. Ops (win-win relationship)

Performance Practices for throughput and stability:

To increase the IT performance on throughput and stability, the following practices were found to be effective:

- Putting code, app configurations, and system configurations in a version control system
- Getting failure alerts from logging and monitoring systems
- Merging developers' code into trunk on a daily basis
- Allowing Development and Operations to interact with each other as it generally results in a win-win situation
- Encouraging developers to break up large features into small, incremental changes

Why DevOps? (The Benefits)

Source: Puppet Labs – State of DevOps 2016

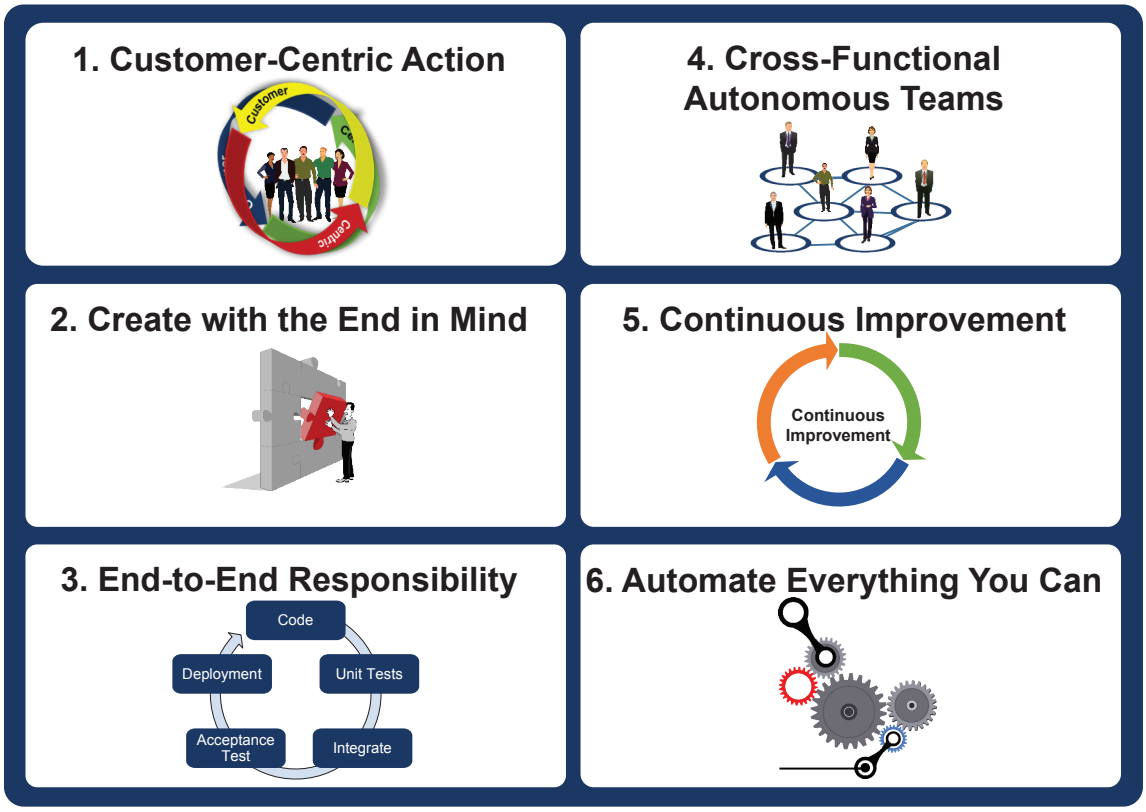
- **Better due to:**
  - Automation results in predictable, standardized, reliable, and repeatable outcomes.
  - Automation eliminates manual task execution errors.
  - Automation (with test automation) results in faster feedback loops. As a result, defects are detected early in the process.
  - Automation enables measurement driven evaluation of the delivered software features.
- **Faster due to:**
  - Automated task execution does not depend on the availability of humans, so the wait time is eliminated.
  - Automated task execution requires no human-to-machine interaction time.
  - Automated task execution reduces the need for (manual) meta tasks (like a review of an installation document).

- **Cheaper/Smarter due to:**
  - Automated task execution is more reliable than manual task execution. Manual execution errors are expensive and might not be detected immediately.
  - Automated task execution is more repeatable than manual task execution.
  - Automated task execution requires no human effort (which is more expensive than machine time).
  - Automated task execution requires standardization, based on minimal required variations. Every variation requires specific procedures and maintenance.

Reference Reading:

PuppetLabs, 2014-state-of-devops-report.pdf, *https://puppet.com/resources/white-paper/2014-state-of-devops-report*

DevOps Core Principles



Many definitions of DevOps exist. Many of these adequately explain one or more aspects important for finding flow in the delivery of IT services. Instead of trying to create a comprehensive definition on your own, refer to the following six principles when adopting or migrating to a DevOps way of working:

1. **Customer-Centric Action**: It is imperative nowadays to have short feedback loops with real customers and end users. Therefore, all activities involved in building IT products and services should revolve around customers.
2. **Create with the End in Mind**: The principle focuses on understanding the real needs of customers and working towards creating products and services that solve the problems. In other words, it considers taking a holistic view of both the creation and use of the IT product/service.

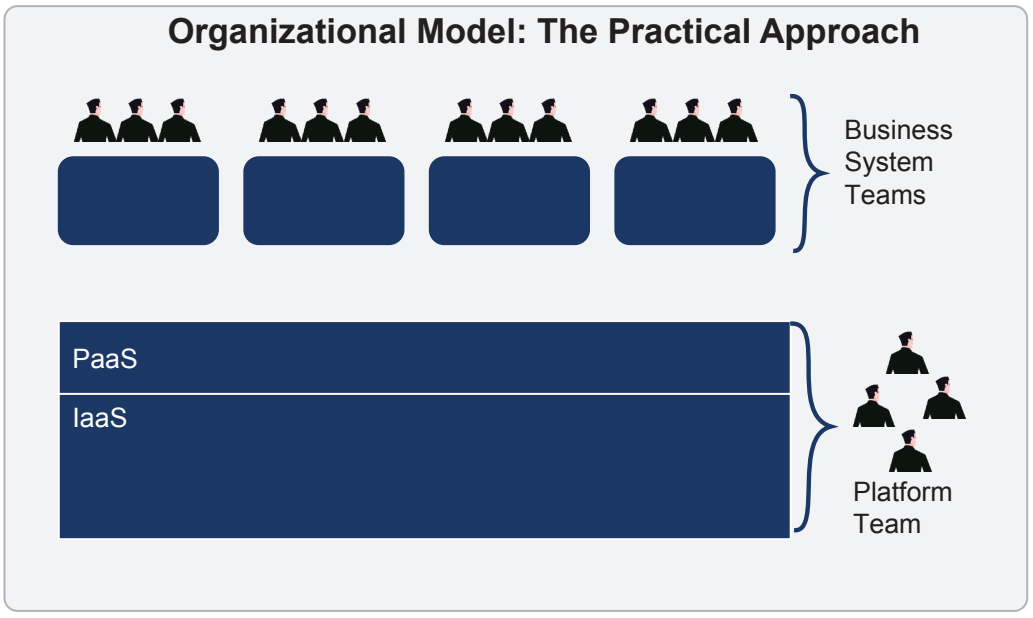
3. **End-to-End Responsibility**: In a DevOps organization, teams are vertically organized to enable team members to be fully accountable for the products and services they deliver. End-to-end responsibility means the team holds itself accountable for the quality and quantity of services it provides to its customers.
4. **Cross-Functional Autonomous Teams**: In product organizations with vertical, fully responsible teams, the teams need to be fully autonomous throughout the entire lifecycle of the product. Therefore, the teams should possess all the necessary expertise to take on the end-to-end responsibility.
5. **Continuous Improvement**: In a DevOps culture, a strong focus is put on continuous improvement to enhance the products/services offered to customers. Some of the improvement activities include minimizing waste and optimizing speed, costs, and ease of delivery.
6. **Automate Everything You Can**: Automation is synonymous with the drive to renew the way through which the team delivers its services. Extensive automation means having a deep understanding of the processes needed to develop and deliver the IT services.

Reference Reading:

DASA Whitepaper - Embracing Digital Disruption by Adopting DevOps Practices (*goo.gl/LfUZX6*)

Defining a DevOps Team

The current preferred way of organizing DevOps teams is to enable autonomous Business System teams that can “land” their application and infrastructure code on a platform that is maintained by a Platform team. However, such a self-service platform is a product in itself and needs maintenance, innovation, and ownership.

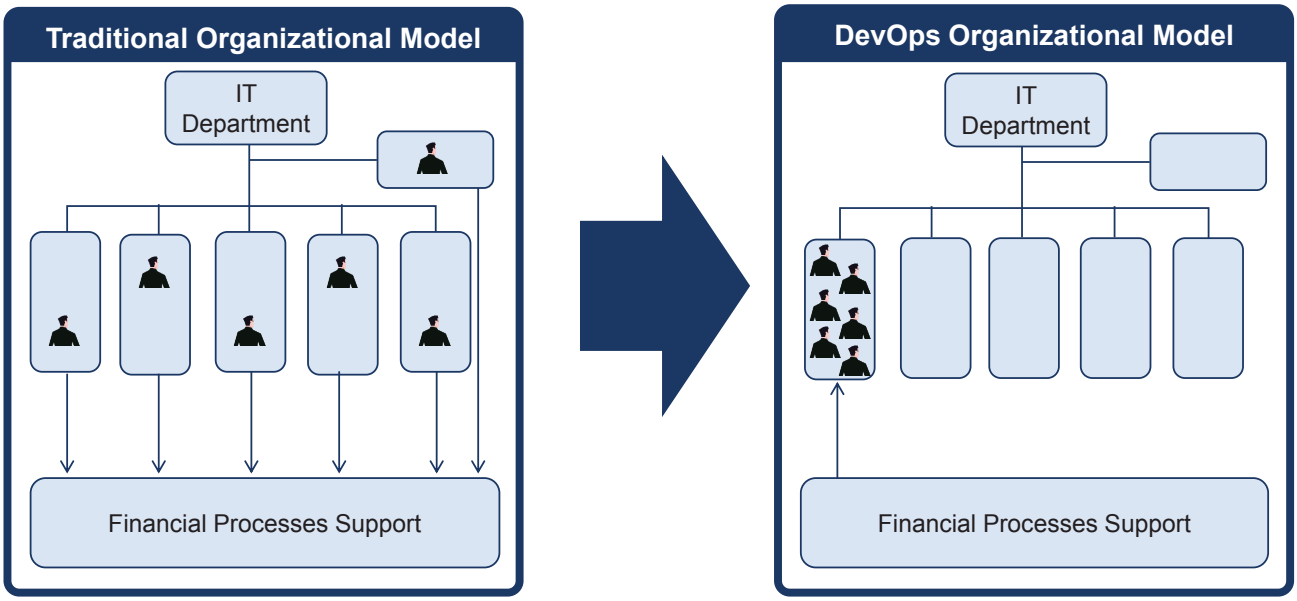


In the model, as depicted in the preceding figure, the infrastructure reuse benefits are largest, without impeding the speed and autonomy of the various Business System teams. It also enables the Business System team to take full responsibility for their service during its complete lifecycle.

Business System teams manage end users (for example, customer), services, and products (for example, applications). On the other hand, Platform teams manage platform products used by Business System teams. Platform teams do not take over the responsibility of the service offered by Business System teams. They offer platform (or Infrastructure) services through self-service and automation to make the Business System teams work more effectively. This way prevents Business System teams from waiting for their request to be dealt with by the Platform team. However, both teams need to work closely together to ensure the services provided by the Platform team are the correct ones.

DevOps Organization

- The organizational model is the business structure or the way a business organizes itself in departments, units, and teams.
- The operational model is the way a business or an organization operates to get the work done.

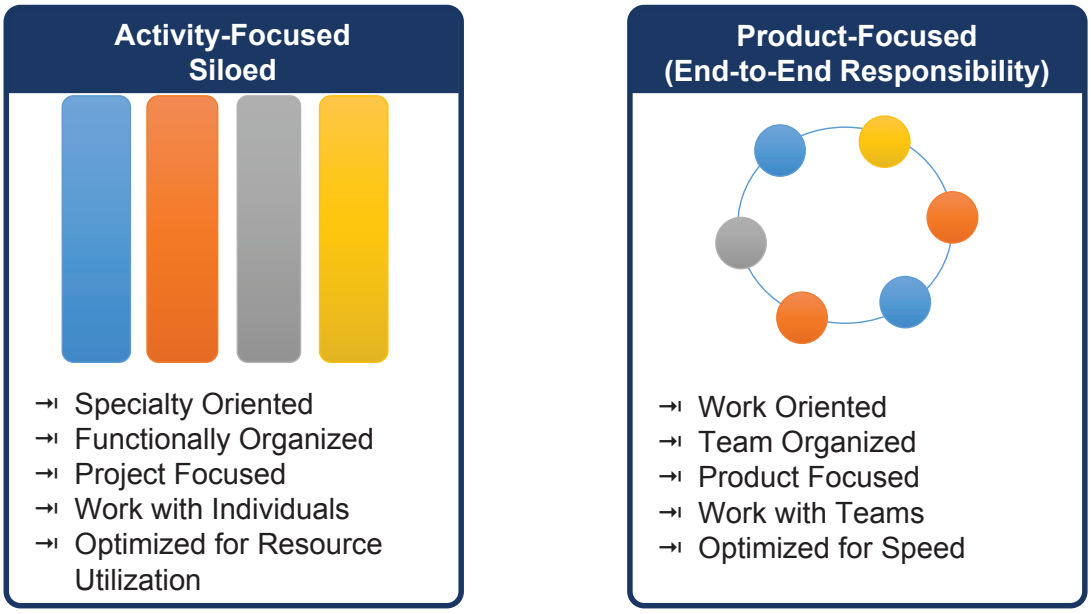


An organizational model describes how the organization is structured. In an organizational model, people spread across all over an organization in technically-oriented silos, which result in communication problems. DevOps organizations aim to work with the organizational model that facilitates collaboration between people to provide the customer with the best products and services. In traditional organizational models used within IT, there is a substantial need for coordination overhead in the form of Service Delivery Managers and Project Managers responsible for bringing people together to create value for the customer. On the other hand, the DevOps organizational model focuses on a single (mostly colocated) team doing the work, hence, reducing communication and coordination issues.

The operational model describes how work gets done. In traditional IT organizations, the operational model is dependent on coordination roles that ensure the people in technical silos are brought together to deliver a service. The operational model of a DevOps IT organization focuses on delivering work through autonomous teams having the required knowledge to develop, deliver, and maintain the IT service. These are two distinctly different operational models.

The organizational model and the operational model are closely related but distinct entities. Both need to be addressed when organizing DevOps teams. For example, it is conceivable to describe a DevOps environment in which the Project Manager has an important role in realizing the IT service (not a preferred solution). It means the organizational model can have autonomous teams, but the operational model is project-based.

Activity-Focused vs. Product-Focused



The characteristics of an activity-focused organization are:

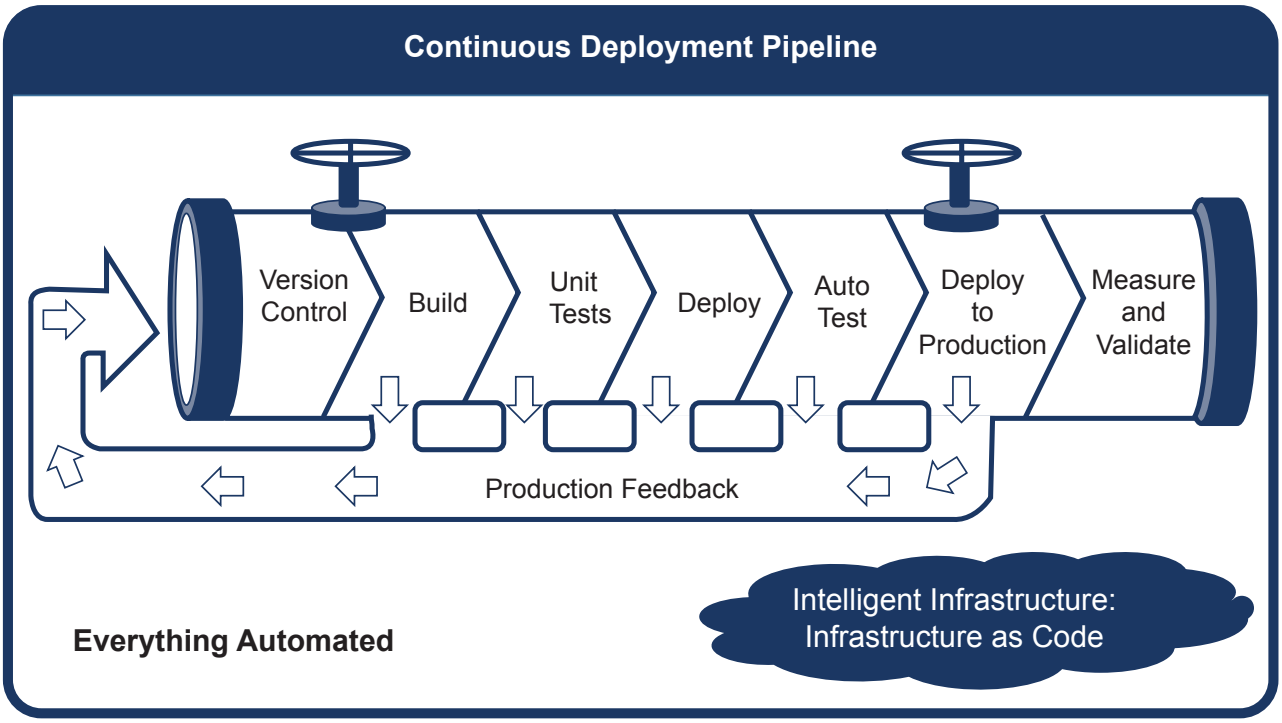
- Resources are specialty oriented and perform a specific task in a chain of events at a time, such as updating a database.
- Resources are functionally organized implying they are added to specific resource pools reflecting specialisms, for example, a pool of database administrators.
- Resources work on projects with a beginning and an end, and they can be assigned to multiple projects at once.
- The organization works with individuals who are seen to be interchangeable.
- This type of organization is optimized for resource utilization.

The characteristics of a product-focused organization are:

- Resources are oriented to deliver work that requires multiple skills, such as a developer who creates test fixtures in code also knows how to automate a delivery process.
- The resources are organized in teams who are responsible for their product delivery and/or maintenance.
- The team relates to the product throughout its entire lifecycle. The applicable ideology being “you build it, you run it.”
- The team is associated with the product throughout its entire lifecycle. The applicable ideology being “you build it, you run it.”
- This type of organization is optimized for speed or lowering cycle-time for a product.

Siloed organizations have resources grouped on specialisms who can be optimized from a resource efficiency perspective but not from a process throughput (or flow) perspective. Product-focused (or Agile) teams are multidisciplinary in nature where the entire team has all disciplines to bring a product to production and keep it working in production. All disciplines working in a team with the focus on the end product helps to reduce expensive handover moments and further improve the process throughput optimization. Remember, for a DevOps organization, it is all about speed and getting features to the customer as soon as possible. Such a speedy delivery of products helps organizations to use valuable feedback loops to determine whether they are on the right track.

Automate Everything and Obtain Immediate Feedback

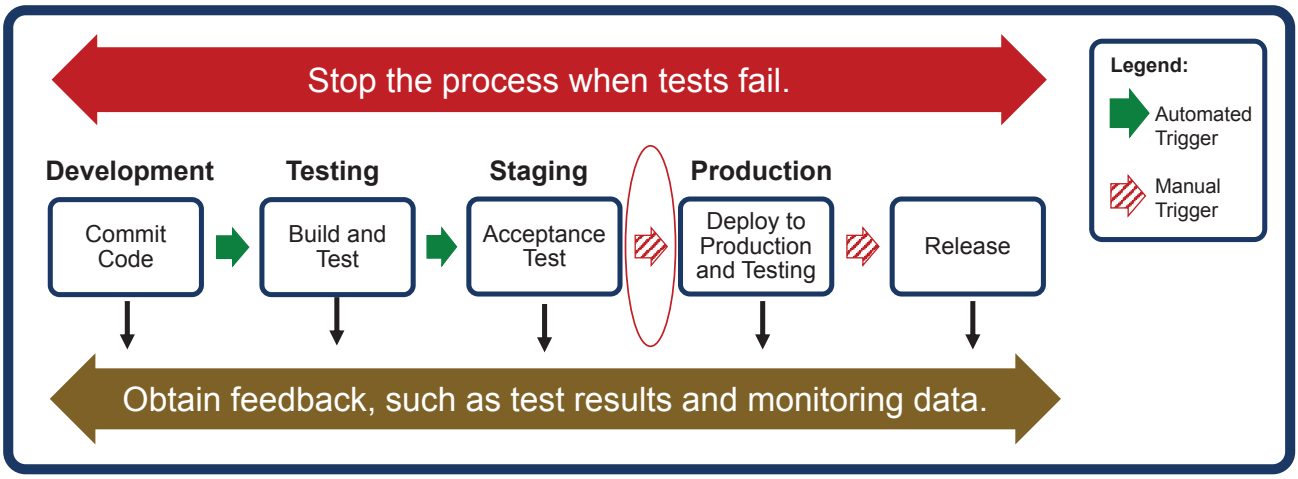


The continuous deployment pipeline breaks down the software delivery process into stages, as shown in the following figure. Each stage is aimed at verifying the quality of new features from a different perspective to validate the new functionality and prevent errors from affecting the users.

The figure shows the software deployment process as a pipeline flowing from left to right. During all the stages of the software development, the measurements provide feedback to the team. They can use the feedback to visualize the flow to everyone involved and identify bottlenecks and issues that need to be solved.

Continuous Delivery

Automated tests in production-like environments assure the code and the environment operate as designed and are always in a deployable state.



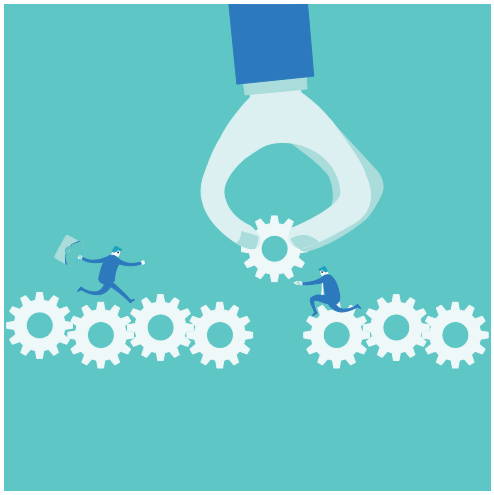
Deployment is the installation of a specified version of the software to a given environment, such as promoting a new build into production.

Some of the key concepts related to continuous delivery are:

- Continuous delivery is sometimes confused with continuous deployment. It is the ability to do frequent deployments, but you can choose not to do it due to several reasons, such as the business requirement of a slower rate of deployment (typically achieved by batching changes into releases).
- Whenever anyone makes a change that causes an automated test to fail and breaks the deployment pipeline, continuous delivery demands developers to stop the line (or process) immediately to bring the system back into a deployable state.
- Organizations can even choose to have the Continuous Integration and Deployment system that rejects any changes that take the code out of a deployable state, such as code or environment commits. Doing this continually and throughout a development project eliminates the common practice of having a separate integration and testing phase at the end of the project. The separate phase often gets compressed or skipped resulting in more technical debt and downward spiral.
- It emphasizes on monitoring throughout the deployment pipeline. It demands to monitor not only the production service but also the preproduction environments, such as Development, Testing, and Staging. Why? It minimizes the cost of correcting potential performance problems by detecting and correcting these long before the production.

Culture

High Trust and Respect



Culture is the sum total of behavior and mindset of an organization, supported and enhanced by the values and beliefs of that organization. As an organizational unit, each team develops its own culture with the ultimate goal to achieve high performance. This counts for each DevOps team.

Culture is shared and maintained through role-model behavior. This means that within and across teams people must exhibit consistent behavior to reinforce the strengths necessary for their teams to be successful from the cultural perspective.

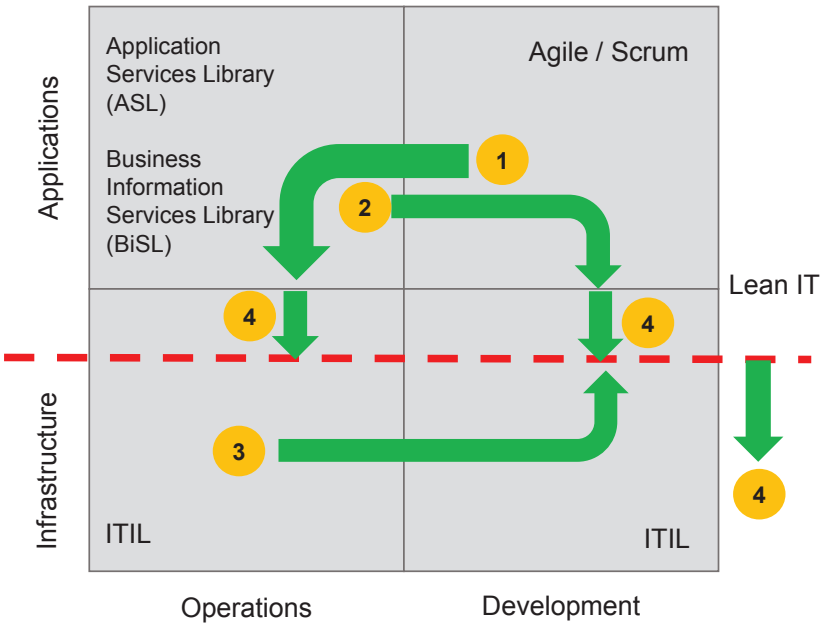
Reference Reading:

https://hbr.org/2013/05/what-is-organizational-culture

DEVOPS JOURNEY EXAMPLES

Different Starting Points

What is your starting point?

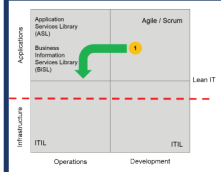


DevOps Journeys

1. From AppDev to Business System DevOps
2. From AppOps to Business System DevOps
3. From InfraOpps to Platform DevOps
4. From Business Systems DevOps to “Full Stack” DevOps

DevOps Journeys

1 From AppDev to Business System DevOps



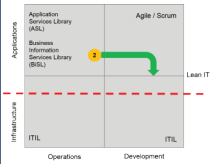
Starting Point of the Journey

- Focus on development, little attention to maintenance and support
- Speed is key
- Focus on automation of non-production environments, such as Continuous Integration (CI) and Test-driven Development (TDD)
- Mature development-oriented team

Key Actions

- Add maintenance focus, “You build it, you run it.”
- Encourage lifecycle responsibility
- Embed focus on quality
- Increase multidisciplinary of team through shared learning

2 From AppOps to Business System DevOps



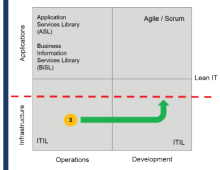
Starting Point of the Journey

- Primary steering based on processes
- Technical silo-based maintenance and support organization
- Focus on risk management regarding production environment
- Low speed of innovation

Key Actions

- Cluster services in customer-oriented domains
- Create multidisciplinary teams
- Strengthen development capability of teams
- Increase focus on speed of delivery

3 From InfraOps to Platform DevOps



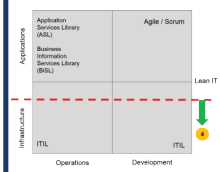
Starting Point of the Journey

- Large diversity of infrastructure and specific solutions for everything
- Technical silo-based infrastructure organization
- Strict separation of production and other environments
- Difficulty to match the pace of change required by customers

Key Actions

- Create Platform teams
- Standardize and automate routine support and maintenance work
- Integrate cloud technology
- Maintain and offer “infrastructure as code”

4 From Business Systems DevOps to “Full Stack” DevOps



Starting Point of the Journey

- Discussions about responsibility for service delivery
- Understanding that change is an integral part of service delivery

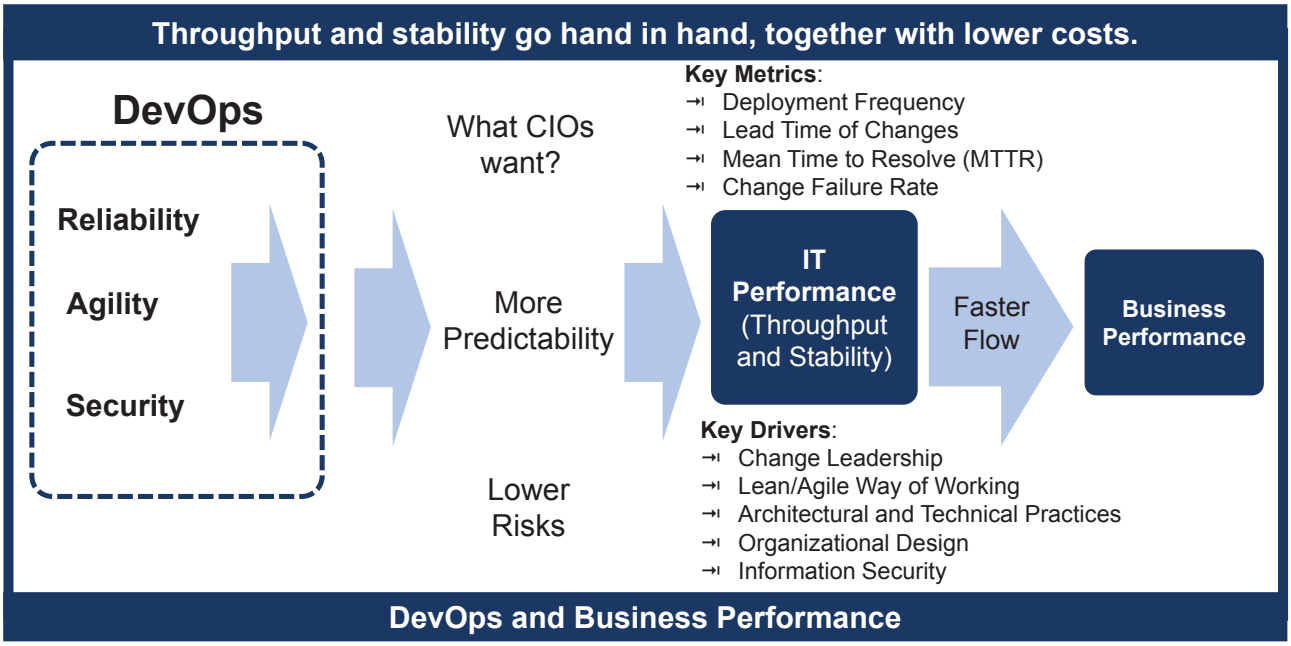
Key Actions

- Collaborate on improving flow
- Increase discipline of Business System and Platform DevOps teams
- Increased self-service provided by Platform DevOps teams
- End-to-end responsibility given to Business System DevOps teams

Activity Time: DevOps Impact on EJ Airways

A Look Back: IT Performance Drives Business Performance

Throughput and stability go hand in hand, together with lower costs



Case Study: EJ Airways

Note:

- *To know about EJ Airways, refer the Case Study: Easy Journey Airways (EJ Airways) provided in Appendix A.*
- *For reading the script of this video, refer to the 'VIDEO SCRIPTS' section of Case Study: Easy Journey Airways (EJ Airways).*

Group Discussion

How can ITIL® and DevOps contribute to the success of EJ Airways?

Note: Many solutions are possible for a given problem depending on the priorities of the business.

