Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Infrastructure Services | the way we see it

DevOps - The Future of Application Lifecycle Automation

A Capgemini Architecture Whitepaper – 2nd Edition

People matter, results count.

Contents

1 DevOps Introduction **04**
1.1 DevOps Market 05
1.2 DevOps overview 06

2 DevOps Challenge **08**
2.1 Complex pre-production/on-production build and run 08
2.2 Error prevention and diagnosis 08
2.3 Wall of confusion 08
2.4 Rate of change versus stability 09

3 DevOps Impact **10**

4 DevOps Value **11**

5 DevOps Concept **12**
5.1 Change culture 12
5.2 Development-to-Operations lifecycle 13
5.3 Common tooling 18

6 DevOps Implementation **19**
6.1 Define a clear target 19
6.2 Establish a clear transformation plan 20
6.3 Actively manage the plan execution 20
6.4 DevOps Maturity Model (DMM) 20

7 Summary **23**

8 Appendix B: References **23**

Table of Figures

Figure 1: Gartner's latest Hype Cycle for Enterprise Architecture 04
Figure 2: Gartner's DevOps market predictions 05
Figure 3: Software Development Lifecycle and DevOps 06
Figure 4: Development-to-Operation Challenges and DevOps "solutions" 06
Figure 5: Gartner's: "Pace Layers for DevOps" 09
Figure 6: DevOps Value 11
Figure 7: Cultural Change 12
Figure 8: Stakeholder Principles and Mindset 13
Figure 9: People Ecosystem 13
Figure 10: Development-to-Operations Lifecycle 14
Figure 11: Environment Reference Model 15
Figure 12: Environment Key Characteristics 17
Figure 13: Common Tooling Approach 18
Figure 14: DevOps Implementation Framework (DIF) 20
Figure 15: Capgemini's DevOps Maturity Model (DMM) 21

Introduction

Development to Operations (DevOps) will have a profound impact on the global IT sector in the near future. Realizing DevOps' full potential, IT vendors have been agile enough in providing new products and services under the label "DevOps inside", at an ever- increasing pace.

However, with the growth in product choices, conflicting definitions and competing services, customers often encounter confusion, while making complex purchase decisions. They often seem to be unsure about how to deploy DevOps and get the most out of the solution.

While not trying to delve deep into DevOps, the Whitepaper tries to answer the following key questions:

- What is DevOps?
- What is DevOps trying to achieve?
- What are the key benefits?
- How will DevOps achieve this?
- How best to make use of the new developments?

Its aim is to help the reader:

- Understand the DevOps concepts
- Understand its current value and restrictions
- Get insight into how we at Capgemini implement DevOps efficiently

For more information on Capgemini visit
www.capgemini.com

Gartner says that by 2016 DevOps will evolve from a niche to a mainstream strategy employed by 25% of Global 2000 organizations

About the Author

Gunnar Menzel has been an IT professional for over 25 years and is the VP and Chief Architect Officer for Capgemini's Infrastructure Business. His main focus is business-enabling technology innovation. Gunnar is also currently the president of the Open Data Centre Alliance. His main focus is business-enabling technology innovation.

Thanks to Andrew Macaulay, Ajith NC, Ajay Dhanesh for their invaluable contributions.

Sustainability

Please consider the environment and do not print this document unless absolutely necessary. Capgemini encourages environmental awareness.

Disclaimer

1 DevOps Introduction

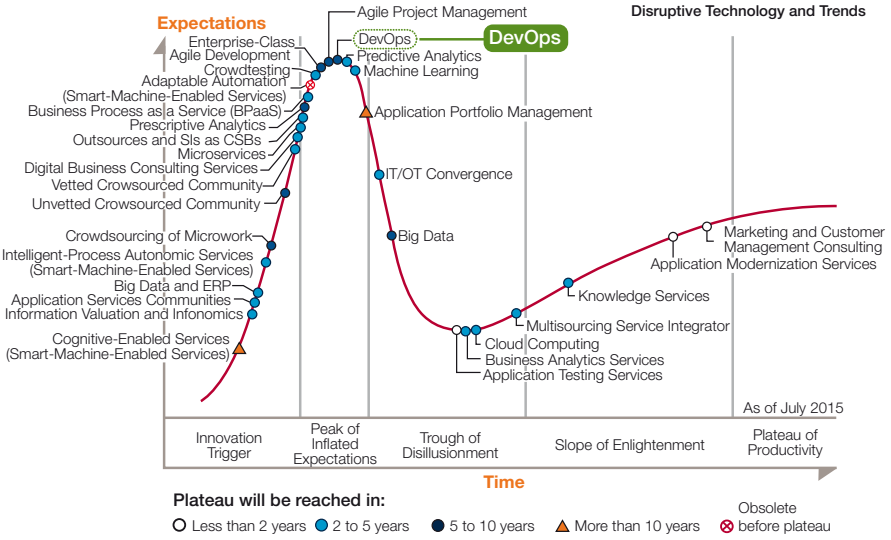
The speed of application development and application change is increasing and with it, the demand of “Rolls Royce like” quality. Companies look to build new capabilities with high expectations being placed on the IT department. Although the IT industry has taken huge leaps in technology innovation, the quality of application development projects has been lagging. Many IT projects run inefficiently, missing implementation deadlines and causing outages during or after implementation and therefore, costing significantly more than anticipated.

DevOps is the new development that addresses such inefficiencies. It connects development, quality assurance, and technical operations personnel in a way that the entire ‘build-release-run-repeat’[1] process operates as a factory, having clear roles and responsibilities and well-defined inputs and outputs[2].

In Gartner’s latest Hype Cycle for application services, 2015;[3] DevOps is positioned right in the Peak of Inflated Expectations:

DevOps is now at its inception point - enabling Business to drive real value.

Figure 1: Hype Cycle for Application Services, 2015



Gartner July 2015

The aim of DevOps is to revolutionize the transition, de-risk IT deployments, eliminate the excuse “but-it-works-on-my-system”, and break the silos between developers, testers, release managers, and system operators. The products and tools developed in this area focus on maximizing predictability, visibility, and flexibility, while maintaining stability and integrity.

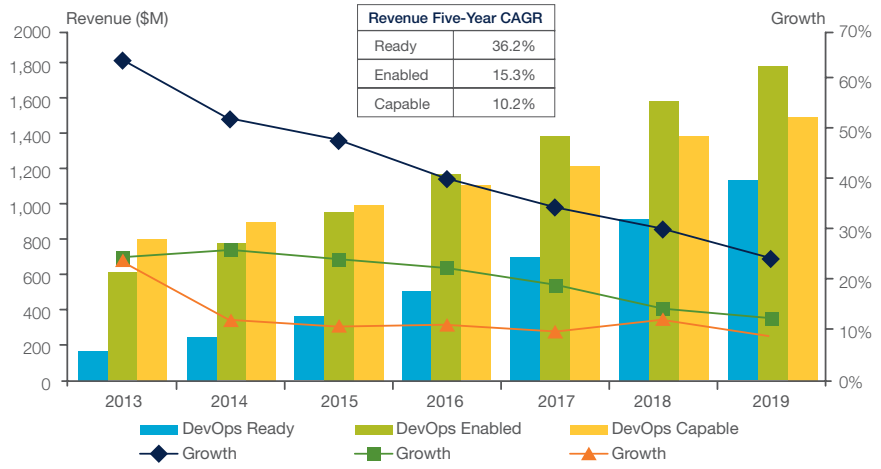
DevOps, in itself, is not a new concept. A development-to-operations lifecycle has existed for quite some time. The latest developments include the ambition to industrialize, automate, and connect the entire process covering infrastructure, application, as well as business changes. The prime focus is on outage reduction and quality improvement.

1.1 The DevOps Market

DevOps has fundamentally changed the way an IT organization works and how it gets things done. Since its inception in 2009, DevOps (coined as the "new Cloud" by market) has been adopted at a rapid pace, evolving from a niche concept to an integral part of enterprise IT strategy. This fast pace in adoption was mainly due to the immediate value realization that DevOps helps business to build better-quality products and services quickly and with greater reliability.

Figure 2: Gartner's DevOps market predictions

Gartner predicts that the DevOps market is set to grow from $1.9bn in 2014 to $2.1bn in 2015, showing a very healthy growth of ~11%.



CAGR = compound annual growth rate
Note: DevOps is a composite market, comprising software tools that are the part of other major markets within Gartner Market Share and Forecast documents.
Source: Gartner (February 2015)

Over the next 5 years, the DevOps market looks very positive, with many sources forecasting double-digit growth and a higher adoption rate

In addition to the research by Cloud providers, major product and tool vendors, DevOps market movement has been catalyzed by increased Cloud adoption, emergence of concepts like containerization, Platform-as-a-Service, micro-service architecture, service virtualization, and a strong contribution from the open-source community with several cost-optimized DevOps enabling tools. The early adopters or the "DevOps Unicorns" like Netflix, Amazon, Google, Etsy, and Snapchat have continuously innovated and showcased successful DevOps model variants like NoOps, ChatOps, and SmartOps.

In a recent study, Rackspace[4] interviewed 700 IT decision makers and found 55% of the organizations had already implemented or adopted DevOps and are looking for enhancements. Further 31% of them plan to use DevOps in the next 2 years. This adoption is among the largest in the side of technology for the initial implementation of tools.

Today, the DevOps trend not only goes beyond technology implementation and management but also focuses on a positive organizational change brought across its processes, cultural shift, and security and compliance aspects of the DevOps platform.

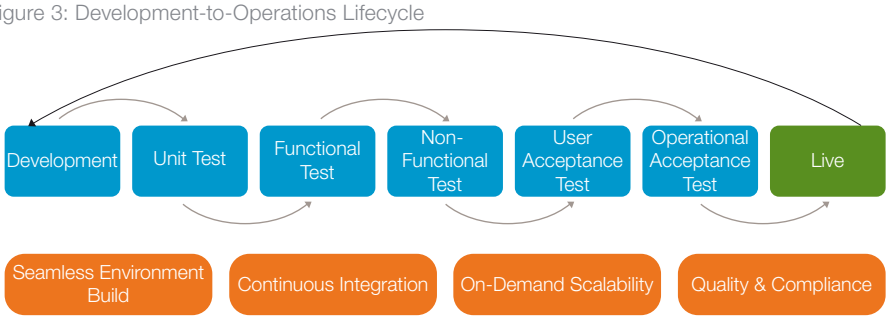
Over the next 5 years, the DevOps market looks very positive, with many sources forecasting double-digit growth and a higher adoption rate as larger enterprises begin to understand the benefits DevOps can bring in terms of cost reduction and agility.

Figure 3 implies a pure waterfall approach. DevOps "works" with multiple approaches: parallel software development lifecycles (rapid, agile, etc.)

1.2 DevOps Overview

In simple terms, DevOps refers to an umbrella concept that encompasses people, processes, and technologies required to connect development to execution.

Figure 3: Development-to-Operations Lifecycle

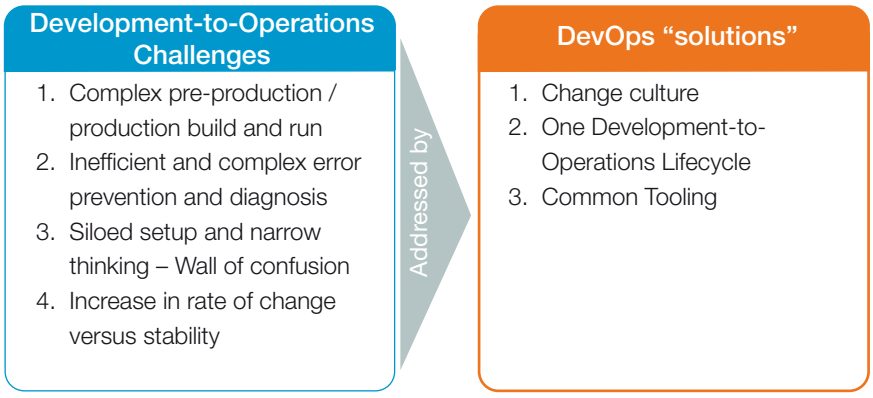


"Connect" in this context means ensuring that the development changes are tested and deployed in a way that is efficient and does not interrupt ongoing operations.

An important goal of DevOps is to reduce change-related outages. A number of factors give rise to outages. Complex releases, change and configuration management, higher change cycles, siloed setups, and narrow thinking are the key contribution factors.

DevOps as a concept (or Gartner[5] refers to it as a "philosophy") is trying to address four main challenges which affect costs of development and impact on run (live services or operations):

Figure 4: Development-to-Operation Challenges and DevOps "solutions"



As mentioned earlier, DevOps is not a new concept, but the efforts to harmonize several aspects of the entire Development-to-Operation process mark the beginning of a new era. Gartner[6] refers to DevOps as, "*… (an) IT service delivery approach rooted in agile philosophy with an emphasis on business outcomes, not business orthodoxy.*"

Many vendors and market analysts emphasize on 'agility', which connects development with operations to address the above-mentioned issues.

Agility is a critical element to:

- Reduce complexity
- Aid error diagnosis
- Remove the siloed setup
- Eliminate the narrow mindedness
- Deal with the rate of application change

To some, DevOps is a panacea for all ills; for others, it is a marketing gimmick, similar to the "Emperor's new clothes[7]". However, the truth lies somewhere in the middle.

Traditionally, we have been focusing too much on the process of delivering the actual solution, designed to create a change. However, to deliver the intended change, we need to focus on tools, processes, and people, while ensuring the stability of our service delivery.

DevOps makes an attempt to reduce inefficiencies by addressing the impact of development changes (new or existing) on three factors: tools, processes, and people. However, the DevOps concept has not attained full maturity. Gartner[8] outlines three key challenges for organizations planning to adopt DevOps:

1. The lack of a standard definition for DevOps has created confusion for infrastructure and operations (I&O) leaders, trying to adopt this philosophy
2. There is no standard or simple approach regarding the adoption of DevOps by an enterprise I&O leader, causing confusion about how and where to start
3. Each DevOps implementation is unique and every customer requires a customized approach

Activities are underway to address the lack of a standard definition (first challenge). The Open Group has announced a new Forum IT4IT[9] during its London Conference in October 2014, which attempts to standardize a DevOps related framework. The remaining two challenges will be subsequently addressed in the near future.

Three key implementation challenges:

1. Lack of standard definition
2. Lack of a standard approach regarding DevOps' adoption
3. Customization required for each implementation

2. DevOps Challenges

During coding and testing functional and non-functional areas, the IT department needs to ensure that the changes are accurately implemented and that the new capabilities comply with business requirements. Development cannot be carried out in a live environment. Hence, new and separate environments (pre-production and on-production environments) are needed to allow developers to write new or change an existing code (or change application configurations for COTS[a] products). This also enables testing of both functional and non-functional aspects of the end product.

Four main challenges are typically[b] prevalent in this scenario:

DevOps Challenges:

1. Complex pre-production/ on-production build and run
2. Error prevention and diagnosis

2.1 Complex Pre-production/On-production Build and Run

For sophisticated application landscapes with complex integration setups, creating, setting-up, and deploying a new environment is costly and prone to errors.

Generally servers, storage, network, and application environments are built and configured in a semi-automatic fashion. At best, different teams are responsible for deploying new servers and applications. Considering the huge cycle time from requisition to receipt of a new environment, the entire process seems to be fraught with inefficiencies. At times, in the absence of dedicated teams, a single resource manages the entire show, which adds to the inefficiency. Moreover, managing the consistency of the environment configuration introduces complexity, and the risk of failures adds to probabilities of lower quality of deliverables. At times, in light of cost considerations, some of the key functional teams like security and compliance may not participate during the environment build. This may impact the live deployment of code.

2.2 Error Prevention and Diagnosis

As a result of the complex challenges (mentioned above) moving/promoting code to the live environment involves risks and may give rise to outages. With higher manual intervention, the risks of errors and outages multiply. Many a time, such errors aren't caught instantly and may have an impact on other processes, for instance, testing of an application change. In this case, the tester would most probably assume that the error is due to the change and would request the developer to fix the issue. The developer would typically use a development environment to write and "unit test" his change. Now, as his environment differs slightly from the test environment no error is reported, i.e. 'it works for me.' Root cause analysis in this case will ultimately result in valuable resources being spent on diagnosis of why the change works in development environment but not in the test environment.

2.3 Wall of Confusion

Developers, on one hand, are incentivized to maximize 'change' — writing new code or enhancing the existing one , while on the other, operations personnel are encouraged to minimize change (in order to maintain KPIs[c] and SLAs[d]). Further, both groups operate in diverse organizations within a firm and may have different budgets. Typically, developers work within a project delivery organization (assisted by project), whereas operations staff work within support organization (assisted by technology). Location of both the resources in physically different areas adds to the problem. Developers often perceive operations staff as 'innovation blockers', while operations staff see developers as those who don't understand the importance of stability. Both the groups are critical for the long-term sustainability of the business, to decrease

a. COTS = Commercial off the Shelf

b. Typically means that these are most common causes

outages and minimize expenditure. To achieve this, a close collaboration among both the groups is crucial.

2.4 Rate of Change versus Stability

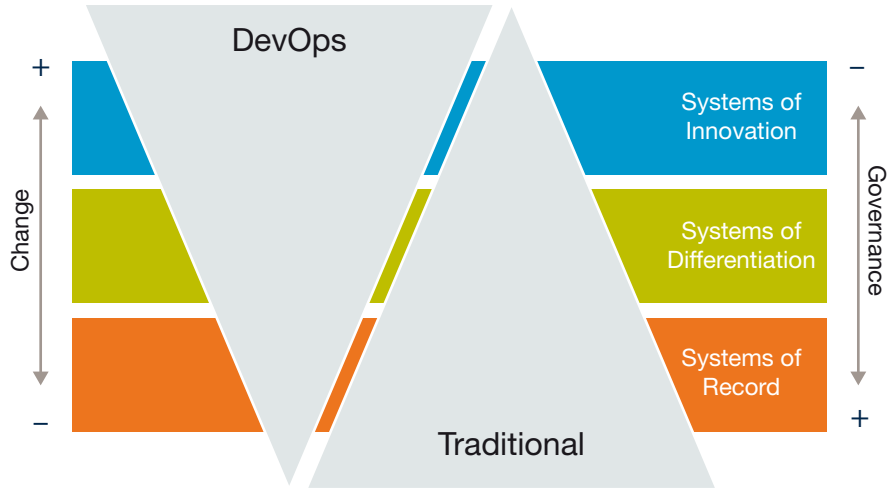
Different applications cause different rates of change versus stability. Gartner's 'Pace-Layered Application Strategy' outlines a way to categorize applications into:

1. Systems of Record (typically ERP-type applications)
2. Systems of Differentiation (typically business-specific applications, often COTS applications with customization)
3. Systems of Innovation (typically new web-based, agile development focused applications)

DevOps Challenges:

1. Wall of confusion
2. Rate of change versus Stability

Figure 5: Gartner's: 'Pace Layers for DevOps'



Source: Gartner (September 2014)

c. KPI=Key Performance Indicator

d. SLA=Service Level Agreements

3. DevOps Impact

80% of all production outages are change related. This is where DevOps can drive real value.

The IT Process Institute's Visible Ops Handbook[10] reports that 80% of unplanned outages are due to ill-planned changes made by administrators ("operations staff") or developers. A recent Gartner study projected[11] that Through 2015, 80% of outages impacting mission-critical services will be caused by people and process issues, and more than 50% of those outages will be caused by change/configuration/release integration and hand-off issues.

According to IDC's "DevOps and the Cost of Downtime: Fortune 1000 Best Practice Metrics Quantified"[12]:

- The average hourly cost of an infrastructure failure is $100,000
- The average hourly cost of a critical application failure is $500,000 – $1M
- The average number of deployments per month is expected to double in 2 years
- Unplanned application downtime costs the Fortune 1000 from $1.25 billion to $2.5 billion every year.

Calculating the costs associated with these outages is not straight forward, as costs include tangible/direct costs, such as lost transaction revenue, lost wages, lost inventory, remedial labor costs, marketing costs, bank fees, and legal penalties from not delivering on service-level agreements; and intangible/indirect costs, including lost business opportunities, loss of employees and/or employee morale, decrease in stock value, loss of customer/partner goodwill, brand damage, driving business to competitors or even bad publicity/press. Taking into consideration any of the above causes, the cost to the organization is in millions of dollars. Furthermore, the increased number of manual changes and deployment relates directly to the average downtime range, thus increasing the associated cost too.

DevOps can increase deployment frequency by factor 30 and lead times by factor 8000!

The goal of DevOps is not to eliminate errors/downtime, rather automate manual tasks and create repeatable processes with greater transparency, traceability, thus zeroing down on the level of manual intervention and helping organizations attain a stable and reliable IT infrastructure, ensuring uninterrupted business continuity with a level of acceptable risks. By this process, Devops help organizations attain a stable and reliable IT infrastructure. Therefore, one of the drivers of increasing interest in DevOps practices and tools is the significant reduction of unplanned downtime and the cost associated with it.

A recent Puppet Labs study — 2015 State of DevOps — states that high-performing IT organizations experience 60 times fewer failures and recover (Mean-time-to-Recover) from failure 168 times faster than their lower-performing peers.

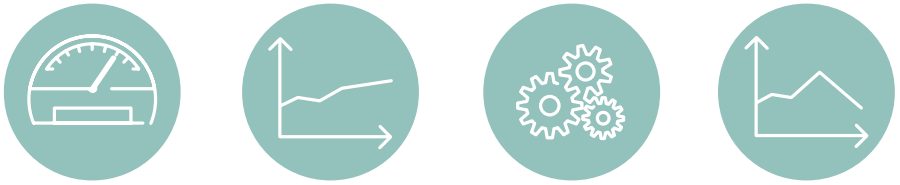
DevOps, if applied correctly to an organization, will play a significant role in improving IT stability and has a direct impact on business.

4. DevOps Value

DevOps can drive agility, quality, innovation whilst reducing outages in production

DevOps has 4 main value propositions

Figure 6: DevOps Value



Increased Agility
To enable near-instant change deployment

Increased Quality
To increase end-user satisfaction

Improved Innovation
To increase Innovation cycles

Reduced Outages
Upto 80% outages are change-related

Agility and outage-reduction are the two main drivers that clients target when implementing DevOps capabilities. However, increased quality and innovation cycles are another set of key benefits clients might want to consider when applying DevOps. To a large extent, most of these drivers can directly be related and translated into cost savings. Reducing outages will have an impact on cost avoidance; increased quality will have an impact on end-user satisfaction and therefore, clients may be able to increase their profits.

4.1 Increased Agility

Speed to market or better-maximized agility through a constant and fully integrated deployment capability is one key aspect here. Moving from a release cycle from every quarter to deploying changes on a minute-by-minute basis is a real aspiration. For many clients this is a massive leap, for some this is a reality.

4.2 Increased Quality

Puppet Labs' survey[13] suggests that high-performing organizations are deploying code 30 times more frequently, with 50% fewer failures than their lower-performing counterparts. Increased quality is one of the key benefits of DevOps. It will, however, increase quality only when the organization reaches a certain level of maturity. As outlined in the Capgemini Maturity Model (detailed below) we differentiate 5 levels of maturity: Basic, Emerging, Co-ordinated, Enhanced, and Top Level.

4.3 Improve Innovation

Experiencing fewer outages and deploying code with increased quality will lead to more time spent thinking about further improvements or new ways of working. It will enable the organization to drive more value, rather than having to dedicate time fixing issues caused by changes deployed.

4.4 Reduced Outages

As outlined before, outage-reduction is a big area of value. By applying a DevOps approach, companies will be able to avoid loss of sales and other outage related implications by improving ways of working, automation, and continuous deployment.

5. DevOps Concept

DevOps tries to address the aforementioned challenges through higher automation, increased visibility, and tighter control of the pre-production and on-production environments and deployment/promotion of code through diverse environments.

It also tries to reduce the “wall of confusion” between development and operations personnel by harmonizing the development and operations tools (allowing feedback from operations to development) as well as re-aligning objectives and incentives.

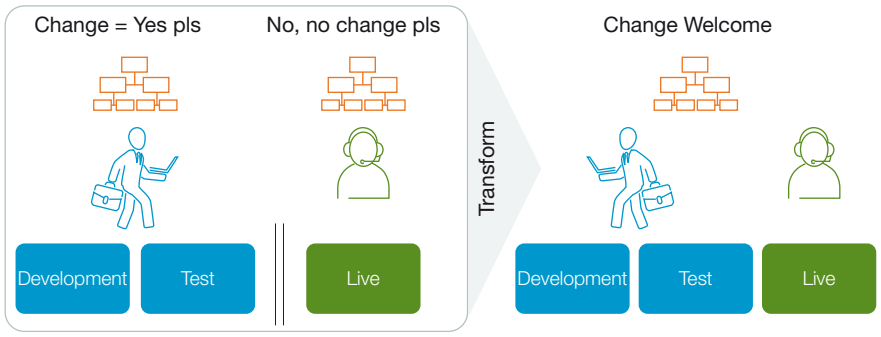
5.1 Change Culture

Ensuring that each individual is part of the entire solution lifecycle covering development and operations is one key DevOps concept — to break the ‘wall of confusion’. Its main aim is to create an appraisal methodology that can jointly measure and reward performance. All key parties that are involved in the “development-to-operations-service” supply chain must share same/similar objectives and targets, and each person’s performance must be assessed against their impact within the context of the entire solution lifecycle.

A key to successful implementation: address the cultural change

Individuals should be assessed according to the overall change development and deployment as well as the resulting process stability. Thus, both the work groups should be measured using a supply chain rather than a siloed approach.

Figure 7: Cultural Change



DevOps is not only tooling; it encompasses people, processes, and tools, with ‘people’ being the critical element of the equation. In order to effectively use the tools across the process or lifecycle, people should be encouraged to abide by a number of guiding principles to drive a common mindset:

Figure 8: Stakeholder Principles and Mindset

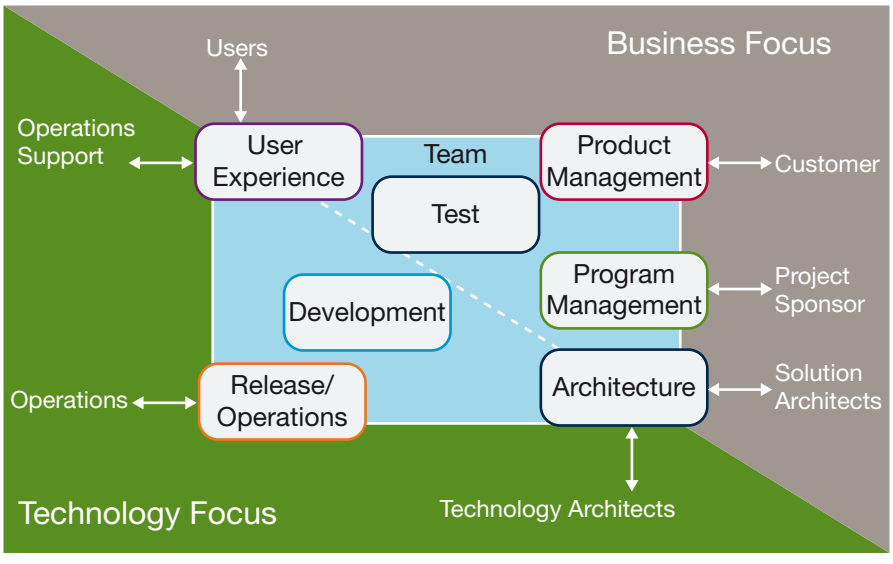
Stakeholders Principles	Mindsets
1. Partner with customers	Focus on business value
2. Work towards a shared vision	Advocate for your constituency
3. Deliver incremental value	Take pride in workmanship
4. Invest in quality	Deliver on your commitments
5. Empower team members	Keep a solution perspective
6. Set clear accountability	Foster a team of peers
7. Learn from experiences	Practice good citizenship
8. Foster open communications	Improve continually
9. Stay agile (expect and adapt to change)	Understand qualities of service

In short, strive towards excellence.

Fostering, driving, and encouraging each individual to follow these principles is important to drive a common objective. However, to reach a common consensus, every individual needs to comprehensively try and understand the role of other resources.

The ideal resources ecosystem

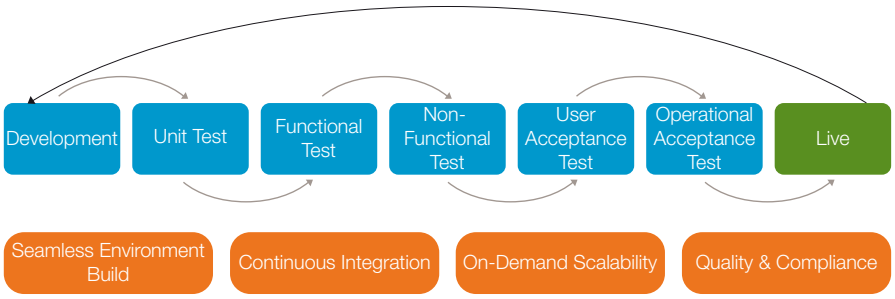
Figure 9: People Ecosystem



5.2 Development-to-Operations Lifecycle

The entire development-to-operations lifecycle must be seen and operated as one coherent end-to-end process, rather than numerous multiple and diverse steps. Individual methodologies can be applied for individual steps, such as agile or waterfall, as long as these can be connected together to form one end-to-end process.

Figure 10: Development-to-Operations Lifecycle



DevOps suggests implementing a continuous deployment cycle and not one that ends at live

'Connect' is a key aspect here. For organizations located across diverse geographies, change is global. Application development is being increasingly carried out in remote locations. Unit code as well as non-functional testing, interface and user acceptance testing, and end-user training are also executed in diverse locations. However, a commonly agreed development-to-operations lifecycle is deployed.

A feedback loop runs from the live environment to the start of the cycle. Constant improvement to DevOps is an important facet of the lifecycle, and this requires:

- Bug reporting – Providing an easy means to capture (through social media for external sites), triage, track, or add to developers' backlogs
- Feature suggestions – Allowing users to provide their feedback on a service which may come through tools such as UserVoice, which can then be integrated with the lifecycle
- Availability, performance, and usage monitoring – Allowing feedback on operational aspects to provide insights (or maybe report defects) into future developments
- Instrumentation – Allowing more specific measures to be captured which are meant to be used as insights into future developments

Part of the process definition is a clear understanding of:

1. Principles and Purposes (see 3.2.1)
2. Environment Reference Model (see 3.2.2)
3. Environment Key Characteristics (see 3.2.3)

5.2.1 Principles and Purposes

To deploy a common development-to-operations lifecycle, one requires a clear definition of the key process principles and purposes. The development-to-operations lifecycle:

- Is necessary to build, test, and deliver the changes to existing applications or the implementation of new applications
- Plays a crucial role to manage the large volume of changes associated with a multi-platform environment
- Provides traceability of change processes from requirements to release without any disruption

- Provides a single version of truth, as end-to-end traceability ensures that the operations team implementing the application change has a solid understanding of the requirements
- Enables a strong foundation to understand the requirements
- Ensures adequate validation during the integration or acceptance-testing phase of the release

5.2.2 Environment Reference Model

Subsequent to understanding the DevOps lifecycle principles, it is crucial to define the use of various environments needed to support the different steps. To achieve this, the characteristics and type of data, SLAs, and the amount of change and release management must be clearly outlined:

Figure 11: Environment Reference Model

Environment	Purpose	Characteristics	Type of Data	SLAs	Change Release Management
Workshop Environment	Temporary environment used for presentation, pilots, etc	Sometimes called sandpit environment. Built and maintained by the project team	Non-live only	No (unless explicitly defined/ agreed)	None
Performance Testing Environment	Test environment to simulate load testing to prove performance characteristics	Temporary only; Built and maintained by the project team	Full live “like” data	No (unless explicitly defined/ agreed)	None
Development Environment	Environment to develop software applications; ideally one environment per application	Usually permanent, either created by project or by central service, testing only at code level	Only subset of live “like” data	No (unless provided as a pre-defined service)	None
Functional Test Environment	Test environment to perform business related testing	Temporary environment, created by the project and functional testing teams	Only subset of live like data	No (unless provided as a pre-defined service)	None
User Acceptance Test Environment	Test environment to perform end-to-end functional testing	Temporary environment, created by the project and functional testing teams	Only subset of live “like” data	No (unless provided as a pre-defined service)	None

Environment	Purpose	Characteristics	Type of Data	SLAs	Change Release Management
Training Environment	Training environment to perform end-to-end functional training	Temporary environment, created by project teams to perform end-user training	Only subset of live “like” data	No	Yes
System Integration Testing Environment	Test environment to test interfaces and data-related aspects	Temporary environment, created by project teams to perform system-related interface testing	Only subset of live “like” data	No	Yes
Operational Testing Environment	Test environment to test all non-functional related aspects	Permanent environment, created to simulate the live environment	Only subset of live “like” data	No	Yes
Support Environment	Environment to replicate live issues and develop/trial fixes from either the frontend or backend	Permanent environment, created to simulate the live environment	Only subset of live data	No	Yes

Although these environments are not exhaustive, they constitute a list of possible environments and should be seen as a reference model. Once the DevOps implementation approach is in place, a decision to choose the right type of development and testing environments should be taken following a thorough analysis of the business operations. After choosing the type of environments, the IT department has to decide upon the data residency. Data may reside either on local servers or in a private or public Cloud environment. Each environment is expected to have the following characteristics:

Figure 12: Environment Key Characteristics

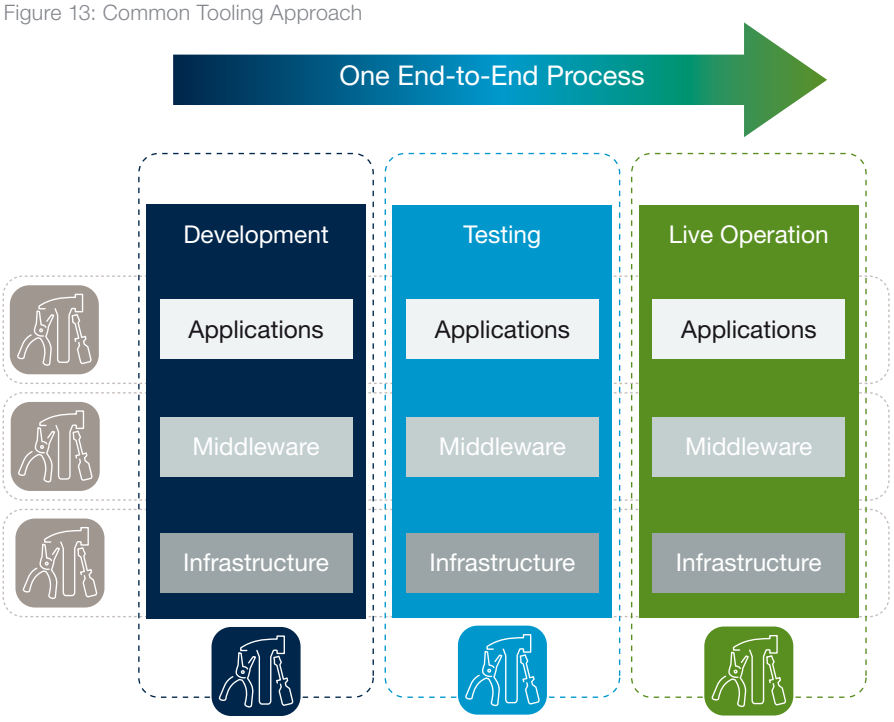
	Title	Statement	Motivation	Implications
1	Complete	Each environment must be complete with all components, even if it is not subject to testing. 'Complete' here implies the level of separation of infrastructure and networking components with an appreciable level of security. It also implies working with same privileges, a live environment would offer. This would ensure that a piece of code produces output in a similar fashion, whether in a development or live environment, despite a firewall between the two	To ensure accurate test results, it is important to include the dedicated and shared components in a single pre-production environment	When dedicated applications use shared components such as databases, etc., it is important to perform complete application specific tests followed with regression tests, using the shared component. This should be done from a perspective of performance and availability
2	Up-To-Date	It is essential to maintain all the environments and their components up-to-date. Changes in the live environment must be routed back to the master and from there pushed, to the different environments	'Up-to-date' for a live environment is performed for an exact simulation of the live environment. This involves binary codes as well as system configuration.	Proper change and configuration management is vital to be up-to-date
3	Management	Each environment must be owned and managed by a single person to ensure that the environment is complete and up-to-date	Ownership is important to ensure consistency during the project lifecycle. Usually live support team takes the ownership of pre-production or new services. However, in case of new projects, ownership might be taken by a different entity	All environments must be owned and managed by a single person – usually the change manager, who ensures quality and integrity of the development environment.
4	Support	Each environment must be supported by adequate support resources	It is important to ensure accurate testing and fast identification and isolation of bugs or errors. To ensure adequate system enhancements for live services, the pre-production environment is usually backed by live support. While identifying errors and raising queries, the pace of responses should be maintained at an optimum rate, before the test cycle starts.	Allocate adequate resources during the project conception stage.
5	Independence	Each environment must be independent, isolated from other environments, and complete with all resources	This is important to ensure that no other pre-production environment is affected during implementation of changes	Ensure independence while determining the components to be included in an environment

There is no single DevOps tool: Use, apply, and connect the right tools using an end-to-end process

Common Tooling

This is a prime aspect that garners a lot of attention. A single tool or a combination of tools that allows for a simple or fully automatic request should be deployed. It should be able to create, operate, and destroy any type of pre-production or on-production environments. It should include infrastructure related, middleware and application-related components. The aim should be to accelerate the transition through environments in an efficient and cost-effective manner. The result would be more testing in an early testing phase, leading to higher quality. A number of IaaS tools, as well as middleware and applications-based process tools are available today. While Docker, Puppet, Chef, and ControlTier are examples of existing tools; VMWare's Codestream is a new entrant. Traditional enterprise vendors provide a host of toolsets. For instance, Microsoft offers Team Foundation Server/Visual Studio Online along with Microsoft System Centre and Azure. They sync with tools like Docker and Chef and also integrate with social media through UserVoice, Azuqua, etc.

Figure 13: Common Tooling Approach



These tools allow developers and operational staff to request, create, operate, and destroy pre-production or on-production environments in an industrialized manner.

Gartner, during April 2014, published a quick overview of the "Cool Vendors in DevOps"[5] . However, as mentioned above, there is no one tool that does it all; instead a 'toolset' approach needs to be employed. The toolset should cover the entire development-to-operations lifecycle, from infrastructure to business. It should empower the resources to execute the work allocated to them in an efficient manner.

6. DevOps Implementation

As outlined in 1.1, the adoption of DevOps is hampered by a number of key aspects[14]:

1. The lack of a standard definition for DevOps has created confusion for infrastructure and operations (I&O) leaders, trying to adopt this philosophy[8]
2. There is no standardized or simplified approach regarding the adoption of DevOps by an enterprise I&O leader, causing confusion about how and where to start[8]
3. Each DevOps implementation is unique, and every customer requires a customized approach[8]

Three steps for a successful DevOps **implementation:**
1. Define a target
2. Establish a plan
3. Manage plan execution

In practice, tools, methods, and technologies are seldom deployed on green-field sites and having implemented DevOps for more than a decade now, we believe that the key to successful implementation is to:

- Define a clear target
- Establish a clear transformation plan
- Actively manage the plan execution

DevOps implementation should not be merely perceived as deploying a new tool like CodeStream or Docker. It should be viewed from a bigger perspective and should be planned and executed in an efficient manner. Poorly planned DevOps implementation may result in significantly higher costs.

DevOps implementation starts with creating a business case, mapping a way for code migration between environments (considering people, processes, and technology), and placing focus on the target. Understanding the 'As-is' scenario, mapping the 'To-be' scenario, and estimating the benefits of moving to the 'To-be' scenario are critical for success. DevOps implementation should be backed by a strong business case. Every environment does not benefit from full or partial DevOps deployment. For instance, environments with little change requirements may not benefit from DevOps implementation at all. In our experience, many DevOps projects have failed due to the absence of a strong business rationale or a poorly planned start.

6.1 Define a clear target

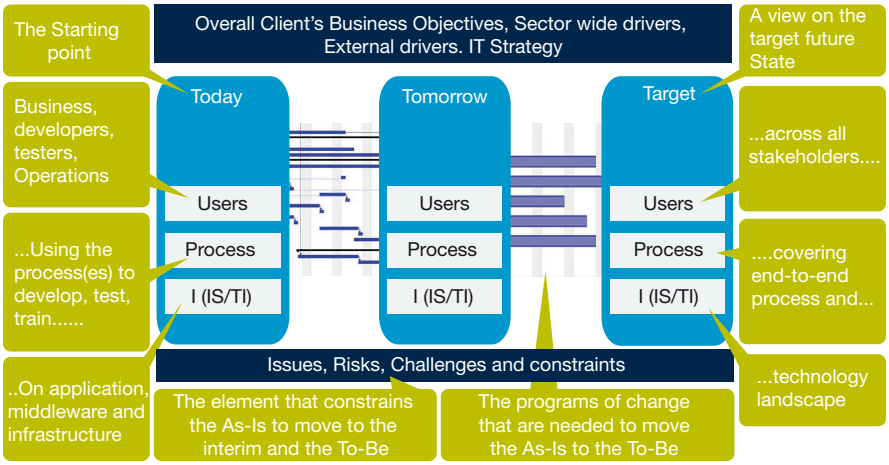
DevOps claims to reduce impact of changes to reduce cost and minimize impact to the live services. As applicable to every change project, the decision to change culture or processes and to deploy the right tools must be backed by a strong business case. Many businesses struggle to take the right decisions at this stage. To estimate the benefits of DevOps implementation within their environment, they should analyze the existing situation — the existing tools, processes, resources, and their skills.

Then, as the "snowflake" point in Gartner's paper[8], each client context is different and what works for one might not work for the next.

Capgemini is working on a generic DevOps Implementation Framework (DIF) that will formulate the "artifacts" needed to define the target and create a business case.

To successfully deploy DevOps a plan that covers people, process and tools is needed

Figure 14: DevOps Implementation Framework (DIF)



6.2 Establish a clear transformation plan

Once a business case is created and approved, a detailed plan of actions is needed to manage the implementation of the changes needed to achieve the anticipated outcomes set out in the business case. Typically, three actions need to be followed:

4. Change the culture
5. Establish one Development-to-Operations Lifecycle
6. Deploy common tooling

The choice of activities that need to be executed for a solution depends on the actual context and needs to be established during the "define a clear target" step.

6.3 Actively manage the plan execution

In addition to careful formulation of the plan, it is important to carry out an efficient execution.

6.4 DevOps Maturity Model (DMM)

DevOps cannot be achieved by executing a single project. Organizations need to implement tools and processes as a coordinated program and evolve the cultural aspect over a period of time. The progress must be tracked by identifying the stage the organizations have reached and then creating a road map to achieve the desired.

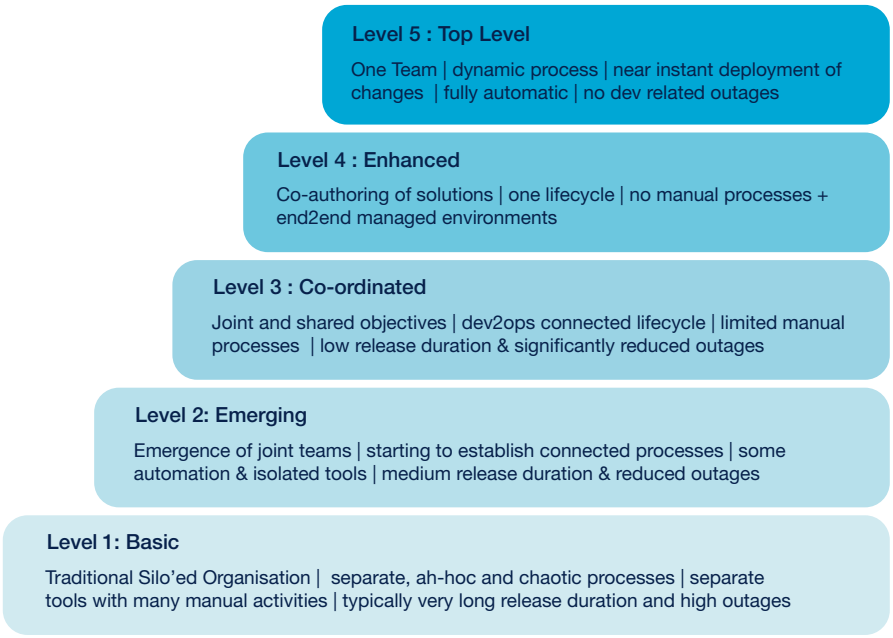
Capgemini's DevOps maturity model (DMM) is a framework that enables businesses to identify the current maturity level and actions required to be taken to reach the next level. DMM gauges the maturity level using the core three dimensions of people, process and tools. This model can also be used as a guide for improvement across a project, division, or an entire organization.

Capgemini DMM helps stakeholders to understand and derive an implementation plan and can be used a reference to outline the current DevOps maturity in an organization.

Capgemini DMM has five levels of maturity: starting at Level 1 stage with siloed team, manual and ad-hoc processes to Level 5 stage of highly matured "One Team" with fully automated, dynamic process.

The maturity model covers 5 Levels from Basic to Top, and 3 focus areas; people, process and tools.

Figure 15: Capgemini's DevOps Maturity Model (DMM)



Seldom can Devops be implemented in one go - a step-by-step approach is needed

6.5 Level 1: Basic

At the Basic level, the following characteristics are apparent:
People: Separate strategy, design, development, testing, and live operations teams. Complete lack of terms of references. No joint sessions, get-togethers. Teams focus on their own direct targets and objectives only. No joint or shared objectives and no overall reward system. People only feel accountable for their immediate area — no common or overarching ownership

Process: Separate and disconnected processes are in place, which are ah-hoc, reactive and chaotic. No common end-to-end process framework, no common sign off criteria, or any joint solution design characteristics that support appropriate "–ilities" (availability, stability, flexibility)

Tools: No automation tools, majority of activities are manual, ad-hoc, and unplanned. No integration between hardware provisioning, operating system installation/configuration and middleware/application related provisioning/installation. No sharing of joint configuration information with all information being stored and retained in different repositories

6.5 Level 2: Emerging

At the Emerging level the following characteristics are apparent:
People: Limited changes to basic — still very siloed and separate teams with no single team/person taking end-to-end responsibility. Developers mainly focus on functional requirements with very limited focus on non-functional requirements. However there is the emergence of some shared/joint touch points, where some developers and operational staff engage

Process: Limited changes to basic — there are some attempts to establish better managed processes, however these are restricted to specific environments only, i.e. covering development or UAT (user acceptance testing) only

Tools: Some minor changes to Basic, mainly targeted at developing automatic scripts covering hardware and operating system related aspects. As per Basic, these are mainly targeted at the development environments. Most other environments such as testing, training, SIT (system integration testing) are being manually installed.

6.5 Level 3: Co-ordinated

At the Co-ordinated level, the following characteristics are apparent:
People: Mainly siloed organizations, however lead architect/lead designer(s) increase their scope to also include operational aspects. Joint sessions are held to increase wider visibility for instance, key operational staffs are actively engaged in the design and build phase. Developers are also measured on operational characteristics

Process: Still mostly separate processes covering the entire solution lifecycle, however there are some joint process points where development and operational aspects are jointly covered. Better understanding of the entire environment setup and characteristics

Tools: most of the development environment setup is being created automatically. Only application related components are manually installed

6.5 Level 4: Enhanced

At the Enhanced level the following characteristics are apparent:
People: Joint teams that cover the entire solution lifecycle. Lead architect owns the entire solution, including functional and non-functional requirements covering design, build, test, and run

Process: Single overall process covering the entire solution lifecycle from design, build, test to run. Clear visibility of all projects/changes that are at different stages, and all compliance levels (functional and non-functional). Clear view of the entire environment setup and characteristics

Tools: most of the development environments, setup, testing and live are being created automatically. This now covers servers, operating system, as well as most middleware and application related components

6.5 Level 5: Top Level

At the Top level, the following characteristics are apparent:
People: One team, co-located and extensive collaboration and knowledge sharing

Process: Single, overall process covering the entire solution lifecycle from strategy, planning to design and build, test to run

Tools: All environment setups created automatically from a single repository. This covers all aspects: servers, operating system, operating system as well as all middleware and application related components. No manual processes in place

7. Summary

For Capgemini DevOps is a way of *collaborating* and *industrializing* using highly automated approaches to deploy solutions that evolve as fast as your business needs it.

For Capgemini DevOps is a way of ***collaborating*** and ***industrializing*** using highly automated approaches to deploy solutions that evolve as fast as your business needs it. By adopting DevOps an organization can dramatically improve the value delivered by its business. The team centric DevOps ethos ***tears down*** traditional silos to tightly integrate business, development and operations to drive agility and service delivery excellence across the entire lifecycle

DevOps is an 'old' approach understood and discussed by a relatively small number of professionals. To master it, companies must apply a much more holistic approach. With the advent of new technologies and growing demand for faster processes and better quality, DevOps has acquired new dimensions. Organizations across the globe have been implementing a full or partial DevOps solution. However, the road to DevOps is not straight. DevOps is a complex concept with no clear definition or list of products. It lacks a common vocabulary and capabilities required for DevOps implementation differ from one environment to the other.

To overcome the challenges we suggest:

1. Define a clear target
2. Establish a clear transformation plan
3. Actively manage the plan execution

To get maximum benefit from the DevOps implementation, we recommend focus on three key areas - change of culture, connection of processes, and common tooling. This is crucial to reduce development-to-operations costs and minimize change related outages.

8. Appendix A: References

1. Capgemini, Technovision 2016, https://www.capgemini.com/blog/cto-blog/2015/11/technovision-2016-build-release-run-repeat
2. Capgemini, TechnoVision 2015, http://www.capgemini.com/blog/cto-blog/2014/11/welcome-to-technovision-2015
3. Gartner, Hype Cycle for Application Services 2015, July 2015, Betsy Burton, Philip Allega
4. Rackspace, DevOps Automation Report, October 2014, https://www.rackspace.co.uk/sites/default/files/devops-automation-report.pdf
5. Gartner, Cool Vendors in DevOps, 16th April 2014, G00262716, by Ronni J. Colville, Jim Duggan, Jonah Kowall, Colin Fletcher
6. Gartner, The Virtualisation Scenario: Servers & Beyond, Philip Dawson, 2013
7. http://en.wikipedia.org/wiki/The_Emperor's_New_Clothes
8. Gartner, Seven Steps to Start Your DevOps Initiative, 16 September 2014, G00270249, Ronni J. Colville
9. IT4IT OpenGroup Forum, http://www.opengroup.org/IT4IT
10. The Visible Ops handbook, http://www.itpi.org/the-visible-ops-handbook-review.htm
11. Gartner, G00208328, Ronni J. Colville, George Spafford, 27th October 2010, http://img2.insight.com/graphics/no/info2/insight_art6.pdf
12. IDC, Stephen Elliot, http://info.appdynamics.com/rs/appdynamics/images/DevOps-metrics-Fortune1K.pdf
13. Puppet Labs, DevOps Report, https://puppetlabs.com/2015-devops-report
14. The Agile Admin, http://theagileadmin.com/what-is-devops/

For more details contact:

Author:

Gunnar Menzel
VP, Capgemini
gunnar.menzel@capgemini.com
0044 870 905 3325

Co-Author:

Andrew Macaulay
Managing Solution Architect
andrew.macaulay@capgemini.com

Contributors:

Ajith NC
ajith.nc@capgemini.com

Ajay Dhanesh
ajay.dhanesh@capgemini.com

About Capgemini

With 180,000 people in over 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.573 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at
www.capgemini.com

MCOS_GI_MK_20151207