Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Infrastructure Services | the way we see it

DevOps - The Future of Application Lifecycle Automation

A Capgemini Architecture Whitepaper

People matter, results count.

Contents

Table of Figures

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.

Development to Operations (DevOps) implementations will increase significantly during 2015-16.

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?
- 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**

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 a part of the OpenGroup IT4IT forum as well as leads Capgemini's Infrastructure Architecture Community.

Thanks to Andrew Macaulay, Ajith NC, Sameer Mutalik and Narasimhan PII 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. While 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 significant more than anticipated.

DevOps is the new development that addresses such inefficiencies. DevOps connects development, quality assurance, and technical operations personnel in a way that the entire ‘build-test-release-run-repeat’ process operates as a factory, having clear roles and responsibilities and well-defined inputs and outputs [9]. It is a concept that connects developers, quality assurance and technical operations people.

DevOps is a part of the “Innovation Trigger Phase” as outlined in Gartner’s latest Hype Cycle for Enterprise Architecture

In Gartner’s latest Hype Cycle for Enterprise Architecture [8] DevOps is positioned right in the Innovation Trigger phase:

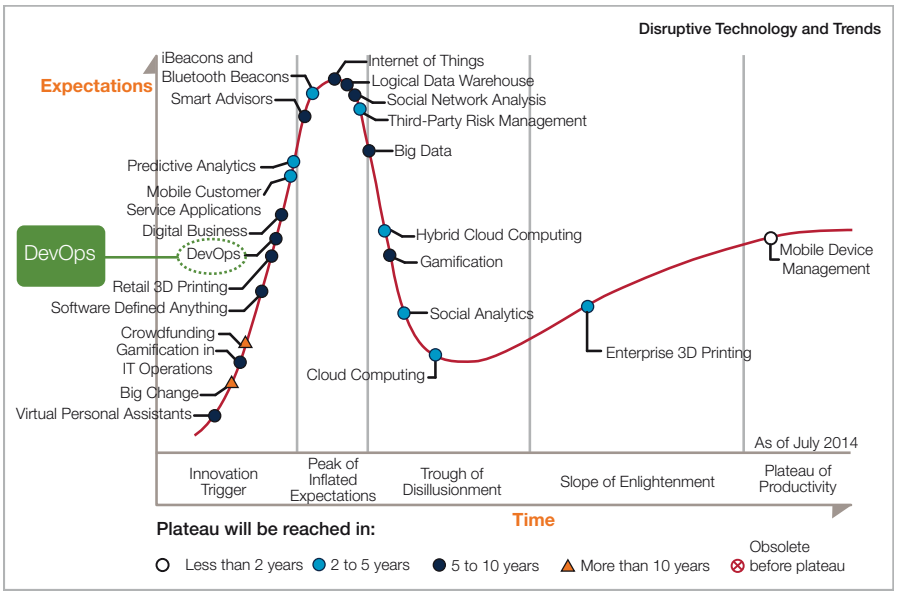


Figure 1 : Gartner’s latest Hype Cycle for Enterprise Architecture [8]

DevOps aim 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 operation lifecycle has been existing 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 DevOps Overview

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

Figure 2 implies a pure waterfall approach. DevOps "works" with multiple approaches – i.e. parallel software development lifecycles (rapid, agile etc).

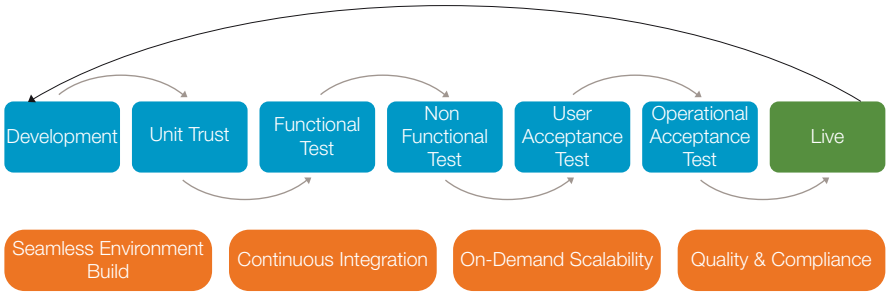


Figure 2 : Development-to-Operations Lifecycle

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

According to estimates, 80% of the outages are due to application changes and/or new application developments. An important goal of DevOps is to reduce such 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[1] 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):

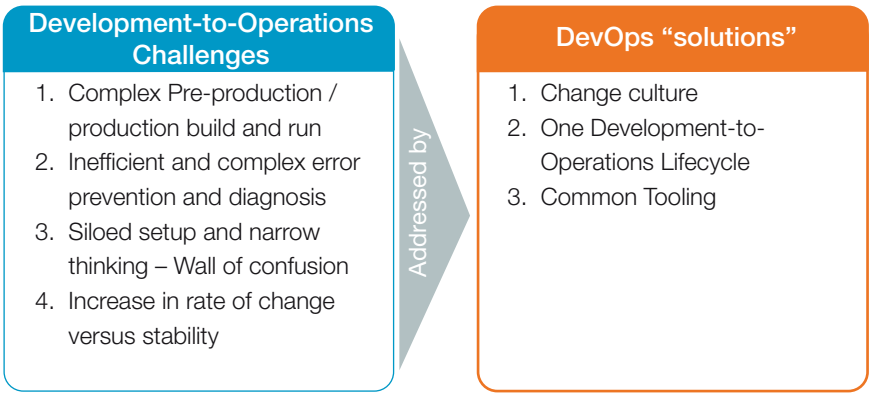


Figure 3 : Development-to-Operation Challenges and DevOps "solutions"

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. In[5] Gartner refers to DevOps as, "…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[1] . 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 in [2] 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 [4] 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. We believe, DevOps will make a key development idea in 2015 and will mark the entry of numerous application development projects.

Three key implementation challenges:

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

[1] http://en.wikipedia.org/wiki/The_Emperor’s_New_Clothes

2. DevOps Challenge

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 non-production environments) are needed to allow developers to write new or change an existing code (or change application configurations for COTS[2] products). This also enables to test both functional and non-functional aspects of the end product.

Four main challenges are typically[3] prevalent in this scenario:

DevOps Challenges:

Complex Pre-production/ on-production build and run
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 as well as 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 absence of dedicated teams, a single resource manages the entire show, which adds to inefficiency. Moreover, managing the consistency of the environment configuration introduces a 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] COTS = Commercial off the Shelf

[3] Typically means that these are most common causes

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[4] and SLAs[5]). Further, both the 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 outages and minimize expenditure. To achieve this, a close collaboration among both the groups is crucial.

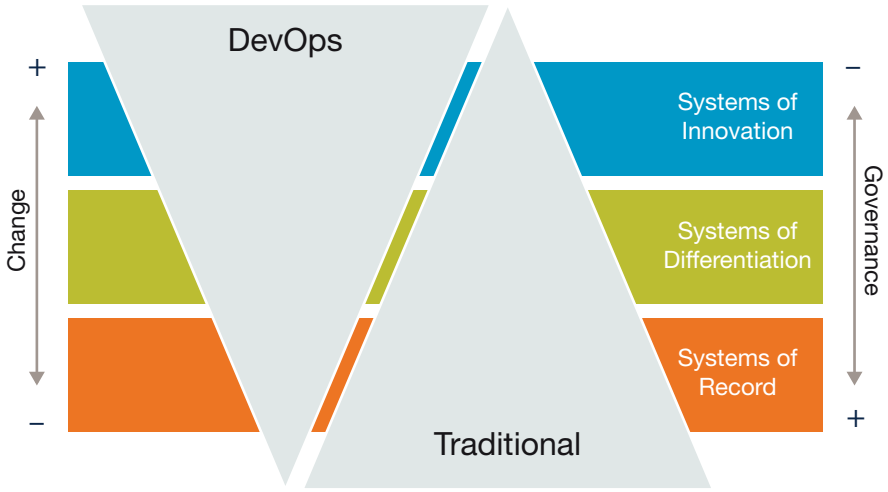
2.4 Rate of change versus Stability

DevOps Challenges:

Wall of confusion
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)



Source: Gartner (September 2014)

Figure 4 : Gartner's: 'Pace Layers for DevOps'

[4] KPI=Key Performance Indicator

[5] SLA=Service Level Agreements

3. DevOps Concept

DevOps tries to address the aforementioned challenges through higher automation, increased visibility, and tighter control of the pre-production and non-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.

3.1 Change culture

A key to successful implementation:

Address the cultural change

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 “from-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.

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.

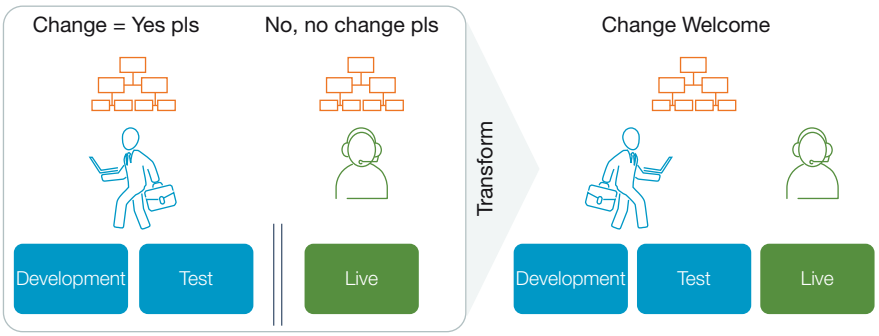


Figure 5 : Cultural Change

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 lifecyle, people should be encouraged to abide by a number of guiding principles to drive a common mindset:

Stakeholder 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 & adapt to change)	Understand qualities of service

Figure 6 : Stakeholder Principles and Mindset

The ideal resources ecosystem

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 role of other resources.

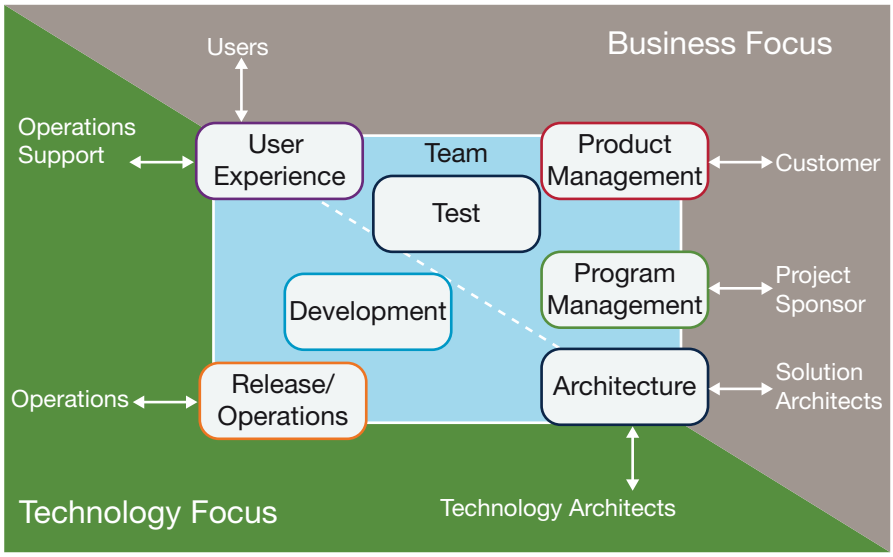
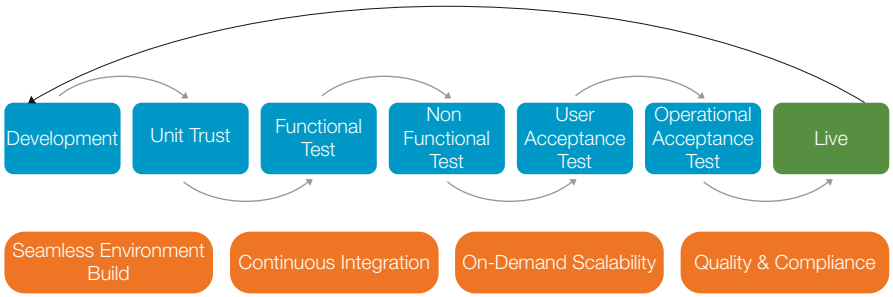


Figure 7 : People Ecosystem

3.2 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 – so long these can be connected together to form one end-to-end process.



Development-to-Operations Lifecycle

'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)

3.2.1 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 change 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

3.2.2 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:

Environment	Purpose	Characteristics	Type of data	SLA's	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

Environment	Purpose	Characteristics	Type of data	SLA's	Change Release Management
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)	Yes
Training Environment	Training environment to perform end-to-end functional training	Temporary environment, created by project teams to perform enduser 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

Figure 8 : Environment Reference Model

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:

	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, ho ensures quality and integrity of the development environment.

	Title	Statement	Motivation	Implication
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

Figure 9: Environment Key Characteristics

3.3 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 non-production environments. It should include infrastructure related as well as 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 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.

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

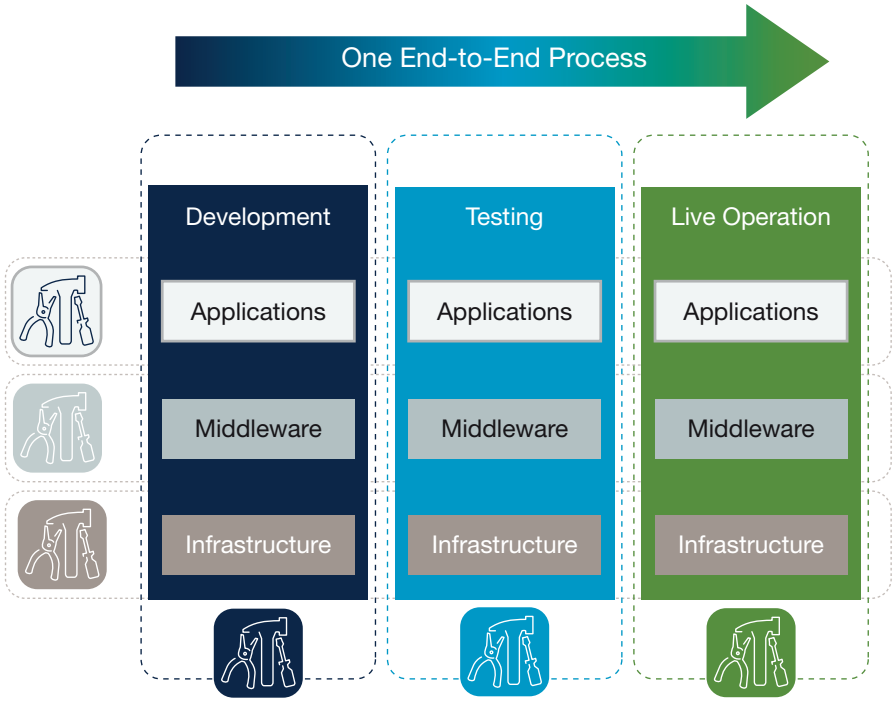


Figure 10: Common Tooling Approach

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

Gartner during April 2014, published a quick overview of the “Cool Vendors in DevOps” – see [1]. 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.

4. DevOps Implementation

As outlined in 1.1 the adoption of DevOps is hampered by a number of key aspects[6]:

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]
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 [2]
3. Each DevOps implementation is unique and every customer requires a customized approach [2]

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 is to:

Three steps for a successful DevOps implementation:

1. Define a target
2. Establish a plan
3. Manage plan execution

Following are the key recommendations:

1. Define a clear target
2. Establish a clear transformation plan and
3. 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 rationale 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' 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.

4.1 Define a clear Target

DevOps claims to reduce impact of changes to reduce cost and minimise 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.

[6] As per Gartner's Seven Steps to Start Your DevOps Initiative [2]

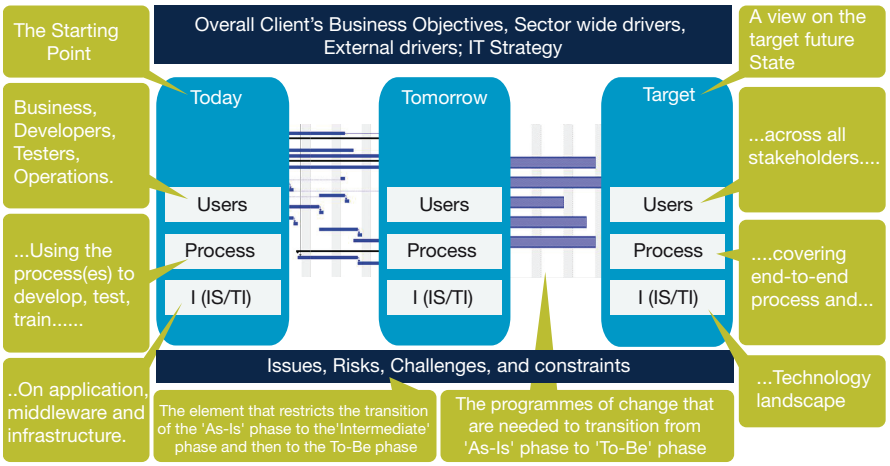


Figure 11 : DevOps Implementation Framework (DIF)

Then, as the "snowflake" point in Gartner's paper[2], 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 "artefacts" needed to define the target and create a business case.

4.2 Establish a clear transformation plan

Once the business case has been 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:

1. Change the culture
2. Establish one Development-to-Operations Lifecycle
3. 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.

4.3 Actively manage the plan execution

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

5. Summary

DevOps is an old technology understood and discussed by a relatively small number of professionals. 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 recommnend 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.

6. Appendix A: References

1. Gartner, Cool Vendors in DevOps, 16th April 2014, G00262716, by Ronni J. Colville, Jim Duggan, Jonah Kowall, Colin Fletcher
2. Gartner, Seven Steps to Start Your DevOps Initiative, 16 September 2014, G00270249, Ronni J. Colville
3. Wikipedia, DevOps, **http://en.wikipedia.org/wiki/DevOps**
4. IT4IT OpenGroup Forum, **http://www.opengroup.org/IT4IT**
5. Gartner, The Virtualisation Scenario: Servers & Beyond, Philip Dawson, 2013
6. The Agile Admin, **http://theagileadmin.com/what-is-devops/**
7. Just enough developed infrastructure, **http://www.jedi.be/blog/2010/02/12/what-is-this-devops-thing-anyway/**
8. Gartner, Hype Cycle for Enterprise Architecture, 22 July 2014, G00261507 by Betsy Burton, Philip Allega
9. Capgemini, TechnoVision 2015, **http://www.capgemini.com/blog/cto-blog/2014/11/welcome-to-technovision-2015**

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

Sameer Mutalik
sameer.mutalik@capgemini.com

Narasimhan PLL
narasimhan.pll@capgemini.com

About Capgemini

With almost 140,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2013 global revenues of EUR 10.1 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want.

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