**Infosys**

**QUALITY SYSTEM DOCUMENTATION**

**References**

**DevOps Guide**

**INFOSYS LIMITED,**

**Bangalore.**

|  |  |  |  |
| --- | --- | --- | --- |
| Document No. : | ITS-GDLN-193 | Version.Rev : | *1.1* |
| Authorized by: | Anoop Kumar | Signature/:  Date | 10 May 2016 |

COPYRIGHT NOTICE

*© 2015 - 2017 Infosys Limited, Bangalore, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/or any named intellectual property rights holders under this document.*

Infosys Limited

Electronic City,

Hosur Road,

Bangalore 560 100

India.

Telephone: 91 80 2852 0261

Fax: 91 80 2852 0362

Website: http://www.infosys.com

Document Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ver #** | **Date** | **Author** | **Reviewer** | **Comments** |
| 1.0 | 28 Jul 2015 | Mugdha Divekar and Anuradha Raghuveer | Philip G, Anupama K, Amit , Naresh, Prasad P, Satya, Adarsh, Dhanashree | Baseline Version |
| 1.1 | 20 May 2016 | ResmipriyaT  Geethanjali N02 | Anoop Kumar | Included points to close gaps with respect to CMMI for Dev V 1.3  Aligned with Infosys DevOps PoV Document |

Table of Contents

1. Purpose 6

2. DevOps Overview 6

2.1. What is DevOps? 6

2.2. Why DevOps? 10

2.3. Key benefits of DevOps 11

2.4. DevOps suitability and qualification criteria 11

2.5. DevOps – A journey of transformation 14

3. Infosys Solution and Key Differentiators 15

4. End to end process for DevOps at Infosys 17

4.1. Propose 19

4.2. Build and Pilot 21

4.3. Execute 23

4.4. Optimize 37

4.5. Consolidated Tools Details 38

5. Key capabilities of DevOps - Detailed approach 40

5.1. Continuous Business Planning 41

5.2. Continuous Integration 44

5.3. Continuous Release and Deploy OR Continuous Delivery 48

5.4. Continuous testing 53

5.5. Continuous Monitoring and Feedback 58

6. DevOps teams 60

6.1. DevOps team in Infosys context 60

6.2. Options for team formation 62

6.3. User stories/work items from Dev and Ops 66

6.4. End to end RACI of dev and ops 66

6.5. Transition challenges to DevOps 67

7. Change management program 69

8. DevOps metrics 72

9. DevOps maturity 78

10. DevOps tool stack 93

11. Myths vs reality of DevOps and its capabilities 93

12. References (used for this document) 94

13. Appendix 95

13.1. Value stream mapping in DevOps 95

13.2. How Lean Principles are used in DevOps 96

13.3. One DevOps team with Ops extension – more details 97

13.4. Typical complementing user stories/work items of dev and ops 98

# Purpose

This document is intended for those who are planning to use DevOps methodology for executing their projects.

This process guide tries to explain the philosophy of DevOps, its capabilities and value, when and where it should be used and the details about how a project can adopt DevOps capabilities to realize the intended business value

# DevOps Overview

In this section, high level overview of DevOps across various facets like – what is DevOps, why is it needed, who should go for it, what is suitability criteria for DevOps, what are key benefits of DevOps are explained

## What is DevOps?

DevOps can be defined in couple of perspectives. One is -

DevOps is a software development methodology that integrates established discipline of IT operation with proven and tested methodology used in development. It stresses communication, collaboration and integration between software developers and Information Technology (IT) operations professionals. They bring in best practices from both the worlds of Dev and Ops on the table. They work as single likeminded team with common goals of high quality software, faster releases and improved customer satisfaction.

Another way of defining DevOps is -

Devops is the operating paradigm facilitating Continuous Release of technology made possible through collaborative, cross functional teams with shared performance metrics tied to real business results, enabled by automation tools and Continuous Delivery practices throughout the entire value stream of delivering software from Idea to Value.

The key principle of DevOps is to drive businesses to see their entire end to end value chain and not just the development aspects. Its principles, based on people and culture, help delivering real valuable software with speed and quality

It is based on strong foundation of 4 pillars

* **Collaboration** – It is not just a collaboration of Dev and Ops team but it also means collaboration across multiple teams in the organization like test, product management, business and leadership. So Business, development, QA , operations and management need to coordinate and play significant role across various life cycle phases. They no longer operate in silos and there are no walls in between the teams. Thus collaboration across all these teams is crucial for successful DevOps implementation.
* **Extreme Automation** – DevOps relies heavily on automation. The Mantra of DevOps is - Automate Everything. Hence there is a critical need for tooling across various phases that will yield end to end automation for software development and deployment. Tools can be built in-house OR bought OR sourced from open source tool repository
* **Agile** - It is based on the core agile principles and practices and thus can be looked at as an extension of Agile methods.
* **Lean** – It adopts 3 key aspects of Lean principles –
  + Optimize the end to end process which can create better value for customer.
  + Identify bottlenecks in the process flow and remove them
  + Identify wasteful activities and remove

(Refer to appendix on more details of *how DevOps uses Lean principles* and usage of *value stream mapping* in defining the roadmap)

*Note - In principle, Devops works on the foundation of Agile and Lean principles. However, for softwares being built using traditional waterfall methodology, early value realization by improving time-to-market can still be achieved to some extent by adopting certain capabilities of Agile / DevOps.*

*There are 2 ways to look at this:*

1. *Development continues to follow traditional waterfall methodology and devops capabilities are only used in making quick releases*
2. *Development methodology takes a journey of transforming from waterfall to Devops and end-to-end devops is embraced*

*In case of former, the production release cycles of software can be considerably reduced by adopting / strengthening release planning and automation techniques. However, the latter is a long term journey involving*

* + - 1. *Cultural shift*
      2. *Adoption of lean and continuous everything principles*

1. *Collaboration & Communication*
2. *Org Change Management*
3. *Senior management sponsorship*

*The methodology described in this document assumes that Agile practices are in place and they are further extended to achieve Devops*

DevOps has following key capabilities. Ideally all these capabilities should be part of DevOps implementation but practically these capabilities are chosen based on the client pain points and requirements. So it is not necessary that each DevOps implementation will have all of these capabilities.

1. **Continuous innovation/business planning** –Our clients are finding the pace of disruption challenging to maintain in this competitive world. To do that in a real sense, the business should focus on enhancing agility, able to quickly react to customer need and feedback. In the traditional approach for product delivery, both the aspect of agility and responding to the customer’s voice gets impacted because of long development cycles and lengthy feedback loops. Often it happens that right feedback is not sought early enough to gain right level of application quality. So to overcome this problem of traditional approaches, Agile/DevOps principles encourage business planning on continuous basis so that newer innovations needed to sustain and grow the business are planned in a continuous manner. Precious feedback from customers and other stakeholders is also plugged into the process of continuous business planning at a much early stage than traditional processes
2. **Continuous integration for collaborative development –** Today’s applications are not monolithic and simple in nature. They typically have multiple platforms, multiple technology and teams spread at multiple locations. Collaborative development helps various team members to work together as it provides a common set of practices and a common platform. Continuous integration (CI) is a key enabler of collaborative development. In this practice, multiple developers continuously OR very frequently integrate their work with rest of the team member’s work using a central (or distributed) single Version control Tool and CI server.The intent of CI is to get early feedback from stakeholders on the quality aspect of recently integrated code. This is possible via shifting testing left and doing it early. It can expose some known and unknown risks of the software delivery early in the cycle
3. **Continuous** **testing –** Continuous testing (CT) refers to the process of continuously doing testing of software which is released to various environment using continuous delivery. All the testing expected in CT is to be done in automated mode only. It starts right from the development environment where test driven development model is used to develop code and automated unit test suites are used frequently to test work-in-progress code.

It is not just creating and running test cases for integration testing, functional testing, performance and security testing. It also enables test environment provisioning and configuration along with the test data management on these environments. Based on the context of the application, the strategy for continuous testing is decided and executed accordingly. It uses concept of service virtualization and thus ensures that virtualized components are used to test the integration aspects much early in the cycle right from development environment. It also uses concept of shift left in which production like environment and data is used much earlier in the phases which help discovering issues early in the cycle.

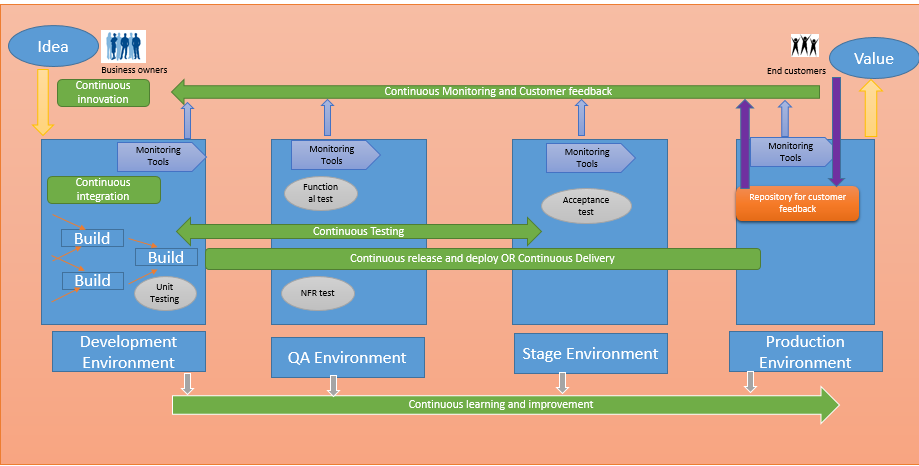
1. **Continuous deploy and release using continuous delivery principles –** Continuously thought through, continuously developed and continuously tested software also needs to be continuously released and deployed on various environments including production environment so as to reap real benefits of DevOps. Continuous deploy and release (CDR) using Continuous Delivery (CD) takes CI to the next level. It operates through delivery pipeline and facilitates the continuous deployment of software to QA and then to production in an efficient and automated manner. The goal of continuous release and deployment is to release new features to customers and users as soon as possible/whenever required. It consist of 2 key capabilities –

* Deployment automation – Deployment automation tools perform orchestrated deployments and track which version is deployed on which environment at any stage of the build and delivery pipeline. They can also help in configuration management of various environments. These tools take care of deployments of software component, database components, middle ware components and configurations, environment configuration
* Release management - Orchestrating the release plans and deployments associated with each release requires coordination across the business, development, QA, and operations teams. Release management tools allow organizations to plan and execute releases, provide a single collaboration portal for all stakeholders in a release, and provide traceability for a release and its components across all stages of the build and delivery pipeline.

1. **Continuous** **monitoring and feedback –** It consists of 2 parts
   * Monitoring feedback -Traditionally we monitor system/application OR environment of only production servers. But DevOps suggests to have continuous monitoring of applications and environments across delivery pipeline so as to get early feedback from these monitoring tools.
   * Customer/stakeholder feedback - Customer feedback comes in different forms through formal and informal channels like – Survey, tickets, bugs, change requests, complaints, ratings etc. Time to time feedback will also come from various stakeholders as well.

So DevOps suggests to have mechanisms to collect these monitoring and stakeholders feedback in continuous manner so that it can be further analysed and actions can be taken to improve on various aspects.

The natural progression of this capability is **Continuous** **learning and improving.** Continuously collected feedback from monitoring mechanism and stakeholder enables teams to learn on continuous basis that what is going well and what needs to be improved. This learning is then converted in improvements of the various aspects on ongoing basis. In this practice the feedback is analysed. Subsequently corrective/preventive action items are identified and implemented and later its effect on actual improvements is tracked. The approach is corrected if the solution was not effective.

All these capabilities can be depicted in integrated manner in the figure below

Details of each of the capability is given in subsequent sections.

## Why DevOps?

Having understood the overview of DevOps, let’s see why we need DevOps capabilities.

Business today demands two key things from software solutions. On one side business wants speed - faster changes through new applications/features, new services generating new or improved revenue streams. Paradoxically they demand these systems to be stable, reliable and high quality with zero downtime. Traditional approaches are not equipped to handle both these requirements simultaneously. So organizations in the past may have needed to prioritise – consciously or by result - one of the requirements of fast delivery OR stable delivery.

Even if we look from the developer and the operation team perspective, the developers’ goal generally is to develop and release software as fast as possible whereas operations team understand that rapid fire changes to production without enough safe guards will destabilize the system and this is surely not in their charter.

DevOps helps to address these 2 contradicting requirements of business which was not easily possible with traditional approaches. It helps to achieve single goal for business, developers and operations. It integrates everyone associated with software development to deployment i.e. business users, developers, architects, testers, operations into a single highly automated way of working which has a common goal of – *Rapid delivery of high quality software that meets users’ needs while maintaining the integrity and stability of system*

These disparate groups subscribe to a common set of principles and work towards common mission. For ex

* + - Maintain short development cycles
    - All teams have same goal and are being measured against that
    - Extremely fast feedback loops for allow almost real time problem identification and resolution

## Key benefits of DevOps

Based on various surveys conducted by groups (such as Puppet Labs - Jez Humble, Annual Sate of DevOps report, various Gartner Quadrant based surveys, and Gene Kim in “DevOps How to Achieve Results”) below are some of the typical benefits organizations have achieved when they adopted DevOps. These benefits are organisation sensitive and will vary in different contexts and are based on peer groups:

* + - Increased frequency of software deployment – by 40-50%
    - Speed of deployment – About 30 times faster
    - Improved quality – Between 30-50% OR half the failures of peers
    - Platform restore after failure– 12 times faster than peers
    - Increased number of customers – 35%

In Infosys context, the perceived benefits of DevOps implementations for our clients are

* + Early time to market with increased frequency of release/deployment- 20-40% improvement than traditional approach
  + Better application quality, reliability and stability in spite of frequent releases – Approximately 10-30% more over other traditional approaches
  + Enhanced satisfaction of end users resulting in increased business for clients – Approximately 30-40%

## DevOps suitability and qualification criteria

Who should go for DevOps implementation? There is no straight forward answer for this question as it depends on various aspects and context. But still, let’s try to see some insights about this point in this section

**Small startup OR mid-size players OR big giants?**

One school of thought is - DevOps is for all and it can *make* or *break* a company of any size.

Other school of thought say - DevOps works only for small “born on net” type of companies OR only for big giants.

So let’s consider the reality. There are big giants like Google/Amazon, where we have seen a huge *make* that DevOps did for these companies.

But also on the other hand, some of the startups and mid-size companies like Github/Etsy are also able to take tough competition with some of these big giants with help of DevOps capabilities.

So essentially, any size organization can opt for a DevOps implementation based on other suitability criteria and their own self-assessment/commitment to change. Non effective implementation of DevOps, on the other hand, can turn out to be a “break” for the organization regardless of its size due to the possible disruptive impact on the culture and the way people actually work (delivery value) in their traditional silo and hand off approach.

**OK. Any size company can adopt DevOps. But what are key necessities it must have for successful DevOps?**

DevOps, at its heart, involves significant people and culture changes, to really drive the collaboration aspects of a transition. This requires significant investment in terms of money and time. So if the company DNA has minimal culture of cross business collaboration then it will take much more time and effort to introduce it successfully, OR in some cases risk potential failure.

If the companies do not have enough investment budget, then the untimely OR inefficient investment in tooling can fail to reap the expected benefits.

Thus ***ready to change mindset*** and ***adequate budget availability*** become key factor of DevOps implementation.

**Can we go with DevOps for all business services in an organization?**

In your organization, there will be few services that always require appropriate/mandatory risk controls and associated processes (for example a system dealing with regulatory compliance). These services may still need traditional approaches as it may need many interventions/approvals in the workflow. So it is OK to have them in traditional approaches and transform towards DevOps only if it is practical and feasible. In the mean time we can get started on other applications which are not so compliance oriented and implement DevOps there.

On achieving certain maturity on overall DevOps implementation, we can go for DevOps implementation in high risk systems.

**Can enterprise with existing ITIL methodologies go for DevOps? Will they get DevOps benefits?**

This again is a topic of debate and there are multiple views and thoughts about the same.

DevOps can be seen as means of process improvement of ITIL. By using ITIL processes as a basis for a DevOps initiative you can greatly simplify and focus your efforts. By adopting a very simple approach as below, you can decide the selection and prioritization of DevOps practices in ITIL environment

* + - Understand the major ITIL processes (say change management, release management etc)
    - Outline what typically goes wrong in its implementation (say issues/defects due to change, manual validation of change, insufficient regression testing)
    - Examine how application of DevOps principles and tools can help to close these issues (for ex - improved collaboration or automation )
    - Prioritize the DevOps practices to be implemented based on above steps
    - Implement and monitor results

**What are the typical criteria for qualifying the projects/programs/portfolio in DevOps?**

Assuming the overall readiness of the organization for DevOps, following are some key considerations to be applied for qualifying projects/programs/portfolio for DevOps. This is just a guideline and not really a hard and fast rule for selection. Based on context, teams can decide to take a go-no go decision.

* **Size** – Sufficiently large team size at least 20-30 people
* **Type** – Fresh development OR application where constant enhancements and bug fixes are done. Application where hardly once in six months release is needed is not a suitable candidate
* **Duration** – Long duration typically 6 months and above
* **Visibility OR business impact**– Typically done for high visibility and high business impact projects after assessing the risk taking ability
* **Release cycle** – Application release cycle need is to be regular and frequent
* **New demands from business and technology** – More the new demands from either business or technology better is the candidate
* **Methodology used** – Agile processes and practices recommended. Waterfall is typically used in long release cycles and we may lose the essence of DevOps implementation in case of waterfall model. Refer to note on waterfall to Devops journey in *Section 2.1*
* **Existing technology and tools** –If it’s a complete replacement of tool chain, then it may not be a good candidate. However few changes/adds in the tool chain should be OK for DevOps
* **Effort per release** – Projects which need significant efforts in one release will be really benefited with introduced automation across delivery pipeline. If this effort is anyways minimum, then you will not see huge difference by introducing DevOps
* **Resistance to change** – If the team’s mind set is not positive for increased collaboration and communication between the teams and there is a strong resistance for this change, then that is not a good candidate for DevOps. Though this appears to be a soft issue, it is the hardest to deal with.
* **Regression test effort in each release** – More effort needed for regression test in each release, more benefits it can get through DevOps due to automation.
* **Dependent systems** – If dependent systems on the applications are large, then perform a risk assessment considering overall integrated applications to decide the suitability for DevOps. All other dependent systems should also comply to suit the processes and release cycle of this application.

## DevOps – A journey of transformation

The most important aspect for DevOps that people should always keep in mind is - : DevOps transition is a journey and not just a process change. It is something that can’t be achieved overnight and teams start getting its benefits from first day. It’s a constant journey of effective implementation. It’s a cultural shift for the teams in the way they ‘work’ – both from technical perspective and non-technical/behavioural aspects as well.

***On non-technical aspects***– With DevOps, the teams now have to essentially remove or at least implement steps to reduce the barriers between the silos that exist in traditionally structured organizations between dev and ops. This segregation of duties has originally been used as a key risk mitigation aspect and process optimisation technique. They need to start doing better communication and collaboration. Their business demands and their goals will also change. The metrics that they were traditionally capturing will change. This means that some HR policies of organizations will also need to be changed. People will need be trained and equipped to work in this mode. So a lot of training and change management programs need to be incorporated.

***On technical aspects*** the complete build, test, deploy and even release cycle can be fully automated. Instead of doing extensive manual testing that focusses on prescriptive and repetitive test case execution for verification, testers can focus on automated testing for all types of testing, as well as more attention to exploratory testing to ensure validity of the software to customer needs. There is a lot of focus on monitoring various environments and collecting various feedbacks on continual basis. There is a complete shift in the way and speed with which environments are provisioned and software is released to various environments in an automated way.

In summation, DevOps is a long journey of transformation that starts from:

* stakeholders deciding to move to DevOps;
* performing readiness assessments;
* doing gap analysis on current and future states;
* defining DevOps roadmaps;
* implementing the basic DevOps operating models;
* establishing change management programs for transition;
* executing in DevOps – it’s better to move ahead with imperfect information and inspect and adapt along the way;
* the people, process and technology matures over time; and
* Finally it runs in most optimized mode with a continuous improvement focus.

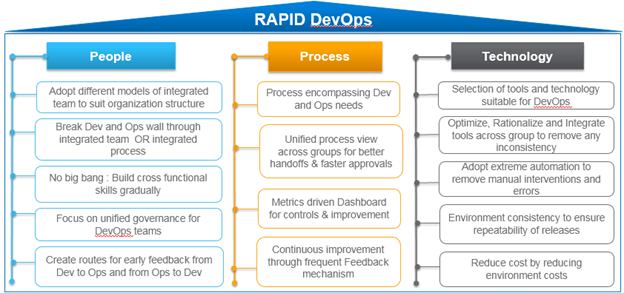
# Infosys Solution and Key Differentiators

Infosys DevOps solution is RAPID – Repeatable, Adaptable, Predictable, Integrated and Dependable.

In order to deliver intended results, it follows the below goals:

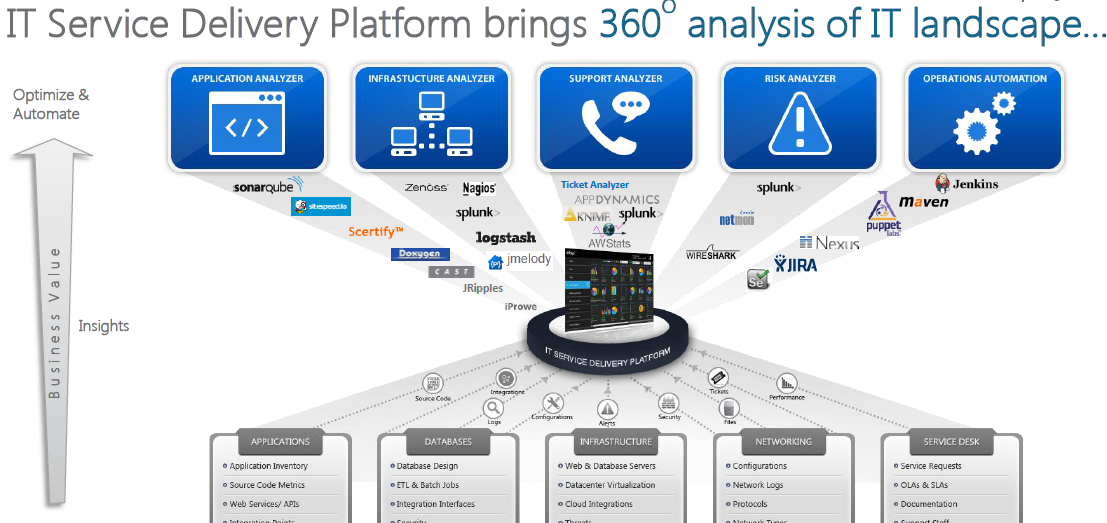
* Ability to build right the first time
* Single KPI and metrics approach across the lifecycle
* Quick response to business requirements ensuring faster time to market
* Reduction in wastage and defects by business and technology teams resulting in improved efficiency

The 3 key pillars – People, Process and Technology creates the base framework for Infosys DevOps.



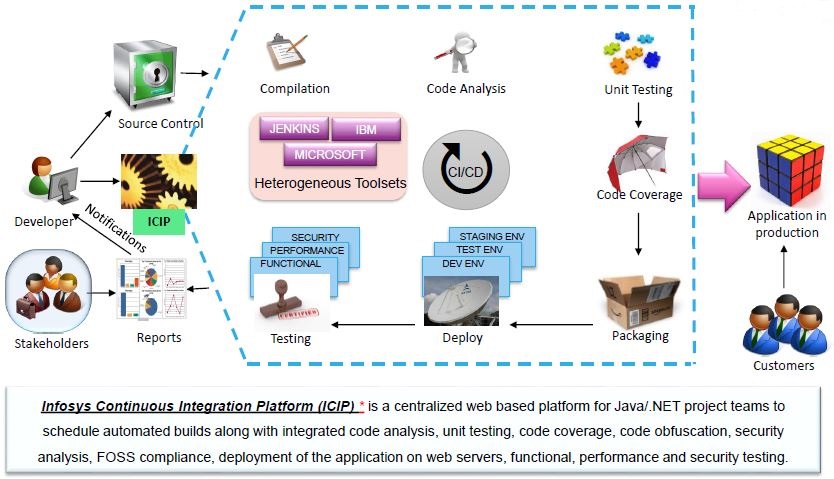
* **Infosys solution** - Refer to <http://teamwiki/DevOps/InfosysSolution> to know more about Infosys DevOps Solution offering
* **Infosys differentiators** –

1. **DevOps Platform and Dashboard** is a dashboard which continuously provides the status of CD pipeline and performance of the IT systems against defined metrics. It integrates with many of the COTS and Infosys products. It is also known as IT Service Delivery Platform (ITSDP).



Please contact - Gouri Shankar Vakkanti ([gouri\_vakkanti@infosys.com](mailto:gouri_vakkanti@infosys.com)) for more details.

1. **Infosys Continuous Integration Platform (ICIP)** – It is centralized web based platform that can be leveraged by project teams working on Java/ .Net to schedule automated builds along with integrated code analysis, unit testing, code coverage, code obfuscation, FOSS compliance, deployment of application on web servers, functional and performance testing.



Please contact Krishna Kanth B. N. ([KrishnaKanth\_BN@infosys.com](mailto:KrishnaKanth_BN@infosys.com)) or

Tapan Munshi ( [Tapan\_Munshi@infosys.com](mailto:Tapan_Munshi@infosys.com)) for further details.

To know more about these differentiators, pls refer to <http://teamwiki/DevOps/DevOpsDifferentiators>.

# End to end process for DevOps at Infosys

DevOps is about enabling rapid delivery of capabilities to the end consumers, which requires collaboration across the software delivery value stream that includes teams spanning across business, development, QA, and infra &ops. This requires embracing a set of capabilities that deliver speed (faster development & frequent releases), quality(stability & resilience in production) and value. Software tools are an essential part of DevOps and the key to orchestrating these capabilities seamlessly across this software delivery value stream

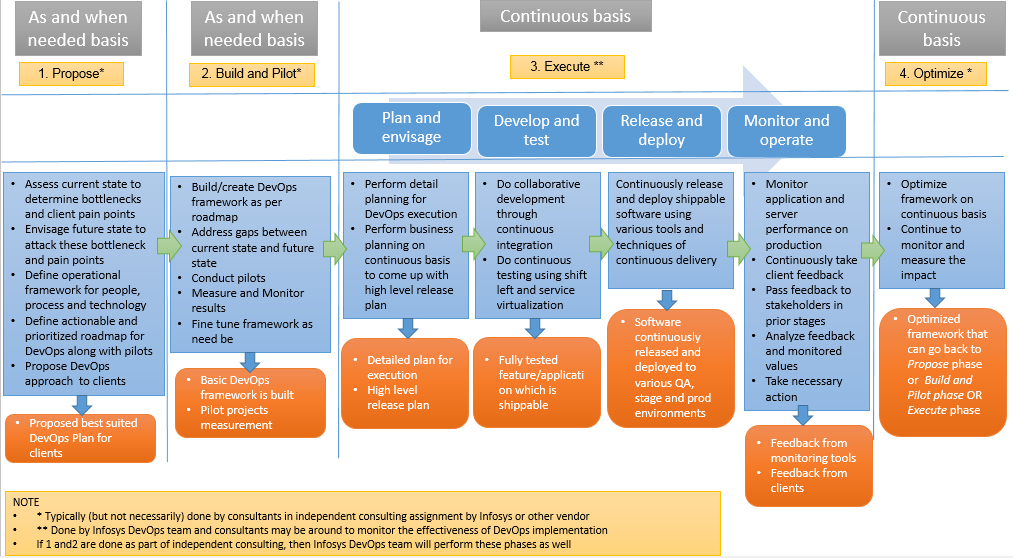
DevOps adoption entails building a set of capabilities across various functions in the organization in terms of people,processes, and tools. This will help IT to stay agile and deliver at the speed which business demands. Infosys believes that there are certain key capabilities that are a necessary minimum for a successful DevOps implementation; these are ‑ Rapid Prototyping, Continuous Integration, Service Virtualization, Functional test automation, Containerization,

Environment Management, Monitoring & Dashboards. As an organization matures in the adoption of these capabilities and practices, additional capabilities can be taken up in a phased manner.

In this section, we will see the overall end to end process that Infosys uses for DevOps implementation for its clients. But it is not necessary that all DevOps engagement at Infosys will have scope for all the phases that are mentioned below. Depending on the scope in the Statement of Work (SOW) OR Letter of engagement (LOE), the respective phases will be executed. Also each engagement will have its own flavour and context for implementing DevOps. Hence this is not a hard and fast prescriptive process that needs to be followed. But this section serves as an overall cook book for how do we go about implementing DevOps. Hence this should be treated more as a guideline for reference. Each engagement can take this as a base and choose to tweak it based on the context of their engagement.

Note: The guideline assumes that Agile/Development/Maintenance and practices are already followed by teams wishing to adopt DevOps

The overall end to end process can be depicted in the diagram below



## Propose

**Overview** - In this phase a proposal is made to client on *DevOps roadmap and implementation. It starts with* performing readiness assessment OR client preparedness assessment, gathering customer pain points and suggesting possible DevOps solution and roadmap. This is typically done as independent consulting engagement. But if it is not performed in that way, then this is done as part of DevOps execution process.

**Entry criteria** –

* There is a willingness from client to go for DevOps and client has engaged Infosys to do this consulting assignment for readiness assessment and roadmap definition
* OR Infosys sees an opportunity to propose client for DevOps framework

**Key activities**

|  |  |
| --- | --- |
| Activity | Additional information/references |
| Consulting team identifies key stakeholders from client end. |  |
| Assess current state and determine bottlenecks/client pain points by having various meetings/workshops with the stakeholders | This is typically done in workshop mode with relevant stakeholders at client side. Multiple techniques like surveys and focused discussions are used to gather the information.  Use Value Stream Mapping (VSM) concepts in this exercise. At its heart, Devops is re-imaging the Value Stream from idea to value, and then proposes change to people, process, tools and overall culture to bring it to life.  For more details on VSM refer to Appendix |
| Brainstorm and put together a future state DevOps solution that will help removing these bottlenecks and pain points | DevOps is essentially a set of capabilities that can together can help you achieving the perceived benefits. But what capabilities need to be chosen purely depends on client pain points and existing challenges. So it is not necessary that all possible capabilities will be chosen and recommended but very likely a subset of relevant capabilities will be prescribed. |
| Do a gap analysis of current state and envisaged future state |  |
| Define operational framework of DevOps for people, process and technology | There are multiple ways in which people and process in DevOps can be defined. So depending upon exact context at client end, appropriate process and people structure recommended.  On technology part also there are multiple capabilities that DevOps offers and each capability can be executed using a variety of tools/technology. The best fit capabilities and associated tool chain is identified that will help client pain points and existing bottlenecks |
| Define actionable and prioritized roadmap with pilots | Based on above, the team defines the DevOps roadmap that is actionable with some defined timelines. ROI is also calculated and presented to client so that client can make a go-no go decision on this roadmap, perceived benefits and possible investment  While coming up with this approach, one should use “Design Thinking” approach of Empathizing/Understanding the Problem Statement; Defining the Problem; and Ideating a Solution |
| Discuss the proposal with client |  |
| Client give OK to the proposal OR can reject | In case of accept, team will proceed to subsequent phases  In case of reject, team will fine tune the approach and resubmit OR the path of DevOps implementation will be closed for the time being |

**Exit criteria** –

* Go ahead from client on proposed DevOps roadmap with approved budgets for investment
* OR project may get stopped at this phase if client’s decision is no.
* OR if Infosys scope is only to propose the roadmap, the engagement will be over after this phase unless we get work order for DevOps implementation
* Organization Value Stream Map for the parts of the company that will be targeted

**Key measurement of the phase**

* Perceived benefits
* ROI of the recommended approach and roadmap
* Effort, money and time taken for completing the phase

**Tools**

None

**Process aids**

|  |  |  |
| --- | --- | --- |
| **Name** | **Reference** | **Remarks** |
| Maturity questionnaire |  | To check the current state and decide roadmap |
| VSM template | <http://172.25.103.244/sites/pride/_Layouts/Pride/prideDocumentDownload.aspx?intDocId=1794> | Helps in identifying bottlenecks in current (as-is) process. This is an important input in defining the to-be state. |
| Gap analysis template |  | This should capture all the gaps identified between to-be state and as-is state across all critical aspects of devops. Additionally, the maturity questionnaire can also be used to identify the gaps between as-is and to-be |
| Design thinking approach |  | DT helps to make sure that the future to-be state is in-line with end user expectations. |

## Build and Pilot

**Overview** - In this phase, based on the approved roadmap, chosen DevOps capabilities are setup. Pilot projects are identified and respective pilot teams are trained on DevOps approach. Pilot projects are executed and measured to see the benefits that it can fetch for the team. Approach is then fine-tuned further based on the learnings from pilots.

**Entry criteria** –

* DevOps roadmap is approved by clients
* Budget approval for investment in DevOps

**Key activities**

|  |  |
| --- | --- |
| Activity | Additional information/references |
| Procure/buy tools that are needed to setup the chosen capabilities of DevOps | Based on the various capabilities and tools agreed, team procures the licenses of the tools which are not available in the organization. This may take some time as it is based on procurement lead time and budget approval |
| Setup the tool chain of deployment pipeline | Deployment pipeline is an almost automated manifestation of the process where the software piece enters from version control and reaches to production servers across multiple stages. So to achieve this automation, tools need to be setup and integrated in a cohesive manner. |
| Create operational procedures, governance structure, RACI that will be applicable in given context | A high level approach will be available with the team from initial phases. If need be they can expand it to make comprehensive operational procedures/RACI etc.  This is typically done in a workshop model. |
| Identify pilot projects for DevOps | Refer to section 2.5 for guidelines on qualification criteria for DevOps projects |
| Identify the team members from Dev and corresponding Ops | Refer to section 6 for more details on team aspects of DevOps |
| Train the team members on chosen DevOps approach and processes | Work with the chosen teams to define their new Way of Working – how is Devops going to be structured for them in their context. The team must focus on Culture and Collaboration – One cannot develop a DevOps model in a box away from the people who will be doing the work. This is the critical juncture between roadmaps, strategy and actual execution. |
| Execute pilot projects for predefined time frame | The projects should be executed in the same approach defined in EXECUTE section below. |
| Measure the results of pilot project and validate the same with expected benefits. | Typical expected benefits is – Quality of application, stability of application, number of deployments done |
| Fine tune the approach if need be | Based on the results and learning of pilot project execution, fine tune the DevOps approach if there is a need |

**Exit criteria** –

* Pilot projects are executed with chosen DevOps approach and capabilities
* Approach is fine-tuned as necessary

**Key measurement of the phase**

* Quality of application in production during pilot phase
* Stability of application in production during pilot phase
* Number of deployments done during pilots

**Tools**

* CI/CD/CM tools chosen

**Process aids**

|  |  |  |
| --- | --- | --- |
| **Name** | **Reference** | **Remarks** |
| Governance model / Structure | <http://172.25.103.244/sites/Pride/_layouts/PRIDE/ProcessDetailsDisplay.aspx?ProcId=910> |  |
| RACI | *Sections 4, 5 and in this document* |  |
| Operational Procedure |  | This document should capture how all applicable capabilities of DevOps will be made operational in the context of the work involved. This should cover technology, people and process aspects |

## Execute

**Overview** – This phase provides details about implementing DevOps in the identified pieces of value stream on ongoing / continuous basis. The scope of implementation can be

* + - application portfolio in single LOB OR
    - application portfolio in multiple LOBs OR
    - enterprise level implementation

This happens in continuous manner where the following phases are executed for various releases across applications in the scope.

The 4 phases of execute are

* + Execute – Plan and envisage
  + Execute – development and test
  + Execute – Release and deploy
  + Execute – Monitor and operate

As mentioned earlier, the earlier phases (i.e. *propose* and *build & pilot*) can be carried out elaborately and in comprehensive way by consultants to define roadmap, setup and pilot. But if it is not the case, a less elaborate process for define roadmap, setup and pilot is carried out by the team in *Plan and Envisage* stage.

### Plan and Envisage

**Overview -** In this phase primarily 2 key things happen –

1. **DevOps detail planning and preparedness** – A detailed planning and setup of DevOps approach for big bang transition is undertaken in this phase. For this prior readiness assessment report for DevOps, Roadmap for DevOps and pilot results may be available with the team as an input from prior consulting engagements. In case it is not available the team comes out with the overall roadmap for DevOps for the unit under the scope and then perform the detailed planning.
2. **Business planning** - The relevant business owners and other key stakeholders form a steering committee. They envisage the business innovations needed for sustenance and growth of the business. The team is also getting the continuous feedback from users and thus have clear idea of what they are looking for from the applications. Based on this, the team comes up with high level release plan for application under purview which will aid these innovations. This release plan later serves as input for further phases of DevOps

This stage needs to be revisited at pre-decided frequency based on various feedbacks and learnings from subsequent phases.

### Entry Criteria

* Teams under scope of DevOps is already following agile methodology
* Agreement of all stakeholders to follow DevOps approach in the defined scope
* DevOps readiness report, approach and roadmap (optional)

### Activities and tasks

|  |  |
| --- | --- |
| Activity | Additional information/references |
| **DevOps detail planning and preparedness** | |
| Identify relevant steering committee members and all other stakeholders |  |
| If above 2 phases (propose and build & pilot) are already executed then understand and validate   * DevOps readiness assessment report * DevOps roadmap and preparedness for the same on people, process and technology front * Pilot project and its results | * In people plan, RACI and way of functioning of DevOps team must have been defined * On process part, we need to check if all the processes are defined and documented well. * In technology plan, validate that the technology stack is available and is setup.   Note – In most of the practical scenario, all these setups may not be readily available. They get built over time and refined continuously |
| If above 2 phases (propose and build & pilot) are not executed, execute all key steps to have   * DevOps readiness assessment report * DevOps roadmap and preparedness for the same on people process and technology front * Pilot project and its results | * In case if these details are not available from prior phases, then the steering team along with DevOps SME need to execute the prior phases and come up with DevOps roadmap for process/people/technology perspective. * Subsequently they will also need to procure and install the technology stack and tryout the complete DevOps setup in a cohesive manner for couple of releases.   For ex –  For CI, if team has chosen ***JenKin***, then they need to procure and install the same to check if its core capabilities are working fine for things like auto build on any commit, auto invoke of various tools and scripts for unit test/code coverage/code quality, automated dashboard for status, automatic alerts for failure in any of the pipeline process so that team can work on it |
| Define resource requirements for projects under scope of DevOps implementation | A detailed plan of - How many dev and ops resources will be needed, what skill levels, what timeframes need to be considered for this |
| Identify properly skilled resources of Dev and Ops teams and form DevOps teams | Though primarily people are identified from dev (developers+designer+testers+) and ops (operation team), there could be other stakeholders in the team like – documentation, business analyst, product management  In most of the cases the dev teams are already existing for the application under purview. So we need to decide more on how Ops teams will be integrated with them and what Ops skills are needed for that particular application. |
| Create and execute change management plan for DevOps | Organization change management (OCM) is an integral part of DevOps adoption, and runs in parallel through all  phases of the adoption journey. OCM includes focusing on redefining the roles and performance measures for  successfully driving the people transformation across dev and ops teams. It also includes defining goals for the  DevOps teams, setting up governance, metrics, creating enablement programs, etc  Please refer to section 6 on more details about change management plan and its execution for DevOps Teams |
| Revisit the DevOps planning process at predefined frequency OR after *optimize* phase to check if it needs any updates/change of approach based on the inputs and feedback from subsequent DevOps releases |  |
| **Continuous business planning** | |
| Brainstorm and finalize various business innovation aspects needed for business sustenance and growth | - Bunch of business innovations needed could be already available with the team and they do not really need to brainstorm and elaborate during these discussion. If they are already available move on to next task - The team can use various techniques, tools and approach suggested by Rami Goldratts while coming up with ideas for business innovation |
| Understand various inputs and feedbacks received from end users and other stakeholders on continuous basis | End users may have some pain points/needs which they are communicating through various channels. Also other stakeholders of application may have certain inputs/ideas to make product and product execution further better and efficient. So we should understand all these feedbacks thoroughly. |
| Analyze various inputs gathered from various end users and stakeholders which will aid in attacking current pain points and issues |  |
| Identify the innovations that can be served through software applications - in existing application portfolio OR need for new applications |  |
| Prioritize the ideas based on ROI, feasibility of implementation, alignment with overall org direction | Identify key requirements that have a strong influence on cost, schedule, performance, or risk |
| Create a release plan based on the inputs gathered in the process | Release plan will typically have – release roadmap, high level schedules and project plans. Though this can be captured and managed in spreadsheets, it is highly recommended that they should be captured and tracked in proper release management tools like BMC release process manager OR UrbanCode Release  So that going ahead the status of all the releases under development can be tracked at the click of button  To know about the different Release Management Tools,refer Section 4.5 Consolidated Tools Details |
| Review, rework and baseline of high level release plan for this iteration of business planning |  |
| Repeat the process of continuous business planning on a predefined time interval as it’s a continuous process (based on your continuous feedback/changing business needs) |  |

**Exit Criteria**

* Detail Devops planning and setup is done
* All teams are enabled on DevOps
* The teams Way of Working has been defined and agreed by all stakeholders across silos
* High level release plans and schedules are ready

**Key measurement of the phase**

* Number of releases that are planned for release in a month/quarter

**Tools**

* BMC release process manager OR UrbanCode Release

**Process aids**

None

### Develop and Test

**Overview -** In this phase the DevOps team get high level release plan from steering committee from the earlier stage of *plan and envisage*. This plan also has inputs/reviews from customer on continual basis for bugs, change requests, new features, complaints etc. while using the application. The key stories/work items of release backlog are identified based on these inputs. The software is built and integrated continuously using CI tools. There is continuous testing of software using extreme automation and service virtualization technology on various environments.

### Entry Criteria

* DevOps setup is ready and tested
* High level release plan is available
* DevOps Teams are formed and enabled on DevOps execution methodology

|  |  |
| --- | --- |
| Activity | Additional information/references |
| Conduct kick off meeting with the DevOps team | In this kickoff meeting DevOps team organizes themselves, discusses the project vision, scope of work and key success factors with the product owner. The high level planning for the project is discussed for various aspects like Sprint duration, Definition of Done and Release Frequency, release management plan is finalized and agreed upon with the Product Owner.  In case of DevOps, it is ensured that ops inputs are always sought at appropriate intervals and activities. There is good amount of handshake and exchange of information between Dev and Ops resources. They have clearly articulated user stories/work items that they need to fulfil for successful implementation of DevOps  A fundamental requirement of DevOps is that the Ops team is continuously engaged with the development team throughout the life cycle of solution development. Ops should participate right from the visioning stage to understand the business vision, the epics, and the release time lines. They should also contribute to determining the solution's technical and schedule feasibility.  From the visioning stage through the development stage, the Ops team should provide the necessary inputs to the development team in order for them to build and validate the Ops-related requirements. For example they can provide relevant inputs on NFR, infra provisioning, platform specific inputs, security aspects, firewall requirements, remote access constraint, inputs for backup and restore, inputs on BCP DR etc.  Also this phase is based on agile processes. Please refer to Infosys Agile processes for further details on agile process base. |
| Elicit the release requirements | Teams will know high level release requirements. But it needs to be elaborated wherever necessary – especially for new features OR major changes in existing feature. Both development team and operations team give inputs in the release requirement. Hence both teams are aware what is needed by both the teams.  This is typically done in requirements workshop mode  Ensure that the requirements captured covers interface , data privacy and security, archival, backup, access related requirements as well |
| Create product/release backlog | Team will already know the high level features that they need to release. But those high level features will still need to be converted into next level epic and user stories/work items to form a backlog.  Product/release backlog also need to have ops stories (as applicable) and not just dev stories i.e. functional and non-functional requirements |
| Review the requirements and product backlogs | Peer review-Single person Review/Group Review cane be conducted for verification |
| Prioritize the user stories/work item in the product backlog for current release |  |
| Estimate the size, effort and release timeline | So that we get fair idea which features/fixes can be released by which date.  Ops also give their inputs on size and effort needed by them and feasibility to release date. |
| Create high level architecture and design | It is created considering ops needs also in mind. For ex – some specific load balancer specific input is given by ops team OR firewall related input given, then the team need to check what aspects of it need to be implemented in design and architecture  Also ensure that interface criteria are considered in design |
| Review the architecture and design | Peer review-Single person Review/Group Review cane be conducted for verification |
| Create test strategy | In all probability, high level test strategy is already in place during prior phases. But still it needs to be revisited based on individual project needs. Testers need to strategize about following aspects to achieve continuous testing in given project context   * Test environment provisioning and configuration * Test data management * Test integration, function, performance, and security   The testing process will vary across various projects and application based on the application context and SLA.  For ex – Customer facing systems may need more security testing than internal systems  Projects where releases are too frequent Vs less frequent, strategy for test data management and test environment provisioning will change |
| Define Done | Ensure that ops aspect are also covered in DoD. Along with the standard coding, testing, and documentation elements, validation of the code in the deployment platform (e.g., a mock production box), specific support instructions as part of the documentation, and a dry run of these instructions should also be included in the Definition of Done. |
| Identify the key risks associated with the project and its associated mitigation plan |  |
| Obtain project plan agreement with relevant stakeholders | Put all above elements in a project plan and obtain agreement of all stakeholders |
| Select product backlog items for sprint planning | So far it was planning done from release perspective. But from here on team gets into breaking the release in further sprints mode and each sprint focuses on releasing something from the product backlog.  So from the list of product backlog/release backlog pick up the stories/work items that will get developed in this sprint. These will be from feature list of dev OR need of Ops |
| Identify tasks and estimate effort | As applicable and needed  Include supporting infrastructure needs when estimating effort and cost |
| Update release timeline and product backlog | As applicable and needed |
| Ensure the team understand the plan for the sprint | Ensure that all team members clearly understand the sprint plan |
| Elaborate architecture and high level design | As applicable and needed. For small changes and bug fixes which are not touching/changing any major component of design, this step will not be done |
| Prepare test cases and automation test scripts | * Based on the test strategy, testers will create all automation test scripts that are needed for functional test and other NFR tests like performance test. * They will also consider the inputs given by ops for the release. * They will be continuously updating the test scripts to add regression tests and additional functional test cases as and when new things are getting developed by the developer and getting committed to the configuration management system. * Progressive way of test automation is highly recommended in DevOps as it only can help you to test the features that are getting developed as we need to test the features on every commit/build |
| Create scripts for automated build | Typically in CI, a build is triggered on commit of a new piece of code. So scripts are required to trigger these build scripts.  To know about the different Build Automation Tools, refer Section 4.5 Consolidated Tools Details |
| Develop the code and automated unit test suite | * It is recommended to write code in TDD, BDD manner. With this the CT aspects gets started write from the point when developer is writing code. * Also developers can use concept of feature toggle for a long effort feature that cannot be released in single release |
| Perform unit testing of the code | This is typically automated but developer can definitely do some amount of manual testing on his own before submitting code to CM tool. Also if possible service virtualization concept can be used to check the integration/communication with other components in the dev environment itself.  To know about the different Unit Testing Tools,refer Section 4.5 Consolidated Tools Details |
| Check in the code in version control system for continuous integration | In the case of complex systems made up of multiple systems or services, developers regularly integrate their work with other systems and services too. Regular integration of results leads to early discovery and exposure of integration risks. In complex systems, it also exposes known and unknown risks — both technical and schedule-related  To know about the different Version Control Tools,refer Section 4.5 Consolidated Tools Details |
| Ensure and monitor that CI tool automatically invokes the configured code quality tool, unit testing and code coverage tools | This really depends on your CI setup and associated tools/quality gates that are implemented at CI level.  To know about the different Code Quality Tools,refer Section 4.5 Consolidated Tools Details |
| Rework to remove any issues that are arising for code quality, unit testing coverage and re-commit the code |  |
| Ensure and monitor that CI tool automatically invokes build script if committed code passes the above quality gate | Automatic build to be triggered as per pre-decided frequency OR on each successful commit |
| Ensure and monitor that CI tool automatically stores the binaries to package repository after successful build | Package repository - A package repository (also referred to as an asset repository or artifact repository) is a common storage mechanism for the binaries created during the build stage. These repositories also need to store the assets associated with the binaries to facilitate their deployment, such as configuration files, infrastructure-as-code files, and deployment scripts. |
| Do infra provisioning for testing | Continuous provisioning of various environments and deployment of software are part as continuous delivery  It essentially uses 2 principles  **Infrastructure automation** – Infrastructure automation automates environment setup & management. It is a process of describing your infrastructure and its configuration as a script or set of scripts so that environments can be replicated quickly and efficiently. By scripting environments and saving them as a code in configuration management system, you can apply the same configuration to multiple instances as well. This technique is also referred as “infra as a code”  **Deployment Automation** – It essentially means automating the application deployment in a fast, repeatable and consistent way through all environments of SDLC. It works on the principle *of Build once, deploy everywhere*. It also has validation in-built to ensure that deployments are successful by means of smoke test. It provides hassle free roll back in case of deployment failure.  Please refer to continuous delivery and continuous testing section in the document below  This step uses infra as a code principle |
| Ensure that provisioned infra and environment is loaded with appropriate test data and automated test suits (as applicable) | This step uses deployment automation principle |
| Deploy the binaries to provisioned infra in automated mode using continuous release and deployment principle and tools | This step uses deployment automation principle |
| Use service virtualization technique to ensure that the integrated components are also getting tested during QA | Pls refer to section 5 below of continuous testing (CT) to know more of service virtualization  To know about the different Service Virtualization Tools, Refer Section 4.5 Consolidated Tools Details |
| Ensure that CI tool automatically invoke the scripts for regression testing, functional testing and performance testing for each build (as applicable and is designed in the CT) | Depending on delivery pipeline, you may have different environments for functional testing, performance testing, UAT, pre-production. It is also possible that there are common environments getting used for couple of types of testing.  Depending upon the deployment pipeline, we need to do infra provisioning, test data and scripts loading and running the test scripts on various environments.  To know about the different Performance Testing Tools,refer Section 4.5 Consolidated Tools Details |
| Perform manual/exploratory testing as applicable | In DevOps most of the testing happens in automated way. But still there could be some features which needs to be tested manually and in exploratory way. So one should spend enough time in doing so in the QA environments. |
| Log defects discovered in various types of testing |  |
| Track defects to closure and recommit the updated code |  |
| Monitor various development and QA environments (as recommended by Ops team) | Based on the monitoring results on these environments, appropriate corrective/preventive action can be taken up by respective stakeholders |
| Track daily progress | Daily progress is tracked through various dashboard reports/alerts from CI components and also through daily stand up meeting.  Understand updates from Dev and Ops  Understand issues faced by them so far and how can we overcome them |
| Conduct mid sprint demos to product owner (As Applicable) |  |
| Introspect the sprint gone by (as applicable) | In this phase, the team looks at the sprint in terms of what went wrong and what was fine – again both from dev and ops perspective  Take the learnings forward in subsequent sprints |
| Repeat sprints as applicable |  |

**Exit criteria** –

* Thoroughly tested and potentially shippable product features are available for final deployment

**Key measurement of the phase (Metrics for build practices)**

|  |  |
| --- | --- |
| Total time taken for the build | Track build processing time which includes compilation, inspecting, testing and deployment  -Helps in making builds more efficient for faster execution & early feedback |
| Unit test coverage | Track efficiency of test driven development  -Helps ensure code quality |
| Unit test success rate |
| Rule compliance/violations | Track non-compliance against set coding conventions  -Helps developers write better quality code |
| Code complexity | Track code documentation compliance  -Helps developers provide better understanding of cod |
| Duplicate lines of codes |
| Percentage of API documented |

**Tools**

* CI/CD/CT tool stack

**Process aids**

None

### Release and Deploy

**Overview –**

The goal of continuous release and deployment is to release new features to customers and users as soon as possible OR whenever it is needed. One should not unnecessarily wait for release to happen only on certain pre-decided release date or at certain pre-decided frequency – say 5th day of every quarter. . In this phase the continuously developed, tested and potentially shippable software is released to production environment as per the business need. Continuous release and deployment to various QA and staging environments is already happening in prior phase. But in this phase it gets deployed in production using the same principle.

### Entry Criteria

* Ready to ship application/features on pre prod environments

|  |  |
| --- | --- |
| Activity | Additional information/references |
| Understand the releases to be made and associated details as per the release management plan | Orchestrating the release plans and deployments associated with each release requires coordination across the business, development, QA, and operations teams.  Release management tools allow organizations to plan and execute releases, provide a single collaboration portal for all stakeholders in a release, and provide traceability for a release and its components across all stages of the build and delivery pipeline.  Ensure that a bidirectional traceability of requirements is maintained.  [Refer Section 4.5 Consolidated Tools Details](#_Consolidated_Tools_Details) |
| Make production environment ready using "Infrastructure as a code" concept | Please refer to section 5.3 for more on infrastructure as a code. |
| Create automated deployment and roll back scripts for deploying the software production using appropriate automation tools. | Deployment automation tools are the core tools in the DevOps space. Such tools perform orchestrated deployments and track which version is deployed on which node at any stage of the build and delivery pipeline. They can also manage the configurations of the environments of all the stages to which the application components must be deployed.  Deployment automation tools manage the software components that get deployed, the middleware components and middleware configurations that need to be updated, the database components that need to be changed, and the configuration changes to the environments to which these components are to be deployed. These tools also capture and automate the processes to carry out these deployments and configuration changes.  To know about the different tools for CD ,CI and Release Management Tools, [Refer Section 1.5 Consolidated Tools Details](#_Consolidated_Tools_Details) |
| Ensure that the release deployment is successful and is communicated to relevant stakeholders by triggered mails |  |
| Ensure successful roll back to previous version in case of deployment failure and communication to relevant stakeholders by triggered mails | This should be done so smoothly that end user should be minimally impacted with this failed attempt of release on production. They should be very quickly able to perform the activities that they were able to do before the failed release. |
| Introspect the release to see what went right and what went wrong for the entire process |  |
| Create a plan for implementing improvements in the subsequent releases. |  |
| Prepare retrospection report for future reference. |  |

**Exit criteria** –

* Software changes are successfully implemented on production server OR
* Software changes are successfully rolled back to previous version in case of some issues in the newer implementation

**Key measurement of the phase**

* Number of successful releases to production in a week/month
* Release date adherence percentage
* Percentage increase in the number of releases as compared to pre DevOps scenario
* Lead time taken for release to production

**Process aids**

None

### Monitor and Operate

**Overview -** In this phase the deployed software and environment are continuously monitored for performance, stability and reliability through various monitoring mechanisms. This is to ensure that software systems and the environment are operating at the optimal level. Many a times the Ops team works with the Dev team to build self-monitoring or analytics gathering capabilities into the software being built. This would allow for true end-to-end monitoring continuously across environments. So though the monitoring will happen predominantly at production environment, as per DevOps principle this will also happen at dev, QA and staging stages as necessary.

In this phase client feedback is also continuously sought through various channels like – usage pattern, tickets, change requests, formal/informal complaints etc.

Output of continuous monitoring and client feedback is passed back to relevant teams for them to further analyse the feedback and take subsequent preventive/corrective action on the same. The teams continuously learn from these inputs and continuously improve their plans and processes.

### Entry Criteria

* Software/Features deployed in production environment and customers are using it (this is also applicable when software are deployed to other environments)
* Monitoring tools are setup and working
* Mechanisms in place to seek continuous feedback from customers

|  |  |
| --- | --- |
| Activity | Additional information/references |
| Continuously monitor the production environment using various monitoring tools | Continuous monitoring provides data and metrics to operations, QA, development, lines-of-business personnel, and other stakeholders about applications at different stages of the delivery cycle.  These metrics aren’t limited to production. Such metrics allow stakeholders to react by enhancing or changing the features being delivered and/or the business plans required to deliver them.  Feedback also comes from monitoring data. This data comes from the servers running the application; from Development, QA, and Production; or from metrics tools embedded in the application that capture user actions |
| Continuously take inputs from customers - based on usage pattern and feedback that those customers provide upon using the application | Customer feedback comes in different forms - such as tickets opened by customers, formal change requests, informal complaints, and application rating. These have to be continuously collected and analyzed |
| Analyze both monitoring output/result and customer feedback by respective stakeholders |  |
| Pass it back to respective stakeholders to take corrective actions for subsequent releases to achieve continuous improvement and optimization | Feedback can be included in the teams Work Item backlog for prioritization of future dev effort  Various teams will have various actions to be taken because of this feedback.  For ex - Lines of business may adjust their business plans, development may adjust the capabilities it delivers, and operations may enhance the environment in which the application is deployed |
| Document the feedback received in continuous monitoring process for future references and learning |  |

**Exit criteria** –

* Feedback from clients and monitoring tools is analyzed and passed back to relevant stakeholders

**Key measurement of the phase**

* Percentage of NFRs met on production environment – like response time, ability to take concurrent users
* Number of client issues
* Number of outages of software per week
* % Capacity utilization
* % downtime

**Tools**

* Nimsoft, Servicenow

**Process aids**

None

## Optimize

**Overview –**

In this phase the DevOps approach is rigorously observed to take a stock of how things have progressed, what were the issues, what benefits it could fetch etc. This analysis/validation will lead to some actions to further optimize the DevOps approach. It could be for tool stack, the governance process, the collaboration mechanism OR any other aspect of DevOps. So this refined approach can again become input for propose, build & pilot, execute phase of DevOps. Again the results are continuously monitored to check if the perceived benefit of this optimized approach are receiving or not

Continuous optimization is the key in DevOps and is not just limited to a phase. It is applicable in all the previous phases too. But the purpose of having explicit phase of Optimize is to have more structured/formal way of assessing the current maturity and see where all we can improve/optimize it further.

### Entry Criteria

* Projects are getting executed in DevOps mode as a steady state for considerable amount of time (at least 3-6 months)

|  |  |
| --- | --- |
| Activity | Additional information/references |
| Current DevOps approach is validated to check if there are any chances of improvement | Pls refer to Infosys DevOps Maturity Assessment framework |
| Based on the gaps and current issues, suggest further optimization to the approach to bring in further maturity. |  |
| Pass on the optimized approach to either propose OR build or pilot OR execute mode (depending on the context of the organization) |  |
| Implement the changes |  |
| Measure metrics and monitor results |  |
| Revisit this phase on pre decided frequency | If this is not done, chances will be that you are stuck with the approach that you had chosen initially. May be with changed context and technology in market, there is a huge scope for improvement in the way DevOps projects are run. So it’s a must to revisit this phase on some pre decided frequency. |

**Exit criteria** –

* Further optimized approach for DevOps (optional as it is not necessary to have optimized roadmap after performing the analysis and validation of current approach)

**Key measurement of the phase**

* Perceived benefits due to optimized approach

**Tools and Process Aids**

* + Process maturity framework by Infosys

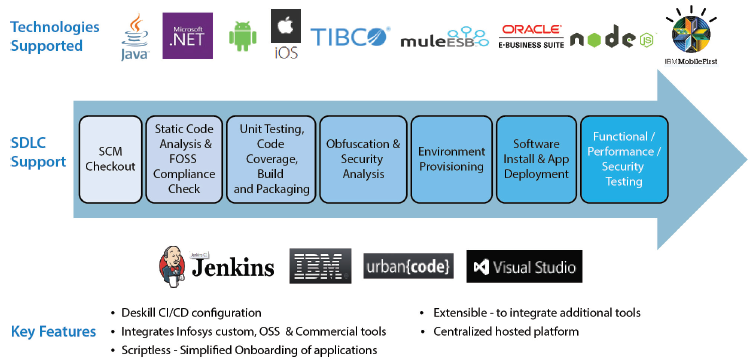
## Consolidated Tools Details

|  |  |
| --- | --- |
| **Stage** | **Tools Details** |
| Application Life Cycle Management Tools | Mingle, Visual Studio Team-Foundation Server, JIRA, RALLY, Rational Team Concert, Rational ReqPro |
| Version Control | Git, CVS, SVN, Visual Studio Team-Foundation Server,IBM Rational Clear Case, PERFORCE, |
| Code Quality | Sonar Qube, JS Hint,FxCop, FORTIFY, PMD, AppScan |
| Code Coverage | JaCoCo, Cobertura,EMMA,NCOVER,Clover |
| Unit Testing | JUnit, KARMA, TestNG, Visual Studio MS Test, NUnit, Jasmine |
| Build Automation | Apache ANT, Maven, Visual Studio MS Build, gradle, IBM Rational Build Forge |
| Artefact Repository | Nexus, Visual Studio Team Foundation Server, Archiva, IBM Rational Asset Manager, artifactory |
| CI Tools | Jenkins, Bamboo, Visual Studio Team Foundation Server, Team City, IBM Rational Collaborative Lifecycle Management, ICDP |
| Database Deploy | LIQUIBASE, dbMaestro, Redgate, Visual Studio, DBDeploy |
| Functional/Acceptance Testing | Selenium, Cucumber, Protractor,FitNesse,Visual Studio Test Professional,IBM Rational Test Workbench |
| Test Management | ZEPHYR, Visual Studio Test Professional, IBM Rational Quality Manager, HP QC |
| Test Data Management | DataFinder, Datamaker, OPTiM,informatica |
| Service Virtualization | SoapUI,IBM Rational Test Virtualisation Server, CA LISA, HP SV |
| Performance Testing | Apache JMeter,Microsoft Visual Studio Team System, IBM Rational Performance Tester, HP LoadRunner |
| Security Testing | AppScan(IBM Rational) , FORTIFY |
| Environment Management | Chef, Docker, Puppet Tabs, ANSIBLE,VAGRANT, HP SA, BMC BL, Bcfg2, CFEngine |
| Infrastructure Management | FlexPod , VBlock, HP MOE |
| CD Automation | Go, UDeploy,Automic UC4 ARA,CA Technologies, Release Automation, HP CDA  Capistrano, ControlTier, Fabric Python Scripts, Ansible, Automic |
| Run Book Automation | HP OO, BMC AO, Cisco Cloupia |
| Release Management | uRelease, Visual Studio Release Management, Automic ARA, CA Technologies, Release Automation |
| Monitoring,reporting and Dashboard | Nagios, Logstash, Splunk,Dynatrace,Visual Studio Application Insights, IBM Rational APM, HP Sitescope |

### Infosys Continuous Delivery Platform

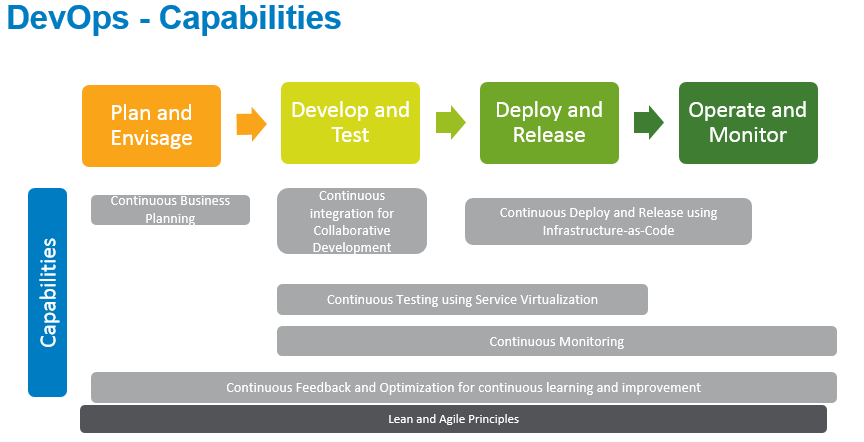
This platforms helps rapid on boarding of applications on continuous

delivery pipeline. This can help significantly reduce the implementation time in a varied tooling landscape



# Key capabilities of DevOps - Detailed approach

In this section we will understand the detailed approach for various capabilities of DevOps on *continuous everything* aspects. The diagram below denotes the key capabilities of DevOps along with its typical spread.

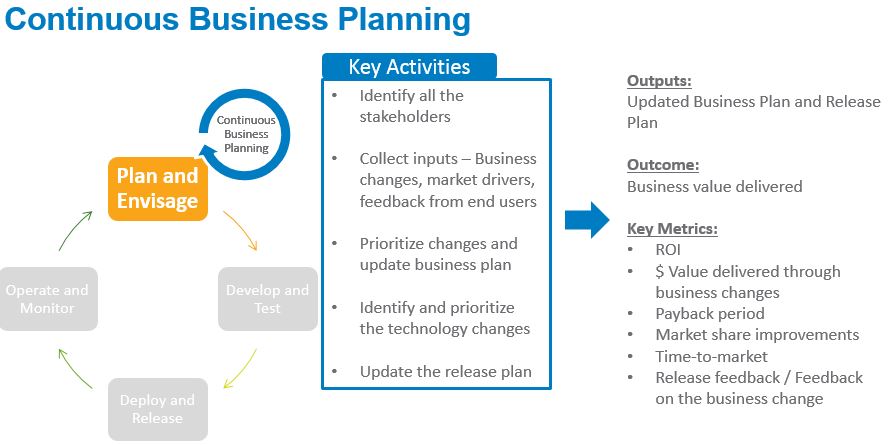


## Continuous Business Planning

One of the key capabilities of DevOps is to enable continuous business planning (CBP). This capability adopts lean principles and provides agility to quickly align the business plan on a continuous basis to cater to current business needs, changing market conditions and also to address continuous feedback received from end users through various channels.

This encourages continuous innovations in all aspects like people, process and technology to aid overall business sustenance. This is usually carried out during the “Plan and Envisage” phase of DevOps.

The diagram below depicts the overall flow of CBP.



**Key Activities:**

1. Identify all the stakeholders
2. Collect inputs based on business needs, market drivers and feedback from end users
3. Identify improvements, innovations, changes etc. required to business plans based on the above inputs
4. Prioritize changes based on the ROI and the business need
5. Update business plan to include the prioritized changes and take approval
6. Identify and prioritize the technology changes required to meet the business plan
7. Update the release plan\* on a continuous basis based on the revised business plan

**\*Key considerations:**

The release plan, which is an output of this capability, should not only include high level software functionalities, but also should cater to the requirements from operations team (for e.g.: server decommissioning, virtualization tools, monitoring features in application, tools for continuous integration etc.), testing team (for e.g.: test data generation, automated test scripts, tools for testing etc.) and overall business/IT requirements (for e.g.: compliance related changes, collaboration tool related, building a particular DevOps capability, architectural changes, rollout in a particular region, integration with a third party package etc.)

The scope, timelines and resource requirements should be broadly specified for all the items identified in the release plan.

Thus the difference between normal water-fall based release planning and DevOps based business planning is

* Continuous planning in DevOps
* Continuous feedback and monitored parameters are getting considered in DevOps
* Typical ops and testers user stories/work items are also getting considered
* Move from up front planning to adaptive planning

**Outcome**: The key outcome of this capability will be to ensure Business value delivered through changes

**Outputs**: Updated and approved business plan and release plan are the key outputs. This plan should be dynamic, estimated, prioritized and able to be changed when new priorities arise.

**Key Metrics (Contextual to the business)**

* ROI
* $ Value delivered through business changes
* Payback period
* Market share improvements
* Time-to-market
* Release feedback / Feedback on the business change

**References**

* Agile Practices
* Lean/Kanban practices
* Portfolio management/Program Management / Project Management
* Business Plan and Release Plan Templates

**Tools**

* Various release management tool
* Agile project management tools

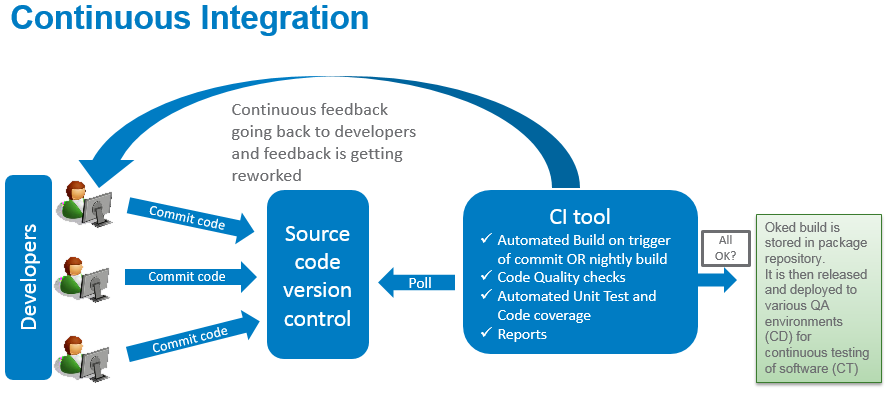
**High level RACI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Key Activity** | **Business Owner (s)** | **Program / Portfolio Manager** | **Sponsor** | **Dev Team Leads** | **Ops Team Leads** | **Test Team Leads** |
| 1. Identify all the stakeholders | C | A, R | S | C | C | C |
| 1. Collect inputs based on business needs, market drivers and feedback from end users | C | A, R |  | C | C | C |
| 1. Identify improvements, innovations, changes etc. required to business plans based on the above inputs | C,S | A, R |  | C,S | C,S | C,S |
| 1. Prioritize changes based on the ROI and the business need | C,S | A,R | C,S | C,S | C,S | C,S |
| 1. Update business plan to include the prioritized changes and take approval | C | A,R | S | I | I | I |
| 1. Identify and prioritize the technology changes required to meet the business plan | I | A | I | R | R | R |
| 1. Update the release plan on a continuous basis based on the revised business plan | I | A,R | I | C,S | C,S | C,S |

## Continuous Integration

Continuous Integration (CI) is one of the core capabilities of collaborative development in DevOps. Many times our customers OR team members are mistaken that successful implementation of CI means successful implementation of DevOps. But it only means that one of the capabilities of DevOps is implemented successfully and not the complete DevOps.

The figure below depicts the overall implementation of CI



In a DevOps scenario, multiple development teams/developers develop their respective functionalities and continuously integrate their work with other teams using a continuous integration platform.

By doing this, developers get an early feedback on the issues or the problems in the software when integrated with other components that needs attention from respective teams. This avoids last minute surprises and ensures that the software being released into production works as expected.

Continuous Integration is achieved with the help of tools and is completely an automated process. Most of the time it is a productized process where in the next stage cannot be triggered unless the previous activity completes successfully. It issues appropriate warnings and errors in case of any issues in the assembly line of the software creation. Thus the selection of tool becomes one of the key steps in ensuring required outcome of this capability.

The selection of tool further depends on the technology landscape and the architecture on which the underlying portfolio of software and its environment is built. All these aspects are considered while defining the roadmap and selecting CI tool.

The workflow in a CI setup is usually configured to take care of code compilation, code quality checks, automated unit testing, build, packaging, deployment, report generation etc. Appropriate processes are created around this and people have to use them with discipline and rigor.

Though “Continuous Integration”, as an activity primarily happens only during “Develop and Test” phase, planning for the same happens during “Plan and envisage” phase. The selection of the tools and the configuration / setup of the same as per the requirements defined in the release plan becomes a part of the “Plan and Envisage” phase OR even in earlier phase of Propose.

The build is moved to a *package repository*. A package repository is a common storage mechanism for the binaries created during the build stage. These repositories also need to store the assets associated with the binaries to facilitate their deployment, such as configuration files, infrastructure-as-code files, and deployment scripts.

The natural progression of CI capability is Continuous Delivery (CD) and Continuous testing (CT) without which the real benefits of DevOps cannot be achieved.

Some of the **key activities** that are required to build this capability are:

1. Understand the plan, scope and requirements of CI platform from the release plan and Sprint plan user stories/work items
2. Understand the technology landscape and the architecture of the underlying software
3. Perform gap-analysis to come up with list of recommended tools and their configuration for the given technology areas and architecture or understand the same from the gap-analysis report and recommended technical solutions
4. Procure the CI tools and provision for the infrastructure as per the requirements defined in Sprint Plan
5. Setup the CI platform as per the Sprint plan user story. For Eg: Platform can have an automated workflow for performing Quality checks, code coverage, testing, build and packaging, deployment, monitoring and reporting, regression, functional, performance and security testing of the checked-in code. The workflow can be setup for automated rollbacks and triggers as per a defined criteria
6. Continuously monitor the CI platform and ensure feedback is provided to all stakeholders regarding any issues either in software or in the environment

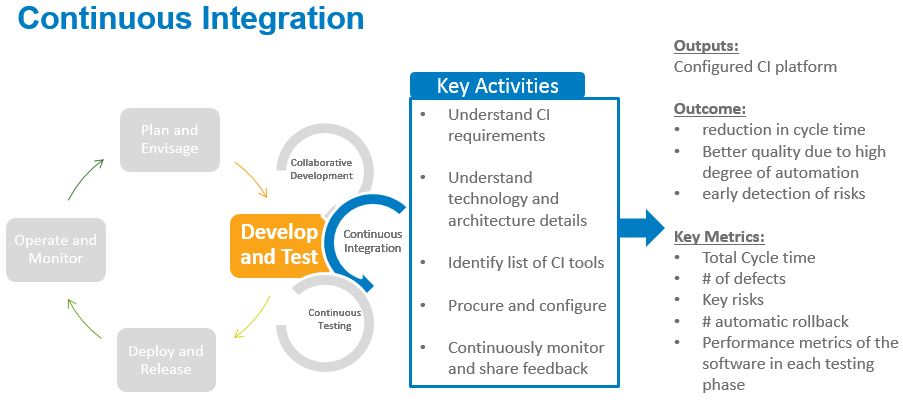
The technology stack can be obtained from: <http://teamwiki/DevOps/CICDDetails>

The **output** from this phase is the configured and used CI platform as per the roadmap.

The **outcome** is the reduction in cycle time, high degree of automation leading to better quality and early detection of issues/defects/risks related to integration and other software quality checks

**Key metrics**

* Total Cycle time (time taken for automated builds, code coverage, testing etc.)
* Defect density
* # automatic rollback
* Performance metrics of the software in each testing phase
* Effort/cost related metric (refer to agile methodology)

****

**References**

* Agile Practices
* Program Management / Project Management
* Business Plan and Release Plan Templates
* Continuous Delivery Guidelines
* <http://teamwiki/DevOps/CICDDetails>

**Tools**

* <http://teamwiki/DevOps/CICDDetails>
* ICIP\*

ICIP\* – This is Infosys IP and is called as “Infosys Continuous Integration Platform (ICIP)”. It is a web based centralized platform for Java and .NET projects.

**High level RACI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Key Activity** | **Business Owner (s)** | **Program / Portfolio Manager** | **Sponsor** | **Dev Team Leads** | **Ops Team Leads** | **Test Team Leads** |
| 1. Understand the plan, scope and requirements of CI platform from the release plan and Sprint plan user stories/work items |  | A,R |  | R | R | R |
| 1. Understand the technology landscape and the architecture of the underlying software |  | A |  | R | R | R |
| 1. Perform gap-analysis to come up with list of recommended tools and their configuration for the given technology areas and architecture or understand the same from the gap-analysis report and recommended technical solutions |  | A |  | R | R | R |
| 1. Procure the CI tools and provision for the infrastructure as per the requirements defined in Sprint Plan |  | A, R | S | I | I | I |
| 1. Setup the CI platform as per the Sprint plan user story. For Eg: Platform can have an automated workflow for performing Quality checks, code coverage, testing, build and packaging, deployment, monitoring and reporting, regression, functional, performance and security testing of the checked-in code. The workflow can be setup for automated rollbacks and triggers as per a defined criteria |  | I |  | C | C | A,R |
| 1. Continuously monitor the CI platform and ensure feedback is provided to all stakeholders regarding any issues either in software or in the environment |  | I |  | I | I | A,R |

## Continuous Release and Deploy OR Continuous Delivery

Continuous release and deploy (CRD) achieved through the principles of Continuous delivery (CD) is an extension of continuous integration where the continuously integrated software is automatically and continuously delivered to various environments for continuous testing. Environments can be like system testing, performance testing, staging and Production.

Continuous delivery is fully automated and in some cases it can be only till pre-production environments where the pre-production environments exactly mirror the production setup.

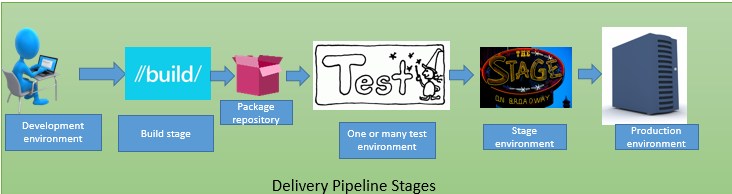
Continuous deployment in multiple environments helps in testing the deployment process itself and reduces the risk of deployment failures in production.

DevOps usually recommends continuous delivery up till production environment. This ensures consistent automated delivery process across all stages, making it more efficient.

Many tools available in the market help achieve continuous delivery through automated configuration, testing, sharing of results etc. But continuous delivery does not mean continuous deployment to production. It means that every change is proven to be deployable at any point in time

CD is based on the concept of automating various stages of *delivery pipeline (DP).*

DP means the collection of all the stages that the software goes through from development to deployment on production. The diagram below shows typical elements of DP. But these can vary depending on the organization context OR application context. The level of automation one can bring in DP also varies from context to context i.e. some teams will do it fully automated but some will have some manual checks and intervention (typically for compliance/regulations part) in the DP workflow. So here the mantra is don’t start doing it for all stages, but pick up the DP automation for critical ones and then slowly expand the same.

**

CD consists of 2 major parts

* Automated deployment – In simple term the deployment can be defined as promotion of components from one environment to the next. Thus automating the deployment part is the key in DevOps for speed and accuracy.
* Release management – Release covers the deployment of complete application OR multiple integrated applications to production. Thus managing a release is another big aspect of CDR.

Let us see these 2 aspects in detail

**Automated deployment –**

While designing right deployment process, the team should keep 3 aspects in mind

* **One should be able to use same deployment script across environments**. Practically the requirements and needs for each of the role (i.e. developers, testers and operators) from deployment is different. So they write their own scripts convenient to their needs. But ideally they should be using the same process to deploy software components across environments. And they should start with designing the deployment process for production first and later try to take the same process in earlier environments too. By doing this, the deployment process also gets tested very frequently when the software is getting deployed across environments
* **Automate the deployment process** – Typically deployment process is a huge long list of steps which is very error prone if done semi-automated way with human intervention. It carries a risk of failures if not automated. So if the deployment process is automated it gives a traceability, visibility and auditability of the process in terms of
  + - What components
    - What version
    - Who deployed
    - When deployed
    - Where deployed
* **Deploy incremental changes**- One should deploy only those components which are changed and they should keep the untouched components as it is. Deploying unchanged components all over again increases the risk of failures

While deciding a solution for deployment automation, the team should understand their priority and then choose the solution which typically should provide capabilities like

* Deployment process reusability across environments
* Deployment coordination across environments
* Secured process based on the roles
* Audit trail capabilities
* Should fit in existing set of technologies
* Traceability of who deployed, what components and to which environment

While rolling out the deployment solution team should follow following practices –

* Choose right time and right application candidate for the solution roll out
* Design a production like environment as a starting point of development. So this environment can be small but should have similar OS, middleware, configuration as that of production
* Practice production style deployments in earlier environments too
* Design for production – Production is the most complex environment. So if the deployment process is designed for production, tweaking it somewhat for earlier environments is easily possible.

**Release management**

End to end release management is a crucial activity and needs a lot of coordination. Most of the organization take help of spreadsheets and documents to do the release management and coordination. The problem with this approach is traceability and auditability of who did, when, where, what etc.

So release coordination tools OR release management tools are designed to help the teams in complete end to end release planning and tracking of a release including application components and infrastructure/environments too.

So while selecting a release management tool, one should look for tool capabilities and then take a call which are critical capabilities from the team’s perspective and accordingly choose the tool

Key capabilities a typical release management solution should have

* Progress dashboard
* Change tracking for application components and infrastructure in a release
* Alert and notifications to stakeholders
* Automatic promotion to next environment when the deployment of lower environment is passed through
* Update the release plan on the fly
* Manage automated and manual activities, various relationships in the activities and all related communication in the workflow

Some important tips while implementing the release management solution is

* Release management solution is critical when organization deploy multiple application to production around same time. It’s planning and tracking becomes crucial because of simultaneous deployments at the same time. Even if you may not have automated deployment solution, you can still have a release management solution
* Before going for solution implementation, understand your current set of applications being released, its key features, people involved in release management and complete process of release. Then choose a sample release that can be used as a pilot for release management solution. This sample release should be good representation of typical releases, process, and people involved etc. So that later this can be scaled to more releases
* While trying out the solution first time there can be couple of approaches
  + Run both current way of release management and new solution in parallel for one release
  + Conduct release using current release process and again use the new solution and perform the release once again using the solution
  + Thoroughly understand the solution and use it directly on the release.

**Infrastructure as a code**

One of the challenges in continuous deployment is the deployment of infrastructure itself for various environments. In order to facilitate smooth, quick infrastructure provisioning, infrastructure is treated as a programmable component which can be automatically deployed repeatedly as and when required. The requirements for this are also captured during planning. This is known as “Infrastructure-as-code”.

Infrastructure as code is a core capability of DevOps that allows organizations to manage the scale and the speed with which environments need to be provisioned and configured to enable continuous delivery.

Another upcoming trend is software-defined environments which is a step ahead for Infra as a code. Infrastructure as code deals with capturing node definitions and configurations as code.

Whereas software-defined environments use technologies that define entire systems made up of multiple nodes — not just their configurations, but also their definitions, topologies, roles, relationships, workloads and workload policies, and behavior.

Typically 3 kinds of automation tools are available for managing infrastructure as code:

✓ Application- or middleware-centric tools: These tools usually are capable of managing as code both application servers and the applications that run on them. Such tools are specialized, bundled with libraries of typical automation tasks for the technologies that they support. They can’t perform low-level tasks such as configuring an operating-system (OS) setting, but they can fully automate server and application-level tasks.

✓ Environment and deployment tools: These tools are a new class of tools that have the capability to deploy both the infrastructure configurations and application code.

✓ Generic tools: These tools aren’t specialized for any technology and can be scripted to perform several kinds of tasks, all the way from configuring an OS on a virtual or physical node to configuring firewall ports. They require much more work up front than application- or middleware centric tools do, but they can handle a greater range of tasks

**Key activities carried out in continuous release and deployment are as below:**

1. Prepare release plan (Ensure acceptance criteria is defined) (Refer CBP)
2. Plan and perform release communications and training activities
3. Evaluate deployment and release automation tools
4. Setup / Configure deployment / release automation tools as per the plan
5. Manage deployments for various activities like
   1. Installing applications
   2. Configuring installed applications for the target environment (test/acceptance/production)
   3. Configuring resources
   4. Configuring middleware components and database layer components
   5. Starting/stopping/restarting server and applications
6. Capture quality of deployment and other operational metrics and share with stakeholders
7. Manage release to production using release management solution

**Outcome**: The key outcome of this capability will be to ensure quick and efficient deployment and release of the software on continuous basis. This will help in reduce time to market and to improve efficiency by reducing cycle time of release.

**Outputs**: Software components are deployed to various environment in continuous basis and released to production as and when required.

**Key Metrics (Contextual to the business)**

* ROI of tool investment
* Number of successful deployments and releases

**References**

**Tools**

* Various automated deployment and release management tools like UrbanCode release and UrbanCode deploy

**High level RACI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Key Activity** | **Business Owner (s)** | **Program / Portfolio Manager** | **Sponsor** | **Dev Team Leads** | **Ops Team Leads** | **Test Team Leads** |
| Prepare release plan - Ensure acceptance criteria is defined (Refer CBP) | R, A, | R | R, C | I | I | i |
| Evaluate deployment and release automation tools | I | C, I | C, I | R | R, A, | R |
| Setup / Configure deployment / release automation tools as per the plan | I | I | I | I | R,A | I |
| Manage deployments   * 1. Installing applications   2. Configuring installed applications for the target environment (test/acceptance/production)   3. Configuring resources   4. Configuring middleware components and database layer components   5. Starting/stopping/restarting server and applications | I | I | I | R, C | R,A | I, C |
| Capture quality of deployment and other operational metrics and share with stakeholders | I | I | I | R,C | C | R,A |
| Manage release to production | I | I | I | R, C | R,A | I, C |

## Continuous testing

Continuous testing (CT) is a key capability of DevOps. Continuous testing means testing earlier and continuously across the life cycle, which results in reduced costs, shortened testing cycles and continuous feedback on quality for early refinement.

This can be achieved by means of extreme automated testing and service virtualization/shift left techniques.

It encompasses 3 key factors

* + Test data management to ensure quick and production like test data available for testing
  + Test environment management and provisioning to ensure quick and automated provisioning of various environments with similar configuration as that of production environment
  + Perform application testing for functional, integration and performance aspects on continuous basis

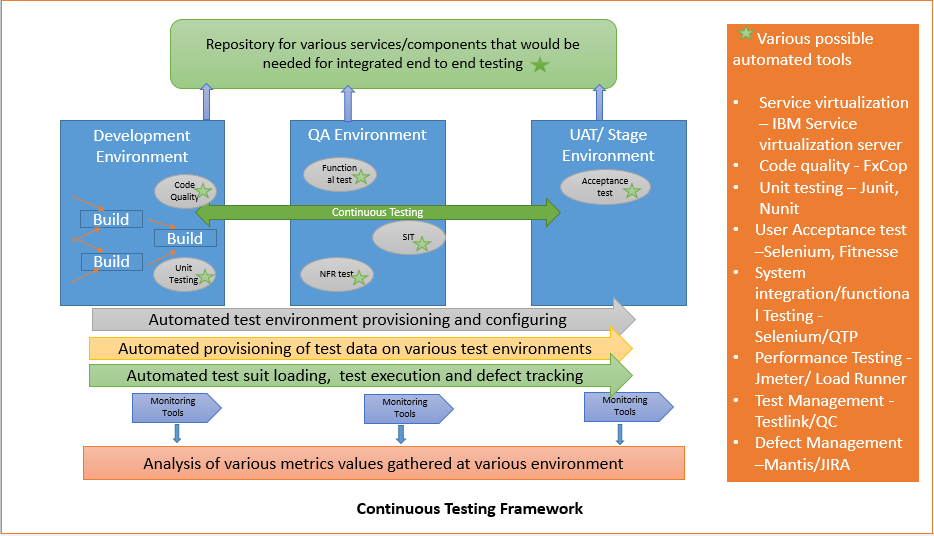
In CT, Testing is not just a mere function of QA. It rather starts from development environment when developers write the code. In DevOps they just can’t throw untested code over the wall to QA for further testing. But developers also need to ensure that their code is passed through quality gates at development end for automated unit testing and code quality aspects. They also use service virtualization technology to test the integration piece of their code with other components. They need to ensure that using test driven development (TDD)/behavioral driven development (BDD) way of development the written code is thoroughly tested during development to demonstrate expected behavior of the system. They also need to do it with test data close to production type test data and environment near to production like environment so that issues are caught earlier. Post these practices that developers can adopt to inspect the quality of application, it then moves to the actual QA function in QA environment for various types of independent testing again on continuous mode after each build.

Speed is crucial in DevOps and typical QA cycle time goes in weeks to complete the SIT and performance testing. Such a long cycle is not what DevOps demands for. It is necessary that each build is tested through all automated test suits like regression test, progressive functional test and performance test. This would help in detecting defects as soon as possible rather than it happening late in the cycle. Thus extreme and progressive automation is essential in DevOps.

Continuous testing enables the ongoing testing and verification of the code written by developers. It also ensures that incremental code produced by various developers which is integrated on continuous basis is getting validated time to time without introducing long delays. Also various other components/services that are needed to test the complete application is made available through **service virtualization techniques**.

Operations function also play an important role in CT. They have access to the load pattern OR usage of software in production environment. If this comes as input to QA team, they can create similar load OR usage pattern in QA environment too. Operations can also let the QA team know about various monitoring aspects that they follow at production server and application. With that the same monitoring aspects can also be implemented at QA servers /application versions as well (wherever appropriate). Operations can participate in testing activities by offering their knowledge and experience in analyzing the data of load/stress/security test of the application. This will ensure early detection and fixing of issues rather than it getting uncovered at production.

The diagram below depicts the overall flow of CT.

****

Some best practices of CT which has helped DevOps teams to get results are

* + Automate repetitive and labor intensive test cases
  + Run automated test suits after every build
  + Estimate and plan the automation work in a separate bucket
  + Use metrics to measure the effectiveness of automation
  + Continuously revisit the automation framework based on metrics and measurement

**Service virtualization (SV)**

SV technology helps in simulating the behavior of various software components with an application so that it can help the end to end testing of the application as a whole even when these components are not available at that point it time. So rather than directly testing the application in integrated way only during SIT or when the other dependent components are ready, the team can now use these virtual services and test the application end to end much early in the cycle. The defects can be detected and fixed early. So that later when the integrated testing with actual components happen, there is very less amount of defects that may be discovered

Typical SV solution should have following characteristics

* Component simulation
* Can be used in various test environments by developers and testers
* Testers and developers can continue using the same testing tools that they were using before SV solution as these tools will not understand the difference between a virtualized service or the real service
* Can be turned on/off when the real service is available/unavailable
* Should be able to respond with realistic data

SV is used predominantly for various types of testing but it can be very well used by development teams and non-production teams.

* *Development teams* – Apart from formal testing being done by an independent team, CT recommends that developer also do the testing of the software at their end. SV helps them performing the integrated testing
* *Testing* *Teams*– This is used at every type of testing starting from unit testing, system testing, integration testing , performance testing, negative testing etc,
* *Non production scenarios* – Typically for training purposes wherein we want production like realistic environment but not production.

**Key benefits of SV are**

* *Cost reduction* – It is an economical option against the cost for test lab infrastructure
* *Improved productivity* – As it can be reused across various stages and it is available 24\*7 without really waiting for real services to be up and running, it definitely helps in improved productivity and cycle time reduction
* *Reduced risks* – As integrated software can be tested much earlier and more frequently, it reduces the risk of discovering something in production
* *Improved quality* – The end quality of software is definitely improved as it has undergone the integration testing in much earlier and more frequently

**Approach for service virtualization is**

* Identify components for service virtualization
* Select vitalization tool
* Calculate ROI
* Build virtual components
* Use it during development and QA activities

**Some tips and best practices of SV are**

* Redesign the testing approach if you team decides to use SV
* Be flexible in the plan of what to virtualize and when
* Developers and testers should use the same SV components rather than creating on their own. So adopt sharing of SV components to the bigger set of people and reuse it appropriately rather than reinventing the wheel

Following are key activities and associated RACI for CT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Key Activity** | **Business Owner (s)** | **Program / Portfolio Manager** | **Sponsor** | **Dev Team Leads** | **Ops Team Leads** | **Test Team Leads** |
| Create test strategy for projects | I | I, A | I | C | C | R, A |
| Install the CT tool setup across various environments | I | I, A | I | I | R, A | R, A |
| Implement various monitoring tools across QA environments | I | I, A | I | I | R,C,A | RA |
| Create various test scripts | I | I, A | I | I | C,I | R,A |
| Perform test data management and ensure production like data is available across dev and QA environments too | I | I, A | I | C,I | C,I | R,A |
| Ensure production like test environments and configurations are created for various types of testing | I | I, A | I | I | R, C, A | R, C |
| Identify SV components | I | I, A | I | C | C | R, A |
| Perform ROI for SV | I | I, A | I | C | C | R, A |
| Install SV tool | I | I, A | I | I | R, A | I |
| Create virtual components | I | I, A | I | C, I | C,I | R, A |
| Deploy them in dev and test environments as appropriate and whenever needed | I | I, A | I | I | R, A | A, C |
| Auto execute various tests by invoking automated test on each build OR each release | I | I, A | I | C, I, A | I | R,A |
| Perform manual testing for the scenarios which are not possible to be done by automated testing (for ex. exploratory testing) | I | I, A | I | C, I, A | I | R,A |
| Trace defects/issues to closure (defects could be functional, non-functional, or issues due to monitoring values) | I | I, A | I | R, A, C, I | I | R, A, C, I |
| Continuously update various test scripts to make them current and relevant | I | I, A | I | I | I | R,A |
| Execute test scripts until it is passed by various QA gates | I | I, A | I | C, I, A | I | R,A |

## Continuous Monitoring and Feedback

Typically monitoring of applications which are in production happens using tools. However, at times they are limited to a particular application and can miss the holistic monitoring of overall portfolio of applications and the infrastructure beneath them.

The DevOps principle around continuous monitoring and feedback emphasizes on “shift-left” approach where continuous monitoring of all the applications using complete automation to cover both functional and non-functional aspects happens right from early life cycle stages and is not just limited to applications in production. Metrics related to quality of both the environment and the applications are captured using appropriate tools continuously and shared with all stakeholders for interventions and corrections.

Other than quality metrics, the end user feedback which is also a key input, is continuously collected using various channels and shared with stakeholders for their action.

The key goals of DevOps is to take feedback, analyse, learn and quickly change the plans to ensure alignment to business and/or end user needs.

This calls for strong collaboration amongst the participating stakeholders to make sure business plans, release plans, project plans and operating environments are altered as per the feedback received and analysed.

One of the key outcomes of this capability is the practice of continuous learning and optimization. The continuous monitoring and feedback received helps in understanding the quality of the software and the infrastructure and also the feedback/comments/reviews from end users/customers and accordingly aids in making required changes to all the plans which helps in overall optimization.

The learnings achieved through this monitoring and feedback means different things to different teams.

For ex –

* Marketing team is able to ensure that enhanced user experience is converting in revenues.
* Business team can see their stores through customer eyes and accordingly enhance the experience
* Services team can reduce the time taken to resolve a ticket.

To support this capability, there is a need to have quantitative and qualitative analytics solution which can capture the behaviour of end users while using the application to uncover hidden issues and opportunities. This data further helps the team to analyse issues and find out the possible improvements to attack the root cause of the problems at wherever possible area. The improvement could be in the software itself OR the process in which we deliver the software OR in the environment.

Various types of monitoring needed in DevOps is as below

* Infrastructure monitoring - Helps you in visualizing events and get alerts/triggers in real time (Tools like Nagios, Zabbix)
* IaaS monitoring - Helps operations to consume usage and performance information (Iaas vendors like Google/Amazon use tools like AQS Cloudwatch)
* Application performance monitoring (APM) – To identify bottlenecks in the app framework (Tools from new relic)
* Monitoring across stack – Nextgen set of tools with deeper integration points, monitoring platform data aggregation, and cross analysis. They also help in quick time to detection and recovery by using data science to automatically correlate the daily flood of alerts, deployments and communications, and turn them into actionable insights. (Tools from BigPanda)

**Key activities**:

1. Identify the monitoring requirements from the release plan for all the environments
2. Identify the tool requirements for monitoring and collaboration
3. Procure the tools and setup / configure as per the requirements to enable continuous monitoring
4. Setup channels to collect end user feedback on the production software
5. Capture the metrics from the tool and share with stakeholders
6. Capture customer/ end user feedback from the channels and share with stakeholders
7. Analyze metrics and customer feedback and make appropriate changes / corrections / optimizations to respective plans

**Output**: Quality metrics (as per the project context) and Customer/End user reviews/feedback

**Outcome**: The key outcome of this capability is to quickly help in making corrections to all plans to ensure early realization of business value

**Key metrics**:

1. # tickets / complaints
2. Customer feedback / review comments / ratings
3. Downtime of application and environment
4. Change Requests

**RACI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Key Activity** | **Business Owner (s)** | **Program / Portfolio Manager** | **Sponsor** | **Dev Team Leads** | **Ops Team Leads** | **Test Team Leads** |
| Identify the monitoring requirements from the release plan for all the environments | I | C | I, A | I, C | R, A, | I, C |
| Identify the tool requirements for monitoring and collaboration | I | I | I | I | R, A, C | I |
| Procure the tools and setup / configure as per the requirements to enable continuous monitoring | I | I | I | I | R, A, C | I |
| Identify and setup channels to collect end user feedback on the production software | A | R | I | I, C, R | I, C, R | I, C, R |
| Capture the metrics from the tool and share with stakeholders | I, C, | I, C | I | A, R | A, R | A, R |
| Capture customer/ end user feedback from the channels and share with stakeholders | I, C, | I, C | I | A, R | A, R | A, R |
| Analyse metrics and customer feedback and make appropriate changes / corrections / optimizations to respective plans | I, C, R | I, C,R,A | I | A, R | A, R | A, R |

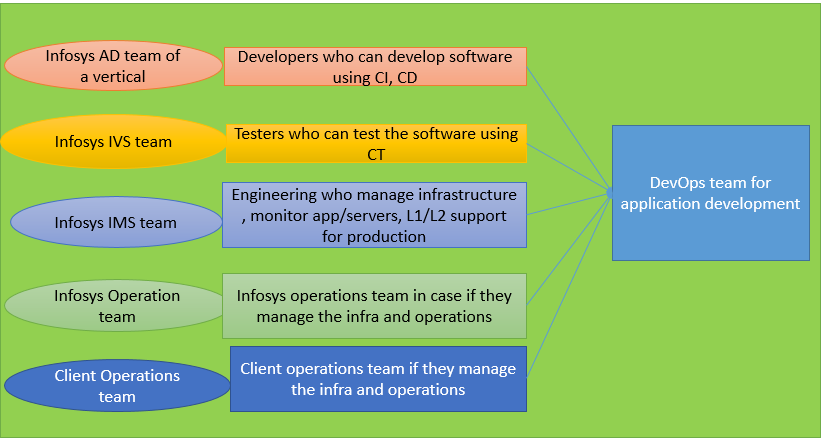
# DevOps teams

## DevOps team in Infosys context

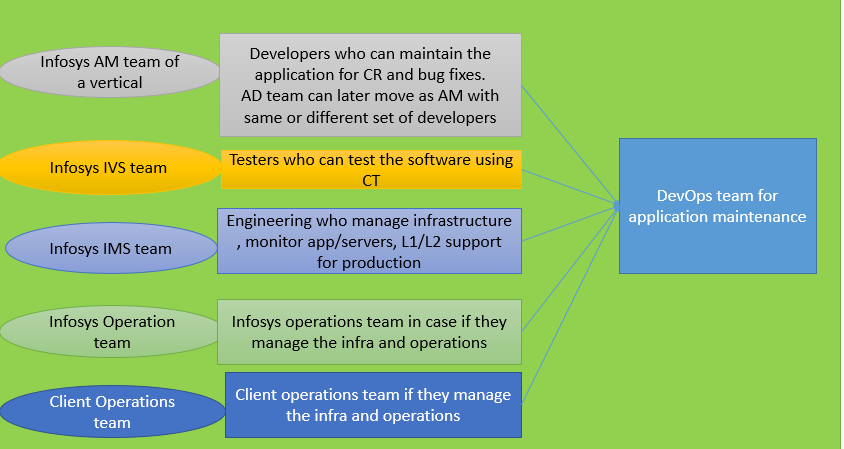
In Infosys context, we have different service lines like AD, AM, IVS, IMS teams who perform the primary job of application development, application maintenance, independent validation and infrastructure management services respectively. But when project is executed in DevOps manner the DevOps team works as a single unit/One Infy team rather than having different SOW for these individual pieces.

There could be couple of possible scenarios

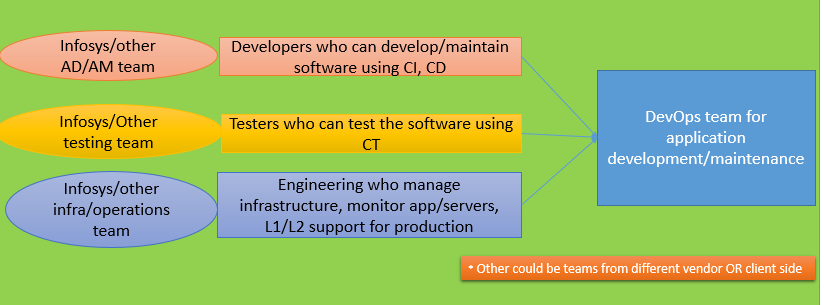
* 1. One Infy scenario – application development - If the application is getting developed from the scratch then the team formation can be as below assuming all key teams are from Infy only, then the team is formed in below manner



* 1. One Infy scenario – application maintenance - If the application is in maintenance mode then, with all key teams from Infosys, then the team is formed in below manner



* 1. Multi vendor scenario - If the application is in development/maintenance mode but all the key teams are not from Infosys (it could be other vendors/client), then the team is formed in below manner



In a multiple vendor scenario where every vendor/partner participating in the program/portfolio is responsible for a portion of the work in delivery pipeline, in order to ensure successful implementation of devops, the following aspects become critical

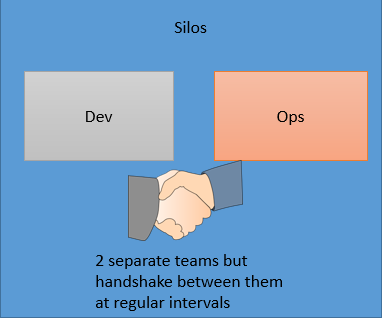
1. Assessment for readiness – the readiness assessment should also be carried out on all the vendors/partners involved along with client readiness
2. Contracts - Establishment of appropriate terms and conditions in all vendor and back-to-back contracts. Emphasis should be on agility, collaboration, continuous feedback and improvement, change management etc in contracts
3. Program / Portfolio management – Strong program / portfolio management practices have to be put in place to establish clear goals, roles and responsibilities, communication and governance models etc

## Options for team formation

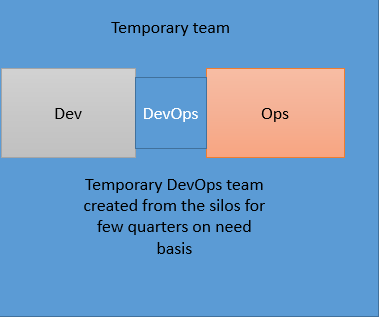
The dev and ops teams can be formed in few different ways and each one comes with its own pros and cons. Each organization can choose the best type from this list which is based on certain factors like

* + - Product range of the organization – vast or couple of products
    - Whether dev and ops can practically have shared goals
    - Team’s culture and willingness to adopt

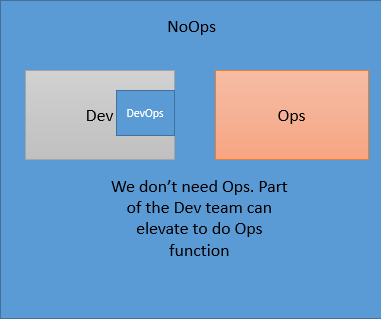
Below are few types of team formation models



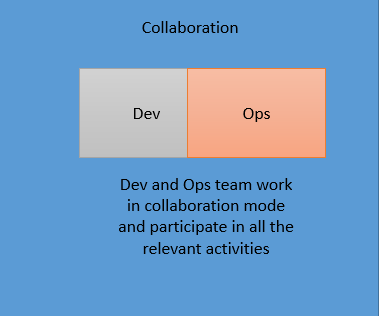
* Both dev and ops continue to operate in silos
* They meet at pre-decided intervals like stand up, release planning, retro etc. in which both understand the key essential from each/others perspective
* Helps in early identification and resolution of issues
* Better preparedness for release



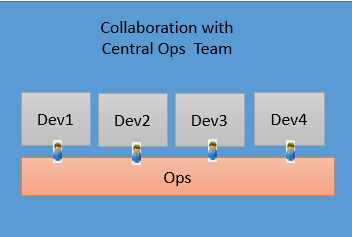
* This model is suitable more for a temporary/pilot sort of arrangement
* People from both silos come together and form a team for that particular time frame
* They agree on the governance, operational requirements and work as per that
* The members of the temporary team will ‘translate’ between Dev-speak and Ops-speak
* The temp team has a mission to bring dev and ops together and share the learnings
* The team gets dismantled after a certain period and they go back to their original teams
* Typically long term crucial responsibilities like production diagnostic/deployments etc. are not given to this temp team



* This is typically misinterpreted as there is no need of Ops and hence Ops people are typically not game for this interpretation
* But it actually means that dev don’t need to speak to Ops
* NoOps is suitable more for cloud based services and products (Infra as a service OR platform as a service) as developers can help themselves in most of the activities of infra provisioning on cloud
* Ops people can run the public, private OR hybrid infra and allow developer to manage and deploy their releases more responsibly
* Not highly recommended



* Mostly used model and suitable where there are fewer product lines. In that case we can have Ops people dedicated in one DevOps team
* It means one team culture with shared responsibilities
* Ops people in the team should have multiple skills OR there is a strong need of cross training
* Helps in smooth communication
* Possible when same product owner for Dev and Ops are same
* Effective as Dev and Ops members are together as a part of single team with shared goals throughout the project phase



* Ops team is central pool and based on required skills of projects, ops team members are assigned to one/more dev teams
* The Ops member associated with the dev team represents the complete gamut of ops activities. If that individual ops member is not skilled in a particular area, then in turn s/he will get more details from other team members of the pool
* The central team has their own backlog and work on that
* Clear-cut distinction of responsibilities performed by central ops team and ops members in dev team – for ex – central team will have 24\*7 support, DR, network management etc. under their backlog whereas ops members form DevOps team will do specific infra management, incident management, server monitoring etc.
* Careful balance needs to be maintained while composing the team as ops team from DevOps will still have some unplanned high priority work like sev1 ticket for which they need to immediately switch on to the ticket closure by leaving the current task in hand. So based on history of such unplanned work, sometime needs to be estimated and kept aside for ops team members. Otherwise it will be chaos.
* For more details on this, refer to appendix.

## User stories/work items from Dev and Ops

None of the team and process models will succeed unless dev and ops have complementing user stories/work items to each other which will aim to common shared goals

Following are some of the key user stories/work items that developers and ops team should have in their backlog. In absence of them there will always be a tussle for shared goal, prioritization, expectation mismatch and RACI issues.

Typically developers should have following items in their work item list

* How work environment can be up and running in 1-2 hours
* Why they need to consider inputs form Ops about environment and configuration of production system
* What are operational requirements / constraints of the production system and how do I take care of that in my design/code

Whereas operations team should have following items in their work items list

* What developer is working on and what the exact requirements and release plan of the same?
* What are deployment goals and how do I plan for the same?
* What inputs I need to provide to developer – in terms of usage pattern of application and report about client tickets

For more details on this refer to appendix

## End to end RACI of dev and ops

Please refer to section 4, section 5 and section 6.3 for the same.

## Transition challenges to DevOps

* **Constrained Resources in Operations - DevOps capacity issue**

Appropriate and smart way of sharing of resources from Ops Team to various Dev teams can help the problem. Also some of the Dev people can be enabled to do few of Ops activities and thus enhancing their knowledge in Ops areas as well and share the Ops load

* **There is a vast list of skill set requirement within Ops and a single resource cannot have all the skills. So how do I form the team**

This depends on the context of the application and what skill sets will be needed in that app's operation requirements. Accordingly only those skill set people can be made available in the DevOps team. In case of some niche area and limited resources, a resource from pool of resources can be assigned as a part time resource to the project.

Additionally the implementation of Automation, CI, CD and Continuous Release tooling and practices reduce the reliance on single resources – removes the bottleneck they create

* **Dev cannot become Ops and Ops cannot become dev overnight**

There’s a misconception that dev can do ops and ops can transition to dev in very short span of time. But that is not true and that should not be approach too. Let dev do the dev's job and ops do the ops job. They can pick up some skills of each other but core capability should be left with respective ops and dev people, at least in the short term. The end game is to build a cross functional team that can share skills and build domain knowledge, but this takes time.

* **Dev and ops come from 2 different worlds. How do they adjust to a common platform of DevOps?**

Dev and Ops typically come from 2 different parts of the organization with varied goals, different policies, different approval requirements, different way of execution, different tools, and different ways of information sharing and information storage. Further the teams may come from geographically distributed areas which adds complexity. By having clearly laid down operational manuals, a proper change management program run to enable the DevOps teams and people abiding to that only can help to attack this problem to a large extent. Further to this, successful DevOps operating paradigms need to create and implement shared outcomes, shared goals, and shared business objectives underpinned by shared and common KPI. This is the true enabler to bring teams together – joint outcomes.

* **DevOps is a culture. Who owns the culture?**

DevOps promotes the culture of collaboration, shared goal and result driven efforts. Typically teams think should they ‘own’ the culture or someone else in the team should. But unless and until each of the team members (who are coming from 2 different silos of dev and ops) starts owning this culture jointly, their actions will not be towards achieving true DevOps. According to some DevOps SMEs, “Until operations can identify how to embrace the new mentality of culture-fostering activities into its daily regimen, there is no way to help this aspect of DevOps grow.”

* **I am already using tools and aps. So why should I change it if it does it fit in overall scheme of things at DevOps? It is not my problem.**

These days each function has some tool and apps available for their day to day work. So many a times there is a reluctance to adopt other tools if the existing tool is not really secured enough OR is integrating with rest of the tool chain at DevOps. People should be open and flexible to adopt this change for the purpose of common and shared goals. They should welcome the change and not resist the change. One key tenet of adopting DevOps is Continual Experimentation and Learning – this includes people, process and tools.

* **Does legacy applications work with the new era of technology?**

Everyone in dev and business team wants to work on cutting edge and latest technology of cloud based system. But operations people cannot do so even if they wish to. This is because they have to support few legacy applications. The new tools may not support legacy applications. So yes, it’s a challenge that legacy and cloud based new tools may not exist together. But we need to overcome this by making it possible to have legacy systems work well with modern tools.

**Note**: More importantly, by extending what people have traditionally viewed as the Value Stream of Software delivery to include ops and support, you are by default bringing them along the journey with you. More people in legacy applications get exposure and consulted on their applications; they get exposed to new tools and techniques; and they become part of a new inclusive culture which re-motivates.

* **Complex networks make ops job difficult in DevOps scenario**

Today’s networks are extremely complex and it becomes a challenge for Ops team to really tweak, break, replicate, replace its components like interdependent IPs, ports, routers, routing tables, perimeter networks etc. And doing this very often and frequently is the Mantra of DevOps.

* **Stay stuck on talking about the transformation and not start.**

Security, reliability and compliance aspects are crucial to business so it is historically challenging to break silos that may have been construed artificially for these reasons and move into a DevOps model. When you will start the journey, there will be challenges in implementing and you should proceed removing each one obstacle as and when it is discovered and move on to the next. We would suggest employing something similar to Constraint Theory which has well defined process for optimizing systems – and DevOps model is a system – where you identify and attack one bottleneck in your system at a time to remove waste and improve flow of value. Unless you do that you will never achieve the goal. Typically teams don’t start on the journey thinking about these challenges, but just keep on talking about the transformation. Also you may be not be perfect first go, but keep on trying various DevOps operating models until you find a right model for your organization.

* **The 5 dysfunction of the team are very well applicable in case of DevOps too**



# Change management program

Moving from current as is stage to the matured DevOps stage is a business transformation programs. These programs may fail due to people and organizational shortfalls. Organization Change management is a conscious effort to address this by managing the creation of a sustainable process / mechanism which would shift the current reality into a desired state which is long lasting and deep rooted and is reflective of and / or an adaptation to internal and external environments.

An integrated approach to change management increases the probability of success for the program and improves productivity.

Change management is crucial when the organization is moving from its current state (as is) to to-be stage (DevOps). While managing this transition lot of current behavior and methodologies need to be dropped and lot of new techniques and methodologies need to be adopted by the team.

Key steps to be followed for Organizational Change Management program are as below

|  |  |
| --- | --- |
| Activity | Remark |
| Identify OCM strategies | Gather information about –   * How will you go about with OCM program? * What strategies will you use? * How are we going to educate the teams? * How are we going to ensure knowledge is maintained, retained and increased? * How are we going to ensure that developers have the right mindset & behavior (eg: think of pipelines, all commit to mainline, JUnit testing usage, build mainline after commit, publish last executables, publish overall status, use quality gates etc)? * How do we ensure that teams have the right (soft) skills? * Mode of training and communication (workshops, roadshows, videos, banners etc) * Knowledge & best practice sharing across (pilot) teams, how do we do this? * How to communicate/maintain and update processes/standards?   OCM is also a key aspect of program management methodology and we can refer to some of the available artifacts of OCM from our program management framework (**TRANSCEED**) as a base and tweak it to suit DevOps specific OCM program.  Template for OCM strategies is available at  http://172.25.103.244/sites/pride/\_layouts/PRIDE/prideDocumentDownload.aspx?DocId=ITS-GDLN-62&&isPreview=no |
| Define a Change Management roadmap that aligns the change management activities to the overall program timeline across Change, Stakeholder Management, Communications and Training | Create a detailed plan and roadmap for OCM.  Sample program OCm roadmap is given in the link below.  We can to take this as base and adopt DevOps specific milestones and activities.  <http://172.25.103.244/sites/pride/_layouts/PRIDE/prideDocumentDownload.aspx?DocId=ITS-SMPLE-165&&isPreview=no> |
| Create structure for managing change | Who will manage the change, who will act like change agents, RACI for the change management |
| Communicate OCM strategies and a detailed plan to the sponsors and key stakeholders |  |
| Conduct briefings to orient the change agents and identified teams on the program and the transformation need | Change agents are set of people who will help in driving the change. This typical is a cross functional team representing the functions that are getting impacted by the change. |
| Create change management related trainings/communication | The team needs to work on creation, communication and training of various aspects of DevOps. They can create full-fledged training material, courses, eLearning, and small practical video for a particular concept. Also reusability should be leveraged heavily as something created for one project/account can be reusable in other project/accounts too. Some of the key things that needs to be taken care in the change management program for DevOps are as below.   * Redefining KPI, goals and measures for the team at org level * New governance structure * New communication methods * New ways of status reporting * How collaboration mechanism will be leveraged * New RACI * New operational procedures * Detailed understanding of - How Ops team will get involved right from the beginning of the projects at requirement phase itself, what kind of inputs they should provide and what is takeaway for them from other team members, what will be typical user stories/work items. * Similarly what/how/when dev teams need to consider inputs from ops team and convert it into actionable at their end * Trainings on tool stack and it will enable end to end automation of deployment pipeline * What kind of monitoring (app and server) will happen at prod and other environments and how customer feedback is continuously sought and analysed |
| Deploy the OCM strategies and roadmap activities through change agents | Conduct trainings/assessment/system changes/policy changes etc. to ensure that the detailed plan and roadmap is implemented |
| Monitor effectiveness of OCM strategies | Have mechanisms for - how OCM effectiveness will be monitored |
| Define a sustenance plan | Define a sustenance plan – how onboarding for new team members will happen? How ongoing training and enablement plan will look like? How it will be tracked on ongoing mode |
| Capture key learnings | Capture key learnings during OCM implementation for further refinements OR for inputs to other OCM programs. |

# DevOps metrics

DevOps benefits can be derived only when you identify certain key metrics, set baselines standards, measure the values and monitor their improvement throughout the journey.

These metrics will be measured various areas like *Customer, Financial, CI, CT, Availability, CD/Release Efficiency, Reliability, Efficiency*

*Infra efficiency, People, Quality*

The key metrics that can be tracked during DevOps journey is mentioned in the table below.

| **Dimensions** | **KPI** | **Metrics to measure** | **Definition** | **Measurement Frequency** | **Remarks** |
| --- | --- | --- | --- | --- | --- |
| Availability | Application Availability | Percentage of availability of application in production | Total Uptime of the application / Uptime + downtime of the Application | Monthly | If you can maintain healthy uptime even given fluctuations in user volume, you’re doing pretty well. This metric averages the number of hours/minutes of maintenance down time that has occurred over a span of time for the measurement |
| Availability | Environment Availability | Percentage of availability of servers grouped at the Environment level (Dev, Stage, Prod, etc…) | Total Uptime of the servers / Uptime + downtime of the Servers | Monthly | Determine stability of environments across the enterprise OR for a selected environment like production |
| CD/Release Efficiency | Lead time for provisioning | How long does it take to provision an environment? | Time taken for environment provisioning request to provisioning | Weekly/Monthly | Avg time taken for this should go down over a period of time |
| CD/Release Efficiency | Production Deployment Frequency | How often is the team deploying new code to production? | Number of deployments on production per unit of time | Daily, weekly, monthly (depending upon org context | This metric should be constant OR should keep upward trend |
| CD/Release Efficiency | Release date adherence percentage | Whether the planned and actual release dates were same? | % of releases where release date was adhered to planned date/ Total number of releases | Monthly/Quarterly | To track whether we are releasing changes as per the defined release date OR delaying it Apart from date if there is any other aspect on which SLAs are defined, we can track releases on that aspect |
| CD/Release Efficiency | Deployments/Day | How often is the team deploying new code to various environments (not just production)? | Total number of deployment across various environments / # of days | Daily, weekly, monthly (depending upon org context | To understand the increased collaboration for auto deployments |
| CD/Release Efficiency | Change Volume | How many user stories/changes are being deployed? | Total number of changes deployed to production per unit of time | Weekly/Monthly | We can also add complexity factor to the change and then calculate the overall number of changes deployed. For ex  Simple change = x, medium change = 2x, complex change = 3x |
| CD/Release Efficiency | Lead Time (from development to deployment) | Time lag from when new code starts getting developed to when it successfully gets deployed into production | Time when change is deployed to production - Time when the development for change started | Monthly/Quarterly | Lead time should reduce as the team gets a better hold of the lifecycle in DevOps |
| CD/Release Efficiency | Percentage of Failed Deployments | What is the percentage of deployments which have caused an outage or negative user reaction OR could not be successfully deployed to production | (Failed deployment/Total deployment)\*100 | Monthly/Quarterly | This metric should be reviewed together with change volume. If the change volume is low or remained the same but the percent of failed deployments increased, then there may be a dysfunction somewhere. |
| CD/Release Efficiency | Percentage of successful Deployments | What is the percentage of deployments which were successfully deployed? | 1- (% Percentage of Failed Deployments ) | Monthly/Quarterly |  |
| CI | Failed builds | Number of build failures | Number of successful build/ Total number of builds | Monthly/Quarterly | -Qualitative analysis of CI process output -Help plan improvements to increase build success % |
| CI | Build Time | Total time taken for the build | Avg (Time when build was completed - time when build was triggered) | Weekly/Monthly | -Track build processing time which includes compilation, inspecting, testing and deployment -Helps in making builds more efficient for faster execution & early feedback |
| CI | Unit test coverage | Unit test coverage | % of code covered in unit test | Per build | -Track efficiency of test driven development -Helps ensure code quality |
| CI | Unit test success rate | Unit test success rate | % of test cases passed during unit testing | Per build | -Track efficiency of test driven development -Helps ensure code quality |
| CI | Rule compliance/violations | Rule compliance/violations | # of compliance/violation issues during reviews/build | Per build | -Track non-compliance against set coding conventions -Helps developers write better quality code |
| CI | Code complexity | Code complexity | How complex is the written code - depends upon tool usage | Per build | -Track non-compliance against set coding conventions -Helps developers write better quality code |
| CT | Working Software | Running Tested Features (RTF) | (# of Features / stories which are implemented & Pass Auto Test)  / ( # of Features /Stories in the Sprint Backlog) | At every sprint level | How many stories are getting through the quality gate of auto test |
| CT | Test Automation Effectiveness | % of Unit , Functional, Non Functional & End to End Automation at the System level | (# Scripts- Unit / Func / NFR/E2E Test)  / (# Test cases- Unit/Func/ NFR/E2E Test) | At every Sprint / Iteration level | % of automation in the entire flow |
| Reliability | Number of outages of software in a unit time | How many times in a unit time does the system face outage issue? | Number of outages of software in unit of time | Weekly/Monthly/Quarterly |  |
| Reliability | Mean Time To Recovery (MTTR) | When failure does occur, how long does it take the team to recover from the issue | Average Time taken from Outage alert to Availability in Production Environment (MTTR ) | Monthly | Spikes in MTTR are fine for complex issues which the team has never encountered before, but the overall trend for this metric should decrease over time. |
| Customer | Thought Leadership (Ideas) | No of Initiatives on thought leadership & Innovation | #  of ideas generated & Implemented # of PoC conducted # of Patents registered | Half Yearly |  |
| Customer | Customer Satisfaction | CSAT Survey score | CSAT Survey Score | Every release OR Quarterly (depends on how the organization collects this) | As per standard org definitions |
| Customer | Devops Adoption Percentage | Percentage of projects with DevOps Adoption | # of projects with DevOps adoption  / Total Projects in the Enterprise | Quarterly |  |
| Customer | Business Continuity & Resiliency | Recovery Time Objective (RTO) | This indicates the maximum time within which the normal business operations should be restored. | As per Business Continuity plan | Recovery time objective (RTO) is the maximum desired length of time allowed between an unexpected failure or disaster and the resumption of normal operations and service levels. The RTO defines the point in time after a failure or disaster at which the consequences of the interruption become unacceptable - Target time uptill which the business can survive post a downtime /disruption or a disaster |
| Customer | Business Continuity & Resiliency | Recovery Point Objective (RPO) | Maximum acceptable data loss represented as Time. | As per Business Continuity plan | RPO is measured in time and dictates disaster recovery procedures. It sets the time or age of data/files that must be recovered from the backup storage for business to resume its normal operations. This time usually determines the frequency of data backups and indicates the data loss tolerance. **For eg: if the RPO is set to 30 minutes, then a backup of the system is required to be done every 30 mintues** |
| Efficiency | Ticket Resolution Productivity | Average Ticket Resolution per Person | Total Tickets Resolved  / Total Persons in the Project | Monthly |  |
| Financial | % Change in User Volume | Number of new users signing up, interacting with service and generating traffic. | # of new users accessing/using the deployed change | Monthly | This metric helps you in tracking 2 things - Is your infra capable enough to handle the increased load - Is the increased traffic in line with the expected outcome of the change. For ex - Adding X category of item in the shopping cart will add 500 new user accesses in a month was the expected benefit. Against that we can check how many new users actually accessed the changed functionality |
| Financial | Cost/Effort per release | Avg cost OR effort needed per release | Avg cost OR effort needed per release | Quarterly | To keep a track of cost of release |
| Financial | Infra Cost Savings | Infra cost Savings due to Virtualization | Infra cost Savings due to Virtualization in USD | Quarterly |  |
| Infra efficiency | Capacity Utilization | % capacity utilization and availability | Utilized capacity of server / Total available capacity |  | Helps to plan for better capacity planning and usage |
| People | Skill Enablement | # Training Hours per Member | (Total no of training hours ) / (Total members in the project) | Quarterly |  |
| People | Knowledge Management | KM Effectiveness at Team / Program Level / Portfolio Level | (No of Hits / references on the KM artifacts) / (No of total artifacts in the KM repository) | Quarterly |  |
| Quality | Customer Ticket Volume | Does a deployment cause any change in the customer ticket value in positive way? | Average number of tickets generated in connection with issues due to change OR Number of tickets pre change and post change | Monthly | This can be interpreted in couple of ways as per the definition. But this metric essentially helps in measuring how many calls, tickets, emails, issues, etc. are received by your customer support team post the deployed change. |
| Quality | Performance (Response Time) | Does the product or service operate within predetermined thresholds? | As per defined response time for various components in the application | Monthly | Ideally this metric should remain stable and in given threshold irrespective of % change in user volume or any new deployment |

# DevOps maturity

As we have seen in prior section, DevOps is a journey of transformation and it evolves over a period of time. DevOps gets matured over a period of time. So there should be a process to measure the current maturity of the DevOps, set the baseline standard, and then carve out a plan to move to the next stage of maturity level.

The maturity can be decided across various crucial points applicable to DevOps like - Agile practices, CI, CT, CD, team, environments, architecture, and operations. These can change based on client context. At Infosys, the maturity levels are named as – Crawl, Walk, Run and Sprint. This can be used during initial *Propose* stage OR in *Optimize* stage as a reference.

The below table captures the questions based on which the maturity level is determined for each of these aspects.

| **Tenet** | **Question** | **Crawl** | **Walk** | **Run** | **Sprint** |
| --- | --- | --- | --- | --- | --- |
| Agile | Vision | Unclear vision | Organization has a well-defined vision and strategy. Project Goals are derived from this vision | Project Takes into account the vision of the company as well as a customer (Case of ODC) to derive the Project vision and its goals. Appropriate measurements are put in place to measure the same but they are manually measured | Project Vision is clearly articulated in terms of currency/cost. Business Tests (wherever applicable) are generated to check the same at various pre-determined check points and they are automated in nature. In case of metrics, they are automatically calculated from the measures generated. |
| Work Allocation | Task is assigned to team members directly by product owner/ PM | Done by the Management/Scrum master to Developers and Testers | Team members choose on their own and implement them | Brainstorming sessions are held to moderate the stories chosen by members. PO along with scrum master ensures that every member is satisfied either in this sprint or the next one. Procedure defined to ensure that all the good stories will never be picked from same person. Everybody gets an equal chance. Scrum Master to ensure that members would select some stories related to Subsystems/modules which they have never worked on. This way team is in Continuous improvement mode |
| Information Radiators | No visibility and traceability | Specific Period is planned to take up the information radiated through various sources | Team members give highest priority to the radiated information. Actions are taken immediately if the radiated information is related to broken software or hinders the concept of working software | Cross functional team related to radiated information gathers together to check the impacted areas and will come up with the areas to be fixed. Cross functional team fixes all the dependent areas and ensures working software is available. |
| Customer Collaboration | Customer interface doesn’t exist for any team members | No Customer near the team. Team sends a mail to customer in case of any clarifications and waits for his inputs. | Proxy customer is identified and he interacts with the customer regularly. He has knowledge on Product as well as market to take right decisions. | Customer representative sits beside the team and immediately clarifies the issues/doubts with the team. |
| Product Backlog Grooming | No backlog grooming activities | Product Owner is the sole responsible person and he takes care of Product Backlog | Senior members of the Team along with the product Owner spends some time for grooming the items in the product backlog but it’s done on need basis or randomly. It’s not a planned activity | 5-10% of the Sprint's effort is planned in Product backlog grooming and all the team members are part of it. This activity happens regularly before the current sprint closure |
| Business value | practice doesn’t exist | User Stories are prioritized by PO and developed by team based on indirect measures like Complexity , Customer importance etc. | User stories are associated with a Business value which is derived indirectly based on predetermined set of the parameters used in Story prioritization | User stories are associated by a value (in terms of currency) to indicate the highest priority and value delivered to Customer. Performance indices are put in place to co-relate all the metrics like Schedule, Defects etc. and measure impact on cost. |
| Customer Demo | Not practices | Office tools /show and tell presentations are mailed to the customer to get the feedback through Mails. | Working software is installed at clients location using remote connections and teams wait for offline customer feedback | Telepresence and webcams with live Videos are used to Demo and Feedback |
| Project Management | Use waterfall project management tools and practices | Manual tracking involving mails and WBS. | Regular Status meetings with a concept of Burn down charts in place. Snapshots of Story Wall/ Burn Down Charts/ Risk Charts etc. are mailed at the end of the day to all stake holders by the scrum master in case of distributed teams/Stakeholders. | Online Story Board like JIRA/Green Hopper/ALM tools are used real time status. These can be accessed online at any point of time to get the latest status. |
| Deliverable Updates | No communication | Updates to Deliverables are communicated manually through mails and Physical Boards to all the team members. | Update communication is automated. Source safes and other databases used to store the deliverables are enabled in such a way that automated mails are sent after every update is checked-in into the source safe. | Deliverables / Artifacts are maintained in an online system giving the live info. Buzzers, Mailers and live indicators are used to communicate the information updates to all the stakeholders. |
| Story Cards | No Story cards | Physical Story Cards are used capturing the info needed to implement the same. | Story cards are present on the story walls which is kept near the team. Members update the card as and when they find any missed items or issues in the story. | Tools like JIRA, Green hopper or SharePoint are used to capture the Story cards and Electronic Media is used as information radiators. Status is updated in the tools and dashboards are accessible by all stakeholders instantly. |
| Organization Structure | Structure is Rigid. Decisions are made at the top and communicated to lower management levels for execution.(Top down approach) | Aware of a change required and conscious effort is made | Key Members from the lower Management level are involved in decision making. Inputs from lower management/teams are given appropriate weightage before finalizing the org level objectives | Top management is easily approachable with viewpoints from the lower teams. To and Fro communication is entertained. Regular discussions/meetings/get-together are held to remove Virtual Barricades /Mindsets/fear to approach the senior leaders. Regular open houses and one on one's are conducted to get Innovative ideas from the teams operating at the ground level( like Coffee with Boss etc.) |
| Monitor , Evaluate and Assess | After the project is completed | Manual Reports are to be prepared at various levels. Separate Meetings at different levels are held to discuss the issues explicitly. Higher importance is on meetings and reports. Lot of effort is spent on the same. | Report generation is automated with metrics getting projected at different levels within the organization. But staged management expects the reporting to happen in presentations. | Virtual Dashboards are used for reporting. These dashboards are customizable at different levels and can be controlled based on access levels. Relevant information is projected based on the user login and various fields are enabled to add appropriate comments. Explicit sections are reserved to highlight the blocks which needs immediate attention. Meetings are called only on need basis. |
| CI | Are you doing any pre check/commit activity? | No. We are not doing any pre checks. | Check only for comments before check in. | Comment on Code Check in policy and bug id reference policy. | We are using the following policies before a developer checks in  1. Policy for Bug ID reference 2. Policy for special characters 3. Policy for wrong Bug ID 4. Policy for Invalid User 5.Alert/Notification/Comment Policy on Check-ins/Commits 6. Automatic code review policy 7. Code Coverage Policy  8. etc. |
| What artifacts are maintained in central version control system? | Version Control is not used or check in happens infrequently. | Source Code only | Source code and database scripts. | We are using central versioning control system for Source code, data model, build and deployment scripts, database scripts, environment-specific configuration and test data. |
| Do you store executables/ Production builds in SourceCode Management System? Do you check in libraries in the source control? | Yes | No but Store in different locations according to need | No but Stores in central repository like FTP or Share Point | No. Whenever it is required we build the code and get it. |
| What is your branching strategy? | Several long-lived branches for new features or projects. Merges are painful and infrequent. | A few short-lived branches for features, merges are frequent and perhaps automated. |  | Trunk development, with branches only used for releases. Feature hiding, incremental development, branch-by-abstraction and componentization used to keep the application releasable at all times. |
| What is your merging strategy? | Manual and it takes lot of time and efforts. We merge once a month or at the time of releases only. | Manual but we merge very often either daily of twice a week. | It is automated but have to do some manual efforts too. | It is fully automated and we do not need to bother about merging. |
| How frequently do developers check in changes? | Only upon feature completion. | Once per week. | About once a day. | Several times a day. |
| Do you build everything from source every time and does it run quickly irrespective of the size of the project? | No | Not every time | It is not quick and depend on size of project | It is quick and size of project is not an issue. |
| Can the team specify the versions of the dependencies their application depends on, and your build system automatically fetches the ones that are known to result in a good build? | No. Dependencies are maintained manually. | Yes few are taken care by Build System. |  | Yes Build system is designed to take care of all dependencies for a given version. |
| Do you have build automation in place? Or Which of the following most closely resembles your build process? | When we want to test the application, somebody builds it manually. | We have an automated build process that is run once a day, or less frequently. | The application is built automatically every time somebody commits a change. | Changes are only committed after developers have run a successful, fully automated pre-commit build. |
| How are artifacts managed in your build process? | Binaries are compiled every time they are required as part of our deployment process. | Binaries are created once, and promoted through the build process. | Binaries are created once, and promoted through the build process but we do not have visibility of what deployed on which environment. | Binaries are created once, and promoted through the build process. Also we can easily see which artifacts have been deployed to which environments, and which stages of my build, deploy, test and release process they have been through. |
| How do you manage dependencies in your build process? | Developers download libraries on an ad-hoc basis. | We build everything from source every time, and the time is depends of the project size. | All libraries are checked in to source control. | Teams specify the versions of the dependencies their application depends on, and my build system automatically fetches the ones that are known to result in a good build (not necessarily the most recent snapshot available). |
| Do you commit changes only after developers run a successful fully automated pre-commit build? | No |  |  | It is a part of CI and developer only can check in code after a successful build and check in policies. |
| Do you do continuous integration? | No, we have no automated tests, or we don't run them regularly. | No, but we do have an automated process that builds the application and runs unit tests regularly. Periodically an attempt is made to make the tests pass. | Yes, we have an automated process that builds the application and run unit tests every time somebody checks in. Whenever the tests break, they are immediately fixed. | Yes, we have a pipelined build and deployment process that builds, runs unit tests, and then subjects the generated artifacts to automated functional and integration testing. |
| Do you have an automated build process that is run once a day or Nightly or weekly? | Not at all | No. We do manual build execution weekly/nightly | Only Nightly | We have weekly, nightly and less frequent automated builds. |
| What is your code coverage strategy/ Technique? | None. | Yes but it is in initial stage | We have automated code coverage tool but it is not giving satisfactory results | We have in built code coverage tool. Our code coverage process is fully automated. |
| Do you have automated build status reporting mechanism? | Not at all | We use manual process to create the report. | Only build pass/fail reports rate report is available. | We have email alert as well desktop notification tool to know the status of build (Build is triggered, by whom, what is the status etc.).All kinds of reports like build pass/fail, build pass/fail rate, Build Duration, test report. |
| Testing | How do you perform Unit testing? | We do not do unit testing | We do but it is not well planned and we are facing lots of difficulties putting in place. | We do and our developers run them. Code coverage is also not more than 50-60 percent. | Fully automated and integrated and is a part of CI. |
| How do you perform Smoke testing? | No | We do and our testers run them manually. It takes a complete day or 4-6 hours to finish them | We do have automated but our teams run them manually from their own environments. | Fully automated and integrated and is a part of CI. |
| How do you perform Non-functional testing? | No | We do and our testers run them manually | We do have automated but our teams run them manually from their own environments. | Fully automated and integrated and is a part of CI. |
| How do you perform regression testing? | No | We do and our testers run them manually. | We do have automated but our teams run them manually from their own environments. | Fully automated and integrated and is a part of CI. |
| When is acceptance testing of your application performed? | When development is complete on a release, the testing team performs manual acceptance and regression testing. | When development is complete on a feature, the testing team performs manual acceptance testing. Regression testing is done at the end of the release. | We have a suite of automated acceptance tests that are run on a regular basis, but they are often broken and are supplemented by manual testing to make sure we catch all the bugs. | We have a suite of automated acceptance tests built by developers and testers in collaboration that cover at least the happy paths of our features that are run on every check-in. Fixing them is the highest priority. |
| How do you perform UAT? | No | We do and our testers run them manually. | We do have automated but our teams run them manually from their own environments. | Fully automated and integrated and is a part of CI. |
| How do you perform database testing? | No | We do and our testers run them manually. | We do have automated but our teams run them manually from their own environments. | Fully automated and integrated and is a part of CI. |
| Who does testing? | There are no dedicated testers. Members of the project team perform testing on an ad-hoc basis. | There is a common testing team responsible for creating and maintaining tests. | We have dedicated manual test engineers for different types of testing. | We have a suite of automated acceptance tests created by testers and developers that cover at least the happy paths of our features that are run and pass frequently. |
| How are databases created and maintained? | Databases are created manually, and all changes to them are performed manually. | Schema changes are handled by a separate team with a different release schedule. So all changes have to be backwards compatible. | There is an automated process for creating the schema of a database, but data migration is performed manually. | All database creation and any migrations as part of an upgrade are automated, and any team member can self-service their own database operations. |
| Interlock delivery dates - Interface Test prior to delivery into E2E Test | Interlocked delivery dates and integration plan intra-platform Intra-platform CIT environment Automated interface testing 0 % Manual interface testing 100 % Automated test data set-up 0% Frequency Prior to release to E2E | Interlocked delivery dates and integration plan intra-platform Intra-platform CIT environment Automated interface testing 25 % Manual interface testing 75 % Automated test data set-up 25% Frequency Fortnightly | Interlocked delivery dates and integration plan intra-platform Intra-platform CIT environment Automated interface testing 50 % Manual interface testing 50 % Automated test data set-up 50% Frequency Weekly | Multiple interlocked delivery dates (i.e. cross LOB) and integration plan cross-platform Cross-platform CIT environment Automated interface testing 75 % Manual interface testing 25 % Automated test data set-up 75% Frequency twice-weekly |
| How is test data managed? | We take a dump of the production database. | We have a dump of a database that is created by the test team. | We automatically set up a minimal representative set of data for each test run that is appropriate to the test suite. | The application is put into the correct state for tests via its API. |
| CD | How long would it take for your project to release to production a change that involved a single line of code? | Almost a week without | Few minutes by replacing effected binary directly in production server | Few Days | Hours or minutes, going through our fully automated build, deploy, test and release process. |
| How long does it typically take to go from deciding to create a new feature to releasing it to users? | Months | Weeks | Days | Hours or minutes, going through our fully automated build, deploy, test and release process. |
| How do you ensure traceability from source control to production? | We follow back the email thread, or do manual archaeology. | We put the build number in artifact filenames, but there is no automatic validation that the binary hasn't changed. | Our automated build and deploy system tells us exactly which build is in every environment at any time. | Our automated build and deploy system tells us exactly which build is in every environment at any time, and can tell us the exact revisions in version control that were used to create it. |
| How do you manage release documents like change log, release notes, User manual, etc. | Manual without any versioning system | Manual but we store them in version control system. | Automated for internal use but we have manual process for customers | It is fully automated and part of CI. |
| Is the application of any version is always ready for demo? | No | No and it takes at least weeks of planning | No but it takes few days of planning | It is always ready for demo |
| Environments | Which of these statements most closely resembles how testing and (if relevant) production environments are provisioned and maintained in your organization? | They are created and maintained manually on an ad-hoc basis. | There is a standard manual install process that is followed to create environments. Patches and other software are installed on an ad-hoc basis. | Operations staff manages all testing and production environments. There is a standard install process, and changes are applied manually after they have been approved. | There is a fully automated process for creating environments. All changes are pushed out by an automated system. Only operations staff can perform changes to these environments, following approval. |
| If you lost one of your environments due to a catastrophic hardware failure, how would you re-create it? | No idea what was on the box. We're hosted. | We have a pretty good idea of what was on the boxes, so we should be able to recreate it in a day or two. | We have a standard installation procedure, some of which is carried out manually. Installed software and patch levels are recorded as part of our configuration management process. | A set of fully automated processes are used to install and configure the operating system and middleware, based on configuration information stored in version control. |
| How are new versions of your applications deployed to testing environments? | They are deployed manually. | Most parts of the deployment process are automated, but some parts are manual. | There is a fully automated deployment process which can be run by operations staff. | There is a fully automated deployment process. Team members can self-service the builds they want to their environments with approval. |
| Is your infrastructure is maintained on a virtual or a physical servers? | No Virtual. All physical | Partial virtualization | Everything is virtualized but new environment creation is manual. | We have fully automated system to recreate the desired release environment on virtual machines. |
| How are your applications configured? | Application configuration is performed manually at deploy time. | Configuration information is packaged inside our binaries. We have to compile a fresh for every environment. | Configuration information is packaged alongside our binaries. We take the same binaries, but package them separately for each environment. | Configuration information is validated before being injected or fetched by the application at deploy time or run time, and we use the same configuration mechanism for all of our applications. |
| Are there major differences between testing and production environments? | Yes we have differences in testing and production environments. |  | No we have similar environments for testing as production. Also we have testing environment which | We have an automated process to create different kinds of environments which are similar to production etc. |
| Org CM and culture | Team composition | Dev & Ops are separate team, never interacts during development activities. Dev handover it after all the activities | Dev & Ops are separate teams, Ops is involved sparingly, when somebody feels like… | Ops specific to Application are involved systemically from the beginning of release plan, along with feature user stories, there are stories of Ops and detention of done is constrained with Ops requirement. Ops bandwidth allocated to sprint teams during a release | One single team - DevOps team, no differentiation between Dev & Ops. |
| Teams measure | Separate manual measure for each groups in the value stream eg: Dev separate, QA separate, Ops separate | Discussions are happening in to have measures across teams, and manual framework is getting established. Tools identified | Basic manual governance framework across the department using tools implemented | End to end visibility, traceability and automatic dashboards |
| Architecture | Architecture Modularity | Monolithic | 2 tier architecture | SOA based | highly modular |
| Designed for production | NFRs not considered at all, dependencies are not managed well | NFR testing is done after testing , knows dependency does trial and error | NFR testing along with sprint, Dependencies are managed structurally | NFR as part of user stories, Dependency are handled automatically |
| Operations | Resilience & Disaster Recovery: | Recovery Environment shared with other environments   RTO (Time taken to recover) is > 12 hrs.   RPO (State of recovered system) is > 12 hours behind the   Process to bring system in alignment with BAU  is Un-documented/ unpredictable | Recovery Environment Dedicated but not identical to Prod (e.g. different zone, size etc.)   RTO is Recovered system > 8 & <12 hours behind BAU  RPO > 8 & <12 hours behind BAU   Process to bring system in alignment with BAU is mostly document only few scenarios are unpredictable | Recovery Environment dedicated & same zone but not synchronized with Prod   RTO > 4hrs & < 8 hrs.   RPO > 4 & <8 hours behind BAU   Process to bring system in alignment with BAU is mostly document & process is 100% predictable | Recovery Environment dedicated & same zone & replicated periodically to synchronize with Prod   RTO < 4 hrs.   RPO < 4 hrs. behind BAU   Process to bring system in alignment with BAU is mostly document & partially automated |
| Performance Scalability Testing: | Availability of Infra <20%  % of performance parameters measured <10%  % of automated tasks to measure performance <10%  % of manual tasks to measure performance >90%  Frequency of performance testing > 12 months  Capacity Planning | Availability of Infra >20% & <40%  % of performance parameters measured >10% & <40%  % of automated tasks to measure performance >10% & <40%  % of manual tasks to measure performance >60% & <90%  Frequency of performance testing > 9 months & < 12 months  Capacity Planning Re-use of regression test harness to prove basic estimation model | Availability of Infra >40% & <60%  % of performance parameters measured >40% & <60%  % of automated tasks to measure performance >40% & <60%  % of manual tasks to measure performance >40% & <60%  Frequency of performance testing > 6 months & < 9 months  Capacity Planning Representative anonymized live data incorporated into basic estimation model | Availability of Infra >60% & <80%  % of performance parameters measured >60% & <80%  % of automated tasks to measure performance >60% & <80%  % of manual tasks to measure performance >10% & <40%  Frequency of performance testing > 3 months & < 6 months  Capacity Planning Estimation model extended to cover cross platform components |

# DevOps tool stack

Pls refer to teamwiki for all possible details on DevOps tool stack <http://teamwiki/DevOps/CICDDetails>

# Myths vs reality of DevOps and its capabilities

* **Devops is just about communication** – People have coined humorous words like ChatOps OR HugOps as they think that DevOps is all about having good communication. Yes having a great communication and collaboration is a must in DevOps but it is not the only thing in DevOps.
* **It means continuous change deployment** – For some organizations, it may be true as they can deploy changes on daily/hourly basis. But for many organizations it is not possible and also not needed to continuously deploy the change to production on daily/hourly basis. But still they can adopt DevOps very successfully as they can deploy the change based on the business need and not by a pre-decided calendar date
* **DevOps is not suitable for large and complex systems** – DevOps rather helps these systems as it has many capabilities which are needed for successful deployment and release of complex systems. So DevOps with great amount of collaboration, automation and coordinated release management, automated deployments comes to the rescue of large and complex systems
* **DevOps is possible only when the entire team is from same organization** – No, even if the development OR testing is outsourced to other vendors, these teams also become a part of overall DevOps teams and can adopt the processes needed to achieve the benefits of DevOps
* **DevOps works only on cloud** – Cloud surely is beneficial in DevOps but it is not a necessity for DevOps. With clouds, the dynamic infra provisioning for dev and testers becomes easy. But without cloud also with good processes/tools around the same, the resources can be obtained easily.
* **I** **write scripts means I have automated deployment** - For small deployments, writing deployment scripts will help in automated deployments in a large way. But for complex and large deployments, one should go for a full-fledged automated deployment solutions and should not just rely on few deployment scripts
* **DevOps means dev+ops so thus affects only these 2 roles** – No. DevOps touches all possible roles and stakeholders in the software delivery organization like LOB, practitioner, BA, business, testers, partners, suppliers. In fact an explicit role like DevOps in the long run should cease to exist as teams may take DevOps as business as usual (BAU).
* **I have ITIL which means I can’t have DevOps** – ITIL and DevOps can surely coexist. Pls refer to section 2.4
* **In DevOps Ops learn to code** –Operation team typically manages the infra through various scripts. Yes! Now with Infra as a code these scripts need to be managed as code and they also need to understand the concepts like version control, check in, check out, branching etc. They also need to understand various tools like Ansible, puppet, chef etc. For ex – For Ansible, they only need to understand secure shell scripts (SSH). There is no server component for it. But they essentially need not learn c# or java code to work coherently in DevOps
* **DevOps is only for small and “born on the web” companies i.e. only for startups** - Pls refer to section 2.4
* **Regulatory heavy sectors cannot use DevOps** – Such sectors need to have lot of manual checks and balances and stakeholder approvals. But if adopted properly, DevOps can enhance the auditability and compliance aspects of the system to a large extent
* **Service virtualization is meant only for testers** – SV is surely helpful to testers but it is equally useful for development team wherein they can do the integration testing right at the time of development.
* **Developers can create best deployment process** – As they know how exactly application works. But they don’t know the environment and topology where the application will be installed. So developers designing the deployment process is not a good idea at all as it may neglect some important operational clusters like load balancer, clusters etc. Hence a collaborative team of operation, development and release management should design the deployment process
* **Even complex releases can be easily managed using spread sheets so no need to do investment in release management solution -** It’s a wise decision to invest in good release management solution especially when you do large and complex releases frequently. You can still manage large releases with spreadsheets but the proper solution provides lot of benefits in terms of trigger, alert, dashboard, collaborative release planning for infra and application
* **One large release is less risky than small frequent releases –** No. small frequent releases are less risky as compared to one large release especially when you are using apt solutions for deployment and release. Smaller releases have fewer components and fewer interdependencies and hence less risky.
* **Deployment automation means reducing the control –** Rather it increases the control. Most of the automated solutions have role based security and a clear audit trail can be established about who did what. So you always know what happened at what point in time.

# References (used for this document)

1. DevOps for dummies – Sanjeev Sharma
2. Service Virtualization for dummies – Marcia Kaufman, Judith Hurwitz
3. Application release and deployment for dummies – Eric, Jeffery, Claudia
4. DevOps for developers – Michael hutterman
5. Kick starting DevOps – New Relic
6. DevOps primer – New Relic
7. ITIL Guide to DevOps – By ScriptRock
8. Infosys COE Teamwiki
9. Infosys EQS references
10. Infosys solutions – ICIP and ITSDP
11. Multiple articles , blogs and video on DevOps

# Appendix

## Value stream mapping in DevOps

DevOps adoption is not one time activity – taken up and done. It’s a journey of continuous improvement by adopting set of capabilities and practices that are based on some of the Lean Practices. It requires, organization change management to enable culture, extreme automation and improved processes.

So how do we start this long journey? How we identify current stage and stage where we want to be in? We typically may need to start from level of immaturity where we may not even have good source code management practices, and we want to reap benefits of the true DevOps implementation is going to be a long journey.

Adoption of Value Stream Mapping concept of Lean Principles help practitioners in this aspect.

**Value stream** mapping is a [lean](https://en.wikipedia.org/wiki/Lean_product_development)-management method for analyzing the current state and designing a future state for the series of events that take a product or service from its beginning through to the customer - *Wiki*

This helps you in understanding the end to end application delivery pipeline and processes of as is scenario, identify the bottlenecks and inefficiencies and then suggest a roadmap to start your DevOps journey

Thus value stream mapping is crucial in any DevOps engagement. It helps you in

* Setting context and background for the technical team who examines the tools and technology part
* Understand performance baseline against which improvements due to DevOps can be measured
* Understand how work of dev, tester, ops is linked to upstream and downstream

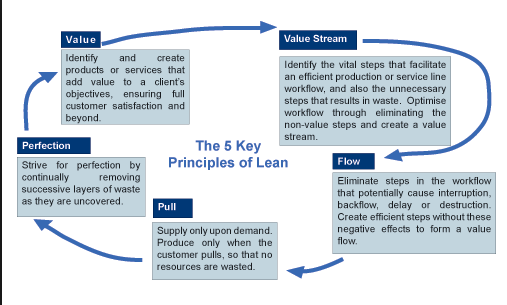
Typical questions that should be asked and validated during VSM exercise are

* What activities are being done? Who does it? What is dedicated time for the same
* What causes rework? What is % of rework – to establish the “right first time” principle
* What tools are being used? What is input and output of the step?
* How long does it take to complete step?
* How many units are work in progress within this step? How many units are waiting in the inbox? How are they prioritized?
* What these units typically wait for? What are dependencies on others?

A VSM template is available at

<http://172.25.103.244/sites/pride/_Layouts/Pride/prideDocumentDownload.aspx?intDocId=1794>

## How Lean Principles are used in DevOps



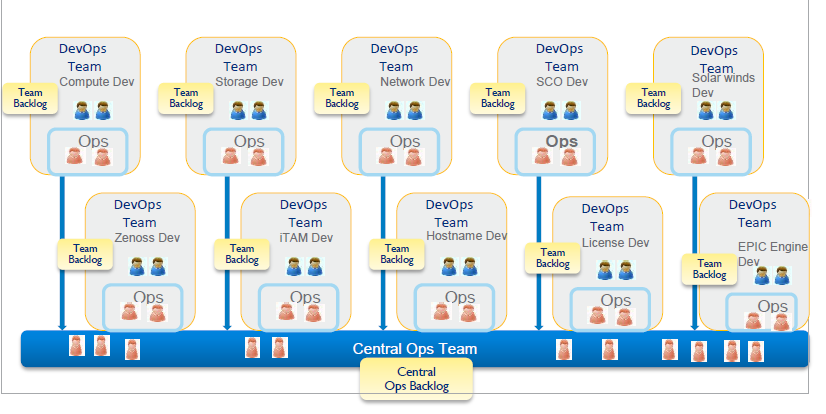
Key Lean principles are shown in the diagram above. The below section shows how it is leveraged in DevOps scenario

1. Value – the principal of DevOps is to deliver great value to customers at right point in time. By doing continuous business planning and with enhanced feedback loops in DevOps we are assuring better chances of having high values ideas delivered to clients at most optimized timeframe
2. Value stream – DevOps recommends the mapping of complete workflow of current service lines to understand the pain points, waste, bottlenecks in the flow. Based on that suggest the changes in the delivery pipeline so as to remove the same and achieve overall efficient workflow.
3. Flow – DevOps also focusses on understanding the steps in the workflow that can create delays/wait time/interruption/roll backs etc. We need to ensure that these steps are made further efficient and these negative effects are removed
4. Pull – Produce only when customer is asking for it. JIT delivery of what is needed. It’s a key point in DevOps
5. Perfection – DevOps’s basis is continuous improvement and optimization throughout the life cycle of idea to value

## One DevOps team with Ops extension – more details

For more details on this model, please refer to one of the team formation case in our ENG unit’s DevOps engagement

In this model, team will include Ops representative from all interfaces –Compute, Storage, Network, SCO, Solar winds, ZenOSS, iTAM, Hostname, License and EPIC (ENG team-Cloud Orchestration Engine) Engine Dev team. This will help in planning and assigning work for every team for each Sprint



* With this model issue resolution is much faster as Ops representative from each team work together with Dev team
* Project Ops team forms a backbone for all development teams.
* Sprint planning and deployment strategy can be planned appropriately resulting in smooth deployment to production environment
* Instead of hiring cross skilled DevOps resources , cross-training team members and rotating the roles of developers/ops can be opted
* With this model, project timelines can be reduced as Ops representative from all team work together
* Ops team normally works on the identified/planned user stories/work item. Whenever they get a high priority issues/tickets, they suspend the planned work and work on high priority item. Once it is done, they resume back to the original planned work from the backlog
* The initial % split of following items should be taken based on historical data
  + Planned and unplanned work
  + Dev and ops distribution

. We can fine tune it as and when we get more actual data about planned/unplanned work and dev/ops effort requirement.

## Typical complementing user stories/work items of dev and ops

* + 1. As a developer, for new development

I want to have my working environment up and ready in one hour

So that I can start quickly on the development

* + 1. As a developer, when to deploy a small change to production

I want to have a cosmetic change deployed to production in 1 hour

So that users are not irritated with the cosmetic errors any more

* + 1. As a developer,

I need to understand the environment and configuration where my application will be deployed

So that I can have similar environment for dev and I can take specific actions to ensure that my code is written in accordance with that

* + 1. As a developer,

I should get a the operational requirements / constraints also and not just the user requirement

So that I can consider them in my design and coding

* + 1. As a developer,

I should get a continuous feedback about system performance on production server

So that I can proactively analyze the issue and take corrective actions in subsequent release if performance is going down

* + 1. As a developer,

I should get usage pattern of application and report about client tickets

So that I can proactively analyze the issue and take corrective actions in subsequent release if performance is going down

* + 1. As a developer,

I want to be notified when applications crash or are consuming too many resources OR performance goes down in a production environment

So that I can analyze the same and take corrective action

* + 1. As a developer

I want to be able to request an environment and all supporting environments with networking constructs on demand or self-serviced

So that it is available faster and I don’t need to spend much of energy in doing so

**Typical ops people user stories are**

* + 1. As a ops team member, for new development

I want to understand what developer is working on and what the exact requirements and release plan of the same

So that I can give the operational requirements for the same to developer well in advance

* + 1. As a ops team member, when to deploy a small change to production

I want to understand the deployment goal and then plan things accordingly from ops perspective

So that deployment happens as per the decided goals

* + 1. As a ops team member,

I need to provide the environment and configuration details to developer where the application will be deployed. I also need to help them creating same environment for other stages too

So that developers understand the final environment and design/code system accordingly

* + 1. As a ops team member,

I should understand what developer’s user requirements are and then provide a the operational requirements and constraints

So that developer develops the software accordingly

* + 1. As a ops team member,

I should ensure continuous feedback about system performance on production server

So that the problems can be eradicated in collaborated manner as soon as possible.

* + 1. As a ops team member,

I should provide usage pattern of application and report about client tickets

So that the problems can be eradicated in collaborated manner as soon as possible.

* + 1. As an ops team member.

I want to provide all details when applications crash or are consuming too many resources OR performance goes down in a production environment

So that collectively we can solve the problem

* + 1. As a ops team member

I want to be able to provide mechanisms for an environment provisioning and all supporting environments with networking constructs on demand or self-serviced

So that it is available faster and I don’t need to spend much of energy in doing so

* + 1. As a ops team member,

I need insight into the internal states and behavior of the applications that are deployed

So that I can operate and tune them most effectively.

* + 1. As a ops team member,

I need to build relationships with the developers

So that I can have an open and positive communication/collaboration with them.

* + 1. As a ops team member,

I need to know what parts of the configuration can be tuned.

So that the configurations are at optimal levels

* + 1. As a ops team member,

I want to have an overview of the application architecture

So that I know which applications depend on which services?