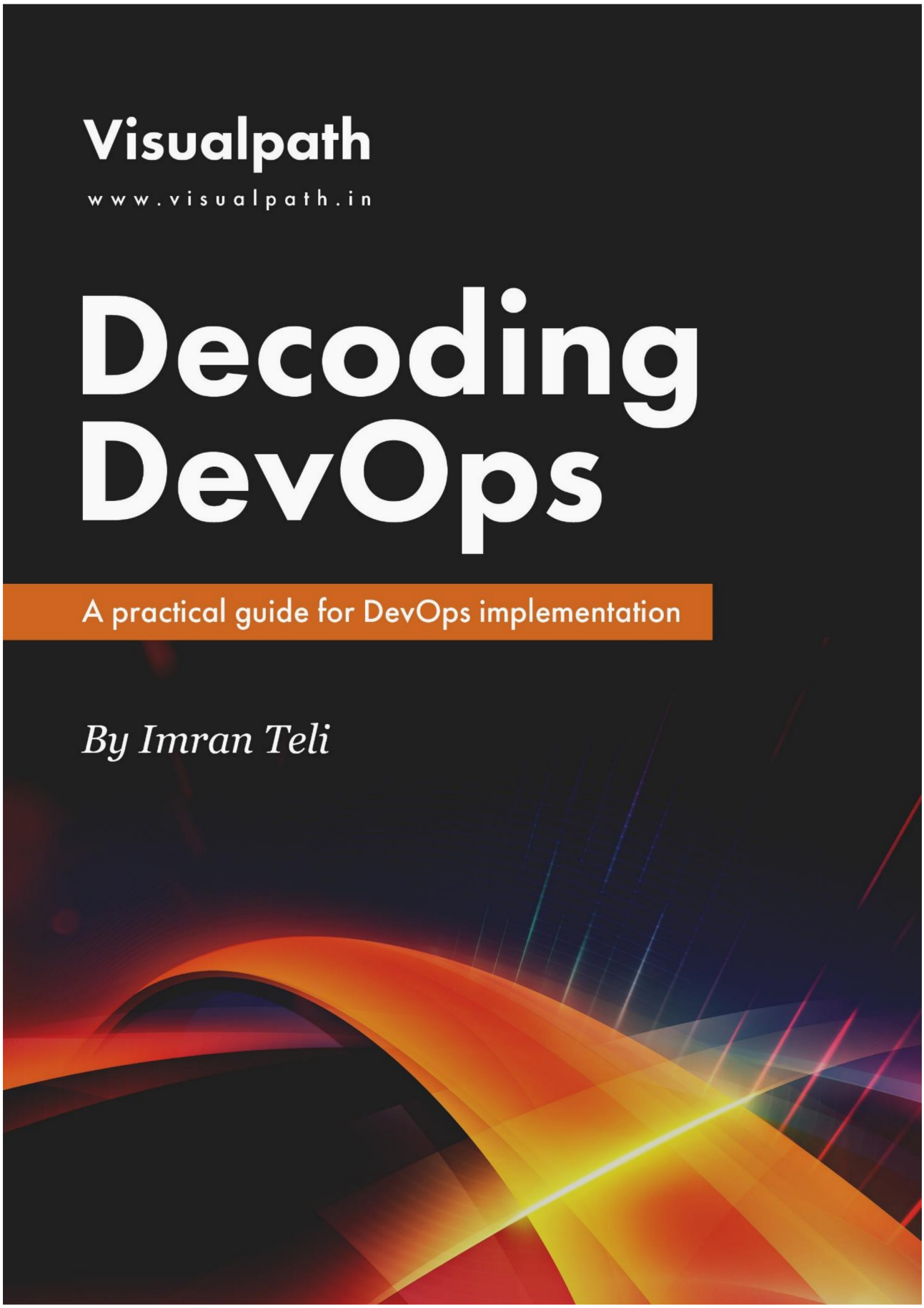


Visualpath Training & Consulting.
Flat no: 205, Nilgiri Block,Aditya Enclave, Ameerpet, Hyderabad, Phone No: - +91-970 445 5959, 961 824 5689 E-Mail ID : online.visualpath@gmail.com, Website : www.visualpath.in.

Decoding DevOps

“A practical guide for DevOps Implementation”

Your personal toolkit for DevOps tasks.

by **Imran Teli**

Co-Authors

- **Gayathri Kilaru** (DevOps Engineer)
- **Kiran P** (DevOps Engineer)
- **Waheed Khan** (Full Stack Developer)

Feedback Email: feedback@visualpath.in
Author’s Email: imran@visualpath.in

VisualPath
www.visualpath.in

Flat no: 205, 2nd Floor, Nilgiri Block,Aditya Enclave, Ameerpet, Hyderabad-16
Phone No : +91 - 970 445 5959, 961 824 5689
E-Mail ID : **online.visualpath@gmail.com**

About Author

Imran Teli has been working as a DevOps Consultant and Architect for a quite a long time. Has a deep understanding of DevOps Processes and tools. Imran has worked in IT industry for 9 years with nearly 5 of those years spent in desigining and managing solutions in some of the most demanding corporate IT environments in the world. Imran has worked his way up from System Admin, all the way upto DevOps consultant for designing resilient, high performance and secured solutions, as well as training people about building and managing such solutions.

Have designed & implemented some of the most complex and mission critical Continous Delivery solutions. Very passionate about sharing the experience and knowledge through training in various technologies in DevOps domain. Current interests include Microservices and Information Security in DevOps domain.

How to contact Author?

You can get in touch with author through Visualpath Training & Consulting for Training in DevOps domain.

www.visualpath.in

Flat no: 205, 2nd Floor, Nilgiri Block,Aditya Enclave, Ameerpet, Hyderabad-16
Phone No : +91 - 970 445 5959, 961 824 5689
E-Mail ID : **online.visualpath@gmail.com**

PREFACE

Who should read this book?

Decoding DevOps is written for anyone who wants to learn DevOps tools and techniques. Its main focus is in doing rather than reading.

That being said, Basic knowledge of Linux and networking is required to grasp concepts and techniques in this book.
While its not necessary to read this book cover-to-cover, we make an attempt to cover the topics in logical order.

That being said, here are few short-cuts:

- If you are a system admin and want to learn scripting, check out Bash scripting & Python scripting.
- If you create virtual machines on your laptop/desktop and want to automate your vm lifecycle, checkout Virtualization & Vagrant chapters
- If you want to learn AWS SysOps services, check out Amazon Web Services chapter.
- If you are Developer and want to learn Build & Release and Continuous Integration, check out chapters Maven & Continuous Integration.
- If you want to learn new way of automation through Configuration Management tools, check out Ansible & Puppet chapters.
- If you want to learn containers and dockers for microservices, checkout Docker and Kubernetes chapters.

What's in this Book?

The book begins by explaining DevOps culture and process. It talks about the problems in delivering software to users and then explaining how DevOps culture solves this problem.
First chapter also talks about DevOps lifecycle and the tools that should be used to create that lifecycle.
Scripting with Bash and Python language is covered if you are interested in doing automation with scripts.
Automating virtual machine lifecycle on your local system is covered which has become so much necessary in day to day practice in IT industry. Vagrant tool helps you doing it and you don't have to waste your time in creating vm manually and installing OS on it.
IT industry is slowly moving their infrastructure to Cloud and Amazon Web services talks about how to use AWS services and create your own infrastructure on cloud. Here we are focusing on AWS sysops services that is used by Sys Admins and DevOps. AWS has lot many services and features for Developers and other DevOps practices which is out of scope for this book to cover.

Web Application architecture, Maven build tool and Software Development process are covered. If you are coming from Sys admin background then you will understand how Development process happens. Also, if you are not aware of different kind of services that is used in Web application platform you will learn general Architectures and services for web apps.

Jenkins tool in covered in detail which is the most famous continuous integration tool and also the most important DevOps tool. Integration of Jenkins with various other tools is also covered. Learning Jenkins is important for Developers, Build & Release, Testers, Sys Admins & DevOps engineers ofcourse.
Ansible and Puppet is covered as a part of configuration management tools tutorial. It is highly important as these are the most famous automation tool used by DevOps and Sys admins around the world.
Lastly Dockers and Kubernetes is explained in detail with all the commands and implementation. Docker is highly important for Microservice architecture and is going to revolutionize the IT industry big time. Docker or containers is important to learn for both Developers and Operations team. Highly important for DevOps engineers
How to contact us?

We'd like to hear from you
As you use this book and try exercises, we invite you to comment and feedback. We always take feedback positive or negative and learn from our mistakes or cherish positive feedbacks. We hope to continue to do this.

This book was written quite quickly. There would be places where we would have made mistakes or errors, your feedback will help us make the next version better.
It was not possible to cover everything in DevOps in one book but we have given our best shot. You can send us some tips, tools or techniques information which we will include in next version or in another book which covers DevSecOps, Microservices, Log management and some Real time POC in DevOps domain.

Please let us know what we need to correct or add; share your insights; and help us create a resource that will server uou better. You can do so by sending email at **feedback@visualpath.in**

Feedback Email: feedback@visualpath.in
Author's Email: imran@visualpath.in

Acknowledgements
I'd like to thank each one of them who helped and inspired me in writing this book. Some of my students have researched and documented scripts and tools implementation, I have included their name wherever their contributation is done in the book. My co-authors have put great deal of time and effort in creating documents on Various topics without them it was not possible to complete this book on time. I'd like to especially thank Eswar for thinking that I should write this book in the first place. Last but not the least my Wife Ziya Teli who spent some sleepless nights in formatting, designing book cover and doing spellchecks for this book because this book was converted from libreoffice to MS office.

I. DevOps Introduction

1. What is DevOps?

I know that's the first answer everybody would be looking for and I will not try to define it. Because so many people have done it already before me and guess what there are many different definitions for it. Everybody in the IT industry is talking about it. Organization are running post to pillars to hire DevOps Engineers and there is a great demand for it.

Some say its automation but some say it’s all about culture, both this belief contradicts with each other.

Well if its automation then system admins are doing automation from ages and there they used Scripting languages and some tool to achieve it but we did not call it DevOps back then, we just said automation. So, then you would say it’s the culture but if it’s just the culture then why so many automation tools? You would be feeling now that I am confusing you but trust me I am going to prove some points later, just keep reading.

If you are reading this you would somehow related to software industry. Yes, software industry, its only goal is to make software's and deliver it to the user. We can say from the very beginning software industry is divided into two parts and those two parts are Development and Operations.
Development focuses on creating and testing software's.
Operations is focussed on delivering those software's to user in form of a website or as an installable software. Once delivered we maintain the software, we deliver new features to our users and make sure the software's stay up and running and healthy.

2. Job Roles & their Motto.

In Development, we have distinct roles like Developers, Software testers(QA), DB Developers & Architects. Their aim is to develop all the latest and greatest features in the software's, Rapidly or quickly.
In Operations, we have roles like System Admins, Cloud Engineers, DB admins & Security professionals. Here the aim is to keep the systems up and running all the time. Systems on which the software is hosted, like your websites and databases hosted on some servers.

You would have understood by now that both the parties have different aims and goals. One focusses on Quick Change and other focusses on Stability. These are poles apart, if we make quick changes (adding new features continuously) then stability becomes an issue. A system that’s changes continuously will have issues with stability. It’s also true the other way around.

We are living in a world where there are frequent changes to our software's and apps. Think about it if you are using a software with some old features and some other software comes into the market with the latest & greatest feature you will ofcourse migrate to the new software. So that means if an organizations Dev and Ops team does not give you latest and greatest feature with stability they may lose the business.

So far, I have established few points that I will list below.

- Developers aim to create latest features quickly and rapidly.
- Operations aim is to keep systems stable.
- Quick changes are the user demand.
- User also needs stable software or Apps.

DevOps whole and sole objective is to Deliver latest and greatest features to the user with stability. It's no more about just creating new features it's also about delivering those features to the user otherwise what's the point of creating if we cannot deliver it on time.

So how does DevOps solve that problem? To understand it first we need to understand the development procedure and then we will focus on the operations.

3. Software Development Processes

Development process is explained in detail in later chapters here we will try to keep it to the minimum.

Development team uses some Software development model to create the software. In layman terms, these software development models are set of rules that everybody in the team follows to get the things done.

Waterfall Model

There is Waterfall model which is a traditional model and does not fit well in today's fast-moving world.

In Waterfall model, the following phases are followed in order:

1.System and software requirements: captured in a product requirements document

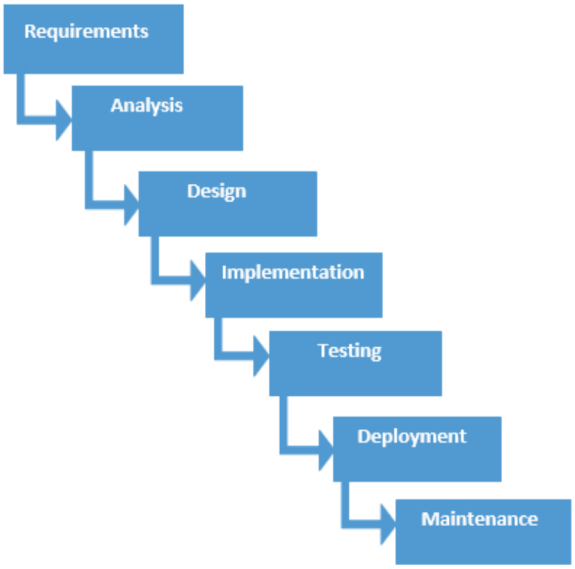
2.Analysis: resulting in models, schema, and business rules

3.Design: resulting in the software architecture

4.Coding: the development, proving, and integration of software

5.Testing: the systematic discovery and debugging of defects

6.Operations: the installation, migration, support, and maintenance of complete systems



Thus, the waterfall model maintains that one should move to a phase only when its preceding phase is reviewed and verified.

This model is good for the Operations team as they get the whole software developed at once which they can deploy and maintain. New changes will also be less frequent and does not put so much burden on the Operations team.

But this model does not scale well with the current fast-moving world. There are so many downfalls with this model and most of the Development is happening with Agile model now.

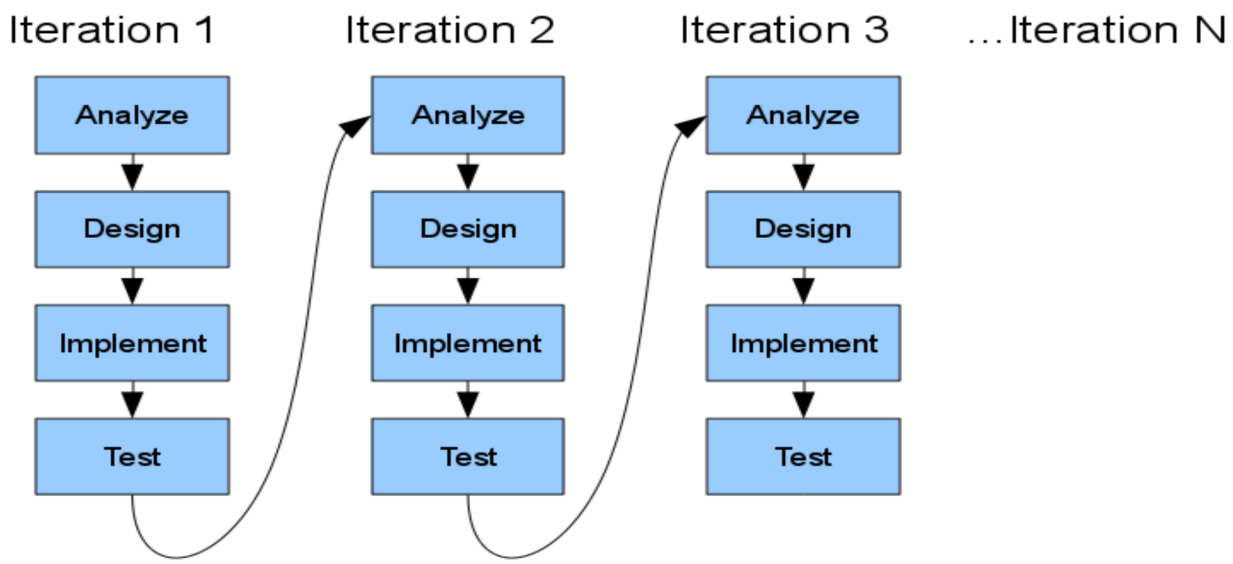
Agile Model

Agile model developed software in small iterations instead of developing entire software at once.

Entire products feature list is divided into multiple list of features. For example, if there are 50 features in a software we can create 5 lists of 10 features each. Now developers will work on 10 features at a time, create and deliver those 10 features in first iteration and continue with the rest of the features until you get the final product.

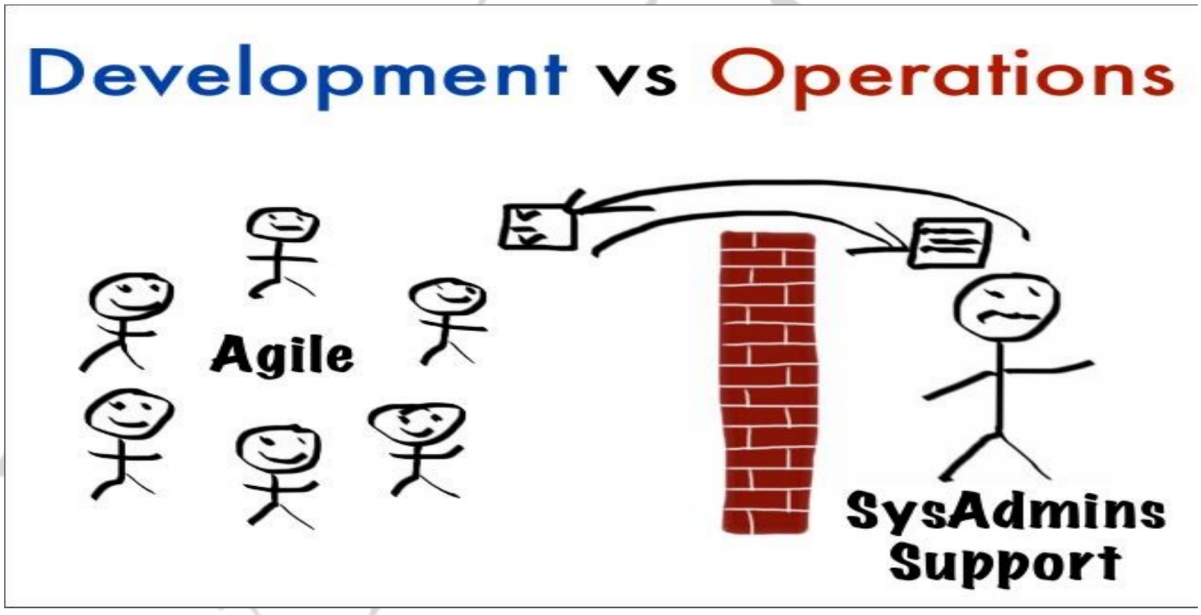
Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.



Now we are not just talking about creating but also delivering it to different environments like Dev, QA, Staging, UAT and Prod. Now this puts lot of burden on the Operations team as they must continuously deliver these changes on multiple environments. By the way these different environments are just some group of servers owned by different teams like QA is owned by software testers where they test the software.

The general approach that Developers take is that once they are done creating a new feature they will send a procedural document to the operations team explaining how to deploy it.

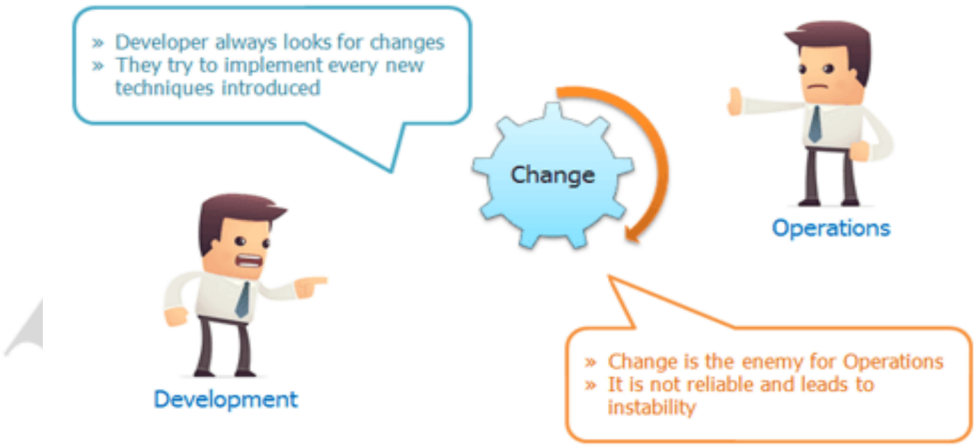


Developers test it on their machine and feels that it should work same in production also. But production systems are different in design as there would be multiple servers for webservice, database service and backend services secured by firewalls and NACL. There would be all stable OS and software with a different version that of Developers systems. Development servers are simple one machine and all the services are generally deployed in one server.

4. The Problem

After following the procedural document and having their own skills and knowledge operations will deploy it to production. This is where the problem comes in, the deployment may fail, failing the entire service. This happened because of lack of communication between Dev and Ops team. Dev does not understand Ops part and the reverse is also true.

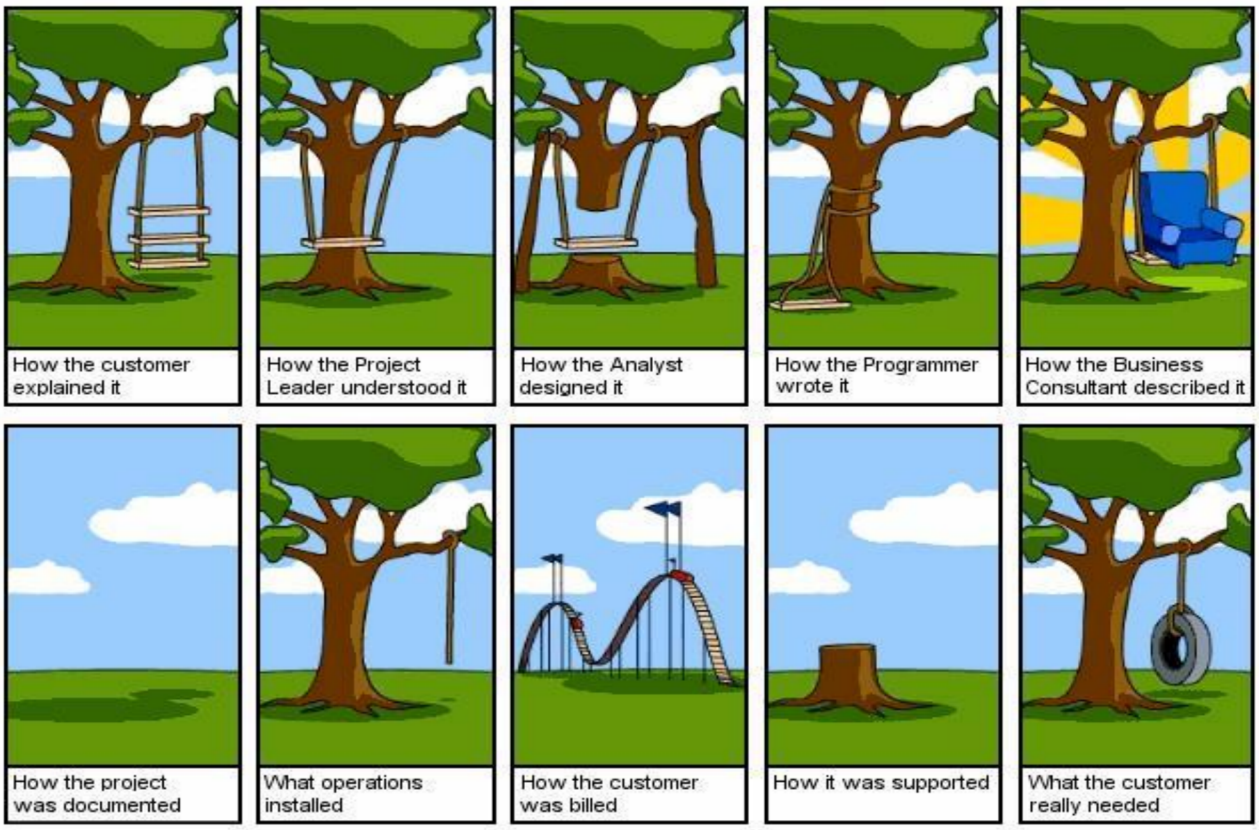
So, Ops feel that frequent changes like that may break the system and Dev feels that there is too much restriction on delivering latest changes. We also need to think about security here. Security testing is done before it goes to production. This entire Delivery procedure is slow and manual most of the time.



Think about quick changes now through agile model, it’s not helping the operations team to deliver the code faster. So, no matter how agile development is Operations is still Waterfall.

If you think about it for a while you will understand that it’s not the technological problem but the culture problem. Both the parts in Software Industry follow different culture. If this culture is not changed then we will not be able to deliver better feature quickly to the users.

A Very famous joke describes the communication problem.



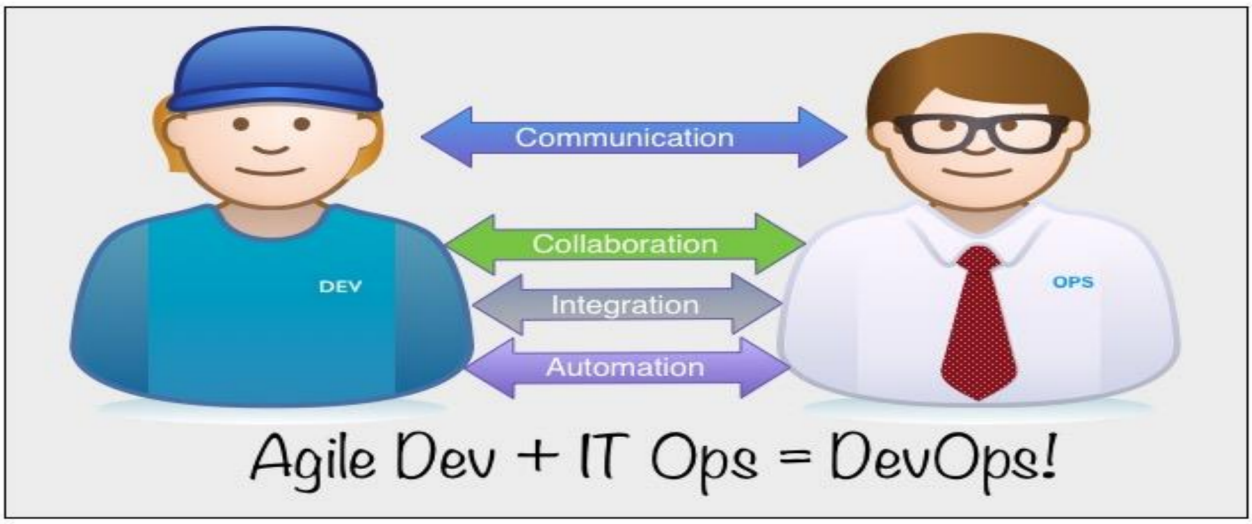
5. Enter the DevOps

DevOps solves this problem by changing the culture and making it one culture Dev+Ops.

There would be one team, DevOps, with one goal, Rapid delivery with stability.

But how?

First step is to establish communication and collaboration between Dev and Ops. Dev must understand the Ops part and Ops must understand the Development Procedure.



DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.

We have seen previously in waterfall and agile lifecycle, that Development and Operations team are separate, they work separately in their own silos and have very different motto.

6. DevOps LifeCycle

DevOps Lifecycle includes Development and Operations teams working together. As Developers work on their agile iterations, Ops must work in setting up systems and automating the procedure of deployment. Automation is the key factor here, because agile model gives code repeatedly to deploy it on systems, that's going to be continuous release of code and that must be continuously deployed to many servers in Dev, QA, Staging & Production environments.

If the code deployment process is not automated then ops team must manually do the deployment. Deployment may include below mentioned procedures.

- Create servers if they don't exist (On cloud or virtual env).
- Install and setup prerequisites or dependencies on servers.
- Build the software from raw source code (If not done by Developers).
- Deploy software to servers.
- Do config changes to OS and software.
- Setup Monitoring.
- Feedback & Report.

Note: All the above process may be less or more depending on the kind of deployment. We will discuss this in later chapters.

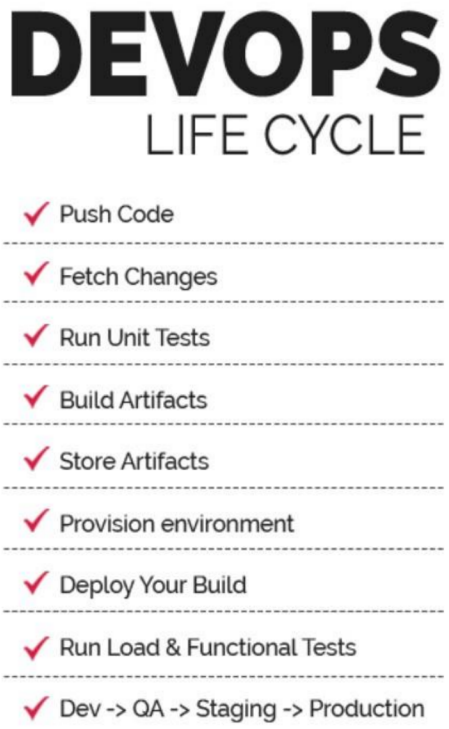
As soon as we get a new code change, it must be deployed to Production or at least staging servers.

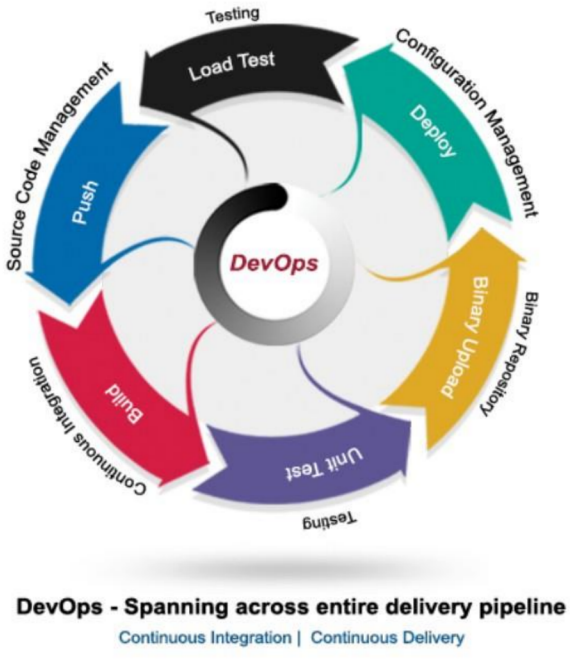
For this, all the process must be automated, we must automate first Build and Release process which includes.

- Developers push the code in a centralised place.
- Fetch the developers code.
- Validate code.
- Build & test code
- Package it into distributable format(software/artifacts).
- Release it.

Next phase is to **deploy** this released software to servers, which we discussed already before this.

Combining this Build & Release with Deployment process gives us the DevOps Lifecycle which is fully automated.





DevOps Engineers must automate all the above process it should be so seamless that when developers push their code to a central repository it should be fetched and run through all the above process and sends it to production systems.

As I say, "**From Code to Prod"**.

What is continuous Integration.

Developers will push their code several times in a day to a central repository, every time there is code change it should be pulled, built, tested and notified. There will be continuous code change so continuously we must do these steps. That's why its Continuous integration. We have a separate chapter for this where it will be discussed in detail. As of now you can understand from above diagram that from step **1 to step 5 is CI**.

What is Continuous Delivery?

After CI, we should be also able to deliver our code changes to all the servers in different environments like Dev, QA, Staging. It should be automatically delivered to QA servers where testers will do functional tests, load tests etc. After it passes the QA tests it should automatically deliver the code to staging area where customer or some set of users can check the changes and give approval to deploy it to Production.

As per Wikipedia.

Continuous delivery and DevOps are similar in their meanings and are often conflated, but they are two different concepts. DevOps has a broader scope, and centres around the cultural change, specifically the collaboration of the various teams involved in software delivery (developers, operations, quality assurance, management, etc.), as well as automating the processes in software delivery. Continuous delivery, on the other hand, is an approach to automate the delivery aspect, and focuses on bringing together different processes and executing them more quickly and more frequently. Thus, DevOps can be a product of continuous delivery, and CD flows directly into DevOps.