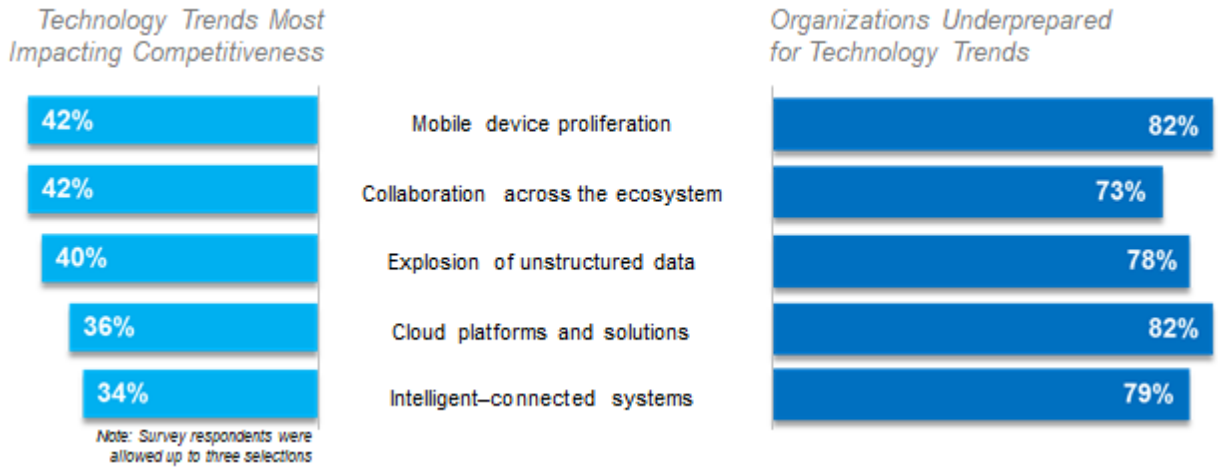
MODULE VI

DEVOPS

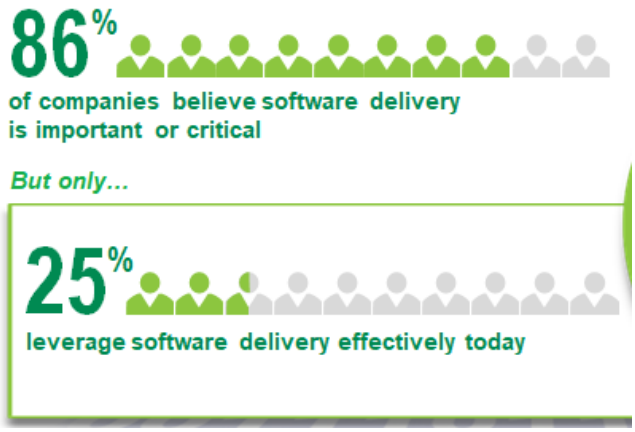
INDUSTRY TRENDS

Top technology trends are impacting how organizations compete, yet approximately 75 percent of companies are underprepared

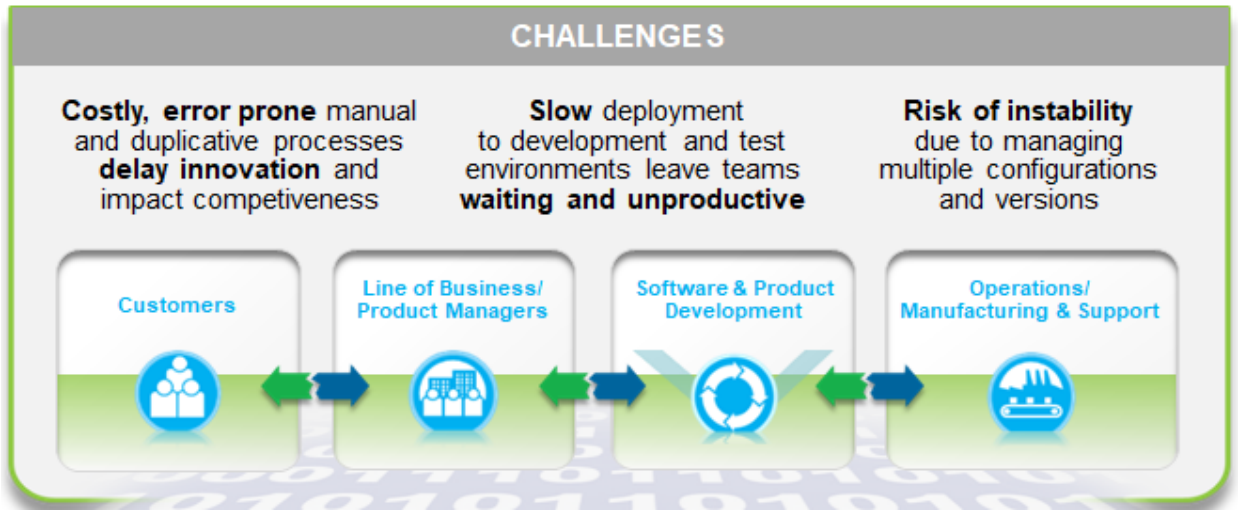


Business innovation is increasingly being delivered via software
Rapid pace of change and the digitization of business drives the need for agility

Organizations that effectively leverage software innovation outperform their competitors... *yet few are able to deliver it effectively*



Expand the lifecycle to manage the challenges affecting individual creativity, team productivity, and ultimately the bottom line



Market shifts require a fundamental change to the way businesses approach the development lifecycle

Evolving Customer Expectations
Consumerization of tools and products drives demand for higher quality and faster delivery

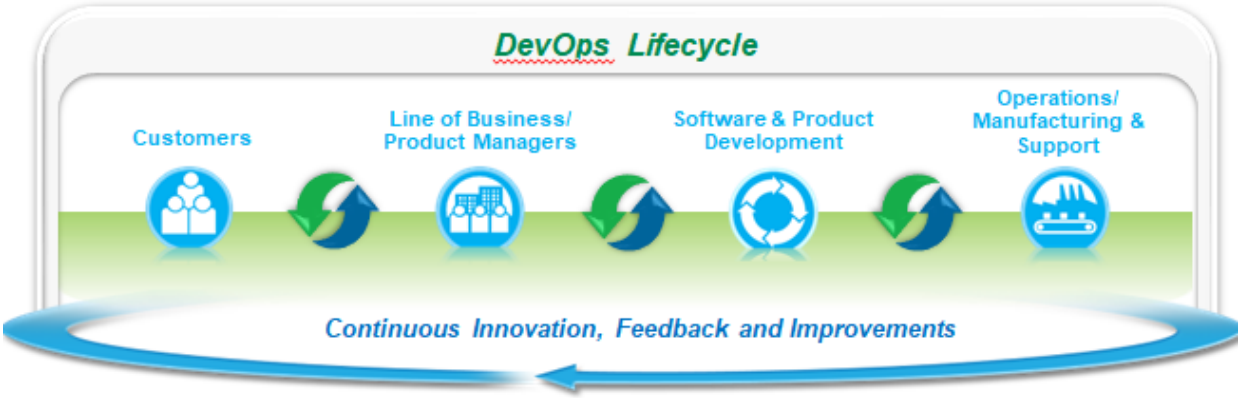
Changing Business Environment
Increasingly volatile economic and regulatory environments drive the need to reduce risk

Multi-sourcing/ Supply Chain
Increased outsourcing drives the need for governance

An approach for continuous delivery of software-driven innovation

dev·ops

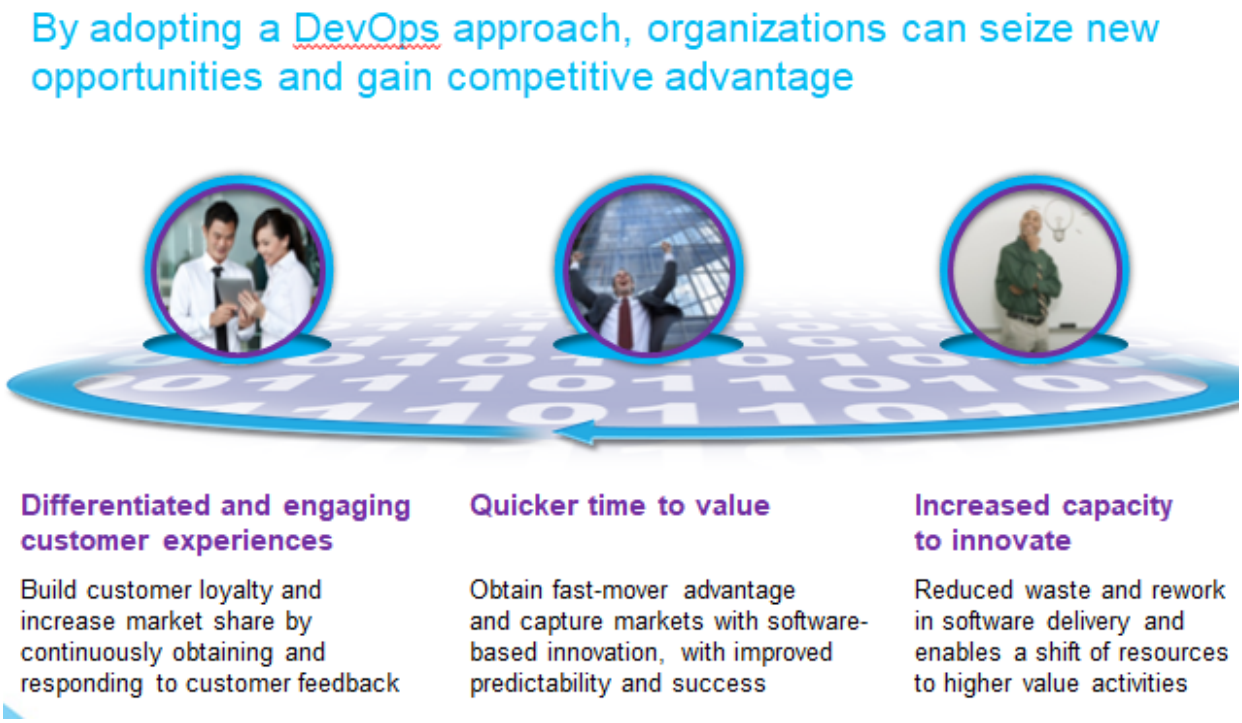
Enterprise capability for continuous software delivery that enables clients to seize market opportunities and reduce time to customer feedback



Reduce time to obtain and respond to customer feedback

Accelerate software delivery

Balance speed, cost, quality and risk



INTRODUCTION TO DEVOPS - A UNIFIED PROCESS BETWEEN DEVELOPMENT AND OPERATIONS

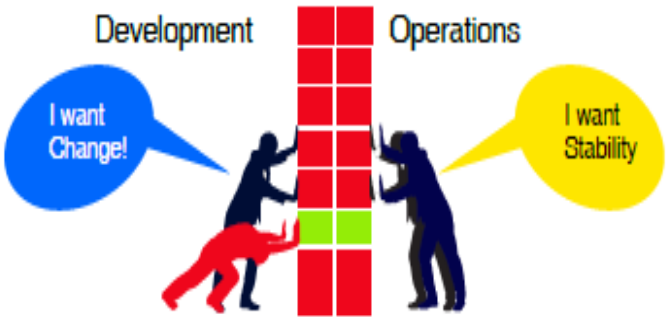
Patrick Debois, who's often called "the father of DevOps", coined the word "DevOps" in 2009. As the word depicts, it was formed by combining two words: "development" and "operations". DevOps is a collaborative way of developing and deploying software.
DevOps (development and operations) is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) operation professionals.

- DevOps is an approach based on agile and lean principles in which business owners, development, operations, and quality assurance team collaborate to deliver software in a continuous stable manner
- DevOps is an environment that promotes cross practicality, shared business tasks and belief

- DevOps is a movement that improves IT service delivery agility
- DevOps is a culture that promotes better working relationship within the company
- DevOps is a set of practices that provides rapid, reliable software delivery

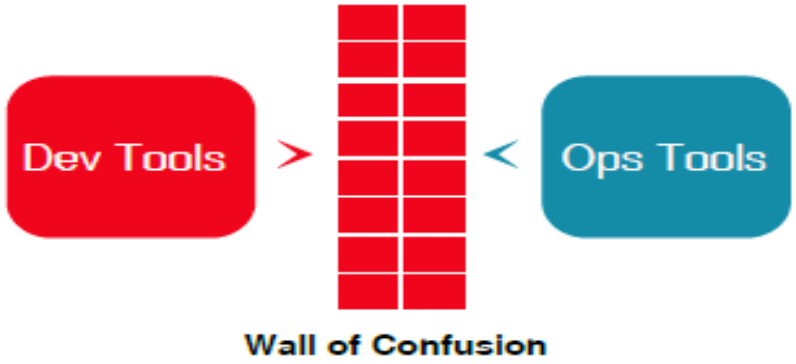
Need for Devops

- Developers always want to deliver changes as soon as possible.
- Operations want reliability and stability.
- Lee Thomson describes this as a wall of confusion between development and operations.
- Wall of confusion not only exists between the mindsets of the two teams but also with the tools they use.
- Development uses some tools and operation uses some other tools to perform the same stuff.



Wall of Confusion

- DevOps break down the walls between development and operations team, unifying development to operations for better, faster outcomes.
- DevOps Focuses both the Apps team's drive for agility responsiveness and the operators's concern with quality and stability on the ultimate goal of providing business value



The adoption of DevOps is being driven by factors:

- Use of agile and other development processes and methodologies
- Demand for an increased rate of production releases from application and business unit stakeholders
- Wide availability of virtualized and cloud infrastructure from internal and external providers
- Increased usage of data center automation and configuration management tools

Devops Benefits

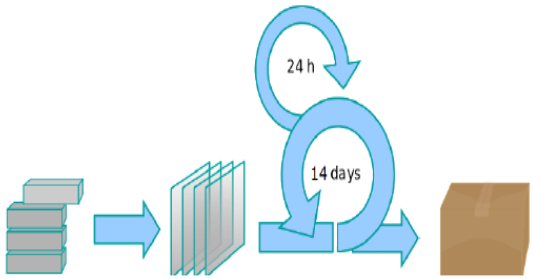
- Improved quality of software deployments
- More frequent software releases
- Improved visibility into IT process and requirements
- Cultural change collaboration/cooperation
- More responsiveness to business needs
- More agile development
- More agile change management process
- Improved quality of code

DevOps vs. Agile

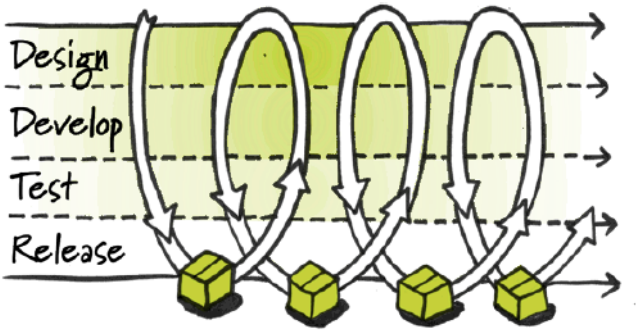
- DevOps is especially complementary to the Agile software development process.
  - extends and completes the continuous integration and release process by ensuring that code is production ready and will provide value to the customer
- DevOps enables a far more continuous flow of work into IT

Operations.

- If development delivers code every two weeks but it's deployed only every two months, customers don't get value and the deployments often result in chaos and disruption.



Agile



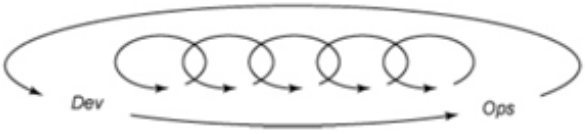
Continuous Delivery

DevOps 3 Basic Principles

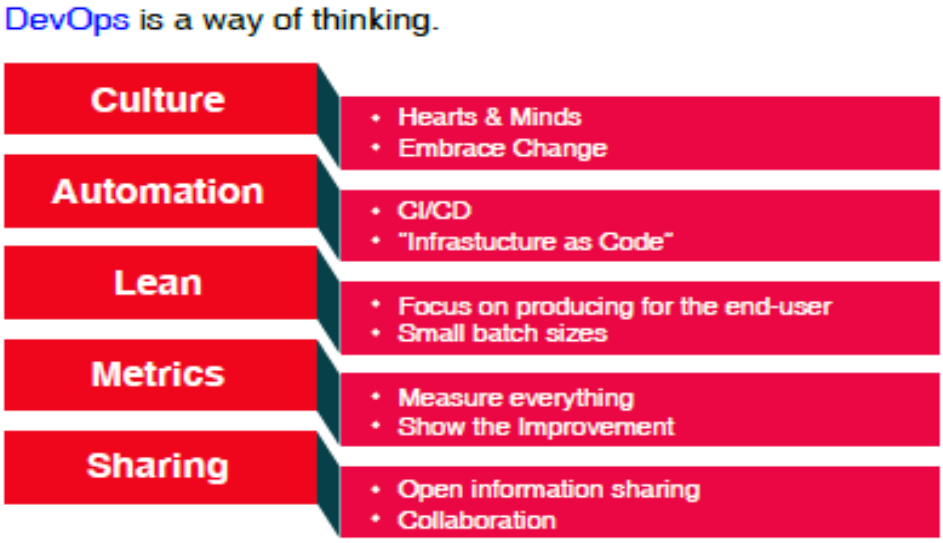
- System thinking

- Amplify feedback loops

- Culture of continual experiment and learning



How does Devops works?



CALMS Model

5 Basic Principles of Devops

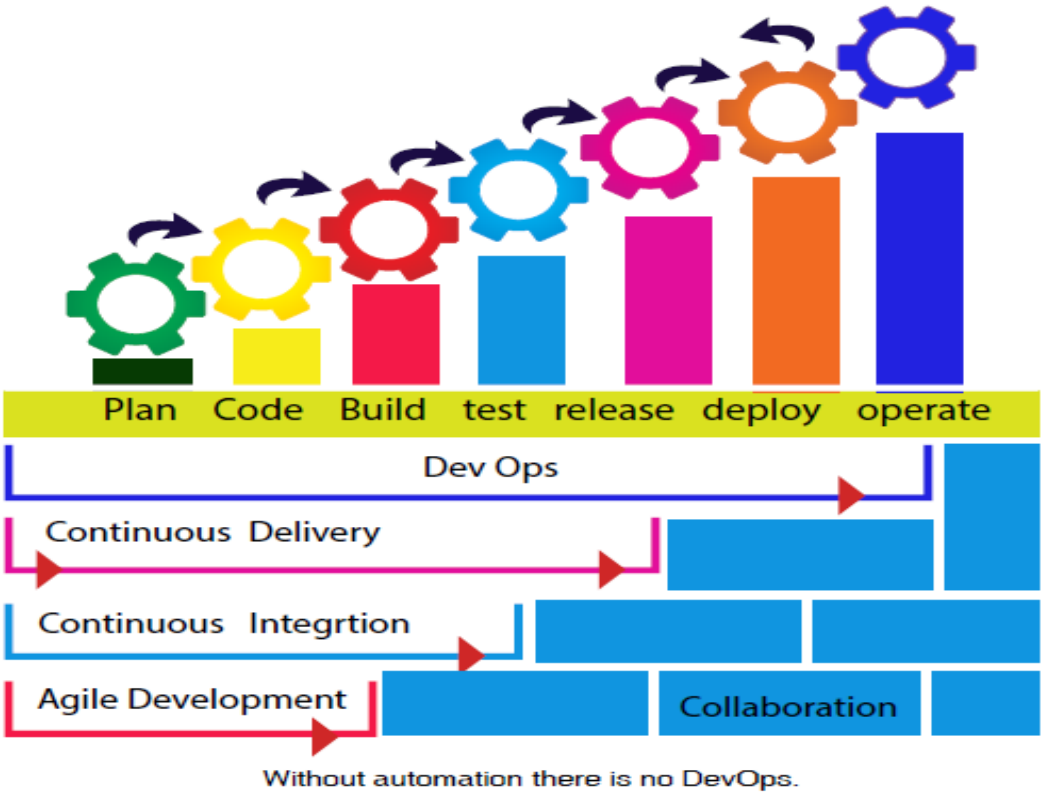
1. Culture:- Eliminate the blame game, Open post-mortems, Feedback,Rewarding failures
2. Automation:- Continous Delivery, Monitoring, Configuration Management
3. Lean:-Business value for end user
4. Metrics:- Performance Metrics, Logs, Business goals Metrics, People Integration Metrics, KPI
5. Sharing:- Ideas, Plans, Goals, Metrics, Complications, Tools

- Believes in the agile mantra “People over Process over Tools”.
- With the right people, we establish the right process and choose the right tools to deliver the end results
- People – Communication & Collaboration
- Process – Source Control Check-ins, Code Review, Code Quality, Change Control
- Tools –
    - For Continous Delivery
        - achieve by the combination of **Continous Integration, Continous Deployment** and **Continous Testing**

- For Continous Monitoring

7 Cs of Devops

- Communication
- Collaboration
- Controlled Process
- Continuous Integration
- Continuous Deployment
- Continuous Testing
- Continuous Monitoring



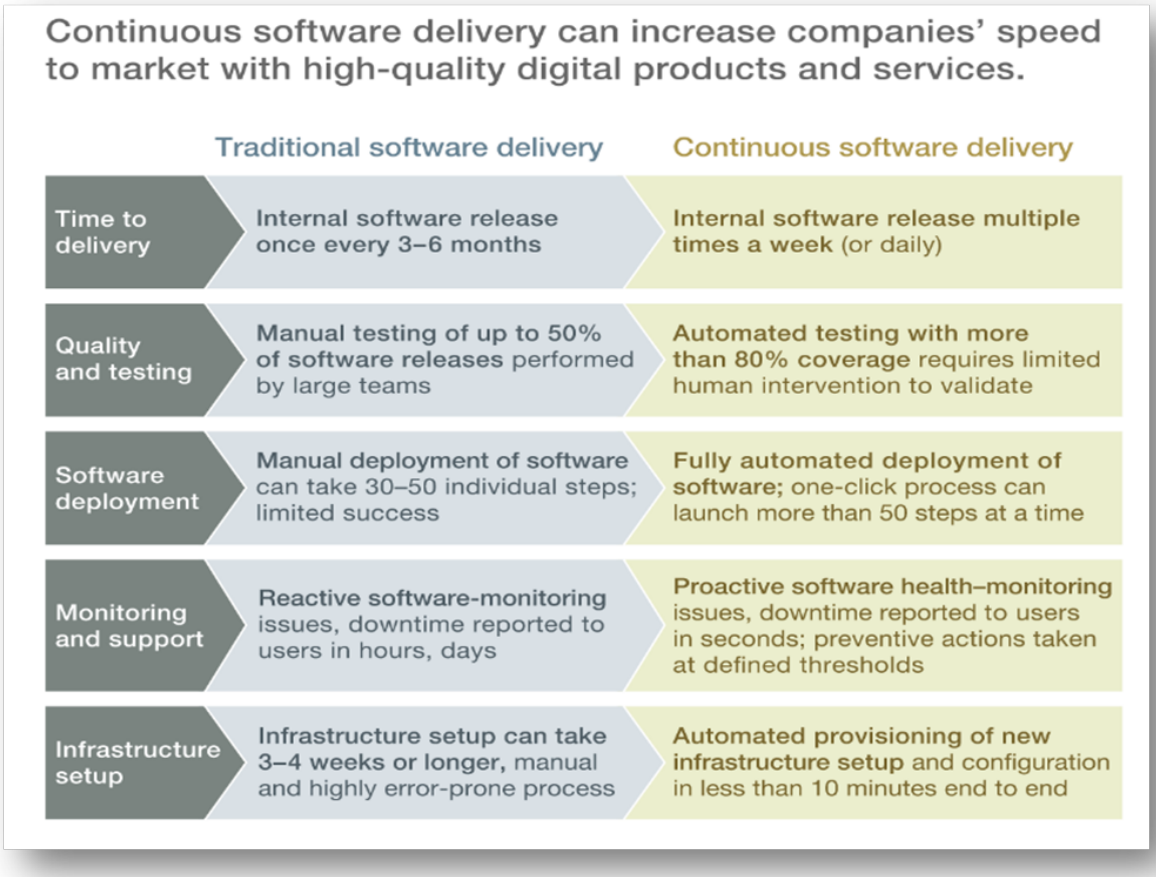
Without automation there is no DevOps.

- Automate Provisioning - Infrastructure as Code
- Automate Builds – Continuous Integration
- Automate Deployments – Defined Deployment Pipeline and Continuous Deployments with appropriate configurations for the environments
- Automate Testing – Continuous Testing, Automated tests after each deployment
- Automate Monitoring – Proper monitors in place sending alerts

- Automate Metrics – Performance Metrics, Logs

Deployment Pipeline

- DevOps is enabled through the Deployment Pipeline:
  - Build
  - Deployment
  - Test
  - Release
- The purpose of the deployment pipeline:
  - Visibility: All aspects of the delivery system are visible to all team members promoting collaboration.
  - Feedback:Team members learn of problems as soon as they occur so that issues are fixed as soon as possible.
  - Continually Deploy: Through a fully automated process, you can deploy and release any version of the software to any environment.



CONTINUOUS INTEGRATION (CI)

- Continuous integration in DevOps cultures because DevOps emerged from Agile culture
- Software engineering practice in which isolated changes are immediately tested and reported on when they are added to a larger code base.
- Goal of CI is to provide rapid feedback so that if a defect is introduced into the code base, it can be identified and corrected as soon as possible.
- The usual rule is for each team member to submit work on a daily (or more frequent) basis and for a build to be conducted with each significant change.
- Forcing developers to integrate their work with other developers frequently— atleast daily—exposes integration issues and conflicts much earlier than is the case with waterfall development.
- To achieve this benefit, developers have to communicate with each other much more frequently
- That seed of open, frequent communication blooms in DevOps

CONTINUOUS TESTING

- "The cost of quality is the cost of failure."
- Continuous testing is not just a QA function.
- In fact, it starts in the development environment.
- The days are over when developers could simply throw the code over the wall to QA and say, "Have at it." In a DevOps environment, everyone is involved in testing.
- Developers make sure that, along with delivering error-free code, they provide test data sets.
- They also help test engineers configure the testing environment to be as close to the production environment as possible.
- On the QA side, the big need is speed.
- Test engineers meet the challenge of quick turnaround by not only automating much of the test process but also redefining test methodologies:
- *Rather than making test a separate and lengthy sequence in the larger deployment process, Continuous Delivery practitioners roll out small upgrades almost constantly, measure their performance, and quickly roll them back as needed.*
- Operations function has an important role to play in testing and

QA:

- *Operations has access to production usage and load patterns.*
- *Patterns are essential to the QA team for creating a load test that properly exercises the application.*
- Operations can also ensure that monitoring tools are in place and test environments are properly configured.
- They can participate in functional, load, stress, and leak tests and offer analysis based on their experience with similar applications running in production.

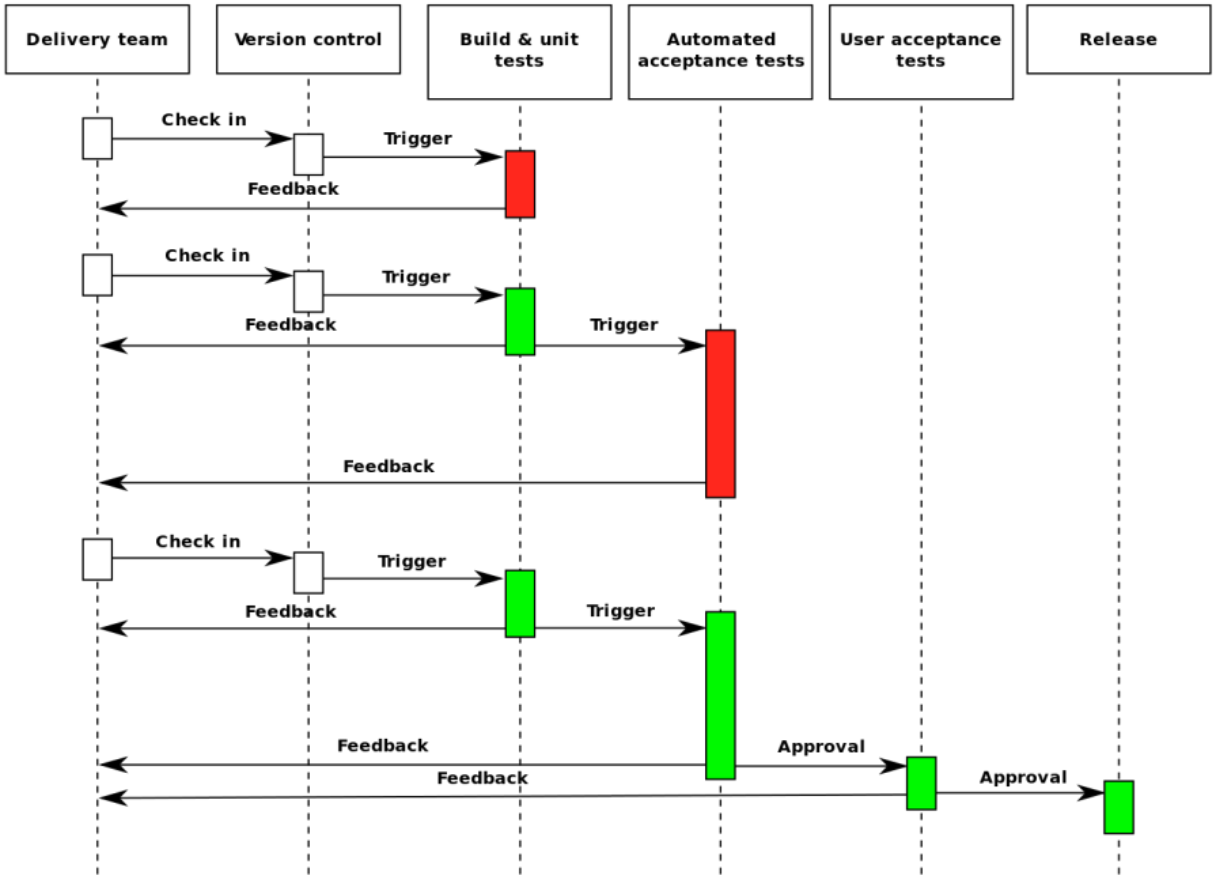
Continuous Testing - Advantages

- The test function in a DevOps environment helps developers to balance quality and speed.
- Using automated tools reduces the cost of testing and allows test engineers to leverage their time more effectively.
- Most importantly, continuous testing shortens test cycles by allowing integration testing earlier in the process.
- Continuous testing also eliminates testing bottlenecks through virtualized dependent services, and it simplifies the creation of virtualized test environments that can be easily deployed, shared, and updated as systems change.
- These capabilities reduce the cost of provisioning and maintaining test environments, and they shorten test cycle times by allowing integration testing earlier in life cycle.

CONTINUOUS DEPLOYMENT

- "Continuous Delivery is nothing but taking this concept of continuous integration to the next step."
- Instead of ending at the door of the development lab, continuous integration in DevOps extends to the entire release chain, including QA and operations.
- The result is that individual releases are far less complex and come out much more frequently.
- The actual release frequency varies greatly depending on the company's legacy and goals.
- For example, one Fortune 100 company improved its release cycle from once a year to once a quarter—a release rate that seems glacial compared to the hundreds of releases an hour achieved by Amazon.

- Exactly what gets released varies as well. In some organizations, QA and operations triage potential releases: many go directly to users, some go back to development, and a few simply are not deployed at all.
- Other companies—Flickr is a notable example— push everything that comes from developers out to users and count on real-time monitoring and rapid remediation to minimize the impact of the rare failure.



Tools for Adopting DevOps

Category	Example Software Tools
Configuration Management	Subversion (SVN), git, Perforce, PassPack, PasswordSafe, ESCAPE, ConfigGen
Continuous Integration	Jenkins, AntHill Pro, Go. Supporting tools: , Doxygen, JavaDoc, NDoc, CheckStyle, Clover, Cobertura, FindBugs, FxCop, PMD, Sonar,
Testing	AntUnit, Cucumber, DbUnit, Fitnesse, JMeter, JUnit, Selenium
Deployment Pipeline	Go, AntHill Pro
Build and Deployment Scripting	Ant, NAnt, MSBuild, Buildr, Gradle, make, Maven, Rake
Infrastructure and Environments	AWS EC2, AWS S3, Windows Azure, Google App Engine, Heroku, Capistrano, Cobbler, BMC Bladelogic, CFEngine, IBM Tivoli Provisioning Manager, Puppet, Chef, Windows Azure
Data	Hibernate, MySQL, Oracle, PostgreSQL, SQL Server, SimpleDB, SQL Azure, MongoDB
Components and Dependencies	Ivy, Archiva, Nexus, Artifactory, Bundler
Collaboration	Mingle, Greenhopper, JIRA

CONFIGURATION MANAGEMENT

Managing all the configurations of the environments that the software application hosts upon.

We have different environments throughout the SDLC in DevOps starting with Unit testing, integration testing, system testing, acceptance testing and end-user testing.

Configuration management is the automated process to manage all the configurations of each of these environments.

Difference between traditional Configuration management and DevOps configuration management

In traditional configuration management methods, the team used to manage these configurations of various environments via formal documentation wherein each of the configurations used to be recorded in the documents and configuration team or manager used to handle the version control of these documents.

And as and when it undergoes changes, he would also take the

responsibility of setting up the environment and managing the configurations manually

Now in DevOps, typically, all these configuration management processes are pretty well automated and the configurations are encapsulated in the form of code or scripts and controlled through the version control tool. i.e. Operations team is integrated with Development in managing the environments through the single version control tool.

Key highlight of configuration management in DevOps is delivering,

1. Infrastructure as a code
2. Configuration as a code

Infrastructure as a code

It is defining the entire environment definition as a code or a script instead of recording in a formal document. Environment definition generally includes, set up of servers, configuring networks, and setting up of other computing resources, which are a part of the IT infrastructure set up. So, all these details would be written out as a file or in the form of a code and checked into version control tool.This script or code, which is checked into the version control would become the single source of defining the environment or even updating those environments.

Example, if we have to add a server to the specific environment, all that we would do is to update these information in to the environment scripts and run the delivery pipeline, instead of manually going and spinning out a new environment with the added server or seeking the help of the system admin people to do this.

So, the beauty here is that the developer or tester need not be a system admin expert to set up their servers for development or testing activity.

Configuration as a code

Configuration as a code is nothing but defining all the configurations of the servers or any other resources as a code or script and checking them into version control.
These configuration scripts which are checked into the version control are run as a part of the deployment pipeline in order to set up the infrastructure and its configurations in an automated fashion.

Well, defining configurations includes parameters that define the recommended settings for the software to run successfully. Or a set of commands to be run initially to set up the software application. Or even it could be configurations of each of the components of the software that are to be set, or specific user roles, user privileges etc.,
Simple Example would be to set the feature toggles, where default values are set up as a part of the configuration parameter.
Adding another port to a firewall would be another Example, which can be updated in the script and later these scripts are run as a part of the delivery pipeline.
Several tools are available to carry out the infrastructure automation in the market. Few of them are Chef, Puppet, Terraform, etc., Chef and Puppet are ruby based configuration management tool, whereas Terraform is a provisioning tool.

Benefits of configuration management in DevOps

#1) This helps the team to manage the changes to the servers and configuration in an automated fashion and helps to debug quickly if anything fails, within a short time span and also allows to rollback quickly to the previous version, without causing any interruptions to the customers.

#2) Since these scripts are located on the central server and everyone in the team knows what is there in each of these scripts and what are the changes made in each of these versions. This also enables the team, to go back to the older version, if there is any problem in the latest versions. Imagine, if there is a server crash, how much time it would have taken to manually restore it. And now by defining infrastructure as a script and version controlling, we can immediately restore by going to the earlier version.

#3) Managing the configurations as a code also prevents someone from making changes to the system accidentally and prevents any damages causing later in the production.
Automation of configuration management not only has benefited in time-saving but also in eliminating such human errors and improving the quality.

Configurations delivering as a code has removed the dependency on a single person or a team called config manager or config team. Development team need not have to wait for the config team to come and

fix any infra or config problem.

RELEASE MANAGEMENT

Managing and maintaining which version of the software or the components are deployed to which environments, when and how it was deployed.
It is not just the responsibility of the Release Manager, but the responsibility of the entire team in DevOps.

Benefits of Release Management

- Faster and consistent deliveries.
- Strong auditing and Traceability of changes.
- Automation of the release process: Higher quality, consistency, confidence.
- Boosts confidence through successful and consistent deliveries.
- Making release – unstressed activity
- No downtime

How the Release Management Process works.

Though there is no formal CCB's (Change Control Board) in DevOps, it does not mean that there are no approvals made on to the changes. Approvals also happen via a tool. The change management tools like Jeera and ClearQuest are used to carry out the recording and approval of changes and routing them into the Dev team for building purpose to a backlog as a technical debt or a new requirement.

These changes picked up by the program team are built, tested and get automatically deployed to the production along with the automated delivery pipeline. But every change is getting logged, in the version control and these changes are audited and tested throughout the delivery pipeline.

So, whatever changes are made by the team, are recorded in the version control tool and what successfully got deployed on to the environments and their configurations are available in the configuration tool.

Basically, it automates the work of a CCB manager, who ideally verifies each of these changes or releases and certifies to let it go to production.

In case of DevOps, it is not the release that gets certified but the entire

delivery pipeline that gets certified in an automated way along with manual gates.

As such release management is not a separate activity as a part of DevOps, but it is built in already as a part of the DevOps pipeline or delivery pipeline along with version control, configuration management, and deployment pipeline.

Tools part of the Release management

Release Management tools which are available on the market ensure that the automatic deployment of changes is timely and error-free and they target to deliver the maximum value to the users.

XL Release is one such release management tool which is specific for Continuous Deployment. This tool help the DevOps teams to design their deployment model and help in monitoring the releases by automating all the tasks related to the deployment and managing the releases.

Plutora is another such robust tool which provides an on-demand Enterprise IT Release Management software toolset that helps in delivering the releases.

BMC Software's Release Lifecycle Management product is also a release management tool from BMC Software that provides end-to-end visibility of the software release's progress.

Benefits of the automated release management system of DevOps

1. The entire release management processes, that is being automated helps the team to have quicker and consistent deliveries made to the customers.

2. Whenever any release or change is pushed through a continuous delivery pipeline in DevOps environment, every information of what actually has happened on the environment, would be clearly written down into the logs. This enables a very strong auditing and traceability of changes maintained in

DevOps.

3. Provides a clear visibility on the details about, what has been delivered, to where it has been delivered, when and how, in case of each release.

4. Automation of the release pipeline provides us the higher quality of delivery within minutes. Human errors, consistency and obviously greater confidence in the deliveries are made.

5. Current DevOps method of deployment and release management has put a curtain to all our earlier woes of stressful moments. No more weekend deployments, no more sleepless nights and no more deployment stress.

MONITORING AND LEARNING

CONTINUOUS MONITORING

- Given the sheer number of releases, there's no way to implement the kind of rigorous pre-release testing that characterizes waterfall development.
- Therefore, in a DevOps environment, failures must be found and fixed in real time.
- How do you do that?
- A big part is continuous monitoring
- Goals of continuous monitoring are
    - to quickly determine when a service is unavailable,
    - understand the underlying causes
        - most importantly, apply these learnings to anticipate problems before they occur.
- Some monitoring experts advocate that the definition of a service must include monitoring—they see it as integral to service delivery.
- Like testing, monitoring starts in development.
- The same tools that monitor the production environment can be employed in development to spot performance problems before they hit production.
- Two kinds of monitoring are required for DevOps:

    - server monitoring
    - application performance monitoring.
- Monitoring discussions quickly get down to tools discussions, because there is no effective monitoring without the proper tools.

Application monitoring or application performance monitoring in simple words is to figure out a way that one can monitor our production site, after the deployment of the software and understand the problems, issues, improvement areas before our customers ever notice these.

Hence, APM is understanding how customers are using our software and engaging ourselves actively with them to understand their requirement to ensure that we are building the right things for the customers.
Earlier even application monitoring also never used to be the task or responsibility of the Dev team but of operations team alone. And Dev team used to support the Operations team if they report any issues.But now it has changed in DevOps.
the software performance will be monitored until there are no issues reported either through the logs or from any of the customers or users reported through the customer service representative.

So, Close monitoring might take one to two days to a week's period till everything settles down and the application usage becomes normal.
It does not mean that the application monitoring stops after that. Once the system and the usage stabilizes, and the users get accustomed to the new software or features, then close monitoring might not be required but, monitoring of the logs continues throughout.
Three ways of monitoring them

- Through the application monitoring tools.
- By constantly going through the logs written out by the applications, which is a manual process.
- By configuring the notifications and alarms in the logs to alert.

Methods 1 and 3 are the ones which are quite often used as an automated method whereas the second one is mainly for internal purposes for the team members.

#1) Monitoring through tools:
There are a lot of tools that are available in the marketplace to carry out the application performance monitoring. These tools provide the

automated way of providing the configured metrics to the team.
All that the team needs to do is to identify the right tool, install and configure them in the live environment so that the tool does the job of monitoring and provides the alarms or alerts whenever there is errors or warnings.

#2) Monitoring through logs:
This is the manual way of monitoring the application, which people used to follow earlier.
Earlier developers used to keep a window on their computer screen from the live environment and constantly run the tail –f command to see the latest and real-time happenings and they used to take care if any issues are found, either by changing the settings or configurations based on the readings from the log.

#3) Monitoring through Notifications and Alarms:
This is quite simple.
Simply configure the notifications and alarms to be triggered in case of any warnings or errors found in the logs that are going to be written out from the Live site and route them to the Mobile numbers of the application monitors, so that it either sends SMS or rings in case of emergency.

Benefit of collecting metrics

#1) Failures, core dumps, error messages, and warnings:

Any failures, error messages and warnings related to servers, networks, databases, websites and even the application functionality gives a clear picture about the quality of the system and the code that is delivered.
So, this metric helps the team to identify the bugs/issues in the application.
The details about the system crash help to understand how did the system crash and then recover, what is the downtime or recovery time to restore the system from the crash.

#2) System and Application Usage Pattern:

This metric aims at finding the pattern in the usage of the computer resources and the application.
This metric helps in identifying if there are any issues in the configurations of the firewalls, load balancing, server memory configurations etc., which later helps the team in fixing and optimizing them.
This also helps the team to get to know how the users are clicking on our applications and fine-tune or enhance those most used features, most

visited screens, websites etc.
So, overall the usage pattern and the trends help in better understanding of the stability of the system and the application.

#3) Performance Metrics:

System and application performance metric is quite important as it provides the performance details of the application, whether the system is too slow or TPS (transactions per second) SLA (service level agreement) is being met or not, whether the system is able to handle the peak load in the live environment or how does the app recovers from the stressed to normal state etc.,
And overall whether the application is performing consistently and reliably. This metric basically adds to the customer satisfaction criteria as speed is key for the customers. They do not want very slow transactions to happen. This metric also attributes to the revenue generation as more the transactions happen, more the revenue generation. The slow performing systems also add more cost due to the excess consumption of infrastructure resources.

#4) Availability Metrics:

Another important metric that the team gathers is the 'Availability metrics'. Keeping the system up and highly available all the times is the expectation from the customers.
It records the no of times that the system has gone down over a period of time, the time taken to recover and hence helps in assessing the business loss due to non-availability. We clearly know that high availability directly contributes to more revenue generation.
The study of the availability pattern provides an opportunity for the team to improve their systems during disaster recovery based on the collected data. Managing high availability and Optimizing infrastructure cost is quite challenging. So, detailed study of this metric pattern helps the team to plan their infrastructure accordingly.

#5) Scalability Metrics:

Well, these days, with the costly affair of the cloud infrastructure adoption, where pay per use model also has been in practice, nobody would like to idle their infrastructure resources.
At the same time, they don't want any of their customers to feel the bite due to any shortfall of these resources. So, the intention is to keep it up always but at the same time not to pay too much for these resources,

unless they are optimally used by adopting the scaling process.
So, it is quite obvious that scaling metrics, which records the scale up, down, scale in – scale out, provides information on whether successful scaling has been done on the system or not.
Hence this metric helps the team in identifying any glitches in the scaling process and aids to plan better.

#6) Custom Telemetry:

Custom telemetry is basically to insert the code to collect a few specific metrics which provides an insight into the application usage and to diagnose the issues in the application during its usage.
It is as good as running the application in the debugging mode. This slows down the application performance and hence this option should be switched on or off as per the requirement.
Actions like tracking events, metrics, exceptions, trace, and timing of the events are captured by making use of the available standard API's. This telemetry helps in understanding the complete behavior of the system which will have a direct bearing on the business and financial benefits.

#7) Other Metrics:

In addition to these, there are a lot more metrics which can be gathered in order to find out how customers are using our software.
Traffic pattern, No of users created, effective load balancing, then
Few other metrics like end user experience, SLA's, etc.
Certain other metrics for product analysis like most used feature, user-friendly screens etc.
So, a collection of these metrics and real-time feedback, which comes directly from the production will help in improving the ease of use and the user experience and hence helps in enhancing the product, and translates into a roadmap to the product enhancement and results in increased customer satisfaction and customer base.

.

APM Tools:

Given below are the few common APM tools available in the market:

1. Newrelic tool is mostly used in performance monitoring of the application like response time, throughput, most time-consuming transactions etc.,
2. ManageEngine is another tool which supports server monitoring, DB monitoring, and cloud monitoring

3. Appdynamics supports managing the performance and availability of the applications across the cloud computing environments and inside the data center.
4. Dynatrace also supports in addition to APM, user experience management solutions, allows monitoring performance globally by emulating real user behavior via PC's across the world.
5. IBM's performance management tool helps in efficiently managing the application, on-premises and hybrid apps and IT infrastructure.

The end