SAINTGITS
LEARN.GROW.EXCEL

Saintgits College of Engineering (Autonomous)

Software Engineering
Module 5

Dr. Rajesh K S
Professor & HOD

What is DevOps?

- DevOps (a combination of development and operations) is a software development method that stresses communication, collaboration and integration between software developers and information technology(IT) professionals thereby
  - Enable rapid evolution of products or services
  - Reduce risk, improve quality across portfolio, and reduce costs

What is DevOps?

- DevOps integration targets product delivery, quality testing, feature development and maintenance releases in order to improve reliability and security and faster development and deployment cycles.
- The adoption of DevOps is being driven by factors such as:
- Use of agile and other development processes and methodologies
- Demand for an increased rate of production releases from application and business stakeholders
- Wide availability of virtualized and cloud infrastructure from internal and external providers
- Increased usage of data center automation and configuration management tools

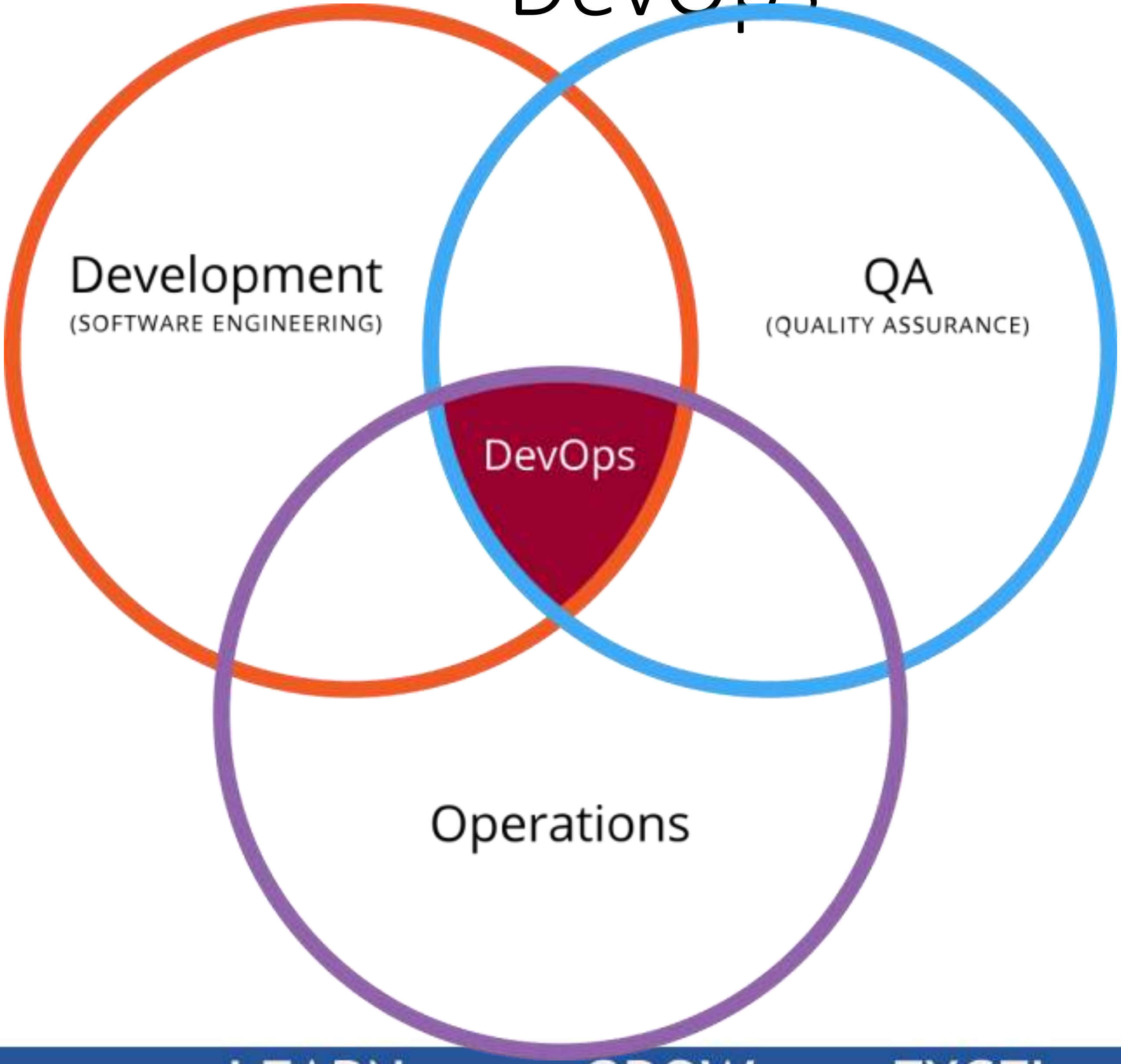
DevOps

- **DevOps** (development and operations) is an enterprise software development phrase used to mean a type of agile relationship between Development and IT Operations.
- The goal of **DevOps** is to change and improve the relationship by advocating better communication and collaboration between the two business units.
- **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.**

IT operations

- **Information technology operations**, or **IT operations**, could be defined as: "IT Operations is responsible for the smooth functioning of the infrastructure and operational environments that support application deployment to internal and external customers, including the network infrastructure; server and device management; computer operations; IT infrastructure library (ITIL) management; and help desk services for an organization."

DevOps


Development
(SOFTWARE ENGINEERING)
QA
(QUALITY ASSURANCE)
DevOps
Operations

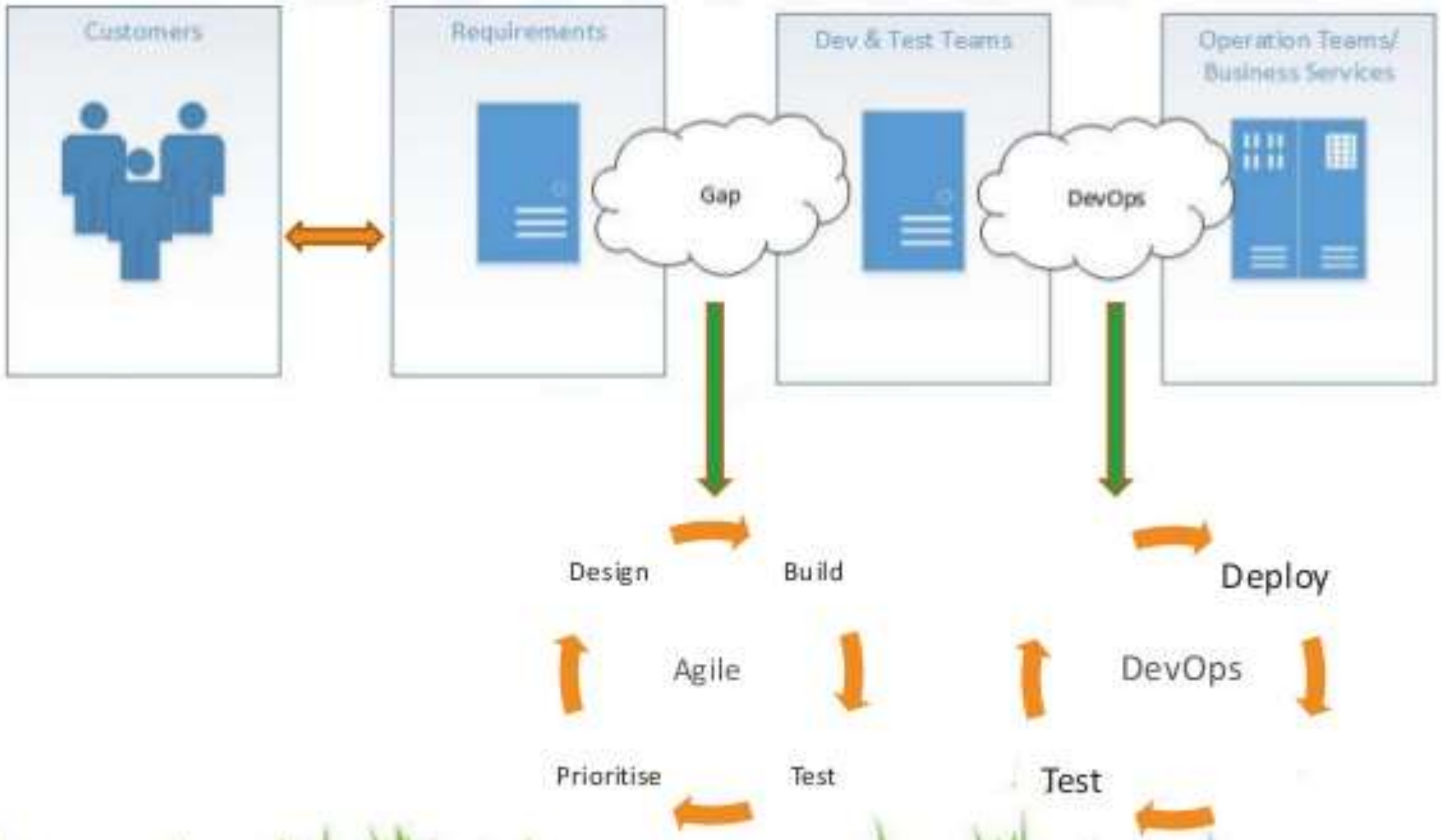
Need of DevOps

- In an enterprise business units operate as individual entities within the enterprise.
- On the software development side -- and for those working in IT operations -- there needs to be better communication and collaboration to best serve the IT business needs of the organization.
- DevOps-based culture that partners developers with operations staff to ensure the organization achieves optimal running of software with minimal problems. This culture is one that supports a willingness to work together and share.
- The DevOps culture puts a focus on creating a fast and stable work flow through development and IT operations.
- One main goal of DevOps is to deploy features into production quickly and to detect and correct problems when they occur, without disrupting other services.

Principles of DevOps

- Develop and test in an environment similar to production
- Deploy builds frequently
- Validate operation quality continuously

Agile and DevOps

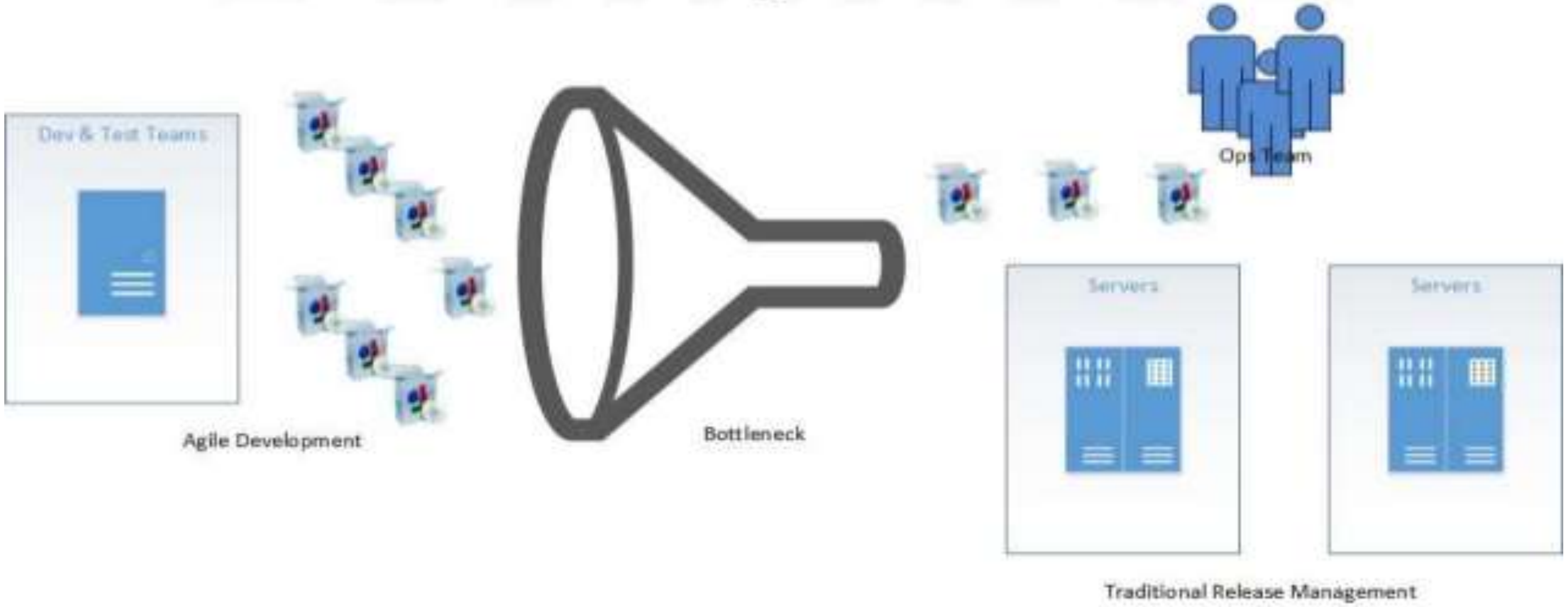


Agile and DevOps

- Agile Development
  - Addresses the gap between customer requirements and dev + testing teams
  - Cross-functional teams to design, develop, and test features/stories prioritised by the PO (Customer)
  - Focuses more on functional and non-functional readiness
- DevOps
  - Addresses the gap between dev + testing and Ops
  - Automated release management
  - Focuses on functional and non-functional plus operational and business readiness
  - Intensifies reusability and automation

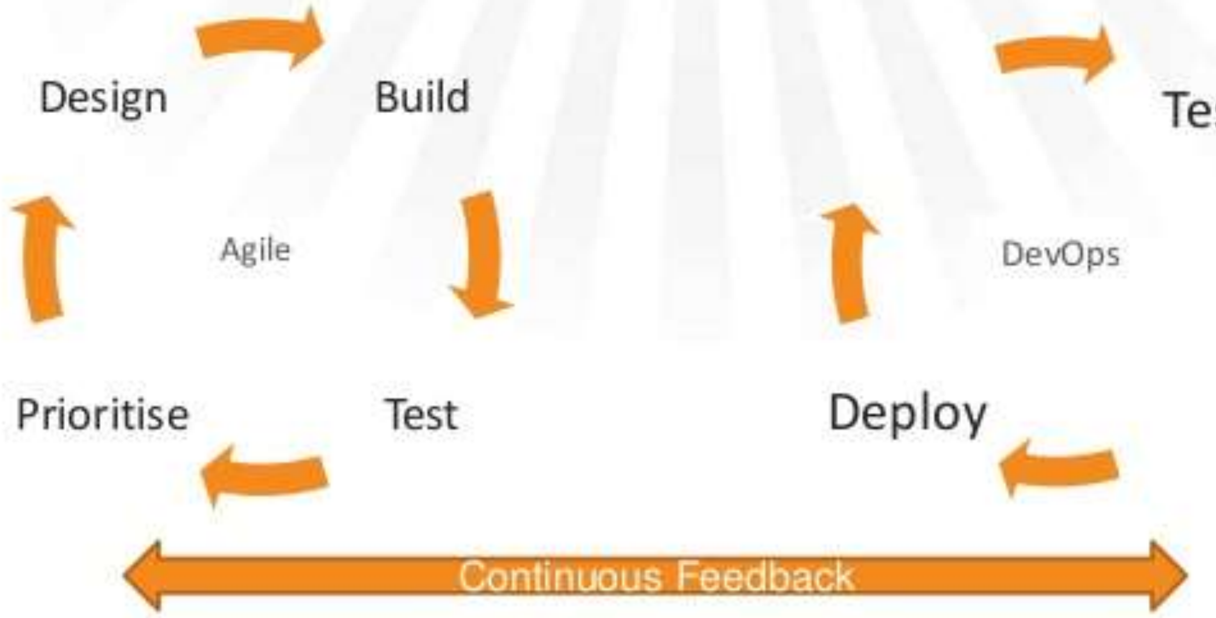
SAINTGITS
LEARN.GROW.EXCEL

Agile + Traditional Release Management



Agile + DevOps

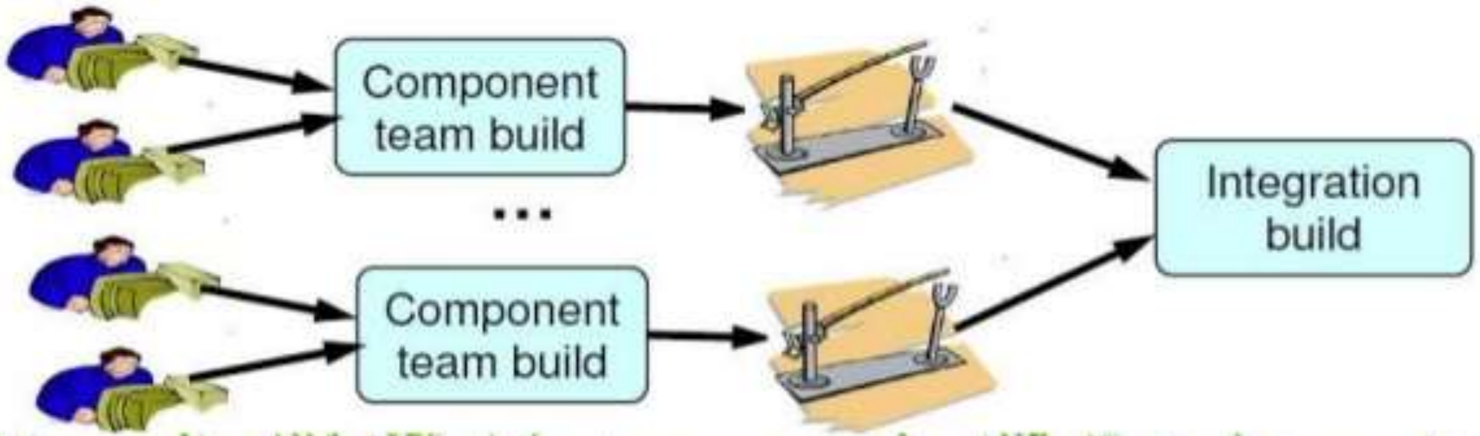
Continuous Integration extended as Continuous Delivery



Faster Delivery reduces risk

Continuous Integration

- Integrate the code changes by each developer so that the main branch remains up-to-date



When to adopt and when not to

When to Adopt:

- For eCommerce and other web site projects (Amazon, Flickr, Groupon,etc)
- Cloud platform (IaaS and PaaS)

When not to Adopt:

- Mission critical applications (Banks, Power Systems, etc)

The DevOps Lifecycle

- Check in code
- Pull code changes for build
- Run tests (continuous integration server to generate builds and arrange releases): Test individual models, run integration tests, and run user acceptance tests.
- Store artifacts and build repository (repository for storing artifacts, results, and releases)
- Deploy and release (release automation product to deploy apps)
- Configure environment
- Update databases
- Update apps
- Push to users – who receive tested app updates frequently and without interruption
- Application and Network Performance Monitoring (preventive safeguard)
- Rinse and repeat

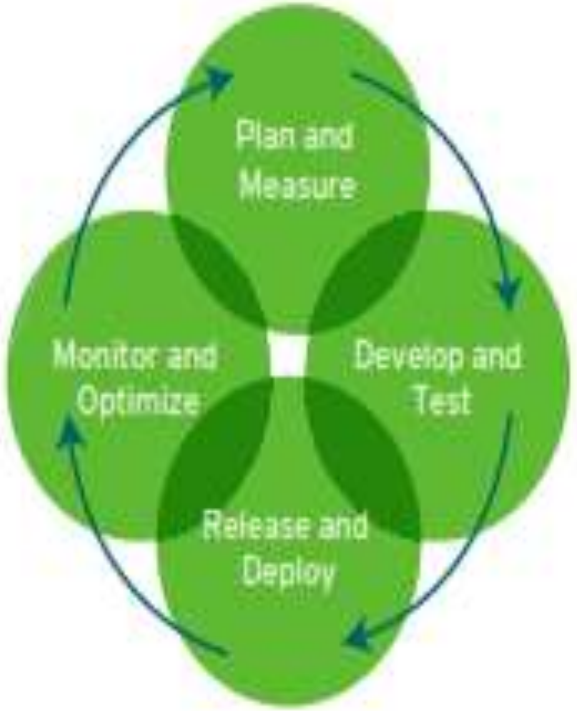
Kev Activities that define DevOp

Collaborative Development:
Increased collaboration between teams

Continuous Integration & Continuous Testing:
Integration of software testing with deployment and operations

Continuous Release and Deployment:
Increased delivery speed and frequency

Continuous Monitoring:
Improved quality by monitoring production performance



SAINTGITS
LEARN.GROW.EXCEL

Continuous Integration

- Doing integrations and builds to verify them on a regular, preferably daily basis, is *Continuous Integration*.
- It forces developers and team of developers to integrate their individual work with each others as early as possible.
- This exposes integration issues and conflicts on a regular basis.
- To make continuous integration work, developers have to communicate more and ensure that their work includes changes from other developers that may impact the code they are working on.

Continuous Integration

- Developers are not only able to find and address integration issues regularly but also verify that the integrated application or component functions as designed.
- If one does not practice continuous integration, the process of integrating ones work with that of other developers on the team would probably only happen at the end of the sprint or worse – a multi-month iteration. This is leaving unaddressed and undiscovered risk in the application till too late. This would require developers to dedicate potentially significant time at the end of every sprint just to integrate their work and fix any issues.
- Agile methodologies require that this be avoided by integrating regularly and exposing and addressing any issues regularly, hence reducing risk.

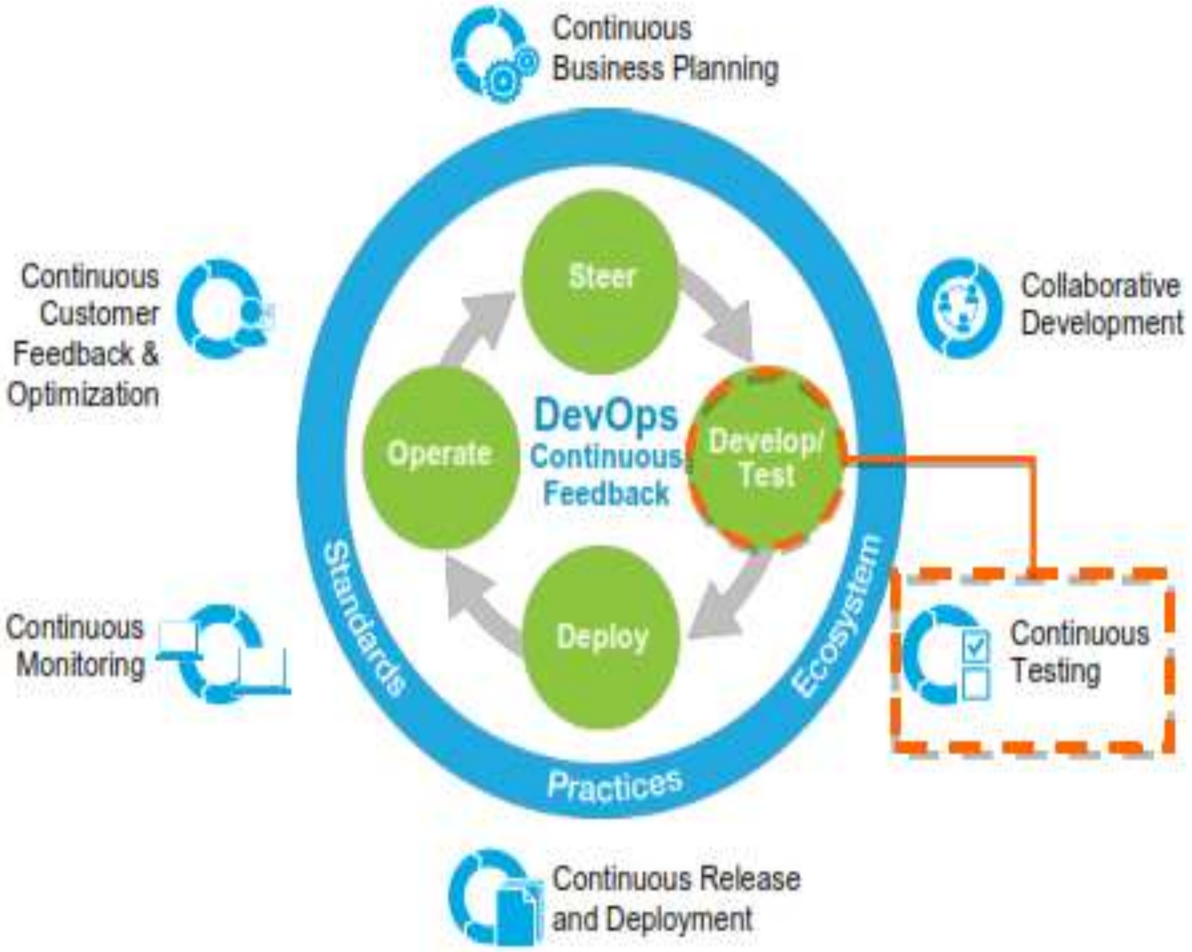
Continuous Testing (CT)

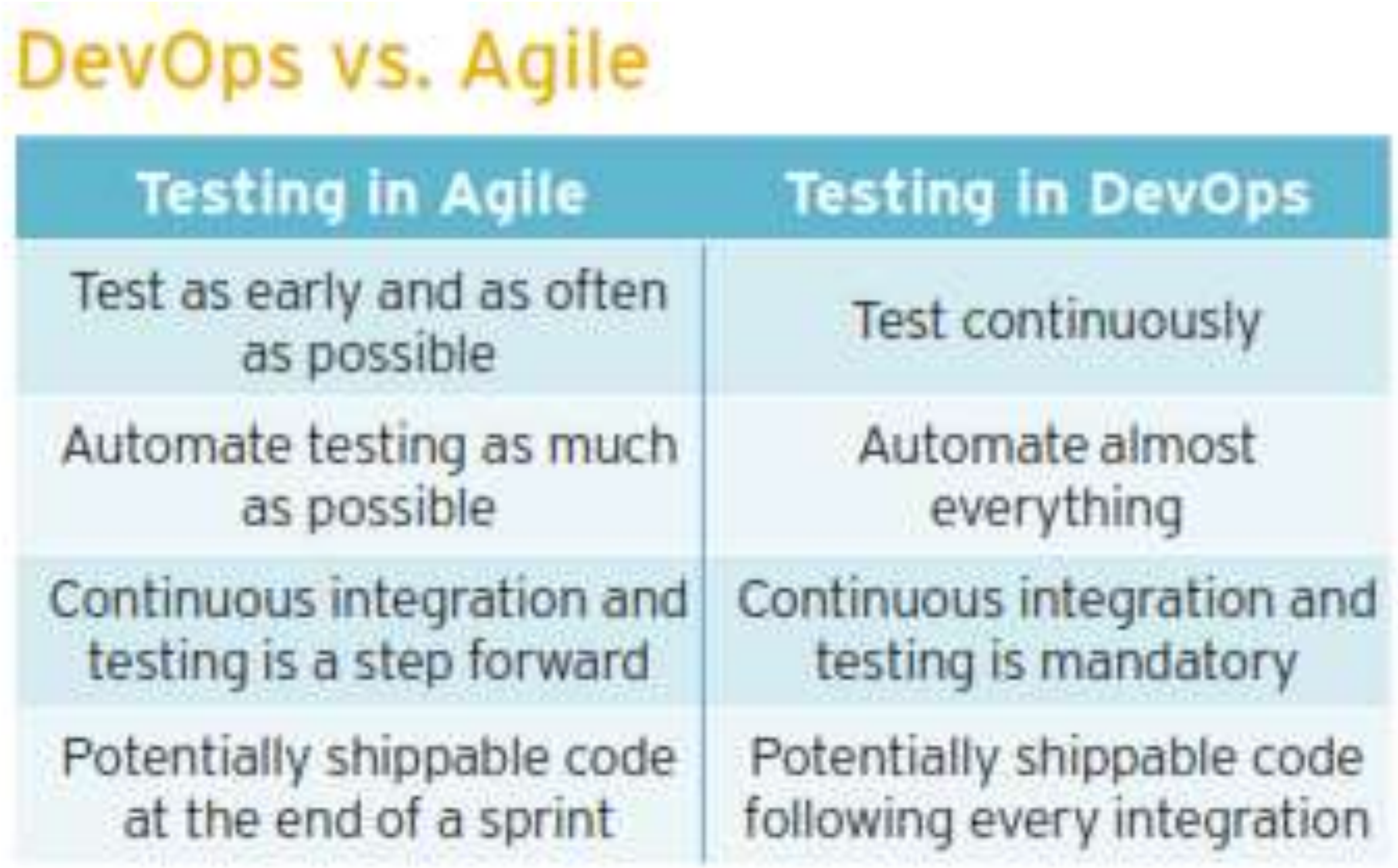
- Removes traditional testing bottlenecks such as unavailable test environments to increase efficiency.
- Enables teams to test earlier and with greater coverage at lower cost.
- Provides project teams with continuous feedback on software quality to reduce business risk.
- Avoid the big bang.

Accelerate software delivery – *for faster time to value*

Balance speed, cost, quality and risk – *for increased capacity to innovate*

Reduce time to customer feedback – *for improved customer experience*


Continuous Business Planning
Continuous Customer Feedback & Optimization
Collaborative Development
Steer
Operate
DevOps
Continuous Feedback
Develop/ Test
Deploy
Continuous Monitoring
Continuous Testing
Standards
Ecosystem
Practices
Continuous Release and Deployment



DevOps vs. Agile

Testing in Agile	Testing in DevOps
Test as early and as often as possible	Test continuously
Automate testing as much as possible	Automate almost everything
Continuous integration and testing is a step forward	Continuous integration and testing is mandatory
Potentially shippable code at the end of a sprint	Potentially shippable code following every integration

Types of tests conducted

- **Health check:** An automated check verifies services are up after deployment. Typically, health checks run for just a few minutes.
- **Smoke tests:** This most critical set of automated tests ensures that **key system features are operational and no blocking defects occur**. Smoke tests are typically executed in less than 15 minutes, and as the CT process matures, the response time should be further optimized.
- **Intelligent regression:** If execution time for overall regression is significantly high, CT setup becomes less effective due to longer feedback cycles. In such a scenario, a subset of regression carved out at run-time based on criticality and impacted areas can only be executed to provide feedback within a reasonable timeframe. Full regression execution can be shifted to overnight or during the weekend depending on its alignment with recurring build frequencies.

SAINTGITS
LEARN.GROW.EXCEL

Types of tests conducted contd…..

- **Full regression**: This is the final feedback from the CT ecosystem. The goal is to minimize feedback time by running a parallel execution of automated tests through multiple threads or machines.

  All of the above tests run as part of the build deployment process.

  If any fail, the deployment process is halted, and everyone involved in delivering software – developers, testers and operations staff – is notified immediately, triggering corrective action.

  Shorter feedback loops enable the team to fail fast and recover quickly.

SAINTGITS
LEARN.GROW.EXCEL

Advantages

1. Improved Availability
2. Faster Deployment
3. Accelerated error correction

Continuous Delivery

- Once the application is built, at the end of every Continuous Integration build, *deliver* it to the next stages in the application delivery lifecycle.
- Deliver it to the QA team for testing and then to the operations team (the *Ops* in DevOps) for delivery to the production system.
- The goal of Continuous Delivery is to get the new features that the developers are creating, out to the customers and users **as soon as possible.**
- Now, all builds that come out of a Continuous Integration effort do not need to go QA, only the 'good' ones with functionality that is at a stage of development that it can be tested needs to go to QA.
- Similarly, all the builds that go thru QA do not need to go to production.
- Only those that are ready to be delivered to the users, in terms of functionality, stability should be delivered to production.
- This practice of regularly delivering the application being developed to QA and Operations for validation and potential release to customers is what is referred to as *Continuous Delivery.*

Continuous Delivery

- Is a set of practices and principles aimed at building, testing, and releasing software faster, and more frequently.

- Highest priority is to satisfy the customer through early and continuous delivery.

Configuration Management (CM

- *Configuration management is the process of standardizing resource configurations and enforcing their state across IT infrastructure in an automated yet agile manner.*
- provides visibility and control of its performance, functional, and physical attributes.
- CM verifies that a system performs as intended, and is identified and documented in sufficient detail to support its projected life cycle.
- The CM process facilitates orderly management of system information and system changes for such beneficial purposes as to revise capability; improve performance, reliability, or maintainability; extend life; reduce cost; reduce risk and liability; or correct defects.
- CM emphasizes the functional relation between parts, subsystems, and systems for effectively controlling system change.

Configuration Management (CM

- It helps to verify that proposed changes are systematically considered to minimize adverse effects.
- Changes to the system are proposed, evaluated, and implemented using a standardized, systematic approach that ensures consistency, and proposed changes are evaluated in terms of their anticipated impact on the entire system.
- CM verifies that changes are carried out as prescribed and that documentation of items and systems reflects their true configuration.
- A complete CM program includes provisions for the storing, tracking, and updating of all system information on a component, subsystem, and system basis.
- A structured CM program ensures that documentation (e.g., requirements, design, test, and acceptance documentation) for items is accurate and consistent with the actual physical design of the item.

SAINTGITS
LEARN.GROW.EXCEL

Configuration Management

- Configuration management solves the problem of having to manually install and configure packages once the hardware is in place.
- The benefit of using configuration automation solutions is that servers are deployed exactly the same way every time.
- If you need to make a change across ten thousand servers you only need to make the change in one place.

Configuration Management (CM Tools

– Chef in Unix, Linux and Windows

– Puppet in Unix and Windows

– Ansible - server and configuration management tool

– Salt Stack - Python-based open-source configuration management software

– Pallet is an open source **DevOps** library for the JVM

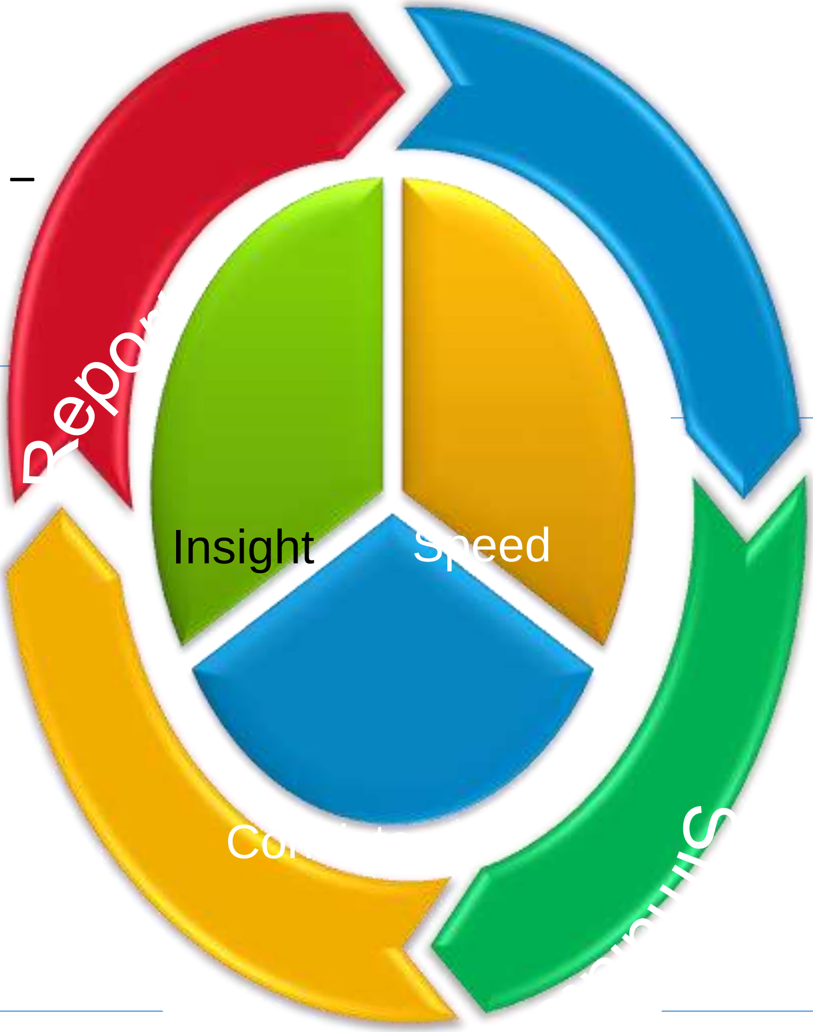
– Bcfg2 (pronounced "bee-config") aids in the infrastructure management lifecycle – configuration analysis, service deployment, and configuration auditing.

- There are other vendor-specific DevOps tools as well:
  – Amazon's OpsWorks and CloudFormation
  – Ubuntu's JuJu

Configuration Management

Gain insight(understanding) into your infrastructure – identify configuration drift.

Define desired state using a powerful, declarative language



Enforce desired state by remediating configuration drift

True simulation of changes before deployment. Then move through nonprod to prod.

Release Management (RM)

- Release management is the part of the software management process dealing with development, testing, deployment and support of **software releases** to the end user.
- The team involved in this process is referred to as the **release management team.**
- The release manager interacts with development, testing, and operations to ensure the software under development meets the business goals.
- ITIL (**Information Technology Infrastructure Library)**, is a set of practices for IT service management (ITSM) that focuses on aligning IT services with the needs of business.

- Release management is the process responsible for planning, scheduling, and controlling the movement of releases to test and live environments.

- ITIL specifies that the primary objective of Release Management is to make sure that the integrity of the live environment is protected and that the correct components are released.

- Release management works closely with Change Management to make sure that changes to the IT infrastructure are implemented to keep the functionality and service levels of the services aligned with the ever-changing business needs of their customers.

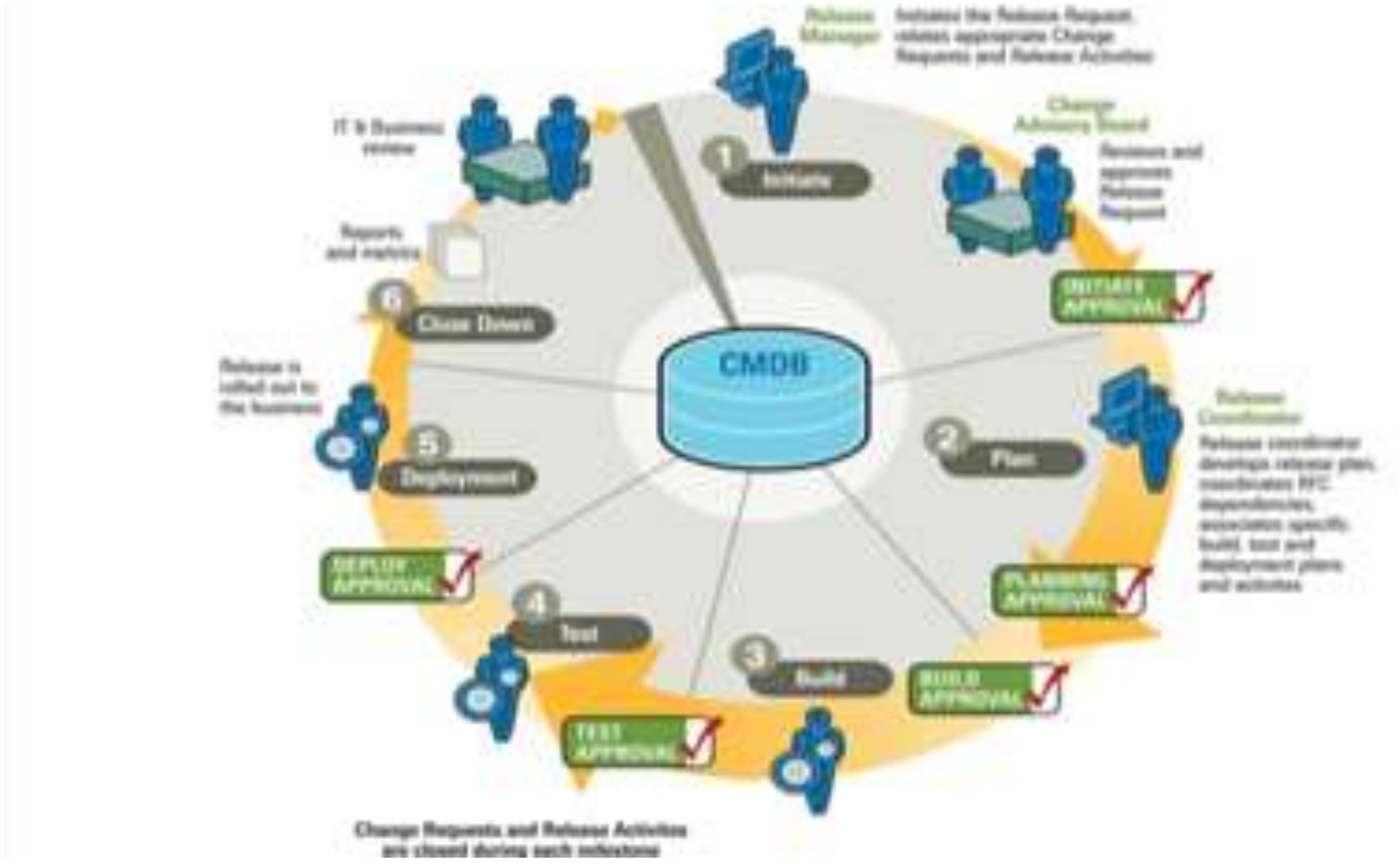
SAINTGITS
LEA

Need of release management

- With numerous changes occurring daily, Release Management is the key component in making sure that applications are successfully deployed without compromising the integrity or availability of the production environment.

- Using a systematic and repeatable release process, organizations can achieve greater success rates of change rollout, higher quality of IT service, and accelerated time-to-market.

Release life cycle


Release Manager
Initiates the Release Request, relates appropriate Change Requests and Release Activities
Change Advisory Board
Reviews and approves Release Request
1 Initiate
INITIATE APPROVAL
Release Coordinator
Release coordinator develops release plan, coordinates RFC dependencies, associates specific build, test and deployment plans and activities
2 Plan
PLANNING APPROVAL
BUILD APPROVAL
3 Build
TEST APPROVAL
4 Test
DEPLOY APPROVAL
5 Deployment
Release is rolled out to the business
6 Close Down
Reports and metrics
IT & Business review
CMDB
Change Requests and Release Activities are closed during each milestone

CMDB

- A configuration management database (**CMDB**) is a repository that acts as a data warehouse for information technology (IT) installations.
- It holds data relating to a collection of IT assets (commonly referred to as configuration items (CI)), as well as to descriptive relationships between such assets.

Release Lifecycle

- **initiate**
  - Release Manager initiates the Release request, relates appropriate Change requests and Release acivities.
  - Change Advisory board reviews and approves release request.
  - Initiates approval.
- **Plan**
  - Release coordinator develops release plan, coordinates RFC (Request for Change) dependencies, associates specific build, test and deployment plans and activities.
  - Planning approval.
  - Build approval
- **Build.**
  - Build
  - Test approval.
- **Test**
  - **Test**
  - Deploy approval.
- **Deployment.**
  - Release is rolled out to the business.
- **Close down.**
  - Reports and metrics.
  - IT and business review.

Release Plan: Defined

- Release: A version of a product placed into production.

- Release Cycle: Frequency at which a version of a product is placed into production.

- Release Plan: An overview of the release, identifying the efforts to make it successful.

Release management strategies

- **Release windows (slow cadence)**. A release window is a period of time during which one or more teams may release into production. A release slot is subset within that release window (and may be the entire window) during which a team may deploy their solution into production.
- **Release train**. The idea with a release train is that every team involved with that "train" has the same release cadence – for example this train releases once a quarter, or once a month, or even once a week. This strategy is commonly used in large programs, or teams of teams, where the individual teams are each working on part of a larger whole. The primary disadvantage is that development teams are constrained to a common release schedule
- **Release windows (quick cadence)**. To support continuous deployment, particularly across many delivery teams, you will need a large number of release slots. The advantage of quick cadence release windows is that they are less likely to suffer from the bottleneck challenges associated with slow cadence release windows and release trains.
- **Continuous release availability.** With this approach delivery teams are allowed to release their solutions into production whenever they need to. This is the only strategy that truly supports continuous delivery.

SAINTGITS
LEARN.GROW.EXCEL

Benefits of DevOps approach

- **Learning about things earlier.**

  Quick feedback shrinks the time between the introduction of a defect and when it is discovered and fixed. When employing continuous delivery, both the development and operations teams are notified when any problem is introduced into any part of the software system, whether the problem occurs in the application code, infrastructure, data, or configuration. Using this approach team members learn of problems much earlier than they would during traditional projects

- **Less complex problems**

  Because the frequency of change is almost constant with teams who embrace DevOps, the change sets are smaller. The software is regularly integrated with tests, so when problems do occur, they often less complex than they would be during traditional projects, because the defect may have only been introduced a few minutes prior to its discovery.

Benefits of DevOps approach contd…

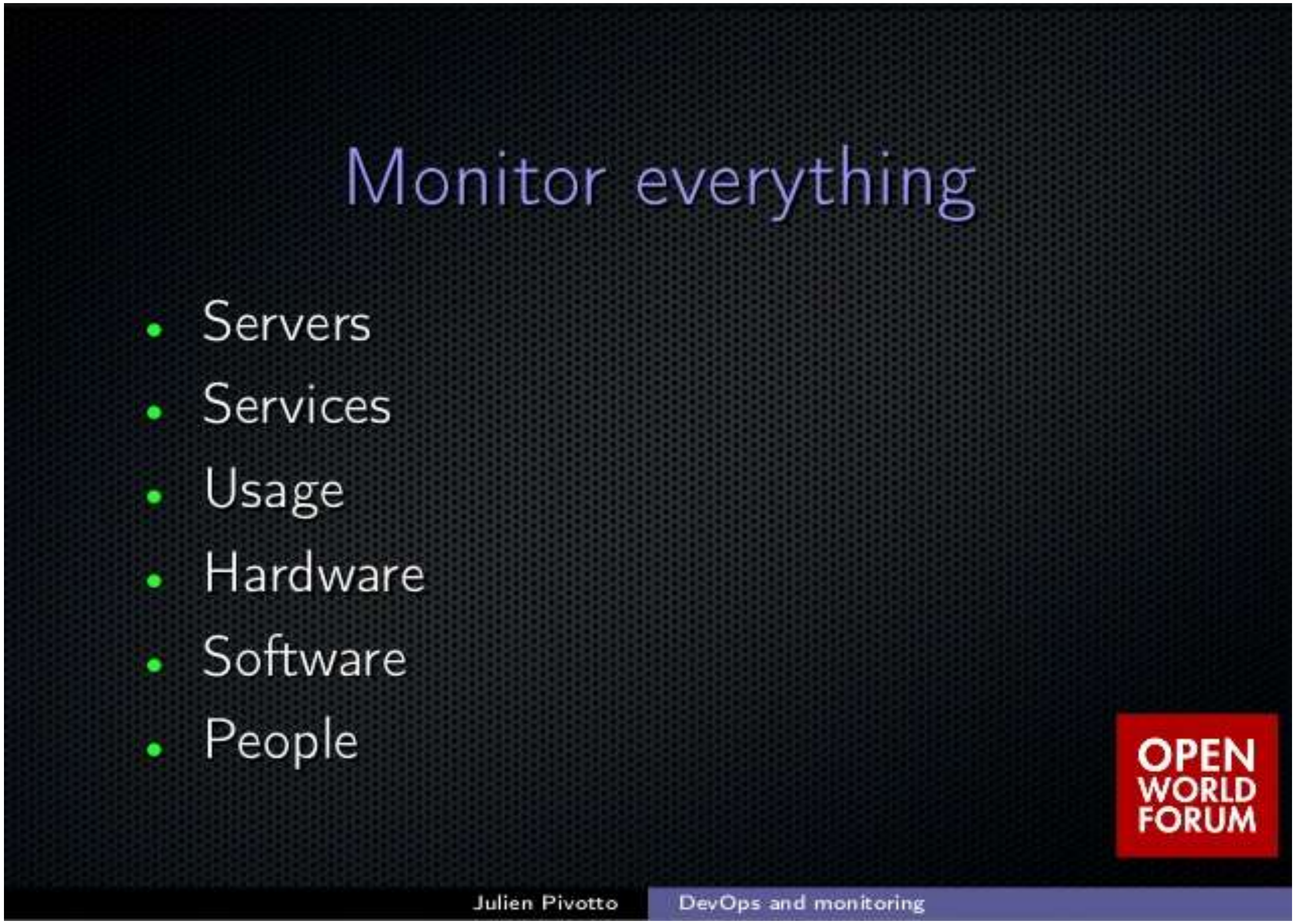
- **Less time fixing problems**
- Because teams that embrace DevOps are cross-functional, team members also spend less time fixing a problem. They do not need to wait for a separate team in a different part of the organization to troubleshoot and fix it. Everyone is on the same team. With changes occurring more frequently and in smaller batches, problems are less complex and team members spend less time assessing and fixing defects.

Monitoring in DevOps

- DevOps monitoring is the process of observing and evaluating the functionality and state of systems and applications to spot and address problems as they arise.
- Involves gathering information on a variety of topics, such as CPU usage, disk space, and application response times.
- The process of routinely and vigilantly looking for indications of performance degradation in systems, networks, and data is known as continuous monitoring.
- Continuous monitoring, which can be carried out manually or automatically, often entails utilizing software to check for vulnerabilities and follow changes in security settings.

Monitoring

- **Two types of monitoring tools** are application and system.
- **Application monitoring** takes a top-down approach, and system monitoring takes a bottom-up approach. **With application monitoring tools,** teams can monitor the performance of the application from the user's perspective, including page load times, database transaction times and so on.
- **System monitoring** focuses on factors such as CPU load, utilized memory, and disk space.
- As a result of the constant feedback elicited through a continuous delivery approach, monitoring becomes more vital to the business as it is able to tweak features, test scenarios, and deliver different features to different users based on application monitors. An application-monitoring tool can sit at the epicenter of an effective cross-functional team that adopts DevOps and Continuous delivery.



Monitor everything

- Servers
- Services
- Usage
- Hardware
- Software
- People

OPEN WORLD FORUM

What needs to be Monitored?

Data Category	Description
Performance	Page loads, query times, response times, upload/download speeds, etc.
Capacity	Disk space, memory, CPU, bandwidth, etc.
Uptime	Availability (e.g.. Four 9's)
Throughput	Every layer (web, cache, database, network, app stack, etc.)
SLAs	Availability, reliability, security, etc.
KPIs	Examples: Revenue per minute, Avg concurrent users, etc.
User Metrics	Registrations, page views, bounce rates, click rates, etc.
Governance/Compliance	Access, permissions, intrusion detection, intrusion prevention, cost containment, etc.
Log file analysis	Predictive analytics, pattern recognition, etc.

- SLA - Service Level Agreement
- KPI - A **key performance indicator** (**KPI**) is a business metric used to evaluate factors that are crucial to the success of an organization. **KPIs** differ per organization; business **KPIs** may be net revenue or a customer loyalty metric, while government might consider unemployment rates.

SAINTGITS
LEARN.GROW.EXCEL

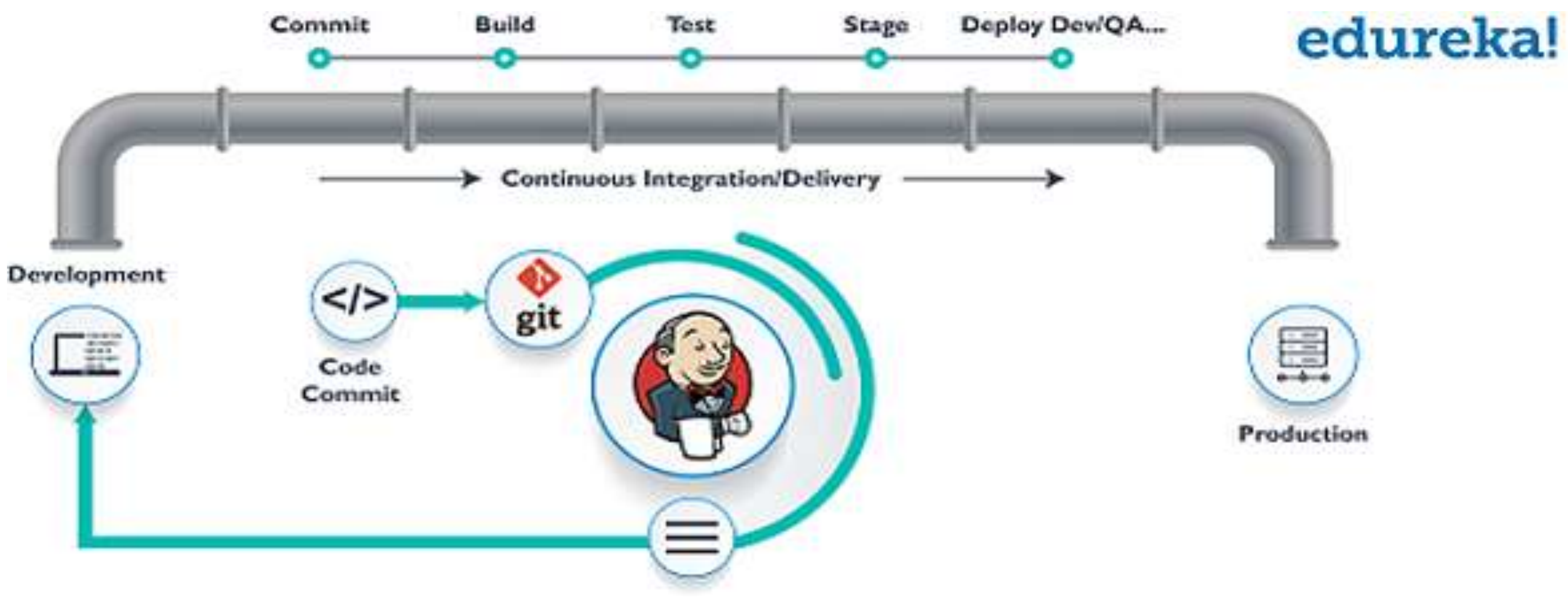
Additional notes (supplement)

Life Cycle Phases

- The various phases such as continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring constitute DevOps Life cycle.
- **Continuous Development**
- This is the phase that involves 'planning' and 'coding' of the software. The vision of the project is decided during the planning phase and the developers begin developing the code for the application. There are no DevOps tools that are required for planning, but there are a number of tools for maintaining the code.
- The code can be written in any language, but it is maintained by using Version Control tools. Maintaining the code is referred to as Source Code Management. The most popular tools used are Git, SVN, Mercurial, CVS, and JIRA. Also tools like Ant, Maven, Gradle can be used in this phase for building/ packaging the code into an executable file that can be forwarded to any of the next phases.

- **Continuous Testing**
- This is the stage where the developed software is continuously tested for bugs. For Continuous testing, automation testing tools like **Selenium**, **TestNG**, **JUnit**, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality. In this phase, Docker Containers can be used for simulating the test environment.
- Selenium does the automation testing, and the reports are generated by TestNG. This entire testing phase can be automated with the help of a Continuous Integration tool called Jenkins. Suppose you have written a selenium code in Java to test your application. Now you can build this code using ant or maven. Once the code is built, it is tested for User Acceptance Testing (UAT). This entire process can be automated using Jenkins.
- Automation testing saves a lot of time, effort and labor for executing the tests instead of doing this manually. Besides that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. We can also schedule the execution of the test cases at predefined times. After testing, the code is continuously integrated with the existing code.

Continuous Integration



- This stage is the heart of the entire DevOps life cycle. It is a software development practice in which the developers require to commit changes to the source code more frequently. This may be on a daily or a weekly basis. Every commit is then built and this allows early detection of problems if they are present. Building code not only involves compilation but it also includes code review, unit testing, integration testing, and packaging.
- The code supporting new functionality is continuously integrated with the existing code. Since there is continuous development of software, the updated code needs to be integrated continuously as well as smoothly with the systems to reflect changes to the end-users.
- Jenkins is a very popular tool used in this phase. Whenever there is a change in the Git repository, Jenkins fetches the updated code and it prepares a build of that code which is an executable file in the form of a war or a jar. This build is then forwarded to the test server or the production server.

Continuous Deployment

- This is the stage where the code is deployed to the production servers. It is also important to ensure that the code is correctly deployed on all the servers.
- Configuration management and Containerization tools. These set of tools here help in achieving Continuous Deployment (CD).
- Configuration Management is the act of establishing and maintaining consistency in an application’s functional requirements and performance. It is the act of releasing deployments to servers, scheduling updates on all servers and most importantly keeping the configurations consistent across all the servers.
- Since the new code is deployed on a continuous basis, configuration management tools play an important role in executing tasks quickly and frequently. Some popular tools that are used here are Puppet, Chef, SaltStack, and Ansible.
- Containerization tools also play an equally important role in the deployment stage. Docker and Vagrant are the popular tools used for this purpose. These tools help produce consistency across Development, Test, Staging and Production environments. Besides this, they also help in scaling-up and scaling-down of instances swiftly.

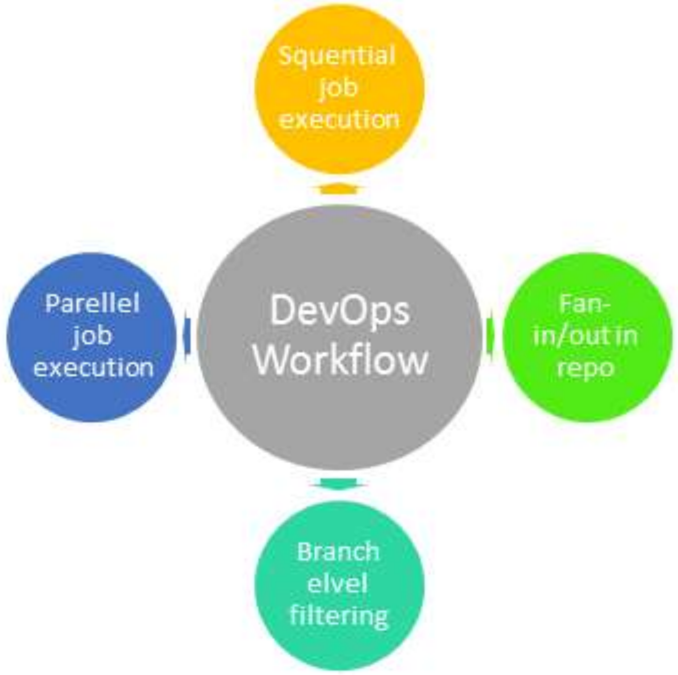
- Containerization tools help in maintaining consistency across the environments where the application is developed, tested and deployed. Using these tools, there is no scope of errors/ failure in the production environment as they package and replicate the same dependencies and packages used in the development/ testing/ staging environment. It makes your application easy to run on different computers.

- **Continuous Monitoring**
- This is a very crucial stage of the DevOps life cycle where you continuously monitor the performance of your application. Here vital information about the use of the software is recorded. This information is processed to recognize the proper functionality of the application. The system errors such as low memory, server not reachable, etc are resolved in this phase.
- The root cause of any issue is determined in this phase. It maintains the security and availability of the services. Also if there are network issues, they are resolved in this phase. It helps us automatically fix the problem as soon as they are detected.

- This practice involves the participation of the Operations team who will monitor the user activity for bugs or any improper behavior of the system. The popular tools used for this are Splunk, ELK Stack, Nagios, NewRelic and Sensu. These tools help you monitor the application’s performance and the servers closely and also enable you to check the health of the system proactively.
- They can also improve productivity and increase the reliability of the systems, which in turn reduces IT support costs. Any major issues if found are reported to the development team so that it can be fixed in the continuous development phase. This leads to a faster resolution of the problems.
- These DevOps stages are carried out on loop continuously till you achieve the desired product quality. Therefore almost all of the major IT companies have shifted to DevOps for building their products.

DevOps Work Flow

- Workflows provide a visual overview of the sequence in which input is provided. It also tells about actions are performed, and output is generated for an operations process.
- Workflow allows the ability to separate and arrange jobs which are top-requested by the users. It also gives the ability to mirror their ideal process in the configuration jobs.


Squential job execution
Parellel job execution
DevOps Workflow
Fan-in/out in repo
Branch elvel filtering

DevOps Principles

- Here, are six principles which are essential when adopting DevOps:
- **1. Customer-Centric Action:** DevOps team must take customer-centric action for that they should constantly invest in products and services.
- **2. End-To-End Responsibility:** The DevOps team need to provide performance support until they become end-of-life. This enhances the level of responsibility and the quality of the products engineered.
- **3. Continuous Improvement:** DevOps culture focuses on continuous improvement to minimize waste. It continuously speeds up the improvement of product or services offered.
- **4. Automate everything:** Automation is a vital principle of DevOps process. This is not only for the software development but also for the entire infrastructure landscape.
- **5. Work as one team:** In the DevOps culture role of the designer, developer, and tester are already defined. All they needed to do is work as one team with complete collaboration.
- **6. Monitor and test everything:** It is very important for DevOps team to have a robust monitoring and testing procedures.

SAINTGITS
LEARN.GROW.EXCEL

Need of DevOps

- In an enterprise business units operate as individual entities within the enterprise.
- On the software development side -- and for those working in IT operations -- there needs to be better communication and collaboration to best serve the IT business needs of the organization.
- DevOps-based culture that partners developers with operations staff to ensure the organization achieves optimal running of software with minimal problems. This culture is one that supports a willingness to work together and share.
- The DevOps culture puts a focus on creating a fast and stable work flow through development and IT operations.
- One main goal of DevOps is to deploy features into production quickly and to detect and correct problems when they occur, without disrupting other services.

Configuration Management

- **The key highlight of configuration management in DevOps is delivering,**
- Infrastructure as a code
- Configuration as a code
- **There are numerous benefits of ‘Infrastructure as a code’ and ‘Configuration as a code’ in DevOps practice.**
  - Configurations are Version controlled
  - Automated and standardized
  - Removes dependency
  - Error-free infra setups
  - Boosts collaboration between Operations and Development team
  - Correcting configuration drift
  - Treating infrastructure as a flexible resource
  - Automated scaling of infrastructure
  - Maintaining consistency in the setups

- Configuration management as the name itself explains, is nothing but 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.
- The environment set up's for these tests would progressively become more complex as it moves towards pre-production and production environment.
- So, basically, configuration management is the automated process to manage all the configurations of each of these environments.

- In our 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.
- In this context, we can call that the Operations team is integrated with Development in managing the environments through the single version control tool.

- **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.
- Just to give a simple <u>**Example**</u>, 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.

SAINTGITS
LEARN.GROW.EXCEL

- The developer or tester need not be a system admin expert to set up their servers for development or testing activity.
- So, the infrastructure set up in DevOps will be completely automated, and basically follows the script that is checked in to version control starting from installing the servers, configuring them, installing the OS, till the communication channels of these instances are established with the deployed software.

- **Configuration as a code**: It 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.
- 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.

SAINTGITS
LEARN.GROW.EXCEL

- All the configurations and infrastructure details are version controlled which is a big benefit in DevOps implementation.
- **#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.
- Since configuration management is totally automated, manual intervention either to set up or update is completely eliminated.

- So, automation of configuration management not only has benefited in time-saving but also in eliminating such human errors and improving the quality. Also, coding standard has helped the team in following the specified standard in coding and automating instead of following the fantasy of each of the person who writes the config guide.
- 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.
- Or even for setting up the infra and configurations, which are completely automated and version controlled. So, anybody in the team be it a developer or tester can spin a server and carry out the configurations for their development and testing purposes. Hence setting up the server and configurations has become person independent.
- This also ensures that the same servers are not used by both the Development and QA teams for their activities, which generally used to be the case earlier.

- Infrastructure and configurations defined as a common code along with automating and version controlling standardize all the environments and set up’s. So, this not only makes the debugging task easy for the developers but also eliminates the human errors resulting in error-free infra setups, otherwise which would cause huge damage, if not detected early.
- This working together to achieve a common goal boosts the collaboration between both the teams, Development, and operations.
- **What is Configuration Drift?**
- Small differences and inconsistencies between the servers, which sometimes happens due to manual update, which accumulates over a period of time are called Configuration drift.
- This needs to be avoided to make the team to use the automation of configurations effectively.

Release Management

- DevOps is the entire team owning the software from its inception until it is delivered to the production and ensuring that the application is performing in the production as per the requirements.
- Hence, 'Release Management' as we all know is to manage which version of the software is deployed to which environment, when and how is not just the responsibility of the Release Manager, but the responsibility of the entire team in DevOps.
- **The prime benefits of Release Management in DevOps can be summarized as,**
  - 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

- Which release is running in which environment, and what patches have been applied there? Which are the hotfixes that have been deployed and for which customer it is?
- It is the headache of the release manager to keep a track of all these information. We know, earlier, release management neither used to be the responsibility of the Dev or the Ops. It was a separate release management team who used to manage the software release activities.
- And a separate board called CCB and CAB, change control board, change approval board, used to handle the responsibility of managing the changes and controlling on what is applied and what is not.
- But now things have changed with the DevOps. And it is no more just the responsibility of the release manager but the entire team.
- As we defined DevOps earlier, DevOps is an entire team owning the software from the inception until it is delivered to the production and ensuring that the application is performing in the production as per the requirements.
- Thus, in DevOps, unless the code is deployed on to the site and is monitored for its performance successfully for over a specific period, the software development task is not complete.
- Hence, the responsibility of the software delivery and its performance in live lies with everyone in the team. So does the release management tasks.

- From a broad perspective, release management is managing and maintaining the information’s like, which version of the software or the components are deployed to which environments, when and how it was deployed.
- So, this is all about release management.

SAINTGITS
LEARN.GROW.EXCEL

APM

- What is this application performance monitoring all about?
- How do we carry out the application performance monitoring, shortly called APM, in the live environment?
- What metrics do we gather to ensure that the software is performing successfully?
- What are the benefits of Application monitoring?
- What are the tools that we engage upon in carrying out these tasks?

- 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.
- 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.
- Monitoring the application in Live is also the responsibility of Development, rather the entire team.
- How to monitor is the biggest aspect of the application monitoring. As the speed and the accuracy expected here is very high and crucial in order to address the issues on time.

pplication Performance Monitorin

- **There are two to three ways of monitoring them.**

1. Through the application monitoring tools.
2. By constantly going through the logs written out by the applications, which is a manual process.
3. 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 a peak or low or 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.
- Now, this practice is not used for application monitoring purposes, but few team members, who are a part of the software upgrade team or developers, sometimes do this, just to understand what's happening on the live.
- **#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.
- Thus, amongst the above 3 methods, the first method, which is using sophisticated APM tools available in the marketplace is the most popular method that is adopted in the DevOps practice.
- The application performance monitoring starts rigorously as soon as the application is deployed and continues throughout maybe at a little slower pace.

l Tools

- As DevOps practice is becoming more and more prominent, and live metrics collection is becoming more important to meet the customers’ expectations and business needs, lots of sophisticated tools which just needs to be installed on the server and the metrics can be configured and gathered on a real-time basis have come in the market.
- So, all these aspects of the metrics collection which we spoke about just now is controlled by a good monitoring tool which provides accurate details. But the selection of the right tool becomes the success of APM.
- **Recommended Tool:**
- **#1) Datadog**
- Datadog APM helps you to easily analyze and isolate dependencies, eliminate bottlenecks, reduce latency, track errors, and increase code efficiency to optimize your applications.
- By correlating distributed traces with logs, browser sessions, profiles, synthetic tests, and infrastructure metrics, you can achieve full visibility into the health of your applications across all hosts, containers, proxies, and serverless functions.

- **Additional APM Tools:**
- **Newrelic** tool is mostly used in performance monitoring of the application like response time, throughput, most time-consuming transactions etc.,
- **ManageEngine** is another tool which supports server monitoring, DB monitoring, and cloud monitoring
- **Appdynamics** supports managing the performance and availability of the applications across the cloud computing environments and inside the data center.
- **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.
- **IBM's performance management tool** helps in efficiently managing the application, on-premises and hybrid apps and IT infrastructure.

THANK YOU