

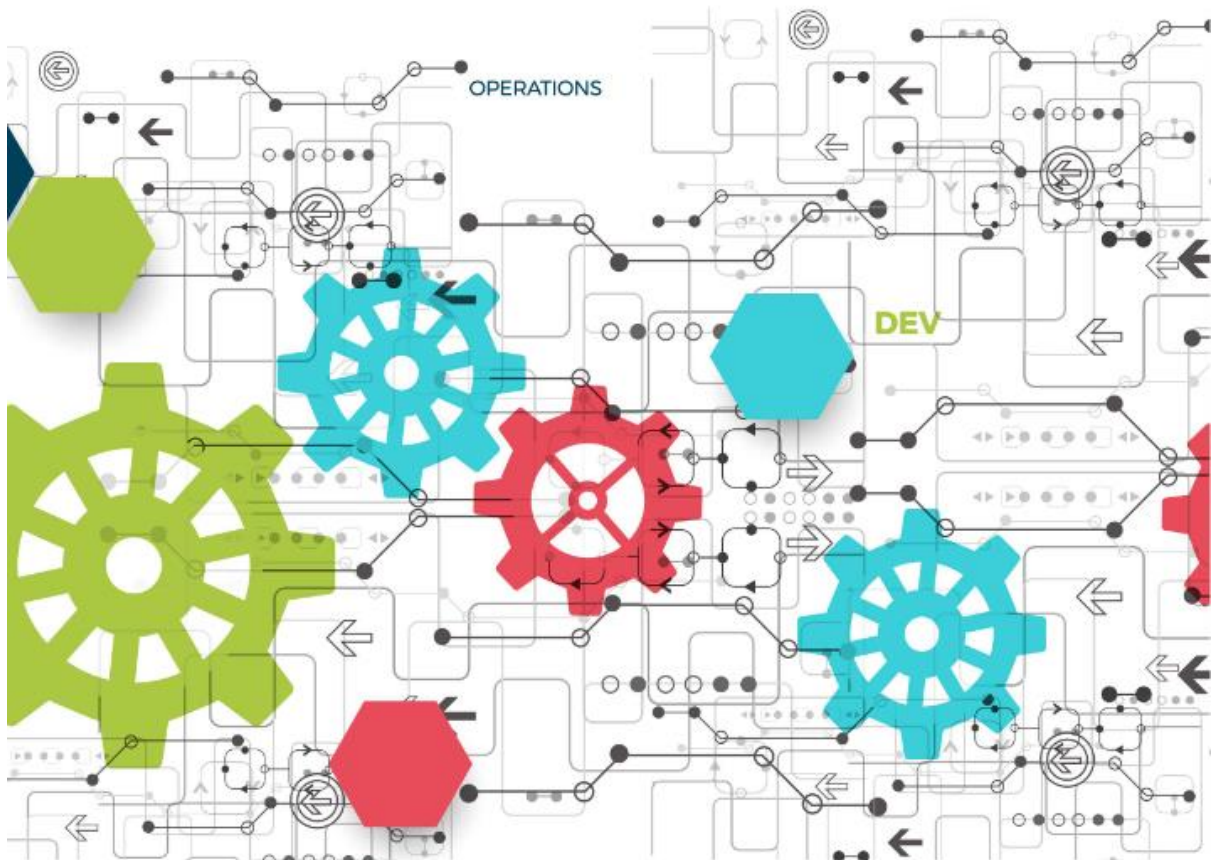


B.Tech Computer Science
and Engineering in DevOps

DEVOPS OVERVIEW

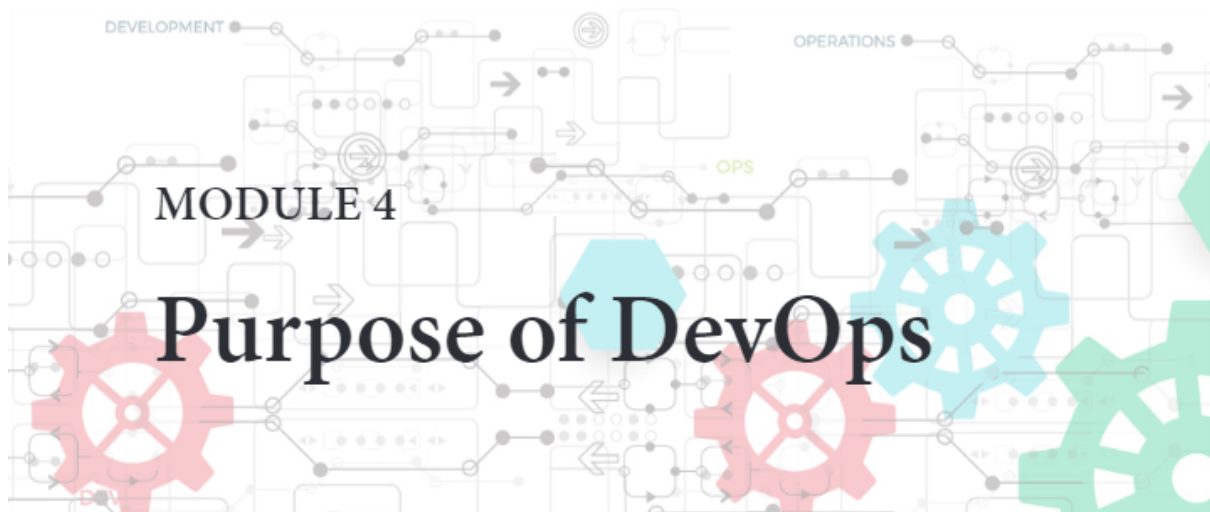
MODULE 4

Purpose of DevOps



■ Contents ■

Module Learning Objectives	1
Module Topics	1
1. Purpose of DevOps	2
2. Minimum Viable Product (MVP)	3
3. Application Deployment	8
4. Continuous Integration	16
5. Continuous Delivery	20
In a nutshell, we learnt:	24



Facilitator Notes:

Tell the participants that you will be talking about the Purpose of DevOps in this next module which is Module on DevOps Overview.

You will learn about the Purpose of DevOps in this next module which is Module on DevOps Overview.

Module Learning Objectives

At the end of the Module you would be able to learn the following

1. Why DevOps is essential in an organization
2. The different processes and methods used for DevOps implementation
3. The purpose of using Minimum Viable Product and how it helps companies get an early launch
4. Learn about the continuous integration, delivery and how it helps organizations in devops implementation
5. About ARA and deployment tools helps in mitigating the various issues that arise



Module Topics

The following topics that will be covered in the module:

1. Purpose of DevOps
2. Introduction to Minimum Viable Product(MVP)
3. The process of building an MVP



4. Application Deployment
5. Continuous Integration
6. Continuous Delivery

Facilitator Notes:

Reiterate the topics that will be covered in the module:

- Purpose of DevOps
- Introduction to Minimum Viable Product(MVP)
- The process of building an MVP
- Application Deployment
- Continuous Integration
- Continuous Deployment

1. Purpose of DevOps

Purpose of DevOps

Reduction in lead time

Enables faster release and deployment cycles

Reduction in failure of product and releases

Optimum utilization of resources

Improved collaboration

Scalability

Reliability

Facilitator Notes:

Revisit the definition of DevOps and explain in detail, how this module goes a step further and discusses in detail, the processes of DevOps.

In Module 3, we introduced the concept of DevOps and its definition. In this module, we will discuss in detail, the processes involved in DevOps and its implementation.

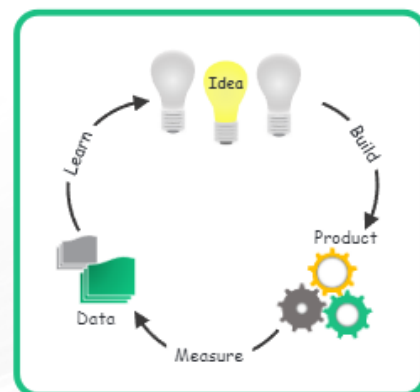
The benefits derived from using DevOps are:

- Lead time is the time taken between customer order and delivery. While working on issues, lead time is the time taken from when an issue is logged until work is completed on that issue. Adopting DevOps principles helps in reduction in lead time between the fixes.
- DevOps encourages communication, collaboration, integration and automation among software developers and IT operations in order to improve the speed and quality of delivering software.
- DevOps takes advantage of agile development methodologies that enables faster release and deployment cycles.
- Reduction in failure of product and releases, due to shorter development cycles and feedback loops and automated tests.
- Optimum utilization of resources
- DevOps fosters improved collaboration between business stakeholders, application development and operations teams.
- Scalability
- Reliability

2. Minimum Viable Product (MVP)

The term was highly popularized by Eric Ries, the author of Lean Start-up. He defined MVP as,

"that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort".



Facilitator Notes:

Revisit the traditional software product building and how organizations shipped products as one full complete unit. Talk about the Agile methodology and help students understand the process of iterative development and why in today's world at the pace with which businesses grow and compete it is not an advisable strategy to release in longer cycles. Give examples of Internet-scale applications and ask students to list down the different features of products and how all of them were not part of the first ever product release.

Introduce the participants to the idea of a Minimum Viable Product and how it has helped the companies.

Traditionally products were built as complete products with all the features and launched in the market. Product launches did not happen as often as it does today. This might not be the best way to build products today, considering the pace at which the companies launch new products and the competition that is prevalent today.

This can again be best explained by looking at the Internet scale companies. For example, let's look at WhatsApp, an application used by millions of users across the world. WhatsApp consists of multiple components such as chat, photo/video sharing, video calling, location sharing, payments etc., But did WhatsApp have all of these components right from the beginning?

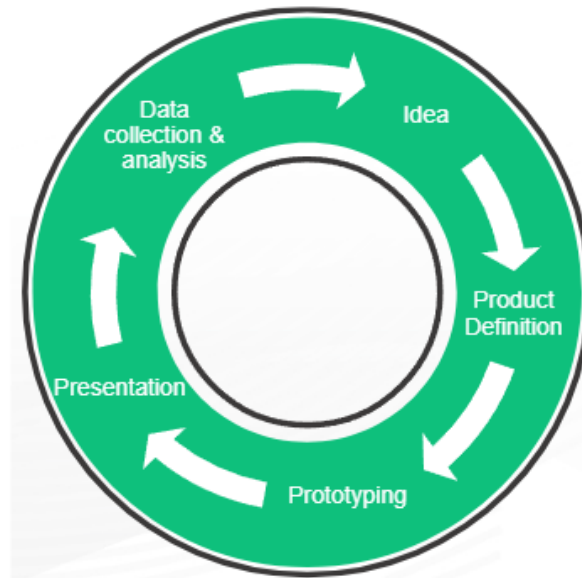
How about Facebook? The product that we use today is much more than what it was, when they started close to a decade back. Facebook's first product release did not have a facebook wall or news feed.

The idea is to roll out a basic version of the final product, get customer feedback and shape the features according to the feedback. Thus minimum viable product or MVP is that version of the product with basic features, just enough to get the attention of the customers. The final product will be shaped and released after getting sufficient feedback from the initial set of users. Definition of MVP as proposed by Eric Ryes, the author of Lean Startup, is given above.

For example, imagine a company who is building a recruitment software. As MVP, they might just release a job board, where all the jobs are listed and applicants can send resumes through email. Upon getting sufficient feedback from customers, they can introduce features like applicant tracking systems, job recommendation engine, job assessments, live interviews, etc. as different versions of the product. Eventually, the company will add more and more features to the product. By releasing the MVP, they will get to know the customer mindset, the demand and reception for the product and more importantly they get the early mover advantage. They will be able to establish themselves as a brand that people prefer.

Businesses should also be careful while adding features to the MVP. In the process of building the perfect product, some organizations, lose focus and load the MVP with large number of irrelevant features. Customers will start losing interest and eventually, the business will fail.

2.1 Minimum Viable Product (MVP) - Process



Facilitator Notes:

Explain the different steps involved in creating an MVP.

Starting from the whiteboard to creating a mini product, walk the participants through the entire process.

MVP is not just a product with half the features of the original product. It's a process that is repeated over and over again.

The five necessary steps in building the MVP are:

1. **Idea:** The first step of a successful MVP is evaluating the business idea. This is done by asking two important questions. What is the need for the product? How will the product solve the problem? These two questions will help in understanding the goal of the product and the future customer's needs.
2. **Product Definition:** Once your idea has taken shape, define the product, its vision and the value, it offers. Creating a clear product definition is important in charting the journey of MVP.
3. **Prototyping:** Once the product is defined, we create a prototype of it with minimal features, yet offering an insight into the value the product offers to its users.

4. **Presentation:** Once the prototype is created, a channel for product launch is identified and launched for customers to use and offer feedback.
5. **Data collection and Analysis:** This is the last phase of an MVP where data such as customer feedback, pain points is collated and analysed. These insights validate the product first hand and also determine the product journey going forward.

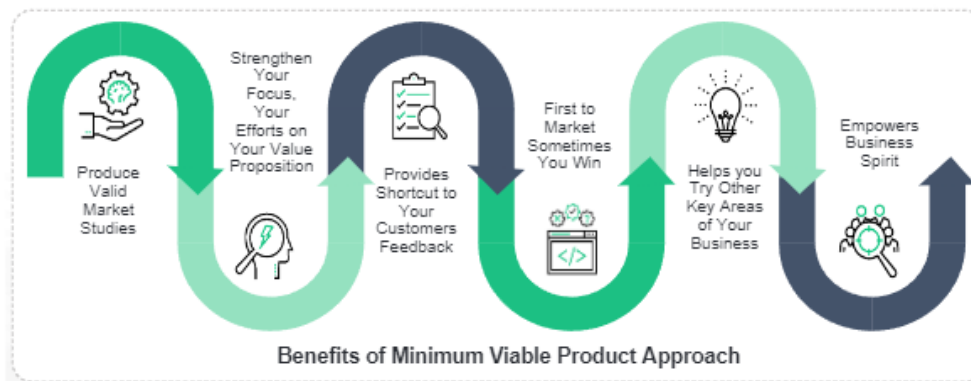
Although it is a minimized version of your product, an MVP requires a careful planning and execution. An MVP is a head-start to a product's journey. Many startups begin their journey with an MVP, primarily to launch and gain feedback from users. This feedback is very critical since it gives insights into how your product is perceived. The early adopters offer criticism, embrace and offer suggestions on what could be better or in other words, what they expect to see.

Hence an MVP is critical for the company and ensures that it creates the right MVP with the right set of features.

In short, an MVP is not a subset of your actual product. Rather it is a fully functional product by itself with a minimal set of features, that is a representative of your final product. An MVP is a miniature of a final product that has scope for more features and can scale to grow into a fully functional complete product. MVP needs to be frequently and iteratively deployed before the final product is rolled out to the market.

2.2 The Benefits of MVP

MVP offers numerous benefits such as:



Facilitator Notes:

Explain to participants in details the benefits of adapting to an iterative product development as opposed to the traditional software building. Discuss how it helps the products to be validated at an early stage by gaining customer feedback, continuously innovate and this, in the long run, reduces time & effort for product development teams.

In today's day and age, where different companies large and small build great products, the cost of failure is very high. Having said that, it is important for the companies to gain feedback and insights at every product phase to ensure they are on the right track and offer value to the customer.

Building an MVP and following an iterative approach to product development helps companies achieve that. Right from the first release, the customers offer feedback and point out what they would like to see. This approach minimizes the Capital investment and allows room for mistakes. By constantly interacting with users, there is continuous learning about the market, its needs and expectations which is critical to the success of a product. In the long-term, this approach helps the teams, save time and money.

Let's look at some of the benefits of building minimum viable product.

- **Produce valid market stores:** MVP is a tool to identify customer intentions. Doing so much market research to build the complete product may produce disastrous results. Rolling out an MVP and validating the results is the best way to understand what customer demands are.
- **Strengthen the focus on value proposition:** It is the value proposition that differentiates one product from the other product. With MVP a business can clearly define its value propositions, in spite of building so many features that are of no use to the customer.
- **A shortcut to customer's feedback:** Until the product is rolled out, a business will not be able to understand the customers' mindset. MVP is a tool to gather real-world data on customer feedback. It will give a clear picture on the ways customers will use the product. This will help in improving the final result. Using MVP, a business can make use of the customer feedback, well before the product is rolled out in the market, in other terms building what the customer wants.
- **First to market:** Early mover certainly has advantages and the chances of the product becoming a hit is twice as that of the one that arrives late in the market. Customers will certainly show interest in an innovative idea that solves an obvious problem. If there is only one solution available in the market, chances of the product becoming a hit doubles. MVP is a way to set the foot first in the market.
- **Try other areas of business:** The product needs to be released in the market to well understand customer behavior. MVP gives you a clue about customer behavior, even before the final product is built and made available.
- **Empowers business spirit:** In case of businesses that want to try out multiple ideas, MVP gets them closer to the target, which will give them more cushion in terms of time and options to try out.

2.3 Activity



Facilitator Notes:

Divide participants into groups and ask them to create an MVP for a popular app they use: Dropbox, Snapchat, WhatsApp etc.

Take an App of your choice and write down what would your feature-set be for an MVP. Write the features down keeping in mind the final product. The MVP should be a fully functional one with basic set of features.

What did you Grasp?



1. MVP stands for
 - A) Minimum Viable Product
 - B) Minimum Version Product
 - C) Maximum Viable Product
 - D) None of the above
2. Minimum viable product is a miniature version of your complete product that can be scaled to a complete product.
 - A) True
 - B) False

Facilitator Notes:

Tell the participants that it is time for a quick knowledge check.

Correct Answers

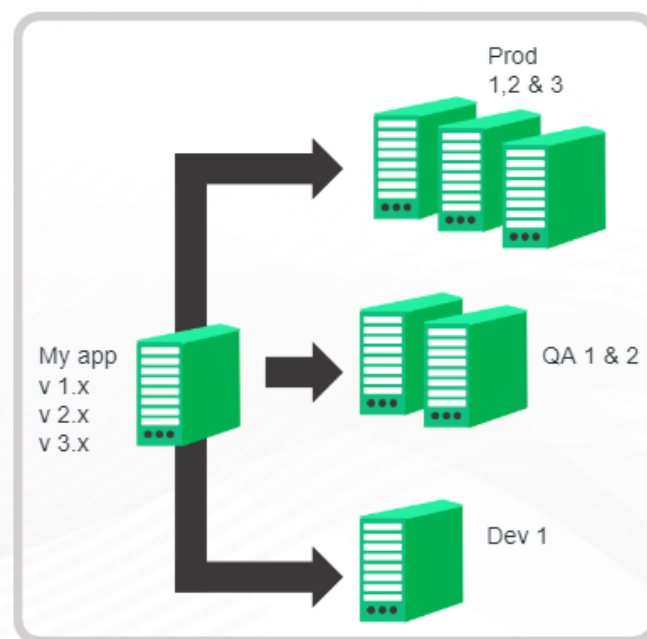
1. A. Minimum Viable Product
2. A. True

3. Application Deployment

Application deployment implies:

- Installing applications
- Updating applications

- Configuring resources
- Configuring middleware components
- Starting/stopping components
- Configuring the installed application
- Configuration systems like load balancers, routers
- Verification of components
- Scaled to the enterprise



Facilitator Notes:

Explain the participants that application deployment has to be done frequently and iteratively, once the MVP is built. Tell them what application deployment comprises and the problems associated with manual deployment.

Providing a new functionality is always a traditional clashing point between development and operations, as operations is accountable for continuity. Business units would like to have new functionality live in production as soon as possible, while Service Management is responsible for maintaining applications on a technical perspective and operations is responsible for hosting the application.

Application deployment is where the development and operations worlds meet! The defining elements of application deployment are given above. There exist a lot of confusion at the boundary of these worlds.

Developers are typically concerned with:

- Understanding requested features
- Designing components
- Writing code
- Writing test cases
- Compiling code
- Testing code

An Operations person is typically concerned with:

- Installing server hardware and Operating Software (OS)
- Configuring servers, network, and storage
- Monitoring servers
- Responding to outages
- Applying security measures
- Maintaining disaster recovery protocols

Manual Application deployment, hence is associated with certain issues. According to a report by Forrester, a majority of failed deployment occurs due to manual error, poor quality of software, incomplete software and random issues that may arise. The intensities of the issues associated with manual application deployment are as follows:

Human error - 40%

Quality of the software - 30%

Missing patches - 20%

Other reasons - 10%

Automated deployment is one of the ways to mitigate these issues. More specifically, tools like Application Release Automation (ARA) help us mitigate these issues effectively.

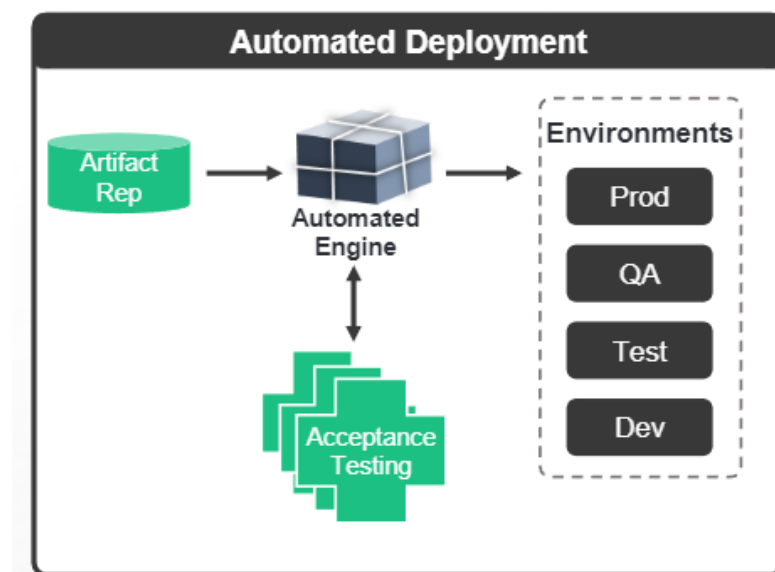
3.1 Automated Application Deployment

- The process of automatically and reliably deploying entire application runtime environments.
- Very critical for eliminating the issues associated with manual application deployment.

- Preceded by automated tests, hence feature cycle time or feature lead time is minimal by means of reducing waste

Benefits of Automated Application Deployment

- Cost effective
- Fast
- Reliable
- Maintainable
- Transparent, informative
- Auditable
- Secured
- Accessibility
- Repeatability
- Consistency
- Scalability



Facilitator Notes:

Explain the participants about the benefits of automated deployment.

Automated deployment is used to eradicate the following errors associated with manual deployment.

- Inconsistency across environments
- Slow, neither repeatable nor reliable
- Require extensive documentation (often outdated)
- Hinder collaboration (usually conducted by a few experts)

Automated deployment is usually preceded by automated tests, hence manual errors are reduced to a large extent. By enforcing an automated deployment process and treating infrastructure as code businesses can make sure that they end up with deployable code and working environments at the end of each iteration.

Benefits of automated deployment:

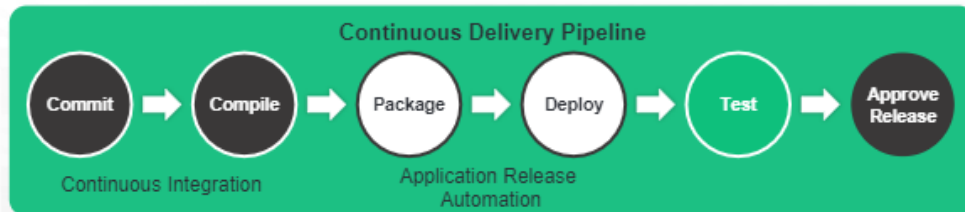
- Cost effective: Activities reusable over and over again
- Fast: Full deployments with a press of a button
- Reliable: Tested and proven deployments
- Maintainable: One standardized central point of access
- Transparent, informative: Deployments with full transparency and informative
- Auditable: Deployments can be audited easily
- Secure: Automation ensures that deployments are fully secure
- Accessible: Automated deployments are highly accessible
- Repeatable: Reuse of deployment patterns
- Consistent: Subsequent deployments identical over time
- Scalable: Deployment patterns can be applied to many environment over time

We'll now see about application release automation, an important set of tools that help in achieving any 'Continuous' effort.

3.2. Application Release Automation (ARA)

- Application Release Automation, or ARA, is the consistent, repeatable and auditable process of packaging and deploying an application or update of an application from development, across various environments, and ultimately to production.
- Xebia's definition of ARA - Tools, scripts or products that automatically install and correctly configure a given version of an application in a target environment, ready for use.

- Comprised of tools that focus on the modeling and deployment of custom application software releases and their associated configurations.
- These tools support continuous release deployment.



Facilitator Notes:

Explain application release automation to the participants.

As given above, ARA is a consistent, repeatable and auditable process of packaging and deploying an application or update of an application from development, across various environments, and ultimately to production.

With successful ARA, the need to build and maintain custom scripts for application deployments, is eliminated. Configuration errors and downtime are also reduced to a greater extent. ARA offers a model-based approach for performing critical automation tasks. As a result, the speed to market is efficiently increased, which gives stakeholders the ability to coordinate and automate releases between multiple groups and people.

3.3 Components of Application Release Automation (ARA)

There are five major components of ARA:



Facilitator Notes:

Explain the key components of ARA to the participants.

Let us discuss each of the components:

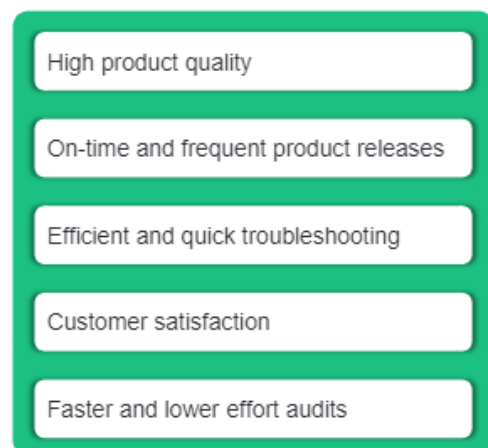
- **Packaging:** This is the first component, where items to be deployed are packed.
- **Dependencies:** This component involves modelling the dependencies required for the application and infrastructure.
- **Promotion:** This component involves delivering the test-packages to high-critical environments, e.g., promoting from Dev to QA to Staging to Production.
- **Deployment:** The Application is installed with the contents of the package and the environments are configured.
- **Compliance:** Adherence to the processes and audit requirements are documented. The Configuration of the deployed application is also validated.

Functions of ARA:

According to Gartner, an ARA solution has the key functions as follows:

- Deployment of data, application code and artifacts
- Deployment of specific configurations for each environment
- Process workflow design for automating tasks, deployment steps, and people
- Environment modeling and/or provisioning binaries

3.4 Benefits of Deployment Automation



Facilitator Notes:

Talk about how application deployment enables application to be deployed across development and production environment and the various benefits it offers. By automating your deployment it enhances the processes of both development and operations team thereby facilitating better delivery of software products.

Variability is controlled to a great extent by deployment modeling, due to which errors are minimized. This results in a higher product quality.

Product release processes are automated and accelerated. This helps in on-time and frequent product releases.

Deployment automation provides a combined access to all tools, processes and resources, which helps in efficient and faster troubleshooting and reduced time-to-market (TTM).

Effective collaboration between the Dev, QA and Ops teams helps in production of high quality software, hence customer satisfaction.

There's a centralized view of all deployment activities and the outcomes, this helps in performing faster and lower effort audits.

What did you grasp?

1. Which of the following is not a component of ARA?
 - A) Promotion
 - B) Deployment
 - C) Staging
 - D) Packaging
2. Which of the following about the functions of ARA is false?
 - A) Specific configurations are deployed for each environment
 - B) Data, application code and artifacts are deployed
 - C) ARA is not for designing process workflow for automating tasks
 - D) Environment is modeled and binaries are provisioned

Facilitator Notes:

Tell the participants that it is time for a quick knowledge check.

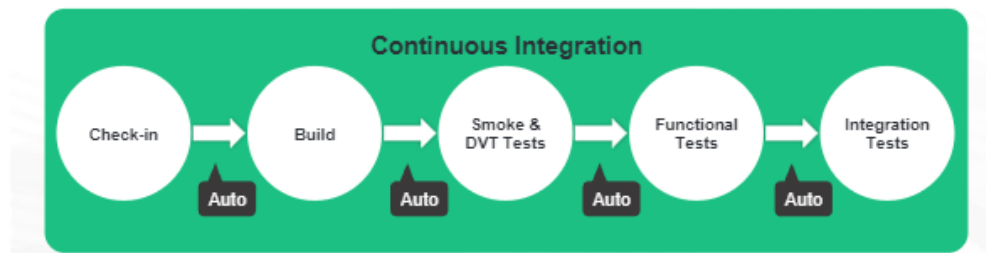
Correct Answers

1. C. Staging
2. C. ARA is not for designing process workflow for automating tasks.

4. Continuous Integration

Eric Ries, the Author of The Lean Start-up popularized the term continuous integration. It can be loosely defined as the process of building and integrating code as many times as possible in a given day.

This method of product building has been well received and vastly used by organizations globally.



Facilitator Notes:

Define continuous integration and how it came into being. Explain the need for automation by drawing analogies from different industries.

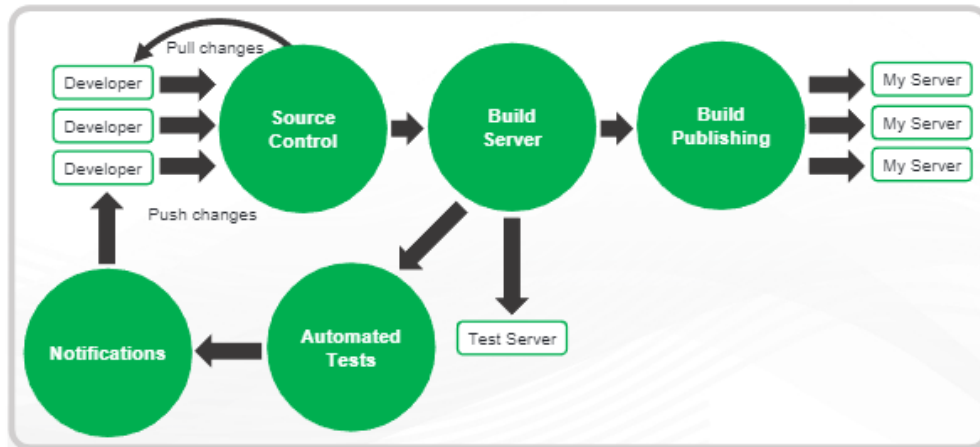
Building software is no different from that of a factory. Continuous integration is like a software assembly line set-up in an IT factory. Let's look at an example of a motor-bike & how it came into being to understand this better. A bike is not a single entity, it is a combination of different elements such as engine, headlight, metal, nuts and bolts that are fused together to create a single complete product as a motor-bike. It is most likely that each of these elements came from different sources. Each of these elements is fully functional components by itself, which went through its own set of processes before becoming a fully functional product. All these elements brought from different sources, manufactured under different conditions are assembled together at a factory to create what we see – a fully functional motorbike.

Likewise, in an IT factory, every complete product is an amalgamation of multiple software components that came from different sources, created by different people, from different geographies, across different time zones. This coming together of different components to create a complete software product is enabled by Continuous Integration in an IT software factory.

According to Wikipedia, Continuous Integration (CI) is the practice, in software engineering, of merging all developer working copies to a shared mainline several times a day.

Martin Fowler, one of the authors of Agile manifesto, defines continuous integration as, "Continuous Integration usually refers to integrating, building, and testing code within the development environment."

4.1 The Process



Facilitator Notes:

Discuss the steps involved in continuous integration.

Look at the picture to understand how an IT/software assembly works to create software products.

Continuous integration involves the following steps repeated several times over and over again.

Code->Test->Integrate->Repeat

Source Control:

Source control system maintains the history of changes made to the source code of a software. Developers first start by pulling the current version of the code from the source control repository. The development team then implements new features or fixes the bugs in the existing code or pushes the newer version of the code to the source code repository. The source code repository has the final project code at any given point in time. When another developer wants to start working and checks out their own copy of the code, the previous developer's new changes will now be included.

Build:

The build server Takes in the source code and "builds" the code into an executable program which will then be tested and deployed. The build server regularly monitors the source control system and detects any new change made to the code by the development team. The latest copy of the code is then pulled and a new program file is built, that will have the latest changes.

During the build process, a compiler converts the human readable code written by developers to machine-readable instructions. The new file can then be sent to users or deployed to servers or otherwise executed. The product building is divided into multiple smaller phases, built and released at regular intervals.

Automated tests:

Automated tests help in testing the software to ensure that the product performs the designated tasks. Automated tests greatly reduce the time spent during manual software testing and reduces the chances of bugs being present in the released product. The code that is undergone automated testing, is sent for final integration. The following are the different types of testing:

- Smoke testing - Special kind of initial checks done to ensure the proper functioning of basic implementations and environmental assumptions. These tests are usually done early during the testing cycle.
- Unit testing - Unit tests are done to test the individual components of code in an isolated and highly targeted fashion. Used for maintaining internal consistency and correctness before subjecting it to a complex context.
- Integration testing - After unit tests, integration testing is performed by grouping together components and testing them as an assembly. Integration tests are performed automatically when code is checked into the repository.
- System testing - System tests are done once integration tests are done. System tests ensure that group of components function as a cohesive whole.
- Acceptance testing - Acceptance tests are one of the last tests types that are performed on software prior to delivery. Acceptance testing is used to determine whether a piece of software satisfies all of the requirements from the business or user's perspective.

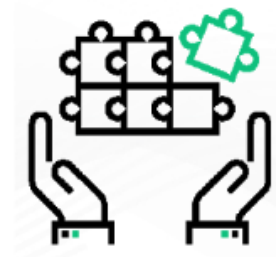
4.2 Best Practices of CI

Maintain a single code repository

- Use version control
- Store all things relevant to product in a single repository

Automate deployment

- Simplifying process of running the source code by automation
- Self testing build
- Commit code everyday
- Fix broken builds without delay
- Test code in a cloned environment



- Keep latest deliverable accessible
- Code should be available to everyone

Facilitator Notes:

Talk about what a code repository is in a development project. Compare and contrast between traditional systems and how the code was stored and maintained to how it is done today. Introduce participants to few tools that are used for version control and code maintenance.

Building software product involves a huge amount of data, files, code etc. While we are working with different teams sitting across different geographies, it gets very difficult to manage and maintain a source unless we have the right tools in place. Source code management tools are typically used to store your code and the developers use this as their reference to build from. Teams actively use what is called version-control to track and understand the different versions, changes that have been carried since day 1 of coding.

A source code management tool typically stores all things associated with the product such as database schema, files, third-party libraries, installation scripts etc.

Typically developers have what is called the main line from which different branches are born. Each branch is responsible for a set of features or a feature and different branches are built and maintained by different developers contributing to the main line.

Once we have the source code, it needs to be converted into a completed working system that can be used. This gets tricky considering the number of files, the compilation, loading schema etc. This like any other software development process can be automated.

Having automated environments for creating builds is the best practice that Development teams should follow.

4.3 Benefits of Continuous Integration


- The primary benefit of continuous integration is the pace. Bugs, errors are identified early on and fixed immediately
- Time to code review is reduced to a great extent.
- Support for parallel builds, builds and tests can be carried out over multiple machines.
- Developers integrate code and identify issues within the next couple of hours
- Reduction in the number of errors or bugs
- Code is tested on a regular basis, that reduces the overall number of errors or bugs in the product.
- With automated testing, code is tested in the same way for every change and every change is tested before deployed to production.
- Latest version of code is available always.

Facilitator Notes:

Discuss the benefits that continuous integration offers as opposed to a traditional software building process.

The main advantage of CI is the rapidity with which code is built and tested. The feedback loop is almost immediate and this helps development teams save a lot of time and cost. Benefits of CI are listed above.

What did you Grasp?



1. Maintaining multiple source code repositories is one of the best practices of continuous integration.
A) True
B) False
2. Which of the following tests is done to test individual elements of the code?
A) Unit test
B) System test
C) Integration test
D) Smoke test

Facilitator Notes:

Tell the participants that it is time for a quick knowledge check.

Correct Answers

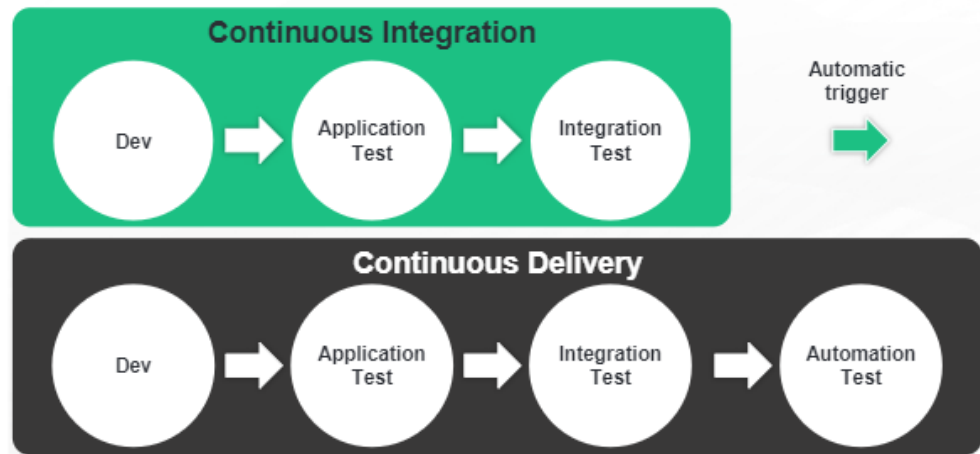
1. b. False
2. a. Unit test

5. Continuous Delivery

Continuous Delivery (CD) is the process to build, test, configure and deploy from a build to a production environment.

A series of processes that ensure code is deployed to production rapidly and safely, by delivering every change to a production-like environment.

An expansion to continuous integration.

**Facilitator Notes:**

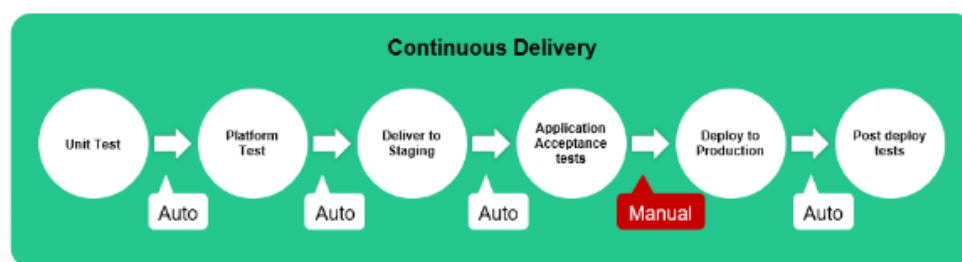
Explain the basic concepts of continuous delivery.

In continuous delivery, the code changes are automatically built, tested and prepared for release to the production. All code changes are deployed to a testing and/or production environment after the build is completed. If implemented properly, continuous delivery will help developers have a deployment-ready build artefact that has already passed through a set of standard testing processes.

Continuous delivery lets the developers do automated testing apart from unit tests, which help them verify the application updates across multiple dimensions before it is deployed to customers. These tests may include UI testing, load testing, integration testing, API reliability testing, etc. By means of these testing processes, developers can detect issues early.

With the advent of cloud, automating creation and replication of multiple environments for testing has become easy and cost-effective, which were earlier difficult to do on-premise.

5.1 Continuous Delivery Process



Facilitator Notes:

Explain the process of 'Continuous Delivery' as per the process chart on the slide.

Continuous delivery automates the complete software release process. Every change made to code enables an automated process to build, test and stage the update. The final step of deploying the update to a production environment is done manually by the developer.

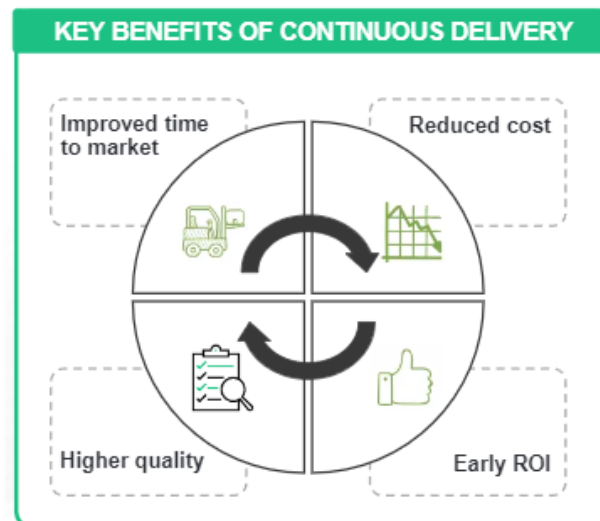
Once a code is committed, automated testing processes are triggered. Tests like a unit test, platform test are done and the update is first deployed in a staging environment. Once delivered to staging, application acceptance tests are done and the code is then deployed to production. There can be multiple, parallel test stages before a production deployment.

Till the process of application acceptance tests, all the steps are automatically carried out. The final step of deployment is manually triggered by the developer and again post-deployment tests are done automatically.

Being a lean process, the goal of CD is to keep the production fresh by achieving the shortest path from coding to deployment.

5.2 Benefits of Continuous Delivery

- Automation of software release process
- Improved developer productivity
- Discover and fix bugs quicker
- Deliver updates faster



Facilitator Notes:

Explain the various benefits of continuous delivery.

From building to production the process is entirely automated. Automation of building, testing and preparing code updates for a release, helps in the rapid and efficient delivery of software.

With continuous delivery, the productivity of developers increase, as they are freed up from doing manual tasks. It also helps in identifying the bugs earlier, so an error-free working software is delivered to the customers in an efficient way.

With automated tests, developers can discover and fix bugs very early during the development process. Tests are more frequent and comprehensive, hence bugs do not grow into larger complex problems. Apart from the unit tests, other types of tests can also be performed easily with continuous delivery.

Updates in the product are delivered faster and more frequently. Deployment-ready updates are always available.

What did you Grasp?

1. In Continuous delivery, deployment to production environment requires manual approval.


A) True
B) False

Facilitator Notes:

Tell the participants that it is time for a quick knowledge check.

Correct Answer

1. A. True



2. Which of the following statements about continuous delivery is NOT correct?

- A) Continuous delivery ensures rapid and safe deployment of code
- B) Developer triggers the final approval before the code is deployed to production
- C) Continuous delivery allows only unit testing to be done
- D) Every change in code is deployed to a staging environment before making it live to production

Facilitator Notes:

Tell the participants that it is time for a quick knowledge check.

Correct Answer:

2. C. Continuous delivery allows only unit testing to be done

In a nutshell, we learnt:



1. The need for DevOps implementation and the processes involved in the implementation
2. The different methods that lead an organization towards DevOps implementation - MVP, CI and CD
3. The process, its benefits and best practices to implement MVP, CI and CD
4. The importance of application deployment and how automation helps facilitate DevOps

Facilitator Notes:

Tell the participants that you will summarize the salient points of what has been covered.

Reiterate the important points of the discussion.

Release Notes

B. TECH CSE with Specialization in DevOps Semester One -Year 01

Release Components.

Facilitator Guide, Facilitator Course Presentations, Student Guide and Mock exams.

Current Release Version.

1.0.0

Current Release Date.

2 July 2018

Course Description.

Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

Copyright © 2018 Xebia. All rights reserved.

Please note that the information contained in this classroom material is subject to change without notice. Furthermore, this material contains proprietary information that is protected by copyright. No part of this material may be photocopied, reproduced, or translated to another language without the prior consent of Xebia or ODW Inc. Any such complaints can be raised at sales@odw.rocks

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Feb 2019