CONTINUOUS DELIVERY AUTOMATION FOCUS TOPICS

Focus Topics – An Overview

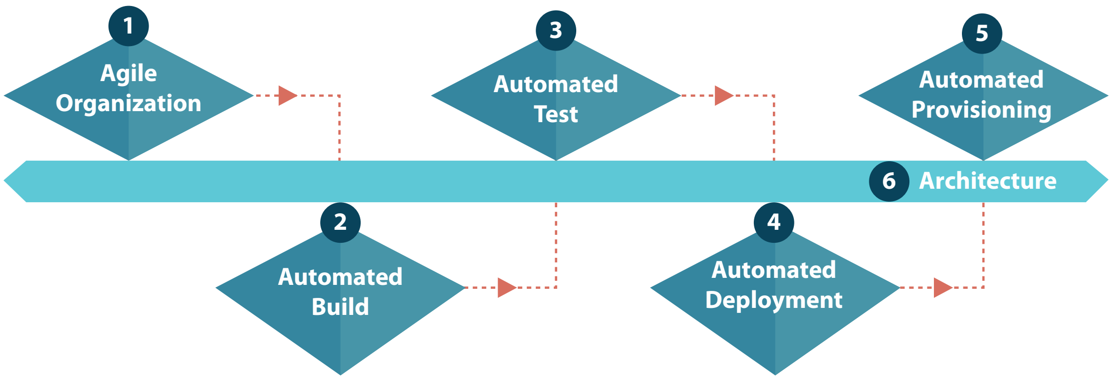
Focus topics, regardless of whether automated or not, help define and implement Continuous Delivery practices. Organizations that have adopted or are adopting Continuous Delivery practices use a number of frameworks and tools for these topics to successfully implement Continuous Delivery.

INFO

Focus topics

Out of the six focus topics of Continuous Delivery, four are automated (2, 3, 4, and 5), and two (1 and 6) are non-automated.

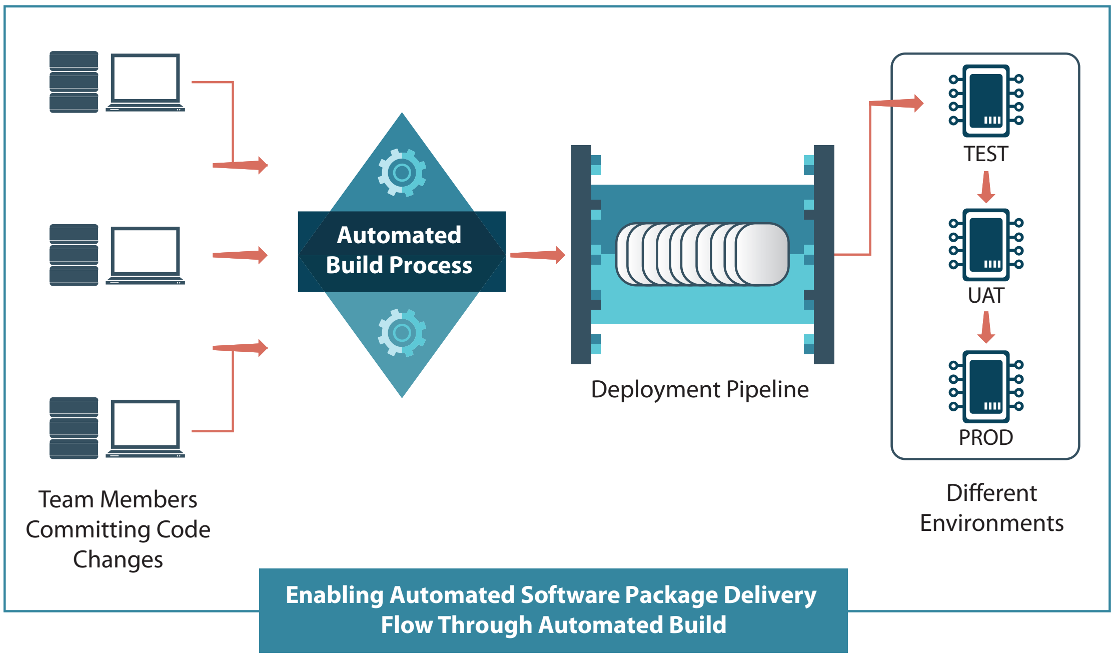
Focus Topics



The best fit solution for each organization depends on the application and infrastructure technology, DevOps maturity, and the problem/ business domain.

This topic focuses on the automated focus topics.

Automated Build: An Automated Flow in Software Package Delivery


Team Members Committing Code Changes
Automated Build Process
Deployment Pipeline
TEST
UAT
PROD
Different Environments
Enabling Automated Software Package Delivery Flow Through Automated Build

UAT = User Acceptance Test and PROD = Production

deployable components

Deployable components are always ready to be deployed in different environments.

INFO

Build automation

An automated build system usually contains Source Version Control System, Continuous Integration Server, Code Quality Analysis and Reporting, Artifact Repository, Build Tools, Static Analysis, and Unit Test and Test Runner Frameworks.

Build automation is the process of automatically transforming the changes in code to a collection of published deployment artifacts (or Deployment Pipeline), which are the **deployable components** of your application software. The preceding figure depicts the automated build process.

Whenever changes in the code are committed by the Development team, the **automated build** process performs the various tasks. These are:

1. Merge the new changes.
2. Analyze and compile the complete code.
3. Perform unit tests.
4. Assemble the code.
5. Create a new deployment package.
6. Publish it to an artifact repository.

The six steps are performed in a sequence, thus, enabling continuous flow from the code commit to the validated deployment package.

The automated build is the first point of receiving feedback related to the quality of your product/application software. It provides feedback on the committed code changes. If there are errors in the code, the build fails. The Development team rapidly takes the required actions to fix the errors and executes the automated build process again. In this way, automated build supports the concept of failing fast and often.

Automated Test: Merging Specification and Verification

Several techniques are available today to automate the testing process. Organizations can use any technique or combination of techniques to perform automated testing. Two of the popular and effective software development techniques that involve automated testing are Test-Driven Development (TDD) and Behavior-Driven Development (BDD).

TDD	BDD
During TDD, developers write the automated test cases before writing the functional pieces of the code.	During BDD, a product owner/ business analyst/QA defines the user behavior before writing the automated test cases and the functional pieces of code.
Both techniques, TDD and BDD, are popular in Agile methodologies and end up with a shippable product at the end of a sprint.	

The various steps involved in TDD are:

1. A developer writes automated test cases/scripts considering the given requirements.
2. The Development team executes the automated test case against the currently developed functionality. All tests fail for the first time in the absence of any functionality.

3. The Development team writes enough code to make the automated test pass.
4. The Development team refactor the code (without adding any new functionality) to improve the quality.
5. The Development team produces a tested deliverable at the end of the sprint.

The various steps involved in BDD are:

1. A product owner/business analyst/QA defines the user's behavior in simple English.
2. A developer (or concerned person) transforms the user's behavior to automated test case/script.
3. The Development team writes the code to implement the required functionality.
4. The Development team refactor the code (without adding any new functionality) to improve the quality.
5. The Development team produces a tested deliverable at the end of the sprint.

Let's discuss some differences between TDD and BDD:

1. "BDD is in a more readable format by every stakeholder since it is in English, unlike TDD test cases written in programming languages such as Ruby and Java.
2. BDD explains the behavior of an application for the end-user, while TDD focuses on how functionality is implemented. Changes in functionality can be accommodated with less impact in BDD as opposed to TDD.
3. BDD enables all the stakeholders to be on the same page with requirements. It makes acceptance easy as opposed to TDD.

The behavior of the application is the central idea in BDD. It focuses on the customer and pushes developers and testers to walk in the customer's shoes. If actions do not affect the end-user, BDD might not represent such a scenario well, whereas TDD can better serve the purpose.

Like many other software development practices, it might not be feasible to identify what works universally for all projects. Systems driven by actions of the end-user, such as an e-commerce website or an HR system, BDD acts as a good medium to capture all the user actions. On the other hand, TDD might be a better solution for systems that have third-party API calls, cron jobs, data exports/imports, and other related."

The quoted text is adapted from https://www.glowtouch.com/test-driven-development-vs-behavior-driven-development/#top.

Application deployments

Application deployments define the package of software components that make up an application in a particular environment.

INFO

confusion point

Developers are concerned with their specific tasks, such as understanding requested features, designing components, writing code and test cases, compiling code, and testing code. Operators are concerned with their specific tasks, such as installing server hardware and Operating Software, configuring servers, network, and storage, monitoring servers, responding to outages, applying security measures, and maintaining disaster recovery protocols.

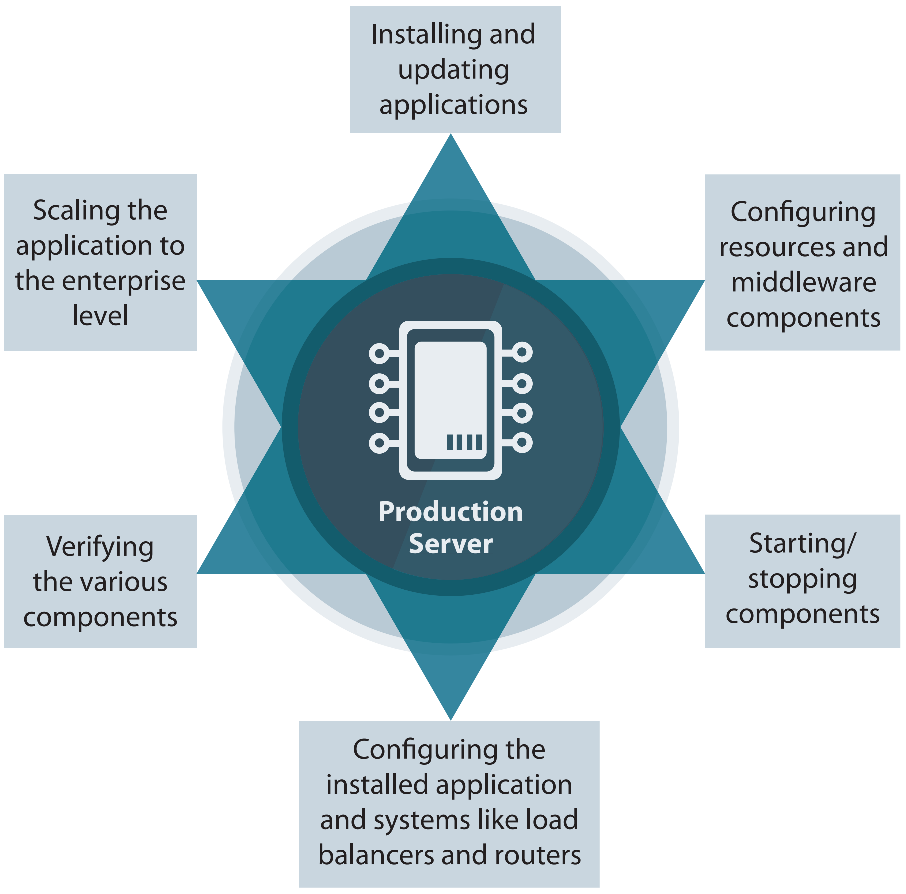
INFO

grown works of art

An unknown number of ad-hoc commands are executed often on the production server over time.

Automated Deployment: Behind the Scene

Automated deployment is about automatic **application deployments**. It is the point where the two worlds (Development and Operations) meet! You can also call it the **confusion point** as the two teams are concerned about their own tasks. However, the flashlight should be on the PRODUCTION server, which can be described as "**grown works of art**."


Installing and updating applications
Configuring resources and middleware components
Starting/ stopping components
Configuring the installed application and systems like load balancers and routers
Verifying the various components
Scaling the application to the enterprise level
Production Server

What does an application deployment include?

Due to the lack of the complete understanding of the Production environment, reproducing it becomes difficult. Restoring an environment through backups will be an option, however, the success will still be uncertain. How frequently do you verify your backup strategy and assess the Mean Time to Recover (MTTR)? In addition, it involves considerable efforts (the resources).

Restoring an environment through backups is manual and time-consuming. You cannot even think of rapidly and reliably recreating any environment on demand - from development to production – due to the lack of flexibility and agility. Continuous Delivery, therefore, focuses on automated push-button deployments to different environments and ensures repeatability and reliability.

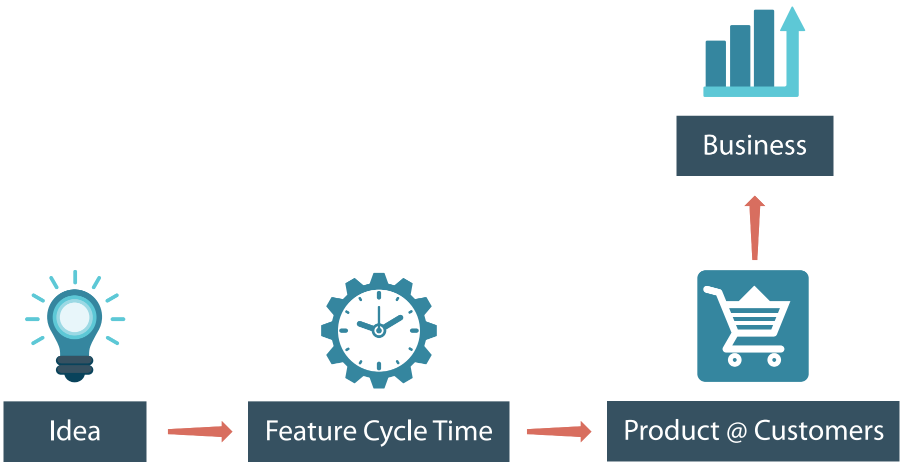
Automated Deployment: Minimizing Feature Cycle Time

The automatic, rapid, and reliable deployments of Agile software products to the different environments help minimize **feature cycle time**, which is the ultimate aim of Continuous Delivery. The automated deployments through Agile development, thus, have the power to minimize the time that it takes from ideation of features to the final product at customers by reducing waste.

DO YOU KNOW?

feature cycle time

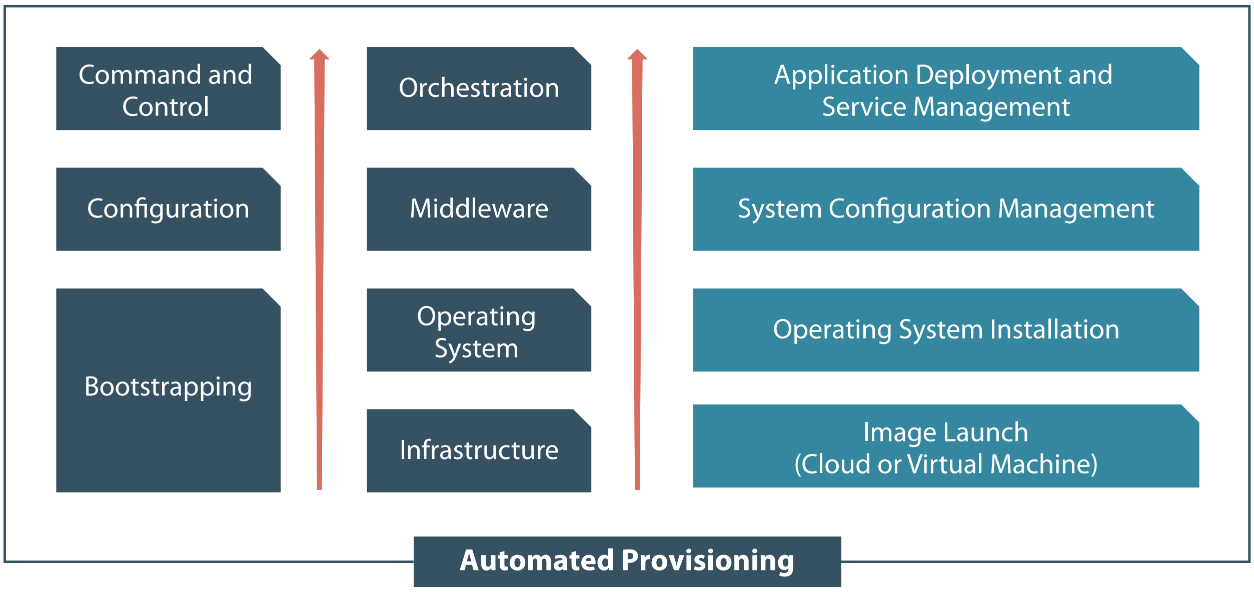
Cycle time is also known as feature cycle time or lead time.



Time-to-market is always the most important consideration. Being fast in making the new feature available to the users requires you to be faster in knowing its market acceptance. As a result, the organization will also be faster in earning profits.

Automated Provisioning

Automated provisioning is the ability to deploy, update, and repair the complete application infrastructure (such as network components, server components, and runtime software stacks) using a predefined set of automated procedures. It usually takes a toolchain approach as shown in the following figure.


Command and Control
Configuration
Bootstrapping
Orchestration
Middleware
Operating System
Infrastructure
Application Deployment and Service Management
System Configuration Management
Operating System Installation
Image Launch (Cloud or Virtual Machine)
Automated Provisioning

The figure consists of three parts:

- **Bootstrapping**: Prepare any specific version of the operating system and have it automatically stamped out across multiple machines, such as physical, local virtual machines and cloud-based virtual machines. A variation of this step that has become popular with the rise of cloud infrastructure is to "pre-bake" and manage a library of mostly configured virtual machine images that can be copied and produced when required. There are trade-offs with either approach.
- **Configuration**: After deploying the operating system, there is still a considerable amount of work that needs to take place. You must ensure that all of the OS, network, and security settings are correct for each specific instance. You also must ensure that all of the correct versions of libraries and base packages are at the required locations. This layer of automation also gives you the ability to automatically spot and correct "configuration drift" or unintended changes as they take place.
- **Command and Control**: It is related to Orchestration, which "is concerned with coordinating actions that have to take place across multiple tiers (Web, application, database) and multiple servers to start, update, and manage the integrated system. Like the Configuration layer of automation, Orchestration must also be specification driven. Dependencies and ordering are critical. The Orchestration layer of automation can also provide the facilities and **runbook** necessary to speed ad-hoc management and emergency troubleshooting.

runbook

In a computer system or network, a runbook is a compilation of routine procedures and operations that the system administrator or operator carries out. System administrators in IT departments and NOCs use runbooks as a reference. Runbooks can be in either electronic or in physical book form.

https://en.wikipedia.org/wiki/Runbook#:~:text=In%20a%20computer%20system%20or,or%20in%20physical%20book%20form.

Automated provisioning enables Development and Quality Assurance teams to operate a more efficient and reliable application lifecycle by providing safe self-service capabilities for deploying and managing their environments. Automated provisioning also enables system administrators in production environments to operate their applications with unachievable levels of efficiency and reliability.