DevOps Terms You Need to Know - A DevOps Glossary

Updated 17-April-2018

A

Agent

A program installed on specific physical servers in order to handle the execution of various processes on that server.

Agile Software Development

A software development methodology and philosophy, focused on user feedback, software quality, and the ability to respond quickly to changes and new product requirements

Application Release Automation (ARA)

A practice of deploying software releases to various environments and their configurations with as little human interaction as possible.

Artifact

Any description of a process used to create a piece of software that can be referred to, including diagrams, user requirements, and UML models.

Autonomy

The ability to make changes with the resources currently available, without the need to defer to something or someone higher up in the hierarchy.

B

Behavior-Driven Development (BDD)

An evolution of test-driven development that focuses on collaboration between development and business stakeholders to define user stories that determine the development of the application using a human-readable DSL.

Branching

The duplication of an object under review in source control so that the same code can be modified by more than one developer in parallel.

Build Agent

A type of agent used in continuous integration that can be installed locally or remotely in relation to the continuous integration server. It sends and receives messages about handling software builds.

Build Artifact Repository

Centralized storage for all binaries used during build. Simplifies dependency management and build processes, helps maintain security and consistency across teams, helps make automated deployment practical and scalable.

C

Capacity Test

A test that is used to determine the maximum number of users a computer, server, or application can support just before failing.

Commit

A way to record the changes to a repository and add a log message to describe the changes that were made.

Complex-Adaptive Systems

Any system made of a collection of similar, smaller pieces that are dynamically connected and can change to adapt to changes for the benefit of a macrostructure.

Configuration Drift

How software and hardware configurations become inconsistent with the master version due to manual and ad hoc changes (like hotfixes) that are not committed back to version control. Often a significant source of technical debt.

Configuration Management

A process for establishing and maintaining consistent settings of a system. These solutions also include SysAdmin tools for IT infrastructure automation (e.g. Chef, Puppet, etc.).

Containerization

Resource isolation at the OS (rather than machine) level, usually (in UNIX-based systems) in user space. Isolated elements vary by containerization strategy and often include file system, disk quota, CPU and memory, I/O rate, root privileges, and network access. Much lighter-weight than machine-level virtualization and sufficient for many isolation requirement sets.

Containers

Resource isolation at the OS (rather than machine) level, usually (in UNIX-based systems) in user space. Isolated elements vary by containerization strategy and often include file system, disk quota,

CPU and memory, I/O rate, root privileges, and network access. Much lighter-weight than machine-level virtualization and sufficient for many isolation requirement sets.

Continuous Delivery

A software engineering approach in which continuous integration, automated testing, and automated deployment capabilities allow software to be developed and deployed rapidly, reliably, and repeatedly with minimal human intervention.

Continuous Deployment

A software development practice in which every code change goes through the entire pipeline and is put into production automatically, resulting in many production deployments every day. It does everything that Continuous Delivery does, but the process is fully automated, and there's no human intervention at all.

Continuous Integration

A software development process where a branch of source code is rebuilt every time code is committed to the source control system. The process is often extended to include deployment, installation, and testing of applications in production environments.

Continuous Quality

A principle that preaches the continuous quest for quality across the entire SDLC, starting from requirements definition, code development, testing, and operations. Another key area of focus for Continuous Quality is the application code pipeline orchestration. There are many opportunities to negatively impact the quality of an application when code is being manually moved across environments.

Continuous Testing

The process of executing unattended automated tests as part of the software delivery pipeline across all environments to obtain immediate feedback on the quality of a code build.

D

Data Driven Development

Data driven programming is a programming model where the data itself controls the flow of the program and not the program logic. It is a model where you control the flow by offering different data sets to the program where the program logic is some generic form of flow or of state-changes. Often a state engine controls the application. Often used in automation of controlling gates.

For example, if you have program that has four states: UP - DOWN - STOP - START

You can control this program by offering input (data) that represents the states:

set1: DOWN - STOP - START - STOP - UP - STOP

set2: UP - DOWN - UP - DOWN

The program code stays the same but data set (which is not of a dynamic input type but statically given to the computer) controls the flow.

Deployment

A term that refers to the grouping of every activity that makes a program available for use and moving that program to the target environment.

Deployment Pipeline

A deployment pipeline is an automated manifestation of your process for getting software from version control into the hands of your users. (source: informIT.com)

DevOps

An IT organizational methodology where all teams in the organization, especially development teams and operations teams, collaborate on both development and deployment of software to increase software production agility and achieve business goals.

E

Event-Driven Architecture

A software architecture pattern where events or messages are produced by the system, and the system is built to react, consume, and detect other events.

Exploratory Testing

A manual testing strategy where human testers have the freedom to test areas where they suspect issues could arise that automated testing won't catch.

F

Fail Fast

A strategy in which you try something, it fails, feedback is delivered quickly, you adapt accordingly, and try again.

Infrastructure-as-a-Service (IaaS)
A self-service computing, networking, and storage utility on-demand over a network.

Integration Testing

Testing that occurs after unit testing, but before validation testing, where individual software components are combined and tested as a single group to ensure they work as a whole.

Issue Tracking
A process that allows programmers and quality assurance personnel to track the flow of defects and new features from identification to resolution.

L

Lead Time
The time it takes to move work in progress (WIP) to a finished state in a manufacturing plant. In software development, this is represented by moving code changes to production.

M

Mean Time Between Failures (MTBF)
Used to measure reliability of a system or component, calculated by averaging the time between system failures.

Mean Time to Recovery (MTTR)
The average time it takes a system or component to recover from a failure and return to production status.

Microservices Architecture
The practice of developing software as an interconnected system of several independent, modular services that communicate with each other.

Model-Based Testing
A software testing technique in which the test cases are derived from a model that describes the functional aspects of the System Under Test (SUT). Visual models can be used to represent the

desired behavior of a SUT, or to represent testing strategies and a test environment. From that model manual tests, test data, and automated tests can be generated automatically.

O

One-Stop Shop / Out-of-the-Box Tools

Tools that provide a set of functionalities that works immediately after installation with hardly any configuration or modification needs. When applied to the software delivery, a one-stop shop solution allows quick setup of a deployment pipeline.

P

Pair Programming

A software development practice where two developers work on a feature, rather than one, so that both developers can review each others' code as it's being written in order to improve code quality.

Platform-as-a-Service (PaaS)

Provides languages, libraries, services, and tools that allow developers to build and deploy applications in the cloud without worrying about underlying OS-level infrastructure (or below).

Production

The final stage in a deployment pipeline where the software will be used by the intended audience.

R

Rollback

An automatic or manual operation that restores a database or program to a previously defined state.

S

Self-Service Deployment

The action of automating deployment processes enough for developers to allow project managers or even clients to directly control deployments.

Source Control

A system for storing, tracking, and managing changes to software. This is commonly done through a process of creating branches (copies for safely creating new features) off of the stable master version of the software and then merging stable feature branches back into the master version. This is also known as version control or revision control.

Staging Environment

Used to test the newer version of your software before it's moved to live production. Staging is meant to replicate as much of your live production environment as possible, giving you the best chance to catch any bugs before you release your software.

T

Technical Debt

A concept in programming that reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution.

Test Automation

The use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes.

Test Driven Development (TDD)

Test-driven development (TDD) is a development technique where you must first write a test that fails before you write new functional code. TDD is being quickly adopted by agile software developers for development of application source code and is even being adopted by Agile DBAs for database development.

U

Unit Testing

A testing strategy in which the smallest unit of testable code is isolated from the rest of the software and tested to determine if it functions properly.

User Acceptance Test

The final phase of software testing where clients and end users determine whether the program will work for the end-user in real world scenarios. This stage is also known as beta testing.

V

Virtual Machine (VM)

A software emulation of a physical computing resource that can be modified independent of the hardware attributes.