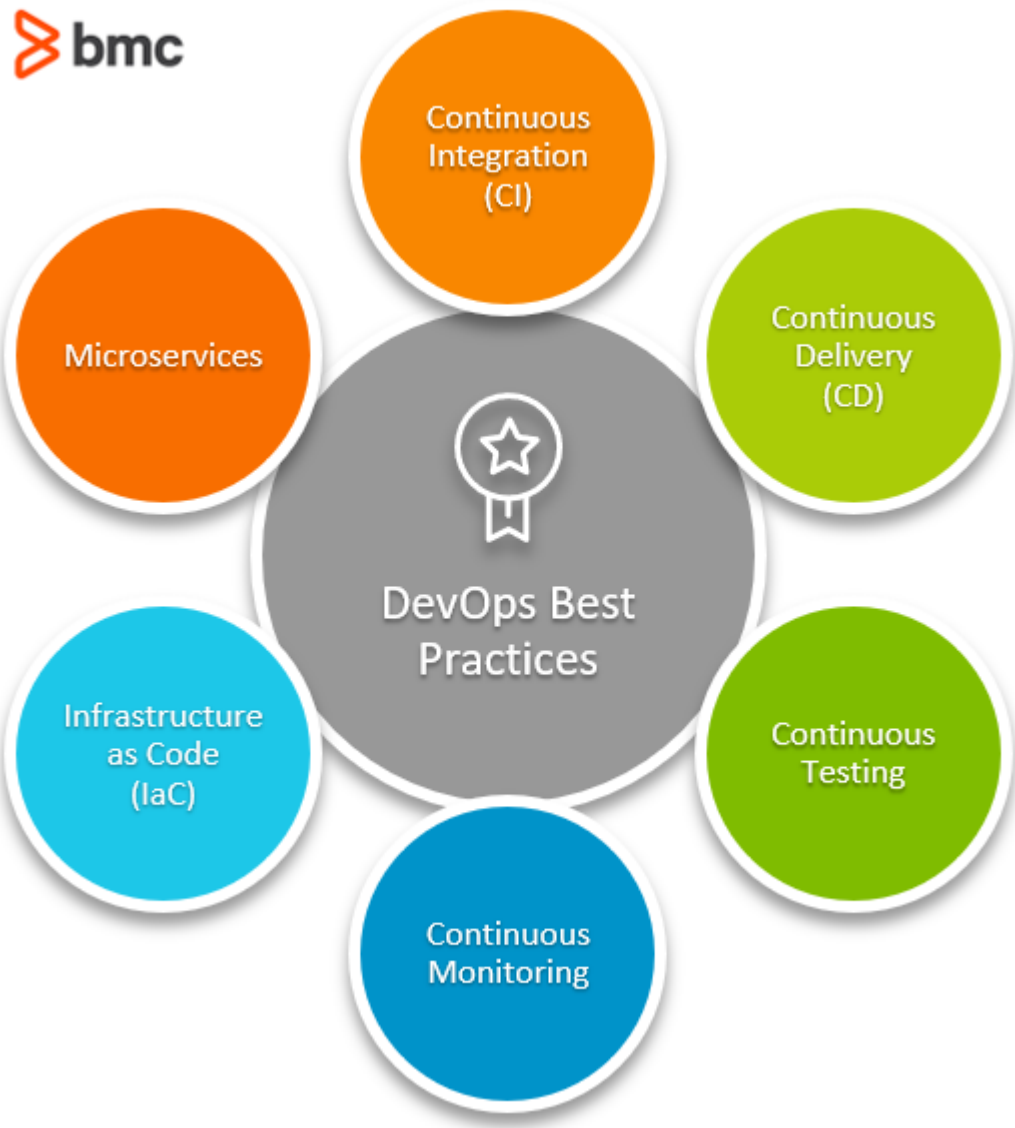
EXPLAINED: MONITORING & TELEMETRY IN DEVOPS

DevOps is a data-driven software development lifecycle (SDLC) framework. DevOps engineers analyze logs and metrics data generated across all software components and the underlying hardware infrastructure. This helps them understand a variety of areas:

- Application and system performance
- Usage patterns
- Bugs
- Security and regulatory issues
- Opportunities for improvement

Extensive application monitoring and telemetry is required before an application achieves the coveted Service Level Agreement (SLA) uptime of five 9's or more: available at least 99.999% of the time. But what exactly is monitoring and telemetry and how does it fit into a DevOps environment? Let's discuss.

(This article is part of our DevOps Guide. Use the right-hand menu to go deeper into individual practices and concepts.)



What is monitoring?

Monitoring is a common IT practice. In the context of DevOps, monitoring entails the process of collecting logs and metrics data to observe and detect performance and compliance at every stage of the SDLC pipeline. Monitoring involves tooling that can be programmed to

- Procure specific log data streams
- Produce an intuitive visual representation of the metrics performance
- Create alerts based on specified criteria

The goals of monitoring in DevOps include:

- **Improve visibility and control of app components and IT infrastructure operations.** Applications can range from cybersecurity to resource optimization. For instance, monitoring tools can alert incidents of network breaches and excessive network traffic at a specific node.
- **Monitor application performance issues, identify bugs, and understand how specific app components behave in production and test environments.** Once deployed, monitoring tools alert on several metrics to track resource utilization and workload distribution. With this information, engineers can allocate resources to account for dynamic traffic and workload demands.

- **Understand user and market behavior.** This information can help engineers make technical decisions such as adding a specific feature, removing a button, or investing in cloud resources to further improve the SLA performance. Proactive decision making in this regard helps organizations maintain and expand their market share in the competitive business landscape.

(Explore continuous delivery metrics, including monitoring.)

What is telemetry?

Telemetry is a subset of monitoring and refers to the mechanism of representing the measurement data provided by a monitoring tool. Telemetry can be seen as agents that can be programmed to extract specific monitoring data such as:

- High-volume time-series information on resource utilization
- Real-time alerting for specific incidents

DevOps monitoring vs telemetry

Consider the case of motor racing where fans get to see metrics such as top speed, G-forces, lap times, race position, and other information that displays on TV screens. These measurement displays refer to the telemetry.

Conversely, the process of installing sensors, extracting data, and providing a limited set of metrics information onto TVs is, in its entirety, called monitoring.

In the context of DevOps, some of the most common metrics measured are related to the health and performance of an application, and various corresponding metrics are always visible at the dashboard.

Monitoring challenges

Before discussing the various DevOps use cases of telemetry, let's discuss the most common monitoring challenges facing DevOps organizations:

- Operations personnel invest significant time and resources to find performance issues on the infrastructure and apps.
- Devs frequently interrupt their development work to address new bugs and issues identified at the production stage.
- The rapid release cycle approach makes apps prone to performance issues—thorough testing takes time and resources that may not be justified from a business perspective.
- The deployment procedure is complex: engineers need to synchronize and coordinate multiple development workstreams, within microservices, multi-cloud, and hybrid IT
- Anomalies are a sign of potential emerging issues. It's important to identify and contain the damages before the impact is realized and spreads across the global user base.
- Security and regulatory restrictions require organizations to exercise deep control and maintain visibility into the hardware resources operating sensitive user data and applications. This is challenging, especially when the underlying infrastructure is a cloud network operating off-premise by a third-party vendor that can offer only limited logs data, metrics information, and insights into the hardware components.

Monitoring & telemetry use cases

In order to address these challenges, DevOps teams use a variety of monitoring tools to carefully identify and understand patterns that could predict future performance of an app, service, or the underlying infrastructure.

Some of the common use cases of telemetry in DevOps include the following metrics and use cases:

- Health of build releases from the source code
- Logs information from artifact repositories
- Continuous integration failures at the CI server
- Testing performance at various stages of the SDLC pipeline
- User engagement and visibility for specific app features
- Uptime and performance metrics such as MTTF, MTTD, and MTTR
- Anomaly detection and network traffic routing.

Data analysis is necessary

Analysis follows monitoring. Telemetry doesn't necessarily include analyzed and processed logs or metrics information. The decision making based on telemetry of log metrics requires extensive analysis of a variety of KPIs and can be integrated with the monitoring systems to trigger automated actions when necessary.

Related reading

- BMC DevOps Blog
- State of IT Monitoring: A Report Roundup
- Choosing IT Metrics That Matter
- DevOps Metrics & KPIs
- Monitoring Microservices with Spring Boot Actuator & AspectJ
- How Workflow Orchestration Improves Application Development & Monitoring