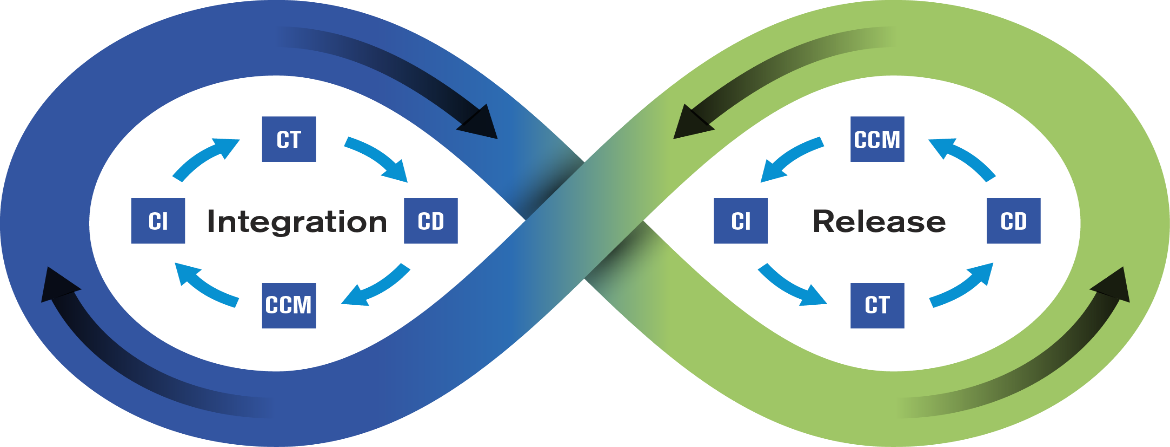
**Role of Continuous Monitoring in DevOps**

DevOps is the blending of tasks performed by a company's application development and systems operations teams. The term DevOps is being used in several ways. In its most broad meaning, DevOps is an operational philosophy that promotes better communication between development and operations as more elements of operations become programmable. In its most narrow interpretation, DevOps describes the part of an organization’s information technology (IT) team that creates and maintains infrastructure. The term may also be used to describe a team of software engineers who strategically look at the entire software delivery chain, overseeing shared services and championing the use of new development tools and best practices.

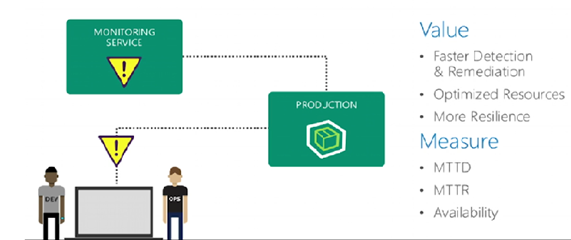
Continuous monitoring offers enterprise-class, easy-to-use reporting that helps developers and testers understand the performance and availability of their application, even before it is deployed to operations. The early feedback provided by continuous monitoring is vital for lowering the cost of errors and change, and helps steer projects toward successful completion.

In Production, the Ops team manages and ensures that the application is performing as desired and the environment is stable via Continuous Monitoring. While the Ops teams have their own tools to monitor their environments and systems, DevOps principles suggest that they also monitor the applications. They need to ensure that the applications are performing at optimal levels – down to levels lower than system monitoring tools would allow. This requires that Ops teams use tools that can monitor application performance and issues. It may also require that they work with Dev to build self-monitoring or analytics gathering capabilities right into the applications being built. This would allow for true end-to-end monitoring continuously.



**Primary Categories of Monitoring Targets-** Here is an attempt to classify the monitoring requirements in production. The requirements are generic and are applicable for any system hosted in-house, in a data centre or in the cloud. A systematic and proactive plan for testing code changes and monitoring production would help to prevent incidents in production.

**Application Performance Monitoring** (APM) provides code-level visibility that enables quick identification of performance issues as well as rapid remediation. At a high level, APM involves some form of an agent in your application in production. It logs all kind of information that application is running in the environment. The more fully featured APM tools provide trending reports, notifications and alerts like hey- there is a problem, performance is slow, something is going wrong etc.



And also give you which kind of data you need to help that resolve your issues. This is naturally can give you elaborated idea, detecting and mediating things. APM helps to determine much faster the delay at code level if any performance problem is there. We can notice improvement in terms of Mean Time To Detection (MTTD) and Mean Time To Repair (MTTR).

The implementation of application level monitoring could be simplified by building hooks or API endpoints in the application. Monitoring is usually an after-thought and the requirement of such instrumentation is overlooked during the design phase of an application. The participation of DevOps team in the design reviews improves operability of a system. Planning for application level monitoring in production is one area where DevOps can provide inputs.

**Monitoring Infrastructure**

The infrastructure that hosts an application environment would be made up of multiple components: servers, storage devices, load balancer etc. Checking the health of these devices is the most basic requirement of monitoring. The popular monitoring platforms support this feature out-of-the-box. Very little customization is required except for setting up right thresholds on those metrics for alerting.

**Monitoring Server**

Monitoring Server operates at the infrastructure level, allowing reliability engineers to track server health in cloud, physical, and hybrid environments. These tools show capacity, memory, and CPU status for each server so that problems can be addressed early—ideally, before they impact application performance.

**Monitoring Platform**

An application would typically be built using multiple third-party tools such as databases, both RDBMS and NoSQL data repositories; full-text search engines, Big Data platforms, messaging systems, memory object caching systems, BI and reporting tools. Checking the health of these application components is important too. Most of these tools provide some interface, mainly via REST API, that can be leveraged to implement plugins on the main monitoring platform.

**Monitoring Business**

The applications run in production to meet certain business goals. You can have an application that runs flawlessly on a healthy infrastructure but still the business might not be meeting its goals. It is important to provide that feedback to the business at the earliest to take corrective actions which might trigger enhancements of the application features and/or require the way business is run using the application. These efforts should only complement the more complex BI based data analysis methods that could provide deeper insights into the state of the business. The business level monitoring can be based on transactional data readily available in the data repositories and the data aggregates generated by the BI systems. Both application and business level monitoring are company specific, and, plugins have to be developed for such monitoring requirements. Implementing some framework to access standard sources of information such as databases and REST APIs from the monitoring platform could minimize the requirement of building plugins from scratch every time.

**Last-Mile Monitoring**

A monitoring platform deployed in the same cloud or data centre environment where the applications also run cannot check on the end-user experience. To address that gap, there are several SaaS products are in the market. These services are backed up by actual infrastructure to monitor the applications in specific geographical locations. Alerts are setup on these tools to notify the site reliability team if the application is not accessible externally or if there are performance issues with the application.

**Monitoring the Monitoring**

It is important to make sure that the monitoring infrastructure itself is up and running. Disabling alerting during a deployment and forgetting about enabling it later is one of the common oversights. Such missteps are hard to monitor and only improvement in deployment process could address such issues.

***Pinging hosts***- If there are multiple instances of the monitoring application running, or if there is a stand-by node, then cross checks can be implemented to verify the availability of hosts used for monitoring.

***Health-check for monitoring***- Checking on the availability of monitoring UI and activity in monitoring application’s log files would ensure that monitoring system itself is fully functional and it continues to watch for issues in production environment.

**Conclusion**

Monitoring efforts are normally in response to issues happen in production. A systematic approach to rolling out monitoring can minimize the reactive ways it is normally associated with. Continuous monitoring adds to better user experience, and, it avoids costly data reprocessing and rollback in production.