Shreya Bose, Technical Content Writer at BrowserStack - February 8, 2023

What is Continuous Monitoring in DevOps?

Fundamentally, Continuous Monitoring (CM), sometimes called Continuous Control Monitoring (CCM), is an automated process by which DevOps personnel can observe and detect compliance issues and security threats during each phase of the DevOps pipeline. Outside DevOps, the process may be expanded to do the same for any segment of the IT infrastructure in question. It helps teams or organizations monitor, detect, study key relevant metrics, and find ways to resolve said issues in real time.

Continuous Monitoring comes in at the end of the DevOps pipeline. Once the software is released into production, Continuous Monitoring will notify dev and QA teams in the event of specific issues arising in the prod environment. It provides feedback on what is going wrong, which allows the relevant people to work on necessary fixes as soon as possible.

Continuous Monitoring basically assists IT organizations, DevOps teams in particular, with procuring real-time data from public and hybrid environments. This is especially helpful with implementing and fortifying various security measures – incident response, threat assessment, computers, and database forensics, and root cause analysis. It also helps provide general feedback on the overall health of the IT setup, including offsite networks and deployed software.

Read More: DevOps Testing Strategy

Table of Contents

- Enhance transparency and visibility of IT and network operations, especially those that can trigger a security breach, and resolve it with a well-timed alert system.

Types of Continuous Monitoring

- **Infrastructure Monitoring**: Monitors and manages the IT infrastructure required to deliver products and services. This includes data centers, networks, hardware, software, servers, storage, and the like. Infrastructure Monitoring collates and examines data from the IT ecosystem to improve product performance as far as possible.
- **Application Monitoring**: Monitors the performance of released software based on metrics like uptime, transaction time and volume, system responses, API responses, and general stability of the back-end and front-end.
- **Network Monitoring:** Monitors and tracks network activity, including the status and functioning of firewalls, routers, switches, servers, Virtual Machines, etc. Network Monitoring detects possible and present issues and alerts the relevant personnel. Its primary goal is to prevent network downtime and crashes.

Also Read: How to optimize test cases for Continuous Integration

Benefits of Continuous Monitoring

- **Better Network Visibility and Transparency:** CM offers DevOps teams clarity on the state of the IT infrastructure by automatically collecting and analyzing data to reflect possible outages and important trends.
- **Facilitates Rapid Responses:** A primary aspect of CM is implementing an alert system that immediately notifies the right people the minute an IT incident emerges. This enables timely response to security threats or functional stop-gaps, minimizing damage and allowing faster restoration of the system to optimal operational levels.
- **Minimizes System Downtime:** Consistent system monitoring and quick, necessary alerts help maintain system uptime by raising the alarm when there is a service outage or any application performance issues.
- **Assists with Healthy Business Performance:** Reduction in system downtime also minimizes negative impact on customer experience, thus safeguarding the organization against losses in revenue or credibility. As mentioned before, Continuous Monitoring tools can also be used to track user reactions to software updates, which is useful for several teams – development, QA, sales, marketing, customer service, etc.

Risk Management and Continuous Monitoring

There are numerous tools for every stage of Continuous Monitoring in DevOps. However, before selecting tools, organizations, and DevOps teams must conduct adequate risk assessment and formulate a risk management plan. Developers can only implement an appropriate CM system after a thorough evaluation of compliance systems, governance, and risk factors. These tend to be quite different between organizations depending on their nature; e.g., a private company will have a different view of risk than a government organization.

To facilitate understanding of these metrics, consider asking the following question when looking for tools to implement CM:

- What is the extent of risk that the organization can withstand and recover from?
- What are the parameters by which to calculate risk?
- For each parameter, is it possible to assign values that denote the highest potential risk?
- What is the level of confidentiality required by the data collected and generated by the organization?
- What are the consequences of security breaches, hardware, or software failure?

Read More: Continous Testing in DevOps: A Detailed Guide

Best Practices for Continuous Monitoring in DevOps

Decide what to monitor: Different organizations need to monitor different aspects of their IT landscape. Primarily, the targets are categorized into the following:

1. Server status and health
2. Application performance log
3. System vulnerabilities

As far as possible, try to track parameters belonging to each category.

- Server & System Uptime
- Response Time to Errors
- Storage
- Database Health
- Storage
- Security
- User permissions
- Network switches
- Process level usage
- Relevant performance trends

Network Monitoring: Tool must monitor:

- Latency
- Multiple port level metrics
- Server bandwidth
- CPU use of hosts
- Network packets flow

Application Monitoring: Tool must monitor:

- availability
- error rate
- throughput
- user response time
- pages with low load speed

By now, the article has revealed that Continuous Monitoring, though essential, is a time and resource-intensive process. The CM system will notify when errors occur in released software, which adds to QA and developers' effort. After every product release, devs and QAs have to move on to other projects, which means that the error they are notified of adds to the strain of their daily operations.

To ensure that the CM system is not going on overdrive, release software that has been thoroughly tested on real browsers and devices. Emulators and simulators simply do not offer the real user conditions that software must run within, making the results of any tests run on them inaccurate. Consider testing websites and apps on a real device cloud, preferably one that offers the latest devices, browsers, and OS versions. This applies to both manual testing and automation testing.

BrowserStack's real device cloud *provides 2000+ real browsers and devices for instant, on-demand testing. It also provides a cloud Selenium grid for automated testing, which can be accelerated by 10X with parallel testing. The cloud also provides integrations with popular CI/CD tools such as Jira, Jenkins, TeamCity, Travis CI, and much more. Additionally, there are in-built debugging tools that let testers identify and resolve bugs immediately.*

Try BrowserStack Now

Continuous Monitoring intends to provide organizations with almost immediate feedback and insight into performance and interactions across servers, networks, and cloud environments, which is pivotal in enhancing operational, security, and business performance. It should be seen as an integral part of every DevOps pipeline, crucial to achieving efficiency, scalability, and better-quality product.

Follow-up Read: How to implement a Continuous Testing strategy for DevOps?

CI CD Tools DevOps

Was this post useful? Yes, Thanks

Featured Articles

Continuous Integration in Agile

How to implement a Continuous Testing strategy for DevOps?

Continuous Integration in Agile

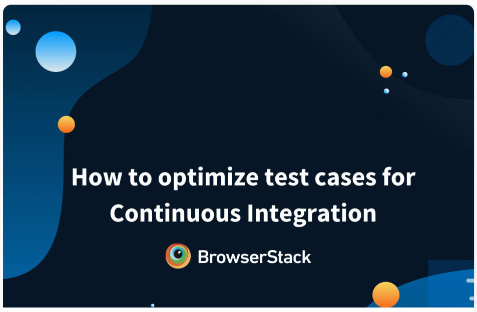
What is Continuous Integration in Agile? Why is it beneficial in Agile processes? What are the steps...

Learn More

How to implement a Continuous Testing strategy for DevOps?

Best practices to implement a Continuous Testing strategy in an Agile development team. Run Continuo...

Learn More



cases for Continuous Integration

PRODUCTS

Live
Automate
Automate TurboScale Beta
Percy
App Live
App Automate
App Percy New
Test Management New
Test Observability New
Accessibility Testing New
Accessibility Automation Beta
Low Code Automation Beta
Nightwatch.js
Enterprise

TOOLS

SpeedLab
Screenshots
Responsive

PLATFORM

Browsers & Devices
Data Centers
Device Features
Security

SOLUTIONS

Test on iPhone
Test on iPad
Test on Galaxy
Test In IE
Android Testing
iOS Testing
Cross Browser Testing
Emulators & Simulators
Selenium
Cypress
Android Emulators
Visual Testing

RESOURCES

Test on Right Devices
Support
Status
Release Notes
Case Studies
Blog
Events
Test University Beta
Champions
Mobile Emulators
Guide
Responsive Design

COMPANY

About Us
Customers
Careers We're hiring!
Open Source
Partners
Press

SOCIAL