WHAT IS SECURITY AS CODE?

Businesses have quickly and sophisticatedly embraced cloud technology to super-serve customers and perform better. With this radical change emerges an increased need for DevOps professionals and codified security to enhance their processes.

DevOps as a methodology allows for faster development that is always occurring, evolving in stages. It's designed to help development teams collaborate with operations by working toward the same goals, instead of working against each other to different ends, resulting in streamlined processes.

DevOps developers are constantly injecting new code into existing infrastructure, continuing to simplify the development lifecycle. Naturally, security comes into question when DevOps developers are forever integrating new code into existing systems. This is where Security As Code comes in. Keep reading to learn more about how Security As Code protects your DevOps organization.

What is Security as Code?

Security as Code is a toolset of resources that help DevOps professionals secure and protect the software development lifecycle (SDLC) throughout the process of development. The practice is interesting because of the popularity of DevOps in enterprise business. Security as Code represents the next evolution of DevOps--an era in development where security is baked into the development process and businesses and their customers operate more safely.

To begin adding security as code to your digital infrastructure, you must carefully examine and map the current process of how infrastructure changes are made, and then look for vulnerabilities that can be minimized by code. Hence the name, Security as Code. Gates, security checks and tests are coded into the infrastructure at vulnerable points throughout the process. A major benefit of this approach is that it doesn't delay DevOps development to implement security.

When testing and security is injected into the development lifecycle, via things like automated testing on new code, developers can make fixes, as they occur, ensuring the development process is not stalled. Some Security as Code best practices include:

- Automating feedback loops
- Automating scans and security testing
- Executing script tests
- Implementing monitoring functions
- Performing routine security policy checks

Understanding the Software Development Lifecycle

The software development lifecycle is the series of systematic processes that standardize how software developed. In today's cloud-driven economy, enterprise businesses can't stop at digital transformation of their basic architecture with pre-made solutions. They must begin to develop their own proprietary infrastructure to best serve employees and customers in order to remain competitive.

That means enterprise business owners and stakeholders must be aware of the software development lifecycle as a backbone of company operations. In this kind of business environment, a DevOps approach is almost required for efficiency and effectiveness.

The software development lifecycle consists of six common phases, defined below:

- Requirement analysis
- Feasibility study
- Design and architecture
- Development
- Testing
- Deployment

Requirement analysis

In this phase, leadership and stakeholders from technology and business departments come together to systematically evaluate software engineering problems, determining the requirements of the solution. This is often part of a larger discussion about SDLC process strategy.

Feasibility study

Next, a feasibility study is done to test the risk and reward of each requirement. During this phase, requirements undergo scrutiny to determine if it's economically feasible, legally compliant, operationally possible, technically available and viable from a timeline standpoint.

Design and architecture

In the design phase, architecture is documented by lead developers and given to stakeholders for review. Once a design schematic and associated configurations are agreed upon, development can begin

Development

Then, a team of professionals, including frontend developers and backend developers, implement the architecture in sets of assigned features and coding activities. During this process developers often work independently in a development environment, perfecting the feature before merging it with existing features in the team codebase.

Testing

When coding activities are complete, the features enter testing, where a team of developers evaluates the code for bugs, effectiveness and optimization. The testing plan is designed in advance based on the requirements of the project determined in the first phase. Testers also work to improve the codebase architecture alongside the development teams

Deployment

When features move from the development environment to the real one where they are used by customers, that's deployment. The goal of DevOps is for new features to be quickly, cleanly and routinely deployed for the best customer experience.

The Impact of Continuous Integration

Since DevOps is centered around continuous delivery, continuous integration is necessary to support the process, ensuring it goes smoothly. Continuous integration refers to the implementation of code in the development process to optimize it. One example is injecting code that automatically screens, tests and initiates a feedback loop with the developer on newly committed code. This is Security as Code.

Securing the SDLC with Security as Code

When you secure your SDLC with Security as Code, you are making an enterprise cultural shift that prioritizes security with requirements, encouraging further opportunities to automate security into the process. Here are **some best practices** for ensuring Security as Code protects your SDLC:

- **Consider security requirements early and codify them.** Security as Code is a process that requires DevOps team leadership and stakeholders to plan for security, systematically, with a codified set of practical measures that automate or must be deployed throughout the SDLC.
- **Create user stories and perform security assessment.** User stories are an agile practice, encouraging developers to review feature requirements from an end-user perspective. This helps to ensure no important features are unintentionally ignored. The same must be done for security in order to have a SecDevOps environment.
- **Ensure code is ready for continuous delivery.** Because security is automated in the

development process and development infrastructure, security checks and tests happen early and often, making software projects ready to fulfill the requirements of continuous delivery.

- **Automate routine compliance checks.** In addition to automating security scans, it's also important to implement compliance checks that ensure your development is operating to legal standards while following industry best practices.
- **Put finishing touches on security in the test environment.** When the product moves to the testing team, much of the security and compliance has already been handled via automation throughout the development process. Development teams fine-tune and optimize application security and compliance using tools and resources like the ones offered by BMC.