WHITE PAPER

Mphasis stelligent
The Next Applied

Intro to Continuous Security on AWS

CONTENTS

1. Abstract

If you are planning on deploying resources on Amazon Web Services (AWS), or if you have already started on the journey and are looking for opportunities to improve the security of your cloud environments, the tools and patterns identified in this paper will accelerate your integration of security into the development process.

As a move to the cloud increases the velocity and agility with which you can deploy infrastructure, it places an ever-increasing burden on your security organization to validate the configuration of your cloud resources at an accelerated pace. Often, organizations are left to choose between giving up the agility and pace of innovation afforded by a move to the cloud, or to accept greater risk in their cloud deployments.

However, through the practice of Continuous Security, we are able to ensure that security standards are represented as code in an unambiguous way, and validate the security of your cloud infrastructure during the development process in a way that scales across an organization.

2. Continuous Security

Continuous Security is the addressing of security concerns and testing in the Continuous Delivery pipeline, and is as much a part of Continuous Delivery as operations, testing, or security is a part of the DevOps culture. This article talks about ways of integrating security testing/validation of both software and infrastructure into the Continuous Delivery pipeline.

DevOps is a culture that emphasizes the collaboration and communication of all IT roles necessary to develop, deliver and support software and infrastructure through

automation while eliminating silos and delays caused by organizational boundaries. Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.

Continuous Delivery is used to build, test and deploy code rapidly, ideally with the automated ability for any given change to flow into production with the click of a button. It relies on the complete automation of the deployment process, such that bottlenecks, handoffs, and human intervention are eliminated. You are capable of deploying any code change immediately to production, and are not shackled by an onerous and manual release process. Only business drivers influence the decision to deploy; uncertainty over the coordination and execution of a labor-intensive deployment never enters the equation.

Continuous Security Principles

- **Automate everything, including security!**
- **If security is difficult or painful, do it more often (and it always is).**
- **Keep everything in source control, including your security posture and tests.**
- **Done means secured.**
- **Build security in.**
- **Everybody has responsibility for security.**
- **Improve security continuously.**

Automated testing is fundamental to the Continuous Delivery process. Testing in the pipeline both eliminates delays from manual verification of code, as well as ensuring consistent verification of functionality. Various types of testing are performed in all of the stages of the pipeline in order to provide feedback to the developer as rapidly as possible. A typical pipeline is depicted in Figure 5.

3.
What does all this have to do with Security?

In the cloud, we talk often about infrastructure-as-code, spinning up compute resources on demand. But we need to go further and treat everything as code. Sure, we create instances from code, but we also codify networks, security, monitoring, operations and feedback mechanisms. The same Continuous Delivery process that allows you to test, verify and deploy application changes applies equally to networking, security monitoring, etc., now that those functions are represented as code.

And while Continuous Delivery is typically looked upon as an accelerator for developers, allowing business value to be deliver much more rapidly, an interesting transformation takes place when security is made a core focus of the pipeline. Change occurs more rapidly, but at the same time governance and security are drastically improved. This is due to the iterative testing of incremental change and the capturing of security requirements as code, rather than being manually, and often inconsistently, applied.

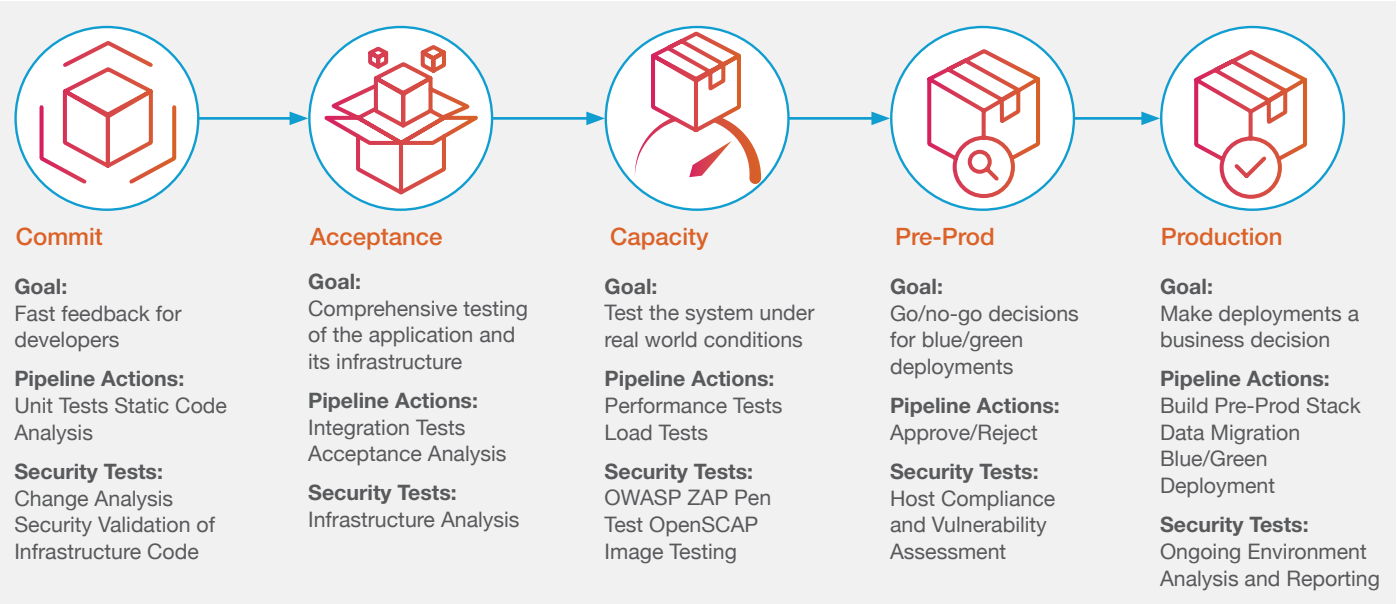
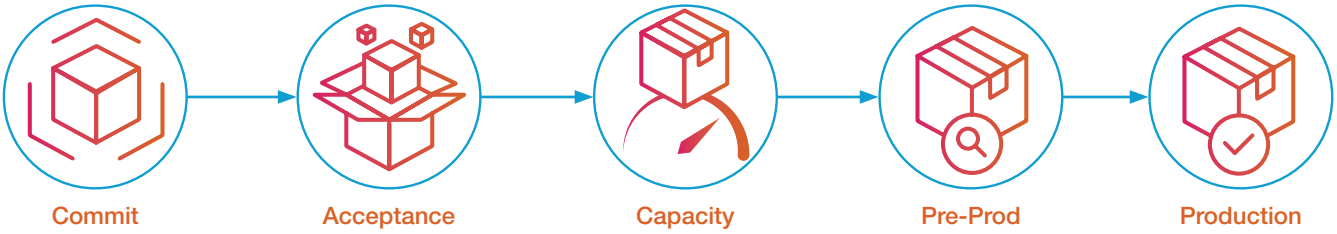


Figure 1: Continuous Delivery Process

4.
Stages in Continuous Security



Commit Stage

What might it look like to validate your security posture in the Continuous Delivery pipeline? Referring back to the pipeline graphic (figure 1), the first stage is Commit. This stage provides very rapid feedback to the developer, as the code is analyzed before building. For security, this means making a decision about the resources and changes that would be triggered by executing the code, before actually triggering resource creation/modification. Not only does this provide rapid feedback (literally seconds), but has powerful security implications in that infrastructure code that would otherwise expose a vulnerability can be vetted and prevented from being provisioned.

In order to bring this type of commit stage testing to CloudFormation, we have created an open source tool called Stelligent cfn_nag, which builds a model of what your templates would do, and analyzes it against a set of user-controllable rules. Only if the template passes all tests, will it be subsequently executed.

Security Static Analysis of AWS CloudFormation

Security static analysis builds a model of templates in order to verify compliance with best practices and organizational standards. This can be a powerful tool to stop bad things before they happen. A security organization can define their policy in code and have all development efforts unambiguously verify against that standard without manual intervention.

The Stelligent cfn_nag tool inspects the JSON of a Cloud Formation template before convergence to find patterns that may indicate:

- Overly permissive IAM policies
- Overly permissive security groups
- Disabled access logs
- Disabled server-side encryption

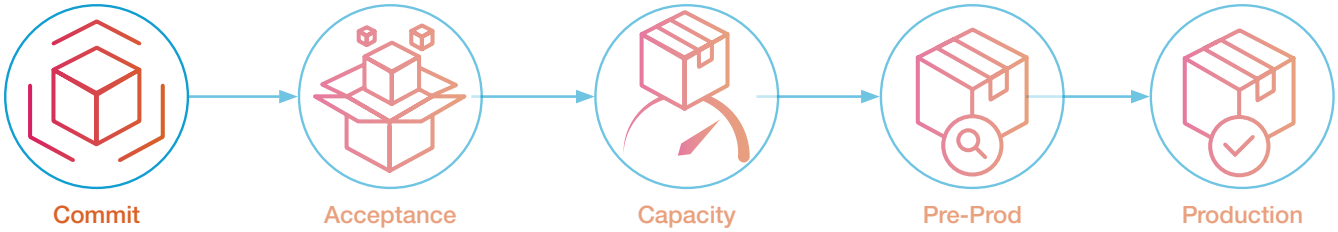


Figure 2: Continuous Delivery Process - Commit Stage

Goal: *Fast feedback for developers*

Acceptance Stage

Once code passes the Commit stage, it is actually built and acceptance testing is performed. In terms of infrastructure code, this means AWS resources are created. Whenever you create AWS resources, you are creating or modifying your security posture, whether you want to or not. AWS has released a powerful tool for analyzing resources against security considerations, called AWS Config Rules, perfect for resource verification in the Acceptance stage. Implementing AWS Config Rules in the Continuous Delivery pipeline presents some interesting challenges, which will be covered in detail in an upcoming article.

Testing Infrastructure Changes

Problems to solve:

- Prevent infrastructure changes that violate company security policies.
- Need the ability to codify security rules and get notifications when violation occur.
- Ability to execute on-demand compliance testing.

Stelligent config-rule-status:

- Config-rule-status is an open source tool that enables continuous monitoring and on-demand testing of security compliance for infrastructure running on AWS.
- Config-rule-status provides a CLI for deploying and managing resources on AWS. It will setup the AWS Config service to monitor the infrastructure and create AWS Config Rules to enable continuous compliance testing and reporting. The compliance rules are implemented as Lambda functions. Several rules are included by default and new rules can be easily added. Config-rule-status enables on-demand compliance testing via a Tester Lambda function that aggregates the compliance status of all AWS Config Rules and outputs a JSON response that contains the status for each rule and an overall status of PASS or FAIL.
- Config-rule-status can be run from a local computer, but is designed to run on a build server as part of a continuous delivery pipeline. The Tester Lambda function should be used during the Acceptance stage of the pipeline to ensure that non-compliant infrastructure changes do not get deployed to production. The Tester can be invoked from the build server using the included CLI, or it can be invoked from anywhere by using the AWS CLI.

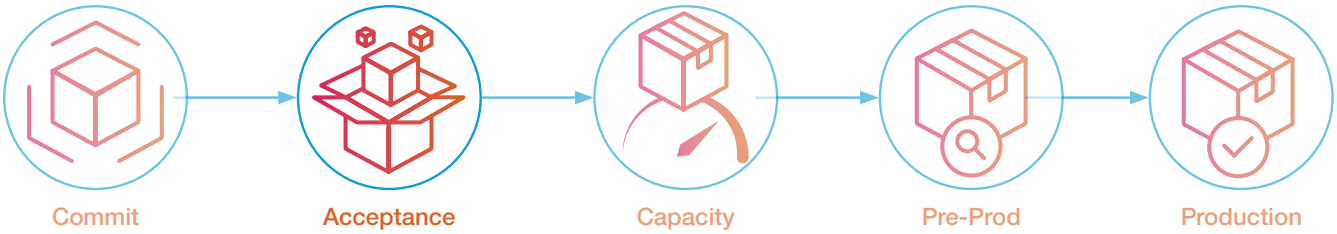


Figure 3: Continuous Delivery Process - Acceptance Stage

Goal: *Comprehensive testing of the application and its infrastructure*

Capacity Stage

After making it past the Acceptance stage gauntlet, We move on to the Capacity stage. Capacity is where we build and validate an environment capable of going into production. Since you anticipate this environment could go live, you want some in-depth security analysis here. One of our favorite security tools for evaluating web applications is the OWASP Zed Attack Proxy (ZAP). ZAP is typically used manually through a GUI, which doesn't work well with our "automate everything" mindset. Utilizing Behave and some custom Python, we are able to parse the JSON output of ZAP and assess the results via an english-like DSL.

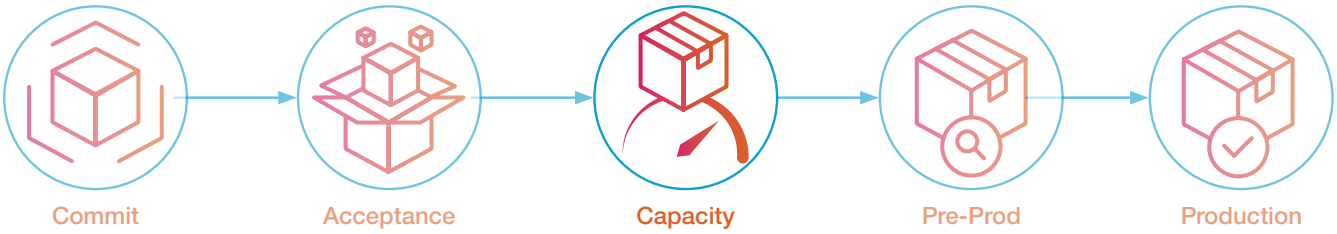


Figure 4: Continuous Delivery Process - Capacity Stage

Goal: *Test the system under real world conditions*

Pre-Production and Production Stages

The last two stages are Exploratory and Production. Production is the stage that promotes a candidate environment that has passed all previous stages into production. Exploratory is really an out-of-band stage that provides a self-service model for creating clones of the production environment for experimentation. We'll focus on Production stage and leave Exploratory for another day.

Production is a very interesting stage. In typical software development CD pipeline, Production is the end of the line. Hit this stage and you are done. But not so much with security. While your application is deployed and immutable, your security posture is still subject to change. You may have firm control of

your CloudFormation templates through the pipeline, but resources can be added, changed and deleted independent of your templates.

For this reason, we strongly recommend that the Production stage provide periodic, ongoing testing of the security of the live environment. Ensuring that an environment remains in conformance with your rules is compliance, and can take the form of internal compliance with your own best practices, or compliance with industry/governmental requirements.

An interesting tool we are currently integrating into the CD pipeline is Inspector, an AWS security assessment service. This tool is still in preview mode, so more from us on integrating it as it hits general availability. We are also experimenting with integrating other open source tools into our pipeline, such as OpenSCAP.

Continuous Delivery is a discipline that provides rapid feedback and consistent, repeatable validation of software and infrastructure. There are a wealth of tools that provide valuable insight into your security postures. Leveraging these tools within your CD pipeline helps promote security to a first-class citizen of the development process. Please join us for the ensuing articles that dive much deeper into the Continuous Security pipeline, and how to integrate security testing the Stelligent way.

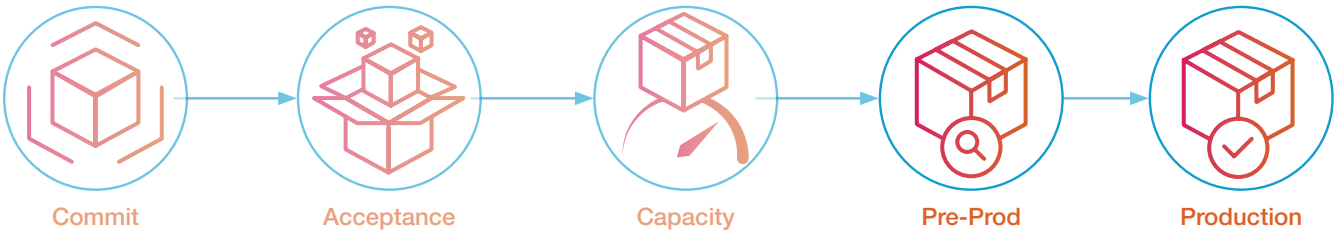


Figure 5: Continuous Delivery Process - Pre-Production & Production Stages

Goal: *Go/no-go decision for blue/green deployment*

5.
Key Takeaways

- Infrastructure IS code, treat it as such. Applying modern development techniques such as TDD and Continuous Delivery yields immense value.
- Infrastructure is part of the solution in application development now. Its development should be integrated into the application development process, treating the solution as an integrated entity.
- From within development team, CD reduces cycle time for releases and improves confidence in released code (including infrastructure code).
- From outside, it allows security/governance/compliance to inject best practices as automated gates in the delivery process without introducing delays for review and approval. This allows for control at scale without grinding to a halt.

6.
Conclusion

In this paper we have covered some of the basic techniques of applying security as code through the Continuous Delivery pipeline; pulling security into the DevOps culture, where it can provide immense value mitigating risk while enabling development teams to build security in.

For more information, please drop by our Github repositories (https://github.com/stelligent), our blog site topic on Continuous Security (https://stelligent.com/category/continuous-security/), or contact us at info@stelligent.com.

To stay updated on DevOps best practices visit www.stelligent.com or contact info@stelligent.com

ABOUT MPHASIS STELLIGENT

Mphasis Stelligent, a professional services and consulting firm with deep expertise in DevOps automation services on Amazon Web Services (AWS), enables security-conscious enterprises to focus on developing software users love by leveraging automation on AWS. Our goal is to work closely with customers to develop fundamentally secure infrastructure automation code, deployment pipelines, and feedback mechanisms for faster, more consistent software and infrastructure deployments. By embedding with our customer's engineering teams, we empower customers through education and knowledge transfer of our expertise while developing the automation to make them self-sufficient on AWS. As a Premier AWS Consulting Partner, AWS Public Sector Partner, and AWS DevOps and Financial Services Competency holder, we use our demonstrated expertise to help customers benefit from continuous AWS innovation.

For more information, contact us at: info@stelligent.com

11710 Plaza America Drive
Suite 2000
Reston, VA 20190-4743
Tel.: +1 888 924 4539

Mphasis stelligent™
The Next Applied