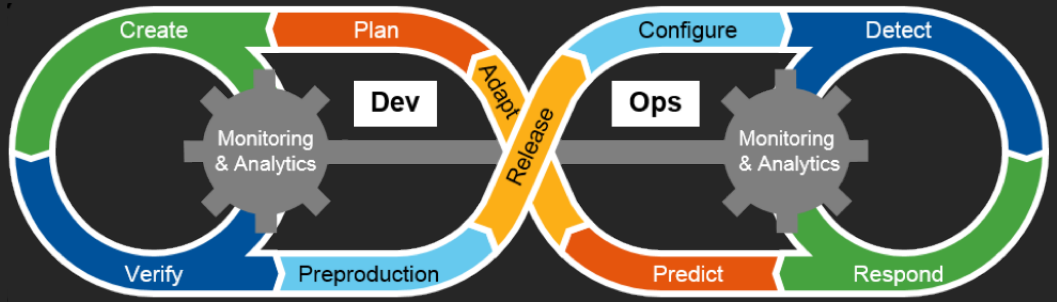
DevSecOps Essentials

Section 1: Introduction

About This Course

About This Course



- **DevSecOps** is new. Gartner published its first report on DevSecOps in May 2016.
- Stakeholders of the Agile and DevOps movements are passionate about change.
- When any movement is new and still evolving, there may be disagreement about particulars.
- This course is designed to give you a broad foundation in DevSecOps terminology, processes, and tools.

DevOps and DevSecOps Are About Culture

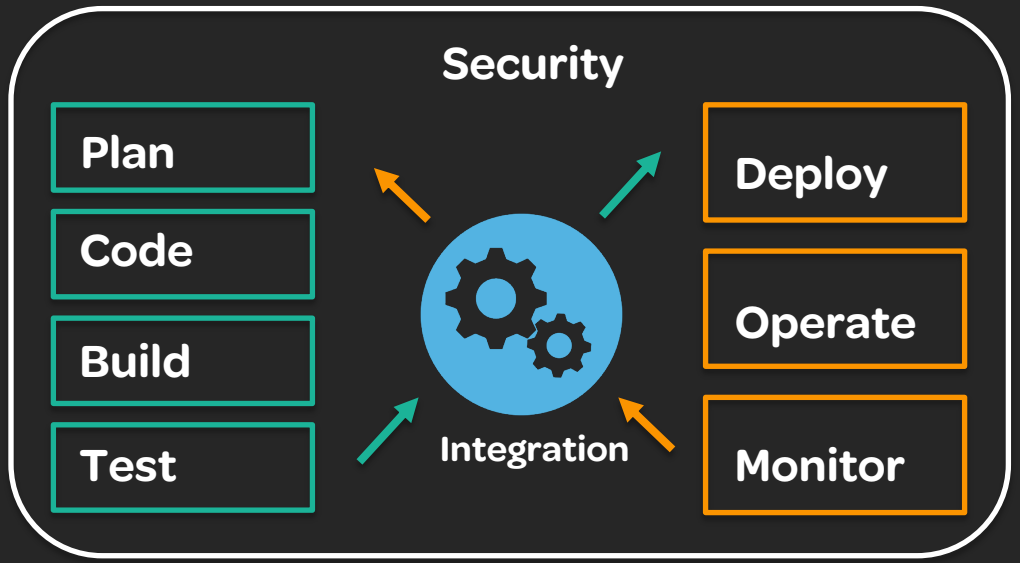
- **Culture** is defined as "the attitudes and behavior of a particular social group."
- Many believe that **culture change** is necessary for large organizations to sustain ongoing process improvement.
- An individual's role and background influence their interest in different aspects of DevSecOps practice.
- Traditional IT departments are fragmented, with Development, Security, and Operations all having different leadership and reporting structures.
- DevSecOps builds on the idea that cross-functional teams must work together and that **everyone is responsible for security**.

DevOps and DevSecOps Are About Automation

- Process experts and coaches often say that DevOps is not about tooling.
- Engineers and technologists grow tired of process quickly and want real-world examples of how processes can be implemented.
- Executive stakeholders measure outcomes and results when making investments in process improvement.
- To achieve the goal of DevOps, quality software must be developed and deployed faster.
- The "**automate everything**" mantra of the DevOps movement is central to DevSecOps as well.

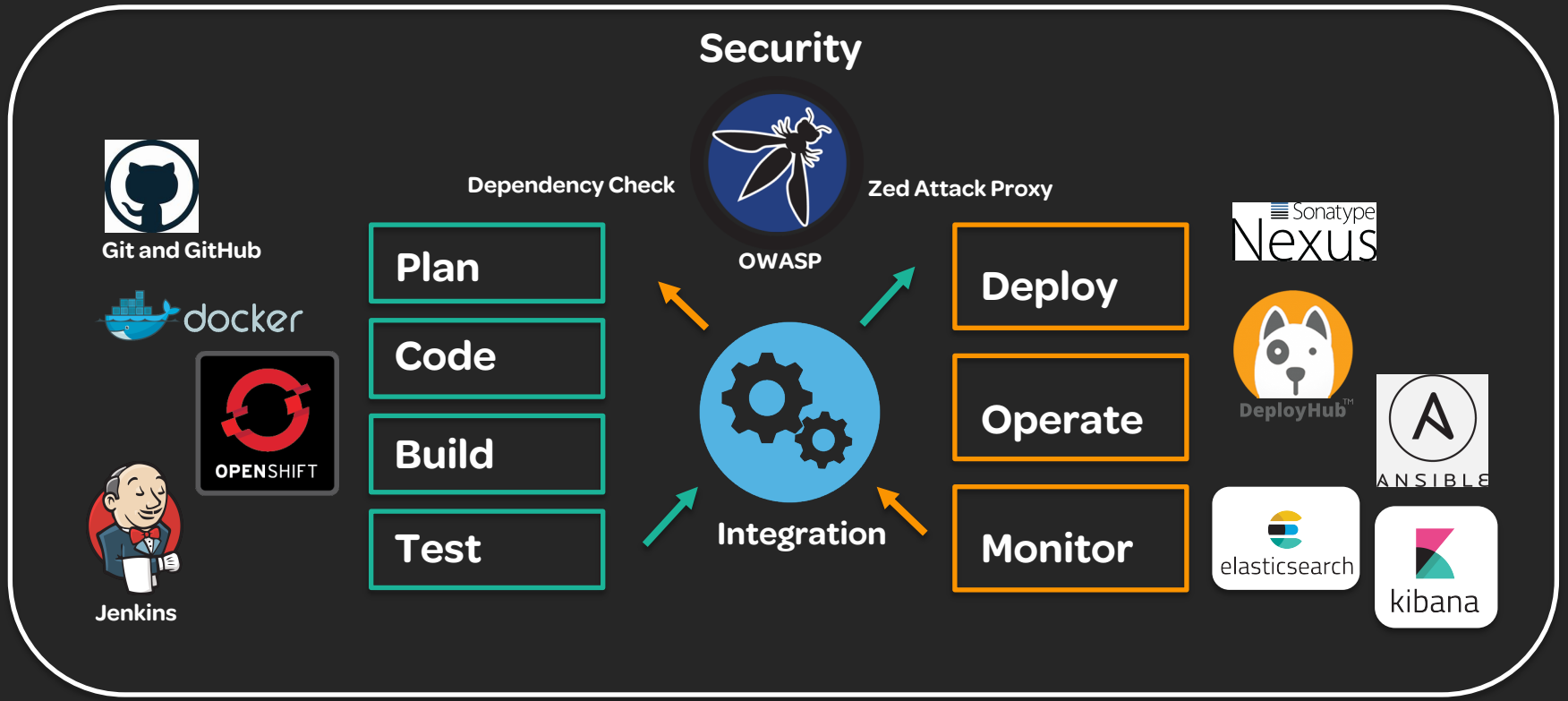
DevSecOps Automation

- Software version control
- Continuous integration
- Continuous testing
- Configuration management and deployment
- Continuous monitoring
- Containerization
- Container orchestration



DevSecOps can be used to enforce proper cybersecurity practices within an automated DevOps CI/CD pipeline.

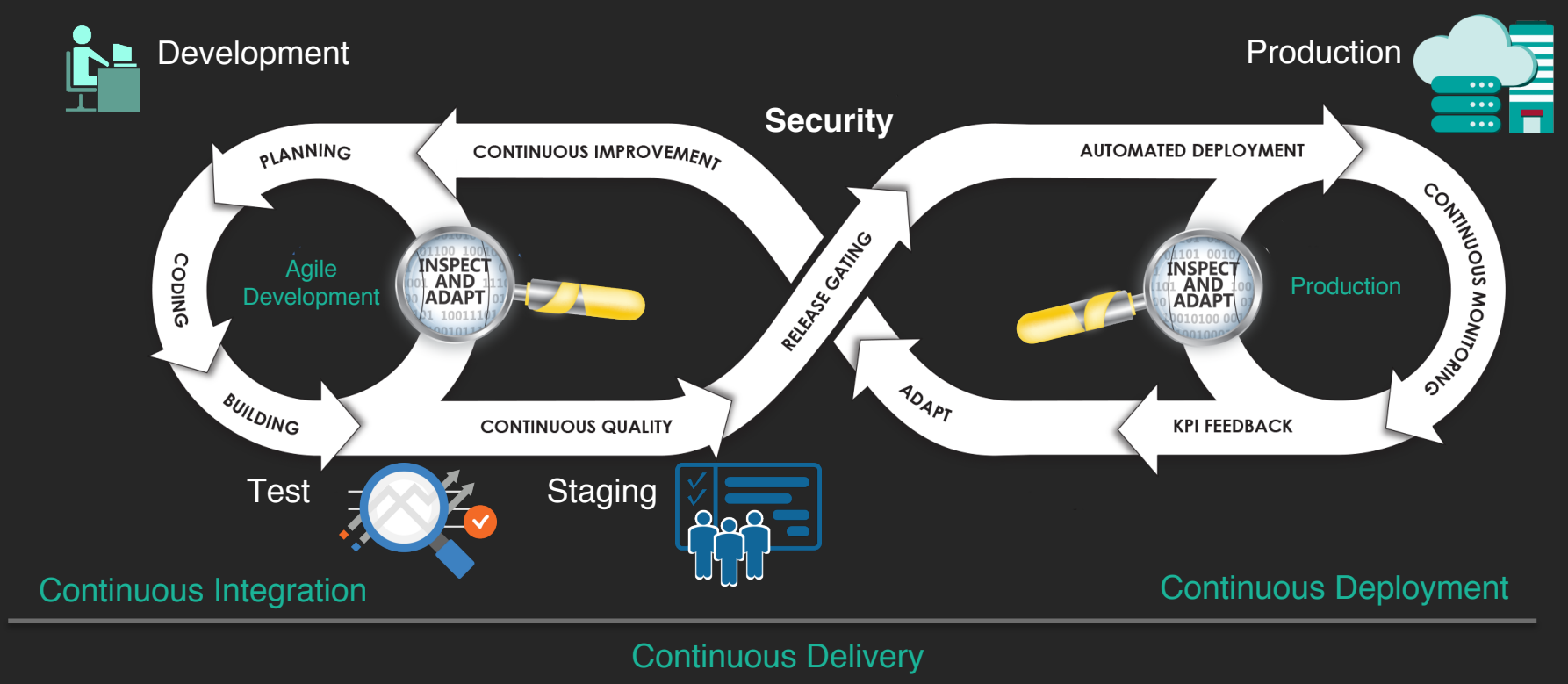
DevSecOps Automation Use Cases



Measuring DevSecOps Success

- Deployment frequency (fast and frequent releases)
- Lead time (code to cash cycle)
- Detection of threats, vulnerabilities, and malware
- Mean time to repair and remediation
- Efficiency of rollback and recovery

Enable business to keep pace with change without compromising security.


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CONTINUOUS MONITORING
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

DevSecOps Essentials

Section 1: Introduction

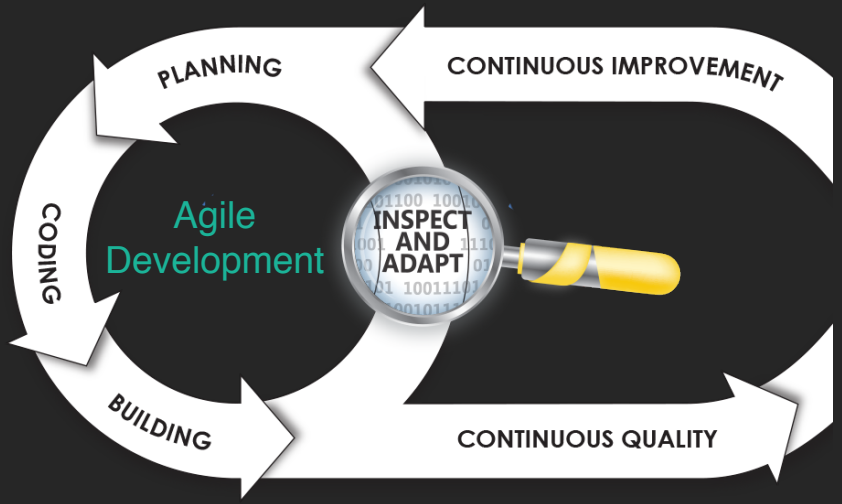
Overview of DevSecOps

1.1 – Overview of DevSecOps

What Is DevSecOps?

- In large corporations, IT departments are generally organized by function.
- In the past, Development, Security and Operations were separate departments.
- Development has traditionally been tasked with writing software applications.
- Operations has traditionally been responsible for maintaining the data center or infrastructure.
- Security has been the role used to enforce regulatory governance and other compliance measures.
- DevSecOps is a portmanteau or mashup of Development, Security and Operations.
- It is a process used to help organizations improve time-to-market and release software faster.

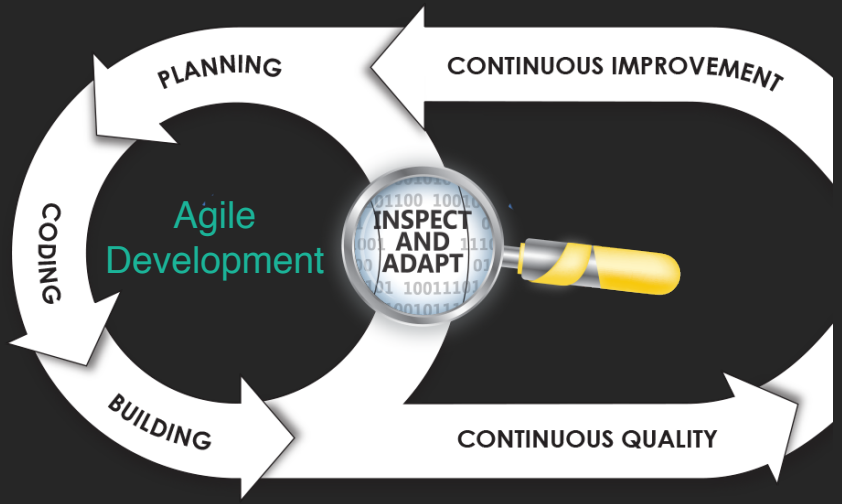
1.1 – Overview of DevSecOps



Agile Development

- As Agile processes have became popular, cross-functional development teams have started using extreme programming and Scrum methodologies to improve productivity.
- By breaking up large releases that could take months into small batches of feature changes that only take days or weeks, teams can drastically shorten time to market.
- Teams also practice "Kaizen" (continuous improvement) to improve systems over time based on feedback from customers.

1.1 – Overview of DevSecOps



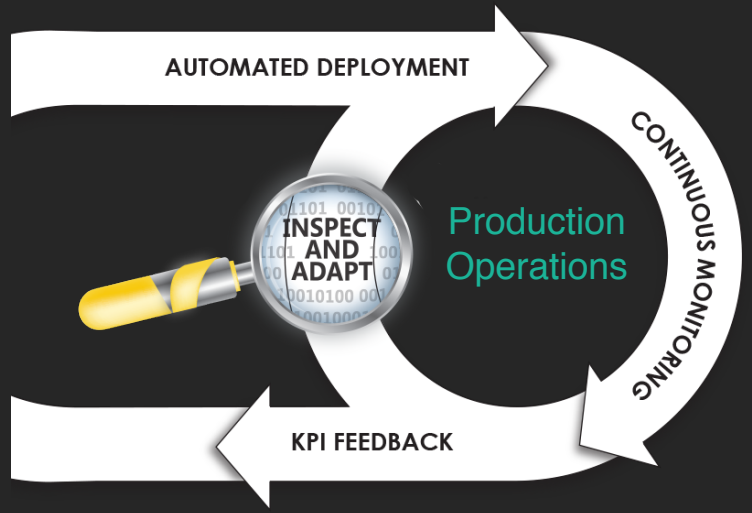
Agile Development

- As Agile teams become more productive, Potentially Shippable Increments build up in staging and Quality Assurance (QA).
- Even though Agile teams can develop applications faster, these products often aren't pushed to production any faster than normal.

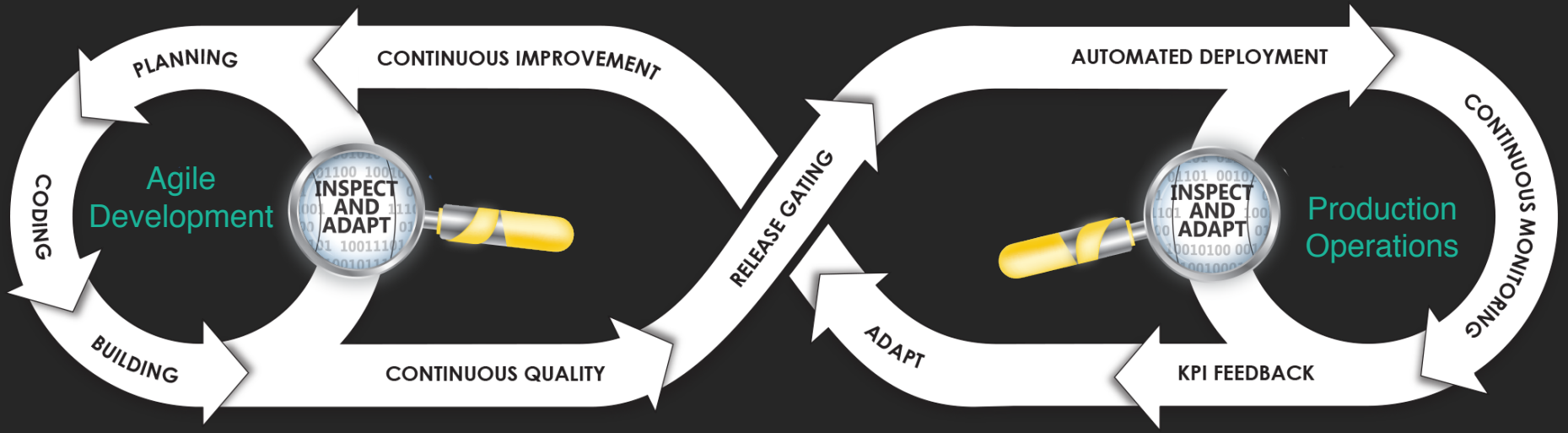
1.1 – Overview of DevSecOps

Operations

- Operations teams can streamline their processes to increase the frequency of releases to production.
- This helps Agile development teams receive feedback from customers in a timely manner and use it to improve applications.
- Operations teams can iterate through release deployments to mimic the iteration cadence of the development team.



1.1 – Overview of DevSecOps

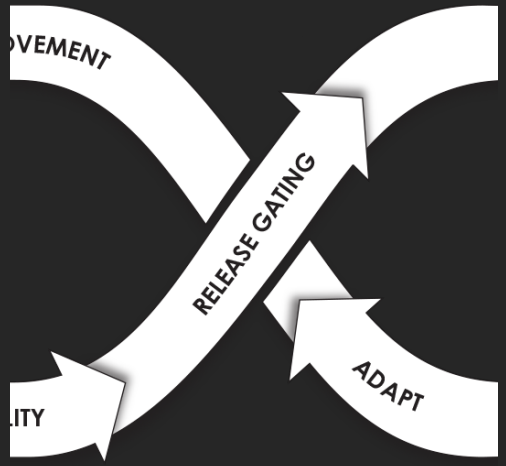


A DevOps CI/CD Pipeline

1.1 — Overview of DevSecOps

Release Management

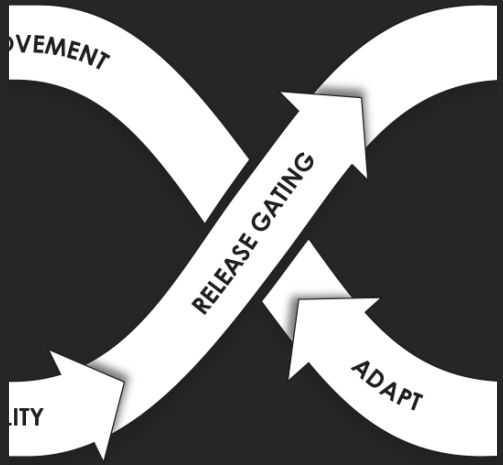
- Release management is the process of coordinating the deployment of finished application increments to production.
- Staging is the interim area where application sub-releases are tested prior to use by customers.
- In highly regulated industries, many approvals are required before code can be released into production.



1.1 – Overview of DevSecOps

Release Gating

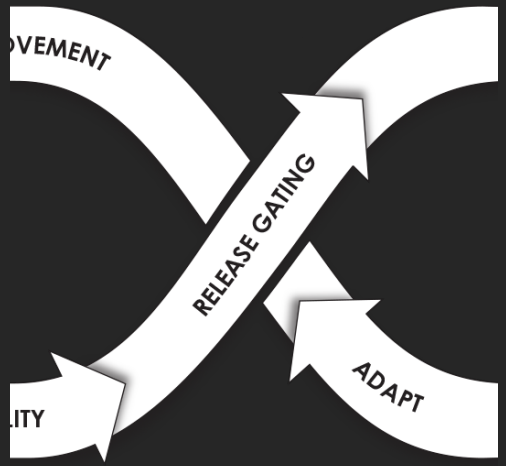
- Governance, risk management, auditing, and compliance groups have had to change the way they approve software releases in order to accommodate Agile's fast and frequent releases.
- Automation is necessary for this rapid approval process.
- Many vendors and open-source projects have sprung up to help automate the Continuous Deployment process.



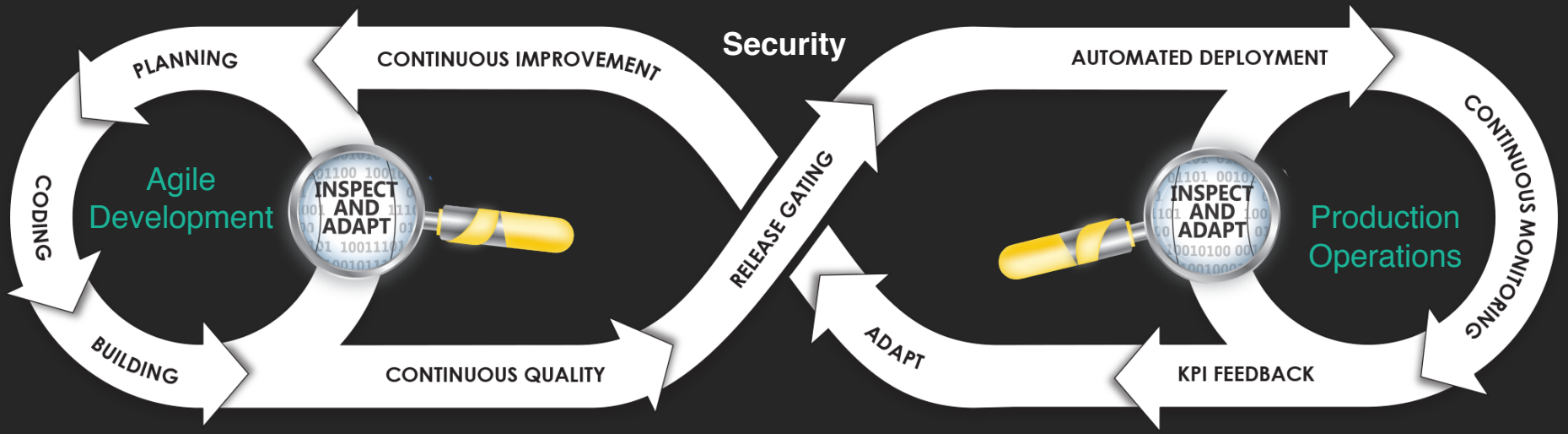
1.1 — Overview of DevSecOps

Security

- To comply with regulations and corporate policy, security departments must verify that software adheres to governance requirements.
- By involving security personnel in the automation process, DevOps practitioners can improve the throughput of security checkpoints and approvals.
- DevSecOps is the evolution of DevOps that emphasizes the importance of security in the software release pipeline.



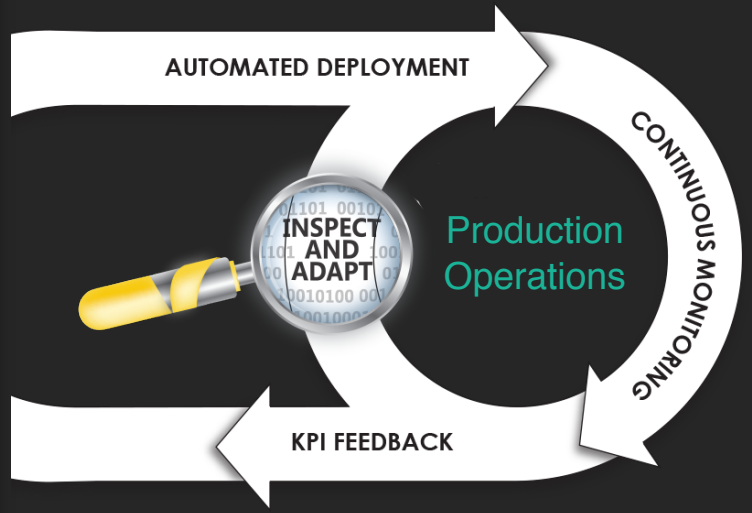
1.1 – Overview of DevSecOps



1.1 – Overview of DevSecOps

Security Monitoring

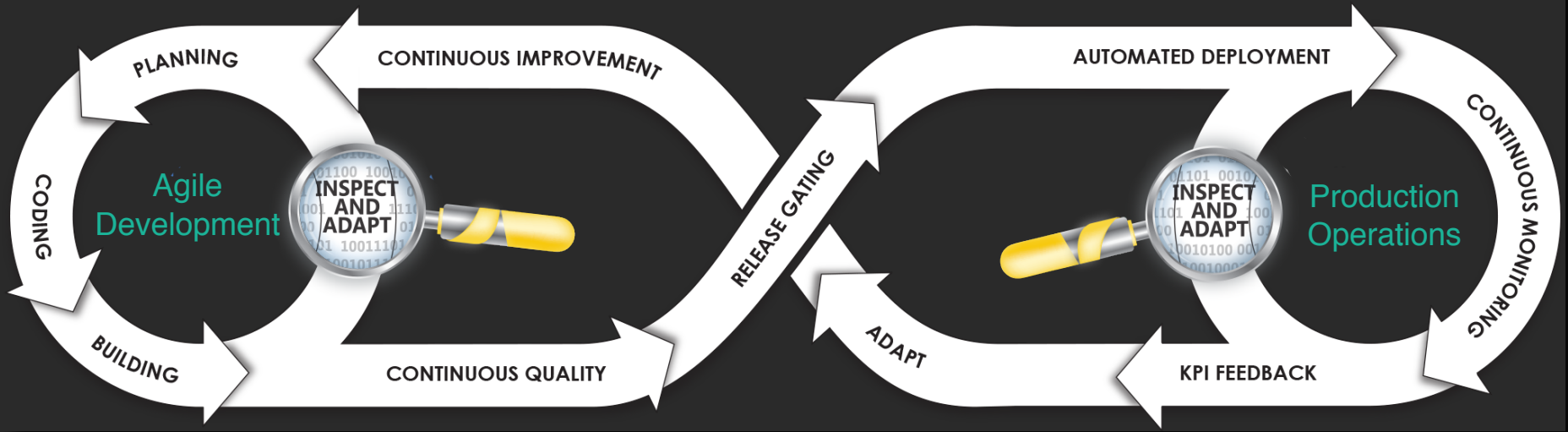
- After applications and systems have been promoted to production, they are continuously monitored for vulnerabilities.
- Depending on the severity of an identified vulnerability, applications might be removed from production or simply cleansed of the vulnerability.
- Typically, remediation involves the development organization, as refactoring may be required in order to update component libraries to versions that have eliminated the threat.



1.1 – Overview of DevSecOps

Release-and-Adapt

- Lessons learned from production performance often prompt subsequent iterations of software enhancement to remediate performance and security defects.
- A release-and-adapt cadence is the process through which key performance indicators are communicated back to developers so that remedial actions may be taken.



DevSecOps Essentials

Section 1: Introduction

Lesson 2: Cyber Security Concepts and Standards

Introductory Remarks

- Cyber security is a vast field of study.
- Qualified security professionals have many years of experience combined with many certifications.
- This section of the course is *not* a comprehensive discussion of this complex topic.
- The concepts, terms, and standards provided in this video are intended to give you a foundational introduction to the DevSecOps challenge.
- If you hope to specialize in a security-related discipline, I encourage you to pursue further study beyond this course.

Attack Surface

- The **attack surface** of a system is the collection of points (**attack vectors**) where an unauthorized user (**attacker**) may enter to inject data to or extract data from an environment.
- Keeping the attack surface as small as possible is a basic security measure.

Malware and Vulnerabilities

- **Malware** is malicious software that attackers deploy to infect individual computers or an entire digital network.
- Malware exploits target system vulnerabilities that can be hijacked, such as bugs in legitimate software (e.g., browser or web application plugins).
- A **vulnerability** is a weakness or deficiency in a computer system that an attacker can exploit to perform unauthorized actions within the system.

Identifying and Scoring Vulnerabilities

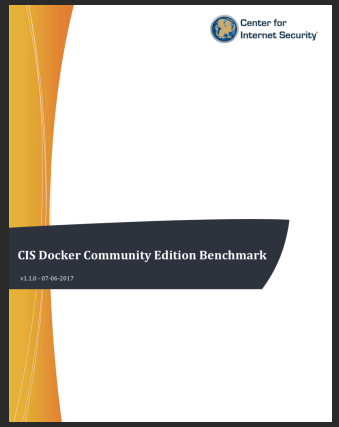
- Common Vulnerabilities and Exposures (CVE) is a dictionary-style list of standardized names for vulnerabilities and other information related to security exposures.
- CVE aims to standardize the names of all publicly known vulnerabilities and security exposures.
- The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of security vulnerabilities.
- CVSS assigns severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat level.

The OpenSCAP Project

- The Security Content Automation Protocol (SCAP) is a U.S. security standard maintained by the National Institute of Standards and Technology (NIST).
- The OpenSCAP Project is a collection of open-source tools for implementing and enforcing this standard.

The Center for Internet Security (CIS)

- The Center for Internet Security (CIS) provides security benchmarks and the National Checklist Program (NCP), defined by the NIST.
- They offer guidance on the security configurations of the operating system, database, virtualization, framework, and applications.
- In addition to the benchmark documents, the CIS also provides downloadable tools for secure configuration scanning.



Malware and Vulnerability Scanners

- Scanners can be deployed to network hosts and run in memory-resident mode to monitor activity in real time.
- By monitoring multiple sensor points, scanners can log vulnerabilities so that DevSecOps stakeholders become aware of the need for remedial action.
- Scanners should be used to interrogate new and existing software to determine whether any system or application may be infected by threat actors or attackers.
- There are two types of scanning: dynamic scanning and static scanning.

Dynamic vs. Static Scanning

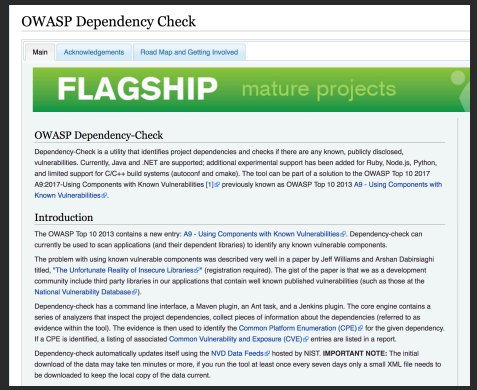
- **Dynamic scanning** is a method of code analysis that identifies vulnerabilities in a *runtime* environment.
- Dynamic tests monitor system memory, functional behavior, response time, and the overall performance of the system.
- Automated scanning tools can be used to analyze applications for which you do not have access to the original source code.
- **Static scanning** is a method of analysis performed in a *non-runtime* environment.
- Typically, a static analysis tool will inspect program code for all possible runtime behaviors and seek out flaws and potentially vulnerable code.
- Although they were developed separately, static and dynamic scanning are *not* in opposition to one another.
- There are strengths and weaknesses associated with both approaches, and many DevSecOps processes benefit from using both.

The Cloud Controls Matrix (CCM)

- The Cloud Security Alliance (CSA) has consolidated most security compliance methods into a single resource called the Cloud Controls Matrix (CCM).
- The CCM includes all security compliance controls such as ISO, FedRAMP, and NIST 800-53.
- It defines the control ID used to uniquely identify vulnerabilities.
- The key benefit of the CCM is that we can consult one aggregate source of security compliance regulations.
- The CSA also provides the Consensus Assessments Initiative Questionnaire (CAIQ). This questionnaire is a means of security self-assessment for both cloud consumers and providers.

The Open Web Application Security Project (OWASP)

- The **Open Web Application Security Project (OWASP)** is an online community project that provides free articles, methodologies, documentation, tools, and technologies for web application security.
- The OWASP Foundation came online on December 1st, 2001.
- It was established as a not-for-profit charitable organization in the United States on April 21, 2004.
- OWASP is an international organization, and the OWASP Foundation supports OWASP efforts around the world.



Federal Information Processing Standards (FIPS)

- The FIPS are security standards developed by the U.S. federal government for use in non-military government computer systems.
- FIPS defines minimum security requirements for the use of cryptographic modules.
- The FIPS publication *Security Requirements for Cryptographic Modules* explains which cryptographic modules are considered safe, legacy, or weak.
- To learn more about cryptographic modules, check out the following resources:
  - https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet
  - https://www.owasp.org/index.php/Guide_to_Cryptography
  - https://www.owasp.org/index.php/Key_Management_Cheat_Sheet

DevSecOps Resources

- The National Checklist Program (NCP) repository provides secure configuration for specific software components.
- DevSecOps stakeholders can use the NCP repository to search the CIS for information on particular software products.
- The NCP repository provides metadata and links to security checklists, including checklists that conform to the SCAP.
- The NIST provides an online search tool as well as data feeds that are used by many SCAP-validated tools to provide intelligence for advanced DevSecOps monitoring.
- The National Vulnerability Database (NVD) is a term security professionals use to refer to this NIST repository and other data stores provided by the NIST.

DevSecOps Essentials

Section 1: Introduction

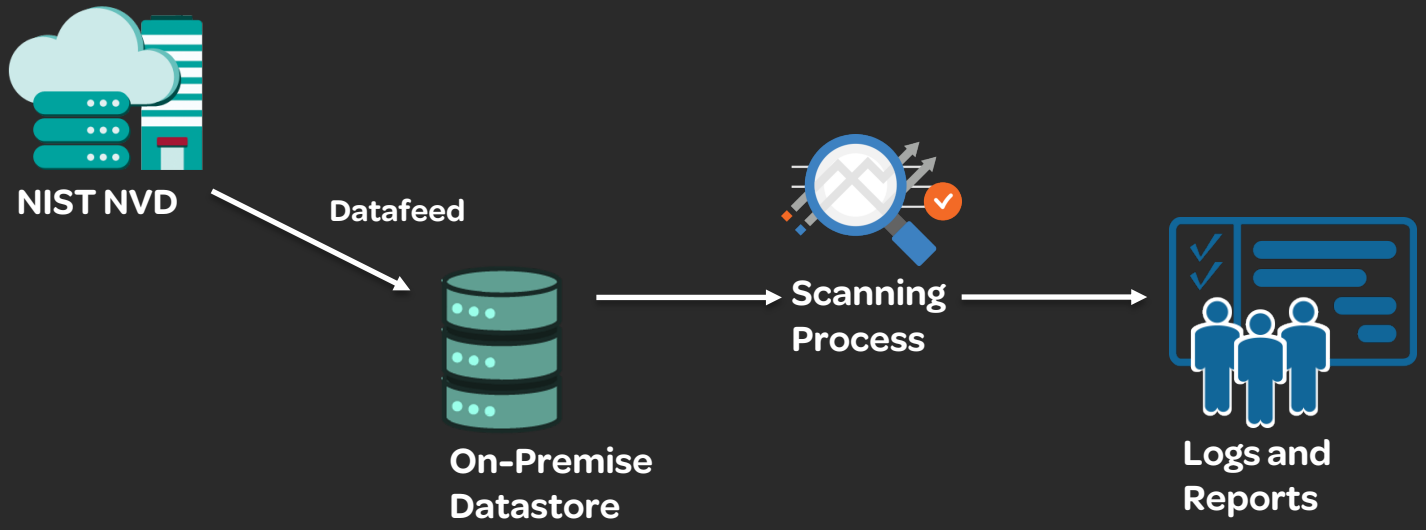
Lesson 2: Cyber Security Concepts and Standards, Part 2

Identifying and Scoring Vulnerabilities

- Common Vulnerabilities and Exposures (CVE) is a dictionary-style list of standardized names for vulnerabilities and other information related to security exposures.
- CVE aims to standardize the names of all publicly known vulnerabilities and security exposures.
- The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of security vulnerabilities.
- CVSS assigns severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat level.

National Vulnerability Database

- The NIST NVD provides Datafeeds and realtime API's that allow tooling to access the data and use it as intelligence for scanning and monitoring tools.



DevSecOps Resources

- The National Checklist Program (NCP) repository provides secure configuration for specific software components.
- DevSecOps stakeholders can use the NCP repository to search the CIS for information on particular software products.
- The NCP repository provides metadata and links to security checklists, including checklists that conform to the SCAP.
- The NIST provides an online search tool as well as data feeds that are used by many SCAP-validated tools to provide intelligence for advanced DevSecOps monitoring.
- The National Vulnerability Database (NVD) is a term security professionals use to refer to this NIST repository and other data stores provided by the NIST.

DevSecOps Essentials

Section 1: Introduction

Lesson 2: Cyber Security Concepts and Standards, Part 1

Introductory Remarks

- Cyber security is a vast field of study.
- Qualified security professionals have many years of experience combined with many certifications.
- This section of the course is *not* a comprehensive discussion of this complex topic.
- The concepts, terms, and standards provided in this video are intended to give you a foundational introduction to the DevSecOps challenge.
- If you hope to specialize in a security-related discipline, I encourage you to pursue further study beyond this course.

Attack Surface

- The **attack surface** of a system is the collection of points (**attack vectors**) where an unauthorized user (**attacker**) may enter to inject data to or extract data from an environment.
- Keeping the attack surface as small as possible is a basic security measure.

Malware and Vulnerabilities

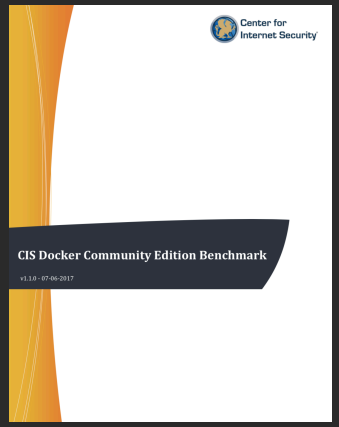
- **Malware** is malicious software that attackers deploy to infect individual computers or an entire digital network.
- Malware exploits target system vulnerabilities that can be hijacked, such as bugs in legitimate software (e.g., browser or web application plugins).
- A **vulnerability** is a weakness or deficiency in a computer system that an attacker can exploit to perform unauthorized actions within the system.

The OpenSCAP Project

- The Security Content Automation Protocol (SCAP) is a U.S. security standard maintained by the National Institute of Standards and Technology (NIST).
- The OpenSCAP Project is a collection of open-source tools for implementing and enforcing this standard.

The Center for Internet Security (CIS)

- The Center for Internet Security (CIS) provides security benchmarks and the National Checklist Program (NCP), defined by the NIST.
- They offer guidance on the security configurations of the operating system, database, virtualization, framework, and applications.
- In addition to the benchmark documents, the CIS also provides downloadable tools for secure configuration scanning.



Malware and Vulnerability Scanners

- Scanners can be deployed to network hosts and run in memory-resident mode to monitor activity in real time.
- By monitoring multiple sensor points, scanners can log vulnerabilities so that DevSecOps stakeholders become aware of the need for remedial action.
- Scanners should be used to interrogate new and existing software to determine whether any system or application may be infected by threat actors or attackers.
- There are two types of scanning: dynamic scanning and static scanning.

Dynamic vs. Static Scanning

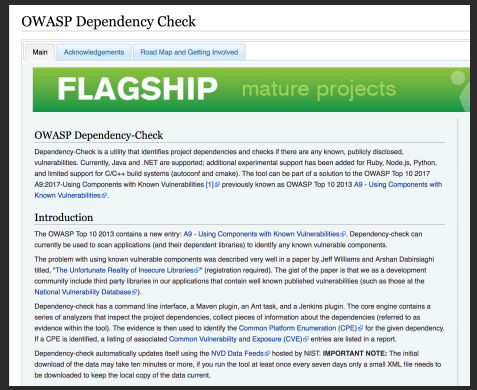
- **Dynamic scanning** is a method of code analysis that identifies vulnerabilities in a *runtime* environment.
- Dynamic tests monitor system memory, functional behavior, response time, and the overall performance of the system.
- Automated scanning tools can be used to analyze applications for which you do not have access to the original source code.
- **Static scanning** is a method of analysis performed in a *non-runtime* environment.
- Typically, a static analysis tool will inspect program code for all possible runtime behaviors and seek out flaws and potentially vulnerable code.
- Although they were developed separately, static and dynamic scanning are *not* in opposition to one another.
- There are strengths and weaknesses associated with both approaches, and many DevSecOps processes benefit from using both.

The Cloud Controls Matrix (CCM)

- The Cloud Security Alliance (CSA) has consolidated most security compliance methods into a single resource called the Cloud Controls Matrix (CCM).
- The CCM includes all security compliance controls such as ISO, FedRAMP, and NIST 800-53.
- It defines the control ID used to uniquely identify vulnerabilities.
- The key benefit of the CCM is that we can consult one aggregate source of security compliance regulations.
- The CSA also provides the Consensus Assessments Initiative Questionnaire (CAIQ). This questionnaire is a means of security self-assessment for both cloud consumers and providers.

The Open Web Application Security Project (OWASP)

- The **Open Web Application Security Project (OWASP)** is an online community project that provides free articles, methodologies, documentation, tools, and technologies for web application security.
- The OWASP Foundation came online on December 1st, 2001.
- It was established as a not-for-profit charitable organization in the United States on April 21, 2004.
- OWASP is an international organization, and the OWASP Foundation supports OWASP efforts around the world.



Federal Information Processing Standards (FIPS)

- The FIPS are security standards developed by the U.S. federal government for use in non-military government computer systems.
- FIPS defines minimum security requirements for the use of cryptographic modules.
- The FIPS publication *Security Requirements for Cryptographic Modules* explains which cryptographic modules are considered safe, legacy, or weak.
- To learn more about cryptographic modules, check out the following resources:
  - https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet
  - https://www.owasp.org/index.php/Guide_to_Cryptography
  - https://www.owasp.org/index.php/Key_Management_Cheat_Sheet

DevSecOps Essentials

Section 1: Introduction

Lesson 3: Identity and Access Management

Secure Automation

- In **DevSecOps**, many of the processes required for software development, testing, and deployment are automated.
- Security checks should be implemented for each server and tool to reduce the number of possible **attack vectors** and prevent malfeasance.
- Security practices are different for every team and system, but there are three main categories: server hardening, application hardening, and identity and access management.

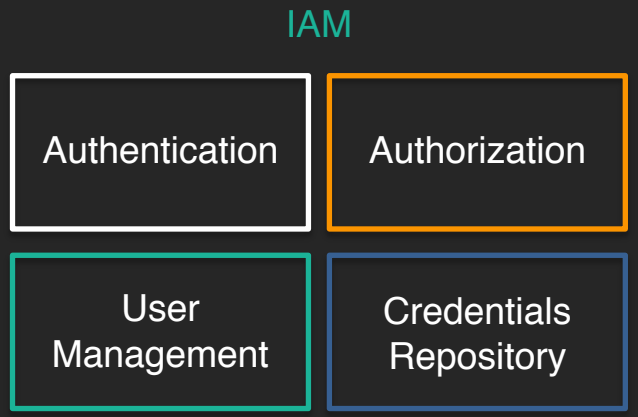
DevSecOps Pipeline Flow

Development — QA Test — Staging — Production

The platforms and systems used to automate DevSecOps processes must be secured individually according to the specific technology they employ.

Identity and Access Management

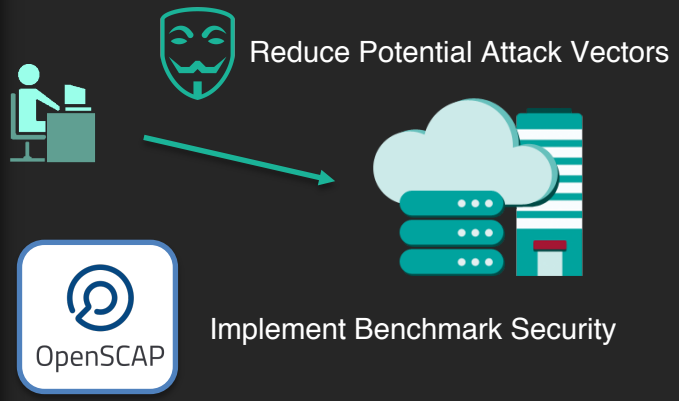
- **Identity and Access Management (IAM)** is the process of granting or restricting access to computing resources for individual users, groups, or systems.
- All modern software applications govern user and group authentication, but their specific methods vary.
- Systems that access applications through APIs (Application Program Interfaces) also have frameworks for managing access to those applications.
- To harden servers and applications in a DevSecOps pipeline, teams have to implement security practices based on the particular tooling being used.



Identity and Access Management involves authenticating, authorizing, and managing users and their credentials via automation and security frameworks.

Server Hardening

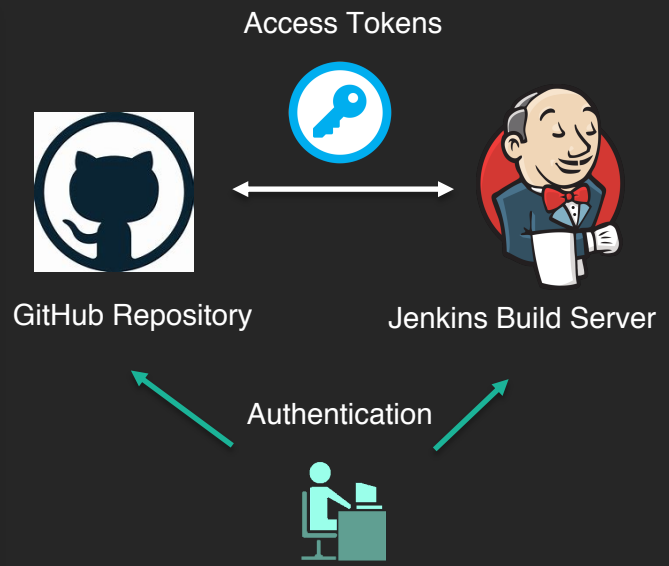
- **Server hardening** is the practice of enhancing each server's security.
- Teams can consult benchmarks from CIS and applications such as OpenSCAP to review possible server vulnerabilities and determine what steps to take to mitigate risks.
- A server must be hardened before the applications and tooling hosted on the server can be secured.



The security community has developed best practice guidelines and automated tooling for server hardening.

Application Hardening

- **Application hardening** is the practice of enhancing an application or framework's security according to the provider's recommendations.
- Most DevSecOps tooling involves automated integration with other third-party systems through APIs.
- The process of hardening an application includes both the initial implementation of security assets and ongoing governance over time.



Application hardening consists of conventional access authentication for humans and token management for application-to-application interfaces.

IAM Fundamentals

- **Identity repositories** are systems that store information about all the users and groups within an enterprise in a single place.
- **Access keys** are encrypted keys that enable applications to securely access servers.
- **Signatures** and **certificates** allow programs to verify the source and authenticity of digital assets.
- **Vaults** store secrets and encrypt login credentials to prevent attackers from accessing them.

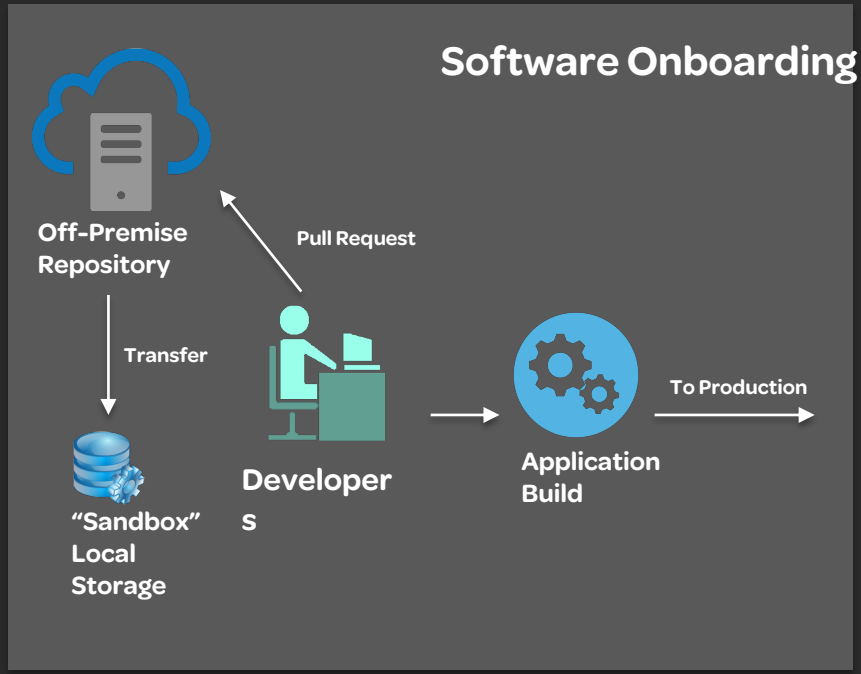
Cyber security is an important part of software development, and all DevSecOps stakeholders should be familiar with fundamental security practices.

DevSecOps Essentials

Section 2: Secure Software Onboarding

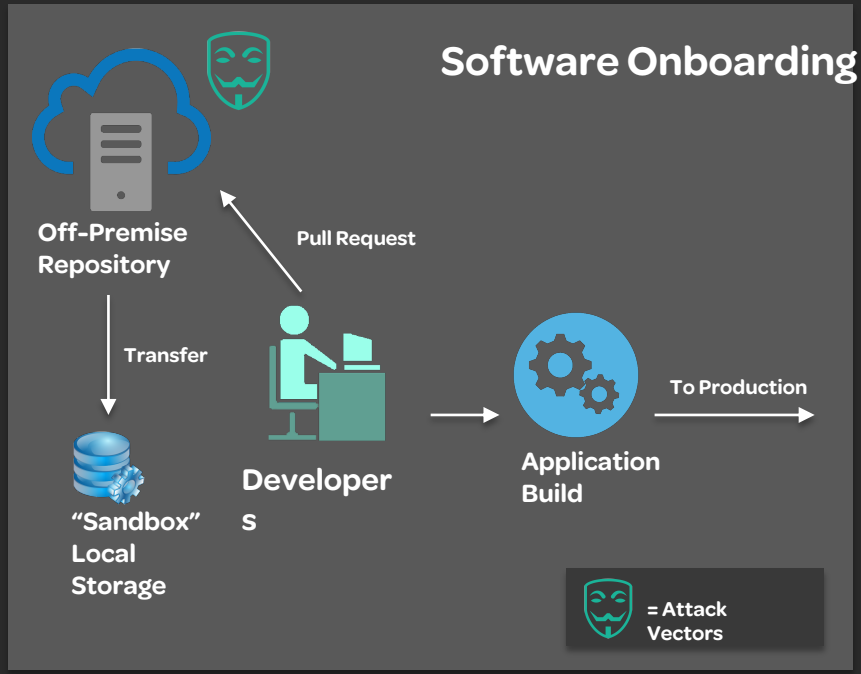
Lesson 1: Clean Repositories

Software Repositories



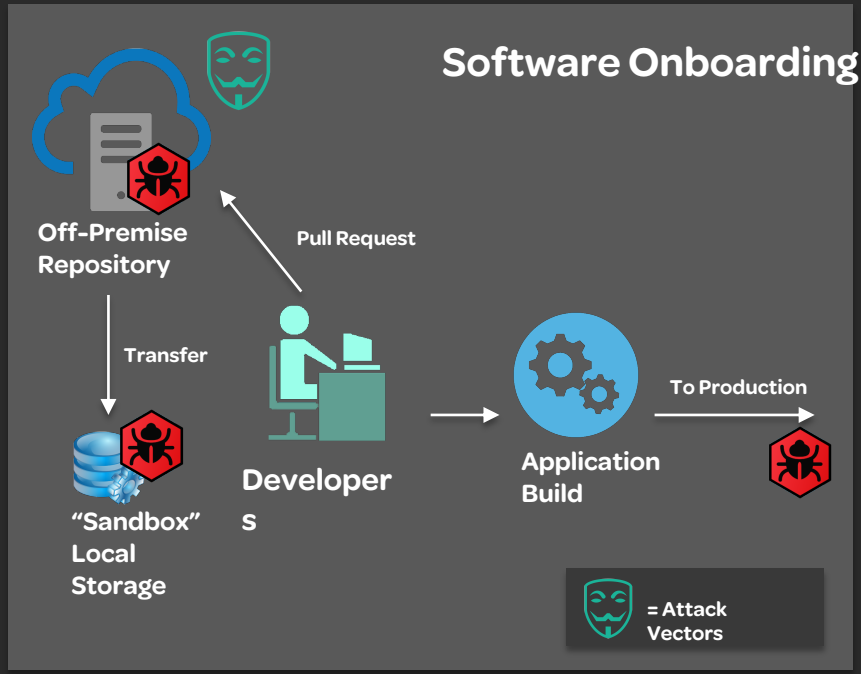
- Software onboarding is the process developers use to acquire the third-party components needed to develop and compile an application.
- With the rise of open source, many third-party component libraries and frameworks are acquired from public repositories.
- In some cases, off-premise repositories are not administered or maintained by the developer's organization.

Security in Public Repositories



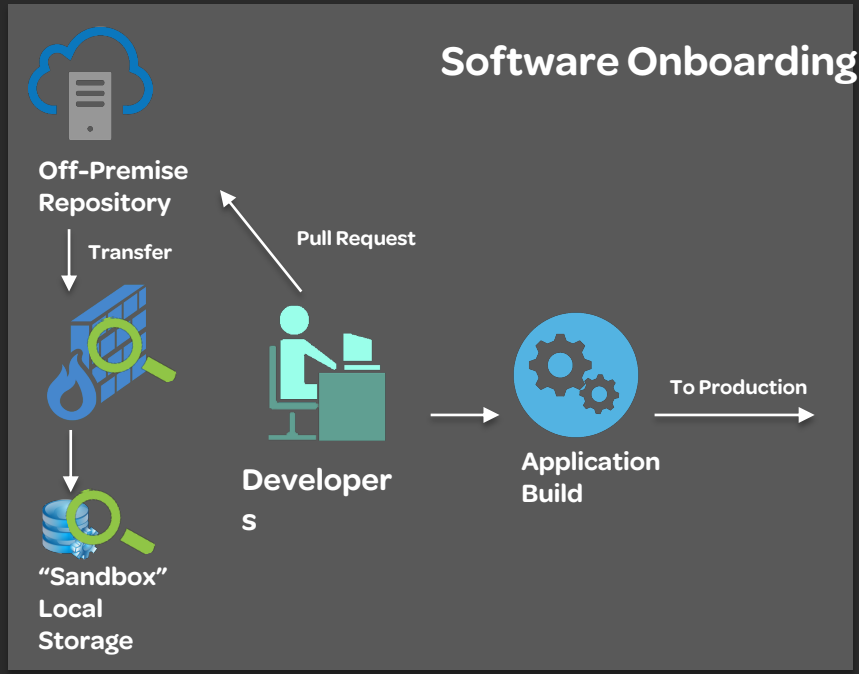
- Attack vectors are often exploited in public systems.
- Since the systems are not on-premise, they must be hardened and maintained by the organizations that provide them.
- Highly regulated industries require levels of security and protection that may exceed those practiced by the organizations maintaining public repositories.

Malware



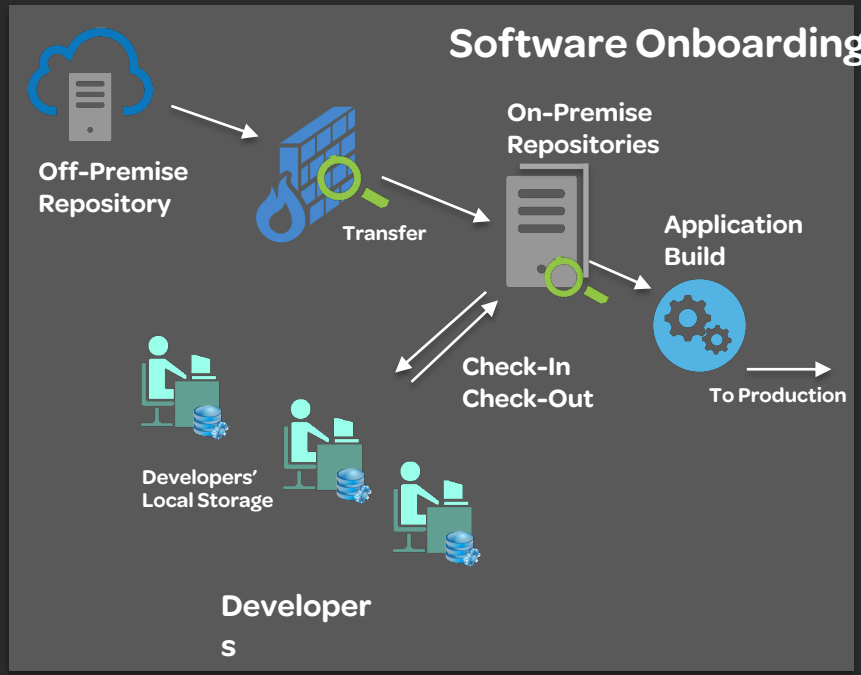
- Malware exists in most public repositories, and some of the most popular software components in use today are known to be vulnerable.
- If the software in public repositories is transferred to internal systems, the malware is brought into the on-premise infrastructure.
- Once inside the private infrastructure, malware can be unknowingly propagated to production where vulnerabilities are further exploited by attackers.

Securing On-Premise Repositories



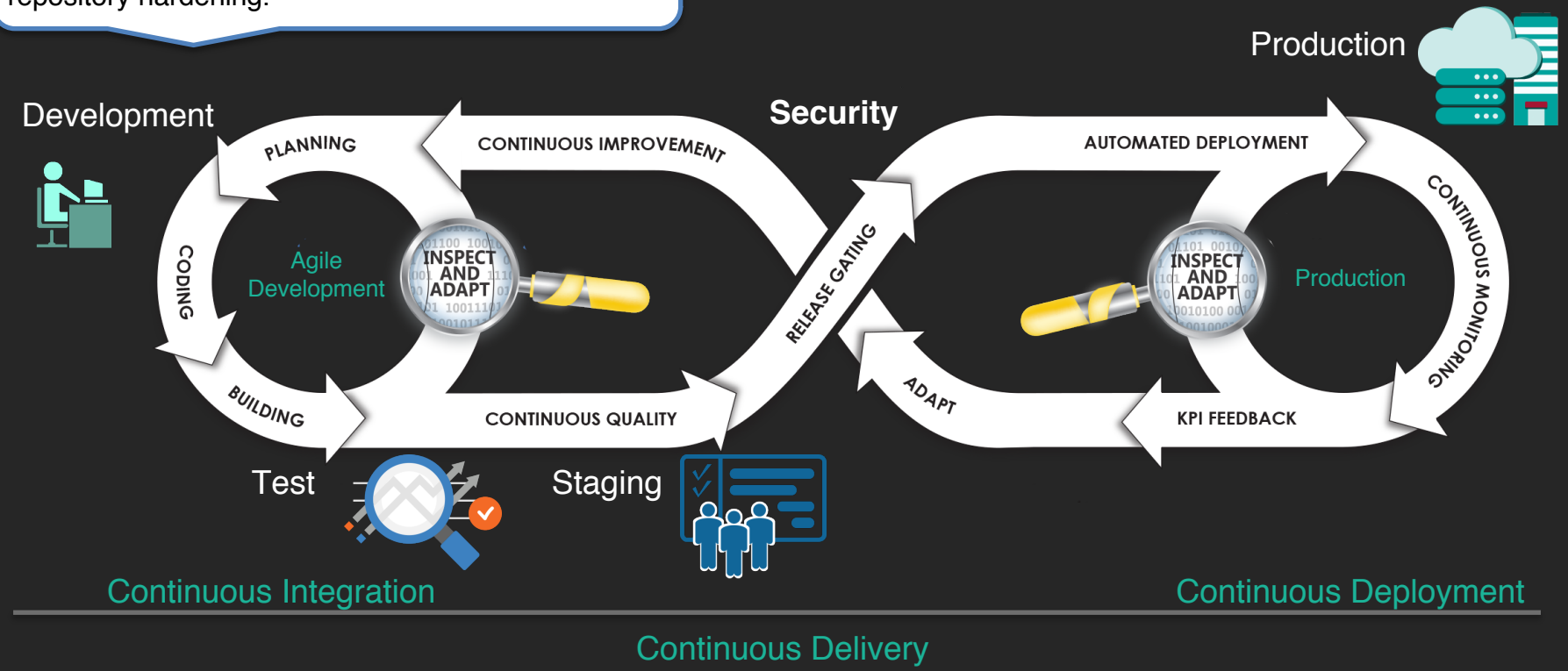
- Firewall systems prevent outside attackers from accessing internal systems.
- At the time of onboarding, software should be scanned for known vulnerabilities.
- On-premise systems must be hardened to prevent malfeasance.
- On-premise repositories must be scanned and monitored on an ongoing basis to ensure threats do not penetrate the security perimeter.

Shared Repositories



- Large organizations have teams of developers using shared on-premise repositories.
- Source code control and source code management are needed when multiple developers are working on the same code base.
- Check-in and check-out processes track code versions and allow branching from the trunk, facilitating sub-releases for release management.
- Continuous integration is a practice that encourages developers to check in their code daily to ensure that one change does not introduce a defect into the shared code base.

Clean repositories affect automated provisioning. Secure Platform as a Service (PaaS) use requires repository hardening.



DevSecOps Essentials

Section 2: Secure Software Onboarding

Lesson 2: Securing Public Repositories

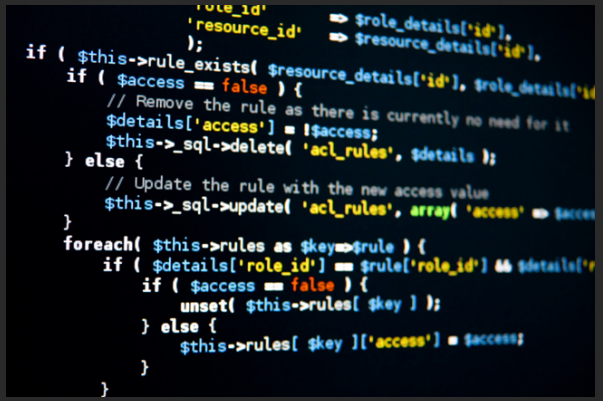
Use Case: GitHub

GitHub is a useful and popular **off-premise repository** used by many organizations.

We will use GitHub as an example of a **public repository** because it is one of the most well-developed repositories in use today.

Why Attackers Target GitHub

- The files stored in GitHub are often application source code and, thus, valuable intellectual property.
- Copying the code may enable other companies or even nation-states to quickly develop derivative applications, saving years or even decades of development time and leveraging trade secrets without paying licensing fees.
- Hackers might also steal code to resell it on the dark web.



Application source code contains millions of lines of instructions where attackers can embed code that creates vulnerabilities when it is run in production.

Threats Facing GitHub

- Hackers can study application source code to learn how to attack the software when it's run in production.
- Stealing source code gives hackers time to look for vulnerabilities that would be more difficult to detect through the penetration of production systems.
- Attackers can even run code in production on their own systems and test attacks against it, refining their attacks to increase their speed, stealth, and effectiveness.
- State-sponsored cyber terrorism and global corporate espionage are well-funded efforts that retain professionals to develop methods of intrusion and attack.

Login and Authentication

Usernames and passwords may be hard-coded in source code because one program module requires access to another system or API. This is a known vulnerability, and best practices dictate that authentication credentials should be stored externally in encrypted digital vaults.

- Sometimes files checked into GitHub inadvertently contain **login credentials** for other internal or external services.
- The rise of software as a service (SaaS) and microservice architectures means that many application components have to **authenticate** and establish a **secure session** with an application program interface (API) on another system.
- Some developers cut corners by embedding login and authentication data in the source code or supporting files checked in to GitHub.
- When hackers gain access to the code, they can gain access to related services, giving them the opportunity to steal more data and disrupt operations.

Unauthorized Access

- In many cases, developers are granted access to company repositories from their personal or corporate **email accounts**.
- These e-mail accounts may be left vulnerable, as some organizations do not purge old e-mail accounts in a timely manner when employees leave the company.
- Another poor security practice is when developers are granted access to all of the company's repositories instead of just what they need for their projects.
- Over time, deficient security policies and procedures create a **wide-open attack surface** that attackers can exploit.

Passwords used for e-mail accounts are often used for other purposes, giving attackers additional information they can use to breach security.

Insider Threats

- A lack of proactive monitoring can allow malicious insiders to hide abnormal activities.
- A single developer accessing many repositories could be an early indicator of an insider threat, and such behavior should be detected and flagged.
- Unfortunately, due to limited resources, many organizations do not monitor the use of GitHub and other external repositories.
- This gives bad actors the opportunity to continue their attacks.

To monitor external repositories, it is often necessary to download the activity logs produced by the remote system, and then use automated parsing to analyze the log for suspicious activity.

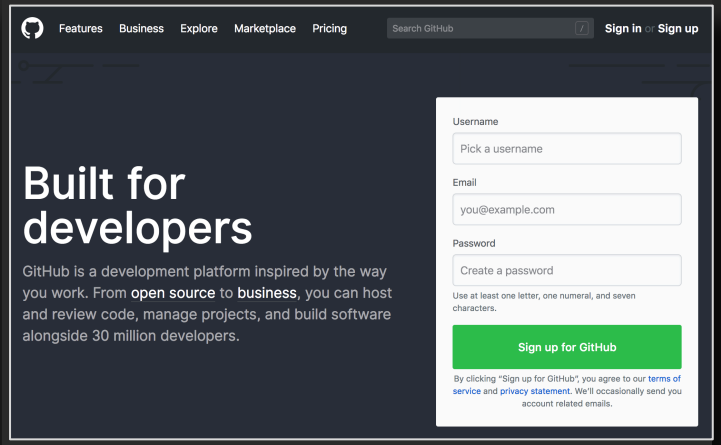
Practical Steps for DevSecOps

Threat Monitoring for GitHub

- DevSecOps practitioners can take practical steps to tighten security when using public repositories.

Clean Up Login Credentials

- Developers should be careful with their GitHub login credentials.
- It is best to **limit access** to only those developers who are involved in a given project.
- When developers leave a project, their credentials should be **revoked**.



Login credentials must be continually updated to prevent broadening the attack surface.

Maintain and Administer Repository Settings

The software behind GitHub — the software version control program *Git* — was originally developed for managing development of the Linux kernel.

Both Git and GitHub are widely used in open-source projects.

Some developers, particularly those that contribute to open-source projects, treat all GitHub repositories as public, whether they're for open-source projects or not.

DevSecOps best practice requires that your organization's GitHub configuration be maintained and administered on an ongoing basis to ensure that access isn't any broader than necessary.

Monitor GitHub For Suspicious Activity

Many attacks originate from remote locations. Organizations can map IP addresses to locations where known employees and business partners reside. Access from outside these locales often indicates suspicious activity.

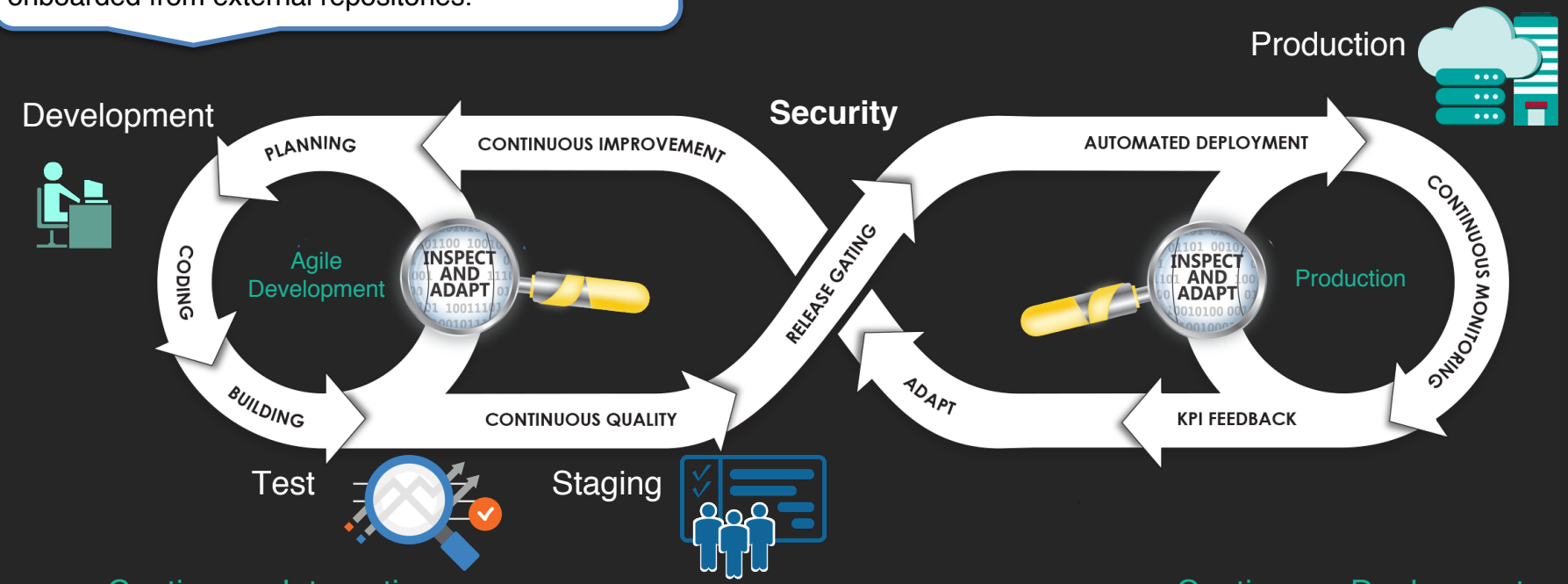
- Automated and manual monitoring of GitHub can detect a sudden spike in source code check-ins.
- Monitoring should also detect the check-out of an unusually large amount of source code.
- It is also possible to monitor and detect logins from unusual locations, or logins or requests from users outside the organization.

Aggregate and Analyze GitHub Logs

Threat Monitoring for GitHub

- GitHub logs can be aggregated and downloaded on a regular basis for further analysis.
- Tools are available for parsing logs and analyzing activity against a baseline of what would be considered normal use.
- These practices can be automated so that constant vigilance is performed.
- When unusual activity is detected it can be reported through dashboards or even messaged to stakeholders as a means of providing early warning of potential threats.

Public repositories must be maintained to ensure that PaaS provisioning does not spread malware onboarded from external repositories.



DevSecOps Essentials

Section 2:Secure Software Onboarding

Lesson 2.3 : Secure Containerization

Containers

- **Containers**: Specially formatted files that contain executable application programs, modules, and the code libraries that they depend upon.
- They isolate an application from other applications and from the operating system and hardware.
- Containers help ensure that the versions of libraries within the container do not mandate patching and upgrades that might affect other applications in other containers.
- In virtual environments, the libraries and dependencies of an application were oftentimes shared with all of the other applications on a particular virtual server.

Use Case: Docker

- While there are many container engines to choose from, our use case is Docker.
- Process Restrictions use root/non-root dichotomy. This makes it difficult to provoke system level damages during an intrusion, even if the intruder manages to escalate to root within a container because the container capabilities are fundamentally restricted.
- Device and File Restrictions further reduces the attack surface by restricting access by containerized applications to the physical devices on a host, through the use of the device resource control groups (cgroups) mechanism.
- Container images are ephemeral. Each container has its own file system and can not write to the host file system unless writes are committed. Committing changes tracks and audits revisions made to the base images as a new layer which can then be pushed as a new image for storage in Docker Hub and run in a container.
- The audit trail is important in providing information to maintain compliance. It also allows for a fast and easy rollback to previous versions if a container has been compromised or a vulnerability introduced.

Docker Hub: The World's Largest Community of Container Images

Docker Hub is the world's largest public repository of container images with an array of content sources including container community developers, open source projects, and independent software vendors (ISV) building and distributing their code in containers.

Docker Hub allows you to:

- Search and browse for millions of container images
- View image popularity, vendor source and certification to inform your selection
- Access free public repositories to store your images and share with the community
- Choose a subscription plan for private repositories to limit access to your images
- Use Autobuilds and Webhooks for easy integration into your DevOps pipeline

Docker Hub

PUBLIC

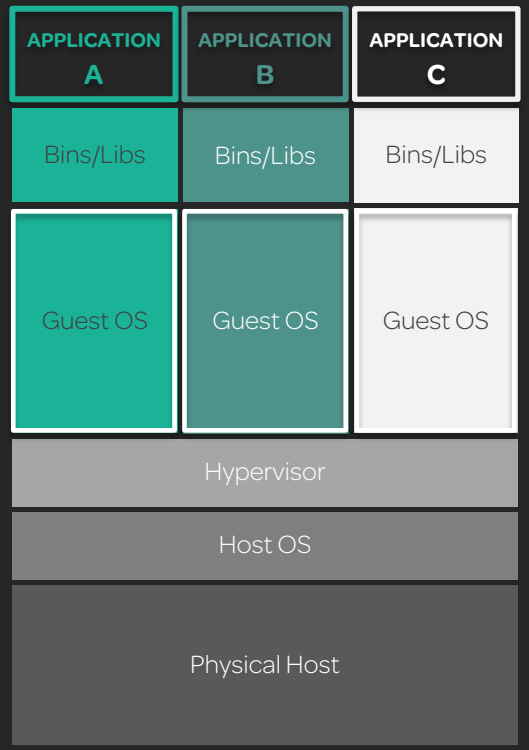
PRIVATE

OFFICIAL

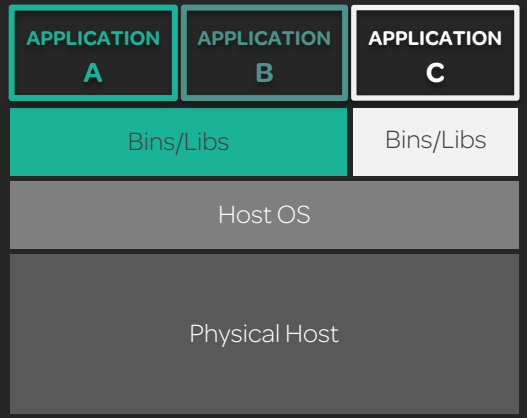
- While Docker Hub is a public repository, it contains public, private, and official images.
- Public images are shared amongst a broad community.
- Private images might be limited to a single organization or even project within an organization.
- Official repositories are certified repositories from independent software vendors (ISV's) and Docker contributors such as Canonical, Oracle, RedHat, and others.

VIRTUAL MACHINES

Virtual machines utilize a host OS with their own libraries. A hypervisor is used to facilitate the installation and running of guest virtual servers. The guest servers have their own image of an operating system and libraries. Applications within each virtual server rely on the version of the libraries and the guest operating system instance.



VS.

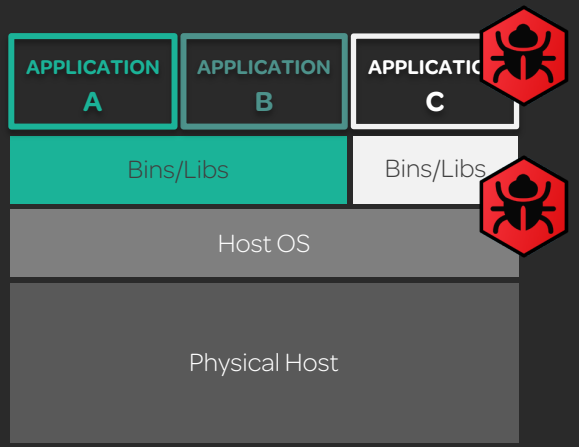


CONTAINERS

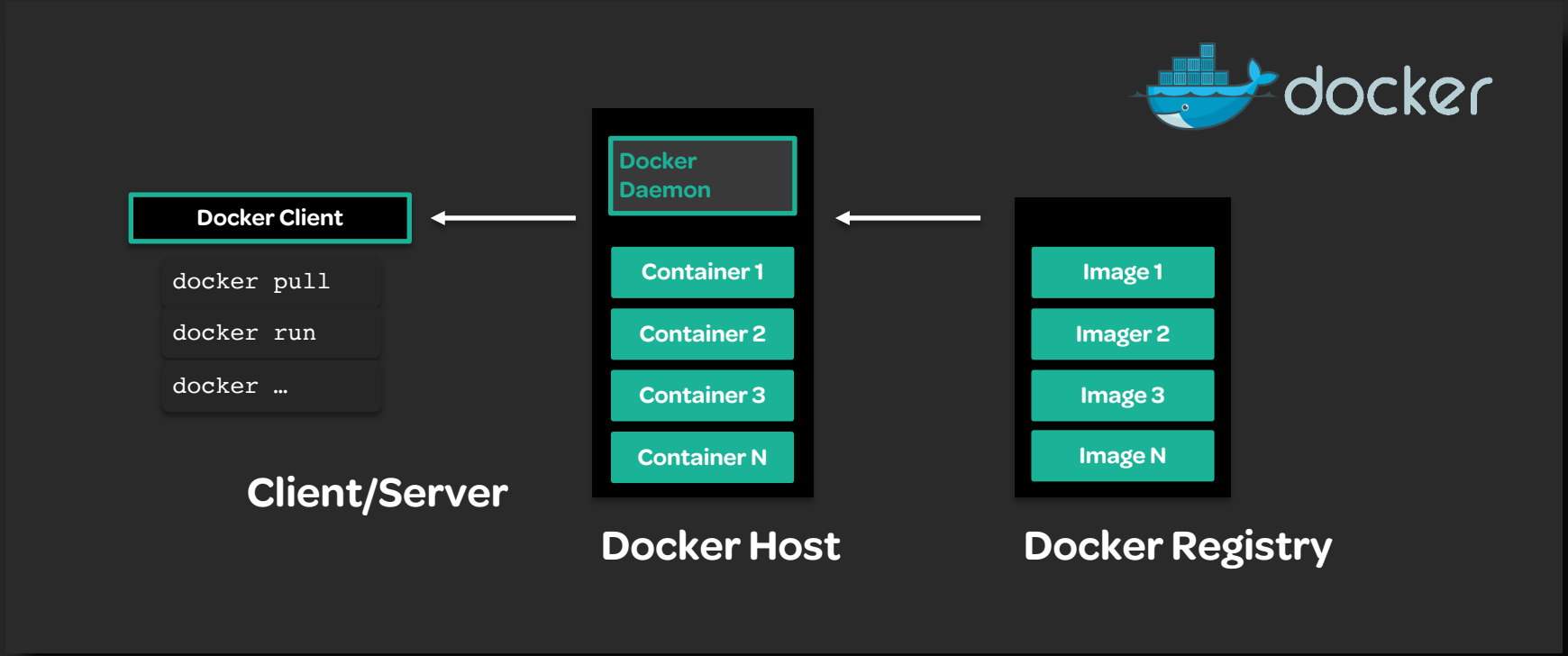
Containers rely on isolation techniques like control groups and namespaces to separate applications. They do not have a hypervisor and can share libraries if needed. Containers share the physical host's kernel, so they do not require an operating system. Container systems have processes in place to prevent a container from making potentially dangerous changes to the kernel.

Docker and Malware

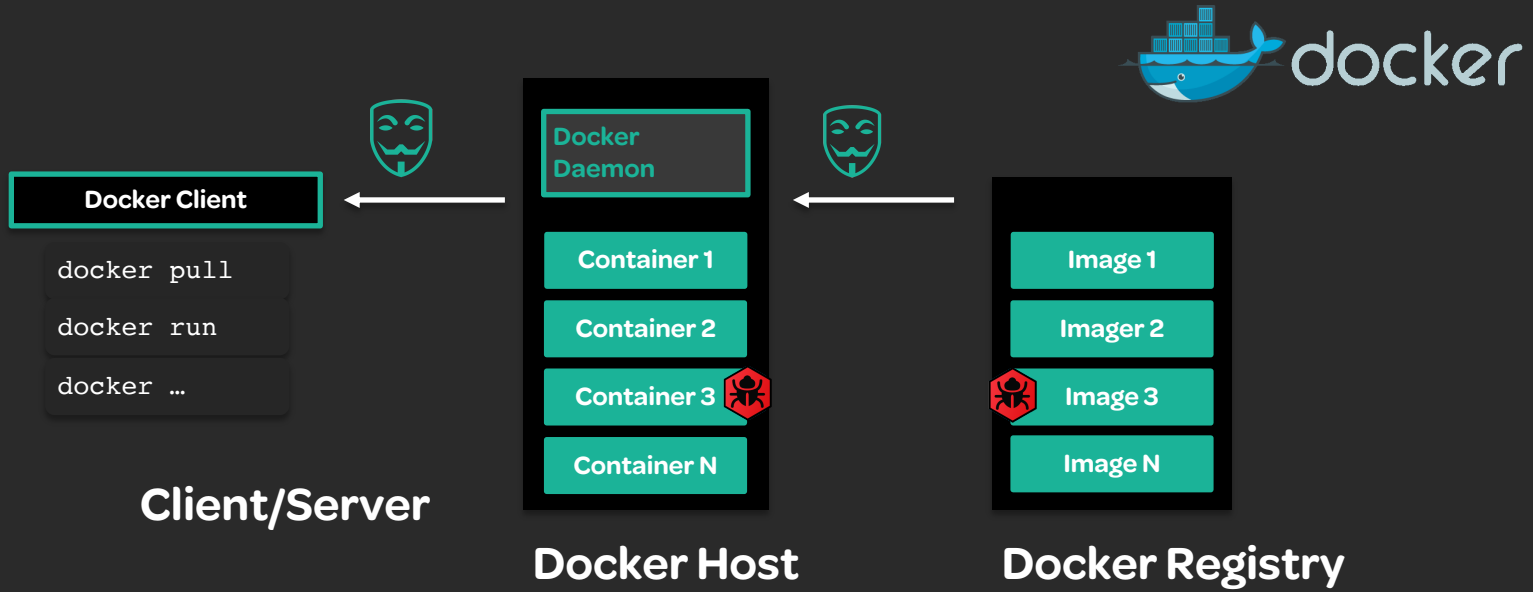
Just as a repository can contain malicious source code or infected libraries, Docker containers may likewise be tainted. When they are replicated broadly throughout on-premise and off-premise infrastructures, they can carry the malware with them and compromise the integrity of production systems.

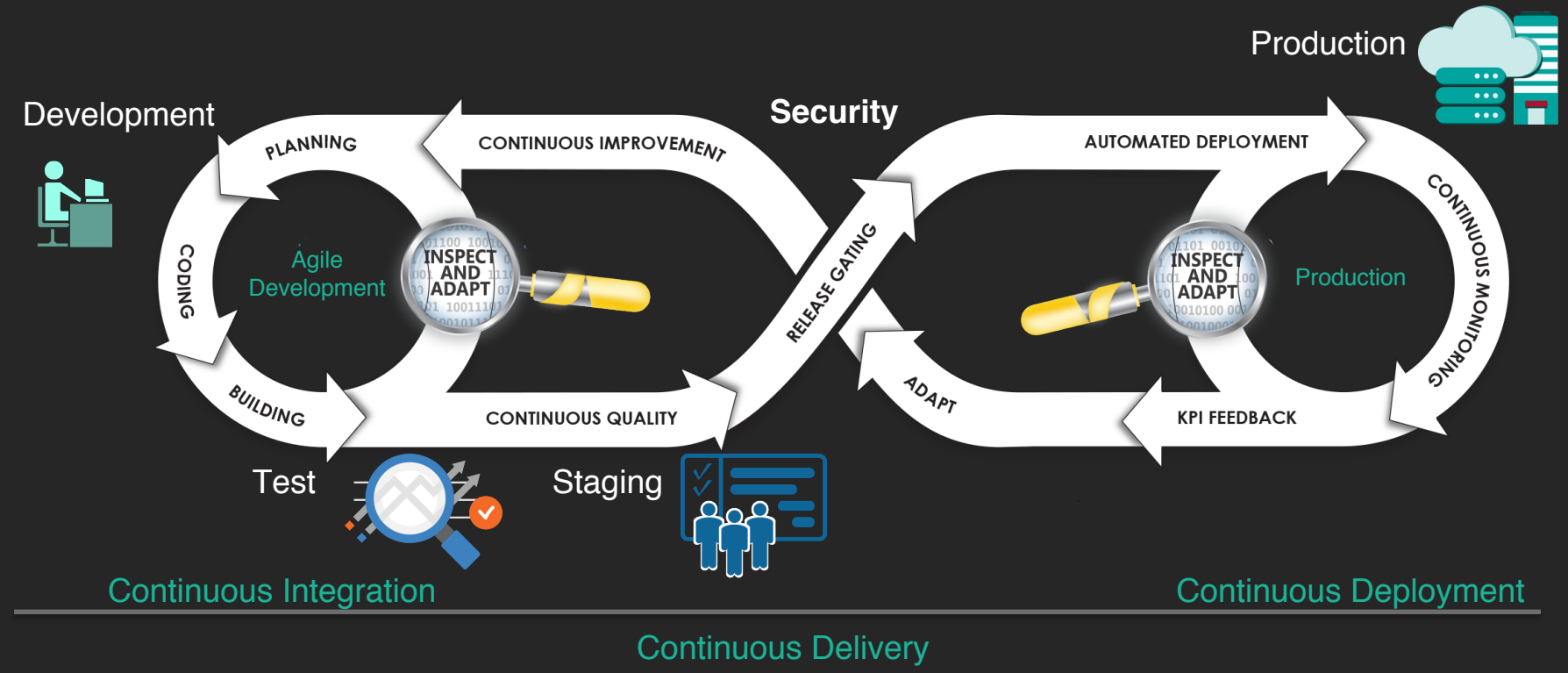


Docker Architecture



Docker Architecture (cont.)




Development
PLANNING
CONTINUOUS IMPROVEMENT
Security
AUTOMATED DEPLOYMENT
Production
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

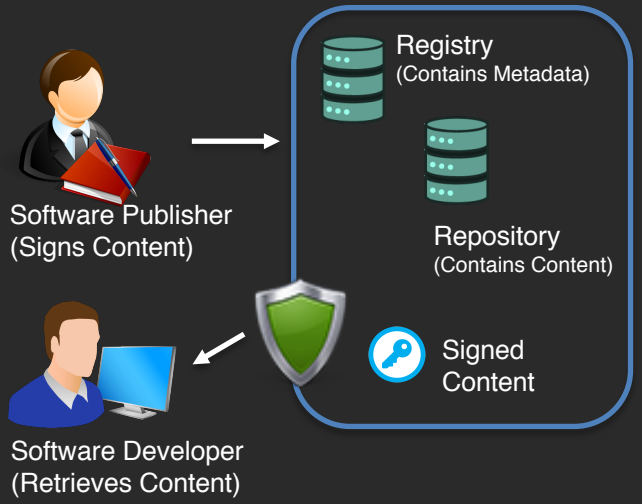
DevSecOps Essentials

Section 2: Secure Software Onboarding

Lesson 4: Docker Trusted Registries

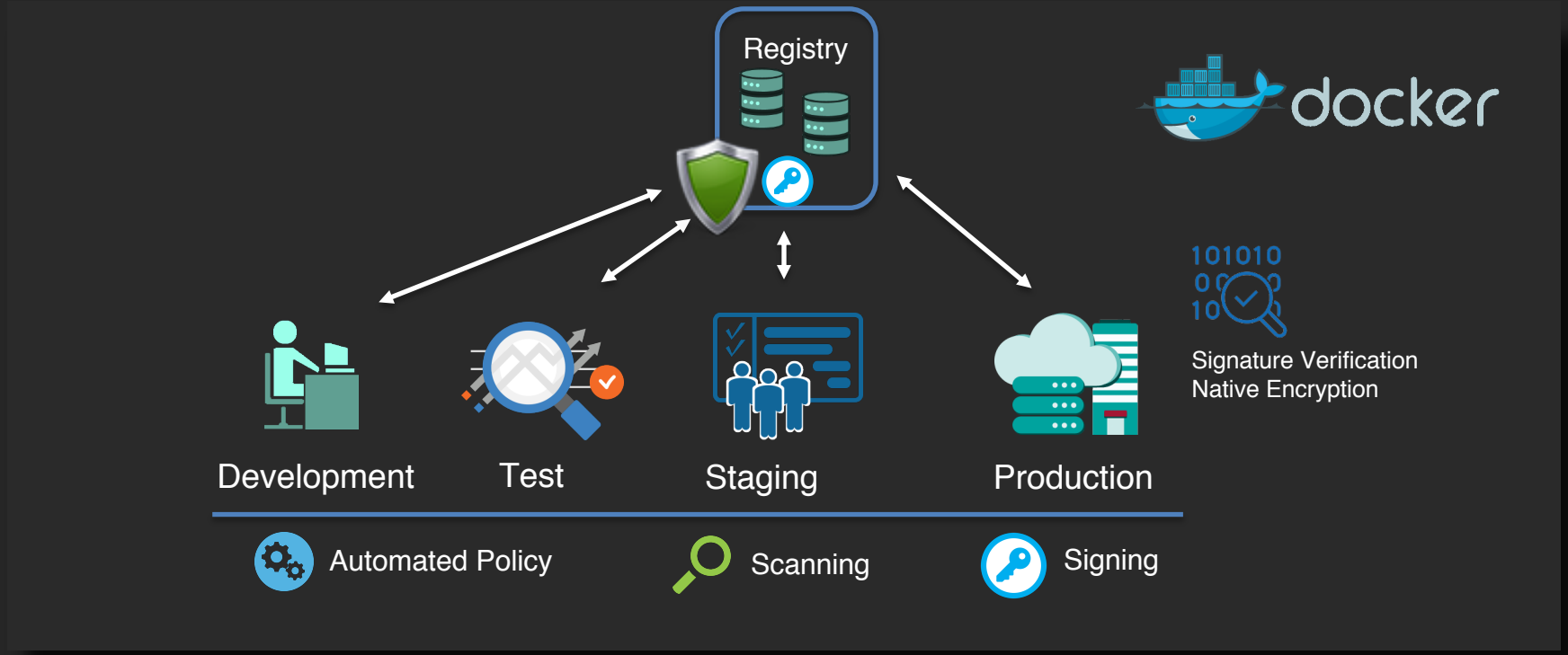
Trusted Content and Digital Signatures

Trusted Registries



- **Trusted registries** are secure managed repositories.
- A **registry** is a data store that contains **metadata** (i.e., data about data).
- A repository contains the actual content of interest.
- Trusted registries use a **digital signature** to indicate which content in the repository is safe.
- Software publishers digitally sign their content when it is published.
- Developers can use systems that verify digital signatures prior to downloading content.

Docker Trusted Repository



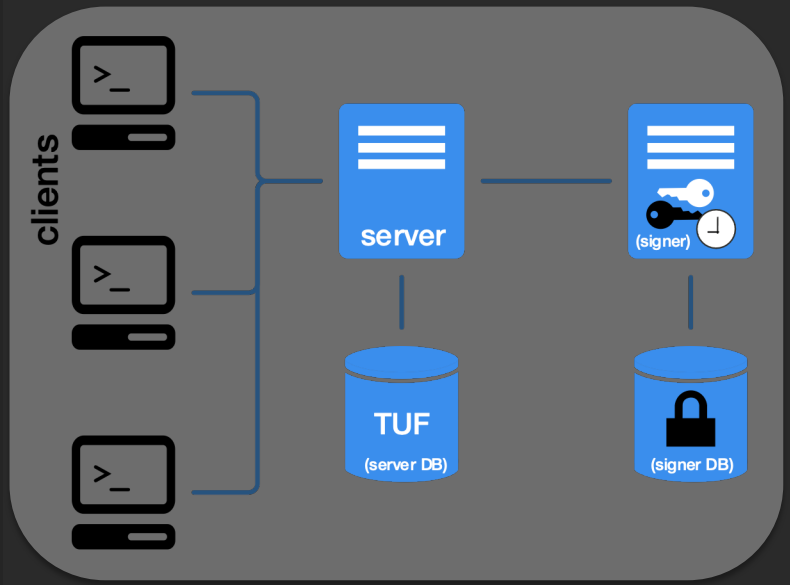
Content Trust in Docker

- Image signing and vulnerability scanning allow operations teams to protect the contents of containers.
- Metadata may contain information about the author, the bill of materials, and whether or not there is a critical vulnerability, which can help ensure that a container is safe at the time of onboarding.
- Automated insights enable organizations to meet compliance requirements and prevent security breaches.

Policy-Based Image Promotion

- **Policy-based image promotion** accelerates the DevSecOps pipeline by providing the data needed for automated or manual promotion.
- The gating process can be configured to require images to pass security scans.
- **Policy-driven automation** improves scale and speed by accommodating hundreds or thousands of container images.

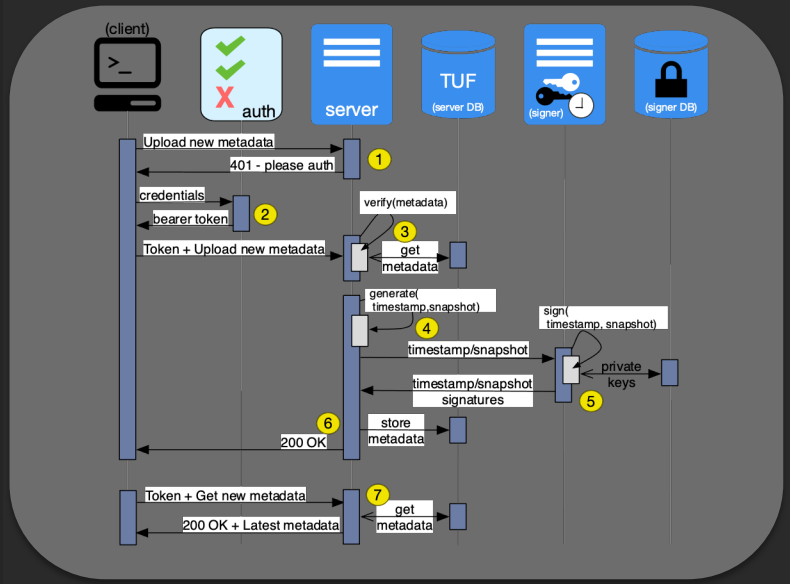
Docker Notary



Architecture and Components

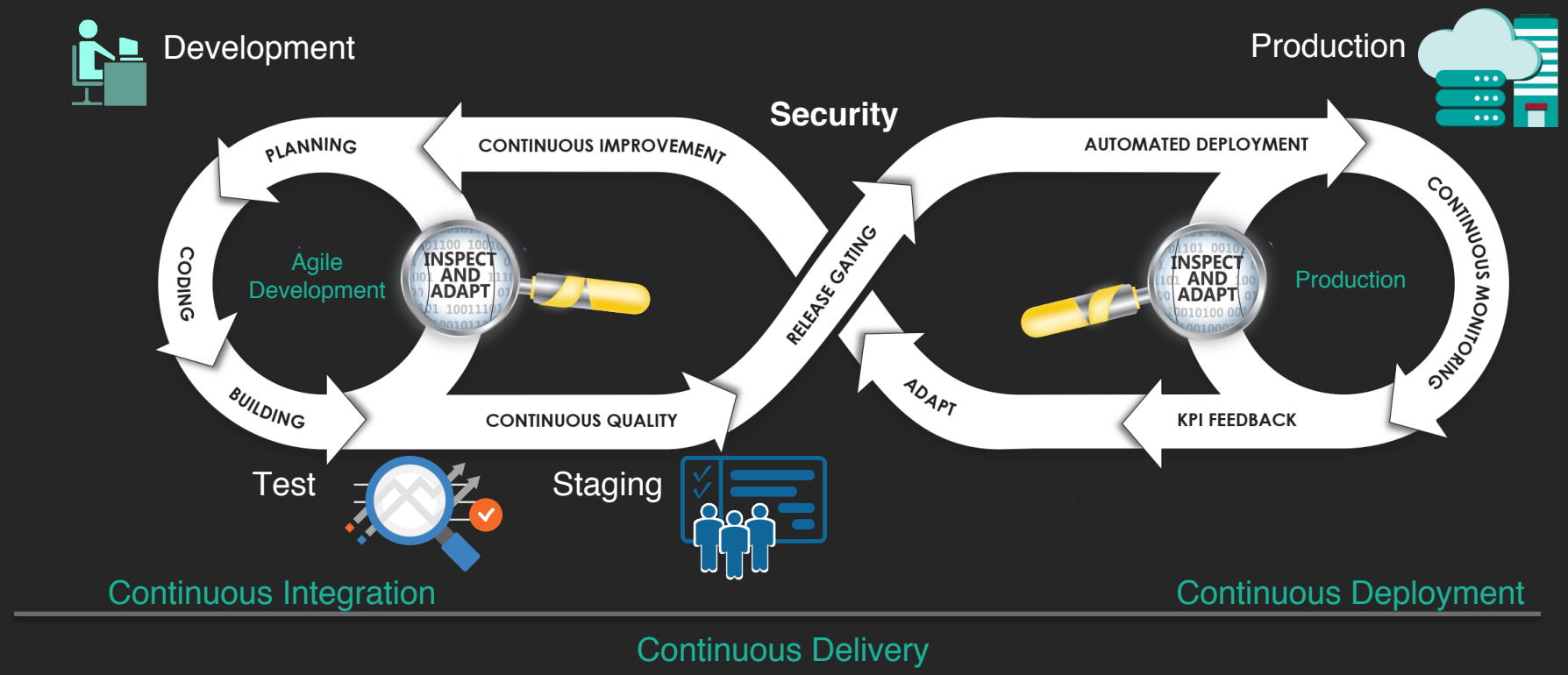
- Notary clients pull metadata from one or more remote Notary services.
- Some Notary clients also push metadata to one or more Notary services.
- A Notary service consists of a Notary server, which stores and updates the signed TUF metadata files for multiple trusted collections in an associated database.
- A Notary signer stores private keys and signs metadata for the Notary server.

Docker Notary (Continued)



Using Private Keys to Sign Containers

- Docker Notary controls server access through strict authentication and open SSL security.
- Clients upload metadata for their container image.
- Once the data is verified, the server sends approval for the container image to be signed.
- Encrypted keys are used for the signature.
- Notary Server is the source of truth for the state of a trusted collection.
- Timestamps and checksums are used as part of the signature to prevent forgery.


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

DevSecOps Essentials

Section 2: Secure Software Onboarding

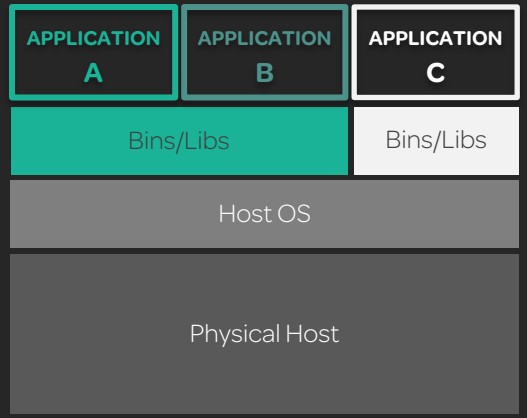
Lesson 5: Docker Bench

CIS Docker Benchmark

- Docker Bench is a software tool that analyzes server infrastructures and Docker components.
- Docker Bench uses the CIS Docker Benchmark to make recommendations on server hardening and proper Docker configuration.
- Since new vulnerabilities are created and discovered over time, it is important to consult the benchmarks as they are revised and updated.
- Most importantly, tools that scan and detect based on configuration files that use benchmarks must be updated to the most recent release in order to be effective.

Docker Bench

- In large enterprises, the engineering and operations departments create container images that include the necessary tooling for ongoing detection and remediation.
- Since new vulnerabilities often come from unprotected sources, teams must maintain certified repositories.
- Trusted registries are the optimal source, and teams must carry out careful onboarding when using unproven sources.
- Docker Bench can be run regularly to detect configuration flaws.



Host Configuration

- Docker Bench recommends host hardening.
- Avoid the common mistake of creating users within the Docker group. This known vulnerability allows privilege escalation on a spawned container process, which can enable a user to gain root access.
- Audit the file system against the Docker directories to receive early warning about unusual access or updates to system logs.

```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.4
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Community Edition Benchmark v1.1.0.
# ------------------------------------------------------------------------------

Initializing Tue Aug 21 10:37:03 EDT 2018


[INFO] 1 - Host Configuration
[WARN] 1.1  - Ensure a separate partition for containers has been created
[NOTE] 1.2  - Ensure the container host has been Hardened
[INFO] 1.3  - Ensure Docker is up to date
[INFO]      * Using 18.06.0, verify is it up to date as deemed necessary
[INFO]      * Your operating system vendor may provide support and security maintenance for Docker
[INFO] 1.4  - Ensure only trusted users are allowed to control Docker daemon
[INFO]      * docker:x:999:
[WARN] 1.5  - Ensure auditing is configured for the Docker daemon
[WARN] 1.6  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.7  - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[WARN] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[WARN] 1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.11 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[WARN] 1.12 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-containerd
[WARN] 1.13 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-runc
```

Docker Daemon Configuration

- Containers allow network bridging to be limited to only those ports necessary for the particular application within the container instance.
- Since SSL authentication has been deprecated, Docker Bench recommends that the Docker daemon be configured to use TLS (Transport Layer Security) authentication.
- When running a container, you can set a `ulimit` to ensure that the container is limited to a specified maximum number of open file descriptors.
- Containers should be prohibited from acquiring new privileges.

```
[INFO] 2 - Docker daemon configuration
[WARN] 2.1  - Ensure network traffic is restricted between containers on the default bridge
[PASS] 2.2  - Ensure the logging level is set to 'info'
[PASS] 2.3  - Ensure Docker is allowed to make changes to iptables
[PASS] 2.4  - Ensure insecure registries are not used
[PASS] 2.5  - Ensure aufs storage driver is not used
[INFO] 2.6  - Ensure TLS authentication for Docker daemon is configured
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.7  - Ensure the default ulimit is configured appropriately
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.8  - Enable user namespace support
[PASS] 2.9  - Ensure the default cgroup usage has been confirmed
[PASS] 2.10 - Ensure base device size is not changed until needed
[WARN] 2.11 - Ensure that authorization for Docker client commands is enabled
[WARN] 2.12 - Ensure centralized and remote logging is configured
[INFO] 2.13 - Ensure operations on legacy registry (v1) are Disabled (Deprecated)
[WARN] 2.14 - Ensure live restore is Enabled
[WARN] 2.15 - Ensure Userland Proxy is Disabled
[PASS] 2.16 - Ensure daemon-wide custom seccomp profile is applied, if needed
[PASS] 2.17 - Ensure experimental features are avoided in production
[WARN] 2.18 - Ensure containers are restricted from acquiring new privileges
```

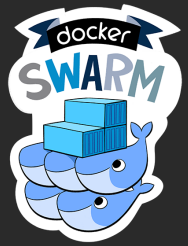
Docker Daemon Configuration (Continued)

- When configuring the Docker daemon, Linux file security must be strictly enforced.
- The Docker daemon runs as `root`, and while Linux allows processes to be configured such that their spawned child processes also run as `root`, this is a vulnerability.
- The 644 octet grants read/write privileges to the owner (in this case, `root`) but only read privileges to group- and user-level processes.
- Network sockets carry a similar security setting when permissions are set.

```
[INFO] 3 - Docker daemon configuration files
[PASS] 3.1  - Ensure that docker.service file ownership is set to root:root
[PASS] 3.2  - Ensure that docker.service file permissions are set to 644 or more restrictive
[PASS] 3.3  - Ensure that docker.socket file ownership is set to root:root
[PASS] 3.4  - Ensure that docker.socket file permissions are set to 644 or more restrictive
[PASS] 3.5  - Ensure that /etc/docker directory ownership is set to root:root
[PASS] 3.6  - Ensure that /etc/docker directory permissions are set to 755 or more restrictive
[INFO] 3.7  - Ensure that registry certificate file ownership is set to root:root
[INFO]      * Directory not found
[INFO] 3.8  - Ensure that registry certificate file permissions are set to 444 or more restrictive
[INFO]      * Directory not found
[INFO] 3.9  - Ensure that TLS CA certificate file ownership is set to root:root
[INFO]      * No TLS CA certificate found
[INFO] 3.10 - Ensure that TLS CA certificate file permissions are set to 444 or more restrictive
[INFO]      * No TLS CA certificate found
[INFO] 3.11 - Ensure that Docker server certificate file ownership is set to root:root
[INFO]      * No TLS Server certificate found
[INFO] 3.12 - Ensure that Docker server certificate file permissions are set to 444 or more restrictive
[INFO]      * No TLS Server certificate found
[INFO] 3.13 - Ensure that Docker server certificate key file ownership is set to root:root
[INFO]      * No TLS Key found
[INFO] 3.14 - Ensure that Docker server certificate key file permissions are set to 400
[INFO]      * No TLS Key found
[PASS] 3.15 - Ensure that Docker socket file ownership is set to root:docker
[PASS] 3.16 - Ensure that Docker socket file permissions are set to 660 or more restrictive
[INFO] 3.17 - Ensure that daemon.json file ownership is set to root:root
[INFO]      * File not found
[INFO] 3.18 - Ensure that daemon.json file permissions are set to 644 or more restrictive
[INFO]      * File not found
[PASS] 3.19 - Ensure that /etc/default/docker file ownership is set to root:root
[PASS] 3.20 - Ensure that /etc/default/docker file permissions are set to 644 or more restrictive
```

Docker Security Operations and Swarm Configuration

Docker Swarm is a native clustering solution from Docker that can be used for container orchestration.



The Docker Bench includes a review of Docker Swarm hardening recommendations.

Note the Docker Bench recommendations for maintaining secrets in a Swarm cluster.

```
[INFO] 6 - Docker Security Operations
[INFO] 6.1  - Avoid image sprawl
[INFO]      * There are currently: 1 images
[INFO] 6.2  - Avoid container sprawl
[INFO]      * There are currently a total of 1 containers, with 0 of them currently running


[INFO] 7 - Docker Swarm Configuration
[PASS] 7.1  - Ensure swarm mode is not Enabled, if not needed
[PASS] 7.2  - Ensure the minimum number of manager nodes have been created in a swarm (Swarm mode not enabled)
[PASS] 7.3  - Ensure swarm services are binded to a specific host interface (Swarm mode not enabled)
[PASS] 7.5  - Ensure Docker's secret management commands are used for managing secrets in a Swarm cluster (Swarm mode not enabled)
[PASS] 7.6  - Ensure swarm manager is run in auto-lock mode (Swarm mode not enabled)
[PASS] 7.7  - Ensure swarm manager auto-lock key is rotated periodically (Swarm mode not enabled)
[PASS] 7.8  - Ensure node certificates are rotated as appropriate (Swarm mode not enabled)
[PASS] 7.9  - Ensure CA certificates are rotated as appropriate (Swarm mode not enabled)
[PASS] 7.10 - Ensure management plane traffic has been separated from data plane traffic (Swarm mode not enabled)
```

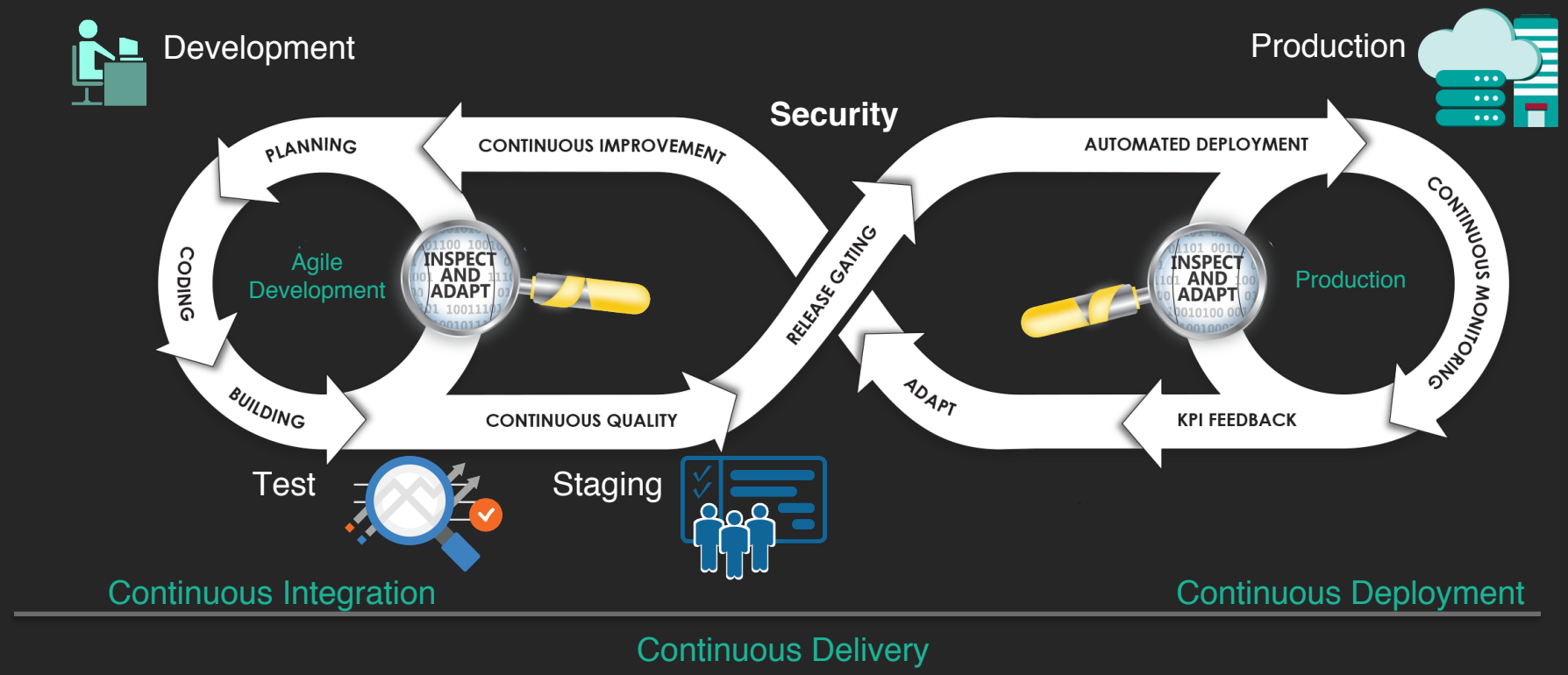
Container Images and Build File

- The Docker content trust system uses The Update Framework (TUF) to enforce content trust.
- Container image signing and verification policies can be set to allow the use of only trusted repositories.
- The following commands can be configured to require content trust:

```
docker pull
docker push
docker build
docker create
docker run
```

```
[INFO] 4 - Container Images and Build File
[INFO] 4.1  - Ensure a user for the container has been created
[INFO]      * No containers running
[NOTE] 4.2  - Ensure that containers use trusted base images
[NOTE] 4.3  - Ensure unnecessary packages are not installed in the container
[NOTE] 4.4  - Ensure images are scanned and rebuilt to include security patches
[WARN] 4.5  - Ensure Content trust for Docker is Enabled
[WARN] 4.6  - Ensure HEALTHCHECK instructions have been added to the container image
[WARN]      * No Healthcheck found: [hello-world:latest]
[PASS] 4.7  - Ensure update instructions are not use alone in the Dockerfile
[NOTE] 4.8  - Ensure setuid and setgid permissions are removed in the images
[PASS] 4.9  - Ensure COPY is used instead of ADD in Dockerfile
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles
[NOTE] 4.11 - Ensure verified packages are only Installed


[INFO] 5  - Container Runtime
[INFO]      * No containers running, skipping Section 5
```


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

DevSecOps Essentials

Section 2:Secure Software Onboarding
Understanding Secure Containerization

2.3 Understanding Secure Containerization

Containers

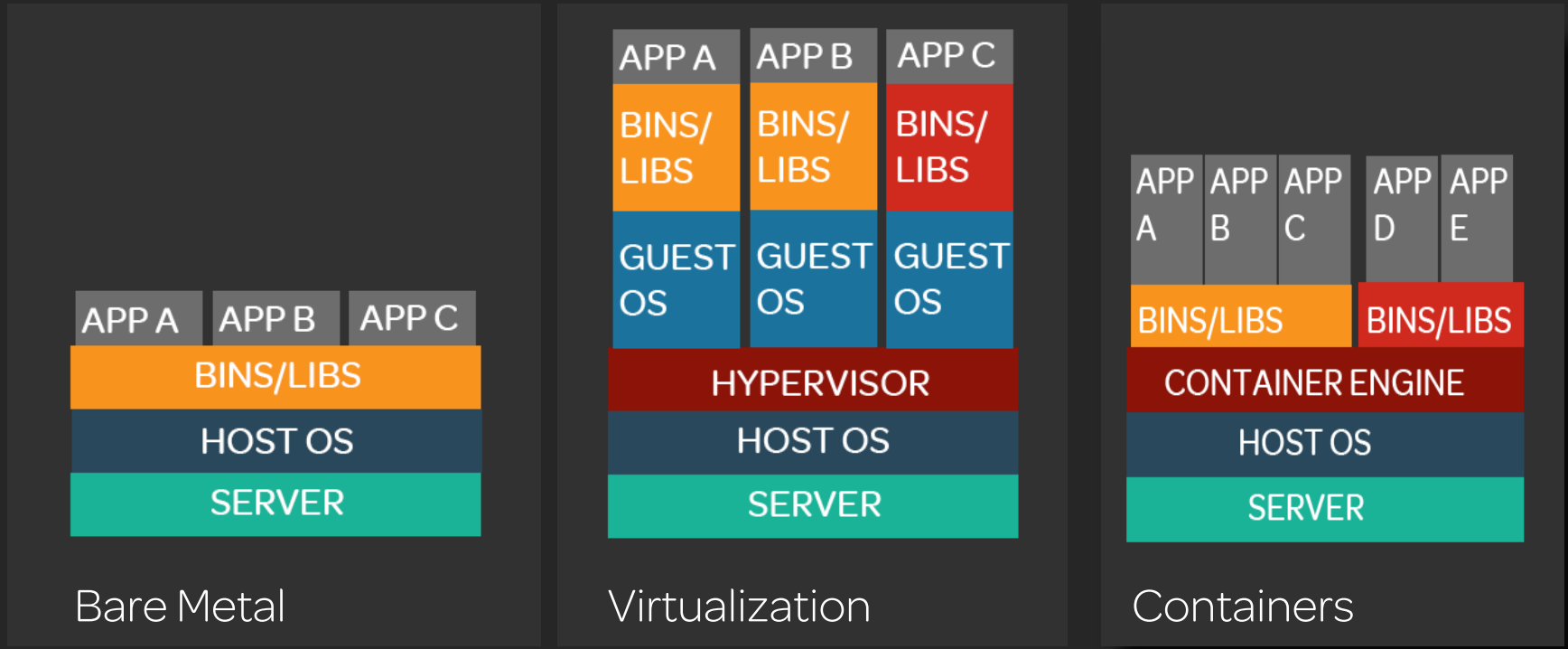
- Containers are specially formatted files that contain executable application programs, modules, and the code libraries that they depend upon.
- One benefit of using containers is to isolate an application from other applications and from the operating system and hardware.
- This isolation helps ensure that the versions of libraries within the container do not mandate patching and upgrades that might affect other applications in other containers.
- In virtual environments, the libraries and dependencies of an application were oftentimes shared with all of the other applications on a particular virtual server.

2.3 Understanding Secure Containerization

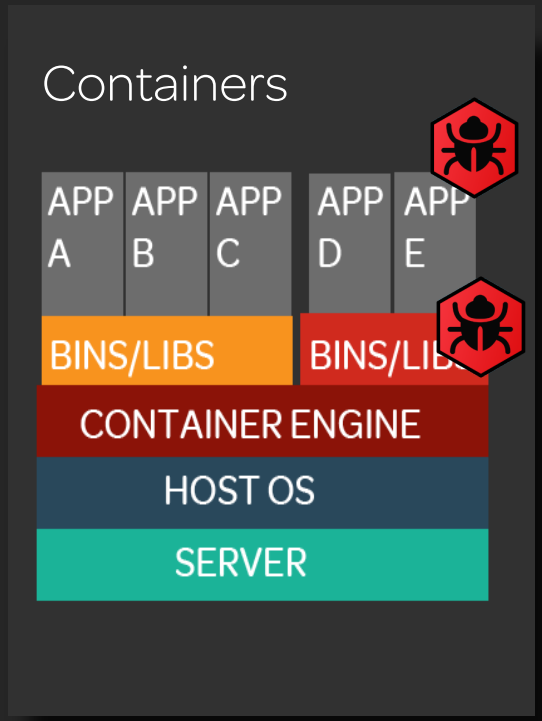
Use Case: Docker

- While there are many container engines to choose from, our use case is Docker.
- Process Restrictions use root/non-root dichotomy. This makes it difficult to provoke system level damages during intrusion, even if the intruder manages to escalate to root within a container because the container capabilities are fundamentally restricted.
- Device and File Restrictions further reduces the attack surface by restricting access by containerized applications to the physical devices on a host, through the use of the device resource control groups (*cgroups*) mechanism.
- Container images are ephemeral. Each container has its own file system and can not write to the host file system unless writes are committed. Committing changes tracks and audits changes made to base images as a new layer which can then be pushed as a new image for storage in Docker Hub and run in a container.
- The audit trail is important in providing information to maintain compliance. It also allows for fast and easy rollback to previous versions, if a container has been compromised or a vulnerability introduced.

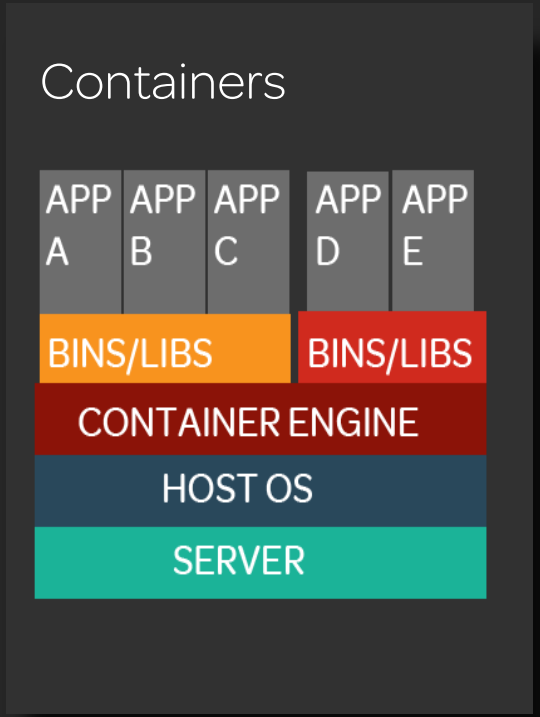
2.3 Understanding Secure Containerization



2.3 Understanding Secure Containerization

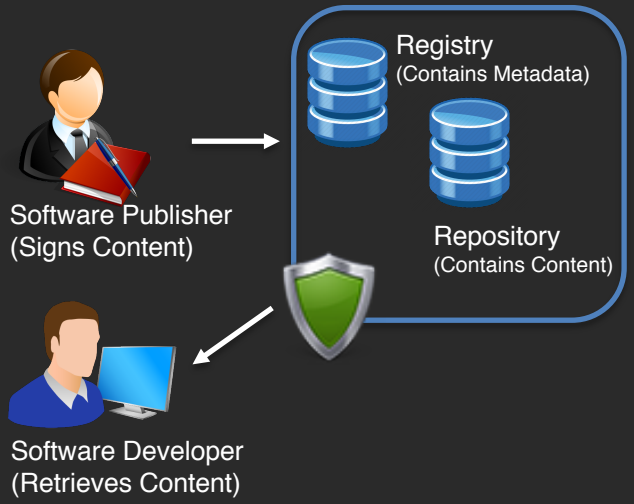


2.3 Understanding Secure Containerization



2.3 Understanding Secure Containerization

Trusted Registries



Understanding Trusted Content and Digital Signatures

Trusted Registries are managed repositories that are secure.

A registry is simply a data store that contains metadata (ie. Data about data).

A repository contains the actual content of interest.

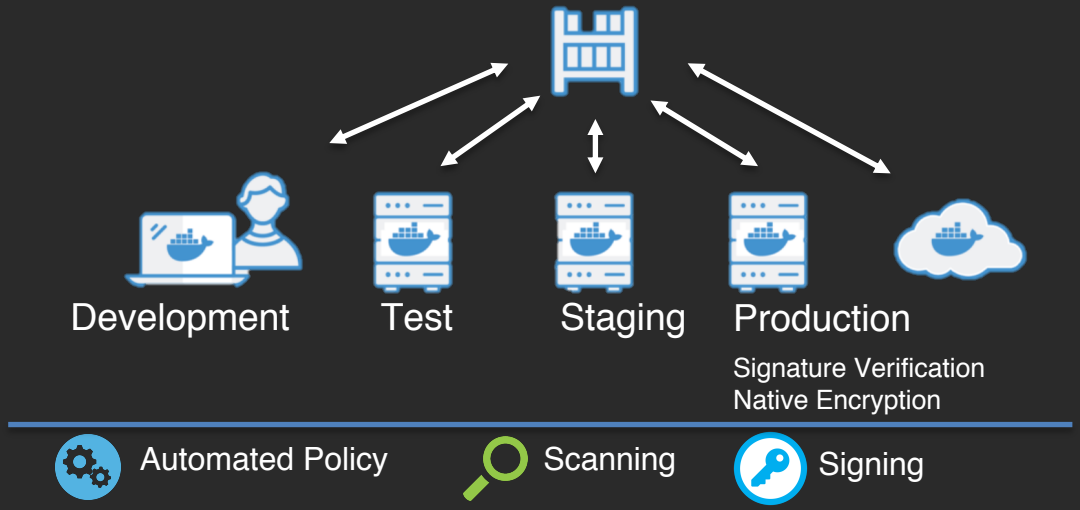
Trusted registries utilize a digital signature to identify content in the repository that is safe.

Software publishers digitally sign their content when it is published.

Developers may utilize systems that will verify signatures prior to downloaded content.

2.3 Understanding Secure Containerization

Docker Trusted Repository



2.3 Understanding Secure Containerization

Docker Trusted Repository

Transport Layer Security (**TLS**) – and its predecessor, Secure Sockets Layer (SSL), which is now deprecated by the Internet Engineering Task Force (IETF) – are cryptographic protocols that provide communications security over a computer network.

Certificates and Public PKI...

https://www.docker.com/resources/security/

2.3 Understanding Secure Containerization

Docker Hub: The World's Largest Community of Container Images

Docker Hub is the world's largest public repository of container images with an array of content sources including container community developers, open source projects and independent software vendors (ISV) building and distributing their code in containers.

Docker Hub allows you to:

- Search and browse for millions of container images
- View image popularity, vendor source and certification to inform your selection
- Access free public repositories to store your images and share with the community
- Choose a subscription plans for private repositories to limit access to your images
- Use Autobuilds and Webhooks for easy integration into your DevOps pipeline

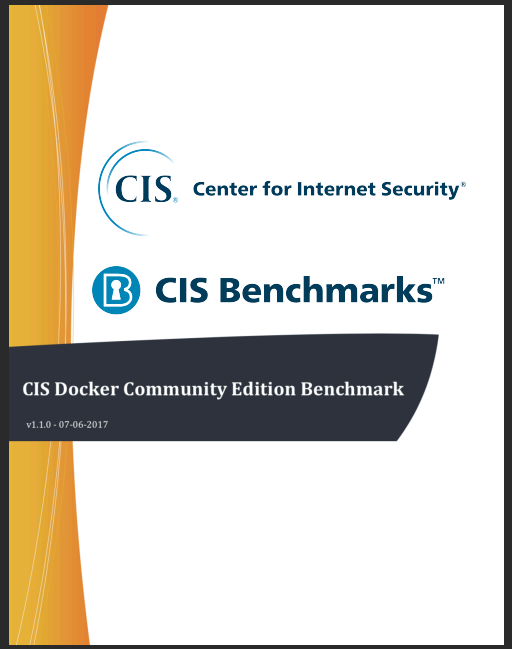
2.3 Understanding Secure Containerization

Docker Hub

- Explain...
- Public, private and official images...
- Official repositories are certified repositories from Independent Software Vendors (ISV) and Docker contributors like Canonical, Oracle, RedHat and many more.

2.3 Understanding Secure Containerization

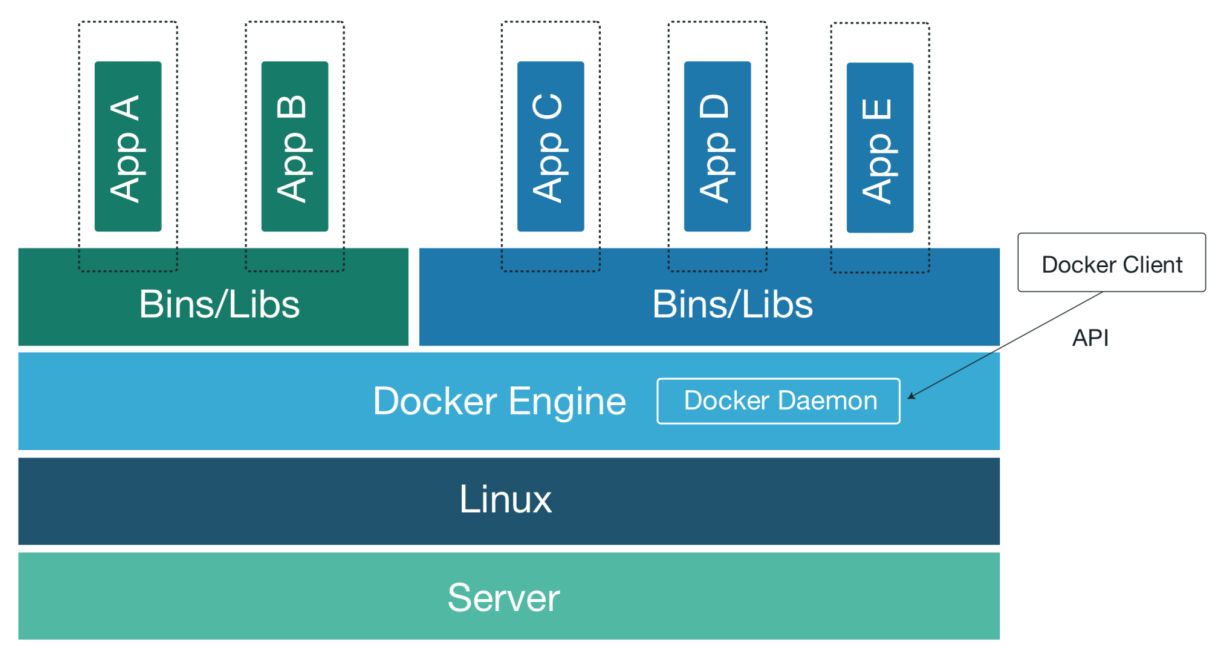
Docker Bench



2.3 Understanding Secure Containerization

Docker Bench

Auditing...



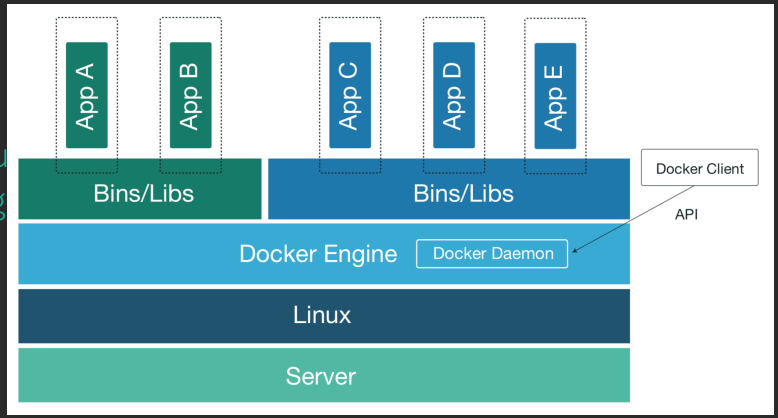
2.3 Understanding Secure Containerization

Docker Bench

Transport Layer Security (**TLS**) – and its predecessor, Secu
by the Internet Engineering Task Force (IETF) – are cryptog
security over a computer network.

Certificates and Public PKI...

https://www.docker.com/resources/security/



2.3 Understanding Secure Containerization

Docker Bench

Auditing...

```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.4
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Community Edition Benchmark v1.1.0.
# ------------------------------------------------------------------------------

Initializing Tue Aug 21 10:37:03 EDT 2018


[INFO] 1 - Host Configuration
[WARN] 1.1  - Ensure a separate partition for containers has been created
[NOTE] 1.2  - Ensure the container host has been Hardened
[INFO] 1.3  - Ensure Docker is up to date
[INFO]      * Using 18.06.0, verify is it up to date as deemed necessary
[INFO]      * Your operating system vendor may provide support and security maintenance for Docker
[INFO] 1.4  - Ensure only trusted users are allowed to control Docker daemon
[INFO]      * docker:x:999:
[WARN] 1.5  - Ensure auditing is configured for the Docker daemon
[WARN] 1.6  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.7  - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[WARN] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[WARN] 1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.11 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[WARN] 1.12 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-containerd
[WARN] 1.13 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-runc
```

2.3 Understanding Secure Containerization

Docker Bench

Docker Daemon

```
[INFO] 2 - Docker daemon configuration
[WARN] 2.1  - Ensure network traffic is restricted between containers on the default bridge
[PASS] 2.2  - Ensure the logging level is set to 'info'
[PASS] 2.3  - Ensure Docker is allowed to make changes to iptables
[PASS] 2.4  - Ensure insecure registries are not used
[PASS] 2.5  - Ensure aufs storage driver is not used
[INFO] 2.6  - Ensure TLS authentication for Docker daemon is configured
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.7  - Ensure the default ulimit is configured appropriately
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.8  - Enable user namespace support
[PASS] 2.9  - Ensure the default cgroup usage has been confirmed
[PASS] 2.10 - Ensure base device size is not changed until needed
[WARN] 2.11 - Ensure that authorization for Docker client commands is enabled
[WARN] 2.12 - Ensure centralized and remote logging is configured
[INFO] 2.13 - Ensure operations on legacy registry (v1) are Disabled (Deprecated)
[WARN] 2.14 - Ensure live restore is Enabled
[WARN] 2.15 - Ensure Userland Proxy is Disabled
[PASS] 2.16 - Ensure daemon-wide custom seccomp profile is applied, if needed
[PASS] 2.17 - Ensure experimental features are avoided in production
[WARN] 2.18 - Ensure containers are restricted from acquiring new privileges
```

2.3 Understanding Secure Containerization

Docker Bench

Daemon Configuration

```
[INFO] 3 - Docker daemon configuration files
[PASS] 3.1  - Ensure that docker.service file ownership is set to root:root
[PASS] 3.2  - Ensure that docker.service file permissions are set to 644 or more restrictive
[PASS] 3.3  - Ensure that docker.socket file ownership is set to root:root
[PASS] 3.4  - Ensure that docker.socket file permissions are set to 644 or more restrictive
[PASS] 3.5  - Ensure that /etc/docker directory ownership is set to root:root
[PASS] 3.6  - Ensure that /etc/docker directory permissions are set to 755 or more restrictive
[INFO] 3.7  - Ensure that registry certificate file ownership is set to root:root
[INFO]      * Directory not found
[INFO] 3.8  - Ensure that registry certificate file permissions are set to 444 or more restrictive
[INFO]      * Directory not found
[INFO] 3.9  - Ensure that TLS CA certificate file ownership is set to root:root
[INFO]      * No TLS CA certificate found
[INFO] 3.10 - Ensure that TLS CA certificate file permissions are set to 444 or more restrictive
[INFO]      * No TLS CA certificate found
[INFO] 3.11 - Ensure that Docker server certificate file ownership is set to root:root
[INFO]      * No TLS Server certificate found
[INFO] 3.12 - Ensure that Docker server certificate file permissions are set to 444 or more restrictive
[INFO]      * No TLS Server certificate found
[INFO] 3.13 - Ensure that Docker server certificate key file ownership is set to root:root
[INFO]      * No TLS Key found
[INFO] 3.14 - Ensure that Docker server certificate key file permissions are set to 400
[INFO]      * No TLS Key found
[PASS] 3.15 - Ensure that Docker socket file ownership is set to root:docker
[PASS] 3.16 - Ensure that Docker socket file permissions are set to 660 or more restrictive
[INFO] 3.17 - Ensure that daemon.json file ownership is set to root:root
[INFO]      * File not found
[INFO] 3.18 - Ensure that daemon.json file permissions are set to 644 or more restrictive
[INFO]      * File not found
[PASS] 3.19 - Ensure that /etc/default/docker file ownership is set to root:root
[PASS] 3.20 - Ensure that /etc/default/docker file permissions are set to 644 or more restrictive
```

2.3 Understanding Secure Containerization

Docker Bench

Security Operations and Swarm Configuration

```
[INFO] 6 - Docker Security Operations
[INFO] 6.1  - Avoid image sprawl
[INFO]      * There are currently: 1 images
[INFO] 6.2  - Avoid container sprawl
[INFO]      * There are currently a total of 1 containers, with 0 of them currently running


[INFO] 7 - Docker Swarm Configuration
[PASS] 7.1  - Ensure swarm mode is not Enabled, if not needed
[PASS] 7.2  - Ensure the minimum number of manager nodes have been created in a swarm (Swarm mode not enabled)
[PASS] 7.3  - Ensure swarm services are binded to a specific host interface (Swarm mode not enabled)
[PASS] 7.5  - Ensure Docker's secret management commands are used for managing secrets in a Swarm cluster (Swarm mode not enabled)
[PASS] 7.6  - Ensure swarm manager is run in auto-lock mode (Swarm mode not enabled)
[PASS] 7.7  - Ensure swarm manager auto-lock key is rotated periodically (Swarm mode not enabled)
[PASS] 7.8  - Ensure node certificates are rotated as appropriate (Swarm mode not enabled)
[PASS] 7.9  - Ensure CA certificates are rotated as appropriate (Swarm mode not enabled)
[PASS] 7.10 - Ensure management plane traffic has been separated from data plane traffic (Swarm mode not enabled)
```

2.3 Understanding Secure Containerization

Docker Bench

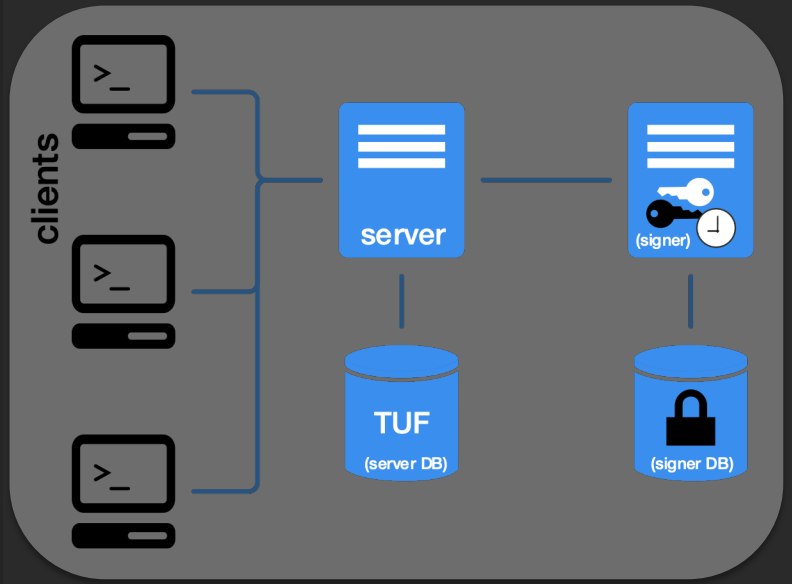
Container Images and Build File

```
[INFO] 4 - Container Images and Build File
[INFO] 4.1  - Ensure a user for the container has been created
[INFO]      * No containers running
[NOTE] 4.2  - Ensure that containers use trusted base images
[NOTE] 4.3  - Ensure unnecessary packages are not installed in the container
[NOTE] 4.4  - Ensure images are scanned and rebuilt to include security patches
[WARN] 4.5  - Ensure Content trust for Docker is Enabled
[WARN] 4.6  - Ensure HEALTHCHECK instructions have been added to the container image
[WARN]      * No Healthcheck found: [hello-world:latest]
[PASS] 4.7  - Ensure update instructions are not use alone in the Dockerfile
[NOTE] 4.8  - Ensure setuid and setgid permissions are removed in the images
[PASS] 4.9  - Ensure COPY is used instead of ADD in Dockerfile
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles
[NOTE] 4.11 - Ensure verified packages are only Installed


[INFO] 5  - Container Runtime
[INFO]      * No containers running, skipping Section 5
```

2.3 Understanding Secure Containerization

Docker Notary



Architecture and components

Notary clients pull metadata from one or more (remote) Notary services.

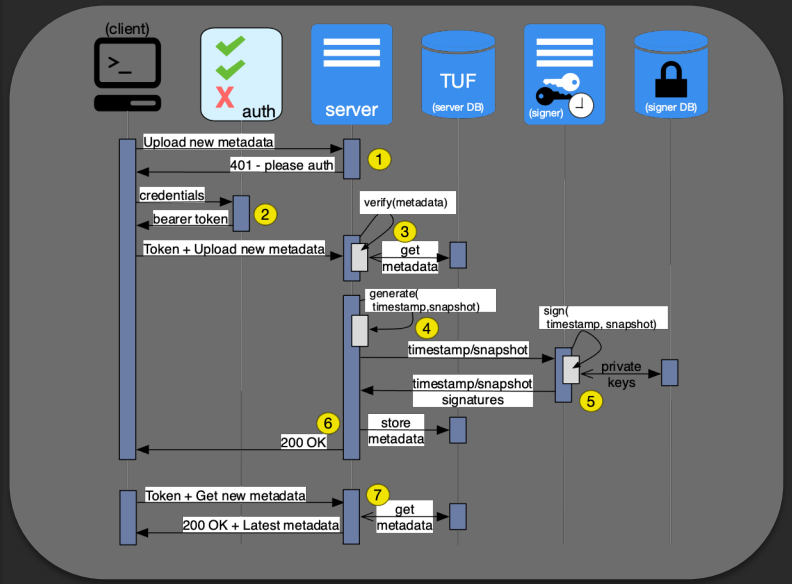
Some Notary clients will push metadata to one or more Notary services.

A Notary service consists of a Notary server, which stores and updates the signed TUF metadata files for multiple trusted collections in an associated database.

A Notary signer stores private keys for and signs metadata for the Notary server.

2.3 Understanding Secure Containerization

Docker Notary



Using Private Keys To Sign Containers

Notary controls server access through strict authentication and open SSL security.

Clients upload metadata for their container image.

Once the data is verified the server sends the approval for the container image to be signed.

Encrypted keys are used for the signature.

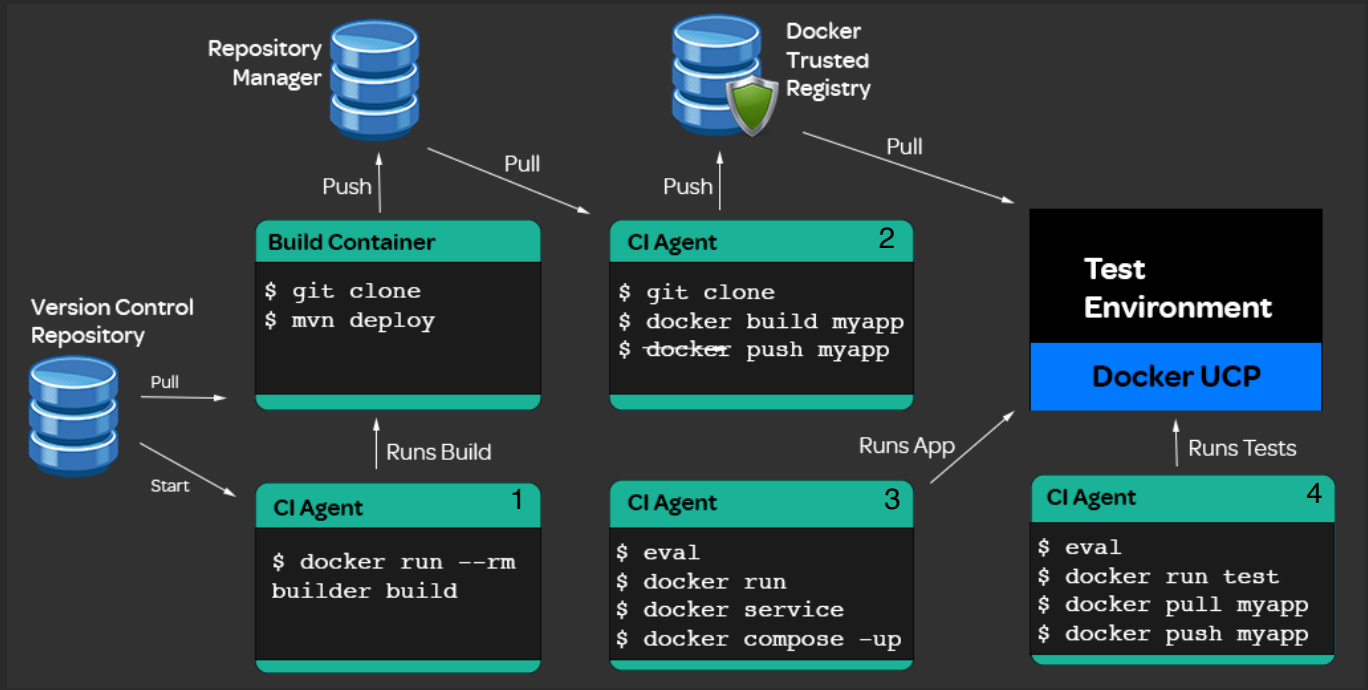
Notary server is the source of truth for the state of a trusted collection.

Timestamps and checksums are used as part of the signature to prevent forgery.

2.3 Understanding Secure Containerization

Docker CI/CD Workflow

1. Build Application
2. Build Image
3. Deploy Application
4. Test Application



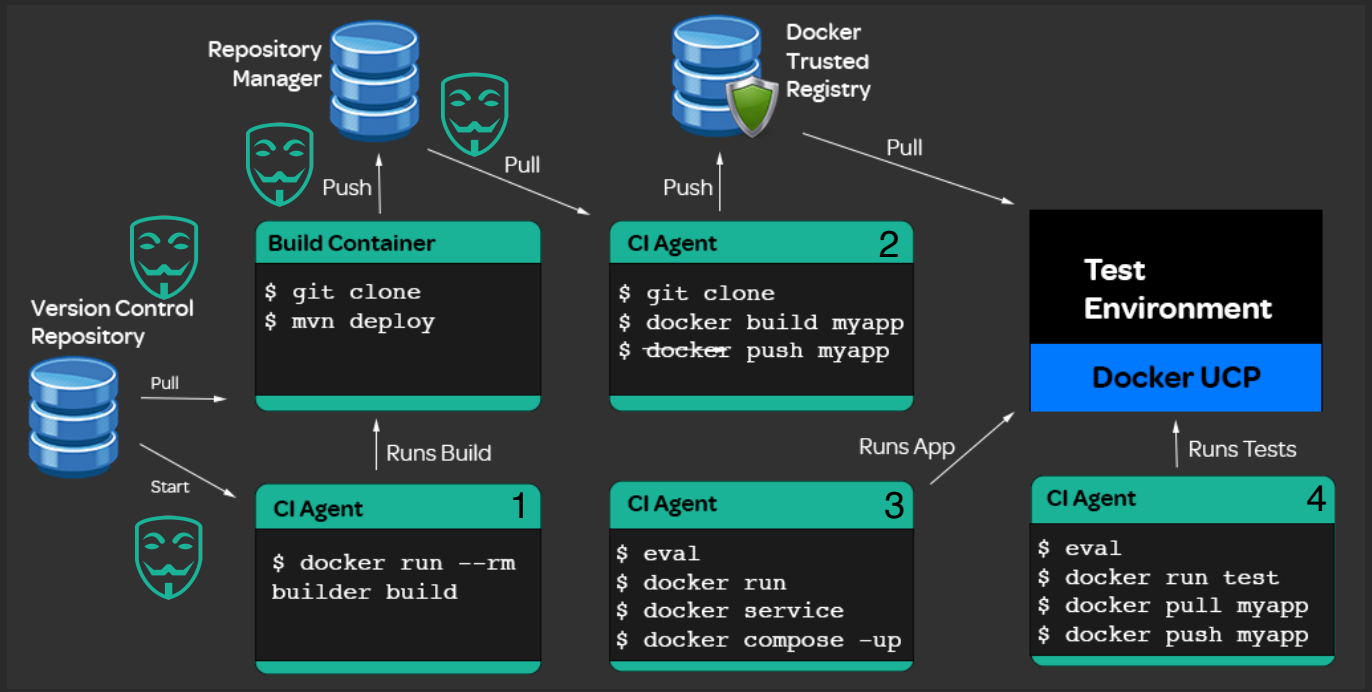
https://success.docker.com/article/dev-pipeline

2.3 Understanding Secure Containerization

Docker CI/CD Workflow

1. Build Application
2. Build Image
3. Deploy Application
4. Test Application

= Attack Vectors



DevSecOps Essentials

Section 1: Introduction

Lesson 2: Cyber Security Concepts and Standards

Introductory Remarks

- Cyber security is a vast field of study.
- Qualified security professionals have many years of experience combined with many certifications.
- This section of the course is *not* a comprehensive discussion of this complex topic.
- The concepts, terms, and standards provided in this video are intended to give you a foundational introduction to the DevSecOps challenge.
- If you hope to specialize in a security-related discipline, I encourage you to pursue further study beyond this course.

Attack Surface

- The **attack surface** of a system is the collection of points (**attack vectors**) where an unauthorized user (**attacker**) may enter to inject data to or extract data from an environment.
- Keeping the attack surface as small as possible is a basic security measure.

Malware and Vulnerabilities

- **Malware** is malicious software that attackers deploy to infect individual computers or an entire digital network.
- Malware exploits target system vulnerabilities that can be hijacked, such as bugs in legitimate software (e.g., browser or web application plugins).
- A **vulnerability** is a weakness or deficiency in a computer system that an attacker can exploit to perform unauthorized actions within the system.

Identifying and Scoring Vulnerabilities

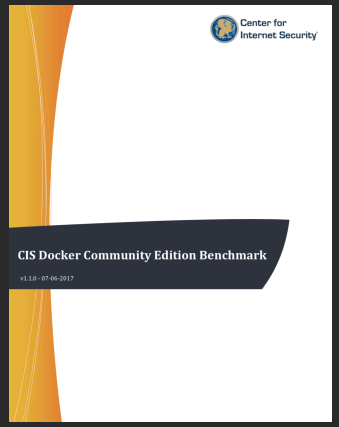
- Common Vulnerabilities and Exposures (CVE) is a dictionary-style list of standardized names for vulnerabilities and other information related to security exposures.
- CVE aims to standardize the names of all publicly known vulnerabilities and security exposures.
- The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of security vulnerabilities.
- CVSS assigns severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat level.

The OpenSCAP Project

- The Security Content Automation Protocol (SCAP) is a U.S. security standard maintained by the National Institute of Standards and Technology (NIST).
- The OpenSCAP Project is a collection of open-source tools for implementing and enforcing this standard.

The Center for Internet Security (CIS)

- The Center for Internet Security (CIS) provides security benchmarks and the National Checklist Program (NCP), defined by the NIST.
- They offer guidance on the security configurations of the operating system, database, virtualization, framework, and applications.
- In addition to the benchmark documents, the CIS also provides downloadable tools for secure configuration scanning.



Malware and Vulnerability Scanners

- Scanners can be deployed to network hosts and run in memory-resident mode to monitor activity in real time.
- By monitoring multiple sensor points, scanners can log vulnerabilities so that DevSecOps stakeholders become aware of the need for remedial action.
- Scanners should be used to interrogate new and existing software to determine whether any system or application may be infected by threat actors or attackers.
- There are two types of scanning: dynamic scanning and static scanning.

Dynamic vs. Static Scanning

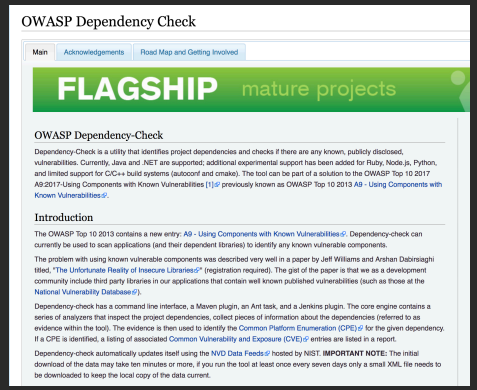
- **Dynamic scanning** is a method of code analysis that identifies vulnerabilities in a *runtime* environment.
- Dynamic tests monitor system memory, functional behavior, response time, and the overall performance of the system.
- Automated scanning tools can be used to analyze applications for which you do not have access to the original source code.
- **Static scanning** is a method of analysis performed in a *non-runtime* environment.
- Typically, a static analysis tool will inspect program code for all possible runtime behaviors and seek out flaws and potentially vulnerable code.
- Although they were developed separately, static and dynamic scanning are *not* in opposition to one another.
- There are strengths and weaknesses associated with both approaches, and many DevSecOps processes benefit from using both.

The Cloud Controls Matrix (CCM)

- The Cloud Security Alliance (CSA) has consolidated most security compliance methods into a single resource called the Cloud Controls Matrix (CCM).
- The CCM includes all security compliance controls such as ISO, FedRAMP, and NIST 800-53.
- It defines the control ID used to uniquely identify vulnerabilities.
- The key benefit of the CCM is that we can consult one aggregate source of security compliance regulations.
- The CSA also provides the Consensus Assessments Initiative Questionnaire (CAIQ). This questionnaire is a means of security self-assessment for both cloud consumers and providers.

The Open Web Application Security Project (OWASP)

- The **Open Web Application Security Project (OWASP)** is an online community project that provides free articles, methodologies, documentation, tools, and technologies for web application security.
- The OWASP Foundation came online on December 1st, 2001.
- It was established as a not-for-profit charitable organization in the United States on April 21, 2004.
- OWASP is an international organization, and the OWASP Foundation supports OWASP efforts around the world.



Federal Information Processing Standards (FIPS)

- The FIPS are security standards developed by the U.S. federal government for use in non-military government computer systems.
- FIPS defines minimum security requirements for the use of cryptographic modules.
- The FIPS publication *Security Requirements for Cryptographic Modules* explains which cryptographic modules are considered safe, legacy, or weak.
- To learn more about cryptographic modules, check out the following resources:
  - https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet
  - https://www.owasp.org/index.php/Guide_to_Cryptography
  - https://www.owasp.org/index.php/Key_Management_Cheat_Sheet

DevSecOps Resources

- The National Checklist Program (NCP) repository provides secure configuration for specific software components.
- DevSecOps stakeholders can use the NCP repository to search the CIS for information on particular software products.
- The NCP repository provides metadata and links to security checklists, including checklists that conform to the SCAP.
- The NIST provides an online search tool as well as data feeds that are used by many SCAP-validated tools to provide intelligence for advanced DevSecOps monitoring.
- The National Vulnerability Database (NVD) is a term security professionals use to refer to this NIST repository and other data stores provided by the NIST.

DevSecOps Essentials

Section 3: Secure Build Automation

Lesson 1: Provisioning with PaaS Tooling

Provisioning

- **Provisioning** means providing something for use.
- In DevSecOps, the process of providing the developer with the software needed to develop an application is automated.
- It's important to ensure that the same software the developers use is provisioned in the test, staging, and production environments.
- Complex systems require automation to ensure that provisioned processes and components remain uniform and managed.

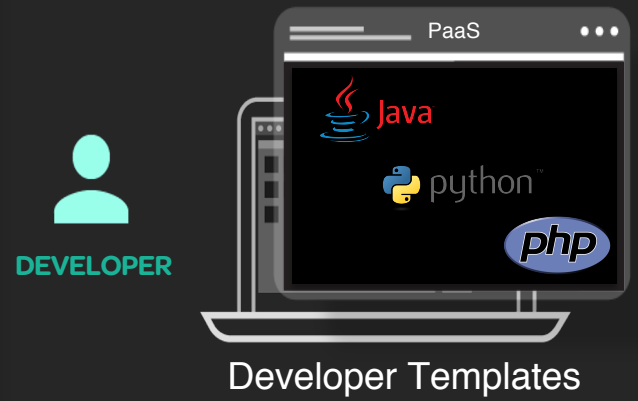
Uniformity

Platform as a Service (PaaS)

- The goal of Agile development is to give developers greater autonomy.
- **Shift left** is a process change in which operations and engineering teams shift processes upstream to developers.
- Anything "as a service" allows consumers to request and receive deliverables on demand via a self-service portal.
- **Platform as a Service (PaaS)** offerings provide developer platforms and infrastructures via self-service tools and automation.

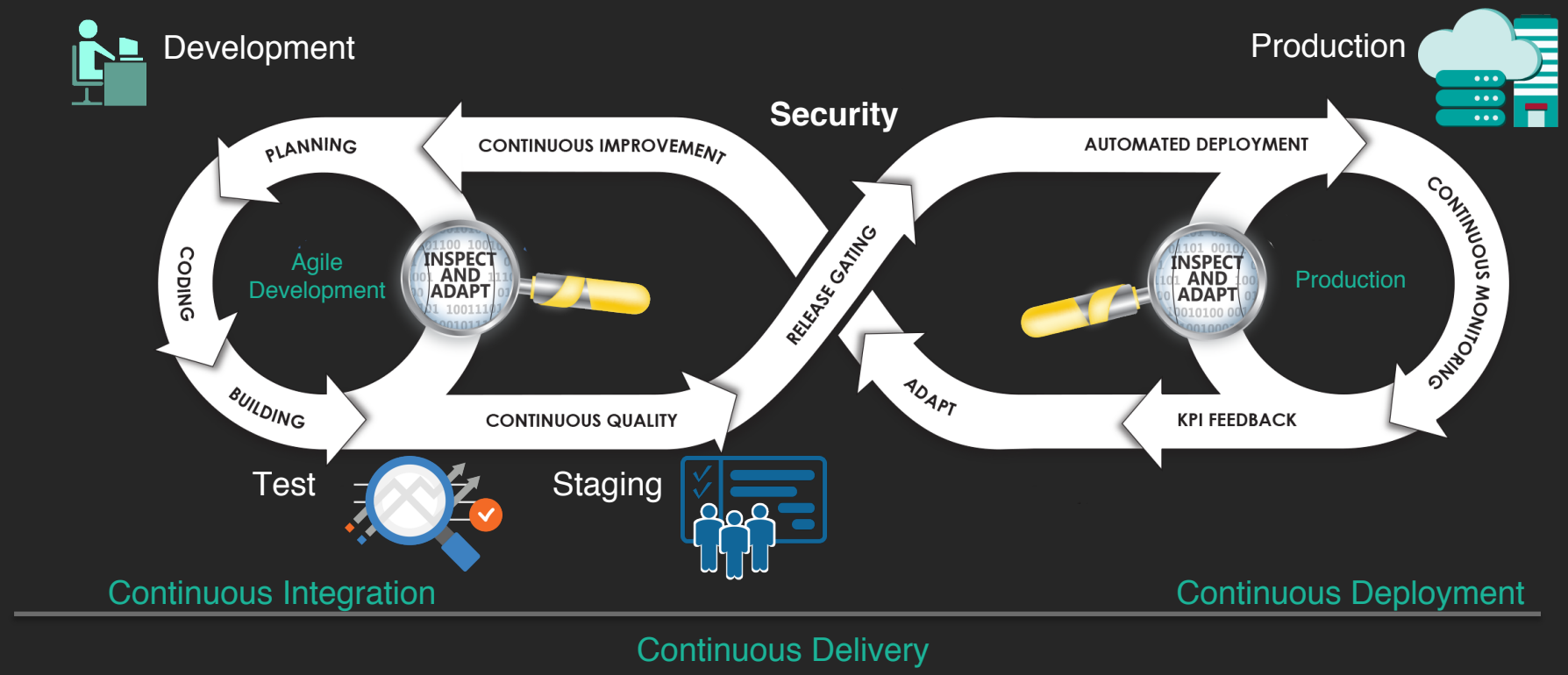
Templates

- Enterprise architects evaluate and test software components for production use.
- PaaS systems facilitate the development of templates (predefined configurations that comply with policy).
- When developers require specific frameworks and infrastructures, they can select from available templates.
- Infrastructure as Code simplifies the configuration and deployment process and reduces maintenance.
- Source to Image (S2I) is a framework for building reproducible container images from application source code and dependent component libraries.



OpenShift

- **OpenShift** is an open-source PaaS offering from Red Hat.
- As containers and clustering have grown in popularity, OpenShift has adopted Docker and Kubernetes as standards.
- Now that OpenShift can be used to deploy and manage production environments, it is considered a **container orchestration** tool instead of a PaaS.
- OpenShift is a fully automated container orchestration system that meets developer provisioning needs and automates S2I as workloads are promoted from development to test, staging, and production.


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

DevSecOps Essentials

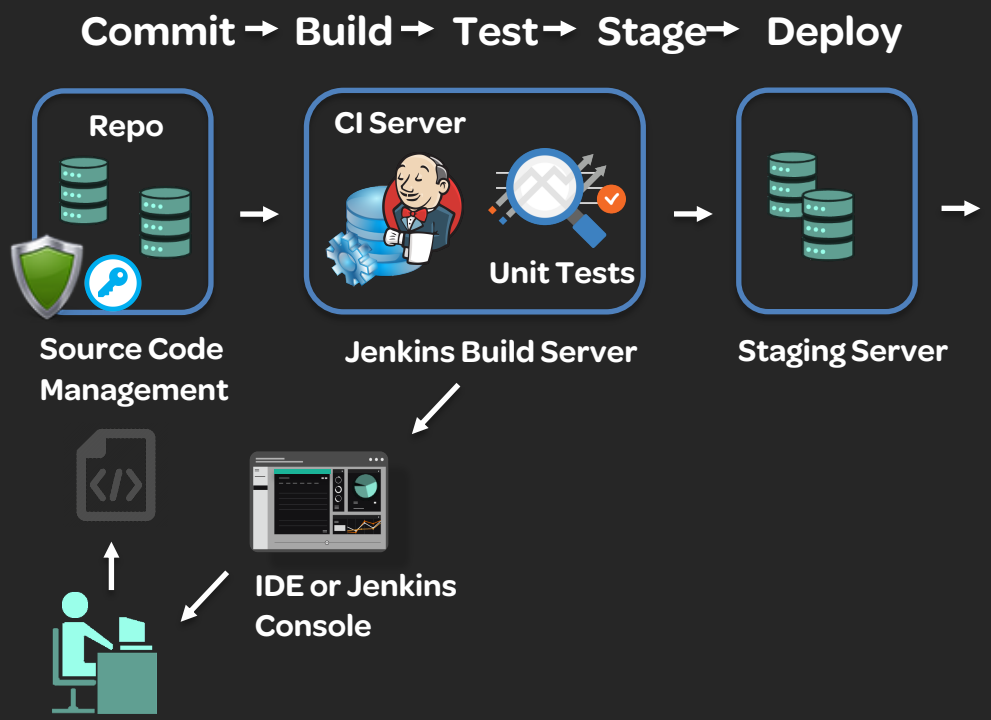
Section 3: Secure Build Automation

Lesson 2: Securing the Automated Build

Jenkins

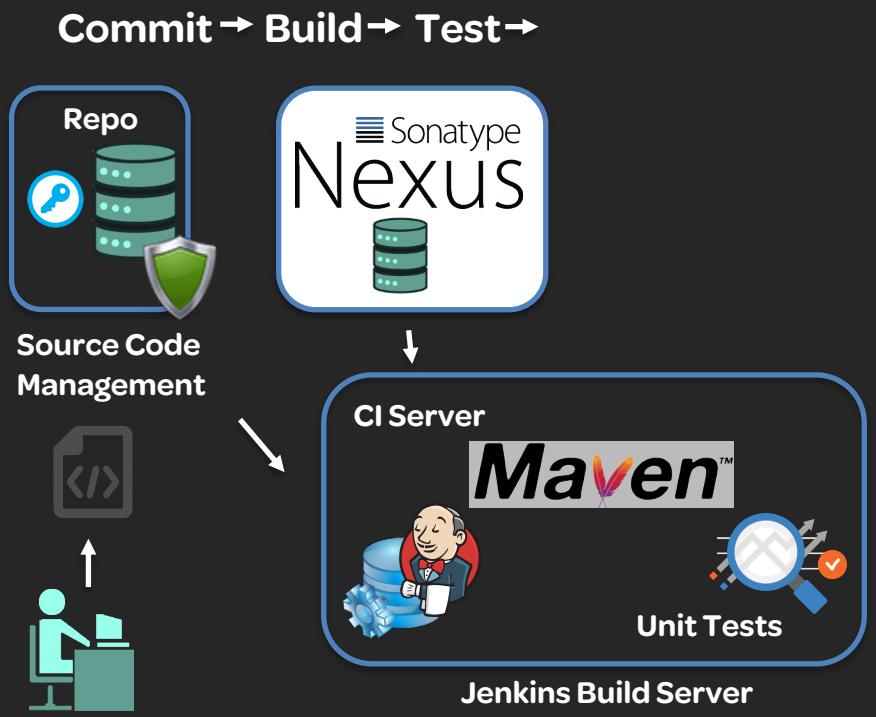
- **Jenkins** is a popular open-source build automation system.
- Jenkins originated from the open-source Hudson project.
- Jenkins is highly extensible and supports hundreds of plugins that handle build tasks.
- Jenkins supports Continuous Integration by integrating with source code management tools.
- Jenkins supports **webhooks** that automatically trigger a build process when developers check in modified application source code.

Typical CI/CD Pipeline Architecture (Using Jenkins)



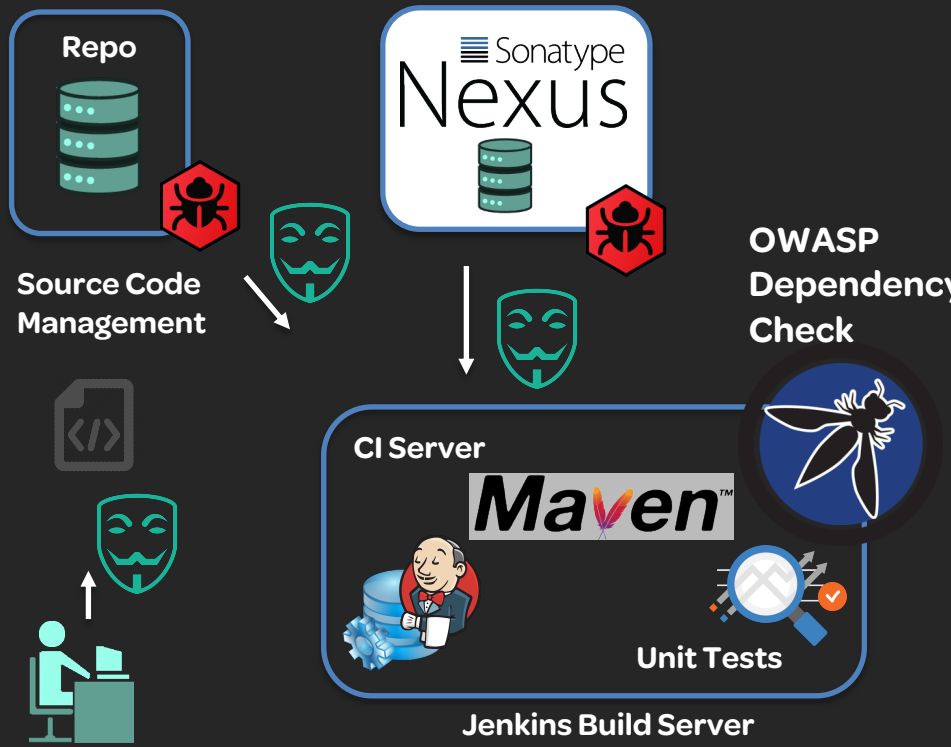
- A **commit** starts the process when the developer checks in code.
- The CI server launches a build process.
- Automated unit tests are typically part of the build process.
- Developers may review the build status via their IDE or the Jenkins console.
- The build server delivers build artifacts to the appropriate target, typically a staging server.

CI/CD Pipeline Architecture (Using Jenkins with Maven)

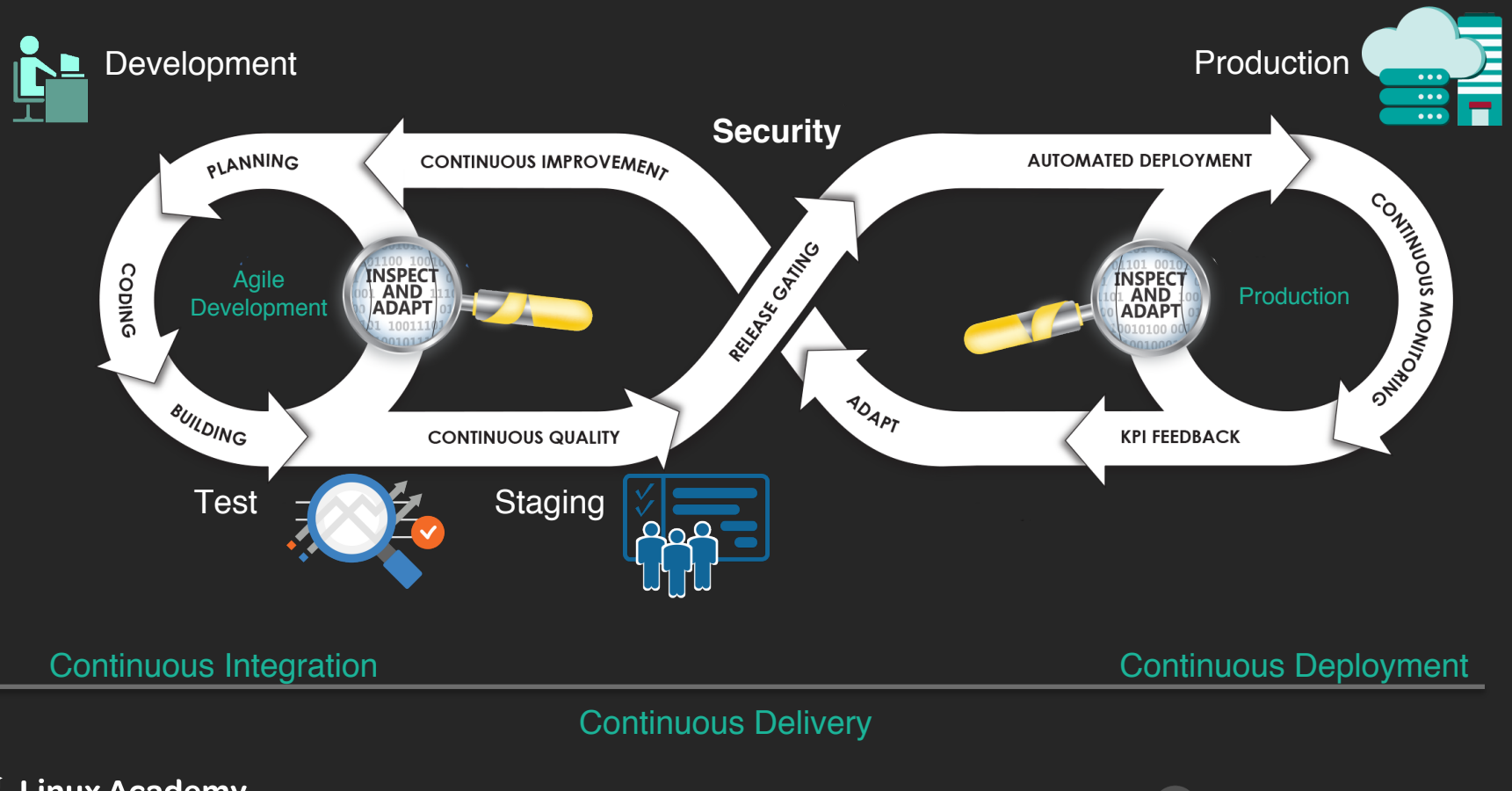


- **Java** builds typically involve a repo that contains needed component libraries.
- Apache **Maven** is an open-source tool often used by Java teams.
- When Maven performs a build, it automatically downloads libraries from **Maven Central**.
- Maven Central is a public repository maintained by Sonatype.
- Most enterprises configure a repository inside their firewall using products like **Nexus**.

Attack Surface of a Typical CI/CD Build Environment



- **Attack vectors** exist at the point of code check-in and anywhere an API session is established to pull code.
- **Malware** and vulnerabilities exist in most component libraries pulled from third-party repos.
- Since it is necessary to use vulnerable code, vulnerability detection and monitoring must be done after the build process.
- The **OWASP Dependency Checker** is one open-source tool that detects and reveals vulnerabilities and their severity levels.


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

DevSecOps Essentials

Section 3: Secure Build Automation

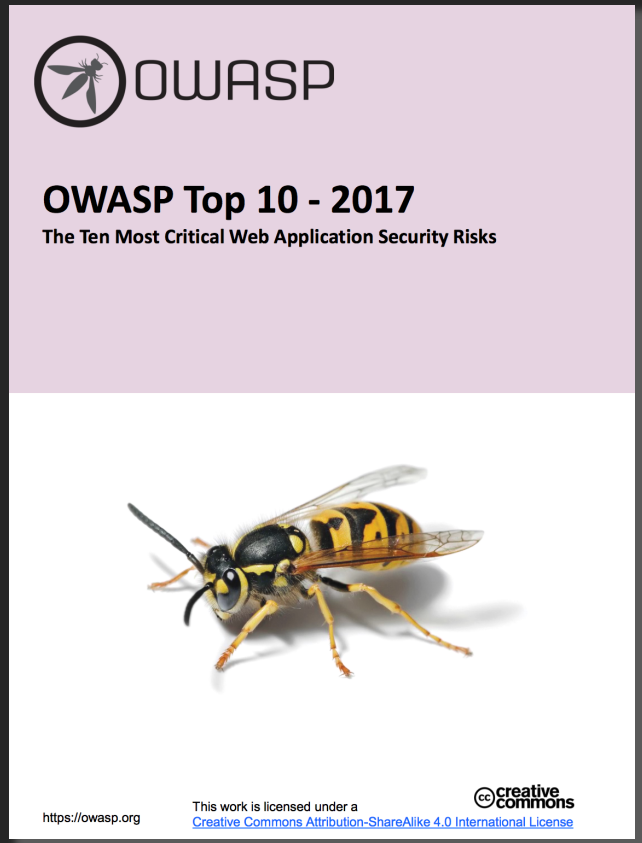
Lesson 3: Vulnerability Detection and Remediation

OWASP Dependency Checker

- **OWASP Dependency Check** is an open-source scanning system.
- Dependency Check can be run from the command line or through an automated build program such as Jenkins.
- OWASP Dependency Check maintains a downloaded CVE database from the NIST NVD.
- OWASP Dependency Check provides human-readable and machine-readable reports.
- To enforce security policy, automated builds can be configured to fail when policy thresholds are exceeded.

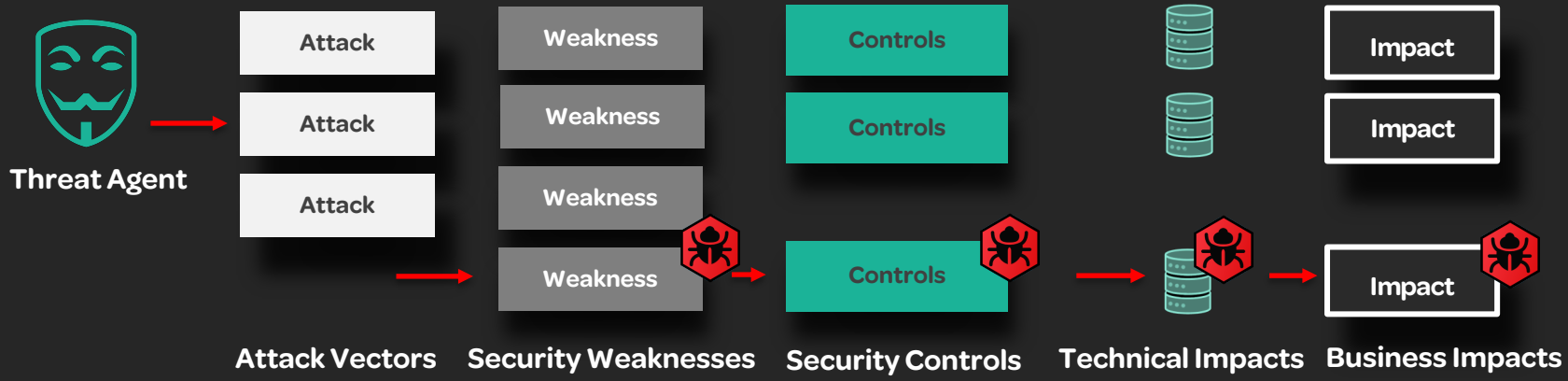
OWASP Top 10 - 2017

- **Prioritization** is vital to an effective security strategy.
- It is impossible to eliminate vulnerabilities.
- The severity of a vulnerability indicates its **potential threat**.
- The **actual threat** factors in its context.
- The key factor for defining and automating a security policy is business impact.
- Automation tolerates vulnerabilities but prevents critical vulnerabilities from promoting to upper environments until they have been **remediated**.



OWASP Risk Management

"Attackers can potentially use many different paths through your application to do harm to your business or organization. Each of these paths represents a risk that may, or may not, be serious enough to warrant attention."



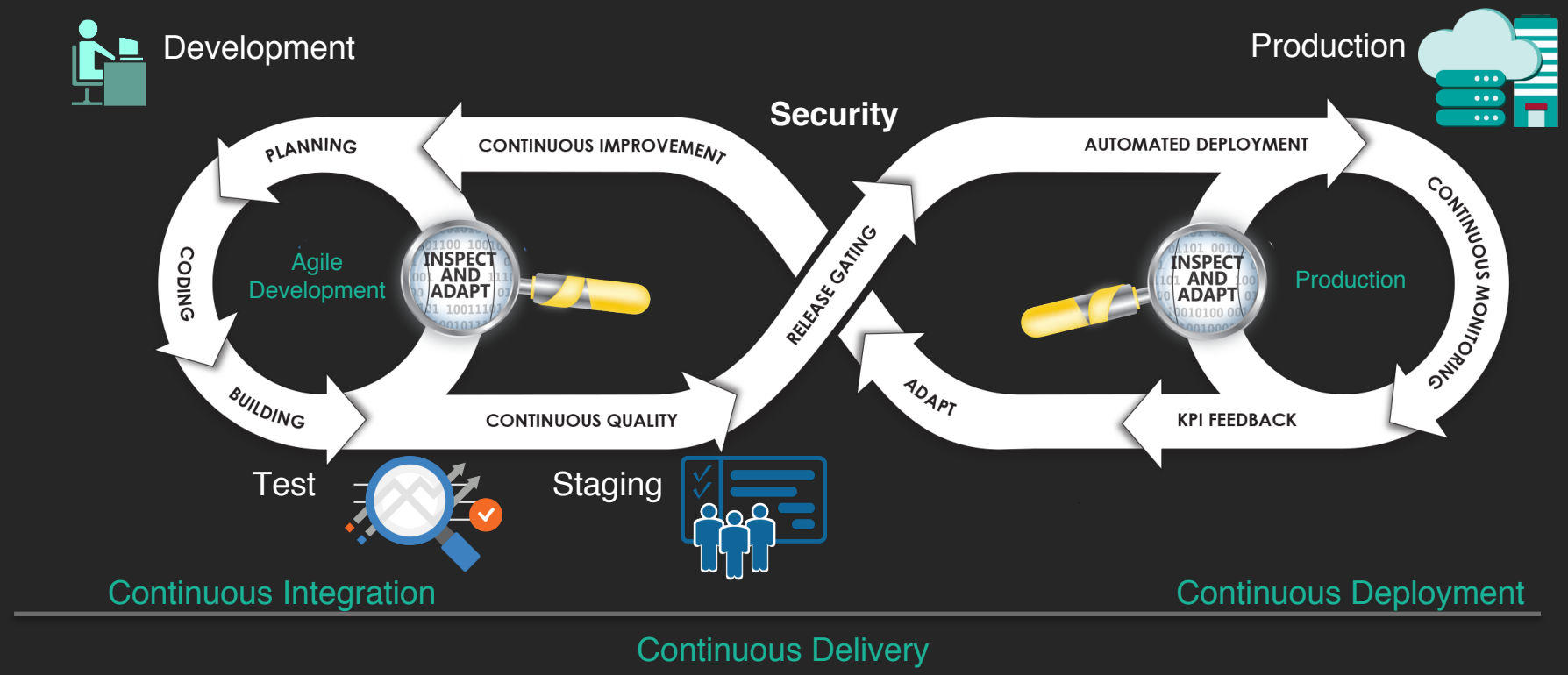
"Sometimes these paths are trivial to find and exploit, and sometimes they are extremely difficult. Similarly, the harm that is caused may be of no consequence, or it may put you out of business. To determine the risk to your organization, you can evaluate the likelihood associated with each threat agent, attack vector, and security weakness and combine it with an estimate of the technical and business impact to your organization. Together, these factors determine your overall risk."

OWASP Top 10 Risks

1) Injection
2) Broken Authentications
3) Sensitive Data Exposure
4) XML External Entities (XXE)
5) Broken Access Control
6) Security Misconfiguration
7) Cross-Site Scripting (XSS)
8) Insecure Deserialization
9) Using Components with Known Vulnerabilities
10) Insufficient Logging and Monitoring

DevSecOps Detection and Remediation

1) Automate scanning throughout the pipeline.
2) Create well-defined security policy.
3) Implement rigorous gating.
4) Reduce the attack surface of DevOps infrastructures.
5) Continuously monitor deployed applications.
6) Establish traceability back to development.
7) Practice ongoing hygiene.
8) Utilize automated Configuration Management.
9) Establish automated roll-back and recovery.
10) Practice Continuous Improvement with ongoing inspection and adaptation to accommodate security enhancements.


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

DevSecOps Essentials

Section 3: Secure Build Automation

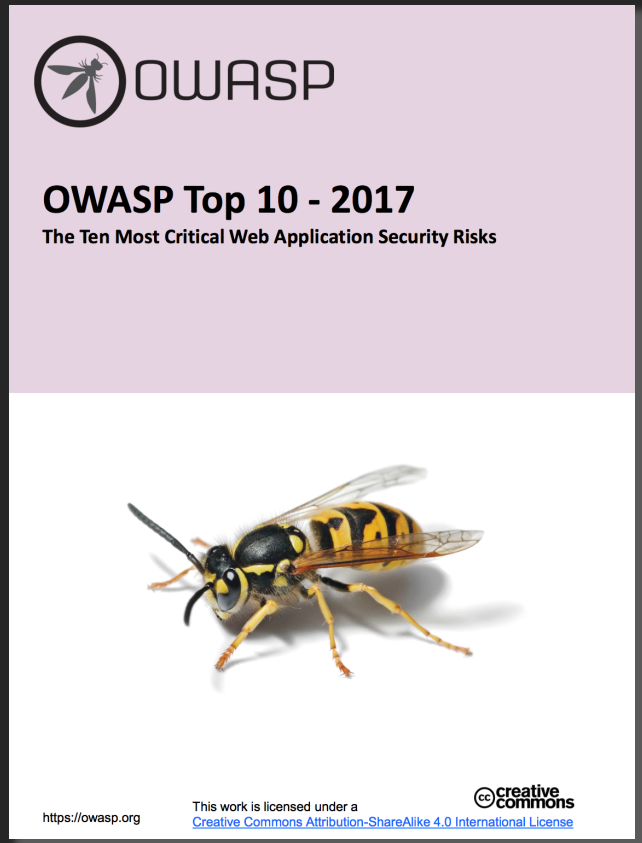
Lesson 3: Vulnerability Detection and Remediation

OWASP Dependency Checker (Use Case)

- OWASP Dependency Check is an open source scanning system.
- Dependency Check may be run from the command line or as part of an automated build such as Jenkins.
- OWASP Dependency Check maintains a downloaded CVE database from the NVD of the NIST.
- OWASP Dependency Check will provide reports in human-readable or machine readable form.
- Security policy may be enforced through automated build failure when policy thresholds are exceeded.

OWASP Top 10 - 2017

- Prioritization fundamental to an effective security strategy.
- It is not possible to eliminate vulnerabilities.
- The severity of a vulnerability indicates its potential threat.
- The actual threat would factor in its context.
- Business impact is the key determinate to define and automate security policy.
- Automation will tolerate vulnerabilities ensuring that critical vulnerabilities in key business contexts may be remediated.



OWASP Risk Management

"Attackers can potentially use many different paths through your application to do harm to your business or organization. Each of these paths represents a risk that may, or may not, be serious enough to warrant attention."

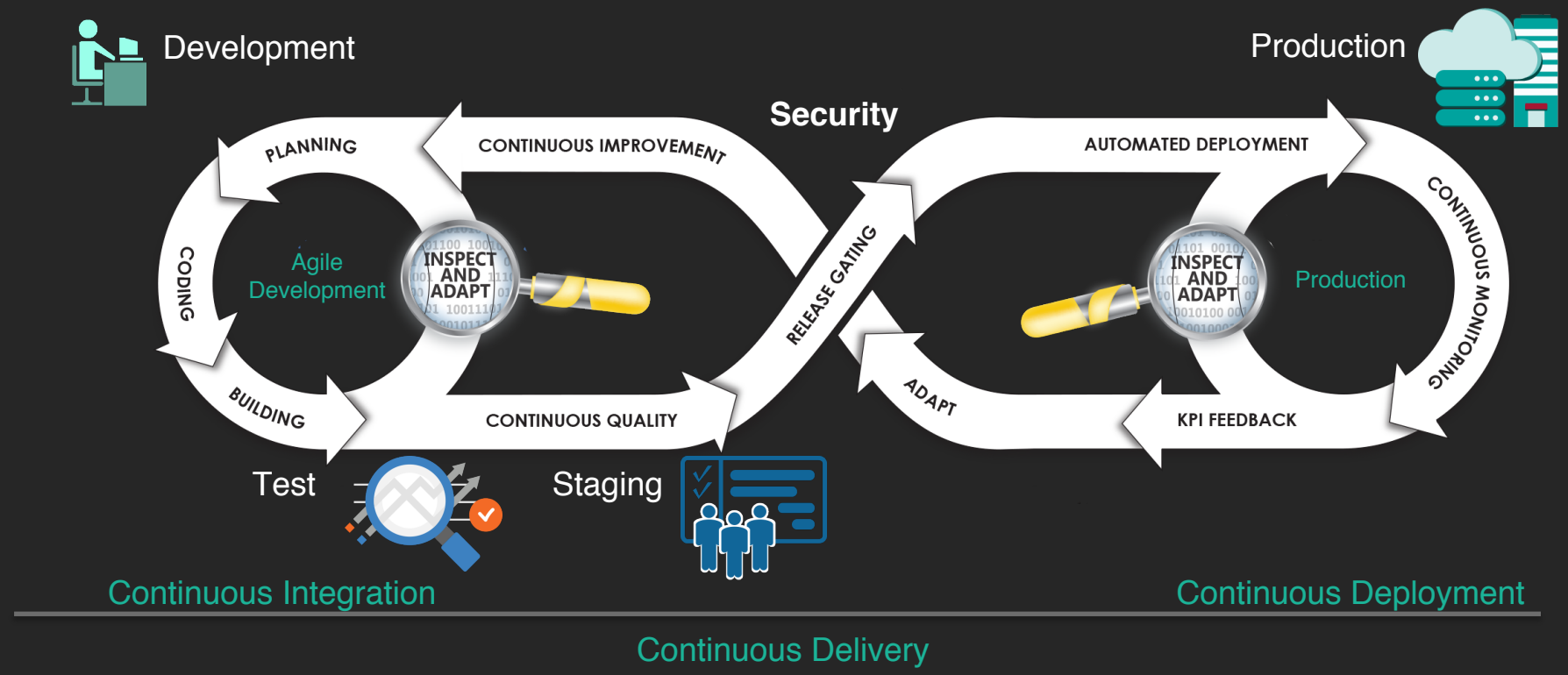
"Sometimes these paths are trivial to find and exploit, and sometimes they are extremely difficult. Similarly, the harm that is caused may be of no consequence, or it may put you out of business. To determine the risk to your organization, you can evaluate the likelihood associated with each threat agent, attack vector, and security weakness and combine it with an estimate of the technical and business impact to your organization. Together, these factors determine your overall risk."

OWASP Top 10 Risks

1) Injection
2) Broken Authentications
3) Sensitive Data Exposure
4) XML External Entities (XXE)
5) Broken Access Control
6) Security Misconfiguration
7) Cross-Site Scripting (XSS)
8) Insecure Deserialization
9) Using Components With Known Vulnerabilities
10) Insufficient Logging and Monitoring

DevSecOps Detection and Remediation

1) Automate Scanning throughout the pipeline
2) Create well-defined security policy
3) Implement rigorous gating
4) Reduce the Attack Surface of DevOps infrastructures
5) Continuously monitor deployed applications
6) Establish traceability back to development
7) Practice ongoing hygiene
8) Utilize automated Configuration Management
9) Establish automated roll-back and recovery
10) Practice Continuous Improvement with ongoing inspection and adaption to accommodate security enhancements


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

DevSecOps Essentials

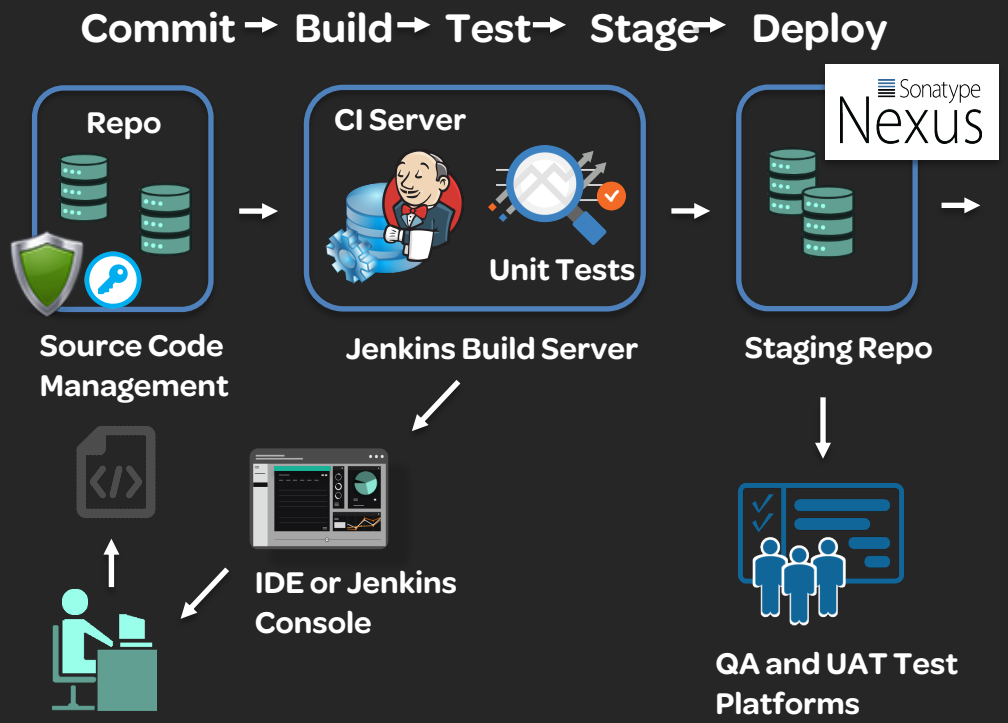
Section 3: Secure Build Automation

Lesson 4: Secure Staging

Nexus Repository (Use Case)

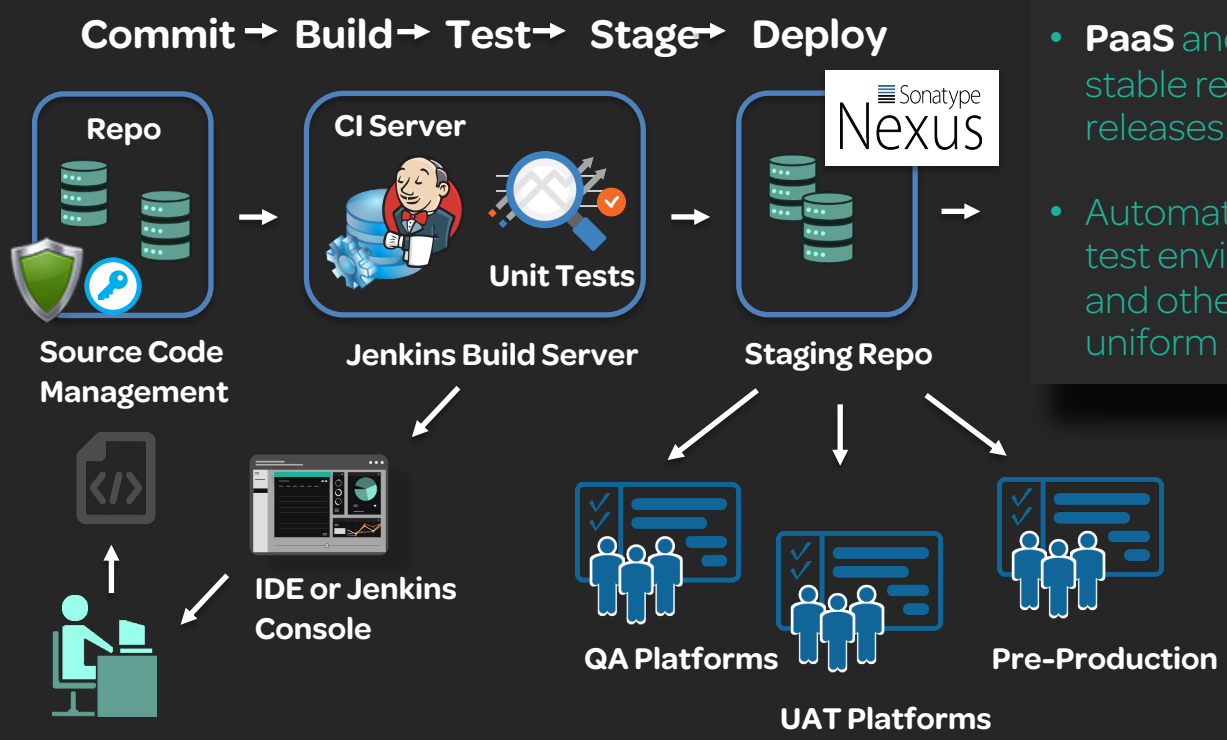
- **Sonatype Nexus** is an open-source repository management system.
- Build artifacts produced by programs like Maven are scanned as part of the automated build pipeline.
- Once a build output is produced, it is important to push the compiled file to a secure repo.
- A private on-premise or hardened cloud-based repo server is needed to prevent unauthorized access to or tampering with executable binaries.
- The use of a repo for build artifacts ensures that test, QA, and staging environments all pull the same file, maintaining uniformity and control of the deployment pipeline.

Typical CI/CD Pipeline Architecture (Using Nexus)

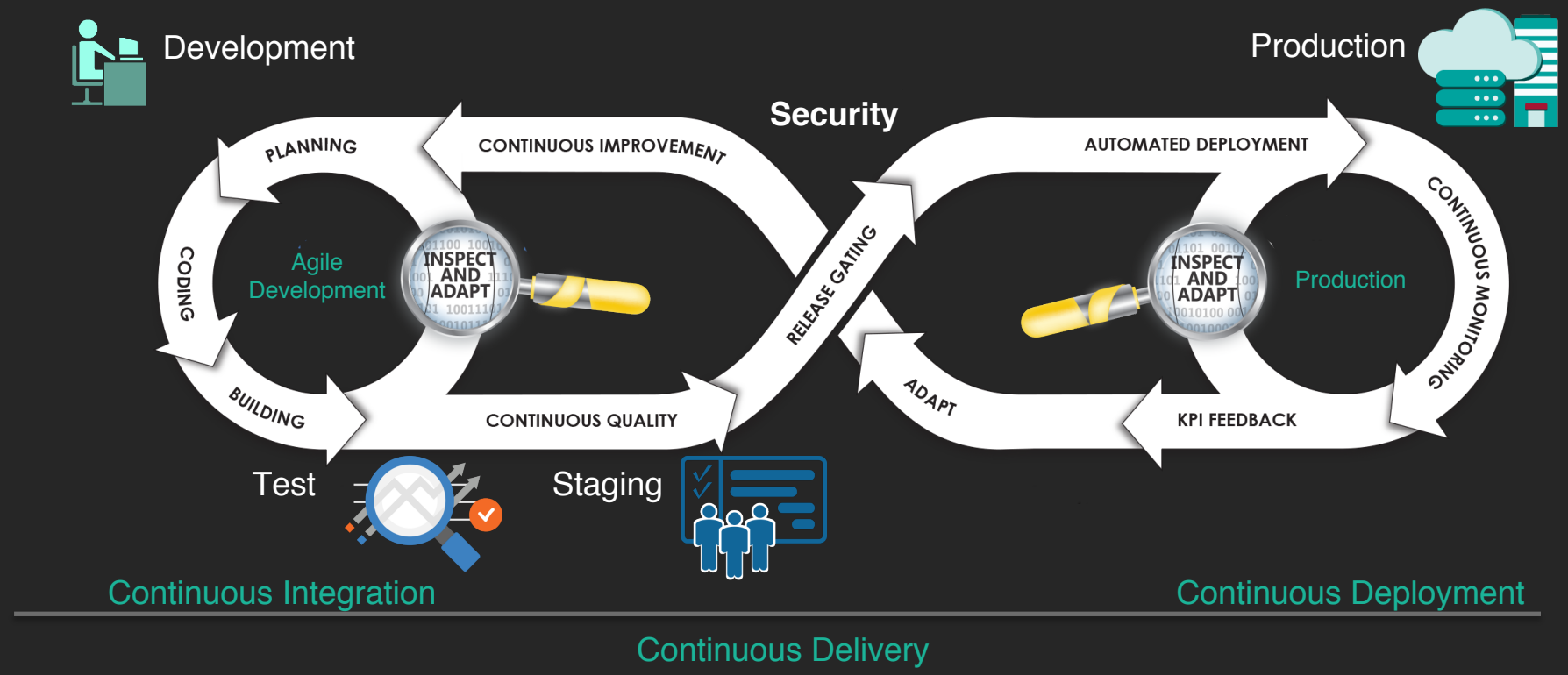


- The **Nexus** repo manager has a secure API that allows the Jenkins pipeline to push artifacts to the repo.
- Once in the repo, artifacts are protected, and the attack surface is limited to authorized and authenticated endpoints.
- QA (Quality Assurance) and UAT (User Acceptance Test) platforms can then pull the executables when needed.
- Container images can be stored in Nexus to ensure that components and binaries are packaged with build executables.
- Once a build artifact is stored, all upstream environments use the original binary, even when it is promoted to production.

Automated Provisioning and Application Release Automation (ARA)



- **PaaS** and **ARA** tooling depends on a stable repo for fast-and-frequent releases.
- Automated release and promotion to test environments prevents human error and other factors from disrupting uniform and consistent platforms.


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

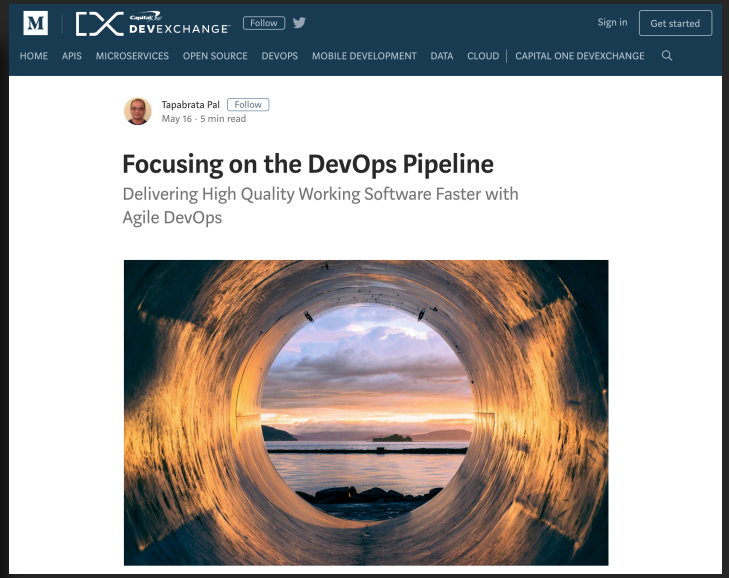
DevSecOps Essentials

Section 4: Release Gating

Lesson 1: The 16 Gates

Capital One

- **Capital One** is a good example of a large organization that practices DevSecOps.
- Capital One Senior Engineering Fellow Tapabrata Pal published an article on Medium.com that forms the basis for this lesson.
- The article is titled *Focusing on the DevOps Pipeline* and lays out the design principles (the 16 gates) that Capital One uses to develop DevSecOps pipelines.
- We will review these principles and emphasize the gates that form a continuous security discipline.



The DevExchange is a great resource for staying up-to-date with best practices in DevSecOps.

Development

Production

Security

PLANNING

CONTINUOUS IMPROVEMENT

AUTOMATED DEPLOYMENT

CODING

Agile Development

INSPECT AND ADAPT

RELEASE GATING

INSPECT AND ADAPT

Production

CONTINUOUS MONITORING

BUILDING

CONTINUOUS QUALITY

ADAPT

KPI FEEDBACK

Test

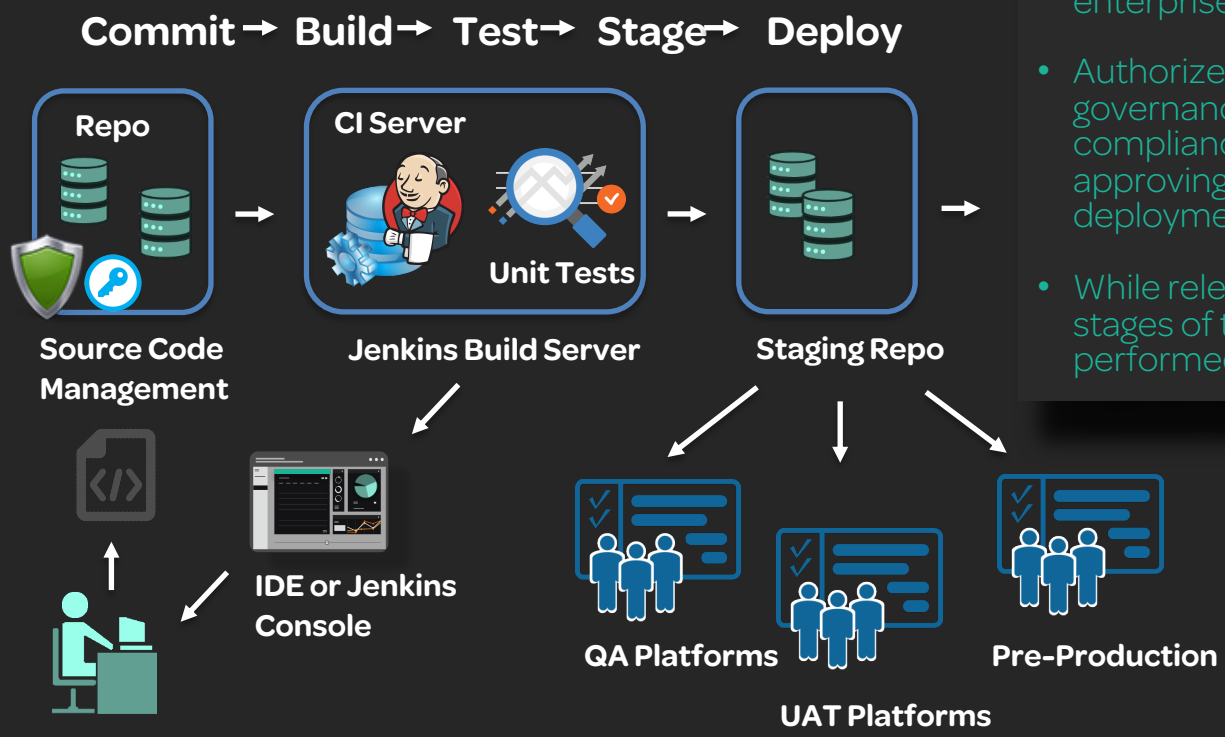
Staging

Continuous Integration

Continuous Deployment

Continuous Delivery

Release Gating



- **Release gating** is a means of control that involves checks and balances in large enterprises.
- Authorized stakeholders responsible for governance, risk management, audit, and compliance review build artifacts prior to approving a software release for deployment.
- While release gating may be done at various stages of the pipeline, it is most often performed in Pre-Production.

The 16 Gates

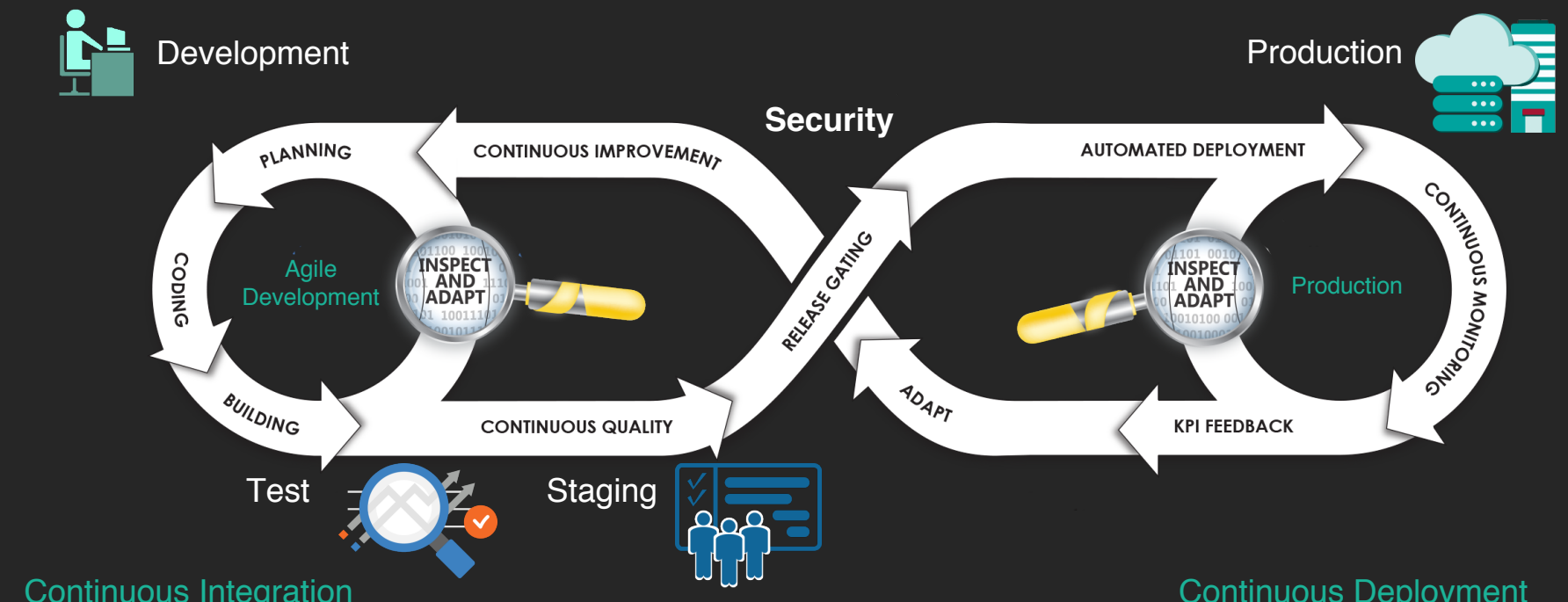
The 16 Gates are a set of software quality attributes that should be evaluated as part of a governed deployment process.

1. Source code version control
2. Optimum branching strategy
3. Static analysis
4. At least 80% code coverage
5. Vulnerability scan
6. Open source scan
7. Artifact version control
8. Auto provisioning
9. Immutable servers
10. Integration testing
11. Performance testing
12. Build deploy testing automated for every commit
13. Automated rollback
14. Automated change order
15. Zero-downtime release
16. Feature toggle

The 16 Gates (Continued)

Since our emphasis is on DevSecOps, let's take a closer look at the gates that are vital to continuous security practice.

1. Source code version control
2. Optimum branching strategy
3. Static analysis
4. At least 80% code coverage
5. Vulnerability scan
6. Open source scan
7. Artifact version control
8. Auto provisioning
9. Immutable servers
10. Integration testing
11. Performance testing
12. Build deploy testing automated for every commit
13. Automated rollback
14. Automated change order
15. Zero-downtime release
16. Feature toggle


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

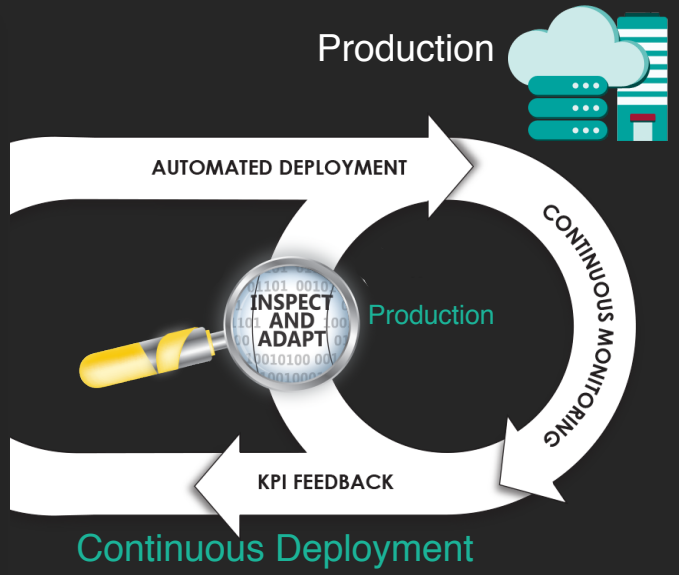
DevSecOps Essentials

Section 5: Continuous Delivery Release Automation

Lesson 1: Automated Deployment

Continuous Delivery Release Automation

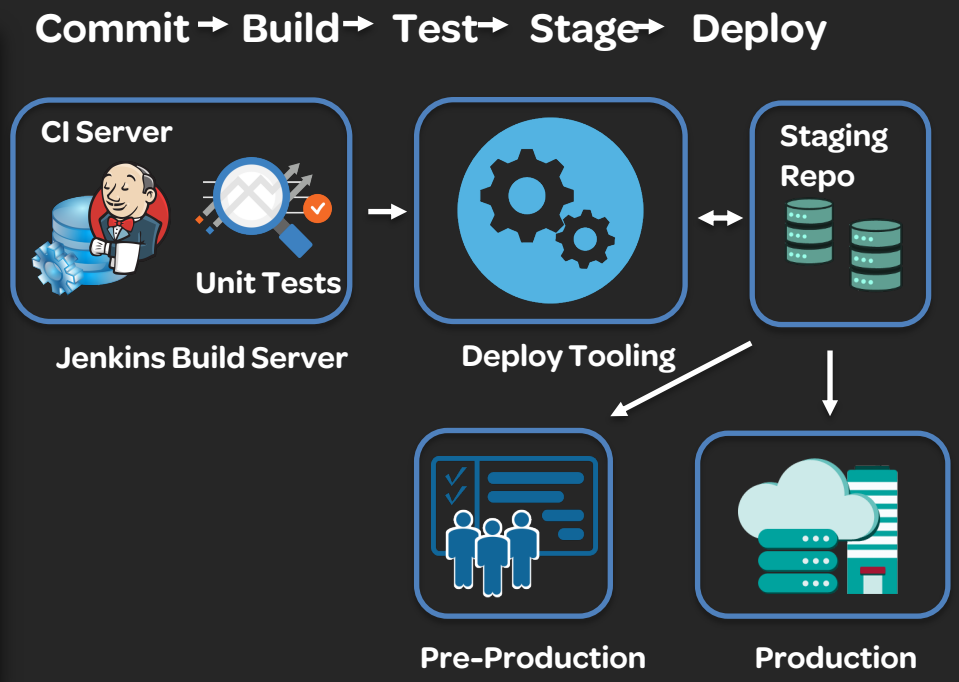
- **Continuous Delivery Release Automation (CDRA)** refers to the process of promoting application workloads to upper environments with minimal or no manual intervention.
- Automated deployment systems are typically employed in all post-build environments, including Test, QA, UAT, Pre-Production, and Production.
- CDRA systems are categorized into two domains: **Application Release Automation (ARA)** and **Application Release Orchestration (ARO)**.
- In this lesson, we'll focus primarily on ARA. ARO includes additional concepts such as self-healing, cloud bursting, and hybrid cloud orchestration.



Automated deployment is critical for control and consistency when promoting workloads into production environments.

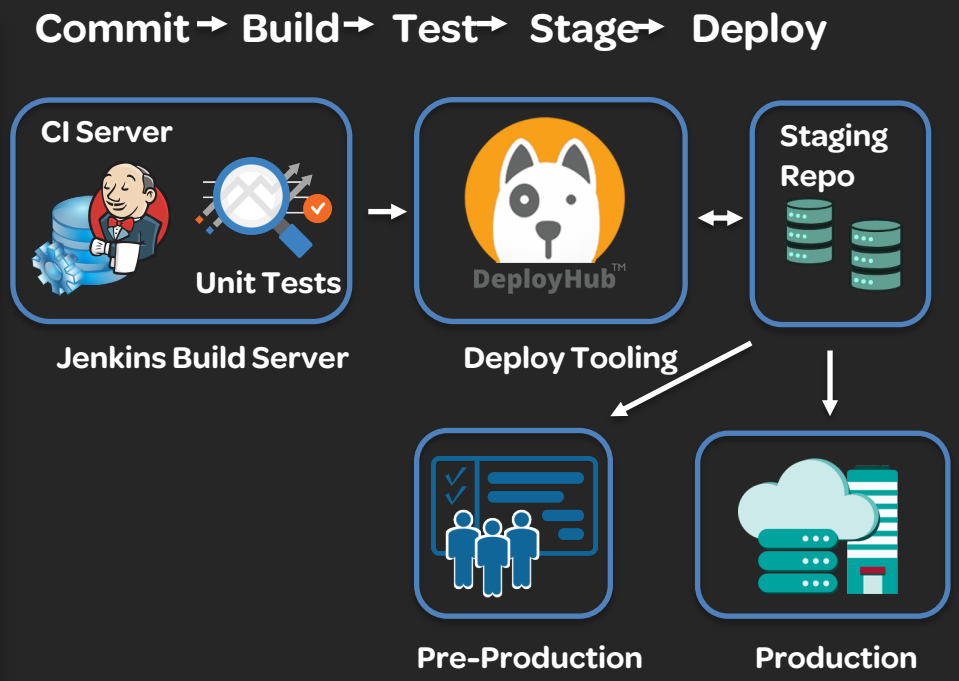
Automated Deployment

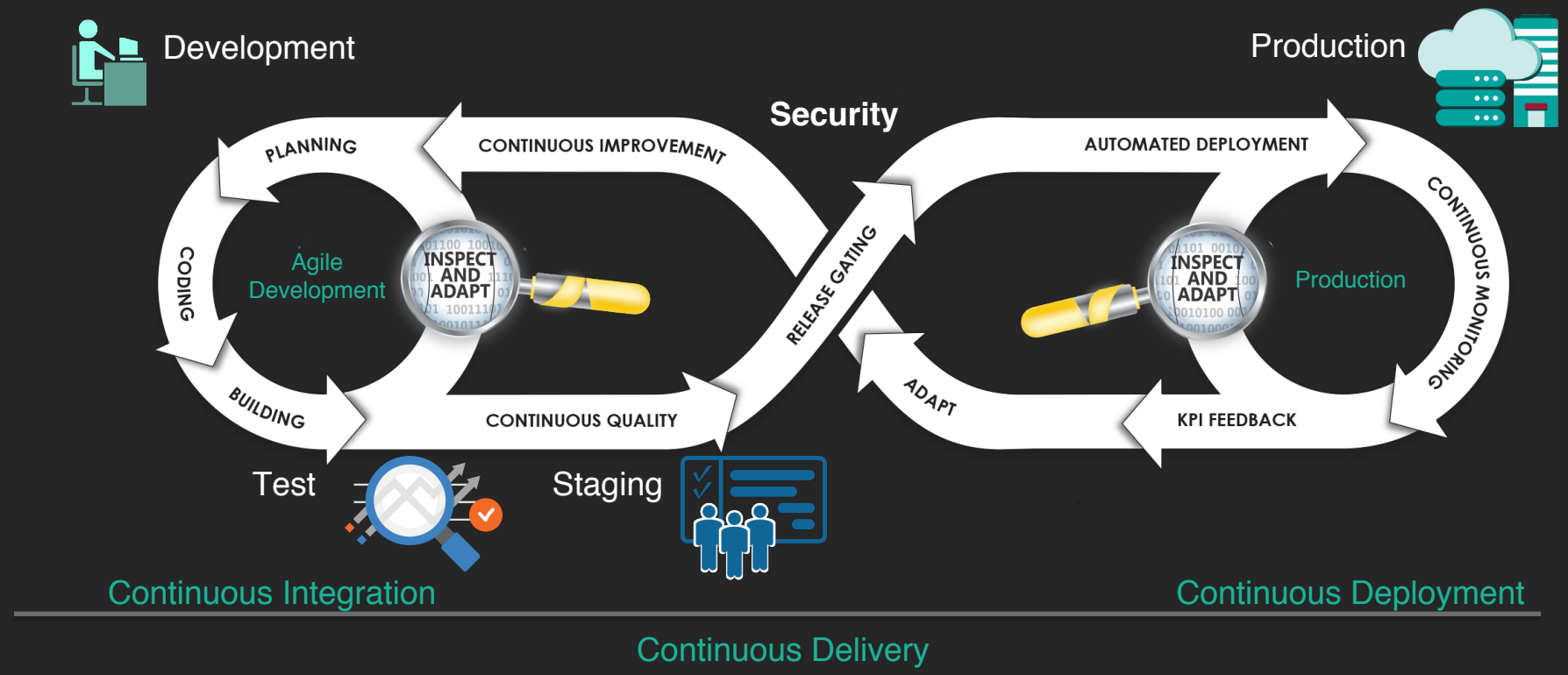
- **Automated Deployment** installs build artifacts on upper-level environments, including production systems.
- Automation is key for improving throughput, controlling outcomes, and ensuring consistency.
- Once systems have cleared gating, tooling triggers call programs that prepare the infrastructure and deploy or install the application so that it can be executed immediately.



DeployHub

- **Automated Deployment tools** vary widely in capability and implementation.
- DeployHub is an open-source solution that is available as a hosted SaaS platform or an on-premise implementation.
- DeployHub integrates with Jenkins build automation, the Nexus Repo, and Ansible playbooks for deployment.




Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

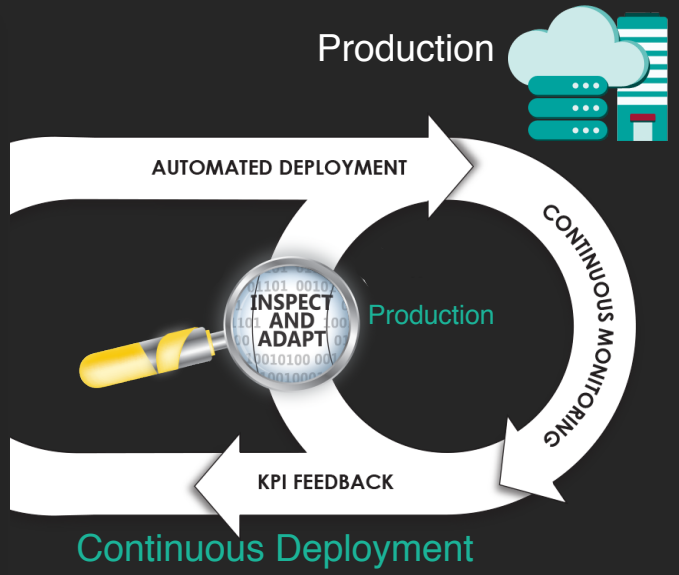
DevSecOps Essentials

Section 5: Continuous Delivery Release Automation

Lesson 2: Configuration Management

Configuration Management

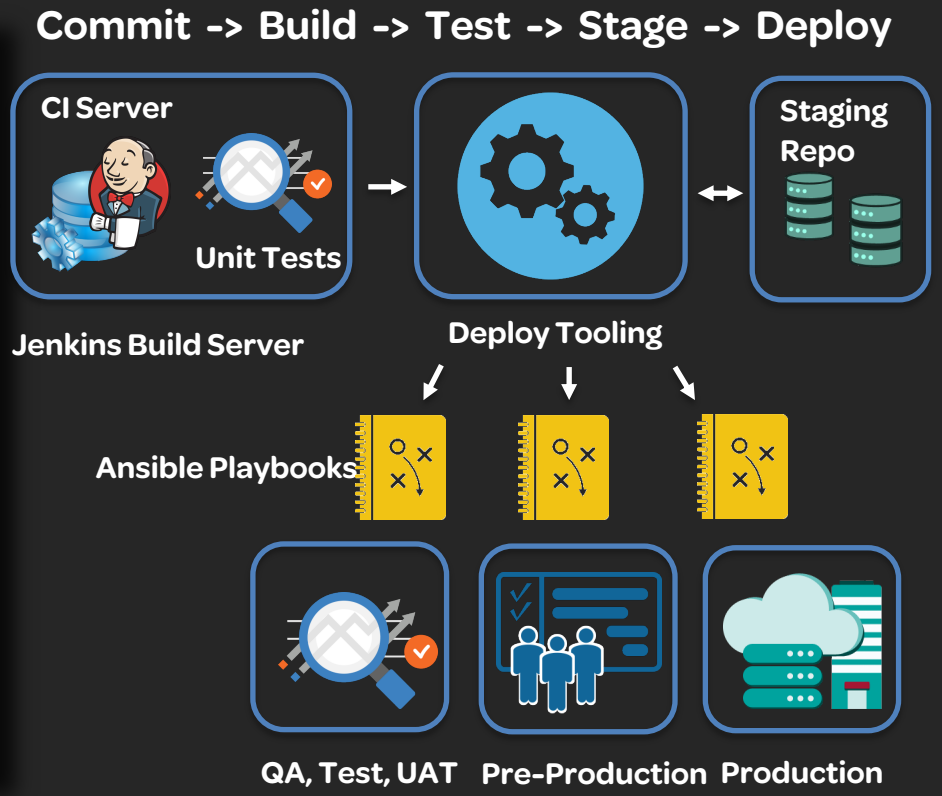
- **Configuration Management** is the practice and tooling required to fully automate the process of configuring target environments and deploying application workloads to them.
- While Configuration Management is useful at all stages of provisioning, it is especially important when dealing with production platforms.
- Configuration Management is an automated approach that replaces traditional scripting and improves control and consistency when deploying applications.
- Configuration management within this lesson is focused on the domain of Automated Deployment.

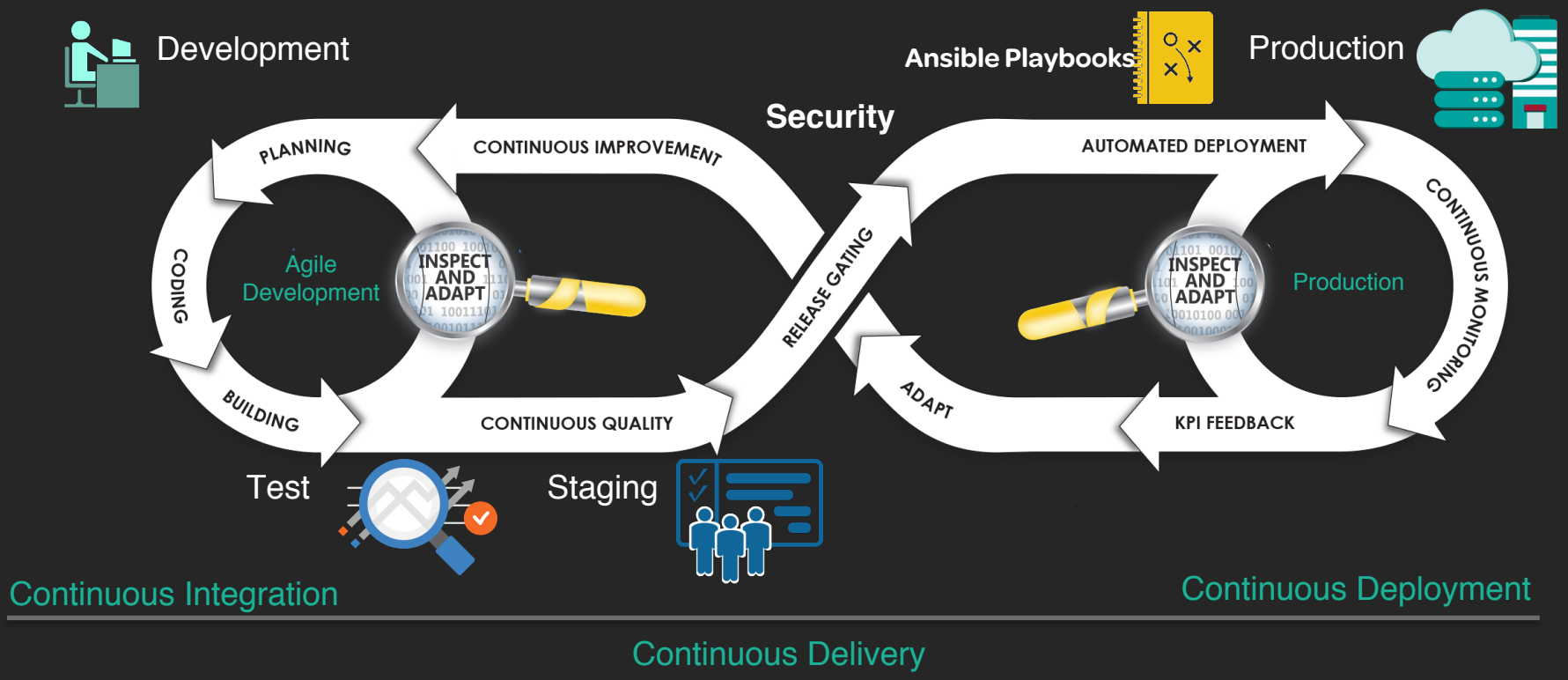


Automated Deployment is critical to ensure control and consistency when promoting workloads into production environment.

Configuration Management (Ansible Use Case)

- Ansible is an open source configuration management tool from Red Hat.
- Ansible utilizes playbooks to deploy both systems-level and application-level components to target environments.
- Ansible uses YAML files known as playbooks to automate deployments.
- Playbooks have the ability to test state on target platforms and only perform those configuration and installation steps necessary to enable the workload.
- Ansible performs its work in a predefined chronology to ensure dependencies are met.
- Infrastructures as Code (IaC) is the key aim of tooling such as Ansible.




Development
Ansible Playbooks
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Continuous Integration
Continuous Deployment
Continuous Delivery

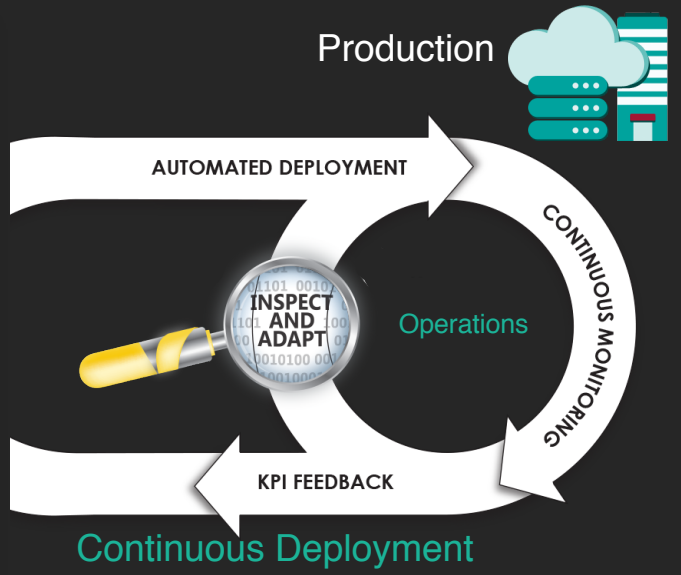
DevSecOps Essentials

Section 6: Production Monitoring and Ongoing Detection and Remediation

Lesson 1: Production Monitoring

Production Monitoring

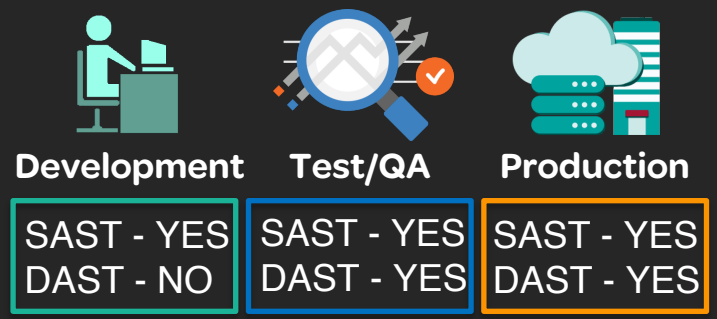
- **Continuous monitoring** is the ongoing, automated evaluation of workloads to assess their health and function.
- **Production monitoring** may include threat detection, vulnerability detection, and malware prevention.
- Since new threats and vulnerabilities are developed and discovered over time, it is important to monitor applications throughout their entire lifespan, not just while they're under development.
- When applications need to be refactored or upgraded with patched third-party libraries, feedback loops and automated reporting notify the development team that vulnerabilities or other defects require attention.



Continuous monitoring of applications in production ensures ongoing stability.

Static Application Security Testing (SAST)

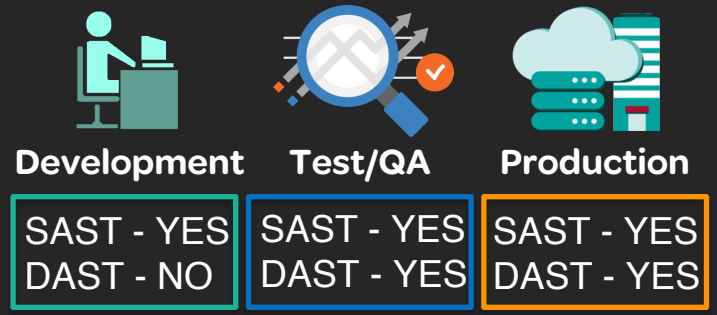
- **Static scanning** is used to test for new vulnerabilities that have become known since an application's release into production.
- Scanning tools use container image manifests and build time dependency lists to determine which known vulnerabilities are present in compiled applications and container images.
- Due to ongoing changes to operating systems, binary libraries, and third-party software components, applications need to be continually upgraded. Scanning is fundamental to hygiene, and as applications are redeployed, automated pipelines ensure persistent security.



Ongoing Static Application Security Testing (SAST) is necessary to keep applications free of vulnerabilities.

Dynamic Application Security Testing (DAST)

- **Dynamic scanning** uses black box security testing methodology.
- A black box test involves testing an application from the outside in while it is running in production.
- In DAST, a security analyst attempts to hack an application in the same way a cyber attacker would.
- Unlike SAST, DAST does not involve examining source code to identify vulnerabilities from the inside out.

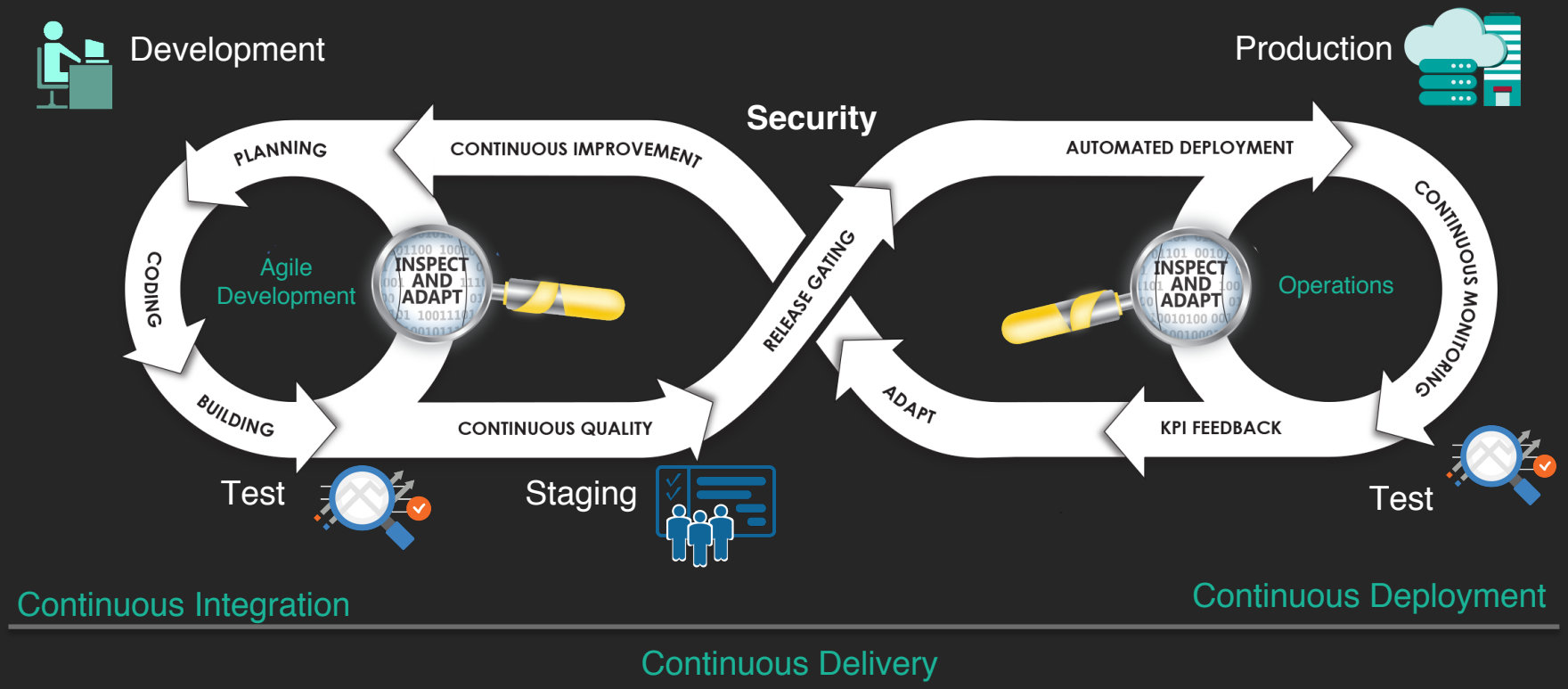


Ongoing Dynamic Application Security Testing (DAST) helps keep production applications free of vulnerabilities.

Penetration Tests

- **Penetration tests (pentests)** are a form of DAST that use external programs to interrogate applications through their exposed API and HTTP endpoints.
- Penetration tests simulate automated cyber attacks on production infrastructure.
- Penetration tests detect common vulnerabilities such as injection, cross-site scripting, and flaws in authentication and identity and access management (IAM).

Penetration tests use known security threats and hacking techniques to test applications in production.


Development
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Operations
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Test
Continuous Integration
Continuous Deployment
Continuous Delivery

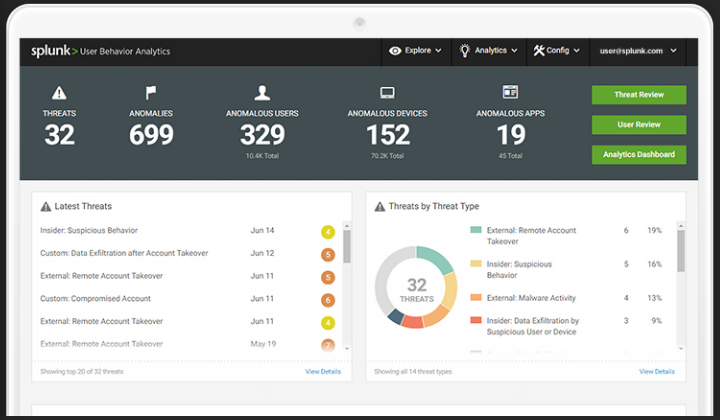
DevSecOps Essentials

Section 6: Production Monitoring and Ongoing Detection and Remediation

Lesson 2: Dashboards and Automated Vulnerability Detection

Dashboards and Automated Vulnerability Detection

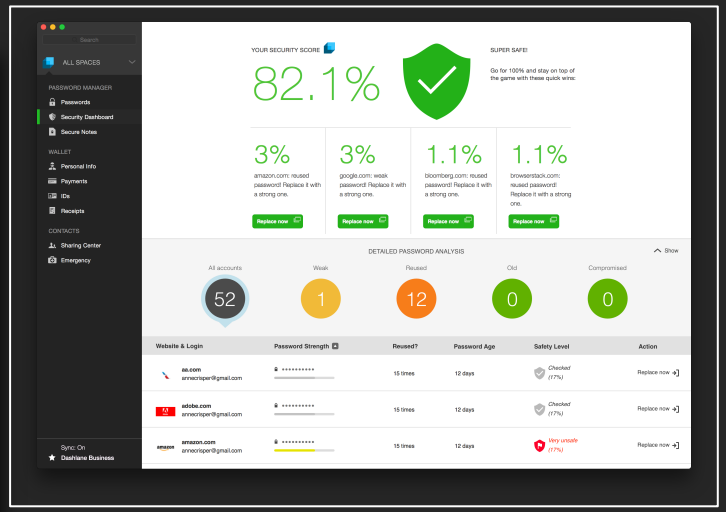
- **Dashboards** provide continual updates on the health of applications running in production.
- Dashboards aggregate and analyze logs, providing a stream of real-time data that is presented visually based on predefined configurations.
- Extensible dashboard software systems support plugins that can be configured according to user preferences.
- Dashboards also let you set up real-time alerts for system events that require immediate attention.



Most log aggregation and dashboard tools support additional plugins for threat monitoring.

Automated Vulnerability Detection

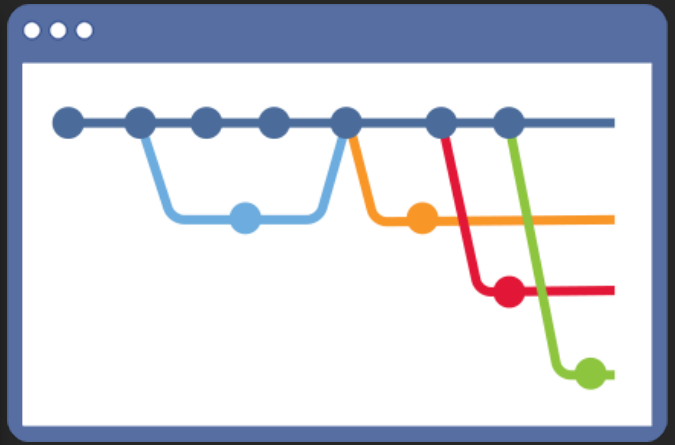
- **DAST** and **SAST** tools can provide logs for aggregation and monitoring in dashboard systems.
- Many dashboard visualization systems generate benchmark thresholds of normal performance and report anomalies that exceed normal limits.
- Advanced machine learning and artificial intelligence algorithms can be used to score workloads over time and provide alerts when threats increase.



Production tests allow applications to be scored based on their threat level and overall health.

Traceability and Rollback

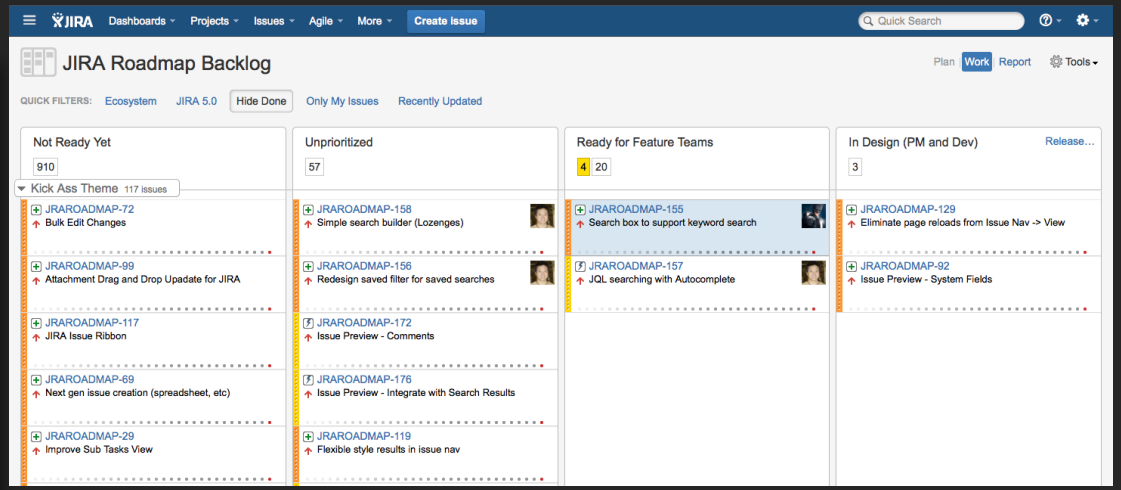
- **Traceability** refers to the process of communicating information about threat detection back to stakeholders, including the development team.
- When release management and source code control is established to track major releases and sub-releases of applications, vulnerability reporting can be traced back to the specific source code being evaluated.
- Alerts and threat detection can be used to automatically **roll back** vulnerable systems when new releases appear flawed.
- Configuration management provides an automatic way to upgrade and downgrade applications when necessary.



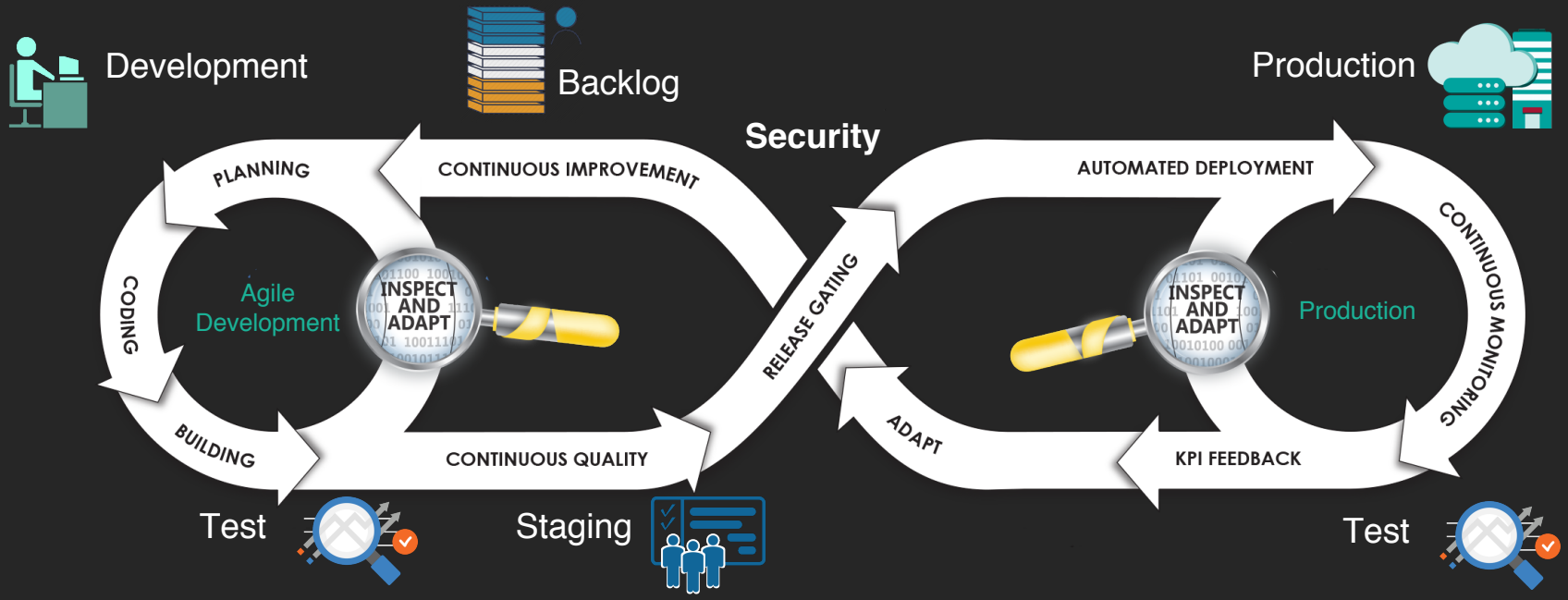
Branching and rebasing source code throughout an automated release cycle helps with identifying vulnerable source code during production monitoring.

Agile Lifecycle Integration

- Development teams use **Agile backlogs** to record user stories.
- Production monitoring systems often provide a way to create tickets and integrate with Agile lifecycle management tools.
- Up-to-date feedback from production systems helps with continuous adaption and improvement.



Many dashboard and production monitoring systems include plugins or modules that enable integration with popular Agile lifecycle management systems like Atlassian's Jira.

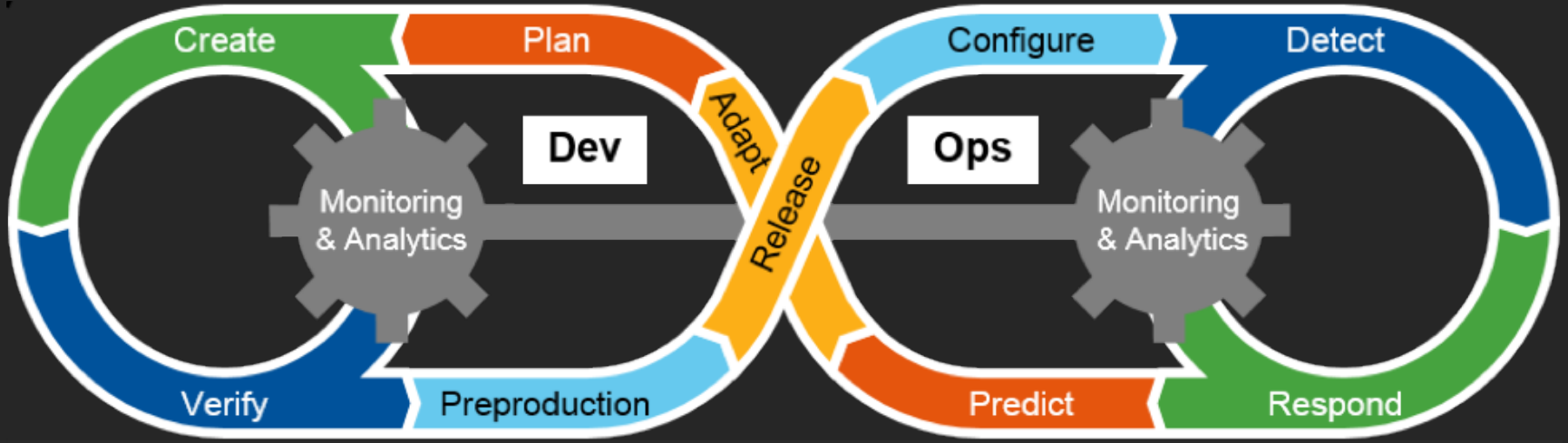

Development
Backlog
Production
Security
PLANNING
CONTINUOUS IMPROVEMENT
AUTOMATED DEPLOYMENT
CODING
Agile Development
INSPECT AND ADAPT
RELEASE GATING
INSPECT AND ADAPT
Production
CONTINUOUS MONITORING
BUILDING
CONTINUOUS QUALITY
ADAPT
KPI FEEDBACK
Test
Staging
Test
Continuous Integration
Continuous Deployment
Continuous Delivery

DevSecOps Essentials

Section 7: Conclusion

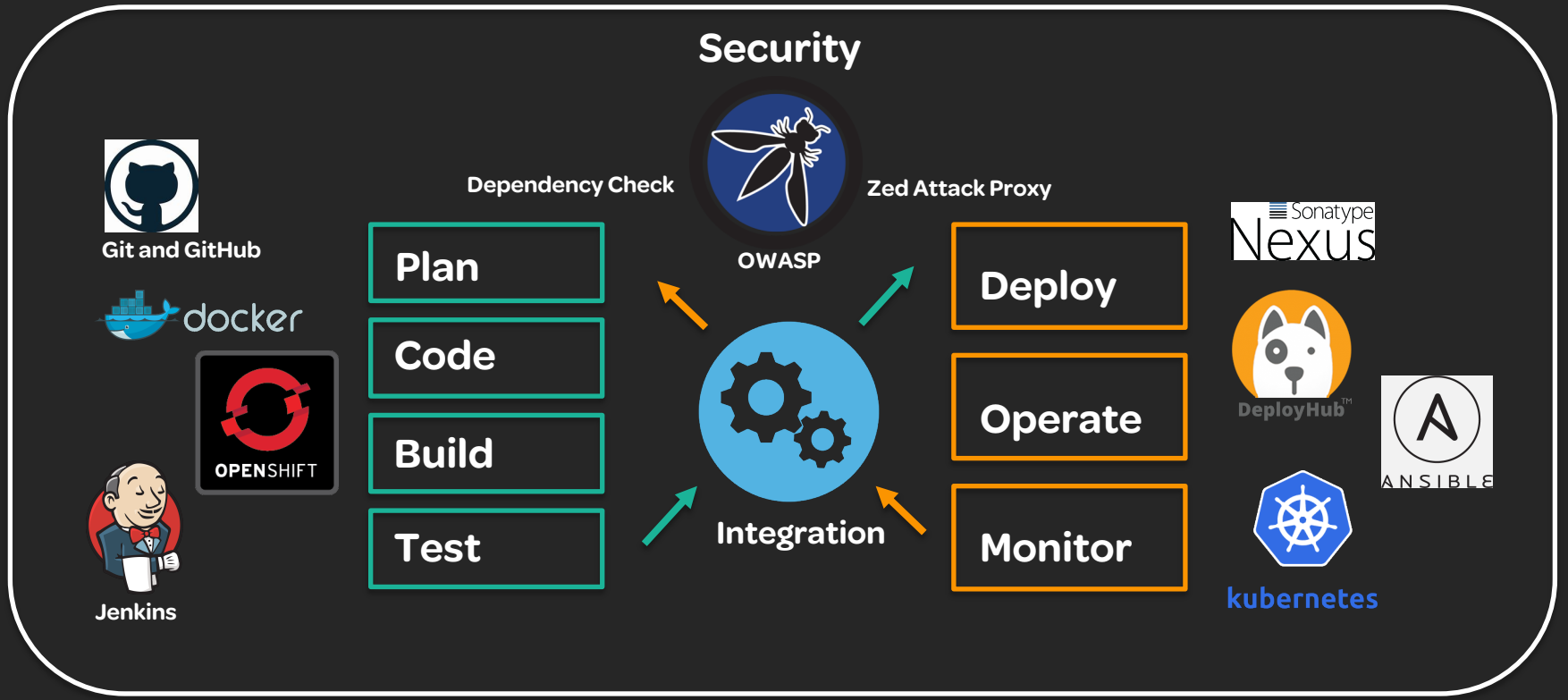
Lesson 1: Summary and Next Steps

Summary



- **DevSecOps** is relevant to all aspects of modern software development and deployment.
- Now that you know the basic vocabulary and concepts, talk to your colleagues about DevSecOps.
- Choose your areas of specialty and consider deep dives or certifications in those technologies.

DevSecOps Automation Use Cases



Next Steps (Areas of Specialty)

Development

- Source Code Management
- Agile Lifecycle Tools
- Test-Driven Development
- PaaS Tooling
- Security Standards
- Containerization

Release Management

- Build Pipeline Automation
- Continuous Integration
- Automated Testing
- Pre-Production Staging
- Release Automation
- Configuration Management

Operations

- Server Hardening
- Application Hardening
- Continuous Deployment
- Continuous Monitoring
- Security Analytics
- Cloud Orchestration

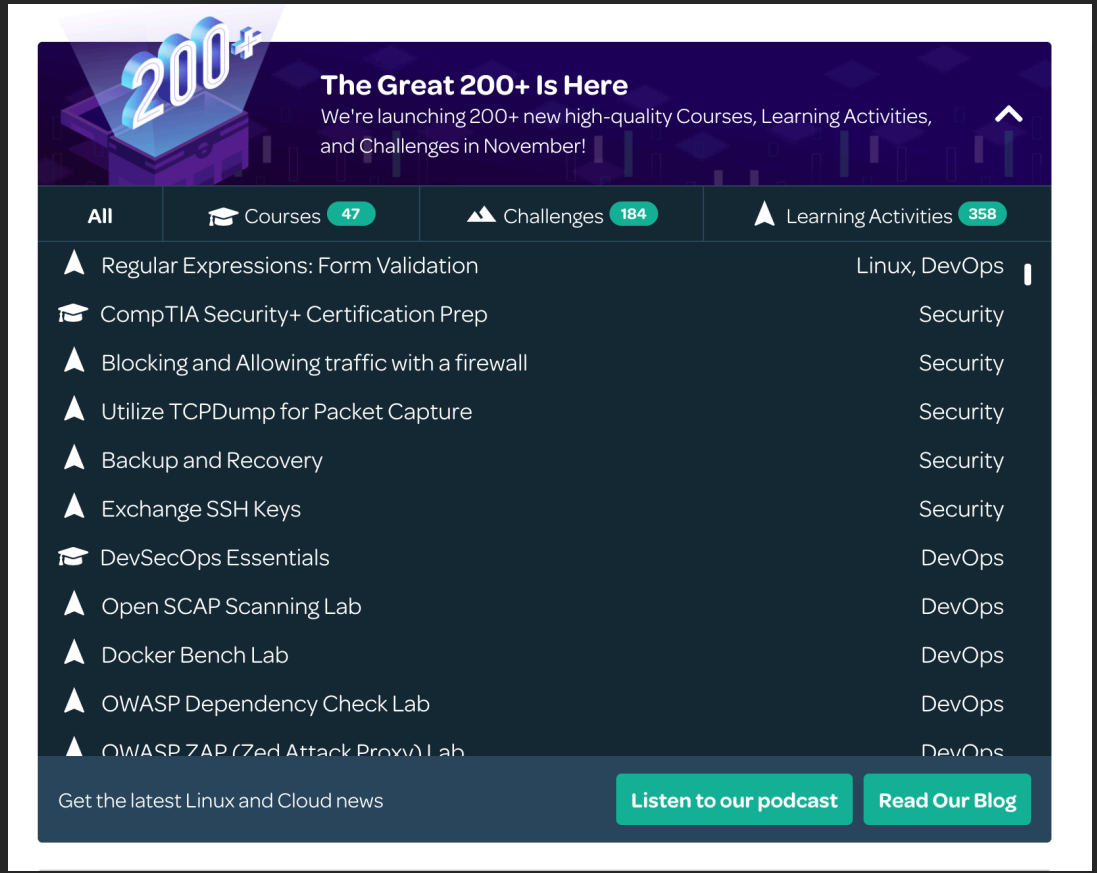
Next Steps (Linux Academy Courses)

- Certified Ethical Hacker
- LPIC-3 Exam 303 Security
- AWS Security Specialty Certification

- Kali Linux Deep Dive
- CompTIA Security+

- CompTIA PenTesting+

- Cloud Security Fundamentals



Measuring DevSecOps Success

- Deployment frequency (fast and frequent releases)
- Lead time (code-to-cash cycle)
- Detection of threats, vulnerabilities, and malware
- Mean time to repair and remediation
- Efficiency of rollback and recovery

Enable businesses to keep pace with change without compromising security.