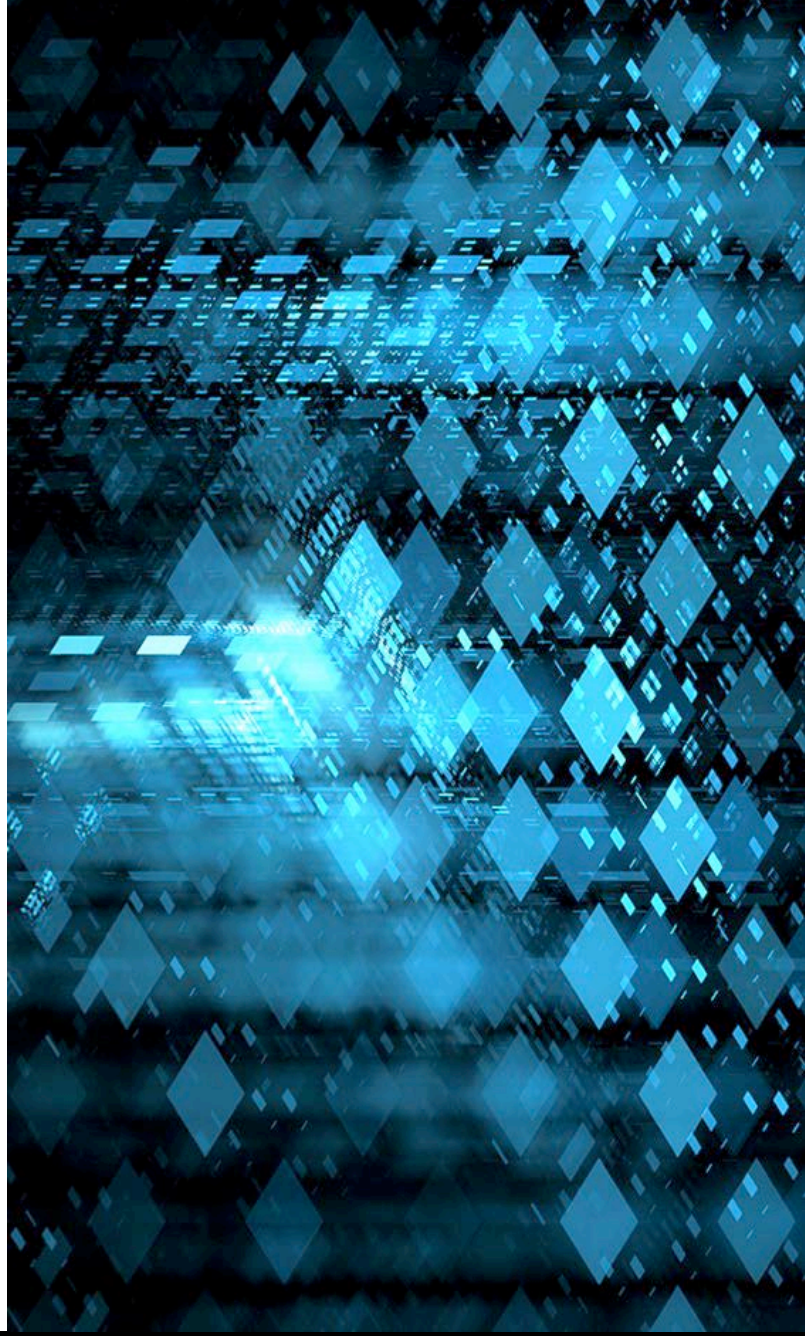
Secure DevOps

Hasan Yasar

Technical Manager

CERT | Secure Lifecycle Solutions Group

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



UNCLASSIFIED

Notices

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0130

UNCLASSIFIED

Topics

DevOps Fundamentals

DevOps and Security

Platform Security

AppSec - Secure DevOps

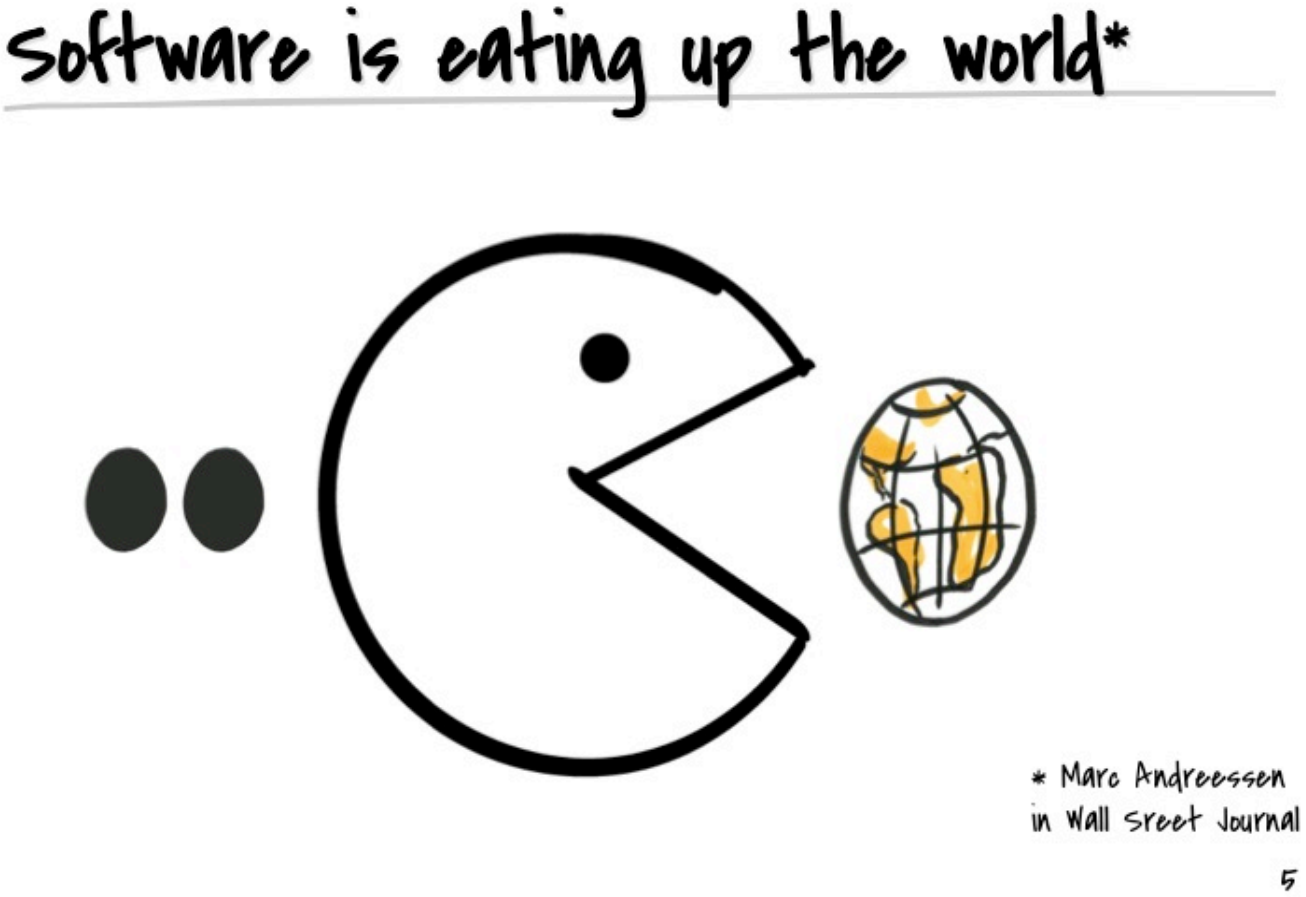
DevOps Anti-Patterns

Background

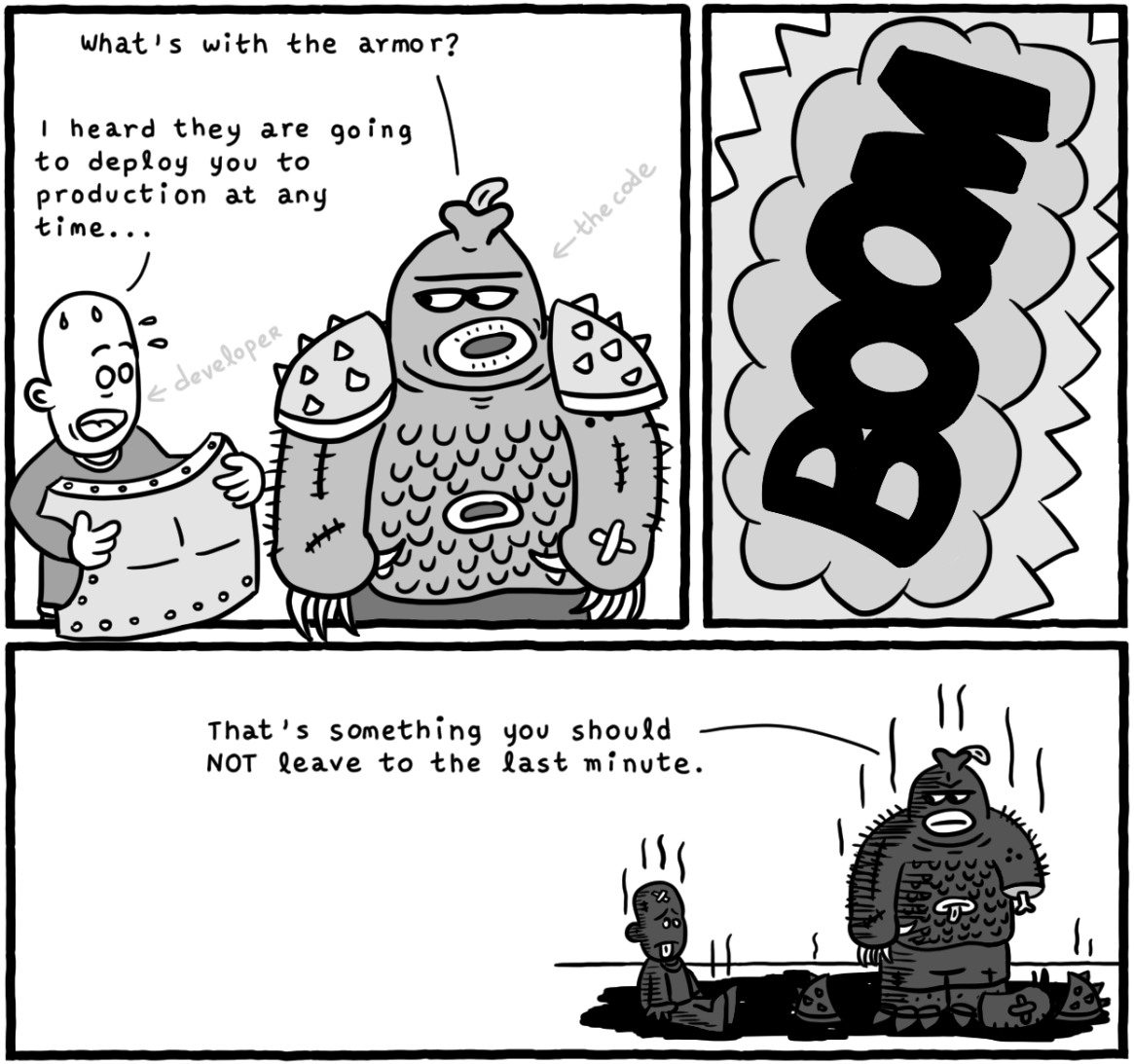
- The Software Engineering Institute (SEI) is a Federally Funded Research and Development Center (FFRDC)
- Source of CMMI, TSP, PSP
- Research software and cybersecurity problems of considerable complexity, create and test innovative technologies, and transition maturing solutions to widespread use.
- Assisted numerous government organizations in modernizing their software development practices in the spirit of DevOps principles.

- 25+ years of software development experiences
- Certified Scrum Practitioner
- Various roles throughout SDLC ; Manager, Architect, Tester, Developer, QA, IT Manager, Project Manager, VP…
- Started with waterfall in 1990
- Started with agile in 2003
- Started with DevOps in 2010
- Instructor on delivering DevOps course at CMU, SEI since 2015

The world we live in..



Last Minute Security..



https://dzone.com/articles/last-minute-security-comic

Topics

- *DevOps Fundamentals*
- DevOps and Security
- Platform Security
- AppSec - Secure DevOps
- DevOps Anti-Patterns

What is DevOps?

DevOps (a portmanteau of "development" and "operations") emphasizes communication, collaboration, and integration between software developers and information technology (IT) operations personnel. [1]

The history of DevOps

- Patrick Debois “Agile infrastructure and operations: how infra-gile are you?”, Agile 2008 Conference
- John Allspaw “ 10+Deploys per Day: Dev and Ops Cooperation”, Velocity 2009
- DevOpsDays, October 30th 2009, #DevOps term born

[1] http://en.wikipedia.org/wiki/DevOps

Who are Dev?

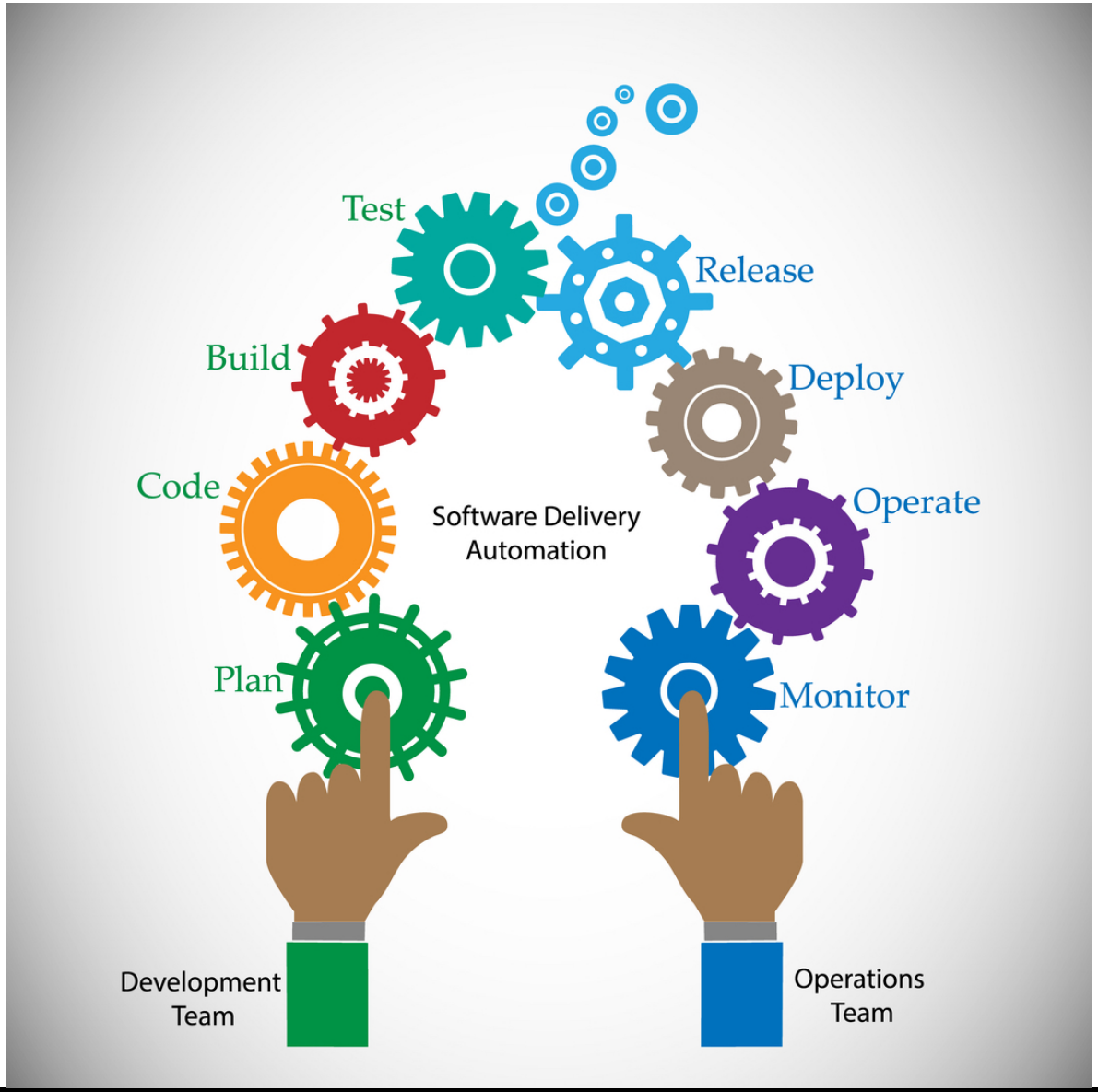
- Follow Agile methodologies
  - Using Scrum, Kanban and modern development approaches
  - Self directing, self managed, self organized
- Using any new technology
  - Each Dev has own development strategy
  - OpenSource,
- Allowed to have
  - Close relationships with the business
  - Software driven economy

Want to deliver software faster with new requirements…

Who are Ops?

- Operations
  - Runs the application
  - Manages the infrastructure
  - Support the applications
- Operations provides (ITIL Says)
  - Service Strategy
  - Service Design
  - Service Transition
  - Service Operations

DevOps connects Dev to Ops to Dev



UNCLASSIFIED

DevOps aims to Increase…

…the pace of **innovation**

…**responsiveness** to business needs

…**collaboration**

…software **stability and quality**

... continuous feedback

DevOps has four Fundamental Principles

Collaboration: between project team roles

Infrastructure as Code: all assets are versioned, scripted, and shared where possible

Automation: deployment, testing, provisioning, any manual or human-error-prone process

Monitoring: any metric in the development or operational spaces that can inform priorities, direction, and policy

DevOps promotes Collaboration

Heavy collaboration between Dev and Ops on:

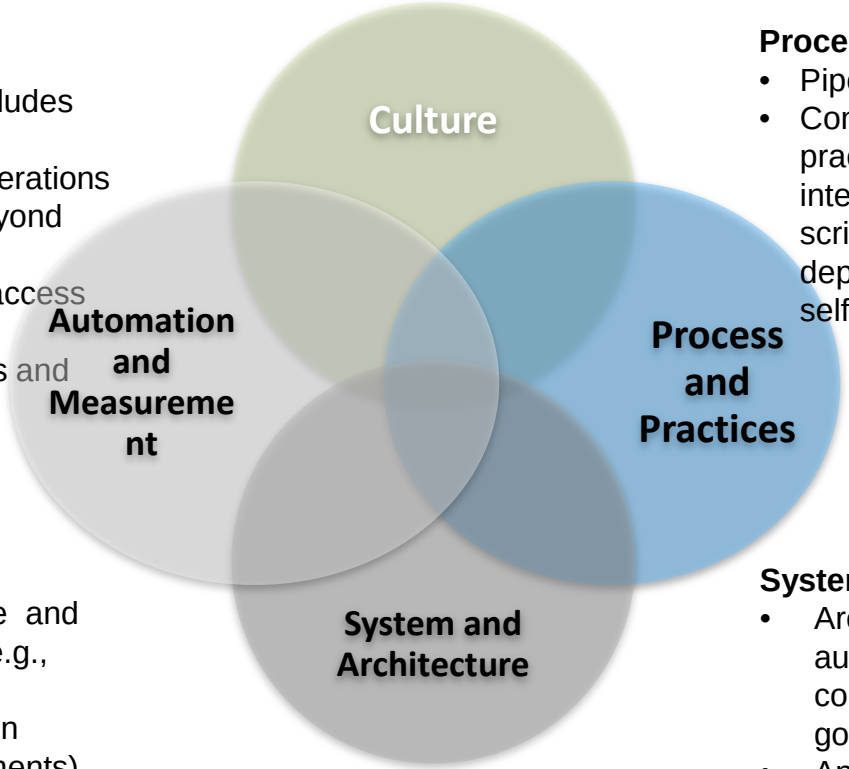
- Design / Architecture decisions
- Environment / Network configuration
- Deployment planning
- Code Review

Constantly available open communication channels:

- Dev and Ops together in all project meetings
- Chat/Email/Wiki services available to all team members
- Dev / Ops report together as one project team

Multiple Dimensions of DevOps

Culture
- Developer and Ops collaborate (Ops includes security)
- Developers and Operations support releases beyond deployment
- Dev and Ops have access to stakeholders who understand business and mission goals

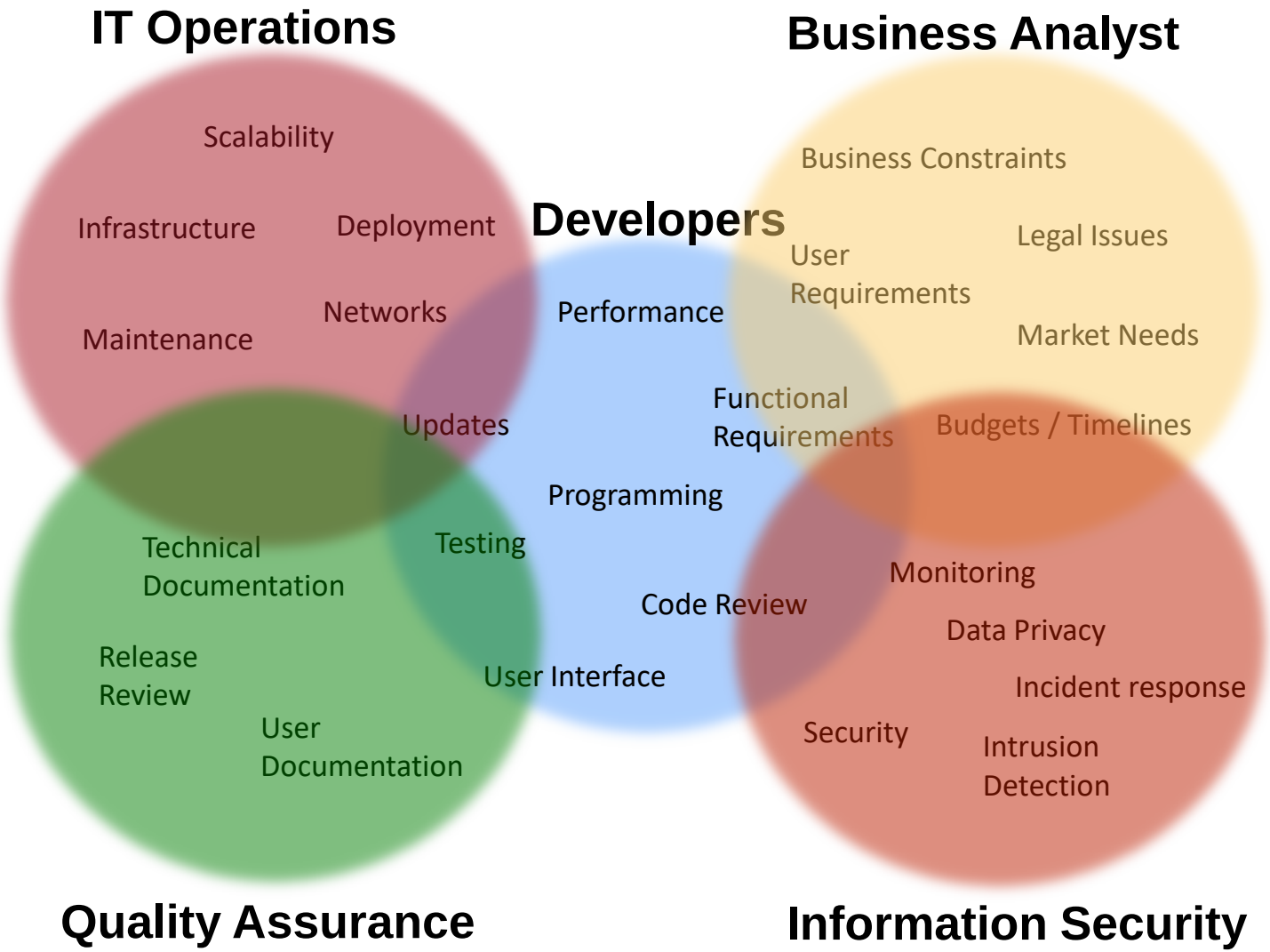


Process and Practices
- Pipeline streamlining
- Continuous-delivery practices (e.g., continuous integration; test automation; script-driven, automated deployment; virtualized, self-service environments)

Automation/ Measurement
- Automate repetitive and error-prone tasks (e.g., build, testing, and deployment maintain consistent environments)
- Static analysis automation (architecture health)
- Performance dashboards

System and Architecture
- Architected to support test automation and continuous-integration goals
- Applications that support changes without release (e.g., late binding)
- Scalable, secure, reliable, etc.

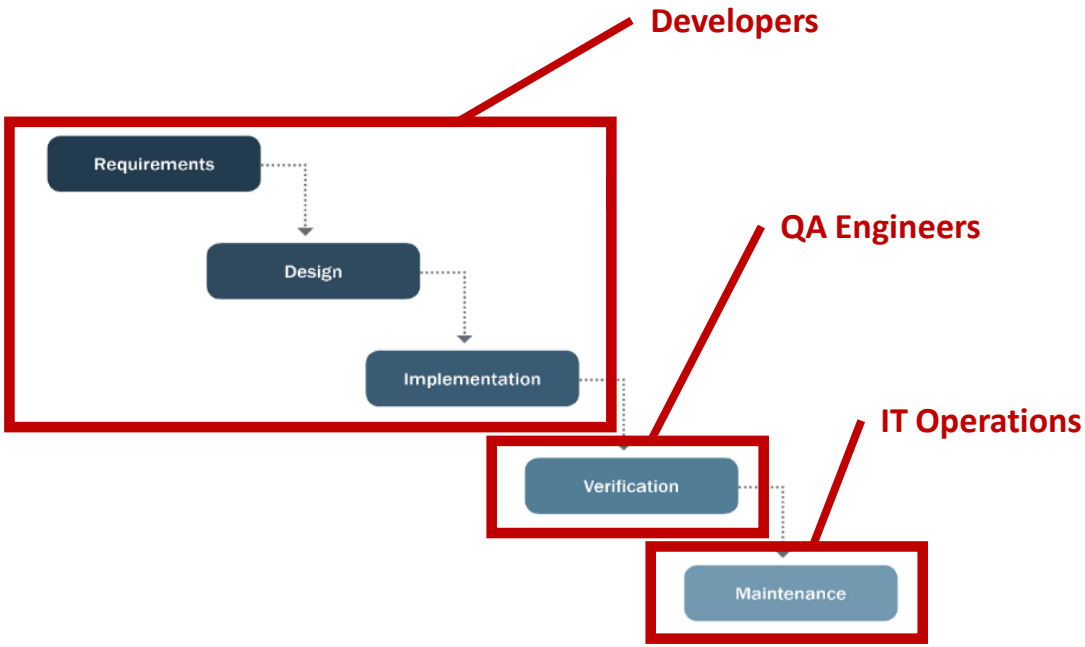
Collaboration: *Many stakeholders*



UNCLASSIFIED

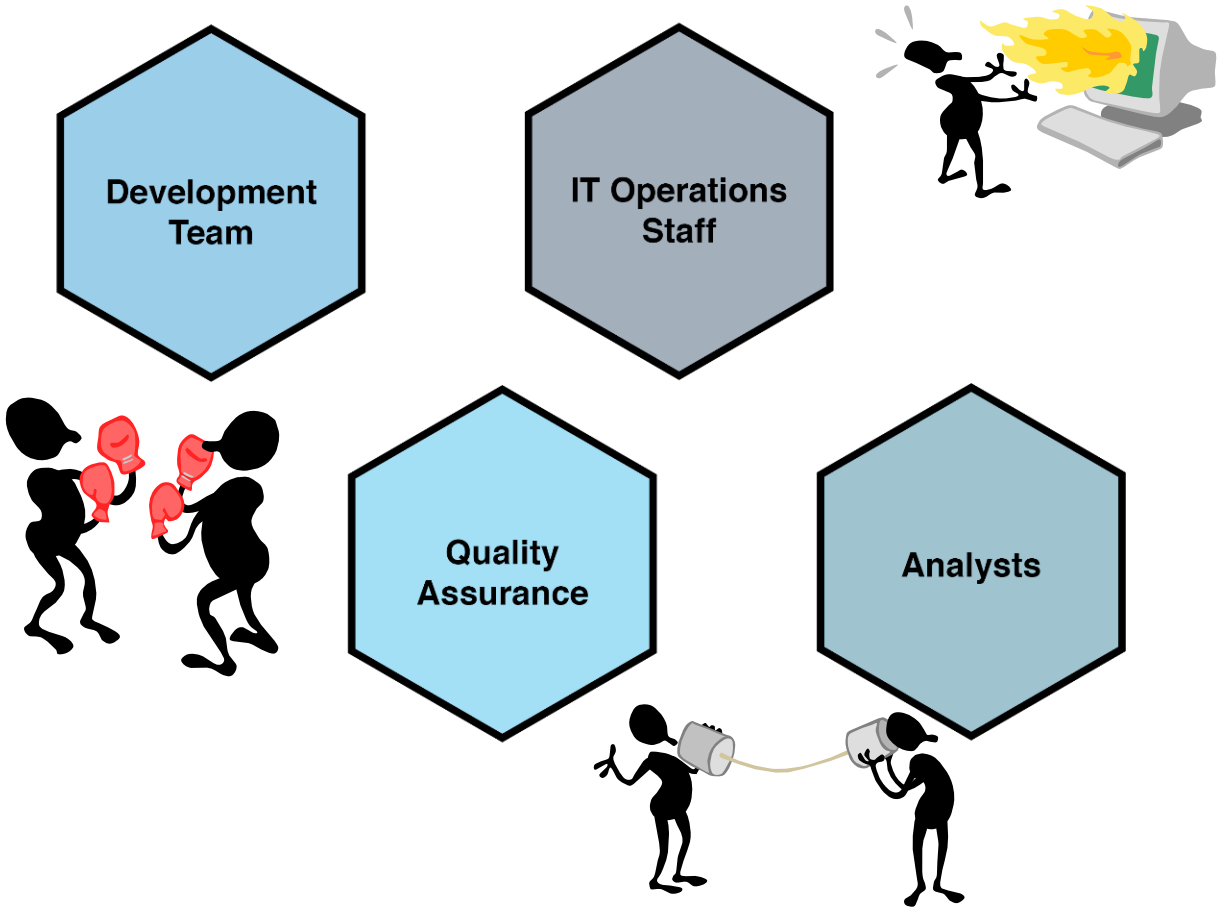
Collaboration: *Silos Reinforce Waterfall*

Teams have moved to Agile methodologies, but roles still align with waterfall methods



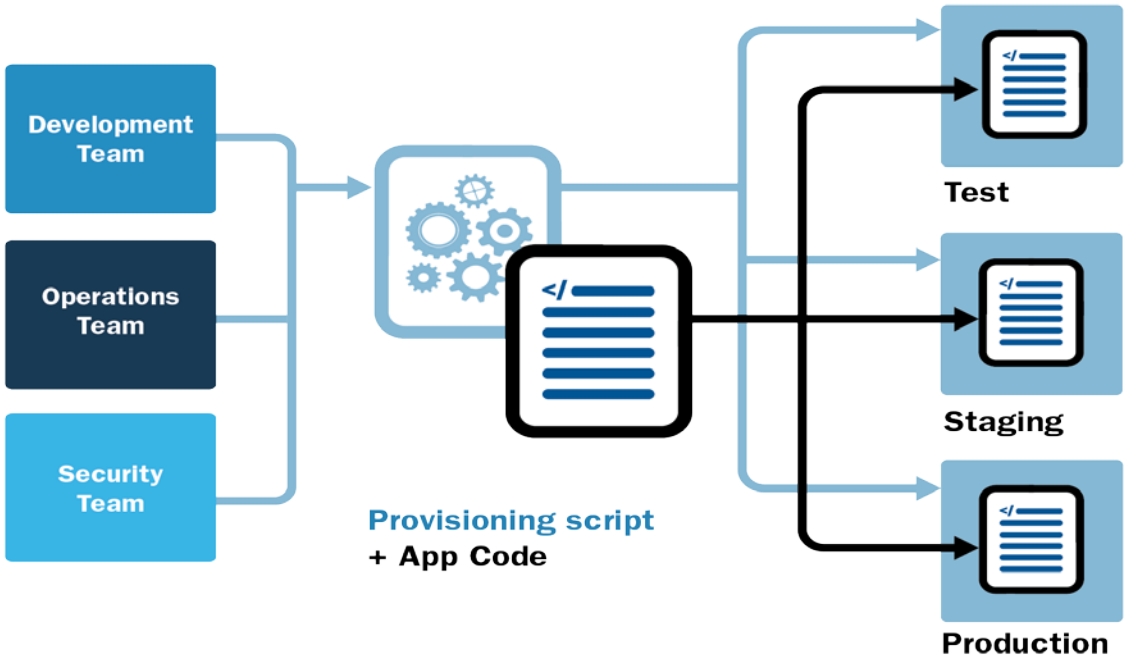
Collaboration:

Silos Inhibit Collaboration and poor communication



Infrastructure as Code (IaC)

A program that creates infrastructure,



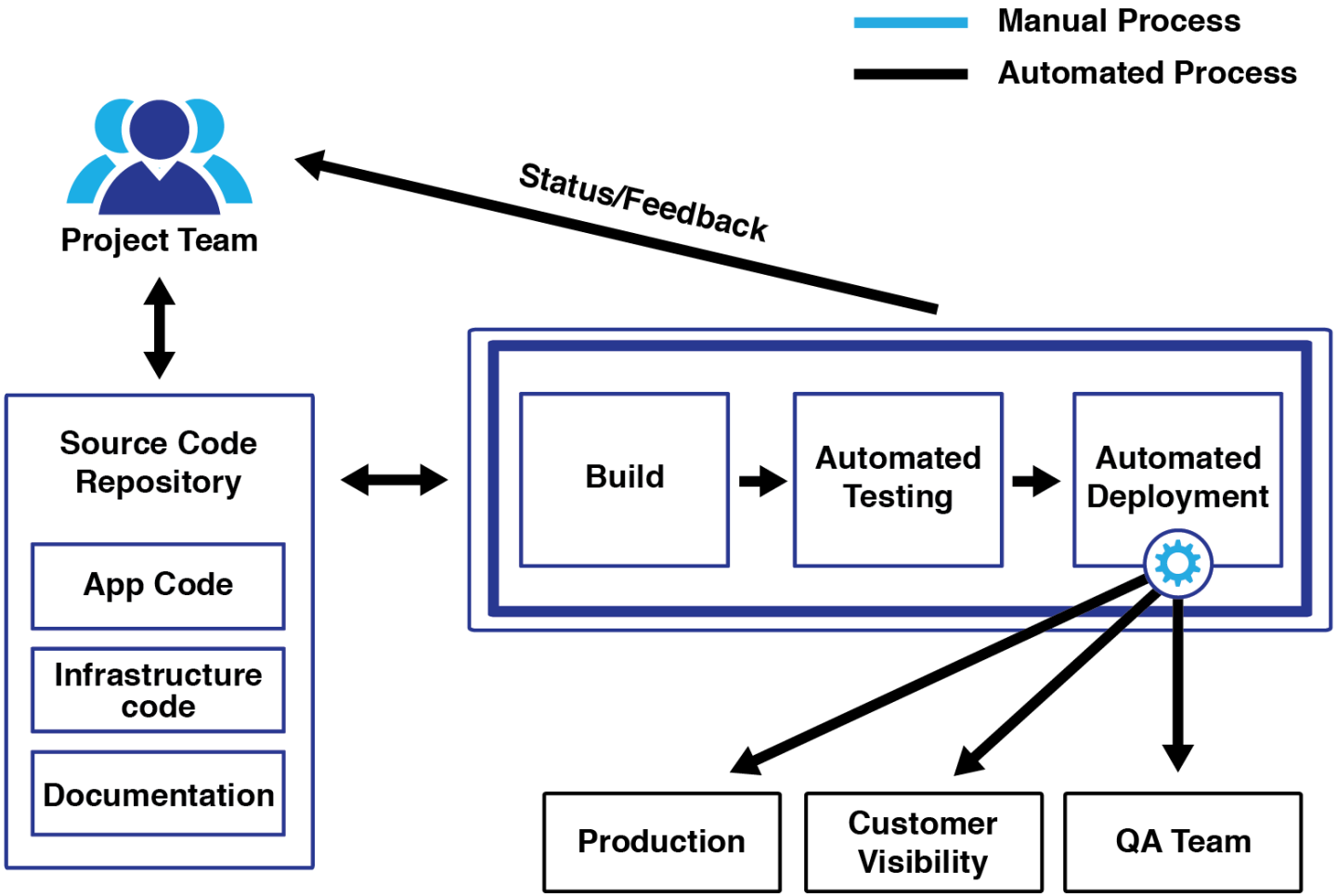
A concretely defined description of the environment is good material for conversation between team members.

IaC Promotes Quality Attributes

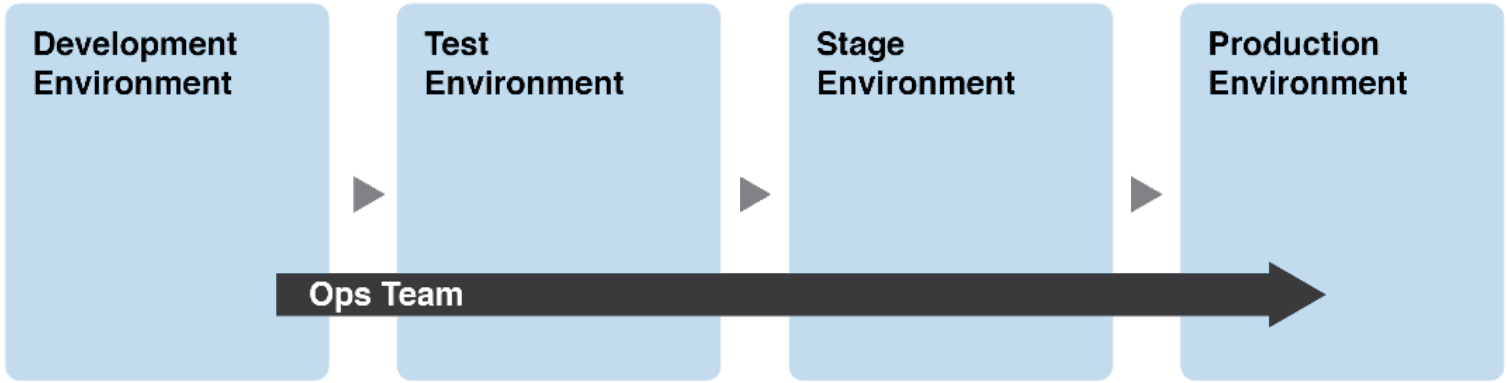
- The process that creates and configures the infrastructure for your application is, itself, an application.

- As an application, environment creation and configuration is now:
  - Automatable
  - Repeatable
  - Versionable
  - Reviewable
  - Diffable
  - Testable (it works)
  - Human-readable
  - Verifiable (it is right)

Automation : *Continuous Integration (CI)*



Automation : *Continuous Delivery /Deployment (CD)*



Shift Left Operational Concerns Enforced by Continuous Delivery

BLUF: People

Heavy collaboration between all stakeholders

- Secure Design / Architecture decisions
- Secure Environment / Network configuration
- Secure Deployment planning
- Secure Code Review

Constantly available open communication channels:

- Dev and OpSec together in all project decision meeting
- Chat/e-mail/Wiki services available to all team members

BLUF: Process

Establish a *process* to enable *people* to succeed using the *platform* to develop secure application

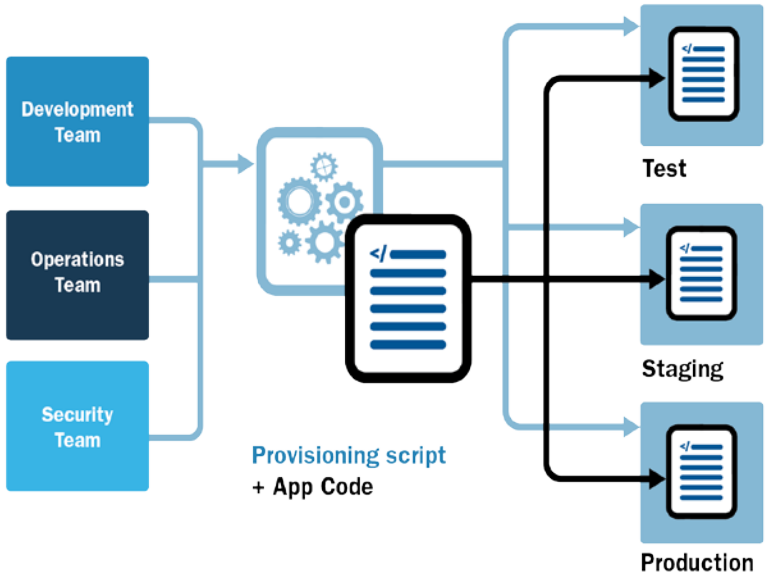
Such that;

- Constant communication and visible to all
- Ensures that tasks are testable and repeatable
- Frees up human experts to do challenging, creative work
- Allows tasks to be performed with minimal effort or cost
- Creates confidence in task success, after past repetitions
- Faster deployment , frequent quality release

BLUF: Platform

Where *people* use *process* to build secure software

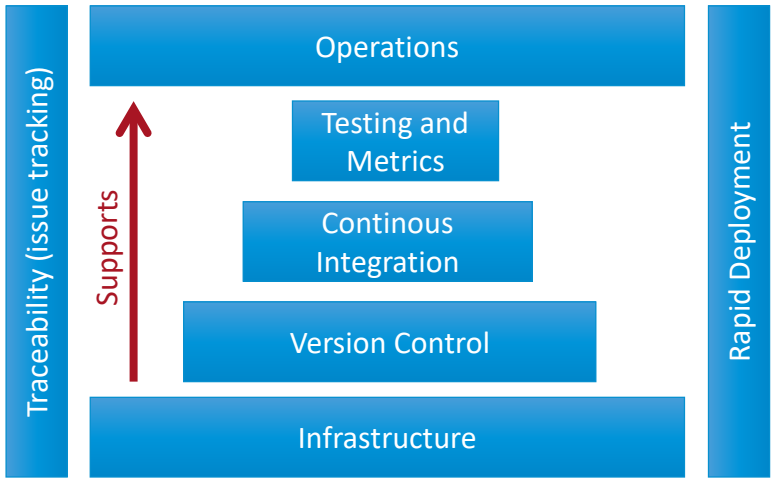
- Automated environment creation and provisioning
- Automated infrastructure testing
- Parity between Development, QA, Staging, and Production environments
- Sharing and versioning of environmental configurations
- Collaborative environment between all stakeholders



Start on building DevOps Pipeline:

Tool Landscape Complexity (~180 tools)

- Release configuration and release software (e.g., Puppet, Chef)
- Scripts and code used to release software (e.g., Python scripts)
- Servers, network or other infrastructure that support release tools
- Software and tools to support developer self-service operations
- External test frameworks (e.g., Jersey Test Framework)
- External operational monitoring and log mining tools (e.g., Splunk)
- Source code repositories (e.g., Git)
- Issue tracking systems (e.g., JIRA)
- Container driven tools (e.g., Docker)
- Rqmts mgmt. (Doors, Blueprint)
- Infrastructure and cloud providers
- IDEs integrated DevOps process



Engineering the Deployment Pipeline is a challenge

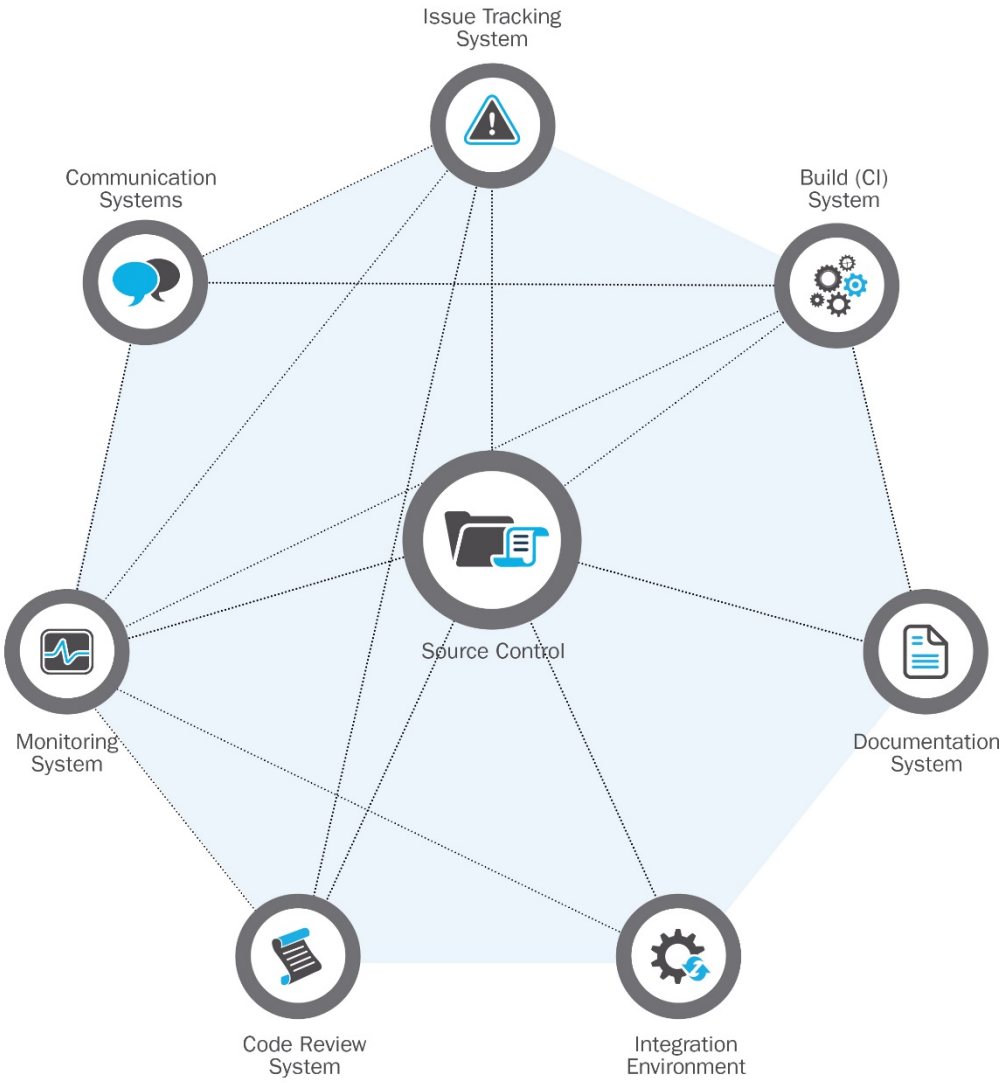
- If pipeline is not engineered, it may require extensive effort integrate tools and share data across the pipeline
- Key questions related to designing the integrated pipeline include:
  - Who owns the integrated deployment pipeline?
  - How/what to measure/monitor to assess pipeline health?
  - What are the key qualities attributes teams should look for as they select tools for pipeline integration?
- Whether designing or buying it is important to understand the end-to-end requirements (e.g., workflow visibility)

Key Quality Attributes

- Integrate-ability
- Interoperability
- Usability
- Portability
- Resilience
- Security/Permissions
- Availability (Error handling)
- Scalability

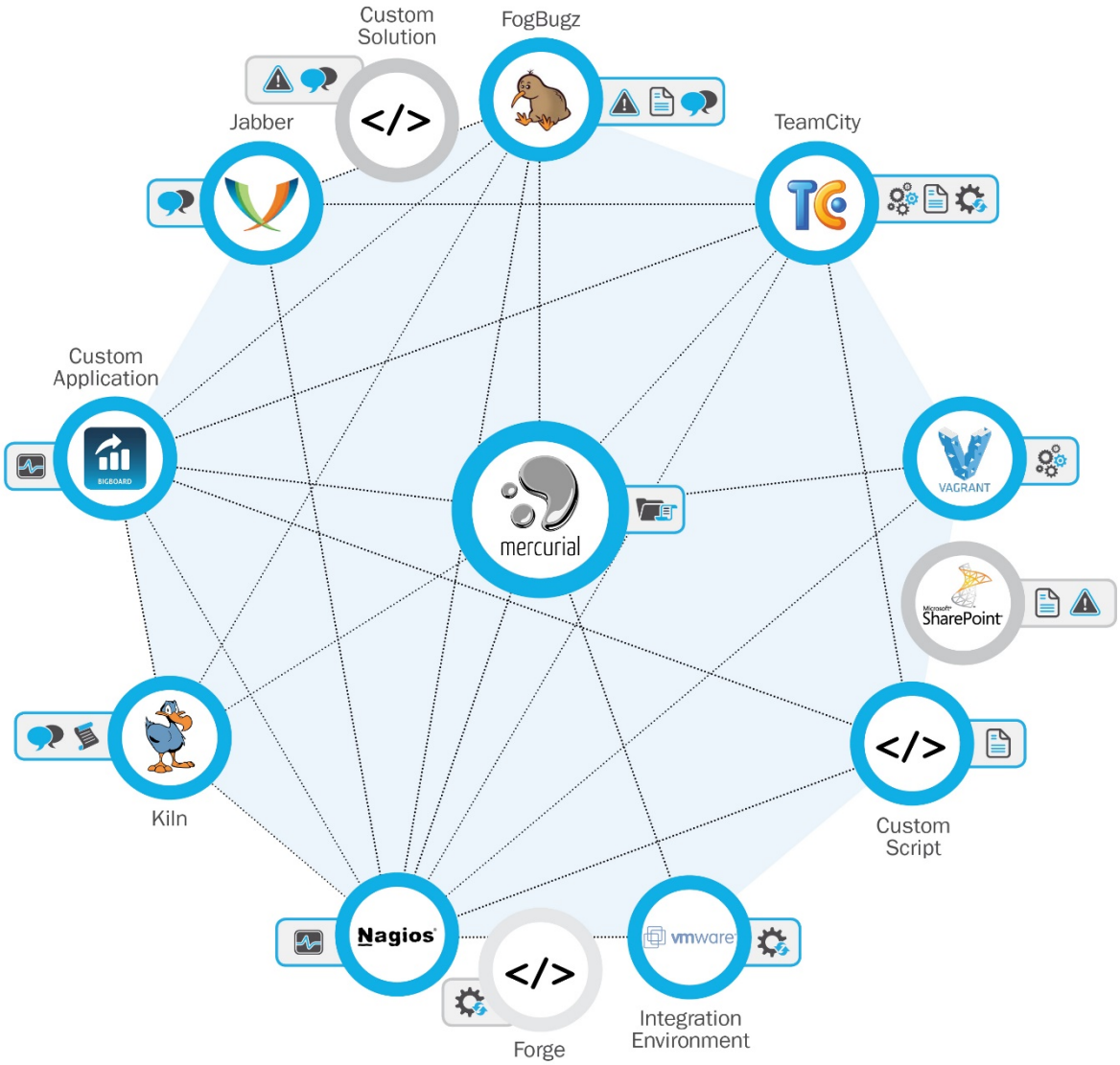
- Performance
- Modifiability
- Configurability
- “Automate-ability” (of manual tasks)
- “Approvability” (allows for manual approval)
- Measurability?
- Other?

Integrated Pipeline - General

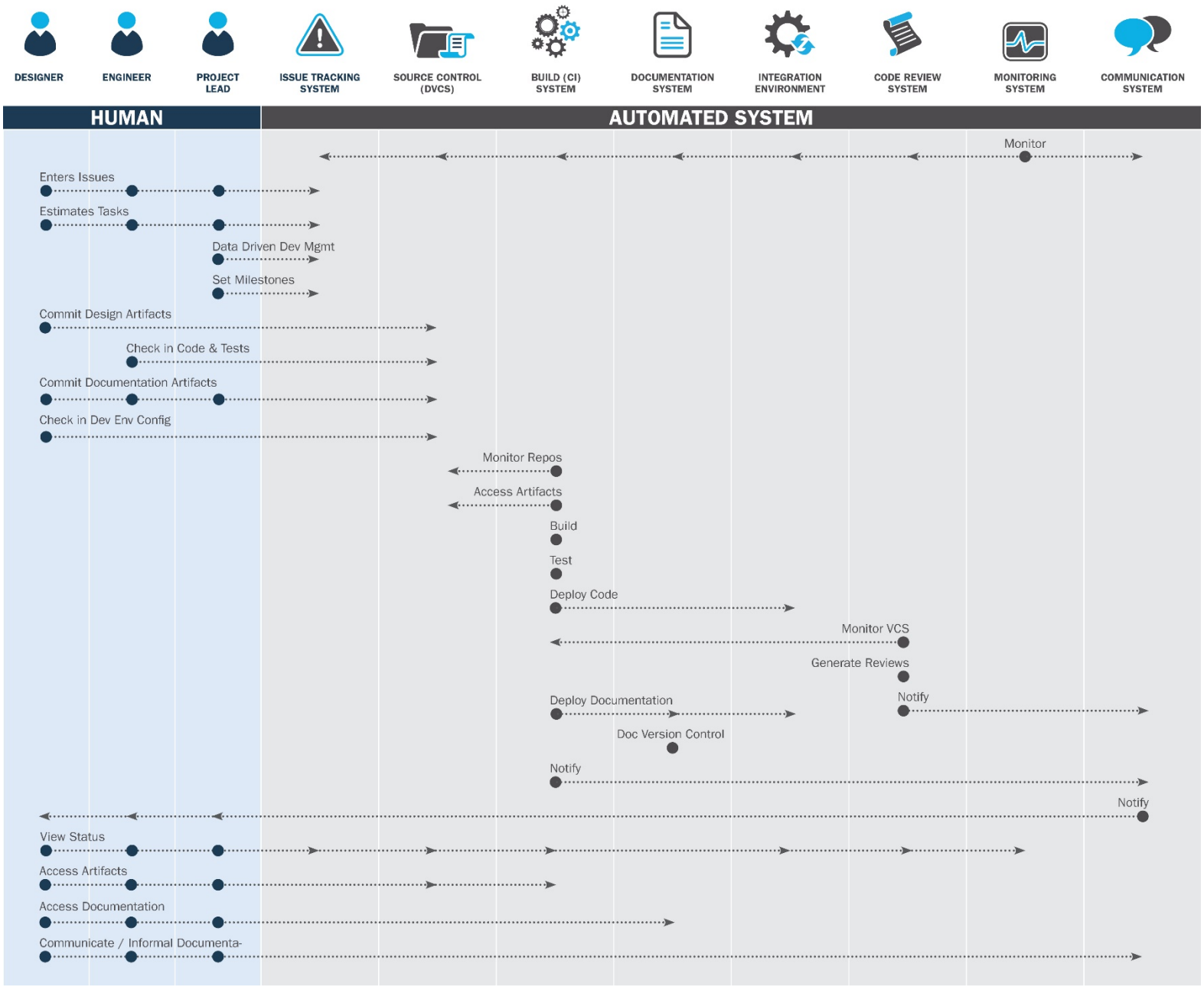


UNCLASSIFIED

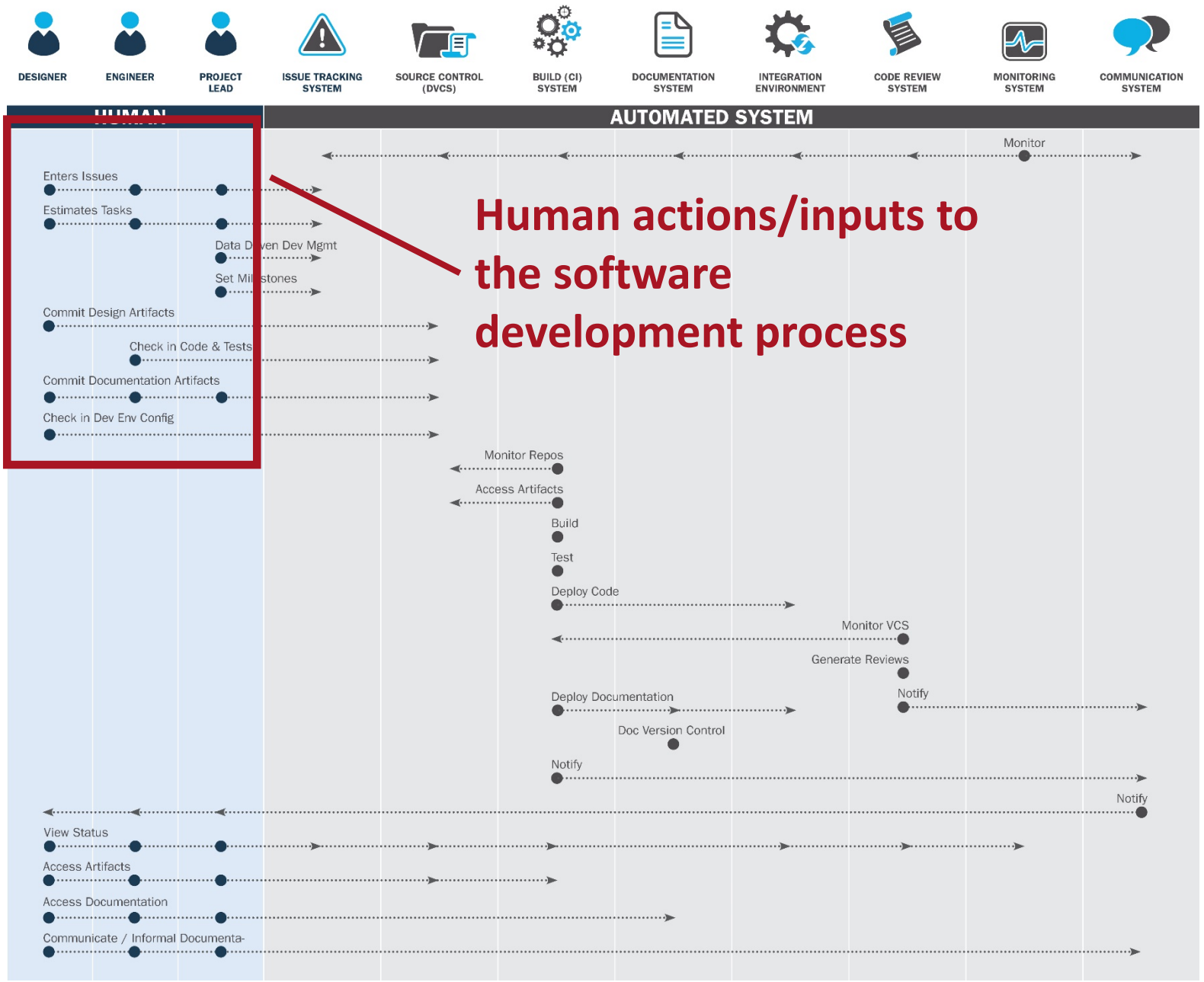
Integrated Pipeline – With Tooling



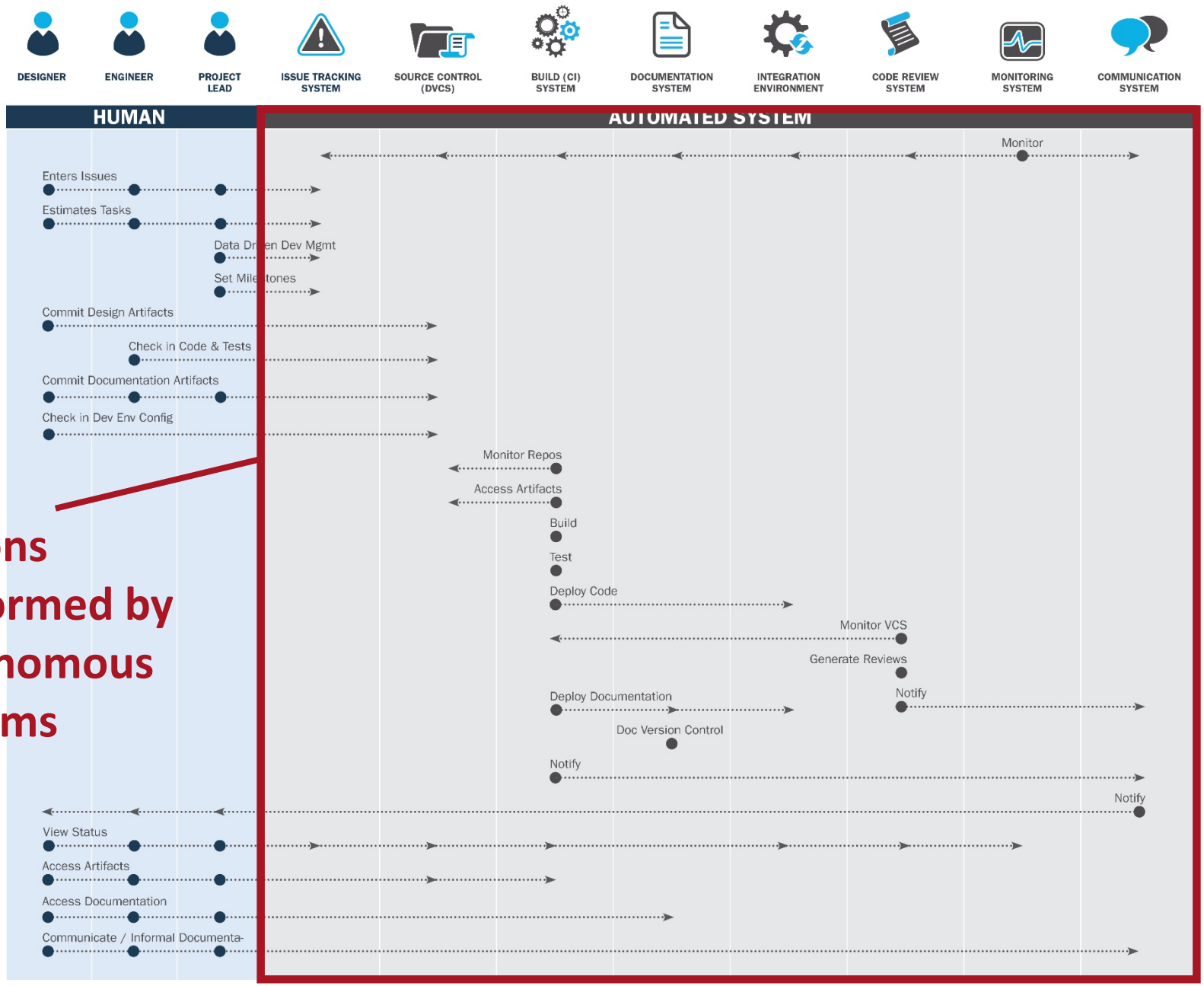
UNCLASSIFIED



UNCLASSIFIED



UNCLASSIFIED



Actions performed by autonomous systems

UNCLASSIFIED

Topics

- DevOps Fundamentals
- *DevOps and Security*
- Platform Security
- AppSec - Secure DevOps
- DevOps Anti-Patterns

DevOps and Security

UNCLASSIFIED

DevOps and Security

UNCLASSIFIED

Rugged{Secure}Dev{Sec}Ops

- DevOps is a Risk Mitigation strategy, built on Situational Awareness, Automation, and Repetition
  - But security is where a lot of DevOps implementations fall down
- Goal:
  - Protecting private user data
  - Restricting access to data / systems
  - Protecting company data / IP
  - Standards compliance
  - Safeguarding disposition / transition

The Rugged Manifesto

I am rugged and, more importantly, my code is rugged.

I recognize that **software** has become a **foundation of our modern world**.

I recognize the awesome **responsibility** that comes with this foundational role.

I recognize that my code will be used in ways I cannot anticipate, in ways it was not designed, and for longer than it was ever intended.

I recognize that my code **will be attacked** by talented and persistent **adversaries** who **threaten** our physical, economic and **national security.**

I recognize these things – and I choose to be rugged.

I am rugged because **I refuse to be a source of vulnerability or weakness**.

I am rugged because I assure my code will support its mission.

I am rugged because my code can **face these challenges and persist** in spite of them.

I am rugged, not because it is easy, but because it is **necessary** and I am up for the challenge.

Rugged Continued …

Culture – NOT a tool, SDLC, or org structure

Rugged != Secure - secure is only an instant in time

Proactive security is better than reactive – Reactive will fail eventually

Team Composition

Developers

- Features
- Quality Attributes
- Efficiency
- Performance
- Users
- Authentication
- Authorization

IT Ops

- Deployment
- Maintenance
- Updates
- Change policy
- Failure
- Data loss
- Risk prevention

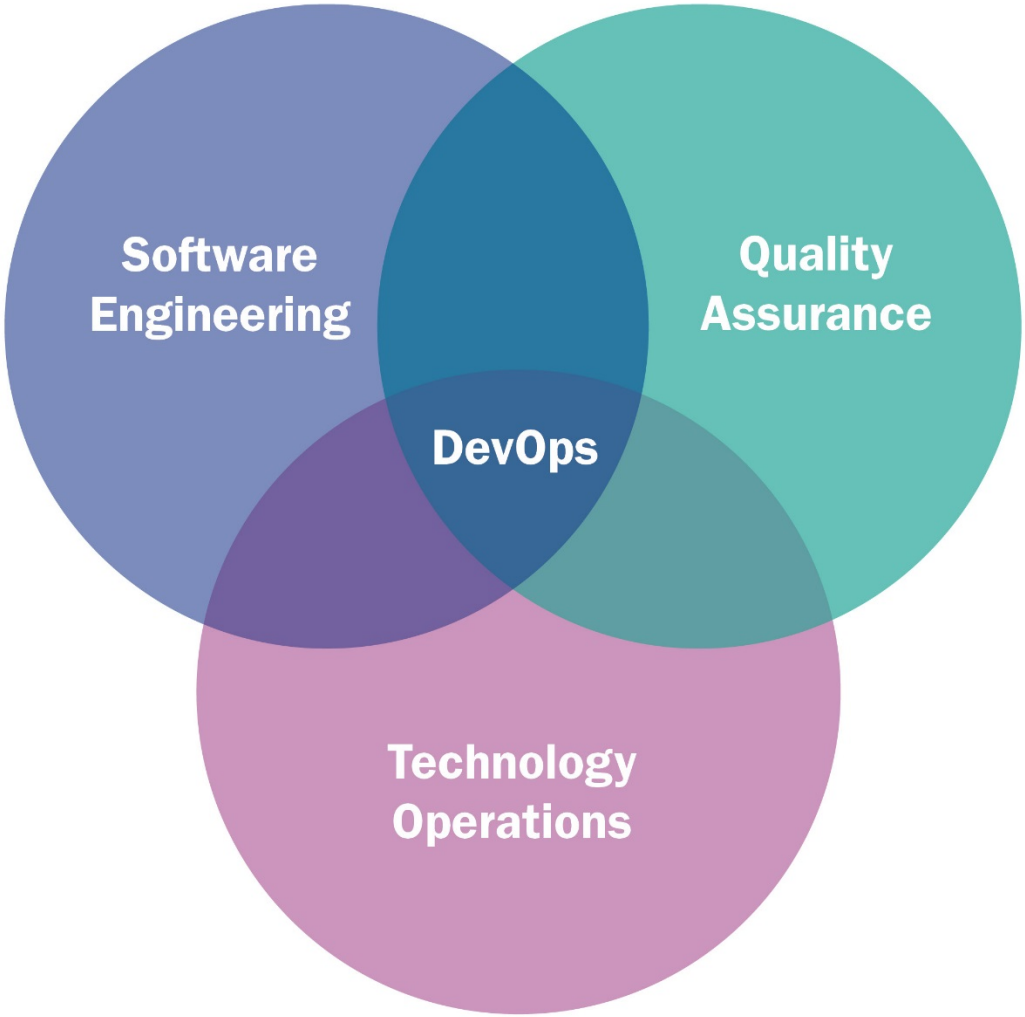
QA

- Testable
- Issue tracking
- Bug Reports
- Usability
- Help Desk

Security Team

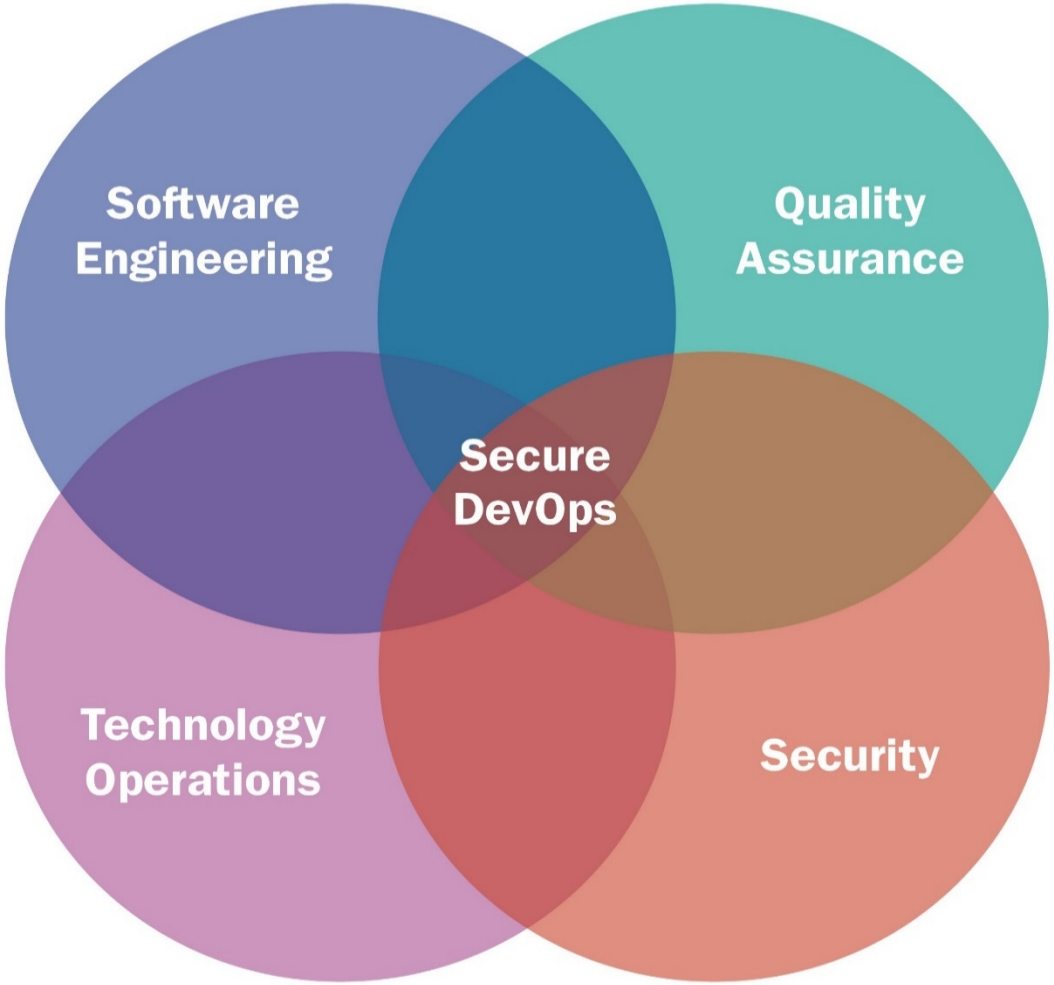
- Data Privacy
- Intrusion detection
- Threat vectors
- CVEs
- Package security
- Authentication
- Authorization
- Security Standards Compliance

DevOps: Multiple Team Integrations



DevOps: Multiple Team Integrations + *With Security Team*

DevOps: Multiple Team Integrations + *With Security Team*



UNCL

Topics

- DevOps Fundamentals
- DevOps and Security
- *Platform Security*
- AppSec - Secure DevOps
- DevOps Anti-Patterns

Deployment Pipeline & Dependencies

- Deployment Pipeline
  - Development Environment
  - Scripting/ Automation
  - Integration and configuration
  - Build Servers
  - Monitoring
  - Code Repositories
  - Container Security
- Supply Chain/Dependencies
  - 3rd party libraries
  - Vulnerability analysis
  - Trusted sources
  - Open source software
  - Code Snippets
  - Application ready frameworks

Evolution of software development

Custom development – context:

- Software was limited
  - Size
  - Function
  - Audience
- Each organization employed developers
- Each organization created their own software

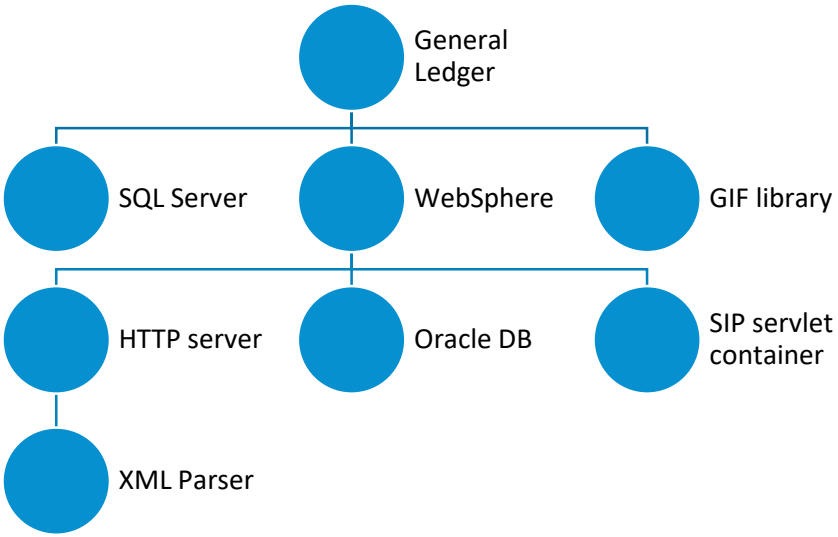
Supply chain: practically none

Shared development – ISVs (COTS) – context:

- Function largely understood
  - Automating existing processes
- Grown beyond ability for using organization to develop economically
- Outside of core competitiveness by acquirers

Supply chain: software supplier

Development is now assembly



Note: hypothetical application composition

Collective development – context:

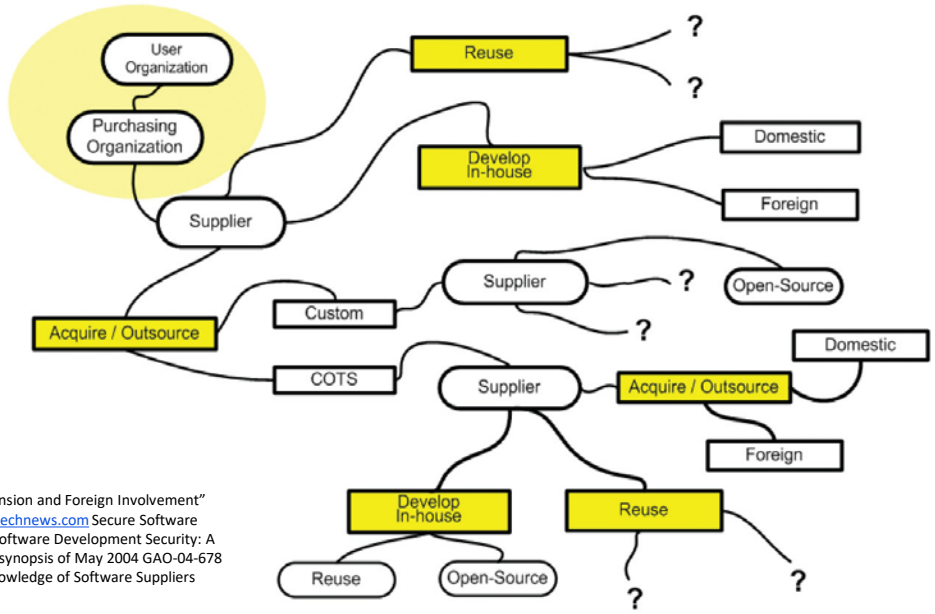
- Too large for single organization
- Too much specialization
- Too little value in individual components

Supply chain: long

Software Asset Management (SAM) for assembled software is a *challange*

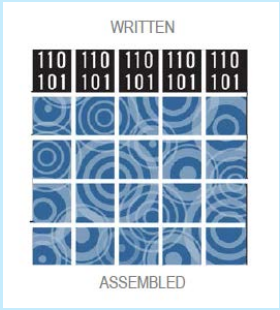
Expanding the scope and complexity of acquisition and deployment

Visibility and direct controls are limited (only in shaded area)



Source: "Scope of Supplier Expansion and Foreign Involvement" graphic in DACS www.softwaretechnews.com Secure Software Engineering, July 2005 article "Software Development Security: A Risk Management Perspective" synopsis of May 2004 GAO-04-678 report "Defense Acquisition: Knowledge of Software Suppliers Needed to Manage Risks"

Substantial open source and 3rd party apps contained in SAM



- 90% of modern applications are assembled from 3rd party components
  - At least 75% of organizations rely on open source as the foundation of their applications
- Most applications are now assembled from hundreds of open source components, often reflecting as much as 90% of an application

Distributed development – context:

- Amortize expense
- Outsource non-differential features
- Lower acquisition (CapEx) expense

Supply chain: opaque

Sources: Geer and Corman, "Almost Too Big To Fail," ;login: (Usenix), Aug 2014; Sonatype, 2014 open source development and application security survey

Reducing SAM Risk

- Open source software usage policy
- Supplier resource management
- Apply policy and management through DevOps pipeline
- Automate and monitor dependencies management
- Track build and deploy decencies list
- Apply discovered(new) vulnerabilities and deployment process
- Continues training and monitoring developers activities

SAM Hygiene: Recommendations

- Supplier security commitment evidence
  - Supplier employees are educated as to security engineering practices
  - Supplier follows suitable security design practices

- Evaluate a product's threat resistance
  - What product characteristics minimize opportunities to enter and change the product's security characteristics?

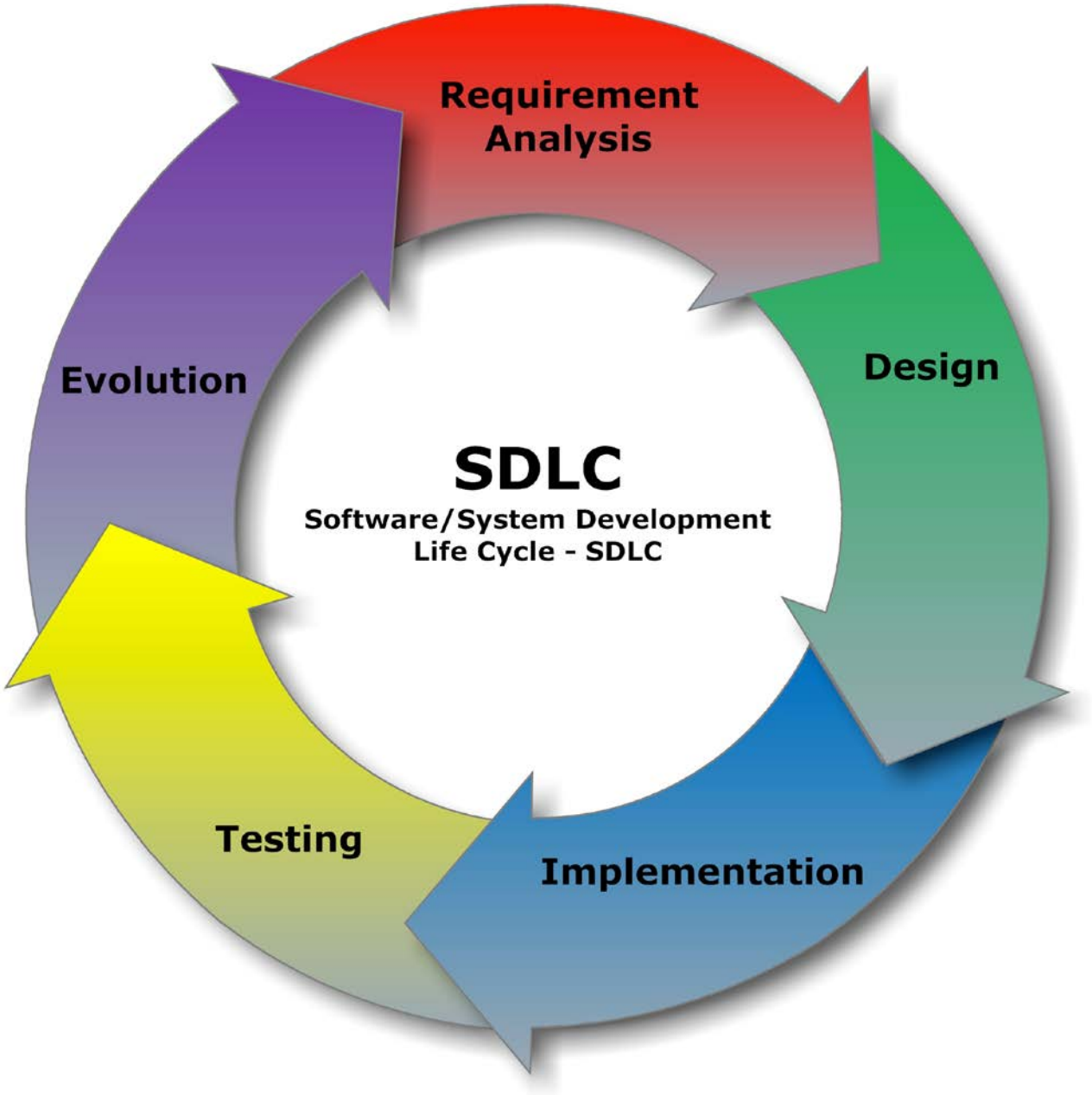
- Create a centralized private repositories of vetted 3rd party components for all developers

- Establish good product distribution practices
  - Recognize that supply chain risks are accumulated
  - Monitor for new vulnerabilities and know where they are in the enterprise to fix

- Minimize variation of components to make things easier (multiple versions, duplicated utility)

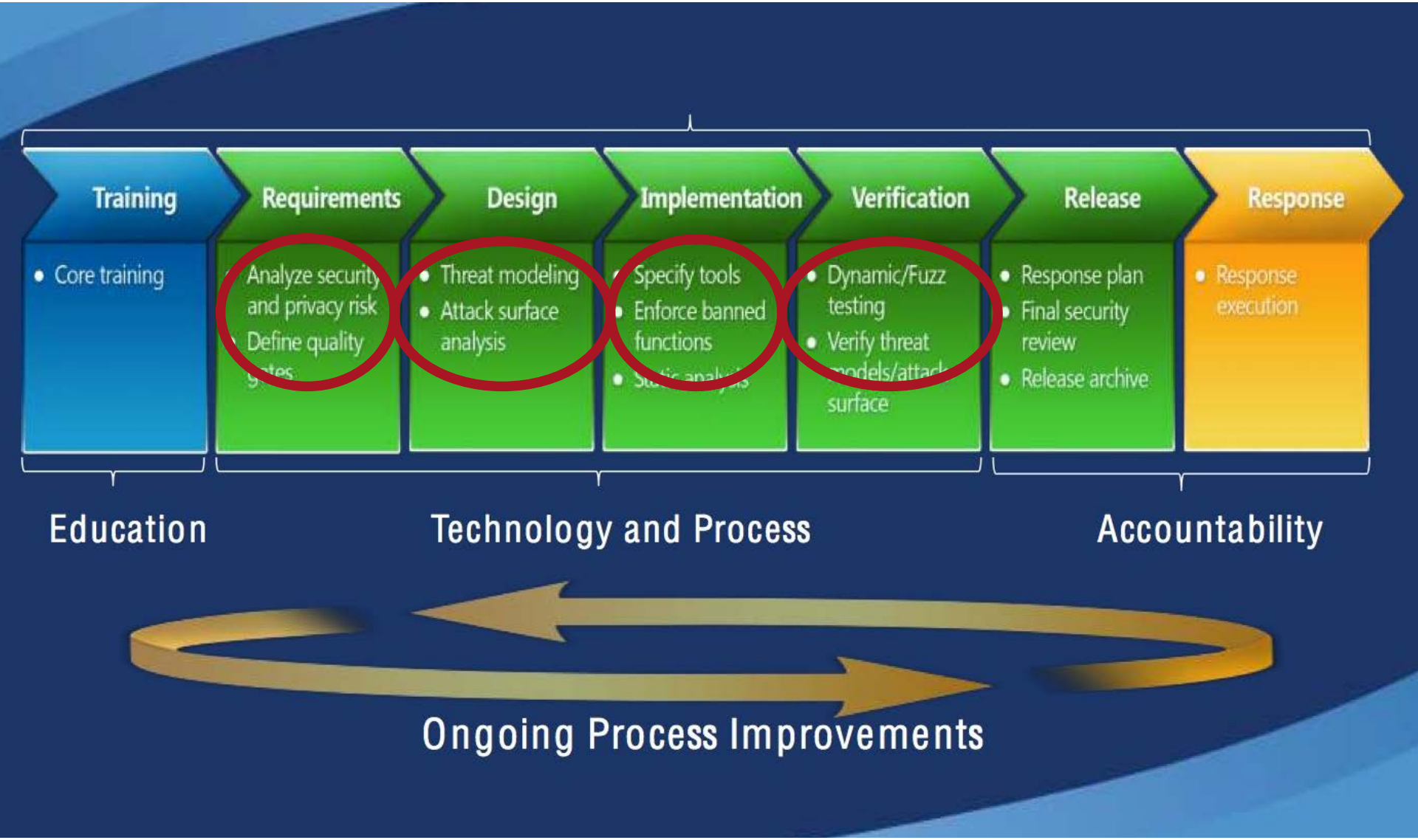
Topics

- DevOps Fundamentals
- DevOps and Security
- Platform Security
- *AppSec - Secure DevOps*
- DevOps Anti-Patterns

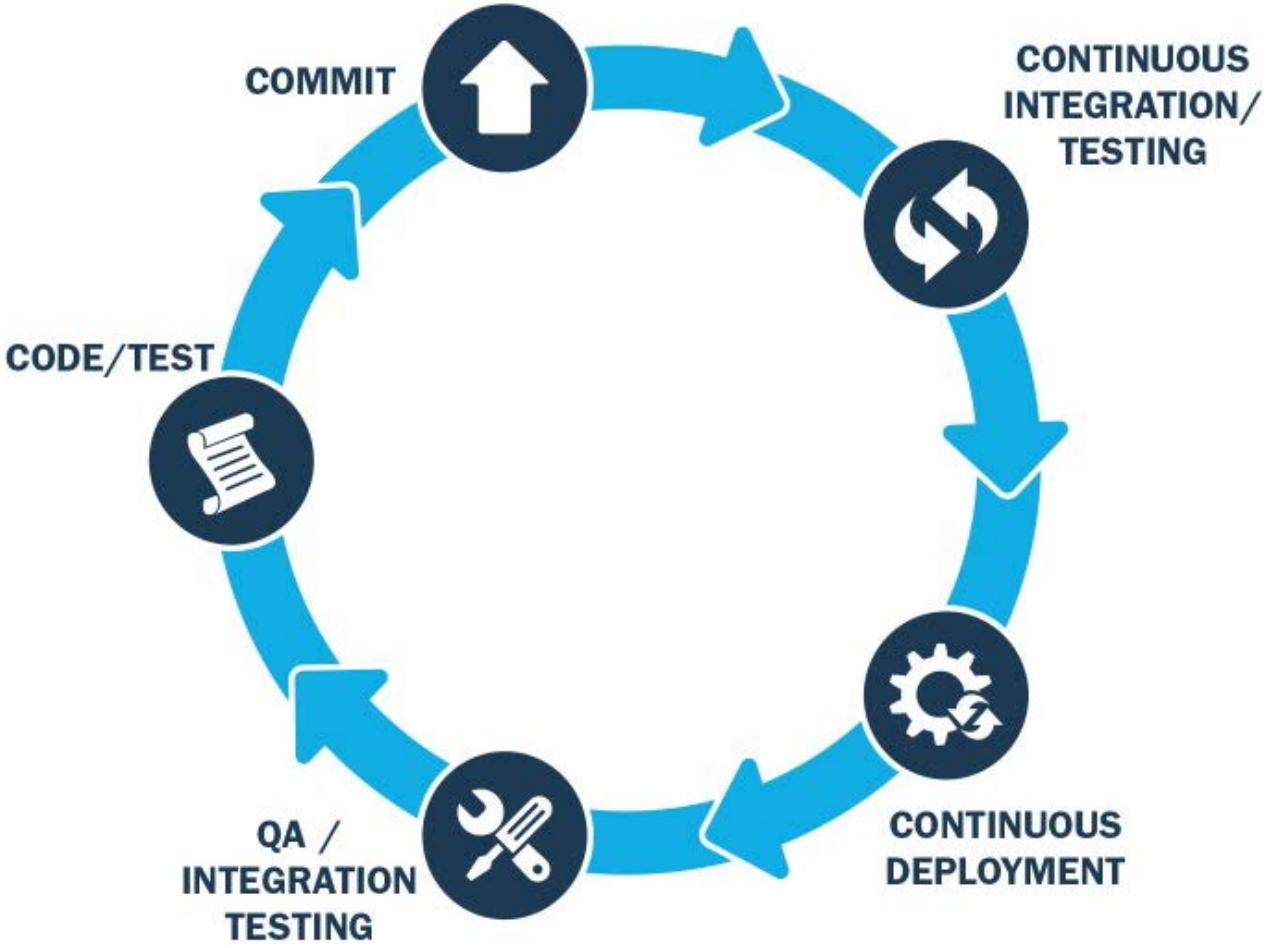


UNCLASSIFIED

Microsoft Secure Development Lifecycle (SDL)

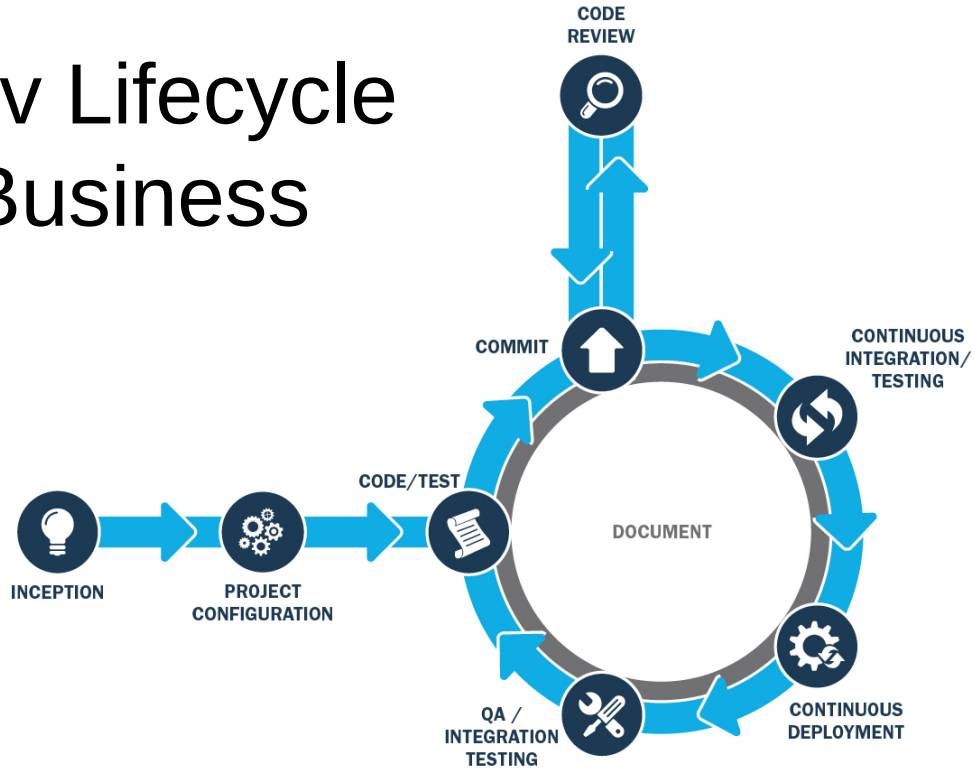


Development Lifecycle



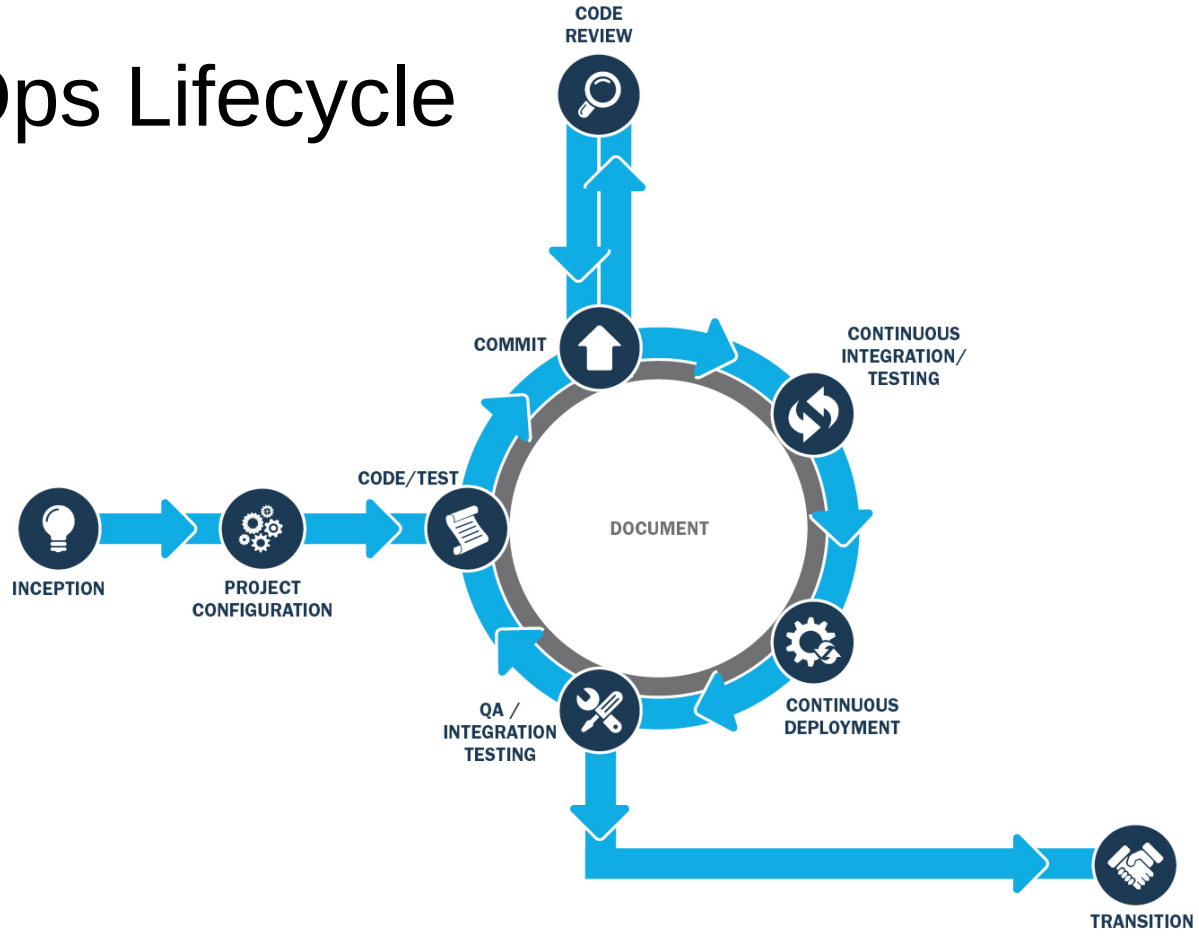
Enhancing the SDLC

Dev Lifecycle
+ Business



Enhancing the SDLC

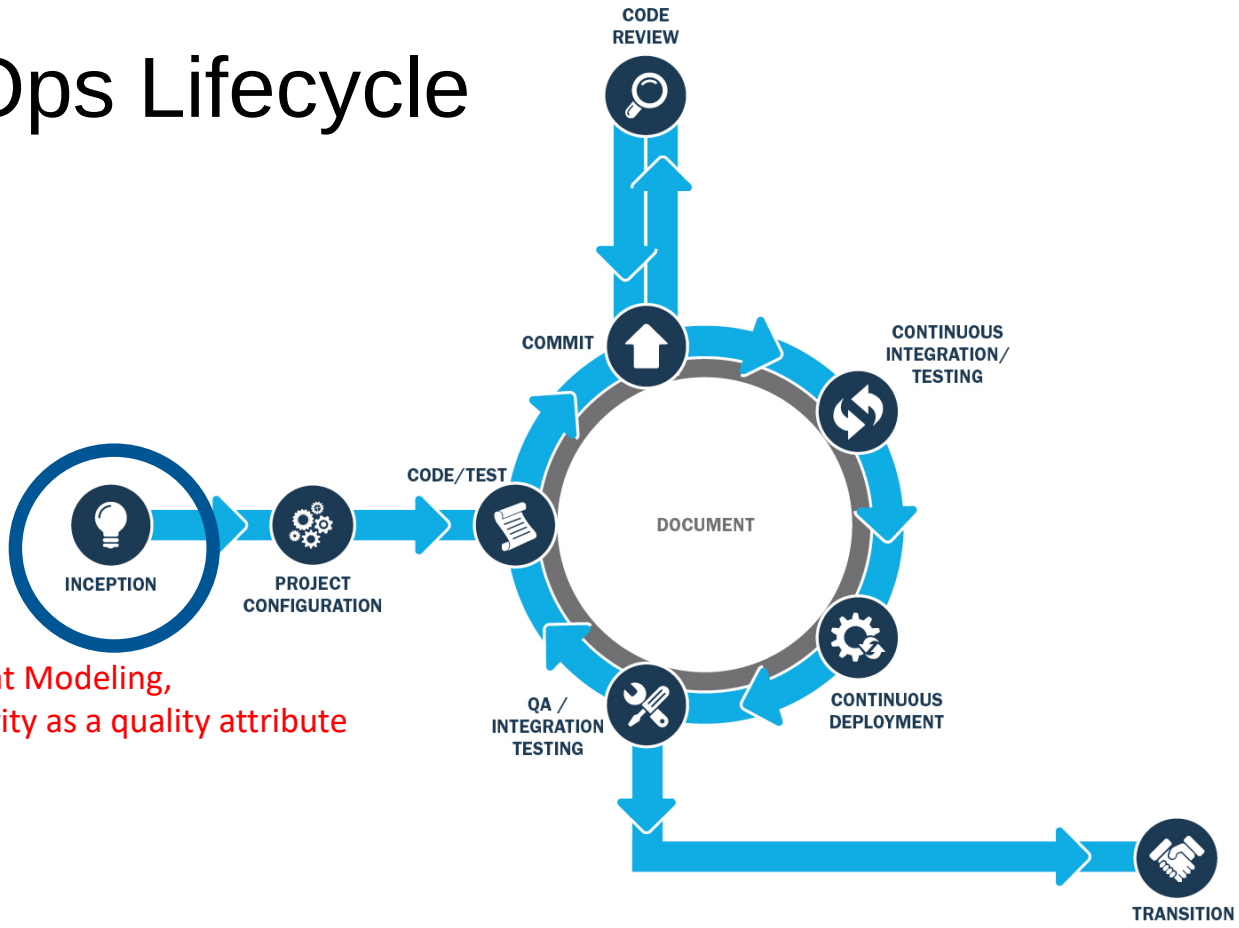
DevOps Lifecycle



Where are opportunities for security processes?

Enhancing SDLC Security

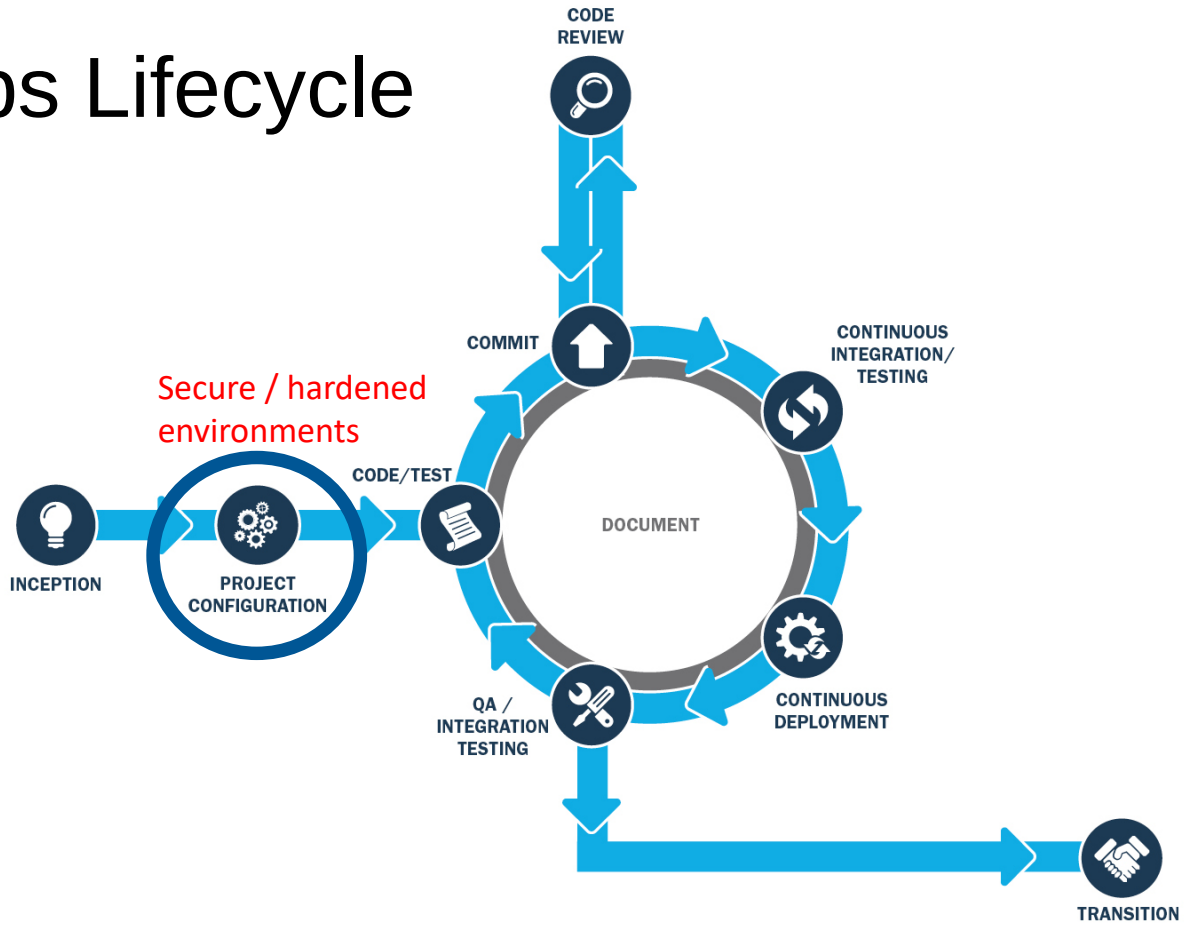
DevOps Lifecycle



Threat Modeling,
Security as a quality attribute

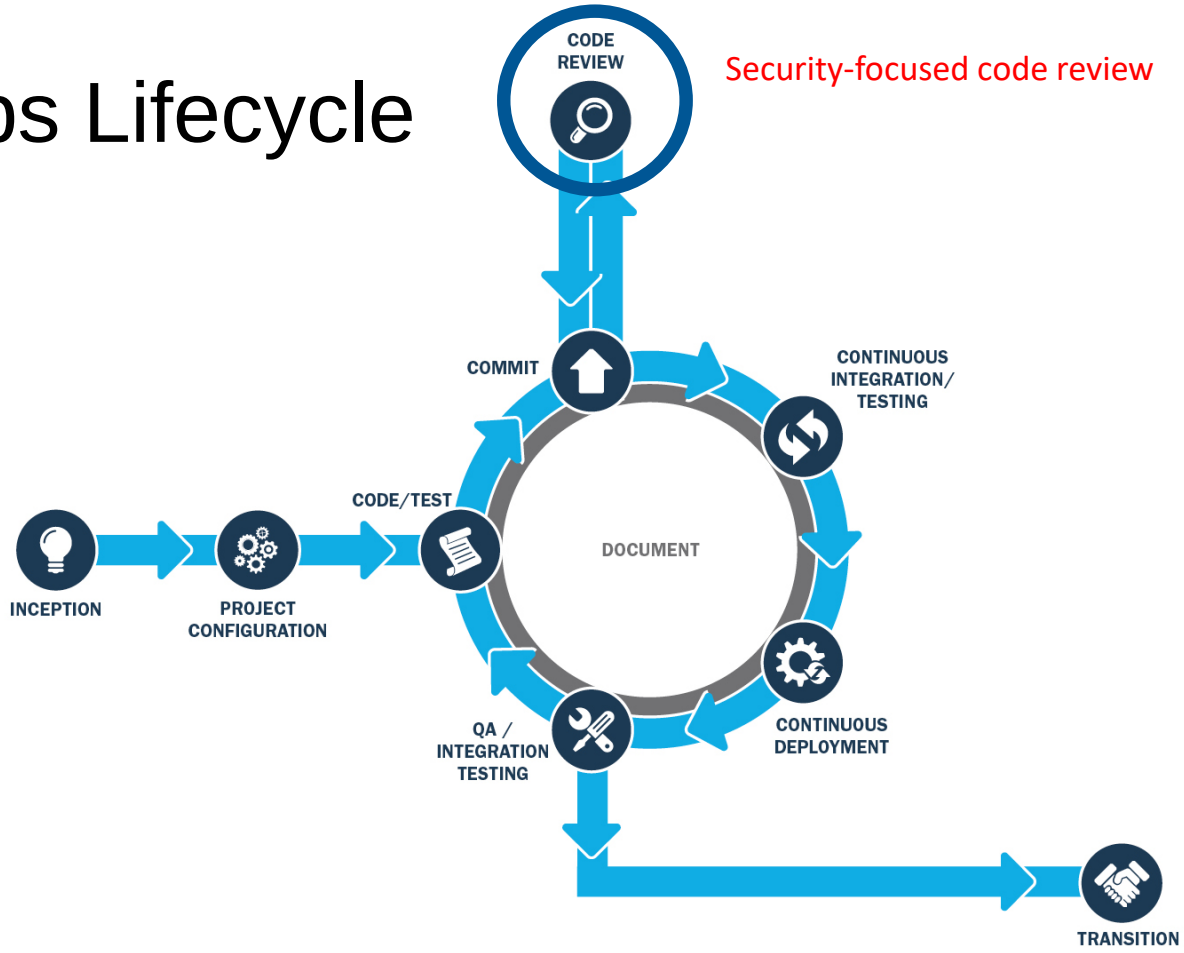
Enhancing SDLC Security

DevOps Lifecycle



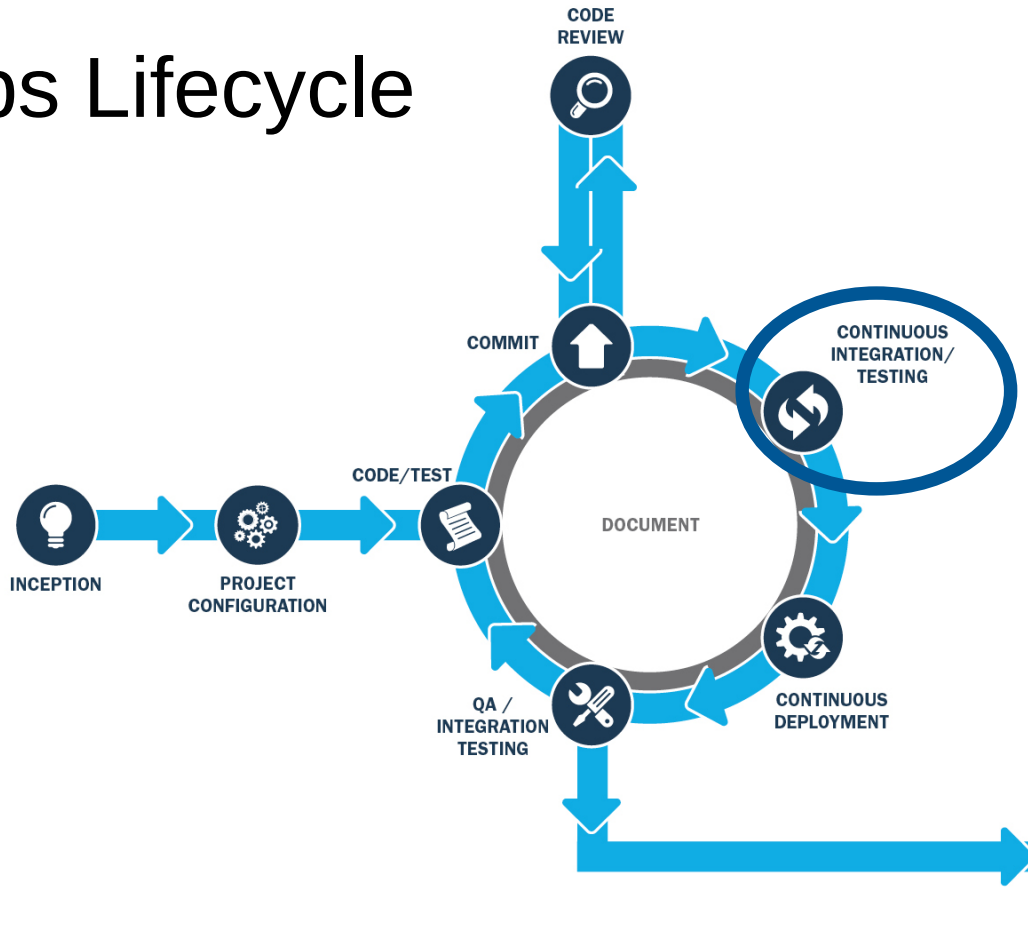
Enhancing SDLC Security

DevOps Lifecycle



Enhancing SDLC Security

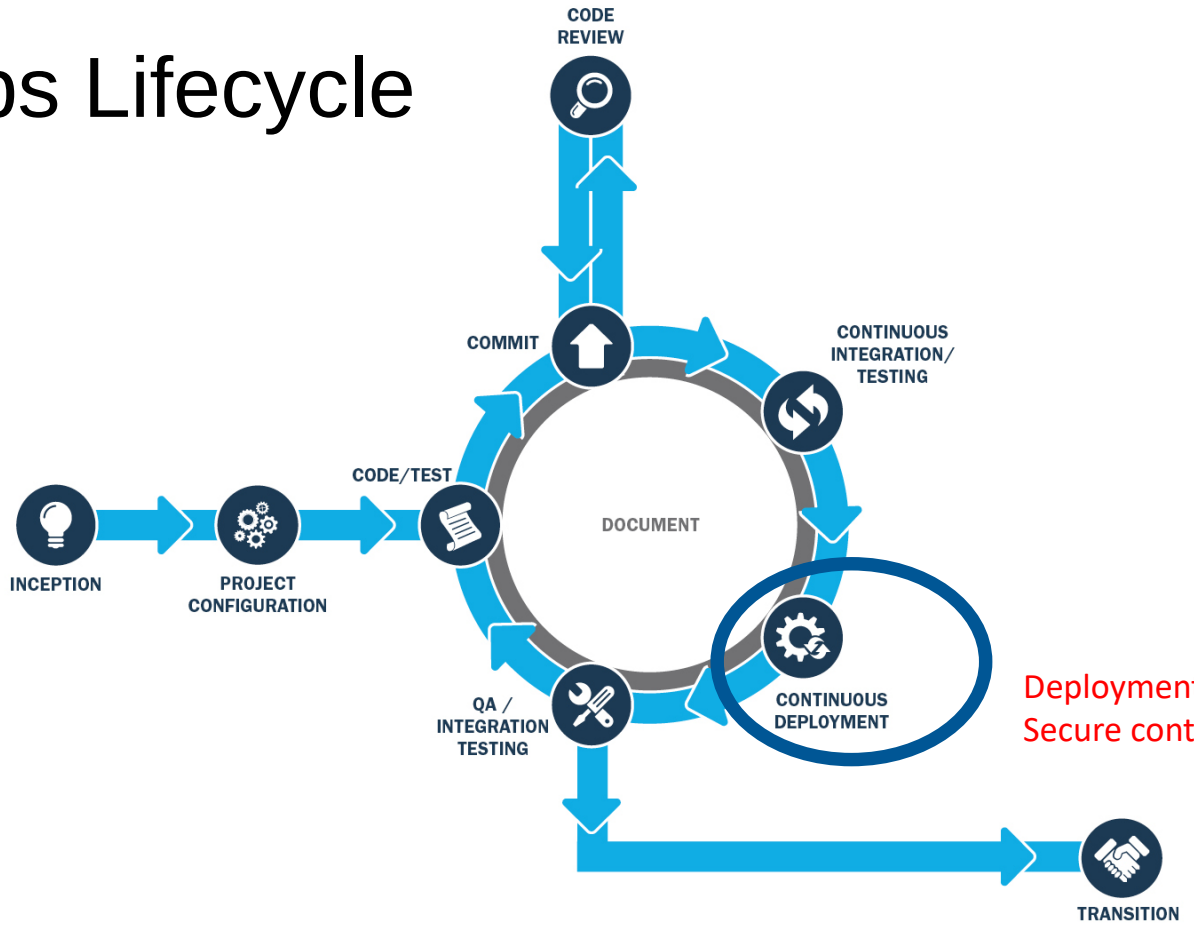
DevOps Lifecycle



Automated Security Testing (Static analysis, etc)

Enhancing SDLC Security

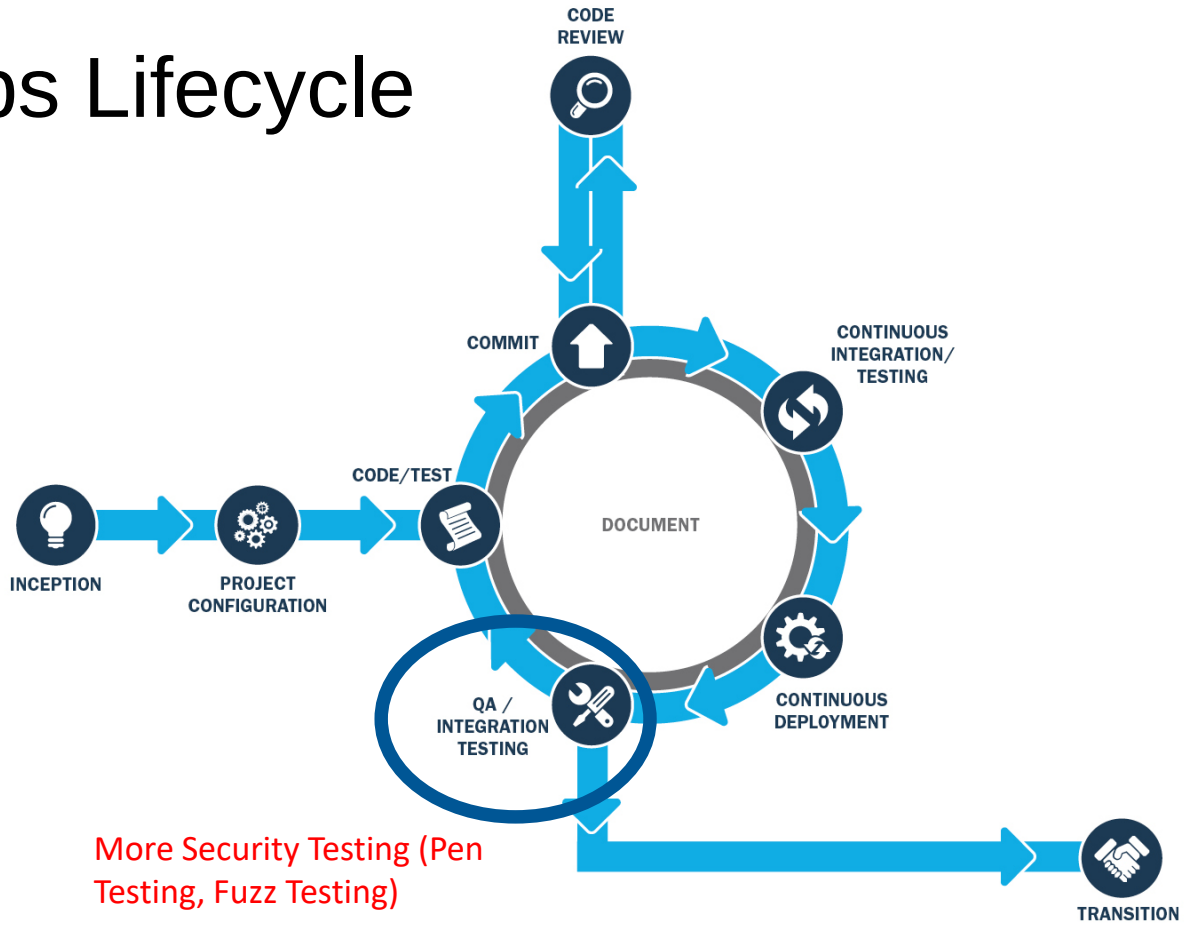
DevOps Lifecycle



Deployment authentication
Secure container, packages

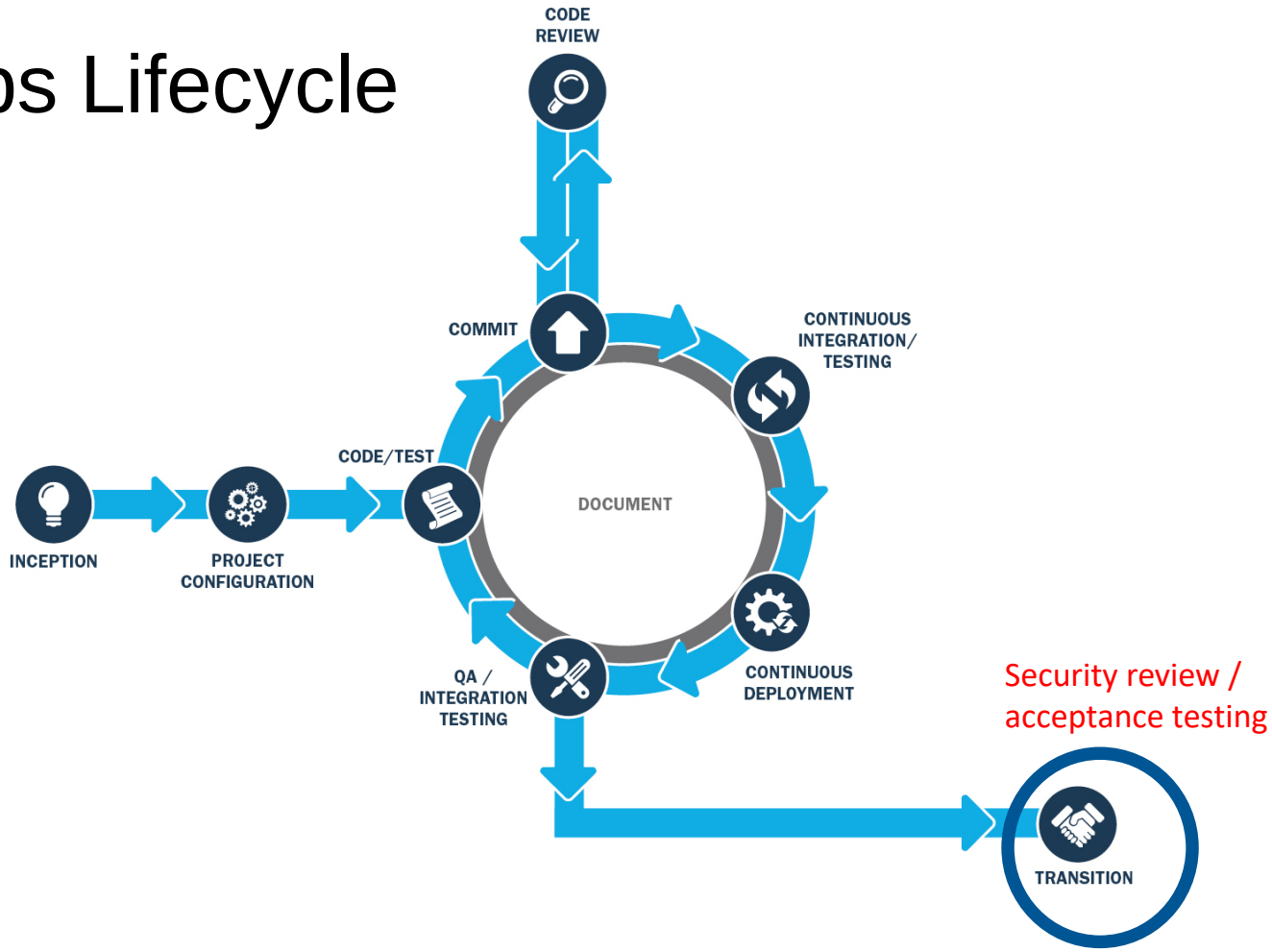
Enhancing SDLC Security

DevOps Lifecycle



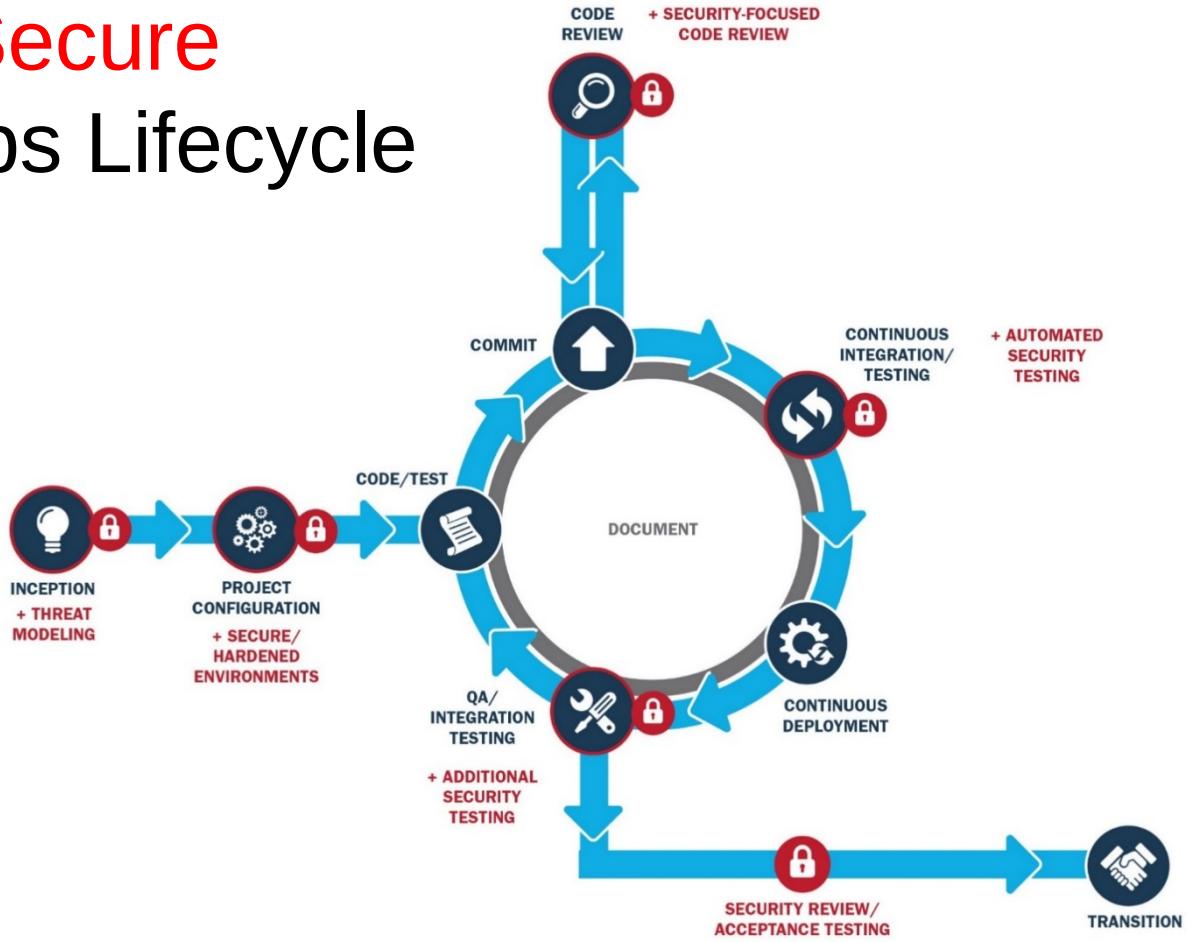
Enhancing SDLC Security

DevOps Lifecycle



Enhancing SDLC Security

Secure DevOps Lifecycle



UNCLASSIFIED

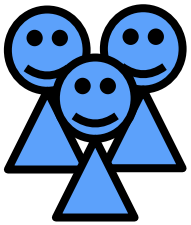
Security must be addressed without breaking the rapid delivery, continuous feedback model

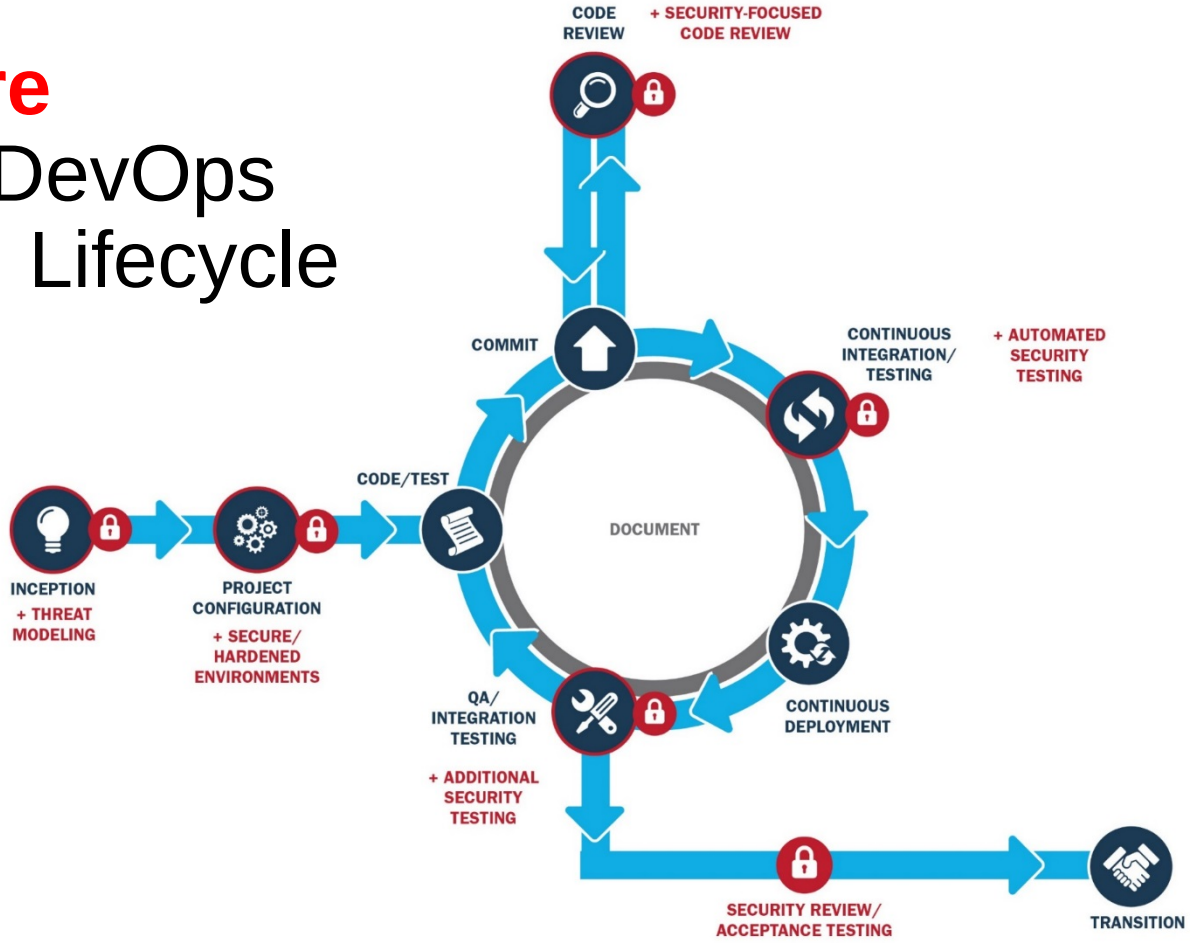
Enhancing SDLC Security

Secure

DevOps Lifecycle

Devs



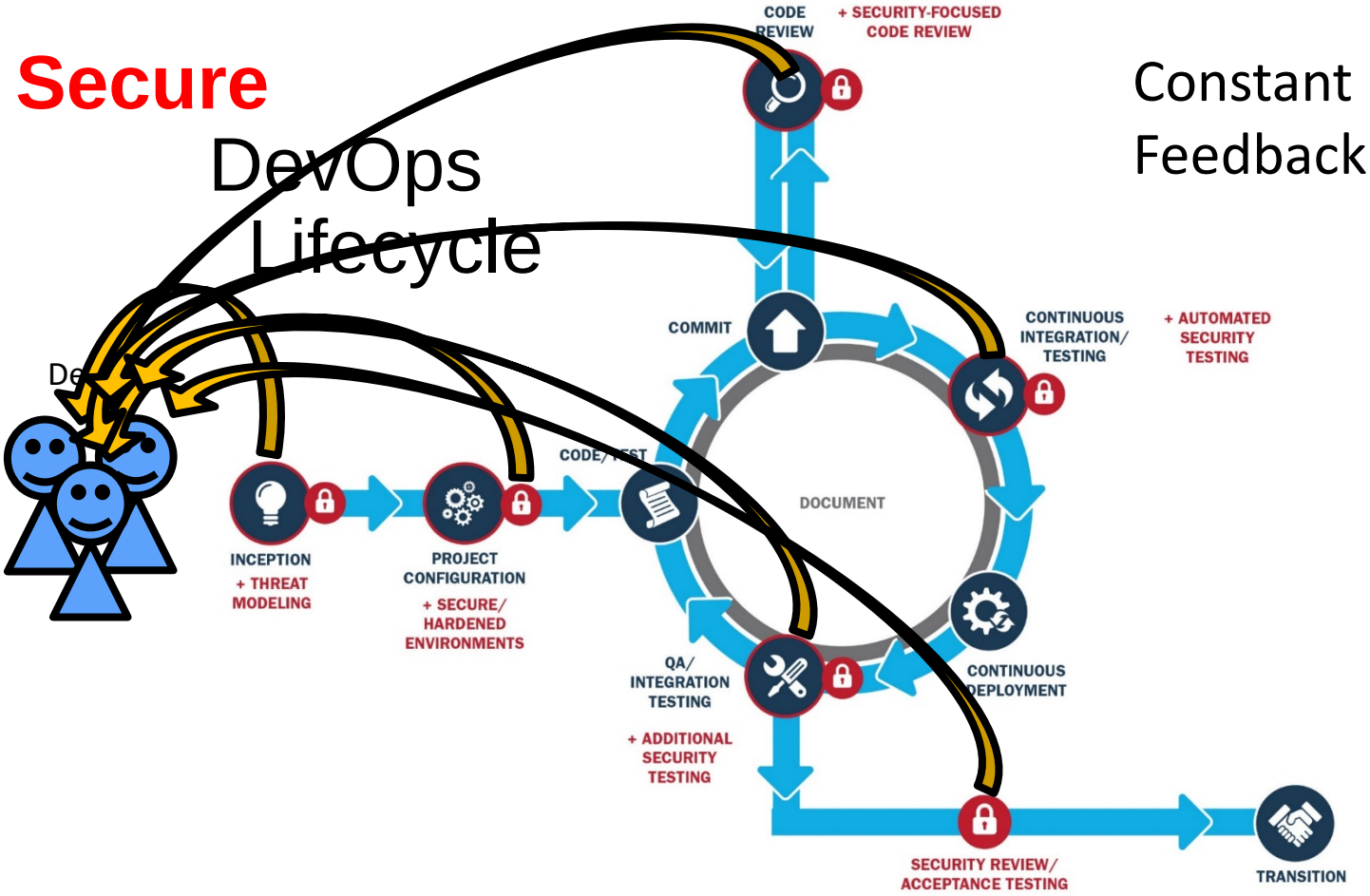


Enhancing SDLC Security

Secure

DevOps Lifecycle

Constant Feedback to Dev

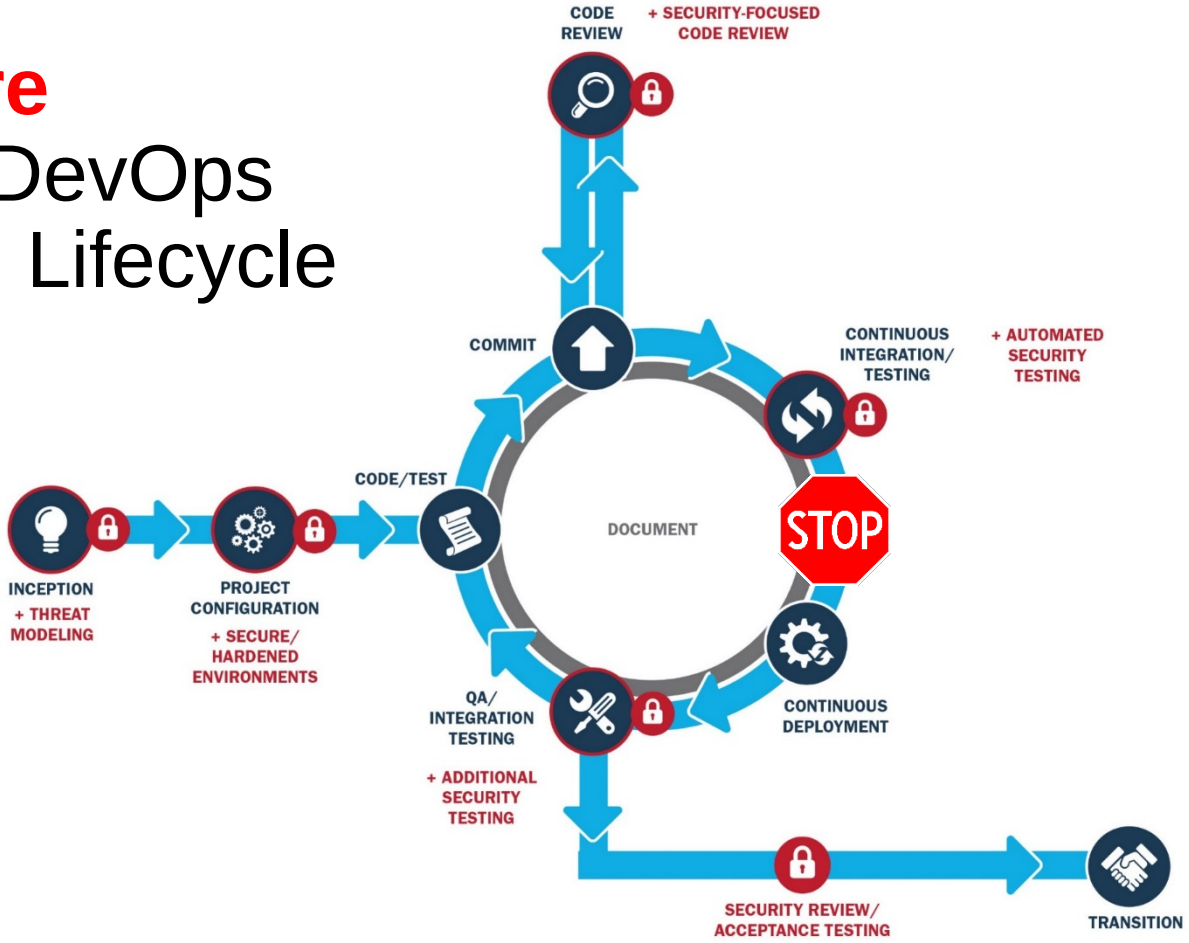


UNCLASSIFIED

Enhancing SDLC Security

Secure

DevOps Lifecycle



Enhancing SDLC Security

Secure

DevOps Lifecycle

- Pausing for manual steps is typical
- Optimize the manual work!
- Persist the output of any tools / work

UNCLASSIFIED

Post-Production Monitoring

- Monitor audit logs produced by CI/CD for anomalies
- Monitor production applications to assure nothing changes outside of the normal change process
- Monitor for new vulnerabilities / threats (a catalog of running components helps!)

Automation (CI/CD) and Security

Not everything can be, needs to be, or should be, automated

- Draw perimeters around things you trust and let that guide where human interaction and verification is needed

Keep track of security assessments

Regimented code management

- Know what source code contributed to a build that's in production so patches are fast and confident

Perform static analysis where possible

Security Overview

Development, operations, teams engineer infrastructure and application

Operations maintains continuous delivery process

Developers write and push code

Continuous integration server internally deploys code

- Docker run / VM provision
- Build
- Test

QA team evaluates the application for correctness

Continuous delivery process deploys code to production servers

Operations maintains production servers

Security Overview with Security Highlights

Development, operations, and security teams engineer infrastructure and application

Operations maintains continuous delivery process

Developers write and push code

Code push triggers security analysis via security controller

Continuous integration server internally deploys code

- Docker run / VM provision
- Build
- Test
- Automated security scan

QA team evaluates the application for correctness

Continuous delivery process deploys code to production servers

Operations maintains production servers

Topics

DevOps Fundamentals

DevOps and Security

Platform Security

AppSec - Secure DevOps

- *Secure DevOps Anti-Patterns*

DevOps Security Anti-Pattern: TheException-1

You automate…

…builds

…functional tests

…deployment

…reporting

…the coffee machine

DevOps Security Anti-Pattern: TheException-2

But security testing is still

manual pen testing,

done only on release.

Recommendations:

- Don’t leave security automation out of your DevOps automation strategy
  - Automated security testing removes human error, infrequent, execution, and excuses
- Don’t try to avoid open source with policies, it is coming whether you like it or not!
- InfoSec must maintain awareness of open source vulnerabilities and continuously check for them

DevOps Security Anti-Pattern: TheException-3

There are great projects out there…

OWASP ZAP

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

GAUNTLT

BE MEAN TO YOUR CODE AND LIKE IT

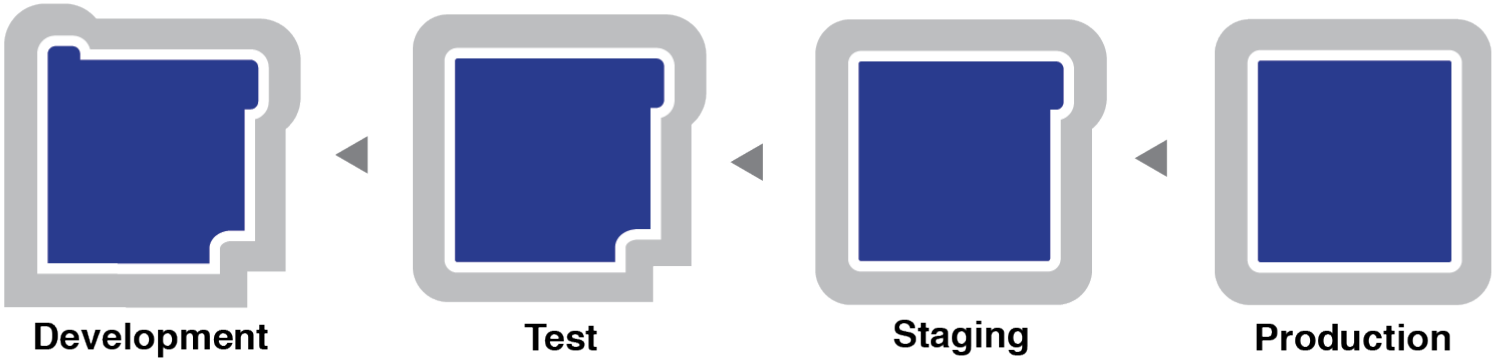
http://gauntlt.org/

DevOps Security Anti-Pattern: Multiverse

When environments are not the same,

your app may never

behave predictably.



Environment parity (between dev, test, prod) is critical for controlling opportunity for security gaps

DevOps Security Anti-Pattern: Multiverse

Recommendations:

- Automate manual steps to the extent possible
- Make development environment parity a priority
- Get Ops involved in creating all environments, including Dev
- Focus on providing fast easy-to-use automation tools for ~~developers~~ **everyone** to keep environments in synch

DevOps Security Anti-Pattern: Configurator

- Uncontrolled configuration changes will lead to an **unmanageable**, **unpredictable**, and **unrepeatable** solution
  - Easy for info security to get out of synch; For example, change in DNS and you have security hole.

Recommendations:

- Avoid the manual quick fix particularly for configuration changes
- Put configuration files under configuration controls

DevOps Security Anti-Pattern: Infiltrator

He sneaks in…

…and alters production …**but he works for you!**

Recommendations:

- Set up roles and revoke administrative access to manually edit production
- Configure prod environment to alert the entire team when manually accessed. Transparency is key.

DevOps Security Anti-Pattern: Survivor

We have all been there…

Intrusions overnight…

…cascading system failures…

…it's all crashing…

…help…me…..

DevOps Security Anti-Pattern: Survivor-2

But you survive…

Glad its over. Going to go sleep for 18 hours…and then back to the normal cycle.

When do we analyze what went wrong?

How do we prevent similar failures in the future?

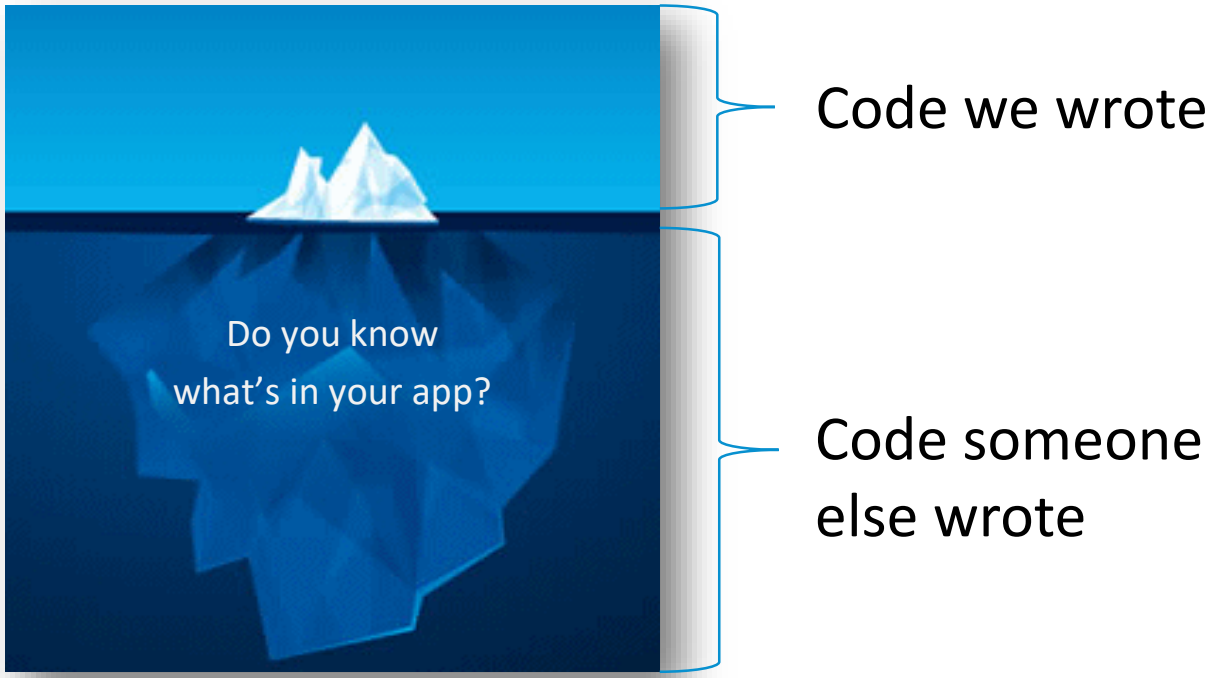
All failures must result in codified change to DevOps process

Recommendations:

- Understand exactly what went wrong
- Never let the same failure happen twice
- Propagate fixes across the enterprise
- Ensure that you teach the next generation

DevOps Security Anti-Pattern: College Party

99% of Global 2000 companies will be using open source code in mission-critical apps by 2016



DevOps Security Anti-Pattern: College Party-2

Recommendations:

- Infosec must enable constant (read: **automated**) checking for open source vulnerabilities

Ways to fail:

- Place infosec outside of the dev workflow
- When UI/UX, infosec and accessibility requirements conflict and never get resolved
- Dictate policy to not use open source
- Document-driven checking is not going catch

Prepare for what is coming….

DevOps Security Anti-Pattern: Skydiver

Once you jump, you can't return to the plane.
You are committed. Permanently*.*

This is not how we should model our deployments

Recommendations:

- Rollback is **essential**; Never be left without an escape route to completely working software
- Strive for approaches that support "one button" rollback (e.g, feature flags or A/B)

SLS team GitHub Projects

- Once Click DevOps deployment
  https://github.com/SLS-ALL/devops-microcosm
- Sample app with DevOps Process
  https://github.com/SLS-ALL/flask_api_sample
  - Tagged checkpoints
    - v0.1.0: base Flask project
    - v0.2.0: Vagrant development configuration
    - v0.3.0: Test environment and Fabric deployment
    - v0.4.0: Upstart services, external configuration files
    - v0.5.0: Production environment
- On YouTube:
  https://www.youtube.com/watch?v=5nQlJ-FWA5A

For more information…

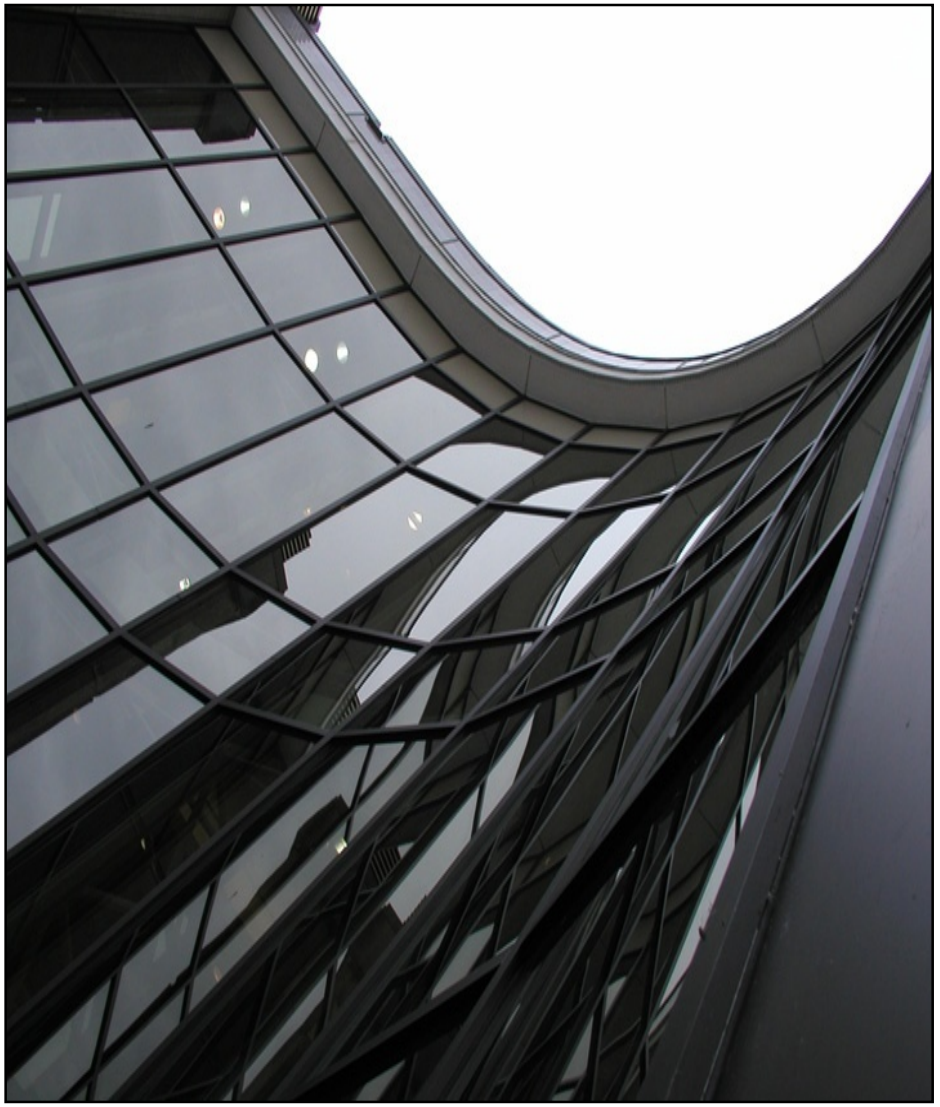
SEI DevOps Blog

https://insights.sei.cmu.edu/devops

Contact Information

Hasan Yasar

Technical Manager,
Secure Lifecycle Solutions
hyasar@sei.cmu.edu
@securelifecycle



Web Resources (CERT/SEI)

http://www.cert.org/

http://www.sei.cmu.edu/