Security Engineering and **DevSecOps** *from an Industry Perspective*

Presented by: Shanelle Ileto

November 18, 2021

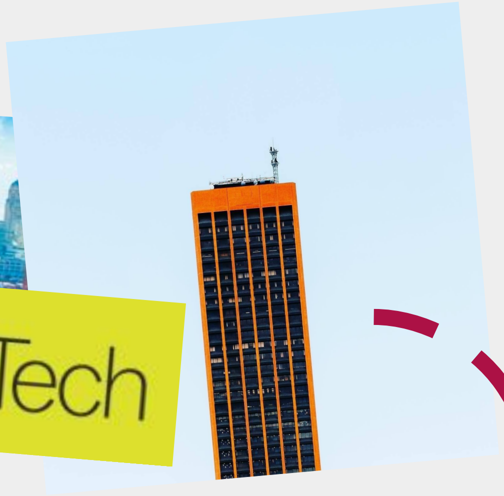
A bit about myself...

UB CSE

Department of Computer Science and Engineering

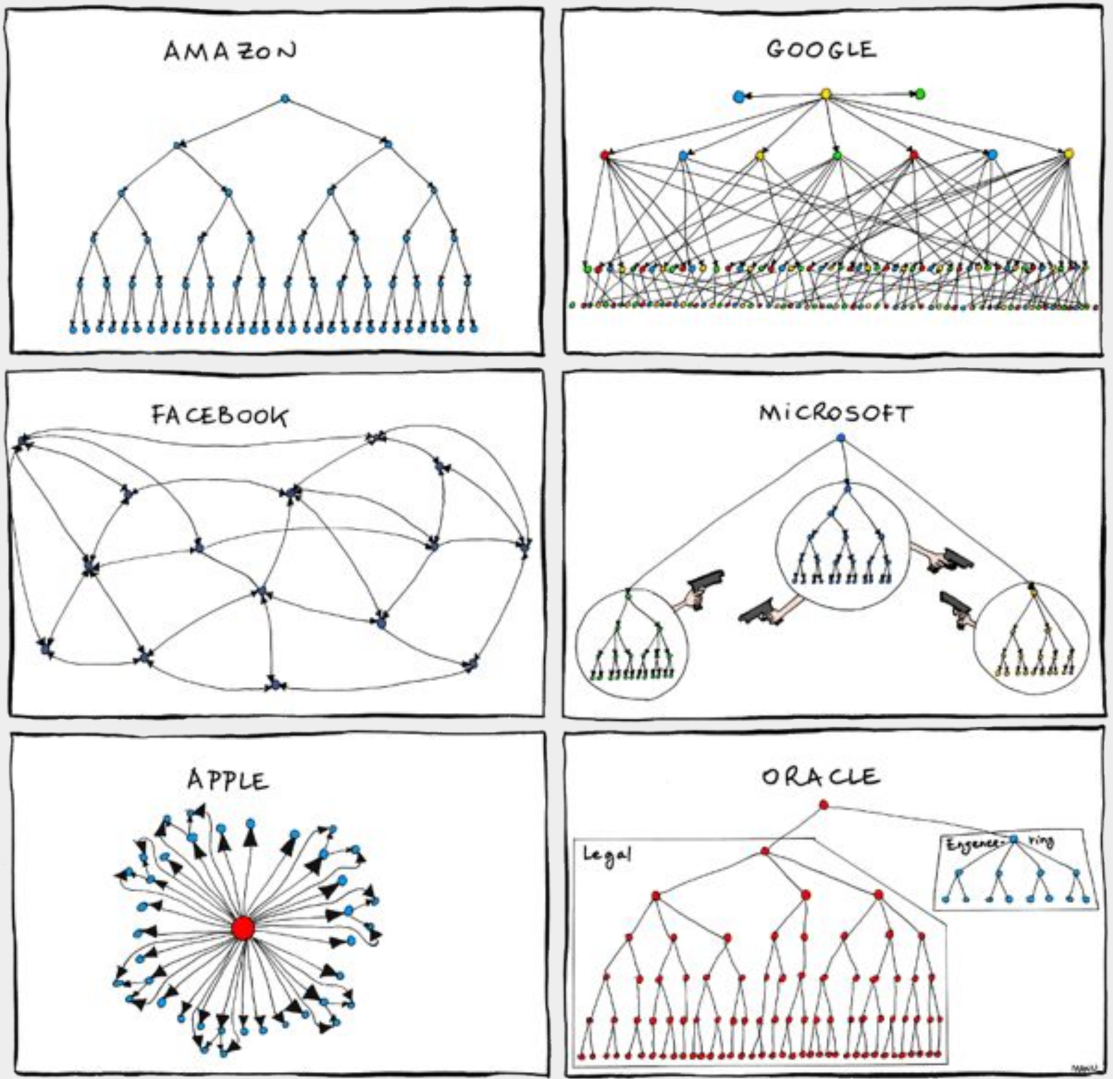
University at Buffalo The State University of New York

NetDef



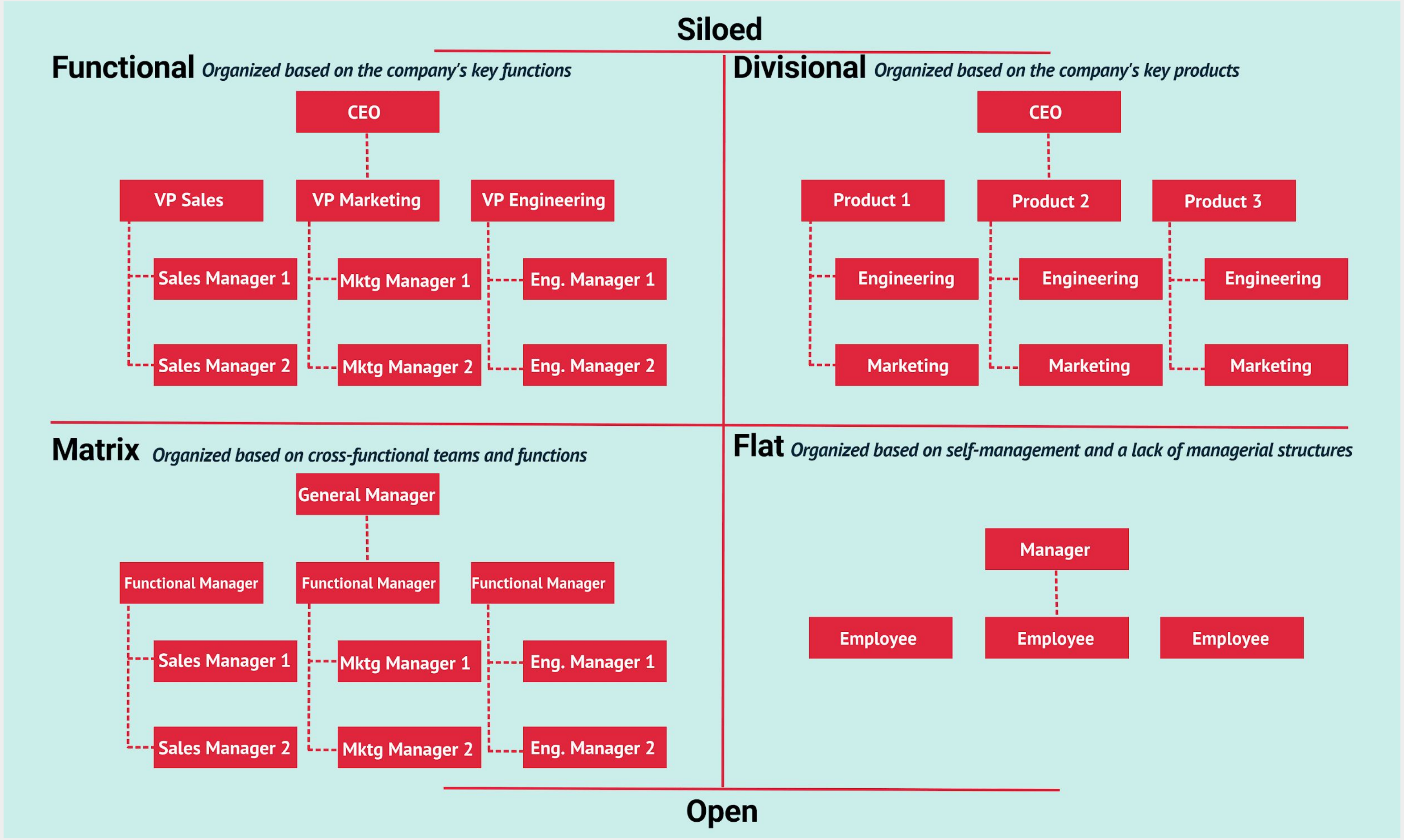
M&T Tech

unqork

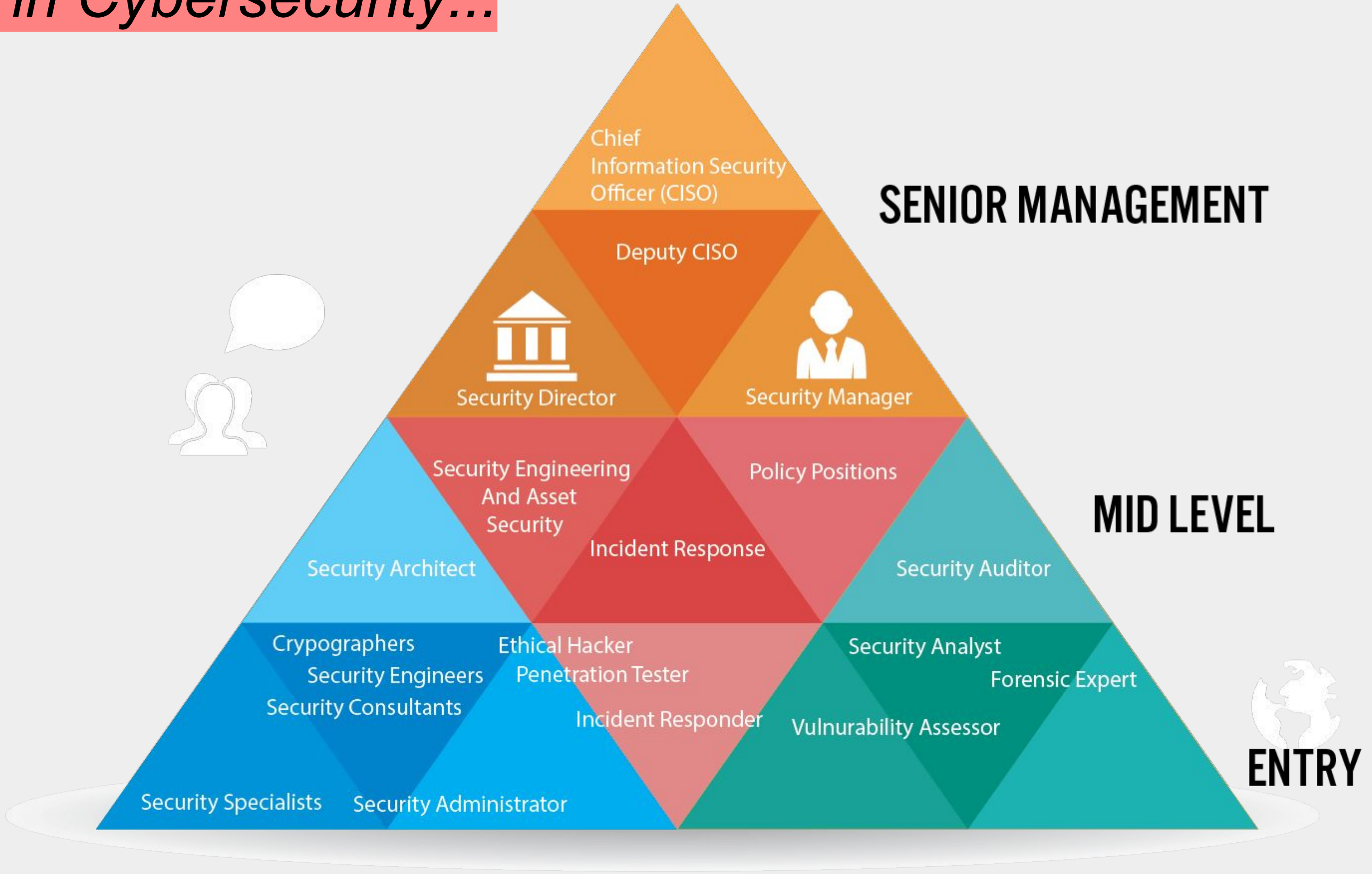


Every company has an **organizational structure...**

Four Key Types of Organizational Structure...

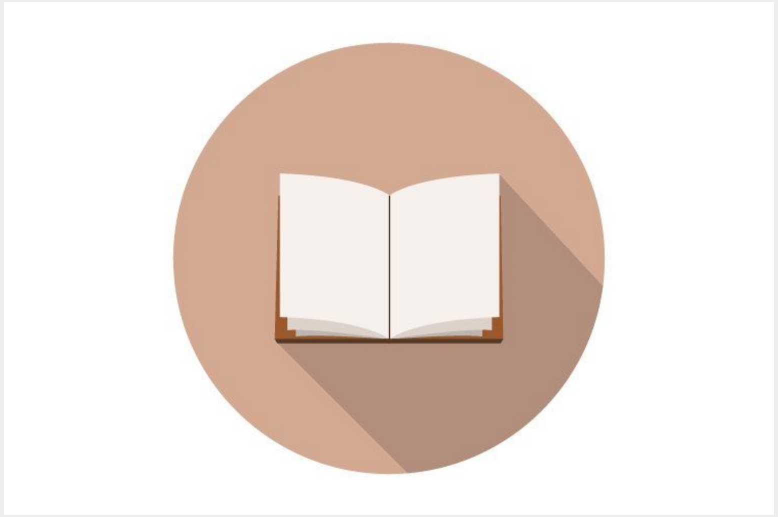


Roles in Cybersecurity...



Activity: Case Study

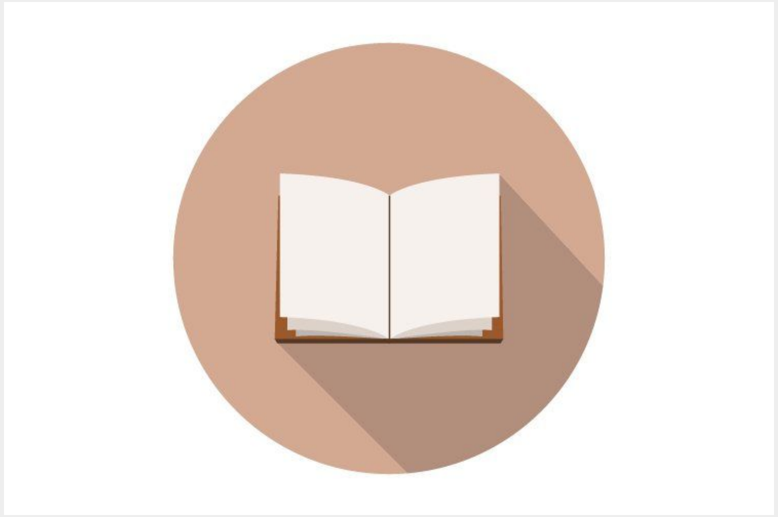
- Your company, Fictionary, is a famous localized bookstore that specializes in acquiring “special” books, ranging from rare pre-first edition versions to unpublished novels.
- Fictionary has just acquired digital rights to these books, and wants to release an online shopping platform to widen their network and boost their sales.



Activity: Case Study

- The tech team has no idea where to start for this online shopping platform and turns to you to help divide tasks and responsibilities…
  - Determine what departments/roles are needed for the project
  - Determine what each department/roles function will be

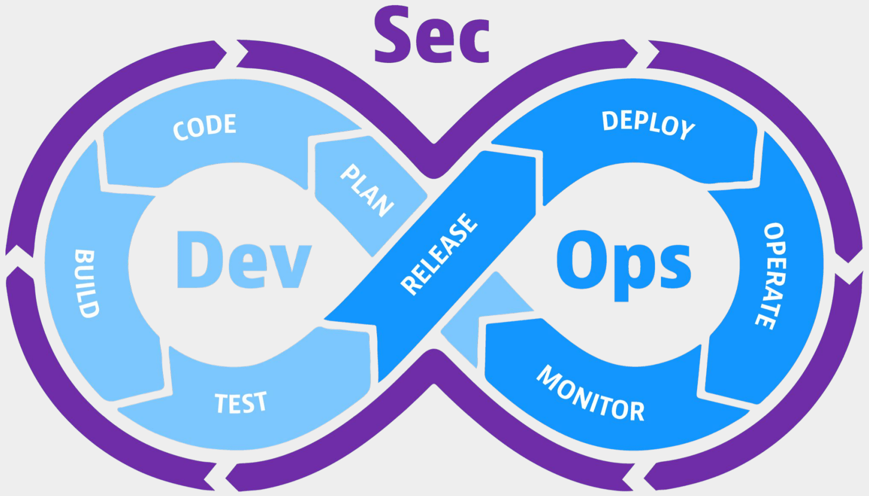
Role/Department	**Responsibilities**
...	...



Security - The *Traditional* Way

REQUIREMENTS
DESIGN
IMPLEMENTATION
TESTING
MAINTENANCE
WATERFALL
DEVELOPMENT MODEL

Introducing **DevSecOps**

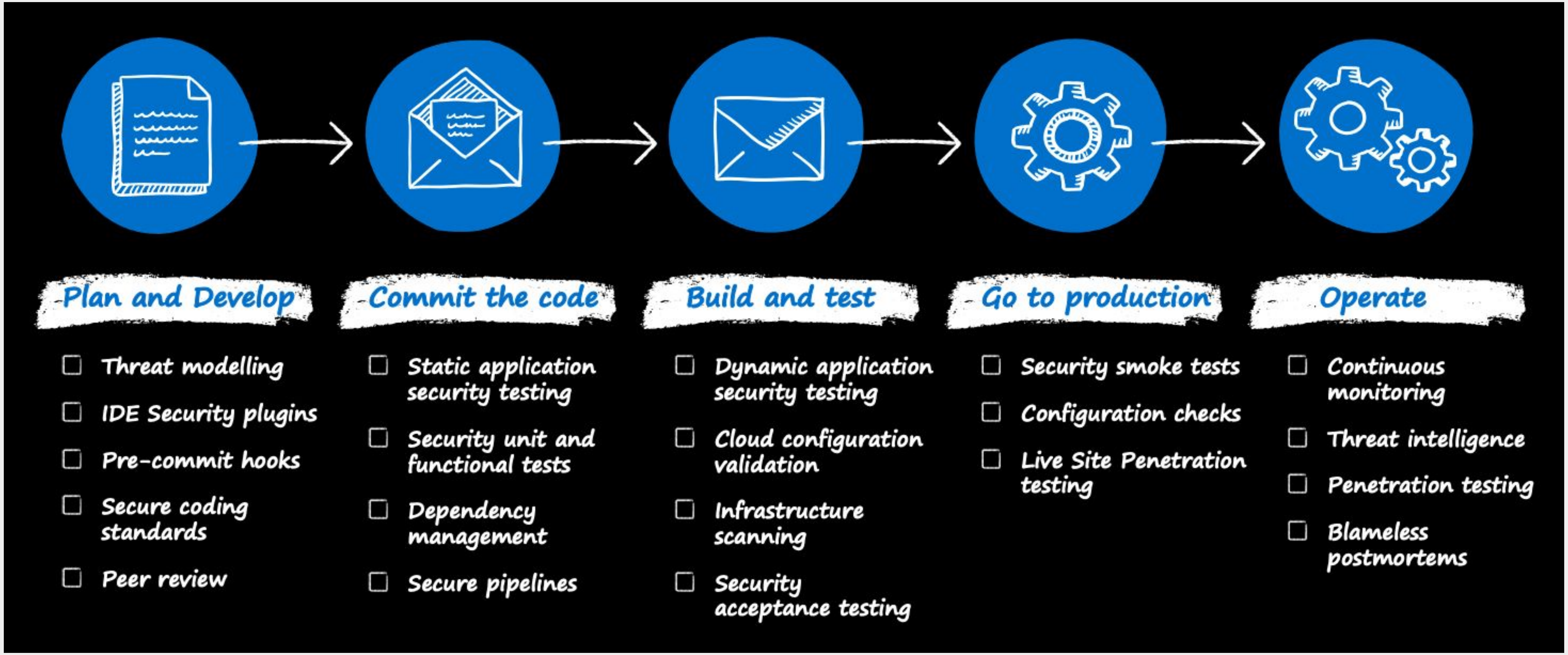


- DevSecOps - *Development Security and Operations*
- An approach to culture, automation and platform design that integrates security as a shared responsibility throughout the entire IT lifecycle
- Brings together people, process, and technology

Benefits *and* Challenges

- *Benefits* include…
  - Speed and rapid delivery
  - Quality and reliability
  - Improved collaboration
  - Security (Shift left)
- *Challenges* include…
  - General resistance to change and culture
  - Organizational/IT department and job changes
  - Integration, usage, and cost of tools and platforms
  - Resource and talent

DevSecOps - A Closer Look



Plan and Develop

- ☐ Threat modelling
- ☐ IDE Security plugins
- ☐ Pre-commit hooks
- ☐ Secure coding standards
- ☐ Peer review

Commit the code

- ☐ Static application security testing
- ☐ Security unit and functional tests
- ☐ Dependency management
- ☐ Secure pipelines

Build and test

- ☐ Dynamic application security testing
- ☐ Cloud configuration validation
- ☐ Infrastructure scanning
- ☐ Security acceptance testing

Go to production

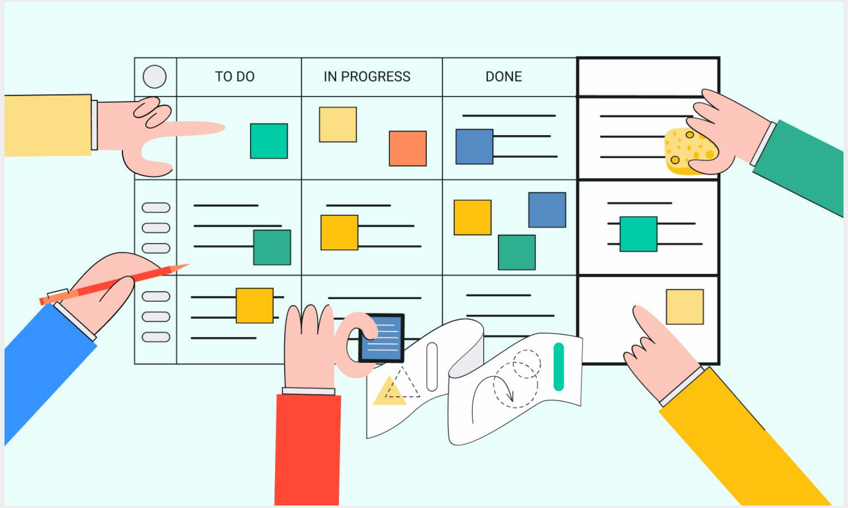
- ☐ Security smoke tests
- ☐ Configuration checks
- ☐ Live Site Penetration testing

Operate

- ☐ Continuous monitoring
- ☐ Threat intelligence
- ☐ Penetration testing
- ☐ Blameless postmortems

Step 1: *Plan*

- Define business value and gather requirements
  - Determine **methodology** (ie Agile vs Waterfall) and **processes** (ie Scrum vs Kanban) if not in place
  - Create **Agile board** which includes user stories in a **backlog**
  - Decide Agile **ceremonies** (ie Planning, Backlog Refinement, Review, Retrospective)
  - Documentation, architecture diagrams, or proof of concepts
- Potential *Tools*
  - Jira, Trello, Git, Confluence, MS Docs, LucidChart

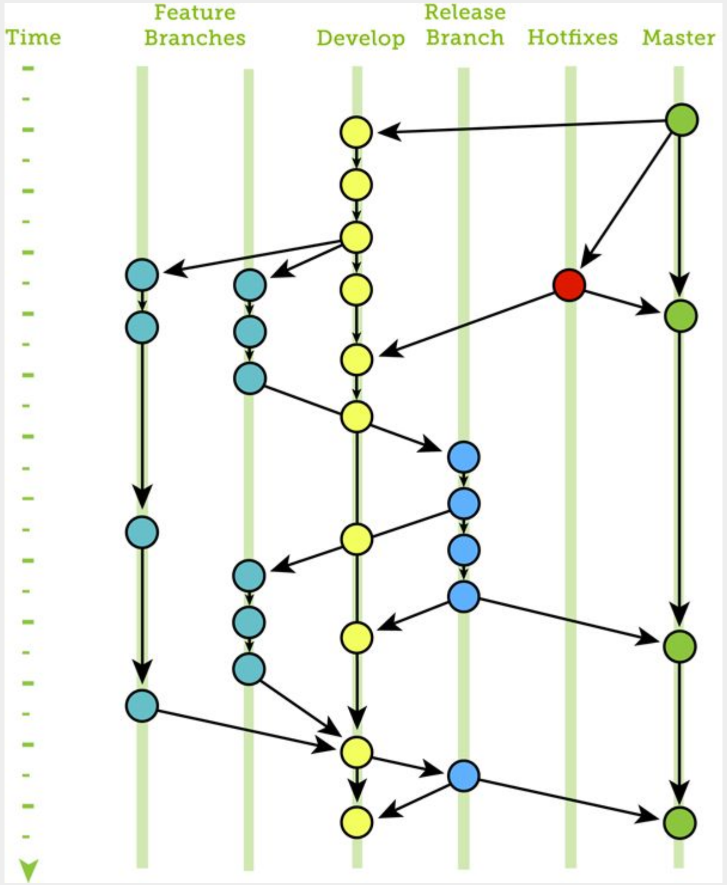


Step 1: *Plan*

- Security focuses at this stage...
  - **Threat modelling** - to view the application through the lens of a potential attacker
    - Methodologies include STRIDE, DREAD, OWASP
    - Four Question Framework:
      - What are we working on?
      - What can go wrong?
      - What are we going to do about it?
      - Did we do a good job?
    - *Benefits* include…
      - Detects problems early in the software development life cycle (SDLC)
      - Prioritize threats, mitigation efforts, and budgeting and lowers risk
      - Identifies and eliminates single points of failure and design flaws
      - Improve organization’s security posture
- Potential *Tools*
  - IriusRisk, MS Threat Modeling Tool, Cairis

Step 2: *Code*

- Software engineers design, develop, and review code
  - Design and develop code
  - Create **code repository** and determine **branching structure**
  - Push code to repo and review with others
- Potential *Tools*
  - GitHub, GitLab, Bitbucket, Stash, Azure DevOps



Step 2: *Code*

- Security focuses at this stage...
    - **IDE security plug-ins** - for lightweight static analysis checking within an integrated development environment before pushing code to repo
    - **Pre-commit hooks** - provides Git hook scripts that enable identification issues before submitting code for code review
    - **Peer reviews** - using pull request process to detect uncovered defects, bugs, or issues
    - **Secure coding standards** - some coding standards include…
        - Input validation and output encoding
        - Authentication and password management
        - Session management and access control
        - Cryptographic practices and data protection
        - Error handling and logging
        - Communication security (HTTPS, TLS)
        - System Configuration
        - Database Security
        - File Management
        - Memory Management

Step 2: *Code*

- Security focuses at this stage...
  - **Software Component Analysis (SCA) or Dependency management** - ensure that third-party libraries and dependencies are secure and their up-to-date versions are installed from reliable sources
    - Black Duck SCA
    - WhiteSource
    - JFrog Xray
    - Snyk
  - **Static application security testing (SAST)** - assessing code quality and style and debugging source code before it is run; some tools include…
    - Veracode
    - SonarQube
    - Fortify
    - Checkmarx
- (Other) Potential *Tools*
  - Gerrit, Phabricator, SpotBugs, PMD, CheckStyle, Find Security Bugs

Step 3: *Build*

- Compile and package the source code into one desired format
  - Manages **software builds** and versions for fast deployment
  - Use of **automated tools** to help compile and package code for future releases to production
- Potential *Tools*
  - Docker, Ansible, Puppet, Chef, Gradle, Maven, JFrog Artifactory

Step 3: *Build*

- Security focuses at this stage...
    - **Cloud/containerization configuration validation and infrastructure scanning** - create policies within service providers, scan containers, or use security tools for infrastructure utilizing infrastructure as code (IaC)
    - **Secure pipelines** - security controls for pipelines
        - **Pipeline** - set of automated processes and tools that allow both developers and operations professionals to build, test, and deploy software into a production environment; allows for both continuous integration and continuous delivery/deployment
    - **Automating (security) tests** - tools can be plugged into existing CI/CD pipeline to automate testing
- Potential *Tools*
    - OWASP Dependency-Check, SonarQube, SourceClear, Retire.js, Checkmarx, Synk

Step 4: *Test*

- Implement testing tools in workflow to ensure best development quality
  - Various types of testing (ie integration, UI, security) that can be **manual** or **automated**
- Potential *Tools*
  - JUnit, Codeception, Selenium, Vagrant

Step 4: *Test*

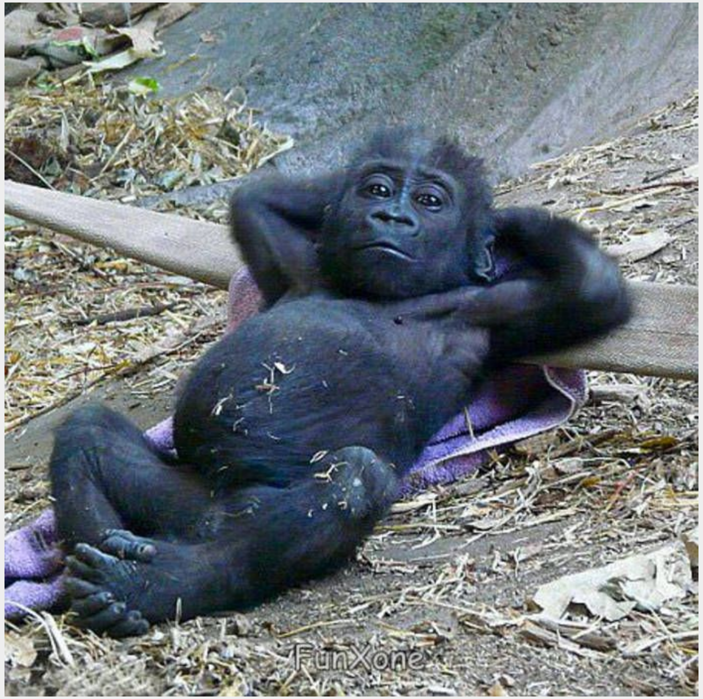
- Security focuses at this stage...
  - **Dynamic application security testing (DAST)** - web application security test that actively investigates running application with penetration tests to detect possible vulnerabilities; some tools include
    - Netsparker
    - Acunetix
    - AppScan
    - Rapid 7
  - **Other data validation and security tests** - ie SQL injection, application security testing, authentication, access control, etc
- (Other) Potential *Tools*
  - BDD Automated Security Tests, JBroFuzz, Boofuzz, OWASP ZAP, IBM AppScan, SecApp suite

Step 5: *Release*

- Code has passed all testing and continuous integration is achieved; application is *ready to be deployed*
- Potential *Tools*
    - Ansible, Puppet, HashiCorp, Terraform, Chef, Docker, Kubernetes

Step 5: *Release*

- Security focuses at this stage…



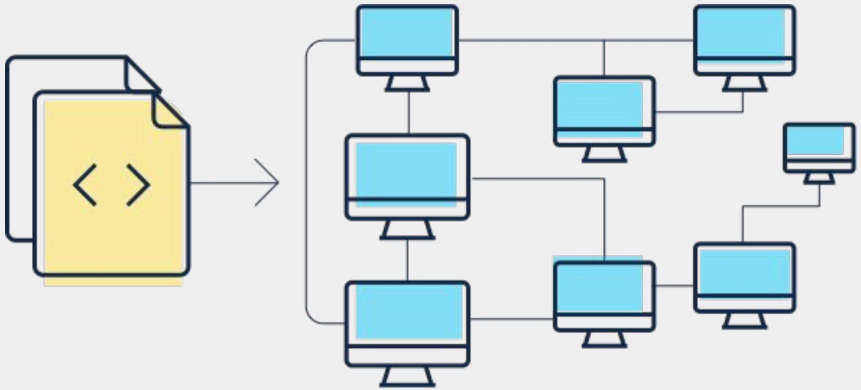
...well *kinda**

Step 5: *Release*

- Security focuses at this stage…
    - **Audit** - working with cyber security to review compliance with standards
    - **Fixing zero-day vulnerabilities** - unknown security vulnerabilities or software flaw

Step 6: *Deploy*

- Operation team is deploying application or new features into production and continuous delivery is achieved
  - Goes through all **environments** applicable (DEV, TEST/CERT, PROD, DR)
  - Usually **automated**, but can potentially have some manual steps for testing depending on application
- Potential *Tools*
  - Puppet, Chef, Ansible, Jenkins, Kubernetes, OpenShift, OpenShift, Docker

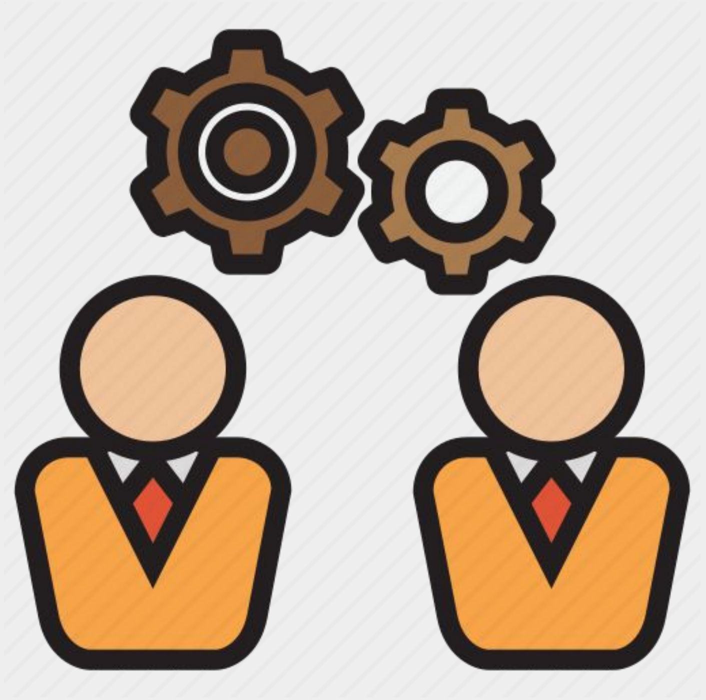


Step 6: *Deploy*

- Security focuses at this stage...
  - **Continue configuration and infrastructure scanning**- scale across multiple environments and automate dynamically using infrastructure as code
  - **Runtime verification** - extract information from a running system in order to determine whether it performs as expected
  - **Automate security updates** - configure pipeline to eliminate the need for admins to manually patch for known vulnerabilities
- Potential *Tools*
  - Osquery, Falco, Tripwire

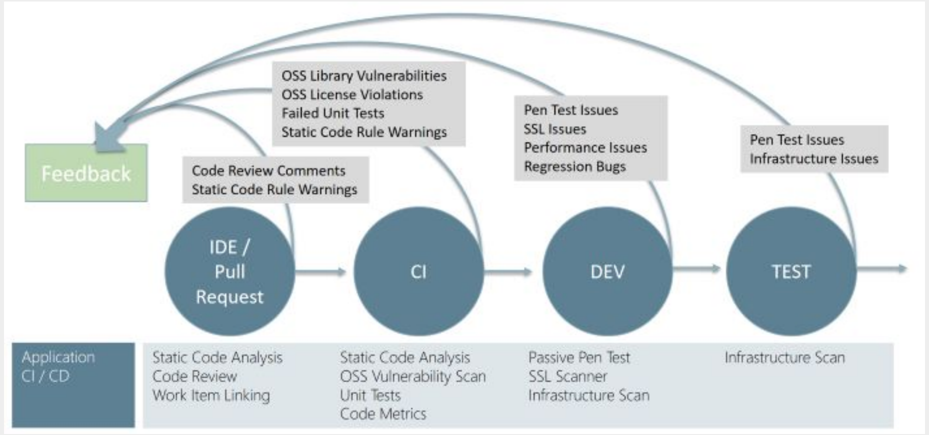
Step 7 and 8: *Operate and Monitor*

- Continues to maintain a scalable infrastructure and check security issues and performance
  - Collect various information about issues from a specific software release in production
  - Log management and intrusion detection or prevention system (IDS/IPS)
- Potential *Tools*
  - New Relic, Datadog, Grafana, Wireshark, Splunk



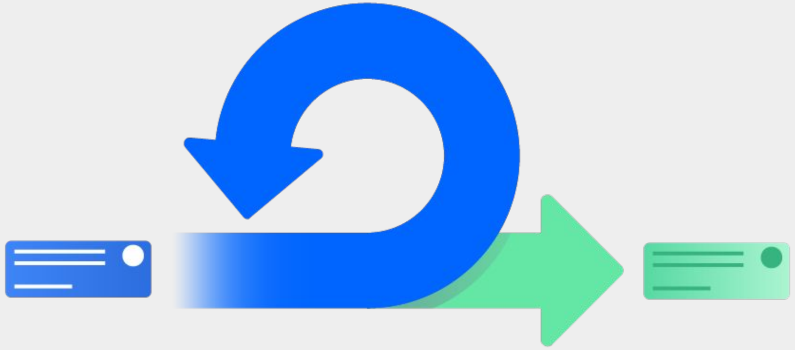
Step 7 and 8: *Operate and Monitor*

- Security focuses at this stage...
  - **Penetration testing or other security testing*** - tests on endpoints to uncover vulnerabilities, fuzz testing, port scanning, network security analysis, etc
  - **Actionable intelligence** - integrate alerts and telemetry into IT service management platform (ITSM)
  - **Feedback loops** - create process to find and flag risks and vulnerabilities that require investigation and potential resolution at any stage



Step 9: *Repeat!*

- Respond and record attacks and threats, if applicable
- Bug fixes/hot fixes and new features for future sprints
- Vertical or horizontal infrastructure scaling to accommodate technical or business requirements
- **<u>AUTOMATION!</u>**



Questions?

Thank you!