DevSecOps

ALI AHMED THAWERANI

About the course

- Prerequisites:
  - Software Engineering Fundamentals
  - Basic knowledge of Linux/Unix/Windows systems
  - Programming and Scripting experience

Course Description

- This course introduces the principles, practices, and tools involved in DevSecOps.
- It covers the integration of development and operations teams, automation of the software development lifecycle (SDLC), and continuous integration/continuous delivery (CI/CD) pipelines.
- Students will gain experience with tools and techniques used in modern DevSecOps practices, including source control, configuration management, containerization, monitoring, and cloud infrastructure.

Course Objectives

- By the end of this course, students should be able to:
  - Understand the core principles and culture of DevOps.
  - Set up and use Continuous Integration and Continuous Delivery (CI/CD) pipelines.
  - Automate deployment processes using containerization tools like Docker.
  - Use infrastructure as code (IaC) tools to manage and automate infrastructure.
  - Monitor applications and infrastructure with appropriate tools.
  - Integrate security into the DevOps pipeline (DevSecOps).

Introduction to DevOps and DevSecOps

LECTURE 1

Overview

- What is DevOps?
  - DevOps is a set of practices that combine software development (Dev) and IT operations (Ops) to shorten the development lifecycle and deliver high-quality software continuously.
- DevOps Principles (CAMS)
  - Culture: Focus on collaboration and shared responsibility across teams.
  - Automation: Use tools to automate repetitive tasks, including builds, testing, and deployments.
  - Measurement: Collect and analyze metrics to improve processes.
  - Sharing: Foster a culture of transparency and knowledge sharing among stakeholders.

Overview

- Benefits of DevOps
  - Speed: Faster delivery of software updates and features.
  - Reliability: Improved stability and performance of systems.
  - Scalability: Seamless scaling of applications and infrastructure.
  - Visibility: Enhanced monitoring and logging capabilities for better insights.
- Introduction to DevSecOps
  - DevSecOps is the practice of integrating security into every phase of the DevOps lifecycle, from planning and development to testing, deployment, and maintenance.

Overview

- Why DevSecOps?
  - Reduced Attack Surface: Identifies and fixes vulnerabilities early in development.
  - Automated Security Testing: Embeds security tools and tests into CI/CD pipelines.
  - Compliance Integration: Ensures regulatory compliance through automated checks.
- Examples
  - Facebook: Uses automated testing in CI/CD to identify vulnerabilities early.
  - Amazon: Adopts "security as code" to ensure secure cloud infrastructure.
  - Google: Implements strict infrastructure security policies, including container security.

The Phoenix Project

Bill's Challenge at Parts Unlimited

- The Crisis:
  - Phoenix Project, critical to the company's future, is massively over budget and delayed.
  - CEO tasks Bill with fixing it within 90 days or faces outsourcing of his department.
- The Turning Point:
  - A prospective board member introduces the "Three Ways" philosophy.
  - Bill sees parallels between IT operations and manufacturing workflows.
- The Solution:
  - Streamline workflows and improve interdepartmental communication.
  - Align IT operations to effectively serve business goals.

Strategy

- The entire future of the company depend upon.
- It requires that IT be a core competency.

Challenges in Reporting, Compliance, and Operations

- Reporting:
  - Inaccurate financial reporting caused by recurring IT general control issues.
  - Failures in accounts payable and inventory systems delayed financial closures.
  - Payroll system failures led to reporting errors and inaccuracies.
- Compliance:
  - Audit failures resulted in adverse external auditor footnotes.
  - Non-compliance risks severe consequences for regulatory assessments.

Challenges in Reporting, Compliance, and Operations

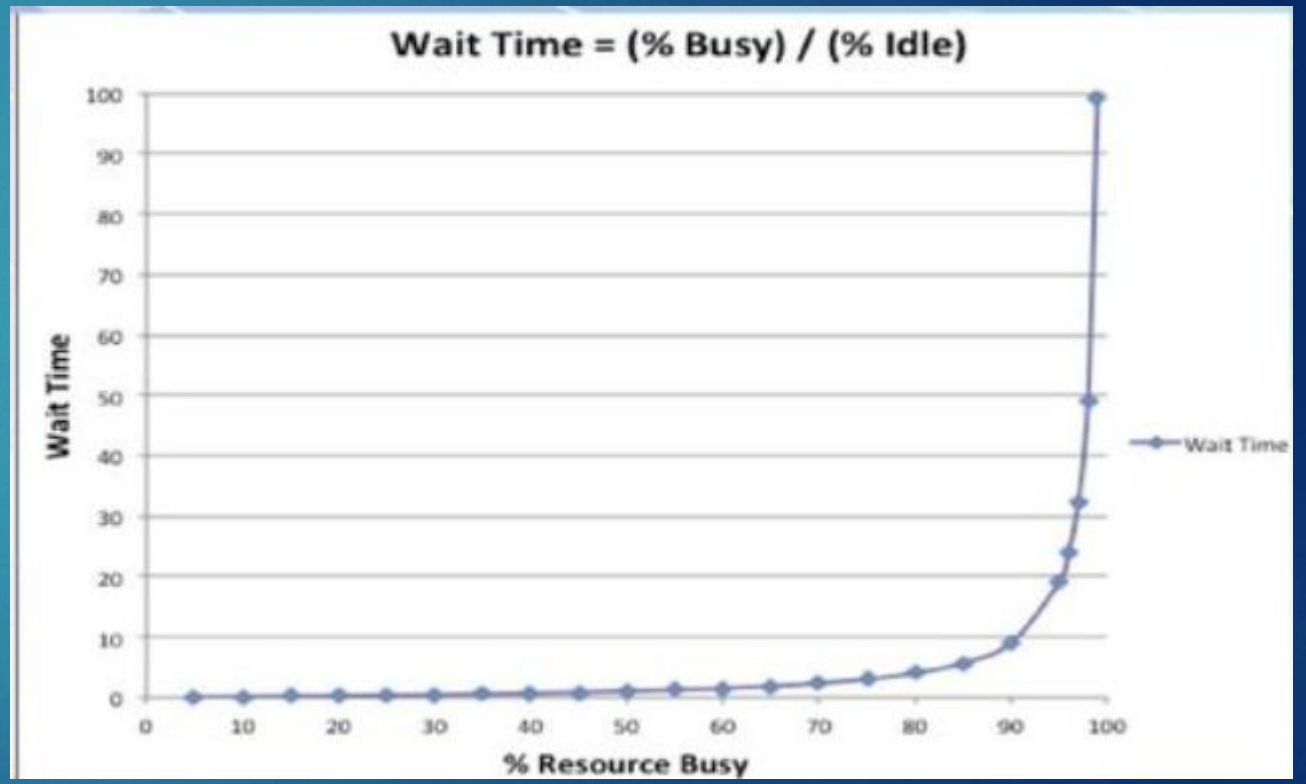
- Operations:
  - Phoenix Project: $20M over budget, three years late, and detrimental upon deployment.
  - Frequent IT service failures disrupted critical business operations.

Generic problems – downward spirals

- Fragile artefacts become more fragile
- Technical debt grows
- Date-driven application project focus only on features, sacrificing non-functional requirements, which results in more fragile artifacts in production.
- Application deployment take longer, more difficult, and gets worse
- IT Ops is stuck fire fighting and therefore cannot do preventive work or new projects
- Long feature delivery cycle times result in more political decision making, meaning more focus on features (vs non-functional requirements)

The wait time / resource busy graph

- Bill realizes that Brent, one of the senior engineers is overcommitted and a number of people and projects are depending on him to be available.
- As the resource utilization goes past 80%, wait time goes through the roof
- Assume that all resources are 90% busy, the graph shows a wait time of 9 hours.
- If there are 7 handoffs (work centers or resources), the total wait time is 63 hours



Optimizing IT Operations and Workflows

- Addressing Overload:
  - Identified excessive WIP (Work in Progress) and over-reliance on Brent for critical tasks.
- Preventive Measures:
  - Elevated preventive work to reduce unplanned tasks, especially for key personnel.
  - Made all work visible using Kanban boards and standardized workflows.
- Workload Management:
  - Introduced project freeze to throttle work intake into IT operations.
  - Limited work dependencies on Brent by documenting processes and managing handoffs.

Optimizing IT Operations and Workflows

- Operational Improvements:
  - Built DevOps workflows to embed non-functional requirements early in development.
  - Reduced batch sizes for faster, single-piece flow through Development and IT Ops.
- Strategic Adjustments:
  - Scoped audit and infosec correctly to align with corporate goals.
  - Adapted to shifting constraints by bringing critical resources back in-house.

4 types of work

- Business project work
  - Development projects
  - Typically run by PMO
- IT project work
  - Infrastructure whether external or internal
  - Reside in the DBA manager, storage manager etc.
  - Creates a problem because there is no way to identify committed capacity
- Changes
  - Generated from the previous two types of work
  - Tracked in a ticketing system
- Unplanned work
  - Operational incidents or problems caused by the previous types of work
  - The IT capacity death spiral – not paying down technical debt

The Three ways

- Way No. 1: The flow
  - Understand the flow of work
    - Without understanding, any changes will have random effects
  - Always increase the flow
  - Never pass on defects downstream
  - Never allow local optimization to cause global degradation
  - Achieve profound understanding of the entire system

The Three ways

- Way No. 2: Consistent Feedback
  - Understand and respond to the needs of all customers, both internal and external
  - Shorten and amplify all feedback loops
    - This rule is modelled after the Toyota Lean manufacturing line, where any employee can stop the line immediately if they see any defect.
  - Create quality at the source
    - This means pro-actively building quality into everything
    - There is no need for developers to rely on QA to find their errors.
  - Create and embed knowledge where we need it

The Three ways

- Way No. 3: Continual learning
  - Create a culture that encourages experimentation
  - Learns from failure
  - Recognizes repetition as the prerequisite to mastery

Top DevOps Myths

- DevOps replaces Agile
  - Compatible
  - Logical continuation of Agile
  - Agile is not a prerequisite for adopting DevOps
- DevOps replaces ITIL or ITSM
  - IT infrastructure Library or IT service Management
  - In order to accommodate the faster lead times and higher deployment frequencies many areas of ITIL require automation (CM configuration and release processes)
- DevOps is only for open source software
  - Principles are universal
  - Independent of the underlying technology

Top DevOps Myths

- DevOps means NoOps
  - DevOps puts more responsibility on Development
  - Requires many operations tasks to become self-service
  - Automate rather than be ticket based (e.g. get a production like environment)
- DevOps is just "infrastructure as code" or automation
  - More than just automation
  - Requires shared goals and pain
- DevOps is only for startups and unicorns
  - For any organization that must increase flow of planned work
  - While maintaining quality, reliability and security for the customer

DevOps Introduction

Vision for Collaborative and High-Performing IT

- Unified Collaboration:
  - Product Owners, Development, QA, IT Operations, and Infosec align to ensure organizational success.
- Common Goals:
  - Enable rapid, smooth workflows and seamless deployments.
  - Deliver world-class stability, security, and availability.
- Enhanced Productivity:
  - Small teams independently develop, test, and deploy code safely.
  - Create systems that reduce friction and maximize developer efficiency.

Vision for Collaborative and High-Performing IT

- Customer and Business Value:
  - Deliver secure, reliable value to customers at scale.
  - Test features quickly to meet user needs and advance business goals.
- Operational Excellence:
  - Maintain continuous flow through the value stream with minimal disruption.
  - Reduce chaos in IT Operations and foster cross-team expertise.
- Cultural Outcomes:
  - Promote organizational learning and high employee satisfaction.
  - Drive market success through innovation and collaboration.

Outcomes of DevOps

- Improved productivity and satisfaction.
- Better organizational performance.
- Enhanced customer satisfaction.

Current Challenges

- Broken systems and poor outcomes.
- Adversarial relationship between Development and IT Operations.
- Late testing and Infosec activities.
- Manual efforts and handoffs causing delays.
- Long lead times and poor quality.
- Chaotic production deployments.
- Negative impacts on customers and business.
- Dissatisfied employees and budget cuts.

The Solution

- Need to change how we work.
- DevOps as the best way forward.

Lean Manufacturing Revolution

- Adoption of Lean principles in the 1980s.
- Improved productivity, lead times, and quality.
- Increased customer satisfaction.
- Reduced lead times from six weeks to less than three weeks.
- Increased on-time shipments from 70% to over 95%.
- Organizations that did not adopt Lean practices lost market share.

Increased Expectations

- What was good enough in previous decades is not good enough now.
- Cost and time to develop and deploy strategic business capabilities have dropped significantly.
- 1970s-1980s: New features required 1-5 years to develop and deploy, costing tens of millions of dollars.
- 2000s: Advances in technology and Agile principles reduced development time to weeks or months, but deployment still took weeks or months.

Table 0.1: The Ever-Accelerating Trend toward Faster, Cheaper, Lower Risk Delivery of Software

	1970s–1980s	1990s	2000s–Present
Era	Mainframes	Client/Server	Commoditization and Cloud
Representative technology of era	COBOL, DB2 on MVS, etc.	C++, Oracle, Solaris, etc.	Java, MySQL, Red Hat, Ruby on Rails, PHP, etc.
Cycle time	1–5 years	3–12 months	2–12 weeks
Cost	$1M–$100M	$100k–$10M	$10k–$1M
At risk	The whole company	A product line or division	A product feature
Cost of failure	Bankruptcy, sell the company, massive layoffs	Revenue miss, CIO's job	Negligible

Technology Value Stream

- Organizations deploy changes hundreds or thousands of times per day.
- Fast time to market and relentless experimentation are key.
- Organizations unable to replicate DevOps outcomes risk losing to nimble competitors.
- Potential to go out of business, similar to manufacturing organizations that did not adopt Lean principles.
- Customer acquisition and value delivery depend on the technology value stream.
- Jeffrey Immelt: “Every industry and company that is not bringing software to the core of their business will be disrupted.”
- Jeffrey Snover: “In previous economic eras, businesses created value by moving atoms. Now they create value by moving bits.”

The Enormity of the Problem

- Affects every organization, regardless of industry or size.
- How technology work is managed predicts organizational success or survival.

- Adoption of new principles and practices is crucial.
- Traditional methods may no longer be effective.

- Inherent conflict between Development and IT Operations.
- Results in slower time to market, reduced quality, increased outages, and growing technical debt.

Understanding Technical Debt

- Term coined by Ward Cunningham.
- Analogous to financial debt: decisions lead to problems that become harder to fix over time.
- Incurs interest, reducing future options.

- IT organizations must pursue two goals simultaneously:
  - Respond to the rapidly changing competitive landscape.
  - Provide stable, reliable, and secure service to customers.

The Core Conflict

- Development's Role
  - Responsible for responding to market changes.
  - Deploy features and changes into production quickly.
- IT Operations' Role
  - Ensure stable, reliable, and secure IT services.
  - Prevent production changes that could jeopardize stability.
- Opposed Goals
  - Development and IT Operations have diametrically opposed goals and incentives.
  - Creates friction and challenges within the organization.

The Core Conflict

- Dr. Eliyahu M. Goldratt's concept of "the core, chronic conflict."
- Organizational measurements and incentives across silos prevent achieving global goals.
- Conflict creates a powerful downward spiral.
- Leads to poor software and service quality, bad customer outcomes, and daily workarounds.

The Core Conflict

- Act 1: IT Operations
  - Goal: Keep applications and infrastructure running.
  - Problems due to complex, poorly documented, and fragile systems.
  - Technical debt and daily workarounds.
- Act 2: Broken Promises
  - Compensating for broken promises.
  - Product managers and executives set new targets.
  - Development tasked with urgent projects, adding to technical debt.
- Act 3: Increasing Difficulty
  - Work becomes more difficult and time-consuming.
  - Increased communication, coordination, and approvals.
  - Quality deteriorates, and work queues lengthen.

The Downward Spiral

- Observed in countless organizations over a decade.
- Requires DevOps principles to mitigate.

- Every IT organization has two opposing goals.
- Every company is a technology company, whether they know it or not.

- Christopher Little: “Every company is a technology company, regardless of what business they think they’re in. A bank is just an IT company with a banking license.”
- Vast majority of capital projects rely on IT.

Importance of Projects

- Projects are the primary mechanism for change inside organizations.
- Management needs to approve, budget for, and be accountable for projects.
- Capital Spending
  - Projects are typically funded through capital spending.
  - 50% of capital spending is now technology-related, even in low-tech industries.
- Reliance on IT
  - Business leaders rely on effective IT management to achieve goals.
  - IT changes are integral to business decisions.

Costs

- Human Costs
  - Feelings of powerlessness and burnout.
  - Fatigue, cynicism, hopelessness, and despair.
  - Long hours, weekend work, decreased quality of life.
- Psychological Impact
  - Systems causing powerlessness are highly damaging.
  - Creates a culture of fear and learned helplessness.
  - People become unwilling or unable to act to avoid future problems.
- Employee Well-being
  - Negative impact on employees and their families.
  - Loss of best employees due to burnout and dissatisfaction.

Costs

- Economic Costs
  - Opportunity cost of missed value creation: $2.6 trillion per year.
  - Equivalent to the annual economic output of France.
- IT Spending
  - 2011: 5% of worldwide GDP ($3.1 trillion) spent on IT.
  - 50% of IT spending on operating costs and maintaining systems.
  - $520 billion wasted on urgent and unplanned work or rework.
- Potential Value Creation
  - Adopting DevOps could halve waste and increase value creation.
  - Potential to create $2.6 trillion of value per year through better management and operational excellence.

Ideal DevOps Environment

- Independent and Agile Teams:
  - Small teams independently implement features in production-like environments.
  - Deploy code quickly, safely, and securely during business hours.
  - Use self-service platforms to deliver value frequently.
- Automation and Immediate Feedback:
  - Fast, automated tests ensure secure, deployable code.
  - Immediate visibility into deployment effects fosters quick fixes and learning.
  - Pervasive telemetry detects and corrects problems early.
- Organizational Outcomes:
  - Achieve low-stress work environments and marketplace success.
  - Build resilient systems through fault injection and large-scale tests.

Ideal DevOps Environment

- Customer-Centric Development:
  - Deliver value with controlled, predictable, low-stress releases.
  - Test and evolve features pre-launch using feature toggles.
  - Ensure customer satisfaction with seamless updates and bug fixes.
- Collaborative and Resilient Culture:
  - Foster a hypothesis-driven, experimental approach to development.
  - Conduct blameless post-mortems and planned failure exercises.
  - Reward problem-solving, risk-taking, and innovation.
- Continuous Learning and Improvement:
  - Treat process improvements as experiments for long-term goals.
  - Host internal technology conferences to enhance skills.
  - Encourage ownership and build confidence in team contributions.

The Business value of DevOps

- Data collected from over 25,000 technology professionals (2013-2016).
- Throughput Metrics
  - Code and change deployments: 30 times more frequent.
  - Code and change deployment lead time: 200 times faster.
- Reliability Metrics
  - Production deployments: 60 times higher change success rate.
  - Mean time to restore service: 168 times faster.
- Organizational Performance Metrics
  - Productivity, market share, and profitability goals: 2 times more likely to exceed.
  - Market capitalization growth: 50% higher over three years.

The Business value of DevOps

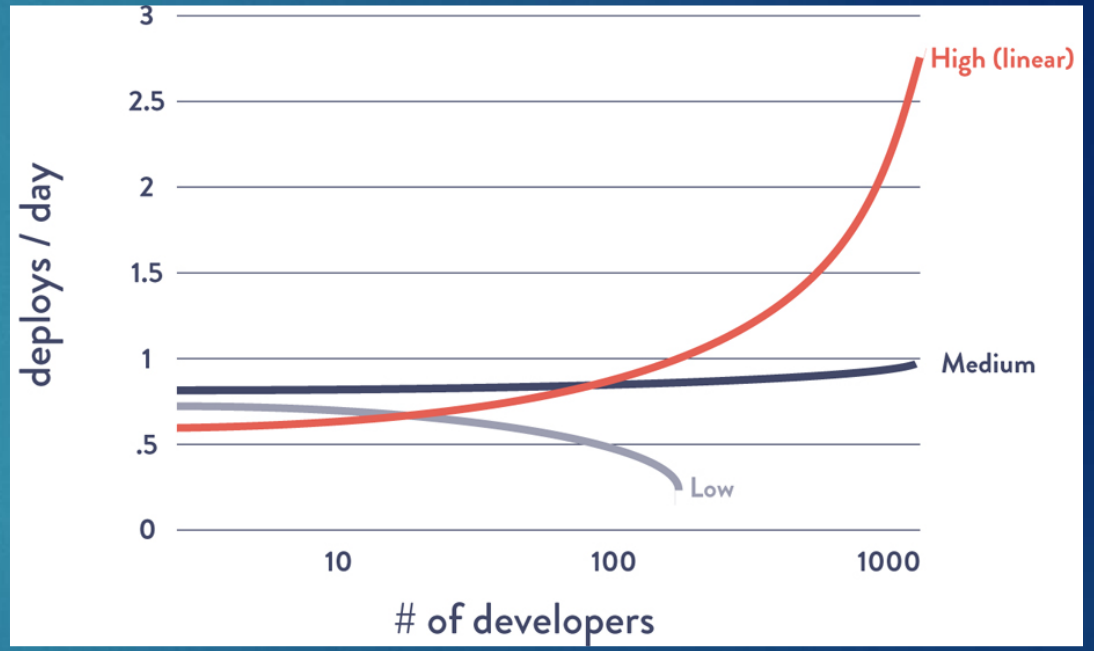
- High Performers vs. Low Performers
  - High performers are more agile and reliable.
  - Lead times measured in minutes or hours vs. weeks, months, or quarters.
- Employee Satisfaction
  - Higher job satisfaction and lower burnout rates.
  - Employees 2.2 times more likely to recommend their organization as a great place to work.
- Information Security Outcomes
  - Better security outcomes for high performers.
  - 50% less time spent remediating security issues.

DevOps and Productivity

- The Mythical Man-Month
  - Adding more developers can decrease individual and overall productivity.
  - Highlighted by Frederick Brooks in "The Mythical Man-Month."
- In contrast DevOps increases Productivity
  - Right architecture, technical practices, and cultural norms.
  - Small teams develop, integrate, test, and deploy changes quickly and safely.
- Small Teams, Big Impact
  - Randy Shoup's observation: Large organizations with DevOps enable small teams to be productive like startups.
  - Example: Google, Amazon, and Netflix.

DevOps and Productivity

- Deploys Per Day Per Developer
  - 2015 State of DevOps Report findings.
  - High performers scale deployments linearly with team size.
- Performance Comparison
  - Low performers: Deploys per day per developer decrease with team size.
  - Medium performers: Deploys per day per developer stay constant.
  - High performers: Deploys per day per developer increase linearly.



A brief history

- Origins and Philosophy:
  - Described by John Willis as the "convergence of Dev and Ops.
  - "Evolved independently across organizations, showing improbable connections.
  - Builds on decades of lessons from manufacturing and high-reliability organizations.
- Core Principles:
  - Applies Lean, Theory of Constraints, Toyota Production
  - System, and resilience engineering.
  - Draws from high-trust management cultures and servant leadership.

A brief history

- Goals and Outcomes:
  - Creates a collaborative, productive environment with high trust.
  - Achieves quality, reliability, stability, and security at reduced cost and effort.
- Impact on IT Value Stream:
  - Accelerates flow and reliability throughout the technology value stream.
  - Adapts principles from physical manufacturing to IT workflows.
- Historical Roots:
  - Derived from Lean, Toyota Kata, and organizational change management.
  - Seen as a continuation of the Agile software journey started in 2001.

The Lean Movement

- Techniques: Value stream mapping, kanban boards, total productive maintenance.
- Codified for the Toyota Production System in the 1980s.

- Lean Enterprise Institute
  - Started researching Lean applications in other value streams in 1997.
  - Focus on service industry and healthcare.

- Manufacturing lead time as a predictor of quality, customer satisfaction, and employee happiness.
- Small batch sizes as a predictor of short lead times.

The Lean Movement

- Lean Principles
  - Create value for the customer through systems thinking.
  - Embrace scientific thinking, create flow and pull, assure quality at the source, lead with humility, and respect every individual.
- The Agile Manifesto
  - Created in 2001 by seventeen experts in “lightweight methods” in software development.
  - Set of values and principles for adaptive methods compared to waterfall development.
- Key Principles of Agile
  - Deliver working software frequently, with a preference for shorter timescales.
  - Emphasize small batch sizes and incremental releases.
  - Small, self-motivated teams in a high-trust management model.

The Lean Movement

- Impact of Agile
  - Increased productivity and responsiveness of development organizations.
  - Key moments in DevOps history occurred within the Agile community.
- Agile Infrastructure and Velocity Movement
  - 2008 Agile conference: Patrick Debois and Andrew Shafer's session on applying Agile principles to infrastructure.
  - Early days referred to as "Agile system administration."

Agile Infrastructure and Velocity Movement

- John Allspaw and Paul Hammond's presentation: "10 Deploys per Day: Dev and Ops Cooperation at Flickr."
  - Shared goals between Dev and Ops, continuous integration practices.
- Patrick Debois created the first DevOpsDays in Ghent, Belgium.
  - The term "DevOps" was coined.
- Jez Humble and David Farley's concept of continuous delivery.
  - Ensuring code and infrastructure are always in a deployable state.
- Tim Fitz's 2009 blog post on "Continuous Deployment."

Agile Infrastructure and Velocity Movement

- Toyota Kata
  - Mike Rother's book: "Toyota Kata: Managing People for Improvement, Adaptiveness, and Superior Results."
  - Importance of the improvement kata for daily, habitual practice of improvement work.
  - Establishing desired future states and setting target outcomes.
  - Continual improvement of daily work guided by the improvement kata.

The Manufacturing Value Stream

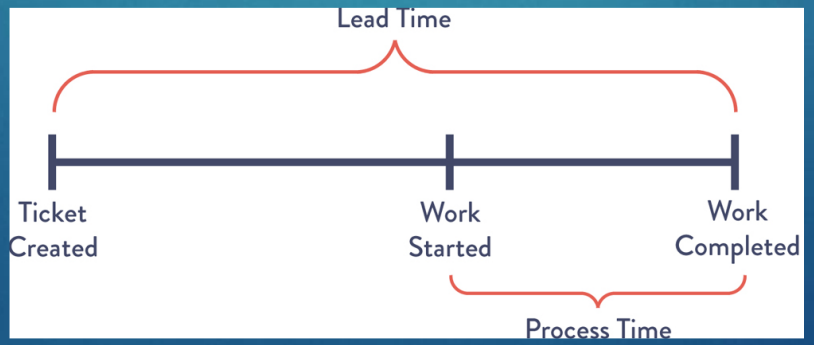
- Defined by Karen Martin and Mike Osterling.
- Sequence of activities to deliver a good or service to a customer.
- Starts with customer order and raw materials release.
- Focus on smooth and even flow of work.
- Small batch sizes, reducing work in process (WIP).
- Preventing rework and optimizing systems toward global goals.

The Technology Value Stream

- Process to convert a business hypothesis into a technology-enabled service or feature.
- Starts with a business objective and ends with value delivery to the customer.
- Agile Development Process
  - Transform ideas into user stories and feature specifications.
  - Implement code, integrate, and test in version control repository.
- Ensuring Fast Flow and Stability
  - Deliver fast flow without causing chaos and disruptions.
  - Avoid service outages, impairments, and security or compliance failures.
- Focus on Deployment Lead Time
  - Begins when a change is checked into version control.
  - Ends when the change is running in production, providing value and feedback.

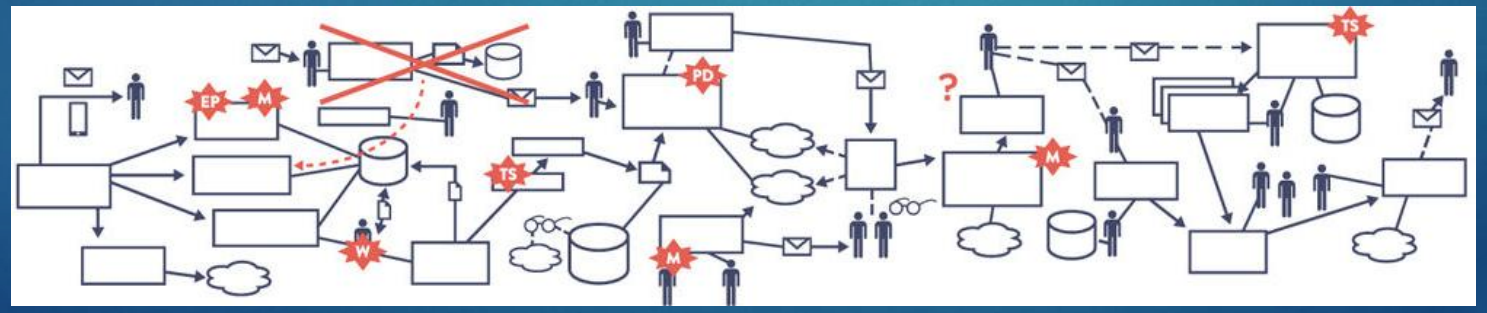
Lean Product Development vs. Lean Manufacturing

- Design and development: High variability and creativity.
- Testing, deployment, and operations: Predictable and mechanistic.
- Testing, deployment, and operations happen simultaneously with design/development.
- Work in small batches and build quality into every part of the value stream.
- Lead time: Starts when the request is made and ends when fulfilled.
- Processing time: Starts when work begins on the customer request, omits queue time.



Common Scenario: Long Deployment Lead Times

- Deployment lead times requiring months.
- Common in large, complex organizations with tightly coupled systems.
- Challenges with Long Lead Times
  - Scarce integration test environments, long test and production environment lead times.
  - High reliance on manual testing and multiple approval processes.

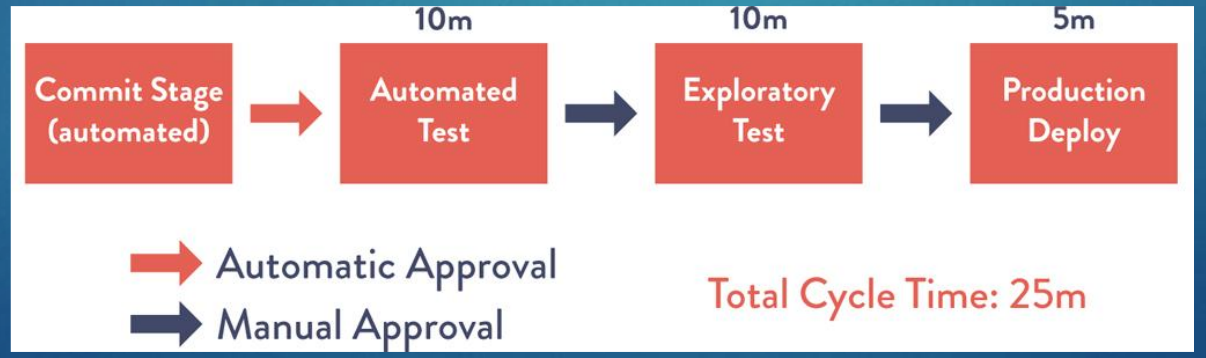


DevOps Ideal: Short Deployment Lead Times

- Fast, constant feedback for developers.
- Quick and independent implementation, integration, and validation of code.
- Achieving Fast Deployment
  - Continually check in small code changes into version control.
  - Perform automated and exploratory testing, deploy into production.
- Modular Architecture
  - Modular, well-encapsulated, and loosely coupled architecture.
  - Small teams work with high autonomy, failures are small and contained.

DevOps Ideal: Short Deployment Lead Times

- Deployment Lead Time in Minutes
  - Deployment lead time measured in minutes or hours.
  - High confidence in changes operating as designed in production.
- Percent Complete and Accurate (%C/A)
  - Metric reflecting the quality of output at each step.
  - Work is "usable as is" without needing corrections or clarifications.



Introduction to The Three Ways

- The First Way
  - Enables fast left-to-right flow of work from Development to Operations to the customer.
  - Make work visible, reduce batch sizes, build in quality, and optimize for global goals.
- Benefits of The First Way
  - Reduces lead time for fulfilling requests.
  - Increases quality, throughput, and innovation.
- Practices of The First Way
  - Continuous build, integration, test, and deployment.
  - Creating environments on demand, limiting WIP, and building safe-to-change systems.

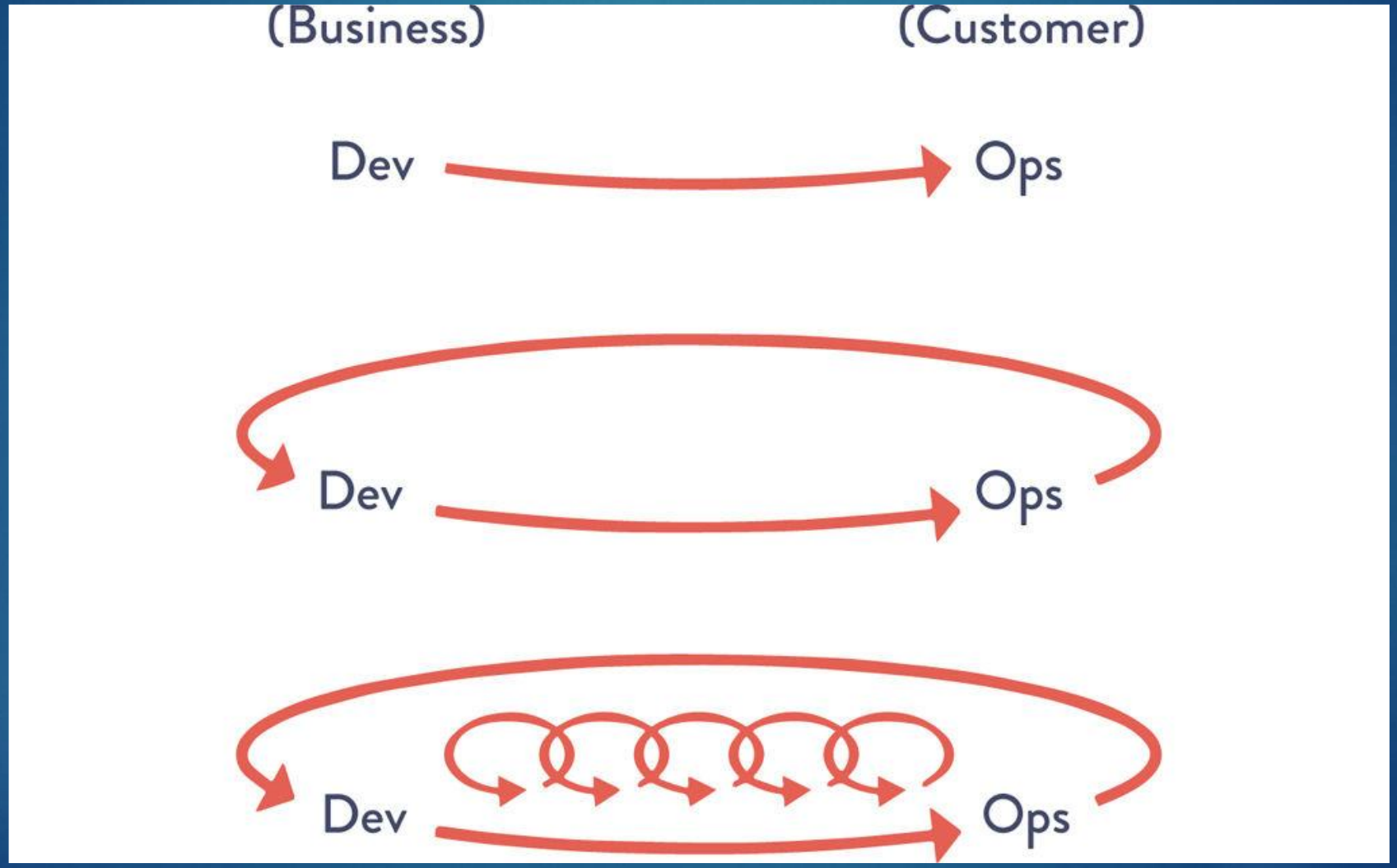
Introduction to The Three Ways

- The Second Way
  - Enables fast and constant flow of feedback from right to left.
  - Amplify feedback to prevent problems and enable faster detection and recovery.
- Benefits of The Second Way
  - Creates quality at the source.
  - Generates knowledge where needed and creates safer systems of work.
- Practices of The Second Way
  - Shorten and amplify feedback loops.
  - Swarm problems until effective countermeasures are in place.

Introduction to The Three Ways

- The Third Way
  - Creates a generative, high-trust culture.
  - Supports experimentation, risk-taking, and organizational learning.
- Benefits of The Third Way
  - Shortens and amplifies feedback loops.
  - Enables faster learning and competitive advantage.
- Practices of The Third Way
  - Design systems to multiply the effects of new knowledge.
  - Transform local discoveries into global improvements.

Introduction to The Three Ways



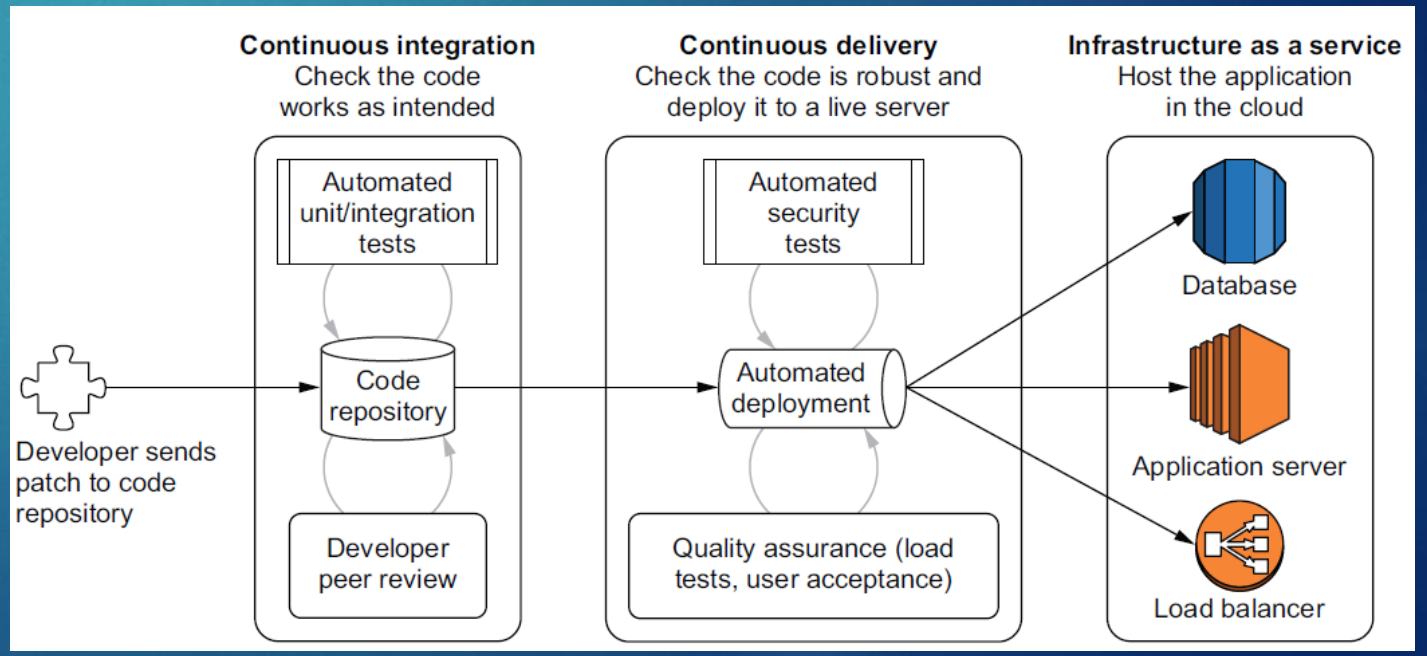
Security in DevOps

Continuous Integration (CI)

- Integrating security practices within the DevOps lifecycle to ensure safe and reliable systems.
- A development practice that integrates code changes into a shared repository frequently, ensuring software reliability.
- Process:
  - Developers submit patches.
  - Automated unit and

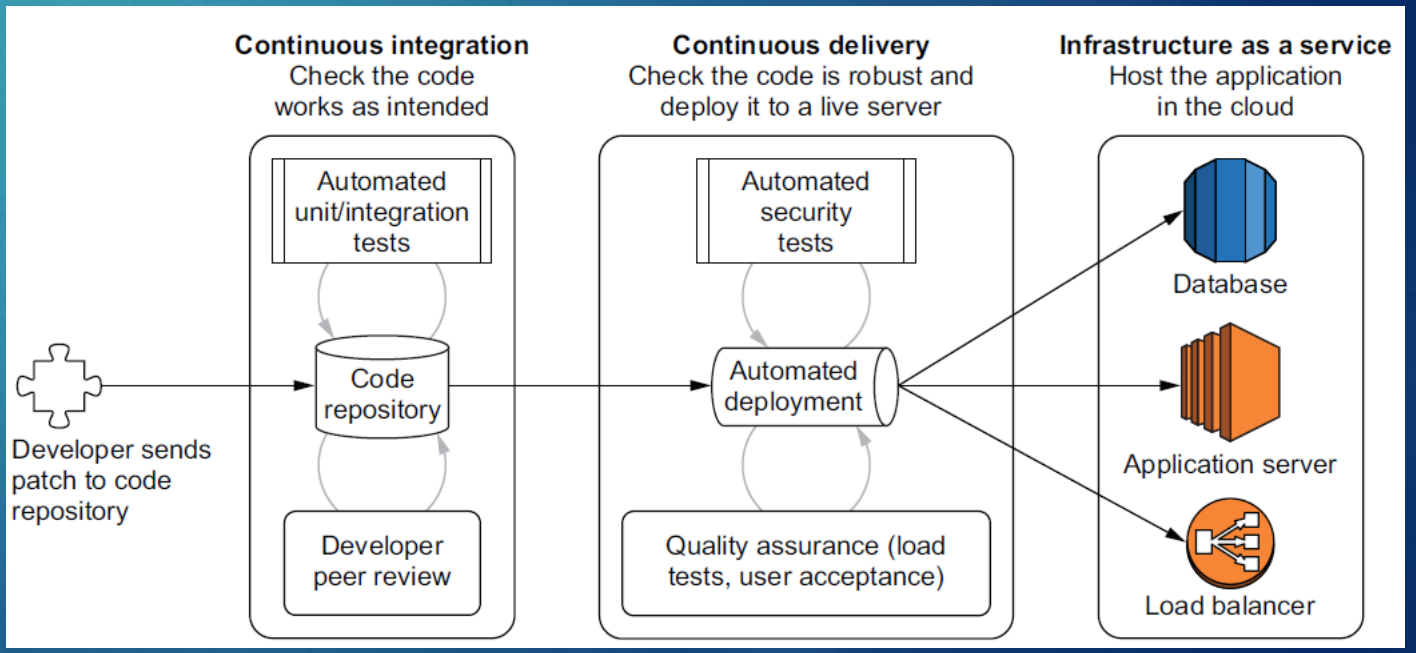
  Integration tests verify

  functionality.
  - Peer reviews validate quality.



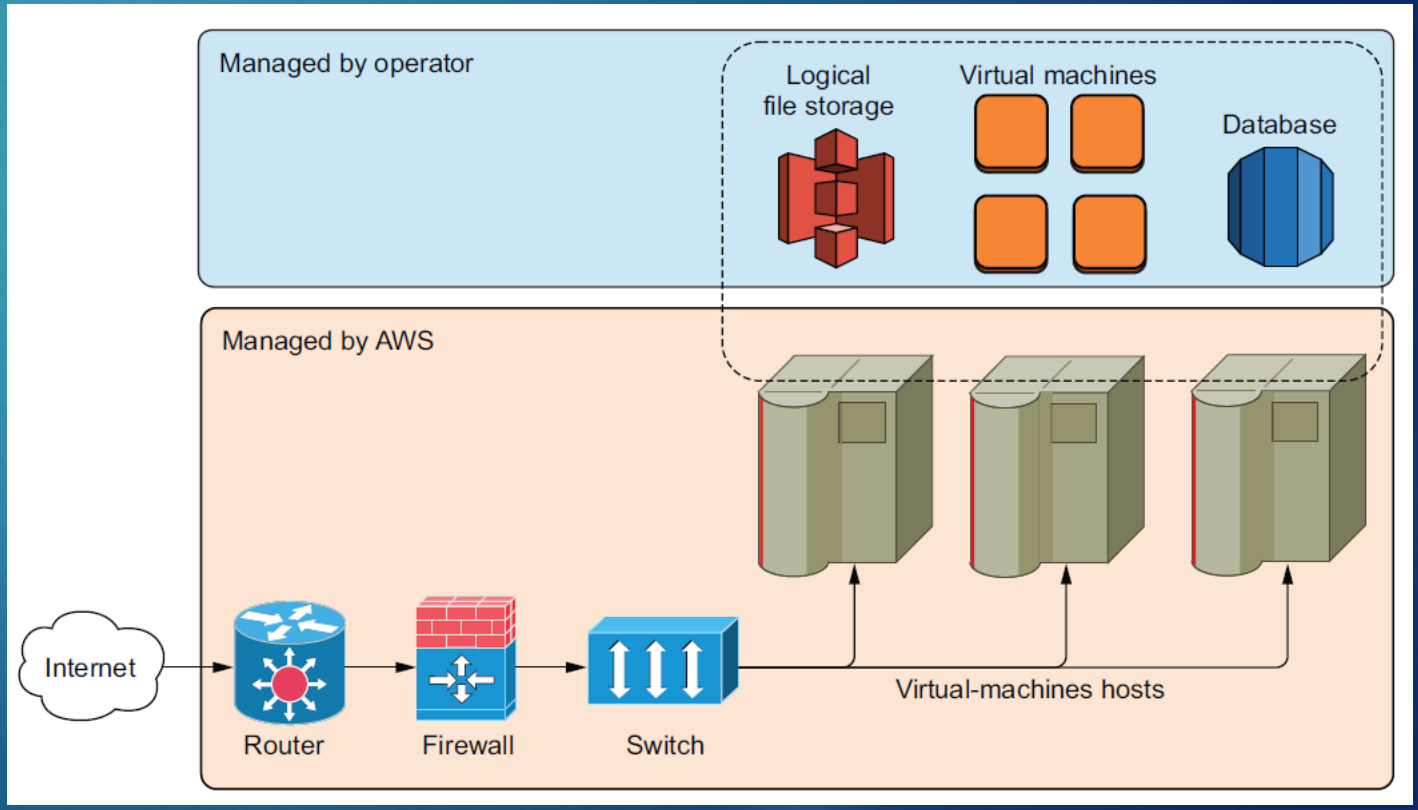
Continuous Delivery (CD)

- Automates the deployment of software changes to production environments after rigorous testing.
- Process:
  - Source code updates trigger automated pipelines.
  - Quality assurance ensures readiness for deployment.
  - Changes are promoted to production.



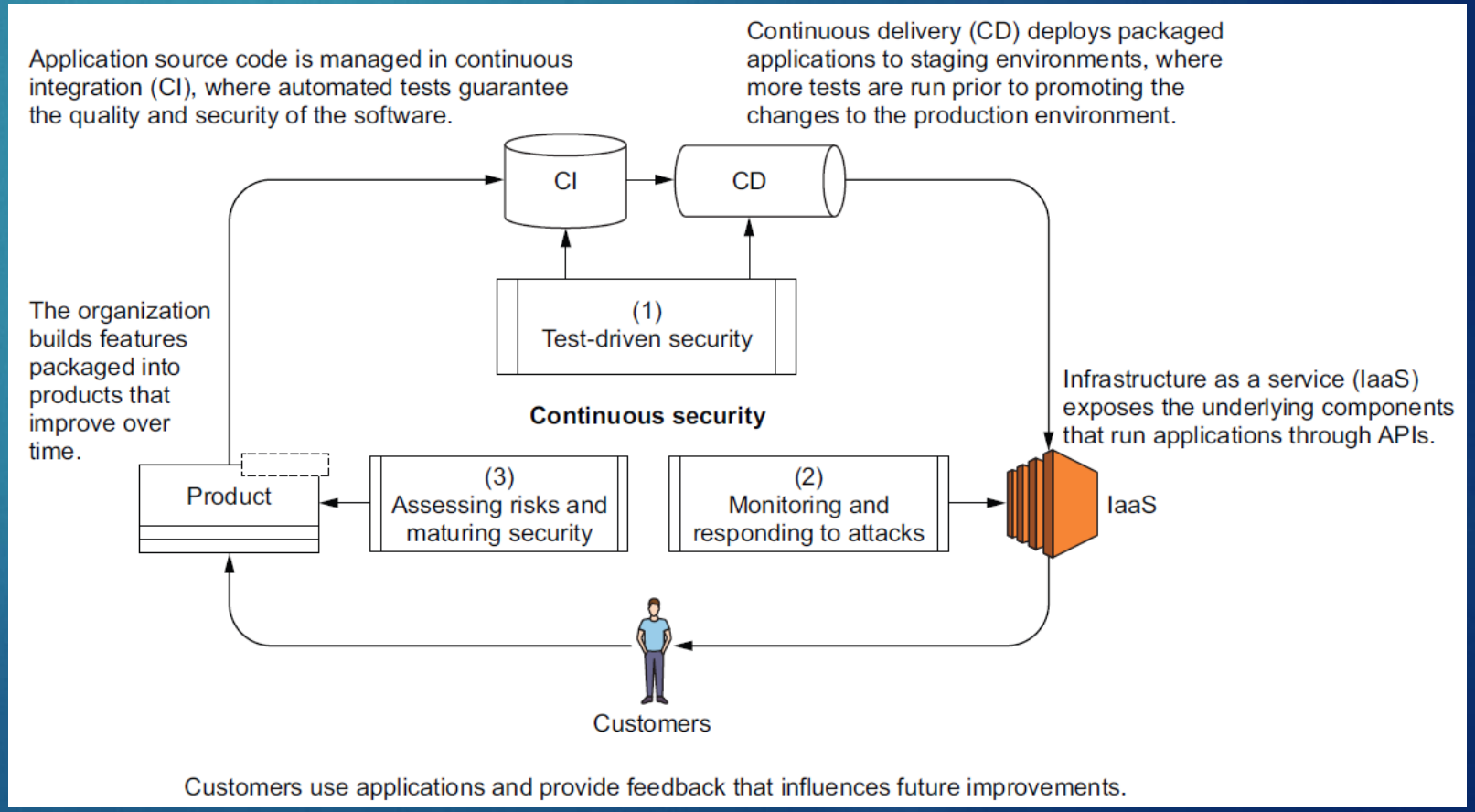
Infrastructure as a Service (IaaS)

- A cloud computing model providing virtualized infrastructure over the internet.
- Advantages:
  - Reduces operational complexity.
  - Allows scalability and flexibility.
- Example Tools: AWS, Azure, OpenStack.



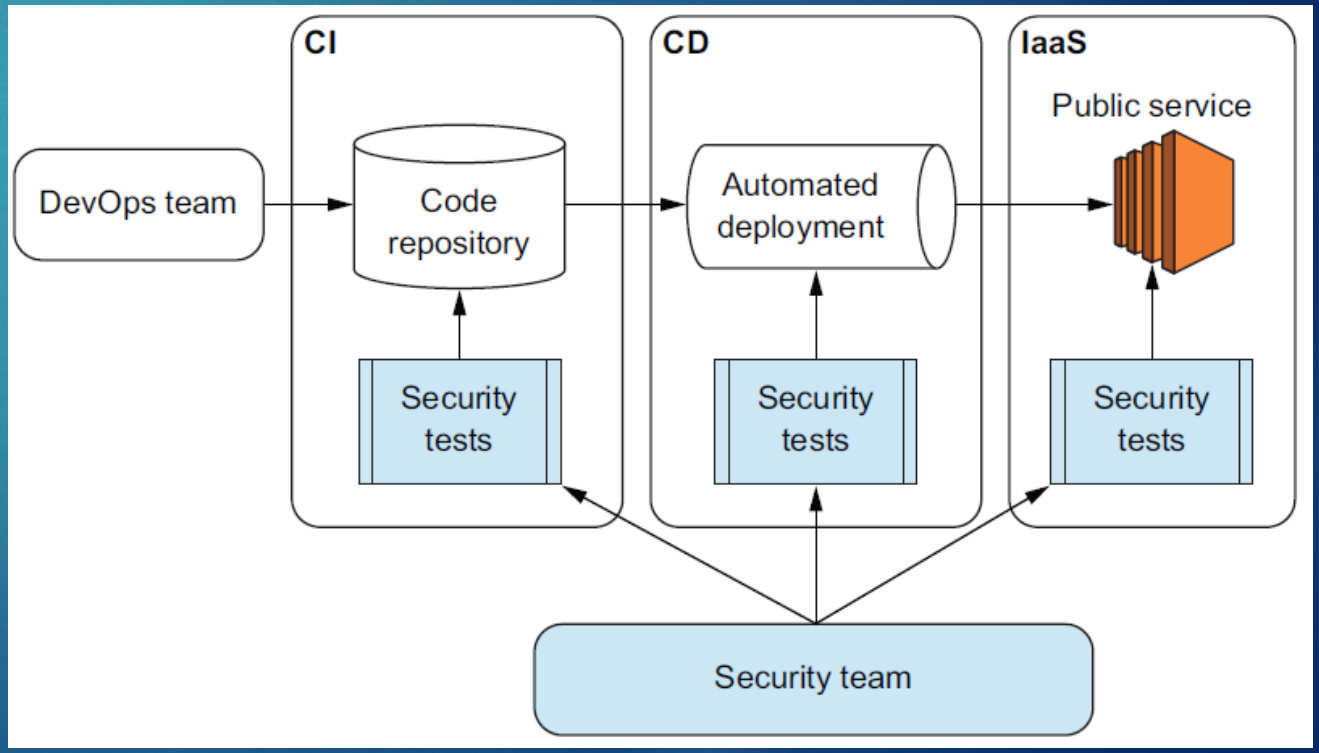
Continuous Security Model

- An iterative process integrating security into all phases of the DevOps lifecycle.
- Phases:
  - Test-Driven Security.
  - Monitoring and Response.
  - Risk Assessment and Maturity.



Test-Driven Security (TDS)

- A strategy to write security tests first, ensuring all components meet predefined security requirements before deployment.
- Key Features:
  - Automation of security tests in CI/CD pipelines.
  - Reusability of test cases.
- Benefits:
  - Early detection of security gaps.
  - Improved clarity and documentation of expectations.



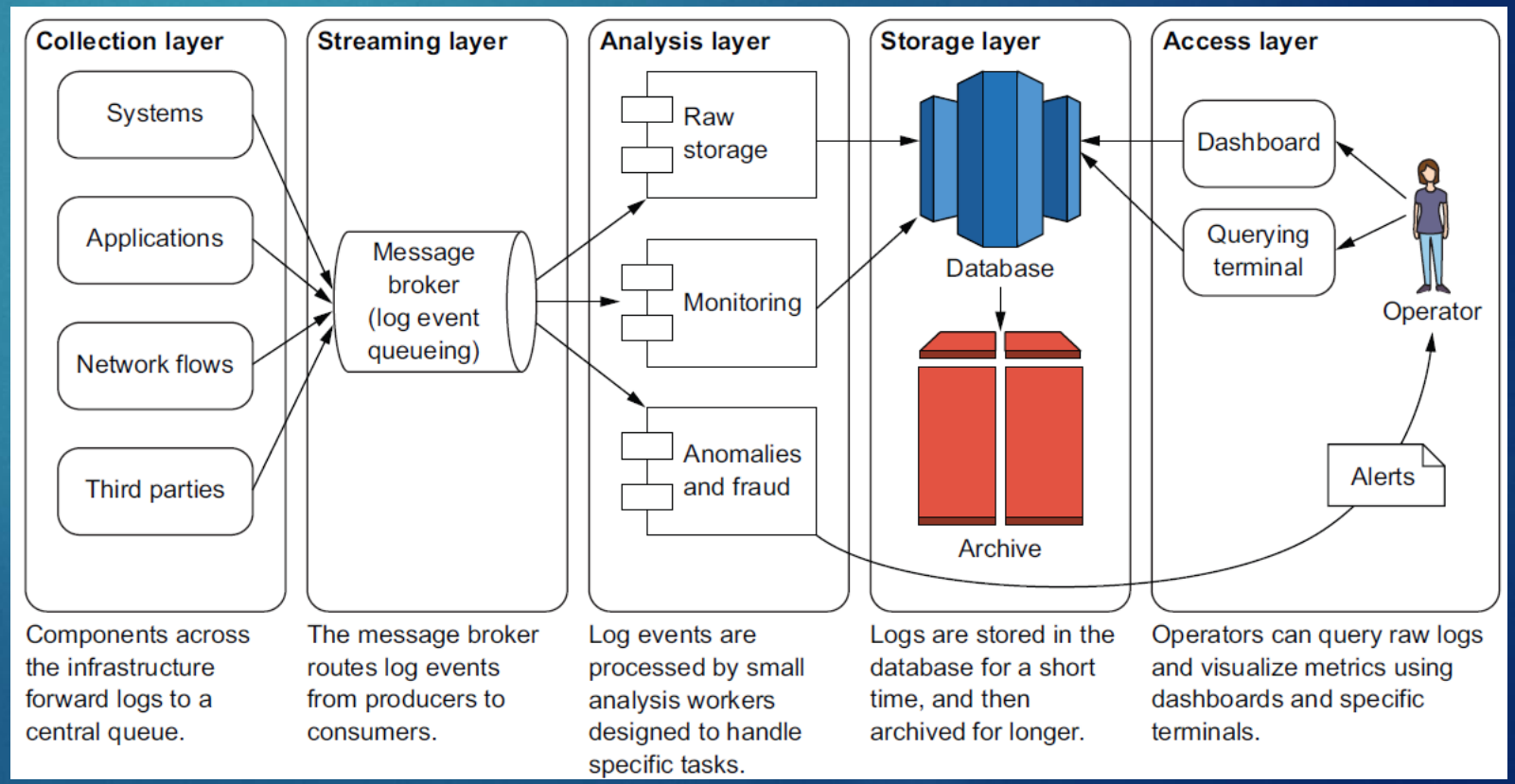
Logging and Monitoring Pipelines

- Centralized systems for collecting, analyzing, and storing logs to monitor application health and detect anomalies.
- Components:
  - Collection layer:

    Records log events.
  - Analysis layer:

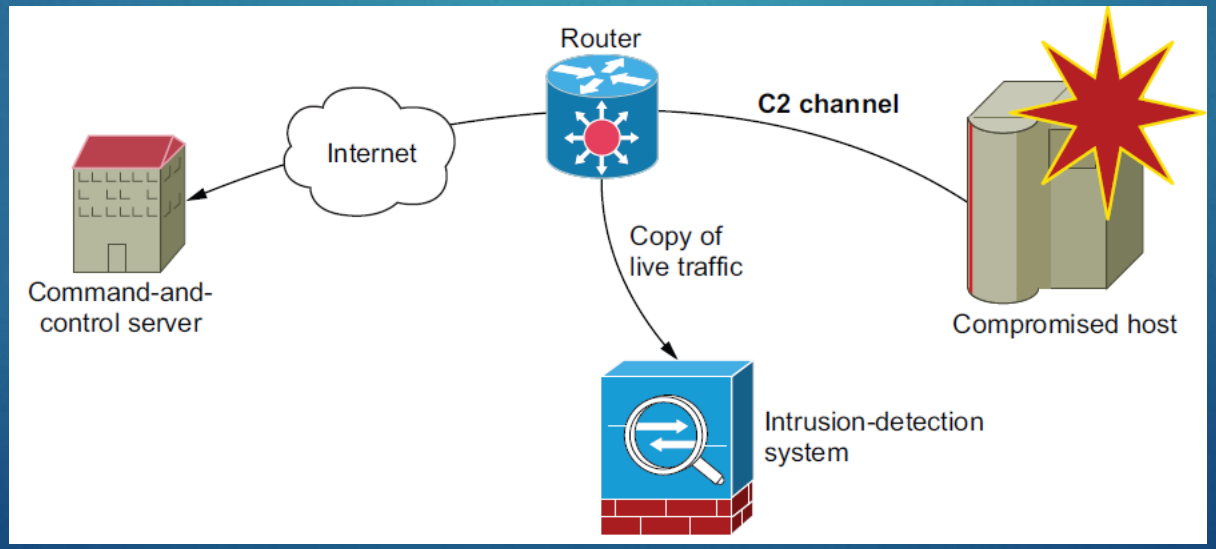
    Detects fraud and raises alerts.
  - Storage layer:

    Archives logs for forensics.



Intrusion Detection Systems (IDS)

- Tools that monitor network traffic for suspicious activities.
- Process:
  - Analyze traffic patterns.
  - Detect command-and-control (C2) channels.
  - Alert security teams of potential breaches.



Risk Assessment in DevOps

- Evaluating and prioritizing potential security threats to ensure resource-efficient mitigation.
- Approach:
  - Small, frequent iterations.
  - Organization-wide participation.
- Benefits:
  - Proactive risk management.
  - Enhanced resilience.

Case Study - Facebook

- Facebook handles billions of users daily.
- Emphasis on scalability, automation, and security.
- Open-source contributions reflect their DevSecOps ethos.
- Automated Testing at Scale
  - Comprehensive CI/CD pipelines.
    - CI/CD pipelines automate code integration, testing, and deployment processes.
  - Integration of static (SAST) and dynamic (DAST) analysis tools.
    - SAST: Identifies vulnerabilities in the source code without executing it.
    - DAST: Simulates attacks on running applications to detect runtime vulnerabilities.
  - Proactive vulnerability detection during development.

Case Study - Facebook

- Key Tools and Technologies
  - OSquery:
    - An open-source tool for endpoint visibility and security monitoring.
    - Enables querying system data using SQL-like queries.
  - Phabricator:
    - A code review tool developed by Facebook.
    - Facilitates collaboration among developers during code changes.
  - Custom tooling for automation and security checks.

Case Study - Facebook

- Cultural Practices
  - Move Fast Philosophy: A principle balancing speed and careful oversight.
  - Developer empowerment with security training.
    - Ensures developers understand and implement security best practices.
  - Security champions embedded in development teams.
    - Act as liaisons between security and development teams.
- Handling Vulnerabilities
  - Bug bounty programs:
    - Incentivize external researchers to identify vulnerabilities.
  - Real-time monitoring:
    - Detects and mitigates security incidents promptly.
  - Integration of external reports with internal workflows.

Case Study - Facebook

- Key Achievements
  - High-frequency deployments with minimal downtime.
  - Transparency through open-source initiatives.
  - Strong security posture for large-scale systems.
- Lessons Learned
  - Early integration of security yields significant returns.
  - Collaboration between teams reduces operational friction.
  - Continuous improvement is essential for evolving threats.

Case Study - Facebook

- Impact on the Industry
  - Influence on open-source DevSecOps tools.
  - Setting benchmarks for security in CI/CD.
  - Adoption of their practices by other organizations.
- Key Takeaways
  - Facebook's success lies in balancing speed and security.
  - Tools like OSquery enhance proactive defense.
  - Security is a shared responsibility across all teams.

Case Study - Amazon

- Amazon’s rapid pace of innovation demands robust DevSecOps.
- Heavy reliance on automation and "security as code."
- CI/CD Security
  - Use of AWS CodePipeline:
    - Automates build, test, and deploy workflows.
  - Automated security checks in build and deploy phases.
  - Continuous monitoring with AWS CloudTrail:
    - Tracks user activity and API usage for security audits.

Case Study - Amazon

- Infrastructure Security
  - IAM Policies: Define granular access controls to enforce least privilege.
  - AWS Config: Ensures compliance by tracking configuration changes.
  - Security Groups:
    - Act as virtual firewalls for controlling inbound and outbound traffic.
- Security at Scale
  - Auto-scaling policies: Manage traffic spikes securely.
  - Robust DDoS protection with AWS Shield.
    - Provides always-on detection and mitigation.
  - Use of WAF (Web Application Firewall):
    - Protects web applications from common exploits.

Case Study - Amazon

- Incident Response
  - Automated incident response with AWS Lambda:
    - Executes custom scripts in response to security triggers.
  - AWS Security Hub: Centralizes threat detection and response workflows.
  - Proactive audits and penetration testing.
- Developer Empowerment
  - Developer access to tools like AWS Inspector:
    - Automates security assessments for applications.
  - Security training integrated into onboarding.
  - Emphasis on "you build it, you secure it" philosophy.

Case Study - Amazon

- Key Tools and Services
  - AWS GuardDuty:
    - Provides threat detection and monitoring.
  - AWS Secrets Manager:
    - Secures and manages application secrets.
  - AWS Key Management Service (KMS):
    - Manages encryption keys for secure data handling.

Case Study - Amazon

- Lessons Learned
  - Automation reduces the margin for human error.
  - Scalability requires built-in security at every layer.
  - Regular audits ensure compliance with evolving standards.
- Key Takeaways
  - Security as code is essential for rapid innovation.
  - Amazon's practices ensure secure scaling.
  - AWS tools provide end-to-end security capabilities.

Case Study - Google

- Google operates at a global scale with complex systems.
- Emphasis on infrastructure security and containerization.
- Site Reliability Engineering (SRE)
  - Combines software engineering and IT operations for high system reliability.
  - Focus on availability, latency, and performance.
  - Security integrated into the SRE workflow.
- Zero Trust Security Model
  - "BeyondCorp" initiative:
    - Removes reliance on VPNs by enforcing identity-based access.
  - Continuous verification of user and device security.

Case Study - Google

- Container Security
  - gVisor: Sandbox that provides isolation for containers.
  - Kubernetes: Manages container orchestration securely at scale.
  - Automated vulnerability scans for container images.
- Open Source Contributions
  - Kubernetes: Sets industry standards for container orchestration.
  - Istio: Manages service mesh security for microservices.
  - Open Policy Agent (OPA):
    - Enables policy enforcement in cloud environments.

Case Study - Google

- Incident Management
  - Centralized incident response system.
  - Regular disaster recovery drills.
  - Real-time dashboards for monitoring and mitigation.
- Developer Support
  - Internal platforms for secure coding practices.
  - Security reviews integrated into the SDLC.
  - Mandatory training for secure development.

Case Study - Google

- Lessons Learned
  - Zero Trust enhances resilience against breaches.
  - Container security is critical for cloud-native environments.
  - Collaboration fosters innovation in security tools.
- Key Takeaways
  - Google's focus on SRE ensures operational and security excellence.
  - Open-source tools like Kubernetes advance industry standards.
  - Zero Trust is pivotal for modern security frameworks.