Traditional Model (Waterfall / Configuration Management)

- **Team Separation:**
  - **Development Team: Responsibilities:** Planning, coding, building, and automated testing. **Incentives:** Ship new features.
  - **Release Team: Responsibilities:** Receives a "final" version from development, builds a release version, tests it, prepares release documentation, and releases the software.
  - **Support/Maintenance Team: Responsibilities:** Handles customer support and implements software changes when necessary.
- **Key Challenge:** The "Wall of Confusion" where delays and miscommunication occur between the separate teams.

DevOps Model

- **Integrated Responsibilities:** All team members share responsibility for developing, delivering, and supporting the software.
- **Key Motivators for Adoption: Agile Development:** Reduced overall development time. **Unified Team Approach:** Inspired by companies like Amazon, where the same team develops and supports the service. **SaaS Revolution:** Moving away from physical media and downloads.

Core DevOps Principles

Shared Responsibility

- **Everyone is Responsible:** From development through deployment to support, every team member participates in all stages of the software lifecycle.

Automation

- **Fundamental Belief:** Anything that can be automated should be.
- **Areas for Automation:** Testing, Deployment, System monitoring and support
- **Encoding Automation:** Use of scripts and system models that are checked, reviewed, versioned, and stored in the project repository.

Measurement & Data-Driven Change

- **"Measure first, change later" Approach:** Collect data on system performance and operational metrics. Use this data to drive improvements in processes and tools.

Benefits of DevOps

- **Faster Deployment:** Reduced communication delays and streamlined handoffs.
- **Reduced Risk:** Incremental changes minimize the potential for large-scale failures.
- **Faster Repair:** Collaborative troubleshooting leads to quicker problem resolution.
- **Enhanced Productivity:** Integrated teams and automation create a more efficient workflow.

DevOps Tools & Code Management

Essential Components

- **Source Code Management:** Tools like Git and GitHub to track changes and manage code repositories.
- **Automated System Building:** Continuous Integration (CI) systems automatically build and test the software with every change.
- **Continuous Deployment:** Automated pipelines that release new versions as soon as changes are verified.

Impact on Time and Cost

- **Time Savings:** Drastically reduce integration, deployment, and delivery times.
- **Cost Efficiency:** Although initial setup of automated systems is resource-intensive, long-term savings are realized through regular, incremental updates.

Continuous Integration (CI)

Process Overview

- **Commit & Build Cycle:** Every time a developer commits changes to the master branch, an executable version of the system is automatically built and tested.
- **Integration Server:** Receives push notifications, initiates builds, and runs system tests.

Best Practices

- **Avoid "Breaking the Build":** Ensure that changes do not cause system tests to fail. Adopt an "integrate twice" approach: Test changes locally on your own machine before pushing them to the shared repository.
- **Speed is Critical:** The integration process must be fast; delays (e.g., due to database population or compiling large codebases) can hamper developer productivity.

Continuous Delivery

- **Definition:** Ensuring that every change is ready for production by replicating the product's operating environment in a test stage.
- **Testing Stages:** Functionality, load, and performance tests in an environment that mimics production.
- **Outcome:** Software is verified as production-ready once all tests pass.

Continuous Deployment

- **Definition:** Every successful change is automatically deployed to production.
- **Deployment Pipeline: Staging:** After initial integration tests, a replica of the production environment is created. **Switching:** New requests are momentarily paused; once the build is complete, traffic is switched over to the updated version.
- **Key Benefits:**
  - **Reduced Costs:** Automation eliminates the manual errors and delays inherent in manual deployment.
  - **Faster Problem Solving:** Smaller changes make it easier to identify and fix issues.
  - **Customer Feedback:** Rapid deployment allows for quicker feedback loops.
  - **A/B Testing:** Option to deploy different versions across servers to gauge feature performance.

Infrastructure as Code (IaC)

Concept and Rationale

- **Definition:** The practice of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration.
- **Why It Matters:** Keeps test and production environments in sync. Automates the deployment of services across multiple servers.
- **Tools:** Examples include Puppet and Chef.

Benefits of IaC

- **Consistency & Reproducibility:** Environments are defined in code, ensuring that installations occur in the same sequence every time.
- **Lower Costs & Risks:** Reduces management overhead and minimizes human error.
- **Visibility & Recovery:** The infrastructure model is versioned and stored in code management systems, enabling easy rollback if changes cause problems.

Containers

Overview

- **Definition:** Containers provide isolated execution environments that run on a shared operating system.
- **Docker & Dockerfiles:** A Dockerfile is used to define the software environment and build an executable container image.

Advantages

- **Consistency:** Containers ensure identical environments for development, testing, and production.
- **Flexibility:** Allows running test systems alongside operational systems.
- **Easy Updates:** Updating software involves rebuilding the container image and deploying it alongside the existing one, with service requests gradually shifted.

Four Key Types of Measurement

1. **Process Measurement:** Data collected about development, testing, and deployment processes.
2. **Service Measurement:** Metrics on the software's performance, reliability, and user satisfaction.
3. **Usage Measurement:** Analysis of how customers interact with the product.
4. **Business Success Measurement:** Evaluates the product's overall contribution to business objectives.

DevOps Scorecard (Based on Payal Chakravarty, IBM)

- **Desired Decreases:** Number of failed deployments (i.e., reduced time to recovery after a failure). Time between development and deployment.
- **Desired Increases:** Deployment frequency. Number of lines of code shipped.
- **Stability Goals:** Maintain or improve availability and performance.
- **Customer Impact:** Decrease in customer complaints. Increase in new customers.

Automated Monitoring

- **Integration with Software:** Systems should be set up to automatically collect performance and availability data.
- **Purpose:** Enable continuous improvement by providing real-time insights into both the process and product performance.

Summary and Final Tips

- **DevOps is a Paradigm Shift:** Merges development, deployment, and support into a unified, automated, and data-driven process.
- **Key Focus Areas: Automation:** Test, build, deploy, and monitor everything. **Continuous Improvement:** Use metrics to drive changes. **Collaboration:** Break down silos by sharing responsibilities across teams.
- **Remember:** The move from a traditional, separated model to an integrated DevOps approach leads to faster deployment, reduced risks, quicker problem resolution, and overall improved productivity.