Introduction to DevOps

Understanding the Basics

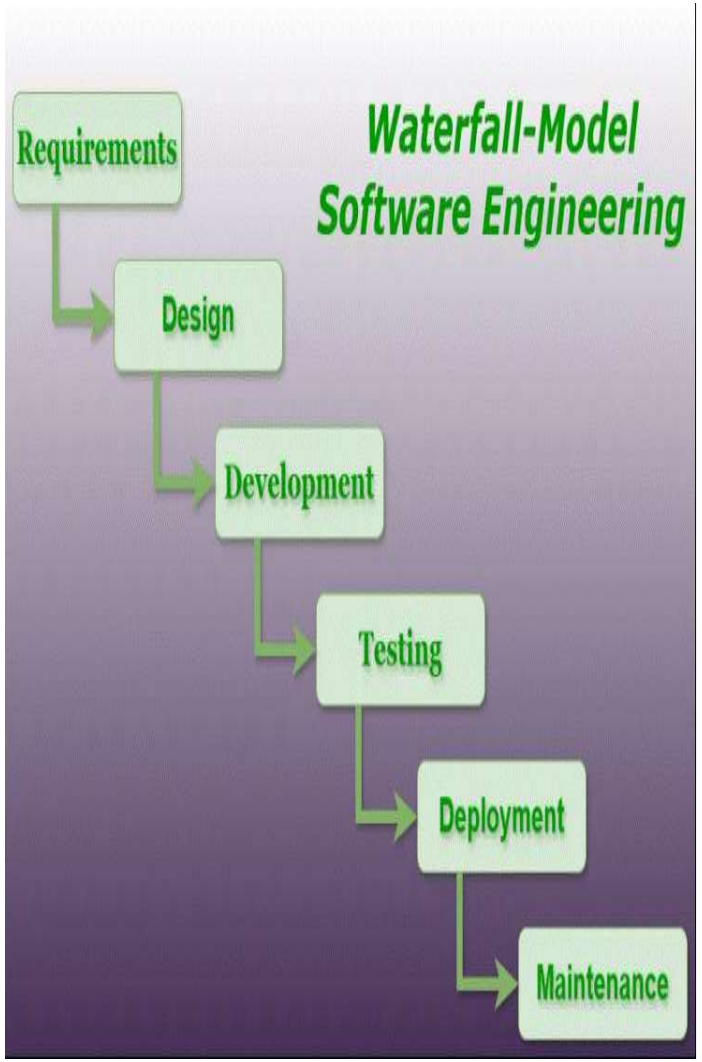
Presented by: [Santhosh Kiran J]

SDLC Waterfall Development Model

The **Waterfall Model** is a traditional software development methodology that follows a linear and sequential approach. Each phase must be completed before the next one begins, and there is no overlapping in the phases.

Phases involved:

- **Requirement Analysis**: Gathering and documenting what the software needs to do.
- **System Design**: Creating the architecture of the system.
- **Implementation**: Writing the actual code.
- **Integration and Testing**: Combining all the pieces and testing for defects.
- **Deployment**: Releasing the software to users.
- **Maintenance**: Performing ongoing support and updates.



SDLC Waterfall Development Model

Advantages

- **Simplicity**: Easy to understand and manage due to its linear nature.
- **Structured Approach**: Each phase has specific deliverables and a review process.
- **Documentation**: Extensive documentation is produced at each phase.

Disadvantages

- **Inflexibility**: Difficult to go back to any stage once it’s completed.
- **Risk**: High risk and uncertainty due to the lack of iteration.
- **Late Testing**: Testing phase comes late in the development process, which can lead to issues being discovered late.

Agile Development Model

The Agile development model is a popular approach in software engineering that emphasizes flexibility, collaboration, and customer satisfaction. It was introduced to address the limitations of traditional models like the Waterfall model, which often struggled with changing requirements and long development cycles.

Key Principles of Agile Development

- **Individuals and Interactions over Processes and Tools:** Emphasizes the importance of communication and collaboration among team members.
- **Working Software over Comprehensive Documentation:** Focuses on delivering functional software frequently, rather than extensive documentation.
- **Customer Collaboration over Contract Negotiation:** Involves customers throughout the development process to ensure the final product meets their needs.
- **Responding to Change over Following a Plan:** Adapts to changing requirements, even late in the development process.

Agile Methodologies

There are several methodologies under the Agile umbrella, each with its unique practices and focus areas:

Scrum:

- Framework: Divides the project into small, manageable units called sprints, typically lasting 2-4 weeks.
- Roles: Includes a Scrum Master (facilitator), Product Owner (defines priorities), and Development Team.
- Ceremonies: Daily stand-ups, sprint planning, sprint reviews, and retrospectives.
- Example: A software company developing a new feature for their app might use Scrum to break down the work into sprints, allowing for regular feedback and adjustments.

Agile Methodologies

Kanban:

- Framework: Focuses on visualizing the workflow and limiting work in progress to improve efficiency.
- Principles: Visualize work, limit work in progress, manage flow, make process policies explicit, and improve collaboratively.
- Example: A support team might use a Kanban board to track customer issues, ensuring they are resolved efficiently without overloading team members.

Extreme Programming (XP):

- Practices: Emphasizes technical excellence and includes practices like pair programming, test-driven development, and continuous integration.
- Example: A startup might use XP to ensure high-quality code and rapid delivery of new features.

Agile Methodologies

Feature-Driven Development (FDD):

- **Approach**: Focuses on building and designing features based on client-valued functionality.
- **Example**: A financial software company might use FDD to develop specific features like transaction processing or reporting.

Steps in Agile Development

1. **Requirement Gathering**: Collaborate with stakeholders to gather and prioritize requirements.
2. **Design**: Create a high-level design that can be adapted as the project progresses.
3. **Development/Iteration**: Develop the software in small, incremental releases.
4. **Testing/Quality Assurance**: Continuously test the software to ensure quality.
5. **Deployment**: Deploy the software to production in small, manageable increments.
6. **Feedback**: Gather feedback from users and stakeholders to refine and improve the product.

Fig. Agile Model

Advantages of Agile

- **Flexibility**: Easily adapts to changing requirements.
- **Customer Satisfaction**: Involves customers throughout the process, ensuring the final product meets their needs.
- **Improved Quality**: Continuous testing and feedback help maintain high quality.
- **Faster Delivery**: Frequent releases ensure that valuable features are delivered quickly.

Disadvantages of Agile

- **Scope Creep**: Continuous changes can lead to scope creep if not managed properly.
- **Requires Experience**: Teams need to be experienced and well-coordinated to implement Agile effectively.
- **Documentation**: Less emphasis on documentation can be a drawback for complex projects.

Spotify's Agile Journey - Use case

Spotify adopted a unique Agile framework known as the "Spotify Model," which has been widely recognized for its effectiveness.

Background: Spotify, a leading music streaming service, needed a way to manage its rapidly growing development teams spread across multiple cities. They aimed to maintain high levels of innovation and efficiency while scaling their operations.

Agile Methodology: Spotify adapted Scrum into their own model, which includes:

- **Squads**: Small, cross-functional teams that operate like mini-startups, each responsible for a specific aspect of the product.
- **Tribes**: Groups of squads that work in related areas, ensuring alignment and collaboration.
- **Chapters**: Communities of practice within tribes, focusing on specific skills or technologies.
- **Guilds**: Informal groups that share knowledge and best practices across the organization.

Spotify’s Agile Journey - Use case

Implementation:

- **Squads** work autonomously, following Agile principles like iterative development and continuous delivery.
- **Tribes** facilitate coordination and knowledge sharing among squads.
- **Chapters** ensure technical excellence and consistency.
- **Guilds** promote innovation and learning across the company.

Outcomes:

- **Increased Innovation**: The model allowed Spotify to innovate rapidly, releasing new features and improvements frequently.
- **Enhanced Collaboration**: The structure promoted better communication and collaboration across teams.
- **Scalability**: Spotify successfully scaled its operations without sacrificing agility or quality.

What is DevOps?

- **Definition:** DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the systems development life cycle and provide continuous delivery with high software quality[1].
- **Usage:** Used to improve collaboration between development and operations teams, automate processes, and enhance the speed and reliability of software delivery.

❑**Pros:**

- ✓ Faster delivery of features
- ✓ Improved collaboration and communication
- ✓ Higher quality and reliability of software

❑**Cons:**

- ❖ Requires cultural change
- ❖ Initial setup can be complex
- ❖ Continuous monitoring and maintenance needed

DevOps Principles

- Key principles include collaboration, automation, continuous improvement, customer-centric action, and monitoring[2].
- **Usage:** These principles guide the implementation of DevOps practices in organizations.

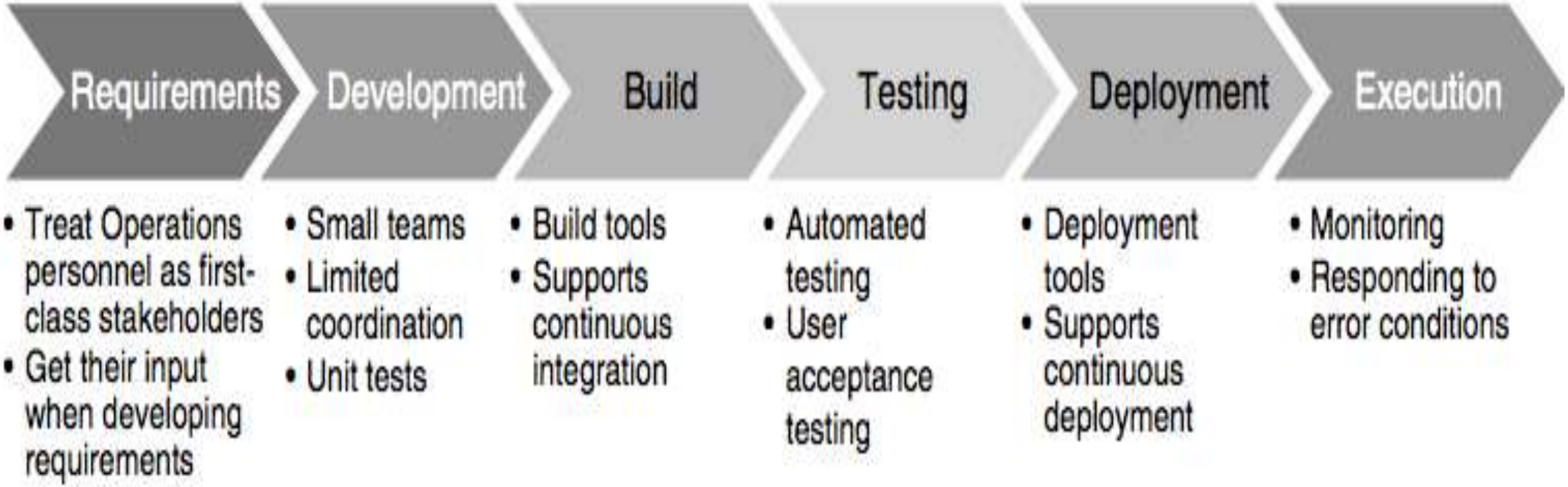
❑**Pros:**

- ✓Enhanced team collaboration
- ✓Reduced human errors through automation
- ✓Continuous feedback and improvement

❑**Cons:**

- ❖Can be challenging to implement across all teams
- ❖Requires investment in tools and training

Devops Practices



Continuous Integration (CI)

- A practice where developers regularly merge their code changes into a central repository, followed by automated builds and tests[3].
- **Usage:** Ensures that code changes are integrated and tested frequently, reducing integration issues.
- **Pros:**
  - ✓Early detection of bugs
  - ✓Faster development cycles
  - ✓Improved code quality

❑**Cons:**

- ❖Requires robust testing infrastructure
- ❖Can be resource-intensive

Continuous Delivery (CD)

- **Definition**: A practice where code changes are automatically built, tested, and prepared for a release to production[3].
- **Usage**: Ensures that software can be reliably released at any time.
- **Pros**:
  - ✓Faster and more reliable releases
  - ✓Reduced deployment risks
  - ✓Improved customer satisfaction
- **Cons**:
  - ❖Requires comprehensive test automation
  - ❖Initial setup can be complex

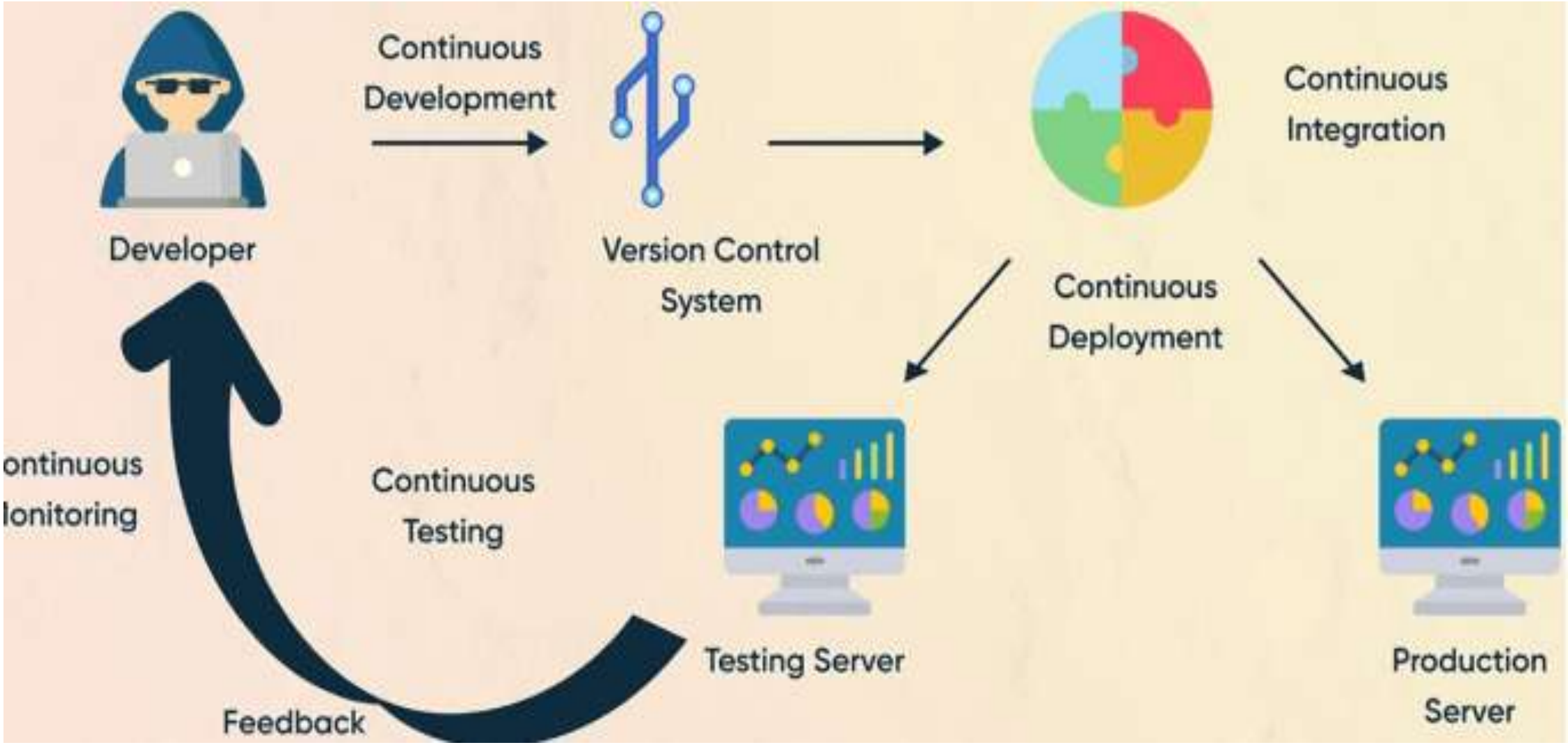
Infrastructure as Code (IaC)

- **Definition**: The practice of managing and provisioning computing infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools[3].
- **Usage**: Automates the setup and management of infrastructure, making it consistent and repeatable.
- **Pros**:
  - ✓Consistency in infrastructure setup
  - ✓Easier to manage and scale
  - ✓Reduces manual errors
- **Cons**:
  - ❖Requires knowledge of scripting and coding
  - ❖Can be complex to implement initially

Monitoring and Logging

- **Definition**: Practices that enable organizations to see how application and infrastructure performance impacts the experience of their product's end user[3]
- **Usage**: Helps in identifying and resolving issues quickly, ensuring system reliability.
- **Pros**:
  - ✓Real-time insights into system performance
  - ✓Faster issue resolution
  - ✓Improved system reliability
- **Cons**:
  - ❖Can generate large volumes of data
  - ❖Requires effective tools and processes to manage

DevOps in Work



References

- How to Use Git and GitHub – a Guide for Beginners and Experienced Developers (freecodecamp.org)