PW SKILLS

Introduction to DevOps & Software Development Fundamentals

Learning Objectives

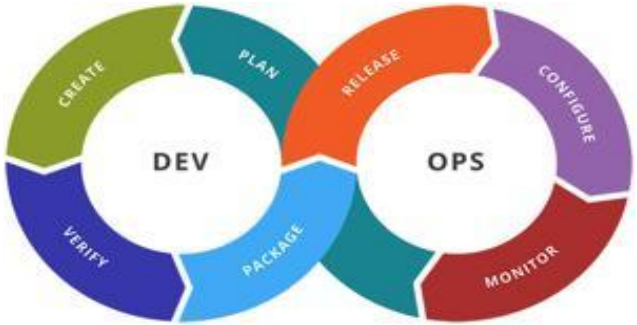
- Define DevOps, its goals, history, and key principles.
- Explain the benefits of DevOps and how it contrasts with traditional development models.
- Understand the Software Development Life Cycle, including Agile methodologies.
- Describe the role of an operating system in managing processes and threads.
- Differentiate between processes and threads and outline the process lifecycle.

PW SKILLS

DevOps & its Purpose in Bridging Development & Operations

Definition of DevOps

- DevOps is a methodology that integrates software development (Dev) and IT operations (Ops) to enhance collaboration, streamline processes, and improve the efficiency of software delivery.
- This approach aims to improve and shorten the systems development lifecycle, fostering a culture of collaboration between traditionally siloed teams.



The DevOps Lifecycle

The DevOps lifecycle (sometimes called the continuous delivery pipeline, when portrayed in a linear fashion) is a series of iterative, automated development processes, or workflows, run within a larger, automated and iterative development lifecycle, designed to optimize the rapid delivery of high-quality software. Workflow names and the number of workflows differ depending on whom you ask, but they often include these seven steps.

DevOps Purpose in Bridging Development & Operations

DevOps serves as a bridge between development and operations by uniting people, processes, and technology across the application lifecycle. Its primary purposes include:

- Collaboration
- Automation
- Continuous Delivery
- Improved Quality
- Faster Time to Market

Pop Quiz

Q. What is a key benefit of adopting DevOps practices?

A

Isolation of development and operations teams

B

Faster delivery of high-quality applications

Pop Quiz

Q. What is a key benefit of adopting DevOps practices?

A

Isolation of development and operations teams

B

Faster delivery of high-quality applications

PW SKILLS

DevOps goals: Faster delivery, Automation & Improved collaboration

Faster Delivery

- DevOps emphasizes rapid software delivery through practices like Continuous Integration (CI) and Continuous Delivery (CD).

- These practices allow teams to release updates more frequently and reliably, thus maintaining a competitive edge. By synchronizing the build-test-deploy cycle, teams can minimize delays caused by traditional handoffs and ensure that software reaches users quickly and efficiently.

Automation

Automation is a fundamental aspect of DevOps that significantly enhances efficiency:

- Automation is a cornerstone of DevOps, streamlining various processes within the software development lifecycle. This includes automating testing, builds, and deployments, which reduces human error and accelerates the release process.

Improved Collaboration

DevOps emphasizes a cultural shift towards collaboration among all stakeholders involved in the software development lifecycle:

- A fundamental goal of DevOps is to foster a culture of collaboration between development and operations teams. By breaking down silos and encouraging cross-functional teamwork, DevOps enhances communication and shared responsibility for project outcomes.

- This collaborative environment not only speeds up problem-solving but also increases transparency in workflows, enabling teams to respond swiftly to issues as they arise.

Pop Quiz

Q. What is one of the primary goals of DevOps?

A

To increase the speed and frequency of software releases.

B

To eliminate all manual processes

Pop Quiz

Q. What is one of the primary goals of DevOps?

A

To increase the speed and frequency of software releases.

B

To eliminate all manual processes

PW SKILLS

Brief History of DevOps & it’s Evolution

Origin & Early influences

- Agile Movement (Early 2000s): The roots of DevOps can be linked to the Agile Software Development movement, which emphasized iterative development and collaboration among teams.

- Agile practices highlighted the need for faster delivery cycles and better communication between developers and operations teams.

The Birth of DevOps

- **Coining of the Term (2009):** The term "DevOps" was first introduced by Patrick Debois during the inaugural DevOpsDays conference in Ghent, Belgium. This event aimed to bridge the gap between development and operations, addressing the dysfunction caused by their separation.

- **Key Presentations:** A pivotal moment occurred at the O'Reilly Velocity Conference in 2009 when John Allspaw and Paul Hammond presented their experiences at Flickr, advocating for enhanced cooperation between Dev and Ops teams. Their talk underscored the necessity of communication in improving deployment frequency and reliability.

Growth & Adoption

- **Early Adoption:** Organizations in the tech sector, particularly those with rapid release cycles like Amazon, were among the first to adopt DevOps practices. They recognized that integrating development and operations could lead to faster feedback loops and improved software quality.

- **Standardization of Practices:** By 2012, the publication of "The Phoenix Project" further popularized DevOps principles, advocating for standardized practices across teams. The annual "State of DevOps" reports began in 2014, documenting the growing adoption and benefits of DevOps across industries.

Current Landscape & Future Trends

Today, DevOps has become a fundamental approach in software development, characterized by:

- **Automation**
- **Collaboration Tools**
- **Cultural Shift**

Pop Quiz

Q. When was the term "DevOps" first introduced?

A

2007

B

2009

Pop Quiz

Q. When was the term "DevOps" first introduced?

A

2007

B

2009

PW SKILLS

DevOps Key Principles

Key Principles

DevOps is a methodology that integrates development and operations teams to improve software delivery and operational efficiency.

Three fundamental principles of DevOps are:

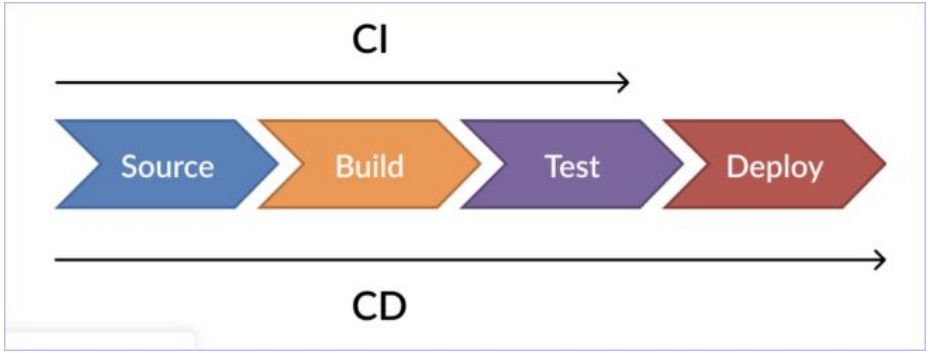
- Continuous Integration & Continuous Deployment (CI/CD).
- Collaboration
- Automation

Each of these principles plays a crucial role in enhancing productivity and ensuring high-quality software releases.

Continuous Integration & Continuous Deployment (CI/CD)

Continuous Integration (CI) is the practice of frequently merging code changes into a shared repository. This process involves:

Continuous Deployment (CD) takes CI a step further by automating the release process. Key aspects include:



Collaboration

Collaboration is at the heart of DevOps, emphasizing the need for seamless communication between development and operations teams. This principle includes:

- DevOps promotes a culture where development and operations teams work together throughout the entire software lifecycle, fostering communication and shared responsibility for outcomes.

Automation

- Automation is a critical principle in DevOps that aims to streamline processes and reduce manual effort.

- Automating repetitive tasks such as testing, deployment, and infrastructure management is crucial for speeding up processes and reducing human error.

Pop Quiz

Q. In which phase are automated tests primarily run in a CI/CD pipeline?

A

Test phase

B

Deploy phase

Pop Quiz

Q. In which phase are automated tests primarily run in a CI/CD pipeline?

A

Test phase

B

Deploy phase

PW SKILLS

DevOps benefits with real-world examples

Benefits with real-world examples

1. Faster Time to Market

Example: Target implemented CI/CD pipelines, which reduced the time to launch new features and updates significantly. This agility allowed them to keep pace with changing consumer preferences and resulted in a 3% increase in online sales due to quicker deployments.

2. Increased Efficiency

Example: Amazon utilizes DevOps practices to automate processes across its vast infrastructure, enabling rapid innovation and service delivery. This automation reduces manual workloads, allowing teams to focus on higher-value tasks

Benefits with real-world examples

3. Improved Software Quality

Example: Etsy adopted automated testing and continuous integration, which enhanced code quality and stability. This led to fewer production issues and quicker resolutions, ultimately boosting customer satisfaction.

4. Cost Reduction

Example: Walmart

Walmart embraced DevOps to optimize its supply chain management, resulting in cost savings through improved data analysis and automated processes. This approach has enhanced operational efficiency while reducing overhead costs.

Benefits with real-world examples

5. Scalability

Example: Netflix

Netflix employs a microservices architecture supported by DevOps practices, enabling it to scale its services efficiently for millions of users worldwide. This scalability is crucial for maintaining performance during peak usage times.

6. Enhanced Customer Satisfaction

Example: Capital One

Capital One implemented DevOps practices that focused on automation and collaboration, resulting in reduced time-to-market for new features. This responsiveness has led to increased customer satisfaction as users benefit from timely updates and improvements.

Pop Quiz

Q. Which benefit of DevOps can lead to enhanced customer satisfaction due to timely updates and improvements?

A

Reduced collaboration between teams

B

Faster deployment of new features and fixes

Pop Quiz

Q. Which benefit of DevOps can lead to enhanced customer satisfaction due to timely updates and improvements?

A

Reduced collaboration between teams

B

Faster deployment of new features and fixes

PW SKILLS

Overview of traditional SDLC models & their limitations

What is Software Development Life Cycle (SDLC)?

The software development lifecycle (SDLC) is the cost-effective and time-efficient process that development teams use to design and build high-quality software. The goal of SDLC is to minimize project risks through forward planning so that software meets customer expectations during production and beyond.

Traditional SDLC Models and Their Limitations

1. Waterfall Model

Description: The Waterfall model is the most traditional and straightforward approach to software development. It follows a linear and sequential process where each phase must be completed before moving on to the next.

Limitations:

- Inflexibility
- Late User Feedback
- High Risk of Failure

Traditional SDLC Models and Their Limitations

2. V-Model

Description: The V-Model extends the Waterfall model by emphasizing verification and validation. Each development phase corresponds directly to a testing phase.

Limitations:

- Rigid Structure
- Assumes Stable Requirements

Traditional SDLC Models and Their Limitations

3. Spiral Model

Description: The Spiral model combines iterative development with systematic risk assessment. It consists of repeated cycles (or spirals) through phases such as planning, risk analysis, engineering, testing, and evaluation.

Limitations:

- Complexity
- High Cost

Traditional SDLC Models and Their Limitations

4. Iterative Model

Description: The Iterative model emphasizes incremental development. Each iteration produces a working version of the software that incorporates user feedback.

Limitations:

- Less Predictable Outcomes
- Documentation Challenges

Pop Quiz

Q. Which model is best suited for projects with well-defined requirements that are unlikely to change?

A

Waterfall Model

B

Spiral Model

Pop Quiz

Q. Which model is best suited for projects with well-defined requirements that are unlikely to change?

A

Waterfall Model

B

Spiral Model

PW SKILLS

Agile Methodologies & Scrum & Kanban frameworks

Agile Methodologies

Overview of Agile Methodologies:

Agile is a project management framework that breaks projects into smaller, manageable phases known as sprints. This iterative approach allows teams to adapt to changes quickly and continuously improve their processes.

Agile Manifesto Values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

These values promote a collaborative environment where teams can respond effectively to changing requirements and deliver value to customers frequently.

Scrum Framework

Scrum is one of the most popular Agile frameworks, particularly suited for projects with rapidly changing requirements.

Roles: Key roles in Scrum include:

- Product Owner
- Scrum Master
- Development Team

Scrum Framework

Ceremonies: Scrum includes several key meetings:

- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

Benefits of Scrum

- Encourages regular feedback from stakeholders.
- Increases transparency and accountability among team members.
- Facilitates continuous improvement through retrospectives.

Kanban Framework

Kanban is another Agile methodology that focuses on visualizing work processes and managing workflow efficiently. It is characterized by:

- Visual Management
- Work In Progress (WIP)
- Limits Continuous Delivery

Benefits of Kanban

- Enhances visibility of work processes and bottlenecks.
- Improves efficiency by limiting WIP and focusing on task completion.
- Offers flexibility in prioritizing tasks without fixed iterations.

Pop Quiz

Q. Which Agile framework focuses on delivering small increments of work frequently?

A

Waterfall

B

Scrum

Pop Quiz

Q. Which Agile framework focuses on delivering small increments of work frequently?

A

Waterfall

B

Scrum

PW SKILLS

Example or Case study to compare Waterfall and Agile outcomes

Case Study : Software Development Project

Project Background: A software development company was tasked with creating a new customer relationship management (CRM) system. The project was initially planned using the Waterfall methodology, but after encountering significant challenges, the team transitioned to Agile.

Waterfall Approach

Process: The project followed a linear sequence: requirements gathering, design, implementation, testing, and deployment.

Outcomes:

- Missed Deadlines
- User Dissatisfaction
- Limited Flexibility

Case Study : Software Development Project

Agile Approach

Process: After recognizing the limitations of Waterfall, the team adopted Agile practices, working in iterative sprints with continuous stakeholder feedback.

Outcomes:

- Improved Collaboration
- Faster Delivery
- Higher Quality Product

Case Study : Software Development Project

Comparison of Outcomes

Aspect	Waterfall Outcomes	Agile Outcomes
Flexibility	Rigid; difficult to implement changes	Highly adaptable; welcomes changes
Stakeholder Involvement	Limited to initial and final phases	Continuous involvement throughout the project
Delivery Time	Longer due to sequential phases	Faster due to iterative sprints
User Satisfaction	Low; final product did not meet expectations	High; regular feedback ensured alignment
Cost Management	Increased costs due to late changes	Better cost control through incremental delivery

PW SKILLS

Emphasizing iterative progress, flexibility, and stakeholder involvement in Agile.

1. Iterative Progress

Definition: Iterative progress refers to the practice of breaking down the development process into small, manageable increments or iterations (often called sprints in Scrum). Each iteration results in a potentially shippable product increment.

Benefits:

- Frequent Deliverables
- Continuous Feedback
- Risk Mitigation

2. Flexibility

Definition: Flexibility in Agile allows teams to adapt to changing requirements and priorities throughout the development process. This adaptability is crucial in today's fast-paced business environment.

Benefits:

- Responsive to Change
- Prioritization of Features
- Enhanced Innovation

3. Stakeholder ~~Involvement~~

Definition: Active involvement of stakeholders such as customers, end-users, and business representatives is a cornerstone of Agile methodologies. Their input is sought throughout the development process.

Benefits:

- Alignment with User Needs
- Increased Satisfaction
- Collaborative Problem Solving

Pop Quiz

Q. Which Agile principle allows teams to adapt to changing requirements even late in development?

A

Continuous integration

B

Iterative development

Pop Quiz

Q. Which Agile principle allows teams to adapt to changing requirements even late in development?

A

Continuous integration

B

Iterative development

PW SKILLS

Take A 5-Minute Break!

- Stretch and relax
- Hydrate
- Clear your mind
- Be back in 5 minutes

PW SKILLS

OS role in managing processes and threads.

OS role in managing processes

Processes Management

Definition and Lifecycle:

A process is defined as a program in execution, which encompasses the program code, current activity, and allocated resources. The OS is responsible for creating, scheduling, and terminating processes. Each process operates within its own memory space, ensuring isolation from others, which enhances security and stability.

Key Responsibilities:

- Creation and Deletion
- Scheduling
- Resource Allocation

OS role in managing Threads

Threads Management

Definition and Characteristics

A thread is a lightweight subprocess that can execute independently while sharing the same address space of its parent process. Multiple threads can exist within a single process, allowing for parallel execution of tasks.

Key Responsibilities:

- Creation and Control
- Scheduling and Prioritization
- Synchronization

Pop Quiz

Q. Which of the following is NOT a valid state of a process in an operating system?

A

Blocked

B

Suspended

Pop Quiz

Q. Which of the following is NOT a valid state of a process in an operating system?

A

Blocked

B

Suspended

PW SKILLS

Processes & Threads with simple examples

Processes

A process is an instance of a computer program that is being executed. It includes the program code, its current activity (represented by the program counter), and the resources allocated to it, such as memory and file handles.

Example of a Process

Consider a web browser application like Google Chrome. When you open Chrome, the operating system creates a process for it. If you open multiple tabs within Chrome, each tab may run as a separate process. This means that if one tab crashes, it does not affect the others because they are isolated in their own processes.

Threads

A thread is a smaller unit of execution within a process. Threads are often referred to as lightweight processes because they require fewer resources to create and manage compared to full processes.

Example of a Thread

Using the same Google Chrome example, within the Chrome process, each tab can have multiple threads handling different tasks simultaneously. For instance, one thread might be responsible for rendering the webpage, another for running JavaScript code, and yet another for handling user input. All these threads share the same resources allocated to the Chrome process but operate concurrently.

Pop Quiz

Q. If a web browser is considered a process, what would each tab opened in that browser represent?

A

Individual threads within the same process

B

Shared resources

Pop Quiz

Q. If a web browser is considered a process, what would each tab opened in that browser represent?

A

Individual threads within the same process

B

Shared resources

PW SKILLS

Differences between Processes & Threads

Processes v/s Threads

PW SKILLS

Processes and threads are fundamental concepts in operating systems, each serving different purposes and characteristics. Here are the key differences between processes and threads, particularly focusing on aspects like resource sharing:

Feature	Processes	Threads
Definition	A process is an independent program in execution with its own memory space.	A thread is a lightweight unit of a process that can execute independently.
Memory Space	Each process has its own separate memory space, ensuring isolation from other processes.	Threads within the same process share the same memory space, allowing for easier communication.
Resource Sharing	Processes do not share resources directly; inter-process communication (IPC) mechanisms are needed for data exchange.	Threads share resources such as data segments and files, facilitating faster communication.
Creation Time	Creating a new process is time-consuming due to the allocation of separate memory space and resources.	Creating a new thread is faster as it does not require a separate memory space.

Processes v/s Threads

Termination	Termination of one process does not affect others; processes operate independently.	If one thread is blocked or terminated, it can affect the execution of other threads within the same process.
Overhead	Processes have higher overhead due to their isolation and resource management requirements.	Threads have lower overhead, making them more efficient for concurrent tasks within a single application.
Context Switching	Context switching between processes is more time-consuming due to the need to switch memory contexts.	Context switching between threads is quicker since they share the same memory space.
Use Cases	Processes are suitable for tasks that require isolated execution environments (e.g., running different applications).	Threads are better for tasks that require frequent communication and shared data (e.g., web browsers handling multiple tabs).

Pop Quiz

Q. What happens to all threads within a process when that process is terminated?

A

All threads within the process are also terminated.

B

Threads continue to run independently.

Pop Quiz

Q. What happens to all threads within a process when that process is terminated?

A

All threads within the process are also terminated.

B

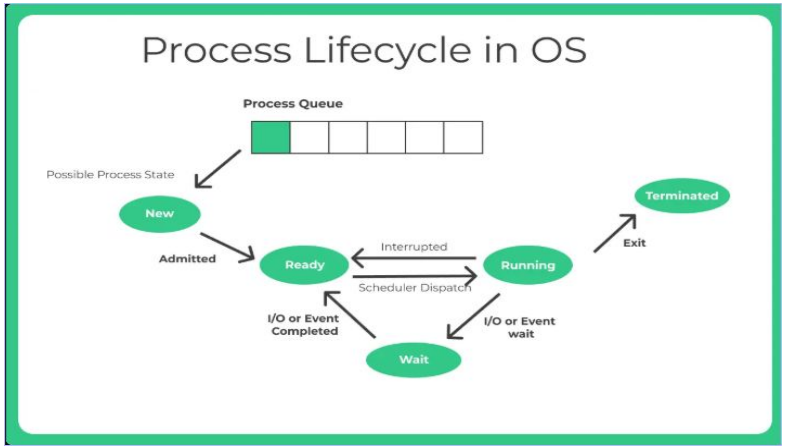
Threads continue to run independently.

PW SKILLS

Walk through the Process Lifecycle

Process Lifecycle

The process lifecycle in an operating system consists of several distinct states that a process transitions through from creation to termination. Understanding these states is crucial for grasping how operating systems manage processes effectively. Here's a detailed walkthrough of the five primary states: New, Ready, Running, Waiting, and Terminated.



Process Lifecycle

1. New State

Definition: This is the initial state of a process when it is created. At this point, the process is being set up and has not yet been loaded into memory.

2. Ready State

Definition: Once the process has been created and is ready to execute, it enters the Ready state. In this state, the process is waiting for CPU allocation.

3. Running State

Definition: When a process is allocated CPU time, it transitions to the Running state, where it actively executes its instructions.

Process Lifecycle

4. Waiting State

Definition: A process enters the Waiting state when it cannot proceed until some condition is met, such as waiting for I/O operations to complete or waiting for resources to become available

5. Terminated State

Definition: Once a process has completed its execution or has been terminated by the OS, it enters the Terminated state.

Pop Quiz

Q. What happens to a process when it requires input/output operations to complete?

A

It enters the Running state.

B

It enters the Waiting state.

Pop Quiz

Q. What happens to a process when it requires input/output operations to complete?

A

It enters the Running state.

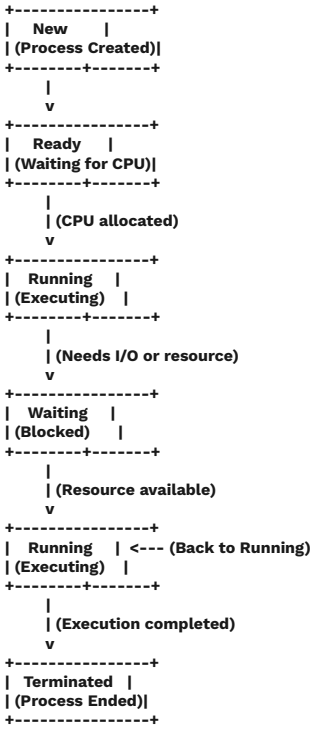
B

It enters the Waiting state.

PW SKILLS

Diagram or flowchart to illustrate the lifecycle visually.

Flowchart to illustrate the life cycle visually



PW SKILLS

Time for case study!

Important

- Complete the post-class assessment
- Complete assignments (if any)
- Practice the concepts and techniques taught in this session
- Review your lecture notes
- Note down questions and queries regarding this session and consult the teaching assistants.

Thanks !