****

**RP-IPRC TUMBA**

**BTech: Information of Technology**

Module: Development Operation

**Group Member**

1. BAGWANEZA Aline foi 23RP01685
2. UMUGIRANEZA Pacifique 23RP01253
3. NZABAHERAHEZA Jean Claude 23RP00463
4. HAGENIMANA Japhet 23RP00375
5. UWIZEYIMANA Isaie 23RP00980
6. MUHIZI Samuel 23RP01417

* **1. DEVOPS OVERVIEW**

**DevOps,** short for Development and Operations, is a set of practices, streamline collaboration between Development and operation Teams, Supporting automation, continuous integration (CI) and Continuous delivery (CD) to enhance software development efficiency, quality and deployment speed.

**Key aspects of DevOps include**:

**Collaboration**: DevOps emphasizes collaboration and communication between development and operations teams. This ensures that both teams work together seamlessly throughout the entire software development lifecycle.

**Automation**: Automation plays a crucial role in DevOps to reduce manual errors, increase efficiency, and accelerate the software delivery process. This includes automating code builds, testing, deployment, and infrastructure provisioning.

**Continuous Integration (CI):** CI involves regularly integrating code changes into a shared repository, where automated builds and tests are performed. This helps identify and address issues early in the development process.

**Continuous Delivery/Continuous Deployment (CD):** Continuous Delivery is the practice of ensuring that code is always in a deployable state, while Continuous Deployment takes this a step further by automatically deploying code changes to production after passing automated tests.

**Infrastructure as Code (IaC):** IaC is the practice of managing and provisioning infrastructure using code and automation tools. This allows for consistent and repeatable infrastructure deployments.

**Monitoring and Logging**: DevOps places a strong emphasis on monitoring application performance and logging relevant data to identify issues, troubleshoot problems, and optimize system performance.

**Microservices**: DevOps often aligns with a microservices architecture, where applications are built as a collection of small, independent services that can be developed, deployed, and scaled independently.

**Containerization**: Containers, such as Docker, are used to package applications and their dependencies into a standardized unit. This ensures consistency across different environments and facilitates easier deployment and scaling.

**Version Control**: Version control systems (e.g., Git) are crucial in DevOps for managing and tracking changes to code, enabling collaboration among team members, and maintaining a history of code changes.

**Feedback Loops**: Continuous feedback is integral to DevOps. Teams use feedback from monitoring, testing, and user experiences to improve and iterate on their processes continually

**DevOps aims:**

* Support collaboration between development and operations teams
* Shorten the software development lifecycle to release a superior product to users faster
* Respond rapidly to changing user needs and market conditions
* Avoid ' Procrastination.'
* Constantly monitor and gather feedback for improvement
* **2. DEVOPS LIFE CYCLE**

DevOps is a continuous cycle, often represented as an infinity symbol, where the 'final' stage connects to the 'first' stage, to emphasize the continuing nature of the process.

DevOps Cycle

1. **Plan:**

Define Objectives: Clearly outline the goals and requirements of the software development and deployment.

Prioritize Tasks: Identify and prioritize tasks based on business needs and project requirements.

Collaboration: Encourage collaboration between development, operations, and other stakeholders to align everyone on the same page.

1. **Code:**

Version Control: Implement version control systems (e.g., Git) to manage and track changes to the source code.

Code Development: Developers write code following best practices, coding standards, and incorporating code review processes.

1. **Build:**

Continuous Integration (CI): Automate the build process to integrate code changes frequently. This ensures early detection of integration issues.

Automated Testing: Conduct automated testing (unit tests, integration tests, and other types) to validate the code's functionality and performance.

1. **Test:**

Continuous Testing: Automate the testing process to identify and fix issues early in the development cycle.

Environment Provisioning: Create and manage testing environments that mirror production environments as closely as possible.

1. **Deploy:**

Continuous Deployment (CD): Automate the deployment process to move code changes seamlessly from the development environment to production.

Containerization: Use containerization tools like Docker to package applications and their dependencies for consistency across different environments.

1. **Release:**

Deploy code to staging or production.

1. **Operate:**

Monitoring and Logging: Implement monitoring tools to track the performance and health of applications in real-time. Utilize logging for troubleshooting and auditing.

Incident Response: Establish processes for responding to incidents and outages promptly. Implement automation for routine operational tasks.

1. **Monitor:**

Performance Monitoring: Continuously monitor application performance, infrastructure, and user experience.

Feedback Loop: Gather feedback from monitoring tools and end-users to inform further improvements and iterations.

**Key Principles of the DevOps Lifecycle:**

**Automation**: Automate repetitive tasks to free up time for creative and strategic work.

**Collaboration**: Foster the breakdown of barriers and cultivate open communication channels between development and operations teams.

**Feedback**: Continuously gather and incorporate feedback to improve the software delivery process.

**Measurement**: Track progress and measure success based on defined metrics.

**Benefits of Adopting the DevOps Lifecycle:**

**Faster software delivery**: Frequent releases and shorter lead times to market.

**Improved software quality**: Early detection and fixing of bugs through continuous testing.

**Increased reliability and performance**: Proactive monitoring and infrastructure management.

**Enhanced developer and operations satisfaction**: Collaborative and automated workflows.

**3. DEVOPS TOOLS- DOCKER**

**Docker** is a key tool in DevOps operation or platform for developers and sysadmins to build, share, and run applications. It streamlines development and opearation by creating portable, isolated containers.

**The Benefits of Using Docker:**

* High ROI and cost savings
* Productivity and standardization
* Maintenance and compatibility
* Rapid deployment
* Faster configurations
* Seamless portability
* Continuous testing and deployment
* Isolation, segregation, and security

**Key benefits of Docker in DevOps:**

**Consistent environments:** Developers can code in a containerized environment identical to production, eliminating the "it works on my machine" problem.

**Simplified deployment:** Applications packaged as Docker containers are easily deployed across different environments with minimal configuration changes.

**Scalability:** Containerized applications scale effortlessly by spinning up new instances quickly and efficiently.

**Agility:** Faster development cycles are enabled by facilitating continuous integration and continuous delivery (CI/CD) pipelines.

**Microservices architecture:** Docker fosters building applications as modular microservices, promoting maintainability and flexibility.

**Resource efficiency:** Containers share the host system's kernel, leading to lower resource consumption compared to virtual machines.

**Improved collaboration:** Developers and operations teams collaborate more effectively with a shared understanding of containerized applications.

**How Docker integrates with the DevOps pipeline**

**Development**: Developers build and test their code within Docker containers, ensuring consistency with production.

**Integration**: Containerized code is easily integrated into CI/CD pipelines for automated testing and deployment.

**Deployment:** Containers are deployed to various environments (staging, production) with minimal configuration changes.

**Monitoring and scaling**: Container orchestration tools (e.g., Kubernetes) manage and scale containerized applications in production.

**Popular DevOps tools that integrate with Docker**

* **Docker Compose**: Defines multi-container applications and their dependencies.
* **Kubernetes**: Orchestrates containerized applications at scale.
* **Jenkins**: Integrates Docker builds and deployments into CI/CD pipelines.
* **Ansible**: Automates infrastructure provisioning and configuration management for Docker deployments.
* **Terraform**: Manages infrastructure as code, provisioning environments for Docker deployments.

To install Docker, you generally need:

1. **Supported Operating System:**

Linux, Windows, or macOS.

**2. Hardware Virtualization:**

Enabled in BIOS/UEFI for Windows and some Linux installations.

**3. Sufficient Resources:**

Adequate CPU, RAM, and disk space as per Docker Desktop requirements.

**4. Internet Connection:**

Needed to download Docker installation packages and images.

**5. Administrative Privileges:**

**Linux:** Sudo or root access for installation.

**Windows:** Administrator rights for installation.

1. **Optional:** Docker account for some features (optional but recommended).

**Docker installation steps:**

Step 1: Downloading Docker. ...

Step 2: Configuration. ...

Step 3: Running the instalation. ...

Step 4: Restart. ...

Step 5: License agreement. ...

Step 6: WSL 2 installation. ...

Step 7 — Starting Docker Desktop. ...

Step 8— Testing Docker.

**Refferences:**

1. Sánchez-Gordón, M., & Colomo-Palacios, R. (2020, June). Security as culture: a systematic literature review of DevSecOps. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (pp. 266-269).

2. Düllmann, T. F., Paule, C., & van Hoorn, A. (2018, May). Exploiting devops practices for dependable and secure continuous delivery pipelines. In *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering* (pp. 27-30).

3. Winkler, F., & Westner, M. (2023, December). A Systematic Literature Review of DevOps Success Factors and Adoption Models. In *Proceedings of the 12th International Symposium on Information and Communication Technology* (pp. 525-532).