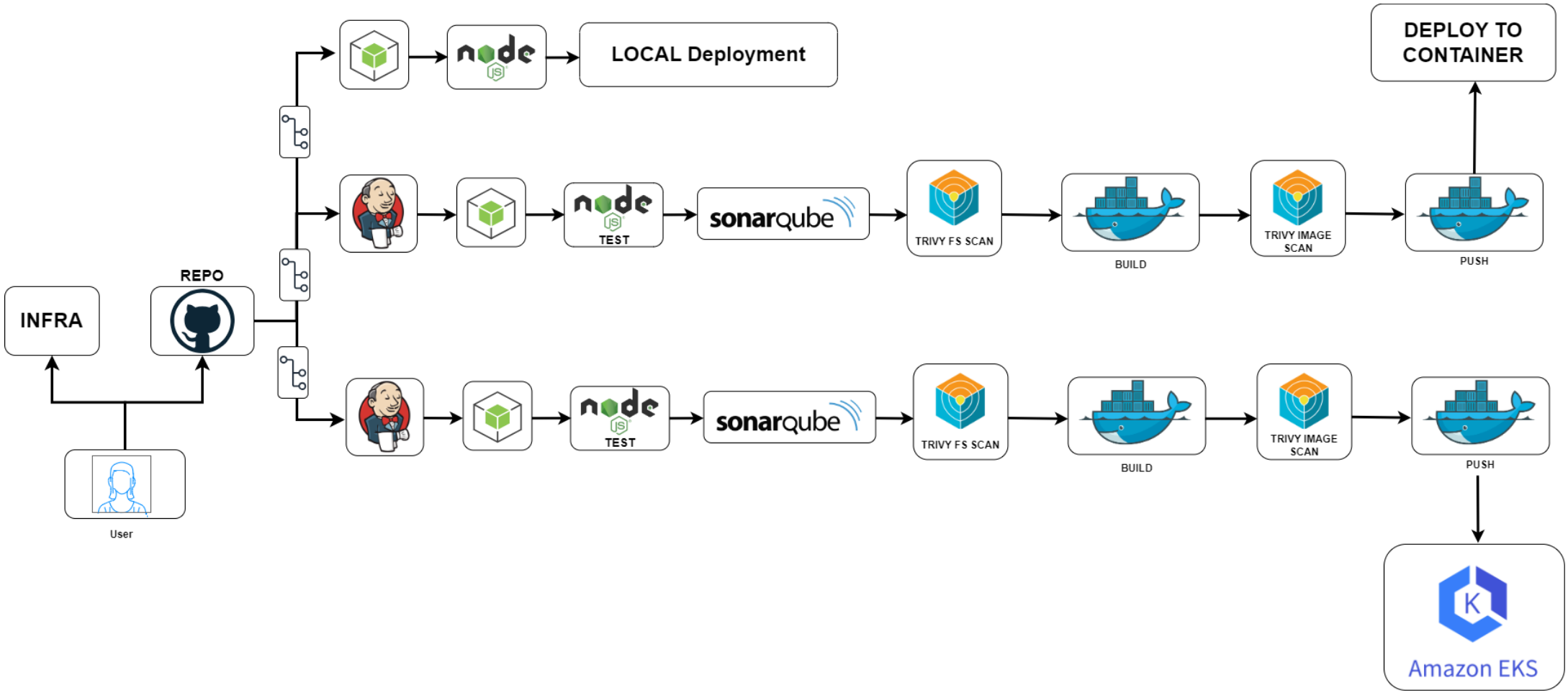
DevOps Shack

Real-Time Corporate DevOps WorkFlow



Introduction

In the dynamic landscape of software development, establishing an efficient workflow is pivotal for ensuring smooth operations and delivering high-quality products. This document delineates a comprehensive real-time workflow employed by companies, elucidating each stage with meticulous detail and accompanied by illustrative diagrams. The workflow centers around a three-tier application utilizing MongoDB as the database technology.

1. Local Development & Testing

Local development and testing form the foundational stage of the workflow. Developers craft source code iteratively and subject it to rigorous testing within their local environments before integration into the main codebase.

Implementation:

1. **Source Code Development:**
    - Developers write and refine code according to project requirements and specifications.
2. **Local Testing:**
    - Execute unit tests to validate individual components.
    - Verify the functionality and integrity of the code within the local environment.
3. **Successful Local Deployment:**
4. Ensure the code operates as expected within the developer's local setup.

2. Development Environment Deployment (Dev)

Description:

Following successful local testing, the code progresses to the development environment (Dev). This stage involves continuous integration and deployment processes orchestrated by DevOps engineers to facilitate efficient collaboration and iterative development.

Implementation:

1. **Dependency Management:**
    - Install and manage project dependencies using package managers like npm or pip.
2. **Dependency Verification:**
    - Validate dependencies to ensure compatibility and version consistency.
3. **Code Testing:**
    - Conduct comprehensive testing, including integration and acceptance tests.
4. **Security Checks:**
    - Utilize tools like Trivy for filesystem scans to identify and address security vulnerabilities.
5. **Docker Image Generation:**
    - Build Docker images encapsulating the application and its dependencies for portability.

6. **Image Vulnerability Scanning:**
    - Scan Docker images for known vulnerabilities using tools like Clair or Aqua Security.
7. **Container Deployment:**
    - Deploy the Docker image to containerized environments within the Dev infrastructure.

3. Production Environment Deployment (Prod)

Description:

Upon successful validation in the Dev environment, the code proceeds to the production environment (Prod), where it undergoes deployment in a scalable and resilient setup.

Implementation:

1. **Dependency Installation:**
    - Ensure all necessary dependencies are installed and configured within the production environment.
2. **Dependency Verification:**

- Validate dependencies to maintain consistency and reliability.

3. **Code Testing:**
    - Execute thorough testing procedures, including load and performance testing, to ensure robustness.
4. **Security Checks:**
    - Conduct filesystem scans and vulnerability assessments to fortify the production environment against potential threats.
5. **Docker Image Generation:**
    - Build Docker images with production-ready configurations and optimizations.
6. **Image Vulnerability Scanning:**
    - Perform comprehensive vulnerability scans on Docker images to mitigate security risks.
7. **Container Orchestration:**
    - Utilize container orchestration platforms like Amazon EKS (Elastic Kubernetes Service) for seamless deployment and management of containers at scale.

Conclusion

Establishing and adhering to a well-defined real-time workflow is indispensable for achieving efficiency, reliability, and scalability in software development endeavors. By meticulously navigating through each stage, companies can streamline their operations, minimize errors, and deliver superior products that meet the evolving needs of their clientele.

Additional Considerations:

- **Continuous Improvement:** Regularly evaluate and refine the workflow to adapt to changing requirements and leverage emerging technologies.
- **Automation:** Embrace automation tools and practices to expedite processes, enhance consistency, and reduce manual intervention.
- **Collaboration and Communication:** Foster a culture of collaboration and open communication among cross-functional teams to promote synergy and innovation.
- **Monitoring and Feedback:** Implement robust monitoring mechanisms to track performance metrics and gather feedback for continuous enhancement and optimization.