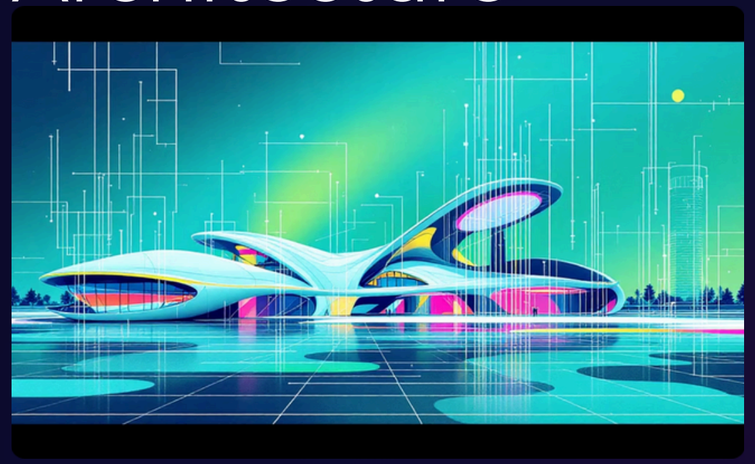
OpenCV Tasks: Architecture & Insights

Kevin Tandon 098 | Malvika Bhadoriya 106 | Manan Bhimjiyani 107 | Manas Jain 109

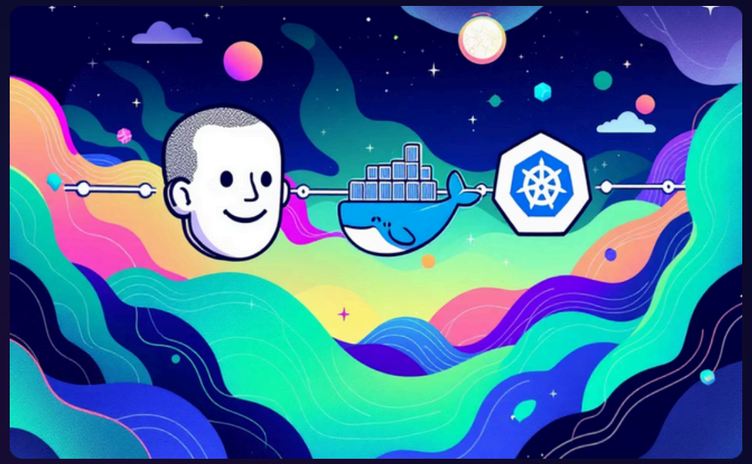
This presentation outlines the architecture, workflow, and key takeaways from a mini-project focused on building a robust and automated DevOps pipeline. We'll explore how Jenkins, Docker, Kubernetes, Prometheus, and Grafana converge to create an efficient CI/CD ecosystem.

Integrated System Architecture



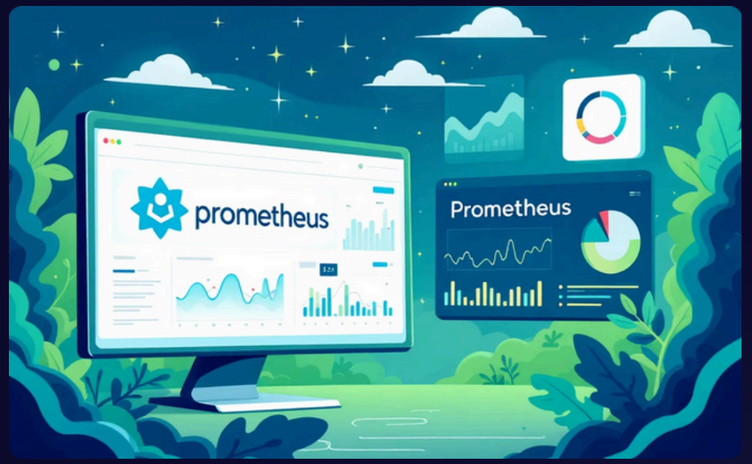
Application Blueprint

The core architecture features a backend and frontend, containerized with Docker, deployed onto a Kubernetes cluster. This setup ensures scalability and resilience for both components.



CI/CD & Orchestration

Jenkins acts as the CI/CD orchestrator, integrating with a Docker Registry for image storage and a Kubernetes cluster for deployment and management of application pods.



Observability Stack

Prometheus provides robust metrics collection, while Grafana offers powerful visualization and alerting capabilities, ensuring comprehensive monitoring of the entire system.

CI/CD Workflow: From Code to Deployment

Our CI/CD pipeline automates the entire software delivery process:

- **Code Commit:** Developers push changes to the version control system.
- **Jenkins Build & Test:** Jenkins automatically triggers builds, runs unit and integration tests.
- **Docker Image Push:** Successfully built images are tagged and pushed to the Docker Registry.

Key Challenges & Solutions

Kubernetes Rolling Updates & Rollbacks

Ensuring zero-downtime updates and seamless rollbacks required careful configuration of Kubernetes deployment strategies and readiness probes.

Ansible for Runtime Environment Setup

Configuring complex runtime environments consistently across different stages using Ansible playbooks proved challenging but critical for reliability.

Docker Image Management

Managing image versions, security scanning, and optimizing image sizes within the Docker Registry added complexity to the pipeline.

Prometheus & Grafana Setup

Initial configuration of Prometheus exporters and Grafana dashboards for relevant metrics and alerts demanded significant effort.

Lessons Learned & Future Insights

- **Container Orchestration Integration:** Seamlessly integrating Kubernetes with CI/CD tools drastically improves deployment speed and reliability.
- **Infrastructure as Code (IaC):** Treating infrastructure configuration as code (e.g., Ansible, Kubernetes manifests) is vital for reproducibility and consistency.
- **Robust Monitoring:** Effective use of Prometheus and Grafana provides invaluable insights into system health and performance, enabling proactive problem-solving.
- **Automation is Key:** Automating repetitive tasks reduces human error and accelerates the development lifecycle.