

DevOps Final Project

Team 6

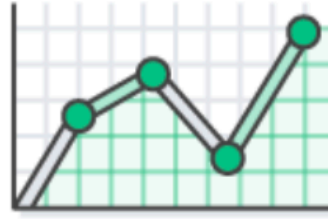
Overview

- What is DevOps ?
- Our Project
- Tools
- Advantages

What is DevOps ?

- DevOps is the combination of "Development" and "Operations" and also cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

Benefits of DevOps



Speed

Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, [microservices](#) and [continuous delivery](#) let teams take ownership of services and then release updates to them quicker.



Rapid Delivery

Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage. [Continuous integration](#) and [continuous delivery](#) are practices that automate the software release process, from build to deploy.



Reliability

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like [continuous integration](#) and [continuous delivery](#) to test that each change is functional and safe. [Monitoring and logging](#) practices help you stay informed of performance in real-time.



Scale

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, [infrastructure as code](#) helps you manage your development, testing, and production environments in a repeatable and more efficient manner.



Improved Collaboration

Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams [collaborate](#) closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).



Security


Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and [policy as code](#), you can define and then track compliance at scale.


Our Project

A DevOps project with robust support for CI/CD, infrastructure automation, monitoring, and DevSecOps for seamless updates and efficient operations across environments, and its key technologies:

Python & Flask: To build the Library web application which we will be the use case for our pipeline.  

Docker: To package the app into containers that run anywhere. 

Kubernetes: To automate deploying, scaling, and managing the app. 
kubernetes

AWS (Amazon Web Services): For hosting everything in the cloud 

Ansible: Automation tool used to streamline and standardize server setups, manage configurations, and deploy the application in an efficient, repeatable manner. 

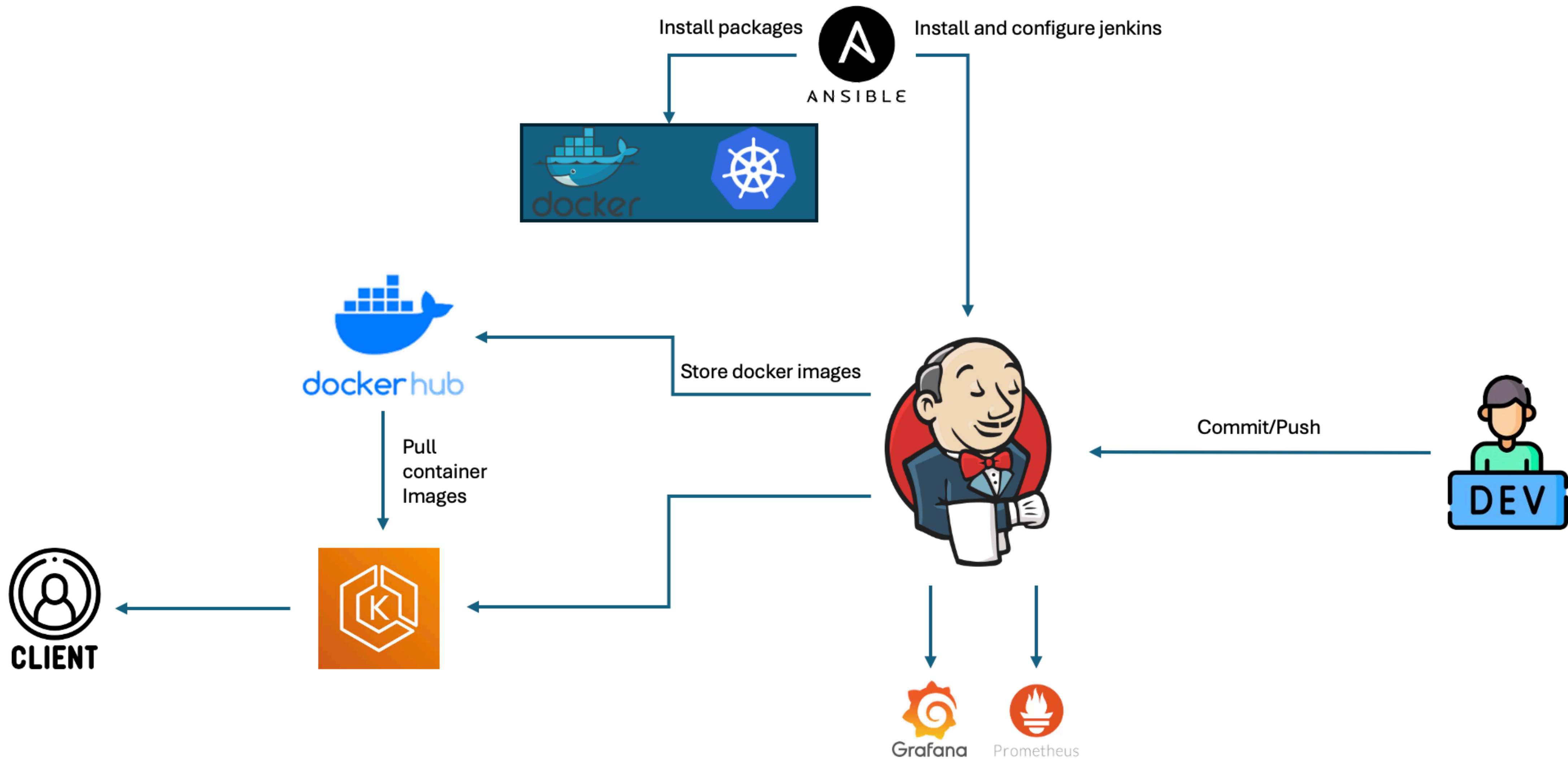
Jenkins: To automate building, testing, and deploying the app (CI/CD). 

Terraform: To automatically set up the cloud infrastructure. 

Prometheus & Grafana: To monitor the system and alert us if something goes wrong.



General Architecture





The Role of Docker & Kubernetes

What is Docker?

Docker is a platform that allows developers to package applications into containers, which are lightweight, portable, and run the same regardless of the environment (local, testing, or production). These containers encapsulate the application along with its dependencies, ensuring consistency.

Kubernetes Simplified:

Think of Kubernetes as a “manager” for the app. It automatically makes sure the app is:

- **Running smoothly** (it fixes issues automatically).
- **Scaled up or down** based on the number of users.
- **Easily accessible** from anywhere, using cloud services.



Terraform on aws

Role in the Project:

- **Automated Cloud Infrastructure:**

We used Terraform to automate the creation of AWS resources for our project, such as:

- **EC2 Instance:** For hosting Jenkins.
- **EKS Cluster:** To run our containerized application on Kubernetes.
- **VPC Setup:** Configured a Virtual Private Cloud with subnets and NAT gateways for secure networking.

Why Terraform?

- **Consistency:** Ensured the same infrastructure across all environments.
- **Efficiency:** Deployed all AWS resources in a single command.
- **Scalability:** Easily scaled resources based on project needs.



The Role of Ansible in the Project

What is Ansible?

Ansible is an open-source automation tool used to configure servers, deploy applications, and manage infrastructure. It helps in automating tasks that would otherwise be manual and repetitive.

How We Used Ansible in the Project:

- **Automating Server Setup:** Ansible was used to automate the configuration of our servers, ensuring that every server had the same software, security settings, and permissions.
- **Consistent Environment:** With Ansible, we ensured that every time a new server was created, it was set up in exactly the same way as the others.
- **Efficient Deployment:** Instead of configuring each server manually, we used Ansible to quickly and reliably set up multiple servers at once.

DevOps Pipeline Explained

CI/CD Pipeline:

Imagine a factory line where the raw material (the code) is constantly built, tested, and then shipped.

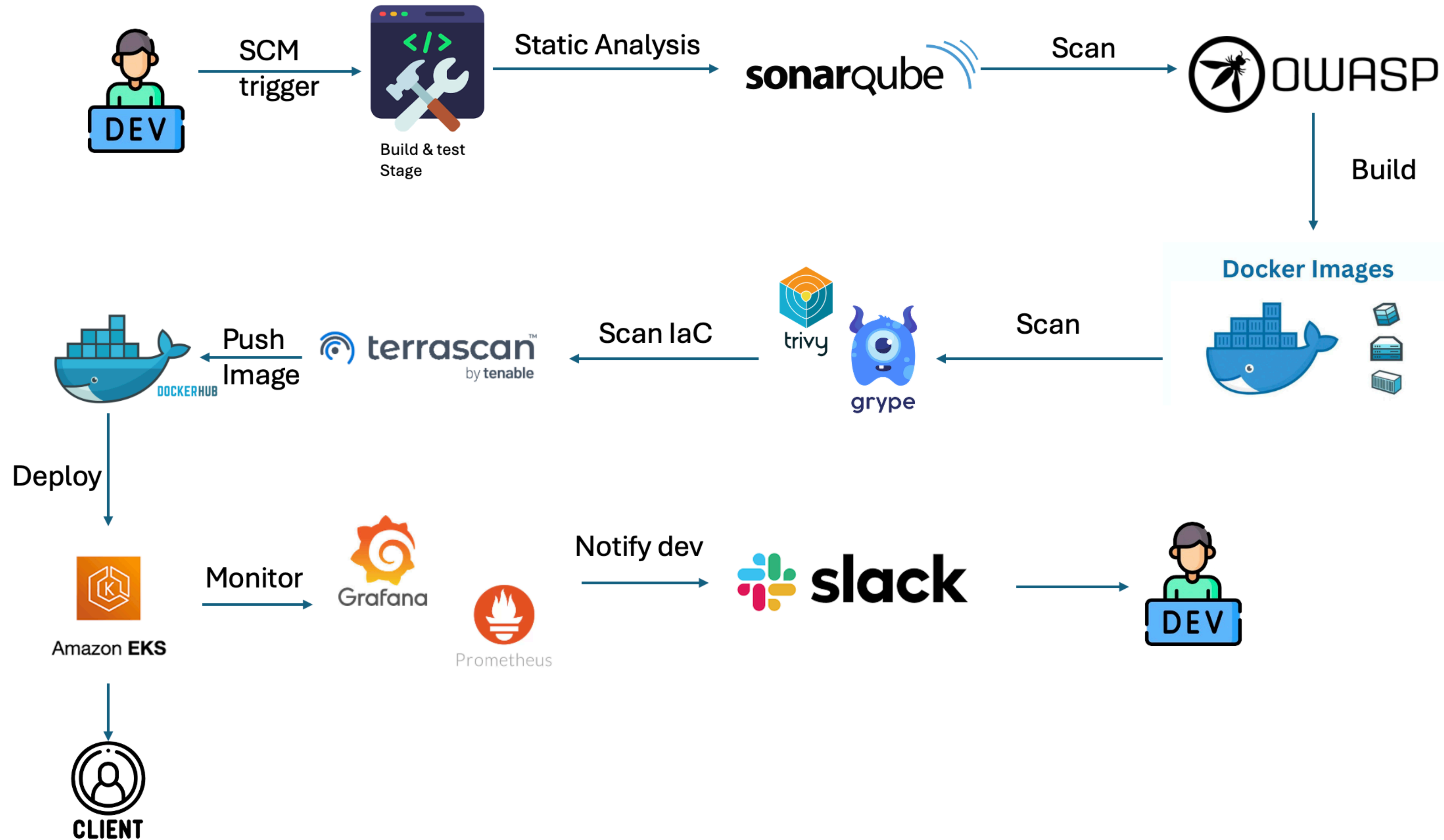
CI (Continuous Integration): The code is automatically tested as soon as it's written, ensuring it's ready for production.

CD (Continuous Deployment): After passing the tests, the system automatically deploys the app, making it live for users.

This means:

- 1. Less manual work and fewer errors.**
- 2. Faster updates and improvements to the app.**

Jenkins Pipeline Architecture



Monitoring and Alerts

Why Monitoring Matters:

Once the system is running, we need to know if everything is working correctly.

Prometheus & Grafana:

- **Prometheus** monitors the app and collects data.
- **Grafana** shows this data in the form of charts and graphs.
- If something goes wrong (e.g., too many users, server down), **Prometheus sends alerts** so we can fix it quickly.



Project Outcomes

Key Benefits for Non-Engineers

Why This is Important for Non-Technical Teams:

- **Speed:** Features can be added or fixed faster, meaning a better experience for users.
- **Automation:** The deployment and updates happen automatically, reducing errors and downtime.
- **Cost-Effective:** Automation reduces the need for manual work and expensive human errors.
- **Security:** The system is constantly checked for vulnerabilities, ensuring users' data is safe.
- **Scalability:** As the library grows, the system grows with it, without needing major rework.
- **Real-Time Monitoring:** We know the system's status at all times, ensuring reliability.

Thank You!