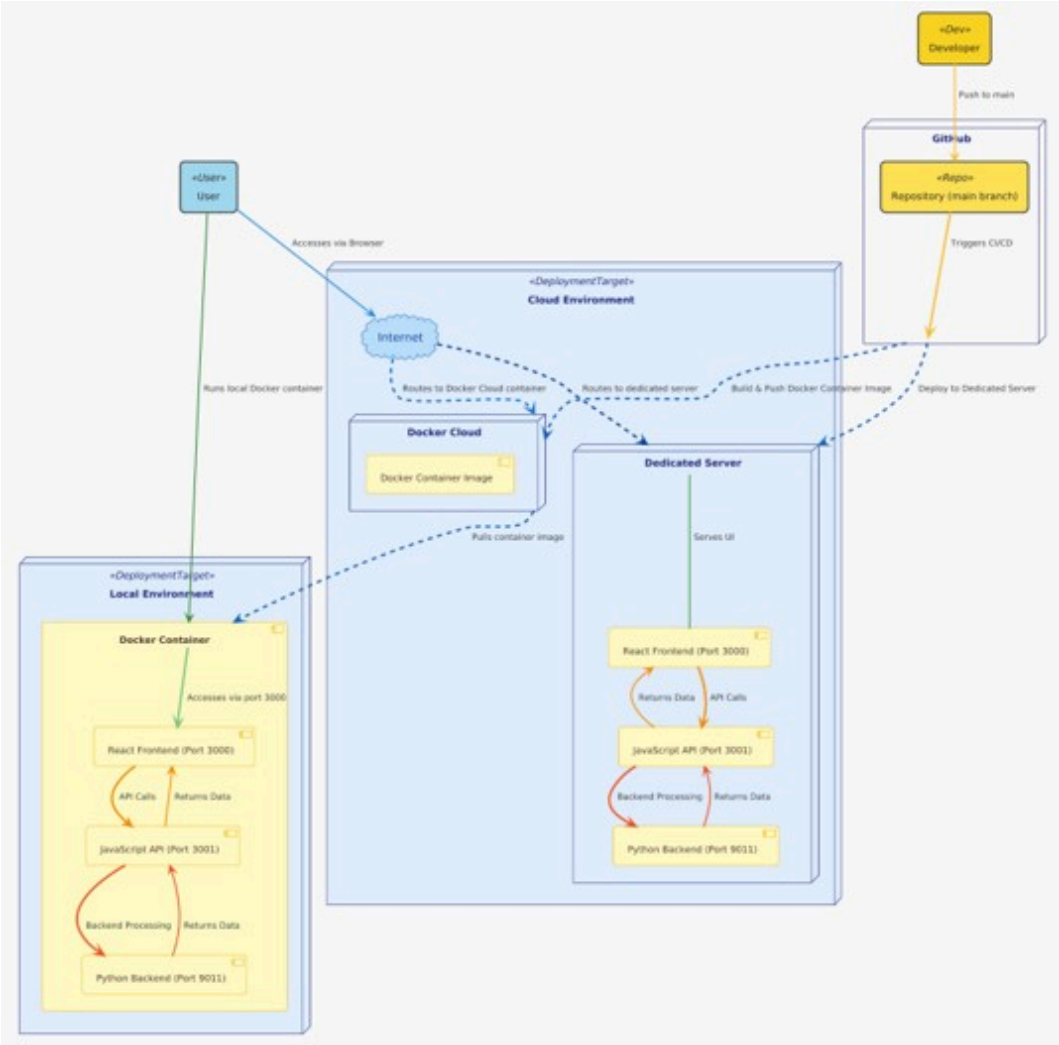
Deployment Overview

Deployment Overview

The API Threat Assessment Tool (AT-AT) system is deployed using a **multi-environment container-based strategy**. It is designed to support both local development and production-grade cloud environments. The deployment process involves pushing Docker container images to cloud-based infrastructure via GitHub CI/CD pipelines.

Deployment Diagram



Environments

Local Development

- Developer runs the **frontend**, **API layer**, and **backend processor** via local Docker containers.
- Services are exposed on:
  - React Frontend: Port `3000`
  - JavaScript API: Port `3001`
  - Python Backend: Port `9011`
- Data is exchanged using HTTP and JSON between these services.

Cloud Deployment

- Docker image is built and pushed to a **container registry** (e.g., DockerHub or GitHub Container Registry) via GitHub CI/CD.
- Containers are deployed to a **dedicated server** within a **cloud environment**.
- Components run on similar ports as in local setup and are exposed through **secure routes**.
- Environment variables (via `.env`) configure service interconnections.

GitHub & CI/CD Flow

- Developer pushes to the `main` branch on GitHub.
- GitHub Actions trigger a CI/CD workflow:
  - Linting and tests are performed.
  - Backend and API Docker images are built and published.
  - The cloud server pulls the latest image and redeploys the service.

Deployment Targets

- `<<device>>` User: Accesses the platform via web browser.
- `<<deploymentTarget>>` Local Docker container: For development and testing.
- `<<deploymentTarget>>` Cloud server: Hosts production services.
- `<<repo>>` GitHub Repository: Central codebase that triggers CI/CD.

Stereotype Labels

- `<<deploymentTarget>>` Cloud Environment / Local Environment
- `<<artifact>>` Docker Container Image

- `<<component>>` React Frontend, JS API Layer, Python Backend
- `<<device>>` User's browser/device

> **Note:** Refer to the `install.md` for detailed steps on launching each environment.