

Project Architecture & Development Guide

[Jump to bottom](#)

Zohreh Asadi edited this page yesterday · [3 revisions](#)

This page explains build/deploy, package managers, CI, Docker images, and service wiring.

Package Managers

Frontend

- **Node 20 + npm**
- Commands:
 - `npm install / npm ci`
 - `npm run build`
- Used for: Vite + React tooling

Backend

- **Python 3.11 with uv**
 - `uv python install 3.11`
 - `uv venv`
 - `uv pip install -r requirements*.txt`
- Used for: fast Python env creation and reproducible builds (same as CI)

Makefile Overview

The Makefile provides a single unified interface for all development tasks. CI workflows also call these same targets.

Common Targets

- **Install**

- `make install`
- `make install-frontend`
- `make install-backend`

- **Lint**

- `make lint`
- `make lint-frontend`
- `make lint-backend`
- `make lint-licensing`

- **Test**

- `make test`

- **Local Runs**

- `make run-frontend-local`
- `make run-webcam-local`
- `make run-analyzer-local`

- **Docker**

- `make docker-build[-frontend|-backend]`
- `make docker-run-*`
- `make docker-clean`

Run `make help` to see all targets.

Continuous Integration (GitHub Actions)

File: `.github/workflows/ci.yml` Triggers: push and pull requests on `main`.

Pipeline Stages

Lint

- REUSE license compliance
- Frontend: `npm run lint`
- Backend (uv): `ruff` + `mypy`

Build

- Frontend: `npm run build`
- Backend: import check

```
uv run python -c "import webcam.main"  
uv run python -c "import analyzer.main"
```



Test

- Frontend: `npm test`
- Backend: `pytest`

Package

- Docker images built via:
 - `make docker-build-frontend`
 - `make docker-build-backend`

Caching

- npm cache keyed by `package-lock.json`
- Python installed fresh with uv (fast)

Docker Images

Frontend

- Build: `src/frontend/Dockerfile`
- Image: `robot-frontend:latest`
- Serves static build on **port 80**

Webcam Service

- Build: `src/backend/Dockerfile.webcam`

- Image: `robot-webcam:latest`
- FastAPI on **8000**

Analyzer Service

- Build: `src/backend/Dockerfile.analyzer`
- Image: `robot-analyzer:latest`
- FastAPI on **8001**
- Needs: `WEBCAM_OFFER_URL` pointing to the webcam service

Useful Commands

- `make docker-build`
- `make docker-run-webcam`
- `make docker-run-analyzer`
- `make docker-run-frontend`
- `make docker-clean`

Service Architecture

Webcam Service (8000)

- Accepts WebRTC offers from the frontend
- Streams raw camera frames
- Uses STUN from `config.STUN_SERVER` (Google STUN by default)

Analyzer Service (8001)

- Accepts WebRTC offers from the frontend
- Creates its own WebRTC session to webcam:
 - `WEBCAM_OFFER_URL` (default: `http://localhost:8000/offer`)
- Runs detection + overlay
- Has metadata data channel (placeholder)

Frontend

- Uses `VITE_BACKEND_URL` → analyzer `/offer`
- Uses `useWebRTCPlayer` for fast connection setup

Local Development

Install

```
make install
```



Lint

```
make lint
```



Test

```
make test
```



Run Services (3 terminals)

Webcam

```
make run-webcam-local
```



Analyzer

```
make run-analyzer-local
```



Frontend

```
VITE_BACKEND_URL=http://localhost:8001 make run-frontend-local
```



Docker Local Run

```
make docker-build  
make docker-run-webcam  
make docker-run-analyzer  
make docker-run-frontend
```



Ports:

- Frontend: **8080** → **80**
- Webcam: **8000**
- Analyzer: **8001**

Analyzer uses:

```
WEBCAM_OFFER_URL=http://host.docker.internal:8000/offer
```



Deployment Notes

Hardware

- If using Torch/ONNX with GPU: install correct CUDA/ROCm drivers.

Environment Variables

- `VITE_BACKEND_URL`
- `WEBCAM_OFFER_URL`
- `DETECTOR_BACKEND`
- `TORCH_DEVICE`
- `ONNX_PROVIDERS`

Networking

- Expose/route:
 - Frontend: 8080 → 80
 - Webcam: 8000
 - Analyzer: 8001

HTTPS

Terminate TLS at the proxy/ingress; backend services run plain HTTP.

▼ **Pages** **3**

Find a page...

▶ **Home**

▼ **Project Architecture & Development Guide**

This page explains build/deploy, package managers, CI, Docker images, and service wiring.

Package Managers

- Frontend

- Backend

Makefile Overview

- Common Targets

Continuous Integration (GitHub Actions)

- Pipeline Stages

 - Lint

 - Build

 - Test

 - Package

 - Caching

Docker Images

- Frontend

- Webcam Service

- Analyzer Service

- Useful Commands

Service Architecture

- Webcam Service (8000)

- Analyzer Service (8001)

- Frontend

Local Development

- Install

- Lint

- Test

- Run Services (3 terminals)

Docker Local Run

Deployment Notes

- Hardware

- Environment Variables

- Networking

- HTTPS

► [Technical Documentation](#)

Clone this wiki locally

<https://github.com/amosproj/amos2025ws04-robot-visual-perception.wiki.git>

