Gaara AI System

AI/ML Docker Containers

Technical Documentation

Version: 4.3

Date: 2025-12-08

Table of Contents

1. Plant Hybridization Container

Port:	8022	**Base Image:**	python:3.11-slim
Purpose:	Advanced plant hybridization simulations using genetic algorithms and ML models for optimal crop breeding.		

Dockerfile:

```
# Plant Hybridization Container
FROM python:3.11-slim
RUN apt-get update && apt-get install -y \
    gcc g++ curl git gfortran \
    liblapack-dev libblas-dev \
    && rm -rf /var/lib/apt/lists/*
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/ ./src/
COPY config/ ./config/
COPY data/ ./data/
RUN mkdir -p /app/data/{varieties,traits,objectives,simulations,results,logs}
ENV PYTHONPATH=/app
ENV HYBRIDIZATION_PORT=8022
ENV SIMULATION_WORKERS=4
ENV MAX_GENERATIONS=100
EXPOSE 8022
HEALTHCHECK --interval=30s --timeout=15s --start-period=60s --retries=3 \
    CMD curl -f http://localhost:8022/health || exit 1
CMD ["python", "src/plant_hybridization_service.py"]
```

Key Features:

* Genetic algorithm-based hybridization simulation
* Multi-objective optimization for crop traits
* Support for up to 100 generations of breeding
* Parallel simulation with 4 workers
* Integration with plant variety database

2. Disease Diagnosis Container

Port:	8001	**Base Image:**	python:3.11-slim
Purpose:	AI-powered plant disease diagnosis using ResNet-50 and custom trained models for accurate disease identification.		

Dockerfile:

```
# Disease Diagnosis Container
FROM python:3.11-slim as diagnosis-base
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
ENV PYTHONPATH=/app
RUN apt-get update && apt-get install -y \
    libgl1-mesa-glx libglib2.0-0 libsm6 \
    libxext6 libxrender-dev libgomp1 curl \
    && rm -rf /var/lib/apt/lists/*
RUN groupadd -r diagnosis && useradd -r -g diagnosis diagnosis
RUN mkdir -p /app /app/models /app/data /app/logs
WORKDIR /app
COPY docker/diagnosis/requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/modules/disease_diagnosis/ ./diagnosis/
COPY src/modules/plant_disease/ ./plant_disease/
COPY src/advanced_ai_system.py ./
USER diagnosis
EXPOSE 8001
HEALTHCHECK --interval=30s --timeout=10s --retries=3 \
    CMD curl -f http://localhost:8001/health || exit 1
CMD ["uvicorn", "diagnosis.api:app", "--host", "0.0.0.0", "--port", "8001"]
```

AI Models Used:

* ResNet-50 (Pre-trained on ImageNet)
* Custom plant disease classifier
* Transfer learning for specific crops
* Confidence scoring system

3. YOLO Detection Container

Port:	8018	**Base Image:**	ultralytics/ultralytics:latest
Purpose:	Real-time object detection for plants, diseases, and pests using YOLOv8 models with GPU acceleration.		

Dockerfile:

```
# YOLO Detection Container
FROM ultralytics/ultralytics:latest
RUN apt-get update && apt-get install -y \
    gcc g++ curl git \
    && rm -rf /var/lib/apt/lists/*
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/ ./src/
COPY config/ ./config/
COPY models/ ./models/
RUN mkdir -p /app/data/{input,output,models,weights,results,logs}
# Download pre-trained YOLO models
RUN python -c "from ultralytics import YOLO; \
    YOLO('yolov8n.pt'); YOLO('yolov8s.pt'); YOLO('yolov8m.pt')"
ENV PYTHONPATH=/app
ENV YOLO_PORT=8018
ENV MODEL_PATH=/app/models
ENV WEIGHTS_PATH=/app/data/weights
EXPOSE 8018
HEALTHCHECK --interval=30s --timeout=15s --retries=3 \
    CMD curl -f http://localhost:8018/health || exit 1
CMD ["python", "src/yolo_detection_service.py"]
```

YOLO Models Included:

* YOLOv8n (Nano) - Fast inference, lower accuracy
* YOLOv8s (Small) - Balanced speed/accuracy
* YOLOv8m (Medium) - Higher accuracy
* Custom trained models for plant diseases

4. Plant Disease Advanced Container

Port:	8021	**Base Image:**	python:3.11-slim
Purpose:	Advanced disease detection with knowledge base integration for symptoms, causes, and treatment recommendations.		

Dockerfile:

```
# Plant Disease Advanced Container
FROM python:3.11-slim
RUN apt-get update && apt-get install -y \
    gcc g++ curl git \
    libopencv-dev python3-opencv \
    && rm -rf /var/lib/apt/lists/*
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/ ./src/
COPY config/ ./config/
COPY knowledge_base/ ./knowledge_base/
RUN mkdir -p /app/data/{diseases,symptoms,treatments,models,results,logs}
ENV PYTHONPATH=/app
ENV DISEASE_PORT=8021
ENV KNOWLEDGE_BASE_PATH=/app/knowledge_base
ENV MODELS_PATH=/app/data/models
EXPOSE 8021
HEALTHCHECK --interval=30s --timeout=15s --retries=3 \
    CMD curl -f http://localhost:8021/health || exit 1
CMD ["python", "src/plant_disease_advanced_service.py"]
```

Knowledge Base Features:

* Disease symptom database
* Treatment recommendations
* Prevention guidelines
* Multi-language support

5. ResNet-50 Container

Port:	8003	**Base Image:**	python:3.11-slim (multi-stage)
Purpose:	Computer vision service using ResNet-50 for image classification and feature extraction.		

Dockerfile:

```
# ResNet-50 Container (Multi-stage build)
FROM python:3.11-slim as builder
RUN apt-get update && apt-get install -y \
    build-essential cmake libopencv-dev \
    libgl1-mesa-glx wget \
    && rm -rf /var/lib/apt/lists/*
RUN python -m venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
# Download pre-trained ResNet-50
RUN python -c "import torch; import torchvision.models as models; \
    model = models.resnet50(pretrained=True); \
    torch.save(model.state_dict(), '/opt/venv/resnet50_pretrained.pth')"
FROM python:3.11-slim as runtime
COPY --from=builder /opt/venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"
WORKDIR /app
COPY src/ ./src/
COPY config/ ./config/
RUN useradd --create-home resnet_user
USER resnet_user
ENV MODEL_PATH=/opt/venv/resnet50_pretrained.pth
ENV OMP_NUM_THREADS=4
EXPOSE 8003
CMD ["uvicorn", "src.resnet50_service:app", "--host", "0.0.0.0", "--port", "8003"]
```

6. GPU Processing Container

Port:	8020	**Base Image:**	nvidia/cuda:11.8-devel-ubuntu20.04
Purpose:	GPU-accelerated processing for AI model inference and training with CUDA support.		

Dockerfile:

```
# GPU Processing Container
FROM nvidia/cuda:11.8-devel-ubuntu20.04
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install -y \
    python3.11 python3.11-dev python3-pip \
    gcc g++ curl git \
    libcudnn8 libcudnn8-dev \
    && rm -rf /var/lib/apt/lists/*
RUN ln -s /usr/bin/python3.11 /usr/bin/python
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/ ./src/
COPY config/ ./config/
RUN mkdir -p /app/data/{gpu_cache,models,processing,results,logs}
ENV PYTHONPATH=/app
ENV GPU_PORT=8020
ENV CUDA_VISIBLE_DEVICES=0
ENV NVIDIA_VISIBLE_DEVICES=all
ENV NVIDIA_DRIVER_CAPABILITIES=compute,utility
EXPOSE 8020
CMD ["python", "src/gpu_processing_service.py"]
```

GPU Capabilities:

* CUDA 11.8 support
* cuDNN 8 for deep learning
* Multi-GPU support
* Batch processing optimization

7. Image Enhancement Container

Port:	8019	**Base Image:**	python:3.11-slim
Purpose:	AI-powered image enhancement for better disease detection including noise reduction, contrast adjustment, and super-resolution.		

Dockerfile:

```
# Image Enhancement Container
FROM python:3.11-slim
RUN apt-get update && apt-get install -y \
    gcc g++ curl \
    libopencv-dev python3-opencv \
    libglib2.0-0 libsm6 libxext6 libxrender-dev libgomp1 \
    && rm -rf /var/lib/apt/lists/*
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/ ./src/
COPY config/ ./config/
RUN mkdir -p /app/data/{input,output,processed,enhanced,filters,logs}
ENV PYTHONPATH=/app
ENV ENHANCEMENT_PORT=8019
ENV MAX_IMAGE_SIZE=4096
ENV QUALITY_LEVEL=95
EXPOSE 8019
HEALTHCHECK --interval=30s --timeout=10s --retries=3 \
    CMD curl -f http://localhost:8019/health || exit 1
CMD ["python", "src/image_enhancement_service.py"]
```

8. Adaptive Learning Container

Port:	8017	**Base Image:**	python:3.11-slim
Purpose:	Self-improving AI system that learns from new data and adapts models based on feedback.		

Dockerfile:

```
# Adaptive Learning Container
FROM python:3.11-slim
RUN apt-get update && apt-get install -y \
    gcc g++ curl \
    && rm -rf /var/lib/apt/lists/*
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/ ./src/
COPY config/ ./config/
RUN mkdir -p /app/data/{models,training,adaptation,logs}
ENV PYTHONPATH=/app
ENV ADAPTIVE_PORT=8017
ENV LEARNING_RATE=0.001
ENV ADAPTATION_THRESHOLD=0.85
EXPOSE 8017
HEALTHCHECK --interval=30s --timeout=15s --start-period=90s --retries=3 \
    CMD curl -f http://localhost:8017/health || exit 1
CMD ["python", "src/adaptive_learning_service.py"]
```

9. Analytics Container

Port:	8002	**Base Image:**	python:3.11-slim
Purpose:	Big data analytics and predictions for agricultural data including yield forecasting and trend analysis.		

Dockerfile:

```
# Analytics Container
FROM python:3.11-slim as analytics-base
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
ENV PYTHONPATH=/app
RUN apt-get update && apt-get install -y \
    gcc g++ libgomp1 curl \
    && rm -rf /var/lib/apt/lists/*
RUN groupadd -r analytics && useradd -r -g analytics analytics
RUN mkdir -p /app /app/data /app/reports /app/logs
WORKDIR /app
COPY docker/analytics/requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src/advanced_analytics_system.py ./
COPY src/big_data_analytics_system.py ./
COPY src/modules/analytics/ ./analytics/
USER analytics
EXPOSE 8002
HEALTHCHECK --interval=30s --timeout=10s --retries=3 \
    CMD curl -f http://localhost:8002/health || exit 1
CMD ["uvicorn", "analytics.api:app", "--host", "0.0.0.0", "--port", "8002"]
```

Summary: AI/ML Container Architecture

```
+-----------------------------------------------------------+
|                    GAARA AI SYSTEM                         |
+-----------------------------------------------------------+
|                                                           |
|  +----------------+    +------------------+                |
|  | Plant Disease  |    | YOLO Detection   |                |
|  | Diagnosis      |    | (Port 8018)      |                |
|  | (Port 8001)    |    +------------------+                |
|  +----------------+                                        |
|                                                           |
|  +----------------+    +------------------+                |
|  | ResNet-50      |    | GPU Processing   |                |
|  | (Port 8003)    |    | (Port 8020)      |                |
|  +----------------+    +------------------+                |
|                                                           |
|  +----------------+    +------------------+                |
|  | Plant          |    | Plant Disease    |                |
|  | Hybridization  |    | Advanced         |                |
|  | (Port 8022)    |    | (Port 8021)      |                |
|  +----------------+    +------------------+                |
|                                                           |
|  +----------------+    +------------------+                |
|  | Adaptive       |    | Image            |                |
|  | Learning       |    | Enhancement      |                |
|  | (Port 8017)    |    | (Port 8019)      |                |
|  +----------------+    +------------------+                |
|                                                           |
|  +----------------+    +------------------+                |
|  | Analytics      |    | Memory Central   |                |
|  | (Port 8002)    |    | (Port 8014)      |                |
|  +----------------+    +------------------+                |
|                                                           |
+-----------------------------------------------------------+
```

Port Summary:

Port	Service	Technologies
8001	Disease Diagnosis	PyTorch, ResNet-50
8002	Analytics	Pandas, NumPy, Scikit-learn
8003	ResNet-50	PyTorch, TorchVision
8017	Adaptive Learning	TensorFlow, Custom ML
8018	YOLO Detection	Ultralytics YOLOv8
8019	Image Enhancement	OpenCV, PIL
8020	GPU Processing	CUDA, cuDNN
8021	Plant Disease Advanced	OpenCV, ML Models
8022	Plant Hybridization	Genetic Algorithms, NumPy