

# DA5402: A2: The Multi-Modal Orchestrated Cluster

**Team Structure:** Teams of 2 (Self-selected)

**Goal:** Develop, containerize, and deploy a multi-modal AI REST API to a distributed cluster.

---

## Part 1: Collaborative AI Microservice Development

Teams must create a single **FastAPI** or **Flask** service. Each student is responsible for two features. Work must be done in separate Git branches and merged via Pull Requests.

### Task Allocation

- **Developer A:**
  - **Translation:** Endpoint using Google Translate API (or equivalent).
  - **Image Generation:** Text-to-image endpoint using a free-tier API (e.g., [fal.ai](#), [SiliconFlow](#), or Stability AI).
- **Developer B:**
  - **NER:** Entity extraction using a library like spaCy or a HuggingFace transformer.
  - **Speech Synthesis:** Text-to-speech endpoint (e.g., Google Cloud TTS or a free HuggingFace model).

### Collaboration Requirements

- **GitHub Repository:** One shared repo per team.
  - **Feature Branches:** No commits directly to `main`. Every feature must come from a branch (e.g., `feature/ner-worker`).
  - **PR Reviews:** Every merge to `main` requires a review and approval from the partner.
  - **Conflict Resolution:** Teams must intentionally trigger and resolve at least one merge conflict, documenting the resolution in a `CONFLICT.md` file.
- 

## Part 2: Dockerization & Swarm Orchestration

Students will take their service from local code to a distributed cluster using Docker Swarm.

### 1. Dockerization

- Build a **Docker image** using a `python:3.11-slim` or similar lightweight base image.
- **Security:** All API keys must be loaded via **Environment Variables**.
- **Metadata:** The API must include an `id` or `hostname` field in its JSON responses. This field should return the **Container ID** (e.g., `socket.gethostname()`) so you can track which container handled the request.

## 2. Docker Swarm Deployment

- **Cluster Setup:** Initialize a Swarm where **Machine 1** is the Leader/Manager and **Machine 2** is a Worker.
- **Stack Deployment:** Use a `docker-stack.yml` to deploy the service with **4 replicas**.
- **Load Balancing:** Demonstrate that Docker Swarm's Ingress Mesh is distributing traffic.

## 3. Load Balance Tester (`tester.py`)

Students must provide a client script that sends 100 concurrent requests to the service. Here is a sample tester script you can modify to create your own.

```
import requests
from collections import Counter
import concurrent.futures

# Template logic for students
URL = "http://<leader-ip>:8000/ner"
def call_api():
    try:
        r = requests.post(URL, json={"text": "Apple is hiring in London."})
        return r.json().get("container_id")
    except:
        return "Failed"

# Send 100 requests and count hits per container
with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
    results = list(executor.map(lambda _: call_api(), range(100)))

print(Counter(results)) # Proof that different containers handled the load
```

---

## Code of Conduct: Fair Use of AI

- **Architectural Ownership:** You may use AI to generate Dockerfile syntax or boilerplate for API endpoints. However, you must be able to explain the "Layer Caching" in your Dockerfile and the "Overlay Network" in your Swarm.
  - **Commit Transparency:** If a specific code block was generated by AI, the **Git Commit Message** must include the tag [AI-Generated].
  - **Security Warning:** Do not paste your API keys into public LLMs. Use placeholders like `YOUR_API_KEY_HERE` in prompts.
  - The above THREE pointers are in addition to the "Code of Conduct" shared previously.
-

# Deadlines

**Part 1** (Wednesday, 18th Feb 2026, 4 pm for in-class submissions, 11 am for students who have permission to be absent)

**Part 2** (Saturday, 21st Feb 2026, 11:59 pm)

---

## Grading Rubric (100 Points)

Category	Points	Criteria
<b>Feature Implementation</b>	20	All 4 AI tasks (Translation, NER, Image, Speech) work via the REST API.
<b>Git &amp; Collaboration</b>	20	Clear history of branches and PRs. Documented resolution of a merge conflict.
<b>Docker Optimization</b>	20	Use of small base images, <code>.dockerignore</code> , and env-var security.
<b>Swarm Configuration</b>	25	Successful deployment across 2 nodes with visible replicas ( <code>docker service ls</code> ).
<b>Distribution Proof</b>	15	The <code>tester.py</code> output shows the load distributed across multiple container IDs.

## Some References

### Docker Swarm Load Balancing & High Availability

This video provides a practical demonstration of how Docker Swarm automatically handles load balancing and scaling across nodes, which is essential for Part 2 of the assignment.

### A Beginner's Guide to Docker Swarm

<https://www.dolthub.com/blog/2025-08-27-a-beginners-guide-to-docker-swarm/>