EDep

ML Services Platform User Manual

eDep

Table of Contents

1 System Requirements

Minimum Hardware Specifications

- **RAM**: 4 GB (8 GB recommended for ML workloads)
- **Storage**: 10 GB free disk space
- **Processor**: x86-64 multi-core CPU
- **GPU**: Optional (NVIDIA GPU recommended for accelerated ML tasks)

1.1 Supported Operating Systems

OS Family	Supported Versions
Windows	10/11 (64-bit)
macOS	10.15 (Catalina)+
Linux	Ubuntu 20.04+, Debian 10+, CentOS 8+

2 Dependencies

2.1 Core Prerequisites

1. Docker Engine - External

- Minimum version: 20.10+
- Required components:
  - Docker Compose v2.4+
  - Docker Desktop (Windows/macOS) or Docker CE (Linux)
- Installation guide: `https://docs.docker.com/engine/install/`

2. Python 3.9+ (Included)

- Required packages:

```
pip install fastapi uvicorn python-jose[cryptography]
```

3. Node.js 16.14+ (Included)

- Required for GUI components
- Includes npm package manager

2.2 Backend Dependencies (Included)

```
# requirements.txt
fastapi>=0.68.0
uvicorn>=0.15.0
python-jose[cryptography]>=3.3.0
docker>=6.0.0
python-dotenv>=0.19.0
```

2.3 Frontend Dependencies (Included)

```
# gui/package.json
{
  "dependencies": {
    "electron": "^23.0.0",
    "node-fetch": "^3.3.0",
    "ps-list": "^8.0.0",
    "dotenv": "^16.0.0"
  }
}
```

2.4 Development Tools

- **Python**: Virtual environment manager (venv/conda)
- **Node.js**: npm scripts for build automation
- **IDE**: VS Code (recommended) with Python/JavaScript extensions

* Python 3.11+ recommended for optimal performance
* Node.js 18.x LTS supported but not required

3 Install eDep

Depending on the user operating system, the installer file should be chosen respectively. Please refer to Installer Page and download the installer for your particular OS.

4 Services

This desktop app is meant to serve as a plattform for the user to manage and control the deployment of machine learning and deep learning applications in local networks. Each service represent a standalone isolated environment that holds the neccessary dependencies to host a particular machine learning model. In order to create any service please make sure to count on handy the following requirements:

- **Model file**: The file that will be used to load the machine learning model. This can be in any of these formats: pkl, h5, onnx and keras.
- **Service URL**: Select a port that's not already in use within your local network and stablish the service url as http://localhost:yourport/servicetag
- **Data Schema**: This is the form of the body that the http requests should meet to effectively poll the service url.
- **requirements.txt**: This is a file containing all of the dependencies neccessary to consume the model, including preprocessing pipeline scripts.
- **Ingestion Pipeline**: This is a .py file serving as the consumption script of the model. This file must host a method named process_input() that takes in one parameter only: the HTTP request body. The process_input(data) method must execute all preprocessing logic to the argument object, apply the model and return its output.

4.1 Create your first service

Once eDep gets installed on your machine, launch it and ensure that the Docker engine service is running after completing app's release. You can check on top of the tool bar to determine whether the intialization of the Docker engine was successful or not.

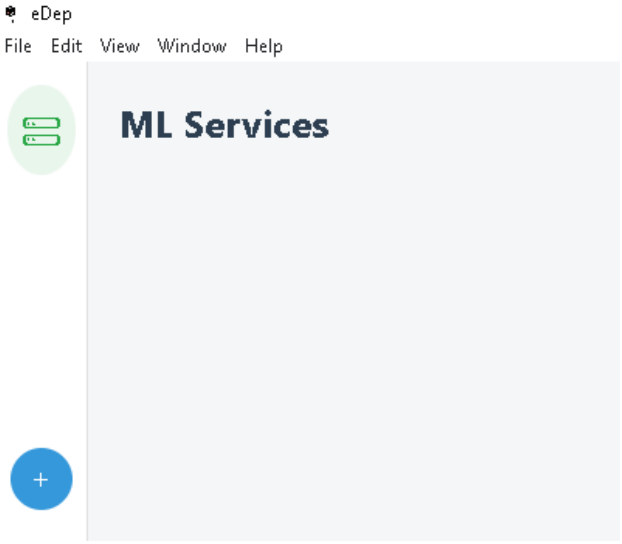

Figure 1: Docker Init Success

Now that Docker is ready to use, it can be time to define your first service. To so, simply click on 'Create First Service':

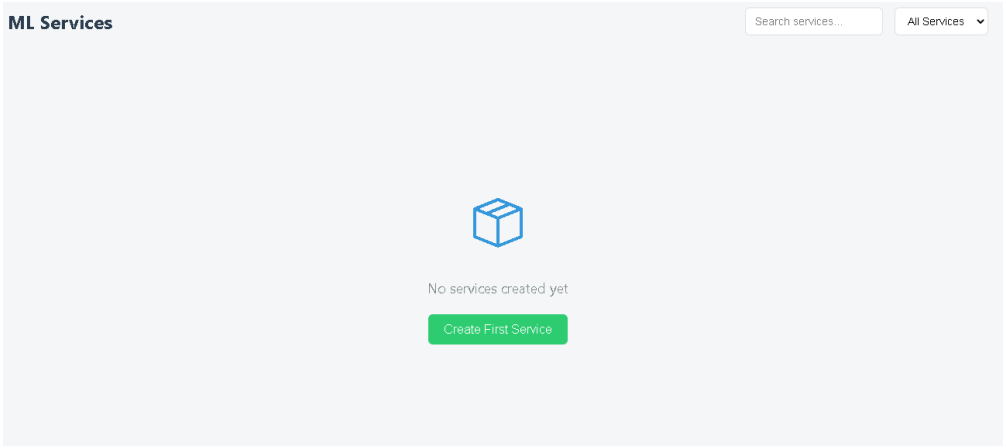

Figure 2: Create brand new service

In many scenarios there could be conflicts getting the Docker engine initiated. In such cases eDep will warn about this by ticking a red indicator over the server icon on top of the tool bar.

A new service creation prompt should show up requiring you to provide the newly service files and details. Make sure you have all of the aforementioned requirements and fill in all spaces with your service data.

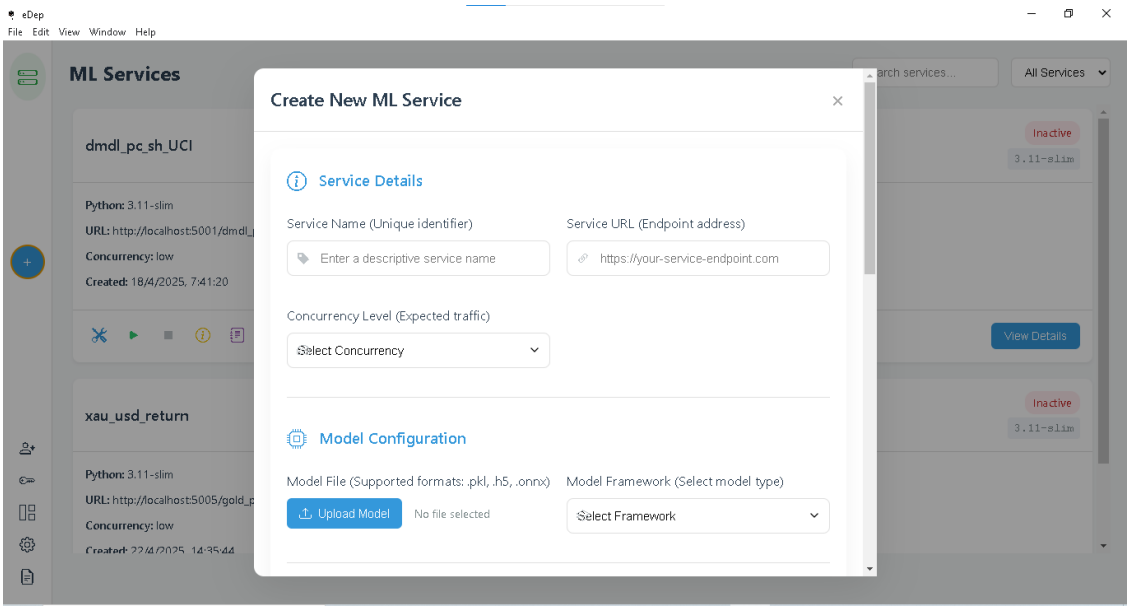

Figure 3: Service Creation Prompt

Once you're done providing the service details, click on submit and a file explorer terminal should show up with the root directory of the program services. Within this root address the program will be storing the all the services metadata files.

4.2 Concurrency

Every service will allow three different levels of concurrency and based on that the program will deploy the solution in the sense of replicas. If low concurrency is selected, then only 1 single replica is set to deploy. For medium level, 3 replicas are employed and for high concurrency 10 replicas are used. It is important to note that this will reflect directly in hardware resources being consumed. The services configured up to any moment in time can be tunned in a different level of concurrency by managing the service metadata files in the root directory, if at any moment there's the need to upgrade/downgrade concurrency on any service, then the user should do it by modifying the medatada json file.

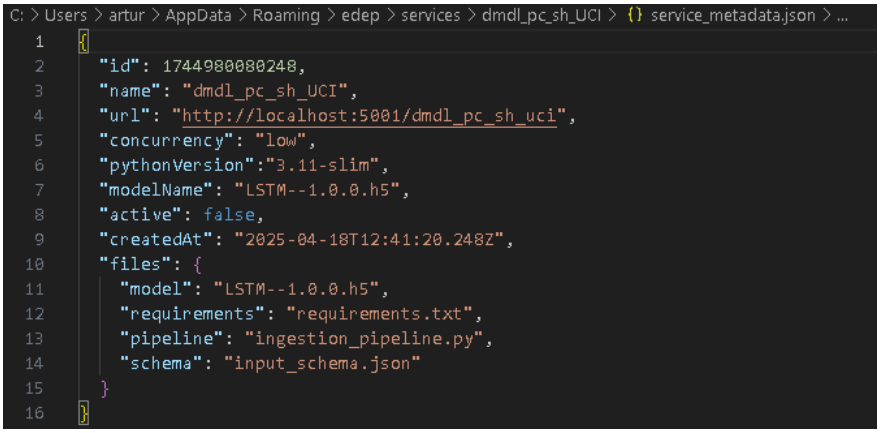

Figure 4: Metadata config file

Simply navigate to

```
C:\Users\youruser\AppData\Roaming\edep\services\yourservice
```

and open up the service metadata json file to edit what's needed.

4.3 Build Service

Once the service is created, the next step upon deplyoment is to build it. Building the service will create the neccessary files to isolate the environment and deploy as a container. This operation can take up to several minutes, it is expected to go over this process only once per service. All of the build files will be located within the program files in a directory named docker_builds. Once the build process has finished, the logs will show up with the build details:

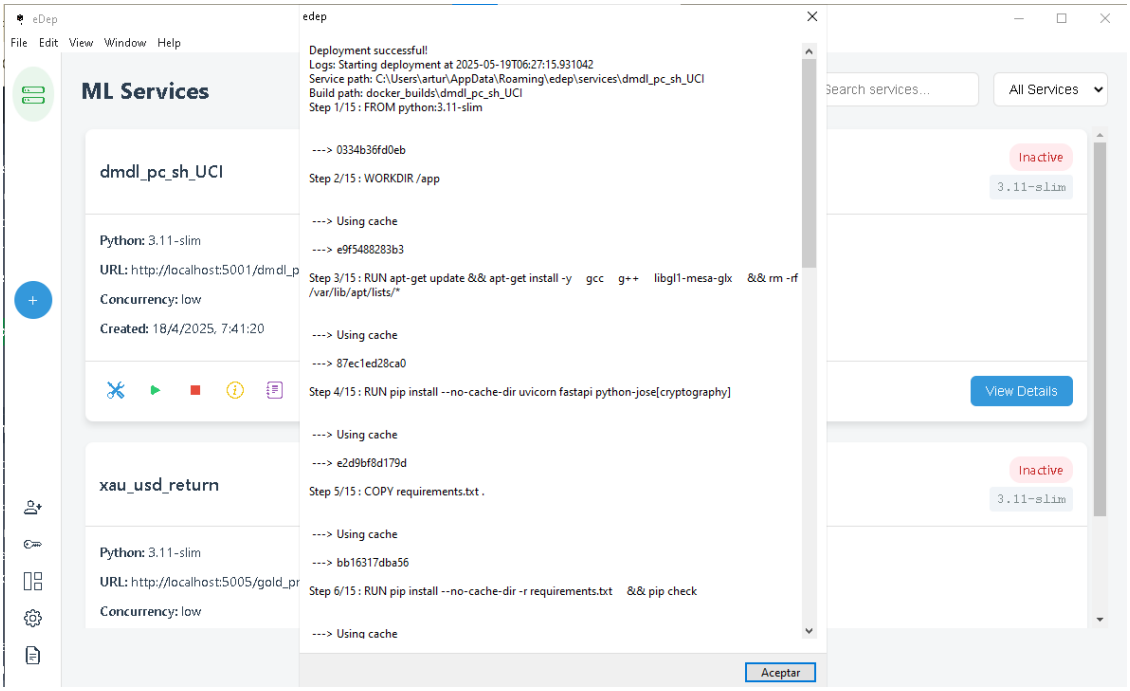

Figure 5: Service build success

If the build is successful, a log message will inform about it and start the service right after finishing the building. To check on your service health status, copy the service url you have assigned to it and add the route 'docs' at the end: http://localhost:serviceport/docs. There should be a swagger interface supporting your service API.

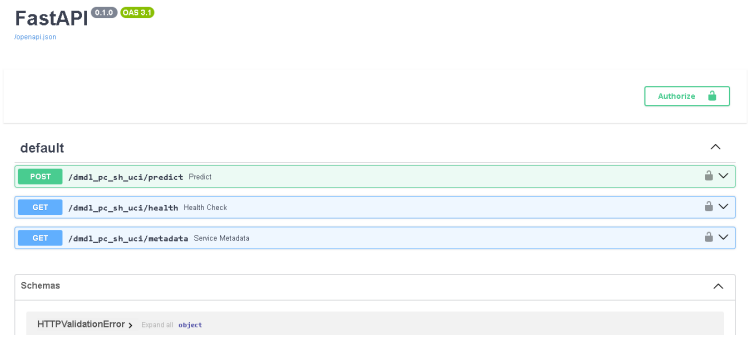

Figure 6: Service app

4.4 Authenticating Services

To grant the right access to any registered service, the admin must assign user identification and record the new licenses that are expected to be using the services. To create a new authorized user of the app, please refer to the add user button in the tool bar just below the add service button. You will be prompted to provide username and password.

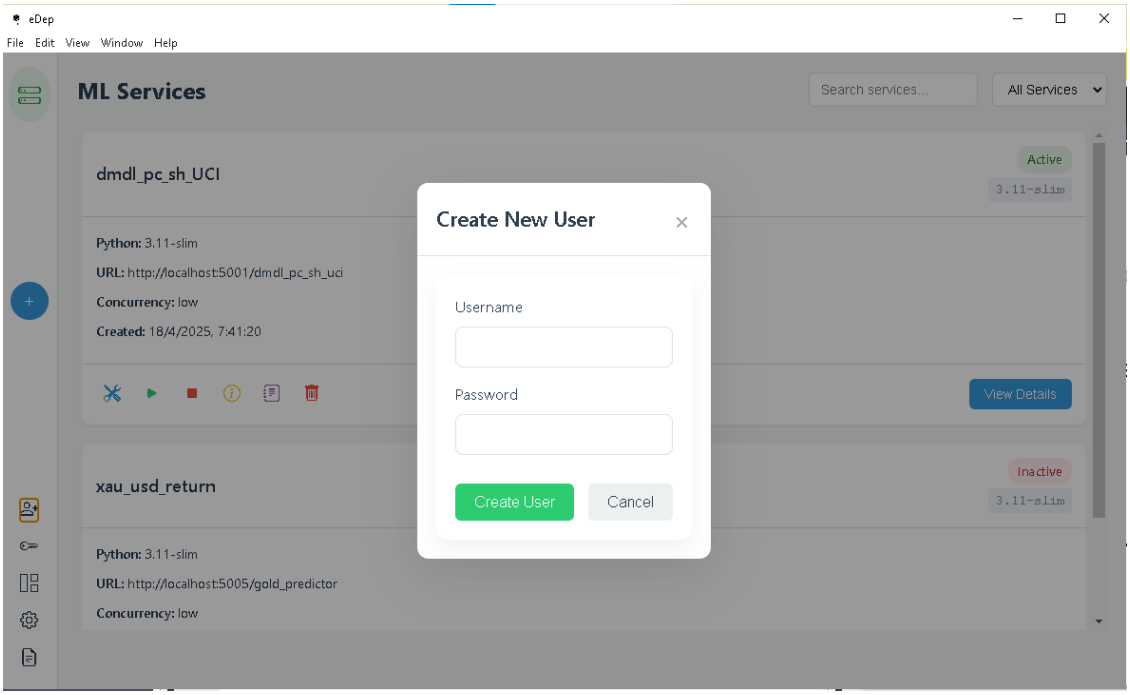

Figure 7: Create new user

Every authorized user must request for a session key so it has the token that will unlock the service apps. These token keys are set to expire after 1 hour of being spawned in the beta version.

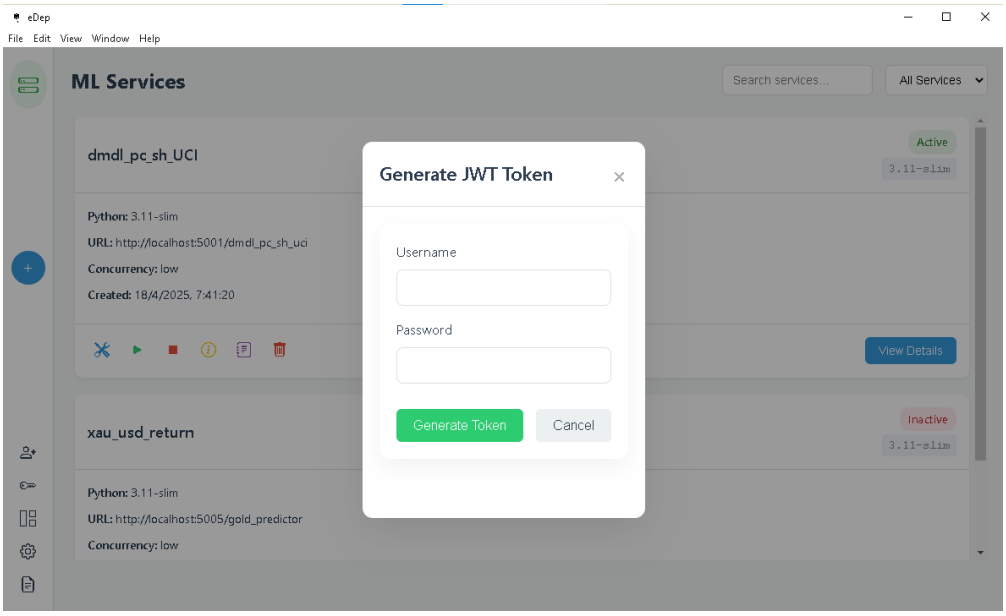

Figure 8: Create new token key

Make sure you copy the token generated and store it in a safe place to later consume the service apps.

4.5 Start/Stop Services

The user can start/stop any service when required by using the service buttons placed at the bottom of the service card. Each service card has got its own flag to inform the admin whether being Active or Inactive. The admin can also click on the information button to double check the service status in the backend and the number of replicas being used.

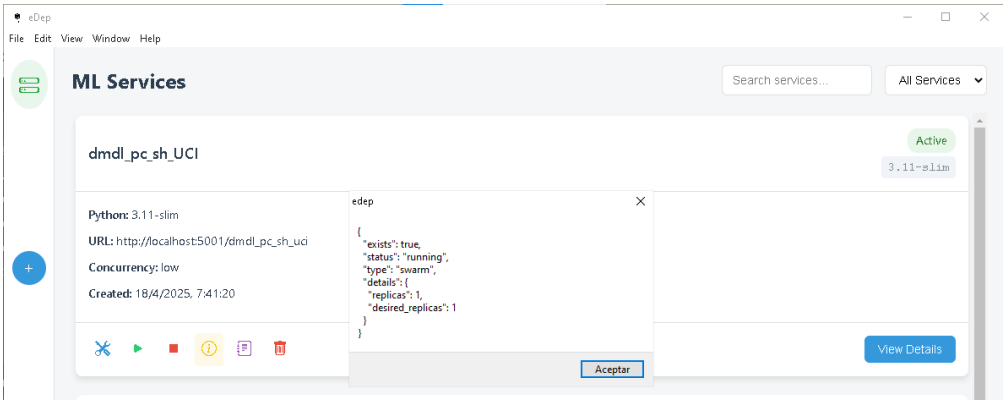

Figure 9: Service Status

Simply click on the 'Stop' button to finish the execution of the particular service and 'Play' to start it if it has not already been launched.

5 FAQ

This section addresses common questions about the eDep ML Services Platform. For additional support, visit our help center.

5.1 Installation & Setup

Q1: What are the minimum system requirements?

Requires 64-bit OS (Windows 10+, macOS 10.15+, Ubuntu 20.04+), 4GB RAM (8GB recommended), and 10GB disk space. Docker Engine 20.10+ must be installed.

Q2: How do I resolve Docker dependency errors?

Verify Docker is running with `docker info`. On Linux, ensure user is in docker group: `sudo usermod -aG docker $USER`

Q3: Where are services stored?

Default service directory: `C:\Users\<username>\AppData\Roaming\edep\services`

5.2 Service Management

Q4: How to create a new ML service?

1. Click + in sidebar 2. Provide unique service name 3. Select Python version (≥3.8) 4. Upload model file (.pkl, .h5, .onnx) 5. Define input schema (JSON format)

Q5: Why is my service status *Inactive*?
Common causes:

- Port conflict (check service URL)
- Missing model file dependencies
- Docker resource constraints

Q6: How to view deployment logs?
Click **View Logs** (notebook icon) on service card. Logs stored in: `<service_dir>/docker_builds/logs`

5.3 User & Security

Q7: How to reset user permissions?
Admin users can regenerate JWT tokens via **Settings → Security**. Existing tokens are invalidated immediately.

Q8: Is my model data secure?
All services run in isolated Docker containers. Model files are encrypted at rest using AES-256 when stored.

Q9: Maximum concurrent users?
Community Edition: 5 users Enterprise Edition: Unlimited (role-based access)

5.4 Troubleshooting

Q10: `ERROR: Port already allocated`
1. Identify conflicting process: `sudo lsof -i :<port>` 2. Terminate process or change service URL

] Ensure requirements.txt includes all dependencies. Use pinned versions: `numpy==1.23.5 # Avoid compatibility issues`

] 1. Check Docker resource allocation 2. Verify free disk space ≥1GB 3. Restart application with: `edep --clear-cache`

Note: For unresolved issues, collect diagnostic logs using `edep-diag` tool and contact support@edep.com with:

- Platform version (`edep --version`)
- Error message screenshot
- Diagnostic bundle ID