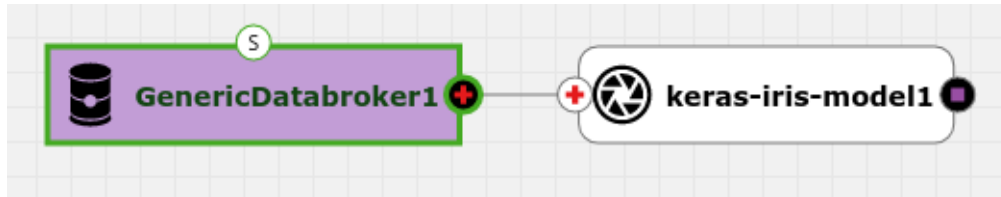


AI4EU Experiments Container Format



1. Introduction

This document specifies the docker container format for tools and models that can be onboarded on the AI4EU Experiments platform so they can be used in the visual composition editor as re-usable, highly interoperable building blocks for AI pipelines.

In short, the container should define its public service methods using protobuf v3 and expose these methods via gRPC. All these technologies are open source and freely available. Here are the respective home pages for documentation and reference:

<https://docs.docker.com/reference/>

<https://developers.google.com/protocol-buffers/docs/overview>

<https://developers.google.com/protocol-buffers/docs/proto3>

<https://www.grpc.io/docs/>

Because the goal is to have re-usable building blocks to compose pipelines, the main reason to choose the above technology stack is to achieve the highest level of interoperability:

- docker is today the defacto standard for serverside software distribution including all dependencies. It is possible to onboard containers for different architectures (x86_64, GPU, ARM, HPC/Singularity)
- gRPC together with protobuf is a proven specification and implementation for remote procedure calls supporting a broad range of programming languages and it is optimized for performance and high throughput.

2. Define model.proto

The public service methods should be defined in a file called **model.proto**:

- It should be self contained, thus contain the service definitions with all input and output data structures.
- The full feature set of protobuf v3 can be used
- a container can define several methods, but all in the same file called model.proto
- the file must define a globally unique package name because AI4EU Experiments needs to distinguish all onboarded protobuf interfaces

```
// set used version of protobuf
syntax = "proto3";

// set unique package name
package fraunhofer.demo.cpp.iris.v1

// define input data structure
message IrisDataFrame {
    repeated double sepal_length = 1;
    repeated double sepal_width = 2;
    repeated double petal_length = 3;
    repeated double petal_width = 4;
}

// define output data structure
message ClassifyOut {
    repeated int64 value = 1;
}

// define exposed service
service Model {
    rpc classify (IrisDataFrame) returns (ClassifyOut);
}
```

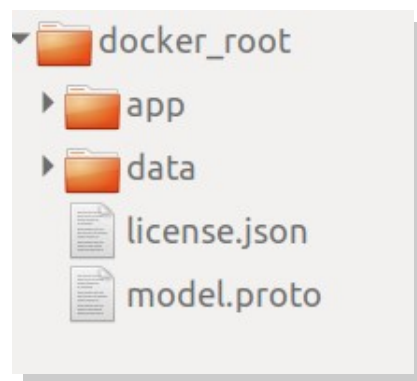
3. Create the gRPC microservice

Based on `model.proto`, you can generate the necessary gRPC stubs and skeletons for the programming language of your choice using the protobuf compiler `protoc` and the respective `protoc`-plugins. Then create a short main executable that will read and initialize the model or tool and starts the gRPC server. This executable will be the entrypoint for the docker container.

The filetree of the docker container should look like below. In the top level folder of the container are the files

- `model.proto`
- `license.json`

And also the folders for the microservice like `app` and `data`, or any supplementary folders:



The license file is not mandatory and can be generated after onboarding with the License Profile Editor in the AI4EU Experiments Web-UI:

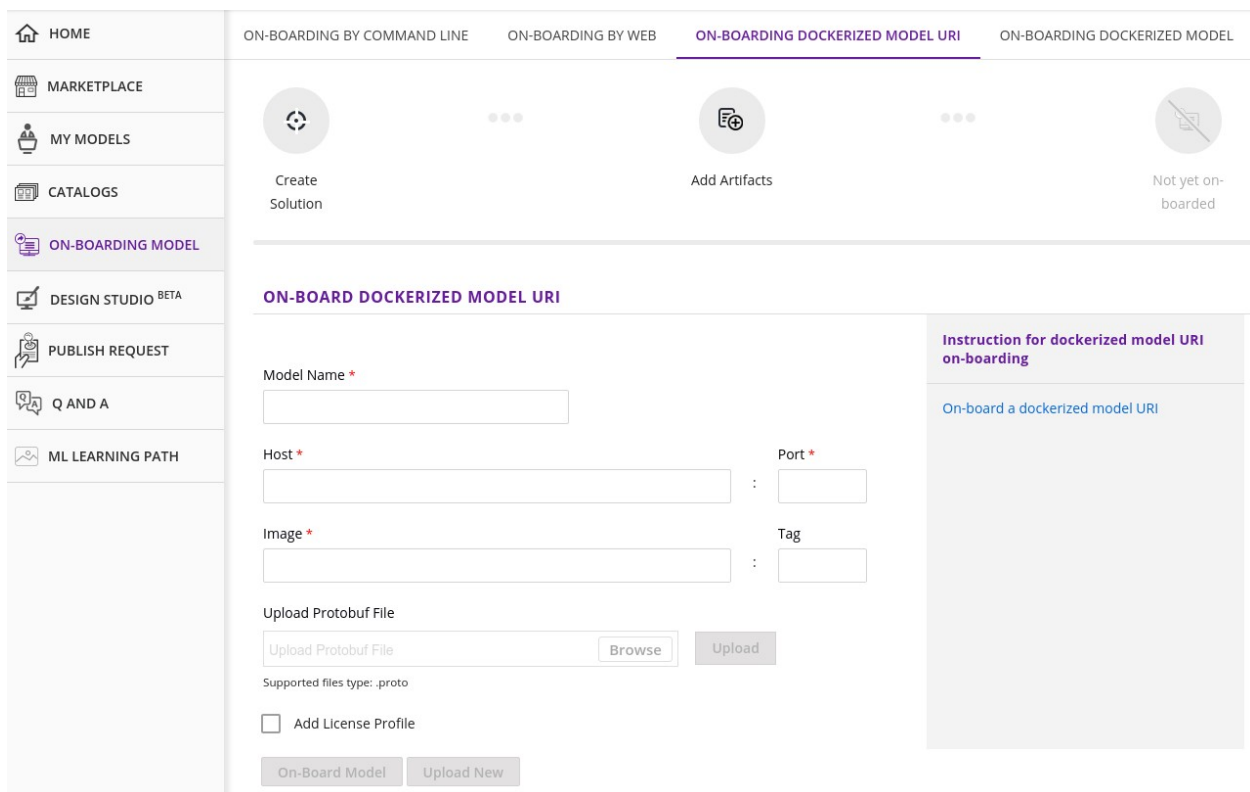
<https://docs.acumos.org/en/clio/submodules/license-manager/docs/user-guide-license-profile-editor.html>

There are several detailed tutorials on how to create a dockerized model in this repository: <https://github.com/ai4eu/tutorials>

4. Onboarding

The final step is to onboard the model. There are several ways to onboard a model into AI4EU Experiments but currently the only recommended way is to use “On-boarding dockerized model URI”:

1. Upload your docker container to a public registry like Docker Hub
2. Start the onboarding process like in the screenshot below
3. Upload the protobuf file
4. Add license profile



HOME

MARKETPLACE

MY MODELS

CATALOGS

ON-BOARDING MODEL

DESIGN STUDIO BETA

PUBLISH REQUEST

Q AND A

ML LEARNING PATH

ON-BOARDING BY COMMAND LINE

ON-BOARDING BY WEB

ON-BOARDING DOCKERIZED MODEL URI

ON-BOARDING DOCKERIZED MODEL

Create Solution

Add Artifacts

Not yet on-boarded

ON-BOARD DOCKERIZED MODEL URI

Model Name *

Host *

Port *

Image *

Tag

Upload Protobuf File

Upload Protobuf File

Browse

Upload

Supported files type: .proto

☐ Add License Profile

On-Board Model

Upload New

Instruction for dockerized model URI on-boarding

[On-board a dockerized model URI](#)