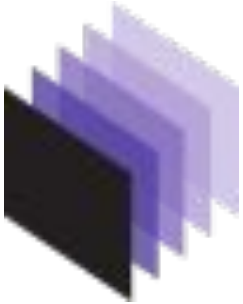
4th IPOL MLBriefs Workshop
How to create an IPOL demo?

May 27th — May 31st

ENS Paris-Saclay, Gif-sur-Yvette, France



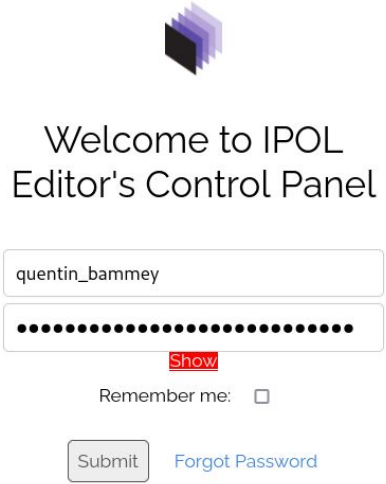
Introduction

Where to create an IPOL demo

- The IPOL Control Panel
- A github repository
- Your computer (to edit the code)

The IPOL control panel

- Account received this weekend by email, if not, ask us!
- Log in to https://ipolcore.ipol.im/cp2/



Github repository

- Your code should be on a **github** repository
- There will be some IPOL-specific files in the repository in addition to the main file
- ⇒ Keep the repository **private**, and separate from the one you show to the public
- Only github is supported, not other git repositories

Creating your first demo

Take the template:
https://github.com/mlbriefs/template-python

1. **Prepare your code**
2. Containerize the code
3. Create the demo on the control panel
4. Push the code to the git repository
5. Create the demo DDL

Your first demo: prepare the code

- Make sure the code works
- What are the **requirements**? (python version, packages, libraries)
- What does the code require as **inputs**? (input data, parameters)
- What are the **outputs** you want to display? (result, intermediary results)
- How to **call** the code from the command line, with the inputs and parameters?
  - Expose the call and the arguments to the command line (for instance, use argparse in python)

Your first demo: prepare the code (with the template)

- Make sure the code works
- What are the **requirements**? (packages, libraries)
  - Python 3.9, numpy==1.26.3, iio=0.0.3
- What does the code require as **inputs**? (input data, parameters)
  - One image, and the standard deviation of the noise to add (float between 0 and 300, default 30)
- What are the **outputs** you want to display? (result, intermediary results)
  - The input image, and the output image with added noise
- How to **call** the code from the command line, with the inputs and parameters?
  - python $bin/main.py --input $input_0 --sigma $sigma --output output.png

Your first demo: prepare the code

Make sure your code can be executed from anywhere on the filesystem:

```
[user@laptop:~/myproject]$ python main.py
...
[user@laptop:~/myproject]$ cd /tmp
[user@laptop:/tmp]$ python ~/myproject/main.py
...
```

Instead of

```
torch.load('weights.pth')
```

use

```
ROOT = os.path.dirname(os.path.realpath(__file__))
torch.load(os.path.join(ROOT, 'weights.pth'))
```

or expose the weights as parameter.

In IPOL, the code is in $bin, but the execution is from somewhere else: call, e. g.,

"python $bin/main.py"

Instead of

"python main.py"

Creating your first demo

Take the template:
https://github.com/mlbriefs/template-python

1. Prepare your code
2. **Containerize the code**
3. Create the demo on the control panel
4. Push the code to the git repository
5. Create the demo DDL

Your first demo: containerize the code

- Create a Dockerfile in your code at .ipol/Dockerfile (already there in the template)
  Start from the template file.
- Choose a base version from here and change it in the Dockerfile

```dockerfile
# use one of the images from this repository: https://github.com/centreborelli/ipol-docker-images/
FROM registry.ipol.im/ipol:v1-py3.9

# install additional debian packages
COPY .ipol/packages.txt packages.txt
RUN apt-get update && apt-get install -y $(cat packages.txt) && rm -rf /var/lib/apt/lists/* && rm packages.txt

# copy the requirements.txt and install python packages
COPY requirements.txt requirements.txt
RUN pip3 install --no-cache-dir -r requirements.txt && rm requirements.txt

# copy the code to $bin
ENV bin /workdir/bin/
RUN mkdir -p $bin
WORKDIR $bin
COPY . .

# the execution will happen in the folder /workdir/exec
# it will be created by IPOL

# some QoL tweaks
ENV PYTHONDONTWRITEBYTECODE 1
ENV PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION python
ENV PATH $bin:$PATH

# $HOME is writable by the user `ipol`, but
ENV HOME /home/ipol
# chmod 777 so that any user can use the HOME, in case the docker is run with -u 1001:1001
RUN groupadd -g 1000 ipol && useradd -m -u 1000 -g 1000 ipol -d $HOME && chmod -R 777 $HOME
USER ipol
```

Your first demo: containerize the code (Docker base)

- How to choose the docker base image?
- Choose from https://github.com/ipol-journal/ipol-docker-images/blob/main/README.md
- octave: registry.ipol.im/ipol:v2-octave
- python: registry.ipol.im/ipol:v2-py3.{9,10,11} (choose the version)
- python+pytorch: registry.ipol.im/ipol:v2-py3.{9,10,11}-pytorch
- python+tensorflow: registry.ipol.im/ipol:v2-py3.{9,10,11}-tensorflow
- python+GPU (python 3.11 only): registry.ipol.im/ipol:v2-py3.11-gpu, registry.ipol.im/ipol:v2-py3.11-pytorch-gpu, registry.ipol.im/ipol:v2-py3.11-tensorflow-gpu
- Other languages: choose any
- Older versions available
- It is possible to use non-IPOL docker images (only if needed: easier for us to help you with IPOL images)

A word on GPUs

- In short: **Avoid using GPUs on IPOL when possible**.
- GPU support is still experimental (harder to debug)
- Few GPU resources on IPOL ⇒ the demo will crash if too many people run GPU demos at the same time
- **IPOL is not a computation cluster** ⇒ when possible, demos should run smoothly without a GPU. Limit max input size if necessary

Install needed packages

- Create the file **.ipol/packages.txt** (already there in the template)
- Add any Debian package you need to install, one per line
- Do not add python libraries (will be installed later with pip)
- Keep the file empty if you do not need to install files, but do not remove it
- Some files already installed in the IPOL images (check the images readme)

```dockerfile
# use one of the images from this repository: https://github.com/centreborelli/ipol-docker-images/
FROM registry.ipol.im/ipol:v1-py3.9

# install additional debian packages
COPY .ipol/packages.txt packages.txt
RUN apt-get update && apt-get install -y $(cat packages.txt) && rm -rf /var/lib/apt/lists/* && rm packages.txt

# copy the requirements.txt and install python packages
COPY requirements.txt requirements.txt
RUN pip3 install --no-cache-dir -r requirements.txt && rm requirements.txt

# copy the code to $bin
ENV bin /workdir/bin/
RUN mkdir -p $bin
WORKDIR $bin
COPY . .

# the execution will happen in the folder /workdir/exec
# it will be created by IPOL

# some QoL tweaks
ENV PYTHONDONTWRITEBYTECODE 1
ENV PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION python
ENV PATH $bin:$PATH

# $HOME is writable by the user `ipol`, but
ENV HOME /home/ipol
# chmod 777 so that any user can use the HOME, in case the docker is run with -u 1001:1001
RUN groupadd -g 1000 ipol && useradd -m -u 1000 -g 1000 ipol -d $HOME && chmod -R 777 $HOME
USER ipol
```

Install python libraries

- Create the file **requirements.txt** at the code root (already there in the template)
- Add any pip libraries you need to install, one per line
- Keep the file empty if you do not need to install files, but do not remove it
- Specify the exact version of each library (for reproducibility)
- Several packages are already installed (check the images readme)
  - If you specify them again, try to use the same version as the base image if possible (to save storage space)
- In the template:

```
numpy==1.26.4
iio==0.0.3
```

```dockerfile
# use one of the images from this repository: https://github.com/centreborelli/ipol-docker-images/
FROM registry.ipol.im/ipol:v1-py3.9

# install additional debian packages
COPY .ipol/packages.txt packages.txt
RUN apt-get update && apt-get install -y $(cat packages.txt) && rm -rf /var/lib/apt/lists/* && rm packages.txt

# copy the requirements.txt and install python packages
COPY requirements.txt requirements.txt
RUN pip3 install --no-cache-dir -r requirements.txt && rm requirements.txt

# copy the code to $bin
ENV bin /workdir/bin/
RUN mkdir -p $bin
WORKDIR $bin
COPY . .

# the execution will happen in the folder /workdir/exec
# it will be created by IPOL

# some QoL tweaks
ENV PYTHONDONTWRITEBYTECODE 1
ENV PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION python
ENV PATH $bin:$PATH

# $HOME is writable by the user `ipol`, but
ENV HOME /home/ipol
# chmod 777 so that any user can use the HOME, in case the docker is run with -u 1001:1001
RUN groupadd -g 1000 ipol && useradd -m -u 1000 -g 1000 ipol -d $HOME && chmod -R 777 $HOME
USER ipol
```

Keep this part of the Dockerfile intact

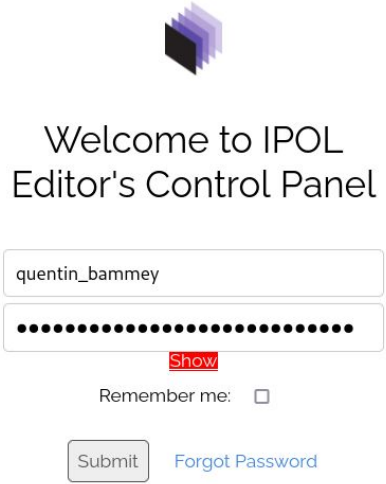
Creating your first demo

Take the template:
https://github.com/mlbriefs/template-python

1. Prepare your code
2. Containerize the code
3. **Create the demo on the control panel**
4. Push the code to the git repository
5. Create the demo DDL

The IPOL control panel

- Account received this weekend by email, if not, ask us!
- Log in to https://ipolcore.ipol.im/cp2/



Demos Templates Status Quentin Bammey

List of Demos

Search

Next >

Your demos

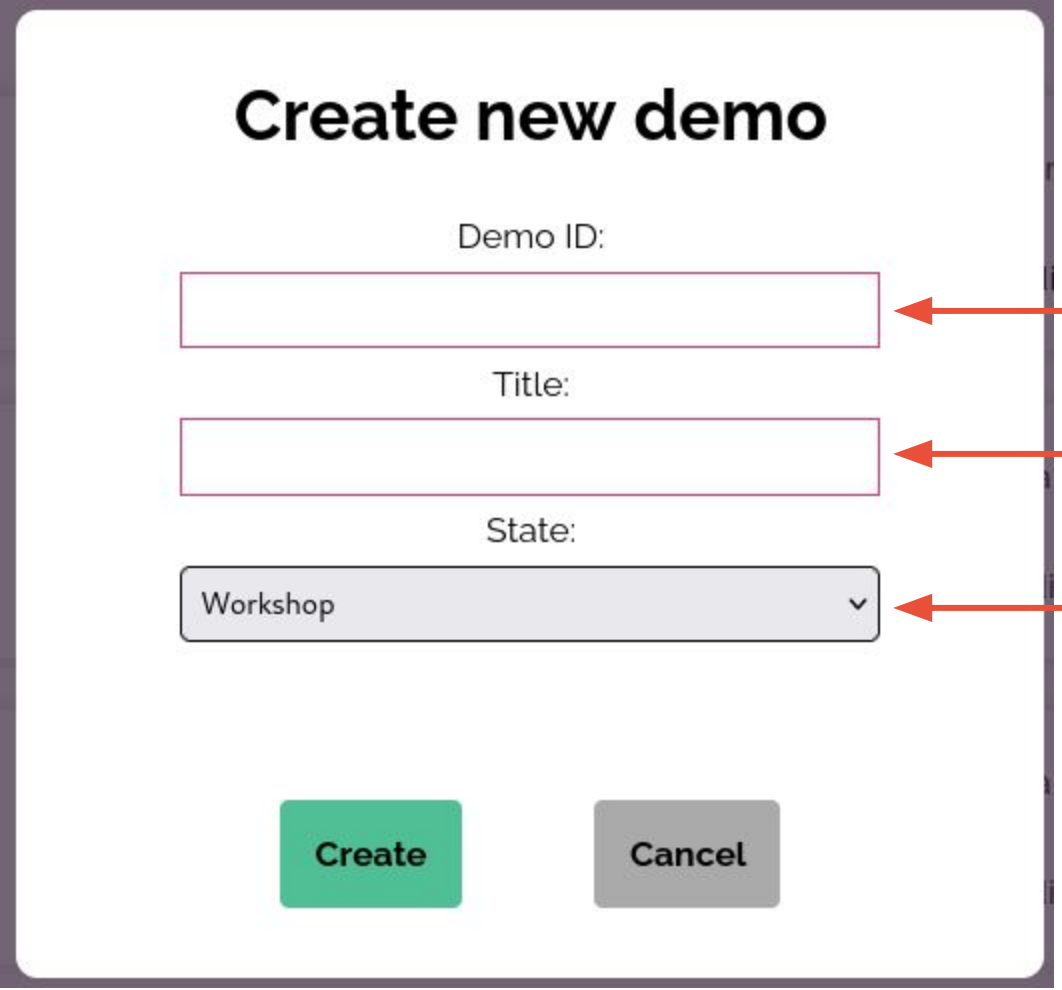
Demo title:
Image Seam Carving
ID: 77777000477
State: workshop
Last modification: 12 Dec 2023
Open demo
Edit demo

Demo title:
Demo Example: Archiving files
ID: 11111000005
State: example
Last modification: 09 Jun 2023
Open demo
Edit demo

Demo title:
Demo Example: Parameters types
ID: 11111000002
State: example
Last modification: 09 Jun 2023
Open demo
Edit demo

Demo title:
Demo Example: Image Inputs
ID: 11111000003
State: example
Last modification: 09 Jun 2023
Open demo
Edit demo

Demo title:
State: workshop
Open demo



Look at https://ipolcore.ipol.im/api/core/demo for the list of existing demos. Under "workshops", find the first non-private demo (that starts with 77777000). Take this demo number and add 1 to it for your own ID

Your title

Only use "workshop"

https://ipolcore.ipol.im/api/core/demo

List of demos

The demos whose ID begins with '77777' are public workshops and those with '33333' are private. Test demos begin with '55555' whereas Example demos begin with '11111'.

test

Demo #55555531082048: Masked Autoencoders for Anomaly Detection
Demo #55555531082047: Masked Autoencoders for Inpainting
Demo #55555531082042: Noiseprint
Demo #55555531082036: Color Image Vectorization
Demo #55555531082028: Experiments volumetric file display
Demo #55555531082027: Maximal accumulation for mesh segmentation
Demo #55555531082021: Oil Tank Detection with A Contrario Clustering
Demo #55555531012020: Two sample hypothesis testing with AUC maximization
Demo #55555512102001: Classification of time series by shapelet transformation
Demo #55555512102000: Image Recovery with Constrained Variational Problems
Demo #55555531091985: Matrix Profile - Test
Demo #55555500043222: docker test with gpu
Demo #55555500043221: docker test, with GPU
Demo #55555587654: Smart Coast Color Balance (Dev)
Demo #55555429730: Exact Optimization for Markov Random Fields with Convex Priors implementation
Demo #55555123999: Exact opti
Demo #55555000320: Shape-from-shading
Demo #55555000205: Numerical simulation
Demo #55555000174: Cradle Removal in X-Ray Images of Panel Paintings
Demo #55555000150: Tesssssttt Small Neural Networks can Denoise Image Textures Well: a Useful Complement to BM3D
Demo #55555000131: Test 133. An Algorithmic Analysis of Variational Models for Perceptual Local Contrast Enhancement
Demo #55555000122: Linear Methods for Image Interpolation
Demo #55555000104: An Analysis of the Viola-Jones Face Detection Algorithm
Demo #55555000024: Parking occupancy monitoring on PlanetScope images
Demo #55555000023: Traffic monitoring on PlanetScope images
Demo #55555000021: Change detection by correlation
Demo #55555000010: Virtualenv example - Adding virtualenv from a new build section
Demo #55555000008: Google Cloud Run Demo
Demo #55555000004: MVA Reproducible Research example
Demo #55555000097: Testtttt Kolmogorov and Zabih's Graph Cuts Stereo Matching Algorithm
Demo #55555000095: Image Demosaicking with Contour Stencils
Demo #55555000074: Extraction of Connected Region Boundary in Multidimensional Images
Demo #55555000070: 133 test
Demo #55555000046: Mino-Gilles Stabilization Algorithm
Demo #55555000006: trying to work on dr_green
Demo #55555000003: probando cosas
Demo #55555000002: Testing a lot
Demo #55555000001: Test replicating a matlab smo implementation
Demo #55555216: MVA demo example
Demo #165: Directional Filters for Cartoon + Texture Image Decomposition
Demo #153: The Inverse Compositional Algorithm for Parametric Registration
Demo #150: Small neural networks can denoise image textures well: a useful complement to BM3D
Demo #119: Accelerating Monte Carlo Renderers by Ray Histogram Fusion
Demo #104: An Analysis of the Viola-Jones Face Detection Algorithm
Demo #76: Cartoon+Texture Image Decomposition
Demo #70: A Near-Linear Time Guaranteed Algorithm for Digital Curve Simplification Under the Fréchet Distance
Demo #68: A Streaming Distance Transform Algorithm for Neighborhood-Sequence Distances
Demo #59: Exemplar-based Texture Synthesis: the Efros-Leung Algorithm

workshop

Demo #333330060001 (private): Magenline
Demo #333330050001 (private): Anomaly detection: local pixel detector
Demo #333330030003 (private): Crack detection
Demo #333330030002 (private): Crack demo
Demo #333330030001 (private): Detection of f
Demo #333330020003 (private): simple color balance
Demo #333330020004 (private): guide generator 4
Demo #333330020003 (private): guide generator 3
Demo #333330020002 (private): guide generator 2
Demo #333330020001 (private): guide generator
Demo #333330010011 (private): Methane detection (temporal)
Demo #333330010010 (private): ESDA - Comparison of Motion Smoothing Strategies for Video Stabilization using Parametric Models
Demo #333330010009 (private): Methane detection (spatial)
Demo #333330010006 (private): Nimbus SIFT for Planet
Demo #333330010005 (private): Hidden
Demo #333330010004 (private): Nimbus
Demo #333330010003 (private): Robust optical flow2
Demo #333330010002 (private): Novelty
Demo #333330010001 (private): Clouds
Demo #77777000501: Change detection from temporal redundancy
Demo #77777000499: PatchFusion
Demo #77777000498: Metric3D
Demo #77777000497: Depth-Anything
Demo #77777000496: Unilength
Demo #77777000495: Zoellength
Demo #77777000494: Solar panel detection on S2 : abundance approaches
Demo #77777000493: Solar panel detection on S2 : unmixing approaches
Demo #77777000492: Solar panel detection on S2 : growing approaches

Demo #70: A Near-Linear Time Guaranteed Algorithm for Digital Curve Sim
Demo #68: A Streaming Distance Transform Algorithm for Neighborhood-Se
Demo #59: Exemplar-based Texture Synthesis: the Efros-Leung Algorithm

workshop

Demo #333330060001 (private): Magentine
Demo #333330050001 (private): Anomaly detection: local pixel detector
Demo #333330030003 (private): Crack detection
Demo #333330030002 (private): Crack demo
Demo #333330030001 (private): Detection of f.
Demo #333330020005 (private): simple color balance
Demo #333330020004 (private): guide generator 4
Demo #333330020003 (private): guide generator 3
Demo #333330020002 (private): guide generator 2
Demo #333330020001 (private): guide generator
Demo #333330010011 (private): Methane detection (temporal)
Demo #333330010010 (private): ESOA - Comparison of Motion Smoothing St
Demo #333330010009 (private): Methane detection (spatial)
Demo #333330010006 (private): Nimbus SIFT for Planet
Demo #333330010005 (private): Hidden
Demo #333330010004 (private): Nimbus
Demo #333330010003 (private): Robust optical flow2
Demo #333330010002 (private): Novelty
Demo #333330010001 (private): Clouds
Demo #77777000501: Change detection from temporal redundancy
Demo #77777000499: PatchFusion
Demo #77777000498: Metric3D
Demo #77777000497: Depth-Anything

This is the first non-private (starting with 77777) demo in the workshop category. Its id is 77777000501, so you should now create a demo with id **77777000502**

https://ipolcore.ipol.im/cp2/showDemo?demo_id=77777000502 80%

Demos Templates Status Quentin Bammey

Demo Editor

Title: test demo

Demo Extras | Archive | Blobs | Editors | Open demo | Edit demo

DDL editor

```
1 {}
```

We will fill it later

We will need it soon

Save DDL | DDL already saved. | DDL History | White Theme | Dark Theme

SSH public key: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPcb/tYZ1b03Dg1VB7sbKoLgrGViYhqkvmHfXrklSUKv | Copy key | Reset key

Creating your first demo

Take the template:
https://github.com/mlbriefs/template-python

1. Prepare your code
2. Containerize the code
3. Create the demo on the control panel
4. **Push the code to the git repository**
5. Create the demo DDL

Github repository

- Your code should be on a **github** repository
- There are some IPOL-specific files in the repository in addition to the main file: .ipol/Dockerfile, .ipol/packages.txt, requirements.txt
- ⇒ Keep the repository **private**, and separate from the one you show to the public
- Only github is supported, not other git repositories
- If you followed the tutorial, you started from the git template and your code is already in a git repo. If not, create the repository and push the code
- Once done: you need to enable IPOL to access the repository (even if it is public)

https://github.com/mlbriefs/template-python/settings/keys

mlbriefs / template-python

Code | Issues | Pull requests | Security | Insights | Settings

1

General

Access

Collaborators and teams

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Pages

Custom properties

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

2

Deploy keys

Add deploy key

3

ipolcore.ipol.im

SHA256:4LNsQgUH1A9AIhRtWSFd5piFFiHGHg3VEhCiM9NJYmQ

Added on Aug 26, 2022 by @kidanger

Last used within the last 4 months — Read-only

SSH

Delete

integration.ipol.im

SHA256:u+vNitGFVqrunEOjecYVXOhkcsEcwWOjtSDGpBXpYtA

Added on Oct 16, 2023 by @hmaciasc

Last used within the last 4 months — Read-only

SSH

Delete

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

https://github.com/mlbriefs/template-python/settings/keys/new

mlbriefs / template-python

Code | Issues | Pull requests | Security | Insights | Settings

General

Access

Collaborators and teams

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Pages

Custom properties

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

Deploy keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

☐ **Allow write access**

Can this key be used to **push** to this repository? Deploy keys always have pull access.

Add key

https://ipolcore.ipol.im/cp2/showDemo?demo_id=77777000502 80%

Demos Templates Status Quentin Bammey

Demo Editor

Title: test demo

Demo Extras | Archive | Blobs | Editors | Open demo | Edit demo

DDL editor

```
1 {}
```

Save DDL | DDL already saved. | DDL History | White Theme | Dark Theme

SSH public key: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPcb/tYZ1b03Dg1VB7sbKoLgrGViYhqkvmHfXrklSUKv | Copy key | Reset key

Copy the key from the control panel

https://github.com/mlbriefs/template-python/settings/keys/new

mlbriefs / template-python

Code | Issues | Pull requests | Security | Insights | Settings

General

Access

Collaborators and teams

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Pages

Custom properties

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

Deploy keys / Add new

Title

ipolcore

Key

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPcb/tYZ1b03Dg1VB7sbKoLgrGViYhqkvmHfXrklSUKv

Paste the key here

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'.

☐ **Allow write access**

Can this key be used to **push** to this repository? Deploy keys always have pull access.

Add key

Confirm (github will ask you to authentify with 2FA)

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Github and large files

Github limits to 100MB per file. After that, the push is rejected and you have to remove the commit from your branch.

If you need large files (e. g. model weights), you will have to upload them somewhere (we can provide you with a link) and download them from the Dockerfile. We will explain that tomorrow.

Creating your first demo

Take the template:
https://github.com/mlbriefs/template-python

1. Prepare your code
2. Containerize the code
3. Create the demo on the control panel
4. Push the code to the git repository
5. **Create the demo DDL**

Writing the DDL

The Demo Description Language (DDL) is a JSON file containing all the info it needs to create the demo, most importantly:

- General information about the demo (name, github URL, etc)
- What inputs are required
- What parameters can be specified by the user
- How the demo is run
- What outputs are shown to the user
- How the archive is populated

Are you ready? Start by copy-pasting the template DDL:
https://ipolcore.ipol.im/cp2/showDemo?demo_id=11111000004

Editing the DDL

DDL editor

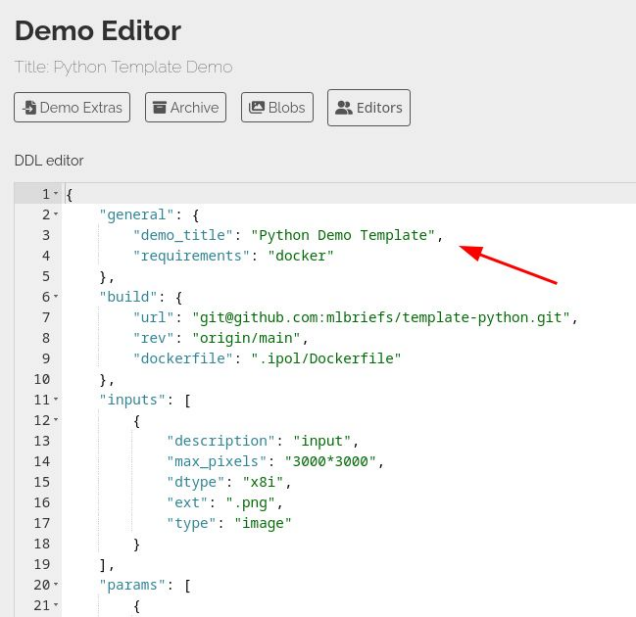
```
1   {
2       "general": {...},
9       "build": {...},
14      "inputs": [...],
23      "params": [...],
147     "run": "python $bin/src/run.py input_0.png -s $size $k $ky $kz $kx $kyx $kyz $kzz $kxz $kxxyy",
148     "results": [...],
166     "archive": {...}
191 }
```

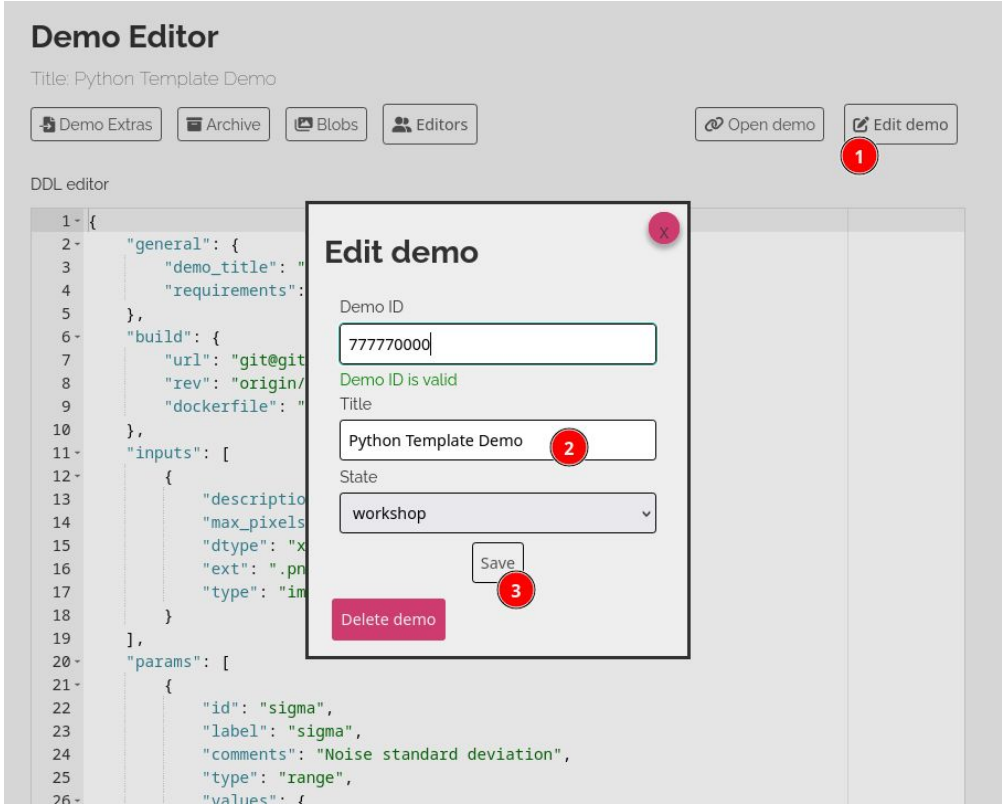
- Today: basic DDL editing to create a simple demo (but there are more options in the slides)
- Tomorrow, we will see the details and various possibilities

General, build

```
1  {
2     "general": {
3        "demo_title": "Python Demo Template",
4        "requirements": "docker"
5     },
6     "build": {
7        "url": "git@github.com:mlbriefs/template-python.git",
8        "rev": "origin/main",
9        "dockerfile": ".ipol/Dockerfile"
10    },
11    "inputs": [
12       {
13          "description": "input",
14          "max_pixels": "3000*3000",
15          "dtype": "x8i",
16          "ext": ".png",
17          "type": "image"
18       }
19    ],
20    "params": [
21       {
22          "id": "sigma",
23          "label": "sigma",
24          "comments": "Noise standard deviation",
25          "type": "range",
26          "values": {
27             "default": "30",
28             "max": 300,
29             "min": 0,
30             "step": 0.1
31          }
32       }
33    ],
34    "results": [
35       {
```

Demo title: edit in both places





Input

- relative to /workdir/exec/ (current working directory of the process)
- named sequentially , retrieve the filenames with $input_0, $input_1 (if any), and so on
- three supported types:
    - “image”: images (8bits)
        - can be resized by the system if too large (“max_pixels”)
    - “video”: video file format (experimental support)
    - “data”: everything else
    - “ext” defines how the file will be renamed by the system, eg:
      the user upload a file “mydata.txt”
      in the DDL: “ext”: “csv”
      at the start of the execution, the file will be named “input_0.csv”
      (but the content is untouched)
    - No format check for the data type: verify yourself that the user sent the correct formatting

Input: examples

```
15      {
16        "description": "input",
17        "dtype": "x8i",
18        "ext": ".png",
19        "max_pixels": "10000*10000",
20        "type": "image"
21      }
```

```
13  "inputs": [
14    {
15      "description": "Time series to analyse",
16      "ext": ".csv",
17      "type": "data",
18      "max_weight": "10*1024*1024"
19    }
20  ],
```

```
"inputs": [
    {
        "description": "input1",
        "max_pixels": "1600*1200",
        "dtype": "3x8i",
        "ext": ".png",
        "type": "image",
        "max_weight": "10* 1024 *1024"

    },
    {
        "description": "input2",
        "max_pixels": "1600*1200",
        "dtype": "3x8i",
        "ext": ".png",
        "type": "image",
        "max_weight": "10* 1024 *1024"

    }
],
```

Parameters

Figure 2: Selection collapsed example. In this case, the selection offers five options to choose.

Figure 3: Radio buttons example. The label description is Mode and the parameter offers two radio buttons. The vertical option is disabled.

Figure 5: Checkbox example. This can be used in the demos that need to activate or not an option.

Parameters

Figure 1: Range type example. It shows a slider with values from 0.02 to 0.2.

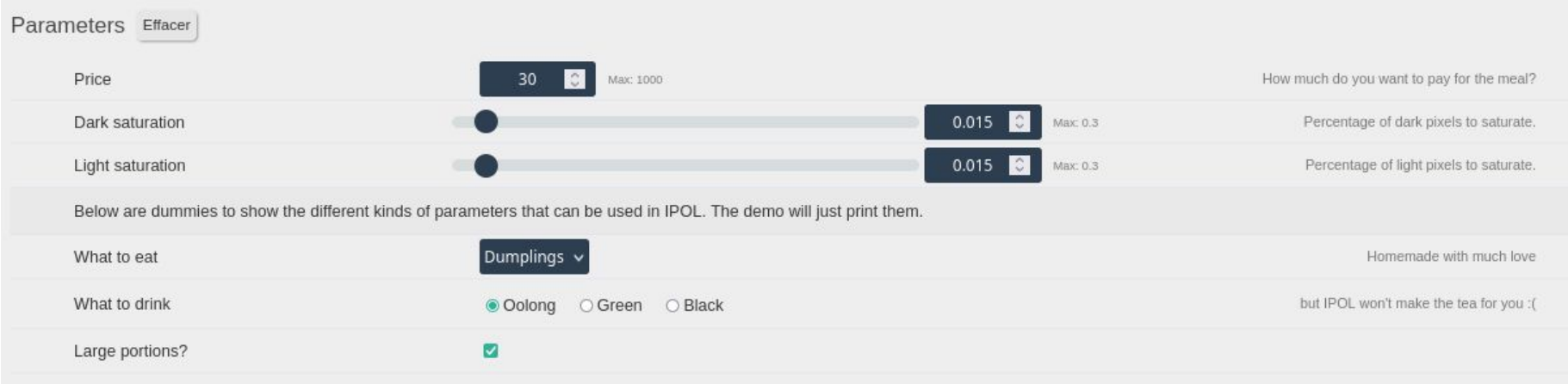
Figure 6: Numeric example. The label explains that the sliders below represent matrix values according to the image depicted in the label.

Figure 7: Text example. The user can write some text as parameter for the demo.

Parameters

Check the example demo:
https://ipolcore.ipol.im/demo/clientApp/demo.html?id=11111000002

And its corresponding DDL
https://ipolcore.ipol.im/cp2/showDemo?demo_id=11111000002



Parameters

```
"params": [
  {
      "id": "price",
      "type": "numeric",
      "label": "Price",
      "comments": "How much do you want to pay for the meal?",
      "values": {
          "min": 0,
          "max": 1000,
          "default": 30
      }
  },
  {
    "id": "s0",
    "label": "Dark saturation",
    "comments": "Percentage of dark pixels to saturate.",
    "type": "range",
    "values": {
      "default": 0.015,
      "max": 0.3,
      "min": 0,
      "step": 0.001
    }
  },
  {
    "id": "s1",
    "label": "Light saturation",
    "comments": "Percentage of light pixels to saturate.",
    "type": "range",
    "values": {
      "default": 0.015,
      "max": 0.3,
      "min": 0,
      "step": 0.001
    }
  },
```

```
  {
      "type": "label",
      "label": "Below are dummies to show the different kinds."
  },
  {
      "id": "food",
      "type": "selection_collapsed",
      "label": "What to eat",
      "comments": "Homemade with much love",
      "values": {
          "Soup": "soup",
          "Dumplings": "dumplings"
      },
      "default_value": "dumplings"
  },
  {
      "id": "drink",
      "type": "selection_radio",
      "label": "What to drink",
      "comments": "but IPOL won't make the tea for you :(",
      "values": {
          "Oolong": "oolong",
          "Green": "green",
          "Black": "black"
      },
      "default_value": "oolong"
  },
  {
      "id": "size",
      "type": "checkbox",
      "label": "Large portions?",
      "comment": "of course !",
      "default_value": "False"
  }
],
```

Run

Make sure your code can be executed from anywhere on the filesystem:

```
[user@laptop:~/myproject]$ python main.py
...
[user@laptop:~/myproject]$ cd /tmp
[user@laptop:/tmp]$ python ~/myproject/main.py
...
```

Instead of

```
torch.load('weights.pth')
```

use

```
ROOT = os.path.dirname(os.path.realpath(__file__))
torch.load(os.path.join(ROOT, 'weights.pth'))
```

or expose the weights as parameter.

In IPOL, the code is in $bin, but the execution is from somewhere else: call, e. g.,

“python $bin/main.py”

Instead of

"python main.py"

Run

- Inputs are named $input_0 (first input), $input_1 (second input, if any), and so on
- A parameter in the DDL with id "foo" can be called with $foo
- For the template:

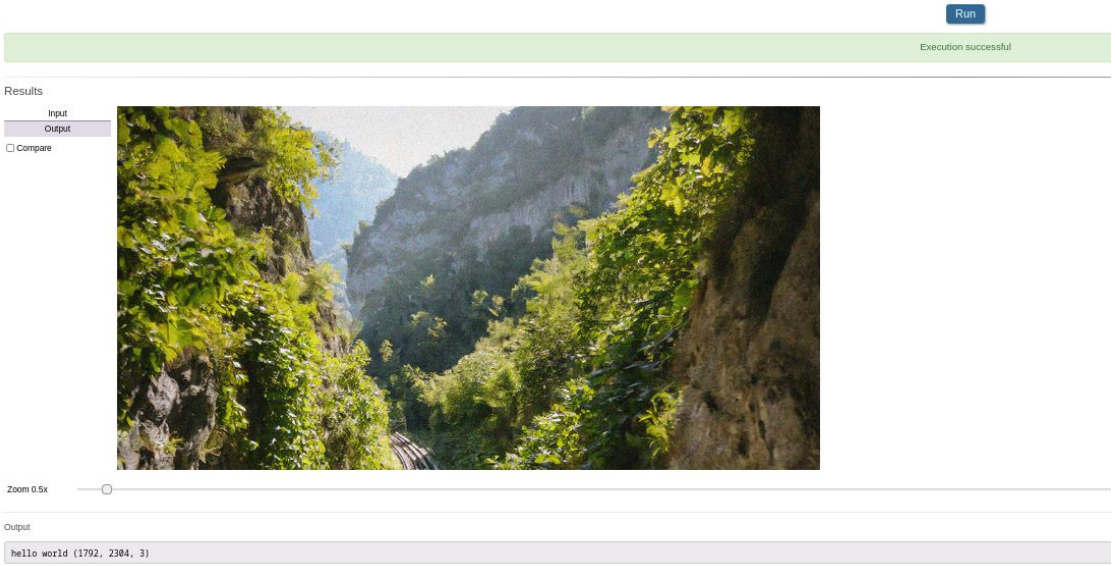
```
62    },
63    "run": "python $bin/main.py --input $input_0 --sigma $sigma --output output.png"
```

Results

- should be saved next to the inputs
- save static plots as images and show them with “type”: “gallery”
  - One gallery can show multiple images
  - To show a single image, we still use a gallery
- save texts to plain files and show them with “type”: “text_file”
- Save other visualizations to HTML and show them with “html_file”
  - A lot of possibilities there (show pandas tables, interactive plots, etc): tomorrow!

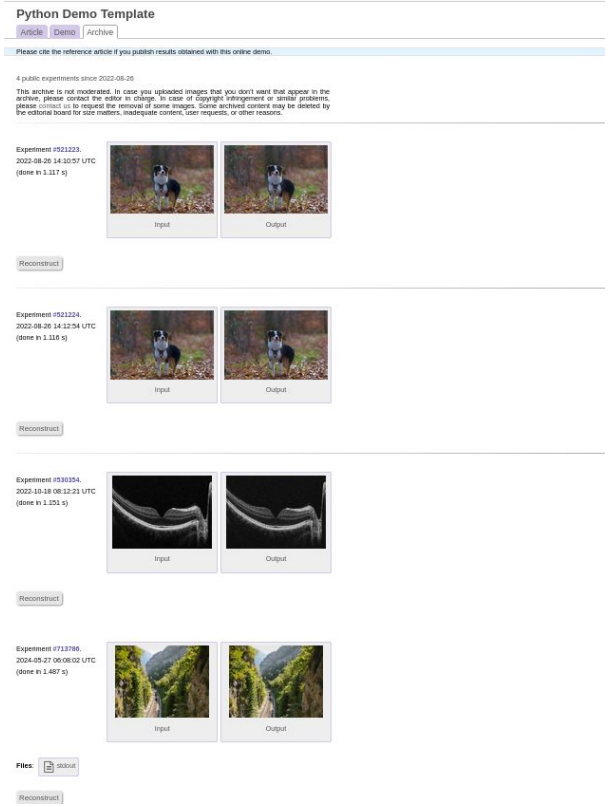
Image and text results

```
34  "results": [
35     {
36        "type": "gallery",
37        "contents": {
38           "Input": {
39              "img": "input_0.png"
40           },
41           "Output": {
42              "img": "output.png"
43           }
44        }
45     },
46     {
47        "contents": "stdout.txt",
48        "label": "<p>Output</p>",
49        "type": "text_file"
50     }
51  ],
```



Archive: not today!

```
52  "archive": {
53     "enable_reconstruct": true,
54     "files": {
55        "input_0.png": "Input",
56        "output.png": "Output",
57        "stdout.txt": "stdout"
58     },
59     "info": {
60        "run_time": "run time"
61     }
62  },
```



Demo creation process

Demos | Templates | Status | Quentin Bammey

Demo Editor

Title: Python Template Demo

Demo Extras | Archive | Blobs | Editors | Open demo | Edit demo

DDL editor

```
20  "params": [
21     {
22        "id": "sigma",
23        "label": "sigma",
24        "comments": "Noise standard deviation",
25        "type": "range",
26        "values": {
27           "default": "30",
28           "max": 300,
29           "min": 0,
30           "step": 0.1
31        }
32     }
33  ],
34  "results": [
35     {
36        "type": "gallery",
37        "contents": {
38           "Input": {
39              "img": "input_0.png"
40           },
41           "Output": {
42              "img": "output.png"
43           }
44        }
45     },
46     {
47        "contents": "stdout.txt",
48        "label": "<p>Output</p>",
49        "type": "text_file"
50     }
51  ],
52  "archive": {
53     "enable_reconstruct": true,
54     "files": {
55        "input_0.png": "Input",
56        "output.png": "Output",
57        "stdout.txt": "stdout"
58     },
59     "info": {
60        "run_time": "run time"
61     }
62  },
63  "run": "python $bin/main.py --input $input_0 --sigma $sigma --output output.png"
64 }
```

Save DDL | DDL already saved. | DDL History | White Theme | Dark Theme

SSH public key: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILVYJrAWQoUlvRxbqzrbMmOM48UyKWKk/JmgtZOxwh2O | Copy key | Reset key

Demo creation process

Demos | Templates | Status | Quentin Bammey

Demo Editor

Title: Python Template Demo

Demo Extras | Archive | Blobs | Editors | Open demo | Edit demo

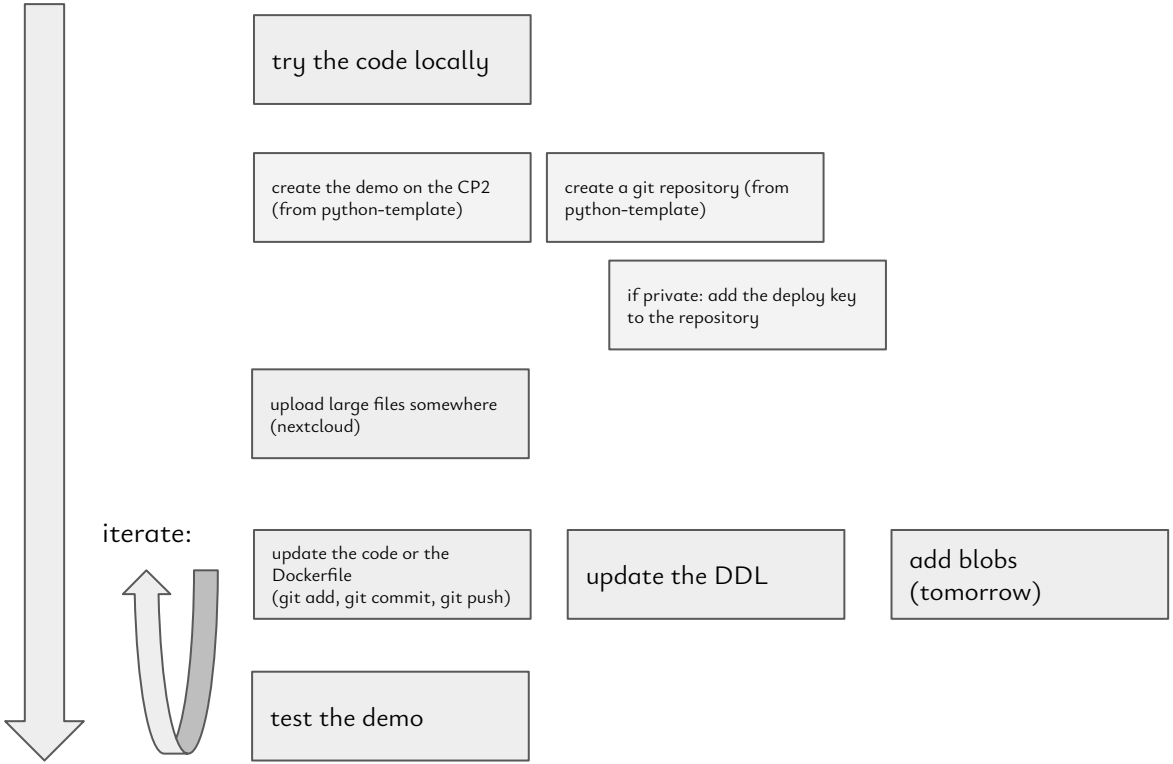
DDL editor

```
20  "params": [
21     {
22        "id": "sigma",
23        "label": "sigma",
24        "comments": "Noise standard deviation",
25        "type": "range",
26        "values": {
27           "default": "30",
28           "max": 300,
29           "min": 0,
30           "step": 0.1
31        }
32     }
33  ],
34  "results": [
35     {
36        "type": "gallery",
37        "contents": {
38           "Input": {
39              "img": "input_0.png"
40           },
41           "Output": {
42              "img": "output.png"
43           }
44        }
45     },
46     {
47        "contents": "stdout.txt",
48        "label": "<p>Output</p>",
49        "type": "text_file"
50     }
51  ],
52  "archive": {
53     "enable_reconstruct": true,
54     "files": {
55        "input_0.png": "Input",
56        "output.png": "Output",
57        "stdout.txt": "stdout"
58     },
59     "info": {
60        "run_time": "run time"
61     }
62  },
63  "run": "python $bin/main.py --input $input_0 --sigma $sigma --output output.png"
64 }
```

Save DDL | DDL already saved. | DDL History | White Theme | Dark Theme

SSH public key: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILVYJrAWQoUlvRxbqzrbMmOM48UyKWKk/JmgtZOxwh2O | Copy key | Reset key

Demo Creation Process



What's next?

Tomorrow: how to create a fully fledged demo

Now: lunch in hall Emmy Noether

This afternoon: work in the workshop rooms

Try to:

- Copy the template and create a new demo with it
- Your goal: add your code to it, and make it run without errors
- You do not know about most inputs, parameters and outputs yet, or using large files:
  - Use dummy inputs (e. g. random noise signals), simple outputs (e. g. stdout), only 1 or 2 simple parameters
  - Comment the weight loading part
- We are here to help you!