

From nanoparticles to neuron segmentation
Accelerating research through digital technology



**Open and Scalable Image and Data Processing Platform to
accelerate your microscopy workflows**

apeer

From image to information

Dr. Sebastian Rhode

Product Owner, Technology Center Software & Digital Solutions, Munich

The challenges of a microscopist / researcher

Select quotes from user interviews



Researchers with no coding skills cannot use algorithms written by others.

Different students report different results on the same dataset.

If my post doc leaves I'm left with valuable but unusable code.

We use different pieces of software with no connectivity between them.

Reproducing published results is a nightmare.



Leveraging digital technologies to accelerate research and improve repeatability

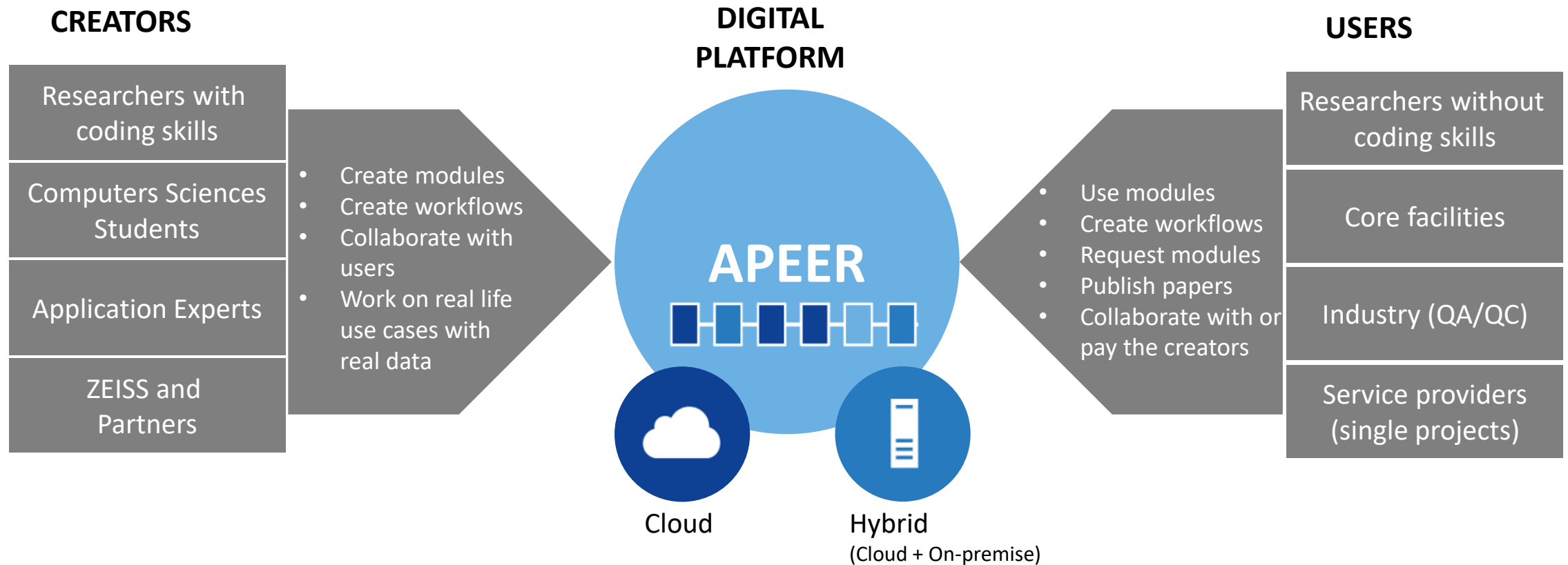


APEER – online platform for microscopists




Goal:

“Drive efficiency in research through a digital platform to easily create, execute and share solutions to various microscopy user challenges.”



APEER – online platform for microscopists






Content ▾Resources ▾DocsBlog


Log inSign up

Your customized image processing solution


Seamlessly combine discrete imaging modules into a meaningful workflow.




Your
Microscope




Your
images




Python
by Frank




Java
by 3rd party




Add new
module




ImageJ
by You




Your
Results

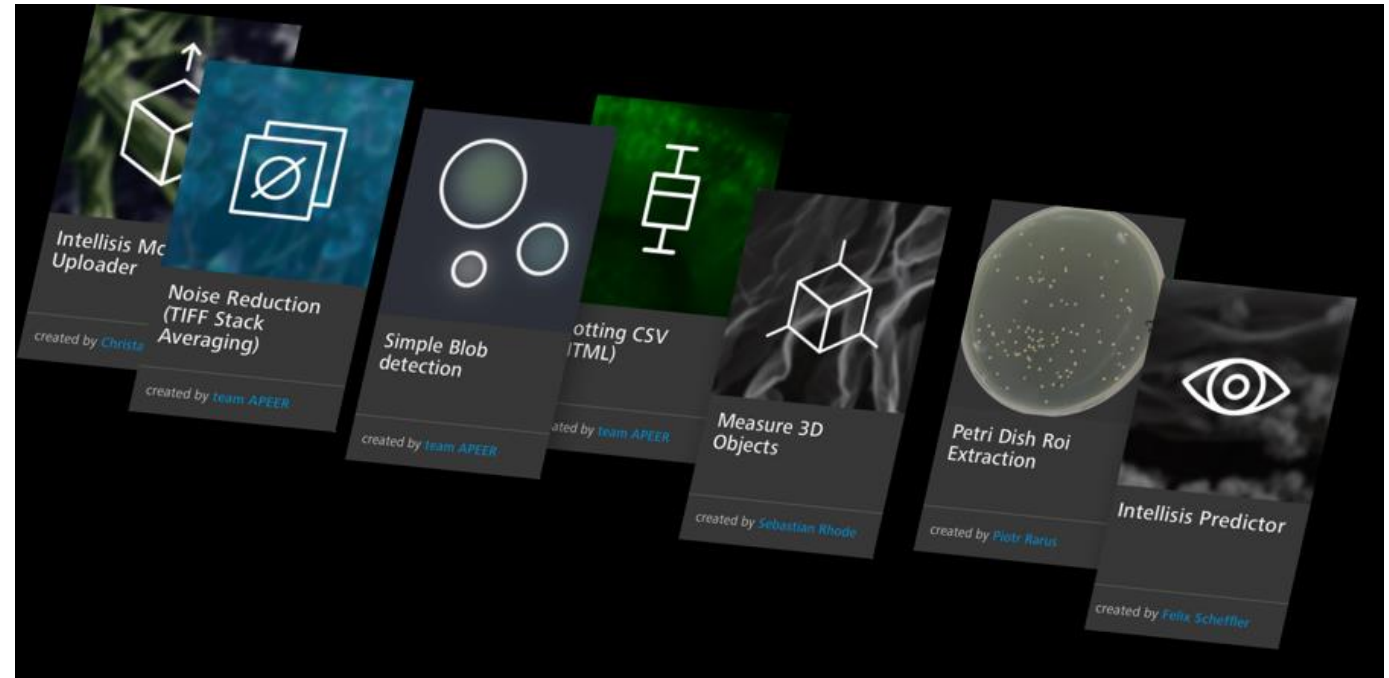
 running on cloud

or

 running locally



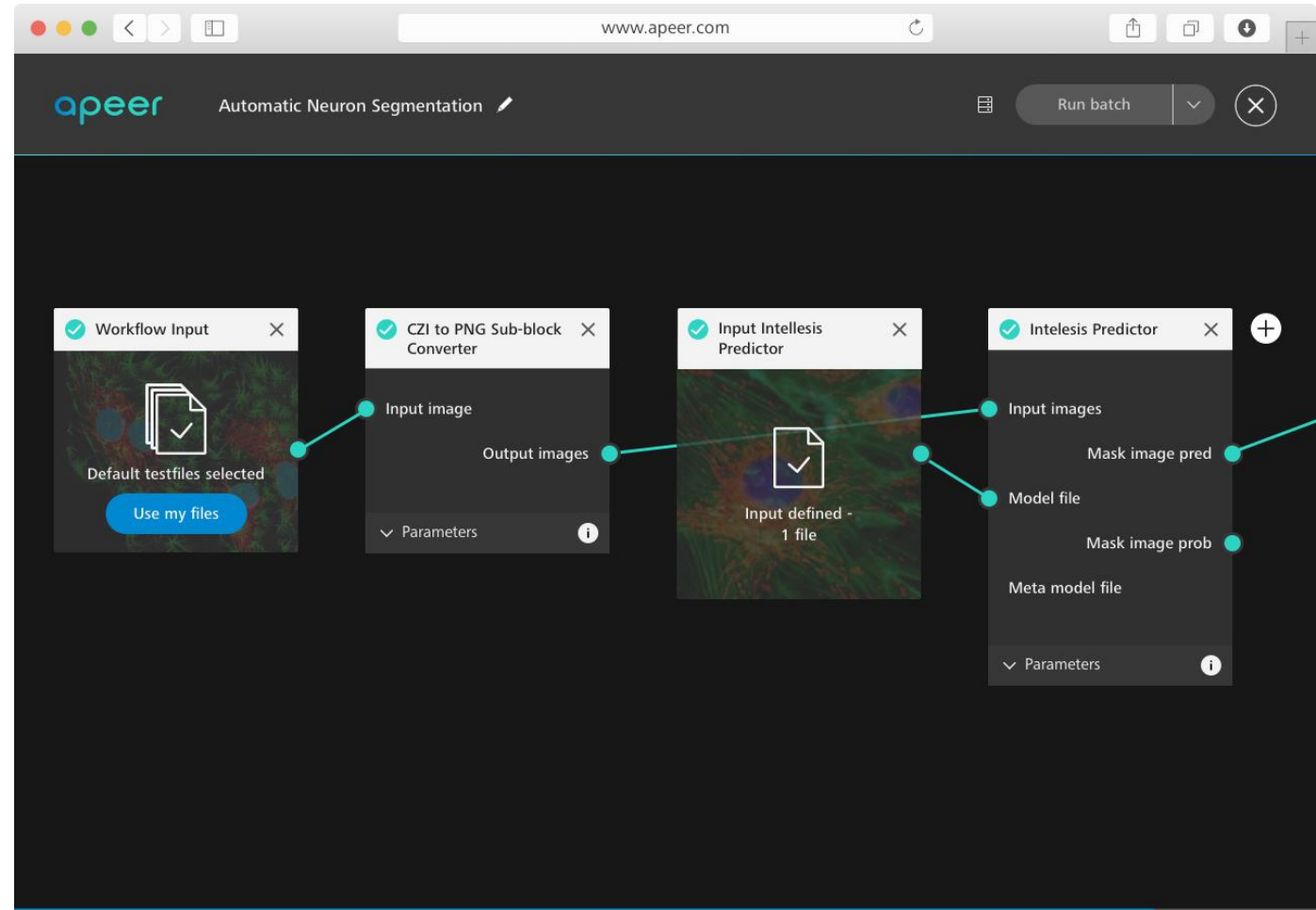
Create your own imaging modules using the programming language of your choice, as we use Docker, or build on others' work.



APEER – Create, Combine and Collaborate



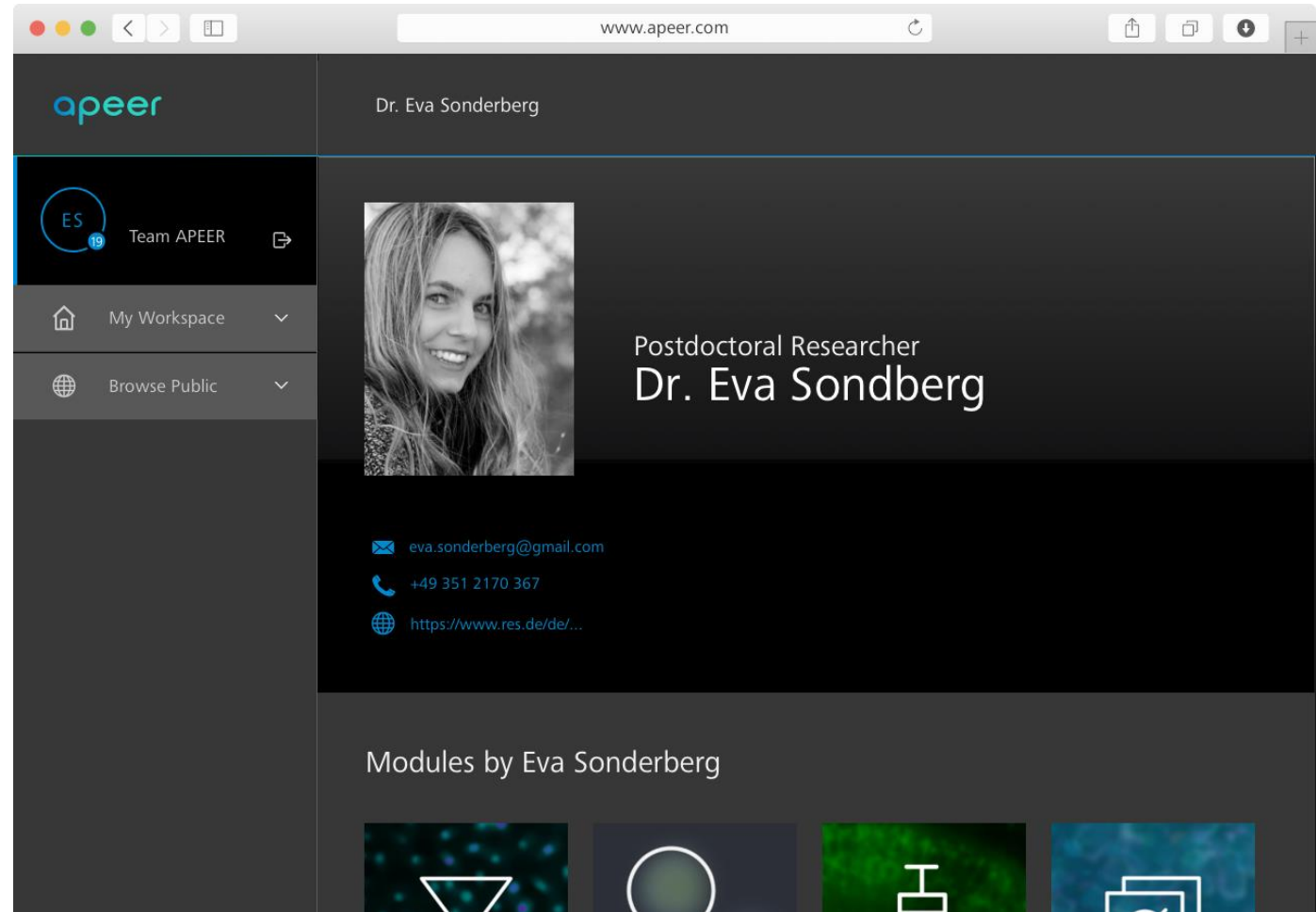
Combine private and public image processing modules into customized solutions. Automate the workflows for increased productivity and reproducibility.



APEER – Create, Combine and Collaborate



Collaborate - Save valuable research time by not reinventing code. Share work with your team or with the community. Work from anywhere using any device.



APEER enables you to **create & customize image and data processing solutions for your specific job** to be done.

You can automate tasks by creating modular **workflows** by combining **modules**. Such workflows can be run either in the cloud or locally depending on your preferences.

You can **save time** and conserve knowledge by **using shared modules & workflows**.

By **publishing workflows** together with your results you can encourage other to **reproduce** your findings easily with their data using the exact same software environment with the need to install anything

APEER – Public Modules



Public Modules ?

go to [apeer.com](#)

Sebastian Rhode

My Workspace

Browse Public

Public Modules

Public Workflows

<<

Collapse

Sort by Creation Date

Filter by Category

Search

☐ Compliant modules

Version 1

Muscle Motion Quantification

Created by Team APEER

Version 1

Ilastik Object Classification

Created by Team APEER

Version 1

Color space converter RGB2CieLab

Created by David Dang

Version 1

rgb-to-lab

Created by Juan Cruz Martinez

Version 1

Ilastik Pixel Classification

Created by Team APEER

Version 1

Image Colorization

Created by Thomas Irmer

Version 1

cli-j-apeer-template

Created by Robert Haase

Version 1

2D Segmentation Trainer

Created by Team APEER

Contact us

APEER - The journey from image to information



1

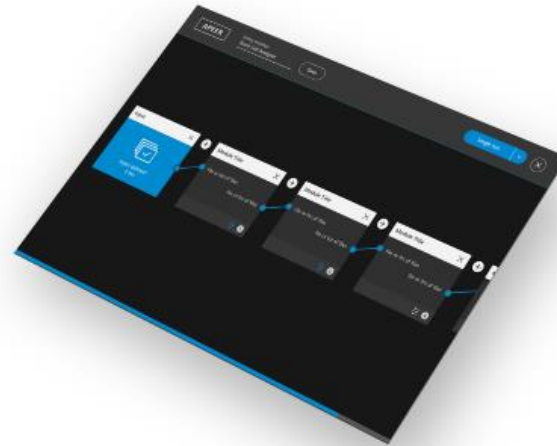
Pick modules



- A module is a piece of software (Docker™) that performs a single task (e.g. file conversion or segmentation)
- Anyone can create a module with a little bit of coding knowledge, **in any language**.

2

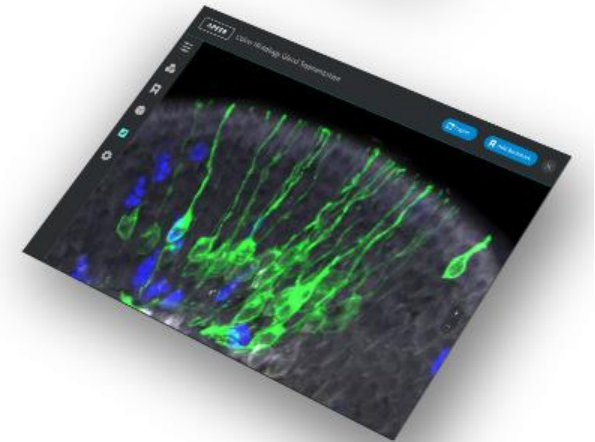
Build the workflow



- A workflow is an arrangement of modules in a specific way customized for a specific job.
- Anyone create a workflow, **no need for coding knowledge**.

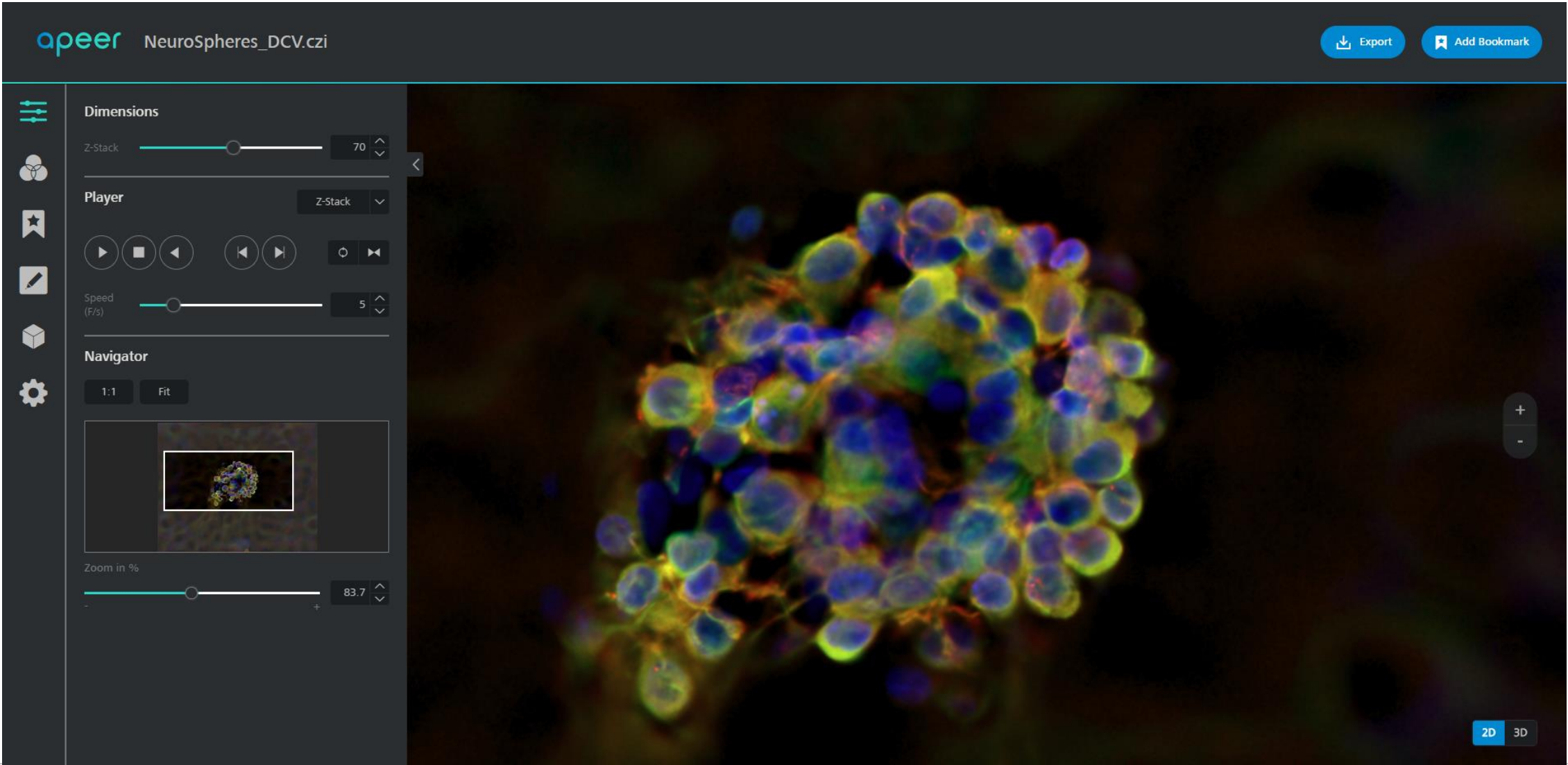
3

Run the workflow

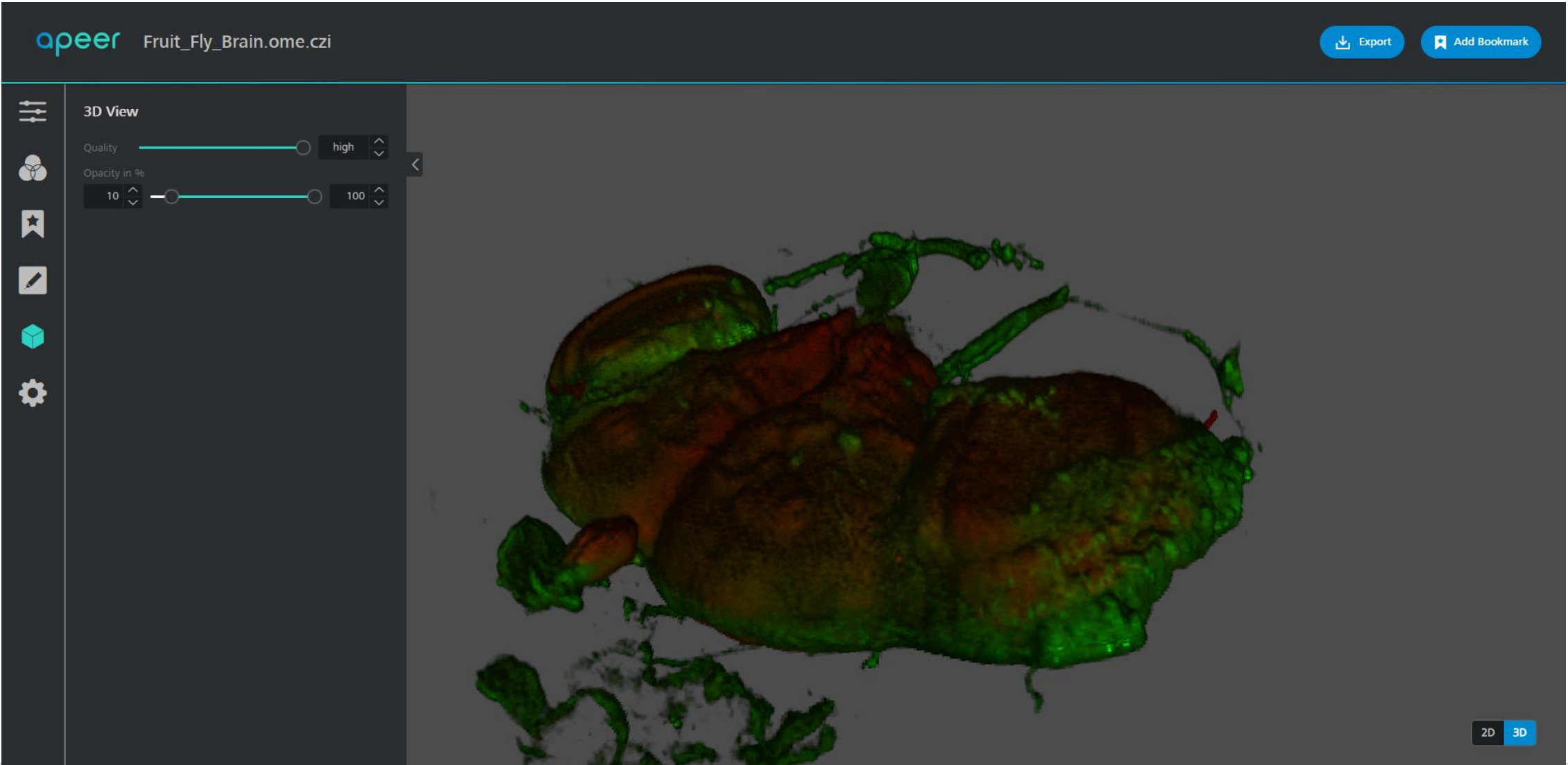


- Run a workflow by just clicking a button.
- Batch runs can be started by selecting multiple input files.
- View and Share the results.

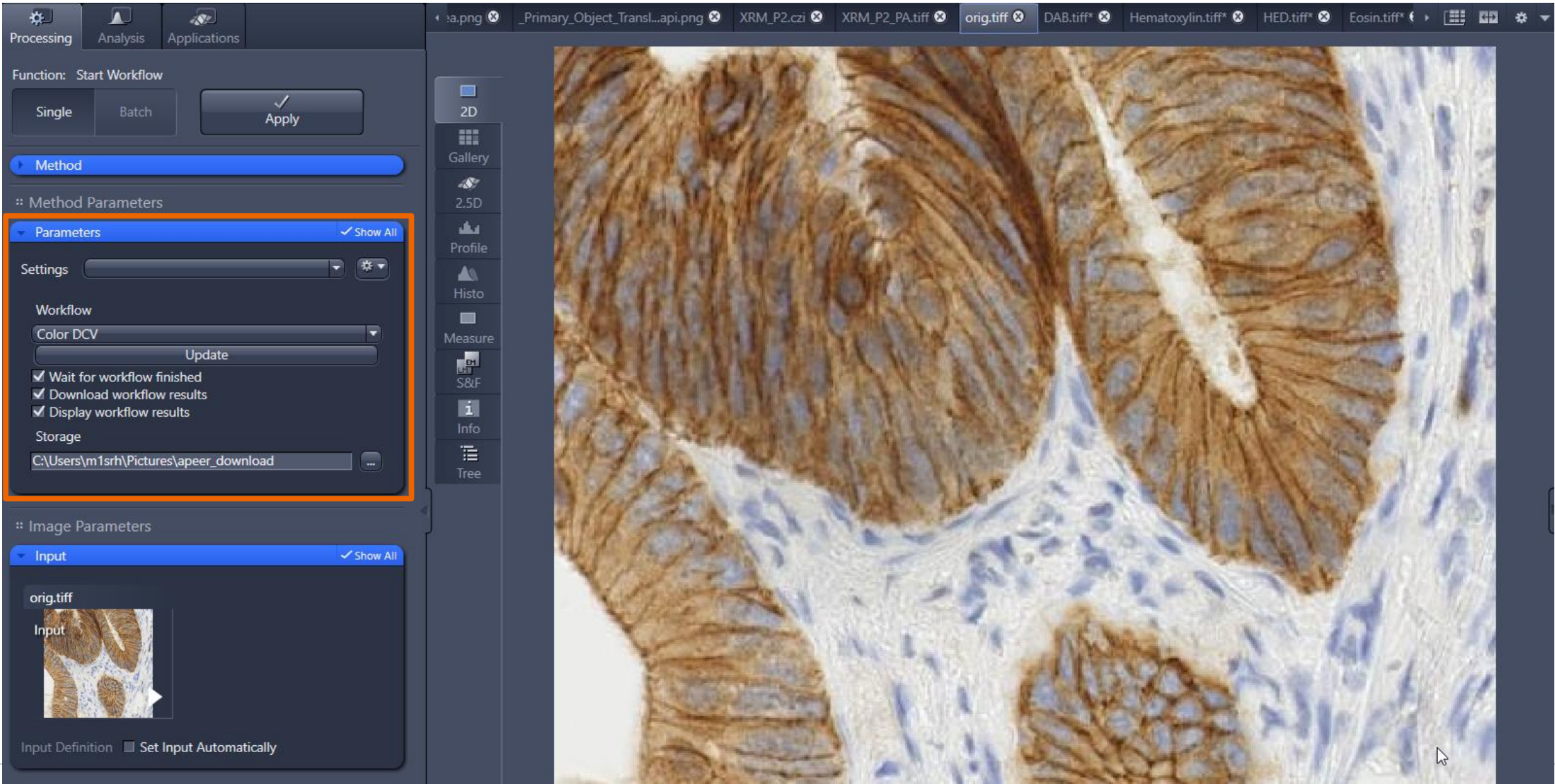
APEER WebViewer - Visualize your images from anywhere using any device



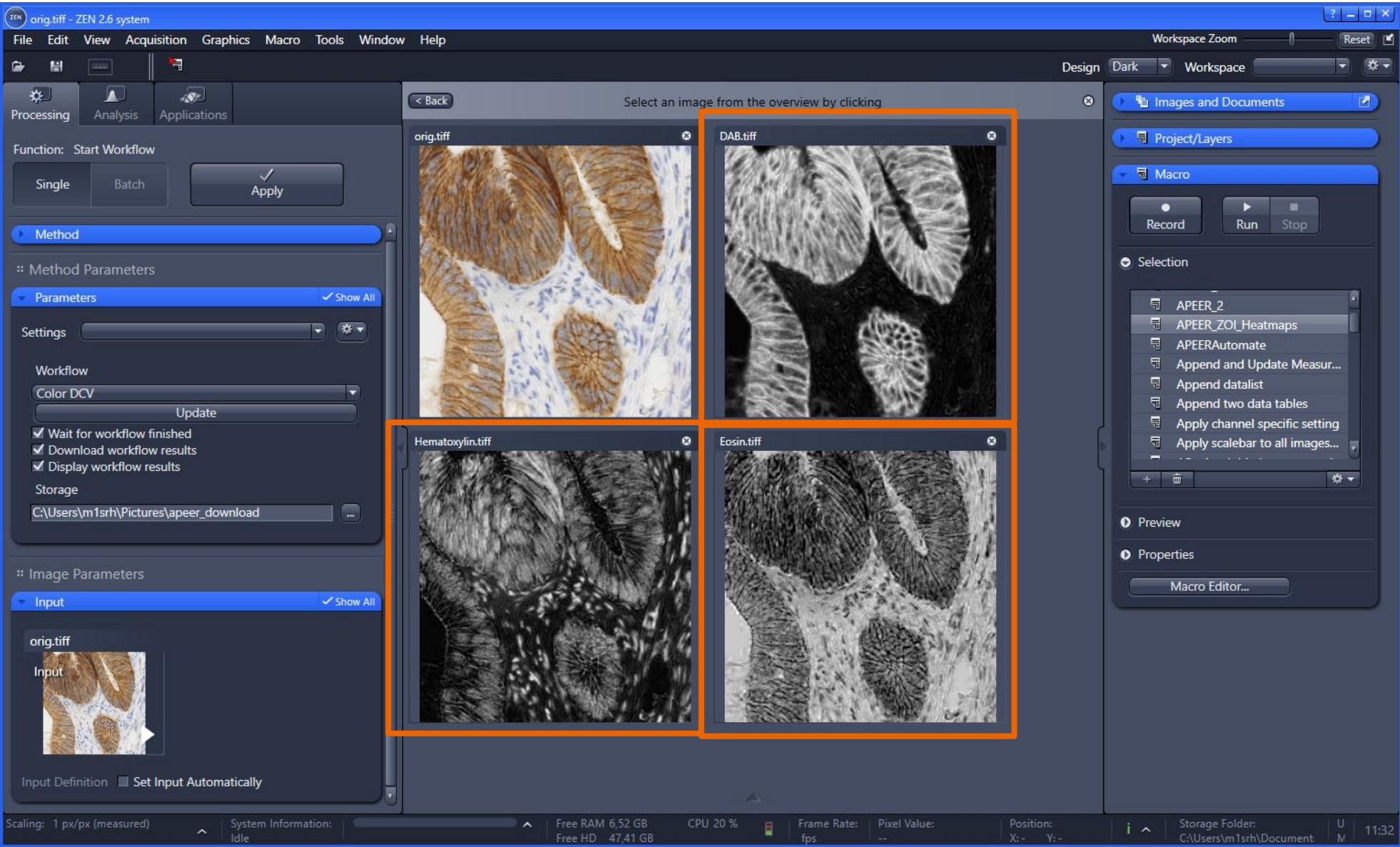
APEER WebViewer - Visualize your images from anywhere using any device



ZEN Blue APEER Connector – Run WorkFlows from “inside” ZEN



ZEN Blue APEER Connector – Run WorkFlows from “inside” ZEN



You have full control over your modules and workflows



The screenshot displays the Apeer platform's module management interface. On the left is a dark sidebar with navigation icons: a square with 'A', a circle with 'SR', a house, a cube, a folder, a bar chart, and a globe. The main content area has a dark background. At the top, a 'Module Sharing' section includes a 'Manage Shares' button highlighted with a yellow box. Below this, a 'Build Status: succeeded' message is shown. A text block explains the use of Git for hosting code, with a 'Module Code ^' link. A yellow box highlights the 'Git Credentials' section, which contains a 'sebi06' username, a 'Your Password' field, an 'Update' button, and a 'Git Url' field with the value 'https://www.appeer.com/_git/zen_heatmap_dd869742-65c2-42c5'. Below this, three numbered steps are listed: '01. Module Properties', '02. Code Files', and '03. Docker File'. A yellow box highlights a row of links: 'Generate Module Specifications', 'Documentation', 'Documentation (Python)', 'Documentation (Java)', and 'Documentation (ImageJ Macro)'. At the bottom, a 'Download Code' section features a 'Download Code' button highlighted with a yellow box.

Module Sharing
Not shared yet.
Share your Module with your peers. They can use it as part of their workflow.

Build Status: succeeded
Built at 2019-Jan-22 20:34:03

We are using Git to host the code of your Module on the platform. In order to get your code into the platform please provide the following 3 things: 1. The JSON Specification 2. Your Code 3. The Dockerfile. Please clone your Git repository by using the URL below. For more information please refer to our [Wiki](#) or feel free to [contact us](#) anytime.

Git Credentials
sebi06 Your Password **Update**

Git Url
https://www.appeer.com/_git/zen_heatmap_dd869742-65c2-42c5

01. Module Properties
Define your module's in- and outputs as well as the required resources.

02. Code Files
Adapt your Code referring to the aforementioned JSON Specification.

03. Docker File
Provide a Dockerfile that contains the necessary environment for your code.

[Generate Module Specifications](#) [Documentation](#) [Documentation \(Python\)](#) [Documentation \(Java\)](#) [Documentation \(ImageJ Macro\)](#)

Download Code
Your Module's code is available for everyone to download.

- Manage the shares of your modules and workflows
- Use GIT to manage your module code
- Generate modules specs and UI with graphical editor
- Use forum and wiki to find information, tutorials and code examples
- Allow downloading the module code for full transparency

Online Documentation, Tutorials and Videos are available



opeer Docs

Q Search...

APEER Docs

About Apeer

GETTING STARTED

APEER Terminology 101 (5 min)

The APEER Architecture (5 min)

What You'll Need (10 min)

Hello, World! (10 min)

Your First Workflow (10 min)

Editing an existing module (30 min)

Python Project Structure

TUTORIALS

Python OME-TIFF Example (CPU)

Python Tensorflow Example (GPU)

Fiji Python Scripting

ImageJ Example

Java Example



Powered by GitBook

Your First Workflow (10 min)

This step by step tutorial is teaching you how to create a simple workflow running on the APEER platform using a default Python template.



As we will create a workflow in this tutorial make sure that you've ...

- ✓ ... a basic understanding what [modules](#) and [workflows](#) are
- ✓ ... successfully registered an account on [APEER](#)
- ✓ ... created a first module (use at least the [default Python template](#))

CONTENTS

Step #1: Create new workflo...

Step #2: Run your created wo...

Step #3 View the results of y...

You couldn't find all the infor...

Step #1: Create new workflow with one module

- Go to "My Workspace" and click "Create new Workflow"
- Click the "+" button and type "OME-TIFF-Converter" and click "Add" (This will convert different image files to a standardized OME-TIFF format)
- Click again the "+" button and type "Python"
- Choose the previously created "Python example" module and add it as well
- Enter the name for the workflow e.g. "My first workflow"

Workflow Editor

My first workflow

Save

Module UI – Use the built-in UI editor



Sebastian Rhode

My Workspace

My Modules (18)

My Workflows (28)

My Files (66)

My Results (71)

Local Execution

Browse Public

01.

Module Properties

Description

Overview

How to use

Module Sharing

Please enter a description

Please enter instructions

Not shared yet

Share your Module v

Build Status: success

We are using Git to

Specification 2. Your

contact us anytime.

Git Credentials

sebi06

Edit Module Properties

Define your Module Inputs and Outputs below. In the next step you can download your Module Specification File, which includes Information about Inputs & Outputs as well as Module Requirements. You can use this file for creating your Module Code.

PSF-Check

add Input

Original images

Red channel

Green channel

Blue channel

add Output

Tinted images

Original images

Input Type *

List of files

Supported File Type *

image

data

all

Other File Formats (e.g. *, html...)

Input Description

The images to be tinted

Next

Create Workflow

Publish Module

Manage Shares

Built at 2019-Feb-13 18:56:00

Module Code

8ffb-076f-4957-a16

3.

Docker File

Module UI – Use the built-in UI editor



Sebastian Rhode

My Workspace

My Modules (18)

My Workflows (28)

My Files (66)

My Results (71)

Local Execution

Browse Public

Description

Overview

Please enter a description

How to use

Please enter instructions

Module Sharing

Not shared yet

Share your Module with others

Build Status: successful

We are using Git to manage your code. If you have any changes, please push them to the repository. You can contact us anytime.

Git Credentials

sebi06

Create Workflow

Publish Module

Manage Shares

Built at 2019-Feb-13 18:56:00

Module Code

8ffb-076f-4957-a16

Edit Module Properties

Define your Module Inputs and Outputs below. In the next step you can download your Module Specification File, which includes Information about Inputs & Outputs as well as Module Requirements. You can use this file for creating your Module Code.

PSF-Check

add Input

add Output

Original images

Red channel

Green channel

Blue channel

Tinted images

Module Requirements

GPU Usage

Internet access

=

Module Specification File (.JSON)

Previous

Download

01.

Module Properties

02.

Code Files

03.

Docker File

Module Creation – Code is stored inside a Git Repository

Use IDE of your choice



```
module_specification.json - srh_apeer_sdk_test_362af2b3-aeed-4de3-a857-57a70571a147 - Visual Studio Code

1 {
2   "spec": {
3     "inputs": {
4       "ihc_image": {
5         "type": "file"
6       }
7     },
8     "outputs": {
9       "processed_images": {
10        "type": "list[file]"
11      }
12    },
13    "ui": {
14      "inputs": {
15        "ihc_image": {
16          "index": 1,
17          "label": "IHC Image",
18          "widget": "textbox"
19        }
20      },
21      "outputs": {}
22    }
23  }
24 }

Allan Menoret, 7 months ago • Initial

color_dcv.py x

You, an hour ago | 2 authors (Rhode and others)
1 from skimage import color, img_as_float, img_as_uint, io, exposure
2 from skimage.color import rgb2hed
3 import numpy as np
4 import tifffile as tf
5
6
7 def run(original_image_path):
8
9     outputPathList = []
10
11     # Read original image
12     ihc_rgb = io.imread(original_image_path)
13
14     # Process
15     ihc_hed = rgb2hed(ihc_rgb)
16     ihc_h = exposure.rescale_intensity(ihc_hed[:, :, 0], out_range=(0, 255))
17     ihc_e = exposure.rescale_intensity(ihc_hed[:, :, 1], out_range=(0, 255))
18     ihc_d = exposure.rescale_intensity(ihc_hed[:, :, 2], out_range=(0, 255))
19
20     images = [ihc_h, ihc_e, ihc_d]
21     outnames = ['Hematoxylin.tiff', 'Eosin.tiff', 'DAB.tiff']
22
23     for img, out in zip(images, outnames):
24
25         outputPathList.append(out)
26         tf.imwrite(out, np.int16(img))
27
28     tf.imwrite('HED.tiff', np.float16(ihc_hed))
29     outputPathList.append('HED.tiff')
30
31     print(outputPathList)
32
33     # Return apeer output values as dictionary
34     return {'processed_images': outputPathList}
35

apeer_main.py x

You, 6 hours ago | 3 authors (Rhode and others)
1 from apeer_dev_kit import adk
2 import color_dcv
3
4 if __name__ == "__main__":
5
6     inputs = adk.get_inputs()
7     outputs = color_dcv.run(inputs['ihc_image'])
8     adk.set_file_output('processed_images', outputs['processed_images'])
9     adk.finalize()
10

requirements.txt x

You, an hour ago | 2 authors (Thomas Irmer and others)
1 apeer-dev-kit>1,<2
2 cloudpickle==0.6.1
3 dask==0.20.2
4 decorator==4.3.0
5 networkx==2.2
6 numpy==1.15.4
7 Pillow==5.3.0
8 PyWavelets==1.0.1
9 scikit-image==0.14.1
10 six==1.11.0
11 toolz==0.9.0
12 scipy==1.1.0
13 tifffile==2018.11.28
```


Module Creation - GIT

Specify module UI and other requirements (in this case Python modules)



The screenshot shows a code editor with two files open: `module_specification.json` and `requirements.txt`. The editor has a dark theme and a sidebar on the left with various icons. The status bar at the bottom indicates the current file is `module_specification.json`, line 24, column 2, with 4 spaces, UTF-8 encoding, CRLF line endings, and JSON format.

```
module_specification.json
1  {
2    "spec": {
3      "inputs": {
4        "ihc_image": {
5          "type:file": {}
6        }
7      },
8      "outputs": {
9        "processed_images": {
10         "type:list[file]": {}
11       }
12     },
13   },
14   "ui": {
15     "inputs": {
16       "ihc_image": {
17         "index": 1,
18         "label": "IHC Image",
19         "widget:textbox": {}
20       }
21     },
22     "outputs": {}
23   }
24 }
```


requirements.txt

```
1  apeer-dev-kit>=1,<2
2  cloudpickle==0.6.1
3  dask==0.20.2
4  decorator==4.3.0
5  networkx==2.2
6  numpy==1.15.4
7  Pillow==5.3.0
8  PyWavelets==1.0.1
9  scikit-image==0.14.1
10 six==1.11.0
11 toolz==0.9.0
12 scipy==1.1.0
13 tifffile==2018.11.28
```

Module Creation – APEER-SDK

Specify module UI and other requirements (in this case Python modules)





[Help](#) [Donate](#) [Log in](#) [Register](#)

apeer-dev-kit 1.0.9

```
pip install apeer-dev-kit
```



 Latest version

Last released: Dec 21, 2018

Development kit for creating modules on apeer

Navigation

 Project description

 Release history

 Download files

Project description

APEER Python SDK aka (ADK) is a Python library for reading inputs and writing outputs of APEER(<https://www.apeer.com>) modules. The ADK will take care of reading inputs from previous modules in APEER and writing your outputs in the correct format for the next module. This project is hosted at <https://github.com/apeer-micro/apeer-python-sdk> The documentation can be found at <https://github.com/apeer-micro/apeer-python-sdk/blob/master/README.md>

Project links

 Homepage

Module Creation – APEER-SDK on GitHub

Specify module UI and other requirements (in this case Python modules)



The screenshot displays the GitHub profile for APEER, a digital microscopy platform. The profile header includes the APEER logo, a description, and a verified website link. Below the header, navigation tabs for Repositories (6), Packages, People (10), Teams, and Projects are visible. A search bar and filters for repository type and language are present. The main content area lists several repositories, each with its name, description, language, license, star count, issue count, and update date. A sidebar on the right shows top languages and a list of contributors.

APEER
APEER is a digital microscopy platform, enabling you to create & customize for imaging tasks.
<https://www.apeer.com> Verified

Repositories 6 Packages People 10 Teams Projects

Find a repository... Type: All Language: All New

apeer-ometiff-library
Easily read and write ome-tiff images in python
Python 1 star 3 5 (1 issue needs help) 1 Updated yesterday

apeer-module-debugger
JavaScript 2 star 4 3 0 Updated 18 days ago

apeer-yeoman-generator
JavaScript MIT 0 star 1 0 0 Updated on 9 Sep

apeer-matlab-sdk
MATLAB MIT 1 star 2 0 0 Updated on 21 Jun

apeer-python-sdk
python sdk apeer
Python MIT 0 star 7 0 0 Updated on 6 Jan

apeer-java-sdk
Java MIT 0 star 3 0 0 Updated on 19 Dec 2018

Top languages
JavaScript Python Java
MATLAB

People 10 >

Module Creation - GIT

Write your code and use the APEER-SDK (PyPi) to make life easy



The screenshot shows the Visual Studio Code editor interface with two Python files open. The left file, `color_dcv.py`, contains a `run` function that processes an image. The right file, `apeer_main.py`, shows the main execution logic using the `apeer_dev_kit` library.

```
color_dcv.py
1 from skimage import color, img_as_float, img_as_uint, io, exposure
2 from skimage.color import rgb2hed
3 import numpy as np
4 import tifffile as tf
5
6
7 def run(original_image_path):
8
9     outputPathList = []
10
11     # Read original image
12     ihc_rgb = io.imread(original_image_path)
13
14     # Process
15     ihc_hed = rgb2hed(ihc_rgb)
16     ihc_h = exposure.rescale_intensity(ihc_hed[:, :, 0], out_range=(0, 255))
17     ihc_e = exposure.rescale_intensity(ihc_hed[:, :, 1], out_range=(0, 255))
18     ihc_d = exposure.rescale_intensity(ihc_hed[:, :, 2], out_range=(0, 255))
19
20     images = [ihc_h, ihc_e, ihc_d]
21     outnames = ['Hematoxylin.tiff', 'Eosin.tiff', 'DAB.tiff']
22
23     for img, out in zip(images, outnames):
24
25         outputPathList.append(out)
26         tf.imwrite(out, np.int16(img))
27
28     tf.imwrite('HED.tiff', np.float16(ihc_hed))
29     outputPathList.append('HED.tiff')
30
31     print(outputPathList)
32
33     # Return apeer output values as dictionary
34     return {'processed_images': outputPathList}
35
```

```
apeer_main.py
1 from apeer_dev_kit import adk
2 import color_dcv
3
4 if __name__ == "__main__":
5
6     inputs = adk.get_inputs()
7     outputs = color_dcv.run(inputs['ihc_image'])
8     adk.set_file_output('processed_images', outputs['processed_images'])
9     adk.finalize()
10
```

Visual Studio Code status bar at the bottom shows: master* | Team | Python 3.7.0 64-bit (Anaconda3: conda) | Ln 10, Col 1 | Spaces: 4 | UTF-8 with BOM | CRLF | Python

Pricing

There will always be a free version - APEER Core. Due to our users' higher demand in storage and processing power, we are working on a subscription based Pro version, which will be rolled out soon.

Please do not hesitate to contact us at support@apeer.com

	APEER CORE Best package to get started for individuals	APEER PRO Fast processing and high storage	APEER ACADEMIC Hybrid solution with local processing and local data or individual cloud storage	APEER CUSTOM Get customized solutions for your image processing problem (industry and academia).
Number of users	1	1	10	customizable
Online storage	100 GB	1 TB	customizable	customizable
Local storage			✓	✓
Normal processing	✓			
Fast processing		✓	✓	✓
Queue priority		✓	✓	✓
E-Mail support	✓	✓	✓	✓
Community support	✓	✓	✓	✓
Phone support		✓	✓	✓
Personal advise support				✓
Build Modules / Workflows	✓	✓	✓	✓
Share with individuals	✓	✓	✓	✓
Share with subgroups			✓	✓
Get a customized Workflow				✓
	FREE	Coming soon	Coming soon	

