# Contributing to this project

Please take a moment to review this document in order to make the contribution

process easy and effective for everyone involved.

Following these guidelines helps to communicate that you respect the time of

the developers managing and developing this open source project. In return,

they should reciprocate that respect in addressing your issue or assessing

patches and features.

## Development environment

Next steps should work on clear Ubuntu 18.04.

- Install necessary dependencies:

```sh

sudo apt-get update && sudo apt-get --no-install-recommends install -y ffmpeg build-essential curl redis-server python3-dev python3-pip python3-venv python3-tk libldap2-dev libsasl2-dev

```

Also please make sure that you have installed ffmpeg with all necessary libav\* libraries and pkg-config package.

```sh

# Node and npm (you can use default versions of these packages from apt (8.\*, 3.\*), but we would recommend to use newer versions)

curl -sL https://deb.nodesource.com/setup\_12.x | sudo -E bash -

sudo apt-get install -y nodejs

# General dependencies

sudo apt-get install -y pkg-config

# Library components

sudo apt-get install -y \

libavformat-dev libavcodec-dev libavdevice-dev \

libavutil-dev libswscale-dev libswresample-dev libavfilter-dev

```

See [PyAV Dependencies installation guide](http://docs.mikeboers.com/pyav/develop/overview/installation.html#dependencies)

for details.

- Install [Visual Studio Code](https://code.visualstudio.com/docs/setup/linux#\_debian-and-ubuntu-based-distributions)

for development

- Install CVAT on your local host:

```sh

git clone https://github.com/opencv/cvat

cd cvat && mkdir logs keys

python3 -m venv .env

. .env/bin/activate

pip install -U pip wheel setuptools

pip install -r cvat/requirements/development.txt

pip install -r datumaro/requirements.txt

python manage.py migrate

python manage.py collectstatic

```

- Create a super user for CVAT:

```sh

$ python manage.py createsuperuser

Username (leave blank to use 'django'): \*\*\*

Email address: \*\*\*

Password: \*\*\*

Password (again): \*\*\*

```

- Install npm packages for UI and start UI debug server (run the following command from CVAT root directory):

```sh

npm install && \

cd cvat-core && npm install && \

cd ../cvat-ui && npm install && npm start

```

- Open new terminal (Ctrl + Shift + T), run Visual Studio Code from the virtual environment

```sh

cd .. && source .env/bin/activate && code

```

- Install following VS Code extensions:

- [Debugger for Chrome](https://marketplace.visualstudio.com/items?itemName=msjsdiag.debugger-for-chrome)

- [Python](https://marketplace.visualstudio.com/items?itemName=ms-python.python)

- [ESLint](https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint)

- [Stylelint](https://marketplace.visualstudio.com/items?itemName=stylelint.vscode-stylelint)

- [vscode-remark-lint](https://marketplace.visualstudio.com/items?itemName=drewbourne.vscode-remark-lint)

- [licenser](https://marketplace.visualstudio.com/items?itemName=ymotongpoo.licenser)

- [Trailing Spaces](https://marketplace.visualstudio.com/items?itemName=shardulm94.trailing-spaces)

- Reload Visual Studio Code from virtual environment

- Select `server: debug` configuration and start it (F5) to run REST server and its workers

You have done! Now it is possible to insert breakpoints and debug server and client of the tool.

### Note for Windows users

You develop CVAT under WSL (Windows subsystem for Linux) following next steps.

- Install WSL using [this guide](https://docs.microsoft.com/ru-ru/windows/wsl/install-win10).

- Following this guide install Ubuntu 18.04 Linux distribution for WSL.

- Run Ubuntu using start menu link or execute next command

```powershell

wsl -d Ubuntu-18.04

```

- Run all commands from this isntallation guide in WSL Ubuntu shell.

## Setup additional components in development environment

### DL models as serverless functions

Install [nuclio platform](https://github.com/nuclio/nuclio):

- You have to install `nuctl` command line tool to build and deploy serverless

functions. Download [the latest release](

https://github.com/nuclio/nuclio/blob/development/docs/reference/nuctl/nuctl.md#download).

- The simplest way to explore Nuclio is to run its graphical user interface (GUI)

of the Nuclio dashboard. All you need in order to run the dashboard is Docker. See

[nuclio documentation](https://github.com/nuclio/nuclio#quick-start-steps)

for more details.

- Create `cvat` project inside nuclio dashboard where you will deploy new

serverless functions and deploy a couple of DL models.

```bash

nuctl create project cvat

```

```bash

nuctl deploy --project-name cvat \

--path serverless/openvino/dextr/nuclio \

--volume `pwd`/serverless/openvino/common:/opt/nuclio/common \

--platform local

```

<details>

```bash

20.07.17 12:02:23.247 nuctl (I) Deploying function {"name": ""}

20.07.17 12:02:23.248 nuctl (I) Building {"versionInfo": "Label: 1.4.8, Git commit: 238d4539ac7783896d6c414535d0462b5f4cbcf1, OS: darwin, Arch: amd64, Go version: go1.14.3", "name": ""}

20.07.17 12:02:23.447 nuctl (I) Cleaning up before deployment

20.07.17 12:02:23.535 nuctl (I) Function already exists, deleting

20.07.17 12:02:25.877 nuctl (I) Staging files and preparing base images

20.07.17 12:02:25.891 nuctl (I) Building processor image {"imageName": "cvat/openvino.dextr:latest"}

20.07.17 12:02:25.891 nuctl.platform.docker (I) Pulling image {"imageName": "quay.io/nuclio/handler-builder-python-onbuild:1.4.8-amd64"}

20.07.17 12:02:29.270 nuctl.platform.docker (I) Pulling image {"imageName": "quay.io/nuclio/uhttpc:0.0.1-amd64"}

20.07.17 12:02:33.208 nuctl.platform (I) Building docker image {"image": "cvat/openvino.dextr:latest"}

20.07.17 12:02:34.464 nuctl.platform (I) Pushing docker image into registry {"image": "cvat/openvino.dextr:latest", "registry": ""}

20.07.17 12:02:34.464 nuctl.platform (I) Docker image was successfully built and pushed into docker registry {"image": "cvat/openvino.dextr:latest"}

20.07.17 12:02:34.464 nuctl (I) Build complete {"result": {"Image":"cvat/openvino.dextr:latest","UpdatedFunctionConfig":{"metadata":{"name":"openvino.dextr","namespace":"nuclio","labels":{"nuclio.io/project-name":"cvat"},"annotations":{"framework":"openvino","spec":"","type":"interactor"}},"spec":{"description":"Deep Extreme Cut","handler":"main:handler","runtime":"python:3.6","env":[{"name":"NUCLIO\_PYTHON\_EXE\_PATH","value":"/opt/nuclio/python3"}],"resources":{},"image":"cvat/openvino.dextr:latest","targetCPU":75,"triggers":{"myHttpTrigger":{"class":"","kind":"http","name":"","maxWorkers":2,"workerAvailabilityTimeoutMilliseconds":10000,"attributes":{"maxRequestBodySize":33554432}}},"volumes":[{"volume":{"name":"volume-1","hostPath":{"path":"/Users/nmanovic/Workspace/cvat/serverless/openvino/common"}},"volumeMount":{"name":"volume-1","mountPath":"/opt/nuclio/common"}}],"build":{"image":"cvat/openvino.dextr","baseImage":"openvino/ubuntu18\_runtime:2020.2","directives":{"postCopy":[{"kind":"RUN","value":"curl -O https://download.01.org/openvinotoolkit/models\_contrib/cvat/dextr\_model\_v1.zip"},{"kind":"RUN","value":"unzip dextr\_model\_v1.zip"},{"kind":"RUN","value":"pip3 install Pillow"},{"kind":"USER","value":"openvino"}],"preCopy":[{"kind":"USER","value":"root"},{"kind":"WORKDIR","value":"/opt/nuclio"},{"kind":"RUN","value":"ln -s /usr/bin/pip3 /usr/bin/pip"}]},"codeEntryType":"image"},"platform":{},"readinessTimeoutSeconds":60,"eventTimeout":"30s"}}}}

20.07.17 12:02:35.012 nuctl.platform (I) Waiting for function to be ready {"timeout": 60}

20.07.17 12:02:37.448 nuctl (I) Function deploy complete {"httpPort": 55274}

```

</details>

```bash

nuctl deploy --project-name cvat \

--path serverless/openvino/omz/public/yolo-v3-tf/nuclio \

--volume `pwd`/serverless/openvino/common:/opt/nuclio/common \

--platform local

```

<details>

```bash

20.07.17 12:05:23.377 nuctl (I) Deploying function {"name": ""}

20.07.17 12:05:23.378 nuctl (I) Building {"versionInfo": "Label: 1.4.8, Git commit: 238d4539ac7783896d6c414535d0462b5f4cbcf1, OS: darwin, Arch: amd64, Go version: go1.14.3", "name": ""}

20.07.17 12:05:23.590 nuctl (I) Cleaning up before deployment

20.07.17 12:05:23.694 nuctl (I) Function already exists, deleting

20.07.17 12:05:24.271 nuctl (I) Staging files and preparing base images

20.07.17 12:05:24.274 nuctl (I) Building processor image {"imageName": "cvat/openvino.omz.public.yolo-v3-tf:latest"}

20.07.17 12:05:24.274 nuctl.platform.docker (I) Pulling image {"imageName": "quay.io/nuclio/handler-builder-python-onbuild:1.4.8-amd64"}

20.07.17 12:05:27.432 nuctl.platform.docker (I) Pulling image {"imageName": "quay.io/nuclio/uhttpc:0.0.1-amd64"}

20.07.17 12:05:31.462 nuctl.platform (I) Building docker image {"image": "cvat/openvino.omz.public.yolo-v3-tf:latest"}

20.07.17 12:05:32.798 nuctl.platform (I) Pushing docker image into registry {"image": "cvat/openvino.omz.public.yolo-v3-tf:latest", "registry": ""}

20.07.17 12:05:32.798 nuctl.platform (I) Docker image was successfully built and pushed into docker registry {"image": "cvat/openvino.omz.public.yolo-v3-tf:latest"}

20.07.17 12:05:32.798 nuctl (I) Build complete {"result": {"Image":"cvat/openvino.omz.public.yolo-v3-tf:latest","UpdatedFunctionConfig":{"metadata":{"name":"openvino.omz.public.yolo-v3-tf","namespace":"nuclio","labels":{"nuclio.io/project-name":"cvat"},"annotations":{"framework":"openvino","name":"YOLO v3","spec":"[\n { \"id\": 0, \"name\": \"person\" },\n { \"id\": 1, \"name\": \"bicycle\" },\n { \"id\": 2, \"name\": \"car\" },\n { \"id\": 3, \"name\": \"motorbike\" },\n { \"id\": 4, \"name\": \"aeroplane\" },\n { \"id\": 5, \"name\": \"bus\" },\n { \"id\": 6, \"name\": \"train\" },\n { \"id\": 7, \"name\": \"truck\" },\n { \"id\": 8, \"name\": \"boat\" },\n { \"id\": 9, \"name\": \"traffic light\" },\n { \"id\": 10, \"name\": \"fire hydrant\" },\n { \"id\": 11, \"name\": \"stop sign\" },\n { \"id\": 12, \"name\": \"parking meter\" },\n { \"id\": 13, \"name\": \"bench\" },\n { \"id\": 14, \"name\": \"bird\" },\n { \"id\": 15, \"name\": \"cat\" },\n { \"id\": 16, \"name\": \"dog\" },\n { \"id\": 17, \"name\": \"horse\" },\n { \"id\": 18, \"name\": \"sheep\" },\n { \"id\": 19, \"name\": \"cow\" },\n { \"id\": 20, \"name\": \"elephant\" },\n { \"id\": 21, \"name\": \"bear\" },\n { \"id\": 22, \"name\": \"zebra\" },\n { \"id\": 23, \"name\": \"giraffe\" },\n { \"id\": 24, \"name\": \"backpack\" },\n { \"id\": 25, \"name\": \"umbrella\" },\n { \"id\": 26, \"name\": \"handbag\" },\n { \"id\": 27, \"name\": \"tie\" },\n { \"id\": 28, \"name\": \"suitcase\" },\n { \"id\": 29, \"name\": \"frisbee\" },\n { \"id\": 30, \"name\": \"skis\" },\n { \"id\": 31, \"name\": \"snowboard\" },\n { \"id\": 32, \"name\": \"sports ball\" },\n { \"id\": 33, \"name\": \"kite\" },\n { \"id\": 34, \"name\": \"baseball bat\" },\n { \"id\": 35, \"name\": \"baseball glove\" },\n { \"id\": 36, \"name\": \"skateboard\" },\n { \"id\": 37, \"name\": \"surfboard\" },\n { \"id\": 38, \"name\": \"tennis racket\" },\n { \"id\": 39, \"name\": \"bottle\" },\n { \"id\": 40, \"name\": \"wine glass\" },\n { \"id\": 41, \"name\": \"cup\" },\n { \"id\": 42, \"name\": \"fork\" },\n { \"id\": 43, \"name\": \"knife\" },\n { \"id\": 44, \"name\": \"spoon\" },\n { \"id\": 45, \"name\": \"bowl\" },\n { \"id\": 46, \"name\": \"banana\" },\n { \"id\": 47, \"name\": \"apple\" },\n { \"id\": 48, \"name\": \"sandwich\" },\n { \"id\": 49, \"name\": \"orange\" },\n { \"id\": 50, \"name\": \"broccoli\" },\n { \"id\": 51, \"name\": \"carrot\" },\n { \"id\": 52, \"name\": \"hot dog\" },\n { \"id\": 53, \"name\": \"pizza\" },\n { \"id\": 54, \"name\": \"donut\" },\n { \"id\": 55, \"name\": \"cake\" },\n { \"id\": 56, \"name\": \"chair\" },\n { \"id\": 57, \"name\": \"sofa\" },\n { \"id\": 58, \"name\": \"pottedplant\" },\n { \"id\": 59, \"name\": \"bed\" },\n { \"id\": 60, \"name\": \"diningtable\" },\n { \"id\": 61, \"name\": \"toilet\" },\n { \"id\": 62, \"name\": \"tvmonitor\" },\n { \"id\": 63, \"name\": \"laptop\" },\n { \"id\": 64, \"name\": \"mouse\" },\n { \"id\": 65, \"name\": \"remote\" },\n { \"id\": 66, \"name\": \"keyboard\" },\n { \"id\": 67, \"name\": \"cell phone\" },\n { \"id\": 68, \"name\": \"microwave\" },\n { \"id\": 69, \"name\": \"oven\" },\n { \"id\": 70, \"name\": \"toaster\" },\n { \"id\": 71, \"name\": \"sink\" },\n { \"id\": 72, \"name\": \"refrigerator\" },\n { \"id\": 73, \"name\": \"book\" },\n { \"id\": 74, \"name\": \"clock\" },\n { \"id\": 75, \"name\": \"vase\" },\n { \"id\": 76, \"name\": \"scissors\" },\n { \"id\": 77, \"name\": \"teddy bear\" },\n { \"id\": 78, \"name\": \"hair drier\" },\n { \"id\": 79, \"name\": \"toothbrush\" }\n]\n","type":"detector"}},"spec":{"description":"YOLO v3 via Intel OpenVINO","handler":"main:handler","runtime":"python:3.6","env":[{"name":"NUCLIO\_PYTHON\_EXE\_PATH","value":"/opt/nuclio/common/python3"}],"resources":{},"image":"cvat/openvino.omz.public.yolo-v3-tf:latest","targetCPU":75,"triggers":{"myHttpTrigger":{"class":"","kind":"http","name":"","maxWorkers":2,"workerAvailabilityTimeoutMilliseconds":10000,"attributes":{"maxRequestBodySize":33554432}}},"volumes":[{"volume":{"name":"volume-1","hostPath":{"path":"/Users/nmanovic/Workspace/cvat/serverless/openvino/common"}},"volumeMount":{"name":"volume-1","mountPath":"/opt/nuclio/common"}}],"build":{"image":"cvat/openvino.omz.public.yolo-v3-tf","baseImage":"openvino/ubuntu18\_dev:2020.2","directives":{"postCopy":[{"kind":"USER","value":"openvino"}],"preCopy":[{"kind":"USER","value":"root"},{"kind":"WORKDIR","value":"/opt/nuclio"},{"kind":"RUN","value":"ln -s /usr/bin/pip3 /usr/bin/pip"},{"kind":"RUN","value":"/opt/intel/openvino/deployment\_tools/open\_model\_zoo/tools/downloader/downloader.py --name yolo-v3-tf -o /opt/nuclio/open\_model\_zoo"},{"kind":"RUN","value":"/opt/intel/openvino/deployment\_tools/open\_model\_zoo/tools/downloader/converter.py --name yolo-v3-tf --precisions FP32 -d /opt/nuclio/open\_model\_zoo -o /opt/nuclio/open\_model\_zoo"}]},"codeEntryType":"image"},"platform":{},"readinessTimeoutSeconds":60,"eventTimeout":"30s"}}}}

20.07.17 12:05:33.285 nuctl.platform (I) Waiting for function to be ready {"timeout": 60}

20.07.17 12:05:35.452 nuctl (I) Function deploy complete {"httpPort": 57308}

```

</details>

- Display a list of running serverless functions using `nuctl` command or see them

in nuclio dashboard:

```bash

nuctl get function

```

<details>

```bash

NAMESPACE | NAME | PROJECT | STATE | NODE PORT | REPLICAS

nuclio | openvino.dextr | cvat | ready | 55274 | 1/1

nuclio | openvino.omz.public.yolo-v3-tf | cvat | ready | 57308 | 1/1

```

</details>

- Test your deployed DL model as a serverless function. The command below

should work on Linux and Mac OS.

```bash

image=$(curl https://upload.wikimedia.org/wikipedia/en/7/7d/Lenna\_%28test\_image%29.png --output - | base64 | tr -d '\n')

cat << EOF > /tmp/input.json

{"image": "$image"}

EOF

cat /tmp/input.json | nuctl invoke openvino.omz.public.yolo-v3-tf -c 'application/json'

```

<details>

```bash

20.07.17 12:07:44.519 nuctl.platform.invoker (I) Executing function {"method": "POST", "url": "http://:57308", "headers": {"Content-Type":["application/json"],"X-Nuclio-Log-Level":["info"],"X-Nuclio-Target":["openvino.omz.public.yolo-v3-tf"]}}

20.07.17 12:07:45.275 nuctl.platform.invoker (I) Got response {"status": "200 OK"}

20.07.17 12:07:45.275 nuctl (I) >>> Start of function logs

20.07.17 12:07:45.275 ino.omz.public.yolo-v3-tf (I) Run yolo-v3-tf model {"worker\_id": "0", "time": 1594976864570.9353}

20.07.17 12:07:45.275 nuctl (I) <<< End of function logs

> Response headers:

Date = Fri, 17 Jul 2020 09:07:45 GMT

Content-Type = application/json

Content-Length = 100

Server = nuclio

> Response body:

[

{

"confidence": "0.9992254",

"label": "person",

"points": [

39,

124,

408,

512

],

"type": "rectangle"

}

]

```

### Run Cypress tests

- Install ?ypress as described in the [documentation](https://docs.cypress.io/guides/getting-started/installing-cypress.html).

- Run cypress tests:

```sh

cd <cvat\_local\_repository>/tests

<cypress\_installation\_directory>/node\_modules/.bin/cypress run --headless --browser chrome

```

For more information, see the [documentation](https://docs.cypress.io/).

## JavaScript/Typescript coding style

We use the [Airbnb JavaScript Style Guide](https://github.com/airbnb/javascript) for JavaScript code with a

little exception - we prefer 4 spaces for indentation of nested blocks and statements.

## Branching model

The project uses [a successful Git branching model](https://nvie.com/posts/a-successful-git-branching-model).

Thus it has a couple of branches. Some of them are described below:

- `origin/master` to be the main branch where the source code of

HEAD always reflects a production-ready state

- `origin/develop` to be the main branch where the source code of

HEAD always reflects a state with the latest delivered development

changes for the next release. Some would call this the integration branch.

## Using the issue tracker

The issue tracker is the preferred channel for [bug reports](#bugs),

[features requests](#features) and [submitting pull

requests](#pull-requests), but please respect the following restrictions:

- Please \*\*do not\*\* use the issue tracker for personal support requests (use

[Stack Overflow](http://stackoverflow.com)).

- Please \*\*do not\*\* derail or troll issues. Keep the discussion on topic and

respect the opinions of others.

<a name="bugs"></a>

## Bug reports

A bug is a \_demonstrable problem\_ that is caused by the code in the repository.

Good bug reports are extremely helpful - thank you!

Guidelines for bug reports:

1. \*\*Use the GitHub issue search\*\* &mdash; check if the issue has already been

reported.

1. \*\*Check if the issue has been fixed\*\* &mdash; try to reproduce it using the

latest `develop` branch in the repository.

1. \*\*Isolate the problem\*\* &mdash; ideally create a reduced test case.

A good bug report shouldn't leave others needing to chase you up for more

information. Please try to be as detailed as possible in your report. What is

your environment? What steps will reproduce the issue? What browser(s) and OS

experience the problem? What would you expect to be the outcome? All these

details will help people to fix any potential bugs.

Example:

> Short and descriptive example bug report title

>

> A summary of the issue and the browser/OS environment in which it occurs. If

> suitable, include the steps required to reproduce the bug.

>

> 1. This is the first step

> 1. This is the second step

> 1. Further steps, etc.

>

> Any other information you want to share that is relevant to the issue being

> reported. This might include the lines of code that you have identified as

> causing the bug, and potential solutions (and your opinions on their

> merits).

<a name="features"></a>

## Feature requests

Feature requests are welcome. But take a moment to find out whether your idea

fits with the scope and aims of the project. It's up to \_you\_ to make a strong

case to convince the project's developers of the merits of this feature. Please

provide as much detail and context as possible.

<a name="pull-requests"></a>

## Pull requests

Good pull requests - patches, improvements, new features - are a fantastic

help. They should remain focused in scope and avoid containing unrelated

commits.

\*\*Please ask first\*\* before embarking on any significant pull request (e.g.

implementing features, refactoring code, porting to a different language),

otherwise you risk spending a lot of time working on something that the

project's developers might not want to merge into the project.

Please adhere to the coding conventions used throughout a project (indentation,

accurate comments, etc.) and any other requirements (such as test coverage).

Follow this process if you'd like your work considered for inclusion in the

project:

1. [Fork](https://docs.github.com/en/github/getting-started-with-github/fork-a-repo) the project, clone your fork,

and configure the remotes:

```bash

# Clone your fork of the repo into the current directory

git clone https://github.com/<your-username>/<repo-name>

# Navigate to the newly cloned directory

cd <repo-name>

# Assign the original repo to a remote called "upstream"

git remote add upstream https://github.com/<upstream-owner>/<repo-name>

```

1. If you cloned a while ago, get the latest changes from upstream:

```bash

git checkout <dev-branch>

git pull upstream <dev-branch>

```

1. Create a new topic branch (off the main project development branch) to

contain your feature, change, or fix:

```bash

git checkout -b <topic-branch-name>

```

1. Commit your changes in logical chunks. Please adhere to these [git commit

message guidelines](http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html)

or your code is unlikely be merged into the main project. Use Git's

[interactive rebase](https://docs.github.com/en/github/using-git/about-git-rebase)

feature to tidy up your commits before making them public.

1. Locally merge (or rebase) the upstream development branch into your topic branch:

```bash

git pull [--rebase] upstream <dev-branch>

```

1. Push your topic branch up to your fork:

```bash

git push origin <topic-branch-name>

```

1. [Open a Pull Request](hhttps://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests)

with a clear title and description.

\*\*IMPORTANT\*\*: By submitting a patch, you agree to allow the project owner to

license your work under the same license as that used by the project.