# Contributing Guidelines

## How to become a contributor and submit your own code

### Contributor License Agreements

We'd love to accept your patches! Before we can take them, we have to jump a couple of legal hurdles.

Please fill out either the individual or corporate Contributor License Agreement (CLA).

\* If you are an individual writing original source code and you're sure you own the intellectual property, then you'll need to sign an [individual CLA](https://identity.linuxfoundation.org/node/285/node/285/individual-signup).

\* If you work for a company that wants to allow you to contribute your work, then you'll need to sign a [corporate CLA](https://identity.linuxfoundation.org/node/285/organization-signup).

Follow either of the two links above to access the appropriate CLA and instructions for how to sign and return it. Once we receive it, we'll be able to accept your pull requests.

## Composition of This Repository and Where/How to Contribute

The Kubernetes Python client consists of this main repository (repo) and the base repo, the [python-base](https://github.com/kubernetes-client/python-base) repository. The main repository contains mostly files that are generated by the OpenAPI generator from [this OpenAPI spec](scripts/swagger.json). The base repo is the utility part of the python client and allows developers to create their own kubernetes clients. The base repo is a submodule of the main repo.

### Where to Submit Your Patch

The following folders are automatically generated. You will need to submit a patch to the upstream Kubernetes repo [kubernetes](https://github.com/kubernetes/kubernetes) or the OpenAPI generator repo [openapi-generator](https://github.com/OpenAPITools/openapi-generator).

- [kubernetes/client](kubernetes/client)

- [kubernetes/test](kubernetes/test)

- [kubernetes/docs](kubernetes/docs).

In this main repo, the following folders contain developer written codes and the patches should be submitted as pull requests here:

- [kubernetes/e2e\_test](kubernetes/e2e\_test)

- [kubernetes/utils](kubernetes/utils)

- [examples](examples)

- [scripts](scripts).

The following folders and symbolic links in this main repo are in fact from the base repo. The base repo contains developer written codes only. The patches should be sent to the base repo instead:

- kubernetes/base

- kubernetes/config

- kubernetes/stream

- kubernetes/watch.

### Contributing A Patch

1. Submit an issue describing your proposed change to the repo in question.

2. The [repo owners](OWNERS) will respond to your issue promptly.

3. If your proposed change is accepted, and you haven't already done so, sign a Contributor License Agreement (see details above).

4. Fork the desired repo, develop and test your code changes. Add a test if possible.

5. Submit a pull request.

### Adding Dependencies

If your patch depends on new packages, add those packages to [requirements.txt](requirements.txt) and/or [setup.py](setup.py). If these package are for testing only, add those to [test-requirements.txt](test-requirements.txt).

### Commits

Generally we would like to see one commit per pull request. However, if the pull request is reasonably large, the PR can be divided into several commits that make logical sense. The commit message should be clear and indicative of the aim of the fix. Sometimes multiple commits in a single pull request is acceptable if it meets the Kubernetes [pull request guidelines](https://github.com/kubernetes/community/blob/master/contributors/guide/pull-requests.md#6-squashing-and-commit-titles).

If you have several commits in a pull request and have been asked to squash your commits, please use ```git reset --soft HEAD~N\_COMMITS``` and commit again to make your PR a single commit.

### Windows Developers

The symbolic links contained in this repo do not work for Windows operating systems. If you are a Windows developer, please run the [fix](scripts/windows-setup-fix.bat) inside the scripts folder or manually copy the content of the [kubernetes/base](https://github.com/kubernetes-client/python-base) folder into the [kubernetes](kubernetes) folder.

### Writing Tests

In addition to running the fix yourself and telling us that your fix works, you can demonstrate that your fix really works by using unit tests and end to end tests. Tests are mainly located in three places. You should put your tests into the places that they fit in.

1. [Generated tests](kubernetes/test) by OpenAPI generator: these tests should pass and do not require modification.

2. [End to end tests](kubernetes/e2e\_test): these are tests that can only be verified with a live kubernetes server.

3. Base repo tests in the [base](https://github.com/kubernetes-client/python-base) repo, in which the test files are named ```test\_\*.py```: These tests use the package ```Mock``` and confirms the functionality of the base repo files.

### Coding Style

We use an automatic coding style checker by using the ```diff``` of the autopep8 output and your code file. To make sure that your code passes the coding style checker, run ```autopep8 --in-place --aggressive --aggressive your\_code.py``` before committing and submitting.

## Running Tests Locally

If you write a new end to end (e2e) test, or change behaviors that affect e2e tests, you should set up a local cluster and test them on your machine. The following steps will help you run the unit tests.

1. Acquire a local cluster. [Minikube](https://github.com/kubernetes/minikube) is a good choice for Windows and Linux developers. Alternatively if you are on Linux, you can clone the [kubernetes](https://github.com/kubernetes/kubernetes) repo and run [install-etcd.sh](https://github.com/kubernetes/kubernetes/blob/master/hack/install-etcd.sh) and then [local-up-cluster.sh](https://github.com/kubernetes/kubernetes/blob/master/hack/local-up-cluster.sh) to get a local cluster up and running.

2. Run the unit tests. In the root directory of the main repo, run ```python -m unittest discover```.

3. Check the test results and make corresponding fixes.

## Update the Base Submodule in the Main Repo After Your python-base PR Is Merged

Your contribution to the base repo will not be automatically reflected in the main repo after your PR is merged. Instead, please update the ```base``` submodule in your fork of the main repo as follows:

```bash

$ git submodule update --remote

```

You may now add a release note to [CHANGELOG.md](CHANGELOG.md) if needed and then commit and push to your fork. You can now send a PR to this main repo to complete your contribution.