# Contributing code

Moto has a [Code of Conduct](https://github.com/spulec/moto/blob/master/CODE\_OF\_CONDUCT.md), you can expect to be treated with respect at all times when interacting with this project.

## Running the tests locally

Moto has a [Makefile](./Makefile) which has some helpful commands for getting set up.

You should be able to run `make init` to install the dependencies and then `make test` to run the tests.

\*NB. On first run, some tests might take a while to execute, especially the Lambda ones, because they may need to download a Docker image before they can execute.\*

## Linting

Run `make lint` or `black --check moto tests` to verify whether your code confirms to the guidelines.

## Getting to grips with the codebase

Moto maintains a list of [good first issues](https://github.com/spulec/moto/contribute) which you may want to look at before

implementing a whole new endpoint.

## Missing features

Moto is easier to contribute to than you probably think. There's [a list of which endpoints have been implemented](https://github.com/spulec/moto/blob/master/IMPLEMENTATION\_COVERAGE.md) and we invite you to add new endpoints to existing services or to add new services.

How to teach Moto to support a new AWS endpoint:

\* Search for an existing [issue](https://github.com/spulec/moto/issues) that matches what you want to achieve.

\* If one doesn't already exist, create a new issue describing what's missing. This is where we'll all talk about the new addition and help you get it done.

\* Create a [pull request](https://help.github.com/articles/using-pull-requests/) and mention the issue # in the PR description.

\* Try to add a failing test case. For example, if you're trying to implement `boto3.client('acm').import\_certificate()` you'll want to add a new method called `def test\_import\_certificate` to `tests/test\_acm/test\_acm.py`.

\* Implementing the feature itself can be done by creating a method called `import\_certificate` in `moto/acm/responses.py`. It's considered good practice to deal with input/output formatting and validation in `responses.py`, and create a method `import\_certificate` in `moto/acm/models.py` that handles the actual import logic.

\* If you can also implement the code that gets that test passing then great! If not, just ask the community for a hand and somebody will assist you.

## Before pushing changes to GitHub

1. Run `black moto/ tests/` over your code to ensure that it is properly formatted

1. Run `make test` to ensure your tests are passing

## Python versions

moto currently supports both Python 2 and 3, so make sure your tests pass against both major versions of Python.

## Missing services

Implementing a new service from scratch is more work, but still quite straightforward. All the code that intercepts network requests to `\*.amazonaws.com` is already handled for you in `moto/core` - all that's necessary for new services to be recognized is to create a new decorator and determine which URLs should be intercepted.

See this PR for an example of what's involved in creating a new service: https://github.com/spulec/moto/pull/2409/files

Note the `urls.py` that redirects all incoming URL requests to a generic `dispatch` method, which in turn will call the appropriate method in `responses.py`.

If you want more control over incoming requests or their bodies, it is possible to redirect specific requests to a custom method. See this PR for an example: https://github.com/spulec/moto/pull/2957/files

## Maintainers

### Releasing a new version of Moto

You'll need a PyPi account and a DockerHub account to release Moto. After we release a new PyPi package we build and push the [motoserver/moto](https://hub.docker.com/r/motoserver/moto/) Docker image.

\* First, `scripts/bump\_version` modifies the version and opens a PR

\* Then, merge the new pull request

\* Finally, generate and ship the new artifacts with `make publish`