# Contributing to vLLM

Thank you for your interest in contributing to vLLM! Our community is open to everyone and welcomes all kinds of contributions, no matter how small or large. There are several ways you can contribute to the project:

- Identify and report any issues or bugs.
- Request or add support for a new model.
- Suggest or implement new features.
- Improve documentation or contribute a how-to guide.

We also believe in the power of community support; thus, answering queries, offering PR reviews, and assisting others are also highly regarded and beneficial contributions.

Finally, one of the most impactful ways to support us is by raising awareness about vLLM. Talk about it in your blog posts and highlight how it's driving your incredible projects. Express your support on social media if you're using vLLM, or simply offer your appreciation by starring our repository!

## License

See <gh-file:LICENSE>.

## Developing

Depending on the kind of development you'd like to do (e.g. Python, CUDA), you can choose to build vLLM with or without compilation.

Check out the [building from source](#build-from-source) documentation for details.

## Testing

```bash
pip install -r requirements-dev.txt

# Linting, formatting and static type checking
pre-commit install --hook-type pre-commit --hook-type commit-msg

# You can manually run pre-commit with
pre-commit run --all-files

# Unit tests
pytest tests/
```

:::{note}
Currently, the repository is not fully checked by `mypy`.
:::

## Issues

If you encounter a bug or have a feature request, please [search existing issues](https://github.com/vllm-project/vllm/issues?q=is%3Aissue) first to see if it has already been reported. If not, please [file a new issue](https://github.com/vllm-project/vllm/issues/new/choose), providing as much relevant information as possible.

:::{important}

If you discover a security vulnerability, please follow the instructions [here](gh-file:SECURITY.md#reporting-a-vulnerability).

:::

## Pull Requests & Code Reviews

Thank you for your contribution to vLLM! Before submitting the pull request, please ensure the PR meets the following criteria. This helps vLLM maintain the code quality and improve the efficiency of the review process.

### DCO and Signed-off-by

When contributing changes to this project, you must agree to the <gh-file:DCO>. Commits must include a `Signed-off-by:` header which certifies agreement with the terms of the DCO.

Using `-s` with `git commit` will automatically add this header.

### PR Title and Classification

Only specific types of PRs will be reviewed. The PR title is prefixed appropriately to indicate the type of change. Please use one of the following:

- `[Bugfix]` for bug fixes.
- `[CI/Build]` for build or continuous integration improvements.

- `[Doc]` for documentation fixes and improvements.
- `[Model]` for adding a new model or improving an existing model. Model name
  should appear in the title.
- `[Frontend]` For changes on the vLLM frontend (e.g., OpenAI API server,
  `LLM` class, etc.)
- `[Kernel]` for changes affecting CUDA kernels or other compute kernels.
- `[Core]` for changes in the core vLLM logic (e.g., `LLMEngine`,
  `AsyncLLMEngine`, `Scheduler`, etc.)
- `[Hardware][Vendor]` for hardware-specific changes. Vendor name should
  appear in the prefix (e.g., `[Hardware][AMD]`).
- `[Misc]` for PRs that do not fit the above categories. Please use this
  sparingly.

:::{note}
If the PR spans more than one category, please include all relevant prefixes.
:::

### Code Quality

The PR needs to meet the following code quality standards:

- We adhere to [Google Python style guide](https://google.github.io/styleguide/pyguide.html) and [Google C++ style guide](https://google.github.io/styleguide/cppguide.html).
- Pass all linter checks. Please use `pre-commit` to format your code. See
  <https://pre-commit.com/#usage> if `pre-commit` is new to you.
- The code needs to be well-documented to ensure future contributors can easily
  understand the code.

- Include sufficient tests to ensure the project stays correct and robust. This
  includes both unit tests and integration tests.
- Please add documentation to `docs/source/` if the PR modifies the
  user-facing behaviors of vLLM. It helps vLLM users understand and utilize the
  new features or changes.

### Adding or Changing Kernels

Each custom kernel needs a schema and one or more implementations to be registered with PyTorch.

- Make sure custom ops are registered following PyTorch guidelines:
  [Custom C++ and CUDA Operators](https://pytorch.org/tutorials/advanced/cpp_custom_ops.html#cpp-custom-ops-tutorial)
  and [The Custom Operators Manual](https://docs.google.com/document/d/1_W62p8WJOQQUzPsJYa7s701JXt0qf2OfLub2sbkHOaU).
- Custom operations that return `Tensors` require meta-functions.
  Meta-functions should be implemented and registered in Python so that dynamic
  dims can be handled automatically. See above documents for a description of
  meta-functions.
- Use [torch.library.opcheck()](https://pytorch.org/docs/stable/library.html#torch.library.opcheck)
  to test the function registration and meta-function for any registered ops.
  See `tests/kernels` for examples.
- When changing the C++ signature of an existing op, the schema must be updated
  to reflect the changes.
- If a new custom type is needed, see the following document:

[Custom Class Support in PT2](https://docs.google.com/document/d/18fBMPuOJ0fY5ZQ6YyrHUppw9FA332CpNtgB6SOIgyuA).

### Notes for Large Changes

Please keep the changes as concise as possible. For major architectural changes
(>500 LOC excluding kernel/data/config/test), we would expect a GitHub issue
(RFC) discussing the technical design and justification. Otherwise, we will tag
it with `rfc-required` and might not go through the PR.

### What to Expect for the Reviews

The goal of the vLLM team is to be a *transparent reviewing machine*. We would
like to make the review process transparent and efficient and make sure no
contributor feels confused or frustrated. However, the vLLM team is small, so we
need to prioritize some PRs over others. Here is what you can expect from the
review process:

- After the PR is submitted, the PR will be assigned to a reviewer. Every
  reviewer will pick up the PRs based on their expertise and availability.
- After the PR is assigned, the reviewer will provide status updates every 2-3
  days. If the PR is not reviewed within 7 days, please feel free to ping the
  reviewer or the vLLM team.
- After the review, the reviewer will put an `action-required` label on the PR
  if there are changes required. The contributor should address the comments and
  ping the reviewer to re-review the PR.

- Please respond to all comments within a reasonable time frame. If a comment
  isn't clear or you disagree with a suggestion, feel free to ask for
  clarification or discuss the suggestion.

## Thank You

Finally, thank you for taking the time to read these guidelines and for your interest in contributing to vLLM.

All of your contributions help make vLLM a great tool and community for everyone!