# Contributing guidelines

## Pull Request Checklist

Before sending your pull requests, make sure you followed this list.

- Read [contributing guidelines](CONTRIBUTING.md).

- Read [Code of Conduct](CODE\_OF\_CONDUCT.md).

- Ensure you have signed the [Contributor License Agreement (CLA)](https://cla.developers.google.com/).

- Check if my changes are consistent with the [guidelines](https://github.com/tensorflow/tensorflow/blob/master/CONTRIBUTING.md#general-guidelines-and-philosophy-for-contribution).

- Changes are consistent with the [Coding Style](https://github.com/tensorflow/tensorflow/blob/master/CONTRIBUTING.md#c-coding-style).

- Run [Unit Tests](https://github.com/tensorflow/tensorflow/blob/master/CONTRIBUTING.md#running-unit-tests).

## 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://code.google.com/legal/individual-cla-v1.0.html).

\* 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://code.google.com/legal/corporate-cla-v1.0.html).

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.

\*\*\*NOTE\*\*\*: Only original source code from you and other people that have signed the CLA can be accepted into the main repository.

### Contributing code

If you have improvements to TensorFlow, send us your pull requests! For those

just getting started, Github has a

[how to](https://help.github.com/articles/using-pull-requests/).

TensorFlow team members will be assigned to review your pull requests. Once the

pull requests are approved and pass continuous integration checks, a TensorFlow

team member will apply `ready to pull` label to your change. This means we are

working on getting your pull request submitted to our internal repository. After

the change has been submitted internally, your pull request will be merged

automatically on GitHub.

If you want to contribute, start working through the TensorFlow codebase,

navigate to the

[Github "issues" tab](https://github.com/tensorflow/tensorflow/issues) and start

looking through interesting issues. If you are not sure of where to start, then

start by trying one of the smaller/easier issues here i.e.

[issues with the "good first issue" label](https://github.com/tensorflow/tensorflow/labels/good%20first%20issue)

and then take a look at the

[issues with the "contributions welcome" label](https://github.com/tensorflow/tensorflow/labels/stat%3Acontributions%20welcome).

These are issues that we believe are particularly well suited for outside

contributions, often because we probably won't get to them right now. If you

decide to start on an issue, leave a comment so that other people know that

you're working on it. If you want to help out, but not alone, use the issue

comment thread to coordinate.

### Contribution guidelines and standards

Before sending your pull request for

[review](https://github.com/tensorflow/tensorflow/pulls),

make sure your changes are consistent with the guidelines and follow the

TensorFlow coding style.

#### General guidelines and philosophy for contribution

\* Include unit tests when you contribute new features, as they help to a)

prove that your code works correctly, and b) guard against future breaking

changes to lower the maintenance cost.

\* Bug fixes also generally require unit tests, because the presence of bugs

usually indicates insufficient test coverage.

\* Keep API compatibility in mind when you change code in core TensorFlow,

e.g., code in

[tensorflow/core](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/core)

and

[tensorflow/python](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/python).

TensorFlow has passed version 1.0 and hence cannot make

non-backward-compatible API changes without a major release. Reviewers of

your pull request will comment on any API compatibility issues.

\* When you contribute a new feature to TensorFlow, the maintenance burden is

(by default) transferred to the TensorFlow team. This means that the benefit

of the contribution must be compared against the cost of maintaining the

feature.

\* Full new features (e.g., a new op implementing a cutting-edge algorithm)

typically will live in

[tensorflow/addons](https://github.com/tensorflow/addons) to get some

airtime before a decision is made regarding whether they are to be migrated

to the core.

\* As every PR requires several CPU/GPU hours of CI testing, we discourage

submitting PRs to fix one typo, one warning,etc. We recommend fixing the

same issue at the file level at least (e.g.: fix all typos in a file, fix

all compiler warning in a file, etc.)

\* Tests should follow the

[testing best practices](https://www.tensorflow.org/community/contribute/tests)

guide.

#### License

Include a license at the top of new files.

\* [C/C++ license example](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/framework/op.cc#L1)

\* [Python license example](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/ops/nn.py#L1)

\* [Java license example](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/java/src/main/java/org/tensorflow/Graph.java#L1)

\* [Go license example](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/go/operation.go#L1)

\* [Bash license example](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/tools/ci\_build/ci\_sanity.sh#L2)

\* [HTML license example](https://github.com/tensorflow/tensorboard/blob/master/tensorboard/components/tf\_backend/tf-backend.html#L2)

\* [JavaScript/TypeScript license example](https://github.com/tensorflow/tensorboard/blob/master/tensorboard/components/tf\_backend/backend.ts#L1)

Bazel BUILD files also need to include a license section, e.g.,

[BUILD example](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/BUILD#L61).

#### C++ coding style

Changes to TensorFlow C++ code should conform to

[Google C++ Style Guide](https://google.github.io/styleguide/cppguide.html).

Use `clang-tidy` to check your C/C++ changes. To install `clang-tidy` on ubuntu:16.04, do:

```bash

apt-get install -y clang-tidy

```

You can check a C/C++ file by doing:

```bash

clang-format <my\_cc\_file> --style=google > /tmp/my\_cc\_file.cc

diff <my\_cc\_file> /tmp/my\_cc\_file.cc

```

#### Python coding style

Changes to TensorFlow Python code should conform to

[Google Python Style Guide](https://github.com/google/styleguide/blob/gh-pages/pyguide.md)

Use `pylint` to check your Python changes. To install `pylint` and check a file

with `pylint` against TensorFlow's custom style definition:

```bash

pip install pylint

pylint --rcfile=tensorflow/tools/ci\_build/pylintrc myfile.py

```

Note `pylint --rcfile=tensorflow/tools/ci\_build/pylintrc` should run from the

top level tensorflow directory.

#### Coding style for other languages

\* [Google Java Style Guide](https://google.github.io/styleguide/javaguide.html)

\* [Google JavaScript Style Guide](https://google.github.io/styleguide/jsguide.html)

\* [Google Shell Style Guide](https://google.github.io/styleguide/shell.xml)

\* [Google Objective-C Style Guide](https://google.github.io/styleguide/objcguide.html)

#### Running sanity check

If you have Docker installed on your system, you can perform a sanity check on

your changes by running the command:

```bash

tensorflow/tools/ci\_build/ci\_build.sh CPU tensorflow/tools/ci\_build/ci\_sanity.sh

```

This will catch most license, Python coding style and BUILD file issues that

may exist in your changes.

#### Running unit tests

There are two ways to run TensorFlow unit tests.

1. Using tools and libraries installed directly on your system.

Refer to the

[CPU-only developer Dockerfile](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/tools/dockerfiles/dockerfiles/devel-cpu.Dockerfile)

and

[GPU developer Dockerfile](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/tools/dockerfiles/dockerfiles/devel-gpu.Dockerfile)

for the required packages. Alternatively, use the said

[Docker images](https://hub.docker.com/r/tensorflow/tensorflow/tags/), e.g.,

`tensorflow/tensorflow:devel` and `tensorflow/tensorflow:devel-gpu` for

development to avoid installing the packages directly on your system (in

which case remember to change directory from `/root` to `/tensorflow` once

you get into the running container so `bazel` can find the `tensorflow`

workspace).

Once you have the packages installed, you can run a specific unit test in

bazel by doing as follows:

If the tests are to be run on GPU, add CUDA paths to LD\_LIBRARY\_PATH and add

the `cuda` option flag

```bash

export LD\_LIBRARY\_PATH="${LD\_LIBRARY\_PATH}:/usr/local/cuda/lib64:/usr/local/cuda/extras/CUPTI/lib64:$LD\_LIBRARY\_PATH"

export flags="--config=opt --config=cuda -k"

```

For example, to run all tests under tensorflow/python, do:

```bash

bazel test ${flags} //tensorflow/python/...

```

2. Using [Docker](https://www.docker.com) and TensorFlow's CI scripts.

```bash

# Install Docker first, then this will build and run cpu tests

tensorflow/tools/ci\_build/ci\_build.sh CPU bazel test //tensorflow/...

```

See

[TensorFlow Builds](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/tools/ci\_build)

for details.