# Contributing

When contributing to this repository, please first discuss the change you wish to make via [GitHub Discussions](https://github.com/vercel/vercel/discussions/new) with the owners of this repository before submitting a Pull Request.

Please read our [code of conduct](CODE\_OF\_CONDUCT.md) and follow it in all your interactions with the project.

## Local development

This project is configured in a monorepo pattern where one repo contains multiple npm packages. Dependencies are installed and managed with `yarn`, not `npm` CLI.

To get started, execute the following:

```

git clone https://github.com/vercel/vercel

yarn install

yarn bootstrap

yarn build

yarn lint

yarn test

```

Make sure all the tests pass before making changes.

## Verifying your change

Once you are done with your changes (we even suggest doing it along the way ), make sure all the test still run by running

```

yarn build && yarn test

```

from the root of the project.

If any test fails, make sure to fix it along with your changes. See [Interpreting test errors](#Interpreting-test-errors) for more information about how the tests are executed, especially the integration tests.

## Pull Request Process

Once you are confident that your changes work properly, open a pull request on the main repository.

The pull request will be reviewed by the maintainers and the tests will be checked by our continuous integration platform.

## Interpreting test errors

There are 2 kinds of tests in this repository – Unit tests and Integration tests.

Unit tests are run locally with `jest` and execute quickly because they are testing the smallest units of code.

### Integration tests

Integration tests create deployments to your Vercel account using the `test` project name. After each test is deployed, the `probes` key is used to check if the response is the expected value. If the value doesn't match, you'll see a message explaining the difference. If the deployment failed to build, you'll see a more generic message like the following:

```

[Error: Fetched page https://test-8ashcdlew.now.sh/root.js does not contain hello Root!. Instead it contains An error occurred with this application.

NO\_STATUS\_CODE\_FRO Response headers:

cache-control=s-maxage=0

connection=close

content-type=text/plain; charset=utf-8

date=Wed, 19 Jun 2019 18:01:37 GMT

server=now

strict-transport-security=max-age=63072000

transfer-encoding=chunked

x-now-id=iad1:hgtzj-1560967297876-44ae12559f95

x-now-trace=iad1]

```

In such cases you can visit the URL of the failed deployment and append `/\_logs` so see the build error. In the case above, that would be https://test-8ashcdlew.now.sh/\_logs

The logs of this deployment will contain the actual error which may help you to understand what went wrong.

### @zeit/node-file-trace

Some of the Builders use `@zeit/node-file-trace` to tree-shake files before deployment. If you suspect an error with this tree-shaking mechanism, you can create the following script in your project:

```js

const trace = require('@zeit/node-file-trace');

trace(['path/to/entrypoint.js'], {

ts: true,

mixedModules: true,

})

.then(o => console.log(o.fileList))

.then(e => console.error(e));

```

When you run this script, you'll see all imported files. If anything file is missing, the bug is in [@zeit/node-file-trace](https://github.com/vercel/node-file-trace) and not the Builder.

## Deploy a Builder with existing project

Sometimes you want to test changes to a Builder against an existing project, maybe with `vercel dev` or an actual deployment. You can avoid publishing every Builder change to npm by uploading the Builder as a tarball.

1. Change directory to the desired Builder `cd ./packages/now-node`

2. Run `yarn build` to compile typescript and other build steps

3. Run `npm pack` to create a tarball file

4. Run `vercel \*.tgz` to upload the tarball file and get a URL

5. Edit any existing `vercel.json` project and replace `use` with the URL

6. Run `vercel` or `vercel dev` to deploy with the experimental Builder

## Add a New Framework

You can add support for a new Framework by creating a Pull Request for this repository and following the steps below:

1. Add the Framework to the `@vercel/frameworks` package: The file is located in `./packages/frameworks/frameworks.json`. You can copy the structure of an existing one and adjust the required fields. Note that the `settings` property either contains a `value` or a `placeholder`. The `value` property is used when something is not configurable, the `placeholder` is used when something is configurable and can be changed with configuration. An example would be the Output Directory for Hugo, it's `public` by default but can be changed through its config file, so we use `placeholder` with an explanation of what can be used.

2. Add an example to the `./examples` directory: The name of the directory should equal the slug of the framework used in `@vercel/frameworks`. The `.github/EXAMPLE\_README\_TEMPLATE.md` file can be used to create a `README.md` file for the example.

3. Update the `@vercel/static-build` package: The file `./packages/now-static-build/src/frameworks.ts` has to be extended. You can add default routes that will always be applied to projects that use this Framework or specify some paths that will be cached to speed up the build process.

4. After your Pull Request has been merged and released, other users can select the example on the Vercel dashboard and deploy it.