# Contributing to Citus

We're happy you want to contribute! You can help us in different ways:

\* Open an [issue](https://github.com/citusdata/citus/issues) with

suggestions for improvements

\* Fork this repository and submit a pull request

Before accepting any code contributions we ask that Citus contributors

sign a Contributor License Agreement (CLA). For an explanation of

why we ask this as well as instructions for how to proceed, see the

[Citus CLA](https://cla.citusdata.com).

### Getting and building

#### Mac

1. Install Xcode

2. Install packages with Homebrew

```bash

brew update

brew install git postgresql

```

3. Get, build, and test the code

```bash

git clone https://github.com/citusdata/citus.git

cd citus

./configure

make

make install

cd src/test/regress

make check

```

#### Debian-based Linux (Ubuntu, Debian)

1. Install build dependencies

```bash

echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb\_release -cs)-pgdg main" | \

sudo tee /etc/apt/sources.list.d/pgdg.list

wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | \

sudo apt-key add -

sudo apt-get update

sudo apt-get install -y postgresql-server-dev-11 postgresql-11 \

libreadline-dev libselinux1-dev libxslt-dev \

libpam0g-dev git flex make libssl-dev \

libicu-dev \

libkrb5-dev libcurl4-gnutls-dev autoconf

```

2. Get, build, and test the code

```bash

git clone https://github.com/citusdata/citus.git

cd citus

./configure

make

sudo make install

cd src/test/regress

make check

```

#### Red Hat-based Linux (RHEL, CentOS, Fedora)

1. Find the RPM URL for your repo at [yum.postgresql.org](http://yum.postgresql.org/repopackages.php)

2. Register its contents with Yum:

```bash

sudo yum install -y <url>

```

3. Register EPEL and SCL repositories for your distro.

On CentOS:

```bash

yum install -y centos-release-scl-rh epel-release

```

On RHEL, see [this RedHat blog post](https://developers.redhat.com/blog/2018/07/07/yum-install-gcc7-clang/) to install set-up SCL first. Then run:

```bash

yum install -y epel-release

```

4. Install build dependencies

```bash

sudo yum update -y

sudo yum groupinstall -y 'Development Tools'

sudo yum install -y postgresql11-devel postgresql11-server \

libxml2-devel libxslt-devel openssl-devel \

pam-devel readline-devel git libcurl-devel \

llvm5.0 llvm-toolset-7-clang

git clone https://github.com/citusdata/citus.git

cd citus

PG\_CONFIG=/usr/pgsql-11/bin/pg\_config ./configure

make

sudo make install

cd src/test/regress

make check

```

### Following our coding conventions

CircleCI will automatically reject any PRs which do not follow our coding

conventions. The easiest way to ensure your PR adheres to those conventions is

to use the [citus\_indent](https://github.com/citusdata/tools/tree/develop/uncrustify)

tool. This tool uses `uncrustify` under the hood.

```bash

# Uncrustify changes the way it formats code every release a bit. To make sure

# everyone formats consistently we use version 0.68.1:

curl -L https://github.com/uncrustify/uncrustify/archive/uncrustify-0.68.1.tar.gz | tar xz

cd uncrustify-uncrustify-0.68.1/

mkdir build

cd build

cmake ..

make -j5

sudo make install

cd ../..

git clone https://github.com/citusdata/tools.git

cd tools

make uncrustify/.install

```

Once you've done that, you can run the `make reindent` command from the top

directory to recursively check and correct the style of any source files in the

current directory. Under the hood, `make reindent` will run `citus\_indent` and

some other style corrections for you.

You can also run the following in the directory of this repository to

automatically format all the files that you have changed before committing:

```bash

cat > .git/hooks/pre-commit << \_\_EOF\_\_

#!/bin/bash

citus\_indent --check --diff || { citus\_indent --diff; exit 1; }

\_\_EOF\_\_

chmod +x .git/hooks/pre-commit

```

### Making SQL changes

Sometimes you need to make change to the SQL that the citus extension runs upon

creations. The way this is done is by changing the last file in

`src/backend/distributed/sql`, or creating it if the last file is from a

published release. If you needed to create a new file, also change the

`default\_version` field in `src/backend/distributed/citus.control` to match your

new version. All the files in this directory are run in order based on

their name. See [this page in the Postgres

docs](https://www.postgresql.org/docs/current/extend-extensions.html) for more

information on how Postgres runs these files.

#### Changing or creating functions

If you need to change any functions defined by Citus. You should check inside

`src/backend/distributed/sql/udfs` to see if there is already a directory for

this function, if not create one. Then change or create the file called

`latest.sql` in that directory to match how it should create the function. This

should be including any DROP (IF EXISTS), COMMENT and REVOKE statements for this

function.

Then copy the `latest.sql` file to `{version}.sql`, where `{version}` is the

version for which this sql change is, e.g. `{9.0-1.sql}`. Now that you've

created this stable snapshot of the function definition for your version you

should use it in your actual sql file, e.g.

`src/backend/distributed/sql/citus--8.3-1--9.0-1.sql`. You do this by using C

style `#include` statements like this:

```

#include "udfs/myudf/9.0-1.sql"

```

#### Other SQL

Any other SQL you can put directly in the main sql file, e.g.

`src/backend/distributed/sql/citus--8.3-1--9.0-1.sql`.