# Local Development Setup

## Holy Smokes Welcome to CallRail!

Just a quick 59 step process (not really) to get your development environment set up so you can get right to it. If anything doesn't go quite right or you have questions about any of the steps post a question in the `#development` slack channel. Let's do this!

## Getting Started

Before going through the steps, here are some basic assumptions:

### 1. You're on a Mac (Intel core)

This guide was first written for High Sierra (10.13) but has been continuously updated since and should work with the latest Mac OS. It isn't updated for M1 chips; only Intel. You can check if you have an intel or m1 chip by clicking the Apple icon in top left and looking at `Processor` under `About This Mac`

### 2. You're using ZSH as your shell

If you're using a different shell, you probably know enough about how it works to know where your configuration files are.dc-serb

When you see `~/.zshrc` mentioned, use the file that works for you. Since `ZSH` is the default shell for Catalina and beyond, `~/.bash_profile` is what you would use if you're on a version below that.

## How This Guide Works

### Don't type the $ in the commands

Most of these steps involve running a command from the terminal. When writing the command that needs to be run I prefix them with a `$`. Don't actually type that $. It just indicates the beginning of the prompt. GitHub does it this way too.

### Replace <your-data> (and the brackets) with your data

In examples where you need to change some of the text to suit your own computer, you'll see that text in angle brackets `<>`. Don't type the angle brackets, just replace the whole thing with your text.

## Installing Docker and CallRail's Dependencies

### 1. Install the XCode command line tools

This will install git and some useful compilers that we'll need later.

```
1  xcode-select --install
```

Then follow the prompt that pops up.

### 2. Configure git with your name and email

I use the email tied to my github account, which is maybe not your http://callrail.com email address. You can manage your github email addresses at https://github.com/settings/emails if you want to keep things separate.

```
$ git config --global user.name "<Your full name>"
$ git config --global user.email "<Your email address>"
```

### 3. Install HomeBrew

Homebrew is the easiest way to install and manage our development toolset.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

### 4. Set up your SSH key

GitHub's guide is nice and thorough so just follow it:

a. https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/

- Skip the ***Generating a new SSH key for a hardware security key*** section at the bottom

b. https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/

c. Enable SSO by clicking the “Configure SSO” button on your new key and authorizing CallRail

Afterwards, your `~/.ssh/config` file should look like this

```
Host *
 AddKeysToAgent yes
 IgnoreUnknown UseKeychain
 UseKeychain yes
 IdentityFile ~/.ssh/id_ed25519
```

### 5. Setup AWS

Your laptop needs local AWS credentials in order to programmatically access configuration stored on AWS. We use OKTA to sign in to AWS via SSO, and `aws-vault` to manage and generate our credentials locally.

1. Follow the **Gaining Access** portion of the aws setup guide
2. Follow the **Setup** portion of the aws setup guide
3. Verify everything is working

   ```
   ave cr-stage-dev aws sts get-caller-identity
   ```

   This should return something like the following:

   ```
   {
       "UserId": "your-id",
       "Account": "callrail-account",
       "Arn": "arn:aws:sts::<account-number>:assumed-role/AWSReservedSSO_Developer_.../<your-email>"
   }
   ```

### 6. Clone the repos

Now we're getting somewhere! Create the source code repositories on your local machine. I like to keep source code in `~/src`, and the rest of this guide will assume you do the same. Only you know what path you will want to type every day, so choose wisely.

> Putting everything in ~/src/ was the best decision I made since literally all the docs and aliases people share assume that's where the projects are.
>
> I pity anyone who didn't put their projects in src.

> - Alex Herbig

**Note:** If you haven't yet cloned any repo from the `callrail` github org, the first time you do it will ask you to accept RSA key fingerprint. Just type `yes` when prompted and hit `enter`

```
$ mkdir ~/src
$ cd ~/src
$ git clone git@github.com:callrail/setup.git
$ git clone git@github.com:callrail/callrail.git
$ git clone git@github.com:callrail/swappy.git
$ git clone git@github.com:callrail/inti.git
$ git clone git@github.com:callrail/rowdy.git
$ git clone git@github.com:callrail/looky.git
$ git clone git@github.com:callrail/mercury.git
$ git clone git@github.com:callrail/phony.git
$ git clone git@github.com:callrail/authy.git
```

### 7. Install some dependencies for your environment

We'll install global dependencies using the Brewfile from the setup repo. This will install rbenv, python 2.7, nvm, yarn, libpq, redis, and RabbitMQ. Click through if you want to learn more about those tools. Then we'll install ruby.

Note: during the `$ brew bundle` step below, it's possible not everything will install the first run through, which is fine. often a simple re-run of `$ brew bundle` or `$ brew cleanup && brew bundle` will fix most problems…

```
$ touch ~/.zshrc

$ cd ~/src/setup
$ brew update
$ brew tap homebrew/bundle
$ brew bundle

$ echo "\neval \"\$(rbenv init -)\"\n" >> ~/.zshrc
$ source ~/.zshrc
$ rbenv install 2.7.7
$ rbenv global 2.7.7
$ rbenv rehash
$ gem update --system 3.3.26
```

#### Sanity check NVM:

Once all this is complete, test your nvm installation with `$ nvm version`

If you see our current version number (16.14) or "null", good. Otherwise, the most likely response will be `zsh: command not found: nvm`

If so, check your `~/.zshrc` file. If you *don't* see anything nvm related there, run `brew info nvm` and check the brew CAVEATS output. Following those directions should allow you to update your `~/.zshrc` file & verify nvm is correctly installed by using `nvm -v` or a similar command. Don't forget to call `source ~/.zshrc` again after you make changes to it.

Next, install the version of node defined here.

```
nvm install 16.14
nvm alias default 16.14
yarn install
```

If you still have issues, check the nvm troubleshooting docs here and/or here to resolve most common issues.

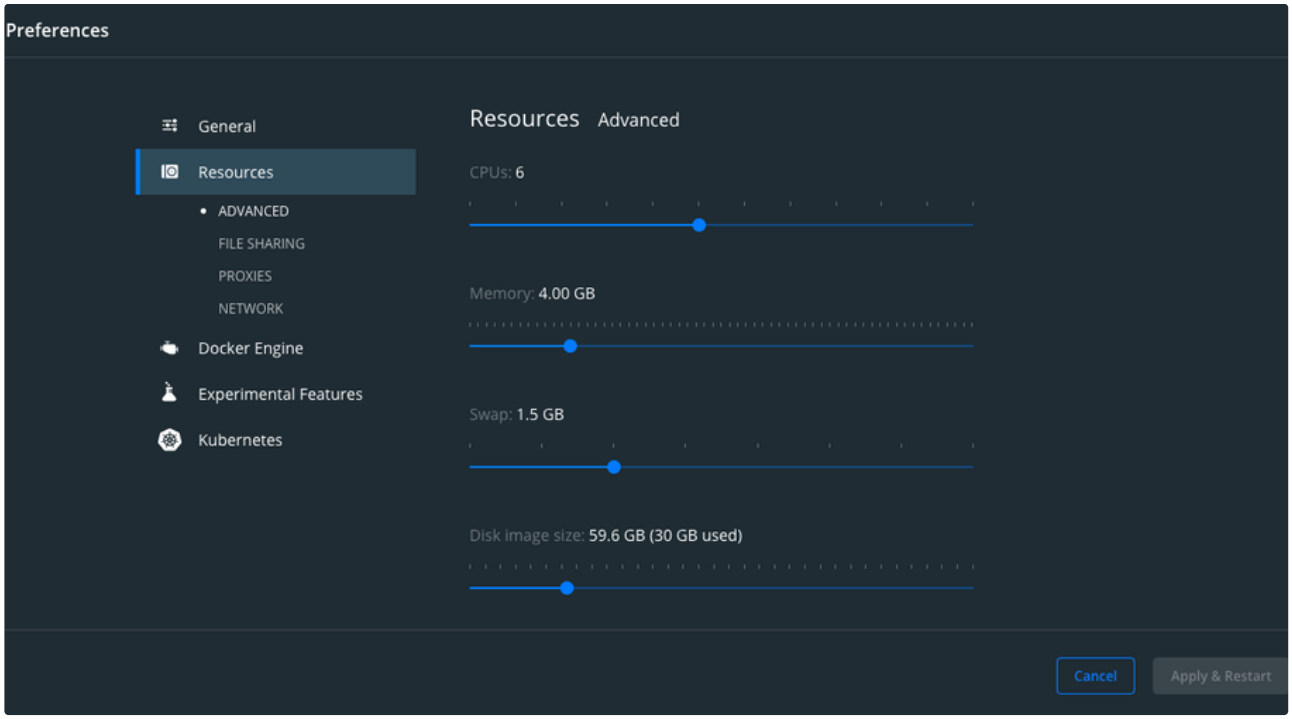
### 8. Install Docker

https://store.docker.com/editions/community/docker-ce-desktop-mac

While Docker no longer requires you to have an account prior to downloading the desktop app, you will need an account to complete the install and use it. Go ahead and create an account using your @callrail.com email address if you don't yet have one. Once your account is created, check with your manager to be added to the Docker org for licensing. Stick with the stable channel when asked.

**Note:** If you have installed Docker before this step, you may run into some issues with configuration. To resolve these, ensure you have the base CallRail Docker config. Docker config is located at `users/your-user-name/.docker/config.json`

Below is a Docker setup that will works for running CallRail in Docker (Docker > Preferences)




### 9. Create your local docker configuration file

1. Copy the example file to your home directory.

```
cp ~/src/setup/copy-to-dockercfg ~/.dockercfg
```

2. Make edits to the new file to correct any path names that don't match your setup. So if your repo's are not in a `/src/` subfolder, you will want to change that in here now. Also check the name of the DOCKER_KEY value and make sure it matches a key in your ~/.ssh folder.

### 10. Generate a GitHub access token for Docker

Docker needs to access GitHub on your behalf.

1. Follow the instructions here for creating a personal access token (classic). Your token will need repository access, so check the box for `repo` and also the box called `read:packages`
2. After you create the token, copy it to your clipboard. You will need it soon
3. You will see a dropdown next to it in GitHub that says `enable SSO`, click+open that dropdown and then click `authorize`. Follow the prompts given to enable SSO.
4. Paste the token into the value of `GITHUB_TOKEN=` in your `~/.dockercfg`

5. Set the your github username to the value of `GITHUB_USER=` in `~/.dockercfg` as well. **Note:** your Github username can be found in Github dashboard by clicking on your avatar in upper-right of page and then clicking “Your profile”. The URL will have your username in it `https://github.com/yourUserName`

### 11. Load your dockercfg from ZSH

```
echo "\n[[ -s \"$HOME/.dockercfg\" ]] && . \"$HOME/.dockercfg\"\n" >> ~/.zshrc
source ~/.zshrc
```

### 12. Install Bingo

`bingo` is a command line interface for interacting with CallRail's AWS environments.

1. To install it, follow this installation guide.
2. To verify its working, run the following (it should output a long list of things)

```
ave cr-stage-dev bingo config list
```

### 13. Get the license key for Sidekiq Pro

Sidekiq Pro requires a license stored in Bundler in order for the gem to install. Copy the value into your `~/.dockercfg` for `SIDEKIQPRO_AUTH` , then source `source ~/.zshrc` and also set it in your global bundler config.

1. get the variable value from config management

```
ave cr-stage-dev bingo config get-param AUTH --context sidekiq | cut -d\= -f2
```

2. set the `SIDEKIQPRO_AUTH` variable to that value in `~/.dockercfg` and save the file
3. source your `~/.zshrc` .

```
source ~/.zshrc
```

4. confirm that `SIDEKIQPRO_AUTH` is accessible in your shell

```
echo $SIDEKIQPRO_AUTH # this should output the value
```

5. Update Bundler

```
bundle config enterprise.contribsys.com $SIDEKIQPRO_AUTH
```

### 14. Start the MacOS Docker application on your laptop.

### 15. Configure Bundler to access CallRail's GitHub Package Registry

Make sure your shell has both of these variables set. If not, source your `~/.zshrc` and echo again

```
echo $GITHUB_TOKEN # this should output the value
echo $GITHUB_USER # this should output the value
```

Now Run the following:

```
bundle config https://rubygems.pkg.github.com/callrail/ $GITHUB_USER:$GITHUB_TOKEN
```

Now you can run `bundle config` to see all the values configured. It can be helpful to run this command in the future if you ever need to delete a value `bundle config --delete <name>` . If you get really stuck and need to start all over you can run `rm -rf`

`~/.bundle/config` to remove all the stored values.

### 16. Conclusion

**bundle Congratulations!** You have completed the local setup of your environment.

The next step is to dive into the domain or service you'll be working on and getting that running locally. This varies greatly based on service, so **you should consult your team for the specific "Getting Started" guide for your service or domain**

One service that almost every developer is sure to work with eventually is the main "CallRail" application (often referred to as **the monolith**). For getting started working with the monolith locally, see Running the CallRail application

## Related

Developing with Docker Locally

Docker Commands

Docker Troubleshooting