# Docker Test Environment

This is a high-level overview of the test configuration using Docker. It explains how to run the tests locally.

## Docker Installation

It is required to have Docker running on your machine in order to build and run the tests in the Dockerfiles. See [https://docs.docker.com/engine/installation/](https://docs.docker.com/engine/installation/) for more information on how to install.

## Convenience Scripts

We have added a number of default run scripts to the `package.json` file to simplify building and running your tests.

### Configuring Docker Images

The following two scripts need to be run first before you can move on to testing:

- `yarn run docker-setup-android` : Pulls down the React Native Community Android image that serves as a base image when building the actual test image.
- `yarn run docker-build-android` : Builds a test image with the latest dependencies and React Native library code, including a compiled Android test app.

### Running Tests

Once the test image has been built, it can be used to run our Android tests.

- `yarn run test-android-run-unit` runs the unit tests, as defined in `scripts/run-android-docker-unit-tests.sh` .
- `yarn run test-android-run-e2e` runs the end to end tests, as defined in `scripts/run-ci-e2e-tests.sh` .
- `yarn run test-android-run-instrumentation` runs the instrumentation tests, as defined in `scripts/run-android-docker-instrumentation-tests.sh` .

#### Instrumentation Tests

The instrumentation test script accepts the following flags in order to customize the execution of the tests:

`--filter` - A regex that filters which instrumentation tests will be run. (Defaults to .*)

`--package` - Name of the java package containing the instrumentation tests (Defaults to com.facebook.react.tests)

`--path` - Path to the directory containing the instrumentation tests. (Defaults to ./ReactAndroid/src/androidTest/java/com/facebook/react/tests)

`--retries` - Number of times to retry a failed test before declaring a failure (Defaults to 2)

For example, if locally you only wanted to run the InitialPropsTestCase, you could do the following: `yarn run test-android-run-instrumentation -- --filter="InitialPropsTestCase"`

## Detailed Android Setup

There are two Dockerfiles for use with the Android codebase. The base image used to build `reactnativecommunity/react-native-android` is located in the [https://github.com/react-native-community/docker-android](https://github.com/react-native-community/docker-android) GitHub repository. It contains all the necessary prerequisites required to run the React Android tests. It is separated out into a separate Dockerfile because these are dependencies that rarely change and also because it is quite a beastly image since it contains all the Android dependencies for running Android and the emulators (~9GB).

The good news is you should rarely have to build or pull down the base image! All iterative code updates happen as part of the `Dockerfile.android` image build.

Lets break it down...

First, you'll need to pull the base image. You can use `docker pull` to grab the latest version of the `reactnativecommunity/react-native-android` base image. This is what you get when you run `yarn run docker-setup-android`.

This will take quite some time depending on your connection and you need to ensure you have ~10GB of free disk space.

Once you have downloaded the base image, the test image can be built using `docker build -t reactnativeci/android -f ./.circleci/Dockerfiles/Dockerfile.android .`. This is what `yarn run docker-build-android` does. Note that the `-t` flag is how you tell Docker what to name this image locally. You can then use `docker run -t reactnativeci/android` to run this image.

Now that you've built the test image, you can run unit tests using what you've learned so far:

```
docker run --cap-add=SYS_ADMIN -it reactnativeci/android bash
.circleci/Dockerfiles/scripts/run-android-docker-unit-tests.sh
```

> *Note: `--cap-add=SYS_ADMIN` flag is required for the `.circleci/Dockerfiles/scripts/run-android-docker-unit-tests.sh` and `.circleci/Dockerfiles/scripts/run-android-docker-instrumentation-tests.sh` in order to allow the remounting of `/dev/shm` as writeable so the `buck` build system may write temporary output to that location.*

Every time you make any modifications to the codebase, including changes to the test scripts inside `.circleci/Dockerfiles/scripts`, you should re-run the `docker build ...` command in order for your updates to be included in your local docker test image.

For rapid iteration, it's useful to keep in mind that Docker can pass along arbitrary commands to an image. For example, you can alternatively use Gradle in this manner:

```
docker run --cap-add=SYS_ADMIN -it reactnativeci/android ./gradlew
RNTester:android:app:assembleRelease
```