1.0 to 2.0 migration guide

Get an overview of the new `.circleci/config.yml` format

- Familiarize yourself with the basics of CircleCI 2.0: https://circleci.com/docs/2.0/basics/
- Check out our list of best practices and useful tips as you work: https://discuss.circleci.com/t/useful-tips-and-best-practices-when-migrating-to-circleci-2-0/23075
- Watch a video overview of CircleCI 2.0 config format: https://www.youtube.com/watch?v=xOSHKNUIkJY

Identify all projects currently building on CircleCI 1.0

- Execute this ruby script to locate all the `circle.yml` files in your organization: https://github.com/CircleCI-Public/find-circle-yml
- Sort your projects by those that need to continue building on CircleCI after 9/1/2018 and those that are inactive.
- Stop building and unfollow all inactive projects.

Create a new branch on your projects for getting 2.0 up and running

For any project that you plan to migrate, start by creating a branch to do your migration work. Our architecture is specifiable at the branch level, so your CircleCI 1.0 builds will continue uninterrupted as you work to update the syntax to CircleCI 2.0.

Begin writing your configuration file for the new 2.0 architecture

- Start by using the config translation API endpoint to generate a `.circleci/config.yml` reference that is specific to your repository: https://circleci.com/docs/2.0/config-translation/
- From there, follow the steps below to build your first `.circleci/config.yml` alongside the generated reference.

Use the CLI to validate your 2.0 configuration (and avoid stacking up failed builds).

- You can use the command line interface to validate your config changes with a pre-commit Git hook. This will help you avoid running a bunch of failed builds while you are fine-tuning your syntax.
- Read how to use the CLI: https://circleci.com/docs/2.0/local-cli/#validate-every-configuration-change
- Watch this video on using the CLI: https://www.youtube.com/watch?v=HB5DehCufG0
Choose your executor type and version
https://circleci.com/docs/2.0/executor-types/

**Docker:**
Use this executor if you need language-specific tools/libraries.
This executor is recommended for most use cases.

**Machine:**
Use this executor if you’re manipulating Docker containers or need root privileges.

**macOS:**
Use this executor if you’re building iOS applications.

If you chose the Docker executor, identify necessary Services, such as databases, caching, queuing, web server, and browser tests

- Pull in the corresponding Docker images for those services.
  https://circleci.com/docs/2.0/executor-types/#using-multiple-docker-images/
- You can select a CircleCI convenience image or build a custom image of your own (ideally extending an official Docker Hub image or a convenience image).
  https://circleci.com/docs/2.0/circleci-images/
  https://circleci.com/docs/2.0/custom-images/
  https://circleci.com/docs/2.0/private-images/
- Consider using remote Docker if you need language-specific tools in addition to using Docker images.

TIP: Keep in mind, the first image is the only image where commands can be run

Specify all necessary commands
Open a previously run green build on CircleCI 1.0 to find all the commands that ran implicitly via inference and the commands that were specified in the UI explicitly.

Create the directory for your artifacts and tests results
Upload them: https://circleci.com/docs/2.0/collect-test-data/

CONGRATULATIONS! Your jobs are passing!
Merge your green build to master.

BONUS: Set up Workflows
If you are building macOS and Linux together in one repository or if you need to run cron jobs, you may want to optimize your build further using Workflows, our job orchestration engine. https://circleci.com/docs/2.0/workflows/