Setting up Circle CI for Ruby app

Back to Articles

UNDER CONSTRUCTION

Overview

This document describes how to use Docker with Circle CI to run continuous integration testing for repos in GitHub.

Changes in App

Setup Docker

- add .dockerignore

```
.git
./tmp
./log
.internal_test_app/*
public/uploads

*.rdb

artifacts/*
coverage/*
```

- add Dockerfile

```
ARG RUBY_VERSION=2.5.8
FROM ruby:$RUBY_VERSION-alpine

## Install dependencies:
## - build-base: (required) To ensure certain gems can be compiled
## - git: (required) Allow bundler to fetch and install ruby gems
## - nodejs: (required) Required by Rails
## - sqlite-dev: (recommended for testing, can be replaced with a different db) For running tests in the
container
## - tzdata: (recommended) add time zone support
## - mariadb-dev: (optional, app db for interactive testing) To allow use of MySQL gem
## - imagemagick: (optional) for image processing
RUN apk add --update --no-cache \
      bash \
      build-base \
      git \
      nodejs \
      sqlite-dev \
      tzdata \
      mariadb-dev \
      imagemagick6-dev imagemagick6-libs

## Set location where the app will live in Docker
WORKDIR /app/cul-it/your_appname-webapp

## Install bundler
RUN gem install bundler:2.1.4

ENV PATH="/app/cul-it/your_appname-webapp:$PATH"
ENV RAILS_ROOT="/app/cul-it/your_appname-webapp"

COPY Gemfile Gemfile.lock ./
RUN gem update --system
RUN bundle install

COPY . .
RUN mkdir -p /app/cul-it/your_appname-webapp/log
RUN echo "" > /app/cul-it/your_appname-webapp/log/debug.log
RUN chmod 666 /app/cul-it/your_appname-webapp/log/debug.log

# I didn't need to precompile assets for testing, so this is commented out
#RUN bundle exec rake assets:precompile

ENV PATH=./bin:$PATH

EXPOSE 3000

## Script runs when container first starts
ENTRYPOINT [ "bin/docker-entrypoint.sh" ]
CMD ["bundle", "exec", "puma", "-v", "-b", "tcp://0.0.0.0:3000"]

# Run tests in terminal:
# ```
# $ docker-compose build
# $ docker-compose up -d
# $ docker-compose exec -w app /app/cul-it/your_appname-webapp sh -c "bundle exec rspec"
# ```
```

- add scripts used during docker startup
    - `bin/db-migrate-seed.sh`

```
#!/bin/sh
set -e

db-wait.sh "$DATABASE_HOST:$DATABASE_PORT"

bundle exec rails db:migrate
bundle exec rails db:seed
```

- bin/db-prepare.sh

```
#! /bin/sh

# If the database exists, migrate. Otherwise setup (create and migrate)
bundle exec rake db:migrate 2>/dev/null || bundle exec rake db:create db:migrate
echo "Database Migration Done!"
```

- bin/db-wait.sh

```
#! /bin/sh

# Wait for MySQL
CMD="nc -z -v -w30 $DATABASE_HOST $DATABASE_PORT"
echo "-----------------------------------"
echo " "
echo "bin/db-wait.sh -- The following command will run to check if MySQL is running..."
echo "bin/db-wait.sh -- $CMD"
echo " "
echo "-----------------------------------"
until nc -z -v -w30 $DATABASE_HOST $DATABASE_PORT; do
  echo 'Waiting for MySQL...'
  sleep 1
done
echo "MySQL is up and running!"
```

- bin/entry_point.sh

```
#!/usr/bin/env bash
# ! /bin/sh

echo '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
echo '---------------------- RUNNING ENTRY POINT ----------------------'
echo '======================================================================='

set -e

# Wait for DB services
sh ./bin/db-wait.sh

# Prepare DB (Migrate if exists; else Create db & Migrate)
sh ./bin/db-prepare.sh

# Run the command defined in docker-compose.yml
exec "$@"
```

- add docker-compose.yml

```
# rails _5.2.4.3_ new .your_appname_app --database=mysql
# rm .your_app_location/config/database.yml # use ENV for database configuration
# docker-compose build
# docker-compose up app
version: '3.2'
services:
app: &app
build:
```

```
context: .
image: cul-it/your_appname:v1
container_name: cul-it-your_appname
stdin_open: true
tty: true
user: root
environment:
- DATABASE_NAME_PREFIX=your_appname
- DATABASE_RAILS_USER=your_appname_user
- DATABASE_RAILS_USER_PW=your_appname_password
- DATABASE_HOST=mysql
volumes:
# - .:/app/cul-it/your_appname-webapp # use for debugging
- app_src:/app/cul-it/your_appname-webapp # use when not debugging
- rails-public:/app/cul-it/your_appname-webapp/public
- rails-tmp:/app/cul-it/your_appname-webapp/tmp
- gems:/usr/local/bundle
ports:
- "3000:3000"
depends_on:
- mysql
- redis
- solr

mysql:
image: mariadb
restart: always
container_name: cul-it-your_appname-mysql
environment:
- MYSQL_ROOT_PASSWORD=your_appname_root_password
- MYSQL_USER=your_appname_user
- MYSQL_PASSWORD=your_appname_password
- MYSQL_DATABASE=your_appname_test
ports:
- "3306:3306"
volumes:
- db-mysql-data:/var/lib/mysql/data

redis:
image: redis:5-alpine
volumes:
- redis:/data

solr:
image: solr:8.7
ports:
- 8988:8983
command:
- sh
- "-c"
- "precreate-core your_appname_test /opt/solr/server/configsets/your_appnameconf; solr-precreate
your_appname /opt/solr/server/configsets/your_appnameconf"
volumes:
- solr_home:/opt/solr/server/solr
- ./solr/config:/opt/solr/server/configsets/your_appnameconf

volumes:
app_src: # comment out when debugging and using . volume for your_appname-webapp
db-mysql-data:
gems:
rails-public:
rails-tmp:
redis:
solr_home:
```

Setup Circle CI

- add .circleci/config.yml

```
version: 2.1
jobs:
  build:
    machine: true # Use a Linux VM instead of docker environment
    working_directory: ~/repo # Default working directory, where your project will be cloned
    steps:
      - checkout
      - run: docker-compose up -d
      - run: docker-compose exec app sh -c "bundle exec rubocop"
      - run: docker-compose exec app sh -c "bundle exec rake db:migrate RAILS_ENV=test"
      - run: docker-compose exec app sh -c "bundle exec rspec"
      # This line is to be able to access failing screenshots in the
      # artifacts section in CircleCI
      - store_artifacts:
          path: ~/repo/features/fail-screenshots
```

- add `/tasks/_MY_APP_NAME_-dev.rake` (e.g. your_appname-dev.rake)

```
# frozen_string_literal: true
require 'bundler/gem_tasks'
require 'rspec/core/rake_task'
require 'rubocop/rake_task'

RSpec::Core::RakeTask.new(:spec)

desc 'Run style checker'
RuboCop::RakeTask.new(:rubocop) do |task|
task.requires << 'rubocop-rspec'
task.fail_on_error = true
end

# desc "Run continuous integration build"
# task ci: ['engine_cart:generate'] do
# Rake::Task['spec'].invoke
# end

desc 'Run continuous integration build'
task ci: ['rubocop', 'spec']

task default: :spec
```

Changes in Github

In the repo settings

Webhooks (left menu)

- Turn off old webhooks
  - click Edit beside webhook
  - uncheck Active
  - click Update webhook button
- Add Travis webhook
  - click Add webhook button (above list)
  - Payload URL: https://notify.travis-ci.org
  - Which events would you like to trigger this webhook? select Let me select individual events

    There may be reasons to do this different, but this is what I general check...

    - Branch or tag creation
    - Branch or tag deletion
    - Collaborator add, remove, or changed
    - Issue comments
    - Pull requests
    - Pushes
    - Repositories
    - Visibility changes
  - click Add webhook button

Branches (left menu)

- if Default branch (e.g. main, dev, etc.) is not listed under Branch protection rules, click Add rule button
    - set Branch name pattern: *default branch's name*
    - check Require status checks to pass before merging
        - check continuous-integration/travis-ci
    - click Save changes button

Changes in Travis

- turn on Travis for the repo
    - login
    - click user icon and select Settings
    - in left menu, select organization
    - in filter, search for repo name
    - click switch button to turn on travis for that repo

Testing that it all works

Travis won't run until the next commit. To test, change a file (e.g. edit README.md and make a minor change), commit and push to GitHub in a branch. This will trigger the Travis build.