TDD for Containers

Why?

How?

Mario Fernandez

Lead Developer

ThoughtWorks

Nov 17 Tech Radar

Nov 17 Tech Radar

> Many development teams have adopted *test-driven development* practices for writing application code because of their benefits.

Nov 17 Tech Radar

> Many development teams have adopted *test-driven development* practices for writing application code because of their benefits.

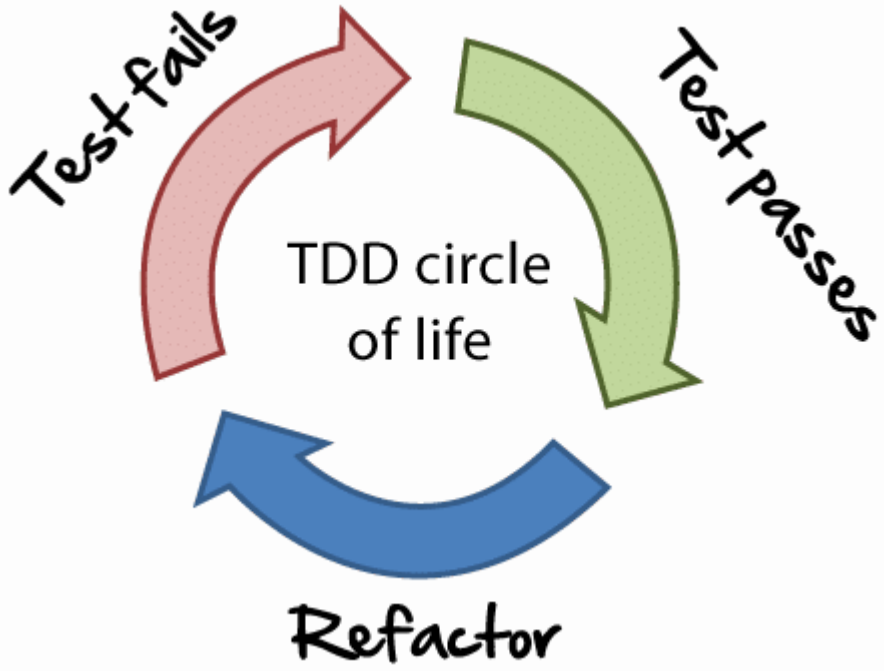
> Others have turned to *containers* to package and deploy their software, and it's accepted practice to use automated scripts to build the containers.

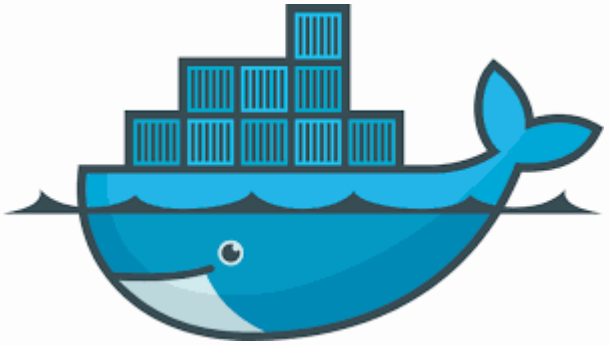
Nov 17 Tech Radar

> Many development teams have adopted *test-driven development* practices for writing application code because of their benefits.

> Others have turned to *containers* to package and deploy their software, and it's accepted practice to use automated scripts to build the containers.

> What we've seen few teams do so far is combine the two trends and drive the writing of the container scripts using tests.


Test fails
Test passes
TDD circle
of life
Refactor

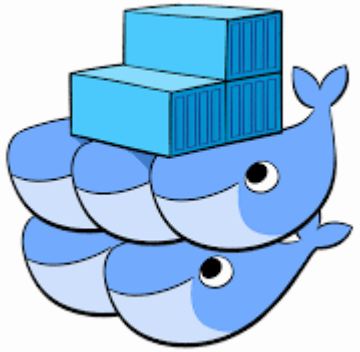


Containers are everywhere

HashiCorp

Nomad

Amazon ECS



Standard approach is try and see what happens

Which can go badly

My first experience with Docker

Setup

8+ years old Ruby monolith

Job to be done

Package the app as a container

Would run in a VM as a sandbox

What we did

What we did

Add stuff to Dockerfile

Wait 40+ mins for it to build

Test manually

Did not work

I don't understand why

Despair

Got to step 1

How it went

Multiple months of work

Flaky, hard to change

Path to production didn't even reach production


FAIL

There has to be a better way

Fast feedback and automation are crucial

Use the right tools

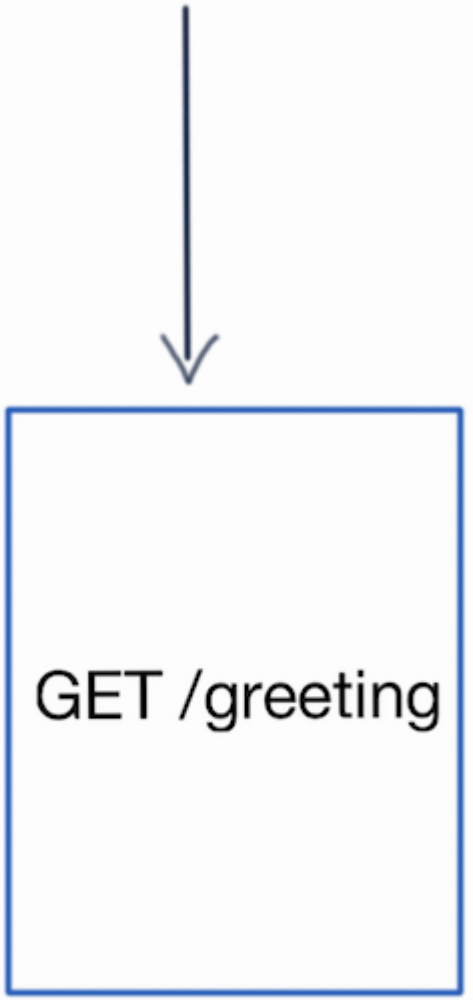

SERVERSPEC

Write RSpec tests for infrastructure

Let's containerize an app

Now with a lot less pain


spring
boot


GET /greeting

Init

```
require 'serverspec'
require 'docker'
require 'rspec/wait'

set :backend, :docker
set :docker_image, 'example-openjdk'

RSpec.configure do |config|
  config.wait_timeout = 60 # seconds
end
```

OS Version

```
describe file('/etc/alpine-release') do
  its(:content) { is_expected.to match(/3.8.2/) }
end
```

OS Version

```
FROM alpine:3.8
```

Java Version

```
describe command('java -version') do
  its(:stderr) { is_expected.to match(/1.8.0_181/) }
end
```

Java Version

```
FROM openjdk:8-jre-alpine3.8
```

JAR

```
describe file('gs-rest-service.jar') do
  it { is_expected.to be_file }
end
```

JAR

```
FROM openjdk:8-jre-alpine3.8

WORKDIR /app
ENV VERSION="0.1.0"

COPY build/libs/gs-rest-service-${VERSION}.jar .
```

App is running

```
describe process('java') do
  it { is_expected.to be_running }
  its(:args) { is_expected.to contain('gs-rest-service.jar') }
end
```

App is running

```
FROM openjdk:8-jre-alpine3.8

WORKDIR /app
ENV VERSION="0.1.0"

COPY build/libs/gs-rest-service-${VERSION}.jar gs-rest-service.jar

CMD ["java", "-jar", "gs-rest-service.jar"]
```

Bound to right port

```
describe 'listens to correct port' do
  it { wait_for(port(8080)).to be_listening.with('tcp') }
end
```

Bound to right port

```
FROM openjdk:8-jre-alpine3.8

WORKDIR /app
EXPOSE 8080
ENV VERSION="0.1.0"

COPY build/libs/gs-rest-service-${VERSION}.jar gs-rest-service.jar

CMD ["java", "-jar", "gs-rest-service.jar"]
```

Not running under root

```
describe process('java') do
  its(:user) { is_expected.to eq('runner') }
end
```

Not running under root

```
FROM openjdk:8-jre-alpine3.8

WORKDIR /app
EXPOSE 8080
ENV VERSION="0.1.0"

COPY build/libs/gs-rest-service-${VERSION}.jar gs-rest-service.jar

RUN adduser -D runner

USER runner
CMD ["java", "-jar", "gs-rest-service.jar"]
```

```
rspec spec/container_spec.rb
Randomized with seed 61858
.......

Top 7 slowest examples (12.86 seconds, 99.9% of total time):
  Application Container java listens to correct port should be listening
    7.82 seconds ./spec/container_spec.rb:20
  Application Container java Process "java" should be running
    4.14 seconds ./spec/container_spec.rb:14
  Application Container java Command "java -version" stderr should match
    0.3948 seconds ./spec/container_spec.rb:10
  Application Container java Process "java" args should contain "gs-rest
    0.15328 seconds ./spec/container_spec.rb:15
  (more output ...)

Finished in 12.87 seconds (files took 1.69 seconds to load)
7 examples, 0 failures
```

Did the tests pass?

Did the tests pass?

Hell yes!

Did the tests pass?

Hell yes!

Cool, let's run them on every commit now

api / build

build #11

started 103d 7h ago
finished 103d 6h ago
duration 3m 23s

11 10 9 8 7 6 5 4 3 2 1

git ref d6ac591366e76b1f00c69a269d9c26c078c8b6f4 ✔

dev-container digest sha256:228c8cd88949edcbf7170d13ee2ae89d78a86f4b40082fcbc7492645813d9fcf ✔

serverspec-container digest sha256:6c73fd7d6d9806677708487f40534c25616ccd0c998b59f553df690c3f623c0d ✔

>_ build ✔

api digest sha256:4aebac174b6c252680174c0bba34259bb00a2fb632a7bb5b525b740222eda544 ✔

api digest sha256:4aebac174b6c252680174c0bba34259bb00a2fb632a7bb5b525b740222eda544 ✔

>_ test ✔

```
17:35:26  Not logged into ECR yet, logging in
17:35:29  Login succeeded
17:35:30  Fetching gem metadata from https://rubygems.org/.......
17:35:30  Resolving dependencies...
17:35:30  Using bundler 1.16.1
17:35:37  Bundled gems are installed into `./vendor/bundle`
17:35:37  Inspecting 3 files
17:35:37  ...
17:35:37
17:35:49  3 files inspected, no offenses detected
17:35:49  .......
17:35:49
17:35:49  Finished in 10.66 seconds (files took 0.53318 seconds to load)
17:35:49  7 examples, 0 failures
17:35:49
```

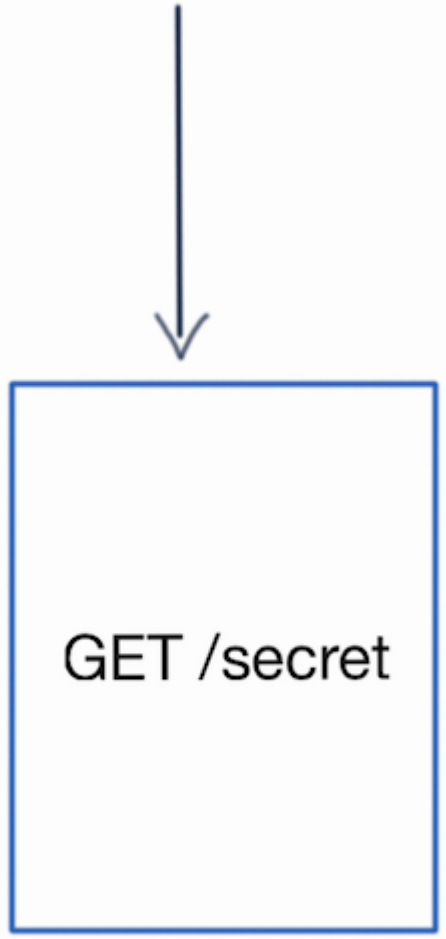
Code

https://github.com/sirech/talk-tdd-infra-redux/tree/master/code

This is but a start

This is not a DSL, but regular Ruby code


node


GET /secret


aws
Secrets Manager

Access secret

```
describe 'fetches a secret' do
  it { wait_for(secret).to match(/the_secret/) }
end

private

def secret
  command('curl localhost:3000/secret').stdout
end
```

Access secret

```
app.get('/secret',
  (req, res) =>
    res.send(`The super secret value is ${process.env.SECRET}`))
```

Injected at runtime

```
describe file('/usr/sbin/entrypoint.sh') do
  it { is_expected.to be_file }
end
```

Injected at runtime

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

set -e

secret=$(
  aws --region "${AWS_REGION}" \
  secretsmanager get-secret-value \
  --secret-id "${SECRET_KEY}" \
  | jq -r .SecretString)
export SECRET="$secret"

exec "$@"
```

Injected at runtime

```
FROM node:11.6-alpine

# ... install awscli
# ... node dependencies
# ... copy app

COPY entrypoint.sh /usr/sbin/entrypoint.sh

RUN adduser -D runner

USER runner
SHELL ["/bin/bash", "-o", "pipefail", "-c"]
ENTRYPOINT ["/usr/sbin/entrypoint.sh"]
CMD ["node", "app.js"]
```

How to make the tests run now?

Localstack

Docker Compose

Dependency setup

```
version: '3'
services:
  localstack:
    container_name: localstack
    image: localstack/localstack

    ports:
      - "4584:4584"

    environment:
      - DEFAULT_REGION=eu-central-1
      - SERVICES=secretsmanager

  ...
```

Dependency setup

```
app:
  container_name: app
  build: ./app

  ports:
    - "3000:3000"

  env_file: .env

  links:
    - localstack
```

Dependency setup

```
require 'docker/compose'
set :docker_container, 'app'

RSpec.configure do |config|
  compose = Docker::Compose.new

  config.before(:all) { compose.up(detached: true, build: true) }

  config.after(:all) do
    compose.kill
    compose.rm(force: true)
  end
end
```

That's a lot of *TDD*

```
  Container File "/usr/sbin/entrypoint.sh" should be file
    1.28 seconds ./spec/container_spec.rb:29
  Container fetches a secret from ASM should match /localstack_secret/
    0.13081 seconds ./spec/container_spec.rb:33

2 examples, 0 failures
```

Code

https://github.com/sirech/example-serverspec-aws

Are you convinced now?

Summary

What: Driving writing container code with tests

Why: Ensure you build high quality images, automate it, fast feedback loop

How: Leverage ServerSpec

Links

- *ServerSpec* resource types: https://serverspec.org/resource_types.html
- https://www.thoughtworks.com/insights/blog/modernizing-your-build-pipelines
- Integrate *ServerSpec* in *Concourse*: https://github.com/sirech/example-concourse-pipeline
- Container example: https://github.com/sirech/talk-tdd-infra-redux/tree/master/code
- Dependencies example: https://github.com/sirech/example-serverspec-aws