Filippo Liverani

@filippo


XPEPPERS


SERVERLESS

WHAT?


There is no cloud
it's just someone else's computer

1. BaaS - Backend as a Service
2. FaaS - Function as a Service

1. BaaS - Backend as a Service
2. FaaS - Function as a Service


VM

CONTAINER

VM

FUNCTION

CONTAINER

VM

CONCEPTS

stateless

event driven

deployment == upload

scaling

restrictions

WHY?

CHANGE OF FOCUS

run code without managing

- your server
- your server applications

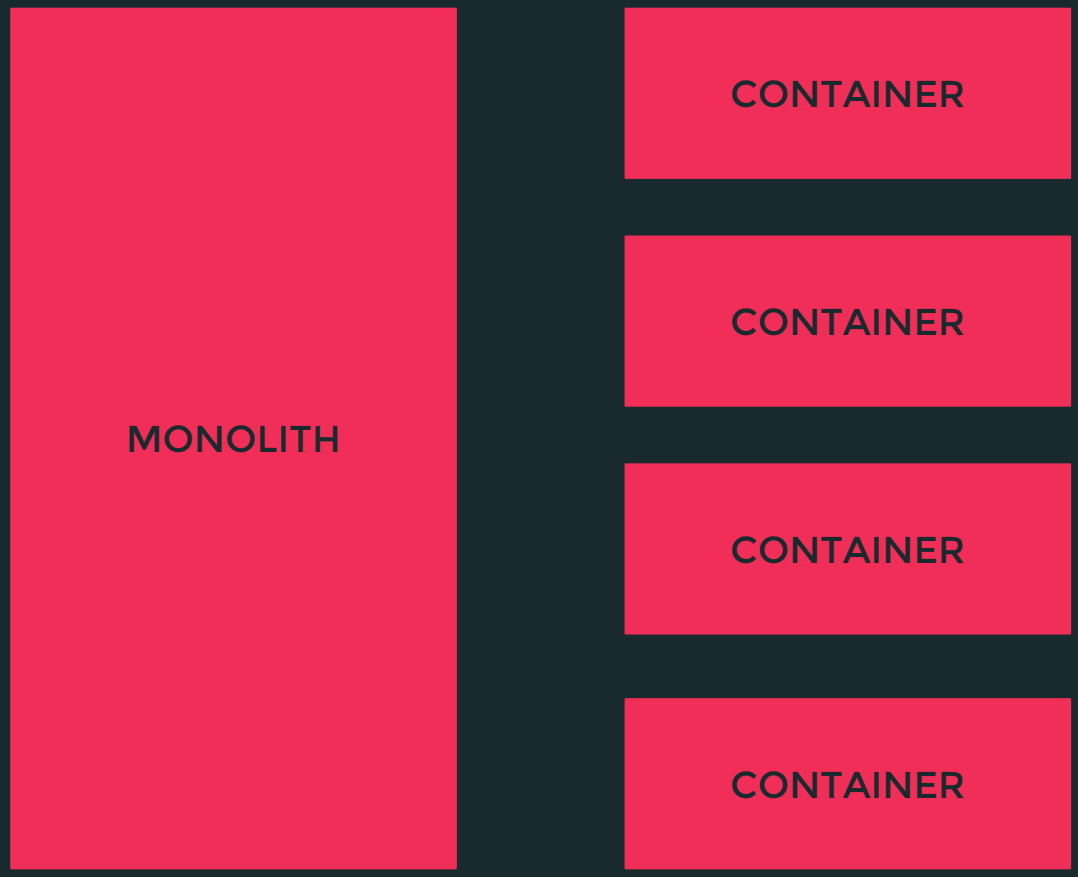
team focus on task

new billing model

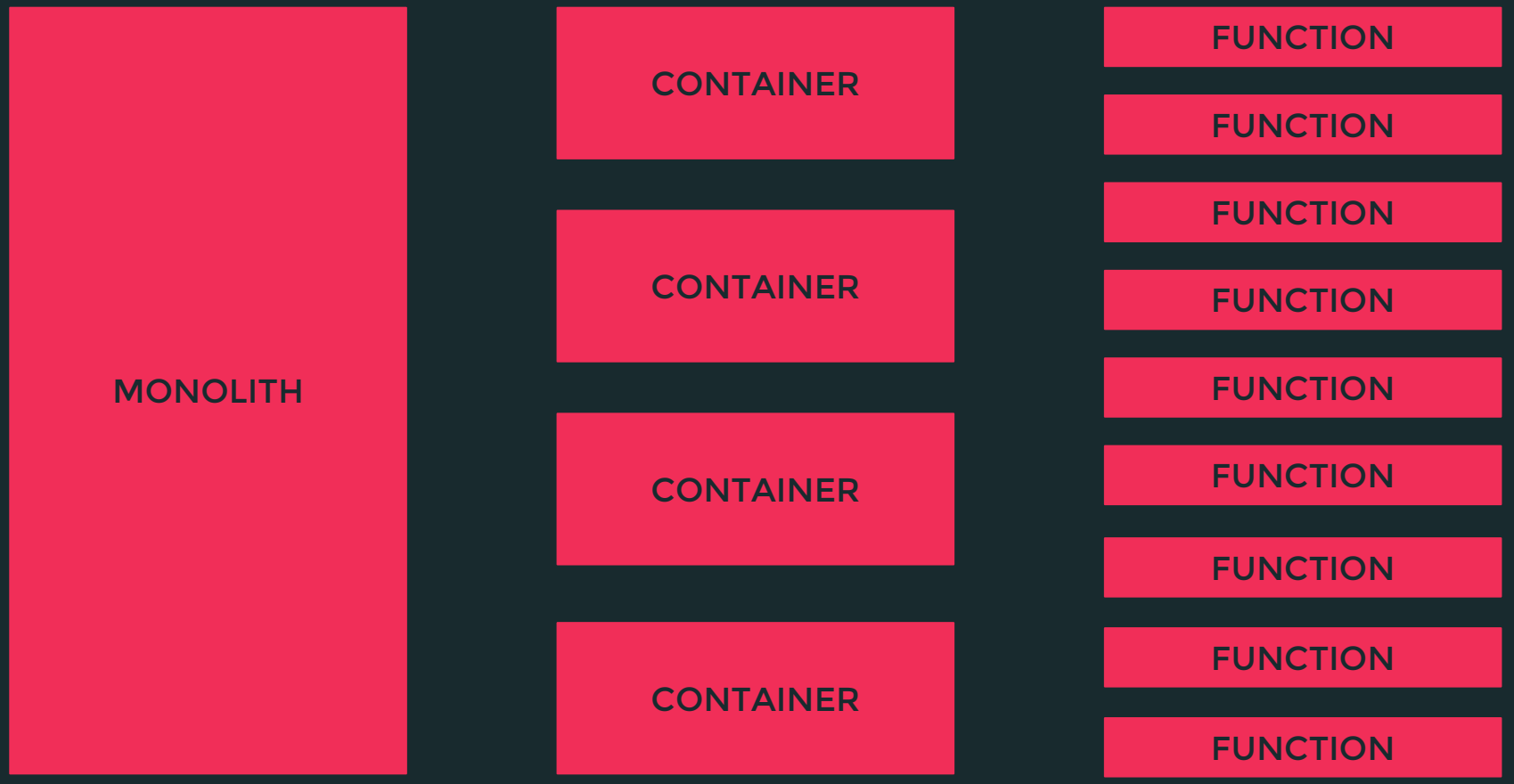
unprecedented speed to market

MICROSERVICES FRIENDLY

MICROSERVICES FRIENDLY



MICROSERVICES FRIENDLY



IS IT WORTH IT?

BENEFITS

operational and scaling costs

fastest scaling

simple deployment

decreased time to market

DRAWBACKS

vendor lock-in and control

restrictions

startup latency

logging , monitoring and debugging

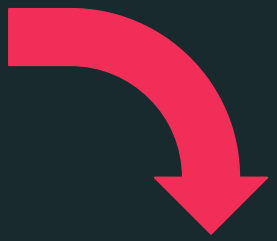
service discovery


TDD

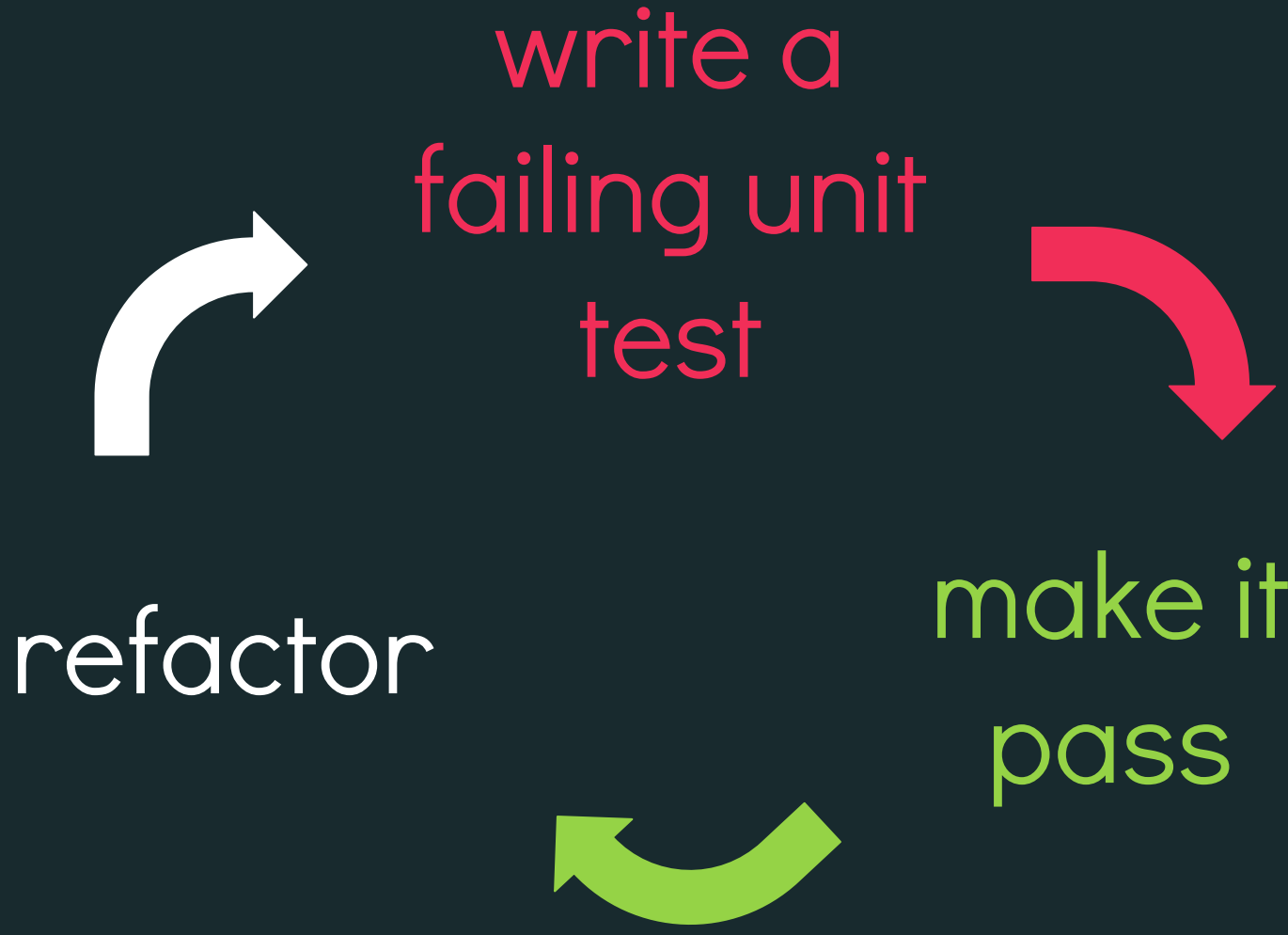
WHAT?

write a
failing unit
test

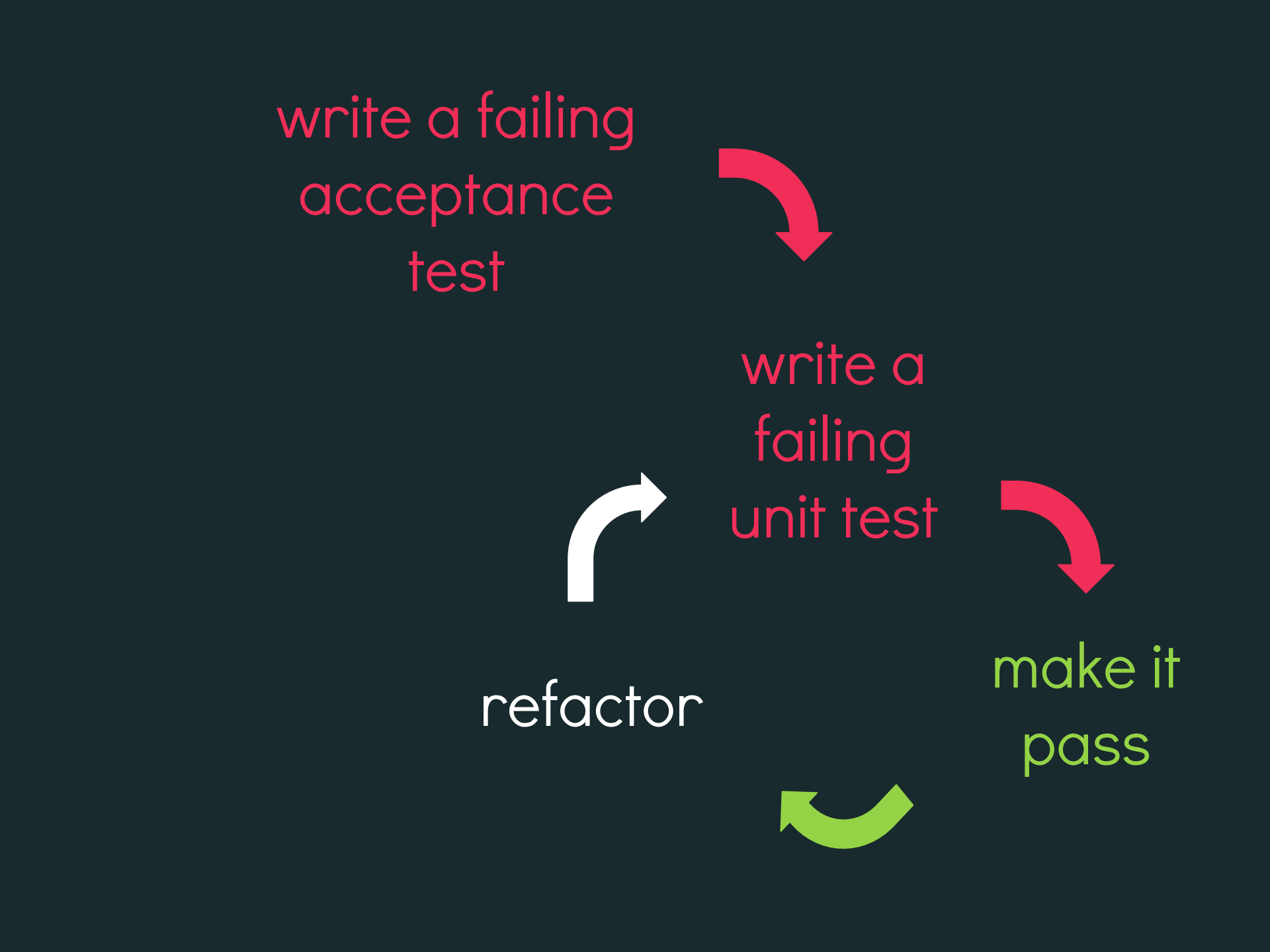
write a
failing unit
test

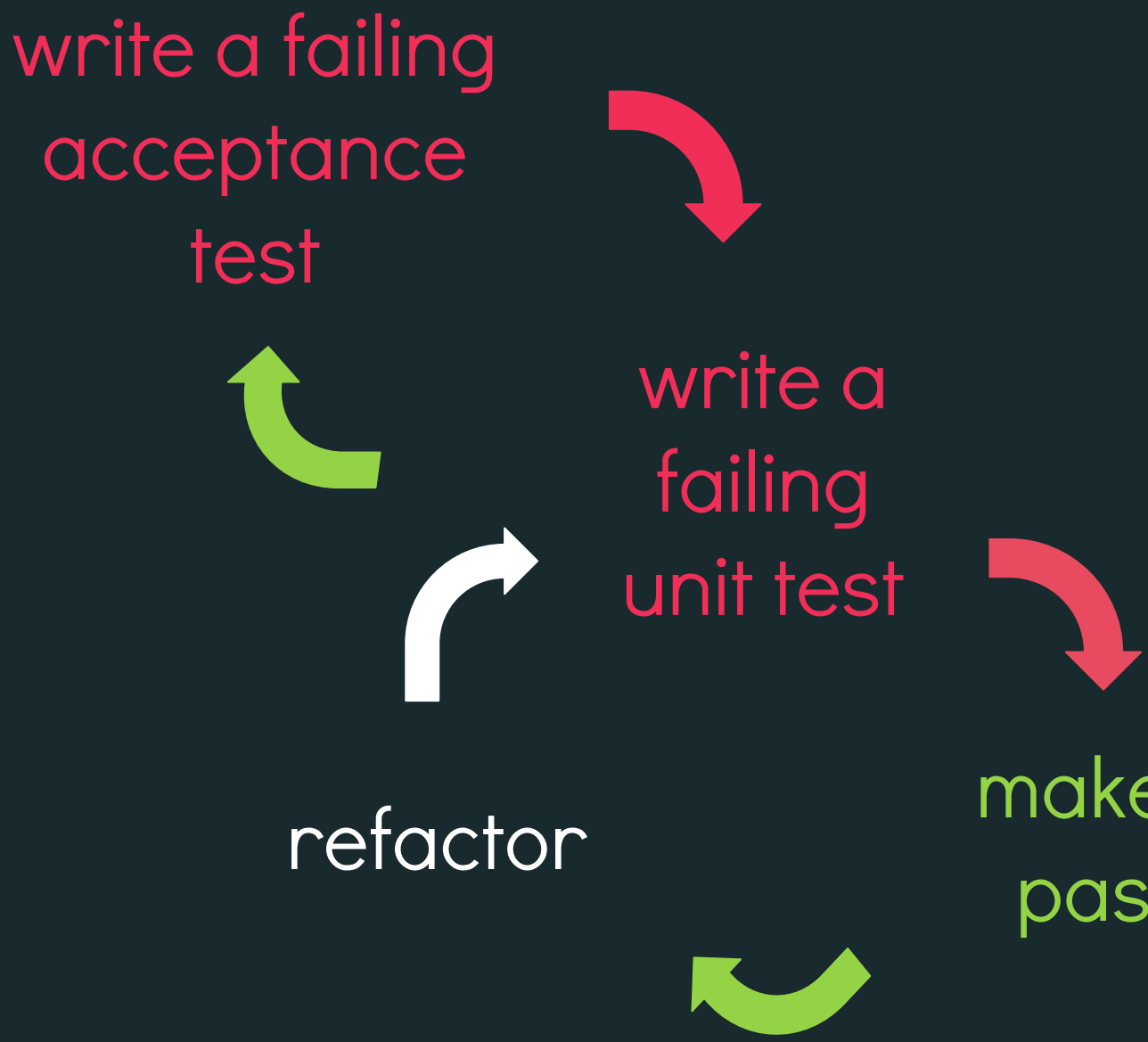


make it
pass


write a
failing unit
test
make it
pass
refactor

write a failing
acceptance
test


write a failing acceptance test
write a failing unit test
make it pass
refactor


write a failing acceptance test
write a failing unit test
make it pass
refactor

WHY?

small incremental changes

continuous validation

rapid feedback

reduces complexity

breaking down a difficult problem
into smaller pieces

it drives design

IS IT WORTH IT?

BENEFITS

higher code quality

regression test suite

feedback and confidence

BENEFITS

implicit acceptance criteria

executable documentation

prevent gold plating

CHALLENGES

takes time to master

requires discipline

changing habits is hard

internal quality perception

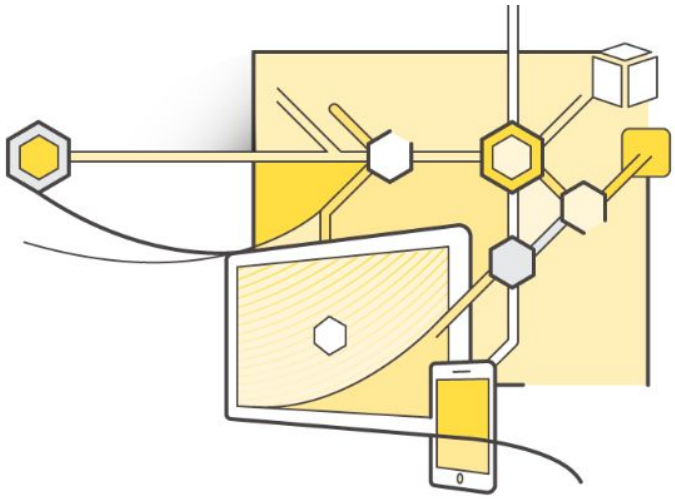

SOME CODE
ENGLISH
JAPAN


Client
HTTP
Movies API

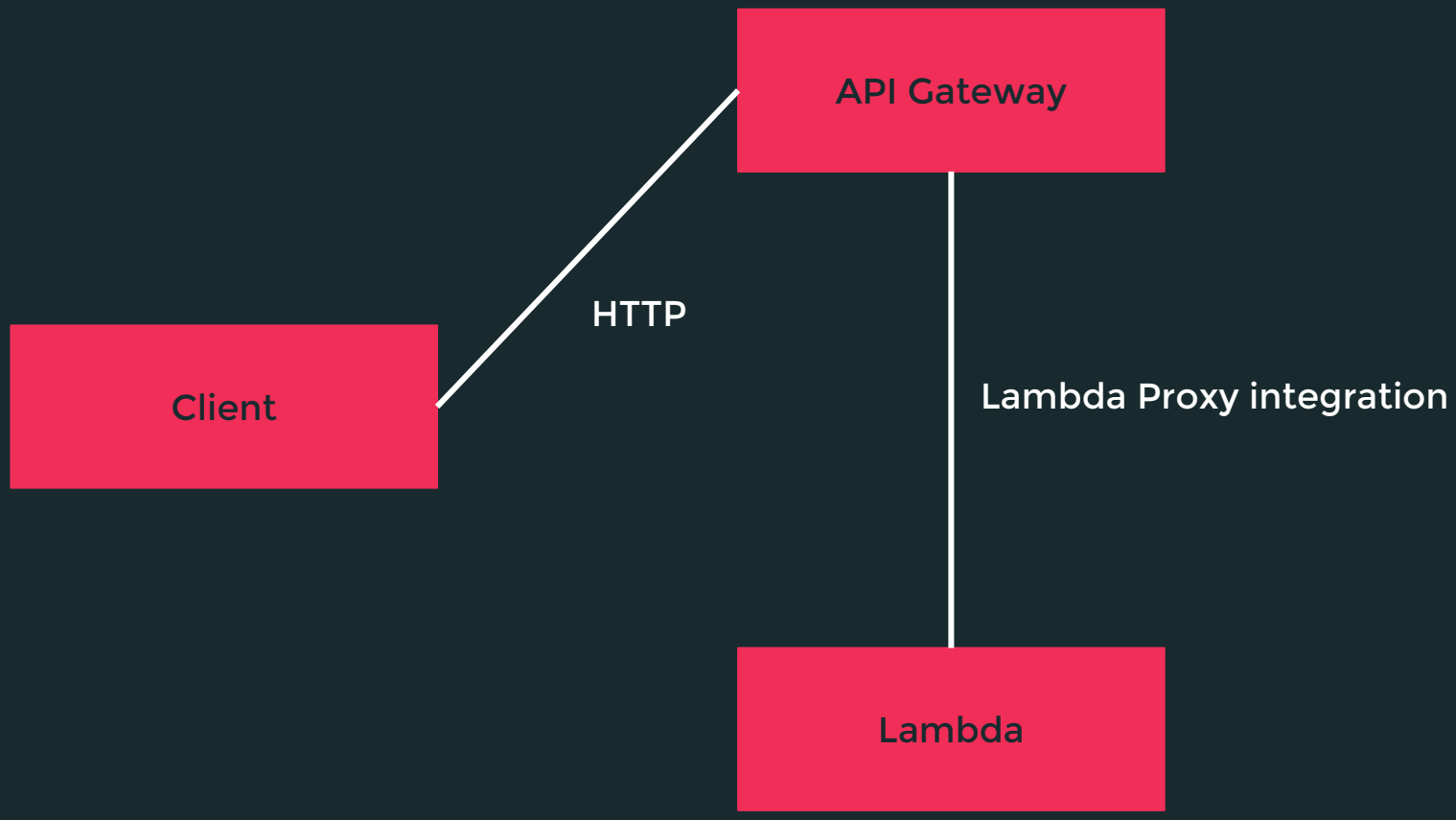
AWS LAMBDA

amazon web services

AWS API GATEWAY

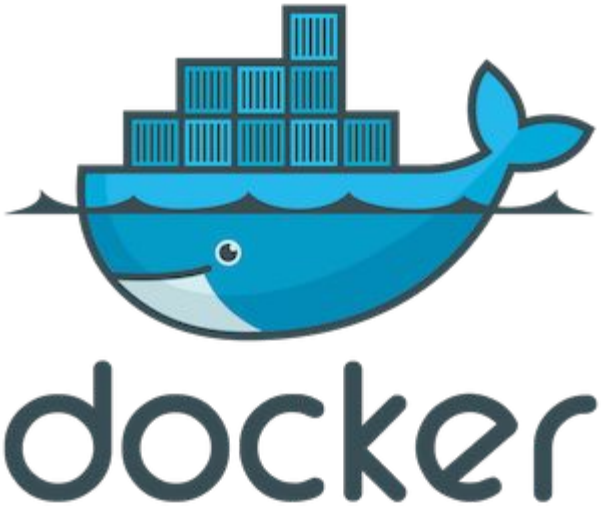


amazon web services


API Gateway
HTTP
Client
Lambda Proxy integration
Lambda

SERVERLESS FRAMEWORK

DOCKER



Dockerfile

```
FROM ambakshi/amazon-linux

RUN curl -k --silent\
  https://nodejs.org/dist/v4.3.2/node-v4.3.2-linux-x64.tar.gz |\
  tar --strip-components 1 -xzf - -C /usr/local/
RUN npm install -g serverless@1.0.3

WORKDIR /home/ec2user

CMD bash
```

```
$ docker build -t xpeppers/lambda .
$ docker run -it -v $(pwd):/home/ec2user xpeppers/lambda
```

```
$ serverless create --template aws-nodejs
```

```
Serverless: Creating new Serverless service...

 _______                             __
|   _   .-----.----.--.--.-----.----|  .-----.-----.-----.
|   |___|  -__|   _|  |  |  -__|   _|  |  -__|__ --|__ --|
|____   |_____|__|  \___/|_____|__| |__|_____|_____|_____|
|   |   |             The Serverless Application Framework
|       |                           serverless.com, v1.0.3
 -------'

Serverless: Successfully created service with template: "aws-nodejs"
Serverless: NOTE: Please update the "service" property in serverless.yml
with your service name
```

serverless.yml

```yaml
service: movie-service
provider:
  name: aws
  runtime: nodejs4.3
  stage: test
  region: eu-central-1
plugins:
  - serverless-offline
functions:
  movies:
    handler: index.movies
    events:
      - http:
          path: movies
          method: get
          cors: true
```

index.js

```
module.exports.movies = (event, context, callback) => {
  const response = {
    statusCode: 200,
    body: '{"message":"Hello!"}'
  };
  callback(null, response);
};
```

```
$ serverless offline start -s development
```

```
Serverless: Starting Offline: test/eu-central-1.

Serverless: Routes for movies:
Serverless: GET /movies

Serverless: Offline listening on http://localhost:3000
```

```
$ curl -v localhost:3000/movies
```

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< vary: origin
< cache-control: no-cache
< content-length: 2
< accept-ranges: bytes
< Date: Fri, 04 Nov 2016 16:24:31 GMT
< Connection: keep-alive

{"message":"Hello!"}
```

```
Serverless: GET /movies (λ: movies)
Serverless: The first request might take a few extra seconds
Serverless: [200]
{"statusCode":200,"body":"{\"message\":\"Hello!\"}"}
```

package.json

```
{
  "name": "serverless-tdd",
  "version": "1.0.0"
  },
  "dependencies": {
    "serverless-offline": "3.2.1"
  },
  "devDependencies": {
    "chai": "3.5.0",
    "mocha": "3.1.2",
    "supertest": "2.0.0",
    "supertest-as-promised": "3.2.0"
  }
}
```

```
$ npm install
$ mkdir -p test/acceptance
```

test/acceptance/get-movies-test.js

```
const endpoint  = process.env.ENDPOINT
  ? process.env.ENDPOINT : 'http://localhost:3000';
const client = supertest(endpoint);

describe('get movies', function() {
  it('return movie list', function() {
    const movies = require('../../movies.json');

    return client.get('/movies')
      .expect(200)
      .then((res) => {
        expect(res.body).to.eql(movies);
      });
  });
});
```

movies.json


```
[{
  "Title":"2001: A Space Odyssey",
  "Year":"1968",
  "Rated":"G",
  "Released":"12 May 1968",
  "Runtime":"149 min",
  "Genre":"Adventure, Mystery, Sci-Fi",
  "Director":"Stanley Kubrick",
  "Writer":"Stanley Kubrick (screenplay), Arthur C. Clarke (screenplay)",
  "Actors":"Keir Dullea, Gary Lockwood, William Sylvester, Daniel Richter",
...
```

```
$ node_modules/.bin/mocha test/acceptance
```

```
get movies
  1) return movie list

  0 passing (28ms)
  1 failing

1) get movies return movie list:
   AssertionError: expected { message: 'Hello!' } to deeply equal [ Array(4) ]
    at test/acceptance/get-movies-test.js:12:29
```

index.js

```
module.exports.movies = (event, context, callback) => {
  const movieService = new MovieService();
  movieService.getMovies()
    .then((response) => {
      callback(null, response);
    });
};
```

```
$ mkdir -p test/unit
```

test/unit/movie-service-test.js

```
describe('MovieService', function() {
  it('get movies from repository', function() {
    const movies = [{Title: "Test Title"}];
    const response = {
      statusCode: 200,
      body: '[{"Title":"Test Title"}]'
    };
    const movieRepository = sinon.stub();
    movieRepository.findAll = sinon.stub().returns(Promise.resolve(movies));
    const movieService = new MovieService(movieRepository);

    return expect(movieService.getMovies()).to.eventually.eql(response);
  });
});
```

```
$ node_modules/.bin/mocha test/unit
```

```
MovieService
  1) get movies from repository

1 failing

1) MovieService get movies from repository:
   TypeError: MovieService is not a function
   at Context.<anonymous> (test/unit/movie-service-test.js:15:26)
```

lib/movie-service.js

```
class MovieService {
  constructor(movieRepository) {
    this.movieRepository = (typeof movieRepository !== 'undefined')
      ? movieRepository : new MovieRepository();
  }

  getMovies() {
    return this.movieRepository.findAll()
      .then((movies) => {
        return {
          statusCode: 200,
          body: JSON.stringify(movies)
        };
      });
  }
}
```

```
$ node_modules/.bin/mocha test/unit
```

```
MovieService
    ✓ get movies from repository

1 passing (21ms)
```

```
$ mkdir -p test/integration
```

test/integration/movie-repository-test.js

```
describe('MovieRepository', function() {
  it('loads movies from file', function() {
    const movies = [{ Title: 'Test Title' }];
    movieRepository = new MovieRepository('test/integration/test-movies.json');

    return expect(movieRepository.findAll()).to.eventually.eql(movies);
  });
});
```

test/integration/test-movies.json

```
[{ "Title": "Test Title" }]
```

```
$ node_modules/.bin/mocha test/integration
```

```
MovieRepository
   1) loads movies from file

1 failing

1) MovieRepository loads movies from file:
   TypeError: MovieRepository is not a function
     at Context.<anonymous> (test/integration/movie-repository-test.js:9:23)
```

lib/movie-repository.js

```
class MovieRepository {
  constructor(filename) {
    this.filename = (typeof filename !== 'undefined')
      ? filename : 'movies.json';
  }

  findAll() {
    return new Promise((resolve, reject) => {
      fs.readFile(this.filename, 'utf8', (err, data) => {
        if (err) reject(err);
        else resolve(JSON.parse(data));
      });
    });
  }
}
```

```
$ node_modules/.bin/mocha test/integration
```

```
MovieRepository
  ✓ loads movies from file

1 passing (25ms)
```

```
$ node_modules/.bin/mocha test/**
```

```
get movies
  ✓ return movie list (57ms)

MovieRepository
  ✓ loads movies from file

MovieService
  ✓ get movies from repository

3 passing (99ms)
```

```
$ curl -v localhost:3000/movies
```

```
< HTTP/1.1 200 OK
< Content-Type: application/json
< vary: origin
< cache-control: no-cache
< content-length: 2
< accept-ranges: bytes
< Connection: keep-alive

[{"Title":"2001: A Space
Odyssey","Year":"1968","Rated":"G","Released":"12 May
1968","Runtime":"149 min","Genre":"Adventure, Mystery,
Sci-Fi","Director":"Stanley Kubrick","Writer":"Stanley Kubrick
(screenplay), Arthur C. Clarke (screenplay)","Actors":"Keir
Dullea, Gary Lockwood, William Sylvester, Daniel Richter", ...
```

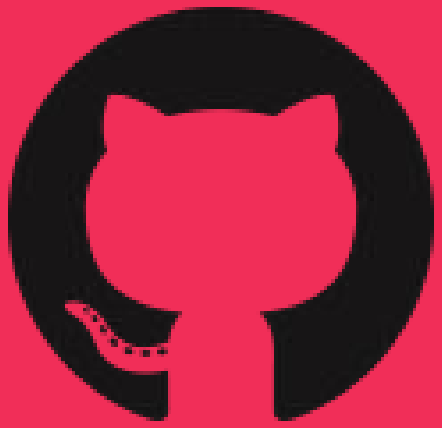
```
$ export AWS_ACCESS_KEY_ID=<key>
$ export AWS_SECRET_ACCESS_KEY=<secret>
$ serverless deploy
```

```
Serverless: Packaging service...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading service .zip file to S3...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
.........
Serverless: Stack update finished...
Service Information
service: movie-service
stage: test
region: eu-central-1
api keys:
  None
endpoints:
  GET - https://xyz.execute-api.eu-central-1.amazonaws.com/test/movies
functions:
  movie-service-test-movies:
arn:aws:lambda:eu-central-1:422553113847:function:movie-service-test-movies
```

```
$ export
ENDPOINT=https://xyz.execute-api.eu-central-1.amazonaws.com/test
$ node_modules/.bin/mocha test/acceptance
```

```
get movies
  ✓ return movie list (906ms)

1 passing (919ms)
```



xpeppers/serverless-tdd


SERVERLESS + TDD
WE CAN DO IT!


THANKS!