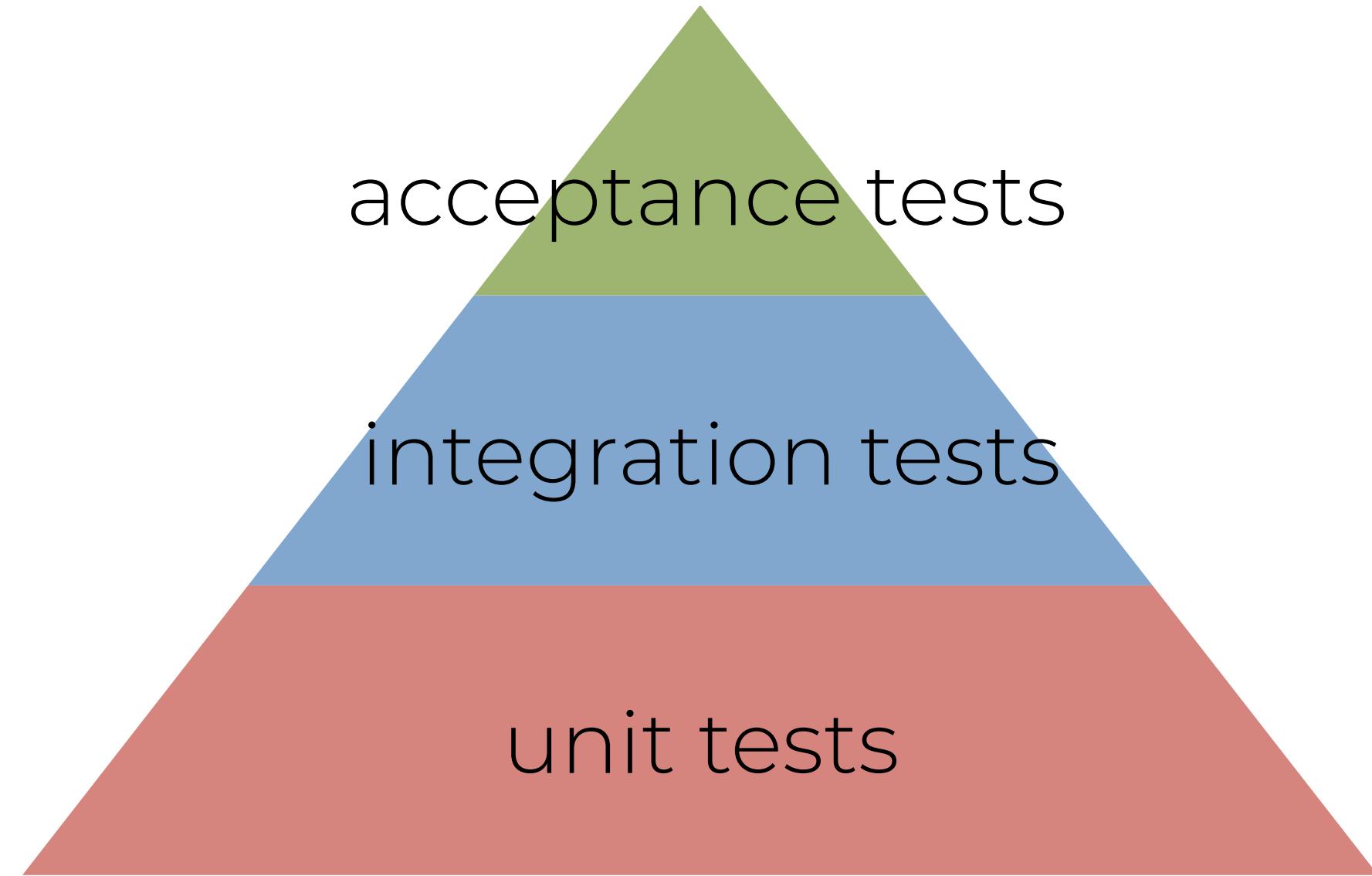


test pyramid



(traditional) test pyramid

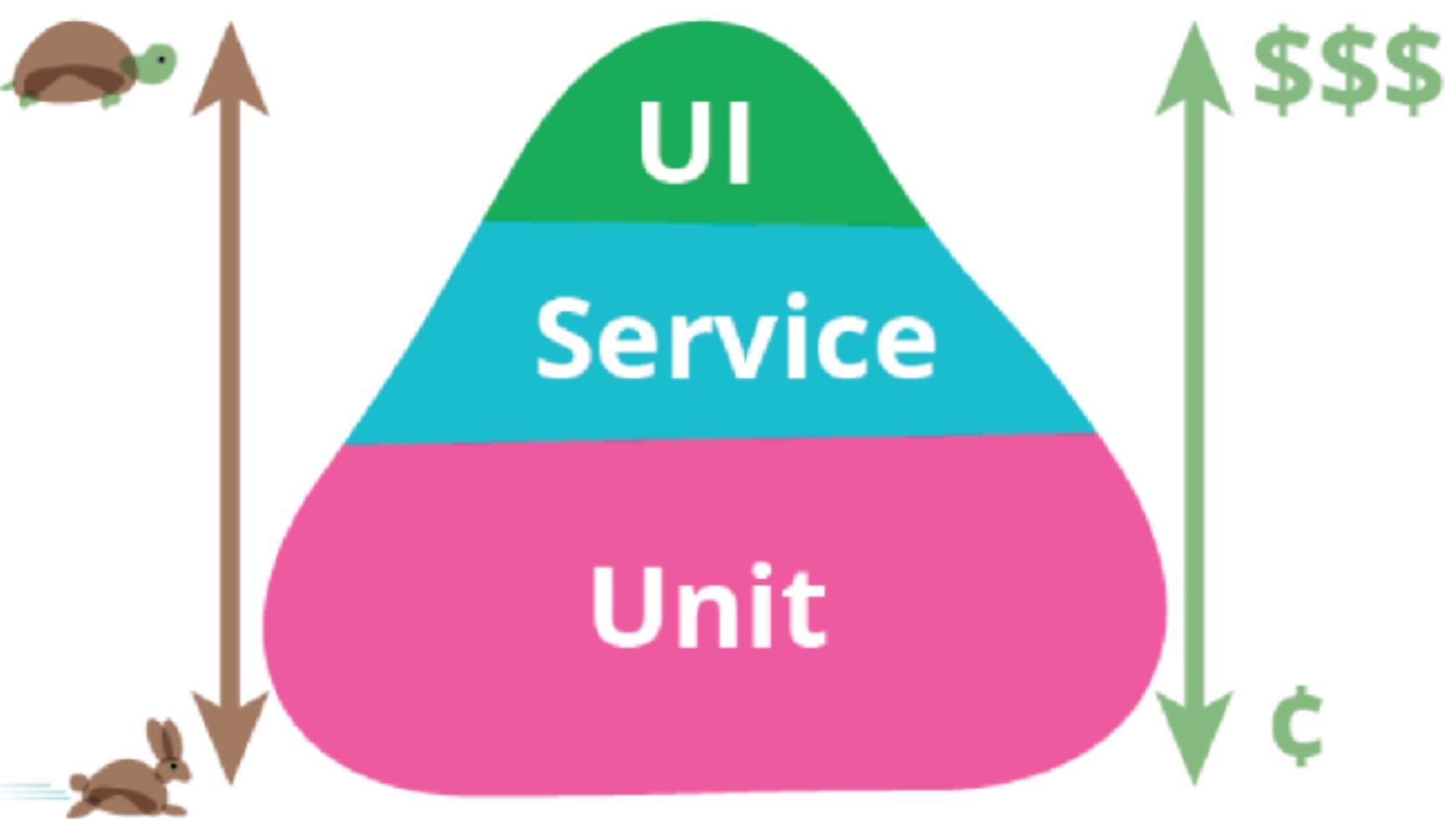
MARTINFOWLER.COM

Intro Videos Design Agile Refactoring FAQ About Me All Sections   

TestPyramid

 Martin Fowler
1 May 2012

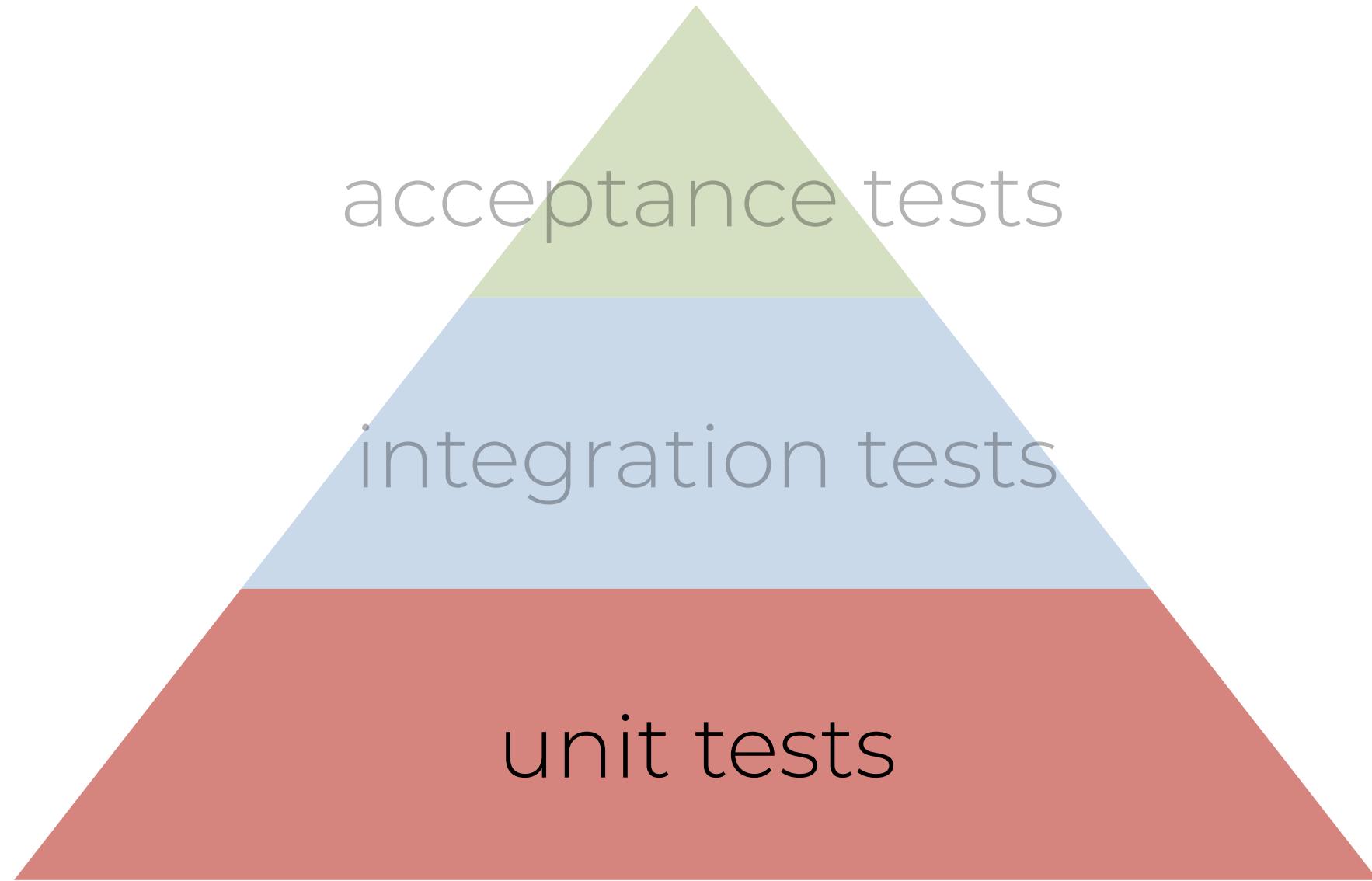
The test pyramid is a way of thinking about different kinds of automated tests should be used to create a balanced portfolio. Its essential point is that you should have many more low-level [UnitTests](#) than high level [BroadStackTests](#) running through a GUI.



Find similar articles at the tag

[testing](#)

test pyramid

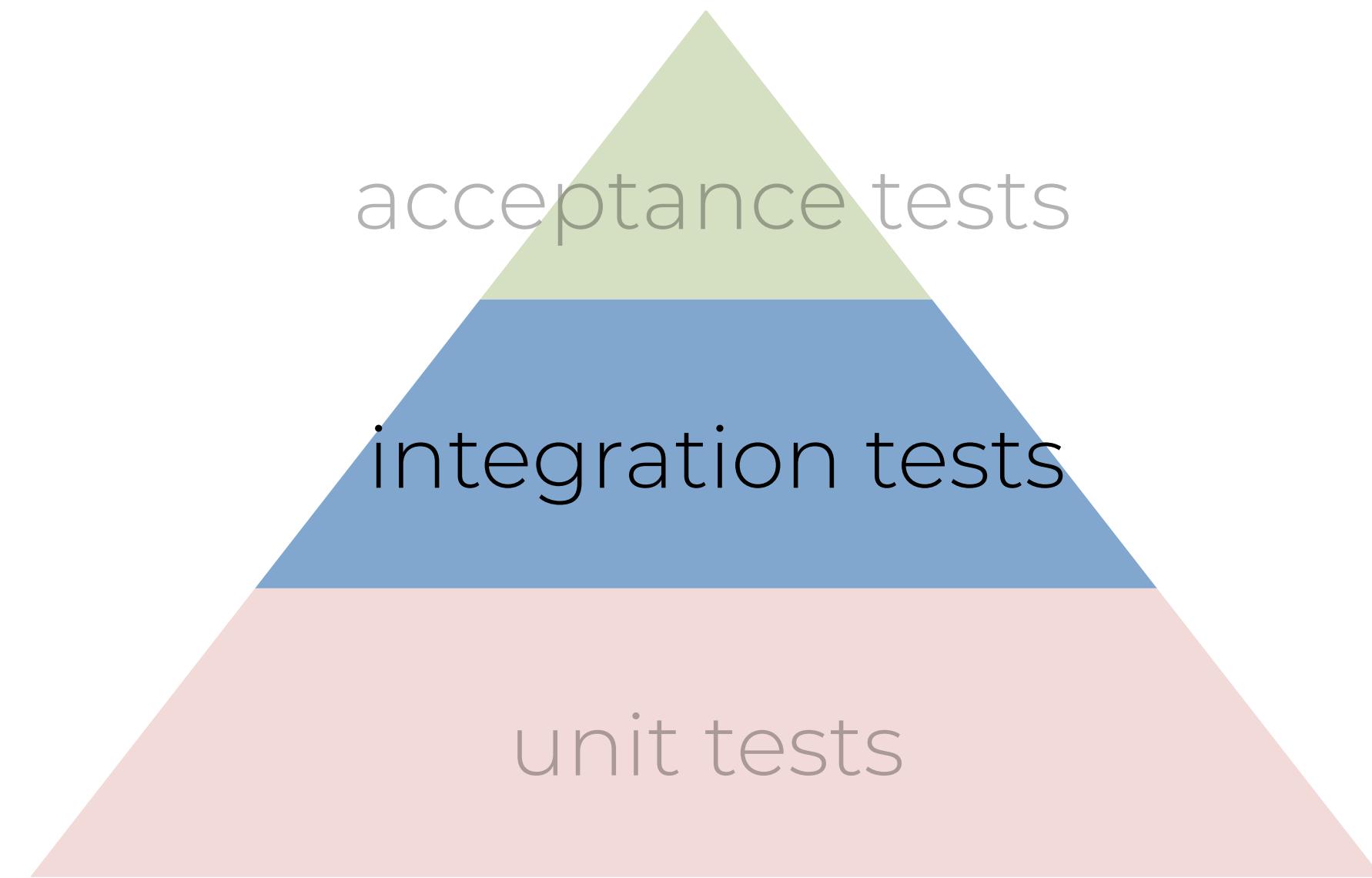


do our objects do the right thing?
are they easy to work with?

```
const APP_ROOT = '.../../';
const ironBorn = require(`.${APP_ROOT}/lib/ironBorn`);
const expect = require('chai').expect;

describe(`What's dead`, () => {
  it(`May never die`, () => {
    expect(ironBorn.die({ isDead: true })).to.be.false;
  });
});
```

test pyramid



does our code work against code we can't change?

```
module.exports.handler = function(event, context, cb) {  
  cb(null, { is_here: true });  
}
```

← handler

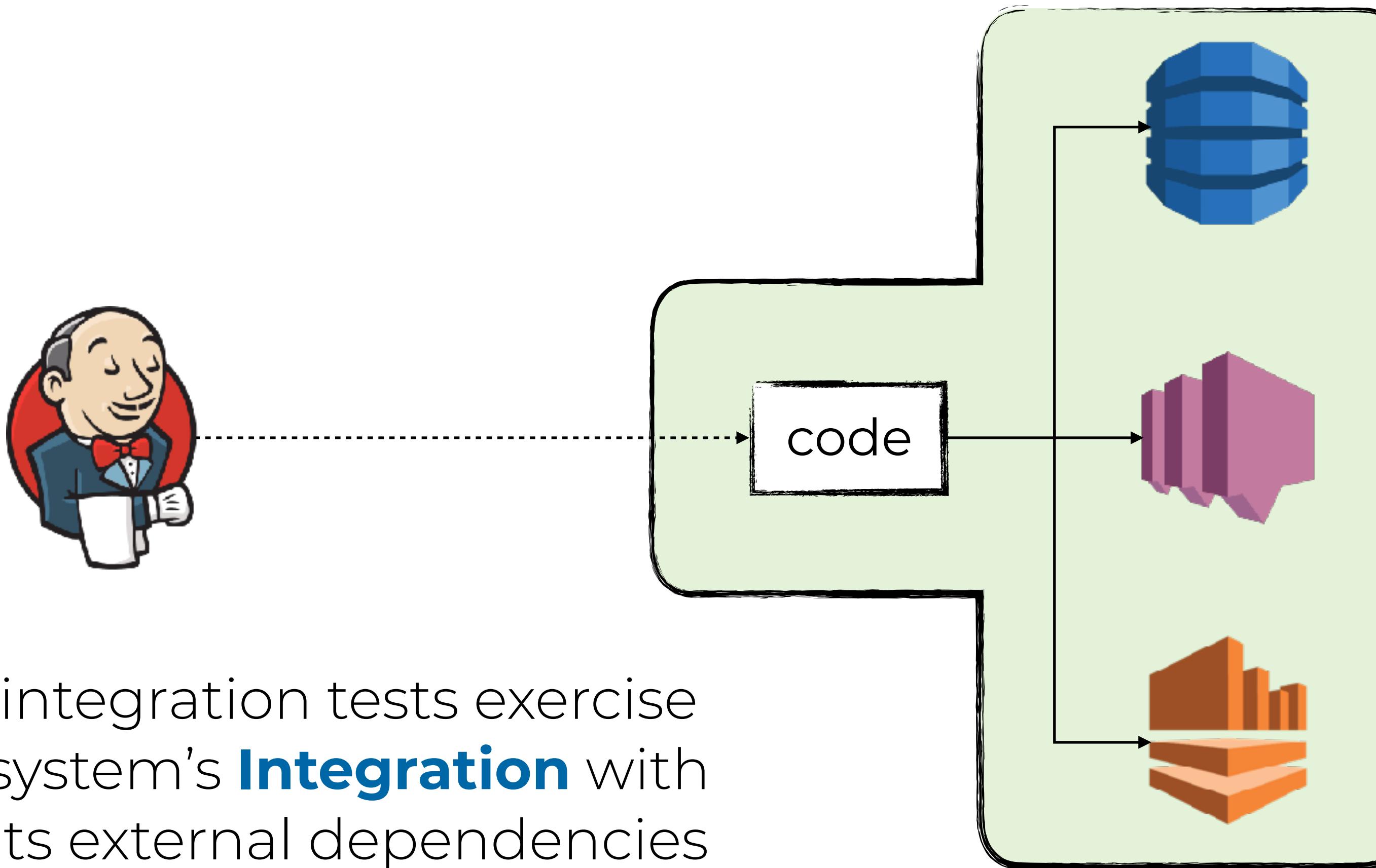
```
module.exports.handler = function(event, context, cb) {  
  cb(null, { is_here: true });  
}  
  
// ↓ handler
```

```
const handler = require('../functions/ned').handler;  
const expect = require('chai').expect;
```

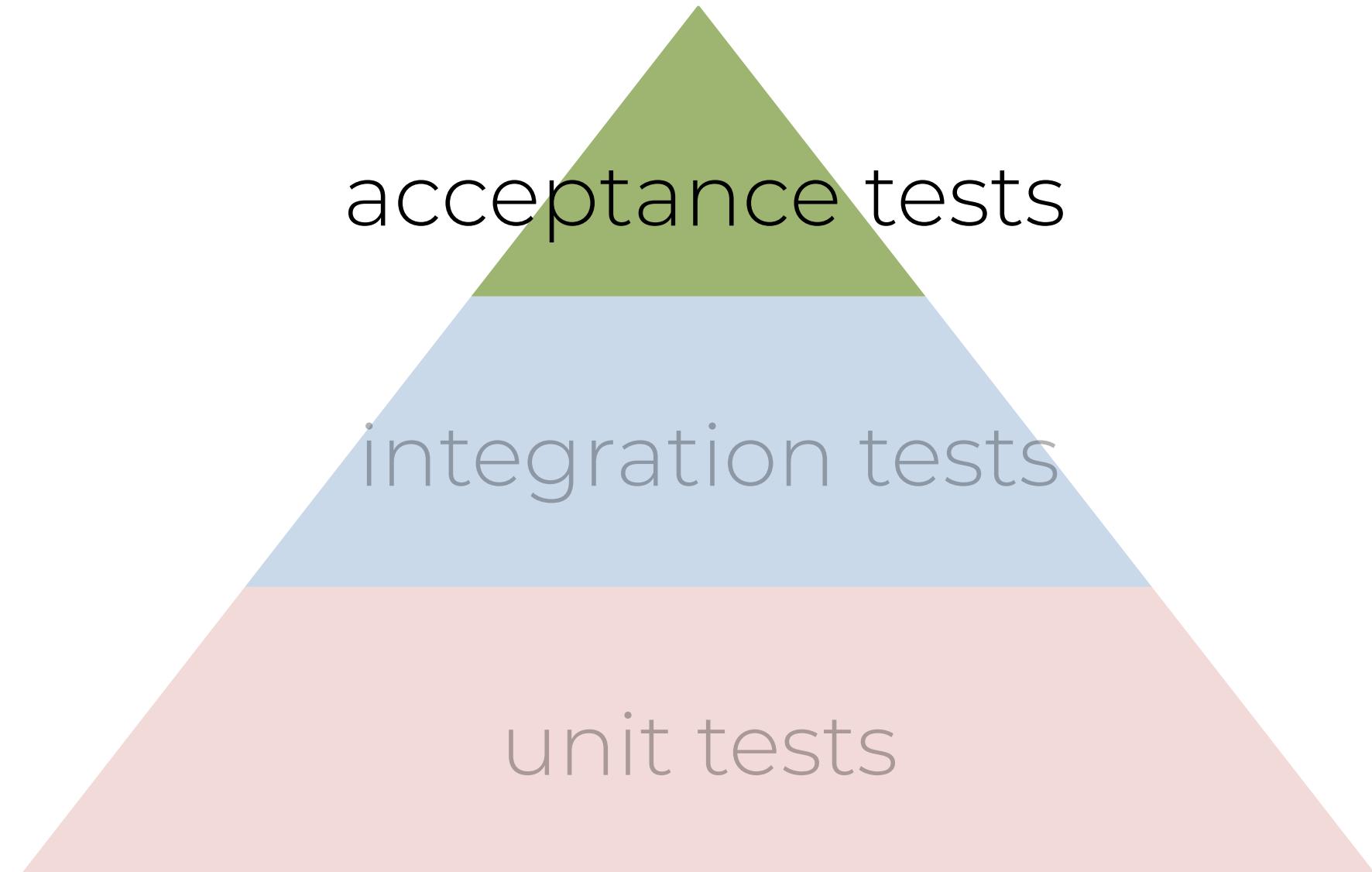
```
describe('winter', () => {  
  it('should be here', done => {  
    let context = {  
      // .. request id, etc.  
    };  
  
    let event = {  
      // eg. mock an API Gateway invocation event  
    };  
  
    let cb = function(err, result) {  
      expect(err).to.be.null;  
      expect(result.is_here).to.be.true;  
  
      done(); // ends test with success  
    };  
  
    // ↓ test by invoking the handler  
    handler(event, context, cb);  
  });  
});
```

← test by invoking the
handler

integration tests

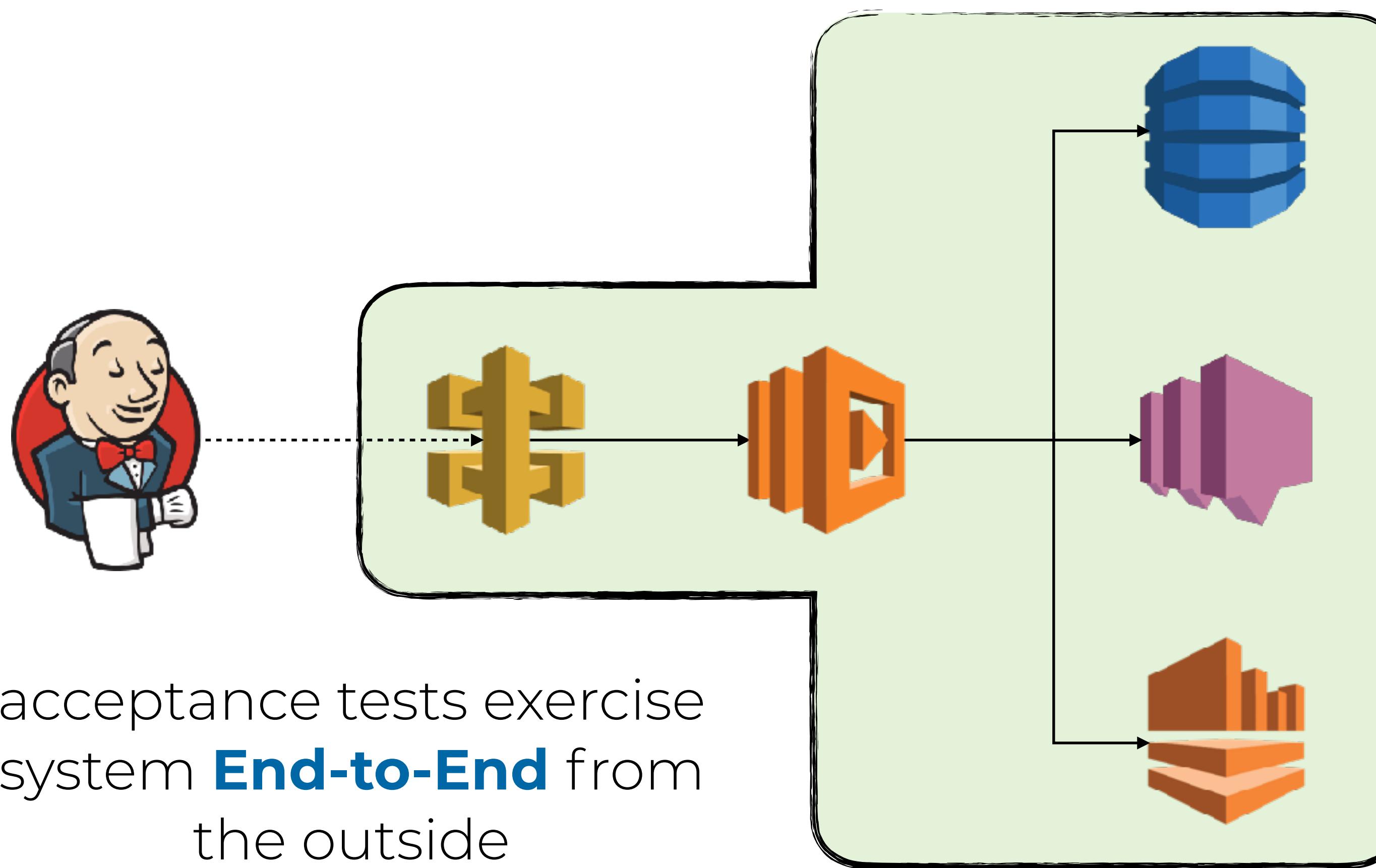


test pyramid

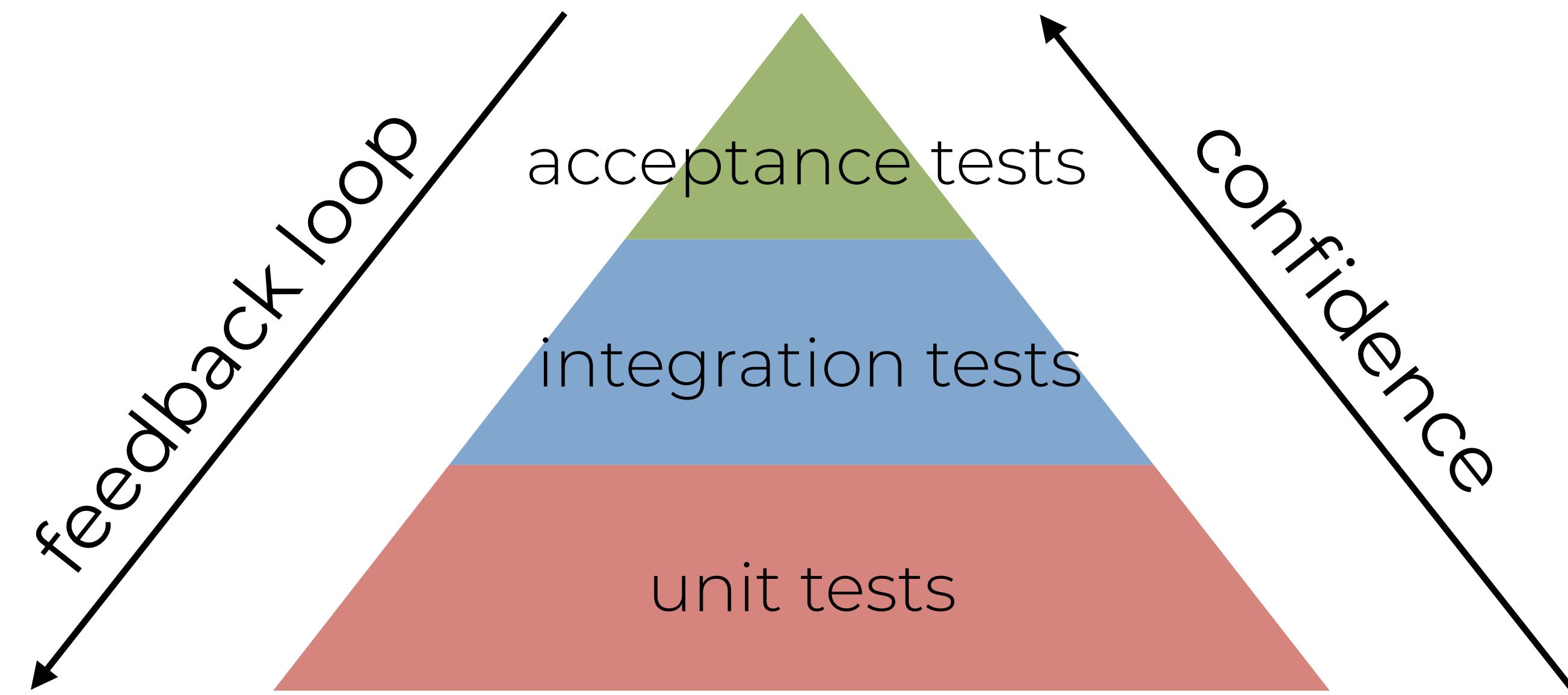


does the whole system work?

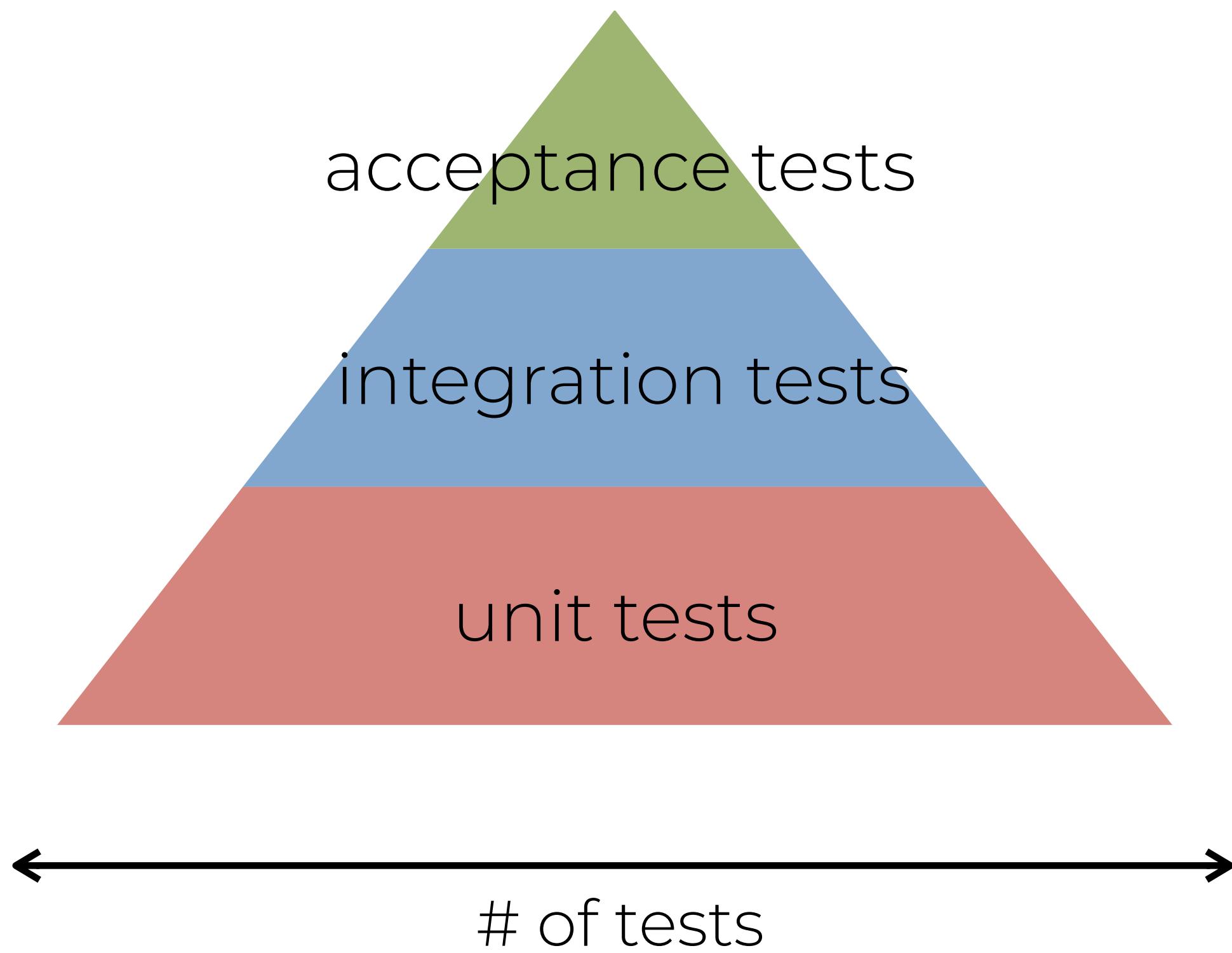
acceptance tests

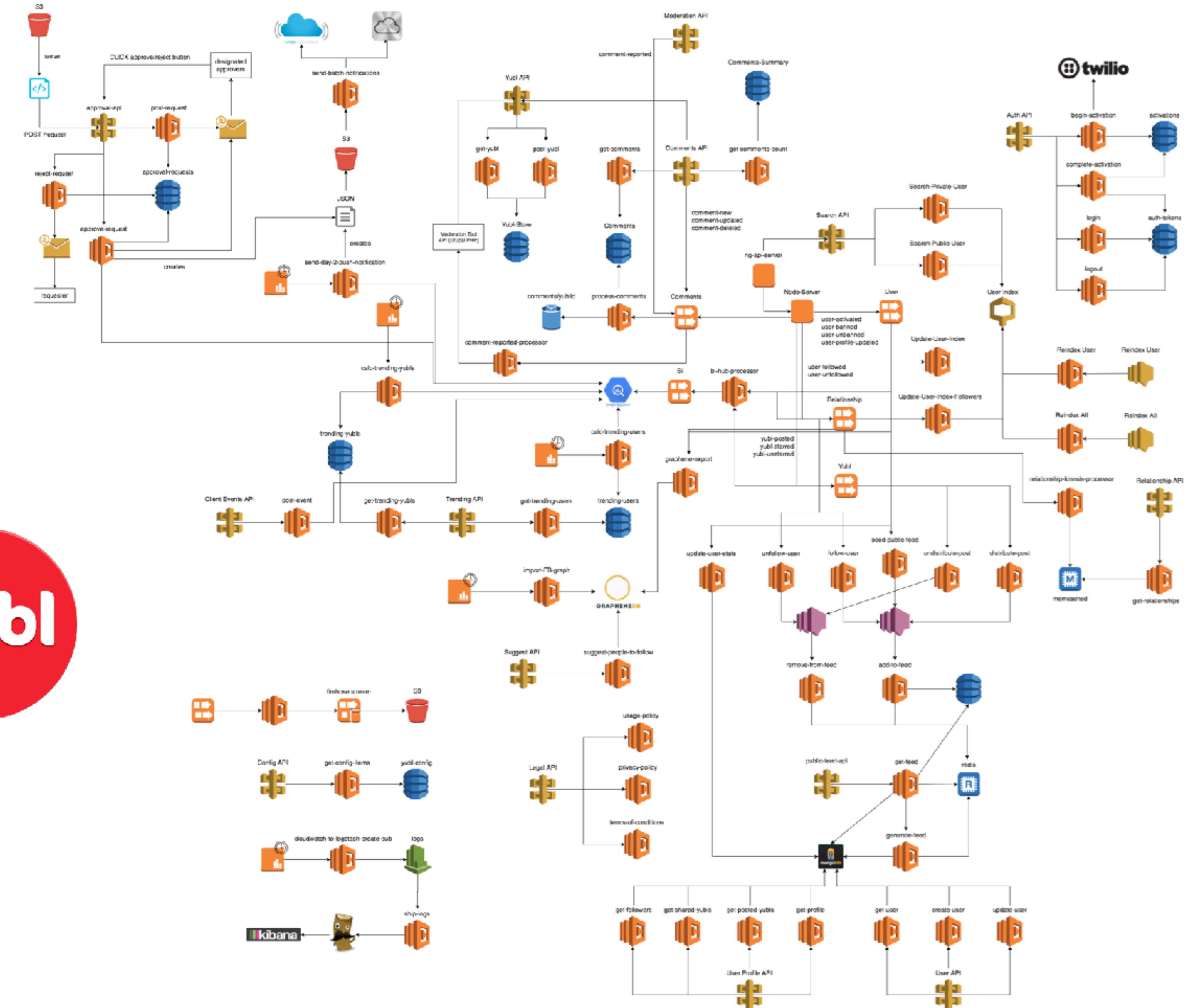


test pyramid



test pyramid





serverless

observation #1

you use more managed services
when working with AWS Lambda

serverless

observation #2

most Lambda functions are simple,
and have a single purpose

serverless

conclusion #1

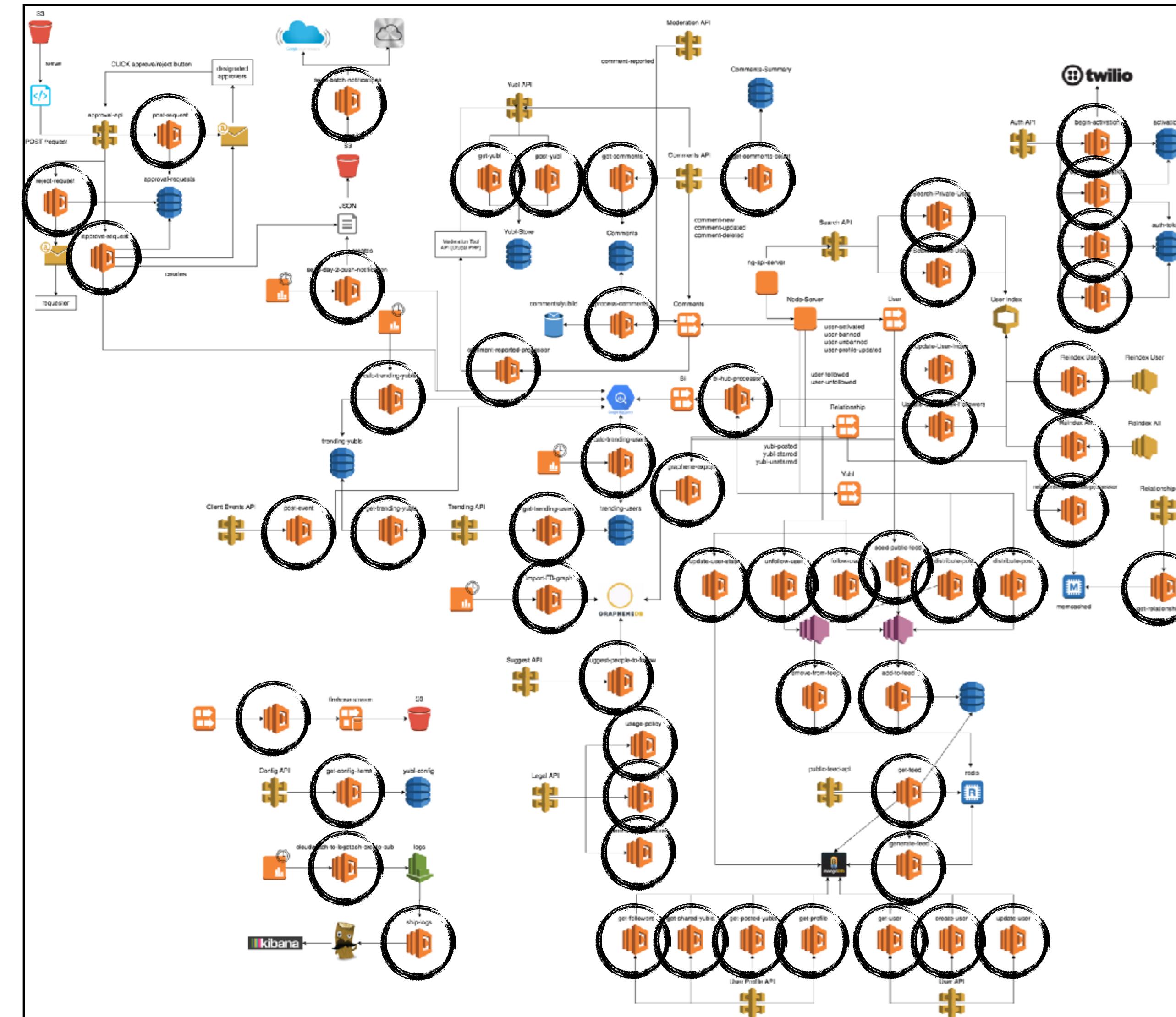
the **risk** of shipping broken software
has largely shifted to how your
Lambda functions **integrate** with
external services

serverless

observation #3

smaller units of deployment means
finer grained control of access, and
more things to secure

serverless

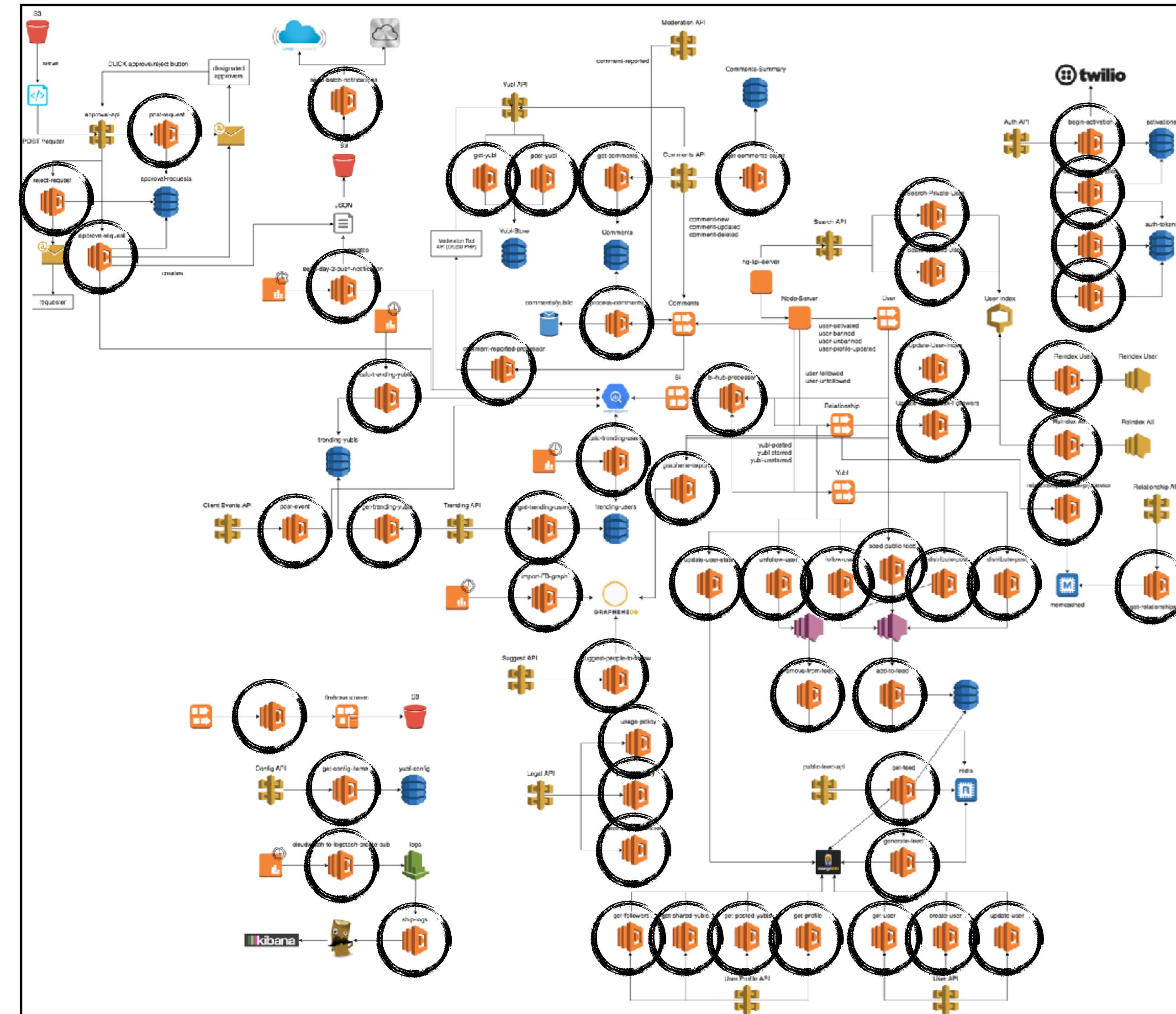


serverless

observation #4

smaller units of deployment also
means more application
configurations in general

serverless



serverless

conclusion #2

the **risk** of misconfiguration (both application & IAM) has exploded

serverless

conclusion #3

the **risk** profile for a serverless application is very different to that of a serverful application

test honeycomb

