#HASHITALKS: CANADA

Bringing software development practices to your infrastructure

@jennapederson

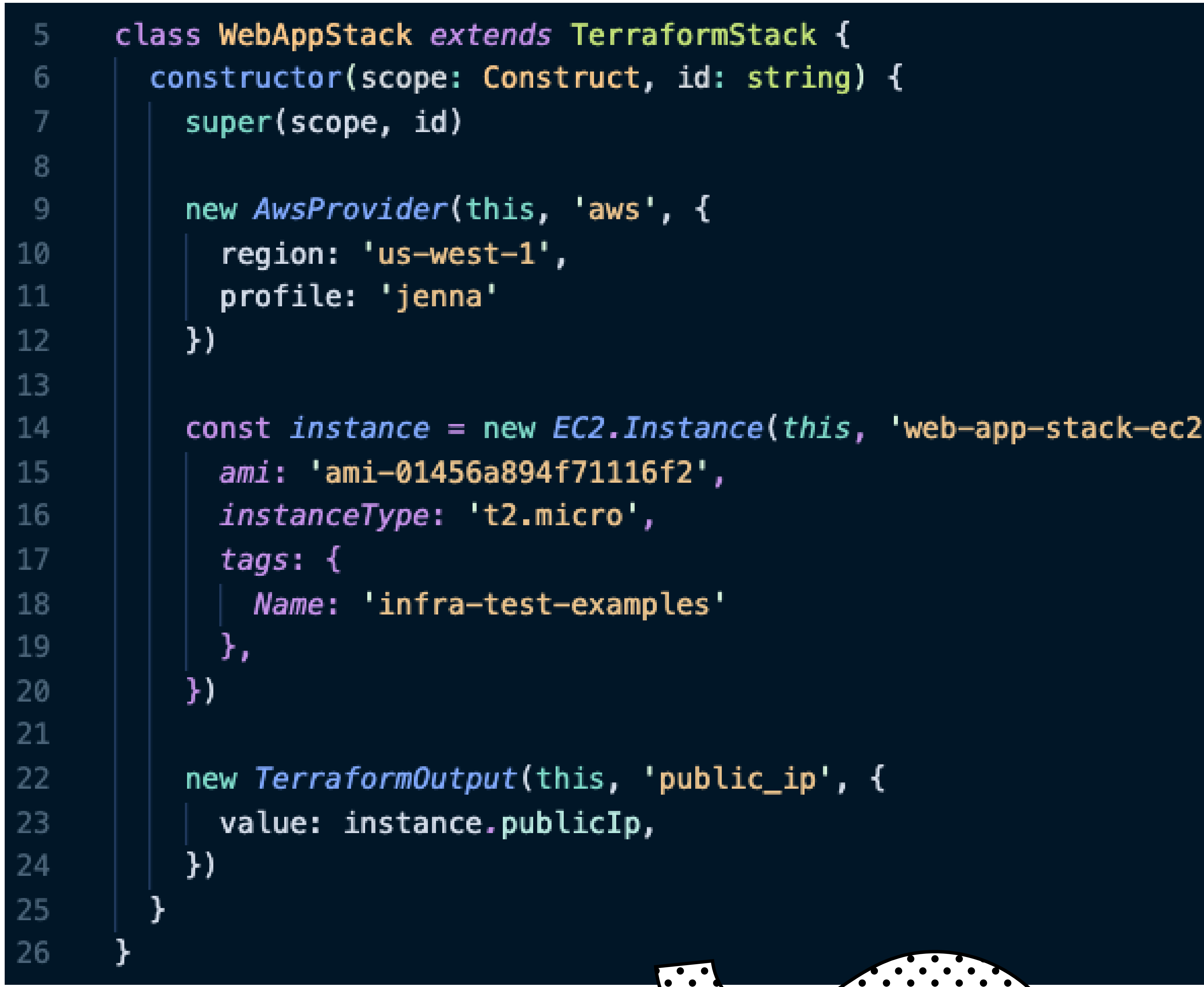
Fun Fact

I once had the phrase "automated test fanatic" on my business card.

The awesomeness of Infrastructure as Code

@jennapederson

Infrastructure as Code
IS Code



```
class WebAppStack extends TerraformStack {
  constructor(scope: Construct, id: string) {
    super(scope, id)

    new AwsProvider(this, 'aws', {
      region: 'us-west-1',
      profile: 'jenna'
    })

    const instance = new EC2.Instance(this, 'web-app-stack-ec2
      ami: 'ami-01456a894f71116f2',
      instanceType: 't2.micro',
      tags: {
        Name: 'infra-test-examples'
      },
    })

    new TerraformOutput(this, 'public_ip', {
      value: instance.publicIp,
    })
  }
}
```

@jennapederson

works on **my** machine

(or account or region)

Agenda

Different Types of Testing

Using the right type at the right time

Using Test Driven Development

Build what you need and only what you need

Testing Your Infrastructure Directly

Making sure it was created correctly and hasn't drifted

Using a CI/CD Pipeline

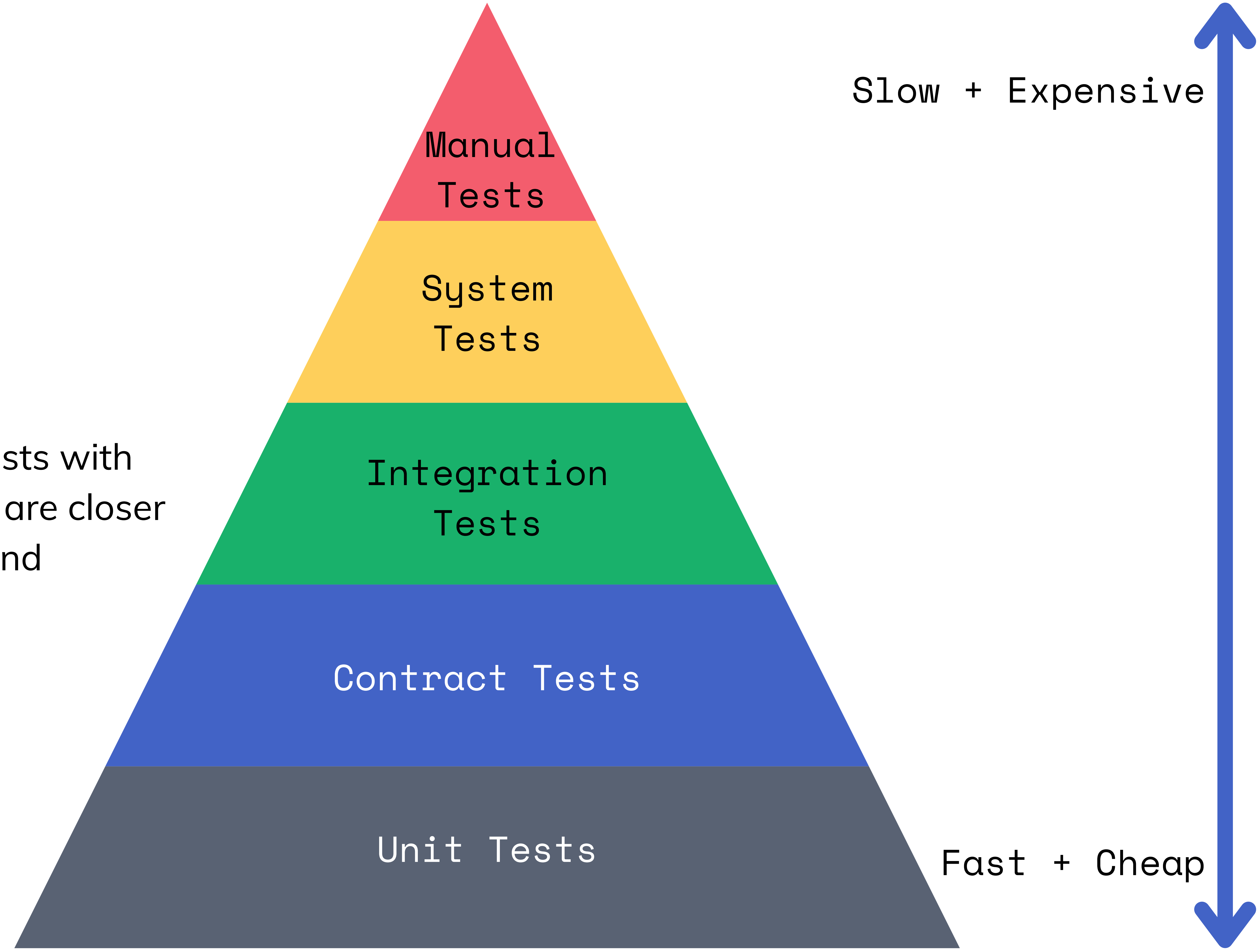
Run tests in the real world and isolate issues quicker

Why Test Infrastructure?

The cloud makes it easier and quicker to provision infrastructure, but there is complexity with that scale.

Failing Fast

Balance fast and cheap tests with more expensive tests that are closer to the real infrastructure and production environment.

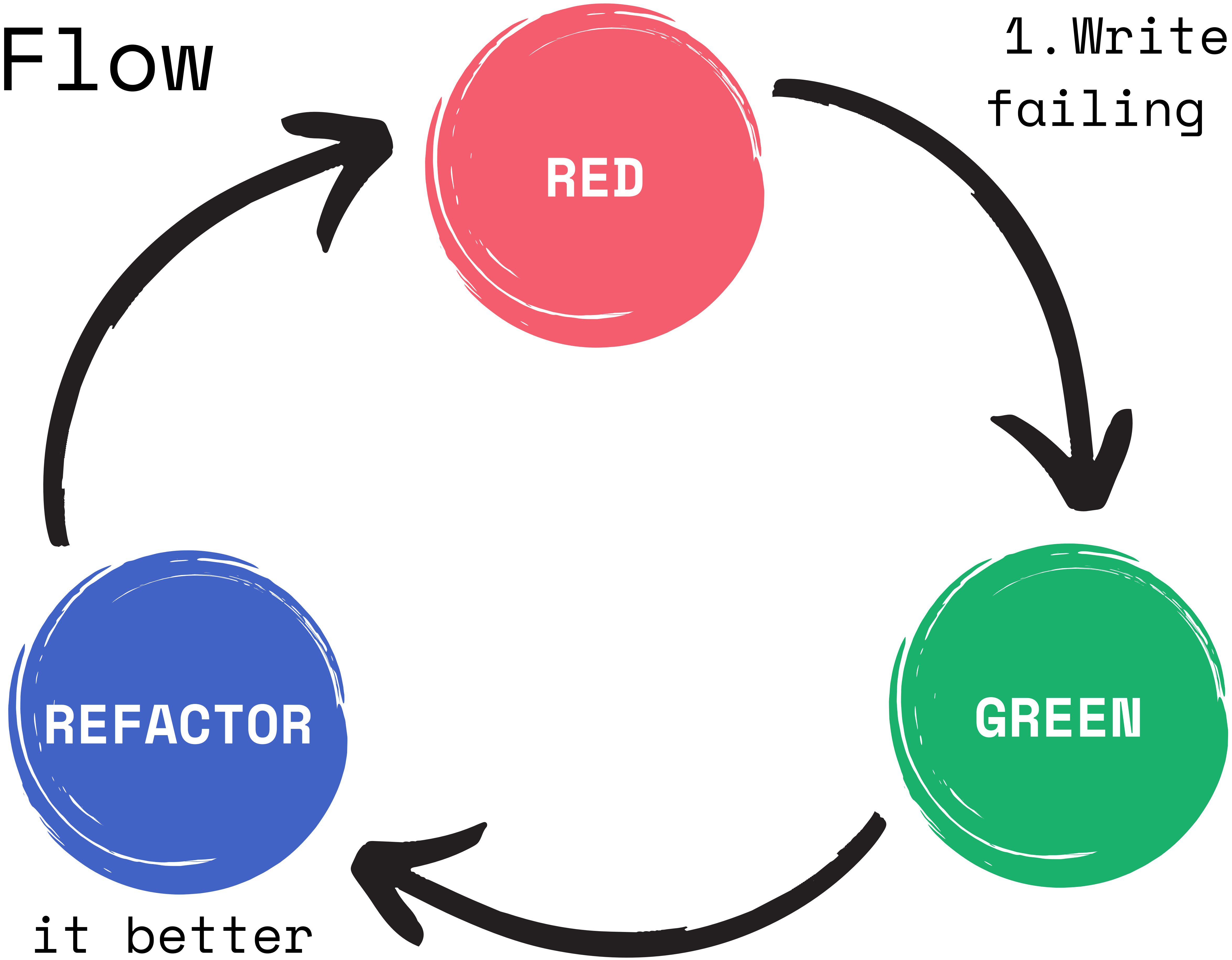


If you're TDDing your application code, why not do the same for your infrastructure code?

Benefits of TDD

- Reduced defect rates
- Improve the overall design
- Focused on requirements
- Focused on small chunks
- Serves as documentation
- Confidence

The Flow



RED

1.Write a failing test

GREEN

2. Write only enough code to make it pass

REFACTOR

3. Make it better

What is a unit test?

- Exercises a small part of your application, one unit, and verifies that it's correct.
- Get feedback early on to shorten the feedback loop between changes
- Serves as documentation
- Can be run in your CI/CD tool
- Isolated from other resources and external APIs

Unit Testing Infrastructure Code

Code. Not infrastructure.

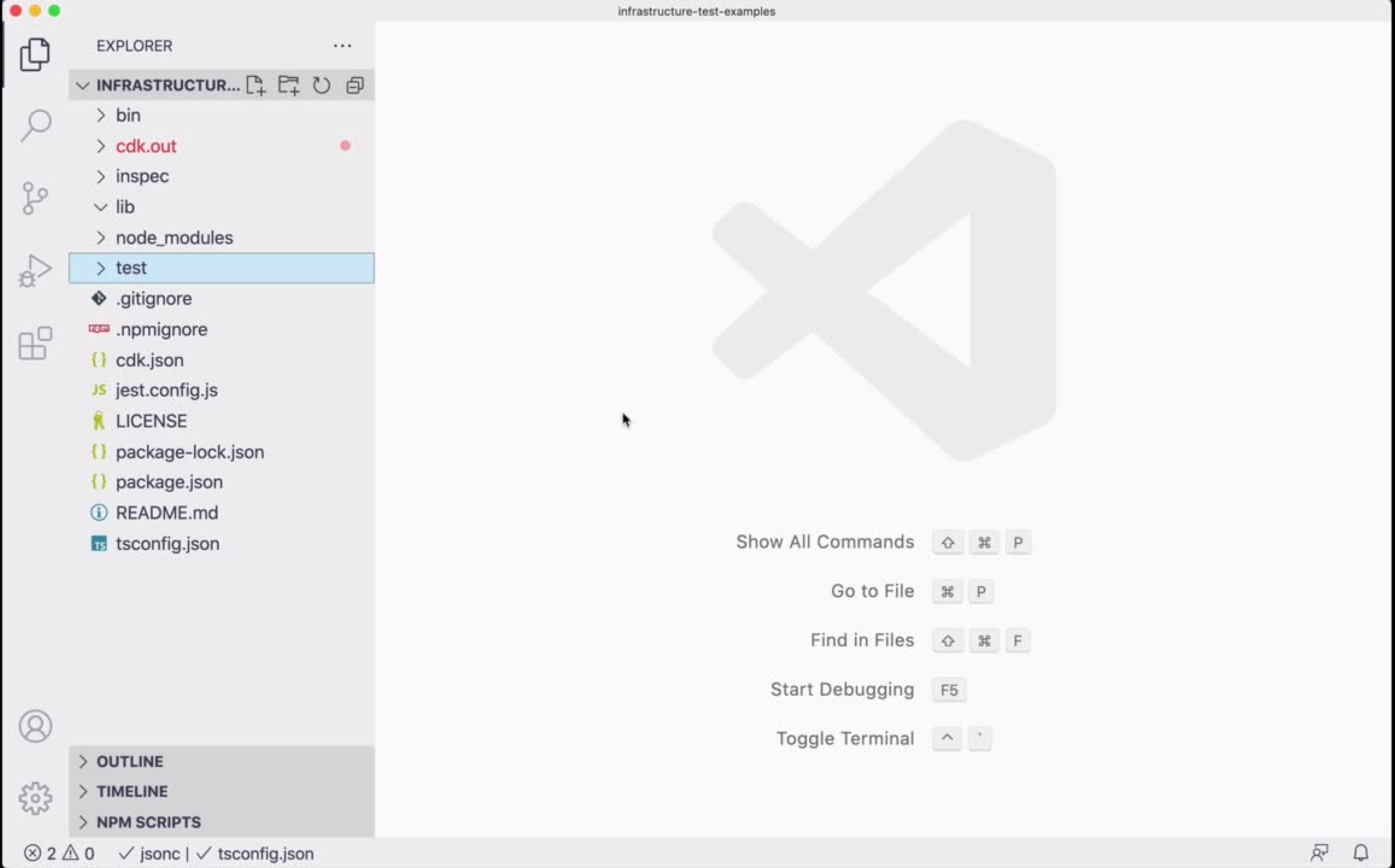
@jennapederson

A unit test checks:

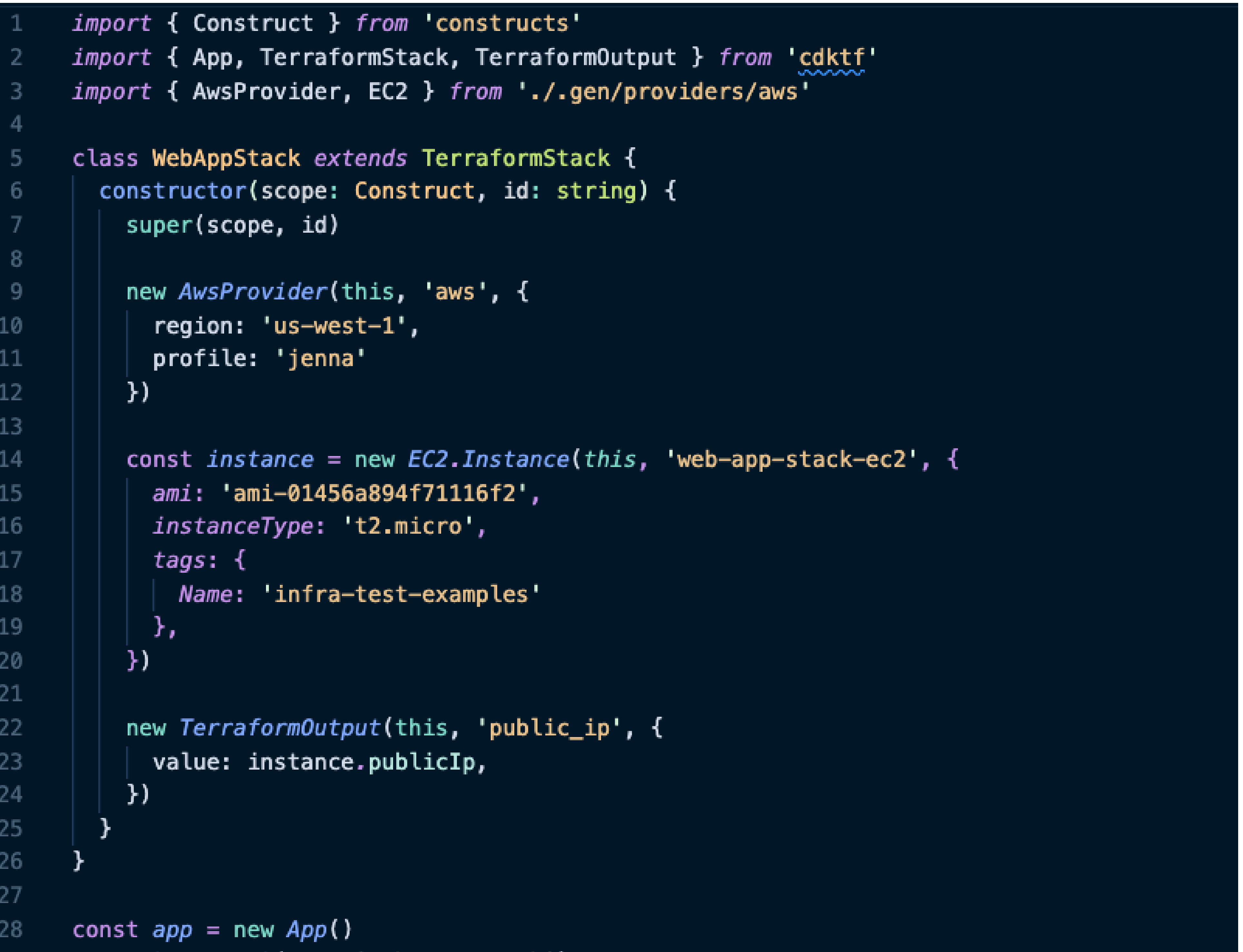
- If a resource will be created with the correct configuration
- The correct number of resources will be created
- Dependencies between resources are correct
- Interpolated values are correct

Demo

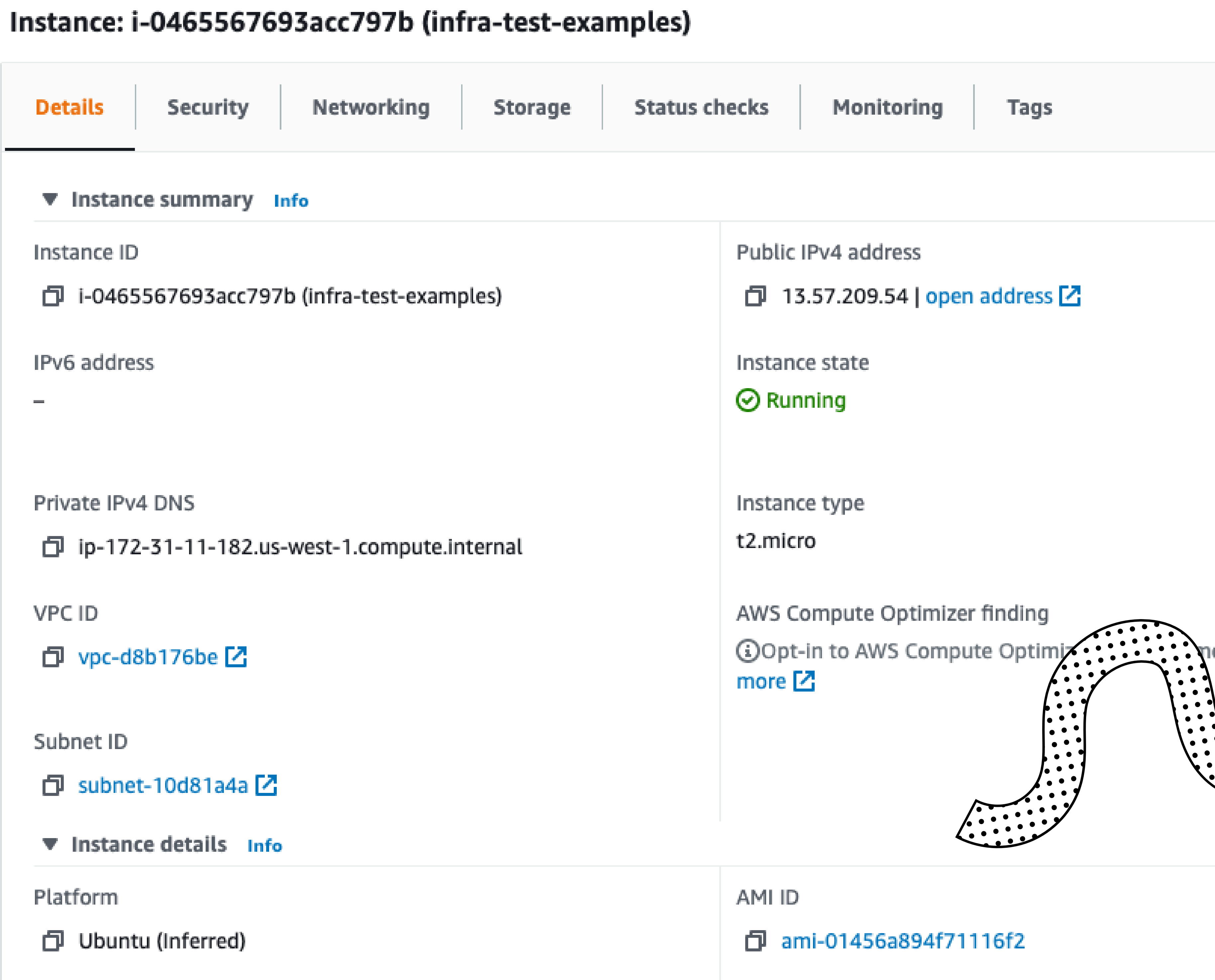
S3 + CDK + Jest


infrastructure-test-examples
EXPLORER
INFRASTRUCTUR...
bin
cdk.out
inspec
lib
node_modules
test
.gitignore
.npmignore
cdk.json
jest.config.js
LICENSE
package-lock.json
package.json
README.md
tsconfig.json
OUTLINE
TIMELINE
NPM SCRIPTS
Show All Commands ⇧ ⌘ P
Go to File ⌘ P
Find in Files ⇧ ⌘ F
Start Debugging F5
Toggle Terminal ^ `
⊗ 2 ⚠ 0 ✓ jsonc | ✓ tsconfig.json

How do we go from code to infrastructure?



```
import { Construct } from 'constructs'
import { App, TerraformStack, TerraformOutput } from 'cdktf'
import { AwsProvider, EC2 } from './.gen/providers/aws'

class WebAppStack extends TerraformStack {
  constructor(scope: Construct, id: string) {
    super(scope, id)

    new AwsProvider(this, 'aws', {
      region: 'us-west-1',
      profile: 'jenna'
    })

    const instance = new EC2.Instance(this, 'web-app-stack-ec2', {
      ami: 'ami-01456a894f71116f2',
      instanceType: 't2.micro',
      tags: {
        Name: 'infra-test-examples'
      },
    })

    new TerraformOutput(this, 'public_ip', {
      value: instance.publicIp,
    })
  }
}

const app = new App()
```



Instance: i-0465567693acc797b (infra-test-examples)

▼ **Instance summary** Info

Instance ID
i-0465567693acc797b (infra-test-examples)

Public IPv4 address
13.57.209.54 | open address

IPv6 address
–

Instance state
Running

Private IPv4 DNS
ip-172-31-11-182.us-west-1.compute.internal

Instance type
t2.micro

VPC ID
vpc-d8b176be

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimi
more

Subnet ID
subnet-10d81a4a

▼ **Instance details** Info

Platform
Ubuntu (Inferred)

AMI ID
ami-01456a894f71116f2

What is an Integration Test?

Tests the interactions across different units or modules, or in the case of infrastructure testing, across cloud resources.

Verifies your provisioned cloud resources are created and configured as you expect them to be.

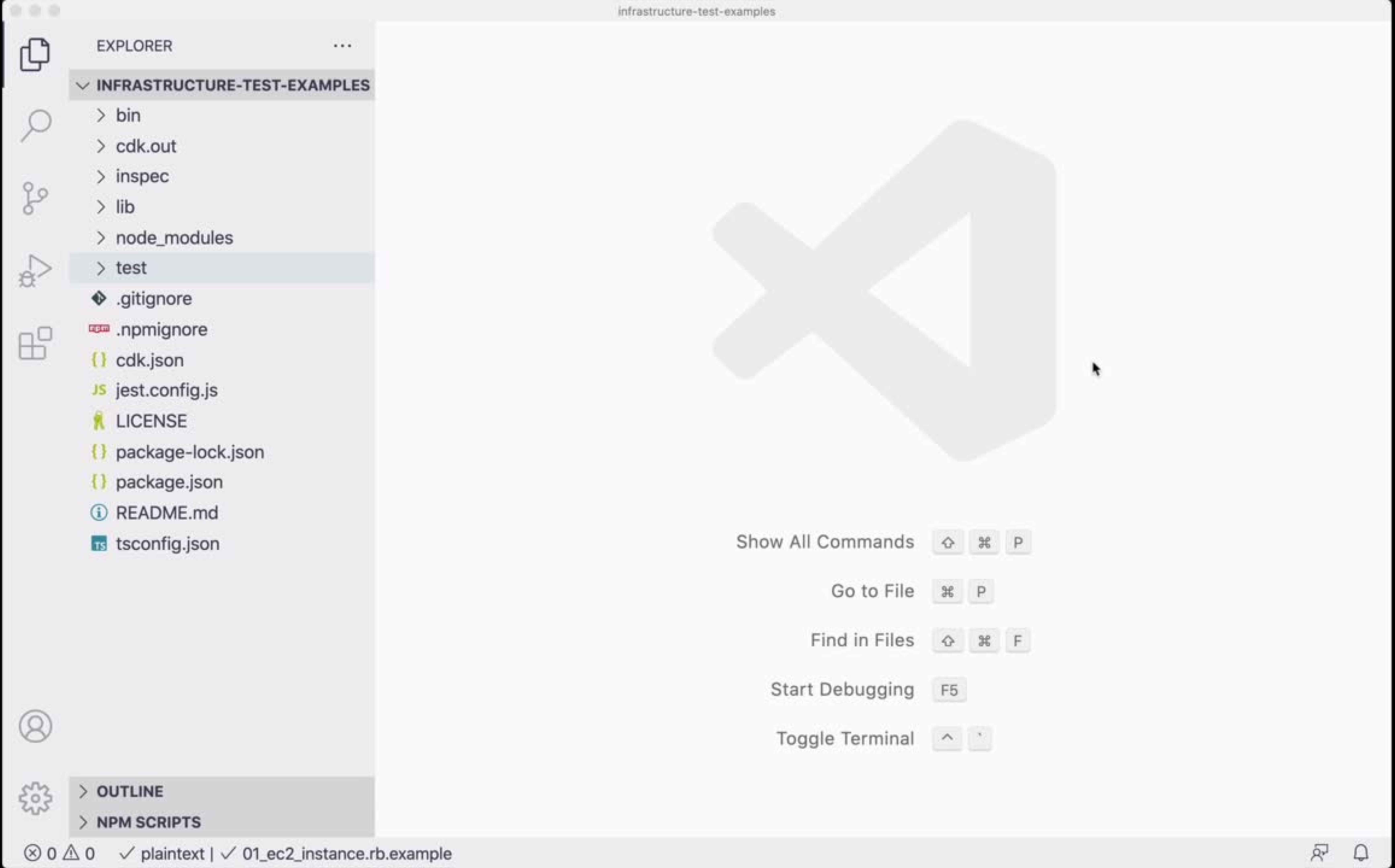
Gives you confidence in infrastructure at scale and at velocity.

Chef InSpec

- Open-source framework to test and audit cloud resources IN the cloud
- Tests are written with a DSL
- Can be used across teams
- Test resources that are managed manually or with code
- Ensures requirements are met at every stage of the SDLC

Demo

EC2 + RDS + CDK + InSpec

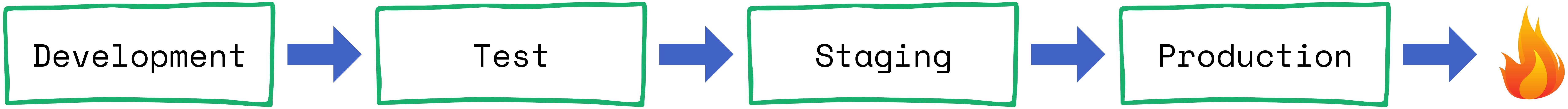

infrastructure-test-examples
EXPLORER
INFRASTRUCTURE-TEST-EXAMPLES
bin
cdk.out
inspec
lib
node_modules
test
.gitignore
.npmignore
cdk.json
jest.config.js
LICENSE
package-lock.json
package.json
README.md
tsconfig.json
OUTLINE
NPM SCRIPTS
Show All Commands ⇧ ⌘ P
Go to File ⌘ P
Find in Files ⇧ ⌘ F
Start Debugging F5
Toggle Terminal ^ `
⊗ 0 ⚠ 0
✓ plaintext | ✓ 01_ec2_instance.rb.example

Detecting Drift

Use InSpec to compare the desired state
with the actual state of your
cloud resources.

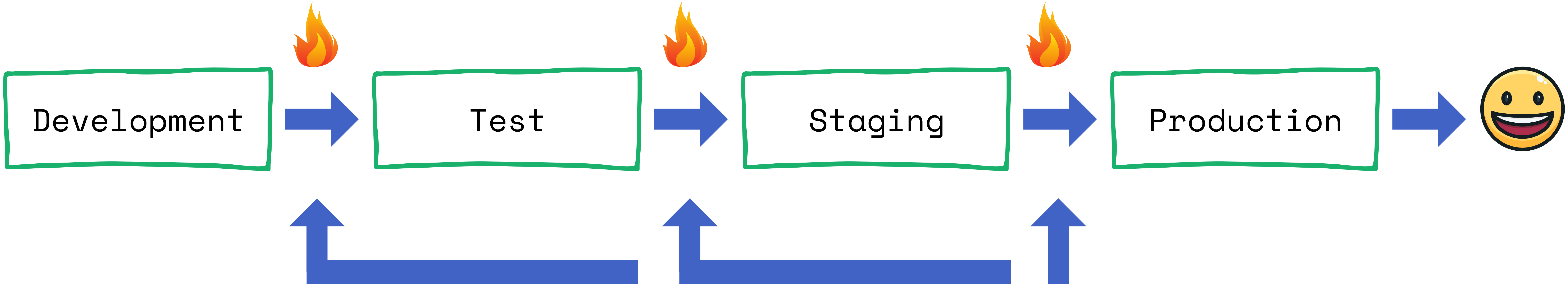
Can be used against any resources,
regardless of how they are managed.

Without CI/CD



Development → Test → Staging → Production → 🔥

With CI/CD



@jennapederson

Wrapping Up

Infrastructure code is like any other code, treat it as such.

Testing is never done, even once you reach production.

It's cheaper to detect broken code early.

Thank you!

- @jennapederson
- /in/jennapederson
- jennapederson
- https://jenna.link/hq7