#JAWSPANKRATION

Bringing software development practices to your infrastructure

@jennapederson

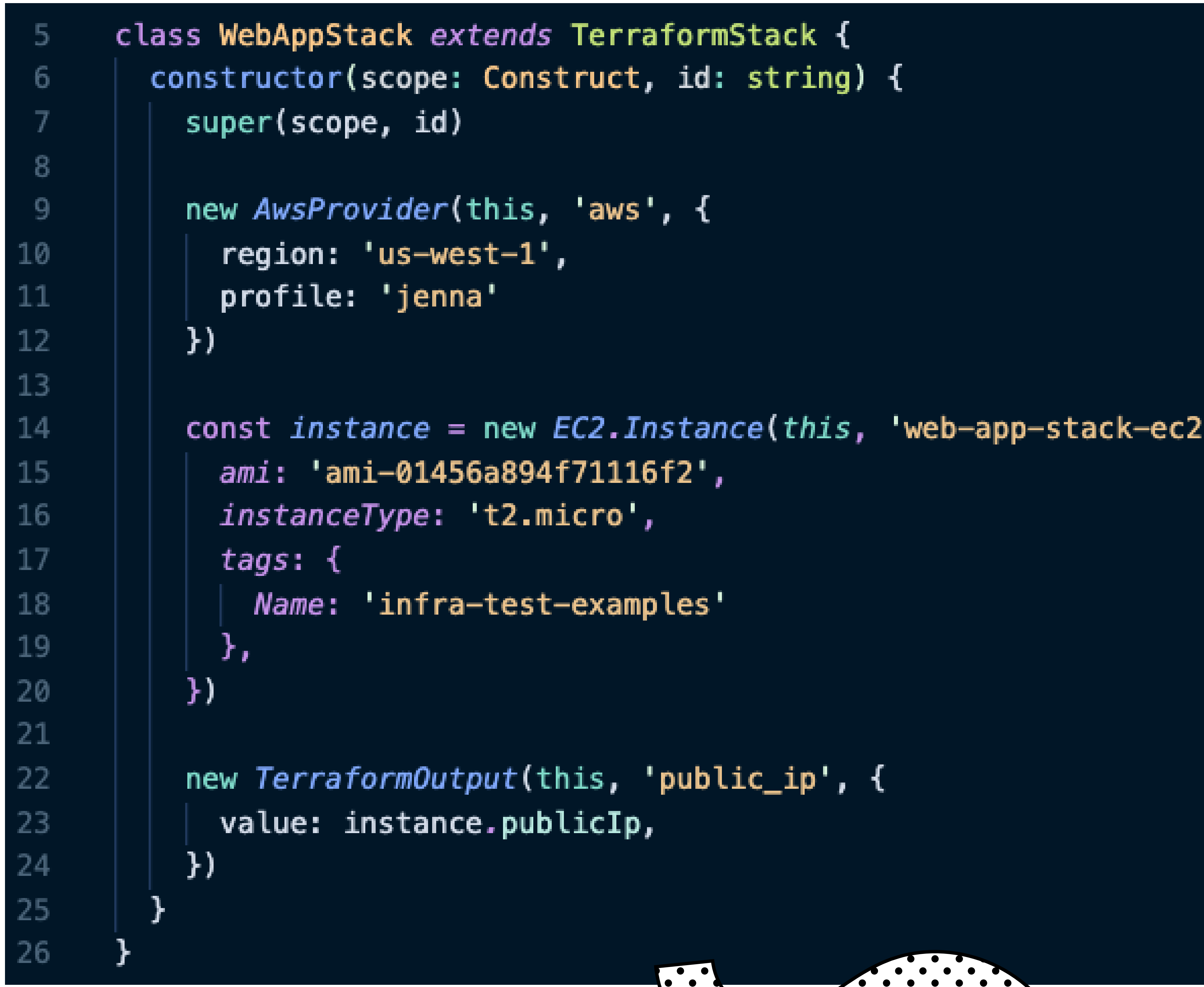
What is Infrastructure as Code?

Code that lets you automate deployments of your infrastructure to facilitate both scaling and quicker, repeatable deployments.

@jennapederson

Infrastructure as Code IS Code

- Version control it
- Code review it
- Test it
- Deploy it to each environment with CI/CD



```
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

(or account or region)

What happens when infrastructure code breaks?

The blast radius is much wider. More resources, regions, accounts, customers, and dollars are impacted.

@jennapederson

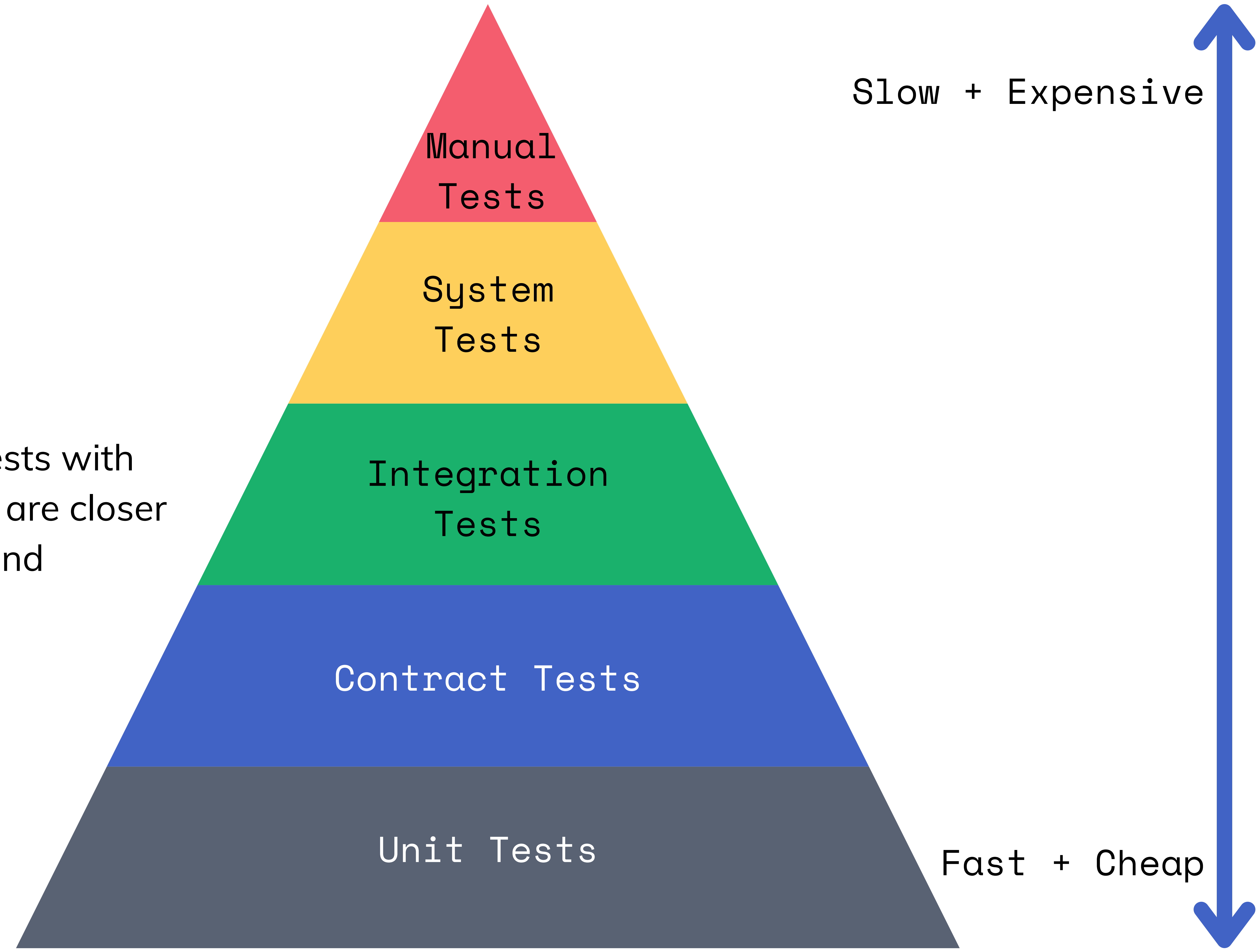
Why Test Infrastructure?

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

@jennapederson

Failing Fast

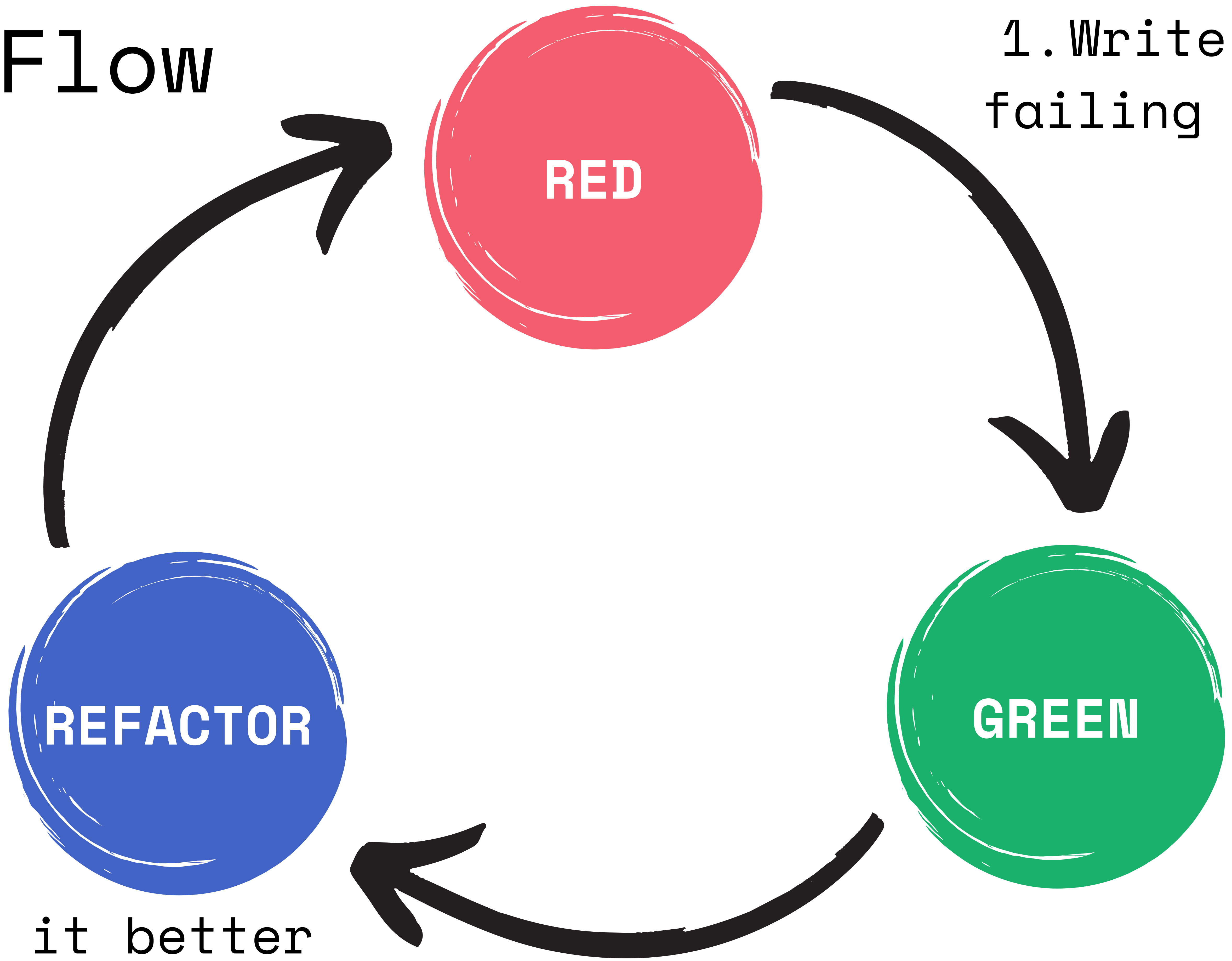
Balance fast and cheap tests with more expensive tests that are closer to the real infrastructure and production environment.



Benefits of TDD

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

The Flow



1. Write a failing test

2. Write only enough code to make it pass

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

@jennapederson

Unit Testing Infrastructure Code

Apply the same process to your infrastructure code.

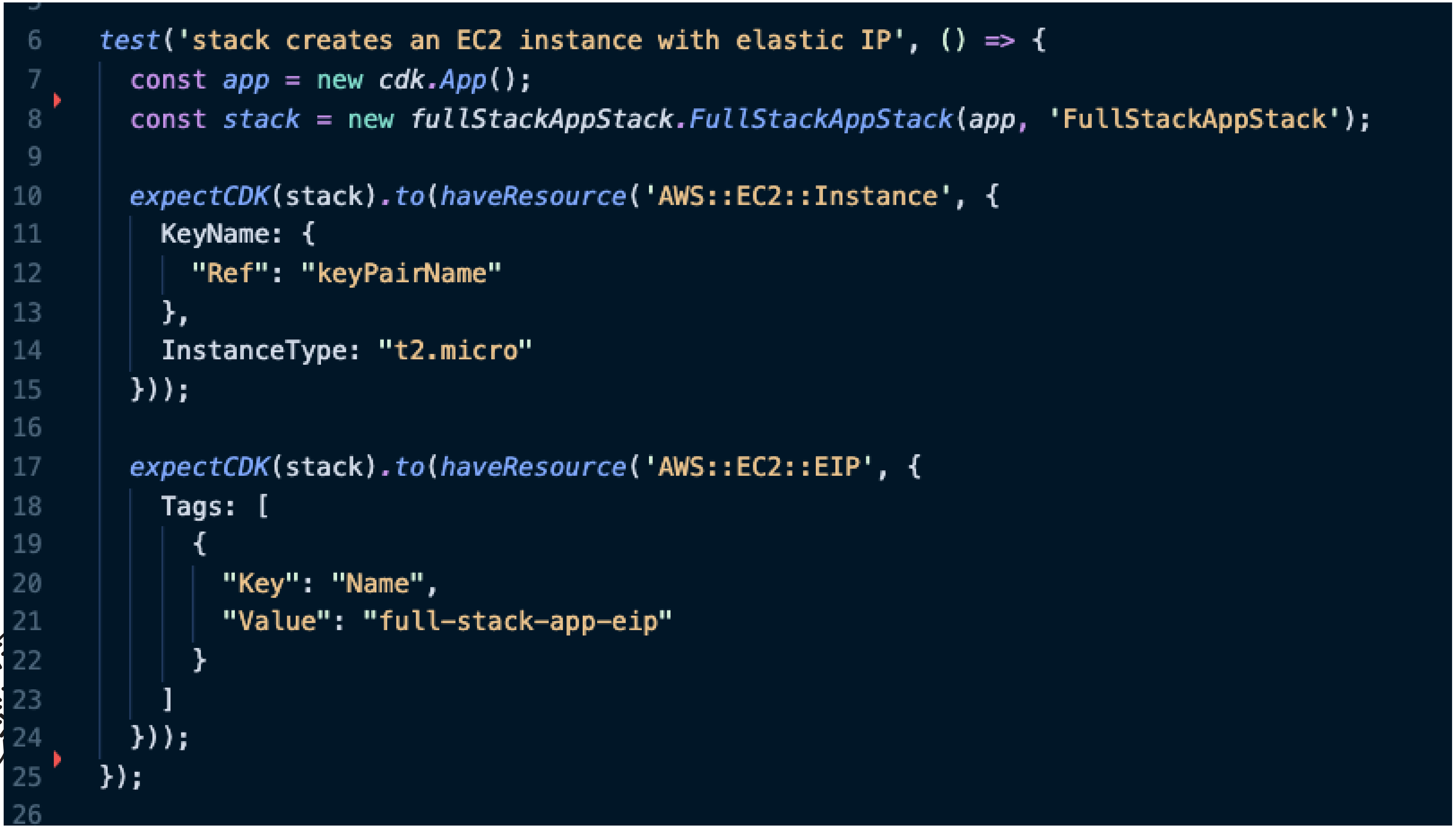
@jennapederson

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

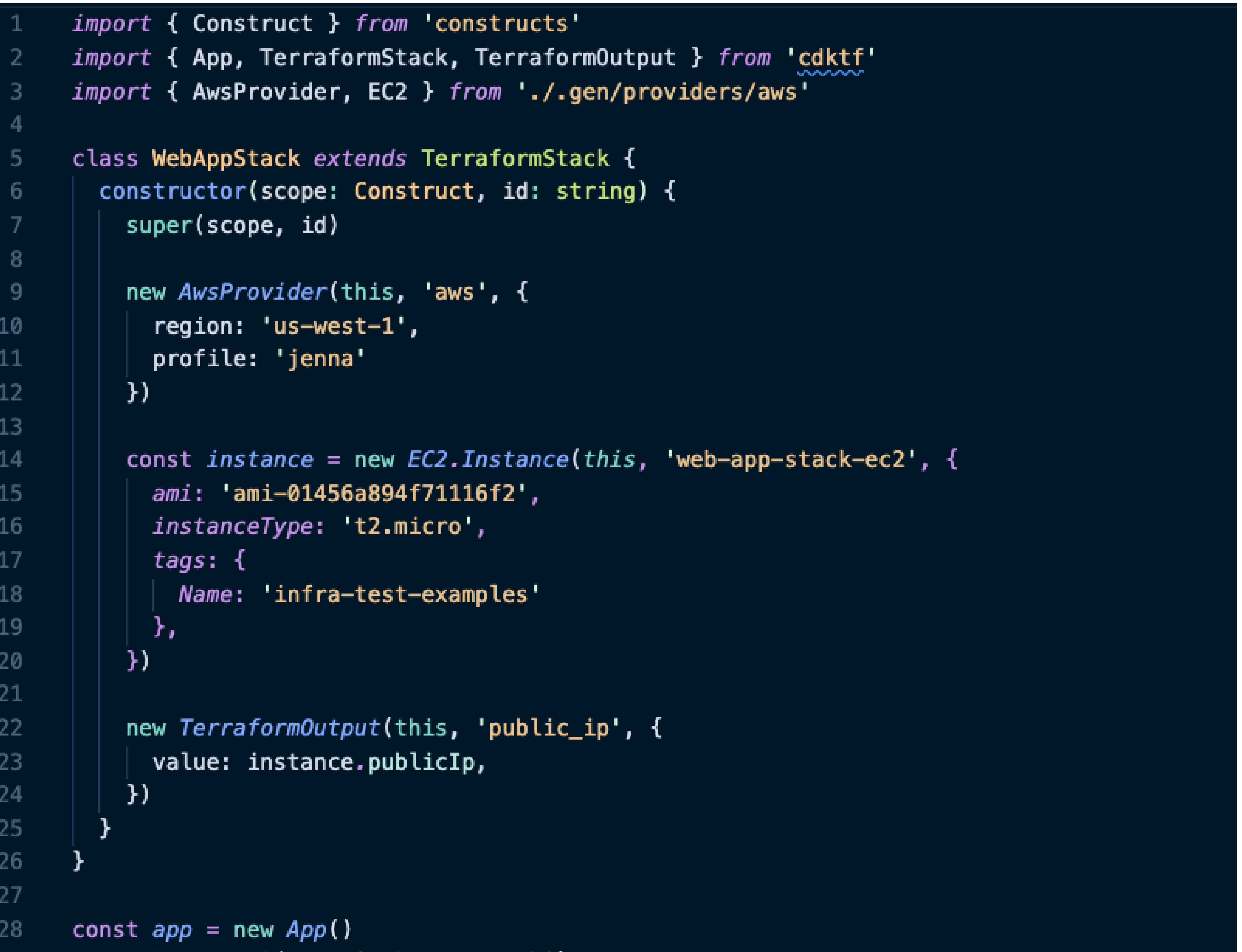
@jennapederson

Example Unit Test

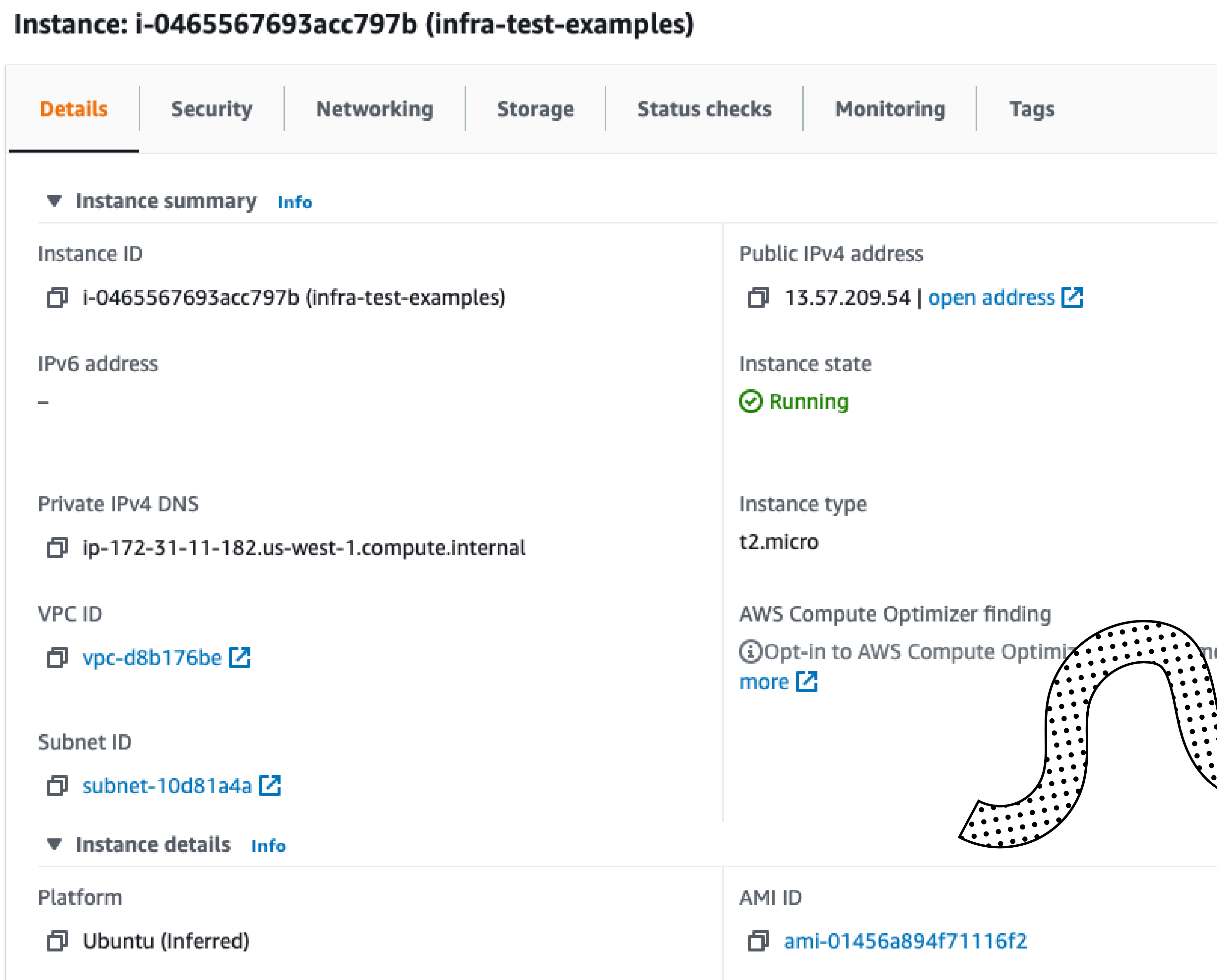


```
test('stack creates an EC2 instance with elastic IP', () => {
  const app = new cdk.App();
  const stack = new fullStackAppStack.FullStackAppStack(app, 'FullStackAppStack');

  expectCDK(stack).to(haveResource('AWS::EC2::Instance', {
    KeyName: {
      "Ref": "keyPairName"
    },
    InstanceType: "t2.micro"
  }));

  expectCDK(stack).to(haveResource('AWS::EC2::EIP', {
    Tags: [
      {
        "Key": "Name",
        "Value": "full-stack-app-eip"
      }
    ]
  }));
});
```

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	Public IPv4 address
i-0465567693acc797b (infra-test-examples)	13.57.209.54 | open address
IPv6 address	Instance state
–	Running
Private IPv4 DNS	Instance type
ip-172-31-11-182.us-west-1.compute.internal	t2.micro
VPC ID	AWS Compute Optimizer finding
vpc-d8b176be	Opt-in to AWS Compute Optimiz more
Subnet ID	
subnet-10d81a4a	

▼ **Instance details** Info

Platform	AMI ID
Ubuntu (Inferred)	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.

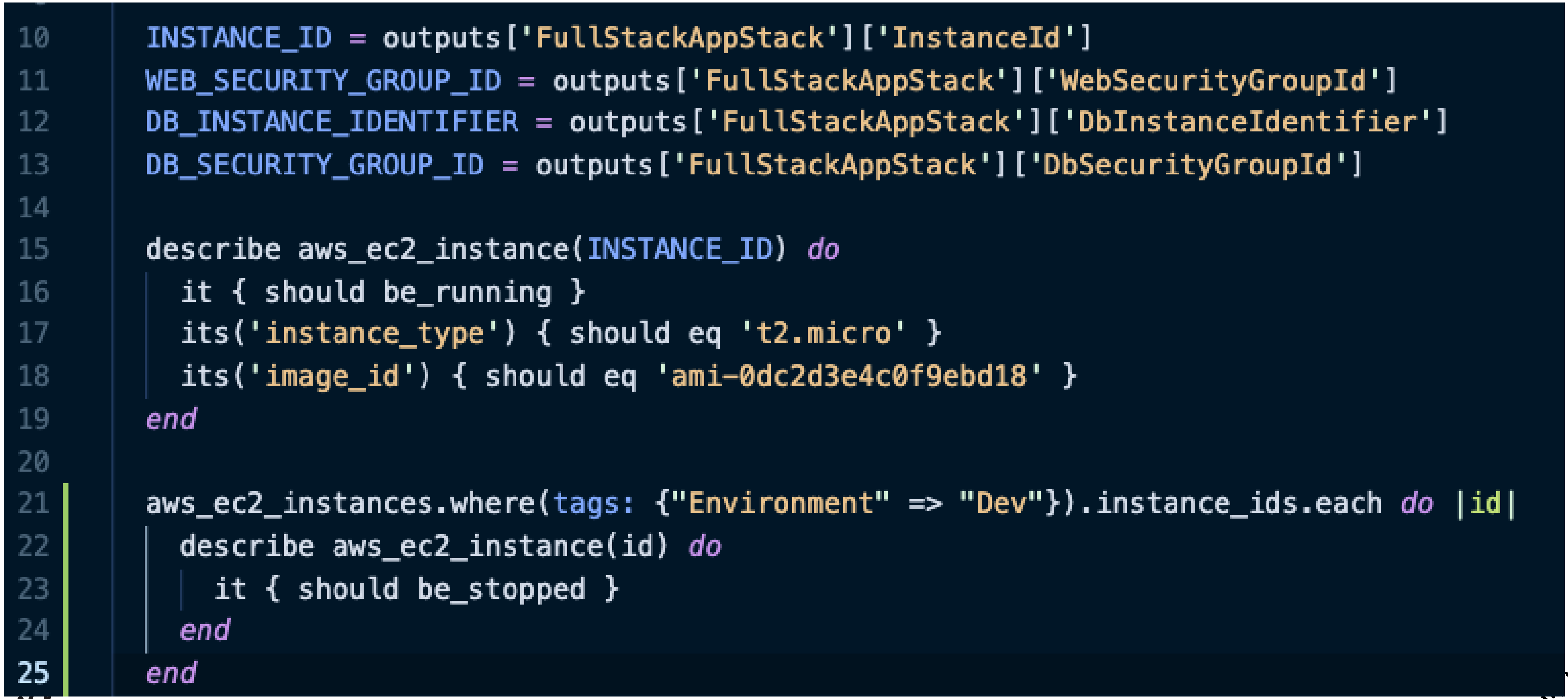
@jennapederson

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

@jennapederson

Example Integration Test



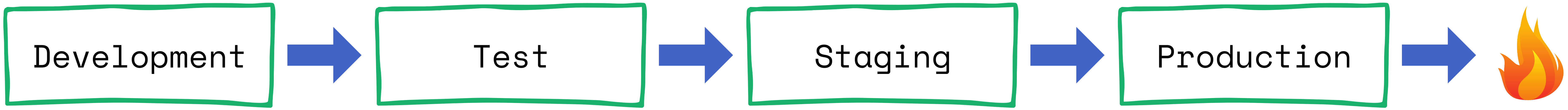
```ruby
INSTANCE_ID = outputs['FullStackAppStack']['InstanceId']
WEB_SECURITY_GROUP_ID = outputs['FullStackAppStack']['WebSecurityGroupId']
DB_INSTANCE_IDENTIFIER = outputs['FullStackAppStack']['DbInstanceIdentifier']
DB_SECURITY_GROUP_ID = outputs['FullStackAppStack']['DbSecurityGroupId']

describe aws_ec2_instance(INSTANCE_ID) do
  it { should be_running }
  its('instance_type') { should eq 't2.micro' }
  its('image_id') { should eq 'ami-0dc2d3e4c0f9ebd18' }
end

aws_ec2_instances.where(tags: {"Environment" => "Dev"}).instance_ids.each do |id|
  describe aws_ec2_instance(id) do
    it { should be_stopped }
  end
end
```

Detecting Drift

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

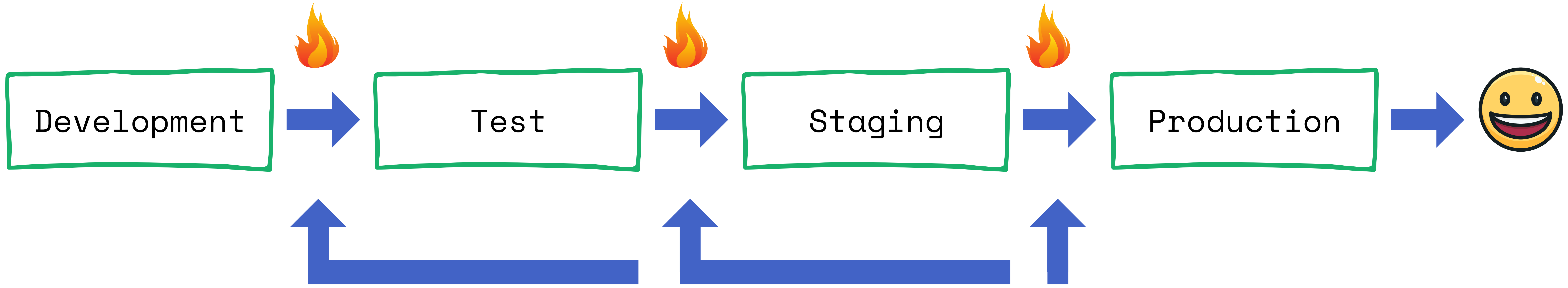
Can be used against any resources,
regardless of how they are managed.

Without CI/CD



@jennapederson

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.

@jennapederson

Thank you!

@jennapederson

/in/jennapederson

jennapederson

https://jenna.link/hq7