CDK driven Testing

Test Driven Cloud Development or CDK driven Testing- gglawe@tecracer.de

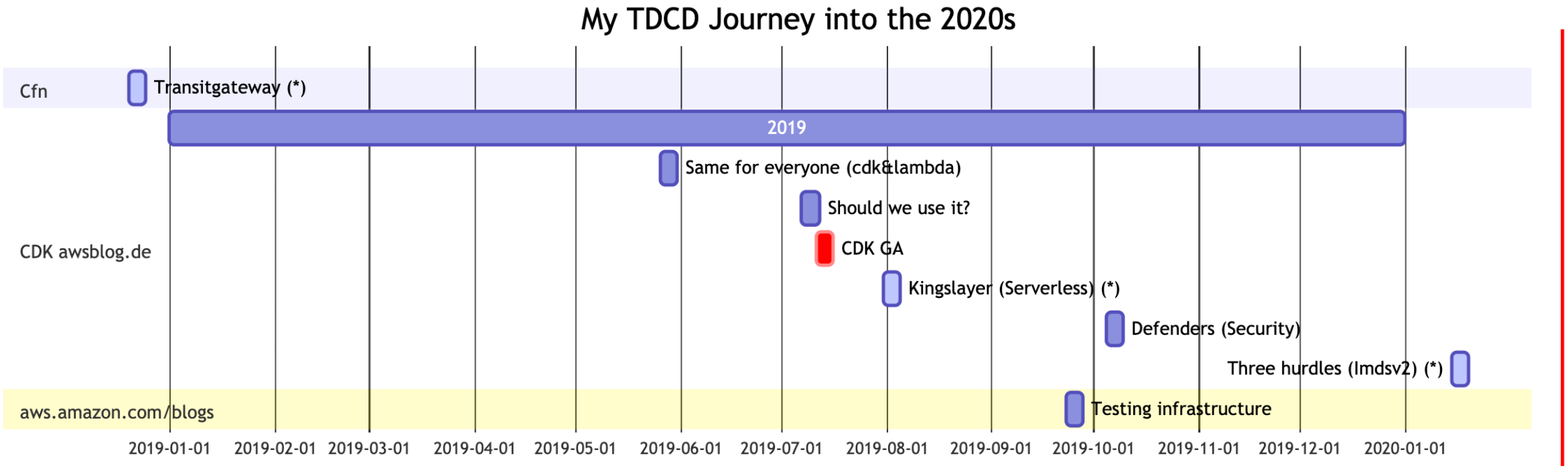
tecRACER
Cloud Enabling Your Business

amazon web services | Partner Network
MANAGED SERVICE PARTNER

amazon web services | Partner Network
PREMIER CONSULTING PARTNER
CHANNEL RESELLER PARTNER

amazon web services | Partner Network
AUTHORIZED TRAINING PARTNER

TDCD- Testing the cloud from the 2010s to the 2020s



TDCD - Testing the cloud from the 2010s to the 2020s

Views to the environment

First Part of the Journey: Create Test Environments

Second Part of the Journey: Test the Cloud

Third Part: Inspec(t) the Cloud

A Pyramid View

Views to the environment

Testing Phases

1. Setup environment
2. Test functionality
3. Teardown environment

Create Test Environments

Use Cases

- Transit Gateway 2010s and 2020s
- Instances: My Testing instance
- IMDBv2

Comparing Aspects :

1. Source Language
2. Management
3. Testing

Why showing UseCases for Testing if its about CDK?

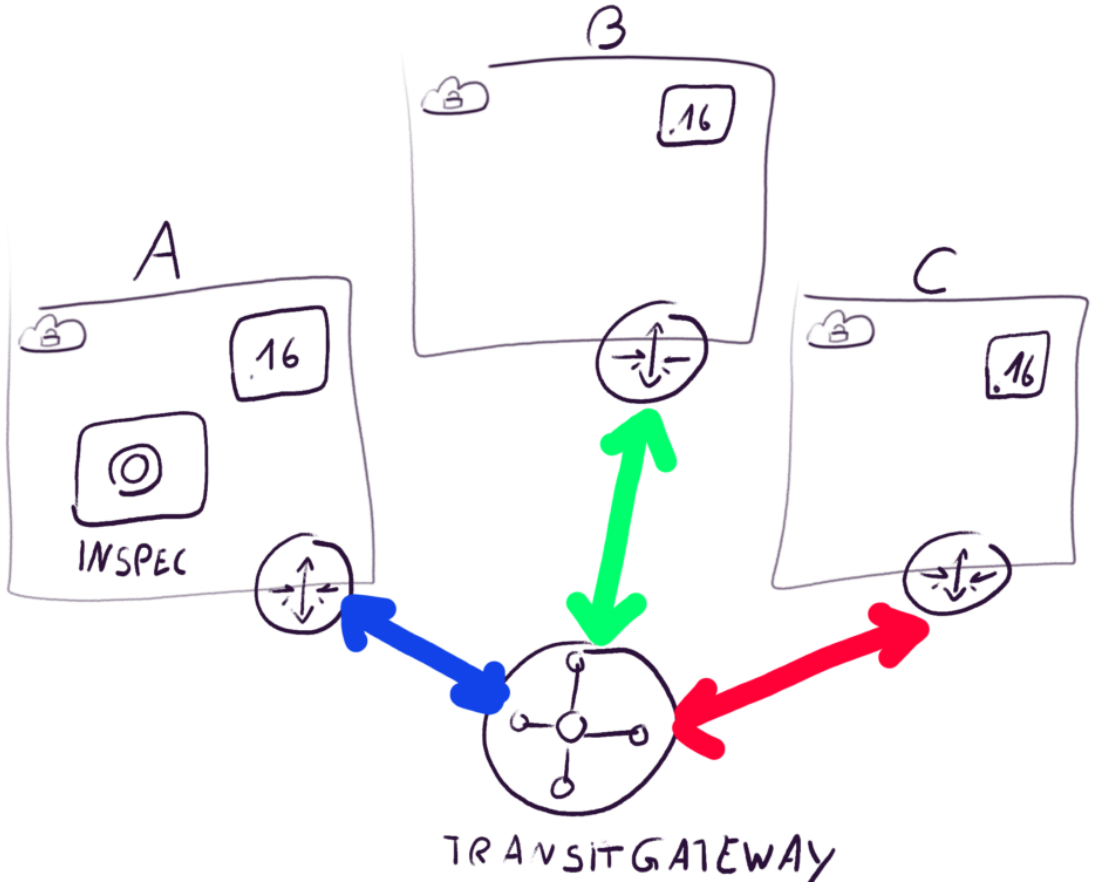
Showing how CDK can be a good tool for test automation

Showing the wiring of tools and CDK

With an new generated infrastructure environment you minimize environmental test pollution

UseCase 1 - The Transit Gateway Test

- The (then) new Transit Gateway connects n VPCs instead of only two like VPC Peering
- As a consultant I have to know how it works
- I have seen people fail presenting with doing it manually, so automate it



Infrastructure: Testing Transit Gateway - 2010s

Source Language: CloudFormation

Management:

- Clouds-aws
- Makefile

Testing: taskCat

See: https://github.com/tecracer/demo-transit-gateway

CloudFormation - 2010s - D=DRY = Repeat yourself

Pure CloudFormation: Ugly

- Count the Lines!

Management - 2010s

Good Old Makefile

- Makefile works, but
  - no good cross OS (i mean windows implementation)
  - no standard way to describe tasks

"Simple" Help Hack = awk magic

```
help:
	@grep -E '^[a-zA-Z_-]+:.*?## .*$$' $(MAKEFILE_LIST) | sort | awk 'BEGIN {FS = ":.*?## "}; {printf "\033[36m%-30s\033[0m %s\n", $$1, $$2}'
```

```
deploy: ## Deploy stack
```

Management - 2010s

"Clouds-aws" as Cloudformation CLI

- Simple stack management tool
- Use directory structure as Template names
- Get deployed Templates
- Manage parameters

```
└── stacks
    ├── demo-vpc-a
    ├── demo-vpc-b
    └── demo-vpc-c
```

Testing 2010 - taskcat

- Test tool for deploying stacks
  - in several regions
  - with parameter sets
  - deploy and destroy automatically

```
.
├── LICENSE
├── Makefile
├── ci
│   ├── debug_a.json
│   ├── debug_b.json
│   ├── debug_c.json
│   └── taskcat.yml
```

Testing 2010 - taskcat run

```
(python3) silberkopf@Gernots-MacBook-Pro:demo-transit-gateway(AWS:ggtrcadmin)>make lint-cfn
cp stacks/demo-vpc-a/template.yaml templates/testa.yaml
cp stacks/demo-vpc-b/template.yaml templates/testb.yaml
cp stacks/demo-vpc-c/template.yaml templates/testc.yaml
taskcat  -c ci/taskcat.yml -l
 _             _              _
| |_ __ _ ___| | _____ __ _| |_
| __/ _` / __| |/ / __/ _` | __|
| || (_| \__ \   < (_| (_| | |_
 \__\__,_|___/_|\_\___\__,_|\__|


version 0.8.19


[taskcat] :AWS AccountNumber:    [            ]
[taskcat] :Authenticated via:    [environment]


[taskcat] :Reading Config form: ci/taskcat.yml
[taskcat] |Queing test => scenario-a
[taskcat] |Queing test => scenario-b
[taskcat] |Queing test => scenario-c
[INFO   ] :Lint passed for test scenario-a on template demo-transit-gateway/templates/testa.yaml:
[INFO   ] :Lint passed for test scenario-b on template demo-transit-gateway/templates/testa.yaml:
[INFO   ] :Lint passed for test scenario-c on template demo-transit-gateway/templates/testa.yaml:
[INFO   ] :Linting completed
```

Testing 2010 - taskcat report

GitHub Repo: https://github.com/aws-quickstart/taskcat
Documentation: http://taskcat.io
Tested on: Tuesday - Dec,11,2018 @ 22:05:38

taskcat

Test Name	Tested Region	Stack Name	Tested Results	Test Logs
scenario-a	eu-central-1	tCaT-tag-scenario-a-89b456d8	CREATE_COMPLETE	View Logs
scenario-a	eu-west-1	tCaT-tag-scenario-a-89b456d8	CREATE_COMPLETE	View Logs
scenario-b	eu-central-1	tCaT-tag-scenario-b-89b456d8	CREATE_COMPLETE	View Logs
scenario-b	eu-west-1	tCaT-tag-scenario-b-89b456d8	CREATE_COMPLETE	View Logs
scenario-c	eu-central-1	tCaT-tag-scenario-c-89b456d8	CREATE_COMPLETE	View Logs
scenario-c	eu-west-1	tCaT-tag-scenario-c-89b456d8	CREATE_COMPLETE	View Logs

Generated by taskcat 0.8.19

Infrastructure: Testing Transit Gateway - 2020s

Source Language:

- CDK
- polyglott

Management:

- CDK: All commands for deploy, destroy, diff included
- task: a better Makefile

Testing:

- CDK

Testing Transit Gateway CDK

DRY Routing Loop

```
for (var i = 0, len = vpnVPC.privateSubnets.length;
  i < len;
  i++) {
  new CfnRoute(this, "tgwVpc"+i,
  {
    routeTableId: vpnVPC.privateSubnets[i]
      .routeTable.routeTableId,
    destinationCidrBlock: net2Cidr,
    transitGatewayId: buildEnvTransitGateway.ref,
  }).addDependsOn(vpnVpcAttachment);      }
```

- Advantage to cfn: Do not repeat yourself, instead loop equals parts

Management - Taskfile

```yaml
# https://taskfile.dev
tasks:
  deploy:
    desc: deploy cdk "SingleInstanceStack"
    deps: [build-cdk]
    dir: single-instance
    cmds:
      - cdk deploy SingleInstanceStack
    silent: true
```

- Runs on "all" OS because of golang
- A better Makefile: Just works

CDK

Management

- Development and Management in one tool
- Supports Change Sets with `diff`
- List status of Stacks missing

Testing

- Deployment to different regions possible
- Region dependencies aware e.g. AMI images
- You can still use taskcat for reporting

Testing with CDK - Simple OK/NOK reporting

Report

```
Testing ec2-autostop
Do 6 Feb 2020 17:41:08 CET
✅ ec2-autostop OK
```

Script from tecracer cdk-templates

```
task edge-of-tomorrow >../log/$i.log 2>../log/$i.err
    if [ $? -eq 0 ]
    then
        echo ✅ $i OK
    else
        echo ❌ $i NOK
    fi
```

Edge of tomorrow is live, die, repeat

- Instead of using tascat, you can let cdk handle deploy and destroy
- The taskfile unifies the access, so that `task deploy` is alwas deploy, decoupled from the tool
- Let it run with each cdk update to detect hidden breaking changes

```yaml
edge-of-tomorrow:
  desc: deploy destroy repeat
  cmds:
    - task: update
    - task: deploy
    - task: destroy
```

UseCase 2 : My Testing Instance

Security: myip

- automatically deploy least privileges Security Group

Cost Efficiency: Stop Rule

- automatically deploy Cloudwatch rule to stop instance at the end of the day

Automating LP* Security groups

- Get external IP address

```
const baseUrl = 'http://checkip.amazonaws.com/';
...
var result = await request.get(options);
...
return clientIp;
```

- Use it in Security Groups

```
clientIp.then((ip) => {
    sg.addIngressRule(Peer.ipv4(ip), Port.tcp(22), "Ssh Client incoming")
```

Cost Efficiency: Stop Rule

Install

```
npm i cdk-instance-stop-rule
```

EC2 Target for CloudWatch events

```
new InstanceStopRule(this, "myTest Instance",{
  machineImage: linuxImage,
      instanceType: instanceType,
      instanceName: "testServer",
      vpc: testVPC,
      stopHour: 17,
  });
```

- See cdk-construct github/tecracer `cdk-instanceStopRule`

Use Case 3: Instance Metadata Service - V2 1of2

What is IMDS?

- Service to give apps on an ec2 the credentials from the Instance-Profile/Role

Security Problems with IMDSv1

- See technical background for Capital One
- V2 - More secure way to get EC2 credentials

What is instance connect?

- Problem: distributing ssh keys for individual users or (dont do it) shared users
- Solution: instance connect creates a one time ssh key for the user authenticated by IAM
- In Terminal: `mssh $instanceid` is fast and secure

But do these services work together?

> It turned out - not yet, so testing was needed and successfull

Goal: Fast&Easy Test enviroment setup

Solutions: CDK & Task work together

CDK

- Spins up VPC, EC2 with UserData

Taskfile /Script

- test mssh (instanconnect) with imdsv2 on/off

```
INSTANCEID:
  sh: aws cloudformation list-exports --region eu-central-1
  --query "Exports[?Name == 'demoInstanceId'].Value" --output text
```

```
    - aws ec2 modify-instance-metadata-options --instance-id {{.INSTANCEID}}
    --http-endpoint enabled --http-token required
```

Testresults IMDSv2 & Instanceconnect

- Software part on EC2 was not ready (but was quickly solved)
- You have to update sdk/cli/ssm agent *before* enabling IMDSv2, then everythings works together
- You have to update the instanceconnect package on the instance to newest version, which is
  - not included by default in AMI
  - maybe not available in repositories yet

=> Trust is good, testing is better

Second Part: Test the Cloud

Integration

- Deploy Constructs for testing *and* example code

Unit Test

- Which CloudFormation code is generated?

Snapshots

- For refactoring and hidden side effects

Integration Test

CDK Source: Functions are located in `/lib`

- `aws-cdk/packages/@aws-cdk/aws-ec2/lib`

Tests are located in `/test`

- `aws-cdk/packages/@aws-cdk/aws-ec2/test`
- Unit Tests: `.../aws-ec2/test/test.vpc.ts`
- Integration Tests: `.../aws-ec2/test/integ.vpc.ts`

Integration test are usage examples

- Execute with `cdk --app ../aws-ec2/test/integ.vpc.ts`

Example of an integration test `integ.vpc.ts`

```
// Test Security Group Rules
const sg = new ec2.SecurityGroup(stack, 'SG', { vpc });
const rules = [
  ec2.Port.icmpPing(),
  ec2.Port.icmpType(128),
  ec2.Port.allIcmp(),
  ec2.Port.allUdp(),
  ec2.Port.udp(123),
  ec2.Port.udpRange(800, 801),
];
for (const rule of rules) {
  sg.addIngressRule(ec2.Peer.anyIpv4(), rule);
}
app.synth();
```

Unit Testing

- What do we test?
  - Generated CloudFormation, not service behaviour
- How do we test
  - Typescript: Jest
  - Java: JUnit
  - Python: pytest
- Maturity in sample
  - Typescript: Test Resource and attributes
  - Java/Python: Test Resource

Unit Test source

```
test('SQS Queue Created', () => {
    const app = new cdk.App();
    // WHEN
    const stack = new TestCfn.TestCfnStack(app, 'MyTestStack');
    // THEN
    expectCDK(stack).to(haveResource("AWS::SQS::Queue",{
      VisibilityTimeout: 300,
    }));
});
```

Unit Test usage in the CDK source

Most Tests use `haveResource`

```
aws-cdk
grep "expect(stack).to(" packages/**/test/*ts | wc
     961    2910  114705

grep "expect(stack).to(haveR" packages/**/test/*ts | wc
     872    2676  105361
```

What to expect in cdk expect

Test for resources with

- `expect(stack).to(haveResource(`
- `expect(stack).to(countResources(`

Test whole template with

- `expect(stack).to(exactlyMatchTemplate({`
- `expect(stack).to(beASupersetOfTemplate({`
- `expect(stack).to(matchTemplate({`

Demo Unit Test - look under the hood

aka: "demo"

src/test-cfn

Snapshots

Why

- Have CloudFormation, write CDK
- Refactoring

How

- First run: `npm run test` - create Snapshot
- Second run: `npm run test` - diff Snapshot
- ReSnapshot: `npm test -- -u`

Snapshots - howto

- store current state in /test/snapshots

```
    expect(SynthUtils.toCloudFormation(stack)).toMatchSnapshot();
```

- run test
- refactor
- save new snapshot

```
npm test -- -u
```

Snapshots - look under the hood

aka "demo"

src/snapshot

Third Part: Inspect the Cloud

Test Resource with chef inspec

Test functionality

Chef InSpec DSL

> Chef InSpec is a run-time framework and rule language used to specify compliance, security, and policy requirements. It includes a collection of resources that help you write auditing controls quickly and easily.

The Chef InSpec DSL is a Ruby DSL for writing audit controls, which includes audit resources that you can invoke.

Are Resouces really created?

- Test with custom (ruby) DSL
- a "Test" just tests, a "Control" defines the impact

```
control 'aws-region-security-group-1.0' do

  impact 1.0
  title 'Ensure AWS Security Groups disallow ssh ingress from 0.0.0.0/0.'


  describe aws_security_group(group_name: 'FrontEnd', vpc_id: 'vpc-0f66cf8e7918f109c') do
    it { should exist }
    it { should_not allow_in(ipv4_range: '0.0.0.0/0', port: 22) }
  end

end
```

Inspec-AWS: Custom AWS Resources

- Used for compliance checks

Management	Security	Infrastructure
auto_scaling	iam	ec2
cloudtrail	kms	ebs
cloudwatch	security groups	rds
config	sts	s3
flow-log		SNS
		SQS
		vpc

Demo InSpec

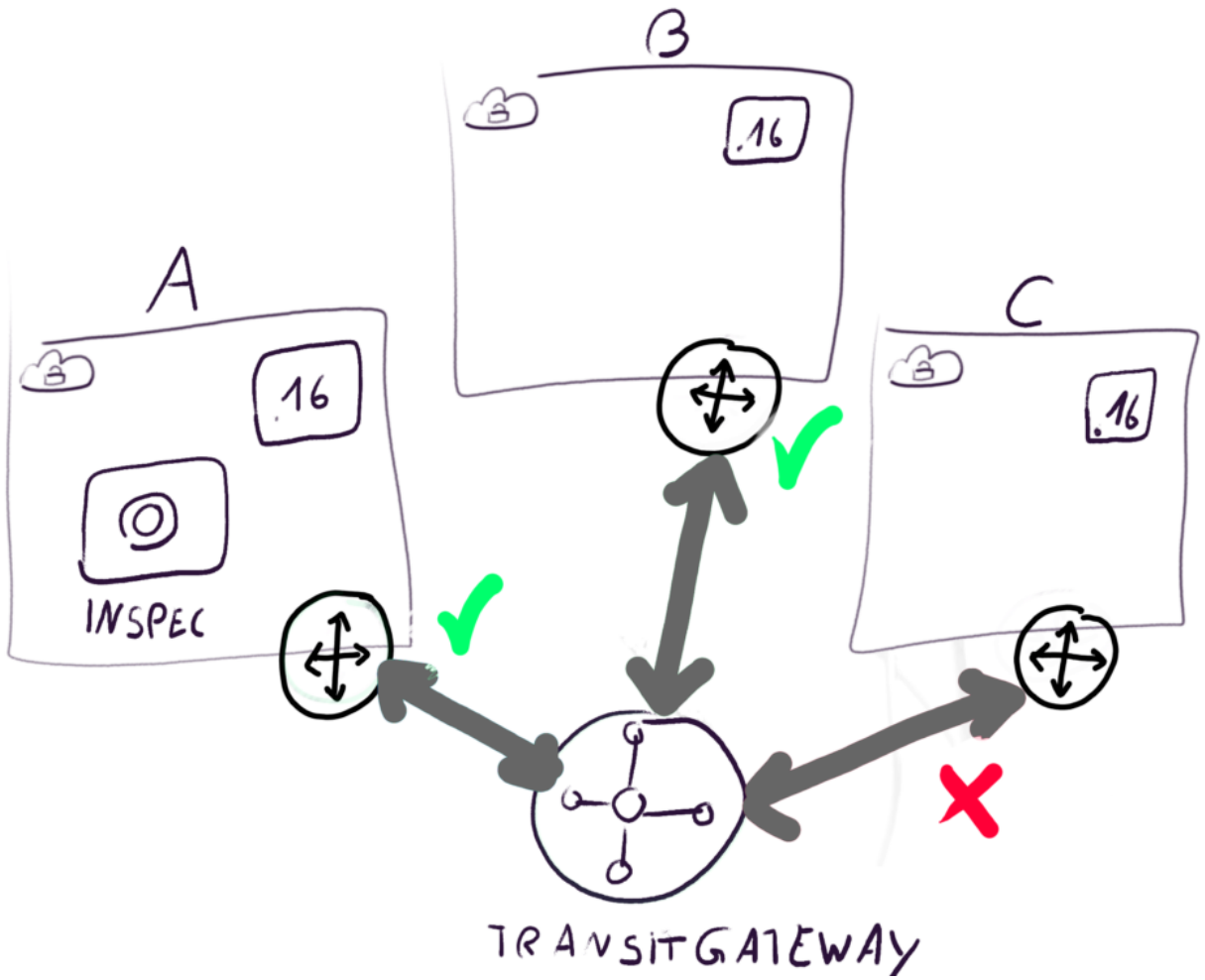
aka look under the hood

src/cdk-inspec

inspec-aws

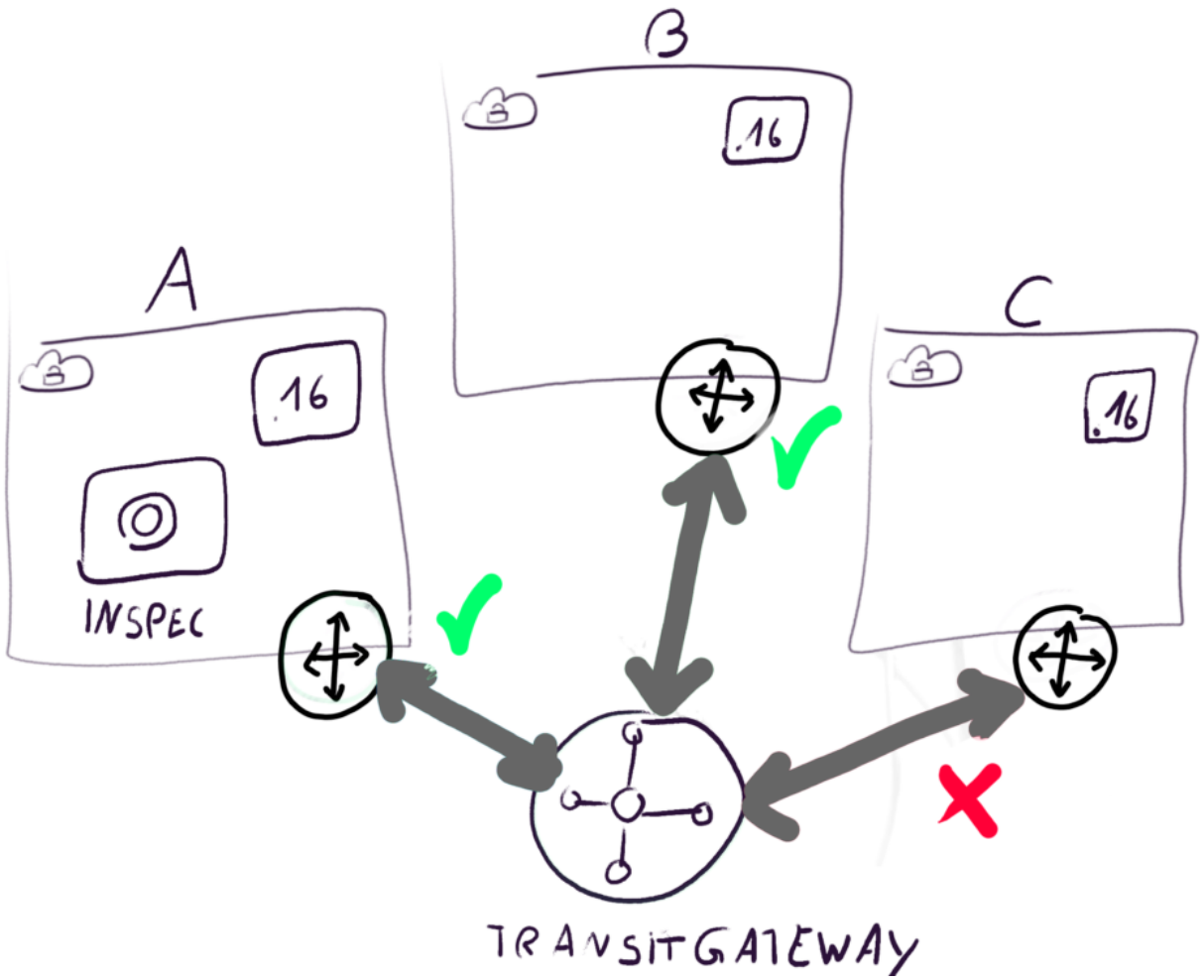
One Step further: Test Instances with Test kitchen

- After checking a resouce exists
- Testing whether it really works
- UseCase TransitGateway:
  Can EC2 instances in VPC A access web server on instance in VPC B across VPCs?



One Step further: Test Instances with Test kitchen

- CDK creates Webserver instances in A,B,C
- Test Kitchen spins up instance in chosen vpc
- Test Kitchen runs test: HTTP Request
- If TGW and routing works, test is ok



Platform Definition & Server Access Test

- CDK creates VPC and gives image_id and subnet_id

```
platforms:
  - name: amazon
    driver:
      region: eu-central-1
      image_id: ami-0cfbf4f6db41068ac
      instance_type: t2.large
      subnet_id: subnet-01cb511f85c1b2356
      tags:
        Name: "Kitchen Test Instance ec2"
```

```
describe http('http://10.0.1.16') do
  its('status') { should cmp 200 }
  its('body') { should eq 'Node A' }
end
```

Fresh from AWS Blog today

"Unit testing IAM policies across multiple accounts"

- Test IAM with SDK Tests

```
response = client.simulate_principal_policy(
    PolicySourceArn=source, ActionNames=actions
)`
```

- Uses Lambda as test agents

A CDK enabled Cloud Test Pyramid

Functionality Tests

Service Tests

Unit Tests

A CDK enabled Cloud Test Pyramid

- Functionality Tests
  - Use kitchen as a driver framework
  - write your own lambda
  - CDK creates test environment infrastructure
- Service Tests
  - Inspec as a DSK with predefined functionality
  - Write in your own language
  - CDK creates test environment infrastructure
- Unit Tests
  - Jest Test generated CloudFormation
  - Test single functions in Construct Code

Unser DevOps Triple

Get a tecRacer Training – https://www.tecracer.de/training

Terraform by HashiCorp

- *Terraform Interactive 101*
- *Terraform Interactive Masterclass*

AWS

- *tecRacer AWS Cloud Development Kit (CDK) Essentials*

CHEF

- *Chef Essentials*
- *Chef Foundations*
- *Chef Intermediate Topics*
- *Chef Automate Compliance*

Thank you !

Visit us

- www.tecracer.de
- awsblog.de
- github.com/tecracer

Image Credits

Photo by Andre Guerra on Unsplash

Photo by EJ Yao on Unsplash

Photo by Tommy Lisbin on Unsplash

Photo by Shane Aldendorff on Unsplash

Photo by Kunle Atekoja on Unsplash