AWS re:Invent

DOP302-R

Best practices for authoring AWS CloudFormation

Dan Blanco
Developer Advocate
AWS CloudFormation

Olivier Munn
Senior Product Manager
AWS CloudFormation

AWS re:Invent

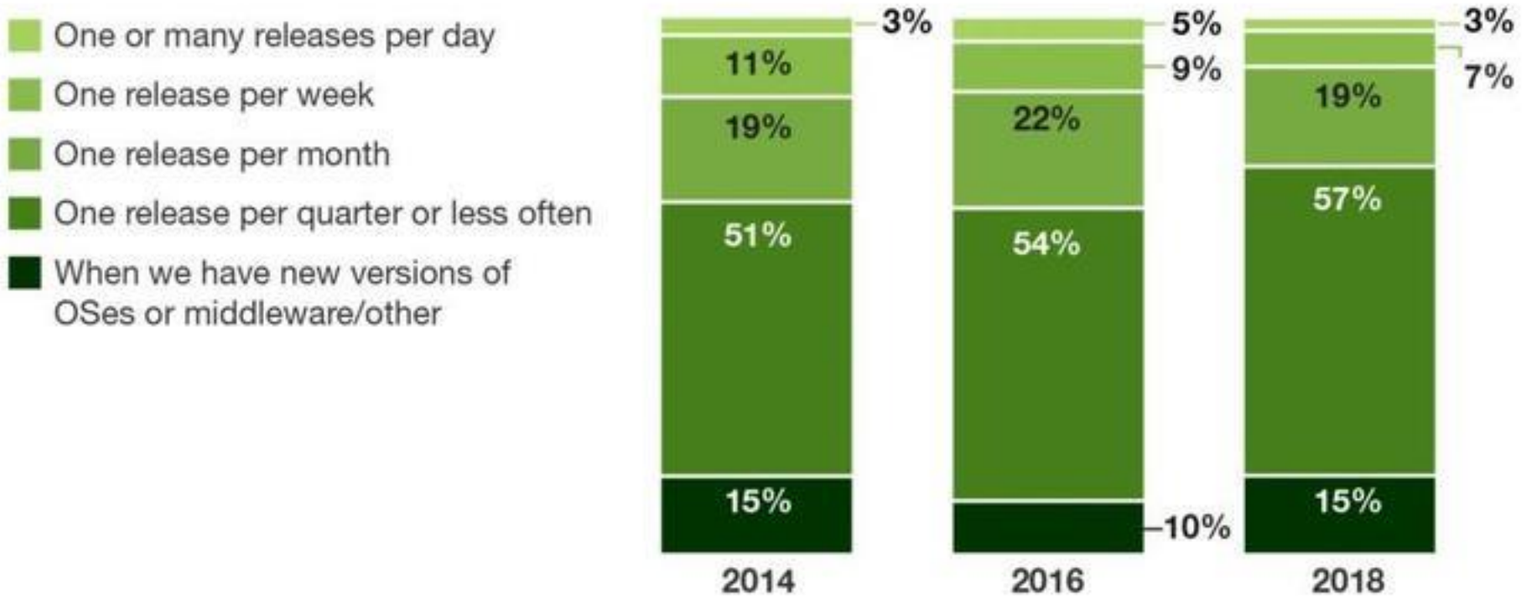
aws

Agenda

- Intro to infrastructure as code
- Authoring
- Testing
- Deploying
- Multiaccount/Multiregion
- Maintaining

How often do you go to production?

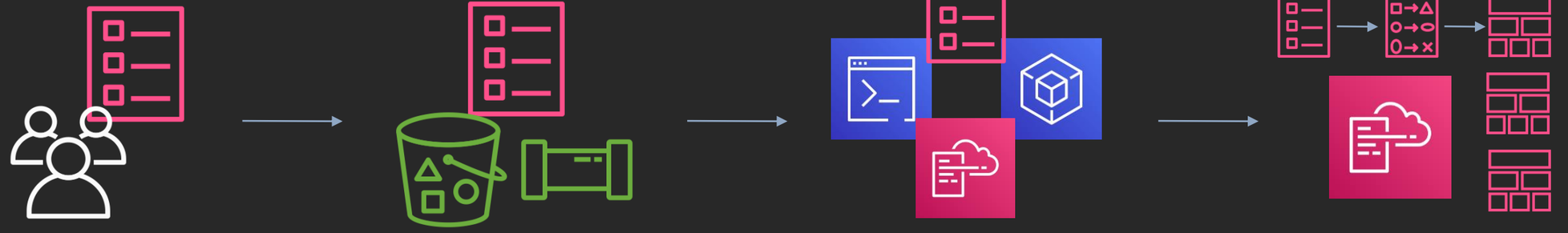
2-1 "How often does your team (or teams) release applications?"
(Percentages may not total 100 because of rounding)



Base: 637 to 2,727 global developers who work for a software company as a game developer, for internal IT, in the digital or design services industry, or in the tech services industry

Source: Forrester Analytics Business Technographics® Global Developer Survey, 2014, and Forrester Analytics Global Business Technographics Developer Surveys, 2016 and 2018

AWS CloudFormation in a nutshell



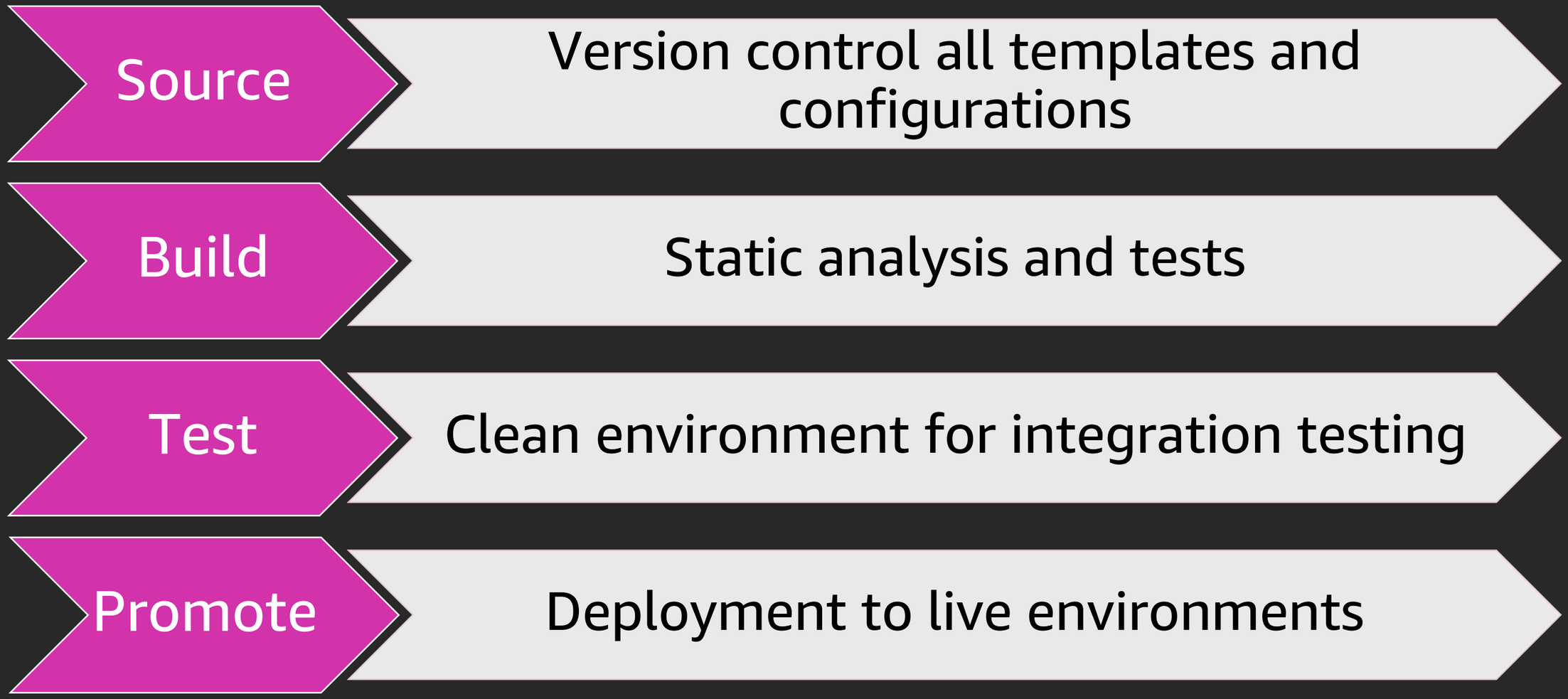
Code in YAML/JSON directly, use SAM or macros, translate high-level languages (CDK), or use sample templates

Upload local files via the browser console, from an S3 bucket, or via pipelines

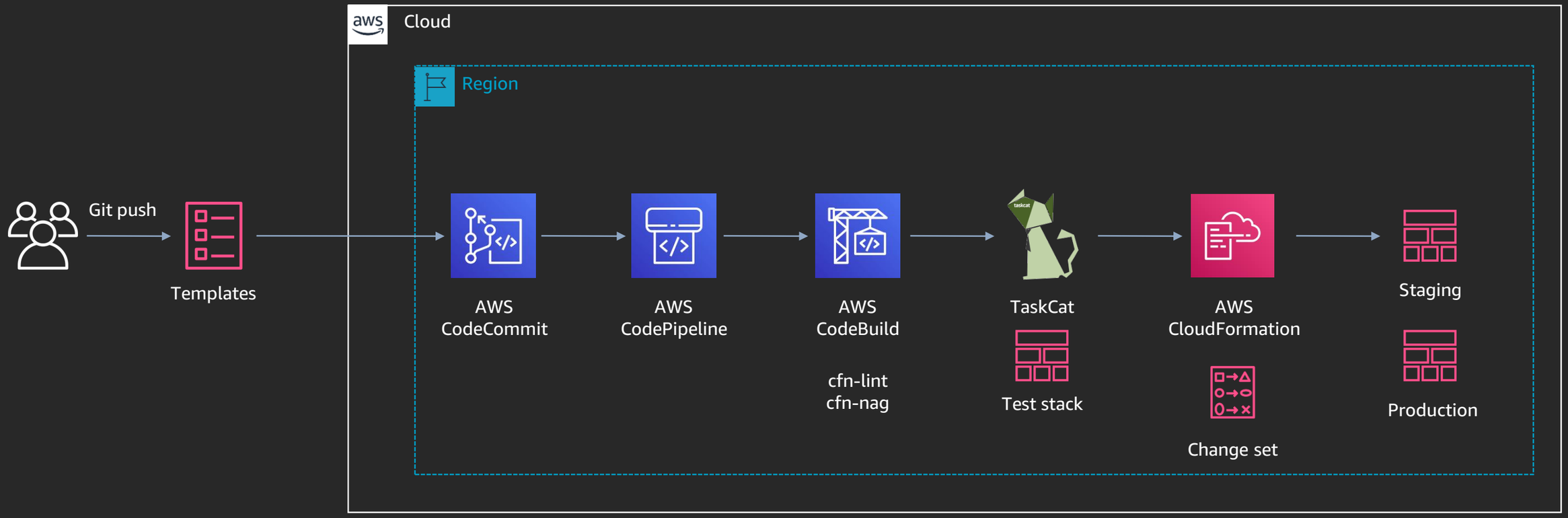
Create stacks or use console, CLI, or AWS SDK, or stack set instances across accounts and regions

Create and manage stacks, stack sets, and resources

Infrastructure as code is code



Start with a pipeline



Authoring your templates

Picking an editor

- Visual Studio Code
- Sublime Text
- Atom
- Cloud9
- IntelliJ
- PyCharm
- Others

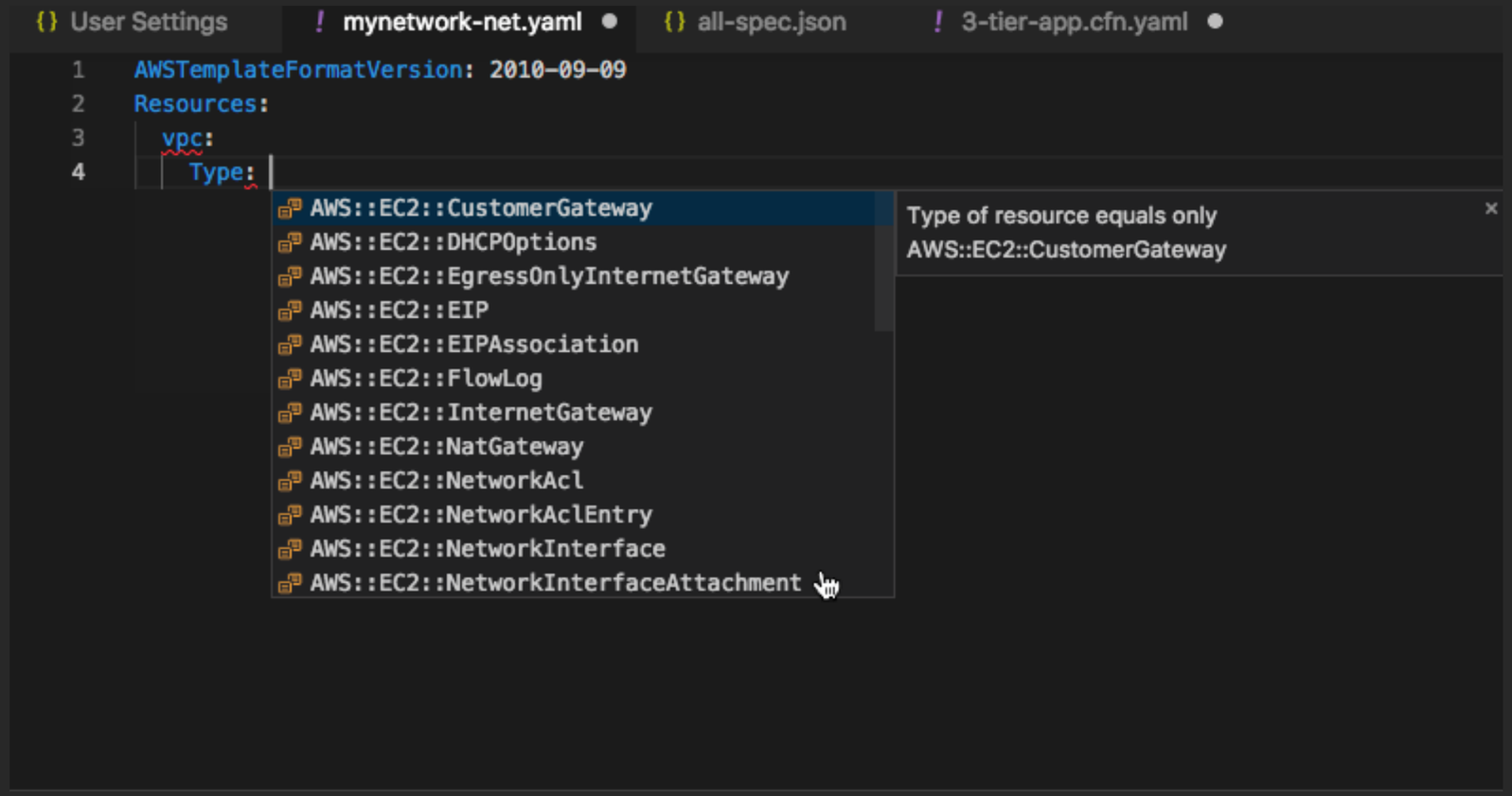
Linting your templates

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
Errors:
  Catch: Missing
Parameters:
  myParam:
    Type: String
    Default: String
    Description: String
Resources:
  ## Missing Properties
  MyEC2Instance1:
    Type: "AWS::EC2::Instance1"
  ## Fake Properties Key on main level
  ## Bad sub properties in BlockDeviceMappings/Ebs and NetworkInterfaces
  MyEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-2f726546"
      InstanceType: t1.micro
      KeyName: 1
      FakeKey: MadeYouLook
      BlockDeviceMappings:
        -
```

Severity	Provider	Description	Line
Warning	Cfn-Lint	Top level item Errors isn't valid	3:1
Warning	Cfn-Lint	Parameter myParam not used	6:1
Warning	Cfn-Lint	Invalid Type AWS::EC2::Instance1 for resource MyEC2Instance1	12:1
Warning	Cfn-Lint	Properties not defined for resource MyEC2Instance1	12:1
Warning	Cfn-Lint	Invalid Property FakeKey for resource MyEC2Instance	18:1
Warning	Cfn-Lint	Invalid Property BadSubX2Key for resource MyEC2Instance	26:1

- Plugins for Atom, Visual Studio Code, Sublime, VIM
- Process multiple files
- Handles conditions/Fn::If
- SAM local integration
- Available now on GitHub, more than 3,000,000 downloads

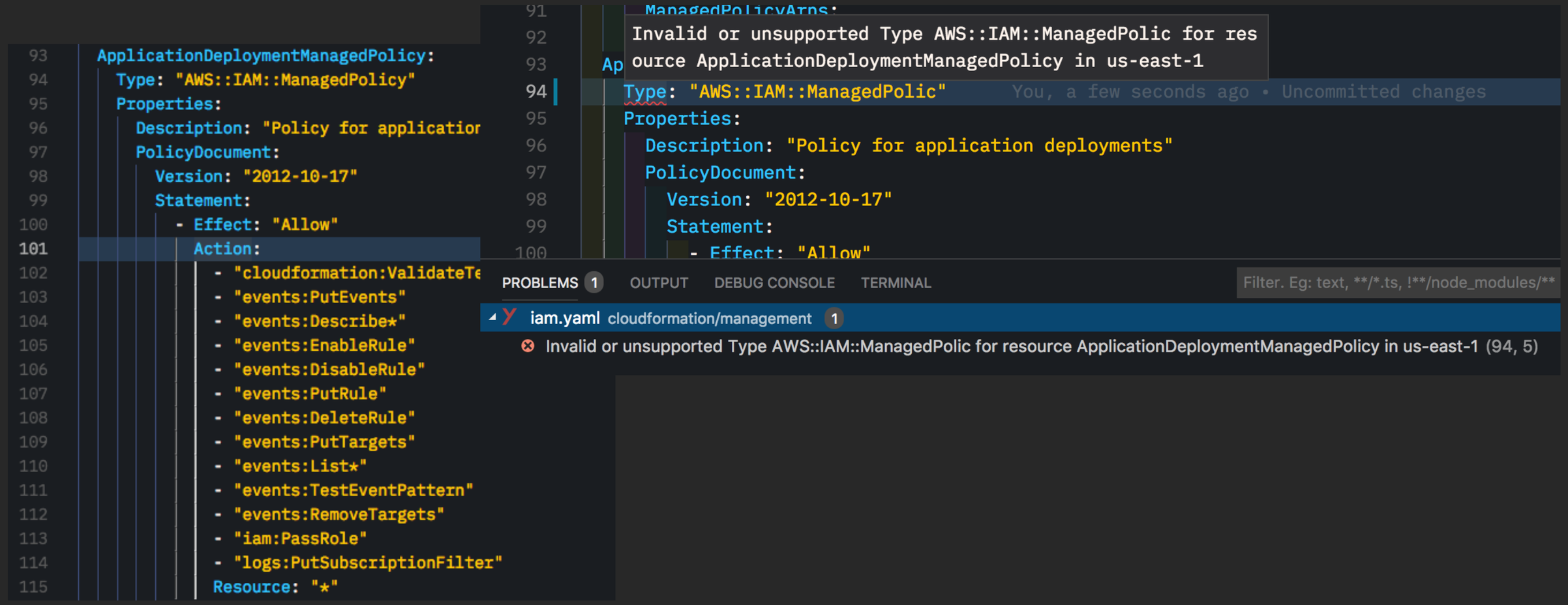
Adding autocomplete



- Works with Visual Studio Code/PyCharm
- Autocompletes resource types and lists required properties
- Autocompletes nonrequired properties as needed
- Links to resource type documentation

https://github.com/aws-cloudformation/aws-cloudformation-template-schema

Supercharging your editor



https://hodgkins.io/up-your-cloudformation-game-with-vscode

Authoring best practices

- *Parameters*: Avoids hardcoding of values, can add validation to users and improve UX with console grouping, labels, and descriptions; keep secrets in AWS Systems Manager Parameter Store and AWS Secrets Manager
- *Mappings*: As a case statement, helps maintain a set of information for various environments
- *Conditions*: Simple if/then statements (e.g., “if dev do this, if prod do that”)
- Imports and exports
- Use !Sub over !Join
- Leverage SSM parameter store for latest AMI instance IDs

YAML/JSON template

Parameters

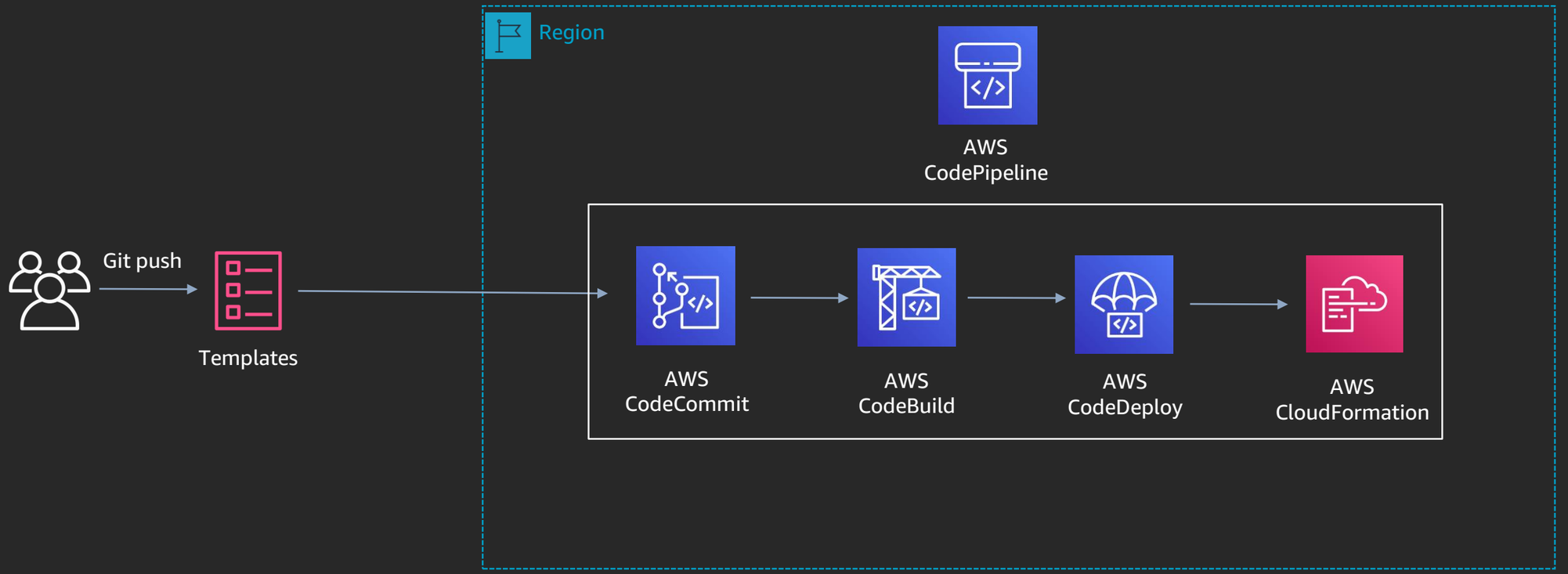
Mappings

Conditions

Resources

Outputs

Commit your template



Testing your templates

Wait, how do you test infrastructure?

> As a general rule, we want to be writing *less* code in the serverless world. It's quite possible to write a useful service in AWS using mostly configuration. For example, you can build a CRUD API using API Gateway and DynamoDB alone—no Lambda functions in the middle required.
>
> The more "serverless" you get, the less code you can usefully unit test, and the more you have to rely on tests of your *deployed infrastructure*.
>
> **— Forrest Brazeal**

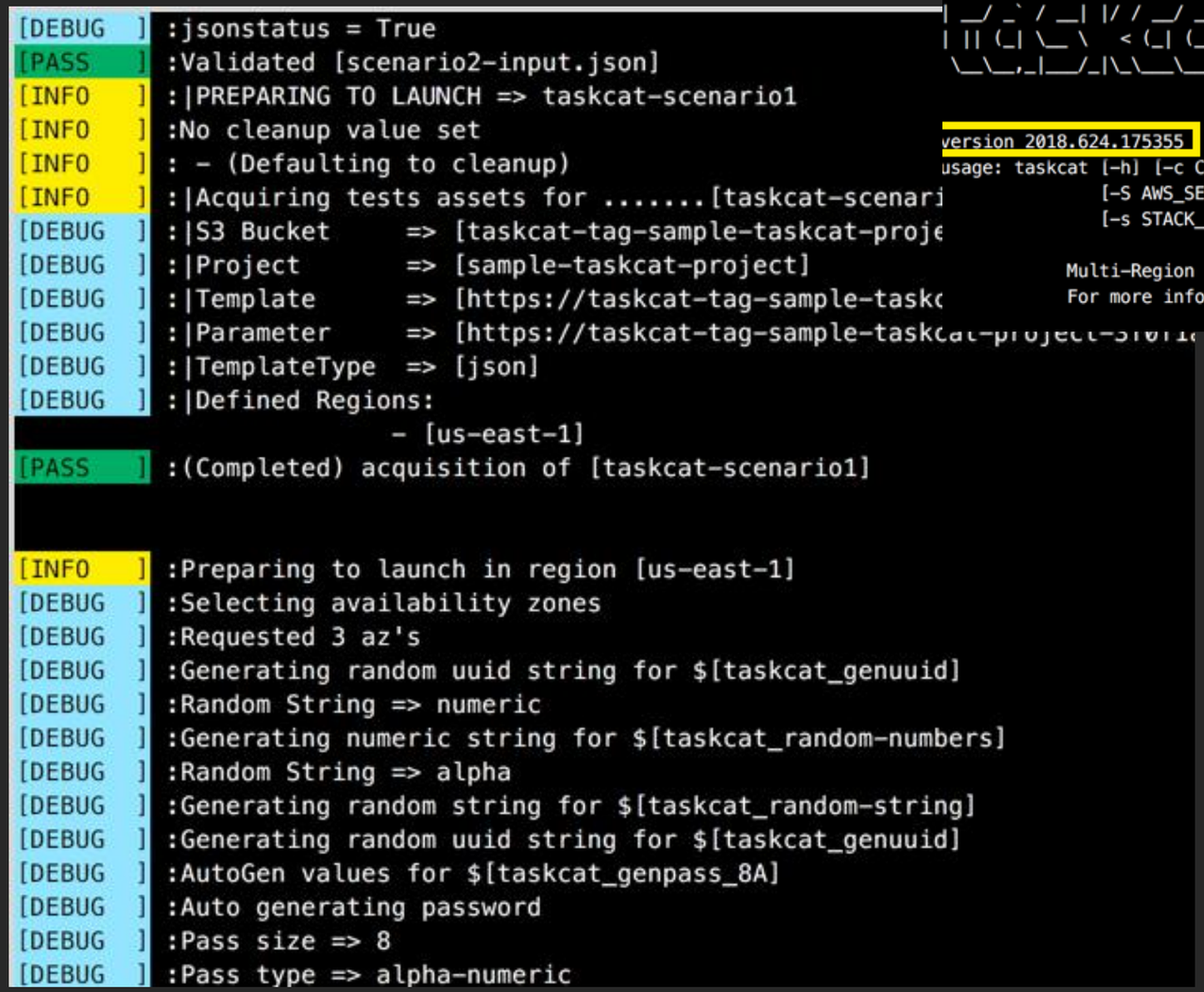
https://dev.to/trek10inc/ci-cd-aws-and-serverless-5-tips-i-learned-the-hard-way-223p

Linting headlessly

```
~/Documents/templates
λ cfn-lint wordpress-original.yaml
W7001 Mapping 'AWSInstanceType2NATArch' is defined but not used
wordpress-original.yaml:222:3

E3012 Property Resources/WebServerSecurityGroup/Properties/SecurityGroupIngress/0/FromPort should be of type Integer
wordpress-original.yaml:395:11

E3012 Property Resources/WebServerSecurityGroup/Properties/SecurityGroupIngress/0/ToPort should be of type Integer
wordpress-original.yaml:396:11

E3012 Property Resources/WebServerSecurityGroup/Properties/SecurityGroupIngress/1/FromPort should be of type Integer
wordpress-original.yaml:399:11

E3012 Property Resources/WebServerSecurityGroup/Properties/SecurityGroupIngress/1/ToPort should be of type Integer
wordpress-original.yaml:400:11
```

- Works completely headlessly
- Runs in a pipeline and prevents promotion if linting fails
- Great for pull requests & open-source projects

Testing with TaskCat

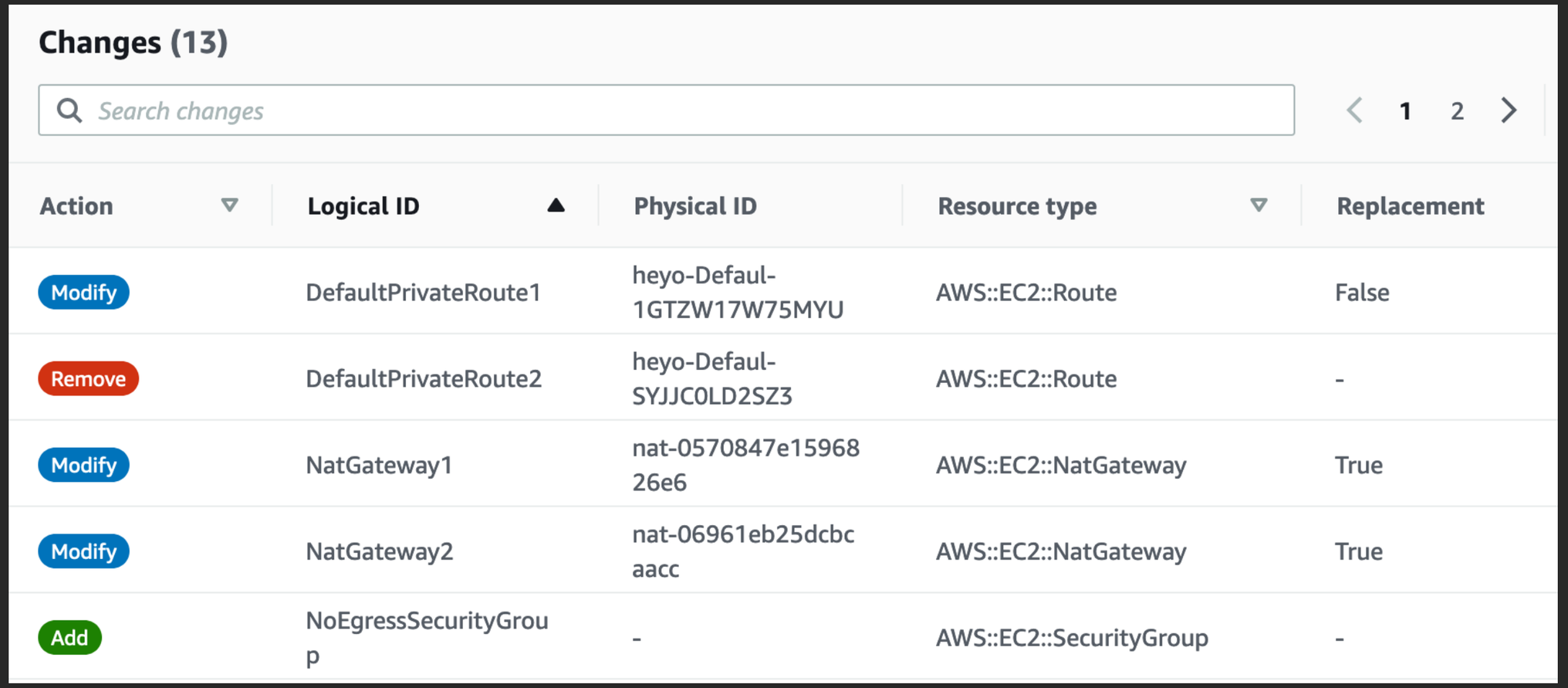


- From the AWS QuickStart team
- Open source
- Tests templates by creating stacks in multiple AWS regions simultaneously
- Catches problems that aren't obvious in a single template or stack
- Generates a report with a pass/fail grade for each region
- Great for AWS CloudFormation StackSets

https://github.com/aws-quickstarts/taskcat

Deploying your template

Change sets



Changes (13)

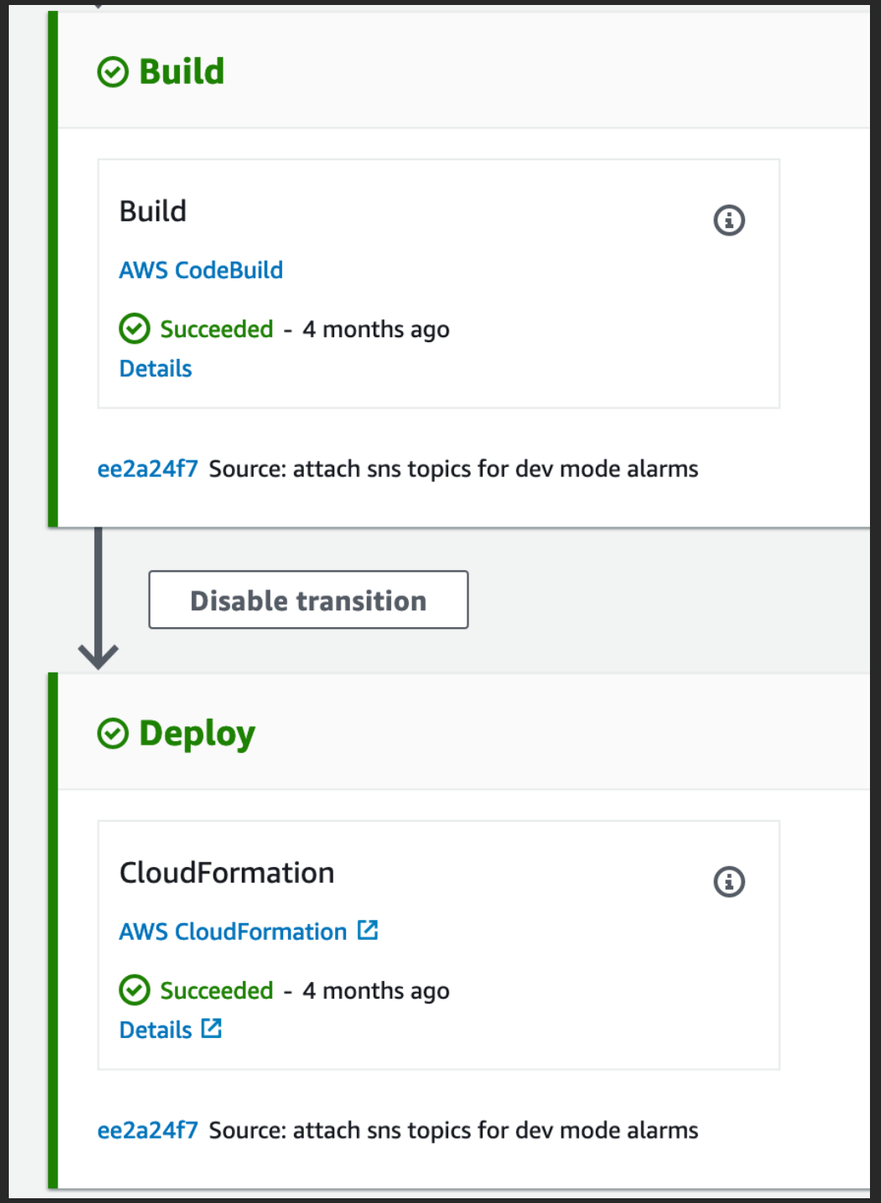
Search changes

1 2

Action	Logical ID	Physical ID	Resource type	Replacement
Modify	DefaultPrivateRoute1	heyo-Defaul-1GTZW17W75MYU	AWS::EC2::Route	False
Remove	DefaultPrivateRoute2	heyo-Defaul-SYJJC0LD2SZ3	AWS::EC2::Route	-
Modify	NatGateway1	nat-0570847e1596826e6	AWS::EC2::NatGateway	True
Modify	NatGateway2	nat-06961eb25dcbcaacc	AWS::EC2::NatGateway	True
Add	NoEgressSecurityGroup	-	AWS::EC2::SecurityGroup	-

Pipelines

- Automated
- Safe
- Repeatable
- Lock down access to deployed resources

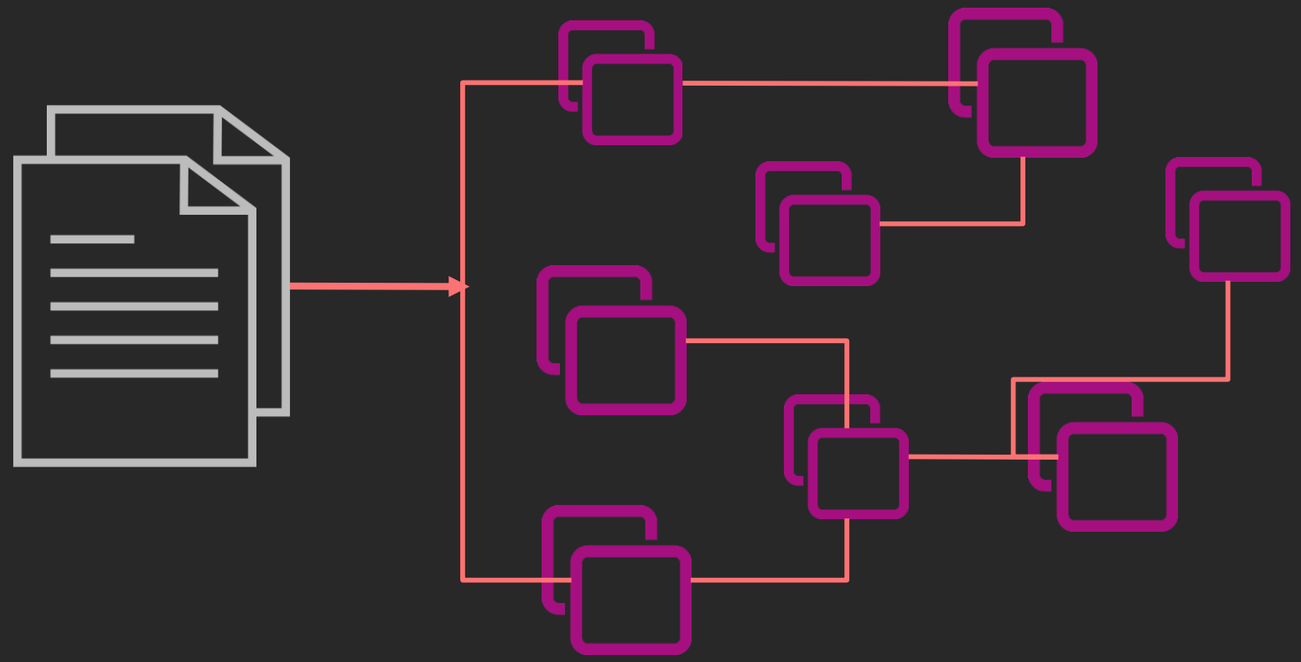


Multiaccount/multiregion

AWS CloudFormation StackSets

AWS CloudFormation StackSets extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation

- Fine-grained control of instance updates
- *New*: Higher limits by default (100 stack sets, 2,000 instances)
- *New*: Automatically deploy to new accounts in an organizational unit (OU)
- *New*: Automatically manage trust relationships via OUs



AWS CloudFormation StackSets common use cases

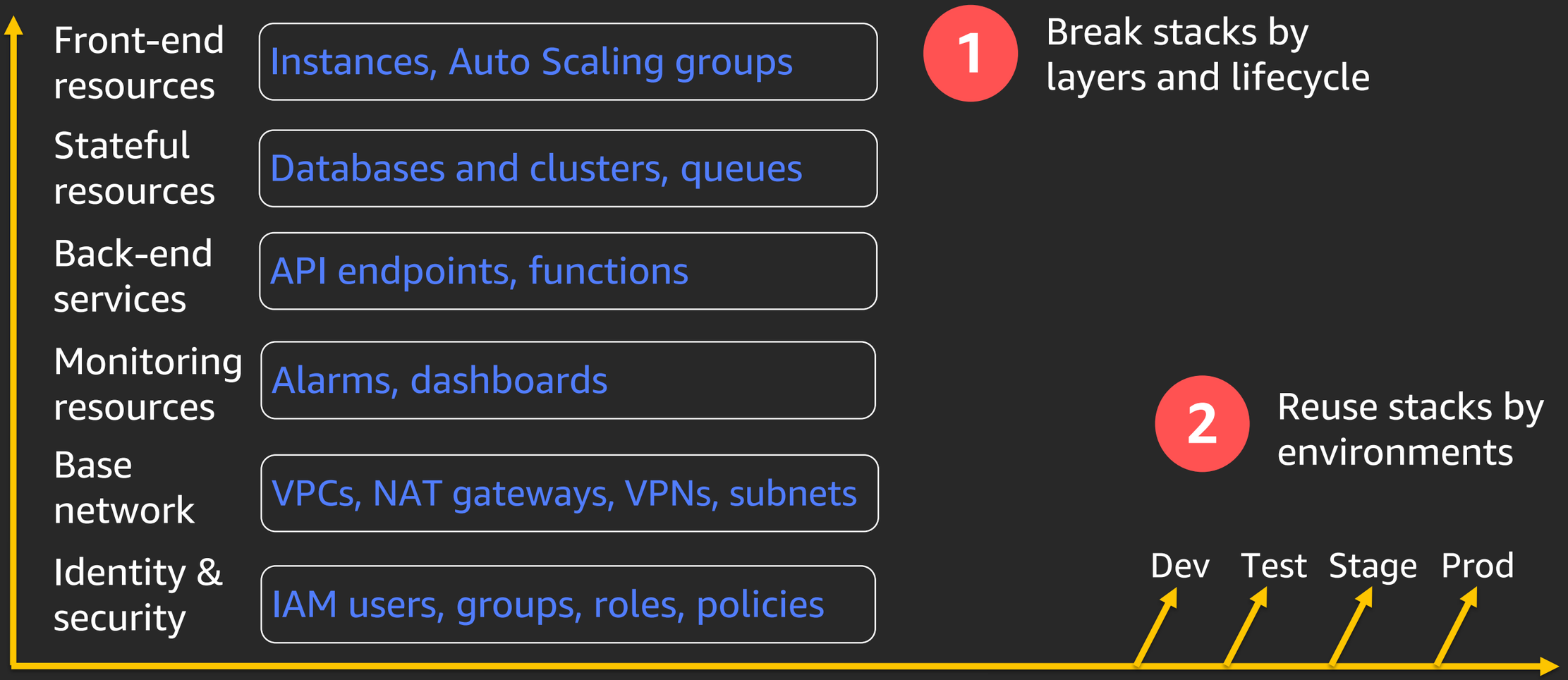
Seeding new accounts with critical prerequisite resources before the account is used	AWS CloudTrail, AWS Identity and Access Management (IAM), AWS Config
Separately set up groups of accounts that have similar purposes consistently	Developers/QA/admins
Controlled rollout of resource changes across multiple accounts and regions	Multiregion production environments

AWS CloudFormation StackSets best practices

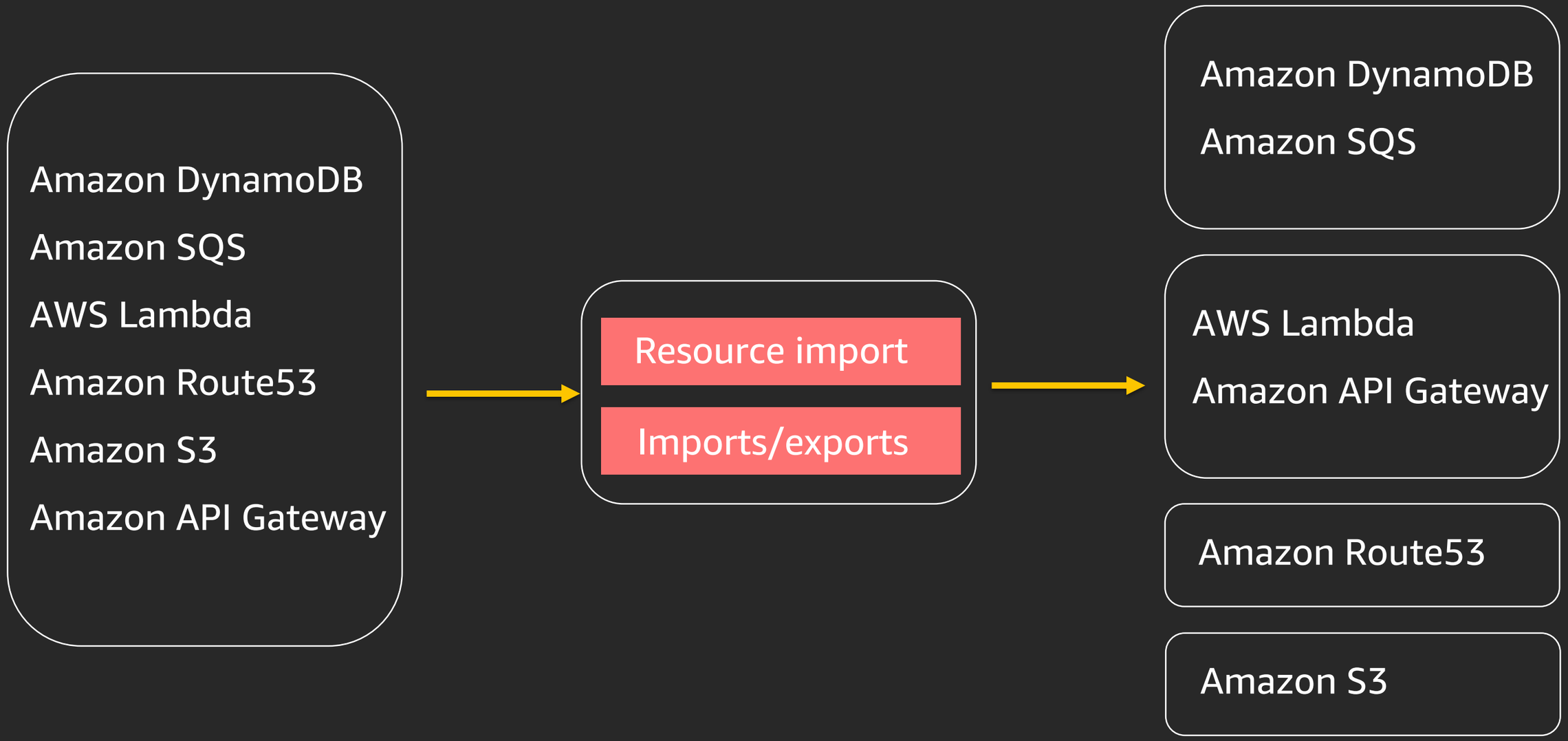
- Partially deploy your AWS CloudFormation StackSet updates to reduce blast radius; great for sanity testing/releasing incrementally
- Depending on your speed needs, consider setting a higher concurrent account limit
- Use parameter overrides to define specific parameters in account–region pairs
- Separate stacks by function and frequency of changes needed

Maintaining your stacks

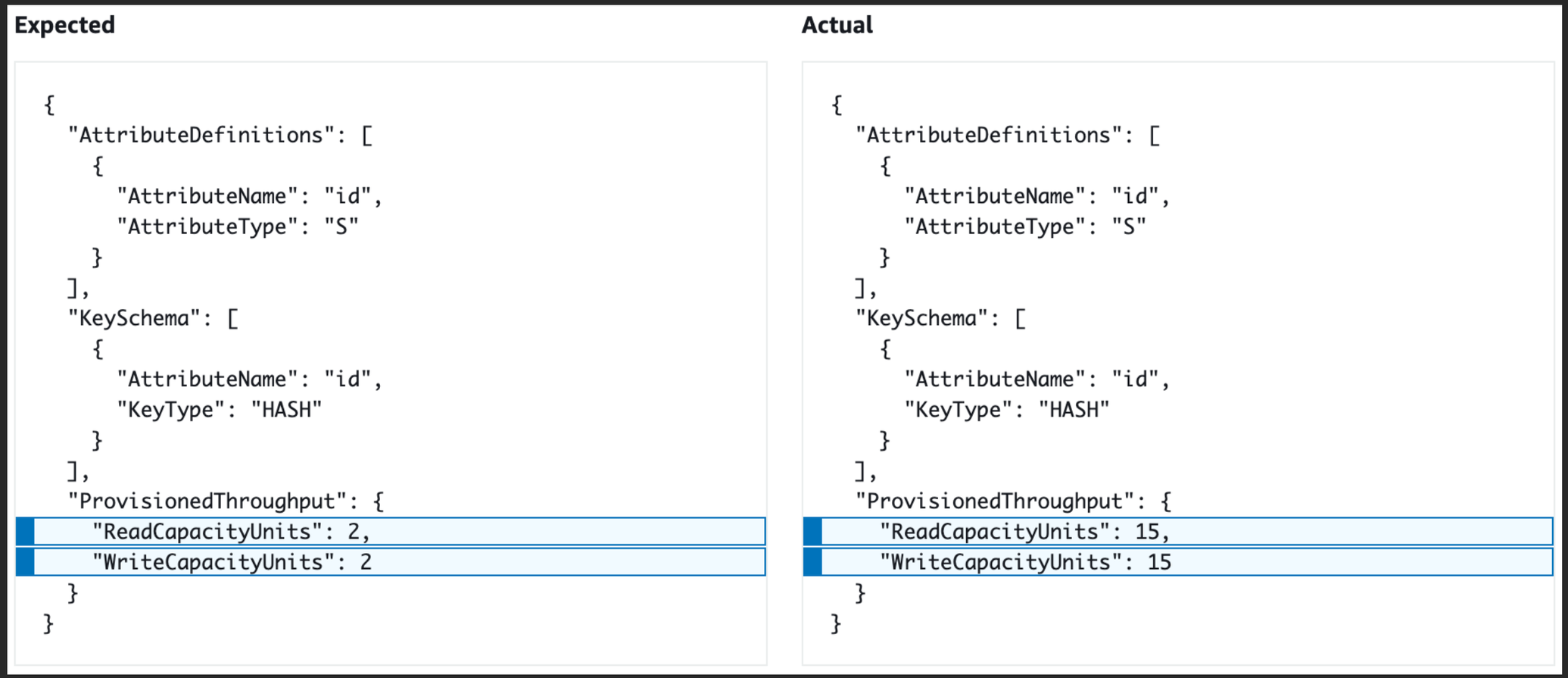
Stacks by lifecycle



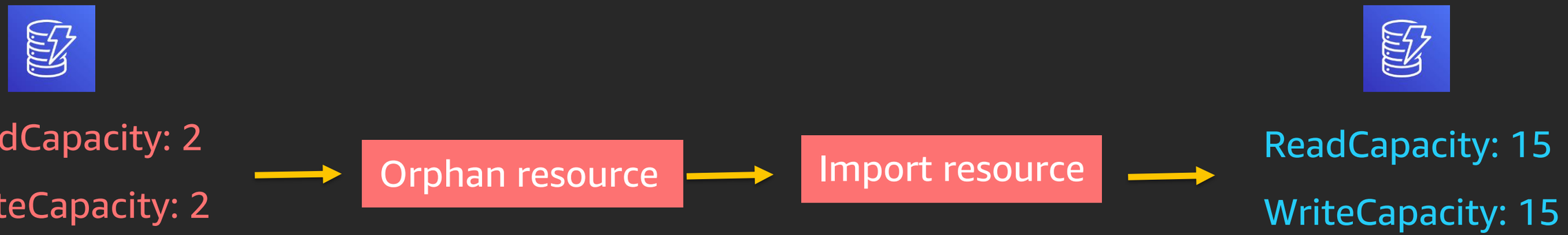
Refactoring existing stacks



Detecting drift



Remediating drift

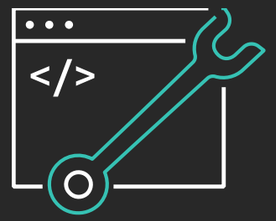


In summary

- Put all your assets in version control from the start
- Leverage modern editors and plugins
- Optimize your editor for AWS CloudFormation
- Use a pipeline
- Use cfn-lint, both for editing and in the pipeline
- Test on different environments with TaskCat
- Use change sets and pipelines to deploy safely
- Refactor your stacks for reuse (parameters, mappings, and so on)
- Smaller templates are easier to test and maintain, quicker to deploy
- Refactor large stacks with resource import
- Detect & remediate drift
- Consider AWS CloudFormation StackSets for cross-account/cross-region use cases

Learn DevOps with AWS Training and Certification

Resources created by the experts at AWS to propel your organization and career forward



Take free digital training to learn best practices for developing, deploying, and maintaining applications

Classroom offerings, like DevOps Engineering on AWS, feature AWS expert instructors and hands-on activities

Validate expertise with the **AWS Certified DevOps Engineer - Professional** or **AWS Certified Developer - Associate** exams

Visit aws.amazon.com/training/path-developing/

aws training and certification

Thank you!

Dan Blanco
@thedanblanco

Olivier Munn
olivmunn@amazon.com

Please complete the session survey in the mobile app.