

The background is a vibrant, multi-colored gradient. It features a diagonal split between a blue-purple gradient on the left and a yellow-orange gradient on the right. The text 'AWS re:Invent' is positioned on the left side, with 'AWS' in a smaller font above 're:Invent'.

AWS  
re:Invent

**D O P 3 0 2 - R**

# Best practices for authoring AWS CloudFormation

## **Dan Blanco**

Developer Advocate  
AWS CloudFormation

## **Olivier Munn**

Senior Product Manager  
AWS CloudFormation

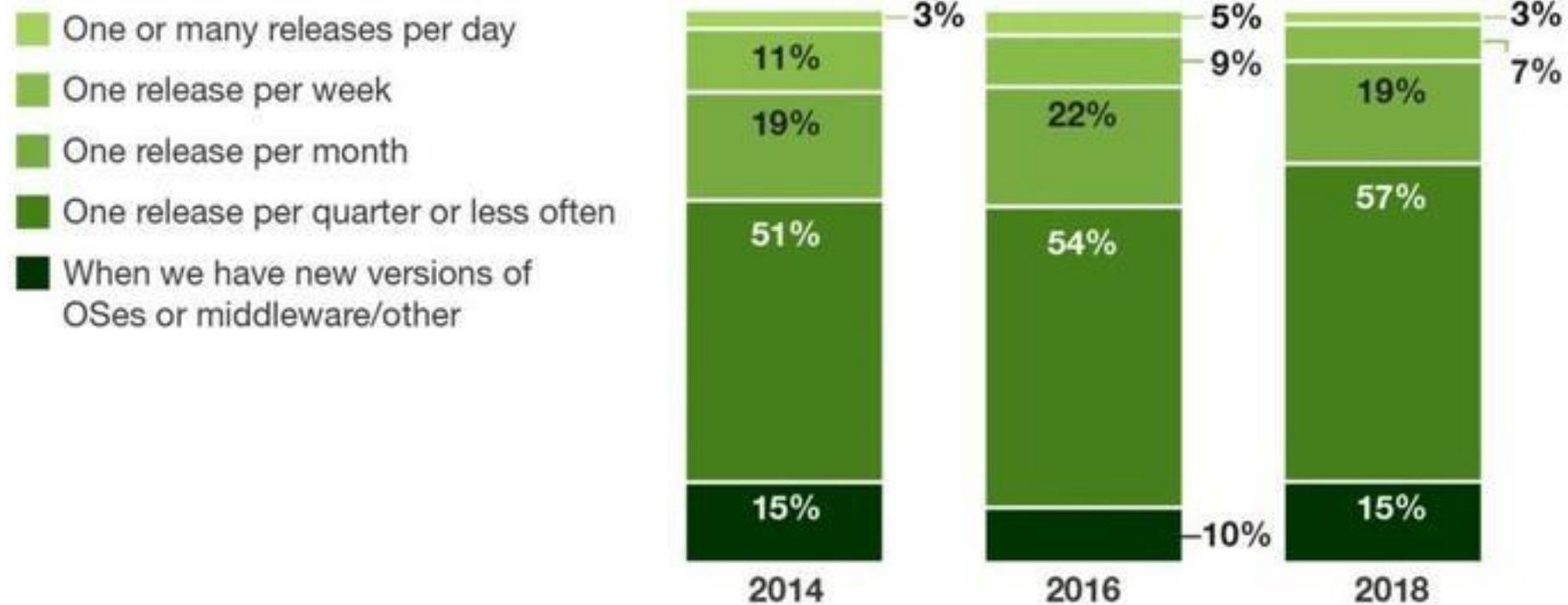
# Agenda

- 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?”

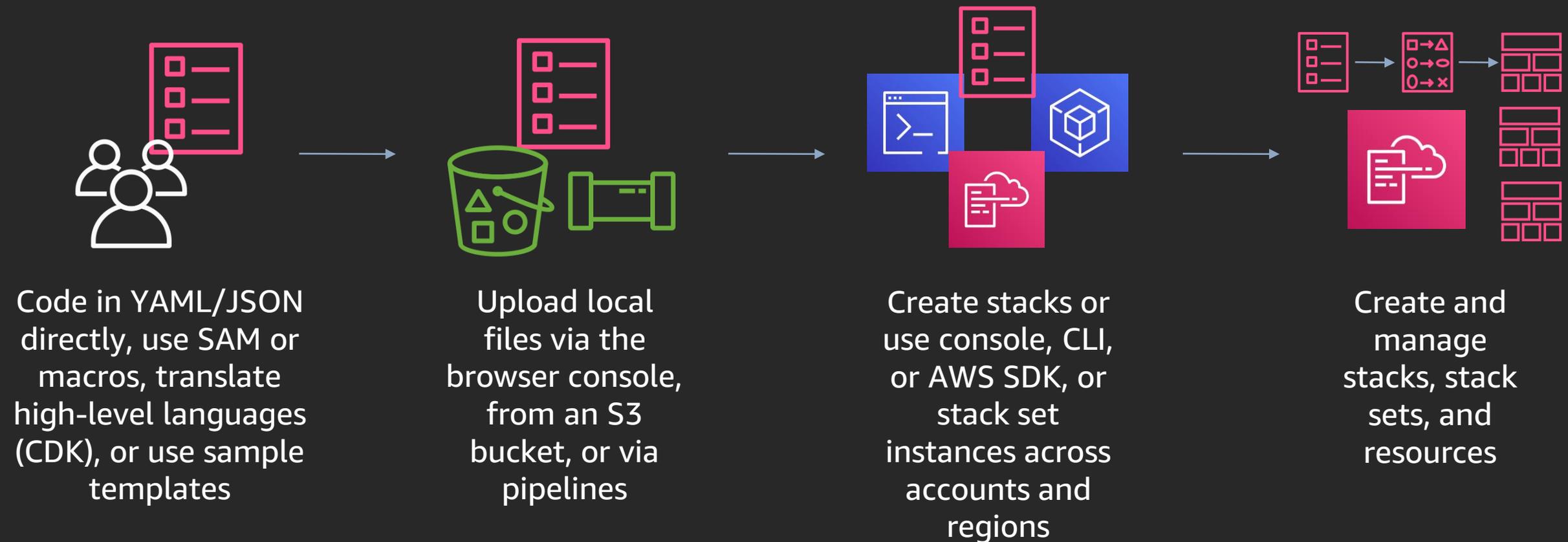
(Percentages may not total 100 because of rounding)



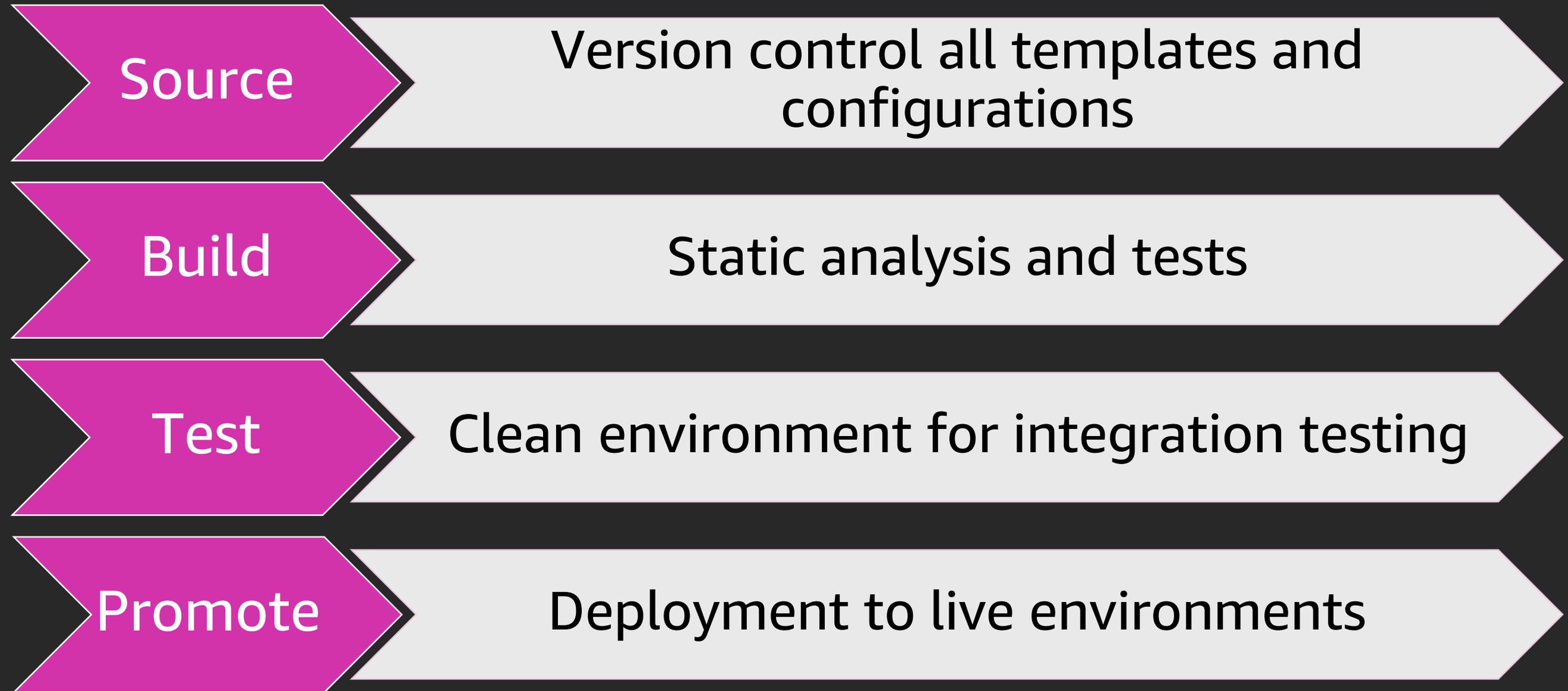
Base: 637 to 2,727 global developers who work for a software company as a game developer, for internal IT, in the digital or design services industry, or in the tech services industry

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

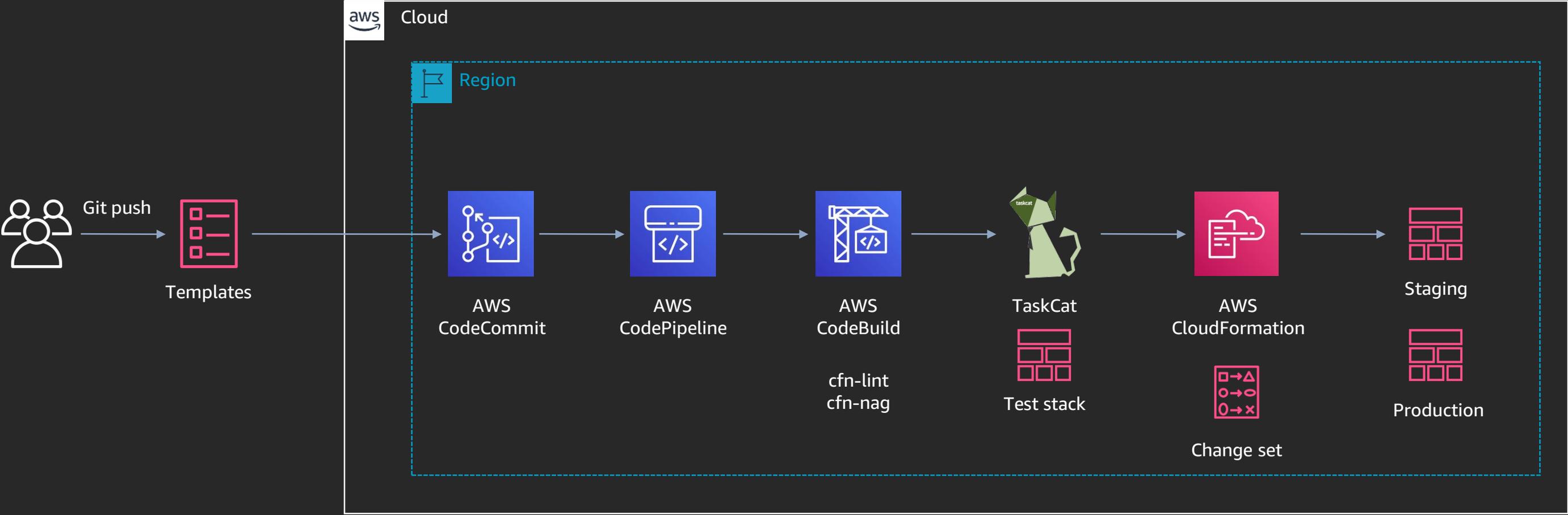
# AWS CloudFormation in a nutshell



# 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

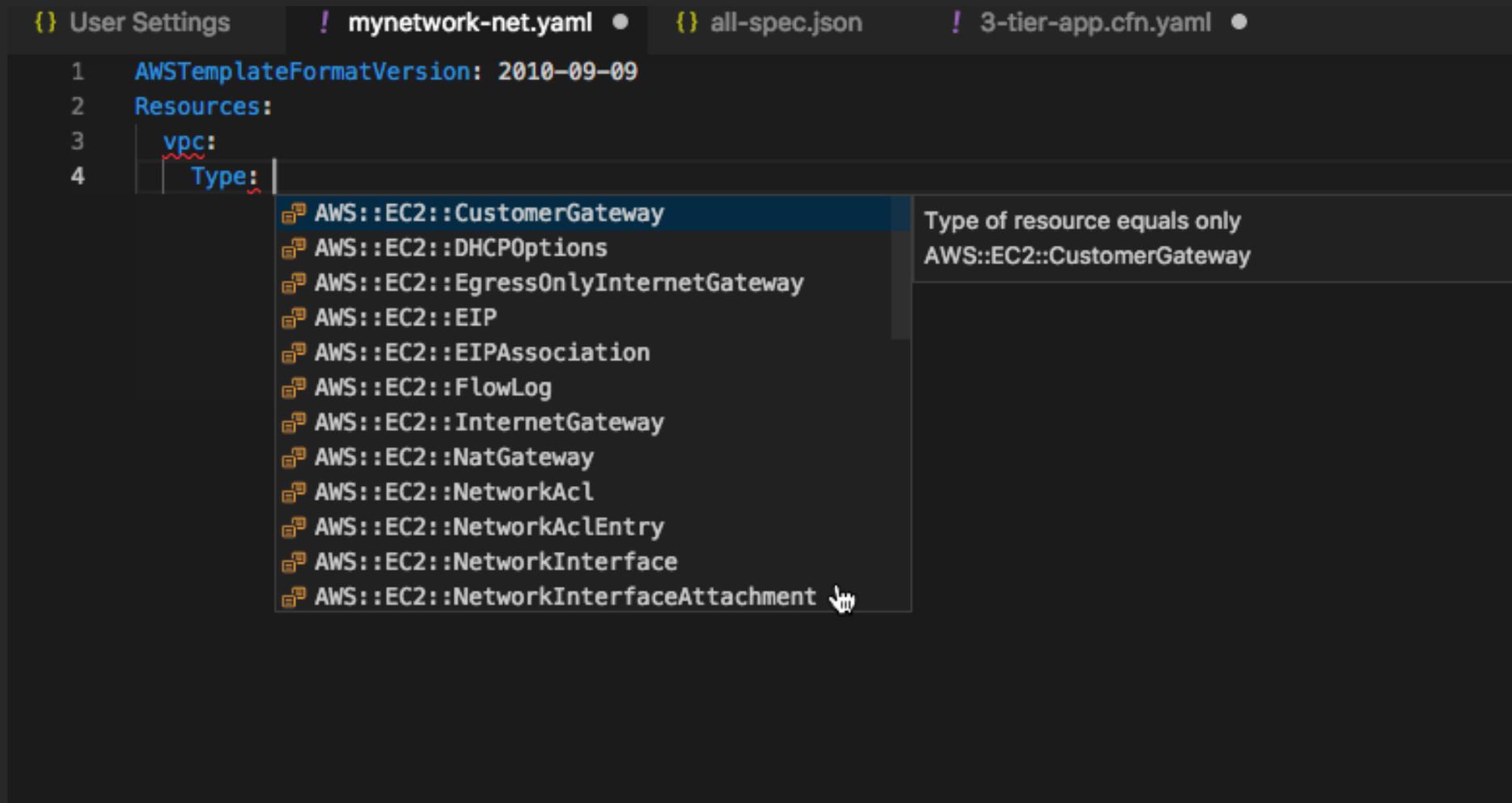
# Linting your templates

```
1  AWSTemplateFormatVersion: "2010-09-09"
2  Description: A sample template
3  ● Errors:
4    Catch: Missing
5  Parameters:
6    ● myParam:
7      Type: String
8      Default: String
9      Description: String
10 Resources:
11   ## Missing Properties
12   ● MyEC2Instance1:
13     Type: "AWS::EC2::Instance1"
14   ## Fake Properties Key on main level
15   ## Bad sub properties in BlockDeviceMappings/Ebs and NetworkInterfaces
16   MyEC2Instance:
17     Type: "AWS::EC2::Instance"
18   ● Properties:
19     ImageId: "ami-2f726546"
20     InstanceType: t1.micro
21     KeyName: 1
22     FakeKey: MadeYouLook
23     BlockDeviceMappings:
24     -
```

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



```
1 AWSTemplateFormatVersion: 2010-09-09
2 Resources:
3   vpc:
4     Type:
```

The screenshot shows a dropdown menu with the following options:

- AWS::EC2::CustomerGateway
- AWS::EC2::DHCPOptions
- AWS::EC2::EgressOnlyInternetGateway
- AWS::EC2::EIP
- AWS::EC2::EIPAssociation
- AWS::EC2::FlowLog
- AWS::EC2::InternetGateway
- AWS::EC2::NatGateway
- AWS::EC2::NetworkAcl
- AWS::EC2::NetworkAclEntry
- AWS::EC2::NetworkInterface
- AWS::EC2::NetworkInterfaceAttachment

A tooltip is visible for the selected item, showing: "Type of resource equals only AWS::EC2::CustomerGateway".

- 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

```
93 ApplicationDeploymentManagedPolicy:
94   Type: "AWS::IAM::ManagedPolicy"
95   Properties:
96     Description: "Policy for application
97     PolicyDocument:
98       Version: "2012-10-17"
99       Statement:
100         - Effect: "Allow"
101         Action:
102           - "cloudformation:ValidateTe
103           - "events:PutEvents"
104           - "events:Describe*"
105           - "events:EnableRule"
106           - "events:DisableRule"
107           - "events:PutRule"
108           - "events>DeleteRule"
109           - "events:PutTargets"
110           - "events:List*"
111           - "events:TestEventPattern"
112           - "events:RemoveTargets"
113           - "iam:PassRole"
114           - "logs:PutSubscriptionFilter"
115   Resource: "*"

91 ManagedPolicyArns:
92   Invalid or unsupported Type AWS::IAM::ManagedPolic for res
93   Ap ource ApplicationDeploymentManagedPolicy in us-east-1
94   Type: "AWS::IAM::ManagedPolic"
95   Properties:
96     Description: "Policy for application deployments"
97     PolicyDocument:
98       Version: "2012-10-17"
99     Statement:
100       - Effect: "Allow"
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

iam.yaml cloudformation/management 1

Invalid or unsupported Type AWS::IAM::ManagedPolic for resource ApplicationDeploymentManagedPolicy in us-east-1 (94, 5)

Filter. Eg: text, \*\*/\*.ts, !\*\*/node\_modules/\*\*

<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

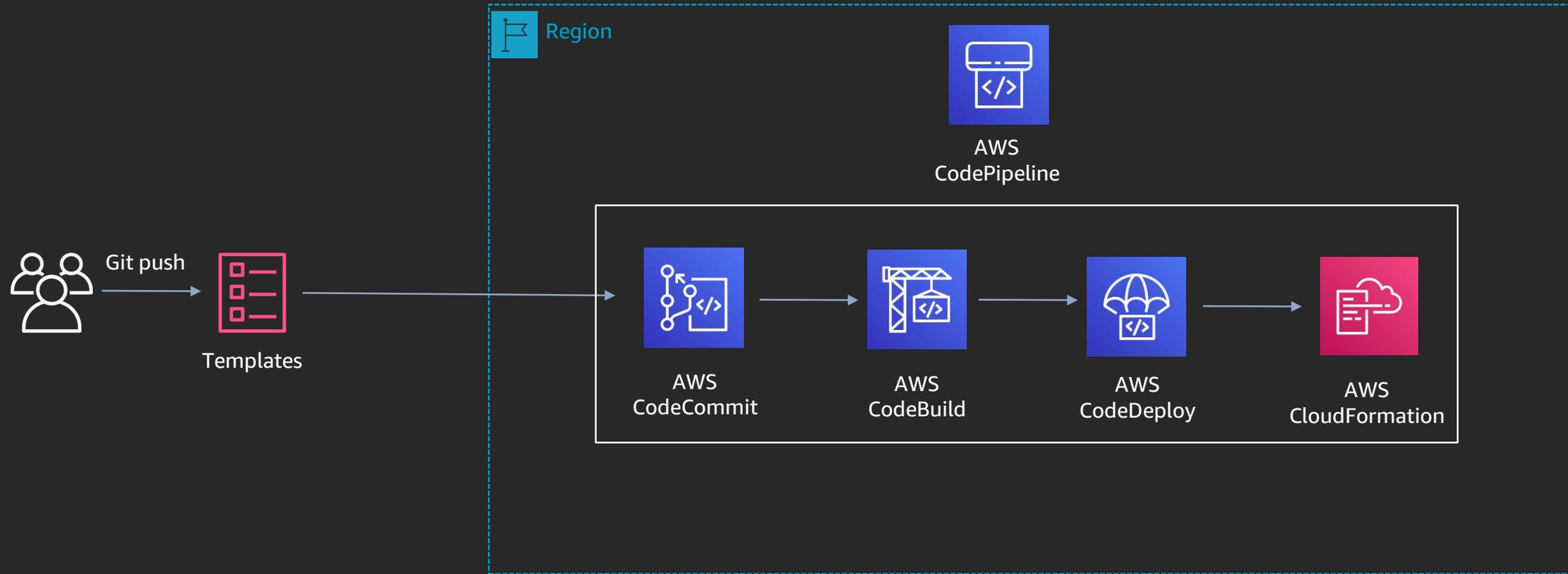
Mappings

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

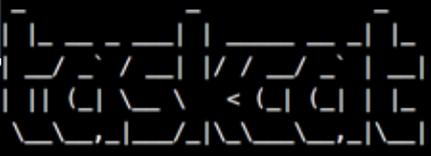
E3012 Property Resources/WebServerSecurityGroup/Properties/SecurityGroupIngress/1/ToPort should be of type Integer
wordpress-original.yaml:400:11
```

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

# Testing with TaskCat

```
[DEBUG ] :jsonstatus = True
[PASS  ] :Validated [scenario2-input.json]
[INFO  ] :|PREPARING TO LAUNCH => taskcat-scenario1
[INFO  ] :No cleanup value set
[INFO  ] : - (Defaulting to cleanup)
[INFO  ] :|Acquiring tests assets for .....[taskcat-scenari
[DEBUG ] :|S3 Bucket      => [taskcat-tag-sample-taskcat-proje
[DEBUG ] :|Project        => [sample-taskcat-project]
[DEBUG ] :|Template       => [https://taskcat-tag-sample-taskc
[DEBUG ] :|Parameter     => [https://taskcat-tag-sample-taskcat-project-31011
[DEBUG ] :|TemplateType  => [json]
[DEBUG ] :|Defined Regions:
                - [us-east-1]
[PASS  ] : (Completed) acquisition of [taskcat-scenario1]

[INFO  ] :Preparing to launch in region [us-east-1]
[DEBUG ] :Selecting availability zones
[DEBUG ] :Requested 3 az's
[DEBUG ] :Generating random uuid string for ${taskcat_genuuid}
[DEBUG ] :Random String => numeric
[DEBUG ] :Generating numeric string for ${taskcat_random-numbers}
[DEBUG ] :Random String => alpha
[DEBUG ] :Generating random string for ${taskcat_random-string}
[DEBUG ] :Generating random uuid string for ${taskcat_genuuid}
[DEBUG ] :AutoGen values for ${taskcat_genpass_8A}
[DEBUG ] :Auto generating password
[DEBUG ] :Pass size => 8
[DEBUG ] :Pass type => alpha-numeric
```



```
version 2018.624.175355
usage: taskcat [-h] [-c CONFIG_YML] [-P BOTO_PROFILE] [-A AWS_ACCESS_KEY]
              [-S AWS_SECRET_KEY] [-n] [-N] [-p] [-v] [-m] [-t TAG]
              [-s STACK_PREFIX]

Multi-Region CloudFormation Test Deployment Tool
For more info see: http://taskcat.io
```

```
version 2018.615.181319
[INFO ] : A newer version of taskcat is available (2018.624.
[INFO ] : To upgrade pip version [ pip install --upgrade
[INFO ] : To upgrade docker version [ docker pull taskcat/t
```

A yellow arrow points from the 'version 2018.624.175355' text in the middle terminal to the 'version 2018.615.181319' text in the right terminal.

- 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)**

Q Search changes < 1 2 >

Action ▾	Logical ID ▲	Physical ID	Resource type ▾	Replacement
Modify	DefaultPrivateRoute1	heyo-Default-1GTZW17W75MYU	AWS::EC2::Route	False
Remove	DefaultPrivateRoute2	heyo-Default-SYJJCOLD2SZ3	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

**Build**

Build ⓘ

[AWS CodeBuild](#)

✔ Succeeded - 4 months ago

[Details](#)

ee2a24f7 Source: attach sns topics for dev mode alarms

Disable transition

**Deploy**

CloudFormation ⓘ

[AWS CloudFormation](#) [↗](#)

✔ Succeeded - 4 months ago

[Details](#) [↗](#)

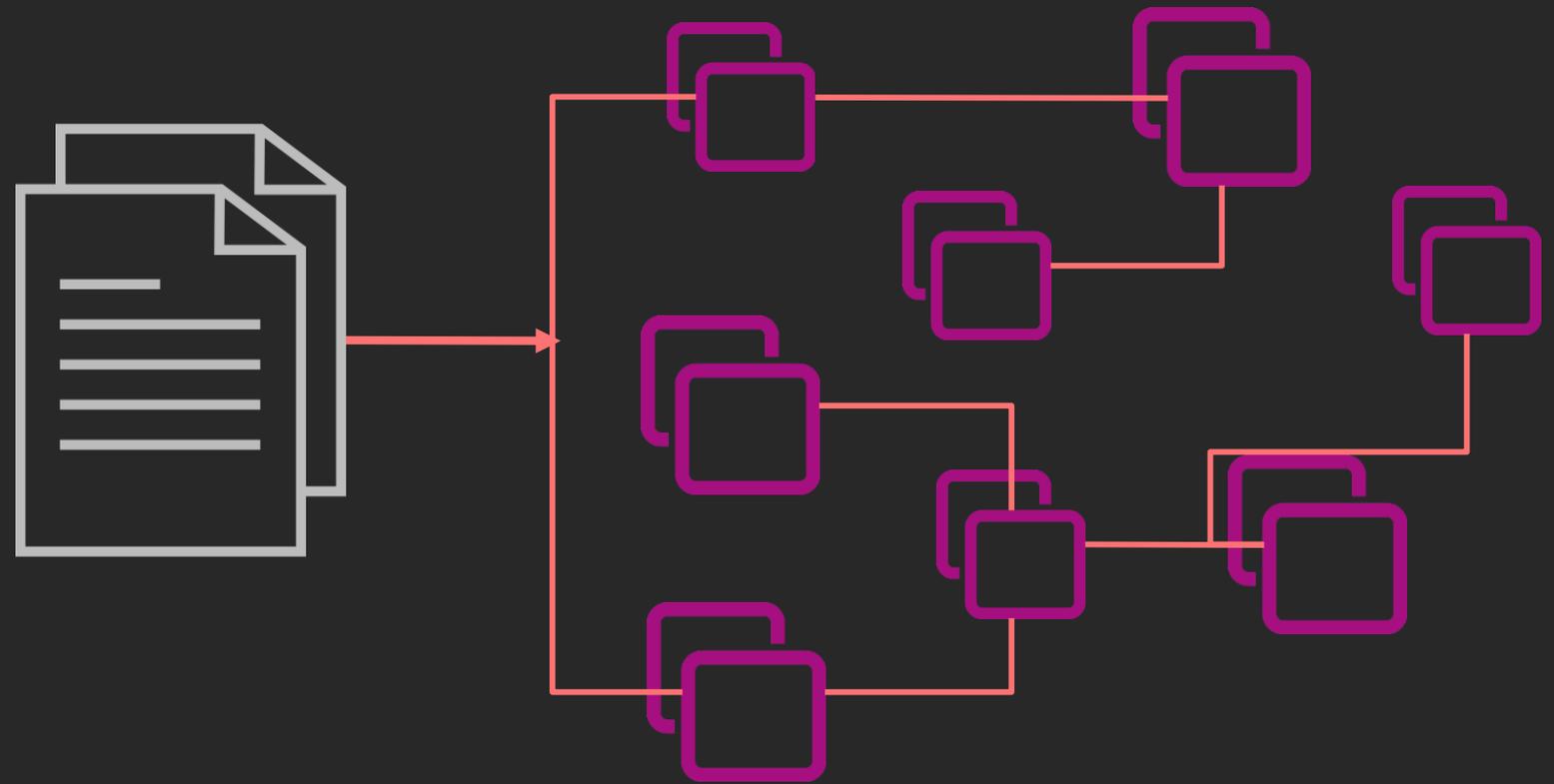
ee2a24f7 Source: attach sns topics for dev mode alarms

# 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

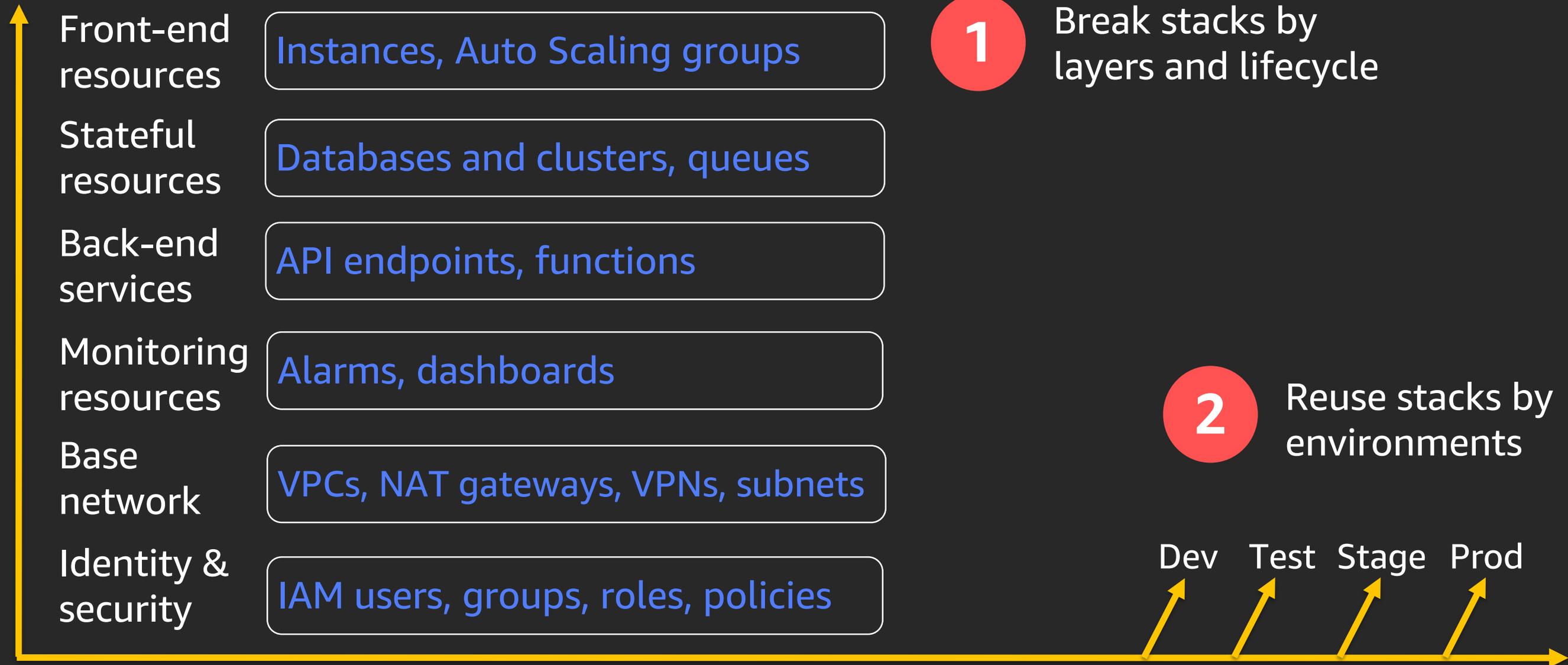
Multiregion production environments

# AWS CloudFormation StackSets best practices

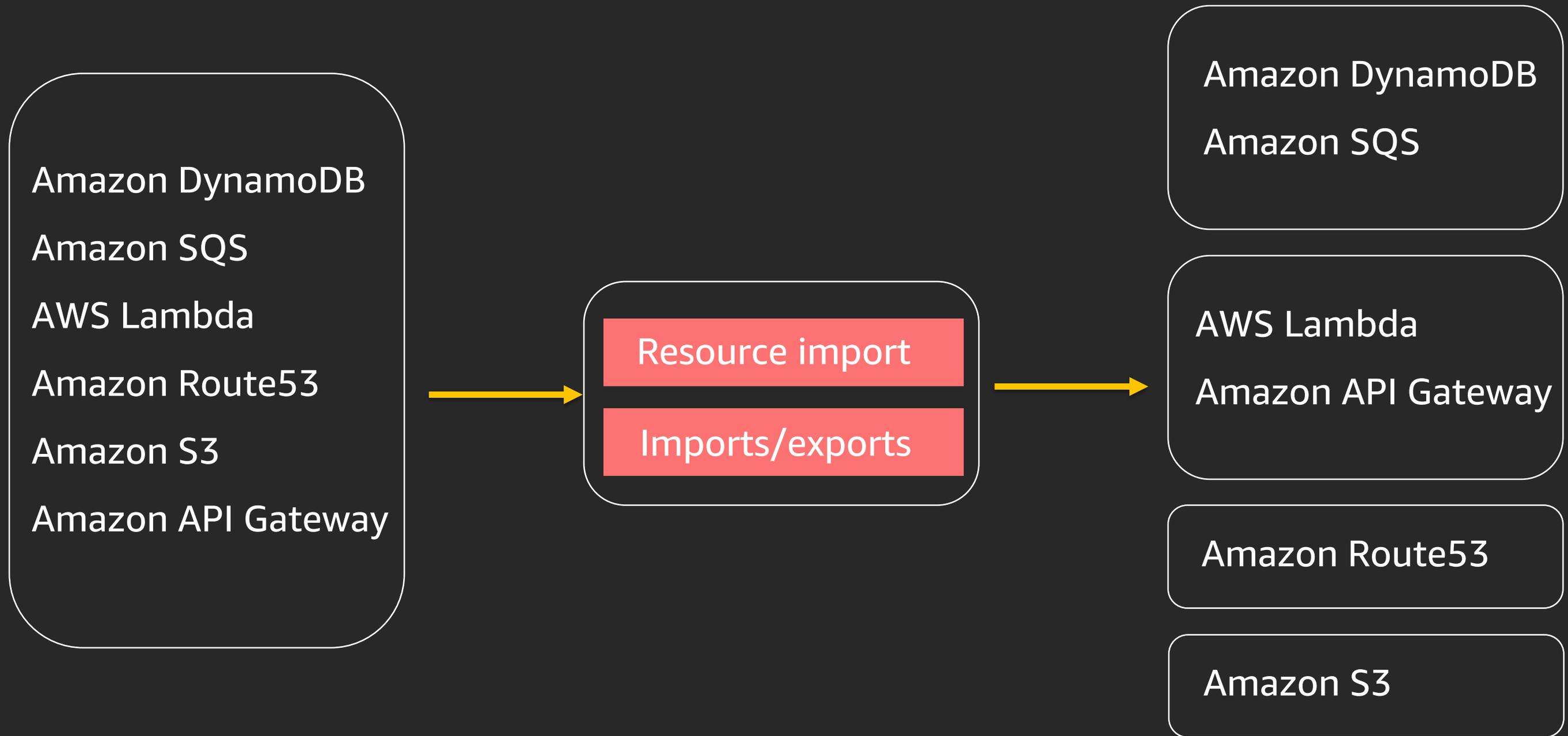
- Partially deploy your AWS CloudFormation StackSet updates to reduce blast radius; great for sanity testing/releasing incrementally
- Depending on your speed needs, consider setting a higher concurrent account limit
- Use parameter overrides to define specific parameters in account–region pairs
- Separate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

# Maintaining your stacks

# Stacks by lifecycle



# Refactoring existing stacks



# Detecting drift

## Expected

```
{
  "AttributeDefinitions": [
    {
      "AttributeName": "id",
      "AttributeType": "S"
    }
  ],
  "KeySchema": [
    {
      "AttributeName": "id",
      "KeyType": "HASH"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 2,
    "WriteCapacityUnits": 2
  }
}
```

## Actual

```
{
  "AttributeDefinitions": [
    {
      "AttributeName": "id",
      "AttributeType": "S"
    }
  ],
  "KeySchema": [
    {
      "AttributeName": "id",
      "KeyType": "HASH"
    }
  ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 15,
    "WriteCapacityUnits": 15
  }
}
```

# Remediating drift



ReadCapacity: 2  
WriteCapacity: 2



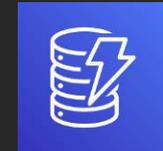
Orphan resource



Import resource



ReadCapacity: 15  
WriteCapacity: 15

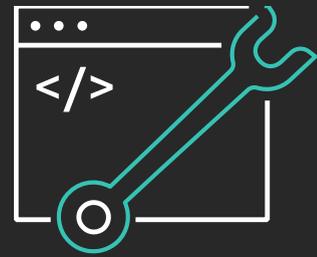


# In summary

- 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/](https://aws.amazon.com/training/path-developing/)

# Thank you!

**Dan Blanco**

@thedanblanco

**Olivier Munn**

olivmunn@amazon.com



Please complete the session survey in the mobile app.