AWS re:Invent

DOP304-R

Building reusable AWS CloudFormation templates

Dan Blanco
Developer Advocate
AWS CloudFormation
Amazon Web Services

Chelsey Salberg
Frontend Engineer
AWS CloudFormation
Amazon Web Services

AWS re:Invent

aws

Agenda

Intro to AWS CloudFormation

Setting up your editor (lab 0)

Template anatomy (lab 1)

Intrinsic functions (lab 2)

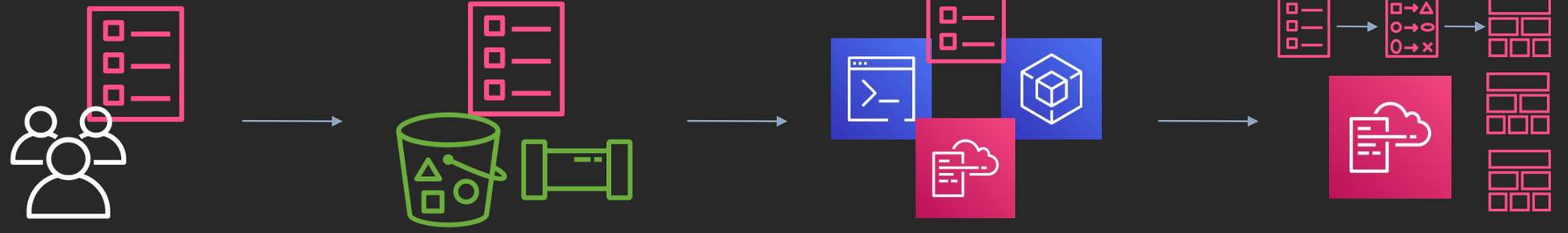
Putting it all together (lab 3)

Intro to AWS CloudFormation

AWS re:Invent

aws

AWS CloudFormation in a nutshell



Code in YAML/JSON directly, use SAM or macros, translate higher 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, AWS CLI, or AWS SDKs, or stack set instances across accounts and regions

Stacks, stack sets, and resources are created and managed

Intro to workshops

AWS re:Invent

aws

Workshop anatomy: Theory & practice

Theory

- Learn from slides
- Review docs

Practice

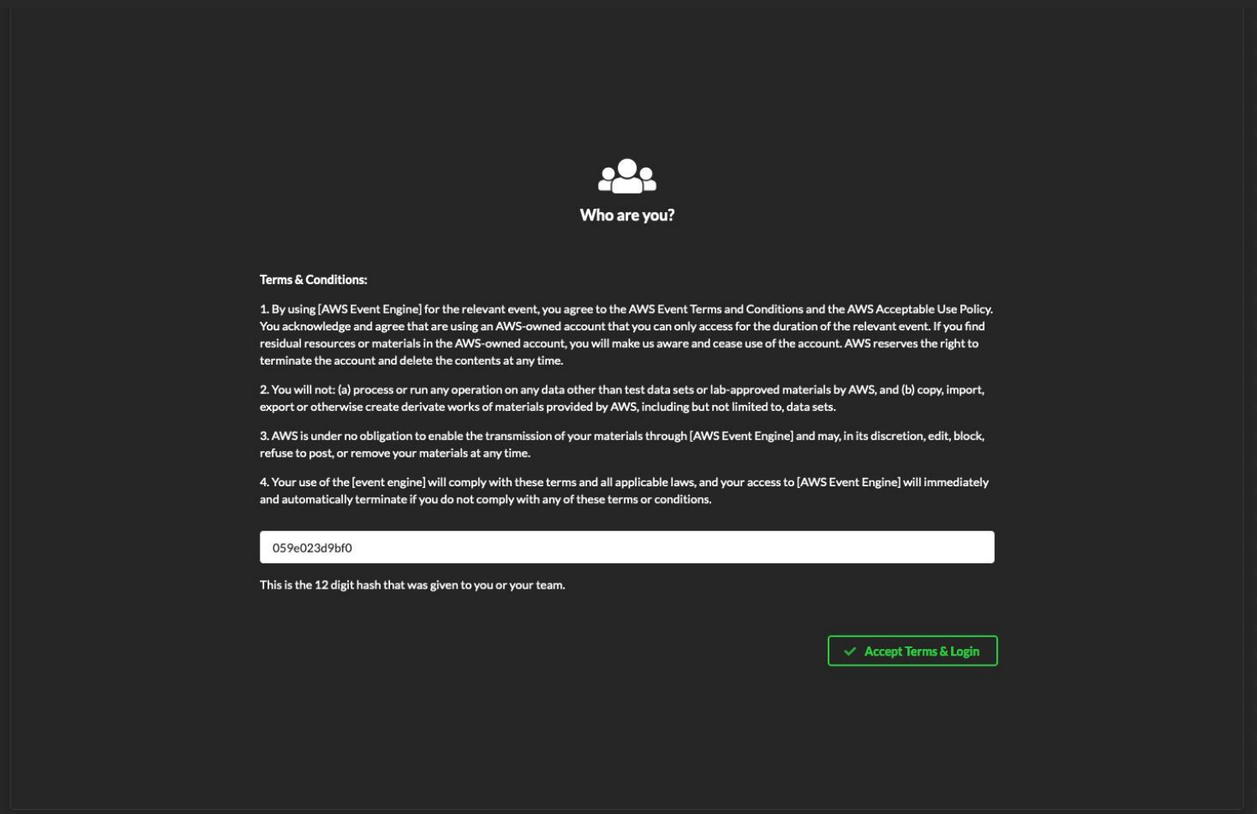
- Learn by doing
- Hands on, writing code

Labs warning: you may not be able to complete the labs in the allotted time, but don’t worry, we will give you the answers and code to follow along, and for you to dive deeper and explore at home

Logging into the workshop

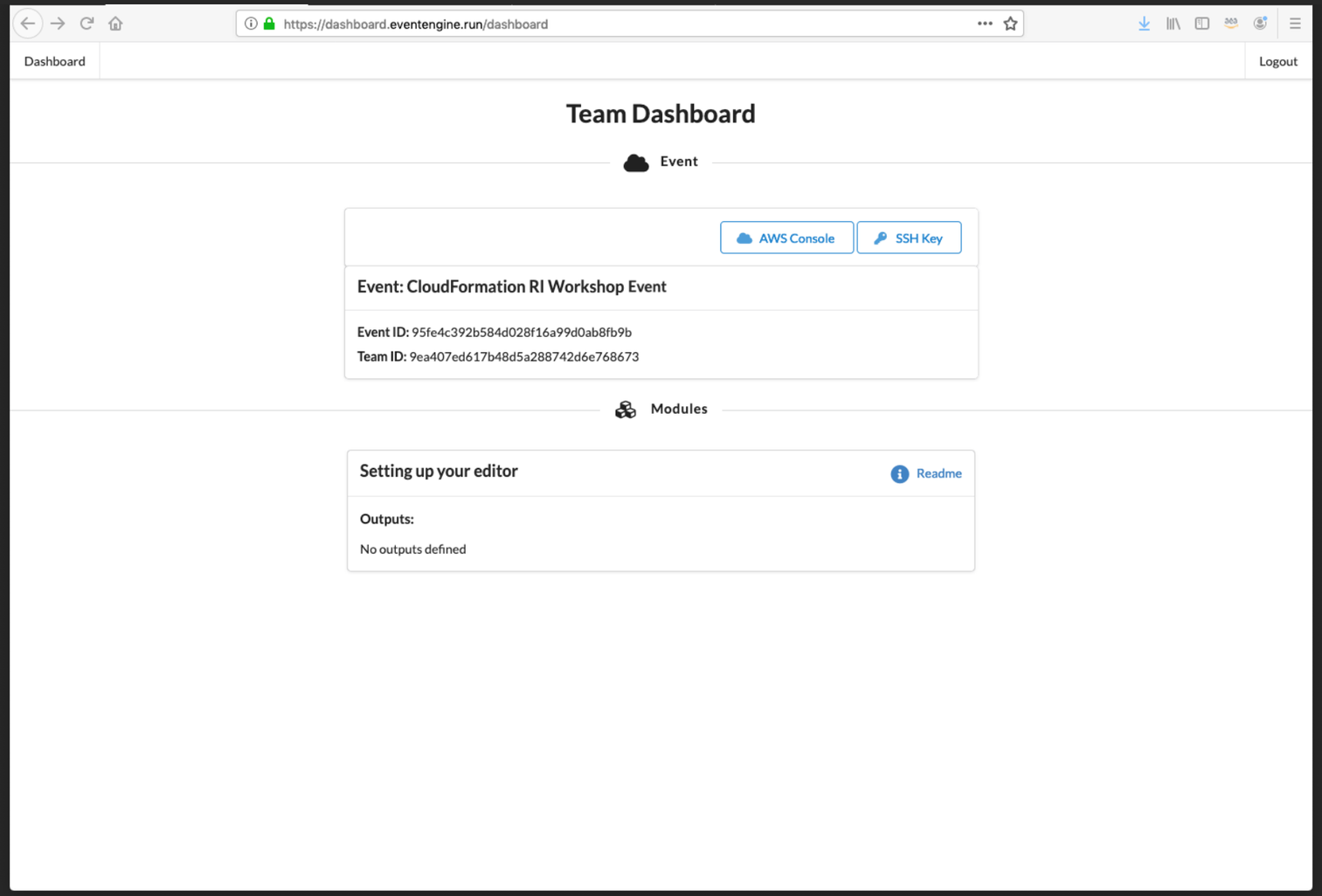
https://dashboard.eventengine.run

Use the hash code provided on your label



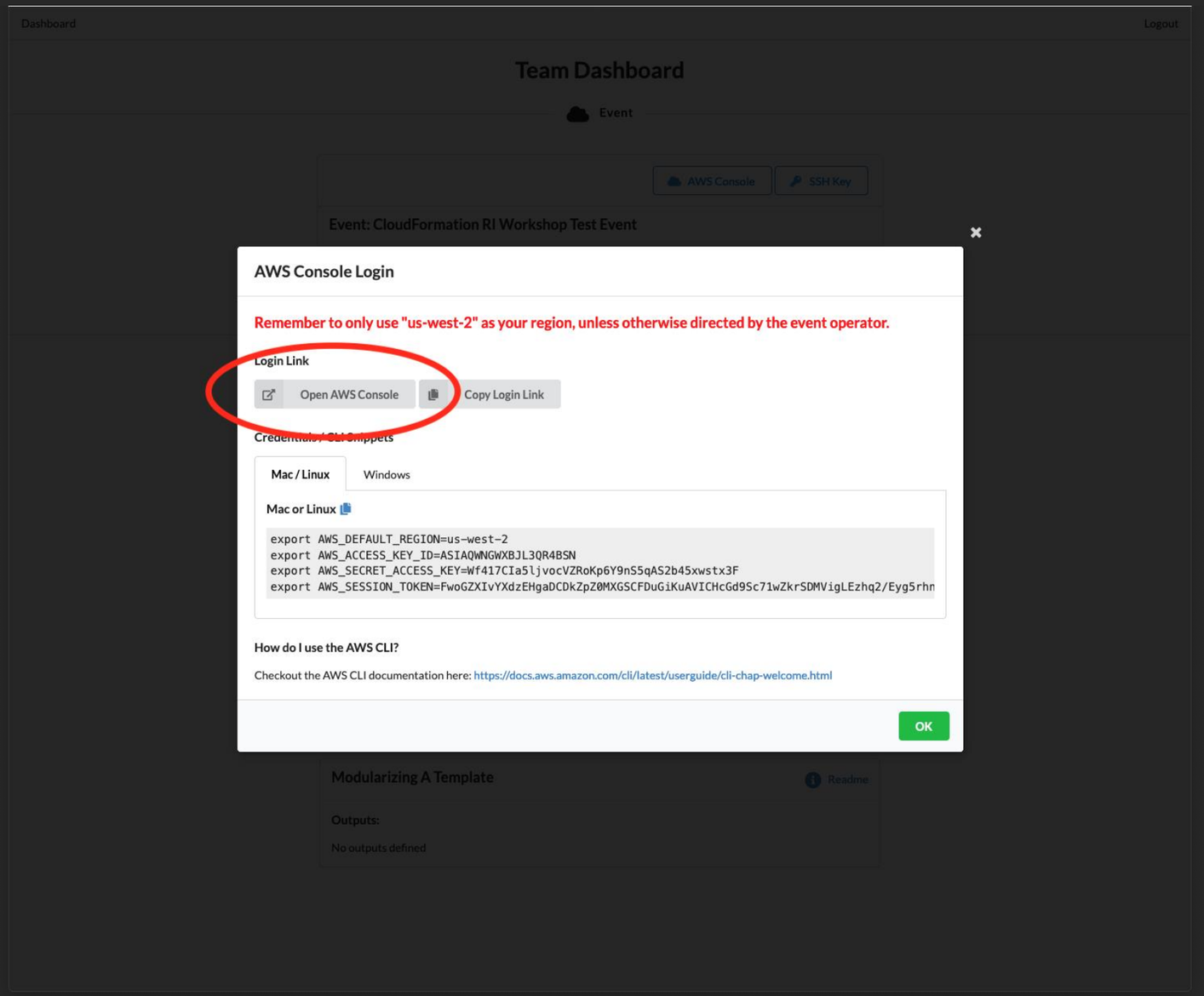
Team dashboard

- Access your AWS account
- Lab instructions
- Sample templates
- Answers to previous labs



Log into the console

- Click on “AWS Console”
- Click on “Open AWS Console”



Setting up your editor

AWS Cloud9

- Code with just a browser
- Start new projects quickly
- Collaborate with other developers in real time
- Enables developers to build serverless applications with ease
- Direct terminal access to AWS services
- Easily configurable via an AWS CloudFormation template

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, VisualStudio Code, Sublime, VIM
- Process multiple files
- Handles Conditions/Fn::If
- SAM Local integration
- Available now on GitHub, over 3,000,000 downloads

Customizing your linter

- Require specific tags
- Blacklist of resource types
  (i.e., can’t create X resource type)
- Enforce/require a property
- Forbid a property value
  (i.e., don’t let people create public buckets)
- More!

https://binx.io/2018/07/07/aws-cloudformation-validation-in-cicd-pipelines/

Picking another editor

- VS Code
- Sublime Text
- Atom
- IntelliJ
- PyCharm
- More!

Lab 0: Setting up your editing environment (20m)

Objectives:

- Deploy AWS Cloud9 Environment
- Install cfn-lint via terminal

Advanced:

- Install VSCode locally, with the vscode-cfn-lint
- Install cfn-lint locally via terminal

Review:

- Environment configured for CFN authoring

Template anatomy

Template sections

```
AWSTemplateFormatVersion: "2010-09-09"
Description:
  A text string that describes the template
Metadata:
  Template metadata provides additional information about the template
Parameters:
  Set of values passed to your template at runtime
Mappings:
  Set of key value mappings, similar to a lookup table
Conditions:
  Set of conditions that control resources during stack creation or update
Transform:
  Set of transforms, including for serverless applications
Resources:
  Set of resources and their properites (required)
Outputs:
  Set values that are returned whenever you view your stack's properties
```

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html

```yaml
Mappings:
  MyMapping:
    Key01:
      Name: Value01
    Key02:
      Name: Value02
Parameters:
  InstanceType:
    Description: EC2 instance type
    Type: String
    Default: t2.small
    AllowedValues:
      - t2.small
      - t2.medium
      - t2.large
    ConstraintDescription: Must be a valid EC2 instance type.
Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: t2.small
  MyWaitHandle1:
    Type: 'AWS::CloudFormation::WaitConditionHandle'
Outputs:
  MyOutput:
    Description: My Output
    Value: Value01
```

Parameters and resources

```yaml
Parameters:
  InstanceType:
    Description: EC2 instance type
    Type: String
    Default: t2.small
    AllowedValues:
      - t2.small
      - t2.medium
      - t2.large
    ConstraintDescription: Must be a valid EC2 instance type.
Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    InstanceType: t2.small
  MyWaitHandle1:
    Type: 'AWS::CloudFormation::WaitConditionHandle'
```

Parameters:
Logical ID:
Type: *Data type*
ParameterProperty: value

Resources:
Logical ID:
Type: *Resource type*
Properties:
Set of properties

Parameters

- Set of values passed to your template at runtime
- Types
  - String
  - Number
  - List<Number>
  - CommaDelimtedList
  - AWS-specific parameter types
  - SSM parameters types

Resources

- Set of resources and their properties (required)
- Properties vary by resource type (refer to documentation)

Mappings and outputs

```
1   Mappings:
2     MyMapping:
3       Key01:
4         Name: Value01
5       Key02:
6         Name: Value02
7   Parameters: …
17  Resources: …
24  Outputs:
25    MyOutput:
26      Description: My Output
27      Value: Value01
```

Mappings:
  Map ID:
    Key ID :
      Value: value

Outputs:
  Logical ID:
    Description: *Value information*
    Value: *Value to return*
    Export:
      Name: *Value to export*

Mappings

- Set of key value mappings, similar to a lookup table
- RegionMap is a popular mapping to allow for easy customization per region

Outputs

- Set values that are returned whenever you view your stack's properties
- Can declare exports that can be referenced cross-stack

Lab 1: Template anatomy (20m)

Objectives:

- Use parameters for better reusability
- Use mappings for better cross-region usability
- Add outputs for using resources between templates

Advanced:

- Group/label/constraints on parameters

Review:

- CFN templates are easily sharable/configurable

Intrinsic functions & pseudo parameters

Intrinsic functions

- Built in functions not part of JSON/YAML
- Assign values that may not be available until runtime
- Create your own via macros

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-macros.html

Intrinsic functions—common

- !Ref – returns a specific value of a resource *or* parameter

```
InstanceId: !Ref MyEc2InstanceIdParameter
Database: !Ref GlueDataCrawler
```

- !GetAtt – returns a value from a resource in the templates

```
Role: !GetAtt GlueIAMRole.Arn
```

- !FindInMap – finds a value in a map by using a key

```
LogLevel: !FindInMap [Variables, !Ref Environment, LogLevel]
```

- !Join – joins two values using a delimiter

```
Accounts: !Join [“,”, !Ref AccountOne, !Ref AccountTwo]
```

- !Sub – Replaces a value in ${} with its value. Works like !Ref and !GetAtt

```
BucketPolicy: !Sub “${BucketName}/*”
```

Intrinsic functions—Sub vs. Join

```
content: !Join
- ''
- - 'CREATE DATABASE '
- !Ref 'DBName'
- ";\n"
- CREATE USER '
- !Ref 'DBUser'
- '''@''localhost'' IDENTIFIED BY '''
- !Ref 'DBPassword'
- "';\n"
- 'GRANT ALL ON '
- !Ref 'DBName'
- .* TO '
- !Ref 'DBUser'
- "'@'localhost';\n"
- "FLUSH PRIVILEGES;\n"
```

```
!Sub |-
CREATE DATABASE ${DBName};
CREATE USER '${DBUser}'@'localhost'
IDENTIFIED BY '${DBPassword}';
GRANT ALL ON ${DBName}.* to
'${DBUser}'@'localhost';
FLUSH PRIVILEGES;
```

Intrinsic functions — conditions

- !Equals – value on the left is exactly the same as the right

```
IsProduction: !Equals [“Prod”, !Ref EnvironmentParam]
```

- !If – if condition is true, use middle value, otherwise end value

```
SecurityGroups: !If [CreateNewSecurityGroup, !Ref NewSecurityGroup,
!Ref ExistingSecurityGroup]
```

- !And – returns true if ALL conditions are true

```
Alarm: !And [ !Equals [“Prod”, !Ref Environment ], ShouldAlarm ]
```

- !Or – returns true if ANY conditions are true

```
Alarm: !Or [ !Equals [“Prod”, !Ref Environment ], ShouldAlarm ]
```

- !Not – returns true if value is false

```
IsDev: !Not [ !Equals [“Prod”, !Ref Environment ] ]
```

Pseudo parameters

- Parameters that exist across all templates
- Automatically managed by AWS CloudFormation
- Referenced just like regular parameters

```
!Sub arn:aws:dynamodb:${AWS::Region}:${AWS::AccountId}:table/${TableName}
```

Pseudo parameters—explained

- AWS::AccountId – current ID of account [11112222333]
- AWS::NotificationArns – list of notification ARNs. Use !Select for single
- AWS::NoValue – an “empty” value. Useful for “else” part of !If
- AWS::Partition – “partition” section of ARN. [arn:aws:iam…]
- AWS::Region – current region [us-east-1]
- AWS::StackId – full stack ID (ARN) [arn:aws:us-west-1:cloudformation:111122223333:stack/test-stack/51af3dc0-da77-11e4-872e-1234567db123]
- AWS::StackName– just the name portion of the stack ID [test-stack]
- AWS::UrlSuffix – suffix for current domain. Usually amazonaws.com

Lab 2: Intrinsics (20m)

Objectives:

- Use !Refs & !GetAtts for referencing resources
- Convert hard-coded parameters to use pseudo params

Advanced:

- Convert !Joins to !Subs

Review:

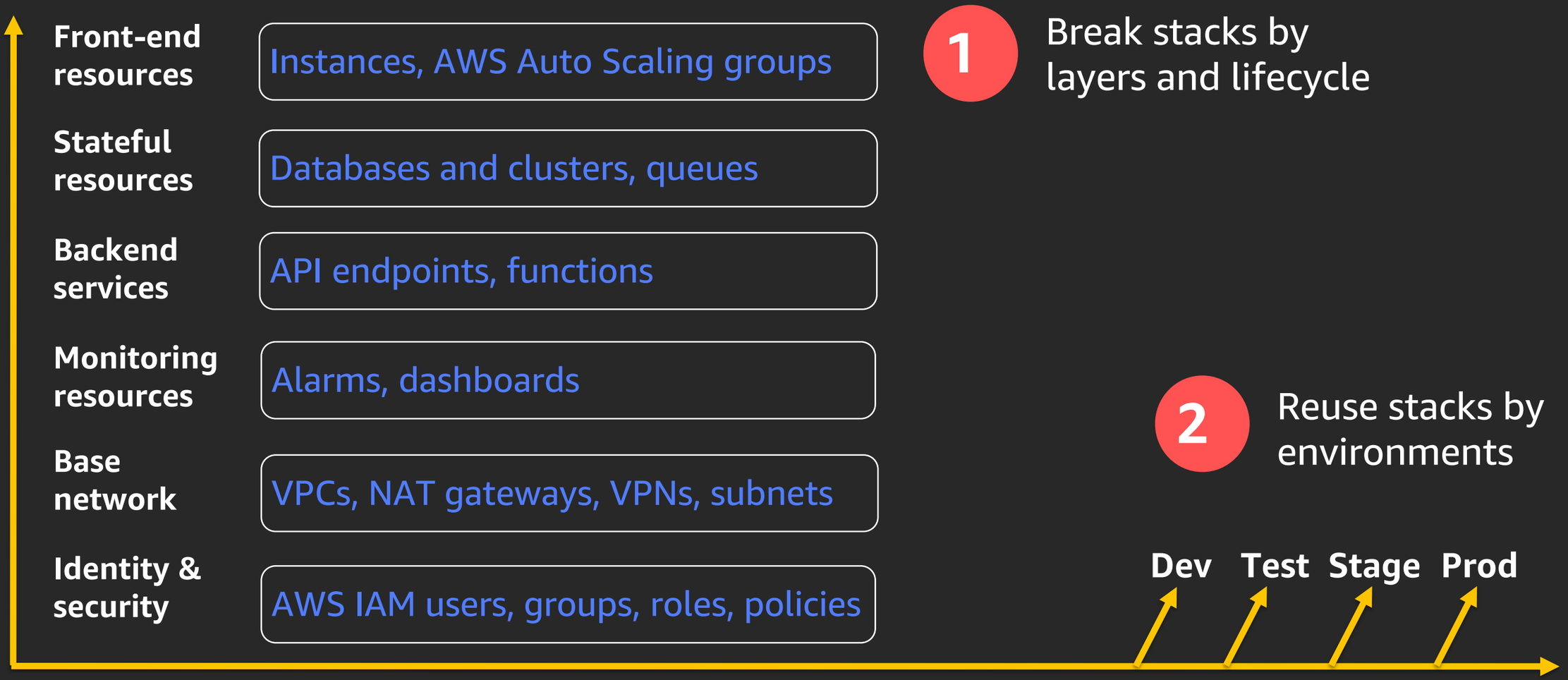
- CFN templates are using intrinsics & pseudo params to simplify authoring

Putting it all together

Recap

- Editor
  - VSCode, Sublime Text, Atom, AWS Cloud9, etc.
  - Plugins
- Template anatomy
  - Resources
  - Parameters
  - Outputs
- Intrinsic functions & pseudo params
  - !Sub vs !Join
  - Conditions, !FindInMap, pseudo params
  - !Refs & !GetAtts

Stacks by lifecycle



Referencing resources across stacks

```
Resources:
    DataLakeBucket:
        Type: AWS::S3::Bucket
Outputs:
    DataLakeBucketName:
        Value: !Ref DataLakeBucket
        Export:
            Name: S3DataLakeBucket
```

```
Resources:
    GlueDataCrawler:
        Type: AWS::Glue::Crawler
        Properties:
            Role: !GetAtt GlueIAMRole.Arn
            Targets:
                - !ImportValue S3DataLakeBucket
```

Names must be unique!

Lab 3: Bringing it all together (30m)

Objectives:

- Split stacks by lifecycle
- Reference cross-stack resources via !ImportValue

Review:

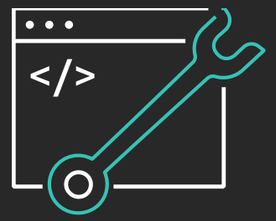
- Multiple CFN templates with intrinsics, parameters, and cross-stack references

In summary

- Leverage a good editor. AWS Cloud9 works great, local editors are preferred.
- Know all the parts of the template anatomy. Resources, parameters, and outputs are usually the most important.
- Know and love your intrinsic functions. Not enough? Look into macros to create your own.
- Pseudo parameters save you a lot of hard-coding
- Keep your stacks small and modular. Reference using imports/exports or SSM parameters.

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
dblanc@amazon.com

Chelsey Salberg

@ChelseySalberg
csalberg@amazon.com

Please complete the session survey in the mobile app.