Deploying Applications to AWS

Ryan Lewis
CLOUD ENGINEER
@ryanmurakami ryanlewis.dev

Previously on AWS Developer...

Coding and Clicking
Deployments

Overview

CloudFormation forms the clouds

How high can you stack?

Template to stack in one step

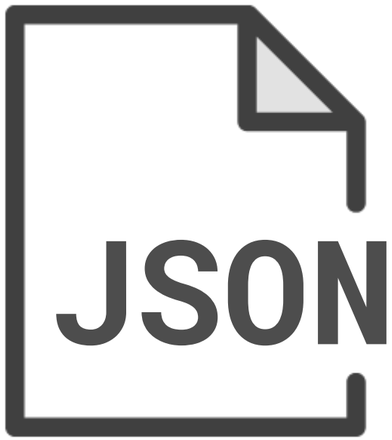
Stack ‘em up, Knock ‘em down

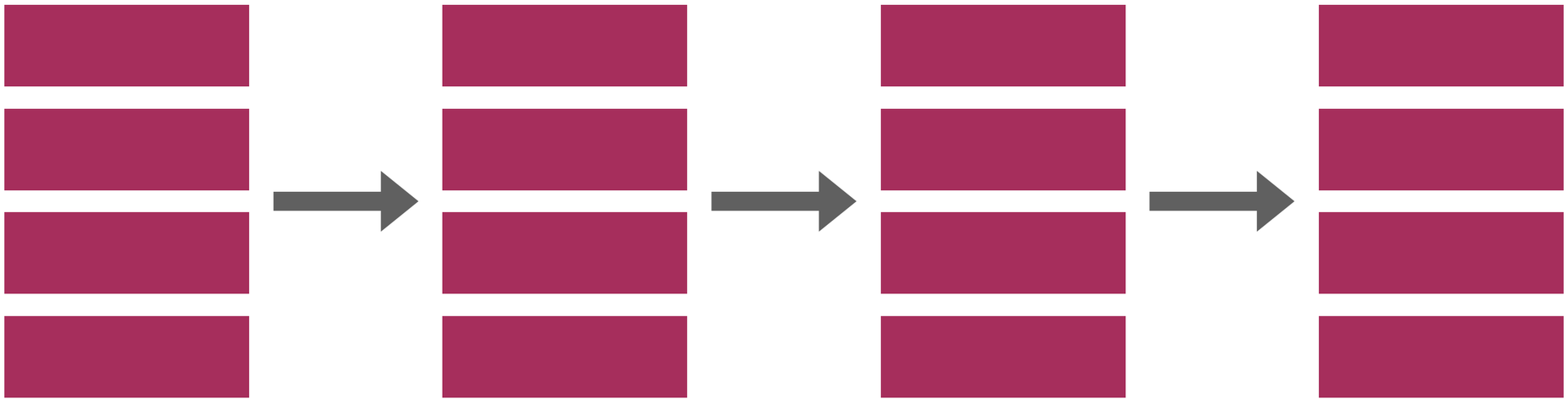
Just one more CLI

Deploying to the Beanstalk

Understanding CloudFormation Stacks

Choose Your Weapon

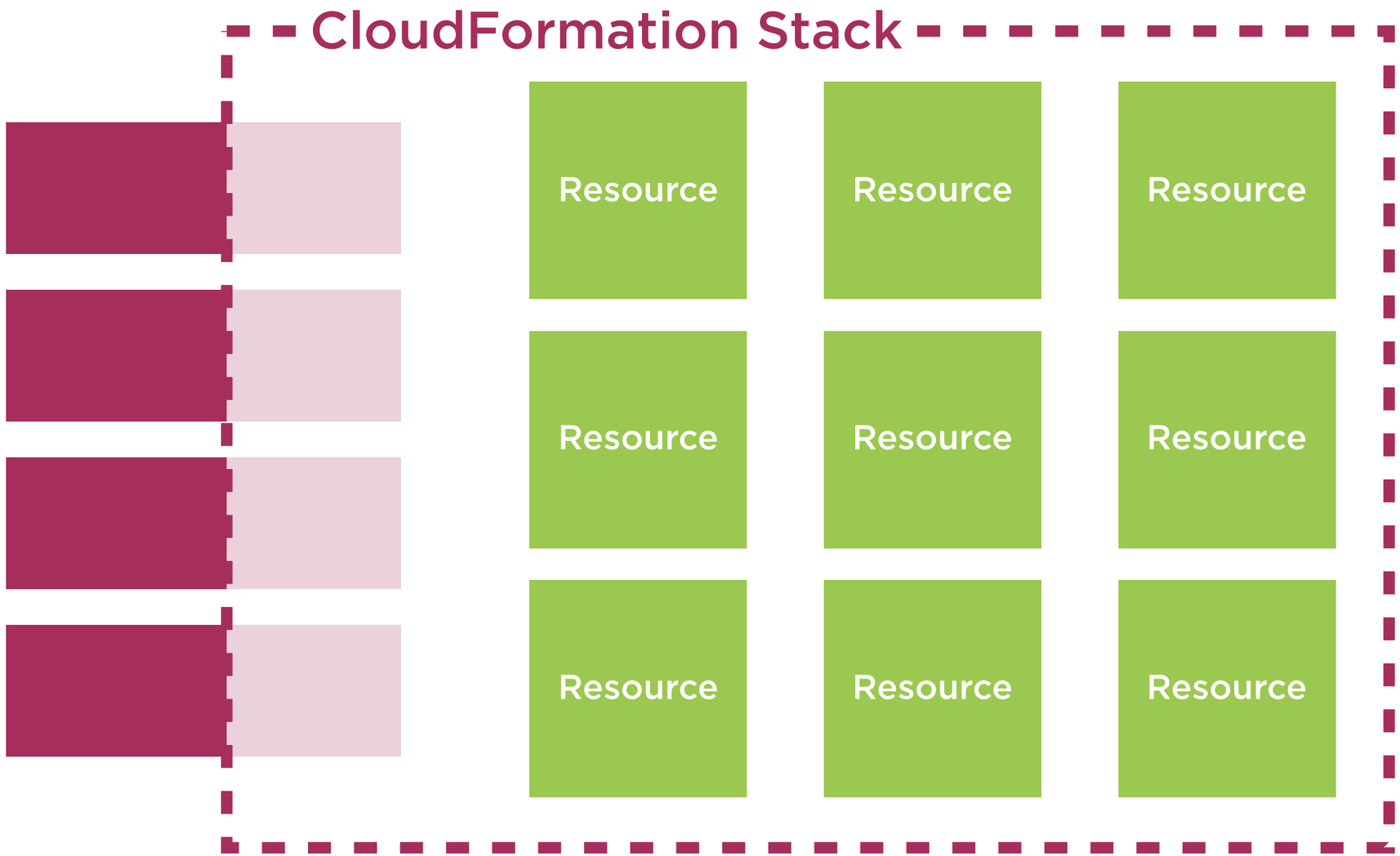




Each CloudFormation template can have a maximum of 100 resources

1.
Template
Executed

2.
Stack
Created


CloudFormation Stack
Resource
Resource
Resource
Resource
Resource
Resource
Resource
Resource
Resource

One command for the entire stack

Create Update Delete

CloudFormation runs with the permissions of the executing user

If one resource creation or update fails, the entire CloudFormation operation fails

Pro-tip

When developing, don’t wait for rollbacks to complete, just use a new stack name.

Infrastructure as Code

Source control

Less human error

Simple resource re-creation

Third-party CloudFormation Alternatives

My Take

Choose CloudFormation over third-party tools.

Why Choose CloudFormation?

You need to understand CloudFormation anyways to be successful with AWS

Almost complete feature parity with third-party tools

AWS continues to update CloudFormation to support new services

Anatomy of a CloudFormation Template

CloudFormation templates
got you down?

Don’t worry, you’ll get used
to them!

Using CloudFormation Functions

CloudFormation Functions

```
Fn::Base64
Fn::FindInMap
Fn::GetAtt
Fn::GetAZs
Fn::ImportValue
Fn::Join
Fn::Select
Fn::Split
Fn::Sub
```

```
Fn::And
Fn::Equals
Fn::If
Fn::Not
Fn::Or
```

GetAtt

Retrieve an attribute from a created resource

Syntax

```
{ "Fn::GetAtt": ["ResourceName", "AttributeName"] }
```

Join

Combines a set of values with a delimiter

Syntax

```
{ "Fn::Join": ["Delimiter", ["Value1", "Value2"]] }
```

Sub

Replace a token in a string value

Syntax

```
{ "Fn::Sub": ["String${Token}", {"Token": "Value"}] }
```

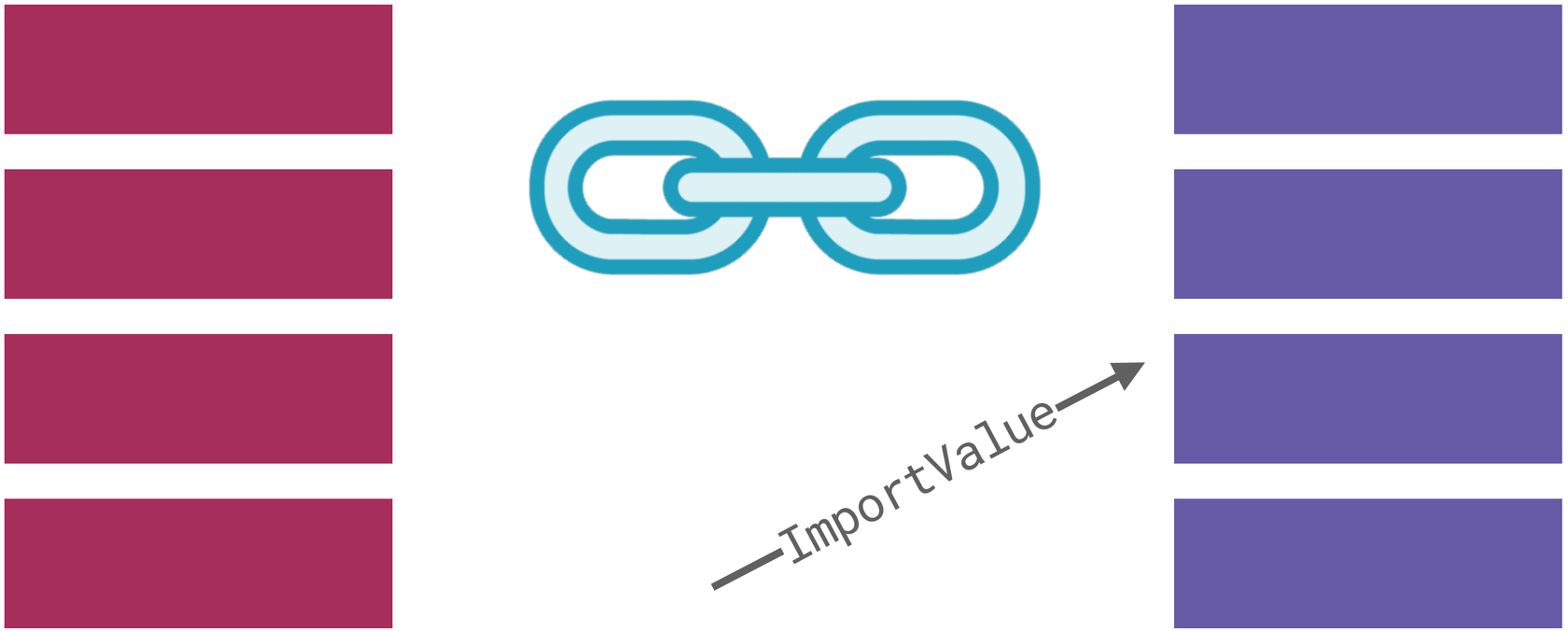
ImportValue

Import an export from another stack

Syntax

```
{ "Fn::ImportValue": "ExportName" }
```

ImportValue Example



Building CloudFormation Templates

Deploying CloudFormation Templates

CloudFormation can create almost anything, except Key Pairs

Updating and Deleting CloudFormation Stacks

`aws cloudformation deploy`

Can be used for creates and updates

Uses previous parameters for updates

create/update

changeset
created

changeset
executed

Installing the Elastic Beanstalk CLI

AWS CLI vs. EB CLI

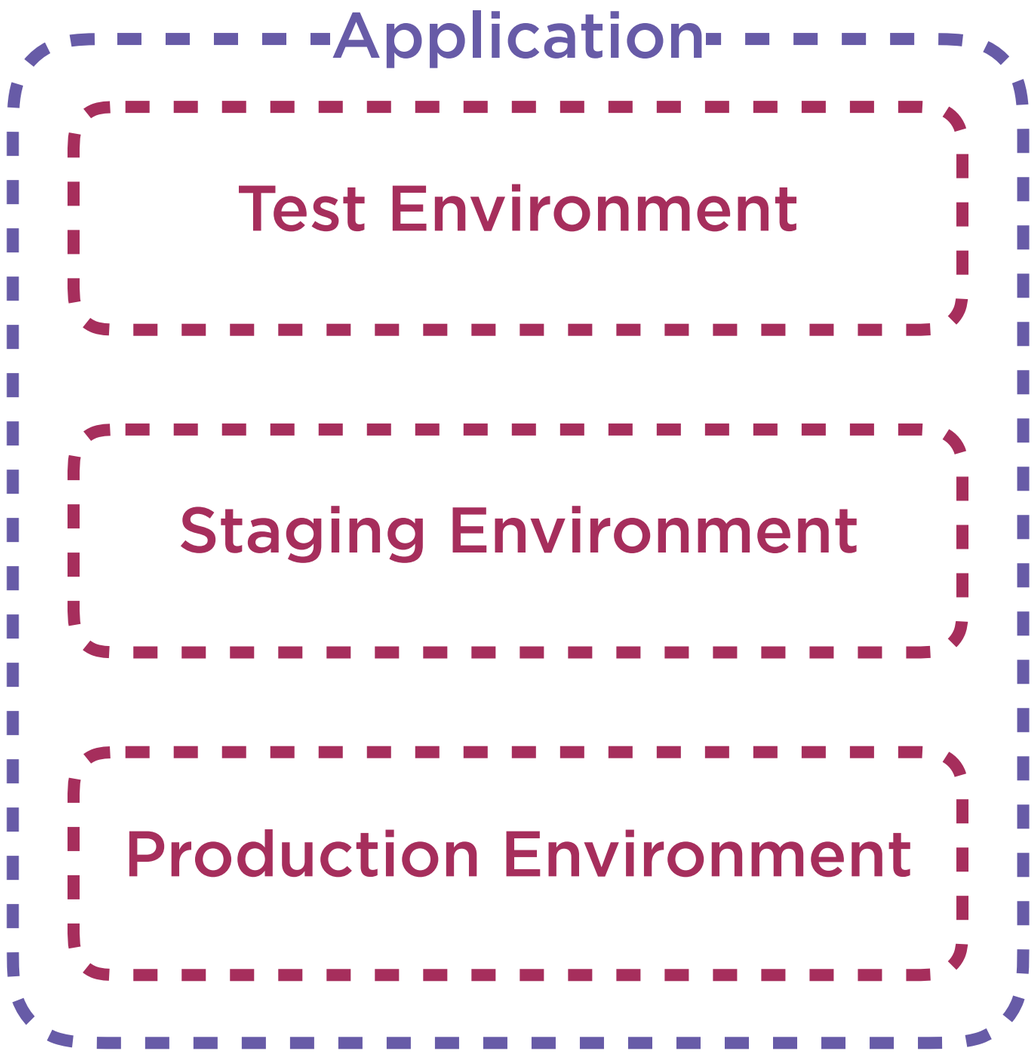
`aws elasticbeanstalk create-environment…`	`eb create…`
`aws elasticbeanstalk update-environment…`	`eb deploy…`
`aws elasticbeanstalk terminate-environment…`	`eb terminate…`

EB CLI Commands

```
eb abort
eb appversion
eb clone
eb codesource
eb config
eb console
eb create
eb deploy
eb events
eb health
eb init
eb labs
eb list
eb local
eb logs
eb open
eb platform
eb printenv
eb restore
eb scale
eb setenv
eb ssh
eb status
eb swap
eb tags
eb terminate
eb upgrade
eb use
```

Deploying Applications with Elastic Beanstalk

Elastic Beanstalk Anatomy



EB CLI Commands

```
eb abort
eb appversion
eb clone
eb codesource
eb config
eb console
eb create
eb deploy
eb events
eb health
eb init
eb labs
eb list
eb local
eb logs
eb open
eb platform
eb printenv
eb restore
eb scale
eb setenv
eb ssh
eb status
eb swap
eb tags
eb terminate
eb upgrade
eb use
```

Delete the **`node_modules`** folder before deploying to Elastic Beanstalk

Use **`eb deploy`** to re-deploy your application quickly

Conclusion

Summary

The building blocks of CloudFormation

Who needs all that JSON?

Stack full of hamsters

Saying goodbye to a good stack

A better CLI for Beanstalk

Smoothest deployment ever

Up Next

Service Coordination

with Simple Workflow