Best Practices with Managing CloudFormation Stacks

Ryan Lewis
CLOUD ARCHITECT

@ryanmurakami   ryanlewis.dev

Overview

Templated stack control

Keep your stacks from drifting away

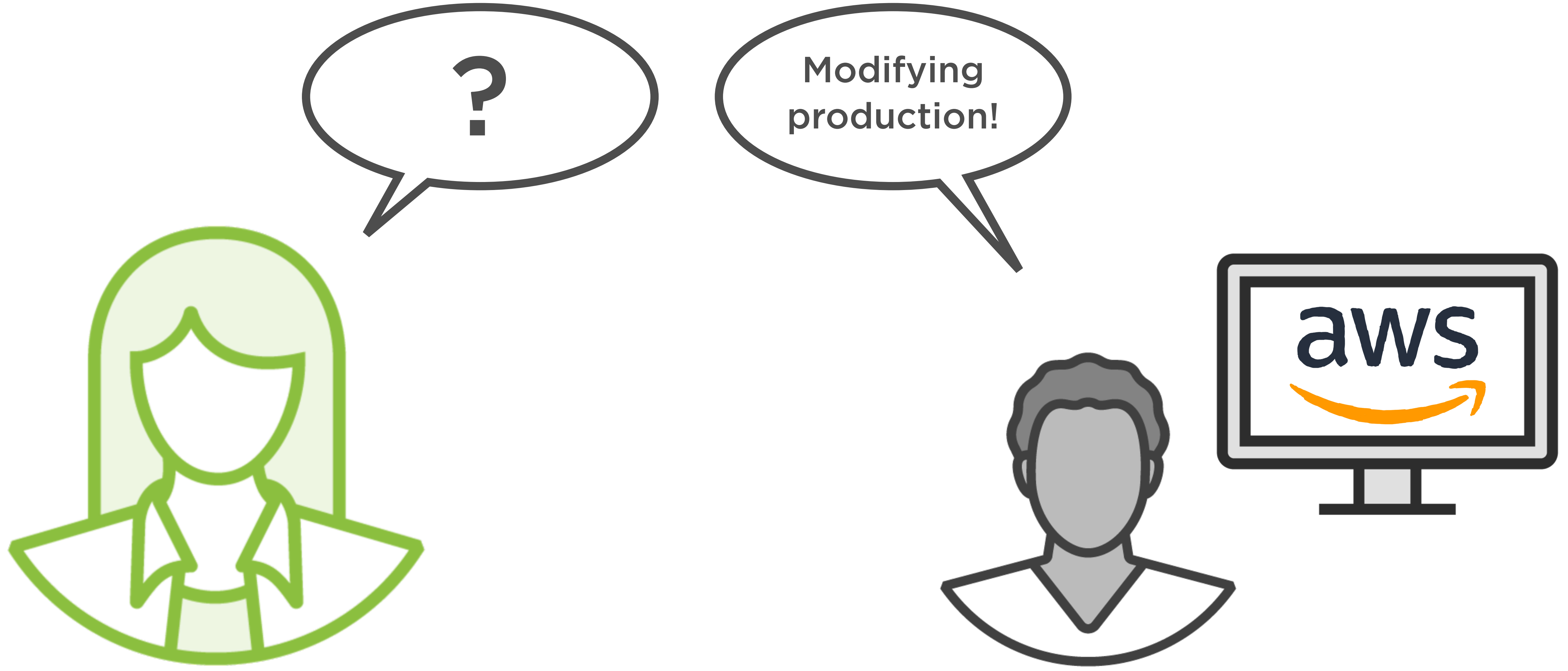
Stack policy protection

A change set path to updates

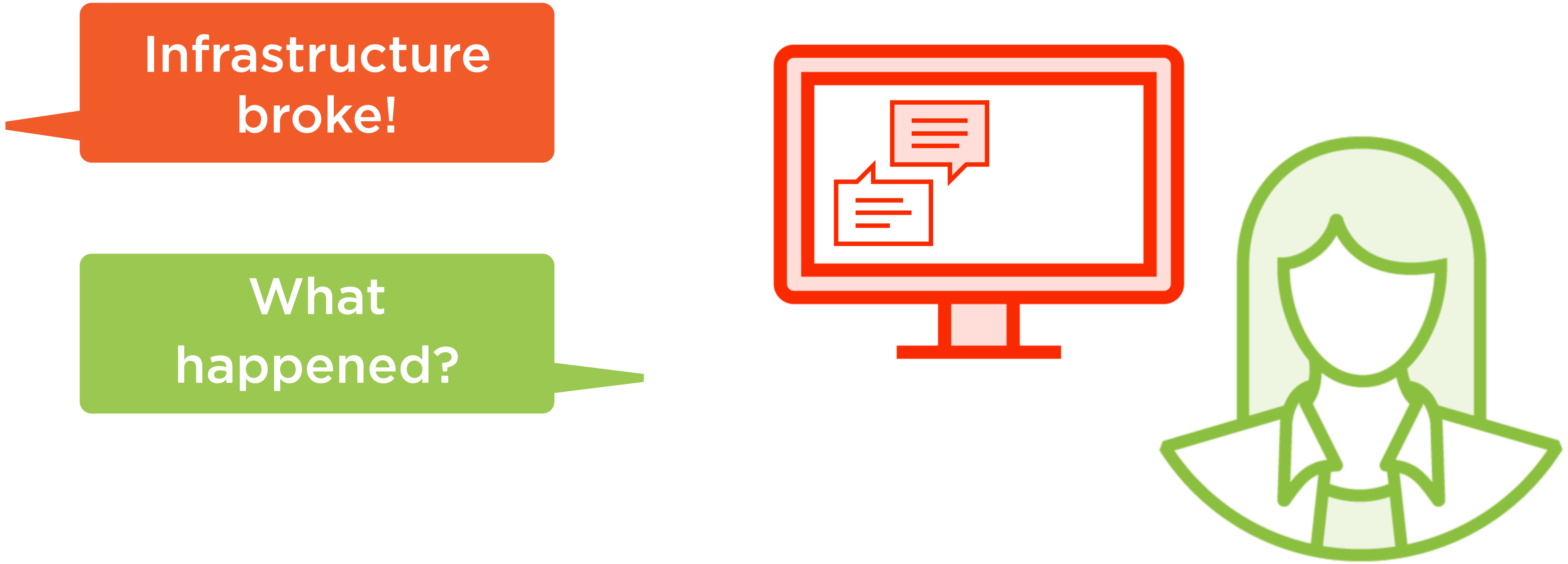
CloudTrail automated surveillance

CloudFormation Best Practices at HBFL

A Normal Tea Break at HBFL



Hard at Work at HBFL



The Steps to Broken Infrastructure

- ✓ **Created a parameters file for new region**
- ✓ **Executed template with parameters**
- ⊘ **Entered manual changes into template**

Fixing Mistakes at HBFL

Got to go, bye!

Always keep your infrastructure in code

CloudFormation Best Practice

Use Code Reviews and Revision Controls to Manage Your Templates

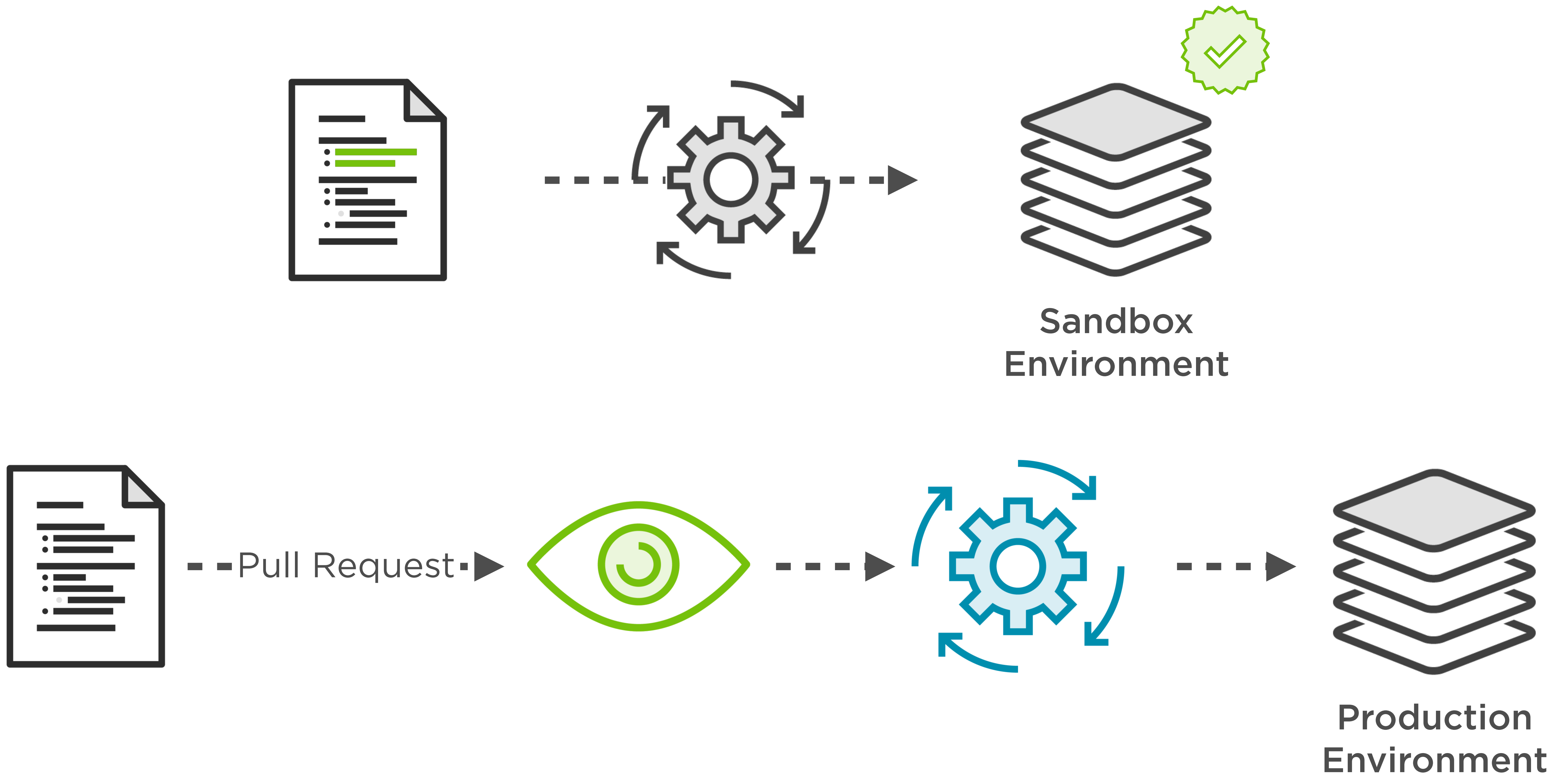
Keeping Your Templates in Source Control

Makes you keep your infrastructure as code

Infrastructure history that provides easy lookup and reversion

Peer review process can catch errors in infrastructure configuration

My Code Review Process



Templates in Source Control at HBFL

Detecting Drift in a CloudFormation Stack

Can You Tell What Has Changed?

CloudFormation Best Practice

Manage All Stack Resources through AWS CloudFormation

Drift

When the resource configuration of a CloudFormation stack has changed through means other than CloudFormation.

Drift Can Cause Major Problems

Changes made can be erased by CloudFormation updates

CloudFormation updates or deletes can fail

Drift Detection

Since 2018!

Drift Detection

Detects changes to a stack outside of CloudFormation

Available in the AWS console

Can use AWS CLI or SDK for automation

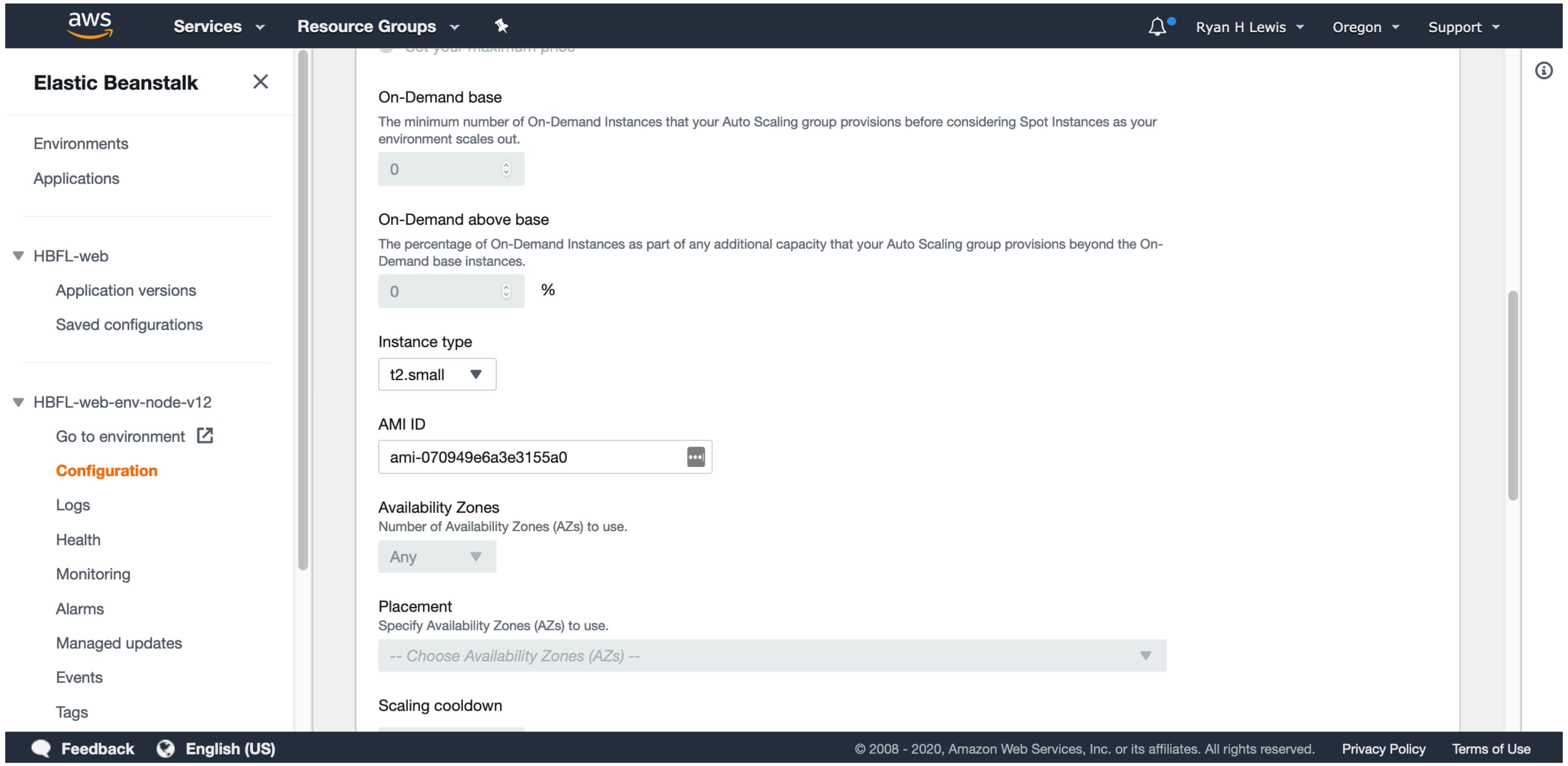
CloudFormation drift detection does not support all resource types

AWS Services Supported by Drift Detection

API Gateway	DynamoDB	IAM	Route 53
Auto Scaling	EC2	IOT	S3
CloudTrail	ECS	Lambda	SNS
CloudWatch	Elastic Load Balancing	RDS	SQS

16 out of 150+ services in AWS!

Modifying Elastic Beanstalk Configuration



On-Demand base

The minimum number of On-Demand Instances that your Auto Scaling group provisions before considering Spot Instances as your environment scales out.

0

On-Demand above base

The percentage of On-Demand Instances as part of any additional capacity that your Auto Scaling group provisions beyond the On-Demand base instances.

0 %

Instance type

t2.small

AMI ID

ami-070949e6a3e3155a0

Availability Zones

Number of Availability Zones (AZs) to use.

Any

Placement

Specify Availability Zones (AZs) to use.

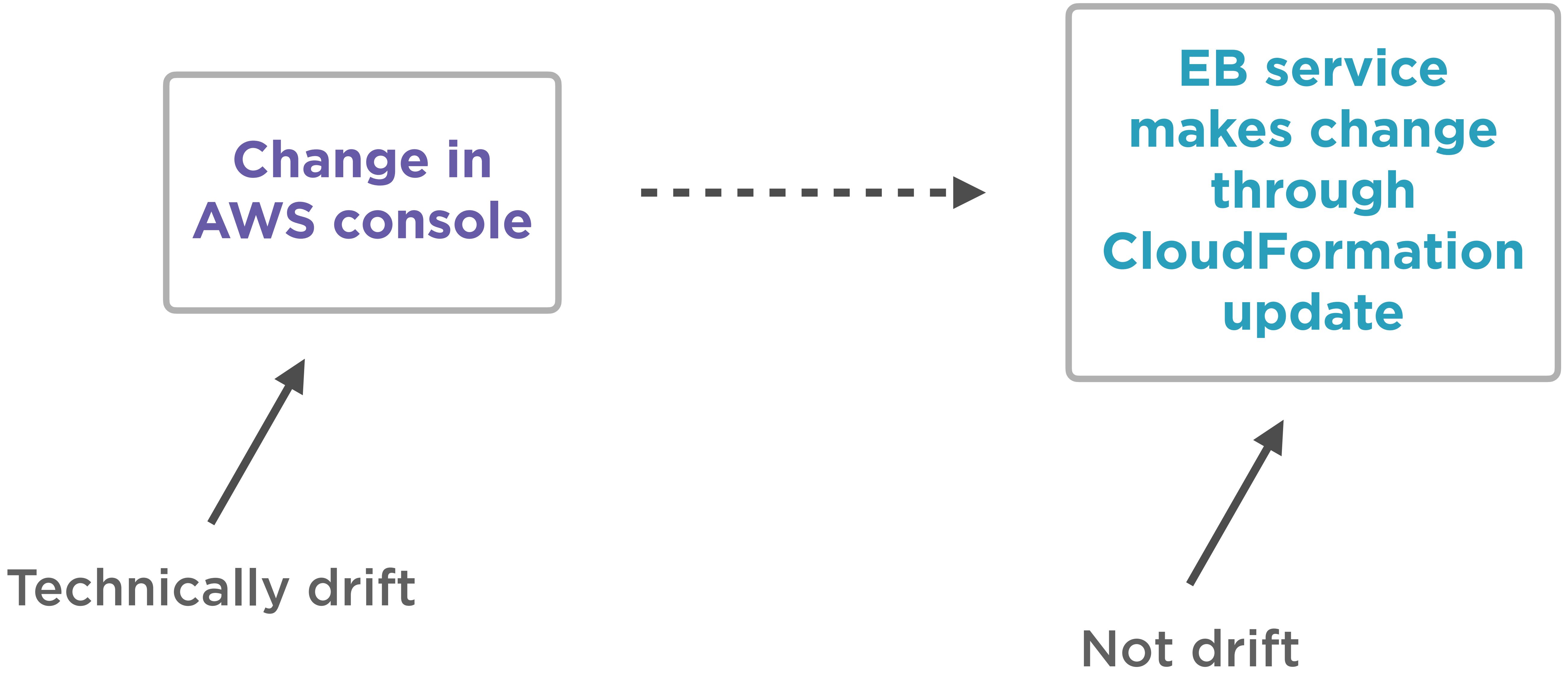
-- Choose Availability Zones (AZs) --

Scaling cooldown

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

t2.small

An Elastic Beanstalk Change in Motion



Demo

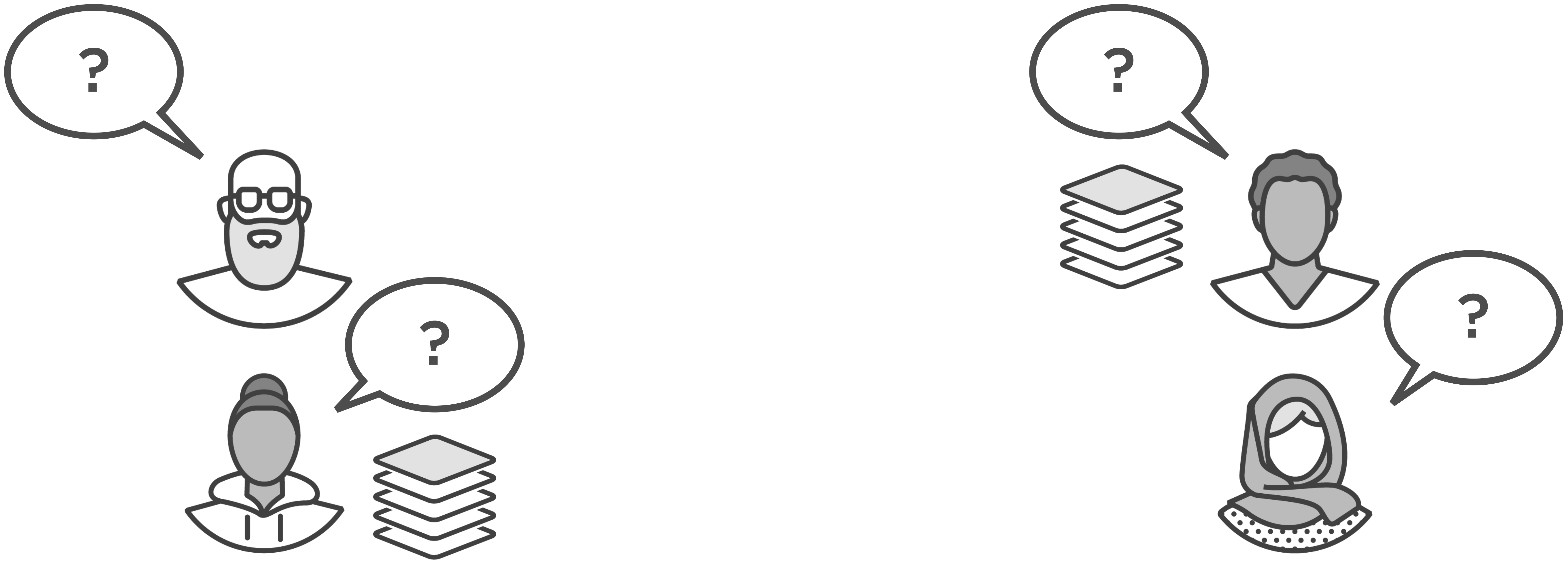
Drift detection in action

CloudFormation Best Practice

Manage All Stack Resources through AWS CloudFormation

Using Stack Policies to Protect Resources

Stack Ownership at HBFL



Updating Infrastructure at HBFL

- ✓ Changes in templates
- ✓ Added parameters
- ✓ Went through source control
- ? Deployment?

CIDR block changes on a VPC through CloudFormation require a replacement of the VPC and all resources inside aws CloudFormation Best Practice

Use Stack Policies

Stack Policy

A policy attached to a CloudFormation stack that defines which types of updates are allowed on resources.

Just like IAM policies, stack policies deny all actions implicitly

Stack Policy Actions

Update:Modify	Update:Replace	Update:Delete
No interruption or minimal interruption	Resources are recreated from scratch	Resource is completely removed

Update:* All of the above

Stack Policies Will Not Protect You From

Stack deletion

** Use stack termination protection instead

Manual changes outside of CloudFormation

** Infrastructure changes should use CloudFormation

Introducing change sets...

Generating Change Sets for CloudFormation Stack Updates

Ping Pong at HBFL

Responsible CloudFormation Updating

Creating stack policies + Consulting documentation + Handling failed updates

=

A whole lotta time

CloudFormation Best Practice

Create Change Sets Before Updating Your Stacks

Change Set

JSON document that shows the changes a CloudFormation update will perform.

Change Set Change Example

my-change-set.json

```
"resourceChange": {
  "logicalResourceId": "ExampleEC2Instance",
  "action": "Modify",
  "resourceType": "AWS::EC2::Instance",
  "replacement": "True",
  "details": [
    {
      "target": {
        "name": "InstanceType",
        "requiresRecreation": "Conditionally",
         "attribute": "Properties"
      },
      "changeSource": "DirectModification"
    }
```

Why Use Change Sets

Prevents inadvertent updates

Shows you intended changes

Better than updating blind

CloudFormation Best Practice

Create Change Sets Before Updating Your Stacks

Logging CloudFormation Changes with AWS CloudTrail

Ultimate CloudFormationing at HBFL

Investigating a Disappearance at HBFL

CloudFormation Best Practice

Use AWS CloudTrail to Log AWS CloudFormation Calls

AWS CloudTrail

Monitoring service for activity in your AWS account.

Why CloudTrail Is Great with CloudFormation

Captures all calls to CloudFormation API

No configuration

Tracking for most recent 90 days is free!

CloudFormation Monitoring at HBFL

Until Next Time

CloudFormation Template Best Practices

Use IAM to Control Access

Verify Quotas for All Resource Types

Organize Your Stacks by Lifecycle and Ownership

Reuse Templates to Replicate Stacks in Multiple Environments

Use Nested Stacks to Reuse Common Template Patterns

Use Cross-stack References to Export Shared Resources

CloudFormation Stack Best Practices

Use Code Reviews and Revision Controls to Manage Your Templates

Manage All Stack Resources through AWS CloudFormation

Use Stack Policies

Create Change Sets Before Updating Your Stacks

Use AWS CloudTrail to Log AWS CloudFormation Calls

Thank you!

Ryan Lewis
CLOUD ARCHITECT
@ryanmurakami ryanlewis.dev