

# Infrastructure as Code

*Software Architecture*

Brae Webb & Richard Thomas

March 17, 2025

*Infrastructure as Code*

How did we get here?

*Pre-2000*

The *Iron Age*

## *Iron Age*



## *Iron Age*



## *Scaling*



*Introducing...*

The *Cloud Age*

## *The Cloud Age*





*When faced with complexity*

Automate it!

# The larger story

Server Config Config Management

# The larger story

Server Config Config Management

Application Config Config Files

# The larger story

Server Config   Config Management

Application Config   Config Files

Provisioning   Infrastructure Code

# The larger story

Server Config   Config Management

Application Config   Config Files

Provisioning   Infrastructure Code

Building   Continuous Integration

# The larger story

Server Config   Config Management

Application Config   Config Files

Provisioning   Infrastructure Code

Building   Continuous Integration

Deployment   Continuous Deployment

# The larger story

Server Config   Config Management

Application Config   Config Files

Provisioning   Infrastructure Code

Building   Continuous Integration

Deployment   Continuous Deployment

Testing   Automated Tests

# The larger story

Server Config Config Management

Application Config Config Files

Provisioning Infrastructure Code

Building Continuous Integration

Deployment Continuous Deployment

Testing Automated Tests

Database Administration Schema Migration



# The larger story

Server Config   Config Management

Application Config   Config Files

Provisioning   Infrastructure Code

Building   Continuous Integration

Deployment   Continuous Deployment

Testing   Automated Tests

Database Administration   Schema Migration

Specifications   Behaviour Driven Development

*Definition 0.* Infrastructure Code

Code that provisions and manages *infrastructure resources*.

*Definition 0. Infrastructure Code*

Code that provisions and manages *infrastructure resources*.

*Definition 0. Infrastructure Resources*

Compute resources, networking resources, and storage resources.

## *Infrastructure Code*



# Shell Scripts

```
1  #!/bin/bash
3  SG=$(aws ec2 create-security-group ...)
5  aws ec2 authorize-security-group-ingress --group-id "$SG"
7  INST=$(aws ec2 run-instances --security-group-ids "$SG" \
8      --instance-type t2.micro)
```

# Python

```
1  import boto3

3  def create_instance():
4      ec2_client = boto3.client("ec2", region_name="us-east-1")
5      response = ec2.create_security_group(...)
6      security_group_id = response['GroupId']

8      data = ec2.authorize_security_group_ingress(...)

10     instance = ec2_client.run_instances(
11         SecurityGroups=[security_group_id],
12         InstanceType="t2.micro",
13         ...
14     )
```

# Terraform

```
1 resource "aws_instance" "hextris-server" {
2     instance_type = "t2.micro"
3     security_groups = [aws_security_group.hextris-server.name]
4     ...
5 }

7 resource "aws_security_group" "hextris-server" {
8     ingress {
9         from_port = 80
10        to_port = 80
11        ...
12    }
13    ...
14 }
```

*Question*

Notice anything different?



*The main difference*

Imperative vs. Declarative

## *Declarative IaC*

- Define your *desired* infrastructure state
  - as code
- Engine interprets difference between the *desired* and *actual* state
  - Modifying infrastructure to deliver *desired* state

## *Infrastructure Code*

- Provisions and manages *infrastructure resources*.

## *Infrastructure Code*

- Provisions and manages *infrastructure resources*.
- Only one part of the movement to *automate* the complexities of development.

## *Infrastructure Code*

- Provisions and manages *infrastructure resources*.
- Only one part of the movement to *automate* the complexities of development.
- Ranges from simple shell scripts up to...?

## *Infrastructure Code*

- Provisions and manages *infrastructure resources*.
- Only one part of the movement to *automate* the complexities of development.
- Ranges from simple shell scripts up to...?
- Tendency to be *declarative*.

*Typo?*

Infrastructure Code  $\neq$  Infrastructure *as* Code

*Definition 0.* Infrastructure as Code

Following the same *good coding practices* to manage Infrastructure Code as standard code.



*Warning!*

Infrastructure as Code still *early* and quite *bad*.

*Question*

What are *good coding practices*?

*Good Coding Practice #1*

*Everything* as Code

```
1  #!/bin/bash
3  ./download-dependencies
4  ./build-resources
5  cp -r output/* artifacts/
```

```
1  #!/bin/bash
3  ./download-dependencies
4  ./build-resources
5  cp -r output/* artifacts/
```

```
$ cp: directory artifacts does not exist
```

```
1 resource "aws_instance" "hextris-server" {  
2     instance_type = "t2.micro"  
3     security_groups = ["sg-6400"]  
4     ...  
5 }
```

```
1 resource "aws_instance" "hextris-server" {
2     instance_type = "t2.micro"
3     security_groups = [aws_security_group.hextris-server.name]
4     ...
5 }

7 resource "aws_security_group" "hextris-server" {
8     ingress {
9         from_port = 80
10        to_port = 80
11        ...
12    }
13    ...
14 }
```

*Everything as code avoids*  
Configuration drift



*Configuration drift creates*

Snowflakes

## *Benefits*

1. Reproducible

*Good Coding Practice #2*

## Version Control

## *Benefits*

1. Restorable
2. Accountable

*Good Coding Practice #3*

Automation

## *Benefits*

1. Consistent

*Good Coding Practice #4*

Code Reuse

## *Benefits*

1. Better<sup>1</sup> code
2. Less work
3. Only one place to update (or verify)

---

<sup>1</sup>generally



*Good Coding Practice #5*

Testing

# Test Pyramid



# IaC Test Pyramid



```
1 func TestTerraformAwsInstance(t *testing.T) {
2     terraformOptions := terraform.WithDefault(t, &terraform.Options{
3         TerraformDir: "../week03/",
4     })
5
6     defer terraform.Destroy(t, terraformOptions)
7     terraform.InitAndApply(t, terraformOptions)
8
9     publicIp := terraform.Output(t, terraformOptions, "public_ip")
10    url := fmt.Sprintf("http://%s:8080", publicIp)
11
12    http_helper.HttpGetWithCustomValidation(t, url, nil, 200,
13        func(code, resp) { code == 200 &&
14            strings.Contains(resp, "hextris")})
15 }
```

```
1  Feature: Define AWS Security Groups

3  Scenario: Only selected ports should be publicly open
4      Given I have AWS Security Group defined
5      When it contains ingress
6      Then it must only have tcp protocol and port 22,443 for 0.0.0.0/0
```

## *Benefits*

### 1. Trust

### *Prac Next Week*

Learn how to use Terraform to write IaC and deploy resources on AWS.