The Security Engineer’s Guide to Infrastructure-As-Code!

Dakota Riley - Senior Security Engineer, Aquia Inc

Introduction

- Dakota Riley
- Senior Security Engineer @ Aquia Inc
- NKU COI class of 2017

Aquia

A Service-Disabled Veteran-Owned Small Business

Cloud Security

- DevSecOps
- Compliance Automation
- Infrastructure-as-Code
- Architecture & Tooling
- Detection & Response

Advisory Practice

- Security Assessments
- Risk Management
- Strategy & Roadmapping
- vCISO Services
- Customized Training

Cyber Security

- Application Security
- Compliance-as-Code
- Incident Response
- Automated Forensics
- Penetration Testing

Security Engineering/Architectural Professional

Development teams are beginning to adopt Infrastructure-As-Code (IaC)

Tasked with securing the implementation and usage of IaC at the company

What is IaC?

What is needed from a Security Perspective?

Should we (Security) be using it?

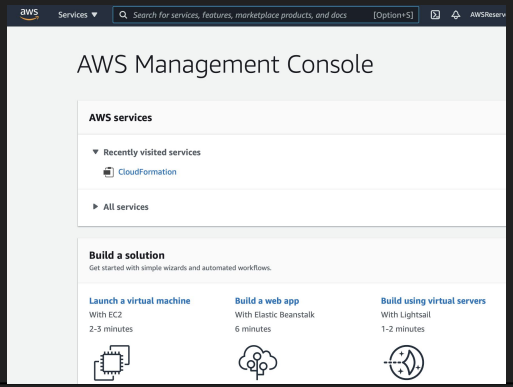
What is IaC?

Tooling that allows you to express the configuration of systems, services or infrastructure via code/templates and apply them automatically

How We Configure Things

Manual Configuration

- Great for learning
- Break Glass Access
- Effort/Time tradeoff



Scripting/SDKs/APIs

- Better than Manual
- Imperative
- Edge cases

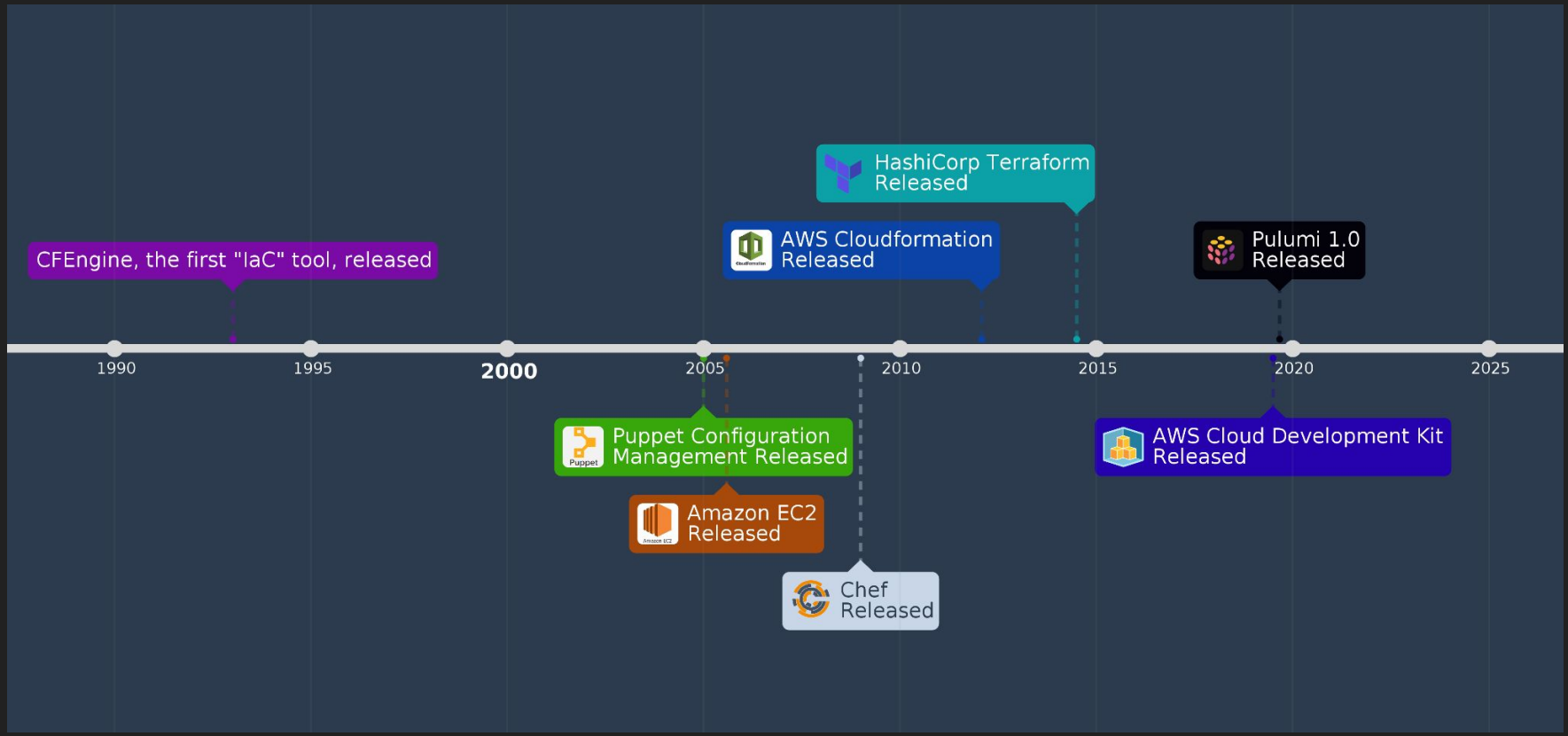
```python
import boto3


def lambda_handler(event, context):

    # Get list of regions
    ec2_client = boto3.client('ec2')
    regions = [region['RegionName']
               for region in ec2_client.describe_regions()['Regions']]

    # Iterate over each region
    for region in regions:
        ec2 = boto3.resource('ec2', region_name=region)

        print("Region:", region)

        # Get only running instances
        instances = ec2.instances.filter(
            Filters=[{'Name': 'instance-state-name',
                      'Values': ['running']}])

        # Stop the instances
        for instance in instances:
            instance.stop()
            print('Stopped instance: ', instance.id)
```

(Modern) Infrastructure-As-Code

- Dependency-Aware
- Declarative (in most cases)
- All the benefits of code!

```hcl
locals {
  is_t_instance_type = replace(var.instance_type, "/^t(2|3|3a)\\..*$/", "1") == "1" ? true : false
}

resource "aws_instance" "this" {
  count = var.create && !var.create_spot_instance ? 1 : 0

  ami                         = var.ami
  instance_type               = var.instance_type
  cpu_core_count              = var.cpu_core_count
  cpu_threads_per_core        = var.cpu_threads_per_core
  user_data                   = var.user_data
  user_data_base64            = var.user_data_base64
  hibernation                 = var.hibernation

  availability_zone           = var.availability_zone
  subnet_id                   = var.subnet_id
  vpc_security_group_ids      = var.vpc_security_group_ids

  key_name                    = var.key_name
  monitoring                  = var.monitoring
  get_password_data           = var.get_password_data
  iam_instance_profile        = var.iam_instance_profile
```

How we got here



Modern IaC Tooling

- Self-Hosted vs SaaS/Managed
- Configuration Languages
  - Data Serialization Languages/Domain Specific Languages
    - JSON
    - YAML
    - HCL
  - Turing Complete Languages
    - Python
    - TypeScript
- Platform Specific vs Vendor Agnostic

Modern IaC Tooling (Con't)

Tool	Vendor	Language	Platform	Custom Resource Types?
AWS Cloudformation	AWS	JSON, YAML	AWS, Agnostic with the addition of Resource Providers	Yes
Terraform	HashiCorp	HCL, JSON	Agnostic	Yes
Pulumi	Pulumi	TypeScript, JS, Python, Go	Agnostic, Major CSPs	No
Cloud Development Kit	AWS	TypeScript, JS, Python, Go	AWS, Terraform, K8s	Yes

Modern IaC Tooling (Con't)

Tool	Vendor	Language	Platform	Custom Resource Types?
Azure Resource Management	Microsoft	JSON	Azure	No
Google Cloud Deployment Manager	Google	Jinja Templates, Python	Google Cloud Platform	No
Kubernetes YAML	Cloud Native Computing Foundation	YAML	Kubernetes	Yes
Dockerfiles	Docker	Text, Bash	Agnostic	N/A

IaC Terminology

Template - File(s) containing the code/configuration

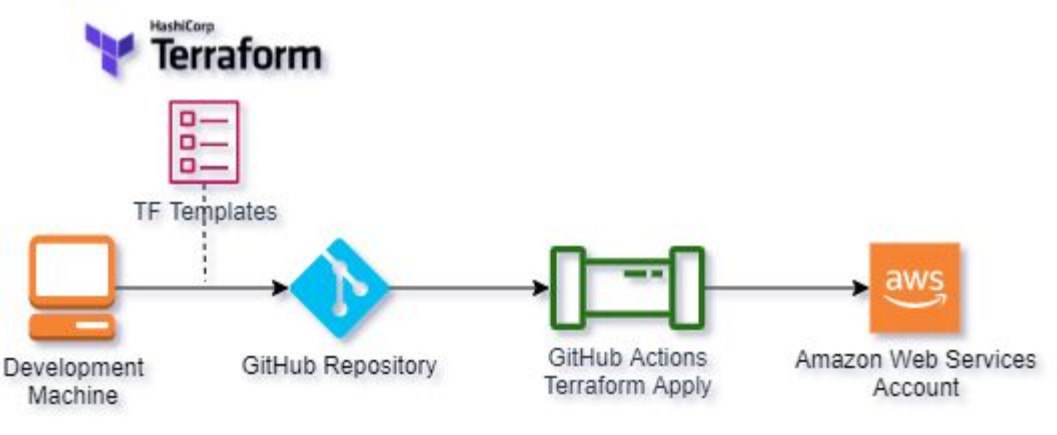
Engine - The Service that takes our templates and makes them reality!

State (State File) - “Lay of the Land” as known by our engine

Plan/Change Set/Diff - What will happen if we apply our Template?

Drift - A difference between the deployed templates and actual configuration

Live Implementation



Security of The Implementation

GitOps

- An approach to working where Git is the Source Of Truth for all System Configurations
- Pull Requests used to modify the state of a system - which is then automatically deployed via CI/CD
- Vital to a secure usage of Infrastructure-As-Code

Credentials/Access

Pipeline Credentials are often privileged!

- Deployment Credentials should be:
  - Provided in a secure manner - via a vault, secrets functionality, etc
  - Ephemeral/Rotated regularly
  - Automatically Redacted from Build Logs if exposed
  - Least Privileged - both by scope and action if possible!
- Monitor the usage of those credentials
  - Abnormal Activity
  - Credential Theft

Credentials/Access (Con’t)

- Common pattern is to only allow Write access to Prod via CI/CD
  - Ensures Git is ACTUALLY the source of truth for your Infrastructure
- Lower environments
- Break-Glass access is still needed!

State

- Most IaC tools have a concept of “State” (Which may not be directly accessible to you in SaaS/Provider Hosted Offerings)
- State is the “Lay of the Land” as your IaC tool understands it
- Terraform utilizes a “State File” that is generated or updated whenever you apply changes
- From a Security Perspective - the state file is:
  - At Best - A manifest of your entire environment
  - At Worst - Contains sensitive data/secrets

State (Con’t)

Specific to AWS and Terraform:

- Utilize either 1) Remote State functionality, or 2) Terraform Cloud
- If using Remote State Functionality via AWS S3:
  - Restrict bucket access via IAM and SCPs appropriately
  - Apply Bucket versioning
  - Apply Bucket Encryption!

Modules/Extensions Security

- Highly dependent on Risk Posture/Threat Model for Org
- Custom Resources, Providers, Modules are CODE!
- Can be used as a vehicle for malicious code
- Private Repositories exist - MAY make sense for your organization

Security Of The Code

Codifying Best Practices: Secure Modules

- Most modern IaC tooling supports some form of Module capability
  - Pulumi & CDK - Language native (Python/NodeJs packages)
  - Terraform - Modules
  - Raw Cfn - Modules
- Secure Architecture Patterns
  - Application Architecture patterns with common security challenges already solved
- Democratizing Security System/Knowledge via providing modules
  - Network Firewalls, WAFs, Authentication
  - Bastion Hosts
  - Detections
  - Tooling

Change Control & Management

- Both Git Diffs and Terraform Plan summaries make it easy to understand the **WHAT** of a change
- Pull Request Body should describe the **WHY**
- Enforcing Change Management in GitHub
  - Branch Protection for Main
  - Required Approvals for Merging Pull Requests
  - Required Checks
- Proven strategy to meet compliance requirements (eg - SOC2)

Finding Misconfigurations Before Deployment

- **Static Code Analysis** - Analyzing code without actually applying it or executing it
- Tooling specifically exists for Infrastructure-As-Code SAST!
- Complimentary with CSPM/Compliance Monitoring toolsets
- Buy or Build - still need to tune rulesets!

Credentials In Code

- Like Application Code - Hard-coded credentials still an issue in IaC
- Best to set a “zero tolerance” regardless of ENV or credential type
  - Environment Variables
  - IaC Native Functions to Retrieve credentials from
    - Secrets Manager
    - SSM
    - Other External Vaults
- Consider both CICD integration and central scanning

Conclusion

- While primarily an Automation tool - IaC brings alot net wins for Security teams
- Knowing IaC as a Security Pro helps to bridge the gap between App/Ops teams
  - Reviewing and Fixing issues
  - Being able to build our own things
- It’s Fun!

Thank you!

Contact Information:

Email: dakota.riley@aquia.us
LinkedIn: https://www.linkedin.com/in/dakota-riley-b48401b7
GitHub: https://github.com/rileydakota

Presentation and Code for the Demo available here:
https://github.com/rileydakota/nku-cyber-2021-iac-security