The IaC Tooling Face-off for Modern Cloud Native Ops

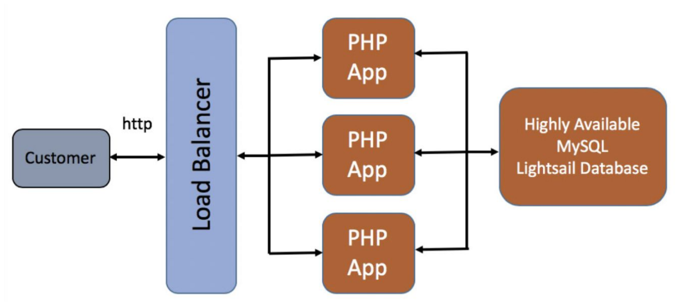
Ronny Orot
Software Engineer at env0
OpenTofu core team member

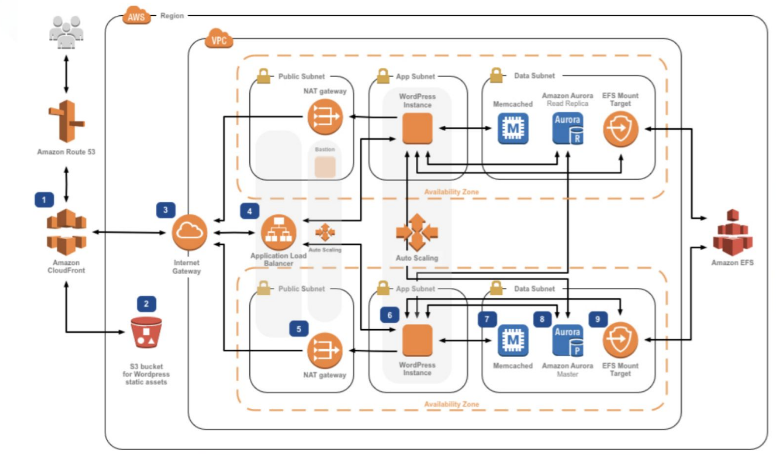
Agenda

- Why do we need IaC
- How to choose the Right IaC Tool
- Organization's IaC Maturity
- Unified DevOps Process

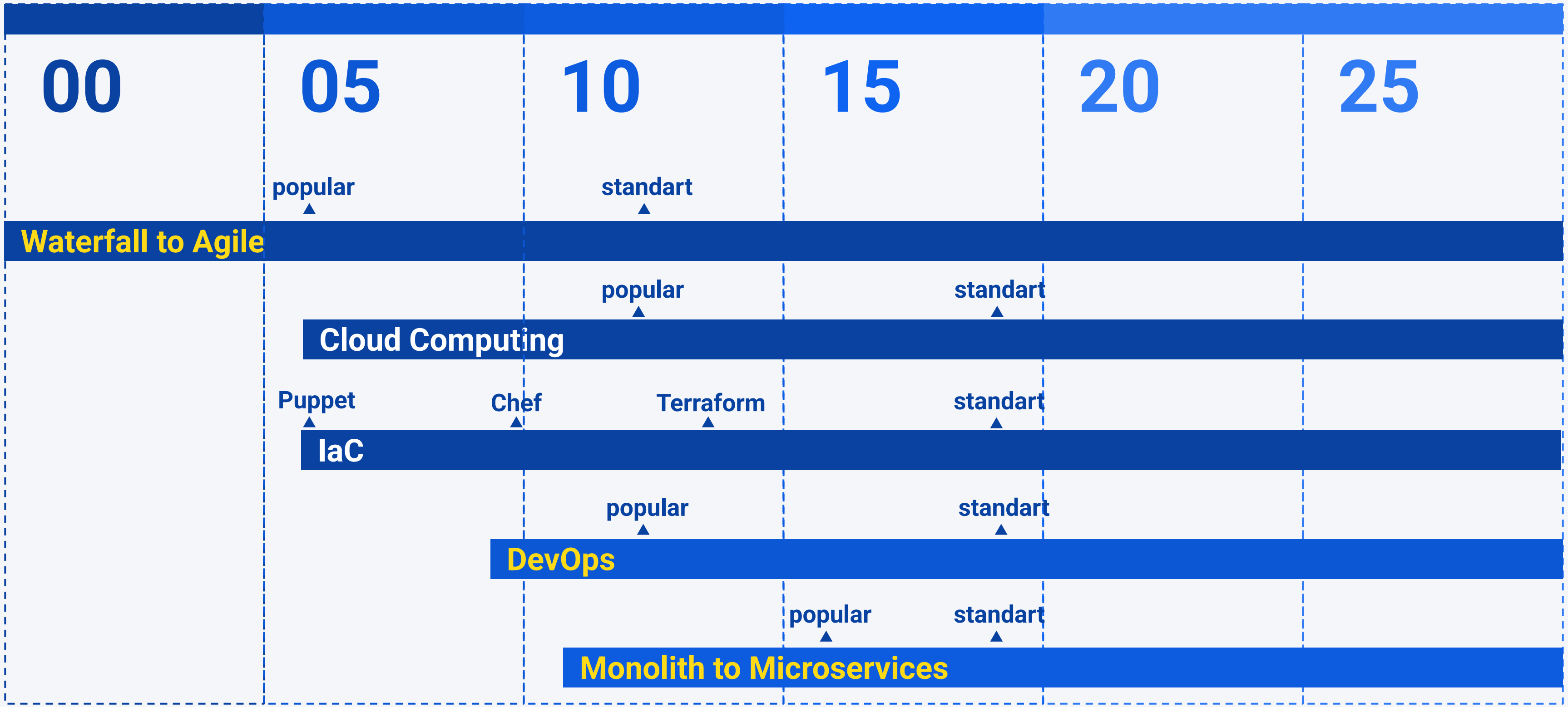
Application Have Grown More Complex

Then → Now



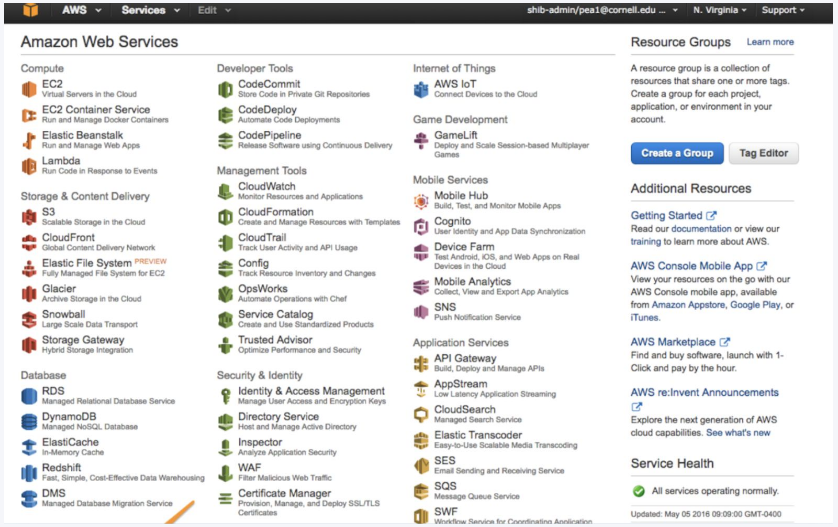


After Y2K


00
05
10
15
20
25
popular
standart
Waterfall to Agile
popular
standart
Cloud Computing
Puppet
Chef
Terraform
standart
IaC
popular
standart
DevOps
popular
standart
Monolith to Microservices

The Rise of IaC

Then → Now



AWS · Services · Edit · shib-admin/pea1@cornell.edu ... · N. Virginia · Support

Amazon Web Services

Compute
- EC2 — Virtual Servers in the Cloud
- EC2 Container Service — Run and Manage Docker Containers
- Elastic Beanstalk — Run and Manage Web Apps
- Lambda — Run Code in Response to Events

Storage & Content Delivery
- S3 — Scalable Storage in the Cloud
- CloudFront — Global Content Delivery Network
- Elastic File System PREVIEW — Fully Managed File System for EC2
- Glacier — Archive Storage in the Cloud
- Snowball — Large Scale Data Transport
- Storage Gateway — Hybrid Storage Integration

Database
- RDS — Managed Relational Database Service
- DynamoDB — Managed NoSQL Database
- ElastiCache — In-Memory Cache
- Redshift — Fast, Simple, Cost-Effective Data Warehousing
- DMS — Managed Database Migration Service

Developer Tools
- CodeCommit — Store Code in Private Git Repositories
- CodeDeploy — Automate Code Deployments
- CodePipeline — Release Software using Continuous Delivery

Management Tools
- CloudWatch — Monitor Resources and Applications
- CloudFormation — Create and Manage Resources with Templates
- CloudTrail — Track User Activity and API Usage
- Config — Track Resource Inventory and Changes
- OpsWorks — Automate Operations with Chef
- Service Catalog — Create and Use Standardized Products
- Trusted Advisor — Optimize Performance and Security

Security & Identity
- Identity & Access Management — Manage User Access and Encryption Keys
- Directory Service — Host and Manage Active Directory
- Inspector — Analyze Application Security
- WAF — Filter Malicious Web Traffic
- Certificate Manager — Provision, Manage, and Deploy SSL/TLS Certificates

Internet of Things
- AWS IoT — Connect Devices to the Cloud

Game Development
- GameLift — Deploy and Scale Session-based Multiplayer Games

Mobile Services
- Mobile Hub — Build, Test, and Monitor Mobile Apps
- Cognito — User Identity and App Data Synchronization
- Device Farm — Test Android, iOS, and Web Apps on Real Devices in the Cloud
- Mobile Analytics — Collect, View and Export App Analytics
- SNS — Push Notification Service

Application Services
- API Gateway — Build, Deploy and Manage APIs
- AppStream — Low Latency Application Streaming
- CloudSearch — Managed Search Service
- Elastic Transcoder — Easy-to-Use Scalable Media Transcoding
- SES — Email Sending and Receiving Service
- SQS — Message Queue Service
- SWF — Workflow Service for Coordinating Application

Resource Groups Learn more

A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.

Create a Group · Tag Editor

Additional Resources

Getting Started
Read our documentation or view our training to learn more about AWS.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

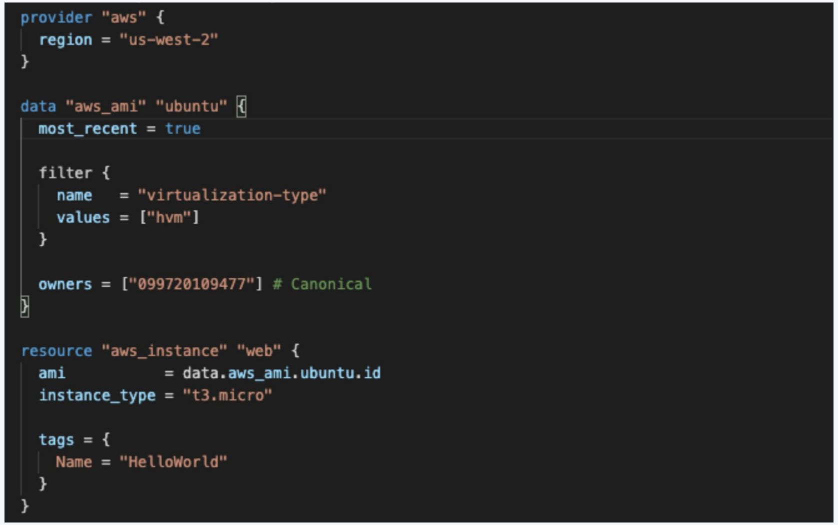
AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

AWS re:Invent Announcements
Explore the next generation of AWS cloud capabilities. See what's new

Service Health

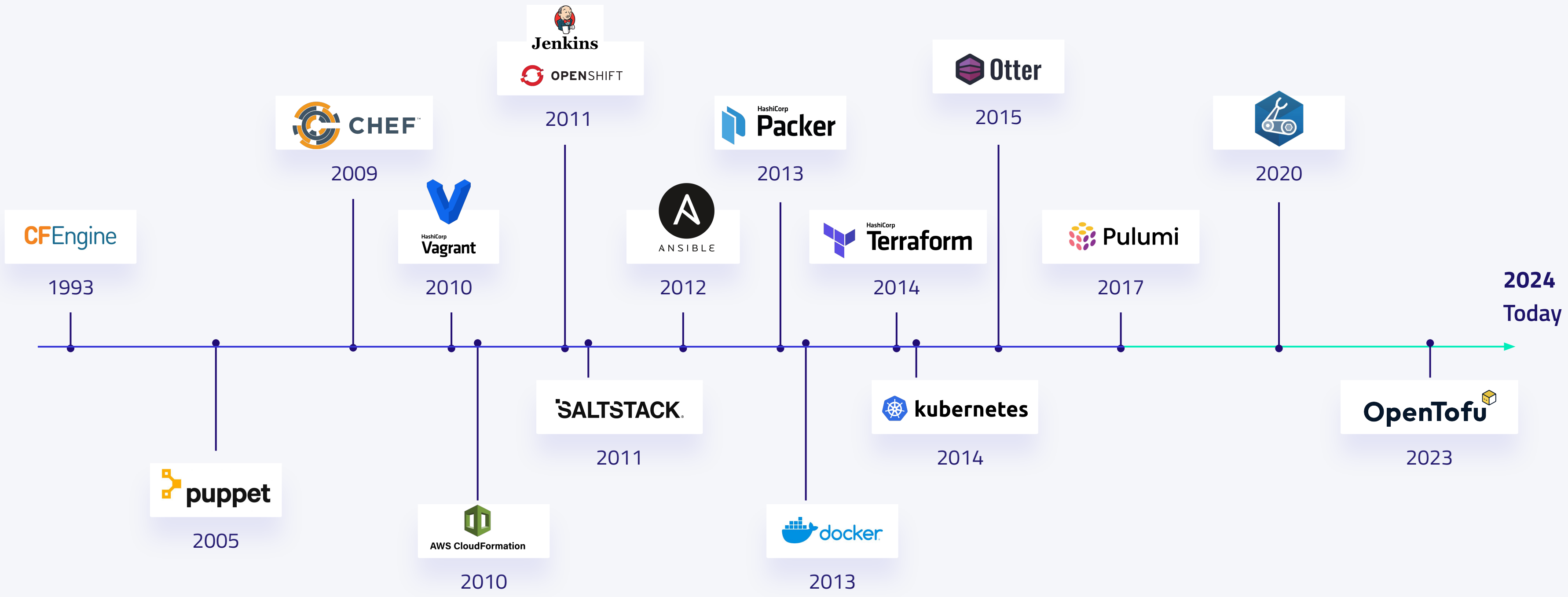
All services operating normally.

Updated: May 05 2016 09:09:00 GMT-0400



```
provider "aws" {
  region = "us-west-2"
}

data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "web" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```

The IaC Tooling Challenge



Choosing the Right IaC Tool

- Security and Compliance
- Scale and Performance
- Learning Curve and Adoption

IaC Battleground

IaC Battleground

Pulumi

Jenkins

GitHub Actions

ANSIBLE

Terragrunt

argo

OpenTofu

HashiCorp Terraform

aws Cloud Development Kit

HELM

IaC Battleground - Round 1

HashiCorp
Terraform

- Industry standard
- Cloud agnostic
- Large community and ecosystem
- Modular & reusable
- DSL declarative language (HCL/JSON)
- Performance & scalability
- State file management
- Vendor lock-in to HashiCorp

IaC Battleground - Round 1

HashiCorp
Terraform

- Industry standard
- Cloud agnostic
- Large community and ecosystem
- Modular & reusable
- DSL declarative language (HCL/JSON)
- Performance & scalability
- State file management
- Vendor lock-in to HashiCorp

VS

AWS CloudFormation

- Maintained by vendor
- No state management
- Built in Drift Detection and GitOps flows
- Verbose Syntax (JSON/YAML)
- Performance & scalability
- AWS lock-in
- Limited modularity

IaC Battleground - Round 2

Pulumi

- Cloud-agnostic
- State management options
- Code reusability & modularity
- Language familiarity (TypeScript, Python, Go & more)
- Community size and support
- Performance on large deployments
- Easy to over complex your IaC

VS

IaC Battleground - Round 2

Pulumi

- Cloud-agnostic
- State management options
- Code reusability & modularity
- Language familiarity (TypeScript, Python, Go & more)
- Community size and support
- Performance on large deployments
- Easy to over complex your IaC

VS

OpenTofu

- Drop-in replacement for Terraform
- True open source by Linux Foundation
- Extensive ecosystem
- State handling capabilities
- DSL declarative language (HCL/JSON)
- Performance & scalability
- Relatively new and smaller (but growing) community

IaC Battleground - Round 3

ANSIBLE

- Configuration using YAML
- Agentless
- Large community & prebuilt roles
- No state management

IaC Battleground - Round 3

ANSIBLE

- Configuration using YAML
- Agentless
- Large community & prebuilt roles
- No state management

VS

SALT PROJECT

- Server-client configuration
- Highly scalable & fast execution
- Utilizes state for configuration management
- Agent maintenance and updates
- Unknowns following VMware/Broadcom acquisitions

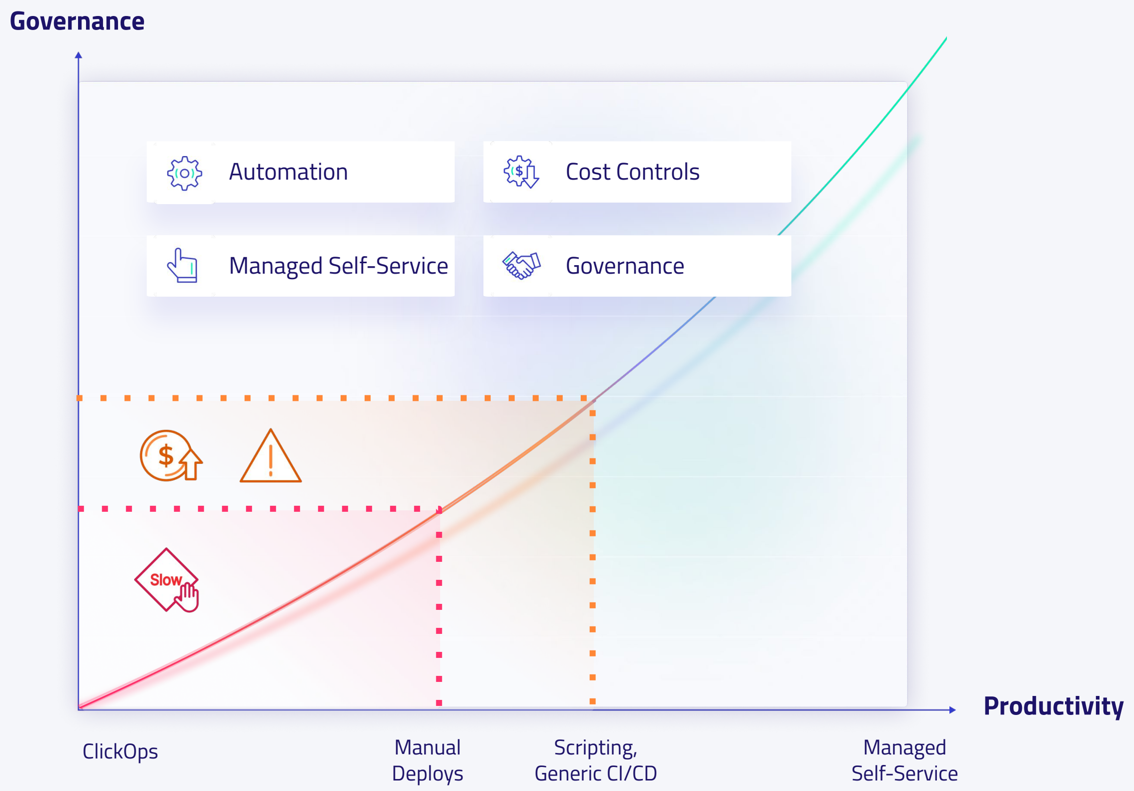
IaC Battleground

So... who won?

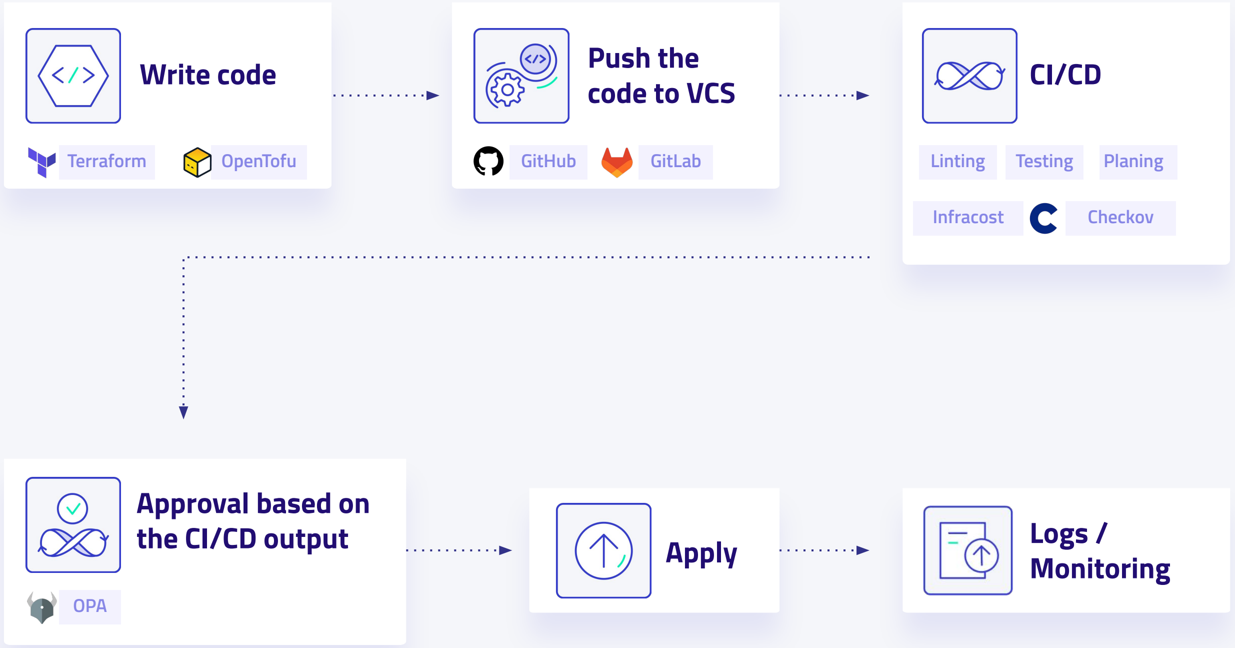
The Multi-Tool Approach

- Potential benefits of using a combination of IaC tools:
  - Leverage strengths of each IaC type
  - Address diverse requirements (cloud-native, traditional, legacy)
  - Facilitate migration and integration strategies
- Consider in-house scripting for specific needs but it requires maintenance.

Understanding Your Organization's IaC Maturity



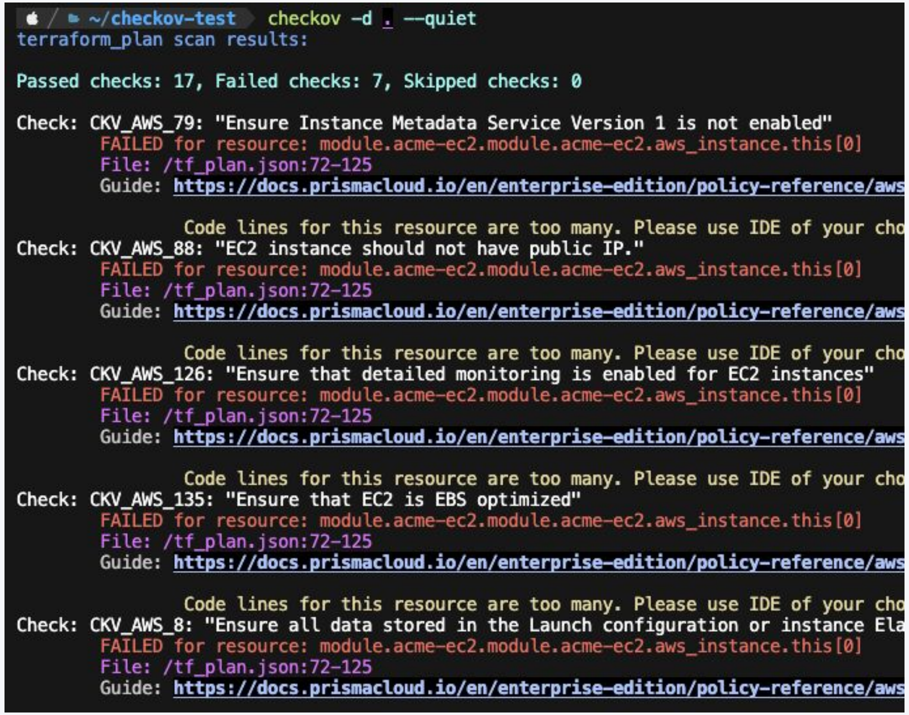
Unified DevOps Process



Static Code Analysis

- Automatically scan for misconfigurations
- Identify security vulnerabilities before deployment
- Ensure compliance with industry standards and internal policies

checkov

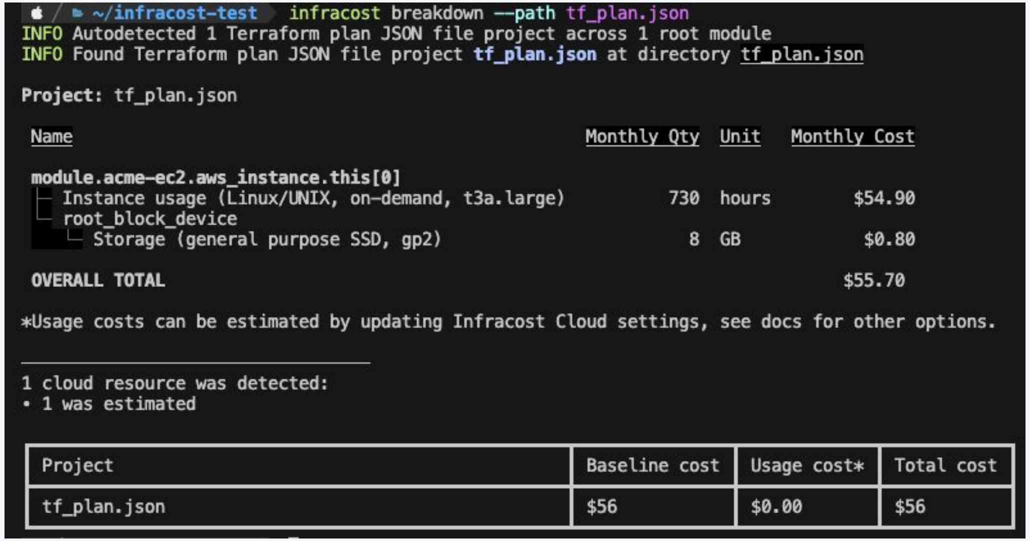


```
 ~/checkov-test  checkov -d . --quiet
terraform_plan scan results:

Passed checks: 17, Failed checks: 7, Skipped checks: 0

Check: CKV_AWS_79: "Ensure Instance Metadata Service Version 1 is not enabled"
        FAILED for resource: module.acme-ec2.module.acme-ec2.aws_instance.this[0]
        File: /tf_plan.json:72-125
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws

                Code lines for this resource are too many. Please use IDE of your cho
Check: CKV_AWS_88: "EC2 instance should not have public IP."
        FAILED for resource: module.acme-ec2.module.acme-ec2.aws_instance.this[0]
        File: /tf_plan.json:72-125
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws

                Code lines for this resource are too many. Please use IDE of your cho
Check: CKV_AWS_126: "Ensure that detailed monitoring is enabled for EC2 instances"
        FAILED for resource: module.acme-ec2.module.acme-ec2.aws_instance.this[0]
        File: /tf_plan.json:72-125
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws

                Code lines for this resource are too many. Please use IDE of your cho
Check: CKV_AWS_135: "Ensure that EC2 is EBS optimized"
        FAILED for resource: module.acme-ec2.module.acme-ec2.aws_instance.this[0]
        File: /tf_plan.json:72-125
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws

                Code lines for this resource are too many. Please use IDE of your cho
Check: CKV_AWS_8: "Ensure all data stored in the Launch configuration or instance Ela
        FAILED for resource: module.acme-ec2.module.acme-ec2.aws_instance.this[0]
        File: /tf_plan.json:72-125
        Guide: https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws
```

Cost Estimation

- Understand IaC updates on cost before deployment
- Shift cost mindset left
- Include cost as part of approval flows

Infracost



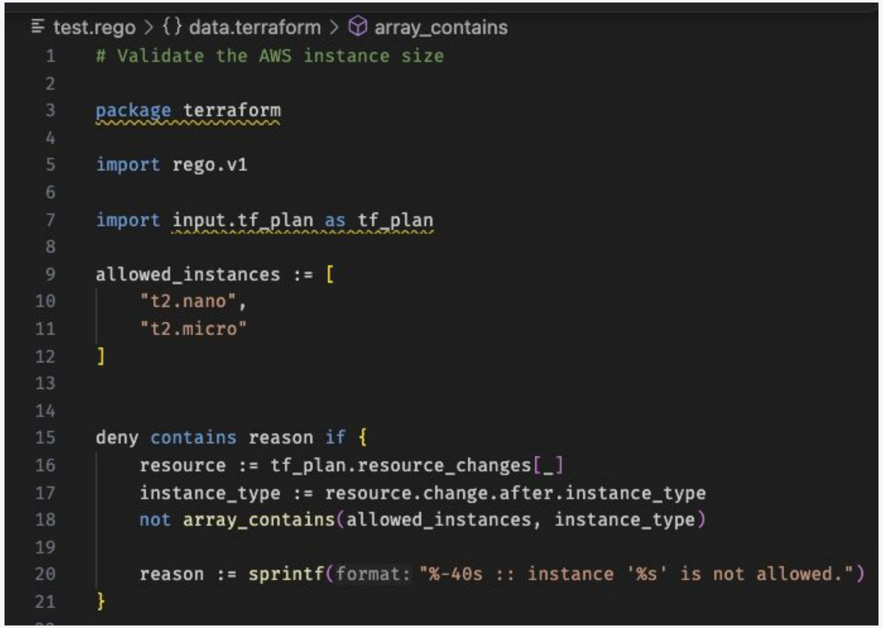
```
 ~/infracost-test  infracost breakdown --path tf_plan.json
INFO Autodetected 1 Terraform plan JSON file project across 1 root module
INFO Found Terraform plan JSON file project tf_plan.json at directory tf_plan.json

Project: tf_plan.json

 Name                                                        Monthly Qty  Unit   Monthly Cost

 module.acme-ec2.aws_instance.this[0]
 ├─ Instance usage (Linux/UNIX, on-demand, t3a.large)               730  hours        $54.90
 └─ root_block_device
    └─ Storage (general purpose SSD, gp2)                             8  GB            $0.80

 OVERALL TOTAL                                                                        $55.70

*Usage costs can be estimated by updating Infracost Cloud settings, see docs for other options.

──────────────────────────────────
1 cloud resource was detected:
• 1 was estimated
```

Project	Baseline cost	Usage cost*	Total cost
tf_plan.json	$56	$0.00	$56

Policy-as-Code

- Codify organizational policies
- Reduce risk and human error when validating code, plans, and outputs
- Reduce bottleneck of human approval

Open Policy Agent



```rego
# Validate the AWS instance size

package terraform

import rego.v1

import input.tf_plan as tf_plan

allowed_instances := [
    "t2.nano",
    "t2.micro"
]


deny contains reason if {
    resource := tf_plan.resource_changes[_]
    instance_type := resource.change.after.instance_type
    not array_contains(allowed_instances, instance_type)

    reason := sprintf(format: "%-40s :: instance '%s' is not allowed.")
}
```

In Summary

- No single "best" IaC tool for all scenarios
- The "winner" depends on your organization's:
  - Maturity Level
  - Technology Stack
  - Use Cases and Requirements
- Adopt a strategic and adaptable approach

Thank you!

Questions?