Which IAC is Right For Me?

Audience

- People who want to know what Infrastructure as Code (IAC) is
- People who want to know why you use IAC
- People who don’t know where to start with IAC
- People new to the cloud
- People who are using IAC but want to know different options

Agenda

- History
- What is Infrastructure as Code?
- Why should you care?
- All the options that are out there
- Azure Bicep + ARM
- Terraform
- AWS Cloudformation + CDK
- Pulumi

Goals

- Understand what and why of IAC
- Understand which IAC fits my environment

Who is Matt Phillips?

- Technical Lead at Integrity
- 13 years in software
- Uses multiple IAC technologies every day

Who is Scott Sauber?

- Director of Engineering at Lean TECHniques
- Microsoft MVP
- Co-organizer of Iowa .NET User Group
- Dometrain Author

In the Before Times

- On-premises
- Sysadmins maintained the infrastructure (pets)
- Clickety Clack Configuration
- “It works in Dev, but not in Production”

“It Works On My Machine”

- Every Developer At Least Once

Then…

- But then cloud replaced the datacenter (still pets)
- Still Clickety Clack Configuration
- “It still works in Dev, but still not in Production”
- But now we don’t own the hardware!

What is Infrastructure as Code (IAC)

- Source code that defined provisioning resources that’s configurable and repeatable across all environments
- Stored in version control
- Declarative – what resources to create, not how to create them
- Deployed via automation (ideally pipeline, but could be local)
- Resource properties can be linked together

Why IAC?

- Promotes consistency and standardization
- “It Works On My Machine” goes away
- “It works in Dev, but not in Production” goes away
- Audit trail of who did what and when
- Resources are Transient not Permanent
- Can be deleted and re-created with ease (besides your DB)
- Standing up a new environment is a few lines of code and minutes

So what can I create with IAC?

- Virtual Machines
- Web Servers
- Databases
- Secret Stores
- Networking
- IAM Policies
- Monitoring
- DNS
- …pretty much everything

Okay I get it
...but how?

What is Azure Bicep?

- Used to configure Azure resources
- Built and maintained by Microsoft
- Domain-specific language (fancy word for custom)
- Provides intellisense, error checking, “whatif,” and orders the resource creations
- Built on top of Azure Resource Manager (ARM) – don’t use ARM directly
- No state file

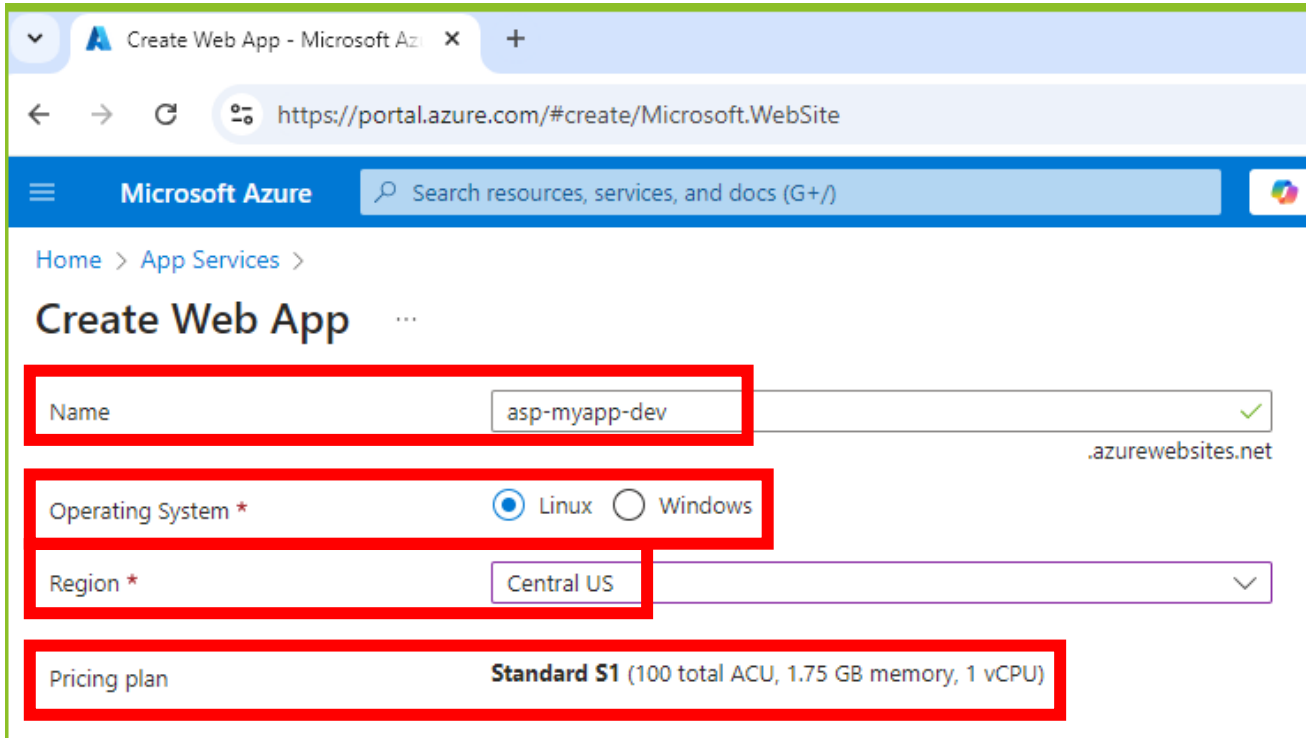
What is Azure Bicep?

appservice.bicep

```
resource appServicePlan 'Microsoft.Web/serverfarms@2022-09-01' = {
  name: 'asp-myapp-dev'
  location: 'centralus'
  sku: {
    name: 'S1'
  }
  kind: 'linux'
}
```

appservice.bicep

```
resource appServicePlan 'Microsoft.Web/serverfarms@2022-09-01' = {
  name: 'asp-myapp-dev'
  kind: 'linux'
  location: 'centralus'
  sku: {
    name: 'S1'
  }
}
```



Demo

Terraform

- Created by HashiCorp (recently acquired by IBM)
- Domain-specific language
- Configure different clouds (Azure, AWS, GCP, VMware, etc) via Providers
- BUT – can’t take same Terraform and run it on both Azure and AWS
- Maintains state of infrastructure via state file
- Likely the most popular IAC tool in the world
- Recently changed to less permissive BSL license
- Recently acquired by IBM

appservice.tf

```
resource "azurerm_service_plan" "myapp" {
  name                = "asp-myapp-dev"
  resource_group_name = "rg-myapp-dev"
  location            = "centralus"
  os_type             = "Linux"
  sku_name            = "S1"
}
```

appservice.bicep

```
resource appServicePlan 'Microsoft.Web/serverfarms@2022-09-01' = {
  name: 'asp-myapp-dev'
  location: 'centralus'
  sku: {
    name: 'S1'
  }
  kind: 'linux'
}
```

appservice.tf

```
resource "azurerm_service_plan" "myapp" {
  name                = "asp-myapp-dev"
  resource_group_name = "rg-myapp-dev"
  location            = "centralus"
  os_type             = "Linux"
  sku_name            = "S1"
}
```

AWS Cloudformation

- Used to configure AWS resources
- Built and maintained by Amazon
- YAML or JSON files

lambda.yml

```yaml
Resources:
  LambdaFunction:
    Type: 'AWS::Lambda::Function'
    Properties:
      FunctionName: mylambda
      Handler: index.handler
      Runtime: nodejs20.x
      MemorySize: 1024
```

AWS CDK

- Used to configure AWS resources
- Built and maintained by Amazon
- You write your language of choice – TypeScript, JavaScript, Python, Java, C#, or Go
- Built on top of Cloudformation

lambda.ts

```
1  import * as cdk from 'aws-cdk-lib';
2  import { Construct } from 'constructs';
3  import * as lambda from 'aws-cdk-lib/aws-lambda';
4
5  export class CdkHelloWorldStack extends cdk.Stack {
6    constructor(scope: Construct, id: string, props?: cdk.StackProps) {
7      super(scope, id, props);
8
9      const myLambdaFunction = new lambda.Function(this, 'HelloWorldFunction', {
10       functionName: 'mylambda'
11       handler: 'index.handler',
12       runtime: lambda.Runtime.NODEJS_20_X,
13       memorySize: 1024,
14     });
15   }
16 }
```

Pulumi

- Created by Pulumi Corporation
- Configure different clouds (Azure, AWS, GCP, etc)
- You write your language of choice – TypeScript, JavaScript, Python, Java, C#, Go, or Visual Basic (yes really)
- First big player to market with IAC using an existing language

Pulumi

lambda.ts

```
1   import * as pulumi from '@pulumi/pulumi';
2   import * as aws from '@pulumi/aws';
3
4   // Define the Lambda function
5   const myLambda = new aws.lambda.Function('myLambda', {
6       functionName: 'mylambda',
7       runtime: aws.lambda.NodeJS20dXRuntime,
8       handler: 'index.handler',
9       memorySize: 1024
10  });
```

Configuration Files vs Code Files

- Bicep, ARM, Terraform, and Cloudformation are DSL config files
- CDK and Pulumi are libraries of existing languages
- CDK for Terraform is CDK but built on Terraform, supported by Hashicorp
- Configuration popular with Ops-focused teams
- Code popular with Dev-focused teams
- Code leverages existing skills – packages, syntax, autocomplete, etc

Okay I get it
...but which do I pick?

Two main decisions

- Do I go first-party with Azure/AWS/my cloud? Or do I go third-party like Terraform or Pulumi?
- Do I choose a DSL config file based tech or GPL code file based tech?

DevSecOps

- Integrate an IAC scanner into your CD pipeline
- Catch security misconfigurations (ie require TLS 1.2, no public S3 buckets, etc) in a Pull Request before it hits your cloud
- Checkov is a free tool, terrascan

```
@samgabrail →/workspaces/env0-iac-scanning/Terraform (main) $
@samgabrail →/workspaces/env0-iac-scanning/Terraform (main) $ checkov -f ec2.tf
[ secrets framework ]: 100%|[1/1], Current File Scanned=ec2.tftf
[ terraform framework ]: 100%|[1/1], Current File Scanned=ec2.tf

By bridgecrew.io | version: 2.3.187

terraform scan results:

Passed checks: 4, Failed checks: 12, Skipped checks: 0

Check: CKV_AWS_88: "EC2 instance should not have public IP."
        PASSED for resource: aws_instance.web_host
        File: /ec2.tf:1-32
        Guide: https://docs.bridgecrew.io/docs/public_12
Check: CKV_AWS_25: "Ensure no security groups allow ingress from 0.0.0.0:0 to port 3389"
        PASSED for resource: aws_security_group.web-node
        File: /ec2.tf:77-115
        Guide: https://docs.bridgecrew.io/docs/networking_2
Check: CKV_AWS_277: "Ensure no security groups allow ingress from 0.0.0.0:0 to port -1"
        PASSED for resource: aws_security_group.web-node
        File: /ec2.tf:77-115
        Guide: https://docs.bridgecrew.io/docs/ensure-aws-security-group-does-not-allow-all-traffic-on-all-ports
Check: CKV2_AWS_5: "Ensure that Security Groups are attached to another resource"
        PASSED for resource: aws_security_group.web-node
        File: /ec2.tf:77-115
        Guide: https://docs.bridgecrew.io/docs/ensure-that-security-groups-are-attached-to-ec2-instances-or-elastic-network-interfaces-enis
Check: CKV_AWS_46: "Ensure no hard-coded secrets exist in EC2 user data"
        FAILED for resource: aws_instance.web_host
        File: /ec2.tf:1-32
        Guide: https://docs.bridgecrew.io/docs/bc_aws_secrets_1

                1 | resource "aws_instance" "web_host" {
                2 |   # ec2 have plain text secrets in user data
                3 |   ami           = "${var.ami}"
                4 |   instance_type = "t2.nano"
                5 |
                6 |   vpc_security_group_ids = [
                7 |   "${aws_security_group.web-node.id}"]
```

Takeaways

- Understand what IAC is and why you should care about it
- Understand which IAC might be a good fit for you
- Use a security scanner for your IAC

Questions?