INFRASTRUCTURE AS CODE IS REAL

USING THE CLOUD TO PROVISION INFRASTRUCTURE WITH CODE

SHAUN MCCULLOUGH
Craftsman

One of a kind infrastructure
Difficult to track changes
Hard to alter design
Blueprints

Recreate anytime
Software dev practices
Constant iteration
The DevOps Story

Set of software development practices that combines software development (Dev) and information technology operations (Ops) to **shorten** the systems development **life cycle** while delivering features, fixes, and updates frequently in close alignment with business objectives.
The Promise of Infrastructure As Code

• Infrastructure is malleable, adaptable, fault resistant, recoverable

• Continuous integration workflows

• Create gates for all infrastructure changes
The Virtual Machine

- Virtualization allows us to use code to build
- Dev and Prod from single code source
- Configure at run time, harder to manage full lifecycle

Vagrant by HashiCorp
https://www.vagrantup.com
The Host Configuration

- Install packages, create config files, manage users
- Manage across fleet of similar systems
- Leverage param, runtime decisions, environmental specifics
- Works across all on-prem and cloud hosts

Ansible  ansible.com
Chef  chef.io
Puppet  puppet.com
Docker  docker.com
IaC + Cloud = Next Level

- Leverage cloud API’s to interact with EVERYTHING
- Network, host, data storage, security
- Clouds are self service, automatable, with standardization
- Tooling is provided to make API interactions easier.

AWS CloudFormation

Azure Resource Manager

Terraform  terraform.io
AWS CloudFormation

“AWS CloudFormation provides a common language for you to describe and provision all the infrastructure resources in your cloud environment. CloudFormation allows you to use a simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all regions and accounts. This file serves as the single source of truth for your cloud environment.”

- YAML or JSON Files
- Build stacks of connected resources
- Manage stacked resources as a group
"Resources" : {
  "MyWebpage" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "UserData" : {
        "Fn::Base64" : {
          "Fn::Join" : ["", ["#!/bin/bash \n","sudo yum update -y \n","sudo yum install nginx -y \n","sudo service nginx start"]]]
      },
      "InstanceType" : "t2.micro",
      "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
      "KeyName" : "cybergoof-aws",
      "ImageId" : "ami-0080e4c5bc078760e"
    }
  }
},
"InstanceSecurityGroup" : {
    "Type" : "AWS::EC2::SecurityGroup",
    "Properties" : {
        "GroupDescription" : "Enable SSH access",
        "SecurityGroupIngress" : [
            { "IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0" },
            { "IpProtocol" : "tcp", "FromPort" : "443", "ToPort" : "443", "CidrIp" : "0.0.0.0/0" }
        ]
    }
}
,
### Events

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Logical ID</th>
<th>Status</th>
<th>Status reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-06-16 17:38:29 UTC-0400</td>
<td>Basic</td>
<td>CREATE_COMPLETE</td>
<td></td>
</tr>
<tr>
<td>2019-06-16 17:38:26 UTC-0400</td>
<td>MyWebpage</td>
<td>CREATE_COMPLETE</td>
<td></td>
</tr>
<tr>
<td>2019-06-16 17:37:53 UTC-0400</td>
<td>MyWebpage</td>
<td>CREATE_IN_PROGRESS</td>
<td>Resource creation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Initiated</td>
</tr>
<tr>
<td>2019-06-16 17:37:52 UTC-0400</td>
<td>MyWebpage</td>
<td>CREATE_IN_PROGRESS</td>
<td></td>
</tr>
<tr>
<td>2019-06-16 17:37:49 UTC-0400</td>
<td>InstanceSecurityGroup</td>
<td>CREATE_COMPLETE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Initiated</td>
</tr>
<tr>
<td>2019-06-16 17:37:43 UTC-0400</td>
<td>InstanceSecurityGroup</td>
<td>CREATE_IN_PROGRESS</td>
<td></td>
</tr>
<tr>
<td>2019-06-16 17:37:39 UTC-0400</td>
<td>Basic</td>
<td>CREATE_IN_PROGRESS</td>
<td>User Initiated</td>
</tr>
</tbody>
</table>
Welcome to nginx on the Amazon Linux AMI!

This page is used to test the proper operation of the nginx HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

Website Administrator

This is the default index.html page that is distributed with nginx on the Amazon Linux AMI. It is located in /var/lib/nginx/html.

You should now put your content in a location of your choice and edit the root configuration directive in the nginx configuration file /etc/nginx/nginx.conf.
Cloud Development Kit (CDK)

Use the AWS CDK to define cloud resources using one of the supported programming languages: C#/.NET, Java, JavaScript, Python, or TypeScript.

Developers can define **reusable** cloud components known as Constructs. You compose these together into Stacks and Apps.
constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
  super(scope, id, props);
  // Get the Amazon Linux Image
  const amznLinux = new ec2.AmazonLinuxImage({
    generation: ec2.AmazonLinuxGeneration.AMAZON_LINUX,
    edition: ec2.AmazonLinuxEdition.STANDARD,
    virtualization: ec2.AmazonLinuxVirt.HVM,
    storage: ec2.AmazonLinuxStorage.GENERAL_PURPOSE,
  });

  // Get my VPC
  const vpc = this.buildVpc();

  const mySecurityGroup = this.buildSecurityGroup(this, vpc);
  const userDataValues = ec2 UserData.forLinux();
  userDataValues.addCommands('sudo yum update -y');
  userDataValues.addCommands('sudo yum install nginx -y');
  userDataValues.addCommands('sudo service nginx start');

  // Build the webpage
  const webpage = new ec2.Instance(this, 'MyWebpage', {
    machineImage: amznLinux,
    userData: userDataValues,
    keyName: "cybergoof-aws",
    vpc,
    instanceType: new ec2.InstanceType("t2.micro"),
    vpcSubnets: { onePerAz: true, subnetType: ec2.SubnetType.PUBLIC },
  });
  webpage.addSecurityGroup(mySecurityGroup);
  new cdk.CfnOutput(this, 'ExternalIP', {value: 'http://' + webpage.instancePublicIp});
}
// Build the security groups
buildSecurityGroup(stack: cdk.Construct, vpc: ec2.IVpc) {
        vpc,
        description: 'Allow ssh access to ec2 instances',
        allowAllOutbound: true  // Can be set to false
    });
    mySecurityGroup.addIngressRule(ec2.Peer.anyIpv4(), ec2.Port.tcp(22), 'allow port 22 for ssh');
    mySecurityGroup.addIngressRule(ec2.Peer.anyIpv4(), ec2.Port.tcp(80), 'allow port 80 access from the world');
    mySecurityGroup.addIngressRule(ec2.Peer.anyIpv4(), ec2.Port.tcp(443), 'allow port 443 access from the world');
    return mySecurityGroup
}

110 lines of Typescript code generated
750 lines of CloudFormation
Things to consider

• Infrastructure team will need new skills
• Creating blueprints will take longer at first
• You will want to adapt your design
THANK YOU

SHAUN MCCULLOUGH

H&A SECURITY SOLUTIONS

@THECYBERGOOF

CYBERGOOF