



Terraform & Infrastructure as Code



Ben Higginbottom

(kind of a big deal)

((seriously - I've helped hijack a spacecraft))

What IaC isn't - The Anti-pattern



Configuration Management:

Chef already does this “very well”, and although there are commonalities (and extensions) it’s first vs second order.



Remote Execution:

It can do this, but not as well as Ansible or Expect

What IaC Isn't

Traditionally a server has been a significant item

(hands up those people whose laptop doesn't have a 'special' name)

HP DL-380 - £10,000 (+ software)

HP C9000 - £30,000 (+ software)

If they get sick, you fix them... And isn't open source great!



What IaC Is

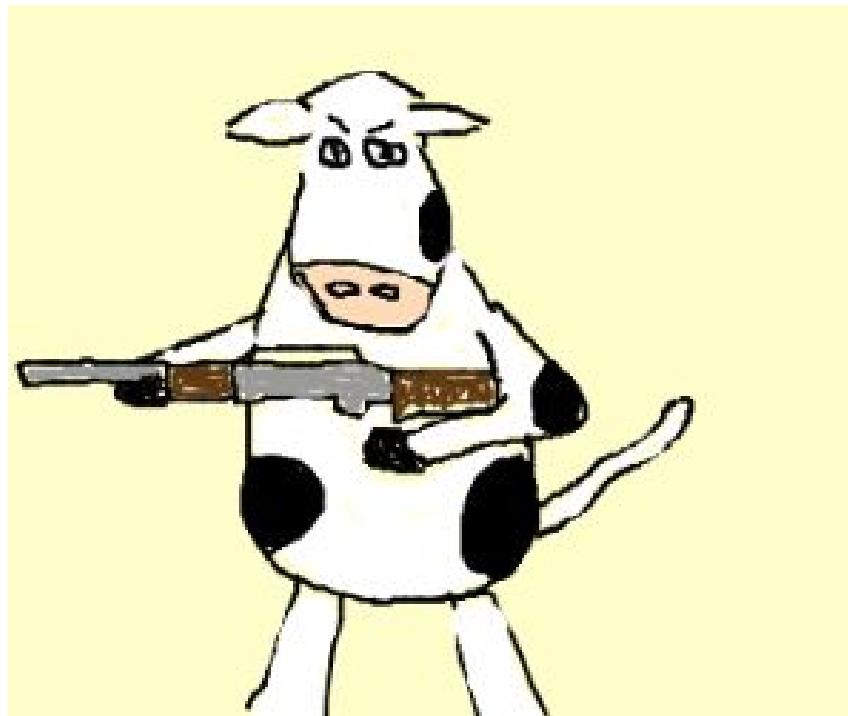
The server is cattle now, not a pet

It has no more existence, or right thereof more than any other software process

It gets sick...



Do I need to paint you a picture...



Seriously...

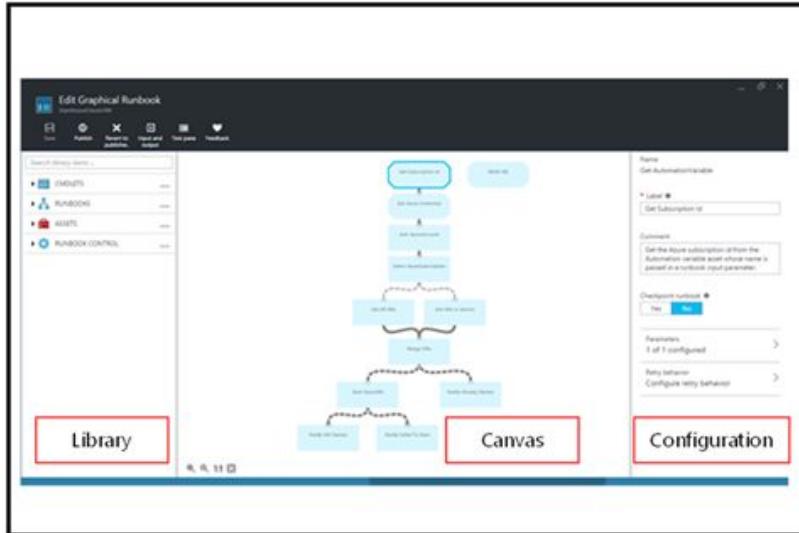
```
le3
/home/tmd # ps -ef | grep -i c
id      17319 17308 81 17:39
oot     18458 18456  0 18:00
/home/tmd # kill -■
```



Public Cloud Providers

Azure

Automation Runbooks -



Powershell based with a GUI

Very useful for a pure Microsoft environment, integrates into OMS, but limited in support of non-Microsoft products

Really configuration management-lite



Cloud Deployment Manager - Similar to CloudFormation, but using a simpler YAML markup structure to work with GCE and GCP (and of course the API's)

```
1 resources:
2 - type: compute.v1.instance
3   name: vm-my-first-deployment
4   properties:
5     zone: us-central1-f
6     machineType: https://www.googleapis.com/compute/v1/projects/bentest/zones/us-central1-f/machineTypes/f1-micro
7     disks:
8       - deviceName: boot
9         type: PERSISTENT
10        boot: true
11        autoDelete: true
12        initializeParams:
13          sourceImage: https://www.googleapis.com/compute/v1/projects/debian-cloud/global/images/debian-8-jessie-v20160301
14   networkInterfaces:
15     - network: https://www.googleapis.com/compute/v1/projects/bentest/global/networks/default
16       accessConfigs:
17         - name: External NAT
18           type: ONE_TO_ONE_NAT
```

Cloud Providers - AWS

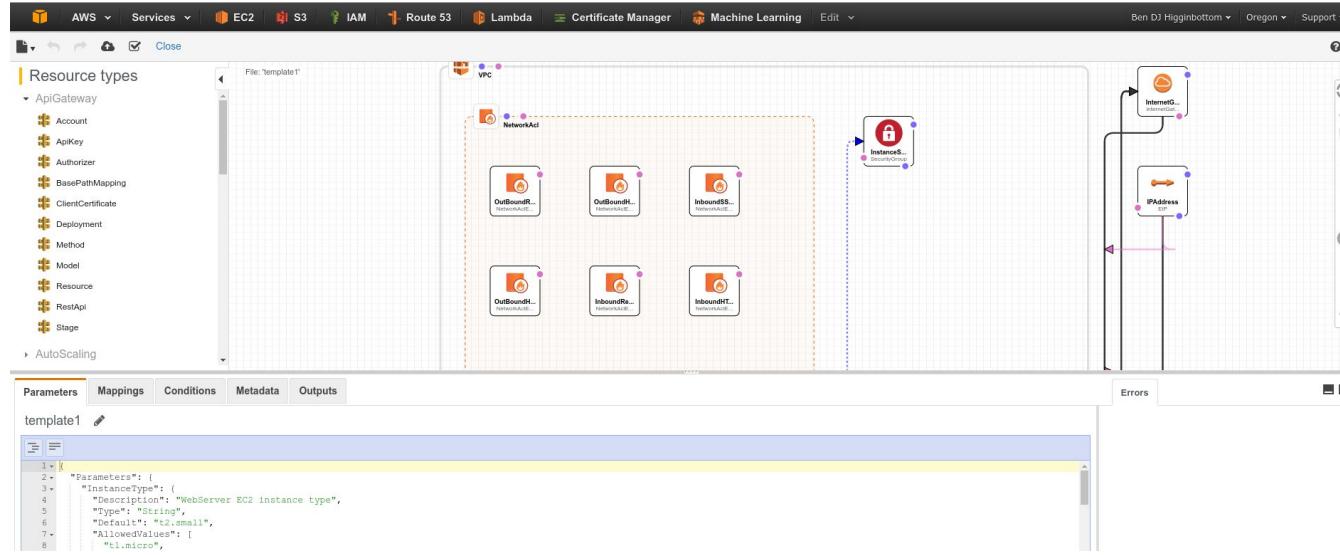
Elastic Beanstalk - often called Amazon's PaaS system, it's really automated provisioning with a templating mechanism.

OpsWorks - Run Chef cookbooks targeting EC2, RDS and most other components, however it's config management that can also do some IaC

API's - Coupled with the SDK's this provides a massively powerful and flexible solution, but requires a considerable amount of effort

Cloudformation + CloudDesigner

Template driven IaC solutions, very flexible, very powerful, coupled with a graphical interface not dissimilar to Visio, but very domain specific - for a Single EC2 instance its this -



Cloudformation

This is approximately ¼ of the JSON definition needed to build a single EC2 instance in a single VPC

Partially this is down to it acting like a silicon compiler from the 00's

Massively powerful, but a task in of itself to work with and maintain

```
"RuleAction" : "allow",
"Egress" : "false",
"CidrBlock" : "0.0.0.0/0",
"PortRange" : {"From" : "22", "To" : "22"}
},
"InboundResponsePortsNetworkAclEntry" : {
  "Type" : "AWS::EC2::NetworkAclEntry",
  "Properties" : {
    "NetworkAclId" : {"Ref" : "NetworkAcl"},
    "RuleNumber" : "102",
    "Protocol" : "6",
    "RuleAction" : "allow",
    "Egress" : "false",
    "CidrBlock" : "0.0.0.0/0",
    "PortRange" : {"From" : "1024", "To" : "65535"}
  }
},
"OutboundHTTPNetworkAclEntry" : {
  "Type" : "AWS::EC2::NetworkAclEntry",
  "Properties" : {
    "NetworkAclId" : {"Ref" : "NetworkAcl"},
    "RuleNumber" : "100",
    "Protocol" : "6",
    "RuleAction" : "allow",
    "Egress" : "true",
    "CidrBlock" : "0.0.0.0/0",
    "PortRange" : {"From" : "80", "To" : "80"}
  }
},
"OutBoundHTTPSPortNetworkAclEntry" : {
  "Type" : "AWS::EC2::NetworkAclEntry",
  "Properties" : {
    "NetworkAclId" : {"Ref" : "NetworkAcl"},
    "RuleNumber" : "101",
    "Protocol" : "6",
    "RuleAction" : "allow",
    "Egress" : "true",
    "CidrBlock" : "0.0.0.0/0",
    "PortRange" : {"From" : "443", "To" : "443"}
  }
},
"OutBoundResponsePortNetworkAclEntry" : {
  "Type" : "AWS::EC2::NetworkAclEntry",
  "Properties" : {
    "NetworkAclId" : {"Ref" : "NetworkAcl"},
    "RuleNumber" : "102",
    "Protocol" : "6",
    "RuleAction" : "allow",
    "Egress" : "true",
    "CidrBlock" : "0.0.0.0/0",
    "PortRange" : {"From" : "1024", "To" : "65535"}
  }
},
"SubnetNetworkAclAssociation" : {
  "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
  "Properties" : {
    "SubnetId" : {"Ref" : "Subnet"},
    "NetworkAclId" : {"Ref" : "NetworkAcl"}
  }
},
"IPAddresses" : {
  "Type" : "AWS::EC2::EIP",
  "DependsOn" : "AttachGateway",
  "Properties" : {
    "Domain" : "vpc",
    "InstanceId" : {"Ref" : "WebServerInstance"}
  }
},
"InstanceSecurityGroup" : {
  "Type" : "AWS::EC2::SecurityGroup",
  "Properties" : {
    "VpcId" : {"Ref" : "VPC"},
    "GroupDescription" : "Enable SSH access via port 22",
    "SecurityGroupIngress" : [
      {"IpProtocol" : "tcp", "FromPort" : "22", "ToPort" : "22", "CidrIp" : {"Ref" : "SSHLocation"}},
      {"IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0"}
    ]
  }
},
```



Terraform - Hashicorp

The one consistent problem is that all these solutions have been domain specific. Terraform however takes an alternate tack, by using pluggable modules called providers, it allows you to implement IaC on multiple public clouds, private clouds and indeed many traditional virtualisation systems

Archive	DNSMadeEasy	Logentries	StatusCake
Atlas	DNSimple	Mailgun	SoftLayer
AWS	Docker	Microsoft Azure	Scaleway
Chef	Dyn	MySQL	Template
CenturyLinkCloud	GitHub	OpenStack	Terraform
CloudFlare	Fastly	Packet	TLS
CloudStack	Google Cloud	PostgreSQL	Triton
Cobbler	Grafana	PowerDNS	UltraDNS
Consul	Heroku	RabbitMQ	vCloud Director
Datadog	InfluxDB	Random	vSphere
DigitalOcean	Librato	Rundeck	

Terraform - The Demo

Let's build a simple development server

```
provider "aws" {
    region           = "${var.aws_region}"
    shared_credentials_file = "/home/ben/.aws/pers"
}

data "aws_ami" "list" {
    most_recent = true
    filter {
        name = "tag:Author"
        values = ["Ben"]
    }
}

resource "aws_instance" "test2" {
    count = 1
    ami   = "${data.aws_ami.list.id}"
    vpc_security_group_ids = ["sg-80a0f4e7"]
    subnet_id = "subnet-d18f60b5"
    key_name = "bentest"
    instance_type = "t2.micro"
    availability_zone = "${var.aws_region}a"
    tags = {
        Name = "Terraformed-${count.index}"
    }
}
```

Terraform - The Demo

The first stanza is the provider which tells terraform what API's to talk to with what permissions

You can have multiple providers in a configuration, so AWS+Azure, AWS+Chef and so on...

```
|provider "aws" {  
    region      = "${var.aws_region}"  
    shared_credentials_file = "/home/ben/.aws/pers"  
}
```

Terraform - The Demo

This is a datasource that lets me query the provider to find a particular value.

In this case I'm searching for an AMI with a Tag 'Author' and a value 'Ben'

Names, ID's and similar can also be used with wildcards

```
data "aws_ami" "list" {
  most_recent = true
  filter {
    name = "tag:Author"
    values = ["Ben"]
  }
}
```

Terraform - The Demo

And finally the resource, or what we want to create, here it's a t2.micro instance using the AMI I searched for with the datasource called test2

I also really like tagging things!

```
resource "aws_instance" "test2" {
    count = 1
    ami   = "${data.aws_ami.list.id}"
    vpc_security_group_ids = ["sg-80a0f4e7"]
    subnet_id = "subnet-d18f60b5"
    key_name = "bentest"
    instance_type = "t2.micro"
    availability_zone = "${var.aws_region}a"
    tags = {
        Name = "Terraformed-${count.index}"
    }
}
```

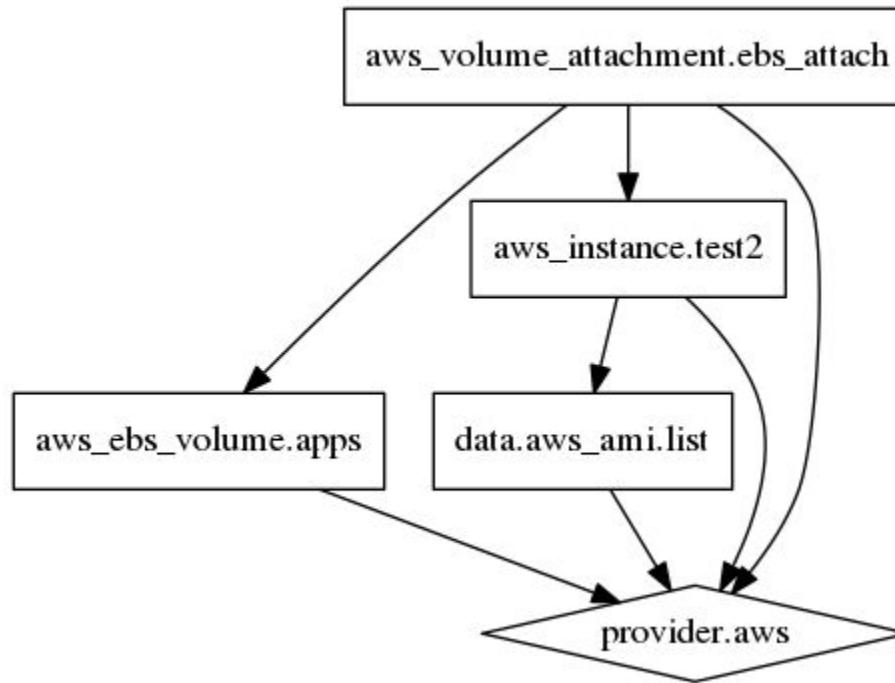
Terraform - The Demo

<http://asciinema.org/a/3ax8ecdrgp18qhmnrk6k2b24z>

Terraform - Change and Destroy

<http://asciinema.org/a/eo50zdfhb15cqw6d55g3zy9>

Terraform - Graph

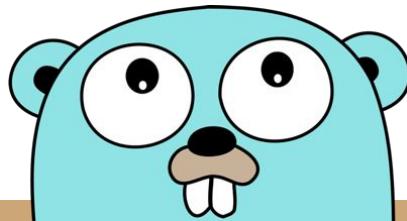


Terraform - So it's like Git

Yes, and No, but let's gloss over that...

The key part is that internally (development is done in golang, so the concurrency model is awesome by default) Terraform generates a graph of your resources.

It by defaults generates 10 resources in a parallel fashion, but the graph itself determines the dependencies and builds everything appropriately



THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



Terraform - The Downside

Terraform maintains a state file (`terraform.tfstate`) that describes what the environment is, therefore although the recipe can be easily shared, anyone managing the environment needs to share this file as well

Although it can be shared by git (and a backend system is in 0.9 beyond Atlas), there is the option of the ‘`terraform refresh`’ to regenerate the state files (or re-align them if any changes are made outside terraform)

There are however several solutions to this issue (DynamoDB...)

Terraform - The Future State

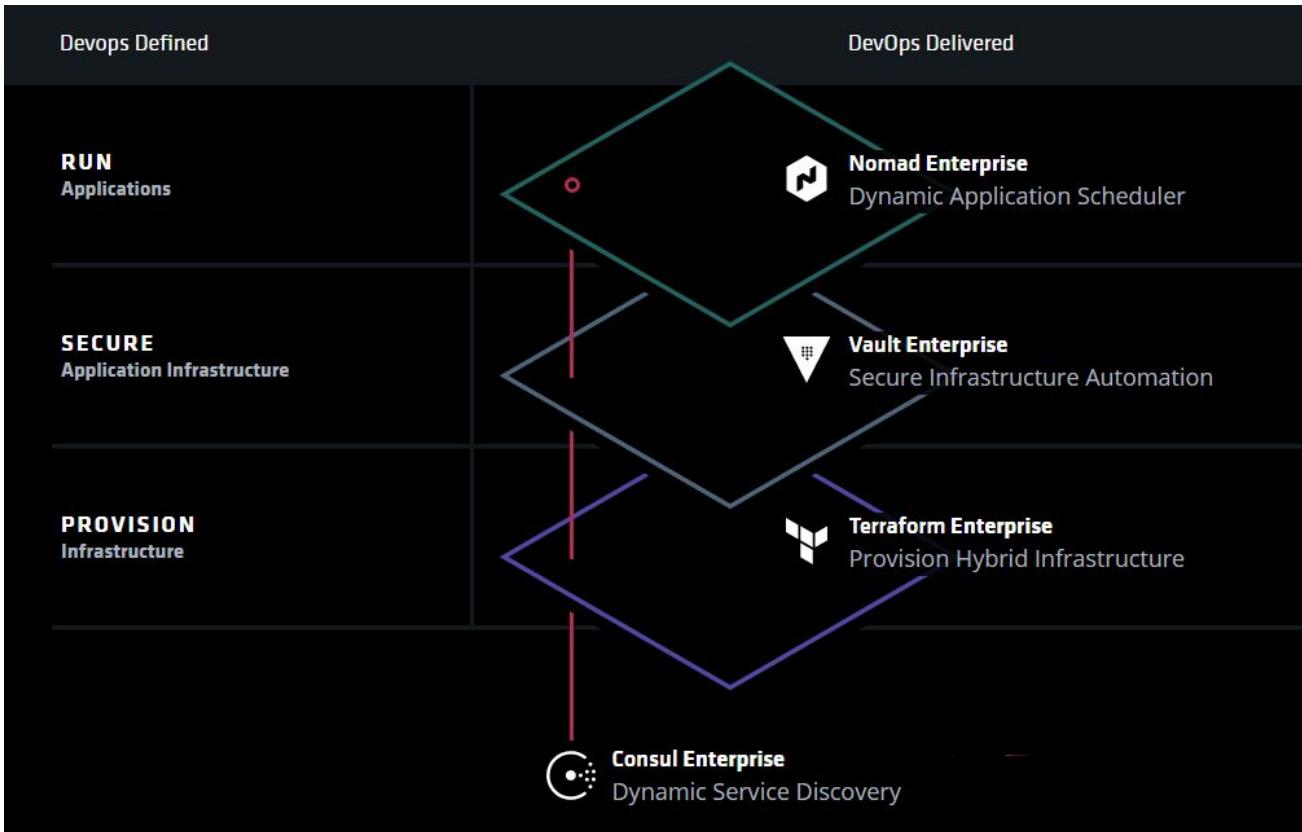
0.9 is now in beta - with backend support, more datasources and resources

Hashicorp are now concentrating on 'Enterprise' implementations

Expect more around compliance to specific standards along with
collaborative development on infrastructure

Much better multi-environment support with pipeline integration (change
requests anyone?)

Hashicorp - Devops Do as opposed to Are



Hashicorp - Just Generally Awesome

PROVISION



Vagrant

Create and configure portable development environments



Packer

Create platform specific machine images from a single source



Terraform

Create, combine and manage infrastructure across multiple providers

SECURE



Vault

Centrally store, secure and control access to distributed secrets

RUN



Nomad

Cluster manager and scheduler to deploy applications across any infrastructure



Consul

Distributed highly available tool for service discovery, configuration and orchestration

This isn't meant to be a Hashicorp fanboy rant - but as a rule I've not found a single company that hasn't looked at their product set and gone

“Oh that's perfect!!!”