Testing Terraform

Nathen Harvey
Chef

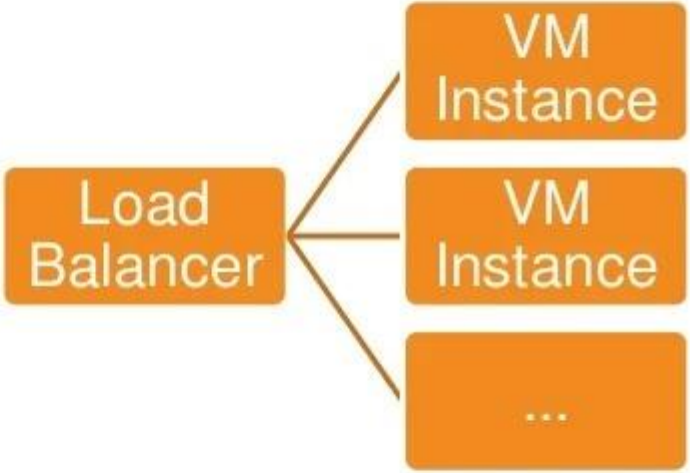
What am I talking about

- Suppose you use a tool like Terraform to create your infrastructure
- You're onboard with infrastructure as code
- Code should be tested
- How can we test Terraform?

HashiCorp
Terraform

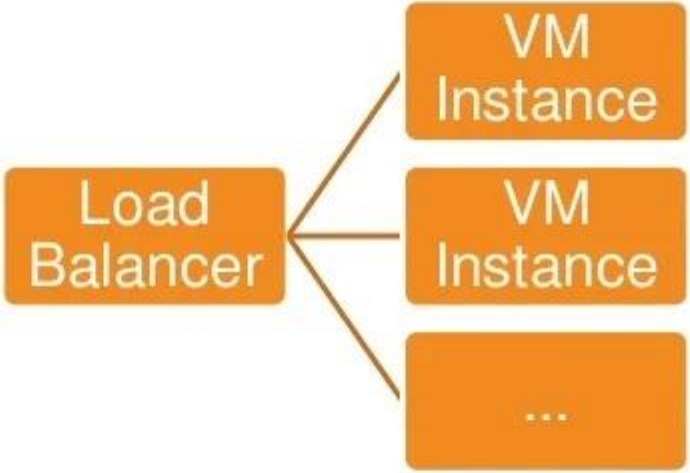
A Motivational Example

1. Deploy n-tier cloud infra
2. Deploy some app
3. Profit



A Motivational Example

1. Deploy n-tier cloud infra
2. ~~Deploy some app~~ ???
3. Profit



So how do you do that in Terraform?

Terraform on AWS: Define an Instance (VM)

```
resource "aws_instance" "web" {
  count         = 3
  instance_type = "t2.micro"
  ami           = "${lookup(var.aws_amis, var.aws_region)}"
  key_name      = "${var.key_name}"
  subnet_id     = "${aws_subnet.default.id}"

  vpc_security_group_ids = [
    "${aws_security_group.lb_to_webservers.id}",
    "${aws_security_group.ssh_from_office.id}",
  ]
  ...
}
```

Terraform on AWS: Define Load Balancer (ELB)

```
resource "aws_elb" "web" {
  name = "testing-terraform-elb"
  subnets = ["${aws_subnet.default.id}"]

  security_groups = [
    "${aws_security_group.world_to_elb.id}",
    "${aws_security_group.lb_to_webservers.id}",
    ]

  instances = ["${aws_instance.web.*.id}"]

  listener {
      instance_port     = 80
    instance_protocol = "http"
      lb_port           = 80
      lb_protocol       = "http"
    }
}
```

Terraform on AWS: Define a Firewall Rule

```
resource "aws_security_group" "ssh_from_office" {
  name   = "testing-terraform-ssh"
  vpc_id = "${aws_vpc.testing_terraform.id}"

  # SSH access from special office addresses
  ingress {
      from_port   = 22
      to_port     = 22
      protocol    = "tcp"
    cidr_blocks = "${var.ssh_cidrs}"
    }

  # outbound internet access
  egress {
      from_port   = 0
      to_port     = 0
      protocol    = "-1"
      cidr_blocks = ["0.0.0.0/0"]
    }
}
```

Terraform on AWS: Define a Firewall Rule

```
resource "aws_security_group" "ssh_from_office" {
 name   = "testing-terraform-ssh"
 vpc_id = "${aws_vpc.testing_terraform.id}"

 # SSH access from special office addresses
 ingress {
     from_port   = 22
     to_port     = 22
     protocol    = "tcp"
  cidr_blocks = "${var.ssh_cidrs}"
  }

 # outbound internet access
 egress {
     from_port   = 0
     to_port     = 0
     protocol    = "-1"
     cidr_blocks = ["0.0.0.0/0"]
  }
}
```

```
variable "ssh_cidrs" {
  default = [
    "173.239.212.22/32",
    "12.130.117.75/32"
  ]
}
```

What's the plan, Phil?

```
$ terraform plan -out plan.out
```

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
...
Plan: 11 to add, 0 to change, 0 to destroy.


------------------------------------------------------------------------


This plan was saved to: plan.out


To perform exactly these actions, run the following command to apply:
```

Make it so!

```
$ terraform apply "plan.out"
```

```
aws_vpc.testing_terraform: Creating...
  assign_generated_ipv6_cidr_block: "" => "false"
  cidr_block:                       "" => "10.0.0.0/16"
  default_network_acl_id:           "" => "<computed>"
  default_route_table_id:           "" => "<computed>"
  default_security_group_id:        "" => "<computed>”
...
aws_elb.web: Creation complete after 25s (ID: testing-terraform-elb)

Apply complete! Resources: 11 added, 0 changed, 0 destroyed.

Outputs:

address = testing-terraform-elb-1345186905.us-east-1.elb.amazonaws.com
```

It's alive!

testing-terraform-elb-1345186905.us-east-1.elb.amazonaws.com

Welcome to nginx!

If you see this page, the nginx web server is successfully installed a
working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Part of a balanced diet

Load balancer: testing-terraform-elb

Description | **Instances** | Health Check | Listeners | Monitoring | Tags | Migration

Connection Draining: Disabled (Edit)

Edit Instances

Instance ID	Name	Availability Zone	Status	Actions
i-05e4a08dd80843d35	Testing Terraform - web-1	us-east-1f	InService	Remove from Load Balancer
i-03663e4b9cb2858d6	Testing Terraform - web-0	us-east-1f	InService	Remove from Load Balancer
i-0c5a9e39db62b555e	Testing Terraform - web-2	us-east-1f	InService	Remove from Load Balancer

Verify with Terraform

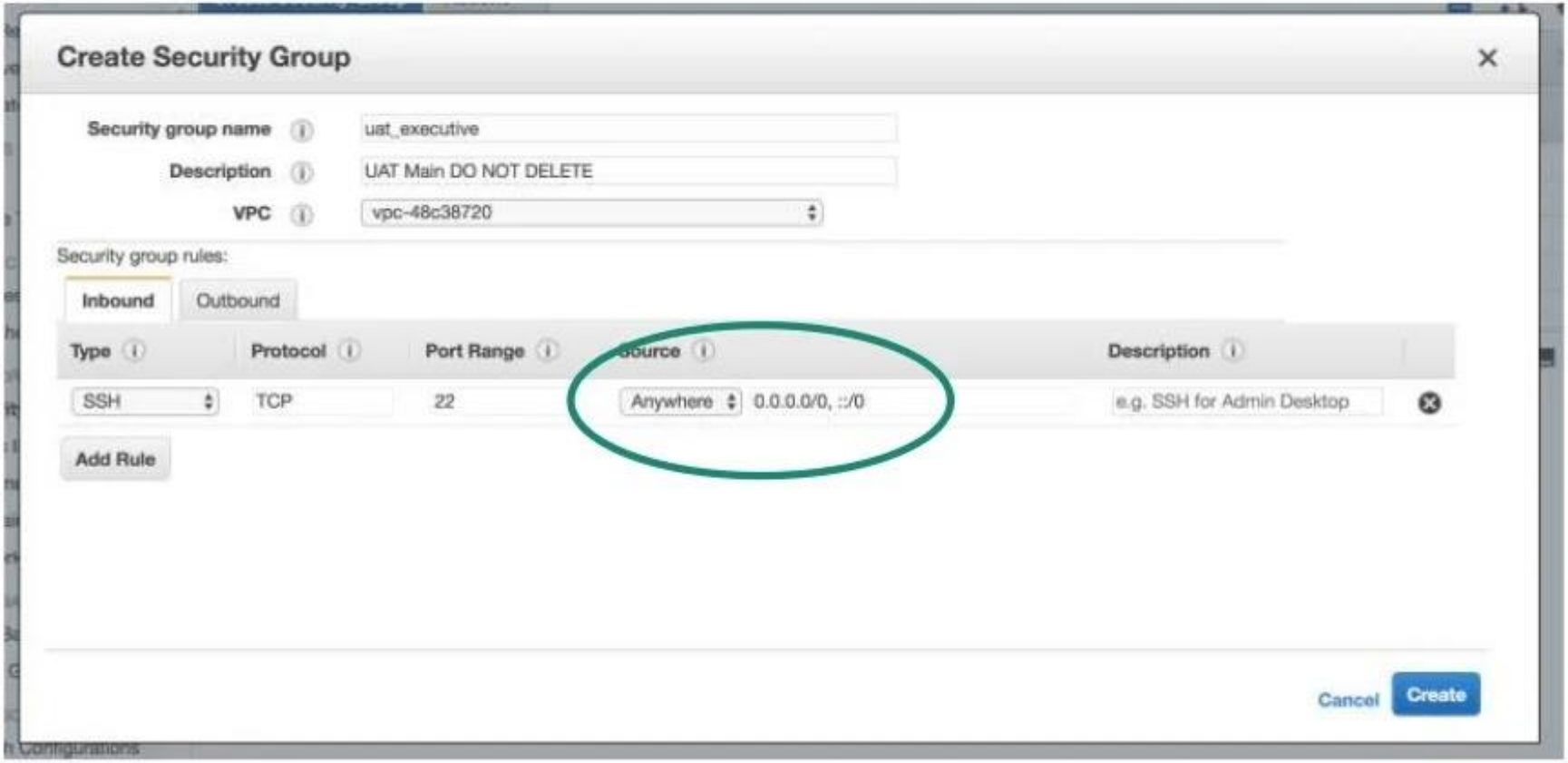
```
$ terraform plan
```

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

aws_vpc.testing_terraform: Refreshing state... (ID: vpc-091427c0ca8ed3c29)
...
aws_elb.web: Refreshing state... (ID: testing-terraform-elb)

------------------------------------------------------------------------

No changes. Infrastructure is up-to-date.

This means that Terraform did not detect any differences between your
configuration and real physical resources that exist. As a result, no
```

And all is well.

Meanwhile, at Team Shady….

An Innocent-Seeming Change…



An Innocent-Seeming Change…

Change Security Groups

Instance ID: i-03663e4b9cb2858d6
Interface ID: eni-0bc37fd4445a824c8

Select Security Group(s) to associate with your instance

	Security Group ID	Security Group Name	Description
☐	sg-003262589de21dbf5	default	default VPC security group
☑	sg-03bca2a0e861c01e7	testing-terraform-ssh	Managed by Terraform
☑	sg-01eefd0164f2de498	testing_terraform	Used in testing terraform
☐	[illegible]	testing_terraform_elb	World-facing ELB SG for Testing Terraform Presentation. nhar…
☑	sg-0c6388fa3e956cc15	uat_executive	UAT Main DO NOT DELETE

Cancel **Assign Security Groups**

Audit with Terraform

```
$ terraform plan
```

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  ~ aws_instance.web[0]
      vpc_security_group_ids.#:          "3" => "2"
      vpc_security_group_ids.1298918294: "sg-01eefd0164f2de498" => "sg-01eefd0164f2de498"
      vpc_security_group_ids.35497876:   "sg-0c6388fa3e956cc15" => ""
      vpc_security_group_ids.645334896:  "sg-03bca2a0e861c01e7" => "sg-03bca2a0e861c01e7”
```

Awesome! Terraform found the change!

Detected the new security group

Will detach it

Is that enough?

How does terraform plan fall short?

It only audits what is known in the Terraform config

All change is drift (considered bad)

Only express what you want,
not what you don't

We need something else.

INSPEC

BY CHEF

InSpec

Turn security and compliance into code

- Translate **compliance** into Code
- **Clearly** express statements of policy
- Move risk to build/test from **runtime**
- Find issues **early**
- Write code **quickly**
- Run code **anywhere**
- **Inspect** machines, data and APIs

INSP

PART OF A PROCESS OF CONTINUOUS COMPLIA

Scan for Compliance | Build & Test Locally | Build & Test CI/CD | Remediate | Ver

A SIMPLE EXAMPLE OF AN INSPEC CIS RULE

```
control 'cis-1.4.1' do
  title '1.4.1 Enable SELinux in /etc/grub.conf'
  desc '
    Do not disable SELinux and enforcing in your GRUB
configuration. These are important security features that
prevent attackers from escalating their access to your syste
For reference see …
    '

  impact 1.0
  expect(grub_conf.param 'selinux').to_not eq '0'
  expect(grub_conf.param 'enforcing').to_not eq '0'
```

InSpec Shell

```
$ inspec shell -t aws://
```

```
Welcome to the interactive InSpec Shell
To find out how to use it, type: help

You are currently running on:

    Name:      aws
    Families:  cloud, api
    Release:   aws-sdk-v2.11.50
```

InSpec Shell

```
$ inspec shell -t aws://

Welcome to the interactive InSpec Shell
To find out how to use it, type: help

You are currently running on:

    Name:       aws
    Families:   cloud, api
    Release:    aws-sdk-v2.11.50
```

Targeting AWS!

InSpec Shell – Tab Completion

```
inspec> aws_<TAB>
aws_cloudtrail_trail              aws_iam_access_keys              aws_iam_users                    aws_security_group
aws_cloudtrail_trails             aws_iam_group                    aws_kms_key                      aws_security_groups
aws_cloudwatch_alarm              aws_iam_groups                   aws_kms_keys                     aws_sns_subscription
aws_cloudwatch_log_metric_filter  aws_iam_password_policy          aws_rds_instance                 aws_sns_topic
aws_config_delivery_channel       aws_iam_policies                 aws_route_table                  aws_sns_topics
aws_config_recorder               aws_iam_policy                   aws_route_tables                 aws_subnet
aws_ec2_instance                  aws_iam_role                     aws_s3_bucket                    aws_subnets
aws_ec2_instances                 aws_iam_root_user                aws_s3_bucket_object             aws_vpc
aws_iam_access_key                aws_iam_user                     aws_s3_buckets                   aws_vpcs
```

Find the shady security group

```
inspec> aws_security_group('sg-0c6388fa3e956cc15').group_name
```

```
=> "uat_executive"
```

Check the Rules

```
inspec> aws_security_group('sg-0c6388fa3e956cc15').inbound_rules
```

```
=> [{:from_port=>22,
  :ip_protocol=>"tcp",
  :ip_ranges=>[{:cidr_ip=>"0.0.0.0/0"}],
  :ipv_6_ranges=>[{:cidr_ipv_6=>"::/0"}],
  :prefix_list_ids=>[],
  :to_port=>22,
  :user_id_group_pairs=>[]}]
```

Write a proper test

```
inspec> describe aws_security_group('sg-0c6388fa3e956cc15') do
inspec>   it { should_not allow_in ipv4_range: '0.0.0.0/0' }
inspec> end
```

Write a proper test

```
inspec> describe aws_security_group('sg-0c6388fa3e956cc15') do
inspec>   it { should_not allow_in ipv4_range: '0.0.0.0/0' }
inspec> end

Profile: inspec-shell
Version: (not specified)

  EC2 Security Group sg-0c6388fa3e956cc15
     ×  should not allow in {}
     expected `EC2 Security Group sg-0c6388fa3e956cc15.allow_in?({})` to
  return false, got true

Test Summary: 0 successful, 1 failure, 0 skipped
```

An InSpec Control

security_groups.rb

```
control 'Make sure unrestricted SSH is not permitted' do
  # Loop over each of the security group IDs in the region
  aws_security_groups.group_ids.each do |group_id|
    # Examine a security group in detail
    describe aws_security_group(group_id) do
      # Examine Ingress rules, and complain if
      # there is unrestricted SSH
      it { should_not allow_in(port: 22, ipv4_range: '0.0.0.0/0') }
    end
  end
end
```

InSpec Shell

```
$ inspec exec -t aws:// secruity_groups.rb
```

```
Profile: tests from security_groups.rb (tests from security_groups.rb)
Version: (not specified)
Target:  aws://


  ×  Make sure unrestricted SSH is not permitted: EC2 Security Group sg-003262589de21dbf5 (15
failed)
     ✓  EC2 Security Group sg-003262589de21dbf5 should not allow in {}
     ✓  EC2 Security Group sg-00f0b54bd7c8d93a1 should not allow in {}
     ×  EC2 Security Group sg-0c6388fa3e956cc15 should not allow in {}
     expected `EC2 Security Group sg-0c6388fa3e956cc15.allow_in?({})` to return false, got
true
...


Profile Summary: 0 successful controls, 1 control failure, 0 controls skipped
```

Where to next?

What else to validate?

Concern	Terraform Plan	InSpec
Number and type of instances requested	✓	✓
All instances are part of our app		✓
Right security groups created and attached	✓	✓
No security group allows in SSH		✓
Only security group open to the world should be port 80 for the ELB		✓
ELB correctly configured	✓	✓
Should only be one ELB		✓

Check instances

instances.rb

```
control "Should only have instances associated with my app" do
  aws_ec2_instances.instance_ids.each do |instance_id|
    describe aws_ec2_instance(instance_id) do
      its('instance_type') { should cmp 't2.micro' }
      its('tags') { should include(key:'X-Application', value:'Testing Terraform') }
      end
    end
end
```

More Security Groups

security_groups.rb

```
control "The only world-open security groups should be on the ELB" do
  elb_sg_ids = aws_elbs.security_group_ids
  aws_security_groups.group_ids.each do |sg_id|
    sg = aws_security_group(sg_id)
    if sg.allow_in? ipv4_range: '0.0.0.0/0'
      describe sg do
        its('group_id') { should be_in elb_sg_ids }
        it { should allow_in_only port: 80 }
          end
        end
    end
end
```

Put it all together with a profile

```
$ inspec init profile my_app
```

```
Create new profile at /Users/nathenharvey/projects/nathenharvey/testing-
terraform/inspec/profiles/my_app
 * Create directory libraries
 * Create file README.md
 * Create directory controls
 * Create file controls/example.rb
 * Create file inspec.yml
 * Create file libraries/.gitkeep
```

Put it all together with a profile

```
$ inspec exec -t aws:// my_app
```

```
Profile: tests from security_groups.rb (tests from security_groups.rb)

Version: (not specified)

Target:  aws://


  ×  Make sure unrestricted SSH is not permitted: EC2 Security Group sg-003262589de21dbf5 (15
failed)

     ✓  EC2 Security Group sg-003262589de21dbf5 should not allow in {}

     ✓  EC2 Security Group sg-00f0b54bd7c8d93a1 should not allow in {}

     ×  EC2 Security Group sg-01b504b800f32e1ff should not allow in {}

     expected `EC2 Security Group sg-01b504b800f32e1ff.allow_in?({})` to return false, got
true

...

Profile Summary: 0 successful controls, 3 control failures, 0 controls skipped
```

Terraform and InSpec?

Terraform: A Power Tool for the Cloud!

InSpec: A Verification tool for OS's and

Terraform and InSpec!

Terraform: A Power Tool for the Cloud!

InSpec: A Verification tool for OS's and

Bonus Round

Iggy

Install InSpec-Iggy

```
$   gem install inspec-iggy
```

```
Successfully installed inspec-iggy-0.2.0
1 gem installed
```

Create a Profile from tfstate file

```
$ inspec terraform generate --tfstate terraform.tfstate > my_terra.rb
```

Run InSpec

```
$   inspec exec -t aws:// my_terra.rb
```

```
inspec exec -t aws:// my_terra.rb

Profile: tests from my_terra.rb (tests from my_terra.rb)
Version: (not specified)
Target:  aws://

  ✓  aws_ec2_instance::i-03663e4b9cb2858d6: Iggy terraform.tfstate
  aws_ec2_instance::i-03663e4b9cb2858d6
     ✓  EC2 Instance i-03663e4b9cb2858d6 should exist
...
Profile Summary: 8 successful controls, 0 control failures, 0 controls skippe
Test Summary: 32 successful, 0 failures, 0 skipped
```

Join us on Slack!

- http://community-slack.chef.io
- `#general` (for Chef stuff)
- `#inspec`

What questions can I answer for you?

https://github.com/nathenharvey/testing-terraform

CHEF™