Deploying Services to AWS Without Leaving the Comfort of your Desktop

Matt Carlton

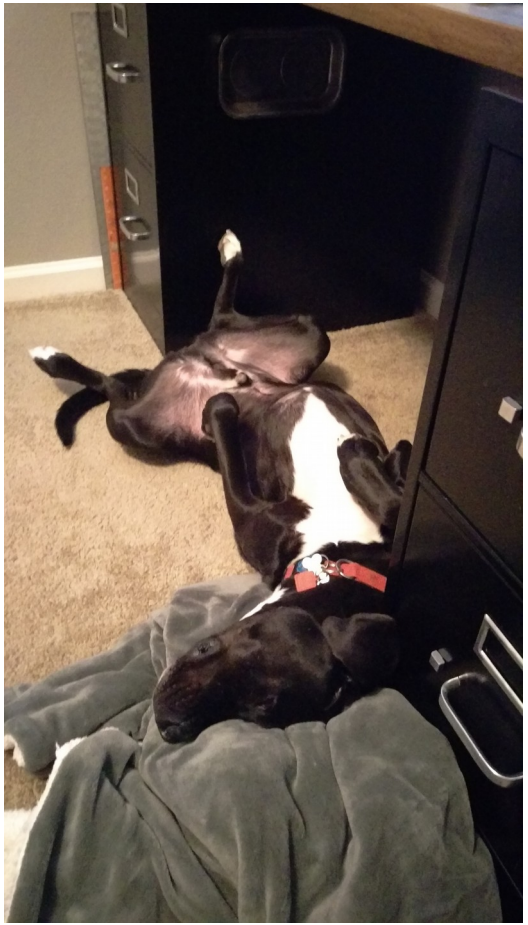
@mcarlton00

matt@sixfeetup.com

About Me

- Purdue University – CIT
- Six Feet Up – Sysadmin/DevOps Engineer
- Open Source Enthusiast
- Maker/tinkerer

- Usually trying to hide in the background



The Problem

- Managing servers by hand isn’t fun
- Replicating setups
- Cloud deployments are hard
- No good “do it all” tool available

Tools

SALTSTACK

Pros

- Configuration management
- Automating tasks
- Inventory/Metadata collection
- Centralized control point

Cons

- Infrastructure Support is Lacking
- Requires a “Master” server to get full benefits

HashiCorp Terraform

Pros

- Infrastructure as code
- Can be managed anywhere
- Multi-cloud potential

Cons

- Server configurations


SALTSTACK
HashiCorp
Terraform
Now kiiiiiiiiss
CN
CARTOON NETWORK

The Plan

- Use Terraform to deploy AWS infrastructure
  - VPC, EC2 Instances, s3 buckets, RDS instances, etc
- Use Saltstack to push content to servers and manage services
  - User accounts
  - Package installation
  - Code deployments

The Best Part

You can do this all from your local computer.
No more manually logging into servers!

Terraform
- File Format

```
resource "aws_vpc" "main" {
  cidr_block = var.base_cidr_block
}

<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  # Block body
  <IDENTIFIER> = <EXPRESSION> # Argument
}
```

https://www.terraform.io/docs/configuration/index.html

Terraform
- File Structure

- Backend – Specifies we're using AWS and which region we're connecting to
- Instances – EC2 Instances
- Network – VPC and Security Groups
- Roles – IAM Roles
- Templates – User data scripts
- Variables – Your main control point

```
.
├── backend.tf
├── instances.tf
├── network.tf
├── roles.tf
├── templates
│   ├── master-userdata.tmpl
│   └── minion-userdata.tmpl
└── variables.tf
```

Terraform - Variables

- Control point
- Define information about your servers here
- Ideally, you only edit this file and not the rest of your modules

```
# Instance sizes, defaults to t3.small
variable "instance_size" {
  type = "map"
  default = {
    plone-app01 = "t3.medium"
  }
}

# The AMI to use for each instance.  Defaults to Amazon Linux 2
variable "instance_ami" {
  type = "map"
  default = {
    plone-app01 = "ami-0a313d6098716f372" # Ubuntu 18.04
    static-site = "ami-0de53d8956e8dcf80" # Amazon Linux 2
  }
}

# Determines what type of server is created
variable "role" {
  type = "map"
  default = {
    static-site = "static"
    plone-app01 = "plone"
  }
}
```

Terraform - Network

Required network services for a simple AWS deployment:

- VPC (+ IP range)
- Subnets (+ IP ranges inside your VPC)
- Internet Gateway
- Routing Tables
- Security Groups
- NAT Gateway (if using a private subnet)

```
# Create security group for web traffic
resource "aws_security_group" "web-acl" {
  name = "web-acl"
  description = "Web access to instances"
  vpc_id = "${aws_vpc.demo-vpc.id}"
  ingress {
    from_port = "80"
    to_port = "80"
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port = "443"
    to_port = "443"
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port = "0"
    to_port = "0"
    protocol = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

You ***must*** have an egress on at least one security group on each instance

Terraform - Roles

- Create a custom role to allow servers to read EC2 tags
- Saltstack will use these tags to determine how to interact with each server.

```
20
21 # Create an IAM policy allowing EC2 instances to see tags
22 resource "aws_iam_policy" "ec2_tags_policy" {
23   name = "ec2_tags_policy"
24   policy = <<EOF
25 {
26   "Version": "2012-10-17",
27   "Statement": [
28     {
29       "Effect": "Allow",
30       "Action": [
31         "ec2:DescribeTags"
32       ],
33       "Resource": "*"
34     }
35   ]
36 }
37 EOF
38 }
39
40 # Attach the Role and Policy into a single entity
41 resource "aws_iam_policy_attachment" "ec2-tags-attach" {
42   name       = "ec2-tags-attach"
43   roles      = ["${aws_iam_role.read_ec2_tags.name}"]
44   policy_arn = "${aws_iam_policy.ec2_tags_policy.arn}"
45 }
46
47 # Allow the roles to be assigned to an EC2 instance
48 resource "aws_iam_instance_profile" "read_ec2_tags" {
49   name = "read_ec2_tags"
50   role = "${aws_iam_role.read_ec2_tags.name}"
51 }
```

Warning:
Beware of red herrings!

Terraform
- Servers (Salt Master)

```
# Create salt master
resource "aws_instance" "salt_master" {
  instance_type = "t3.small"
  key_name = "cloud-conf-demo"
  ami = "ami-0a313d6098716f372"
  iam_instance_profile = "${aws_iam_instance_profile.read_ec2_tags.id}"
  associate_public_ip_address = "True"
  availability_zone = "${var.region}a"
  subnet_id = "${aws_subnet.servers.id}"
  user_data = "${file("templates/master-userdata.tmpl")}"
  vpc_security_group_ids = [
    "${aws_security_group.salt-acl.id}",
    "${aws_security_group.ssh-acl.id}"
  ]
  root_block_device = {
    volume_size = 20
  }
  tags {
    Name = "salt-master"
    Managed = "terraform"
  }
}
```

```
# Checkout Salt states from github
git clone https://github.com/mcarlton00/aws-auto-deploy-demo.git /srv/

# Create Salt directory
mkdir -p /etc/salt/master.d
mkdir -p /etc/salt/minion.d

# Tell salt where to find states
cat <<CONFIG >> /etc/salt/master.d/custom.conf
file_roots:
  base:
    - /srv/saltstack/states
pillar_roots:
  base:
    - /srv/saltstack/pillar

auto_accept: True
CONFIG

# Tell Salt that we are the master
echo "master: 127.0.0.1" > /etc/salt/minion.d/master.conf

# Install salt master
wget -O /tmp/bootstrap-salt.sh https://bootstrap.saltstack.com
sh /tmp/bootstrap-salt.sh -M

# Sync the grains to get the ec2 tags
salt '*' saltutil.sync_grains
```

Use a bash script to set up the salt master

https://github.com/saltstack/salt-bootstrap

Terraform
- Servers (App Servers)

Almost all of those variables we defined way at the beginning are in use here.

Our control point between Terraform and Salt

```
# Create EC2 instances based on info from variables.tf
resource "aws_instance" "demo_servers" {
  instance_type = "${lookup(var.instance_size, element(var.vm_names, count.index), "t3.small")}"
  key_name = "cloud-conf-demo"
  count = "${length(var.vm_names)}"
  ami = "${lookup(var.instance_ami, element(var.vm_names, count.index), "ami-0de53d8956e8dcf80")}"
  availability_zone = "${var.region}a"
  subnet_id = "${aws_subnet.servers.id}"
  iam_instance_profile = "${aws_iam_instance_profile.read_ec2_tags.id}"
  user_data = "${data.template_file.init.rendered}"
  vpc_security_group_ids = [
    "${aws_security_group.web-acl.id}",
    "${aws_security_group.ssh-acl.id}"
  ]
  # Pull disk size from variables file
  root_block_device = {
    volume_size = "${lookup(var.data_disk_size, element(var.vm_names, count.index), 100)}"
  }
  tags {
    Name = "${element(var.vm_names, count.index)}"
    Managed = "terraform"
    Roles = "${lookup(var.role, element(var.vm_names, count.index))}"
  }
}

# Render the minion userdata script with the correct info
data "template_file" "init" {
  template = "${file("${path.module}/templates/minion-userdata.tmpl")}"
  vars = {
    salt_master = "${aws_instance.salt_master.private_ip}"
  }
}
```

Saltstack
- File Structure

- Pillar – A variable store
  - Should be your central control point
- States – What actually does the tasks
- top.sls – Determines what servers get what values
- map.jinja – Dictionary lookup for different OS values

```
.
├── pillar
│   ├── plone.sls
│   ├── static.sls
│   └── top.sls
├── states
│   ├── base
│   │   ├── init.sls
│   │   └── map.jinja
│   ├── map.jinja
│   ├── static
│   │   ├── files
│   │   │   └── nginx-static.conf
│   │   └── init.sls
└   └   └── top.sls
```

Saltstack
- Pillar and top.sls

```
static:
  repo: https://github.com/mcarlton00/aws-auto-deploy-demo.git
```

```
plone:
  repo: https://github.com/plone/simple-plone-buildout.git
  project_dir: /srv/plone
```

```
base:
  'ec2_roles:plone':
      - match: grain
      - plone
  'ec2_roles:static':
      - match: grain
      - static
```

Short and simple right now. Ways this can be expanded:

- URL to respond to
- Whether to generate a SSL cert
- Whether to include load balancing or caching
- Determining what user accounts should be on each type of server

The top file utilizes the Roles tag we gave to our servers to determine what operations need to be ran

Saltstack - Map file

- Great way to make your salt states compatible with multiple operating systems
- Easy to structure and reference in your code.

```
1 {% from "plone/map.jinja" import plone with context %}
2
3 plone-deps:
4   pkg.installed:
5     - pkgs:
6       - {{ plone.deps.freetype }}
7       - {{ plone.deps.png }}
8       - {{ plone.deps.jpeg }}
9       - {{ plone.deps.tiff }}
```

```
'RedHat': {
    'deps': {
        'freetype': 'freetype',
        'png': 'libpng-devel',
        'jpeg': 'libjpeg-turbo-devel',
        'tiff': 'libtiff-devel',
    }
},
'Debian': {
    'deps': {
        'freetype': 'libfreetype6-dev',
        'png': 'libpng16-dev',
        'jpeg': 'libjpeg8-dev',
        'tiff': 'libtiff5-dev',
    }
},
'FreeBSD': {
    'deps': {
        'freetype': 'freetype2',
        'png': 'png',
        'jpeg': 'openjpeg15',
        'tiff': 'tiff',
    }
```

Saltstack - Simple State

A Static HTML website

- Puts the nginx server block in place
- Creates a directory to use as the site root
- Checks out the latest code into this directory

```
{% from "map.jinja" import global with context %}
{% set static_pillar = salt['pillar.get']('static') %}

include:
  - nginx

static-nginx-config:
  file.managed:
    - name: {{ global.config_prefix }}/nginx/conf.d/static.conf
    - user: root
    - group: {{ global.group }}
    - mode: 664
    - makedirs: True
    - replace: False
    - source: salt://static/files/nginx-static.conf
    - template: jinja
    - require_in:
      - service: nginx

website-source:
  git.latest:
    - name: {{ static_pillar.repo }}
    - user: root
    - target: /srv
    - rev: master
    - branch: master
    - require_in:
      - service: nginx

source-directory:
  file.directory:
    - name: /srv/static-html
    - user: www-data
    - group: www-data
    - dir_mode: 775
    - require_in:
      - service: nginx
    - require:
      - git: website-source
```


PLEASE DEMO GODS
BE KIND!
memegenerator.net


LET US PRAY
TO THE DEMO GODS

No Limits

- As of this morning, there are 316 formulas publicly available on Salt's Github
- Formulas are pre-made state files. You just edit the pillar with your desired info

Questions?

- https://github.com/mcarlton00/aws-auto-deploy-demo
- https://learn.hashicorp.com/terraform/getting-started/install
- https://docs.saltstack.com/en/getstarted/
- https://github.com/saltstack/salt-bootstrap