CINTEO

https://cinteo.com
https://cinteo.com/en/jobs

12 factor infrastructure with terraform

2017, Stuttgart, Sergiu Bordeianu

The Twelve-Factor App

- 12factor.net

- Methodology for Dev and Ops to build and manage SaaS:
  - Code base - IaC
  - Build, release, run
  - Concurrency
  - …………….
  - Admin processes

Why terraform ?

Tools

- AWS CloudFormation
- Ansible
- Puppet
- terraform

Terraform

- Manage resource life cycle
- Integration with other providers and data sources
- Easy to integrate in DevOps process

Methodology
12 factor

Provisioning
Config Management
App Deployment
Continuous Delivery
Security &
Compliance
Orchestration

Modern infrastructure

- Low Maintenance
- High Security
- Cloud based ..

Engineers: We know what to do and how to do

JUST DO IT.

Project Management:
You don’t know how to work !

Code Base - Infrastructure as Code

- Machine Image(AMI)/App and service deployment
  - Packer template https://www.packer.io
  - One image one purpose : jenkins MI, DNS MI, NEXUS MI, VPN MI …
  - The same image in all environments
- Infrastructure
  - Terraform scripts
  - The same script in all environments with different parameters

Code Base - Infrastructure as Code

- Security
  - Terraform scripts: Network access list (ACL), security groups, ssh management, roles and users
- Compliance/Testing
  - Terraform template, terraform output and goss template
    - https://github.com/aelsabbahy/goss
- Config Management/Provisioning
  - Terraform scripts, template and variables

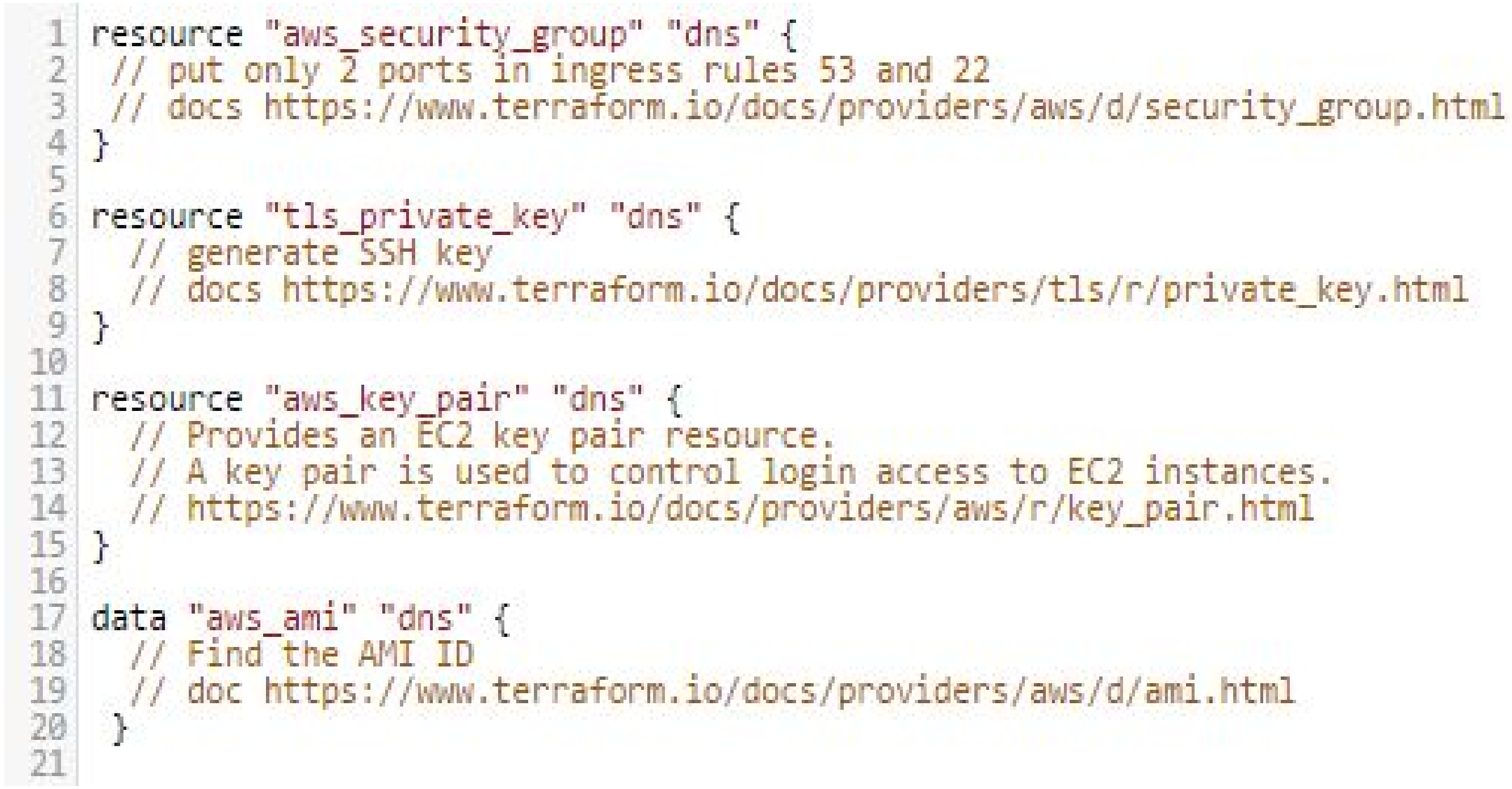
Deployment concepts

- Strong IP architecture planning
- Strong isolation between Dev, Testing and Prod areas
- Testing and Prod don’t have access at internet
- All is code - everything is a release
- Server components
    - Main disk and OS - immutable
    - Network interface - attached to the instance
    - External disk or persistent storage - attached to the instance

Deployment

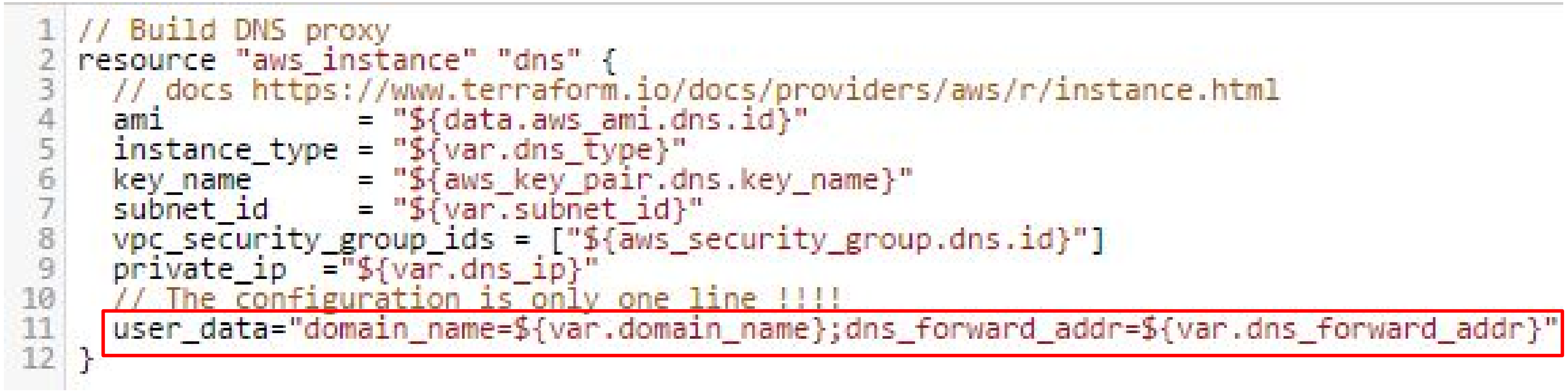
- Zero deployment, Zero update and reconfiguration
- Immutable infrastructure
- Deployment of a new version:
    - PreDeploy : update or (re)create infrastructure
    - Detach the disk/network interface
    - Destroy instance
    - Create new instance with new parameters
    - Attach the disk/network interface
    - PostDeploy: update or (re)create infrastructure
- All the tasks are handled by terraform

Deployment code



```
resource "aws_security_group" "dns" {
 // put only 2 ports in ingress rules 53 and 22
 // docs https://www.terraform.io/docs/providers/aws/d/security_group.html
}

resource "tls_private_key" "dns" {
  // generate SSH key
  // docs https://www.terraform.io/docs/providers/tls/r/private_key.html
}

resource "aws_key_pair" "dns" {
  // Provides an EC2 key pair resource.
  // A key pair is used to control login access to EC2 instances.
  // https://www.terraform.io/docs/providers/aws/r/key_pair.html
}

data "aws_ami" "dns" {
  // Find the AMI ID
  // doc https://www.terraform.io/docs/providers/aws/d/ami.html
 }
```

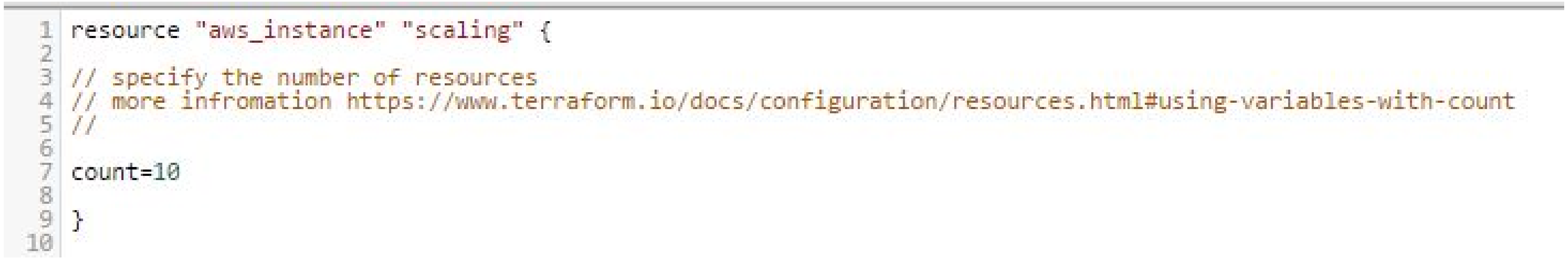
Deployment code



```
// Build DNS proxy
resource "aws_instance" "dns" {
  // docs https://www.terraform.io/docs/providers/aws/r/instance.html
  ami             = "${data.aws_ami.dns.id}"
  instance_type = "${var.dns_type}"
  key_name        = "${aws_key_pair.dns.key_name}"
  subnet_id       = "${var.subnet_id}"
  vpc_security_group_ids = ["${aws_security_group.dns.id}"]
  private_ip  ="${var.dns_ip}"
  // The configuration is only one line !!!!
  user_data="domain_name=${var.domain_name};dns_forward_addr=${var.dns_forward_addr}"
}
```

Terraform scaling

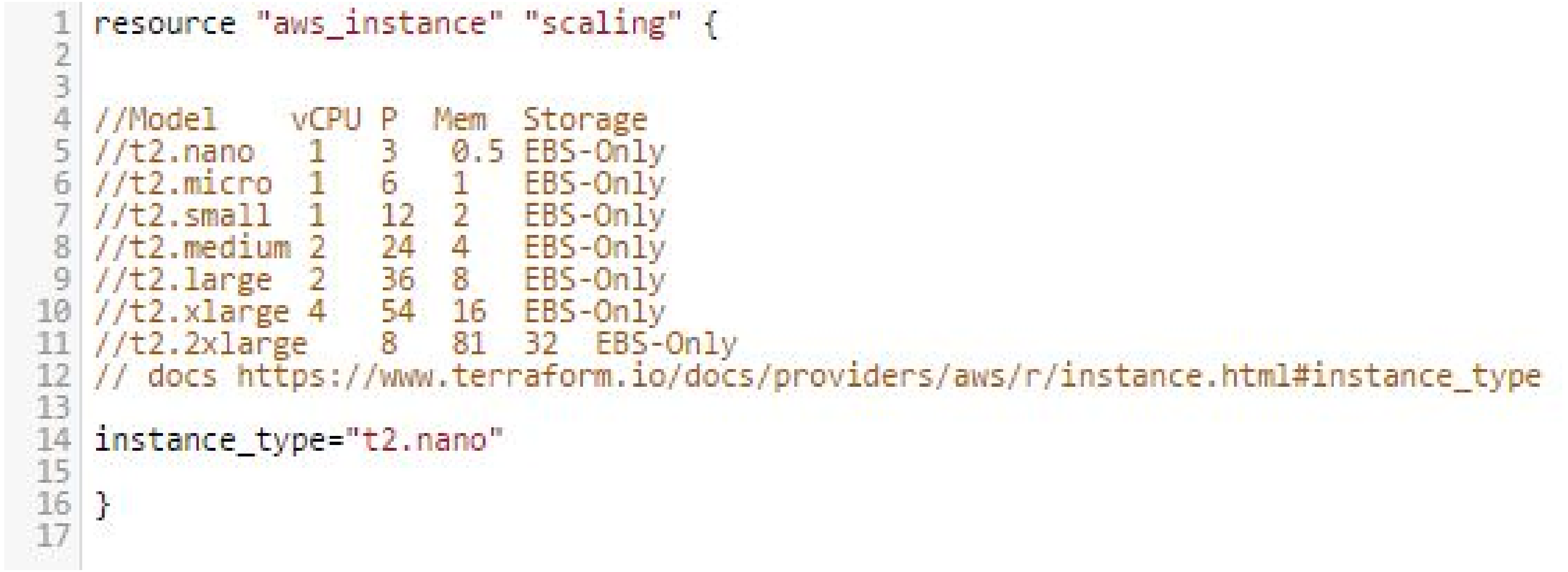
- Horizontal scaling
  - Using load balancers or other mecanisme from cloud providers
  - No code efforts just change `count=n`



```
resource "aws_instance" "scaling" {

// specify the number of resources
// more infromation https://www.terraform.io/docs/configuration/resources.html#using-variables-with-count
//

count=10

}

```

Terraform scaling

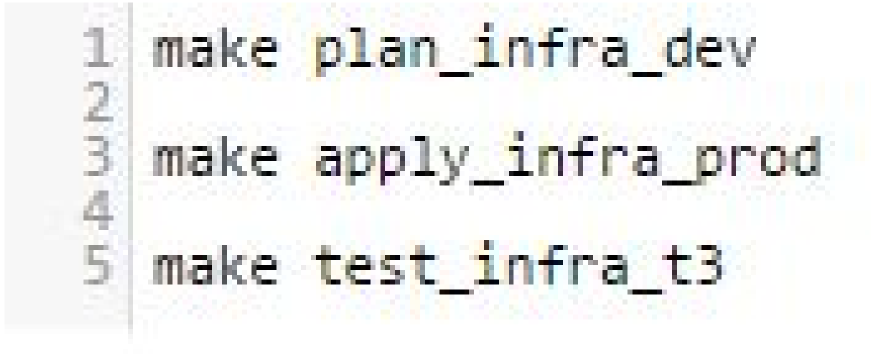
- Vertical scaling
  - Attaching and detaching the network interface
  - No code efforts just change the `instance_type=”large-instance”`



```
resource "aws_instance" "scaling" {


//Model    vCPU P  Mem  Storage
//t2.nano   1   3   0.5 EBS-Only
//t2.micro  1   6   1   EBS-Only
//t2.small  1   12  2   EBS-Only
//t2.medium 2   24  4   EBS-Only
//t2.large  2   36  8   EBS-Only
//t2.xlarge 4   54  16  EBS-Only
//t2.2xlarge    8   81  32  EBS-Only
// docs https://www.terraform.io/docs/providers/aws/r/instance.html#instance_type

instance_type="t2.nano"

}
```

Admin processes

- Easy to invoke
- All team members can use it
- Using `gnu make` like a launch wrapper



```
make plan_infra_dev

make apply_infra_prod

make test_infra_t3
```

Request
a Demo

Deduction

- Keep terraform code as simple as possible
- More security for terraform state
- The gap between Dev and Ops is still **big**
- A lot of tools and products are not **cloud native**
- Remote state management available in Atlas

Questions !

CINTEO

https://cinteo.com/

https://cinteo.com/en/jobs