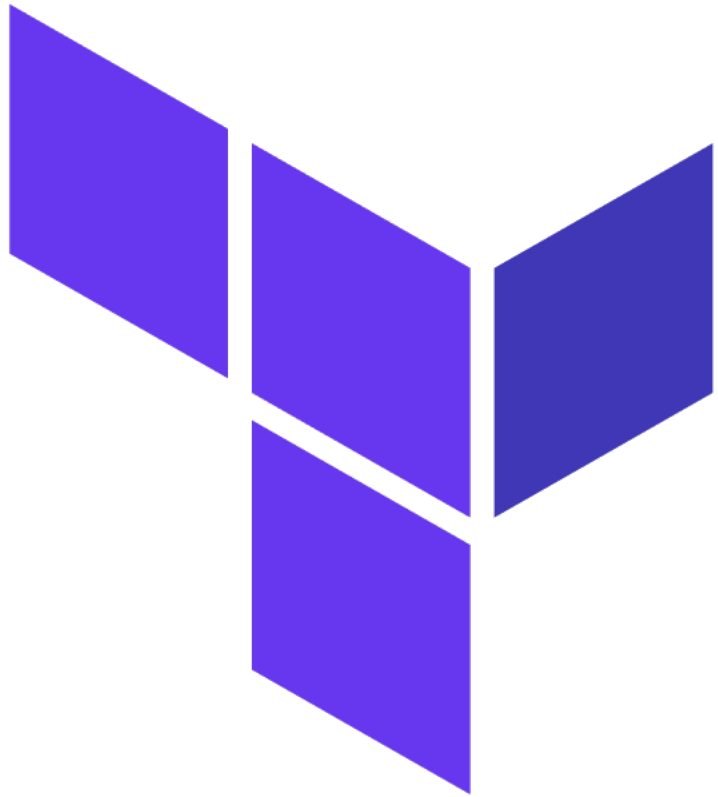
Introduction to Terraform



wifi: ON2IT-guest
password: sinaasappel

ZERO TRUST INNOVATORS

This is a Meetup

(not a one-way presentation)

- We most likely won't be done on time
- Questions are welcome and encouraged
- There are no bad questions
- Interact with each-other
- Get dirty, fix and break things

No marketers
No sales people

Thank you

ZERO TRUST INNOVATORS

Agenda

- What is Terraform (and what it is not)
- Terraform ’basics’
  - Lab 1 – Get your system online
- Terraform ’repeat’
  - Lab 2 – Deploy multiple instances
- Terraform ‘advanced’
  - Lab 3 – Working with ‘dynamic’ parameters
- Terraform clean-up
  - Lab 4 – Destroy Everything
- Conclusion

‘Infrastructure-as-Code’

Infrastructure is anything that supports an application ecosystem

- Rosemary Wang

What it is NOT

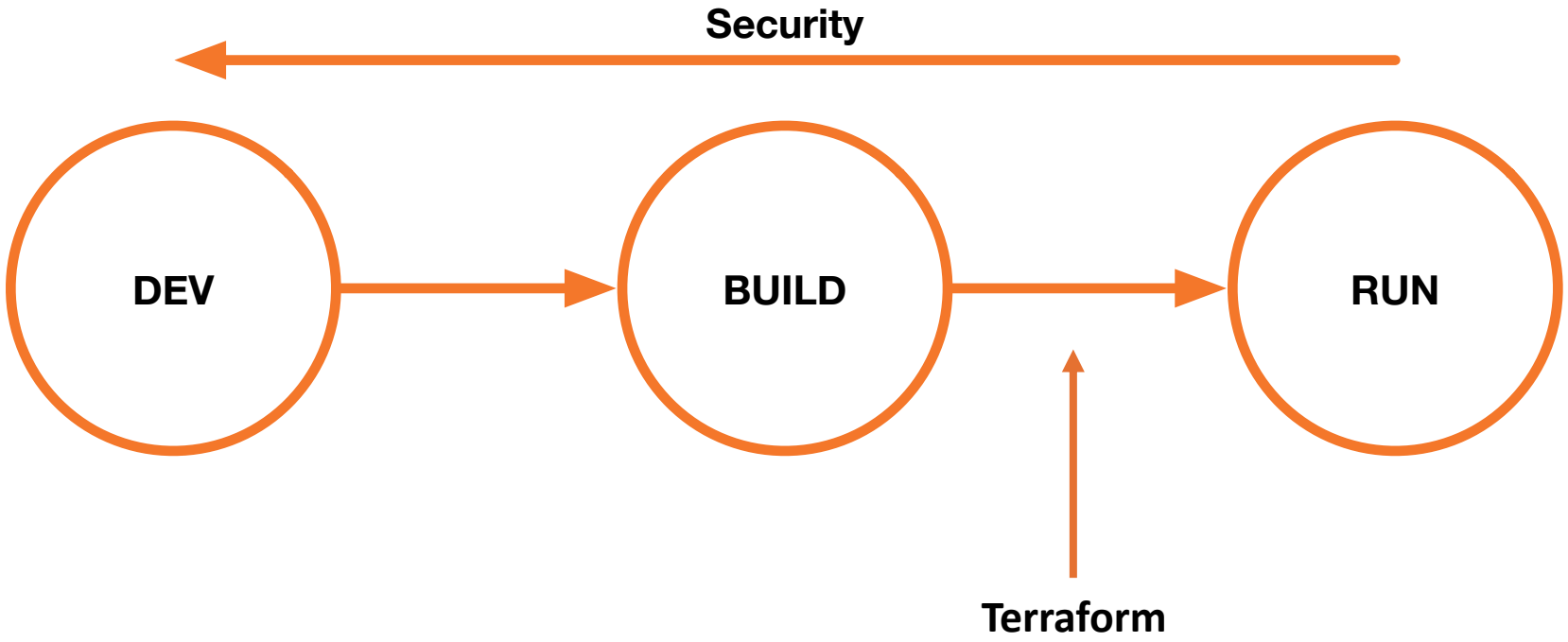
- Abstraction language for deploying multi-cloud.
  - Specific template (provider) per (cloud)-environment

- Application/Server configuration tool
  - There are others that are better suited e.g. Chef, Puppet, Ansible.

What is Terraform

- ‘Infrastructure-as-Code’
- Create once, deploy *X* times (i.e. Dev, Test, Prod)
- Keep control of your assets/resources
  - Will also help you clean up (=reduce costs!)
- Auditable infrastructure !
- TAG everything (i.e. microsegments)

Shift left & Security



Terraform concepts

- HashiCorp Configuration Language (HCL)
  - Declarative, Describing and Goal based
- Provider(s)
  - Interface between Terraform and *any* API
  - Azure, AWS, vSphere, etc. (https://www.terraform.io/docs/providers/index.html)
- Data-sources
  - Can fetch data to use elsewhere in your configuration
- Resources
  - 'resources' you want to deploy, configure, etc.

Terraform concepts

- Variables
  - Can have defaults
  - Can be overridden by CLI or variables file
- Output
  - ‘return values’ of a Terraform template (or Module)
- Module
  - Container for multiple resources that are used together

That’s enough concepts,
How does Terraform work ?

Terraform init

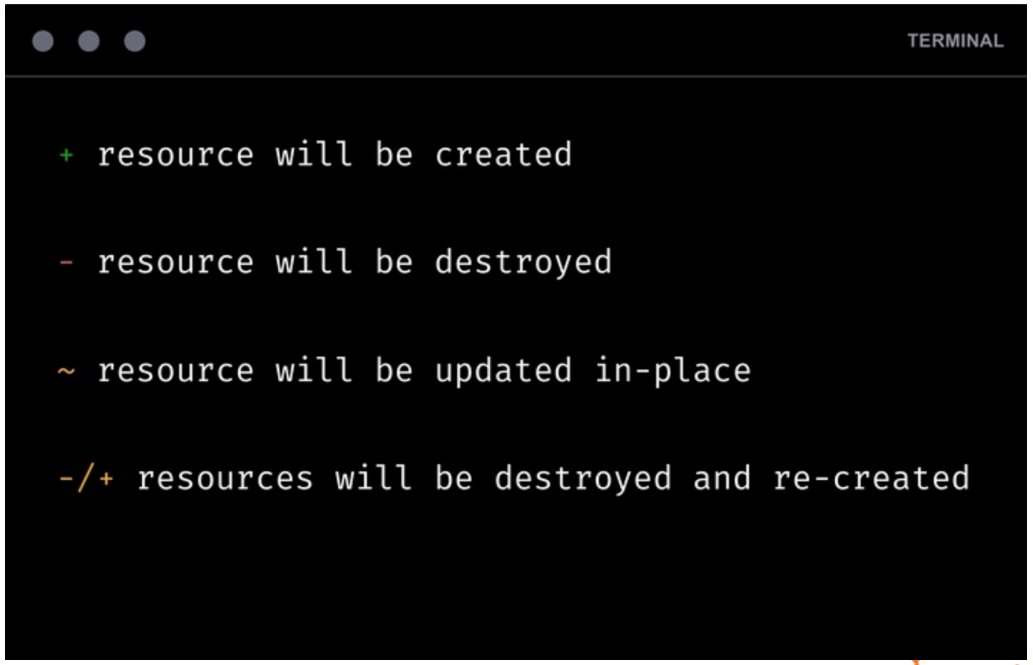
- Initialize terraform directory *(.terraform)*
- Only needs to run after changing the provider configuration

Terraform plan

- Shows what will happen
- Shows why it will happen
- Plans can be saved or can be used as input for a RFC !

- It will not make changes !

- Use ‘-out’ for input during apply.



ZERO TRUST INNOVATORS

Terraform apply

- Execution !
- Parallel execution where possible
- Error handling and recovery
- Produces a **state file**

What is STATE ?

- Before ‘Infrastructure as Code’
  - What is the ‘current’ configuration ?
  - Manual changes

- State must be maintained appropriately
  - Like configuration backup (which it essentially is)
  - Separate file per “environment”
  - Backends: Can be helpful to share the ‘state-file’ *(don’t edit simultaneously)*
    - *i.e. Terraform Cloud (free up to 5 users)*

Tips for the labs

- Visual Code with extension Terraform
- Make sure you have version 0.12 of Terraform (terraform version)
- Don’t specify passwords or secrets in any files

- Be aware a lot of sample code form Terraform has still the 0.11 syntax, which will show a lot of ‘warnings’.
  i.e. “${var.variable1}” -> var.variable1

Tips for the labs

- The labs are not fully written out.
  - Help each-other
  - Visit https://www.terraform.io/docs/
  - Peek on Github for examples of the labs (after 15 minutes of the start of each lab) https://github.com/on2itsecurity/meetup-terraform-introduction

- https://portal.azure.com can help you to give an idea what happened

- Use “Terraform plan” often, to check your configuration and proposed changes !

Practice:
Lab 0 – Preparation

10 min

Lab 0 – Connect to Azure (with Azure CLI)

- Lab guide: https://github.com/on2itsecurity/meetup-terraform-introduction

- Username: lab*XX*@labs.on2it.net
  Password: P@ssw0rd!123

- Login with ’az login’

Food & Drinks

30 min

Theory:
Lab 1 – Get your VM up & Running

Define your provider

- Version is optional, but highly recommended
- Credentials can be provided

  (By default will use the .Azure/.AWS in your home-dir, created by AZ/AWS CLI)
- Additional options per provider

```
provider "aws"{
        version = "~> 2.0”
        region = "eu-central-1"
}
```

```
provider ”azurerm"{
        version = "=1.36.0"
}
```

ON2IT ZERO TRUST INNOVATORS

Define resources

- Every provider has its own resources (with arguments)

```
//Resourcegroup in Azure
resource "azurerm_resource_group" "rg" {
    name     = "The Best Meetup"
    location = "West Europe"
}
```

```
resource "aws_vpc" "vpc" {
        cidr_block = "10.10.0.0/16"
}
```

Define variables

- Create variables where possible/wanted
- Can be in a separate ‘.tf’ file in the same directory as your main.*tf*
- Environment deviations can be stored in ’.tfvars’ file.

```
//Variable declaration with default setting
variable "region" {
       default="eu-central-1"
}
```

```
//Set variable (separate file, *.tfvars)
region = ”eu-west-1”
```

```
//Usage of a variable
provider "aws"{
       version = "~> 2.0”
       region = var.region
}
```

ZERO TRUST INNOVATORS

Practice:
Lab 1 – Get your VM up & Running

30 min

Theory:
Lab 2 – Deploy multiple VMs

Count on it

- count – will deploy a resource a number of times
- count.index – will keep track of the current index (starts with 0)

  (comes in handy for naming)

```
//Create 3 resource groups (rg1, rg2, rg3)
resource "azurerm_resource_group" "rg" {
    count    = 3
    //name   = "rg${count.index}"
    name     = format("%s%s", "rg", count.index)
    location = "West Europe"
}
```

Count on it

- Resources, will be in a list and can be addressed by the index

```
//Create 3 network interfaces
resource "azurerm_network_interface" "vm-nic" {
    count = 3
    …
}
```

```
//Add each nic to each VM
resource "azurerm_virtual_machine" "vm" {
    count = 3
    …
    network_interface_ids = azurerm_network_interface.vm-nic[count.index]
}
```

Creating an if statement with count

- count can be used to create an if-statement whether a resource should be created or not.

```
//Create resource-group if var is set to true
resource "azurerm_resource_group" "rg" {
    count    = create_resourcegroup ? 1 : 0
    name     = "42"
    location = "West Europe"
}
```

- The if statement is a ternary operator

```
count = condition ? true : false
```

Practice:
Lab 2 – Deploy multiple VMs

30 min

Theory:
Lab 3 – Add multiple NICs to a VM

Dynamic blocks

- Dynamic blocks ≈ count for arguments
- Used for ‘repeatable nested blocks’
- i.e. NSG rules, LB rules, IP-Configurations on NIC (Azure)

```
//What if we want multiple ip_configurations on a nic ?
resource "azurerm_network_interface" "vm-nic" {
  …

  ip_configuration {
    name = "ipconfig1"
    subnet_id = azurerm_subnet.subnet-servers.id
    private_ip_address_allocation = "Dynamic”
  }
}
```

Dynamic blocks

- Keywords: dynamic, for_each, content
- Key, Value

```
resource "azurerm_network_interface" "vm-nic" {
  …

  dynamic ip_configuration {
    for_each = [1,2,3]
    content {
      name = format("%s%s","ipconfig", ip_configuration.value)
      subnet_id = azurerm_subnet.subnet-servers.id
      private_ip_address_allocation = "Dynamic"
    }
}
```

Practice:
Lab 3 – Add multiple NICs to a VM

30 min

Final remarks

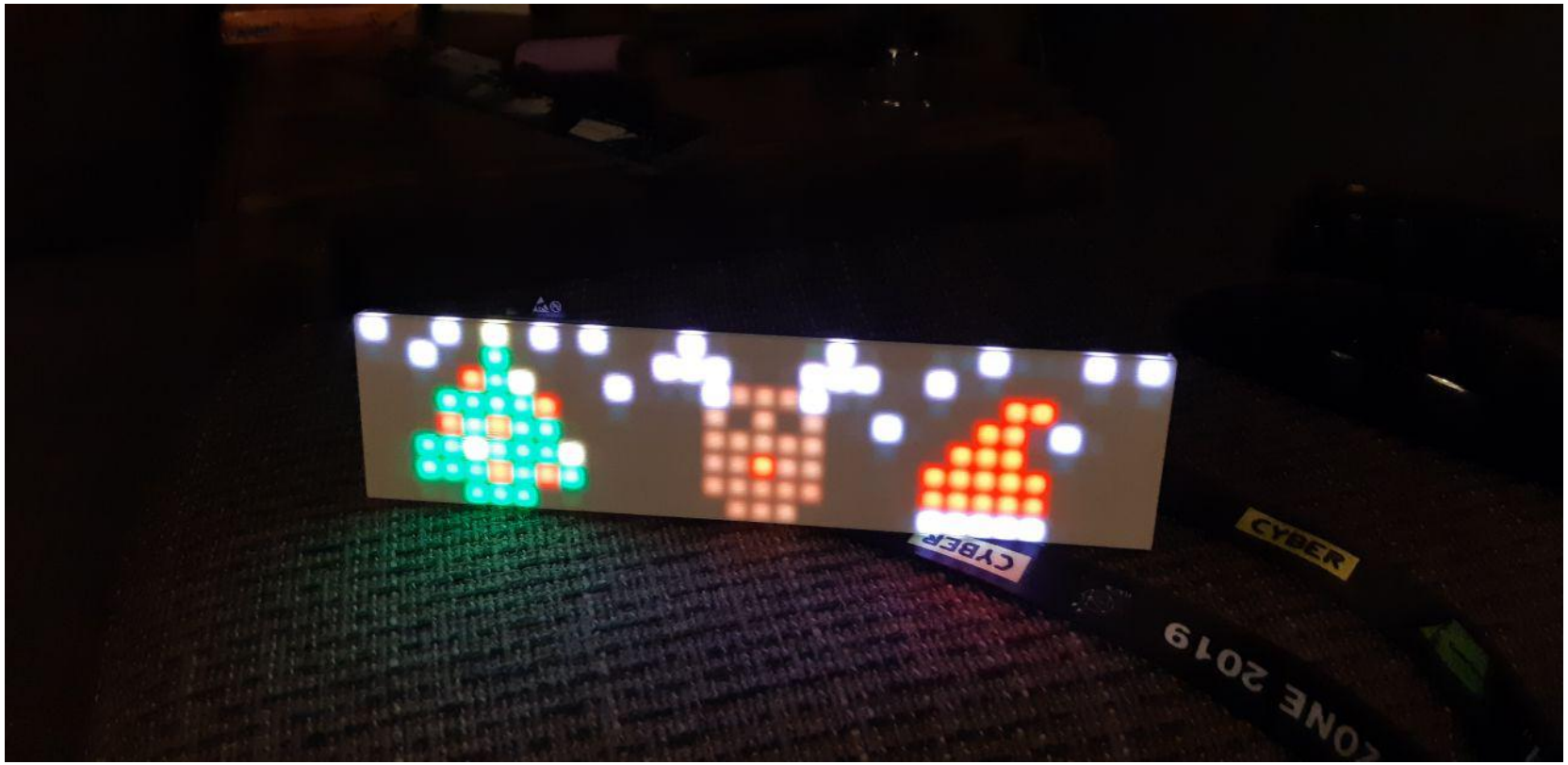
- Create templates with variables
  - i.e. use the same template for Dev, Test, Acceptance and Production
- Version control on templates, helps auditability
- Protect your “STATE” files
- HCL one language/standard across providers

Practice:
Lab 4 – Destroy your workloads
(terraform destroy)

10 min

Questions and next meetup

- Questions ?
- Next Meetup



ZERO TRUST INNOVATORS

ZERO TRUST INNOVATORS

Committed to innovation that delivers leading edge Zero Trust cyber security solutions, managed security services and support to our clients.