

“GUI is dead, long live IaC”

IaC in the aspect of cloud automation
using a standardized module library



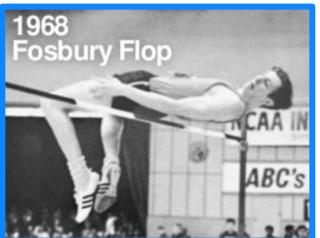
Olympic High Jump

Motto: “Faster, Higher & Stronger”



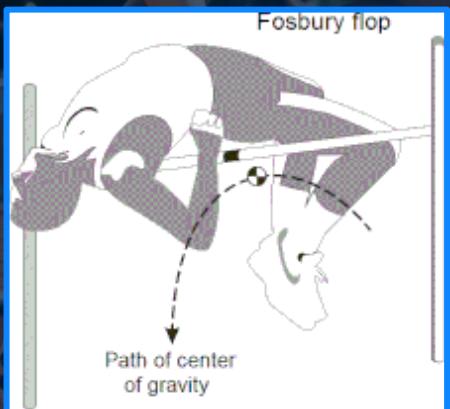
- › Standing Jump
- › Scissors
- › Straddle
- › Western Roll

? Techniques



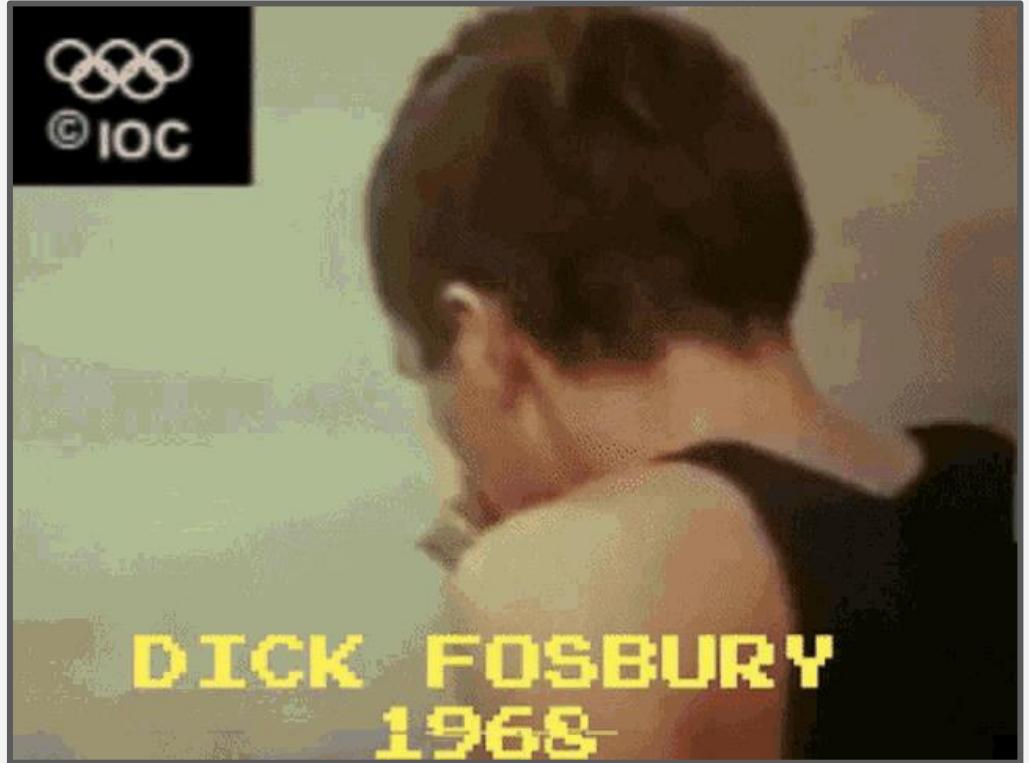
- › In 1968, Mexico City at the Olympic Games
- › High Jump + Engineering Know-How = Center of gravity
- › Newspapers wrote, ...like a “two-legged-camel”... 🐾
- › Result: 2,24 m – Gold Medal, Olympic Record ⚡
- › “I think quite a few kids will begin trying it my way now.”

Fosbury Flop



Revolutionary

“Fosbury Flop”



It is now the only way to do high jump! – Revolutionary!



Technology Shift/Change

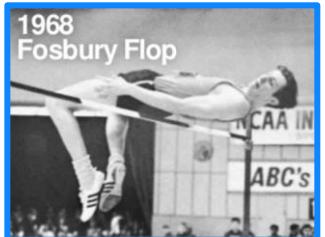
Why: "*Infrastructure as Code*" (IaC)



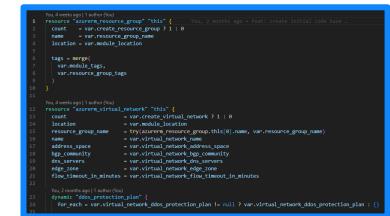
Techniques



GUI Deployment



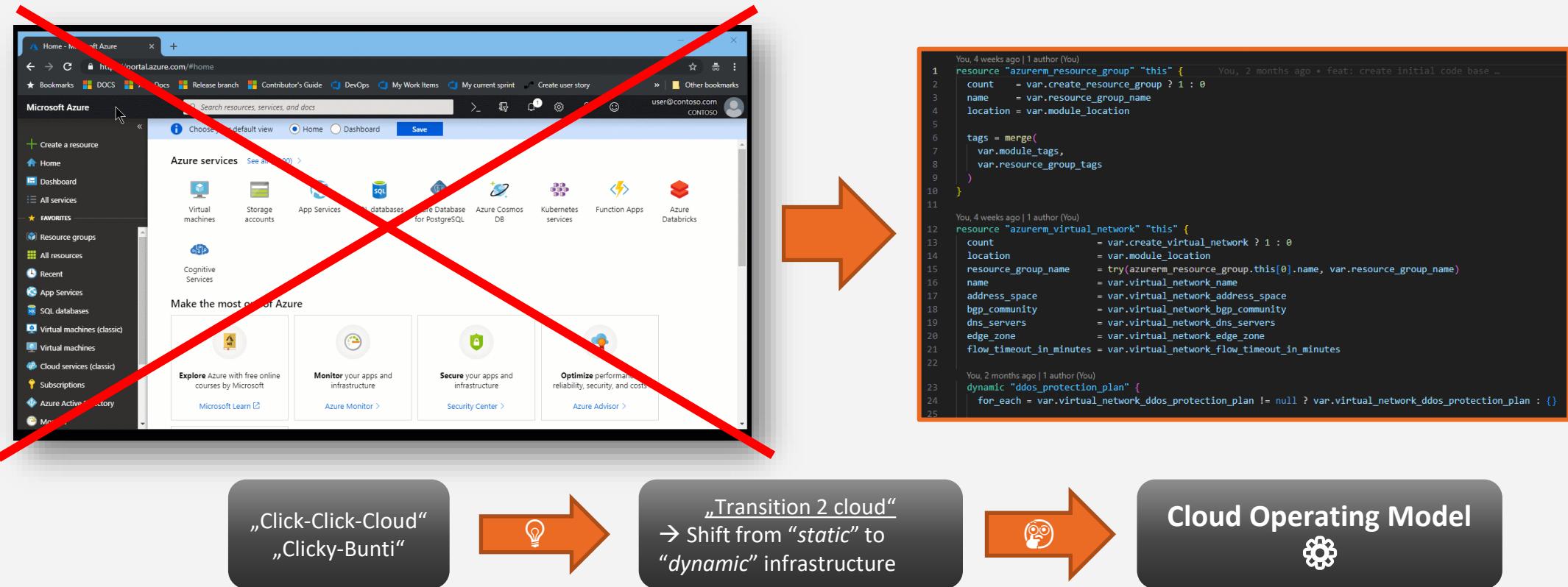
Fosbury Flop



Cloud Automation

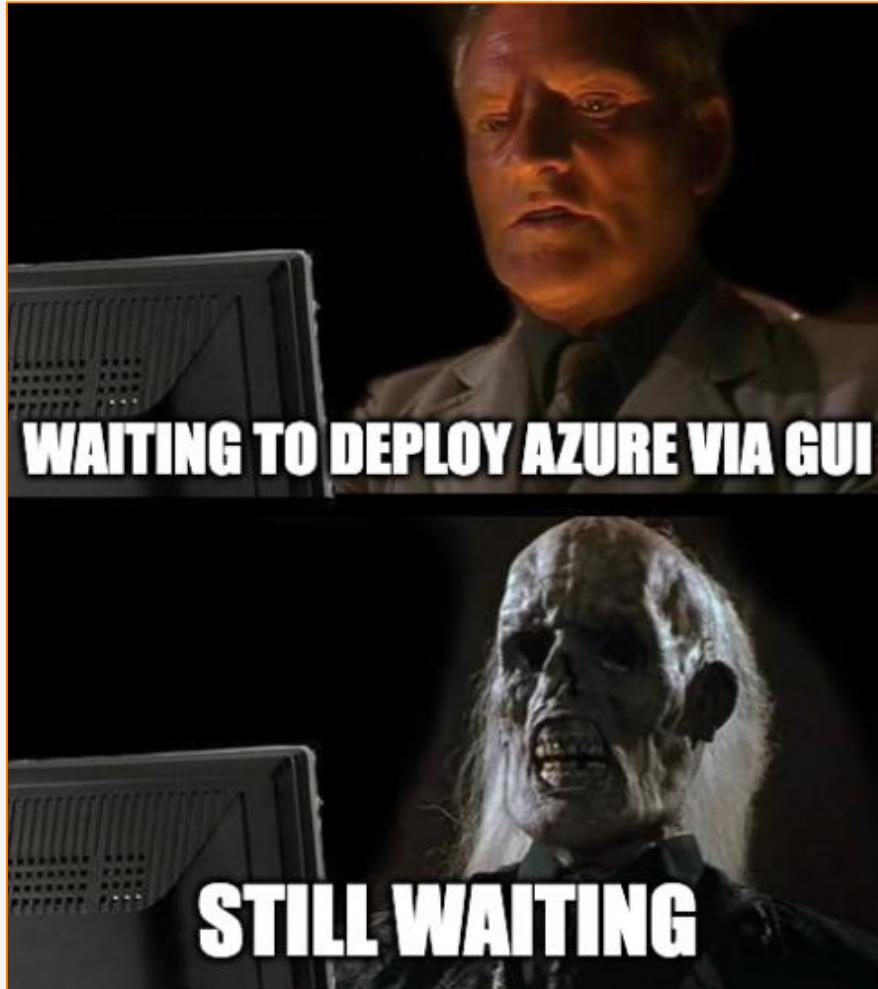
Mindset of IaC?

Mind change administration interface

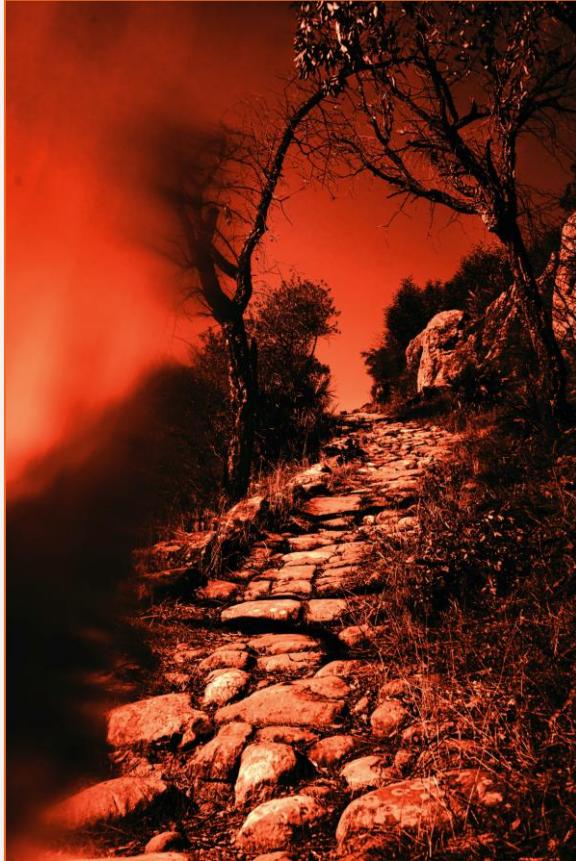


Using **IaC** deployment **templates** & **modules** in building out **your** individual Cloud Infrastructure!

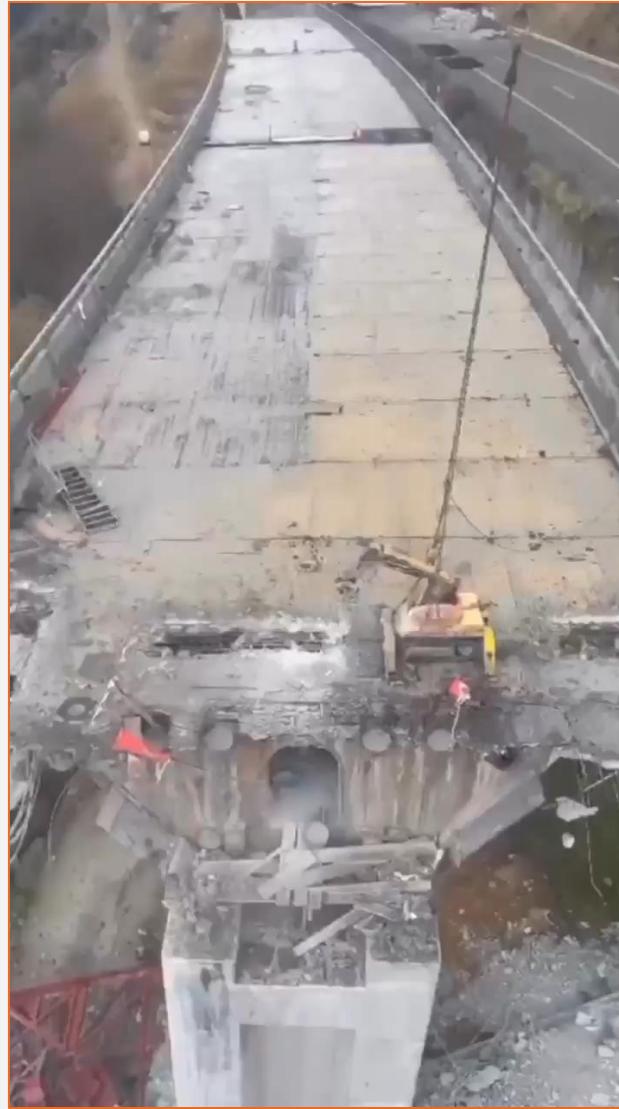
Why using IaC instead of a GUI?



Avoid: Road to “Modularization Hell”



Hell – How bad Cloud Automation (IaC) works?



Focus on laC

What is IaC?

Managing/Provisioning of **infrastructures** using **code**.

How resources, applications and environments are **configured**.

Code describes the **resources** and the **whole architecture landscapes**.

Declarative syntax can be used to specify the resources and detailed settings.

Human readable code files (`.tfvars`) describe how the infrastructure looks like.

Definition of **variables** and their **values** of the environment.



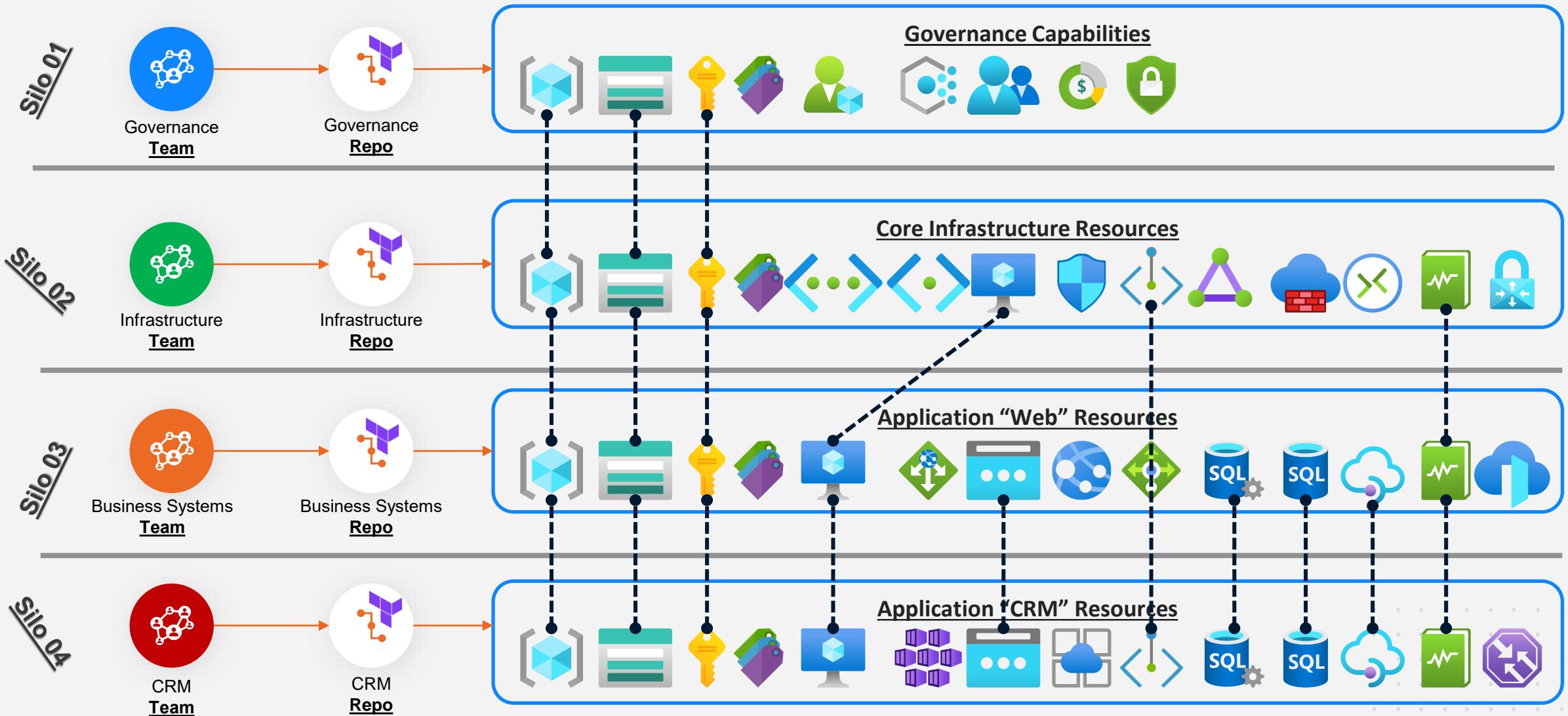
Demo: Terraform Code Example

Advantages of using an IaC approach

- › Reusability
 - Creation of "repeating" infrastructures (**standardization/template**).
- › Speed
 - **Faster** creation of resources and environments.
- › Parallelization
 - Several developers can work on the same infrastructure **simultaneously**.
- › Documentation
 - The **code definition file** is also the documentation of the environment.
- › Consistency
 - The code reflects the **overall state** of the environment.
- › Tracking
 - Use of source code management (**version control**).
- › Order
 - Order of resource creation is managed **automatically**. Automate changes.
- › Error/state handling
 - **No check** necessary if resources already exist ("Desired State").
- › Quality
 - Less **human error** by using the same code ("no-brainer").
- › Reproducibility
 - The **same code** always leads to the **same result**.
- › Cost Reduction
 - Remove **manual** tasks. People refocus their efforts on enterprise tasks.

How IaC? – Modularization

IaC – Status quo Situation



What is a **module**?

Create lightweight **abstractions**

Don't describe **physical** objects

Describe infrastructure in terms of its **architecture**

Package and **reuse** resource configurations

Separated in code **repositories**

Independently usable & testable

“A **module** is a container for multiple resources that are used together.”





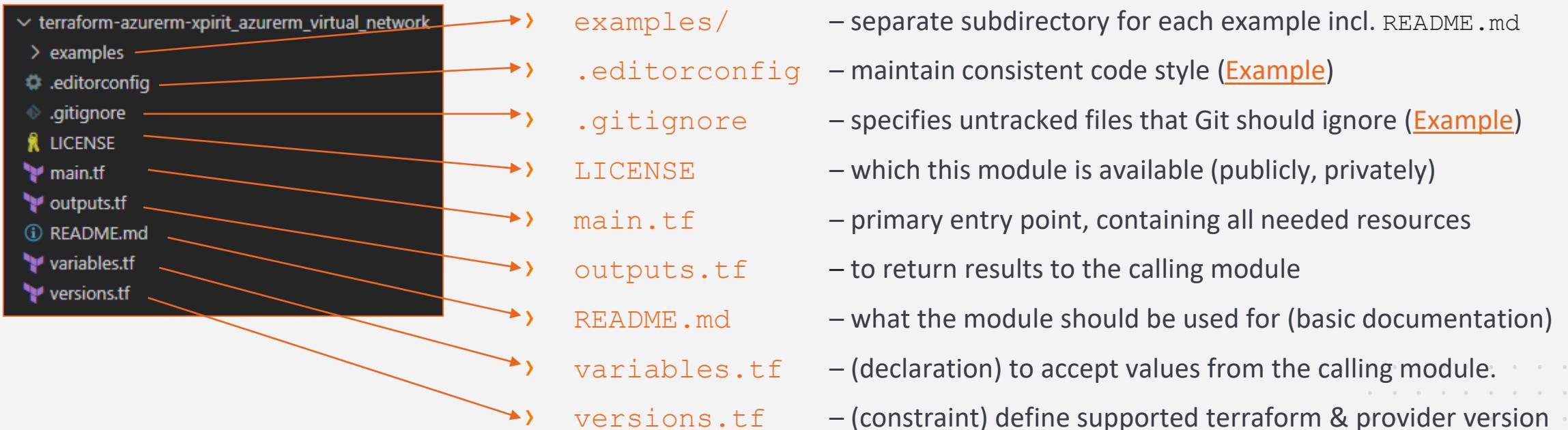
Demo: Terraform Module Example

Standard (Terraform) Module Structure

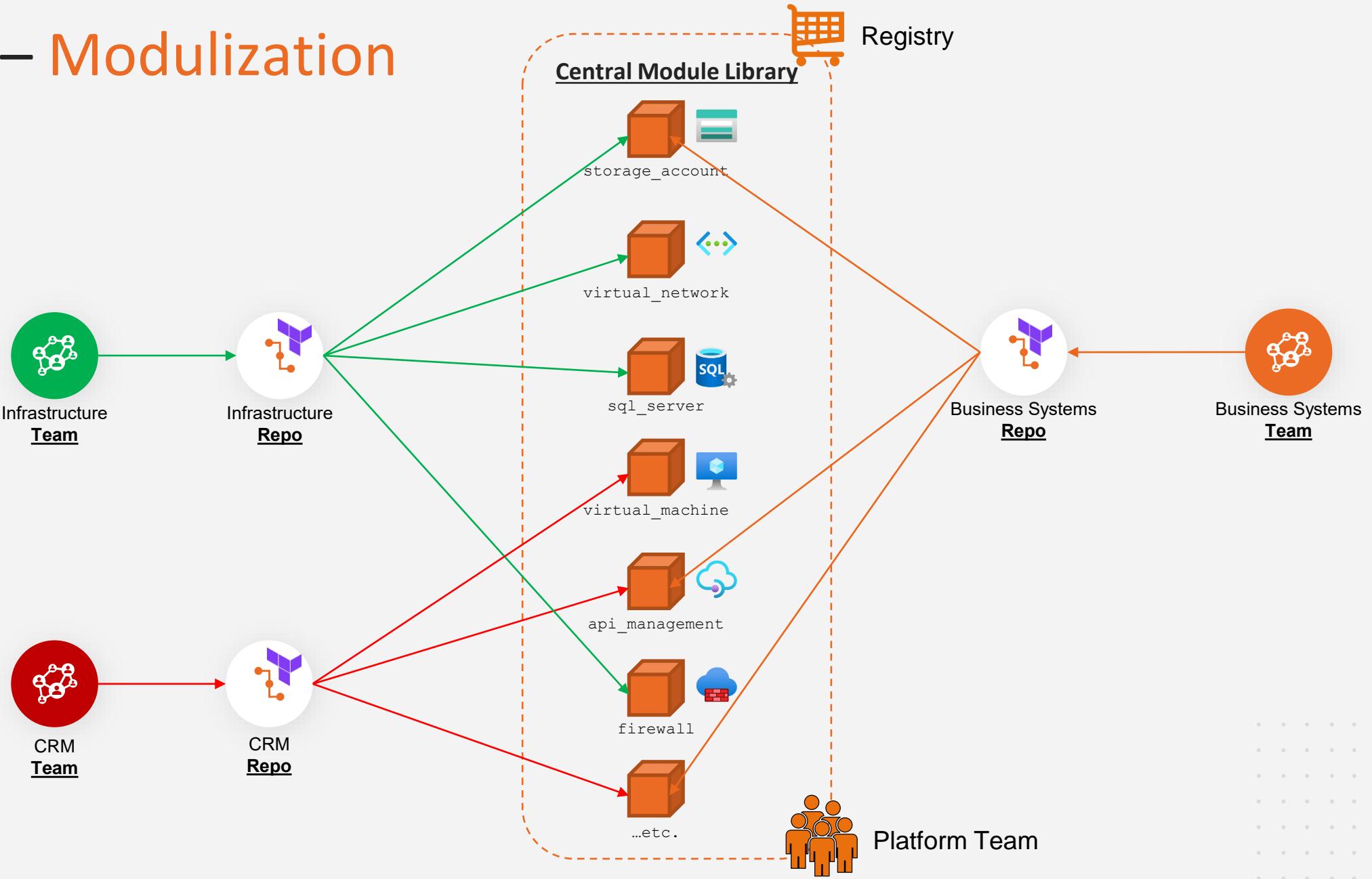
1.
file & directory layout

2.
Distributed in separate
repositories

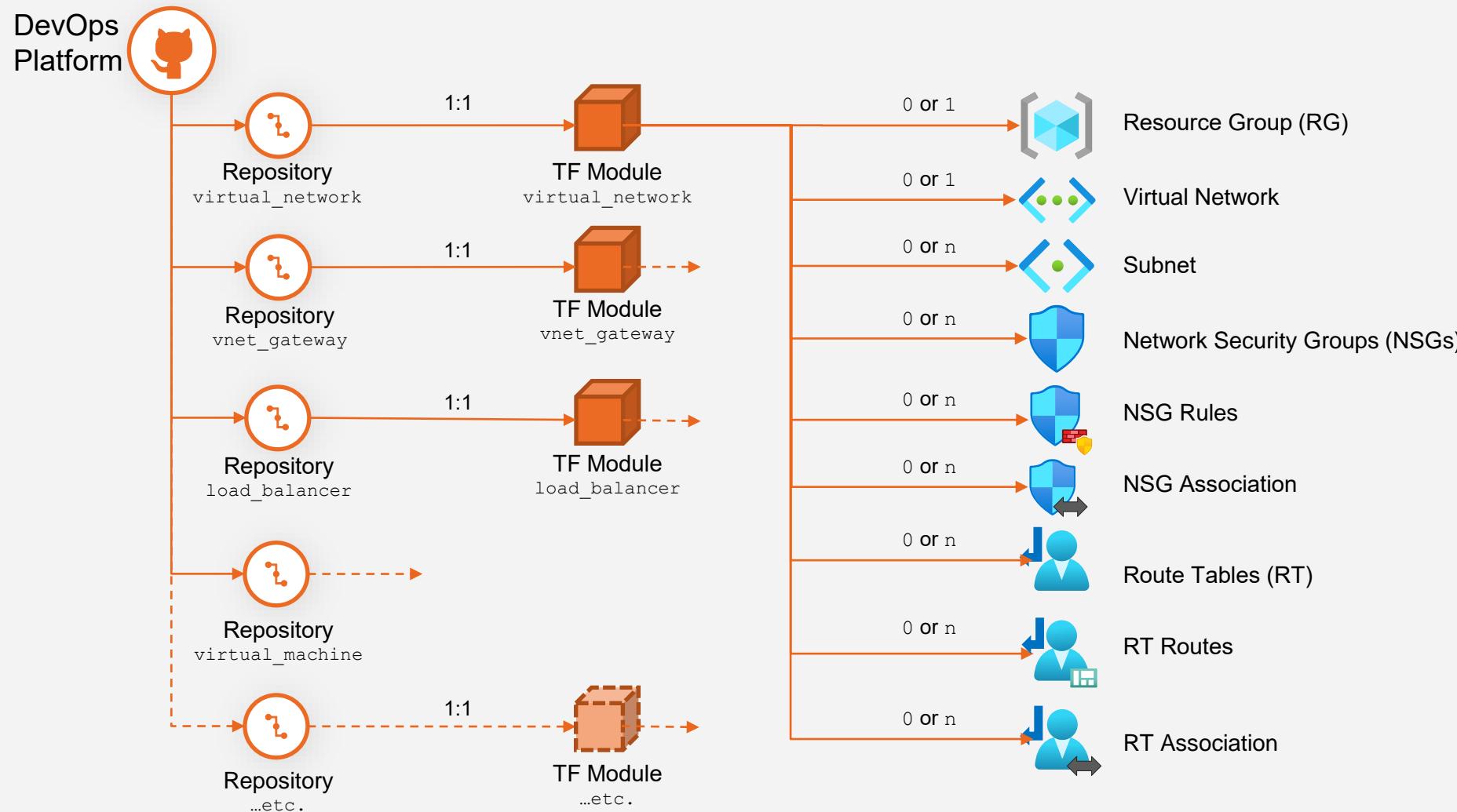
3.
Structure to generate
documentation



IaC – Modulization



Standard Module Architecture



Creating Good Modules

1. Do one thing and do it as **simply** as possible 1
2. Provide **options**, but not too many option ✓
3. Add **value**, do more than wrap a resource +
4. **Version** appropriately vs
5. Make them **easy** to obtain E
6. Keep them **updated** ⚡
7. Start simple and **evolve** ↑_{TOP}
8. Create great **documentation** 📄
9. Evangelize and **engage** 🔊
10. KISS & DRY → But avoid “thin wrappers” ☁

Source: Sam Cogan (Video: [Reuse, Don't Repeat](#)) & Thomas Thornton ([Link](#)) & Stefan Rapp ([Link](#))

How to publish Modules?

Use a Terraform Registry

TF Cloud helps teams to **provision** infrastructure.

Support for **versioning** and list of **providers** and **modules**.

Handles **downloads** and **access** with TF Cloud API tokens.

Consumers do not need **direct** access to the module's repository.



```
module_location = var.bastion_host_module_location
module_tags     = var.bastion_host_module_tags

# azurerm_resource_group
create_resource_group = var.create_bastion_host_resource_group
resource_group_name   = var.bastion_host_resource_group_name

# azurerm_public_ip
create_public_ip      = var.create_bastion_host_public_ip
public_ip_name        = var.bastion_host_public_ip_name
public_ip_domain_name_label = var.bastion_host_public_ip_name
public_ip_zones       = var.bastion_host_public_ip_zones

+ azurerm_bastion_host
create_bastion_host = var.create_bastion_host
bastion_host_name   = var.bastion_host_name
weeks_ago | 1 author (You)
bastion_host_ip_configurations = {
  4 weeks ago | 1 author (You)
  a8947-2efa-4b0e-ac61-c4086259a419 = {
    public_ip_address_id = null
    subnet_id           = module.virtual_network.subnet_ids["AzureBastio
  }

  4 weeks ago | 1 author (You)
  e_dns_resolver" {
    op.terraform.io/Customer-ML-Pritix/xpirit_azurerm_private_dns
    .53.0-0.0.2"
  }

  4 weeks ago | 1 author (You)
  ion = var.private_dns_resolver_module_location
  = var.private_dns_resolver_module_tags

  + resource_group
  _group = var.create_private_dns_resolver_resource_group
  name   = var.private_dns_resolver_resource_group_name

  + azurerm_dns_resolver
  _resolver = var.create_private_dns_resolver
  resolver_name = var.private_dns_resolver_name
  virtual_network_id = module.virtual_network.virtual_
  resolver_inbound_endpoint
  inbound_endpoints      = var.private_dns_resolver_
  inbound_endpoint_subnets = module.virtual_network.su
  resolver_outbound_endpoint
  outbound_endpoints     = var.private_dns_resolver_
  bound endpoint_subnets = module.virtual_network.su
}
```



Demo: Terraform Cloud – Private Registry

How to publish Modules?

› Local file system

```
module "dd34208a_7c40_4dd2_beb5_4a5bf4d21d22" {  
  source = "../../module-library-azurerm/terraform-azurerm-xpirit_azurerm_management_group"  
  
  create_management_group = true  
  management_group_name   = "c06835d9-1135-4b19-b403-749c06443f4c"  
  management_group_display_name = "mg-pritix"  
  management_group_subscription_association_subscription_ids = []  
}
```

› Private registry

```
module "dd34208a_7c40_4dd2_beb5_4a5bf4d21d22" {  
  source = "app.terraform.io/Customer-ML-Pritix/xpirit_azurerm_management_group/azurerm"  
  version = "3.42.0-0.0.2"  
  
  create_management_group = true  
  management_group_name   = "c06835d9-1135-4b19-b403-749c06443f4c"  
  management_group_display_name = "mg-pritix"  
  management_group_subscription_association_subscription_ids = []  
}
```

› Directly from the repository

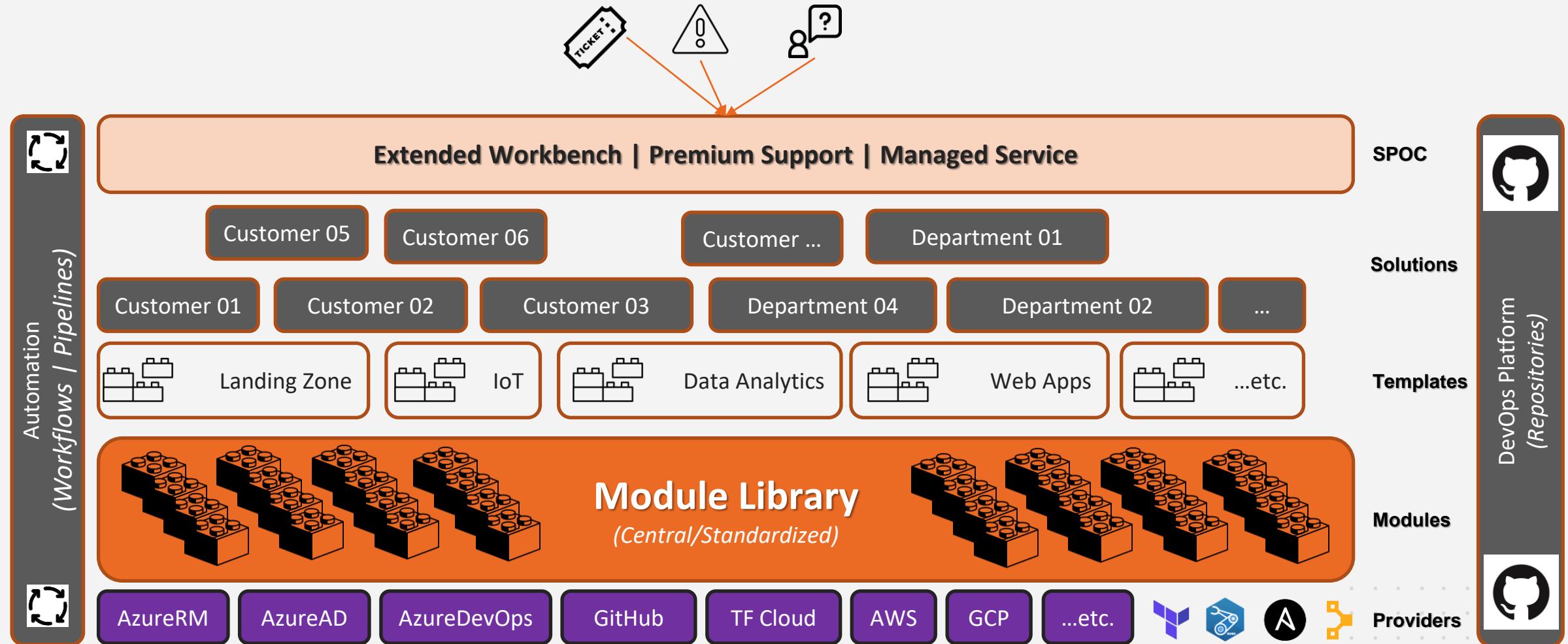
```
module "dd34208a_7c40_4dd2_beb5_4a5bf4d21d22" {  
  source = "git::https://github.com/rapster83/terraform-azurerm-xpirit_azurerm_management_group?ref=v3.42.0-0.0.3"  
  
  create_management_group = true  
  management_group_name   = "c06835d9-1135-4b19-b403-749c06443f4c"  
  management_group_display_name = "mg-pritix"  
  management_group_subscription_association_subscription_ids = []  
}
```

› Public registry (→ TF Registry)

```
module "dd34208a_7c40_4dd2_beb5_4a5bf4d21d22" {  
  source = "azrappster83/management_group/azurerm"  
  version = "4.0.0"  
  
  create_management_group = true  
  management_group_name   = "c06835d9-1135-4b19-b403-749c06443f4c"  
  management_group_display_name = "mg-pritix"  
  management_group_subscription_association_subscription_ids = []  
}
```

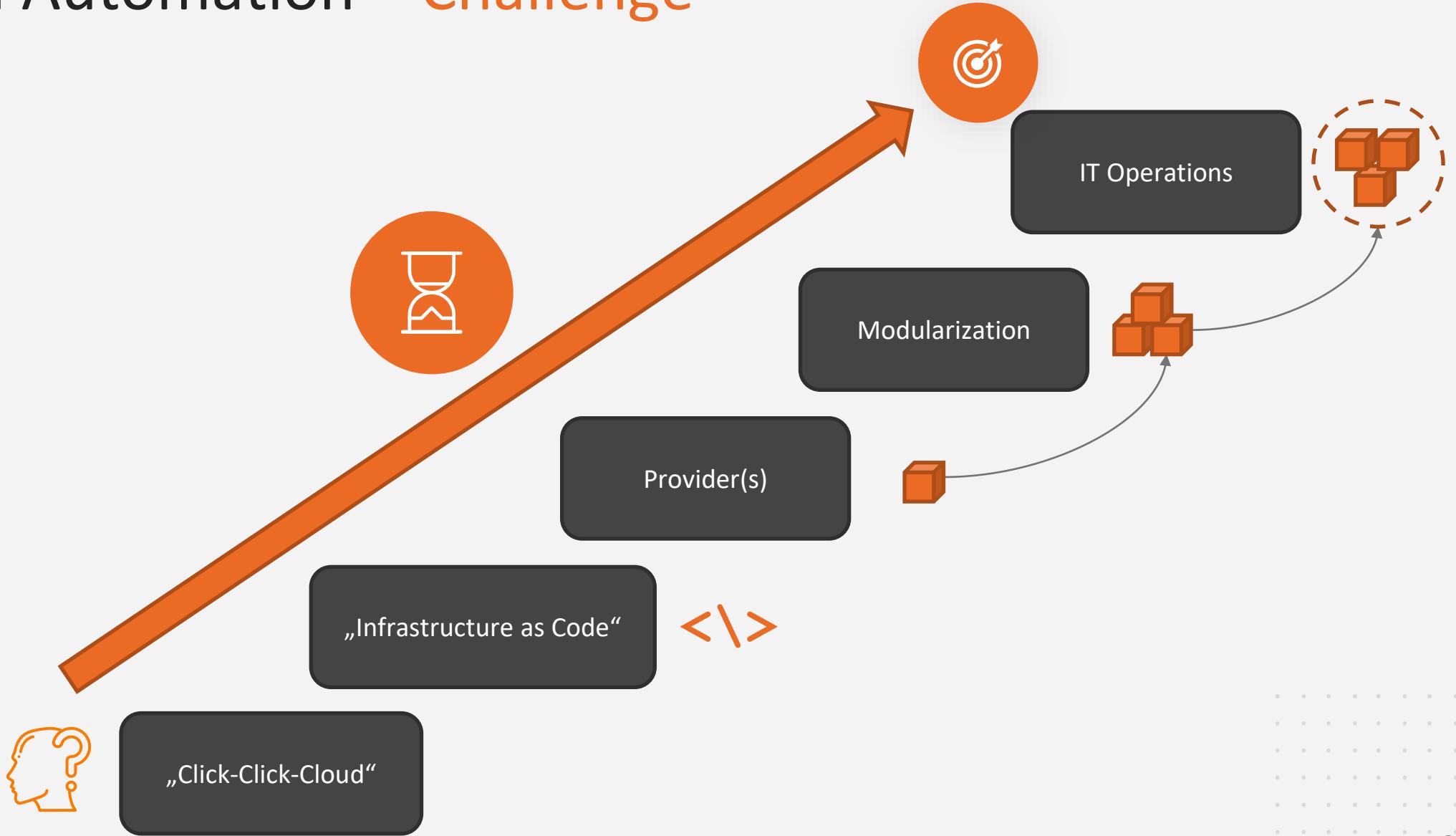
Cloud Operating Model

IaC – Organization (Modules & Templates)



Essentials (Summary)

Cloud Automation – Challenge



Modularization Best-Practices

1. Establish central (standardized) module libraries (**Registry**)
2. Multi-Cloud **Scenarios** → Multi-Provider **Modules**
3. Be familiar provider docs, updates and changelogs
4. Keep pace with the provider **releases** ("Don't get left behind!")
5. Strive for feature completion & automation
6. Provide `default` values and descriptions for variables
7. Put code guidelines in place (naming consistency)
8. Use `versions.tf` in the **module**
9. Follow the "*shit in – shit out*" principal (less validations)
10. Separate the state files into logical pieces → 🧩



Stefan Rapp



Cloud Solution Architect (CSA)
Xpirit Germany GmbH

<https://www.linkedin.com/in/rapster83>



@rapster83



<https://blog.misterazure.com>



- 15 years in IT Consulting
- 6 years MS Development & Infrastructure
- Since 2018 Azure Governance & Infrastructure
- Application Modernization towards Azure
- Pushing IaC (Terraform) at customers

Thank you (Q&A)



Blog : <https://blog.misterazure.com>



GitHub : @rapster83



LinkedIn : <https://www.linkedin.com/in/rapster83/>