Infrastructure as Code & Terraform Basics

Robert Rozas Navarro
Premier Field Engineer
Apps Domain

Agenda

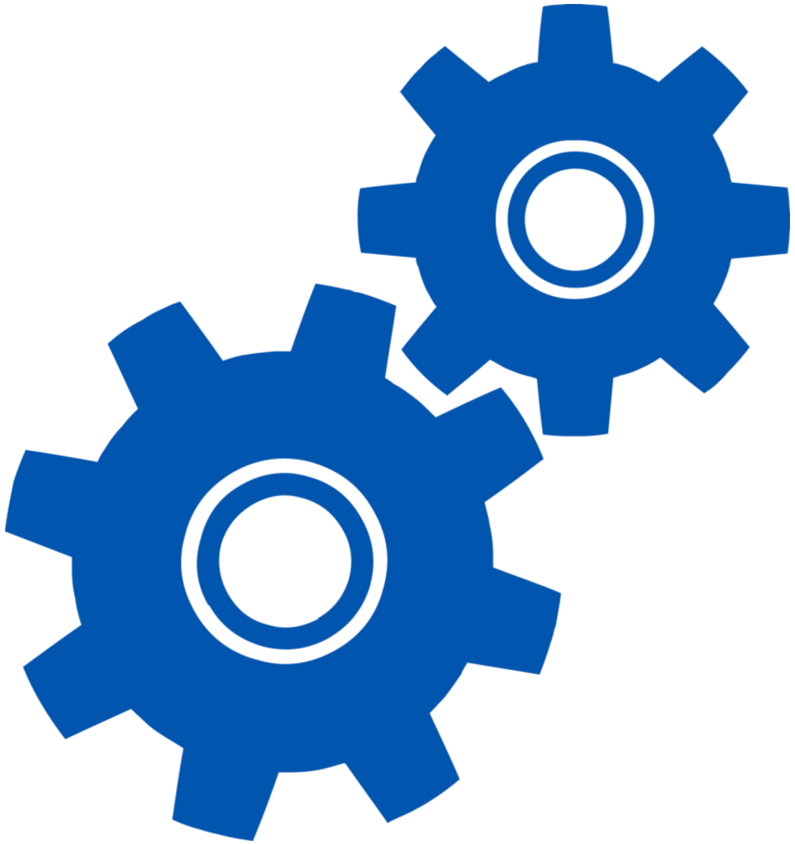
1. Infrastructure as Code (IaC)
2. Terraform
3. Known Providers
4. Q&A

Infrastructure as Code

Microsoft

What is Infrastructure as Code (IaC)

- Build the infrastructure for an App all at once through automation
- Not just for Cloud, Software Defined Data Center
- Embedded Documentation
- Source Control
- Flexible Build Process



Why Infrastructure as Code (IaC)

- Less errors
- Faster to deliver
- Flexibility
- Code is documentation

Provisioning Services is Complicated

No Visibility and Control

Many Departments

Siloed Tools

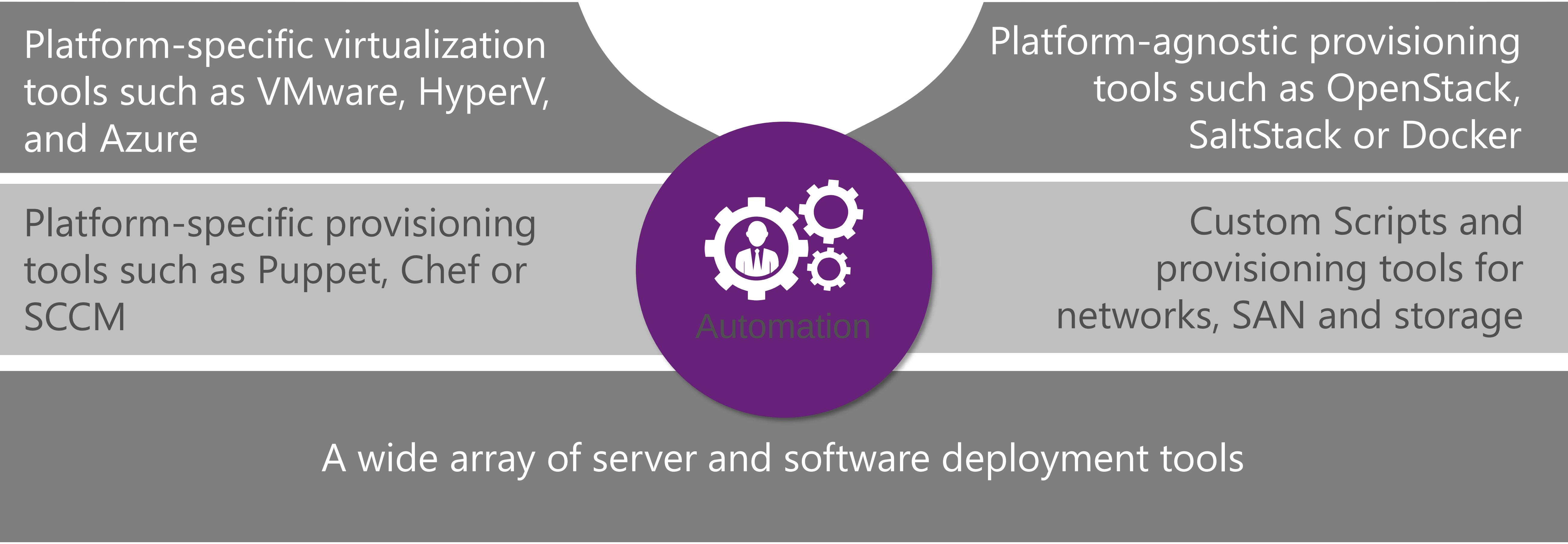
Manual Hand-offs

Lots of Wait Time

Business	Request Service		wait	wait	wait	wait	wait	wait	wait	wait	wait	Use Service
Service Mgmt.		Approve		CAB						CAB		
Server Admins			Provision Servers				Provision Middleware					
Network Engnrg.					Configure Network						Open Access	
Storage Admins						Provision Storage						
Database Admins								Provision Database				
Application Ops									Deploy Apps		Open Access	

10 to 30 Days

Emerging Islands of Automation



Platform-specific virtualization tools such as VMware, HyperV, and Azure

Platform-agnostic provisioning tools such as OpenStack, SaltStack or Docker

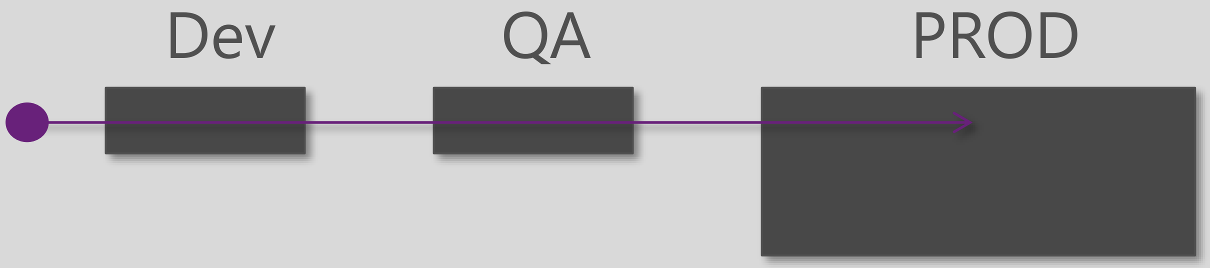
Platform-specific provisioning tools such as Puppet, Chef or SCCM

Custom Scripts and provisioning tools for networks, SAN and storage

A wide array of server and software deployment tools

DevOps Confronts the Agile Challenge

Circa 2010



Development team leads DevOps

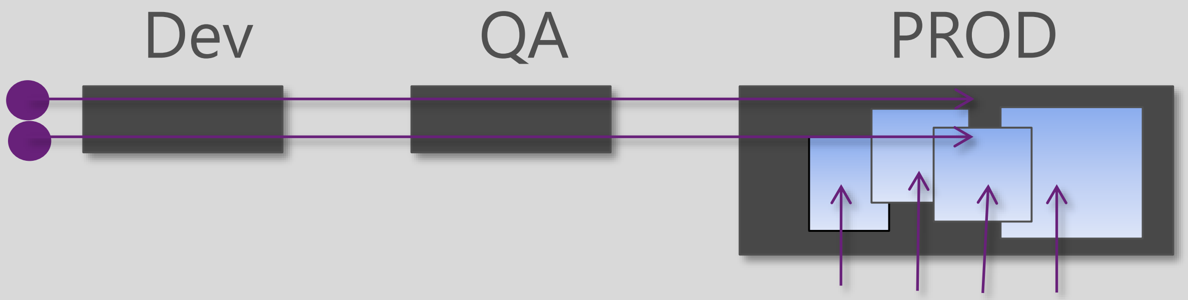
Select Single Applications

Virtual Machines

Infrastructure as Code

Continuous Integration

Circa 2014



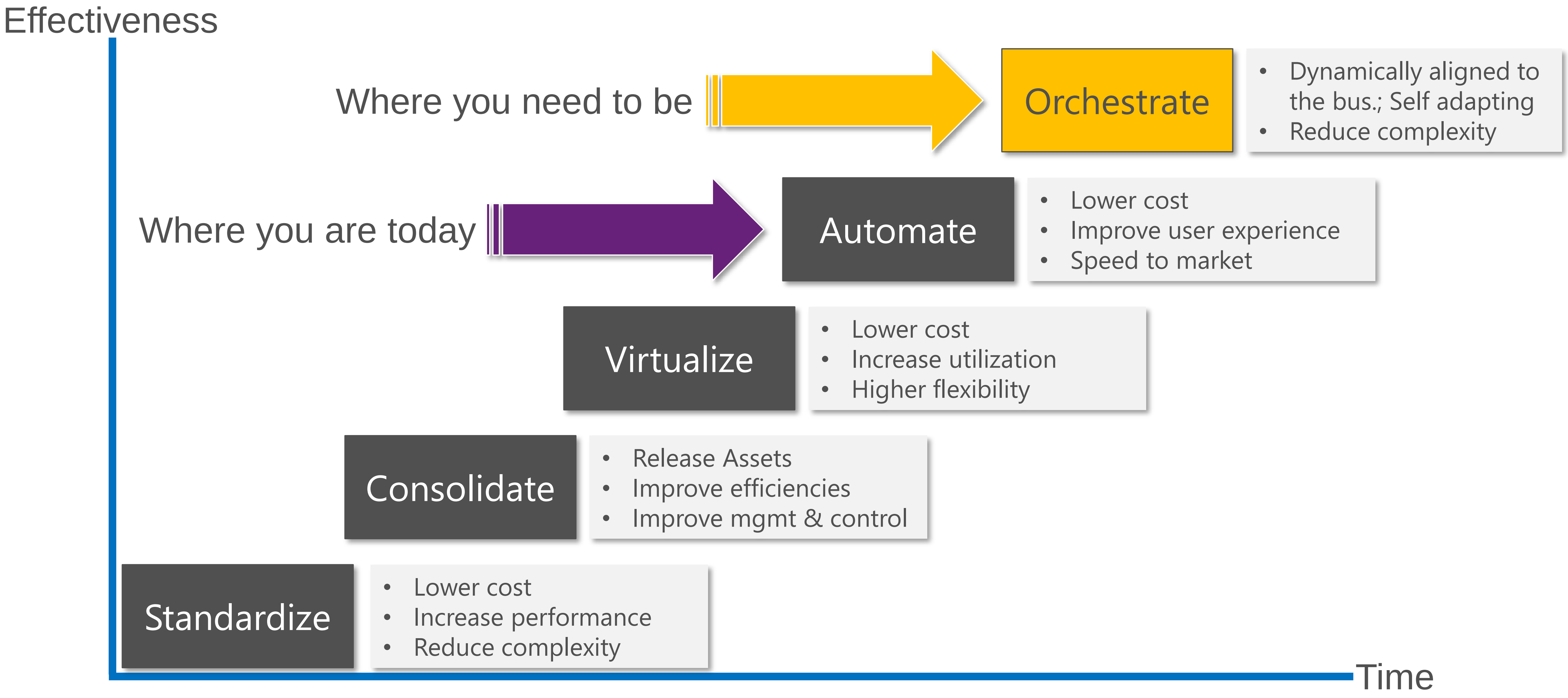
Enterprise-wide DevOps

Shared Private/Hybrid Cloud Environments

Multiple enterprise applications

Orchestration and co-ordination is needed

You Are on the Doorstep of Better Results



Effectiveness

Where you need to be

Orchestrate

- Dynamically aligned to the bus.; Self adapting
- Reduce complexity

Where you are today

Automate

- Lower cost
- Improve user experience
- Speed to market

Virtualize

- Lower cost
- Increase utilization
- Higher flexibility

Consolidate

- Release Assets
- Improve efficiencies
- Improve mgmt & control

Standardize

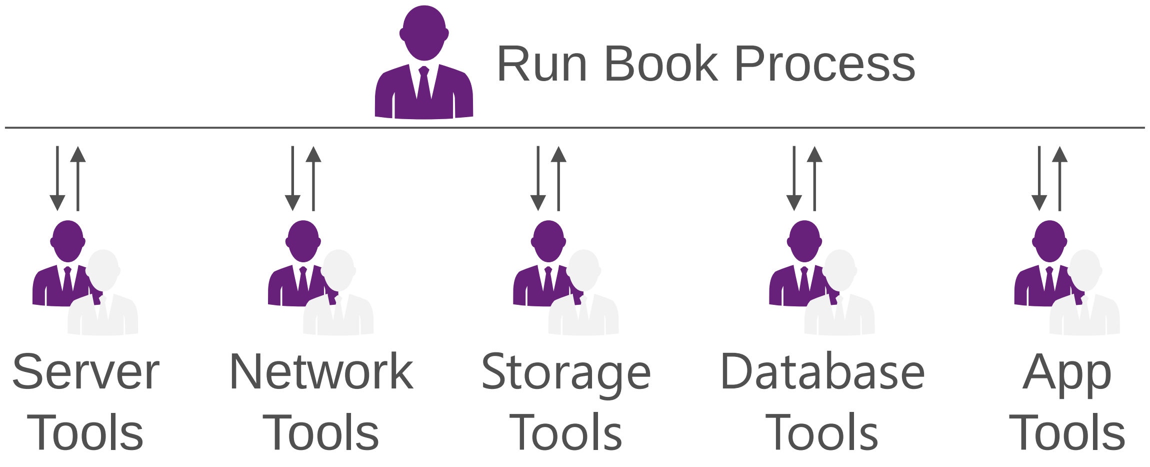
- Lower cost
- Increase performance
- Reduce complexity

Time

Source: Cloud Computing – Wave 51 ©2013 541 Research LLC www.451research.com

Tying Together Islands of Automation

Open Loop Task Coordination



Run Book Process

Server Tools

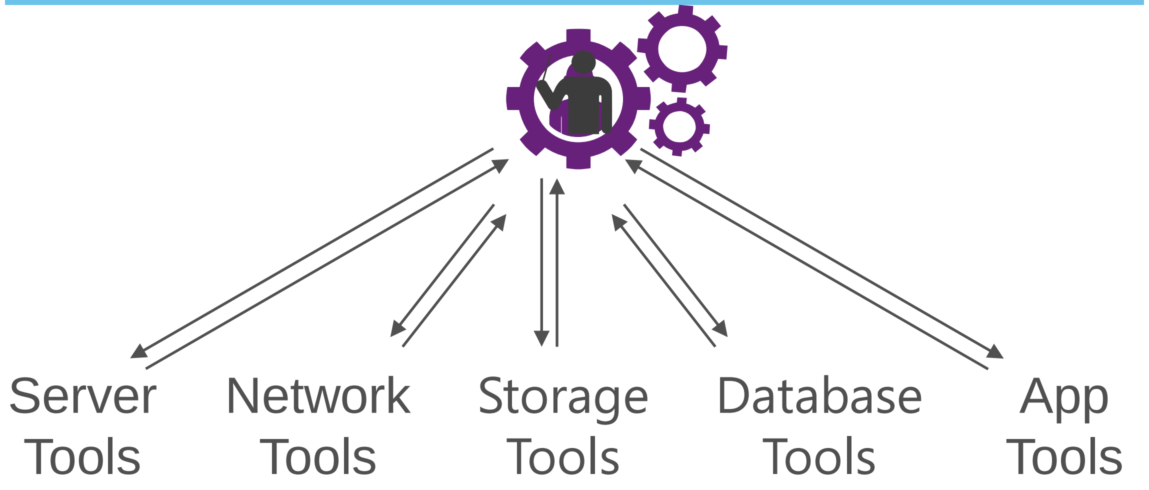
Network Tools

Storage Tools

Database Tools

App Tools

Closed Loop Orchestration



Server Tools

Network Tools

Storage Tools

Database Tools

App Tools

Manual or semi-automated; simple process → Fully automated, simple or complex process

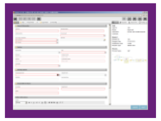
Specialized task execution tools in each group → Existing specialized task execution tools

Serial stepping from group to group → Parallel or serial group operations

Semi-automated data exchange → Automated data exchange

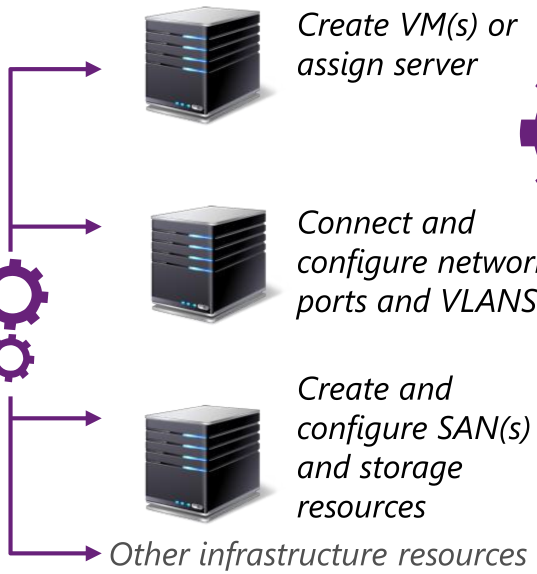
An Orchestrated Example

1 Customer requests application for new branch office



2 Customer information is collected in Web form and workflow initiated by request

3 Orchestration triggers specialized tools to automate provisioning of required infrastructure



Create VM(s) or assign server

Connect and configure network ports and VLANS

Create and configure SAN(s) and storage resources

Other infrastructure resources

4 Orchestration triggers software deployment by specialized tools and loads application data

- Install and configure OS
- Install and configure WebLogic and other middleware
- Install and configure database
- Install application(s)
- Load data
- Test operation

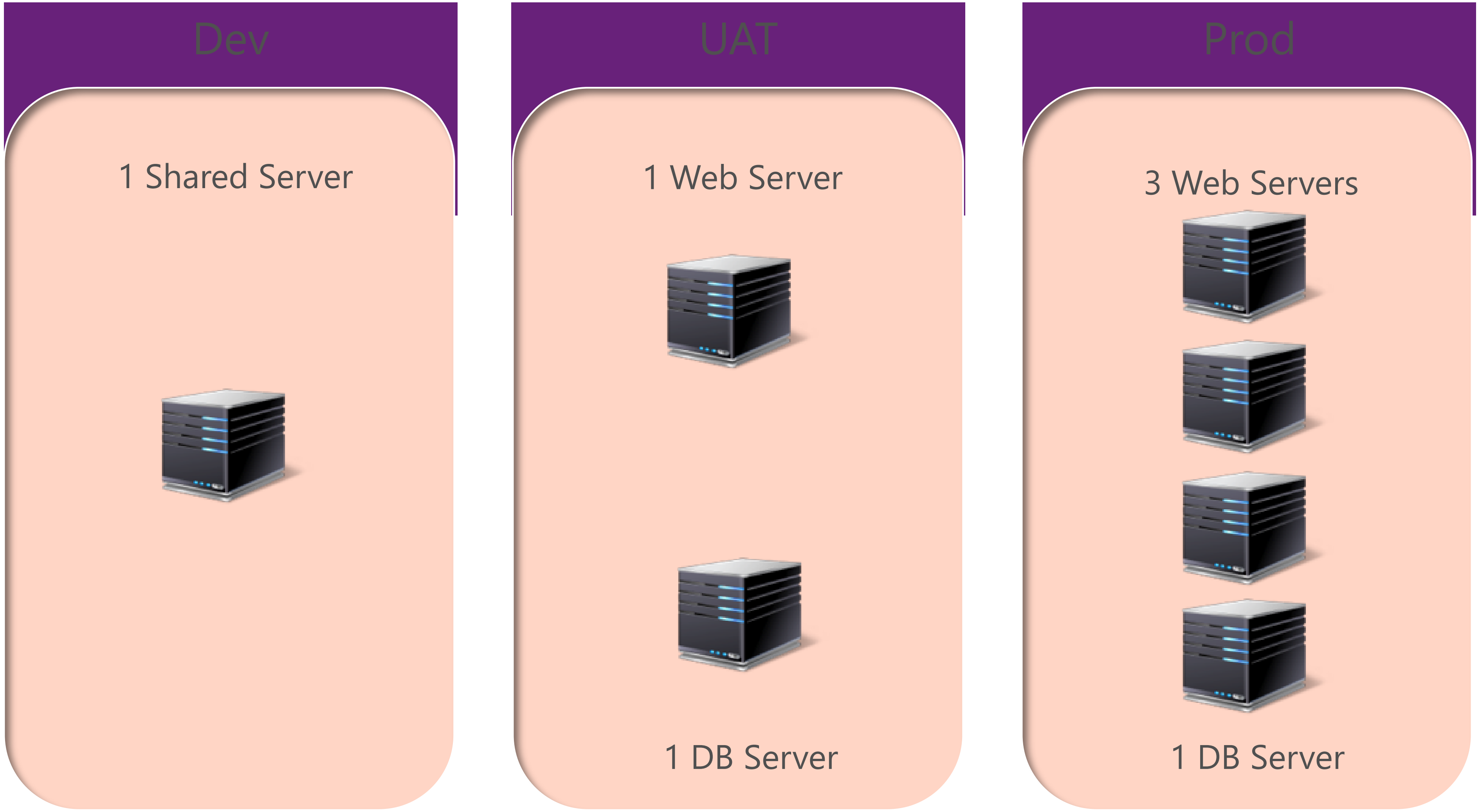
Real-time Progress

5 Network and application access automatically enabled so branch office can use app.

An hour or less later the branch office starts using application

No staff, no hand-off delays, no errors and no compliance problems

Infrastructure Changes over Cycle



Dev

1 Shared Server

UAT

1 Web Server

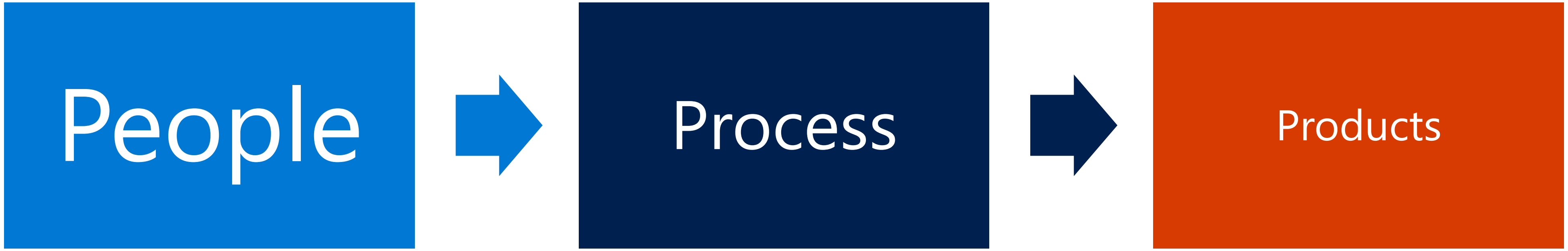
1 DB Server

Prod

3 Web Servers

1 DB Server

How to Get Started



People

Process

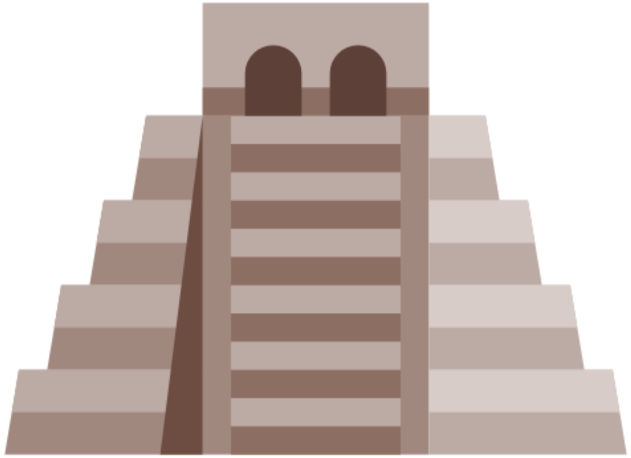
Products

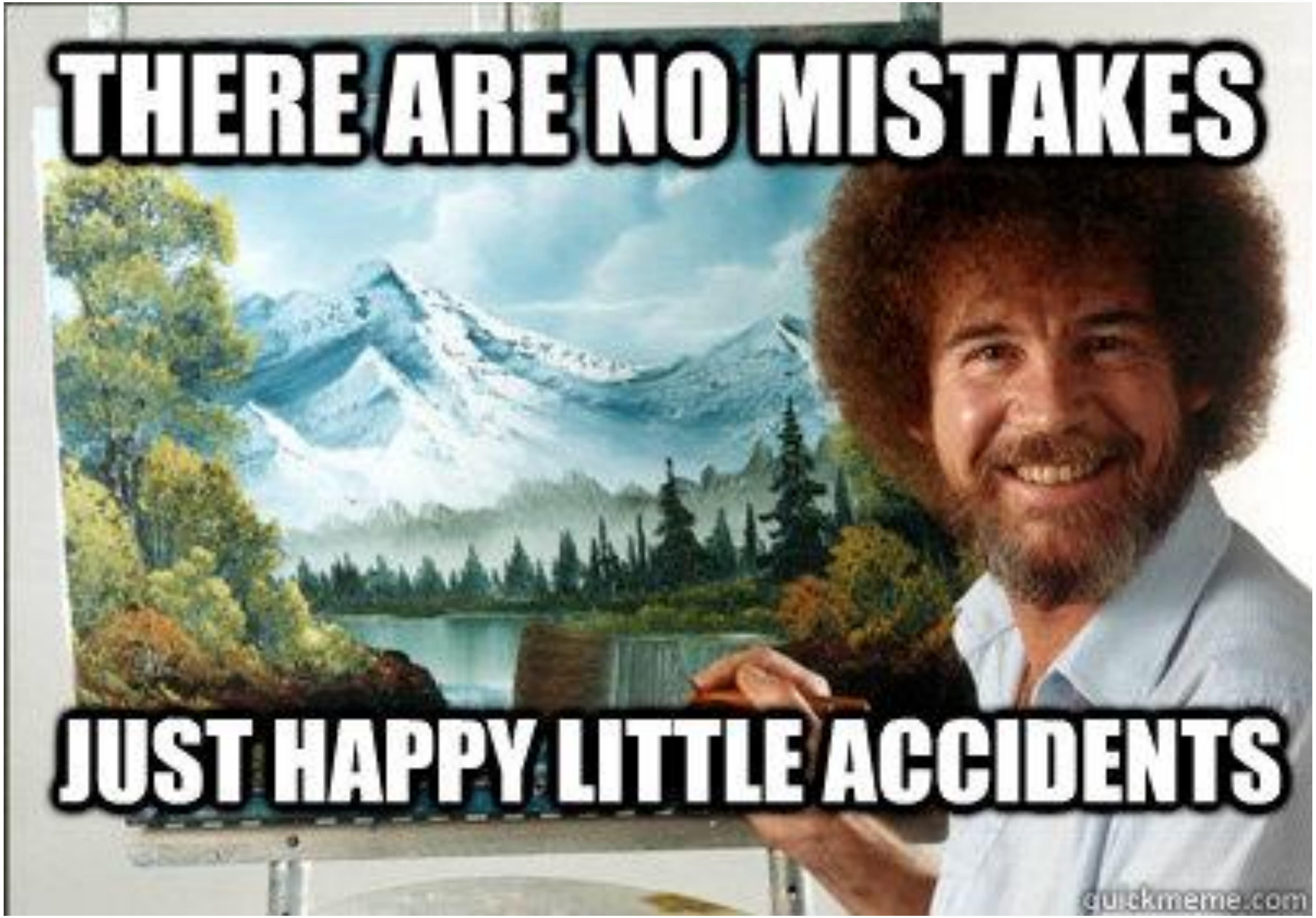
- Simplicity
- Modular
- Flexible
- Versioning

- Powershell/Bash
- VS Code
- GitHub
- Azure Automation, Ansible, Terraform

Steps to Implement IaC

1. Find something easy to automate – low effort, low risk
2. Set the right expectations – experimentation is necessary
3. Prove that it works – show the time savings and effort needed
4. Don’t be shy about it – advocate
5. Do it again




THERE ARE NO MISTAKES
JUST HAPPY LITTLE ACCIDENTS
quickmeme.com

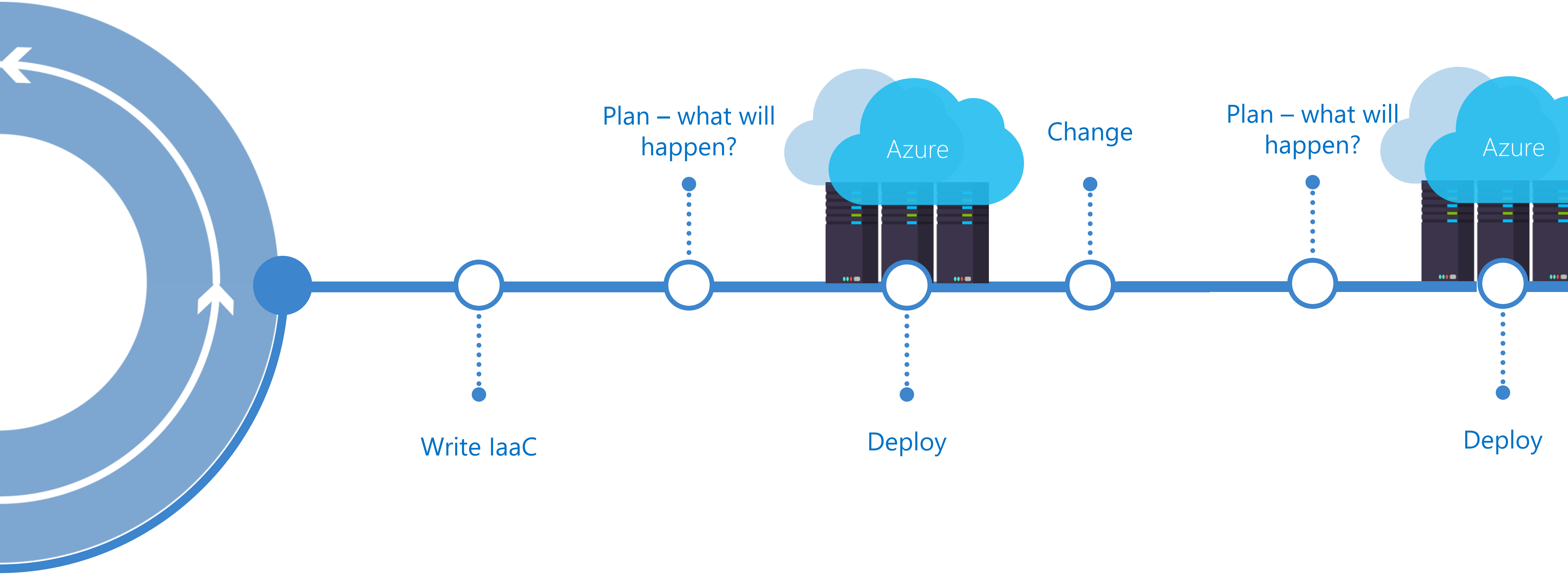
Terraform

Microsoft

Terraform

Write, *plan* and create infrastructure as code

Same workflow for all deployment scenarios



Terraform

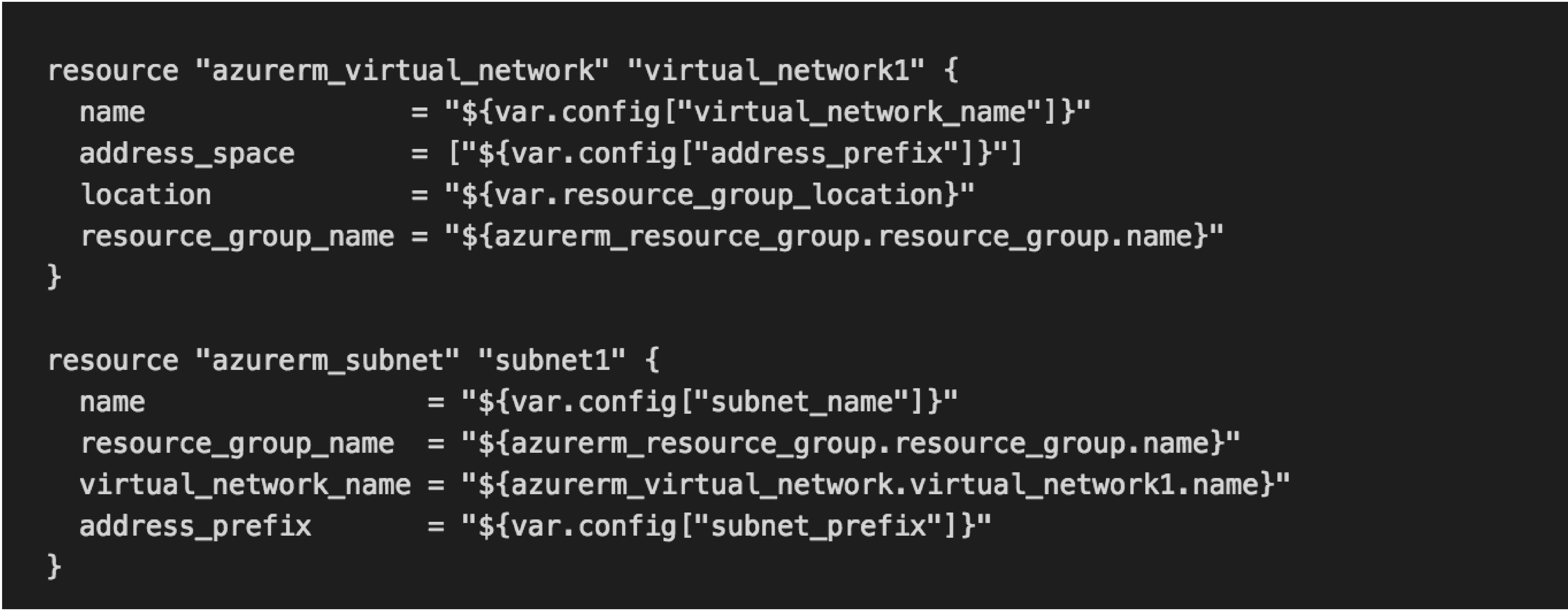
- Ansible, Chef, Puppet, Saltstack have a focus on automating the installation and configuration of software
- Keeping the machines in compliance, in a certain state
- Terraform can automate provisioning of the infrastructure itself

  e.g. Using the GCP, DigitalOcean, Azure API
- Works well with automation software like ansible to install software after the infrastructure is provisioned

Terraform

- Everything in one file is not great.
- Use variables to hide secrets
- You don't want your Azure credentials in your git repository
- Use variables for elements that might change
- Use variables to make it yourself easier to reuse terraform files

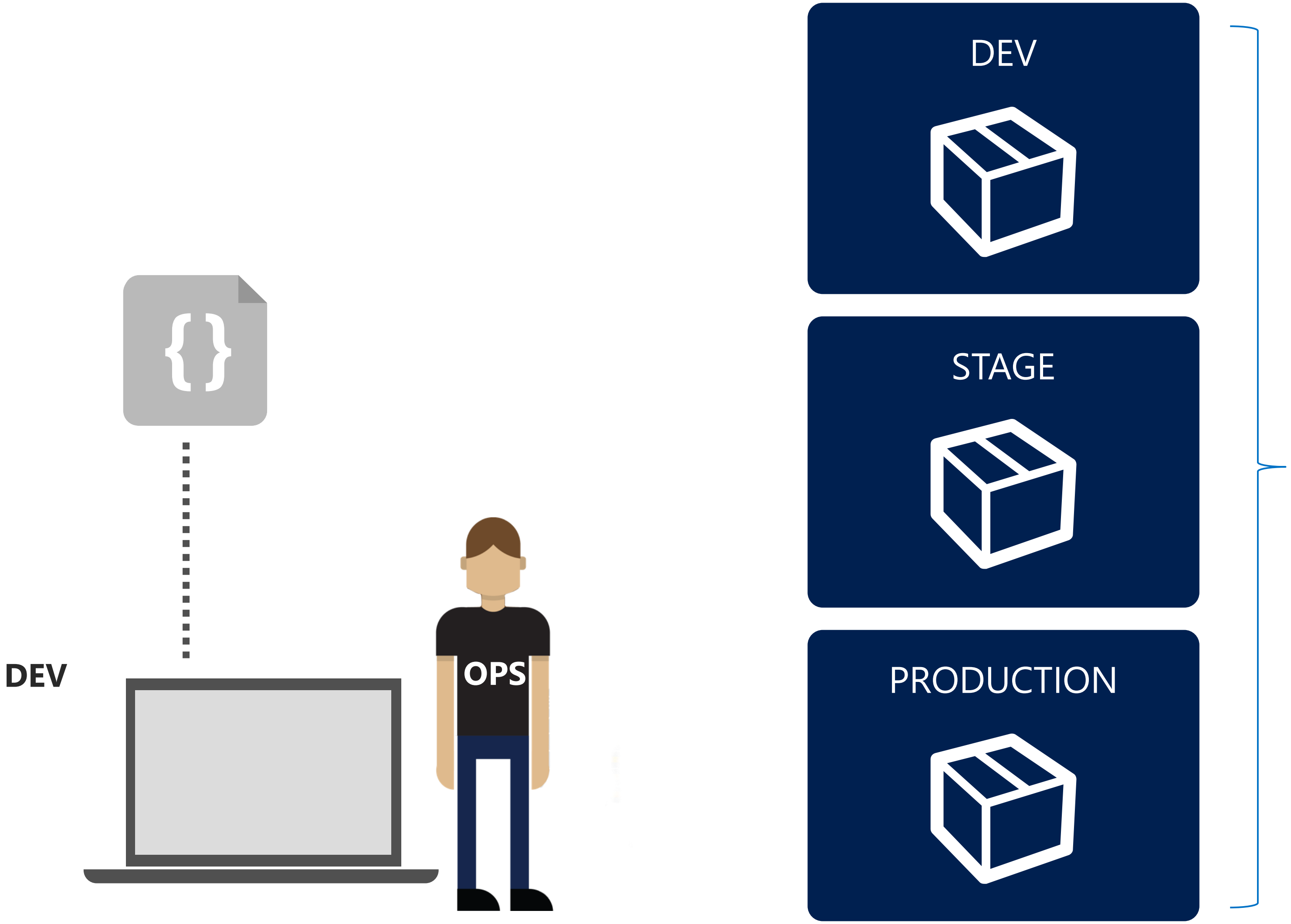
Creating Terraform Templates



```
resource "azurerm_virtual_network" "virtual_network1" {
  name                = "${var.config["virtual_network_name"]}"
  address_space       = ["${var.config["address_prefix"]}"]
  location            = "${var.resource_group_location}"
  resource_group_name = "${azurerm_resource_group.resource_group.name}"
}

resource "azurerm_subnet" "subnet1" {
  name                 = "${var.config["subnet_name"]}"
  resource_group_name  = "${azurerm_resource_group.resource_group.name}"
  virtual_network_name = "${azurerm_virtual_network.virtual_network1.name}"
  address_prefix       = "${var.config["subnet_prefix"]}"
}
```

Environment Parity



Same Terraform Code

Replace Dedicated Staging

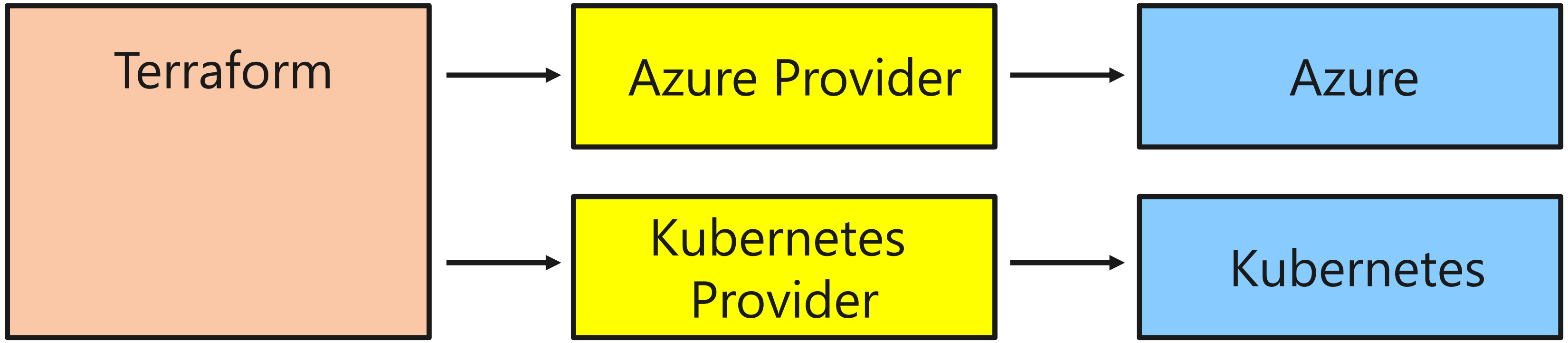
Speed Up Infra provisioning

Known Providers

Providers

What is a Terraform provider?

- Terraform 'extensions' for deploying resources
- Manages cloud / endpoint specific API interactions
- Available for major clouds and other platforms
- Hand authored (azurerm)

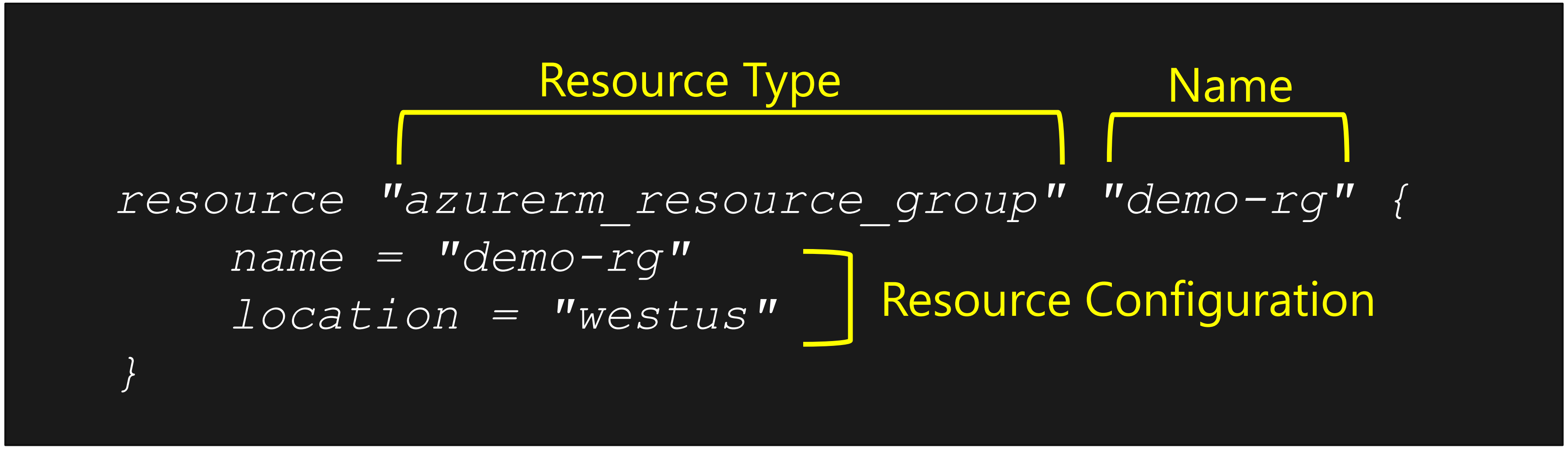


@nepeters

Basic resource creation

Deployment foundations.

- Resource Type: required provider
- Name: internal name
- Configuration: deployment details



```
resource "azurerm_resource_group" "demo-rg" {
    name = "demo-rg"
    location = "westus"
}
```

Basic Terraform commands

Once we have authored, how do we deploy?

- Terraform init – initializes working directory
- Terraform plan – pre-flight validation
- Terraform apply – deploys and updates resources
- Terraform destroy – removes all resources defined in a configuration

Variables and output

- Input variables: parameters for Terraform modules
- Environment variables: TF_VAR_azureclientid
- Output: Displayed and retrieved from state

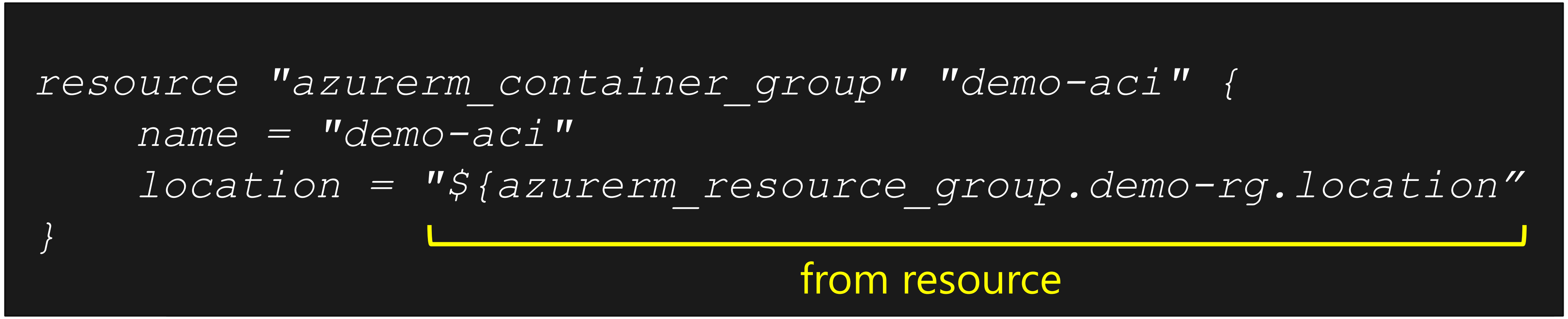
```
$ TF_VAR_azureclientid = “00000000-0000-0000-0000-00000000000”

variable “azureclientid” {}
```

String Interpolation

Interpolation: the insertion of something of a different nature into something else.

- Variables
- Other resources
- Functions: ${count.index + 1}
- Others (Docs)



```
resource "azurerm_container_group" "demo-aci" {
    name = "demo-aci"
    location = "${azurerm_resource_group.demo-rg.location”
}
```

from resource

Dependencies

How are resource dependencies managed?

- Implicit – derived from interpolation
- Explicit – hard coded / explicit dependency

```
resource "azurerm_container_group" "demo-aci” {
    name = “demo-aci”

    depends_on = [“azure_cosmosdb_account.vote-db”]
}
```

State / Backend

What is Terraform state and why store it remotely?

Issues with local state:

- No collaboration
- Easy to delete / loose
- State files include secrets

Alternative:

- Store state in a backend (Azure Storage)

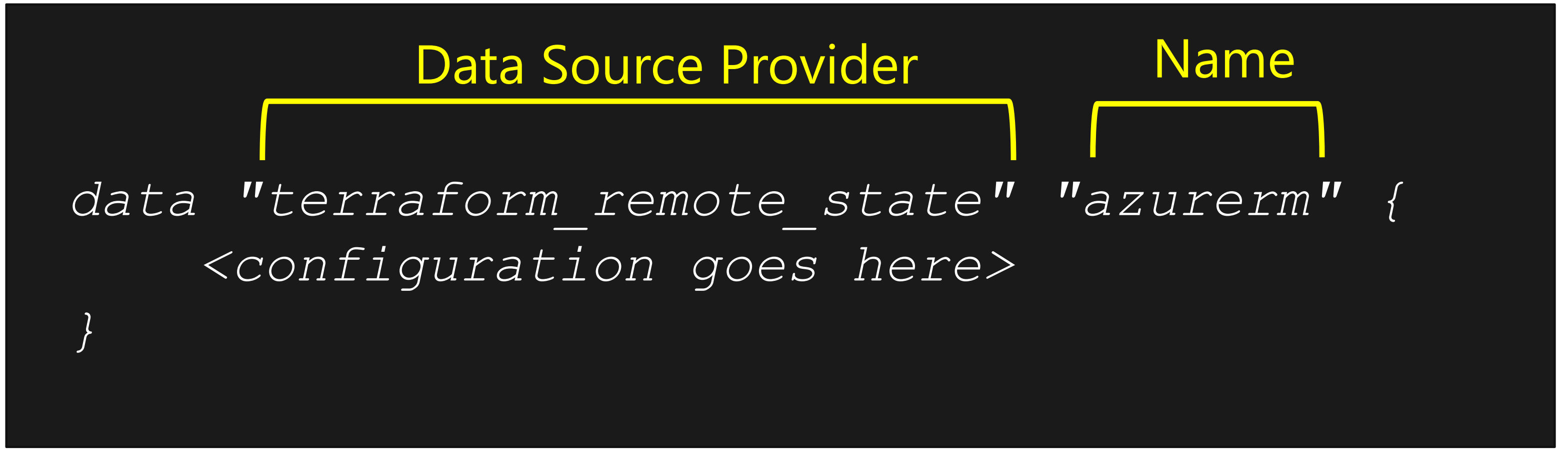
State / Backend

- You can keep the terraform.tfstate in version control
- e.g. git
- It gives you a history of your terraform.tfstate file (which is just a big JSON file)
- It allows you to collaborate with other team members
- Unfortunately, you can get conflicts when 2 people work at the same time
- Local state works well in the beginning, but when you project becomes bigger, you might want to store your state remote

Data Sources

What is a Terraform data source?

- External data source for Terraform configuration
- Uses a provider just like in resource creation



```
data "terraform_remote_state" "azurerm" {
    <configuration goes here>
}
```

```
"${data.terraform_remote_state.azurerm.resource-group}"
```

Automation and process integration

Once we are cooking, many opportunities for automation and process integration.

- Terraform Backends
- Environment variables
- GitHub
- Web Hooks
- Azure DevOps
- Etc.

Q&A?

Microsoft