plain concepts

Scripts básicos Azure

- Cómo hacer algunas tareas en Azure

CREAR UNA RED VIRTUAL

```
New-AzureRmResourceGroup -ResourceGroupName “ALMrecursos” -Location NorthEurope

$virtualNetwork = New-AzureRmVirtualNetwork -ResourceGroupName “ALMrecursos” -Location NorthEurope -Name myVirtualNetwork -AddressPrefix 10.0.0.0/16

$subnetConfigPublic = Add-AzureRmVirtualNetworkSubnetConfig -Name Public -AddressPrefix 10.0.0.0/24 -VirtualNetwork $virtualNetwork

$subnetConfigPrivate = Add-AzureRmVirtualNetworkSubnetConfig -Name Private -AddressPrefix 10.0.1.0/24 -VirtualNetwork $virtualNetwork

$virtualNetwork | Set-AzureRmVirtualNetwork

Remove-AzureRmResourceGroup -Name ALMrecursos -Force
```

CREAR UNA VM COMPLETA (1/2)

```
$resourceGroup = "ALMgrupoRecursos"
$location = "northeurope"
$vmName = "testVM"
$cred = Get-Credential -Message "Usuario para la VM."

New-AzureRmResourceGroup -Name $resourceGroup -Location $location
$subnetConfig = New-AzureRmVirtualNetworkSubnetConfig -Name mySubnet -AddressPrefix 192.168.1.0/24
$vnet = New-AzureRmVirtualNetwork -ResourceGroupName $resourceGroup -Location $location -Name MYvNET -AddressPrefix 192.168.0.0/16 -Subnet $subnetConfig
$pip = New-AzureRmPublicIpAddress -ResourceGroupName $resourceGroup -Location $location -Name "mypublicdns$(Get-Random)" -AllocationMethod Static -IdleTimeoutInMinutes 4

#regla de entrada RDP para NSG
$nsgRuleRDP = New-AzureRmNetworkSecurityRuleConfig -Name myNetworkSecurityGroupRuleRDP  -Protocol Tcp -Direction Inbound -Priority 1000 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * -DestinationPortRange 3389 -Access Allow
```

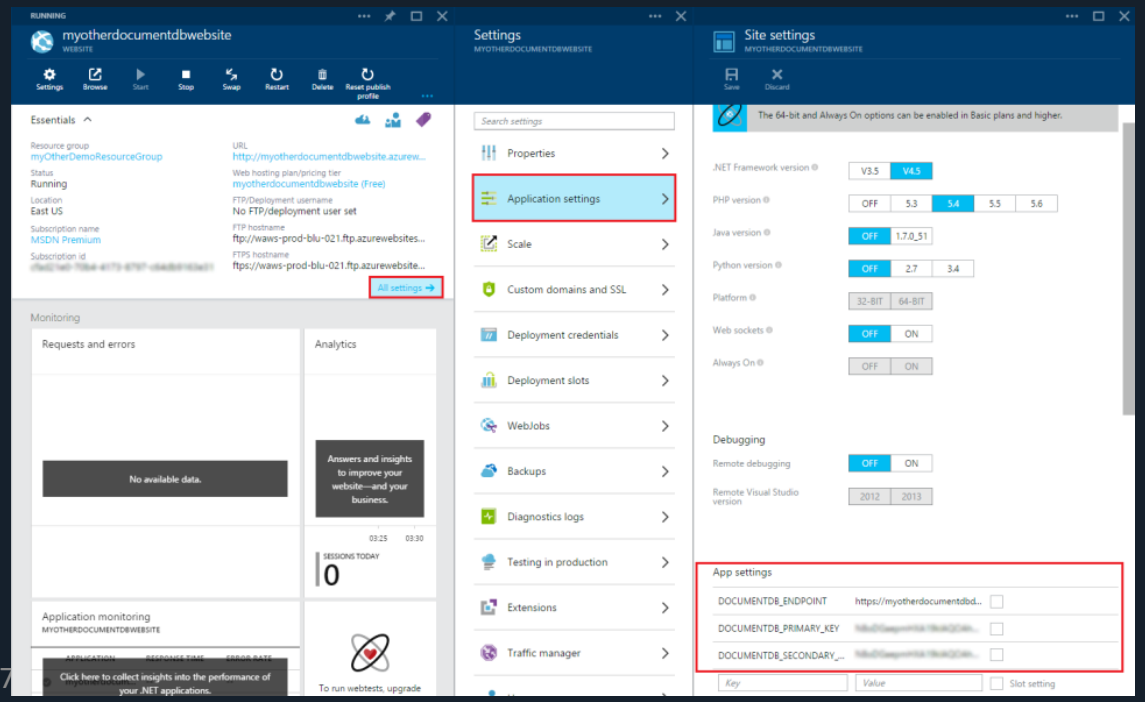
CREAR UNA VM COMPLETA (2/2)

```
#NSG
$nsg = New-AzureRmNetworkSecurityGroup -ResourceGroupName $resourceGroup -Location $location -Name myNetworkSecurityGroup -SecurityRules $nsgRuleRDP

#Crea Vnic
$nic = New-AzureRmNetworkInterface -Name myNic -ResourceGroupName $resourceGroup -Location $location -SubnetId $vnet.Subnets[0].Id -PublicIpAddressId $pip.Id -NetworkSecurityGroupId $nsg.Id

#crea configuracion de VM
$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize Standard_D1 | `
Set-AzureRmVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred | `
Set-AzureRmVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer -Skus 2016-Datacenter -Version latest | `
Add-AzureRmVMNetworkInterface -Id $nic.Id

#Crea VM
New-AzureRmVM -ResourceGroupName $resourceGroup -Location $location -VM $vmConfig
```

plain concepts

TEMPLATE STRUCTURE

Azure Resource Manager allows you to provision your applications using a declarative template. In a single template, you can deploy multiple services along with their dependencies. You use the same template to repeatedly deploy your application during every stage of the application life cycle.



plain concepts

INFRAESTRUCTURE AS CODE

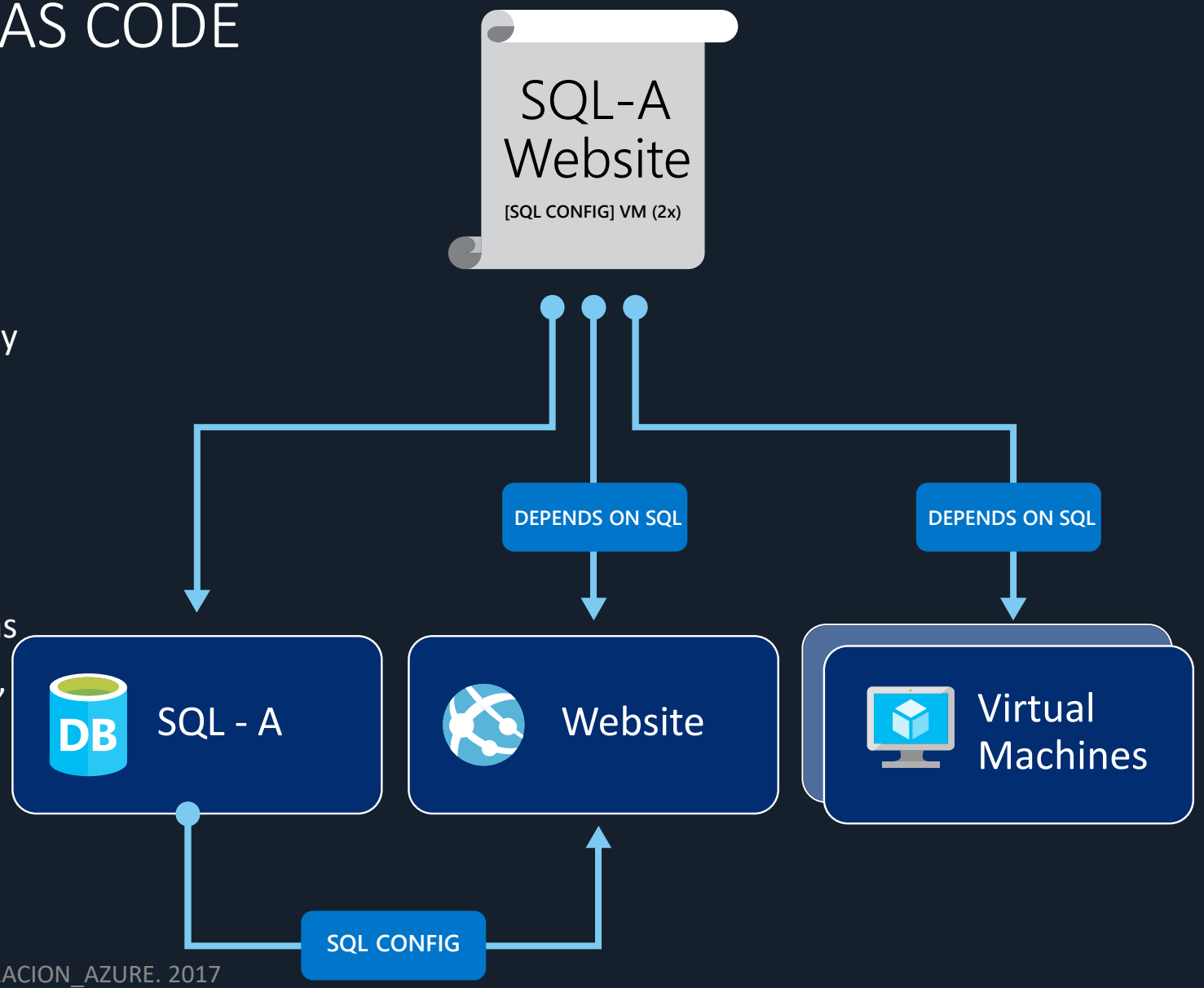
Pueden:

- Simplificar el despliegue
- Simplifica el roll-back
- Proporciona cross-resource Configuration y soporte de actualización

Contienen:

- Especificaciones a recursos y dependencias (VMs, websites, DBs) y conexiones (config, LB sets)
- Entrada/salida parametrizada

plain concepts



PLANTILLA MÍNIMA

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
    },
    "variables": {
    },
    "resources": [
    ],
    "outputs": {
    }
}
```

plain concepts

EJEMPLO DE PLANTILLA

```
{
 "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
 "contentVersion": "1.0.0.0",
 "parameters": {
  "storageAccountType": {
   "type": "string",
   "defaultValue": "Standard_LRS",
   "allowedValues": [
    "Standard_LRS",
    "Standard_GRS",
    "Standard_ZRS",
    "Premium_LRS"
   ],
   "metadata": {
    "description": "Storage Account type"
   }
  }
 },
 "variables": {
  "storageAccountName": "[concat(uniquestring(resourceGroup().id), 'standardsa')]"
 },
 "resources": [
  {
   "type": "Microsoft.Storage/storageAccounts",
   "name": "[variables('storageAccountName')]",
   "apiVersion": "2016-01-01",
   "location": "[resourceGroup().location]",
   "sku": {
```

plain concepts

DESPLIEGUE DE PLANTILLAS CON POWERSHELL

1. Iniciar sesion en Azure
2. Usar un grupo de recursos (o crear uno) como contenedor para los recursos desplegados.
3. Desplegar en el grupo de recursos elegido la plantilla que define los recursos a crear

```
Login-AzureRmAccount

New-AzureRmResourceGroup -Name ExampleResourceGroup -Location "South Central US"
New-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName
ExampleResourceGroup `
  -TemplateFile c:\MyTemplates\storage.json -storageAccountType Standard_GRS
```

plain concepts

CÓMO COMPROBAR UNA PLANTILLA CON POWERSHELL

Para comprobar la plantilla y parámetros sin desplegar ningún recurso, está el cmdlet Test-AzureRmResourceGroupDeployment.

```
Test-AzureRmResourceGroupDeployment -Name ExampleDeployment -ResourceGroupName
ExampleResourceGroup `
  -TemplateFile c:\MyTemplates\storage.json -storageAccountType Standard_GRS
```

Si no hay errores, el cmdlet finaliza sin respuesta. Si se detecta un error, devuelve un mensaje adecuado.

plain concepts

Gracias!

Barcelona

Bilbao

Madrid

Sevilla

Dubai

London

Seattle

plain concepts

plain concepts

Apéndices

- Información adicional

plain concepts

Módulos en PowerShell

- Ampliar la funcionalidad de la Shell

MÓDULOS

Creación de módulos

- Se pueden crear en la carpeta de cada usuario o a nivel de Sistema.
- Módulos de usuario:
  - Se almacenan en C:\users\<UserName>\Documents\WindowsPowerShell\Modules\<ModuleName>\<**ModuleName.psm1**>
  - Archivo de definición <**ModuleName.psd1**>
- Módulos de Sistema:
  - Se guardan en C:\Windows\System32\WindowsPowerShell\v1.0\Modules\
- Se pueden crear como un script estándar, pero se guardan con una extension particular.

MÓDULOS

Creación de módulos

*C:\Users\amarquez\Documents\WindowsPowerShell\Modules\plainIT\plainIT.psm1 - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

new 1 | plainIT.psm1

```
1  ##############################################################################
2  ############# PlainIT Module (by Amarquez) ###################################
3  ##############################################################################
4
5  ############# CUSTOMIZATION ##################################################
6  #Set Title for easy window ID
7  function Set-Title(){$a = (Get-Host).UI.RawUI;$title="Windows PowerShell";if($args.
   length -ge 0){$title="";$args|foreach-object{$title=$title+" "+$_}};$a.windowTitle=
   $title;}
8  function Get-WinVersion(){(Get-WmiObject -class Win32_OperatingSystem).Caption}
9
```

plain concepts

MÓDULOS

Módulos existentes

Administrator: Windows PowerShell

```
PS C:\>
PS C:\> Get-Module -ListAvailable

ModuleType Name                        ExportedCommands
---------- ----                        ----------------
Manifest   ActiveDirectory             {}
Manifest   ADRMS                       {}
Manifest   AppLocker                   {}
Manifest   BestPractices               {}
Manifest   BitsTransfer                {}
Manifest   GroupPolicy                 {}
Manifest   NetworkLoadBalancingCl...   {}
Manifest   PSDiagnostics               {}
Manifest   ServerManager               {}
Manifest   TroubleshootingPack         {}


PS C:\> Import-Module act*
PS C:\> Get-Command -Module Act*

CommandType     Name                              Definition
-----------     ----                              ----------
Cmdlet          Add-ADComputerServiceAccount      Add-ADComputerServiceAccount...
Cmdlet          Add-ADDomainControllerPasswo...   Add-ADDomainControllerPasswo...
Cmdlet          Add-ADFineGrainedPasswordPol...   Add-ADFineGrainedPasswordPol...
```

plain concepts

MÓDULOS: EJEMPLO, EXTENDER WINDOWS

Ampliar la funcionalidad de Windows Server con el módulo “ServerManager”

Administrator: Windows PowerShell NP

```
PS C:\>
PS C:\> Import-Module ServerManager
PS C:\> gcm -Module Server*

CommandType     Name                                       Definition
-----------     ----                                       ----------
Cmdlet          Add-WindowsFeature                         Add-WindowsFeature [-Name] <Featu.
Cmdlet          Get-WindowsFeature                         Get-WindowsFeature [[-Name] <Stri.
Cmdlet

PS C:\>
```

Administrator: Windows PowerShell NP

```
PS C:\> Get-WindowsFeature

Display Name                                            Name
------------                                            ----
[ ] Active Directory Certificate Services               AD-Certificate
    [ ] Certification Authority                         ADCS-Cert-Authority
    [ ] Certification
    [ ] Online Respon
    [ ] Network Devic
    [ ] Certificate E
    [ ] Certificate E
[X] Active Directory
    [X] Active Direct
```

Administrator: Windows PowerShell NP

```
PS C:\> Add-WindowsFeature telnet-client, telnet-server -Restart

Success Restart Needed Exit Code Feature Result
------- -------------- --------- --------------
True    No             Success   {Telnet Server, Telnet Client}


PS C:\>
```

plain concepts