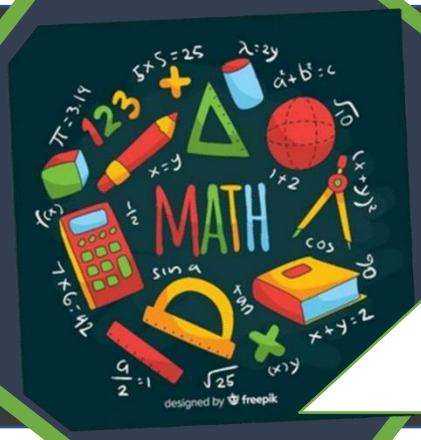
UNIVERSIDAD DE COLIMA



09/05/2022

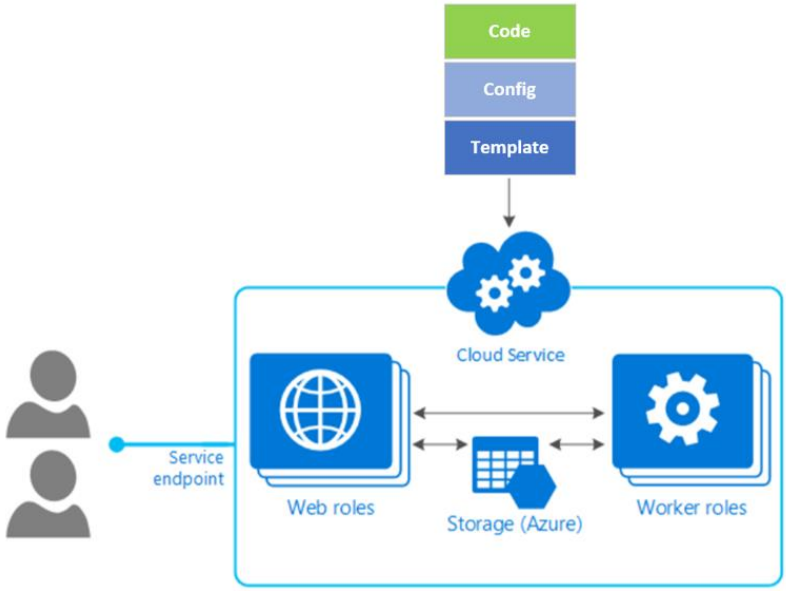
Computo en la nube

CARLOS ALEJANDRO BALTAZAR PADILLA

6 B

Servicios en la nube (soporte ampliado) es un nuevo modelo de implementación basado en Azure Resource Manager para el producto Servicios en la nube de Azure y ahora está disponible de forma general. Los servicios en la nube (soporte ampliado) tienen la ventaja principal de proporcionar resiliencia regional junto con la paridad de funciones con los servicios en la nube de Azure implementados mediante Azure Service Manager. También ofrece algunas capacidades de ARM, como acceso y control basados en roles (RBAC), etiquetas, políticas y plantillas de implementación compatibles. Con este cambio, el modelo de implementación basado en Azure Service Manager para Cloud Services pasará a llamarse Cloud Services (clásico). Conservará la capacidad de crear e implementar rápidamente sus aplicaciones y servicios web y en la nube. Podrá escalar su infraestructura de servicios en la nube en función de la demanda actual y asegurarse de que el rendimiento de sus aplicaciones pueda mantenerse al mismo tiempo que reduce los costos.

lo que no cambia Usted crea el código, define las configuraciones y lo implementa en Azure. Azure configura el entorno informático, ejecuta su código, luego lo supervisa y lo mantiene por usted. Cloud Services (soporte extendido) también admite dos tipos de roles, web y trabajador. No hay cambios en el diseño, la arquitectura o los componentes de los roles web y de trabajo. Los tres componentes de un servicio en la nube, la definición del servicio (.csdef), la configuración del servicio (.cscfg) y el paquete del servicio (.cspkg) se transfieren y no hay cambios en los formatos. No se requieren cambios en el código de tiempo de ejecución ya que el plano de datos es el mismo y el plano de control solo cambia. Las versiones de Azure GuestOS y las actualizaciones asociadas están alineadas con los servicios en la nube (clásico) El proceso de actualización subyacente con respecto a los dominios de actualización, cómo procede la actualización, la reversión y los cambios de servicio permitidos durante una actualización no cambian.



Requisitos previos para implementar Azure Cloud Services (soporte extendido)

1) red virtual Las implementaciones de Cloud Service (soporte extendido) deben estar en una red virtual. La red virtual se puede crear a través de Azure Portal, PowerShell, CLI de Azure o Plantilla ARM. También se debe hacer referencia a la red virtual y las subredes en la configuración del servicio (.cscfg) en la sección NetworkConfiguration. Para redes virtuales que pertenecen al mismo grupo de recursos que el servicio en la nube, es suficiente hacer referencia solo al nombre de la red virtual en el archivo de configuración del servicio (.cscfg). Si la red virtual y el servicio en la nube se encuentran en dos grupos de recursos diferentes, se debe especificar el identificador completo de Azure Resource Manager de la red virtual en el archivo de configuración del servicio (.cscfg). Red virtual ubicada en el mismo grupo de recursos

<VirtualNetworkSite name="<vnet-name>"/>

<AddressAssignments>

<InstanceAddress roleName="<role-name>">

<Subnets>

<Subnet name="<subnet-name>"/>

</Subnets>

</InstanceAddress>

</AddressAssignments>

Virtual network located in different resource group

XMLCopy

```
<VirtualNetworkSite name="/subscriptions/<sub-id>/resourceGroups/<rg-
name>/providers/Microsoft.Network/virtualNetworks/<vnet-name>"/>
   <AddressAssignments>
     <InstanceAddress roleName="<role-name>">
       <Subnets>
        <Subnet name="<subnet-name>"/>
       </Subnets>
     </InstanceAddress>
   </AddressAssignments>
```

2) Remove the old plugins

Remove old remote desktop settings from the Service Configuration (.cscfg) file.

XMLCopy

```
<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.Enabled" value="true"
/>
<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountUsername"
value="gachandw" />
<Setting
name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountEncryptedPassword"
value="XXXX" />
<Setting name="Microsoft.WindowsAzure.Plugins.RemoteAccess.AccountExpiration"
value="2021-12-17T23:59:59.0000000+05:30" />
<Setting name="Microsoft.WindowsAzure.Plugins.RemoteForwarder.Enabled"
value="true" />
```

Remove old diagnostics settings for each role in the Service Configuration (.cscfg) file.

XMLCopy

```
<Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
```

Required Service Definition file (.csdef) updates

Note

Changes in service definition file (.csdef) requires the package file (.cspkg) to be generated again. Please build and repackage your .cspkg post making the following changes in the .csdef file to get the latest settings for your cloud service

1) Virtual Machine sizes

The following sizes are deprecated in Azure Resource Manager. However, if you want to continue to use them update the `vmsize` name with the associated Azure Resource Manager naming convention.

Previous size name	Updated size name
ExtraSmall	Standard_A1_v2
Small	Standard_A1_v2
Medium	Standard_A2_v2
Large	Standard_A4_v2
ExtraLarge	Standard_A8_v2
A5	Standard_A2m_v2
A6	Standard_A4m_v2
A7	Standard_A8m_v2
A8	Deprecated
A9	Deprecated

Previous size name	Updated size name
A10	Deprecated
A11	Deprecated
MSODSG5	Deprecated

For example, `<WorkerRole name="WorkerRole1" vmsize="Medium"` would become `<WorkerRole name="WorkerRole1" vmsize="Standard_A2"`.

Note

To retrieve a list of available sizes see **Resource Skus - List** and apply the following filters:

```
ResourceType = virtualMachines
VMDeploymentTypes = PaaS
```

2) Remove old remote desktop plugins

Deployments that utilized the old remote desktop plugins need to have the modules removed from the Service Definition (.csdef) file and any associated certificates.

XMLCopy

```
<Imports>
<Import moduleName="RemoteAccess" />
<Import moduleName="RemoteForwarder" />
</Imports>
```

Deployments that utilized the old diagnostics plugins need the settings removed for each role from the Service Definition (.csdef) file

XMLCopy

```
<Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" />
```

Access Control

La suscripción que contiene recursos de red debe tener acceso de colaborador de red o superior para servicios en la nube (soporte extendido). Para obtener más detalles sobre, consulte los roles integrados de RBAC

Creación de Key Vault

Key Vault se utiliza para almacenar certificados asociados a servicios en la nube (soporte ampliado). Agregue los certificados a Key Vault, luego haga referencia a las huellas

digitales del certificado en el archivo de configuración del servicio. También debe habilitar las 'políticas de acceso' de Key Vault (en el portal) para 'Azure Virtual Machines para implementación' para que el recurso de servicios en la nube (soporte extendido) pueda recuperar el certificado almacenado como secretos de Key Vault. Puede crear un almacén de claves en Azure Portal o mediante PowerShell. El almacén de claves debe crearse en la misma región y suscripción que el servicio en la nube. Para obtener más información, consulte Uso de certificados con Azure Cloud Services (soporte ampliado).

ServiceDefinition.csdef

The **ServiceDefinition.csdef** file specifies the settings that are used by Azure to configure a cloud service. The Azure Service Definition Schema (.csdef File) provides the allowable format for a service definition file. The following example shows the settings that can be defined for the Web and Worker roles:

XMLCopy

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="MyServiceName"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WebRole name="WebRole1" vmsize="Standard_D1_v2">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="HttpIn" endpointName="HttpIn" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="HttpIn" protocol="http" port="80" />
      <InternalEndpoint name="InternalHttpIn" protocol="http" />
    </Endpoints>
    <Certificates>
      <Certificate name="Certificate1" storeLocation="LocalMachine"
storeName="My" />
    </Certificates>
    <Imports>
      <Import moduleName="Connect" />
      <Import moduleName="Diagnostics" />
      <Import moduleName="RemoteAccess" />
      <Import moduleName="RemoteForwarder" />
    </Imports>
    <LocalResources>
      <LocalStorage name="localStoreOne" sizeInMB="10" />
      <LocalStorage name="localStoreTwo" sizeInMB="10"
cleanOnRoleRecycle="false" />
    </LocalResources>
    <Startup>
```

```
      <Task commandLine="Startup.cmd" executionContext="limited"
taskType="simple" />
    </Startup>
  </WebRole>

  <WorkerRole name="WorkerRole1">
    <ConfigurationSettings>
      <Setting name="DiagnosticsConnectionString" />
    </ConfigurationSettings>
    <Imports>
      <Import moduleName="RemoteAccess" />
      <Import moduleName="RemoteForwarder" />
    </Imports>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="tcp" port="10000" />
      <InternalEndpoint name="Endpoint2" protocol="tcp" />
    </Endpoints>
  </WorkerRole>
</ServiceDefinition>
```

You can refer to the Service Definition Schema) for a better understanding of the XML schema used here, however, here is a quick explanation of some of the elements:

Sites
Contains the definitions for websites or web applications that are hosted in IIS7.

InputEndpoints
Contains the definitions for endpoints that are used to contact the cloud service.

InternalEndpoints
Contains the definitions for endpoints that are used by role instances to communicate with each other.

ConfigurationSettings
Contains the setting definitions for features of a specific role.

Certificates
Contains the definitions for certificates that are needed for a role. The previous code example shows a certificate that is used for the configuration of Azure Connect.

LocalResources
Contains the definitions for local storage resources. A local storage resource is a reserved directory on the file system of the virtual machine in which an instance of a role is running.

Imports
Contains the definitions for imported modules. The previous code example shows the modules for Remote Desktop Connection and Azure Connect.

Startup
Contains tasks that are run when the role starts. The tasks are defined in a .cmd or executable file.

ServiceConfiguration.cscfg

The configuration of the settings for your cloud service is determined by the values in the **ServiceConfiguration.cscfg** file. You specify the number of instances that you want to deploy for each role in this file. The values for the configuration settings that you defined in the service definition file are added to the service configuration file. The thumbprints for any management certificates that are associated with the cloud service are also added to the file. The Azure Service Configuration Schema (.cscfg File) provides the allowable format for a service configuration file.

The service configuration file is not packaged with the application, but is uploaded to Azure as a separate file and is used to configure the cloud service. You can upload a new service configuration file without redeploying your cloud service. The configuration values for the cloud service can be changed while the cloud service is running. The following example shows the configuration settings that can be defined for the Web and Worker roles:

XMLCopy

```
<?xml version="1.0"?>
<ServiceConfiguration serviceName="MyServiceName"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration"
>
  <Role name="WebRole1">
    <Instances count="2" />
    <ConfigurationSettings>
      <Setting name="SettingName" value="SettingValue" />
    </ConfigurationSettings>

    <Certificates>
      <Certificate name="CertificateName" thumbprint="CertThumbprint"
thumbprintAlgorithm="sha1" />
      <Certificate
name="Microsoft.WindowsAzure.Plugins.RemoteAccess.PasswordEncryption"
         thumbprint="CertThumbprint" thumbprintAlgorithm="sha1" />
    </Certificates>
  </Role>
```

```
</ServiceConfiguration>
```

You can refer to the Service Configuration Schema for better understanding the XML schema used here, however, here is a quick explanation of the elements:

Instances
Configures the number of running instances for the role. To prevent your cloud service from potentially becoming unavailable during upgrades, it is recommended that you deploy more than one instance of your web-facing roles. By deploying more than one instance, you are adhering to the guidelines in the Azure Compute Service Level Agreement (SLA), which guarantees 99.95% external connectivity for Internet-facing roles when two or more role instances are deployed for a service.

ConfigurationSettings
Configures the settings for the running instances for a role. The name of the `<Setting>` elements must match the setting definitions in the service definition file.

Certificates
Configures the certificates that are used by the service. The previous code example shows how to define the certificate for the RemoteAccess module. The value of the *thumbprint* attribute must be set to the thumbprint of the certificate to use.

Note

The thumbprint for the certificate can be added to the configuration file by using a text editor. Or, the value can be added on the **Certificates** tab of the **Properties** page of the role in Visual Studio.

Defining ports for role instances

Azure allows only one entry point to a web role. Meaning that all traffic occurs through one IP address. You can configure your websites to share a port by configuring the host header to direct the request to the correct location. You can also configure your applications to listen to well-known ports on the IP address.

The following sample shows the configuration for a web role with a website and web application. The website is configured as the default entry location on port 80, and the web applications are configured to receive requests from an alternate host header that is called “mail.mysite.cloudapp.net”.

XMLCopy

```
<WebRole>
  <ConfigurationSettings>
    <Setting name="DiagnosticsConnectionString" />
  </ConfigurationSettings>
  <Endpoints>
    <InputEndpoint name="HttpIn" protocol="http" port="80" />
    <InputEndpoint name="Https" protocol="https" port="443" certificate="SSL"/>
    <InputEndpoint name="NetTcp" protocol="tcp" port="808" certificate="SSL"/>
  </Endpoints>
  <LocalResources>
    <LocalStorage name="Sites" cleanOnRoleRecycle="true" sizeInMB="100" />
  </LocalResources>
  <Site name="Mysite" packageDir="Sites\Mysite">
    <Bindings>
      <Binding name="http" endpointName="HttpIn" />
      <Binding name="https" endpointName="Https" />
      <Binding name="tcp" endpointName="NetTcp" />
    </Bindings>
  </Site>
  <Site name="MailSite" packageDir="MailSite">
    <Bindings>
      <Binding name="mail" endpointName="HttpIn"
hostHeader="mail.mysite.cloudapp.net" />
    </Bindings>
    <VirtualDirectory name="artifacts" />
    <VirtualApplication name="storageproxy">
      <VirtualDirectory name="packages"
packageDir="Sites\storageProxy\packages"/>
    </VirtualApplication>
  </Site>
</WebRole>
```

Changing the configuration of a role

You can update the configuration of your cloud service while it is running in Azure, without taking the service offline. To change configuration information, you can either upload a new configuration file, or edit the configuration file in place and apply it to your running service. The following changes can be made to the configuration of a service:

- **Changing the values of configuration settings**
  When a configuration setting changes, a role instance can choose to apply the change while the instance is online, or to recycle the instance gracefully and apply the change while the instance is offline.
- **Changing the service topology of role instances**
  Topology changes do not affect running instances, except where an instance is being removed. All remaining instances generally do not need to be

recycled; however, you can choose to recycle role instances in response to a topology change.

- **Changing the certificate thumbprint**
  You can only update a certificate when a role instance is offline. If a certificate is added, deleted, or changed while a role instance is online, Azure gracefully takes the instance offline to update the certificate and bring it back online after the change is complete.

Handling configuration changes with Service Runtime Events

The Azure Runtime Library includes the Microsoft.WindowsAzure.ServiceRuntime namespace, which provides classes for interacting with the Azure environment from a role. The RoleEnvironment class defines the following events that are raised before and after a configuration change:

- **Changing event**
  This occurs before the configuration change is applied to a specified instance of a role giving you a chance to take down the role instances if necessary.
- **Changed event**
  Occurs after the configuration change is applied to a specified instance of a role.

Note

Because certificate changes always take the instances of a role offline, they do not raise the RoleEnvironment.Changing or RoleEnvironment.Changed events.

ServicePackage.cspkg

Note

The maximum package size that can be deployed is 600MB

To deploy an application as a cloud service in Azure, you must first package the application in the appropriate format. You can use the **CSPack** command-line tool (installed with the Azure SDK) to create the package file as an alternative to Visual Studio.

CSPack uses the contents of the service definition file and service configuration file to define the contents of the package. **CSPack** generates an application package file (.cspkg) that you can upload to Azure by using the Azure portal. By default, the package is named `[ServiceDefinitionFileName].cspkg`, but you can specify a different name by using the `/out` option of **CSPack**.

CSPack is located at
`C:\Program Files\Microsoft SDKs\Azure\.NET SDK\[sdk-version]\bin\`

Note

CSPack.exe (on windows) is available by running the **Microsoft Azure Command Prompt** shortcut that is installed with the SDK.

Run the CSPack.exe program by itself to see documentation about all the possible switches and commands.

Tip

Run your cloud service locally in the **Microsoft Azure Compute Emulator**, use the **/copyonly** option. This option copies the binary files for the application to a directory layout from which they can be run in the compute emulator.

Example command to package a cloud service

The following example creates an application package that contains the information for a web role. The command specifies the service definition file to use, the directory where binary files can be found, and the name of the package file.

cmdCopy

```
cspack [DirectoryName]\[ServiceDefinition]
       /role:[RoleName];[RoleBinariesDirectory]
       /sites:[RoleName];[VirtualPath];[PhysicalPath]
       /out:[OutputFileName]
```

If the application contains both a web role and a worker role, the following command is used:

cmdCopy

```
cspack [DirectoryName]\[ServiceDefinition]
       /out:[OutputFileName]
       /role:[RoleName];[RoleBinariesDirectory]
       /sites:[RoleName];[VirtualPath];[PhysicalPath]
       /role:[RoleName];[RoleBinariesDirectory];[RoleAssemblyName]
```