|  |  |
| --- | --- |
|  | Cloud customer ITPros and developers have an experience identical to that which they experience in Microsoft Azure. This is fundamentally made possible because the portal environment is the actual portal from Azure. |
|  | * This is the Azure Portal * From this portal, Azure customers can create and manage their resources in the cloud. |
|  | * In Azure, resource groups are used to provide a wrapper around resources. This way, a customer can create a simple or complex set of resources in Azure and manage them as a whole. |
|  | If we take a look at this resource group… |
|  |  |
|  | We see that every individual component of this simple virtual machine can be managed individually…So if we look at the nic for this VM, we can see all the information about the nic as well as our management actions we can take. |
|  |  |
|  | Here’s the Azure Stack portal. It’s the same code as Azure, so the same experiences that we offer from our datacenters are available to be offered from yours. |
|  | Just as in Azure, we can look at resource groups… |
|  | Browse the same resource group… |
|  | And manage the nic on the server in the exact same way we would in Azure. |
|  | Transition: But we’ve taken the consisitancy between Azure and Azure stack deeper than the UI. |
|  | We’ve taken consistency to the API level. With this, investments made against Azure can apply to Azure Stack. |
|  | Let’s look at one of those automations. This is a deployment script that does three things. It 1) registers the ISE with Azure or Azure stack. 2) sets the region, like West US, North Europe, or East Asia. And finally, deploys a template to the cloud.  Here, we’re going to register this ISE with Azure. |
|  | Next, we’ll specify that the region in Azure we want to deploy to, in this case, West US. |
|  | Next, we’ll run the deployment, specifying the parameters of the machine we want to create.  There goes the deployment, creating that VM in Azure. |
|  | Now we’re going to take this same script, and change it to be registered with an Azure Stack deployment. |
|  | We’re going to set the region we want to deploy to… |
|  | And then run the exact same deployment steps…  And there it goes, creating that virtual machine in an Azure Stack deployment. |
|  | Enabling developers to go fast means that they have to have access to resources from the tools they need to do their job. |
|  | This is a Visual Studio development environment that is using the Azure SDK. |
|  | The project that is open is designing the deployment to an Azure cloud. So from their development environment, a developer can deploy their work… |
|  | And choose to deploy to the Azure cloud they are subscribed to.  Here you can see the Azure subscriptions available to this user as well as the Azure Stack subscriptions available. |
|  | Once that choice has been made, they simply finish setting up the deployment and deploy. |
|  | The deep consistency between Azure and Azure Stack means that the ecosystem of tools and templates that work against Azure can work against Azure Stack. |
|  | Let’s take a look at one of those tools.  Azure Stack has the same blob storage capabilities as Azure. If we look at our storage accounts in this Azure Stack deployment… |
|  | We can see that there is a storage container called myphotos. |
|  | Currently that container is empty. |
|  | This is the Azure Storage Explorer. It is a community tool for working with Storage in Azure. With no changes to the tool itself, all we did was connect it to the Azure Stack storage endpoint. So, once connected, we can choose the storage account we want to work with… |
|  | And see that the Myphotos container is available. |
|  | We can upload a file to the blob storage, in this case a photo. |
|  | When we switch back to the Azure Stack portal and refresh the container… |
|  | The photo we uploaded is in the cloud. We can copy the url to the photo…. |
|  | And paste it into a browser… |
|  | And see the picture we uploaded into Azure blob storage running in the datacenter. |