Azure Stack White Paper:

"Getting Ahead of the Game with Azure Stack Configuration"

Authors: Kath McBride, Joel Yoker, Paul Appleby

Date: Tuesday, February 27, 2017

Contents

[First Thoughts 3](#_Toc476051859)

[Azure Stack: Core concepts 6](#_Toc476051860)

[Interacting with Azure Stack 15](#_Toc476051861)

[Subscriptions 27](#_Toc476051862)

[Services: Resource Providers 37](#_Toc476051863)

[Quotas, Plans and Offers 45](#_Toc476051864)

[Delegation 53](#_Toc476051865)

[Tags, Policies, and Locks 55](#_Toc476051866)

[Pulling it all together: The Service Catalogue 60](#_Toc476051867)

[Lab Script Kit 73](#_Toc476051868)

# First Thoughts

This is the first of a multi-part paper detailing core concepts of Azure Stack, driven from information Microsoft Azure CAT team have had in discussions with customers. This is not intended to be prescriptive guidance, but rather help you understand foundational elements of Azure Stack to assist in evaluation and planning.

Microsoft Azure Stack is a solution that allows you to deploy set of key Azure services in your datacenter. There are many reasons why you may want to deploy Azure Stack like meeting compliance and security requirements, accommodating application latency issues.

In this series, I'll show you all the things you need to do to configure Azure Stack to offer services to your tenants. Some of these are things that the Azure team is responsible for in the public cloud, and most will never see. This series will serve as a guide on the steps you need to take as you plan and deploy your Azure Stack POC for your service or application.

While the current focus of this series is on Microsoft Azure Stack TP3 (link to download/docs), as we continue down the path to general availability, we will continue to add information. With this additional information, you can add new capabilities as they are lit up at each preview release, so please keep checking back!

We will look at considerations when designing and deploying services; everything from subscriptions, to offers and plans, quota design, and resource providers for the services you want to offer. Not only will we look at some of the key considerations, but also at the available dials and options and when and how you might choose to use them.

In short, the series is designed to provide you, the IT pro, with guidance on how to build an end to end solution for validation of an Azure Stack environment that best matches your business requirements.

This is not designed to be a best practice production design. It’s a simple design touching on many of the key areas to include when you are testing the Microsoft Azure Stack solution. It will give you insights into our vision and the many things you will want to think about as you do go about designing your production Azure Stack environment. We are still in early days with Azure Stack and as we get further into the cycle operational guidance will become available.

While this series will not provide specific guidance for the three scenarios below, we will provide some insights into key considerations for each.

1. Architecting for the Enterprise: Enterprises tend to deploy self-service solutions to meet the needs of their business. While these must be properly secured, focus tends to be on business drivers, reduced time to market, solution cost, availability, and scoping the solution to provide scalability in a cost-effective manner.
2. Architecting for the Service Provider: The focus of Service Provider tends to relate to the market the Service Provider is targeting. Any services provided must be available, secure in a multi-tenant environment, billable, and provide value to both customer and the Service Provider themselves.
3. Architecting for DevOps Scenario. This is an add-on scenario to either the enterprise or Service Provider solution in the sense that either the enterprise or Service Provider will deploy the base services for consumption by their developers. Developers are interested in the ability to rapidly spin up development environments, integration with the various tools that the development organization uses, and the ability to create applications that can be deployed to any cloud with one solution. Upgrade processes for the applications are also of key interest.

For the step-by-step guides, we will provide pointers to the many documents already released for Azure Stack to walk you through deployment of each component.

Let's look at what we plan to cover.

First thoughts: Azure Stack business model and how it impacts your design decisions, how will regions affect your design decisions, what you need to know about the marketplace

Interacting with Azure Stack: tools and methods of connecting to Azure Stack and why you may want to select various tools

Subscriptions: How, why, when, what to watch for.

Resource Providers; why to add a resource provider, use of multiple Resource Provider's in one environment and the considerations for compute/storage/network, any gotchas

Offers: When, why, and how to create them, why regional offers might be a good thing, as well as any best practices and gotchas

Plans: when and why to create new plans, when to use add on plans, how plans and offers are linked, why you might want regional plans, security for plans, gotchas

Quotas: when and why to create them, how they tie to plans, use of regional quotas, impact of changing global quotas, and what happens when a user runs out of quota

Tags, Policies, and Locks; what they are and when to use them.

Through the course of this series, we will build out an Azure Stack lab environment while accounting for real-world enterprise design goals. The goal is to d and end up with a lab configured with the basics, from which you can start to load your application that you have designed for and start testing that. If the lab I am building does not fit your requirements, you can easily change it using the information tables, and customize to fit your needs! Please note this is a lab and does not represent a production grade environment.

Remember that this is TP3 and is a great build to learn all about Azure Stack! This is an end-to-end lab for TP3 that will grow over time as new releases are available!

Building Your Azure Stack Lab

The best way to understand Azure Stack is to dive right in and begin using it. The first problem you'll run into is - where to start? With Azure in the public cloud many "back-end" actions are prepared for you by the service. As a tenant in Azure, you simply start consuming once your subscription is created for you through your enterprise enrollment. With Azure Stack, you are responsible for several areas before you start planning your solutions and using Azure services. This series will use a lab design to help you understand the various options and design decisions that need to be made when deploying Azure Stack. For a service provider, rather than looking at specific application(s) decisions will be based upon the services they want to sell to their customers. Our lab will be based upon an Enterprise customer named Contoso. Contoso has four departments that will use this Cloud. IT and development will use standard services as they build and deploy the applications into the Cloud. HR and Finance both have specific applications that they will require. It is important that we understand the applications or services our tenants will deploy, as these will have an impact on the design decision you make. Please keep in mind that while these decisions are specific to the creation of a lab, these are considerations you will need to think about as you design your production deployments.

What are Subscriptions and why should we start there? In Azure a Subscription is a *boundary*, specifically a boundary of scale (quota), billing and administration. While there are ways to use technologies such as Policy, Resource Tags and Role-based Access Control to address the three areas listed above, a Subscription provides a firm boundary for these elements within Azure.

Like Azure Subscriptions, Azure Stack implementations are expected follow common deployment models across customers:

There are two key categories of deployments, each with many common decisions, but some individual considerations:

**Customer-Owned Models (Enterprise):**

* **Application Owners hosting their application in a dedicated Azure Stack instance** – A cloud scenario where an application development team within a customer environment wants to take advantage of Azure Stack capabilities to host their application taking advantage of services managed by Enterprise IT.
* **Application Divisions (Business Unit) hosting their services in a shared Azure Stack instance** - A native private cloud scenario where an application development division within a customer environment wants to take advantage of Azure Stack capabilities to host their suite of applications and development/test environment

using services managed by Enterprise IT.

* **Enterprise IT extending their datacenter infrastructure to Azure Stack** - Typically a private cloud scenario where a mature Enterprise IT organization wants to extend their existing physical or virtual environment to Azure Stack to support the large number of growing IT requirements for their organization and its customers.

**Service Provider-Managed Models:**

* **Service Provider - Connect Through** – A Service Provider public cloud scenario where the service provider provides cloud services and/or hosts to tenant users. The service provider directly manages the customer application workloads deployed on top of Azure Stack services. In the “Connect-Through” model, the customer consumes the provider’s cloud services delivered via the provider’s network with end services hosted in the provider’s provisioned Azure Stack subscription. The Azure Stack subscription is created, owned, and managed by the service provider.
* **Service Provider - Connect To** – A Service Provider public cloud scenario where a service provider hosts and manages customer application workloads deployed on top of Azure Stack services. In the “Connect-To” model, the provider makes cloud services accessible directly to the customer’s network. The Azure Stack subscription is created, owned, and managed by the service provider but the customer consumes cloud services by interacting directly with Azure Stack cloud footprint.

In this series, we will focus on the customer owned scenario based upon business units. As we walk thru the design decisions in each of the chapters, we will continually refer back to this business scenario and explore how different capabilities can be used to address the needs of the business departments and of the IT team. Below is a diagram of the environment we will strive to build over the next several posts, with the goal to build out a lab configuration to support this type of Enterprise.

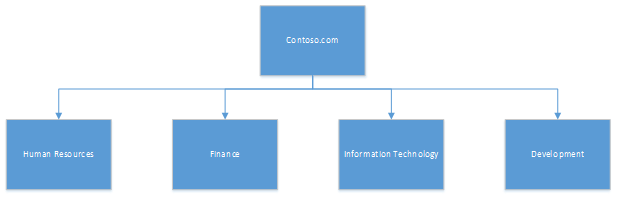


Figure 1. Contoso Labs high level design for this series

# Azure Stack: Core concepts

In Azure Stack subscriptions, offers, plans, quotas, and services are all connected and define the capabilities and quantities that a tenant can consume. Figure 1 is a logical diagram of this interaction between these capabilities, and shows a what is available in a single region. Thru this series will take a deeper look at each component.

Machine generated alternative text:
o 
Offers 
with Offas 
Q uotas As•ciated 
with ach Plan 
Tenantsignsupfora NbcriptOn 
2 Tenant then seæSfrom the 
EVE la bleoffas 
3. Eæh cffz isasoziated with a baæ 
plan Tid eæh 
quotæ 
92—-ctirg offer theservices 
within tte asoziæd fid the 
quot asigned t e 
asigned t tare nt 

Figure 2: View of how components interact

Azure Stack provides on-premises versions of several popular Azure Services, with more to be added in the future. These services include the commonly referenced cloud computing models:

* Infrastructure as a Service (IaaS)
* Platform as a Service (PaaS)

These models can be combined and integrated to build complex robust solutions for different audiences and use cases.

This is the key difference Azure Stack can bring to your organization since Azure Stack brings cloud-native PaaS services to your datacenter. Azure Stack services are fundamentally different than traditional IaaS services (or platforms hosted within a traditional OS) as they provide an Azure-consistent experience and API that can be consumed on-premises by your developers. These services include AppService (Web Apps, Mobile Apps, API Apps and Functions), Service Fabric, Key Vault, SQL and MySQL to name a few. All PaaS Services require IaaS to be in place in order to offer this service to your customers.

At the top of the stack is Software-as-a-Service SaaS). SaaS is the ability to provide software on a subscription basis that is centrally hosted and relies upon IaaS, and in most cases PaaS services. An example of SaaS service is providing Sharepoint services in Azure Stack and tenants can build their SharePoint sites on top of this service. Some notable examples can be found in the [Azure Stack GitHub Templates](https://aka.ms/ASWP_QSTemplates).

For more information on what will be available at GA please refer to the Ignite session:

[Explore Azure Stack “State of the Union” – Foundation 1](https://aka.ms/ASWP_Video2119).

Core Subscription Building Blocks:

**Roles**

By default, the Azure Active Directory (AAD) account that you use to deploy Azure Stack, will become the main management account for Azure Stack. This account will also be the subscription owner for the "Default Provider Subscription". Additional management users can be added with rights to manage Azure Stack via the portal or PowerShell.

Azure Stack has two general types of users:

* **Service administrator** can configure and manage resource providers, tenant offers, plans, services, quotas, and pricing
* **Tenant** acquires (or purchases) services that the service administrator offers. Tenants can provision, monitor, and manage services that they have subscribed to.

NOTE: The delegated administrator used in delegation scenarios falls under the category of tenant though with increased rights to be able to manage tenants and configure delegated offers for their tenants. Please see the post on delegation for details on what this is and why you may want it to be part of your design.

To add additional service administrators, configure them in the Default Provider Subscription as subscription admins. Three levels of rights are available out of the box:

* **Owner**: ability to manage everything including access to resources
* **Contributor:** ability to manage everything *except* access to resources
* **Reader:** view everything but cannot make changes

Role considerations:

* By default, the tenant that signs up for a subscription will become a subscription admin for that subscription

**Portal**

The portal allows administrators to complete the following:

* + Manage access
  + Manage accounts
  + Manage subscriptions
  + View usage summary
  + Provision/de-provision Azure Stack resource providers
  + Create plans and offers
  + Manage co-administrators on subscriptions
  + Allows a view of health of Azure Stack (not available in TP3 and out of scope for this blog series)

The portal is the method most of your users will use to access services you offer in Azure Stack. Two portals are available at TP3 accessible via separate URLs; one for administrators, and a second for tenants. Once the tenant logs in the portal is "scoped" based upon the username and the access rights you have defined for that user. For instance, a general user will access the portal and sign up for a subscription. Based on the plan and quota parameters you define, they will be given access to compute, storage, and network resources. These are the minimum services required to be able to deploy a virtual machine, storage account, or network as a tenant. Additional resource providers can be added (i.e. SQL Resource Provider) and access for your tenants provided via plans.

For a complete list of the resource providers that will be available at GA please refer to the Ignite On Demand session: Explore Microsoft Azure Stack “State of the Union” – Foundation 1

Tenants will be able to sign up for subscriptions via the administrator created offers, and deploy compute, network, and storage as well as any other services the administrator has enabled.

[Alternative ways of interacting with Azure Stack.](#_Interacting_with_Azure)

Scale Units, Regions and Clouds

**Scale Units**

In Azure Stack a Scale Unit is a defined collection of compute (servers), storage and networking that represents a unit of capacity expansion, an Azure fault domain and a homogenous set of hardware. One or more scale units can be contained within an Azure Stack Region support.

**Regions**

Although only a single region is currently available in the Azure Stack TP3 release, the concept of regions is important and worthy of discussion. Design decisions made for your lab should take into consideration multiple regions so as not to impact the ability to expand once multi region capabilities are available.

Azure Stack uses regions to represent set of scale units which share a common physical location and are logically managed by a single administrative entity. Multiple regions are optional and may not be required in your design. You may, for instance, decide to deploy a single Azure Stack instance to meet your business requirements. However, if you have multiple locations and datacenters, or you need to separate the services you offer for compliance reasons, you can elect to have multiple regions in your design. In Azure, regions are service-defined boundaries which help tenants make decisions on where they host workloads within the public cloud. Azure Stack brings new flexibility to both enterprises and service providers by allowing you to define regions based on your organization or tenant's needs.

Regions allow you to architect your Azure Stack solution to physically manage the delivery of services and applications in way which is visibly exposed to your tenants. Regions enable you to deliver services with the following factors in mind:

* **Network latency**: If response time for your applications is a concern, it is important that you deploy applications in close proximity to your users. If you have multiple locations, deploying applications closer to the user will ensure their experience is a good one. Azure Stack will provide methods for ensuring users are directed to the correct region. Synchronization of data, if applications reside in multiple regions, is also a consideration when it comes to regions.
* **Availability**: Applications need to be highly available. Implementing multiple regions enables you to reduce the impact of an outage should an outage occur. Users can be redirected to a secondary region, maintaining the availability of the application.   Additionally, you can make applications highly available in Azure Stack.
* **Workload segregation**: In some environments workloads must be kept isolated for security reasons. Multiple regions can be deployed to ensure hard boundaries between such applications.
* **Scale**: in large environments, your Azure Stack requirements may exceed the capabilities of a single Azure Stack region. In this scenario, you can use multiple regions to meet scale demand.

One thing to note about regions is that many of the capabilities in Azure Stack are region dependent. For instance, marketplace items, role based access, resource providers, quotas, offers and plans, and resource groups are deployed on a per-region basis. Users can be configured such that they can access multiple regions and different services in each, however keep in mind latency and workload segregation requirements when considering this.

Understanding the number of regions and their placement, as well as the user base that will access each region is important when starting to design your Azure Stack solution. As new functionality is released, and regions become available, your early planning will allow you to quickly add the regions you require. Also remember that while the notion of region is an integral part of the Azure Stack design, it will be implemented over time. Maximum scale for region and capacity for the first release has not been disclosed yet at this stage (TP3).

**Clouds**

An Azure Stack Cloud (or "instance") is a single instance of Azure Resource Manager (ARM). A cloud may contain one or more regions which are managed ARM and contains one or more scale units.

The relationship of these elements using a single region, is illustrated in the diagram below:

Scale Unit 1 
Aggregate Switch 
TOR Switch 
TOR Switch 
BMC switch 
Server 
Server 
Server 
Server 
Region 
Border 
Device 
Scale Unit n 
Aggregate Switch 
TOR Switch 
TOR Switch 
BMC switch 
Server 
Server 
Server 
Server 

Figure 3. Regions and Scale Units

Marketplace

The Marketplace in Azure Stack is the location in the portal a tenant will access to deploy Azure Stack resources like networks, virtual machines, storage, etc. When deploying, a tenant must have subscribed to an offer that has been defined by their administrator.

The Marketplace items that are available will be defined by the service administrator and will then be accessible by the tenant. Items in the Marketplace include but not limited to Virtual machines, networking components, storage, databases and websites if the PaaS Resource Providers are installed. For some, the defaults may be enough, however at minimum most would probably like to include a Windows 2016 server image.

We can think of the marketplace as the shopping catalogue of all the items that are available for a tenant to select from. The first thing one needs to do when designing their marketplace is determine what your users require. In an enterprise, you need to reach out to the various business units to determine exactly what each requires.

It is important to understand that all marketplace items are visible to all tenants. At this time, there is no way to prevent Business Unit A from seeing a custom image designed for Business Unit B. The Roles Based Access Security currently does not allow preventing some tenants from seeing and consuming items contained in the gallery. Assume if it is there and I can access the portal, I can use it. We can regulate to a certain degree through our design of the Offers and Plans, but it requires careful planning. For instance, if my HR department does not require access to deploy SQL databases, I can create an offer that does not include that service. I would also need to ensure that my offers are not public. This is a great use of the delegation model and custom URL's and assigned offers.

Coming in TP3 is the ability to use [Marketplace Syndication](https://aka.ms/aswp_marketplacesynd) to access a select set of images available in Azure for consumption in Azure Stack. This was demonstrated at Ignite, and was extremely well received. The list of what will be available is not yet published, but keep watch for announcements about this great new capability! It will be integrated, and the service admin will acquire the images he/she would like to offer to their tenants. Once they are selected, downloaded, and added to the Marketplace Gallery they will then become visible to tenants.

 We can easily add things like Linux images, additional operating systems, custom images, etc. to the marketplace as well without using the syndication feature. JSON templates allow us to declaratively describe the resources in a resource group using JSON. The easiest way to do this is to use Azure Stack specific templates available on [GitHub](https://aka.ms/ASWP_JSON). While we can use the templates that are available for Azure, we must review each carefully to ensure the parameters, variables, resources, and schema match those that are available in the current version of Azure Stack. To assist with this is the Template Validator is available in the [Azure Stack Tools](https://aka.ms/ASWP_Tools) repository.

Links to add Linux Images, as well as instructions to create your own Linux image can be found [here](https://aka.ms/ASWP_Linux).

For more information on the configuration of [JSON templates](https://aka.ms/ASWP_JSON) and adding items to the [Marketplace](https://aka.ms/ASWP_MP) refer to the linked video and documentation for Azure Stack.

In addition to the Marketplace items, tenants can download (or create) JSON templates to deploy applications. Examples of these templates can be found on [GitHub](https://aka.ms/ASWP_QSTemplates). This allows customers to create the custom deployments that they require to support their business.

Figure 2 shows the tenant view of the Azure Stack Marketplace and some of the items that are available in the default TP3 release (Note: Ubuntu is not included by default). Service administrators can define what is shown in the Marketplace. Custom virtual machine images, applications, and resource providers can be added by the service administrator using GitHub or custom templates, and then published to the tenant for consumption.

Microsoft Azure Stack 
Resource groups 
All resources 
Recent 
Virtual machines 
Browse > 
v New 
New 
Virtual Machines 
x 
p Search the marketplace 
MARKETPLACE 
Virtual Machines 
Data + Storage 
Networking 
Custom 
RECENT 
Virtual Machines 
FEATURED APPS 
See all 
14.04.3-... 
ee al 
WindowsServer-2012-R2- 
Datacenter 
Create a virtual machine from a 
platform image. 
Canonical-UbuntuServer- 
14.04.3-LTS 
Create a virtual machine from a 
platform image. 
Availability Set 
An availability set is a group of 
virtual machines that are deployed 
across fault domains and update 
Storage account 
Microsoft 
Canonical-UbuntuServer- 
Canonical 
Template deployment 
Microsoft 
WindowsServer-2012-R2-Datace... 
MicrosoftWndowsServer 
Virtual network 
Microsoft 

Figure 4: Marketplace tenant experience

A service administrator needs to keep in mind that the items they add to the marketplace will be visible and available to all users of the region that have a valid subscription and offer.

Below are links to the step by step documents for adding content to the Marketplace:

[Azure Stack Marketplace](https://aka.ms/ASWP_MP): overview of Marketplace

[Creating and Publishing Marketplace Items](https://aka.ms/ASWP_MPP): overview of the process, link to the Marketplace Generator tool, and general guidance

[Adding Custom Images to Azure Stack:](https://aka.ms/ASWP_VMImage) overview of the process

[GitHub templates for Azure Stack](https://aka.ms/ASWP_QSTemplates): these are specifically configured to work with Azure Stack. While you can use Azure templates they may require some modifications to work correctly with Azure Stack at this time.

**Good to Know Stuff**

In order to remove items from the Marketplace you will use the portal

and then use Remove-AzureRmGalleryItem to delete the azpkg.

Marketplace syndication enables download of an image and a gallery package into Azure Stack from Azure. If you need to remove an image downloaded in this manner navigate to the “Marketplace Management” UI and click delete on an item and it will take care of cleaning up both the image and the azpkg.

**Lab Design Decisions:**

So where do we start when we are thinking about the design of the lab…. First, we should understand the application or the services that will be deployed in our cloud. The cloud will be used by four departments;

* HR
* Finance
* IT
* Development

IT and development will use the cloud to build and test the cloud applications that the business requires, with IT also managing the Azure Stack Cloud itself. HR and Finance both have specific application requirements. By understanding the applications that each group will require, we can make logical design decisions as we move forward.

As we move thru the process of designing the lab we will need to consider the plans and offers that will be required by each of these groups, as well as determine how regions will be utilized when this functionality becomes available.

An outline of the applications each group will require is listed below.

|  |  |  |
| --- | --- | --- |
| **Domain** | **Business Units** | **Applications** |
| local.azurestack.external | Human Resources | Multiple websites using MySQL backend |
|  | Finance | Off the shelf Finance application using SQL Database |
|  | Information Technology | No specific applications |
|  | Development | Dev is working on adding Key Vault capabilities to new applications they are building  Responsible for maintenance and upgrades to the HR Websites and the Finance application |

In order to provide redundancy for administration a second service admin will also be added as a co-administrator

|  |  |  |
| --- | --- | --- |
| **Region** | **Service admin** | **Co-administrators** |
| Local | serviceadmin@"mylab".onmicrosoft.com | ITAdmin1 |

# Interacting with Azure Stack

 One of the most important design goals for Azure Stack was consistency with public Azure that would give rise to an unparalleled hybrid experience for organizations embracing and using cloud technologies whether one of the Azure public clouds or a private cloud built on Azure Stack. Consistency between Azure and Azure Stack allows you to interact with Azure Stack and Azure using the same skills and tools that you may already have from working with the public cloud. Whilst much of the focus of this series will be on using the portal to perform the operations needed to build out the lab environment and then define and utilize resources to build your applications and services, it isn’t the only option. The consistency between Azure and Azure Stack is built at the API level so other existing tools like the Azure PowerShell and Azure Cross Platform Command Line tools are also available to use against Azure Stack.

The portal is an excellent way to start using both Azure and Azure Stack and remains a good option for experimenting and performing one off tasks, the command line tools come in to their own as you move towards a more "devops" approach to managing your applications and services.

DevOps definition (Wikipedia):

DevOps is a term used to refer to a set of practices that emphasize the collaboration and communication of both software developers and information technology professionals while automating the process of software delivery and infrastructure changes. It aims at establishing a culture and environment, where building, testing, and releasing software can happen rapidly, frequently, and more reliably.

[https://en.wikipedia.org/wiki/DevOps](https://aka.ms/ASWP_DevOps)

**PowerShell**

Azure PowerShell 1.0 Preview was first released back in October of 2105, and has and will continue to evolve as a powerful command-line configuration environment for administrators. New and updated capabilities are being added and more management tasks are being exposed to PowerShell allowing administrators to accomplish nearly everything that is possible using the Graphical User Interface (GUI). In this [blog](https://aka.ms/ASWP_PSBenefits), Jeffery Snover (the father of PowerShell!) discusses some of the many benefits for using PowerShell to manage your infrastructure.

Why Does Azure Stack Require a Specific Version of PowerShell

Simple answer: Azure Stack requires a specific version of Azure Resource Manager (AzureRM) PowerShell in the TP releases. This is due to the rapid releases of PowerShell for AzureRM, and the development cycle for Azure Stack. As the product is developed a version must be aligned to, to ensure that all the PowerShell cmdlets work as expected. Updated schema versions, new cmdlets, and updates to existing cmdlets all can impact the proper operations of PowerShell in Azure Stack. As the development of Azure Stack continues, newer versions of AzureRM PowerShell will be approved for use with Azure Stack. At this time the required version is 1.2.8

Setting up an Environment & Authenticating

The first step to setting up PowerShell in your Azure Stack environment is to install the AzureRM module. Up to date instructions on how to complete this are located [here](https://aka.ms/ASWP_PS). You may choose to scope your module installation to a specific user using the *-Scope* parameter. This allows you to restrict the set of users that are allowed to use the commands available in the module. Ensure you deploy the correct version of the AzureRM module or you will run into issues when trying to run operations within Azure Stack. Whenever I have any issues with PowerShell configurations this is the first thing I check. You can easily verify the version of AzureRM using PowerShell. Simply type the command below and look for the version number for the AzureRM module(s).

PowerShell (and associated tools) can be installed on and run from the MAS-CON01 virtual machine. You may also chose to install these on a dedicated device that has access to your Azure Stack lab, or on the host server. I do the later as I have limited resources on my server and shut down MAS-CON01. If you chose to do this, keep in mind that any required reboots for installations/uninstalls will require rebooting the host server.

Next let's install the Azure Stack Utilities. The first step involves installing the required PowerShell module using the following command. This is a tool that has been created to simplify the login process for Azure Stack. It is available on GitHub, but can be installed easily with the following command

*Install-Module AzureStackUtils*

Next let's verify the Azure Stack modules are installed.

*Get-Module –ListAvailable AzureRM\**

PS Get-Module 
—Li stAvai I able 
AzureRM2 
Directory: Fi I\ModuIes 
0.4 
1.2 
AzureRN . Compute 
0.4 
0.4 
0.4 
0.4 
1.052 
0.4 
AzureRN . Network 
0.4 
0.4 
0.4 
0.4 
0.4 
0.4 
0.4 
AzureRM . Storage 
0.4 
0.4 
0.4 
0.4 
AzureRN . Websites 
0.4 
oduIeType Version 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
Scr i pt 
1.2 7 
1.042 
1.042 
o. 10.3 
o. 10.2 
1. 
1. 
1. 
1. 
1. 
1. 
1.1. 3. 
1. 
1. 
1. 
1. 
1.05 
1.1. 2. 
1. 
1.1. 3. 
1. 
1. 
1. 
1. 
1. 
1. 
1. 
Name 
AzureRN 
. ApiManagenent 
AzureRN 
AzureRN . Automati on 
Azur eRN . Azur eStackAdmi n 
. Azur eStackStor age 
AzureRN 
AzureRN . Satch 
AzureRN . DataFactori es 
. DataLakeAnaIytics 
AzureRN 
AzureRN . DataLakeStore 
AzureRN. Ons 
. HDInsight 
AzureRN 
. Insights 
AzureRN 
. KeyVauIt 
AzureRN 
AzureRN . Notifi cat ionHubs 
. Operational Insights 
AzureRN 
. profi I e 
AzureRN 
. RecoveryServi ces 
AzureRN 
AzureRN . Redi sCache 
AzureRN. Resources 
. SiteRecovery 
AzureRN 
AzureRN. Sql 
. StreamAnaIyti 
AzureRN 
AzureRN . Tags 
. Traffi d•lanager 
AzureRN 
AzureRM . UsageAggregates 
Export edCownands 
{Add-AzureRmApiManagenentRegion, Get-AzureRmApiManagenentS... 
{Get-AzureRmAutomati onJobOutputRecord, Import-AzureRmAutom... 
{Add-AzureRNLlsageConnection, Get-AzureRMLlsageConnection, 
{Remove-ACSAcquisition, Get-ACSAcquisition, Add-ACSFarm, C. 
{Remove-AzureRm8atchAccount, Get-AzureRm8atchAccount, Get- 
{Remove-AzureRmAvai labilitySet, Get-AzureRmAvai labilitySet. 
{Remove-AzureRmDataFactory, Get-AzureRmDataFactoryRun, Get. 
{Remove-AzureRmDataLakeAnaIyticsCataIogSecret, Set-AzureRm.. 
{Add-AzureRmDataLakeStoreIte«ontent, Export-AzureRmDataLa... 
{Get-AzureRmDnsRecordSet, Remove-AzureRmDnsRecordSet, Set- 
{Get-AzureRmHDInsightJob, New-AzureRmHDInsightSqoopJobDefi ... 
{Add-AlertRuIe, Get-AlertHistory, Get-AlertRuIe, Renove-AI 
{Get-AzureRmKeyVauIt, New-AzureRmKeyVauIt, Remove-AzureRmK... 
{Add-AzureRmAppIicationGateway8ackendAddressPooI, Get-Azur 
{Get-AzureRmNotifi cationHubsNamespaceAuthorizationRuIes, 
{Get-AzureRnOperationaIInsi ghtsSavedSearch, Get-AzureRnOpe... 
{Enable-AzureRmDataCoI lection, Disable-AzureRmDataCoI 
{Get-AzureRmRecoveryServi cesVauIt, Get-AzureRmRecoveryServ... 
{Remove-AzureRmRedi sCacheDi agnostics, Set-AzureRmRedi sCach... 
{Get-AzureRmADAppIication, Get-AzureRmADGroupNember, Get-A.. 
{Stop-AzureRmSiteRecoveryJob, Get-AzureRmSiteRecoveryNetwo... 
{Get-AzureRmSqIDatabaseThreatDetecti onPoIicy, Set-AzureRmS.. 
{Get-AzureRmStorageAccount, Get-AzureRmStorageAccountKey, . 
{Get-AzureRmStreamAnaIyticsFunction, Get-AzureRmStreamAnaI. 
{Remove-AzureRmTag, Get-AzureRmTag, New-AzureRmTag} 
{Disable-AzureRmTrafficNanagerEndpoint, Enable-AzureRmTraf... 
Get - us ageAggr egat es 
{Get-AzureRmAppServi cePIanMetrics, 

Figure 5. AzureRM Modules

For a list of all the commands available to you in this release of AzureRM type:

*Get-Command -Module AzureRM.<name of specific module>*

i.e. *Get-Command -Module AzureRM.AzureStackAdmin*

PS C get-cormand 
-Module 
Azur eRm. Azur estackAdmi n 
ormandType 
lias 
lias 
Cmdl et 
mdl et 
Cmdl et 
mdl et 
Cmdl et 
mdl et 
Cmdl et 
mdl et 
Cmdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
Cmdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
mdl et 
Name 
Logi n -Azur eRmAccount 
Sel ect -Azur eRmSubscri ption 
Add-Azur eRNGaI I eryltem 
Add-Azur eRNResour ceprovi der Regi st r ati on 
Add-AzureRNUsageConnection 
Get -Azur eRNGaI I eryltem 
Get -Azur eRNManagedLocat ion 
Get -Azur eRNMan agedSu bs cri pt ion 
Get -Azur eRNOffer 
Get-AzureRNPI an 
Get -Azur eRNResour der Regi st r ation 
Get -AzureRNTenantSubscription 
Get-AzureRNUsageConnect ion 
Get-AzureStackToken 
New-Azur eRNManag edLocat ion 
New-Azur eRNMan agedSu bs cr i pt ion 
New-Azur eRNOffer 
New-AzureRNPI an 
New-AzureRNTenantSubscription 
Remove-Azur eRNGaI I eryltem 
Remove-Azur eRNManagedLocat ion 
Remove-Azu r eRNMan agedSu bs cr i pt i on 
Remove-Azur eRNOffer 
an 
Remove-Azur eRNResour ceprovi der Regi st r on 
cri pt ion 
Remove-AzureRNUsageConnect ion 
Set -Azur eRNManagedLocat ion 
Set -Azur eRNMan agedSu bs cri pt ion 
Set -Azur eRNOffer 
Set -AzureRNPI an 
Set -Azur eRNResour ceprovi der Regi st r ation 
Set -AzureRNTenantSubscription 
Version 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
o. 10.3 
Source 
Azur eRm. Azur estackAdmi n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur n 
Azur eRm. Azur estackAdmi n 
Azur eRm. Azur estackAdmi n 

*Figure 7. AzureRM AzureStackAdmin Commands*

Now that we have PowerShell installed we need to connect to our Azure Stack environment. Using the Azure Stack Utilities, connecting to Azure Stack with PowerShell is now as simple as typing the command below. A popup will appear where you input your Azure Active Directory account information, and voila -> you are now connected!

*Connect-AzureStack*

ps C: Win 
s system32> connect-azurestac 
cmdlet Connect-AzureStack at conunand pipeline position I 
Supply values for the following parameters: 
redential 
Windows PowerSheII credential request 
Enter your credenbals. 
Password: 

Figure 8. Connect Azure Stack Credentials Popup

To test this run the command:

*Get-AzureRMSubscription*

You should see something similar to the image below indicating PowerShell is now working against your subscription in Azure Stack.

s sys 
SubscriptionName 
Subscriptionld 
enantld 
Stat e 
PS 
em > g 
-azurermsu scri 
1 on 
. Default Provider Subscription 
. Enabled 

Figure 9. Get-AzureRMSubscription Results

PowerShell Example

Now that we have the correct version of AzureRM installed and are connected to our Azure Stack environment, let's look at an example of what we can do with it.

First I would like to look at a list of all the resources deployed in my environment. To do this I type the following command:

*Get-AzureRMResource*

Next I want to deploy a template that I have obtained from GitHub. In order to do this, I will download the [Azure Stack Quick Start Templates](https://aka.ms/ASWP_QSTemplates) and extract the example templates to a local folder and work with the one I require from there. The template I will use is the 101-simple-linux-vm template.

1. Download the [Ubuntu 14.04](https://aka.ms/ASWP_Canonical) image to your Azure Stack host
2. Add the image to your Marketplace using the instructions Add a VM image to Marketplace with PowerShell located [here](https://aka.ms/ASWP_AddImage).
3. Download and extract Quick Start Templates
4. In PowerShell (admin mode) navigate to the folder containing the template (azuredeploy.json) and the template parameters (azuredeploy.parameters) files
5. Connect to your Azure Stack environment using the *Connect-AzureStack cmdlet we used above*
6. Run the following command to deploy the template:

NOTE If you are completing multiple deployments of this virtual machine, update the following variable:

* $myNum = "001"

# Set Deployment Variables  
$myNum = "001" #Modify this per deployment  
$RGName = "myRG$myNum"  
$myLocation = "local"

$templateFile= "azuredeploy.json"  
$templateParameterFile= "azuredeploy.parameters.json"

# Create Resource Group for Template Deployment  
New-AzureRmResourceGroup -Name $RGName -Location $myLocation

# Deploy Template   
New-AzureRmResourceGroupDeployment `  
 -ResourceGroupName $RGName `  
 -TemplateFile $templateFile `  
 -TemplateParameterFile $templateParameterFile

**Visual Studio**

Developers in many organizations do not wish to use the portal to manage their deployments. They also require tools to be able to create and configure JSON templates. In many cases this tool will be Visual Studio. IT staff may also need to understand, configure, and will need to deploy ARM templates as part of their role. By using the same tools as their development teams, they can easily create, manage, and deploy the templates to Azure Stack …or Azure.

There are many versions of Visual Studio available, but I am going to focus on two that will allow you to do the tasks you need to do to manage and edit ARM templates, and they are **FREE**.

Let's start with [Visual Studio Community edition.](https://aka.ms/ASWP_VSComm) This is the version I use on a regular basis. This edition will allow you to do the tasks required in Azure Stack. It enables you to build cloud applications accurately and efficiently without losing the ARM template context. You can easily zoom into details such as parameters, related functions, check syntax, and deploy your cloud applications to Azure Stack or Azure.

Some of the main components that will be of interest when developing or managing your ARM Templates include:

**Azure Resource Management tools:**

* Create an application using the Azure Stack Quickstart templates available in GitHub.
* Use Azure Stack KeyVault to store secrets accessed by deployment templates.
* Create and edit Azure Resource Manager deployment templates and parameter files with different settings for multiple environments.
* Add resource snippets to your template.
* Create resource groups and deploy templates using a wizard or automated scripts.
* Visualize the resources and parameters using JSON outline. Leverage rich IntelliSense and validation in the JSON editor.

**Cloud Explorer:**

* Create and configure Microsoft Azure Stack / Azure environments.
* Simultaneously log into multiple Azure Stack accounts and access the subscriptions and resources.
* Use Cloud Explorer to view, manage, and debug and diagnose your Azure resources.
* Connect to and target any Microsoft cloud: public, private, hosted, and government.
* Start and stop App Services and VMs with Cloud Explorer.

Once I have Visual Studio, installed I need to do a few things. First and foremost, I need to check the AzureRM PowerShell version in my Azure Stack lab. When I deployed Visual Studio, it updates my version to the current one Visual Studio uses, and this may not match what I require for Azure Stack. The Azure Stack version we are using is 1.2.8

To check this do the following:

1. Open PowerShell in elevated mode (administrator mode)
2. Run the following command

(Get-Module -ListAvailable | Where-Object{ $\_.Name -like 'AzureRM' }) `

| Select Version, Name, Author, PowerShellVersion | Format-List;

Directory: Fi I\ModuIes 
oduIeType Version 
Name 
AzureRN 
AzureRN 
Export edCownands 
Scr i pt 
Scr i pt 
1. 2.8 
1. 2.6 

Figure 10: Versions of AzureRM PowerShell in my lab

If your version is not 1.2.8, uninstall the version of Azure PowerShell currently installed using the Control Panel => Programs. Once uninstalled reboot the machine, and then install the correct version using the commands below

Get-PSRepository

Install-Module -Name AzureRM -RequiredVersion 1.2.8

Install-Module -Name AzureStack

The next step is to start Visual Studio and create a new project. Visual Studio Developer edition has the option to select Cloud, which gives us access to the ARM template tools.

1. Start Visual Studio Developer Edition
2. Login with your account when prompted
3. Select New Project => Cloud =>Azure Resource Group

Start Page -a X 
Visual Studio 
Start 
New Project... 
Open Project... 
Open from Source Control... 
Recent 
Ex 
New P roject 
Recent 
Installed 
Templates 
Visual 
D Windows 
Web 
Android 
CIO ud 
Extensibility 
Reporting 
Silverlight 
WCF 
Workflow 
D Visual Basic 
Visual 
D Visual C++ 
SQL Server 
Python 
D Azure Data Lake 
D JavaScript 
D Online 
Location: 
Solution name: 
anquaqe 
.NET Framework 45.2 
• Sort by. 
Default 
Azure Cloud Service 
ASP.NET Web Application 
Azure WebJob 
Azure Mobile App 
Azure Mobile Service 
Azure Resource Group 
Click here to go online and find templates. 
Visual C# 
Visual C# 
Visual C# 
Visual C# 
Visual C# 
Visual C# 
M TestRG 
studio 
MyTestRG 
ra eqv 
Search Installed Templates (Ctrl+E) 
Type: Visual 
This template creates an Azure Resource 
Group deployment project. The 
deployment project will contain artifacts 
needed to provision Azure resources using 
Azure Resource Manager that will create 
an environment for your application. 
Browse... 
[Q] Create directory for solution 
[3 Add to Source Control 

Figure 11: Open a new project in Visual Studio

1. Select "Show Templates from This Location"

Select Azure Template 
Show templates from this location: 
Visual Studio Templates 
Visual Studio Templates 
Azure QuickStart - Featured (github.com/Azure/azure-quickstart-tempIates) 
Azure QuickStart (github.com/Azure/azure-quickstart-templates) 
Azure Stack QuickStart (github.com/AzureStack-QuickStart-TempIates) 
Windows Virtual Machine Scale Set 
MICROSOFT 
Blank Template 
By: Microsoft 
The blank template will add a deploym 
template file and other files needed for 
Resource Manager. 
VERSION: 1.0 

Figure 12: Visual Studio link to Azure Stack QuickStart templates in GitHub

d. Select the template you would like to use

Select Azure Template 
Show templates from this location: 
Azure Stack QuickStart (qithub.com/AzureStad 
Search 
co 
(0 
co 
101 Iinux-customscript-ext 
AZURESTACK 
101 Iinux-docker-extension 
AZURESTACK 
I O I - Iinux- v maccess-eR 
A.zuRESTACK 
101 -simple- Iinux-vm 
AZURESTACK 
101 -simple-windows-vm 
AZURESTACK 
101 -simple-windows-vm-diag 
AZIJRESTACK 
01- two- vm-extensions 
101 -simple-windows-vm 
By. azurestack 
This template takes a minimum amount of parameters and 
deploys a Windows VM, with the appropriate DNS setting for 
the Azure Stack POC environment. 
VERSION: 2016-09-20 
Deploys a simple Windows VM. This template also deploys a Virtual Network (with 
DNS), Public IP address, Network Security Group, and a Network Interface. 

Figure 13: 101-simple-windows-vm template from Azure Stack QuickStart templates on GitHub

Edit View Project Build Debug Team 
Debug 
JSON Outline 
AzureResourceGroup2 - Microsoft Visual Studio 
"description": "Unique DNS Name 
Tools Test 
Any CPU 
Analyze Window Help 
Start 
azuredeployjson 
Quick Launch (Ctrl+Q 
Solution Explorer 
Search Solution Explorer (Ctrl+;) 
Solution 'AzureResourceGroup2' (I project) 
Azure ResourceGroup2 
References 
azuredeploy.json 
azuredeploy.parameters.json 
Deploy AzureResourceGroup.psI 
Sign in 
parameters (6) 
X variables (15) 
resources (5) 
How do deploy project atifacts 
with an Azure deployment 
temolate? 
Schema: https://schema.management.azure.com/schemas/2015-OI 01/depIoymentTempIate.json* 
"$schema": "https://schema.management.azure.com/schemas/2ß15-ßI-ßI/depIoymentTempIate.json#", 
' contentVersion " : 
"l.ø.ø.ø", 
parameters" : 
' countName" : 
"type" • 
" string' 
"metadata": { 
Machine's disks will 
be 
placed . " 
12 
14 
19 
21 
26 
28 
33 
35 
- 
- 
- 
- 
- 
- 
- 
- 
for 
"defaultVaIue": " 'vm' , 
"vmName": { 
"type" • 
"metadata": { 
"description": "Name of the WI. " 
the Storage Account where the Virtual 
"defaultVaIue": " simplewinvm' , resource6roup() 12)) " 
" adminl_lsername": { 
"type" • 
" string" 
"metadata": { 
"description": "Username for the Virtual Machine. " 
"defaultVaIue": "vmadminl" 
"adminpassword": { 
README.md 
Solution Explorer 
Team Explorer 
Properties 
azuredeploy.json File P roperties 
Advanced 
Build Action 
Class View 
"type" • 
'securestring' 
"metadata": { 
"description": 
"The password for the 
Administrator account of the 
new vr•ls. 
15, 36))) " 
Defa u It 
value 
subscription 
"defaultVaIue": " [concat( 'Subscription#' , . id, 
"dnsNameForPubIicIP" : 
"type" • 
" string" 
"metadata": { 
Copy to Output Directory 
File Name 
Full Path 
Full Path 
Location of the file. 
Do not copy 
azuredeploy.json 
"description": "Unique DNS Name for the Public IP 
used to access the Virtual 
Machine. " 
"defaultVaIue" : 
" ' dns ' , resource6roup() . name)) , 
'windowsOSVersion " : 
"type" • 
"string", 
"defaultVaIue" : 
"2ø12- 
R2- 
Datacenter" , 

Figure 14: Template imported into Visual Studio

We can make changes to the template if we chose or just go ahead and deploy it. To make changes see the documentation [here](https://aka.ms/ASWP_AuthTem). Keep in mind that schema versions available in Azure may not be available in Azure Stack at the current time.

To deploy we need to connect to our Azure Stack environment.

1. In Visual Studio Click on File => Account Settings. This will open the window below

Machine generated alternative text:
Sign in to \Tsua Studio 
Visual Studio will help you plan projects, collaborate with your 
team, and manage your code online from anymjhere. 
Learn more 
Sign in to start using your Azure credits, publish code to a private 
Git repository, sync your settings, and unlock the IDE. 
Visual Studio 
Communty 2015 
License: MSDN Subscription 
This product is licensed to: 
shadowdancerlg@hotmall.com 
Check for an updated license 
Sign in 
Don't have an account? Sign up 
A Accounts 
AzureStack 
Rem ove 
ServiceAdmin@AzureStackLab.onmicr... 
McBride 
Rem ove 
admin I@KMM caride.onmicrosoft.com 
Microsoft account 
shadowdancer I go h otmail.com 
Add an account... 
Remove 
Apply filter... 

Figure 15: Visual Studio sign in page

1. Click on Add an Account and add your Azure Stack account you would like to use for this deployment. Note that the account you chose will need to have a subscription with the ability to deploy IaaS components (Storage, Network, Compute)
2. Next let’s go back to the template we have been working with and deploy it. In the Solution Explorer section select the project name and right click it.

Machine generated alternative text:
Deploy 
guild 
Rebuild 
Clean 
Scope to This 
New Solution Explorer View 
Add 
Manage NuGet Packages... 
Source Control 
Cut 
Remove 
Rename 
Search Solution Explorer (Ctrl+;) 
Solution 'AzureResourceGroup2' (I project) 
esourceGroup2 
ences 
deploy.json 
deploy.parametersjson 
oy AzureResourceGroup.psI 
ME.md 
Ctrl+X 

Figure 16: Deploy template with Visual Studio

1. Click Deploy => New, and verify you are connected to the correct environment. I also need to configure the Resource Group this will deploy to. I can create a new Resource Group or select an existing one if available

If I click on parameters I can configure any available parameters. This is also where I will configure my KeyVault parameters if I am using KeyVault.

Lastly, I have the option to only validate the template (bottom) if I like.

Deploy to Resource Group 
AzureStack 
ServiceAdmin@AzureStackLab.onmicrosoft.com 
Subscription: 
Default Provider Subscription (ServiceAdmin@AzureStackLab.onmicrosoft.com) 
Resource group: 
<Create New...> 
azuredeploy.json 
Template parameters file: 
azuredeploy.parametersjson 
Artifact storage account: (D 
Validate only (don't deploy) 
Edit Parameters... 
How do deploy project atifacts with an Azure deployment template ? 
Cancel 

Figure 17: Deployment Properties

1. Once I am satisfied I am ready to deploy, simply click OK

71 
76 
100 8', 
Output 
Schema: https://schema.management.azure.com/schemas/2015-01 01/depIoymentTempIate.json* 
11:23: 
11:23: 
11:23: 
11:23: 
11:23: 
2 3 : 48 
- 
- 
- 
resources " 
"apiVersion" . 
• "2ø15-ø5-ø1-preview", 
"type" • 
"Microsoft . Network/ networkSecurity6roups " , 
name' . 
' [variables ( ' networkSecurity6roupName ' " , 
"location": " [variables( 'location ' " , 
"tags " 
"displayName" • "NetworkSecurity6roup" 
Show output from: AzureResourceGroup2 
39 - 
39 - 
39 - 
39 - 
39 - 
11 • 
.23. 
11: 
Project "AzureResourceGnup2 .deployproj" (StageArtifacts target(s)): 
Project "AzureResourceGnup2 .deployproff' (ContentFiIesProjectOutputGroup target(s)): 
Done building project "AzureResourceGnup2 .deployproff' 
Done building project "AzureResourceGnup2 .deployproff' 
Build succeeded. 
Launching PowerSheII script with the following command: 
studio 

Figure 18: Deployment

Once we get comfortable with Visual Studio, we can develop new JSON templates, edit templates, manage templates, etc.

I mentioned a second tool that may be of interest to IT professionals who do not want to use Community Edition, but still want the power of Visual Studio to manage ARM Templates; Visual Studio Code. Although you cannot deploy templates from Visual Studio Code it is a great tool for managing ARM templates.

DO www.ts node- express-ts Visual Studio Code 
Edit View Goto Help 
EXTENSONS 
@popular 
122 
C# Visual Studio Code 
o. 0211K 
Lintm Dåugging (mutti-t„ 
fw Owome 
Dåug your JavaScript code. „ 
Wmft JS - 
oz..u3K 
Go 063g 
Rch Go su»ort t„ 
ESLü't 
Integrates ESLint into VS 
app.ts 
import app from ' ./app' ; 
import debu"dule = require( 'debug'); 
import http = require('http'); 
const debug = 'node-express-typescript: sewer'); 
// Get port fron environnent and store in Express. 
const port = nor-RIizePort(process.env.PNT Il • 3øøø•); 
app. set(• , 
// create 
const 
server. Ii 
server.on 
server .on e 
s a gePM•t 
Normal const I 
function normalizeport(val: any) : 
let port = parselnt(val, 10); 
string 
nur%erl string I boolean { 

Figure 19: Visual Studio Code

[Code Edition](https://aka.ms/ASWP_Code) is extensible and customizable, has built in Git Commands, and uses Intellisense and can be run on Windows, MAC, and Xplat operating systems. It is also much lighter "weight" than Community Editor and will allow IT administrators to manage and edit JSON templates.

1. First you will need to install the modules. Click on Extensions and search for Azure Resource Manager Tools and install it.

Machine generated alternative text:
Selecta:' 
EXTENSIONS 
search Extensions in Marketplace 
Azure CLI 
020 
Azure CLIs for VS Code 
Christof Marti 
Azure Funcüons 
00.1 
vscode-icons rww ships with official API Go to Re > Preferences > Re Icon Theme and select VSCode 
January 2017 (version 1.9) 
Learn more about file & 
LeMn about this extension 
Close 
+611 
An easier way to deal with Azure Functi... 
AhnEd EISayed 
Azure Resource Manag« Tools o 
Template language support for Azure Z.. 
Azure Storage Ut"iw 0.02 +729 * s 
Generates Shared Access Signature for 
Compulim 
Azure Tools for Visual Studio Code 
Convenient features for Microsoft Azur... 
Git History log) 
0.1S +367K as 
View git log, file or line History 
Don Jayamanne 
og_o 
+393K 
Develop PowerShell scripts in Visual Stu... 
Proj«t 
+ 1a3K 
0_132 
Easily switch between pr*cts 
Alessandro Fragnani 
Spe"ing and Grunmar Check« os 1 
Uses a web service to detect mistakes a... 
Sean McBreen 
vscode-icoru 
7.12 
Icons for Visual Studio Code 
Rolkrto 
Welcome to the first 2017 release of Visual Studio Code. There are a number of significant updates in this version that we hope you will like, some of the key highlights 
New Welcome page - Whether you are a new or returning user, the Welcome page helps get you started quickly. 
• Interactive Playground - Try out VS Code's advanced editing features without creating files or a project. 
Synchronized Markdown Preview - Coupled view of a Markdown preview and its editor. 
Format on Paste - Format source code as soon as you bring it into your project. 
Language specific settings - Customize your settings for specific languages. 
TypeScript references CodeLens - VS Code ships with Typescript 2.1.5 and now includes References CodeLens. 
Single file debugging - Debug without a configuration file to enable quick debugging of single files. 
• Inline variables display -See variable values inline while debugging. 
Expanded Node.js debugging configurations - Easily debug Mocha tests, gulp tasks and even Yeoman generators. 
• Improved task running support - Run multiple commands from the same task. 
Faster Integrated Terminal - We improved the integrated terminal's performance and Windows support. 
The release notes are arranged in the following sections related to VS Code focus areas. Here are some further updates: 
Workbench - New settings to control new windows, improved Tabbed headings, Zen Mode customization. 
Editor - Add keyboard shortcutkeys for snippets, Go to Implementation, fast search navigation. 
Languages - Emmet abbreviations from extemal files, HTML format settings, Markdown editing improvements. 
Extensions - Create Extension Packs through the VS Code Yeoman generator. 
Debugging - User level launch.json, copy callstack action. 
Node.js Debugging - Just My Code improvements, restart support for launch configurations. 
Extension Authoring - New insertSnippet API, openTextDocument can now take language. 
Workbench 
Welcome experience 
We have added a Welcome page (Help > Welcome) with QIick links to help new users as well as a list of Recent folders and up documentation links for fast navi 

Figure 20: Visual Studio Extensions

1. Once installed click file => Open File and navigate to the JSON template you would like to use

File Edit 
Selection View 
EXPLORER 
OPEN EDITORS 
Welcome 
AzureResourceGroup5.sln 
X DeploymentTemplate.json AzureResourceGroup5\».. 
Deployment Template.param.dev.json AzureResou.. 
DeploymentTempIatejson 
WindowsVlrtualMachine.json 
AZURERESOURCEGROUP5 
AzureResourceGroup5 
AzureResourceGroup5 
obj 
Scripts 
Templates 
DeploymentTemplate.json 
DeploymentTemplate.param.dev.json 
Tools 
AzureResourceGroup5.deployproj 
Deployment.targets 
AzureResourceGroup5.sln 
DeploymentTemplate.json AzureResourceGroup5\... 
DeploymentTemplate.paramdev.json 
DeploymentTemplate.json ...uzureResourceGroup4\... 
WindowsVirtualMachinej 
"$schema": "https://schema.management.azure.com/schemas/2015-ø1-ø1/dep10ymentTemp1ate.json#" , 
•contentVersion : 
" "I.o.e.e", 
•parameters": { 
" adminUsername": { 
"type": "string" , 
"minLength": 1, 
"metadata": { 
"description": "Username for the Virtual Machine. " 
" adminPassword": { 
"type": "securestring" , 
"metadata": { 
"description": "Password for the Virtual Machine. " 
"dnsNameForPub1icIP": { 
"type": "string" , 
"minLength": 1, 
"metadata": { 
"description" : 
"Globally unique DNS Name for the Public IP used to access the Virtual Machine." 
"windowsOSVersion": { 
"type": "string" , 
"defaultVa1ue" : 
" 2e12-R2 -Datacenter" , 
"allowedVa1ues" : 
"20e8-R2-SP1", 
"2812-Datacenter" , 
" 2812-R2 -Datacenter" 
"metadata": { 

Figure 21: JSON Template

1. Once you have reviewed and edited the template to your satisfaction, you can save it and deploy using PowerShell, CLI, or the portal.

**Azure CLI**

Command-Line Interface (Azure CLI) is another tool you can use to deploy and manage resources with Azure Stack Resource Manager. This tool is of special importance to those using Linux and Mac client platforms for their management devices. CLI can be downloaded and installed using the instructions [here](https://aka.ms/ASWP_CLI).

Instructions to connect to Azure Stack using CLI are also contained in the previous link. To verify connectivity, you can run this command and you should see the results in the image below. Running this command will also give you a view into the commands that are available to you with CLI.

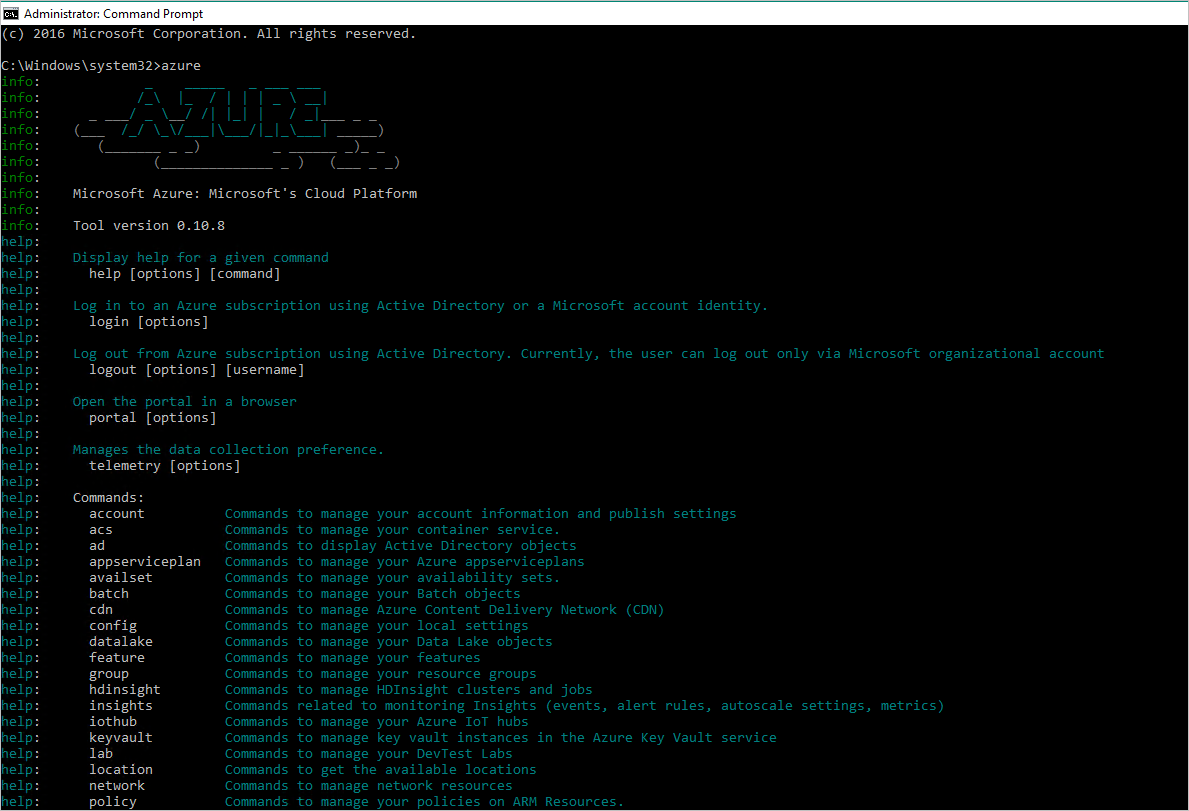


Figure 22. Results of Azure command in CLI

Now let's deploy the same Linux image, using the same template, changing the parameters in the below example to fit your deployment requirements. First ensure you have CLI configured, and you are logged in and then navigate to the folder containing the JSON files for the deployment. Finally update and run the command below, changing the <RG Name> and <Deployment Name> fields.

Log into Azure Stack using the instructions [here](https://aka.ms/ASWP_CLI).

Example: azure group create "<RG Name>" "local" -f azuredeploy.json -d <Deployment Name> -e azuredeploy.parameters.json

**Azure API**

Connecting to Azure Stack via API's is out of the scope of this blog post. However, if you would like to explore this capability there is great information [here](https://aka.ms/ASWP_API). More in depth information about Azure Stack API management will come over time, so please keep checking back to Azure Stack docs and GitHub.

# Subscriptions

~~S~~ubscriptions are the first thing a tenant sees when beginning to use Azure.  Due to this, and the fact that all other design decisions will be based upon the subscriptions tenants ultimately consume, let’s look at those now.  As you work through this series, it is good to always keep in mind the subscription, as that is what your tenants ultimately see.

Azure Stack subscriptions are used to grant users access to the Azure Stack services that you are offering, as well as granting access to the Azure Stack portal itself. Like Azure, Azure Stack subscriptions provide a logical boundary of scale, administration and billing for the tenants which consume services from Azure Stack.

* **Scale** - Subscriptions are a logical limit of scale by which resources can be allocated. These limits include hard and soft caps of various resource types that are offered by an administrator.
* **Administration** - Tenant administration of Azure Stack resources is defined at the subscription level. In addition, Azure Resource Manager provides granular Role-Based Access Control (RBAC) model for assigning administrative privileges at the resource level.
* **Billing** - A subscription additionally forms the billing unit as consumption of Azure Stack services can be measured at this level. Additionally, Azure Resource Manager provides the ability to assign Resource Tags to provide additional information on a per resource group or resource basis to support granular chargeback/show back scenarios.

Of the three, scalability is the key element for understanding how the subscription strategy will account for growth as consumption increases.

In order for your tenants to access resources within Azure Stack you will need to configure them in Azure Active Directory. With the TP3 release users need to be members of your Azure Active Directory domain and will use the account and password assigned to that user when logging into Azure Stack. Tenants are usually configured as "users" in Azure Active Directory while service administrators completing the infrastructure installation will need to be provisioned with "global admin" organizational rights.

Microsoft Azure 
AzureStack 
Search ICtr/+/) 
Cwervie-w 
Quick start 
Users and groups 
Enterprise applications 
App registrations 
Azure AD Connect 
Domain names 
Password reset 
Company branding 
user settings 
Properties 
Conditional access 
Named networks 
Sign-ins 
AzureStack 
Users and groups - All users 
Users and groups - All users 
- Actiw GrætMy - 
Overview 
All users 
All groups 
Password reset 
Company branding 
user settings 
Group settings 
evice Settings 
Sig n -ins 
Audit logs 
+ Add 
columns 
users 
Contosol 
Contos02 
ContosoAdmin 
Dev Admin 
Fabrikam Admin 
Fabrikaml 
Fabrikam2 (Dev) 
Finance Admin 
HR Admin 
IT Admin 
Kath McBride 
NWT2 (Dev) 
NWTAdmin 
ServiceAdmin 
Multi-Factor m.. 
p Search resources 
ContosoI@AzureStackLab.onmicrosoft.com 
Contos02@AzureStackLab.onmicrosoft.com . 
ContosoAdmin@AzureStackLab.onmicros... . 
DevAdmin@azurestacklab.onmicrosoft.com 
FabrikamAdmin@FabrikamAS.onmicrosoft... . 
FabrikamI@FabrikamAS.onmicrosoft.com 
Fabrikam2@FabrikamAS.onmicrosoft.com 
FinAdmin@azurestacklab.onmicrosoft.com 
HRAdmin@azurestackIab.onmicrosoft.com 
ITA dmin@azurestacklab.onmicrosoftcom 
kathm@microsoft.com 
NWT I@NWTAS.onmicrosoft.com 
NWT2@NWTAS.onmicrosoft.com 
N V'.'TAd m in @NWTA S.onmicrosoft.com 
ServiceAdmin@AzureStackLab.onmicrosof... . 

Figure 23: Configuring users in Azure Active Directory (note this is a preview feature)

For instructions on adding tenants to your Azure Stack environment refer to the link below:

[Add New User to Azure Active Directory](https://aka.ms/ASWP_AAD)

Now that you have your tenants configured let’s go back to subscriptions. There are two aspects to a subscription:

1. The subscription, which governs access to and use of the services via offers, plans that the user will subscribe to (see note below). The service administrator manages the services that will be offered to tenants.
2. The Azure Active Directory account as this is how usage and billing will be managed

NOTE: For details on offers, plans and quotas, see upcoming post

We can think about how to use subscriptions in two key ways:

1. Self Service: the tenant clicks on get a subscription and browses thru the offers their Service Administrator has made available, and creates their own subscription
2. The administrator (or delegated administrator) creates subscriptions with offers, plans, and quotas aligned to them, and then assigns them to tenants.

It is critical to develop the Subscription, Fabric, and Administrative models together to have a cohesive approach. Understanding how each component is limited and how each impacts the others is critical to a solution that can scale and be flexible enough to support the needs of the business.

Many of the early decisions in architecting and planning an Azure Stack environment and related subscriptions can have an impact on future decisions and designs as the cloud environment grows. As such, it is important to have participation and input from groups within an organization including networking, security, identity, domain administrators, and IT leadership.

Below are a few considerations when defining the subscriptions that you will need to consider:

* Do you require different configurations of offers and plans for various tenant groups or organizations
* Do you require segregation of users for security or compliance reasons
* Do you require separate billing and usage monitoring for tenant groups/organizations
* Do you require co-admins from business units or tenant organizations

While for some organizations a single subscription will work just fine, other organizations may require multiple subscriptions for more granular control. So, let’s look at these two use cases.

**Subscription Management**

As a service administrator, you can create or delete subscriptions via the portal or PowerShell. This applies to both their own subscriptions and their tenants’ subscriptions. A tenant who creates a subscription can also delete their own subscriptions. Deletion of a subscription will remove all resources that are part of that subscription and is non-recoverable other than via backups. Use care when deleting subscriptions.

One of the big decisions that will be required, is to determine if offers will be made private or public. When an offer is made public, all tenants can see it and chose to select the offer when they sign up for a subscription. The exception to this is if you chose to use delegation and custom URLs. By pointing tenants to a custom URL, the delegated admin can ensure that only their delegated offers are visible to their tenant base.

The choice to leave an offer private and assign tenants to it, requires thought as there is additional overhead to this choice. Tenants must be assigned to private offers, increasing the workload for the administrator/delegated administrator to manage adds/deletes. The service administrator may decide to use delegation, whereby he/she configures [delegation](#_Delegation) and the delegated admin will then be responsible for determining how they will make the delegated offer available to their tenants.

In order to allow others to assist with management of offers and plans, you can also add co-administrators to a subscription using role based access. Click on the subscription you would like to add a co-administrator to and then click on the Access icon as shown in the image below:

AdminSub 
Settings Add Plan 
Essentials 
AdminSub 
Gold Offer 
Delete 
Subscription ID 
6e6a785d-c565-40g6-bbbo-ebgd413032d3 
All settings 

Figure 24: Role Base Security

This will open a blade where you can then add users.

Users 
AdminSub 
Add 
Roles 
Subscription admins O 
ACCESS 
Inherited 

Figure 25: Users

For more information on adding users please see the [Manage User Permissions](https://aka.ms/ASWP_Perms) documentation.

When creating a subscription, you must align it to an offer. Thus, your offers and plans that you want to align to your various subscriptions must be in place first. But more on them later in this series, first let’s complete our review of subscriptions.

The use of multiple subscriptions can create complexity. If isolation is required, then you will need to consider subscription administration very carefully. Some considerations for multiple subscriptions include:

* A subscription on its own doesn’t cost anything
* A subscription has its own administrators
* Overcome soft limits per subscription (i.e. number of vNets, CPU cores, etc.).

Complexities are introduced when you consider that the on-premises networking and security infrastructures are typically shared resources.

Patching, monitoring, and auditing are frequently provided by dedicated staff who are trained in the related tools. As you design your subscriptions consider that you may need to duplicate services, including monitoring, patching, and anti-virus.

**USE CASES**

**Self Service Subscription Model**

Some organizations want to leverage self-service for user subscriptions. The same care around configuring offers and plans will be required, however all their users will go to a single portal and get a subscription of the size and type they require, and start consuming the services that are offered.

Of-fem a m-'•æota 
z rip t 
(put 
High Evel proc—s 
I. Service (or del*3ed guder 
admin cr eaesOffers/PIa-s and asgns 
quotæ 
2 Service Admin (or del*3ed guder 
admin makesofferspub'c 
3. T enatTs seet get a 
subscrCtOn 
4 Tan arts seet 
th me they want 
5 TenatTsdepö/ r eourcesthe,• requie 
within the &undaris impcæd b'/ the 
quotæ for that subgr Otion 

Figure 26: example self-service subscriptions

In my lab, I use the Default Provider Subscription to create my self-service offerings.

For my self-service users to access resources, I need to create plans and offers. Information on plans and offers is in upcoming posts, however if you want to build a few now the instructions are here:

[Create a Plan](https://aka.ms/ASWP_Plan)

[Create an Offer](https://aka.ms/ASWP_Offer)

**Assigned Subscription Model**

As mentioned above there are several reasons why an organization may choose to use assigned subscriptions including compliance, requirement for co-administrators that do not have visibility to other subscriptions, billing requirements, etc.

In this model tenants are specifically assigned to subscriptions via tenant assignment at the plan level. Offers are not made public, and the service administrator (or delegated provider admin) can chose to utilize custom URL's as well to ensure the boundaries imposed by these hidden subscriptions are not compromised. Basically, the service administrator (or delegated provider admin) will send users a link to the specific portal URL they want those users to access, and when the user logs in they will have access to the subscription and the offers/plans/quotas that are configured for that subscription.

While requiring slightly more management from IT this solution best fits Service Providers, or enterprises, where knowledge of other tenants is either a security issue, or just unwanted as in the case of an enterprise that selects this model.

Another scenario here is approvals (ITSM). While I am not convinced that ITSM is a good companion to cloud computing, it may make sense for some customers to have approvals at the subscription level. This can easily be achieved by automation, with their existing ITSM portal listing offers through the APIs (possibly even syncing them in their CMDB) and automation assigning the subscriptions after the required approval workflow.

Azure Stack 
Infrastructure 
HR Subscription 
Offer/PIa n/ 
Quota 
(private) 
HR Tenant 
HR Tenant 
HR Tenant 
HR Tenant 
IT Subscription 
Offer/PIa n/ 
Quota 
(private) 
IT Tenant 
Tenant 
Tenant 
IT Tenant 
Development 
Subscription 
Offer/PIa n/ 
Quota 
(private) 
Dev Tenant 
Dev Tenant 
Dev Tenant 
Dev Tenant 
High level process 
1. Service Admin (or delegated provider 
admin) creates Offers/Plans and assigns 
quotas 
2. Service Admin (or delegated provider 
admin) creates HR/lT/Dev subscriptions and 
aligns to appropriate offers/plans/quotas 
3. Service Admin (or delegated provider 
admin) assigns ten ants to the subscription 
they require access to. 
4. Tenants access portal and assigned 
subscriptions are available to them 
5. Tenants deploy resources they require 
within the boundaries imposed by the 
quotas for each subscription 

Figure 27: example assigned subscriptions

**Hybrid Subscription Model**

In some cases, you may opt to use a hybrid model that includes both assigned and self-service subscriptions based upon your business requirements. For instance, you may have some areas of your organization that require the ability to just come and get a subscription or like Azure, you may choose to have a trial offer that contains minimal quotas for resources, so tenants can get their feet wet with Azure Stack capabilities. For those users where you want them to consume a specific subscription you can either using subscription naming standards and publicly visible offers they come and get or you can create offers that are private and deliver assigned subscriptions to them.



Figure 28: Example hybrid subscriptions

**Naming Conventions**

When naming your subscriptions or any other objects within Azure Stack, you will want to consider naming conventions to ease management as well as for billing purposes.

* Company: Service providers will want to ensure that company name is included in the naming of subscriptions
* Department: Enterprises may wish to use department names as the identifier for the groups within the organization
* Product line: Some enterprises may want to include the name of the product/application in the subscription name
* Environment: Some organizations will have multiple environments for the lifecycle management of applications (Prod, QA, DEV) and may choose to include this in the naming
* Characters: some characters are not allowed or will cause issues in naming conventions i.e. &

Naming standards extends beyond just subscriptions, so let's take a quick look at that now. There are many objects within Microsoft Azure Stack that require proper naming so that they can be easily identified. (i.e. Storage Accounts, Virtual Machines, Availability Sets, Virtual Networks, etc.) When choosing your naming standard keep in mind that there are several constraints that you must work within:

* Storage names must be unique within an Azure Stack environment
* Some resource names must be alpha-numeric
* Some resources are case-sensitive
* Some of the objects names are constrained by length. This can be determined in the portal by reviewing the information provided in the portal

Create storage account 
The cost of your storage account 
depends on the usage and the options 
* Name O 
. AzureSta Local 
Account kind O 
General purpose 
The field can contain only lowercase letters and numbers. Name must be between 3 and 
24 characters. 

Figure 29. Constraints error in Azure Stack Portal

As example in the lab associated with this series we will use the following naming convention. This does not cover all naming standards for the lab but is intended to be high level guidance only.

* Contoso Azure Stack: CAS
* Business Units:
  + Human Resources: HR
  + Finance: Fin
  + Information Technology: IT
  + Development: Dev
* And so on

**Good to Know Stuff**

**Azure Resource Manager Limits:**

Azure Resource Manager quotas are applied to Azure Stack. These are applied at the subscription level and in general are the same values as those found in Azure. The exception is any recent changes in Azure that have not yet propagated to Azure Stack.

These limits are applied per-region for each region accessible by your subscription. Service management quotas are limits that are applied per-subscription. These will be discussed in a later port.

Subscription Level Throttling

|  |  |
| --- | --- |
| Resource | Default Limits |
| Resource Quota Limit | 800 |
| Resource Group Name Length Limit | 90 |
| Resource Group Quota Limit | 800 |
| Resource Move Limit | 800 |
| Deployment Name Length Limit | 64 |
| Deployment Quota Limit | 800 |
| Subscription Tag Quota Limit | 900 |
| Subscription Tag Name Quota Limit | 100 |
| Tenant Resources Query Subscription Count Limit | 40 |
| Tenant Resources Query Subscription Batch Size | 10 |
| Tag Key Limit | 512 |
| Tag Value Limit | 256 |
| Max Tags per Resource | 15 |
| Realtime Tag Count Threshold | 600 |

Tenant Level Request Throttling

|  |  |
| --- | --- |
| Resource | Default Limits |
| Max Tenant Read Requests | 15000 |
| Max Tenant Write Requests | 1200 |
| Tenant Throttling Window Time | 01:00:00 |
| Tenant Throttling Bucket Size | 00:05:00 |

Health Check Request Throttling

|  |  |
| --- | --- |
| Resource | Default Limits |
| Unauthenticated Throttling Max Requests | 15000 |
| Unauthenticated Throttling Window Time | 01:00:00 |
| Unauthenticated Throttling Bucket Size | 00:05:00 |

Per Subscription Throttling

|  |  |
| --- | --- |
| Resource | Default Limits |
| Max Subscription Bad Requests | 15000 |
| Max Subscription Write Requests | 1200 |
| Subscription Throttle Window Time | 01:00:00 |
| Subscription Throttle Bucket Size | 00:05:00 |

**Timers:** Updates to the subscriptions in the portal are based on timers. If you create a new subscription the change is immediate once the subscription creation is completed. However, in order to see the changes, you must refresh the portal.

General portal updates to subscriptions are hourly, so will take place post change within 60:00 minutes of the changes.

**Lab Design Decisions:**

Contoso has naming standards that they require their tenants to follow. As you can see in the table below we have defined the naming for the HR and Finance subscriptions. As these will be set up as publicly visible offers, HR and Finance tenants will be required to follow a naming standard when creating their subscriptions.

IT and Dev will have delegation enabled (this capability is described in a later post) and as such the subscriptions will be created for each business unit and a delegated admin will manage each teams’ tenants, naming standards, etc.

Goals:

* Design enterprise strategy based upon business units
* Local admins will manage users/offers for their business unit
* Use an easy to track naming strategy

|  |  |  |
| --- | --- | --- |
| **Business Units** | **Subscription Name** | **Subscription Administrator** |
| Human Resources | CAS-HR-HRAdmin  CAS-HR-HRuser1 | HR Admin |
| Finance | CAS-Fin-FinAdmin  CAS-FIN-Finuser1 | Fin Admin |
| Information Technology | CAS-IT-Subscription | IT Admin |
| Development | CAS-Dev-Subscription | Dev Admin |

# Services: Resource Providers

Azure Stack Services

By now you know that the key value of Azure Stack is about running Azure services in your datacenter. This allows you to transform your on-premises datacenter resources and services into cloud services and cloud-native applications within your organization. At this point you are likely starting to think about what solutions you could begin to build in your Azure Stack lab environment. There are several categories of tenant solutions which such as custom line-of-business (LOB) applications, microservice applications, managed SharePoint and SQL solutions or dev/test environments. You might be asking what you need to do to get prepared for this transformation, what services can you expect in the first version of Azure Stack and how should you be thinking about these services in comparison to those in the public Azure cloud?

**Azure Solutions and Services**

Microsoft Azure is comprised of a series of discrete infrastructure and platform *services* deployed across our global datacenters. Think of these as the discrete capabilities which by themselves provide a type of service such as storage, web sites or virtual machines. One or more of these services can be composed to support various customer *solutions*, which are tangible outcomes which customers use to enable various aspects of their business. Using an infrastructure example, services such as Virtual Machines, Virtual Network and Blob/Table Storage would enable traditional server-based application services like SQL Server or SharePoint Server. Alternatively, these services allow customers to run cloud-first applications as solutions using on-premises deployments of platform services Azure App Service (Web Apps) and Docker-integrated containers. This is a key difference between what most customers can achieve today with traditional server virtualization since Azure platform services allow your developers to describe, deploy and control their applications and solutions using standard Azure APIs and using familiar Azure services that expand beyond traditional virtual machine-based solutions. By bringing these to each customer’s on-premises environment, we expand the Azure ecosystem of services to your datacenter and allow you to envision new hybrid scenarios which were simply not possible in the past.

At general availability (GA) Microsoft will provide the first wave of Azure services which will enable a series of the solutions available today. These services were announced at our Ignite conference and include the following capabilities:

* Azure Compute services including Virtual Machines, Virtual Machine Scale Sets, Key Vault and Virtual Machine Extensions including support for Linux and Windows Server containers on Docker.
* Azure Networking services including Virtual Networks, Load Balancers and the VPN Gateway
* Azure Storage services including Blobs, Tables and Queues
* Azure App Services including Web Apps, Mobile Apps, API Apps and Service Fabric
* Azure Portal services including the familiar end-user experience and an Azure service admin portal experience for managing subscriptions, quotas, image gallery and metering.

The following diagram outlines this relationship.



Figure 30. Understanding Azure Services in Azure Stack

A key focus of Azure Stack is to provide an Azure-consistent experience for developers and consumers of these services in your environments that goes beyond the user interface. This means that the Azure Resource Manager (ARM) API can be used to compose services to build your solutions and host your organization’s applications on Azure stack in a consistent way and using a consistent set of tools by your developers. However, Azure services on Azure Stack can sometimes contain some differences due to the following reasons: *API Version*, *Scale* and *Dependencies*.

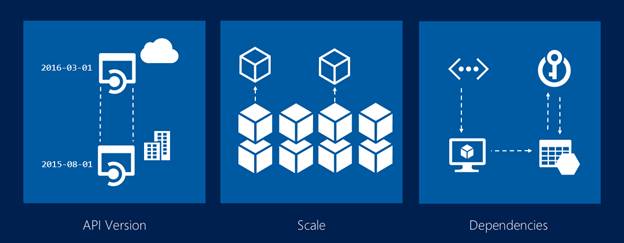


Figure 31: Azure Stack differences view

A primary consideration when developing solutions on Azure Stack is understanding the **API version** of the service you plan to deploy. The version of the API on Azure Stack may be different than the current version in Azure, specifically when using traditional infrastructure services as outlined above. Azure services are expected to debut in Microsoft’s Azure datacenter first, then eventually be supported within Azure Stack where feasible. While there may be API versions different at GA, we expect the gap between API versions to get closer over time once the first Azure Stack is released and platform-based Azure services (such as App Service) are expected to evolve at a faster pace much like they do today.

The next consideration around Azure services on Azure Stack is their **scale**. In many cases the scale of an Azure service is hardware dependent. To illustrate this point, consider Azure’s [N-Series Virtual Machine](https://aka.ms/ASWP_VMSize) sizes. One of the key differentiators of this virtual machine size is that these systems leverage specific NVIDIA GPUs to support solutions such as desktop accelerated applications and operating system experiences. In addition, these virtual machines are assigned a large amount of memory, high CPU counts and are backed by large quantities of SSD storage. Using this example its clear to understand that to provide a similar virtual machine sizes on-premises a similar hardware set with a large amount of scale would be required to support even a single virtual machine at this size, let alone a large amount of these. In Microsoft Azure datacenters, the smallest scale unit spans hundreds of servers while the smallest scale unit in Azure Stack for on-premises infrastructures is envisioned to be much smaller (under 10 servers). This scale difference spans each of the Azure services outlined above, which makes it critical to the planning process for any solution you wish to provide to your tenants.

The last consideration is each service’s **dependencies**. To enable familiar solutions and capabilities within Azure Stack, each service may have a dependency on another *resource provider*, a specified *agent version* or dependent *hardware capabilities* to support an expected set of functionalities. First let’s look at agent-based dependencies. Leveraging custom Linux-based images in the Azure Stack Marketplace is a good example of an *agent-based dependency*. In order for a Linux distribution to be supported on Azure Stack the [Azure Linux Agent](https://aka.ms/ASWP_WALinuxAgent) must be version 2.1.3 and higher. Some versions of Linux distributions (and network virtual appliance vendors who base their solutions on Linux) may contain pre-installed versions of the Azure Linux Agent which are below the required version, therefore these images will require agent updates prior to being made available on Azure Stack. Next, let’s look at resource provider dependencies. An example of a *resource provider dependency* would be the [Azure App Service](https://aka.ms/ASWP_AppSvc) autoscaling feature. In Azure, the App Service autoscaling feature provides scalability (scale up or scale down) of an App Service worker pool based on a given set administratively set parameters. This functionality, while expected in future releases, will not be available to the App Service in the first release of this service on Azure Stack as it is dependent on the Application Insights resource provider. While present in Azure Stack, support for specific features within the Microsoft Insights Namespace are currently not present in its first release to enable this specific feature within the App Service within Azure Stack environments. Finally, hardware available within a given OEM Azure Stack solution may influence how a given resource provider functions. An example of a hardware *capability dependency* can be seen in the Azure Key Vault service. [Azure Key Vault](https://aka.ms/ASWP_KV) provides the ability to safeguard cryptographic key information for Azure-based solutions, applications and services and comes in two service tiers: Standard and Premium. In the initial release of Azure Stack, OEMs will not be providing Hardware Security Modules (HSMs) as part of the first wave of hyper-converged solutions and as such, features associated with the Azure Key Vault Premium service tier will not be available upon the first wave of Azure Stack OEM solutions.

**Azure Services Capabilities in Azure Stack**

Understanding these potential differences in Azure Stack, you might be wondering how to plan to onboard your solution around how these dependencies may influence your solution design on Azure Stack at general availability. Furthermore, if you are an early adopter of Azure Stack you might be wondering which technical preview (TP) you can expect to see these features and port your solution for early testing. The following table is provided to support a greater understanding of the Azure Services envisioned for the first release of Azure Stack:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Solution Area** | **Service or Resource Provider** | **Preview Milestone** | **API Version** | **General Considerations** |
| **Azure Resource Manager** | ARM API and Portal | TP1 | N/A | The API functionality is not significantly different and specific Microsoft-based tools such as the Azure CLI, Azure PowerShell and Azure SDK (Visual Studio) are expected to be compatible with Azure Stack. Third-party tools are also expected to function properly, but some may have hard-coded values such as API endpoints, URIs and the ability to connect to multiple ARM endpoints (not just Azure) which could cause unforeseen incompatibilities. |
| **Compute** | Virtual Machines | TP1 | 2015-06-15 | While the virtual machine sizes in Azure are equivalent to those in Azure Stack (like Azure, custom sizes are not supported) support will initially come for A, D Dv2, DSv2 and F-series Virtual Machines (specific sizes/SKUs). Note that some current and legacy distributions of Linux which may have an incompatible version of the Azure Agent will require that updated versions of the agent be installed as outlined above. |
|  | VM Scale Sets | TP3 | 2015-06-15 | Due to the initial scale of OEM solutions at general availability, it is expected that VM Scale Sets would focus on smaller scale scenarios to support customer applications. |
|  | Service Fabric | GA | 5.3 in preview and the final version is to be determined | Service Fabric in Microsoft Azure currently requires three fault domains however Azure Stack will support deployments with an instance of one to support Azure Stack POC deployments. |
| **App Service** | WebApps  API Apps  Mobile Apps  Functions | TP1  TP2  TP2  TP3 | 2016-03-01,  2015-08-01,  2015-07-01,  2015-06-01,  2015-05-01,  2015-04-01,  2015-02-01,  2014-11-01,  2014-06-01,  2014-04-01,  2014-04-01-preview | Being a development-focused platform, App Service API versions in Azure Stack will be expected to keep pace with the existing change cycle in Azure. It is important to understand that App Service is a single RP. The API version is for the App Service RP specifically, not for each feature. As discussed above, App Service autoscale is not supported due to dependencies on the Microsoft Insights Namespace. Note that while this functionality was introduced first in TP1, some functionality requires later technical previews to support. |
| **Data & Storage** | Storage Blobs, Tables, Queues | TP1 | 2015-04-05 | This service is expected to generally work the same as it does today in Azure and will be focused on single-region support in Azure Stack (known as Locally redundant storage – LRS). Features such as Premium Storage are supported at the API level (template support for Premium Storage) however speeds may vary based on the OEM solution selected. Note that some capabilities such as IOPs throttling will come in later technical previews. |
|  | SQL Server | TP3 | N/A | This is an IaaS-based service that supports SQL Server versions 2014 SP1 and SQL Server 2016 in technical preview to support create, update and delete commands of using familiar SQL Server virtual machine Gallery Items. |
|  | MySQL Server | TP3 | N/A | This is an IaaS-based service that supports MySQL Server versions 5.5 and 5.6 in technical preview to support create, update and delete commands of using familiar MySQL Server virtual machine Gallery Items. |
|  | KeyVault | TP2 | 2015-06-15 | As discussed above, the Azure Stack version of Key Vault only supports the “Standard” service tier and therefore HSM protection of keys is not available at the general availability milestone. |
| **Networking** | Virtual Network | TP1 | 2015-06-15 | While most functionality will be the same, an important consideration is that standard public facing DNS URIs for services and solutions would need to be handled manually by customers outside of Azure Stack as the internet-facing DNS system that Azure provides is not available in the initial release. In addition, solutions should leverage IPv4 IP addressing in the initial release of Azure Stack. |
|  | Load Balancer | TP1 | 2015-06-15 | This feature is expected to generally work the same as Azure and Windows Server 2016 today and uses the Windows Server 2016 Network Controller to provide this functionality. |
|  | VPN Gateway | TP1 | 2015-06-15 | This feature is expected to generally work the same as Azure and VPN connections from on-premises environments to Azure Stack, Azure to Azure Stack and Azure Stack to Azure Stack is supported. Note that recently released network capabilities such as network peering are not currently available in the initial release of Azure Stack. |

**Envisioning your solution in Azure Stack**

Taking the above into consideration, it’s important to look at existing solutions which are likely based on server virtualization and consider the possibilities when running these systems on Azure services within your datacenter. As an example, take a traditional tiered web application running a virtualized infrastructure using Windows Server web server (IIS) as front-end and a database running MySQL on the back-end. In this example, you could look to these solutions and determine if the features being used by the web server front-end are compatible with the version of Azure App Services that exists in Azure Stack. The same could be said for the data which exists in MySQL, which you could at a minimum evaluate if the versions were compatible to support porting this solution from a single virtual machine to a shared IaaS instance of MySQL Server. In doing this analysis, you may find that many of your existing applications or solutions which currently require isolated virtual machines could be transitioned to using native Azure cloud-based services. Not only would your organization find potential savings in operating system and associated third party management licenses, you would likely see operational benefits given the differences between management and maintenance of individual virtual machines and consolidated platform services such as Azure Stack.

NOTE:

In order to offer PaaS services in your Azure Stack environment, you will need to add additional resources providers. At this time three PaaS resource providers are available. The step by step install process for each of these can be found by clicking on the hyperlinks above.

* [MySQL Resource Provider](https://aka.ms/ASWP_MySQLRP): create MySQL servers and databases through Azure Resource Manager deployment templates and provide MySQL databases as a service. MySQL databases, which are common on web sites, support many website platforms.
* [SQL Resource Provider](https://aka.ms/ASWP_SQLRP): adapter to expose SQL databases as a service of Azure Stack. After you install the resource provider and connect it to a SQL Server instance, you and your users can create databases for cloud-native apps, websites that are based on SQL, and workloads that are based on SQL without having to provision a virtual machine (VM) that hosts SQL Server each time.
* [App Service Resource Provider](https://aka.ms/ASWP_AppSvrOver): Infrastructure VM's deployed to support hosting Web, Mobile, and API Apps.

Once a Tenant obtains a subscription (regardless of method) they will have to register some of the RPs.

Example: Register-AzureRmResourceProvider -ProviderNamespace Microsoft.Insights

**Good to Know:**

The following core resource providers are visible in the portal in the Resource Providers blade. Removal of any of these Resource Providers will break core functionality in Azure Stack.

|  |  |
| --- | --- |
| Resource Provider | NameSpace |
| Compute | Microsoft.Compute.Admin |
| Health | Microsoft.InfrastructureInsights.Admin |
| Network | Microsoft.Network.Admin |
| Storage | Microsoft.Storage.Admin |

Additional resource providers that are installed by default are visible under the Resource Explorer => Resource Providers blade. These are the base resource providers that enable the functionality we see in the portal after installation. Removal of any of these Resource Providers will break core functionality in Azure Stack.

|  |  |
| --- | --- |
| Resource Provider | NameSpace |
| Authorization | Microsoft.Authorization |
| BackupAdmin | Microsoft.Backup.Admin |
| Commerce | Microsoft.Commerce |
| CommerceProviders | Microsoft.Commerce.Providers |
| Fabric | Microsoft.Fabric |
| FabricAdmin | Microsoft.Fabric.Admin |
| Gallery | Microsoft.Gallery |
| GalleryAdmin | Microsoft.Gallery.Admin |
| GalleryProviders | Microsoft.Gallery.Providers |
| InfrastuctureInsightsProviders | Microsoft.Infrastucture.Insights.Providers |
| Insights | Microsoft.Insights |
| InsightsProviders | Microsoft.Insights.Providers |
| KeyVault | Microsoft.KeyVault |
| KeyVaultAdmin | Microsoft.KeyVault.Admin |
| Resources | Microsoft.Resources |
| ResourcesAdmin | Microsoft.Resources.Admin |
| Subscriptions | Microsoft.Subscriptions |
| SubscriptionsAdmin | Microsoft.Subscriptions.Admin |
| SubscriptionsProviders | Microsoft.Subscriptions.Providers |
| UpdateAdmin | Microsoft.Update.Admin |

SQL RP:

The SQL resource provider allows you to provide your tenants with SQL databases for cloud-native apps, as well as websites / workloads that are based on SQL without having to deploy a SQL server every time. One or more SQL 2014 SP1 Evaluation instances can be configured to host tenant DB's. This is an adaptor designed for Azure Stack TP3, and differs from SQL DB as a Service in Azure. A single script is used to deploy this resource provider in Azure Stack TP3.

It is important to note that the databases are on a shared host server. If an organization requires segregation of data, they will either require a specific host server or they can deploy a full SQL installation on a virtual machine. The latter option also enables customers to take advantage of additional SQL capabilities such as high availability, disaster recovery, etc.

App Service RP:

App Service is deployed using a custom application and several scripts. The App Service environment deployed will consist of:

* Controller
* Management (Two instances will be created)
* Front-End
* Publisher
* Worker (in Shared mode)

Installation of app service will create several logins and DB's in your SQL instance. If a failure to install occurs, please ensure you remove these prior to attempting a reinstall. It will also take a significant amount of time for the installation to complete. This is due to the wait time required to ensure all components are fully deployed and in "ready" mode before the installer will report success.

If you require additional worker roles, documentation to add these is located [here](https://aka.ms/ASWP_AppWR).

Key Vault RP:

Key vault is installed by default during the deployment of Azure Stack TP3, however at this time Key Vault needs to be explicitly registered. Please see instructions [here](https://aka.ms/aswp_kvreg) for how to do this.

Custom RP's

If your ambition is to develop custom resource providers, you must provide the following:

* User Interface Extensions for the portal
* REST Endpoints accessible via HTTPS covering the following:
  + Portal Endpoint
  + Notification Endpoint
  + Usage Endpoint

Resource Provider creation will be possible but is out of scope of this series and is not publicly documented yet.

**Lab Design Decisions:**

The following resource providers will be included in the lab to enable provisioning of both IaaS and PaaS services to tenants.

|  |  |
| --- | --- |
| **Resource Provider** | **Default** |
| Compute | Yes |
| Health | Yes |
| Network | Yes |
| Storage | Yes |
| Key Vault | Yes |
| MySQL | No |
| SQL | No |
| App Service | No |

# Quotas, Plans and Offers

The bulk of the design decisions you will need to make once Azure Stack is installed are related to the configuration of the services you will offer to your tenants. These are configured using quotas, plans, and offers.

The various components that make up Azure Stack are all connected and define the capabilities and quantities that a tenant can consume. The core components include regions, subscriptions, offers, plans, quotas, and services. Other than subscriptions, these are the components that the Azure team manages, that you will need to manage in Azure Stack. The diagram below, while a bit of an eye chart, outlines how these pieces all interact.

At a high level this is how it works:

1. Add services that you want to deliver to your tenants
2. Create quotas to define the amounts of the service resources you want tenants to be able to consume
3. Create plans that contain the services and applicable quotas
4. Create offers that align to the plans
5. Create ad on plans as needed and add to offers

Once all this is competed, your tenants will be able to create subscriptions aligned to the offers. Let's dive into each of these components and look at how each works and some things you will want to keep in mind when designing your solutions.

How Quotas, Plans, Offers, and Subscriptions Interact 
in Azure Stack 
Services 
Compute 
Storage 
Network 
A 
Note: these are illustrative GA 
services; not meant as an 
accurate TP2 list 
Compute Quota - 
small 
Compute Quota - 
la e 
Storage Quota 
Network Quota 
Quota 
Quotas determine 
the amount of 
resources a user can 
consume 
Plan - VMs 
Compute 
Storage 
Network 
Plan #2 — Increase 
mpute quota 
Compute 
Plan #3 - Web 
A Service 
Plans allow administrators 
to group services and their 
quotas to be offered to 
tenants 
Offer #1 — laaS only 
Base 
Plan #1 
Offer #2 — laaS and Web 
Base 
Plan #1 
Add On 
Plan #2 
Add On 
Plan #3 
Offers allow administrators 
to group plans to be offered 
to tenants 
Subscription 
1 
Subscription 
2 
Subscription 
3 
Account 1 
user@ contoso. 
onmicrosoft.com 
Account 2 
user@ fabrikam. 
onmicrosoft.com 
Subscriptions connect users to 
offers; each user can have 
multiple subscriptions 

Figure 32: View of how components interact

**Quotas:**

Now that we have defined the services we would like to offer, we need to determine how much of each service we want to allow a tenant to consume. To do this we configure quotas. You can create the upper limits for quotas only.

Each resource provider will have quotas configured for it. It is critical to keep in mind the amounts of each resource (i.e. storage, public IP's) that you have available in the environment, to ensure that you do not exceed your capacity. A basic view of capacity within the infrastructure can be reviewed by selecting Region Management => Local (Region) => Resource Providers, and selecting each resource provider. Coming soon is a capacity resource provider which will provide a view into the actual and used capacity within the environment.

|  |  |  |
| --- | --- | --- |
| Resource Provider | Configurable Quotas | Default Quotas |
| Compute | Yes | Default |
| Key Vault | No | Unlimited |
| Network | Yes | Default |
| Storage | Yes | Default |
| MySQL | Yes | None |
| SQL | Yes | Default Basic  Default Standard  Default Premium  Default Admin |
| App Service | Yes | Default |

In most cases, you will have several different quotas that align to the needs of different business groups or customers. Several of the resource providers offer a default set of quotas, however you can add your own that best meet your needs. Default quotas tend to have the maximum amount of resources available that can be configured for that resource provider. NOTE: they do not necessarily align to the actual resources available in your Azure Stack environment.

Enterprises may want to configure quotas based upon the various business units that will consume resources, for instance you may want to have specific quotas for developers. You may also choose to configure quotas based on a specific application.

When defining your quotas, you will need to keep in mind the following:

* Infrastructure resources: the amount of compute, storage and network resources available in your azure stack environment.
* Resource Providers installed: each resource provider, with the exception of Key Vault, has quotas that will need to be configured. All resource providers consume at minimum compute, network, and storage resources
* Number of tenants: each tenant will consume resources, most notably compute, storage, and network. Depending upon the services offered to them, they may also consume other resources (i.e. sql databases)
* Tenant requirements: Designing quotas specific to an application is a fairly easy process as the compute, storage, and network requirements of the application are known entities. To create quotas for groups of users, you will need to understand their resource usage patterns, if they will deploy a set of resources with limited growth, or they will deploy with a predictable growth rate, or if they tend to be volatile users who will be deploying and deleting on a regular basis (i.e. developers).
* Applications: Each application instance will require as specific group of resource providers and a known amount of resources. This tends to be the easiest quota to define

Defining quotas will require discussions with the various groups of tenants within the enterprise, including developers of the applications to be deployed, to determine exactly what their needs are. For the service provider, defining quotas is more aligned to the offerings they would like to sell to their tenants.

Although you can create quotas at the same time as you define your plans, I tend to create my quotas separately. In this manner, I can be sure I am deploying exactly the quotas I need.

In the future, the capacity resource provider will enable you to view capacity for CPU, memory, storage, public IP pool usage, etc. and alert when reaching capacity. At this time when designing you need to keep in mind the capacity you have in your lab and design accordingly.

When you are ready to configure your quotas, you can access step by step instructions [here](https://aka.ms/aswp_quotas).

**Plans:**

Plans are required to create the groupings of services and their applicable quotas that will be offered to tenants. Most designs will include more than one plan. Each plan will be tied to a specific set of quotas based upon the services offered in that plan.

The first step to designing your plans is to understand the following:

* Services to be included in each plan
* Users who will access each plan
* Will users be assigned to specific plans/offers, or will you advertise them via offers
* Quotas limits required for each service in a plan

Tenants can consume one or more plans within a single subscription. This configuration may be beneficial when a tenant requires more or different services that are not offered within a single plan.

Two distinct types of plans are available:

1. Base plan: contains the core services to be offered to customers
   1. One base plan can be added per offer
2. Add on Plan: contains additional services that can be added to a base plan
   1. Can be used to extend compute/storage/network to a base plan
   2. Can add multiple add ons to a base plan
   3. Cannot offer an add on by itself. To do this it would need to be created as a base pan

In the case of the service provider add on plans could be upgrade offerings to a base plan. While combining everything in a single plan may be optimal in some cases, organizations may want to have a base plan, and offer additional services using add on plans. For instance, a service provider may decide to offer IaaS services as part of their base plans, but all PaaS services will be add on plans with additional charges.

For example if you want tenants to be mindful of their resource usage, you could have a relatively small base plan (what small actually means depends on the VM sizes your tenants deploy), and as tenants run out of capacity they will need to add an add-on plan. This will function as an alert to them that they have already consumed a certain amount of resources.

A good naming standard is recommended when configuring your plans, so you can easily track the tenants using the plan as well as its purpose. Naming can include the business unit, the sizing, customer name, or the application that the plan is designed for. Similarly, a smaller organization may decide that one base plan is all that is required however they may choose to offer SQL as an add on to select users, or for select applications that require it. When thinking about the naming conventions for plans and offers, adding "add on" to any plans that can be acquired and added to a base plan is a good practice.

 Below are a few high-level examples of plans. These are examples only, and each customer will have plans designed to fit their specific needs.

**Enterprise Example**

This example is based upon departments within an organization. It can easily be combined with the Application example as required. This example is based on base plans only, however add on plans could easily be added as required

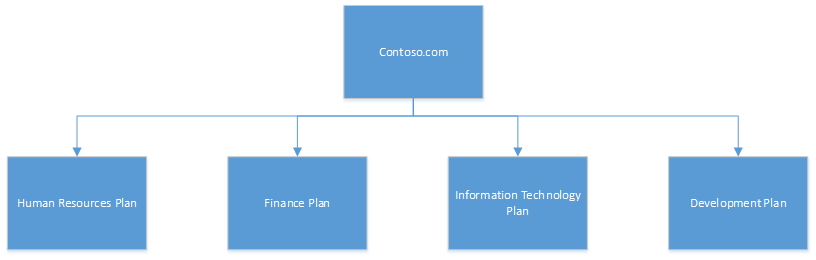


Figure 33: Enterprise High Level Example

**Application Example**

This example is based upon applications deployed within azure stack that are available across business units within an organization. This example is based on base plans only, however add on plans could easily be added as required

V ddV 
utu g ddv 
ueld'ddv 
ueu a 

Figure 34: Application Focused High Level Example

**Service Provider Example**

This example represents customers accessing a service provider infrastructure to obtain IaaS/PaaS service

Trial 
AaS•'Su m 
App Ed 
su Ed on *Lan 

Figure 35: Service Provider High Level Example

As you can see from the above examples, plans can easily get quite complex. Note that each plan you create will have some administration required, and as you add more and more plan options the planning calculations for resource consumption will become more difficult.

The step by step instructions for creating a plan are [here](https://aka.ms/ASWP_Plan).

**Offers:**

Selection of an offer when a user creates a subscription, is how your tenants will access the resources they require. When a tenant using the portal signs up for a new subscription they will only able to select a single offer. Once the subscriptions is created they can add additional offers/add on plans. As a service administrator, you can create these add on plans and make them available so a tenant can then subscribe to them to provide them with the additional resources they require.

At minimum, an offer must contain at least one base plan. You can add multiple base plans as well as add on plans to an offer to provide your tenants who will consume this offer with exactly the resources they require. All tenants that subscribe to a specific offer will have access to the plans and associated quotas that are contained in the offer.

Offers can either be advertised or assigned.

**Advertise:**

An offer that is advertised is visible to all tenants accessing the portal and represents the self-service model for obtaining services. This means that any tenant with rights to access the portal can add an advertised plan to a subscription. This is the same behavior for add on plans. Advertised offers must be set to public state. By default, all new offers are set to private state.

To create a subscription and configure an advertised offer, follow the instructions [here](https://aka.ms/aswp_sub).

**Assign:**

Assigning offers allows you to configure multiple offers that are not visible to your tenants. In order for them to access the offer you must assign their tenant user ID to the offer. There is an overhead to this option as every user must be assigned to the applicable offers by a service administrator. Once assigned when the tenant logs into the portal they will see the new subscription and have access to the resources and quotas aligned to the offer.

Why you may want to do this:

* As a service provider, you want to assign company specific offers to customers
* Within an enterprise, you want to ensure only users of a specific business unit have access to an offer
* You want to tightly manage the resources that users can consume

Why you may not want to do this:

* Managing all access to offers manually has a huge overhead. One should think carefully before they define all their offers as assigned.

Note: for businesses that have local administrators within customer tenants or business tenants, please see the information on delegation. This will allow you to provide a set of offers to specific business units or tenants, and allow a local administrator to manage the access to the resources

There is a third state option that can be applied to an offer, decommissioned. This state prevents an offer from being advertised. Tenants already consuming this offer will continue to have access to the resources provided within the associated plan and quotas, however no new users can be added.

To create an offer, see the step by step instructions [here](https://aka.ms/ASWP_Offer).

**Good to Know Stuff**

* Roles Based Access Control (RBAC) can be configured to secure access to these (and other) resources
* Quotas can be configured that exceed the capacity of your infrastructure. This could cause the provider to run out of resources.
* Additional capacity can be applied to a quota, however users already consuming the current plan and quota will not see the updates for up to an hour
* When a user adds an add on to an existing subscription, the additional resources could take up to an hour to appear
* To add resources immediately, you may need to configure a new add on for the tenant to consume immediately
* When a user runs out of capacity, they will receive errors when they try to create new resources within their subscription. Capacity Management capabilities will be coming in a future release
* To change plans the tenant will require a new subscription
* When a plan or offer is decommissioned tenants already consuming the plan and quotas will continue to have access to those resources
* When a plan or offer is removed tenants already consuming the plan and quotas will continue to have access to those resources
* If you change a quota amount to reduce it, tenants already consuming more that the new quota amount will continue to receive the quota they already are consuming but will not be able to add more resources
* Changes to a quota, plan, or offer are global and will impact all tenants

**Lab Design Decisions:**

In order for our tenants to access services we need to set up offers, plans and align quotas to each plan. As mentioned at the being of this series when we design our offers we need to understand the applications each of the business units will require.

HR has a series of websites that they require. These use MYSQL on the backend. To meet the application needs of this team they will require the AppService and MySQL resource providers, as well as Compute, Storage and Networking. They don’t use a lot of resources, so quotas will be set at 50% of the defaults and this will support both current needs as well as known future needs.

Finance on the other hand relies heavily on an off the shelf cloud ready application that uses SQL server as its database. Along with the base IaaS Services, they will also require the SQL resource provider. Again, their usage is not projected to be overly heavy so quotas at 50% of the defaults will meet their needs

The IT department will manage all the operational tasks required for the Contoso Cloud, as well as consume resources for deployment / quality testing of the applications the business requires, both current and future releases. IT will also be responsible for ensuring any custom images that the business requires are made available in the marketplace. Development will develop and maintain all the applications that Contoso requires. They will work closely with IT on the release of these applications to the business.

Both groups will require all the IaaS services as well as the additional resource providers that Finance and HR have access to. In addition, Development is also working integrating Key Vault capabilities into the business so both require this additional resource provider as well.

**Base Offers and Plans**

Goals:

* Use an easy to follow naming standard
* Create separate quotas/plans/offers for each business unit
* Delegate administration for non-IT users to team admins
* Configure quotas so they can be adjusted on a per team basis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Offers** | **Advertised /Assigned** | **Assigned Tenants** | **Plans** | Base Services | **Quota Name** | **Quota Limits** |
| CAS-HR-Offer | Advertised | None | CAS-HR-Plan | Compute  Network  Storage  *App Service*  *My SQL* | CAS- Compute  CAS-Network  CAS- Storage  *CAS-AppSvc-Premium*  *CAS-MySQL-Premium* | 50% of defaults |
| CAS-Finance-Offer | Advertised | None | CAS-FIN- Plan | Compute  Network  Storage  *SQL* | CAS- Compute  CAS-Network  CAS- Storage  *CAS-SQL-Premium* | 50% of defaults |
| CAS-IT-Offer | Assigned | IT1  IT2  IT3 | CAS-IT-Plan | Compute  Network  Storage | CAS-IT-Compute  CAS-IT-Network  CAS-IT-Storage | 100% of Defaults |
| CAS-Dev-Offer | Assigned | Dev1  Dev2  Dev3 | CAS-Dev-Plan | Compute  Network  Storage  *App Service*  *My SQL*  *SQL*  *Key Vault* | CAS-Dev-Compute  CAS-Dev-Network  CAS-Dev-Storage  *CAS-AppSvc*  *CAS-MySQL*  *CAS-SQL*  *CAS-KeyVault* | 75% of Defaults |
| CAS-Trial-Offer | Advertised | None | CAS-Trial-Plan | Compute  Network  Storage | CAS-Trial-Compute  CAS-Trial-Network  CAS-Trial-Storage | 2 VM 10GB Storage 4 x IP |

**Add On Plans**

|  |  |  |
| --- | --- | --- |
| **Add On Plans** | **Quotas** | **Quota Limits** |
| SQL Plan | CAS-SQL-AO | CAS-SQL-Premium |
| App Service Plan | CAS-AppSrv-AO | CAS-AppSvc-Premium |
| MySQL Plan | CAS-MySQL-AO | CAS-MySQL-Premium |
| Key Vault Plan | CAS-KeyVault-AO | Unlimited |

Note: add on plans may result in errors when a tenant tries to add them if added after a base plan is created in TP2

# Delegation

Delegation is the ability to put other people in charge of creating offers and signing up users. As a service provider, you may want to offload the creation of offers and signing up users to the organizations of resellers that are purchasing your services. In an enterprise, you can delegate the same ability to add users and create offers to the various divisions or subsidiaries. This offloads the day to day operations from the service administrator(s) tasked with operating Azure Stack, to local resources.

Setting up delegation is relatively easy, with huge benefits for the service administrator. There are three roles that are used when setting up delegation:

* **Service administrator:**  manages the Azure Stack infrastructure, creates an offer template, and delegates others to offer it to their users.
* **Delegated providers**: Create offers for their tenants and sign up tenants
* **Tenants:**  sign up offers, or can be assigned the offers (See subscriptions for a description of these two options)

Below is a diagram of how this process works:

Service Admin 
1. Configure Delegation Offer 
3. CreateOffer w/Required 
Servi ces 
Delegated Provider 
2. Sign up for / Assigned 
Delegation Offer 
4. Create Offers using Template 
in step 3 
5. Optional: Create Custom LJRL 
Tenant 
6. Sign up for / Assigned 
Subscri ption /Offer 

Figure 36: Delegation Process

The step by step instructions for configuring delegation are [here](https://aka.ms/ASWP_DP).

**Good to Know**

* Delegation offer only contains the subscription service
* Any changes to offers with services will apply to all tenants consuming the offer
* Delegated admins can create customer URL's for portal access for their tenants.
* Offers tied to a custom URL will only be visible to tenants using that URL
* Offers that a public will be visible to all tenants

**Lab Design**

In this lab, we will take advantage of delegation for two of the business units. Both IT and development have determined that they will assign an administrator to manage users and redistribute offers. Both will also use a custom URL for their delegated site.

HR and finance however will rely upon the IT team to provision offers to them and manage their users. Offers for both groups will be set to public, so they are visible to the business units, and they will select the one appropriate to them group.

Goals:

* Design a naming standard that is easy to follow
* Business unit admins will manage users and delegated offers

|  |  |  |
| --- | --- | --- |
| **Business Units** | **Delegated Administrator** | **Custom URL** |
| Information Technology | IT Admin | ContosoIT |
| Development | Dev Admin | ContosoDev |

# Tags, Policies, and Locks

**Tags**

Azure Resource Manager provides the ability to use tags to logically categorize and group resources across resources groups within the portal and across subscriptions in Azure Stack. Applications of tags allows you to group resources outside of a single deployment. To do this, you simply apply the same tag to everything you want included.

When creating tags, we use a name/value pair. Some common examples of when you may want to use tags include:

* Grouping components of an application
* Grouping of resources related to applications, cost centers, or tenant units for billing purposes
* Organization of components for management purposes

Careful planning is required when applying tags as each resource or resource group can have a maximum of 15 tags. The tag name is limited to 512 characters, and the tag value is limited to 256 characters.

Tags can be applied thru the portal once resources have been deployed, or we can take advantage of JSON templates to apply tags at runtime. This ensures that any new resources that should be part of a logical grouping are automatically added to the proper groups. Tags can also be added using PowerShell scripts. This is especially valuable if you have a lot of existing resources that require updating with a new tag.

In the lab we are building, I will use tags for various purposes. Tags will be applied to each business unit to enable chargeback or billing for each group. Tags will also be used to define all Windows Operating System Virtual Machines, vs Linux based Virtual Machines.

Portal View of Tags:

Microsoft Azure Stack 
Filter 
By category v 
t) Resource groups 
All resources 
Tags 
Recent 
Tags 
Shift+Space to toggle t 

Figure 37: Tags in the Azure Stack Portal

PowerShell Commands:

The command below will allow you to obtain a list of all tags in a subscription and the number of resources that tag is applied to:

Get-AzureRmTag

To obtain a list of all tags applied to a resource use the following command:

(Get-AzureRmResource -ResourceName <Name of Resource> -ResourceGroupName <Name of RG>).Tags | %{ $\_.Name + ": " + $\_.Value }

To obtain a list of all resources with a specific tag applied, you can use this command:

(Find-AzureRmResource -TagName <Name of Tag> -TagValue <Value of Tag>).Name

Eg. (Find-AzureRmResource -TagName Dept -TagValue Development).Name

For guidance on how to add a tag thru a JSON template check this [webpage](https://aka.ms/ASWP_NewARRG).

**Policy Management of Resources**

To prevent users in your organization from breaking conventions that are needed to manage your organization's resources, Azure Resource Manager has policies. Policy definitions can be utilized to describe the actions or resources that are specifically denied. The policy definitions are applied at the desired scope, such as the subscription, resource group, or a resource. Policies allow you to set explicit allow (default) or deny to resources.

It is important to note that all policies are inherited by all child resources. So, if a policy is applied to a resource group, it is applicable to all the resources in that resource group.

While they are different, policies work in conjunction with Roles Based Access Control settings. Policies require that you are authenticated via RBAC, and they focus on the resource actions at various scopes. Policies are evaluated when a resource is created.

Use Cases:

* Application of required tags on resources for chargeback / billing purposes
* Manage environments of resources
* Apply naming conventions

Policy definitions are created within the JSON templates used to deploy a resource. One or more conditions or logical operators are configured, and these define the actions that are to be taken and the effect defining what will happen when a condition is fulfilled.

A policy requires the following three components:

**Parameters:** values that are specified when the policy is assigned

**Condition/Logical operators:** a set of conditions that can be manipulated through a set of logical operators

**Effect:** what happens when the condition is satisfied – either deny or audit.

The example below allows you to configure a deny policy based upon the naming prefix and suffix not being met.

{

"if" : {

"not" : {

"field" : "name",

"like" : "namePrefix\*nameSuffix"

}

},

"then" : {

"effect" : "deny"

}

}

More information on the configuration and usage of policies can be found [here](https://aka.ms/ASWP_RMP). [Azure Stack Tools](https://aka.ms/ASWP_Policy) on GitHub also contains a policy example for Azure Stack.

**Locks**

To restrict the operations that can be taken on resources we use locks. Locks can be applied at the subscription, resource group, or resource level and they will when applied allow two types of enforcement:

* CanNotDelete: Authorized users can only read and modify resources
* ReadOnly: authorized users can only read from the resource but may not modify or delete it

By using locks, you can prevent deletion of or tampering with high value assets. It is best practice to apply locks on resources that tend to be static in nature, and those that will have high impact if they are deleted.

Locks can be applied thru the portal or using any of the common tools for managing Azure Stack; PowerShell, CLI, or REST API's. To create or delete a lock you require Owner access to the resource.

Locks applied to a parent will flow down to all child resources thru inheritance, with the most restrictive lock being applied if there is more than one lock on a resource.

**Good to Know Stuff**

* A maximum of 15 tags can be applied to any one resource or resource group
* Policies work together with RBAC to deny or audit access to resources
* Policies are inherited by all child resources
* Resource locks should be used on static resources

**Lab Design Decisions:**

To provide the Operating System images Contoso requires, two images will be downloaded using Market Syndication with Azure. Contoso will also implement tagging within the organization. Extensions for application of tags to the operating systems will be utilized, however tagging of resources within the business units will be based upon corporate policy with the expectation tenants will tag their own resources.

Should resources without tags be found, the IT department will tag them using PoSH.

Given there will be multiple administrators of the Azure Stack solution on the IT team, and to meet the needs of change management, locks will be utilized for all offers.

Marketplace items to add:

|  |  |
| --- | --- |
| Item | Customization |
| Windows 2016 Server | Tags |
| Linux Image | Tags |

Tags:

|  |  |  |  |
| --- | --- | --- | --- |
| Tag Name | Tag Value | Tag Application | Application Method |
| Dept | CAS-Finance | All Fin Resources | Portal/PoSH |
|  | CAS-HR | All HR Resources | Portal/PoSH |
|  | CAS-IT | All IT Resources | Portal/PoSH |
|  | CAS-Dev | All Dev Resources | Portal/PoSH |
| Operating System | Server2016 | All 2016 VMs | JSON |
|  | Server2012R2 | All 2012 R2 VMs | JSON |
| Application |  |  |  |

Locks:

|  |  |  |
| --- | --- | --- |
| Lock Name | Locked Item | Lock Value |
| DelegationLock | CAS-Delegation-Offer |  |
| ITLock | CAS-IT-Delegated-Services-Offer | Read Only |
| DevLock | CAS-Dev-Delegated-Services-Offer | Read Only |

# Pulling it all together: The Service Catalogue

Now that we have reviewed the core capabilities and features that need to be configured to deliver services to your tenants, lets deploy a lab.

Thru this series, we have created a number of tables of information that represent the high-level design for this lab. Your content may and can differ to better match your requirements, but I will use the information we have put together to deliver and enterprise Azure Stack environment. Let's get started!!

This is your lab; feel free to change the names of users, subscriptions, quotas, offers, plans, etc. to meet your business needs. Simply change the info in the tables and update scripts with the new values!! You can also add and remove various items as you desire!

Configuration of this lab can be completed using the portal (quotas, plans, offers). Some customers may want to use PowerShell to do these tasks, so we have added scripts that will allow you to do this. For those that want to use the portal to do many of these steps, I have added links to the documentation on how to do each task.

1. **Install PowerShell**

As we will be using PowerShell to configure several items in the lab, we first need to configure the correct PowerShell version in the lab.

First run the command to obtain the PowerShell repositories that are registered to the current user.

Get-PSRepository

Machine generated alternative text:
PS 
Name 
PSGaI ery 
Get -PSRepos i tory 
Instal lationPoIicy 
Untrusted 
Sour ceLocat ion 
https : powershel I gal ery. com/api/v2/ 

Figure 38: Results of Get-PSRepository

Next, we want to install the correct version of Azure RM PowerShell. Additionally, we need to install the AzureStack module. If you want to scope the install to a specific user, you can optionally add -scope to the command.

i.e. -Scope CurrentUser

At the time of this writing the version for Azure Stack is 1.2.8. As we will install for all users, we need to run the command below from an elevated command prompt.

Install-Module -Name AzureRM -RequiredVersion 1.2.8

Install-Module -Name AzureStack

1. **Install PaaS Services**

Follow the instructions here to install [SQL](https://aka.ms/ASWP_SQLRP), [App Service](https://aka.ms/ASWP_AppSvrOver), and [MySQL](https://aka.ms/ASWP_MySQLRP) Resource Providers

1. **Configure Admins**

Create your delegated admins in Azure Active Directory using these [instructions](https://aka.ms/ASWP_AddUser).

|  |  |
| --- | --- |
| **Business Units** | **Administrator** |
| Information Technology | IT Admin |
| Development | Dev Admin |
| Human Resources | HR Admin |
| Finance | Fin Admin |

1. **Import the required Azure Stack modules**

Two modules are required for Azure Stack TP3, with a required version of 1.2.8

Azure RM

AzureStack

Import-Module -Name AzureRM -RequiredVersion 1.2.8 -ErrorAction SilentlyContinue -ErrorVariable ErrorMsg

if ($ErrorMsg)

{

Install-Module -Name AzureRM -RequiredVersion 1.2.8 -Force

}

Import-Module -Name AzureStack -RequiredVersion 1.2.8 -ErrorAction SilentlyContinue -ErrorVariable ErrorMsg

if ($ErrorMsg)

{

Install-Module -Name AzureStack -RequiredVersion 1.2.8 -Force

}

1. **Import Utilities and Templates**

Two of the utilities available for Azure Stack will be used during the lab build process. We will also import a local copy of the Azure Stack Templates for use in the lab once it is configured

invoke-webrequest https://github.com/Azure/AzureStack-Tools/archive/master.zip -OutFile "$env:TEMP\master.zip"

expand-archive "$env:TEMP\master.zip" -DestinationPath C:\ -Force

Remove-Item "$env:TEMP\master.zip"

invoke-webrequest "https://github.com/Azure/AzureStack-QuickStart-Templates/archive/master.zip" -OutFile "$env:TEMP\master.zip"

expand-archive "$env:TEMP\master.zip" -DestinationPath C:\ -Force

Remove-Item "$env:TEMP\master.zip"

In order for the script to run correctly, we need to import these modules

Import-Module C:\AzureStack-Tools-master\Connect\AzureStack.Connect.psm1

Import-Module C:\AzureStack-Tools-master\ComputeAdmin\AzureStack.ComputeAdmin.psm1

1. **Configure Delegation**

We will use the information below to configure delegation. Step by step instructions [here](https://aka.ms/ASWP_DP) will be used for the configuration. If you prefer to create the delegation plans and offers using the portal, the instructions are in the proceeding link.

You will need to configure one offer containing the subscription service for all the business units, then add your delegated admins to that one offer. This will simplify management.

NOTE: you will need to ensure you have created the delegation admins specific to your lab in Azure Active Directory to complete this task. To create these, we will use the commands in the table below

Let's look at how we can do this with PowerShell.

1. First let’s determine the Subscription ID you will require for many of the PowerShell commands. Look for <SubscriptionID> and insert your Subscription ID

Get-AzureRMContext

Machine generated alternative text:
PS Offers> get-azurermcontext 
Environment 
Account 
Subscr i pt i onld 
count 
• AzureStackCIoud 

Figure 39: Result of Get-AzureRMContext

Save the ID you will need it later

1. Run the following command to create the resource group to store delegation offer

New-AzureRmResourceGroup -Name CASDelegatedOffers -Location local

1. Next run the commands in the table below to create the delegated Offer/Plan

NOTE you must change the <SubscriptionID> that is highlighted to your SubscriptionID

1. Open the offer
   1. Click subscriptions and add the two members listed below
   2. Click Delegated providers and add the two members listed below (align to their subscriptions)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Offer Name** | **Plan Name** | **Services** | **PowerShell Command: Plan** | **PowerShell Command: Offer** | **Members** |
| Contoso-Delegation Offer | Contoso-Delegation Plan | Subscription | New-AzureRMPlan -Name IT-Del-Plan -DisplayName "Delegation-Plan" -ArmLocation "local" -ResourceGroup CASDelegatedOffers -QuotaIds "/subscriptions/<SubscriptionID>/providers/Microsoft.Subscriptions.Admin/locations/local/quotas/delegatedProviderQuota" | New-AzureRMOffer -name cas-delegation-offer -DisplayName CAS-Delegation-Offer -ARMLocation "local" -ResourceGroup "OffersandPlans" -BasePlanIds "/subscriptions/<SubscriptionID>/resourceGroups/CASDelegatedOffers/providers/Microsoft.Subscriptions.Admin/plans/Delegation-Plan" | IT Admin    Dev Admin |

I will show you shortly how I came up with the command for the plan command

Finally, we can customize the URL that each business unit will use to access the portal. Use the instructions in section Delegated provider customizes the offer [here](https://aka.ms/ASWP_DP).

Login as each delegated admin to configure the Provider settings => Portal Branding

|  |  |  |
| --- | --- | --- |
| Business Unit | URL | Custom URL |
| IT Dept | ContosoIT | https://publicportal.local.azurestack.external/?microsoft\_azure\_billing\_provider=ContosoIT |
| Dev Dept | ContosoDev | https://publicportal.local.azurestack.external/?microsoft\_azure\_billing\_provider=ContosoDev |

Tenants access using custom URL https://publicportal.local.azurestack.external/?microsoft\_azure\_billing\_provider=<URL>

1. **Configure Quotas**

Quotas will be defined using the information in the table below. If you prefer to use the portal to create quotas, plans, and offers, instructions are available [here](https://aka.ms/ASWP_Quotas). For the purpose of this series however let's explore how we can do this using PowerShell and dot sourcing.

1. First save the script below as a Quotas.ps1 file on your host machine. This script will use functions to expose information we collect via API, and allow us to automate the build of quotas. We will also use the same script to gather information we require to create our plans with PowerShell.
2. Edit the values that are highlighted to fit the requirements for the quota(s) you will create. I created custom files for each of the business units and the trial quotas. This allows me to save the configurations as backup, and easily reapply the settings as needed.

The scripts will configure custom values for storage, network, and compute. Within each section there are several values that can be changed within the script to set the values, for each of the parameters that are available, to those that you desire. The tables below outline the available parameters:

Storage:

|  |  |
| --- | --- |
| Parameter | Value in Script |
| CapacityinGb | 100 |
| NumberofStorageAccounts | 20 |

Network:

|  |  |
| --- | --- |
| Parameter | Value in Script |
| vNetsPerSubscription | 50 |
| gatewaysPerSubscription | 1 |
| connectionsPerSubscription | 2 |
| loadBalancersPerSubscription | 50 |
| nicsPerSubscription | 50 |
| securityGroupsPerSubscription | 50 |

Compute:

|  |  |
| --- | --- |
| Parameter | Value in Script |
| vmcount | 10 |
| memoryLimitMB | 10240 |
| Coreslimit | 10 |

1. Open PowerShell in administrator mode
2. [Download the script from GitHub](https://aka.ms/aswp_labscriptkit)
3. Run the following section of the script (#region Quota Functions) to configure api’s to access the quota values.
4. Finally run the PowerShell commands specific to quotas in the section (#region Quotas Plans and Offers) to create each quota

Each line of the command contains the values that will be applied to the specific quota. If you would like to change the values you can make the changes in the script.

|  |  |  |  |
| --- | --- | --- | --- |
| **Business Group** | **Quota Name** | **Quota Limits** | **PS Command** |
| HR | CAS-Compute  CAS-Network  CAS-Storage  CAS-AppSvc-Premium  CAS-MySQL-Premium | ~50% of defaults | New-CASComputeQuota -QuotaName CAS-Compute  New-CASNetworkQuota -QuotaName CAS-Network  New-CASStorageQuota -QuotaName CAS-Storage |
| Finance | CAS-Compute  CAS-Network  CAS-Storage  CAS-SQL-Premium | ~50% of defaults | Finance uses same quotas as HR |
| IT | CAS-IT-Compute  CAS-IT-Network  CAS-IT-Storage | ~100% of Defaults | New-ITComputeQuota -QuotaName CAS-IT-Compute  New-ITNetworkQuota -QuotaName CAS-IT-Network  New-ITStorageQuota -QuotaName CAS-IT-Storage |
| Dev | CAS-Dev-Compute  CAS-Dev-Network  CAS-Dev-Storage  CAS-AppSvc-Premium  CAS-MySQL-Premium  CAS-SQL-Premium  KeyVault | ~75% of Defaults | New-DEVComputeQuota -QuotaName CAS-Dev-Compute  New-DEVNetworkQuota -QuotaName CAS-Dev-Network  New-DEVStorageQuota -QuotaName CAS-Dev-Storage |
| Trial | Trial-Compute  Trial-Network  Trial-Storage | 4 VM  10GB Storage  4 X IP | New-TrialComputeQuota -QuotaName Trial-Compute  New-TrialNetworkQuota -QuotaName Trial-Network  New-TrialStorageQuota -QuotaName Trial-Storage |

That’s it, we have now created the quotas we require using PowerShell!

1. **Configure Services Plans and Offers**

Now we need to create the plans and offers we want. Again these can be built using the portal, using the guidance here: [offer](https://aka.ms/ASWP_Offer) and [plan](https://aka.ms/ASWP_Plan) if you prefer.

Just for fun let's look at how we can do this with PowerShell.

**Plans:**

First, we need to create our plans. We want them to use the appropriate Quotas that we created in the previous step.

The following is the PowerShell command for creating a plan: New-AzureRMPlan -Name $PlanName -DisplayName "ComputePlan" -ArmLocation "local" -ResourceGroup $ResourceGroupName -QuotaIds $quotaIds

At this time, there is not a PowerShell cmdlet that allows us to get the Quota Id's, so we will use the script we used to create the quotas to do this. When the script was created, not only did we include the "new" function for creating a quota, but we also configured the "get" function to return the information about all quotas in the Azure Stack environment.

1. If you have left the PowerShell window open that we used to create the Quotas, you can simply run the commands below. If not, then re-run the dot sourcing command we used above (. .\LabBuild.ps1) and then run the commands below. For each quota, you want to add to a plan, simply copy the ID from the results as we will need this in the next step.

Get-CASComputeQuota

Get-CASNetworkQuota

New-CASStorageQuota

Get-ITComputeQuota

Get-ITNetworkQuota

Get-ITStorageQuota

Get-DEVComputeQuota

Get-DEVNetworkQuota

Get-DEVStorageQuota

Get-TrialComputeQuota

Get-TrialNetworkQuota

Get-TrialStorageQuota

1. In this step, we use the information we have gathered in the previous step to write the command to create our plans. Each line in the script to create the plans applies the required quotas

NOTE: for the Trial, HR and Finance plans I am going to add -Status Public to the command as I want these to be set to public. If you don’t want your plans to be publicly visible simple remove this from the command.

First let’s create out resource group

New-AzureRmResourceGroup -Name OffersandPlans -Location "local"

The example below is for our HR Plan

$CasHrPlan = New-AzureRmPlan -Name cas-hr-plan -DisplayName CAS-HR-Plan -ArmLocation $ArmLocation -ResourceGroup $OfferRG -QuotaIds @($CASComputeQuota.id,$CASNetworkQuota.id,$CASStorageQuota.id)

1. Now that you have one plan built let's rinse and repeat for the rest of our plans using the information in the table below!

**Offers**

Almost there! Again, we can use the portal to create our offers using the documentation [here](https://aka.ms/ASWP_Offer), but let's stick with doing this with PowerShell. We require the base plan IDs for this command.

Get-AzureRMPlan -Managed -ResourceGroup OffersandPlans

1. Copy the base plan Id to the command below and the update other parameters.
2. Run the command

New-AzureRMOffer -name <offer name - small letters> -DisplayName <display Name> -ARMLocation "local" -ResourceGroup "< resource group>" -BasePlanIds "<Base Plan ID>"

Example:

New-AzureRMOffer -name CAS-HR-offer -DisplayName CAS-HR-Offer -State Public -ARMLocation $ArmLocation -ResourceGroup $OfferRG -BasePlanIds $CasHrPlan.Id

1. Rinse and repeat for the balance of the offers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Offers** | **Advertised/Assigned** | **Assigned Tenants** | **Plans** | Base Services |
| CAS-HR-Offer | Advertised | None | CAS-HR-Plan | Compute  Network  Storage  App Service  My SQL |
| CAS-Finance-Offer | Advertised | None | CAS-FIN-Plan | Compute  Network  Storage  SQL |
| CAS-IT-Offer | Assigned | IT1  IT2 | CAS-IT-Plan | Compute  Network  Storage |
| CAS-Dev-Offer | Assigned | Dev1  Dev2 | CAS-Dev-Plan | Compute  Network  Storage  App Service  My SQL  SQL  Key Vault |
| Trial-Offer | Advertised | None | CAS-Trial-Plan | Compute  Network  Storage |

**Last Steps**

Once these are created we can assign each delegated admin to their specific offer. Instructions on how to complete this step are as follows:

* In the portal, navigate to each service offer to be delegated (IT / Dev) and locate "subscriptions"
  + Add the appropriate administrator, assigning the subscription to them
* Next and locate "delegated provider"
  + Provide a Name for the delegated offer
  + Add the appropriate delegated admin

|  |
| --- |
| Delegation Offers Names |
| CAS-IT-Delegated-Services-Offer |
| CAS-Dev-Delegated-Services-Offer |

1. **Marketplace Items**

We can add marketplace items in two ways; using the documentation on the Microsoft Azure Stack Documentation site, or using the tool available in the Azure Stack Tools on GitHub. There is a third method coming in TP3 - [Marketplace Syndication](https://aka.ms/aswp_marketplacesynd). We will add an Ubuntu image using the tool in [GitHub](https://aka.ms/ASWP_CA).

The instructions for this task are located [here](https://aka.ms/ASWP_Linux).

1. **Configure Tags**

Let's now configure a couple of tags in the lab. While these are not reflective of tags you may use in production, they are an example of how to complete the task. First let's look at the default tags created in Azure Stack.

1. In the Azure Stack portal click on "More Services" => Tags
2. In PowerShell run the following command:

Get-AzureRmTag -Detailed

S get-azurermtag 
: displayName 
ount 
alues 
Name 
Controllers Availability Set 
Front-Ends Availability Set 
-Detai -I ed 
Management Servers Availability Set 
Controllers Virtual Machines 
Front-Ends Virtual Machines 
Standalone file server Virtual Machines 
Publishers Virtual Machines 
Management Servers Virtual Machines 
Shared Web Workers Virtual Machines 
Front-Ends External Load Sal ancer 
Management External Load Balancer 
Control I ers Network Interface 
Front-Ends Network Interface 
Standalone file server Network Interface 
Publisher Servers Network Interface 
Management Servers Network Interface 
Shared Web Workers Network Interface 
Controller Direct IP Endpoint 
Front-Ends Load Sal anced Endpoint 
Publisher Direct IP Endpoint 
Management Servers Load Balanced Endpoint 
Vi rtuaINetwork 
VirtualMachine 
Networklnterface 
Publ i clPAddr ess 
Stor ageAccount 
Count 
1 

Figure 40: Results of Get-AzureRMTag -Detailed

We will create two tags, one with the portal and one with PowerShell. Tags use a name/value pair and are applied to resources within Azure Stack.

|  |  |
| --- | --- |
| **Display Name** | **Value** |
| AppSrv- all VMs | AppSvc |
| VM Operating System | Windows Server 2016 |

**Portal:**

1. In the portal navigate to Virtual Machines=> any one of the AppSvc Virtual Machines => Tags
2. Add the name and value as documented in the table below
3. Click save

The tag has now been applied. You can continue to add tags to other VM's that make up the AppSvc infrastructure using this method.

Machine generated alternative text:
Microsoft Azure Stack 
Virtual machines 
Virtual machines 
> 
FEO-VM - Tags 
FEO-VM 
Tags 
Columns Refresh 
Subscriptions: Default Provider Subscription 
Filter 
CNO-VM 
FEO-VM 
FS-VM 
p Search resources 
Tags are key/value pairs that enable you to categorize resources and view 
O 
consolidated billing by applying the same tag to multiple resources and resource 
Search (CtH+D 
Ove rview 
Activ ty' log 
Access control (IAM) 
Tags 
groups. Learn more 
Key 
AppSvc-A11 VMS 
displayName : Front-Ends Virtual Machines 
* Value 
• AppSvc V Ml 

Figure 41: Tag Properties

**PowerShell**

1. Open PowerShell in Administrator mode
2. Type the following command

New-AzureRMTag -Name "VM Operating System" -Value "Windows Server 2016"

PS New-AzureRNTag 
- Name 
"VM Operating Systen;" 
-Value 
"'Si nc10'.vs Server 2016" 
Name 
ount 
alues 
. W Operating System 
Name 
Wi ndows 
Server 
2016 
Count 
o 

Figure 42: Results of New-AzureRMTag

To view all available tags type:

Get-AzureRMTag -Detailed

Front-Ends Load Sal anced Endpoint 
Publisher Direct IP Endpoint 
Management Servers Load Balanced Endpoint 
1 
Vi rtuaINetwork 
VirtualMachine 
Networklnterface 
Publ i clPAddr ess 
Stor ageAccount 
: AppSvc-AII VMS 
Name 
ount 
alues • 
Name 
ount 
alues . 
Name 
Count 
AppSvc W I 
. W Operating System 
Name 
Wi n dows 
Server 
2016 
Count 
o 

Figure43: Results of Get-AzureRMTag -Detailed

Now that you have created the tags, you can either apply them to resources using the portal, or add the following to a variable in a JSON template and have them applied automatically when deploying resources.

"tags": {

"displayName": "<display name>"

},

Example:

{

"name": "[parameters('mystoaccountName')]",

"type": "Microsoft.Storage/storageAccounts",

"location": "[parameters('mystoaccountLocation')]",

"apiVersion": "2015-05-01-preview",

"dependsOn": [ ],

"tags": {

"displayName": "mystoaccount"

},

"properties": {

"accountType": "[parameters('mystoaccountType')]"

}

}

1. **Configure Locks**

At this time locks can only be applied using the portal. Apply locks to each of the Delegation Offers using the table below. To create a lock

1. In the portal navigate to the offer
2. Select locks
3. Add a lock

|  |  |  |
| --- | --- | --- |
| Lock Name | Locked Item | Lock Value |
| DelegationLock | CAS-Delegation-Offer |  |
| ITLock | CAS-IT-Delegated-Services-Offer | Read Only |
| DevLock | CAS-Dev-Delegated-Services-Offer | Read Only |

**WOW WE ARE DONE!!!!!**

Congratulations!!! That was a lot of work, BUT you should now have an environment that looks similar to the diagram below!!!!

pawlbay d suouduasms 
ueld leu 
Jago 
euaL 
55au!5n€ 
EIN ä] 
ueld ul.d 
a•ual 
ue1d8H 
ueld 
Jam 
ueld LI 
u on pal Salaa 
aunosay 
5 a onO 
51340 
EÆE 
peas a.nzv osowo. 

Figure 44: Final Results - The lab

# Lab Script Kit

Once you have deployed Azure Stack, log in as Service admin. You will need to install the resource providers using the instructions in the Pulling It All Together post.

**IMPORTANT: for the scripts to work correctly you MUST be running the correct version of PowerShell AzureRM**

Once that is done we can start the configuration of the lab. These scripts will do the following:

* Install the correct version of Azure Stack PowerShell
* Install the Azure Stack utilities for connecting to Azure Stack
* Create plans, offers, and quotas with specific quotas aligned to offers and plans
* Set specific offers to public state
* Configure delegation plan and offer

NOTE: you must add your delegated admin to the delegation subscription and the services subscription to be assigned to this delegated admin as defined in the instructions in the Pulling It All Together post

* Create a custom tag
* Adds an eval version of Server 2016 to PIR
* Adds an Ubuntu image to the PIR

[Download Lab Script Kit](https://aka.ms/aswp_labscriptkit)

The deployment requires several scripts. There are two that run the deployment that are contained in the C:\Build Lab folder. There are also several scripts that define each of the quotas in the C:\Build Lab\Scripts\Create Quotas folder.

The folder structure is as follows:

C:\Build Lab\Scripts\Create Quotas

**Instructions**

1. Copy either the Enterprise or Service provider script from GitHub and save to your machine in the C:\Build Lab folder and save as a ps1 file. It does not matter what you name it
2. Ensure the additional scripts are in the C:\Build Lab\Scripts\Create Quotas folder and that the names of each exactly match the list below
3. Once all files are saved in the correct folders, open PowerShell as administrator
4. Run the following command to get your subscription id

Get-AzureRMContext

1. In the Enterprise / Service provider lab file, change following values for the variables located at the top. Save the PS1 file
2. Run the Enterprise/Service provider script to install the lab

NOTE: you will be asked to provide service admin credentials several times during the build of the lab.

These scripts will build the lab (either in an Enterprise or Service Provider version), setting up the basics for you. The Service Provider script create three offers (Premium, Standard, Trial) and sets them to public status. It also sets up delegation, however you will need to complete the final steps of configuring the administrator for delegation (use the documentation here starting at step 2 under "Service admin creates the delegated offer"

The Enterprise script creates four offers aligned to the business groups HR, Finance, IT, and Development. HR and Finance are set to public, while the other two are left private. These will be use the delegation model. As with the service provider model you will need to configure your administrators for delegation using the same instructions listed in the paragraph above.

Keep in mind you can easily change several things to better fit your lab modeling. These scripts are designed to be configurable. Instructions in the Pulling It All Together section outline how to make custom changes.

Some of the things you can change are:

* Business Group names
* Quotas values
* Quota, plan, and offer names
* ….

**Final Thoughts:**

Thanks for sticking with us through this series, and we hope it has provided good insights into the development of your Microsoft Azure Stack TP lab. We also hope that it has led to more questions and thoughts on how to best provide the best possible solution for your organization!

Now that you have deployed Azure Stack and completed basic configuration, it's time to start testing your solution that you want to deploy to Azure Stack. I hope these scripts not only simplify the lab build for you, but also show you how to break the various tasks apart so you can run them individually.

We will continue to add to the content as Azure Stack continues to evolve, so please continue to check the [Azure Stack Documentation](https://docs.microsoft.com/azure/azure-stack), [GitHub](https://aka.ms/ASWP_Tools), videos on [Channel 9](https://aka.ms/ASWP_Ch9Blog) and the location you obtained this paper from for new information!

Additional things you can explore:

[Configure Key Vault](https://aka.ms/ASWP_KVIntro)

[Testing the billing integration capabilities](https://aka.ms/ASWP_Billing)

Test the various tools for deployment of JSON templates ([PowerShell](https://aka.ms/ASWP_DeployPS), [CLI](https://aka.ms/ASWP_DeployCLI),

[Visual Studio](https://aka.ms/ASWP_VS))

[Develop Applications for Azure Stack](https://aka.ms/ASWP_Dev)