Contents

Analysis Services team blog

Azure updates

Azure status

Feedback

Pricing

Pricing calculator

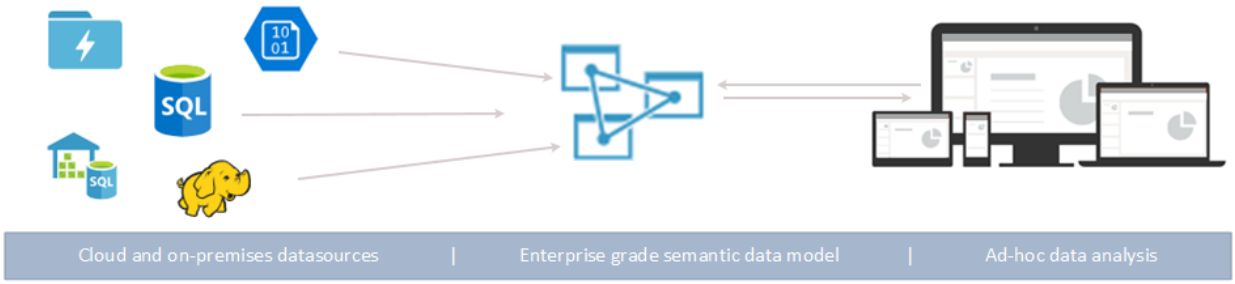
Stack Overflow

Videos

What is Azure Analysis Services?

3/29/2021 • 13 minutes to read • Edit Online

Azure Analysis Services is a fully managed platform as a service (PaaS) that provides enterprise-grade data models in the cloud. Use advanced mashup and modeling features to combine data from multiple data sources, define metrics, and secure your data in a single, trusted tabular semantic data model. The data model provides an easier and faster way for users to perform ad hoc data analysis using tools like Power BI and Excel.



Video: Check out Azure Analysis Services Overview to learn how Azure Analysis Services fits in with Microsoft's overall BI capabilities.

Get up and running quickly

In Azure portal, you can create a server within minutes. And with Azure Resource Manager templates and PowerShell, you can create servers using a declarative template. With a single template, you can deploy server resources along with other Azure components such as storage accounts and Azure Functions.

Video: Check out Automating deployment to learn more about how you can use Azure Automation to speed server creation.

Azure Analysis Services integrates with many Azure services enabling you to build sophisticated analytics solutions. Integration with Azure Active Directory provides secure, role-based access to your critical data. Integrate with Azure Data Factory pipelines by including an activity that loads data into the model. Azure Automation and Azure Functions can be used for lightweight orchestration of models using custom code.

The right tier when you need it

Azure Analysis Services is available in **Developer**, **Basic**, and **Standard** tiers. Within each tier, plan costs vary according to processing power, Query Processing Units (QPUs), and memory size. When you create a server, you select a plan within a tier. You can change plans up or down within the same tier, or upgrade to a higher tier, but you can't downgrade from a higher tier to a lower tier.

Developer tier

This tier is recommended for evaluation, development, and test scenarios. A single plan includes the same functionality of the standard tier, but is limited in processing power, QPUs, and memory size. Query replica scale-out *is not available* for this tier. This tier does not offer an SLA.

PLAN	QPUS	MEMORY (GB)
D1	20	3

Basic tier

This tier is recommended for production solutions with smaller tabular models, limited user concurrency, and simple data refresh requirements. Query replica scale-out *is not available* for this tier. Perspectives, multiple partitions, and DirectQuery tabular model features *are not supported* in this tier.

PLAN	QPUS	MEMORY (GB)
B1	40	10
B2	80	16

Standard tier

This tier is for mission-critical production applications that require elastic user-concurrency, and have rapidly growing data models. It supports advanced data refresh for near real-time data model updates, and supports all tabular modeling features.

PLAN	QPUS	MEMORY (GB)
S0	40	10
S1	100	25
S2	200	50
S4	400	100
S8 [1], [2]	320	200
S9 [1], [2]	640	400
S8v2 [1]	640	200
S9v2 [1]	1280	400

1 - Not available in all regions.
2 - S8 and S9 are deprecated. v2 is recommended.

Availability by region

Azure Analysis Services is supported in regions throughout the world. Supported plans and query replica availability depend on the region you choose. Plan and query replica availability can change depending on need and available resources for each region.

Americas

REGION	SUPPORTED PLANS	QUERY REPLICAS (STANDARD PLANS ONLY)
Brazil South	B1, B2, S0, S1, S2, S4, D1	1

REGION	SUPPORTED PLANS	QUERY REPLICAS (STANDARD PLANS ONLY)
Canada Central	B1, B2, S0, S1, S2, S4, D1	1
Canada Central	S8v2, S9v2	1
East US	B1, B2, S0, S1, S2, S4, D1	1
East US	S8v2, S9v2	1
East US 2	B1, B2, S0, S1, S2, S4, D1	7
East US 2	S8v2, S9v2	1
North Central US	B1, B2, S0, S1, S2, S4, D1	1
North Central US	S8v2, S9v2	1
Central US	B1, B2, S0, S1, S2, S4, D1	1
Central US	S8v2, S9v2	1
South Central US	B1, B2, S0, S1, S2, S4, D1	1
South Central US	S8v2, S9v2	1
West Central US	B1, B2, S0, S1, S2, S4, D1	3
West US	B1, B2, S0, S1, S2, S4, D1	7
West US	S8v2, S9v2	2
West US2	B1, B2, S0, S1, S2, S4, D1	3
West US2	S8v2, S9v2	1

Europe

REGION	SUPPORTED PLANS	QUERY REPLICAS (STANDARD PLANS ONLY)
North Europe	B1, B2, S0, S1, S2, S4, D1	7
North Europe	S8v2, S9v2	3
UK South	B1, B2, S0, S1, S2, S4, D1	1
West Europe	B1, B2, S0, S1, S2, S4, D1	7
West Europe	S8v2, S9v2	1

Asia Pacific

REGION	SUPPORTED PLANS	QUERY REPLICAS (STANDARD PLANS ONLY)
Australia East	B1, B2, S0, S1, S2, S4	3
Australia East	S8v2, S9v2	1
Australia Southeast	B1, B2, S0, S1, S2, S4, D1	1
Japan East	B1, B2, S0, S1, S2, S4, D1	1
Japan East	S8v2, S9v2	1
Southeast Asia	B1, B2, S0, S1, S2, S4, D1	1
Southeast Asia	S8v2, S9v2	1
West India	B1, B2, S0, S1, S2, S4, D1	1

Scale to your needs

Scale up\down, pause, and resume

Go up, down, or pause your server. Use the Azure portal or have total control on-the-fly by using PowerShell. You only pay for what you use.

Scale out resources for fast query responses

With scale out, client queries are distributed among multiple *query replicas* in a query pool. Query replicas have synchronized copies of your tabular models. By spreading the query workload, response times during high query workloads can be reduced. Model processing operations can be separated from the query pool, ensuring client queries are not adversely affected by processing operations.

You can create a query pool with up to seven additional query replicas (eight total, including your server). The number of query replicas you can have in your pool depend on your chosen plan and region. Query replicas cannot be spread outside your server's region. Query replicas are billed at the same rate as your server.

Just like with changing tiers, you can scale out query replicas according to your needs. Configure scale out in the portal or by using REST APIs. To learn more, see Azure Analysis Services scale out.

Pricing

Total cost depends on a number of factors. For example, your chosen region, tier, query replicas, and pause/resume. Use the Azure Analysis Services Pricing calculator to determine typical pricing for your region. This tool calculates pricing for a single-server instance for a single region. Keep in mind, query replicas are billed at the same rate as the server.

Built on SQL Server Analysis Services

Azure Analysis Services is compatible with many great features already in SQL Server Analysis Services Enterprise Edition. Azure Analysis Services supports tabular models at the 1200 and higher compatibility levels. Tabular models are relational modeling constructs (model, tables, columns), articulated in tabular metadata object definitions in Tabular Model Scripting Language (TMSL) and Tabular Object Model (TOM) code. Partitions, perspectives, row-level security, bi-directional relationships, and translations are all supported*. Multidimensional models and PowerPivot for SharePoint *are not* supported in Azure Analysis Services.

Tabular models in both in-memory and DirectQuery modes are supported. In-memory mode (default) tabular models support multiple data sources. Because model data is highly compressed and cached in-memory, this mode provides the fastest query response over large amounts of data. It also provides the greatest flexibility for complex datasets and queries.

Partitioning enables incremental loads, increases parallelization, and reduces memory consumption. Other advanced data modeling features like calculated tables, and all DAX functions are supported. In-memory models must be refreshed (processed) to update cached data from data sources. With Azure service principal support, unattended refresh operations using PowerShell, TOM, TMSL, and REST offer flexibility in making sure your model data is always up to date.

DirectQuery mode* leverages the backend relational database for storage and query execution. Extremely large data sets in single SQL Server, SQL Server Data Warehouse, Azure SQL Database, Azure Synapse Analytics, Oracle, and Teradata data sources are supported. Backend data sets can exceed available server resource memory. Complex data model refresh scenarios aren't needed. There are also some restrictions, such as limited data source types, DAX formula limitations, and some advanced data modeling features aren't supported. Before determining the best mode for you, see Direct Query mode.

* Feature availability depends on tier.

Supported data sources

Tabular models in Azure Analysis Services support a wide variety of data sources from simple text files to Big Data in Azure Data Lake Store. To learn more, see Data sources supported in Azure Analysis Services.

Compatibility level

Compatibility level refers to release-specific behaviors in the Analysis Services engine. Azure Analysis Services supports tabular models at the 1200 and higher compatibility levels. To learn more, see Compatibility level for tabular models.

Your data is secure

Azure Analysis Services provides security for your sensitive data at multiple levels. As an Azure service, Analysis Services provides the **Basic** level protection of Distributed denial of service (DDoS) attacks automatically enabled as part of the Azure platform. To learn more, see Azure DDoS Protection Standard overview.

At the server level, Analysis Services provides firewall, Azure authentication, server administrator roles, and Server-Side Encryption. At the data model level, user roles, row-level, and object-level security ensure your data is safe and gets seen by only those users who are meant to see it.

Firewall

Azure Analysis Services Firewall blocks all client connections other than those IP addresses specified in rules. By default, firewall protection is not enabled for new servers. It's recommended firewall protection is enabled and rules are configured as part of a server provisioning script or in the portal immediately after the server is created. Configure rules specifying allowed IP addresses by individual client IPs or by range. Power BI (service) connections can also be allowed or blocked. Configure firewall and rules in the portal or by using PowerShell. To learn more, see Configure a server firewall.

Authentication

User authentication is handled by Azure Active Directory (AAD). When logging in, users use an organization account identity with role-based access to the database. User identities must be members of the default Azure Active Directory for the subscription that the server is in. To learn more, see Authentication and user permissions.

Data security

Azure Analysis Services uses Azure Blob storage to persist storage and metadata for Analysis Services databases. Data files within Blob are encrypted using Azure Blob Server Side Encryption (SSE). When using Direct Query mode, only metadata is stored. The actual data is accessed through encrypted protocol from the data source at query time.

Secure access to data sources on-premises in your organization is achieved by installing and configuring an On-premises data gateway. Gateways provide access to data for both DirectQuery and in-memory modes.

Roles

Analysis Services uses role-based authorization that grants access to server and model database operations, objects, and data. All users who access a server or database do so with their Azure AD user account within an assigned role. The server administrator role is at the server resource level. By default, the account used when creating a server is automatically included in the Server Admins role. Additional user and group accounts are added by using the portal, SSMS, or PowerShell.

Non-administrative end users who query data are granted access through database roles. A database role is created as a separate object in the database, and applies only to the database in which that role is created. Database roles are defined by (database) Administrator, Read, and Read and Process permissions. User and group accounts are added by using SSMS or PowerShell.

Row-level security

Tabular models at all compatibility levels support row-level security. Row-level security is configured in the model by using DAX expressions that define the rows in a table, and any rows in the many direction of a related table that a user can query. Row filters using DAX expressions are defined for the Read and Read and Process permissions.

Object-level security

Tabular models at the 1400 compatibility level support object-level security, which includes table-level security and column-level security. Object level security is set in the JSON-based metadata in the Model.bim file by using TMSL, or TOM. To learn more, see Object-level security.

Automation through service principals

Service principals are an Azure Active Directory application resource you create within your tenant to perform unattended resource and service level operations. Service principals are used with Azure Automation, PowerShell unattended mode, custom client applications, and web apps to automate common tasks like data refresh, scale up/down, and pause/resume. Permissions are assigned to service principals through role membership. To learn more, see Automation with service principals.

Azure governance

Azure Analysis Services is governed by the Microsoft Online Services Terms and the Microsoft Privacy Statement. To learn more about Azure Security, see the Microsoft Trust Center.

Use the tools you already know

Visual Studio

Develop and deploy models with Visual Studio with Analysis Services projects. The Analysis Services projects extension includes templates and wizards that get you up and going quickly. The model authoring environment in Visual Studio now includes the modern Get Data data source query and mashup functionality for tabular

1400 and higher models. If you're familiar with Get Data in Power BI Desktop and Excel 2016, you already know how easy it is to create highly customized data source queries.

Microsoft Analysis Services Projects is available as a free installable VSIX package. Download from Marketplace. The extension works with any version of Visual Studio 2017 and later, including the free Community edition.

SQL Server Management Studio

Manage your servers and model databases by using SQL Server Management Studio (SSMS). Connect to your servers in the cloud. Run TMSL scripts right from the XMLA query window, and automate tasks by using TMSL scripts and PowerShell. New features and functionality happen fast - SSMS is updated monthly.

Open-source tools

Analysis Services has a vibrant community of developers who create tools. Be sure to check out Tabular Editor, an open-source tool for creating, maintaining, and managing tabular models using an intuitive, lightweight editor. DAX Studio, is a great open-source tool for DAX authoring, diagnosis, performance tuning, and analysis.

PowerShell

Server resource management tasks like creating server resources, suspending or resuming server operations, or changing the service level (tier) use Azure PowerShell cmdlets. Other tasks for managing databases such as adding or removing role members, processing, or running TMSL scripts use cmdlets in the SqlServer module. To learn more, see Manage Azure Analysis Services with PowerShell.

Object model and scripting

Tabular models offer rapid development and are highly customizable. Tabular models include the Tabular Object Model (TOM) to describe model objects. TOM is exposed in JSON through the Tabular Model Scripting Language (TMSL) and the AMO data definition language through the Microsoft.AnalysisServices.Tabular namespace.

Supports the latest client tools

Modern data exploration and visualization tools like Power BI, Excel, Reporting Services, and third-party tools are all supported, providing users with highly interactive and visually rich insights into your model data.

Monitoring and diagnostics

Azure Analysis Services is integrated with Azure Monitor metrics, providing an extensive number of resource-specific metrics to help you monitor the performance and health of your servers. To learn more, see Monitor server metrics. Record metrics with resource platform logs. Monitor and send logs to Azure Storage, stream them to Azure Event Hubs, and export them to Azure Monitor logs, a service of Azure. To learn more, see Setup diagnostic logging.

Azure Analysis Services also supports using Dynamic Management Views (DMVs). Based on SQL syntax, DMVs interface schema rowsets that return metadata and monitoring information about server instance.

Documentation

Documentation specific to Azure Analysis Services is included here. Use the table of contents on the left side of your browser screen to find articles.

Because tabular models in Azure Analysis Services are much the same as tabular models in SQL Server Analysis Services and Power BI Premium datasets, there's an extensive library of shared data modeling tutorials,

conceptual, procedural, developer, and reference articles in Analysis Services documentation. Articles in the shared Analysis Services documentation show if they also apply to Azure Analysis Services by an APPLIES TO banner beneath the title. You can also use the Version selector above the table of contents to see only those articles that apply to the platform you're using.

Contribute!

Analysis Services documentation, like this article, is open source. To learn more about how you can contribute, see the Docs contributor guide.

Azure Analysis Services documentation also uses GitHub Issues. You can provide feedback about the product or documentation. Use **Feedback** at the bottom of an article. GitHub Issues are not enabled for the shared Analysis Services documentation.

Blogs

Things are changing rapidly. Get the latest information on the Power BI blog and Azure blog.

Community

Analysis Services has a vibrant community of users. Join the conversation on Azure Analysis Services forum.

Next steps

Sign up for a Free Azure Trial

Quickstart: Create a server - Portal

Quickstart: Create a server - PowerShell

Quickstart: Create a server - Portal

4/15/2021 • 2 minutes to read • Edit Online

This quickstart describes how to create an Analysis Services server resource in your Azure subscription by using the portal.

Prerequisites

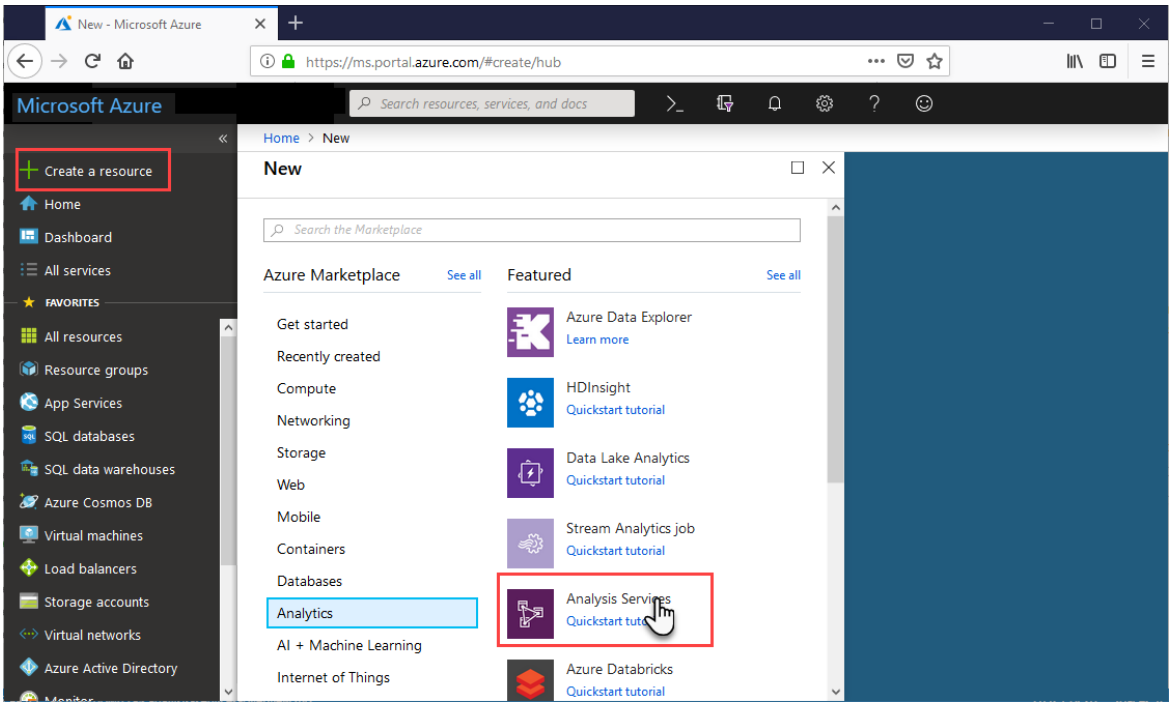
- **Azure subscription**: Visit Azure Free Trial to create an account.
- **Azure Active Directory**: Your subscription must be associated with an Azure Active Directory tenant. And, you need to be signed in to Azure with an account in that Azure Active Directory. To learn more, see Authentication and user permissions.

Sign in to the Azure portal

Sign in to the portal

Create a server

1. Click **+ Create a resource** > **Analytics** > **Analysis Services**.



2. In **Analysis Services**, fill in the required fields, and then press **Create**.

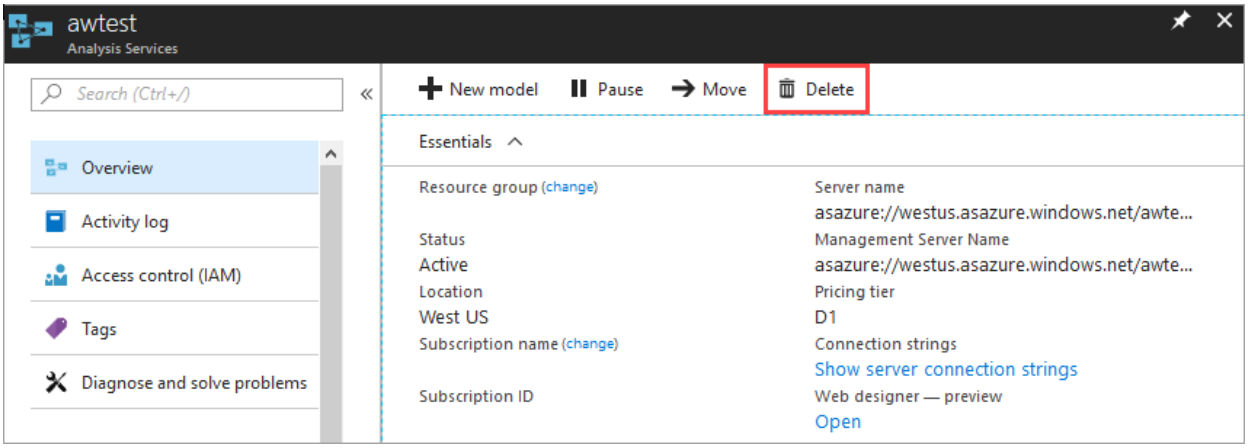
   - **Server name**: Type a unique name used to reference the server. The server name must begin with a lowercase character and contain between 3 to 128 lowercase characters and numbers. Whitespaces and special characters are not allowed.
   - **Subscription**: Select the subscription this server will be associated with.
   - **Resource group**: Create a new resource group or select one you already have. Resource groups are designed to help you manage a collection of Azure resources. To learn more, see resource groups.
   - **Location**: This Azure datacenter location hosts the server. Choose a location nearest your largest user base.

- **Pricing tier**: Select a pricing tier. If you are testing and intend to install the sample model database, select the free **D1** tier. To learn more, see Azure Analysis Services pricing.
- **Administrator**: By default, this will be the account you are logged in with. You can choose a different account from your Azure Active Directory.
- **Backup Storage setting**: Optional. If you already have a storage account, you can specify it as the default for model database backup. You can also specify backup and restore settings later.
- **Storage key expiration**: Optional. Specify a storage key expiration period.

Creating the server usually takes under a minute. If you selected **Add to Portal**, navigate to your portal to see your new server. Or, navigate to **All services** > **Analysis Services** to see if your server is ready. Servers support tabular models at the 1200 and higher compatibility levels. Model compatibility level is specified in Visual Studio or SSMS.

Clean up resources

When no longer needed, delete your server. In your server's **Overview**, click **Delete**.



Next steps

In this quickstart, you learned how to create a server in your Azure subscription. Now that you have server, you can help secure it by configuring an (optional) server firewall. You can also add a basic sample data model to your server right from the portal. Having a sample model is helpful when learning about configuring model database roles and testing client connections. To learn more, continue to the tutorial for adding a sample model.

Quickstart: Configure server firewall - Portal

Quickstart: Create a server - PowerShell

4/21/2021 • 2 minutes to read • Edit Online

This quickstart describes using PowerShell from the command line to create an Azure Analysis Services server in your Azure subscription.

Prerequisites

> **NOTE**
> This article has been updated to use the Azure Az PowerShell module. The Az PowerShell module is the recommended PowerShell module for interacting with Azure. To get started with the Az PowerShell module, see Install Azure PowerShell. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

- **Azure subscription**: Visit Azure Free Trial to create an account.
- **Azure Active Directory**: Your subscription must be associated with an Azure Active Directory tenant and you must have an account in that directory. To learn more, see Authentication and user permissions.
- **Azure PowerShell**. To find the installed version, run `Get-Module -ListAvailable Az`. To install or upgrade, see Install Azure PowerShell module.

Import Az.AnalysisServices module

To create a server in your subscription, you use the Az.AnalysisServices module. Load the Az.AnalysisServices module into your PowerShell session.

```
Import-Module Az.AnalysisServices
```

Sign in to Azure

Sign in to your Azure subscription by using the Connect-AzAccount command. Follow the on-screen directions.

```
Connect-AzAccount
```

Create a resource group

An Azure resource group is a logical container where Azure resources are deployed and managed as a group. When you create your server, you must specify a resource group in your subscription. If you do not already have a resource group, you can create a new one by using the New-AzResourceGroup command. The following example creates a resource group named `myResourceGroup` in the West US region.

```
New-AzResourceGroup -Name "myResourceGroup" -Location "WestUS"
```

Create a server

Create a new server by using the New-AzAnalysisServicesServer command. The following example creates a server named myServer in myResourceGroup, in the WestUS region, at the D1 (free) tier, and specifies

philipc@adventureworks.com as a server administrator.

```
New-AzAnalysisServicesServer -ResourceGroupName "myResourceGroup" -Name "myserver" -Location WestUS -Sku D1
-Administrator "philipc@adventure-works.com"
```

Clean up resources

You can remove the server from your subscription by using the Remove-AzAnalysisServicesServer command. If you continue with other quickstarts and tutorials in this collection, do not remove your server. The following example removes the server created in the previous step.

```
Remove-AzAnalysisServicesServer -Name "myserver" -ResourceGroupName "myResourceGroup"
```

Next steps

In this quickstart, you learned how to create a server in your Azure subscription by using PowerShell. Now that you have server, you can help secure it by configuring an (optional) server firewall. You can also add a basic sample data model to your server right from the portal. Having a sample model is helpful when learning about configuring model database roles and testing client connections. To learn more, continue to the tutorial for adding a sample model.

Quickstart: Configure server firewall - Portal

Quickstart: Create a server - ARM template

6/9/2021 • 4 minutes to read • Edit Online

This quickstart describes how to create an Analysis Services server resource in your Azure subscription by using an Azure Resource Manager template (ARM template).

An ARM template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

If your environment meets the prerequisites and you're familiar with using ARM templates, select the **Deploy to Azure** button. The template will open in the Azure portal.

Prerequisites

- **Azure subscription**: Visit Azure Free Trial to create an account.
- **Azure Active Directory**: Your subscription must be associated with an Azure Active Directory tenant. And, you need to be signed in to Azure with an account in that Azure Active Directory. To learn more, see Authentication and user permissions.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

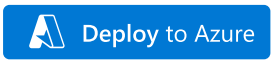
```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "serverName": {
      "type": "string",
      "metadata": {
        "description": "The name of the Azure Analysis Services server to create. Server name must begin
with a letter, be lowercase alphanumeric, and between 3 and 63 characters in length. Server name must be
unique per region."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location of the Azure Analysis Services server. For supported regions, see
https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-overview#availability-by-region"
      }
    },
    "skuName": {
      "type": "string",
      "defaultValue": "S0",
      "metadata": {
        "description": "The sku name of the Azure Analysis Services server to create. Choose from: B1, B2,
D1, S0, S1, S2, S3, S4, S8, S9. Some skus are region specific. See https://docs.microsoft.com/en-
us/azure/analysis-services/analysis-services-overview#availability-by-region"
      }
    },
    "capacity": {
      "type": "int",
```

```
      "defaultValue": 1,
      "metadata": {
        "description": "The total number of query replica scale-out instances. Scale-out of more than one
instance is supported on selected regions only. See https://docs.microsoft.com/en-us/azure/analysis-
services/analysis-services-overview#availability-by-region"
      }
    },
    "firewallSettings": {
      "type": "object",
      "defaultValue": {
        "firewallRules": [
          {
            "firewallRuleName": "AllowFromAll",
            "rangeStart": "0.0.0.0",
            "rangeEnd": "255.255.255.255"
          }
        ],
        "enablePowerBIService": true
      },
      "metadata": {
        "description": "The inbound firewall rules to define on the server. If not specified, firewall is
disabled."
      }
    },
    "backupBlobContainerUri": {
      "type": "string",
      "defaultValue": "",
      "metadata": {
        "description": "The SAS URI to a private Azure Blob Storage container with read, write and list
permissions. Required only if you intend to use the backup/restore functionality. See
https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-backup"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.AnalysisServices/servers",
      "apiVersion": "2017-08-01",
      "name": "[parameters('serverName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[parameters('skuName')]",
        "capacity": "[parameters('capacity')]"
      },
      "properties": {
        "ipV4FirewallSettings": "[parameters('firewallSettings')]",
        "backupBlobContainerUri": "[parameters('backupBlobContainerUri')]"
      }
    }
  ]
}
```

A single Microsoft.AnalysisServices/servers resource with a firewall rule is defined in the template.

Deploy the template

1. Select the following Deploy to Azure link to sign in to Azure and open a template. The template is used to create an Analysis Services server resource and specify required and optional properties.

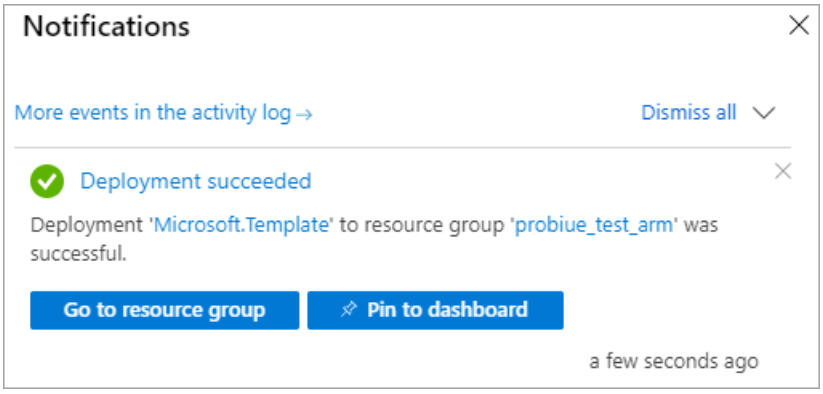
   

2. Select or enter the following values.

   Unless specified otherwise, use default values.

- **Subscription**: Select an Azure subscription.
- **Resource group**: Click **Create new**, and then enter a unique name for the new resource group.
- **Location**: Select a default location for resources created in the resource group.
- **Server Name**: Enter a name for the server resource.
- **Location**: Ignore for Analysis Services. Location is specified in Server Location.
- **Server location**: Enter the location of the Analysis Services server. This is often the same region as the default Location specified for the Resource Group, but not required. For example **North Central US**. For supported regions, see Analysis Services availability by region.
- **Sku Name**: Enter the sku name for the Analysis Services server to create. Choose from: B1, B2, D1, S0, S1, S2, S3, S4, S8v2, S9v2. Sku availability depends on region. S0 or D1 is recommended for evaluation and testing.
- **Capacity**: Enter the total number of query replica scale-out instances. Scale-out of more than one instance is supported in select regions only.
- **Firewall Settings**: Enter inbound firewall rules to define for the server. If not specified, firewall is disabled.
- **Backup Blob Container Uri**: Enter the SAS URI to a private Azure Blob Storage container with read, write and list permissions. Required only if you intend to use Backup/restore.
- **I agree to the terms and conditions state above**: Select.

3. Select **Purchase**. After the server has been deployed successfully, you get a notification:



Validate the deployment

Use the Azure portal or Azure PowerShell to verify the resource group and server resource was created.

PowerShell

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
(Get-AzResource -ResourceType "Microsoft.AnalysisServices/servers" -ResourceGroupName
$resourceGroupName).Name
 Write-Host "Press [ENTER] to continue..."
```

Clean up resources

When no longer needed, use the Azure portal, Azure CLI, or Azure PowerShell to delete the resource group and the server resource.

- CLI
- PowerShell

```
echo "Enter the Resource Group name:" &&
read resourceGroupName &&
az group delete --name $resourceGroupName &&
echo "Press [ENTER] to continue ..."
```

Next steps

In this quickstart, you used an ARM template to create a new resource group and an Azure Analysis Services server resource. After you've created a server resource by using the template, consider the following:

Quickstart: Configure server firewall - Portal

Quickstart: Configure server firewall - Portal

4/15/2021 • 2 minutes to read • Edit Online

This quickstart helps you configure a firewall for your Azure Analysis Services server. Enabling a firewall and configuring IP address ranges for only those computers accessing your server are an important part of securing your server and data.

Prerequisites

- An Analysis Services server in your subscription. To learn more, see Quickstart: Create a server - Portal or Quickstart: Create a server - PowerShell
- One or more IP address ranges for client computers (if needed).

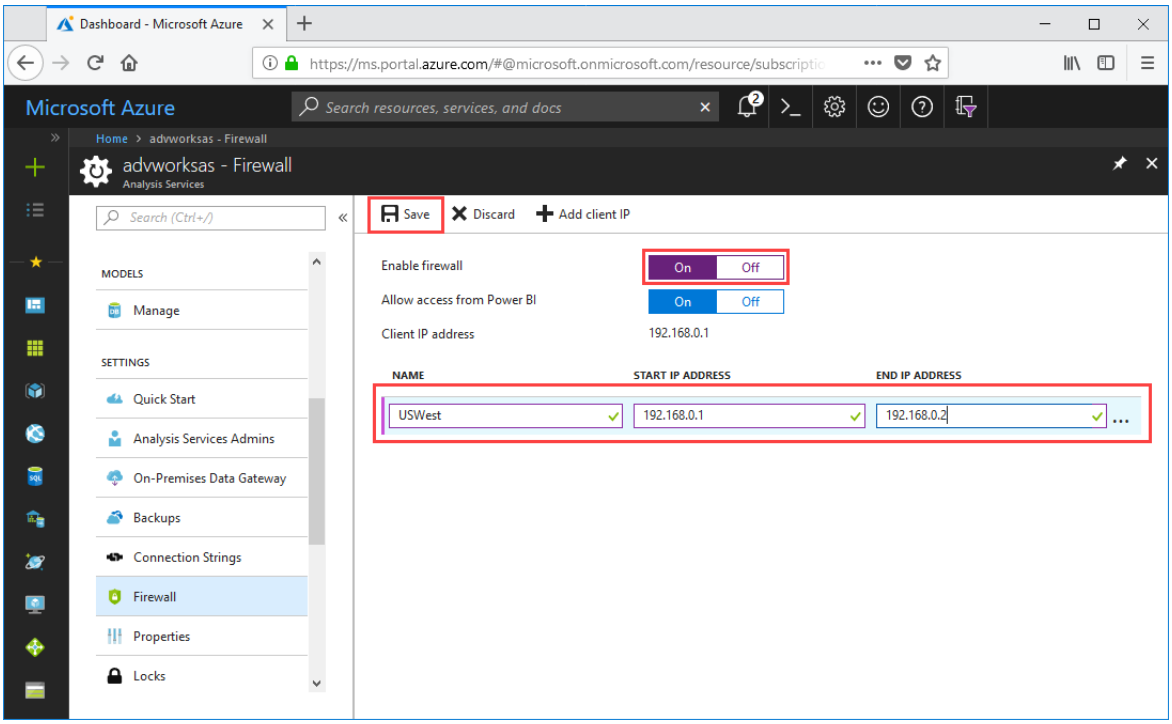
> **NOTE**
> Data import (refresh) and paginated report connections from Power BI Premium in Microsoft Cloud Germany are currently not supported when a firewall is enabled, even when the Allow access from Power BI setting is set to On.

Sign in to the Azure portal

Sign in to the portal

Configure a firewall

1. Click on your server to open the Overview page.
2. In **SETTINGS** > **Firewall** > **Enable firewall**, select **On**.
3. To enable connections from Power BI and Power BI Premium, in **Allow access from Power BI**, select **On**.
4. (Optional) Specify one or more IP address ranges. Enter a name, starting, and ending IP address for each range. Firewall rule name should be limited to 128 characters and can only contain uppercase characters, lowercase characters, numbers, underscore, and hyphen. Blank spaces and other special characters are not allowed.
5. Click **Save**.



Clean up resources

When no longer needed, delete IP address ranges, or disable the firewall.

Next steps

In this quickstart, you learned how to configure a firewall for your server. Now that you have server, and secured it with a firewall, you can add a basic sample data model to it from the portal. Having a sample model is helpful to learn about configuring model database roles and testing client connections. To learn more, continue to the tutorial for adding a sample model.

Tutorial: Add a sample model to your server

Tutorial: Add a sample model from the portal

11/2/2020 • 2 minutes to read • Edit Online

In this tutorial, you add a sample Adventure Works tabular model database to your server. The sample model is a completed version of the Adventure Works Internet Sales (1200) sample data model. A sample model is useful for testing model management, connecting with tools and client applications, and querying model data. This tutorial uses the Azure portal and SQL Server Management Studio (SSMS) to:

- Add a completed sample tabular data model to a server
- Connect to the model with SSMS

If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

To complete this tutorial, you need:

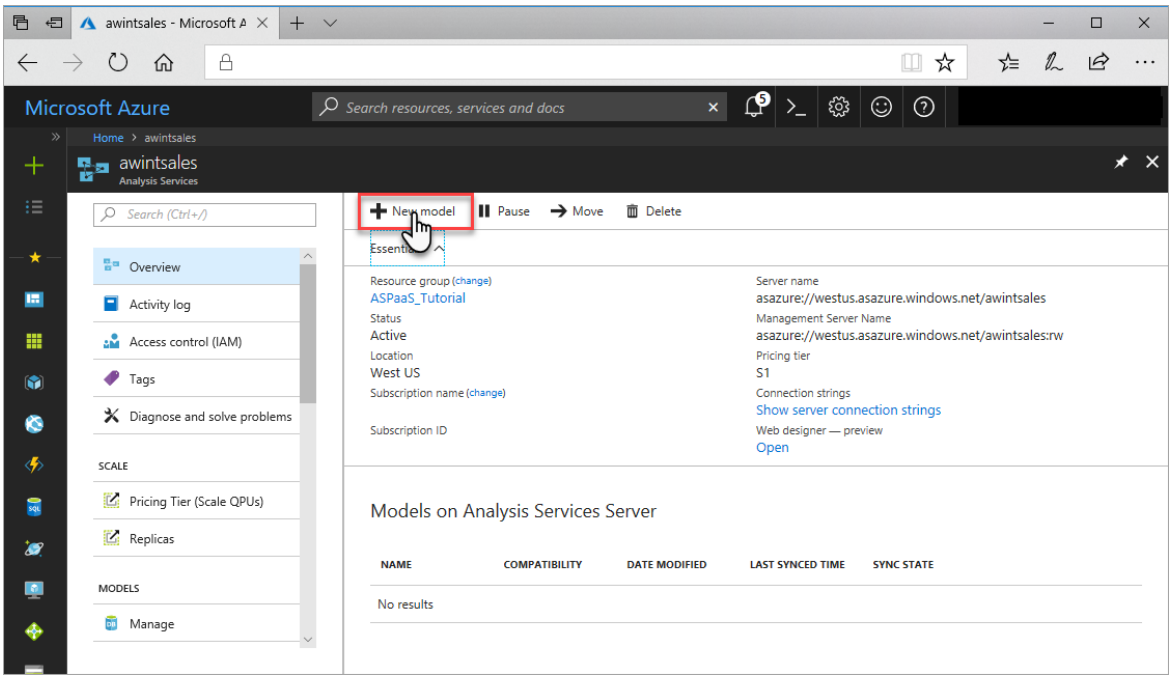
- An Azure Analysis Services server. To learn more, see Create a server - portal.
- Server administrator permissions
- SQL Server Management Studio

Sign in to the Azure portal

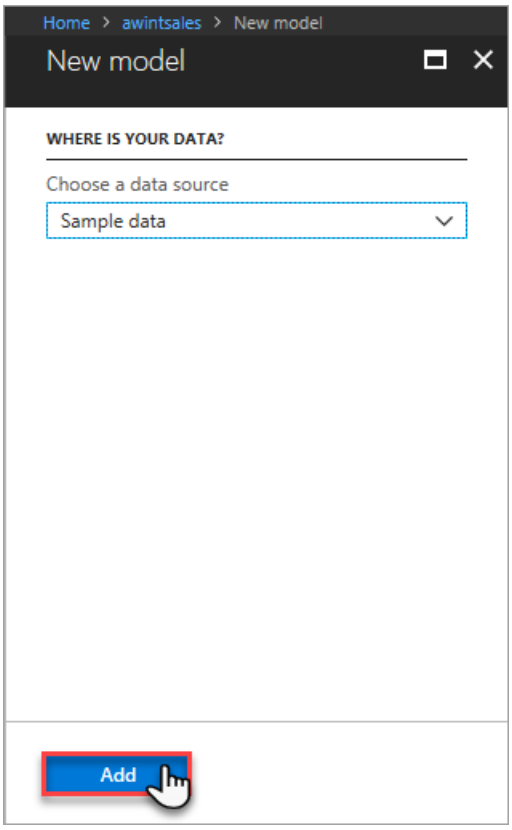
Sign in to the portal.

Add a sample model

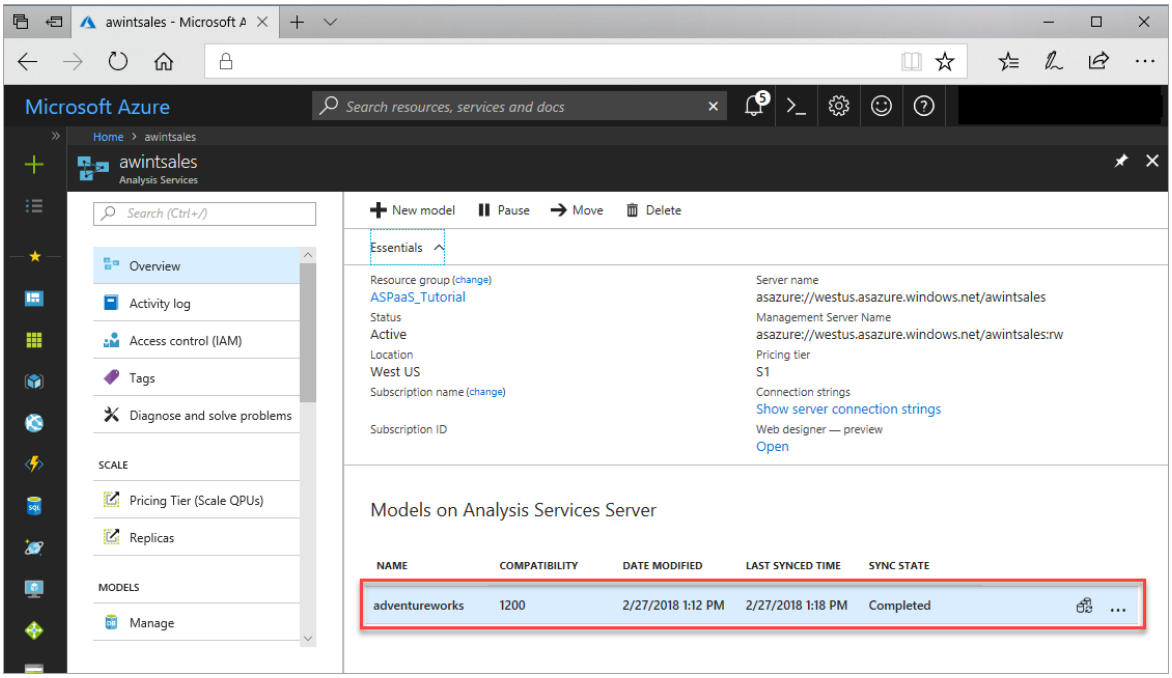
1. In server **Overview**, click **New model**.



2. In **New model** > **Choose a data source**, verify **Sample data** is selected, and then click **Add**.



3. In **Overview**, verify the `adventureworks` sample model is added.

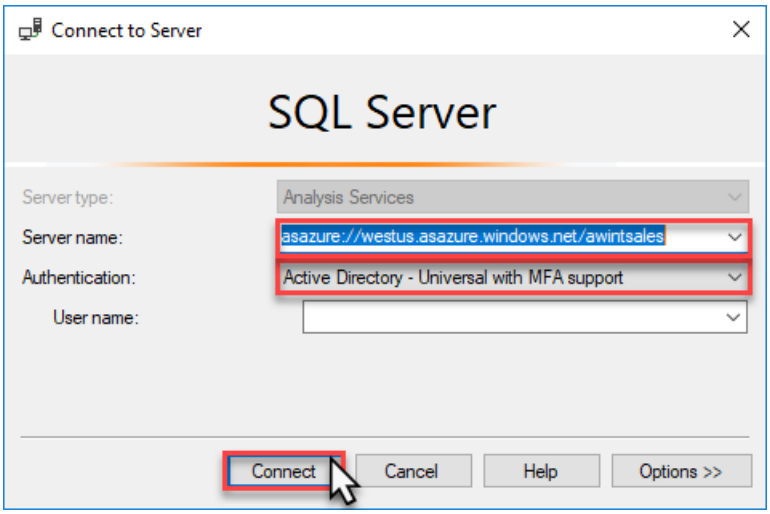


Clean up resources

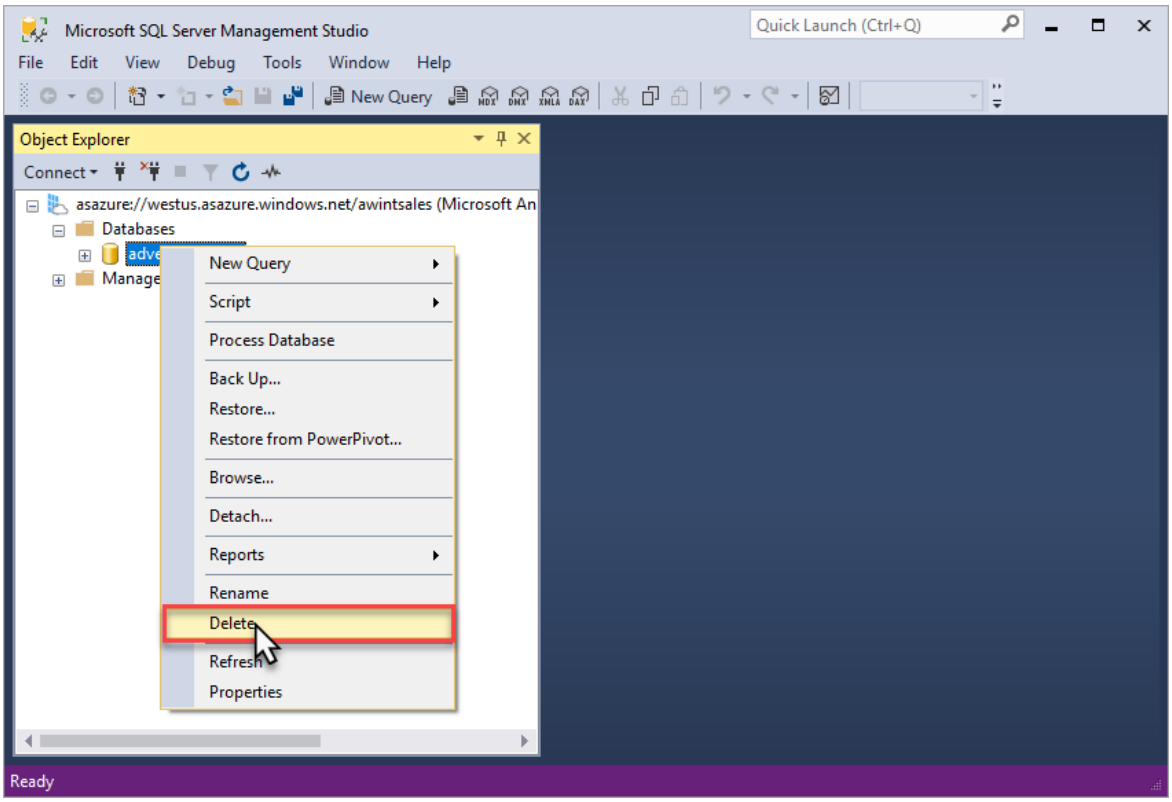
Your sample model is using cache memory resources. If you are not using your sample model for testing, you should remove it from your server.

These steps describe how to delete a model from a server by using SSMS.

1. In SSMS > **Object Explorer**, click **Connect** > **Analysis Services**.
2. In **Connect to Server**, paste in the server name, then in **Authentication**, choose **Active Directory - Universal with MFA support**, enter your username, and then click **Connect**.



3. In **Object Explorer**, right-click the `adventureworks` sample database, and then click **Delete**.



Next steps

In this tutorial, you learned how to add a basic, sample model to your server. Now that you have a model database, you can connect to it from SQL Server Management Studio and add user roles. To learn more, continue with the next tutorial.

Tutorial: Configure server administrator and user roles

Tutorial: Configure server administrator and user roles

11/2/2020 • 4 minutes to read • Edit Online

In this tutorial, you use SQL Server Management Studio (SSMS) to connect to your server in Azure to configure server administrator and model database roles. You're also introduced to Tabular Model Scripting Language (TMSL). TMSL is a JSON-based scripting language for tabular models at the 1200 and higher compatibility levels. It can be used to automate many tabular modeling tasks. TMSL is often used with PowerShell, but in this tutorial, you use the XMLA query editor in SSMS. With this tutorial, you complete these tasks:

- Get your server name from the portal
- Connect to your server by using SSMS
- Add a user or group to the server administrator role
- Add a user or group to the model database administrator role
- Add a new model database role and add a user or group

To learn more about user security in Azure Analysis Services, see Authentication and user permissions.

Prerequisites

- An Azure Active Directory in your subscription.
- Created an Azure Analysis Services server in your subscription.
- Have server administrator permissions.
- Add the adventureworks sample model to your server.
- Install the latest version of SQL Server Management Studio (SSMS).

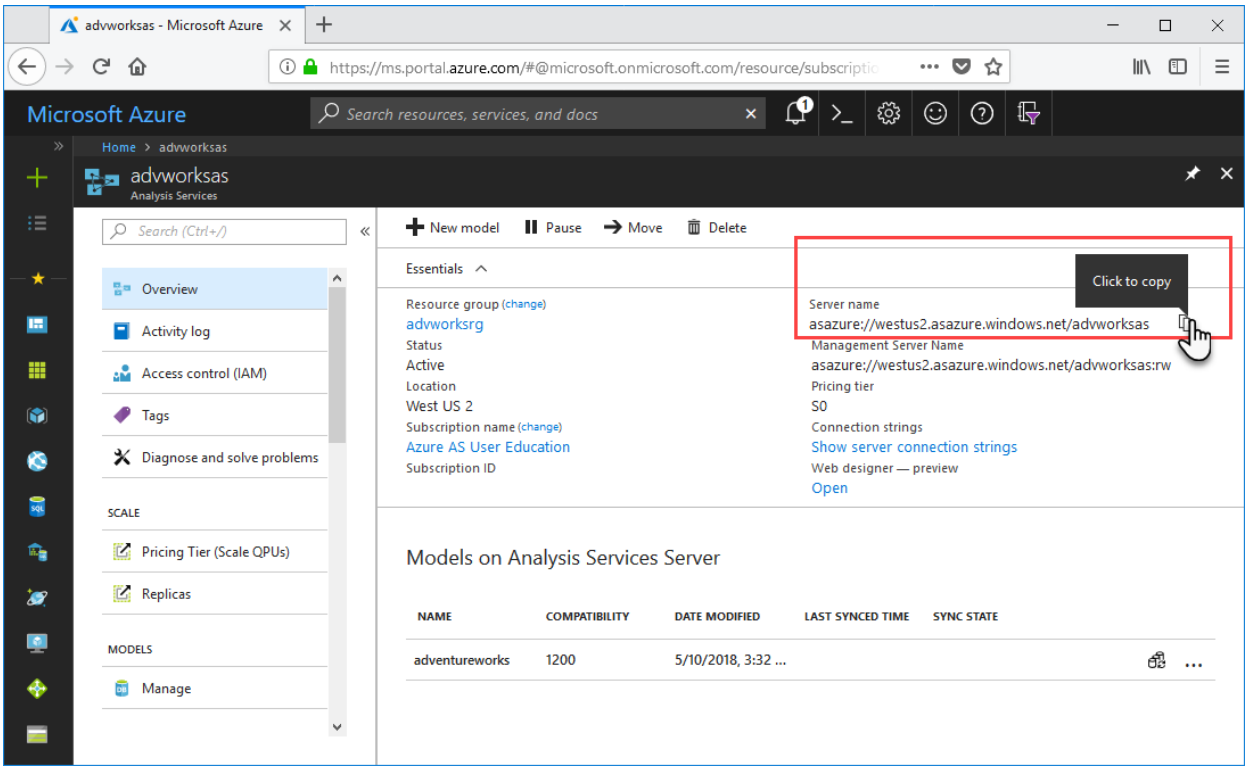
Sign in to the Azure portal

Sign in to the portal.

Get server name

In order to connect to your server from SSMS, you first need the server name. You can get the server name from the portal.

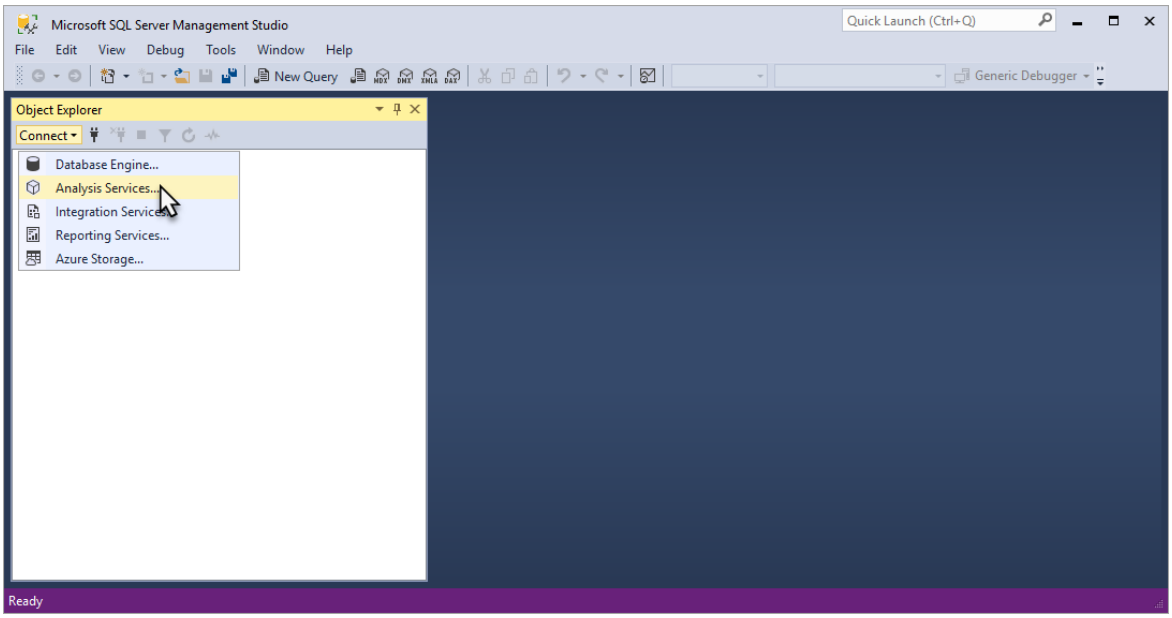
In **Azure portal** > server > **Overview** > **Server name**, copy the server name.



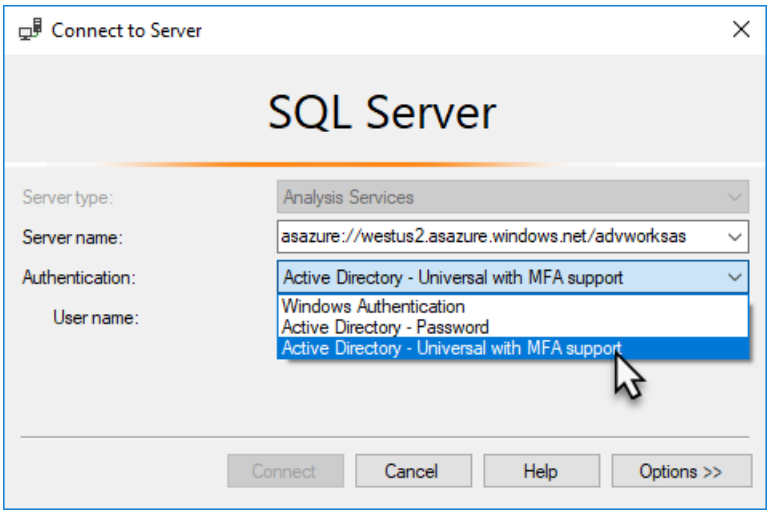
Connect in SSMS

For the remaining tasks, you use SSMS to connect to and manage your server.

1. In SSMS > **Object Explorer**, click **Connect** > **Analysis Services**.

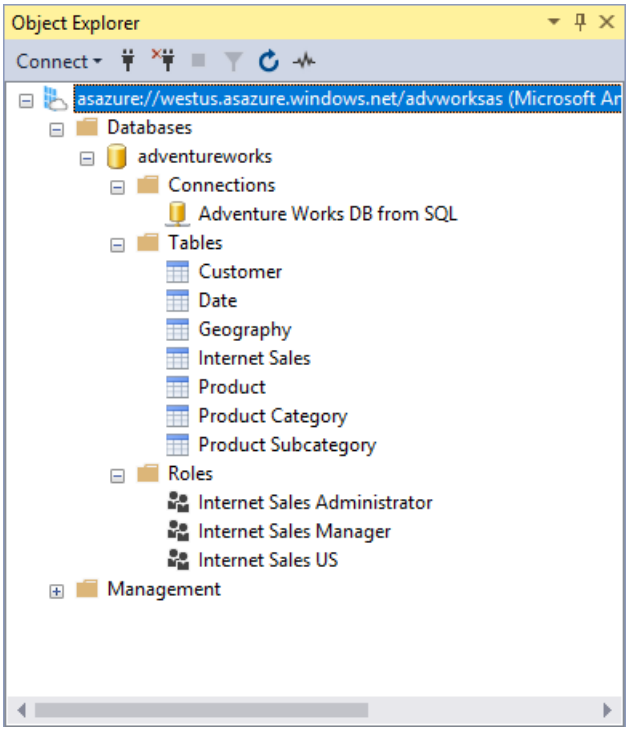


2. In the **Connect to Server** dialog box, in **Server name**, paste in the server name you copied from the portal. In **Authentication**, choose **Active Directory Universal with MFA Support**, then enter your user account, and then press **Connect**.



> **TIP**
>
> Choosing Active Directory Universal with MFA Support is recommended. This type of authentication type supports non-interactive and multi-factor authentication.

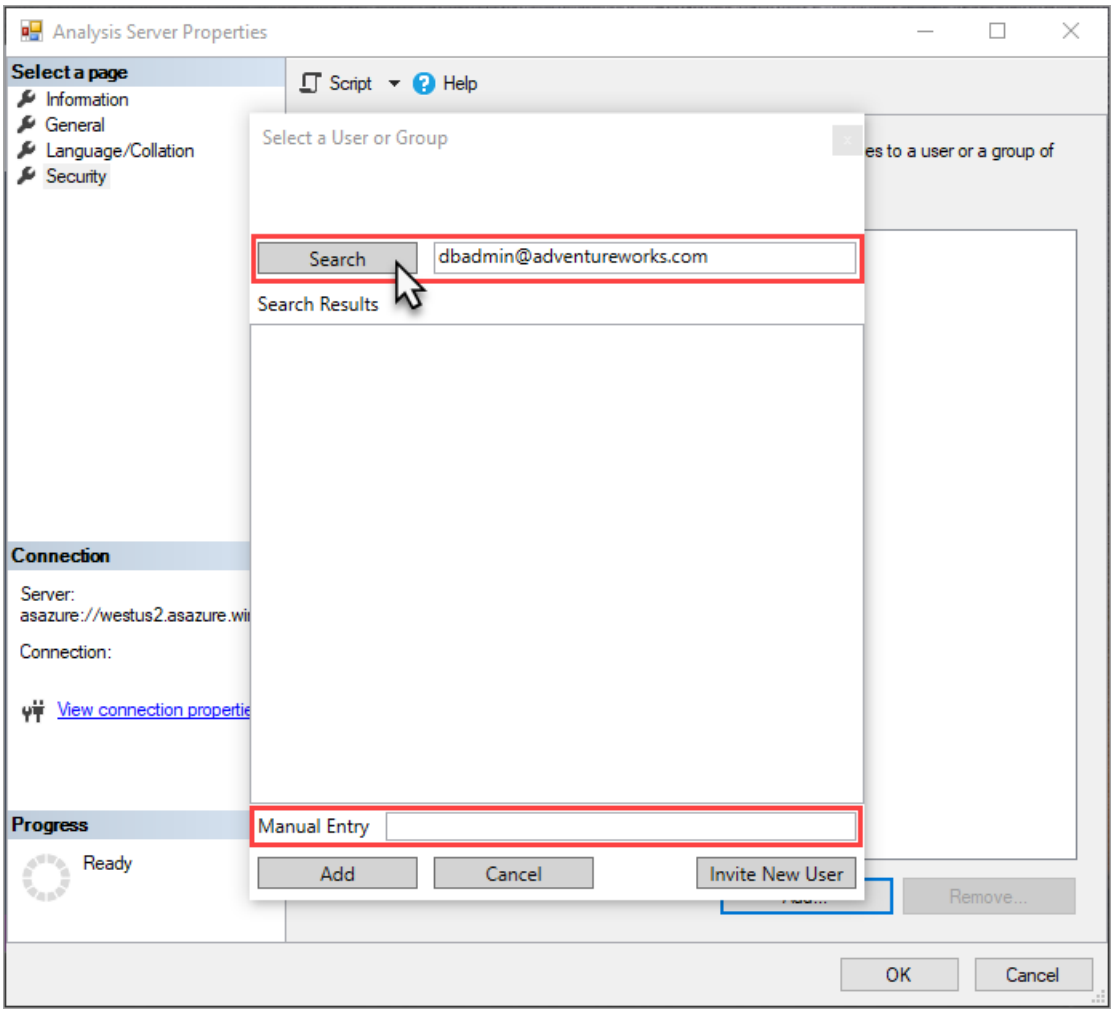
3. In **Object Explorer**, expand to see server objects. Right-click to see server properties.



Add a user account to the server administrator role

In this task, you add a user or group account from your Azure AD to the server administrator role. If specifying a security group, use `obj:groupid@tenantid`.

1. In **Object Explorer**, right-click your server name, and then click **Properties**.
2. In the **Analysis Server Properties** window, click **Security** > **Add**.
3. In the **Select a User or Group** window, enter a user or group account in your Azure AD, and then click **Add**.



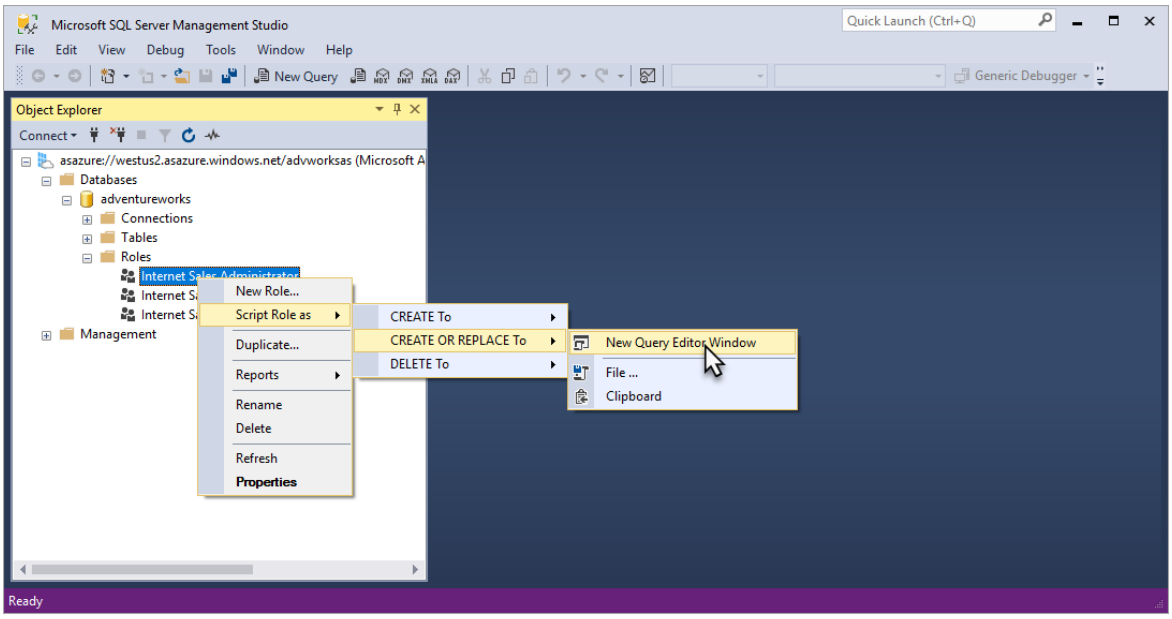
4. Click **OK**, to close **Analysis Server Properties**.

> **TIP**
> You can also add server administrators by using **Analysis Services Admins** in the portal.

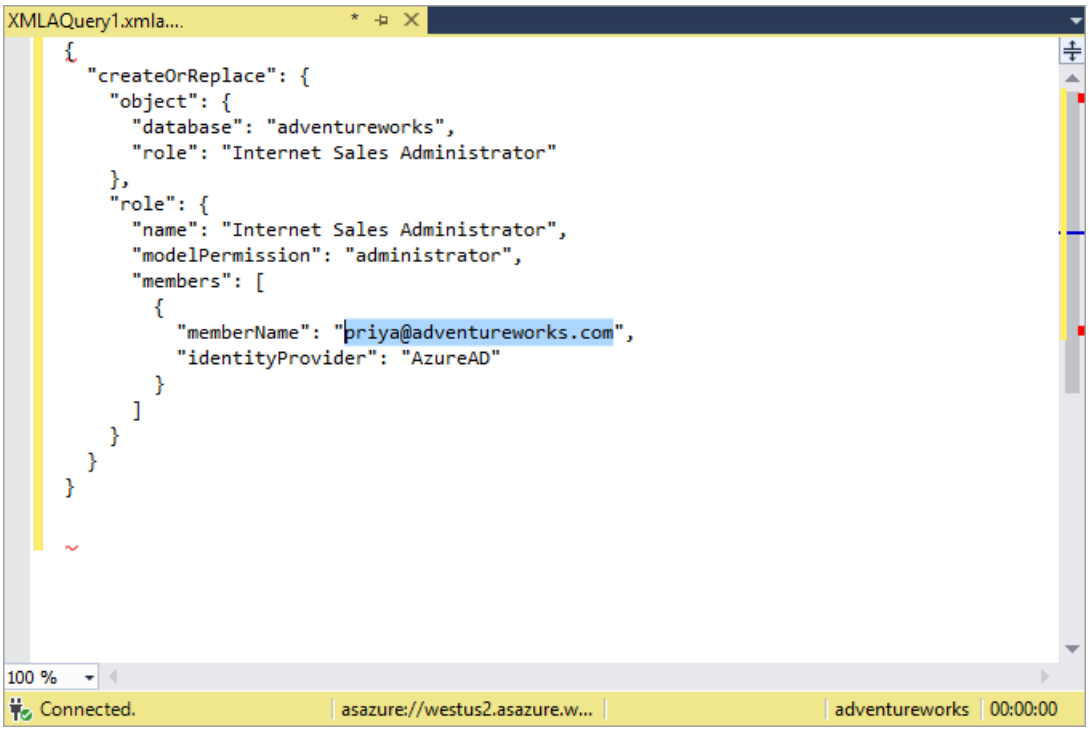
Add a user to the model database administrator role

In this task, you add a user or group account to the Internet Sales Administrator role that already exists in the model. This role has Full control (Administrator) permissions for the adventureworks sample model database. This task uses the CreateOrReplace TMSL command in a script created for you.

1. In **Object Explorer**, expand **Databases** > **adventureworks** > **Roles**.
2. Right-click **Internet Sales Administrator**, then click **Script Role as** > **CREATE OR REPLACE To** > **New Query Editor Window**.



3. In the **XMLAQuery**, change the value for **"memberName":** to a user or group account in your Azure AD. By default, the account you're signed in with is included; however, you do not need to add your own account because you are already a server administrator.



4. Press **F5**, to execute the script.

Add a new model database role and add a user or group

In this task, you use the Create command in a TMSL script to create a new Internet Sales Global role, specify *read* permissions for the role, and add a user or group account from your Azure AD.

1. In **Object Explorer**, right-click **adventureworks**, and then click **New Query** > **XMLA**.

2. Copy and paste the following TMSL script into the query editor:

```
{
"create": {
  "parentObject": {
    "database": "adventureworks",
   },
   "role": {
     "name": "Internet Sales Global",
     "description": "All users can query model data",
     "modelPermission": "read",
     "members": [
       {
         "memberName": "globalsales@adventureworks.com",
         "identityProvider": "AzureAD"
       }
     ]
   }
  }
}
```

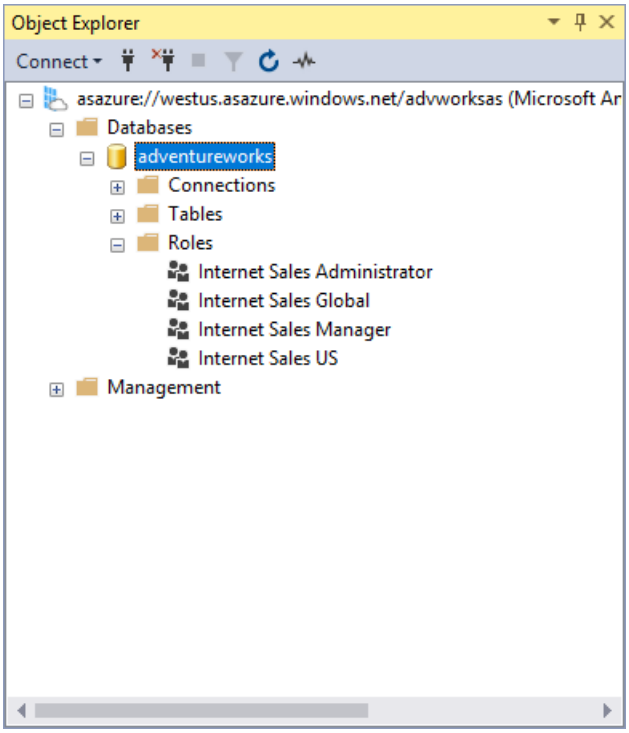
3. Change `"memberName": "globalsales@adventureworks.com"` object value to a user or group account in your Azure AD.

4. Press **F5**, to execute the script.

Verify your changes

1. In **Object Explorer**, click your servername, and then click **Refresh** or press **F5**.

2. Expand **Databases** > **adventureworks** > **Roles**. Verify the user account and new role changes you added in the previous tasks appear.



Clean up resources

When no longer needed, delete the user or group accounts and roles. To do so, use **Role Properties** > **Membership** to remove user accounts, or right-click a role and then click **Delete**.

Next steps

In this tutorial, you learned how to connect to your Azure AS server and explore the adventureworks sample model databases and properties in SSMS. You also learned how to use SSMS and TMSL scripts to add users or groups to existing and new roles. Now that you have user permissions configured for your server and sample model database, you and other users can connect to it by using client applications like Power BI. To learn more, continue to the next tutorial.

Tutorial: Connect with Power BI Desktop

Tutorial: Connect with Power BI Desktop

11/2/2020 • 2 minutes to read • Edit Online

In this tutorial, you use Power BI Desktop to connect to the adventureworks sample model database on your server. The tasks you complete simulate a typical user connection to the model and creating a basic report from model data.

- Get your server name from the portal
- Connect by using Power BI Desktop
- Create a basic report

Prerequisites

- Add the adventureworks sample model database to your server.
- Have *read* permissions for the adventureworks sample model database.
- Install the newest Power BI Desktop.

Sign in to the Azure portal

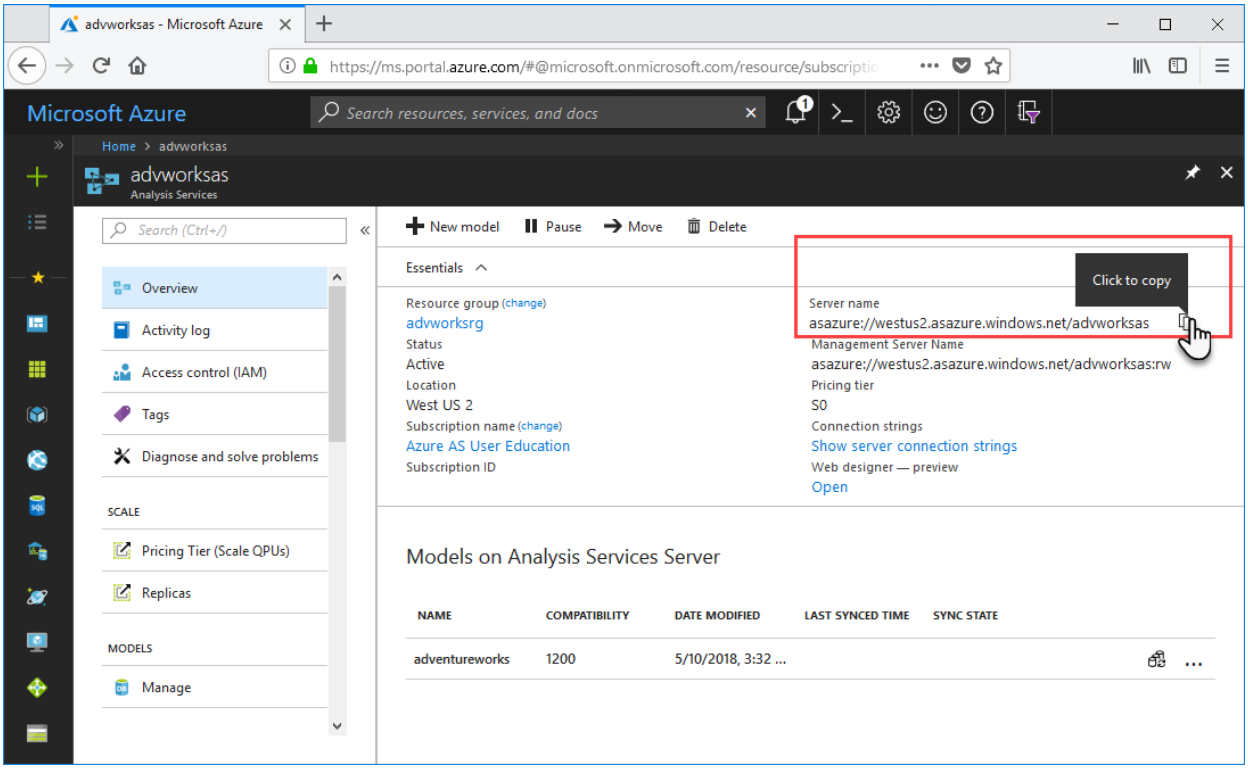
In this tutorial, you sing in to the portal to get the server name only. Typically, users would get the server name from the server administrator.

Sign in to the portal.

Get server name

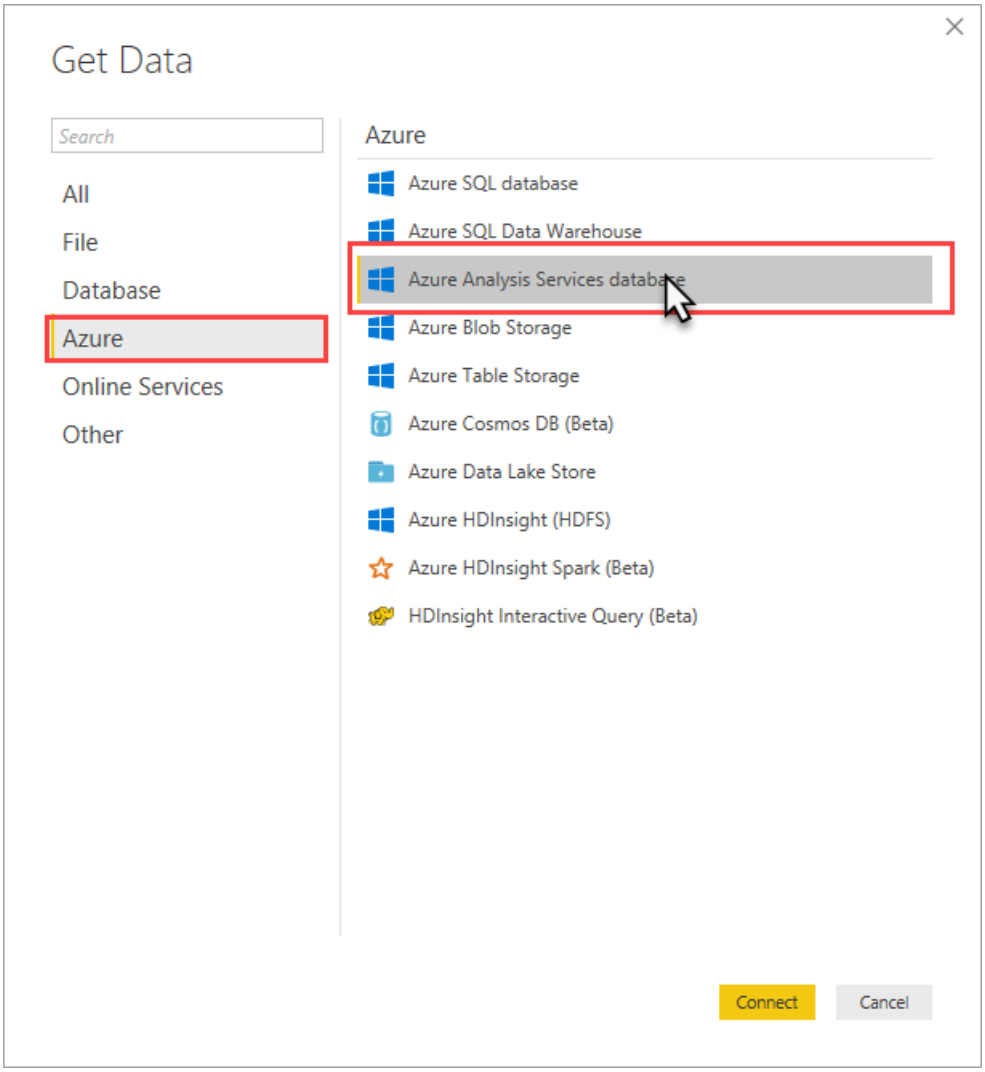
In order to connect to your server from Power BI Desktop, you first need the server name. You can get the server name from the portal.

In **Azure portal** > server > **Overview** > **Server name**, copy the server name.

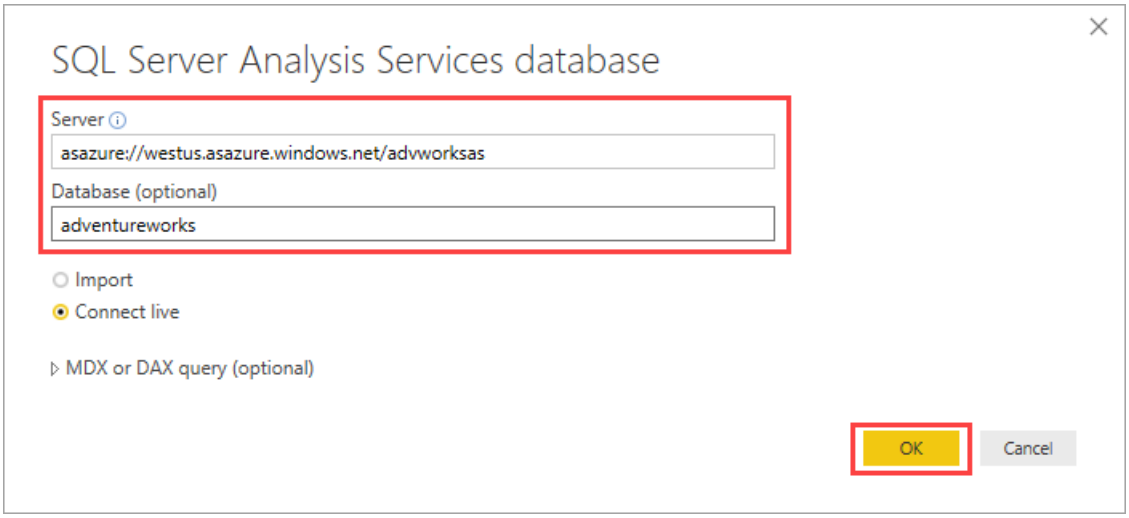


Connect in Power BI Desktop

1. In Power BI Desktop, click **Get Data** > **Azure** > **Azure Analysis Services database**.



2. In **Server**, paste the server name, then in **Database**, enter **adventureworks**, and then click **OK**.

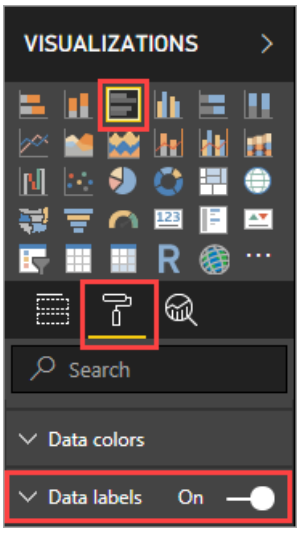


3. When prompted, enter your credentials. The account you enter must have at least read permissions for the adventureworks sample model database.

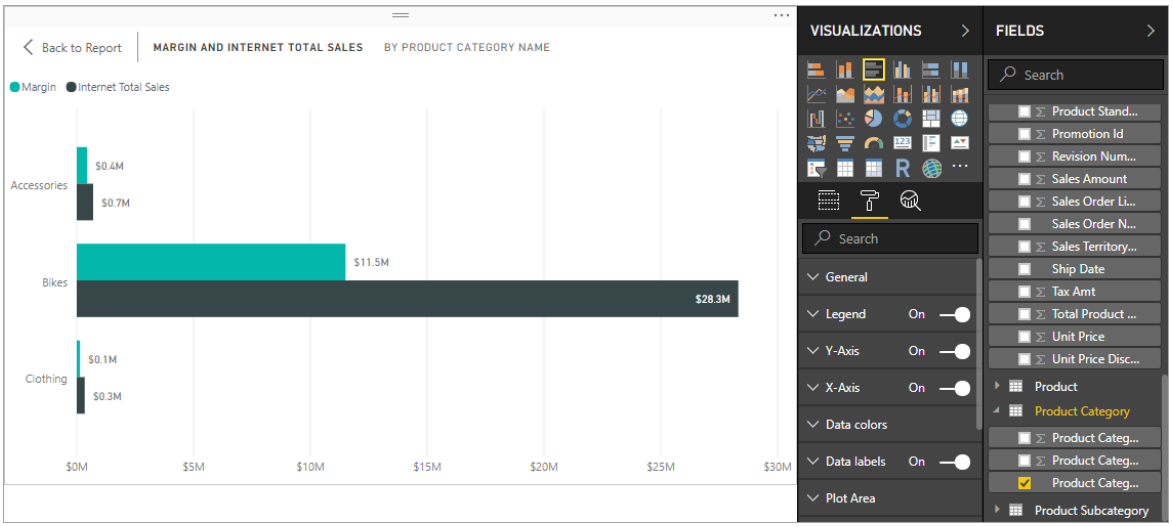
   The adventureworks model opens in Power BI Desktop with a blank report in Report view. The **Fields** list displays all non-hidden model objects. Connection status is displayed in the lower-right corner.

4. In **VISUALIZATIONS**, select **Clustered Bar Chart**, then click **Format** (paint roller icon), and then turn

on **Data labels**.



5. In **FIELDS** > **Internet Sales** table, select **Internet Sales Total** and **Margin** measures. In **Product Category** table, select **Product Category Name**.



Take a few minutes to explore the adventureworks sample model by creating different visualizations and slicing on data and metrics. When you're happy with your report, be sure to save.

Clean up resources

If no longer needed, do not save your report or delete the file if you did save.

Next steps

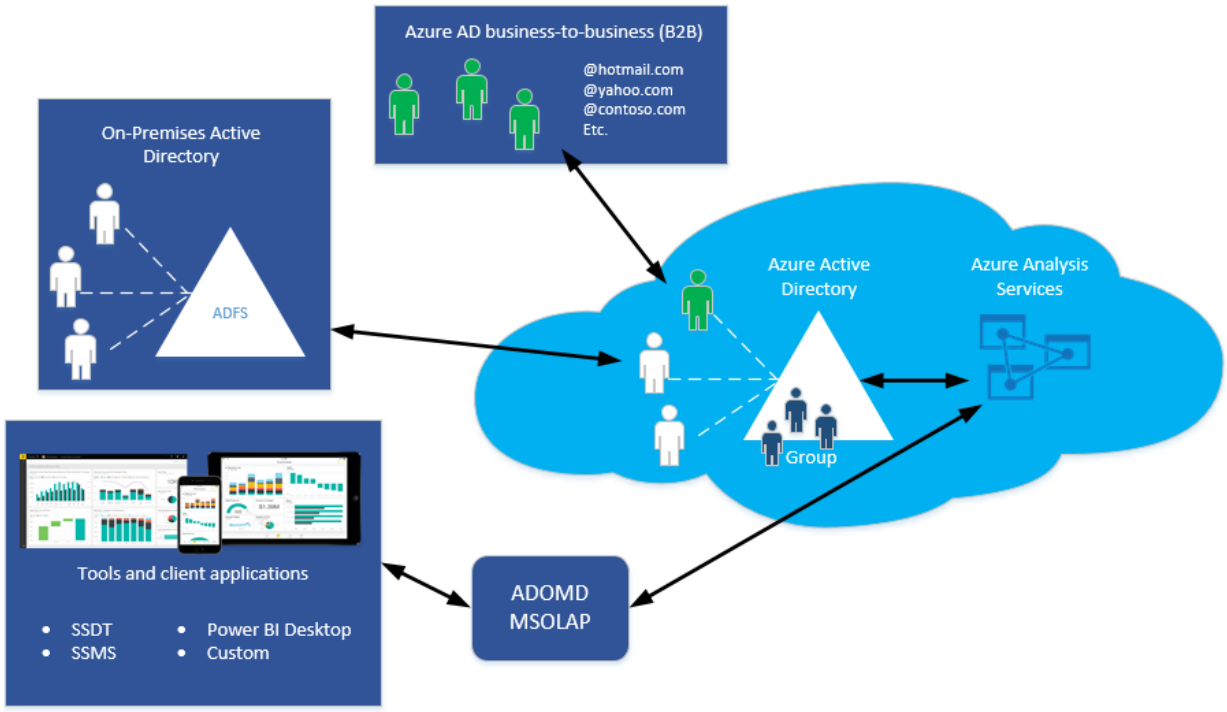
In this tutorial, you learned how to use Power BI Desktop to connect to a data model on a server and create a basic report. If you're not familiar with how to create a data model, see the Adventure Works Internet Sales tabular data modeling tutorial in the SQL Server Analysis Services docs.

Authentication and user permissions

3/5/2021 • 5 minutes to read • Edit Online

Azure Analysis Services uses Azure Active Directory (Azure AD) for identity management and user authentication. Any user creating, managing, or connecting to an Azure Analysis Services server must have a valid user identity in an Azure AD tenant in the same subscription.

Azure Analysis Services supports Azure AD B2B collaboration. With B2B, users from outside an organization can be invited as guest users in an Azure AD directory. Guests can be from another Azure AD tenant directory or any valid email address. Once invited and the user accepts the invitation sent by email from Azure, the user identity is added to the tenant directory. Those identities can be added to security groups or as members of a server administrator or database role.



Authentication

All client applications and tools use one or more of the Analysis Services client libraries (AMO, MSOLAP, ADOMD) to connect to a server.

All three client libraries support both Azure AD interactive flow, and non-interactive authentication methods. The two non-interactive methods, Active Directory Password and Active Directory Integrated Authentication methods can be used in applications utilizing AMOMD and MSOLAP. These two methods never result in pop-up dialog boxes.

Client applications like Excel and Power BI Desktop, and tools like SSMS and Analysis Services projects extension for Visual Studio install the latest versions of the libraries when updated to the latest release. Power BI Desktop, SSMS, and Analysis Services projects extension are updated monthly. Excel is updated with Microsoft 365. Microsoft 365 updates are less frequent, and some organizations use the deferred channel, meaning updates are deferred up to three months.

Depending on the client application or tool you use, the type of authentication and how you sign in may be different. Each application may support different features for connecting to cloud services like Azure Analysis

Services.

Power BI Desktop, Visual Studio, and SSMS support Active Directory Universal Authentication, an interactive method that also supports Azure AD Multi-Factor Authentication (MFA). Azure AD MFA helps safeguard access to data and applications while providing a simple sign-in process. It delivers strong authentication with several verification options (phone call, text message, smart cards with pin, or mobile app notification). Interactive MFA with Azure AD can result in a pop-up dialog box for validation. **Universal Authentication is recommended**.

If signing in to Azure by using a Windows account, and Universal Authentication is not selected or available (Excel), Active Directory Federation Services (AD FS) is required. With Federation, Azure AD and Microsoft 365 users are authenticated using on-premises credentials and can access Azure resources.

SQL Server Management Studio (SSMS)

Azure Analysis Services servers support connections from SSMS V17.1 and higher by using Windows Authentication, Active Directory Password Authentication, and Active Directory Universal Authentication. In general, it's recommended you use Active Directory Universal Authentication because:

- Supports interactive and non-interactive authentication methods.
- Supports Azure B2B guest users invited into the Azure AS tenant. When connecting to a server, guest users must select Active Directory Universal Authentication when connecting to the server.
- Supports Multi-Factor Authentication (MFA). Azure AD MFA helps safeguard access to data and applications with a range of verification options: phone call, text message, smart cards with pin, or mobile app notification. Interactive MFA with Azure AD can result in a pop-up dialog box for validation.

Visual Studio

Visual Studio connects to Azure Analysis Services by using Active Directory Universal Authentication with MFA support. Users are prompted to sign in to Azure on the first deployment. Users must sign in to Azure with an account with server administrator permissions on the server they are deploying to. When signing in to Azure the first time, a token is assigned. The token is cached in-memory for future reconnects.

Power BI Desktop

Power BI Desktop connects to Azure Analysis Services using Active Directory Universal Authentication with MFA support. Users are prompted to sign in to Azure on the first connection. Users must sign in to Azure with an account that is included in a server administrator or database role.

Excel

Excel users can connect to a server by using a Windows account, an organization ID (email address), or an external email address. External email identities must exist in the Azure AD as a guest user.

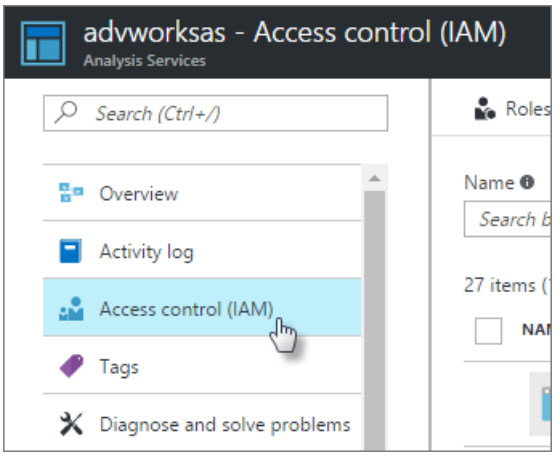
User permissions

Server administrators are specific to an Azure Analysis Services server instance. They connect with tools like Azure portal, SSMS, and Visual Studio to perform tasks like adding databases and managing user roles. By default, the user that creates the server is automatically added as an Analysis Services server administrator. Other administrators can be added by using Azure portal or SSMS. Server administrators must have an account in the Azure AD tenant in the same subscription. To learn more, see Manage server administrators.

Database users connect to model databases by using client applications like Excel or Power BI. Users must be added to database roles. Database roles define administrator, process, or read permissions for a database. It's important to understand database users in a role with administrator permissions is different than server administrators. However, by default, server administrators are also database administrators. To learn more, see Manage database roles and users.

Azure resource owners. Resource owners manage resources for an Azure subscription. Resource owners can

add Azure AD user identities to Owner or Contributor Roles within a subscription by using **Access control** in Azure portal, or with Azure Resource Manager templates.



Roles at this level apply to users or accounts that need to perform tasks that can be completed in the portal or by using Azure Resource Manager templates. To learn more, see Azure role-based access control (Azure RBAC).

Database roles

Roles defined for a tabular model are database roles. That is, the roles contain members consisting of Azure AD users and security groups that have specific permissions that define the action those members can take on a model database. A database role is created as a separate object in the database, and applies only to the database in which that role is created.

By default, when you create a new tabular model project, the model project does not have any roles. Roles can be defined by using the Role Manager dialog box in Visual Studio. When roles are defined during model project design, they are applied only to the model workspace database. When the model is deployed, the same roles are applied to the deployed model. After a model has been deployed, server and database administrators can manage roles and members by using SSMS. To learn more, see Manage database roles and users.

Next steps

Manage access to resources with Azure Active Directory groups
Manage database roles and users
Manage server administrators
Azure role-based access control (Azure RBAC)

Automation with service principals

4/27/2021 • 3 minutes to read • Edit Online

Service principals are an Azure Active Directory application resource you create within your tenant to perform unattended resource and service level operations. They're a unique type of *user identity* with an application ID and password or certificate. A service principal has only those permissions necessary to perform tasks defined by the roles and permissions for which it's assigned.

In Analysis Services, service principals are used with Azure Automation, PowerShell unattended mode, custom client applications, and web apps to automate common tasks. For example, provisioning servers, deploying models, data refresh, scale up/down, and pause/resume can all be automated by using service principals. Permissions are assigned to service principals through role membership, much like regular Azure AD UPN accounts.

Analysis Services also supports operations performed by managed identities using service principals. To learn more, see Managed identities for Azure resources and Azure services that support Azure AD authentication.

Create service principals

Service principals can be created in the Azure portal or by using PowerShell. To learn more, see:

Create service principal - Azure portal
Create service principal - PowerShell

Store credential and certificate assets in Azure Automation

Service principal credentials and certificates can be stored securely in Azure Automation for runbook operations. To learn more, see:

Credential assets in Azure Automation
Certificate assets in Azure Automation

Add service principals to server admin role

Before you can use a service principal for Analysis Services server management operations, you must add it to the server administrators role. Service principals must be added directly to the server administrator role. Adding a service principal to a security group, and then adding that security group to the server administrator role is not supported. To learn more, see Add a service principal to the server administrator role.

Service principals in connection strings

Service principal appID and password or certificate can be used in connection strings much the same as a UPN.

PowerShell

> **NOTE**
> This article has been updated to use the Azure Az PowerShell module. The Az PowerShell module is the recommended PowerShell module for interacting with Azure. To get started with the Az PowerShell module, see Install Azure PowerShell. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Using Az.AnalysisServices module

When using a service principal for resource management operations with the Az.AnalysisServices module, use `Connect-AzAccount` cmdlet.

In the following example, appID and a password are used to perform control plane operations for synchronization to read-only replicas and scale up/out:

```
Param (
        [Parameter(Mandatory=$true)] [String] $AppId,
        [Parameter(Mandatory=$true)] [String] $PlainPWord,
        [Parameter(Mandatory=$true)] [String] $TenantId
       )
$PWord = ConvertTo-SecureString -String $PlainPWord -AsPlainText -Force
$Credential = New-Object -TypeName "System.Management.Automation.PSCredential" -ArgumentList $AppId, $PWord

# Connect using Az module
Connect-AzAccount -Credential $Credential -SubscriptionId "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

# Syncronize a database for query scale out
Sync-AzAnalysisServicesInstance -Instance "asazure://westus.asazure.windows.net/testsvr" -Database "testdb"

# Scale up the server to an S1, set 2 read-only replicas, and remove the primary from the query pool. The
new replicas will hydrate from the synchronized data.
Set-AzAnalysisServicesServer -Name "testsvr" -ResourceGroupName "testRG" -Sku "S1" -ReadonlyReplicaCount 2 -
DefaultConnectionMode Readonly
```

Using SQLServer module

In the following example, appID and a password are used to perform a model database refresh operation:

```
Param (
        [Parameter(Mandatory=$true)] [String] $AppId,
        [Parameter(Mandatory=$true)] [String] $PlainPWord,
        [Parameter(Mandatory=$true)] [String] $TenantId
       )
$PWord = ConvertTo-SecureString -String $PlainPWord -AsPlainText -Force

$Credential = New-Object -TypeName "System.Management.Automation.PSCredential" -ArgumentList $AppId, $PWord

Invoke-ProcessTable -Server "asazure://westcentralus.asazure.windows.net/myserver" -TableName "MyTable" -
Database "MyDb" -RefreshType "Full" -ServicePrincipal -ApplicationId $AppId -TenantId $TenantId -Credential
$Credential
```

AMO and ADOMD

When connecting with client applications and web apps, AMO and ADOMD client libraries version 15.0.2 and higher installable packages from NuGet support service principals in connection strings using the following syntax: `app:AppID` and password or `cert:thumbprint`.

In the following example, `appID` and a `password` are used to perform a model database refresh operation:

```
string appId = "xxx";
string authKey = "yyy";
string connString = $"Provider=MSOLAP;Data Source=asazure://westus.asazure.windows.net/<servername>;User
ID=app:{appId};Password={authKey};";
Server server = new Server();
server.Connect(connString);
Database db = server.Databases.FindByName("adventureworks");
Table tbl = db.Model.Tables.Find("DimDate");
tbl.RequestRefresh(RefreshType.Full);
db.Model.SaveChanges();
```

Next steps

Sign in with Azure PowerShell
Refresh with Logic Apps
Refresh with Azure Automation
Add a service principal to the server administrator role
Automate Power BI Premium workspace and dataset tasks with service principals

Connecting to on-premises data sources with On-premises data gateway

4/27/2021 • 3 minutes to read • Edit Online

The on-premises data gateway provides secure data transfer between on-premises data sources and your Azure Analysis Services servers in the cloud. In addition to working with multiple Azure Analysis Services servers in the same region, the latest version of the gateway also works with Azure Logic Apps, Power BI, Power Apps, and Power Automate. While the gateway you install is the same across all of these services, Azure Analysis Services and Logic Apps have some additional steps.

Information provided here is specific to how Azure Analysis Services works with the On-premises Data Gateway. To learn more about the gateway in general and how it works with other services, see What is an on-premises data gateway?.

For Azure Analysis Services, getting setup with the gateway the first time is a four-part process:

- **Download and run setup** - This step installs a gateway service on a computer in your organization. You also sign in to Azure using an account in your tenant's Azure AD. Azure B2B (guest) accounts are not supported.
- **Register your gateway** - In this step, you specify a name and recovery key for your gateway and select a region, registering your gateway with the Gateway Cloud Service. Your gateway resource can be registered in any region, but it's recommended it be in the same region as your Analysis Services servers.
- **Create a gateway resource in Azure** - In this step, you create a gateway resource in Azure.
- **Connect the gateway resource to servers** - Once you have a gateway resource, you can begin connecting servers to it. You can connect multiple servers and other resources provided they are in the same region.

Installing

When installing for an Azure Analysis Services environment, it's important you follow the steps described in Install and configure on-premises data gateway for Azure Analysis Services. This article is specific to Azure Analysis Services. It includes additional steps required to setup an On-premises data gateway resource in Azure, and connect your Azure Analysis Services server to the resource.

Connecting to a gateway resource in a different subscription

It's recommended you create your Azure gateway resource in the same subscription as your server. However, you can configure your servers to connect to a gateway resource in another subscription. Connecting to a gateway resource in another subscription is not supported when configuring existing server settings or creating a new server in the portal, but can be configured by using PowerShell. To learn more, see Connect gateway resource to server.

Ports and communication settings

The gateway creates an outbound connection to Azure Service Bus. It communicates on outbound ports: TCP 443 (default), 5671, 5672, 9350 through 9354. The gateway does not require inbound ports.

You may need to include IP addresses for your data region in your firewall. You can download the Microsoft

Azure Datacenter IP list. This list is updated weekly. The IP Addresses listed in the Azure Datacenter IP list are in CIDR notation. To learn more, see Classless Inter-Domain Routing.

The following are fully qualified domain names used by the gateway.

DOMAIN NAMES	OUTBOUND PORTS	DESCRIPTION
*.powerbi.com	80	HTTP used to download the installer.
*.powerbi.com	443	HTTPS
*.analysis.windows.net	443	HTTPS
*.login.windows.net, login.live.com, aadcdn.msauth.net	443	HTTPS
*.servicebus.windows.net	5671-5672	Advanced Message Queuing Protocol (AMQP)
*.servicebus.windows.net	443, 9350-9354	Listeners on Service Bus Relay over TCP (requires 443 for Access Control token acquisition)
*.frontend.clouddatahub.net	443	HTTPS
*.core.windows.net	443	HTTPS
login.microsoftonline.com	443	HTTPS
*.msftncsi.com	80	Used to test internet connectivity if the gateway is unreachable by the Power BI service.
*.microsoftonline-p.com	443	Used for authentication depending on configuration.
dc.services.visualstudio.com	443	Used by AppInsights to collect telemetry.

Next steps

The following articles are included in the On-premises data gateway general content that applies to all services the gateway supports:

- On-premises data gateway FAQ
- Use the on-premises data gateway app
- Tenant level administration
- Configure proxy settings
- Adjust communication settings
- Configure log files
- Troubleshoot
- Monitor and optimize gateway performance

Connecting to servers

3/5/2021 • 2 minutes to read • Edit Online

This article describes connecting to a server by using data modeling and management applications like SQL Server Management Studio (SSMS) or Visual Studio with Analysis Services projects, or with client reporting applications like Microsoft Excel, Power BI Desktop, or custom applications. Connections to Azure Analysis Services use HTTPS.

Client libraries

Get the latest Client libraries

All connections to a server, regardless of type, require updated AMO, ADOMD.NET, and OLEDB client libraries to connect to and interface with an Analysis Services server. For SSMS, Visual Studio, Excel 2016 and later, and Power BI, the latest client libraries are installed or updated with monthly releases. However, in some cases, it's possible an application may not have the latest. For example, when policies delay updates, or Microsoft 365 updates are on the Deferred Channel.

> **NOTE**
> The client libraries cannot connect to Azure Analysis Services through proxy servers that require a username and password.

Server name

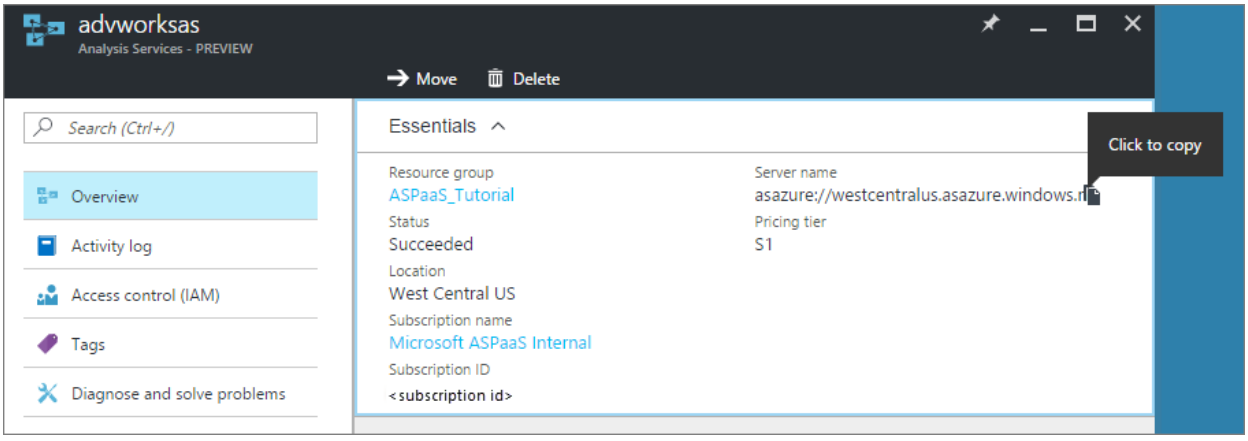
When you create an Analysis Services server in Azure, you specify a unique name and the region where the server is to be created. When specifying the server name in a connection, the server naming scheme is:

```
<protocol>://<region>/<servername>
```

Where protocol is string **asazure**, region is the Uri where the server was created (for example, westus.asazure.windows.net) and servername is the name of your unique server within the region.

Get the server name

In **Azure portal** > server > **Overview** > **Server name**, copy the entire server name. If other users in your organization are connecting to this server too, you can share this server name with them. When specifying a server name, the entire path must be used.



> **NOTE**
> The protocol for East US 2 region is **aspaaseastus2**.

Connection string

When connecting to Azure Analysis Services using the Tabular Object Model, use the following connection string formats:

Integrated Azure Active Directory authentication

Integrated authentication picks up the Azure Active Directory credential cache if available. If not, the Azure login window is shown.

```
"Provider=MSOLAP;Data Source=<Azure AS instance name>;"
```

Azure Active Directory authentication with username and password

```
"Provider=MSOLAP;Data Source=<Azure AS instance name>;User ID=<user name>;Password=<password>;Persist
Security Info=True; Impersonation Level=Impersonate;";
```

Windows authentication (Integrated security)

Use the Windows account running the current process.

```
"Provider=MSOLAP;Data Source=<Azure AS instance name>; Integrated Security=SSPI;Persist Security Info=True;"
```

Connect using an .odc file

With older versions of Excel, users can connect to an Azure Analysis Services server by using an Office Data Connection (.odc) file. To learn more, see Create an Office Data Connection (.odc) file.

Connect as a linked server from SQL Server

SQL Server can connect to an Azure Analysis Services resource as a Linked server by specifying MSOLAP as the data source provider. Before configuring a linked server connection, be sure to install the latest MSOLAP client library (provider).

For linked server connections to Azure Analysis Services, the MSOLAP provider must be instantiated outside the SQL Server process. When configuring linked server options, make sure the **Allow inprocess** option **is not selected**.

If **Allow inprocess** is selected and the provider is instantiated in the SQL Server process, the following error is returned:

```
OLE DB provider "MSOLAP" for linked server "(null)" returned message "The following system error occurred:
".

OLE DB provider "MSOLAP" for linked server "(null)" returned message "The connection failed because user
credentials are needed and Sign-In UI is not allowed.".

Msg 7303, Level 16, State 1, Line 2
Cannot initialize the data source object of OLE DB provider "MSOLAP" for linked server "(null)".
```

Next steps

Connect with Excel
Connect with Power BI
Manage your server

Analysis Services high availability

3/29/2021 • 2 minutes to read • Edit Online

This article describes assuring high availability for Azure Analysis Services servers.

Assuring high availability during a service disruption

While rare, an Azure data center can have an outage. When an outage occurs, it causes a business disruption that might last a few minutes or might last for hours. High availability is most often achieved with server redundancy. With Azure Analysis Services, you can achieve redundancy by creating additional, secondary servers in one or more regions. When creating redundant servers, to assure the data and metadata on those servers is in-sync with the server in a region that has gone offline, you can:

- Deploy models to redundant servers in other regions. This method requires processing data on both your primary server and redundant servers in-parallel, assuring all servers are in-sync.
- Backup databases from your primary server and restore on redundant servers. For example, you can automate nightly backups to Azure storage, and restore to other redundant servers in other regions.

In either case, if your primary server experiences an outage, you must change the connection strings in reporting clients to connect to the server in a different regional datacenter. This change should be considered a last resort and only if a catastrophic regional data center outage occurs. It's more likely a data center outage hosting your primary server would come back online before you could update connections on all clients.

To avoid having to change connection strings on reporting clients, you can create a server alias for your primary server. If the primary server goes down, you can change the alias to point to a redundant server in another region. You can automate alias to server name by coding an endpoint health check on the primary server. If the health check fails, the same endpoint can direct to a redundant server in another region.

Related information

Backup and restore
Manage Azure Analysis Services
Alias server names

Best practices for long running operations

9/15/2021 • 2 minutes to read • Edit Online

In Azure Analysis Services, a *node* represents a host virtual machine where a server resource is running. Some operations such as long running queries, refresh operations, and query scale-out synchronization can fail if a server resource moves to a different node. Common error messages in this scenario include:

- "An error has occurred while trying to locate a long running XMLA request. The request might have been interrupted by service upgrade or server restart."
- "Job with ID '<guid>for model '<database>' was canceled due to service error (inactivity) with message 'Cancelling the refresh request since it was stuck without any updates. This is an internal service issue. Please resubmit the job or file a ticket to get help if this issue happens repeatedly."

There are many reasons why long running operations can be disrupted. For example, updates in Azure such as:

- Operating System patches
- Security updates
- Azure Analysis Services service updates
- Service Fabric updates. Service Fabric is a platform component used by a number of Microsoft cloud services, including Azure Analysis Services.

Besides updates that occur in the service, there is a natural movement of services across nodes due to load balancing. Node movements are an expected part of a cloud service. Azure Analysis Services tries to minimize impacts from node movements, but it's impossible to eliminate them entirely.

In addition to node movements, there are other failures that can occur. For example, a data source database system might be offline or network connectivity is lost. If during refresh, a partition has 10 million rows and a failure occurs at the 9 millionth row, there is no way to restart refresh at the point of failure. The service has to start again from the beginning.

Refresh REST API

Service interruptions can be challenging for long running operations like data refresh. Azure Analysis Services includes a REST API to help mitigate negative impacts from service interruptions. To learn more, see Asynchronous refresh with the REST API.

Besides the REST API, there are other approaches you can use to minimize potential issues during long running refresh operations. The main goal is to avoid having to restart the refresh operation from the beginning, and instead perform refreshes in smaller batches that can be committed in stages.

The REST API allows for such restart, but it doesn't allow for full flexibility of partition creation and deletion. If a scenario requires complex data management operations, your solution should include some form of batching in its logic. For example, by using transactions to process data in multiple, separate batches allows for a failure to not require restart from the beginning, but instead from an intermediate checkpoint.

Scale-out query replicas

Whether using REST or custom logic, client application queries can still return inconsistent or intermediate results while batches are being processed. If consistent data returned by client application queries is required while processing is happening, and model data is in an intermediate state, you can use scale-out with read-only query replicas.

By using read-only query replicas, while refreshes are being performed in batches, client application users can continue to query the old snapshot of data on the read-only replicas. Once refreshes are finished, a Synch operation can be performed to bring the read-only replicas up to date.

Next steps

Asynchronous refresh with the REST API
Azure Analysis Services scale-out
Analysis Services high availability
Retry guidance for Azure services

Data sources supported in Azure Analysis Services

3/29/2021 • 6 minutes to read • Edit Online

Data sources and connectors shown in Get Data or Table Import Wizard in Visual Studio with Analysis Services projects are shown for both Azure Analysis Services and SQL Server Analysis Services. However, not all data sources and connectors shown are supported in Azure Analysis Services. The types of data sources you can connect to depend on many factors such as model compatibility level, available data connectors, authentication type, and On-premises data gateway support. The following tables describe supported data sources for Azure Analysis Services.

Azure data sources

DATA SOURCE	IN-MEMORY	DIRECTQUERY	NOTES
Azure SQL Database	Yes	Yes	2, 3
Azure Synapse Analytics (SQL DW)	Yes	Yes	2
Azure Blob Storage	Yes	No	1
Azure Table Storage	Yes	No	1
Azure Cosmos DB	Yes	No	1
Azure Data Lake Store Gen1	Yes	No	1
Azure Data Lake Store Gen2	Yes	No	1, 5
Azure HDInsight HDFS	Yes	No	1
Azure HDInsight Spark	Yes	No	1, 4

Notes:

1 - Tabular 1400 and higher models only.
2 - When specified as a *provider* data source in tabular 1200 and higher models, both in-memory and DirectQuery models require Microsoft OLE DB Driver for SQL Server MSOLEDBSQL (recommended) or .NET Framework Data Provider for SQL Server.
3 - Azure SQL Managed Instance is supported. Because SQL Managed Instance runs within Azure VNet with a private IP address, public endpoint must be enabled on the instance. If not enabled, an On-premises data gateway is required.
4 - Azure Databricks using the Spark connector is currently not supported.
5 - ADLS Gen2 connector is currently not supported, however, Azure Blob Storage connector can be used with an ADLS Gen2 data source.

Other data sources

DATA SOURCE	IN-MEMORY	DIRECTQUERY	NOTES
Access Database	Yes	No	
Active Directory	Yes	No	[6]
Analysis Services	Yes	No	
Analytics Platform System	Yes	No	
CSV file	Yes	No	
Dynamics 365	Yes	No	[6]
Excel workbook	Yes	No	
Exchange	Yes	No	[6]
Folder	Yes	No	[6]
IBM Informix	Yes	No	
JSON document	Yes	No	[6]
Lines from binary	Yes	No	[6]
MySQL Database	Yes	No	
OData Feed	Yes	No	[6]
ODBC query	Yes	No	
OLE DB	Yes	No	
Oracle	Yes	Yes	[9]
PostgreSQL Database	Yes	No	[6]
Salesforce Objects	Yes	No	[6]
Salesforce Reports	Yes	No	[6]
SAP HANA	Yes	No	
SAP Business Warehouse	Yes	No	[6]
SharePoint List	Yes	No	[6], [11]
SQL Server	Yes	Yes	[7], [8]
SQL Server Data Warehouse	Yes	Yes	[7], [8]

DATA SOURCE	IN-MEMORY	DIRECTQUERY	NOTES
Sybase Database	Yes	No	
Teradata	Yes	Yes	10
TXT file	Yes	No	
XML table	Yes	No	6

Notes:
6 - Tabular 1400 and higher models only.
7 - When specified as a *provider* data source in tabular 1200 and higher models, specify Microsoft OLE DB Driver for SQL Server MSOLEDBSQL (recommended), SQL Server Native Client 11.0, or .NET Framework Data Provider for SQL Server.
8 - If specifying MSOLEDBSQL as the data provider, it may be necessary to download and install the Microsoft OLE DB Driver for SQL Server on the same computer as the On-premises data gateway.
9 - For tabular 1200 models, or as a *provider* data source in tabular 1400+ models, specify Oracle Data Provider for .NET. If specified as a structured data source, be sure to enable Oracle managed provider.
10 - For tabular 1200 models, or as a *provider* data source in tabular 1400+ models, specify Teradata Data Provider for .NET.
11 - Files in on-premises SharePoint are not supported.

Connecting to on-premises data sources from an Azure Analysis Services server require an On-premises gateway. When using a gateway, 64-bit providers are required.

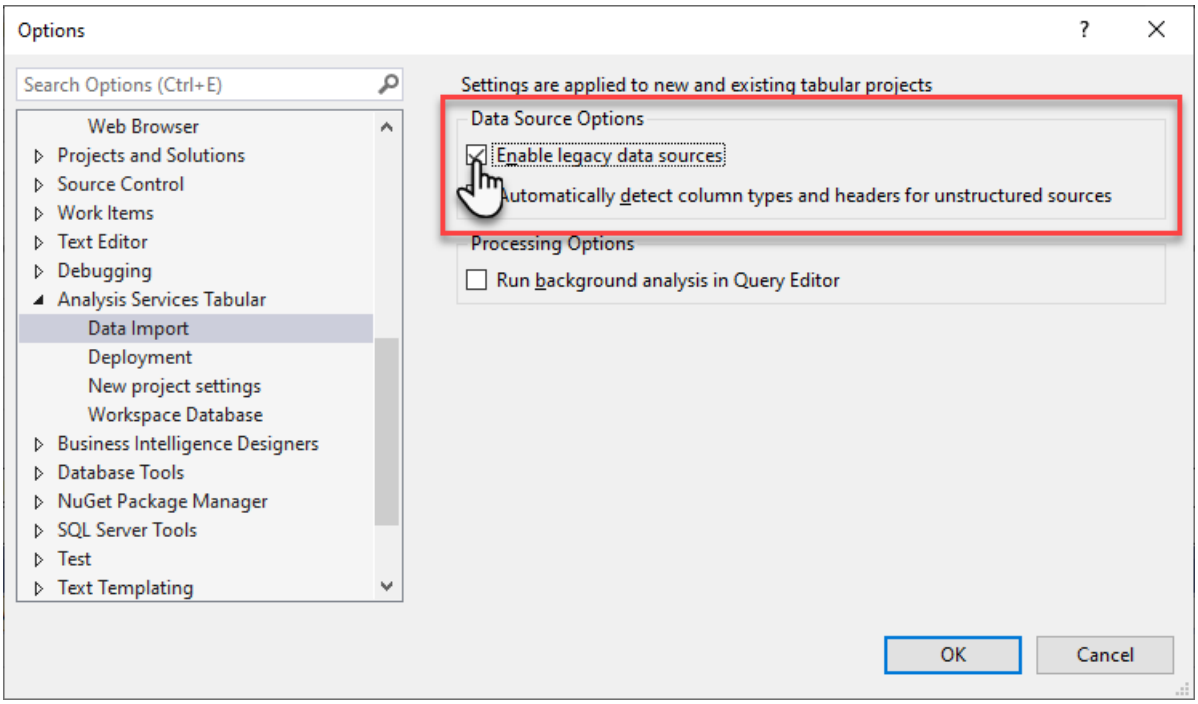
Understanding providers

When creating tabular 1400 and higher model projects in Visual Studio, by default you do not specify a data provider when connecting to a data source by using **Get Data**. Tabular 1400 and higher models use Power Query connectors to manage connections, data queries, and mashups between the data source and Analysis Services. These are sometimes referred to as *structured* data source connections in that connection property settings are set for you. You can, however, enable legacy data sources for a model project in Visual Studio. When enabled, you can use **Table Import Wizard** to connect to certain data sources traditionally supported in tabular 1200 and lower models as *legacy*, or *provider* data sources. When specified as a provider data source, you can specify a particular data provider and other advanced connection properties. For example, you can connect to a SQL Server Data Warehouse instance or even an Azure SQL Database as a legacy data source. You can then select the OLE DB Driver for SQL Server MSOLEDBSQL data provider. In this case, selecting an OLE DB data provider may provide improved performance over the Power Query connector.

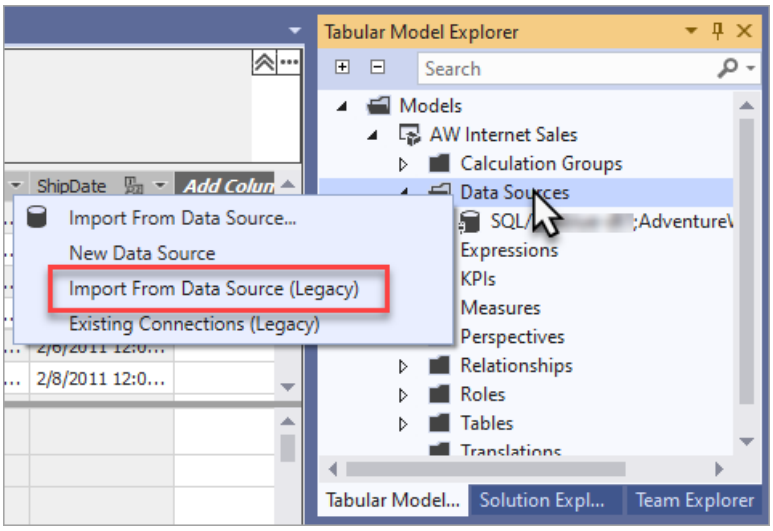
When using the Table Import Wizard in Visual Studio, connections to any data source require a data provider. A default data provider is selected for you. You can change the data provider if needed. The type of provider you choose can depend on performance, whether or not the model is using in-memory storage or DirectQuery, and which Analysis Services platform you deploy your model to.

Specify provider data sources in tabular 1400 and higher model projects

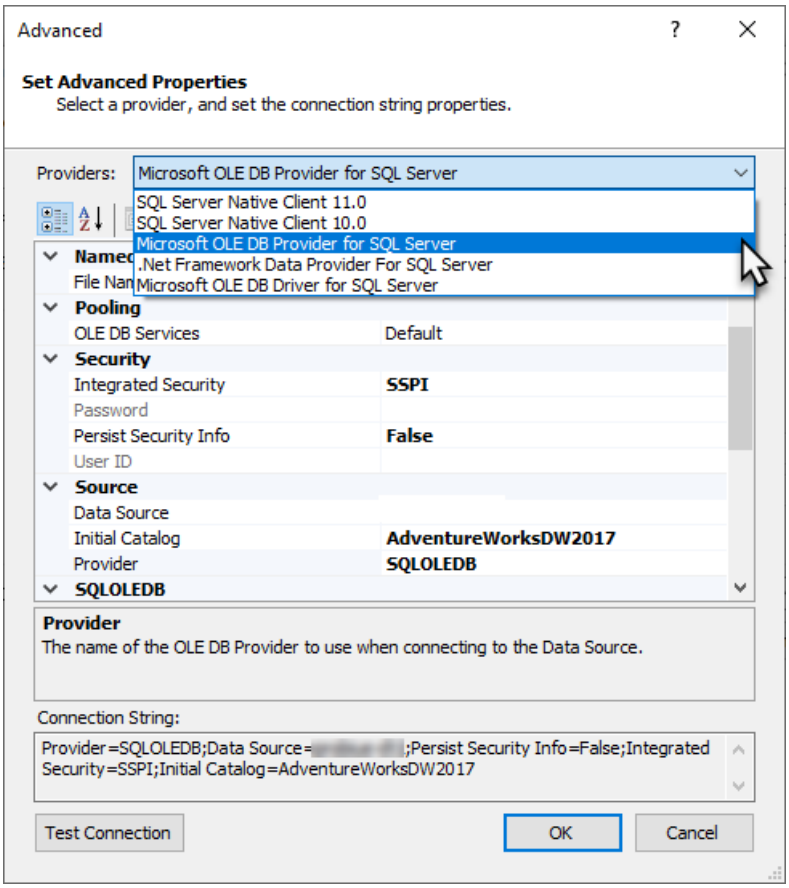
To enable provider data sources, in Visual Studio, click **Tools** > **Options** > **Analysis Services Tabular** > **Data Import**, select **Enable legacy data sources**.



With legacy data sources enabled, in **Tabular Model Explorer**, right-click **Data Sources** > **Import From Data Source (Legacy)**.



Just like with tabular 1200 model projects, use **Table Import Wizard** to connect to a data source. On the connect page, click **Advanced**. Specify data provider and other connection settings in **Set Advanced Properties**.



Impersonation

In some cases, it may be necessary to specify a different impersonation account. Impersonation account can be specified in Visual Studio or SQL Server Management Studio (SSMS).

For on-premises data sources:

- If using SQL authentication, impersonation should be Service Account.
- If using Windows authentication, set Windows user/password. For SQL Server, Windows authentication with a specific impersonation account is supported only for in-memory data models.

For cloud data sources:

- If using SQL authentication, impersonation should be Service Account.

OAuth credentials

For tabular models at the 1400 and higher compatibility level using *in-memory* mode, Azure SQL Database, Azure Synapse, Dynamics 365, and SharePoint List support OAuth credentials. To generate valid tokens, set credentials by using Power Query. Azure Analysis Services manages token refresh for OAuth data sources to avoid timeouts for long-running refresh operations.

> **NOTE**
>
> Managed token refresh is not supported for data sources accessed through a gateway. For example, one or more mashup query data sources is accessed through a gateway, and/or the ASPaaS\AlwaysUseGateway property is set to **true**.

Direct Query mode is not supported with OAuth credentials.

Enable Oracle managed provider

In some cases, DAX queries to an Oracle data source may return unexpected results. This can be due to the provider being used for the data source connection.

As described in the Understanding providers section, tabular models connect to data sources as either a *structured* data source or a *provider* data source. For models with an Oracle data source specified as a provider data source, ensure the specified provider is Oracle Data Provider for .NET (Oracle.DataAccess.Client).

If the Oracle data source is specified as a structured data source, enable the **MDataEngine\UseManagedOracleProvider** server property. Setting this property ensures your model connects to the Oracle data source using the recommended Oracle Data Provider for .NET managed provider.

To enable Oracle managed provider:

1. In SQL Server Management Studio, connect to your server.
2. Create an XMLA query with the following script. Replace **ServerName** with the full server name, and then execute the query.

```
<Alter AllowCreate="true" ObjectExpansion="ObjectProperties"
xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
    <Object />
    <ObjectDefinition>
        <Server xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/engine/2"
xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/2"
xmlns:ddl100_100="http://schemas.microsoft.com/analysisservices/2008/engine/100/100"
xmlns:ddl200="http://schemas.microsoft.com/analysisservices/2010/engine/200"
xmlns:ddl200_200="http://schemas.microsoft.com/analysisservices/2010/engine/200/200"
xmlns:ddl300="http://schemas.microsoft.com/analysisservices/2011/engine/300"
xmlns:ddl300_300="http://schemas.microsoft.com/analysisservices/2011/engine/300/300"
xmlns:ddl400="http://schemas.microsoft.com/analysisservices/2012/engine/400"
xmlns:ddl400_400="http://schemas.microsoft.com/analysisservices/2012/engine/400/400"
xmlns:ddl500="http://schemas.microsoft.com/analysisservices/2013/engine/500"
xmlns:ddl500_500="http://schemas.microsoft.com/analysisservices/2013/engine/500/500">
            <ID>ServerName</ID>
            <Name>ServerName</Name>
            <ServerProperties>
                <ServerProperty>
                    <Name>MDataEngine\UseManagedOracleProvider</Name>
                    <Value>1</Value>
                </ServerProperty>
            </ServerProperties>
        </Server>
    </ObjectDefinition>
</Alter>
```

3. Restart the server.

Next steps

- On-premises gateway
- Manage your server

Analysis Services database backup and restore

6/28/2021 • 2 minutes to read • Edit Online

Backing up tabular model databases in Azure Analysis Services is much the same as for on-premises Analysis Services. The primary difference is where you store your backup files. Backup files must be saved to a container in an Azure storage account. You can use a storage account and container you already have, or they can be created when configuring storage settings for your server.

> **NOTE**
> Creating a storage account can result in a new billable service. To learn more, see Azure Storage Pricing.

> **NOTE**
> If the storage account is in a different region, configure storage account firewall settings to allow access from **Selected networks**. In Firewall **Address range**, specify the IP address range for the region the Analysis Services server is in. Configuring storage account firewall settings to allow access from All networks is supported, however choosing Selected networks and specifying an IP address range is preferred. To learn more, see Network connectivity FAQ.

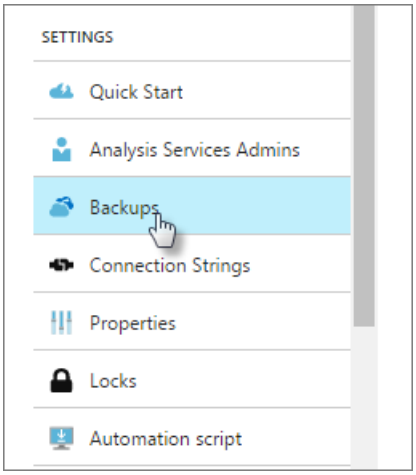
Backups are saved with an .abf extension. For in-memory tabular models, both model data and metadata are stored. For DirectQuery tabular models, only model metadata is stored. Backups can be compressed and encrypted, depending on the options you choose.

Configure storage settings

Before backing up, you need to configure storage settings for your server.

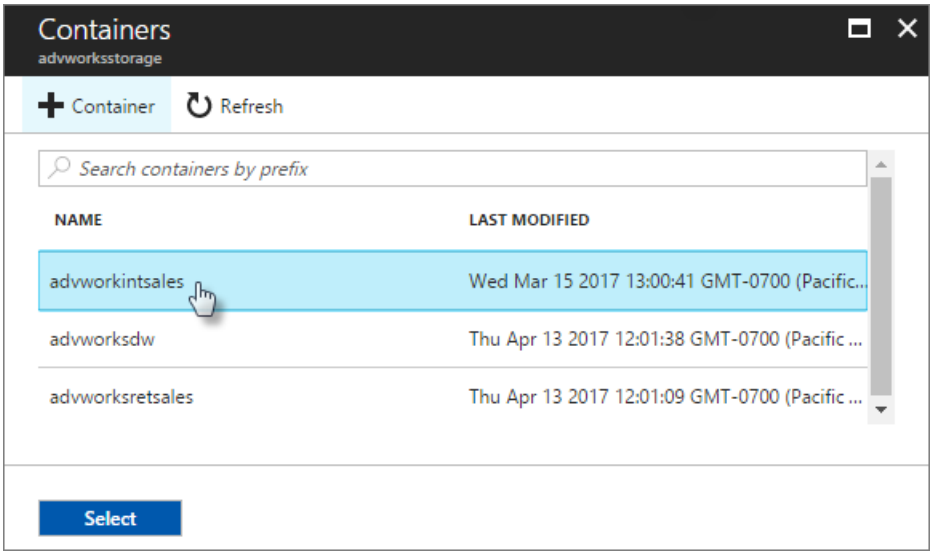
To configure storage settings

1. In Azure portal > **Settings**, click **Backup**.



2. Click **Enabled**, then click **Storage Settings**.

3. Select your storage account or create a new one.
4. Select a container or create a new one.



5. Save your backup settings.

Backup

To backup by using SSMS

1. In SSMS, right-click a database > **Back Up**.
2. In **Backup Database** > **Backup file**, click **Browse**.
3. In the **Save file as** dialog, verify the folder path, and then type a name for the backup file.
4. In the **Backup Database** dialog, select options.

   Allow file overwrite - Select this option to overwrite backup files of the same name. If this option is not selected, the file you are saving cannot have the same name as a file that already exists in the same location.

   Apply compression - Select this option to compress the backup file. Compressed backup files save disk space, but require slightly higher CPU utilization.

   Encrypt backup file - Select this option to encrypt the backup file. This option requires a user-supplied password to secure the backup file. The password prevents reading of the backup data any other means

than a restore operation. If you choose to encrypt backups, store the password in a safe location.

5. Click **OK** to create and save the backup file.

PowerShell

Use Backup-ASDatabase cmdlet.

Restore

When restoring, your backup file must be in the storage account you've configured for your server. If you need to move a backup file from an on-premises location to your storage account, use Microsoft Azure Storage Explorer or the AzCopy command-line utility.

> **NOTE**
> If you're restoring from an on-premises server, you must remove all the domain users from the model's roles and add them back to the roles as Azure Active Directory users.

To restore by using SSMS

1. In SSMS, right-click a database > **Restore**.
2. In the **Backup Database** dialog, in **Backup file**, click **Browse**.
3. In the **Locate Database Files** dialog, select the file you want to restore.
4. In **Restore database**, select the database.
5. Specify options. Security options must match the backup options you used when backing up.

PowerShell

Use Restore-ASDatabase cmdlet.

Related information

Azure storage accounts
High availability
Analysis Services network connectivity FAQ

Alias server names

3/5/2021 • 2 minutes to read • Edit Online

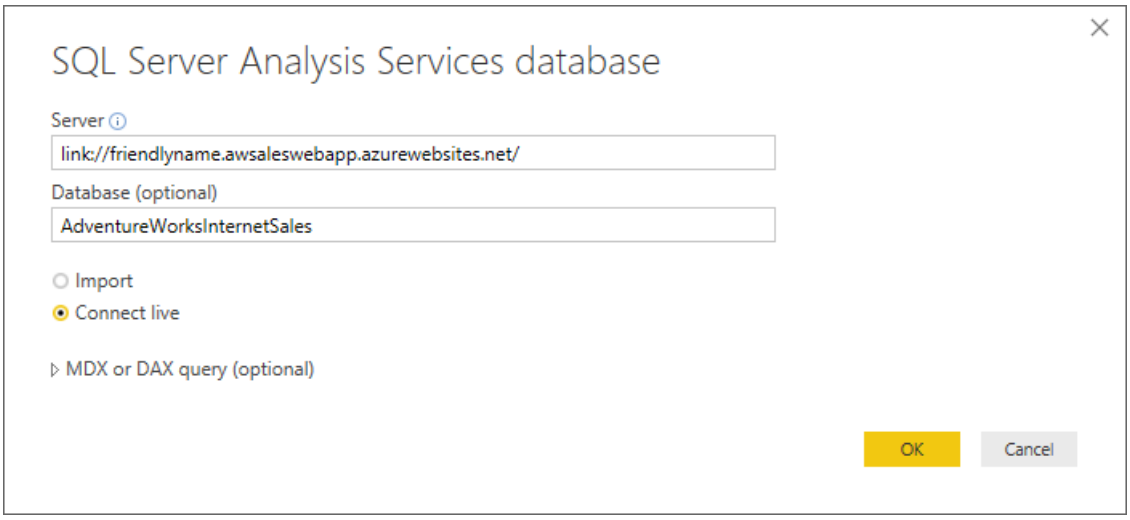
By using a server name alias, users can connect to your Azure Analysis Services server with a shorter *alias* instead of the server name. When connecting from a client application, the alias is specified as an endpoint using the **link://** protocol format. The endpoint then returns the real server name in order to connect.

Alias server names are good for:

- Migrating models between servers without affecting users.
- Friendly server names are easier for users to remember.
- Direct users to different servers at different times of the day.
- Direct users in different regions to instances that are geographically closer, like when using Azure Traffic Manager.

Any HTTPS endpoint that returns a valid Azure Analysis Services server name can serve as an alias. The endpoint must support HTTPS over port 443 and the port must not be specified in the URI.

When connecting from a client, the alias server name is entered using **link://** protocol format. For example, in Power BI Desktop:



Create an alias

To create an alias endpoint, you can use any method that returns a valid Azure Analysis Services server name. For example, a reference to a file in Azure Blob Storage containing the real server name, or create and publish an ASP.NET Web Forms application.

In this example, an ASP.NET Web Forms Application is created in Visual Studio. The page reference and user control are removed from the Default.aspx page. The contents of Default.aspx are simply the following Page directive:

```
<%@ Page Title="Home Page" Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="FriendlyRedirect._Default" %>
```

The Page_Load event in Default.aspx.cs uses the Response.Write() method to return the Azure Analysis Services server name.

```
protected void Page_Load(object sender, EventArgs e)
{
    this.Response.Write("asazure://<region>.asazure.windows.net/<servername>");
}
```

See also

Client libraries
Connect from Power BI Desktop

Azure Analysis Services scale-out

4/27/2021 • 11 minutes to read • Edit Online

With scale-out, client queries can be distributed among multiple *query replicas* in a *query pool*, reducing response times during high query workloads. You can also separate processing from the query pool, ensuring client queries are not adversely affected by processing operations. Scale-out can be configured in Azure portal or by using the Analysis Services REST API.

Scale-out is available for servers in the Standard pricing tier. Each query replica is billed at the same rate as your server. All query replicas are created in the same region as your server. The number of query replicas you can configure are limited by the region your server is in. To learn more, see Availability by region. Scale-out does not increase the amount of available memory for your server. To increase memory, you need to upgrade your plan.

Why scale out?

In a typical server deployment, one server serves as both processing server and query server. If the number of client queries against models on your server exceeds the Query Processing Units (QPU) for your server's plan, or model processing occurs at the same time as high query workloads, performance can decrease.

With scale-out, you can create a query pool with up to seven additional query replica resources (eight total, including your *primary* server). You can scale the number of replicas in the query pool to meet QPU demands at critical times, and you can separate a processing server from the query pool at any time.

Regardless of the number of query replicas you have in a query pool, processing workloads are not distributed among query replicas. The primary server serves as the processing server. Query replicas serve only queries against the model databases synchronized between the primary server and each replica in the query pool.

When scaling out, it can take up to five minutes for new query replicas to be incrementally added to the query pool. When all new query replicas are up and running, new client connections are load balanced across resources in the query pool. Existing client connections are not changed from the resource they are currently connected to. When scaling in, any existing client connections to a query pool resource that is being removed from the query pool are terminated. Clients can reconnect to a remaining query pool resource.

How it works

When configuring scale-out the first time, model databases on your primary server are *automatically* synchronized with new replicas in a new query pool. Automatic synchronization occurs only once. During automatic synchronization, the primary server's data files (encrypted at rest in blob storage) are copied to a second location, also encrypted at rest in blob storage. Replicas in the query pool are then *hydrated* with data from the second set of files.

While an automatic synchronization is performed only when you scale-out a server for the first time, you can also perform a manual synchronization. Synchronizing assures data on replicas in the query pool match that of the primary server. When processing (refresh) models on the primary server, a synchronization must be performed *after* processing operations are completed. This synchronization copies updated data from the primary server's files in blob storage to the second set of files. Replicas in the query pool are then hydrated with updated data from the second set of files in blob storage.

When performing a subsequent scale-out operation, for example, increasing the number of replicas in the query pool from two to five, the new replicas are hydrated with data from the second set of files in blob storage. There is no synchronization. If you then perform a synchronization after scaling out, the new replicas in the query pool

would be hydrated twice - a redundant hydration. When performing a subsequent scale-out operation, it's important to keep in mind:

- Perform a synchronization *before the scale-out operation* to avoid redundant hydration of the added replicas. Concurrent synchronization and scale-out operations running at the same time are not allowed.
- When automating both processing *and* scale-out operations, it's important to first process data on the primary server, then perform a synchronization, and then perform the scale-out operation. This sequence assures minimal impact on QPU and memory resources.
- During scale-out operations, all servers in the query pool, including the primary server, are temporarily offline.
- Synchronization is allowed even when there are no replicas in the query pool. If you are scaling out from zero to one or more replicas with new data from a processing operation on the primary server, perform the synchronization first with no replicas in the query pool, and then scale-out. Synchronizing before scaling out avoids redundant hydration of the newly added replicas.
- When deleting a model database from the primary server, it does not automatically get deleted from replicas in the query pool. You must perform a synchronization operation by using the Sync-AzAnalysisServicesInstance PowerShell command that removes the file/s for that database from the replica's shared blob storage location and then deletes the model database on the replicas in the query pool. To determine if a model database exists on replicas in the query pool but not on the primary server, ensure the **Separate the processing server from querying pool** setting is to **Yes**. Then use SSMS to connect to the primary server using the `:rw` qualifier to see if the database exists. Then connect to replicas in the query pool by connecting without the `:rw` qualifier to see if the same database also exists. If the database exists on replicas in the query pool but not on the primary server, run a sync operation.
- When renaming a database on the primary server, there's an additional step necessary to ensure the database is properly synchronized to any replicas. After renaming, perform a synchronization by using the Sync-AzAnalysisServicesInstance command specifying the `-Database` parameter with the old database name. This synchronization removes the database and files with the old name from any replicas. Then perform another synchronization specifying the `-Database` parameter with the new database name. The second synchronization copies the newly named database to the second set of files and hydrates any replicas. These synchronizations cannot be performed by using the Synchronize model command in the portal.

Synchronization mode

By default, query replicas are rehydrated in full, not incrementally. Rehydration happens in stages. They are detached and attached two at a time (assuming there are at least three replicas) to ensure at least one replica is kept online for queries at any given time. In some cases, clients may need to reconnect to one of the online replicas while this process is taking place. By using the **ReplicaSyncMode** setting, you can now specify query replica synchronization occurs in parallel. Parallel synchronization provides the following benefits:

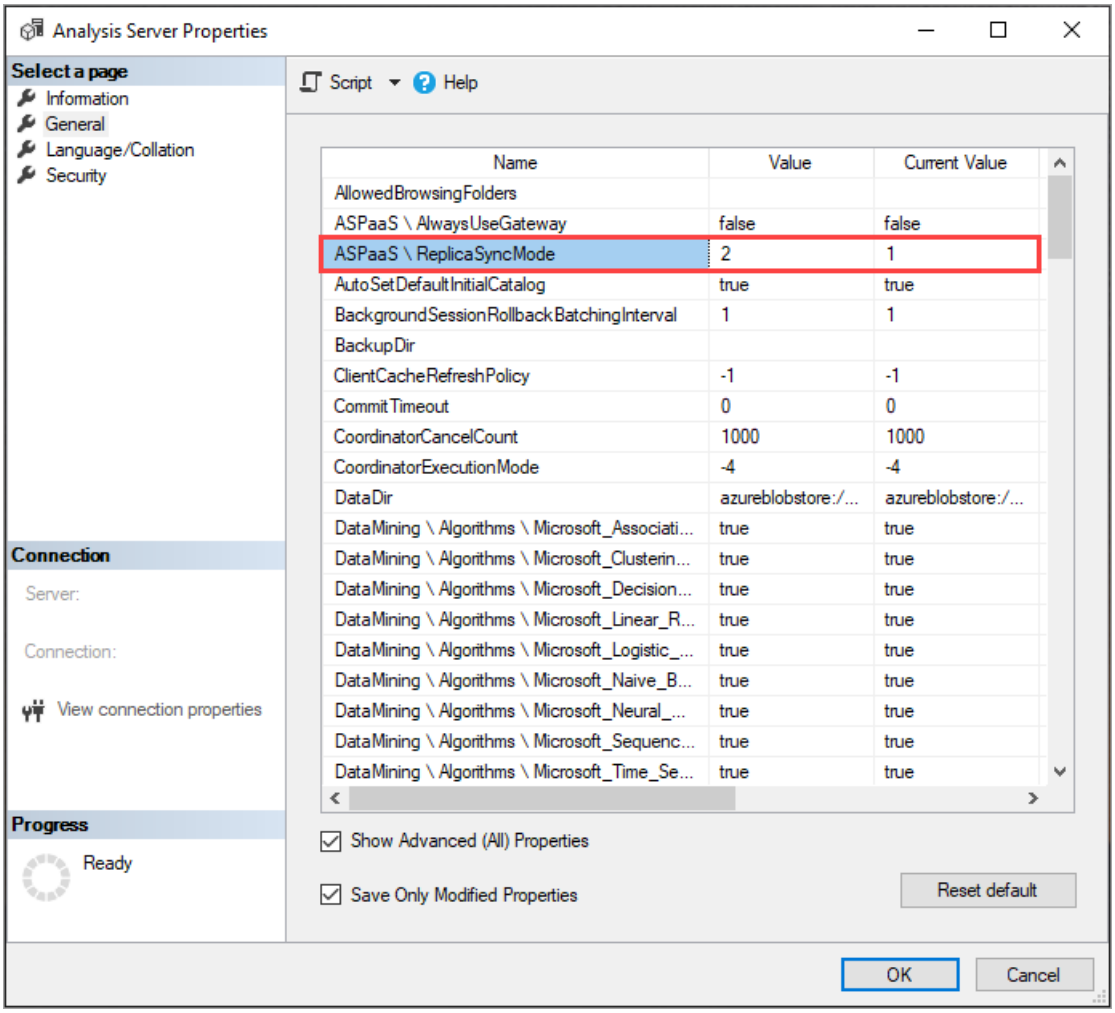
- Significant reduction in synchronization time.
- Data across replicas are more likely to be consistent during the synchronization process.
- Because databases are kept online on all replicas throughout the synchronization process, clients do not need to reconnect.
- The in-memory cache is updated incrementally with only the changed data, which can be faster than fully rehydrating the model.

Setting ReplicaSyncMode

Use SSMS to set ReplicaSyncMode in Advanced Properties. The possible values are:

- `1` (default): Full replica database rehydration in stages (incremental).

- `2`: Optimized synchronization in parallel.



When setting **ReplicaSyncMode=2**, depending on how much of the cache needs to be updated, additional memory may be consumed by the query replicas. To keep the database online and available for queries, depending on how much of the data has changed, the operation can require up to *double the memory* on the replica because both the old and new segments are kept in memory simultaneously. Replica nodes have the same memory allocation as the primary node, and there is normally extra memory on the primary node for refresh operations, so it may be unlikely that the replicas would run out of memory. Additionally, the common scenario is that the database is incrementally updated on the primary node, and therefore the requirement for double the memory should be uncommon. If the Sync operation does encounter an out of memory error, it will retry using the default technique (attach/detach two at a time).

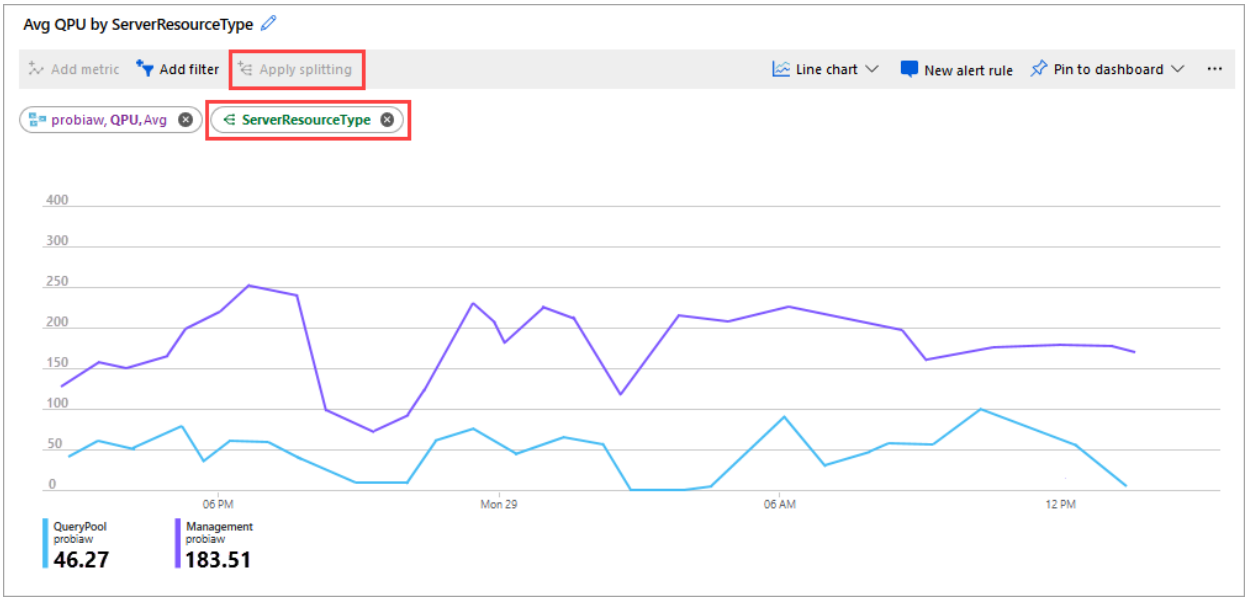
Separate processing from query pool

For maximum performance for both processing and query operations, you can choose to separate your processing server from the query pool. When separated, new client connections are assigned to query replicas in the query pool only. If processing operations only take up a short amount of time, you can choose to separate your processing server from the query pool only for the amount of time it takes to perform processing and synchronization operations, and then include it back into the query pool. When separating the processing server from the query pool, or adding it back into the query pool can take up to five minutes for the operation to complete.

Monitor QPU usage

To determine if scale-out for your server is necessary, monitor your server in Azure portal by using Metrics. If your QPU regularly maxes out, it means the number of queries against your models is exceeding the QPU limit for your plan. The Query pool job queue length metric also increases when the number of queries in the query thread pool queue exceeds available QPU.

Another good metric to watch is average QPU by ServerResourceType. This metric compares average QPU for the primary server with the query pool.



To configure QPU by ServerResourceType

1. In a Metrics line chart, click **Add metric**.
2. In **RESOURCE**, select your server, then in **METRIC NAMESPACE**, select **Analysis Services standard metrics**, then in **METRIC**, select **QPU**, and then in **AGGREGATION**, select **Avg**.
3. Click **Apply Splitting**.
4. In **VALUES**, select **ServerResourceType**.

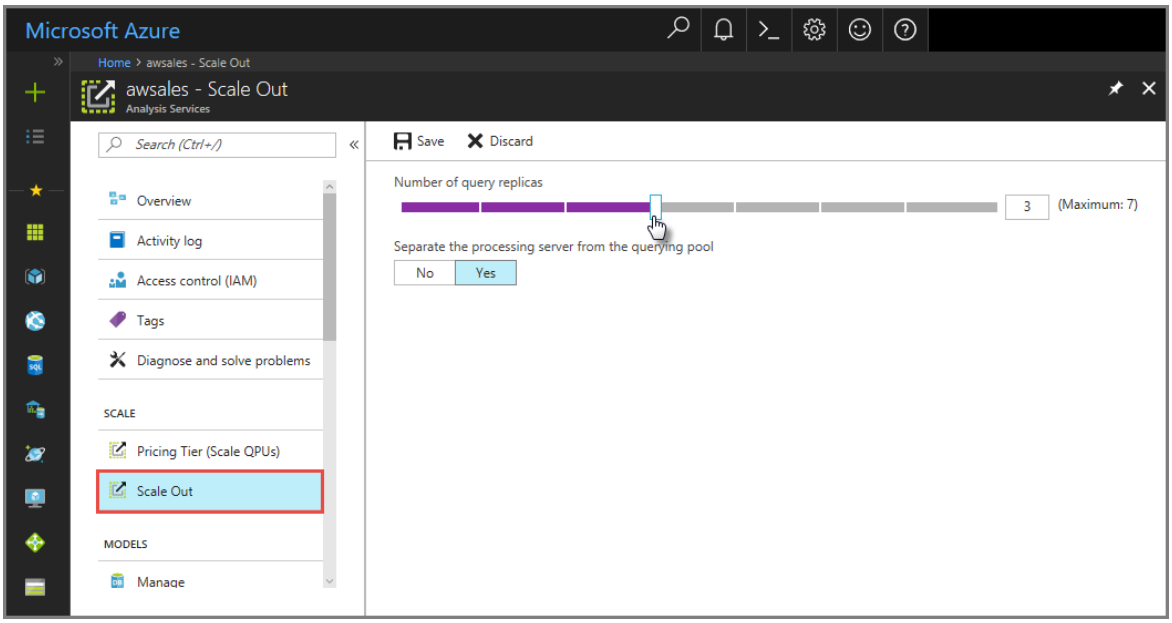
Detailed diagnostic logging

Use Azure Monitor Logs for more detailed diagnostics of scaled out server resources. With logs, you can use Log Analytics queries to break out QPU and memory by server and replica. To learn more, see example queries in Analysis Services diagnostics logging.

Configure scale-out

In Azure portal

1. In the portal, click **Scale-out**. Use the slider to select the number of query replica servers. The number of replicas you choose is in addition to your existing server.
2. In **Separate the processing server from the querying pool**, select yes to exclude your processing server from query servers. Client connections using the default connection string (without `:rw`) are redirected to replicas in the query pool.



3. Click **Save** to provision your new query replica servers.

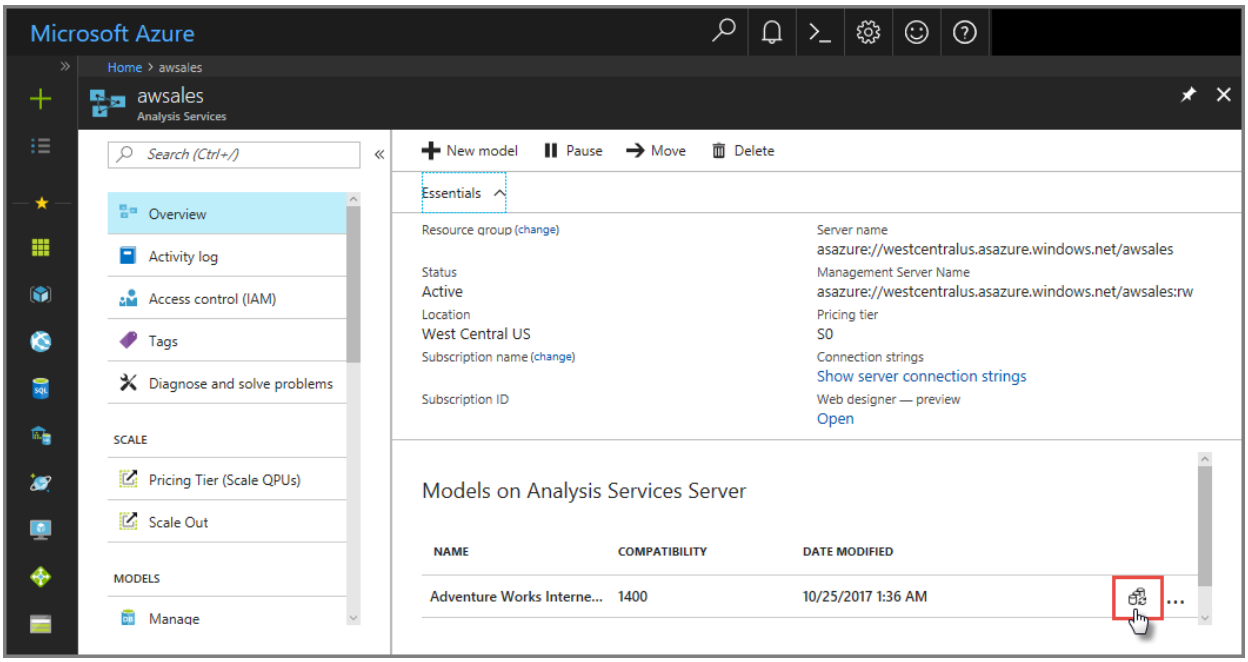
When configuring scale-out for a server the first time, models on your primary server are automatically synchronized with replicas in the query pool. Automatic synchronization only occurs once, when you first configure scale-out to one or more replicas. Subsequent changes to the number of replicas on the same server *will not trigger another automatic synchronization*. Automatic synchronization will not occur again even if you set the server to zero replicas and then again scale-out to any number of replicas.

Synchronize

Synchronization operations must be performed manually or by using the REST API.

In Azure portal

In **Overview** > model > **Synchronize model**.



REST API

Use the **sync** operation.

Synchronize a model

```
POST https://<region>.asazure.windows.net/servers/<servername>:rw/models/<modelname>/sync
```

Get sync status

```
GET https://<region>.asazure.windows.net/servers/<servername>/models/<modelname>/sync
```

Return status codes:

CODE	DESCRIPTION
-1	Invalid
0	Replicating
1	Rehydrating
2	Completed
3	Failed
4	Finalizing

PowerShell

> **NOTE**
> This article has been updated to use the Azure Az PowerShell module. The Az PowerShell module is the recommended PowerShell module for interacting with Azure. To get started with the Az PowerShell module, see Install Azure PowerShell. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Before using PowerShell, install or update the latest Azure PowerShell module.

To run sync, use Sync-AzAnalysisServicesInstance.

To set the number of query replicas, use Set-AzAnalysisServicesServer. Specify the optional `-ReadonlyReplicaCount` parameter.

To separate the processing server from the query pool, use Set-AzAnalysisServicesServer. Specify the optional `-DefaultConnectionMode` parameter to use `Readonly`.

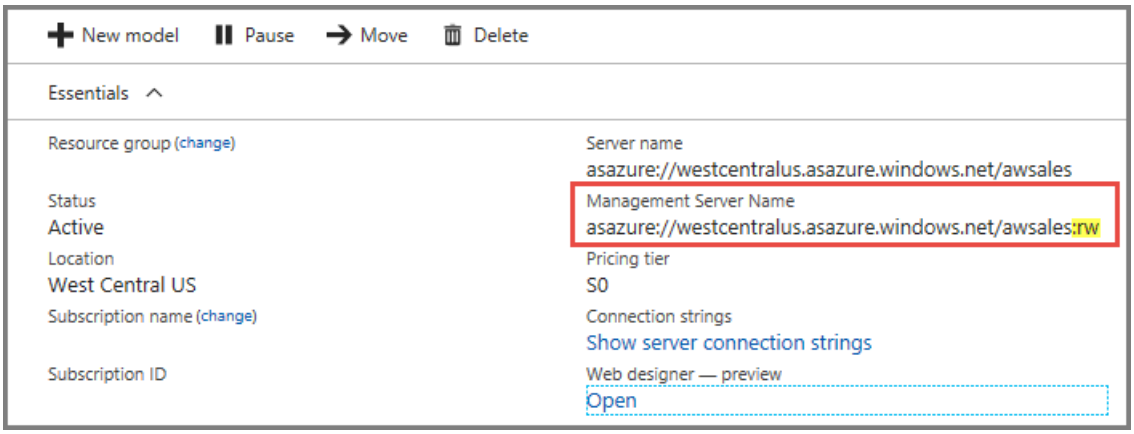
To learn more, see Using a service principal with the Az.AnalysisServices module.

Connections

On your server's Overview page, there are two server names. If you haven't yet configured scale-out for a server, both server names work the same. Once you configure scale-out for a server, you need to specify the appropriate server name depending on the connection type.

For end-user client connections like Power BI Desktop, Excel, and custom apps, use **Server name**.

For SSMS, Visual Studio, and connection strings in PowerShell, Azure Function apps, and AMO, use **Management server name**. The management server name includes a special `:rw` (read-write) qualifier. All processing operations occur on the (primary) management server.



Scale-up, Scale-down vs. Scale-out

You can change the pricing tier on a server with multiple replicas. The same pricing tier applies to all replicas. A scale operation will first bring down all replicas all at once then bring up all replicas on the new pricing tier.

Troubleshoot

Issue: Users get error **Cannot find server '<Name of the server>' instance in connection mode 'ReadOnly'.**

Solution: When selecting the **Separate the processing server from the querying pool** option, client connections using the default connection string (without `:rw`) are redirected to query pool replicas. If replicas in the query pool are not yet online because synchronization has not yet been completed, redirected client connections can fail. To prevent failed connections, there must be at least two servers in the query pool when performing a synchronization. Each server is synchronized individually while others remain online. If you choose to not have the processing server in the query pool during processing, you can choose to remove it from the pool for processing, and then add it back into the pool after processing is complete, but prior to synchronization. Use Memory and QPU metrics to monitor synchronization status.

Related information

Monitor server metrics
Manage Azure Analysis Services

Install and configure an on-premises data gateway

6/28/2021 • 4 minutes to read • Edit Online

An on-premises data gateway is required when one or more Azure Analysis Services servers in the same region connect to on-premises data sources. While the gateway you install is the same as used by other services like Power BI, Power Apps, and Logic Apps, when installing for Azure Analysis Services, there are some additional steps you need to complete. This install article is specific to **Azure Analysis Services**.

To learn more about how Azure Analysis Services works with the gateway, see Connecting to on-premises data sources. To learn more about advanced installation scenarios and the gateway in general, see On-premises data gateways documentation.

Prerequisites

Minimum Requirements:

- .NET 4.5 Framework
- 64-bit version of Windows 8 / Windows Server 2012 R2 (or later)

Recommended:

- 8 Core CPU
- 8 GB Memory
- 64-bit version of Windows 8 / Windows Server 2012 R2 (or later)

Important considerations:

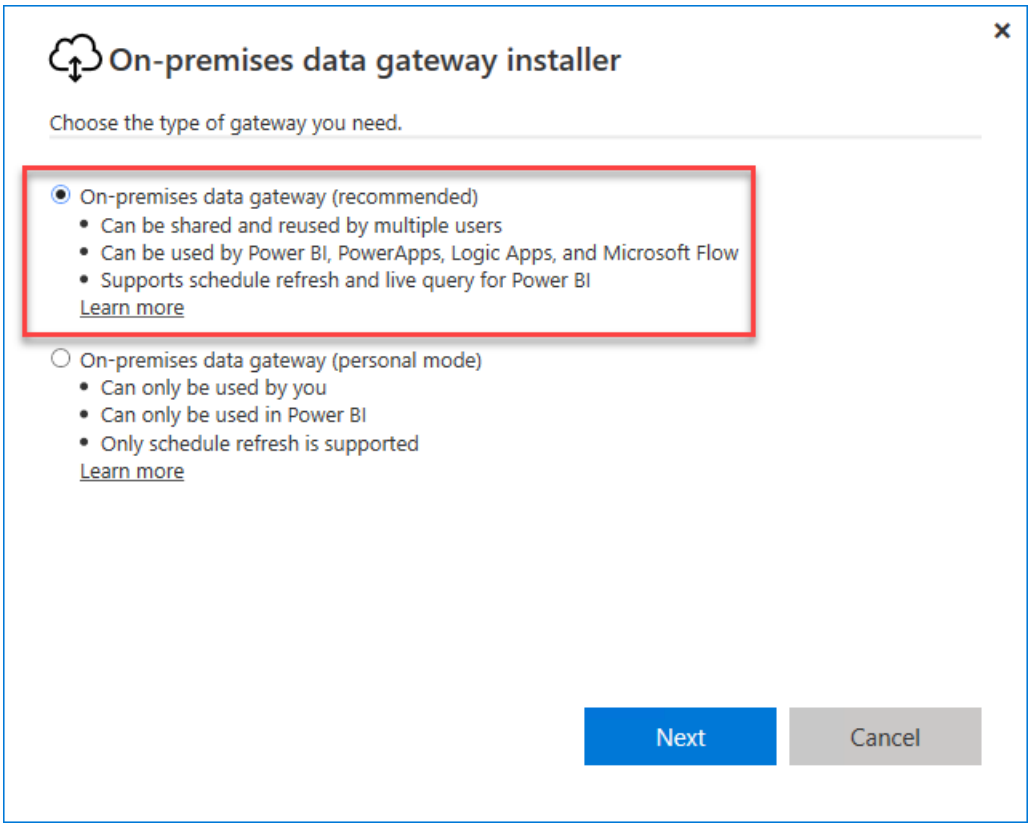
- During setup, when registering your gateway with Azure, the default region for your subscription is selected. You can choose a different subscription and region. If you have servers in more than one region, you must install a gateway for each region.
- The gateway cannot be installed on a domain controller.
- Only one gateway can be installed on a single computer.
- Install the gateway on a computer that remains on and does not go to sleep.
- Do not install the gateway on a computer with a wireless only connection to your network. Performance can be diminished.
- When installing the gateway, the user account you're signed in to your computer with must have Log on as service privileges. When install is complete, the On-premises data gateway service uses the NT SERVICE\PBIEgwService account to log on as a service. A different account can be specified during setup or in Services after setup is complete. Ensure Group Policy settings allow both the account you're signed in with when installing and the service account you choose have Log on as service privileges.
- Sign in to Azure with an account in Azure AD for the same tenant as the subscription you are registering the gateway in. Azure B2B (guest) accounts are not supported when installing and registering a gateway.
- If data sources are on an Azure Virtual Network (VNet), you must configure the AlwaysUseGateway server property.

Download

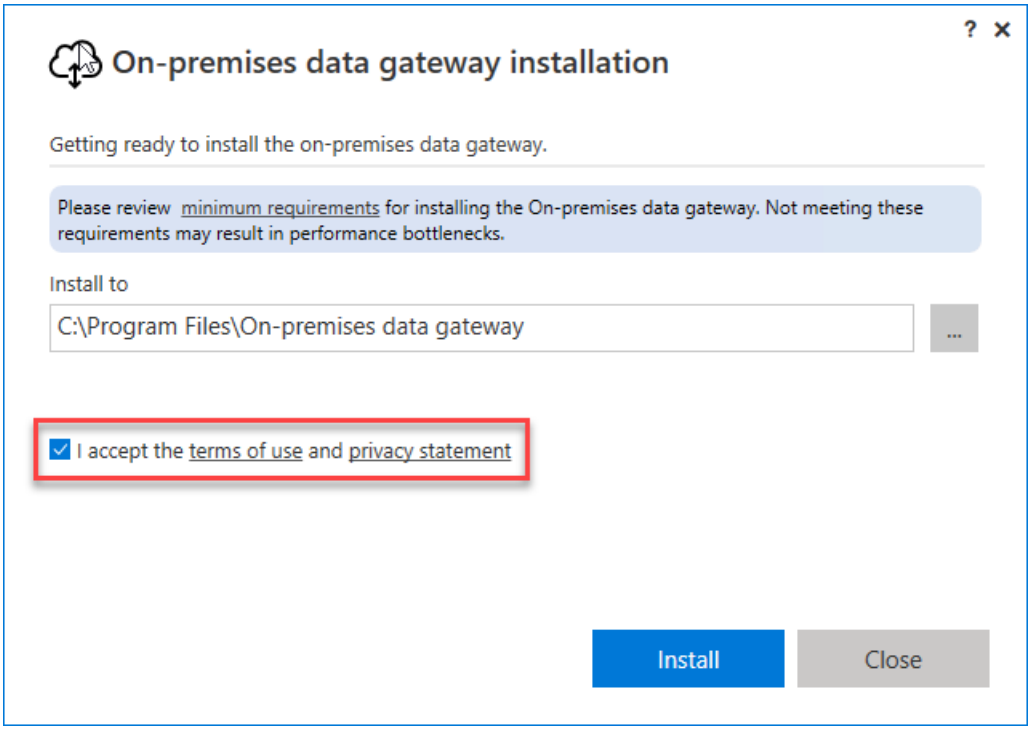
Download the gateway

Install

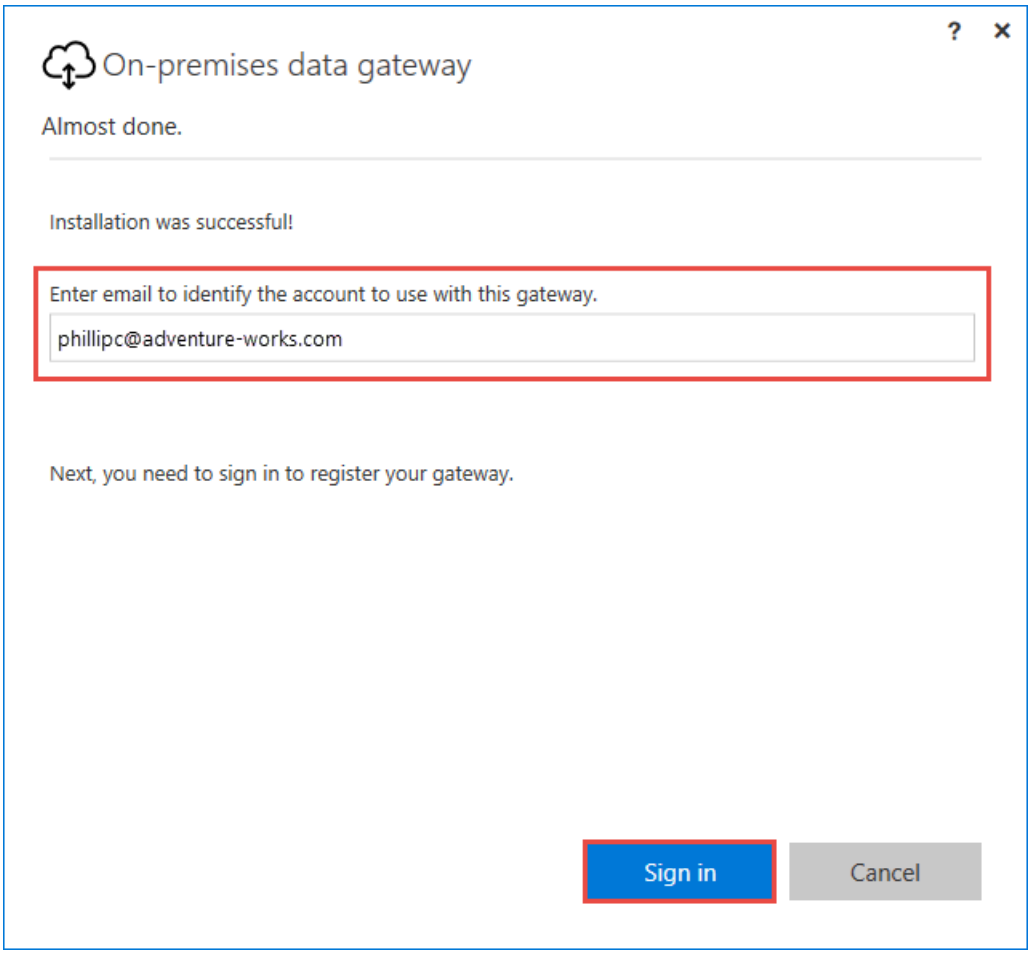
1. Run setup.
2. Select **On-premises data gateway**.



3. Select a location, accept the terms, and then click **Install**.



4. Sign in to Azure. The account must be in your tenant's Azure Active Directory. This account is used for the gateway administrator. Azure B2B (guest) accounts are not supported when installing and registering the gateway.



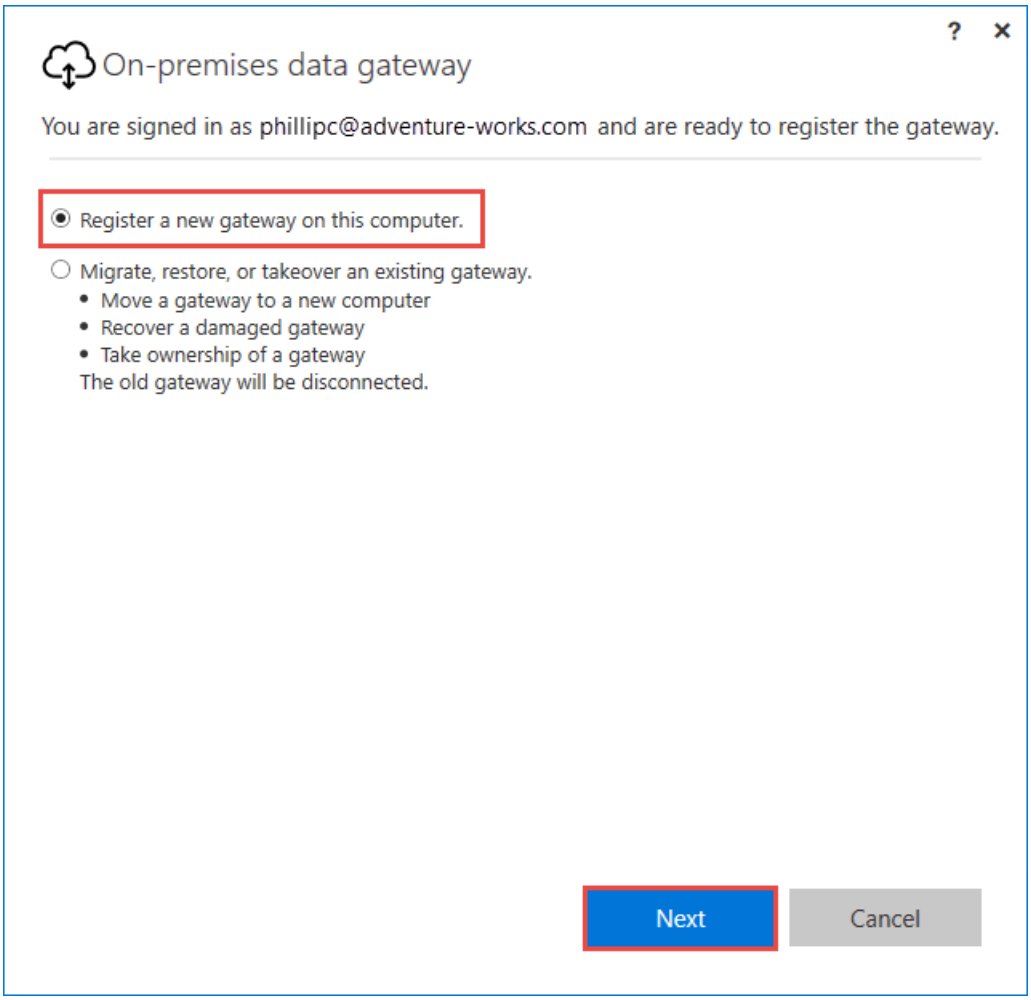
NOTE

If you sign in with a domain account, it's mapped to your organizational account in Azure AD. Your organizational account is used as the gateway administrator.

Register

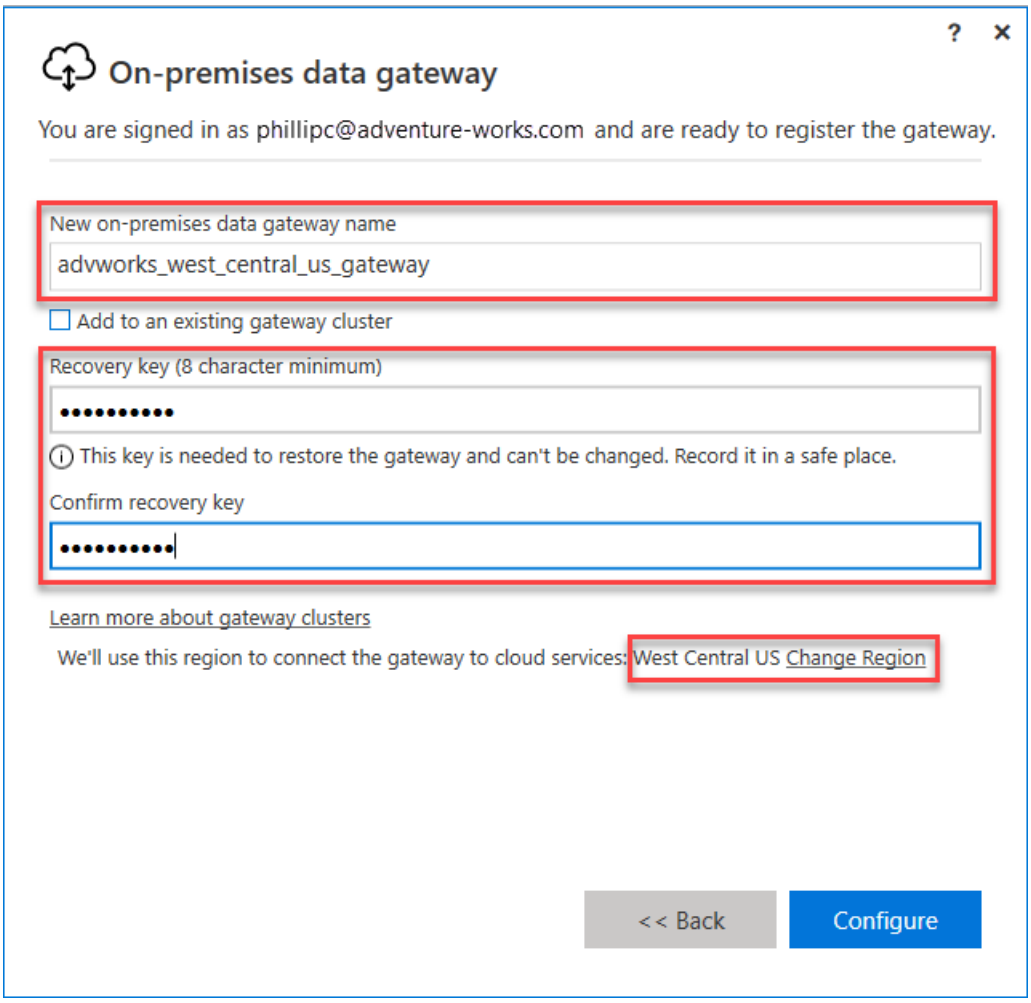
In order to create a gateway resource in Azure, you must register the local instance you installed with the Gateway Cloud Service.

1. Select **Register a new gateway on this computer**.



2. Type a name and recovery key for your gateway. By default, the gateway uses your subscription's default region. If you need to select a different region, select **Change Region**.

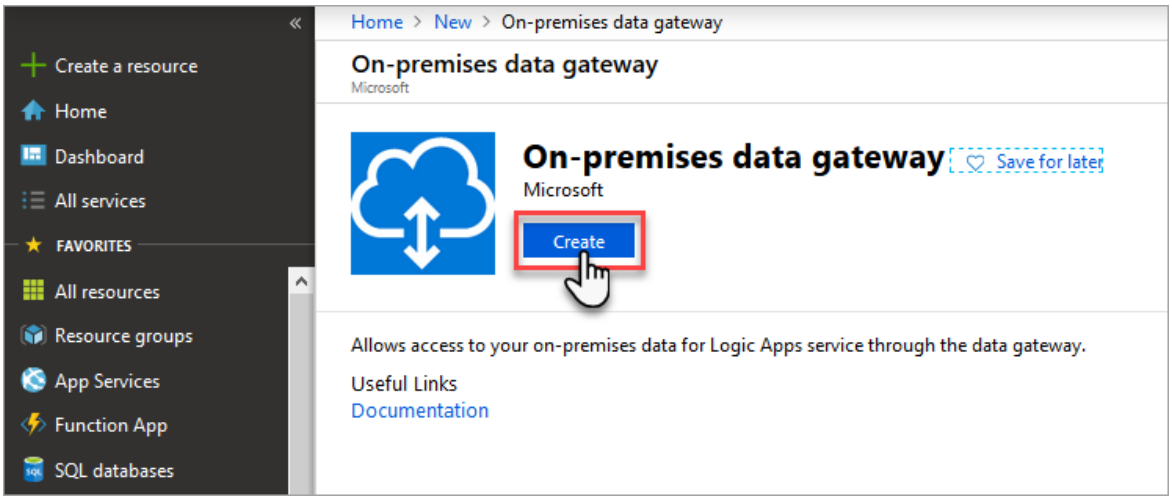
> **IMPORTANT**
>
> Save your recovery key in a safe place. The recovery key is required in-order to takeover, migrate, or restore a gateway.



Create an Azure gateway resource

After you've installed and registered your gateway, you need to create a gateway resource in Azure. Sign in to Azure with the same account you used when registering the gateway.

1. In Azure portal, click **Create a resource**, then search for **On-premises data gateway**, and then click **Create**.



2. In **Create connection gateway**, enter these settings:

   - **Name**: Enter a name for your gateway resource.
   - **Subscription**: Select the Azure subscription to associate with your gateway resource.

     The default subscription is based on the Azure account that you used to sign in.

- **Resource group**: Create a resource group or select an existing resource group.
- **Location**: Select the region you registered your gateway in.
- **Installation Name**: If your gateway installation isn't already selected, select the gateway you installed on your computer and registered.

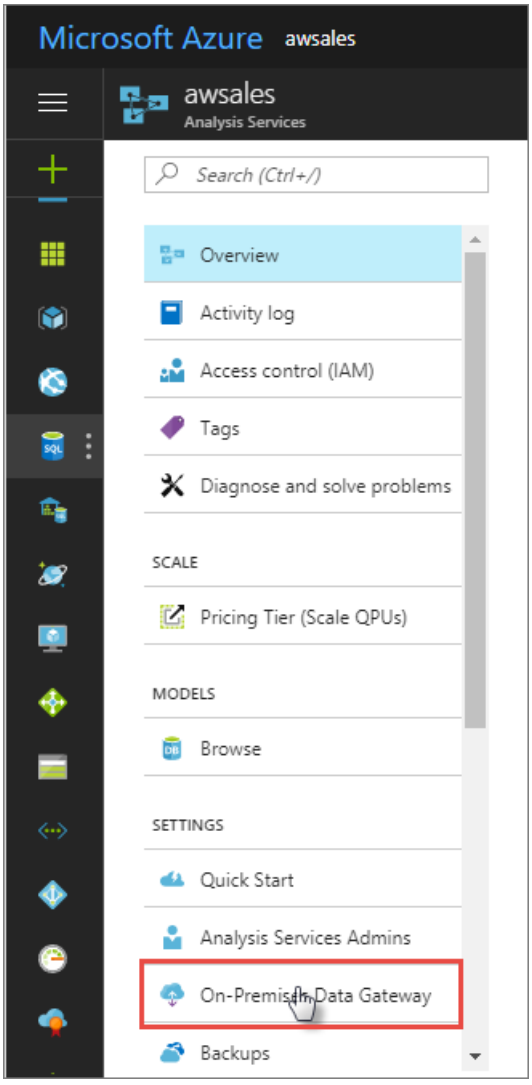
  When you're done, click **Create**.

Connect gateway resource to server

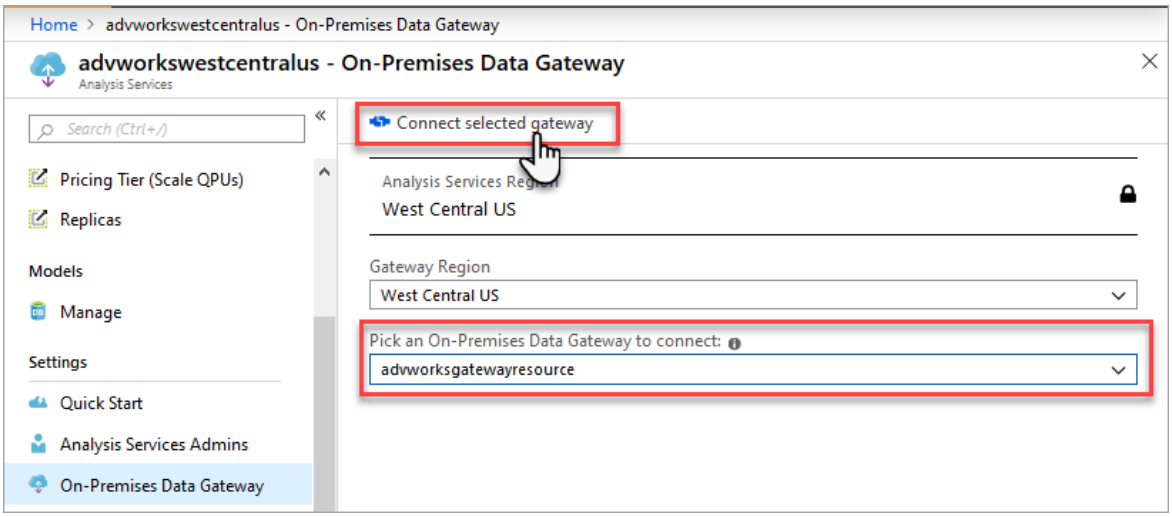
> **NOTE**
> Connecting to a gateway resource in a different subscription from your server is not supported in the portal, but is supported using PowerShell.

- Portal
- PowerShell

1. In your Azure Analysis Services server overview, click **On-Premises Data Gateway**.



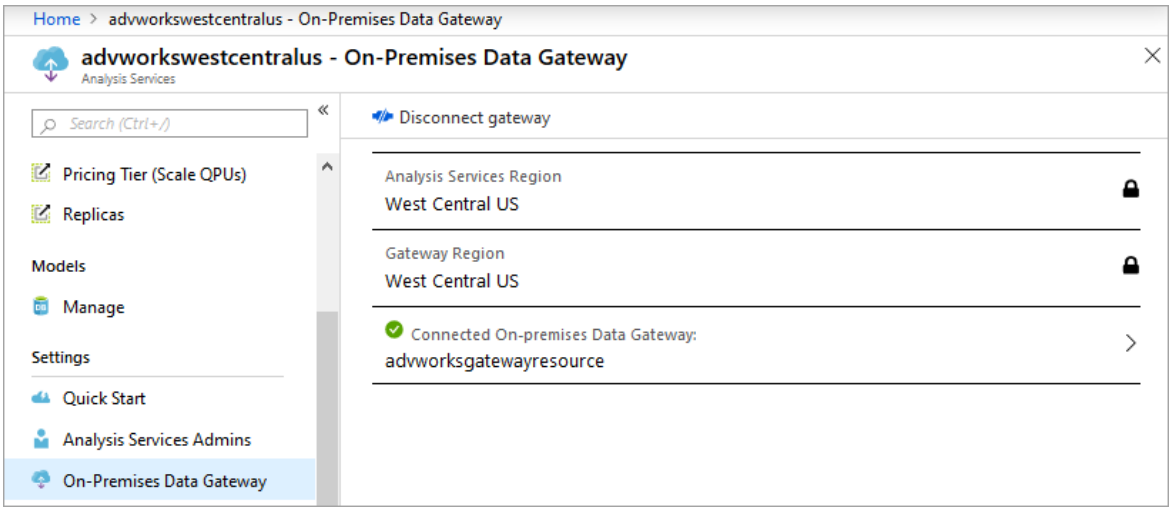
2. In **Pick an On-Premises Data Gateway to connect**, select your gateway resource, and then click **Connect selected gateway**.



NOTE

If your gateway does not appear in the list, your server is likely not in the same region as the region you specified when registering the gateway.

When connection between your server and gateway resource is successful, status will show **Connected**.



That's it. If you need to open ports or do any troubleshooting, be sure to check out On-premises data gateway.

Next steps

- Connecting to on-premises data sources
- Data sources supported in Azure Analysis Services
- Use gateway for data sources on an Azure Virtual Network
- Frequently asked questions about Analysis Services network connectivity

Use gateway for data sources on an Azure Virtual Network (VNet)

4/27/2021 • 2 minutes to read • Edit Online

This article describes the Azure Analysis Services **AlwaysUseGateway** server property for use when data sources are on an Azure Virtual Network (VNet).

Server access to VNet data sources

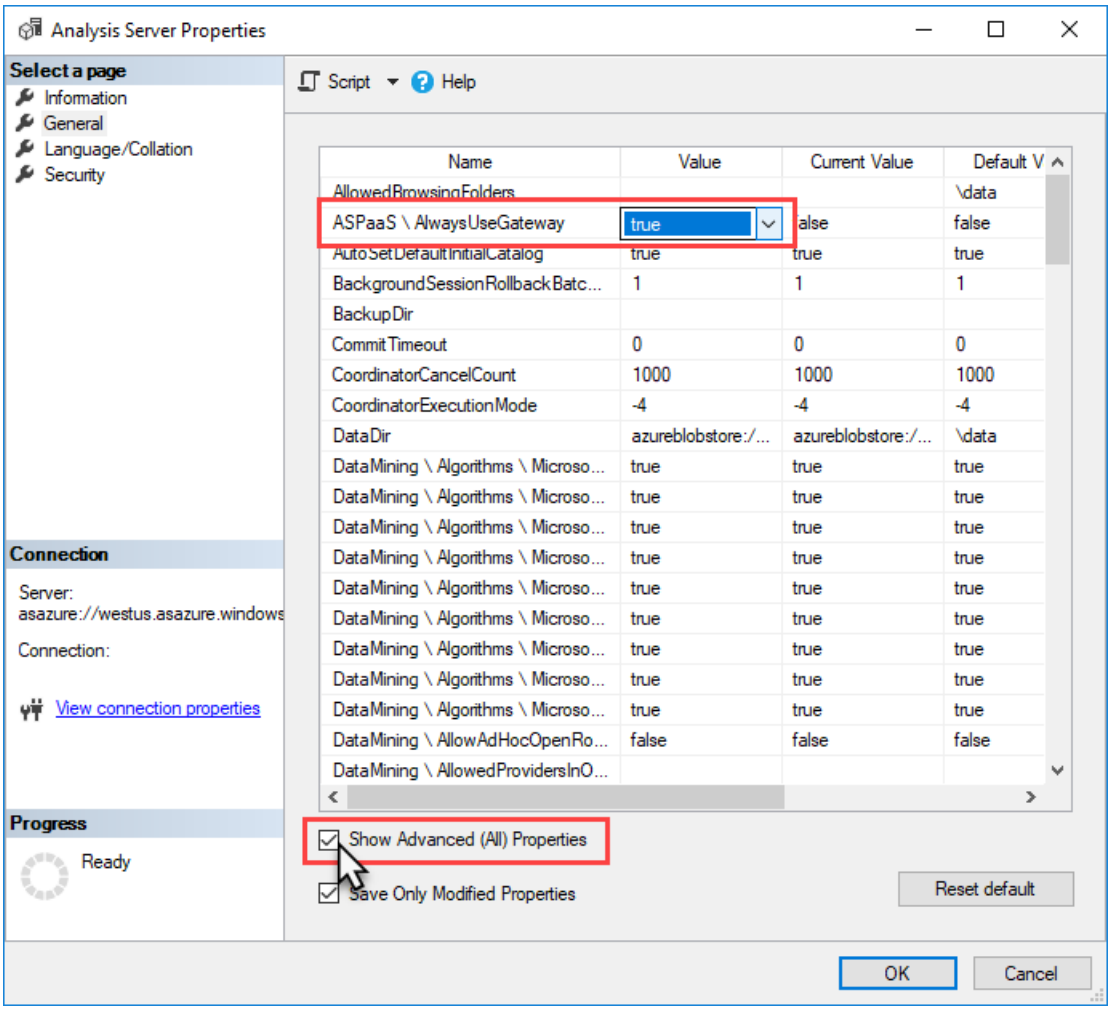
If your data sources are accessed through a VNet, your Azure Analysis Services server must connect to those data sources as if they are on-premises, in your own environment. You can configure the **AlwaysUseGateway** server property to specify the server to access all data sources through an On-premises gateway.

Azure SQL Managed Instance data sources run within Azure VNet with a private IP address. If public endpoint is enabled on the instance, a gateway is not required. If public endpoint is not enabled, an On-premises Data Gateway is required and the AlwaysUseGateway property must be set to true.

> **NOTE**
> This property is effective only when an On-premises Data Gateway is installed and configured. The gateway can be on the VNet.

Configure AlwaysUseGateway property

1. In SSMS > server > **Properties** > **General**, select **Show Advanced (All) Properties**.
2. In the **ASPaaS\AlwaysUseGateway**, select **true**.



See also

Connecting to on-premises data sources
Install and configure an on-premises data gateway
Azure Virtual Network (VNET)

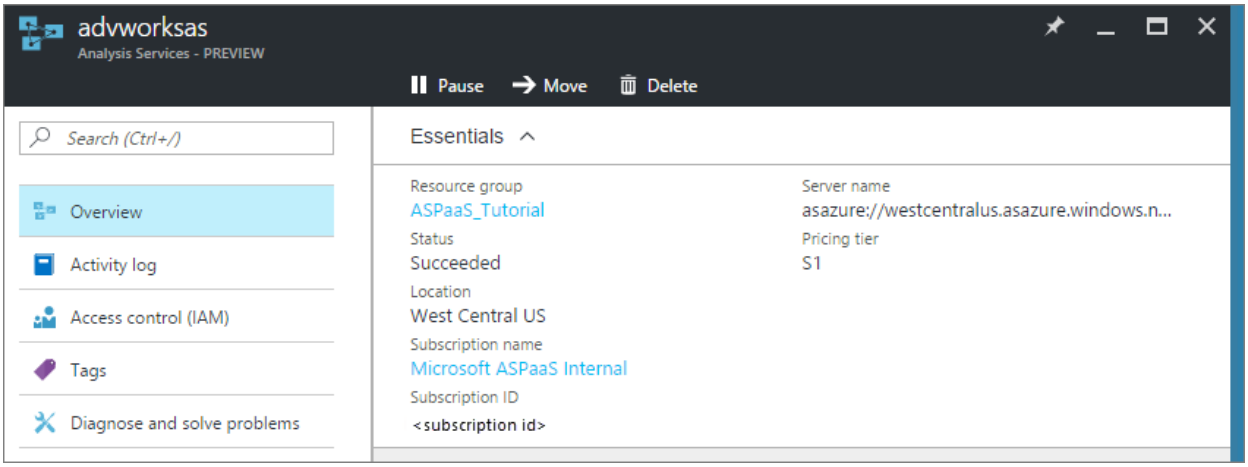
Manage Analysis Services

11/2/2020 • 2 minutes to read • Edit Online

Once you've created an Analysis Services server in Azure, there may be some administration and management tasks you need to perform right away or sometime down the road. For example, run processing to the refresh data, control who can access the models on your server, or monitor your server's health. Some management tasks can only be performed in Azure portal, others in SQL Server Management Studio (SSMS), and some tasks can be done in either.

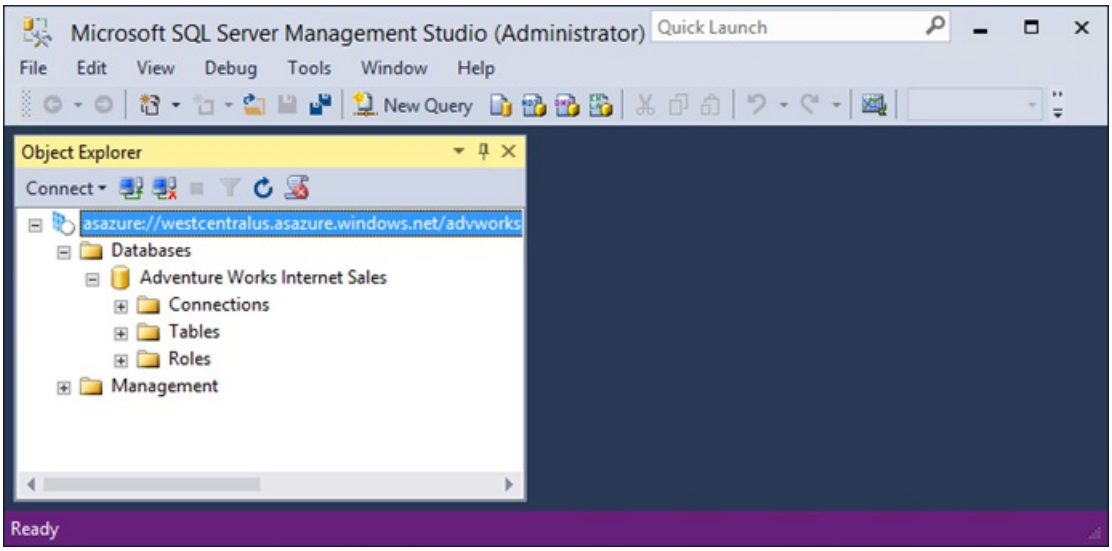
Azure portal

Azure portal is where you can create and delete servers, monitor server resources, change size, and manage who has access to your servers. If you're having some problems, you can also submit a support request.



SQL Server Management Studio

Connecting to your server in Azure is just like connecting to a server instance in your own organization. From SSMS, you can perform many of the same tasks such as process data or create a processing script, manage roles, and use PowerShell.



Download and install SSMS

To get all the latest features, and the smoothest experience when connecting to your Azure Analysis Services

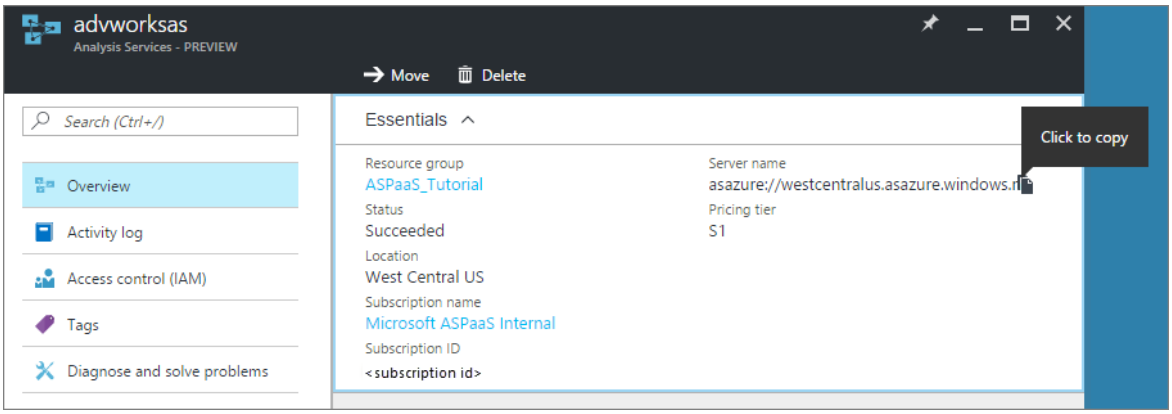
server, be sure you're using the latest version of SSMS.

Download SQL Server Management Studio.

To connect with SSMS

When using SSMS, before connecting to your server the first time, make sure your username is included in the Analysis Services Admins group. To learn more, see Server administrators and database users later in this article.

1. Before you connect, you need to get the server name. In **Azure portal** > server > **Overview** > **Server name**, copy the server name.



2. In SSMS > **Object Explorer**, click **Connect** > **Analysis Services**.

3. In the **Connect to Server** dialog box, paste in the server name, then in **Authentication**, choose one of the following authentication types:

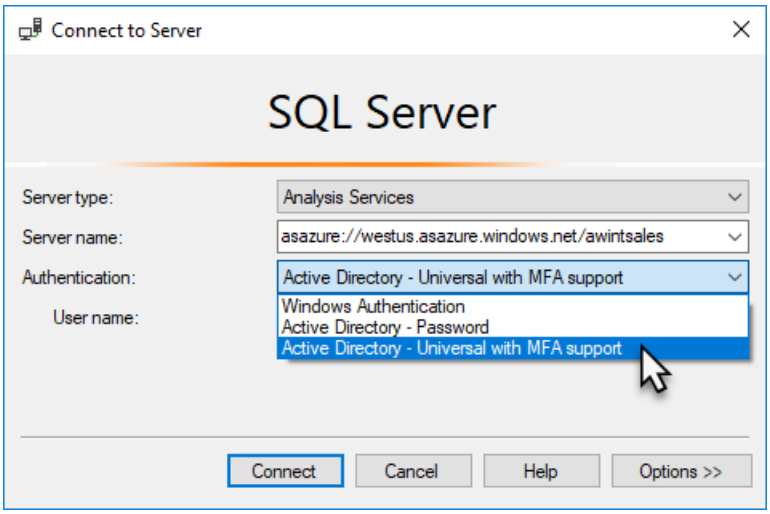
> **NOTE**
> Authentication type, **Active Directory - Universal with MFA support**, is recommended.

> **NOTE**
> If you sign in with a Microsoft Account, Live ID, Yahoo, Gmail, etc., leave the password field blank. You are prompted for a password after clicking Connect.

Windows Authentication to use your Windows domain\username and password credentials.

Active Directory Password Authentication to use an organizational account. For example, when connecting from a non-domain joined computer.

Active Directory - Universal with MFA support to use non-interactive or multi-factor authentication.



Server administrators and database users

In Azure Analysis Services, there are two types of users, server administrators and database users. Both types of users must be in your Azure Active Directory and must be specified by organizational email address or UPN. To learn more, see Authentication and user permissions.

Troubleshooting connection problems

When connecting using SSMS, if you run into problems, you may need to clear the login cache. Nothing is cached to disc. To clear the cache, close and restart the connect process.

Next steps

If you haven't already deployed a tabular model to your new server, now is a good time. To learn more, see Deploy to Azure Analysis Services.

If you've deployed a model to your server, you're ready to connect to it using a client or browser. To learn more, see Get data from Azure Analysis Services server.

Monitor server metrics

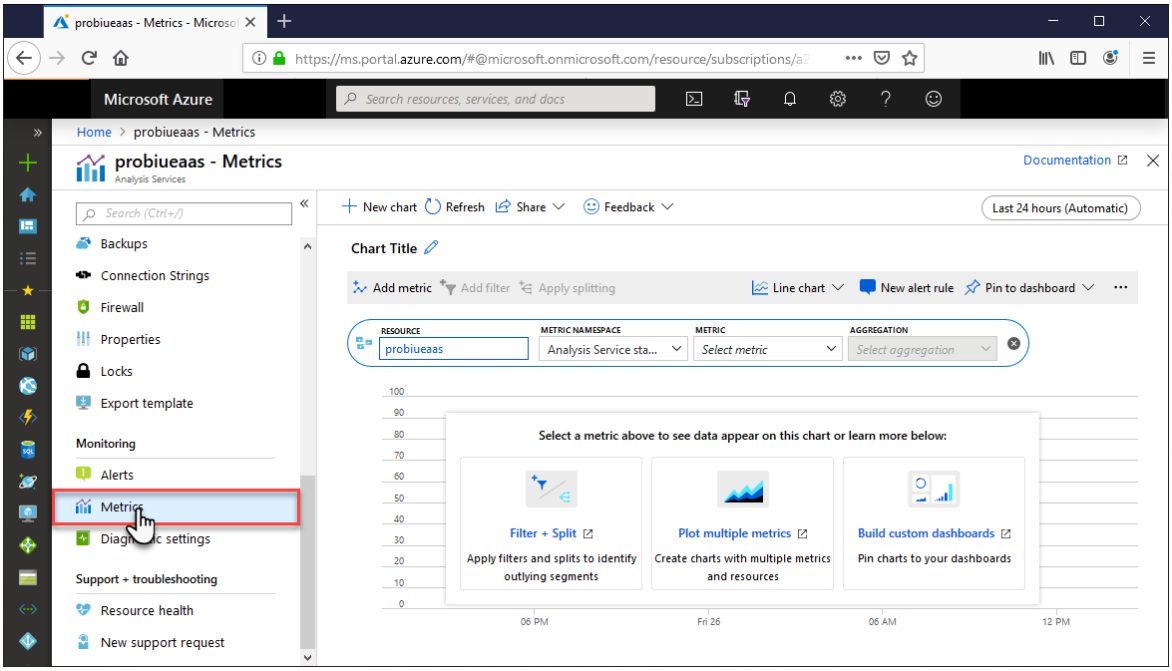
4/12/2021 • 5 minutes to read • Edit Online

Analysis Services provides metrics in Azure Metrics Explorer, a free tool in the portal, to help you monitor the performance and health of your servers. For example, monitor memory and CPU usage, number of client connections, and query resource consumption. Analysis Services uses the same monitoring framework as most other Azure services. To learn more, see Getting started with Azure Metrics Explorer.

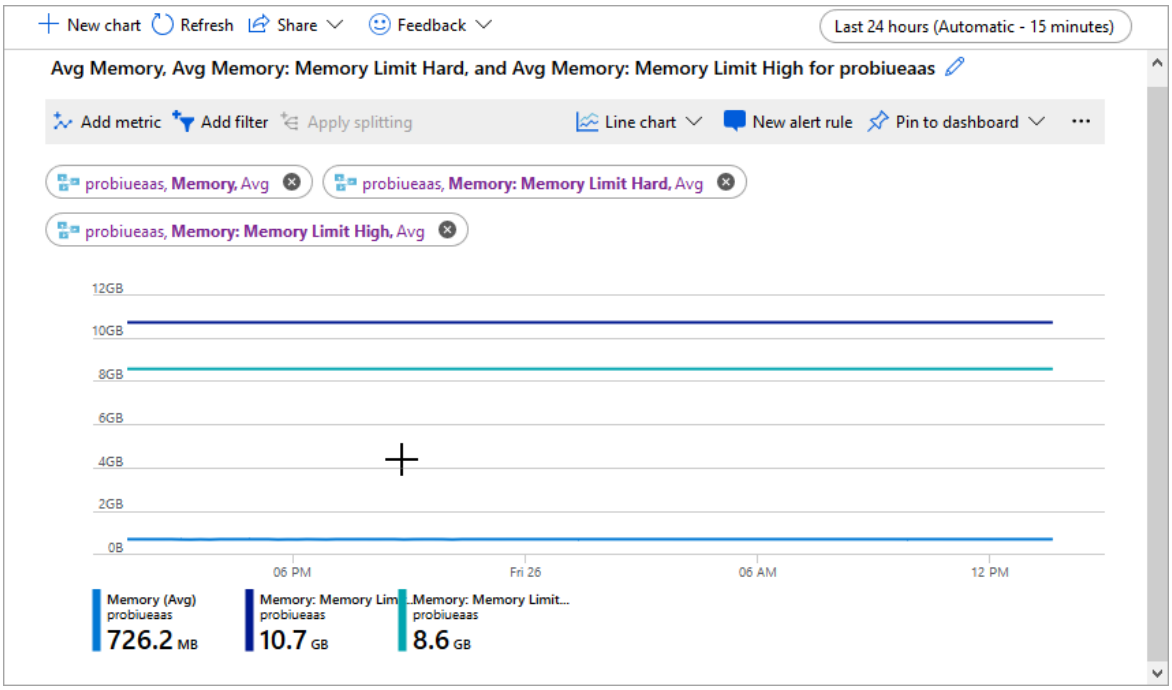
To perform more in-depth diagnostics, track performance, and identify trends across multiple service resources in a resource group or subscription, use Azure Monitor. Azure Monitor (service) may result in a billable service.

To monitor metrics for an Analysis Services server

1. In Azure portal, select **Metrics**.



2. In **Metric**, select the metrics to include in your chart.



Server metrics

Use this table to determine which metrics are best for your monitoring scenario. Only metrics of the same unit can be shown on the same chart.

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
CommandPoolJobQueueLength	Command Pool Job Queue Length	Count	Average	Number of jobs in the queue of the command thread pool.
CurrentConnections	Connection: Current connections	Count	Average	Current number of client connections established.
CurrentUserSessions	Current User Sessions	Count	Average	Current number of user sessions established.
mashup_engine_memory_metric	M Engine Memory	Bytes	Average	Memory usage by mashup engine processes
mashup_engine_qpu_metric	M Engine QPU	Count	Average	QPU usage by mashup engine processes
memory_metric	Memory	Bytes	Average	Memory. Range 0-25 GB for S1, 0-50 GB for S2 and 0-100 GB for S4
memory_thrashing_metric	Memory Thrashing	Percent	Average	Average memory thrashing.

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
CleanerCurrentPrice	Memory: Cleaner Current Price	Count	Average	Current price of memory, $/byte/time, normalized to 1000.
CleanerMemoryNonshrinkable	Memory: Cleaner Memory nonshrinkable	Bytes	Average	Amount of memory, in bytes, not subject to purging by the background cleaner.
CleanerMemoryShrinkable	Memory: Cleaner Memory shrinkable	Bytes	Average	Amount of memory, in bytes, subject to purging by the background cleaner.
MemoryLimitHard	Memory: Memory Limit Hard	Bytes	Average	Hard memory limit, from configuration file.
MemoryLimitHigh	Memory: Memory Limit High	Bytes	Average	High memory limit, from configuration file.
MemoryLimitLow	Memory: Memory Limit Low	Bytes	Average	Low memory limit, from configuration file.
MemoryLimitVertiPaq	Memory: Memory Limit VertiPaq	Bytes	Average	In-memory limit, from configuration file.
MemoryUsage	Memory: Memory Usage	Bytes	Average	Memory usage of the server process as used in calculating cleaner memory price. Equal to counter Process\PrivateBytes plus the size of memory-mapped data, ignoring any memory, which was mapped or allocated by the in-memory analytics engine (VertiPaq) in excess of the engine Memory Limit.

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
private_bytes_metric	Private Bytes	Bytes	Average	The total amount of memory the Analysis Services engine process and Mashup container processes have allocated, not including memory shared with other processes.
virtual_bytes_metric	Virtual Bytes	Bytes	Average	The current size of the virtual address space that Analysis Services engine process and Mashup container processes are using.
mashup_engine_private_bytes_metric	M Engine Private Bytes	Bytes	Average	The total amount of memory Mashup container processes have allocated, not including memory shared with other processes.
mashup_engine_virtual_bytes_metric	M Engine Virtual Bytes	Bytes	Average	The current size of the virtual address space Mashup container processes are using.
Quota	Memory: Quota	Bytes	Average	Current memory quota, in bytes. Memory quota is also known as a memory grant or memory reservation.
QuotaBlocked	Memory: Quota Blocked	Count	Average	Current number of quota requests that are blocked until other memory quotas are freed.
VertiPaqNonpaged	Memory: VertiPaq Nonpaged	Bytes	Average	Bytes of memory locked in the working set for use by the in-memory engine.
VertiPaqPaged	Memory: VertiPaq Paged	Bytes	Average	Bytes of paged memory in use for in-memory data.

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
ProcessingPoolJobQueueLength	Processing Pool Job Queue Length	Count	Average	Number of non-I/O jobs in the queue of the processing thread pool.
RowsConvertedPerSec	Processing: Rows converted per sec	CountPerSecond	Average	Rate of rows converted during processing.
RowsReadPerSec	Processing: Rows read per sec	CountPerSecond	Average	Rate of rows read from all relational databases.
RowsWrittenPerSec	Processing: Rows written per sec	CountPerSecond	Average	Rate of rows written during processing.
qpu_metric	QPU	Count	Average	QPU. Range 0-100 for S1, 0-200 for S2 and 0-400 for S4
QueryPoolBusyThreads	Query Pool Busy Threads	Count	Average	Number of busy threads in the query thread pool.
SuccessfullConnectionsPerSec	Successfull Connections Per Sec	CountPerSecond	Average	Rate of successful connection completions.
CommandPoolBusyThreads	Threads: Command pool busy threads	Count	Average	Number of busy threads in the command thread pool.
CommandPoolIdleThreads	Threads: Command pool idle threads	Count	Average	Number of idle threads in the command thread pool.
LongParsingBusyThreads	Threads: Long parsing busy threads	Count	Average	Number of busy threads in the long parsing thread pool.
LongParsingIdleThreads	Threads: Long parsing idle threads	Count	Average	Number of idle threads in the long parsing thread pool.
LongParsingJobQueueLength	Threads: Long parsing job queue length	Count	Average	Number of jobs in the queue of the long parsing thread pool.
ProcessingPoolIOJobQueueLength	Threads: Processing pool I/O job queue length	Count	Average	Number of I/O jobs in the queue of the processing thread pool.

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
ProcessingPoolBusyIOJobThreads	Threads: Processing pool busy I/O job threads	Count	Average	Number of threads running I/O jobs in the processing thread pool.
ProcessingPoolBusyNonIOThreads	Threads: Processing pool busy non-I/O threads	Count	Average	Number of threads running non-I/O jobs in the processing thread pool.
ProcessingPoolIdleIOJobThreads	Threads: Processing pool idle I/O job threads	Count	Average	Number of idle threads for I/O jobs in the processing thread pool.
ProcessingPoolIdleNonIOThreads	Threads: Processing pool idle non-I/O threads	Count	Average	Number of idle threads in the processing thread pool dedicated to non-I/O jobs.
QueryPoolIdleThreads	Threads: Query pool idle threads	Count	Average	Number of idle threads for I/O jobs in the processing thread pool.
QueryPoolJobQueueLength	Threads: Query pool job queue length	Count	Average	Number of jobs in the queue of the query thread pool.
ShortParsingBusyThreads	Threads: Short parsing busy threads	Count	Average	Number of busy threads in the short parsing thread pool.
ShortParsingIdleThreads	Threads: Short parsing idle threads	Count	Average	Number of idle threads in the short parsing thread pool.
ShortParsingJobQueueLength	Threads: Short parsing job queue length	Count	Average	Number of jobs in the queue of the short parsing thread pool.
TotalConnectionFailures	Total Connection Failures	Count	Average	Total failed connection attempts.
TotalConnectionRequests	Total Connection Requests	Count	Average	Total connection requests.

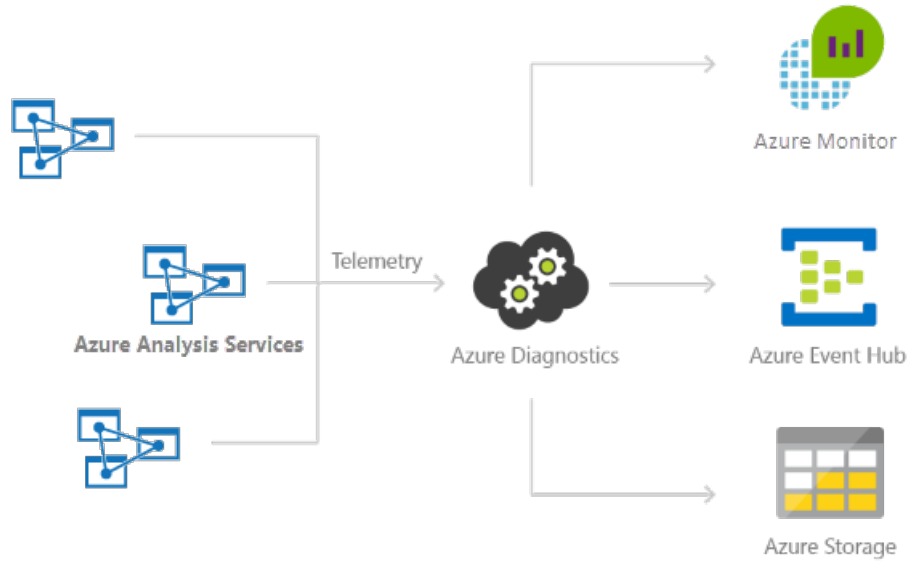
Next steps

Azure Monitor overview
Getting started with Azure Metrics Explorer
Metrics in Azure Monitor REST API

Setup diagnostic logging

4/27/2021 • 9 minutes to read • Edit Online

An important part of any Analysis Services solution is monitoring how your servers are performing. Azure Analysis services is integrated with Azure Monitor. With Azure Monitor resource logs, you can monitor and send logs to Azure Storage, stream them to Azure Event Hubs, and export them to Azure Monitor logs.



NOTE

This article has been updated to use the Azure Az PowerShell module. The Az PowerShell module is the recommended PowerShell module for interacting with Azure. To get started with the Az PowerShell module, see Install Azure PowerShell. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

What's logged?

You can select **Engine**, **Service**, and **Metrics** categories.

Engine

Selecting **Engine** logs all xEvents. You cannot select individual events.

XEVENT CATEGORIES	EVENT NAME
Security Audit	Audit Login
Security Audit	Audit Logout
Security Audit	Audit Server Starts And Stops
Progress Reports	Progress Report Begin
Progress Reports	Progress Report End

XEVENT CATEGORIES	EVENT NAME
Progress Reports	Progress Report Current
Queries	Query Begin
Queries	Query End
Commands	Command Begin
Commands	Command End
Errors & Warnings	Error
Discover	Discover End
Notification	Notification
Session	Session Initialize
Locks	Deadlock
Query Processing	VertiPaq SE Query Begin
Query Processing	VertiPaq SE Query End
Query Processing	VertiPaq SE Query Cache Match
Query Processing	Direct Query Begin
Query Processing	Direct Query End

Service

OPERATION NAME	OCCURS WHEN
ResumeServer	Resume a server
SuspendServer	Pause a server
DeleteServer	Delete a server
RestartServer	User restarts a server through SSMS or PowerShell
GetServerLogFiles	User exports server log through PowerShell
ExportModel	User exports a model in the portal by using Open in Visual Studio

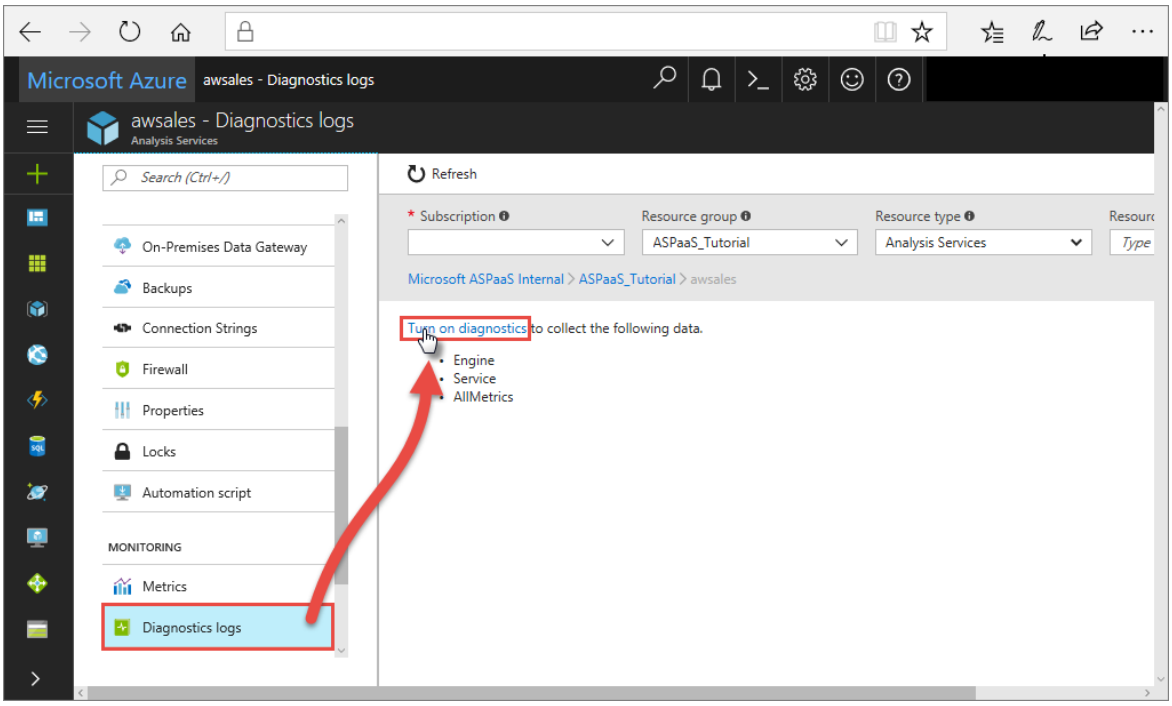
All metrics

The Metrics category logs the same Server metrics to the AzureMetrics table. If you're using query scale-out and need to separate metrics for each read replica, use the AzureDiagnostics table instead, where **OperationName** is equal to **LogMetric**.

Setup diagnostics logging

Azure portal

1. In Azure portal > server, click **Diagnostic settings** in the left navigation, and then click **Turn on diagnostics**.



2. In **Diagnostic settings**, specify the following options:

   - **Name**. Enter a name for the logs to create.
   - **Archive to a storage account**. To use this option, you need an existing storage account to connect to. See Create a storage account. Follow the instructions to create a Resource Manager, general-purpose account, then select your storage account by returning to this page in the portal. It may take a few minutes for newly created storage accounts to appear in the drop-down menu.
   - **Stream to an event hub**. To use this option, you need an existing Event Hub namespace and event hub to connect to. To learn more, see Create an Event Hubs namespace and an event hub using the Azure portal. Then return to this page in the portal to select the Event Hub namespace and policy name.
   - **Send to Azure Monitor (Log Analytics workspace)**. To use this option, either use an existing workspace or create a new workspace resource in the portal. For more information on viewing your logs, see View logs in Log Analytics workspace in this article.
   - **Engine**. Select this option to log xEvents. If you're archiving to a storage account, you can select the retention period for the resource logs. Logs are autodeleted after the retention period expires.
   - **Service**. Select this option to log service level events. If you are archiving to a storage account, you can select the retention period for the resource logs. Logs are autodeleted after the retention period expires.
   - **Metrics**. Select this option to store verbose data in Metrics. If you are archiving to a storage account, you can select the retention period for the resource logs. Logs are autodeleted after the retention period expires.

3. Click **Save**.

   If you receive an error that says "Failed to update diagnostics for <workspace name>. The subscription

<subscription id> is not registered to use microsoft.insights." follow the Troubleshoot Azure Diagnostics instructions to register the account, then retry this procedure.

If you want to change how your resource logs are saved at any point in the future, you can return to this page to modify settings.

PowerShell

Here are the basic commands to get you going. If you want step-by-step help on setting up logging to a storage account by using PowerShell, see the tutorial later in this article.

To enable metrics and resource logging by using PowerShell, use the following commands:

- To enable storage of resource logs in a storage account, use this command:

  ```
  Set-AzDiagnosticSetting -ResourceId [your resource id] -StorageAccountId [your storage account id] -
  Enabled $true
  ```

  The storage account ID is the resource ID for the storage account where you want to send the logs.

- To enable streaming of resource logs to an event hub, use this command:

  ```
  Set-AzDiagnosticSetting -ResourceId [your resource id] -ServiceBusRuleId [your service bus rule id] -
  Enabled $true
  ```

  The Azure Service Bus rule ID is a string with this format:

  ```
  {service bus resource ID}/authorizationrules/{key name}
  ```

- To enable sending resource logs to a Log Analytics workspace, use this command:

  ```
  Set-AzDiagnosticSetting -ResourceId [your resource id] -WorkspaceId [resource id of the log analytics
  workspace] -Enabled $true
  ```

- You can obtain the resource ID of your Log Analytics workspace by using the following command:

  ```
  (Get-AzOperationalInsightsWorkspace).ResourceId
  ```

You can combine these parameters to enable multiple output options.

REST API

Learn how to change diagnostics settings by using the Azure Monitor REST API.

Resource Manager template

Learn how to enable diagnostics settings at resource creation by using a Resource Manager template.

Manage your logs

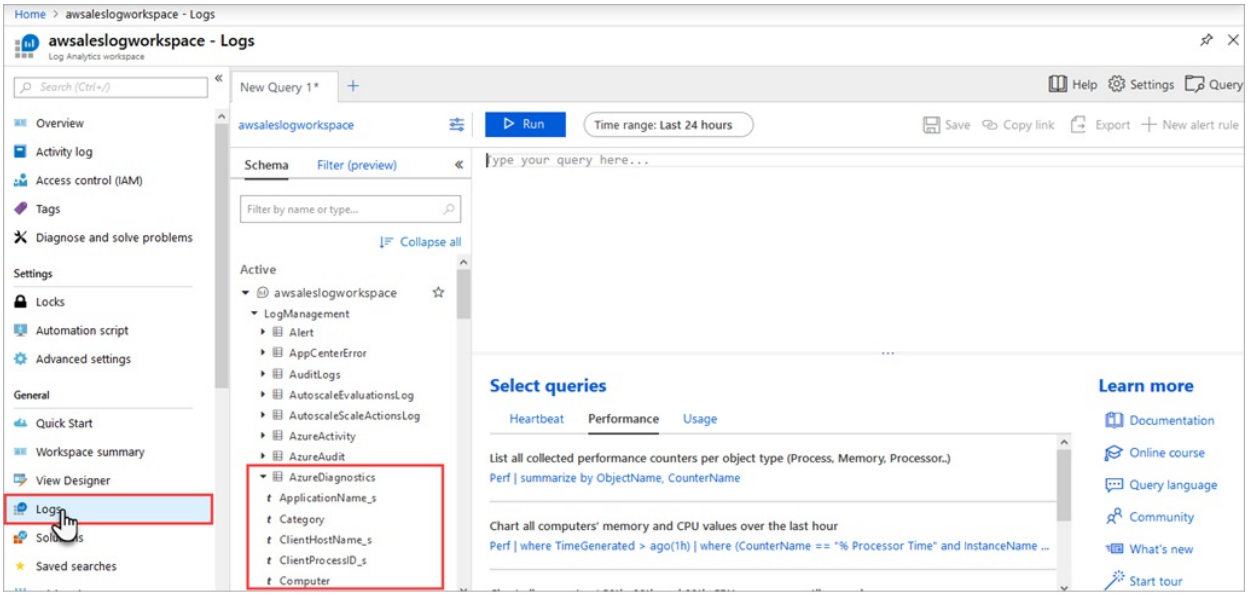
Logs are typically available within a couple hours of setting up logging. It's up to you to manage your logs in your storage account:

- Use standard Azure access control methods to secure your logs by restricting who can access them.
- Delete logs that you no longer want to keep in your storage account.
- Be sure to set a retention period for so old logs are deleted from your storage account.

View logs in Log Analytics workspace

Metrics and server events are integrated with xEvents in your Log Analytics workspace resource for side-by-side analysis. Log Analytics workspace can also be configured to receive events from other Azure services providing a holistic view of diagnostic logging data across your architecture.

To view your diagnostic data, in Log Analytics workspace, open **Logs** from the left menu.



In the query builder, expand **LogManagement** > **AzureDiagnostics**. AzureDiagnostics includes Engine and Service events. Notice a query is created on-the-fly. The EventClass_s field contains xEvent names, which may look familiar if you've used xEvents for on-premises logging. Click **EventClass_s** or one of the event names and Log Analytics workspace continues constructing a query. Be sure to save your queries to reuse later.

Example queries

Example 1

The following query returns durations for each query end/refresh end event for a model database and server. If scaled out, the results are broken out by replica because the replica number is included in ServerName_s. Grouping by RootActivityId_g reduces the row count retrieved from the Azure Diagnostics REST API and helps stay within the limits as described in Log Analytics Rate limits.

```
let window = AzureDiagnostics
   | where ResourceProvider == "MICROSOFT.ANALYSISSERVICES" and Resource =~ "MyServerName" and
DatabaseName_s =~ "MyDatabaseName" ;
window
| where OperationName has "QueryEnd" or (OperationName has "CommandEnd" and EventSubclass_s == 38)
| where extract(@"([^,]*)", 1,Duration_s, typeof(long)) > 0
| extend DurationMs=extract(@"([^,]*)", 1,Duration_s, typeof(long))
| project
StartTime_t,EndTime_t,ServerName_s,OperationName,RootActivityId_g,TextData_s,DatabaseName_s,ApplicationName_
s,Duration_s,EffectiveUsername_s,User_s,EventSubclass_s,DurationMs
| order by StartTime_t asc
```

Example 2

The following query returns memory and QPU consumption for a server. If scaled out, the results are broken out by replica because the replica number is included in ServerName_s.

```
let window = AzureDiagnostics
   | where ResourceProvider == "MICROSOFT.ANALYSISSERVICES" and Resource =~ "MyServerName";
window
| where OperationName == "LogMetric"
| where name_s == "memory_metric" or name_s == "qpu_metric"
| project ServerName_s, TimeGenerated, name_s, value_s
| summarize avg(todecimal(value_s)) by ServerName_s, name_s, bin(TimeGenerated, 1m)
| order by TimeGenerated asc
```

Example 3

The following query returns the Rows read/sec Analysis Services engine performance counters for a server.

```
let window =  AzureDiagnostics
   | where ResourceProvider == "MICROSOFT.ANALYSISSERVICES" and Resource =~ "MyServerName";
window
| where OperationName == "LogMetric"
| where parse_json(tostring(parse_json(perfobject_s).counters))[0].name == "Rows read/sec"
| extend Value = tostring(parse_json(tostring(parse_json(perfobject_s).counters))[0].value)
| project ServerName_s, TimeGenerated, Value
| summarize avg(todecimal(Value)) by ServerName_s, bin(TimeGenerated, 1m)
| order by TimeGenerated asc
```

There are hundreds of queries you can use. To learn more about queries, see Get started with Azure Monitor log queries.

Turn on logging by using PowerShell

In this quick tutorial, you create a storage account in the same subscription and resource group as your Analysis Service server. You then use Set-AzDiagnosticSetting to turn on diagnostics logging, sending output to the new storage account.

Prerequisites

To complete this tutorial, you must have the following resources:

- An existing Azure Analysis Services server. For instructions on creating a server resource, see Create a server in Azure portal, or Create an Azure Analysis Services server by using PowerShell.

Connect to your subscriptions

Start an Azure PowerShell session and sign in to your Azure account with the following command:

```
Connect-AzAccount
```

In the pop-up browser window, enter your Azure account user name and password. Azure PowerShell gets all the subscriptions that are associated with this account and by default, uses the first one.

If you have multiple subscriptions, you might have to specify a specific one that was used to create your Azure Key Vault. Type the following to see the subscriptions for your account:

```
Get-AzSubscription
```

Then, to specify the subscription that's associated with the Azure Analysis Services account you are logging, type:

```
Set-AzContext -SubscriptionId <subscription ID>
```

> **NOTE**
> If you have multiple subscriptions associated with your account, it is important to specify the subscription.

Create a new storage account for your logs

You can use an existing storage account for your logs, provided it's in the same subscription as your server. For this tutorial, you create a new storage account dedicated to Analysis Services logs. To make it easy, you're storing the storage account details in a variable named **sa**.

You also use the same resource group as the one that contains your Analysis Services server. Substitute values for `awsales_resgroup`, `awsaleslogs`, and `West Central US` with your own values:

```
$sa = New-AzStorageAccount -ResourceGroupName awsales_resgroup `
-Name awsaleslogs -Type Standard_LRS -Location 'West Central US'
```

Identify the server account for your logs

Set the account name to a variable named **account**, where ResourceName is the name of the account.

```
$account = Get-AzResource -ResourceGroupName awsales_resgroup `
-ResourceName awsales -ResourceType "Microsoft.AnalysisServices/servers"
```

Enable logging

To enable logging, use the Set-AzDiagnosticSetting cmdlet together with the variables for the new storage account, server account, and the category. Run the following command, setting the **-Enabled** flag to **$true**:

```
Set-AzDiagnosticSetting  -ResourceId $account.ResourceId -StorageAccountId $sa.Id -Enabled $true -Categories
Engine
```

The output should look something like this example:

```
StorageAccountId           :
/subscriptions/a23279b5-xxxx-xxxx-xxxx-
47b7c6d423ea/resourceGroups/awsales_resgroup/providers/Microsoft.Storage/storageAccounts/awsaleslogs
ServiceBusRuleId           :
EventHubAuthorizationRuleId :
Metrics
    TimeGrain       : PT1M
    Enabled         : False
    RetentionPolicy
    Enabled : False
    Days    : 0


Logs
    Category        : Engine
    Enabled         : True
    RetentionPolicy
    Enabled : False
    Days    : 0


    Category        : Service
    Enabled         : False
    RetentionPolicy
    Enabled : False
    Days    : 0


WorkspaceId                :
Id                         : /subscriptions/a23279b5-xxxx-xxxx-xxxx-
47b7c6d423ea/resourcegroups/awsales_resgroup/providers/microsoft.analysisservic
es/servers/awsales/providers/microsoft.insights/diagnosticSettings/service
Name                       : service
Type                       :
Location                   :
Tags                       :
```

This output confirms that logging is now enabled for the server, saving information to the storage account.

You can also set retention policy for your logs so older logs are automatically deleted. For example, set retention policy using **-RetentionEnabled** flag to **$true**, and set **-RetentionInDays** parameter to **90**. Logs older than 90 days are automatically deleted.

```
Set-AzDiagnosticSetting -ResourceId $account.ResourceId`
 -StorageAccountId $sa.Id -Enabled $true -Categories Engine`
  -RetentionEnabled $true -RetentionInDays 90
```

Next steps

Learn more about Azure Monitor resource logging.

See Set-AzDiagnosticSetting in PowerShell help.

Use the portal to create an Azure AD application and service principal that can access resources

8/27/2021 • 8 minutes to read • Edit Online

This article shows you how to create a new Azure Active Directory (Azure AD) application and service principal that can be used with the role-based access control. When you have applications, hosted services, or automated tools that needs to access or modify resources, you can create an identity for the app. This identity is known as a service principal. Access to resources is restricted by the roles assigned to the service principal, giving you control over which resources can be accessed and at which level. For security reasons, it's always recommended to use service principals with automated tools rather than allowing them to log in with a user identity.

This article shows you how to use the portal to create the service principal in the Azure portal. It focuses on a single-tenant application where the application is intended to run within only one organization. You typically use single-tenant applications for line-of-business applications that run within your organization. You can also use Azure PowerShell to create a service principal.

> **IMPORTANT**
> Instead of creating a service principal, consider using managed identities for Azure resources for your application identity. If your code runs on a service that supports managed identities and accesses resources that support Azure AD authentication, managed identities are a better option for you. To learn more about managed identities for Azure resources, including which services currently support it, see What is managed identities for Azure resources?.

App registration, app objects, and service principals

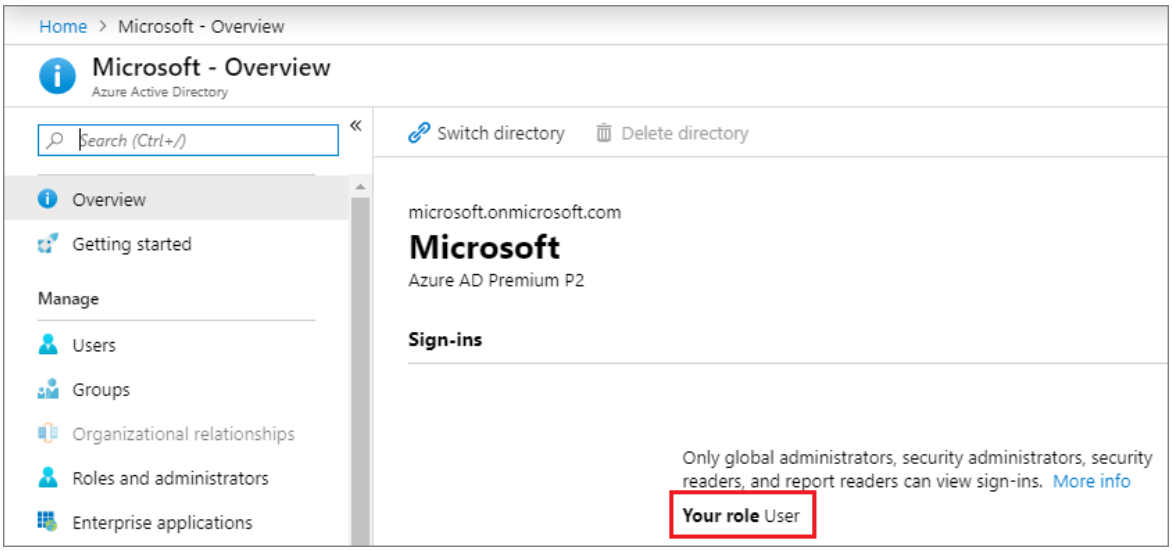
There is no way to directly create a service principal using the Azure portal. When you register an application through the Azure portal, an application object and service principal are automatically created in your home directory or tenant. For more information on the relationship between app registration, application objects, and service principals, read Application and service principal objects in Azure Active Directory.

Permissions required for registering an app

You must have sufficient permissions to register an application with your Azure AD tenant, and assign to the application a role in your Azure subscription.

Check Azure AD permissions

1. Select **Azure Active Directory**.
2. Note your role. If you have the **User** role, you must make sure that non-administrators can register applications.



3. In the left pane, select **User settings**.

4. Check the **App registrations** setting. This value can only be set by an administrator. If set to **Yes**, any user in the Azure AD tenant can register an app.

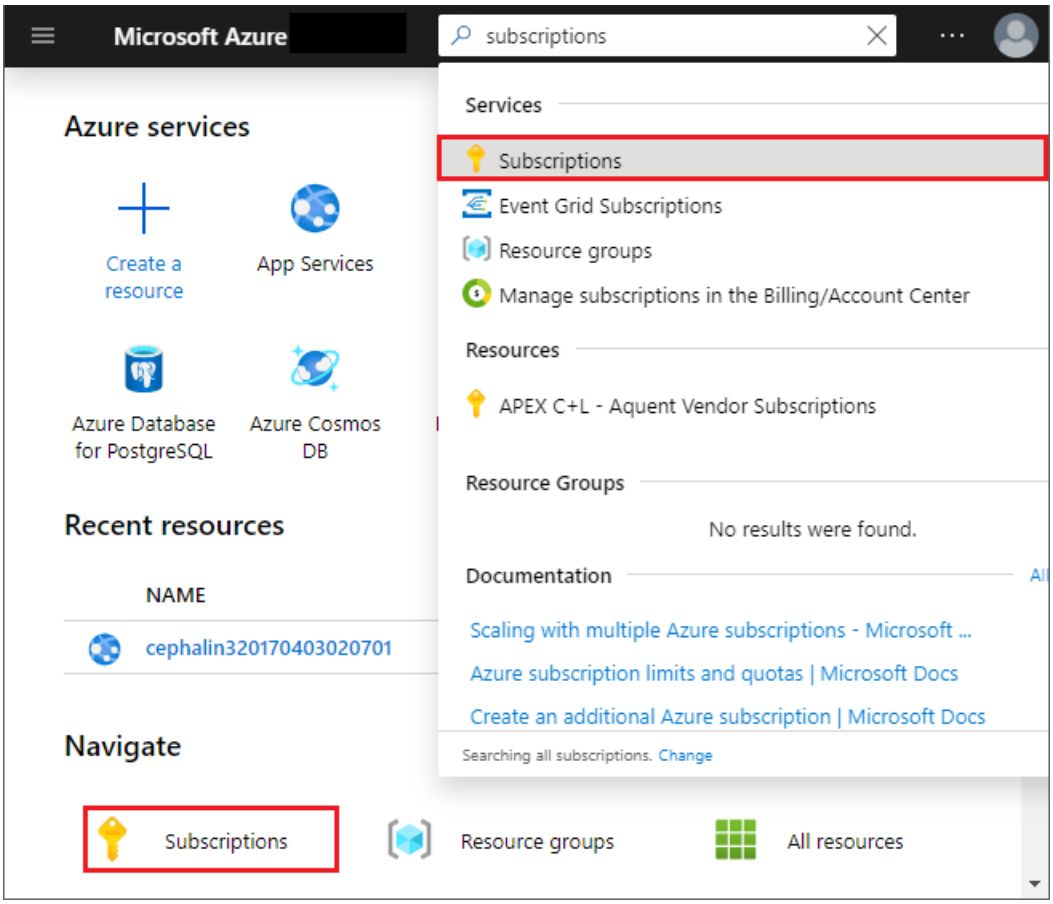
If the app registrations setting is set to **No**, only users with an administrator role may register these types of applications. See Azure AD built-in roles to learn about available administrator roles and the specific permissions in Azure AD that are given to each role. If your account is assigned the User role, but the app registration setting is limited to admin users, ask your administrator to either assign you one of the administrator roles that can create and manage all aspects of app registrations, or to enable users to register apps.

Check Azure subscription permissions

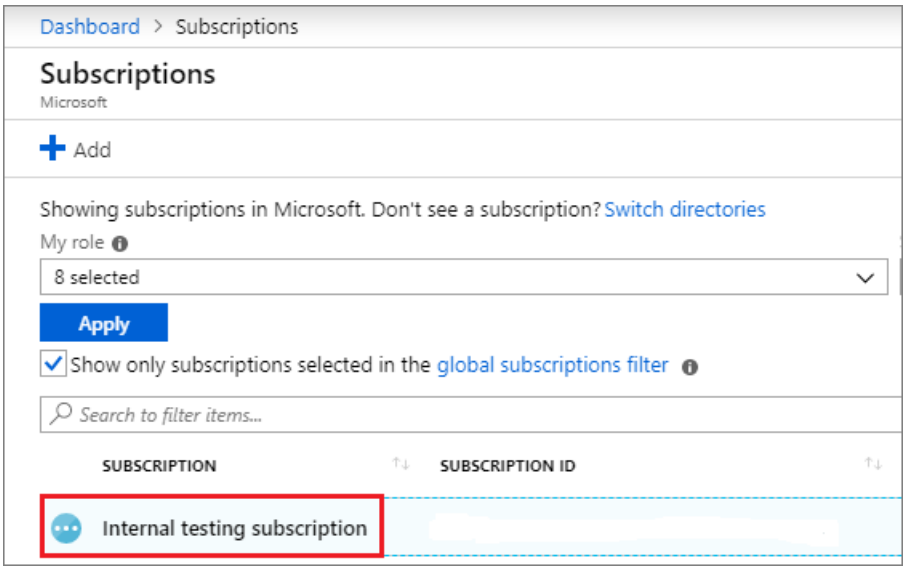
In your Azure subscription, your account must have `Microsoft.Authorization/*/Write` access to assign a role to an AD app. This action is granted through the Owner role or User Access Administrator role. If your account is assigned the **Contributor** role, you don't have adequate permission. You will receive an error when attempting to assign the service principal a role.

To check your subscription permissions:

1. Search for and select **Subscriptions**, or select **Subscriptions** on the **Home** page.

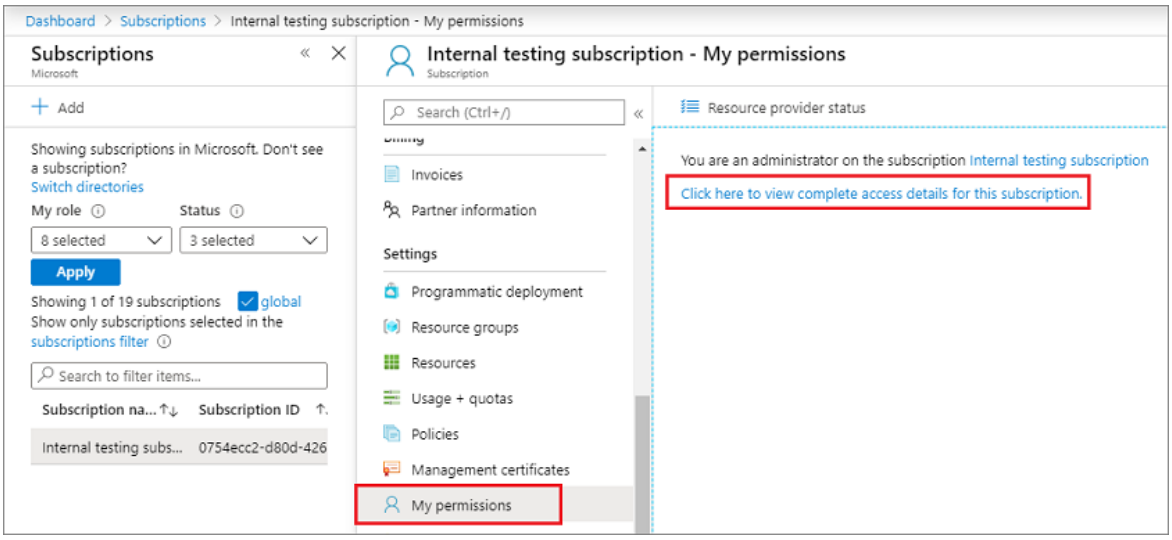


2. Select the subscription you want to create the service principal in.

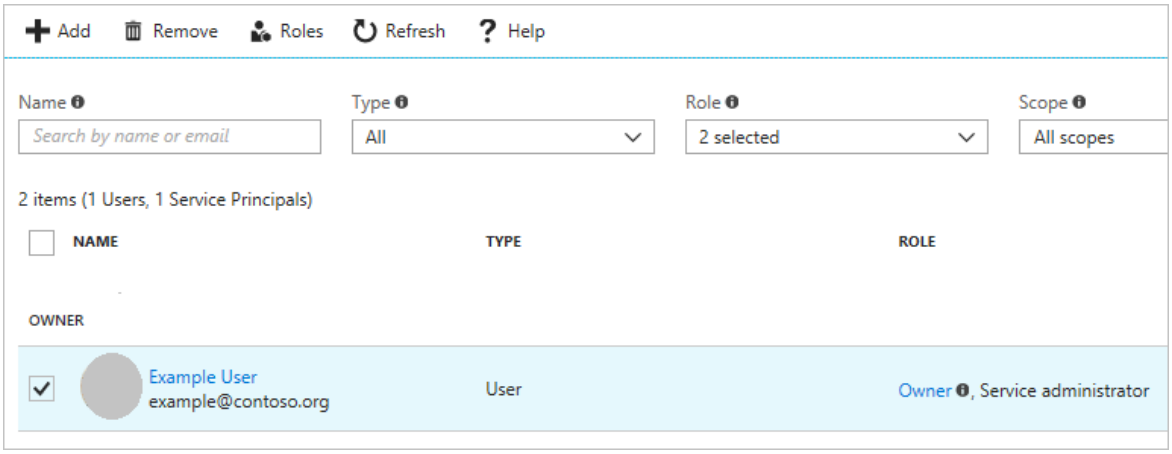


If you don't see the subscription you're looking for, select **global subscriptions filter**. Make sure the subscription you want is selected for the portal.

3. Select **My permissions**. Then, select **Click here to view complete access details for this subscription**.



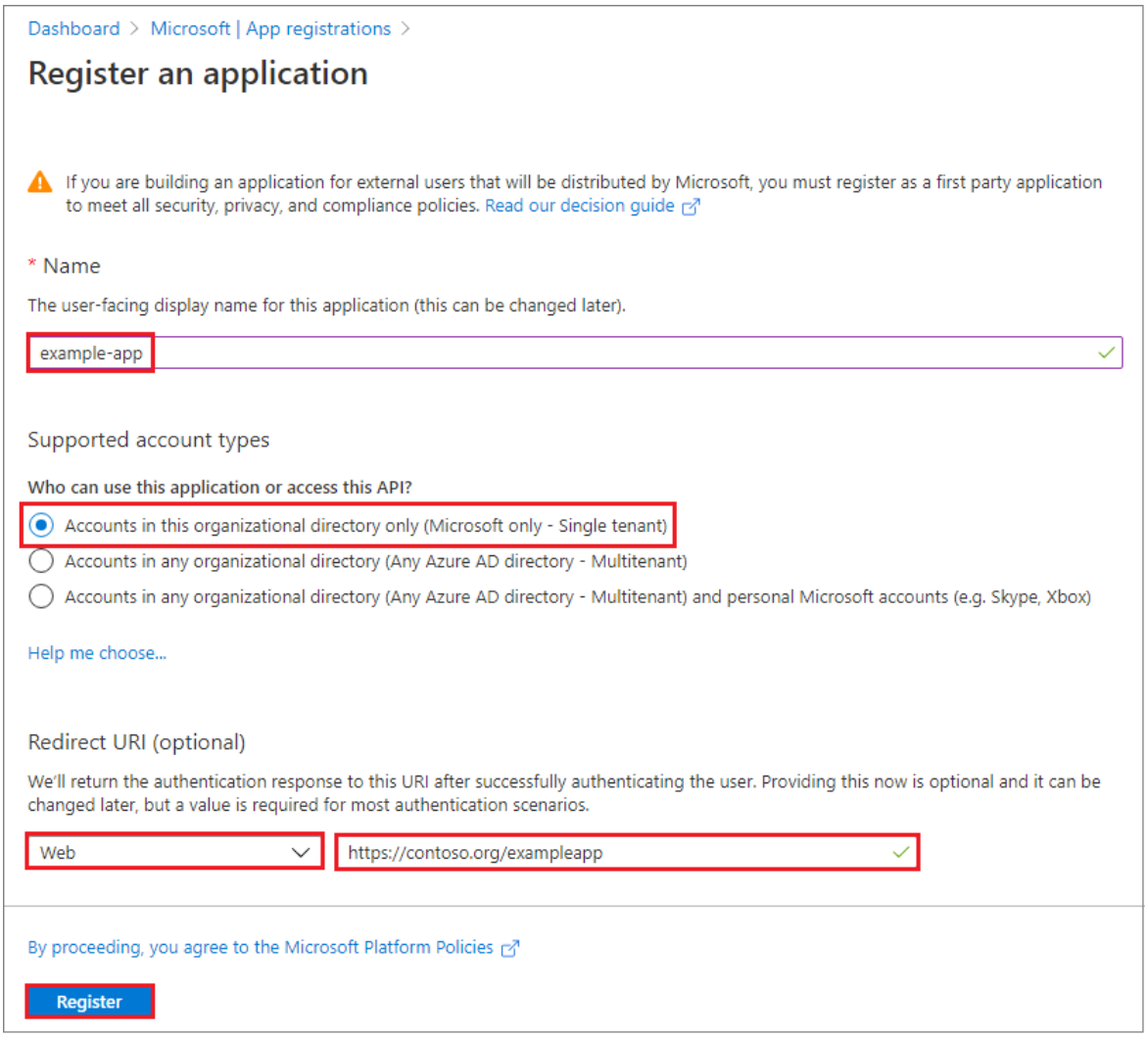
4. Select **View** in **Role assignments** to view your assigned roles, and determine if you have adequate permissions to assign a role to an AD app. If not, ask your subscription administrator to add you to User Access Administrator role. In the following image, the user is assigned the Owner role, which means that user has adequate permissions.



Register an application with Azure AD and create a service principal

Let's jump straight into creating the identity. If you run into a problem, check the required permissions to make sure your account can create the identity.

1. Sign in to your Azure Account through the Azure portal.
2. Select **Azure Active Directory**.
3. Select **App registrations**.
4. Select **New registration**.
5. Name the application. Select a supported account type, which determines who can use the application. Under **Redirect URI**, select **Web** for the type of application you want to create. Enter the URI where the access token is sent to. You can't create credentials for a Native application. You can't use that type for an automated application. After setting the values, select **Register**.



You've created your Azure AD application and service principal.

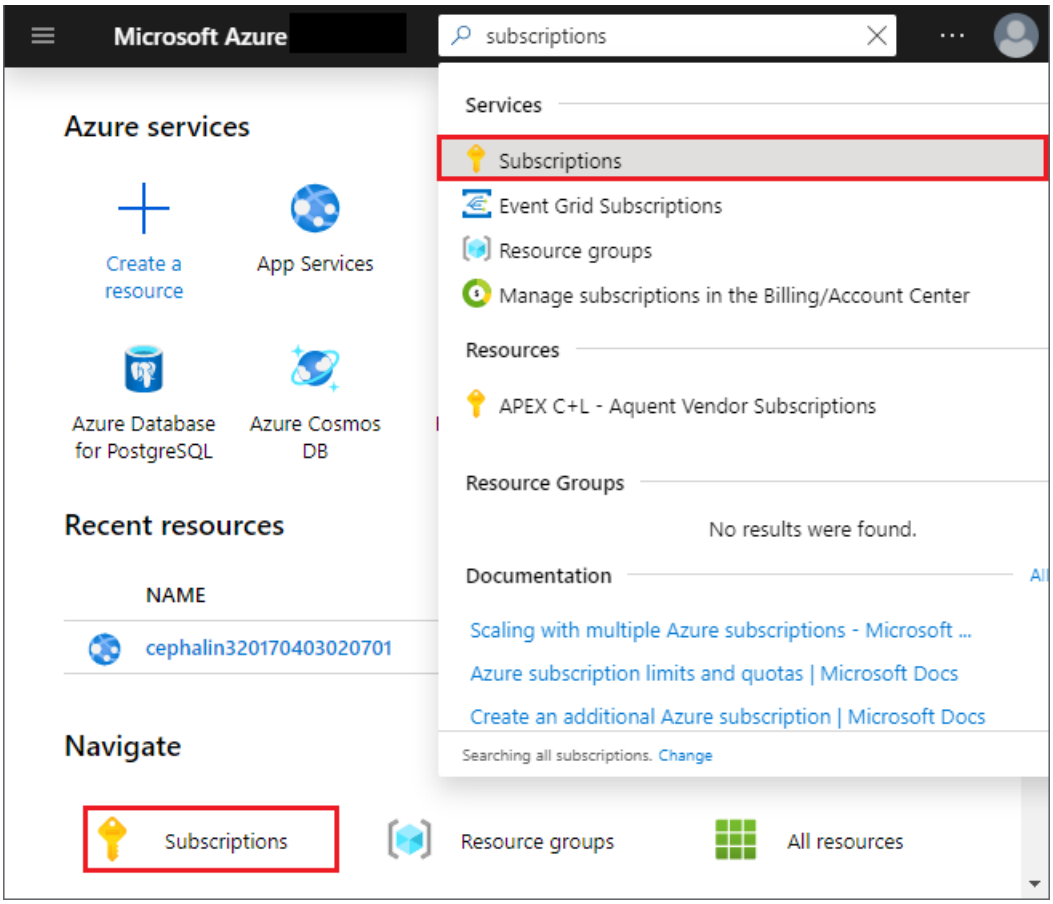
> **NOTE**
> You can register multiple applications with the same name in Azure AD, but the applications must have different Application (client) IDs.

Assign a role to the application

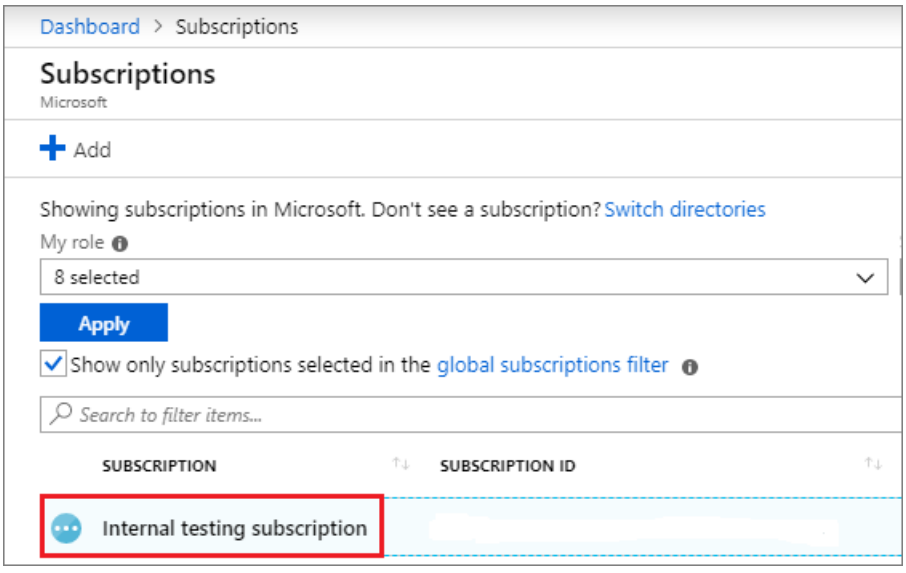
To access resources in your subscription, you must assign a role to the application. Decide which role offers the right permissions for the application. To learn about the available roles, see Azure built-in roles.

You can set the scope at the level of the subscription, resource group, or resource. Permissions are inherited to lower levels of scope. For example, adding an application to the *Reader* role for a resource group means it can read the resource group and any resources it contains.

1. In the Azure portal, select the level of scope you wish to assign the application to. For example, to assign a role at the subscription scope, search for and select **Subscriptions**, or select **Subscriptions** on the **Home** page.



2. Select the particular subscription to assign the application to.



   If you don't see the subscription you're looking for, select **global subscriptions filter**. Make sure the subscription you want is selected for the portal.

3. Select **Access control (IAM)**.

4. Select Select **Add** > **Add role assignment** to open the **Add role assignment** page.

5. Select the role you wish to assign to the application. For example, to allow the application to execute actions like **reboot**, **start** and **stop** instances, select the **Contributor** role. Read more about the available roles By default, Azure AD applications aren't displayed in the available options. To find your application, search for the name and select it.

   Assign the Contributor role to the application at the subscription scope. For detailed steps, see Assign Azure roles using the Azure portal.

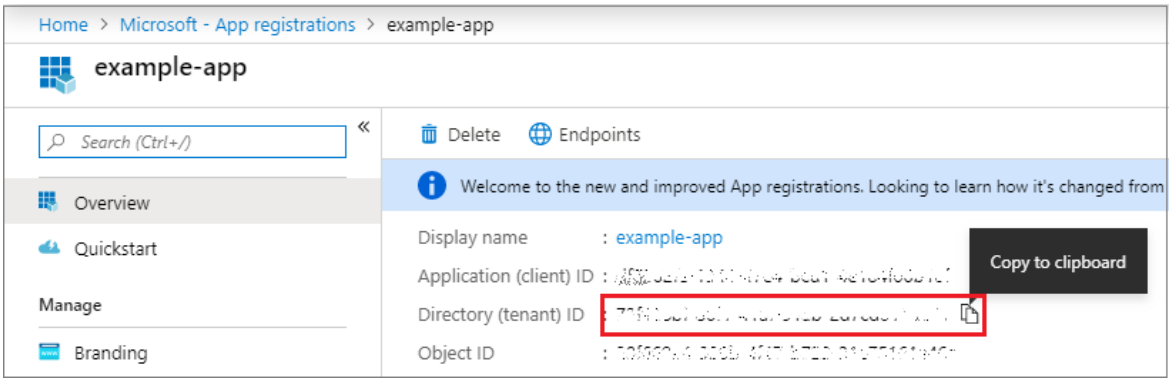
Your service principal is set up. You can start using it to run your scripts or apps. To manage your service principal (permissions, user consented permissions, see which users have consented, review permissions, see sign in information, and more), go to **Enterprise applications**.

The next section shows how to get values that are needed when signing in programmatically.

Get tenant and app ID values for signing in

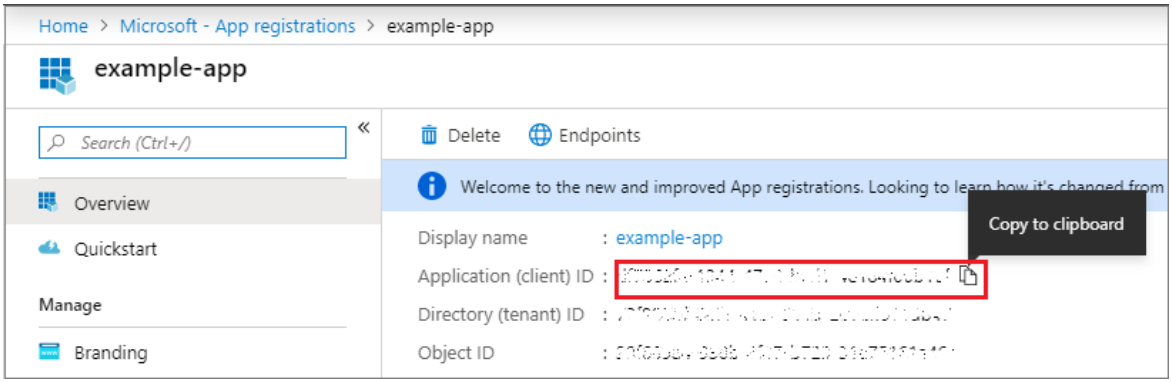
When programmatically signing in, pass the tenant ID with your authentication request and the application ID. You also need a certificate or an authentication key (described in the following section). To get those values, use the following steps:

1. Select **Azure Active Directory**.
2. From **App registrations** in Azure AD, select your application.
3. Copy the Directory (tenant) ID and store it in your application code.



The directory (tenant) ID can also be found in the default directory overview page.

4. Copy the **Application ID** and store it in your application code.



Authentication: Two options

There are two types of authentication available for service principals: password-based authentication (application secret) and certificate-based authentication. *We recommend using a certificate*, but you can also create an application secret.

Option 1: Upload a certificate

You can use an existing certificate if you have one. Optionally, you can create a self-signed certificate for *testing purposes only*. To create a self-signed certificate, open PowerShell and run New-SelfSignedCertificate with the following parameters to create the cert in the user certificate store on your computer:

```
$cert=New-SelfSignedCertificate -Subject "CN=DaemonConsoleCert" -CertStoreLocation "Cert:\CurrentUser\My"  -
KeyExportPolicy Exportable -KeySpec Signature
```

Export this certificate to a file using the Manage User Certificate MMC snap-in accessible from the Windows Control Panel.

1. Select **Run** from the **Start** menu, and then enter **certmgr.msc**.

   The Certificate Manager tool for the current user appears.

2. To view your certificates, under **Certificates - Current User** in the left pane, expand the **Personal** directory.

3. Right-click on the cert you created, select **All tasks->Export**.

4. Follow the Certificate Export wizard. Do not export the private key, and export to a .CER file.

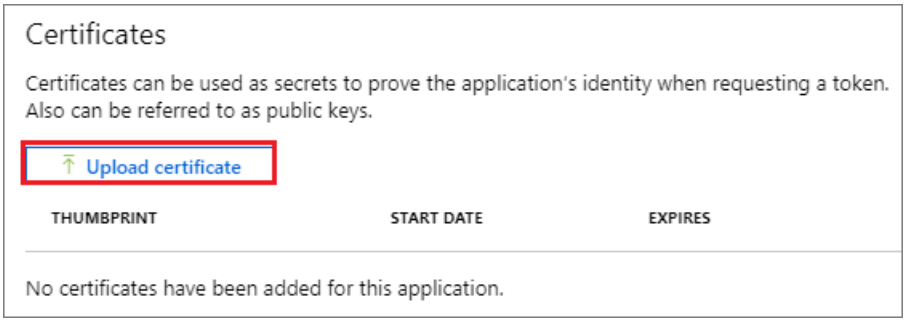
To upload the certificate:

1. Select **Azure Active Directory**.

2. From **App registrations** in Azure AD, select your application.

3. Select **Certificates & secrets**.

4. Select **Upload certificate** and select the certificate (an existing certificate or the self-signed certificate you exported).



5. Select **Add**.

After registering the certificate with your application in the application registration portal, enable the client application code to use the certificate.

Option 2: Create a new application secret

If you choose not to use a certificate, you can create a new application secret.

1. Select **Azure Active Directory**.

2. From **App registrations** in Azure AD, select your application.

3. Select **Certificates & secrets**.

4. Select **Client secrets -> New client secret**.

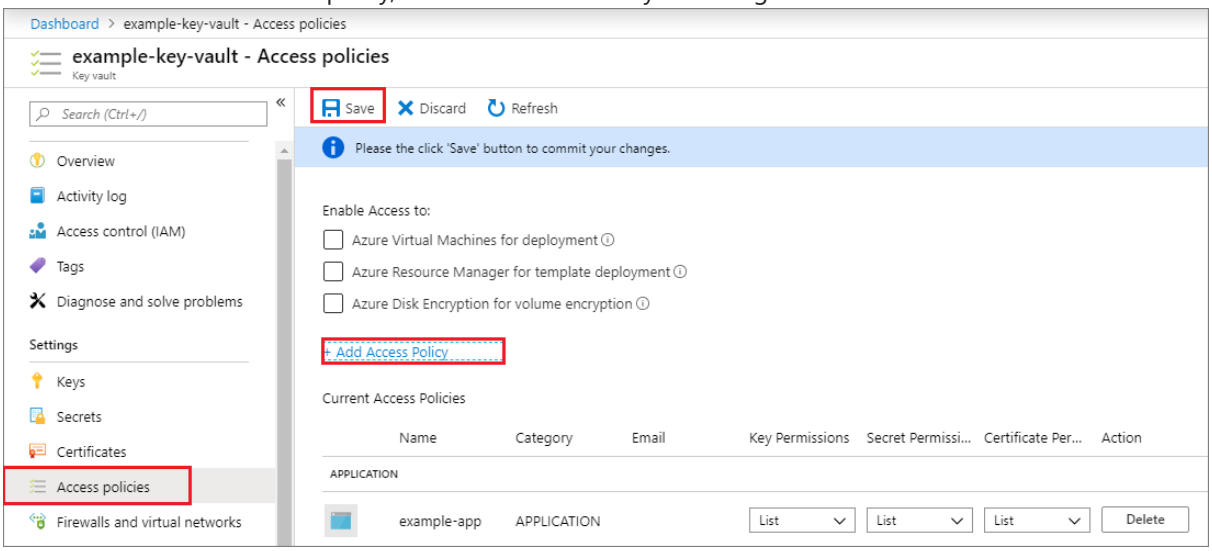
5. Provide a description of the secret, and a duration. When done, select **Add**.

   After saving the client secret, the value of the client secret is displayed. Copy this value because you won't be able to retrieve the key later. You will provide the key value with the application ID to sign in as the application. Store the key value where your application can retrieve it.

Configure access policies on resources

Keep in mind, you might need to configure additional permissions on resources that your application needs to access. For example, you must also update a key vault's access policies to give your application access to keys, secrets, or certificates.

1. In the Azure portal, navigate to your key vault and select **Access policies**.
2. Select **Add access policy**, then select the key, secret, and certificate permissions you want to grant your application. Select the service principal you created previously.
3. Select **Add** to add the access policy, then **Save** to commit your changes.



Next steps

- Learn how to use Azure PowerShell to create a service principal.
- To learn about specifying security policies, see Azure role-based access control (Azure RBAC).
- For a list of available actions that can be granted or denied to users, see Azure Resource Manager Resource Provider operations.
- For information about working with app registrations by using **Microsoft Graph**, see the Applications API reference.

Use Azure PowerShell to create a service principal with a certificate

8/27/2021 • 6 minutes to read • Edit Online

When you have an app or script that needs to access resources, you can set up an identity for the app and authenticate the app with its own credentials. This identity is known as a service principal. This approach enables you to:

- Assign permissions to the app identity that are different than your own permissions. Typically, these permissions are restricted to exactly what the app needs to do.
- Use a certificate for authentication when executing an unattended script.

> **IMPORTANT**
> Instead of creating a service principal, consider using managed identities for Azure resources for your application identity. If your code runs on a service that supports managed identities and accesses resources that support Azure Active Directory (Azure AD) authentication, managed identities are a better option for you. To learn more about managed identities for Azure resources, including which services currently support it, see What is managed identities for Azure resources?.

This article shows you how to create a service principal that authenticates with a certificate. To set up a service principal with password, see Create an Azure service principal with Azure PowerShell.

You must have the latest version of PowerShell for this article.

> **NOTE**
> This article has been updated to use the Azure Az PowerShell module. The Az PowerShell module is the recommended PowerShell module for interacting with Azure. To get started with the Az PowerShell module, see Install Azure PowerShell. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Required permissions

To complete this article, you must have sufficient permissions in both your Azure AD and Azure subscription. Specifically, you must be able to create an app in the Azure AD, and assign the service principal to a role.

The easiest way to check whether your account has adequate permissions is through the portal. See Check required permission.

Assign the application to a role

To access resources in your subscription, you must assign the application to a role. Decide which role offers the right permissions for the application. To learn about the available roles, see Azure built-in roles.

You can set the scope at the level of the subscription, resource group, or resource. Permissions are inherited to lower levels of scope. For example, adding an application to the *Reader* role for a resource group means it can read the resource group and any resources it contains. To allow the application to execute actions like reboot, start and stop instances, select the *Contributor* role.

Create service principal with self-signed certificate

The following example covers a simple scenario. It uses New-AzADServicePrincipal to create a service principal with a self-signed certificate, and uses New-AzRoleAssignment to assign the Reader role to the service principal. The role assignment is scoped to your currently selected Azure subscription. To select a different subscription, use Set-AzContext.

> **NOTE**
> The New-SelfSignedCertificate cmdlet and the PKI module are currently not supported in PowerShell Core.

```
$cert = New-SelfSignedCertificate -CertStoreLocation "cert:\CurrentUser\My" `
  -Subject "CN=exampleappScriptCert" `
  -KeySpec KeyExchange
$keyValue = [System.Convert]::ToBase64String($cert.GetRawCertData())

$sp = New-AzADServicePrincipal -DisplayName exampleapp `
  -CertValue $keyValue `
  -EndDate $cert.NotAfter `
  -StartDate $cert.NotBefore
Sleep 20
New-AzRoleAssignment -RoleDefinitionName Reader -ServicePrincipalName $sp.ApplicationId
```

The example sleeps for 20 seconds to allow some time for the new service principal to propagate throughout Azure AD. If your script doesn't wait long enough, you'll see an error stating: "Principal {ID} does not exist in the directory {DIR-ID}." To resolve this error, wait a moment then run the **New-AzRoleAssignment** command again.

You can scope the role assignment to a specific resource group by using the **ResourceGroupName** parameter. You can scope to a specific resource by also using the **ResourceType** and **ResourceName** parameters.

If you **do not have Windows 10 or Windows Server 2016**, download the New-SelfSignedCertificateEx cmdlet from PKI Solutions. Extract its contents and import the cmdlet you need.

```
# Only run if you could not use New-SelfSignedCertificate
Import-Module -Name c:\ExtractedModule\New-SelfSignedCertificateEx.ps1
```

In the script, substitute the following two lines to generate the certificate.

```
New-SelfSignedCertificateEx -StoreLocation CurrentUser `
  -Subject "CN=exampleapp" `
  -KeySpec "Exchange" `
  -FriendlyName "exampleapp"
$cert = Get-ChildItem -path Cert:\CurrentUser\my | where {$PSitem.Subject -eq 'CN=exampleapp' }
```

Provide certificate through automated PowerShell script

Whenever you sign in as a service principal, provide the tenant ID of the directory for your AD app. A tenant is an instance of Azure AD.

```
$TenantId = (Get-AzSubscription -SubscriptionName "Contoso Default").TenantId
$ApplicationId = (Get-AzADApplication -DisplayNameStartWith exampleapp).ApplicationId

$Thumbprint = (Get-ChildItem cert:\CurrentUser\My\ | Where-Object {$_.Subject -eq "CN=exampleappScriptCert"
}).Thumbprint
Connect-AzAccount -ServicePrincipal `
  -CertificateThumbprint $Thumbprint `
  -ApplicationId $ApplicationId `
  -TenantId $TenantId
```

Create service principal with certificate from Certificate Authority

The following example uses a certificate issued from a Certificate Authority to create service principal. The assignment is scoped to the specified Azure subscription. It adds the service principal to the Reader role. If an error occurs during the role assignment, it retries the assignment.

```
Param (
 [Parameter(Mandatory=$true)]
 [String] $ApplicationDisplayName,

 [Parameter(Mandatory=$true)]
 [String] $SubscriptionId,

 [Parameter(Mandatory=$true)]
 [String] $CertPath,

 [Parameter(Mandatory=$true)]
 [String] $CertPlainPassword
 )

 Connect-AzAccount
 Import-Module Az.Resources
 Set-AzContext -Subscription $SubscriptionId

 $CertPassword = ConvertTo-SecureString $CertPlainPassword -AsPlainText -Force

 $PFXCert = New-Object -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2 -
ArgumentList @($CertPath, $CertPassword)
 $KeyValue = [System.Convert]::ToBase64String($PFXCert.GetRawCertData())

 $ServicePrincipal = New-AzADServicePrincipal -DisplayName $ApplicationDisplayName
 New-AzADSpCredential -ObjectId $ServicePrincipal.Id -CertValue $KeyValue -StartDate $PFXCert.NotBefore -
EndDate $PFXCert.NotAfter
 Get-AzADServicePrincipal -ObjectId $ServicePrincipal.Id

 $NewRole = $null
 $Retries = 0;
 While ($NewRole -eq $null -and $Retries -le 6)
 {
    # Sleep here for a few seconds to allow the service principal application to become active (should only
take a couple of seconds normally)
    Sleep 15
    New-AzRoleAssignment -RoleDefinitionName Reader -ServicePrincipalName $ServicePrincipal.ApplicationId |
Write-Verbose -ErrorAction SilentlyContinue
    $NewRole = Get-AzRoleAssignment -ObjectId $ServicePrincipal.Id -ErrorAction SilentlyContinue
    $Retries++;
 }

 $NewRole
```

Provide certificate through automated PowerShell script

Whenever you sign in as a service principal, provide the tenant ID of the directory for your AD app. A tenant is

an instance of Azure AD.

```
Param (

 [Parameter(Mandatory=$true)]
 [String] $CertPath,

 [Parameter(Mandatory=$true)]
 [String] $CertPlainPassword,

 [Parameter(Mandatory=$true)]
 [String] $ApplicationId,

 [Parameter(Mandatory=$true)]
 [String] $TenantId
 )

 $CertPassword = ConvertTo-SecureString $CertPlainPassword -AsPlainText -Force
 $PFXCert = New-Object `
  -TypeName System.Security.Cryptography.X509Certificates.X509Certificate2 `
  -ArgumentList @($CertPath, $CertPassword)
 $Thumbprint = $PFXCert.Thumbprint

 Connect-AzAccount -ServicePrincipal `
  -CertificateThumbprint $Thumbprint `
  -ApplicationId $ApplicationId `
  -TenantId $TenantId
```

The application ID and tenant ID aren't sensitive, so you can embed them directly in your script. If you need to retrieve the tenant ID, use:

```
(Get-AzSubscription -SubscriptionName "Contoso Default").TenantId
```

If you need to retrieve the application ID, use:

```
(Get-AzADApplication -DisplayNameStartWith {display-name}).ApplicationId
```

Change credentials

To change the credentials for an AD app, either because of a security compromise or a credential expiration, use the Remove-AzADAppCredential and New-AzADAppCredential cmdlets.

To remove all the credentials for an application, use:

```
Get-AzADApplication -DisplayName exampleapp | Remove-AzADAppCredential
```

To add a certificate value, create a self-signed certificate as shown in this article. Then, use:

```
Get-AzADApplication -DisplayName exampleapp | New-AzADAppCredential `
  -CertValue $keyValue `
  -EndDate $cert.NotAfter `
  -StartDate $cert.NotBefore
```

Debug

You may get the following errors when creating a service principal:

- **"Authentication_Unauthorized"** or **"No subscription found in the context."** - You see this error when your account doesn't have the required permissions on the Azure AD to register an app. Typically, you see this error when only admin users in your Azure Active Directory can register apps, and your account isn't an admin. Ask your administrator to either assign you to an administrator role, or to enable users to register apps.
- Your account **"does not have authorization to perform action 'Microsoft.Authorization/roleAssignments/write' over scope '/subscriptions/{guid}'."** - You see this error when your account doesn't have sufficient permissions to assign a role to an identity. Ask your subscription administrator to add you to User Access Administrator role.

Next steps

- To set up a service principal with password, see Create an Azure service principal with Azure PowerShell.
- For a more detailed explanation of applications and service principals, see Application Objects and Service Principal Objects.
- For more information about Azure AD authentication, see Authentication Scenarios for Azure AD.
- For information about working with app registrations by using **Microsoft Graph**, see the Applications API reference.

Add a service principal to the server administrator role

5/14/2021 • 2 minutes to read • Edit Online

To automate unattended PowerShell tasks, a service principal must have **server administrator** privileges on the Analysis Services server being managed. This article describes how to add a service principal to the server administrators role on an Azure AS server. You can do this using SQL Server Management Studio or a Resource Manager template.

> **NOTE**
> Service principals must be added directly to the server administrator role. Adding a service principal to a security group, and then adding that security group to the server administrator role is not supported.

Before you begin

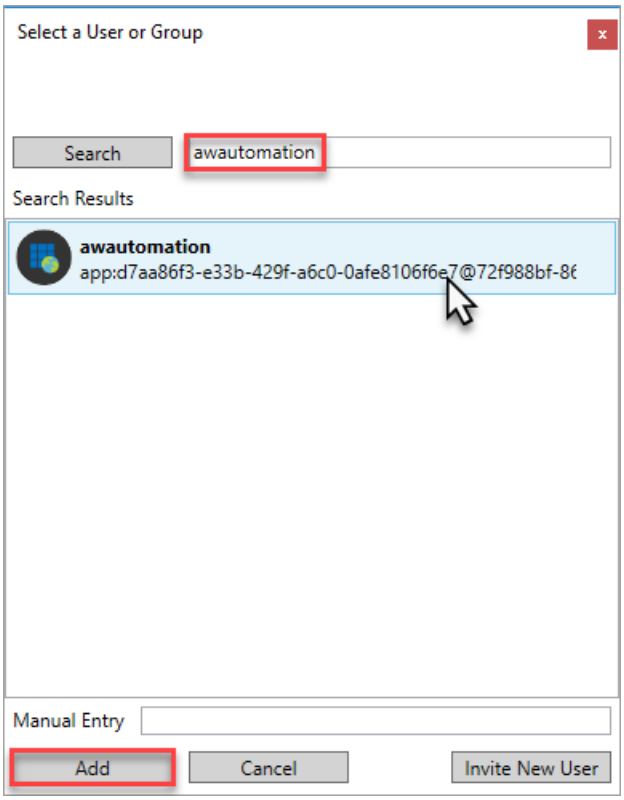
Before completing this task, you must have a service principal registered in Azure Active Directory.

Create service principal - Azure portal
Create service principal - PowerShell

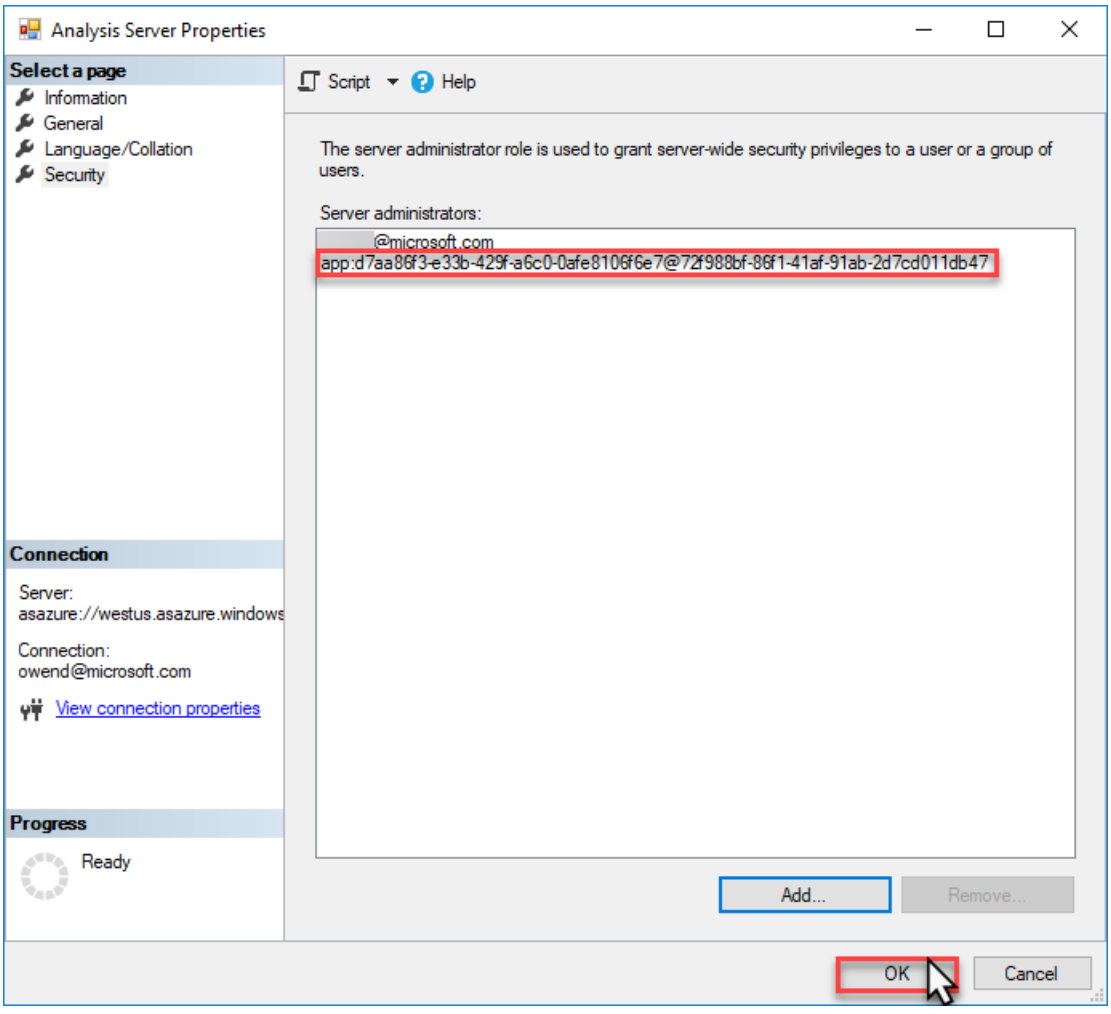
Using SQL Server Management Studio

You can configure server administrators using SQL Server Management Studio (SSMS). To complete this task, you must have server administrator permissions on the Azure AS server.

1. In SSMS, connect to your Azure AS server.
2. In **Server Properties** > **Security**, click **Add**.
3. In **Select a User or Group**, search for your registered app by name, select, and then click **Add**.



4. Verify the service principal account ID, and then click **OK**.



Using a Resource Manager template

You can also configure server administrators by deploying the Analysis Services server using an Azure Resource Manager template. The identity running the deployment must belong to the **Contributor** role for the resource

in Azure role-based access control (Azure RBAC).

> **IMPORTANT**
> The service principal must be added using the format `app:{service-principal-client-id}@{azure-ad-tenant-id}`.

The following Resource Manager template deploys an Analysis Services server with a specified service principal added to the Analysis Services Admin role:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "analysisServicesServerName": {
            "type": "string"
        },
        "location": {
            "type": "string"
        },
        "analysisServicesSkuName": {
            "type": "string"
        },
        "analysisServicesCapacity": {
            "type": "int"
        },
        "servicePrincipalClientId": {
            "type": "string"
        },
        "servicePrincipalTenantId": {
            "type": "string"
        }
    },
    "resources": [
        {
            "name": "[parameters('analysisServicesServerName')]",
            "type": "Microsoft.AnalysisServices/servers",
            "apiVersion": "2017-08-01",
            "location": "[parameters('location')]",
            "sku": {
                "name": "[parameters('analysisServicesSkuName')]",
                "capacity": "[parameters('analysisServicesCapacity')]"
            },
            "properties": {
                "asAdministrators": {
                    "members": [
                        "[concat('app:', parameters('servicePrincipalClientId'), '@',
parameters('servicePrincipalTenantId'))]"
                    ]
                }
            }
        }
    ]
}
```

Using managed identities

A managed identity can also be added to the Analysis Services Admins list. For example, you might have a Logic App with a system-assigned managed identity, and want to grant it the ability to administer your Analysis Services server.

In most parts of the Azure portal and APIs, managed identities are identified using their service principal object ID. However, Analysis Services requires that they be identified using their client ID. To obtain the client ID for a

service principal, you can use the Azure CLI:

```
az ad sp show --id <ManagedIdentityServicePrincipalObjectId> --query appId -o tsv
```

Alternatively you can use PowerShell:

```
(Get-AzureADServicePrincipal -ObjectId <ManagedIdentityServicePrincipalObjectId>).AppId
```

You can then use this client ID in conjunction with the tenant ID to add the managed identity to the Analysis Services Admins list, as described above.

Related information

- Download SQL Server PowerShell Module
- Download SSMS

Move Analysis Services to a different region

6/28/2021 • 8 minutes to read • Edit Online

This article describes how to move an Analysis Services server resource to a different Azure region. You might move your server to another region for a number of reasons, for example, to take advantage of an Azure region closer to users, to use service plans supported in specific regions only, or to meet internal policy and governance requirements.

In this and associated linked articles, you learn how to:

- Backup a source server model database to Blob storage.
- Export a source server resource template.
- Get a storage shared access signature (SAS).
- Modify the resource template.
- Deploy the template to create a new target server.
- Restore a model database to the new target server.
- Verify the new target server and database.
- Delete the source server.

This article describes using a resource template to migrate a single Analysis Services server with a **basic configuration** to a different region *and* resource group in the same subscription. Using a template retains configured server properties ensuring the target server is configured with the same properties, except region and resource group, as the source server. This article does not describe moving associated resources that may be part of the same resource group such as data source, storage, and gateway resources.

Before moving a server to a different region, it's recommended you create a detailed plan. Consider additional resources such as gateways and storage that may also need to be moved. With any plan, it's important to complete one or more trial move operations using test servers prior to moving a production server.

IMPORTANT

Client applications and connection strings connect to Analysis Services by using the full server name, which is a Uri that includes the region the server is in. For example, `asazure://westcentralus.asazure.windows.net/advworks01`. When moving a server to a different region, you are effectively creating a new server resource in a different region, which will have a different region in the server name Uri. Client applications and connection strings used in scripts must connect to the new server using the new server name Uri. Using a Server name alias can mitigate the number of places the server name Uri has to be changed, but must be implemented prior to a region move.

IMPORTANT

Azure regions use different IP address ranges. If you have firewall exceptions configured for the region your server and/or storage account is in, it may be necessary to configure a different IP address range. To learn more, see Frequently asked questions about Analysis Services network connectivity.

> **NOTE**
> This article describes restoring a database backup to a target server from a storage container in the source server's region. In some cases, restoring backups from a different region can have poor performance, especially for large databases. For the best performance during database restore, migrate or create a a new storage container in the target server region. Copy the .abf backup files from the source region storage container to the target region storage container prior to restoring the database to the target server. While out of scope for this article, in some cases, particularly with very large databases, scripting out a database from your source server, recreating, and then processing on the target server to load database data may be more cost effective than using backup/restore.

> **NOTE**
> If using an On-premises data gateway to connect to data sources, you must also move the gateway resource to the target server region. To learn more, see Install and configure an on-premises data gateway.

Prerequisites

- **Azure storage account**: Required to store an .abf backup file.
- **SQL Server Management Studio (SSMS)**: Required to backup and restore model databases.
- **Azure PowerShell**. Required only if you choose to complete this task by using PowerShell.

Prepare

Backup model databases

If **Storage settings** are not already configured for the source server, follow the steps in Configure storage settings.

When storage settings are configured, follow the steps in Backup to create a model database .abf backup in your storage container. You later restore the .abf backup to your new target server.

Export template

The template contains configuration properties of the source server.

- Portal
- PowerShell

To export a template by using Azure portal:

1. Sign in to the Azure portal.
2. Select **All resources**, and then select your Analysis Services server.
3. Select > **Settings** > **Export template**.
4. Choose **Download** in the **Export template** blade.
5. Locate the .zip file that you downloaded from the portal, and then unzip that file to a folder.

   The zip file contains the .json files that comprise the template and parameters necessary to deploy a new server.

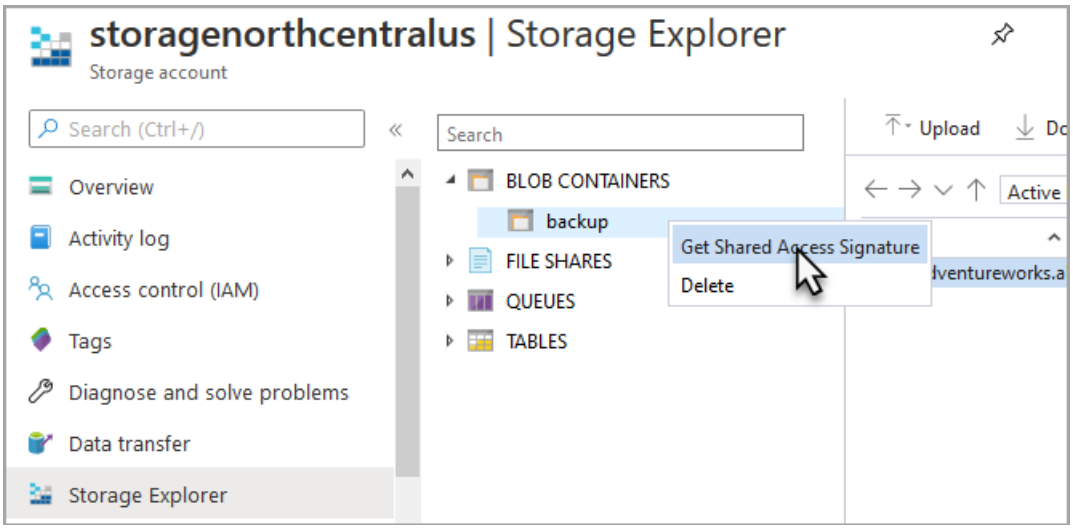
Get storage shared access signature (SAS)

When deploying a target server from a template, a user delegation SAS token (as a Uri) is required to specify the storage container containing the database backup.

- Portal
- PowerShell

To get a shared access signature by using the portal:

1. In the portal, select the storage account used to backup your server database.
2. Select **Storage Explorer**, and then expand **BLOB CONTAINERS**.
3. Right-click your storage container, and then select **Get Shared Access Signature**.



4. In **Shared Access Signature**, select **Create**. By default, the SAS will expire in 24 hours.
5. Copy and save the **URI**.

Modify the template

Use a text editor to modify the template.json file you exported, changing the region and blob container properties.

To modify the template:

1. In a text editor, in the **location** property, specify the new target region. In the **backupBlobContainerUri** property, paste the storage container Uri with SAS key.

   The following example sets the target region for server advworks1 to `South Central US` and specifies the storage container Uri with shared access signature:

```
"resources": [
    {
        "type": "Microsoft.AnalysisServices/servers",
        "apiVersion": "2017-08-01",
        "name": "[parameters('servers_advworks1_name')]",
        "location": "South Central US",
        "sku": {
            "name": "S1",
            "tier": "Standard",
            "capacity": 1
        },
        "properties": {
            "asAdministrators": {
                "members": [
                    "asadmins@adventure-works.com"
                ]
            },
            "backupBlobContainerUri": "https://storagenorthcentralus.blob.core.windows.net/backup?
sp=rl&st=2020-06-01T19:30:42Z&se=2020-06-02T19:30:42Z&sv=2019-10-
10&sr=c&sig=PCQ4s9RujJkxu89gO4tiDTbE3%2BFECx6zAdcv8x0cVUQ%3D",
            "querypoolConnectionMode": "All"
        }
    }
]
```

2. Save the template.

Regions

To get Azure regions, see Azure locations. To get regions by using PowerShell, run the Get-AzLocation command.

```
Get-AzLocation | format-table
```

Move

To deploy a new server resource in a different region, you'll use the **template.json** file you exported and modified in the previous sections.

- Portal
- PowerShell

1. In the portal, select **Create a resource**.
2. In **Search the Marketplace**, type **template deployment**, and then press **ENTER**.
3. Select **Template deployment**.
4. Select **Create**.
5. Select **Build your own template in the editor**.
6. Select **Load file**, and then follow the instructions to load the **template.json** file you exported and modified.
7. Verify the template editor shows the correct properties for your new target server.
8. Select **Save**.
9. Enter or select the property values:
    - **Subscription**: Select the Azure subscription.

- **Resource group**: Select **Create new**, and then enter a resource group name. You can select an existing resource group provided it does not already contain an Analysis Services server with the same name.
- **Location**: Select the same region you specified in the template.

10. Select **Review and Create**.
11. Review the terms and Basics, and then select **Create**.

Get target server Uri

In order to connect to the new target server from SSMS to restore the model database, you need to get the new target server Uri.

- Portal
- PowerShell

To get the server Uri in the portal:

1. In the portal, go to the new target server resource.
2. On the **Overview** page, copy the **Server name** Uri.

Restore model database

Follow steps described in Restore to restore the model database .abf backup to the new target server.

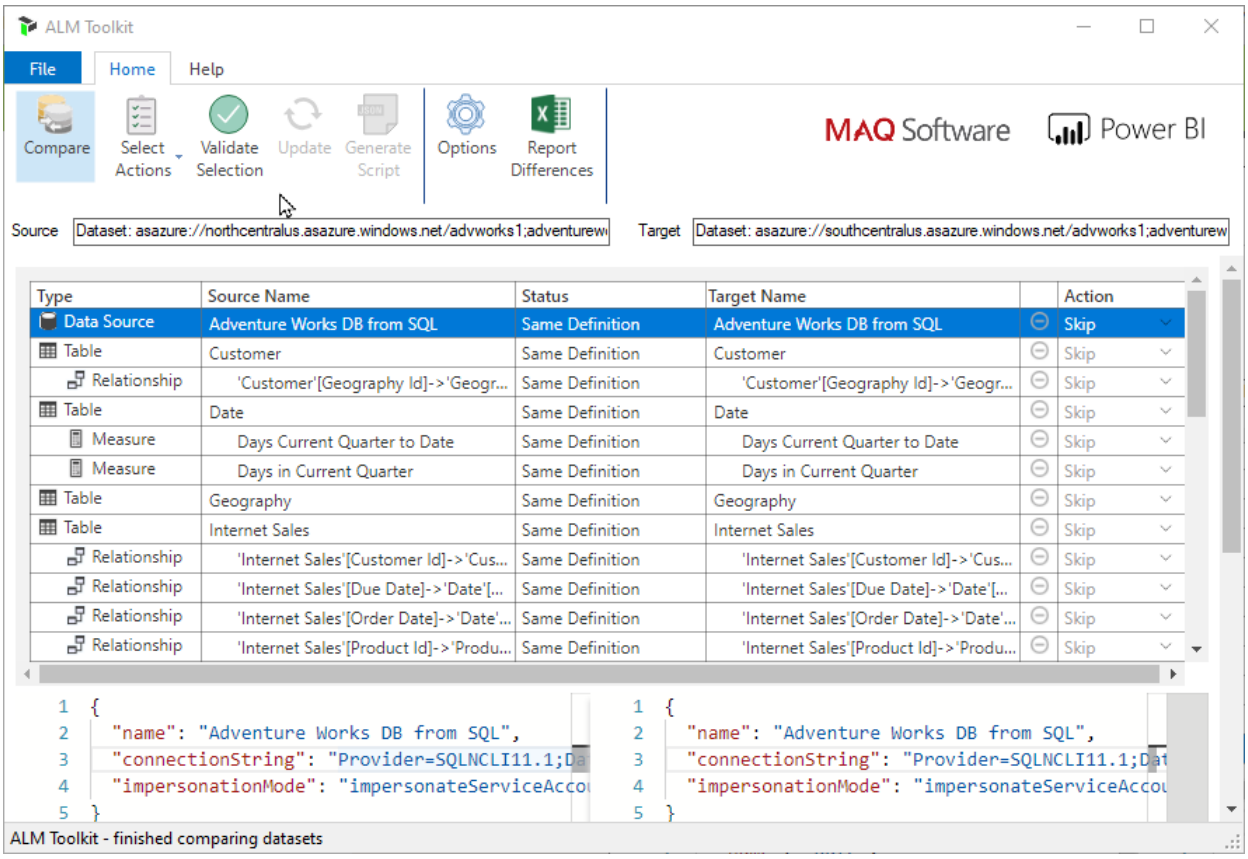
Optional: After restoring the model database, process the model and tables to refresh data from data sources. To process the model and table by using SSMS:

1. In SSMS, right-click the model database > **Process Database**.
2. Expand **Tables**, right-click a table. In **Process Table(s)**, select all tables, and then select **OK**.

Verify

1. In the portal, go to the new target server.
2. On the Overview page, in **Models on Analysis Services server**, verify restored models appear.
3. Use a client application like Power BI or Excel to connect to the model on the new server. Verify model objects such as tables, measures, hierarchies appear.
4. Run any automation scripts. Verify they executed successfully.

Optional: ALM Toolkit is an *open source* tool for comparing and managing Power BI Datasets *and* Analysis Services tabular model databases. Use the toolkit to connect to both source and target server databases and compare. If your database migration is successful, model objects will the same definition.



Clean up resources

After verifying client applications can connect to the new server and any automation scripts are executing correctly, delete your source server.

- Portal
- PowerShell

To delete the source server from the portal:

In your source server's **Overview** page, select **Delete**.

> **NOTE**
> After completing a region move, it's recommended your new target server use a storage container in the same region for backups, rather than the storage container in the source server region.

Deploy a model from Visual Studio

3/5/2021 • 2 minutes to read • Edit Online

Once you've created a server in your Azure subscription, you're ready to deploy a tabular model database to it. You can use Visual Studio with Analysis Services projects to build and deploy a tabular model project you're working on.

Prerequisites

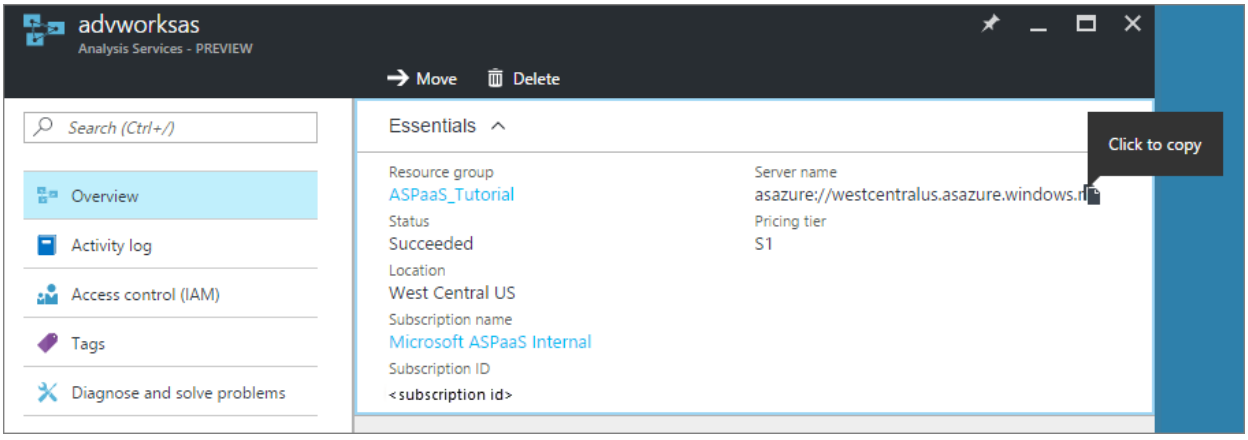
To get started, you need:

- **Analysis Services server** in Azure. To learn more, see Create an Azure Analysis Services server.
- **Tabular model project** in Visual Studio or an existing tabular model at the 1200 or higher compatibility level. Never created one? Try the Adventure Works Internet sales tabular modeling tutorial.
- **On-premises gateway** - If one or more data sources are on-premises in your organization's network, you need to install an On-premises data gateway. The gateway is necessary for your server in the cloud connect to your on-premises data sources to process and refresh data in the model.

> **TIP**
> Before you deploy, make sure you can process the data in your tables. In Visual Studio, click **Model** > **Process** > **Process All**. If processing fails, you cannot successfully deploy.

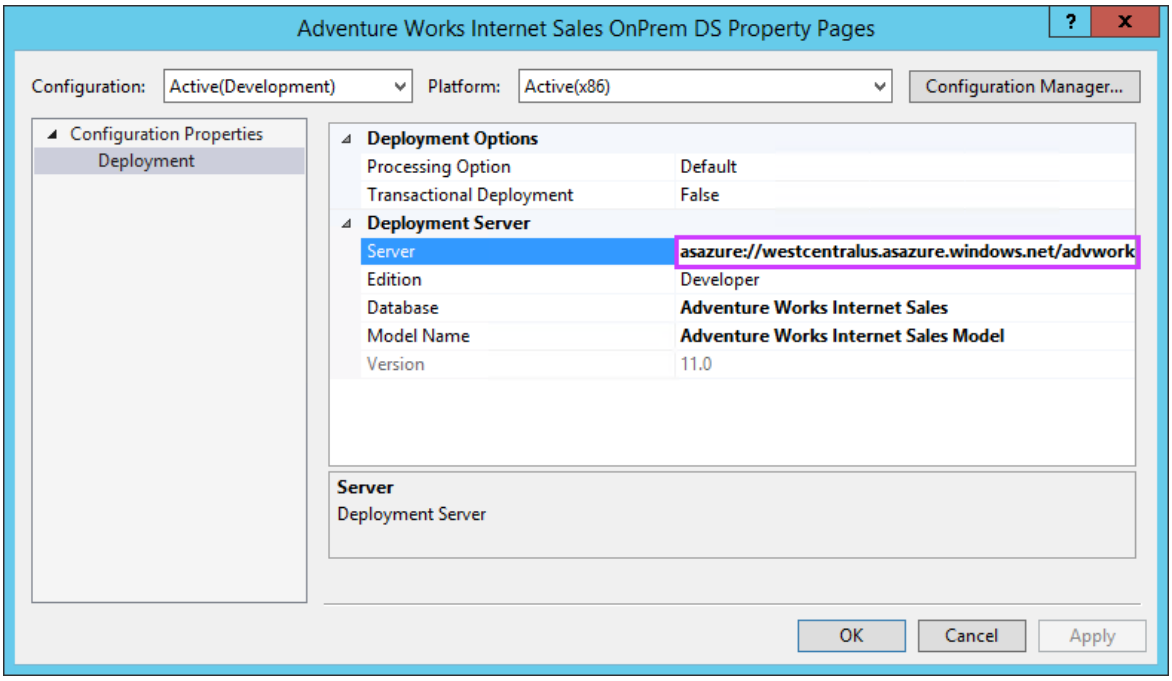
Get the server name

In **Azure portal** > server > **Overview** > **Server name**, copy the server name.

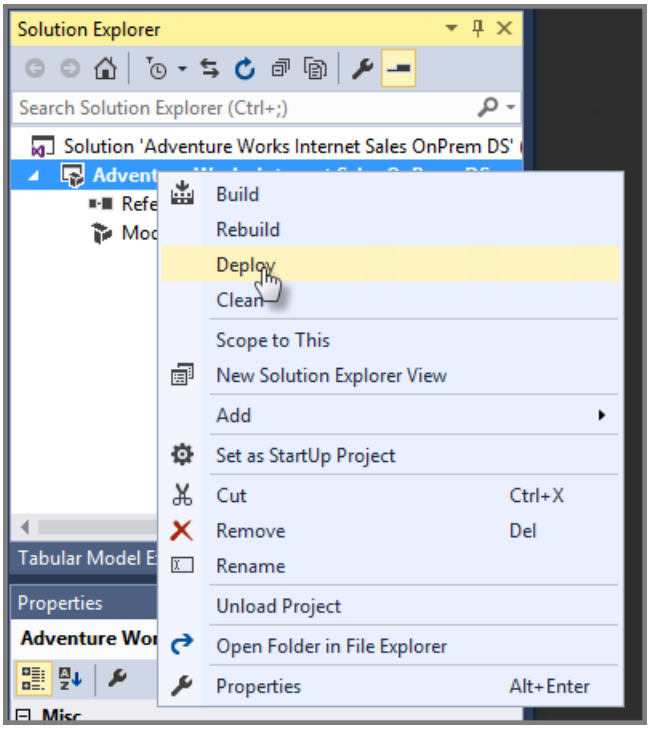


To deploy from Visual Studio

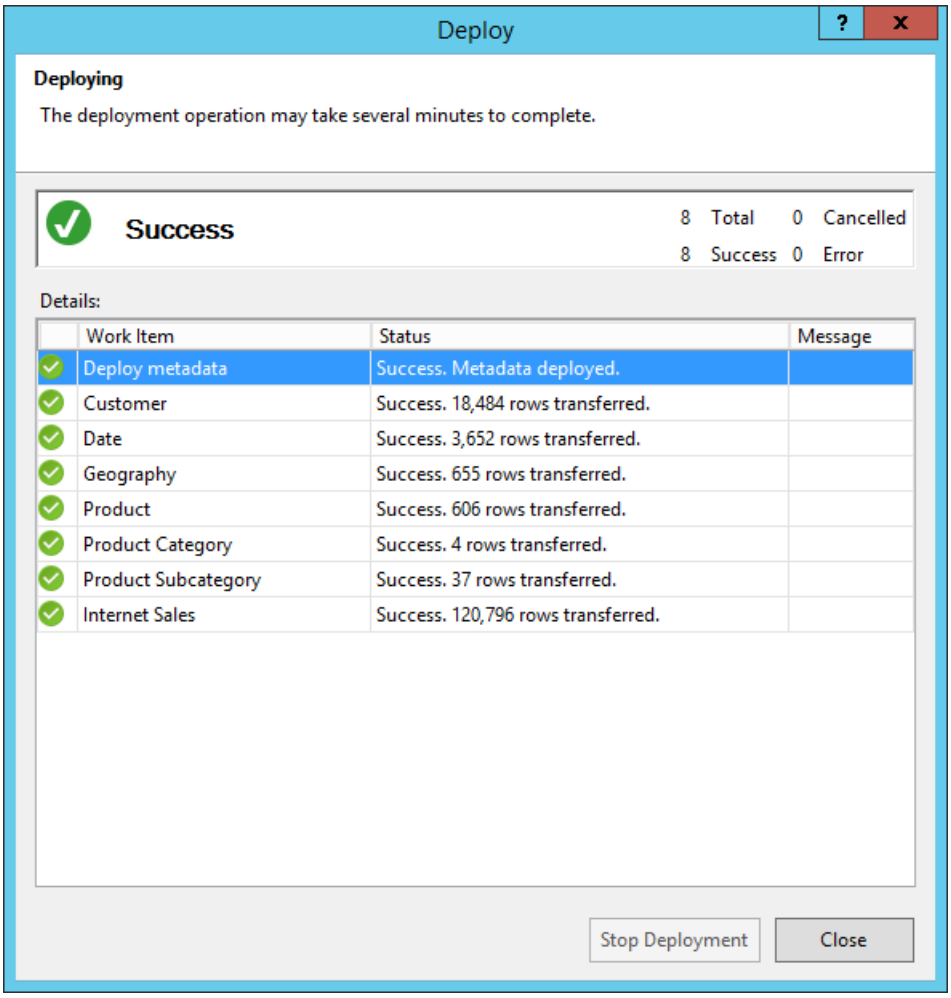
1. In Visual Studio > **Solution Explorer**, right-click the project > **Properties**. Then in **Deployment** > **Server** paste the server name.



2. In **Solution Explorer**, right-click **Properties**, then click **Deploy**. You may be prompted to sign in to Azure.



Deployment status appears in both the Output window and in Deploy.



That's all there is to it!

Troubleshooting

If deployment fails when deploying metadata, it's likely because Visual Studio couldn't connect to your server. Make sure you can connect to your server using SQL Server Management Studio (SSMS). Then make sure the Deployment Server property for the project is correct.

If deployment fails on a table, it's likely because your server couldn't connect to a data source. If your data source is on-premises in your organization's network, be sure to install an On-premises data gateway.

Next steps

Now that you have your tabular model deployed to your server, you're ready to connect to it. You can connect to it with SQL Server Management Studio (SSMS) to manage it. And, you can connect to it using a client tool like Power BI, Power BI Desktop, or Excel, and start creating reports.

To learn about advanced deployment methods, see Tabular model solution deployment.

Asynchronous refresh with the REST API

11/2/2020 • 5 minutes to read • Edit Online

By using any programming language that supports REST calls, you can perform asynchronous data-refresh operations on your Azure Analysis Services tabular models. This includes synchronization of read-only replicas for query scale-out.

Data-refresh operations can take some time depending on a number of factors including data volume, level of optimization using partitions, etc. These operations have traditionally been invoked with existing methods such as using TOM (Tabular Object Model), PowerShell cmdlets, or TMSL (Tabular Model Scripting Language). However, these methods can require often unreliable, long-running HTTP connections.

The REST API for Azure Analysis Services enables data-refresh operations to be carried out asynchronously. By using the REST API, long-running HTTP connections from client applications aren't necessary. There are also other built-in features for reliability, such as auto retries and batched commits.

Base URL

The base URL follows this format:

```
https://<rollout>.asazure.windows.net/servers/<serverName>/models/<resource>/
```

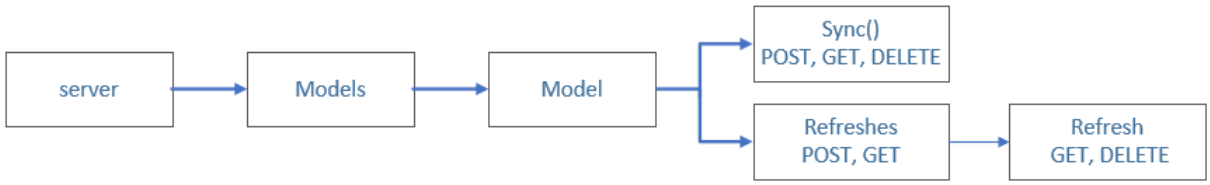
For example, consider a model named AdventureWorks on a server named `myserver`, located in the West US Azure region. The server name is:

```
asazure://westus.asazure.windows.net/myserver
```

The base URL for this server name is:

```
https://westus.asazure.windows.net/servers/myserver/models/AdventureWorks/
```

By using the base URL, resources and operations can be appended based on the following parameters:



- Anything that ends in **s** is a collection.
- Anything that ends with **()** is a function.
- Anything else is a resource/object.

For example, you can use the POST verb on the Refreshes collection to perform a refresh operation:

```
https://westus.asazure.windows.net/servers/myserver/models/AdventureWorks/refreshes
```

Authentication

All calls must be authenticated with a valid Azure Active Directory (OAuth 2) token in the Authorization header and must meet the following requirements:

- The token must be either a user token or an application service principal.
- The token must have the correct audience set to `https://*.asazure.windows.net`.
- The user or application must have sufficient permissions on the server or model to make the requested call. The permission level is determined by roles within the model or the admin group on the server.

> **IMPORTANT**
> Currently, **server admin** role permissions are necessary.

POST /refreshes

To perform a refresh operation, use the POST verb on the /refreshes collection to add a new refresh item to the collection. The Location header in the response includes the refresh ID. The client application can disconnect and check the status later if required because it is asynchronous.

Only one refresh operation is accepted at a time for a model. If there's a current running refresh operation and another is submitted, the 409 Conflict HTTP status code is returned.

The body may resemble the following:

```
{
    "Type": "Full",
    "CommitMode": "transactional",
    "MaxParallelism": 2,
    "RetryCount": 2,
    "Objects": [
        {
            "table": "DimCustomer",
            "partition": "DimCustomer"
        },
        {
            "table": "DimDate"
        }
    ]
}
```

Parameters

Specifying parameters is not required. The default is applied.

NAME	TYPE	DESCRIPTION	DEFAULT
`Type`	Enum	The type of processing to perform. The types are aligned with the TMSL refresh command types: full, clearValues, calculate, dataOnly, automatic, and defragment. Add type is not supported.	automatic

NAME	TYPE	DESCRIPTION	DEFAULT
`CommitMode`	Enum	Determines if objects will be committed in batches or only when complete. Modes include: default, transactional, partialBatch.	transactional
`MaxParallelism`	Int	This value determines the maximum number of threads on which to run processing commands in parallel. This value aligned with the MaxParallelism property that can be set in the TMSL Sequence command or using other methods.	10
`RetryCount`	Int	Indicates the number of times the operation will retry before failing.	0
`Objects`	Array	An array of objects to be processed. Each object includes: "table" when processing the entire table or "table" and "partition" when processing a partition. If no objects are specified, the whole model is refreshed.	Process the entire model

CommitMode is equal to partialBatch. It's used when doing an initial load of large datasets that could take hours. If the refresh operation fails after successfully committing one or more batches, the successfully committed batches will remain committed (it will not roll back successfully committed batches).

> **NOTE**
> At time of writing, the batch size is the MaxParallelism value, but this value could change.

Status values

STATUS VALUE	DESCRIPTION
`notStarted`	Operation not yet started.
`inProgress`	Operation in progress.
`timedOut`	Operation timed out based on user specified timeout.
`cancelled`	Operation canceled by user or system.
`failed`	Operation failed.
`succeeded`	Operation succeeded.

GET /refreshes/<refreshId>

To check the status of a refresh operation, use the GET verb on the refresh ID. Here's an example of the response body. If the operation is in progress, `inProgress` is returned in status.

```
{
    "startTime": "2017-12-07T02:06:57.1838734Z",
    "endTime": "2017-12-07T02:07:00.4929675Z",
    "type": "full",
    "status": "succeeded",
    "currentRefreshType": "full",
    "objects": [
        {
            "table": "DimCustomer",
            "partition": "DimCustomer",
            "status": "succeeded"
        },
        {
            "table": "DimDate",
            "partition": "DimDate",
            "status": "succeeded"
        }
    ]
}
```

GET /refreshes

To get a list of historical refresh operations for a model, use the GET verb on the /refreshes collection. Here's an example of the response body.

> **NOTE**
> At time of writing, the last 30 days of refresh operations are stored and returned, but this number could change.

```
[
    {
        "refreshId": "1344a272-7893-4afa-a4b3-3fb87222fdac",
        "startTime": "2017-12-07T02:06:57.1838734Z",
        "endTime": "2017-12-07T02:07:00.4929675Z",
        "status": "succeeded"
    },
    {
        "refreshId": "474fc5a0-3d69-4c5d-adb4-8a846fa5580b",
        "startTime": "2017-12-07T01:05:54.157324Z",
        "endTime": "2017-12-07T01:05:57.353371Z",
        "status": "succeeded"
    }
]
```

DELETE /refreshes/<refreshId>

To cancel an in-progress refresh operation, use the DELETE verb on the refresh ID.

POST /sync

Having performed refresh operations, it may be necessary to synchronize the new data with replicas for query scale-out. To perform a synchronize operation for a model, use the POST verb on the /sync function. The Location header in the response includes the sync operation ID.

GET /sync status

To check the status of a sync operation, use the GET verb passing the operation ID as a parameter. Here's an example of the response body:

```
{
    "operationId": "cd5e16c6-6d4e-4347-86a0-762bdf5b4875",
    "database": "AdventureWorks2",
    "UpdatedAt": "2017-12-09T02:44:26.18",
    "StartedAt": "2017-12-09T02:44:20.743",
    "syncstate": 2,
    "details": null
}
```

Values for `syncstate`:

- 0: Replicating. Database files are being replicated to a target folder.
- 1: Rehydrating. The database is being rehydrated on read-only server instance(s).
- 2: Completed. The sync operation completed successfully.
- 3: Failed. The sync operation failed.
- 4: Finalizing. The sync operation has completed but is performing cleanup steps.

Code sample

Here's a C# code sample to get you started, RestApiSample on GitHub.

To use the code sample

1. Clone or download the repo. Open the RestApiSample solution.
2. Find the line **client.BaseAddress** = ... and provide your base URL.

The code sample uses service principal authentication.

Service principal

See Create service principal - Azure portal and Add a service principal to the server administrator role for more info on how to set up a service principal and assign the necessary permissions in Azure AS. Once you've completed the steps, complete the following additional steps:

1. In the code sample, find **string authority** = ..., replace **common** with your organization's tenant ID.
2. Comment/uncomment so the ClientCredential class is used to instantiate the cred object. Ensure the <App ID> and <App Key> values are accessed in a secure way or use certificate-based authentication for service principals.
3. Run the sample.

See also

Samples
REST API

Refresh with Logic Apps

11/2/2020 • 2 minutes to read • Edit Online

By using Logic Apps and REST calls, you can perform automated data refresh operations on your Azure Analysis tabular models, including synchronization of read-only replicas for query scale-out.

To learn more about using REST APIs with Azure Analysis Services, see Asynchronous refresh with the REST API.

Authentication

All calls must be authenticated with a valid Azure Active Directory (OAuth 2) token. The examples in this article will use a Service Principal (SPN) to authenticate to Azure Analysis Services. To learn more, see Create a service principal by using Azure portal.

Design the logic app

> **IMPORTANT**
>
> The following examples assume that the Azure Analysis Services firewall is disabled. If the firewall is enabled, the public IP address of the request initiator must be added to the approved list in the Azure Analysis Services firewall. To learn more about Azure Logic Apps IP ranges per region, see Limits and configuration information for Azure Logic Apps.

Prerequisites

Create a Service Principal (SPN)

To learn about creating a Service Principal, see Create a service principal by using Azure portal.

Configure permissions in Azure Analysis Services

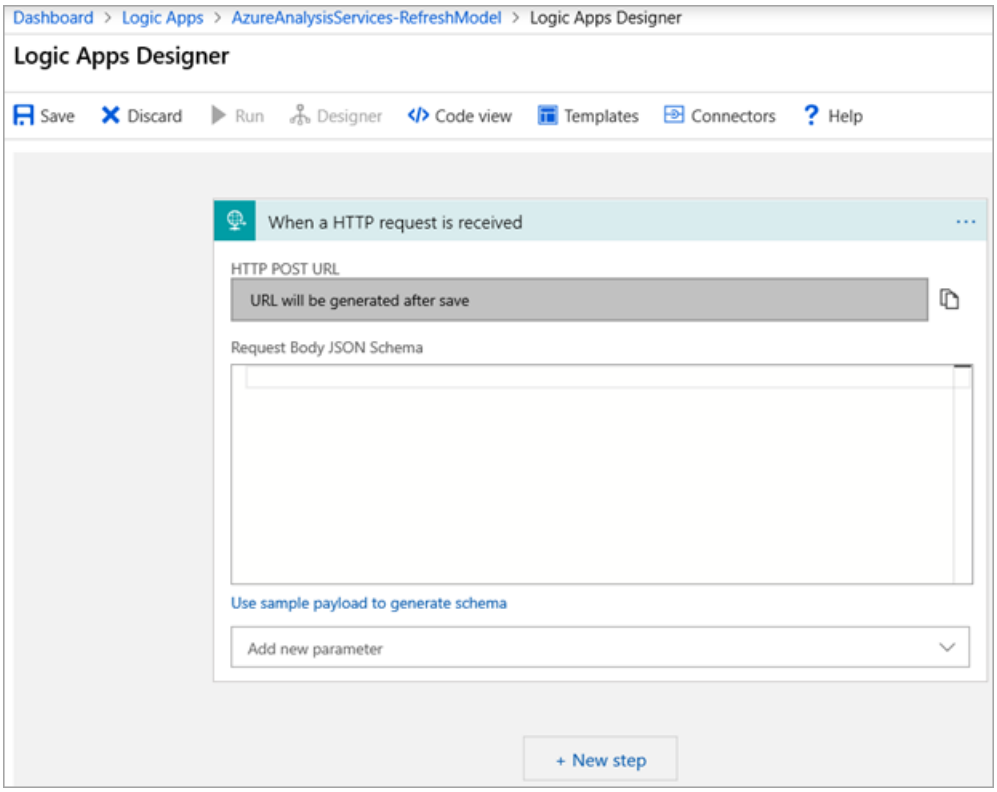
The Service Principal you create must have server administrator permissions on the server. To learn more, see Add a service principal to the server administrator role.

Configure the Logic App

In this example, the Logic App is designed to trigger when a HTTP request is received. This will enable the use of an orchestration tool, such as Azure Data Factory, to trigger the Azure Analysis Services model refresh.

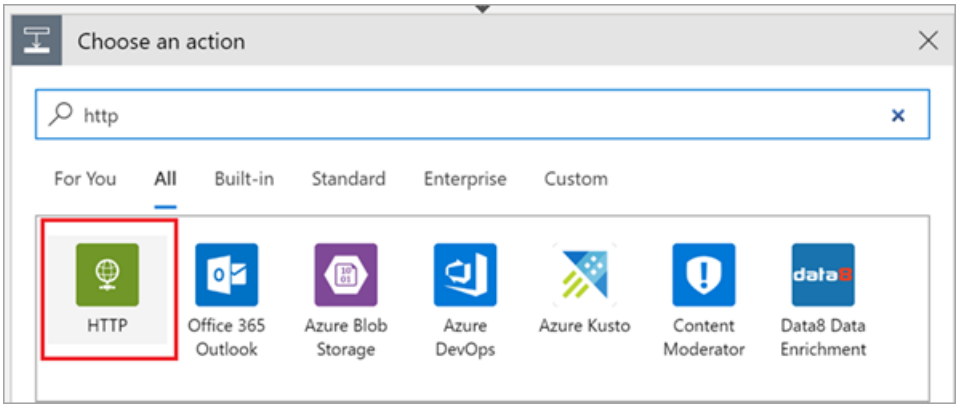
Once you have created a Logic App:

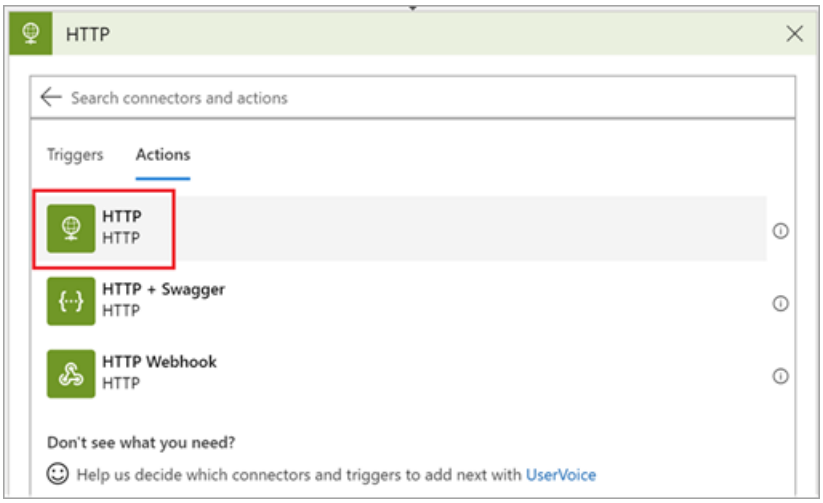
1. In the Logic App designer, choose the first action as **When a HTTP request is received**.



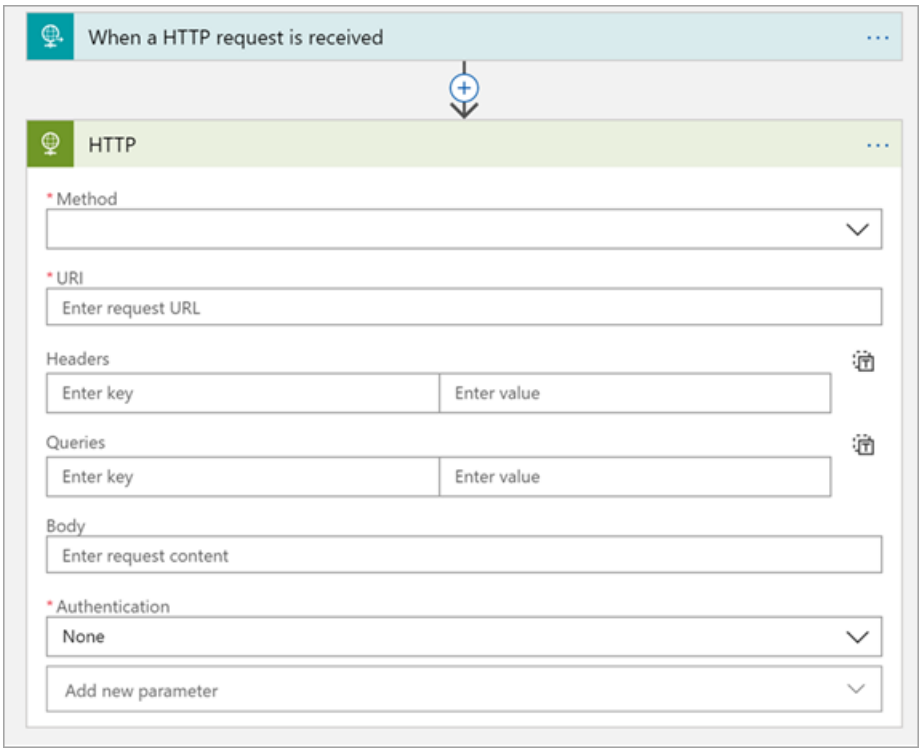
This step will populate with the HTTP POST URL once the Logic App is saved.

2. Add a new step and search for **HTTP**.





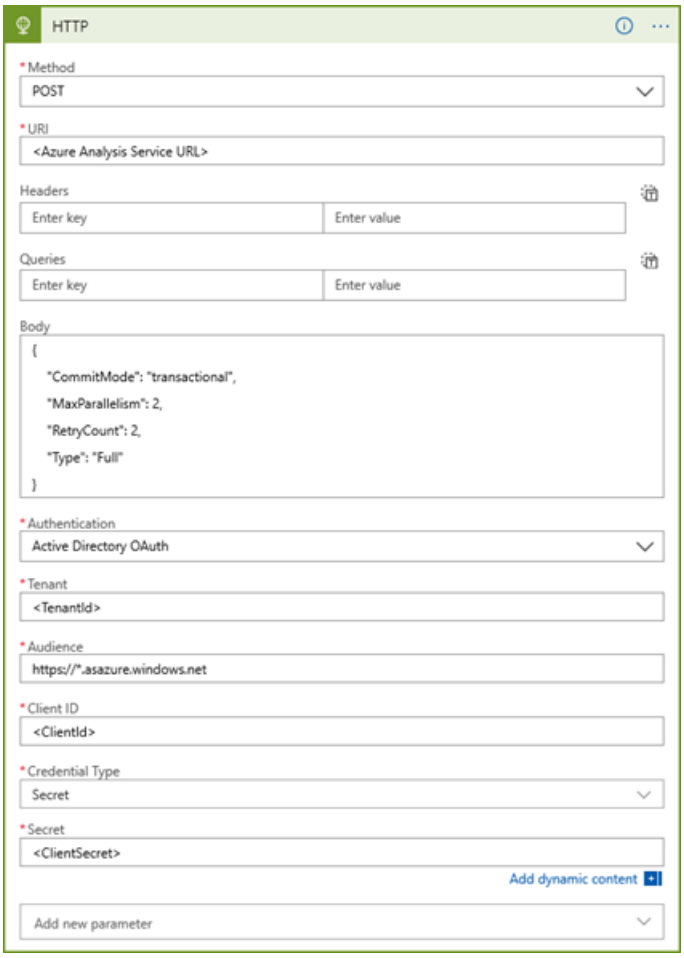
3. Select **HTTP** to add this action.



Configure the HTTP activity as follows:

PROPERTY	VALUE
Method	POST
URI	https://*your server region*/servers/*aas server name*/models/*your database name*/refreshes For example: https://westus.asazure.windows.net/servers/myserver/models/AdventureWorks/refreshes
Headers	Content-Type, application/json Headers Content-Type | application/json Enter key | Enter value
Body	To learn more about forming the request body, see Asynchronous refresh with the REST API - POST /refreshes.
Authentication	Active Directory OAuth
Tenant	Fill in your Azure Active Directory TenantId
Audience	https://*.asazure.windows.net
Client ID	Enter your Service Principal Name ClientID
Credential Type	Secret
Secret	Enter your Service Principal Name Secret

Example:

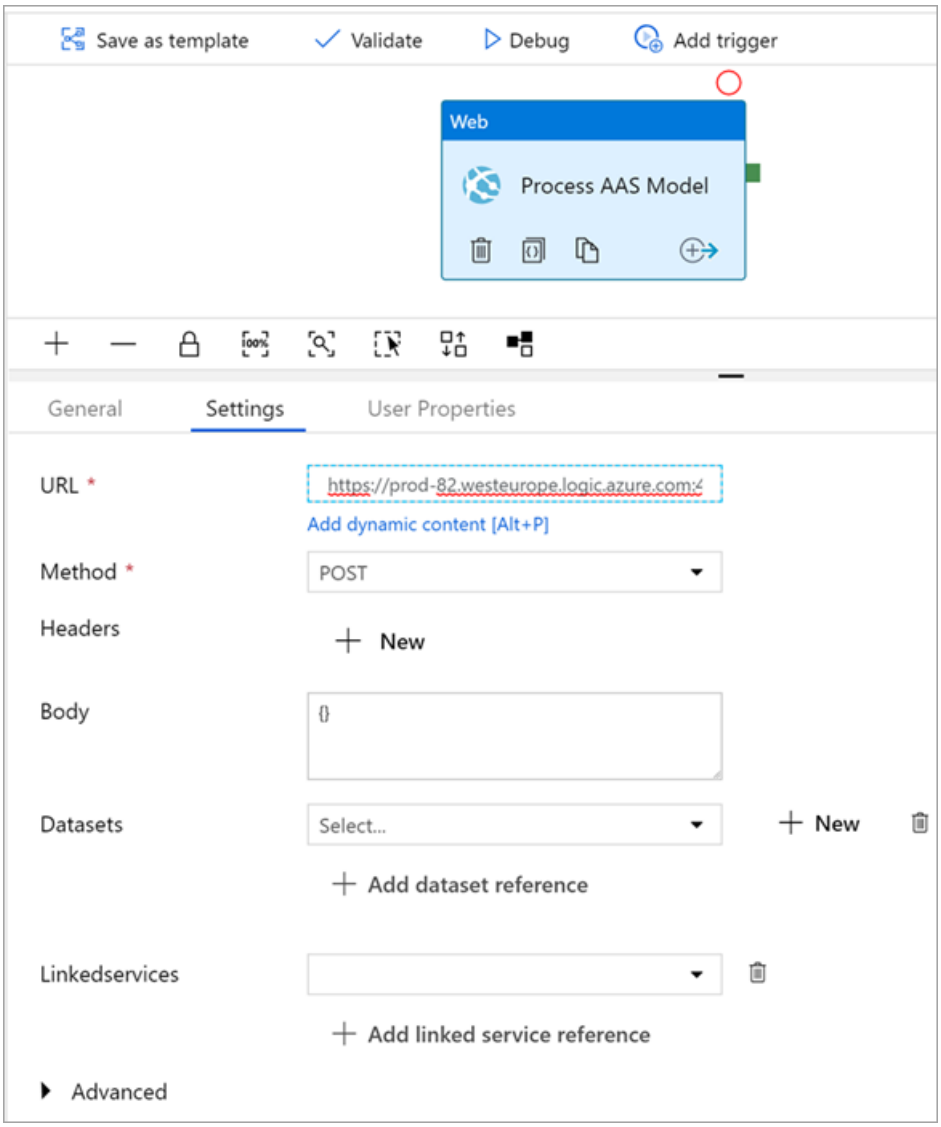


Now test the Logic App. In the Logic App designer, click **Run**.

Consume the Logic App with Azure Data Factory

Once the Logic App is saved, review the **When a HTTP request is received** activity and then copy the **HTTP POST URL** that is now generated. This is the URL that can be used by Azure Data Factory to make the asynchronous call to trigger the Logic App.

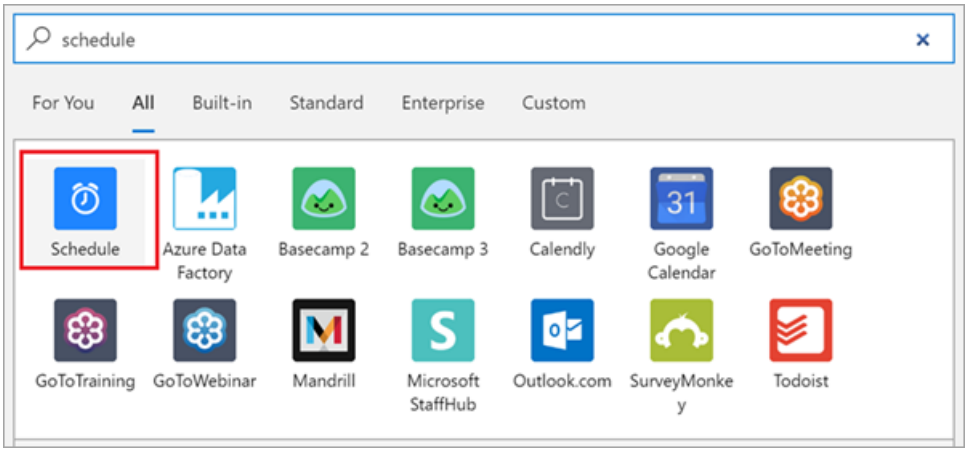
Here's an example Azure Data Factory Web Activity that does this action.

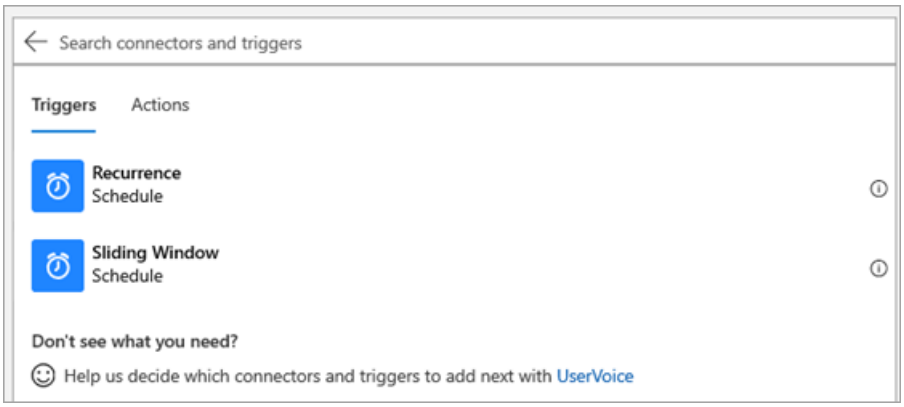


Use a self-contained Logic App

If you don't plan on using an Orchestration tool such as Data Factory to trigger the model refresh, you can set the logic app to trigger the refresh based on a schedule.

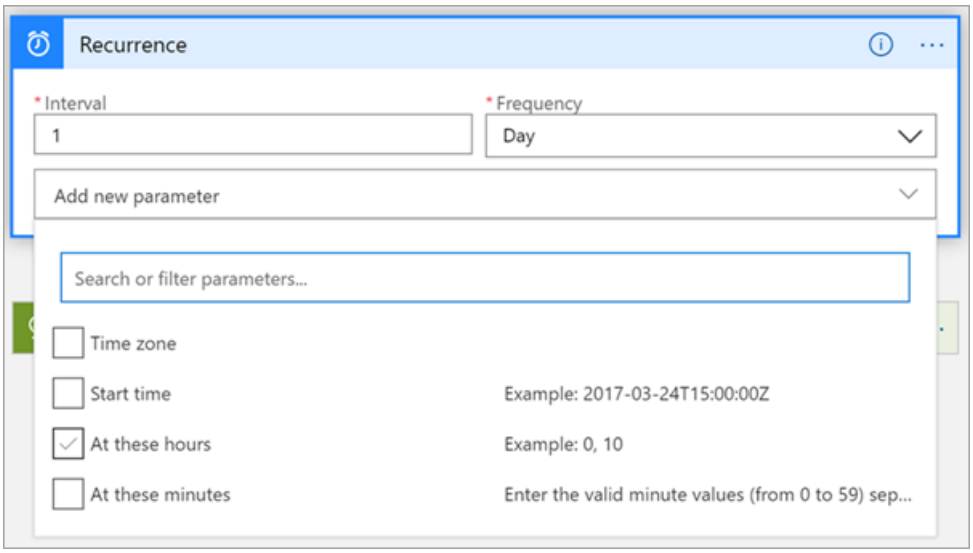
Using the example above, delete the first activity and replace it with a **Schedule** activity.



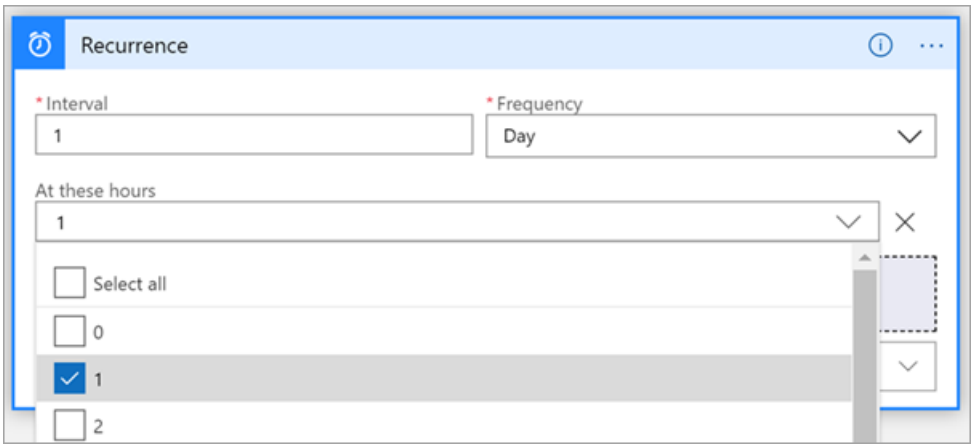


This example will use **Recurrence**.

Once the activity has been added, configure the Interval and Frequency, then add a new parameter and choose **At these hours**.



Select the wanted hours.



Save the Logic App.

Next steps

Samples
REST API

Refresh with Azure Automation

3/5/2021 • 4 minutes to read • Edit Online

By using Azure Automation and PowerShell Runbooks, you can perform automated data refresh operations on your Azure Analysis tabular models.

The example in this article uses the SqlServer PowerShell module. A sample PowerShell Runbook, which demonstrates refreshing a model is provided later in this article.

Authentication

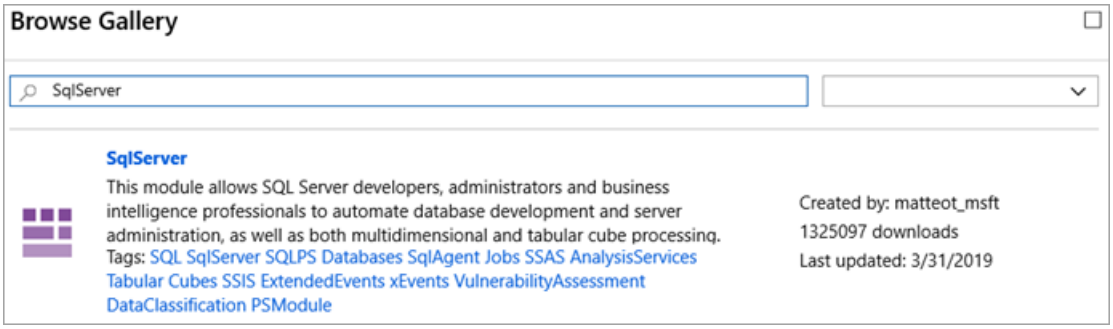
All calls must be authenticated with a valid Azure Active Directory (OAuth 2) token. The example in this article uses a Service Principal (SPN) to authenticate to Azure Analysis Services. To learn more, see Create a service principal by using Azure portal.

Prerequisites

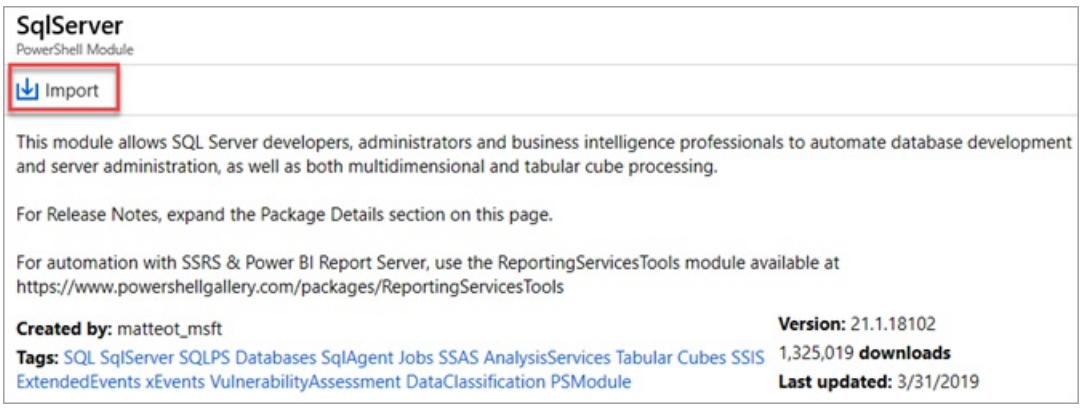
> **IMPORTANT**
> The following example assumes the Azure Analysis Services firewall is disabled. If a firewall is enabled, the public IP address of the request initiator must be included in a firewall rule.

Install SqlServer modules from PowerShell gallery.

1. In your Azure Automation Account, Click **Modules**, then **Browse gallery**.
2. In the search bar, search for **SqlServer**.



3. Select SqlServer, then click **Import**.



4. Click **OK**.

Create a Service Principal (SPN)

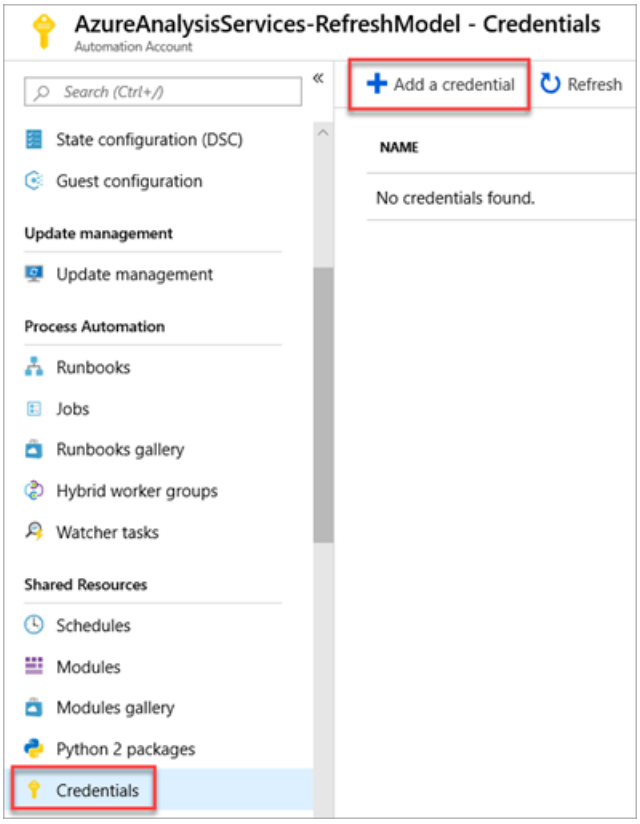
To learn about creating a Service Principal, see Create a service principal by using Azure portal.

Configure permissions in Azure Analysis Services

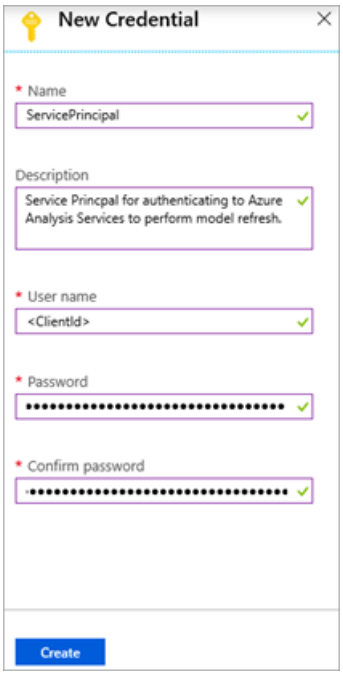
The Service Principal you create must have server administrator permissions on the server. To learn more, see Add a service principal to the server administrator role.

Design the Azure Automation Runbook

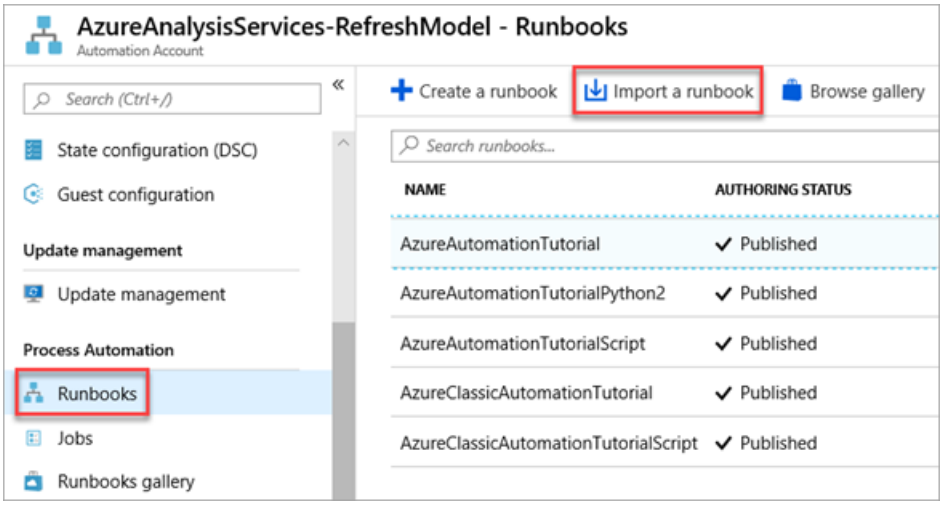
1. In the Automation Account, create a **Credentials** resource which will be used to securely store the Service Principal.



2. Enter the details for the credential. In **User name**, enter the service principal Application ID (appid), and then in **Password**, enter the service principal Secret.

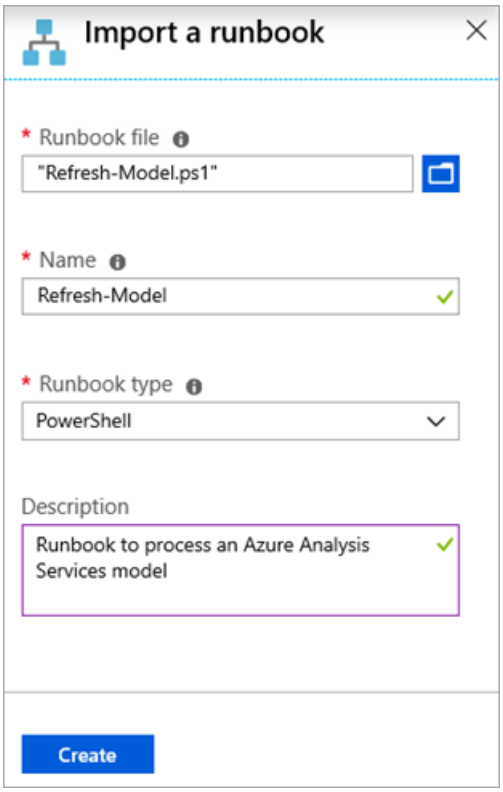


3. Import the Automation Runbook.

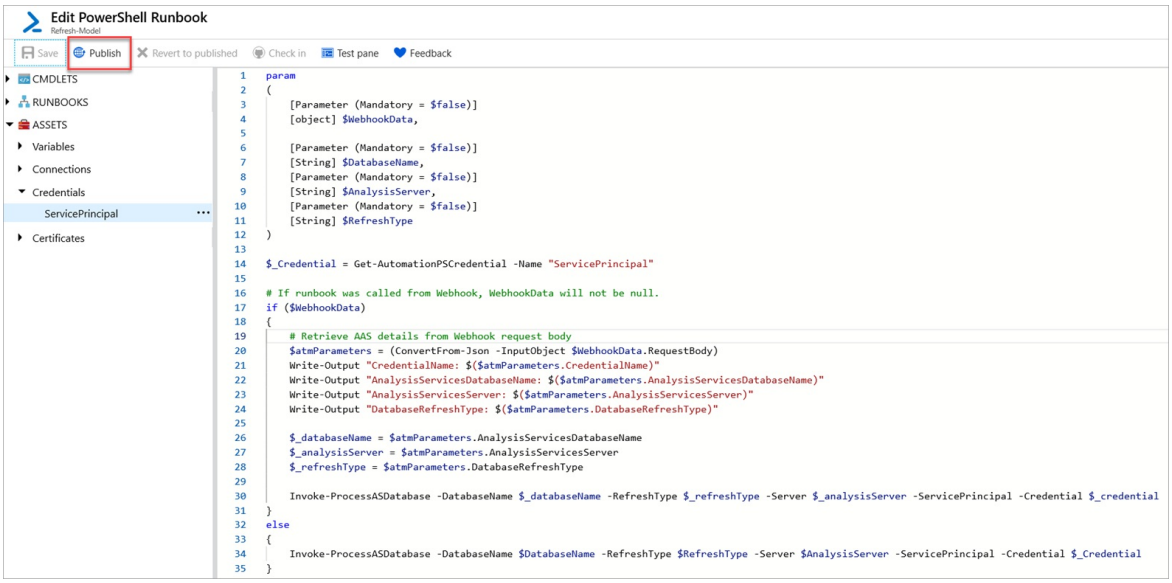


4. Browse for the Refresh-Model.ps1 file, provide a **Name** and **Description**, and then click **Create**.

> **NOTE**
> Use script from Sample Powershell Runbook section at the bottom of this document to create a file called Refresh-Model.ps1 and save to local machine to import into Runbook.



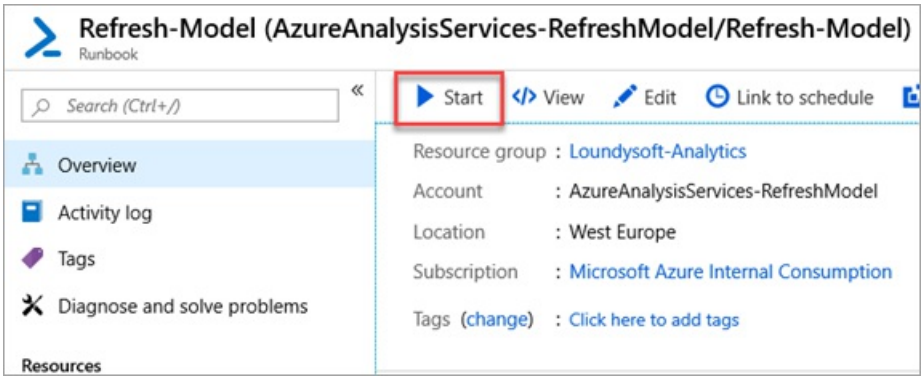
5. When the Runbook has been created, it will automatically go into edit mode. Select **Publish**.



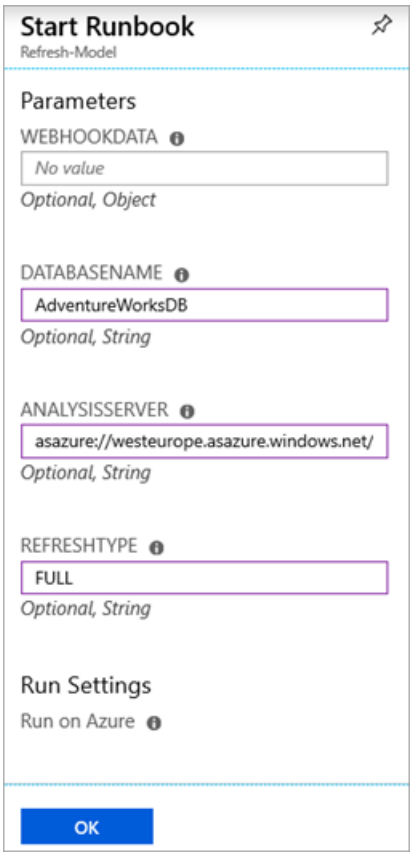
NOTE

The credential resource that was created previously is retrieved by the runbook by using the **Get-AutomationPSCredential** command. This command is then passed to the **Invoke-ProcessASADatabase** PowerShell command to perform the authentication to Azure Analysis Services.

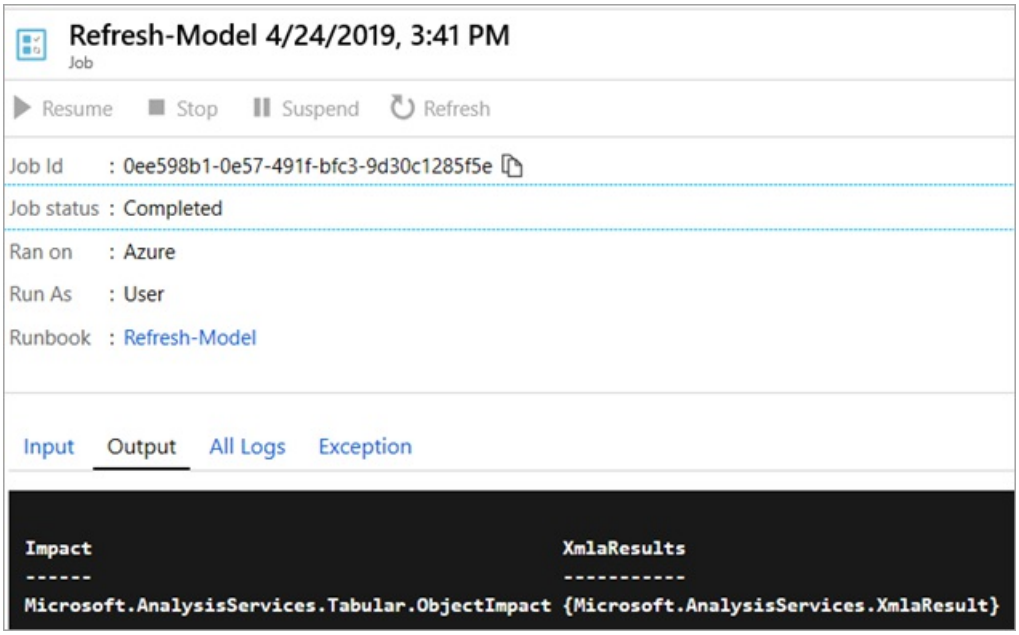
6. Test the runbook by clicking **Start**.



7. Fill out the **DATABASENAME**, **ANALYSISSERVER**, and **REFRESHTYPE** parameters, and then click **OK**. The **WEBHOOKDATA** parameter is not required when the Runbook is run manually.



If the Runbook executed successfully, you will receive an output like the following:

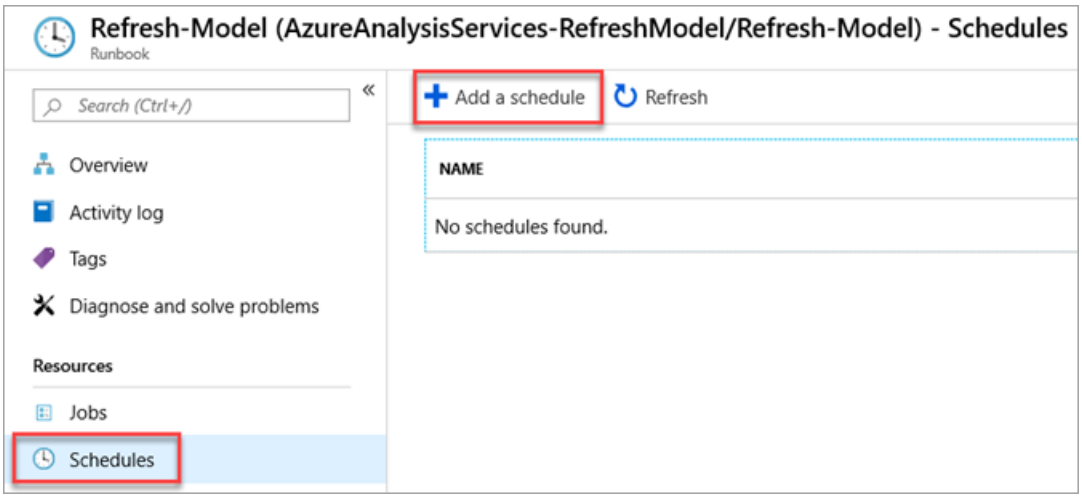


Use a self-contained Azure Automation Runbook

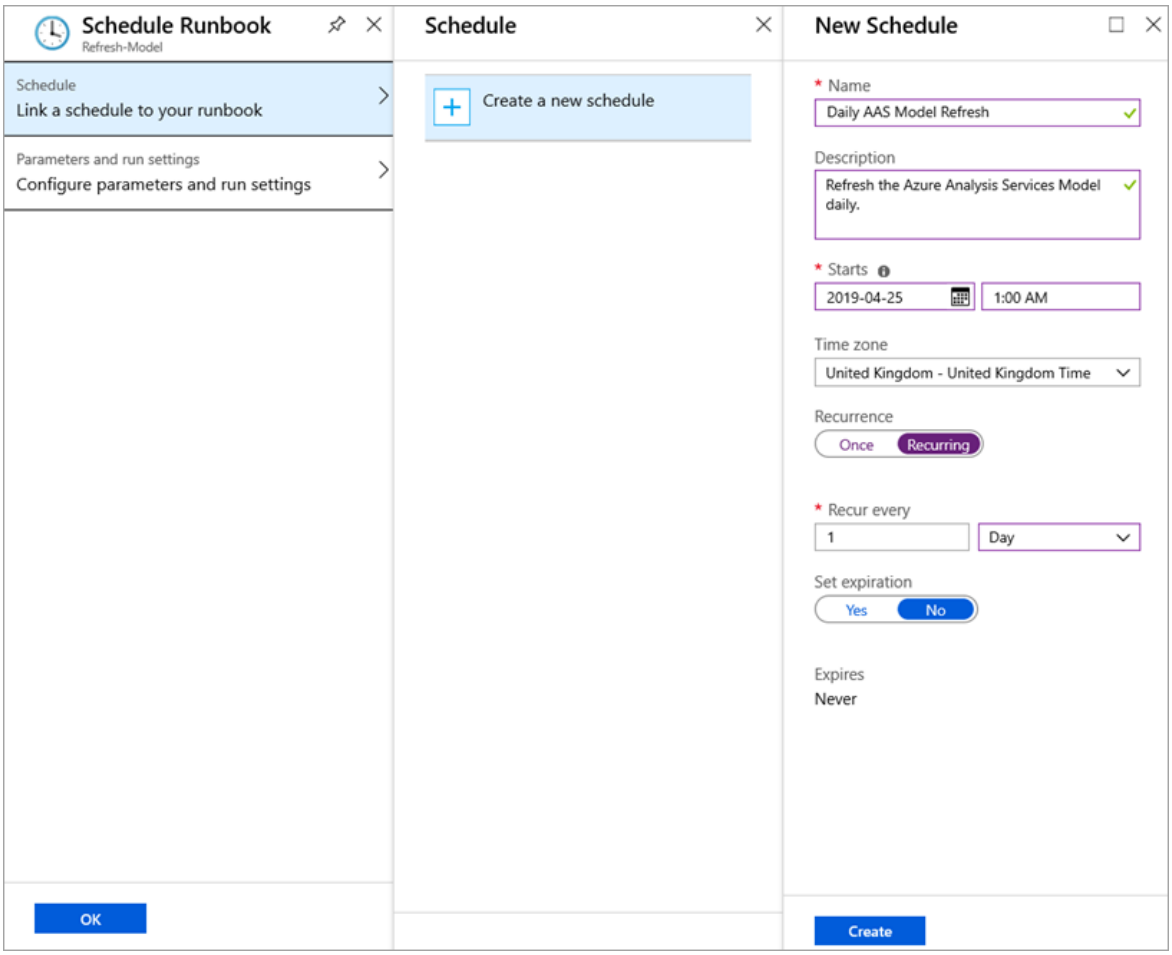
The Runbook can be configured to trigger the Azure Analysis Services model refresh on a scheduled basis.

This can be configured as follows:

1. In the Automation Runbook, click **Schedules**, then **Add a Schedule**.

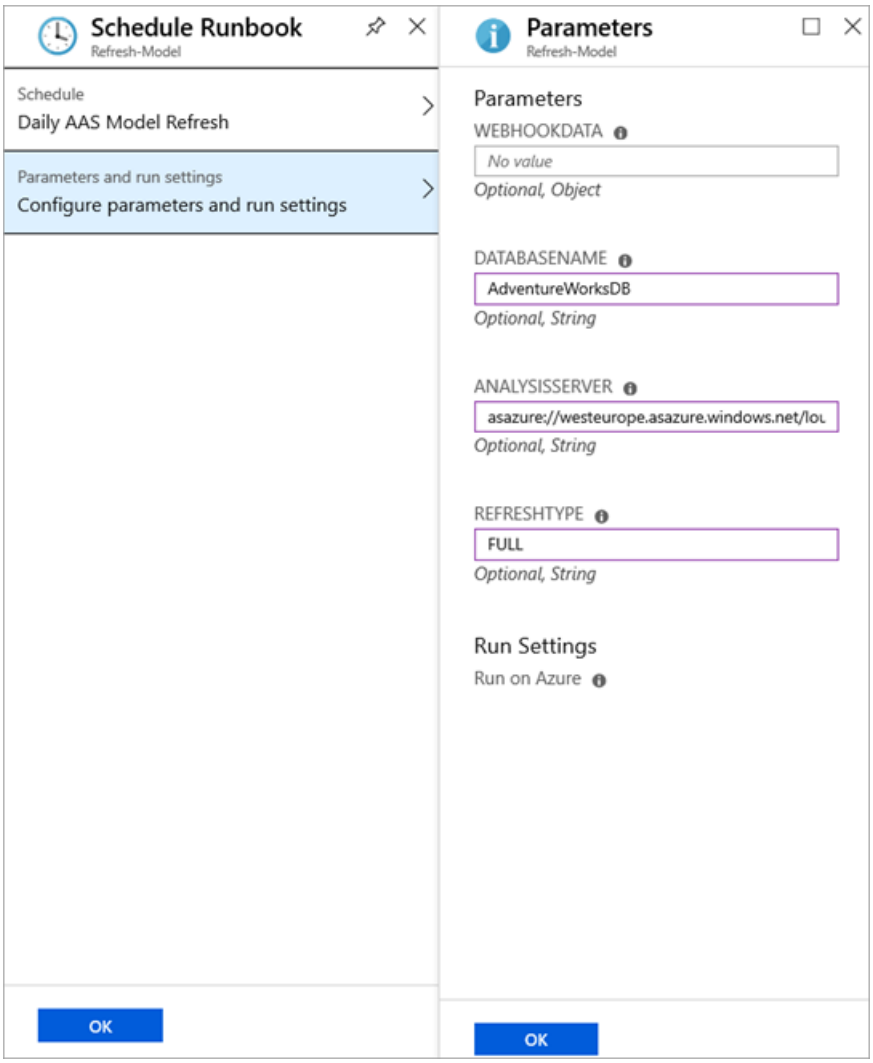


2. Click **Schedule** > **Create a new schedule**, and then fill in the details.



3. Click **Create**.

4. Fill in the parameters for the schedule. These will be used each time the Runbook triggers. The **WEBHOOKDATA** parameter should be left blank when running via a schedule.



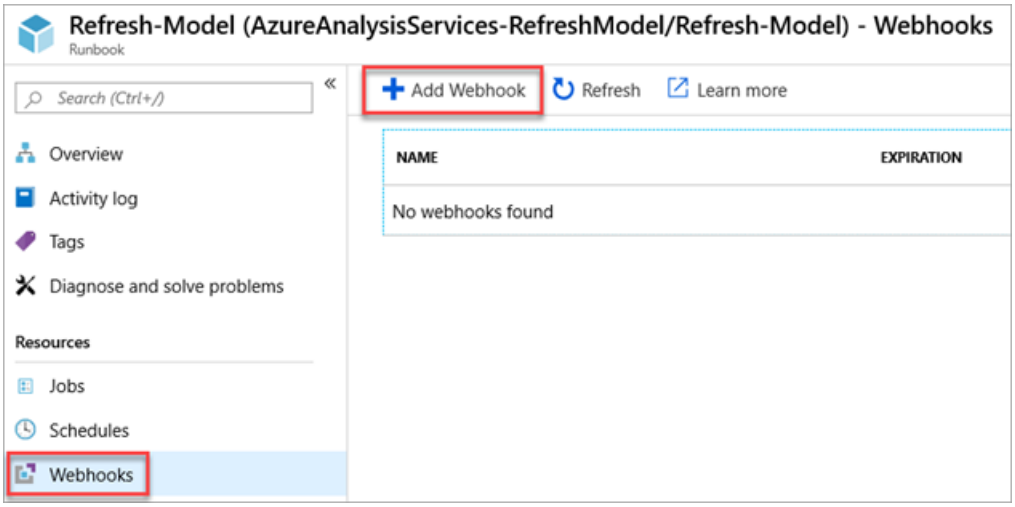
5. Click **OK**.

Consume with Data Factory

To consume the runbook by using Azure Data Factory, first create a **Webhook** for the runbook. The **Webhook** will provide a URL which can be called via an Azure Data Factory web activity.

> **IMPORTANT**
> To create a **Webhook**, the status of the Runbook must be **Published**.

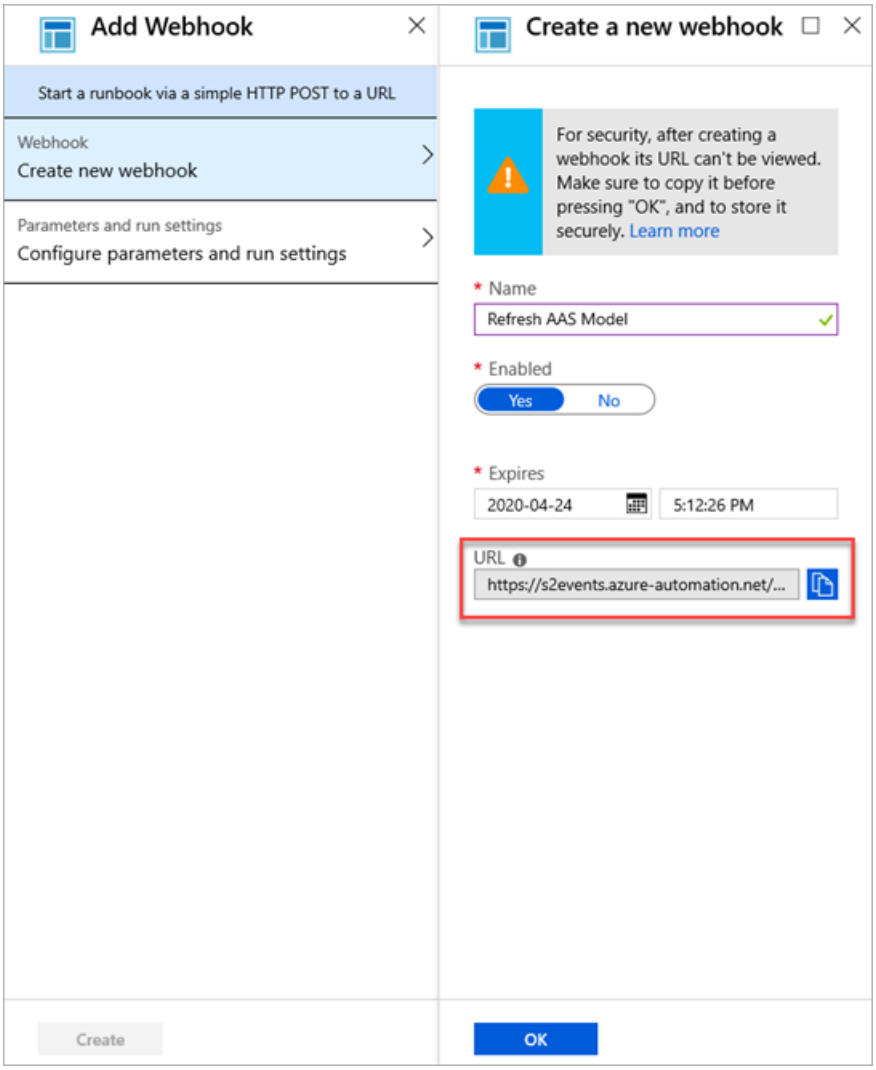
1. In your Automation Runbook, click **Webhooks**, and then click **Add Webhook**.



2. Give the Webhook a name and an expiry. The name only identifies the Webhook inside the Automation Runbook, it doesn't form part of the URL.

   Caution

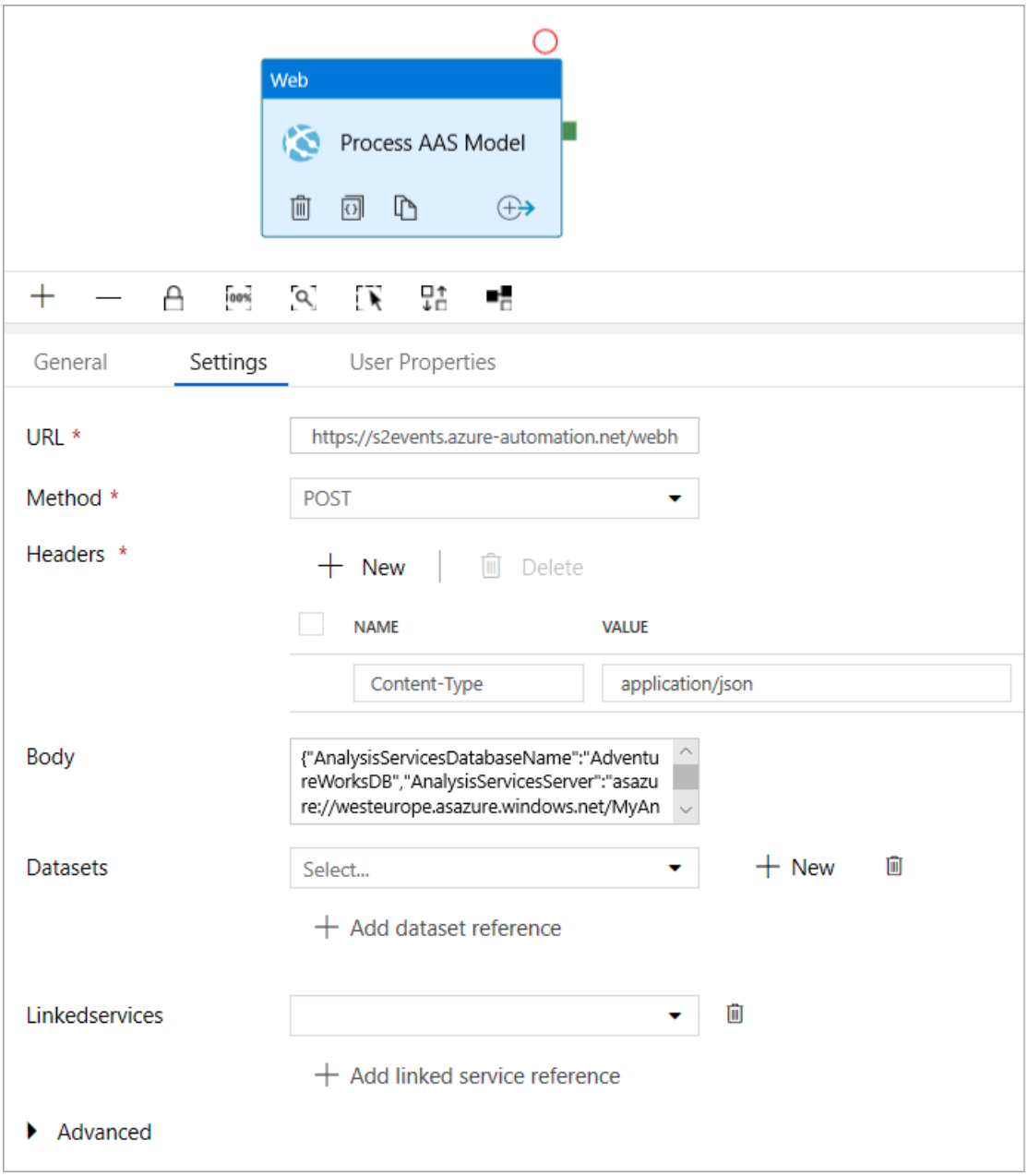
   Ensure you copy the URL before closing the wizard as you cannot get it back once closed.



   The parameters for the webhook can remain blank. When configuring the Azure Data Factory web activity, the parameters can be passed into the body of the web call.

3. In Data Factory, configure a **web activity**

Example



The **URL** is the URL created from the Webhook.

The **body** is a JSON document which should contain the following properties:

PROPERTY	VALUE
AnalysisServicesDatabase	The name of the Azure Analysis Services database Example: AdventureWorksDB
AnalysisServicesServer	The Azure Analysis Services server name. Example: https://westus.asazure.windows.net/servers/myserver/models/AdventureWorks/
DatabaseRefreshType	The type of refresh to perform. Example: Full

Example JSON body:

```
{
    "AnalysisServicesDatabaseName": "AdventureWorksDB",
    "AnalysisServicesServer": "asazure://westeurope.asazure.windows.net/MyAnalysisServer",
    "DatabaseRefreshType": "Full"
}
```

These parameters are defined in the runbook PowerShell script. When the web activity is executed, the JSON payload passed is WEBHOOKDATA.

This is deserialized and stored as PowerShell parameters, which are then used by the Invoke-ProcesASDatabase PowerShell command.

```
param
(
    [Parameter (Mandatory = $false)]
    [object] $WebhookData,

    [Parameter (Mandatory = $false)]
    [String] $DatabaseName,
    [Parameter (Mandatory = $false)]
    [String] $AnalysisServer,
    [Parameter (Mandatory = $false)]
    [String] $RefreshType
)

$_Credential = Get-AutomationPSCredential -Name "ServicePrincipal"

# If runbook was called from Webhook, WebhookData will not be null.
if ($WebhookData)
{
    # Retrieve AAS details from Webhook request body
    $atmParameters = (ConvertFrom-Json -InputObject $WebhookData.RequestBody)
    Write-Output "CredentialName: $($atmParameters.CredentialName)"
    Write-Output "AnalysisServicesDatabaseName: $($atmParameters.AnalysisServicesDatabaseName)"
    Write-Output "AnalysisServicesServer: $($atmParameters.AnalysisServicesServer)"
    Write-Output "DatabaseRefreshType: $($atmParameters.DatabaseRefreshType)"

    $_databaseName = $atmParameters.AnalysisServicesDatabaseName
    $_analysisServer = $atmParameters.AnalysisServicesServer
    $_refreshType = $atmParameters.DatabaseRefreshType

    Invoke-ProcessASDatabase -DatabaseName $_databaseName -RefreshType $_refreshType -Server $_analysisServer -ServicePrincipal -Credential $_credential
}
else
{
    Invoke-ProcessASDatabase -DatabaseName $DatabaseName -RefreshType $RefreshType -Server $AnalysisServer -ServicePrincipal -Credential $_Credential
}
```

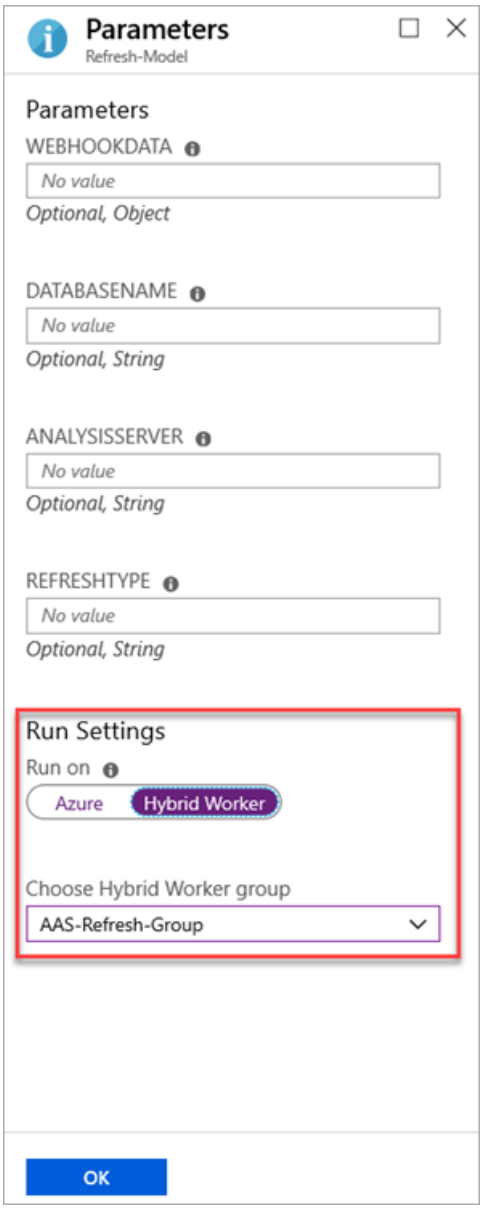
Use a Hybrid Worker with Azure Analysis Services

An Azure Virtual Machine with a static public IP address can be used as an Azure Automation Hybrid Worker. This public IP address can then be added to the Azure Analysis Services firewall.

> **IMPORTANT**
>
> Ensure the Virtual Machine public IP address is configured as static.
>
> To learn more about configuring Azure Automation Hybrid Workers, see Hybrid Runbook Worker installation.

Once a Hybrid Worker is configured, create a Webhook as described in the section Consume with Data Factory. The only difference here is to select the **Run on** > **Hybrid Worker** option when configuring the Webhook.

Example webhook using Hybrid Worker:



Sample PowerShell Runbook

The following code snippet is an example of how to perform the Azure Analysis Services model refresh using a PowerShell Runbook.

```
param
(
    [Parameter (Mandatory = $false)]
    [object] $WebhookData,

    [Parameter (Mandatory = $false)]
    [String] $DatabaseName,
    [Parameter (Mandatory = $false)]
    [String] $AnalysisServer,
    [Parameter (Mandatory = $false)]
    [String] $RefreshType
)

$_Credential = Get-AutomationPSCredential -Name "ServicePrincipal"

# If runbook was called from Webhook, WebhookData will not be null.
if ($WebhookData)
{
    # Retrieve AAS details from Webhook request body
    $atmParameters = (ConvertFrom-Json -InputObject $WebhookData.RequestBody)
    Write-Output "CredentialName: $($atmParameters.CredentialName)"
    Write-Output "AnalysisServicesDatabaseName: $($atmParameters.AnalysisServicesDatabaseName)"
    Write-Output "AnalysisServicesServer: $($atmParameters.AnalysisServicesServer)"
    Write-Output "DatabaseRefreshType: $($atmParameters.DatabaseRefreshType)"

    $_databaseName = $atmParameters.AnalysisServicesDatabaseName
    $_analysisServer = $atmParameters.AnalysisServicesServer
    $_refreshType = $atmParameters.DatabaseRefreshType

    Invoke-ProcessASDatabase -DatabaseName $_databaseName -RefreshType $_refreshType -Server
$_analysisServer -ServicePrincipal -Credential $_credential
}
else
{
    Invoke-ProcessASDatabase -DatabaseName $DatabaseName -RefreshType $RefreshType -Server $AnalysisServer -
ServicePrincipal -Credential $_Credential
}
```

Next steps

Samples
REST API

Manage database roles and users

4/27/2021 • 5 minutes to read • Edit Online

At the model database level, all users must belong to a role. Roles define users with particular permissions for the model database. Any user or security group added to a role must have an account in an Azure AD tenant in the same subscription as the server.

How you define roles is different depending on the tool you use, but the effect is the same.

Role permissions include:

- **Administrator** - Users have full permissions for the database. Database roles with Administrator permissions are different from server administrators.
- **Process** - Users can connect to and perform process operations on the database, and analyze model database data.
- **Read** - Users can use a client application to connect to and analyze model database data.

When creating a tabular model project, you create roles and add users or groups to those roles by using Role Manager in Visual Studio with Analysis Services projects. When deployed to a server, use SQL Server Management Studio (SSMS), Analysis Services PowerShell cmdlets, or Tabular Model Scripting Language (TMSL) to add or remove roles and user members.

When adding a **security group**, use `obj:groupid@tenantid`.

When adding a **service principal** use `app:appid@tenantid`.

To add or manage roles and users in Visual Studio

1. In **Tabular Model Explorer**, right-click **Roles**.
2. In **Role Manager**, click **New**.
3. Type a name for the role.

   By default, the name of the default role is incrementally numbered for each new role. It's recommended you type a name that clearly identifies the member type, for example, Finance Managers or Human Resources Specialists.
4. Select one of the following permissions:

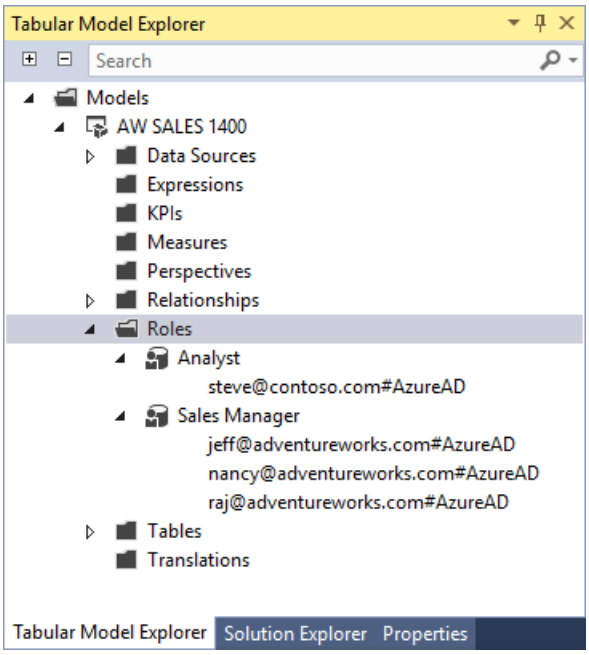
PERMISSION	DESCRIPTION
None	Members cannot read or modify the model schema and cannot query data.
Read	Members can query data (based on row filters) but cannot modify the model schema.
Read and Process	Members can query data (based on row-level filters) and run Process and Process All operations, but cannot modify the model schema.

PERMISSION	DESCRIPTION
Process	Members can run Process and Process All operations. Cannot read or modify the model schema and cannot query data.
Administrator	Members can modify the model schema and query all data.

5. If the role you are creating has Read or Read and Process permission, you can add row filters by using a DAX formula. Click the **Row Filters** tab, then select a table, then click the **DAX Filter** field, and then type a DAX formula.

6. Click **Members** > **Add External**.

7. In **Add External Member**, enter users or groups in your tenant Azure AD by email address. After you click OK and close Role Manager, roles and role members appear in Tabular Model Explorer.



8. Deploy to your Azure Analysis Services server.

To add or manage roles and users in SSMS

To add roles and users to a deployed model database, you must be connected to the server as a Server administrator or already in a database role with administrator permissions.

1. In Object Exporer, right-click **Roles** > **New Role**.

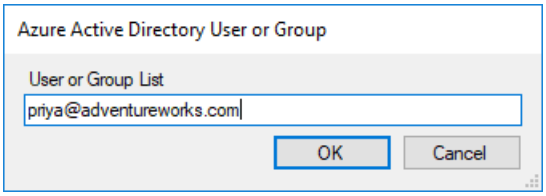
2. In **Create Role**, enter a role name and description.

3. Select a permission.

PERMISSION	DESCRIPTION
Full control (Administrator)	Members can modify the model schema, process, and can query all data.
Process database	Members can run Process and Process All operations. Cannot modify the model schema and cannot query data.

PERMISSION	DESCRIPTION
Read	Members can query data (based on row filters) but cannot modify the model schema.

4. Click **Membership**, then enter a user or group in your tenant Azure AD by email address.



5. If the role you are creating has Read permission, you can add row filters by using a DAX formula. Click **Row Filters**, select a table, and then type a DAX formula in the **DAX Filter** field.

To add roles and users by using a TMSL script

You can run a TMSL script in the XMLA window in SSMS or by using PowerShell. Use the CreateOrReplace command and the Roles object.

Sample TMSL script

In this sample, a B2B external user and a group are added to the Analyst role with Read permissions for the SalesBI database. Both the external user and group must be in same tenant Azure AD.

```
{
  "createOrReplace": {
    "object": {
      "database": "SalesBI",
      "role": "Analyst"
    },
    "role": {
      "name": "Users",
      "description": "All allowed users to query the model",
      "modelPermission": "read",
      "members": [
        {
          "memberName": "user1@contoso.com",
          "identityProvider": "AzureAD"
        },
        {
          "memberName": "group1@adventureworks.com",
          "identityProvider": "AzureAD"
        }
      ]
    }
  }
}
```

To add roles and users by using PowerShell

The SqlServer module provides task-specific database management cmdlets and the general-purpose Invoke-ASCmd cmdlet that accepts a Tabular Model Scripting Language (TMSL) query or script. The following cmdlets are used for managing database roles and users.

CMDLET	DESCRIPTION
Add-RoleMember	Add a member to a database role.
Remove-RoleMember	Remove a member from a database role.
Invoke-ASCmd	Execute a TMSL script.

Row filters

Row filters define which rows in a table can be queried by members of a particular role. Row filters are defined for each table in a model by using DAX formulas.

Row filters can be defined only for roles with Read and Read and Process permissions. By default, if a row filter is not defined for a particular table, members can query all rows in the table unless cross-filtering applies from another table.

Row filters require a DAX formula, which must evaluate to a TRUE/FALSE value, to define the rows that can be queried by members of that particular role. Rows not included in the DAX formula cannot be queried. For example, the Customers table with the following row filters expression, *=Customers [Country] = "USA"*, members of the Sales role can only see customers in the USA.

Row filters apply to the specified rows and related rows. When a table has multiple relationships, filters apply security for the relationship that is active. Row filters are intersected with other row filers defined for related tables, for example:

TABLE	DAX EXPRESSION
Region	=Region[Country]="USA"
ProductCategory	=ProductCategory[Name]="Bicycles"
Transactions	=Transactions[Year]=2016

The net effect is members can query rows of data where the customer is in the USA, the product category is bicycles, and the year is 2016. Users cannot query transactions outside of the USA, transactions that are not bicycles, or transactions not in 2016 unless they are a member of another role that grants these permissions.

You can use the filter, *=FALSE()*, to deny access to all rows for an entire table.

Next steps

Manage server administrators
Manage Azure Analysis Services with PowerShell
Tabular Model Scripting Language (TMSL) Reference

Manage server administrators

4/21/2021 • 2 minutes to read • Edit Online

Server administrators must be a valid user, service principal, or security group in the Azure Active Directory (Azure AD) for the tenant in which the server resides. You can use **Analysis Services Admins** for your server in Azure portal, Server Properties in SSMS, PowerShell, or REST API to manage server administrators.

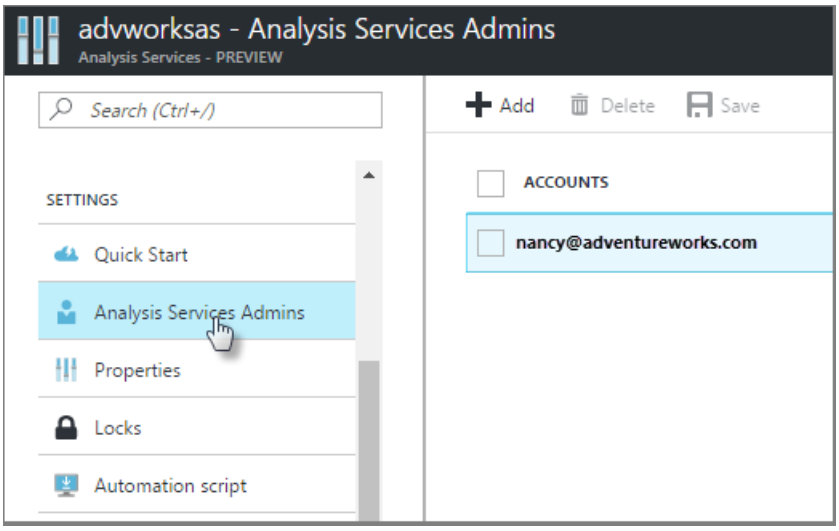
When adding a **security group**, use `obj:groupid@tenantid`. Service principals are not supported in security groups added to the server administrator role.

To learn more about adding a service principal to the server admin role, see Add a service principal to the server administrator role.

If server firewall is enabled, server administrator client computer IP addresses must be included in a firewall rule. To learn more, see Configure server firewall.

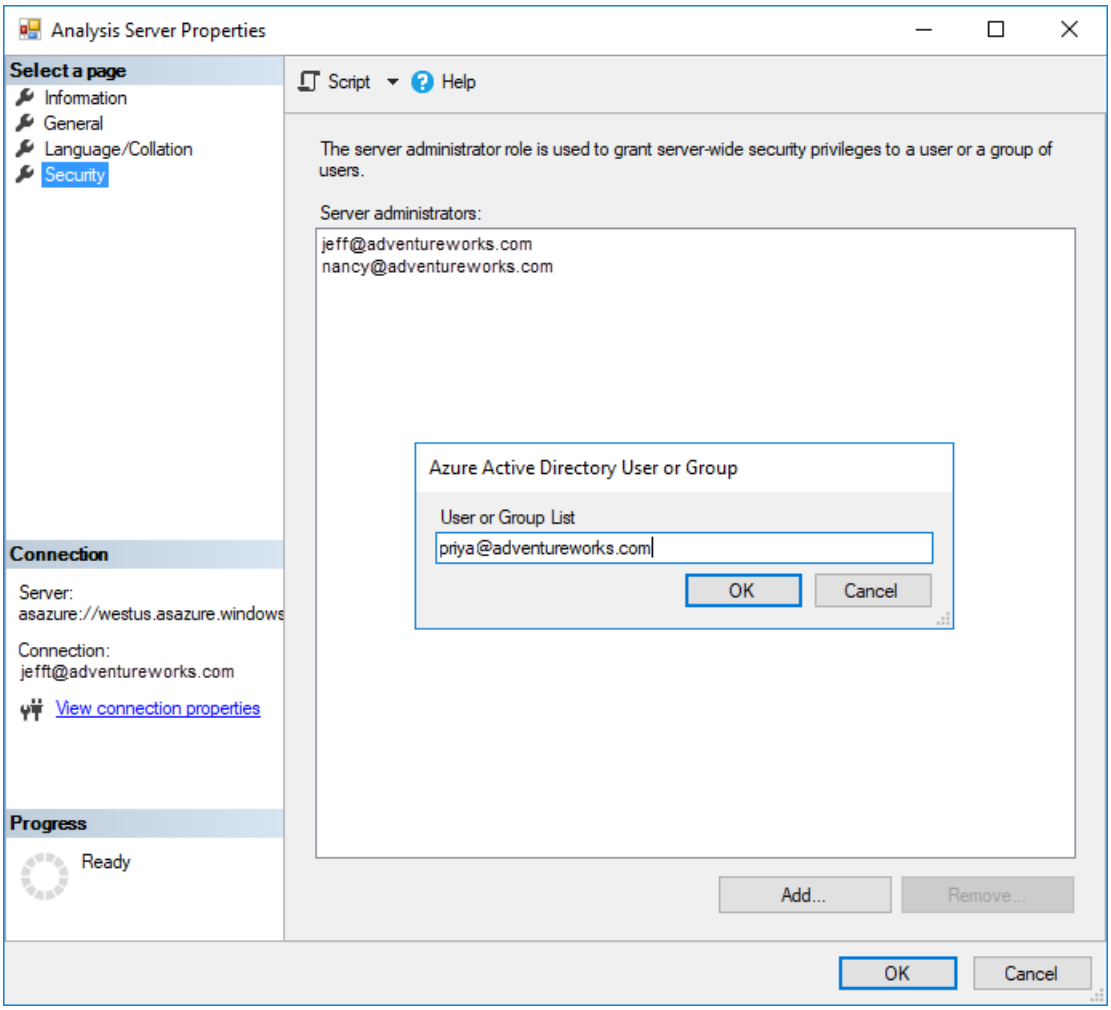
To add server administrators by using Azure portal

1. In the portal, for your server, click **Analysis Services Admins**.
2. In **<servername> - Analysis Services Admins**, click **Add**.
3. In **Add Server Administrators**, select user accounts from your Azure AD or invite external users by email address.



To add server administrators by using SSMS

1. Right-click the server > **Properties**.
2. In **Analysis Server Properties**, click **Security**.
3. Click **Add**, and then enter the email address for a user or group in your Azure AD.



PowerShell

> **NOTE**
> This article has been updated to use the Azure Az PowerShell module. The Az PowerShell module is the recommended PowerShell module for interacting with Azure. To get started with the Az PowerShell module, see Install Azure PowerShell. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Use New-AzAnalysisServicesServer cmdlet to specify the Administrator parameter when creating a new server.
Use Set-AzAnalysisServicesServer cmdlet to modify the Administrator parameter for an existing server.

REST API

Use Create to specify the asAdministrator property when creating a new server.
Use Update to specify the asAdministrator property when modifying an existing server.

Next steps

Authentication and user permissions
Manage database roles and users
Azure role-based access control (Azure RBAC)

Connect with Excel

3/5/2021 • 2 minutes to read • Edit Online

Once you've created a server, and deployed a tabular model to it, clients can connect and begin exploring data.

Before you begin

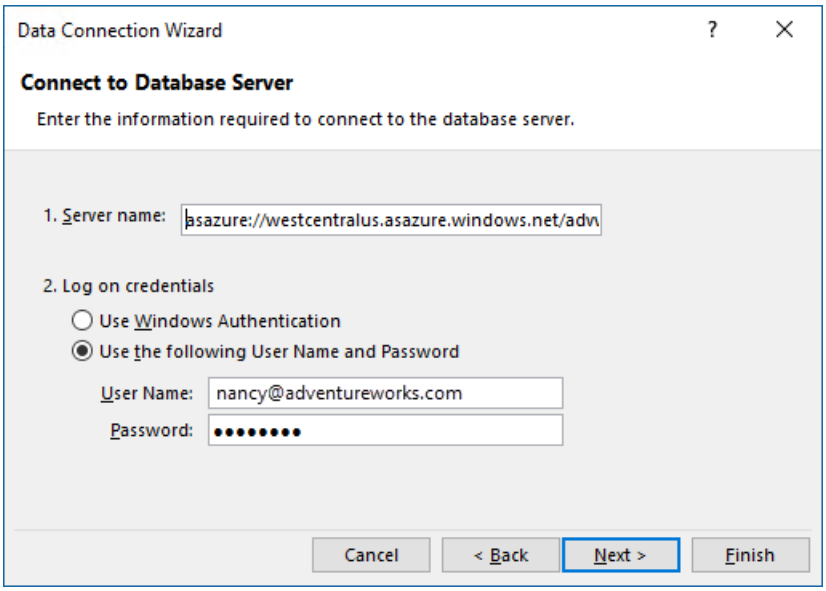
The account you sign in with must belong to a model database role with at least read permissions. To learn more, see Authentication and user permissions.

Connect in Excel

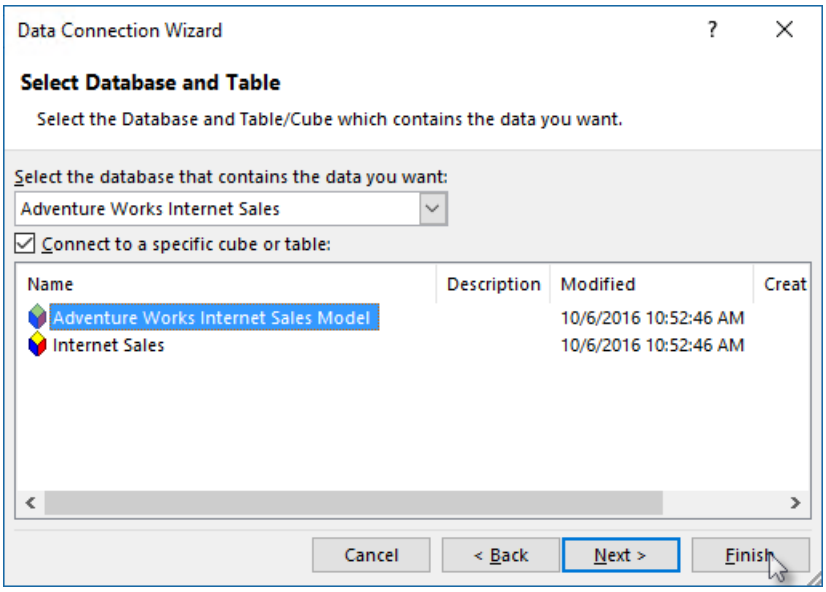
Connecting to a server in Excel is supported by using Get Data in Excel 2016 and later. Connecting by using the Import Table Wizard in Power Pivot is not supported.

1. In Excel, on the **Data** ribbon, click **Get Data** > **From Database** > **From Analysis Services**.
2. In the Data Connection Wizard, in **Server name**, enter the server name including protocol and URI. For example, asazure://westcentralus.asazure.windows.net/advworks. Then, in **Logon credentials**, select **Use the following User Name and Password**, and then type the organizational user name, for example nancy@adventureworks.com, and password.

   > **IMPORTANT**
   > If you sign in with a Microsoft Account, Live ID, Yahoo, Gmail, etc., or you are required to sign in with multi-factor authentication, leave the password field blank. You are prompted for a password after clicking Next.

3. In **Select Database and Table**, select the database and model or perspective, and then click **Finish**.



See also

Client libraries

Manage your server

Connect with Power BI

7/2/2021 • 2 minutes to read • Edit Online

Once you've created a server in Azure, and deployed a tabular model to it, users in your organization are ready to connect and begin exploring data.

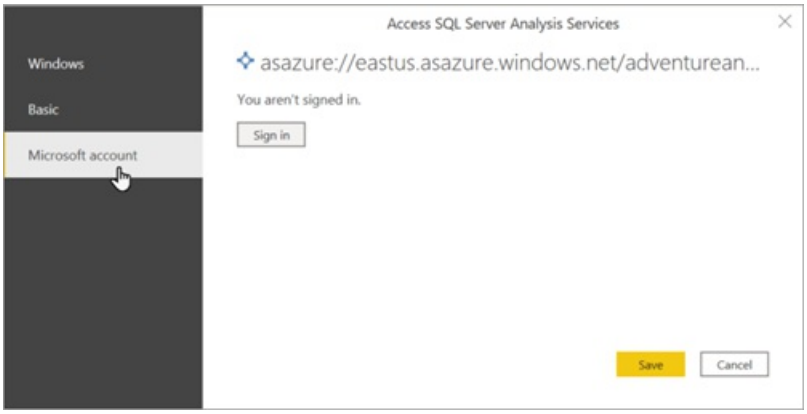
> **NOTE**
> If publishing a Power BI Desktop model to the Power BI service, on the Azure Analysis Services server, ensure the Case-Sensitive collation server property is not selected (default). The Case-Sensitive server property can be set by using SQL Server Management Studio.

Connect in Power BI Desktop

1. In Power BI Desktop, click **Get Data** > **Azure** > **Azure Analysis Services database**.
2. In **Server**, enter the server name. Be sure to include the full URL; for example, asazure://westcentralus.asazure.windows.net/advworks.
3. In **Database**, if you know the name of the tabular model database or perspective you want to connect to, paste it here. Otherwise, you can leave this field blank and select a database or perspective later.
4. Select a connection option and then press **Connect**.

   Both **Connect live** and **Import** options are supported. However, we recommended you use live connections because Import mode does have some limitations; most notably, server performance might be impacted during import.

   If you have a Power BI model in Mixed storage mode, the **Connect live** option is replaced by the **DirectQuery** option. Live connections are also automatically upgraded to DirectQuery if the model is switched from Import to Mixed storage mode.
5. When prompted to enter your credentials, select **Microsoft account**, and then click **Sign in**.



> **NOTE**
> Windows and Basic authentication are not supported.

6. In **Navigator**, expand the server, then select the model or perspective you want to connect to, and then click **Connect**. Click a model or perspective to show all objects for that view.

The model opens in Power BI Desktop with a blank report in Report view. The Fields list displays all non-hidden model objects. Connection status is displayed in the lower-right corner.

Connect in Power BI (service)

1. Create a Power BI Desktop file that has a live connection to your model on your server.
2. In Power BI, click **Get Data** > **Files**, and then locate and select your .pbix file.

Request Memory Limit

To safeguard the performance of the system, a memory limit is enforced for all queries issued by Power BI reports against Azure Analysis Services, regardless of the Query Memory Limit configured on the Azure Analysis Services server. Users should consider simplifying the query or its calculations if the query is too memory intensive.

QUERY TYPE	REQUEST MEMORY LIMIT
Live connect from Power BI	10 GB
DirectQuery from Power BI report in Shared workspace	1 GB
DirectQuery from Power BI report in Premium workspace	10 GB
Power BI Q&A	100 MB

See also

Connect to Azure Analysis Services
Client libraries

Create an Office Data Connection file

4/27/2021 • 2 minutes to read • Edit Online

Information in this article describes how you can create an Office Data Connection file to connect to an Azure Analysis Services server from Excel 2016 version number 16.0.7369.2117 or earlier, or Excel 2013. An updated MSOLAP.7 provider is also required.

1. Copy the sample connection file below and paste into a text editor.
2. In `odc:ConnectionString`, change the following properties:
   - In `Data Source=asazure://<region>.asazure.windows.net/<servername>;` change `<region>` to the region of your Analysis Services server and `<servername>` to the name of your server.
   - In `Initial Catalog=<database>;` change `<database>` to the name of your database.
3. In `<odc:CommandText>Model</odc:CommandText>` change `Model` to the name of your model or perspective.
4. Save the file with an `.odc` extension to the C:\Users*username*\Documents\My Data Sources folder.
5. Right-click the file, and then click **Open in Excel**. Or in Excel, on the **Data** ribbon, click **Existing Connections**, select your file, and then click **Open**.

Sample connection file

```
<html xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns="https://www.w3.org/TR/REC-html40">

<head>
<meta http-equiv=Content-Type content="text/x-ms-odc; charset=utf-8">
<meta name=ProgId content=ODC.Cube>
<meta name=SourceType content=OLEDB>
<meta name=Catalog content="Database">
<meta name=Table content=Model>
<title>AzureAnalysisServicesConnection</title>
<xml id=docprops><o:DocumentProperties
  xmlns:o="urn:schemas-microsoft-com:office:office"
  xmlns="https://www.w3.org/TR/REC-html40">
  <o:Name>SampleAzureAnalysisServices</o:Name>
 </o:DocumentProperties>
</xml><xml id=msodc><odc:OfficeDataConnection
  xmlns:odc="urn:schemas-microsoft-com:office:odc"
  xmlns="https://www.w3.org/TR/REC-html40">
  <odc:Connection odc:Type="OLEDB">
   <odc:ConnectionString>Provider=MSOLAP.7;Data
Source=asazure://<region>.asazure.windows.net/<servername>;Initial Catalog=<database>;
</odc:ConnectionString>
   <odc:CommandType>Cube</odc:CommandType>
   <odc:CommandText>Model</odc:CommandText>
  </odc:Connection>
 </odc:OfficeDataConnection>
</xml>
<style>
<!--
    .ODCDataSource
    {
    behavior: url(dataconn.htc);
    }
-->
</style>
```

```
</head>

<body onload='init()' scroll=no leftmargin=0 topmargin=0 rightmargin=0 style='border: 0px'>
<table style='border: solid 1px threedface; height: 100%; width: 100%' cellpadding=0 cellspacing=0
width='100%'>
  <tr>
    <td id=tdName style='font-family:arial; font-size:medium; padding: 3px; background-color: threedface'>
       
    </td>
     <td id=tdTableDropdown style='padding: 3px; background-color: threedface; vertical-align: top; padding-
bottom: 3px'>

       
    </td>
  </tr>
  <tr>
    <td id=tdDesc colspan='2' style='border-bottom: 1px threedshadow solid; font-family: Arial; font-size:
1pt; padding: 2px; background-color: threedface'>

       
    </td>
  </tr>
  <tr>
    <td colspan='2' style='height: 100%; padding-bottom: 4px; border-top: 1px threedhighlight solid;'>
      <div id='pt' style='height: 100%' class='ODCDataSource'></div>
    </td>
  </tr>
</table>


<script language='javascript'>

function init() {
  var sName, sDescription;
  var i, j;

  try {
    sName = unescape(location.href)

    i = sName.lastIndexOf(".")
    if (i>=0) { sName = sName.substring(1, i); }

    i = sName.lastIndexOf("/")
    if (i>=0) { sName = sName.substring(i+1, sName.length); }

    document.title = sName;
    document.getElementById("tdName").innerText = sName;

    sDescription = document.getElementById("docprops").innerHTML;

    i = sDescription.indexOf("escription>")
    if (i>=0) { j = sDescription.indexOf("escription>", i + 11); }

    if (i>=0 && j >= 0) {
      j = sDescription.lastIndexOf("</", j);

      if (j>=0) {
          sDescription = sDescription.substring(i+11, j);
        if (sDescription != "") {
            document.getElementById("tdDesc").style.fontSize="x-small";
          document.getElementById("tdDesc").innerHTML = sDescription;
          }
        }
      }
    }
  catch(e) {

    }
```

```
  }
</script>

</body>

</html>
```

Manage Azure Analysis Services with PowerShell

4/27/2021 • 2 minutes to read • Edit Online

This article describes PowerShell cmdlets used to perform Azure Analysis Services server and database management tasks.

Server resource management tasks like creating or deleting a server, suspending or resuming server operations, or changing the service level (tier) use Azure Analysis Services cmdlets. Other tasks for managing databases like adding or removing role members, processing, or partitioning use cmdlets included in the same SqlServer module as SQL Server Analysis Services.

> **NOTE**
> This article has been updated to use the Azure Az PowerShell module. The Az PowerShell module is the recommended PowerShell module for interacting with Azure. To get started with the Az PowerShell module, see Install Azure PowerShell. To learn how to migrate to the Az PowerShell module, see Migrate Azure PowerShell from AzureRM to Az.

Permissions

Most PowerShell tasks require you have Admin privileges on the Analysis Services server you are managing. Scheduled PowerShell tasks are unattended operations. The account or service principal running the scheduler must have Admin privileges on the Analysis Services server.

For server operations using Azure PowerShell cmdlets, your account or the account running scheduler must also belong to the Owner role for the resource in Azure role-based access control (Azure RBAC).

Resource and server operations

Install module - Az.AnalysisServices
Documentation - Az.AnalysisServices reference

Database operations

Azure Analysis Services database operations use the same SqlServer module as SQL Server Analysis Services. However, not all cmdlets are supported for Azure Analysis Services.

The SqlServer module provides task-specific database management cmdlets as well as the general-purpose Invoke-ASCmd cmdlet that accepts a Tabular Model Scripting Language (TMSL) query or script. The following cmdlets in the SqlServer module are supported for Azure Analysis Services.

Install module - SqlServer
Documentation - SqlServer reference

Supported cmdlets

CMDLET	DESCRIPTION
Add-RoleMember	Add a member to a database role.
Backup-ASDatabase	Backup an Analysis Services database.

CMDLET	DESCRIPTION
Remove-RoleMember	Remove a member from a database role.
Invoke-ASCmd	Execute a TMSL script.
Invoke-ProcessASDatabase	Process a database.
Invoke-ProcessPartition	Process a partition.
Invoke-ProcessTable	Process a table.
Merge-Partition	Merge a partition.
Restore-ASDatabase	Restore an Analysis Services database.

Related information

- SQL Server PowerShell
- Download SQL Server PowerShell Module
- Download SSMS
- SqlServer module in PowerShell Gallery
- Tabular Model Programming for Compatibility Level 1200 and higher

Analysis Services resource and object limits

3/29/2021 • 2 minutes to read • Edit Online

This article describes resource and model object limits.

Tier limits

For QPU and Memory limits for developer, basic, and standard tiers, see the Azure Analysis Services pricing page.

Object limits

These limits are theoretical. Performance will be diminished at lower numbers.

OBJECT	MAXIMUM SIZES/NUMBERS
Databases in an instance	16,000
Combined number of tables and columns in a database	16,000
Rows in a table	Unlimited **Warning:** With the restriction that no single column in the table can have more than 1,999,999,997 distinct values.
Hierarchies in a table	15,999
Levels in a hierarchy	15,999
Relationships	8,000
Key Columns in all table	15,999
Measures in tables	2^31-1 = 2,147,483,647
Cells returned by a query	2^31-1 = 2,147,483,647
Record size of the source query	64 K
Length of object names	512 characters

Azure Analysis Services samples

4/27/2021 • 2 minutes to read • Edit Online

Use the following sample resources to help you learn about and test Analysis Services for your environment.

Code samples

The Analysis Services repository on GitHub includes open source code samples and community projects.

Tabular model project and database samples

Adventure Works for Analysis Services on GitHub is the most commonly used sample tabular model project. You can download a VS project or a completed sample tabular model database.

Sample database on Azure

Azure Synapse Analytics provides a sample AdventureWorksDW database that can be included in a provisioned resource. To learn more, see Quickstart: Create and query a dedicated SQL pool in Azure Synapse Analytics .

Sample databases on GitHub

These sample databases on GitHub can be used for creating and testing your own models.

Adventure Works sample databases

Wide World Importers sample databases

Adventure Works Internet Sales sample model in Azure portal

If you have an Analysis Services server in Azure portal, you can quickly and easily create a sample model. In your server's overview page in Azure portal, click **New model**, and then in **Choose a data source**, select **Sample data**.

Next steps

Adventure Works tutorial
Azure On-premises Data Gateway