

Request Body

```
{
  "customKeyIdentifier": "SUFNX1NTMF80LzYvMjAx0CAxMjoxMT0zNyBBTQ==",
  "endDateTime": "2021-04-06T00:11:36Z",
  "keyId": "e101c4ec-3e89-4d3b-9ba3-4f524cb6eeda",
  "startDateTime": "2018-04-06T00:11:36Z",
  "secretText": null,
  "hint": null
}
```

Response Preview

```
{
  "@odata.context": "https://graph.microsoft.com/beta/$metadata#servicePrincipals/$entity",
  "id": "8164e784-8a01-46c9-89fd-3076bd9656d0",
  "deletedDateTime": null,
  "accountEnabled": true,
```

g. Extract the appRoles property from the service principal object.

```
"appRoles": [
  {
    "allowedMemberTypes": [
      "User"
    ],
    "description": "msiam_access",
    "displayName": "msiam_access",
    "id": "7dfd756e-8c27-4472-b2b7-38c17fc5de5e",
    "isEnabled": true,
    "origin": "Application",
    "value": null
  },
  {
    "allowedMemberTypes": [
      "User"
    ],
    "description": "redacted",
    "displayName": "redacted",
    "id": "redacted",
    "isEnabled": true,
    "origin": "ServicePrincipal",
    "value": "arn:aws:iam::redacted:role/redacted"
  }
],
```

h. You now need to generate new roles for your application.

i. The following JSON code is an example of an appRoles object. Create a similar object to add the roles you want for your application.

```
{
  "appRoles": [
    {
      "allowedMemberTypes": [
        "User"
      ],
      "description": "msiam_access",
      "displayName": "msiam_access",
      "id": "7dfd756e-8c27-4472-b2b7-38c17fc5de5e",
      "isEnabled": true,
      "origin": "Application",
      "value": null
    },
    {
      "allowedMemberTypes": [
        "User"
      ],
      "description": "Admin,WAAD",
      "displayName": "Admin,WAAD",
      "id": "4aacf5a4-f38b-4861-b909-bae023e88dde",
      "isEnabled": true,
      "origin": "ServicePrincipal",
      "value": "arn:aws:iam::12345:role/Admin,arn:aws:iam::12345:saml-provider/WAAD"
    },
    {
      "allowedMemberTypes": [
        "User"
      ],
      "description": "Auditors,WAAD",
      "displayName": "Auditors,WAAD",
      "id": "bcad6926-67ec-445a-80f8-578032504c09",
      "isEnabled": true,
      "origin": "ServicePrincipal",
      "value": "arn:aws:iam::12345:role/Auditors,arn:aws:iam::12345:saml-provider/WAAD"
    }
  ]
}
```

ⓘ Note

You can add new roles only after you've added *msiam_access* for the patch operation. You can also add as many roles as you want, depending on your organization's needs. Microsoft Entra ID sends the *value* of these roles as the claim value in the SAML response.

- j. In Microsoft Graph Explorer, change the method from **GET** to **PATCH**. Patch the service principal object with the roles you want by updating the *appRoles* property,

like the one shown in the preceding example. Select **Run Query** to execute the patch operation. A success message confirms the creation of the role for your AWS application.

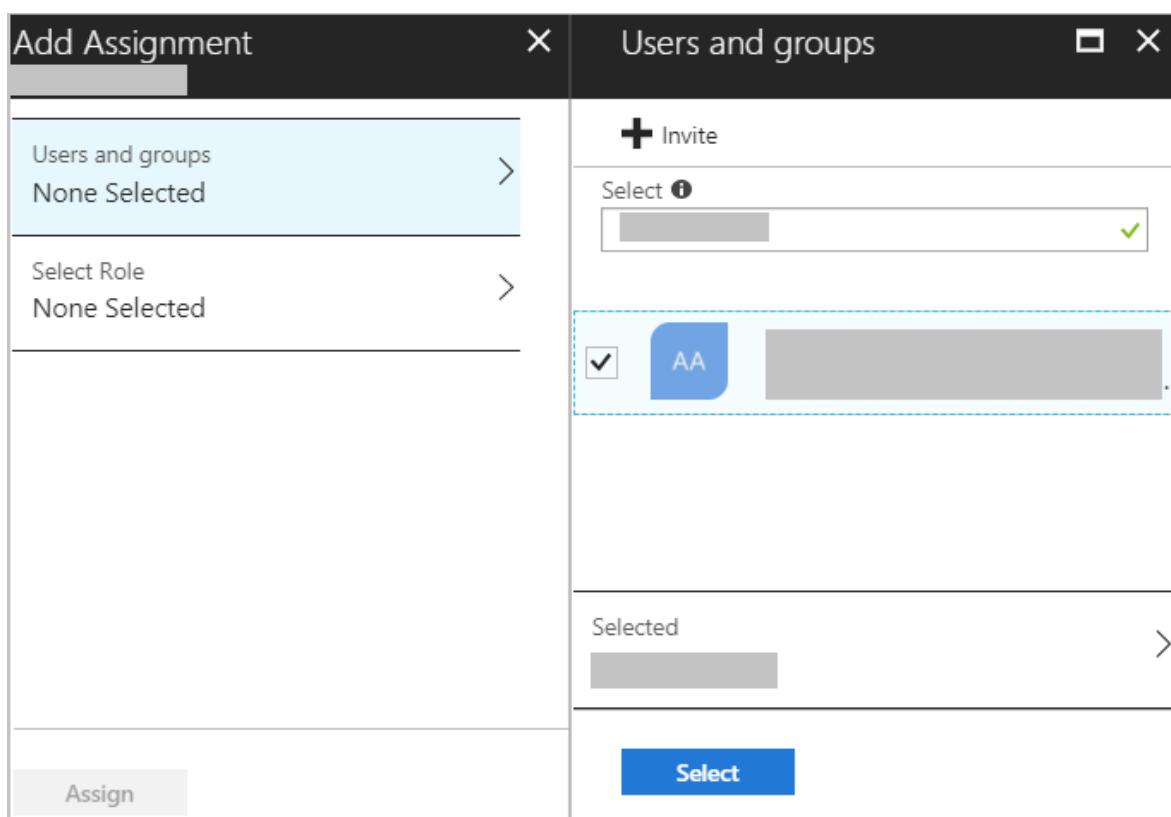


The screenshot shows a PATCH request to the Microsoft Graph API endpoint `https://graph.microsoft.com/beta/servicePrincipals/e02179ea-4f97-42f9-b9e7-8216e26e1d69`. The Request Body contains the following JSON payload:

```
{ "allowedMemberTypes": [ "User" ], "description": "Admin_WAAD", "displayName": "Admin_WAAD", "id": "4aacf5a4-f38b-4861-b909-bae023e88dde", "isEnabled": true, "origin": "ServicePrincipal", "value": "arn:aws:iam:::role/Admin,arn:aws:iam:::saml-provider/WAAD" }
```

The response status is **Success - Status Code 204** with a **704ms** response time.

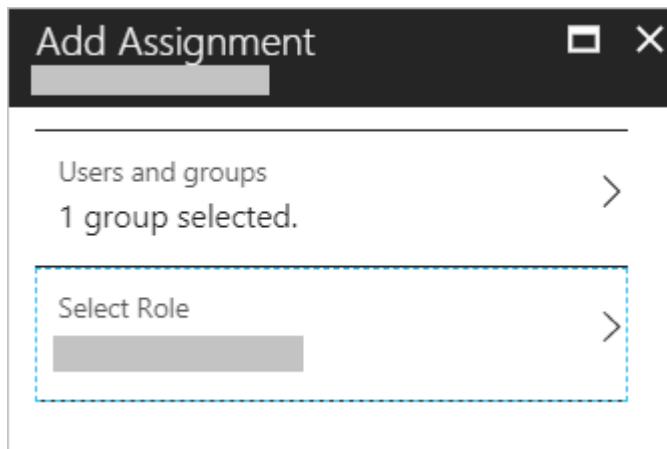
4. After the service principal is patched with more roles, you can assign users and groups to their respective roles. You do this in the Azure portal by going to the AWS application and then selecting the **Users and Groups** tab at the top.
5. We recommend that you create a new group for every AWS role so that you can assign that particular role in the group. This one-to-one mapping means that one group is assigned to one role. You can then add the members who belong to that group.
6. After you've created the groups, select the group and assign it to the application.



! Note

Nested groups are not supported when you assign groups.

7. To assign the role to the group, select the role, and then select **Assign**.



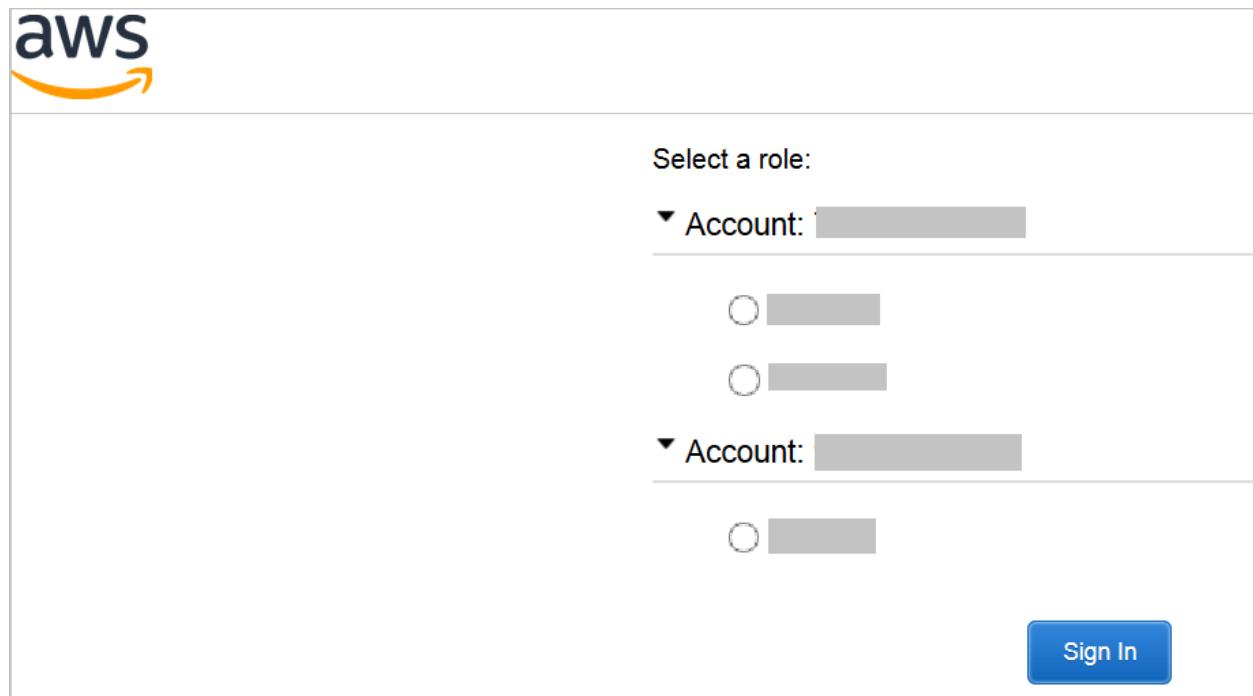
ⓘ Note

After you've assigned the roles, you can view them by refreshing your Azure portal session.

Test SSO

In this section, you test your Microsoft Entra single sign-on configuration by using Microsoft My Apps.

When you select the AWS tile in My Apps, the AWS application page opens with an option to select the role.



You can also verify the SAML response to see the roles being passed as claims.

```
<Attribute Name="http://schemas.microsoft.com/ws/2008/06/identity/claims/role">
  <AttributeValue>arn:aws:iam::[REDACTED]:role/Admin1,arn:aws:iam::[REDACTED]:saml-provider/WAAD1</AttributeValue>
  <AttributeValue>arn:aws:iam::[REDACTED]:role/Admin,arn:aws:iam::[REDACTED]:saml-provider/WAAD</AttributeValue>
  <AttributeValue>arn:aws:iam::[REDACTED]:role/Auditors,arn:aws:iam::[REDACTED]:saml-provider/WAAD1</AttributeValue>
</Attribute>
```

For more information about My Apps, see [Sign in and start apps from the My Apps portal](#).

Next steps

After you configure AWS you can enforce session control, which protects the exfiltration and infiltration of your organization's sensitive data in real time. Session control extends from Conditional Access. For more information, see [Learn how to enforce session control with Microsoft Defender for Cloud Apps](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Tutorial: Microsoft Entra single sign-on (SSO) integration with AWS ClientVPN

Article • 10/23/2023

In this tutorial, you'll learn how to integrate AWS ClientVPN with Microsoft Entra ID. When you integrate AWS ClientVPN with Microsoft Entra ID, you can:

- Control in Microsoft Entra ID who has access to AWS ClientVPN.
- Enable your users to be automatically signed-in to AWS ClientVPN with their Microsoft Entra accounts.
- Manage your accounts in one central location.

Prerequisites

To get started, you need the following items:

- A Microsoft Entra subscription. If you don't have a subscription, you can get a [free account](#).
- AWS ClientVPN single sign-on (SSO) enabled subscription.

Scenario description

In this tutorial, you configure and test Microsoft Entra SSO in a test environment.

- AWS ClientVPN supports **SP** initiated SSO.
- AWS ClientVPN supports **Just In Time** user provisioning.

Note

Identifier of this application is a fixed string value so only one instance can be configured in one tenant.

Add AWS ClientVPN from the gallery

To configure the integration of AWS ClientVPN into Microsoft Entra ID, you need to add AWS ClientVPN from the gallery to your list of managed SaaS apps.

1. Sign in to the Microsoft Entra admin center [↗](#) as at least a [Cloud Application Administrator](#).
2. Browse to **Identity > Applications > Enterprise applications > New application**.
3. In the **Add from the gallery** section, type **AWS ClientVPN** in the search box.
4. Select **AWS ClientVPN** from results panel and then add the app. Wait a few seconds while the app is added to your tenant.

Alternatively, you can also use the [Enterprise App Configuration Wizard ↗](#). In this wizard, you can add an application to your tenant, add users/groups to the app, assign roles, as well as walk through the SSO configuration as well. [Learn more about Microsoft 365 wizards](#).

Configure and test Microsoft Entra SSO for AWS ClientVPN

Configure and test Microsoft Entra SSO with AWS ClientVPN using a test user called **B.Simon**. For SSO to work, you need to establish a link relationship between a Microsoft Entra user and the related user in AWS ClientVPN.

To configure and test Microsoft Entra SSO with AWS ClientVPN, perform the following steps:

1. [Configure Microsoft Entra SSO](#) - to enable your users to use this feature.
 - a. [Create a Microsoft Entra test user](#) - to test Microsoft Entra single sign-on with B.Simon.
 - b. [Assign the Microsoft Entra test user](#) - to enable B.Simon to use Microsoft Entra single sign-on.
2. [Configure AWS ClientVPN SSO](#) - to configure the single sign-on settings on application side.
 - a. [Create AWS ClientVPN test user](#) - to have a counterpart of B.Simon in AWS ClientVPN that is linked to the Microsoft Entra representation of user.
3. [Test SSO](#) - to verify whether the configuration works.

Configure Microsoft Entra SSO

Follow these steps to enable Microsoft Entra SSO.

1. Sign in to the Microsoft Entra admin center [↗](#) as at least a [Cloud Application Administrator](#).

2. Browse to **Identity > Applications > Enterprise applications > AWS ClientVPN > Single sign-on.**
3. On the **Select a single sign-on method** page, select **SAML**.
4. On the **Set up single sign-on with SAML** page, click the pencil icon for **Basic SAML Configuration** to edit the settings.

Set up Single Sign-On with SAML

An SSO implementation based on federation protocols improves security, reliability, and end user experiences and is easier to implement. Choose SAML single sign-on whenever possible for existing applications that do not use OpenID Connect or OAuth. [Learn more.](#)

Read the [configuration guide](#) for help integrating <App Name>

1	Basic SAML Configuration Identifier (Entity ID) Reply URL (Assertion Consumer Service URL) Sign on URL Relay State (Optional) Logout Url (Optional)	 Edit
---	---	--

5. On the **Basic SAML Configuration** section, perform the following steps:

- a. In the **Sign on URL** text box, type a URL using the following pattern:

`https://<LOCALHOST>`

- b. In the **Reply URL** text box, type a URL using one of the following patterns:

Reply URL

`http://<LOCALHOST>`

`https://self-service.clientvpn.amazonaws.com/api/auth/sso/saml`

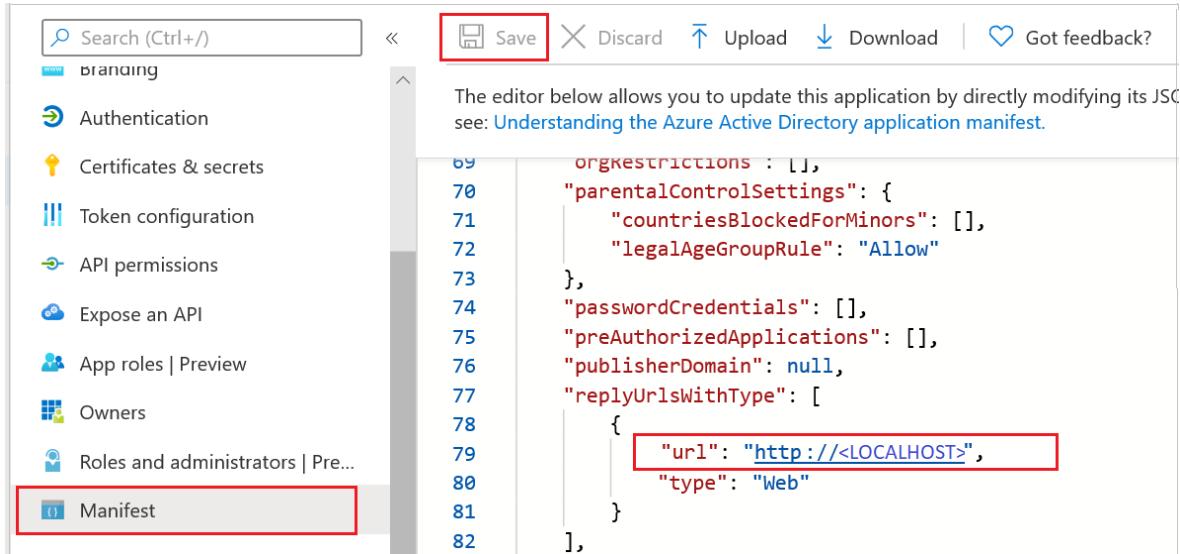
! Note

These values are not real. Update these values with the actual Sign on URL and Reply URL. The Sign on URL and Reply URL can have the same value (`http://127.0.0.1:35001`). Refer to [AWS Client VPN Documentation](#) for details. You can also refer to the patterns shown in the **Basic SAML Configuration** section. Contact [AWS ClientVPN support team](#) for any configuration issues.

6. In the Microsoft Entra service, navigate to **App registrations** and then select **All Applications**.

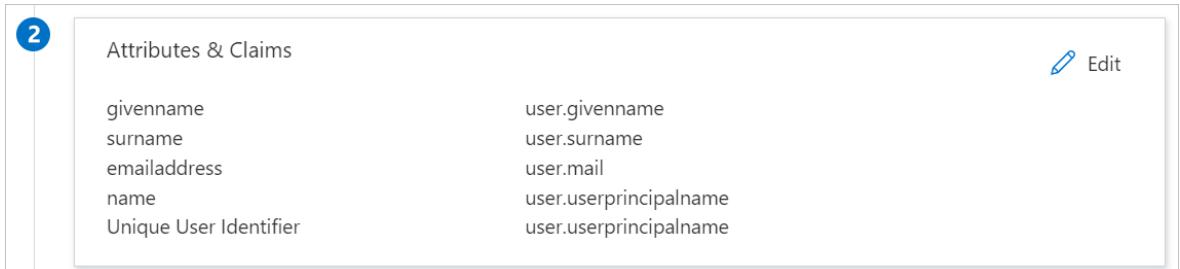
7. Type AWS ClientVPN in the search box and select **AWS ClientVPN** from the search panel.

8. Click on **Manifest** and you need to keep the Reply URL as **http** instead of **https** to get the integration working, click on **Save**.



```
69  "orgRestrictions": [],
70  "parentalControlSettings": {
71    "countriesBlockedForMinors": [],
72    "legalAgeGroupRule": "Allow"
73  },
74  "passwordCredentials": [],
75  "preAuthorizedApplications": [],
76  "publisherDomain": null,
77  "replyUrlsWithType": [
78    {
79      "url": "http://<LOCALHOST>",
80      "type": "Web"
81    }
82  ],
83}
```

9. AWS ClientVPN application expects the SAML assertions in a specific format, which requires you to add custom attribute mappings to your SAML token attributes configuration. The following screenshot shows the list of default attributes.



Attributes & Claims	
givenname	user.givenname
surname	user.surname
emailaddress	user.mail
name	user.userprincipalname
Unique User Identifier	user.userprincipalname

10. In addition to above, AWS ClientVPN application expects few more attributes to be passed back in SAML response which are shown below. These attributes are also pre populated but you can review them as per your requirements.

Name	Source Attribute
memberOf	user.groups
FirstName	user.givenname
LastName	user.surname

11. On the **Set up single sign-on with SAML** page, in the **SAML Signing Certificate** section, find **Federation Metadata XML** and select **Download** to download the certificate and save it on your computer.

3 SAML Signing Certificate

Status	Active
Thumbprint	<Thumbprintvalue >
Expiration	<Expiration >
Notification Email	<Email address>
App Federation Metadata Url	<App Federation Metadata Url> Download
Certificate (Base64)	Download
Certificate (Raw)	Download
Federation Metadata XML	Download

12. In the **SAML Signing Certificate** section, click the edit icon and change the **Signing Option** to **Sign SAML response and assertion**. Click **Save**.

SAML Signing Certificate

Manage the certificate used by Azure AD to sign SAML tokens issued to your app

[Save](#) [New Certificate](#) [Import Certificate](#) [Got feedback?](#)

Status	Expiration Date	Thumbprint
Active		...
Signing Option	Sign SAML response and assertion	▼
Signing Algorithm	SHA-256	▼

13. On the **Set up AWS ClientVPN** section, copy the appropriate URL(s) based on your requirement.

Set up <Application Name>

You'll need to configure the application to link with Microsoft Entra ID.

Login URL	https://login.microsoftonline.com/4f74... Download
Microsoft Entra ID Identifier	https://sts.windows.net/4f7437a6-3d76... Download
Logout URL	https://login.microsoftonline.com/4f74... Download

Create a Microsoft Entra test user

In this section, you'll create a test user called B.Simon.

1. Sign in to the [Microsoft Entra admin center](#) as at least a **User Administrator**.
2. Browse to **Identity > Users > All users**.
3. Select **New user > Create new user**, at the top of the screen.
4. In the **User** properties, follow these steps:
 - a. In the **Display name** field, enter **B.Simon**.

- b. In the **User principal name** field, enter the `username@companydomain.extension`. For example, `B.Simon@contoso.com`.
 - c. Select the **Show password** check box, and then write down the value that's displayed in the **Password** box.
 - d. Select **Review + create**.
5. Select **Create**.

Assign the Microsoft Entra test user

In this section, you'll enable B.Simon to use single sign-on by granting access to AWS ClientVPN.

1. Sign in to the [Microsoft Entra admin center](#) as at least a **Cloud Application Administrator**.
2. Browse to **Identity** > **Applications** > **Enterprise applications** > **AWS ClientVPN**.
3. In the app's overview page, select **Users and groups**.
4. Select **Add user/group**, then select **Users and groups** in the **Add Assignment** dialog.
 - a. In the **Users and groups** dialog, select **B.Simon** from the **Users** list, then click the **Select** button at the bottom of the screen.
 - b. If you are expecting a role to be assigned to the users, you can select it from the **Select a role** dropdown. If no role has been set up for this app, you see "Default Access" role selected.
 - c. In the **Add Assignment** dialog, click the **Assign** button.

Configure AWS ClientVPN SSO

Follow the instructions given in the [link](#) to configure single sign-on on AWS ClientVPN side.

Create AWS ClientVPN test user

In this section, a user called Britta Simon is created in AWS ClientVPN. AWS ClientVPN supports just-in-time user provisioning, which is enabled by default. There is no action item for you in this section. If a user doesn't already exist in AWS ClientVPN, a new one is created after authentication.

Test SSO

In this section, you test your Microsoft Entra single sign-on configuration with following options.

- Click on **Test this application**, this will redirect to AWS ClientVPN Sign-on URL where you can initiate the login flow.
- Go to AWS ClientVPN Sign-on URL directly and initiate the login flow from there.
- You can use Microsoft My Apps. When you click the AWS ClientVPN tile in the My Apps, this will redirect to AWS ClientVPN Sign-on URL. For more information about the My Apps, see [Introduction to the My Apps](#).

Next steps

Once you configure AWS ClientVPN you can enforce session control, which protects exfiltration and infiltration of your organization's sensitive data in real time. Session control extends from Conditional Access. [Learn how to enforce session control with Microsoft Defender for Cloud Apps](#).

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

Attach and detach policies for Amazon Web Services (AWS) identities

Article • 10/23/2023

This article describes how you can attach and detach permissions for users, roles, and groups for Amazon Web Services (AWS) identities using the **Remediation** dashboard.

Note

To view the **Remediation** tab, you must have **Viewer**, **Controller**, or **Administrator** permissions. To make changes on this tab, you must have **Controller** or **Administrator** permissions. If you don't have these permissions, contact your system administrator.

View permissions

1. On the Permissions Management home page, select the **Remediation** tab, and then select the **Permissions** subtab.
2. From the **Authorization System Type** dropdown, select **AWS**.
3. From the **Authorization System** dropdown, select the accounts you want to access.
4. From the **Search For** dropdown, select **Group**, **User**, or **Role**.
5. To search for more parameters, you can make a selection from the **User States**, **Permission Creep Index**, and **Task Usage** dropdowns.
6. Select **Apply**. Permissions Management displays a list of users, roles, or groups that match your criteria.
7. In **Enter a username**, enter or select a user.
8. In **Enter a group name**, enter or select a group, then select **Apply**.
9. Make a selection from the results list.

The table displays the related **Username**, **Domain/Account**, **Source** and **Policy Name**.

Attach policies

1. On the Permissions Management home page, select the **Remediation** tab, and then select the **Permissions** subtab.
2. From the **Authorization System Type** dropdown, select **AWS**.
3. In **Enter a username**, enter or select a user.
4. In **Enter a Group Name**, enter or select a group, then select **Apply**.
5. Make a selection from the results list.
6. To attach a policy, select **Attach Policies**.
7. In the **Attach Policies** page, from the **Available policies** list, select the plus sign (+) to move the policy to the **Selected policies** list.
8. When you have finished adding policies, select **Submit**.
9. When the following message displays: **Are you sure you want to change permission?**, select:
 - **Generate Script** to generate a script where you can manually add/remove the permissions you selected.
 - **Execute** to change the permission.
 - **Close** to cancel the action.

Detach policies

1. On the Permissions Management Permissions Management home page, select the **Remediation** tab, and then select the **Permissions** subtab.
2. From the **Authorization System Type** dropdown, select **AWS**.
3. In **Enter a username**, enter or select a user.
4. In **Enter a Group Name**, enter or select a group, then select **Apply**.
5. Make a selection from the results list.
6. To remove a policy, select **Detach Policies**.
7. In the **Detach Policies** page, from the **Available policies** list, select the plus sign (+) to move the policy to the **Selected policies** list.
8. When you have finished selecting policies, select **Submit**.
9. When the following message displays: **Are you sure you want to change permission?**, select:
 - **Generate Script** to generate a script where you can manually add/remove the permissions you selected.
 - **Execute** to change the permission.
 - **Close** to cancel the action.

Next steps

- To revoke high-risk and unused tasks or assign read-only status for Microsoft Azure and Google Cloud Platform (GCP) identities, see [Revoke high-risk and unused tasks or assign read-only status for Azure and GCP identities](#) To create or approve a request for permissions, see [Create or approve a request for permissions](#).

AKS for Amazon EKS professionals

Article • 01/03/2023

This series of articles helps professionals who are familiar with Amazon Elastic Kubernetes Service (Amazon EKS) to understand [Azure Kubernetes Service \(AKS\)](#). The series highlights key similarities and differences between these two managed Kubernetes solutions.

The series articles compare AKS with Amazon EKS for the following Kubernetes design areas:

- [Identity and access management](#)
- [Cluster logging and monitoring](#)
- [Secure network topologies](#)
- [Storage options](#)
- [Cost optimization and management](#)
- [Agent node and node pool management](#)
- [Cluster governance](#)

These articles provide recommended architectures and practices to improve AKS deployment security, compliance, management, and observability. For basic AKS implementation, see [Baseline architecture for an Azure Kubernetes Service \(AKS\) cluster](#) and [AKS landing zone accelerator](#).

AKS isn't the only way to run containers in Azure, and Amazon EKS is only one of the container options for Amazon Web Services (AWS). These articles don't compare Azure services like Azure Container Apps, Azure Container Instances, and Azure App Service with AWS services like Amazon Elastic Container Service or AWS Fargate.

For more information about other Azure services that can host containerized workloads, see [Choose an Azure compute service](#) and [Compare Container Apps with other Azure container options](#).

The following articles compare Azure and AWS core platform components and capabilities:

- [Azure and AWS accounts and subscriptions](#)
- [Compute services on Azure and AWS](#)
- [Relational database technologies on Azure and AWS](#)
- [Messaging services on Azure and AWS](#)
- [Networking on Azure and AWS](#)
- [Regions and zones on Azure and AWS](#)

- Resource management on Azure and AWS
- Multicloud security and identity with Azure and AWS
- Compare storage on Azure and AWS

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Laura Nicolas](#) | Senior Software Engineer
- [Paolo Salvatori](#) | Principal Service Engineer

Other contributors:

- [Chad Kittel](#) | Principal Software Engineer
- [Ed Price](#) | Senior Content Program Manager
- [Theano Petersen](#) | Technical Writer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- Kubernetes identity and access management
- Kubernetes monitoring and logging
- Secure network access to Kubernetes
- Storage options for a Kubernetes cluster
- Cost management for Kubernetes
- Kubernetes node and node pool management
- Cluster governance

Related resources

- [Baseline architecture for an Azure Kubernetes Service \(AKS\) cluster](#)
- [AKS landing zone accelerator](#)

Kubernetes workload identity and access

Microsoft Entra ID

Azure Kubernetes Service (AKS)

This article describes how Amazon Elastic Kubernetes Service (Amazon EKS) and Azure Kubernetes Service (AKS) provide identity for Kubernetes workloads to access cloud platform services. For a detailed comparison of Amazon Web Services (AWS) Identity and Access Management (IAM) and Microsoft Entra ID, see:

- [Microsoft Entra identity management and access management for AWS](#)
- [Mapping AWS IAM concepts to similar ones in Azure](#)

This guide explains how AKS clusters, built-in services, and add-ons use [managed identities](#) to access Azure resources like load balancers and managed disks. The article also demonstrates how to use [Microsoft Entra Workload ID](#) so AKS workloads can access Azure resources without needing a connection string, access key, or user credentials.

ⓘ Note

This article is part of a [series of articles](#) that helps professionals who are familiar with [Amazon EKS](#) to understand [AKS](#).

Amazon EKS identity and access management

Amazon EKS has two native options to call AWS services from within a Kubernetes pod: IAM roles for service accounts, and Amazon EKS service-linked roles.

[IAM roles for service accounts](#) associate IAM roles with a Kubernetes service account. This service account provides AWS permissions to the containers in any pod that uses the service account. IAM roles for service accounts provide the following benefits:

- **Least privilege:** You don't need to provide extended permissions to the node IAM role for pods on that node to call AWS APIs. You can scope IAM permissions to a service account, and only pods that use that service account have access to those permissions. This feature also eliminates the need for third-party solutions such as `kiam` or `kube2iam`.

- **Credential isolation:** A container can only retrieve credentials for the IAM role associated with the service account that it belongs to. A container never has access to credentials for another container that belongs to another pod.
- **Auditability:** [Amazon CloudTrail](#) provides access and event logging to help ensure retrospective auditing.

[Amazon EKS service-linked roles](#) are unique IAM roles that are linked directly to Amazon EKS. Service-linked roles are predefined by Amazon EKS and include all the permissions required to call other AWS services on behalf of the role. For the [Amazon EKS node IAM role](#), the Amazon EKS node `kubelet` daemon calls AWS APIs on behalf of the node. Nodes get permissions for these API calls from an IAM instance profile and associated policies.

AKS cluster managed identities

An AKS cluster requires an identity to access Azure resources like load balancers and managed disks. This identity can be either a managed identity or a service principal. By default, creating an AKS cluster automatically creates a [system-assigned managed identity](#). The Azure platform manages the identity, and you don't need to provision or rotate any secrets. For more information about Microsoft Entra managed identities, see [Managed identities for Azure resources](#).

AKS doesn't create a [service principal](#) automatically, so if you want to use a service principal, you must create it. The service principal eventually expires, and you must renew it to keep the cluster working. Managing service principals adds complexity, so it's easier to use managed identities.

Managed identities are essentially wrappers around service principals that simplify management. The same permission requirements apply both to service principals and managed identities. Managed identities use certificate-based authentication. Each managed identities credential has an expiration of 90 days and is rotated after 45 days.

AKS uses both system-assigned and user-assigned managed identity types, and these identities are immutable. When you create or use an AKS virtual network, attached Azure disk, static IP address, route table, or user-assigned `kubelet` identity with resources outside the [node resource group](#), the Azure CLI adds the role assignment automatically.

If you use another method to create the AKS cluster, such as a Bicep template, Azure Resource Manager (ARM) template, or Terraform module, you need to use the principal ID of the cluster managed identity to do a role assignment. The AKS cluster identity

must have at least [Network Contributor](#) role on the subnet within your virtual network. To define a custom role instead of using the built-in Network Contributor role, you need the following permissions:

- `Microsoft.Network/virtualNetworks/subnets/join/action`
- `Microsoft.Network/virtualNetworks/subnets/read`

When the cluster identity needs to access an existing resource, for example when you deploy an AKS cluster to an existing virtual network, you should use a user-assigned managed identity. If you use a system-assigned control plane identity, the resource provider can't get its principal ID before it creates the cluster, so it's impossible to create the proper role assignments before cluster provisioning.

Summary of managed identities

AKS uses the following [user-assigned managed identities](#) for built-in services and add-ons.

Expand table

Identity	Name	Use case	Default permissions	Bring your own identity
Control plane	AKS Cluster Name	Manages cluster resources including ingress load balancers and AKS-managed public IPs, cluster autoscaler, and Azure Disk and Azure File CSI drivers	Contributor role for node resource group	Supported
Kubelet	AKS Cluster Name-agentpool	Authenticates with Azure	NA (for kubernetes v1.15+)	Supported

Identity	Name	Use case	Default permissions	Bring your own identity
		Container Registry		
Add-on	HTTPApplicationRouting	Manages required network resources	Reader role for node resource group, contributor role for DNS zone	No
Add-on	Ingress application gateway	Manages required network resources	Contributor role for node resource group	No
Add-on	omsagent	Send AKS metrics to Azure Monitor	Monitoring Metrics Publisher role	No
Add-on	Virtual-Node (ACIConnector)	Manages required network resources for Azure Container Instances	Contributor role for node resource group	No

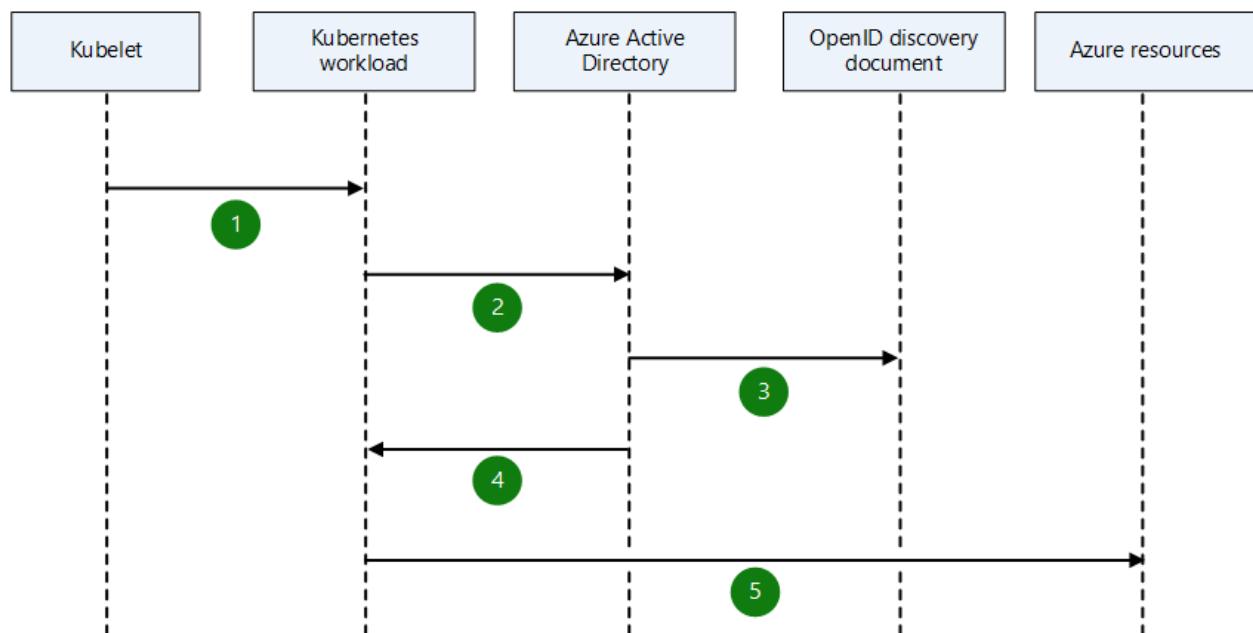
For more information, see [Use a managed identity in Azure Kubernetes Service](#).

Microsoft Entra Workload ID for Kubernetes

Kubernetes workloads require Microsoft Entra application credentials to access Microsoft Entra ID protected resources, such as Azure Key Vault and Microsoft Graph. A common challenge for developers is managing secrets and credentials to secure communication between different components of a solution.

Microsoft Entra Workload ID for Kubernetes [\[2\]](#) eliminates the need to manage credentials to access cloud services like Azure Cosmos DB, Azure Key Vault, or Azure Blob Storage. An AKS-hosted workload application can use Microsoft Entra Workload ID to access an Azure managed service by using a Microsoft Entra security token, instead of explicit credentials like a connection string, username and password, or primary key.

As shown in the following diagram, the Kubernetes cluster becomes a security token issuer that issues tokens to Kubernetes service accounts. You can configure these tokens to be trusted on Microsoft Entra applications. The tokens can then be exchanged for Microsoft Entra access tokens by using the [Azure Identity SDKs](#) or the [Microsoft Authentication Library \(MSAL\)](#) [\[3\]](#).



Microsoft Entra Workload ID federation for Kubernetes is currently supported only for Microsoft Entra applications, but the same model could potentially extend to Azure managed identities.

For more information, automation, and documentation for Microsoft Entra Workload ID, see:

- Azure Workload Identity open-source project [↗](#).
- Workload identity federation
- Microsoft Entra Workload ID federation with Kubernetes [↗](#)
- Microsoft Entra Workload ID federation with external OIDC identity providers [↗](#)
- Minimal Microsoft Entra Workload ID federation [↗](#)
- Microsoft Entra Workload ID [↗](#)
- Microsoft Entra Workload ID quick start [↗](#)
- Use Microsoft Entra Workload ID for Kubernetes with a user-assigned managed identity in a .NET Standard application

Example workload

The example workload runs a frontend and a backend service on an AKS cluster. The workload services use Microsoft Entra Workload ID to access the following Azure services by using Microsoft Entra security tokens:

- Azure Key Vault
- Azure Cosmos DB
- Azure Storage account
- Azure Service Bus namespace

Prerequisites

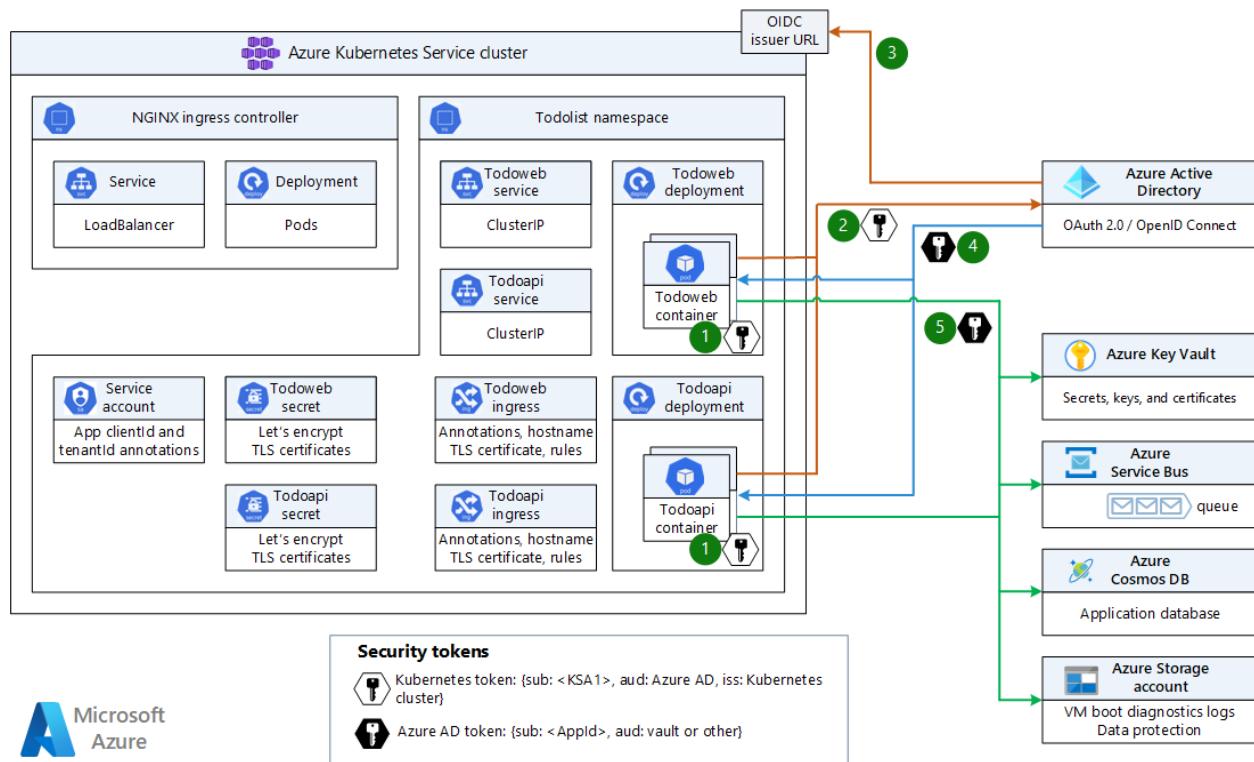
1. Set up an AKS cluster with the [OIDC issuer](#) enabled.
2. Install the [mutating admission webhook](#) [↗](#).
3. Create a Kubernetes service account for the workloads.
4. Create a Microsoft Entra application as shown in the [quickstart](#) [↗](#).
5. Assign roles with the right permissions to the needed Microsoft Entra registered applications.
6. Establish a [federated identity credential](#) [↗](#) between the Microsoft Entra application and the service account issuer and subject.
7. Deploy the workload application to the AKS cluster.

Microsoft Entra Workload ID message flow

AKS applications get security tokens for their service account from the [OIDC issuer](#) of the AKS cluster. Microsoft Entra Workload ID exchanges the security tokens with security

tokens issued by Microsoft Entra ID, and the applications use the Microsoft Entra ID-issued security tokens to access Azure resources.

The following diagram shows how the frontend and backend applications acquire Microsoft Entra security tokens to use Azure platform as a service (PaaS) services.



Download a [Visio file](#) of this architecture.

1. Kubernetes issues a token to the pod when it's scheduled on a node, based on the pod or deployment spec.
2. The pod sends the OIDC-issued token to Microsoft Entra ID to request a Microsoft Entra token for the specific `appId` and resource.
3. Microsoft Entra ID checks the trust on the application and validates the incoming token.
4. Microsoft Entra ID issues a security token: `{sub: appId, aud: requested-audience}`.
5. The pod uses the Microsoft Entra token to access the target Azure resource.

To use Microsoft Entra Workload ID end-to-end in a Kubernetes cluster:

1. You configure the AKS cluster to issue tokens and publish an OIDC discovery document to allow validation of these tokens.
2. You configure the Microsoft Entra applications to trust the Kubernetes tokens.
3. Developers configure their deployments to use the Kubernetes service accounts to get Kubernetes tokens.
4. Microsoft Entra Workload ID exchanges the Kubernetes tokens for Microsoft Entra tokens.

5. AKS cluster workloads use the Microsoft Entra tokens to access protected resources such as Microsoft Graph.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Paolo Salvatori](#) | Principal Service Engineer
- [Martin Gjoshevski](#) | Senior Service Engineer

Other contributors:

- [Laura Nicolas](#) | Senior Software Engineer
- [Chad Kittel](#) | Principal Software Engineer
- [Ed Price](#) | Senior Content Program Manager
- [Theano Petersen](#) | Technical Writer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- AKS for Amazon EKS professionals
- Kubernetes monitoring and logging
- Secure network access to Kubernetes
- Storage options for a Kubernetes cluster
- Cost management for Kubernetes
- Kubernetes node and node pool management
- Cluster governance
- Microsoft Entra identity management and access management for AWS

Related resources

- Use a service principal with Azure Kubernetes Service (AKS)
- Use a managed identity in Azure Kubernetes Service
- Implement Azure Kubernetes Service (AKS)
- Manage identity and access in Microsoft Entra ID

Kubernetes monitoring and logging

Azure Kubernetes Service (AKS)

Azure Log Analytics

Azure Monitor

This article describes how Azure Kubernetes Service (AKS) monitoring compares to Amazon Elastic Kubernetes Service (Amazon EKS). The article guides you on different options to monitor and manage the logs of an AKS cluster and its workloads.

ⓘ Note

This article is part of a [series of articles](#) that helps professionals who are familiar with [Amazon EKS](#) to understand [AKS](#).

Amazon EKS monitoring and logging

Like any Kubernetes service, EKS has two major components, the control plane and worker nodes. There are specific capabilities for each layer.

Amazon EKS control plane and cluster monitoring

Amazon EKS integrates with [Amazon CloudWatch Logs](#) to provide logging and monitoring for the Amazon EKS control plane. This integration isn't enabled by default, but when configured, it gathers logs on:

- API server and API calls.
- Audit logs and user interactions.
- Authenticator logs.
- Scheduler and controller logs.

Amazon EKS exposes [control plane metrics](#) at the `/metrics` endpoint, in Prometheus text format. CloudWatch Container Insights can collect and store [Prometheus metrics](#). You can deploy and self-manage Prometheus on top of your EKS cluster, or use [Amazon Managed service for Prometheus](#).

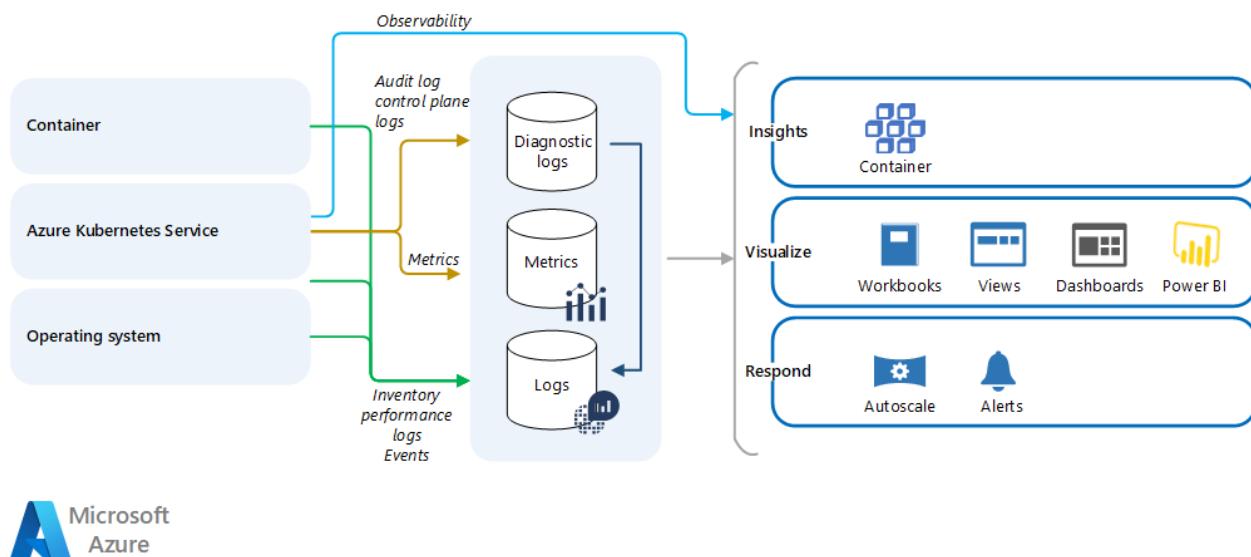
Amazon EKS also integrates with Amazon Web Services (AWS) CloudTrail to track actions and API calls. For more information, see [Logging Amazon EKS API calls with AWS CloudTrail](#).

Amazon EKS workload monitoring

CloudWatch Container Insights [↗](#) can collect and aggregate metrics and logs from containerized applications deployed in EKS. You can implement Container Insights on Amazon EKS with a containerized version of the CloudWatch agent, or by using the AWS Distro for OpenTelemetry [↗](#) as a DaemonSet. You can send logs with FluentBit.

AKS monitoring and logging

Like other Azure resources, AKS generates [platform metrics and resource logs](#) that you can use to monitor its basic health and performance.



Download a [Visio file](#) [↗](#) of this architecture.

Azure Monitor

AKS natively integrates with [Azure Monitor](#). Azure Monitor stores metrics and logs in a central location called a [Log Analytics workspace](#). This data is processed and analyzed to provide insights and alerts. For more information, see [Monitor Azure Kubernetes Service \(AKS\) with Azure Monitor](#).

[Container Insights](#) is the feature of Azure Monitor that collects, indexes, and stores the data your AKS cluster generates. You can configure Container Insights to monitor managed Kubernetes clusters hosted on AKS and other cluster configurations. Container Insights can monitor AKS health and performance with visualization tailored to Kubernetes environments. Similar to EKS, enabling Container Insights for your AKS cluster deploys a containerized version of the Log Analytics agent, which is responsible for sending data to your Log Analytics workspace.

Microsoft Sentinel

[Microsoft Sentinel](#) delivers intelligent security analytics and threat intelligence across the enterprise. With Microsoft Sentinel, you get a single solution for attack detection, threat visibility, proactive hunting, and threat response.

Microsoft Sentinel must be connected with your [AKS](#). This connector lets you stream your Azure Kubernetes Service (AKS) diagnostics logs into Microsoft Sentinel, allowing you to continuously monitor activity in all your instances.

Once you have connected your data sources to Microsoft Sentinel, you can [visualize and monitor the data](#) using the Microsoft Sentinel and Azure Monitor Workbooks, which provides versatility in creating custom dashboards.

AKS cluster and workload monitoring

An AKS deployment can divide into cluster level components, managed AKS components, Kubernetes objects and workloads, applications, and external resources. The following table shows a common strategy for monitoring an AKS cluster and workload applications. Each level has distinct monitoring requirements.

[Expand table](#)

Level	Description	Monitoring requirements
Cluster level components	Virtual machine scale sets abstracted as AKS nodes and node pools	Node status and resource utilization including CPU, memory, disk, and network
Managed AKS components	AKS control plane components including API servers, cloud controller, and <code>kubelet</code>	Control plane logs and metrics from the <code>kube-system</code> namespace
Kubernetes objects and workloads	Kubernetes objects such as deployments, containers, and replica sets	Resource utilization and failures
Applications	Application workloads running on the AKS cluster	Monitoring specific to architecture, but including application logs and service transactions

Level	Description	Monitoring requirements
External	External resources that aren't part of AKS but are required for cluster scalability and management	Specific to each component

- **Cluster level components:** You can use existing Container Insights views and reports to monitor cluster level components to understand their health, readiness, performance, CPU and memory resource utilization, and trends.
- **Managed AKS components:** You can use Metrics Explorer to view the **Inflight Requests** counter. This view includes request latency and work queue processing time.
- **Kubernetes objects and workloads:** You can use existing Container Insights views and reports to monitor deployment, controllers, pods, and containers. Use the **Nodes** and **Controllers** views to view the health and performance of the pods that are running on nodes and controllers, and their resource consumption in terms of CPU and memory.

From the Container Insights **Containers** view, you can view the health and performance of containers, or select an individual container and monitor its events and logs in real time. For details about using this view and analyzing container health and performance, see [Monitor your Kubernetes cluster performance with Container Insights](#).

- **Applications:** You can use [Application Insights](#) to monitor applications that are running on AKS and other environments. Application Insights is an application performance management tool that provides support for many programming languages. Depending on your needs, you can instrument your application code to capture requests, traces, logs, exceptions, custom metrics, and end-to-end transactions, and send this data to Application Insights. If you have a Java application, you can provide monitoring without instrumenting your code. For more information, see [Zero instrumentation application monitoring for Kubernetes](#).
- **External components:** You can monitor external components like service mesh, ingress, and egress with Prometheus and Grafana or other tools. You can use Azure Monitor features to monitor any platform as a service (PaaS) that your workload applications use, such as databases and other Azure resources.

Third-party monitoring solutions

You can set up third-party monitoring solutions like Grafana or Prometheus in your AKS node pools.

- For Grafana, [Grafana Labs](#) provides a dashboard with views of critical API server metrics. You can use this dashboard on your existing Grafana server or set up a new Grafana server in Azure. For more information, see [Monitor your Azure services in Grafana](#).
- [Prometheus](#) is a popular open-source metrics monitoring solution from the [Cloud Native Compute Foundation](#). You can integrate Prometheus with Azure Monitor so you don't need to set up and manage a Prometheus server with a store.

Container Insights provides a seamless onboarding experience to collect Prometheus metrics. You can expose the Prometheus metrics endpoint through your exporters or pod applications, and the containerized agent for Container Insights can scrape the metrics. Container Insights complements and completes end-to-end AKS monitoring, including log collection, which Prometheus as a stand-alone tool doesn't provide. For more information, see [Configure scraping of Prometheus metrics with Container insights](#).

AKS monitoring costs

The Azure Monitor pricing model is primarily based on the amount of data that's ingested per day into your Log Analytics workspace. The cost varies by the plan and retention periods you select.

Before enabling Container Insights, estimate costs and understand how to control data ingestion and its costs. For detailed guidance, see [Estimating costs to monitor your AKS cluster](#).

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Ketan Chawda](#) | Senior Customer Engineer
- [Paolo Salvatori](#) | Principal Service Engineer

- [Laura Nicolas](#) | Senior Software Engineer

Other contributors:

- [Chad Kittel](#) | Principal Software Engineer
- [Ed Price](#) | Senior Content Program Manager
- [Theano Petersen](#) | Technical Writer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- [AKS for Amazon EKS professionals](#)
- [Kubernetes identity and access management](#)
- [Secure network access to Kubernetes](#)
- [Storage options for a Kubernetes cluster](#)
- [Cost management for Kubernetes](#)
- [Kubernetes node and node pool management](#)
- [Cluster governance](#)

Related resources

- [Use Azure Monitor Private Link Scope](#)
- [Monitor Azure Kubernetes Service \(AKS\) with Azure Monitor](#)
- [Monitoring AKS data reference](#)
- [Container Insights overview](#)
- [Enable Container Insights](#)
- [AKS resource logs](#)
- [Configure scraping of Prometheus metrics with Container Insights](#)
- [How to query logs from Container Insights](#)
- [Azure Monitor data source for Grafana](#)
- [Monitor and back up Azure resources](#)
- [Instrument solutions to support monitoring and logging](#)
- [Design a solution to log and monitor Azure resources](#)
- [Monitor the usage, performance, and availability of resources with Azure Monitor](#)

Secure network access to Kubernetes

Azure Bastion Azure DNS Azure Kubernetes Service (AKS) Azure Private Link Azure Virtual Network

This article compares networking modes for Azure Kubernetes Service (AKS) and Amazon Elastic Kubernetes Service (Amazon EKS). The article describes how to improve connection security to the managed API server of an AKS cluster, and the different options to restrict public network access.

ⓘ Note

This article is part of a [series of articles](#) that helps professionals who are familiar with [Amazon EKS](#) to understand [AKS](#).

Amazon EKS networking modes

With [Amazon Virtual Private Cloud \(Amazon VPC\)](#), you can launch Amazon Web Services (AWS) resources into a virtual network composed of public and private [subnets](#), or ranges of IP addresses in the VPC. A public subnet hosts resources that must be connected to the internet, and a private subnet hosts resources that aren't connected to the public internet. Amazon EKS can provision managed node groups in both public and private subnets.

Endpoint access control lets you configure whether the API Server endpoint is reachable from the public internet or through the VPC. EKS provides several ways to [control access to the cluster endpoint](#). You can enable the default public endpoint, a private endpoint, or both endpoints simultaneously. When you enable the public endpoint, you can add Classless Inter-Domain Routing (CIDR) restrictions to limit the client IP addresses that can connect to the public endpoint.

How Amazon EKS nodes connect to the managed Kubernetes control plane is determined by which endpoint setting is configured for the cluster. You can change the endpoint settings anytime through the Amazon EKS console or the API. For more information, see [Amazon EKS cluster endpoint access control](#).

Public endpoint only

Exposing the control plane via a public endpoint is the default mode for new Amazon EKS clusters. When only the public endpoint for the cluster is enabled, Kubernetes API requests that originate from within the Amazon VPC, such as worker node to control plane communication, leave the VPC but don't leave Amazon's network. For nodes to connect to the control plane, they must use a public IP address and a route to an internet gateway, or a route to a network address translation (NAT) gateway where they can use the NAT gateway's public IP address.

Public and private endpoints

When both the public and private endpoints are enabled, Kubernetes API requests from within the VPC communicate to the control plane via the Amazon EKS-managed Elastic Network Interfaces (ENIs) in the VPC. The cluster API server is accessible from the internet.

Private endpoint only

When only the private endpoint is enabled, all traffic to the cluster API server, such as `kubectl` or `helm` commands, must come from within the cluster's VPC or a connected network. Public access to the API server from the internet is disabled. You can implement this access mode by using [AWS Virtual Private Network \(AWS VPN\)](#) or [AWS DirectConnect](#) to the VPC. To restrict access to the endpoint without AWS VPN or DirectConnect, you can add CIDR restrictions to the public endpoint to limit connections without setting up more networking.

For more information on connectivity options, see [Accessing a Private Only API Server](#).

AKS network access to the API server

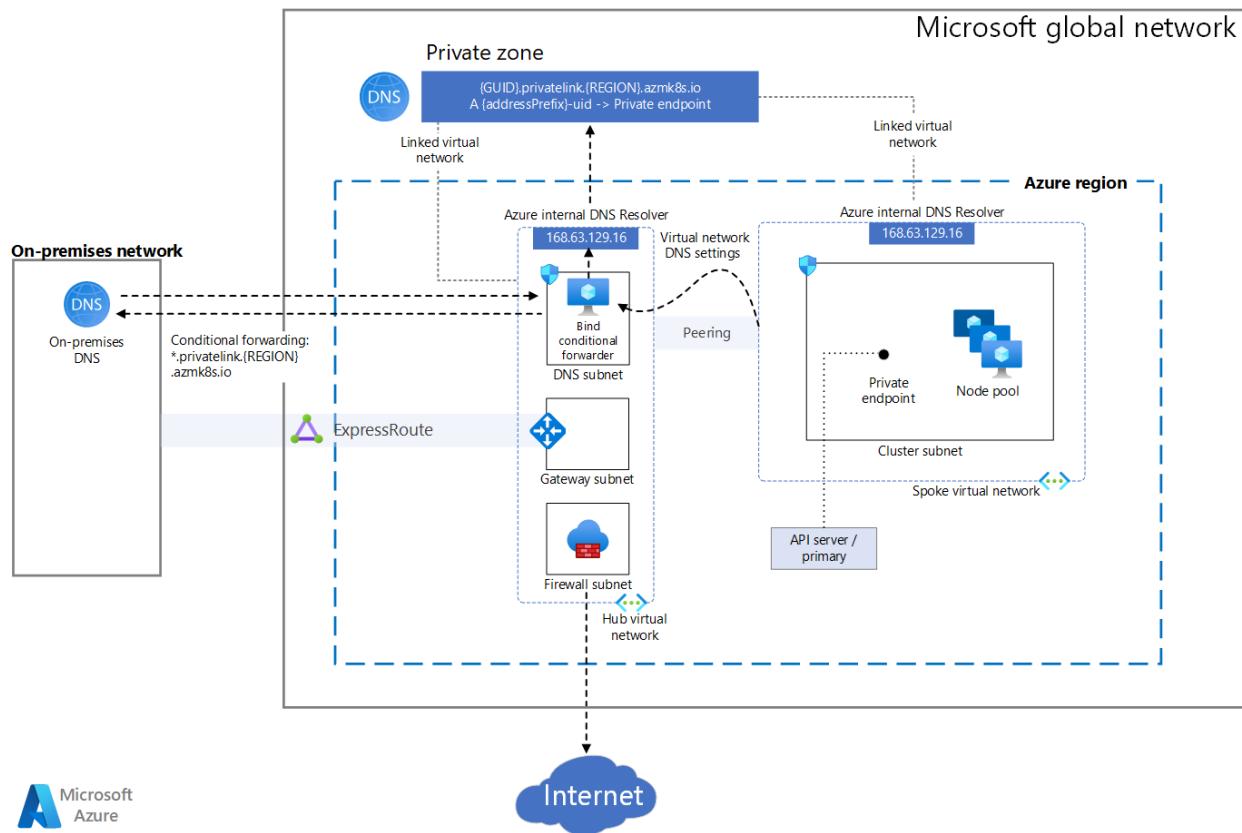
There are two options to secure network access to the Kubernetes API in AKS, a private AKS cluster or authorized IP ranges.

Private AKS cluster

An [AKS private cluster](#) ensures that network traffic between the API server and the node pools remains within the virtual network. In a private AKS cluster, the control plane or API server has an internal IP address that's only accessible via an Azure [private endpoint](#) located in the same virtual network. Any virtual machine (VM) in the same virtual network can privately communicate with the control plane via the private endpoint. The

control plane or API server is hosted in the Azure-managed subscription, while the AKS cluster and its node pools are in the customer's subscription.

The following diagram illustrates a private cluster configuration.



Download a [Visio file](#) of this architecture.

To provision a private AKS cluster, the AKS resource provider creates a private fully qualified domain name (FQDN) for the node resource group in a private DNS zone. Optionally, AKS can also create a public FQDN with a corresponding address (A) record in the Azure public DNS zone. The agent nodes use the A record in the private DNS zone to resolve the private endpoint IP address for communication to the API server.

The AKS resource provider can create the private DNS zone in the node resource group, or you can create the private DNS zone and pass its resource ID to the provisioning system. You can create a private cluster when you use [Terraform with Azure](#), [Bicep](#), [ARM templates](#), [Azure CLI](#), [Azure PowerShell module](#), or [Azure REST API](#) to create the cluster.

You can enable a public FQDN for the API server during provisioning or by using the [az aks update](#) command with the --enable-public-fqdn parameter on existing clusters. If you enable the public FQDN, any VM that accesses the server, such as an Azure DevOps self-hosted agent or a GitHub Actions self-hosted runner, must be in the same virtual network that hosts the cluster, or in a network connected via [virtual network peering](#) or [site-to-site VPN](#).

For a private AKS cluster, you disable the public FQDN of the API server. To communicate with the private control plane, a VM must be in the same virtual network, or in a peered virtual network with a [virtual network link](#) to the [private DNS zone](#). The `A` record in the private DNS zone resolves the FQDN of the API server to the private endpoint IP address that communicates with the underlying control plane. For more information, see [Create a private Azure Kubernetes Service cluster](#).

Private cluster deployment options

The AKS resource provider exposes the following parameters to customize private AKS cluster deployment:

- `authorizedIpRanges` (string) specifies allowed IP ranges in CIDR format.
- `disableRunCommand` (boolean) specifies whether or not to disable the `run` command for the cluster.
- `enablePrivateCluster` (boolean) specifies whether or not to create the cluster as private.
- `enablePrivateClusterPublicFQDN` (boolean) specifies whether or not to create another, public FQDN for the private cluster.
- `privateDnsZone` (string) specifies a private DNS zone in the node resource group. If you don't specify a value, the resource provider creates the zone. You can specify the following values:
 - `System` is the default value.
 - `None` defaults to public DNS, so AKS doesn't create a private DNS zone.
 - `<Your own private DNS zone resource ID>` uses a private DNS zone you create in the format `privatelink.<region>.azmk8s.io` or `<subzone>.privatelink.<region>.azmk8s.io`.

The following table shows the DNS configuration options for deploying a private AKS cluster:

[Expand table](#)

Private DNS zone options	<code>enablePrivateClusterPublicFQDN: true</code>	<code>enablePrivateClusterPublicFQDN: false</code>
<code>System</code>	Agent nodes, and any other VMs in the AKS cluster virtual network or any virtual network connected to	Agent nodes, and any other VMs in the AKS cluster virtual network or any virtual network connected to

Private DNS zone options	enablePrivateClusterPublicFQDN: true	enablePrivateClusterPublicFQDN: false
	<p>the private DNS zone, use the private DNS zone <code>A</code> record to resolve the private IP address of the private endpoint.</p> <p>Any other VM uses the public FQDN of the API server.</p>	<p>the private DNS zone, use the private DNS zone <code>A</code> record to resolve the private IP address of the private endpoint.</p> <p>No public API server FQDN is available.</p>
None	All the VMs, including agent nodes, use the public FQDN of the API server available via an <code>A</code> record in an Azure-managed public DNS zone.	Wrong configuration. The private AKS cluster needs at least a public or a private DNS zone for the name resolution of the API server.
<Your own private DNS zone resource ID>	<p>Agent nodes, and any other VMs in the AKS cluster virtual network or any virtual network connected to the private DNS zone, use the private DNS zone <code>A</code> record to resolve the private IP address of the private endpoint.</p> <p>Any other VMs use the public FQDN of the API server.</p>	<p>Agent nodes, and any other VMs in the AKS cluster virtual network or any virtual network connected to the private DNS zone, use the private DNS zone <code>A</code> record to resolve the private IP address of the private endpoint.</p> <p>No public API server FQDN is available.</p>

Private cluster connectivity and management

There are several options for establishing network connectivity to the private cluster.

- Create VMs in the same virtual network as the AKS cluster.
- Use VMs in a separate virtual network and set up [virtual network peering](#) with the AKS cluster virtual network.
- Use an [Azure ExpressRoute or VPN](#) connection.
- Use the Azure CLI command `az aks command invoke` to run `kubectl` and `helm` commands on the private cluster without directly connecting to the cluster.
- Use an [Azure Private Endpoint](#) connection.

You can manage a private AKS cluster by using the [kubectl](#) command-line tool from a management VM in the same virtual network or a peered virtual network.

You can use [Azure Bastion](#) in the same virtual network or a peered virtual network to connect to a jumpbox management VM. Azure Bastion is a fully managed platform as a service (PaaS) that lets you connect to a VM by using your browser and the Azure portal. Azure Bastion provides secure and seamless remote desktop protocol (RDP) or secure shell (SSH) VM connectivity over transport layer security (TLS) directly from the Azure portal. When VMs connect via Azure Bastion, they don't need a public IP address, agent, or special client software.

You can also use [az aks command invoke](#) to run `kubectl` or `helm` commands on your private AKS cluster without having to connect to a jumpbox VM.

Authorized IP ranges

The second option to improve cluster security and minimize attacks to the API server is to use [authorized IP ranges](#). Authorized IPs restrict access to the control plane of a public AKS cluster to a known list of IP addresses and CIDRs. When you use this option, the API server is still publicly exposed, but access is limited. For more information, see [Secure access to the API server using authorized IP address ranges in Azure Kubernetes Service \(AKS\)](#).

The following `az aks update` Azure CLI command authorizes IP ranges:

Azure CLI

```
az aks update \
  --resource-group myResourceGroup \
  --name myAKSCluster \
  --api-server-authorized-ip-ranges 73.140.245.0/24
```

AKS connectivity considerations

- An AKS private cluster provides higher security and isolation than authorized IPs. However, you can't convert an existing public AKS cluster into a private cluster. You can enable authorized IPs for any existing AKS cluster.
- You can't apply authorized IP ranges to a private API server endpoint. Authorized IPs apply only to the public API server.
- Private clusters don't support Azure DevOps-hosted agents. Consider using self-hosted agents.

- To enable Azure Container Registry to work with a private AKS cluster, set up a private link for the container registry in the cluster virtual network. Or, set up peering between the Container Registry virtual network and the private cluster's virtual network.
- Azure Private Link service limitations apply to private clusters.
- If you delete or modify the private endpoint in the customer subnet of a private cluster, the cluster will stop functioning.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Paolo Salvatori](#) | Principal Service Engineer
- [Martin Gjoshevski](#) | Senior Service Engineer
- [Laura Nicolas](#) | Senior Software Engineer

Other contributors:

- [Chad Kittel](#) | Principal Software Engineer
- [Ed Price](#) | Senior Content Program Manager
- [Theano Petersen](#) | Technical Writer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- [AKS for Amazon EKS professionals](#)
- [Kubernetes identity and access management](#)
- [Kubernetes monitoring and logging](#)
- [Storage options for a Kubernetes cluster](#)
- [Cost management for Kubernetes](#)
- [Kubernetes node and node pool management](#)
- [Cluster governance](#)

Related resources

The following references provide links to documentation and automation samples to deploy AKS clusters with a secured API:

- [Create a Private AKS cluster with a Public DNS Zone ↗](#)
- [Create a private Azure Kubernetes Service cluster using Terraform and Azure DevOps ↗](#)
- [Create a public or private Azure Kubernetes Service cluster with Azure NAT Gateway and Azure Application Gateway ↗](#)
- [Use Private Endpoints with a Private AKS Cluster ↗](#)
- [Introduction to Azure Private Link](#)
- [Introduction to Secure Network Infrastructure with Azure network security](#)

Storage options for a Kubernetes cluster

Article • 04/10/2023

This article compares the storage capabilities of Amazon Elastic Kubernetes Service (Amazon EKS) and Azure Kubernetes Service (AKS) and describes the options to store workload data on AKS.

ⓘ Note

This article is part of a [series of articles](#) that helps professionals who are familiar with [Amazon EKS](#) to understand [AKS](#).

Amazon EKS storage options

In Amazon EKS, after Kubernetes version 1.11, the cluster has a default [StorageClass](#) called `gp2` for [persistent volume claims](#). Administrators can add drivers to define more storage classes, such as:

- Amazon EBS CSI driver as an Amazon EKS add-on
- Amazon EBS CSI self-managed add-on
- Amazon EFS CSI driver
- Amazon FSx for Lustre CSI driver
- Amazon FSx for NetApp ONTAP CSI driver

By adding drivers and storage classes, you can use storage services such as:

- Amazon Elastic Block Store (Amazon EBS), a block-level storage solution used with Amazon Elastic Compute Cloud (EC2) instances to store persistent data. This service is similar to Azure Disk Storage, which has several SKUs like Standard SSD, Premium SSD, or Ultra Disk, depending on needed performance.
- Amazon Elastic File System (Amazon EFS), which provides Network File System (NFS) access to external file systems that can be shared across instances. The equivalent Azure solution is Azure Files and Azure Files Premium with both Server Message Block (SMB) 3.0 and NFS access.
- Lustre, an open-source file system commonly used in high performance computing (HPC). In Azure, you can use Ultra Disks or Azure HPC Cache for workloads where speed matters, such as machine learning and HPC.

- NetApp ONTAP, fully managed ONTAP shared storage in Amazon Web Services (AWS). Azure NetApp Files is a similar Azure file storage service built on NetApp technology.

AKS storage options

Each AKS cluster includes the following [pre-created storage classes](#) by default:

- The default storage class, `managed-csi`, uses Disk Storage Standard SSD. Standard SSD is a cost-effective storage option optimized for workloads that need consistent performance at lower input-output operations per second (IOPS).
- The `managed-csi-premium` class uses Disk Storage Premium SSD managed disks.
- The `azurefile-csi` class uses Azure Files to provide concurrent shared access to the same storage volume, using SMB or NFS.
- The `azurefile-csi-premium` class uses Azure Files Premium for file shares with IOPS-intensive workloads. Azure Files Premium provides low latency and high throughput backed by SSD storage.

You can extend these options by adding other storage classes and integrating with other available storage solutions, such as:

- Ultra Disk Storage
- Azure NetApp Files
- HPC Cache
- NFS server
- Third-party storage solutions

Azure Disk Storage

By default, an AKS cluster comes with pre-created `managed-csi` and `managed-csi-premium` storage classes that use [Disk Storage](#). Similar to Amazon EBS, these classes create a managed disk or block device that's attached to the node for pod access.

The Disk Storage storage classes allow both [static](#) and [dynamic](#) volume provisioning. Reclaim policy ensures that the disk is deleted with the persistent volume. You can expand the disk by editing the persistent volume claim.

These storage classes use Azure managed disks with [locally redundant storage \(LRS\)](#). LRS means that the data has three synchronous copies within a single physical location in an Azure primary region. LRS is the least expensive replication option, but doesn't offer protection against a datacenter failure. To mitigate this risk, take regular backups

or snapshots of Disk Storage data by using solutions like [Velero](#) or [Azure Backup](#) that can use built-in snapshot technologies.

Both storage classes are backed by managed disks, and both use solid-state disk (SSD) drives. It's important to understand the differences between Standard and Premium disks:

- Standard disks are priced based on size and storage transactions.
- Premium disks charge only by size, which can make them cheaper for workloads that require a high number of transactions.
- Premium SSDs provide a higher max throughput and IOPS, as [shown in this comparison](#).
- Premium storage is recommended for most production and development workloads.

If you use Azure managed disks as your primary storage class, consider the virtual machine (VM) SKU that you choose for your Kubernetes cluster. Azure VMs limit the number of disks that you can attach to them, and the limit varies with the VM size. Also, since Azure disks are mounted as `ReadWriteOnce`, they're available only to a single pod.

Ultra Disk Storage

Ultra Disk Storage is an Azure managed disk tier that offers high throughput, high IOPS, and consistent low latency disk storage for Azure VMs. Ultra Disk Storage is intended for workloads that are data and transaction heavy. Like other Disk Storage SKUs, and Amazon EBS, Ultra Disk Storage mounts one pod at a time and doesn't provide concurrent access.

Use the flag `--enable-ultra-ssd` to [enable Ultra Disk Storage on your AKS cluster](#).

If you choose Ultra Disk Storage, be aware of its [limitations](#), and make sure to select a compatible VM size. Ultra Disk Storage is available with locally redundant storage (LRS) replication.

Azure Files

Disk Storage can't provide concurrent access to a volume, but you can use [Azure Files](#) to connect by using the SMB protocol, and then mount a shared volume that's backed by Azure Storage. This process provides a network attached storage that's similar to Amazon EFS. As with Disk Storage, there are two options:

- Azure Files Standard storage is backed by regular hard disk drives (HDDs).

- Azure Files Premium storage backs the file share with high-performance SSD drives. The minimum file share size for Premium is 100 GB.

Azure Files has the following storage account replication options to protect your data in case of failure:

- Standard_LRS with [LRS](#)
- Standard_GRS with [geo-redundant storage \(GRS\)](#)
- Standard_ZRS with [zone-redundant storage \(ZRS\)](#)
- Standard_RAGRS with [read-access geo-redundant storage \(RA-GRS\)](#)
- Premium_LRS premium LRS
- Premium_ZRS premium ZRS

To optimize costs for Azure Files, purchase [Azure Files capacity reservations](#).

Azure NetApp Files

Like AWS NetApp ONTAP, Azure NetApp Files is an enterprise-class, high-performance, metered file storage service. Azure NetApp Files is fully managed in Azure using NetApp solutions. Like Azure Files, Azure NetApp Files lets multiple pods mount a volume. You can use [Astra Trident](#), an open-source dynamic storage orchestrator for Kubernetes, to [configure your AKS cluster to use Azure NetApp Files](#).

Be aware of the [Resource limits for Azure NetApp Files](#). The minimum size of a capacity pool for Azure NetApp Files is 4 TiB. Azure NetApp Files charges by provisioned size rather than used capacity.

Azure HPC Cache

[Azure HPC Cache](#) speeds access to your data for HPC tasks, with all the scalability of cloud solutions. If you choose this storage solution, make sure to deploy your AKS cluster in a [region that supports Azure HPC cache](#).

NFS server

The best option for shared NFS access is to use Azure Files or Azure NetApp Files. You can also [create an NFS Server on an Azure VM](#) that exports volumes.

Be aware that this option only supports static provisioning. You must provision the NFS shares manually on the server, and can't do so from AKS automatically.

This solution is based on infrastructure as a service (IaaS) rather than platform as a service (PaaS). You're responsible for managing the NFS server, including OS updates, high availability, backups, disaster recovery, and scalability.

Third-party solutions

Like Amazon EKS, AKS is a Kubernetes implementation, and you can integrate third-party Kubernetes storage solutions. Here are some examples of third-party storage solutions for Kubernetes:

- [Rook](#) turns distributed storage systems into self-managing storage services by automating storage administrator tasks. Rook delivers its services via a Kubernetes operator for each storage provider.
- [GlusterFS](#) is a free and open-source scalable network filesystem that uses common off-the-shelf hardware to create large, distributed storage solutions for data-heavy and bandwidth-intensive tasks.
- [Ceph](#) provides a reliable and scalable unified storage service with object, block, and file interfaces from a single cluster built from commodity hardware components.
- [MinIO](#) multicloud object storage lets enterprises build AWS S3-compatible data infrastructure on any cloud, providing a consistent, portable interface to your data and applications.
- [Portworx](#) is an end-to-end storage and data management solution for Kubernetes projects and container-based initiatives. Portworx offers container-granular storage, disaster recovery, data security, and multicloud migrations.
- [Quobyte](#) provides high-performance file and object storage you can deploy on any server or cloud to scale performance, manage large amounts of data, and simplify administration.
- [Ondat](#) delivers a consistent storage layer across any platform. You can run a database or any persistent workload in a Kubernetes environment without having to manage the storage layer.

Kubernetes storage considerations

Consider the following factors when you choose a storage solution for Amazon EKS or AKS.

Storage class access modes

In Kubernetes version 1.21 and newer, AKS and Amazon EKS storage classes use [Container Storage Interface \(CSI\) drivers](#) only and by default.

Different services support storage classes that have different access modes.

Service	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
Azure Disks	X		
Azure Files	X	X	X
Azure NetApp Files	X	X	X
NFS server	X	X	X
Azure HPC Cache	X	X	X

Dynamic vs static provisioning

[Dynamically provision volumes](#) to reduce the management overhead of statically creating persistent volumes. Set a correct reclaim policy to avoid having unused disks when you delete pods.

Backup

Choose a tool to back up persistent data. The tool should match your storage type, such as snapshots, [Azure Backup](#), [Velero](#) or [Kasten](#).

Cost optimization

To optimize Azure Storage costs, use Azure Reservations. Make sure to [check which services support Azure Reservations](#). Also see [Cost management for a Kubernetes cluster](#).

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Laura Nicolas](#) | Senior Software Engineer
- [Paolo Salvatori](#) | Principal System Engineer

Other contributors:

- [Chad Kittel](#) | Principal Software Engineer
- [Ed Price](#) | Senior Content Program Manager
- [Theano Petersen](#) | Technical Writer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- AKS for Amazon EKS professionals
- Kubernetes identity and access management
- Kubernetes monitoring and logging
- Secure network access to Kubernetes
- Cost management for Kubernetes
- Kubernetes node and node pool management
- Cluster governance

Related resources

- [Azure Storage services](#)
- [Store data in Azure](#)
- [Configure AKS storage](#)
- [Introduction to Azure NetApp Files](#)

Cost management for Kubernetes

Azure Cost Management

Azure Kubernetes Service (AKS)

Azure Managed Disks

Azure Storage

Azure Virtual Machines

This guide explains how pricing and cost management work in Azure Kubernetes Service (AKS) compared to Amazon Elastic Kubernetes Service (Amazon EKS). The article describes how to optimize costs and implement cost governance solutions for your AKS cluster.

ⓘ Note

This article is part of a [series of articles](#) that helps professionals who are familiar with [Amazon EKS](#) to understand [AKS](#).

Amazon EKS cost basics

In [Amazon EKS](#), you pay a fixed price per hour for each Amazon EKS cluster. You also pay for the networking, operations tools, and storage that the cluster uses.

Amazon EKS worker nodes are standard Amazon EC2 instances, so they incur regular Amazon EC2 prices. You also pay for other Amazon Web Services (AWS) resources that you provision to run your Kubernetes worker nodes.

There are no extra costs to use Amazon EKS [managed node groups](#). You pay only for the AWS resources you provision, including Amazon EC2 instances, Amazon EBS volumes, Amazon EKS cluster hours, and any other AWS infrastructure.

When creating a managed node group, you can choose to use the [On-Demand or Spot Instances capacity type](#) to manage the cost of agent nodes. Amazon EKS deploys a managed node group with an [Amazon EC2 Auto Scaling group](#) that contains either all On-Demand or all Spot Instances.

With On-Demand Instances, you pay for compute capacity by the second, with no long-term commitments. Amazon EC2 Spot Instances are spare Amazon EC2 capacity that offers discounts compared to On-Demand prices.

- Amazon EC2 Spot Instances can be interrupted with a two-minute interruption notice when EC2 needs the capacity back.

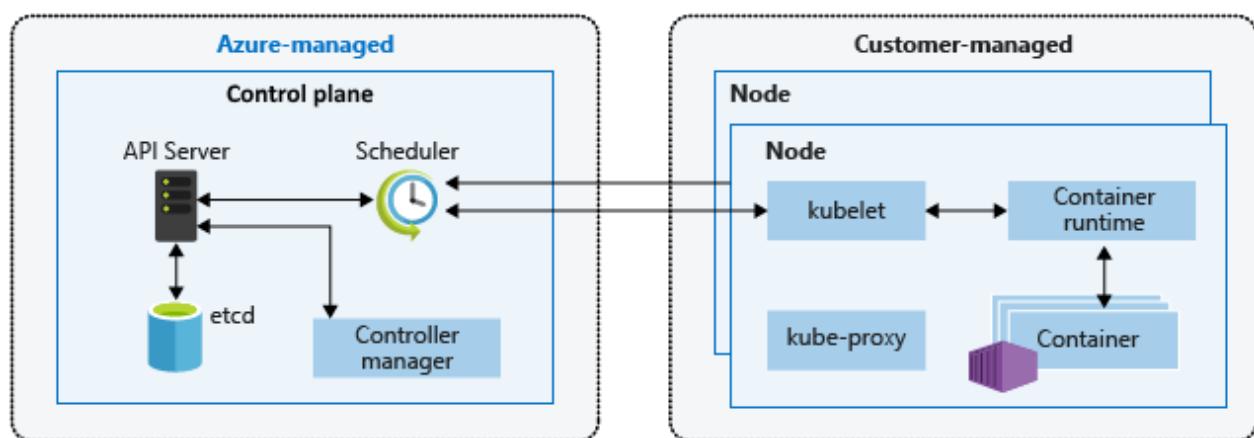
- Amazon provides Spot Fleet, a method to automate groups of On-Demand and Spot Instances, and Spot Instance Advisor to help predict which region or Availability Zone might provide minimal disruption.
- AWS Spot Instance prices vary. AWS sets the price depending on long-term supply and demand trends for Spot Instance capacity, and you pay the price in effect for the time period the instance is up.

AKS cost basics

Kubernetes architecture is based on two layers, the control plane and one or more nodes or node pools. The AKS pricing model is based on the two Kubernetes architecture layers.

- The [control plane](#) provides [core Kubernetes services](#), such as the API server and `etcd`, and application workload orchestration. The Azure platform manages the AKS control plane, and for the AKS free tier, the control plane has [no cost](#).
- The [nodes](#), also called *agent nodes* or *worker nodes*, host Kubernetes workloads and applications. In AKS, customers fully manage and pay all costs for the agent nodes.

The following diagram shows the relationship between the control plane and nodes in an AKS Kubernetes architecture.



Control plane

Azure automatically provisions and configures the control plane layer when you create an AKS cluster. For the AKS Free tier, the control plane is free.

For a higher control plane service-level agreement (SLA), you can create an AKS cluster in the [Standard tier](#). Uptime SLA is included by default in the Standard tier and is

enabled per cluster. The pricing is \$0.10 per cluster per hour. For more information, see [AKS pricing details](#).

Clusters in the Standard tier have more control plane resources, such as the number of API server instances, Etcd resource limits, [scalability up to 5,000 nodes](#), and the existing financially-backed Uptime SLA support. AKS uses main node replicas across update and fault domains to meet availability requirements.

It's best to use the Standard tier in production workloads to provide higher control plane component availability. Free tier clusters have fewer replicas and limited control plane resources and aren't recommended for production workloads.

Nodes

In AKS, you create agent or worker nodes in one or more node pools, which can use many Azure core capabilities within the Kubernetes environment. AKS charges only for the nodes attached to the AKS cluster.

AKS nodes use several Azure infrastructure resources, including virtual machine scale sets, virtual networks, and managed disks. For example, you can use most Azure virtual machine (VM) types directly within AKS. You can use [Azure Reservations](#) and [Azure savings plan for compute](#) to get significant discounts on these resources.

AKS cluster pricing is based on the class, number, and size of the VMs in the node pools. VM cost depends on size, CPU type, number of vCPUs, memory, family, and storage type available, such as high-performance SSD or standard HDD. For more information, see [Virtual Machine Series](#). Plan node size according to application requirements, number of nodes, and cluster scalability needs.

For more information about agent nodes and node pools, see the [Node pools](#) article in this series, and [Create and manage multiple node pools for a cluster in Azure Kubernetes Service \(AKS\)](#).

AKS cluster deployment

Each AKS deployment spans two Azure resource groups.

- You create the first resource group, which contains only the Kubernetes service resource and has no costs associated with it.
- The AKS resource provider automatically creates the second or node resource group during deployment. The default name for this resource group is `MC_<resourcegroupname>_<clustername>_<location>`, but you can specify another

name. For more information, see [Provide my own name for the AKS node resource group](#).

The node resource group contains all the cluster infrastructure resources and is the one that shows charges to your subscription. The resources include the Kubernetes node VMs, virtual networking, storage, and other services. AKS automatically deletes the node resource group when the cluster is deleted, so you should use it only for resources that share the cluster's lifecycle.

Compute costs

You pay for Azure VMs according to their size and usage. For information about how Azure compute compares to AWS, see [Compute services on Azure and AWS](#).

Generally, the bigger the VM size you select for a node pool, the higher the hourly cost for the agent nodes. The more specialized the VM series you use for the node pool, for example graphics processing unit (GPU)-enabled or memory-optimized, the more expensive the pool.

When investigating Azure VM pricing, be aware of the following points:

- Pricing differs per region, and not all services and VM sizes are available in every region.
- There are multiple VM families, optimized for different types of workloads.
- Managed disks used as OS drives are charged separately, and you must add their cost to your estimates. Managed disk size depends on the class, such as Standard HDDs, Standard SSDs, Premium SSDs, or Ultra SSDs. Input-output operations per second (IOPS) and throughput in MB/sec depend on size and class. [Ephemeral OS disks](#) are free and are included in the VM price.
- Data disks, including those created with persistent volume claims, are optional and are charged individually based on their class, such as Standard HDDs, Standard SSDs, Premium SSDs, and Ultra SSDs. You must explicitly add data disks to cost estimations. The number of allowed data disks, temporary storage SSDs, IOPS, and throughput in MB/sec depend on VM size and class.
- The more time that agent nodes are up and running, the higher the total cluster cost. Development environments don't usually need to run continuously.
- Network interfaces (NICs) are free.

Storage costs

The Container Storage Interface (CSI) is a standard for exposing block and file storage systems to containerized workloads on Kubernetes. By adopting and using CSI, AKS can write, deploy, and iterate plug-ins that expose Kubernetes storage systems without touching the core Kubernetes code or waiting for its release cycles.

If you run workloads that use CSI persistent volumes on your AKS cluster, consider the associated cost of the storage your applications provision and use. CSI storage drivers on AKS provide native support for the following storage options:

- [Azure Disks](#) creates Kubernetes data disk resources. Disks can use Azure Premium Storage, backed by high-performance SSDs, or Azure Standard Storage, backed by regular HDDs or Standard SSDs. Most production and development workloads use Premium Storage. Azure disks are mounted as `ReadWriteOnce`, which makes them available to only one AKS node. For storage volumes that multiple pods can access simultaneously, use Azure Files. For cost information, see [Managed Disks pricing](#).
- [Azure Files](#) mounts server messaging block (SMB) 3.0/3.1 file shares backed by an Azure Storage account to AKS pods. You can share data across multiple nodes and pods. Azure Files can use Standard storage backed by regular HDDs, or Premium storage backed by high-performance SSDs. Azure Files uses an Azure Storage account, and accrues charges based on the following factors:
 - Service: Blob, File, Queue, Table or unmanaged disks
 - Storage account type: GPv1, GPv2, Blob, or Premium Blob
 - Resiliency: Locally redundant storage (LRS), zone-redundant storage (ZRS), geo-redundant storage (GRS), or read-access geo-redundant storage (RA-GRS)
 - Access tier: Hot, cool, or archive
 - Operations and data transfers
 - Used capacity in GB
- [Azure NetApp Files](#) is available in several SKU tiers and requires a minimum provisioned capacity of 4 TiB, with 1 TiB increments. Azure NetApp Files charges are based on the following factors:
 - SKU
 - Resiliency: LRS, ZRS, or GRS
 - Size or capacity provisioned, not capacity used
 - Operations and data transfer
 - Backups and restores

Networking costs

Several Azure networking services can provide access to your applications that run in AKS:

- [Azure Load Balancer](#). By default, Load Balancer uses Standard SKU. Load Balancer charges are based on:
 - Rules: The number of configured load-balancing and outbound rules. Inbound network address translation (NAT) rules don't count in the total number of rules.
 - Data processed: The amount of data processed inbound and outbound, independent of rules. There's no hourly charge for Standard Load Balancer with no rules configured.
- [Azure Application Gateway](#). AKS often uses Application Gateway through [Application Gateway Ingress Controller](#), or by fronting a different ingress controller with manually managed Application Gateway. Application Gateway supports gateway routing, transport-layer security (TLS) termination, and Web Application Firewall (WAF) functionality. Application Gateway charges are based on:
 - Fixed price set by hour or partial hour.
 - Capacity unit price, an added consumption-based cost. Each capacity unit has at most one compute unit, 2,500 persistent connections, and 2.22-Mbps throughput.
- [Public IP addresses](#) have an associated cost that depends on:
 - Reserved vs. dynamic association.
 - Basic vs. secured and zone-redundant Standard tier.

Scale-out costs

There are multiple options for scaling an AKS cluster to add extra capacity to node pools:

- On demand, you can manually update the number of VMs that are part of a node pool, or add more node pools.
- The AKS [cluster autoscaler](#) watches for pods that can't be scheduled on nodes because of resource constraints, and automatically increases the number of nodes.
- AKS supports running containers on [Azure Container Instances](#) by using the [virtual kubelet](#) implementation. An AKS virtual node provisions Container Instances pods that start in seconds, letting AKS run with just enough capacity for an average workload. As the AKS cluster runs out of capacity, you can scale out more Container Instances pods without managing any additional servers. You can combine this approach with the cluster autoscaler and manual scaling.

If you use on-demand scaling or the cluster autoscaler, account for the added VMs.

Container Instances charges are based on the following factors:

- Usage-based metrics billing per container group
- Collection vCPU and memory
- Single container use or multiple container sharing
- Use of co-scheduled containers that share network and node lifecycle
- Usage duration calculated from image pull start or restart until stop
- Added charge for Windows container groups

Upgrade costs

Part of the AKS cluster lifecycle involves periodic upgrades to the latest Kubernetes version. It's important to apply the latest security releases and get the latest features. You can upgrade AKS clusters and single node pools manually or automatically. For more information, see [Upgrade an Azure Kubernetes Service \(AKS\) cluster](#).

By default, AKS configures upgrades to surge with one extra node. A default value of 1 for the `max-surge` setting minimizes workload disruption by creating an extra node to replace older-versioned nodes before cordoning or draining existing applications. You can customize the `max-surge` value per node pool to allow for a tradeoff between upgrade speed and upgrade disruption. Increasing the `max-surge` value completes the upgrade process faster, but a large value for `max-surge` might cause disruptions during the upgrade process and incur added costs for extra VMs.

Other costs

Depending on usage and requirements, AKS clusters can incur the following added costs:

- [Azure Container Registry](#) costs depending on [Basic, Standard, or Premium SKU](#), image builds, and storage used. Deploy Container Registry on the same region as the cluster to avoid added data transfer charges. Use replication if needed, and reduce image sizes as much as possible to reduce storage costs and deployment times.
- Outbound [data transfers](#) from Azure, as well as inter-region traffic.
- Other storage or platform as a service (PaaS) services such as databases.
- Global networking services such as [Azure Traffic Manager](#) or [Azure Front Door](#) that route traffic to the public endpoints of AKS workloads.

- Firewall and protection services like [Azure Firewall](#) that inspect and allow or block traffic to and from AKS clusters.
- Monitoring and logging services such as [Azure Monitor Container Insights](#), [Azure Monitor Application Insights](#), and [Microsoft Defender for Cloud](#). For more information, see [Understand monitoring costs for Container Insights](#).
- Costs associated with DevOps tools like [Azure DevOps Services](#) or [GitHub](#).

Cost optimization

The following recommendations help you optimize your AKS cluster costs:

- Review the [Cost optimization](#) section of the Azure Well-Architected Framework for AKS.
- For multitenant solutions, physical isolation is more costly and adds management overhead. Logical isolation requires more Kubernetes experience and increases the surface area for changes and security threats, but shares the costs.
- [Azure Reservations](#) can help you save money by committing to one-year or three-year plans for several products, such as the VMs in your AKS cluster. You get discounts by reserving capacity. Use Azure Reservations for [Storage](#) and [Compute](#) to reduce the cost of agent nodes.

Reservations can reduce your resource costs by up to 72% from pay-as-you-go prices, and don't affect the runtime state of your resources. After you purchase a reservation, the discount automatically applies to matching resources. You can purchase reservations from the Azure portal, or by using Azure REST APIs, PowerShell, or Azure CLI. If you use operational tools that rely on [Log Analytics workspaces](#), consider using Reservations for this storage also.

- Add one or more spot node pools to your AKS cluster. A spot node pool is a node pool backed by [Azure Spot Virtual Machine Scale Sets](#). Using spot VMs for your AKS cluster nodes takes advantage of unused Azure capacity at significant cost savings. The amount of available unused capacity varies based on several factors, including node size, region, and time of day. Azure allocates the spot nodes if there's capacity available, but there's no SLA for spot nodes. A spot scale set that backs the spot node pool is deployed in a single fault domain, and offers no high availability guarantees. When Azure needs the capacity back, the Azure infrastructure evicts the spot nodes.

When you create a spot node pool, you can define the maximum price to pay per hour and enable the cluster autoscaler, which is recommended for spot node pools. The cluster autoscaler scales out and scales in the number of nodes in the node pool based on the running workloads. For spot node pools, the cluster autoscaler scales out the number of nodes after an eviction if the nodes are still needed. For more information, see [Add a spot node pool to an Azure Kubernetes Service \(AKS\) cluster](#).

- Choose the right [VM size](#) for your AKS cluster node pools based on your workloads' CPU and memory needs. Azure offers many different VM instance types that match a wide range of use cases, with different combinations of CPU, memory, storage, and networking capacity. Every type comes in one or more sizes, so you can easily scale your resources.

You can now deploy and manage containerized applications with AKS running on Ampere Altra ARM-based processors. For more information, see [Azure Virtual Machines with Ampere Altra ARM-based processors](#).

- Create multiple node pools with different VM sizes for special purposes and workloads. Use Kubernetes [taints and tolerations](#) and [node labels](#) to place resource-intensive applications on specific node pools to avoid noisy neighbor issues. Keep these node resources available for workloads that require them, and don't schedule other workloads on these nodes. Using different VM sizes for different node pools can also optimize costs. For more information, see [Use multiple node pools in Azure Kubernetes Service \(AKS\)](#).
- System-mode node pools must contain at least one node, while user-mode node pools can contain zero or more nodes. Whenever possible, you can configure a user-mode node pool to automatically scale from `0` to `N` nodes. You can configure your workloads to scale out and scale in by using a horizontal pod autoscaler. Base autoscaling on CPU and memory, or use [Kubernetes Event-driven Autoscaling \(KEDA\)](#) to base autoscaling on the metrics of an external system like Apache Kafka, RabbitMQ, or Azure Service Bus.
- Make sure to properly set [requests and limits](#) for your pods to improve application density and avoid assigning too many CPU and memory resources to your workloads. Observe the average and maximum consumption of CPU and memory by using Prometheus or Container Insights. Properly configure limits and quotas for your pods in the YAML manifests, Helm charts, and Kustomize manifests for your deployments.
- Use [ResourceQuota](#) objects to set quotas for the total amount of memory and CPU for all pods that are running in a given [namespace](#). The systematic use of

resource quotas avoids noisy neighbor issues, improves application density, and reduces the number of agent nodes and total costs. Also use [LimitRange](#) objects to configure the default CPU and memory requests for pods in a namespace.

- Use Container Instances for bursting.
- Your AKS workloads might not need to run continuously, such as specific workloads in development cluster node pools. To optimize costs, you can completely turn off an AKS cluster or stop one or more node pools in your AKS cluster. For more information, see [Stop and start an Azure Kubernetes Service \(AKS\) cluster](#) and [Start and stop a node pool on Azure Kubernetes Service \(AKS\)](#).
- Azure Policy integrates with AKS through built-in policies to apply centralized, consistent, at-scale enforcements and safeguards. Enable the Azure Policy add-on on your cluster, and apply the default CPU requests and limits and [memory resource limits](#), which ensure that CPU and memory resource limits are defined on cluster containers.
- Use [Azure Advisor](#) to monitor and release unused resources.
- Use [Microsoft Cost Management](#) budgets and reviews to keep track of expenditures.

Cost governance

The cloud can significantly improve the technical performance of business workloads. Cloud technologies can also reduce the cost and overhead of managing organizational assets. However, this business opportunity also creates risk, because cloud deployments can increase the potential for waste and inefficiencies.

Cost governance is the process of continuously implementing policies or controls to limit spending and costs. Native Kubernetes tooling and Azure tools both support cost governance with proactive monitoring and underlying infrastructure cost optimization.

- [Azure Cost Management + Billing](#) is a suite of Microsoft tools that helps you analyze, manage, and optimize your Azure workload costs. Use the suite to help ensure that your organization is taking advantage of the benefits the cloud provides.
- Review the [Cloud Adoption Framework](#) governance best practices for the [Cost Management Discipline](#) to better understand how to manage and govern cloud costs.

- Explore open-source tools like [KubeCost](#) to monitor and govern AKS cluster cost. You can scope cost allocation to a deployment, service, label, pod, and namespace, which provides flexibility in showing and charging cluster users.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Laura Nicolas](#) | Senior Software Engineer
- [Paolo Salvatori](#) | Principal System Engineer

Other contributors:

- [Chad Kittel](#) | Principal Software Engineer
- [Ed Price](#) | Senior Content Program Manager
- [Theano Petersen](#) | Technical Writer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- AKS for Amazon EKS professionals
- Kubernetes identity and access management
- Kubernetes monitoring and logging
- Secure network access to Kubernetes
- Storage options for a Kubernetes cluster
- Kubernetes node and node pool management
- Cluster governance

Related resources

- Cost governance with Kubecost
- Cost Management discipline overview
- Video - Can cloud native architectures lower your long-term costs? ↗
- Azure pricing calculator ↗
- Plan and manage your Azure costs

Kubernetes node and node pool management

Azure Kubernetes Service (AKS)

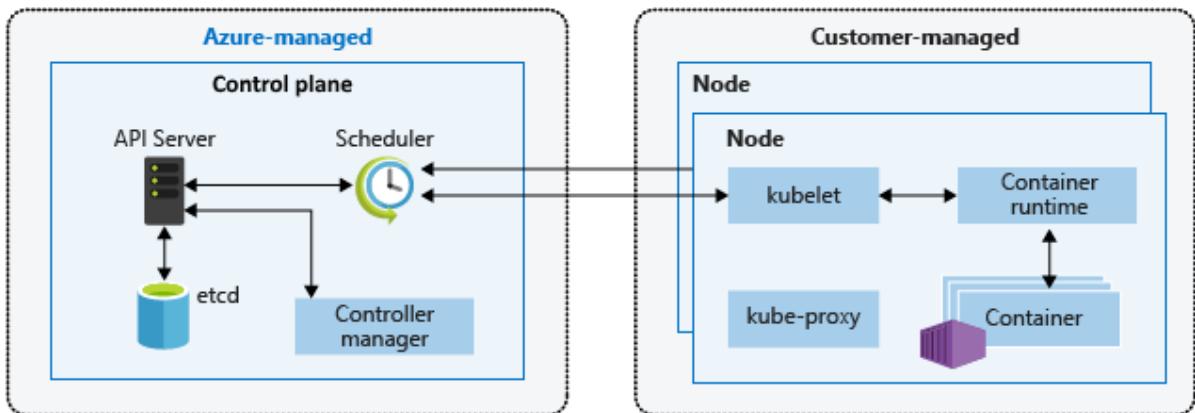
Azure Virtual Machines

Kubernetes architecture is based on two layers: The [control plane](#) and one or more [nodes in node pools](#). This article describes and compares how Amazon Elastic Kubernetes Service (Amazon EKS) and Azure Kubernetes Service (AKS) manage agent or worker nodes.

ⓘ Note

This article is part of a [series of articles](#) that helps professionals who are familiar with [Amazon EKS](#) to understand [AKS](#).

In both Amazon EKS and AKS, the cloud platform provides and manages the control plane layer, and the customer manages the node layer. The following diagram shows the relationship between the control plane and nodes in AKS Kubernetes architecture.



Amazon EKS managed node groups

Amazon EKS [managed node groups](#) automate the provisioning and lifecycle management of Amazon Elastic Compute Cloud (EC2) worker nodes for Amazon EKS clusters. Amazon Web Services (AWS) users can use the [eksctl](#) command-line utility to create, update, or terminate nodes for their EKS clusters. Node updates and terminations automatically cordon and drain nodes to ensure that applications remain available.

Every managed node is provisioned as part of an [Amazon EC2 Auto Scaling group](#) that Amazon EKS operates and controls. The [Kubernetes cluster autoscaler](#) automatically adjusts the number of worker nodes in a cluster when pods fail or are rescheduled onto other nodes. Each node group can be configured to run across multiple [Availability Zones](#) within a region.

For more information about Amazon EKS managed nodes, see [Creating a managed node group](#) and [Updating a managed node group](#).

You can also run Kubernetes pods on [AWS Fargate](#). Fargate provides on-demand, right-sized compute capacity for containers. For more information on how to use Fargate with Amazon EKS, see [AWS Fargate](#).

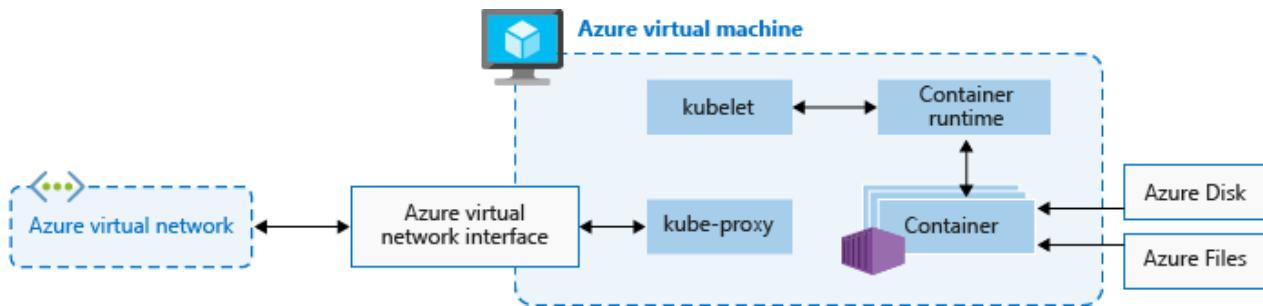
AKS nodes and node pools

Creating an AKS cluster automatically creates and configures a control plane, which provides [core Kubernetes services](#) and application workload orchestration. The Azure platform provides the AKS control plane at no cost as a managed Azure resource. The control plane and its resources exist only in the region where you created the cluster.

The [nodes](#), also called *agent nodes* or *worker nodes*, host the workloads and applications. In AKS, customers fully manage and pay for the agent nodes attached to the AKS cluster.

To run applications and supporting services, an AKS cluster needs at least one node: An Azure virtual machine (VM) to run the Kubernetes node components and container runtime. Every AKS cluster must contain at least one [system node pool](#) with at least one node.

AKS groups nodes of the same configuration into *node pools* of VMs that run AKS workloads. System node pools serve the primary purpose of hosting critical system pods such as CoreDNS. User node pools serve the primary purpose of hosting workload pods. If you want to have only one node pool in your AKS cluster, for example in a development environment, you can schedule application pods on the system node pool.



You can also create multiple user node pools to segregate different workloads on different nodes to avoid the [noisy neighbor problem](#), or to support applications with different compute or storage demands.

Every agent node of a system or user node pool is a VM provisioned as part of [Azure Virtual Machine Scale Sets](#) and managed by the AKS cluster. For more information, see [Nodes and node pools](#).

You can define the initial number and [size](#) for worker nodes when you create an AKS cluster, or when you add new nodes and node pools to an existing AKS cluster. If you don't specify a VM

size, the default size is Standard_D2s_v3 for Windows node pools and Standard_DS2_v2 for Linux node pools.

ⓘ Important

To provide better latency for intra-node calls and communications with platform services, select a VM series that supports [Accelerated Networking](#).

Node pool creation

You can add a node pool to a new or existing AKS cluster by using the Azure portal, [Azure CLI](#), the [AKS REST API](#), or infrastructure as code (IaC) tools such as [Bicep](#), [Azure Resource Manager \(ARM\) templates](#), or [Terraform](#). For more information on how to add node pools to an existing AKS cluster, see [Create and manage multiple node pools for a cluster in Azure Kubernetes Service \(AKS\)](#).

When you create a new node pool, the associated virtual machine scale set is created in the [node resource group](#), an [Azure resource group](#) that contains all the infrastructure resources for the AKS cluster. These resources include the Kubernetes nodes, virtual networking resources, managed identities, and storage.

By default, the node resource group has a name like

`MC_<resourcegroupname>_<clustername>_<location>`. AKS automatically deletes the node resource group when deleting a cluster, so you should use this resource group only for resources that share the cluster's lifecycle.

Add a node pool

The following code example uses the Azure CLI `az aks nodepool add` command to add a node pool named `mynodepool1` with three nodes to an existing AKS cluster called `myAKScluster` in the `myResourceGroup` resource group.

Azure CLI

```
az aks nodepool add \
  --resource-group myResourceGroup \
  --cluster-name myAKScluster \
  --node-vm-size Standard_D8ds_v4 \
  --name mynodepool1 \
  --node-count 3
```

Spot node pools

A spot node pool is a node pool backed by a [spot virtual machine scale set](#). Using spot virtual machines for nodes with your AKS cluster takes advantage of unutilized Azure capacity at a significant cost savings. The amount of available unutilized capacity varies based on many factors, including node size, region, and time of day.

When deploying a spot node pool, Azure allocates the spot nodes if there's capacity available. But there's no SLA for the spot nodes. A spot scale set that backs the spot node pool is deployed in a single fault domain and offers no high availability guarantees. When Azure needs the capacity back, the Azure infrastructure evicts spot nodes, and you get at most a 30-second notice before eviction. Be aware that a spot node pool can't be the cluster's default node pool. A spot node pool can be used only for a secondary pool.

Spot nodes are for workloads that can handle interruptions, early terminations, or evictions. For example, batch processing jobs, development and testing environments, and large compute workloads are good candidates for scheduling on a spot node pool. For more details, see the [spot instance's limitations](#).

The following `az aks nodepool add` command adds a spot node pool to an existing cluster with autoscaling enabled.

Azure CLI

```
az aks nodepool add \
  --resource-group myResourceGroup \
  --cluster-name myAKScluster \
  --name myspotnodepool \
  --node-vm-size Standard_D8ds_v4 \
  --priority Spot \
  --eviction-policy Delete \
  --spot-max-price -1 \
  --enable-cluster-autoscaler \
  --min-count 1 \
  --max-count 3 \
  --no-wait
```

For more information on spot node pools, see [Add a spot node pool to an Azure Kubernetes Service \(AKS\) cluster](#).

Ephemeral OS disks

By default, Azure automatically replicates the VM operating system (OS) disk to Azure Storage to avoid data loss if the VM needs to be relocated to another host. But because containers aren't designed to persist local state, keeping the OS disk in storage offers limited value for AKS. There are some drawbacks, such as slower node provisioning and higher read/write latency.

By contrast, ephemeral OS disks are stored only on the host machine, like a temporary disk, and provide lower read/write latency and faster node scaling and cluster upgrades. Like a

temporary disk, an ephemeral OS disk is included in the VM price, so you incur no extra storage costs.

ⓘ Important

If you don't explicitly request managed disks for the OS, AKS defaults to an ephemeral OS if possible for a given node pool configuration.

To use ephemeral OS, the OS disk must fit in the VM cache. Azure VM documentation shows VM cache size in parentheses next to IO throughput as **cache size in GiB**.

For example, the AKS default Standard_DS2_v2 VM size with the default 100-GB OS disk size supports ephemeral OS, but has only 86 GB of cache size. This configuration defaults to managed disk if you don't explicitly specify otherwise. If you explicitly request ephemeral OS for this size, you get a validation error.

If you request the same Standard_DS2_v2 VM with a 60-GB OS disk, you get ephemeral OS by default. The requested 60-GB OS size is smaller than the maximum 86-GB cache size.

Standard_D8s_v3 with a 100-GB OS disk supports ephemeral OS and has 200 GB of cache space. If a user doesn't specify the OS disk type, a node pool gets ephemeral OS by default.

The following `az aks nodepool add` command shows how to add a new node pool to an existing cluster with an ephemeral OS disk. The `--node-osdisk-type` parameter sets the OS disk type to `Ephemeral`, and the `--node-osdisk-size` parameter defines the OS disk size.

Azure CLI

```
az aks nodepool add \
  --resource-group myResourceGroup \
  --cluster-name myAKScluster \
  --name mynewnodepool \
  --node-vm-size Standard_D8ds_v4 \
  --node-osdisk-type Ephemeral \
  --node-osdisk-size 48
```

For more information about ephemeral OS disks, see [Ephemeral OS](#).

Virtual nodes

You can use virtual nodes to quickly scale out application workloads in an AKS cluster. Virtual nodes give you quick pod provisioning, and you only pay per second for execution time. You don't need to wait for the cluster autoscaler to deploy new worker nodes to run more pod replicas. Virtual nodes are supported only with Linux pods and nodes. The virtual nodes add-on for AKS is based on the open-source [Virtual Kubelet](#) project.

Virtual node functionality depends on [Azure Container Instances](#). For more information about virtual nodes, see [Create and configure an Azure Kubernetes Services \(AKS\) cluster to use virtual nodes](#).

Taints, labels, and tags

When you create a node pool, you can add Kubernetes [taints](#) and [labels](#), and [Azure tags](#), to that node pool. When you add a taint, label, or tag, all nodes within that node pool get that taint, label, or tag.

To create a node pool with a taint, you can use the `az aks nodepool add` command with the `--node-taints` parameter. To label the nodes in a node pool, you can use the `--labels` parameter and specify a list of labels, as shown in the following code:

Azure CLI

```
az aks nodepool add \
  --resource-group myResourceGroup \
  --cluster-name myAKScluster \
  --name mynodepool \
  --node-vm-size Standard_NC6 \
  --node-taints sku=gpu:NoSchedule \
  --labels dept=IT costcenter=9999
```

For more information, see [Specify a taint, label, or tag for a node pool](#).

Reserved system labels

Amazon EKS adds automated Kubernetes labels to all nodes in a managed node group like `eks.amazonaws.com/capacityType`, which specifies the capacity type. AKS also automatically adds system labels to agent nodes.

The following prefixes are reserved for AKS use and can't be used for any node:

- `kubernetes.azure.com/`
- `kubernetes.io/`

For other reserved prefixes, see [Kubernetes well-known labels, annotations, and taints](#).

The following table lists labels that are reserved for AKS use and can't be used for any node. In the table, the **Virtual node usage** column specifies whether the label is supported on virtual nodes.

In the **Virtual node usage** column:

- **N/A** means the property doesn't apply to virtual nodes because it would require modifying the host.

- **Same** means the expected values are the same for a virtual node pool as for a standard node pool.
- **Virtual** replaces VM SKU values, because virtual node pods don't expose any underlying VM.
- **Virtual node version** refers to the current version of the [virtual Kubelet-ACI connector release](#).
- **Virtual node subnet name** is the subnet that deploys virtual node pods into Azure Container Instances.
- **Virtual node virtual network** is the virtual network that contains the virtual node subnet.

Expand table

Label	Value	Example, options	Virtual node usage
kubernetes.azure.com/agentpool	<agent pool name>	nodepool1	Same
kubernetes.io/arch	amd64	runtime.GOARCH	N/A
kubernetes.io/os	<OS Type>	Linux or Windows	Linux
node.kubernetes.io/instance-type	<VM size>	Standard_NC6	Virtual
topology.kubernetes.io/region	<Azure region>	westus2	Same
topology.kubernetes.io/zone	<Azure zone>	0	Same
kubernetes.azure.com/cluster	<MC_RgName>	MC_aks_myAKSCluster_westus2	Same
kubernetes.azure.com/mode	<mode>	User or System	User
kubernetes.azure.com/role	agent	Agent	Same
kubernetes.azure.com/scalesetpriority	<scale set priority>	Spot or Regular	N/A
kubernetes.io/hostname	<hostname>	aks-nodepool-00000000-vms00000	Same

Label	Value	Example, options	Virtual node usage
kubernetes.azure.com/storageprofile	<OS disk storage profile>	Managed	N/A
kubernetes.azure.com/storagetier	<OS disk storage tier>	Premium_LRS	N/A
kubernetes.azure.com/instance-sku	<SKU family>	Standard_N	Virtual
kubernetes.azure.com/node-image-version	<VHD version>	AKSUbuntu-1804-2020.03.05	Virtual node version
kubernetes.azure.com/subnet	<nodepool subnet name>	subnetName	Virtual node subnet name
kubernetes.azure.com/vnet	<nodepool virtual network name>	vnetName	Virtual node virtual network
kubernetes.azure.com/ppg	<nodepool ppg name>	ppgName	N/A
kubernetes.azure.com/encrypted-set	<nodepool encrypted-set name>	encrypted-set-name	N/A
kubernetes.azure.com/accelerator	<accelerator>	Nvidia	N/A
kubernetes.azure.com/fips_enabled	<fips enabled>	True	N/A
kubernetes.azure.com/os-sku	<os/sku>	See Create or update OS SKU	Linux SKU

Windows node pools

AKS supports creating and using Windows Server container node pools through the [Azure CNI](#) network plugin. To plan the required subnet ranges and network considerations, see [configure Azure CNI networking](#).

The following `az aks nodepool add` command adds a node pool that runs Windows Server containers.

Azure CLI

```
az aks nodepool add \
  --resource-group myResourceGroup \
  --cluster-name myAKSCluster \
  --name mywindowsnodepool \
  --node-vm-size Standard_D8ds_v4 \
  --os-type Windows \
  --node-count 1
```

The preceding command uses the default subnet in the AKS cluster virtual network. For more information about how to build an AKS cluster with a Windows node pool, see [Create a Windows Server container in AKS](#).

Node pool considerations

The following considerations and limitations apply when you create and manage node pools and multiple node pools:

- [Quotas, VM size restrictions, and region availability](#) apply to AKS node pools.
- System pools must contain at least one node. You can delete a system node pool if you have another system node pool to take its place in the AKS cluster. User node pools can contain zero or more nodes.
- You can't change the VM size of a node pool after you create it.
- For multiple node pools, the AKS cluster must use the Standard SKU load balancers. Basic SKU load balancers don't support multiple node pools.
- All cluster node pools must be in the same virtual network, and all subnets assigned to any node pool must be in the same virtual network.
- If you create multiple node pools at cluster creation time, the Kubernetes versions for all node pools must match the control plane version. You can update versions after the cluster has been provisioned by using per-node-pool operations.

Node pool scaling

As your application workload changes, you might need to change the number of nodes in a node pool. You can scale the number of nodes up or down manually by using the [az aks nodepool scale](#) command. The following example scales the number of nodes in `mynodepool` to five:

Azure CLI

```
az aks nodepool scale \
--resource-group myResourceGroup \
--cluster-name myAKSCluster \
--name mynodepool \
--node-count 5
```

Scale node pools automatically by using the cluster autoscaler

AKS supports scaling node pools automatically with the [cluster autoscaler](#). You enable this feature on each node pool, and define a minimum and a maximum number of nodes.

The following `az aks nodepool add` command adds a new node pool called `mynodepool` to an existing cluster. The `--enable-cluster-autoscaler` parameter enables the cluster autoscaler on the new node pool, and the `--min-count` and `--max-count` parameters specify the minimum and maximum number of nodes in the pool.

Azure CLI

```
az aks nodepool add \
--resource-group myResourceGroup \
--cluster-name myAKSCluster \
--name mynewnodepool \
--node-vm-size Standard_D8ds_v4 \
--enable-cluster-autoscaler \
--min-count 1 \
--max-count 5
```

The following `az aks nodepool update` command updates the minimum number of nodes from one to three for the `mynewnodepool` node pool.

Azure CLI

```
az aks nodepool update \
--resource-group myResourceGroup \
--cluster-name myAKSCluster \
--name mynewnodepool \
--update-cluster-autoscaler \
--min-count 1 \
--max-count 3
```

You can disable the cluster autoscaler with `az aks nodepool update` by passing the `--disable-cluster-autoscaler` parameter.

Azure CLI

```
az aks nodepool update \
--resource-group myResourceGroup \
--cluster-name myAKScluster \
--name mynodepool \
--disable-cluster-autoscaler
```

To re-enable the cluster autoscaler on an existing cluster, use `az aks nodepool update`, specifying the `--enable-cluster-autoscaler`, `--min-count`, and `--max-count` parameters.

For more information about how to use the cluster autoscaler for individual node pools, see [Automatically scale a cluster to meet application demands on Azure Kubernetes Service \(AKS\)](#).

Updates and upgrades

Azure periodically updates its VM hosting platform to improve reliability, performance, and security. These updates range from patching software components in the hosting environment to upgrading networking components or decommissioning hardware. For more information about how Azure updates VMs, see [Maintenance for virtual machines in Azure](#).

VM hosting infrastructure updates don't usually affect hosted VMs, such as agent nodes of existing AKS clusters. For updates that affect hosted VMs, Azure minimizes the cases that require reboots by pausing the VM while updating the host, or live-migrating the VM to an already updated host.

If an update requires a reboot, Azure provides notification and a time window so you can start the update when it works for you. The self-maintenance window for host machines is typically 35 days, unless the update is urgent.

You can use Planned Maintenance to update VMs, and manage planned maintenance notifications with [Azure CLI](#), [PowerShell](#), or the [Azure portal](#). Planned Maintenance detects if you're using Cluster Auto-Upgrade, and schedules upgrades during your maintenance window automatically. For more information about Planned Maintenance, see the [az aks maintenanceconfiguration](#) command and [Use Planned Maintenance to schedule maintenance windows for your Azure Kubernetes Service \(AKS\) cluster](#).

Kubernetes upgrades

Part of the AKS cluster lifecycle is periodically upgrading to the latest Kubernetes version. It's important to apply upgrades to get the latest security releases and features. To upgrade the

Kubernetes version of existing node pool VMs, you must cordon and drain nodes and replace them with new nodes that are based on an updated Kubernetes disk image.

By default, AKS configures upgrades to surge with one extra node. A default value of one for the `max-surge` settings minimizes workload disruption by creating an extra node to replace older-versioned nodes before cordoning or draining existing applications. You can customize the `max-surge` value per node pool to allow for a tradeoff between upgrade speed and upgrade disruption. Increasing the `max-surge` value completes the upgrade process faster, but a large value for `max-surge` might cause disruptions during the upgrade process.

For example, a `max-surge` value of 100% provides the fastest possible upgrade process by doubling the node count, but also causes all nodes in the node pool to be drained simultaneously. You might want to use this high value for testing, but for production node pools, a `max-surge` setting of 33% is better.

AKS accepts both integer and percentage values for `max-surge`. An integer such as `5` indicates five extra nodes to surge. Percent values for `max-surge` can be a minimum of `1%` and a maximum of `100%`, rounded up to the nearest node count. A value of `50%` indicates a surge value of half the current node count in the pool.

During an upgrade, the `max-surge` value can be a minimum of `1` and a maximum value equal to the number of nodes in the node pool. You can set larger values, but the maximum number of nodes used for `max-surge` won't be higher than the number of nodes in the pool.

Important

For upgrade operations, node surges need enough subscription quota for the requested `max-surge` count. For example, a cluster that has five node pools, each with four nodes, has a total of 20 nodes. If each node pool has a `max-surge` value of 50%, you need additional compute and IP quota of 10 nodes, or two nodes times five pools, to complete the upgrade.

If you use Azure Container Networking Interface (CNI), also make sure you have enough IPs in the subnet to [meet CNI requirements for AKS](#).

Upgrade node pools

To see available upgrades, use [az aks get-upgrades](#).

Azure CLI

```
az aks get-upgrades --resource-group <myResourceGroup> --name <myAKSCluster>
```

To see the status of node pools, use [az aks nodepool list](#).

Azure CLI

```
az aks nodepool list -g <myResourceGroup> --cluster-name <myAKSCluster>
```

The following command uses `az aks nodepool upgrade` to upgrade a single node pool.

Azure CLI

```
az aks nodepool upgrade \  
  --resource-group <myResourceGroup> \  
  --cluster-name <myAKSCluster> \  
  --name <mynodepool> \  
  --kubernetes-version <KUBERNETES_VERSION>
```

For more information about how to upgrade the Kubernetes version for a cluster control plane and node pools, see:

- [Azure Kubernetes Service \(AKS\) node image upgrade](#)
- [Upgrade a cluster control plane with multiple node pools](#)

Upgrade considerations

Note these best practices and considerations for upgrading the Kubernetes version in an AKS cluster.

- It's best to upgrade all node pools in an AKS cluster to the same Kubernetes version. The default behavior of `az aks upgrade` upgrades all node pools and the control plane.
- Manually upgrade, or set an auto-upgrade channel on your cluster. If you use Planned Maintenance to patch VMs, auto-upgrades also start during your specified maintenance window. For more information, see [Upgrade an Azure Kubernetes Service \(AKS\) cluster](#).
- The `az aks upgrade` command with the `--control-plane-only` flag upgrades only the cluster control plane and doesn't change any of the associated node pools in the cluster. To upgrade individual node pools, specify the target node pool and Kubernetes version in the `az aks nodepool upgrade` command,
- An AKS cluster upgrade triggers a cordon and drain of your nodes. If you have low compute quota available, the upgrade could fail. For more information about increasing your quota, see [Increase regional vCPU quotas](#).
- Configure the `max-surge` parameter based on your needs, using an integer or a percentage value. For production node pools, use a `max-surge` setting of 33%. For more information, see [Customize node surge upgrade](#).
- When you upgrade an AKS cluster that uses CNI networking, make sure the subnet has enough available private IP addresses for the extra nodes the `max-surge` settings create.

For more information, see [Configure Azure CNI networking in Azure Kubernetes Service \(AKS\)](#).

- If your cluster node pools span multiple Availability Zones within a region, the upgrade process can temporarily cause an unbalanced zone configuration. For more information, see [Special considerations for node pools that span multiple Availability Zones](#).

Node virtual networks

When you create a new cluster or add a new node pool to an existing cluster, you specify the resource ID of a subnet within the cluster [virtual network](#) where you deploy the agent nodes. A workload might require splitting a cluster's nodes into separate node pools for logical isolation. You can achieve this isolation with separate subnets, each dedicated to a separate node pool. The node pool VMs each get a private IP address from their associated subnet.

AKS supports two networking plugins:

- [Kubenet](#) is a basic, simple network plugin for Linux. With `kubenet`, nodes get a private IP address from the Azure virtual network subnet. Pods get an IP address from a logically different address space. Network address translation (NAT) lets the pods reach resources on the Azure virtual network by translating the source traffic's IP address to the node's primary IP address. This approach reduces the number of IP addresses you need to reserve in your network space for pods.
- [Azure Container Networking Interface \(CNI\)](#) gives every pod an IP address to call and access directly. These IP addresses must be unique across your network space. Each node has a configuration parameter for the maximum number of pods that it supports. The equivalent number of IP addresses per node are then reserved for that node. This approach requires advance planning, and can lead to IP address exhaustion or the need to rebuild clusters in a larger subnet as application demands grow.

When you create a new cluster or add a new node pool to a cluster that uses Azure CNI, you can specify the resource ID of two separate subnets, one for the nodes and one for the pods. For more information, see [Dynamic allocation of IPs and enhanced subnet support](#).

Dynamic IP allocation

Pods that use [Azure CNI](#) get private IP addresses from a subnet of the hosting node pool. Azure CNI [dynamic IP allocation](#) can allocate private IP addresses to pods from a subnet that's separate from the node pool hosting subnet. This feature provides the following advantages:

- The pod subnet dynamically allocates IPs to pods. Dynamic allocation provides better IP utilization compared to the traditional CNI solution, which does static allocation of IPs for every node.

- You can scale and share node and pod subnets independently. You can share a single pod subnet across multiple node pools or clusters deployed in the same virtual network. You can also configure a separate pod subnet for a node pool.
- Because pods have virtual network private IPs, they have direct connectivity to other cluster pods and resources in the virtual network. This ability supports better performance for very large clusters.
- If pods have a separate subnet, you can configure virtual network policies for pods that are different from node policies. Separate policies allow many useful scenarios, such as allowing internet connectivity only for pods and not for nodes, fixing the source IP for a pod in a node pool by using NAT Gateway, and using [Network Security Groups \(NSGs\)](#) to filter traffic between node pools.
- Both *Network Policy* and *Calico* Kubernetes network policies work with dynamic IP allocation.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Paolo Salvatori](#) | Principal System Engineer

Other contributors:

- [Laura Nicolas](#) | Senior Software Engineer
- [Chad Kittel](#) | Principal Software Engineer
- [Ed Price](#) | Senior Content Program Manager
- [Theano Petersen](#) | Technical Writer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- [AKS for Amazon EKS professionals](#)
- [Kubernetes identity and access management](#)
- [Kubernetes monitoring and logging](#)
- [Secure network access to Kubernetes](#)
- [Storage options for a Kubernetes cluster](#)
- [Cost management for Kubernetes](#)
- [Cluster governance](#)
- [Azure Kubernetes Service \(AKS\) solution journey](#)
- [Azure Kubernetes Services \(AKS\) day-2 operations guide](#)

- Choose a Kubernetes at the edge compute option
- GitOps for Azure Kubernetes Service

Related resources

- [AKS cluster best practices](#)
- [Create a Private AKS cluster with a Public DNS Zone ↗](#)
- [Create a private Azure Kubernetes Service cluster using Terraform and Azure DevOps ↗](#)
- [Create a public or private Azure Kubernetes Service cluster with Azure NAT Gateway and Azure Application Gateway ↗](#)
- [Use Private Endpoints with a Private AKS Cluster ↗](#)
- [Create an Azure Kubernetes Service cluster with the Application Gateway Ingress Controller ↗](#)
- [Introduction to Kubernetes](#)
- [Introduction to Kubernetes on Azure](#)
- [Implement Azure Kubernetes Service \(AKS\)](#)
- [Develop and deploy applications on Kubernetes](#)
- [Optimize compute costs on Azure Kubernetes Service \(AKS\)](#)

Kubernetes cluster governance

Article • 03/29/2023

Governance refers to an organization's ability to enforce and validate rules to guarantee compliance with corporate standards. Governance helps organizations mitigate risks, comply with corporate standards and external regulations, and minimize interruption to adoption or innovation.

Governance includes planning initiatives, setting strategic priorities, and using mechanisms and processes to control applications and resources. For Kubernetes clusters in a cloud environment, governance means implementing policies across Kubernetes clusters and the applications that run in those clusters.

Kubernetes governance includes both the cloud environment and cluster deployment infrastructure, and the clusters themselves and their applications. This guide focuses on governance within Kubernetes clusters. The article describes and compares how Amazon Elastic Kubernetes Service (Amazon EKS) and Azure Kubernetes Service (AKS) manage Kubernetes cluster governance.

ⓘ Note

This article is part of a [series of articles](#) that helps professionals who are familiar with [Amazon EKS](#) to understand [AKS](#).

Kubernetes governance dimensions

Three dimensions define a consistent Kubernetes governance strategy:

- **Targets** describe the security and compliance policy goals a governance strategy should meet. For example, targets specify which users can access a Kubernetes cluster, namespace, or application, or which container registries and images to use in which clusters. The security operations team usually sets these targets as the first step in defining a company's governance strategy.
- **Scopes** detail the elements that the target policies apply to. Scopes must address all Kubernetes-visible components. Scopes can be organizational units like departments, teams, and groups, or environments like clouds, regions, or namespaces, or both.

- **Policy directives** use Kubernetes capabilities to enforce the target rules across the specified scopes to enforce the governance policies.

For more information, see [Kubernetes governance, what you should know](#).

Governance in EKS and AKS

- Amazon Web Services (AWS) customers usually use Kyverno, Gatekeeper, or other third-party solutions to define and implement a governance strategy for their Amazon EKS clusters. The [aws-eks-best-practices/policies](#) GitHub repository contains a collection of example policies for Kyverno and Gatekeeper.
- Azure customers can also use Kyverno or Gatekeeper, and can use the [Azure Policy for Kubernetes Add-on](#) to extend Gatekeeper for an AKS governance strategy.

Gatekeeper

The Cloud Native Computing Foundation (CNCF) [sponsors](#) the open-source [Gatekeeper Policy Controller for Kubernetes](#) for enforcing policies in Kubernetes clusters. Gatekeeper is a Kubernetes admission controller that enforces policies created with [Open Policy Agent \(OPA\)](#), a general-purpose policy engine.

OPA uses a high-level declarative language called [Rego](#) to create policies that can run pods from tenants on separate instances or at different priorities. For a collection of common OPA policies, see the [OPA Gatekeeper Library](#).

Kyverno

CNCF also sponsors the [Kyverno](#) open-source project for enforcing policies in Kubernetes clusters. Kyverno is a Kubernetes-native policy engine that can validate, mutate, and generate Kubernetes resource configurations with policies.

With Kyverno, you can define and manage policies as Kubernetes resources without using a new language. This approach allows using familiar tools such as [kubectl](#), [git](#), and [kustomize](#) to manage policies.

Kyverno uses `kustomize`-style overlays for validation, supports JSON patch and strategic merge patch for mutation, and can clone resources across namespaces based on flexible triggers. You can deploy policies individually by using their YAML manifests, or package and deploy them by using Helm charts.

Kyverno, unlike Gatekeeper or Azure Policy for AKS, can generate new Kubernetes objects with policies, not just validate or mutate existing resources. For example, you can define a Kyverno policy to automate the creation of a default network policy for any new namespace.

For more information, see the official [Kyverno installation guide](#). For a list of ready-to-use or customizable policies, see the Kyverno [Policies](#) library.

Optionally, you can deploy Kyverno's implementation of the [Kubernetes Pod Security Standards \(PSS\)](#) as [Kyverno policies](#). The PSS controls provide a starting point for general Kubernetes cluster operational security.

Azure Policy Add-on for AKS

The [Azure Policy Add-on for AKS](#) extends [Gatekeeper](#) to apply at-scale enforcements and safeguards on AKS clusters in a centralized, consistent manner. [Azure Policy](#) enables centralized compliance management and reporting for multiple Kubernetes clusters from a single location. This capability makes management and governance of multicloud environments more efficient than deploying and managing Kyverno or Gatekeeper for each cluster.

The Azure Policy Add-on for AKS enacts the following functions:

- Checks with the Azure Policy service for policy assignments to the cluster.
- Deploys policy definitions into the cluster as constraint template and constraint custom resources.
- Reports auditing and compliance details back to the Azure Policy service.

The Azure Policy Add-on supports the [AKS](#) and [Azure Arc-enabled Kubernetes](#) cluster environments. For more information, see [Understand Azure Policy for Kubernetes clusters](#). To install the add-on on new and existing clusters, see [Install the Azure Policy Add-on for AKS](#).

After you install the Azure Policy Add-on for AKS, you can apply individual policy definitions or groups of policy definitions called initiatives to your AKS cluster. You can apply and enforce [Azure Policy built-in policy and initiative definitions](#) from the outset, or [create and assign your own custom policy definitions](#). The [Azure Policy](#) built-in security policies help improve the security posture of your AKS cluster, enforce organizational standards, and assess compliance at scale.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Martin Gjoshevski](#) | Senior Service Engineer
- [Paolo Salvatori](#) | Principal Service Engineer

Other contributors:

- [Chad Kittel](#) | Principal Software Engineer
- [Ed Price](#) | Senior Content Program Manager
- [Theano Petersen](#) | Technical Writer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- [AKS for Amazon EKS professionals](#)
- [Kubernetes identity and access management](#)
- [Kubernetes monitoring and logging](#)
- [Secure network access to Kubernetes](#)
- [Storage options for a Kubernetes cluster](#)
- [Cost management for Kubernetes](#)
- [Kubernetes node and node pool management](#)

Related resources

- [Policy for Kubernetes](#)
- [Secure your AKS cluster with Azure Policy](#)
- [Governance disciplines for AKS](#)
- [OPA Gatekeeper: Policy and Governance for Kubernetes](#)

Tutorial: Discover AWS instances with Azure Migrate: Discovery and assessment

Article • 11/03/2023

As part of your migration journey to Azure, you discover your servers for assessment and migration.

This tutorial shows you how to discover Amazon Web Services (AWS) instances with the Azure Migrate: Discovery and assessment tool, using a lightweight Azure Migrate appliance. You deploy the appliance as a physical server, to continuously discover machine and performance metadata.

In this tutorial, you learn how to:

- ✓ Set up an Azure account.
- ✓ Prepare AWS instances for discovery.
- ✓ Create a project.
- ✓ Set up the Azure Migrate appliance.
- ✓ Start continuous discovery.

ⓘ Note

Tutorials show the quickest path for trying out a scenario, and use default options.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

Before you start this tutorial, check you have these prerequisites in place.

Requirement	Details
Appliance	<p>You need an EC2 VM on which to run the Azure Migrate appliance. The machine should have:</p> <ul style="list-style-type: none">- Windows Server 2019 or Windows Server 2022 installed.- 16-GB RAM, 8 vCPUs, around 80 GB of disk storage, and an external virtual switch.- A static or dynamic IP address, with internet access, either directly or through a proxy.
Windows instances	Allow inbound connections on WinRM port 5985 (HTTP) for discovery of Windows servers.
Linux instances	<p>Allow inbound connections on port 22 (TCP) for discovery of Linux servers.</p> <p>The instances should use <code>bash</code> as the default shell, otherwise discovery will fail.</p>

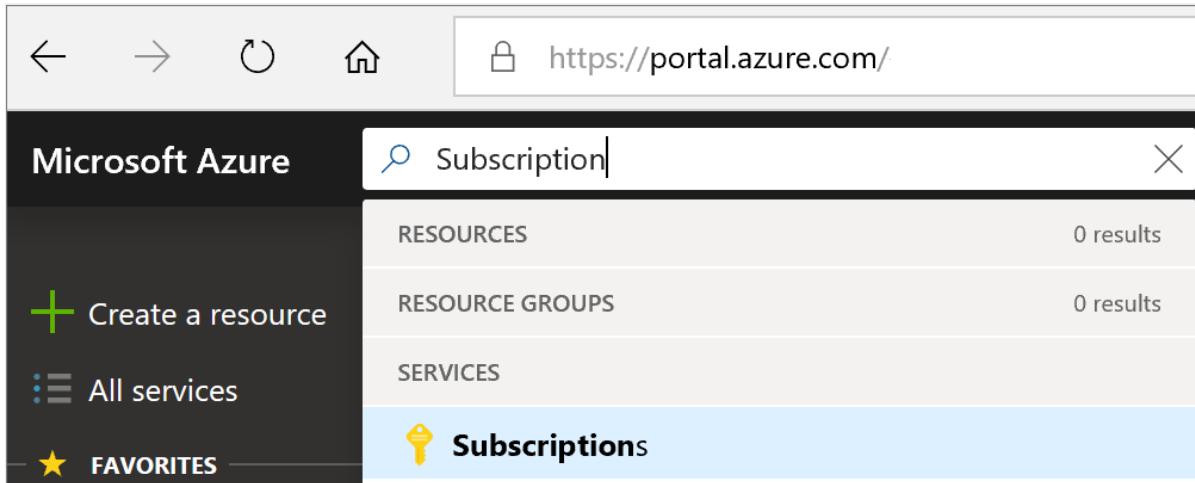
Prepare an Azure user account

To create a project and register the Azure Migrate appliance, you need an account with:

- Contributor or Owner permissions on an Azure subscription.
- Permissions to register Microsoft Entra apps.

If you just created a free Azure account, you're the owner of your subscription. If you're not the subscription owner, work with the owner to assign the permissions as follows:

1. In the Azure portal, search for "subscriptions", and under **Services**, select **Subscriptions**.



2. In the **Subscriptions** page, select the subscription in which you want to create a project.
3. Select **Access control (IAM)**.
4. Select **Add > Add role assignment** to open the **Add role assignment** page.
5. Assign the following role. For detailed steps, see [Assign Azure roles using the Azure portal](#).

Setting	Value
Role	Contributor or Owner
Assign access to	User
Members	azmigrateuser

Name ↑↓	Description ↑↓	Type ↑↓	Category ↑↓	Details
Owner	Grants full access to manage all resources, including the ability to a...	BuiltInRole	General	View
Contributor	Grants full access to manage all resources, but does not allow you ...	BuiltInRole	General	View
Reader	View all resources, but does not allow you to make any changes.	BuiltInRole	General	View
AcrDelete	acr delete	BuiltInRole	Containers	View
AcrImageSigner	acr image signer	BuiltInRole	Containers	View
AcrPull	acr pull	BuiltInRole	Containers	View
AcrPush	acr push	BuiltInRole	Containers	View
AcrQuarantineReader	acr quarantine data reader	BuiltInRole	Containers	View
AcrQuarantineWriter	acr quarantine data writer	BuiltInRole	Containers	View

6. To register the appliance, your Azure account needs **permissions to register Microsoft Entra apps**.
7. In the portal, go to **Microsoft Entra ID > Users**.
8. Request the tenant or global admin to assign the **Application Developer role** to the account to allow Microsoft Entra app registration by users. [Learn more](#).

Prepare AWS instances

Set up an account that the appliance can use to access AWS instances.

- For **Windows servers**, set up a local user account on all the Windows servers that you want to include in the discovery. Add the user account to the following groups: - Remote Management Users - Performance Monitor Users - Performance Log users.
- For **Linux servers**, you need a root account on the Linux servers that you want to discover. Refer to the instructions in the [support matrix](#) for an alternative.
- Azure Migrate uses password authentication when discovering AWS instances. AWS instances don't support password authentication by default. Before you can discover instance, you need to enable password authentication.
 - For Windows servers, allow WinRM port 5985 (HTTP). This allows remote WMI calls.
 - For Linux servers:
 1. Sign into each Linux machine.
 2. Open the sshd_config file: `vi /etc/ssh/sshd_config`
 3. In the file, locate the **PasswordAuthentication** line, and change the value to **yes**.
 4. Save the file and close it. Restart the ssh service.

- If you are using a root user to discover your Linux servers, ensure root sign in is allowed on the servers.
 1. Sign into each Linux machine
 2. Open the sshd_config file: vi /etc/ssh/sshd_config
 3. In the file, locate the **PermitRootLogin** line, and change the value to **yes**.
 4. Save the file and close it. Restart the ssh service.

Set up a project

Set up a new project.

1. In the Azure portal > **All services**, search for **Azure Migrate**.
2. Under **Services**, select **Azure Migrate**.
3. In **Get started**, select **Create project**.
4. In **Create project**, select your Azure subscription and resource group. Create a resource group if you don't have one.
5. In **Project Details**, specify the project name and the geography in which you want to create the project. Review supported geographies for **public** and **government clouds**.
6. Select **Create**.
7. Wait a few minutes for the project to deploy. The **Azure Migrate: Discovery and assessment** tool is added by default to the new project.

Home > Azure Migrate

Azure Migrate | Servers, databases and web apps

Microsoft

Search | Create project | Refresh | Feedback

Get started | Explore more

Migration goals

- Servers, databases and web apps
- Databases (only)
- VDI
- Web apps
- Data Box

Assessment tools

Azure Migrate: Discovery and assessment

Discover | Build business case | Dependency analysis | Assess | Overview | Resolve issues

Quick start

- 1: Discover**
Discover your on-premises servers by using an appliance or importing in a CSV format. Click "Discover" to get started.
- 2: Build and review business case**
Review the recommended migration strategy and financial analysis for migrating your datacenter to Azure.
- 3: Analyze dependencies**
Analyze dependencies between servers. Click 'Dependency analysis' to get started.
- 4: Assess**
Assess discovered servers for migration to Azure. Click 'Assess' to get started.

Add more assessment tools? [Click here](#).

① Note

If you have already created a project, you can use the same project to register additional appliances to discover and assess more no of servers. [Learn more](#).

Set up the appliance

The Azure Migrate appliance is a lightweight appliance, used by Azure Migrate: Discovery and assessment to do the following:

- Discover on-premises servers.
- Send metadata and performance data for discovered servers to Azure Migrate: Discovery and assessment.

[Learn more](#) about the Azure Migrate appliance.

To set up the appliance you:

1. Provide an appliance name and generate a project key in the portal.
2. Download a zipped file with Azure Migrate installer script from the Azure portal.
3. Extract the contents from the zipped file. Launch the PowerShell console with administrative privileges.
4. Execute the PowerShell script to launch the appliance web application.
5. Configure the appliance for the first time, and register it with the project using the project key.

1. Generate the project key

1. In **Migration goals > Servers, databases and web apps > Azure Migrate: Discovery and assessment**, select **Discover**.
2. In **Discover servers > Are your servers virtualized?**, select **Physical or other (AWS, GCP, Xen, etc.)**.
3. In **1:Generate project key**, provide a name for the Azure Migrate appliance that you will set up for discovery of physical or virtual servers. The name should be alphanumeric with 14 characters or fewer.
4. Select **Generate key** to start the creation of the required Azure resources. Do not close the Discover servers page during the creation of resources.
5. After the successful creation of the Azure resources, a **project key** is generated.
6. Copy the key as you will need it to complete the registration of the appliance during its configuration.

2. Download the installer script

In 2: **Download Azure Migrate appliance**, select **Download**.

Verify security

Check that the zipped file is secure, before you deploy it.

1. On the machine to which you downloaded the file, open an administrator command window.
2. Run the following command to generate the hash for the zipped file:

- `C:\>CertUtil -HashFile <file_location> [Hashing Algorithm]`
- Example usage for public cloud: `C:\>CertUtil -HashFile C:\Users\administrator\Desktop\AzureMigrateInstaller-Server-Public.zip SHA256`
- Example usage for government cloud: `C:\>CertUtil -HashFile C:\Users\administrator\Desktop\AzureMigrateInstaller-Server-USGov.zip SHA256`

3. Verify the latest appliance versions and hash values:

- For the public cloud:

Scenario	Download*	Hash value
Physical (85 MB)	Latest version ↗	7EF01AE30F7BB8F4486EDC1688481DB656FB8ECA7B9EF6363B4DAB1CFCFDA141

- For Azure Government:

Scenario	Download*	Hash value
Physical (85 MB)	Latest version ↗	7EF01AE30F7BB8F4486EDC1688481DB656FB8ECA7B9EF6363B4DAB1CFCFDA141

3. Run the Azure Migrate installer script

The installer script does the following:

- Installs agents and a web application for physical server discovery and assessment.
- Install Windows roles, including Windows Activation Service, IIS, and PowerShell ISE.
- Download and installs an IIS rewritable module.
- Updates a registry key (HKLM) with persistent setting details for Azure Migrate.
- Creates the following files under the path:
 - **Config Files:** %Programdata%\Microsoft Azure\Config
 - **Log Files:** %Programdata%\Microsoft Azure\Logs

Run the script as follows:

1. Extract the zipped file to a folder on the server that will host the appliance. Make sure you don't run the script on a machine on an existing Azure Migrate appliance.
2. Launch PowerShell on the above server with administrative (elevated) privilege.
3. Change the PowerShell directory to the folder where the contents have been extracted from the downloaded zipped file.
4. Run the script named **AzureMigrateInstaller.ps1** by running the following command:

- For the public cloud:

```
PS C:\Users\administrator\Desktop\AzureMigrateInstaller-Server-Public>
.\AzureMigrateInstaller.ps1
```

- For Azure Government:

```
PS C:\Users\Administrators\Desktop\AzureMigrateInstaller-Server-
USGov>.\AzureMigrateInstaller.ps1
```

The script will launch the appliance web application when it finishes successfully.

If you come across any issues, you can access the script logs at C:\ProgramData\Microsoft Azure\Logs\AzureMigrateScenarioInstaller_Timestamp.log for troubleshooting.

Verify appliance access to Azure

Make sure that the appliance can connect to Azure URLs for [public](#) and [government](#) clouds.

4. Configure the appliance

Set up the appliance for the first time.

1. Open a browser on any machine that can connect to the appliance, and open the URL of the appliance web app: <https://appliance name or IP address: 44368>.

Alternately, you can open the app from the desktop by selecting the app shortcut.

2. Accept the [license terms](#), and read the third-party information.

Set up prerequisites and register the appliance

In the configuration manager, select **Set up prerequisites**, and then complete these steps:

1. **Connectivity:** The appliance checks that the server has internet access. If the server uses a proxy:

- Select **Setup proxy** to specify the proxy address (in the form `http://ProxyIPAddress` or `http://ProxyFQDN`, where *FQDN* refers to a *fully qualified domain name*) and listening port.
- Enter credentials if the proxy needs authentication.
- If you have added proxy details or disabled the proxy or authentication, select **Save** to trigger connectivity and check connectivity again.

Only HTTP proxy is supported.

2. **Time sync:** Check that the time on the appliance is in sync with internet time for discovery to work properly.
3. **Install updates and register appliance:** To run auto-update and register the appliance, follow these steps:

Appliance Configuration Manager
Cloud: Public

Azure Migrate appliance helps you discover, assess and migrate **VMware virtual machines (VMs)**. Complete the following steps to initiate discovery. [Learn more](#) about Azure Migrate discovery capabilities.

1. Set up prerequisites

- Check connectivity to Azure Set up proxy
- Check time is in sync with Azure
- Check latest updates and register appliance

Verification of Azure Migrate project key
Please provide the Azure Migrate project key, generated on the portal to register the appliance. When you click on 'Verify', the Azure Migrate project key will be validated. [Learn more](#) about how the Azure Migrate project key is generated.

Register VMware appliance by pasting the key here

Appliance auto-update status
Auto-update will be run automatically after successful verification of the key.

Azure user login and appliance registration status
You need to Login to your Azure account to proceed. Ensure you have the [required permissions](#) on the Azure account.

Login **Logout**

Check if VMware Virtual Disk Development Kit is installed Installation instructions

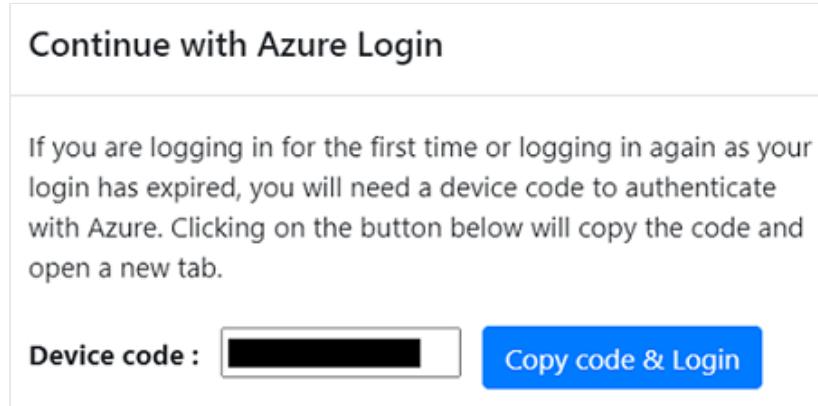
Rerun prerequisites

⚠ Note

This is a new user experience in Azure Migrate appliance which is available only if you have set up an appliance using the latest OVA/Installer script downloaded from the portal. The appliances which have already been registered will continue seeing the older version of the user experience and will continue to work without any issues.

- For the appliance to run auto-update, paste the project key that you copied from the portal. If you don't have the key, go to **Azure Migrate: Discovery and assessment > Overview > Manage existing appliances**. Select the appliance name you provided when you generated the project key, and then copy the key that's shown.
- The appliance will verify the key and start the auto-update service, which updates all the services on the appliance to their latest versions. When the auto-update has run, you can select **View appliance services** to see the status and versions of the services running on the appliance server.
- To register the appliance, you need to select **Login**. In **Continue with Azure Login**, select **Copy code & Login** to copy the device code (you must have a device code to authenticate

with Azure) and open an Azure Login prompt in a new browser tab. Make sure you've disabled the pop-up blocker in the browser to see the prompt.



- d. In a new tab in your browser, paste the device code and sign in by using your Azure username and password. Signing in with a PIN isn't supported.

! Note

If you close the sign in tab accidentally without signing in, refresh the browser tab of the appliance configuration manager to display the device code and Copy code & Login button.

- e. After you successfully sign in, return to the browser tab that displays the appliance configuration manager. If the Azure user account that you used to sign in has the required permissions for the Azure resources that were created during key generation, appliance registration starts.

After the appliance is successfully registered, to see the registration details, select **View details**.

You can *rerun prerequisites* at any time during appliance configuration to check whether the appliance meets all the prerequisites.

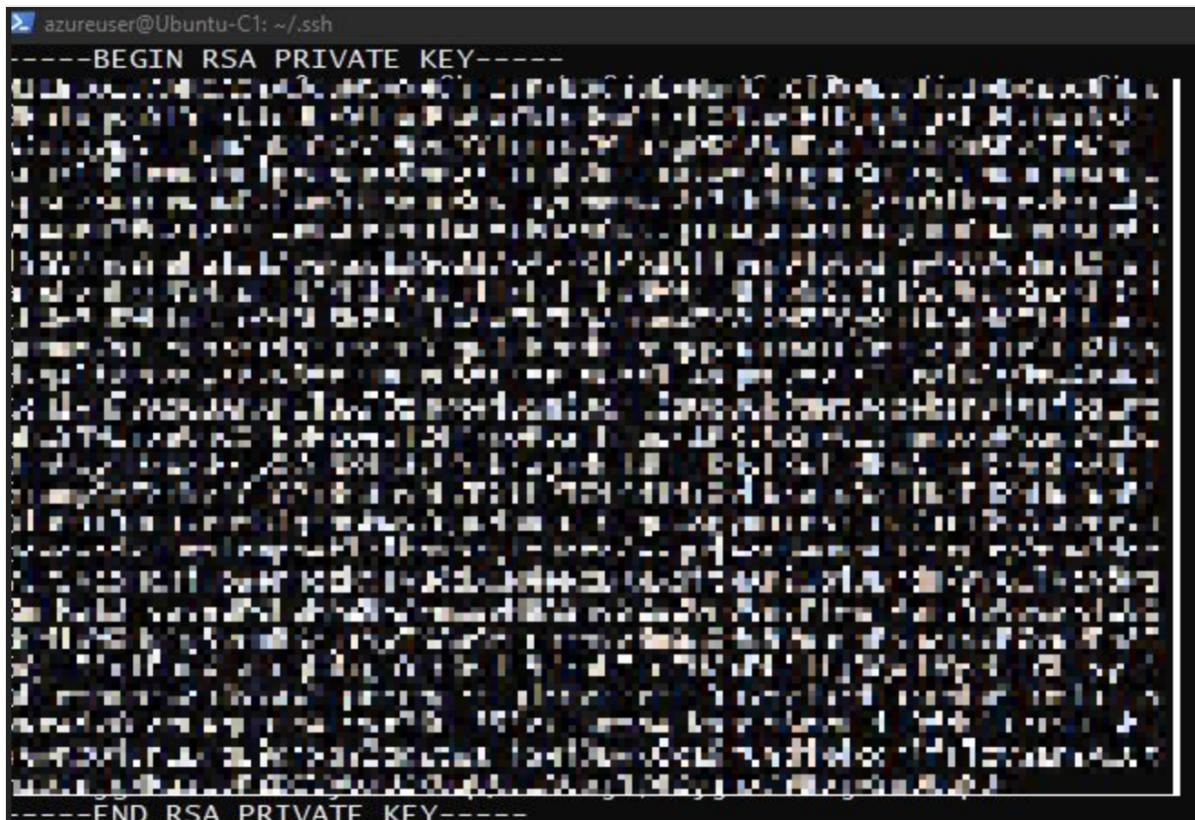
Start continuous discovery

Now, connect from the appliance to the physical servers to be discovered, and start the discovery.

1. In **Step 1: Provide credentials for discovery of Windows and Linux physical or virtual servers**, select **Add credentials**.
2. For Windows server, select the source type as **Windows Server**, specify a friendly name for credentials, add the username and password. Select **Save**.
3. If you are using password-based authentication for Linux server, select the source type as **Linux Server (Password-based)**, specify a friendly name for credentials, add the username and password. Select **Save**.

4. If you are using SSH key-based authentication for Linux server, you can select source type as **Linux Server (SSH key-based)**, specify a friendly name for credentials, add the username, browse, and select the SSH private key file. Select **Save**.

- Azure Migrate supports the SSH private key generated by ssh-keygen command using RSA, DSA, ECDSA, and ed25519 algorithms.
- Currently Azure Migrate does not support passphrase-based SSH key. Use an SSH key without a passphrase.
- Currently Azure Migrate does not support SSH private key file generated by PuTTY.
- Azure Migrate supports OpenSSH format of the SSH private key file as shown below:



```
azureuser@Ubuntu-C1: ~/.ssh
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----
```

5. If you want to add multiple credentials at once, select **Add more** to save and add more credentials. Multiple credentials are supported for physical servers discovery.

 **Note**

By default, the credentials will be used to gather data about the installed applications, roles, and features, and also to collect dependency data from Windows and Linux servers, unless you disable the slider to not perform these features (as instructed in the last step).

6. In **Step 2:Provide physical or virtual server details**, select **Add discovery source** to specify the server IP address/FQDN and the friendly name for credentials to connect to the server.

7. You can either **Add single item** at a time or **Add multiple items** in one go. There is also an option to provide server details through **Import CSV**.

- If you choose **Add single item**, you can choose the OS type, specify friendly name for credentials, add server IP address/FQDN, and select **Save**.

- If you choose **Add multiple items**, you can add multiple records at once by specifying server **IP address/FQDN** with the friendly name for credentials in the text box. Verify** the added records and select **Save**.
- If you choose **Import CSV (selected by default)**, you can download a CSV template file, populate the file with the server **IP address/FQDN** and friendly name for credentials. You then import the file into the appliance, **verify** the records in the file and select **Save**.

8. On selecting **Save**, appliance will try validating the connection to the servers added and show the **Validation status** in the table against each server.

- If validation fails for a server, review the error by selecting **Validation failed** in the Status column of the table. Fix the issue, and validate again.
- To remove a server, select **Delete**.

9. You can **revalidate** the connectivity to servers anytime before starting the discovery.

10. Before initiating discovery, you can choose to disable the slider to not perform software inventory and agentless dependency analysis on the added servers. You can change this option at any time.

Note: After server discovery, appliance will automatically initiate software inventory and validate prerequisites for agentless dependency analysis on discovered servers.

[Disable the slider if you don't want to perform these features \(you can change this later\)](#)

Click on the button below to initiate discovery. After the discovery is complete, you can check the discovery status of the physical or virtual servers in the table above. [Learn more](#) about the metadata collected during discovery.

[Start discovery](#)

11. To perform discovery of SQL Server instances and databases, you can add additional credentials (Windows domain/non-domain, SQL authentication credentials) and the appliance will attempt to automatically map the credentials to the SQL servers. If you add domain credentials, the appliance will authenticate the credentials against Active Directory of the domain to prevent any user accounts from locking out. To check validation of the domain credentials, follow these steps:

- In the configuration manager credentials table, see **Validation status** for domain credentials. Only the domain credentials are validated.
- If validation fails, you can select a Failed status to see the validation error. Fix the issue, and then select **Revalidate credentials** to reattempt validation of the credentials.

Start discovery

Select **Start discovery**, to kick off discovery of the successfully validated servers. After the discovery has been successfully initiated, you can check the discovery status against each server in the table.

How discovery works

- It takes approximately 2 minutes to complete discovery of 100 servers and their metadata to appear in the Azure portal.
- [Software inventory](#) (discovery of installed applications) is automatically initiated when the discovery of servers is finished.
- [Software inventory](#) identifies the SQL Server instances that are running on the servers. Using the information it collects, the appliance attempts to connect to the SQL Server instances through the Windows authentication credentials or the SQL Server authentication credentials that are provided on the appliance. Then, it gathers data on SQL Server databases and their properties. The SQL Server discovery is performed once every 24 hours.
- Appliance can connect to only those SQL Server instances to which it has network line of sight, whereas software inventory by itself might not need network line of sight.
- The time taken for discovery of installed applications depends on the number of discovered servers. For 500 servers, it takes approximately one hour for the discovered inventory to appear in the Azure Migrate project in the portal.
- During software inventory, the added server credentials are iterated against servers and validated for agentless dependency analysis. When the discovery of servers is finished, in the portal, you can enable agentless dependency analysis on the servers. Only the servers on which validation succeeds can be selected to enable [agentless dependency analysis](#).
- SQL Server instances and databases data begin to appear in the portal within 24 hours after you start discovery.
- By default, Azure Migrate uses the most secure way of connecting to SQL instances that is, Azure Migrate encrypts communication between the Azure Migrate appliance and the source SQL Server instances by setting the TrustServerCertificate property to `true`. Additionally, the transport layer uses SSL to encrypt the channel and bypass the certificate chain to validate trust. Hence, the appliance server must be set up to trust the certificate's root authority. However, you can modify the connection settings, by selecting [Edit SQL Server connection properties](#) on the appliance. [Learn more](#) to understand what to choose.

Step 3: Provide server credentials to perform software inventory, agentless dependency analysis, discovery of SQL Server instances and databases and discovery of ASP.NET web apps in your VMware environment.

I don't need to perform these features (you can change this later)

You can add multiple credentials and the appliance will attempt to automatically map the credentials to the servers.

If you add domain credentials, appliance will authenticate the credentials against Active Directory of the domain to prevent any user accounts from locking out.

[Learn more](#) about how to provide credentials and how we handle them.

[Add credentials](#)

#	Credentials type	Friendly name	Username	Status	Edit
1		SQL Server connection properties			Edit
2		Connection properties: Help me choose			Edit
3		<input checked="" type="checkbox"/> Encrypt connection			Edit
4		<input checked="" type="checkbox"/> Trust Server certificate			Edit
5					Edit

You can revalidate the added domain credentials by clicking on the button below.

[Revalidate credentials](#)

You can edit SQL Server connection properties by clicking on button below. [Learn more](#)

[Edit properties](#)

Verify servers in the portal

After discovery finishes, you can verify that the servers appear in the portal.

1. Open the Azure Migrate dashboard.
2. In **Servers, databases and web apps > Azure Migrate: Discovery and assessment** page, select the icon that displays the count for **Discovered servers**.

Next steps

- [Assess physical servers](#) for migration to Azure VMs.
- [Review the data](#) that the appliance collects during discovery.

Tutorial: Assess AWS instances for migration to Azure

Article • 03/14/2023

As part of your migration journey to Azure, you assess your on-premises workloads to measure cloud readiness, identify risks, and estimate costs and complexity.

This article shows you how to assess Amazon Web Services (AWS) instances for migration to Azure, using the Azure Migrate: Discovery and assessment tool.

In this tutorial, you learn how to:

- Run an assessment based on server metadata and configuration information.
- Run an assessment based on performance data.

ⓘ Note

Tutorials show the quickest path for trying out a scenario, and use default options where possible.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

- Before you follow the steps in this tutorial, complete the first tutorial in this series to [discover your on-premises inventory](#).
- Make sure AWS instances aren't running Windows Server 2003, or SUSE Linux. Assessment isn't supported for these servers.

Decide which assessment to run

Decide whether you want to run an assessment using sizing criteria based on server configuration data/metadata that's collected as-is on-premises, or on dynamic performance data.

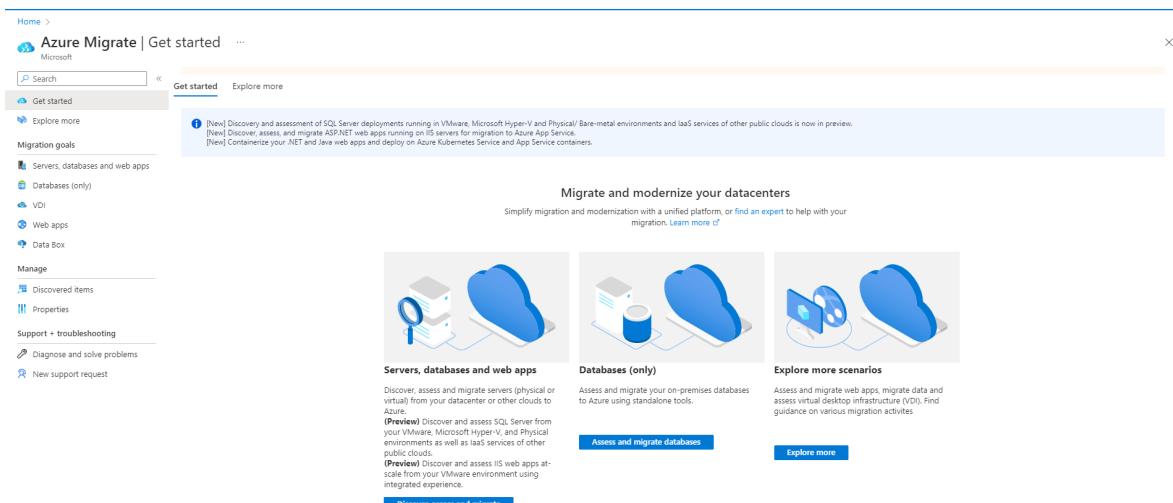
Assessment	Details	Recommendation
------------	---------	----------------

Assessment	Details	Recommendation
As-is on-premises	Assess based on server configuration data/metadata.	<p>Recommended Azure VM size is based on the on-premises VM size.</p> <p>The recommended Azure disk type is based on what you select in the storage type setting in the assessment.</p>
Performance-based	Assess based on collected dynamic performance data.	<p>Recommended Azure VM size is based on CPU and memory utilization data.</p> <p>The recommended disk type is based on the IOPS and throughput of the on-premises disks.</p>

Run an assessment

Run an assessment as follows:

1. On the **Get started** page > **Servers, databases and web apps**, select **Discover, assess and migrate**.



2. In Azure Migrate: Discovery and assessment, select **Assess > Azure VM**.

3. In Assess servers > Assessment type > Azure VM.

4. In Discovery source:

- If you discovered servers using the appliance, select **Servers discovered from Azure Migrate appliance**.
- If you discovered servers using an imported CSV file, select **Imported servers**.

5. Select **Edit** to review the assessment properties.

6. In Assessment properties > Target Properties:

- In **Target location**, specify the Azure region to which you want to migrate.

- Size and cost recommendations are based on the location that you specify. Once you change the target location from default, you'll be prompted to specify **Reserved Instances** and **VM series**.
- In Azure Government, you can target assessments in [these regions](#)
- In **Storage type**,
 - If you want to use performance-based data in the assessment, select **Automatic** for Azure Migrate to recommend a storage type, based on disk IOPS and throughput.
 - Alternatively, select the storage type you want to use for VM when you migrate it.
- In **Savings options (compute)**, specify the savings option that you want the assessment to consider, to help optimize your Azure compute cost.
 - [Azure reservations](#) (1 year or 3 year reserved) are a good option for the most consistently running resources.
 - [Azure Savings Plan](#) (1 year or 3 year savings plan) provide additional flexibility and automated cost optimization. Ideally post migration, you could use Azure reservation and savings plan at the same time (reservation will be consumed first), but in the Azure Migrate assessments, you can only see cost estimates of 1 savings option at a time.
 - When you select 'None', the Azure compute cost is based on the Pay as you go rate or based on actual usage.
 - You need to select pay-as-you-go in offer/licensing program to be able to use Reserved Instances or Azure Savings Plan. When you select any savings option other than 'None', the 'Discount (%)' and 'VM uptime' properties aren't applicable.

7. In VM Size:

- In **Sizing criterion**, select if you want to base the assessment on server configuration data/metadata, or on performance-based data. If you use performance data:
 - In **Performance history**, indicate the data duration on which you want to base the assessment
 - In **Percentile utilization**, specify the percentile value you want to use for the performance sample.
- In **VM Series**, specify the Azure VM series you want to consider.
 - If you're using performance-based assessment, Azure Migrate suggests a value for you.
 - Tweak settings as needed. For example, if you don't have a production environment that needs A-series VMs in Azure, you can exclude A-series from the list of series.

- In **Comfort factor**, indicate the buffer you want to use during assessment. This accounts for issues like seasonal usage, short performance history, and likely increases in future usage. For example, if you use a comfort factor of two:

Component	Effective utilization	Add comfort factor (2.0)
Cores	2	4
Memory	8 GB	16 GB

8. In **Pricing**:

- In **Offer**, specify the [Azure offer](#) if you're enrolled. The assessment estimates the cost for that offer.
- In **Currency**, select the billing currency for your account.
- In **Discount (%)**, add any subscription-specific discounts you receive on top of the Azure offer. The default setting is 0%.
- In **VM Uptime**, specify the duration (days per month/hour per day) that VMs will run.
 - This is useful for Azure VMs that won't run continuously.
 - Cost estimates are based on the duration specified.
 - Default is 31 days per month/24 hours per day.
- In **EA Subscription**, specify whether to take an Enterprise Agreement (EA) subscription discount into account for cost estimation.
- In **Azure Hybrid Benefit**, specify whether you already have a Windows Server license. If you do and they're covered with active Software Assurance or Windows Server Subscriptions, you can apply for the [Azure Hybrid Benefit](#) when you bring licenses to Azure.

9. Select **Save** if you make changes.

10. In **Assess Servers**, select **Next**.

11. In **Select servers to assess > Assessment name**, specify a name for the assessment.

12. In **Select or create a group**, select **Create New** and specify a group name.

13. Select the appliance, and select the VMs you want to add to the group. Then select **Next**.

14. In **Review + create assessment**, review the assessment details and select **Create Assessment** to create the group and run the assessment.

15. After the assessment is created, view it in **Servers, databases and web apps > Azure Migrate: Discovery and assessment > Assessments**.

16. Select **Export assessment** to download it as an Excel file.

! Note

For performance-based assessments, we recommend that you wait at least a day after starting discovery before you create an assessment. This provides time to collect performance data with higher confidence. Ideally, after you start discovery, wait for the performance duration you specify (day/week/month) for a high-confidence rating.

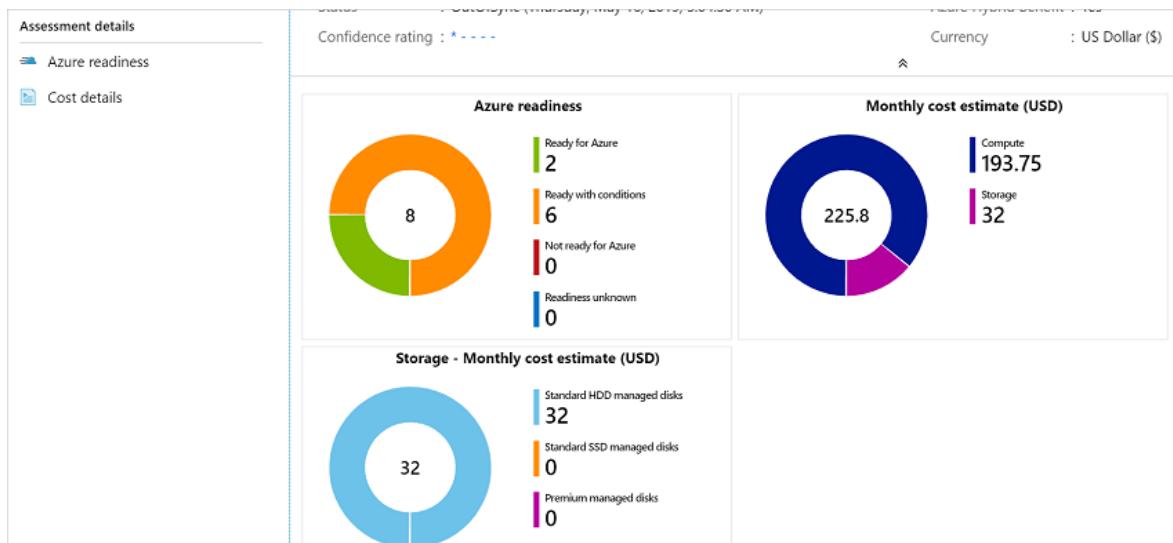
Review an assessment

An assessment describes:

- **Azure readiness:** Whether VMs are suitable for migration to Azure.
- **Monthly cost estimation:** The estimated monthly compute and storage costs for running the VMs in Azure.
- **Monthly storage cost estimation:** Estimated costs for disk storage after migration.

To view an assessment:

1. In **Servers, databases and web apps > Azure Migrate: Discovery and assessment**, select the number next to **Assessments**.
2. In **Assessments**, select an assessment to open it. As an example (estimations and costs, for example, only):



3. Review the assessment summary. You can also edit the assessment properties, or recalculate the assessment.

Review readiness

1. Select **Azure readiness**.

2. In **Azure readiness**, review the VM status:

- **Ready for Azure**: Used when Azure Migrate recommends a VM size and cost estimates, for VMs in the assessment.
- **Ready with conditions**: Shows issues and suggested remediation.
- **Not ready for Azure**: Shows issues and suggested remediation.
- **Readiness unknown**: Used when Azure Migrate can't assess readiness, because of data availability issues.

3. Select an **Azure readiness** status. You can view VM readiness details. You can also drill down to see VM details, including compute, storage, and network settings.

Review cost estimates

The assessment summary shows the estimated compute and storage cost of running VMs in Azure.

1. Review the monthly total costs. Costs are aggregated for all VMs in the assessed group.

- Cost estimates are based on the size recommendations for a server, its disks, and its properties.
- Estimated monthly costs for compute and storage are shown.
- The cost estimation is for running the on-premises VMs on Azure VMs. The estimation doesn't consider PaaS or SaaS costs.

2. Review monthly storage costs. The view shows the aggregated storage costs for the assessed group, split over different types of storage disks.

3. You can drill down to see cost details for specific VMs.

Review confidence rating

Azure Migrate assigns a confidence rating to performance-based assessments. Rating is from one star (lowest) to five stars (highest).

Assess servers		Columns							
		Search to filter assessments							
NAME	GROUP	STATUS	MACHINES	LOCATION	SIZING CRITERION	CONFIDENCE RATING			
assessment_5_16_2019_17_34_29	day2	OutOfSync	0	North Europe	Performance-based	★★★★	★	★	★
assessment_5_20_2019_17_42_6	Mygroup	Ready	6	North Europe	Performance-based	★★★★	★	★	★
assessment_5_22_2019_22_56_13	Day2-group	OutDated	14	North Europe	Performance-based	★★★★	★	★	★

The confidence rating helps you estimate the reliability of size recommendations in the assessment. The rating is based on the availability of data points needed to compute the assessment.

Note

Confidence ratings aren't assigned if you create an assessment based on a CSV file.

Confidence ratings are as follows.

Data point availability	Confidence rating
0%-20%	1 star
21%-40%	2 stars
41%-60%	3 stars
61%-80%	4 stars
81%-100%	5 stars

[Learn more about confidence ratings.](#)

Next steps

- Find server dependencies using [dependency mapping](#).
- Set up [agent-based](#) dependency mapping.

Discover, assess, and migrate Amazon Web Services (AWS) VMs to Azure

Article • 10/27/2023

This tutorial shows you how to discover, assess, and migrate Amazon Web Services (AWS) virtual machines (VMs) to Azure VMs, using Azure Migrate: Server Assessment and Migration and modernization tools.

ⓘ Note

You migrate AWS VMs to Azure by treating them as physical servers.

In this tutorial, you'll learn how to:

- ✓ Verify prerequisites for migration.
- ✓ Prepare Azure resources with the Migration and modernization tool. Set up permissions for your Azure account and resources to work with Azure Migrate.
- ✓ Prepare AWS EC2 instances for migration.
- ✓ Add the Migration and modernization tool in the Azure Migrate hub.
- ✓ Set up the replication appliance and deploy the configuration server.
- ✓ Install the Mobility service on AWS VMs you want to migrate.
- ✓ Enable replication for VMs.
- ✓ Track and monitor the replication status.
- ✓ Run a test migration to make sure everything's working as expected.
- ✓ Run a full migration to Azure.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Discover and assess

Before you migrate to Azure, we recommend that you perform a VM discovery and migration assessment. This assessment helps right-size your AWS VMs for migration to Azure, and estimate potential Azure run costs.

Set up an assessment as follows:

1. Follow the [tutorial](#) to set up Azure and prepare your AWS VMs for an assessment.
Note that:

- Azure Migrate uses password authentication when discovering AWS instances. AWS instances don't support password authentication by default. Before you can discover instance, you need to enable password authentication.
 - For Windows machines, allow WinRM port 5985 (HTTP). This allows remote WMI calls.
 - For Linux machines:
 - a. Sign into each Linux machine.
 - b. Open the `sshd_config` file : `vi /etc/ssh/sshd_config`
 - c. In the file, locate the **PasswordAuthentication** line, and change the value to **yes**.
 - d. Save the file and close it. Restart the ssh service.
- If you're using a root user to discover your Linux VMs, ensure root login is allowed on the VMs.
 - a. Sign into each Linux machine
 - b. Open the `sshd_config` file : `vi /etc/ssh/sshd_config`
 - c. In the file, locate the **PermitRootLogin** line, and change the value to **yes**.
 - d. Save the file and close it. Restart the ssh service.

2. Then, follow this [tutorial](#) to set up an Azure Migrate project and appliance to discover and assess your AWS VMs.

Although we recommend that you try out an assessment, performing an assessment isn't a mandatory step to be able to migrate VMs.

Prerequisites

- Ensure that the AWS VMs you want to migrate are running a supported OS version. AWS VMs are treated like physical machines for the purpose of the migration. Review the [supported operating systems and kernel versions](#) for the physical server migration workflow. You can use standard commands like `hostnamectl` or `uname -a` to check the OS and kernel versions for your Linux VMs. We recommend you perform a test migration (test failover) to validate if the VM works as expected before proceeding with the actual migration.
- Make sure your AWS VMs comply with the [supported configurations](#) for migration to Azure.
- Verify that the AWS VMs that you replicate to Azure comply with [Azure VM requirements](#).
- There are some changes needed on the VMs before you migrate them to Azure.
 - For some operating systems, Azure Migrate makes these changes automatically.

- It's important to make these changes before you begin migration. If you migrate the VM before you make the change, the VM might not boot up in Azure. Review [Windows](#) and [Linux](#) changes you need to make.

Prepare Azure resources for migration

Prepare Azure for migration with the Migration and modernization tool.

[Expand table](#)

Task	Details
Create an Azure Migrate project	Your Azure account needs Contributor or Owner permissions to create a new project .
Verify permissions for your Azure account	Your Azure account needs permissions to create a VM, and write to an Azure managed disk.

Assign permissions to create project

1. In the Azure portal, open the subscription, and select **Access control (IAM)**.
2. In **Check access**, find the relevant account, and click it to view permissions.
3. You should have **Contributor** or **Owner** permissions.
 - If you just created a free Azure account, you're the owner of your subscription.
 - If you're not the subscription owner, work with the owner to assign the role.

Assign Azure account permissions

Assign the Virtual Machine Contributor role to the Azure account. This provides permissions to:

- Create a VM in the selected resource group.
- Create a VM in the selected virtual network.
- Write to an Azure managed disk.

Create an Azure network

[Set up](#) an Azure virtual network (VNet). When you replicate to Azure, the Azure VMs that are created are joined to the Azure VNet that you specify when you set up migration.

Prepare AWS instances for migration

To prepare for AWS to Azure migration, you need to prepare and deploy a replication appliance for migration.

Prepare a machine for the replication appliance

The Migration and modernization tool uses a replication appliance to replicate machines to Azure. The replication appliance runs the following components.

- **Configuration server:** The configuration server coordinates communications between the AWS environment and Azure, and manages data replication.
- **Process server:** The process server acts as a replication gateway. It receives replication data, optimizes it with caching, compression, and encryption, and sends it to a cache storage account in Azure.

Prepare for appliance deployment as follows:

- Set up a separate EC2 VM to host the replication appliance. This instance must be running Windows Server 2012 R2 or Windows Server 2016. [Review](#) the hardware, software, and networking requirements for the appliance.
- The appliance shouldn't be installed on a source VM that you want to replicate or on the Azure Migrate discovery and assessment appliance you may have installed before. It should be deployed on a different VM.
- The source AWS VMs to be migrated should have a network line of sight to the replication appliance. Configure necessary security group rules to enable this. It's recommended that the replication appliance is deployed in the same VPC as the source VMs to be migrated. If the replication appliance needs to be in a different VPC, the VPCs need to be connected through VPC peering.
- The source AWS VMs communicate with the replication appliance on ports HTTPS 443 (control channel orchestration) and TCP 9443 (data transport) inbound for replication management and replication data transfer. The replication appliance in turn orchestrates and sends replication data to Azure over port HTTPS 443 outbound. To configure these rules, edit the security group inbound/outbound rules with the appropriate ports and source IP information.

The screenshot shows the AWS VPC Security Groups list and the details page for a specific security group. The list table has columns: Security group ID, Security group name, VPC ID, Description, Owner, and Inbound rules count. The details page shows the security group's name, ID, description, owner, and rule counts. It also lists inbound rules, each with columns: Type, Protocol, Port range, Source, and Description - optional. The 'Edit inbound rules' button is highlighted with a red box.

Security Groups (1/2) Info					
Filter security groups				Actions ▼	
search: sg-b0cb2edb X		Clear filters		Create security group	
Security group ID	Security group name	VPC ID	Description	Owner	Inb
<input checked="" type="checkbox"/> sg-b0cb2edb	launch-wizard-2	vpc-0512e36d Edit	launch-wizard-2 create... 409655286455	4 P	
<input type="checkbox"/> sg-cfff1fa4	launch-wizard-4	vpc-0512e36d Edit	launch-wizard-4 create... 409655286455	2 P	

sg-b0cb2edb - launch-wizard-2

Details			
Security group name launch-wizard-2	Security group ID sg-b0cb2edb	Description launch-wizard-2 created 2018-01-05T15:20:26.169+05:30	VPC ID vpc-0512e36d Edit
Owner 409655286455	Inbound rules count 4 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#) [Outbound rules](#) [Tags](#)

Inbound rules

Type	Protocol	Port range	Source	Description - optional
All traffic	All	All	sg-cfff1fa4 (launch-wizard-4)	-
Custom TCP	TCP	9443	0.0.0.0/0	-
RDP	TCP	3389	0.0.0.0/0	-
HTTPS	TCP	443	0.0.0.0/0	-

[Edit inbound rules](#)

- The replication appliance uses MySQL. Review the [options](#) for installing MySQL on the appliance.
- Review the Azure URLs required for the replication appliance to access [public](#) and [government](#) clouds.

Set up the replication appliance

The first step of migration is to set up the replication appliance. To set up the appliance for AWS VMs migration, you must download the installer file for the appliance, and then run it on the [VM you prepared](#).

Download the replication appliance installer

1. In the Azure Migrate project > **Servers, databases and web apps**, in **Migration and modernization**, select **Discover**.

 **Azure Migrate - Servers**
Microsoft

«

 Overview

 Migration goals

 Servers

 Databases

 Data Box

 Manage

 Discovered items

 Support + troubleshooting

 New support request

 Refresh

Last refreshed at: 6/20/2019, 8:28:41 PM (Click on "Refresh" to update the

Assessment tools

 **Azure Migrate: Server Assessment**

 Discover  Assess  Overview

Quick start

1: Discover
Click 'Discover' to start discovering your on-premises machines.

2: Assess
Once your on-premises machines are discovered, click "Assess" to create assessments.

Add more assessment tools? [Click here.](#)

Migration tools

 **Azure Migrate: Server Migration**

 Discover  Replicate  Migrate  Overview

Quick start

1: Discover
Click 'Discover' to start discovering your on-premises machines.

2: Replicate
Once your on-premises machines are discovered, click "Replicate" to start replicating the discovered machines.

3: Migrate
Once your machines have replicated, click "Migrate" to migrate your machines.

Add more migration tools? [Click here.](#)

2. In **Discover machines** > **Are your machines virtualized?**, click **Not virtualized/Other**.
3. In **Target region**, select the Azure region to which you want to migrate the machines.
4. Select **Confirm that the target region for migration is <region-name>**.
5. Click **Create resources**. This creates an Azure Site Recovery vault in the background.

- If you've already set up migration with the Migration and modernization tool, the target option can't be configured, since resources were set up previously.
- You can't change the target region for this project after clicking this button.
- To migrate your VMs to a different region, you'll need to create a new/different Azure Migrate project.

! Note

If you selected private endpoint as the connectivity method for the Azure Migrate project when it was created, the Recovery Services vault will also be configured for private endpoint connectivity. Ensure that the private endpoints are reachable from the replication appliance. [Learn more](#)

6. In **Do you want to install a new replication appliance?**, select **Install a replication appliance**.
7. In **Download and install the replication appliance software**, download the appliance installer, and the registration key. You need to the key in order to register the appliance. The key is valid for five days after it's downloaded.

Discover machines

Are your machines virtualized? [!](#)
Not virtualized / Other

Target region [!](#)
(US) Central US

Do you want to install a new replication appliance or scale-out existing setup?
Install a replication appliance [Help me choose](#)

The replication appliance (Configuration Server) is a virtual appliance that is deployed on-premises. The replication appliance coordinates and manages replication for the servers that are being migrated. Follow the steps outlined below to set up and configure the replication appliance

1. Download and install the replication appliance software.
Create a new Windows Server 2016 machine by following the [Configuration Server sizing guidelines](#).
[Download](#) the replication appliance software installer and use it to complete installation of the replication appliance software on the newly created Windows Server 2016 machine.

2. Configure the replication appliance and register it to the Azure Migrate project
Download the registration key file and use it to register the replication appliance to this project. The replication appliance installer will ask for a registration key.
[Download](#)

8. Copy the appliance setup file and key file to the Windows Server 2016 or Windows Server 2012 AWS VM you created for the replication appliance.
9. Run the replication appliance setup file, as described in the next procedure.
 - a. Under **Before You Begin**, select **Install the configuration server and process server**, and then select **Next**.

- b. In **Third-Party Software License**, select **I accept the third-party license agreement**, and then select **Next**.
 - c. In **Registration**, select **Browse**, and then go to where you put the vault registration key file. Select **Next**.
 - d. In **Internet Settings**, select **Connect to Azure Site Recovery without a proxy server**, and then select **Next**.
 - e. The **Prerequisites Check** page runs checks for several items. When it's finished, select **Next**.
 - f. In **MySQL Configuration**, provide a password for the MySQL DB, and then select **Next**.
 - g. In **Environment Details**, select **No**. You don't need to protect your VMs. Then, select **Next**.
 - h. In **Install Location**, select **Next** to accept the default.
 - i. In **Network Selection**, select **Next** to accept the default.
 - j. In **Summary**, select **Install**.
 - k. **Installation Progress** shows you information about the installation process. When it's finished, select **Finish**. A window displays a message about a reboot. Select **OK**.
 - l. Next, a window displays a message about the configuration server connection passphrase. Copy the passphrase to your clipboard and save the passphrase in a temporary text file on the source VMs. You'll need this passphrase later, during the mobility service installation process.
10. After the installation completes, the Appliance configuration wizard will be launched automatically (You can also launch the wizard manually by using the **cspconfigtool** shortcut that is created on the desktop of the appliance). In this tutorial, we'll be manually installing the Mobility Service on source VMs to be replicated, so create a dummy account in this step and proceed. You can provide the following details for creating the dummy account - "guest" as the friendly name, "username" as the username, and "password" as the password for the account. You'll be using this dummy account in the Enable Replication stage.
11. After the appliance has restarted after setup, in **Discover machines**, select the new appliance in **Select Configuration Server**, and click **Finalize registration**. Finalize registration performs a couple of final tasks to prepare the replication appliance.

 **3. Finalize registration**
Prepare for replication by finalizing registration for the replication appliance (Configuration Server). Select the replication appliance from the drop down to finalize registration for it.

* Select Configuration Server  Registration finalized

Install the Mobility service agent

A Mobility service agent must be pre-installed on the source AWS VMs to be migrated before you can initiate replication. The approach you choose to install the Mobility service agent may depend on your organization's preferences and existing tools, but be aware that the "push" installation method built into Azure Site Recovery is not currently supported. Approaches you may want to consider:

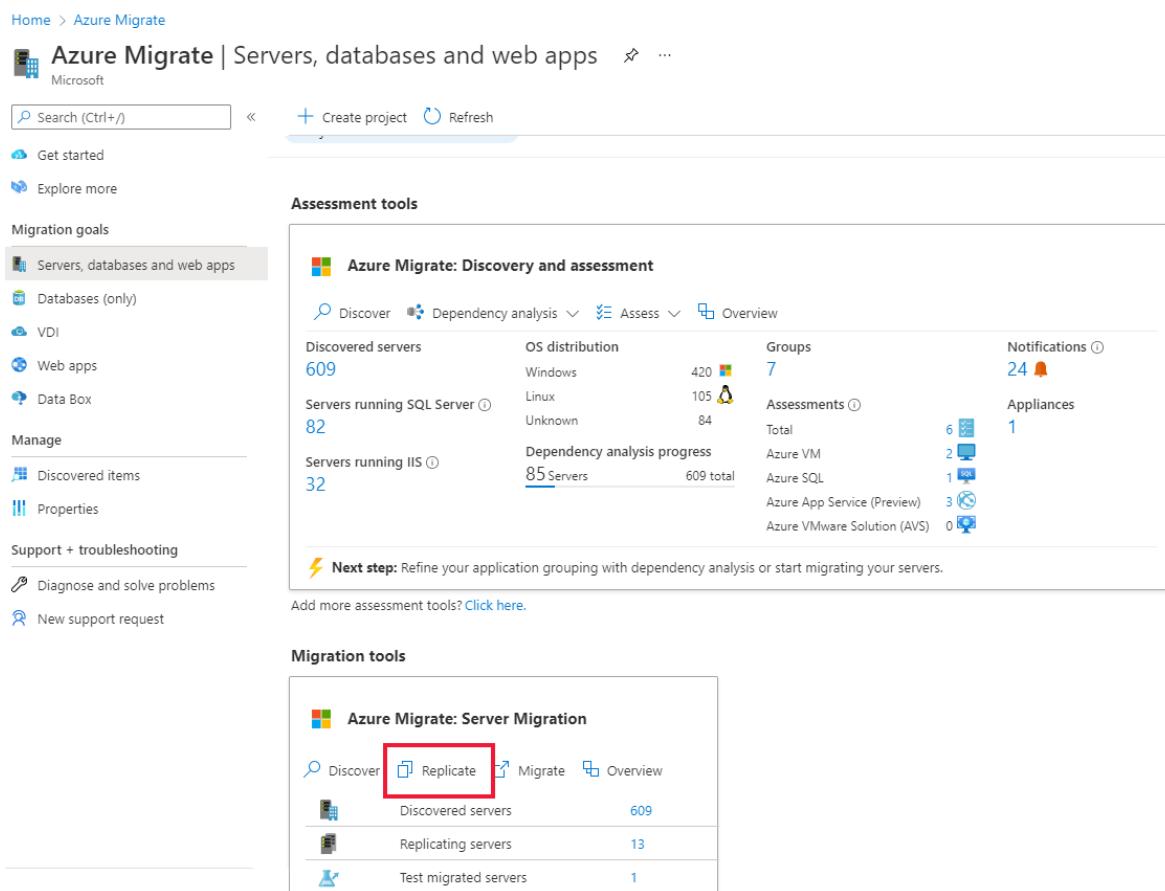
- [AWS System Manager ↗](#)
- [System Center Configuration Manager](#)
- [Arc for Servers and Custom Script Extensions](#)
- [Manual installation](#)

Enable replication for AWS VMs

ⓘ Note

Through the portal, you can add up to 10 VMs for replication at once. To replicate more VMs simultaneously, you can add them in batches of 10.

1. In the Azure Migrate project > **Servers, databases and web apps** > **Migration and modernization**, select **Replicate**.



The screenshot shows the Azure Migrate portal interface. The top navigation bar includes 'Home > Azure Migrate', a search bar, and a 'Create project' button. The main content area is divided into 'Assessment tools' and 'Migration tools' sections.

Assessment tools section:

- Azure Migrate: Discovery and assessment** card:
 - Discover, Dependency analysis, Assess, Overview buttons.
 - Discover: 609 servers.
 - Dependency analysis: 420 Windows, 105 Linux, 84 Unknown.
 - Assessments: 7 groups.
 - Notifications: 24.
- Next step: Refine your application grouping with dependency analysis or start migrating your servers.

Migration tools section:

- Azure Migrate: Server Migration** card:
 - Discover, Replicate, Migrate, Overview buttons.
 - Discover: 609 servers.
 - Replicate: 13 servers.
 - Migrate: 1 server.

A red box highlights the 'Replicate' button in the 'Migration tools' section.

2. In **Replicate**, > **Source settings** > **Are your machines virtualized?**, select **Not virtualized/Other**.
3. In **On-premises appliance**, select the name of the Azure Migrate appliance that you set up.
4. In **Process Server**, select the name of the replication appliance.
5. In **Guest credentials**, please select the dummy account created previously during the [replication installer setup](#) to install the Mobility service manually (push install is not supported). Then click **Next: Virtual machines**.

Replicate

The first step in migrating servers is to replicate them. Once replication completes, you can perform test migration before finally moving the servers to Azure.

Are your machines virtualized? *

Physical or other (AWS, GCP, Xen, etc.)

On-premises appliance *

CONTOSO-0122 (Azure Migrate Appliance)

6. In **Virtual Machines**, in **Import migration settings from an assessment?**, leave the default setting **No, I'll specify the migration settings manually**.
7. Check each VM you want to migrate. Then click **Next: Target settings**.

Name	Azure VM Readiness	IP Address	Operating System	Boot Type	Source
ContosoSales	Ready	2404:f801:4800:1c:2d77:36ade95b:35d7...	Microsoft Windows Server 2016 or later ...	bios	idclab-vcen67.fareast.corp.microsoft.com
ContosoHR-SQLDB	Ready	2404:f801:4800:25:c823:79ff:6a48:2666:1...	Microsoft Windows Server 2016 or later ...	bios	idclab-vcen67.fareast.corp.microsoft.com
CONTOSO-WEBINAR-...	Ready	2404:f801:4800:25:c823:79ff:6a48:2666:1...	Microsoft Windows Server 2016 or later ...	bios	idclab-vcen67.fareast.corp.microsoft.com

8. In **Target settings**, select the subscription, and target region to which you'll migrate, and specify the resource group in which the Azure VMs will reside after migration.
9. In **Virtual Network**, select the Azure VNet/subnet to which the Azure VMs will be joined after migration.

10. In **Cache storage account**, keep the default option to use the cache storage account that is automatically created for the project. Use the dropdown if you'd like to specify a different storage account to use as the cache storage account for replication.

 **Note**

- If you selected private endpoint as the connectivity method for the Azure Migrate project, grant the Recovery Services vault access to the cache storage account. [Learn more](#)
- To replicate using ExpressRoute with private peering, create a private endpoint for the cache storage account. [Learn more](#)

11. In **Availability options**, select:

- Availability Zone to pin the migrated machine to a specific Availability Zone in the region. Use this option to distribute servers that form a multi-node application tier across Availability Zones. If you select this option, you'll need to specify the Availability Zone to use for each of the selected machine in the Compute tab. This option is only available if the target region selected for the migration supports Availability Zones
- Availability Set to place the migrated machine in an Availability Set. The target Resource Group that was selected must have one or more availability sets in order to use this option.
- No infrastructure redundancy required option if you don't need either of these availability configurations for the migrated machines.

12. In **Disk encryption type**, select:

- Encryption-at-rest with platform-managed key
- Encryption-at-rest with customer-managed key
- Double encryption with platform-managed and customer-managed keys

 **Note**

To replicate VMs with CMK, you'll need to [create a disk encryption set](#) under the target Resource Group. A disk encryption set object maps Managed Disks to a Key Vault that contains the CMK to use for SSE.

13. In **Azure Hybrid Benefit**:

- Select **No** if you don't want to apply Azure Hybrid Benefit. Then click **Next**.
- Select **Yes** if you have Windows Server machines that are covered with active Software Assurance or Windows Server subscriptions, and you want to apply the benefit to the machines you're migrating. Then click **Next**.

Replicate

Configure settings and target properties for migration.

Region *	<input type="text" value="East US"/>
Storage account *	<input type="text" value="(new) migratersa899899444"/>
Subscription *	<input type="text" value="Azure Migrate Test2"/>
Resource group *	<input type="text" value="Select a resource group"/>

VMs on this subscription, wherein SQL Server is running, are being onboarded to the SQL IaaS extension. You can choose to disable the same by following steps listed [here](#)

Register with SQL IaaS extension

Apply Azure Hybrid with an eligible Windows Server license and save up to 49% vs. pay-as-you-go virtual machine costs.

I have a Windows Server license

Virtual network *

<input type="text" value="Select a network"/>

Subnet *

<input type="text" value="Select a subnet"/>
--

Availability options *

<input type="text" value="No infrastructure redundancy required"/>
--

Automatically repair replication

Security details

Target VM security type	<input type="text" value="Standard"/>
Disk encryption type	<input type="text" value="Encryption at-rest with a platform-managed key"/>

Test Migration

Select the virtual network and subnet for test migration. Network properties can be changed from Compute and Network settings of replicating machine or when test migration is performed.

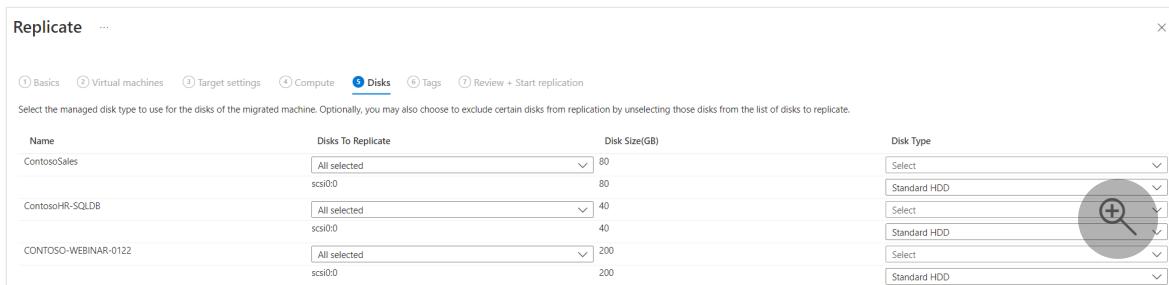
[Previous](#) [Next](#)

14. In **Compute**, review the VM name, size, OS disk type, and availability configuration (if selected in the previous step). VMs must conform with [Azure requirements](#).

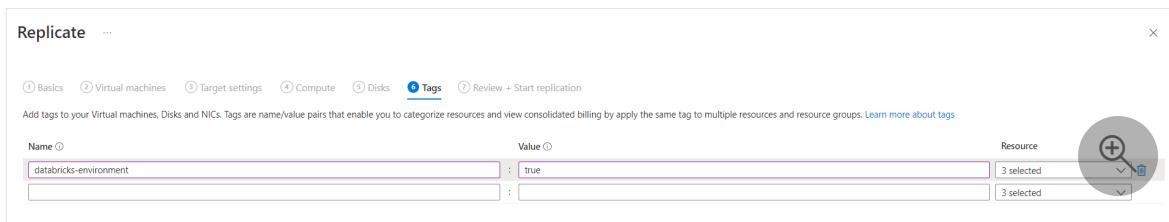
- **VM size:** If you're using assessment recommendations, the VM size dropdown shows the recommended size. Otherwise Azure Migrate picks a size based on the closest match in the Azure subscription. Alternatively, pick a manual size in [Azure VM size](#).
- **OS disk:** Specify the OS (boot) disk for the VM. The OS disk is the disk that has the operating system bootloader and installer.
- **Availability Zone:** Specify the Availability Zone to use.
- **Availability Set:** Specify the Availability Set to use.

15. In **Disks**, specify whether the VM disks should be replicated to Azure, and select the disk type (standard SSD/HDD or premium managed disks) in Azure. Then click **Next**.

- You can exclude disks from replication.
- If you exclude disks, won't be present on the Azure VM after migration.



16. In **Tags**, choose to add tags to your Virtual machines, Disks, and NICs.



17. In **Review and start replication**, review the settings, and click **Replicate** to start the initial replication for the servers.

Note

You can update replication settings any time before replication starts, **Manage > Replicating machines**. Settings can't be changed after replication starts.

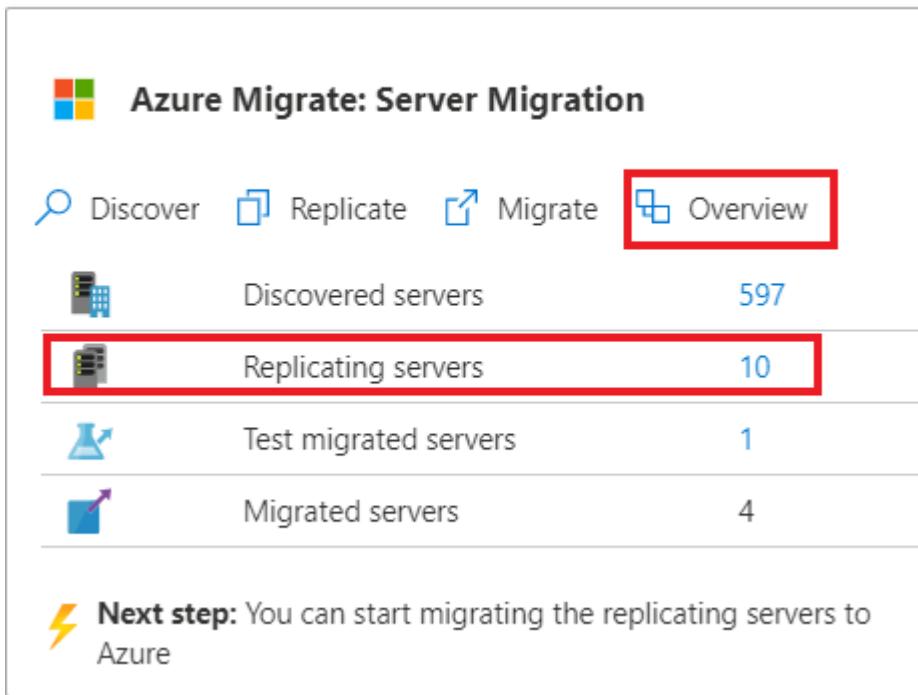
Track and monitor replication status

- When you click **Replicate** a Start Replication job begins.
- When the Start Replication job finishes successfully, the VMs begin their initial replication to Azure.
- After initial replication finishes, delta replication begins. Incremental changes to AWS VM disks are periodically replicated to the replica disks in Azure.

You can track job status in the portal notifications.

You can monitor replication status by clicking on **Replicating servers** in **Migration and modernization**.

Migration tools



Azure Migrate: Server Migration

Discover Replicate Migrate **Overview**

	Discovered servers	597
	Replicating servers	10
	Test migrated servers	1
	Migrated servers	4

Next step: You can start migrating the replicating servers to Azure

Run a test migration

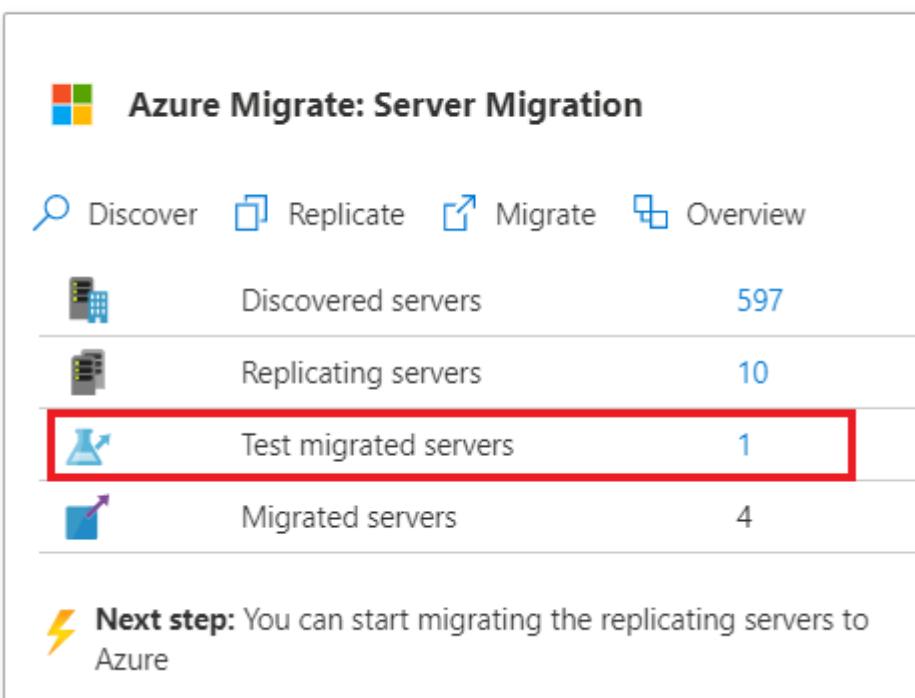
When delta replication begins, you can run a test migration for the VMs, before running a full migration to Azure. The test migration is highly recommended and provides an opportunity to discover any potential issues and fix them before you proceed with the actual migration. It's advised that you do this at least once for each VM before you migrate it.

- Running a test migration checks that migration will work as expected, without impacting the AWS VMs, which remain operational, and continue replicating.
- Test migration simulates the migration by creating an Azure VM using replicated data (usually migrating to a non-production VNet in your Azure subscription).
- You can use the replicated test Azure VM to validate the migration, perform app testing, and address any issues before full migration.

Do a test migration as follows:

1. In **Migration goals > Servers, databases and web apps > Migration and modernization**, select **Test migrated servers**.

Migration tools



Azure Migrate: Server Migration

Discover Replicate Migrate Overview

	Discovered servers	597
	Replicating servers	10
	Test migrated servers	1
	Migrated servers	4

Next step: You can start migrating the replicating servers to Azure

2. Right-click the VM to test, and click **Test migrate**.



Refresh Migrate Columns

Other

Last refreshed at 9/28/2021, 8:24:13 PM

Filter items...

Name	Status	Health	Migration phase	Test migration status	...
<Machine Name>	Protected	Healthy	-	Never performed	...

3. In **Test Migration**, select the Azure VNet in which the Azure VM will be located after the migration. We recommend you use a non-production VNet.
4. The **Test migration** job starts. Monitor the job in the portal notifications.
5. After the migration finishes, view the migrated Azure VM in **Virtual Machines** in the Azure portal. The machine name has a suffix **-Test**.
6. After the test is done, right-click the Azure VM in **Replicating machines**, and click **Clean up test migration**.

Name	Status	Health	Migration phase	Last sync	Test migration status
RBWin2K16-2	Migrated	Not applicable	Not applicable	1/19/2022, 6:05:47 PM	Never performed
a404-W2k8r2vm4	Migrated	Not applicable	Not applicable	1/19/2022, 8:15:38 PM	Never performed
Win11	Delta sync	Healthy	Ready to migrate	2/3/2022, 5:02:40 PM	2/1/2022, 5:37:18 PM Test migration
10. REDHAT63	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:18 PM	New
PD-GEN-MIG-CBT-0921	Migrated	Not applicable	Not applicable	1/28/2022, 7:55:41 PM	New
WIN2003R2-18	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:18 PM	New
ULVLM2	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:19 PM	New

ⓘ Note

You can now register your servers running SQL server with SQL VM RP to take advantage of automated patching, automated backup and simplified license management using SQL IaaS Agent Extension.

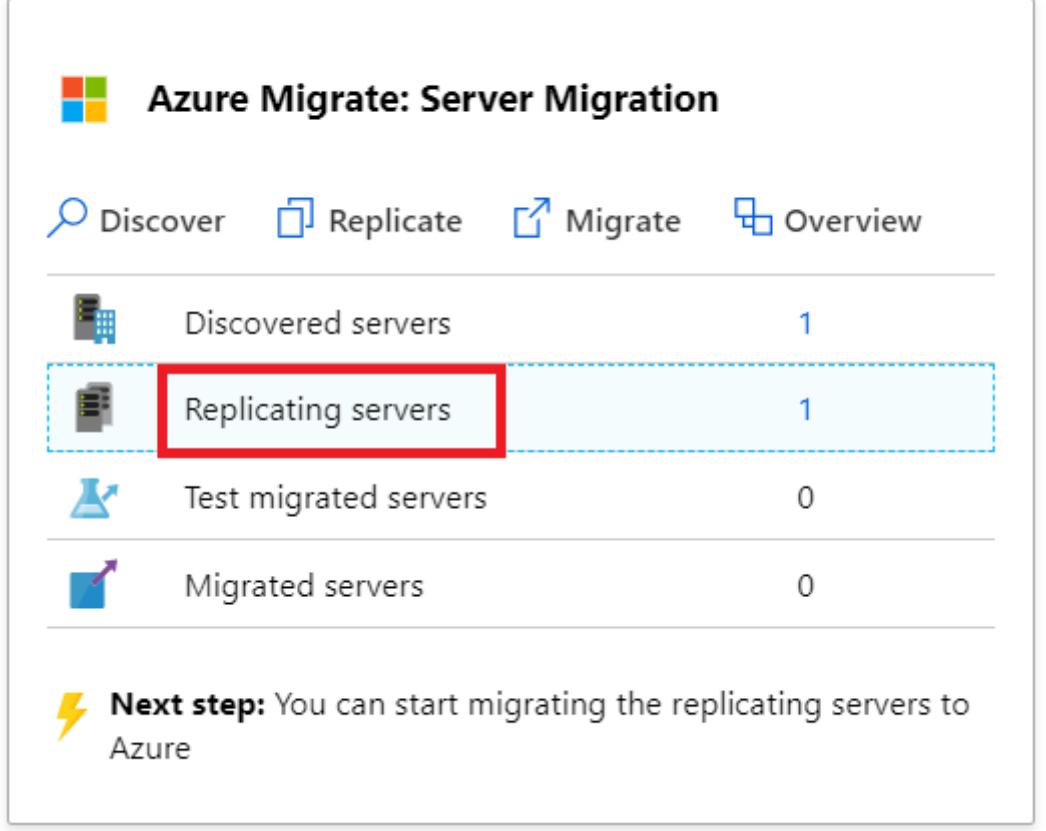
- Select **Manage > Replicating servers > Machine containing SQL server > Compute and Network** and select **yes** to register with SQL VM RP.
- Select Azure Hybrid benefit for SQL Server if you have SQL Server instances that are covered with active Software Assurance or SQL Server subscriptions and you want to apply the benefit to the machines you're migrating.

Migrate AWS VMs

After you've verified that the test migration works as expected, you can migrate the AWS VMs.

1. In the Azure Migrate project > **Servers, databases and web apps > Migration and modernization**, select **Replicating servers**.

Migration tools



The screenshot shows the Azure Migrate: Server Migration interface. At the top, there are four tabs: Discover, Replicate, Migrate, and Overview. Below the tabs, there is a list of server categories with counts:

Category	Count
Discovered servers	1
Replicating servers	1
Test migrated servers	0
Migrated servers	0

At the bottom, there is a note: **Next step:** You can start migrating the replicating servers to Azure.

2. In **Replicating machines**, right-click the VM > **Migrate**.
3. In **Migrate > Shut down virtual machines and perform a planned migration with no data loss**, select **Yes > OK**.

! **Note**

Automatic shutdown isn't supported while migrating AWS virtual machines.

4. A migration job starts for the VM. You can view the job status by clicking the notification bell icon on the top right of the portal page or by going to the jobs page of the Migration and modernization tool (Click **Overview** on the tool tile > Select **Jobs** from the left menu).
5. After the job finishes, you can view and manage the VM from the Virtual Machines page.

Complete the migration

1. After the migration is done, right-click the VM > **Stop migration**. This does the following:

- Stops replication for the AWS VM.
 - Removes the AWS VM from the **Replicating servers** count in the Migration and modernization tool.
 - Cleans up replication state information for the VM.
2. Verify and [troubleshoot any Windows activation issues on the Azure VM](#).
 3. Perform any post-migration app tweaks, such as updating host names, database connection strings, and web server configurations.
 4. Perform final application and migration acceptance testing on the migrated application now running in Azure.
 5. Cut over traffic to the migrated Azure VM instance.
 6. Update any internal documentation to show the new location and IP address of the Azure VMs.

Post-migration best practices

- For increased resilience:
 - Keep data secure by backing up Azure VMs using the Azure Backup service. [Learn more](#).
 - Keep workloads running and continuously available by replicating Azure VMs to a secondary region with Site Recovery. [Learn more](#).
- For increased security:
 - Lock down and limit inbound traffic access with [Microsoft Defender for Cloud - Just in time administration](#).
 - Manage and govern updates on Windows and Linux machines with [Azure Update Manager](#).
 - Restrict network traffic to management endpoints with [Network Security Groups](#).
 - Deploy [Azure Disk Encryption](#) to help secure disks, and keep data safe from theft and unauthorized access.
 - Read more about [securing IaaS resources](#), and visit the [Microsoft Defender for Cloud](#).
- For monitoring and management:
 - Consider deploying [Azure Cost Management](#) to monitor resource usage and spending.

Troubleshooting / Tips

Question: I cannot see my AWS VM in the discovered list of servers for migration

Answer: Check if your replication appliance meets the requirements. Make sure Mobility Agent is installed on the source VM to be migrated and is registered the Configuration

Server. Check the network setting and firewall rules to enable a network path between the replication appliance and source AWS VMs.

Question: How do I know if my VM was successfully migrated

Answer: Post-migration, you can view and manage the VM from the Virtual Machines page. Connect to the migrated VM to validate.

Question: I am unable to import VMs for migration from my previously created Server Assessment results

Answer: Currently, we don't support the import of assessment for this workflow. As a workaround, you can export the assessment and then manually select the VM recommendation during the Enable Replication step.

Question: I am getting the error "Failed to fetch BIOS GUID" while trying to discover my AWS VMs

Answer: Always use root login for authentication and not any pseudo user. Also review supported operating systems for AWS VMs.

Question: My replication status is not progressing. **Answer:** Check if your replication appliance meets the requirements. Make sure you've enabled the required ports on your replication appliance TCP port 9443 and HTTPS 443 for data transport. Ensure that there are no stale duplicate versions of the replication appliance connected to the same project.

Question: I am unable to Discover AWS Instances using Azure Migrate due to HTTP status code of 504 from the remote Windows management service

Answer: Make sure to review the Azure migrate appliance requirements and URL access needs. Make sure no proxy settings are blocking the appliance registration.

Question: Do I have to make any changes before I migrate my AWS VMs to Azure

Answer: You may have to make these changes before migrating your EC2 VMs to Azure:

- If you're using cloud-init for your VM provisioning, you may want to disable cloud-init on the VM before replicating it to Azure. The provisioning steps performed by cloud-init on the VM maybe AWS specific and won't be valid after the migration to Azure.
- If the VM is a PV VM (para-virtualized) and not HVM VM, you may not be able to run it as-is on Azure because para-virtualized VMs use a custom boot sequence in AWS. You may be able to get over this challenge by uninstalling PV drivers before you perform a migration to Azure.
- We always recommend you run a test migration before the final migration.

Question: Can I migrate AWS VMs running Amazon Linux Operating system

Answer: VMs running Amazon Linux can't be migrated as-is as Amazon Linux OS is only supported on AWS. To migrate workloads running on Amazon Linux, you can spin up a CentOS/RHEL VM in Azure and migrate the workload running on the AWS Linux machine using a relevant workload migration approach. For example, depending on the workload, there may be workload-specific tools to aid the migration – such as for databases or deployment tools in case of web servers.

Next steps

Investigate the [cloud migration journey](#) in the Azure Cloud Adoption Framework.

Migrate from Amazon Web Services (AWS) and other platforms to managed disks in Azure

Article • 05/26/2022

Applies to:  Windows VMs

You can upload VHD files from AWS or on-premises virtualization solutions to Azure to create virtual machines (VMs) that use managed disks. Azure managed disks removes the need to manage storage accounts for Azure IaaS VMs. You just specify the type of disk, and the size of that disk that you need, and Azure creates and manages the disk for you.

You can upload either generalized and specialized VHDs.

- **Generalized VHD** - has had all of your personal account information removed using Sysprep.
- **Specialized VHD** - maintains the user accounts, applications, and other state data from your original VM.

Important

Before uploading any VHD to Azure, you should follow [Prepare a Windows VHD or VHDX to upload to Azure](#)

Scenario	Documentation
You have existing AWS EC2 instances that you would like to migrate to Azure VMs using managed disks	Move a VM from Amazon Web Services (AWS) to Azure
You have a VM from another virtualization platform that you would like to use as an image to create multiple Azure VMs.	Upload a generalized VHD and use it to create a new VM in Azure
You have a uniquely customized VM that you would like to recreate in Azure.	Upload a specialized VHD to Azure and create a new VM

Overview of managed disks

Azure managed disks simplifies VM management by removing the need to manage storage accounts. Managed disks also benefit from better reliability of VMs in an Availability Set. It ensures that the disks of different VMs in an Availability Set are sufficiently isolated from each other to avoid a single point of failure. It automatically places disks of different VMs in an Availability Set in different Storage scale units (stamps) which limits the impact of single Storage scale unit failures caused due to hardware and software failures. Based on your needs, you can choose from four types of storage options. To learn about the available disk types, see our article [Select a disk type](#).

Plan for the migration to managed disks

This section helps you to make the best decision on VM and disk types.

If you are planning on migrating from unmanaged disks to managed disks, you should be aware that users with the [Virtual Machine Contributor](#) role will not be able to change the VM size (as they could pre-conversion). This is because VMs with managed disks require the user to have the Microsoft.Compute/disks/write permission on the OS disks.

Location

Pick a location where Azure managed disks are available. If you are migrating to premium SSDs, also ensure that premium storage is available in the region where you are planning to migrate to. See [Azure Services by Region](#) for up-to-date information on available locations.

VM sizes

If you are migrating to premium SSDs, you have to update the size of the VM to premium storage capable size available in the region where VM is located. Review the VM sizes that are premium storage capable. The Azure VM size specifications are listed in [Sizes for virtual machines](#). Review the performance characteristics of virtual machines that work with premium storage and choose the most appropriate VM size that best suits your workload. Make sure that there is sufficient bandwidth available on your VM to drive the disk traffic.

Disk sizes

For information on available disk types and sizes, see [What disk types are available in Azure?](#).

Disk caching policy

Premium Managed Disks

By default, disk caching policy is *Read-Only* for all the Premium data disks, and *Read-Write* for the Premium operating system disk attached to the VM. This configuration setting is recommended to achieve the optimal performance for your application's IOs. For write-heavy or write-only data disks (such as SQL Server log files), disable disk caching so that you can achieve better application performance.

Pricing

Review the [pricing for managed disks](#).

Next Steps

- Before uploading any VHD to Azure, you should follow [Prepare a Windows VHD or VHDX to upload to Azure](#)

Discover, assess, and migrate Amazon Web Services (AWS) VMs to Azure

Article • 10/27/2023

This tutorial shows you how to discover, assess, and migrate Amazon Web Services (AWS) virtual machines (VMs) to Azure VMs, using Azure Migrate: Server Assessment and Migration and modernization tools.

ⓘ Note

You migrate AWS VMs to Azure by treating them as physical servers.

In this tutorial, you'll learn how to:

- ✓ Verify prerequisites for migration.
- ✓ Prepare Azure resources with the Migration and modernization tool. Set up permissions for your Azure account and resources to work with Azure Migrate.
- ✓ Prepare AWS EC2 instances for migration.
- ✓ Add the Migration and modernization tool in the Azure Migrate hub.
- ✓ Set up the replication appliance and deploy the configuration server.
- ✓ Install the Mobility service on AWS VMs you want to migrate.
- ✓ Enable replication for VMs.
- ✓ Track and monitor the replication status.
- ✓ Run a test migration to make sure everything's working as expected.
- ✓ Run a full migration to Azure.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Discover and assess

Before you migrate to Azure, we recommend that you perform a VM discovery and migration assessment. This assessment helps right-size your AWS VMs for migration to Azure, and estimate potential Azure run costs.

Set up an assessment as follows:

1. Follow the [tutorial](#) to set up Azure and prepare your AWS VMs for an assessment.
Note that:

- Azure Migrate uses password authentication when discovering AWS instances. AWS instances don't support password authentication by default. Before you can discover instance, you need to enable password authentication.
 - For Windows machines, allow WinRM port 5985 (HTTP). This allows remote WMI calls.
 - For Linux machines:
 - a. Sign into each Linux machine.
 - b. Open the `sshd_config` file : `vi /etc/ssh/sshd_config`
 - c. In the file, locate the **PasswordAuthentication** line, and change the value to **yes**.
 - d. Save the file and close it. Restart the ssh service.
- If you're using a root user to discover your Linux VMs, ensure root login is allowed on the VMs.
 - a. Sign into each Linux machine
 - b. Open the `sshd_config` file : `vi /etc/ssh/sshd_config`
 - c. In the file, locate the **PermitRootLogin** line, and change the value to **yes**.
 - d. Save the file and close it. Restart the ssh service.

2. Then, follow this [tutorial](#) to set up an Azure Migrate project and appliance to discover and assess your AWS VMs.

Although we recommend that you try out an assessment, performing an assessment isn't a mandatory step to be able to migrate VMs.

Prerequisites

- Ensure that the AWS VMs you want to migrate are running a supported OS version. AWS VMs are treated like physical machines for the purpose of the migration. Review the [supported operating systems and kernel versions](#) for the physical server migration workflow. You can use standard commands like `hostnamectl` or `uname -a` to check the OS and kernel versions for your Linux VMs. We recommend you perform a test migration (test failover) to validate if the VM works as expected before proceeding with the actual migration.
- Make sure your AWS VMs comply with the [supported configurations](#) for migration to Azure.
- Verify that the AWS VMs that you replicate to Azure comply with [Azure VM requirements](#).
- There are some changes needed on the VMs before you migrate them to Azure.
 - For some operating systems, Azure Migrate makes these changes automatically.

- It's important to make these changes before you begin migration. If you migrate the VM before you make the change, the VM might not boot up in Azure. Review [Windows](#) and [Linux](#) changes you need to make.

Prepare Azure resources for migration

Prepare Azure for migration with the Migration and modernization tool.

[Expand table](#)

Task	Details
Create an Azure Migrate project	Your Azure account needs Contributor or Owner permissions to create a new project .
Verify permissions for your Azure account	Your Azure account needs permissions to create a VM, and write to an Azure managed disk.

Assign permissions to create project

1. In the Azure portal, open the subscription, and select **Access control (IAM)**.
2. In **Check access**, find the relevant account, and click it to view permissions.
3. You should have **Contributor** or **Owner** permissions.
 - If you just created a free Azure account, you're the owner of your subscription.
 - If you're not the subscription owner, work with the owner to assign the role.

Assign Azure account permissions

Assign the Virtual Machine Contributor role to the Azure account. This provides permissions to:

- Create a VM in the selected resource group.
- Create a VM in the selected virtual network.
- Write to an Azure managed disk.

Create an Azure network

[Set up](#) an Azure virtual network (VNet). When you replicate to Azure, the Azure VMs that are created are joined to the Azure VNet that you specify when you set up migration.

Prepare AWS instances for migration

To prepare for AWS to Azure migration, you need to prepare and deploy a replication appliance for migration.

Prepare a machine for the replication appliance

The Migration and modernization tool uses a replication appliance to replicate machines to Azure. The replication appliance runs the following components.

- **Configuration server:** The configuration server coordinates communications between the AWS environment and Azure, and manages data replication.
- **Process server:** The process server acts as a replication gateway. It receives replication data, optimizes it with caching, compression, and encryption, and sends it to a cache storage account in Azure.

Prepare for appliance deployment as follows:

- Set up a separate EC2 VM to host the replication appliance. This instance must be running Windows Server 2012 R2 or Windows Server 2016. [Review](#) the hardware, software, and networking requirements for the appliance.
- The appliance shouldn't be installed on a source VM that you want to replicate or on the Azure Migrate discovery and assessment appliance you may have installed before. It should be deployed on a different VM.
- The source AWS VMs to be migrated should have a network line of sight to the replication appliance. Configure necessary security group rules to enable this. It's recommended that the replication appliance is deployed in the same VPC as the source VMs to be migrated. If the replication appliance needs to be in a different VPC, the VPCs need to be connected through VPC peering.
- The source AWS VMs communicate with the replication appliance on ports HTTPS 443 (control channel orchestration) and TCP 9443 (data transport) inbound for replication management and replication data transfer. The replication appliance in turn orchestrates and sends replication data to Azure over port HTTPS 443 outbound. To configure these rules, edit the security group inbound/outbound rules with the appropriate ports and source IP information.

The screenshot shows the AWS VPC Security Groups list and the details page for a specific security group. The security group list table has columns: Security group ID, Security group name, VPC ID, Description, Owner, and Inbound rules count. The 'sg-b0cb2edb' row is selected and highlighted with a red box. The details page for 'sg-b0cb2edb - launch-wizard-2' shows the following information:

Security group name	Security group ID	Description	VPC ID
launch-wizard-2	sg-b0cb2edb	launch-wizard-2 created 2018-01-05T15:20:26.169+05:30	vpc-0512e36d
Owner	Inbound rules count	Outbound rules count	
409655286455	4 Permission entries	1 Permission entry	

Below the details, there are tabs for Inbound rules, Outbound rules, and Tags. The Inbound rules table lists the following rules:

Type	Protocol	Port range	Source	Description - optional
All traffic	All	All	sg-cfff1fa4 (launch-wizard-4)	-
Custom TCP	TCP	9443	0.0.0.0/0	-
RDP	TCP	3389	0.0.0.0/0	-
HTTPS	TCP	443	0.0.0.0/0	-

At the top right of the Inbound rules table, there is a red box around the 'Edit inbound rules' button.

- The replication appliance uses MySQL. Review the [options](#) for installing MySQL on the appliance.
- Review the Azure URLs required for the replication appliance to access [public](#) and [government](#) clouds.

Set up the replication appliance

The first step of migration is to set up the replication appliance. To set up the appliance for AWS VMs migration, you must download the installer file for the appliance, and then run it on the [VM you prepared](#).

Download the replication appliance installer

1. In the Azure Migrate project > **Servers, databases and web apps**, in **Migration and modernization**, select **Discover**.

 **Azure Migrate - Servers**
Microsoft

«

 Overview

 Migration goals

 Servers

 Databases

 Data Box

 Manage

 Discovered items

 Support + troubleshooting

 New support request

 Refresh

Last refreshed at: 6/20/2019, 8:28:41 PM (Click on "Refresh" to update the

Assessment tools

 **Azure Migrate: Server Assessment**

 Discover  Assess  Overview

Quick start

1: Discover
Click 'Discover' to start discovering your on-premises machines.

2: Assess
Once your on-premises machines are discovered, click "Assess" to create assessments.

Add more assessment tools? [Click here.](#)

Migration tools

 **Azure Migrate: Server Migration**

 Discover  Replicate  Migrate  Overview

Quick start

1: Discover
Click 'Discover' to start discovering your on-premises machines.

2: Replicate
Once your on-premises machines are discovered, click "Replicate" to start replicating the discovered machines.

3: Migrate
Once your machines have replicated, click "Migrate" to migrate your machines.

Add more migration tools? [Click here.](#)

2. In **Discover machines** > **Are your machines virtualized?**, click **Not virtualized/Other**.
3. In **Target region**, select the Azure region to which you want to migrate the machines.
4. Select **Confirm that the target region for migration is <region-name>**.
5. Click **Create resources**. This creates an Azure Site Recovery vault in the background.

- If you've already set up migration with the Migration and modernization tool, the target option can't be configured, since resources were set up previously.
- You can't change the target region for this project after clicking this button.
- To migrate your VMs to a different region, you'll need to create a new/different Azure Migrate project.

! Note

If you selected private endpoint as the connectivity method for the Azure Migrate project when it was created, the Recovery Services vault will also be configured for private endpoint connectivity. Ensure that the private endpoints are reachable from the replication appliance. [Learn more](#)

6. In **Do you want to install a new replication appliance?**, select **Install a replication appliance**.
7. In **Download and install the replication appliance software**, download the appliance installer, and the registration key. You need to the key in order to register the appliance. The key is valid for five days after it's downloaded.

Discover machines

Are your machines virtualized? [!](#)
Not virtualized / Other

Target region [!](#)
(US) Central US

Do you want to install a new replication appliance or scale-out existing setup?
Install a replication appliance [Help me choose](#)

The replication appliance (Configuration Server) is a virtual appliance that is deployed on-premises. The replication appliance coordinates and manages replication for the servers that are being migrated. Follow the steps outlined below to set up and configure the replication appliance

1. Download and install the replication appliance software.
Create a new Windows Server 2016 machine by following the [Configuration Server sizing guidelines](#).
[Download](#) the replication appliance software installer and use it to complete installation of the replication appliance software on the newly created Windows Server 2016 machine.

2. Configure the replication appliance and register it to the Azure Migrate project
Download the registration key file and use it to register the replication appliance to this project. The replication appliance installer will ask for a registration key.
[Download](#)

8. Copy the appliance setup file and key file to the Windows Server 2016 or Windows Server 2012 AWS VM you created for the replication appliance.
9. Run the replication appliance setup file, as described in the next procedure.
 - a. Under **Before You Begin**, select **Install the configuration server and process server**, and then select **Next**.

- b. In **Third-Party Software License**, select **I accept the third-party license agreement**, and then select **Next**.
- c. In **Registration**, select **Browse**, and then go to where you put the vault registration key file. Select **Next**.
- d. In **Internet Settings**, select **Connect to Azure Site Recovery without a proxy server**, and then select **Next**.
- e. The **Prerequisites Check** page runs checks for several items. When it's finished, select **Next**.
- f. In **MySQL Configuration**, provide a password for the MySQL DB, and then select **Next**.
- g. In **Environment Details**, select **No**. You don't need to protect your VMs. Then, select **Next**.
- h. In **Install Location**, select **Next** to accept the default.
- i. In **Network Selection**, select **Next** to accept the default.
- j. In **Summary**, select **Install**.
- k. **Installation Progress** shows you information about the installation process. When it's finished, select **Finish**. A window displays a message about a reboot. Select **OK**.
10. After the installation completes, the Appliance configuration wizard will be launched automatically (You can also launch the wizard manually by using the **cspconfigtool** shortcut that is created on the desktop of the appliance). In this tutorial, we'll be manually installing the Mobility Service on source VMs to be replicated, so create a dummy account in this step and proceed. You can provide the following details for creating the dummy account - "guest" as the friendly name, "username" as the username, and "password" as the password for the account. You'll be using this dummy account in the Enable Replication stage.
11. After the appliance has restarted after setup, in **Discover machines**, select the new appliance in **Select Configuration Server**, and click **Finalize registration**. Finalize registration performs a couple of final tasks to prepare the replication appliance.

 **3. Finalize registration**
Prepare for replication by finalizing registration for the replication appliance (Configuration Server). Select the replication appliance from the drop down to finalize registration for it.

* Select Configuration Server  Registration finalized

Install the Mobility service agent

A Mobility service agent must be pre-installed on the source AWS VMs to be migrated before you can initiate replication. The approach you choose to install the Mobility service agent may depend on your organization's preferences and existing tools, but be aware that the "push" installation method built into Azure Site Recovery is not currently supported. Approaches you may want to consider:

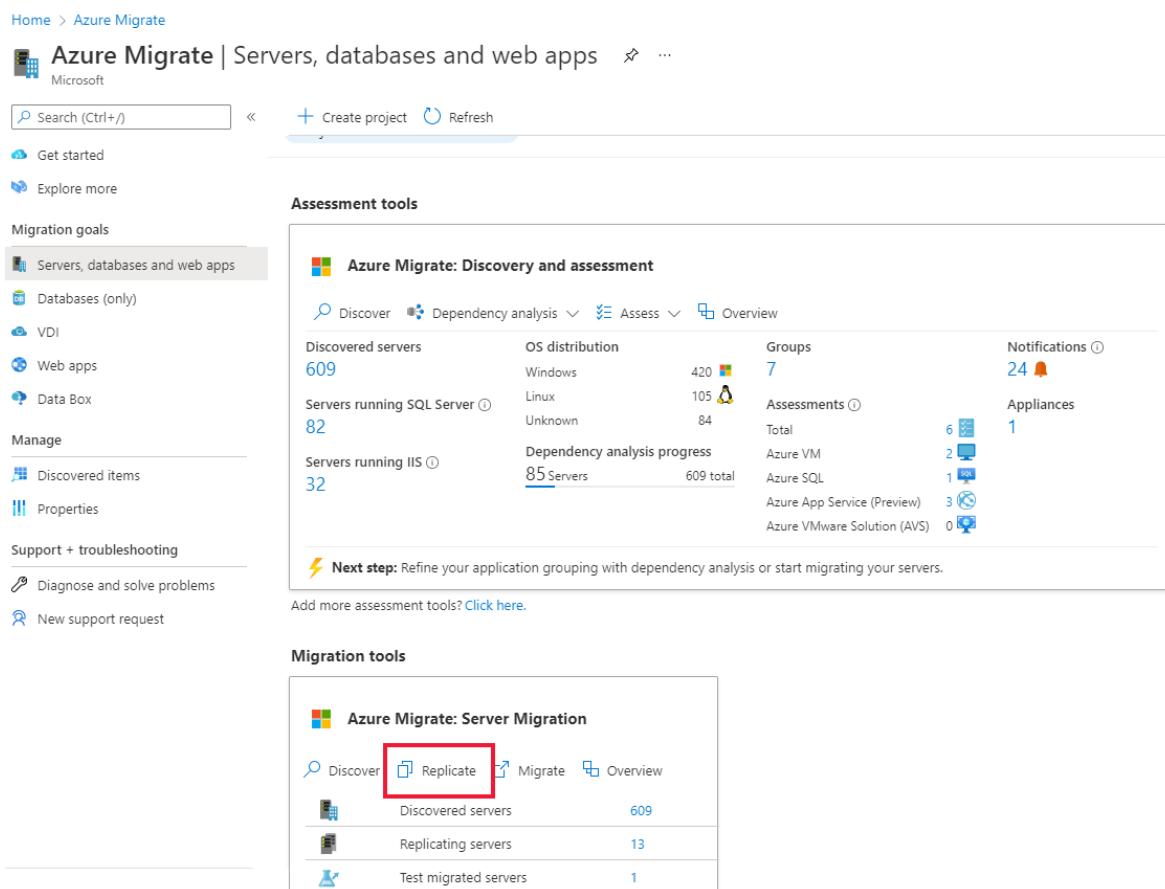
- [AWS System Manager ↗](#)
- [System Center Configuration Manager](#)
- [Arc for Servers and Custom Script Extensions](#)
- [Manual installation](#)

Enable replication for AWS VMs

ⓘ Note

Through the portal, you can add up to 10 VMs for replication at once. To replicate more VMs simultaneously, you can add them in batches of 10.

1. In the Azure Migrate project > **Servers, databases and web apps** > **Migration and modernization**, select **Replicate**.



The screenshot shows the Azure Migrate portal interface. The top navigation bar includes 'Home > Azure Migrate', a search bar, and a 'Create project' button. The main content area is divided into 'Assessment tools' and 'Migration tools' sections.

Assessment tools section:

- Azure Migrate: Discovery and assessment** card:
 - Discover, Dependency analysis, Assess, Overview buttons.
 - Discover: 609 servers.
 - Dependency analysis: 420 Windows, 105 Linux, 84 Unknown.
 - Assessments: 7 groups.
 - Notifications: 24.
- Next step: Refine your application grouping with dependency analysis or start migrating your servers.

Migration tools section:

- Azure Migrate: Server Migration** card:
 - Discover, Replicate, Migrate, Overview buttons.
 - Discover: 609 servers.
 - Replicate: 13 servers.
 - Migrate: 1 server.

A red box highlights the 'Replicate' button in the 'Migration tools' section.

2. In **Replicate**, > **Source settings** > **Are your machines virtualized?**, select **Not virtualized/Other**.
3. In **On-premises appliance**, select the name of the Azure Migrate appliance that you set up.
4. In **Process Server**, select the name of the replication appliance.
5. In **Guest credentials**, please select the dummy account created previously during the [replication installer setup](#) to install the Mobility service manually (push install is not supported). Then click **Next: Virtual machines**.

Replicate

The first step in migrating servers is to replicate them. Once replication completes, you can perform test migration before finally moving the servers to Azure.

Are your machines virtualized? *

Physical or other (AWS, GCP, Xen, etc.)

On-premises appliance *

CONTOSO-0122 (Azure Migrate Appliance)

6. In **Virtual Machines**, in **Import migration settings from an assessment?**, leave the default setting **No, I'll specify the migration settings manually**.
7. Check each VM you want to migrate. Then click **Next: Target settings**.

Name	Azure VM Readiness	IP Address	Operating System	Boot Type	Source
ContosoSales	Ready	2404:f801:4800:1c:2d77:36ade95b:35d7...	Microsoft Windows Server 2016 or later ...	bios	idclab-vcen67.fareast.corp.microsoft.com
ContosoHR-SQLDB	Ready	2404:f801:4800:25:c823:79ff:6a48:2666:1...	Microsoft Windows Server 2016 or later ...	bios	idclab-vcen67.fareast.corp.microsoft.com
CONTOSO-WEBINAR-...	Ready	2404:f801:4800:25:c823:79ff:6a48:2666:1...	Microsoft Windows Server 2016 or later ...	bios	idclab-vcen67.fareast.corp.microsoft.com

8. In **Target settings**, select the subscription, and target region to which you'll migrate, and specify the resource group in which the Azure VMs will reside after migration.
9. In **Virtual Network**, select the Azure VNet/subnet to which the Azure VMs will be joined after migration.

10. In **Cache storage account**, keep the default option to use the cache storage account that is automatically created for the project. Use the dropdown if you'd like to specify a different storage account to use as the cache storage account for replication.

 **Note**

- If you selected private endpoint as the connectivity method for the Azure Migrate project, grant the Recovery Services vault access to the cache storage account. [Learn more](#)
- To replicate using ExpressRoute with private peering, create a private endpoint for the cache storage account. [Learn more](#)

11. In **Availability options**, select:

- Availability Zone to pin the migrated machine to a specific Availability Zone in the region. Use this option to distribute servers that form a multi-node application tier across Availability Zones. If you select this option, you'll need to specify the Availability Zone to use for each of the selected machine in the Compute tab. This option is only available if the target region selected for the migration supports Availability Zones
- Availability Set to place the migrated machine in an Availability Set. The target Resource Group that was selected must have one or more availability sets in order to use this option.
- No infrastructure redundancy required option if you don't need either of these availability configurations for the migrated machines.

12. In **Disk encryption type**, select:

- Encryption-at-rest with platform-managed key
- Encryption-at-rest with customer-managed key
- Double encryption with platform-managed and customer-managed keys

 **Note**

To replicate VMs with CMK, you'll need to [create a disk encryption set](#) under the target Resource Group. A disk encryption set object maps Managed Disks to a Key Vault that contains the CMK to use for SSE.

13. In **Azure Hybrid Benefit**:

- Select **No** if you don't want to apply Azure Hybrid Benefit. Then click **Next**.
- Select **Yes** if you have Windows Server machines that are covered with active Software Assurance or Windows Server subscriptions, and you want to apply the benefit to the machines you're migrating. Then click **Next**.

Replicate

Configure settings and target properties for migration.

Region *	<input type="text" value="East US"/>
Storage account *	<input type="text" value="(new) migratersa899899444"/>
Subscription *	<input type="text" value="Azure Migrate Test2"/>
Resource group *	<input type="text" value="Select a resource group"/>

VMs on this subscription, wherein SQL Server is running, are being onboarded to the SQL IaaS extension. You can choose to disable the same by following steps listed [here](#)

Register with SQL IaaS extension

Apply Azure Hybrid with an eligible Windows Server license and save up to 49% vs. pay-as-you-go virtual machine costs.

I have a Windows Server license

Virtual network *

<input type="text" value="Select a network"/>

Subnet *

<input type="text" value="Select a subnet"/>
--

Availability options *

<input type="text" value="No infrastructure redundancy required"/>
--

Automatically repair replication

Security details

Target VM security type	<input type="text" value="Standard"/>
Disk encryption type	<input type="text" value="Encryption at-rest with a platform-managed key"/>

Test Migration

Select the virtual network and subnet for test migration. Network properties can be changed from Compute and Network settings of replicating machine or when test migration is performed.

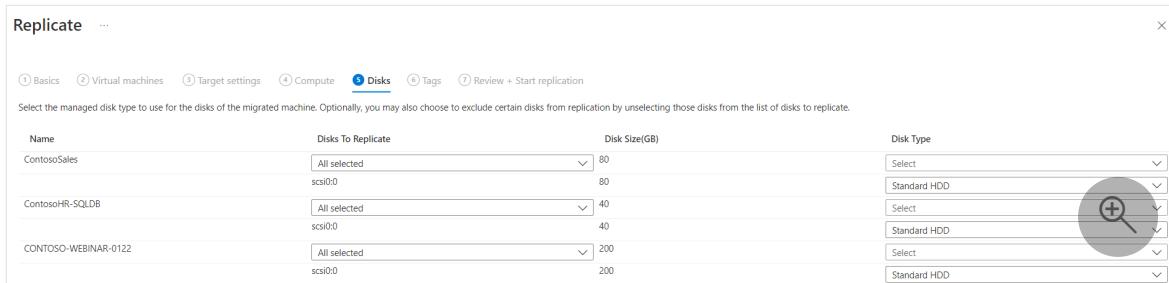
Previous **Next**

14. In **Compute**, review the VM name, size, OS disk type, and availability configuration (if selected in the previous step). VMs must conform with [Azure requirements](#).

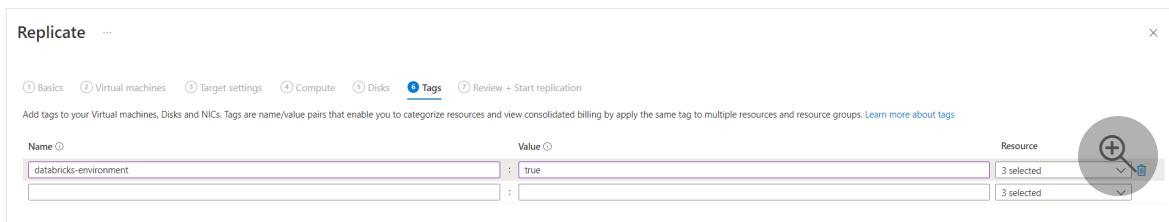
- **VM size:** If you're using assessment recommendations, the VM size dropdown shows the recommended size. Otherwise Azure Migrate picks a size based on the closest match in the Azure subscription. Alternatively, pick a manual size in [Azure VM size](#).
- **OS disk:** Specify the OS (boot) disk for the VM. The OS disk is the disk that has the operating system bootloader and installer.
- **Availability Zone:** Specify the Availability Zone to use.
- **Availability Set:** Specify the Availability Set to use.

15. In **Disks**, specify whether the VM disks should be replicated to Azure, and select the disk type (standard SSD/HDD or premium managed disks) in Azure. Then click **Next**.

- You can exclude disks from replication.
- If you exclude disks, won't be present on the Azure VM after migration.



16. In **Tags**, choose to add tags to your Virtual machines, Disks, and NICs.



17. In **Review and start replication**, review the settings, and click **Replicate** to start the initial replication for the servers.

Note

You can update replication settings any time before replication starts, **Manage > Replicating machines**. Settings can't be changed after replication starts.

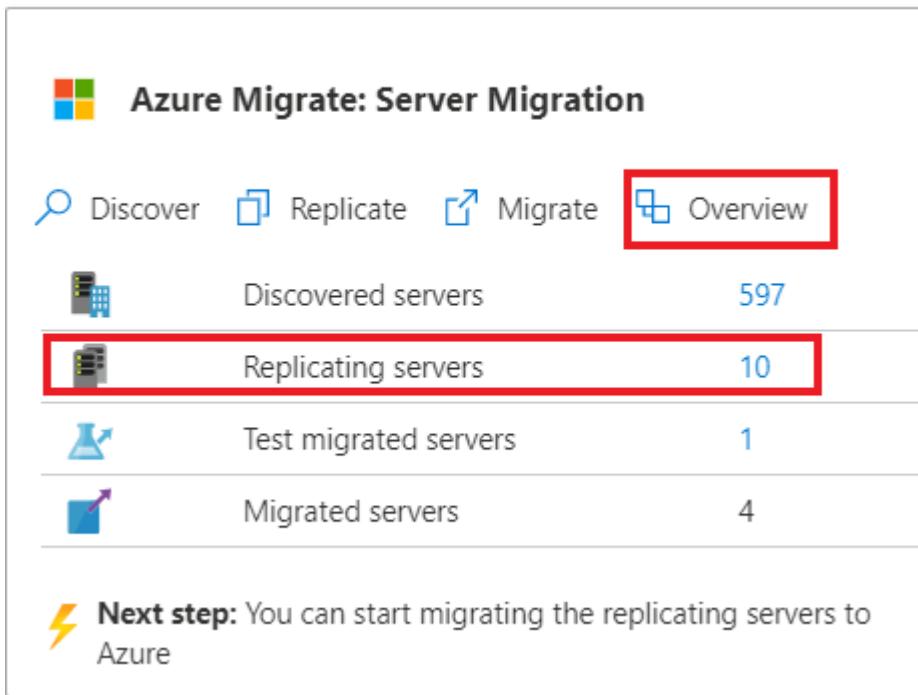
Track and monitor replication status

- When you click **Replicate** a Start Replication job begins.
- When the Start Replication job finishes successfully, the VMs begin their initial replication to Azure.
- After initial replication finishes, delta replication begins. Incremental changes to AWS VM disks are periodically replicated to the replica disks in Azure.

You can track job status in the portal notifications.

You can monitor replication status by clicking on **Replicating servers** in **Migration and modernization**.

Migration tools



Azure Migrate: Server Migration

Discover Replicate Migrate **Overview**

	Discovered servers	597
	Replicating servers	10
	Test migrated servers	1
	Migrated servers	4

Next step: You can start migrating the replicating servers to Azure

Run a test migration

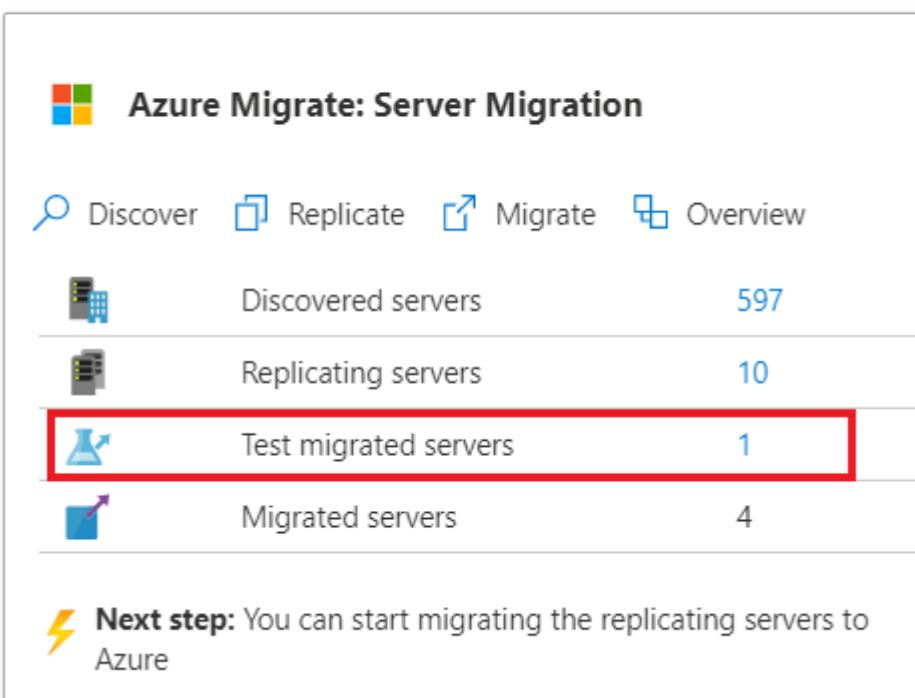
When delta replication begins, you can run a test migration for the VMs, before running a full migration to Azure. The test migration is highly recommended and provides an opportunity to discover any potential issues and fix them before you proceed with the actual migration. It's advised that you do this at least once for each VM before you migrate it.

- Running a test migration checks that migration will work as expected, without impacting the AWS VMs, which remain operational, and continue replicating.
- Test migration simulates the migration by creating an Azure VM using replicated data (usually migrating to a non-production VNet in your Azure subscription).
- You can use the replicated test Azure VM to validate the migration, perform app testing, and address any issues before full migration.

Do a test migration as follows:

1. In **Migration goals > Servers, databases and web apps > Migration and modernization**, select **Test migrated servers**.

Migration tools



Azure Migrate: Server Migration

Discover Replicate Migrate Overview

	Discovered servers	597
	Replicating servers	10
	Test migrated servers	1
	Migrated servers	4

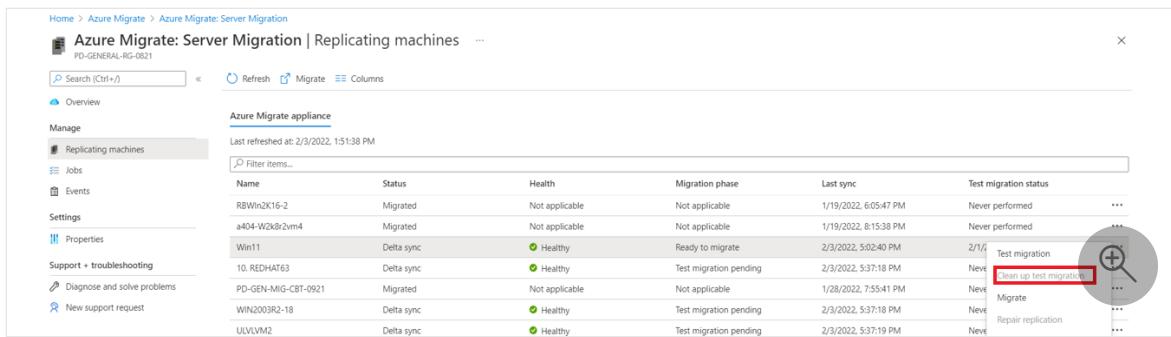
Next step: You can start migrating the replicating servers to Azure

2. Right-click the VM to test, and click **Test migrate**.



Name	Status	Health	Migration phase	Test migration status	...
<Machine Name>	Protected	Healthy	-	Never performed	...

3. In **Test Migration**, select the Azure VNet in which the Azure VM will be located after the migration. We recommend you use a non-production VNet.
4. The **Test migration** job starts. Monitor the job in the portal notifications.
5. After the migration finishes, view the migrated Azure VM in **Virtual Machines** in the Azure portal. The machine name has a suffix **-Test**.
6. After the test is done, right-click the Azure VM in **Replicating machines**, and click **Clean up test migration**.



Name	Status	Health	Migration phase	Last sync	Test migration status
RBWin2K16-2	Migrated	Not applicable	Not applicable	1/19/2022, 6:05:47 PM	Never performed
a404-W2k8r2vm4	Migrated	Not applicable	Not applicable	1/19/2022, 8:15:38 PM	Never performed
Win11	Delta sync	Healthy	Ready to migrate	2/3/2022, 5:02:40 PM	2/1/2022, 1:45:18 PM Test migration
10. REDHAT63	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:18 PM	New
PD-GEN-MIG-CBT-0921	Migrated	Not applicable	Not applicable	1/28/2022, 7:55:41 PM	New
WIN2003R2-18	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:18 PM	New
ULVLM2	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:19 PM	New

ⓘ Note

You can now register your servers running SQL server with SQL VM RP to take advantage of automated patching, automated backup and simplified license management using SQL IaaS Agent Extension.

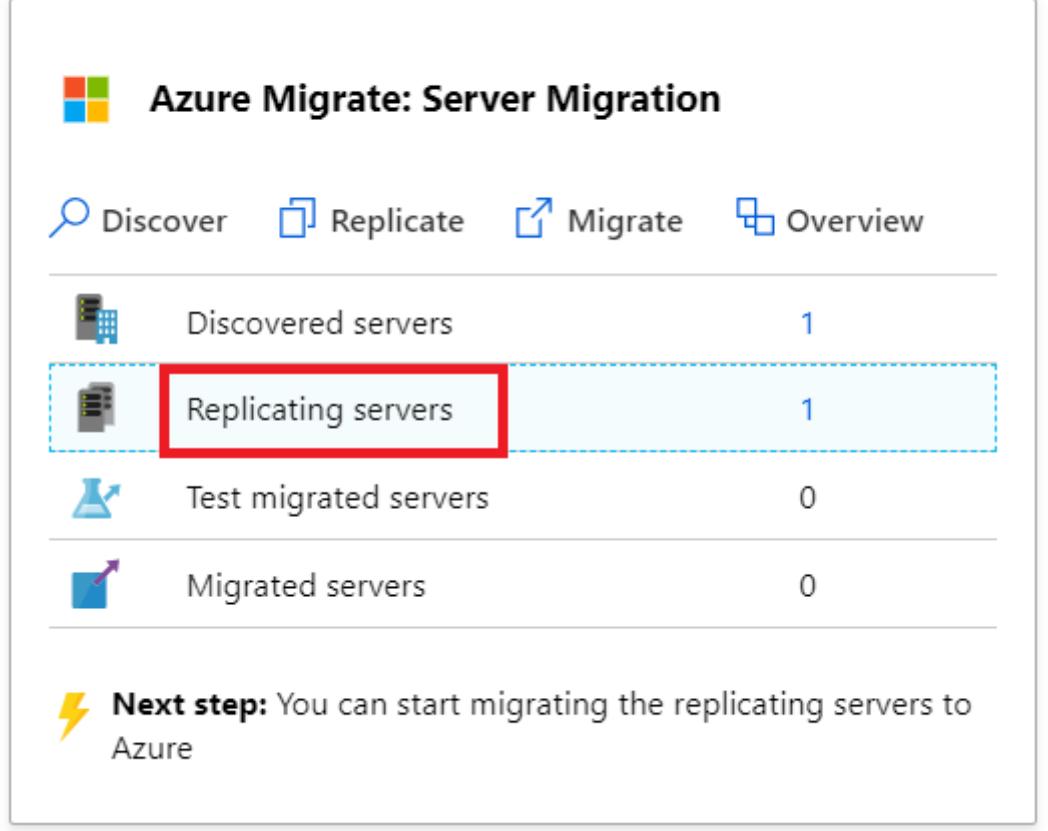
- Select **Manage > Replicating servers > Machine containing SQL server > Compute and Network** and select **yes** to register with SQL VM RP.
- Select Azure Hybrid benefit for SQL Server if you have SQL Server instances that are covered with active Software Assurance or SQL Server subscriptions and you want to apply the benefit to the machines you're migrating.

Migrate AWS VMs

After you've verified that the test migration works as expected, you can migrate the AWS VMs.

1. In the Azure Migrate project > **Servers, databases and web apps > Migration and modernization**, select **Replicating servers**.

Migration tools



The screenshot shows the Azure Migrate: Server Migration interface. At the top, there are four tabs: Discover, Replicate, Migrate, and Overview. Below the tabs, there is a list of server categories with counts:

Category	Count
Discovered servers	1
Replicating servers	1
Test migrated servers	0
Migrated servers	0

At the bottom, there is a note: **Next step:** You can start migrating the replicating servers to Azure.

2. In **Replicating machines**, right-click the VM > **Migrate**.
3. In **Migrate > Shut down virtual machines and perform a planned migration with no data loss**, select **Yes > OK**.

! **Note**

Automatic shutdown isn't supported while migrating AWS virtual machines.

4. A migration job starts for the VM. You can view the job status by clicking the notification bell icon on the top right of the portal page or by going to the jobs page of the Migration and modernization tool (Click **Overview** on the tool tile > Select **Jobs** from the left menu).
5. After the job finishes, you can view and manage the VM from the Virtual Machines page.

Complete the migration

1. After the migration is done, right-click the VM > **Stop migration**. This does the following:

- Stops replication for the AWS VM.
 - Removes the AWS VM from the **Replicating servers** count in the Migration and modernization tool.
 - Cleans up replication state information for the VM.
2. Verify and [troubleshoot any Windows activation issues on the Azure VM](#).
 3. Perform any post-migration app tweaks, such as updating host names, database connection strings, and web server configurations.
 4. Perform final application and migration acceptance testing on the migrated application now running in Azure.
 5. Cut over traffic to the migrated Azure VM instance.
 6. Update any internal documentation to show the new location and IP address of the Azure VMs.

Post-migration best practices

- For increased resilience:
 - Keep data secure by backing up Azure VMs using the Azure Backup service. [Learn more](#).
 - Keep workloads running and continuously available by replicating Azure VMs to a secondary region with Site Recovery. [Learn more](#).
- For increased security:
 - Lock down and limit inbound traffic access with [Microsoft Defender for Cloud - Just in time administration](#).
 - Manage and govern updates on Windows and Linux machines with [Azure Update Manager](#).
 - Restrict network traffic to management endpoints with [Network Security Groups](#).
 - Deploy [Azure Disk Encryption](#) to help secure disks, and keep data safe from theft and unauthorized access.
 - Read more about [securing IaaS resources](#)  , and visit the [Microsoft Defender for Cloud](#) .
- For monitoring and management:
 - Consider deploying [Azure Cost Management](#) to monitor resource usage and spending.

Troubleshooting / Tips

Question: I cannot see my AWS VM in the discovered list of servers for migration

Answer: Check if your replication appliance meets the requirements. Make sure Mobility Agent is installed on the source VM to be migrated and is registered the Configuration

Server. Check the network setting and firewall rules to enable a network path between the replication appliance and source AWS VMs.

Question: How do I know if my VM was successfully migrated

Answer: Post-migration, you can view and manage the VM from the Virtual Machines page. Connect to the migrated VM to validate.

Question: I am unable to import VMs for migration from my previously created Server Assessment results

Answer: Currently, we don't support the import of assessment for this workflow. As a workaround, you can export the assessment and then manually select the VM recommendation during the Enable Replication step.

Question: I am getting the error "Failed to fetch BIOS GUID" while trying to discover my AWS VMs

Answer: Always use root login for authentication and not any pseudo user. Also review supported operating systems for AWS VMs.

Question: My replication status is not progressing. **Answer:** Check if your replication appliance meets the requirements. Make sure you've enabled the required ports on your replication appliance TCP port 9443 and HTTPS 443 for data transport. Ensure that there are no stale duplicate versions of the replication appliance connected to the same project.

Question: I am unable to Discover AWS Instances using Azure Migrate due to HTTP status code of 504 from the remote Windows management service

Answer: Make sure to review the Azure migrate appliance requirements and URL access needs. Make sure no proxy settings are blocking the appliance registration.

Question: Do I have to make any changes before I migrate my AWS VMs to Azure

Answer: You may have to make these changes before migrating your EC2 VMs to Azure:

- If you're using cloud-init for your VM provisioning, you may want to disable cloud-init on the VM before replicating it to Azure. The provisioning steps performed by cloud-init on the VM maybe AWS specific and won't be valid after the migration to Azure.
- If the VM is a PV VM (para-virtualized) and not HVM VM, you may not be able to run it as-is on Azure because para-virtualized VMs use a custom boot sequence in AWS. You may be able to get over this challenge by uninstalling PV drivers before you perform a migration to Azure.
- We always recommend you run a test migration before the final migration.

Question: Can I migrate AWS VMs running Amazon Linux Operating system

Answer: VMs running Amazon Linux can't be migrated as-is as Amazon Linux OS is only supported on AWS. To migrate workloads running on Amazon Linux, you can spin up a CentOS/RHEL VM in Azure and migrate the workload running on the AWS Linux machine using a relevant workload migration approach. For example, depending on the workload, there may be workload-specific tools to aid the migration – such as for databases or deployment tools in case of web servers.

Next steps

Investigate the [cloud migration journey](#) in the Azure Cloud Adoption Framework.

Azure security solutions for AWS

Azure

Microsoft Sentinel

Microsoft Defender for Cloud Apps

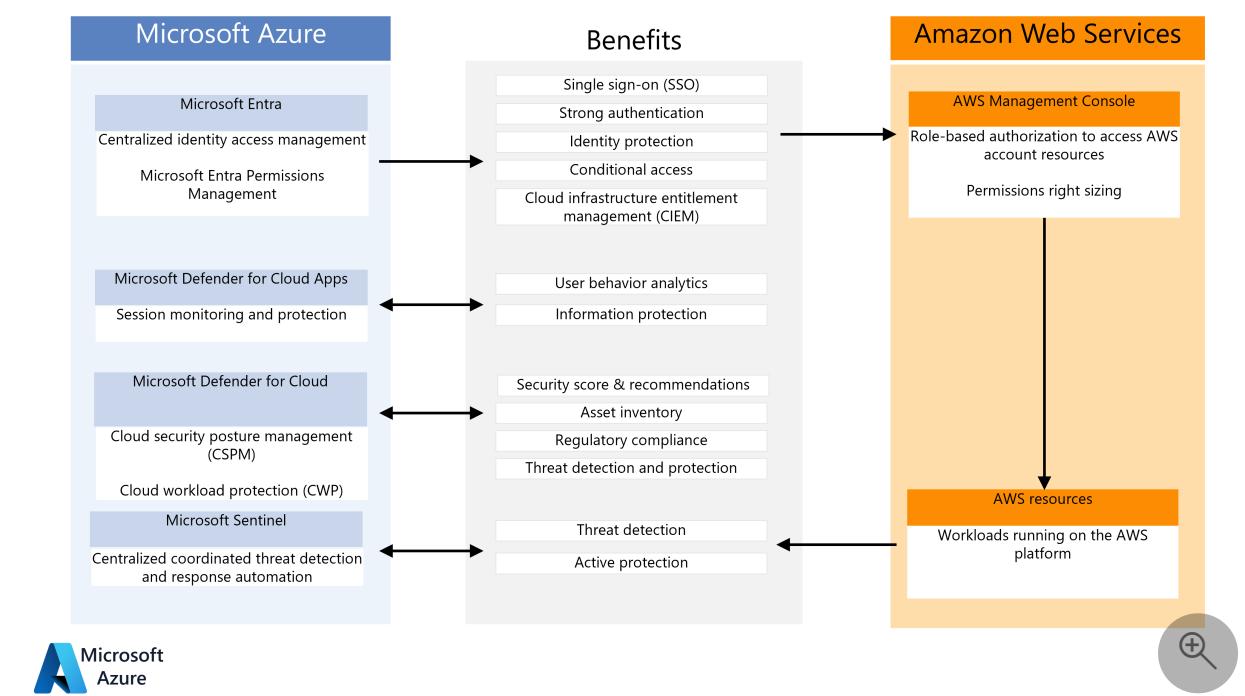
Microsoft Defender for Cloud

This guide shows how Microsoft Defender for Cloud Apps and Microsoft Sentinel can help secure and protect Amazon Web Services (AWS) account access and environments.

AWS organizations that use Microsoft Entra ID for Microsoft 365 or hybrid cloud identity and access protection can quickly and easily [deploy Microsoft Entra ID for AWS accounts](#), often without additional cost.

Architecture

This diagram summarizes how AWS installations can benefit from key Microsoft security components:



Download a [PowerPoint file](#) of this architecture.

Workflow

- Microsoft Entra ID provides centralized *single sign-on (SSO)* and strong authentication through *multifactor authentication* and the *conditional access* feature. Microsoft Entra ID supports AWS role-based identities and authorization for access to AWS resources. For more information and detailed instructions, see [Microsoft Entra identity and access management for AWS](#).

Permissions Management is a *cloud infrastructure entitlement management (CIEM)* product that provides comprehensive visibility and control over permissions for any AWS identity or resource. You can use Microsoft Entra Permissions Management to:

- Get a multi-dimensional view of your risk by assessing identities, permissions, and resources.
- Automate the enforcement of the [least privilege](#) policy in your entire multicloud infrastructure.
- Use anomaly and outlier detection to prevent data breaches that are caused by misuse and malicious exploitation of permissions.

For more information and detailed onboarding instructions, see [Onboard an Amazon Web Services \(AWS\) account](#).

- Defender for Cloud Apps:
 - Integrates with the Microsoft Entra Conditional Access feature to enforce additional restrictions.
 - Helps monitor and protect sessions after sign-in.
 - Uses *user behavior analytics (UBA)* and other AWS APIs to monitor sessions and users and to support information protection.
- Microsoft Defender for Cloud displays AWS security recommendations in the Defender for Cloud portal together with Azure recommendations. Defender for Cloud offers more than 160 out-of-the-box recommendations for infrastructure as a service (IaaS) and platform as a service (PaaS) services. It also provides support for regulatory standards, including Center for Internet Security (CIS) and payment card industry (PCI) standards, and for the AWS Foundational Security Best Practices standard. Defender for Cloud also provides cloud workload protection (CWP) for [Amazon EKS clusters](#), [AWS EC2 instances](#), and [SQL servers that run on AWS EC2](#).
- Microsoft Sentinel integrates with Defender for Cloud Apps and AWS to detect and automatically respond to threats. Microsoft Sentinel monitors the AWS environment for misconfiguration, potential malware, and advanced threats to AWS identities, devices, applications, and data.

Components

- [Microsoft Defender for Cloud Apps](#)
- [Microsoft Defender for Cloud](#)
- [Microsoft Sentinel](#)
- [Microsoft Entra ID](#)

Defender for Cloud Apps for visibility and control

When several users or roles make administrative changes, a consequence can be *configuration drift* away from intended security architecture and standards. Security standards can also change over time. Security personnel must constantly and consistently detect new risks, evaluate mitigation options, and update security architecture to prevent potential breaches. Security management across multiple public cloud and private infrastructure environments can become burdensome.

Defender for Cloud Apps is a *cloud access security broker (CASB)* platform with *cloud security posture management (CSPM)* capabilities. Defender for Cloud Apps can connect to multiple cloud services and applications to collect security logs, monitor user behavior, and impose restrictions that the platforms themselves might not offer.

Defender for Cloud Apps provides several capabilities that can integrate with AWS for immediate benefits:

- The Defender for Cloud Apps app connector uses several AWS APIs, including UBA, to search for configuration issues and threats on the AWS platform.
- AWS Access Controls can enforce sign-in restrictions that are based on application, device, IP address, location, registered ISP, and specific user attributes.
- Session Controls for AWS block potential malware uploads or downloads based on Microsoft Defender Threat Intelligence or real-time content inspection.
- Session controls can also use real-time content inspection and sensitive data detection to impose *data loss prevention (DLP)* rules that prevent cut, copy, paste, or print operations.

Defender for Cloud Apps is available standalone, or as part of Microsoft Enterprise Mobility + Security E5, which includes Microsoft Entra ID P2. For pricing and licensing information, see [Enterprise Mobility + Security pricing options](#) .

Defender for Cloud for CSPM and CWP platforms (CWPP)

With cloud workloads commonly spanning multiple cloud platforms, cloud security services must do the same. Defender for Cloud helps protect workloads in Azure, AWS, and Google Cloud Platform (GCP).

Defender for Cloud provides an agentless connection to your AWS account. Defender for Cloud also offers plans to secure your AWS resources:

- The [Defender for Cloud overview page](#) displays CSPM metrics, alerts, and insights. Defender for Cloud assesses your AWS resources according to [AWS-specific security recommendations](#) and incorporates your security posture into your secure

score. The [asset inventory](#) provides a single place to view all your protected AWS resources. The [regulatory compliance dashboard](#) reflects the status of your compliance with built-in standards that are specific to AWS. Examples include AWS CIS standards, PCI data security standards (PCI-DSS), and the AWS Foundational Security Best Practices standard.

- [Microsoft Defender for Servers](#) brings threat detection and advanced defenses to supported Windows and Linux EC2 instances.
- [Microsoft Defender for Containers](#) brings threat detection and advanced defenses to supported Amazon EKS clusters.
- [Microsoft Defender for SQL](#) brings threat detection and advanced defenses to your SQL servers that run on AWS EC2 and AWS RDS Custom for SQL Server.

Microsoft Sentinel for advanced threat detection

Threats can come from a wide range of devices, applications, locations, and user types. DLP requires inspecting content during upload or download, because post-mortem review might be too late. AWS doesn't have native capabilities for device and application management, risk-based conditional access, session-based controls, or inline UBA.

It's critical that security solutions reduce complexity and deliver comprehensive protection regardless of whether resources are in multicloud, on-premises, or hybrid environments. Defender for Cloud provides CSPM and CWP. Defender for Cloud identifies configuration weak spots across AWS to help strengthen your overall security posture. It also helps provide threat protection for Amazon EKS Linux clusters, AWS EC2 instances, and SQL servers in AWS EC2.

[Microsoft Sentinel](#) is a *security information and event management (SIEM) and security orchestration, automation, and response (SOAR)* solution that centralizes and coordinates threat detection and response automation for modern security operations. Microsoft Sentinel can monitor AWS accounts to compare events across multiple firewalls, network devices, and servers. Microsoft Sentinel combines monitoring data with threat intelligence, analytics rules, and machine learning to discover and respond to advanced attack techniques.

You can connect AWS and Defender for Cloud Apps with Microsoft Sentinel. Then you can see Defender for Cloud Apps alerts and run additional threat checks that use multiple Defender Threat Intelligence feeds. Microsoft Sentinel can initiate a coordinated response that's outside Defender for Cloud Apps. Microsoft Sentinel can also integrate with IT service management (ITSM) solutions and retain data on a long-term basis for compliance purposes.

Scenario details

Microsoft offers several security solutions that can help secure and protect AWS accounts and environments.

Other Microsoft security components can integrate with Microsoft Entra ID to provide additional security for AWS accounts:

- Defender for Cloud Apps backs up Microsoft Entra ID with session protection and user-behavior monitoring.
- Defender for Cloud provides threat protection to AWS workloads. It also helps proactively strengthen security for AWS environments and uses an agentless approach to connect to those environments.
- Microsoft Sentinel integrates with Microsoft Entra ID and Defender for Cloud Apps to detect and automatically respond to threats against AWS environments.

These Microsoft security solutions are extensible and offer multiple levels of protection. You can implement one or more of these solutions along with other types of protection for a full-security architecture that helps protect current and future AWS deployments.

Potential use cases

This article provides AWS identity architects, administrators, and security analysts with immediate insights and detailed guidance for deploying several Microsoft security solutions.

Recommendations

Keep the following points in mind when you develop a security solution.

Security recommendations

The following principles and guidelines are important for any cloud security solution:

- Ensure that the organization can monitor, detect, and automatically protect user and programmatic access into cloud environments.
- Continually review current accounts to ensure identity and permission governance and control.
- Follow least privilege and [zero trust](#) principles. Make sure that users can access only the specific resources that they require, from trusted devices and known

locations. Reduce the permissions of every administrator and developer to provide only the rights that they need for the role that they perform. Review regularly.

- Continuously monitor platform configuration changes, especially if they provide opportunities for privilege escalation or attack persistence.
- Prevent unauthorized data exfiltration by actively inspecting and controlling content.
- Take advantage of solutions that you might already own, like Microsoft Entra ID P2, that can increase security without additional expense.

Basic AWS account security

To ensure basic security hygiene for AWS accounts and resources:

- Review the AWS security guidance at [Best practices for securing AWS accounts and resources](#).
- Reduce the risk of uploading and downloading malware and other malicious content by actively inspecting all data transfers through the AWS Management Console. Content that you upload or download directly to resources within the AWS platform, such as web servers or databases, might need additional protection.
- Consider protecting access to other resources, including:
 - Resources created within the AWS account.
 - Specific workload platforms, like Windows Server, Linux Server, or containers.
 - Devices that administrators and developers use to access the AWS Management Console.

Deploy this scenario

Take the steps in the following sections to implement a security solution.

Plan and prepare

To prepare for deployment of Azure security solutions, review and record current AWS and Microsoft Entra account information. If you've deployed more than one AWS account, repeat these steps for each account.

1. In the [AWS Billing Management Console](#), record the following current AWS account information:
 - **AWS Account ID**, a unique identifier
 - **Account name**, or root user

- **Payment method**, whether assigned to a credit card or a company billing agreement
- **Alternate contacts** who have access to AWS account information
- **Security questions**, securely updated and recorded for emergency access
- **AWS regions** that are enabled or disabled to comply with data security policy

2. In the [Azure portal](#), review the Microsoft Entra tenant:

- Assess **Tenant information** to see whether the tenant has a Microsoft Entra ID P1 or P2 license. A P2 license provides [advanced Microsoft Entra identity management](#) features.
- Assess **Enterprise applications** to see whether any existing applications use the AWS application type, as shown by `http://aws.amazon.com/` in the **Homepage URL** column.

Deploy Defender for Cloud Apps

After you deploy the central management and strong authentication that modern identity and access management require, you can implement Defender for Cloud Apps to:

- Collect security data and carry out threat detections for AWS accounts.
- Implement advanced controls to mitigate risk and prevent data loss.

To deploy Defender for Cloud Apps:

1. Add a Defender for Cloud Apps app connector for AWS.
2. Configure Defender for Cloud Apps monitoring policies for AWS activities.
3. Create an enterprise application for SSO to AWS.
4. Create a conditional access app control application in Defender for Cloud Apps.
5. Configure Microsoft Entra session policies for AWS activities.
6. Test Defender for Cloud Apps policies for AWS.

Add an AWS app connector

1. In the [Defender for Cloud Apps portal](#), expand **Investigate** and then select **Connected apps**.
2. On the **App connectors** page, select the **Plus Sign (+)** and then select **Amazon Web Services** from the list.
3. Use a unique name for the connector. In the name, include an identifier for the company and specific AWS account, for example *Contoso-AWS-Account1*.

4. Follow the instructions at [Connect AWS to Microsoft Defender for Cloud Apps](#) to create an appropriate AWS identity and access management (IAM) user.
 - a. Define a policy for restricted permissions.
 - b. Create a service account to use those permissions on behalf of the Defender for Cloud Apps service.
 - c. Provide the credentials to the app connector.

The time it takes to establish the initial connection depends on the AWS account log sizes. When the connection is complete, you see a connection confirmation:

App	Status	Was connected on	Last activity	Accounts	⋮
 Contoso-AWS-Account1 Cloud computing platform	Connected	Oct 5, 2020, 5:3...	Oct 13, 2020, 7:...	465	⋮

Configure Defender for Cloud Apps monitoring policies for AWS activities

After you turn on the app connector, Defender for Cloud Apps shows new templates and options in the policy configuration builder. You can create policies directly from the templates and modify them for your needs. You can also develop a policy without using the templates.

To implement policies by using the templates:

1. In the Defender for Cloud Apps left navigation window, expand **Control** and then select **Templates**.

Defender for Cloud Apps



Dashboard

Discover



Investigate



Control



Policies

Templates

Alerts

40

2. Search for **aws** and review the available policy templates for AWS.

Policy templates

TYPE  SEVERITY  NAME **aws** CATEGORY  

1 - 10 of 10 Templates  

Template	Severity	Linked policies	Published
 Publicly accessible S3 buckets (AWS) Alert when an S3 bucket in AWS is publicly accessible.		1	Oct 11, 2020, 2:29 A... 
 Virtual Private Network (VPC) changes (AWS) Alert on any API calls made to create, update, or delete an Amazon VPC, an A...		1	Oct 11, 2020, 2:29 A... 
 IAM Policy changes (AWS) Alert on any API calls made to change IAM policy		1	Oct 11, 2020, 2:29 A... 
 Console Sign-in Failures (AWS) Alert of multiple sign-in failures to AWS console.		1	Oct 11, 2020, 2:29 A... 
 CloudTrail changes (AWS) Alert on any API call made to create, update, or delete a CloudTrail trail, or to ...		1	Oct 11, 2020, 2:29 A... 
 EC2 Instance changes (AWS) Alert on any API call is made to create, terminate, start, stop, or reboot an Am...		1	Oct 11, 2020, 2:29 A... 
 Network Gateway changes (AWS) Alert on API call made to create, update, or delete customer's internet gateway.		1	Oct 11, 2020, 2:29 A... 
 Network Access Control List (ACL) changes (AWS) Alert on any configuration changes involving Network ACLs.		1	Oct 11, 2020, 2:29 A... 
 S3 Bucket Activity (AWS) Alert when AWS S3 API call is made to PUT or DELETE bucket policy, bucket lif...		1	Oct 11, 2020, 2:29 A... 
 Security Group Configuration changes (AWS) Alert on configuration changes which involve security groups.		1	Oct 11, 2020, 2:29 A... 

3. To use a template, select the **Plus Sign (+)** to the right of the template item.
4. Each policy type has different options. Review the configuration settings and save the policy. Repeat this step for each template.

Create file policy



Policy template

Publicly accessible S3 buckets (AWS)

Policy name

Publicly accessible S3 buckets (AWS)

Description

Alert when an S3 bucket in AWS is publicly accessible.

Policy severity

Medium

Category

Sharing control

Create a filter for the files this policy will act on

FILES MATCHING ALL OF THE FOLLOWING

Edit and preview results

Access level	equals	Public (Internet), Public
App	equals	Amazon Web Services

Apply to:

all files

Apply to:

all file owners

Inspection method

None

To use file policies, make sure the file monitoring setting is turned on in Defender for Cloud Apps settings:

Files



Enable file monitoring

This enables Defender for Cloud Apps to see files in your SaaS apps.

As Defender for Cloud Apps detects alerts, it displays them on the **Alerts** page in the Defender for Cloud Apps portal:

Richard 67.184.79.203 United States

RESOLUTION STATUS CATEGORY SEVERITY APP USER NAME POLICY Advanced

OPEN DISMISSED RESOLVED Select ris... Select ap... Select us... Select p...

1 - 6 of 6 alerts

Alert	Resolution	Severity	Date
CloudTrail changes (AWS) CloudTrail changes (AWS) Contoso-AWS-Account1 aws root user 67.184.79.203 United States	OPEN	Low	10/12/20, ...
S3 Bucket Activity (AWS) S3 Bucket Activity (AWS) Contoso-AWS-Account1 aws root user 67.184.79.203 United States	OPEN	Low	10/12/20, ...
IAM Policy changes (AWS) IAM Policy changes (AWS) Contoso-AWS-Account1 aws root user 67.184.79.203 United States	OPEN	Low	10/12/20, ...
Activity from infrequent country Activity from infrequent co... Microsoft Defender for Cloud Apps Richard 67.184.79.203 United	OPEN	Medium	10/12/20, ...
Suspicious administrative activity Unusual administrative acti... Contoso-AWS-Account1 aws root user 67.184.79.203	OPEN	Medium	10/9/20, 3...
Block upload of potential malware (based on Microsoft Threat Intelligence) Block upload of potential m... Amazon WorkDocs - General Richard 67.184.79.203 United State...	OPEN	High	10/8/20, 8...

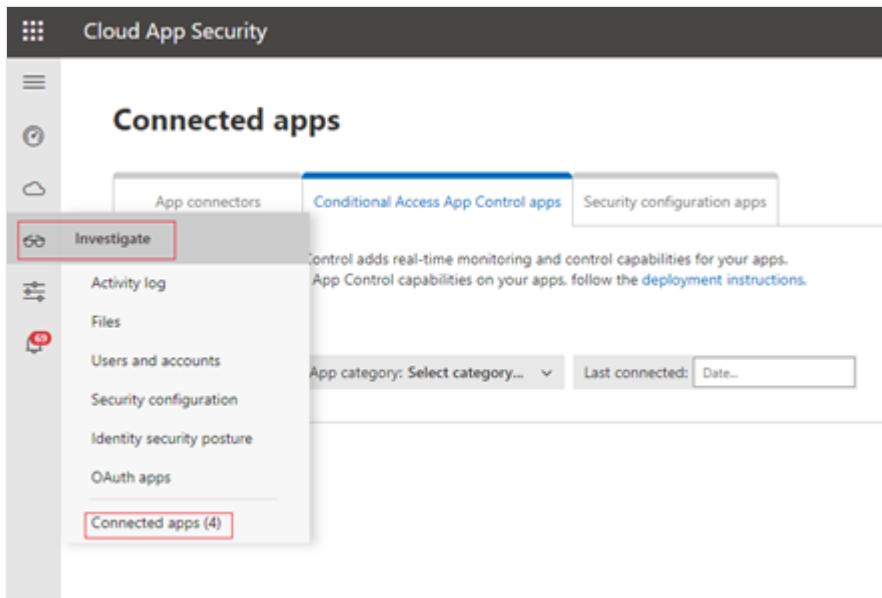
Create an enterprise application for SSO to AWS

Follow the instructions at [Tutorial: Microsoft Entra single sign-on \(SSO\) integration with AWS single sign-on](#) to create an enterprise application for SSO to AWS. Here's a summary of the procedure:

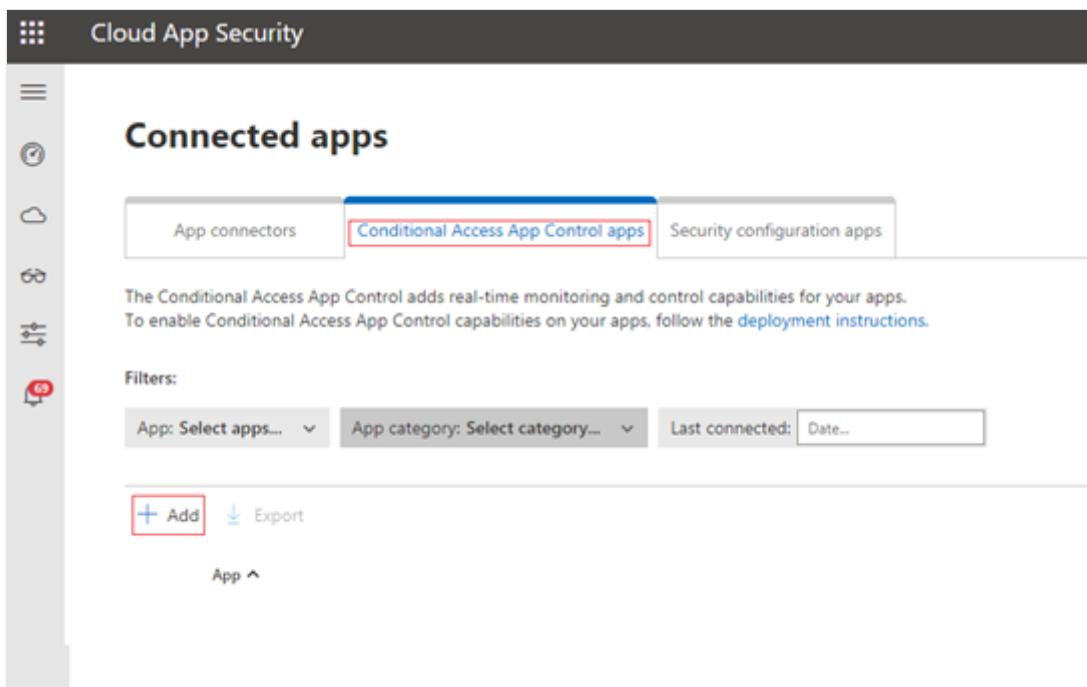
1. Add AWS SSO from the gallery.
2. Configure and test Microsoft Entra SSO for AWS SSO:
 - a. Configure Microsoft Entra SSO.
 - b. Configure AWS SSO.
 - c. Create an AWS SSO test user.
 - d. Test SSO.

Create a conditional access app control application in Defender for Cloud Apps

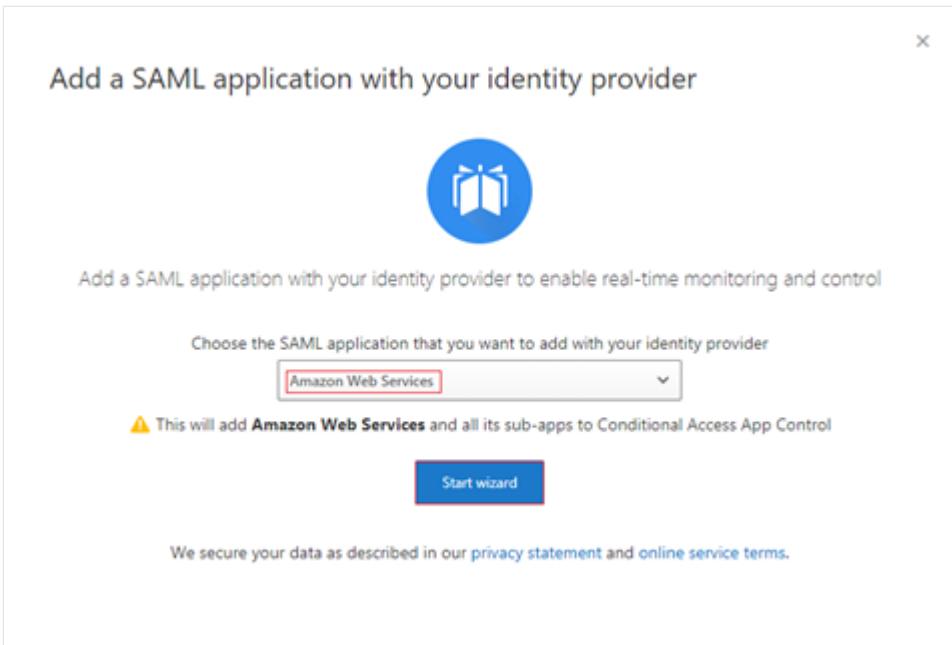
1. Go to the [Defender for Cloud Apps portal](#), select **Investigate**, and then select **Connected apps**.



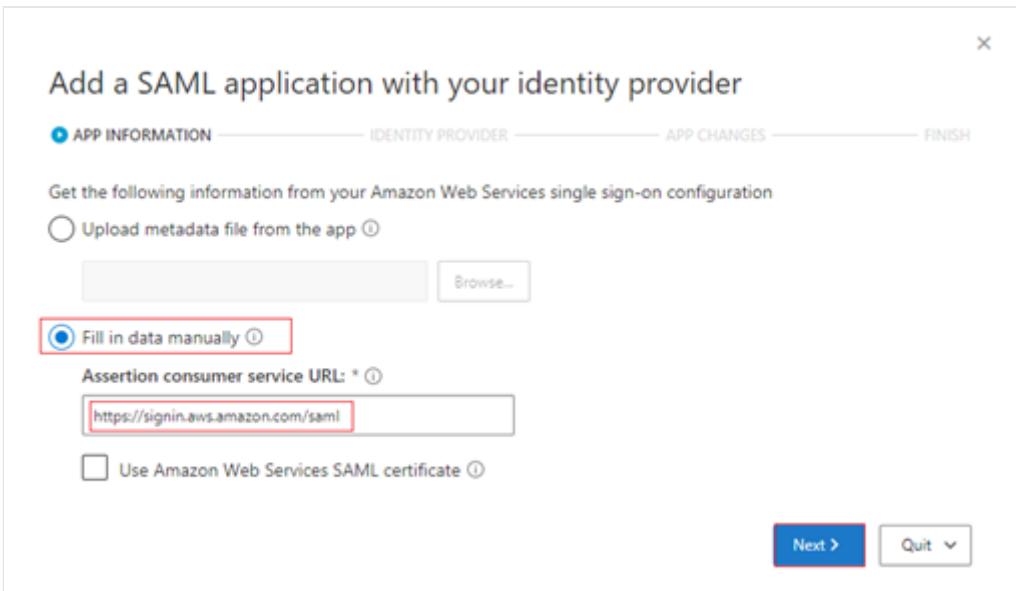
2. Select **Conditional Access App Control apps**, and then select **Add**.



3. In the **Search for an app** box, enter **Amazon Web Services**, and then select the application. Select **Start wizard**.



4. Select **Fill in data manually**. Enter the **Assertion consumer service URL** value that's shown in the following screenshot, and then select **Next**.



5. On the next page, ignore the **External configuration** steps. Select **Next**.

Add a SAML application with your identity provider

APP INFORMATION

IDENTITY PROVIDER

APP CHANGES

FINISH



External configuration

Perform the following steps in your identity provider's portal, under **Applications**

- 1 Go to your identity provider's portal and **create a new custom SAML app** ⓘ
- 2 **Copy the single sign-on configuration** of the existing Amazon Web Services app to the new custom app ⓘ
- 3 **Assign users** to the new custom app

◀ Previous

Next ➤

Quit ▾

6. Select **Fill in data manually**, and then take the following steps to enter the data:
 - a. Under **Single sign-on service URL**, enter the **Login URL** value for the enterprise application that you created for AWS.
 - b. Under **Upload identity provider's SAML certificate**, select **Browse**.
 - c. Locate the certificate for the enterprise application that you created.
 - d. Download the certificate to your local device, and then upload it to the wizard.
 - e. Select **Next**.

LAB1-AWS Single-Account Access | SAML-based Sign-on ...

Enterprise Application

Overview

Deployment Plan

Manage

Properties

Owners

Roles and administrators (Preview)

Users and groups

Single sign-on

Provisioning

Self-service

App Federation Metadata Url <https://login.microsoftonline.com/69e13e16-254d...>

Certificate (Base64) [Download](#)

Certificate (Raw) [Download](#)

Federation Metadata XML [Download](#)

4

Set up LAB1-AWS Single-Account Access

You'll need to configure the application to link with Azure AD.

Login URL <https://login.microsoftonline.com/69e13e16-254d...>

Azure AD identifier <https://sts.windows.net/69e13e16-254d-49f5-97d...>

Logout URL <https://login.microsoftonline.com/69e13e16-254d...>

[View step-by-step instructions](#)

Add a SAML application with your identity provider

APP INFORMATION IDENTITY PROVIDER APP CHANGES FINISH

Get the following information from your identity provider's new custom app configuration

Upload metadata file from your identity provider

Fill in data manually

Single sign-on service URL: <https://login.microsoftonline.com/69e13e16-254d...>

Upload identity provider's SAML certificate: [LAB1-AWS Single-Account Access.cer](#) [Browse...](#)

[Previous](#) [Next >](#) [Quit](#)

7. On the next page, ignore the **External configuration** steps. Select **Next**.

Add a SAML application with your identity provider

APP INFORMATION IDENTITY PROVIDER APP CHANGES FINISH

External configuration
Perform the following steps in your identity provider's portal, under the **custom app settings**

- 1 Copy the following URL. Go to your identity provider's portal and paste it as the **single sign-on URL** in the new custom SAML app you created: https://eu2.saml.cas.ms/saml/sso_login_consumer?orig_consumer=https%3A%2F%2Fsignin.aws.amazon.com%2F
- 2 Add the following **attributes and values**:

Attribute (case sensitive):	McasSigningCert	Value:	MIIIfjCCA54CAQEcDQYJKoZ
Attribute (case sensitive):	McasAppId	Value:	2146446671
- 3 Verify that the **name identifier** is in email address format
- 4 Save your settings

[Previous](#) [Next >](#) [Quit](#)

8. On the next page, ignore the **External configuration** steps. Select **Finish**.

Add a SAML application with your identity provider

APP INFORMATION IDENTITY PROVIDER APP CHANGES FINISH

 **External configuration**
Perform the following steps in Amazon Web Services, under [single sign-on settings](#)

- 1 Go to [Amazon Web Services single sign-on](#) settings and create a backup of your current settings (Recommended)
- 2 Copy the following URL and paste it as the **SAML single sign-on URL**: ⓘ
https://eu2.saml.cas.ms/saml/sso_login?orig_idp=https%3A%2F%2Flogin.microsoftonline.com%2F69e13e16-25 ⓘ
- 3 Download the [Cloud App Security SAML certificate](#) for the app
- 4 Upload the new certificate to Amazon Web Services instead of the previous identity provider SAML certificate
- 5 Save your settings.
After you save your changes, all login requests will be routed through Conditional Access App Control (this may take a few minutes).

We secure your data as described in our [privacy statement](#) and [online service terms](#).

[Previous](#) [Finish >](#) [Quit ▾](#)

9. On the next page, ignore the **Verify your settings** steps. Select **Close**.

Add a SAML application with your identity provider

 Congratulations!

You successfully added Amazon Web Services with your identity provider

Verify your settings

- 1 Log in to [Amazon Web Services](#) ⓘ. If the login fails, verify that you completed all the wizard steps correctly.
- 2 Make sure your login appears in the [Activity log](#) ⓘ. If not, wait a few minutes and try to log in again.

What's next?

- 1 Add a root certificate to identify your [managed devices](#) ⓘ ⓘ
- 2 Update your [access policies](#) ⓘ with the new app or create a new [access policy](#) ⓘ

[Previous](#) [Close](#)

Configure Microsoft Entra session policies for AWS activities

Session policies are a powerful combination of Microsoft Entra Conditional Access policies and the reverse proxy capability of Defender for Cloud Apps. These policies provide real-time suspicious behavior monitoring and control.

1. In Microsoft Entra ID, create a new conditional access policy with the following settings:

- Under **Name**, enter **AWS Console – Session Controls**.
- Under **Users and Groups**, select the two role groups that you created earlier:
 - **AWS-Account1-Administrators**
 - **AWS-Account1-Developers**
- Under **Cloud apps or actions**, select the enterprise application that you created earlier, such as **Contoso-AWS-Account 1**.
- Under **Session**, select **Use Conditional Access App Control**.

2. Under **Enable policy**, select **On**.

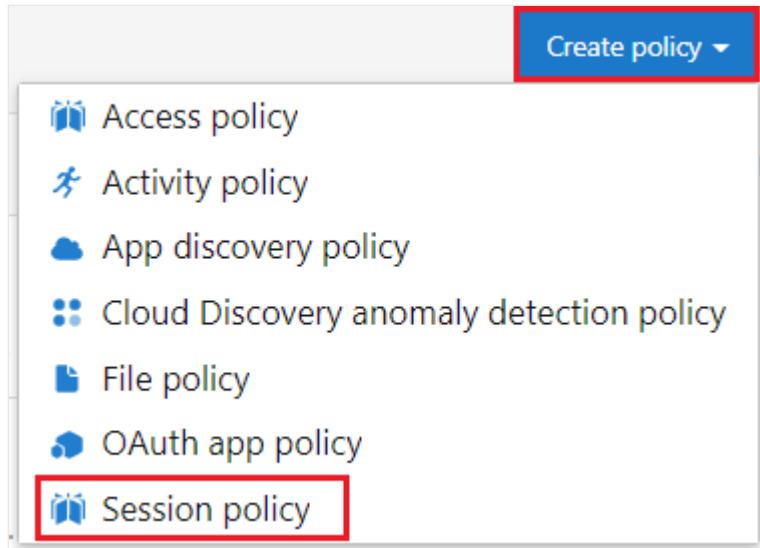
The screenshot shows the 'AWS Console - Session Controls' conditional access policy configuration. The 'Session' tab is selected. The 'Enable policy' section at the bottom is highlighted with a red box, showing the 'On' radio button selected. The 'Session' control settings are visible on the right, including options for app enforced restrictions (unchecked), Conditional Access App Control (checked), and a 'Use custom policy...' dropdown. A note indicates that this control only works with supported apps (Office 365, Exchange Online, SharePoint Online). The 'Access controls' section shows 'Grant' and 'Session' controls, with 'Session' being the active tab. A note states that custom policies need to be configured in the Cloud App Security portal. The 'Configure custom policy' section includes 'Sign-in frequency' and 'Persistent browser session' checkboxes. A warning note at the bottom states that this policy impacts the Azure AD device registration service, so session controls that require device registration are not available.

3. Select **Create**.

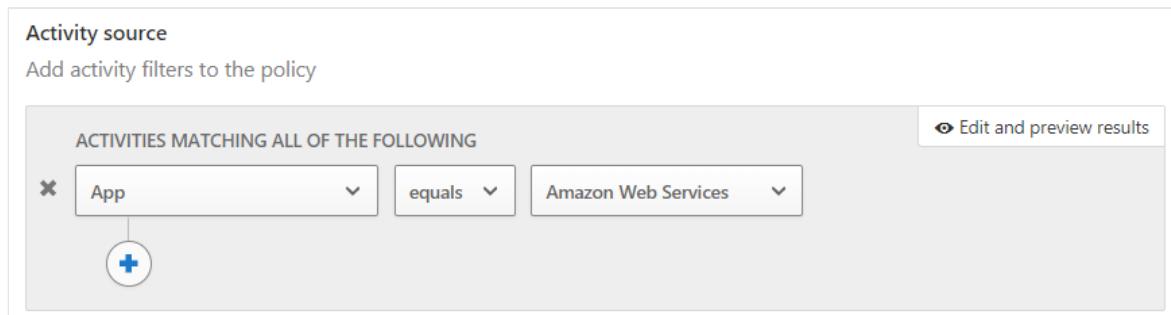
After you create the Microsoft Entra Conditional Access policy, set up a Defender for Cloud Apps session policy to control user behavior during AWS sessions.

1. In the Defender for Cloud Apps portal, expand **Control** and then select **Policies**.

2. On the **Policies** page, select **Create policy** and then select **Session policy** from the list.



3. On the **Create session policy** page, under **Policy template**, select **Block upload of potential malware (based on Microsoft Threat Intelligence)**.
4. Under **Activities matching all of the following**, modify the activity filter to include **App**, **equals**, and **Amazon Web Services**. Remove the default device selection.



5. Review the other settings, and then select **Create**.

Test Defender for Cloud Apps policies for AWS

Test all policies regularly to ensure that they're still effective and relevant. Here are a few recommended tests:

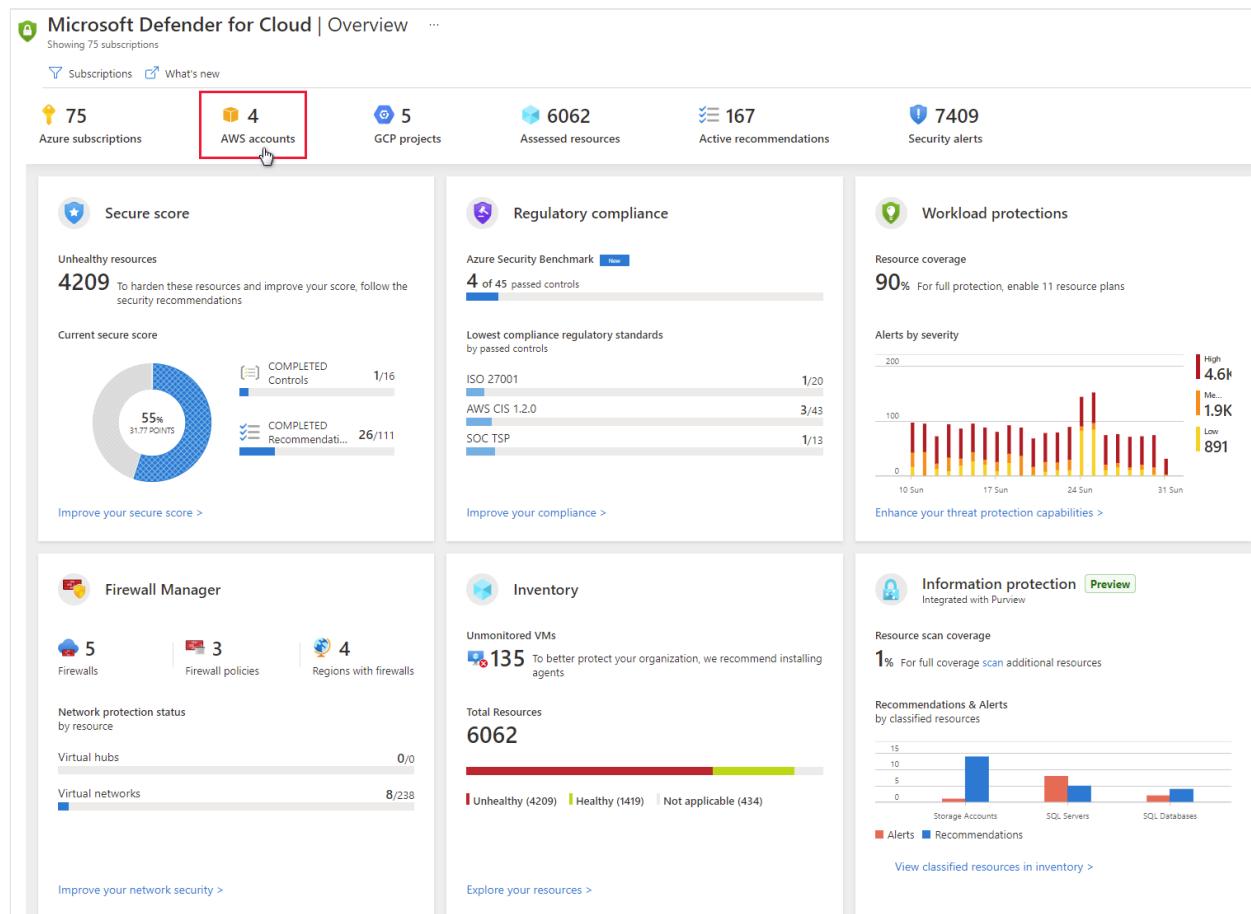
- **IAM policy changes:** This policy is triggered each time that you attempt to modify the settings within AWS IAM. For instance, when you follow the procedure later in this deployment section to create a new IAM policy and account, you see an alert.
- **Console sign-in failures:** Any failed attempts to sign in to one of the test accounts trigger this policy. The alert details show that the attempt came from one of the Azure regional datacenters.

- S3 bucket activity policy: When you attempt to create a new AWS S3 storage account and set it to be publicly available, you trigger this policy.
- Malware detection policy: If you configure malware detection as a session policy, you can test it by following these steps:
 1. Download a safe test file from the [European Institute for Computer Anti-Virus Research \(EICAR\)](#).
 2. Try to upload that file to an AWS S3 storage account.

The policy immediately blocks the upload attempt, and an alert appears in the Defender for Cloud Apps portal.

Deploy Defender for Cloud

You can use a native cloud connector to connect an AWS account to Defender for Cloud. The connector provides an agentless connection to your AWS account. You can use this connection to gather CSPM recommendations. By using Defender for Cloud plans, you can secure your AWS resources with CWP.



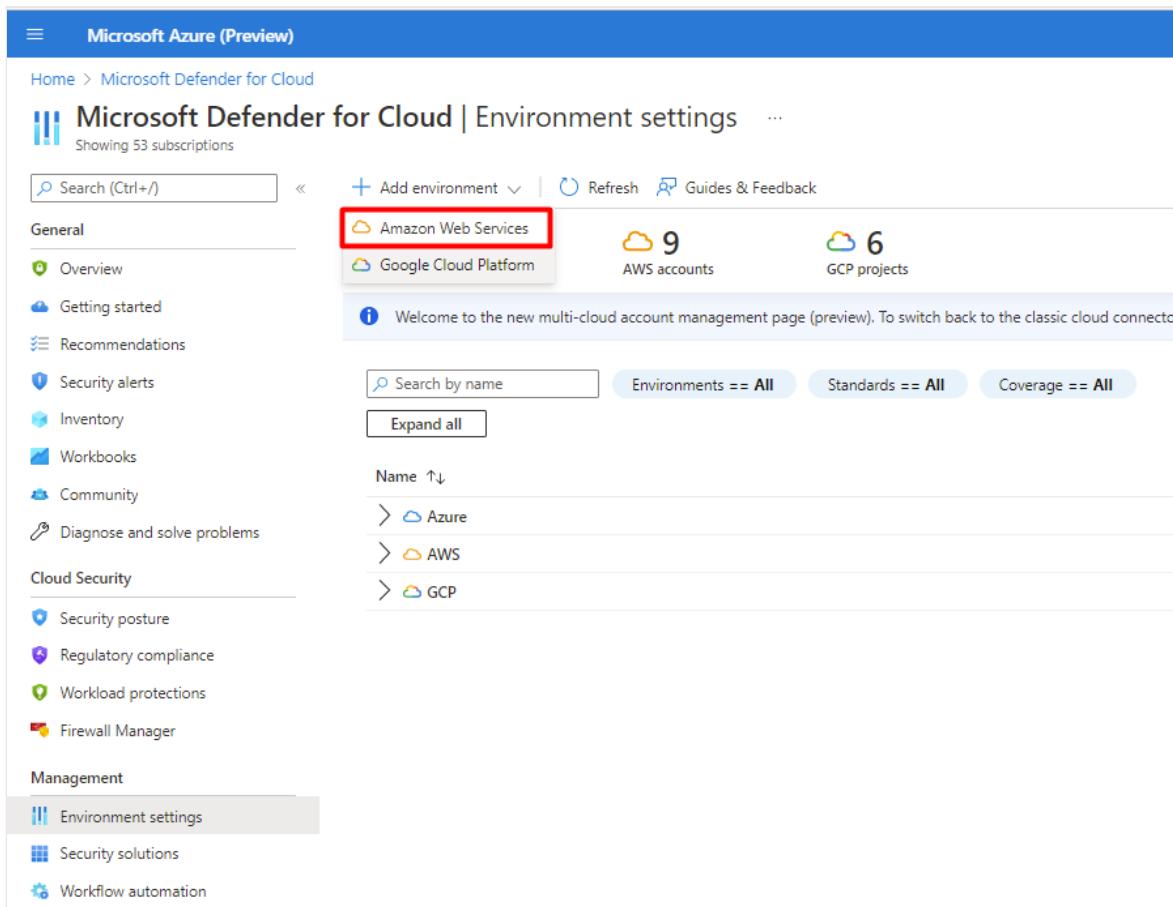
To protect your AWS-based resources, take these steps, which the following sections describe in detail:

1. Connect an AWS account.
2. Monitor AWS.

Connect your AWS account

To connect your AWS account to Defender for Cloud by using a native connector, follow these steps:

1. Review the [prerequisites](#) for connecting an AWS account. Ensure that you complete them before you proceed.
2. If you have any classic connectors, remove them by following the steps in [Remove classic connectors](#). Using both the classic and native connectors can produce duplicate recommendations.
3. Sign in to the [Azure portal](#).
4. Select **Microsoft Defender for Cloud**, and then select **Environment settings**.
5. Select **Add environment > Amazon Web Services**.



The screenshot shows the Microsoft Defender for Cloud Environment settings page. The 'Amazon Web Services' button is highlighted with a red box. The page displays 9 AWS accounts and 6 GCP projects. The left sidebar shows a navigation menu with 'Environment settings' selected. The main content area includes a search bar, a 'Welcome' message, and a list of environments.

Environment	Count
AWS accounts	9
GCP projects	6

Environment settings

Name
Azure
AWS
GCP

6. Enter the details of the AWS account, including the storage location of the connector resource. Optionally, select **Management account** to create a connector to a management account. Connectors are created for each member account that's

discovered under the provided management account. Auto-provisioning is turned on for all newly onboarded accounts.

Microsoft Azure (Preview)

Home > Microsoft Defender for Cloud >

Add account

Amazon Web Services

1 Account details 2 Select plans 3 Configure access 4 Review and generate

Enter a descriptive name for the cloud account connector and choose where to save the connector resource.

Connector name *

Onboard *

Subscription *

Resource group *

Location *

AWS account Id *

Excluded accounts

7. Select Next: Select plans.

Microsoft Azure (Preview)

Home > Microsoft Defender for Cloud >

Add account

Amazon Web Services

1 Account details 2 Select plans 3 Configure access 4 Review and generate

Select plans

Select the desired capabilities. Each capability will require different access permissions and might incur charges.

Plan name & Description	Configurations	Pricing	Plan status
Security posture management	Permissions: Read (SecurityAudit)	Free (preview)	<input type="button" value="On"/> <input type="button" value="Off"/>
Servers	<input checked="" type="checkbox"/> Auto-provisioning enabled Configure >	X\$/Server/Month	<input checked="" type="button" value="On"/> <input type="button" value="Off"/>
Containers	<input checked="" type="checkbox"/> Audit logs enabled Will incur additional AWS costs <small>ⓘ</small> Configure >	Free (preview)	<input checked="" type="button" value="On"/> <input type="button" value="Off"/>

< Previous Next : Configure access >

8. By default, the servers plan is turned on. This setting is necessary to extend Defender for Servers coverage to your AWS EC2. Ensure you've fulfilled the [network requirements for Azure Arc](#). Optionally, to edit the configuration, select [Configure](#).

9. By default, the containers plan is turned on. This setting is necessary to have Defender for Containers protection for your AWS EKS clusters. Ensure you've fulfilled the [network requirements](#) for the Defender for Containers plan. Optionally, to edit the configuration, select **Configure**. If you disable this configuration, the threat detection feature for the control plane is disabled. To view a list of features, see [Defender for Containers feature availability](#).

10. By default, the databases plan is turned on. This setting is necessary to extend Defender for SQL coverage to your AWS EC2 and RDS Custom for SQL Server. Optionally, to edit the configuration, select **Configure**. We recommend that you use the default configuration.

11. Select **Next: Configure access**.

12. Download the CloudFormation template.

13. Follow the on-screen instructions to use the downloaded CloudFormation template to create the stack in AWS. If you onboard a management account, you need to run the CloudFormation template as Stack and as StackSet. Connectors are created for the member accounts within 24 hours of onboarding.

14. Select **Next: Review and generate**.

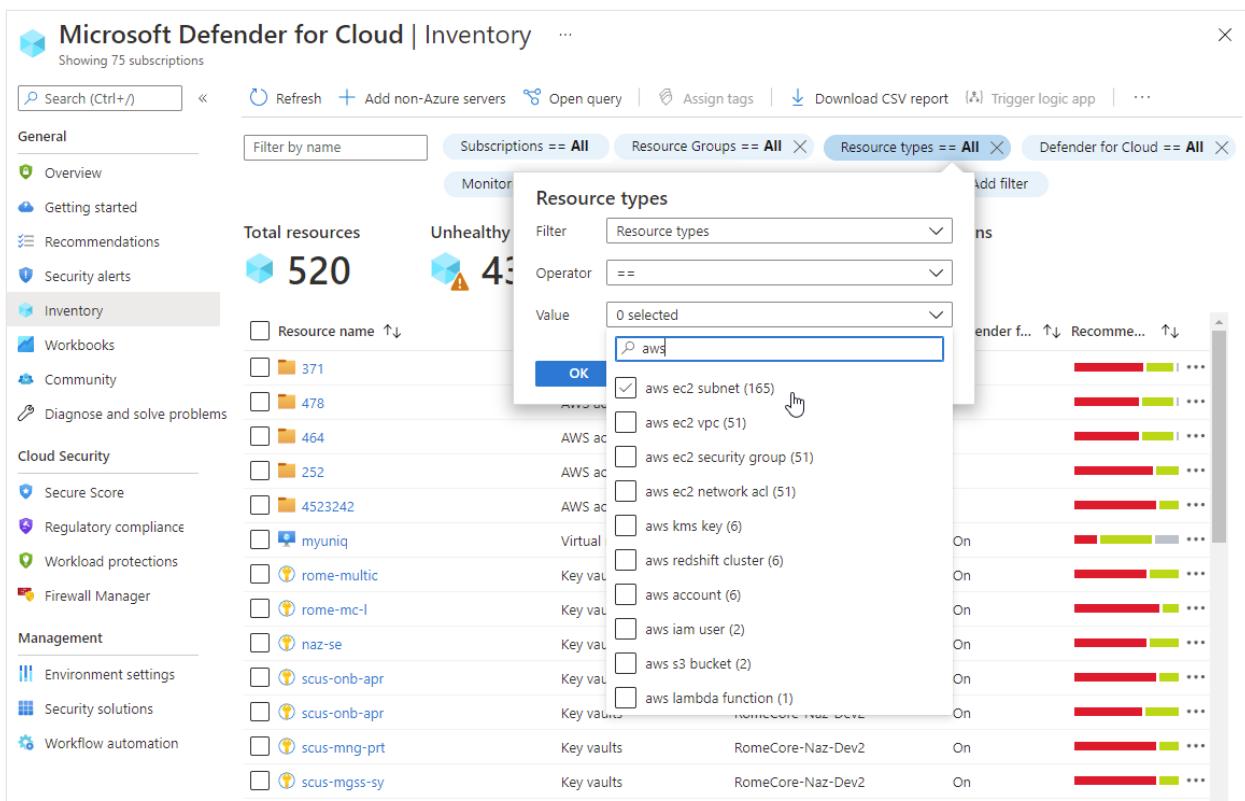
15. Select **Create**.

Defender for Cloud immediately starts scanning your AWS resources. Within a few hours, you see security recommendations. For a list of all the recommendations Defender for Cloud can provide for AWS resources, see [Security recommendations for AWS resources - a reference guide](#).

Monitor your AWS resources

The Defender for Cloud security recommendations page displays your AWS resources. You can use the environments filter to take advantage of the multicloud capabilities of Defender for Cloud, such as viewing the recommendations for Azure, AWS, and GCP resources together.

To view all the active recommendations for your resources by resource type, use the Defender for Cloud asset inventory page. Set the filter to display the AWS resource type that you're interested in.



The screenshot shows the Microsoft Defender for Cloud Inventory interface. On the left, a sidebar lists various sections: General (Overview, Getting started, Recommendations, Security alerts, Inventory, Workbooks, Community, Diagnose and solve problems), Cloud Security (Secure Score, Regulatory compliance, Workload protections, Firewall Manager), and Management (Environment settings, Security solutions, Workflow automation). The 'Inventory' section is currently selected. The main area displays 'Total resources' (520) and 'Unhealthy' resources (45). A search bar at the top is set to 'Search (Ctrl+/' and has a dropdown for 'Filter by name'. Below the search bar are several filter buttons: 'Subscriptions == All', 'Resource Groups == All', 'Resource types == All', and 'Defender for Cloud == All'. A 'Resource types' filter is open, showing a dropdown for 'Resource types', an 'Operator' dropdown set to '==', and a 'Value' dropdown set to '0 selected'. A search input field contains 'aws', with a blue 'OK' button and a list of AWS resources: 'aws ec2 subnet (165)', 'aws ec2 vpc (51)', 'aws ec2 security group (51)', 'aws ec2 network acl (51)', 'aws kms key (6)', 'aws redshift cluster (6)', 'aws account (6)', 'aws iam user (2)', 'aws s3 bucket (2)', and 'aws lambda function (1)'. The list is scrollable, with a 'More' button at the bottom. To the right of the search results, there is a table with columns for 'Name', 'Status', and 'Actions'. The table shows several entries, including 'RomeCore-Naz-Dev2' with 'On' status and 'aws ec2 subnet (165)' selected. A 'More' button is also present in this table.

Deploy Microsoft Sentinel

If you connect an AWS account and Defender for Cloud Apps to Microsoft Sentinel, you can use monitoring capabilities that compare events across multiple firewalls, network devices, and servers.

Enable the Microsoft Sentinel AWS connector

After you enable the Microsoft Sentinel connector for AWS, you can monitor AWS incidents and data ingestion.

As with the Defender for Cloud Apps configuration, this connection requires configuring AWS IAM to provide credentials and permissions.

1. In AWS IAM, follow the steps at [Connect Microsoft Sentinel to AWS CloudTrail](#).
2. To complete the configuration in the Azure portal, under **Microsoft Sentinel > Data connectors**, select the **Amazon Web Services** connector.

Microsoft Azure

Search resources, services, and docs (G+)

Home > Microsoft Sentinel

Microsoft Sentinel | Data connectors

Selected workspace: 'ispectre-loganalytics-azuresentinel'

Search (Ctrl+ /) Refresh

General

- Overview
- Logs
- News & guides

Threat management

- Incidents
- Workbooks
- Hunting
- Notebooks (Preview)
- Entity behavior (Preview)
- Threat intelligence (Preview)

Configuration

- Data connectors**
- Analytics
- Watchlist (Preview)
- Playbooks
- Community
- Settings

60 Connectors 9 Connected 0 Coming soon

Search by name or provider Providers : All

	Status ↑	Connector name ↑↓
	AI Vectra Detect (Preview)	Vectra AI
	Alcide kAudit (Preview)	Alcide
	Amazon Web Services	Amazon
	Azure Active Directory	Microsoft
	Azure Active Directory Identity Protection	Microsoft
	Azure Activity	Microsoft
	Azure Advanced Threat Protection (Preview)	Microsoft
	Azure DDoS Protection (Preview)	Microsoft

3. Select **Open connector page**.
4. Under **Configuration**, enter the **Role ARN** value from the AWS IAM configuration in the **Role to add** field, and select **Add**.
5. Select **Next steps**, and then select the **AWS Network Activities** and **AWS User Activities** activities to monitor.
6. Under **Relevant analytic templates**, select **Create rule** next to the AWS analytic templates that you want to turn on.
7. Set up each rule, and select **Create**.

The following table shows the rule templates that are available for checking AWS entity behaviors and threat indicators. The rule names describe their purpose, and the potential data sources list the data sources that each rule can use.

Expand table

Analytic template name	Data sources
Known IRIDIUM IP	DNS, Azure Monitor, Cisco ASA, Palo Alto Networks, Microsoft Entra ID, Azure Activity, AWS
Full Admin policy created and then attached to Roles, Users, or Groups	AWS
Failed AzureAD logons but success logon to AWS Console	Microsoft Entra ID, AWS
Failed AWS Console logons but success logon to AzureAD	Microsoft Entra ID, AWS
Multifactor authentication disabled for a user	Microsoft Entra ID, AWS
Changes to AWS Security Group ingress and egress settings	AWS
Monitor AWS Credential abuse or hijacking	AWS
Changes to AWS Elastic Load Balancer security groups	AWS
Changes to Amazon VPC settings	AWS
New UserAgent observed in last 24 hours	Microsoft 365, Azure Monitor, AWS
Login to AWS Management Console without multifactor authentication	AWS
Changes to internet facing AWS RDS Database instances	AWS
Changes made to AWS CloudTrail logs	AWS

Analytic template name	Data sources
Defender Threat Intelligence map IP entity to AWS CloudTrail	Defender Threat Intelligence Platforms, AWS

Enabled templates have an **IN USE** indicator on the connector details page.

Relevant analytic templates (14)					Go to analytics template list
Severity ↑↓	Name ↑↓	Rule type ↑↓	Data sources	Tactics	
High	IN USE Known IRIDIUM IP	Scheduled	Office 365 +10	Command and Control	
Medium	IN USE Full Admin policy created and t...	Scheduled	Amazon Web Services	Privilege Escalation	
Medium	IN USE Failed AzureAD logons but suc...	Scheduled	Azure Active Directory +1	Brute Force	

Monitor AWS incidents

Microsoft Sentinel creates incidents based on the analyses and detections that you turn on. Each incident can include one or more events, which reduces the overall number of investigations that are necessary to detect and respond to potential threats.

Microsoft Sentinel shows incidents that Defender for Cloud Apps generates, if it's connected, and incidents that Microsoft Sentinel creates. The **Product names** column shows the incident source.

Microsoft Sentinel | Incidents
Selected workspace: 'ispectre-loganalytics-azuresentinel'

General

- Overview
- Logs
- News & guides

Threat management

- Incidents
- Workbooks
- Hunting
- Notebooks (Preview)

16 Open incidents | 16 New incidents | 0 Active incidents | Open incidents by severity: High (0) | Medium (4) | Low (12) | Informational (0)

Search by id or title

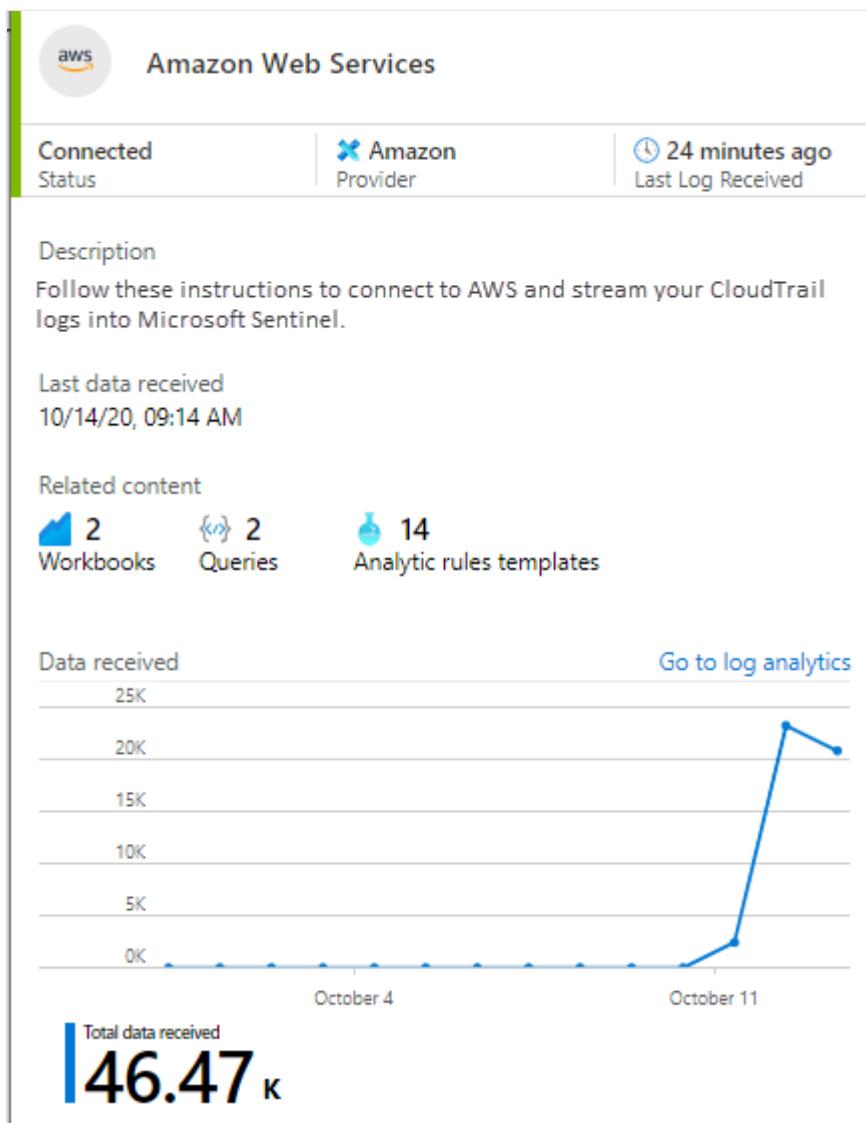
Severity: All | Status: New, Active | Product name: All | Owner: All

Auto-refresh incidents

Incident id ↑↓	Title ↑↓	Alerts	Product names	Created time ↑↓
145	Login to AWS Management Console...	1	Microsoft Sentinel	10/14/20, 09:51 AM
144	Suspicious administrative activity	1	Microsoft Cloud Ap...	10/14/20, 04:41 AM
143	Console Sign-in Failures (AWS)	1	Microsoft Cloud Ap...	10/14/20, 04:36 AM

Check data ingestion

Check that data is continuously ingested into Microsoft Sentinel by regularly viewing the connector details. The following chart shows a new connection.



If the connector stops ingesting data and the line chart value drops, check the credentials that you use to connect to the AWS account. Also check that AWS CloudTrail can still collect the events.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributor.

Principal author:

- [Lavanya Murthy](#) | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- For security guidance from AWS, see [Best practices for securing AWS accounts and resources](#).
- For the latest Microsoft security information, see [Microsoft Security](#).
- For full details of how to implement and manage Microsoft Entra ID, see [Securing Azure environments with Microsoft Entra ID](#).
- For an overview of AWS asset threats and corresponding protective measures, see [How Defender for Cloud Apps helps protect your Amazon Web Services \(AWS\) environment](#).
- For information about connectors and how to establish connections, see these resources:
 - [Connect your AWS accounts to Microsoft Defender for Cloud](#)
 - [New AWS connector in Microsoft Defender for Cloud](#)
 - [Connect AWS to Microsoft Defender for Cloud Apps](#)
 - [Connect Microsoft Sentinel to AWS CloudTrail](#)

Related resources

- For in-depth coverage and comparison of Azure and AWS features, see the [Azure for AWS professionals](#) content set.
- For guidance for deploying Microsoft Entra identity and access solutions for AWS, see [Microsoft Entra identity and access management for AWS](#).

Connect Microsoft Sentinel to Amazon Web Services to ingest AWS service log data

Article • 12/28/2023

Use the Amazon Web Services (AWS) connectors to pull AWS service logs into Microsoft Sentinel. These connectors work by granting Microsoft Sentinel access to your AWS resource logs. Setting up the connector establishes a trust relationship between Amazon Web Services and Microsoft Sentinel. This is accomplished on AWS by creating a role that gives permission to Microsoft Sentinel to access your AWS logs.

This connector is available in two versions: the legacy connector for CloudTrail management and data logs, and the new version that can ingest logs from the following AWS services by pulling them from an S3 bucket (links are to AWS documentation):

- [Amazon Virtual Private Cloud \(VPC\)](#) - [VPC Flow Logs](#)
- [Amazon GuardDuty](#) - [Findings](#)
- [AWS CloudTrail](#) - [Management](#) and [data](#) events
- [AWS CloudWatch](#) - [CloudWatch logs](#)

ⓘ Important

- The Amazon Web Services S3 connector is currently in **PREVIEW**. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for additional legal terms that apply to Azure features that are in beta, preview, or otherwise not yet released into general availability.

S3 connector (new)

This tab explains how to configure the AWS S3 connector. The process of setting it up has two parts: the AWS side and the Microsoft Sentinel side. Each side's process produces information used by the other side. This two-way authentication creates secure communication.

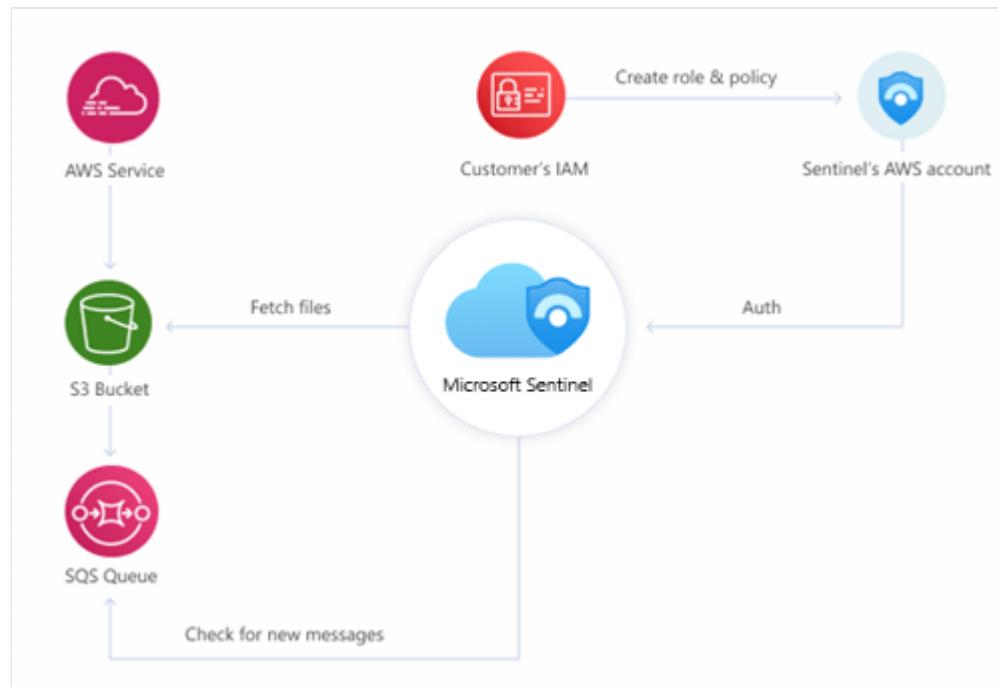
Prerequisites

- Make sure that the logs from your selected AWS service use the format accepted by Microsoft Sentinel:

- **Amazon VPC**: .csv file in GZIP format with headers; delimiter: space.
 - **Amazon GuardDuty**: json-line and GZIP formats.
 - **AWS CloudTrail**: .json file in a GZIP format.
 - **CloudWatch**: .csv file in a GZIP format without a header. If you need to convert your logs to this format, you can use this [CloudWatch Lambda function](#).
- You must have write permission on the Microsoft Sentinel workspace.
 - Install the Amazon Web Services solution from the **Content Hub** in Microsoft Sentinel. For more information, see [Discover and manage Microsoft Sentinel out-of-the-box content](#).

Architecture overview

This graphic and the following text show how the parts of this connector solution interact.



- AWS services are configured to send their logs to S3 (Simple Storage Service) storage buckets.
- The S3 bucket sends notification messages to the SQS (Simple Queue Service) message queue whenever it receives new logs.
- The Microsoft Sentinel AWS S3 connector polls the SQS queue at regular, frequent intervals. If there is a message in the queue, it will contain the path to the log files.

- The connector reads the message with the path, then fetches the files from the S3 bucket.
- To connect to the SQS queue and the S3 bucket, Microsoft Sentinel uses AWS credentials and connection information embedded in the AWS S3 connector's configuration. The AWS credentials are configured with a role and a permissions policy giving them access to those resources. Similarly, the Microsoft Sentinel workspace ID is embedded in the AWS configuration, so there is in effect two-way authentication.

Connect the S3 connector

- In your AWS environment:
 - Configure your AWS service(s) to send logs to an **S3 bucket**.
 - Create a **Simple Queue Service (SQS) queue** to provide notification.
 - Create an **assumed role** to grant permissions to your Microsoft Sentinel account (external ID) to access your AWS resources.
 - Attach the appropriate **IAM permissions policies** to grant Microsoft Sentinel access to the appropriate resources (S3 bucket, SQS).

We have made available, in our GitHub repository, a script that **automates the AWS side of this process**. See the instructions for [automatic setup](#) later in this document.

- In Microsoft Sentinel:
 - Enable and configure the **AWS S3 Connector** in the Microsoft Sentinel portal. See the instructions below.

Automatic setup

To simplify the onboarding process, Microsoft Sentinel has provided a [PowerShell script to automate the setup](#) of the AWS side of the connector - the required AWS resources, credentials, and permissions.

The script takes the following actions:

- Creates an *IAM assumed role* with the minimal necessary permissions, to grant Microsoft Sentinel access to your logs in a given S3 bucket and SQS queue.

- Enables specified AWS services to send logs to that S3 bucket, and notification messages to that SQS queue.
- If necessary, creates that S3 bucket and that SQS queue for this purpose.
- Configures any necessary IAM permissions policies and applies them to the IAM role created above.

Prerequisites for automatic setup

- You must have PowerShell and the AWS CLI on your machine.
 - [Installation instructions for PowerShell](#)
 - [Installation instructions for the AWS CLI](#) (from AWS documentation)

Instructions

To run the script to set up the connector, use the following steps:

1. From the Microsoft Sentinel navigation menu, select **Data connectors**.
2. Select **Amazon Web Services S3** from the data connectors gallery.
If you don't see the connector, install the Amazon Web Services solution from the **Content Hub** in Microsoft Sentinel.
3. In the details pane for the connector, select **Open connector page**.
4. In the **Configuration** section, under **1. Set up your AWS environment**, expand **Setup with PowerShell script (recommended)**.
5. Follow the on-screen instructions to download and extract the [AWS S3 Setup Script](#) (link downloads a zip file containing the main setup script and helper scripts) from the connector page.
6. Before running the script, run the `aws configure` command from your PowerShell command line, and enter the relevant information as prompted. See [AWS Command Line Interface | Configuration basics](#) (from AWS documentation) for details.
7. Now run the script. Copy the command from the connector page (under "Run script to set up the environment") and paste it in your command line.
8. The script will prompt you to enter your Workspace ID. This ID appears on the connector page. Copy it and paste it at the prompt of the script.

Amazon Web Services S3 (Preview) ...

Configuration

1. Set up your AWS environment

There are two options for setting up your AWS environment to send logs from an S3 bucket to your Log Analytics Workspace:

Setup with PowerShell script (recommended)

Download and extract the files from the following link: [AWS S3 Setup Script](#).

1. Make sure that you have PowerShell on your machine: [Installation instructions for PowerShell](#).
2. Make sure that you have the AWS CLI on your machine: [Installation instructions for the AWS CLI](#).

Before running the script, run the aws configure command from your PowerShell command line, and enter the relevant information as prompted. See [AWS Command Line Interface | Configuration basics](#) for details.

Run script to set up the environment

7. ./ConfigAwsConnector.ps1

External ID (Workspace ID)

8. 12345678-abcd-abcd-123456abcdef

9. When the script finishes running, copy the **Role ARN** and the **SQS URL** from the script's output (see example in first screenshot below) and paste them in their respective fields in the connector page under **2. Add connection** (see second screenshot below).

```

Updating the SQS policy to allow S3 notifications, and ARN to read/delete/change visibility of SQS messages and get queue url
Changes S3: SQS SendMessage permission to 'vpc-sentinel-demo' s3 bucket
Changes Role ARN: SQS ChangeMessageVisibility, DeleteMessage, ReceiveMessage and GetQueueUrl permissions to 'SentinelDemo' rule
Attaching S3 read policy to Sentinel role.

Changes Role arn: S3 Get and List permissions to 'SentinelDemo' rule

Updating the S3 policy to allow Sentinel to read the data.
Changes: S3 Get and List permissions to 'SentinelDemo' rule

Enabling S3 event notifications (for *.gz file)

Please enter the event notifications name: demo
Using event notification name: demo
Event notification prefix definition, to Limit the notifications to objects with key starting with specified characters.

The default prefix is 'AWSLogs/123412341234/vpcflowlogs/'.
Do you want to override the event notification prefix? [y/n]: n

Use the values below to configure the Amazon Web Service S3 data connector in the Azure Sentinel portal.

Role Arn: arn:aws:iam::123412341234:role/SentinelDemo
Sqs Url: https://sqs.us-east-1.amazonaws.com/123412341234/vpc-sentinel-demo
Destination Table: AWSVPCFlow

```

10. Select a data type from the **Destination table** drop-down list. This tells the connector which AWS service's logs this connection is being established to collect, and into which Log Analytics table it will store the ingested data. Then select **Add connection**.

(!) Note

The script may take up to 30 minutes to finish running.

Manual setup

Microsoft recommends using the automatic setup script to deploy this connector. If for whatever reason you do not want to take advantage of this convenience, follow the steps below to set up the connector manually.

- [Prepare your AWS resources](#)
- [Create an AWS assumed role and grant access to the AWS Sentinel account](#)
- [Add the AWS role and queue information to the S3 data connector](#)
- [Configure an AWS service to export logs to an S3 bucket](#)

Prepare your AWS resources

- Create an **S3 bucket** to which you will ship the logs from your AWS services - VPC, GuardDuty, CloudTrail, or CloudWatch.

- See the [instructions to create an S3 storage bucket](#) in the AWS documentation.
- Create a standard **Simple Queue Service (SQS)** message queue to which the S3 bucket will publish notifications.
 - See the [instructions to create a standard Simple Queue Service \(SQS\) queue](#) in the AWS documentation.
- Configure your S3 bucket to send notification messages to your SQS queue.
 - See the [instructions to publish notifications to your SQS queue](#) in the AWS documentation.

Create an AWS assumed role and grant access to the AWS Sentinel account

1. In Microsoft Sentinel, select **Data connectors** from the navigation menu.
2. Select **Amazon Web Services S3** from the data connectors gallery.

If you don't see the connector, install the Amazon Web Services solution from the **Content Hub** in Microsoft Sentinel. For more information, see [Discover and manage Microsoft Sentinel out-of-the-box content](#).
3. In the details pane for the connector, select **Open connector page**.
4. Under **Configuration**, expand **Setup with PowerShell script (recommended)**, then copy the **External ID (Workspace ID)** to your clipboard.
5. In a different browser window or tab, open the AWS console. Follow the [instructions in the AWS documentation for creating a role for an AWS account](#).
 - For the account type, instead of **This account**, choose **Another AWS account**.
 - In the **Account ID** field, enter the number **197857026523** (you can copy and paste it from here). This number is **Microsoft Sentinel's service account ID for AWS**. It tells AWS that the account using this role is a Microsoft Sentinel user.
 - In the options, select **Require external ID** (*do not select Require MFA*). In the **External ID** field, paste your Microsoft Sentinel **Workspace ID** that you copied in the previous step. This identifies *your specific Microsoft Sentinel account* to AWS.

- Assign the necessary permissions policies. These policies include:
 - `AmazonSQSReadOnlyAccess`
 - `AWSLambdaSQSQueueExecutionRole`
 - `AmazonS3ReadOnlyAccess`
 - `ROSAKMSProviderPolicy`
 - Additional policies for ingesting the different types of AWS service logs.

For information on these policies, see the [AWS S3 connector permissions policies page](#) in the Microsoft Sentinel GitHub repository.

- Name the role with a meaningful name that includes a reference to Microsoft Sentinel. Example: "*MicrosoftSentinelRole*".

Add the AWS role and queue information to the S3 data connector

1. In the browser tab open to the AWS console, enter the **Identity and Access Management (IAM)** service and navigate to the list of **Roles**. Select the role you created above.
2. Copy the **ARN** to your clipboard.
3. Enter the **Simple Queue Service**, select the SQS queue you created, and copy the **URL** of the queue to your clipboard.
4. Return to your Microsoft Sentinel browser tab, which should be open to the **Amazon Web Services S3 (Preview)** data connector page. Under **2. Add connection**:
 - a. Paste the IAM role ARN you copied two steps ago into the **Role to add** field.
 - b. Paste the URL of the SQS queue you copied in the last step into the **SQS URL** field.
 - c. Select a data type from the **Destination table** drop-down list. This tells the connector which AWS service's logs this connection is being established to collect, and into which Log Analytics table it will store the ingested data.
 - d. Select **Add connection**.

Configure an AWS service to export logs to an S3 bucket

See Amazon Web Services documentation (linked below) for the instructions for sending each type of log to your S3 bucket:

- [Publish a VPC flow log to an S3 bucket](#).

ⓘ Note

If you choose to customize the log's format, you must include the *start* attribute, as it maps to the *TimeGenerated* field in the Log Analytics workspace. Otherwise, the *TimeGenerated* field will be populated with the event's *ingested time*, which doesn't accurately describe the log event.

- [Export your GuardDuty findings to an S3 bucket](#).

ⓘ Note

- In AWS, findings are exported by default every 6 hours. Adjust the export frequency for updated Active findings based on your environment requirements. To expedite the process, you can modify the default setting to export findings every 15 minutes. See [Setting the frequency for exporting updated active findings](#).

- The *TimeGenerated* field is populated with the finding's *Update at* value.
- AWS CloudTrail trails are stored in S3 buckets by default.
 - [Create a trail for a single account](#).
 - [Create a trail spanning multiple accounts across an organization](#).
- [Export your CloudWatch log data to an S3 bucket](#).

Known issues and troubleshooting

Known issues

- Different types of logs can be stored in the same S3 bucket, but should not be stored in the same path.
- Each SQS queue should point to one type of message, so if you want to ingest GuardDuty findings *and* VPC flow logs, you should set up separate queues for each type.
- Similarly, a single SQS queue can serve only one path in an S3 bucket, so if for any reason you are storing logs in multiple paths, each path requires its own dedicated SQS queue.

Troubleshooting

Learn how to [troubleshoot Amazon Web Services S3 connector issues](#).

Next steps

In this document, you learned how to connect to AWS resources to ingest their logs into Microsoft Sentinel. To learn more about Microsoft Sentinel, see the following articles:

- Learn how to [get visibility into your data, and potential threats](#).
- Get started [detecting threats with Microsoft Sentinel](#).
- [Use workbooks](#) to monitor your data.

How Defender for Cloud Apps helps protect your Amazon Web Services (AWS) environment

Article • 01/22/2024

Amazon Web Services is an IaaS provider that enables your organization to host and manage their entire workloads in the cloud. Along with the benefits of leveraging infrastructure in the cloud, your organization's most critical assets may be exposed to threats. Exposed assets include storage instances with potentially sensitive information, compute resources that operate some of your most critical applications, ports, and virtual private networks that enable access to your organization.

Connecting AWS to Defender for Cloud Apps helps you secure your assets and detect potential threats by monitoring administrative and sign-in activities, notifying on possible brute force attacks, malicious use of a privileged user account, unusual deletions of VMs, and publicly exposed storage buckets.

Main threats

- Abuse of cloud resources
- Compromised accounts and insider threats
- Data leakage
- Resource misconfiguration and insufficient access control

How Defender for Cloud Apps helps to protect your environment

- Detect cloud threats, compromised accounts, and malicious insiders
- Limit exposure of shared data and enforce collaboration policies
- Use the audit trail of activities for forensic investigations

Control AWS with built-in policies and policy templates

You can use the following built-in policy templates to detect and notify you about potential threats:

[Expand table](#)

Type	Name
Activity policy template	Admin console sign-in failures CloudTrail configuration changes EC2 instance configuration changes IAM policy changes Logon from a risky IP address Network access control list (ACL) changes Network gateway changes S3 configuration changes Security group configuration changes Virtual private network changes
Built-in anomaly detection policy	Activity from anonymous IP addresses Activity from infrequent country Activity from suspicious IP addresses Impossible travel Activity performed by terminated user (requires Microsoft Entra ID as IdP) Multiple failed login attempts Unusual administrative activities Unusual multiple storage deletion activities (preview) Multiple delete VM activities Unusual multiple VM creation activities (preview) Unusual region for cloud resource (preview)
File policy template	S3 bucket is publicly accessible

For more information about creating policies, see [Create a policy](#).

Automate governance controls

In addition to monitoring for potential threats, you can apply and automate the following AWS governance actions to remediate detected threats:

[Expand table](#)

Type	Action
User governance	- Notify user on alert (via Microsoft Entra ID) - Require user to sign in again (via Microsoft Entra ID) - Suspend user (via Microsoft Entra ID)
Data governance	- Make an S3 bucket private - Remove a collaborator for an S3 bucket

For more information about remediating threats from apps, see [Governing connected apps](#).

Protect AWS in real time

Review our best practices for [blocking and protecting the download of sensitive data to unmanaged or risky devices](#).

Connect Amazon Web Services to Microsoft Defender for Cloud Apps

This section provides instructions for connecting your existing Amazon Web Services (AWS) account to Microsoft Defender for Cloud Apps using the connector APIs. For information about how Defender for Cloud Apps protects AWS, see [Protect AWS](#).

You can connect AWS **Security auditing** to Defender for Cloud Apps connections to gain visibility into and control over AWS app use.

Step 1: Configure Amazon Web Services auditing

1. In your [Amazon Web Services console](#), under **Security, Identity & Compliance**, select **IAM**.

AWS services

Find a service by name (for example, EC2, S3, Elastic Beanstalk).

All services

- Compute**
 - EC2
 - EC2 Container Service
 - Lightsail
 - Elastic Beanstalk
 - Lambda
 - Batch
- Storage**
 - S3
 - EFS
 - Glacier
 - Storage Gateway
- Database**
 - RDS
 - DynamoDB
 - ElastiCache
 - Redshift
- Networking & Content Delivery**
 - VPC
 - CloudFront
 - Direct Connect
 - Route 53
- Migration**
 - Application Discovery Service
 - DMS
 - Server Migration
 - Snowball
- Compute**
 - CodeCommit
 - CodeBuild
 - CodeDeploy
 - CodePipeline
- Developer Tools**
 - CloudWatch
 - CloudFormation
 - CloudTrail
 - Config
 - OpsWorks
 - Service Catalog
 - Trusted Advisor
- Management Tools**
 - Inspector
 - Certificate Manager
 - Directory Service
 - WAF & Shield
 - Compliance Reports
- Internet of Things**
 - AWS IoT
- Game Development**
 - GameLift
- Mobile Services**
 - Mobile Hub
 - Cognito
 - Device Farm
 - Mobile Analytics
 - Pinpoint
- Application Services**
 - Step Functions
 - SWF
 - API Gateway
 - Elastic Transcoder
- Analytics**
 - Athena
 - EMR
 - CloudSearch
 - Elasticsearch Service
 - Kinesis
 - Data Pipeline
 - QuickSight
- Business Productivity**
 - WorkDocs
 - WorkMail
 - Amazon Chime
- Desktop & App Streaming**
 - WorkSpaces
 - AppStream 2.0
- Artificial Intelligence**
 - Lex
 - Polly
 - Rekognition
 - Machine Learning
- Security, Identity & Compliance**
 - IAM

2. Select **Users** and then select **Add user**.

Search IAM

Dashboard

Groups

Users

Roles

Policies

Identity providers

Account settings

Credential report

Encryption keys

Add user

Delete users

Find user

username or access key

Showing 0 results

User name	Groups	Password	Last sign-in	Access keys	Creation time
No results					

3. In the **Details** step, provide a new user name for Defender for Cloud Apps. Make sure that under **Access type** you select **Programmatic access** and select **Next Permissions**.

Featured next steps



Manage your costs

Get real-time billing alerts based on your cost and usage budgets. [Start now](#)



Get best practices

Use AWS Trusted Advisor for security, performance, cost and availability best practices. [Start now](#)

What's new?

Announcing AWS Batch

Now generally available, AWS Batch enables developers, scientists, and engineers to process large-scale batch jobs with ease. [Learn more](#)

Announcing Amazon Lightsail

See how this new service allows you to launch and manage your VPS with AWS for a low, predictable price. [Learn more](#)

[See all](#)

AWS Marketplace

Discover, procure, and deploy popular [software products](#) that run on AWS.

Have feedback?

[Submit feedback](#) to tell us about your experience with the AWS Management Console.

Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name* CloudAppSecurityAWS

[+ Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type* **Programmatic access**

Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access

Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required

[Cancel](#)

[Next: Permissions](#)

4. Select **Attach existing policies directly**, and then **Create policy**.

Add user

1 2 3 4 5

Set permissions

Add user to group

Copy permissions from existing user

Attach existing policies directly

[Create policy](#)

[Filter policies](#)

[Search](#)

Showing 668 results

Policy name

Type

Used as

5. Select the **JSON** tab:

Create policy

1 2
Editor Review

A policy defines the AWS permissions that can be assigned to a user, group, role, or resource. You can construct a policy using the visual editor or create a policy document using the JSON editor.

[Visual editor](#)

[JSON](#)

[Import managed policy](#)

Use the visual editor to create and edit a policy by choosing services, actions, resources, and request conditions to add permissions to your policy. You can add multiple permission blocks to define complex permissions or to grant access to more than one service. [Learn more](#)

[Expand all](#) | [Collapse all](#)

[Select a service](#)

[Clone](#) | [Remove](#)

Service Choose a service

Actions Choose a service before defining actions

Resources Choose actions before applying resources

Request conditions Choose actions before specifying conditions

[+ Add additional permissions](#)

[Cancel](#) [Review policy](#)

6. Paste the following script into the provided area:

JSON

```
{  
  "Version" : "2012-10-17",  
  "Statement" : [  
    {"Action" : [  
      "cloudtrail:DescribeTrails",  
      "cloudtrail:LookupEvents",  
      "cloudtrail:GetTrailStatus",  
      "cloudwatch:Describe*",  
      "cloudwatch:Get*",  
      "cloudwatch>List*",  
      "iam>List*",  
      "iam:Get*",  
      "s3>ListAllMyBuckets",  
      "s3:PutBucketAcl",  
      "s3:GetBucketAcl",  
      "s3:GetBucketLocation"  
    ],  
    "Effect" : "Allow",  
    "Resource" : "*"  
  }  
]  
}
```

7. Select Next: Tags

Create policy

1 2 3

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor JSON Import managed policy

```
1  {  
2   "Version" : "2012-10-17",  
3   "Statement" : [  
4     {"Action" : [  
5       "cloudtrail:DescribeTrails",  
6       "cloudtrail:LookupEvents",  
7       "cloudtrail:GetTrailStatus",  
8       "cloudwatch:Describe*",  
9       "cloudwatch:Get*",  
10      "cloudwatch>List*",  
11      "iam>List*",  
12      "iam:Get*",  
13      "s3>ListAllMyBuckets",  
14      "s3:PutBucketAcl",  
15      "s3:GetBucketAcl",  
16      "s3:GetBucketLocation"  
17    ],  
18    "Effect" : "Allow",  
19    "Resource" : "*"  
20  }  
21]  
22}
```

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

Character count: 329 of 6,144. Cancel Next: Tags

8. Select Next: Review.

Create policy

1 2 3

Add tags (Optional)
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add tag
You can add up to 50 more tags

Cancel **Previous** **Next: Review**

9. Provide a **Name** and select **Create policy**.

Create policy

1 2

Editor **Review**

Review policy
Before you create this policy, provide the required information and review this policy.

Name* Maximum 128 characters. Use alphanumeric and '+, -, @, _' characters.

Description Maximum 1000 characters. Use alphanumeric and '+, -, @, _' characters.

Summary

Service	Access level	Resource	Request condition
CloudTrail	Full: List Limited: Read	All resources	None
CloudWatch	Full: List, Read	All resources	None
IAM	Full: List Limited: Read	All resources	None

* Required **Create policy**

10. Back in the **Add user** screen, refresh the list if necessary, and select the user you created, and select **Next: Tags**.

Add user

1 2 3 4 5

Set permissions

 Add user to group  Copy permissions from existing user  Attach existing policies directly

Create policy 

Filter policies Showing 3 results

	Policy name	Type	Used as
<input checked="" type="checkbox"/>	CloudAppSecurityPolicy	Customer managed	Permissions policy (2)

Set permissions boundary

Cancel **Previous** **Next: Tags**

11. Select **Next: Review**.

12. If all the details are correct, select **Create user**.

Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	demo_user
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

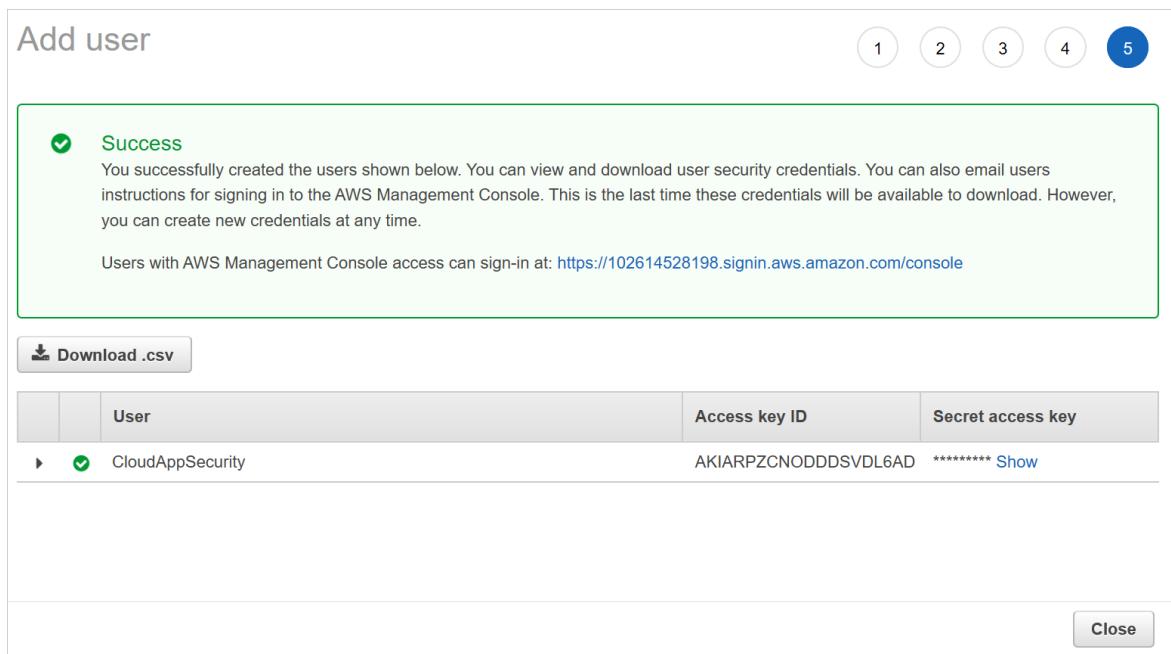
Type	Name
Managed policy	CloudAppSecurityPolicy

Tags

No tags were added.

Create user

13. When you get the success message, select **Download .csv** to save a copy of the new user's credentials. You'll need these later.



Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://102614528198.signin.aws.amazon.com/console>

Download .csv

	User	Access key ID	Secret access key
▶	CloudAppSecurity	AKIARPZCNODDSVLD6AD	***** Show

Close

ⓘ Note

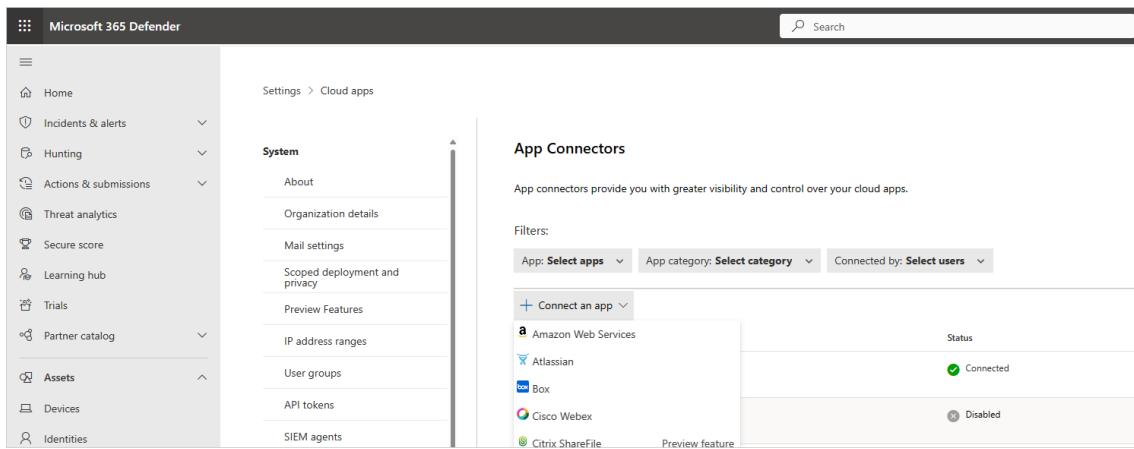
After connecting AWS, you'll receive events for seven days prior to connection. If you just enabled CloudTrail, you'll receive events from the time you enabled CloudTrail.

Step 2: Connect Amazon Web Services auditing to Defender for Cloud Apps

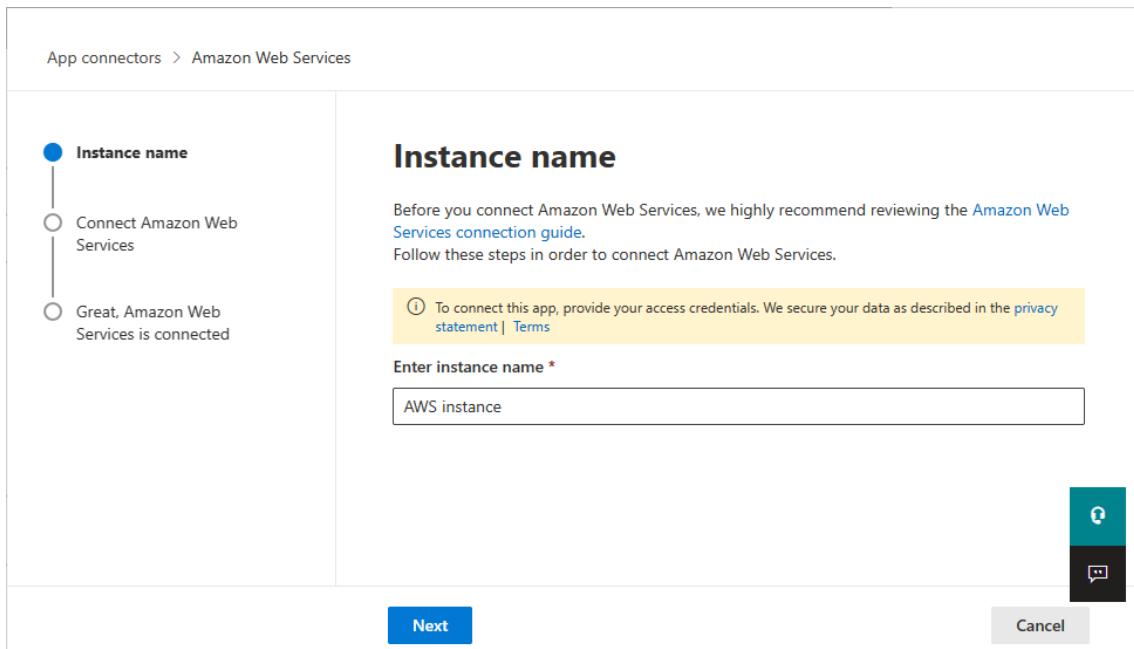
1. In the Microsoft Defender Portal, select **Settings**. Then choose **Cloud Apps**. Under **Connected apps**, select **App Connectors**.
2. In the **App connectors** page, to provide the AWS connector credentials, do one of the following:

For a new connector

- a. Select the **+Connect an app**, followed by **Amazon Web Services**.



b. In the next window, provide a name for the connector, and then select **Next**.



c. On the **Connect Amazon Web Services** page, select **Security auditing**, and then select **Next**.

d. On the **Security auditing** page, paste the **Access key** and **Secret key** from the .csv file into the relevant fields, and select **Next**.

App connectors > Amazon Web Services

Instance name

Connect Amazon Web Services

Security auditing

Great, Amazon Web Services is connected

Security auditing

To connect this app, provide your access credentials. We secure your data as described in the [privacy statement](#) | [Terms](#)

Access key *

Secret key *

Back Next Cancel

For an existing connector

- In the list of connectors, on the row in which the AWS connector appears, select **Edit settings**.

App Connectors

App connectors provide you with greater visibility and control over your cloud apps.

Filters: App: **Amazon Web Services** App category: **Select category** Connected by: **Select users** Advanced filters

+ Connect an app Show details Hide filters Table settings

App	Status	Was connected on	Last activity	Accounts	⋮
aws AWS Dev Cloud computing platform	Connected	Jul 16, 2019 5:18 ...	Feb 18, 2023 11:0...	127	⋮
aws Amazon Web Services - US Cloud computing platform	Connection error	Jul 1, 2019 2:48 PM	—	59	⋮

⋮

Edit settings

Disable App connector

Edit instance name

- On the **Instance name** and **Connect Amazon Web Services** pages, select **Next**.
On the **Security auditing** page, paste the **Access key** and **Secret key** from the .csv file into the relevant fields, and select **Next**.

App connectors > Amazon Web Services

Instance name

Connect Amazon Web Services

Security auditing

Great, Amazon Web Services is connected

Security auditing

To connect this app, provide your access credentials. We secure your data as described in the [privacy statement](#) | [Terms](#)

Access key *

Secret key *

Back

Next

Cancel

3. In the Microsoft Defender Portal, select **Settings**. Then choose **Cloud Apps**. Under **Connected apps**, select **App Connectors**. Make sure the status of the connected App Connector is **Connected**.

Next steps

[Control cloud apps with policies](#)

If you run into any problems, we're here to help. To get assistance or support for your product issue, please [open a support ticket](#).

Security recommendations for AWS resources - a reference guide

Article • 06/27/2023

This article lists the recommendations you might see in Microsoft Defender for Cloud if you've connected an AWS account from the [Environment settings](#) page. The recommendations shown in your environment depend on the resources you're protecting and your customized configuration.

To learn about how to respond to these recommendations, see [Remediate recommendations in Defender for Cloud](#).

Your secure score is based on the number of security recommendations you've completed. To decide which recommendations to resolve first, look at the severity of each one and its potential impact on your secure score.

AWS Compute recommendations

There are 18 AWS recommendations in this category.

Recommendation	Description	Severity
Amazon EC2 instances managed by Systems Manager should have a patch compliance status of COMPLIANT after a patch installation ↗	This control checks whether the compliance status of the Amazon EC2 Systems Manager patch compliance is COMPLIANT or NON_COMPLIANT after the patch installation on the instance. It only checks instances that are managed by AWS Systems Manager Patch Manager. It does not check whether the patch was applied within the 30-day limit prescribed by PCI DSS requirement '6.2'. It also does not validate whether the patches applied were classified as security patches. You should create patching groups with the appropriate baseline settings and ensure in-scope systems are managed by those patch groups in Systems Manager. For more information about patch groups, see the AWS Systems Manager User Guide ↗ .	Medium

Recommendation	Description	Severity
Amazon EFS should be configured to encrypt file data at rest using AWS KMS	<p>This control checks whether Amazon Elastic File System is configured to encrypt the file data using AWS KMS. The check fails in the following cases:</p> <ul style="list-style-type: none"> *"Encrypted" is set to "false" in the DescribeFileSystems response. <p>The "KmsKeyId" key in the DescribeFileSystems response does not match the KmsKeyId parameter for efs-encrypted-check. Note that this control does not use the "KmsKeyId" parameter for efs-encrypted-check. It only checks the value of "Encrypted".</p> <p>For an added layer of security for your sensitive data in Amazon EFS, you should create encrypted file systems.</p> <p>Amazon EFS supports encryption for file systems at-rest. You can enable encryption of data at rest when you create an Amazon EFS file system.</p> <p>To learn more about Amazon EFS encryption, see Data encryption in Amazon EFS in the Amazon Elastic File System User Guide.</p>	Medium
Amazon EFS volumes should be in backup plans	<p>This control checks whether Amazon Elastic File System (Amazon EFS) file systems are added to the backup plans in AWS Backup. The control fails if Amazon EFS file systems are not included in the backup plans.</p> <p>Including EFS file systems in the backup plans helps you to protect your data from deletion and data loss.</p>	Medium
Application Load Balancer deletion protection should be enabled	<p>This control checks whether an Application Load Balancer has deletion protection enabled. The control fails if deletion protection is not configured.</p> <p>Enable deletion protection to protect your Application Load Balancer from deletion.</p>	Medium
Auto Scaling groups associated with a load balancer should use health checks	<p>Auto Scaling groups that are associated with a load balancer are using Elastic Load Balancing health checks.</p> <p>PCI DSS does not require load balancing or highly available configurations. This is recommended by AWS best practices.</p>	Low
AWS accounts should have Azure Arc auto provisioning enabled	<p>For full visibility of the security content from Microsoft Defender for servers, EC2 instances should be connected to Azure Arc. To ensure that all eligible EC2 instances automatically receive Azure Arc, enable auto-provisioning from Defender for Cloud at the AWS account level. Learn more about Azure Arc, and Microsoft Defender for Servers.</p>	High

Recommendation	Description	Severity
CloudFront distributions should have origin failover configured	<p>This control checks whether an Amazon CloudFront distribution is configured with an origin group that has two or more origins. CloudFront origin failover can increase availability. Origin failover automatically redirects traffic to a secondary origin if the primary origin is unavailable or if it returns specific HTTP response status codes.</p>	Medium
CodeBuild GitHub or Bitbucket source repository URLs should use OAuth	<p>This control checks whether the GitHub or Bitbucket source repository URL contains either personal access tokens or a user name and password.</p> <p>Authentication credentials should never be stored or transmitted in clear text or appear in the repository URL. Instead of personal access tokens or user name and password, you should use OAuth to grant authorization for accessing GitHub or Bitbucket repositories.</p> <p>Using personal access tokens or a user name and password could expose your credentials to unintended data exposure and unauthorized access.</p>	High
CodeBuild project environment variables should not contain credentials	<p>This control checks whether the project contains the environment variables <code>AWS_ACCESS_KEY_ID</code> and <code>AWS_SECRET_ACCESS_KEY</code>.</p> <p>Authentication credentials <code>AWS_ACCESS_KEY_ID</code> and <code>AWS_SECRET_ACCESS_KEY</code> should never be stored in clear text, as this could lead to unintended data exposure and unauthorized access.</p>	High
DynamoDB Accelerator (DAX) clusters should be encrypted at rest	<p>This control checks whether a DAX cluster is encrypted at rest. Encrypting data at rest reduces the risk of data stored on disk being accessed by a user not authenticated to AWS. The encryption adds another set of access controls to limit the ability of unauthorized users to access to the data.</p> <p>For example, API permissions are required to decrypt the data before it can be read.</p>	Medium
DynamoDB tables should automatically scale capacity with demand	<p>This control checks whether an Amazon DynamoDB table can scale its read and write capacity as needed. This control passes if the table uses either on-demand capacity mode or provisioned mode with auto scaling configured.</p> <p>Scaling capacity with demand avoids throttling exceptions, which helps to maintain availability of your applications.</p>	Medium
EC2 instances should be connected to Azure Arc	<p>Connect your EC2 instances to Azure Arc in order to have full visibility to Microsoft Defender for Servers security content. Learn more about Azure Arc, and about Microsoft Defender for Servers on hybrid-cloud environment.</p>	High

Recommendation	Description	Severity
EC2 instances should be managed by AWS Systems Manager	<p>Status of the Amazon EC2 Systems Manager patch compliance is 'COMPLIANT' or 'NON_COMPLIANT' after the patch installation on the instance.</p> <p>Only instances that are managed by AWS Systems Manager Patch Manager are checked. Patches that were applied within the 30-day limit prescribed by PCI DSS requirement '6' are not checked.</p>	Medium
Instances managed by Systems Manager should have an association compliance status of COMPLIANT	<p>This control checks whether the status of the AWS Systems Manager association compliance is COMPLIANT or NON_COMPLIANT after the association is run on an instance. The control passes if the association compliance status is COMPLIANT.</p> <p>A State Manager association is a configuration that is assigned to your managed instances. The configuration defines the state that you want to maintain on your instances. For example, an association can specify that antivirus software must be installed and running on your instances, or that certain ports must be closed.</p> <p>After you create one or more State Manager associations, compliance status information is immediately available to you in the console or in response to AWS CLI commands or corresponding Systems Manager API operations. For associations, "Configuration" Compliance shows statuses of Compliant or Non-compliant and the severity level assigned to the association, such as "Critical" or "Medium". To learn more about State Manager association compliance, see About State Manager association compliance in the AWS Systems Manager User Guide.</p> <p>You must configure your in-scope EC2 instances for Systems Manager association. You must also configure the patch baseline for the security rating of the vendor of patches, and set the autoapproval date to meet PCI DSS '3.2.1' requirement '6.2'. For additional guidance on how to Create an association, see Create an association in the AWS Systems Manager User Guide. For additional information on working with patching in Systems Manager, see AWS Systems Manager Patch Manager in the AWS Systems Manager User Guide.</p>	Low

Recommendation	Description	Severity
Lambda functions should have a dead-letter queue configured	<p>This control checks whether a Lambda function is configured with a dead-letter queue. The control fails if the Lambda function is not configured with a dead-letter queue.</p> <p>As an alternative to an on-failure destination, you can configure your function with a dead-letter queue to save discarded events for further processing.</p> <p>A dead-letter queue acts the same as an on-failure destination. It is used when an event fails all processing attempts or expires without being processed.</p> <p>A dead-letter queue allows you to look back at errors or failed requests to your Lambda function to debug or identify unusual behavior.</p> <p>From a security perspective, it is important to understand why your function failed and to ensure that your function does not drop data or compromise data security as a result.</p> <p>For example, if your function cannot communicate to an underlying resource, that could be a symptom of a denial of service (DoS) attack elsewhere in the network.</p>	Medium
Lambda functions should use supported runtimes	<p>This control checks that the Lambda function settings for runtimes match the expected values set for the supported runtimes for each language. This control checks for the following runtimes:</p> <p><code>nodejs14.x, nodejs12.x, nodejs10.x, python3.8, python3.7, python3.6, ruby2.7, ruby2.5, java11, java8, java8.al2, go1.x, dotnetcore3.1, dotnetcore2.1</code></p> <p>Lambda runtimes are built around a combination of operating system, programming language, and software libraries that are subject to maintenance and security updates. When a runtime component is no longer supported for security updates, Lambda deprecates the runtime. Even though you cannot create functions that use the deprecated runtime, the function is still available to process invocation events. Make sure that your Lambda functions are current and do not use out-of-date runtime environments.</p> <p>To learn more about the supported runtimes that this control checks for the supported languages, see AWS Lambda runtimes in the AWS Lambda Developer Guide.</p>	Medium
Management ports of EC2 instances should be protected with just-in-time network access control	<p>Microsoft Defender for Cloud has identified some overly-permissive inbound rules for management ports in your network. Enable just-in-time access control to protect your Instances from internet-based brute-force attacks. Learn more.</p>	High

Recommendation	Description	Severity
Unused EC2 security groups should be removed ↗	Security groups should be attached to Amazon EC2 instances or to an ENI. A healthy finding can indicate there are unused Amazon EC2 security groups.	Low

AWS Container recommendations

There are 3 AWS recommendations in this category.

Recommendation	Description	Severity
EKS clusters should grant the required AWS permissions to Microsoft Defender for Cloud ↗	<p>Microsoft Defender for Containers provides protections for your EKS clusters.</p> <p>To monitor your cluster for security vulnerabilities and threats, Defender for Containers needs permissions for your AWS account. These permissions will be used to enable Kubernetes control plane logging on your cluster and establish a reliable pipeline between your cluster and Defender for Cloud's backend in the cloud.</p> <p>Learn more about Microsoft Defender for Cloud's security features for containerized environments.</p>	High
EKS clusters should have Microsoft Defender's extension for Azure Arc installed ↗	<p>Microsoft Defender's cluster extension provides security capabilities for your EKS clusters. The extension collects data from a cluster and its nodes to identify security vulnerabilities and threats.</p> <p>The extension works with Azure Arc-enabled Kubernetes.</p> <p>Learn more about Microsoft Defender for Cloud's security features for containerized environments.</p>	High
Microsoft Defender for Containers should be enabled on AWS connectors ↗	<p>Microsoft Defender for Containers provides real-time threat protection for containerized environments and generates alerts about suspicious activities.</p> <p>Use this information to harden the security of Kubernetes clusters and remediate security issues.</p> <p>Important: When you've enabled Microsoft Defender for Containers and deployed Azure Arc to your EKS clusters, the protections - and charges - will begin. If you don't deploy Azure Arc on a cluster, Defender for Containers will not protect it and no charges will be incurred for this Microsoft Defender plan for that cluster.</p>	High

Data plane recommendations

All the data plane recommendations listed [here](#) are supported under AWS after [enabling the Azure policy extension](#).

AWS Data recommendations

There are 66 AWS recommendations in this category.

Recommendation	Description	Severity
Amazon Aurora clusters should have backtracking enabled ↗	<p>This control checks whether Amazon Aurora clusters have backtracking enabled.</p> <p>Backups help you to recover more quickly from a security incident. They also strengthen the resilience of your systems.</p> <p>Aurora backtracking reduces the time to recover a database to a point in time. It doesn't require a database restore to do so.</p> <p>For more information about backtracking in Aurora, see Backtracking an Aurora DB cluster ↗ in the Amazon Aurora User Guide.</p>	Medium
Amazon EBS snapshots shouldn't be publicly restorable ↗	Amazon EBS snapshots shouldn't be publicly restorable by everyone unless explicitly allowed, to avoid accidental exposure of data. Additionally, permission to change Amazon EBS configurations should be restricted to authorized AWS accounts only.	High
Amazon ECS task definitions should have secure networking modes and user definitions ↗	<p>This control checks whether an active Amazon ECS task definition that has host networking mode also has privileged or user container definitions.</p> <p>The control fails for task definitions that have host network mode and container definitions where privileged=false or is empty and user=root or is empty.</p> <p>If a task definition has elevated privileges, it is because the customer has specifically opted in to that configuration.</p> <p>This control checks for unexpected privilege escalation when a task definition has host networking enabled but the customer hasn't opted in to elevated privileges.</p>	High

Recommendation	Description	Severity
Amazon Elasticsearch Service domains should encrypt data sent between nodes	This control checks whether Amazon ES domains have node-to-node encryption enabled. HTTPS (TLS) can be used to help prevent potential attackers from eavesdropping on or manipulating network traffic using person-in-the-middle or similar attacks. Only encrypted connections over HTTPS (TLS) should be allowed. Enabling node-to-node encryption for Amazon ES domains ensures that intra-cluster communications are encrypted in transit. There can be a performance penalty associated with this configuration. You should be aware of and test the performance trade-off before enabling this option.	Medium
Amazon Elasticsearch Service domains should have encryption at rest enabled	It's important to enable encryption of the rest of Amazon ES domains to protect sensitive data	Medium
Amazon RDS database should be encrypted using customer managed key	This check identifies RDS databases that are encrypted with default KMS keys and not with customer managed keys. As a leading practice, use customer managed keys to encrypt the data on your RDS databases and maintain control of your keys and data on sensitive workloads.	Medium
Amazon RDS instance should be configured with automatic backup settings	This check identifies RDS instances, which aren't set with the automatic backup setting. If Automatic Backup is set, RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases, which provide for point-in-time recovery. The automatic backup will happen during the specified backup window time and keeps the backups for a limited period of time as defined in the retention period. It's recommended to set automatic backups for your critical RDS servers that will help in the data restoration process.	Medium
Amazon Redshift clusters should have audit logging enabled	This control checks whether an Amazon Redshift cluster has audit logging enabled. Amazon Redshift audit logging provides additional information about connections and user activities in your cluster. This data can be stored and secured in Amazon S3 and can be helpful in security audits and investigations. For more information, see Database audit logging in the <i>Amazon Redshift Cluster Management Guide</i> .	Medium

Recommendation	Description	Severity
Amazon Redshift clusters should have automatic snapshots enabled	This control checks whether Amazon Redshift clusters have automated snapshots enabled. It also checks whether the snapshot retention period is greater than or equal to seven. Backups help you to recover more quickly from a security incident. They strengthen the resilience of your systems. Amazon Redshift takes periodic snapshots by default. This control checks whether automatic snapshots are enabled and retained for at least seven days. For more details on Amazon Redshift automated snapshots, see Automated snapshots in the <i>Amazon Redshift Cluster Management Guide</i> .	Medium
Amazon Redshift clusters should prohibit public access	We recommend Amazon Redshift clusters to avoid public accessibility by evaluating the 'publiclyAccessible' field in the cluster configuration item.	High
Amazon Redshift should have automatic upgrades to major versions enabled	This control checks whether automatic major version upgrades are enabled for the Amazon Redshift cluster. Enabling automatic major version upgrades ensures that the latest major version updates to Amazon Redshift clusters are installed during the maintenance window. These updates might include security patches and bug fixes. Keeping up to date with patch installation is an important step in securing systems.	Medium
Amazon SQS queues should be encrypted at rest	This control checks whether Amazon SQS queues are encrypted at rest. Server-side encryption (SSE) allows you to transmit sensitive data in encrypted queues. To protect the content of messages in queues, SSE uses keys managed in AWS KMS. For more information, see Encryption at rest in the Amazon Simple Queue Service Developer Guide.	Medium
An RDS event notifications subscription should be configured for critical cluster events	This control checks whether an Amazon RDS event subscription exists that has notifications enabled for the following source type, event category key-value pairs. DBCluster: ["maintenance" and "failure"]. RDS event notifications use Amazon SNS to make you aware of changes in the availability or configuration of your RDS resources. These notifications allow for rapid response. For more information about RDS event notifications, see Using Amazon RDS event notification in the Amazon RDS User Guide .	Low

Recommendation	Description	Severity
An RDS event notifications subscription should be configured for critical database instance events	<p>This control checks whether an Amazon RDS event subscription exists with notifications enabled for the following source type, event category key-value pairs. DBInstance: ["maintenance", "configuration change" and "failure"].</p> <p>RDS event notifications use Amazon SNS to make you aware of changes in the availability or configuration of your RDS resources. These notifications allow for rapid response.</p> <p>For more information about RDS event notifications, see Using Amazon RDS event notification in the Amazon RDS User Guide.</p>	Low
An RDS event notifications subscription should be configured for critical database parameter group events	<p>This control checks whether an Amazon RDS event subscription exists with notifications enabled for the following source type, event category key-value pairs. DBParameterGroup: ["configuration", "change"].</p> <p>RDS event notifications use Amazon SNS to make you aware of changes in the availability or configuration of your RDS resources. These notifications allow for rapid response.</p> <p>For more information about RDS event notifications, see Using Amazon RDS event notification in the Amazon RDS User Guide.</p>	Low
An RDS event notifications subscription should be configured for critical database security group events	<p>This control checks whether an Amazon RDS event subscription exists with notifications enabled for the following source type, event category key-value pairs. DBSecurityGroup: ["configuration", "change", "failure"].</p> <p>RDS event notifications use Amazon SNS to make you aware of changes in the availability or configuration of your RDS resources. These notifications allow for a rapid response.</p> <p>For more information about RDS event notifications , see Using Amazon RDS event notification in the Amazon RDS User Guide.</p>	Low
API Gateway REST and WebSocket API logging should be enabled	<p>This control checks whether all stages of an Amazon API Gateway REST or WebSocket API have logging enabled.</p> <p>The control fails if logging isn't enabled for all methods of a stage or if logging Level is neither ERROR nor INFO.</p> <p>API Gateway REST or WebSocket API stages should have relevant logs enabled. API Gateway REST and WebSocket API execution logging provides detailed records of requests made to API Gateway REST and WebSocket API stages.</p> <p>The stages include API integration backend responses, Lambda authorizer responses, and the requestId for AWS integration endpoints.</p>	Medium

Recommendation	Description	Severity
API Gateway REST API cache data should be encrypted at rest	<p>This control checks whether all methods in API Gateway REST API stages that have cache enabled are encrypted. The control fails if any method in an API Gateway REST API stage is configured to cache and the cache isn't encrypted.</p> <p>Encrypting data at rest reduces the risk of data stored on disk being accessed by a user not authenticated to AWS. It adds another set of access controls to limit unauthorized users ability access the data. For example, API permissions are required to decrypt the data before it can be read.</p> <p>API Gateway REST API caches should be encrypted at rest for an added layer of security.</p>	Medium
API Gateway REST API stages should be configured to use SSL certificates for backend authentication	<p>This control checks whether Amazon API Gateway REST API stages have SSL certificates configured.</p> <p>Backend systems use these certificates to authenticate that incoming requests are from API Gateway.</p> <p>API Gateway REST API stages should be configured with SSL certificates to allow backend systems to authenticate that requests originate from API Gateway.</p>	Medium
API Gateway REST API stages should have AWS X-Ray tracing enabled	<p>This control checks whether AWS X-Ray active tracing is enabled for your Amazon API Gateway REST API stages.</p> <p>X-Ray active tracing enables a more rapid response to performance changes in the underlying infrastructure. Changes in performance could result in a lack of availability of the API.</p> <p>X-Ray active tracing provides real-time metrics of user requests that flow through your API Gateway REST API operations and connected services.</p>	Low
API Gateway should be associated with an AWS WAF web ACL	<p>This control checks whether an API Gateway stage uses an AWS WAF web access control list (ACL).</p> <p>This control fails if an AWS WAF web ACL isn't attached to a REST API Gateway stage.</p> <p>AWS WAF is a web application firewall that helps protect web applications and APIs from attacks. It enables you to configure an ACL, which is a set of rules that allow, block, or count web requests based on customizable web security rules and conditions that you define.</p> <p>Ensure that your API Gateway stage is associated with an AWS WAF web ACL to help protect it from malicious attacks.</p>	Medium

Recommendation	Description	Severity
Application and Classic Load Balancers logging should be enabled	<p>This control checks whether the Application Load Balancer and the Classic Load Balancer have logging enabled. The control fails if <code>access_logs.s3.enabled</code> is false.</p> <p>Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and to troubleshoot issues.</p> <p>To learn more, see Access logs for your Classic Load Balancer in User Guide for Classic Load Balancers.</p>	Medium
Attached EBS volumes should be encrypted at-rest	<p>This control checks whether the EBS volumes that are in an attached state are encrypted. To pass this check, EBS volumes must be in use and encrypted. If the EBS volume isn't attached, then it isn't subject to this check.</p> <p>For an added layer of security of your sensitive data in EBS volumes, you should enable EBS encryption at rest. Amazon EBS encryption offers a straightforward encryption solution for your EBS resources that doesn't require you to build, maintain, and secure your own key management infrastructure. It uses AWS KMS customer master keys (CMK) when creating encrypted volumes and snapshots.</p> <p>To learn more about Amazon EBS encryption, see Amazon EBS encryption in the Amazon EC2 User Guide for Linux Instances.</p>	Medium
AWS Database Migration Service replication instances shouldn't be public	<p>To protect your replicated instances from threats. A private replication instance should have a private IP address that you can't access outside of the replication network.</p> <p>A replication instance should have a private IP address when the source and target databases are in the same network, and the network is connected to the replication instance's VPC using a VPN, AWS Direct Connect, or VPC peering.</p> <p>You should also ensure that access to your AWS DMS instance configuration is limited to only authorized users.</p> <p>To do this, restrict users' IAM permissions to modify AWS DMS settings and resources.</p>	High

Recommendation	Description	Severity
Classic Load Balancer listeners should be configured with HTTPS or TLS termination ↗	<p>This control checks whether your Classic Load Balancer listeners are configured with HTTPS or TLS protocol for front-end (client to load balancer) connections. The control is applicable if a Classic Load Balancer has listeners. If your Classic Load Balancer doesn't have a listener configured, then the control doesn't report any findings.</p> <p>The control passes if the Classic Load Balancer listeners are configured with TLS or HTTPS for front-end connections.</p> <p>The control fails if the listener isn't configured with TLS or HTTPS for front-end connections.</p> <p>Before you start to use a load balancer, you must add one or more listeners. A listener is a process that uses the configured protocol and port to check for connection requests. Listeners can support both HTTP and HTTPS/TLS protocols. You should always use an HTTPS or TLS listener, so that the load balancer does the work of encryption and decryption in transit.</p>	Medium
Classic Load Balancers should have connection draining enabled ↗	<p>This control checks whether Classic Load Balancers have connection draining enabled.</p> <p>Enabling connection draining on Classic Load Balancers ensures that the load balancer stops sending requests to instances that are de-registering or unhealthy. It keeps the existing connections open. This is useful for instances in Auto Scaling groups, to ensure that connections aren't severed abruptly.</p>	Medium
CloudFront distributions should have AWS WAF enabled ↗	<p>This control checks whether CloudFront distributions are associated with either AWS WAF or AWS WAFv2 web ACLs. The control fails if the distribution isn't associated with a web ACL.</p> <p>AWS WAF is a web application firewall that helps protect web applications and APIs from attacks. It allows you to configure a set of rules, called a web access control list (web ACL), that allow, block, or count web requests based on customizable web security rules and conditions that you define. Ensure your CloudFront distribution is associated with an AWS WAF web ACL to help protect it from malicious attacks.</p>	Medium

Recommendation	Description	Severity
CloudFront distributions should have logging enabled ↴	<p>This control checks whether server access logging is enabled on CloudFront distributions. The control fails if access logging isn't enabled for a distribution.</p> <p>CloudFront access logs provide detailed information about every user request that CloudFront receives. Each log contains information such as the date and time the request was received, the IP address of the viewer that made the request, the source of the request, and the port number of the request from the viewer. These logs are useful for applications such as security and access audits and forensics investigation. For more guidance on how to analyze access logs, see Querying Amazon CloudFront logs in the Amazon Athena User Guide.</p>	Medium
CloudFront distributions should require encryption in transit ↴	<p>This control checks whether an Amazon CloudFront distribution requires viewers to use HTTPS directly or whether it uses redirection. The control fails if <code>ViewerProtocolPolicy</code> is set to <code>allow-all</code> for <code>defaultCacheBehavior</code> or for <code>cacheBehaviors</code>.</p> <p>HTTPS (TLS) can be used to help prevent potential attackers from using person-in-the-middle or similar attacks to eavesdrop on or manipulate network traffic. Only encrypted connections over HTTPS (TLS) should be allowed. Encrypting data in transit can affect performance. You should test your application with this feature to understand the performance profile and the impact of TLS.</p>	Medium
CloudTrail logs should be encrypted at rest using KMS CMKs ↴	<p>We recommended to configure CloudTrail use SSE-KMS. Configuring CloudTrail to use SSE-KMS provides more confidentiality controls on log data as a given user must have S3 read permission on the corresponding log bucket and must be granted decrypt permission by the CMK policy.</p>	Medium
Connections to Amazon Redshift clusters should be encrypted in transit ↴	<p>This control checks whether connections to Amazon Redshift clusters are required to use encryption in transit. The check fails if the Amazon Redshift cluster parameter <code>require_SSL</code> isn't set to '1'. TLS can be used to help prevent potential attackers from using person-in-the-middle or similar attacks to eavesdrop on or manipulate network traffic. Only encrypted connections over TLS should be allowed. Encrypting data in transit can affect performance. You should test your application with this feature to understand the performance profile and the impact of TLS.</p>	Medium

Recommendation	Description	Severity
Connections to Elasticsearch domains should be encrypted using TLS 1.2	<p>This control checks whether connections to Elasticsearch domains are required to use TLS 1.2. The check fails if the Elasticsearch domain TLSSecurityPolicy isn't Policy-Min-TLS-1-2-2019-07.</p>	Medium
	<p>HTTPS (TLS) can be used to help prevent potential attackers from using person-in-the-middle or similar attacks to eavesdrop on or manipulate network traffic. Only encrypted connections over HTTPS (TLS) should be allowed. Encrypting data in transit can affect performance. You should test your application with this feature to understand the performance profile and the impact of TLS. TLS 1.2 provides several security enhancements over previous versions of TLS.</p>	
DynamoDB tables should have point-in-time recovery enabled	<p>This control checks whether point-in-time recovery (PITR) is enabled for an Amazon DynamoDB table.</p> <p>Backups help you to recover more quickly from a security incident. They also strengthen the resilience of your systems.</p> <p>DynamoDB point-in-time recovery automates backups for DynamoDB tables. It reduces the time to recover from accidental delete or write operations.</p> <p>DynamoDB tables that have PITR enabled can be restored to any point in time in the last 35 days.</p>	Medium
EBS default encryption should be enabled	<p>This control checks whether account-level encryption is enabled by default for Amazon Elastic Block Store(Amazon EBS).</p> <p>The control fails if the account level encryption isn't enabled.</p> <p>When encryption is enabled for your account, Amazon EBS volumes and snapshot copies are encrypted at rest. This adds an more layer of protection for your data.</p> <p>For more information, see Encryption by default in the Amazon EC2 User Guide for Linux Instances.</p> <p>Note that following instance types don't support encryption: R1, C1, and M1.</p>	Medium
Elastic Beanstalk environments should have enhanced health reporting enabled	<p>This control checks whether enhanced health reporting is enabled for your AWS Elastic Beanstalk environments.</p> <p>Elastic Beanstalk enhanced health reporting enables a more rapid response to changes in the health of the underlying infrastructure. These changes could result in a lack of availability of the application.</p> <p>Elastic Beanstalk enhanced health reporting provides a status descriptor to gauge the severity of the identified issues and identify possible causes to investigate. The Elastic Beanstalk health agent, included in supported Amazon Machine Images (AMIs), evaluates logs and metrics of environment EC2 instances.</p>	Low

Recommendation	Description	Severity
Elastic Beanstalk managed platform updates should be enabled ↗	<p>This control checks whether managed platform updates are enabled for the Elastic Beanstalk environment. Enabling managed platform updates ensures that the latest available platform fixes, updates, and features for the environment are installed. Keeping up to date with patch installation is an important step in securing systems.</p>	High
Elastic Load Balancer shouldn't have ACM certificate expired or expiring in 90 days. ↗	<p>This check identifies Elastic Load Balancers (ELB) which are using ACM certificates expired or expiring in 90 days. AWS Certificate Manager (ACM) is the preferred tool to provision, manage, and deploy your server certificates. With ACM you can request a certificate or deploy an existing ACM or external certificate to AWS resources. As a best practice, it's recommended to reimport expiring/expired certificates while preserving the ELB associations of the original certificate.</p>	High
Elasticsearch domain error logging to CloudWatch Logs should be enabled ↗	<p>This control checks whether Elasticsearch domains are configured to send error logs to CloudWatch Logs. You should enable error logs for Elasticsearch domains and send those logs to CloudWatch Logs for retention and response. Domain error logs can assist with security and access audits, and can help to diagnose availability issues.</p>	Medium
Elasticsearch domains should be configured with at least three dedicated master nodes ↗	<p>This control checks whether Elasticsearch domains are configured with at least three dedicated master nodes. This control fails if the domain doesn't use dedicated master nodes. This control passes if Elasticsearch domains have five dedicated master nodes. However, using more than three master nodes might be unnecessary to mitigate the availability risk, and will result in more cost.</p> <p>An Elasticsearch domain requires at least three dedicated master nodes for high availability and fault-tolerance. Dedicated master node resources can be strained during data node blue/green deployments because there are more nodes to manage. Deploying an Elasticsearch domain with at least three dedicated master nodes ensures sufficient master node resource capacity and cluster operations if a node fails.</p>	Medium
Elasticsearch domains should have at least three data nodes ↗	<p>This control checks whether Elasticsearch domains are configured with at least three data nodes and zoneAwarenessEnabled is true. An Elasticsearch domain requires at least three data nodes for high availability and fault-tolerance. Deploying an Elasticsearch domain with at least three data nodes ensures cluster operations if a node fails.</p>	Medium

Recommendation	Description	Severity
Elasticsearch domains should have audit logging enabled ↗	<p>This control checks whether Elasticsearch domains have audit logging enabled. This control fails if an Elasticsearch domain doesn't have audit logging enabled.</p> <p>Audit logs are highly customizable. They allow you to track user activity on your Elasticsearch clusters, including authentication successes and failures, requests to OpenSearch, index changes, and incoming search queries.</p>	Medium
Enhanced monitoring should be configured for RDS DB instances and clusters ↗	<p>This control checks whether enhanced monitoring is enabled for your RDS DB instances.</p> <p>In Amazon RDS, Enhanced Monitoring enables a more rapid response to performance changes in underlying infrastructure. These performance changes could result in a lack of availability of the data. Enhanced Monitoring provides real-time metrics of the operating system that your RDS DB instance runs on. An agent is installed on the instance. The agent can obtain metrics more accurately than is possible from the hypervisor layer.</p> <p>Enhanced Monitoring metrics are useful when you want to see how different processes or threads on a DB instance use the CPU.</p> <p>For more information, see Enhanced Monitoring ↗ in the <i>Amazon RDS User Guide</i>.</p>	Low
Ensure rotation for customer created CMKs is enabled ↗	<p>AWS Key Management Service (KMS) allows customers to rotate the backing key which is key material stored within the KMS which is tied to the key ID of the Customer Created customer master key (CMK).</p> <p>It's the backing key that is used to perform cryptographic operations such as encryption and decryption.</p> <p>Automated key rotation currently retains all prior backing keys so that decryption of encrypted data can take place transparently.</p> <p>It's recommended that CMK key rotation be enabled.</p> <p>Rotating encryption keys helps reduce the potential impact of a compromised key as data encrypted with a new key can't be accessed with a previous key that may have been exposed.</p>	Medium

Recommendation	Description	Severity
Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket	<p>S3 Bucket Access Logging generates a log that contains access records. Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket for each request made to your S3 bucket. An access log record contains details about the request, such as the request type, the resources specified in the request worked, and the time and date the request was processed.</p> <p>It's recommended that bucket access logging be enabled on the CloudTrail S3 bucket.</p> <p>By enabling S3 bucket logging on target S3 buckets, it's possible to capture all events, which may affect objects within target buckets. Configuring logs to be placed in a separate bucket allows access to log information, which can be useful in security and incident response workflows.</p>	Low
Ensure the S3 bucket used to store CloudTrail logs isn't publicly accessible	<p>CloudTrail logs a record of every API call made in your AWS account. These log files are stored in an S3 bucket.</p> <p>It's recommended that the bucket policy, or access control list (ACL), applied to the S3 bucket that CloudTrail logs to prevents public access to the CloudTrail logs.</p> <p>Allowing public access to CloudTrail log content may aid an adversary in identifying weaknesses in the affected account's use or configuration.</p>	High
IAM shouldn't have expired SSL/TLS certificates	<p>This check identifies expired SSL/TLS certificates. To enable HTTPS connections to your website or application in AWS, you need an SSL/TLS server certificate. You can use ACM or IAM to store and deploy server certificates. Removing expired SSL/TLS certificates eliminates the risk that an invalid certificate will be deployed accidentally to a resource such as AWS Elastic Load Balancer (ELB), which can damage the credibility of the application/website behind the ELB. This check generates alerts if there are any expired SSL/TLS certificates stored in AWS IAM. As a best practice, it's recommended to delete expired certificates.</p>	High
Imported ACM certificates should be renewed after a specified time period	<p>This control checks whether ACM certificates in your account are marked for expiration within 30 days. It checks both imported certificates and certificates provided by AWS Certificate Manager. ACM can automatically renew certificates that use DNS validation. For certificates that use email validation, you must respond to a domain validation email.</p> <p>ACM also doesn't automatically renew certificates that you import. You must renew imported certificates manually.</p> <p>For more information about managed renewal for ACM certificates, see Managed renewal for ACM certificates in the AWS Certificate Manager User Guide.</p>	Medium

Recommendation	Description	Severity
Over-provisioned identities in accounts should be investigated to reduce the Permission Creep Index (PCI) ↗	Over-provisioned identities in accounts should be investigated to reduce the Permission Creep Index (PCI) and to safeguard your infrastructure. Reduce the PCI by removing the unused high risk permission assignments. High PCI reflects risk associated with the identities with permissions that exceed their normal or required usage	Medium
RDS automatic minor version upgrades should be enabled ↗	This control checks whether automatic minor version upgrades are enabled for the RDS database instance. Enabling automatic minor version upgrades ensures that the latest minor version updates to the relational database management system (RDBMS) are installed. These upgrades might include security patches and bug fixes. Keeping up to date with patch installation is an important step in securing systems.	High
RDS cluster snapshots and database snapshots should be encrypted at rest ↗	This control checks whether RDS DB snapshots are encrypted. This control is intended for RDS DB instances. However, it can also generate findings for snapshots of Aurora DB instances, Neptune DB instances, and Amazon DocumentDB clusters. If these findings aren't useful, then you can suppress them. Encrypting data at rest reduces the risk that an unauthenticated user gets access to data that is stored on disk. Data in RDS snapshots should be encrypted at rest for an added layer of security.	Medium
RDS clusters should have deletion protection enabled ↗	This control checks whether RDS clusters have deletion protection enabled. This control is intended for RDS DB instances. However, it can also generate findings for Aurora DB instances, Neptune DB instances, and Amazon DocumentDB clusters. If these findings aren't useful, then you can suppress them. Enabling cluster deletion protection is an more layer of protection against accidental database deletion or deletion by an unauthorized entity. When deletion protection is enabled, an RDS cluster can't be deleted. Before a deletion request can succeed, deletion protection must be disabled.	Low
RDS DB clusters should be configured for multiple Availability Zones ↗	RDS DB clusters should be configured for multiple the data that is stored. Deployment to multiple Availability Zones allows for automate Availability Zones to ensure availability of ed failover in the event of an Availability Zone availability issue and during regular RDS maintenance events.	Medium

Recommendation	Description	Severity
RDS DB clusters should be configured to copy tags to snapshots ↴	<p>Identification and inventory of your IT assets is a crucial aspect of governance and security.</p> <p>You need to have visibility of all your RDS DB clusters so that you can assess their security posture and act on potential areas of weakness.</p> <p>Snapshots should be tagged in the same way as their parent RDS database clusters.</p> <p>Enabling this setting ensures that snapshots inherit the tags of their parent database clusters.</p>	Low
RDS DB instances should be configured to copy tags to snapshots ↴	<p>This control checks whether RDS DB instances are configured to copy all tags to snapshots when the snapshots are created.</p> <p>Identification and inventory of your IT assets is a crucial aspect of governance and security.</p> <p>You need to have visibility of all your RDS DB instances so that you can assess their security posture and take action on potential areas of weakness.</p> <p>Snapshots should be tagged in the same way as their parent RDS database instances. Enabling this setting ensures that snapshots inherit the tags of their parent database instances.</p>	Low
RDS DB instances should be configured with multiple Availability Zones ↴	<p>This control checks whether high availability is enabled for your RDS DB instances.</p> <p>RDS DB instances should be configured for multiple Availability Zones (AZs). This ensures the availability of the data stored.</p> <p>Multi-AZ deployments allow for automated failover if there's an issue with Availability Zone availability and during regular RDS maintenance.</p>	Medium
RDS DB instances should have deletion protection enabled ↴	<p>This control checks whether your RDS DB instances that use one of the listed database engines have deletion protection enabled.</p> <p>Enabling instance deletion protection is an more layer of protection against accidental database deletion or deletion by an unauthorized entity.</p> <p>While deletion protection is enabled, an RDS DB instance can't be deleted. Before a deletion request can succeed, deletion protection must be disabled.</p>	Low

Recommendation	Description	Severity
RDS DB instances should have encryption at rest enabled	<p>This control checks whether storage encryption is enabled for your Amazon RDS DB instances.</p> <p>This control is intended for RDS DB instances. However, it can also generate findings for Aurora DB instances, Neptune DB instances, and Amazon DocumentDB clusters. If these findings aren't useful, then you can suppress them.</p> <p>For an added layer of security for your sensitive data in RDS DB instances, you should configure your RDS DB instances to be encrypted at rest. To encrypt your RDS DB instances and snapshots at rest, enable the encryption option for your RDS DB instances. Data that is encrypted at rest includes the underlying storage for DB instances, its automated backups, read replicas, and snapshots.</p> <p>RDS encrypted DB instances use the open standard AES-256 encryption algorithm to encrypt your data on the server that hosts your RDS DB instances. After your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently with a minimal impact on performance. You don't need to modify your database client applications to use encryption.</p> <p>Amazon RDS encryption is currently available for all database engines and storage types. Amazon RDS encryption is available for most DB instance classes. To learn about DB instance classes that don't support Amazon RDS encryption, see Encrypting Amazon RDS resources in the <i>Amazon RDS User Guide</i>.</p>	Medium
RDS DB Instances should prohibit public access	<p>We recommend that you also ensure that access to your RDS instance's configuration is limited to authorized users only, by restricting users' IAM permissions to modify RDS instances' settings and resources.</p>	High
RDS snapshots should prohibit public access	<p>We recommend only allowing authorized principals to access the snapshot and change Amazon RDS configuration.</p>	High

Recommendation	Description	Severity
Remove unused Secrets Manager secrets ↗	<p>This control checks whether your secrets have been accessed within a specified number of days. The default value is 90 days. If a secret wasn't accessed within the defined number of days, this control fails.</p> <p>Deleting unused secrets is as important as rotating secrets. Unused secrets can be abused by their former users, who no longer need access to these secrets. Also, as more users get access to a secret, someone might have mishandled and leaked it to an unauthorized entity, which increases the risk of abuse.</p> <p>Deleting unused secrets helps revoke secret access from users who no longer need it. It also helps to reduce the cost of using Secrets Manager. Therefore, it's essential to routinely delete unused secrets.</p>	Medium
S3 buckets should have cross-region replication enabled ↗	<p>Enabling S3 cross-region replication ensures that multiple versions of the data are available in different distinct Regions. This allows you to protect your S3 bucket against DDoS attacks and data corruption events.</p>	Low
S3 buckets should have server-side encryption enabled ↗	<p>Enable server-side encryption to protect data in your S3 buckets. Encrypting the data can prevent access to sensitive data in the event of a data breach.</p>	Medium
Secrets Manager secrets configured with automatic rotation should rotate successfully ↗	<p>This control checks whether an AWS Secrets Manager secret rotated successfully based on the rotation schedule. The control fails if RotationOccurringAsScheduled is false. The control doesn't evaluate secrets that don't have rotation configured.</p> <p>Secrets Manager helps you improve the security posture of your organization. Secrets include database credentials, passwords, and third-party API keys. You can use Secrets Manager to store secrets centrally, encrypt secrets automatically, control access to secrets, and rotate secrets safely and automatically.</p> <p>Secrets Manager can rotate secrets. You can use rotation to replace long-term secrets with short-term ones. Rotating your secrets limits how long an unauthorized user can use a compromised secret. For this reason, you should rotate your secrets frequently.</p> <p>In addition to configuring secrets to rotate automatically, you should ensure that those secrets rotate successfully based on the rotation schedule.</p> <p>To learn more about rotation, see Rotating your AWS Secrets Manager secrets ↗ in the AWS Secrets Manager User Guide.</p>	Medium

Recommendation	Description	Severity
Secrets Manager secrets should be rotated within a specified number of days	<p>This control checks whether your secrets have been rotated at least once within 90 days. Rotating secrets can help you to reduce the risk of an unauthorized use of your secrets in your AWS account. Examples include database credentials, passwords, third-party API keys, and even arbitrary text. If you don't change your secrets for a long period of time, the secrets are more likely to be compromised.</p> <p>As more users get access to a secret, it can become more likely that someone mishandled and leaked it to an unauthorized entity. Secrets can be leaked through logs and cache data. They can be shared for debugging purposes and not changed or revoked once the debugging completes. For all these reasons, secrets should be rotated frequently.</p> <p>You can configure your secrets for automatic rotation in AWS Secrets Manager. With automatic rotation, you can replace long-term secrets with short-term ones, significantly reducing the risk of compromise.</p> <p>Security Hub recommends that you enable rotation for your Secrets Manager secrets. To learn more about rotation, see Rotating your AWS Secrets Manager secrets in the AWS Secrets Manager User Guide.</p>	Medium
SNS topics should be encrypted at rest using AWS KMS	<p>This control checks whether an SNS topic is encrypted at rest using AWS KMS.</p> <p>Encrypting data at rest reduces the risk of data stored on disk being accessed by a user not authenticated to AWS. It also adds another set of access controls to limit the ability of unauthorized users to access the data.</p> <p>For example, API permissions are required to decrypt the data before it can be read. SNS topics should be encrypted at-rest for an added layer of security. For more information, see Encryption at rest in the Amazon Simple Notification Service Developer Guide.</p>	Medium
VPC flow logging should be enabled in all VPCs	<p>VPC Flow Logs provide visibility into network traffic that passes through the VPC and can be used to detect anomalous traffic or insight during security events.</p>	Medium

AWS IdentityAndAccess recommendations

There are 46 AWS recommendations in this category.

Recommendation	Description	Severity

Recommendation	Description	Severity
Amazon Elasticsearch Service domains should be in a VPC	<p>VPC cannot contain domains with a public endpoint. Note: this does not evaluate the VPC subnet routing configuration to determine public reachability.</p>	High
Amazon S3 permissions granted to other AWS accounts in bucket policies should be restricted	<p>Implementing least privilege access is fundamental to reducing security risk and the impact of errors or malicious intent. If an S3 bucket policy allows access from external accounts, it could result in data exfiltration by an insider threat or an attacker. The 'blacklistedactionpatterns' parameter allows for successful evaluation of the rule for S3 buckets. The parameter grants access to external accounts for action patterns that are not included in the 'blacklistedactionpatterns' list.</p>	High
Avoid the use of the "root" account	<p>The "root" account has unrestricted access to all resources in the AWS account. It is highly recommend that the use of this account be avoided.</p> <p>The "root" account is the most privileged AWS account. Minimizing the use of this account and adopting the principle of least privilege for access management will reduce the risk of accidental changes and unintended disclosure of highly privileged credentials.</p>	High
AWS KMS keys should not be unintentionally deleted	<p>This control checks whether KMS keys are scheduled for deletion. The control fails if a KMS key is scheduled for deletion.</p> <p>KMS keys cannot be recovered once deleted. Data encrypted under a KMS key is also permanently unrecoverable if the KMS key is deleted. If meaningful data has been encrypted under a KMS key scheduled for deletion, consider decrypting the data or re-encrypting the data under a new KMS key unless you are intentionally performing a cryptographic erasure.</p> <p>When a KMS key is scheduled for deletion, a mandatory waiting period is enforced to allow time to reverse the deletion, if it was scheduled in error. The default waiting period is 30 days, but it can be reduced to as short as 7 days when the KMS key is scheduled for deletion. During the waiting period, the scheduled deletion can be canceled and the KMS key will not be deleted.</p> <p>For additional information regarding deleting KMS keys, see Deleting KMS keys in the AWS Key Management Service Developer Guide.</p>	High

Recommendation	Description	Severity
AWS WAF Classic global web ACL logging should be enabled ↴	<p>This control checks whether logging is enabled for an AWS WAF global Web ACL. This control fails if logging is not enabled for the web ACL.</p> <p>Logging is an important part of maintaining the reliability, availability, and performance of AWS WAF globally. It is a business and compliance requirement in many organizations, and allows you to troubleshoot application behavior. It also provides detailed information about the traffic that is analyzed by the web ACL that is attached to AWS WAF.</p>	Medium
CloudFront distributions should have a default root object configured ↴	<p>This control checks whether an Amazon CloudFront distribution is configured to return a specific object that is the default root object. The control fails if the CloudFront distribution does not have a default root object configured.</p> <p>A user might sometimes request the distributions root URL instead of an object in the distribution. When this happens, specifying a default root object can help you to avoid exposing the contents of your web distribution.</p>	High
CloudFront distributions should have origin access identity enabled ↴	<p>This control checks whether an Amazon CloudFront distribution with Amazon S3 Origin type has Origin Access Identity (OAI) configured. The control fails if OAI is not configured.</p> <p>CloudFront OAI prevents users from accessing S3 bucket content directly. When users access an S3 bucket directly, they effectively bypass the CloudFront distribution and any permissions that are applied to the underlying S3 bucket content.</p>	Medium
CloudTrail log file validation should be enabled ↴	<p>To ensure additional integrity checking of CloudTrail logs, we recommend enabling file validation on all CloudTrails.</p>	Low
CloudTrail should be enabled ↴	<p>AWS CloudTrail is a web service that records AWS API calls for your account and delivers log files to you. Not all services enable logging by default for all APIs and events.</p> <p>You should implement any additional audit trails other than CloudTrail and review the documentation for each service in CloudTrail Supported Services and Integrations.</p>	High

Recommendation	Description	Severity
CloudTrail trails should be integrated with CloudWatch Logs ↴	<p>In addition to capturing CloudTrail logs within a specified S3 bucket for long term analysis, real-time analysis can be performed by configuring CloudTrail to send logs to CloudWatch Logs.</p> <p>For a trail that is enabled in all regions in an account, CloudTrail sends log files from all those regions to a CloudWatch Logs log group. We recommended that CloudTrail logs will be sent to CloudWatch Logs to ensure AWS account activity is being captured, monitored, and appropriately alarmed on.</p> <p>Sending CloudTrail logs to CloudWatch Logs facilitates real-time and historic activity logging based on user, API, resource, and IP address, and provides opportunity to establish alarms and notifications for anomalous or sensitivity account activity.</p>	Low
Database logging should be enabled ↴	<p>This control checks whether the following logs of Amazon RDS are enabled and sent to CloudWatch Logs:</p> <ul style="list-style-type: none"> - Oracle: (Alert, Audit, Trace, Listener) - PostgreSQL: (Postgresql, Upgrade) - MySQL: (Audit, Error, General, SlowQuery) - MariaDB: (Audit, Error, General, SlowQuery) - SQL Server: (Error, Agent) - Aurora: (Audit, Error, General, SlowQuery) - Aurora-MySQL: (Audit, Error, General, SlowQuery) - Aurora-PostgreSQL: (Postgresql, Upgrade). <p>RDS databases should have relevant logs enabled. Database logging provides detailed records of requests made to RDS. Database logs can assist with security and access audits and can help to diagnose availability issues.</p>	Medium
Disable direct internet access for Amazon SageMaker notebook instances ↴	<p>Direct internet access should be disabled for an SageMaker notebook instance.</p> <p>This checks whether the 'DirectInternetAccess' field is disabled for the notebook instance.</p> <p>Your instance should be configured with a VPC and the default setting should be Disable - Access the internet through a VPC.</p> <p>In order to enable internet access to train or host models from a notebook, make sure that your VPC has a NAT gateway and your security group allows outbound connections. Ensure access to your SageMaker configuration is limited to only authorized users, and restrict users' IAM permissions to modify SageMaker settings and resources.</p>	High

Recommendation	Description	Severity
Do not setup access keys during initial user setup for all IAM users that have a console password ↴	<p>AWS console defaults the checkbox for creating access keys to enabled. This results in many access keys being generated unnecessarily.</p> <p>In addition to unnecessary credentials, it also generates unnecessary management work in auditing and rotating these keys.</p> <p>Requiring that additional steps be taken by the user after their profile has been created will give a stronger indication of intent that access keys are [a] necessary for their work and [b] once the access key is established on an account that the keys may be in use somewhere in the organization</p>	Medium
Ensure a support role has been created to manage incidents with AWS Support ↴	<p>AWS provides a support center that can be used for incident notification and response, as well as technical support and customer services.</p> <p>Create an IAM Role to allow authorized users to manage incidents with AWS Support.</p> <p>By implementing least privilege for access control, an IAM Role will require an appropriate IAM Policy to allow Support Center Access in order to manage Incidents with AWS Support.</p>	Low
Ensure access keys are rotated every 90 days or less ↴	<p>Access keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS.</p> <p>AWS users need their own access keys to make programmatic calls to AWS from the AWS Command Line Interface (AWS CLI), Tools for Windows PowerShell, the AWS SDKs, or direct HTTP calls using the APIs for individual AWS services.</p> <p>It is recommended that all access keys be regularly rotated. Rotating access keys will reduce the window of opportunity for an access key that is associated with a compromised or terminated account to be used.</p> <p>Access keys should be rotated to ensure that data cannot be accessed with an old key which might have been lost, cracked, or stolen.</p>	Medium

Recommendation	Description	Severity
Ensure AWS Config is enabled in all regions	<p>AWS Config is a web service that performs configuration management of supported AWS resources within your account and delivers log files to you.</p> <p>The recorded information includes the configuration item (AWS resource), relationships between configuration items (AWS resources), any configuration changes between resources.</p> <p>It is recommended to enable AWS Config be enabled in all regions.</p> <p>The AWS configuration item history captured by AWS Config enables security analysis, resource change tracking, and compliance auditing.</p>	Medium
Ensure CloudTrail is enabled in all regions	<p>AWS CloudTrail is a web service that records AWS API calls for your account and delivers log files to you.</p> <p>The recorded information includes the identity of the API caller, the time of the API call, the source IP address of the API caller, the request parameters, and the response elements returned by the AWS service. CloudTrail provides a history of AWS API calls for an account, including API calls made via the Management Console, SDKs, command line tools, and higher-level AWS services (such as CloudFormation).</p> <p>The AWS API call history produced by CloudTrail enables security analysis, resource change tracking, and compliance auditing.</p> <p>Additionally,</p> <ul style="list-style-type: none"> * ensuring that a multi-regions trail exists will ensure that unexpected activity occurring in otherwise unused regions is detected * ensuring that a multi-regions trail exists will ensure that "Global Service Logging" is enabled for a trail by default to capture recording of events generated on AWS global services * for a multi-regions trail, ensuring that management events configured for all type of Read/Writes ensures recording of management operations that are performed on all resources in an AWS account. 	High
Ensure credentials unused for 90 days or greater are disabled	<p>AWS IAM users can access AWS resources using different types of credentials, such as passwords or access keys.</p> <p>It is recommended that all credentials that have been unused in 90 or greater days be removed or deactivated.</p> <p>Disabling or removing unnecessary credentials will reduce the window of opportunity for credentials associated with a compromised or abandoned account to be used.</p>	Medium

Recommendation	Description	Severity
Ensure IAM password policy expires passwords within 90 days or less	<p>IAM password policies can require passwords to be rotated or expired after a given number of days.</p> <p>It is recommended that the password policy expire passwords after 90 days or less.</p> <p>Reducing the password lifetime increases account resiliency against brute force login attempts. Additionally, requiring regular password changes help in the following scenarios:</p> <ul style="list-style-type: none"> * Passwords can be stolen or compromised sometimes without your knowledge. This can happen via a system compromise, software vulnerability, or internal threat. * Certain corporate and government web filters or proxy servers have the ability to intercept and record traffic even if it's encrypted. * Many people use the same password for many systems such as work, email, and personal. * Compromised end user workstations might have a keystroke logger. 	Low
Ensure IAM password policy prevents password reuse	<p>IAM password policies can prevent the reuse of a given password by the same user.</p> <p>It is recommended that the password policy prevent the reuse of passwords.</p> <p>Preventing password reuse increases account resiliency against brute force login attempts.</p>	Low
Ensure IAM password policy requires at least one lowercase letter	<p>Password policies are, in part, used to enforce password complexity requirements. IAM password policies can be used to ensure password are comprised of different character sets.</p> <p>It is recommended that the password policy require at least one lowercase letter.</p> <p>Setting a password complexity policy increases account resiliency against brute force login attempts</p>	Medium
Ensure IAM password policy requires at least one number	<p>Password policies are, in part, used to enforce password complexity requirements. IAM password policies can be used to ensure password are comprised of different character sets.</p> <p>It is recommended that the password policy require at least one number.</p> <p>Setting a password complexity policy increases account resiliency against brute force login attempts.</p>	Medium

Recommendation	Description	Severity
Ensure IAM password policy requires at least one symbol ↴	<p>Password policies are, in part, used to enforce password complexity requirements.</p> <p>IAM password policies can be used to ensure password are comprised of different character sets.</p> <p>It is recommended that the password policy require at least one symbol.</p> <p>Setting a password complexity policy increases account resiliency against brute force login attempts.</p>	Medium
Ensure IAM password policy requires at least one uppercase letter ↴	<p>Password policies are, in part, used to enforce password complexity requirements. IAM password policies can be used to ensure password are comprised of different character sets.</p> <p>It is recommended that the password policy require at least one uppercase letter.</p> <p>Setting a password complexity policy increases account resiliency against brute force login attempts.</p>	Medium
Ensure IAM password policy requires minimum length of 14 or greater ↴	<p>Password policies are, in part, used to enforce password complexity requirements. IAM password policies can be used to ensure password are at least a given length.</p> <p>It is recommended that the password policy require a minimum password length '14'.</p> <p>Setting a password complexity policy increases account resiliency against brute force login attempts.</p>	Medium
Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password ↴	<p>Multi-Factor Authentication (MFA) adds an extra layer of protection on top of a user name and password.</p> <p>With MFA enabled, when a user signs in to an AWS website, they will be prompted for their user name and password as well as for an authentication code from their AWS MFA device.</p> <p>It is recommended that MFA be enabled for all accounts that have a console password.</p> <p>Enabling MFA provides increased security for console access as it requires the authenticating principal to possess a device that emits a time-sensitive key and have knowledge of a credential.</p>	Medium
GuardDuty should be enabled ↴	<p>To provide additional protection against intrusions, GuardDuty should be enabled on your AWS account and region.</p> <p>Note: GuardDuty might not be a complete solution for every environment</p>	Medium

Recommendation	Description	Severity
Hardware MFA should be enabled for the "root" account ↗	<p>The root account is the most privileged user in an account. MFA adds an extra layer of protection on top of a user name and password. With MFA enabled, when a user signs in to an AWS website, they're prompted for their user name and password and for an authentication code from their AWS MFA device.</p> <p>For Level 2, it is recommended that you protect the root account with a hardware MFA. A hardware MFA has a smaller attack surface than a virtual MFA. For example, a hardware MFA doesn't suffer the attack surface introduced by the mobile smartphone that a virtual MFA resides on.</p> <p>Using hardware MFA for many, many accounts might create a logistical device management issue. If this occurs, consider implementing this Level 2 recommendation selectively to the highest security accounts. You can then apply the Level 1 recommendation to the remaining accounts.</p>	Low
IAM authentication should be configured for RDS clusters ↗	<p>This control checks whether an RDS DB cluster has IAM database authentication enabled.</p> <p>IAM database authentication allows for password-free authentication to database instances. The authentication uses an authentication token. Network traffic to and from the database is encrypted using SSL. For more information, see IAM database authentication in the Amazon Aurora User Guide.</p>	Medium
IAM authentication should be configured for RDS instances ↗	<p>This control checks whether an RDS DB instance has IAM database authentication enabled.</p> <p>IAM database authentication allows authentication to database instances with an authentication token instead of a password. Network traffic to and from the database is encrypted using SSL. For more information, see IAM database authentication in the Amazon Aurora User Guide.</p>	Medium

Recommendation	Description	Severity
IAM customer managed policies should not allow decryption actions on all KMS keys	<p>Checks whether the default version of IAM customer managed policies allow principals to use the AWS KMS decryption actions on all resources. This control uses Zelkova, an automated reasoning engine, to validate and warn you about policies that may grant broad access to your secrets across AWS accounts. This control fails if the "kms:Decrypt" or "kms:ReEncryptFrom" actions are allowed on all KMS keys. The control evaluates both attached and unattached customer managed policies. It does not check inline policies or AWS managed policies.</p> <p>With AWS KMS, you control who can use your KMS keys and gain access to your encrypted data. IAM policies define which actions an identity (user, group, or role) can perform on which resources. Following security best practices, AWS recommends that you allow least privilege. In other words, you should grant to identities only the "kms:Decrypt" or "kms:ReEncryptFrom" permissions and only for the keys that are required to perform a task. Otherwise, the user might use keys that are not appropriate for your data.</p> <p>Instead of granting permissions for all keys, determine the minimum set of keys that users need to access encrypted data. Then design policies that allow users to use only those keys. For example, do not allow "kms:Decrypt" permission on all KMS keys. Instead, allow "kms:Decrypt" only on keys in a particular Region for your account. By adopting the principle of least privilege, you can reduce the risk of unintended disclosure of your data.</p>	Medium

Recommendation	Description	Severity
IAM customer managed policies that you create should not allow wildcard actions for services ↗	<p>This control checks whether the IAM identity-based policies that you create have Allow statements that use the * wildcard to grant permissions for all actions on any service. The control fails if any policy statement includes 'Effect': 'Allow' with 'Action': 'Service:'. <i>For example, the following statement in a policy results in a failed finding.</i></p>	Low

Recommendation	Description	Severity
IAM policies should be attached only to groups or roles	<p>By default, IAM users, groups, and roles have no access to AWS resources. IAM policies are the means by which privileges are granted to users, groups, or roles.</p> <p>It is recommended that IAM policies be applied directly to groups and roles but not users.</p> <p>Assigning privileges at the group or role level reduces the complexity of access management as the number of users grow. Reducing access management complexity may in-turn reduce opportunity for a principal to inadvertently receive or retain excessive privileges.</p>	Low
IAM policies that allow full ":" administrative privileges should not be created	<p>IAM policies are the means by which privileges are granted to users, groups, or roles.</p> <p>It is recommended and considered a standard security advice to grant least privilege—that is, granting only the permissions required to perform a task.</p> <p>Determine what users need to do and then craft policies for them that let the users perform only those tasks, instead of allowing full administrative privileges.</p> <p>It's more secure to start with a minimum set of permissions and grant additional permissions as necessary, rather than starting with permissions that are too lenient and then trying to tighten them later.</p> <p>Providing full administrative privileges instead of restricting to the minimum set of permissions that the user is required to do exposes the resources to potentially unwanted actions.</p> <p>IAM policies that have a statement with "Effect": "Allow" with "Action": "" over "Resource": "" should be removed.</p>	High

Recommendation	Description	Severity
IAM principals should not have IAM inline policies that allow decryption actions on all KMS keys ↗	<p>Checks whether the inline policies that are embedded in your IAM identities (role, user, or group) allow the AWS KMS decryption actions on all KMS keys. This control uses Zelkova ↗, an automated reasoning engine, to validate and warn you about policies that may grant broad access to your secrets across AWS accounts.</p> <p>This control fails if "kms:Decrypt" or "kms:ReEncryptFrom" actions are allowed on all KMS keys in an inline policy.</p> <p>With AWS KMS, you control who can use your KMS keys and gain access to your encrypted data. IAM policies define which actions an identity (user, group, or role) can perform on which resources. Following security best practices, AWS recommends that you allow least privilege. In other words, you should grant to identities only the permissions they need and only for keys that are required to perform a task. Otherwise, the user might use keys that are not appropriate for your data.</p> <p>Instead of granting permission for all keys, determine the minimum set of keys that users need to access encrypted data. Then design policies that allow the users to use only those keys. For example, do not allow "kms:Decrypt" permission on all KMS keys. Instead, allow them only on keys in a particular Region for your account. By adopting the principle of least privilege, you can reduce the risk of unintended disclosure of your data.</p>	Medium
Lambda functions should restrict public access ↗	<p>Lambda function resource-based policy should restrict public access. This recommendation does not check access by internal principals.</p> <p>Ensure access to the function is restricted to authorized principals only by using least privilege resource-based policies.</p>	High
MFA should be enabled for all IAM users ↗	<p>All IAM users should have multi-factor authentication (MFA) enabled.</p>	Medium
MFA should be enabled for the "root" account ↗	<p>The root account is the most privileged user in an account. MFA adds an extra layer of protection on top of a user name and password. With MFA enabled, when a user signs in to an AWS website, they're prompted for their user name and password and for an authentication code from their AWS MFA device.</p> <p>When you use virtual MFA for root accounts, it is recommended that the device used is not a personal device. Instead, use a dedicated mobile device (tablet or phone) that you manage to keep charged and secured independent of any individual personal devices.</p> <p>This lessens the risks of losing access to the MFA due to device loss, device trade-in, or if the individual owning the device is no longer employed at the company.</p>	Low

Recommendation	Description	Severity
Password policies for IAM users should have strong configurations ↗	<p>Checks whether the account password policy for IAM users uses the following minimum configurations.</p> <ul style="list-style-type: none"> * <code>RequireUppercaseCharacters</code>- Require at least one uppercase character in password. (Default = true) * <code>RequireLowercaseCharacters</code>- Require at least one lowercase character in password. (Default = true) * <code>RequireNumbers</code>- Require at least one number in password. (Default = true) * <code>MinimumPasswordLength</code>- Password minimum length. (Default = 7 or longer) * <code>PasswordReusePrevention</code>- Number of passwords before allowing reuse. (Default = 4) * <code>MaxPasswordAge</code>- Number of days before password expiration. (Default = 90) 	Medium
Root account access key shouldn't exist ↗	<p>The root account is the most privileged user in an AWS account. AWS Access Keys provide programmatic access to a given AWS account.</p> <p>It is recommended that all access keys associated with the root account be removed.</p> <p>Removing access keys associated with the root account limits vectors by which the account can be compromised.</p> <p>Additionally, removing the root access keys encourages the creation and use of role based accounts that are least privileged.</p>	High
S3 Block Public Access setting should be enabled ↗	<p>Enabling Block Public Access setting for your S3 bucket can help prevent sensitive data leaks and protect your bucket from malicious actions.</p>	Medium
S3 Block Public Access setting should be enabled at the bucket level ↗	<p>This control checks whether S3 buckets have bucket-level public access blocks applied. This control fails if any of the following settings are set to false:</p> <ul style="list-style-type: none"> * <code>ignorePublicAcls</code> * <code>blockPublicPolicy</code> * <code>blockPublicAcls</code> * <code>restrictPublicBuckets</code> <p>Block Public Access at the S3 bucket level provides controls to ensure that objects never have public access. Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, or both.</p> <p>Unless you intend to have your S3 buckets publicly accessible, you should configure the bucket level Amazon S3 Block Public Access feature.</p>	High

Recommendation	Description	Severity
S3 buckets public read access should be removed ↗	Removing public read access to your S3 bucket can help protect your data and prevent a data breach.	High
S3 buckets public write access should be removed ↗	Allowing public write access to your S3 bucket can leave you vulnerable to malicious actions such as storing data at your expense, encrypting your files for ransom, or using your bucket to operate malware.	High
Secrets Manager secrets should have automatic rotation enabled ↗	<p>This control checks whether a secret stored in AWS Secrets Manager is configured with automatic rotation. Secrets Manager helps you improve the security posture of your organization. Secrets include database credentials, passwords, and third-party API keys. You can use Secrets Manager to store secrets centrally, encrypt secrets automatically, control access to secrets, and rotate secrets safely and automatically. Secrets Manager can rotate secrets. You can use rotation to replace long-term secrets with short-term ones. Rotating your secrets limits how long an unauthorized user can use a compromised secret. For this reason, you should rotate your secrets frequently. To learn more about rotation, see Rotating your AWS Secrets Manager secrets ↗ in the AWS Secrets Manager User Guide.</p>	Medium
Stopped EC2 instances should be removed after a specified time period ↗	This control checks whether any EC2 instances have been stopped for more than the allowed number of days. An EC2 instance fails this check if it is stopped for longer than the maximum allowed time period, which by default is 30 days. A failed finding indicates that an EC2 instance has not run for a significant period of time. This creates a security risk because the EC2 instance is not being actively maintained (analyzed, patched, updated). If it is later launched, the lack of proper maintenance could result in unexpected issues in your AWS environment. To safely maintain an EC2 instance over time in a nonrunning state, start it periodically for maintenance and then stop it after maintenance. Ideally this is an automated process.	Medium

AWS Networking recommendations

There are **36** AWS recommendations in this category.

Recommendation	Description	Severity
----------------	-------------	----------

Recommendation	Description	Severity
Amazon EC2 should be configured to use VPC endpoints	<p>This control checks whether a service endpoint for Amazon EC2 is created for each VPC. The control fails if a VPC does not have a VPC endpoint created for the Amazon EC2 service.</p> <p>To improve the security posture of your VPC, you can configure Amazon EC2 to use an interface VPC endpoint. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to access Amazon EC2 API operations privately. It restricts all network traffic between your VPC and Amazon EC2 to the Amazon network. Because endpoints are supported within the same Region only, you cannot create an endpoint between a VPC and a service in a different Region. This prevents unintended Amazon EC2 API calls to other Regions.</p> <p>To learn more about creating VPC endpoints for Amazon EC2, see Amazon EC2 and interface VPC endpoints in the Amazon EC2 User Guide for Linux Instances.</p>	Medium
Amazon ECS services should not have public IP addresses assigned to them automatically	<p>A public IP address is an IP address that is reachable from the internet.</p> <p>If you launch your Amazon ECS instances with a public IP address, then your Amazon ECS instances are reachable from the internet.</p> <p>Amazon ECS services should not be publicly accessible, as this may allow unintended access to your container application servers.</p>	High
Amazon EMR cluster master nodes should not have public IP addresses	<p>This control checks whether master nodes on Amazon EMR clusters have public IP addresses.</p> <p>The control fails if the master node has public IP addresses that are associated with any of its instances. Public IP addresses are designated in the PublicIp field of the NetworkInterfaces configuration for the instance.</p> <p>This control only checks Amazon EMR clusters that are in a RUNNING or WAITING state.</p>	High
Amazon Redshift clusters should use enhanced VPC routing	<p>This control checks whether an Amazon Redshift cluster has EnhancedVpcRouting enabled.</p> <p>Enhanced VPC routing forces all COPY and UNLOAD traffic between the cluster and data repositories to go through your VPC. You can then use VPC features such as security groups and network access control lists to secure network traffic. You can also use VPC Flow Logs to monitor network traffic.</p>	High
Application Load Balancer should be configured to redirect all HTTP requests to HTTPS	<p>To enforce encryption in transit, you should use redirect actions with Application Load Balancers to redirect client HTTP requests to an HTTPS request on port 443.</p>	Medium

Recommendation	Description	Severity
Application load balancers should be configured to drop HTTP headers	<p>This control evaluates AWS Application Load Balancers (ALB) to ensure they are configured to drop invalid HTTP headers. The control fails if the value of <code>routing.http.drop_invalid_header_fields.enabled</code> is set to false. By default, ALBs are not configured to drop invalid HTTP header values. Removing these header values prevents HTTP desync attacks.</p>	Medium
Configure Lambda functions to a VPC	<p>This control checks whether a Lambda function is in a VPC. It does not evaluate the VPC subnet routing configuration to determine public reachability.</p> <p>Note that if Lambda@Edge is found in the account, then this control generates failed findings. To prevent these findings, you can disable this control.</p>	Low
EC2 instances should not have a public IP address	<p>This control checks whether EC2 instances have a public IP address. The control fails if the "publicIp" field is present in the EC2 instance configuration item. This control applies to IPv4 addresses only.</p> <p>A public IPv4 address is an IP address that is reachable from the internet. If you launch your instance with a public IP address, then your EC2 instance is reachable from the internet. A private IPv4 address is an IP address that is not reachable from the internet. You can use private IPv4 addresses for communication between EC2 instances in the same VPC or in your connected private network.</p> <p>IPv6 addresses are globally unique, and therefore are reachable from the internet. However, by default all subnets have the IPv6 addressing attribute set to false. For more information about IPv6, see IP addressing in your VPC in the Amazon VPC User Guide.</p> <p>If you have a legitimate use case to maintain EC2 instances with public IP addresses, then you can suppress the findings from this control. For more information about front-end architecture options, see the AWS Architecture Blog or the This Is My Architecture series.</p>	High
EC2 instances should not use multiple ENIs	<p>This control checks whether an EC2 instance uses multiple Elastic Network Interfaces (ENIs) or Elastic Fabric Adapters (EFAs). This control passes if a single network adapter is used. The control includes an optional parameter list to identify the allowed ENIs. Multiple ENIs can cause dual-homed instances, meaning instances that have multiple subnets. This can add network security complexity and introduce unintended network paths and access.</p>	Low

Recommendation	Description	Severity
EC2 instances should use IMDSv2 ↴	<p>This control checks whether your EC2 instance metadata version is configured with Instance Metadata Service Version 2 (IMDSv2). The control passes if "HttpTokens" is set to "required" for IMDSv2. The control fails if "HttpTokens" is set to "optional". You use instance metadata to configure or manage the running instance. The IMDS provides access to temporary, frequently rotated credentials. These credentials remove the need to hard code or distribute sensitive credentials to instances manually or programmatically. The IMDS is attached locally to every EC2 instance. It runs on a special 'link local' IP address of 169.254.169.254. This IP address is only accessible by software that runs on the instance.</p> <p>Version 2 of the IMDS adds new protections for the following types of vulnerabilities. These vulnerabilities could be used to try to access the IMDS.</p> <ul style="list-style-type: none"> * Open website application firewalls * Open reverse proxies * Server-side request forgery (SSRF) vulnerabilities * Open Layer 3 firewalls and network address translation (NAT) <p>Security Hub recommends that you configure your EC2 instances with IMDSv2.</p>	High
EC2 subnets should not automatically assign public IP addresses ↴	<p>This control checks whether the assignment of public IPs in Amazon Virtual Private Cloud (Amazon VPC) subnets have "MapPublicIpOnLaunch" set to "FALSE". The control passes if the flag is set to "FALSE".</p> <p>All subnets have an attribute that determines whether a network interface created in the subnet automatically receives a public IPv4 address. Instances that are launched into subnets that have this attribute enabled have a public IP address assigned to their primary network interface.</p>	Medium
Ensure a log metric filter and alarm exist for AWS Config configuration changes ↴	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for detecting changes to CloudTrail's configurations.</p> <p>Monitoring changes to AWS Config configuration will help ensure sustained visibility of configuration items within the AWS account.</p>	Low

Recommendation	Description	Severity
Ensure a log metric filter and alarm exist for AWS Management Console authentication failures ↴	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for failed console authentication attempts.</p> <p>Monitoring failed console logins may decrease lead time to detect an attempt to brute force a credential, which may provide an indicator, such as source IP, that can be used in other event correlation.</p>	Low
Ensure a log metric filter and alarm exist for changes to Network Access Control Lists (NACL) ↴	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms. NACLs are used as a stateless packet filter to control ingress and egress traffic for subnets within a VPC.</p> <p>It is recommended that a metric filter and alarm be established for changes made to NACLs.</p> <p>Monitoring changes to NACLs will help ensure that AWS resources and services are not unintentionally exposed.</p>	Low
Ensure a log metric filter and alarm exist for changes to network gateways ↴	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms. Network gateways are required to send/receive traffic to a destination outside of a VPC.</p> <p>It is recommended that a metric filter and alarm be established for changes to network gateways.</p> <p>Monitoring changes to network gateways will help ensure that all ingress/egress traffic traverses the VPC border via a controlled path.</p>	Low
Ensure a log metric filter and alarm exist for CloudTrail configuration changes ↴	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for detecting changes to CloudTrail's configurations.</p> <p>Monitoring changes to CloudTrail's configuration will help ensure sustained visibility to activities performed in the AWS account.</p>	Low
Ensure a log metric filter and alarm exist for disabling or scheduled deletion of customer created CMKs ↴	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for customer created CMKs which have changed state to disabled or scheduled deletion.</p> <p>Data encrypted with disabled or deleted keys will no longer be accessible.</p>	Low

Recommendation	Description	Severity
Ensure a log metric filter and alarm exist for IAM policy changes	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for changes made to Identity and Access Management (IAM) policies.</p> <p>Monitoring changes to IAM policies will help ensure authentication and authorization controls remain intact.</p>	Low
Ensure a log metric filter and alarm exist for Management Console sign-in without MFA	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for console logins that are not protected by multi-factor authentication (MFA).</p> <p>Monitoring for single-factor console logins will increase visibility into accounts that are not protected by MFA.</p>	Low
Ensure a log metric filter and alarm exist for route table changes	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms. Routing tables are used to route network traffic between subnets and to network gateways.</p> <p>It is recommended that a metric filter and alarm be established for changes to route tables.</p> <p>Monitoring changes to route tables will help ensure that all VPC traffic flows through an expected path.</p>	Low
Ensure a log metric filter and alarm exist for S3 bucket policy changes	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for changes to S3 bucket policies.</p> <p>Monitoring changes to S3 bucket policies may reduce time to detect and correct permissive policies on sensitive S3 buckets.</p>	Low
Ensure a log metric filter and alarm exist for security group changes	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms. Security Groups are a stateful packet filter that controls ingress and egress traffic within a VPC.</p> <p>It is recommended that a metric filter and alarm be established for changes to Security Groups.</p> <p>Monitoring changes to security group will help ensure that resources and services are not unintentionally exposed.</p>	Low

Recommendation	Description	Severity
Ensure a log metric filter and alarm exist for unauthorized API calls 	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for unauthorized API calls.</p> <p>Monitoring unauthorized API calls will help reveal application errors and may reduce time to detect malicious activity.</p>	Low
Ensure a log metric filter and alarm exist for usage of 'root' account 	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is recommended that a metric filter and alarm be established for root login attempts.</p> <p>Monitoring for root account logins will provide visibility into the use of a fully privileged account and an opportunity to reduce the use of it.</p>	Low
Ensure a log metric filter and alarm exist for VPC changes 	<p>Real-time monitoring of API calls can be achieved by directing CloudTrail Logs to CloudWatch Logs and establishing corresponding metric filters and alarms.</p> <p>It is possible to have more than 1 VPC within an account, in addition it is also possible to create a peer connection between 2 VPCs enabling network traffic to route between VPCs. It is recommended that a metric filter and alarm be established for changes made to VPCs.</p> <p>Monitoring changes to IAM policies will help ensure authentication and authorization controls remain intact.</p>	Low
Ensure no security groups allow ingress from 0.0.0.0/0 to port 3389 	<p>Security groups provide stateful filtering of ingress/egress network traffic to AWS resources. It is recommended that no security group allows unrestricted ingress access to port 3389.</p> <p>Removing unfettered connectivity to remote console services, such as RDP, reduces a server's exposure to risk.</p>	High
Management ports of EC2 instances should be protected with just-in-time network access control 	<p>Microsoft Defender for Cloud has identified some overly-permissive inbound rules for management ports in your network.</p> <p>Enable just-in-time access control to protect your Instances from internet-based brute-force attacks. Learn more.</p>	High

Recommendation	Description	Severity
RDS databases and clusters should not use a database engine default port	<p>This control checks whether the RDS cluster or instance uses a port other than the default port of the database engine. If you use a known port to deploy an RDS cluster or instance, an attacker can guess information about the cluster or instance. The attacker can use this information in conjunction with other information to connect to an RDS cluster or instance or gain additional information about your application.</p> <p>When you change the port, you must also update the existing connection strings that were used to connect to the old port. You should also check the security group of the DB instance to ensure that it includes an ingress rule that allows connectivity on the new port.</p>	Low
RDS instances should be deployed in a VPC	<p>VPCs provide a number of network controls to secure access to RDS resources. These controls include VPC Endpoints, network ACLs, and security groups. To take advantage of these controls, we recommend that you move EC2-Classic RDS instances to EC2-VPC.</p>	Low
S3 buckets should require requests to use Secure Socket Layer	<p>We recommend to require requests to use Secure Socket Layer (SSL) on all Amazon S3 bucket. S3 buckets should have policies that require all requests ('Action: S3:*') to only accept transmission of data over HTTPS in the S3 resource policy, indicated by the condition key 'aws:SecureTransport'.</p>	Medium
Security groups should not allow ingress from 0.0.0.0/0 to port 22	<p>To reduce the server's exposure, it is recommended not to allow unrestricted ingress access to port '22'.</p>	High

Recommendation	Description	Severity
Security groups should not allow unrestricted access to ports with high risk	<p>This control checks whether unrestricted incoming traffic for the security groups is accessible to the specified ports that have the highest risk. This control passes when none of the rules in a security group allow ingress traffic from 0.0.0.0/0 for those ports. Unrestricted access (0.0.0.0/0) increases opportunities for malicious activity, such as hacking, denial-of-service attacks, and loss of data.</p> <p>Security groups provide stateful filtering of ingress and egress network traffic to AWS resources. No security group should allow unrestricted ingress access to the following ports:</p> <ul style="list-style-type: none"> - 3389 (RDP) - 20, 21 (FTP) - 22 (SSH) - 23 (Telnet) - 110 (POP3) - 143 (IMAP) - 3306 (mySQL) - 8080 (proxy) - 1433, 1434 (MSSQL) - 9200 or 9300 (Elasticsearch) - 5601 (Kibana) - 25 (SMTP) - 445 (CIFS) - 135 (RPC) - 4333 (ahsp) - 5432 (postgresql) - 5500 (fcp-addr-srvr1) 	Medium

Recommendation	Description	Severity
Security groups should only allow unrestricted incoming traffic for authorized ports ↴	<p>This control checks whether the security groups that are in use allow unrestricted incoming traffic. Optionally the rule checks whether the port numbers are listed in the "authorizedTcpPorts" parameter.</p> <ul style="list-style-type: none"> - If the security group rule port number allows unrestricted incoming traffic, but the port number is specified in "authorizedTcpPorts", then the control passes. The default value for "authorizedTcpPorts" is 80, 443. - If the security group rule port number allows unrestricted incoming traffic, but the port number is not specified in authorizedTcpPorts input parameter, then the control fails. - If the parameter is not used, then the control fails for any security group that has an unrestricted inbound rule. <p>Security groups provide stateful filtering of ingress and egress network traffic to AWS. Security group rules should follow the principle of least privileged access. Unrestricted access (IP address with a /0 suffix) increases the opportunity for malicious activity such as hacking, denial-of-service attacks, and loss of data.</p> <p>Unless a port is specifically allowed, the port should deny unrestricted access.</p>	High
Unused EC2 EIPs should be removed ↴	<p>Elastic IP addresses that are allocated to a VPC should be attached to Amazon EC2 instances or in-use elastic network interfaces (ENIs).</p>	Low
Unused network access control lists should be removed ↴	<p>This control checks whether there are any unused network access control lists (ACLs).</p> <p>The control checks the item configuration of the resource "AWS::EC2::NetworkAcl" and determines the relationships of the network ACL.</p> <p>If the only relationship is the VPC of the network ACL, then the control fails.</p> <p>If other relationships are listed, then the control passes.</p>	Low
VPC's default security group should restricts all traffic ↴	<p>Security group should restrict all traffic to reduce resource exposure.</p>	Low

Next steps

For related information, see the following:

- [Connect your AWS accounts to Microsoft Defender for Cloud](#)

- What are security policies, initiatives, and recommendations?
- Review your security recommendations

How Defender for Cloud Apps helps protect your Amazon Web Services (AWS) environment

Article • 01/22/2024

Amazon Web Services is an IaaS provider that enables your organization to host and manage their entire workloads in the cloud. Along with the benefits of leveraging infrastructure in the cloud, your organization's most critical assets may be exposed to threats. Exposed assets include storage instances with potentially sensitive information, compute resources that operate some of your most critical applications, ports, and virtual private networks that enable access to your organization.

Connecting AWS to Defender for Cloud Apps helps you secure your assets and detect potential threats by monitoring administrative and sign-in activities, notifying on possible brute force attacks, malicious use of a privileged user account, unusual deletions of VMs, and publicly exposed storage buckets.

Main threats

- Abuse of cloud resources
- Compromised accounts and insider threats
- Data leakage
- Resource misconfiguration and insufficient access control

How Defender for Cloud Apps helps to protect your environment

- Detect cloud threats, compromised accounts, and malicious insiders
- Limit exposure of shared data and enforce collaboration policies
- Use the audit trail of activities for forensic investigations

Control AWS with built-in policies and policy templates

You can use the following built-in policy templates to detect and notify you about potential threats:

[Expand table](#)

Type	Name
Activity policy template	Admin console sign-in failures CloudTrail configuration changes EC2 instance configuration changes IAM policy changes Logon from a risky IP address Network access control list (ACL) changes Network gateway changes S3 configuration changes Security group configuration changes Virtual private network changes
Built-in anomaly detection policy	Activity from anonymous IP addresses Activity from infrequent country Activity from suspicious IP addresses Impossible travel Activity performed by terminated user (requires Microsoft Entra ID as IdP) Multiple failed login attempts Unusual administrative activities Unusual multiple storage deletion activities (preview) Multiple delete VM activities Unusual multiple VM creation activities (preview) Unusual region for cloud resource (preview)
File policy template	S3 bucket is publicly accessible

For more information about creating policies, see [Create a policy](#).

Automate governance controls

In addition to monitoring for potential threats, you can apply and automate the following AWS governance actions to remediate detected threats:

[Expand table](#)

Type	Action
User governance	- Notify user on alert (via Microsoft Entra ID) - Require user to sign in again (via Microsoft Entra ID) - Suspend user (via Microsoft Entra ID)
Data governance	- Make an S3 bucket private - Remove a collaborator for an S3 bucket

For more information about remediating threats from apps, see [Governing connected apps](#).

Protect AWS in real time

Review our best practices for [blocking and protecting the download of sensitive data to unmanaged or risky devices](#).

Connect Amazon Web Services to Microsoft Defender for Cloud Apps

This section provides instructions for connecting your existing Amazon Web Services (AWS) account to Microsoft Defender for Cloud Apps using the connector APIs. For information about how Defender for Cloud Apps protects AWS, see [Protect AWS](#).

You can connect AWS **Security auditing** to Defender for Cloud Apps connections to gain visibility into and control over AWS app use.

Step 1: Configure Amazon Web Services auditing

1. In your [Amazon Web Services console](#), under **Security, Identity & Compliance**, select **IAM**.

AWS services

Find a service by name (for example, EC2, S3, Elastic Beanstalk).

All services

- Compute**
 - EC2
 - EC2 Container Service
 - Lightsail
 - Elastic Beanstalk
 - Lambda
 - Batch
- Storage**
 - S3
 - EFS
 - Glacier
 - Storage Gateway
- Database**
 - RDS
 - DynamoDB
 - ElastiCache
 - Redshift
- Networking & Content Delivery**
 - VPC
 - CloudFront
 - Direct Connect
 - Route 53
- Migration**
 - Application Discovery Service
 - DMS
 - Server Migration
 - Snowball
- Compute**
 - CodeCommit
 - CodeBuild
 - CodeDeploy
 - CodePipeline
- Developer Tools**
 - CloudWatch
 - CloudFormation
 - CloudTrail
 - Config
 - OpsWorks
 - Service Catalog
 - Trusted Advisor
- Management Tools**
 - Inspector
 - Certificate Manager
 - Directory Service
 - WAF & Shield
 - Compliance Reports
- Internet of Things**
 - AWS IoT
- Game Development**
 - GameLift
- Mobile Services**
 - Mobile Hub
 - Cognito
 - Device Farm
 - Mobile Analytics
 - Pinpoint
- Application Services**
 - Step Functions
 - SWF
 - API Gateway
 - Elastic Transcoder
- Analytics**
 - Athena
 - EMR
 - CloudSearch
 - Elasticsearch Service
 - Kinesis
 - Data Pipeline
 - QuickSight
- Business Productivity**
 - WorkDocs
 - WorkMail
 - Amazon Chime
- Desktop & App Streaming**
 - WorkSpaces
 - AppStream 2.0
- Artificial Intelligence**
 - Lex
 - Polly
 - Rekognition
 - Machine Learning
- Security, Identity & Compliance**
 - IAM

2. Select **Users** and then select **Add user**.

Search IAM

Dashboard

Groups

Users

Roles

Policies

Identity providers

Account settings

Credential report

Encryption keys

Add user

Delete users

Find user

username or access key

Showing 0 results

User name	Groups	Password	Last sign-in	Access keys	Creation time
No results					

3. In the **Details** step, provide a new user name for Defender for Cloud Apps. Make sure that under **Access type** you select **Programmatic access** and select **Next Permissions**.

Featured next steps



Manage your costs

Get real-time billing alerts based on your cost and usage budgets. [Start now](#)



Get best practices

Use AWS Trusted Advisor for security, performance, cost and availability best practices. [Start now](#)

What's new?

Announcing AWS Batch

Now generally available, AWS Batch enables developers, scientists, and engineers to process large-scale batch jobs with ease. [Learn more](#)

Announcing Amazon Lightsail

See how this new service allows you to launch and manage your VPS with AWS for a low, predictable price. [Learn more](#)

[See all](#)

AWS Marketplace

Discover, procure, and deploy popular [software products](#) that run on AWS.

Have feedback?

[Submit feedback](#) to tell us about your experience with the AWS Management Console.

Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name* CloudAppSecurityAWS

[+ Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type* **Programmatic access**

Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access

Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required

[Cancel](#)

[Next: Permissions](#)

4. Select **Attach existing policies directly**, and then **Create policy**.

Add user

1 2 3 4 5

Set permissions

[Add user to group](#)

[Copy permissions from existing user](#)

[Attach existing policies directly](#)

[Create policy](#)

[Filter policies](#)

[Search](#)

Showing 668 results

Policy name ▾

Type

Used as

5. Select the **JSON** tab:

Create policy

1 2
Editor Review

A policy defines the AWS permissions that can be assigned to a user, group, role, or resource. You can construct a policy using the visual editor or create a policy document using the JSON editor.

[Visual editor](#)

[JSON](#)

[Import managed policy](#)

Use the visual editor to create and edit a policy by choosing services, actions, resources, and request conditions to add permissions to your policy. You can add multiple permission blocks to define complex permissions or to grant access to more than one service. [Learn more](#)

[Expand all](#) | [Collapse all](#)

[Select a service](#)

[Clone](#) | [Remove](#)

Service Choose a service

Actions Choose a service before defining actions

Resources Choose actions before applying resources

Request conditions Choose actions before specifying conditions

[+ Add additional permissions](#)

[Cancel](#) [Review policy](#)

6. Paste the following script into the provided area:

JSON

```
{  
  "Version" : "2012-10-17",  
  "Statement" : [  
    {"Action" : [  
      "cloudtrail:DescribeTrails",  
      "cloudtrail:LookupEvents",  
      "cloudtrail:GetTrailStatus",  
      "cloudwatch:Describe*",  
      "cloudwatch:Get*",  
      "cloudwatch>List*",  
      "iam>List*",  
      "iam:Get*",  
      "s3>ListAllMyBuckets",  
      "s3:PutBucketAcl",  
      "s3:GetBucketAcl",  
      "s3:GetBucketLocation"  
    ],  
    "Effect" : "Allow",  
    "Resource" : "*"  
  }  
]  
}
```

7. Select Next: Tags

Create policy

1 2 3

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor JSON Import managed policy

```
1  {  
2   "Version" : "2012-10-17",  
3   "Statement" : [  
4     {"Action" : [  
5       "cloudtrail:DescribeTrails",  
6       "cloudtrail:LookupEvents",  
7       "cloudtrail:GetTrailStatus",  
8       "cloudwatch:Describe*",  
9       "cloudwatch:Get*",  
10      "cloudwatch>List*",  
11      "iam>List*",  
12      "iam:Get*",  
13      "s3>ListAllMyBuckets",  
14      "s3:PutBucketAcl",  
15      "s3:GetBucketAcl",  
16      "s3:GetBucketLocation"  
17    ],  
18    "Effect" : "Allow",  
19    "Resource" : "*"  
20  }  
21]  
22}
```

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

Character count: 329 of 6,144. Cancel Next: Tags

8. Select Next: Review.

Create policy

1 2 3

Add tags (Optional)
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add tag
You can add up to 50 more tags

Cancel **Previous** **Next: Review**

9. Provide a **Name** and select **Create policy**.

Create policy

1 2

Editor **Review**

Review policy
Before you create this policy, provide the required information and review this policy.

Name* Maximum 128 characters. Use alphanumeric and '+, -, @, _' characters.

Description Maximum 1000 characters. Use alphanumeric and '+, -, @, _' characters.

Summary

Service	Access level	Resource	Request condition
CloudTrail	Full: List Limited: Read	All resources	None
CloudWatch	Full: List, Read	All resources	None
IAM	Full: List Limited: Read	All resources	None

*** Required** **Cancel** **Previous** **Create policy**

10. Back in the **Add user** screen, refresh the list if necessary, and select the user you created, and select **Next: Tags**.

Add user

1 2 3 4 5

Set permissions

 Add user to group  Copy permissions from existing user  Attach existing policies directly

Create policy 

Filter policies Showing 3 results

	Policy name	Type	Used as
<input checked="" type="checkbox"/>	CloudAppSecurityPolicy	Customer managed	Permissions policy (2)

Set permissions boundary

Cancel **Previous** **Next: Tags**

11. Select **Next: Review**.

12. If all the details are correct, select **Create user**.

Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	demo_user
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

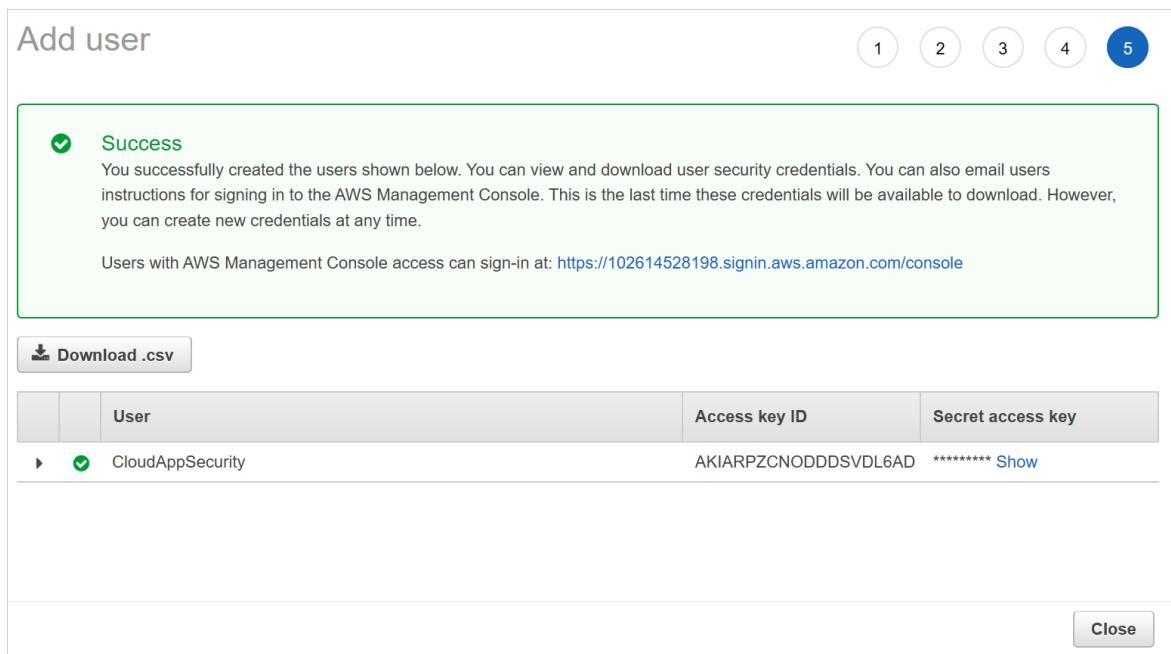
Type	Name
Managed policy	CloudAppSecurityPolicy

Tags

No tags were added.

Create user

13. When you get the success message, select **Download .csv** to save a copy of the new user's credentials. You'll need these later.



Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://102614528198.signin.aws.amazon.com/console>

	User	Access key ID	Secret access key
▶	CloudAppSecurity	AKIARPZCNODDSVLD6AD	***** Show

Close

ⓘ Note

After connecting AWS, you'll receive events for seven days prior to connection. If you just enabled CloudTrail, you'll receive events from the time you enabled CloudTrail.

Step 2: Connect Amazon Web Services auditing to Defender for Cloud Apps

1. In the Microsoft Defender Portal, select **Settings**. Then choose **Cloud Apps**. Under **Connected apps**, select **App Connectors**.
2. In the **App connectors** page, to provide the AWS connector credentials, do one of the following:

For a new connector

- a. Select the **+Connect an app**, followed by **Amazon Web Services**.

Microsoft 365 Defender

Settings > Cloud apps

System

App Connectors

App connectors provide you with greater visibility and control over your cloud apps.

Filters:

App: Select apps App category: Select category Connected by: Select users

		Status
	Amazon Web Services	Connected
	Atlassian	
	Box	
	Cisco Webex	
	Citrix ShareFile	Disabled

b. In the next window, provide a name for the connector, and then select **Next**.

App connectors > Amazon Web Services

Instance name

Before you connect Amazon Web Services, we highly recommend reviewing the [Amazon Web Services connection guide](#). Follow these steps in order to connect Amazon Web Services.

To connect this app, provide your access credentials. We secure your data as described in the [privacy statement](#) | [Terms](#)

Enter instance name *

AWS instance

Next Cancel

c. On the **Connect Amazon Web Services** page, select **Security auditing**, and then select **Next**.

d. On the **Security auditing** page, paste the **Access key** and **Secret key** from the .csv file into the relevant fields, and select **Next**.

App connectors > Amazon Web Services

Security auditing

To connect this app, provide your access credentials. We secure your data as described in the [privacy statement](#) | [Terms](#)

Access key *

Secret key *

Back Next Cancel

For an existing connector

- In the list of connectors, on the row in which the AWS connector appears, select **Edit settings**.

App Connectors

App connectors provide you with greater visibility and control over your cloud apps.

Filters: App: **Amazon Web Services** App category: **Select category** Connected by: **Select users** Advanced filters

+ Connect an app Show details Hide filters Table settings

App	Status	Was connected on	Last activity	Accounts	⋮
aws AWS Dev Cloud computing platform	Connected	Jul 16, 2019 5:18 ...	Feb 18, 2023 11:0...	127	⋮
aws Amazon Web Services - US Cloud computing platform	Connection error	Jul 1, 2019 2:48 PM	—	59	⋮

⋮

Edit settings
Disable App connector
Edit instance name

- On the **Instance name** and **Connect Amazon Web Services** pages, select **Next**.
On the **Security auditing** page, paste the **Access key** and **Secret key** from the .csv file into the relevant fields, and select **Next**.

App connectors > Amazon Web Services

Instance name

Connect Amazon Web Services

Security auditing

Great, Amazon Web Services is connected

Security auditing

To connect this app, provide your access credentials. We secure your data as described in the [privacy statement](#) | [Terms](#)

Access key *

Secret key *

Back

Next

Cancel

3. In the Microsoft Defender Portal, select **Settings**. Then choose **Cloud Apps**. Under **Connected apps**, select **App Connectors**. Make sure the status of the connected App Connector is **Connected**.

Next steps

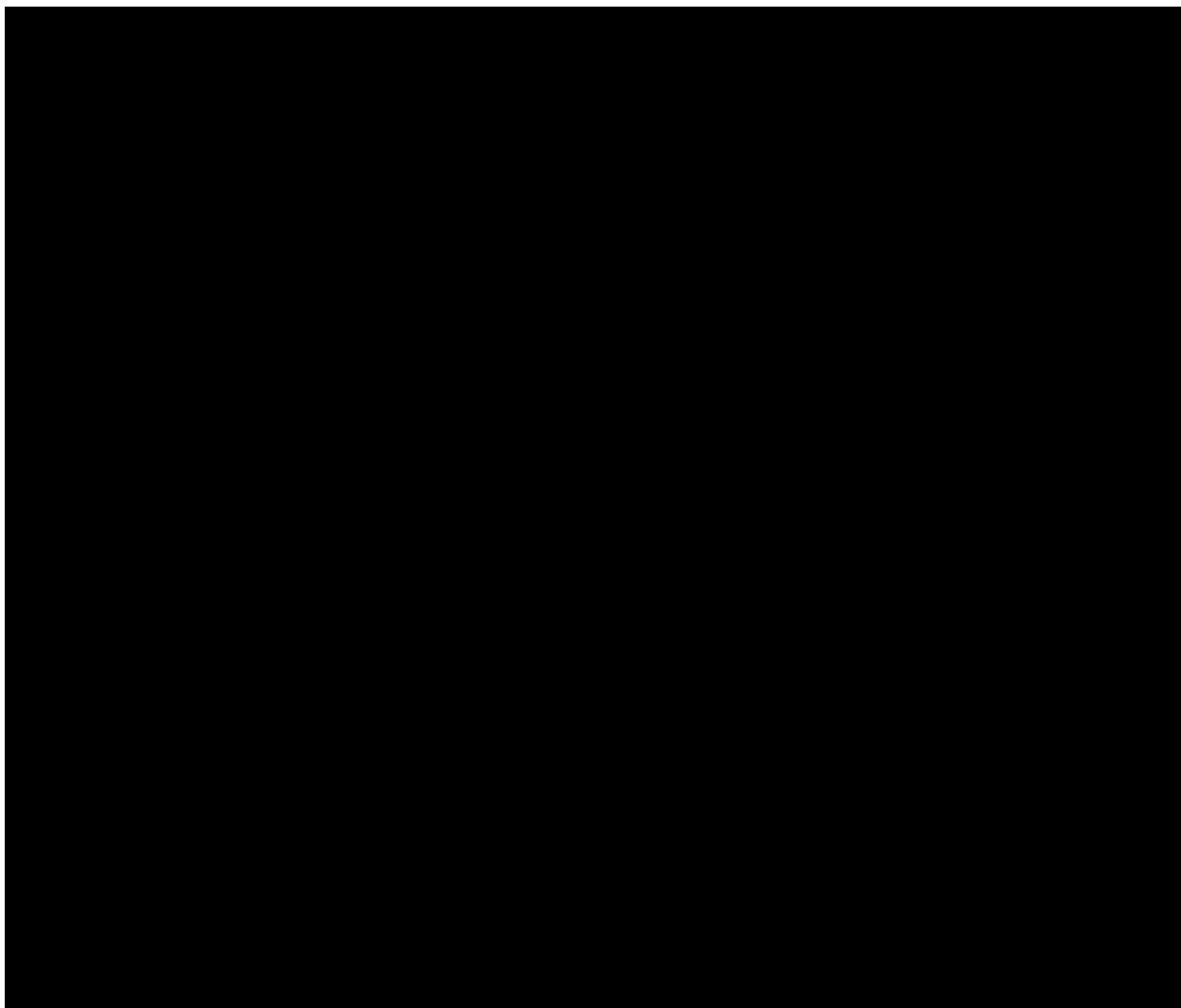
[Control cloud apps with policies](#)

If you run into any problems, we're here to help. To get assistance or support for your product issue, please [open a support ticket](#).

New AWS connector in Microsoft Defender for Cloud

Article • 04/27/2023

Episode description: In this episode of Defender for Cloud in the field, Or Serok joins Yuri Diogenes to share the new AWS connector in Microsoft Defender for Cloud, which was released at Ignite 2021. Or explains the use case scenarios for the new connector and how the new connector work. She demonstrates the onboarding process to connect AWS with Microsoft Defender for Cloud and talks about the centralized management of all security recommendations.



- [00:00 - Introduction](#)
- [2:20 - Understanding the new AWS connector.](#)
- [3:45 - Overview of the new onboarding experience.](#)

- [4:30](#) - Customizing recommendations for AWS workloads.
- [7:03](#) - Beyond CSPM capabilities.
- [11:14](#) - Demonstration of the recommendations and onboarding process.
- [23:20](#) - Demonstration of how to customize AWS assessments.

Recommended resources

Learn more about the new [AWS connector](#)

- Subscribe to [Microsoft Security on YouTube](#)
- Follow us on social media: [LinkedIn](#) [Twitter](#)
- Join our [Tech Community](#)
- For more about [Microsoft Security](#)

Next steps

[Integrate Azure Purview with Microsoft Defender for Cloud](#)

Configure Git credentials & connect a remote repo to Azure Databricks

Article • 01/26/2024

This article describes how to configure your Git credentials in Databricks so that you can connect a remote repo to Databricks Repos.

For a list of supported Git providers (cloud and on-premises), read [Supported Git providers](#).

Configuring GitHub and GitHub AE credentials

The following information applies to GitHub and GitHub AE users.

Why use the Databricks GitHub App instead of a PAT?

Databricks Repos allows you to choose the Databricks GitHub App for user authentication instead of PATs if you are using a hosted GitHub account. Using the GitHub App provides the following benefits over PATs:

- It uses OAuth 2.0 for user authentication. OAuth 2.0 repo traffic is encrypted for strong security.
- It is easier to integrate ([see the steps below](#)) and does not require individual tracking of tokens.
- Token renewal is handled automatically.
- The integration can be scoped to specific attached Git repos, allowing you more granular control over access.

Important

As per standard OAuth 2.0 integration, Databricks stores a user's access and refresh tokens—all other access control is handled by GitHub. Access and refresh tokens follow GitHub's default expiry rules with access tokens expiring after 8 hours (which minimizes risk in the event of credential leak). Refresh tokens have a 6-month lifetime if unused. Linked credentials expire after 6 months of inactivity, requiring the user to reconfigure them.

You can optionally encrypt Databricks tokens using [customer-managed keys](#) (CMK).

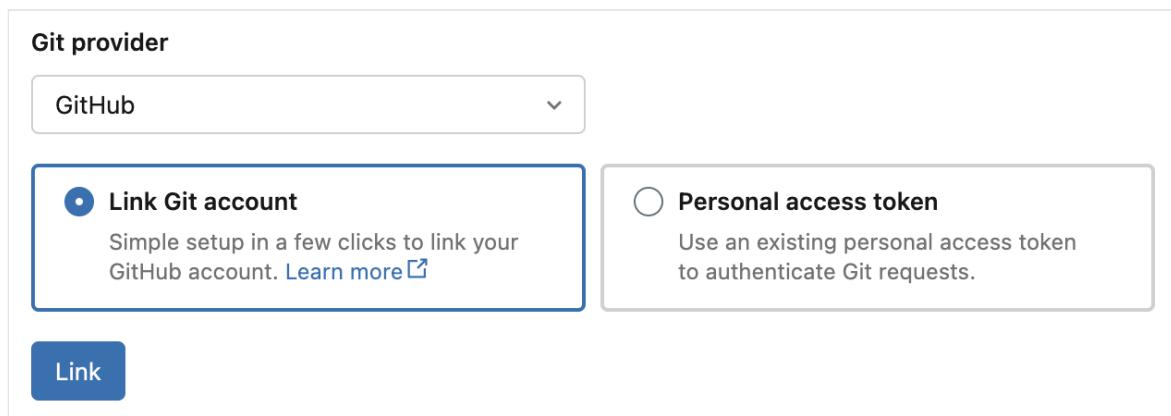
Link GitHub account using Databricks GitHub App

Note

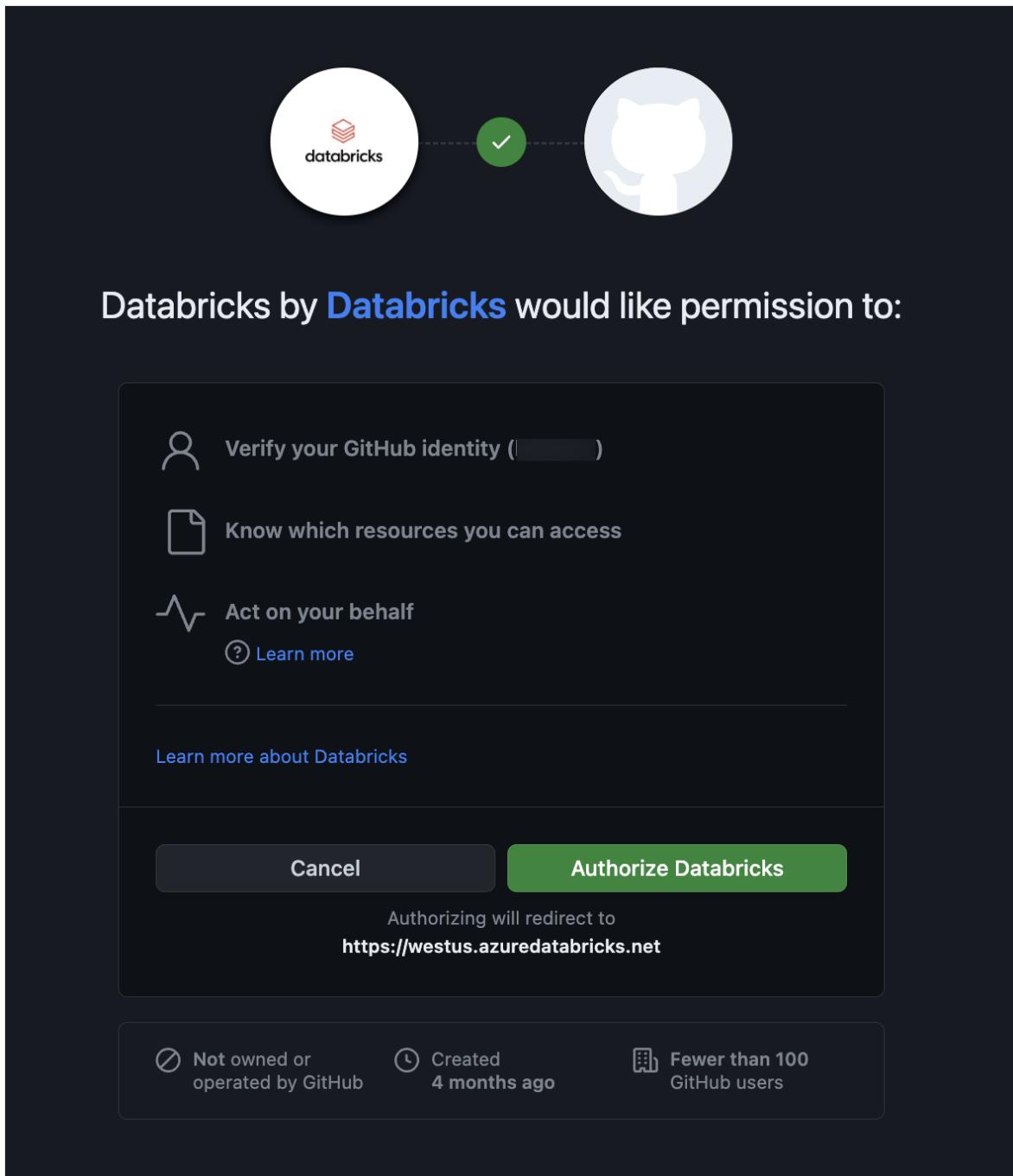
- This feature is not supported in GitHub Enterprise Server. Use a personal access token instead.

In Azure Databricks, link your GitHub account on the User Settings page:

1. In the upper-right corner of any page, click your username, then select **User Settings**.
2. Click the **Linked accounts** tab.
3. Change your provider to GitHub, select **Link Git account**, and click **Link**.



4. The Databricks GitHub app authorization page appears. Authorize the app to complete the setup. Authorizing the app allows Databricks to act on your behalf when you perform Git operations in Repos (such as cloning a repository). See the [GitHub documentation](#) for more details on app authorization.

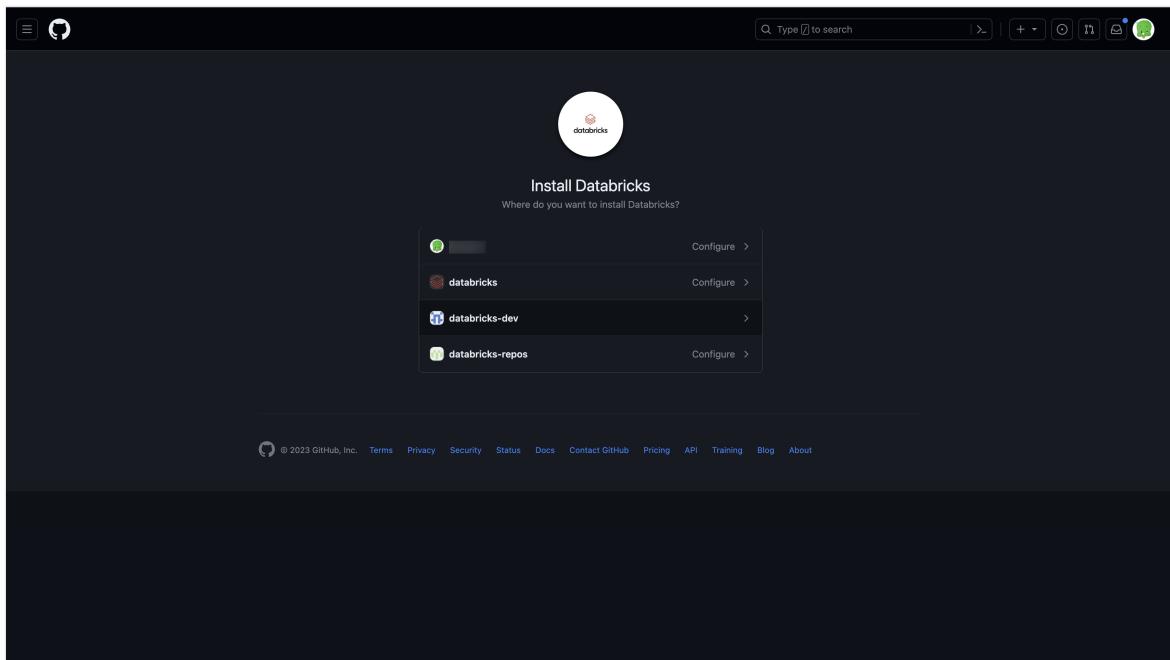


5. To allow access to GitHub repositories, follow the steps below to install and configure the Databricks GitHub app.

Install and configure the [Databricks GitHub app](#) to allow access to repositories

You must install and configure the Databricks GitHub app on GitHub repositories that you want to access from Databricks Repos. See the [GitHub documentation](#) for more details on app installation.

1. Open the [Databricks GitHub app installation page](#).
2. Select the account that owns the repositories you want to access.



3. If you are not an owner of the account, you must have the account owner install and configure the app for you.
4. If you are the account owner, install the app. Installing the app gives read and write access to code. Code is only accessed on behalf of users (for example, when a user clones a repository in Databricks Repos).
5. Optionally, you can give access to only a subset of repositories by selecting the **Only select repositories** option.

Connect to a GitHub repo using a personal access token

In GitHub, follow these steps to create a personal access token that allows access to your repositories:

1. In the upper-right corner of any page, click your profile photo, then click **Settings**.
2. Click **Developer settings**.
3. Click the **Personal access tokens** tab.
4. Click the **Generate new token** button.
5. Enter a token description.
6. Select the **repo** scope and **workflow** scope, and click the **Generate token** button. **workflow** scope is needed in case your repository has GitHub Action workflows.

OAuth Apps

GitHub Apps

Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Token description

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations

7. Copy the token to your clipboard. You enter this token in Azure Databricks under [User Settings > Linked accounts](#).

To use single sign-on, see [Authorizing a personal access token for use with SAML single sign-on](#).

GitLab

In GitLab, follow these steps to create a personal access token that allows access to your repositories:

1. From GitLab, click your user icon in the upper right corner of the screen and select [Preferences](#).
2. Click [Access Tokens](#) in the sidebar.

GitLab Projects Groups More Search or jump to... User Settings > Access Tokens

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

Name

Expires at

Scopes

api
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

read_user
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

read_api
Grants read access to the API, including all groups and projects, the container registry, and the package registry.

read_repository
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

write_repository
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

read_registry
Grants read-only access to container registry images on private projects.

write_registry
Grants write access to container registry images on private projects.

Create personal access token

3. Enter a name for the token.
4. Check the `read_repository` and `write_repository` permissions, and click **Create personal access token**.
5. Copy the token to your clipboard. Enter this token in Azure Databricks under **User Settings > Linked accounts**.

See the [GitLab documentation](#) to learn more about how to create and manage personal access tokens.

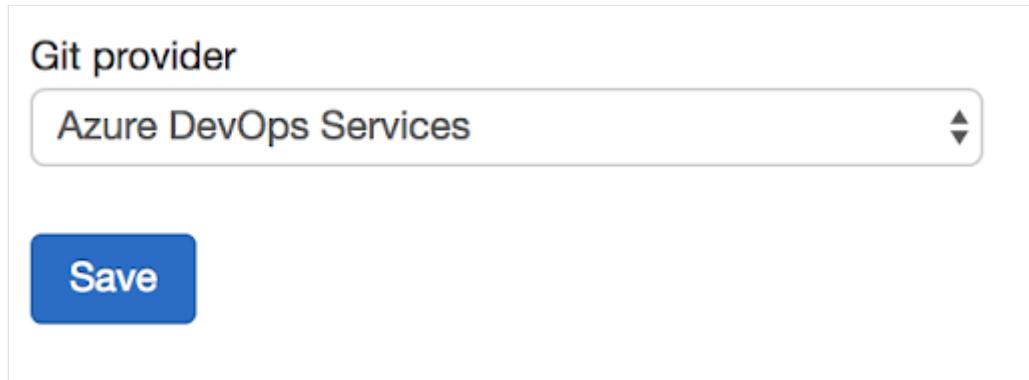
Azure DevOps Services

Connect to an Azure DevOps repo using Microsoft Entra ID (formerly Azure Active Directory)

Authentication with [Azure DevOps Services](#) is done automatically when you authenticate using Microsoft Entra ID. The Azure DevOps Services organization must be linked to the same Microsoft Entra ID tenant as Databricks.

In Azure Databricks, set your Git provider to Azure DevOps Services on the User Settings page:

1. In the upper-right corner of any page, click your username, then select **User Settings**.
2. Click the **Linked accounts** tab.
3. Change your provider to Azure DevOps Services.



Connect to an Azure DevOps repo using a token

The following steps show you how to connect an Azure Databricks repo to an Azure DevOps repo when they aren't in the same Microsoft Entra ID tenancy.

Get an access token for the repository in Azure DevOps:

1. Go to dev.azure.com, and then sign in to the DevOps organization containing the repository you want to connect Azure Databricks to.
2. In the upper-right side, click the User Settings icon and select **Personal Access Tokens**.
3. Click **+ New Token**.
4. Enter information into the form:
 - a. Name the token.
 - b. Select the organization name, which is the repo name.
 - c. Set an expiration date.
 - d. Choose the the scope required, such as **Full access**.
5. Copy the access token displayed.
6. Enter this token in Azure Databricks under **User Settings > Linked accounts**.
7. In **Git provider username or email**, enter the email address you use to log in to the DevOps organization.

Bitbucket

In Bitbucket, follow these steps to create an app password that allows access to your repositories:

1. Go to Bitbucket Cloud and create an app password that allows access to your repositories. See the [Bitbucket Cloud documentation](#).
2. Record the password.
3. In Azure Databricks, enter this password under **User Settings > Linked accounts**.

Tutorial: Create AWS infrastructure to host a Service Fabric cluster

Article • 07/15/2022

Service Fabric standalone clusters offer you the option to choose your own environment and create a cluster as part of the "any OS, any cloud" approach that Service Fabric is taking. In this tutorial series, you create a standalone cluster hosted on AWS and install an application into it.

This tutorial is part one of a series. In this article, you generate the AWS resources required to host your standalone cluster of Service Fabric. In future articles you need install the Service Fabric standalone suite, install a sample application into your cluster, and finally, clean up your cluster.

In part one of the series, you learn how to:

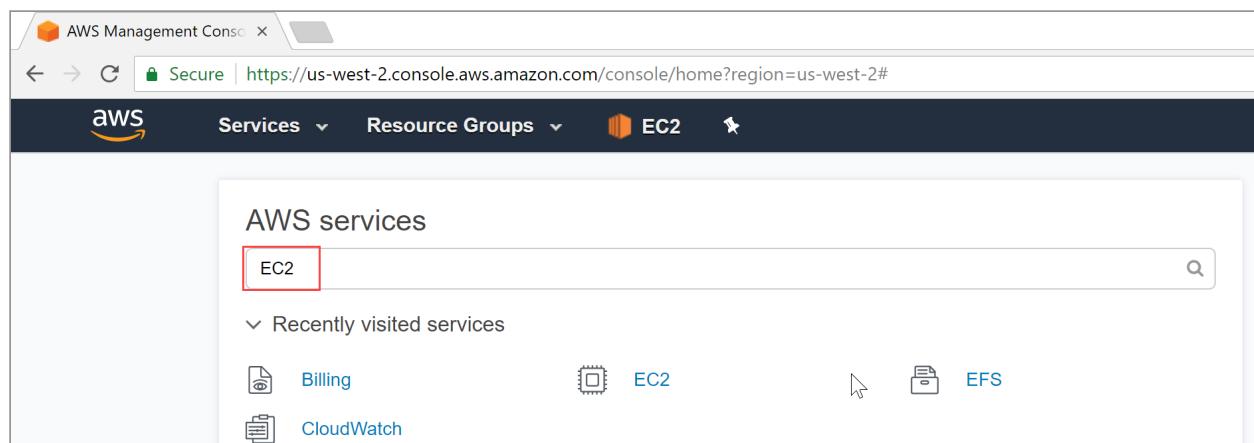
- ✓ Create a set of EC2 instances
- ✓ Modify the security group
- ✓ Sign in to one of the instances
- ✓ Prep the instance for Service Fabric

Prerequisites

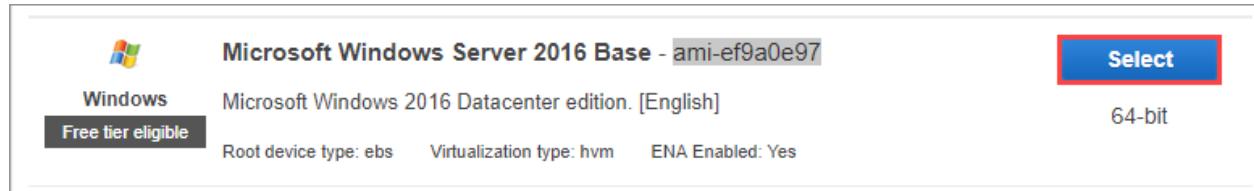
To complete this tutorial, you need an AWS account. If you don't already have an account, go to the [AWS console](#) to create one.

Create EC2 instances

Sign in to the AWS Console > Enter EC2 in the search box > **EC2 Virtual Servers in the Cloud**



Select **Launch Instance**, on the next screen choose **Select** next to Microsoft Windows Server 2016 Base.



Select **t2.medium**, then select **Next: Configure Instance Details**, on the next screen change the number of instances to **3**, then select **Advanced Details** to expand that section.

To connect your virtual machines together in Service Fabric, the VMs that are hosting your infrastructure need to have the same credentials. There are two common ways to get consistent credentials: join them all to the same domain, or set the same administrator password on each VM. For this tutorial, you use a user data script to set the EC2 instances to all have the same password. In a production environment, joining the hosts to a windows domain is more secure.

Enter the following script in the user data field on the console:

```
PowerShell

<powershell>
$user = [adsi]"WinNT://localhost/Administrator,user"
$user.SetPassword("serv1ceF@bricP@ssword")
$user.SetInfo()
netsh advfirewall firewall set rule group="File and Printer Sharing" new
enable=Yes
New-NetFirewallRule -DisplayName "Service Fabric Ports" -Direction Inbound -
Action Allow -RemoteAddress LocalSubnet -Protocol TCP -LocalPort 135, 137-
139, 445
</powershell>
```

Once you've entered the PowerShell script select **Review and Launch**



On the review screen, select **Launch**. Then change the drop-down to **Proceed without a key pair** and select the checkbox indicating that you know the password.



Finally, select **Launch Instances**, and then **View Instances**. You have the basis for your Service Fabric cluster created, now you need to add a few final configurations to the instances themselves to prep them for the Service Fabric configuration.

Modify the security group

Service Fabric requires a number of ports open between the hosts in your cluster. To open these ports in the AWS infrastructure, select one of the instances that you created. Then select the name of the security group, for example, **launch-wizard-1**. Now, select the **Inbound** tab.

To avoid opening these ports to the world, you instead open them only for hosts in the same security group. Take note of the security group ID, in the example it's **sg-c4fb1eba**. Then select **Edit**.

Next, add four rules to the security group for service dependencies, and then three more for Service Fabric itself. The first rule is to allow ICMP traffic, for basic connectivity checks. The others rules open the required ports to enable SMB and Remote Registry.

For the first rule select **Add Rule**, then from the dropdown menu selects **All ICMP - IPv4**. Select the entry box next to custom and enter your security group ID from above.

For the last three dependencies, you need to follow a similar process. Select **Add Rule**, from the drop-down select **Custom TCP Rule**, in the port range enter one of **135**, **137**-**139**, and **445** for each rule. Finally, in the source box enter your security group ID.

Edit inbound rules

Type	Protocol	Port Range	Source	Description	Actions	
Custom TC	TCP	3389	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop	
Custom ICN	Echo Rep	N/A	Custom	sg-c4fb1eba	e.g. SSH for Admin Desktop	
Custom TC	TCP	135	Custom	sg-c4fb1eba	e.g. SSH for Admin Desktop	
Custom TC	TCP	137-139	Custom	sg-c4fb1eba	e.g. SSH for Admin Desktop	
Custom TC	TCP	445	Custom	sg-c4fb1eba	e.g. SSH for Admin Desktop	

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel **Save**

Now that the ports for the dependencies are open, you need to do the same thing for the ports that Service Fabric itself uses to communicate. Select **Add Rule**, from the drop-down select **Custom TCP Rule**, in the port range enter **20001-20031** enter the security group in the source box.

Next, add a rule for the ephemeral port range. Select **Add Rule**, from the drop-down select **Custom TCP Rule**, in the port range enter **20606-20861**. Finally, in the source box enter your security group ID.

For the final two rules for Service Fabric, open it up to the world so you can manage your service fabric cluster from your personal computer. Select **Add Rule**, from the drop-down select **Custom TCP Rule**, in the port range enter one of **19000-19003**, and **19080-19081** then change the Source drop down to Anywhere.

Finally, we just need to open up port 8080 so you can see the application when it's deployed. Select **Add Rule**, from the drop-down select **Custom TCP Rule**, in the port range enter **8080** then change the Source drop down to Anywhere.

All of the rules are now entered. Select **Save**.

Connect to an instance and validate connectivity

From the security group tab, select **Instances** from the left-hand menu. Select each of the instances that you've created and note their private IP addresses for the examples below will use **172.31.21.141** and **172.31.20.163**.

Once you have all of the IP addresses select one of the instances to connect to, right-click on the instance and select **Connect**. From here, you can download the RDP file for this particular instance. Select **Download Remote Desktop File**, and then open the file

that is downloaded to establish your remote desktop connection (RDP) to this instance. When prompted enter your password `serv1ceF@bricP@ssword`.

Connect To Your Instance

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

[Download Remote Desktop File](#)

If you've joined your instance to a directory, you can use your directory credentials to connect to your instance.

If you need any assistance connecting to your instance, please see our [connection documentation](#).

[Close](#)

Once you have successfully connected to your instance validate that you can connect between them and also share files. You've gathered the IP addresses for all the instances, select one that you are not currently connected to. Go to **Start**, enter `cmd` and select **Command Prompt**.

In these examples the RDP connection was established to the following IP address: 172.31.21.141. All connectivity test then occur to the other IP address: 172.31.20.163.

To validate that basic connectivity works, use the ping command.

```
ping 172.31.20.163
```

If your output looks like `Reply from 172.31.20.163: bytes=32 time<1ms TTL=128` repeated four times then your connection between the instances is working. Now validate that your SMB sharing works with the following command:

```
net use * \\172.31.20.163\c$
```

It should return `Drive Z: is now connected to \\172.31.20.163\c$.` as the output.

Prep instances for Service Fabric

If you were creating this from scratch, you'd need to take a couple extra steps. Namely, you'd need to validate that remote registry was running, enable SMB, and open the requisite ports for SMB and remote registry.

To make it easier you embedded all of this work when you bootstrapped the instances with your user data script.

To enable SMB, this is the PowerShell command you used:

PowerShell

```
netsh advfirewall firewall set rule group="File and Printer Sharing" new  
enable=Yes
```

To open the ports in the firewall here is the PowerShell command:

PowerShell

```
New-NetFirewallRule -DisplayName "Service Fabric Ports" -Direction Inbound -  
Action Allow -RemoteAddress LocalSubnet -Protocol TCP -LocalPort 135, 137-  
139, 445
```

Next steps

In part one of the series, you learned how to launch three EC2 instances and get them configured for the Service Fabric installation:

- ✓ Create a set of EC2 instances
- ✓ Modify the security group
- ✓ Sign in to one of the instances
- ✓ Prep the instance for Service Fabric

Advance to part two of the series to configure Service Fabric on your cluster.

[Install Service Fabric](#)

How to connect AWS and Azure using a BGP-enabled VPN gateway

Article • 08/10/2023

This article walks you through the setup of a BGP-enabled connection between Azure and Amazon Web Services (AWS). You'll use an Azure VPN gateway with BGP and active-active enabled and an AWS virtual private gateway with two site-to-site connections.

Architecture

In this setup, you create the following resources:

Azure

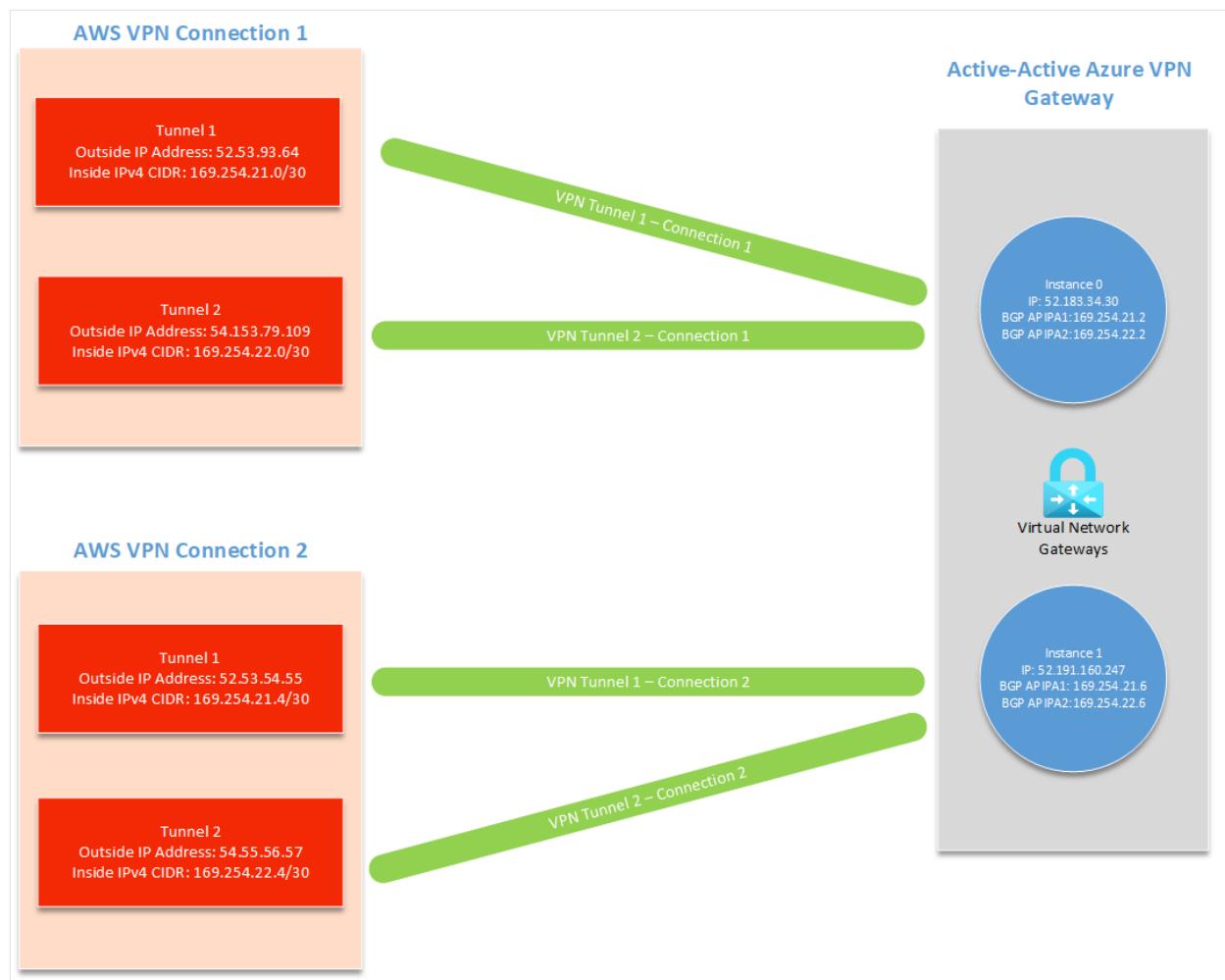
- One virtual network
- One virtual network gateway with active-active and BGP enabled
- Four local network gateways
- Four site-to-site connections

AWS

- One virtual private cloud (VPC)
- One virtual private gateway
- Two customer gateways
- Two site-to-site connections, each with two tunnels (total of four tunnels)

A site-to-site connection on AWS has two tunnels, each with their own outside IP address and inside IPv4 CIDR (used for BGP APIPA). An active-passive VPN gateway only supports **one** custom BGP APIPA. You'll need to enable **active-active** on your Azure VPN gateway to connect to multiple AWS tunnels.

On the AWS side, you create a customer gateway and site-to-site connection for **each of the two Azure VPN gateway instances** (total of four outgoing tunnels). In Azure, you need to create four local network gateways and four connections to receive these four AWS tunnels.



Choosing BGP APIPA addresses

You can use the following values for your BGP APIPA configuration throughout the tutorial.

Tunnel	Azure Custom Azure APIPA BGP IP Address	AWS BGP Peer IP Address	AWS Inside IPv4 CIDR
AWS Tunnel 1 to Azure Instance 0	169.254.21.2	169.254.21.1	169.254.21.0/30
AWS Tunnel 2 to Azure Instance 0	169.254.22.2	169.254.22.1	169.254.22.0/30
AWS Tunnel 1 to Azure Instance 1	169.254.21.6	169.254.21.5	169.254.21.4/30
AWS Tunnel 2 to Azure Instance 1	169.254.22.6	169.254.22.5	169.254.22.4/30

You can also set up your own custom APIPA addresses. AWS requires a /30 **Inside IPv4 CIDR** in the APIPA range of **169.254.0.0/16** for each tunnel. This CIDR must also be in the Azure-reserved APIPA range for VPN, which is from **169.254.21.0** to **169.254.22.255**. AWS

will use the first IP address of your /30 inside CIDR and Azure will use the second. This means you need to reserve space for two IP addresses in your AWS /30 CIDR.

For example, if you set your AWS Inside IPv4 CIDR to be 169.254.21.0/30, AWS will use the BGP IP address 169.254.21.1 and Azure will use the IP address 169.254.21.2.

ⓘ Important

- Your APIPA addresses must not overlap between the on-premises VPN devices and all connected Azure VPN gateways.
- If you choose to configure multiple APIPA BGP peer addresses on the VPN gateway, you must also configure all Connection objects with their corresponding IP address of your choice. If you fail to do so, all connections use the first APIPA IP address in the list no matter how many IPs are present.

Prerequisites

You must have both an Azure account and AWS account with an active subscription. If you don't already have an Azure subscription, you can activate your [MSDN subscriber benefits](#) or sign up for a [free account](#).

Part 1: Create an active-active VPN gateway in Azure

Create a VNet

Create a virtual network with the following values. You can refer to the steps in the [Site-to-site Tutorial](#).

- **Subscription:** If you have more than one subscription, verify that you're using the correct one.
- **Resource group:** TestRG1
- **Name:** VNet1
- **Location:** East US
- **IPv4 address space:** 10.1.0.0/16
- **Subnet name:** FrontEnd
- **Subnet address range:** 10.1.0.0/24

Create an active-active VPN gateway with BGP

Create a VPN gateway using the following values:

- **Name:** VNet1GW
- **Region:** East US
- **Gateway type:** VPN
- **VPN type:** Route-based
- **SKU:** VpnGw2AZ
- **Generation:** Generation2
- **Virtual network:** VNet1
- **Gateway subnet address range:** 10.1.1.0/24
- **Public IP address:** Create new
- **Public IP address name:** VNet1GWpip
- **Availability zone:** Zone-redundant
- **Enable active-active mode:** Enabled
- **SECOND PUBLIC IP ADDRESS:** Create new
- **Public IP address 2 name:** VNet1GWpip2
- **Availability zone:** Zone-redundant
- **Configure BGP:** Enabled
- **Autonomous system number (ASN):** 65000
- **Custom Azure APIPA BGP IP address:** 169.254.21.2, 169.254.22.2
- **Second Custom Azure APIPA BGP IP address:** 169.254.21.6, 169.254.22.6

1. In the Azure portal, navigate to the **Virtual network gateway** resource from the Marketplace, and select **Create**.
2. Fill in the parameters as shown in the following examples.

Create virtual network gateway

Basics Tags Review + create

Azure has provided a planning and design guide to help you configure the various VPN gateway options. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Content Development

Resource group ⓘ

TestRG1 (derived from virtual network's resource group)

Instance details

Name *

VNet1GW

Region *

East US

Gateway type * ⓘ

VPN ExpressRoute

VPN type * ⓘ

Route-based Policy-based

SKU * ⓘ

VpnGw2AZ

Generation ⓘ

Generation2

Virtual network * ⓘ

VNet1

[Create virtual network](#)

Only virtual networks in the currently selected subscription and region are listed.

Gateway subnet address range * ⓘ

10.1.1.0/24

10.1.1.0 - 10.1.1.255 (256 addresses)



3. Configure both Public IP addresses and enable active-active mode. The public IP address objects created here are associated to the VPN gateway. The public IP address is dynamically assigned to the object when the VPN gateway is created.

Public IP address

Public IP address * ⓘ

Create new Use existing

Public IP address name *

VNet1GWpip

Public IP address SKU

Standard

Assignment

Dynamic Static

Availability zone *

Zone-redundant

Enable active-active mode * ⓘ

Enabled Disabled

SECOND PUBLIC IP ADDRESS

SECOND PUBLIC IP ADDRESS * ⓘ

Create new Use existing

Public IP address name *

VNet1GWpip2

Public IP address SKU

Standard

Availability zone *

Zone-redundant



4. Configure BGP.

Configure BGP * Enabled Disabled

Autonomous system number (ASN) * 65000

Custom Azure APIPA BGP IP address 169.254.21.2

Peer Address

Second Custom Azure APIPA BGP IP address 169.254.21.6

Peer Address

- Select **Enabled** for **Configure BGP** to show the BGP configuration section.
- Fill in a **ASN (Autonomous System Number)**. This ASN must be different than the ASN used by AWS.
- Add two addresses to **Custom Azure APIPA BGP IP address**. Include the IP addresses for **AWS Tunnel 1 to Azure Instance 0** and **AWS Tunnel 2 to Azure Instance 0** from the [APIPA configuration you chose](#). The second input will only appear after you add your first APIPA BGP IP address.
- Add two addresses to **Second Custom Azure APIPA BGP IP address**. Include the IP addresses for **AWS Tunnel 1 to Azure Instance 1** and **AWS Tunnel 2 to Azure Instance 1** from the [APIPA configuration you chose](#). The second input will only appear after you add your first APIPA BGP IP address.

5. Select **Review + create** to run validation. Once validation passes, select **Create** to deploy the VPN gateway. Creating a gateway can often take 45 minutes or more, depending on the selected gateway SKU. You can see the deployment status on the Overview page for your gateway.

Part 2: Connect to your VPN gateway from AWS

In this section, you connect to your Azure VPN gateway from AWS. For updated instructions, always refer to the [official AWS documentation](#).

Create a VPC

Create a VPC using the following values and the [most recent AWS documentation](#).

- **Name:** VPC1
- **CIDR block:** 10.2.0.0/16

Make sure that your CIDR block doesn't overlap with the virtual network you created in Azure.

Create a virtual private gateway

Create a virtual private gateway using the following values and the [most recent AWS documentation](#).

- **Name:** AzureGW
- **ASN:** Amazon default ASN (64512)
- **VPC:** Attached to VPC1

If you choose to use a custom ASN, make sure it's different than the ASN you used in Azure.

Enable route propagation

Enable route propagation on your virtual private gateway using the [most recent AWS documentation](#).

Create customer gateways

Create two customer gateways using the following values and the [most recent AWS documentation](#).

Customer gateway 1 settings:

- **Name:** ToAzureInstance0
- **Routing:** Dynamic
- **BGP ASN:** 65000 (the ASN for your Azure VPN gateway)
- **IP Address:** the first public IP address of your Azure VPN gateway

Customer gateway 2 settings:

- **Name:** ToAzureInstance1
- **Routing:** Dynamic
- **BGP ASN:** 65000 (the ASN for your Azure VPN gateway)
- **IP Address:** the second public IP address of your Azure VPN gateway

You can locate your **Public IP address** and your **Second Public IP address** on Azure in the **Configuration** section of your virtual network gateway.

Create site-to-site VPN connections

Create two site-to-site VPN connections using the following values and the [most recent AWS documentation](#).

Site-to-site connection 1 settings:

- **Name:** ToAzureInstance0
- **Target Gateway Type:** Virtual Private Gateway
- **Virtual Private Gateway:** AzureGW
- **Customer Gateway:** Existing
- **Customer Gateway:** ToAzureInstance0
- **Routing Options:** Dynamic (requires BGP)
- **Local IPv4 Network CIDR:** 0.0.0.0/0
- **Tunnel Inside Ip Version:** IPv4
- **Inside IPv4 CIDR for Tunnel 1:** 169.254.21.0/30
- **Pre-Shared Key for Tunnel 1:** choose a secure key
- **Inside IPv4 CIDR for Tunnel 2:** 169.254.22.0/30
- **Pre-Shared Key for Tunnel 2:** choose a secure key
- **Startup Action:** Start

Site-to-site connection 2 settings:

- **Name:** ToAzureInstance1
- **Target Gateway Type:** Virtual Private Gateway
- **Virtual Private Gateway:** AzureGW
- **Customer Gateway:** Existing
- **Customer Gateway:** ToAzureInstance1
- **Routing Options:** Dynamic (requires BGP)
- **Local IPv4 Network CIDR:** 0.0.0.0/0
- **Tunnel Inside Ip Version:** IPv4
- **Inside IPv4 CIDR for Tunnel 1:** 169.254.21.4/30
- **Pre-Shared Key for Tunnel 1:** choose a secure key
- **Inside IPv4 CIDR for Tunnel 2:** 169.254.22.4/30
- **Pre-Shared Key for Tunnel 2:** choose a secure key
- **Startup Action:** Start

For **Inside IPv4 CIDR for Tunnel 1** and **Inside IPv4 CIDR for Tunnel 2** for both connections, refer to the APIPA configuration you [choose](#).

Part 3: Connect to your AWS customer gateways from Azure

Next, you connect your AWS tunnels to Azure. For each of the four tunnels, you'll have both a local network gateway and a site-to-site connection.

Important

Repeat the following sections for **each of your four AWS tunnels**, using their respective **outside IP address**.

Create local network gateways

Repeat these instructions to create each local network gateway.

1. In the Azure portal, navigate to the **Local network gateway** resource from the Marketplace, and select **Create**.
2. Select the same **Subscription**, **Resource Group**, and **Region** you used to create your virtual network gateway.
3. Enter a name for your local network gateway.
4. Leave **IP Address** as the value for **Endpoint**.
5. For **IP Address**, enter the **Outside IP Address** (from AWS) for the tunnel you're creating.
6. Leave **Address Space** as blank and select **Advanced**.

Basics Advanced Review + create

A local network gateway is a specific object that represents an on-premises location (the site) for routing purposes. [Learn more](#).

Project details

Subscription *

Resource group * [Create new](#)

Instance details

Region *

Name * ✓

Endpoint [ⓘ](#)

IP address * [ⓘ](#) ✓

Address space [ⓘ](#)

7. On the **Advanced** tab, configure the following settings:

- Select **Yes** for **Configure BGP settings**.
- For **Autonomous system number (ASN)**, enter the ASN for your AWS Virtual Private Network. Use the ASN **64512** if you left your ASN as the AWS default value.
- For **BGP peer IP address**, enter the AWS BGP Peer IP Address based on the [APIPA configuration you chose](#).

Basics **Advanced** Review + create

Configure BGP settings

Autonomous system number (ASN) * [ⓘ](#) ✓

BGP peer IP address * ✓

Create connections

Repeat these steps to create each of the required connections.

1. Open the page for your **virtual network gateway**, navigate to the **Connections** page.
2. On the **Connections** page, select **+ Add**.

3. On the **Basics** page, complete the following values:

- **Connection type:** Site-to-site (IPsec)
- **Name:** Enter a name for your connection. Example: AWSTunnel1toAzureInstance0.

4. On the **Settings** page, complete the following values:

- **Virtual network gateway:** Select the VPN gateway.
- **Local network gateway:** Select the local network gateway you created.
- Enter the **Shared key (PSK)** that matches the preshared key you entered when making the AWS connections.
- **Enable BGP:** Select to enable.
- **Enable Custom BGP Addresses:** Select to enable.

5. Under **Custom BGP Addresses:**

- Enter the Custom BGP Address based on the [APIPA configuration you chose](#).
- The **Custom BGP Address** (Inside IPv4 CIDR in AWS) must match with the **IP Address** (Outside IP Address in AWS) that you specified in the local network gateway you're using for this connection.
- Only one of the two custom BGP addresses will be used, depending on the tunnel you're specifying it for.
- For making a connection from AWS to the **first public IP address** of your VPN gateway (instance 0), **only the Primary Custom BGP Address** is used.
- For making a connection from AWS to the **second public IP address** of your VPN gateway (instance 1), **only the Secondary Custom BGP Address** is used.
- Leave the other **Custom BGP Address** as default.

If you used the [default APIPA configuration](#), you can use the following addresses.

Tunnel	Primary Custom BGP Address	Secondary Custom BGP Address
AWS Tunnel 1 to Azure Instance 0	169.254.21.2	Not used (select 169.254.21.6)
AWS Tunnel 2 to Azure Instance 0	169.254.22.2	Not used (select 169.254.21.6)
AWS Tunnel 1 to Azure Instance 1	Not used (select 169.254.21.2)	169.254.21.6
AWS Tunnel 2 to Azure Instance 1	Not used (select 169.254.21.2)	169.254.22.6

6. Configure the following settings:

- **FastPath:** leave the default (deselected)
- **IPsec / IKE policy:** Default
- **Use policy based traffic selector:** Disable
- **DPD timeout in seconds:** leave the default
- **Connection Mode:** You can select any of the available options (Default, Initiator Only, Responder Only). For more information, see [VPN Gateway settings - connection modes](#).

7. Select **Save**.

8. **Review + create** to create the connection.

9. Repeat these steps to create additional connections.

10. Before continuing to the next section, verify that you have a **local network gateway** and **connection** for each of your four AWS tunnels.

Part 4: (Optional) Check the status of your connections

Check your connections status on Azure

1. Open the page for your **virtual network gateway**, navigate to the **Connections** page.
2. Verify that all 4 connections show as **Connected**.

AWSTunnel1ToInstance0	Connected	Site-to-site (IPsec)	AWSTunnel1ToInstance0	***
AWSTunnel1ToInstance1	Connected	Site-to-site (IPsec)	AWSTunnel1ToInstance1	***
AWSTunnel2ToInstance0	Connected	Site-to-site (IPsec)	AWSTunnel2ToInstance0	***
AWSTunnel2ToInstance1	Connected	Site-to-site (IPsec)	AWSTunnel2ToInstance1	***

Check your BGP peers status on Azure

1. Open the page for your **virtual network gateway**, navigate to the **BGP Peers** page.
2. In the **BGP Peers** table, verify that all of the connections with the **Peer address** you specified show as **Connected** and are exchanging routes.

BGP peers															
Peer address	↑↓	Local address	↑↓	Asn	↑↓	Status	↑↓	Connected duration	↑↓	Routes received	↑↓	Messages sent	↑↓	Messages Received	↑↓
169.254.21.1		10.1.1.5		64512		Connected		00:08:53.8145971		1		71		64	...
169.254.21.5		10.1.1.5		64512		Connecting		-		0		0		0	...
169.254.22.5		10.1.1.5		64512		Connecting		-		0		0		0	...
169.254.22.1		10.1.1.5		64512		Connected		00:08:52.8976799		1		71		64	...
10.1.1.5		10.1.1.5		65000		Unknown		-		0		0		0	...
10.1.1.4		10.1.1.5		65000		Connected		01:22:04.4960357		3		130		130	...
169.254.21.1		10.1.1.4		64512		Connecting		-		0		0		0	...
169.254.21.5		10.1.1.4		64512		Connected		00:08:06.6855230		1		65		56	...
169.254.22.5		10.1.1.4		64512		Connected		00:07:40.1965117		1		60		54	...
169.254.22.1		10.1.1.4		64512		Connecting		-		0		0		0	...
10.1.1.5		10.1.1.4		65000		Connected		01:21:51.3370272		3		131		132	...
10.1.1.4		10.1.1.4		65000		Unknown		-		0		0		0	...

Check your connections status on AWS

1. Open the [Amazon VPC console](#) ↗
2. In the navigation pane, select **Site-to-Site VPN Connections**.
3. Select the first connection you made and then select the **Tunnel Details** tab.
4. Verify that the **Status** of both tunnels shows as **UP**.
5. Verify that the **Details** of both tunnels shows one or more BGP routes.

Next steps

For more information about VPN Gateway, see the [FAQ](#).

Azure for Google Cloud Professionals

Article • 11/30/2022

This article helps Google Cloud experts understand the basics of Microsoft Azure accounts, platform, and services. It also covers key similarities and differences between the Google Cloud and Azure platforms. (Note that Google Cloud was previously called Google Cloud Platform (GCP).)

You'll learn:

- How accounts and resources are organized in Azure.
- How available solutions are structured in Azure.
- How the major Azure services differ from Google Cloud services.

Azure and Google Cloud built their capabilities independently over time so that each has important implementation and design differences.

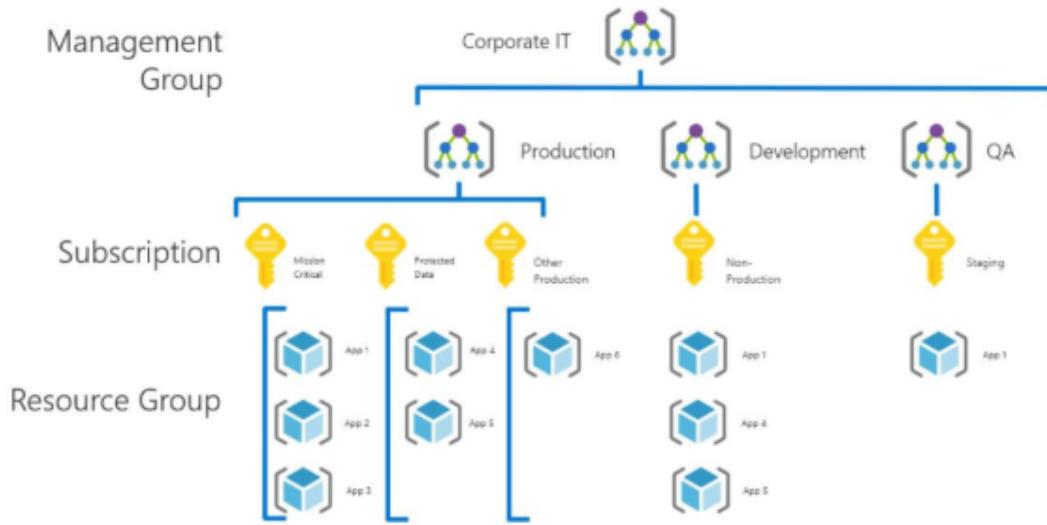
Azure for Google Cloud overview

Like Google Cloud, Microsoft Azure is built around a core set of compute, storage, database, and networking services. In many cases, both platforms offer a basic equivalence between the products and services they offer. Both Google Cloud and Azure allow you to build highly available solutions based on Linux or Windows hosts. So, if you're used to development using Linux and OSS technology, both platforms can do the job.

While the capabilities of both platforms are similar, the resources that provide those capabilities are often organized differently. Exact one-to-one relationships between the services required to build a solution are not always clear. In other cases, a particular service might be offered on one platform, but not the other.

Managing accounts and subscription

Azure has a hierarchy of Management group and subscriptions and resource groups to manage resources effectively. This is similar to the Folders and Project hierarchy for resources in Google Cloud.



Azure levels of management scope

- **Management groups:** These groups are containers that help you manage access, policy, and compliance for multiple subscriptions. All subscriptions in a management group automatically inherit the conditions applied to the management group.
- **Subscriptions:** A subscription logically associates user accounts and the resources that were created by those user accounts. Each subscription has limits or quotas on the amount of resources you can create and use. Organizations can use subscriptions to manage costs and the resources that are created by users, teams, or projects.
- **Resource groups:** A resource group is a logical container into which Azure resources like web apps, databases, and storage accounts are deployed and managed.
- **Resources:** Resources are instances of services that you create, like virtual machines, storage, or SQL databases.

Azure services can be purchased using several pricing options, depending on your organization's size and needs. See the [pricing overview](#) page for details.

[Azure subscriptions](#) are a grouping of resources with an assigned owner responsible for billing and permissions management.

A Google Cloud *project* is conceptually similar to the Azure subscription, in terms of billing, quotas, and limits. However, from a functional perspective, a Google Cloud project is more like a resource group in Azure. It's a logical unit that cloud resources are deployed to.

Note that unlike in Google Cloud, there is no maximum number of Azure subscriptions. Each Azure subscription is linked to a single Azure Active Directory (Azure AD) tenant

(an *account*, in Google Cloud terms). An Azure AD tenant can contain an unlimited number of subscriptions, whereas Google Cloud has a default limit of 30 projects per account.

Subscriptions are assigned three types of administrator accounts:

- **Account Administrator.** The subscription owner and the account billed for the resources used in the subscription. The account administrator can only be changed by transferring ownership of the subscription.
- **Service Administrator.** This account has rights to create and manage resources in the subscription but is not responsible for billing. By default, the account administrator and service administrator are assigned to the same account. The account administrator can assign a separate user to the service administrator account for managing the technical and operational aspects of a subscription. Only one service administrator is assigned per subscription.
- **Co-administrator.** There can be multiple co-administrator accounts assigned to a subscription. Co-administrators cannot change the service administrator, but otherwise have full control over subscription resources and users.

For fine-grained access management to Azure resources, you can use Azure role-based access control ([Azure RBAC](#)), which includes over 70 built-in roles. You can also create your own custom roles.

Below the subscription level user roles and individual permissions can also be assigned to specific resources. In Azure, all user accounts are associated with either a Microsoft Account or Organizational Account (an account managed through Azure AD).

Subscriptions have default service quotas and limits. For a full list of these limits, see [Azure subscription and service limits, quotas, and constraints](#). These limits can be increased up to the maximum by [filing a support request in the management portal](#).

See also

- [How to add or change Azure administrator roles](#)
- [How to download your Azure billing invoice and daily usage data](#)

Resource management

The term "resource" in Azure means any compute instance, storage object, networking device, or other entity you can create or configure within the platform.

Azure resources are deployed and managed using one of two models: [Azure Resource Manager](#), or the older Azure [classic deployment model](#). Any new resources are created using the Resource Manager model.

Resource groups

Azure additionally has an entity called "resource groups" that organize resources such as VMs, storage, and virtual networking devices. An Azure resource is always associated with one resource group. A resource created in one resource group can be moved to another group but can only be in one resource group at a time. For more information, see [Move Azure resources across resource groups, subscriptions, or regions](#). Resource groups are the fundamental grouping used by Azure Resource Manager.

Resources can also be organized using [tags](#). Tags are key-value pairs that allow you to group resources across your subscription irrespective of resource group membership.

Management interfaces

Azure offers several ways to manage your resources:

- [Web interface](#). The Azure portal provides a full web-based management interface for Azure resources.
- [REST API](#). The Azure Resource Manager REST API provides programmatic access to most of the features available in the Azure portal.
- [Command Line](#). The Azure CLI provides a command-line interface capable of creating and managing Azure resources. The Azure CLI is available for [Windows](#), [Linux](#), and [macOS](#).
- [PowerShell](#). The Azure modules for PowerShell allow you to execute automated management tasks using a script. PowerShell is available for [Windows](#), [Linux](#), and [macOS](#).
- [Templates](#). Azure Resource Manager templates provide JSON template-based resource management capabilities.
- [SDK](#). The SDKs are a collection of libraries that allows users to programmatically manage and interact with Azure services.

In each of these interfaces, the resource group is central to how Azure resources get created, deployed, or modified.

In addition, many third-party management tools like [Hashicorp's Terraform](#) and [Netflix Spinnaker](#), are also available on Azure.

See also

- [Azure resource group guidelines](#)

Regions and Availability Zones

Failures can vary in the scope of their impact. Some hardware failures, such as a failed disk, may affect a single host machine. A failed network switch could affect a whole server rack. Less common are failures that disrupt a whole datacenter, such as loss of power in a datacenter. In rare situations, an entire region could become unavailable.

One of the main ways to make an application resilient is through redundancy. However, you need to plan for this redundancy when you design the application. Also, the level of redundancy that you need depends on your business requirements. Not every application needs redundancy across regions to guard against a regional outage. In general, a tradeoff exists between greater redundancy and reliability versus higher cost and complexity.

In Google Cloud, a region has two or more Availability Zones. An Availability Zone corresponds with a physically isolated datacenter in the geographic region. Azure has numerous features for providing application redundancy at every level of potential failure, including **availability sets** , **availability zones** , and **paired regions**.

The following table summarizes each option.

	Availability Set	Availability Zone	Paired region
Scope of failure	Rack	Datacenter	Region
Request routing	Load Balancer	Cross-zone Load Balancer	Traffic Manager
Network latency	Very low	Low	Mid to high
Virtual networking	VNet	VNet	Cross-region VNet peering

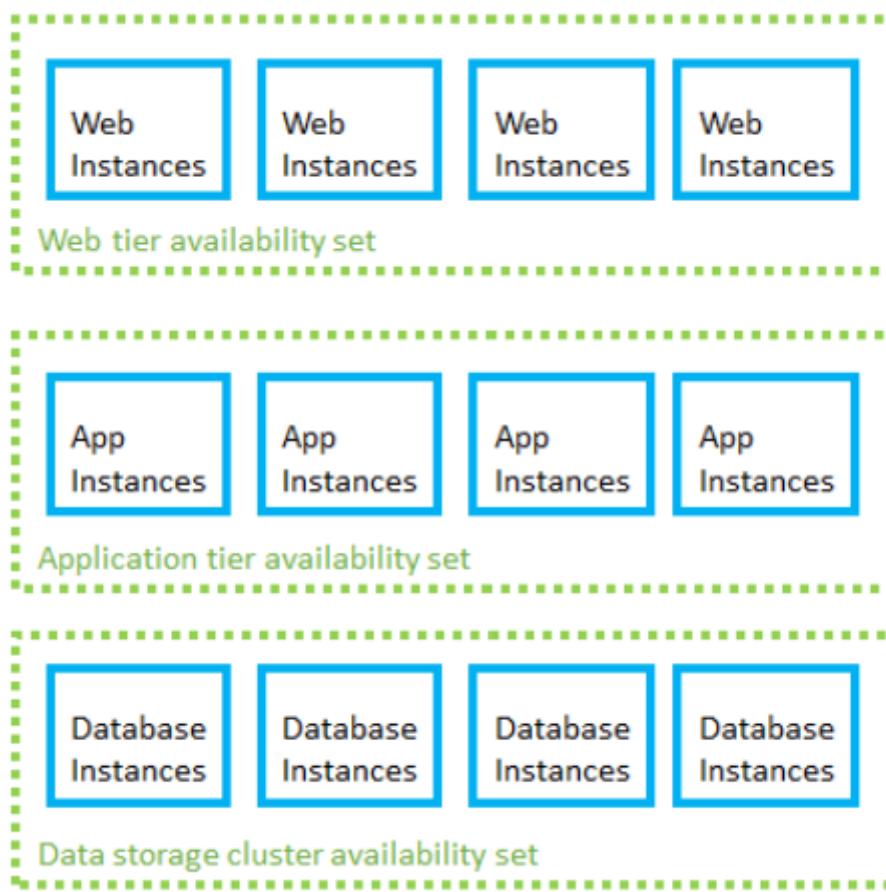
Availability sets

To protect against localized hardware failures, such as a disk or network switch failing, deploy two or more VMs in an availability set. An availability set consists of two or more *fault domains* that share a common power source and network switch. VMs in an availability set are distributed across the fault domains, so if a hardware failure affects one fault domain, network traffic can still be routed the VMs in the other fault domains.

For more information about Availability Sets, see [Manage the availability of Windows virtual machines in Azure](#).

When VM instances are added to availability sets, they are also assigned an [update domain](#). An update domain is a group of VMs that are set for planned maintenance events at the same time. Distributing VMs across multiple update domains ensures that planned update and patching events affect only a subset of these VMs at any given time.

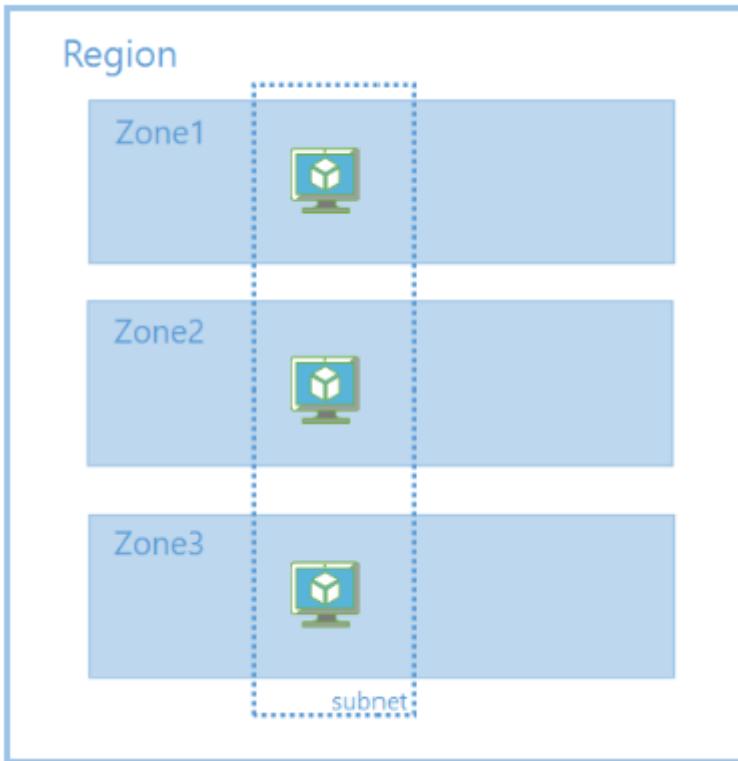
Availability sets should be organized by the instance's role in your application to ensure one instance in each role is operational. For example, in a three-tier web application, create separate availability sets for the front-end, application, and data tiers.



Availability sets

Availability Zones

Like Google Cloud, Azure regions can have Availability zones. An [Availability Zone](#) is a physically separate zone within an Azure region. Each Availability Zone has a distinct power source, network, and cooling. Deploying VMs across availability zones helps to protect an application against datacenter-wide failures.



Zone redundant VM deployment on Azure

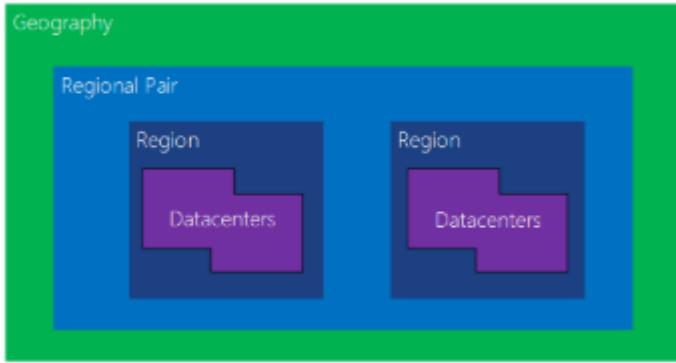
For more information, see [Build solutions for high availability using Availability Zones](#).

Paired regions

To protect an application against a regional outage, you can deploy the application across multiple regions, using [Azure Traffic Manager](#) to distribute internet traffic to the different regions. Each Azure region is paired with another region. Together, these form a [regional pair](#). With the exception of Brazil South, regional pairs are located within the same geography in order to meet data residency requirements for tax and law enforcement jurisdiction purposes.

Unlike Availability Zones, which are physically separate datacenters but may be in relatively nearby geographic areas, paired regions are typically separated by at least 300 miles. This design ensures that large-scale disasters only affect one of the regions in the pair. Neighboring pairs can be set to sync database and storage service data, and are configured so that platform updates are rolled out to only one region in the pair at a time.

Azure [geo-redundant storage](#) is automatically backed up to the appropriate paired region. For all other resources, creating a fully redundant solution using paired regions means creating a full copy of your solution in both regions.



Region Pairs in Azure

See also

- [Regions for virtual machines in Azure](#)
- [Availability options for virtual machines in Azure](#)
- [High availability for Azure applications](#)
- [Failure and disaster recovery for Azure applications](#)
- [Planned maintenance for Linux virtual machines in Azure](#)

Services

For a listing of how services map between platforms, see [Google Cloud to Azure services comparison](#).

Not all Azure products and services are available in all regions. Consult the [Products by Region](#) page for details. You can find the uptime guarantees and downtime credit policies for each Azure product or service on the [Service Level Agreements](#) page.

Next steps

- [Get started with Azure](#)
- [Azure Reference Architectures](#)

Google Cloud to Azure services comparison

Article • 03/21/2023

This article helps you understand how Microsoft Azure services compare to Google Cloud. (Note that Google Cloud used to be called the Google Cloud Platform (GCP).) Whether you are planning a multi-cloud solution with Azure and Google Cloud, or migrating to Azure, you can compare the IT capabilities of Azure and Google Cloud services in all the technology categories.

This article compares services that are roughly comparable. Not every Google Cloud service or Azure service is listed, and not every matched service has exact feature-for-feature parity.

For an overview of Azure for Google Cloud users, see the introduction to [Azure for Google Cloud Professionals](#).

Marketplace

Google Cloud service	Azure service	Description
Google Cloud Marketplace	Azure Marketplace	Easy-to-deploy and automatically configured third-party applications, including single virtual machine or multiple virtual machine solutions.

Data platform

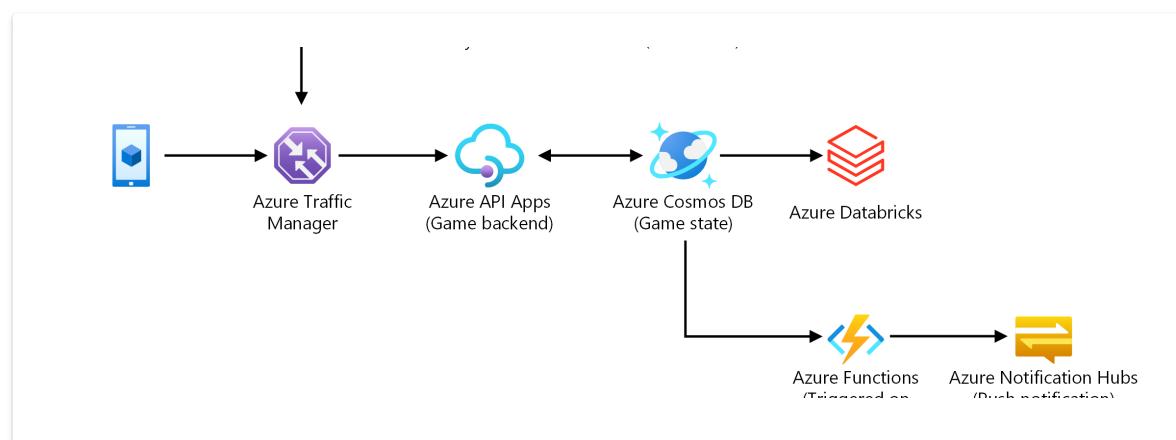
Database

Type	Google Cloud service	Azure service	Azure service description
------	----------------------	---------------	---------------------------

Type	Google Cloud service	Azure service	Azure service description
Relational database	<p>Cloud SQL - SQL Server</p> <p>Cloud SQL - MySQL & PostgreSQL</p>	<p>Azure SQL family</p> <p>Azure SQL Database</p> <p>Azure SQL Managed Instance</p> <p>SQL Server on Azure VM</p> <p>Azure SQL Edge</p>	<p>Azure SQL family of SQL Server database engine products in the cloud</p> <p>Azure SQL Database is a fully managed platform as a service (PaaS) database engine</p> <p>Azure SQL Managed Instance is the intelligent, scalable cloud database service that combines the broadest SQL Server database engine compatibility with all the benefits of a fully managed and evergreen platform as a service</p> <p>SQL Server IaaS deployed on Azure Windows or Linux VM</p> <p>Azure SQL Edge is an optimized relational database engine geared for IoT and edge deployments</p>
Horizontally scalable relational database	Cloud Spanner	<p>Azure Database for MySQL (Single & Flexible Server)</p> <p>Azure Database for PostgreSQL (Single & Flexible Server)</p>	Managed relational database service where resiliency, security, scale, and maintenance are primarily handled by the platform
	Cloud Spanner	Azure Cosmos DB for NoSQL	A globally-distributed database system that limitlessly scales horizontally. Is multi-modal -- key-value, graph, and document data). Supports multiple APIs: SQL, JavaScript, Gremlin, MongoDB, and Azure Table storage. Compute and storage can be scaled independently

Type	Google Cloud service	Azure service	Azure service description
		Azure PostgreSQL Hyperscale (Citus)	Azure Database for PostgreSQL is a fully managed database-as-a-service based on the open-source Postgres relational database engine. The Hyperscale (Citus) deployment option scales queries across multiple machines using sharding, to serve applications that require greater scale and performance
NoSQL	Cloud Bigtable ↗	Azure Table storage	A highly scalable NoSQL key-value store for rapid development using massive semi-structured datasets. Store semi-structured data that's highly available. Supporting flexible data schema and OData-based queries
	Cloud Firestore ↗	Azure Cosmos DB ↗	Globally distributed, multi-model database that natively supports multiple data models: key-value, documents, graphs, and columnar
	Firebase Realtime Database ↗	Azure Cosmos DB change feed	Change feed in Azure Cosmos DB is a persistent record of changes to a container in the order they occur. Change feed works by listening to an Azure Cosmos DB container for any changes. It then outputs the sorted list of documents that were changed in the order in which they were modified. The persisted changes can be processed asynchronously and incrementally, and the output can be distributed across one or more consumers for parallel processing
In-memory	Cloud Memorystore ↗	Azure Cache for Redis ↗	A secure data cache and messaging broker that provides high throughput and low-latency access to data for applications

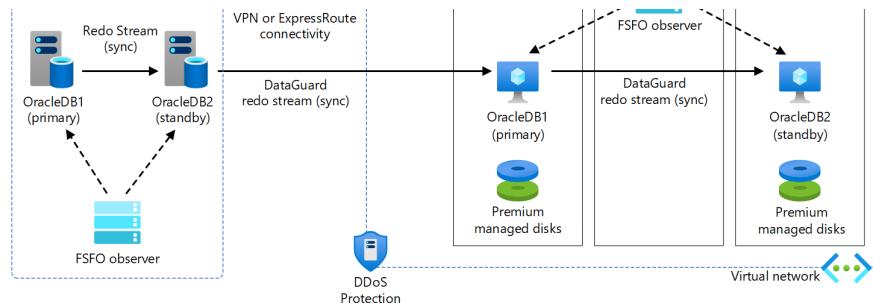
Database architectures



Gaming using Azure Cosmos DB

12/16/2019 1 min read

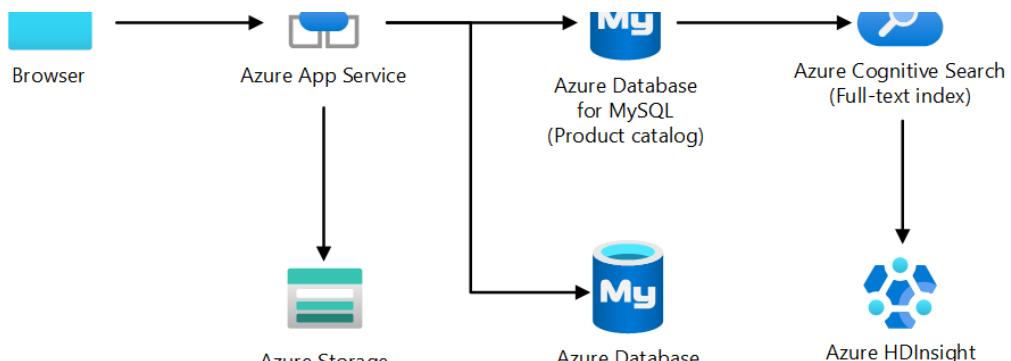
Elastically scale your database to accommodate unpredictable bursts of traffic and deliver low-latency multi-player experiences on a global scale.



Oracle Database Migration to Azure

12/16/2019 2 min read

Oracle DB migrations can be accomplished in multiple ways. This architecture covers one of these options wherein Oracle Active Data Guard is used to migrate the Database.



Retail and e-commerce using Azure MySQL

12/16/2019 1 min read

Build secure and scalable e-commerce solutions that meet the demands of both customers and business using Azure Database for MySQL.

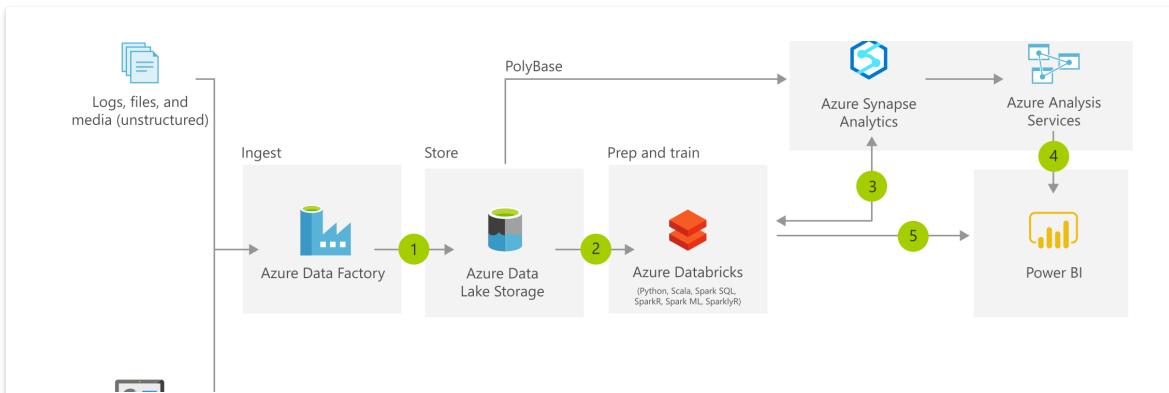
[view all](#)

Data warehouse

Google Cloud service	Azure service	Description

Google Cloud service	Azure service	Description
BigQuery ↗	Azure Synapse Analytics ↗	Cloud-based Enterprise Data Warehouse (EDW) that uses Massively Parallel Processing (MPP) to quickly run complex queries across petabytes of data.
	SQL Server	
	Big Data Clusters	Allow you to deploy scalable clusters of SQL Server, Spark, and HDFS containers running on Kubernetes. These components are running side by side to enable you to read, write, and process big data from Transact-SQL or Spark, allowing you to easily combine and analyze your high-value relational data with high-volume big data.
	Azure Databricks ↗	

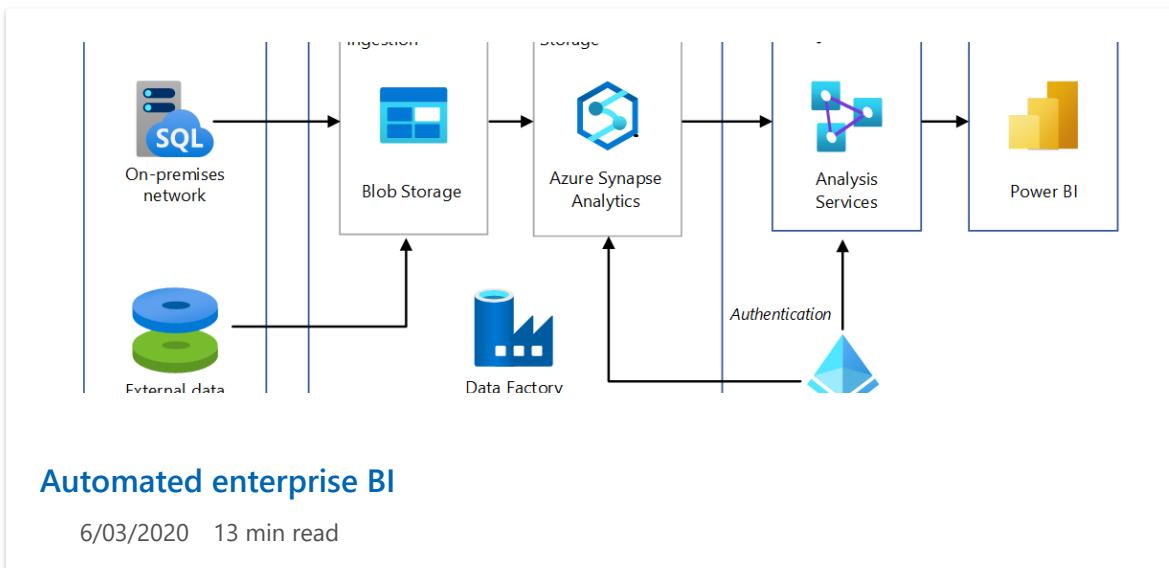
Data warehouse architectures



Modern Data Warehouse Architecture

12/16/2019 2 min read

Explore a cloud data warehouse that uses big data. Modern data warehouse brings together all your data and scales easily as your data grows.



Automate an extract, load, and transform (ELT) workflow in Azure using Azure Data Factory with Azure Synapse Analytics.

[view all](#)

Data orchestration and ETL

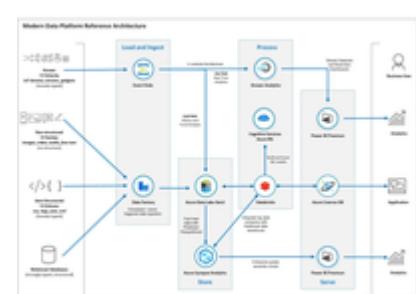
Google Cloud service	Azure service	Description
Cloud Data Fusion	Azure Data Factory Azure Synapse Analytics	Processes and moves data between different compute and storage services, as well as on-premises data sources at specified intervals. Create, schedule, orchestrate, and manage data pipelines.

Big data and analytics

Big data processing

Google Cloud service	Azure service	Description
Dataproc	Azure HDInsight Azure Synapse Analytics Azure Databricks	Managed Apache Spark-based analytics platform.

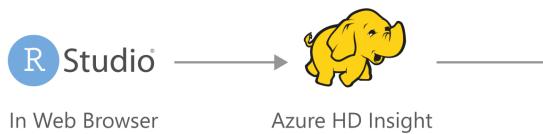
Big data architectures



Azure data platform end-to-end

1/31/2020 7 min read

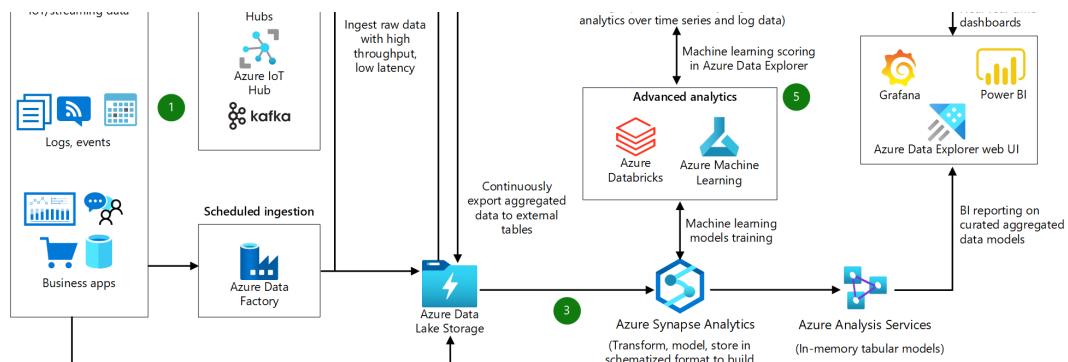
Use Azure services to ingest, process, store, serve, and visualize data from different sources.



Campaign Optimization with Azure HDInsight Spark Clusters

12/16/2019 4 min read

This solution demonstrates how to build and deploy a machine learning model with Microsoft R Server on Azure HDInsight Spark clusters to recommend actions to maximize the purchase rate of leads targeted by a campaign. This solution enables efficient handling of big data on Spark with Microsoft R Server.



Big data analytics with Azure Data Explorer

8/11/2020 3 min read

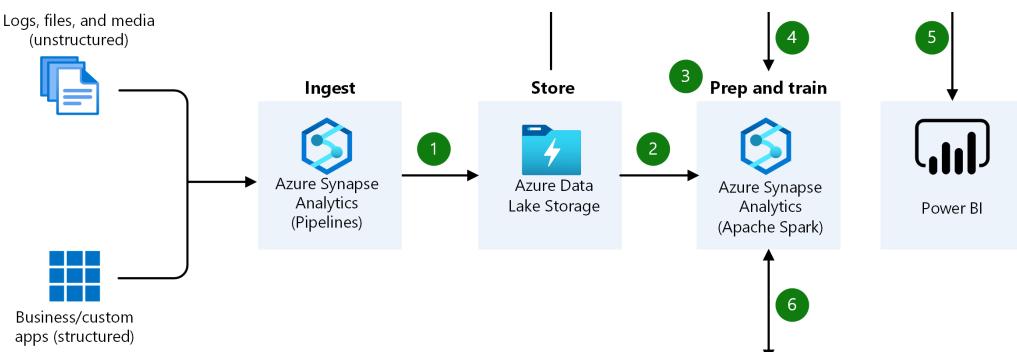
Big Data Analytics with Azure Data Explorer demonstrates Azure Data Explorer's abilities to cater to volume, velocity, and variety of data, the three V's of big data.

[view all](#)

Analytics and visualization

Google Cloud service	Azure service	Description
Cloud Dataflow ↗	Azure Databricks ↗	Managed platform for streaming batch data based on Open Source Apache products.
Data Studio ↗	Power BI ↗	Business intelligence tools that build visualizations, perform ad hoc analysis, and develop business insights from data.
Looker ↗		
Cloud Search ↗	Azure Search ↗	Delivers full-text search and related search analytics and capabilities.
BigQuery ↗	SQL Server Analysis Services	Provides a serverless non-cloud interactive query service that uses standard SQL for analyzing databases.

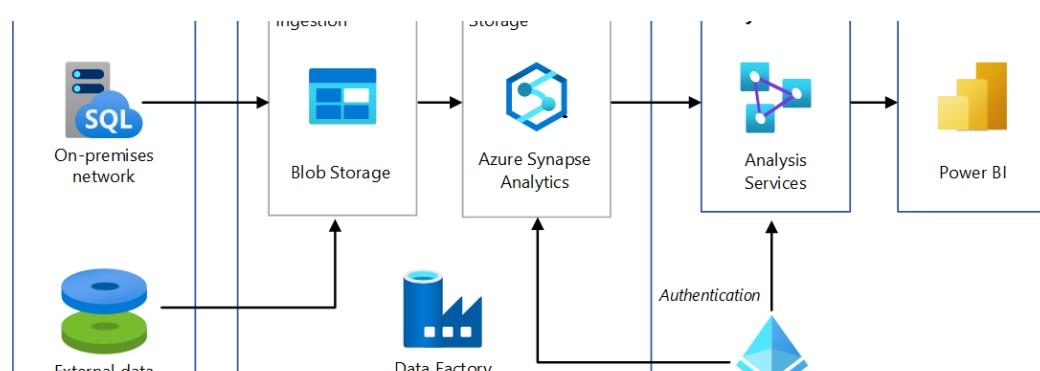
Analytics architectures



Advanced Analytics Architecture

12/16/2019 2 min read

Get near real-time data analytics on streaming services. This big data architecture allows you to combine any data at any scale with custom machine learning.



Automated enterprise BI

6/03/2020 13 min read

Automate an extract, load, and transform (ELT) workflow in Azure using Azure Data Factory with Azure Synapse Analytics.



Mass ingestion and analysis of news feeds on Azure

2/01/2019 5 min read

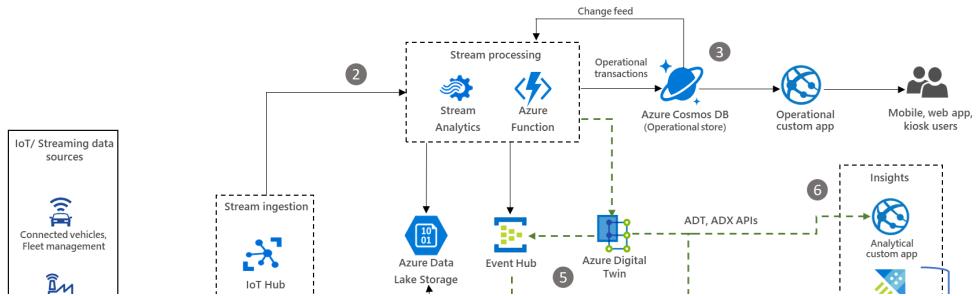
Create a pipeline for ingesting and analyzing text, images, sentiment, and other data from RSS news feeds using only Azure services, including Azure Cosmos DB and Azure Cognitive Services.

[view all](#)

Time series & IOT data

Google Cloud service	Azure service	Description
BigQuery ↗	Azure Data Explorer ↗	Fully managed, low latency, and distributed big data analytics platform that runs complex queries across petabytes of data. Highly optimized for log and time series data.
	Azure Time Series Insights ↗	Open and scalable end-to-end IoT analytics service. Collect, process, store, query, and visualize data at Internet of Things (IoT) scale--data that's highly contextualized and optimized for time series.
	Azure Cosmos DB	

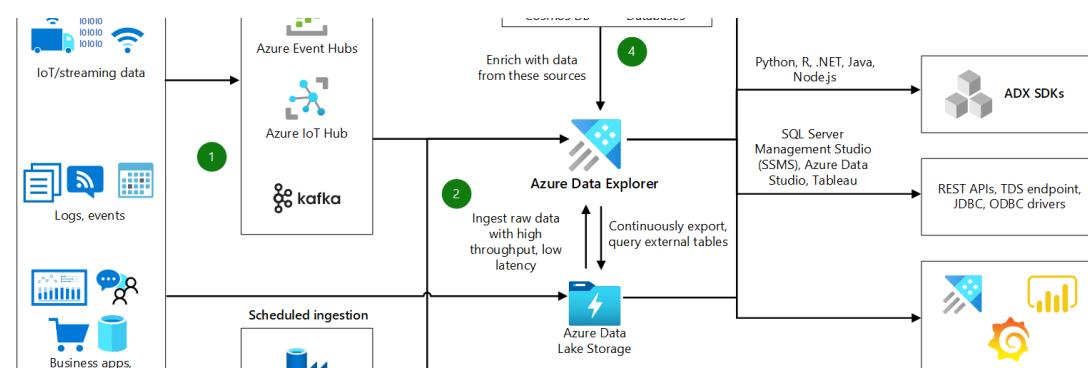
Time series architectures



IoT analytics with Azure Data Explorer

8/11/2020 3 min read

IoT Telemetry Analytics with Azure Data Explorer demonstrates near real-time analytics over fast flowing, high volume, wide variety of streaming data from IoT devices.



Azure Data Explorer interactive analytics

8/11/2020 3 min read

Interactive Analytics with Azure Data Explorer focuses on its integration with the rest of the data platform ecosystem.

AI and machine learning

Google Cloud service	Azure service	Description
Vertex AI	Azure Machine Learning	A cloud service to train, deploy, automate, and manage machine learning models.
TensorFlow	ML.NET	ML.NET is an open source and cross-platform machine learning framework for both machine learning & AI.

Google Cloud service	Azure service	Description
TensorFlow ↗	ONNX (Open Neural Network Exchange) ↗	ONNX is an open format built to represent machine learning models that facilitates maximum compatibility and increased inference performance.
Vision AI ↗	Azure Cognitive Services Computer Vision ↗	Use visual data processing to label content, from objects to concepts, extract printed and handwritten text, recognize familiar subjects like brands and landmarks, and moderate content. No machine learning expertise is required.
Natural Language AI ↗	Azure Cognitive Services Text Analytics ↗	Cloud-based services that provides advanced natural language processing over raw text, and includes four main functions: sentiment analysis, key phrase extraction, language detection, and named entity recognition.
Natural Language AI ↗	Azure Cognitive Services Language Understanding (LUIS) ↗	A machine learning-based service to build natural language understanding into apps, bots, and IoT devices. Quickly create enterprise-ready, custom models that continuously improve.
Speech-to-Text ↗	Azure Cognitive Services Speech To Text ↗	Swiftly convert audio into text from a variety of sources. Customize models to overcome common speech recognition barriers, such as unique vocabularies, speaking styles, or background noise.
AutoML Tables – Structured Data ↗	Azure ML - Automated Machine Learning ↗	Empower professional and non-professional data scientists to build machine learning models rapidly. Automate time-consuming and iterative tasks of model development using breakthrough research-and accelerate time to market. Available in Azure Machine learning, Power BI, ML.NET & Visual Studio.
AutoML Tables – Structured Data ↗	ML.NET Model Builder ↗	ML.NET Model Builder provides an easy to understand visual interface to build, train, and deploy custom machine learning models. Prior machine learning expertise is not required. Model Builder supports AutoML, which automatically explores different machine learning algorithms and settings to help you find the one that best suits your scenario.

Google Cloud service	Azure service	Description
AutoML Vision ↗	Azure Cognitive Services Custom Vision ↗	Customize and embed state-of-the-art computer vision for specific domains. Build frictionless customer experiences, optimize manufacturing processes, accelerate digital marketing campaigns-and more. No machine learning expertise is required.
AutoML Video Intelligence ↗	Azure Video Analyzer ↗	Easily extract insights from your videos and quickly enrich your applications to enhance discovery and engagement.
Dialogflow ↗	Azure Cognitive Services QnA Maker ↗	Build, train and publish a sophisticated bot using FAQ pages, support websites, product manuals, SharePoint documents or editorial content through an easy-to-use UI or via REST APIs.
AI Platform Notebooks ↗	Azure Notebooks ↗	Develop and run code from anywhere with Jupyter notebooks on Azure.
Deep Learning VM Image ↗	Data Science Virtual Machines ↗	Pre-Configured environments in the cloud for Data Science and AI Development.
Deep Learning Containers ↗	GPU support on Azure Kubernetes Service (AKS)	Graphical processing units (GPUs) are often used for compute-intensive workloads such as graphics and visualization workloads. AKS supports the creation of GPU-enabled node pools to run these compute-intensive workloads in Kubernetes.
Data Labeling Service ↗	Azure ML - Data Labeling	A central place to create, manage, and monitor labeling projects (public preview). Use it to coordinate data, labels, and team members to efficiently manage labeling tasks. Machine Learning supports image classification, either multi-label or multi-class, and object identification with bounded boxes.
AI Platform Training ↗	Azure ML – Compute Targets	Designated compute resource/environment where you run your training script or host your service deployment. This location may be your local machine or a cloud-based compute resource. Using compute targets make it easy for you to later change your compute environment without having to change your code.
AI Platform Predictions ↗	Azure ML - Deployments	Deploy your machine learning model as a web service in the Azure cloud or to Azure IoT Edge devices. Leverage serverless Azure Functions for model inference for dynamic scale.

Google Cloud service	Azure service	Description
Continuous Evaluation ↗	Azure ML – Data Drift	Monitor for data drift between the training dataset and inference data of a deployed model. In the context of machine learning, trained machine learning models may experience degraded prediction performance because of drift. With Azure Machine Learning, you can monitor data drift and the service can send an email alert to you when drift is detected.
What-If Tool ↗	Azure ML – Model Interpretability	Ensure machine learning model compliance with company policies, industry standards, and government regulations.
Cloud TPU ↗	Azure ML – FPGA (Field Programmable Gate Arrays)	FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects. The interconnects allow these blocks to be configured in various ways after manufacturing. Compared to other chips, FPGAs provide a combination of programmability and performance.
Kubeflow ↗	Machine Learning Operations (MLOps) ↗	MLOps, or DevOps for machine learning, enables data science and IT teams to collaborate and increase the pace of model development and deployment via monitoring, validation, and governance of machine learning models.
Dialogflow ↗	Microsoft Bot Framework ↗	Build and connect intelligent bots that interact with your users using text/SMS, Skype, Teams, Slack, Microsoft 365 mail, Twitter, and other popular services.

AI and machine learning architectures

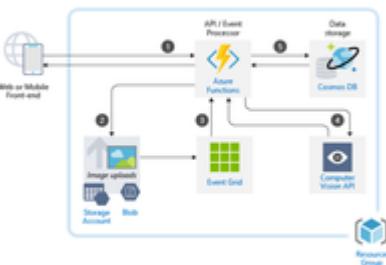
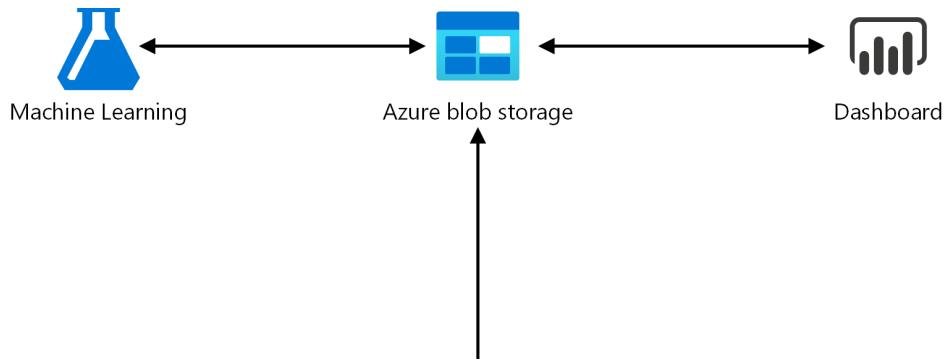


Image classification on Azure

7/05/2018 4 min read

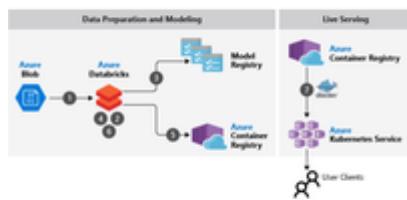
Learn how to build image processing into your applications by using Azure services such as the Computer Vision API and Azure Functions.



Predictive Marketing with Machine Learning

12/16/2019 2 min read

Learn how to build a machine-learning model with Microsoft R Server on Azure HDInsight Spark clusters to recommend actions to maximize the purchase rate.



Scalable personalization on Azure

5/31/2019 6 min read

Use machine learning to automate content-based personalization for customers.

[view all](#)

Data catalog & governance

Google Cloud service	Azure service	Description
Cloud Data Catalog	Azure Purview	Azure Purview is a unified data governance service that helps you manage and govern your on-premises, multi-cloud, and software-as-a-service (SaaS) data.

Compute

Virtual servers

Google Cloud service	Azure service	Description
Compute Engine ↗	Azure Virtual Machines ↗	Virtual servers allow users to deploy, manage, and maintain OS and server software. Instance types provide combinations of CPU/RAM. Users pay for what they use with the flexibility to change sizes.
Sole-tenant nodes ↗	Azure Dedicated Host ↗	Host your VMs on hardware that's dedicated only to your project.
Batch ↗	Azure Batch ↗	Run large-scale parallel and high-performance computing applications efficiently in the cloud.
Compute Engine Autoscaler ↗	Azure virtual machine scale sets	Allows you to automatically change the number of VM instances. You set defined metric and thresholds that determine if the platform adds or removes instances.
Compute Engine managed instance groups ↗		
Cloud GPUs ↗	GPU Optimized VMs	GPU-optimized VM sizes are specialized virtual machines that are available with single, multiple, or fractional GPUs. The sizes are designed for compute-intensive, graphics-intensive, and visualization workloads.
VMware Engine ↗	Azure VMware Solution ↗	Redeploy and extend your VMware-based enterprise workloads to Azure with Azure VMware Solution. Seamlessly move VMware-based workloads from your datacenter to Azure and integrate your VMware environment with Azure. Keep managing your existing environments with the same VMware tools that you already know, while you modernize your applications with Azure native services. Azure VMware Solution is a Microsoft service that is verified by VMware, and it runs on Azure infrastructure.

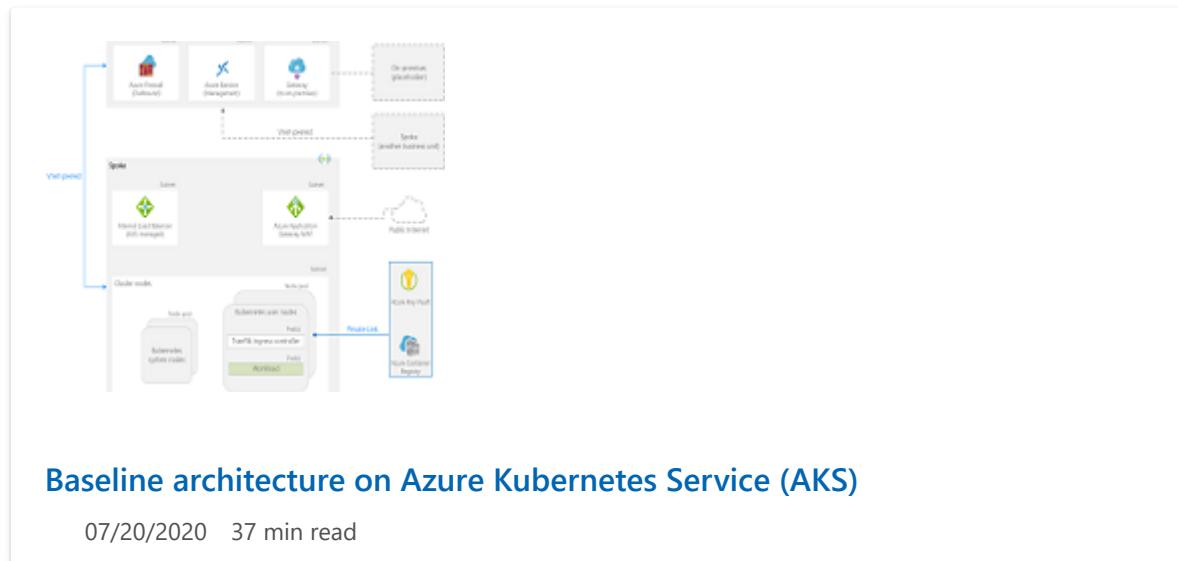
Containers and container orchestrators

Google Cloud service	Azure service	Description
----------------------	---------------	-------------

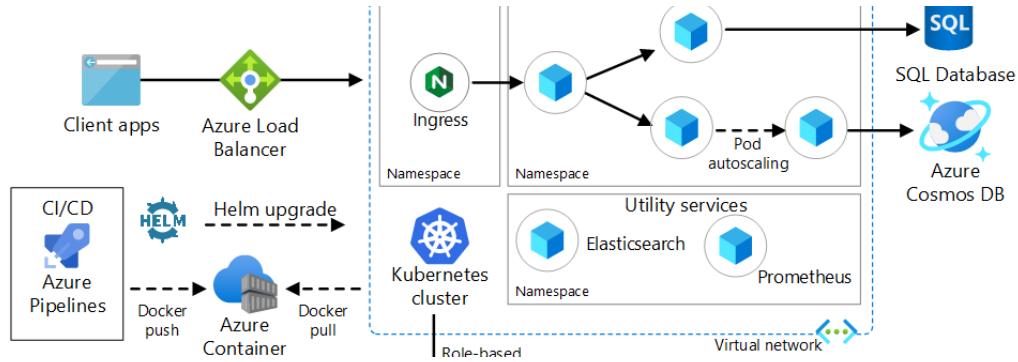
Google Cloud service	Azure service	Description
Cloud Run ↗	Azure Container Apps ↗	Azure Container Apps is the fastest and simplest way to run a container in Azure, without having to provision any virtual machines or adopt a higher-level orchestration service.
Artifact Registry (beta) ↗	Azure Container Registry ↗	Allows customers to store Docker formatted images. Used to create all types of container deployments on Azure.
Container Registry ↗		
Kubernetes Engine (GKE) ↗	Azure Kubernetes Service (AKS) ↗	Deploy orchestrated containerized applications with Kubernetes. Simplify cluster management and monitoring through automatic upgrades and a built-in operations console. See AKS solution journey .
Kubernetes Engine Monitoring ↗	Azure Monitor container insights	Azure Monitor container insights is a feature designed to monitor the performance of container workloads deployed to: Managed Kubernetes clusters hosted on Azure Kubernetes Service (AKS); Self-managed Kubernetes clusters hosted on Azure using AKS Engine ↗ ; Azure Container Instances, Self-managed Kubernetes clusters hosted on Azure Stack or on-premises; or Azure Red Hat OpenShift .
Anthos Service Mesh ↗	Open Service Mesh (OSM) ↗	It is a lightweight and extensible cloud native service mesh. OSM takes a simple approach for users to uniformly manage, secure, and get out-of-the box observability features for highly dynamic microservice environments

Container architectures

Here are some architectures that use AKS as the orchestrator.



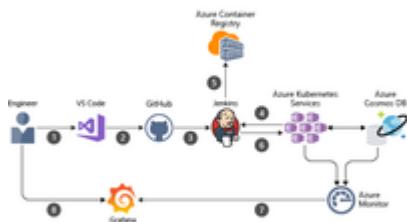
Deploy a baseline infrastructure that deploys an AKS cluster with focus on security.



Microservices architecture on Azure Kubernetes Service (AKS)

5/07/2020 17 min read

Deploy a microservices architecture on Azure Kubernetes Service (AKS)



CI/CD pipeline for container-based workloads

7/05/2018 7 min read

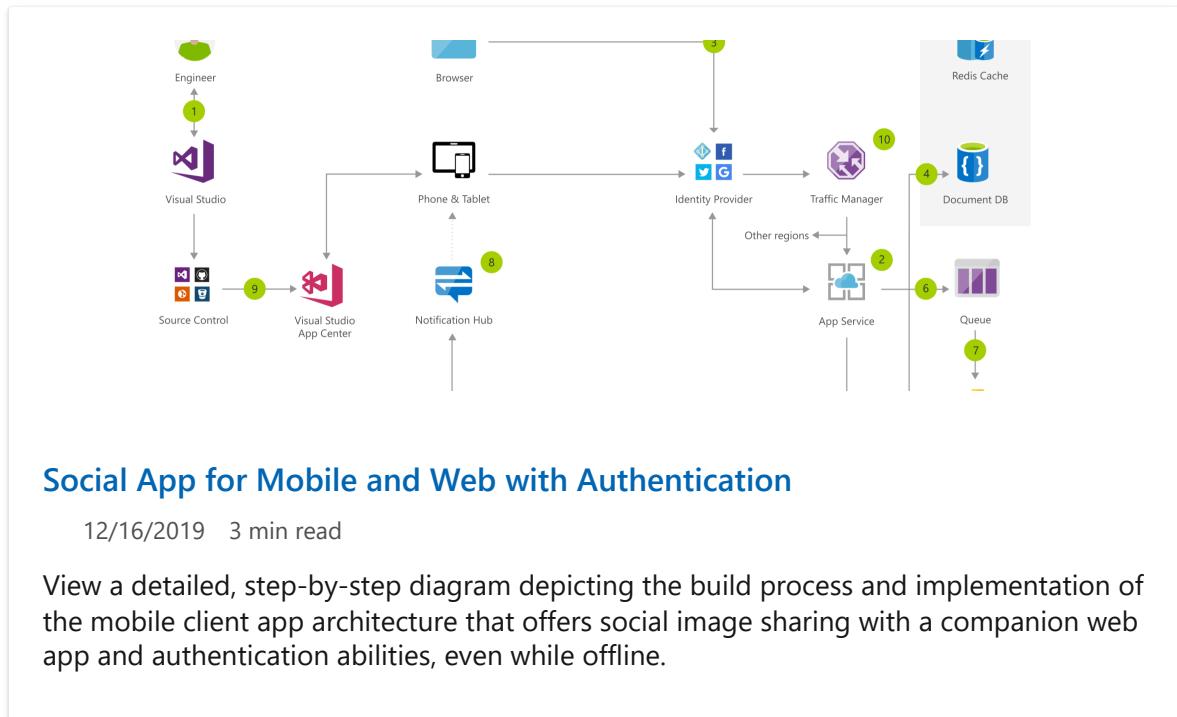
Build a DevOps pipeline for a Node.js web app with Jenkins, Azure Container Registry, Azure Kubernetes Service, Azure Cosmos DB, and Grafana.

[view all](#)

Functions

Google Cloud service	Azure service	Description
Cloud Functions	Azure Functions	Integrate systems and run backend processes in response to events or schedules without provisioning or managing servers.

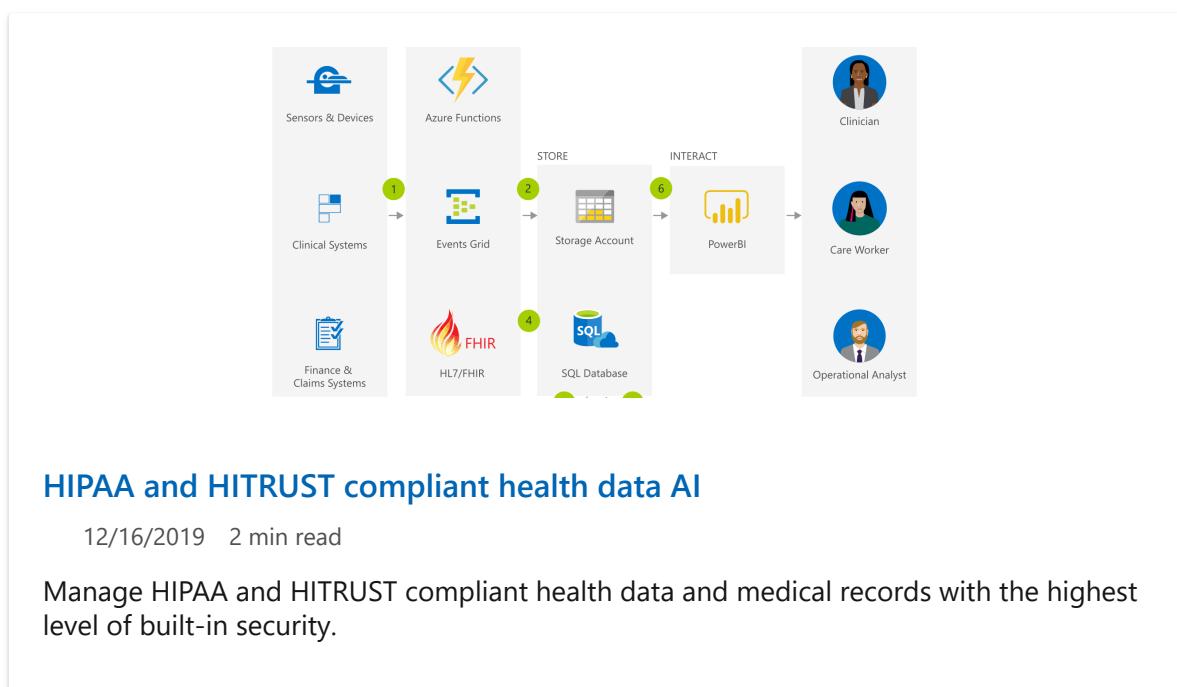
Serverless architectures



Social App for Mobile and Web with Authentication

12/16/2019 3 min read

View a detailed, step-by-step diagram depicting the build process and implementation of the mobile client app architecture that offers social image sharing with a companion web app and authentication abilities, even while offline.



HIPAA and HITRUST compliant health data AI

12/16/2019 2 min read

Manage HIPAA and HITRUST compliant health data and medical records with the highest level of built-in security.



Cross Cloud Scaling Architecture

12/16/2019 1 min read

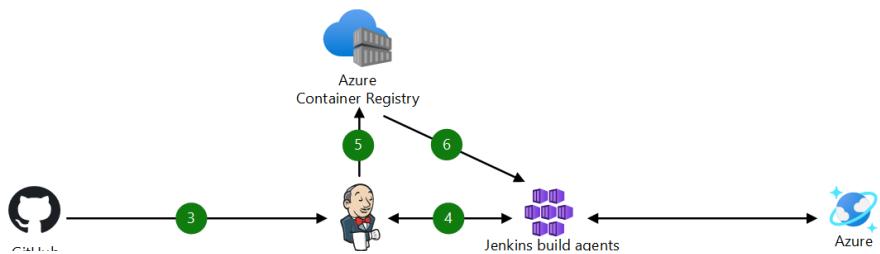
Learn how to improve cross cloud scalability with solution architecture that includes Azure Stack. A step-by-step flowchart details instructions for implementation.

DevOps and application monitoring

Google Cloud service	Azure service	Description
Operations (formerly Stackdriver) 	Azure Monitor 	Maximizes the availability and performance of your applications and services by delivering a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources on which they depend.
Cloud Trace 	Azure Monitor 	Maximizes the availability and performance of your applications and services by delivering a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources on which they depend.
Cloud Debugger 	Azure Monitor 	Maximizes the availability and performance of your applications and services by delivering a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources on which they depend.
Cloud Profiler 	Azure Monitor 	Maximizes the availability and performance of your applications and services by delivering a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. It helps you understand how your applications are performing and proactively identifies issues affecting them and the resources on which they depend.
Cloud Source Repositories 	Azure Repos  , GitHub Repos 	A cloud service for collaborating on code development.
Cloud Build 	Azure Pipelines  , GitHub Actions 	Fully managed build service that supports continuous integration and deployment.

Google Cloud service	Azure service	Description
Artifact Registry ↗	Azure Artifacts ↗, GitHub Packages ↗	Add fully integrated package management to your continuous integration/continuous delivery (CI/CD) pipelines with a single click. Create and share Maven, npm, NuGet, and Python package feeds from public and private sources with teams of any size.
Cloud Developer Tools ↗ (including Cloud Code)	Azure Developer Tools ↗	Collection of tools for building, debugging, deploying, diagnosing, and managing multiplatform scalable apps and services.
Gcloud SDK ↗	Azure CLI	The Azure command-line interface (Azure CLI) is a set of commands used to create and manage Azure resources. The Azure CLI is available across Azure services and is designed to get you working quickly with Azure, with an emphasis on automation.
Cloud Shell ↗	Azure Cloud Shell	Azure Cloud Shell is an interactive, authenticated, browser-accessible shell for managing Azure resources. It provides the flexibility of choosing the shell experience that best suits the way you work, either Bash or PowerShell.
PowerShell on Google Cloud ↗	Azure PowerShell	Azure PowerShell is a set of cmdlets for managing Azure resources directly from the PowerShell command line. Azure PowerShell is designed to make it easy to learn and get started with, but provides powerful features for automation. Written in .NET Standard, Azure PowerShell works with PowerShell 5.1 on Windows, and PowerShell 6.x and higher on all platforms.
Cloud Deployment Manager ↗	Azure Automation ↗	Delivers a cloud-based automation and configuration service that supports consistent management across your Azure and non-Azure environments. It comprises process automation, configuration management, update management, shared capabilities, and heterogeneous features. Automation gives you complete control during deployment, operations, and decommissioning of workloads and resources.
Cloud Deployment Manager ↗	Azure Resource Manager ↗	Provides a way for users to automate the manual, long-running, error-prone, and frequently repeated IT tasks.

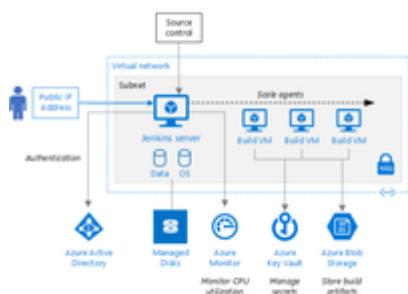
DevOps architectures



Container CI/CD using Jenkins and Kubernetes on Azure Kubernetes Service (AKS)

12/16/2019 2 min read

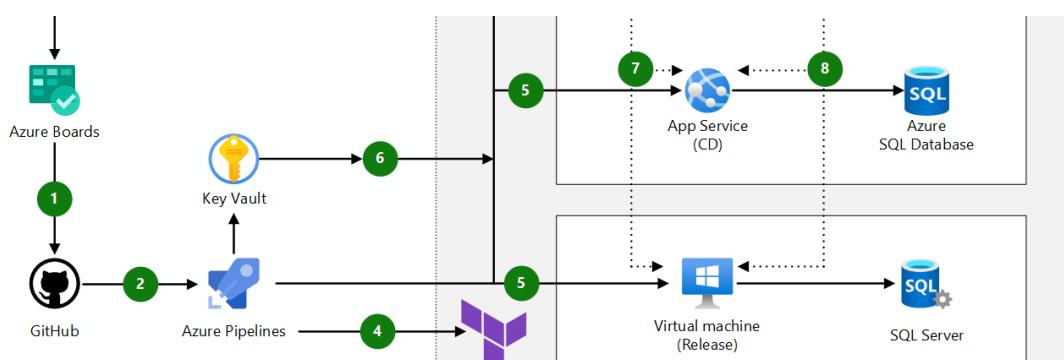
Containers make it easy for you to continuously build and deploy applications. By orchestrating the deployment of those containers using Azure Kubernetes Service (AKS), you can achieve replicable, manageable clusters of containers.



Run a Jenkins server on Azure

11/19/2020 6 min read

Recommended architecture that shows how to deploy and operate a scalable, enterprise-grade Jenkins server on Azure secured with single sign-on (SSO).



DevOps in a hybrid environment

12/16/2019 3 min read

The tools provided in Azure allow for the implementation of a DevOps strategy that capably manages both cloud and on-premises environments in tandem.

[view all](#)

Internet of things (IoT)

Google Cloud service	Azure service	Description
Cloud IoT Core ↗	Azure IoT Hub ↗, Azure Event Hubs ↗	A cloud gateway for managing bidirectional communication with billions of IoT devices, securely and at scale.
Cloud Pub/Sub ↗	Azure Stream Analytics ↗, HDInsight Kafka	Process and route streaming data to a subsequent processing engine or to a storage or database platform.
Edge TPU ↗	Azure IoT Edge ↗	Deploy cloud intelligence directly on IoT devices to run in on-premises scenarios.

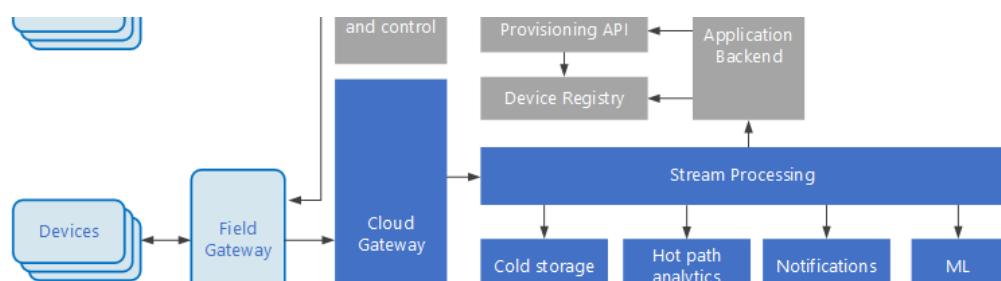
IoT architectures



IoT Architecture ↗ Azure IoT Subsystems

12/16/2019 1 min read

Learn about our recommended IoT application architecture that supports hybrid cloud and edge computing. A flowchart details how the subsystems function within the IoT application.





Azure IoT reference architecture

9/10/2020 12 min read

Recommended architecture for IoT applications on Azure using PaaS (platform-as-a-service) components



Process real-time vehicle data using IoT

11/17/2020 5 min read

This example builds a real-time data ingestion/processing pipeline to ingest and process messages from IoT devices into a big data analytic platform in Azure.

[view all](#)

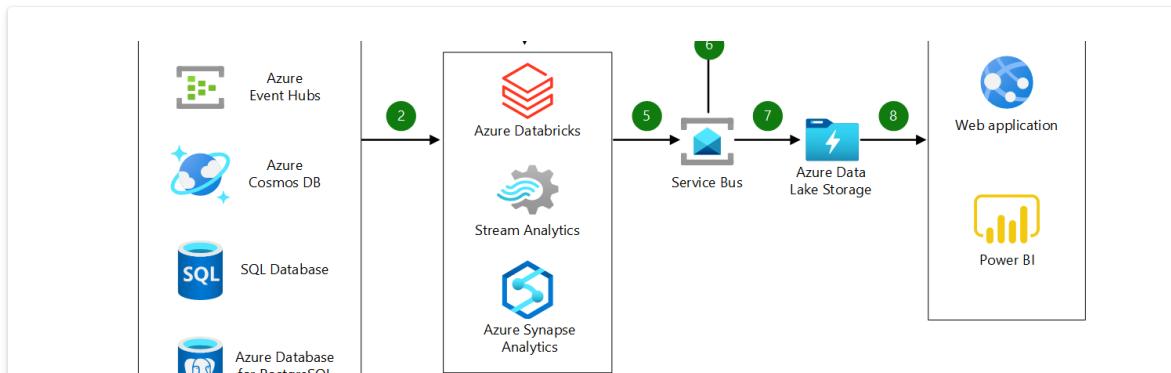
Management

Google Cloud service	Azure service	Description
Cloud Billing	Azure Billing API	Services to help generate, monitor, forecast, and share billing data for resource usage by time, organization, or product resources.
Cloud Console	Azure portal	A unified management console that simplifies building, deploying, and operating your cloud resources.
Operations (formerly Stackdriver)	Azure Monitor	Comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments.
Cost Management	Azure Cost Management	Azure Cost Management helps you understand your Azure invoice, manage your billing account and subscriptions, control Azure spending, and optimize resource use.

Messaging and eventing

Google Cloud service	Azure service	Description
Cloud Pub/Sub	Azure Service Bus	Supports a set of cloud-based, message-oriented middleware technologies including reliable message queuing and durable publish/subscribe messaging.
Cloud Pub/Sub	Azure Event Grid	A fully managed event routing service that allows for uniform event consumption using a publish/subscribe model.
Cloud Pub/Sub	Azure Event Hubs	A real-time data ingestion and microbatching service used to build dynamic data pipelines and integrates with other Azure services.

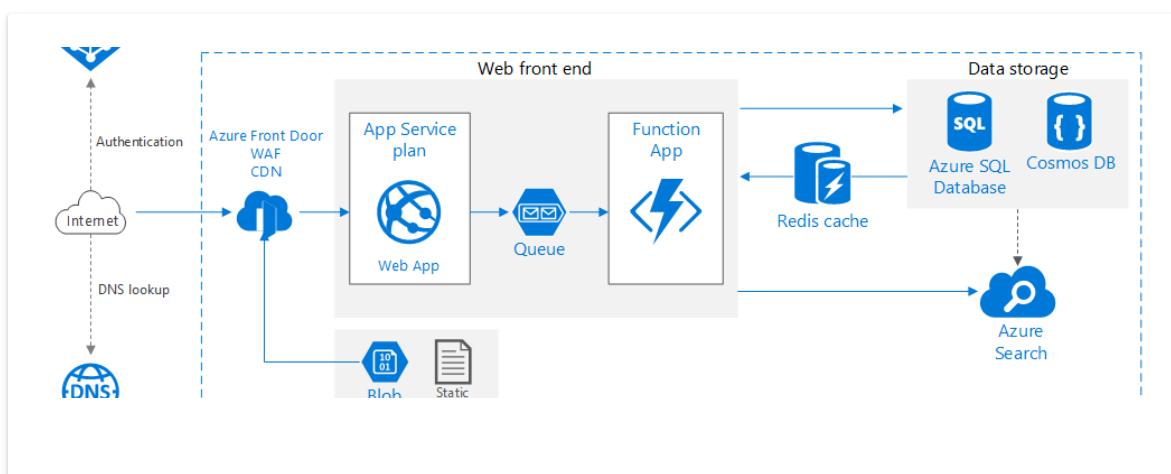
Messaging architectures



Anomaly Detector Process

12/16/2019 1 min read

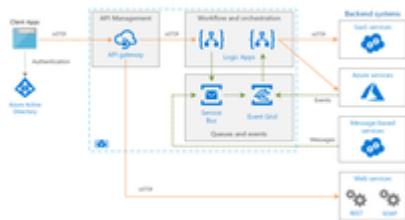
Learn more about Anomaly Detector with a step-by-step flowchart that details the process. See how anomaly detection models are selected with time-series data.



Scalable web application

10/03/2019 7 min read

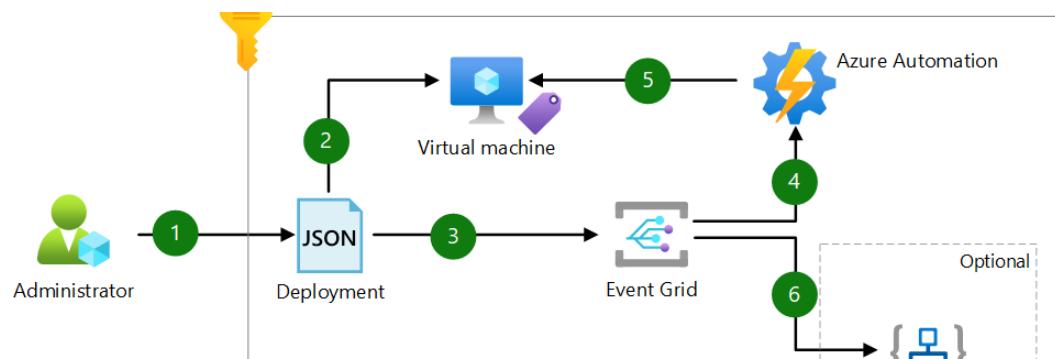
Use the proven practices in this reference architecture to improve scalability and performance in an Azure App Service web application..



Enterprise integration using queues and events

12/03/2018 5 min read

Recommended architecture for implementing an enterprise integration pattern with Azure Logic Apps, Azure API Management, Azure Service Bus, and Azure Event Grid.



Ops automation using Event Grid

12/16/2019 1 min read

Event Grid allows you to speed automation and simplify policy enforcement. For example, Event Grid can notify Azure Automation when a virtual machine is created, or a SQL Database is spun up. These events can be used to automatically check that service configurations are compliant, put metadata into operations tools, tag virtual machines, ...

Networking

Area	Google Cloud service	Azure service	Description
------	----------------------	---------------	-------------

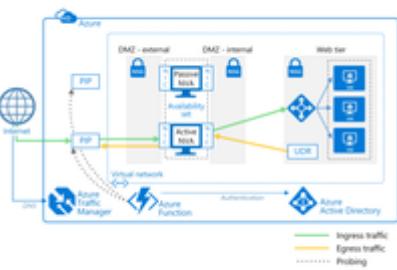
Area	Google Cloud service	Azure service	Description
Cloud virtual networking	Virtual Private Network (VPC) ↗	Azure Virtual Network (Vnet)	Provides an isolated, private environment in the cloud. Users have control over their virtual networking environment, including selection of their own IP address range, adding/updating address ranges, creation of subnets, and configuration of route tables and network gateways.
DNS management	Cloud DNS ↗	Azure DNS	Manage your DNS records using the same credentials that are used for billing and support contract as your other Azure services
	Cloud DNS ↗	Azure Traffic Manager	Azure Traffic Manager is a DNS-based load balancer that enables you to distribute traffic optimally to services across global Azure regions, while providing high availability and responsiveness.
	Internal DNS ↗	Azure Private DNS	Manages and resolves domain names in the virtual network, without the need to configure a custom DNS solution, and it provides a naming resolution for virtual machines (VMs) within a virtual network and any connected virtual networks.
Hybrid Connectivity	Cloud Interconnect ↗	Azure ExpressRoute	Establishes a private network connection from a location to the cloud provider (not over the Internet).
	Cloud VPN Gateway ↗	Azure Virtual Network Gateway	Connects Azure virtual networks to other Azure virtual networks, or customer on-premises networks (site-to-site). Allows end users to connect to Azure services through VPN tunneling (point-to-site).
	Cloud VPN Gateway ↗	Azure Virtual WAN	Azure virtual WAN simplifies large-scale branch connectivity with VPN and ExpressRoute.
	Cloud router ↗	Azure Virtual Network Gateway	Enables dynamic routes exchange using BGP.

Area	Google Cloud service	Azure service	Description
Load balancing	Network Load Balancing ↗	Azure Load Balancer	Azure Load Balancer load-balances traffic at layer 4 (all TCP or UDP).
	Global load balancing ↗	Azure Front door	Azure front door enables global load balancing across regions using a single anycast IP.
	Global load balancing ↗	Azure Application Gateway	Application Gateway is a layer 7 load balancer. It takes backends with any IP that is reachable. It supports SSL termination, cookie-based session affinity, and round robin for load-balancing traffic.
	Global load balancing ↗	Azure Traffic Manager	Azure Traffic Manager is a DNS-based load balancer that enables you to distribute traffic optimally to services across global Azure regions, while providing high availability and responsiveness.
Content delivery network	Cloud CDN ↗	Azure CDN	A content delivery network (CDN) is a distributed network of servers that can efficiently deliver web content to users.
Firewall	Firewall rules ↗	Application security groups	Azure Application security groups allow you to group virtual machines and define network security policies based on those groups.
	Firewall rules ↗	Network Security groups	Azure network security group filters network traffic to and from Azure resources in an Azure virtual network.
	Firewall rules ↗	Azure Firewall	Azure Firewall is a managed, cloud-based network security service that protects your Azure Virtual Network resources. It's a fully stateful firewall as a service with built-in high availability and unrestricted cloud scalability.
Web Application Firewall	Cloud Armor ↗	Application Gateway - Web Application Firewall	Azure Web Application Firewall (WAF) provides centralized protection of your web applications from common exploits and vulnerabilities.

Area	Google Cloud service	Azure service	Description
	Cloud Armor ↗	Front door – Azure Web Application Firewall	Azure Web Application Firewall (WAF) on Azure Front Door provides centralized protection for your web applications.
	Cloud Armor ↗	CDN – Azure Web Application Firewall	Azure Web Application Firewall (WAF) on Azure Content Delivery Network (CDN) from Microsoft provides centralized protection for your web content.
NAT Gateway	Cloud NAT ↗	Azure NAT Gateway	NAT Gateway (network address translation) provides outbound NAT translations for internet connectivity for virtual networks.
Private Connectivity to PaaS	Private Service Connect ↗	Azure Private Link	Azure Private Link enables you to access Azure PaaS Services and Azure hosted customer-owned/partner services over a private endpoint in your virtual network.
Telemetry	VPC Flow logs ↗	NSG Flow logs	Network security group (NSG) flow logs are a feature of Network Watcher that allows you to view information about ingress and egress IP traffic through an NSG.
	Firewall Rules Logging ↗	NSG Flow logs	Network security group (NSG) flow logs are a feature of Network Watcher that allows you to view information about ingress and egress IP traffic through an NSG.
	Operations (formerly Stackdriver) ↗	Azure Monitor	Azure Monitor delivers a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments. Log queries help you maximize the value of the data collected in Azure Monitor Logs.
	Network Intelligence Center ↗	Azure Network Watcher	Azure Network Watcher provides tools to monitor, diagnose, view metrics, and enable or disable logs for resources in an Azure virtual network.

Area	Google Cloud service	Azure service	Description
Other Connectivity Options	S2S,P2S	Direct Interconnect <small>↗</small> , Partner Interconnect <small>↗</small> , Carrier Peering <small>↗</small>	Point to Site lets you create a secure connection to your virtual network from an individual client computer. Site to Site is a connection between two or more networks, such as a corporate network and a branch office network.

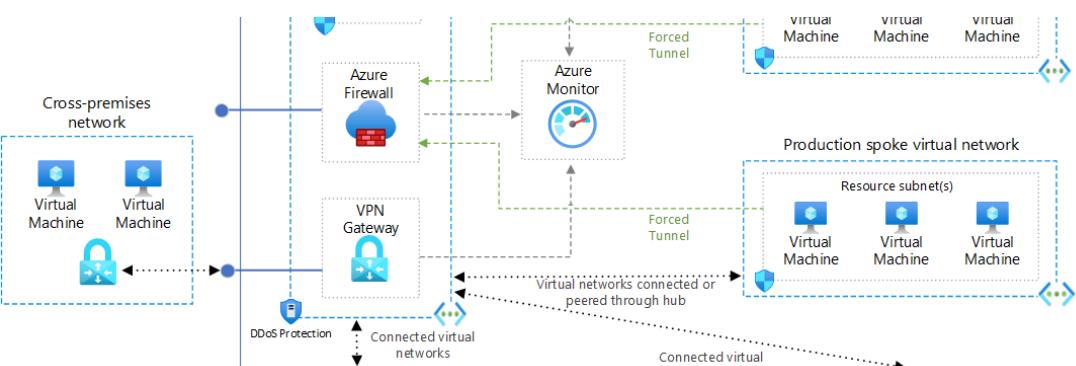
Networking architectures



Deploy highly available NVAs

12/08/2018 7 min read

Learn how to deploy network virtual appliances for high availability in Azure. This article includes example architectures for ingress, egress, and both.



Hub-spoke network topology in Azure

9/30/2020 7 min read

Learn how to implement a hub-spoke topology in Azure, where the hub is a virtual network and the spokes are virtual networks that peer with the hub.



Implement a secure hybrid network

1/07/2020 9 min read

See a secure hybrid network that extends an on-premises network to Azure with a perimeter network between the on-premises network and an Azure virtual network.

[view all](#)

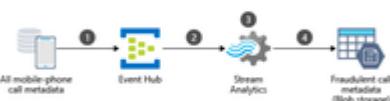
Security and identity

Area	Google Cloud service	Azure service	Description
Authentication and authorization	Cloud Identity ↗ Identity platform ↗	Azure Active Directory ↗ Azure Active Directory B2C ↗	The Azure Active Directory (Azure AD) enterprise identity service provides single sign-on and multi-factor authentication, which enable the central management of users/groups and external identities federation.
Multi-factor Authentication	Multi-factor Authentication ↗	Azure Active Directory Multi-factor Authentication ↗	A highly available and global identity management service for consumer-facing applications, which scales to hundreds of millions of identities. Manage customer, consumer, and citizen access to your business-to-consumer (B2C) applications.
RBAC	Identity and Access Management ↗	Azure role-based access control	Azure role-based access control (Azure RBAC) helps you manage who has access to Azure resources, what they can do with those resources, and what areas they have access to.

Area	Google Cloud service	Azure service	Description
ABAC	Identity and Access Management ↗	Azure attribute-based access control	Azure attribute-based access control (Azure ABAC) is an authorization system that defines access, based on attributes that are associated with security principals, resources, and environment.
Zero trust	BeyondCorp Enterprise ↗	Azure AD Conditional Access	Conditional Access is the tool used by Azure Active Directory to bring signals together, to make decisions, and to enforce organizational policies.
Resource management	Resource Manager ↗	Azure Resource Manager	Provides a management layer that enables you to create, update, and delete resources in your Azure account, like access control, locks, and tags, to secure and organize your resources after deployment.
Encryption	Cloud KMS ↗, Secret Manager ↗	Azure Key Vault ↗	Provides a security solution and works with other services by allowing you to manage, create, and control encryption keys that are stored in hardware security modules (HSM).
Data-at-rest encryption	Encryption at rest ↗	Azure Storage Service Encryption - encryption by default	Azure Storage Service Encryption helps you protect and safeguard your data and meet your organizational security and compliance commitments.
Data in-use	Confidential Computing ↗	Azure Confidential Computing	Encrypt data in-use.
Hardware security module (HSM)	Cloud HSM ↗	Azure Dedicated HSM	Azure service that provides cryptographic key storage in Azure, to host encryption keys and perform cryptographic operations in a high-availability service of FIPS 140-2 Level 3 certified hardware security modules (HSMs).
Data loss prevention (DLP)	Cloud Data Loss Prevention ↗	Azure Information Protection	Azure Information Protection (AIP) is a cloud-based solution that enables organizations to discover, classify, and protect documents and emails by applying labels to content.

Area	Google Cloud service	Azure service	Description
Security	Security Command Center ↗ , Web Security Scanner ↗	Microsoft Defender for Cloud ↗	An automated security assessment service that improves the security and compliance of applications. Automatically assess applications for vulnerabilities or deviations from best practices.
Threat detection	Event Threat Detection ↗	Azure Advanced Threat Protection ↗	Detect and investigate advanced attacks on-premises and in the cloud.
SIEM	Chronicle ↗	Microsoft Sentinel ↗	A cloud-native security information and event manager (SIEM) platform that uses built-in AI to help analyze large volumes of data from all sources, including users, applications, servers, and devices that are running on-premises or in any cloud.
Container security	Container Security ↗	Container Security in Microsoft Defender for Cloud	Microsoft Defender for Cloud is the Azure-native solution for securing your containers.
	Artifact Registry ↗	Azure Container Registry	A managed, private Docker registry service that's based on the open-source Docker Registry 2.0. Create and maintain Azure container registries to store and manage your private Docker container images and related artifacts that allow you to only deploy trusted containers.
	Container Analysis ↗	Microsoft Defender for container registries	Perform vulnerability scans on all container images when they're pushed to the registry, imported into the registry, or pulled within the last 30 days.

Security architectures



Real-time fraud detection

7/05/2018 4 min read

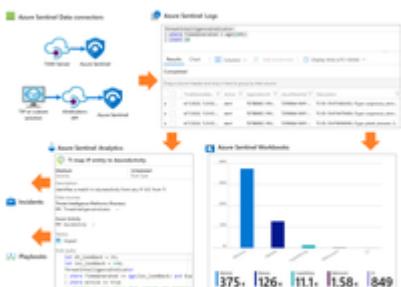
Detect fraudulent activity in real-time using Azure Event Hubs and Stream Analytics.



Securely managed web applications

5/09/2019 8 min read

Learn about deploying secure applications using the Azure App Service Environment, the Azure Application Gateway service, and Web Application Firewall.



Threat indicators for cyber threat intelligence in Azure Sentinel

4/13/2020 13 min read

Import threat indicators, view logs, create rules to generate security alerts and incidents, and visualize threat intelligence data with Azure Sentinel.

[view all](#)

Storage

Object storage

Google Cloud service	Azure service	Description
Cloud Storage for Firebase ↗	Azure Blob storage	Object storage service, for use cases including cloud applications, content distribution, backup, archiving, disaster recovery, and big data analytics.

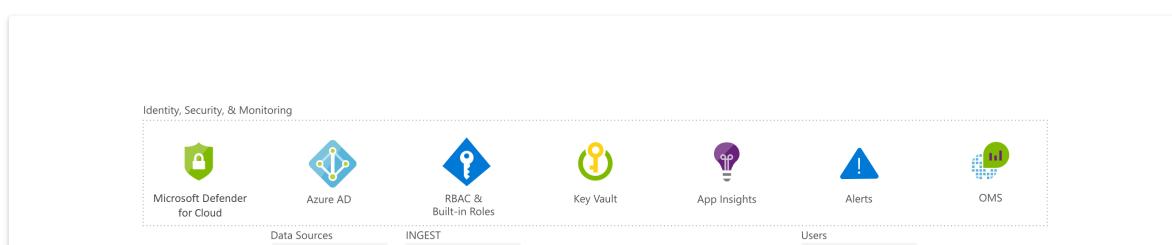
Block storage

Google Cloud service	Azure service	Description
Persistant Disk ↗	Azure managed disks ↗	SSD storage optimized for I/O intensive read/write operations. For use as high-performance Azure virtual machine storage.

File storage

Google Cloud service	Azure service	Description
Filestore ↗	Azure Files ↗ , Azure NetApp Files ↗	File based storage and hosted NetApp Appliance Storage.
Google Drive ↗	OneDrive For business ↗	Cloud storage and file sharing solution for businesses to store, access, and share files anytime and anywhere.

Storage architectures

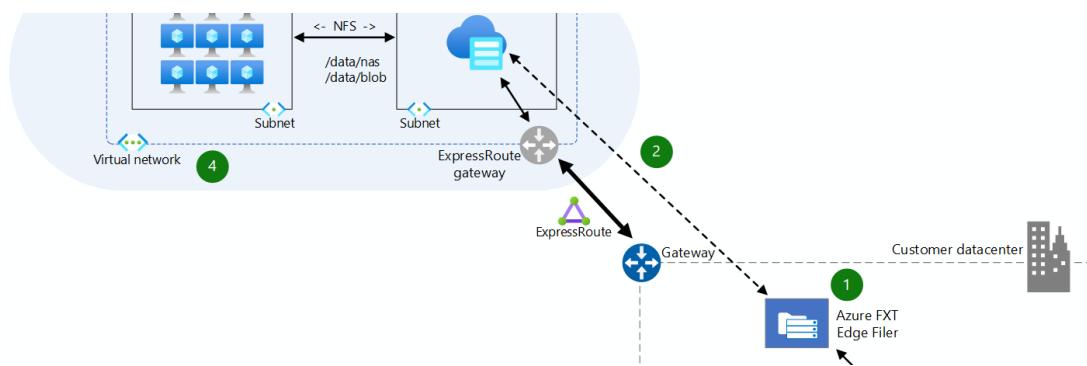




HIPAA and HITRUST compliant health data AI

12/16/2019 2 min read

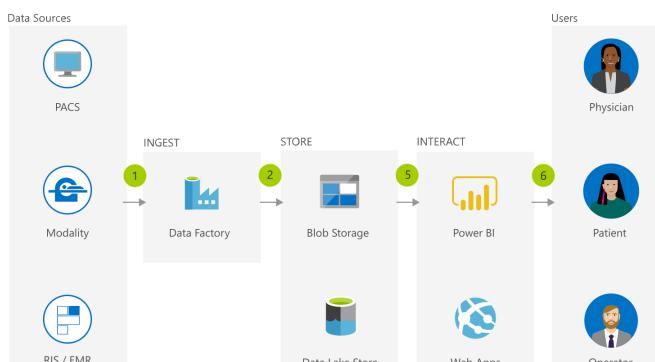
Manage HIPAA and HITRUST compliant health data and medical records with the highest level of built-in security.



HPC Media Rendering

11/04/2020 2 min read

Optimize the media rendering process with a step-by-step HPC solution architecture from Azure that combines Azure CycleCloud and HPC Cache.



Medical Data Storage Solutions

12/16/2019 2 min read

Store healthcare data effectively and affordably with cloud-based solutions from Azure. Manage medical records with the highest level of built-in security.

[view all](#)

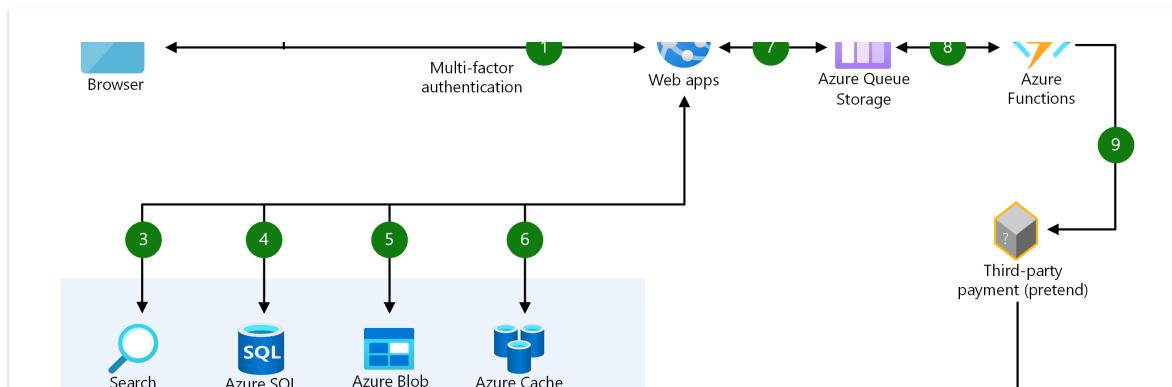
Bulk data transfer

Google Cloud service	Azure service	Description
Transfer Appliance ↗	Azure Import/Export	A data transport solution that uses secure disks and appliances to transfer large amounts of data. Also offers data protection during transit.
Transfer Appliance ↗	Azure Data Box ↗	Petabyte- to exabyte-scale data transport solution that uses secure data storage devices to transfer large amounts of data to and from Azure.

Application services

Google Cloud service	Azure service	Description
App Engine ↗	Azure App Service ↗	Managed hosting platform providing easy to use services for deploying and scaling web applications and services.
Apigee ↗	Azure API Management ↗	A turnkey solution for publishing APIs to external and internal consumers.

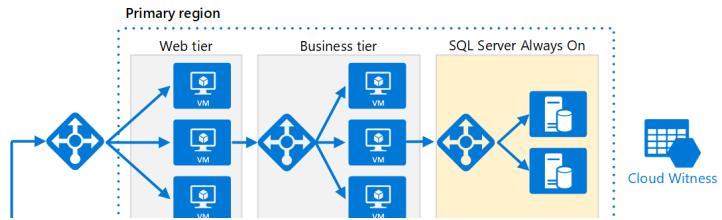
Web architectures



Architect scalable e-commerce web app

12/16/2019 1 min read

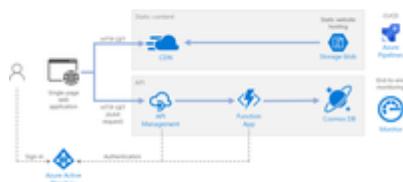
The e-commerce website includes simple order processing workflows with the help of Azure services. Using Azure Functions and Web Apps, developers can focus on building personalized experiences and let Azure take care of the infrastructure.



Multi-region N-tier application

6/18/2019 10 min read

Deploy an application on Azure virtual machines in multiple regions for high availability and resiliency.



Serverless web application

5/28/2019 16 min read

This reference architecture shows a serverless web application, which serves static content from Azure Blob Storage and implements an API using Azure Functions.

[view all](#)

Miscellaneous

Area	Google Cloud service	Azure service	Description
Workflow	Composer	Azure Logic Apps	Serverless technology for connecting apps, data and devices anywhere, whether on-premises or in the cloud for large ecosystems of SaaS and cloud-based connectors.
Enterprise application services	G Suite	Microsoft 365	Fully integrated Cloud service providing communications, email, document management in the cloud and available on a wide variety of devices.

Area	Google Cloud service	Azure service	Description
Gaming	Game Servers	Azure PlayFab	Managed services for hosting dedicated game servers.
Hybrid	Anthos	Azure Arc	For customers who want to simplify complex and distributed environments across on-premises, edge and multi-cloud, Azure Arc enables deployment of Azure services anywhere and extends Azure management to any infrastructure.
Blockchain	Digital Asset	Azure Confidential Ledger	Tamperproof, unstructured data store hosted in trusted execution environments and backed by cryptographically verifiable evidence.
Monitoring	Cloud Monitoring	Application Insights	Service that provides visibility into the performance, uptime, and overall health of cloud-powered applications.
Logging	Cloud Logging	Log Analytics	Service for real-time log management and analysis.

Migration tools

Area	Google Cloud service	Azure Service	Description
App migration to containers	Migrate for Anthos	Azure Migrate: App Containerization tool	Modernize your application by migrating it to AKS or App Services containers.
Migration of virtual machines	Migrate for Compute Engine	Azure Migrate: Server Migration tool	Migrate servers from anywhere to Azure.
VMware migration	Google Cloud VMware Engine	Azure VMware Solution	Move or extend on-premises VMware environments to Azure.
Migration of databases	Database Migration Service	Azure Database Migration Service	Fully managed service designed to enable seamless migrations from multiple database sources to Azure data platforms with minimal downtime.

Area	Google Cloud service	Azure Service	Description
Migration programs	Google Cloud Rapid Assessment & Migration Program (RAMP) 	Azure Migration and Modernization Program 	Learn how to move your apps, data, and infrastructure to Azure using a proven cloud migration and modernization approach.
Server assessment		Movere	Increases business intelligence by accurately presenting entire IT environments within a single day.
Database assessment		Data Migration Assistant	It helps pinpoint potential problems blocking migration. It identifies unsupported features, new features that can benefit you after migration, and the right path for database migration.
Web app assessment and migration		Web app migration assistant 	Assess on-premises web apps and migrate them to Azure.

Next steps

If you are new to Azure, review the interactive [Core Cloud Services - Introduction to Azure](#) module on [Microsoft Learn training](#).

Use a Terraform plan to deploy a Google Cloud Platform Ubuntu instance and connect it to Azure Arc

Article • 03/31/2023

This article provides guidance for using the provided [Terraform](#) plan to deploy Google Cloud Platform (GCP) instance and connect it as an Azure Arc-enabled server resource.

Prerequisites

1. Clone the Azure Arc Jumpstart repository.

Console

```
git clone https://github.com/microsoft/azure_arc.git
```

2. [Install or update Azure CLI to version 2.7 and above](#). Use the following command to check your current installed version.

Console

```
az --version
```

3. [Generate SSH key](#) (or use existing SSH key)

4. [Create free Google Cloud Platform account](#)

5. [Install Terraform >= 0.12](#)

6. Create an Azure service principal.

To connect the GCP virtual machine to Azure Arc, an Azure service principal assigned with the Contributor role is required. To create it, sign in to your Azure account and run the following command. You can also run this command in [Azure Cloud Shell](#).

Console

```
az login
az ad sp create-for-rbac -n "<Unique SP Name>" --role contributor
```

For example:

Console

```
az ad sp create-for-rbac -n "http://AzureArcGCP" --role contributor
```

Output should look like this:

JSON

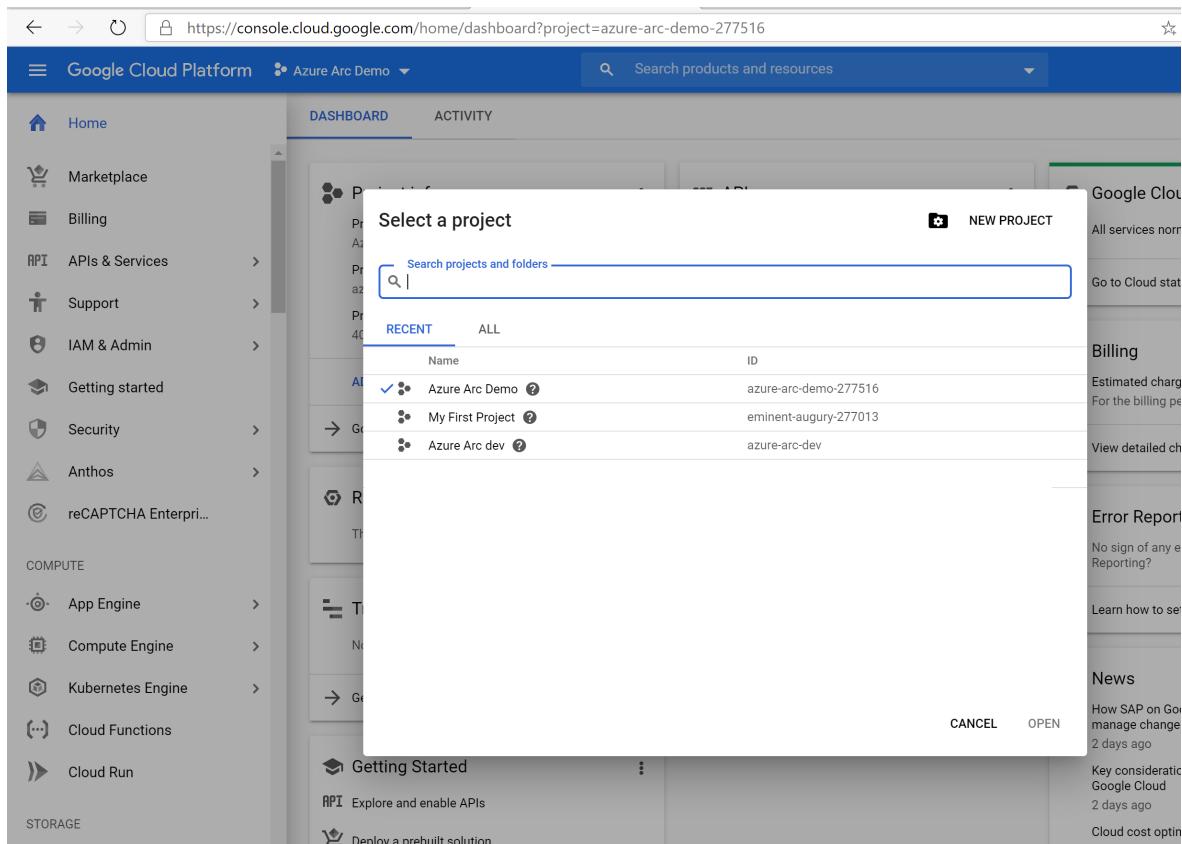
```
{  
  "appId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",  
  "displayName": "AzureArcGCP",  
  "name": "http://AzureArcGCP",  
  "password": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",  
  "tenant": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"  
}
```

ⓘ Note

We highly recommend that you scope the service principal to a specific **Azure subscription and resource group**.

Create a new GCP project

1. Browse to the [Google API console](#) and sign-in with your Google account. Once logged in, [create a new project](#) named `Azure Arc demo`. After creating it, be sure to copy down the project ID since it's usually different then the project name.



← → ⏪ https://console.cloud.google.com/projectcreate?previousPage=%2F

Google Cloud Platform

New Project

⚠ You have 20 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name * [?](#)

Project ID: `azure-arc-demo-277522`. It cannot be changed later. [EDIT](#)

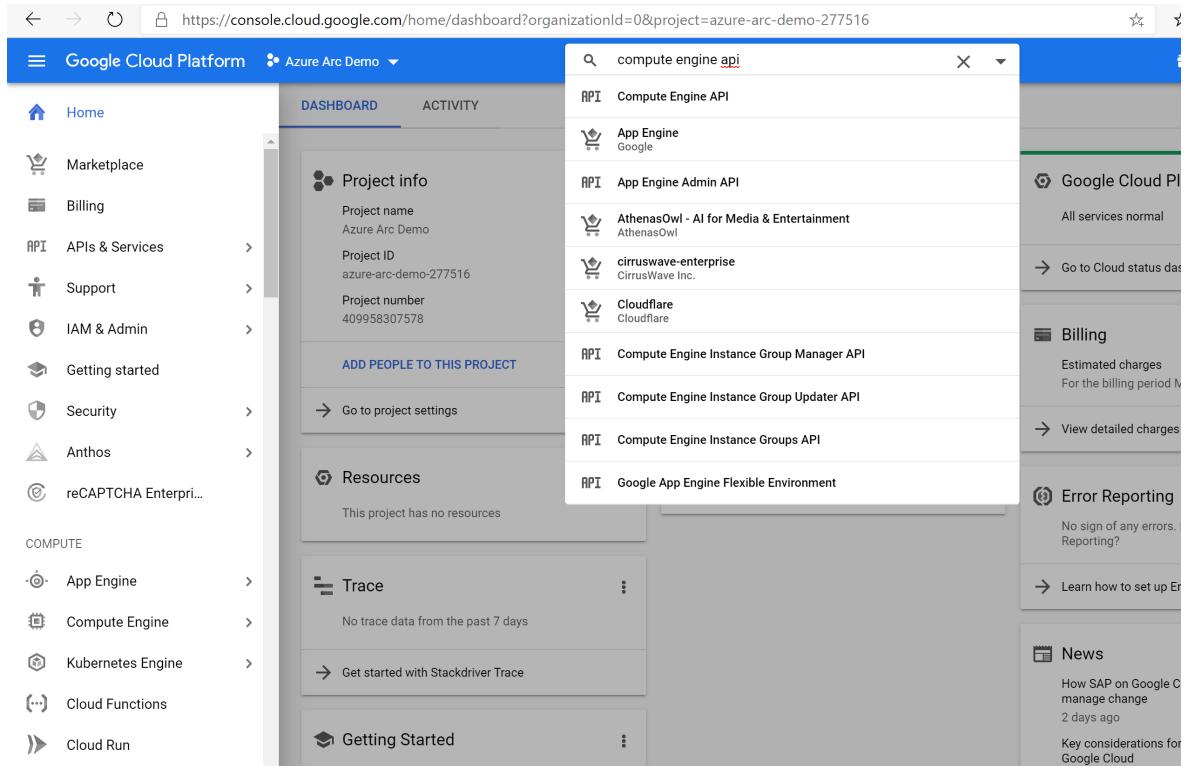
Location * [BROWSE](#)

Parent organization or folder

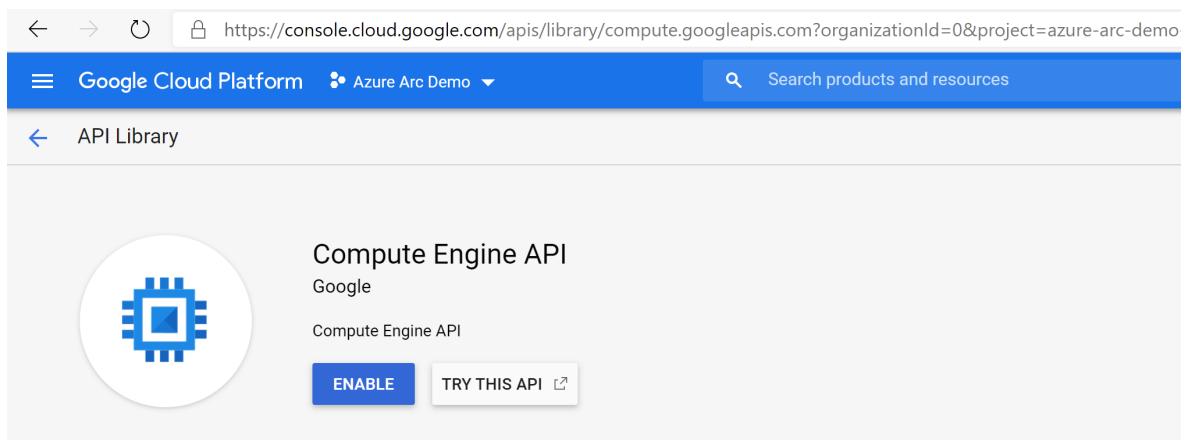
CREATE [CANCEL](#)

2. Once the new project is created and selected in the dropdown list at the top of the page, you must enable Compute Engine API access for the project. Click on +

Enable APIs and Services and search for *compute engine*. Then select **Enable** to enable API access.



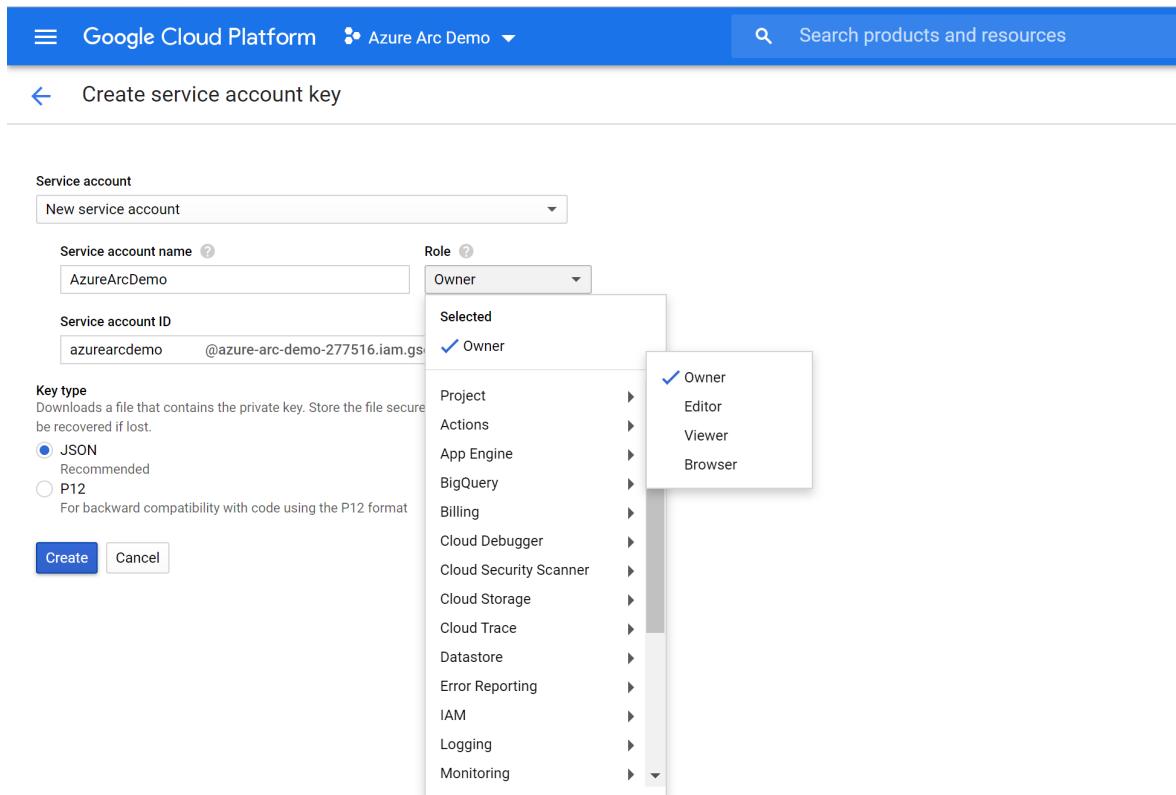
The screenshot shows the Google Cloud Platform Dashboard for the project 'Azure Arc Demo'. A search bar at the top right is set to 'compute engine api'. Below the search bar, a list of API results is displayed, including 'Compute Engine API' (Google), 'App Engine Admin API' (Google), 'Compute Engine Instance Group Manager API', 'Compute Engine Instance Group Updater API', 'Compute Engine Instance Groups API', and 'Google App Engine Flexible Environment'. The 'Compute Engine API' entry is the top result.



The screenshot shows the Google Cloud Platform API Library page for the 'Compute Engine API' (Google). The page features a large circular icon of a microchip. Below the icon, there are two buttons: 'ENABLE' and 'TRY THIS API'. The 'ENABLE' button is highlighted in blue. To the right of the button, there is a 'Search products and resources' bar. The page also includes sections for 'Overview' (describing the API as creating and running virtual machines), 'About Google' (mentioning Google's mission to organize information), and 'Tutorials and documentation'.

3. Next, set up a service account key, which Terraform will use to create and manage resources in your GCP project. Go to the [create service account key page](#). Select **New Service Account** from the dropdown list, give it a name, select project then owner as the role, JSON as the key type, and select **Create**. This downloads a JSON file with all the credentials that will be needed for Terraform to manage the

resources. Copy the downloaded JSON file to the `azure_arc_servers_jumpstart/gcp/ubuntu/terraform` directory.



4. Finally, make sure your SSH keys are available in `~/.ssh` and named `id_rsa.pub` and `id_rsa`. If you followed the `ssh-keygen` guide above to create your key then this should already be set up correctly. If not, you may need to modify [main.tf](#) to use a key with a different path.

Deployment

Before executing the Terraform plan, you must export the environment variables which will be used by the plan. These variables are based on the Azure service principal you've just created, your Azure subscription and tenant, and the GCP project name.

1. Retrieve your Azure subscription ID and tenant ID using the `az account list` command.
2. The Terraform plan creates resources in both Microsoft Azure and Google Cloud Platform. It then executes a script on a GCP virtual machine to install the Azure Arc agent and all necessary artifacts. This script requires certain information about your GCP and Azure environments. Edit [scripts/vars.sh](#) and update each of the variables with the appropriate values.
 - `TF_VAR_subscription_id` = your Azure subscription ID

- `TF_VAR_client_id` = your Azure service principal application ID
- `TF_VAR_client_secret` = your Azure service principal password
- `TF_VAR_tenant_id` = your Azure tenant ID
- `TF_VAR_gcp_project_id` = GCP project ID
- `TF_VAR_gcp_credentials_filename` = GCP credentials JSON filename

3. From CLI, navigate to the `azure_arc_servers_jumpstart/gcp/ubuntu/terraform` directory of the cloned repo.

4. Export the environment variables you edited by running [scripts/vars.sh](#) with the source command as shown below. Terraform requires these to be set for the plan to execute properly. Note that this script will also be automatically executed remotely on the GCP virtual machine as part of the Terraform deployment.

Console

```
source ./scripts/vars.sh
```

5. Run the `terraform init` command which will download the Terraform AzureRM provider.

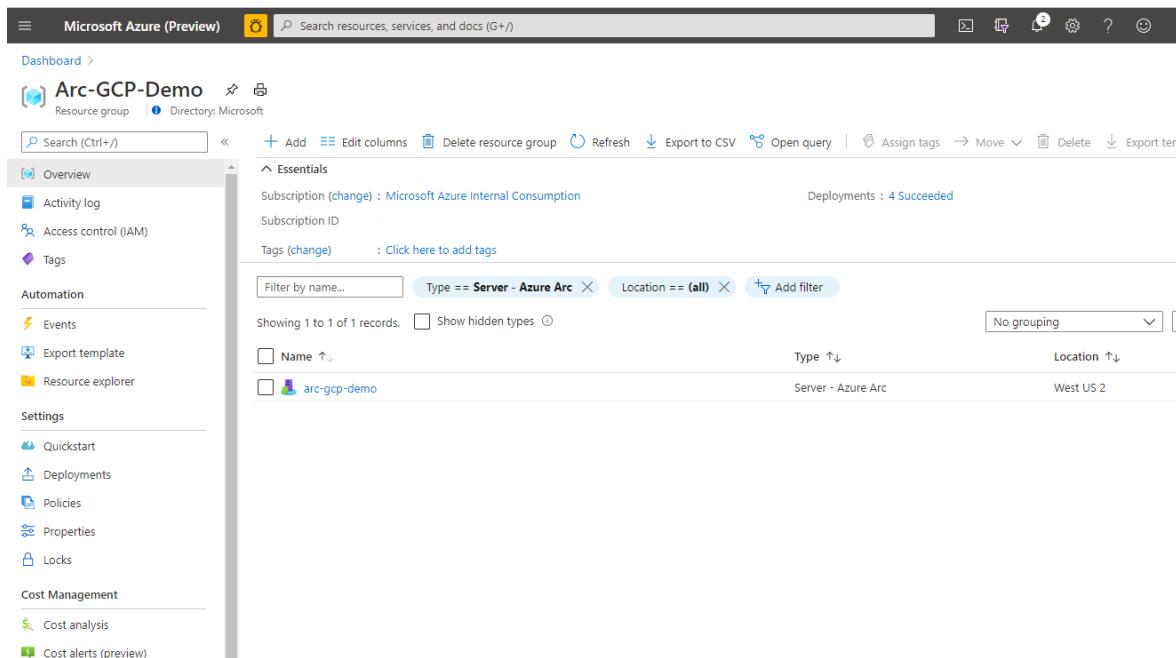
```
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/ubuntu/terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/ubuntu/terraform$
```

6. Next, run the `terraform apply --auto-approve` command and wait for the plan to finish. Upon completion, you will have a GCP Ubuntu VM deployed and connected as a new Azure Arc-enabled server inside a new resource group.

7. Open the Azure portal and navigate to the `arc-gcp-demo` resource group. The virtual machine created in GCP will be visible as a resource.



The screenshot shows the Microsoft Azure (Preview) portal. In the top left, it says 'Arc-GCP-Demo' and 'Resource group'. The top navigation bar includes 'Search resources, services, and docs (G+)', 'Refresh', 'Export to CSV', 'Open query', 'Assign tags', 'Move', 'Delete', and 'Export item'. The left sidebar has sections for 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Automation', 'Events', 'Export template', 'Resource explorer', 'Settings' (Quickstart, Deployments, Policies, Properties, Locks), 'Cost Management' (Cost analysis, Cost alerts (preview)), and 'Logs'. The main content area is titled 'Essentials' and shows 'Subscription (change) : Microsoft Azure Internal Consumption' and 'Deployments : 4 Succeeded'. Below that is a table with a single row: 'arc-gcp-demo' (Type: Server - Azure Arc, Location: West US 2). There are filters at the top of the table: 'Filter by name...', 'Type == Server - Azure Arc', 'Location == (all)', and 'Add filter'.

Semi-automated deployment (optional)

As you may have noticed, the last step of the run is to register the VM as a new Azure Arc-enabled server resource.

```
11  # Run connect command
12 ✓ azcmagent connect \
13   --service-principal-id $TF_VAR_client_id \
14   --service-principal-secret $TF_VAR_client_secret \
15   --resource-group "Arc-Servers-Demo" \
16   --tenant-id $TF_VAR_tenant_id \
17   --location "westus2" \
18   --subscription-id $TF_VAR_subscription_id
```

If you want to demo/control the actual registration process, do the following:

1. In the [install_arc_agent.sh.tpl](#) script template, comment out the `run connect` command section and save the file.

```
11  # Run connect command
12  # azcmagent connect \
13  #   --service-principal-id $TF_VAR_client_id \
14  #   --service-principal-secret $TF_VAR_client_secret \
15  #   --resource-group "Arc-Servers-Demo" \
16  #   --tenant-id $TF_VAR_tenant_id \
17  #   --location "westus2" \
18  #   --subscription-id $TF_VAR_subscription_id
```

2. Get the public IP of the GCP VM by running `terraform output`.

```
$ terraform output
ip = 34.83.189.233
```

3. SSH the VM using the `ssh arcadmin@xx.xx.xx.xx` where `xx.xx.xx.xx` is the host IP.

```
$ ssh arcadmin@34.83.189.233
The authenticity of host '34.83.189.233 (34.83.189.233)' can't be established.
ECDSA key fingerprint is SHA256:+qJV20JPW8CDuzbsPJB2H2ekiTRw709xahNLtn+g8uo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.83.189.233' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-1061-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

17 packages can be updated.
9 updates are security updates.

New release '18.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu May 14 19:24:45 2020 from 99.106.206.153
arcadmin@gcp-vm-ff65732d953ef823:~$
```

4. Export all the environment variables in `vars.sh`

```
arcadmin@gcp-vm-ff65732d953ef823:~$ export TF_VAR_client_id="3079...9118fee"
arcadmin@gcp-vm-ff65732d953ef823:~$ export TF_VAR_client_secret="f0b65...cc...384f"
arcadmin@gcp-vm-ff65732d953ef823:~$ export TF_VAR_tenant_id="72...11db47"
arcadmin@gcp-vm-ff65732d953ef823:~$ export TF_VAR_subscription_id="28d8...f4ba9"
arcadmin@gcp-vm-ff65732d953ef823:~$ export TF_VAR_gcp_project_id="azure-arc-demo-277216"
arcadmin@gcp-vm-ff65732d953ef823:~$ export TF_VAR_gcp_credentials_filename="azure-arc-demo-277216-3dbb6bbce8ad.json"
arcadmin@gcp-vm-ff65732d953ef823:~$
```

5. Run the following command:

Console

```
azcmagent connect --service-principal-id $TF_VAR_client_id --service-principal-secret $TF_VAR_client_secret --resource-group "Azure Arc gcp-demo" --tenant-id $TF_VAR_tenant_id --location "westus2" --subscription-id $TF_VAR_subscription_id
```

```
arcadmin@gcp-vm-ff65732d953ef823:~$ azcmagent connect --service-principal-id $TF_VAR_client_id --service-principal-secret $TF_VAR_client_secret --resource-group "Arc-Servers-Demo" --tenant-id $TF_VAR_tenant_id --location "westus2" --subscription-id $TF_VAR_subscription_id
level=info msg="Onboarding Machine. It usually takes a few minutes to complete. Sometimes it may take longer depending on network and server load status."
level=info msg="Successfully Onboarded Resource to Azure" VM Id=ae8b32df-c1fe-4d97-bf2f-aebc7f60d54f
arcadmin@gcp-vm-ff65732d953ef823:~$
```

6. When complete, your VM will be registered with Azure Arc and visible in the resource group via the Azure portal.

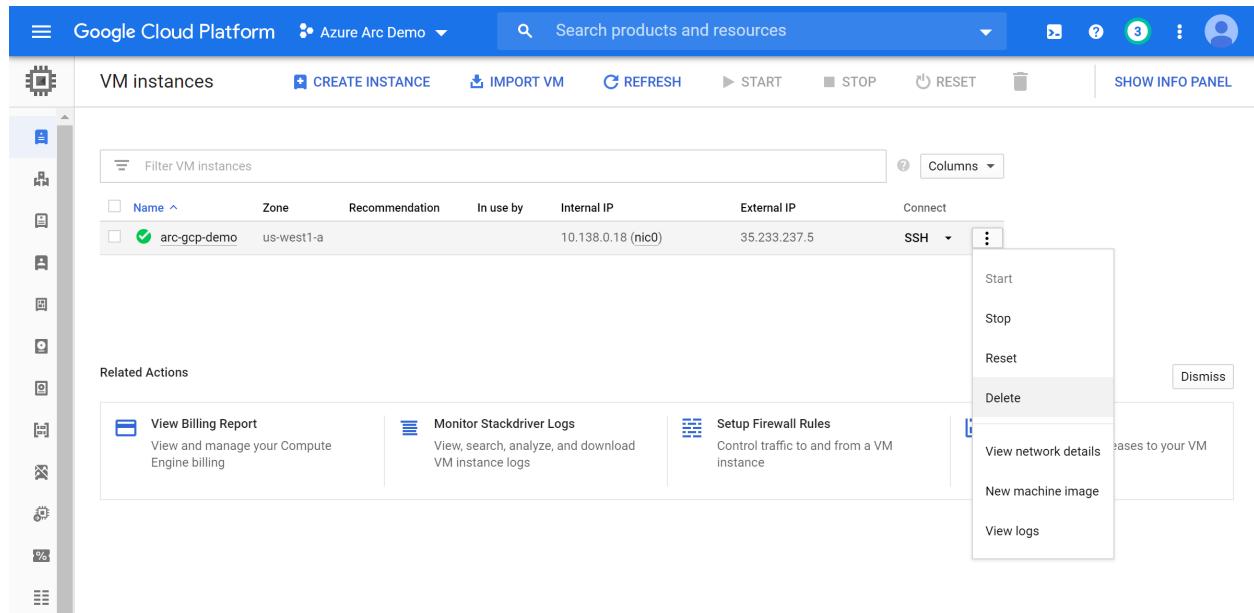
Delete the deployment

To delete all the resources you created as part of this demo use the `terraform destroy -auto-approve` command as shown below.

```
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/ubuntu/terraform$ terraform destroy --auto-approve
local_file.install_arc_agent_sh: Refreshing state... [id=90c12ba7a84d03ad453e0105f3145858ba9732a5]
google_compute_instance.default: Refreshing state... [id=projects/azure-arc-demo-277516/zones/us-west1-a/instances/arc-gcp-demo]
azurerm_resource_group.azure_rg: Refreshing state... [id=/subscriptions/_____/resourceGroups/Arc-Servers-Demo]
local_file.install_arc_agent_sh: Destroying... [id=90c12ba7a84d03ad453e0105f3145858ba9732a5]
local_file.install_arc_agent_sh: Destruction complete after 0s
google_compute_instance.default: Destroying... [id=projects/azure-arc-demo-277516/zones/us-west1-a/instances/arc-gcp-demo]
azurerm_resource_group.azure_rg: Destroying... [id=/subscriptions/_____/resourceGroups/Arc-Servers-Demo]
azurerm_resource_group.azure_rg: Still destroying... [id=/subscriptions/_____/resourceGroups/Arc-Servers-Demo, 30s elapsed]
google_compute_instance.default: Still destroying... [id=projects/azure-arc-demo-277516/zones/us-west1-a/instances/arc-gcp-demo, 40s elapsed]
google_compute_instance.default: Destruction complete after 43s
azurerm_resource_group.azure_rg: Still destroying... [id=/subscriptions/_____/resourceGroups/Arc-Servers-Demo, 40s elapsed]
azurerm_resource_group.azure_rg: Destruction complete after 1m49s

Destroy complete! Resources: 3 destroyed.
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/ubuntu/terraform$
```

Alternatively, you can delete the GCP VM directly from [GCP console](#).



The screenshot shows the Google Cloud Platform interface for managing VM instances. In the center, a table lists a single VM instance named "arc-gcp-demo" in the "us-west1-a" zone. The instance has an Internal IP of 10.138.0.18 (nic0) and an External IP of 35.233.237.5. A context menu is open over this instance, with the "Delete" option highlighted. The menu also includes "Start", "Stop", "Reset", and "View logs" options. Below the table, there are "Related Actions" such as "View Billing Report", "Monitor Stackdriver Logs", and "Setup Firewall Rules".

Use a Terraform plan to deploy a Google Cloud Platform Windows instance and connect it to Azure Arc

Article • 03/31/2023

This article provides guidance for using the provided [Terraform](#) plan to deploy a Windows Server Google Cloud Platform (GCP) instance and connect it as an Azure Arc-enabled server resource.

Prerequisites

1. Clone the Azure Arc Jumpstart repository.

Console

```
git clone https://github.com/microsoft/azure_arc.git
```

2. [Install or update Azure CLI to version 2.7 and above](#). Use the following command to check your current installed version.

Console

```
az --version
```

3. [Install Terraform >= 0.12](#)

4. **Google Cloud Platform account with billing enabled:** [Create a free trial account](#).

To create Windows Server virtual machines, you must upgrade your account to enable billing. Select **Billing** from the menu and then select **Upgrade** at the lower right.

Home

Marketplace

Billing

API APIs & Services

Support

IAM & Admin

Getting started

Security

Anthos

Overview My Billing Account ▾

BILLING ACCOUNT OVERVIEW

[View report](#)

PAYMENT OVERVIEW

Cost trend

May 1, 2019 – May 31, 2020

Average monthly total cost

\$0.00



[View report](#)

Top projects

API APIs

Requests (requests/sec)

⚠ No data is available

9:30 9:45

[Go to APIs overview](#)

Billing health checks

Check out your account health results to avoid common billing-related issues and adopt our best practice recommendations. [Learn more](#)

!

0

!

1

✓

0

[View all health checks](#)

Free trial credit



\$300

Free trial credit

Out of \$300



358

Days remaining

Ends May 12, 2021

You will not be billed during your free trial. To keep your projects running after the free trial is up, upgrade to a paid account.

[UPGRADE](#)

[LEARN MORE](#)

Upgrade your account

You're one step away from unlocking all of Google Cloud Platform.

You won't be charged until after your free credits run out or expire (whichever comes first). [Learn more](#)

You only pay for what you use. [View pricing details](#)

CANCEL

UPGRADE

Disclaimer: To prevent unexpected charges, follow the "delete the deployment" section at the end of this article.

5. Create an Azure service principal.

To connect the GCP virtual machine to Azure Arc, an Azure service principal assigned with the Contributor role is required. To create it, sign in to your Azure account and run the following command. You can also run this command in [Azure Cloud Shell](#).

Console

```
az login
az ad sp create-for-rbac -n "<Unique SP Name>" --role contributor
```

For example:

Console

```
az ad sp create-for-rbac -n "http://AzureArcGCP" --role contributor
```

Output should look like this:

JSON

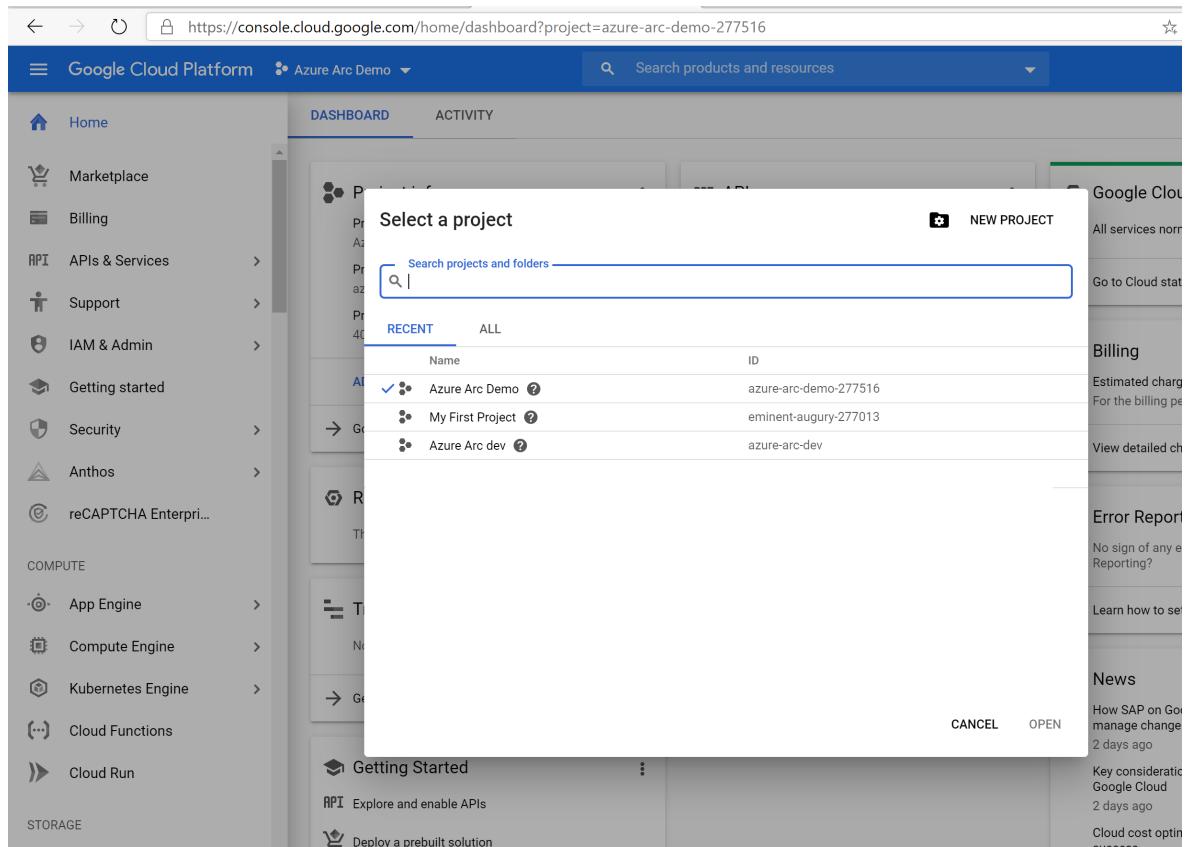
```
{
  "appId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "displayName": "AzureArcGCP",
  "name": "http://AzureArcGCP",
  "password": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "tenant": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
}
```

ⓘ Note

We highly recommend that you scope the service principal to a specific **Azure subscription and resource group**.

Create a new GCP project

1. Browse to the [Google API console](https://console.cloud.google.com/) and sign-in with your Google account. Once logged in, [create a new project](#) named `Azure Arc demo`. After creating it, be sure to copy the project ID since it's usually different from the project name.



← → 🔍 https://console.cloud.google.com/projectcreate?previousPage=%2F

☰ Google Cloud Platform Search product

New Project

⚠ You have 20 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name * ?

Project ID: azure-arc-demo-277522. It cannot be changed later. [EDIT](#)

Location * [BROWSE](#)

Parent organization or folder

CREATE **CANCEL**

2. Once the new project is created and selected in the dropdown list at the top of the page, you must enable compute engine API access for the project. Click on **+** **Enable APIs and Services** and search for *compute engine*. Then select **Enable** to enable API access.

← → 🔍 https://console.cloud.google.com/home/dashboard?organizationId=0&project=azure-arc-demo-277516

☰ Google Cloud Platform Azure Arc Demo

[Home](#)

[Marketplace](#)

[Billing](#)

[APIs & Services](#) >

[Support](#) >

[IAM & Admin](#) >

[Getting started](#)

[Security](#) >

[Anthos](#) >

[reCAPTCHA Enterprise](#)

COMPUTE

[App Engine](#) >

[Compute Engine](#) >

[Kubernetes Engine](#) >

[Cloud Functions](#) >

[Cloud Run](#)

DASHBOARD

ACTIVITY

Project info

Project name: Azure Arc Demo

Project ID: azure-arc-demo-277516

Project number: 409958307578

[ADD PEOPLE TO THIS PROJECT](#)

[Go to project settings](#)

Resources

This project has no resources

Trace

No trace data from the past 7 days

[Get started with Stackdriver Trace](#)

Getting Started

compute engine api

API Compute Engine API

App Engine Google

API App Engine Admin API

AthenasOwl - AI for Media & Entertainment AthenasOwl

cirruswave-enterprise CirrusWave Inc.

Cloudflare Cloudflare

API Compute Engine Instance Group Manager API

API Compute Engine Instance Group Updater API

API Compute Engine Instance Groups API

API Google App Engine Flexible Environment

← → ⌂ https://console.cloud.google.com/apis/library/compute.googleapis.com?organizationId=0&project=azure-arc-demo

≡ Google Cloud Platform • Azure Arc Demo ▾

Search products and resources

← API Library

 **Compute Engine API**
Google
Compute Engine API

ENABLE TRY THIS API ↗

Type	Overview
APIs & services	Creates and runs virtual machines on Google Cloud Platform.
Last updated	About Google
10/15/18, 11:36 PM	Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.
Category	Tutorials and documentation
Compute	Compute
Networking	Networking
Service name	compute.googleapis.com
	Learn more ↗

3. Next, set up a service account key, which Terraform will use to create and manage resources in your GCP project. Go to the [create service account key page](#) ↗. Select **New Service Account** from the dropdown list, give it a name, select project then owner as the role, JSON as the key type, and select **Create**. This downloads a JSON file with all the credentials needed for Terraform to manage the resources. Copy the downloaded JSON file to the

`azure_arc_servers_jumpstart/gcp/windows/terraform` directory.

≡ Google Cloud Platform • Azure Arc Demo ▾

Search products and resources

← Create service account key

Service account

New service account

Service account name Role

Service account ID Selected Owner

Key type Downloads a file that contains the private key. Store the file securely if lost.

JSON Recommended

P12 For backward compatibility with code using the P12 format

Create **Cancel**

- Owner
- Editor
- Viewer
- Browser

Deployment

Before executing the Terraform plan, you must set and then export the environment variables which will be used by the plan. These variables are based on the Azure service principal you've just created, your Azure subscription and tenant, and the GCP project name.

1. Retrieve your Azure subscription ID and tenant ID using the `az account list` command.
2. The Terraform plan creates resources in both Microsoft Azure and Google Cloud Platform. It then executes a script on a GCP virtual machine to install the Azure Arc agent and all necessary artifacts. This script requires certain information about your GCP and Azure environments. Edit [scripts/vars.sh](#) and update each of the variables with the appropriate values.
 - `TF_VAR_subscription_id` = your Azure subscription ID
 - `TF_VAR_client_id` = your Azure service principal application ID
 - `TF_VAR_client_secret` = your Azure service principal password
 - `TF_VAR_tenant_id` = your Azure tenant ID
 - `TF_VAR_gcp_project_id` = GCP project ID
 - `TF_VAR_gcp_credentials_filename` = GCP credentials JSON filename
3. From CLI, navigate to the `azure_arc_servers_jumpstart/gcp/windows/terraform` directory of the cloned repo.
4. Export the environment variables you edited by running [scripts/vars.sh](#) with the source command as shown below. Terraform requires these to be set for the plan to execute properly.

Console

```
source ./scripts/vars.sh
```

5. Run the `terraform init` command which will download the Terraform AzureRM provider.

```
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/windows/terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.random: version = "~> 2.2"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/windows/terraform$
```

6. Next, run the `terraform apply --auto-approve` command and wait for the plan to finish. Upon completion of the Terraform script, you will have deployed a GCP Windows Server 2019 VM and initiated a script to download the Azure Arc agent to the VM and connect the VM as a new Azure Arc-enabled server inside a new Azure resource group. It will take a few minutes for the agent to finish provisioning, so grab a cup of coffee.

```
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/windows/terraform$ terraform apply --auto-approve
local_file.install_arc_agent_ps1: Creating...
local_file.install_arc_agent_ps1: Creation complete after 0s [id=5f4472875f2634d1a6ea0b29349969fc5d2d14b8]
google_compute_instance.default: Creating...
azurerm_resource_group.azure_rg: Creating...
azurerm_resource_group.azure_rg: Creation complete after 2s [id=/subscriptions/.../resourceGroups/Arc-Servers-Demo]
google_compute_instance.default: Still creating... [10s elapsed]
google_compute_instance.default: Creation complete after 14s [id=projects/azure-arc-demo-277516/zones/us-west1-a/instances/arc-gcp-demo]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:
ip = 34.83.26.90
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/windows/terraform$
```

7. After a few minutes, you should be able to open the Azure portal and navigate to the `arc-gcp-demo` resource group. The Windows Server virtual machine created in GCP will be visible as a resource.

Semi-automated deployment (optional)

The Terraform plan automatically installs the Azure Arc agent and connects the VM to Azure as a managed resource by executing a PowerShell script when the VM is first booted.

```
azure_arc_servers_jumpstart > gcp > windows > terraform > scripts > > install_arc_agent.ps1.tpl > ...
1 # Injecting environment variables
2
3 # Set Azure vars
4 $env:subscriptionId="${subscriptionId}"
5 $env:appId="${appId}"
6 $env:password="${appPassword}"
7 $env:tenantId="${tenantId}"
8 $env:resourceGroup="${resourceGroup}"
9 $env:location="${location}"
10
11 # Download the package
12 function download() { $ProgressPreference="SilentlyContinue"; Invoke-WebRequest -Uri https://aka.ms/AzureConnectedMachineAgent -OutFile .\AzureConnectedMachineAgent.msi
13 download
14
15 # Install the package
16 msieexec /i AzureConnectedMachineAgent.msi /l*v installationlog.txt /qn | Out-String
17
18 # Run connect command
19 & "$env:ProgramFiles\AzureConnectedMachineAgent\azcmagent.exe" connect ` 
20 --service-principal-id $env:appId ` 
21 --service-principal-secret $env:password ` 
22 --resource-group $env:resourceGroup ` 
23 --tenant-id $env:tenantId ` 
24 --location $env:location ` 
25 --subscription-id $env:subscriptionId
```

If you want to demo/control the actual registration process, do the following:

1. Before running the `terraform apply` command, open `main.tf` and comment out the `windows-startup-script-ps1 = local-file.install_arc_agent-ps1.content` line and save the file.

```

28
29   metadata = [
30     // windows-startup-script-ps1 = local_file.install_arc_agent_ps1.content
31   ]
32 }
33

```

2. Run `terraform apply --auto-approve` as instructed above.
3. Open the GCP console and navigate to the [compute instance page](#), and then select the VM that was created.

The screenshot shows the Google Cloud Platform VM instances page. The VM 'arc-gcp-demo' is listed with the following details:

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
arc-gcp-demo	us-west1-a			10.138.0.17 (nic0)	34.83.26.90	RDP

The screenshot shows the Google Cloud Platform VM instance details page for 'arc-gcp-demo'. The instance details are as follows:

- Details** tab is selected.
- Remote access**: RDP is selected, and a 'Set Windows password' button is present.
- Logs**: Stackdriver Logging is selected.
- Serial port 1**: More options are available.
- Instance Id**: 5708896675463435496
- Machine type**: n1-standard-1 (1 vCPU, 3.75 GB memory)
- Reservation**: Automatically choose (default)
- CPU platform**: Intel Broadwell
- Display device**: Turn on a display device if you want to use screen capturing and recording tools.
- Zone**: us-west1-a
- Labels**: None
- Creation time**: May 20, 2020, 5:07:03 PM
- Network interfaces** table:

Name	Network	Subnetwork	Primary internal IP	Alias IP ranges	External IP	Network Tier	IP forwarding	Network details
nic0	default	default	10.138.0.17	—	34.83.26.90 (ephemeral)	Premium	Off	View details

4. Create a user and password for the VM by selecting **Set Password** and specifying a user name.
-

Set new Windows password

If a Windows account with the following username does not exist, it will be created and a new password assigned. If the account exists, its password will be reset.

⚠ If the account already exists, resetting the password can cause the loss of encrypted data secured with the current password, including files and stored passwords. [Learn more](#)

Username 

dale

[CANCEL](#) [SET](#)

5. RDP into the VM by selecting the RDP button from the VM page in the GCP console, and sign in with the username and password you just created.

VM instance details

arc-gcp-demo

Details Monitoring

Remote access

RDP Set Windows password Connect to serial console

Logs Stackdriver Serial port More

View gcloud command to reset password

Download the RDP file

Learn about Windows auth

Instance Id

5708896675463435496

Machine type

n1-standard-1 (1 vCPU, 3.75 GB memory)

Reservation

Automatically choose (default)

CPU platform

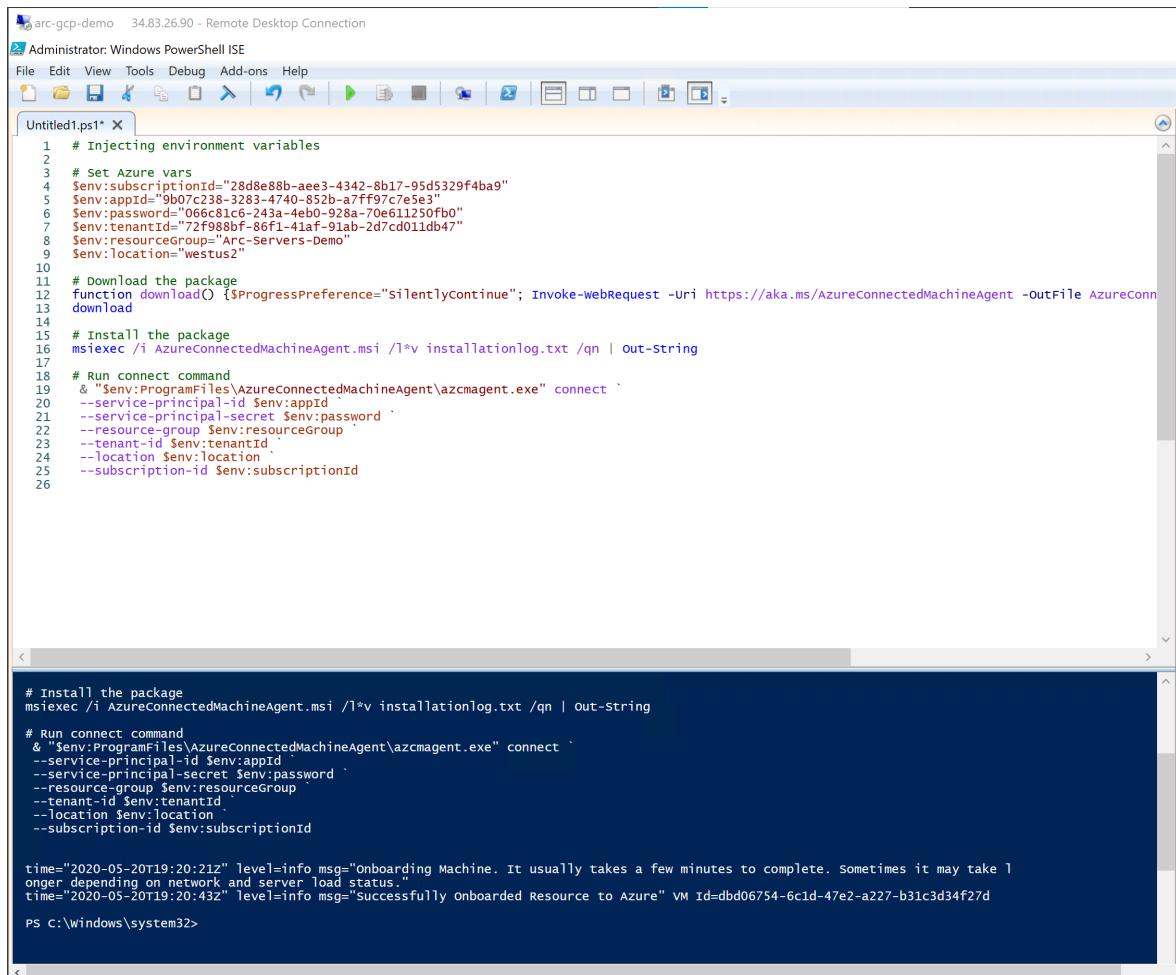
Intel Broadwell

Display device

Turn on a display device if you want to use screen capturing and recording tools.

Turn on display device

6. Once logged in, open PowerShell ISE as **Administrator**. Make sure you are running the x64 version of PowerShell ISE and not the x86 version. Once opened, select **File** > **New** to create an empty `.ps1` file. Then paste in the entire contents of `./scripts/install_arc_agent.ps1`. Click the play button to execute the script. When complete, you should see the output showing successful onboarding of the machine.



```

arc-gcp-demo 34.83.26.90 - Remote Desktop Connection
Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1* X
1 # Injecting environment variables
2
3 # Set Azure vars
4 $env:subscriptionId="28d8e88b-aee3-4342-8b17-95d5329f4ba9"
5 $env:appId="9b07c238-3283-4740-852b-a7ff97c7e5e3"
6 $env:password="066c81c6-243a-4eb0-928a-70e611250fb0"
7 $env:tenantId="72f988bf-86f1-41af-91ab-2d7cd011db47"
8 $env:resourceGroup="Arc-Servers-Demo"
9 $env:location="westus2"
10
11 # Download the package
12 function download() { $ProgressPreference="silentlyContinue"; Invoke-WebRequest -Uri https://aka.ms/AzureConnectedMachineAgent -OutFile AzureConn
13 download
14
15 # Install the package
16 msieexec /i AzureConnectedMachineAgent.msi /l*v installationlog.txt /qn | out-string
17
18 # Run connect command
19 & "$env:ProgramFiles\AzureConnectedMachineAgent\azcmagent.exe" connect `
20 --service-principal-id $env:appId `
21 --service-principal-secret $env:password `
22 --resource-group $env:resourceGroup
23 --tenant-id $env:tenantId
24 --location $env:location
25 --subscription-id $env:subscriptionId
26

```

```

# Install the package
msieexec /i AzureConnectedMachineAgent.msi /l*v installationlog.txt /qn | out-string
# Run connect command
& "$env:ProgramFiles\AzureConnectedMachineAgent\azcmagent.exe" connect `
--service-principal-id $env:appId `
--service-principal-secret $env:password `
--resource-group $env:resourceGroup
--tenant-id $env:tenantId
--location $env:location
--subscription-id $env:subscriptionId

time="2020-05-20T19:20:21Z" level=info msg="Onboarding Machine. It usually takes a few minutes to complete. Sometimes it may take longer depending on network and server load status."
time="2020-05-20T19:20:43Z" level=info msg="Successfully onboarded Resource to Azure" VM Id=dbd06754-6c1d-47e2-a227-b31c3d34f27d
PS C:\Windows\system32>

```

Delete the deployment

To delete all the resources you created as part of this demo use the `terraform destroy -auto-approve` command as shown below.

```

:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/windows/terraform$ terraform destroy --auto-approve
local_file.install_arc_agent_ps1: Refreshing state... [id=5f4472875f2634d1a6ea0b29349969fc5d2d14b8]
google_compute_instance.default: Refreshing state... [id=projects/azure-arc-demo-277516/zones/us-west1-a/instances/arc-gcp-demo]
azurerm_resource_group.azure_rg: Refreshing state... [id=/subscriptions/.../resourceGroups/Arc-Servers-Demo]
google_compute_instance.default: Destroying... [id=projects/azure-arc-demo-277516/zones/us-west1-a/instances/arc-gcp-demo]
azurerm_resource_group.azure_rg: Destroying... [id=/subscriptions/.../resourceGroups/Arc-Servers-Demo]
google_compute_instance.default: Still destroying... [id=projects/azure-arc-demo-277516/zones/us-west1-a/instances/arc-gcp-demo, 10s elapsed]
google_compute_instance.default: Destruction complete after 53s
local_file.install_arc_agent_ps1: Destroying... [id=5f4472875f2634d1a6ea0b29349969fc5d2d14b8]
local_file.install_arc_agent_ps1: Destruction complete after 0s
azurerm_resource_group.azure_rg: Still destroying... [id=/subscriptions/.../resourceGroups/Arc-Servers-Demo, 1m30s elapsed]
azurerm_resource_group.azure_rg: Destruction complete after 1m35s

Destroy complete! Resources: 3 destroyed.
:/mnt/c/dev/azure_arc/azure_arc_servers_jumpstart/gcp/windows/terraform$ 

```

Alternatively, you can delete the GCP VM directly from [GCP console](#).

Google Cloud Platform Azure Arc Demo Search products and resources

VM instances CREATE INSTANCE IMPORT VM REFRESH START STOP RESET SHOW INFO PANEL

Filter VM instances Columns

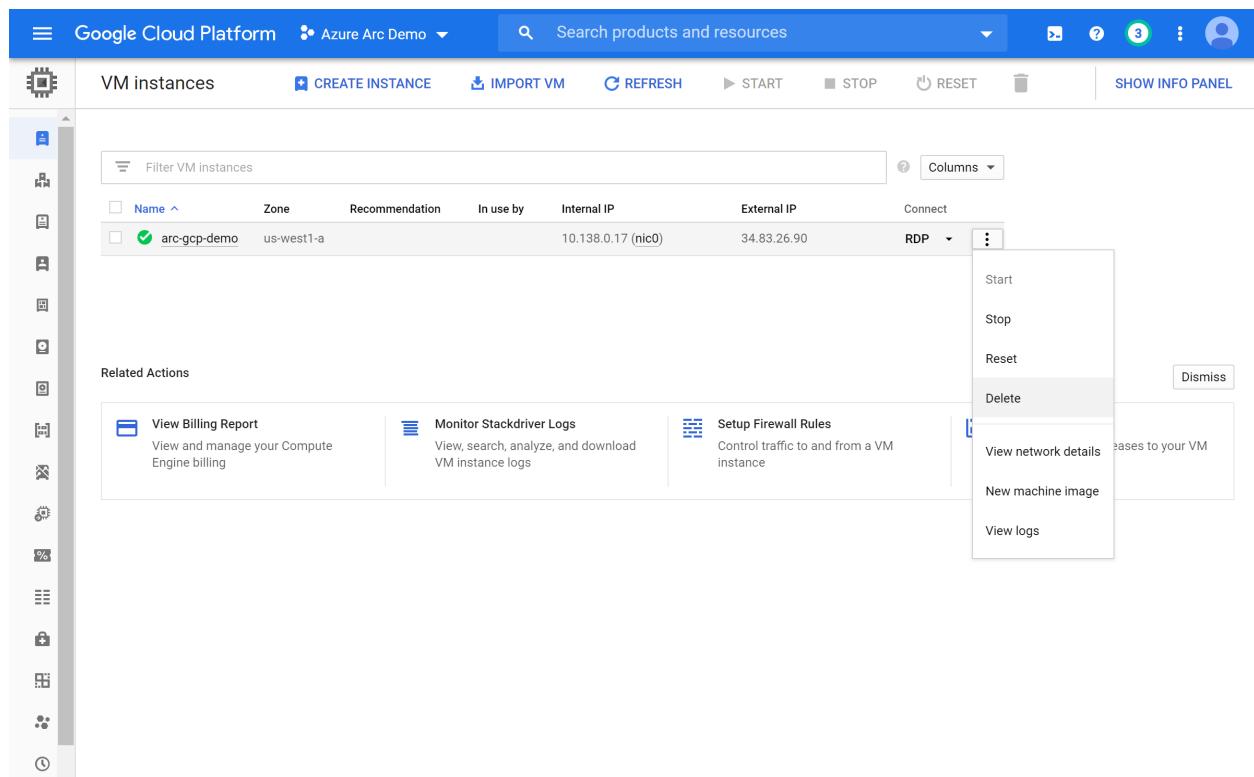
Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
<input checked="" type="checkbox"/> arc-gcp-demo	us-west1-a			10.138.0.17 (nic0)	34.83.26.90	RDP

Related Actions

- [View Billing Report](#)
View and manage your Compute Engine billing
- [Monitor Stackdriver Logs](#)
View, search, analyze, and download VM instance logs
- [Setup Firewall Rules](#)
Control traffic to and from a VM instance

Start
Stop
Reset
Delete
View network details
New machine image
View logs

Dismiss



Onboard a Google Cloud Platform (GCP) project

Article • 12/20/2023

This article describes how to onboard a Google Cloud Platform (GCP) project in Microsoft Entra Permissions Management.

ⓘ Note

You must have the Global Administrator role assignment to perform the tasks in this article.

Explanation

For GCP, Permissions Management is scoped to a *GCP project*. A GCP project is a logical collection of your resources in GCP, like a subscription in Azure, but with further configurations you can perform such as application registrations and OIDC configurations.

There are several moving parts across GCP and Azure, which should be configured before onboarding.

- A Microsoft Entra OIDC App
- A Workload Identity in GCP
- OAuth2 confidential client grants utilized
- A GCP service account with permissions to collect

Onboard a GCP project

1. If the **Data Collectors** dashboard isn't displayed when Permissions Management launches:
 - In the Permissions Management home page, select **Settings** (the gear icon), and then select the **Data Collectors** subtab.
2. On the **Data Collectors** tab, select **GCP**, then select **Create Configuration**.

1. Create a Microsoft Entra OIDC app.

1. On the Permissions Management Onboarding - Microsoft Entra OIDC App

Creation page, enter the **OIDC Azure App Name**.

This app is used to set up an OpenID Connect (OIDC) connection to your GCP project. OIDC is an interoperable authentication protocol based on the OAuth 2.0 family of specifications. The scripts generated creates the app of this specified name in your Microsoft Entra tenant with the right configuration.

2. To create the app registration, copy the script and run it in your command-line app.

① Note

- a. To confirm the app was created, open **App registrations** in Azure and, on the **All applications** tab, locate your app.
- b. Select the app name to open the **Expose an API** page. The **Application ID URI** displayed in the **Overview** page is the *audience value* used while making an OIDC connection with your GCP account.
- c. Return to the Permissions Management window, and in the **Permissions Management Onboarding - Microsoft Entra OIDC App Creation**, select **Next**.

2. Set up a GCP OIDC project.

1. In the **Permissions Management Onboarding - GCP OIDC Account Details & IDP Access** page, enter the **OIDC Project Number** and **OIDC Project ID** of the GCP project in which the OIDC provider and pool is created. You can change the role name to your requirements.

① Note

You can find the **Project number** and **Project ID** of your GCP project on the **GCP Dashboard** page of your project in the **Project info** panel.

2. You can change the **OIDC Workload Identity Pool Id**, **OIDC Workload Identity Pool Provider Id** and **OIDC Service Account Name** to meet your requirements.

Optionally, specify **G-Suite IDP Secret Name** and **G-Suite IDP User Email** to enable G-Suite integration.

3. You can either download and run the script at this point or you can do it in the Google Cloud Shell.

4. Select **Next** after successfully running the setup script.

Choose from three options to manage GCP projects.

Option 1: Automatically manage

The automatically manage option allows you to automatically detect and monitor projects without extra configuration. Steps to detect a list of projects and onboard for collection:

1. Grant **Viewer** and **Security Reviewer** roles to a service account created in the previous step at a project, folder or organization level.

To enable Controller mode **On** for any projects, add these roles to the specific projects:

- Role Administrators
- Security Admin

The required commands to run in Google Cloud Shell are listed in the Manage Authorization screen for each scope of a project, folder or organization. This is also configured in the GCP console.

3. Select **Next**.

Option 2: Enter authorization systems

You have the ability to specify only certain GCP member projects to manage and monitor with Permissions Management (up to 100 per collector). Follow the steps to configure these GCP member projects to be monitored:

1. In the **Permissions Management Onboarding - GCP Project IDs** page, enter the **Project IDs**.

You can enter up to comma separated 100 GCP project IDs.

2. You can choose to download and run the script at this point, or you can do it via Google Cloud Shell.

To enable controller mode 'On' for any projects, add these roles to the specific projects:

- Role Administrators

- Security Admin

3. Select **Next**.

Option 3: Select authorization systems

This option detects all projects accessible by the Cloud Infrastructure Entitlement Management application.

1. Grant **Viewer** and **Security Reviewer** roles to a service account created in the previous step at a project, folder or organization level.

To enable Controller mode **On** for any projects, add these roles to the specific projects:

- Role Administrators
- Security Admin

The required commands to run in Google Cloud Shell are listed in the Manage Authorization screen for each scope of a project, folder or organization. This is also configured in the GCP console.

3. Select **Next**.

3. Review and save.

1. In the **Permissions Management Onboarding – Summary** page, review the information you've added, and then select **Verify Now & Save**.

The following message appears: **Successfully Created Configuration**.

On the **Data Collectors** tab, the **Recently Uploaded On** column displays **Collecting**. The **Recently Transformed On** column displays **Processing**.

The status column in your Permissions Management UI shows you which step of data collection you're at:

- **Pending**: Permissions Management has not started detecting or onboarding yet.
- **Discovering**: Permissions Management is detecting the authorization systems.
- **In progress**: Permissions Management has finished detecting the authorization systems and is onboarding.
- **Onboarded**: Data collection is complete, and all detected authorization systems are onboarded to Permissions Management.

4. View the data.

1. To view the data, select the **Authorization Systems** tab.

The **Status** column in the table displays **Collecting Data**.

The data collection process takes some time and occurs in approximately 4-5 hour intervals in most cases. The time frame depends on the size of the authorization system you have and how much data is available for collection.

Next steps

- To enable or disable the controller after onboarding is complete, see [Enable or disable the controller](#).
- To add an account/subscription/project after onboarding is complete, see [Add an account/subscription/project after onboarding is complete](#).

Add and remove roles and tasks for Microsoft Azure and Google Cloud Platform (GCP) identities

Article • 10/23/2023

This article describes how you can add and remove roles and tasks for Microsoft Azure and Google Cloud Platform (GCP) identities using the **Remediation** dashboard.

ⓘ Note

To view the **Remediation** tab, you must have **Viewer**, **Controller**, or **Administrator** permissions. To make changes on this tab, you must have **Controller** or **Administrator** permissions. If you don't have these permissions, contact your system administrator.

View permissions

1. On the Permissions Management home page, select the **Remediation** tab, and then select the **Permissions** subtab.
2. From the **Authorization System Type** dropdown, select **Azure** or **GCP**.
3. From the **Authorization System** dropdown, select the accounts you want to access.
4. From the **Search For** dropdown, select **Group**, **User**, or **APP**.
5. To search for more parameters, you can make a selection from the **User States**, **Permission Creep Index**, and **Task Usage** dropdowns.
6. Select **Apply**. Microsoft Entra ID displays a list of groups, users, and service accounts that match your criteria.
7. In **Enter a username**, enter or select a user.
8. In **Enter a Group Name**, enter or select a group, then select **Apply**.
9. Make a selection from the results list.

The table displays the **Username**, **Domain/Account**, **Source**, **Resource** and **Current Role**.

Add a role

1. On the Permissions Management home page, select the **Remediation** tab, and then select the **Permissions** subtab.
2. From the **Authorization System Type** dropdown, select **Azure** or **GCP**.
3. From the **Authorization System** dropdown, select the accounts you want to access.
4. From the **Search For** dropdown, select **Group**, **User**, or **APP/Service Account**, and then select **Apply**.
5. Make a selection from the results list.
6. To attach a role, select **Add role**.
7. In the **Add Role** page, from the **Available Roles** list, select the plus sign (+) to move the role to the **Selected Roles** list.
8. When you have finished adding roles, select **Submit**.
9. When the following message displays: **Are you sure you want to change permission?**, select:
 - **Generate Script** to generate a script where you can manually add/remove the permissions you selected.
 - **Execute** to change the permission.
 - **Close** to cancel the action.

Remove a role

1. On the Permissions Management home page, select the **Remediation** tab, and then select the **Permissions** subtab.
2. From the **Authorization System Type** dropdown, select **Azure** or **GCP**.
3. From the **Authorization System** dropdown, select the accounts you want to access.
4. From the **Search For** dropdown, select **Group**, **User**, or **APP/Service Account**, and then select **Apply**.
5. Make a selection from the results list.
6. To remove a role, select **Remove Role**.

7. In the Remove Role page, from the **Available Roles** list, select the plus sign (+) to move the role to the **Selected Roles** list.

8. When you have finished selecting roles, select **Submit**.

9. When the following message displays: **Are you sure you want to change permission?**, select:

- **Generate Script** to generate a script where you can manually add/remove the permissions you selected.
- **Execute** to change the permission.
- **Close** to cancel the action.

Add a task

1. On the Permissions Management home page, select the **Remediation** tab, and then select the **Permissions** subtab.

2. From the **Authorization System Type** dropdown, select **Azure** or **GCP**.

3. From the **Authorization System** dropdown, select the accounts you want to access.

4. From the **Search For** dropdown, select **Group**, **User**, or **APP/Service Account**, and then select **Apply**.

5. Make a selection from the results list.

6. To attach a role, select **Add Tasks**.

7. In the **Add Tasks** page, from the **Available Tasks** list, select the plus sign (+) to move the task to the **Selected Tasks** list.

8. When you have finished adding tasks, select **Submit**.

9. When the following message displays: **Are you sure you want to change permission?**, select:

- **Generate Script** to generate a script where you can manually add/remove the permissions you selected.
- **Execute** to change the permission.
- **Close** to cancel the action.

Remove a task

1. On the Permissions Management home page, select the **Remediation** tab, and then select the **Permissions** subtab.
2. From the **Authorization System Type** dropdown, select **Azure** or **GCP**.
3. From the **Authorization System** dropdown, select the accounts you want to access.
4. From the **Search For** dropdown, select **Group**, **User**, or **APP/Service Account**, and then select **Apply**.
5. Make a selection from the results list.
6. To remove a task, select **Remove Tasks**.
7. In the **Remove Tasks** page, from the **Available Tasks** list, select the plus sign (+) to move the task to the **Selected Tasks** list.
8. When you have finished selecting tasks, select **Submit**.
9. When the following message displays: **Are you sure you want to change permission?**, select:
 - **Generate Script** to generate a script where you can manually add/remove the permissions you selected.
 - **Execute** to change the permission.
 - **Close** to cancel the action.

Next steps

- For information on how to view existing roles/policies, requests, and permissions, see [View roles/policies, requests, and permission in the Remediation dashboard](#).
- To view information about roles/policies, see [View information about roles/policies](#).

Tutorial: Discover Google Cloud Platform (GCP) instances with Azure Migrate: Discovery and assessment

Article • 11/03/2023

As part of your migration journey to Azure, you discover your servers for assessment and migration.

This tutorial shows you how to discover Google Cloud Platform (GCP) instances with the Azure Migrate: Discovery and assessment tool, using a lightweight Azure Migrate appliance. You deploy the appliance on a server on GCP, to continuously discover machine and performance metadata.

In this tutorial, you learn how to:

- ✓ Set up an Azure account.
- ✓ Prepare server on GCP for discovery.
- ✓ Create a project.
- ✓ Set up the Azure Migrate appliance.
- ✓ Start continuous discovery.

ⓘ Note

Tutorials show the quickest path for trying out a scenario and using default options.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

Before you start this tutorial, check you have these prerequisites in place.

Requirement	Details
Appliance	<p>You need a server on GCP on which to run the Azure Migrate appliance. The machine should have:</p> <ul style="list-style-type: none">- Windows Server 2019 or Windows Server 2022 installed. <i>Running the appliance on a machine with Windows Server 2019 isn't supported.</i>- 16-GB RAM, 8 vCPUs, around 80 GB of disk storage, and an external virtual switch.- A static or dynamic IP address, with internet access, either directly or through a proxy.
Windows server instances	Allow inbound connections on WinRM port 5985 (HTTP) for discovery of Windows servers.

Requirement	Details
Linux server instances	Allow inbound connections on port 22 (TCP) for discovery of Linux servers.

Prepare an Azure user account

To create a project and register the Azure Migrate appliance, you need an account with:

- Contributor or Owner permissions on an Azure subscription.
- Permissions to register Microsoft Entra apps.

If you just created a free Azure account, you're the owner of your subscription. If you're not the subscription owner, work with the owner to assign the permissions as follows:

1. In the Azure portal, search for "subscriptions", and under **Services**, select **Subscriptions**.

2. In the **Subscriptions** page, select the subscription in which you want to create a project.
3. Select **Access control (IAM)**.
4. Select **Add > Add role assignment** to open the **Add role assignment** page.
5. Assign the following role. For detailed steps, see [Assign Azure roles using the Azure portal](#).

Setting	Value
Role	Contributor or Owner
Assign access to	User
Members	azmigrateuser

Name ↑↓	Description ↑↓	Type ↑↓	Category ↑↓	Details
Owner	Grants full access to manage all resources, including the ability to a...	BuiltInRole	General	View
Contributor	Grants full access to manage all resources, but does not allow you ...	BuiltInRole	General	View
Reader	View all resources, but does not allow you to make any changes.	BuiltInRole	General	View
AcrDelete	acr delete	BuiltInRole	Containers	View
AcrImageSigner	acr image signer	BuiltInRole	Containers	View
AcrPull	acr pull	BuiltInRole	Containers	View
AcrPush	acr push	BuiltInRole	Containers	View
AcrQuarantineReader	acr quarantine data reader	BuiltInRole	Containers	View
AcrQuarantineWriter	acr quarantine data writer	BuiltInRole	Containers	View

6. To register the appliance, your Azure account needs [permissions to register Microsoft Entra apps](#).
7. In the portal, go to [Microsoft Entra ID > Users](#).
8. Request the tenant or global admin to assign the [Application Developer role](#) to the account to allow Microsoft Entra app registration by users. [Learn more](#).

Prepare GCP instances

Set up an account that the appliance can use to access servers on GCP.

- For Windows servers:
 - Set up a local user account on non-domain joined servers, and a domain account on domain joined servers that you want to include in the discovery. Add the user account to the following groups:
 - Remote Management Users
 - Performance Monitor Users
 - Performance Log users.
- For Linux servers:
 - You need a root account on the Linux servers that you want to discover. If you aren't able to provide a root account, refer to the instructions in the [support matrix](#) for an alternative.
 - Azure Migrate uses password authentication when discovering GCP instances. GCP instances don't support password authentication by default. Before you can discover instance, you need to enable password authentication.
 1. Sign into each Linux machine.
 2. Open the sshd_config file: vi /etc/ssh/sshd_config

3. In the file, locate the **PasswordAuthentication** line, and change the value to **yes**.
 4. Save the file and close it. Restart the ssh service.
- o If you're using a root user to discover your Linux servers, ensure root login is allowed on the servers.
 1. Sign into each Linux machine
 2. Open the `sshd_config` file: `vi /etc/ssh/sshd_config`
 3. In the file, locate the **PermitRootLogin** line, and change the value to **yes**.
 4. Save the file and close it. Restart the ssh service.

Set up a project

Set up a new project.

1. In the Azure portal > **All services**, search for **Azure Migrate**.
2. Under **Services**, select **Azure Migrate**.
3. In **Get started**, select **Create project**.
4. In **Create project**, select your Azure subscription and resource group. Create a resource group if you don't have one.
5. In **Project Details**, specify the project name and the geography in which you want to create the project. Review supported geographies for **public** and **government** clouds.
6. Select **Create**.
7. Wait a few minutes for the project to deploy. The **Azure Migrate: Discovery and assessment** tool is added by default to the new project.

Home > Azure Migrate

Azure Migrate | Servers, databases and web apps

Microsoft

Search < Create project Refresh Feedback

Get started Explore more

Migration goals

- Servers, databases and web apps
- Databases (only)
- VDI
- Web apps
- Data Box

Assessment tools

Azure Migrate: Discovery and assessment

Discover Build business case Dependency analysis Assess Overview Resolve issues

Quick start

- 1: Discover**
Discover your on-premises servers by using an appliance or importing in a CSV format. Click "Discover" to get started.
- 2: Build and review business case**
Review the recommended migration strategy and financial analysis for migrating your datacenter to Azure.
- 3: Analyze dependencies**
Analyze dependencies between servers. Click 'Dependency analysis' to get started.
- 4: Assess**
Assess discovered servers for migration to Azure. Click 'Assess' to get started.

Add more assessment tools? [Click here](#).

! Note

If you have already created a project, you can use the same project to register additional appliances to discover and assess more no of servers. [Learn more](#).

Set up the appliance

The Azure Migrate appliance is a lightweight appliance, used by Azure Migrate: Discovery and assessment to do the following:

- Discover on-premises servers.
- Send metadata and performance data for discovered servers to Azure Migrate: Discovery and assessment.

[Learn more](#) about the Azure Migrate appliance.

To set up the appliance, you:

1. Provide an appliance name and generate a project key in the portal.
2. Download a zipped file with Azure Migrate installer script from the Azure portal.
3. Extract the contents from the zipped file. Launch the PowerShell console with administrative privileges.
4. Execute the PowerShell script to launch the appliance web application.
5. Configure the appliance for the first time and register it with the project using the project key.

1. Generate the project key

1. In **Migration goals > Servers, databases and web apps > Azure Migrate: Discovery and assessment**, select **Discover**.
2. In **Discover servers > Are your servers virtualized?**, select **Physical or other (AWS, GCP, Xen, etc.)**.
3. In **1:Generate project key**, provide a name for the Azure Migrate appliance that you'll set up for discovery of your GCP virtual servers. The name should be alphanumeric with 14 characters or fewer.
4. Click **Generate key** to start the creation of the required Azure resources. Don't close the **Discover servers** page during the creation of resources.
5. After the successful creation of the Azure resources, a **project key** is generated.
6. Copy the key as you'll need it to complete the registration of the appliance during its configuration.

2. Download the installer script

In 2: **Download Azure Migrate appliance**, click **Download**.

Verify security

Check that the zipped file is secure before you deploy it.

1. On the machine to which you downloaded the file, open an administrator command window.
2. Run the following command to generate the hash for the zipped file:

- `C:\>CertUtil -HashFile <file_location> [Hashing Algorithm]`
- Example usage for public cloud: `C:\>CertUtil -HashFile C:\Users\administrator\Desktop\AzureMigrateInstaller-Server-Public.zip SHA256`
- Example usage for government cloud: `C:\>CertUtil -HashFile C:\Users\administrator\Desktop\AzureMigrateInstaller-Server-USGov.zip SHA256`

3. Verify the latest appliance versions and hash values:

- For the public cloud:

Scenario	Download	Hash value
Physical (85 MB)	Latest version ↗	7EF01AE30F7BB8F4486EDC1688481DB656FB8ECA7B9EF6363B4DAB1CFCFDA141

- For Azure Government:

Scenario	Download	Hash value
Physical (85 MB)	Latest version ↗	7EF01AE30F7BB8F4486EDC1688481DB656FB8ECA7B9EF6363B4DAB1CFCFDA141

3. Run the Azure Migrate installer script

The installer script does the following:

- Installs agents and a web application for GCP server discovery and assessment.
- Install Windows roles, including Windows Activation Service, IIS, and PowerShell ISE.
- Download and installs an IIS rewritable module.
- Updates a registry key (HKLM) with persistent setting details for Azure Migrate.
- Creates the following files under the path:
 - **Config Files:** %Programdata%\Microsoft Azure\Config
 - **Log Files:** %Programdata%\Microsoft Azure\Logs

Run the script as follows:

1. Extract the zipped file to a folder on the server that will host the appliance. Make sure you don't run the script on a machine on an existing Azure Migrate appliance.
2. Launch PowerShell on the above server with administrative (elevated) privilege.
3. Change the PowerShell directory to the folder where the contents have been extracted from the downloaded zipped file.

4. Run the script named `AzureMigrateInstaller.ps1` by running the following command:

- For the public cloud:

```
PS C:\Users\administrator\Desktop\AzureMigrateInstaller-Server-Public>
.\AzureMigrateInstaller.ps1
```

- For Azure Government:

```
PS C:\Users\Administrators\Desktop\AzureMigrateInstaller-Server-
USGov>.\AzureMigrateInstaller.ps1
```

The script will launch the appliance web application when it finishes successfully.

If you come across any issues, you can access the script logs at `C:\ProgramData\Microsoft Azure\Logs\AzureMigrateScenarioInstaller_Timestamp.log` for troubleshooting.

Verify appliance access to Azure

Make sure that the appliance can connect to Azure URLs for [public](#) and [government](#) clouds.

4. Configure the appliance

Set up the appliance for the first time.

1. Open a browser on any machine that can connect to the appliance and open the URL of the appliance web app: `https://appliance name or IP address: 44368`.

Alternately, you can open the app from the desktop by clicking the app shortcut.

2. Accept the [license terms](#) and read the third-party information.

Set up prerequisites and register the appliance

In the configuration manager, select [Set up prerequisites](#), and then complete these steps:

1. **Connectivity:** The appliance checks that the server has internet access. If the server uses a proxy:

- Select **Setup proxy** to specify the proxy address (in the form `http://ProxyIPAddress` or `http://ProxyFQDN`, where *FQDN* refers to a *fully qualified domain name*) and listening port.
- Enter credentials if the proxy needs authentication.
- If you have added proxy details or disabled the proxy or authentication, select **Save** to trigger connectivity and check connectivity again.

Only HTTP proxy is supported.

2. **Time sync:** Check that the time on the appliance is in sync with internet time for discovery to work properly.
3. **Install updates and register appliance:** To run auto-update and register the appliance, follow these steps:

Appliance Configuration Manager
Cloud: Public

Azure Migrate appliance helps you discover, assess and migrate **VMware virtual machines (VMs)**. Complete the following steps to initiate discovery. [Learn more](#) about Azure Migrate discovery capabilities.

1. Set up prerequisites

- Check connectivity to Azure
- Check time is in sync with Azure
- Check latest updates and register appliance

Verification of Azure Migrate project key
Please provide the Azure Migrate project key, generated on the portal to register the appliance. When you click on 'Verify', the Azure Migrate project key will be validated. [Learn more](#) about how the Azure Migrate project key is generated.

Register VMware appliance by pasting the key here

Verify

Appliance auto-update status
Auto-update will be run automatically after successful verification of the key.

Azure user login and appliance registration status
You need to Login to your Azure account to proceed. Ensure you have the [required permissions](#) on the Azure account.

Login Logout

Check if VMware Virtual Disk Development Kit is installed

Rerun prerequisites

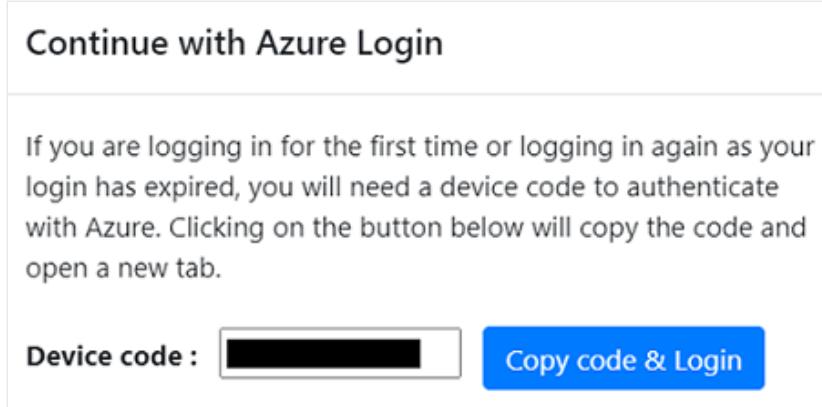
Installation instructions

ⓘ Note

This is a new user experience in Azure Migrate appliance which is available only if you have set up an appliance using the latest OVA/Installer script downloaded from the portal. The appliances which have already been registered will continue seeing the older version of the user experience and will continue to work without any issues.

- For the appliance to run auto-update, paste the project key that you copied from the portal. If you don't have the key, go to **Azure Migrate: Discovery and assessment > Overview > Manage existing appliances**. Select the appliance name you provided when you generated the project key, and then copy the key that's shown.
- The appliance will verify the key and start the auto-update service, which updates all the services on the appliance to their latest versions. When the auto-update has run, you can select **View appliance services** to see the status and versions of the services running on the appliance server.

- c. To register the appliance, you need to select **Login**. In **Continue with Azure Login**, select **Copy code & Login** to copy the device code (you must have a device code to authenticate with Azure) and open an Azure Login prompt in a new browser tab. Make sure you've disabled the pop-up blocker in the browser to see the prompt.



- d. In a new tab in your browser, paste the device code and sign in by using your Azure username and password. Signing in with a PIN isn't supported.

! Note

If you close the sign in tab accidentally without logging in, refresh the browser tab of the appliance configuration manager to display the device code and **Copy code & Login** button.

- e. After you successfully sign in, return to the browser tab that displays the appliance configuration manager. If the Azure user account that you used to sign in has the required permissions for the Azure resources that were created during key generation, appliance registration starts.

After the appliance is successfully registered, to see the registration details, select **View details**.

You can *rerun prerequisites* at any time during appliance configuration to check whether the appliance meets all the prerequisites.

Start continuous discovery

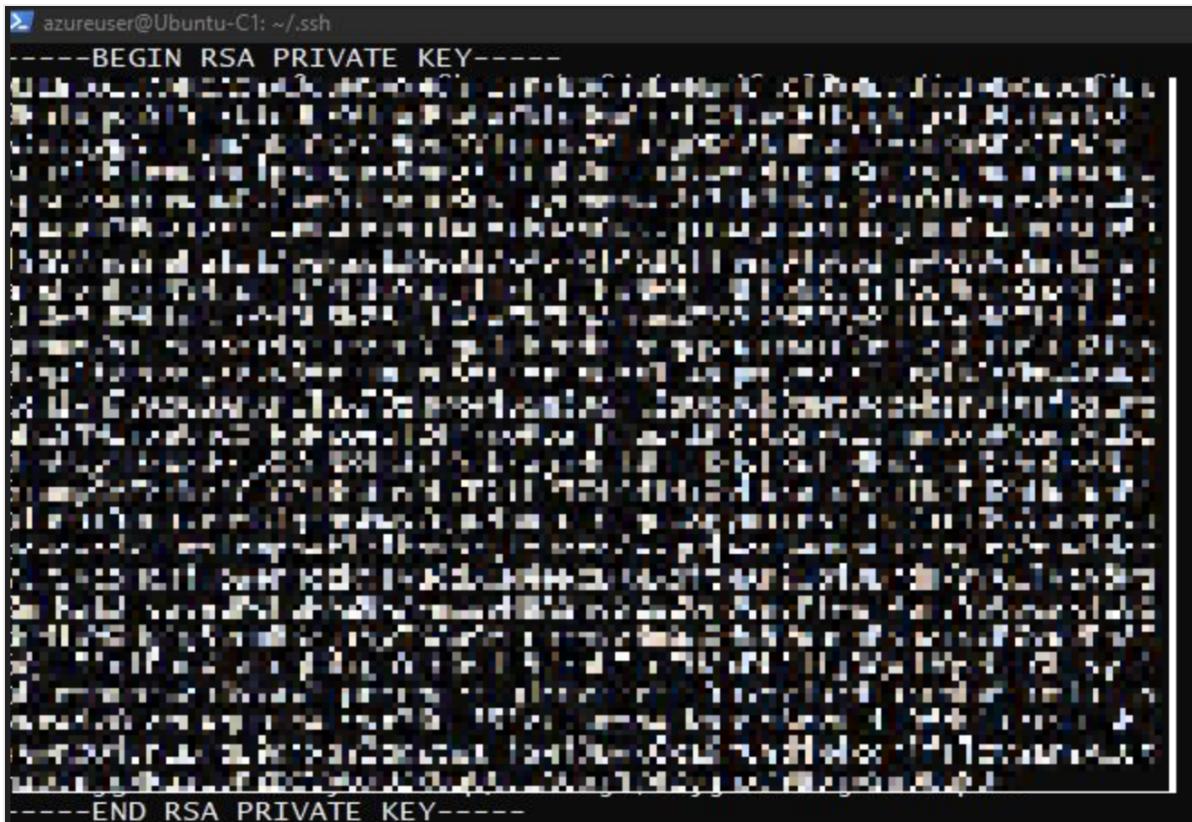
Now, connect from the appliance to the GCP servers to be discovered, and start the discovery.

1. In **Step 1: Provide credentials for discovery of Windows and Linux physical or virtual servers**, select **Add credentials**.
2. For Windows server, select the source type as **Windows Server**, specify a friendly name for credentials, add the username and password. Select **Save**.
3. If you're using password-based authentication for Linux server, select the source type as **Linux Server (Password-based)**, specify a friendly name for credentials, add the username

and password. Select **Save**.

4. If you're using SSH key-based authentication for Linux server, you can select source type as **Linux Server (SSH key-based)**, specify a friendly name for credentials, add the username, browse and select the SSH private key file. Select **Save**.

- Azure Migrate supports the SSH private key generated by ssh-keygen command using RSA, DSA, ECDSA, and ed25519 algorithms.
- Currently Azure Migrate doesn't support passphrase-based SSH key. Use an SSH key without a passphrase.
- Currently Azure Migrate doesn't support SSH private key file generated by PuTTY.
- Azure Migrate supports OpenSSH format of the SSH private key file as shown below:



The screenshot shows a terminal window with the following text:

```
azureuser@Ubuntu-C1: ~/.ssh
-----BEGIN RSA PRIVATE KEY-----
[REDACTED]
-----END RSA PRIVATE KEY-----
```

5. If you want to add multiple credentials at once, select **Add more** to save and add more credentials.

Note

By default, the credentials will be used to gather data about the installed applications, roles, and features, and also to collect dependency data from Windows and Linux servers, unless you disable the slider to not perform these features (as instructed in the last step).

6. In **Step 2:Provide physical or virtual server details**, select **Add discovery source** to specify the server IP address/FQDN and the friendly name for credentials to connect to the server.

7. You can either **Add single item** at a time or **Add multiple items** in one go. There's also an option to provide server details through **Import CSV**.

- If you choose **Add single item**, you can choose the OS type, specify friendly name for credentials, add server **IP address/FQDN** and select **Save**.
- If you choose **Add multiple items**, you can add multiple records at once by specifying server **IP address/FQDN** with the friendly name for credentials in the text box. Verify** the added records and select **Save**.
- If you choose **Import CSV** (*selected by default*), you can download a CSV template file, populate the file with the server **IP address/FQDN** and friendly name for credentials. You then import the file into the appliance, verify the records in the file and select **Save**.

8. On clicking **Save**, the appliance will try validating the connection to the servers added and show the **Validation status** in the table against each server.

- If validation fails for a server, review the error by clicking on **Validation failed** in the **Status** column of the table. Fix the issue, and validate again.
- To remove a server, select **Delete**.

9. You can **revalidate** the connectivity to servers anytime before starting the discovery.

10. Before initiating discovery, you can choose to disable the slider to not perform software inventory and agentless dependency analysis on the added servers. You can change this option at any time.

Note: After server discovery, appliance will automatically initiate software inventory and validate prerequisites for agentless dependency analysis on discovered servers.

Disable the slider if you don't want to perform these features (*you can change this later*)

Click on the button below to initiate discovery. After the discovery is complete, you can check the discovery status of the physical or virtual servers in the table above. [Learn more](#) about the metadata collected during discovery.

Start discovery

11. To perform discovery of SQL Server instances and databases, you can add additional credentials (Windows domain/non-domain, SQL authentication credentials) and the appliance will attempt to automatically map the credentials to the SQL servers. If you add domain credentials, the appliance will authenticate the credentials against Active Directory of the domain to prevent any user accounts from locking out. To check validation of the domain credentials, follow these steps:

- In the configuration manager credentials table, see **Validation status** for domain credentials. Only the domain credentials are validated.
- If validation fails, you can select a **Failed** status to see the validation error. Fix the issue, and then select **Revalidate credentials** to reattempt validation of the credentials.

Start discovery

Click **Start discovery**, to kick off discovery of the successfully validated servers. After the discovery has been successfully initiated, you can check the discovery status against each server in the table.

How discovery works

- It takes approximately 2 minutes to complete discovery of 100 servers and their metadata to appear in the Azure portal.
- [Software inventory](#) (discovery of installed applications) is automatically initiated when the discovery of servers is finished.
- [Software inventory](#) identifies the SQL Server instances that are running on the servers. Using the information it collects, the appliance attempts to connect to the SQL Server instances through the Windows authentication credentials or the SQL Server authentication credentials that are provided on the appliance. Then, it gathers data on SQL Server databases and their properties. The SQL Server discovery is performed once every 24 hours.
- Appliance can connect to only those SQL Server instances to which it has network line of sight, whereas software inventory by itself might not need network line of sight.
- The time taken for discovery of installed applications depends on the number of discovered servers. For 500 servers, it takes approximately one hour for the discovered inventory to appear in the Azure Migrate project in the portal.
- During software inventory, the added server credentials are iterated against servers and validated for agentless dependency analysis. When the discovery of servers is finished, in the portal, you can enable agentless dependency analysis on the servers. Only the servers on which validation succeeds can be selected to enable [agentless dependency analysis](#).
- SQL Server instances and databases data begin to appear in the portal within 24 hours after you start discovery.
- By default, Azure Migrate uses the most secure way of connecting to SQL instances that is, Azure Migrate encrypts communication between the Azure Migrate appliance and the source SQL Server instances by setting the TrustServerCertificate property to `true`. Additionally, the transport layer uses SSL to encrypt the channel and bypass the certificate chain to validate trust. Hence, the appliance server must be set up to trust the certificate's root authority. However, you can modify the connection settings, by selecting [Edit SQL Server connection properties](#) on the appliance. [Learn more](#) to understand what to choose.

Step 3: Provide server credentials to perform software inventory, agentless dependency analysis, discovery of SQL Server instances and databases and discovery of ASP.NET web apps in your VMware environment.

I don't need to perform these features (you can change this later)

You can add multiple credentials and the appliance will attempt to automatically map the credentials to the servers.

If you add domain credentials, appliance will authenticate the credentials against Active Directory of the domain to prevent any user accounts from locking out.

[Learn more](#) about how to provide credentials and how we handle them.

[Add credentials](#)

#	Credentials type	Friendly name	Username	Status	Edit
1		SQL Server connection properties			Edit
2					Edit
3					Edit
4					Edit
5					Edit

You can revalidate the added domain credentials by clicking on the button below.

[Revalidate credentials](#)

You can edit SQL Server connection properties by clicking on button below. [Learn more](#)

[Edit properties](#)

Verify servers in the portal

After discovery finishes, you can verify that the servers appear in the portal.

1. Open the Azure Migrate dashboard.
2. In **Servers, databases and web apps > Azure Migrate: Discovery and assessment** page, click the icon that displays the count for **Discovered servers**.

Next steps

- [Assess GCP servers](#) for migration to Azure VMs.
- [Review the data](#) that the appliance collects during discovery.

Tutorial: Assess Google Cloud Platform (GCP) VM instances for migration to Azure

Article • 06/29/2023

As part of your migration journey to Azure, you assess your on-premises workloads to determine cloud readiness, identify risks, and estimate costs and complexity.

This article shows you how to assess Google Cloud Platform (GCP) VM instances for migration to Azure, using the Azure Migrate: Discovery and assessment tool.

In this tutorial, you learn how to:

- Run an assessment based on server metadata and configuration information.
- Run an assessment based on performance data.

ⓘ Note

Tutorials show the quickest path for trying out a scenario and using default options where possible.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Prerequisites

- Before you follow the steps in this tutorial, complete the first tutorial in this series to [discover your on-premises inventory](#).

Decide which assessment to run

Decide whether you want to run an assessment using sizing criteria based on server configuration data/metadata that's collected as-is on-premises or based on performance data.

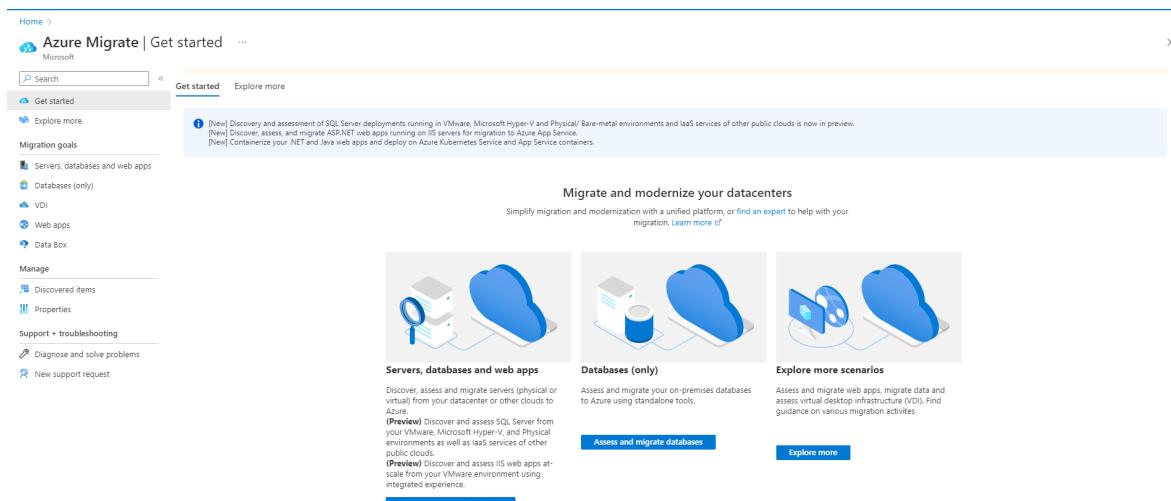
Assessment	Details	Recommendation
------------	---------	----------------

Assessment	Details	Recommendation
As-is on-premises	Assess based on server configuration data/metadata.	<p>Recommended Azure VM size is based on the on-premises VM size.</p> <p>The recommended Azure disk type is based on what you select in the storage type setting in the assessment.</p>
Performance-based	Assess based on collected performance data.	<p>Recommended Azure VM size is based on CPU and memory utilization data.</p> <p>The recommended disk type is based on the IOPS and throughput of the on-premises disks.</p>

Run an assessment

Run an assessment as follows:

1. a. On the **Get started** page > **Servers, databases and web apps**, select **Discover, assess and migrate**.



2. In **Azure Migrate: Discovery and assessment**, select **Assess**.

Microsoft

Search (Ctrl+ /) Refresh

Last refreshed at: 2/8/2021, 7:57:21 PM (Click on "Refresh" to update the page)

Migration goals

- Windows, Linux and SQL Server
- SQL Server (only)
- VDI
- Web Apps
- Data Box

Manage

- Discovered items

Support + troubleshooting

- New support request

Assessment tools (Add more tools)

Azure Migrate: Discovery and assessment

Discover Dependency analysis (Preview) Assess Overview

Discovered servers: 3

OS distribution	Count
Windows	3
Linux	0
Unknown	0

Servers running SQL Server: 0

Dependency enabled machines: 0 Machines (3 total)

Notifications: 2 Groups, 0 Appliances, 3

Next step: Start migrating your servers or optionally you can refine your application grouping with dependency analysis

3. In **Assess servers > Assessment type**, select **Azure VM**.

4. In **Discovery source**:

- If you discovered servers using the appliance, select **Servers discovered from Azure Migrate appliance**.
- If you discovered servers using an imported CSV file, select **Imported servers**.

5. Select **Edit** to review the assessment properties.

Home > Azure Migrate > Create assessment ...

Basics Select servers to assess Review + create assessment

An assessment is created on a group of servers that you migrate together. Assessment helps you determine the Azure readiness of your Windows, Linux and SQL servers running on-premises or on any cloud. You can assess the servers discovered via the Azure Migrate appliance as well as the servers imported into Azure Migrate. [Learn more](#).

Assessment details

Assessment type *	Azure VM	Help me choose
Discovery source *	Servers discovered from Azure Migrate appliance	

Assessment properties (Showing 4 of 14) **Edit**

Sizing criteria	Performance-based
Target location	Australia Southeast
Reserved capacity (compute)	3 years reserved
Azure Hybrid Benefit	Yes

< Previous Next >

6. In **Assessment properties > Target Properties**, do the following:

- In **Target location**, specify the Azure region to which you want to migrate.

- Size and cost recommendations are based on the location that you specify. Once you change the target location from default, you'll be prompted to specify **Reserved Instances** and **VM series**.
- In Azure Government, you can target assessments in [these regions](#).
- In **Storage type**,
 - If you want to use performance-based data in the assessment, select **Automatic** for Azure Migrate to recommend a storage type, based on the disk IOPS and throughput.
 - Alternatively, select the storage type you want to use for VM when you migrate it.
- In **Savings options (compute)**, specify the savings option that you want the assessment to consider, helping to optimize your Azure compute cost.
 - [Azure reservations](#) (1 year or 3 year reserved) are a good option for the most consistently running resources.
 - [Azure Savings Plan](#) (1 year or 3 year savings plan) provide additional flexibility and automated cost optimization. Ideally post migration, you could use Azure reservation and savings plan at the same time (reservation will be consumed first), but in the Azure Migrate assessments, you can only see cost estimates of 1 savings option at a time.
 - When you select 'None', the Azure compute cost is based on the Pay as you go rate or based on actual usage.
 - You need to select pay-as-you-go in offer/licensing program to be able to use Reserved Instances or Azure Savings Plan. When you select any savings option other than 'None', the 'Discount (%)' and 'VM uptime' properties aren't applicable.

7. In VM Size:

- In **Sizing criteria**, select if you want to base the assessment on server configuration data/metadata, or on performance-based data. If you use performance data:
 - In **Performance history**, indicate the data duration on which you want to base the assessment.
 - In **Percentile utilization**, specify the percentile value you want to use for the performance sample.
- In **VM Series**, specify the Azure VM series you want to consider.
 - If you're using performance-based assessment, Azure Migrate suggests a value for you.
 - Tweak the settings as needed. For example, if you don't have a production environment that needs A-series VMs in Azure, you can exclude A-series from the list of series.

- In **Comfort factor**, indicate the buffer you want to use during the assessment. This accounts for issues like seasonal usage, short performance history, and likely increases during future usage. For example, if you use a comfort factor of two:

Component	Effective utilization	Add comfort factor (2.0)
Cores	2	4
Memory	8 GB	16 GB

8. In **Pricing**:

- In **Offer**, specify the [Azure offer](#) if you're enrolled. The assessment estimates the cost for that offer.
- In **Currency**, select the billing currency for your account.
- In **Discount (%)**, add any subscription-specific discounts you receive on top of the Azure offer. The default setting is 0%.
- In **VM Uptime**, specify the duration (days per month/hour per day) for which the VMs will run.
 - This is useful for Azure VMs that won't run continuously.
 - Cost estimates are based on the duration specified.
 - Default is 31 days per month/24 hours per day.
- In **EA Subscription**, specify whether to take an Enterprise Agreement (EA) subscription discount into account for cost estimation.
- In **Azure Hybrid Benefit**, specify whether you already have a Windows Server license. If you do and they're covered with active Software Assurance or Windows Server Subscriptions, you can apply for the [Azure Hybrid Benefit](#) when you bring licenses to Azure.

9. Select **Save** if you make changes.

10. In **Assess Servers**, select **Next**.

11. In **Select servers to assess > Assessment name**, specify a name for the assessment.

12. In **Select or create a group > Create New** and specify a group name.

13. Select the appliance, and select the VMs you want to add to the group. Select **Next**.

14. In **Review + create assessment**, review the assessment details, and select **Create Assessment** to create the group and run the assessment.

15. After the assessment is created, view it in **Servers > Azure Migrate: Discovery and assessment > Assessments**.

16. Select **Export assessment** to download it as an Excel file.

! Note

For performance-based assessments, we recommend that you wait at least a day after starting discovery before you create an assessment. This provides time to collect performance data with higher confidence. Ideally, after you start discovery, wait for the performance duration you specify (day/week/month) for a high-confidence rating.

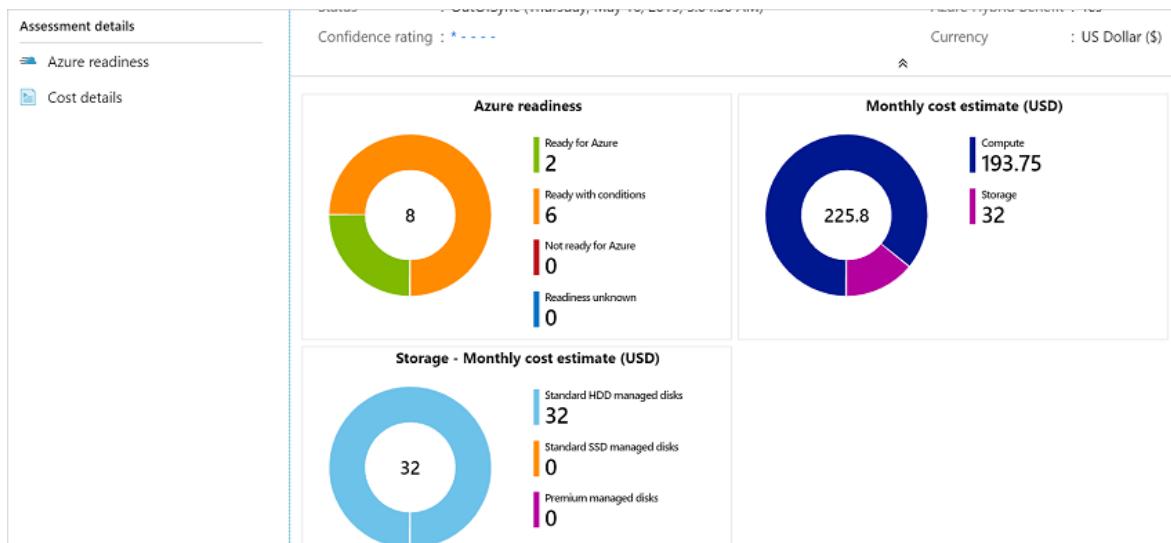
Review an assessment

An assessment describes:

- **Azure readiness:** Whether VMs are suitable for migration to Azure.
- **Monthly cost estimation:** The estimated monthly compute and storage costs for running the VMs in Azure.
- **Monthly storage cost estimation:** Estimated costs for disk storage after migration.

To view an assessment:

1. In **Servers, databases and web apps > Azure Migrate: Discovery and assessment**, select the number next to **Assessments**.
2. In **Assessments**, select an assessment to open it. As an example (estimations and costs, for example, only):



3. Review the assessment summary. You can also edit the assessment properties or recalculate the assessment.

Review readiness

1. Select **Azure readiness**.

2. In **Azure readiness**, review the VM status:

- **Ready for Azure**: Used when Azure Migrate recommends a VM size and cost estimates, for VMs in the assessment.
- **Ready with conditions**: Shows issues and suggested remediation.
- **Not ready for Azure**: Shows issues and suggested remediation.
- **Readiness unknown**: Used when Azure Migrate can't assess readiness, because of data availability issues.

3. Select an **Azure readiness** status. You can view the VM readiness details. You can also drill down to see VM details, including compute, storage, and network settings.

Review cost estimates

The assessment summary shows the estimated compute and storage cost of running VMs in Azure.

1. Review the monthly total costs. Costs are aggregated for all VMs in the assessed group.

- Cost estimates are based on the size recommendations for a server, its disks, and its properties.
- Estimated monthly costs for compute and storage are shown.
- The cost estimation is for running the on-premises VMs on Azure VMs. The estimation doesn't consider PaaS or SaaS costs.

2. Review monthly storage costs. The view shows the aggregated storage costs for the assessed group, split over different types of storage disks.

3. You can drill down to see cost details for specific VMs.

Review confidence rating

Azure Migrate assigns a confidence rating to performance-based assessments. Rating is from one star (lowest) to five stars (highest).

Assess servers		Columns							
		Search to filter assessments							
NAME	GROUP	STATUS	MACHINES	LOCATION	SIZING CRITERION	CONFIDENCE RATING			
assessment_5_16_2019_17_34_29	day2	OutOfSync	0	North Europe	Performance-based	★★★★	★	★	★
assessment_5_20_2019_17_42_6	Mygroup	Ready	6	North Europe	Performance-based	★★★★	★	★	★
assessment_5_22_2019_22_56_13	Day2-group	OutDated	14	North Europe	Performance-based	★★★★	★	★	★

The confidence rating helps you estimate the reliability of size recommendations in the assessment. The rating is based on the availability of data points needed to compute the assessment.

Note

Confidence ratings aren't assigned if you create an assessment based on a CSV file.

Confidence ratings are as follows.

Data point availability	Confidence rating
0%-20%	1 star
21%-40%	2 stars
41%-60%	3 stars
61%-80%	4 stars
81%-100%	5 stars

[Learn more about confidence ratings.](#)

Next steps

- Find server dependencies using [dependency mapping](#).
- Set up [agent-based](#) dependency mapping.

Discover, assess, and migrate Google Cloud Platform (GCP) VMs to Azure

Article • 01/15/2024

This tutorial shows you how to discover, assess, and migrate Google Cloud Platform (GCP) virtual machines (VMs) to Azure VMs, using Azure Migrate: Server Assessment and Migration and modernization tools.

In this tutorial, you will learn how to:

- ✓ Verify prerequisites for migration.
- ✓ Prepare Azure resources with the Migration and modernization tool. Set up permissions for your Azure account and resources to work with Azure Migrate.
- ✓ Prepare GCP VM instances for migration.
- ✓ Add the Migration and modernization tool in the Azure Migrate hub.
- ✓ Set up the replication appliance and deploy the configuration server.
- ✓ Install the Mobility service on GCP VMs you want to migrate.
- ✓ Enable replication for VMs.
- ✓ Track and monitor the replication status.
- ✓ Run a test migration to make sure everything's working as expected.
- ✓ Run a full migration to Azure.

If you don't have an Azure subscription, create a [free account](#) before you begin.

Discover and assess

Before you migrate to Azure, we recommend that you perform a VM discovery and migration assessment. This assessment helps right-size your GCP VMs for migration to Azure, and estimate potential Azure run costs.

Set up an assessment as follows:

1. Follow the [tutorial](#) to set up Azure and prepare your GCP VMs for an assessment.

Note that:

- Azure Migrate uses password authentication when discovering GCP VM instances. GCP instances don't support password authentication by default. Before you can discover, you need to enable password authentication.
 - For Windows machines, allow WinRM port 5985 (HTTP). This allows remote WMI calls.
 - For Linux machines:

- a. Sign in to each Linux machine.
 - b. Open the `sshd_config` file: `vi /etc/ssh/sshd_config`
 - c. In the file, locate the **PasswordAuthentication** line, and change the value to **yes**.
 - d. Save the file and close it. Restart the ssh service.
- If you are using a root user to discover your Linux VMs, ensure root login is allowed on the VMs.
 - a. Sign into each Linux machine
 - b. Open the `sshd_config` file: `vi /etc/ssh/sshd_config`
 - c. In the file, locate the **PermitRootLogin** line, and change the value to **yes**.
 - d. Save the file and close it. Restart the ssh service.
2. Then, follow this [tutorial](#) to set up an Azure Migrate project and appliance to discover and assess your GCP VMs.

Although we recommend that you try out an assessment, performing an assessment isn't a mandatory step to be able to migrate VMs.

Prerequisites

- Ensure that the GCP VMs you want to migrate are running a supported OS version. GCP VMs are treated like physical machines for the purpose of the migration. Review the [supported operating systems and kernel versions](#) for the physical server migration workflow. You can use standard commands like `hostnamectl` or `uname -a` to check the OS and kernel versions for your Linux VMs. We recommend you perform a test migration to validate if the VM works as expected before proceeding with the actual migration.
- Make sure your GCP VMs comply with the [supported configurations](#) for migration to Azure.
- Verify that the GCP VMs that you replicate to Azure comply with [Azure VM requirements](#).
- There are some changes needed on the VMs before you migrate them to Azure.
 - For some operating systems, Azure Migrate makes these changes automatically.
 - It's important to make these changes before you begin migration. If you migrate the VM before you make the change, the VM might not boot up in Azure. Review [Windows](#) and [Linux](#) changes you need to make.

Prepare Azure resources for migration

Prepare Azure for migration with the Migration and modernization tool.

Task	Details
Create an Azure Migrate project	Your Azure account needs Contributor or Owner permissions to create a new project .
Verify permissions for your Azure account	Your Azure account needs permissions to create a VM, and write to an Azure managed disk.

Assign permissions to create project

1. In the Azure portal, open the subscription, and select **Access control (IAM)**.
2. In **Check access**, find the relevant account, and click it to view permissions.
3. You should have **Contributor** or **Owner** permissions.
 - If you just created a free Azure account, you're the owner of your subscription.
 - If you're not the subscription owner, work with the owner to assign the role.

Assign Azure account permissions

Assign the Virtual Machine Contributor role to the Azure account. This provides permissions to:

- Create a VM in the selected resource group.
- Create a VM in the selected virtual network.
- Write to an Azure managed disk.

Create an Azure network

[Set up](#) an Azure virtual network (VNet). When you replicate to Azure, the Azure VMs that are created are joined to the Azure VNet that you specify when you set up migration.

Prepare GCP instances for migration

To prepare for GCP to Azure migration, you need to prepare and deploy a replication appliance for migration.

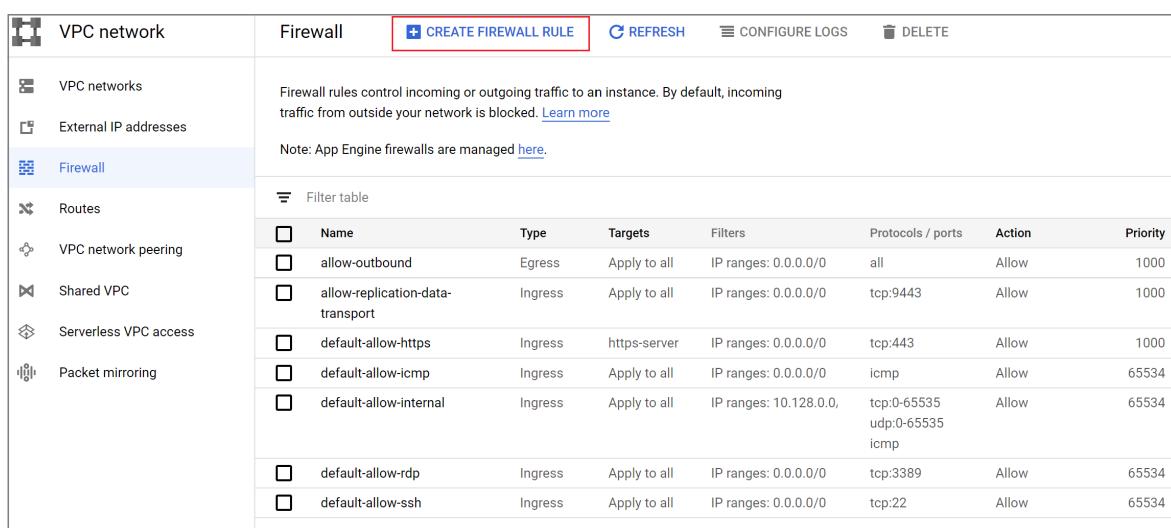
Prepare a machine for the replication appliance

The Migration and modernization tool uses a replication appliance to replicate machines to Azure. The replication appliance runs the following components.

- **Configuration server:** The configuration server coordinates communications between the GCP VMs and Azure, and manages data replication.
- **Process server:** The process server acts as a replication gateway. It receives replication data, optimizes it with caching, compression, and encryption, and sends it to a cache storage account in Azure.

Prepare for appliance deployment as follows:

- Set up a separate GCP VM to host the replication appliance. This instance must be running Windows Server 2012 R2 or Windows Server 2016. [Review](#) the hardware, software, and networking requirements for the appliance.
- The appliance shouldn't be installed on a source VM that you want to replicate or on the Azure Migrate discovery and assessment appliance you may have installed before. It should be deployed on a different VM.
- The source GCP VMs to be migrated should have a network line of sight to the replication appliance. Configure necessary Firewall rules to enable this. It is recommended that the replication appliance is deployed in the same VPC network as the source VMs to be migrated. If the replication appliance needs to be in a different VPC, the VPCs need to be connected through VPC peering.
- The source GCP VMs communicate with the replication appliance on ports HTTPS 443 (control channel orchestration) and TCP 9443 (data transport) inbound for replication management and replication data transfer. The replication appliance in turn orchestrates and sends replication data to Azure over port HTTPS 443 outbound. To configure these rules, edit the security group inbound/outbound rules with the appropriate ports and source IP information.



The screenshot shows the Google Cloud Firewall interface for a VPC network. The top navigation bar includes 'VPC network', 'Firewall' (selected), 'CREATE FIREWALL RULE', 'REFRESH', 'CONFIGURE LOGS', and 'DELETE'. The left sidebar lists 'VPC networks', 'External IP addresses', and 'Firewall' (selected). The main content area shows a table of firewall rules:

Name	Type	Targets	Filters	Protocols / ports	Action	Priority
allow-outbound	Egress	Apply to all	IP ranges: 0.0.0.0/0	all	Allow	1000
allow-replication-data-transport	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:9443	Allow	1000
default-allow-https	Ingress	https-server	IP ranges: 0.0.0.0/0	tcp:443	Allow	1000
default-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow	65534
default-allow-internal	Ingress	Apply to all	IP ranges: 10.128.0.0/16	tcp:0-65535 udp:0-65535 icmp	Allow	65534
default-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow	65534
default-allow-ssh	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow	65534

 VPC network <ul style="list-style-type: none">  VPC networks  External IP addresses  Firewall  Routes  VPC network peering  Shared VPC  Serverless VPC access  Packet mirroring 	← Firewall rule details <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 5px;">  EDIT  DELETE </div> <div style="margin-top: 10px;"> allow-replication-data-transport </div> <div style="margin-top: 10px;"> Logs  Off view </div> <div style="margin-top: 10px;"> Network default </div> <div style="margin-top: 10px;"> Priority 1000 </div> <div style="margin-top: 10px;"> Direction Ingress </div> <div style="margin-top: 10px;"> Action on match Allow </div> <div style="margin-top: 10px;"> Source filters </div> <div style="display: flex; align-items: center;"> IP ranges <input style="width: 150px; margin-left: 10px;" type="text" value="0.0.0.0/0"/> </div> <div style="margin-top: 10px;"> Protocols and ports tcp:9443 </div>
--	---

- The replication appliance uses MySQL. Review the [options](#) for installing MySQL on the appliance.
- Review the Azure URLs required for the replication appliance to access [public](#) and [government](#) clouds.

Set up the replication appliance

The first step of migration is to set up the replication appliance. To set up the appliance for GCP VMs migration, you must download the installer file for the appliance, and then run it on the [VM you prepared](#).

Download the replication appliance installer

1. In the Azure Migrate project > **Servers, databases and web apps**, in **Migration and modernization**, select **Discover**.

 **Azure Migrate - Servers**
Microsoft

«

 Overview

 Migration goals

 Servers

 Databases

 Data Box

 Manage

 Discovered items

 Support + troubleshooting

 New support request

 Refresh

Last refreshed at: 6/20/2019, 8:28:41 PM (Click on "Refresh" to update the

Assessment tools

 **Azure Migrate: Server Assessment**

 Discover  Assess  Overview

Quick start

1: Discover
Click 'Discover' to start discovering your on-premises machines.

2: Assess
Once your on-premises machines are discovered, click "Assess" to create assessments.

Add more assessment tools? [Click here.](#)

Migration tools

 **Azure Migrate: Server Migration**

 Discover  Replicate  Migrate  Overview

Quick start

1: Discover
Click 'Discover' to start discovering your on-premises machines.

2: Replicate
Once your on-premises machines are discovered, click "Replicate" to start replicating the discovered machines.

3: Migrate
Once your machines have replicated, click "Migrate" to migrate your machines.

Add more migration tools? [Click here.](#)

2. In **Discover machines** > **Are your machines virtualized?**, click **Not virtualized/Other**.
3. In **Target region**, select the Azure region to which you want to migrate the machines.
4. Select **Confirm that the target region for migration is <region-name>**.
5. Click **Create resources**. This creates an Azure Site Recovery vault in the background.

- If you've already set up migration with the Migration and modernization tool, the target option can't be configured, since resources were set up previously.
- You can't change the target region for this project after clicking this button.
- To migrate your VMs to a different region, you'll need to create a new/different Azure Migrate project.

! Note

If you selected private endpoint as the connectivity method for the Azure Migrate project when it was created, the Recovery Services vault will also be configured for private endpoint connectivity. Ensure that the private endpoints are reachable from the replication appliance: [Learn more](#)

6. In **Do you want to install a new replication appliance?**, select **Install a replication appliance**.
7. In **Download and install the replication appliance software**, download the appliance installer, and the registration key. You need to the key in order to register the appliance. The key is valid for five days after it's downloaded.

Discover machines

Are your machines virtualized? [!](#)
Not virtualized / Other

Target region [!](#)
(US) Central US

Do you want to install a new replication appliance or scale-out existing setup?
Install a replication appliance [Help me choose](#)

The replication appliance (Configuration Server) is a virtual appliance that is deployed on-premises. The replication appliance coordinates and manages replication for the servers that are being migrated. Follow the steps outlined below to set up and configure the replication appliance

1. Download and install the replication appliance software.
Create a new Windows Server 2016 machine by following the [Configuration Server sizing guidelines](#).
[Download](#) the replication appliance software installer and use it to complete installation of the replication appliance software on the newly created Windows Server 2016 machine.

2. Configure the replication appliance and register it to the Azure Migrate project
Download the registration key file and use it to register the replication appliance to this project. The replication appliance installer will ask for a registration key.
[Download](#)

8. Copy the appliance setup file and key file to the Windows Server 2016 or Windows Server 2012 GCP VM you created for the replication appliance.
9. Run the replication appliance setup file, as described in the next procedure.
- 9.1. Under **Before You Begin**, select **Install the configuration server and process server**, and then select **Next**.

- 9.2 In **Third-Party Software License**, select **I accept the third-party license agreement**, and then select **Next**.
- 9.3 In **Registration**, select **Browse**, and then go to where you put the vault registration key file. Select **Next**.
- 9.4 In **Internet Settings**, select **Connect to Azure Site Recovery without a proxy server**, and then select **Next**.
- 9.5 The **Prerequisites Check** page runs checks for several items. When it's finished, select **Next**.
- 9.6 In **MySQL Configuration**, provide a password for the MySQL DB, and then select **Next**.
- 9.7 In **Environment Details**, select **No**. You don't need to protect your VMs. Then, select **Next**.
- 9.8 In **Install Location**, select **Next** to accept the default.
- 9.9 In **Network Selection**, select **Next** to accept the default.
- 9.10 In **Summary**, select **Install**.
- 9.11 **Installation Progress** shows you information about the installation process. When it's finished, select **Finish**. A window displays a message about a reboot. Select **OK**.
- 9.12 Next, a window displays a message about the configuration server connection passphrase. Copy the passphrase to your clipboard and save the passphrase in a temporary text file on the source VMs. You'll need this passphrase later, during the mobility service installation process.
10. After the installation completes, the Appliance configuration wizard will be launched automatically (You can also launch the wizard manually by using the **cspconfigtool** shortcut that is created on the desktop of the appliance). In this tutorial, we'll be manually installing the Mobility Service on source VMs to be replicated, so create a dummy account in this step and proceed. You can provide the following details for creating the dummy account - "guest" as the friendly name, "username" as the username, and "password" as the password for the account. You will be using this dummy account in the Enable Replication stage.

 **3. Finalize registration**
Prepare for replication by finalizing registration for the replication appliance (Configuration Server). Select the replication appliance from the drop down to finalize registration for it.

* Select Configuration Server  Registration finalized

Install the Mobility service agent

A Mobility service agent must be pre-installed on the source GCP VMs to be migrated before you can initiate replication. The approach you choose to install the Mobility service agent may depend on your organization's preferences and existing tools, but be aware that the "push" installation method built into Azure Site Recovery is not currently supported. Approaches you may want to consider:

- [System Center Configuration Manager](#)
- [Arc for Servers and Custom Script Extensions](#)
- [Manual installation](#)

1. Extract the contents of the installer tarball to a local folder (for example /tmp/MobSvclnstaller) on the GCP VM, as follows:

```
mkdir /tmp/MobSvclnstaller
tar -C /tmp/MobSvclnstaller -xvf <Installer tarball>
cd /tmp/MobSvclnstaller
```

2. Run the installer script:

```
sudo ./install -r MS -v VmWare -q -c CSLegacy
```

3. Register the agent with the replication appliance:

```
/usr/local/ASR/Vx/bin/UnifiedAgentConfigurator.sh -i <replication
appliance IP address> -P <Passphrase File Path>
```

Enable replication for GCP VMs

Note

Through the portal, you can add up to 10 VMs for replication at once. To replicate more VMs simultaneously, you can add them in batches of 10.

1. In the Azure Migrate project > **Servers, databases and web apps** > **Migration and modernization**, select **Replicate**.

The screenshot shows the Azure Migrate dashboard. On the left, a sidebar lists 'Migration goals' (Servers, databases and web apps, Databases (only), VDI, Web apps, Data Box), 'Manage' (Discovered items, Properties), and 'Support + troubleshooting' (Diagnose and solve problems, New support request). The main area is divided into 'Assessment tools' and 'Migration tools'.

Assessment tools: Displays 'Azure Migrate: Discovery and assessment' with metrics: Discovered servers (609), OS distribution (Windows 420, Linux 105, Unknown 84), Groups (7), Assessments (24), and Notifications (24). It also shows Dependency analysis progress (85 servers, 609 total) and various migration paths to Azure VM, Azure SQL, Azure App Service (Preview), and Azure VMware Solution (AVS).

Migration tools: Displays 'Azure Migrate: Server Migration' with metrics: Discovered servers (609), Replicating servers (13), and Test migrated servers (1). The 'Replicate' tab is highlighted with a red box.

2. In **Replicate**, > **Source settings** > **Are your machines virtualized?**, select **Not virtualized/Other**.
3. In **On-premises appliance**, select the name of the Azure Migrate appliance that you set up.
4. In **Process Server**, select the name of the replication appliance.
5. In **Guest credentials**, please select the dummy account created previously during the [replication installer setup](#) to install the Mobility service manually (push install is not supported). Then click **Next: Virtual machines**.

Replicate

The screenshot shows the 'Replicate' step of the migration wizard. The top navigation bar includes 'Basics' (selected), 'Virtual machines', 'Target settings', 'Compute', 'Disks', 'Tags', and 'Review + Start replication'.

The 'Basics' tab content includes a note: 'The first step in migrating servers is to replicate them. Once replication completes, you can perform test migration before finally moving the servers to Azure.' Below this is a question: 'Are your machines virtualized? *' with a help icon. A dropdown menu shows 'Physical or other (AWS, GCP, Xen, etc.)'.

Below the dropdown is a field for 'On-premises appliance *' with a help icon, showing 'CONTOSO-0122 (Azure Migrate Appliance)'.

6. In **Virtual Machines**, in **Import migration settings from an assessment?**, leave the default setting **No, I'll specify the migration settings manually**.

7. Check each VM you want to migrate. Then click **Next: Target settings**.

8. In **Target settings**, select the subscription, and target region to which you'll migrate, and specify the resource group in which the Azure VMs will reside after migration.

9. In **Virtual Network**, select the Azure VNet/subnet to which the Azure VMs will be joined after migration.

10. In **Cache storage account**, keep the default option to use the cache storage account that is automatically created for the project. Use the dropdown if you'd like to specify a different storage account to use as the cache storage account for replication.

(!) Note

- If you selected private endpoint as the connectivity method for the Azure Migrate project, grant the Recovery Services vault access to the cache storage account. [Learn more](#)
- To replicate using ExpressRoute with private peering, create a private endpoint for the cache storage account. [Learn more](#)

11. In **Availability options**, select:

- Availability Zone to pin the migrated machine to a specific Availability Zone in the region. Use this option to distribute servers that form a multi-node application tier across Availability Zones. If you select this option, you'll need to specify the Availability Zone to use for each of the selected machine in the Compute tab. This option is only available if the target region selected for the migration supports Availability Zones

- Availability Set to place the migrated machine in an Availability Set. The target Resource Group that was selected must have one or more availability sets in order to use this option.
- No infrastructure redundancy required option if you don't need either of these availability configurations for the migrated machines.

12. In **Disk encryption type**, select:

- Encryption-at-rest with platform-managed key
- Encryption-at-rest with customer-managed key
- Double encryption with platform-managed and customer-managed keys

 **Note**

To replicate VMs with CMK, you'll need to **create a disk encryption set** under the target Resource Group. A disk encryption set object maps Managed Disks to a Key Vault that contains the CMK to use for SSE.

13. In **Azure Hybrid Benefit**:

- Select **No** if you don't want to apply Azure Hybrid Benefit. Then click **Next**.
- Select **Yes** if you have Windows Server machines that are covered with active Software Assurance or Windows Server subscriptions, and you want to apply the benefit to the machines you're migrating. Then click **Next**.

Replicate

Configure settings and target properties for migration.

Region *

Storage account *

Subscription *

Resource group *

VMs on this subscription, wherein SQL Server is running, are being onboarded to the SQL IaaS extension. You can choose to disable the same by following steps listed [here](#)

Register with SQL IaaS extension

Apply Azure Hybrid with an eligible Windows Server license and save up to 49% vs. pay-as-you-go virtual machine costs.

I have a Windows Server license

Virtual network *

Subnet *

Availability options *

Automatically repair replication

Security details

Target VM security type

Disk encryption type

Test Migration

Select the virtual network and subnet for test migration. Network properties can be changed from Compute and Network settings of replicating machine or when test migration is performed.

[Previous](#) [Next](#)

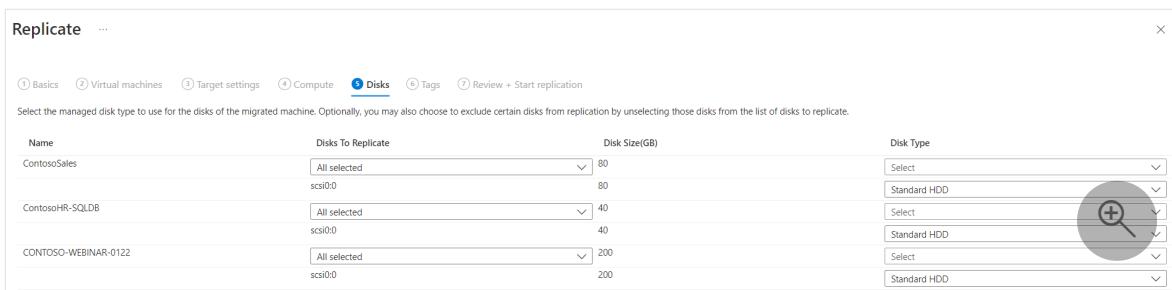
14. In **Compute**, review the VM name, size, OS disk type, and availability configuration (if selected in the previous step). VMs must conform with [Azure requirements](#).

- **VM size:** If you're using assessment recommendations, the VM size dropdown shows the recommended size. Otherwise Azure Migrate picks a size based on the closest match in the Azure subscription. Alternatively, pick a manual size in [Azure VM size](#).
- **OS disk:** Specify the OS (boot) disk for the VM. The OS disk is the disk that has the operating system bootloader and installer.
- **Availability Zone:** Specify the Availability Zone to use.
- **Availability Set:** Specify the Availability Set to use.

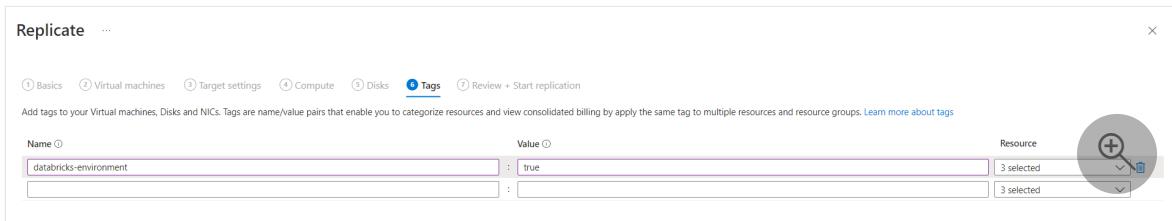
15. In **Disks**, specify whether the VM disks should be replicated to Azure, and select the disk type (standard SSD/HDD or premium managed disks) in Azure. Then click **Next**.

- You can exclude disks from replication.

- If you exclude disks, won't be present on the Azure VM after migration.



16. In **Tags**, choose to add tags to your Virtual machines, Disks, and NICs.



17. In **Review and start replication**, review the settings, and click **Replicate** to start the initial replication for the servers.

(!) Note

You can update replication settings any time before replication starts, **Manage > Replicating machines**. Settings can't be changed after replication starts.

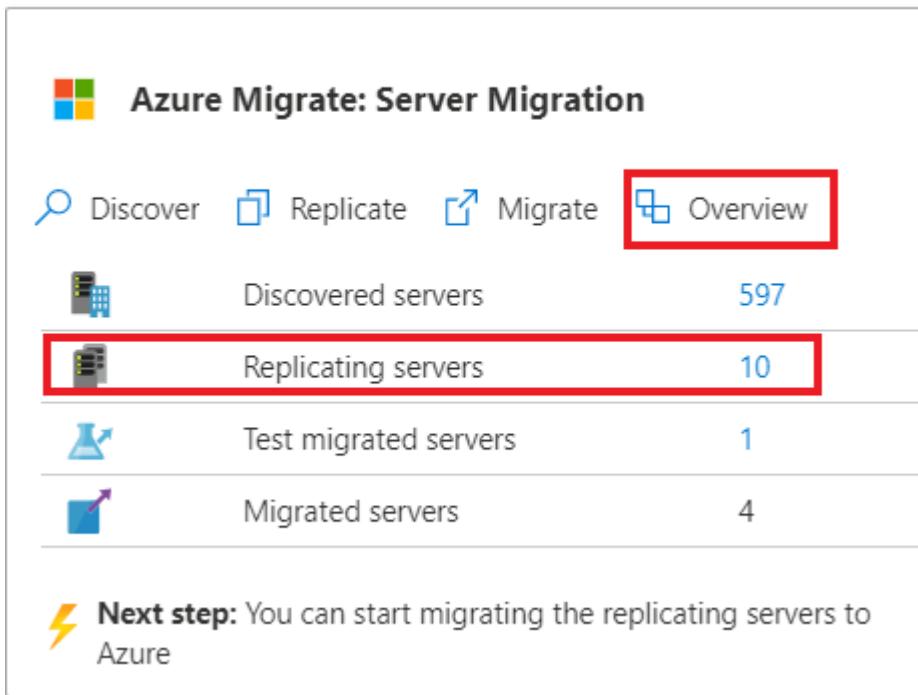
Track and monitor replication status

- When you click **Replicate** a Start Replication job begins.
- When the Start Replication job finishes successfully, the VMs begin their initial replication to Azure.
- After initial replication finishes, delta replication begins. Incremental changes to GCP VM disks are periodically replicated to the replica disks in Azure.

You can track job status in the portal notifications.

You can monitor replication status by clicking on **Replicating servers** in **Migration and modernization**.

Migration tools



The screenshot shows the Azure Migrate: Server Migration interface. At the top, there are four tabs: Discover, Replicate, Migrate, and Overview. The Overview tab is highlighted with a red box. Below the tabs, there are four categories with counts: Discovered servers (597), Replicating servers (10, highlighted with a red box), Test migrated servers (1), and Migrated servers (4). At the bottom, a note says: "⚡ **Next step:** You can start migrating the replicating servers to Azure".

Category	Count
Discovered servers	597
Replicating servers	10
Test migrated servers	1
Migrated servers	4

⚡ **Next step:** You can start migrating the replicating servers to Azure

Run a test migration

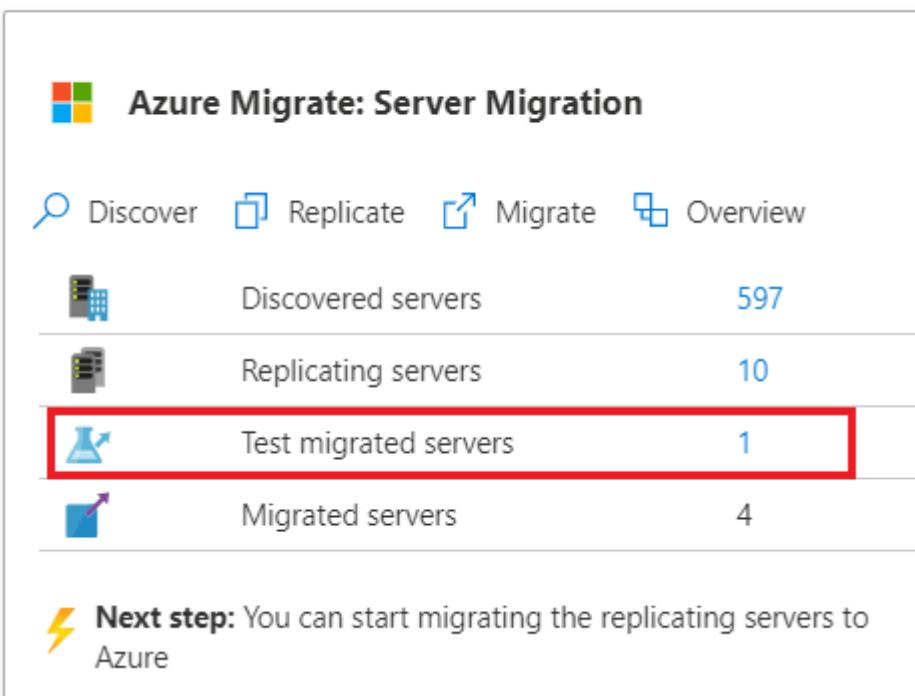
When delta replication begins, you can run a test migration for the VMs, before running a full migration to Azure. The test migration is highly recommended and provides an opportunity to discover any potential issues and fix them before you proceed with the actual migration. It is advised that you do this at least once for each VM before you migrate it.

- Running a test migration checks that migration will work as expected, without impacting the GCP VMs, which remain operational, and continue replicating.
- Test migration simulates the migration by creating an Azure VM using replicated data (usually migrating to a non-production VNet in your Azure subscription).
- You can use the replicated test Azure VM to validate the migration, perform app testing, and address any issues before full migration.

Do a test migration as follows:

1. In **Migration goals > Servers, databases and web apps > Migration and modernization**, select **Test migrated servers**.

Migration tools



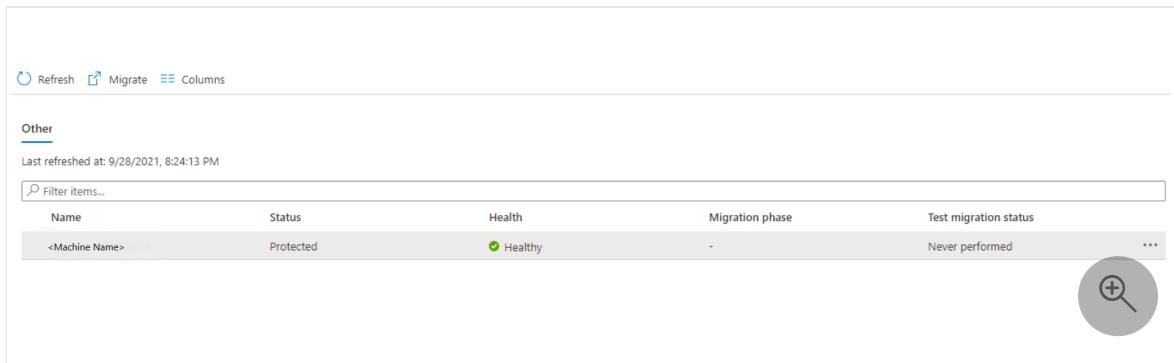
Azure Migrate: Server Migration

Discover Replicate Migrate Overview

	Discovered servers	597
	Replicating servers	10
	Test migrated servers	1
	Migrated servers	4

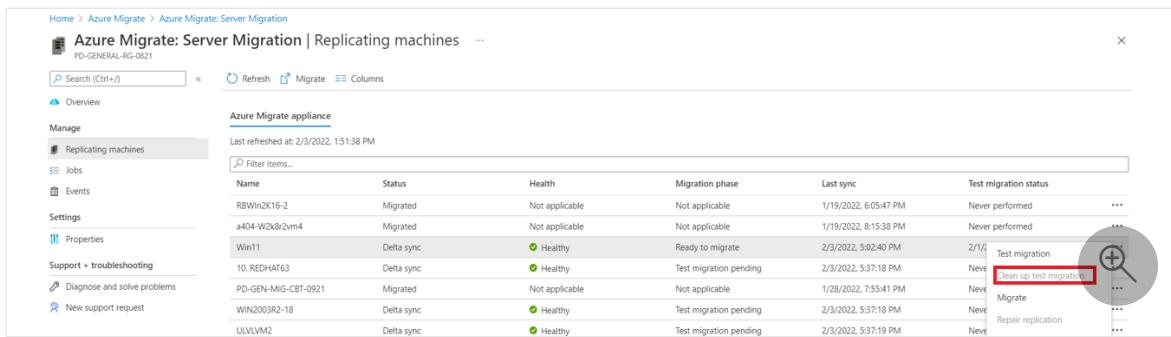
Next step: You can start migrating the replicating servers to Azure

2. Right-click the VM to test, and click **Test migrate**.



Name	Status	Health	Migration phase	Test migration status
<Machine Name>	Protected	Healthy	-	Never performed

3. In **Test Migration**, select the Azure VNet in which the Azure VM will be located after the migration. We recommend you use a non-production VNet.
4. The **Test migration** job starts. Monitor the job in the portal notifications.
5. After the migration finishes, view the migrated Azure VM in **Virtual Machines** in the Azure portal. The machine name has a suffix **-Test**.
6. After the test is done, right-click the Azure VM in **Replicating machines**, and click **Clean up test migration**.



Name	Status	Health	Migration phase	Last sync	Test migration status
RBWin2K16-2	Migrated	Not applicable	Not applicable	1/19/2022, 6:05:47 PM	Never performed
a404-W2k8r2vm4	Migrated	Not applicable	Not applicable	1/19/2022, 8:15:38 PM	Never performed
Win11	Delta sync	Healthy	Ready to migrate	2/3/2022, 5:02:40 PM	2/1/2022, 5:02:40 PM Test migration
10. REDHAT63	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:18 PM	New
PD-GEN-MIG-CBT-0921	Migrated	Not applicable	Not applicable	1/28/2022, 7:55:41 PM	New
WIN2003R2-18	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:18 PM	New
ULVLM2	Delta sync	Healthy	Test migration pending	2/3/2022, 5:37:19 PM	New

ⓘ Note

You can now register your servers running SQL server with SQL VM RP to take advantage of automated patching, automated backup and simplified license management using SQL IaaS Agent Extension.

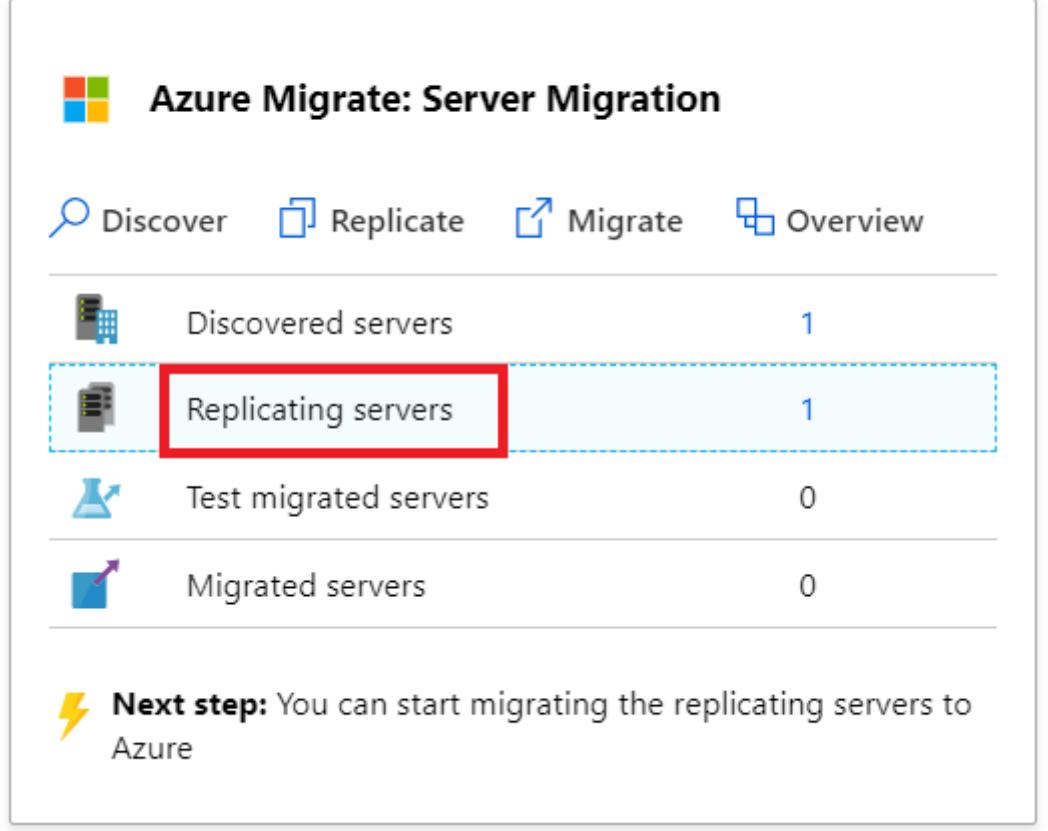
- Select **Manage > Replicating servers > Machine containing SQL server > Compute and Network** and select **yes** to register with SQL VM RP.
- Select Azure Hybrid benefit for SQL Server if you have SQL Server instances that are covered with active Software Assurance or SQL Server subscriptions and you want to apply the benefit to the machines you're migrating.

Migrate GCP VMs

After you've verified that the test migration works as expected, you can migrate the GCP VMs.

1. In the Azure Migrate project > **Servers, databases and web apps > Migration and modernization**, click **Replicating servers**.

Migration tools



The screenshot shows the Azure Migrate: Server Migration interface. At the top, there are four tabs: Discover, Replicate, Migrate, and Overview. Below the tabs, there is a list of server categories with counts:

Category	Count
Discovered servers	1
Replicating servers	1
Test migrated servers	0
Migrated servers	0

At the bottom, there is a note: **Next step:** You can start migrating the replicating servers to Azure.

2. In **Replicating machines**, right-click the VM > **Migrate**.
3. In **Migrate > Shut down virtual machines and perform a planned migration with no data loss**, select **Yes > OK**.

ⓘ Note

Automatic shutdown is not supported while migrating GCP virtual machines.

4. A migration job starts for the VM. You can view the job status by clicking the notification bell icon on the top right of the portal page or by going to the jobs page of the Migration and modernization tool (Click **Overview** on the tool tile > Select **Jobs** from the left menu).
5. After the job finishes, you can view and manage the VM from the Virtual Machines page.

Complete the migration

1. After the migration is done, right-click the VM > **Stop migration**. This does the following:

- Stops replication for the GCP VM.
 - Removes the GCP VM from the **Replicating servers** count in the Migration and modernization tool.
 - Cleans up replication state information for the VM.
2. Verify and [troubleshoot any Windows activation issues on the Azure VM](#).
 3. Perform any post-migration app tweaks, such as updating host names, database connection strings, and web server configurations.
 4. Perform final application and migration acceptance testing on the migrated application now running in Azure.
 5. Cut over traffic to the migrated Azure VM instance.
 6. Update any internal documentation to show the new location and IP address of the Azure VMs.

Post-migration best practices

- For increased resilience:
 - Keep data secure by backing up Azure VMs using the Azure Backup service. [Learn more](#).
 - Keep workloads running and continuously available by replicating Azure VMs to a secondary region with Site Recovery. [Learn more](#).
- For increased security:
 - Lock down and limit inbound traffic access with [Microsoft Defender for Cloud - Just in time administration](#).
 - Manage and govern updates on Windows and Linux machines with [Azure Update Manager](#).
 - Restrict network traffic to management endpoints with [Network Security Groups](#).
 - Deploy [Azure Disk Encryption](#) to help secure disks, and keep data safe from theft and unauthorized access.
 - Read more about [securing IaaS resources](#), and visit the [Microsoft Defender for Cloud](#).
- For monitoring and management:
 - Consider deploying [Azure Cost Management](#) to monitor resource usage and spending.

Troubleshooting / Tips

Question: I cannot see my GCP VM in the discovered list of servers for migration.

Answer: Check if your replication appliance meets the requirements. Make sure Mobility Agent is installed on the source VM to be migrated and is registered the Configuration

Server. Check the firewall rules to enable a network path between the replication appliance and source GCP VMs.

Question: How do I know if my VM was successfully migrated? **Answer:** Post-migration, you can view and manage the VM from the Virtual Machines page. Connect to the migrated VM to validate.

Question: I am unable to import VMs for migration from my previously created Server Assessment results. **Answer:** Currently, we do not support the import of assessment for this workflow. As a workaround, you can export the assessment and then manually select the VM recommendation during the Enable Replication step.

Question: I am getting the error “Failed to fetch BIOS GUID” while trying to discover my GCP VMs. **Answer:** Use root login for authentication and not any pseudo user. If you are not able to use a root user, ensure the required capabilities are set on the user, as per the instructions provided in the [support matrix](#). Also review supported operating systems for GCP VMs.

Question: My replication status is not progressing.

Answer: Check if your replication appliance meets the requirements. Make sure you've enabled the required ports on your replication appliance TCP port 9443 and HTTPS 443 for data transport. Ensure that there are no stale duplicate versions of the replication appliance connected to the same project.

Question: I am unable to Discover GCP Instances using Azure Migrate due to HTTP status code of 504 from the remote Windows management service.

Answer: Make sure to review the Azure migrate appliance requirements and URL access needs. Make sure no proxy settings are blocking the appliance registration.

Question: Do I have to make any changes before I migrate my GCP VMs to Azure?

Answer: You may have to make these changes before migrating your GCP VMs to Azure:

- If you are using cloud-init for your VM provisioning, you may want to disable cloud-init on the VM before replicating it to Azure. The provisioning steps performed by cloud-init on the VM maybe GCP specific and won't be valid after the migration to Azure.
- Review the [prerequisites](#) section to determine whether there are any changes necessary for the operating system before you migrate them to Azure.
- We always recommend you run a test migration before the final migration.

Next steps

Investigate the [cloud migration journey](#) in the Azure Cloud Adoption Framework.

How Defender for Cloud Apps helps protect your Google Cloud Platform (GCP) environment

Article • 01/22/2024

Google Cloud Platform is an IaaS provider that enables your organization to host and manage their entire workloads in the cloud. Along with the benefits of leveraging infrastructure in the cloud, your organization's most critical assets may be exposed to threats. Exposed assets include storage instances with potentially sensitive information, compute resources that operate some of your most critical applications, ports, and virtual private networks that enable access to your organization.

Connecting GCP to Defender for Cloud Apps helps you secure your assets and detect potential threats by monitoring administrative and sign-in activities, notifying on possible brute force attacks, malicious use of a privileged user account, and unusual deletions of VMs.

Main threats

- Abuse of cloud resources
- Compromised accounts and insider threats
- Data leakage
- Resource misconfiguration and insufficient access control

How Defender for Cloud Apps helps to protect your environment

- [Detect cloud threats, compromised accounts, and malicious insiders](#)
- [Use the audit trail of activities for forensic investigations](#)

Control GCP with built-in policies and policy templates

You can use the following built-in policy templates to detect and notify you about potential threats:

[+] [Expand table](#)

Type	Name
Built-in anomaly detection policy	Activity from anonymous IP addresses Activity from infrequent country Activity from suspicious IP addresses Impossible travel Activity performed by terminated user (requires Microsoft Entra ID as IdP) Multiple failed login attempts Unusual administrative activities Multiple delete VM activities Unusual multiple VM creation activities (preview)
Activity policy template	Changes to compute engine resources Changes to StackDriver configuration Changes to storage resources Changes to Virtual Private Network Logon from a risky IP address

For more information about creating policies, see [Create a policy](#).

Automate governance controls

In addition to monitoring for potential threats, you can apply and automate the following GCP governance actions to remediate detected threats:

[+] [Expand table](#)

Type	Action
User governance	<ul style="list-style-type: none">- Require user to reset password to Google (requires connected linked Google Workspace instance)- Suspend user (requires connected linked Google Workspace instance)- Notify user on alert (via Microsoft Entra ID)- Require user to sign in again (via Microsoft Entra ID)- Suspend user (via Microsoft Entra ID)

For more information about remediating threats from apps, see [Governing connected apps](#).

Protect GCP in real time

Review our best practices for [securing and collaborating with external users](#) and [blocking and protecting the download of sensitive data to unmanaged or risky devices](#).

Connect Google Cloud Platform to Microsoft Defender for Cloud Apps

This section provides instructions for connecting Microsoft Defender for Cloud Apps to your existing Google Cloud Platform (GCP) account using the connector APIs. This connection gives you visibility into and control over GCP use. For information about how Defender for Cloud Apps protects GCP, see [Protect GCP](#).

We recommend that you use a dedicated project for the integration and restrict access to the project to maintain stable integration and prevent deletions/modifications of the setup process.

Note

The instructions for connecting your GCP environment for auditing follow [Google's recommendations](#) for consuming aggregated logs. The integration leverages Google StackDriver and will consume additional resources that might impact your billing. The consumed resources are:

- Aggregated export sink – Organization level 
- Pub/Sub topic – GCP project level 
- Pub/Sub subscription – GCP project level 

The Defender for Cloud Apps auditing connection only imports Admin Activity audit logs; Data Access and System Event audit logs are not imported. For more information about GCP logs, see [Cloud Audit Logs](#).

Prerequisites

The integrating GCP user must have the following permissions:

- **IAM and Admin edit** – Organization level
- **Project creation and edit**

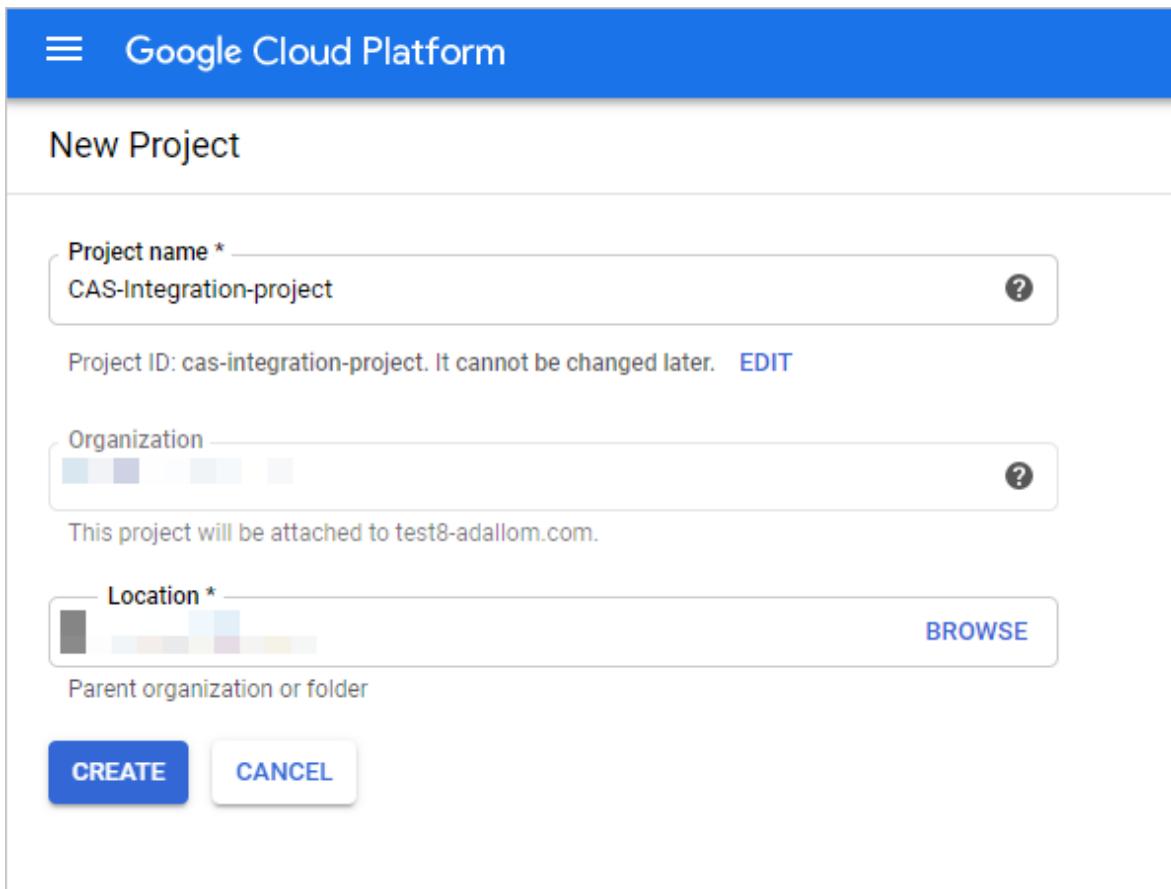
You can connect GCP **Security auditing** to your Defender for Cloud Apps connections to gain visibility into and control over GCP app use.

Configure Google Cloud Platform

Create a dedicated project

Create a dedicated project in GCP under your organization to enable integration isolation and stability

1. Sign in to your GCP portal using your integrating GCP user account.
2. Select **Create Project** to start a new project.
3. In the **New project** screen, name your project and select **Create**.



The screenshot shows the 'New Project' screen in the Google Cloud Platform. The top navigation bar is blue with the text 'Google Cloud Platform'. Below it, the main title is 'New Project'. The 'Project name *' field contains 'CAS-Integration-project'. The 'Project ID' field shows 'cas-integration-project' with a note that it cannot be changed later and an 'EDIT' link. The 'Organization' section shows a dropdown menu with 'test8-adallom.com' selected. The 'Location *' section shows a dropdown menu with 'US' selected. There is a 'BROWSE' button and a 'Parent organization or folder' link. At the bottom are 'CREATE' and 'CANCEL' buttons.

Enable required APIs

1. Switch to the dedicated project.
2. Go to the **Library** tab.
3. Search for and select **Cloud Logging API**, and then on the API page, select **ENABLE**.
4. Search for and select **Cloud Pub/Sub API**, and then on the API page, select **ENABLE**.

(!) Note

Make sure that you do not select Pub/Sub Lite API.

Create a dedicated service account for the security auditing integration

1. Under **IAM & admin**, select **Service accounts**.
2. Select **CREATE SERVICE ACCOUNT** to create a dedicated service account.
3. Enter an account name, and then select **Create**.
4. Specify the **Role** as **Pub/Sub Admin** and then select **Save**.

Create service account

Service account details — 2 Grant this service account access to project (optional) — 3 Grant users access to this service account (optional)

Service account permissions (optional)
Grant this service account access to MCAS for GCP - dev so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

Role: Pub/Sub Admin

Full access to topics, subscriptions, and snapshots.

+ ADD ANOTHER ROLE

CONTINUE **CANCEL**

5. Copy the **Email** value, you'll need this later.

Google Cloud Platform		CAS-Integration-project	CREATE SERVICE ACCOUNT	DELETE	
IAM & admin		Service accounts			
<input type="checkbox"/> IAM					
<input type="checkbox"/> Identity & Organization					
<input type="checkbox"/> Troubleshooter					
<input type="checkbox"/> Organization policies					
<input type="checkbox"/> Quotas					
<input type="checkbox"/> Service accounts					
<input type="checkbox"/> Labels					
<input type="checkbox"/> Settings					
<input type="checkbox"/> Privacy & Security					
<input type="checkbox"/> Cryptographic keys					
<input type="checkbox"/> Identity-Aware Proxy					
<input type="checkbox"/> Roles					
<input type="checkbox"/> Audit Logs					

Service accounts for project "CAS-Integration-project"
A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more](#)

Filter table

Email	Status	Name	Description	Key ID	Key creation date	Actions
cas-service-account@cas-integration-project.iam.gserviceaccount.com	Green checkmark	Cas-Service-Account		No keys		⋮

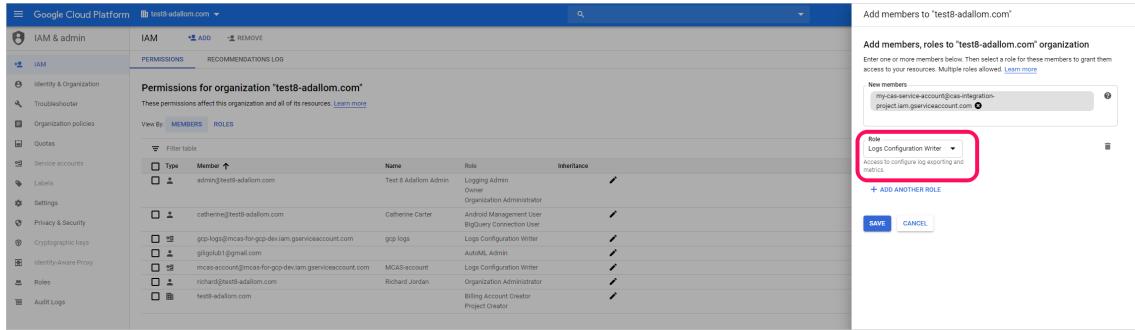
6. Under **IAM & admin**, select **IAM**.

- a. Switch to organization level.

b. Select **ADD**.

c. In the **New members** box, paste the **Email** value you copied earlier.

d. Specify the **Role** as **Logs Configuration Writer** and then select **Save**.



The screenshot shows the Google Cloud Platform IAM & admin interface. The left sidebar is expanded to show 'Service accounts'. The main area is titled 'Permissions for organization "test8-adaliam.com"'. It shows a table of members with their roles. A new member is being added, with the email 'my-cas-service-account@cas-integration-project.iam.gserviceaccount.com' in the 'New members' input field. The 'Role' dropdown is set to 'Logs Configuration Writer'. The 'SAVE' button is highlighted with a red box.

Create a private key for the dedicated service account

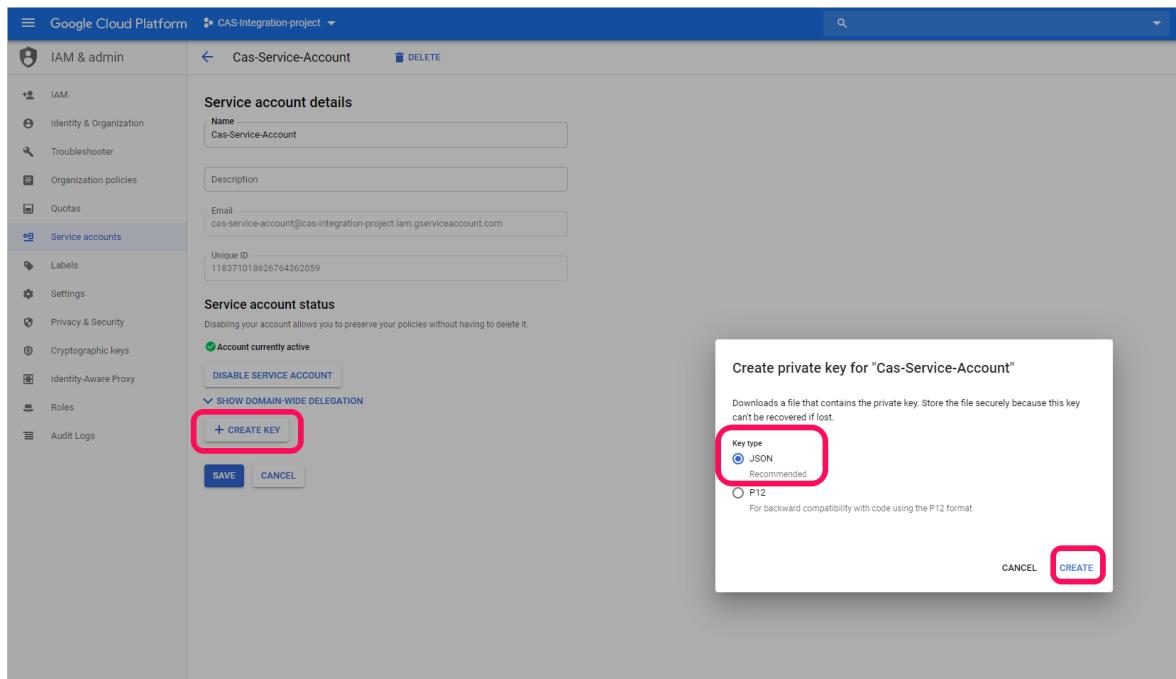
1. Switch to project level.

2. Under **IAM & admin**, select **Service accounts**.

3. Open the dedicated service account and select **Edit**.

4. Select **CREATE KEY**.

5. In the **Create private key** screen, select **JSON**, and then select **CREATE**.



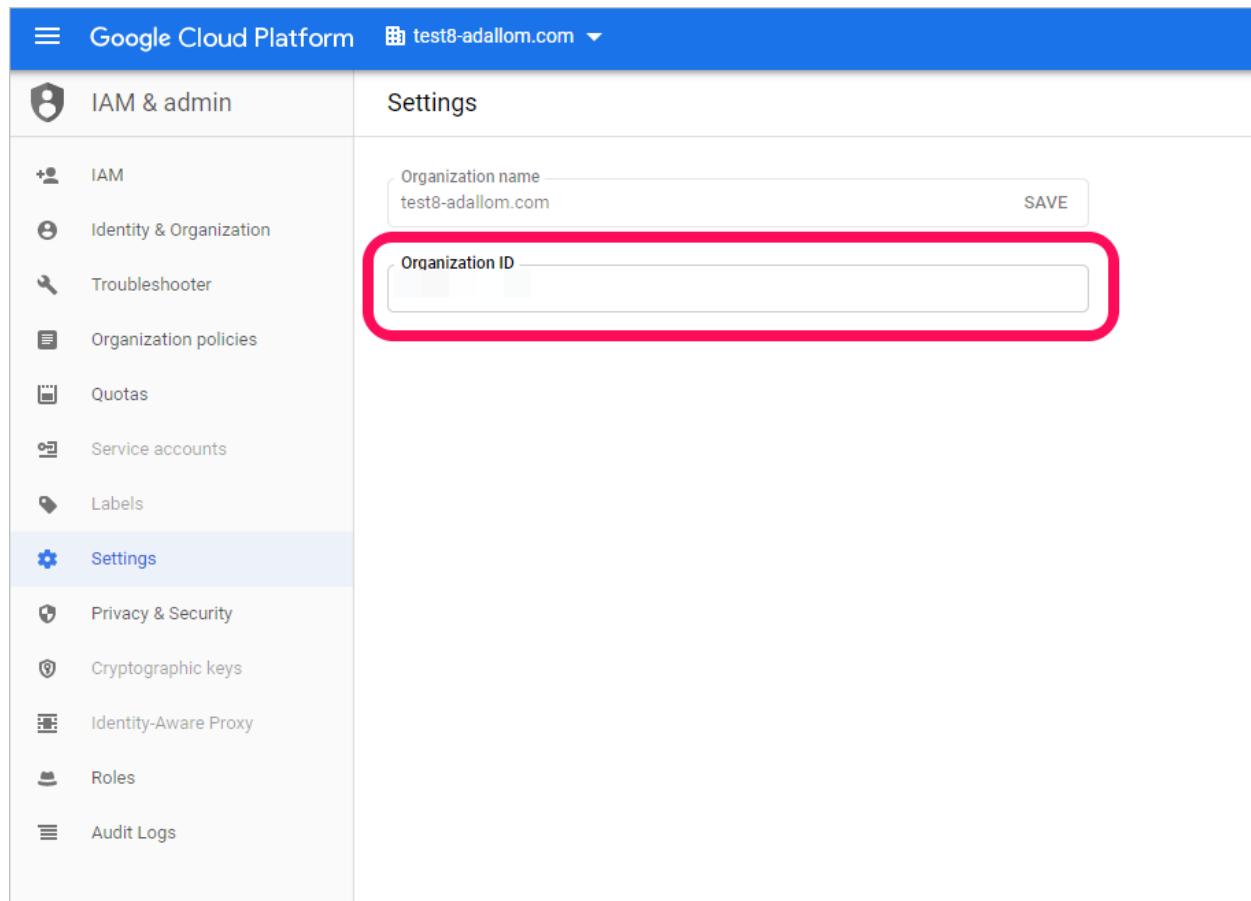
The screenshot shows the Google Cloud Platform IAM & admin interface. The left sidebar is expanded to show 'Service accounts'. A service account named 'Cas-Service-Account' is selected. The 'Service account details' section shows the name, email, and unique ID. The 'Service account status' section shows the account is currently active. A modal window titled 'Create private key for "Cas-Service-Account"' is open. The 'Key type' dropdown is set to 'JSON' (highlighted with a red box). The 'CREATE' button is highlighted with a red box.

! Note

You'll need the JSON file that is downloaded to your device later.

Retrieve your Organization ID

Make a note of your **Organization ID**, you'll need this later. For more information, see [Getting your organization ID](#).



The screenshot shows the Google Cloud Platform IAM & admin Settings page. On the left, a sidebar lists various options: IAM, Identity & Organization, Troubleshooter, Organization policies, Quotas, Service accounts, Labels, Settings (which is selected and highlighted in blue), Privacy & Security, Cryptographic keys, Identity-Aware Proxy, Roles, and Audit Logs. The main content area is titled 'Settings' and shows the 'Organization name' as 'test8-adallom.com' with a 'SAVE' button. Below it is the 'Organization ID' field, which is highlighted with a red box. The 'Organization ID' field contains the value 'test8-adallom'.

Connect Google Cloud Platform auditing to Defender for Cloud Apps

This procedure describes how to add the GCP connection details to connect Google Cloud Platform auditing to Defender for Cloud Apps.

1. In the Microsoft Defender Portal, select **Settings**. Then choose **Cloud Apps**. Under **Connected apps**, select **App Connectors**.
2. In the **App connectors** page, to provide the GCP connector credentials, do one of the following:

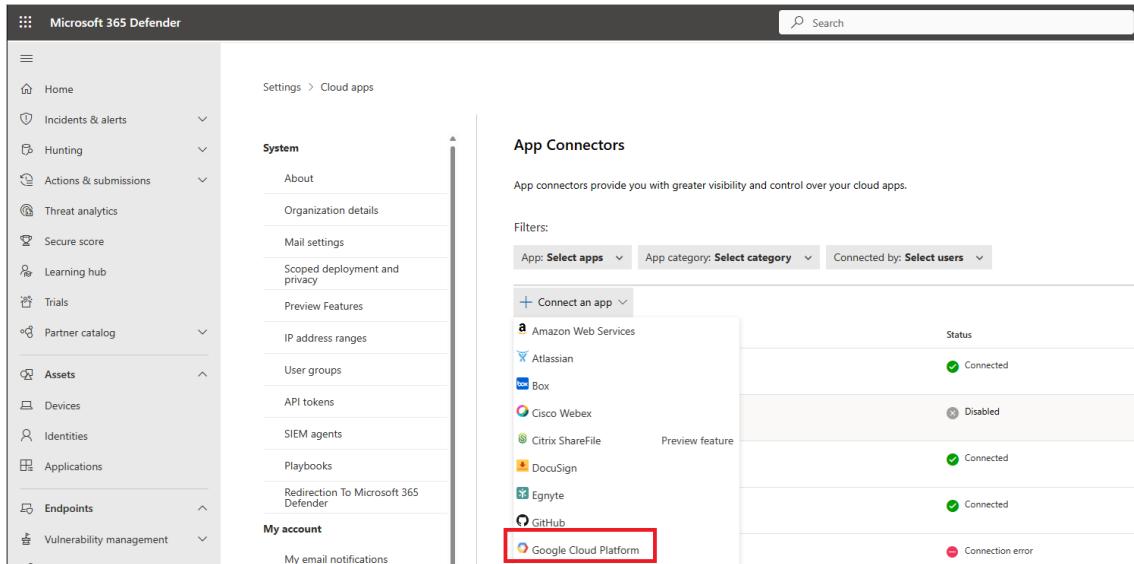
ⓘ Note

We recommend that you connect your Google Workspace instance to get unified user management and governance. This is the recommended even if

you do not use any Google Workspace products and the GCP users are managed via the Google Workspace user management system.

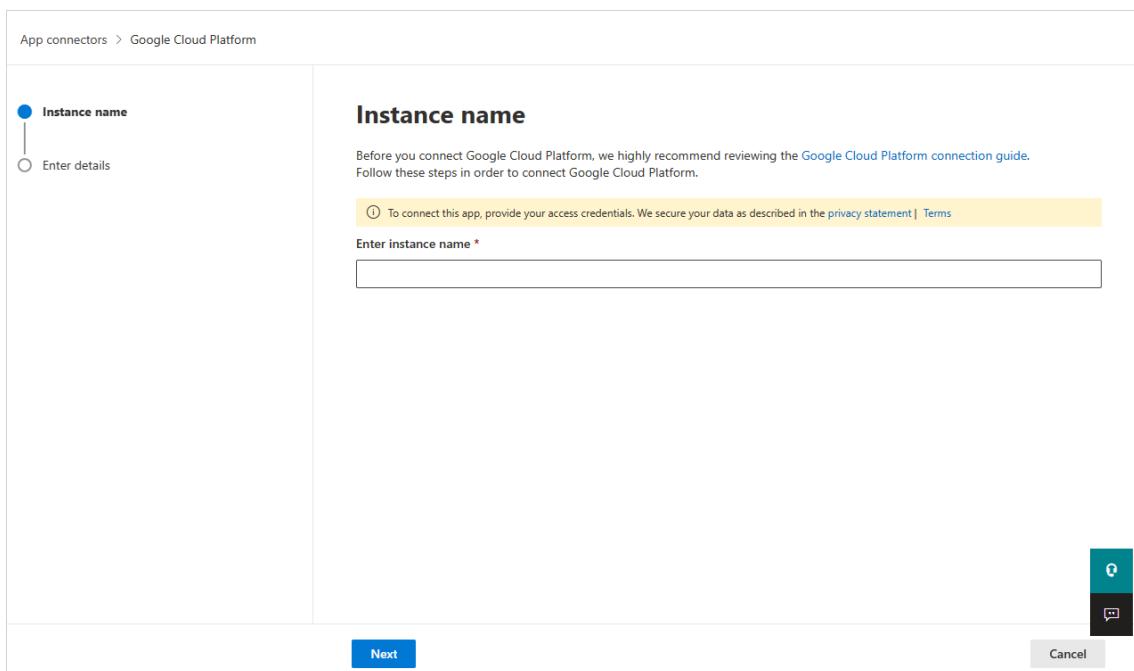
For a new connector

- Select **+Connect an app**, followed by **Google Cloud Platform**.



The screenshot shows the Microsoft 365 Defender Settings interface. On the left, there's a navigation sidebar with various options like Home, Incidents & alerts, Hunting, Actions & submissions, Threat analytics, Secure score, Learning hub, Trials, Partner catalog, Assets, Devices, Identities, Applications, Endpoints, and Vulnerability management. The 'Cloud apps' section is selected. On the right, under 'App Connectors', it says 'App connectors provide you with greater visibility and control over your cloud apps.' There are filters for 'App: Select apps', 'App category: Select category', and 'Connected by: Select users'. A list of connectors is shown, including Amazon Web Services (Connected), Atlassian (Disabled), Box (Connected), Cisco Webex (Preview feature), Citrix ShareFile (Connected), DocuSign (Connected), Egnyte (Connected), GitHub (Connected), and Google Cloud Platform (highlighted with a red box). The 'Google Cloud Platform' entry has a status of 'Connection error'.

- In the next window, provide a name for the connector, and then select **Next**.



The screenshot shows the 'App connectors > Google Cloud Platform' configuration page. On the left, there are two radio button options: 'Instance name' (selected) and 'Enter details'. On the right, the 'Instance name' section is displayed with the sub-section 'Before you connect this app, we highly recommend reviewing the [Google Cloud Platform connection guide](#). Follow these steps in order to connect Google Cloud Platform.' Below this is a note: '(i) To connect this app, provide your access credentials. We secure your data as described in the [privacy statement](#) | [Terms](#)'. A text input field is labeled 'Enter instance name *' with a placeholder '(i) To connect this app, provide your access credentials. We secure your data as described in the [privacy statement](#) | [Terms](#)'. At the bottom right are 'Next' and 'Cancel' buttons, and a feedback icon.

- In the **Enter details** page, do the following, and then select **Submit**.
 - In the **Organization ID** box, enter the organization you made a note of earlier.
 - In the **Private key file** box, browse to the JSON file you downloaded earlier.

App connectors > Google Cloud Platform

Instance name

Enter details

Enter details

Organization ID *

Private key file No file chosen

?

Back Submit Cancel

For an existing connector

- In the list of connectors, on the row in which the GCP connector appears, select **Edit settings**.

App Connectors

App connectors provide you with greater visibility and control over your cloud apps.

Filters: App: Google Cloud, Google Cloud Platform, Google... App category: Select category Connected by: Select users Advanced filters

1 - 2 of 2 connected apps Show details Hide filters Table settings

App	Status	Was connected on	Last activity	Accounts
Google Workspace-US Collaboration	Connected	Dec 28, 2017 1:59 PM	Jan 23, 2023 4:34 PM	11
GCP - US	Connected	Oct 27, 2019 4:53 PM	Jan 19, 2023 12:35 PM	2

Edit settings

Disable App connector

Edit instance name

Connect Google Cloud Platform instance...

- In the **Edit settings** page, do the following, and then select **Submit**.
 - In the **Organization ID** box, enter the organization you made a note of earlier.
 - In the **Private key file** box, browse to the JSON file you downloaded earlier.

The screenshot shows a 'Enter details' page for connecting an app to Google Cloud Platform. On the left, there's a sidebar with a 'Instance name' section and a 'Enter details' section, both marked with a blue checkmark. The main area is titled 'Enter details' and contains fields for 'Organization ID' (with a required asterisk) and 'Private key file' (with a 'Choose File' button and a note 'No file chosen'). At the bottom are 'Back', 'Submit', and 'Cancel' buttons, along with a help icon and a copy icon.

3. In the Microsoft Defender Portal, select **Settings**. Then choose **Cloud Apps**. Under **Connected apps**, select **App Connectors**. Make sure the status of the connected App Connector is **Connected**.

Note

Defender for Cloud Apps will create an aggregated export sink (organization level), a Pub/Sub topic and Pub/Sub subscription using the integration service account in the integration project.

Aggregated export sink is used to aggregate logs across the GCP organization and the Pub/Sub topic created is used as the destination. Defender for Cloud Apps subscribes to this topic through the Pub/Sub subscription created to retrieve the admin activity logs across the GCP organization.

If you have any problems connecting the app, see [Troubleshooting App Connectors](#).

Next steps

[Control cloud apps with policies](#)

If you run into any problems, we're here to help. To get assistance or support for your product issue, please [open a support ticket](#).

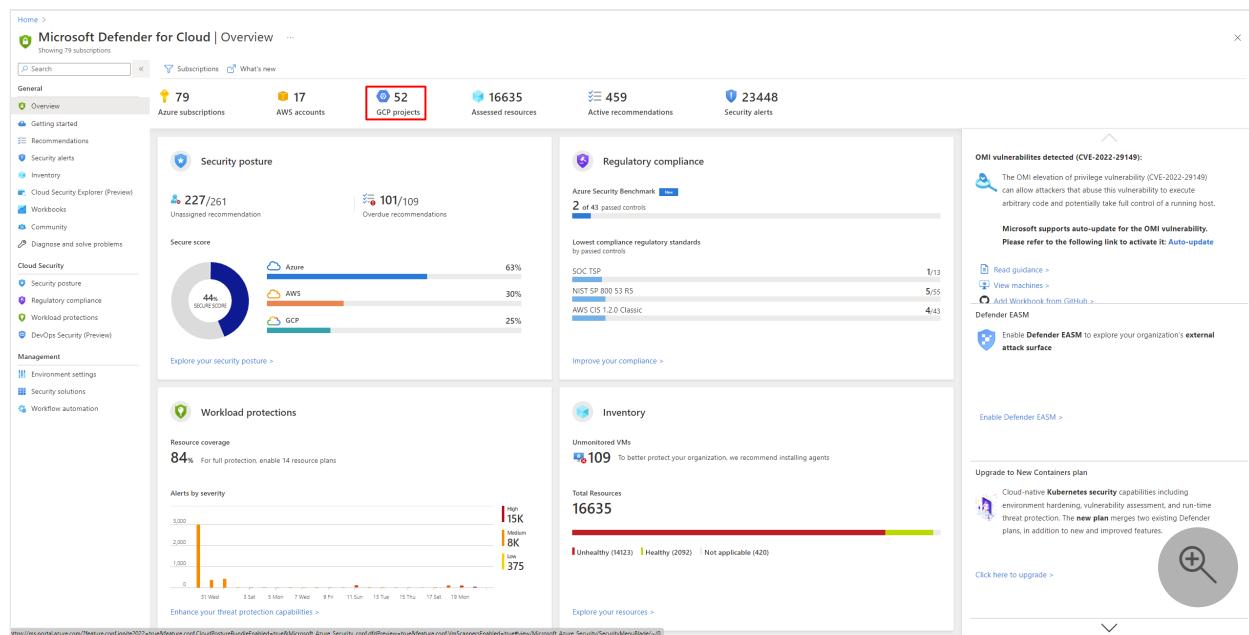
Connect your GCP project to Microsoft Defender for Cloud

Article • 01/15/2024

Workloads commonly span multiple cloud platforms. Cloud security services must do the same. Microsoft Defender for Cloud helps protect workloads in Google Cloud Platform (GCP), but you need to set up the connection between them and Defender for Cloud.

If you're connecting a GCP project that you previously connected by using the classic connector, you must [remove it](#) first. Using a GCP project connected by both the classic and native connectors can produce duplicate recommendations.

This screenshot shows GCP accounts displayed in the Defender for Cloud [overview dashboard](#).



The screenshot shows the Microsoft Defender for Cloud Overview dashboard. At the top, there are several summary metrics: 79 Azure subscriptions, 17 AWS accounts, 52 GCP projects (highlighted with a red box), 16635 Assessed resources, 459 Active recommendations, and 23448 Security alerts. The main dashboard is divided into several sections: Security posture, Regulatory compliance, Workload protections, and Inventory. The Workload protections section shows a chart of Resource coverage at 84% and a bar chart of Alerts by severity. The Inventory section shows a chart of Total Resources at 16635. On the right side, there is a sidebar with a section for OMI vulnerabilities detected (CVE-2022-29149), which details the OMI elevation of privilege vulnerability and provides a link to auto-update. There are also buttons for Read guidance, View machines, Add Workbook from GitHub, and Enable Defender EASM.

Prerequisites

To complete the procedures in this article, you need:

- A Microsoft Azure subscription. If you don't have an Azure subscription, you can [sign up for a free one](#).
- Microsoft Defender for Cloud set up on your Azure subscription.
- Access to a GCP project.

- **Contributor** permission on the relevant Azure subscription, and **Owner** permission on the GCP organization or project.

You can learn more about Defender for Cloud pricing on [the pricing page](#).

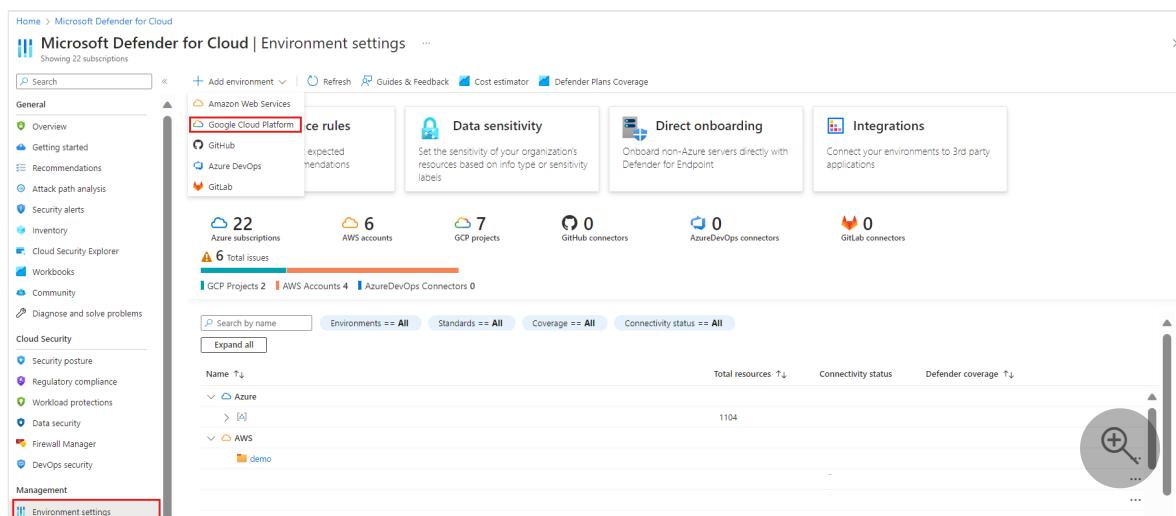
When you're connecting GCP projects to specific Azure subscriptions, consider the [Google Cloud resource hierarchy](#) and these guidelines:

- You can connect your GCP projects to Microsoft Defender for Cloud at the *project* level.
- You can connect multiple projects to one Azure subscription.
- You can connect multiple projects to multiple Azure subscriptions.

Connect your GCP project

To connect your GCP project to Defender for Cloud by using a native connector:

1. Sign in to the [Azure portal](#).
2. Go to **Defender for Cloud > Environment settings**.
3. Select **Add environment > Google Cloud Platform**.



4. Enter all relevant information.

Add GCP project

Google cloud

[Project details](#) [Select plans](#) [Configure access](#) [Review and generate](#)

The first step to onboarding your GCP project is to enter a descriptive name for the cloud connector and choose whether to connect one project or the whole organization.

Connector name *	<input type="text"/>
Onboard *	<input type="radio"/> Organization <input checked="" type="radio"/> Single project
Subscription *	<input type="text" value="ASC Multi-Cloud Demo"/>
Resource group *	<input type="text" value="Demo"/> Create new
Location *	<input type="text" value="East US"/>
GCP project number *	<input type="text"/>
GCP project id *	<input type="text"/>



[< Previous](#) [Next : Select plans >](#)

Optionally, if you select **Organization**, a management project and an organization custom role are created on your GCP project for the onboarding process.

Autoprovisioning is enabled for the onboarding of new projects.

Select Defender plans

In this section of the wizard, you select the Defender for Cloud plans that you want to enable.

1. Select **Next: Select plans**.
2. For the plans that you want to connect, turn the toggle to **On**. By default, all necessary prerequisites and components are provisioned. [Learn how to configure each plan](#).

Home > Microsoft Defender for Cloud | Environment settings >

Add GCP project ...

Google cloud

Organization details **Select plans** Configure access Review and generate

Select plans

Cloud Security Posture Management (CSPM)

Microsoft Defender CSPM provides advanced security posture capabilities including agentless vulnerability scanning, data-aware security posture, the cloud security graph, and advanced threat hunting. Pricing is based on project size, with billing applying only for Servers, Databases, and Storage resources at \$1/billable resource/month. Foundational CSPM includes asset discovery, continuous assessment and security recommendations for posture hardening and a Secure score which measure the current status of your organization's posture

Plan	Pricing*	Monitoring coverage	Status
Foundational CSPM	Free Details >	Permissions: Read (Security/Audit)	Off On
Defender CSPM	\$/Billable resource/Month, free until February 1st 2024 Details >	Full Settings >	Off On

Cloud Workload Protection (CWP)

Microsoft Defender for Cloud provides comprehensive, cloud-native protections from development to runtime in multi-cloud environments.

Plan	Pricing*	Monitoring coverage	Status
Servers	Plan 2 (\$/Server/Month) Select tier > May incur GCP costs. (i)	Partial Settings >	Off On
Databases	\$/Server/Month Details >	Partial Settings >	Off On
Containers	Free (during preview) Details > Will incur GCP costs. (i)	Full Settings >	Off On

* The price displayed represents the list price prior to any discounts or special offers being applied.

< Previous Next : Configure access >

If you choose to turn on the Microsoft Defender for Containers plan, ensure that you meet the [network requirements](#) for it.

3. Select **Configure access** and make the following selections:

a. Select the deployment type:

- **Default access:** Allows Defender for Cloud to scan your resources and automatically include future capabilities.
- **Least privilege access:** Grants Defender for Cloud access to only the current permissions needed for the selected plans. If you select the least privileged permissions, you receive notifications on any new roles and permissions that are required to get full functionality for connector health.

b. Select the deployment method: **GCP Cloud Shell** or **Terraform**.

Add GCP project ...

Google cloud

X

 Project details Select plans **Configure access** Review and generatePermissions type Default access Least privilege accessDeployment method GCP Cloud shell Terraform

To configure access on GCP, a template has been generated based on the plans selected in the previous tab.
Upon executing the template, custom role(s) will be created to facilitate the onboarding of the GCP project.

```
# Setting The Environment Variables
CspmCustomRoleId="MDCCspmCustomRole"
# Create a custom role for MDC CSPM

gcloud iam roles create MDCCspmCustomRole \
--project=123456789 \
--title="MDCCspmCustomRole" \
--description="Microsoft Defender for cloud CSPM custom role"

gcloud iam roles update MDCCspmCustomRole \
```

Deployment template is auto filled with default parameters. [To edit service account email and identity providers, click here >](#)

Run the GCP Cloud Shell template

1. Copy the template.
2. Log in to [GCP Cloud Shell](#).
3. Paste the script into the Cloud Shell terminal and run it.

[< Previous](#) [Next : Review and generate >](#)

4. Follow the on-screen instructions for the selected deployment method to complete the required dependencies on GCP.

5. Select **Next: Review and generate**.

6. Select **Create**.

! **Note**

The following APIs must be enabled in order to discover your GCP resources and allow the authentication process to occur:

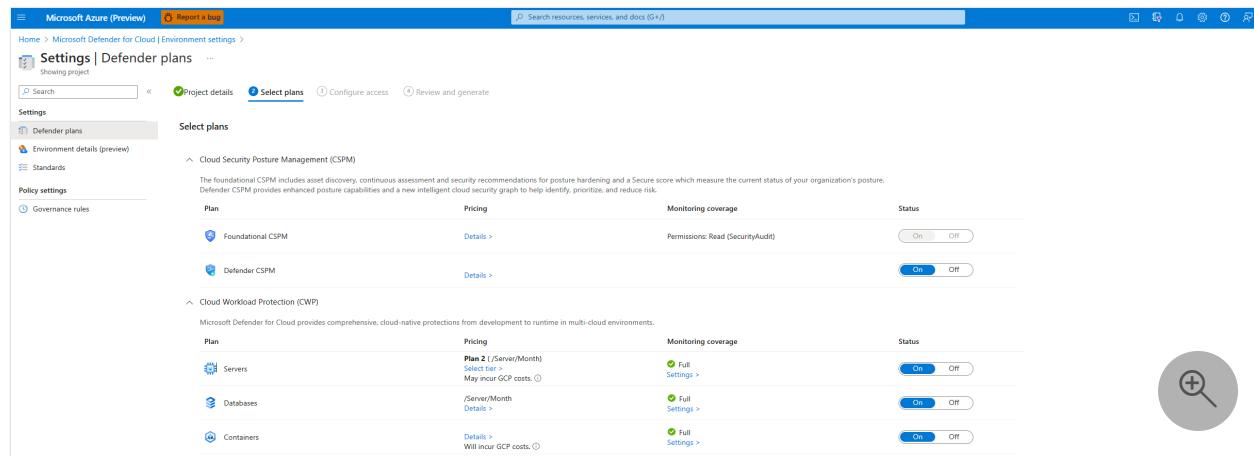
- `iam.googleapis.com`
- `sts.googleapis.com`
- `cloudresourcemanager.googleapis.com`
- `iamcredentials.googleapis.com`
- `compute.googleapis.com` If you don't enable these APIs at this time, you can enable them during the onboarding process by running the GCloud script.

After you create the connector, a scan starts on your GCP environment. New recommendations appear in Defender for Cloud after up to 6 hours. If you enabled

autoprovisioning, Azure Arc and any enabled extensions are installed automatically for each newly detected resource.

Optional: Configure selected plans

By default, all plans are **On**. You can turn off plans that you don't need.



Configure the Defender for Servers plan

Microsoft Defender for Servers brings threat detection and advanced defenses to your GCP virtual machine (VM) instances. To have full visibility into Microsoft Defender for Servers security content, connect your GCP VM instances to Azure Arc. If you choose the Microsoft Defender for Servers plan, you need:

- Microsoft Defender for Servers enabled on your subscription. Learn how to enable plans in [Enable enhanced security features](#).
- Azure Arc for servers installed on your VM instances.

We recommend that you use the autoprovisioning process to install Azure Arc on your VM instances. Autoprovisioning is enabled by default in the onboarding process and requires **Owner** permissions on the subscription. The Azure Arc autoprovisioning process uses the OS Config agent on the GCP end. [Learn more about the availability of the OS Config agent on GCP machines](#).

The Azure Arc autoprovisioning process uses the VM manager on GCP to enforce policies on your VMs through the OS Config agent. A VM that has an [active OS Config agent](#) incurs a cost according to GCP. To see how this cost might affect your account, refer to the [GCP technical documentation](#).

Microsoft Defender for Servers doesn't install the OS Config agent to a VM that doesn't have it installed. However, Microsoft Defender for Servers enables communication

between the OS Config agent and the OS Config service if the agent is already installed but not communicating with the service. This communication can change the OS Config agent from `inactive` to `active` and lead to more costs.

Alternatively, you can manually connect your VM instances to Azure Arc for servers. Instances in projects with the Defender for Servers plan enabled that aren't connected to Azure Arc are surfaced by the recommendation **GCP VM instances should be connected to Azure Arc**. Select the **Fix** option in the recommendation to install Azure Arc on the selected machines.

The respective Azure Arc servers for EC2 instances or GCP virtual machines that no longer exist (and the respective Azure Arc servers with a status of **Disconnected or Expired**) are removed after seven days. This process removes irrelevant Azure Arc entities to ensure that only Azure Arc servers related to existing instances are displayed.

Ensure that you fulfill the [network requirements for Azure Arc](#).

Enable these other extensions on the Azure Arc-connected machines:

- Microsoft Defender for Endpoint
- A vulnerability assessment solution (Microsoft Defender Vulnerability Management or Qualys)
- The Log Analytics agent on Azure Arc-connected machines or the Azure Monitor agent

Make sure the selected Log Analytics workspace has a security solution installed. The Log Analytics agent and the Azure Monitor agent are currently configured at the *subscription* level. All the multicloud accounts and projects (from both AWS and GCP) under the same subscription inherit the subscription settings for the Log Analytics agent and the Azure Monitor agent. [Learn more about monitoring components for Defender for Servers](#).

Defender for Servers assigns tags to your GCP resources to manage the autoprovioning process. You must have these tags properly assigned to your resources so that Defender for Servers can manage your resources: `Cloud`, `InstanceName`, `MDFCSecurityConnector`, `MachineId`, `ProjectId`, and `ProjectNumber`.

To configure the Defender for Servers plan:

1. Follow the [steps to connect your GCP project](#).
2. On the **Select plans** tab, select **Configure**.

 Auto-provisioning enabled
[Configure >](#)

3. On the **Auto-provisioning configuration** pane, turn the toggles to **On** or **Off**, depending on your need.

Auto-provisioning configuration

X

To prevent, detect, and respond to threats, Microsoft Defender for Cloud collects security data and events from your machines. [Learn more](#)

Azure Arc agent

 On

Connects your servers to Azure. Enable to install Azure Arc on new and existing machines with OS config agent.

 Note: When Arc auto-provisioning is enabled, it will connect existing OS config agents on GCP's side that are not communicating with the OS config service. This may lead to additional charges. For more information, see [GCP documentation](#)

Additional extensions for Arc connected machines (preview)

The selected extensions will be automatically provisioned on machines connected to Azure Arc.

Microsoft Defender for Endpoint extension

 On

Provides comprehensive endpoint detection and response (EDR) capabilities. [Learn more](#)

Vulnerability assessment

 On

Enable vulnerability discovery and management tools for your machines. [Learn more](#)

Vulnerability assessment solution

Microsoft threat and vulnerability management

Microsoft Defender for Cloud integrated Qualys scanner

 If you've already configured auto provisioning for a BYOL solution, you'll need to disable it before you can configure this agent. [Learn more](#)

[Save](#)

[Cancel](#)

If **Azure Arc agent** is **Off**, you need to follow the manual installation process mentioned earlier.

4. Select **Save**.

5. Continue from step 8 of the [Connect your GCP project](#) instructions.

Configure the Defender for Databases plan

To have full visibility into Microsoft Defender for Databases security content, connect your GCP VM instances to Azure Arc.

To configure the Defender for Databases plan:

1. Follow the [steps to connect your GCP project](#).

2. On the **Select plans** tab, select **Configure**.



3. On the **Auto-provisioning configuration** pane, turn the toggles to **On** or **Off**, depending on your need.

Auto-provisioning configuration

X

To prevent, detect, and respond to threats, Azure Defender for Cloud collects security data and events from your machines. [Learn more](#).

Azure Arc agent

On

Connects your servers to Azure. Enable to install Azure Arc on new and existing machines with OS config agent.

i Note: When Arc auto-provisioning is enabled, it will connect existing OS config agents on GCP's side that are not communicating with the OS config service. This may lead to additional charges. For more information, see [GCP documentation](#)

Additional agents for Arc connected machines

Settings for the Log Analytics and discovery and registration service for SQL on Arc agents are managed at the subscription level. For advanced configuration, edit the subscription settings.

SQL servers on machines

On

Applies to SQL on Azure virtual machines, SQL servers on-premise, and Azure Arc enabled SQL servers.

[Learn more](#).

Log Analytics extension

On

Collects security-related configurations and event logs from the machine and stores the data in your Log Analytics workspace for analysis. [Learn more](#).

Note: Settings for the Log Analytics agent are managed at the subscription level. For advanced configuration, edit the subscription settings.

i Any other solutions enabled on the selected workspace will be applied to machines that are connected to it. For paid solutions, this could result in additional charges. X

Automatic SQL server discovery and registration

Off

This feature will discover and register both existing and future SQL servers 2012+ which are running on your virtual machines, enabling better manageability of security posture and more comprehensive data assets inventory.

The machines will be registered as SQL ARC enabled servers. Neither the SQL nor the host will be required a restart.

[Learn more](#).

Save

Cancel

If the toggle for Azure Arc is **Off**, you need to follow the manual installation process mentioned earlier.

4. Select **Save**.

5. Continue from step 8 of the [Connect your GCP project](#) instructions.

Configure the Defender for Containers plan

Microsoft Defender for Containers brings threat detection and advanced defenses to your GCP Google Kubernetes Engine (GKE) Standard clusters. To get the full security value out of Defender for Containers and to fully protect GCP clusters, ensure that you meet the following requirements.

ⓘ Note

- If you choose to disable the available configuration options, no agents or components will be deployed to your clusters. [Learn more about feature availability](#).
- Defender for Containers when deployed on GCP, might incur external costs such as [logging costs](#), [pub/sub costs](#) and [egress costs](#).
- **Kubernetes audit logs to Defender for Cloud:** Enabled by default. This configuration is available at the GCP project level only. It provides agentless collection of the audit log data through [GCP Cloud Logging](#) to the Microsoft Defender for Cloud back end for further analysis. Defender for Containers requires control plane audit logs to provide [runtime threat protection](#). To send Kubernetes audit logs to Microsoft Defender, toggle the setting to **On**.

ⓘ Note

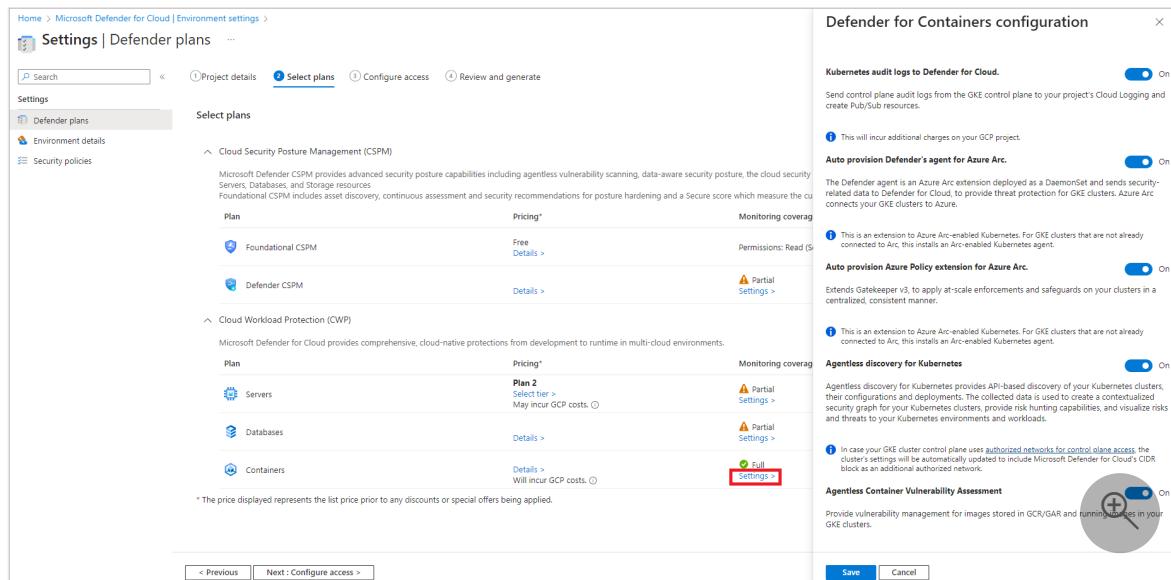
If you disable this configuration, then the [Threat detection \(control plane\)](#) feature will be disabled. [Learn more about features availability](#).

- **Auto provision Defender's agent for Azure Arc and Auto provision Azure Policy extension for Azure Arc:** Enabled by default. You can install Azure Arc-enabled Kubernetes and its extensions on your GKE clusters in three ways:
 - Enable Defender for Containers autoprovisioning at the project level, as explained in the instructions in this section. We recommend this method.
 - Use Defender for Cloud recommendations for per-cluster installation. They appear on the Microsoft Defender for Cloud recommendations page. [Learn how to deploy the solution to specific clusters](#).
 - Manually install [Arc-enabled Kubernetes](#) and [extensions](#).

- [Agentless discovery for Kubernetes](#) provides API-based discovery of your Kubernetes clusters. To enable the **Agentless discovery for Kubernetes** feature, toggle the setting to **On**.
- The [Agentless Container Vulnerability Assessment](#) provides vulnerability management for images stored in Google Container Registry (GCR) and Google Artifact Registry (GAR) and running images on your GKE clusters. To enable the **Agentless Container Vulnerability Assessment** feature, toggle the setting to **On**.

To configure the Defender for Containers plan:

1. Follow the steps to [connect your GCP project](#).
2. On the **Select plans** tab, select **Configure**. Then, on the **Defender for Containers configuration** pane, turn the toggles to **On**.



3. Select **Save**.
4. Continue from step 8 of the [Connect your GCP project](#) instructions.

Configure the Defender CSPM plan

If you choose the Microsoft Defender CSPM plan, you need:

- A Microsoft Azure subscription. If you don't have an Azure subscription, you can [sign up for a free subscription](#).
- You must [enable Microsoft Defender for Cloud](#) on your Azure subscription.
- In order to gain access to all of the features available from the CSPM plan, the plan must be enabled by the **Subscription Owner**.

Learn more about how to [enable Defender CSPM](#).

To configure the Defender CSPM plan:

1. Follow the [steps to connect your GCP project](#).
2. On the **Select plans** tab, select **Configure**.



3. On the **Plan configuration** pane, turn the toggles to **On** or **Off**. To get the full value of Defender CSPM, we recommend that you turn all toggles to **On**.

Plan Configuration

To prevent, detect, and respond to threats, Microsoft Defender for Cloud collects security data and events from your machines. [Learn more](#)

Agentless scanning On

Scan your GCP VM instances for installed software and vulnerabilities without requiring agents, network connectivity or impacting machine performance. Results are powered by Microsoft Defender Vulnerability Management engine. [Learn more](#)

Sensitive data discovery On

Sensitive data discovery automatically discovers managed cloud data resources containing sensitive data at scale. This feature accesses your data, it is agentless, uses smart sampling scanning, and integrates with Microsoft Purview sensitive information types and labels.

Permissions Management (Preview) On

Insights into Cloud Infrastructure Entitlement Management (CIEM). CIEM is a way of ensuring that the identities and access rights of entities, such as users, groups, roles, or applications, are appropriate and secured in cloud environments. Permissions Management helps to understand the access permissions to cloud resources, such as virtual machines, storage, or databases, and risks associated with those permissions. The setup, data collection and the recommendations generation could take up to 24 hours. [Learn more](#)

Save **Cancel**

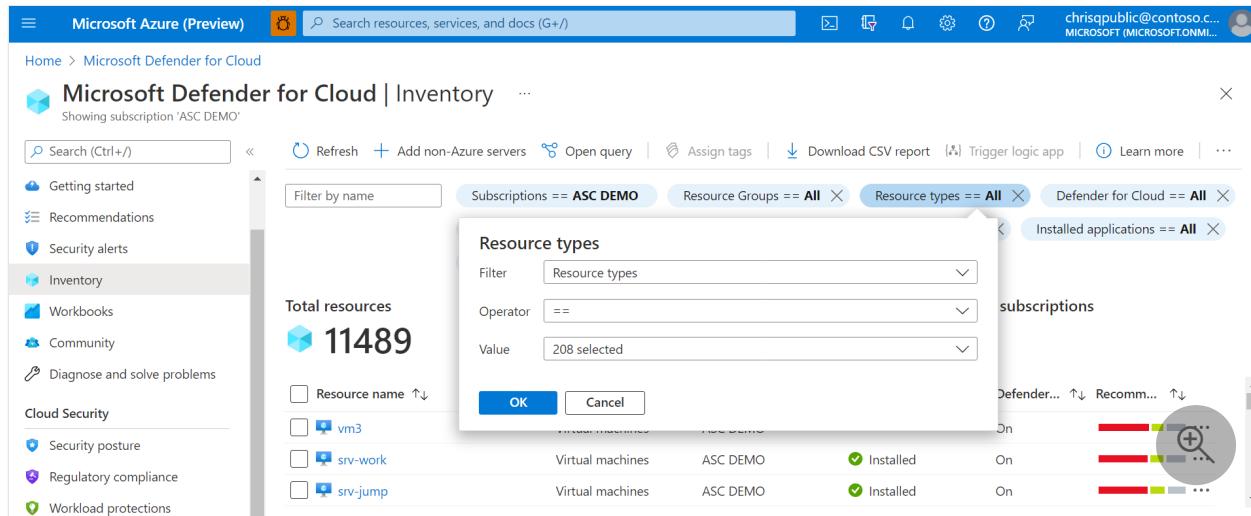
4. Select Save.

5. Continue from step 8 of the [Connect your GCP project](#) instructions.

Monitor your GCP resources

The security recommendations page in Defender for Cloud displays your GCP resources together with your Azure and AWS resources for a true multicloud view.

To view all the active recommendations for your resources by resource type, use the asset inventory page in Defender for Cloud and filter to the GCP resource type that you're interested in.



The screenshot shows the Microsoft Defender for Cloud Inventory page. A modal window is open, titled 'Resource types', with the following settings: Filter: Resource types, Operator: ==, Value: 208 selected. The background shows a list of 'Total resources' (11489) and a sidebar with various navigation links.

Integrate with Microsoft Defender XDR

When you enable Defender for Cloud, Defender for Cloud alerts are automatically integrated into the Microsoft Defender Portal. No further steps are needed.

The integration between Microsoft Defender for Cloud and Microsoft Defender XDR brings your cloud environments into Microsoft Defender XDR. With Defender for Cloud's alerts and cloud correlations integrated into Microsoft Defender XDR, SOC teams can now access all security information from a single interface.

Learn more about Defender for Cloud's [alerts in Microsoft Defender XDR](#).

Next steps

Connecting your GCP project is part of the multicloud experience available in Microsoft Defender for Cloud:

- Protect all of your resources with Defender for Cloud.
- Set up your [on-premises machines](#) and [AWS account](#).
- [Troubleshoot your multicloud connectors](#).
- Get answers to [common questions](#) about connecting your GCP project.

How Defender for Cloud Apps helps protect your Google Cloud Platform (GCP) environment

Article • 01/22/2024

Google Cloud Platform is an IaaS provider that enables your organization to host and manage their entire workloads in the cloud. Along with the benefits of leveraging infrastructure in the cloud, your organization's most critical assets may be exposed to threats. Exposed assets include storage instances with potentially sensitive information, compute resources that operate some of your most critical applications, ports, and virtual private networks that enable access to your organization.

Connecting GCP to Defender for Cloud Apps helps you secure your assets and detect potential threats by monitoring administrative and sign-in activities, notifying on possible brute force attacks, malicious use of a privileged user account, and unusual deletions of VMs.

Main threats

- Abuse of cloud resources
- Compromised accounts and insider threats
- Data leakage
- Resource misconfiguration and insufficient access control

How Defender for Cloud Apps helps to protect your environment

- [Detect cloud threats, compromised accounts, and malicious insiders](#)
- [Use the audit trail of activities for forensic investigations](#)

Control GCP with built-in policies and policy templates

You can use the following built-in policy templates to detect and notify you about potential threats:

[+] [Expand table](#)

Type	Name
Built-in anomaly detection policy	Activity from anonymous IP addresses Activity from infrequent country Activity from suspicious IP addresses Impossible travel Activity performed by terminated user (requires Microsoft Entra ID as IdP) Multiple failed login attempts Unusual administrative activities Multiple delete VM activities Unusual multiple VM creation activities (preview)
Activity policy template	Changes to compute engine resources Changes to StackDriver configuration Changes to storage resources Changes to Virtual Private Network Logon from a risky IP address

For more information about creating policies, see [Create a policy](#).

Automate governance controls

In addition to monitoring for potential threats, you can apply and automate the following GCP governance actions to remediate detected threats:

[+] [Expand table](#)

Type	Action
User governance	<ul style="list-style-type: none">- Require user to reset password to Google (requires connected linked Google Workspace instance)- Suspend user (requires connected linked Google Workspace instance)- Notify user on alert (via Microsoft Entra ID)- Require user to sign in again (via Microsoft Entra ID)- Suspend user (via Microsoft Entra ID)

For more information about remediating threats from apps, see [Governing connected apps](#).

Protect GCP in real time

Review our best practices for [securing and collaborating with external users](#) and [blocking and protecting the download of sensitive data to unmanaged or risky devices](#).

Connect Google Cloud Platform to Microsoft Defender for Cloud Apps

This section provides instructions for connecting Microsoft Defender for Cloud Apps to your existing Google Cloud Platform (GCP) account using the connector APIs. This connection gives you visibility into and control over GCP use. For information about how Defender for Cloud Apps protects GCP, see [Protect GCP](#).

We recommend that you use a dedicated project for the integration and restrict access to the project to maintain stable integration and prevent deletions/modifications of the setup process.

Note

The instructions for connecting your GCP environment for auditing follow [Google's recommendations](#) for consuming aggregated logs. The integration leverages Google StackDriver and will consume additional resources that might impact your billing. The consumed resources are:

- Aggregated export sink – Organization level 
- Pub/Sub topic – GCP project level 
- Pub/Sub subscription – GCP project level 

The Defender for Cloud Apps auditing connection only imports Admin Activity audit logs; Data Access and System Event audit logs are not imported. For more information about GCP logs, see [Cloud Audit Logs](#).

Prerequisites

The integrating GCP user must have the following permissions:

- **IAM and Admin edit** – Organization level
- **Project creation and edit**

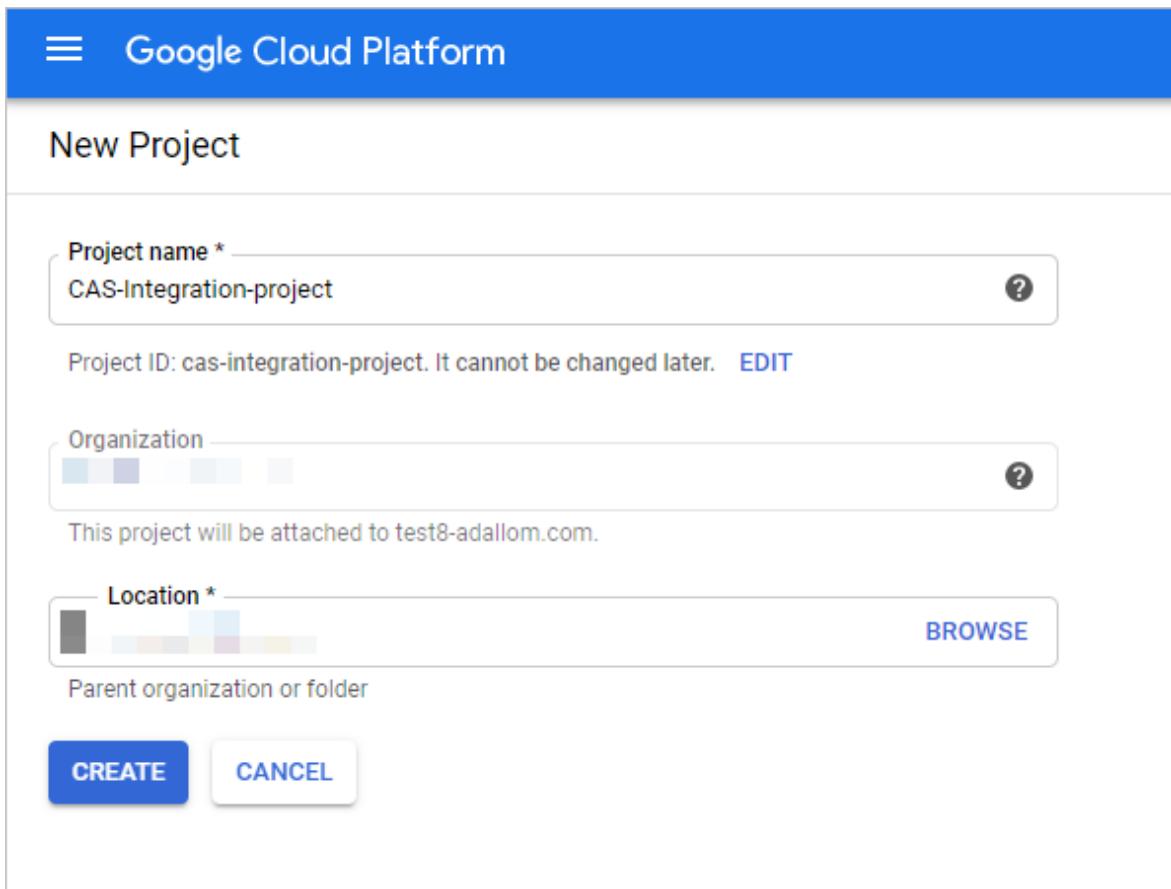
You can connect GCP **Security auditing** to your Defender for Cloud Apps connections to gain visibility into and control over GCP app use.

Configure Google Cloud Platform

Create a dedicated project

Create a dedicated project in GCP under your organization to enable integration isolation and stability

1. Sign in to your GCP portal using your integrating GCP user account.
2. Select **Create Project** to start a new project.
3. In the **New project** screen, name your project and select **Create**.



The screenshot shows the 'New Project' screen in the Google Cloud Platform. The top navigation bar is blue with the text 'Google Cloud Platform'. Below it, the main title is 'New Project'. The 'Project name *' field contains 'CAS-Integration-project'. The 'Project ID' field shows 'cas-integration-project' with a note that it cannot be changed later and an 'EDIT' link. The 'Organization' section shows a dropdown menu with 'test8-adallom.com' selected. The 'Location *' section shows a dropdown menu with 'US' selected. There is a 'BROWSE' button and a 'Parent organization or folder' link. At the bottom are 'CREATE' and 'CANCEL' buttons.

Enable required APIs

1. Switch to the dedicated project.
2. Go to the **Library** tab.
3. Search for and select **Cloud Logging API**, and then on the API page, select **ENABLE**.
4. Search for and select **Cloud Pub/Sub API**, and then on the API page, select **ENABLE**.

(!) Note

Make sure that you do not select Pub/Sub Lite API.

Create a dedicated service account for the security auditing integration

1. Under **IAM & admin**, select **Service accounts**.
2. Select **CREATE SERVICE ACCOUNT** to create a dedicated service account.
3. Enter an account name, and then select **Create**.
4. Specify the **Role** as **Pub/Sub Admin** and then select **Save**.

Create service account

Service account details — 2 Grant this service account access to project (optional) — 3 Grant users access to this service account (optional)

Service account permissions (optional)
Grant this service account access to MCAS for GCP - dev so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

Role: Pub/Sub Admin

Full access to topics, subscriptions, and snapshots.

+ ADD ANOTHER ROLE

CONTINUE CANCEL

5. Copy the **Email** value, you'll need this later.

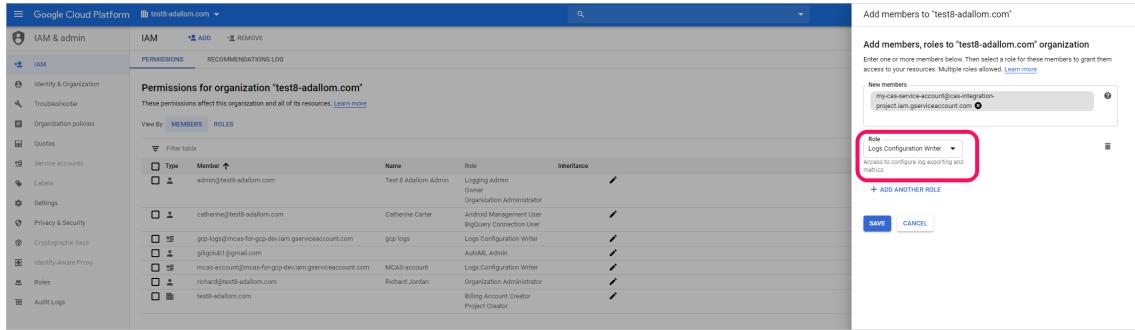
Service accounts	+ CREATE SERVICE ACCOUNT	DELETE
Service accounts for project "CAS-Integration-project" A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more		
Email	Status	Name ↑
cas-service-account@cas-integration-project.iam.gserviceaccount.com	Green checkmark	Cas-Service-Account

6. Under **IAM & admin**, select **IAM**.
 - a. Switch to organization level.

b. Select **ADD**.

c. In the **New members** box, paste the **Email** value you copied earlier.

d. Specify the **Role** as **Logs Configuration Writer** and then select **Save**.



The screenshot shows the Google Cloud Platform IAM & admin interface. The left sidebar is expanded to show 'Service accounts'. The main area is titled 'Permissions for organization "test8-adaliam.com"'. It shows a table of members with their roles. A new member is being added, with the email 'my-cas-service-account@cas-integration-project.iam.gserviceaccount.com' in the 'New members' input field. The 'Role' dropdown is set to 'Logs Configuration Writer'. The 'SAVE' button is highlighted with a red box.

Create a private key for the dedicated service account

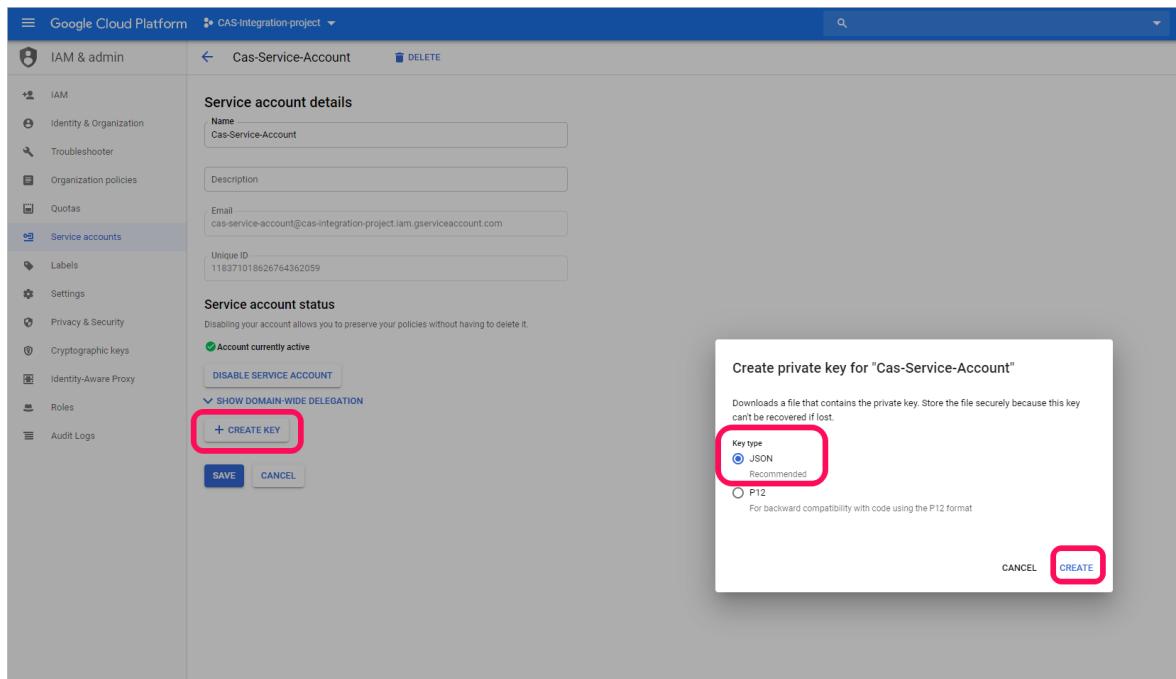
1. Switch to project level.

2. Under **IAM & admin**, select **Service accounts**.

3. Open the dedicated service account and select **Edit**.

4. Select **CREATE KEY**.

5. In the **Create private key** screen, select **JSON**, and then select **CREATE**.



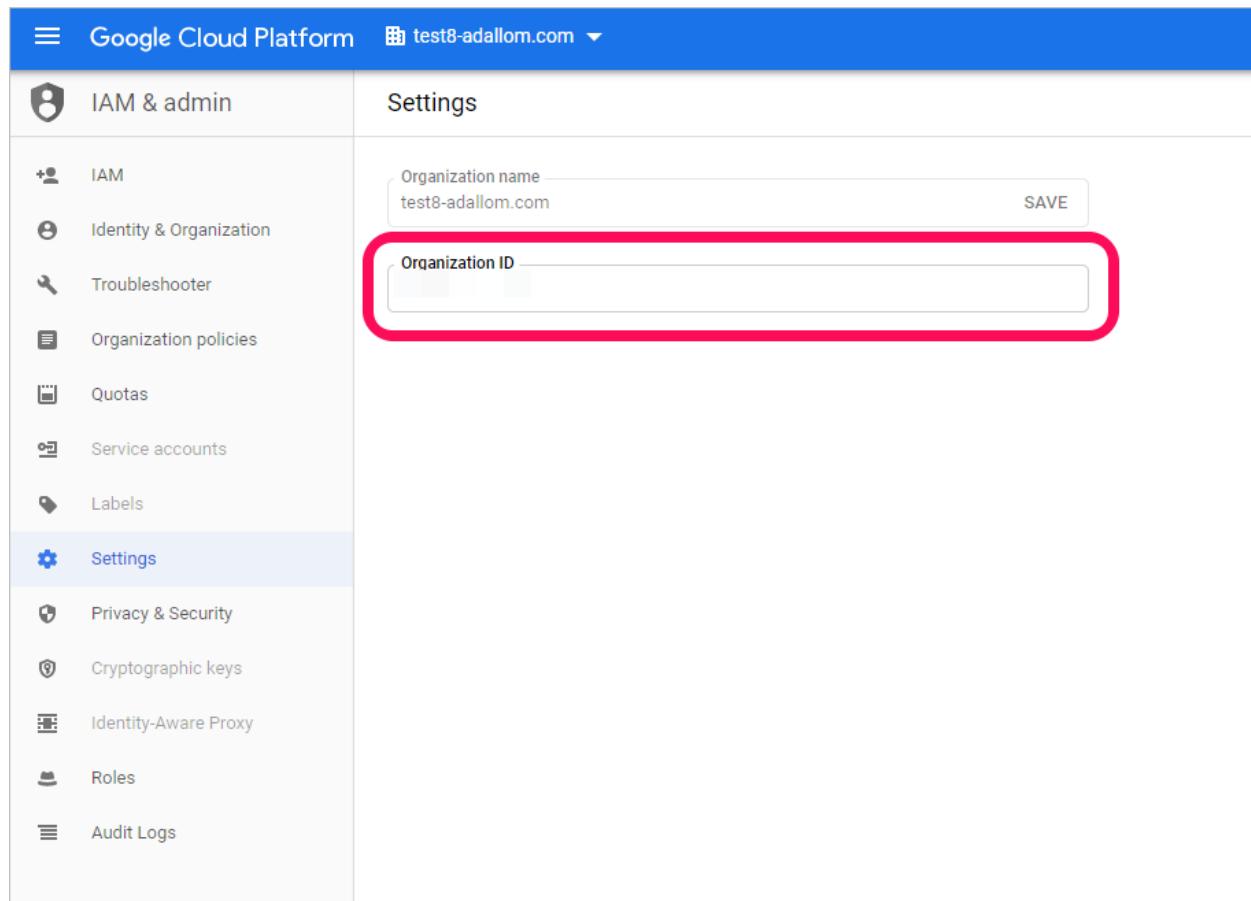
The screenshot shows the Google Cloud Platform IAM & admin interface. The left sidebar is expanded to show 'Service accounts'. A service account named 'Cas-Service-Account' is selected. The 'Service account details' section shows the name, email, and unique ID. The 'Service account status' section shows the account is currently active. A modal window titled 'Create private key for "Cas-Service-Account"' is open. The 'Key type' dropdown is set to 'JSON' (highlighted with a red box). The 'CREATE' button is highlighted with a red box.

! Note

You'll need the JSON file that is downloaded to your device later.

Retrieve your Organization ID

Make a note of your **Organization ID**, you'll need this later. For more information, see [Getting your organization ID](#).



The screenshot shows the Google Cloud Platform IAM & admin Settings page. On the left, there is a sidebar with various options: IAM, Identity & Organization, Troubleshooter, Organization policies, Quotas, Service accounts, Labels, Settings (which is selected and highlighted in blue), Privacy & Security, Cryptographic keys, Identity-Aware Proxy, Roles, and Audit Logs. The main content area is titled 'Settings' and shows the 'Organization name' as 'test8-adallom.com' with a 'SAVE' button. Below it is the 'Organization ID' field, which is highlighted with a red box. The 'Organization ID' field contains the value 'test8-adallom'.

Connect Google Cloud Platform auditing to Defender for Cloud Apps

This procedure describes how to add the GCP connection details to connect Google Cloud Platform auditing to Defender for Cloud Apps.

1. In the Microsoft Defender Portal, select **Settings**. Then choose **Cloud Apps**. Under **Connected apps**, select **App Connectors**.
2. In the **App connectors** page, to provide the GCP connector credentials, do one of the following:

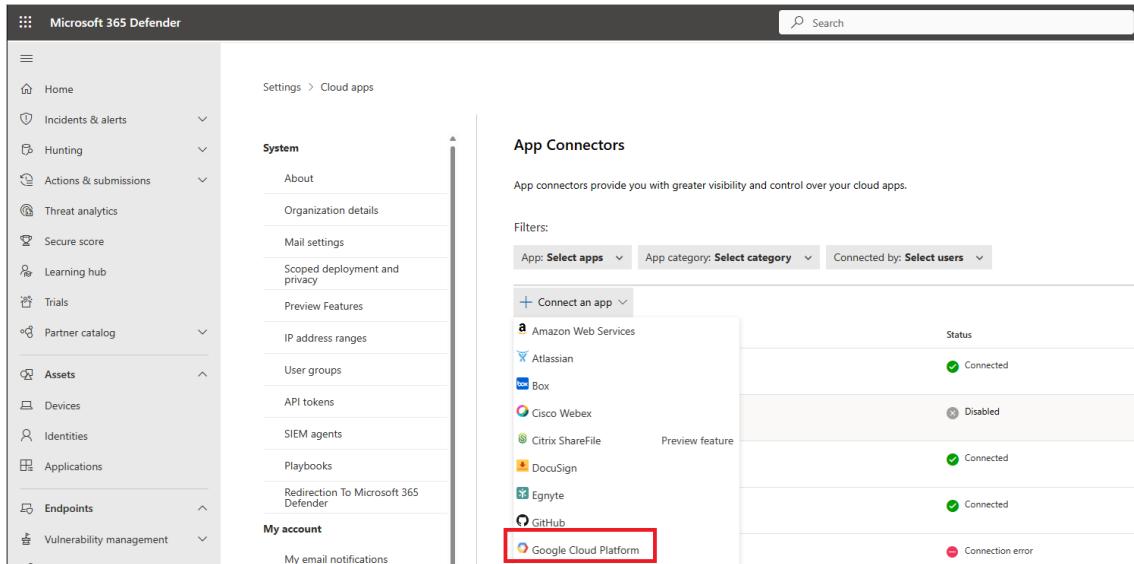
ⓘ Note

We recommend that you connect your Google Workspace instance to get unified user management and governance. This is the recommended even if

you do not use any Google Workspace products and the GCP users are managed via the Google Workspace user management system.

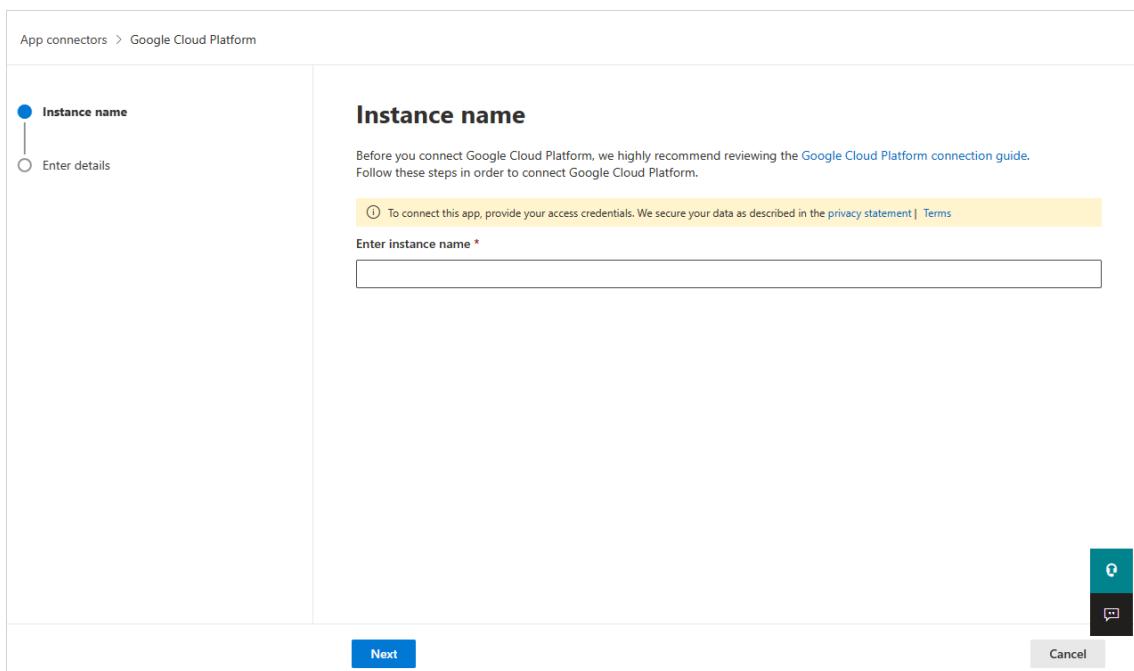
For a new connector

- Select **+Connect an app**, followed by **Google Cloud Platform**.



The screenshot shows the Microsoft 365 Defender Settings interface. On the left, there's a navigation sidebar with various sections like Home, Incidents & alerts, Hunting, Actions & submissions, Threat analytics, Secure score, Learning hub, Trials, Partner catalog, Assets, Devices, Identities, Applications, Endpoints, and Vulnerability management. The 'Cloud apps' section is selected under 'System'. On the right, the 'App Connectors' page is displayed. It has a header with a search bar and a sub-header 'App connectors provide you with greater visibility and control over your cloud apps.' Below this are filters for 'App', 'App category', and 'Connected by'. A button '+ Connect an app' is visible. A list of connectors is shown, including Amazon Web Services (Connected), Atlassian (Disabled), Box (Connected), Cisco Webex (Preview feature), Citrix ShareFile (Connected), DocuSign (Connected), Egnyte (Connected), GitHub (Connected), and Google Cloud Platform (highlighted with a red box). The 'Status' column indicates the connection status for each app.

- In the next window, provide a name for the connector, and then select **Next**.



The screenshot shows the 'Instance name' configuration page for connecting Google Cloud Platform. The left sidebar has a radio button 'Instance name' selected, with 'Enter details' as an alternative. The main area is titled 'Instance name' and contains a note: 'Before you connect this app, we highly recommend reviewing the [Google Cloud Platform connection guide](#). Follow these steps in order to connect Google Cloud Platform.' Below this is a note: 'To connect this app, provide your access credentials. We secure your data as described in the [privacy statement](#) | [Terms](#)'. A text input field 'Enter instance name *' is present. The bottom right of the page has 'Next' and 'Cancel' buttons, and icons for help and feedback.

- In the **Enter details** page, do the following, and then select **Submit**.
 - In the **Organization ID** box, enter the organization you made a note of earlier.
 - In the **Private key file** box, browse to the JSON file you downloaded earlier.

App connectors > Google Cloud Platform

Instance name

Enter details

Enter details

Organization ID *

Private key file No file chosen

?

Back

Submit

Cancel

For an existing connector

- In the list of connectors, on the row in which the GCP connector appears, select **Edit settings**.

App Connectors

App connectors provide you with greater visibility and control over your cloud apps.

Filters: App: Google Cloud, Google Cloud Platform, Google... App category: Select category Connected by: Select users Advanced filters

1 - 2 of 2 connected apps Show details Hide filters Table settings

App	Status	Was connected on	Last activity	Accounts
Google Workspace-US Collaboration	Connected	Dec 28, 2017 1:59 PM	Jan 23, 2023 4:34 PM	11
GCP - US	Connected	Oct 27, 2019 4:53 PM	Jan 19, 2023 12:35 PM	2

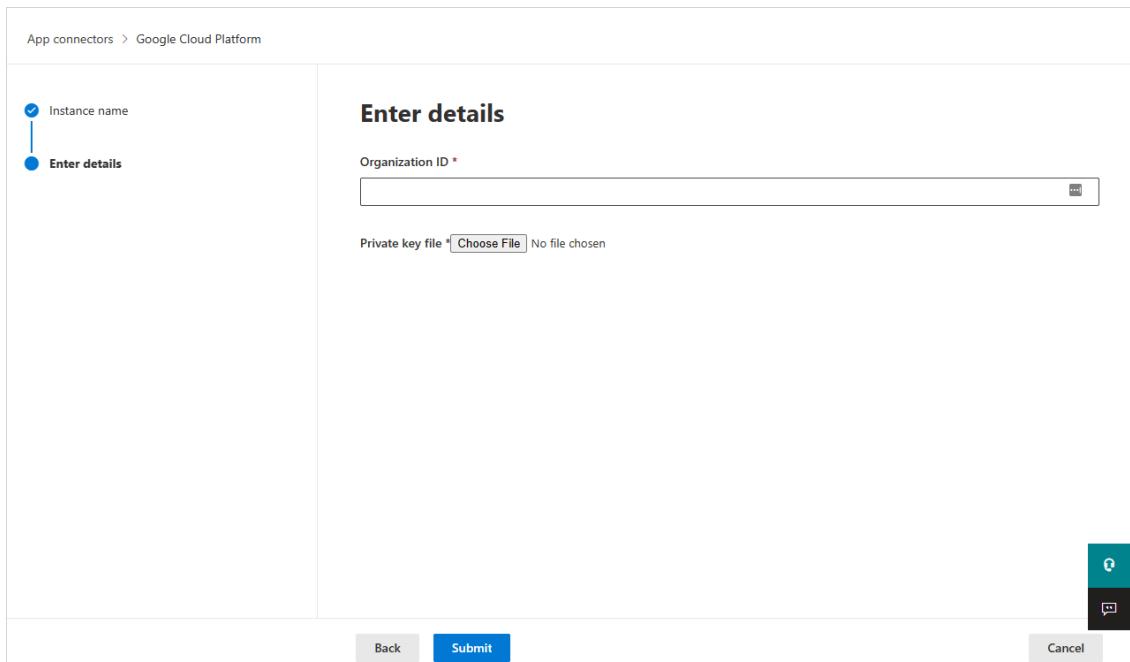
Edit settings

Disable App connector

Edit instance name

Connect Google Cloud Platform instance...

- In the **Edit settings** page, do the following, and then select **Submit**.
 - In the **Organization ID** box, enter the organization you made a note of earlier.
 - In the **Private key file** box, browse to the JSON file you downloaded earlier.



App connectors > Google Cloud Platform

Instance name

Enter details

Enter details

Organization ID *

Private key file No file chosen

Back Cancel

3. In the Microsoft Defender Portal, select **Settings**. Then choose **Cloud Apps**. Under **Connected apps**, select **App Connectors**. Make sure the status of the connected App Connector is **Connected**.

! **Note**

Defender for Cloud Apps will create an aggregated export sink (organization level), a Pub/Sub topic and Pub/Sub subscription using the integration service account in the integration project.

Aggregated export sink is used to aggregate logs across the GCP organization and the Pub/Sub topic created is used as the destination. Defender for Cloud Apps subscribes to this topic through the Pub/Sub subscription created to retrieve the admin activity logs across the GCP organization.

If you have any problems connecting the app, see [Troubleshooting App Connectors](#).

Next steps

[Control cloud apps with policies](#)

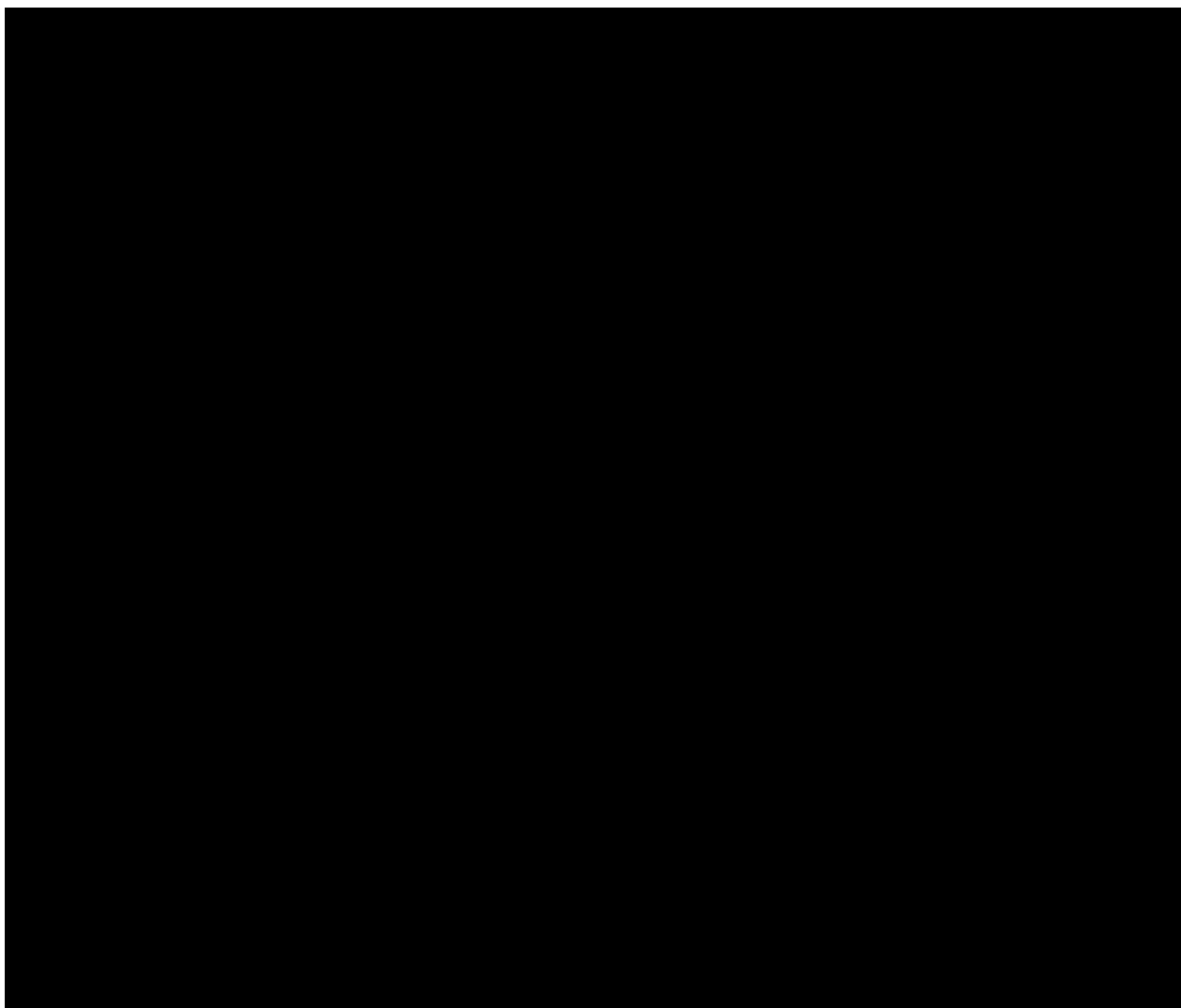
If you run into any problems, we're here to help. To get assistance or support for your product issue, please [open a support ticket](#).

Protecting containers in GCP with Defender for Containers

Article • 04/27/2023

Episode description: In this episode of Defender for Cloud in the field, Nadav Wolfin joins Yuri Diogenes to talk about how to use Defender for Containers to protect Containers that are located at Google Cloud (GCP).

Nadav gives insights about workload protection for GKE and how to obtain visibility of this type of workload across Azure and AWS. Nadav also demonstrates the overall onboarding experience and provides an overview of the architecture of this solution.



- 00:55 - Architecture solution for Defender for Containers and support for GKE
- 06:42 - How the onboard process works
- 08:46 - Demonstration

- 26:18 - Integration with Azure Arc

Recommended resources

Learn how to [Enable Microsoft Defender for Containers](#).

- Subscribe to [Microsoft Security on YouTube](#)
- Follow us on social media: [LinkedIn](#) [Twitter](#)
- Join our [Tech Community](#)
- For more about [Microsoft Security](#)

Next steps

[Threat landscape for Containers](#)

Cloud Design Patterns

Article • 04/13/2023

These design patterns are useful for building reliable, scalable, secure applications in the cloud.

Each pattern describes the problem that the pattern addresses, considerations for applying the pattern, and an example based on Microsoft Azure. Most patterns include code samples or snippets that show how to implement the pattern on Azure. However, most patterns are relevant to any distributed system, whether hosted on Azure or other cloud platforms.

Challenges in cloud development

Data Management

Data management is the key element of cloud applications, and it influences most of the quality attributes. Data is typically hosted in different locations and across multiple servers for performance, scalability or availability. This can present various challenges. For example, data consistency must be maintained, and data will typically need to be synchronized across different locations.

Design and Implementation

Good design encompasses consistency and coherence in component design and deployment, maintainability to simplify administration and development, and reusability to allow components and subsystems to be used in other applications and scenarios. Decisions made during the design and implementation phase significantly impact the quality and total cost of ownership of cloud-hosted applications and services.

Messaging

The distributed nature of cloud applications requires a messaging infrastructure that connects the components and services, ideally loosely coupled to maximize scalability. Asynchronous messaging is widely used and provides many benefits, but it also brings challenges such as ordering messages, poison message management, idempotency, and more.

Catalog of patterns

Pattern	Summary	Category
Ambassador	Create helper services that send network requests on behalf of a consumer service or application.	Design and Implementation , Operational Excellence
Anti-Corruption Layer	Implement a façade or adapter layer between a modern application and a legacy system.	Design and Implementation , Operational Excellence
Asynchronous Request-Reply	Decouple backend processing from a frontend host, where backend processing needs to be asynchronous, but the frontend still needs a clear response.	Messaging
Backends for Frontends	Create separate backend services to be consumed by specific frontend applications or interfaces.	Design and Implementation
Bulkhead	Isolate elements of an application into pools so that if one fails, the others will continue to function.	Reliability
Cache-Aside	Load data on demand into a cache from a data store	Data Management , Performance Efficiency
Choreography	Let each service decide when and how a business operation is processed, instead of depending on a central orchestrator.	Messaging , Performance Efficiency
Circuit Breaker	Handle faults that might take a variable amount of time to fix when connecting to a remote service or resource.	Reliability
Claim Check	Split a large message into a claim check and a payload to avoid overwhelming a message bus.	Messaging
Compensating Transaction	Undo the work performed by a series of steps, which together define an eventually consistent operation.	Reliability
Competing Consumers	Enable multiple concurrent consumers to process messages received on the same messaging channel.	Messaging
Compute Resource Consolidation	Consolidate multiple tasks or operations into a single computational unit	Design and Implementation

Pattern	Summary	Category
CQRS	Segregate operations that read data from operations that update data by using separate interfaces.	Data Management, Design and Implementation, Performance Efficiency
Deployment Stamps	Deploy multiple independent copies of application components, including data stores.	Reliability, Performance Efficiency
Edge Workload Configuration	The great variety of systems and devices on the shop floor can make workload configuration a difficult problem.	Design and Implementation
Event Sourcing	Use an append-only store to record the full series of events that describe actions taken on data in a domain.	Data Management, Performance Efficiency
External Configuration Store	Move configuration information out of the application deployment package to a centralized location.	Design and Implementation, Operational Excellence
Federated Identity	Delegate authentication to an external identity provider.	Security
Gatekeeper	Protect applications and services by using a dedicated host instance that acts as a broker between clients and the application or service, validates and sanitizes requests, and passes requests and data between them.	Security
Gateway Aggregation	Use a gateway to aggregate multiple individual requests into a single request.	Design and Implementation, Operational Excellence
Gateway Offloading	Offload shared or specialized service functionality to a gateway proxy.	Design and Implementation, Operational Excellence
Gateway Routing	Route requests to multiple services using a single endpoint.	Design and Implementation, Operational Excellence

Pattern	Summary	Category
Geodes	Deploy backend services into a set of geographical nodes, each of which can service any client request in any region.	Reliability , Operational Excellence
Health Endpoint Monitoring	Implement functional checks in an application that external tools can access through exposed endpoints at regular intervals.	Reliability , Operational Excellence
Index Table	Create indexes over the fields in data stores that are frequently referenced by queries.	Data Management , Performance Efficiency
Leader Election	Coordinate the actions performed by a collection of collaborating task instances in a distributed application by electing one instance as the leader that assumes responsibility for managing the other instances.	Design and Implementation , Reliability
Materialized View	Generate prepopulated views over the data in one or more data stores when the data isn't ideally formatted for required query operations.	Data Management , Operational Excellence , Performance Efficiency
Pipes and Filters	Break down a task that performs complex processing into a series of separate elements that can be reused.	Design and Implementation , Messaging
Priority Queue	Prioritize requests sent to services so that requests with a higher priority are received and processed more quickly than those with a lower priority.	Messaging , Performance Efficiency
Publisher/Subscriber	Enable an application to announce events to multiple interested consumers asynchronously, without coupling the senders to the receivers.	Messaging
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.	Reliability , Messaging , Resiliency , Performance Efficiency
Rate Limit Pattern	Limiting pattern to help you avoid or minimize throttling errors related to these throttling limits and to help you more accurately predict throughput.	Reliability

Pattern	Summary	Category
Retry	Enable an application to handle anticipated, temporary failures when it tries to connect to a service or network resource by transparently retrying an operation that's previously failed.	Reliability
Saga	Manage data consistency across microservices in distributed transaction scenarios. A saga is a sequence of transactions that updates each service and publishes a message or event to trigger the next transaction step.	Messaging
Scheduler Agent Supervisor	Coordinate a set of actions across a distributed set of services and other remote resources.	Messaging, Reliability
Sequential Convoy	Process a set of related messages in a defined order, without blocking processing of other groups of messages.	Messaging
Sharding	Divide a data store into a set of horizontal partitions or shards.	Data Management, Performance Efficiency
Sidecar	Deploy components of an application into a separate process or container to provide isolation and encapsulation.	Design and Implementation, Operational Excellence
Static Content Hosting	Deploy static content to a cloud-based storage service that can deliver them directly to the client.	Design and Implementation, Data Management, Performance Efficiency
Strangler Fig	Incrementally migrate a legacy system by gradually replacing specific pieces of functionality with new applications and services.	Design and Implementation, Operational Excellence
Throttling	Control the consumption of resources used by an instance of an application, an individual tenant, or an entire service.	Reliability, Performance Efficiency
Valet Key	Use a token or key that provides clients with restricted direct access to a specific resource or service.	Data Management, Security

Data management patterns

Article • 12/16/2022

Data management is the key element of cloud applications, and influences most of the quality attributes. Data is typically hosted in different locations and across multiple servers for reasons such as performance, scalability or availability, and this can present a range of challenges. For example, data consistency must be maintained, and data will typically need to be synchronized across different locations.

Additionally data should be protected at rest, in transit, and via authorized access mechanisms to maintain security assurances of confidentiality, integrity, and availability. Refer to the Azure Security Benchmark [Data Protection Control](#) for more information.

Pattern	Summary
Cache-Aside	Load data on demand into a cache from a data store
CQRS	Segregate operations that read data from operations that update data by using separate interfaces.
Event Sourcing	Use an append-only store to record the full series of events that describe actions taken on data in a domain.
Index Table	Create indexes over the fields in data stores that are frequently referenced by queries.
Materialized View	Generate prepopulated views over the data in one or more data stores when the data isn't ideally formatted for required query operations.
Sharding	Divide a data store into a set of horizontal partitions or shards.
Static Content Hosting	Deploy static content to a cloud-based storage service that can deliver them directly to the client.
Valet Key	Use a token or key that provides clients with restricted direct access to a specific resource or service.

Design and implementation patterns

Article • 04/13/2023

Good design encompasses factors such as consistency and coherence in component design and deployment, maintainability to simplify administration and development, and reusability to allow components and subsystems to be used in other applications and in other scenarios. Decisions made during the design and implementation phase have a huge impact on the quality and the total cost of ownership of cloud hosted applications and services.

Pattern	Summary
Ambassador	Create helper services that send network requests on behalf of a consumer service or application.
Anti-Corruption Layer	Implement a façade or adapter layer between a modern application and a legacy system.
Backends for Frontends	Create separate backend services to be consumed by specific frontend applications or interfaces.
CQRS	Segregate operations that read data from operations that update data by using separate interfaces.
Compute Resource Consolidation	Consolidate multiple tasks or operations into a single computational unit
Edge Workload Configuration	The great variety of systems and devices on the shop floor can make workload configuration a difficult problem.
External Configuration Store	Move configuration information out of the application deployment package to a centralized location.
Gateway Aggregation	Use a gateway to aggregate multiple individual requests into a single request.
Gateway Offloading	Offload shared or specialized service functionality to a gateway proxy.
Gateway Routing	Route requests to multiple services using a single endpoint.

Pattern	Summary
Leader Election	Coordinate the actions performed by a collection of collaborating task instances in a distributed application by electing one instance as the leader that assumes responsibility for managing the other instances.
Pipes and Filters	Break down a task that performs complex processing into a series of separate elements that can be reused.
Sidecar	Deploy components of an application into a separate process or container to provide isolation and encapsulation.
Static Content Hosting	Deploy static content to a cloud-based storage service that can deliver them directly to the client.
Strangler Fig	Incrementally migrate a legacy system by gradually replacing specific pieces of functionality with new applications and services.

Messaging patterns

Article • 04/13/2023

The distributed nature of cloud applications requires a messaging infrastructure that connects the components and services, ideally in a loosely coupled manner in order to maximize scalability. Asynchronous messaging is widely used, and provides many benefits, but also brings challenges such as the ordering of messages, poison message management, idempotency, and more.

Pattern	Summary
Asynchronous Request-Reply	Decouple backend processing from a frontend host, where backend processing needs to be asynchronous, but the frontend still needs a clear response.
Claim Check	Split a large message into a claim check and a payload to avoid overwhelming a message bus.
Choreography	Have each component of the system participate in the decision-making process about the workflow of a business transaction, instead of relying on a central point of control.
Competing Consumers	Enable multiple concurrent consumers to process messages received on the same messaging channel.
Pipes and Filters	Break down a task that performs complex processing into a series of separate elements that can be reused.
Priority Queue	Prioritize requests sent to services so that requests with a higher priority are received and processed more quickly than those with a lower priority.
Publisher-Subscriber	Enable an application to announce events to multiple interested consumers asynchronously, without coupling the senders to the receivers.
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.
Saga	Manage data consistency across microservices in distributed transaction scenarios. A saga is a sequence of transactions that updates each service and publishes a message or event to trigger the next transaction step.
Scheduler Agent Supervisor	Coordinate a set of actions across a distributed set of services and other remote resources.
Sequential Convoy	Process a set of related messages in a defined order, without blocking processing of other groups of messages.

Pattern implementation for network secure ingress

Article • 08/01/2023

Network secure ingress encapsulates several design patterns, including the patterns for global routing, global offloading, and health endpoint monitoring. You can use the pattern implementation in this article as a gateway for any HTTP or HTTPS workload that requires high availability or reliability by providing secure global routing to workloads in differing regions with low-latency failover.

Video: Network secure ingress implementation

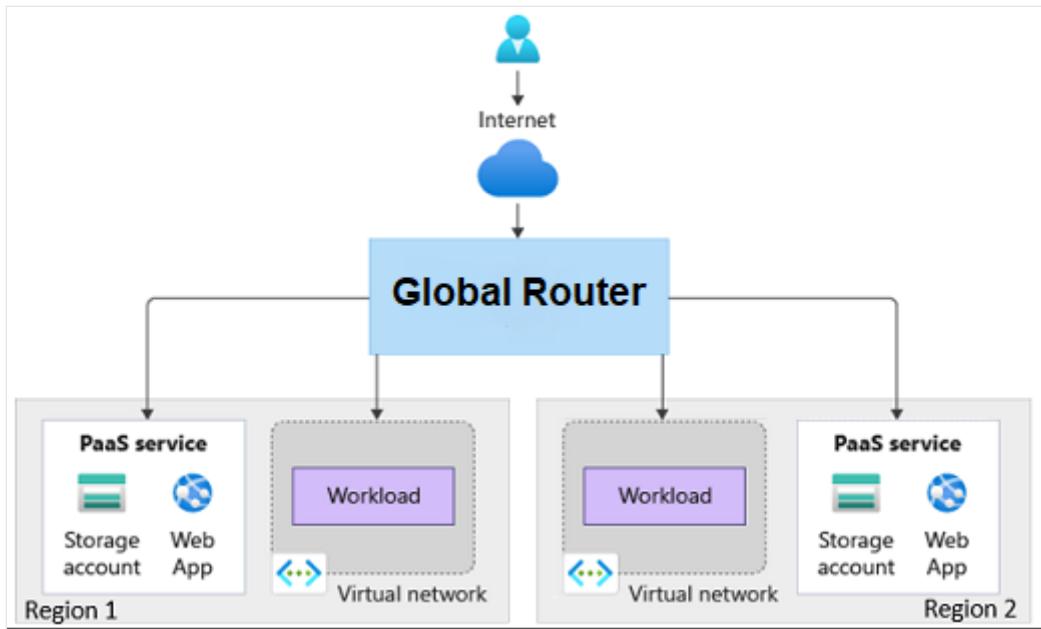
<https://learn-video.azurefd.net/vod/player?id=163c161c-0f7e-4fea-b126-c8f540fc84e0&embedUrl=%2Fazure%2Farchitecture%2Fpattern-implementations%2Fnetwork-secure-ingress&locale=en-us>

Pattern requirements

This article describes three requirements that the pattern implementation for network secure ingress focuses on: global routing, low-latency failover, and mitigating attacks at the edge.

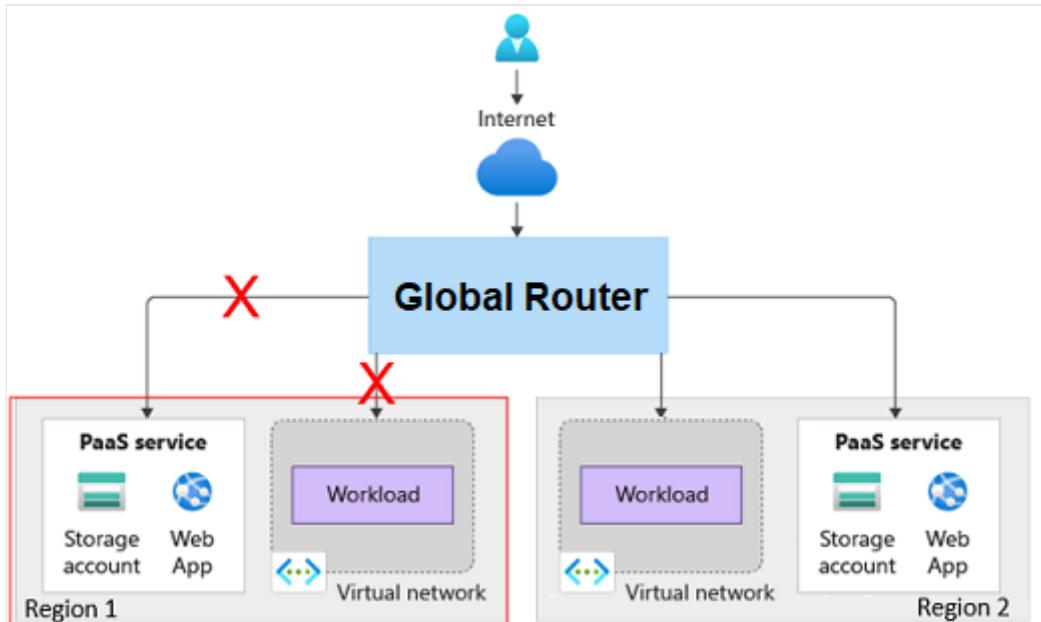
Global routing

The network secure ingress pattern encapsulates the global routing pattern. As such, the implementation can route requests to workloads in different regions.



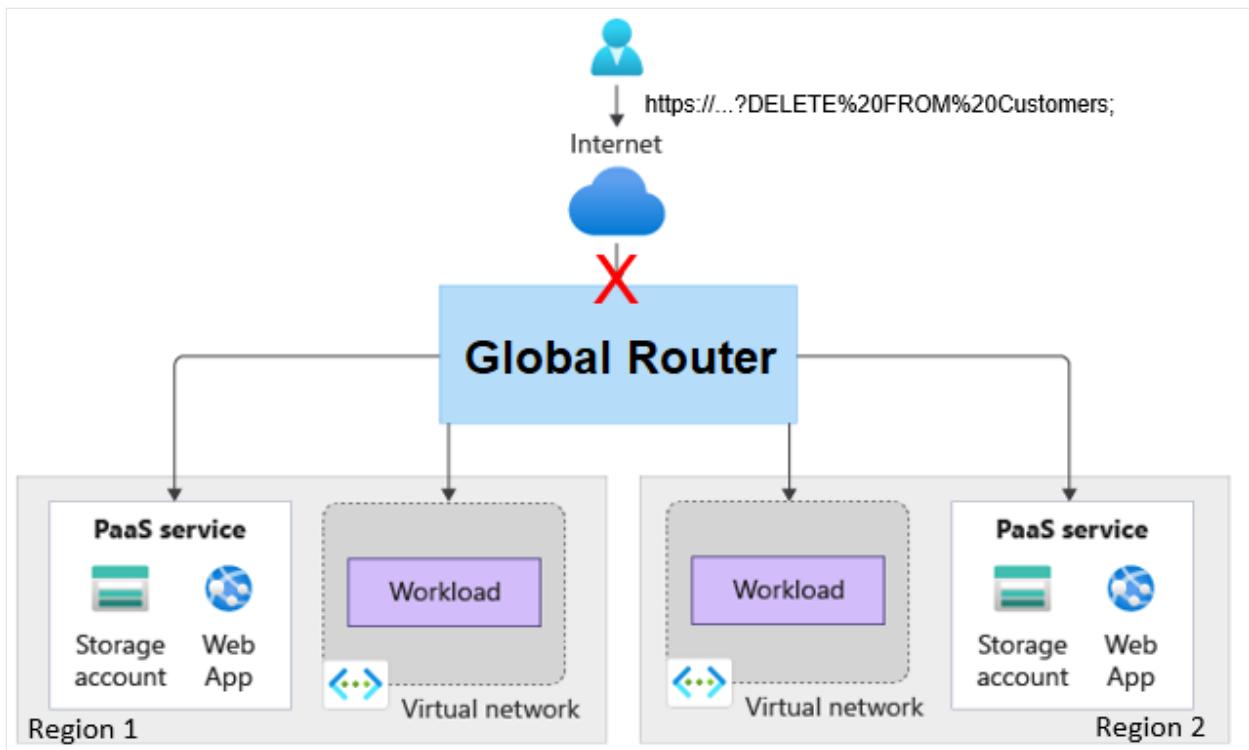
Low-latency failover

The implementation must be able to identify healthy and unhealthy workloads and adjust the routing accordingly in a time-sensitive way. The latency should be able to support adjusting the routing in a matter of minutes.



Mitigating attacks at the edge

Mitigating attacks at the edge necessitates the "network secure" part of the implementation. The workloads or platform as a service (PaaS) services shouldn't be accessible via the internet. Internet traffic should only be able to route through the gateway. The gateway should have the ability to mitigate exploits.

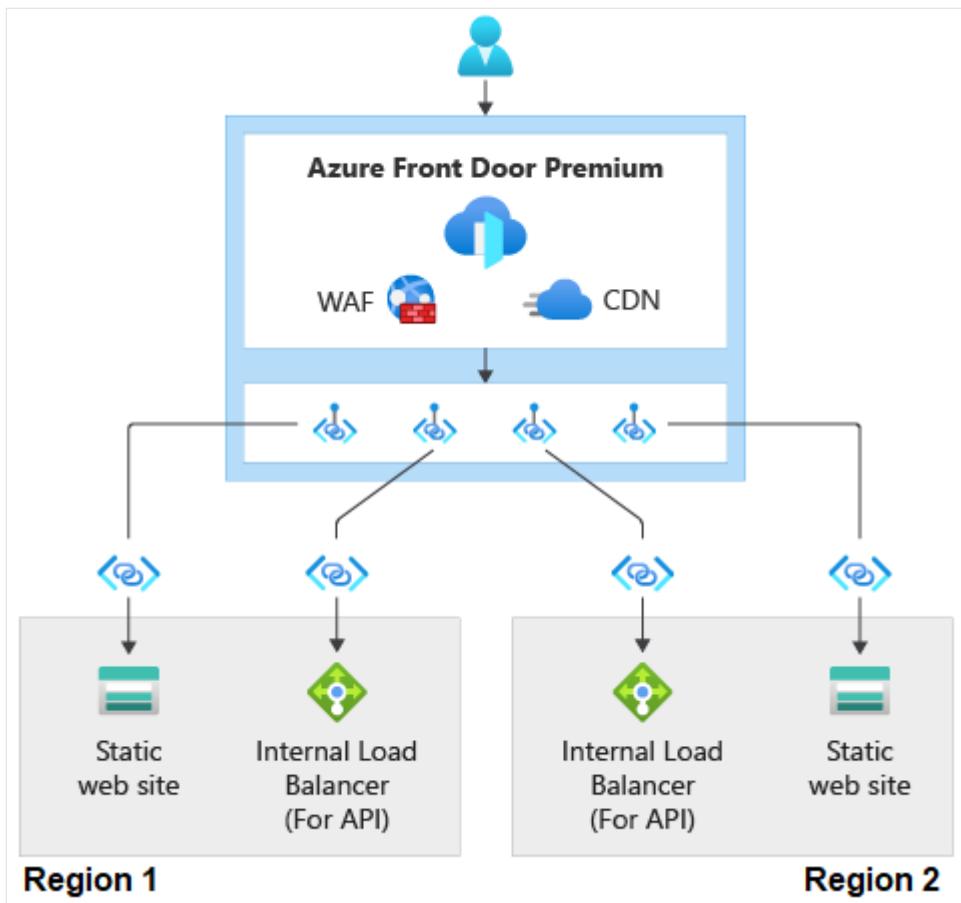


Patterns

This solution implements the following design patterns:

- **Gateway routing pattern:** Route requests to multiple services or service instances that can reside in different regions.
- **Gateway offloading pattern:** Offload functionality, such as mitigating attacks, to a gateway proxy.
- **Health endpoint monitoring pattern:** Expose endpoints that validate the health of the workload.

Design



This implementation includes the following details:

- It uses Azure Blob Storage accounts to simulate static web workloads running in two regions. This implementation doesn't include any workloads running behind an internal load balancer (ILB). The diagram shows an ILB to illustrate that this implementation would work for private workloads running behind an ILB.
- It uses Azure Front Door Premium tier as the global gateway.
- The Azure Front Door instance has a global web application firewall (WAF) policy configured with managed rules that help protect against common exploits.
- The storage accounts aren't exposed over the internet.
- The Azure Front Door Premium tier accesses the storage accounts via Azure Private Link.
- The Azure Front Door instance has the following high-level configuration:
 - An endpoint with a single route that points to a single origin group. An origin group is a collection of origins or back ends.
 - The origin group has an origin configured to point to each storage account.
 - Each origin requests Private Link access to the storage account.
 - The origin group has health probes configured to access an HTML page in the storage accounts. The HTML page is acting as the health endpoint for the static workloads. If the probes can successfully access the origin in three out of the last four attempts, the origin is deemed healthy.

Components

Web request

- [Azure Web Application Firewall](#): The Premium tier of Web Application Firewall supports Microsoft-managed rules that help protect against common exploits.
- [Azure Private Link](#): Private endpoints in Azure Private Link expose an Azure PaaS service to a private IP address in a virtual network. This exposure allows the communication to flow across the Microsoft backbone network and not on the public internet.
- [Azure Front Door Premium tier](#): Azure Front Door provides Layer 7 global load balancing. Azure Front Door has integration with Web Application Firewall. The Premium tier supports:
 - [Azure Private Link](#): Private Link support allows Azure Front Door to communicate with PaaS services or workloads running in a private virtual network over the Microsoft backbone network.
 - [Microsoft-managed rule sets](#): The premium tier of Azure Front Door supports the premium tier of Web Application Firewall, which supports the managed rule set in the WAF.
- [Azure Storage](#): This implementation uses Blob Storage accounts to represent a static website or workload.
- [Internal load balancer](#): This implementation doesn't use the internal load balancer. It's pictured to represent a private workload running behind that load balancer. The routing to the storage account is the same as it would be to load balancers.

Operations

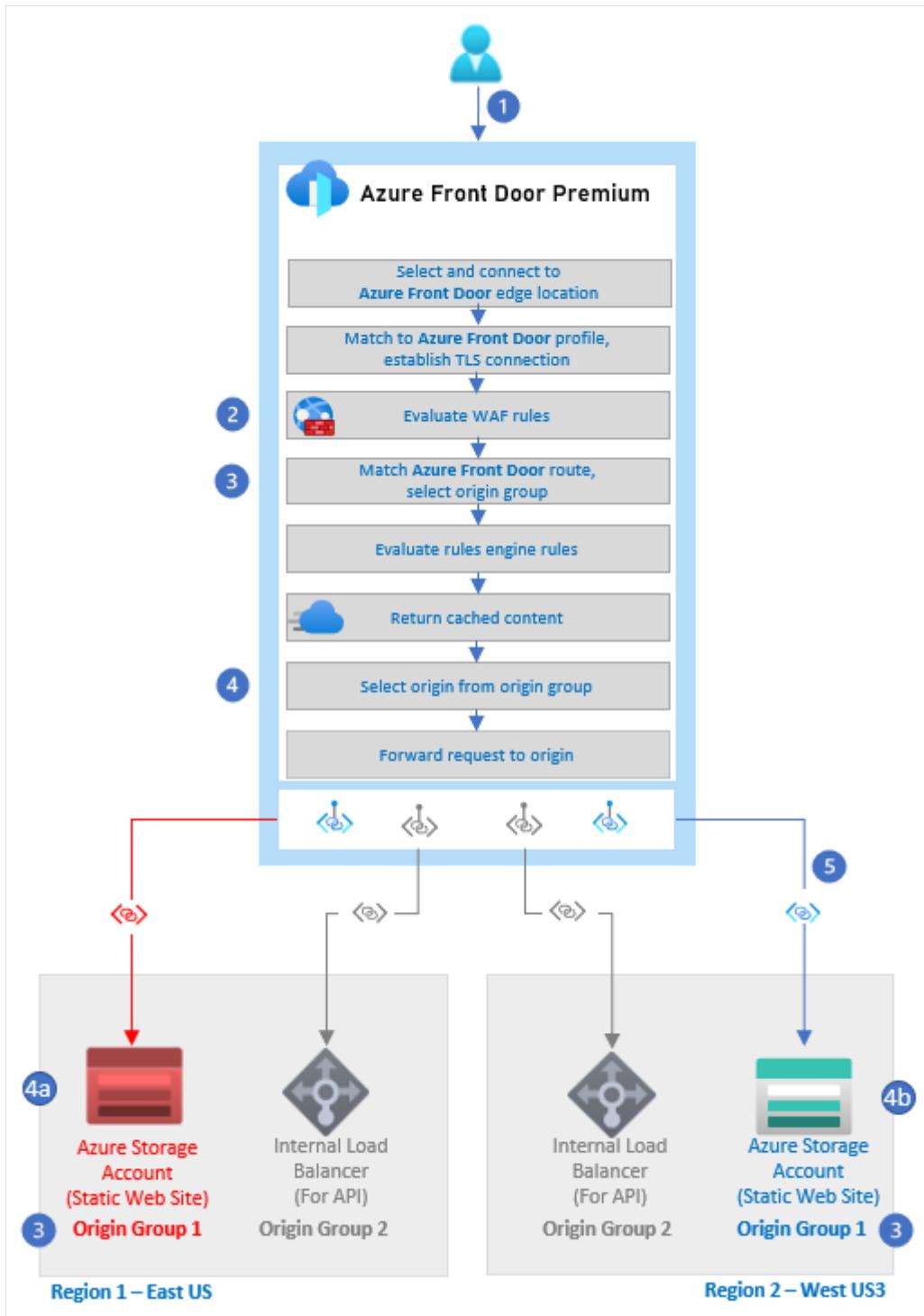
Securing resources from a network perspective helps protect against exploits, but it also isolates the resources from processes or administrators who might need to access those resources. For example, a build agent in a DevOps pipeline might need to access the storage account in order to deploy an update to the web application. Also, an administrator might need to access the resource for troubleshooting purposes.

To illustrate providing access to network secure access for operational purposes, this implementation deploys a virtual machine (VM) in a virtual network that has Private Link access to the storage accounts. This implementation deploys Azure Bastion, which the administrator can use to connect to the VM. For the deployment scenario, a private build agent could be deployed to the virtual network, similar to how the VM was.

Here are details about the components for operations:

- [Azure Virtual Network](#): This implementation uses the virtual network to contain the components required for an administrator to securely communicate with the storage account over the private Microsoft backbone network.
- [Azure Virtual Machines](#): This implementation uses a VM as a jumpbox for administrators to connect to. The VM is deployed in the private virtual network.
- [Azure Bastion](#): Azure Bastion allows the administrator to securely connect to the jumpbox VM over Secure Shell (SSH) without requiring the VM to have a public IP address.
- [Private Link endpoint](#): The private endpoint is assigned a private IP address from the virtual network and connects to the storage account PaaS service. This connection allows resources in the private virtual network to communicate with the storage account over the private IP address.
- [Private Azure DNS zone](#): The private Azure DNS zone is a DNS service that's used to resolve the Azure storage account's Private Link host name to the private endpoint's private IP address.

Web request flow



1. The user issues an HTTP or HTTPS request to an Azure Front Door endpoint.
2. The WAF rules are evaluated. Rules that match are always logged. If the Azure Front Door WAF policy mode is set to *prevention* and the matching rule has an action set to *block on anomaly*, the request is blocked. Otherwise, the request continues or is redirected, or the subsequent rules are evaluated.
3. The route configured in Azure Front Door is matched and the correct origin group is selected. In this example, the path was to the static content in the website.
4. The origin is selected from the origin group.

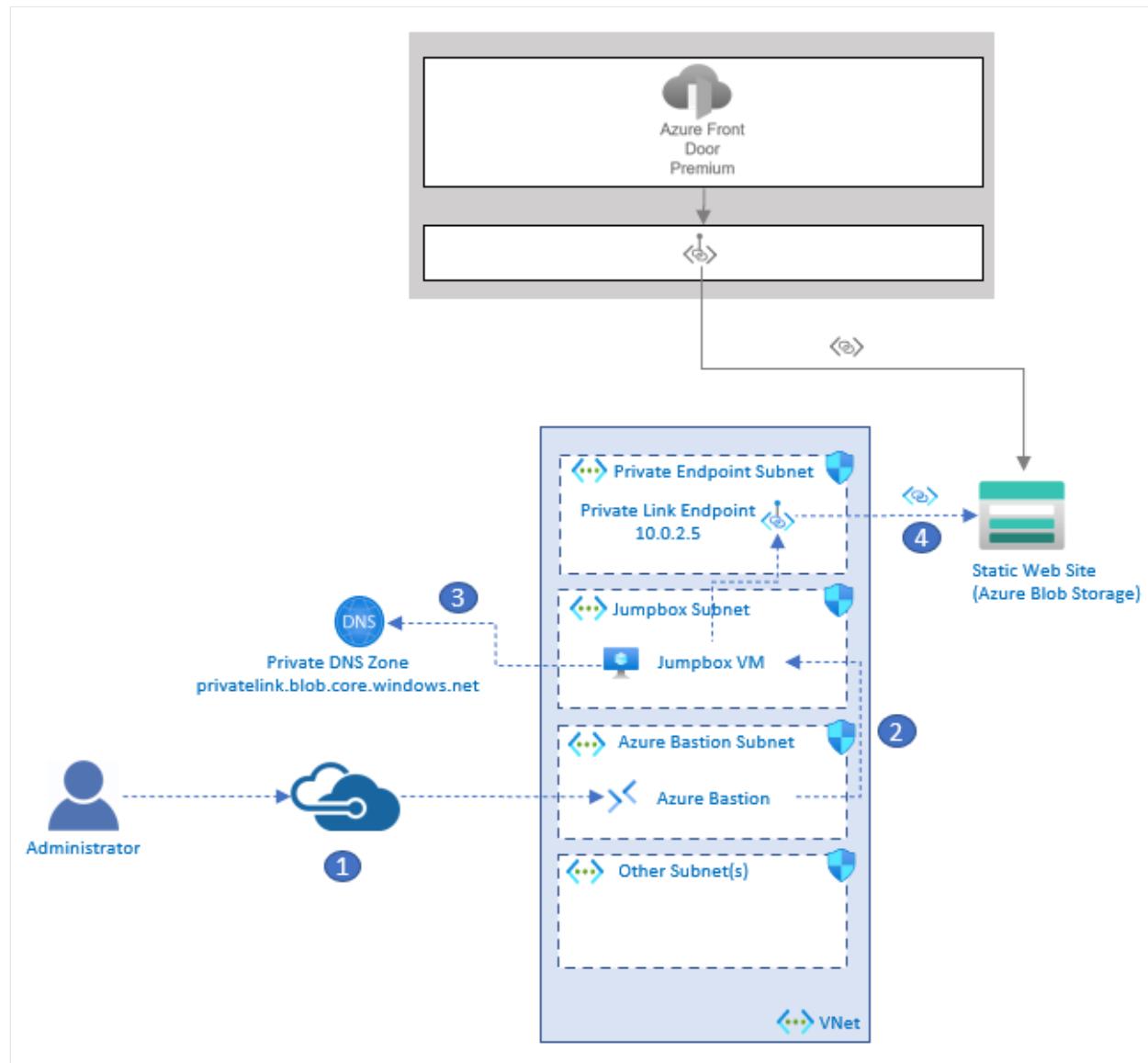
a. In this example, the health probes deemed the website unhealthy, so it's eliminated from the possible origins.

b. This website is selected.

5. The request is routed to the Azure storage account via Private Link over the Microsoft backbone network.

For more information about the Azure Front Door routing architecture, see [Routing architecture overview](#).

Operational flow



1. An administrator connects to the Azure Bastion instance that's deployed in the virtual network.
2. Azure Bastion provides SSH connectivity to the jumpbox VM.

3. The administrator on the jumpbox tries to access the storage account via the Azure CLI. The jumpbox queries DNS for the public Azure Blob Storage account endpoint: storageaccountname.blob.core.windows.net.

Private DNS ultimately resolves to storageaccountname.privatelink.blob.core.windows.net. It returns the private IP address of the Private Link endpoint, which is 10.0.2.5 in this example.
4. A private connection to the storage account is established through the Private Link endpoint.

Considerations

Keep the following points in mind when you use this solution.

Reliability

Reliability ensures that your application can meet the commitments that you make to your customers. For more information, see [Overview of the reliability pillar](#).

This scenario addresses the following key points about reliability:

- Global routing with low latency, through the use of health probes, enables reliability by insulating the application against regional outages.
- [Web Application Firewall on Azure Front Door](#) provides centralized protection for HTTP and HTTPS requests.

Security

Security provides assurances against deliberate attacks and the abuse of your valuable data and systems. For more information, see [Overview of the security pillar](#).

This scenario addresses the following key points about security:

- [Private Link support in Azure Front Door Premium](#) eliminates the need to expose your internal or PaaS services over the internet. Private Link allows Azure Front Door to communicate to your private services or PaaS services over the Microsoft backbone network.
- [Web Application Firewall on Azure Front Door](#) provides centralized protection for HTTP and HTTPS requests.
- [Managed rules in Web Application Firewall Premium](#) are Microsoft-managed rules that help protect you against a common set of security threats.

Cost optimization

Cost optimization is about looking at ways to reduce unnecessary expenses and improve operational efficiencies. For more information, see [Overview of the cost optimization pillar](#).

Although both Azure Front Door Premium and Web Application Firewall Premium provide advanced security features over the Standard tier, there's additional cost to both. Review the following resources to learn more about pricing for Azure Front Door and Web Application Firewall:

- [Azure Front Door pricing](#)
- [Web Application Firewall pricing](#)
- [Azure pricing calculator](#)

Operational excellence

Operational excellence covers the operations processes that deploy an application and keep it running in production. For more information, see [Overview of the operational excellence pillar](#).

Implementing network security boundaries adds complexity to operations and deployment. Keep these points in mind:

- The [IP ranges for Microsoft-hosted agents vary over time](#). Consider implementing self-hosted agents in your virtual network.
- Implement [Azure Bastion](#) for scenarios where operations teams need to access network secure resources.
- The use of [Web Application Firewall on Azure Front Door](#) to provide centralized protection for HTTP and HTTPS requests is an example of the gateway offloading pattern. The responsibility of examining requests for exploits is offloaded to Web Application Firewall in Azure Front Door. The benefit from an operational excellence perspective is that you need to manage the rules in only one place.

Important

The [network secure ingress sample](#) allows you to deploy all of the resources required for you to connect to a jumpbox through Azure Bastion and connect to a network secure VM.

Performance efficiency

Performance efficiency is the ability of your workload to scale to meet the demands that users place on it. For more information, see [Overview of the performance efficiency pillar](#).

Global routing enables horizontal scaling through the deployment of more resources in the same region or different regions.

Next steps

- Deploy this implementation by following the steps outlined in the [network secured ingress sample](#).

Ambassador pattern

Azure

Create helper services that send network requests on behalf of a consumer service or application. An ambassador service can be thought of as an out-of-process proxy that is co-located with the client.

This pattern can be useful for offloading common client connectivity tasks such as monitoring, logging, routing, security (such as TLS), and [resiliency patterns](#) in a language agnostic way. It is often used with legacy applications, or other applications that are difficult to modify, in order to extend their networking capabilities. It can also enable a specialized team to implement those features.

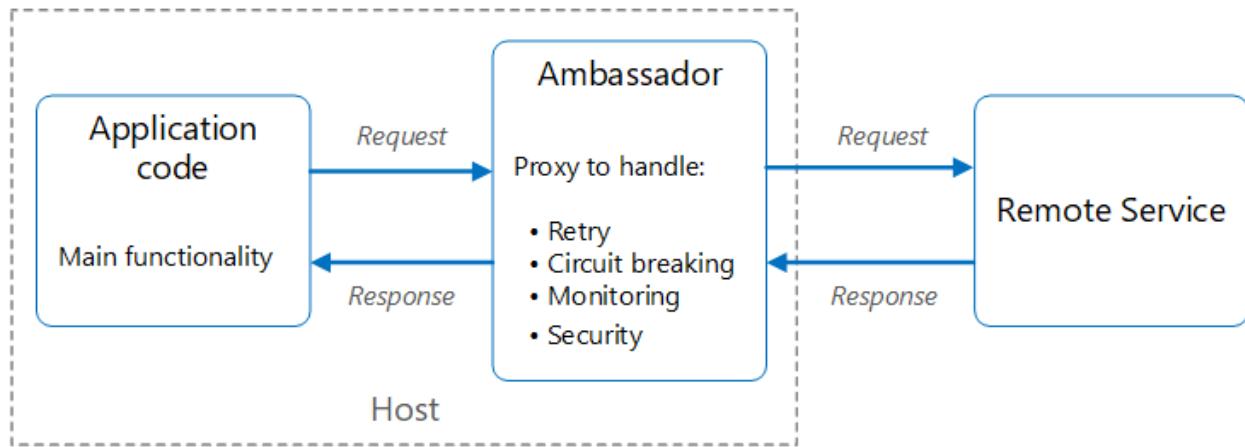
Context and problem

Resilient cloud-based applications require features such as [circuit breaking](#), routing, metering and monitoring, and the ability to make network-related configuration updates. It may be difficult or impossible to update legacy applications or existing code libraries to add these features, because the code is no longer maintained or can't be easily modified by the development team.

Network calls may also require substantial configuration for connection, authentication, and authorization. If these calls are used across multiple applications, built using multiple languages and frameworks, the calls must be configured for each of these instances. In addition, network and security functionality may need to be managed by a central team within your organization. With a large code base, it can be risky for that team to update application code they aren't familiar with.

Solution

Put client frameworks and libraries into an external process that acts as a proxy between your application and external services. Deploy the proxy on the same host environment as your application to allow control over routing, resiliency, security features, and to avoid any host-related access restrictions. You can also use the ambassador pattern to standardize and extend instrumentation. The proxy can monitor performance metrics such as latency or resource usage, and this monitoring happens in the same host environment as the application.



Features that are offloaded to the ambassador can be managed independently of the application. You can update and modify the ambassador without disturbing the application's legacy functionality. It also allows for separate, specialized teams to implement and maintain security, networking, or authentication features that have been moved to the ambassador.

Ambassador services can be deployed as a [sidecar](#) to accompany the lifecycle of a consuming application or service. Alternatively, if an ambassador is shared by multiple separate processes on a common host, it can be deployed as a daemon or Windows service. If the consuming service is containerized, the ambassador should be created as a separate container on the same host, with the appropriate links configured for communication.

Issues and considerations

- The proxy adds some latency overhead. Consider whether a client library, invoked directly by the application, is a better approach.
- Consider the possible impact of including generalized features in the proxy. For example, the ambassador could handle retries, but that might not be safe unless all operations are idempotent.
- Consider a mechanism to allow the client to pass some context to the proxy, as well as back to the client. For example, include HTTP request headers to opt out of retry or specify the maximum number of times to retry.
- Consider how you will package and deploy the proxy.
- Consider whether to use a single shared instance for all clients or an instance for each client.

When to use this pattern

Use this pattern when you:

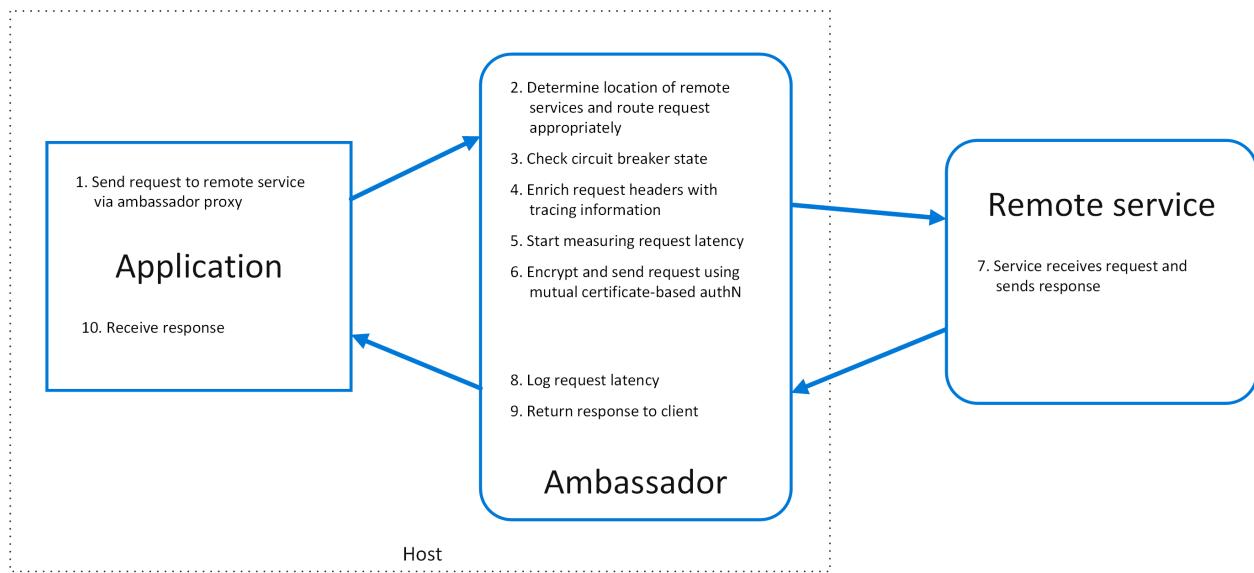
- Need to build a common set of client connectivity features for multiple languages or frameworks.
- Need to offload cross-cutting client connectivity concerns to infrastructure developers or other more specialized teams.
- Need to support cloud or cluster connectivity requirements in a legacy application or an application that is difficult to modify.

This pattern may not be suitable:

- When network request latency is critical. A proxy will introduce some overhead, although minimal, and in some cases this may affect the application.
- When client connectivity features are consumed by a single language. In that case, a better option might be a client library that is distributed to the development teams as a package.
- When connectivity features cannot be generalized and require deeper integration with the client application.

Example

The following diagram shows an application making a request to a remote service via an ambassador proxy. The ambassador provides routing, circuit breaking, and logging. It calls the remote service and then returns the response to the client application:



Related resources

- [Sidecar pattern](#)

Anti-corruption Layer pattern

Azure

Azure Logic Apps

Implement a façade or adapter layer between different subsystems that don't share the same semantics. This layer translates requests that one subsystem makes to the other subsystem. Use this pattern to ensure that an application's design is not limited by dependencies on outside subsystems. This pattern was first described by Eric Evans in *Domain-Driven Design*.

Context and problem

Most applications rely on other systems for some data or functionality. For example, when a legacy application is migrated to a modern system, it may still need existing legacy resources. New features must be able to call the legacy system. This is especially true of gradual migrations, where different features of a larger application are moved to a modern system over time.

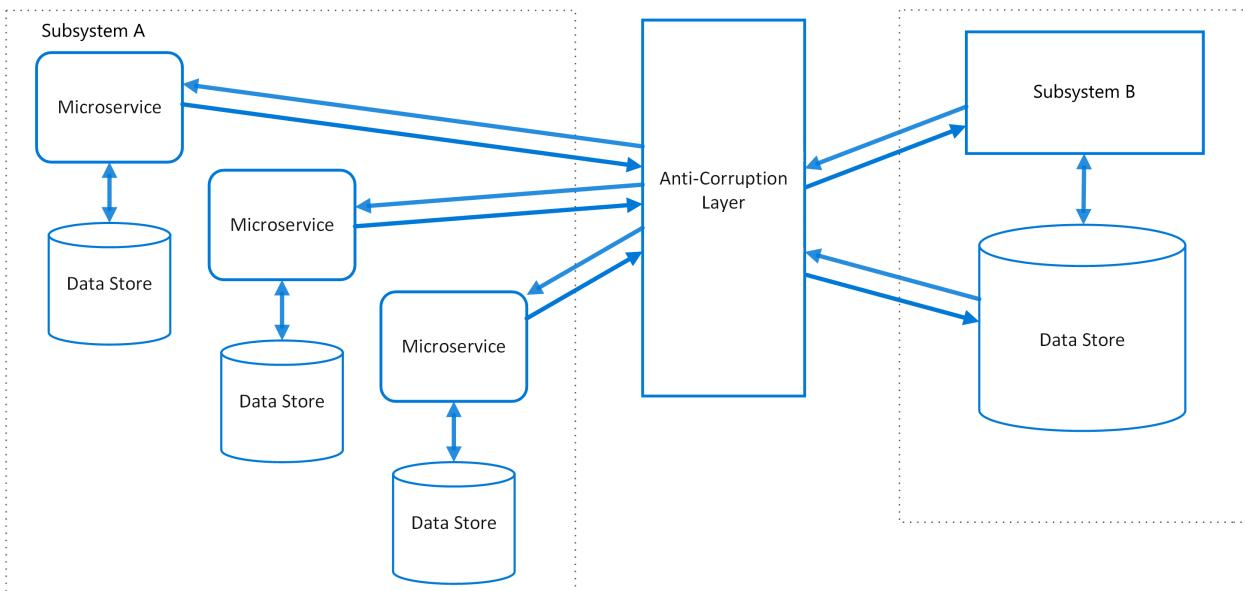
Often these legacy systems suffer from quality issues such as convoluted data schemas or obsolete APIs. The features and technologies used in legacy systems can vary widely from more modern systems. To interoperate with the legacy system, the new application may need to support outdated infrastructure, protocols, data models, APIs, or other features that you wouldn't otherwise put into a modern application.

Maintaining access between new and legacy systems can force the new system to adhere to at least some of the legacy system's APIs or other semantics. When these legacy features have quality issues, supporting them "corrupts" what might otherwise be a cleanly designed modern application.

Similar issues can arise with any external system that your development team doesn't control, not just legacy systems.

Solution

Isolate the different subsystems by placing an anti-corruption layer between them. This layer translates communications between the two systems, allowing one system to remain unchanged while the other can avoid compromising its design and technological approach.



The diagram above shows an application with two subsystems. Subsystem A calls to subsystem B through an anti-corruption layer. Communication between subsystem A and the anti-corruption layer always uses the data model and architecture of subsystem A. Calls from the anti-corruption layer to subsystem B conform to that subsystem's data model or methods. The anti-corruption layer contains all of the logic necessary to translate between the two systems. The layer can be implemented as a component within the application or as an independent service.

Issues and considerations

- The anti-corruption layer may add latency to calls made between the two systems.
- The anti-corruption layer adds an additional service that must be managed and maintained.
- Consider how your anti-corruption layer will scale.
- Consider whether you need more than one anti-corruption layer. You may want to decompose functionality into multiple services using different technologies or languages, or there may be other reasons to partition the anti-corruption layer.
- Consider how the anti-corruption layer will be managed in relation with your other applications or services. How will it be integrated into your monitoring, release, and configuration processes?
- Make sure transaction and data consistency are maintained and can be monitored.
- Consider whether the anti-corruption layer needs to handle all communication between different subsystems, or just a subset of features.
- If the anti-corruption layer is part of an application migration strategy, consider whether it will be permanent, or will be retired after all legacy functionality has been migrated.
- This pattern is illustrated with distinct subsystems above, but can apply to other service architectures as well, such as when integrating legacy code together in a

monolithic architecture.

When to use this pattern

Use this pattern when:

- A migration is planned to happen over multiple stages, but integration between new and legacy systems needs to be maintained.
- Two or more subsystems have different semantics, but still need to communicate.

This pattern may not be suitable if there are no significant semantic differences between new and legacy systems.

Related resources

- [Strangler Fig pattern](#)
- [Messaging Bridge pattern](#)

Asynchronous Request-Reply pattern

Azure

Azure Logic Apps

Decouple backend processing from a frontend host, where backend processing needs to be asynchronous, but the frontend still needs a clear response.

Context and problem

In modern application development, it's normal for client applications — often code running in a web-client (browser) — to depend on remote APIs to provide business logic and compose functionality. These APIs may be directly related to the application or may be shared services provided by a third party. Commonly these API calls take place over the HTTP(S) protocol and follow REST semantics.

In most cases, APIs for a client application are designed to respond quickly, on the order of 100 ms or less. Many factors can affect the response latency, including:

- An application's hosting stack.
- Security components.
- The relative geographic location of the caller and the backend.
- Network infrastructure.
- Current load.
- The size of the request payload.
- Processing queue length.
- The time for the backend to process the request.

Any of these factors can add latency to the response. Some can be mitigated by scaling out the backend. Others, such as network infrastructure, are largely out of the control of the application developer. Most APIs can respond quickly enough for responses to arrive back over the same connection. Application code can make a synchronous API call in a non-blocking way, giving the appearance of asynchronous processing, which is recommended for I/O-bound operations.

In some scenarios, however, the work done by backend may be long-running, on the order of seconds, or might be a background process that is executed in minutes or even hours. In that case, it isn't feasible to wait for the work to complete before responding to the request. This situation is a potential problem for any synchronous request-reply pattern.

Some architectures solve this problem by using a message broker to separate the request and response stages. This separation is often achieved by use of the [Queue-Based Load Leveling pattern](#). This separation can allow the client process and the backend API to scale independently. But this separation also brings additional complexity when the client requires success notification, as this step needs to become asynchronous.

Many of the same considerations discussed for client applications also apply for server-to-server REST API calls in distributed systems — for example, in a microservices architecture.

Solution

One solution to this problem is to use HTTP polling. Polling is useful to client-side code, as it can be hard to provide call-back endpoints or use long running connections. Even when callbacks are possible, the extra libraries and services that are required can sometimes add too much extra complexity.

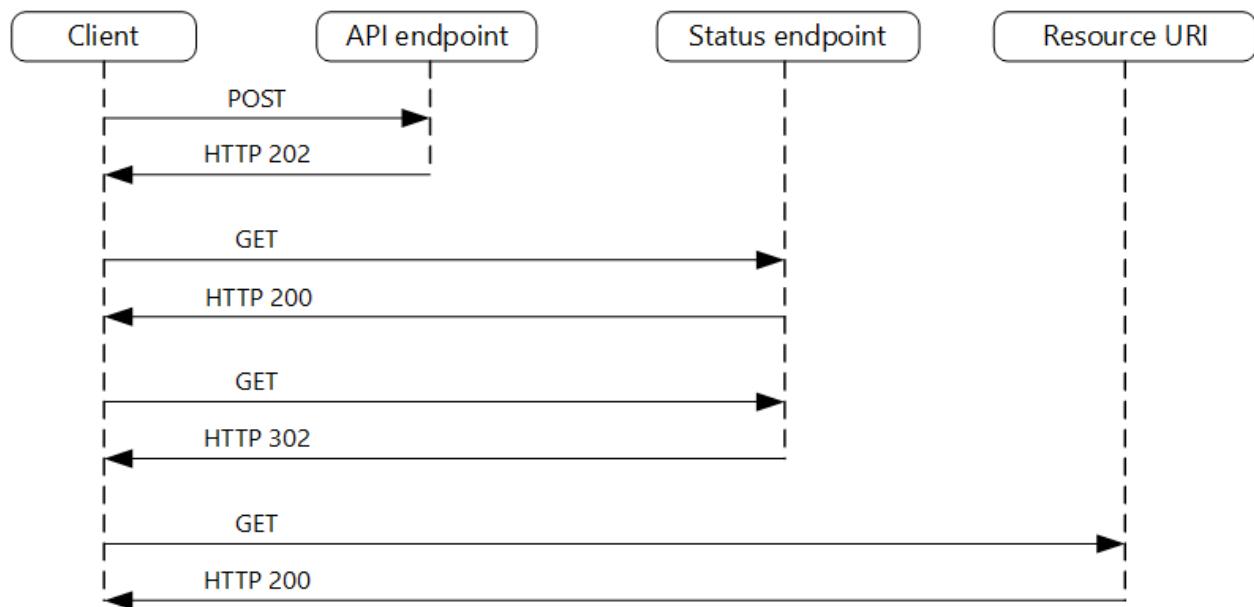
- The client application makes a synchronous call to the API, triggering a long-running operation on the backend.
- The API responds synchronously as quickly as possible. It returns an HTTP 202 (Accepted) status code, acknowledging that the request has been received for processing.

Note

The API should validate both the request and the action to be performed before starting the long running process. If the request is invalid, reply immediately with an error code such as HTTP 400 (Bad Request).

- The response holds a location reference pointing to an endpoint that the client can poll to check for the result of the long running operation.
- The API offloads processing to another component, such as a message queue.
- For every successful call to the status endpoint, it returns HTTP 200. While the work is still pending, the status endpoint returns a resource that indicates the work is still in progress. Once the work is complete, the status endpoint can either return a resource that indicates completion, or redirect to another resource URL. For example, if the asynchronous operation creates a new resource, the status endpoint would redirect to the URL for that resource.

The following diagram shows a typical flow:



1. The client sends a request and receives an HTTP 202 (Accepted) response.
2. The client sends an HTTP GET request to the status endpoint. The work is still pending, so this call returns HTTP 200.
3. At some point, the work is complete and the status endpoint returns 302 (Found) redirecting to the resource.
4. The client fetches the resource at the specified URL.

Issues and considerations

- There are a number of possible ways to implement this pattern over HTTP and not all upstream services have the same semantics. For example, most services won't return an HTTP 202 response back from a GET method when a remote process hasn't finished. Following pure REST semantics, they should return HTTP 404 (Not Found). This response makes sense when you consider the result of the call isn't present yet.
- An HTTP 202 response should indicate the location and frequency that the client should poll for the response. It should have the following additional headers:

expand table Expand table

Header	Description	Notes
Location	A URL the client should poll for a response status.	This URL could be a SAS token with the Valet Key Pattern being appropriate if this location needs access control. The valet key pattern is

Header	Description	Notes
Retry-After	An estimate of when processing will complete	This header is designed to prevent polling clients from overwhelming the back-end with retries. also valid when response polling needs offloading to another backend

- You may need to use a processing proxy or facade to manipulate the response headers or payload depending on the underlying services used.
- If the status endpoint redirects on completion, either [HTTP 302](#) or [HTTP 303](#) are appropriate return codes, depending on the exact semantics you support.
- Upon successful processing, the resource specified by the Location header should return an appropriate HTTP response code such as 200 (OK), 201 (Created), or 204 (No Content).
- If an error occurs during processing, persist the error at the resource URL described in the Location header and ideally return an appropriate response code to the client from that resource (4xx code).
- Not all solutions will implement this pattern in the same way and some services will include additional or alternate headers. For example, Azure Resource Manager uses a modified variant of this pattern. For more information, see [Azure Resource Manager Async Operations](#).
- Legacy clients might not support this pattern. In that case, you might need to place a facade over the asynchronous API to hide the asynchronous processing from the original client. For example, Azure Logic Apps supports this pattern natively can be used as an integration layer between an asynchronous API and a client that makes synchronous calls. See [Perform long-running tasks with the webhook action pattern](#).
- In some scenarios, you might want to provide a way for clients to cancel a long-running request. In that case, the backend service must support some form of cancellation instruction.

When to use this pattern

Use this pattern for:

- Client-side code, such as browser applications, where it's difficult to provide call-back endpoints, or the use of long-running connections adds too much additional

complexity.

- Service calls where only the HTTP protocol is available and the return service can't fire callbacks because of firewall restrictions on the client-side.
- Service calls that need to be integrated with legacy architectures that don't support modern callback technologies such as WebSockets or webhooks.

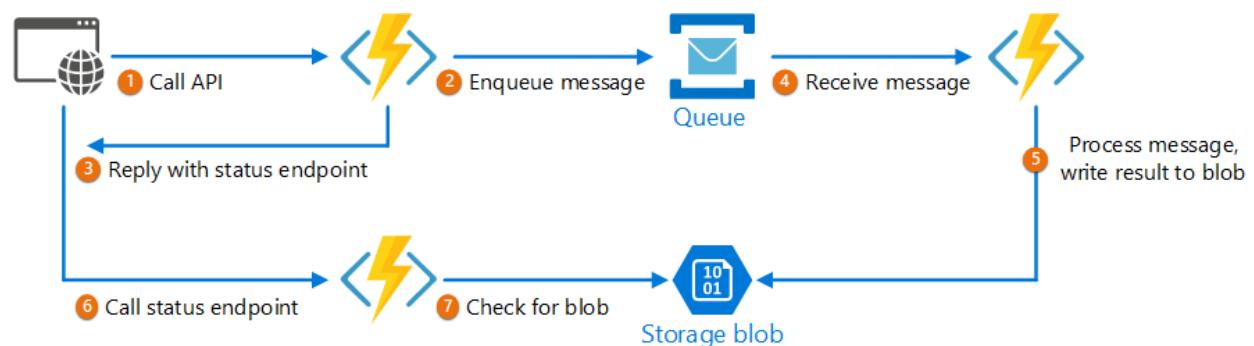
This pattern might not be suitable when:

- You can use a service built for asynchronous notifications instead, such as Azure Event Grid.
- Responses must stream in real time to the client.
- The client needs to collect many results, and received latency of those results is important. Consider a service bus pattern instead.
- You can use server-side persistent network connections such as WebSockets or SignalR. These services can be used to notify the caller of the result.
- The network design allows you to open up ports to receive asynchronous callbacks or webhooks.

Example

The following code shows excerpts from an application that uses Azure Functions to implement this pattern. There are three functions in the solution:

- The asynchronous API endpoint.
- The status endpoint.
- A backend function that takes queued work items and executes them.



This sample is available on [GitHub](#).

AsyncProcessingWorkAcceptor function

The `AsyncProcessingWorkAcceptor` function implements an endpoint that accepts work from a client application and puts it on a queue for processing.

- The function generates a request ID and adds it as metadata to the queue message.
- The HTTP response includes a location header pointing to a status endpoint. The request ID is part of the URL path.

C#

```
public static class AsyncProcessingWorkAcceptor
{
    [FunctionName("AsyncProcessingWorkAcceptor")]
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route = null)]
        CustomerPOCO customer,
        [ServiceBus("outqueue", Connection =
        "ServiceBusConnectionAppSetting")] IAsyncCollector<ServiceBusMessage>
        OutMessages,
        ILogger log)
    {
        if (String.IsNullOrEmpty(customer.id) ||
        string.IsNullOrEmpty(customer.customername))
        {
            return new BadRequestResult();
        }

        string reqid = Guid.NewGuid().ToString();

        string rqs =
        $"http://{{Environment.GetEnvironmentVariable("WEBSITE_HOSTNAME")}}/api/Reques
        tStatus/{reqid}";

        var messagePayload = JsonConvert.SerializeObject(customer);
        var message = new ServiceBusMessage(messagePayload);
        message.ApplicationProperties.Add("RequestGUID", reqid);
        message.ApplicationProperties.Add("RequestSubmittedAt",
        DateTime.Now);
        message.ApplicationProperties.Add("RequestStatusURL", rqs);

        await OutMessages.AddAsync(message);

        return new AcceptedResult(rqs, $"Request Accepted for
        Processing{Environment.NewLine}ProxyStatus: {rqs}");
    }
}
```

AsyncProcessingBackgroundWorker function

The `AsyncProcessingBackgroundWorker` function picks up the operation from the queue, does some work based on the message payload, and writes the result to a storage account.

C#

```
public static class AsyncProcessingBackgroundWorker
{
    [FunctionName("AsyncProcessingBackgroundWorker")]
    public static async Task RunAsync(
        [ServiceBusTrigger("outqueue", Connection =
"ServiceBusConnectionAppSetting")] BinaryData customer,
        IDictionary<string, object> applicationProperties,
        [Blob("data", FileAccess.ReadWrite, Connection =
"StorageConnectionAppSetting")] BlobContainerClient inputContainer,
        ILogger log)
    {
        // Perform an actual action against the blob data source for the
        // async readers to be able to check against.
        // This is where your actual service worker processing will be
        // performed

        var id = applicationProperties["RequestGUID"] as string;

        BlobClient blob = inputContainer.GetBlobClient($"{id}.blobdata");

        // Now write the results to blob storage.
        await blob.UploadAsync(customer);
    }
}
```

AsyncOperationStatusChecker function

The `AsyncOperationStatusChecker` function implements the status endpoint. This function first checks whether the request was completed

- If the request was completed, the function either returns a valet-key to the response, or redirects the call immediately to the valet-key URL.
- If the request is still pending, then we should return a **200 code, including the current state**.

C#

```
public static class AsyncOperationStatusChecker
{
    [FunctionName("AsyncOperationStatusChecker")]
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Anonymous, "get", Route =
"RequestStatus/{thisGUID}")] HttpRequest req,
```

```

    [Blob("data/{thisGuid}.blobdata", FileAccess.Read, Connection =
"StorageConnectionAppSetting")] BlockBlobClient inputBlob, string thisGUID,
    ILogger log)
{

    OnCompleteEnum OnComplete = Enum.Parse<OnCompleteEnum>
(req.Query["OnComplete"].FirstOrDefault() ?? "Redirect");
    OnPendingEnum OnPending = Enum.Parse<OnPendingEnum>
(req.Query["OnPending"].FirstOrDefault() ?? "OK");

    log.LogInformation($"C# HTTP trigger function processed a request
for status on {thisGUID} - OnComplete {OnComplete} - OnPending
{OnPending}");

    // Check to see if the blob is present
    if (await inputBlob.ExistsAsync())
    {
        // If it's present, depending on the value of the optional
        "OnComplete" parameter choose what to do.
        return await OnCompleted(OnComplete, inputBlob, thisGUID);
    }
    else
    {
        // If it's NOT present, then we need to back off. Depending on
        the value of the optional "OnPending" parameter, choose what to do.
        string rqs =
$http://{Environment.GetEnvironmentVariable("WEBSITE_HOSTNAME")}/api/Reques
tStatus/{thisGUID}";

        switch (OnPending)
        {
            case OnPendingEnum.OK:
            {
                // Return an HTTP 200 status code.
                return new OkObjectResult(new { status = "In
progress", Location = rqs });
            }

            case OnPendingEnum.Synchronous:
            {
                // Back off and retry. Time out if the backoff
                period hits one minute.
                int backoff = 250;

                while (!await inputBlob.ExistsAsync() && backoff <
64000)
                {
                    log.LogInformation($"Synchronous mode
{thisGUID}.blob - retrying in {backoff} ms");
                    backoff = backoff * 2;
                    await Task.Delay(backoff);
                }

                if (await inputBlob.ExistsAsync())
                {

```

```

                log.LogInformation($"Synchronous Redirect mode
{thisGUID}.blob - completed after {backoff} ms");
                return await OnCompleted(OnComplete, inputBlob,
thisGUID);
            }
            else
            {
                log.LogInformation($"Synchronous mode
{thisGUID}.blob - NOT FOUND after timeout {backoff} ms");
                return new NotFoundResult();
            }
        }

        default:
        {
            throw new InvalidOperationException($"Unexpected
value: {OnPending}");
        }
    }
}

private static async Task<IActionResult> OnCompleted(OnCompleteEnum
OnComplete, BlockBlobClient inputBlob, string thisGUID)
{
    switch (OnComplete)
    {
        case OnCompleteEnum.Redirect:
        {
            // Redirect to the SAS URI to blob storage

            return new RedirectResult(inputBlob.GenerateSASURI());
        }

        case OnCompleteEnum.Stream:
        {
            // Download the file and return it directly to the
caller.

            // For larger files, use a stream to minimize RAM usage.
            return new OkObjectResult(await
inputBlob.DownloadContentAsync());
        }

        default:
        {
            throw new InvalidOperationException($"Unexpected value:
{OnComplete}");
        }
    }
}

public enum OnCompleteEnum
{

```

```
    Redirect,  
    Stream  
}  
  
public enum OnPendingEnum  
{  
  
    OK,  
    Synchronous  
}
```

Next steps

The following information may be relevant when implementing this pattern:

- [Azure Logic Apps](#) - Perform long-running tasks with the polling action pattern.
- For general best practices when designing a web API, see [Web API design](#).

Related resources

- [Backends for Frontends pattern](#)

Backends for Frontends pattern

Azure

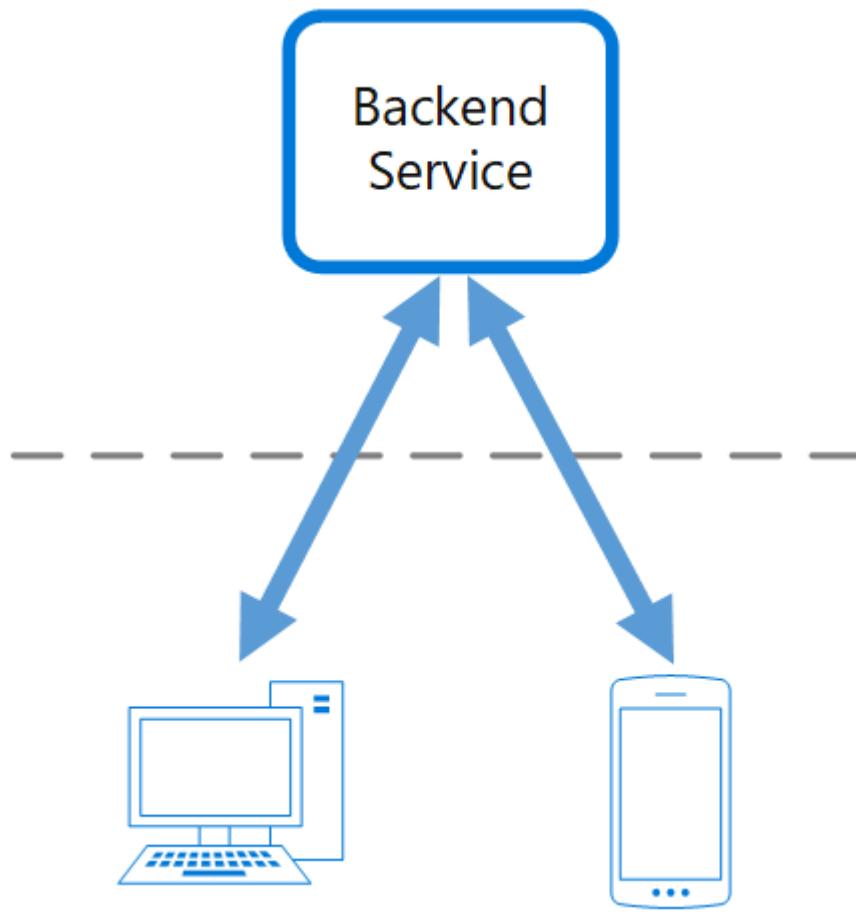
Create separate backend services to be consumed by specific frontend applications or interfaces. This pattern is useful when you want to avoid customizing a single backend for multiple interfaces. This pattern was first described by Sam Newman.

Context and problem

An application may initially be targeted at a desktop web UI. Typically, a backend service is developed in parallel that provides the features needed for that UI. As the application's user base grows, a mobile application is developed that must interact with the same backend. The backend service becomes a general-purpose backend, serving the requirements of both the desktop and mobile interfaces.

But the capabilities of a mobile device differ significantly from a desktop browser, in terms of screen size, performance, and display limitations. As a result, the requirements for a mobile application backend differ from the desktop web UI.

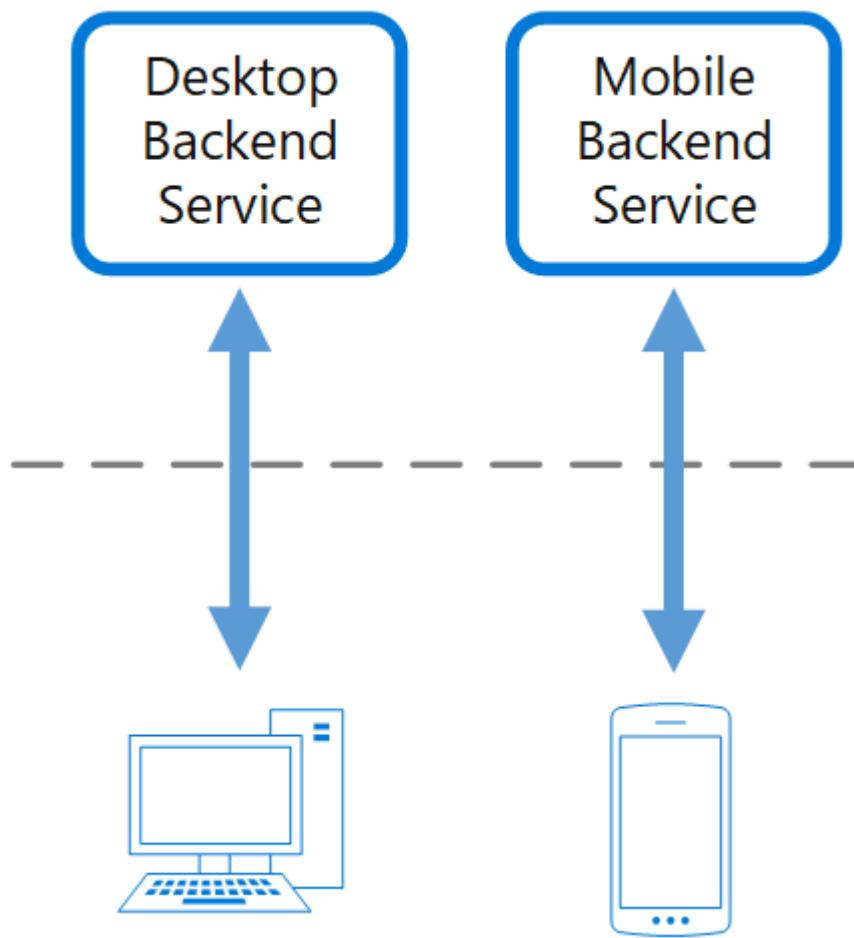
These differences result in competing requirements for the backend. The backend requires regular and significant changes to serve both the desktop web UI and the mobile application. Often, separate interface teams work on each frontend, causing the backend to become a bottleneck in the development process. Conflicting update requirements, and the need to keep the service working for both frontends, can result in spending a lot of effort on a single deployable resource.



As the development activity focuses on the backend service, a separate team may be created to manage and maintain the backend. Ultimately, this results in a disconnect between the interface and backend development teams, placing a burden on the backend team to balance the competing requirements of the different UI teams. When one interface team requires changes to the backend, those changes must be validated with other interface teams before they can be integrated into the backend.

Solution

Create one backend per user interface. Fine-tune the behavior and performance of each backend to best match the needs of the frontend environment, without worrying about affecting other frontend experiences.



Because each backend is specific to one interface, it can be optimized for that interface. As a result, it will be smaller, less complex, and likely faster than a generic backend that tries to satisfy the requirements for all interfaces. Each interface team has autonomy to control their own backend and doesn't rely on a centralized backend development team. This gives the interface team flexibility in language selection, release cadence, prioritization of workload, and feature integration in their backend.

For more information, see [Pattern: Backends For Frontends ↗](#).

Issues and considerations

- Consider how many backends to deploy.
- If different interfaces (such as mobile clients) will make the same requests, consider whether it is necessary to implement a backend for each interface, or if a single backend will suffice.
- Code duplication across services is highly likely when implementing this pattern.
- Frontend-focused backend services should only contain client-specific logic and behavior. General business logic and other global features should be managed elsewhere in your application.
- Think about how this pattern might be reflected in the responsibilities of a development team.

- Consider how long it will take to implement this pattern. Will the effort of building the new backends incur technical debt, while you continue to support the existing generic backend?

When to use this pattern

Use this pattern when:

- A shared or general purpose backend service must be maintained with significant development overhead.
- You want to optimize the backend for the requirements of specific client interfaces.
- Customizations are made to a general-purpose backend to accommodate multiple interfaces.
- A programming language is better suited for the backend of a specific user interface, but not all user interfaces.

This pattern may not be suitable:

- When interfaces make the same or similar requests to the backend.
- When only one interface is used to interact with the backend.

Next steps

- [Pattern: Backends For Frontends ↗](#)

Related resources

- [Gateway Aggregation pattern](#)
- [Gateway Offloading pattern](#)
- [Gateway Routing pattern](#)

Bulkhead pattern

Azure

The Bulkhead pattern is a type of application design that is tolerant of failure. In a bulkhead architecture, elements of an application are isolated into pools so that if one fails, the others will continue to function. It's named after the sectioned partitions (bulkheads) of a ship's hull. If the hull of a ship is compromised, only the damaged section fills with water, which prevents the ship from sinking.

Context and problem

A cloud-based application may include multiple services, with each service having one or more consumers. Excessive load or failure in a service will impact all consumers of the service.

Moreover, a consumer may send requests to multiple services simultaneously, using resources for each request. When the consumer sends a request to a service that is misconfigured or not responding, the resources used by the client's request may not be freed in a timely manner. As requests to the service continue, those resources may be exhausted. For example, the client's connection pool may be exhausted. At that point, requests by the consumer to other services are affected. Eventually the consumer can no longer send requests to other services, not just the original unresponsive service.

The same issue of resource exhaustion affects services with multiple consumers. A large number of requests originating from one client may exhaust available resources in the service. Other consumers are no longer able to consume the service, causing a cascading failure effect.

Solution

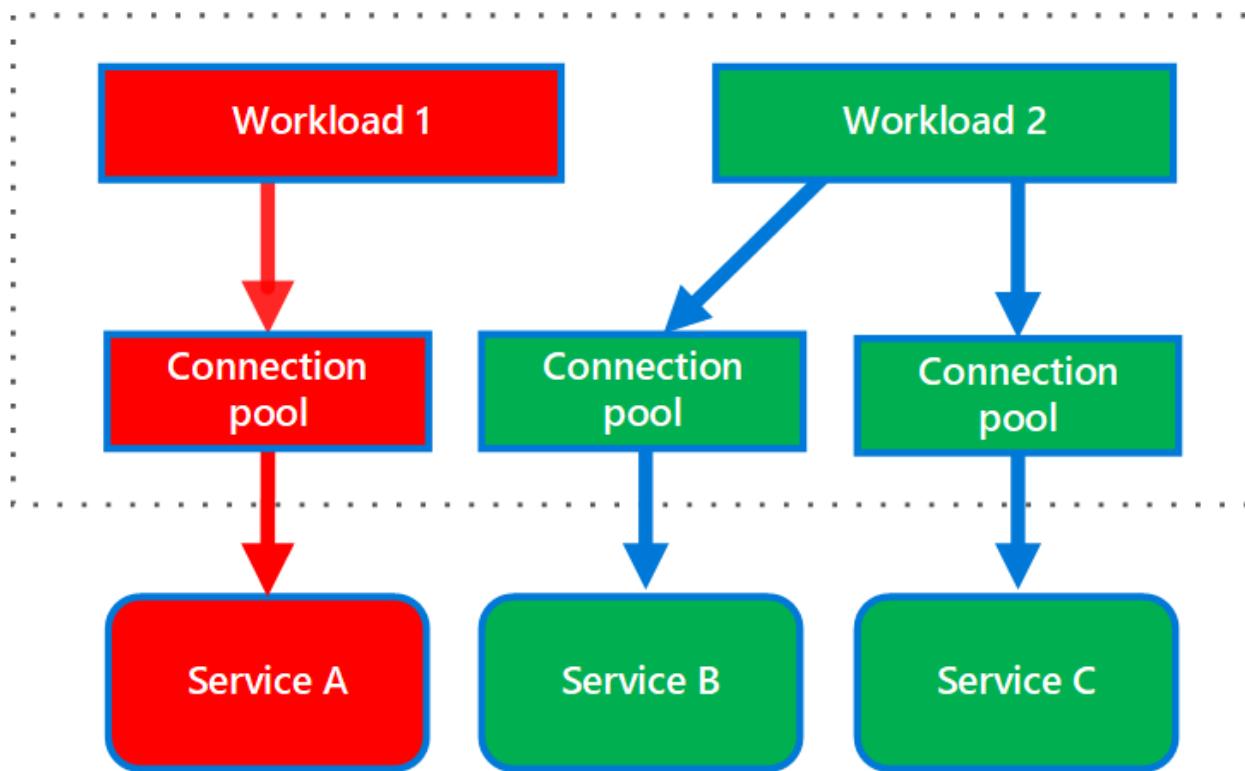
Partition service instances into different groups, based on consumer load and availability requirements. This design helps to isolate failures, and allows you to sustain service functionality for some consumers, even during a failure.

A consumer can also partition resources, to ensure that resources used to call one service don't affect the resources used to call another service. For example, a consumer that calls multiple services may be assigned a connection pool for each service. If a service begins to fail, it only affects the connection pool assigned for that service, allowing the consumer to continue using the other services.

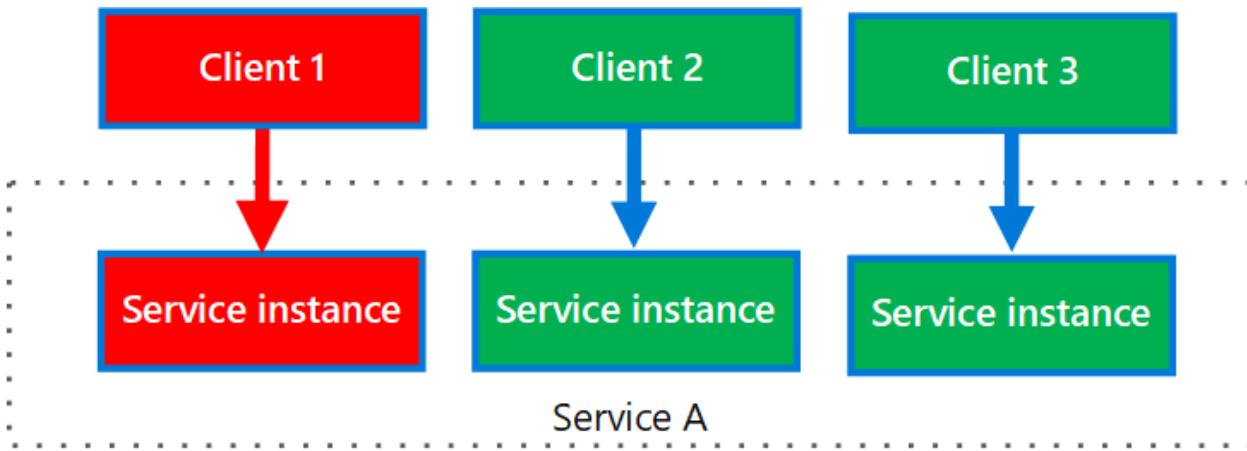
The benefits of this pattern include:

- Isolates consumers and services from cascading failures. An issue affecting a consumer or service can be isolated within its own bulkhead, preventing the entire solution from failing.
- Allows you to preserve some functionality in the event of a service failure. Other services and features of the application will continue to work.
- Allows you to deploy services that offer a different quality of service for consuming applications. A high-priority consumer pool can be configured to use high-priority services.

The following diagram shows bulkheads structured around connection pools that call individual services. If Service A fails or causes some other issue, the connection pool is isolated, so only workloads using the thread pool assigned to Service A are affected. Workloads that use Service B and C are not affected and can continue working without interruption.



The next diagram shows multiple clients calling a single service. Each client is assigned a separate service instance. Client 1 has made too many requests and overwhelmed its instance. Because each service instance is isolated from the others, the other clients can continue making calls.



Issues and considerations

- Define partitions around the business and technical requirements of the application.
- When partitioning services or consumers into bulkheads, consider the level of isolation offered by the technology as well as the overhead in terms of cost, performance and manageability.
- Consider combining bulkheads with retry, circuit breaker, and throttling patterns to provide more sophisticated fault handling.
- When partitioning consumers into bulkheads, consider using processes, thread pools, and semaphores. Projects like [resilience4j](#) and [Polly](#) offer a framework for creating consumer bulkheads.
- When partitioning services into bulkheads, consider deploying them into separate virtual machines, containers, or processes. Containers offer a good balance of resource isolation with fairly low overhead.
- Services that communicate using asynchronous messages can be isolated through different sets of queues. Each queue can have a dedicated set of instances processing messages on the queue, or a single group of instances using an algorithm to dequeue and dispatch processing.
- Determine the level of granularity for the bulkheads. For example, if you want to distribute tenants across partitions, you could place each tenant into a separate partition, or put several tenants into one partition.
- Monitor each partition's performance and SLA.

When to use this pattern

Use this pattern to:

- Isolate resources used to consume a set of backend services, especially if the application can provide some level of functionality even when one of the services

is not responding.

- Isolate critical consumers from standard consumers.
- Protect the application from cascading failures.

This pattern may not be suitable when:

- Less efficient use of resources may not be acceptable in the project.
- The added complexity is not necessary

Example

The following Kubernetes configuration file creates an isolated container to run a single service, with its own CPU and memory resources and limits.

```
yml

apiVersion: v1
kind: Pod
metadata:
  name: drone-management
spec:
  containers:
    - name: drone-management-container
      image: drone-service
      resources:
        requests:
          memory: "64Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
          cpu: "1"
```

Next steps

- [Designing reliable Azure applications](#)

Related resources

- [Circuit Breaker pattern](#)
- [Retry pattern](#)
- [Throttling pattern](#)

Cache-Aside pattern

Azure Cache for Redis

Load data on demand into a cache from a data store. This can improve performance and also helps to maintain consistency between data held in the cache and data in the underlying data store.

Context and problem

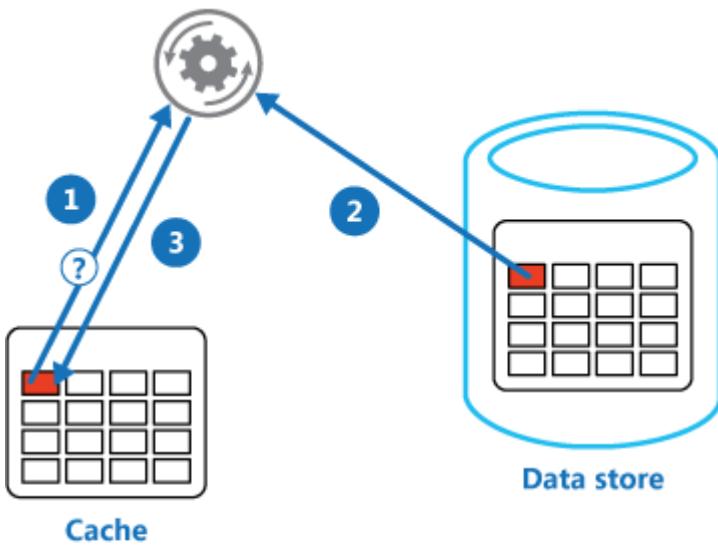
Applications use a cache to improve repeated access to information held in a data store. However, it's impractical to expect that cached data will always be completely consistent with the data in the data store. Applications should implement a strategy that helps to ensure that the data in the cache is as up-to-date as possible, but can also detect and handle situations that arise when the data in the cache has become stale.

Solution

Many commercial caching systems provide read-through and write-through/write-behind operations. In these systems, an application retrieves data by referencing the cache. If the data isn't in the cache, it's retrieved from the data store and added to the cache. Any modifications to data held in the cache are automatically written back to the data store as well.

For caches that don't provide this functionality, it's the responsibility of the applications that use the cache to maintain the data.

An application can emulate the functionality of read-through caching by implementing the cache-aside strategy. This strategy loads data into the cache on demand. The figure illustrates using the Cache-Aside pattern to store data in the cache.



- 1: Determine whether the item is currently held in the cache.
- 2: If the item is not currently in the cache, read the item from the data store.
- 3: Store a copy of the item in the cache.

If an application updates information, it can follow the write-through strategy by making the modification to the data store, and by invalidating the corresponding item in the cache.

When the item is next required, using the cache-aside strategy will cause the updated data to be retrieved from the data store and added back into the cache.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

Lifetime of cached data. Many caches implement an expiration policy that invalidates data and removes it from the cache if it's not accessed for a specified period. For cache-aside to be effective, ensure that the expiration policy matches the pattern of access for applications that use the data. Don't make the expiration period too short because this can cause applications to continually retrieve data from the data store and add it to the cache. Similarly, don't make the expiration period so long that the cached data is likely to become stale. Remember that caching is most effective for relatively static data, or data that is read frequently.

Evicting data. Most caches have a limited size compared to the data store where the data originates, and they'll evict data if necessary. Most caches adopt a least-recently-used policy for selecting items to evict, but this might be customizable. Configure the global expiration property and other properties of the cache, and the expiration

property of each cached item, to ensure that the cache is cost effective. It isn't always appropriate to apply a global eviction policy to every item in the cache. For example, if a cached item is very expensive to retrieve from the data store, it can be beneficial to keep this item in the cache at the expense of more frequently accessed but less costly items.

Priming the cache. Many solutions prepopulate the cache with the data that an application is likely to need as part of the startup processing. The Cache-Aside pattern can still be useful if some of this data expires or is evicted.

Consistency. Implementing the Cache-Aside pattern doesn't guarantee consistency between the data store and the cache. An item in the data store can be changed at any time by an external process, and this change might not be reflected in the cache until the next time the item is loaded. In a system that replicates data across data stores, this problem can become serious if synchronization occurs frequently.

Local (in-memory) caching. A cache could be local to an application instance and stored in-memory. Cache-aside can be useful in this environment if an application repeatedly accesses the same data. However, a local cache is private and so different application instances could each have a copy of the same cached data. This data could quickly become inconsistent between caches, so it might be necessary to expire data held in a private cache and refresh it more frequently. In these scenarios, consider investigating the use of a shared or a distributed caching mechanism.

When to use this pattern

Use this pattern when:

- A cache doesn't provide native read-through and write-through operations.
- Resource demand is unpredictable. This pattern enables applications to load data on demand. It makes no assumptions about which data an application will require in advance.

This pattern might not be suitable:

- When the cached data set is static. If the data will fit into the available cache space, prime the cache with the data on startup and apply a policy that prevents the data from expiring.
- For caching session state information in a web application hosted in a web farm. In this environment, you should avoid introducing dependencies based on client-server affinity.

Example

In Microsoft Azure you can use Azure Cache for Redis to create a distributed cache that can be shared by multiple instances of an application.

This following code example uses the [StackExchange.Redis](#) client, which is a Redis client library written for .NET. To connect to an Azure Cache for Redis instance, call the static `ConnectionMultiplexer.Connect` method and pass in the connection string. The method returns a `ConnectionMultiplexer` that represents the connection. One approach to sharing a `ConnectionMultiplexer` instance in your application is to have a static property that returns a connected instance, similar to the following example. This approach provides a thread-safe way to initialize only a single connected instance.

C#

```
private static ConnectionMultiplexer Connection;

// Redis connection string information
private static Lazy<ConnectionMultiplexer> lazyConnection = new
Lazy<ConnectionMultiplexer>(() =>
{
    string cacheConnection =
ConfigurationManager.AppSettings["CacheConnection"].ToString();
    return ConnectionMultiplexer.Connect(cacheConnection);
});

public static ConnectionMultiplexer Connection => lazyConnection.Value;
```

The `GetMyEntityAsync` method in the following code example shows an implementation of the Cache-Aside pattern. This method retrieves an object from the cache using the read-through approach.

An object is identified by using an integer ID as the key. The `GetMyEntityAsync` method tries to retrieve an item with this key from the cache. If a matching item is found, it's returned. If there's no match in the cache, the `GetMyEntityAsync` method retrieves the object from a data store, adds it to the cache, and then returns it. The code that actually reads the data from the data store is not shown here, because it depends on the data store. Note that the cached item is configured to expire to prevent it from becoming stale if it's updated elsewhere.

C#

```
// Set five minute expiration as a default
private const double DefaultExpirationTimeInMinutes = 5.0;

public async Task<MyEntity> GetMyEntityAsync(int id)
{
    // Define a unique key for this method and its parameters.
```

```

var key = $"MyEntity:{id}";
var cache = Connection.GetDatabase();

// Try to get the entity from the cache.
var json = await cache.StringGetAsync(key).ConfigureAwait(false);
var value = string.IsNullOrWhiteSpace(json)
    ? default(MyEntity)
    : JsonConvert.DeserializeObject<MyEntity>(json);

if (value == null) // Cache miss
{
    // If there's a cache miss, get the entity from the original store and
    // cache it.
    // Code has been omitted because it is data store dependent.
    value = ...;

    // Avoid caching a null value.
    if (value != null)
    {
        // Put the item in the cache with a custom expiration time that
        // depends on how critical it is to have stale data.
        await cache.StringSetAsync(key,
JsonConvert.SerializeObject(value)).ConfigureAwait(false);
        await cache.KeyExpireAsync(key,
TimeSpan.FromMinutes(DefaultExpirationTimeInMinutes)).ConfigureAwait(false);
    }
}

return value;
}

```

The examples use Azure Cache for Redis to access the store and retrieve information from the cache. For more information, see [Using Azure Cache for Redis](#) and [How to create a Web App with Azure Cache for Redis](#).

The `UpdateEntityAsync` method shown below demonstrates how to invalidate an object in the cache when the value is changed by the application. The code updates the original data store and then removes the cached item from the cache.

C#

```

public async Task UpdateEntityAsync(MyEntity entity)
{
    // Update the object in the original data store.
    await this.store.UpdateEntityAsync(entity).ConfigureAwait(false);

    // Invalidate the current cache object.
    var cache = Connection.GetDatabase();
    var id = entity.Id;
    var key = $"MyEntity:{id}"; // The key for the cached object.

```

```
    await cache.KeyDeleteAsync(key).ConfigureAwait(false); // Delete this
    key from the cache.
}
```

ⓘ Note

The order of the steps is important. Update the data store *before* removing the item from the cache. If you remove the cached item first, there is a small window of time when a client might fetch the item before the data store is updated. That will result in a cache miss (because the item was removed from the cache), causing the earlier version of the item to be fetched from the data store and added back into the cache. The result will be stale cache data.

Related resources

The following information might be relevant when implementing this pattern:

- [Reliable web app pattern](#) shows you how to apply the cache-aside pattern to web applications converging on the cloud.
- [Caching Guidance](#). Provides additional information on how you can cache data in a cloud solution, and the issues that you should consider when you implement a cache.
- [Data Consistency Primer](#). Cloud applications typically use data that's spread across data stores. Managing and maintaining data consistency in this environment is a critical aspect of the system, particularly the concurrency and availability issues that can arise. This primer describes issues about consistency across distributed data, and summarizes how an application can implement eventual consistency to maintain the availability of data.

Choreography pattern

Azure Event Grid

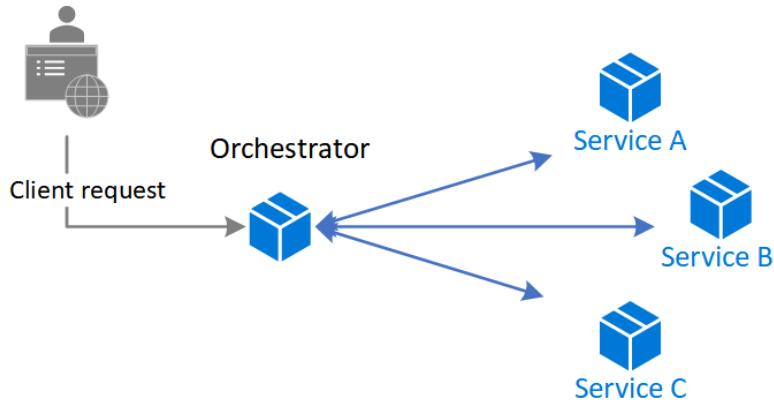
Have each component of the system participate in the decision-making process about the workflow of a business transaction, instead of relying on a central point of control.

Context and problem

In microservices architecture, it's often the case that a cloud-based application is divided into several small services that work together to process a business transaction end-to-end. To lower coupling between services, each service is responsible for a single business operation. Some benefits include faster development, smaller code base, and scalability. However, designing an efficient and scalable workflow is a challenge and often requires complex interservice communication.

The services communicate with each other by using well-defined APIs. Even a single business operation can result in multiple point-to-point calls among all services. A common pattern for communication is to use a centralized service that acts as the orchestrator. It acknowledges all incoming requests and delegates operations to the respective services. In doing so, it also manages the workflow of the entire business transaction. Each service just completes an operation and is not aware of the overall workflow.

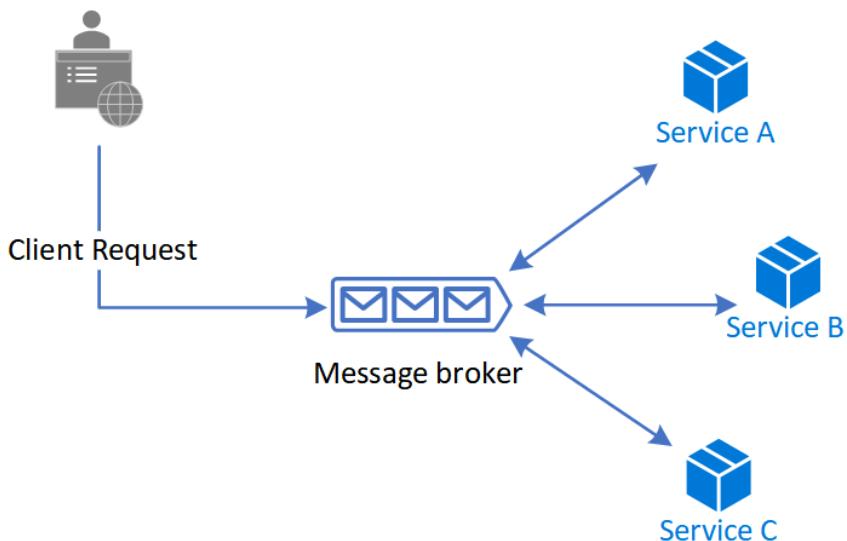
The orchestrator pattern reduces point-to-point communication between services but has some drawbacks because of the tight coupling between the orchestrator and other services that participate in processing of the business transaction. To execute tasks in a sequence, the orchestrator needs to have some domain knowledge about the responsibilities of those services. If you want to add or remove services, existing logic will break, and you'll need to rewire portions of the communication path. While you can configure the workflow, add or remove services easily with a well-designed orchestrator, such an implementation is complex and hard to maintain.



Solution

Let each service decide when and how a business operation is processed, instead of depending on a central orchestrator.

One way to implement choreography is to use the [asynchronous messaging pattern](#) to coordinate the business operations.



A client request publishes messages to a message queue. As messages arrive, they are pushed to subscribers, or services, interested in that message. Each subscribed service does their operation as indicated by the message and responds to the message queue with success or failure of the operation. In case of success, the service can push a message back to the same queue or a different message queue so that another service can continue the workflow if needed. If an operation fails, the message bus can retry that operation.

This way, the services choreograph the workflow among themselves without depending on an orchestrator or having direct communication between them.

Because there isn't point-to-point communication, this pattern helps reduce coupling between services. Also, it can remove the performance bottleneck caused by the orchestrator when it has to deal with all transactions.

When to use this pattern

Use the choreography pattern if you expect to update or replace services frequently, and add or remove some services eventually. The entire app can be modified with less effort and minimal disruption to existing services.

Consider this pattern if you experience performance bottlenecks in the central orchestrator.

This pattern is a natural model for the serverless architecture where all services can be short lived, or event driven. Services can spin up because of an event, do their task, and are removed when the task is finished.

Issues and considerations

Decentralizing the orchestrator can cause issues while managing the workflow.

If a service fails to complete a business operation, it can be difficult to recover from that failure. One way is to have the service indicate failure by firing an event. Another service subscribes to those failed events takes necessary actions such as applying [compensating transactions](#) to undo successful operations in a request. The failed service might also fail to fire an event for the failure. In that case, consider using a retry and, or time out mechanism to recognize that operation as a failure. For an example, see the [Example](#) section.

It's simple to implement a workflow when you want to process independent business operations in parallel. You can use a single message bus. However, the workflow can become complicated when choreography needs to occur in a sequence. For instance, Service C can start its operation only after Service A and Service B have completed their operations with success. One approach is to have multiple message buses that get messages in the required order. For more information, see the [Example](#) section.

The choreography pattern becomes a challenge if the number of services grow rapidly. Given the high number of independent moving parts, the workflow between services tends to get complex. Also, distributed tracing becomes difficult.

The orchestrator centrally manages the resiliency of the workflow and it can become a single point of failure. On the other hand, for choreography, the role is distributed

between all services and resiliency becomes less robust.

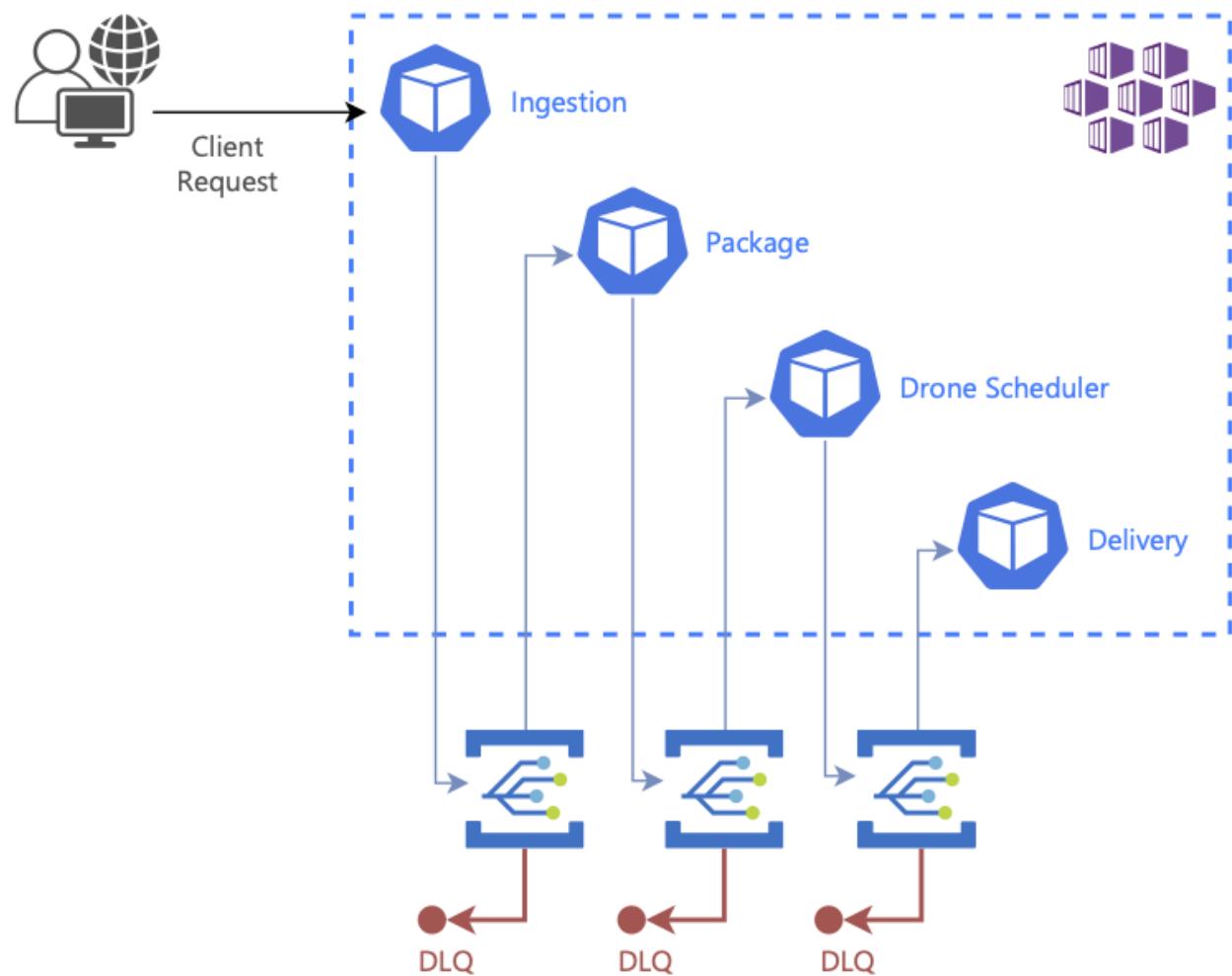
Each service isn't only responsible for the resiliency of its operation but also the workflow. This responsibility can be burdensome for the service and hard to implement. Each service must retry transient, nontransient, and time-out failures, so that the request terminates gracefully, if needed. Also, the service must be diligent about communicating the success or failure of the operation so that other services can act accordingly.

Example

This example shows the choreography pattern with the [Drone Delivery app](#). When a client requests a pickup, the app assigns a drone and notifies the client.



An example of this pattern is available on [GitHub](#).



A single client business transaction requires three distinct business operations: creating or updating a package, assigning a drone to deliver the package, and checking the delivery status. Those operations are performed by three microservices: Package, Drone Scheduler, and Delivery services. Instead of a central orchestrator, the services use messaging to collaborate and coordinate the request among themselves.

Design

The business transaction is processed in a sequence through multiple hops. Each hop has a message bus and the respective business service.

When a client sends a delivery request through an HTTP endpoint, the Ingestion service receives it, raises an operation event, and sends it to a message bus. The bus invokes the subscribed business service and sends the event in a POST request. On receiving the event, the business service can complete the operation with success, failure, or the request can time out. If successful, the service responds to the bus with the Ok status code, raises a new operation event, and sends it to the message bus of the next hop. In case of a failure or time-out, the service reports failure by sending the BadRequest code to the message bus that sent the original POST request. The message bus retries the operation based on a retry policy. After that period expires, message bus flags the failed operation and further processing of the entire request stops.

This workflow continues until the entire request has been processed.

The design uses multiple message buses to process the entire business transaction.

[Microsoft Azure Event Grid](#) provides the messaging service. The app is deployed in an [Azure Kubernetes Service \(AKS\)](#) cluster with [two containers in the same pod](#). One container runs the [ambassador](#) that interacts with Event Grid while the other runs a business service. The approach with two containers in the same pod improves performance and scalability. The ambassador and the business service share the same network allowing for low latency and high throughput.

To avoid cascading retry operations that might lead to multiple efforts, only Event Grid retries an operation instead of the business service. It flags a failed request by sending a messaging to a [dead letter queue \(DLQ\)](#).

The business services are idempotent to make sure retry operations don't result in duplicate resources. For example, the Package service uses upsert operations to add data to the data store.

The example implements a custom solution to correlate calls across all services and Event Grid hops.

Here's a code example that shows the choreography pattern between all business services. It shows the workflow of the Drone Delivery app transactions. Code for exception handling and logging have been removed for brevity.

C#

```

[HttpPost]
[Route("/api/[controller]/operation")]
[ProducesResponseType(typeof(void), 200)]
[ProducesResponseType(typeof(void), 400)]
[ProducesResponseType(typeof(void), 500)]

public async Task<IActionResult> Post([FromBody] EventGridEvent[] events)
{

    if (events == null)
    {
        return BadRequest("No Event for Choreography");
    }

    foreach(var e in events)
    {

        List<EventGridEvent> listEvents = new List<EventGridEvent>();
        e.Topic = eventRepository.GetTopic();
        e.EventTime = DateTime.Now;
        switch (e.EventType)
        {
            case Operations.ChoreographyOperation.ScheduleDelivery:
            {
                var packageGen = await
packageServiceCaller.UpsertPackageAsync(delivery.PackageInfo).ConfigureAwait
(false);
                if (packageGen is null)
                {
                    //BadRequest allows the event to be reprocessed by Event
Grid
                    return BadRequest("Package creation failed.");
                }

                //we set the event type to the next choreography step
                e.EventType =
Operations.ChoreographyOperation.CreatePackage;
                listEvents.Add(e);
                await eventRepository.SendEventAsync(listEvents);
                return Ok("Created Package Completed");
            }
            case Operations.ChoreographyOperation.CreatePackage:
            {
                var droneId = await
droneSchedulerServiceCaller.GetDroneIdAsync(delivery).ConfigureAwait(false);
                if (droneId is null)
                {
                    //BadRequest allows the event to be reprocessed by Event
Grid
                    return BadRequest("could not get a drone id");
                }
                e.Subject = droneId;
                e.EventType = Operations.ChoreographyOperation.GetDrone;
                listEvents.Add(e);
            }
        }
    }
}

```

```
        await eventRepository.SendEventAsync(listEvents);
        return Ok("Drone Completed");
    }
    case Operations.ChoreographyOperation.GetDrone:
    {
        var deliverySchedule = await
deliveryServiceCaller.ScheduleDeliveryAsync(delivery, e.Subject);
        return Ok("Delivery Completed");
    }
    return BadRequest();
}
}
```

Related resources

Consider these patterns in your design for choreography.

- Modularize the business service by using the [ambassador design pattern](#).
- Implement [queue-based load leveling pattern](#) to handle spikes of the workload.
- Use asynchronous distributed messaging through the [publisher-subscriber pattern](#).
- Use [compensating transactions](#) to undo a series of successful operations in case one or more related operation fails.
- For information about using a message broker in a messaging infrastructure, see [Asynchronous messaging options in Azure](#).

Circuit Breaker pattern

Azure

Handle faults that might take a variable amount of time to recover from, when connecting to a remote service or resource. This can improve the stability and resiliency of an application.

Context and problem

In a distributed environment, calls to remote resources and services can fail due to transient faults, such as slow network connections, timeouts, or the resources being overcommitted or temporarily unavailable. These faults typically correct themselves after a short period of time, and a robust cloud application should be prepared to handle them by using a strategy such as the [Retry pattern](#).

However, there can also be situations where faults are due to unanticipated events, and that might take much longer to fix. These faults can range in severity from a partial loss of connectivity to the complete failure of a service. In these situations it might be pointless for an application to continually retry an operation that is unlikely to succeed, and instead the application should quickly accept that the operation has failed and handle this failure accordingly.

Additionally, if a service is very busy, failure in one part of the system might lead to cascading failures. For example, an operation that invokes a service could be configured to implement a timeout, and reply with a failure message if the service fails to respond within this period. However, this strategy could cause many concurrent requests to the same operation to be blocked until the timeout period expires. These blocked requests might hold critical system resources such as memory, threads, database connections, and so on. Consequently, these resources could become exhausted, causing failure of other possibly unrelated parts of the system that need to use the same resources. In these situations, it would be preferable for the operation to fail immediately, and only attempt to invoke the service if it's likely to succeed. Note that setting a shorter timeout might help to resolve this problem, but the timeout shouldn't be so short that the operation fails most of the time, even if the request to the service would eventually succeed.

Solution

The Circuit Breaker pattern, popularized by Michael Nygard in his book, [Release It!](#) , can prevent an application from repeatedly trying to execute an operation that's likely to fail. Allowing it to continue without waiting for the fault to be fixed or wasting CPU cycles while it determines that the fault is long lasting. The Circuit Breaker pattern also enables an application to detect whether the fault has been resolved. If the problem appears to have been fixed, the application can try to invoke the operation.

The purpose of the Circuit Breaker pattern is different than the Retry pattern. The Retry pattern enables an application to retry an operation in the expectation that it'll succeed. The Circuit Breaker pattern prevents an application from performing an operation that is likely to fail. An application can combine these two patterns by using the Retry pattern to invoke an operation through a circuit breaker. However, the retry logic should be sensitive to any exceptions returned by the circuit breaker and abandon retry attempts if the circuit breaker indicates that a fault is not transient.

A circuit breaker acts as a proxy for operations that might fail. The proxy should monitor the number of recent failures that have occurred, and use this information to decide whether to allow the operation to proceed, or simply return an exception immediately.

The proxy can be implemented as a state machine with the following states that mimic the functionality of an electrical circuit breaker:

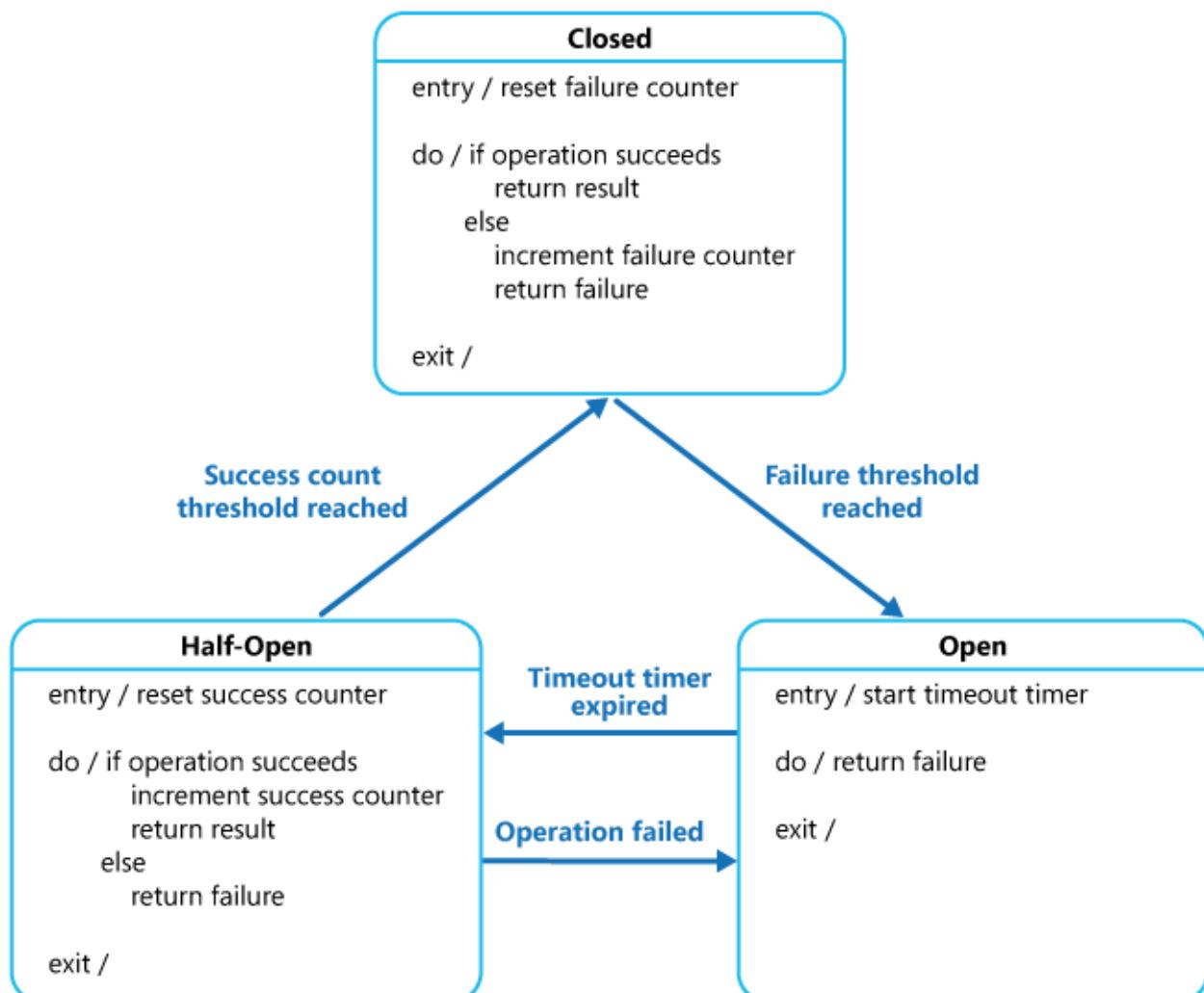
- **Closed:** The request from the application is routed to the operation. The proxy maintains a count of the number of recent failures, and if the call to the operation is unsuccessful the proxy increments this count. If the number of recent failures exceeds a specified threshold within a given time period, the proxy is placed into the **Open** state. At this point the proxy starts a timeout timer, and when this timer expires the proxy is placed into the **Half-Open** state.

The purpose of the timeout timer is to give the system time to fix the problem that caused the failure before allowing the application to try to perform the operation again.

- **Open:** The request from the application fails immediately and an exception is returned to the application.
- **Half-Open:** A limited number of requests from the application are allowed to pass through and invoke the operation. If these requests are successful, it's assumed that the fault that was previously causing the failure has been fixed and the circuit breaker switches to the **Closed** state (the failure counter is reset). If any request fails, the circuit breaker assumes that the fault is still present so it reverts to the

Open state and restarts the timeout timer to give the system a further period of time to recover from the failure.

The **Half-Open** state is useful to prevent a recovering service from suddenly being flooded with requests. As a service recovers, it might be able to support a limited volume of requests until the recovery is complete, but while recovery is in progress a flood of work can cause the service to time out or fail again.



In the figure, the failure counter used by the **Closed** state is time based. It's automatically reset at periodic intervals. This helps to prevent the circuit breaker from entering the **Open** state if it experiences occasional failures. The failure threshold that trips the circuit breaker into the **Open** state is only reached when a specified number of failures have occurred during a specified interval. The counter used by the **Half-Open** state records the number of successful attempts to invoke the operation. The circuit breaker reverts to the **Closed** state after a specified number of consecutive operation invocations have been successful. If any invocation fails, the circuit breaker enters the **Open** state immediately and the success counter will be reset the next time it enters the **Half-Open** state.

How the system recovers is handled externally, possibly by restoring or restarting a failed component or repairing a network connection.

The Circuit Breaker pattern provides stability while the system recovers from a failure and minimizes the impact on performance. It can help to maintain the response time of the system by quickly rejecting a request for an operation that's likely to fail, rather than waiting for the operation to time out, or never return. If the circuit breaker raises an event each time it changes state, this information can be used to monitor the health of the part of the system protected by the circuit breaker, or to alert an administrator when a circuit breaker trips to the **Open** state.

The pattern is customizable and can be adapted according to the type of the possible failure. For example, you can apply an increasing timeout timer to a circuit breaker. You could place the circuit breaker in the **Open** state for a few seconds initially, and then if the failure hasn't been resolved increase the timeout to a few minutes, and so on. In some cases, rather than the **Open** state returning failure and raising an exception, it could be useful to return a default value that is meaningful to the application.

Issues and considerations

You should consider the following points when deciding how to implement this pattern:

Exception Handling. An application invoking an operation through a circuit breaker must be prepared to handle the exceptions raised if the operation is unavailable. The way exceptions are handled will be application specific. For example, an application could temporarily degrade its functionality, invoke an alternative operation to try to perform the same task or obtain the same data, or report the exception to the user and ask them to try again later.

Types of Exceptions. A request might fail for many reasons, some of which might indicate a more severe type of failure than others. For example, a request might fail because a remote service has crashed and will take several minutes to recover, or because of a timeout due to the service being temporarily overloaded. A circuit breaker might be able to examine the types of exceptions that occur and adjust its strategy depending on the nature of these exceptions. For example, it might require a larger number of timeout exceptions to trip the circuit breaker to the **Open** state compared to the number of failures due to the service being completely unavailable.

Logging. A circuit breaker should log all failed requests (and possibly successful requests) to enable an administrator to monitor the health of the operation.

Recoverability. You should configure the circuit breaker to match the likely recovery pattern of the operation it's protecting. For example, if the circuit breaker remains in the **Open** state for a long period, it could raise exceptions even if the reason for the failure has been resolved. Similarly, a circuit breaker could fluctuate and reduce the response times of applications if it switches from the **Open** state to the **Half-Open** state too quickly.

Testing Failed Operations. In the **Open** state, rather than using a timer to determine when to switch to the **Half-Open** state, a circuit breaker can instead periodically ping the remote service or resource to determine whether it's become available again. This ping could take the form of an attempt to invoke an operation that had previously failed, or it could use a special operation provided by the remote service specifically for testing the health of the service, as described by the [Health Endpoint Monitoring pattern](#).

Manual Override. In a system where the recovery time for a failing operation is extremely variable, it's beneficial to provide a manual reset option that enables an administrator to close a circuit breaker (and reset the failure counter). Similarly, an administrator could force a circuit breaker into the **Open** state (and restart the timeout timer) if the operation protected by the circuit breaker is temporarily unavailable.

Concurrency. The same circuit breaker could be accessed by a large number of concurrent instances of an application. The implementation shouldn't block concurrent requests or add excessive overhead to each call to an operation.

Resource Differentiation. Be careful when using a single circuit breaker for one type of resource if there might be multiple underlying independent providers. For example, in a data store that contains multiple shards, one shard might be fully accessible while another is experiencing a temporary issue. If the error responses in these scenarios are merged, an application might try to access some shards even when failure is highly likely, while access to other shards might be blocked even though it's likely to succeed.

Accelerated Circuit Breaking. Sometimes a failure response can contain enough information for the circuit breaker to trip immediately and stay tripped for a minimum amount of time. For example, the error response from a shared resource that's overloaded could indicate that an immediate retry isn't recommended and that the application should instead try again in a few minutes.

Note

A service can return HTTP 429 (Too Many Requests) if it is throttling the client, or HTTP 503 (Service Unavailable) if the service is not currently available. The response can include additional information, such as the anticipated duration of the delay.

Replaying Failed Requests. In the **Open** state, rather than simply failing quickly, a circuit breaker could also record the details of each request to a journal and arrange for these requests to be replayed when the remote resource or service becomes available.

Inappropriate Timeouts on External Services. A circuit breaker might not be able to fully protect applications from operations that fail in external services that are configured with a lengthy timeout period. If the timeout is too long, a thread running a circuit breaker might be blocked for an extended period before the circuit breaker indicates that the operation has failed. In this time, many other application instances might also try to invoke the service through the circuit breaker and tie up a significant number of threads before they all fail.

When to use this pattern

Use this pattern:

- To prevent an application from trying to invoke a remote service or access a shared resource if this operation is highly likely to fail.

This pattern isn't recommended:

- For handling access to local private resources in an application, such as in-memory data structure. In this environment, using a circuit breaker would add overhead to your system.
- As a substitute for handling exceptions in the business logic of your applications.

Example

In a web application, several of the pages are populated with data retrieved from an external service. If the system implements minimal caching, most hits to these pages will cause a round trip to the service. Connections from the web application to the service could be configured with a timeout period (typically 60 seconds), and if the service doesn't respond in this time the logic in each web page will assume that the service is unavailable and throw an exception.

However, if the service fails and the system is very busy, users could be forced to wait for up to 60 seconds before an exception occurs. Eventually resources such as memory, connections, and threads could be exhausted, preventing other users from connecting to the system, even if they aren't accessing pages that retrieve data from the service.

Scaling the system by adding further web servers and implementing load balancing might delay when resources become exhausted, but it won't resolve the issue because

user requests will still be unresponsive and all web servers could still eventually run out of resources.

Wrapping the logic that connects to the service and retrieves the data in a circuit breaker could help to solve this problem and handle the service failure more elegantly. User requests will still fail, but they'll fail more quickly and the resources won't be blocked.

The `CircuitBreaker` class maintains state information about a circuit breaker in an object that implements the `ICircuitBreakerStateStore` interface shown in the following code.

C#

```
interface ICircuitBreakerStateStore
{
    CircuitBreakerStateEnum State { get; }

    Exception LastException { get; }

    DateTime LastStateChangedDateUtc { get; }

    void Trip(Exception ex);

    void Reset();

    void HalfOpen();

    bool IsClosed { get; }
}
```

The `State` property indicates the current state of the circuit breaker, and will be either `Open`, `HalfOpen`, or `Closed` as defined by the `CircuitBreakerStateEnum` enumeration. The `IsClosed` property should be true if the circuit breaker is closed, but false if it's open or half open. The `Trip` method switches the state of the circuit breaker to the open state and records the exception that caused the change in state, together with the date and time that the exception occurred. The `LastException` and the `LastStateChangedDateUtc` properties return this information. The `Reset` method closes the circuit breaker, and the `HalfOpen` method sets the circuit breaker to half open.

The `InMemoryCircuitBreakerStateStore` class in the example contains an implementation of the `ICircuitBreakerStateStore` interface. The `CircuitBreaker` class creates an instance of this class to hold the state of the circuit breaker.

The `ExecuteAction` method in the `CircuitBreaker` class wraps an operation, specified as an `Action` delegate. If the circuit breaker is closed, `ExecuteAction` invokes the `Action` delegate. If the operation fails, an exception handler calls `TrackException`, which sets the circuit breaker state to open. The following code example highlights this flow.

C#

```
public class CircuitBreaker
{
    private readonly ICircuitBreakerStateStore stateStore =
        CircuitBreakerStateStoreFactory.GetCircuitBreakerStateStore();

    private readonly object halfOpenSyncObject = new object ();
    ...

    public bool IsClosed { get { return stateStore.IsClosed; } }

    public bool IsOpen { get { return !IsClosed; } }

    public void ExecuteAction(Action action)
    {
        ...
        if (IsOpen)
        {
            // The circuit breaker is Open.
            ... (see code sample below for details)
        }

        // The circuit breaker is Closed, execute the action.
        try
        {
            action();
        }
        catch (Exception ex)
        {
            // If an exception still occurs here, simply
            // retrip the breaker immediately.
            this.TrackException(ex);

            // Throw the exception so that the caller can tell
            // the type of exception that was thrown.
            throw;
        }
    }

    private void TrackException(Exception ex)
    {
        // For simplicity in this example, open the circuit breaker on the first
        exception.
        // In reality this would be more complex. A certain type of exception,
        such as one
        // that indicates a service is offline, might trip the circuit breaker
        immediately.
        // Alternatively it might count exceptions locally or across multiple
```

```

instances and
    // use this value over time, or the exception/success ratio based on the
exception
    // types, to open the circuit breaker.
    this.stateStore.Trip(ex);
}
}

```

The following example shows the code (omitted from the previous example) that is executed if the circuit breaker isn't closed. It first checks if the circuit breaker has been open for a period longer than the time specified by the local `OpenToHalfOpenWaitTime` field in the `CircuitBreaker` class. If this is the case, the `ExecuteAction` method sets the circuit breaker to half open, then tries to perform the operation specified by the `Action` delegate.

If the operation is successful, the circuit breaker is reset to the closed state. If the operation fails, it is tripped back to the open state and the time the exception occurred is updated so that the circuit breaker will wait for a further period before trying to perform the operation again.

If the circuit breaker has only been open for a short time, less than the `OpenToHalfOpenWaitTime` value, the `ExecuteAction` method simply throws a `CircuitBreakerOpenException` exception and returns the error that caused the circuit breaker to transition to the open state.

Additionally, it uses a lock to prevent the circuit breaker from trying to perform concurrent calls to the operation while it's half open. A concurrent attempt to invoke the operation will be handled as if the circuit breaker was open, and it'll fail with an exception as described later.

C#

```

...
if (IsOpen)
{
    // The circuit breaker is Open. Check if the Open timeout has expired.
    // If it has, set the state to HalfOpen. Another approach might be to
    // check for the HalfOpen state that had be set by some other
    operation.
    if (stateStore.LastStateChangedDateUtc + OpenToHalfOpenWaitTime <
DateTime.UtcNow)
    {
        // The Open timeout has expired. Allow one operation to execute.
        Note that, in
        // this example, the circuit breaker is set to HalfOpen after being
        // in the Open state for some period of time. An alternative would
        be to set

```

```

        // this using some other approach such as a timer, test method,
manually, and
        // so on, and check the state here to determine how to handle
execution
        // of the action.
        // Limit the number of threads to be executed when the breaker is
HalfOpen.
        // An alternative would be to use a more complex approach to
determine which
        // threads or how many are allowed to execute, or to execute a
simple test
        // method instead.
        bool lockTaken = false;
        try
        {
            Monitor.TryEnter(halfOpenSyncObject, ref lockTaken);
            if (lockTaken)
            {
                // Set the circuit breaker state to HalfOpen.
                stateStore.HalfOpen();

                // Attempt the operation.
                action();

                // If this action succeeds, reset the state and allow other
operations.
                // In reality, instead of immediately returning to the Closed
state, a counter
                    // here would record the number of successful operations and
return the
                    // circuit breaker to the Closed state only after a specified
number succeed.
                this.stateStore.Reset();
                return;
            }
        }
        catch (Exception ex)
        {
            // If there's still an exception, trip the breaker again
immediately.
            this.stateStore.Trip(ex);

            // Throw the exception so that the caller knows which exception
occurred.
            throw;
        }
        finally
        {
            if (lockTaken)
            {
                Monitor.Exit(halfOpenSyncObject);
            }
        }
    }
    // The Open timeout hasn't yet expired. Throw a CircuitBreakerOpen

```

```
exception to
    // inform the caller that the call was not actually attempted,
    // and return the most recent exception received.
    throw new CircuitBreakerOpenException(stateStore.LastException);
}
...

```

To use a `CircuitBreaker` object to protect an operation, an application creates an instance of the `CircuitBreaker` class and invokes the `ExecuteAction` method, specifying the operation to be performed as the parameter. The application should be prepared to catch the `CircuitBreakerOpenException` exception if the operation fails because the circuit breaker is open. The following code shows an example:

C#

```
var breaker = new CircuitBreaker();

try
{
    breaker.ExecuteAction(() =>
    {
        // Operation protected by the circuit breaker.
        ...
    });
}
catch (CircuitBreakerOpenException ex)
{
    // Perform some different action when the breaker is open.
    // Last exception details are in the inner exception.
    ...
}
catch (Exception ex)
{
    ...
}
```

Related resources

The following patterns might also be useful when implementing this pattern:

- [Reliable web app pattern](#) shows you how to apply the circuit-breaker pattern to web applications converging on the cloud.
- [Retry pattern](#). Describes how an application can handle anticipated temporary failures when it tries to connect to a service or network resource by transparently retrying an operation that has previously failed.

- [Health Endpoint Monitoring pattern](#). A circuit breaker might be able to test the health of a service by sending a request to an endpoint exposed by the service. The service should return information indicating its status.

Claim-Check pattern

Azure Event Grid

Azure Blob Storage

Split a large message into a claim check and a payload. Send the claim check to the messaging platform and store the payload to an external service. This pattern allows large messages to be processed, while protecting the message bus and the client from being overwhelmed or slowed down. This pattern also helps to reduce costs, as storage is usually cheaper than resource units used by the messaging platform.

This pattern is also known as Reference-Based Messaging, and was originally [described](#) in the book *Enterprise Integration Patterns*, by Gregor Hohpe and Bobby Woolf.

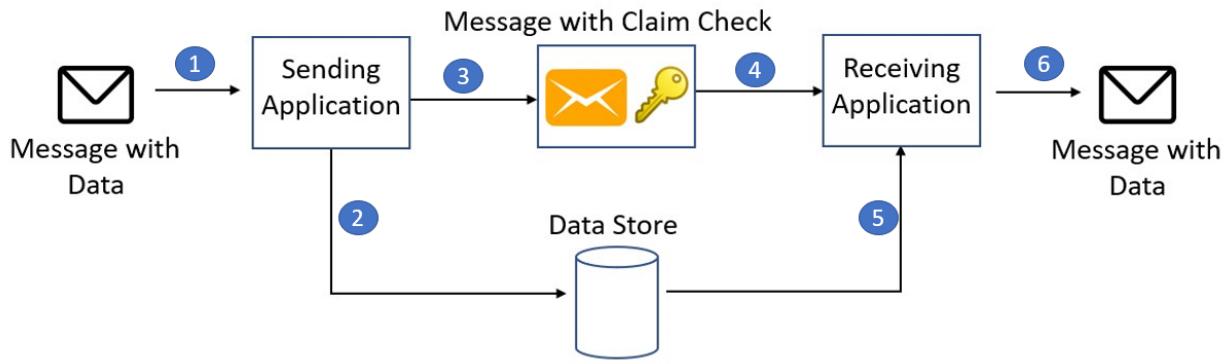
Context and problem

A messaging-based architecture at some point must be able to send, receive, and manipulate large messages. Such messages may contain anything, including images (for example, MRI scans), sound files (for example, call-center calls), text documents, or any kind of binary data of arbitrary size.

Sending such large messages to the message bus directly is not recommended, because they require more resources and bandwidth to be consumed. Large messages can also slow down the entire solution, because messaging platforms are usually fine-tuned to handle huge quantities of small messages. Also, most messaging platforms have limits on message size, so you may need to work around these limits for large messages.

Solution

Store the entire message payload into an external service, such as a database. Get the reference to the stored payload, and send just that reference to the message bus. The reference acts like a claim check used to retrieve a piece of luggage, hence the name of the pattern. Clients interested in processing that specific message can use the obtained reference to retrieve the payload, if needed.



1. Send message
2. Store message on the data store
3. Enqueue the message's reference
4. Read the message's reference
5. Retrieve the message
6. Process the message

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- Consider deleting the message data after consuming it, if you don't need to archive the messages. Although blob storage is relatively cheap, it costs some money in the long run, especially if there is a lot of data. Deleting the message can be done synchronously by the application that receives and processes the message, or asynchronously by a separate dedicated process. The asynchronous approach removes old data with no impact on the throughput and message processing performance of the receiving application.
- Storing and retrieving the message causes some additional overhead and latency. You may want to implement logic in the sending application to use this pattern only when the message size exceeds the data limit of the message bus. The pattern would be skipped for smaller messages. This approach would result in a conditional claim-check pattern.

When to use this pattern

This pattern could be used whenever a message cannot fit the supported message limit of the chosen message bus technology. For example, Service Bus currently has a limit of 100 MB (premium tier), while Event Grid supports up to 1 MB messages.

The pattern can also be used if the payload should be accessed only by services that are authorized to see it. By offloading the payload to an external resource, stricter

authentication and authorization rules can be put in place, to ensure that security is enforced when sensitive data is stored in the payload.

Examples

On Azure, this pattern can be implemented in several ways and with different technologies, but there are two main categories. In both cases, the receiver has the responsibility to read the claim check and use it to retrieve the payload.

- **Automatic claim-check generation.** This approach uses [Azure Event Grid](#) to automatically generate the claim check and push it into the message bus.
- **Manual claim-check generation.** In this approach, the sender is responsible for managing the payload. The sender stores the payload using the appropriate service, gets or generates the claim check, and sends the claim check to the message bus.

Event Grid is an event routing service and tries to deliver events within a configurable amount of time up to 24 hours. After that, events are either discarded or dead lettered. If you need to archive the event payloads or replay the event stream, you can add an Event Grid subscription to Event Hubs or Queue Storage, where messages can be retained for longer periods and archiving messages is supported. For information about fine tuning Event Grid message delivery and retry, and dead letter configuration, see [Dead letter and retry policies](#).

Automatic claim-check generation with Blob Storage and Event Grid

In this approach, the sender drops the message payload into a designated Azure Blob Storage container. Event Grid automatically generates a tag/reference and sends it to a supported message bus, such as Azure Storage Queues. The receiver can poll the queue, get the message, and then use the stored reference data to download the payload directly from Blob Storage.

The same Event Grid message can be directly consumed by [Azure Functions](#), without needing to go through a message bus. This approach takes full advantage of the serverless nature of both Event Grid and Functions.

You can find example code for this approach [here ↗](#).

Event Grid with Event Hubs

Similar to the previous example, Event Grid automatically generates a message when a payload is written to an Azure Blob container. But in this example, the message bus is implemented using Event Hubs. A client can register itself to receive the stream of messages as they are written to the event hub. The event hub can also be configured to archive messages, making them available as an Avro file that can be queried using tools like Apache Spark, Apache Drill, or any of the available Avro libraries.

You can find example code for this approach [here](#).

Claim check generation with Service Bus

This solution takes advantage of a specific Service Bus plugin, [ServiceBus.AttachmentPlugin](#), which makes the claim-check workflow easy to implement. The plugin converts any message body into an attachment that gets stored in Azure Blob Storage when the message is sent.

C#

```
using ServiceBus.AttachmentPlugin;
...

// Getting connection information
var serviceBusConnectionString =
Environment.GetEnvironmentVariable("SERVICE_BUS_CONNECTION_STRING");
var queueName = Environment.GetEnvironmentVariable("QUEUE_NAME");
var storageConnectionString =
Environment.GetEnvironmentVariable("STORAGE_CONNECTION_STRING");

// Creating config for sending message
var config = new
AzureStorageAttachmentConfiguration(storageConnectionString);

// Creating and registering the sender using Service Bus Connection String
// and Queue Name
var sender = new MessageSender(serviceBusConnectionString, queueName);
sender.RegisterAzureStorageAttachmentPlugin(config);

// Create payload
var payload = new { data = "random data string for testing" };
var serialized = JsonConvert.SerializeObject(payload);
var payloadAsBytes = Encoding.UTF8.GetBytes(serialized);
var message = new Message(payloadAsBytes);

// Send the message
await sender.SendAsync(message);
```

The Service Bus message acts as a notification queue, which a client can subscribe to. When the consumer receives the message, the plugin makes it possible to directly read

the message data from Blob Storage. You can then choose how to process the message further. An advantage of this approach is that it abstracts the claim-check workflow from the sender and receiver.

You can find example code for this approach [here ↗](#).

Manual claim-check generation with Kafka

In this example, a Kafka client writes the payload to Azure Blob Storage. Then it sends a notification message using [Kafka-enabled Event Hubs](#). The consumer receives the message and can access the payload from Blob Storage. This example shows how a different messaging protocol can be used to implement the claim-check pattern in Azure. For example, you might need to support existing Kafka clients.

You can find example code for this approach [here ↗](#).

Next steps

- The examples described above are available on [GitHub ↗](#).
- The Enterprise Integration Patterns site has a [description ↗](#) of this pattern.
- For another example, see [Dealing with large Service Bus messages using claim check pattern ↗](#) (blog post).
- An alternative pattern for handling large messages is [Split ↗](#) and [Aggregate ↗](#).
- Libraries like NServiceBus provide support for this pattern out-of-the-box with their "data bus" [functionality ↗](#).

Related resources

- [Asynchronous Request-Reply Pattern](#)
- [Competing Consumers pattern](#)
- [Sequential Convoy pattern](#)

Compensating Transaction pattern

Azure

When you use an eventually consistent operation that consists of a series of steps, the Compensating Transaction pattern can be useful. Specifically, if one or more of the steps fail, you can use the Compensating Transaction pattern to undo the work that the steps performed. Typically, you find operations that follow the eventual consistency model in cloud-hosted applications that implement complex business processes and workflows.

Context and problem

Applications that run in the cloud frequently modify data. This data is sometimes spread across various data sources in different geographic locations. To avoid contention and improve performance in a distributed environment, an application shouldn't try to provide strong transactional consistency. Rather, the application should implement eventual consistency. In the eventual consistency model, a typical business operation consists of a series of separate steps. While the operation performs these steps, the overall view of the system state might be inconsistent. But when the operation finishes and all the steps have run, the system should become consistent again.

The [Data Consistency Primer](#) provides information about why distributed transactions don't scale well. This resource also lists principles of the eventual consistency model.

A challenge in the eventual consistency model is how to handle a step that fails. After a failure, you might need to undo all the work that previous steps in the operation completed. However, you can't always roll back the data, because other concurrent instances of the application might have changed it. Even in cases where concurrent instances haven't changed the data, undoing a step might be more complex than restoring the original state. It might be necessary to apply various business-specific rules. For an example, see the travel website that the [Example](#) section describes later in this article.

If an operation that implements eventual consistency spans several heterogeneous data stores, undoing the steps in the operation requires visiting each data store in turn. To prevent the system from remaining inconsistent, you must reliably undo the work that you performed in every data store.

The data that's affected by an operation that implements eventual consistency isn't always held in a database. For example, consider a service-oriented architecture (SOA) environment. An SOA operation can invoke an action in a service and cause a change in the state that's held by that service. To undo the operation, you also have to undo this state change. This process can involve invoking the service again and performing another action that reverses the effects of the first.

Solution

The solution is to implement a compensating transaction. The steps in a compensating transaction undo the effects of the steps in the original operation. An intuitive approach is to replace the current state with the state the system was in at the start of the operation. But a compensating transaction can't always take that approach, because it might overwrite changes that other concurrent instances of an application have made. Instead, a compensating transaction must be an intelligent process that takes into account any work that concurrent instances do. This process is usually application-specific, driven by the nature of the work that the original operation performs.

A common approach is to use a workflow to implement an eventually consistent operation that requires compensation. As the original operation proceeds, the system records information about each step, including how to undo the work that the step performs. If the operation fails at any point, the workflow rewinds back through the steps it has completed. At each step, the workflow performs the work that reverses that step.

Two important points are:

- A compensating transaction might not have to undo the work in the exact reverse order of the original operation.
- It might be possible to perform some of the undo steps in parallel.

This approach is similar to the Sagas strategy that's discussed in [Clemens Vasters' blog](#).

A compensating transaction is an eventually consistent operation itself, so it can also fail. The system should be able to resume the compensating transaction at the point of failure and continue. It might be necessary to repeat a step that fails, so you should define the steps in a compensating transaction as [idempotent commands](#). For more information, see [Idempotency Patterns](#) on Jonathan Oliver's blog.

In some cases, manual intervention might be the only way to recover from a step that has failed. In these situations, the system should raise an alert and provide as much

information as possible about the reason for the failure.

Issues and considerations

Consider the following points when you decide how to implement this pattern:

- It might not be easy to determine when a step in an operation that implements eventual consistency fails. A step might not fail immediately. Instead, it might get blocked. You might need to implement a time-out mechanism.
- It's not easy to generalize compensation logic. A compensating transaction is application-specific. It relies on the application having sufficient information to be able to undo the effects of each step in a failed operation.
- You should define the steps in a compensating transaction as idempotent commands. If you do, the steps can be repeated if the compensating transaction itself fails.
- The infrastructure that handles the steps must meet the following criteria:
 - It's resilient in the original operation and in the compensating transaction.
 - It doesn't lose the information that's required to compensate for a failing step.
 - It reliably monitors the progress of the compensation logic.
- A compensating transaction doesn't necessarily return the system data to its state at the start of the original operation. Instead, the transaction compensates for the work that the operation completed successfully before it failed.
- The order of the steps in the compensating transaction isn't necessarily the exact opposite of the steps in the original operation. For example, one data store might be more sensitive to inconsistencies than another. The steps in the compensating transaction that undo the changes to this store should occur first.
- Certain measures can help increase the likelihood that the overall activity succeeds. Specifically, you can place a short-term, time-out-based lock on each resource that's required to complete an operation. You can also obtain these resources in advance. Then, perform the work only after you've acquired all the resources. Finalize all actions before the locks expire.
- Retry logic that's more forgiving than usual can help minimize failures that trigger a compensating transaction. If a step in an operation that implements eventual consistency fails, try handling the failure as a transient exception and repeating the step. Stop the operation and initiate a compensating transaction only if a step fails repeatedly or can't be recovered.

- When you implement a compensating transaction, you face many of the same challenges that you face when you implement eventual consistency. For more information, see the "Considerations for Implementing Eventual Consistency" section in [Data Consistency Primer](#).

When to use this pattern

Use this pattern only for operations that must be undone if they fail. If possible, design solutions to avoid the complexity of requiring compensating transactions.

Example

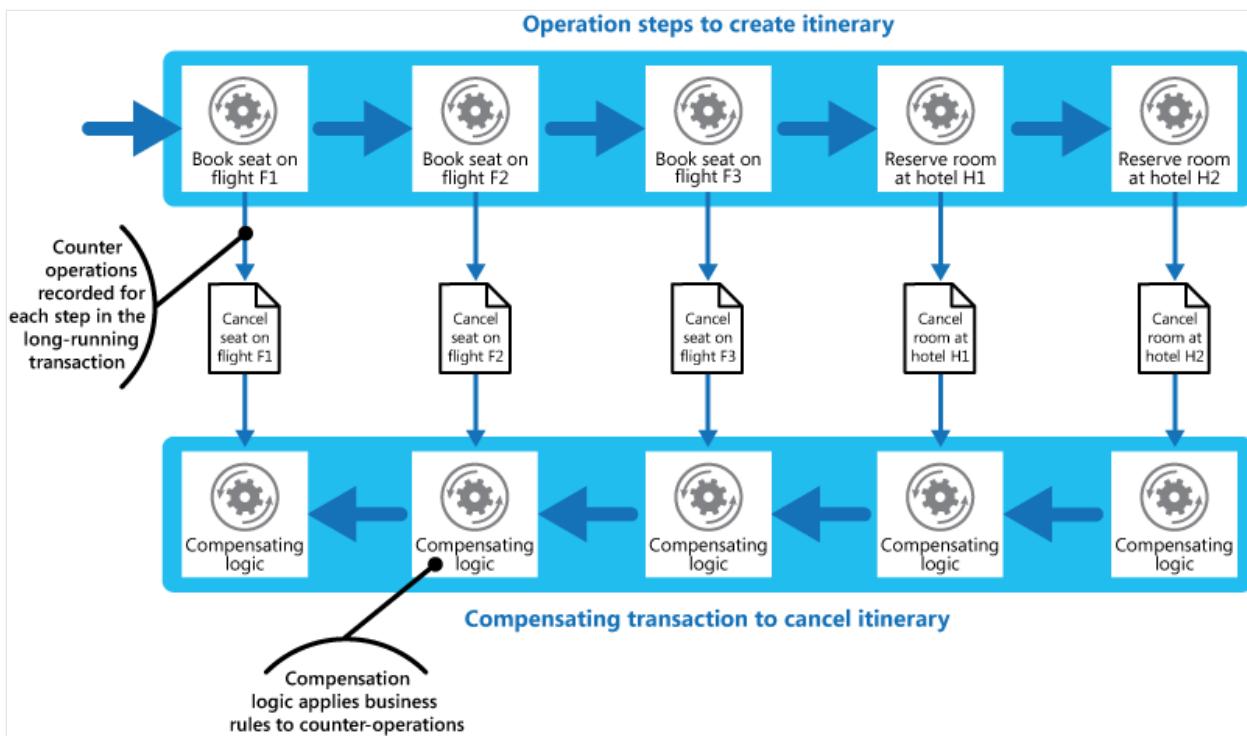
Customers use a travel website to book itineraries. A single itinerary might consist of a series of flights and hotels. A customer who travels from Seattle to London and then on to Paris might perform the following steps when creating an itinerary:

1. Book a seat on flight F1 from Seattle to London.
2. Book a seat on flight F2 from London to Paris.
3. Book a seat on flight F3 from Paris to Seattle.
4. Reserve a room at hotel H1 in London.
5. Reserve a room at hotel H2 in Paris.

These steps constitute an eventually consistent operation, although each step is a separate action. Besides performing these steps, the system must also record the counter operations for undoing each step. This information is needed in case the customer cancels the itinerary. The steps that are necessary to perform the counter operations can then run as a compensating transaction.

The steps in the compensating transaction might not be the exact opposite of the original steps. Also, the logic in each step in the compensating transaction must take business-specific rules into account. For example, canceling a flight reservation might not entitle the customer to a complete refund.

The following figure shows the steps in a long-running transaction for booking a travel itinerary. You can also see the compensating transaction steps that undo the transaction.



ⓘ Note

You might be able to perform the steps in the compensating transaction in parallel, depending on how you design the compensating logic for each step.

In many business solutions, failure of a single step doesn't always necessitate rolling back the system by using a compensating transaction. For example, consider the travel website scenario. Suppose the customer books flights F1, F2, and F3 but can't reserve a room at hotel H1. It's preferable to offer the customer a room at a different hotel in the same city rather than canceling the flights. The customer can still decide to cancel. In that case, the compensating transaction runs and undoes the bookings for flights F1, F2, and F3. But the customer should make this decision, not the system.

Next steps

- [Data Consistency Primer](#). The Compensating Transaction pattern is often used to undo operations that implement the eventual consistency model. This primer provides information about the benefits and trade-offs of eventual consistency.
- [Idempotency Patterns ↗](#). In a compensating transaction, it's best to use idempotent commands. This blog post describes factors to consider when you implement idempotency.

Related resources

- [Scheduler Agent Supervisor pattern](#). This article describes how to implement resilient systems that perform business operations that use distributed services and resources. In these systems, you sometimes need to use a compensating transaction to undo the work that an operation performs.
- [Retry pattern](#). Compensating transactions can be computationally demanding. You can try to minimize their use by using the Retry pattern to implement an effective policy of retrying failed operations.
- [Saga distributed transactions pattern](#). This article explains how to use the Saga pattern to manage data consistency across microservices in distributed transaction scenarios. The Saga pattern handles failure recovery with compensating transactions.
- [Pipes and Filters pattern](#). This article describes the Pipes and Filters pattern, which you can use to decompose a complex processing task into a series of reusable elements. You can use the Pipes and Filters pattern with the Compensating Transaction pattern as an alternative to implementing distributed transactions.
- [Design for self healing](#). This guide explains how to design self-healing applications. You can use compensating transactions as part of a self-healing approach.

Competing Consumers pattern

Azure Functions

Azure Service Bus

Enable multiple concurrent consumers to process messages received on the same messaging channel. With multiple concurrent consumers, a system can process multiple messages concurrently to optimize throughput, to improve scalability and availability, and to balance the workload.

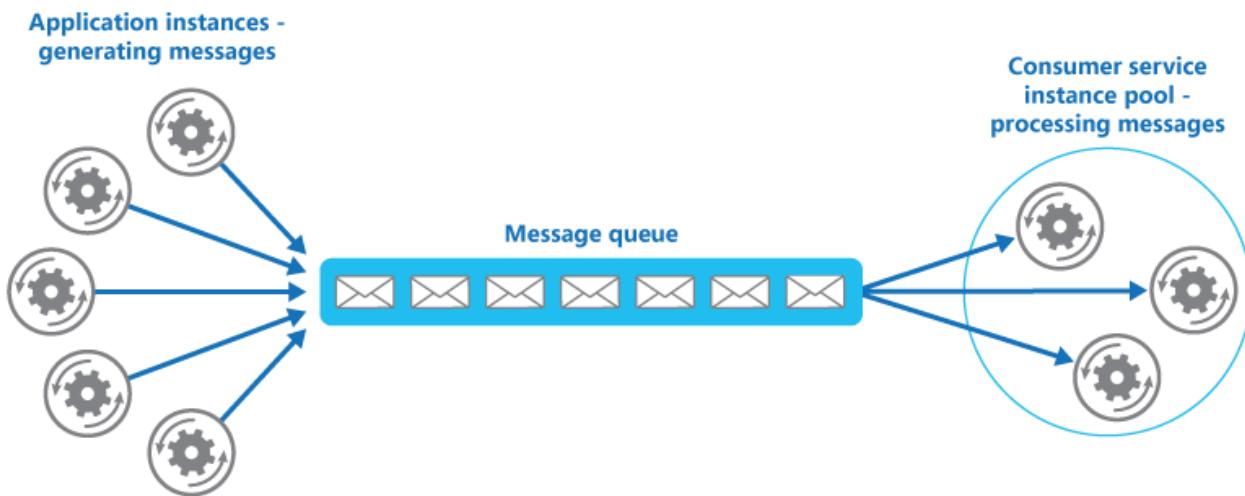
Context and problem

An application running in the cloud is expected to handle a large number of requests. Rather than process each request synchronously, a common technique is for the application to pass them through a messaging system to another service (a consumer service) that handles them asynchronously. This strategy helps to ensure that the business logic in the application isn't blocked, while the requests are being processed.

The number of requests can vary significantly over time for many reasons. A sudden increase in user activity or aggregated requests coming from multiple tenants can cause an unpredictable workload. At peak hours, a system might need to process many hundreds of requests per second, while at other times the number could be very small. Additionally, the nature of the work performed to handle these requests might be highly variable. By using a single instance of the consumer service, you can cause that instance to become flooded with requests. Or, the messaging system might be overloaded by an influx of messages that come from the application. To handle this fluctuating workload, the system can run multiple instances of the consumer service. However, these consumers must be coordinated to ensure that each message is only delivered to a single consumer. The workload also needs to be load balanced across consumers to prevent an instance from becoming a bottleneck.

Solution

Use a message queue to implement the communication channel between the application and the instances of the consumer service. The application posts requests in the form of messages to the queue, and the consumer service instances receive messages from the queue and process them. This approach enables the same pool of consumer service instances to handle messages from any instance of the application. The figure illustrates using a message queue to distribute work to instances of a service.



This solution has the following benefits:

- It provides a load-leveled system that can handle wide variations in the volume of requests sent by application instances. The queue acts as a buffer between the application instances and the consumer service instances. This buffer can help minimize the impact on availability and responsiveness, for both the application and the service instances. For more information, see [Queue-based Load Leveling pattern](#). Handling a message that requires some long-running processing doesn't prevent other messages from being handled concurrently by other instances of the consumer service.
- It improves reliability. If a producer communicates directly with a consumer instead of using this pattern, but doesn't monitor the consumer, there's a high probability that messages could be lost or fail to be processed if the consumer fails. In this pattern, messages aren't sent to a specific service instance. A failed service instance won't block a producer, and messages can be processed by any working service instance.
- It doesn't require complex coordination between the consumers, or between the producer and the consumer instances. The message queue ensures that each message is delivered at least once.
- It's scalable. When you apply [auto-scaling](#), the system can dynamically increase or decrease the number of instances of the consumer service as the volume of messages fluctuates.
- It can improve resiliency if the message queue provides transactional read operations. If a consumer service instance reads and processes the message as part of a transactional operation, and the consumer service instance fails, this pattern can ensure that the message will be returned to the queue to be picked up and handled by another instance of the consumer service. In order to mitigate the risk

of a message continuously failing, we recommend you make use of [dead-letter queues](#).

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- **Message ordering.** The order in which consumer service instances receive messages isn't guaranteed, and doesn't necessarily reflect the order in which the messages were created. Design the system to ensure that message processing is idempotent because this will help to eliminate any dependency on the order in which messages are handled. For more information, see [Idempotency Patterns](#) on Jonathon Oliver's blog.

Microsoft Azure Service Bus Queues can implement guaranteed first-in-first-out ordering of messages by using message sessions. For more information, see [Messaging Patterns Using Sessions](#).

- **Designing services for resiliency.** If the system is designed to detect and restart failed service instances, it might be necessary to implement the processing performed by the service instances as idempotent operations to minimize the effects of a single message being retrieved and processed more than once.
- **Detecting poison messages.** A malformed message, or a task that requires access to resources that aren't available, can cause a service instance to fail. The system should prevent such messages being returned to the queue, and instead capture and store the details of these messages elsewhere so that they can be analyzed if necessary.
- **Handling results.** The service instance handling a message is fully decoupled from the application logic that generates the message, and they might not be able to communicate directly. If the service instance generates results that must be passed back to the application logic, this information must be stored in a location that's accessible to both. In order to prevent the application logic from retrieving incomplete data the system must indicate when processing is complete.

If you're using Azure, a worker process can pass results back to the application logic by using a dedicated message reply queue. The application logic must be able to correlate these results with the original message. This scenario is described in more detail in the [Asynchronous Messaging Primer](#).

- **Scaling the messaging system.** In a large-scale solution, a single message queue could be overwhelmed by the number of messages and become a bottleneck in the system. In this situation, consider partitioning the messaging system to send messages from specific producers to a particular queue, or use load balancing to distribute messages across multiple message queues.
- **Ensuring reliability of the messaging system.** A reliable messaging system is needed to guarantee that after the application enqueues a message it won't be lost. This system is essential for ensuring that all messages are delivered at least once.

When to use this pattern

Use this pattern when:

- The workload for an application is divided into tasks that can run asynchronously.
- Tasks are independent and can run in parallel.
- The volume of work is highly variable, requiring a scalable solution.
- The solution must provide high availability, and must be resilient if the processing for a task fails.

This pattern might not be useful when:

- It's not easy to separate the application workload into discrete tasks, or there's a high degree of dependence between tasks.
- Tasks must be performed synchronously, and the application logic must wait for a task to complete before continuing.
- Tasks must be performed in a specific sequence.

Some messaging systems support sessions that enable a producer to group messages together and ensure that they're all handled by the same consumer. This mechanism can be used with prioritized messages (if they are supported) to implement a form of message ordering that delivers messages in sequence from a producer to a single consumer.

Example

Azure provides Service Bus Queues and Azure Function queue triggers that, when combined, are a direct implementation of this cloud design pattern. Azure Functions integrate with Azure Service Bus via triggers and bindings. Integrating with Service Bus allows you to build functions that consume queue messages sent by publishers. The

publishing application(s) will post messages to a queue, and consumers, implemented as Azure Functions, can retrieve messages from this queue and handle them.

For resiliency, a Service Bus queue enables a consumer to use `PeekLock` mode when it retrieves a message from the queue; this mode doesn't actually remove the message, but simply hides it from other consumers. The Azure Functions runtime receives a message in `PeekLock` mode, if the function finishes successfully it calls `Complete` on the message, or it may call `Abandon` if the function fails, and the message will become visible again, allowing another consumer to retrieve it. If the function runs for a period longer than the `PeekLock` timeout, the lock is automatically renewed as long as the function is running.

Azure Functions can scale out/in based on the depth of the queue, all acting as competing consumers of the queue. If multiple instances of the functions are created they all compete by independently pulling and processing the messages.

For detailed information on using Azure Service Bus queues, see [Service Bus queues, topics, and subscriptions](#).

For information on Queue triggered Azure Functions, see [Azure Service Bus trigger for Azure Functions](#).

The following code shows how you can create a new message and send it to a Service Bus Queue by using a `ServiceBusClient` instance.

C#

```
private string serviceBusConnectionString = ...;
...

public async Task SendMessagesAsync(CancellationToken ct)
{
    try
    {
        var msgNumber = 0;

        var serviceBusClient = new ServiceBusClient(serviceBusConnectionString);

        // create the sender
        ServiceBusSender sender = serviceBusClient.CreateSender("myqueue");

        while (!ct.IsCancellationRequested)
        {
            // Create a new message to send to the queue
            string messageBody = $"Message {msgNumber}";
            var message = new ServiceBusMessage(messageBody);

            // Write the body of the message to the console
            await sender.SendMessageAsync(message);
            msgNumber++;
        }
    }
}
```

```

        this._logger.LogInformation($"Sending message: {messageBody}");

        // Send the message to the queue
        await sender.SendMessageAsync(message);

        this._logger.LogInformation("Message successfully sent.");
        msgNumber++;
    }
}

catch (Exception exception)
{
    this._logger.LogError(exception.Message);
}
}

```

The following code example shows a consumer, written as a C# Azure Function, that reads message metadata and logs a Service Bus Queue message. Note how the `ServiceBusTrigger` attribute is used to bind it to a Service Bus Queue.

C#

```

[FunctionName("ProcessQueueMessage")]
public static void Run(
    [ServiceBusTrigger("myqueue", Connection =
"ServiceBusConnectionString")]
    string myQueueItem,
    Int32 deliveryCount,
    DateTime enqueuedTimeUtc,
    string messageId,
    ILogger log)
{
    log.LogInformation($"C# ServiceBus queue trigger function consumed
message: {myQueueItem}");
    log.LogInformation($"EnqueuedTimeUtc={enqueuedTimeUtc}");
    log.LogInformation($"DeliveryCount={deliveryCount}");
    log.LogInformation($"MessageId={messageId}");
}

```

Next steps

- [Asynchronous Messaging Primer](#). Message queues are an asynchronous communications mechanism. If a consumer service needs to send a reply to an application, it might be necessary to implement some form of response messaging. The Asynchronous Messaging Primer provides information on how to implement request/reply messaging using message queues.
- [Autoscaling Guidance](#). It might be possible to start and stop instances of a consumer service since the length of the queue applications post messages on

varies. Autoscaling can help to maintain throughput during times of peak processing.

Related resources

The following patterns and guidance might be relevant when implementing this pattern:

- [Compute Resource Consolidation pattern](#). It might be possible to consolidate multiple instances of a consumer service into a single process to reduce costs and management overhead. The Compute Resource Consolidation pattern describes the benefits and tradeoffs of following this approach.
- [Queue-based Load Leveling pattern](#). Introducing a message queue can add resiliency to the system, enabling service instances to handle widely varying volumes of requests from application instances. The message queue acts as a buffer, which levels the load. The Queue-based Load Leveling pattern describes this scenario in more detail.

Compute Resource Consolidation pattern

Azure App Service

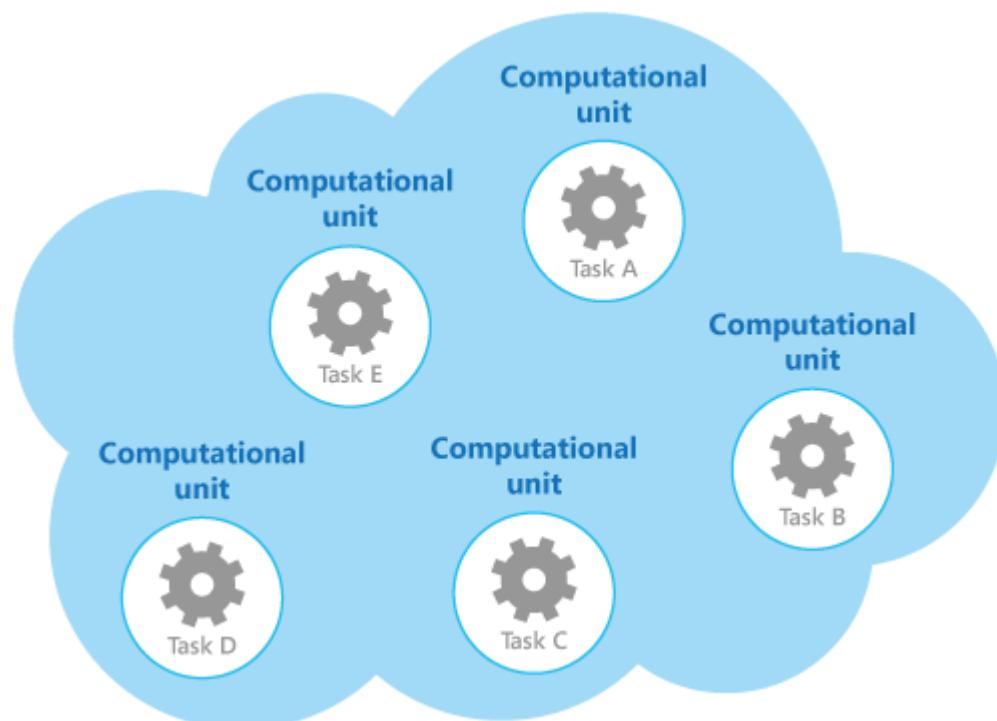
Azure Kubernetes Service (AKS)

Consolidate multiple tasks or operations into a single computational unit. This can increase compute resource utilization, and reduce the costs and management overhead associated with performing compute processing in cloud-hosted applications.

Context and problem

A cloud application often implements a variety of operations. In some solutions it makes sense to follow the design principle of separation of concerns initially, and divide these operations into separate computational units that are hosted and deployed individually (for example, as separate App Service web apps or separate Virtual Machines). However, although this strategy can help simplify the logical design of the solution, deploying a large number of computational units as part of the same application can increase runtime hosting costs and make management of the system more complex.

As an example, the figure shows the simplified structure of a cloud-hosted solution that is implemented using more than one computational unit. Each computational unit runs in its own virtual environment. Each function has been implemented as a separate task (labeled Task A through Task E) running in its own computational unit.



Each computational unit consumes chargeable resources, even when it's idle or lightly used. Therefore, this isn't always the most cost-effective solution.

In Azure, this concern applies to App Services, Container Apps, and Virtual Machines. These items run in their own environment. Running a collection of separate websites, microservices, or virtual machines that are designed to perform a set of well-defined operations, but that need to communicate and cooperate as part of a single solution, can be an inefficient use of resources.

Solution

To help reduce costs, increase utilization, improve communication speed, and reduce management it's possible to consolidate multiple tasks or operations into a single computational unit.

Tasks can be grouped according to criteria based on the features provided by the environment and the costs associated with these features. A common approach is to look for tasks that have a similar profile concerning their scalability, lifetime, and processing requirements. Grouping these together allows them to scale as a unit. The elasticity provided by many cloud environments enables additional instances of a computational unit to be started and stopped according to the workload. For example, Azure provides autoscaling that you can apply to App Services and Virtual Machine Scale Sets. For more information, see [Autoscaling Guidance](#).

As a counter example to show how scalability can be used to determine which operations shouldn't be grouped together, consider the following two tasks:

- Task 1 polls for infrequent, time-insensitive messages sent to a queue.
- Task 2 handles high-volume bursts of network traffic.

The second task requires elasticity that can involve starting and stopping a large number of instances of the computational unit. Applying the same scaling to the first task would simply result in more tasks listening for infrequent messages on the same queue, and is a waste of resources.

In many cloud environments it's possible to specify the resources available to a computational unit in terms of the number of CPU cores, memory, disk space, and so on. Generally, the more resources specified, the greater the cost. To save money, it's important to maximize the work an expensive computational unit performs, and not let it become inactive for an extended period.

If there are tasks that require a great deal of CPU power in short bursts, consider consolidating these into a single computational unit that provides the necessary power.

However, it's important to balance this need to keep expensive resources busy against the contention that could occur if they are over stressed. Long-running, compute-intensive tasks shouldn't share the same computational unit, for example.

Issues and considerations

Consider the following points when implementing this pattern:

Scalability and elasticity. Many cloud solutions implement scalability and elasticity at the level of the computational unit by starting and stopping instances of units. Avoid grouping tasks that have conflicting scalability requirements in the same computational unit.

Lifetime. The cloud infrastructure periodically recycles the virtual environment that hosts a computational unit. When there are many long-running tasks inside a computational unit, it might be necessary to configure the unit to prevent it from being recycled until these tasks have finished. Alternatively, design the tasks by using a check-pointing approach that enables them to stop cleanly, and continue at the point they were interrupted when the computational unit is restarted.

Release cadence. If the implementation or configuration of a task changes frequently, it might be necessary to stop the computational unit hosting the updated code, reconfigure and redeploy the unit, and then restart it. This process will also require that all other tasks within the same computational unit are stopped, redeployed, and restarted.

Security. Tasks in the same computational unit might share the same security context and be able to access the same resources. There must be a high degree of trust between the tasks, and confidence that one task isn't going to corrupt or adversely affect another. Additionally, increasing the number of tasks running in a computational unit increases the attack surface of the unit. Each task is only as secure as the one with the most vulnerabilities.

Fault tolerance. If one task in a computational unit fails or behaves abnormally, it can affect the other tasks running within the same unit. For example, if one task fails to start correctly it can cause the entire startup logic for the computational unit to fail, and prevent other tasks in the same unit from running.

Contention. Avoid introducing contention between tasks that compete for resources in the same computational unit. Ideally, tasks that share the same computational unit should exhibit different resource utilization characteristics. For example, two compute-intensive tasks should probably not reside in the same computational unit, and neither

should two tasks that consume large amounts of memory. However, mixing a compute-intensive task with a task that requires a large amount of memory is a workable combination.

ⓘ Note

Consider consolidating compute resources only for a system that's been in production for a period of time so that operators and developers can monitor the system and create a *heat map* that identifies how each task uses differing resources. This map can be used to determine which tasks are good candidates for sharing compute resources.

Complexity. Combining multiple tasks into a single computational unit adds complexity to the code in the unit, possibly making it more difficult to test, debug, and maintain.

Stable logical architecture. Design and implement the code in each task so that it shouldn't need to change, even if the physical environment the task runs in does change.

Other strategies. Consolidating compute resources is only one way to help reduce costs associated with running multiple tasks concurrently. It requires careful planning and monitoring to ensure that it remains an effective approach. Other strategies might be more appropriate, depending on the nature of the work and where the users these tasks are running are located. For example, functional decomposition of the workload (as described by the [Compute Partitioning Guidance](#)) might be a better option.

When to use this pattern

Use this pattern for tasks that are not cost effective if they run in their own computational units. If a task spends much of its time idle, running this task in a dedicated unit can be expensive.

This pattern might not be suitable for tasks that perform critical fault-tolerant operations, or tasks that process highly sensitive or private data and require their own security context. These tasks should run in their own isolated environment, in a separate computational unit.

Application platform choices

This pattern can be achieved in different ways, depending on the compute service you use. See the following example services:

- **Azure App Service and Azure Functions:** Deploy shared App Service plans, which represent the hosting server infrastructure. One or more apps can be configured to run on the same computing resources (or in the same App Service plan).
- **Azure Container Apps:** Deploy container apps to the same shared environments; especially in situations when you need to manage related services or you need to deploy different applications to the same virtual network.
- **Azure Kubernetes Service (AKS):** AKS is a container-based hosting infrastructure in which multiple applications or application components can be configured to run co-located on the same computing resources (nodes), grouped by computational requirements such as CPU or memory needs (node pools).
- **Virtual machines:** Deploy a single set of virtual machines for all tenants to use, that way the management costs are shared across the tenants. Virtual Machine Scale Sets is a feature that supports shared resource management, load-balancing, and horizontal scaling of Virtual Machines.

Related resources

The following patterns and guidance might also be relevant when implementing this pattern:

- [Autoscaling Guidance](#). Autoscaling can be used to start and stop instances of service hosting computational resources, depending on the anticipated demand for processing.
- [Compute Partitioning Guidance](#). Describes how to allocate the services and components in a cloud service in a way that helps to minimize running costs while maintaining the scalability, performance, availability, and security of the service.
- [Architectural approaches for compute in multitenant solutions](#). Provides guidance about the considerations and requirements that are essential for solution architects, when they're planning the compute services of a multitenant solution.

CQRS pattern

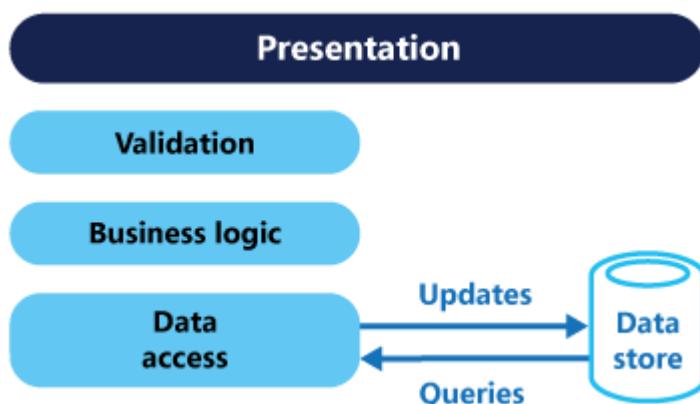
Azure Storage

CQRS stands for Command and Query Responsibility Segregation, a pattern that separates read and update operations for a data store. Implementing CQRS in your application can maximize its performance, scalability, and security. The flexibility created by migrating to CQRS allows a system to better evolve over time and prevents update commands from causing merge conflicts at the domain level.

Context and problem

In traditional architectures, the same data model is used to query and update a database. That's simple and works well for basic CRUD operations. In more complex applications, however, this approach can become unwieldy. For example, on the read side, the application may perform many different queries, returning data transfer objects (DTOs) with different shapes. Object mapping can become complicated. On the write side, the model may implement complex validation and business logic. As a result, you can end up with an overly complex model that does too much.

Read and write workloads are often asymmetrical, with very different performance and scale requirements.



- There is often a mismatch between the read and write representations of the data, such as additional columns or properties that must be updated correctly even though they aren't required as part of an operation.
- Data contention can occur when operations are performed in parallel on the same set of data.
- The traditional approach can have a negative effect on performance due to load on the data store and data access layer, and the complexity of queries required to

retrieve information.

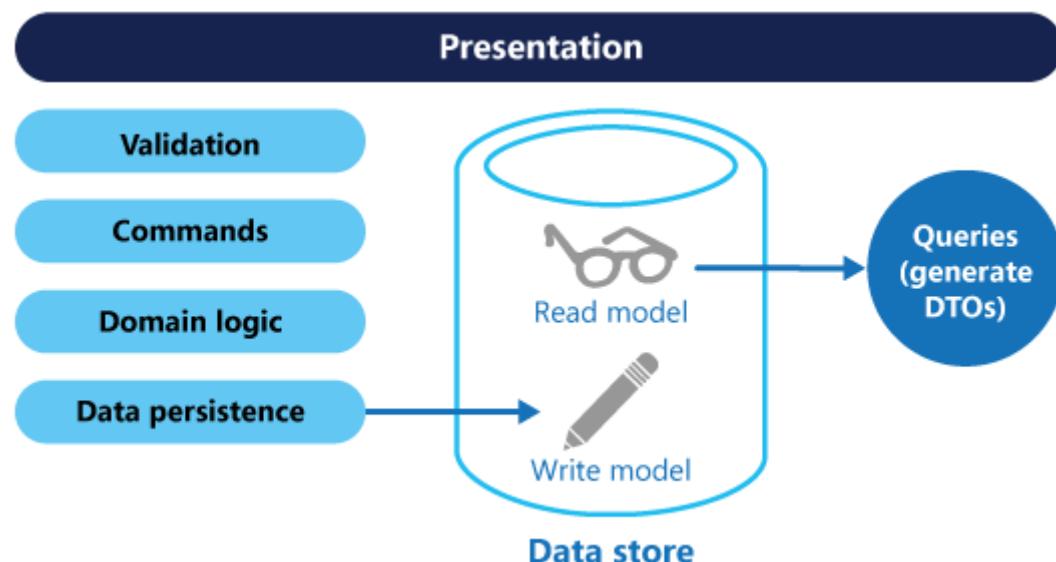
- Managing security and permissions can become complex, because each entity is subject to both read and write operations, which might expose data in the wrong context.

Solution

CQRS separates reads and writes into different models, using **commands** to update data, and **queries** to read data.

- Commands should be task-based, rather than data centric. ("Book hotel room", not "set ReservationStatus to Reserved"). This may require some corresponding changes to the user interaction style. The other part of that is to look at modifying the business logic processing those commands to be successful more frequently. One technique that supports this is to run some validation rules on the client even before sending the command, possibly disabling buttons, explaining why on the UI ("no rooms left"). In that manner, the cause for server-side command failures can be narrowed to race conditions (two users trying to book the last room), and even those can sometimes be addressed with some more data and logic (putting a guest on a waiting list).
- Commands may be placed on a queue for **asynchronous processing**, rather than being processed synchronously.
- Queries never modify the database. A query returns a DTO that does not encapsulate any domain knowledge.

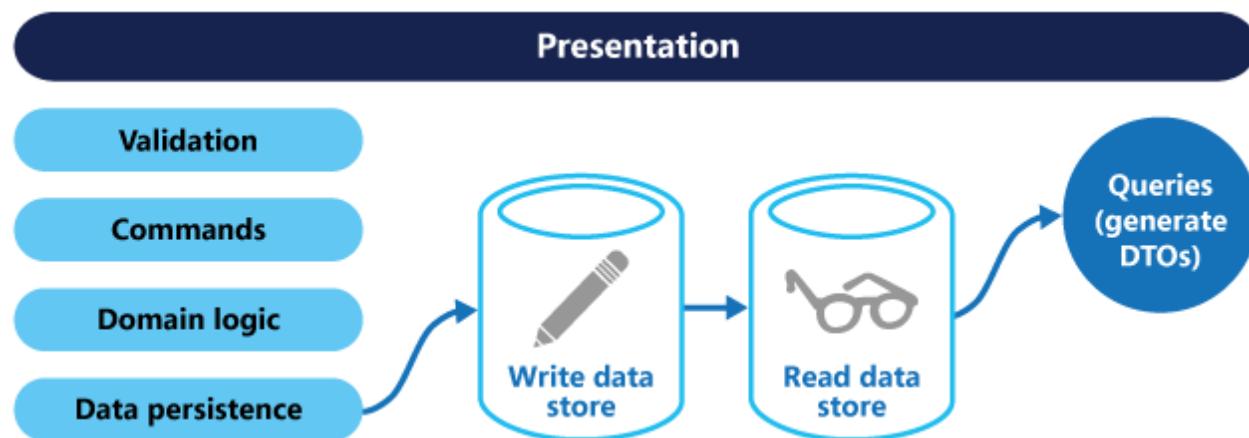
The models can then be isolated, as shown in the following diagram, although that's not an absolute requirement.



Having separate query and update models simplifies the design and implementation. However, one disadvantage is that CQRS code can't automatically be generated from a database schema using scaffolding mechanisms such as O/RM tools (However, you will be able to build your customization on top of the generated code).

For greater isolation, you can physically separate the read data from the write data. In that case, the read database can use its own data schema that is optimized for queries. For example, it can store a [materialized view](#) of the data, in order to avoid complex joins or complex O/RM mappings. It might even use a different type of data store. For example, the write database might be relational, while the read database is a document database.

If separate read and write databases are used, they must be kept in sync. Typically this is accomplished by having the write model publish an event whenever it updates the database. For more information about using events, see [Event-driven architecture style](#). Since message brokers and databases usually cannot be enlisted into a single distributed transaction, there can be challenges in guaranteeing consistency when updating the database and publishing events. For more information, see the [guidance on idempotent message processing](#).



The read store can be a read-only replica of the write store, or the read and write stores can have a different structure altogether. Using multiple read-only replicas can increase query performance, especially in distributed scenarios where read-only replicas are located close to the application instances.

Separation of the read and write stores also allows each to be scaled appropriately to match the load. For example, read stores typically encounter a much higher load than write stores.

Some implementations of CQRS use the [Event Sourcing pattern](#). With this pattern, application state is stored as a sequence of events. Each event represents a set of changes to the data. The current state is constructed by replaying the events. In a CQRS context, one benefit of Event Sourcing is that the same events can be used to notify

other components — in particular, to notify the read model. The read model uses the events to create a snapshot of the current state, which is more efficient for queries. However, Event Sourcing adds complexity to the design.

Benefits of CQRS include:

- **Independent scaling.** CQRS allows the read and write workloads to scale independently, and may result in fewer lock contentions.
- **Optimized data schemas.** The read side can use a schema that is optimized for queries, while the write side uses a schema that is optimized for updates.
- **Security.** It's easier to ensure that only the right domain entities are performing writes on the data.
- **Separation of concerns.** Segregating the read and write sides can result in models that are more maintainable and flexible. Most of the complex business logic goes into the write model. The read model can be relatively simple.
- **Simpler queries.** By storing a materialized view in the read database, the application can avoid complex joins when querying.

Implementation issues and considerations

Some challenges of implementing this pattern include:

- **Complexity.** The basic idea of CQRS is simple. But it can lead to a more complex application design, especially if they include the Event Sourcing pattern.
- **Messaging.** Although CQRS does not require messaging, it's common to use messaging to process commands and publish update events. In that case, the application must handle message failures or duplicate messages. See the guidance on [Priority Queues](#) for dealing with commands having different priorities.
- **Eventual consistency.** If you separate the read and write databases, the read data may be stale. The read model store must be updated to reflect changes to the write model store, and it can be difficult to detect when a user has issued a request based on stale read data.

When to use CQRS pattern

Consider CQRS for the following scenarios:

- Collaborative domains where many users access the same data in parallel. CQRS allows you to define commands with enough granularity to minimize merge

conflicts at the domain level, and conflicts that do arise can be merged by the command.

- Task-based user interfaces where users are guided through a complex process as a series of steps or with complex domain models. The write model has a full command-processing stack with business logic, input validation, and business validation. The write model may treat a set of associated objects as a single unit for data changes (an aggregate, in DDD terminology) and ensure that these objects are always in a consistent state. The read model has no business logic or validation stack, and just returns a DTO for use in a view model. The read model is eventually consistent with the write model.
- Scenarios where performance of data reads must be fine-tuned separately from performance of data writes, especially when the number of reads is much greater than the number of writes. In this scenario, you can scale out the read model, but run the write model on just a few instances. A small number of write model instances also helps to minimize the occurrence of merge conflicts.
- Scenarios where one team of developers can focus on the complex domain model that is part of the write model, and another team can focus on the read model and the user interfaces.
- Scenarios where the system is expected to evolve over time and might contain multiple versions of the model, or where business rules change regularly.
- Integration with other systems, especially in combination with event sourcing, where the temporal failure of one subsystem shouldn't affect the availability of the others.

This pattern isn't recommended when:

- The domain or the business rules are simple.
- A simple CRUD-style user interface and data access operations are sufficient.

Consider applying CQRS to limited sections of your system where it will be most valuable.

Event Sourcing and CQRS pattern

The CQRS pattern is often used along with the Event Sourcing pattern. CQRS-based systems use separate read and write data models, each tailored to relevant tasks and often located in physically separate stores. When used with the [Event Sourcing pattern](#), the store of events is the write model, and is the official source of information. The read

model of a CQRS-based system provides materialized views of the data, typically as highly denormalized views. These views are tailored to the interfaces and display requirements of the application, which helps to maximize both display and query performance.

Using the stream of events as the write store, rather than the actual data at a point in time, avoids update conflicts on a single aggregate and maximizes performance and scalability. The events can be used to asynchronously generate materialized views of the data that are used to populate the read store.

Because the event store is the official source of information, it is possible to delete the materialized views and replay all past events to create a new representation of the current state when the system evolves, or when the read model must change. The materialized views are in effect a durable read-only cache of the data.

When using CQRS combined with the Event Sourcing pattern, consider the following:

- As with any system where the write and read stores are separate, systems based on this pattern are only eventually consistent. There will be some delay between the event being generated and the data store being updated.
- The pattern adds complexity because code must be created to initiate and handle events, and assemble or update the appropriate views or objects required by queries or a read model. The complexity of the CQRS pattern when used with the Event Sourcing pattern can make a successful implementation more difficult, and requires a different approach to designing systems. However, event sourcing can make it easier to model the domain, and makes it easier to rebuild views or create new ones because the intent of the changes in the data is preserved.
- Generating materialized views for use in the read model or projections of the data by replaying and handling the events for specific entities or collections of entities can require significant processing time and resource usage. This is especially true if it requires summation or analysis of values over long periods, because all the associated events might need to be examined. Resolve this by implementing snapshots of the data at scheduled intervals, such as a total count of the number of a specific action that has occurred, or the current state of an entity.

Example of CQRS pattern

The following code shows some extracts from an example of a CQRS implementation that uses different definitions for the read and the write models. The model interfaces

don't dictate any features of the underlying data stores, and they can evolve and be fine-tuned independently because these interfaces are separated.

The following code shows the read model definition.

```
C#
```

```
// Query interface
namespace ReadModel
{
    public interface ProductsDao
    {
        ProductDisplay FindById(int productId);
        ICollection<ProductDisplay> FindByName(string name);
        ICollection<ProductInventory> FindOutOfStockProducts();
        ICollection<ProductDisplay> FindRelatedProducts(int productId);
    }

    public class ProductDisplay
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public decimal UnitPrice { get; set; }
        public bool IsOutOfStock { get; set; }
        public double UserRating { get; set; }
    }

    public class ProductInventory
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int CurrentStock { get; set; }
    }
}
```

The system allows users to rate products. The application code does this using the `RateProduct` command shown in the following code.

```
C#
```

```
public interface ICommand
{
    Guid Id { get; }
}

public class RateProduct : ICommand
{
    public RateProduct()
    {
        this.Id = Guid.NewGuid();
```

```
    }
    public Guid Id { get; set; }
    public int ProductId { get; set; }
    public int Rating { get; set; }
    public int UserId { get; set; }
}
```

The system uses the `ProductsCommandHandler` class to handle commands sent by the application. Clients typically send commands to the domain through a messaging system such as a queue. The command handler accepts these commands and invokes methods of the domain interface. The granularity of each command is designed to reduce the chance of conflicting requests. The following code shows an outline of the `ProductsCommandHandler` class.

C#

```
public class ProductsCommandHandler :
    ICommandHandler<AddNewProduct>,
    ICommandHandler<RateProduct>,
    ICommandHandler<AddToInventory>,
    ICommandHandler<ConfirmItemShipped>,
    ICommandHandler<UpdateStockFromInventoryRecount>
{
    private readonly IRepository<Product> repository;

    public ProductsCommandHandler ( IRepository<Product> repository )
    {
        this.repository = repository;
    }

    void Handle ( AddNewProduct command )
    {
        ...
    }

    void Handle ( RateProduct command )
    {
        var product = repository.Find ( command.ProductId );
        if ( product != null )
        {
            product.RateProduct ( command.UserId, command.Rating );
            repository.Save ( product );
        }
    }

    void Handle ( AddToInventory command )
    {
        ...
    }

    void Handle ( ConfirmItemsShipped command )
```

```
{  
  ...  
}  
  
void Handle (UpdateStockFromInventoryRecount command)  
{  
  ...  
}  
}
```

Next steps

The following patterns and guidance are useful when implementing this pattern:

- [Data Consistency Primer](#). Explains the issues that are typically encountered due to eventual consistency between the read and write data stores when using the CQRS pattern, and how these issues can be resolved.
- [Horizontal, vertical, and functional data partitioning](#). Describes best practices for dividing data into partitions that can be managed and accessed separately to improve scalability, reduce contention, and optimize performance.
- The patterns & practices guide [CQRS Journey](#). In particular, [Introducing the Command Query Responsibility Segregation pattern](#) explores the pattern and when it's useful, and [Epilogue: Lessons Learned](#) helps you understand some of the issues that come up when using this pattern.

Martin Fowler's blog posts:

- [What do you mean by "Event-Driven"? ↗](#)
- [CQRS ↗](#)

Related resources

- [Event Sourcing pattern](#). Describes in more detail how Event Sourcing can be used with the CQRS pattern to simplify tasks in complex domains while improving performance, scalability, and responsiveness. As well as how to provide consistency for transactional data while maintaining full audit trails and history that can enable compensating actions.
- [Materialized View pattern](#). The read model of a CQRS implementation can contain materialized views of the write model data, or the read model can be used to generate materialized views.

- Presentation on better CQRS through asynchronous user interaction patterns ↗

Deployment Stamps pattern

Azure Front Door Azure

The deployment stamp pattern involves provisioning, managing, and monitoring a heterogeneous group of resources to host and operate multiple workloads or tenants. Each individual copy is called a *stamp*, or sometimes a *service unit*, *scale unit*, or *cell*. In a multitenant environment, every stamp or scale unit can serve a predefined number of tenants. Multiple stamps can be deployed to scale the solution almost linearly and serve an increasing number of tenants. This approach can improve the scalability of your solution, allow you to deploy instances across multiple regions, and separate your customer data.

ⓘ Note

For more information about designing multitenant solutions for Azure, see [Architect multitenant solutions on Azure](#).

Context and problem

When hosting an application in the cloud, it's important to consider the performance and reliability of your application. If you host a single instance of your solution, you might be subject to the following limitations:

- **Scale limits.** Deploying a single instance of your application might result in natural scaling limits. For example, you might use services that have limits on the number of inbound connections, host names, TCP sockets, or other resources.
- **Non-linear scaling or cost.** Some of your solution's components might not scale linearly with the number of requests or the amount of data. Instead, there can be a sudden decrease in performance or increase in cost once a threshold has been met. For example, you might use a database and discover that the marginal cost of adding more capacity (scaling up) becomes prohibitive, and that scaling out is a more cost-effective strategy.
- **Separation of customers.** You might need to keep certain customers' data isolated from other customers' data. Similarly, you might have some customers that require more system resources to service than others, and consider grouping them on different sets of infrastructure.
- **Handling single- and multitenant instances.** You might have some large customers who need their own independent instances of your solution. You might

also have a pool of smaller customers who can share a multitenant deployment.

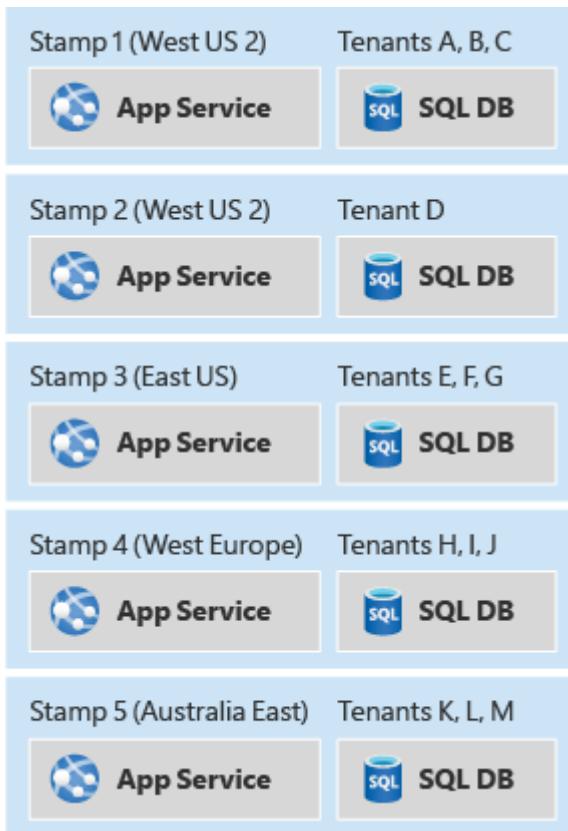
- **Complex deployment requirements.** You might need to deploy updates to your service in a controlled manner, and to deploy to different subsets of your customer base at different times.
- **Update frequency.** You might have some customers who are tolerant of having frequent updates to your system, while others might be risk-averse and want infrequent updates to the system that services their requests. It might make sense to have these customers deployed to isolated environments.
- **Geographical or geopolitical restrictions.** To architect for low latency, or to comply with data sovereignty requirements, you might deploy some of your customers into specific regions.

These limitations are often applicable to independent software vendors (ISVs) who build software as a service (SaaS), which are frequently designed to be multitenanted.

However, the same limitations can also apply to other scenarios too.

Solution

To avoid these issues, consider grouping resources in *scale units* and provisioning multiple copies of your *stamps*. Each *scale unit* will host and serve a subset of your tenants. Stamps operate independently of each other and can be deployed and updated independently. A single geographical region might contain a single stamp, or might contain multiple stamps to allow for horizontal scale-out within the region. Stamps contain a subset of your customers.



Deployment stamps can apply whether your solution uses infrastructure as a service (IaaS) or platform as a service (PaaS) components, or a mixture of both. Typically IaaS workloads require more intervention to scale, so the pattern might be useful for IaaS-heavy workloads to allow for scaling out.

Stamps can be used to implement [deployment rings](#). If different customers want to receive service updates at different frequencies, they can be grouped onto different stamps, and each stamp could have updates deployed at different cadences.

Because stamps run independently from each other, data is implicitly *sharded*. Furthermore, a single stamp can make use of further sharding to internally allow for scalability and elasticity within the stamp.

The deployment stamp pattern is used internally by many Azure services, including [App Service](#), [Azure Stack](#), and [Azure Storage](#).

Deployment stamps are related to, but distinct from, [geodes](#). In a deployment stamp architecture, multiple independent instances of your system are deployed and contain a subset of your customers and users. In geodes, all instances can serve requests from any users, but this architecture is often more complex to design and build. You might also consider mixing the two patterns within one solution; the [traffic routing approach](#) described later in this article is an example of such a hybrid scenario.

Deployment

Because of the complexity that is involved in deploying identical copies of the same components, good DevOps practices are critical to ensure success when implementing this pattern. Consider describing your infrastructure as code, such as by using [Bicep](#), [JSON Azure Resource Manager templates \(ARM templates\)](#), [Terraform](#), and scripts. With this approach, you can ensure that the deployment of each stamp is predictable and repeatable. It also reduces the likelihood of human errors such as accidental mismatches in configuration between stamps.

You can deploy updates automatically to all stamps in parallel, in which case you might consider technologies like [Bicep](#) or Resource Manager templates to coordinate the deployment of your infrastructure and applications. Alternatively, you might decide to gradually roll out updates to some stamps first, and then progressively to others. Consider using a release management tool like [Azure Pipelines](#) or [GitHub Actions](#) to orchestrate deployments to each stamp. For more information, see:

- [Integrate Bicep with Azure Pipelines](#)
- [Integrate JSON ARM templates with Azure Pipelines](#)

Carefully consider the topology of the Azure subscriptions and resource groups for your deployments:

- Typically a subscription contains all of the resources for a single solution, so in general consider using a single subscription for all stamps. However, [some Azure services impose subscription-wide quotas](#), so if you are using this pattern to allow for a high degree of scale-out, you might need to consider deploying stamps across different subscriptions.
- Resource groups are generally used to deploy components with the same lifecycle. If you plan to deploy updates to all of your stamps at once, consider using a single resource group to contain all of the components for all of your stamps, and use resource naming conventions and tags to identify the components that belong to each stamp. Alternatively, if you plan to deploy updates to each stamp independently, consider deploying each stamp into its own resource group.

Capacity planning

Use load and performance testing to determine the approximate load that a given stamp can accommodate. Load metrics might be based on the number of customers/tenants that a single stamp can accommodate, or metrics from the services that the components within the stamp emit. Ensure that you have sufficient instrumentation to measure when a given stamp is approaching its capacity, and the ability to deploy new stamps quickly to respond to demand.

Traffic routing

The Deployment Stamp pattern works well if each stamp is addressed independently. For example, if Contoso deploys the same API application across multiple stamps, they might consider using DNS to route traffic to the relevant stamp:

- `unit1.aus.myapi.contoso.com` routes traffic to stamp `unit1` within an Australian region.
- `unit2.aus.myapi.contoso.com` routes traffic to stamp `unit2` within an Australian region.
- `unit1.eu.myapi.contoso.com` routes traffic to stamp `unit1` within a European region.

Clients are then responsible for connecting to the correct stamp.

If a single ingress point for all traffic is required, a traffic routing service can be used to resolve the stamp for a given request, customer, or tenant. The traffic routing service either directs the client to the relevant URL for the stamp (for example, using an HTTP 302 response status code), or it might act as a reverse proxy and forward the traffic to the relevant stamp, without the client being aware.

A centralized traffic routing service can be a complex component to design, especially when a solution runs across multiple regions. Consider deploying the traffic routing service into multiple regions (potentially including every region that stamps are deployed into), and then ensuring the data store (mapping tenants to stamps) is synchronized. The traffic routing component might itself be an instance of the [geode pattern](#).

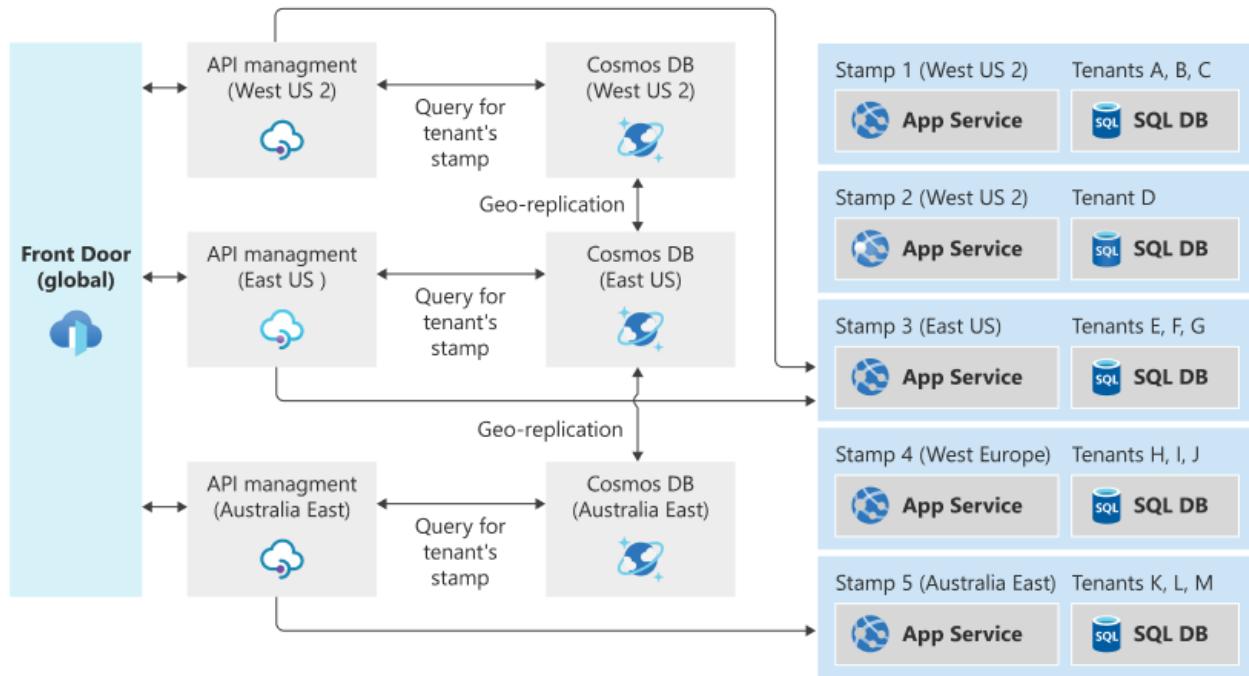
For example, [Azure API Management](#) could be deployed to act in the traffic routing service role. It can determine the appropriate stamp for a request by looking up data in an [Azure Cosmos DB](#) collection storing the mapping between tenants and stamps. API Management can then [dynamically set the back-end URL](#) to the relevant stamp's API service.

To enable geo-distribution of requests and geo-redundancy of the traffic routing service, [API Management can be deployed across multiple regions](#), or [Azure Front Door](#) can be used to direct traffic to the closest instance. Front Door can be configured with a [backend pool](#), enabling requests to be directed to the closest available API Management instance. If your application is not exposed via HTTP/S, you can use a [cross-region Azure Load Balancer](#) to distribute incoming calls to regional Azure Load Balancers. The [global distribution feature of Azure Cosmos DB](#) can be used to keep the mapping information updated across each region.

If a traffic-routing service is included in your solution, consider whether it acts as a [gateway](#) and could therefore perform [gateway offloading](#) for the other services, such as token validation, throttling, and authorization.

Example traffic routing architecture

Consider the following example traffic routing architecture, which uses Azure Front Door, Azure API Management, and Azure Cosmos DB for global traffic routing, and then a series of region-specific stamps:



Suppose a user normally resides in New York. Their data is stored in the stamp 3, in the East US region.

If the user travels to California and then accesses the system, their connection will likely be routed through the West US 2 region because that's closest to where they are geographically when they make the request. However, the request has to ultimately be served by stamp 3, because that's where their data is stored. The traffic routing system ensures that the request is routed to the correct stamp.

Issues and considerations

You should consider the following points when deciding how to implement this pattern:

- **Deployment process.** When deploying multiple stamps, it's highly advisable to have automated and fully repeatable deployment processes. Consider using [Bicep](#), [JSON ARM templates](#), or [Terraform](#) modules to declaratively define your stamps, and to keep the definitions consistent.

- **Cross-stamp operations.** When your solution is deployed independently across multiple stamps, questions like "how many customers do we have across all of our stamps?" can become more complex to answer. Queries might need to be executed against each stamp and the results aggregated. Alternatively, consider having all of the stamps publish data into a centralized data warehouse for consolidated reporting.
- **Determining scale-out policies.** Stamps have a finite capacity, which might be defined using a proxy metric such as the number of tenants that can be deployed to the stamp. It's important to monitor the available capacity and used capacity for each stamp, and to proactively deploy additional stamps to allow for new tenants to be directed to them.
- **Minimum number of stamps.** When you use the Deployment Stamp pattern, it's advisable to deploy at least two stamps of your solution. If you only deploy a single stamp, it's easy to accidentally hard-code assumptions into your code or configuration that won't apply when you scale out.
- **Cost.** The Deployment Stamp pattern involves deploying multiple copies of your infrastructure component, which will likely involve a substantial increase in the cost of operating your solution.
- **Moving between stamps.** Each stamp is deployed and operated independently, so moving tenants between stamps can be difficult. Your application would need custom logic to transmit the information about a given customer to a different stamp, and then to remove the tenant's information from the original stamp. This process might require a backplane for communication between stamps, further increasing the complexity of the overall solution.
- **Traffic routing.** As described earlier in this article, routing traffic to the correct stamp for a given request can require an additional component to resolve tenants to stamps. This component, in turn, might need to be made highly available.
- **Shared components.** You might have some components that can be shared across stamps. For example, if you have a shared single-page app for all tenants, consider deploying that into one region and using [Azure CDN](#) to replicate it globally.

When to use this pattern

This pattern is useful when you have:

- Natural limits on scalability. For example, if some components cannot or should not scale beyond a certain number of customers or requests, consider scaling out using stamps.
- A requirement to separate certain tenants from others. If you have customers that cannot be deployed into a multitenant stamp with other customers due to security concerns, they can be deployed onto their own isolated stamp.

- A need to have some tenants on different versions of your solution at the same time.
- Multi-region applications where each tenant's data and traffic should be directed to a specific region.
- A desire to achieve resiliency during outages. As stamps are independent of one another, if an outage affects a single stamp then the tenants deployed to other stamps should not be affected. This isolation helps to contain the 'blast radius' of an incident or outage.

This pattern is not suitable for:

- Simple solutions that do not need to scale to a high degree.
- Systems that can be easily scaled out or up within a single instance, such as by increasing the size of the application layer or by increasing the reserved capacity for databases and the storage tier.
- Solutions in which data should be replicated across all deployed instances. Consider the [geode pattern](#) for this scenario.
- Solutions in which only some components need to be scaled, but not others. For example, consider whether your solution could be scaled by [sharding the data store](#) rather than deploying a new copy of all of the solution components.
- Solutions comprised solely of static content, such as a front-end JavaScript application. Consider storing such content in a [storage account](#) and using [Azure CDN](#).

Supporting technologies

- Infrastructure as code. For example, Bicep, Resource Manager templates, Azure CLI, Terraform, PowerShell, Bash.
- [Azure Front Door](#), which can route traffic to a specific stamp or to a traffic routing service.

Example

The following example deploys multiple stamps of a simple PaaS solution, with an app service and a SQL Database in each stamp. While stamps can be configured in any region that support the services deployed in the template, for illustration purposes this template deploys two stamps within the West US 2 region and a further stamp in the West Europe region. Within a stamp, the app service receives its own public DNS hostname and it can receive connections independently of all other stamps.

⚠️ Warning

The example below uses a SQL Server administrator account. It's generally not a good practice to use an administrative account from your application. For a real application, consider **using a managed identity to connect from your application to a SQL database**, or use a least-privilege account.

Click the link below to deploy the solution.

 [Deploy to Azure](#) 

ⓘ Note

There are alternative approaches to deploying stamps with a Resource Manager template, including using **nested templates** or **linked templates** to decouple the definition of each stamp from the iteration required to deploy multiple copies.

Example traffic routing approach

The following example deploys an implementation of a traffic routing solution that could be used with a set of deployment stamps for a hypothetical API application. To allow for geographical distribution of incoming requests, Front Door is deployed alongside multiple instances of API Management on the consumption tier. Each API Management instance reads the tenant ID from the request URL and then looks up the relevant stamp for the request from a geo-distributed Azure Cosmos DB data store. The request is then forwarded to the relevant back-end stamp.

Click the link below to deploy the solution.

 [Deploy to Azure](#) 

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [John Downs](#)  | Principal Program Manager

Other contributors:

- [Daniel Larsen](#) | Principal Customer Engineer, FastTrack for Azure
- [Angel Lopez](#) | Senior Software Engineer, Azure Patterns and Practices
- [Paolo Salvatori](#) | Principal Customer Engineer, FastTrack for Azure
- [Arsen Vladimirs](#) | Principal Customer Engineer, FastTrack for Azure

To see non-public LinkedIn profiles, sign in to LinkedIn.

Related resources

- Sharding can be used as another, simpler, approach to scale out your data tier. Stamps implicitly shard their data, but sharding does not require a Deployment Stamp. For more information, see the [Sharding pattern](#).
- If a traffic routing service is deployed, the [Gateway Routing](#) and [Gateway Offloading](#) patterns can be used together to make the best use of this component.

Edge Workload Configuration pattern

Article • 11/21/2022

The great variety of systems and devices on the shop floor can make workload configuration a difficult problem. This article provides approaches to solving it.

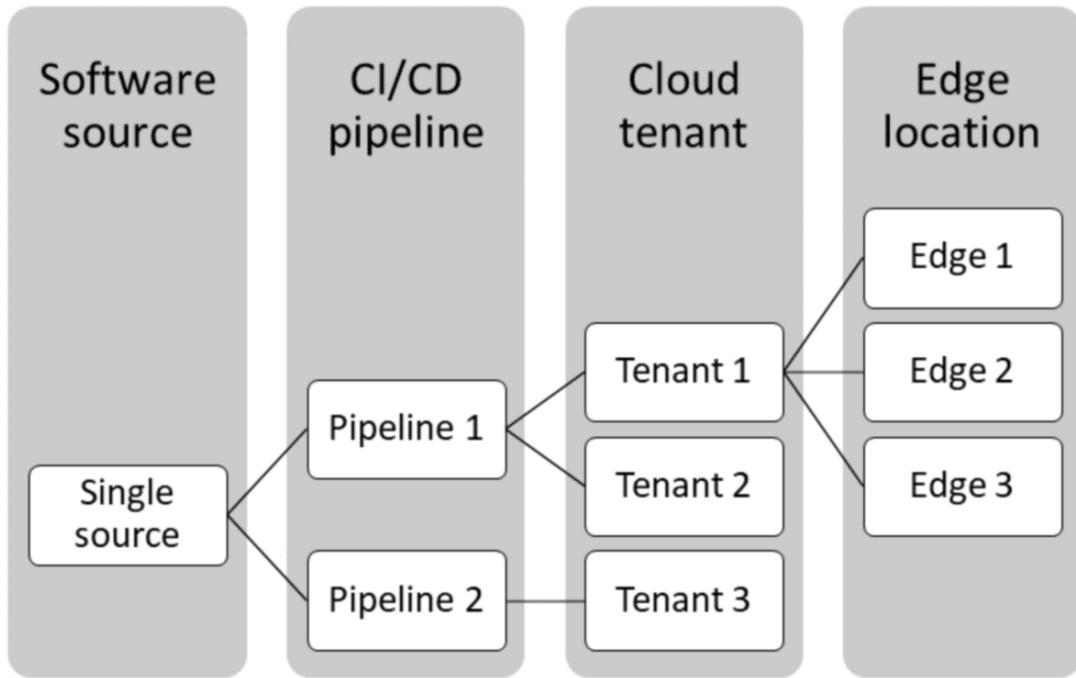
Context and problem

Manufacturing companies, as part of their digital transformation journey, focus increasingly on building software solutions that can be reused as shared capabilities. Due to the variety of devices and systems on the shop floor, the modular workloads are configured to support different protocols, drivers, and data formats. Sometimes even multiple instances of a workload are run with different configurations in the same edge location. For some workloads, the configurations are updated more than once a day. Therefore, configuration management is increasingly important to the scaling out of edge solutions.

Solution

There are a few common characteristics of configuration management for edge workloads:

- There are several configuration points that can be grouped into distinct layers, like software source, CI/CD pipeline, cloud tenant, and edge location:



- The various layers can be updated by different people.
- No matter how the configurations are updated, they need to be carefully tracked and audited.
- For business continuity, it's required that configurations can be accessed offline at the edge.
- It's also required that there's a global view of configurations that's available on the cloud.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- Allowing edits when the edge isn't connected to the cloud significantly increases the complexity of configuration management. It's possible to replicate changes to the cloud, but there are challenges with:
 - User authentication, because it relies on a cloud service like Azure Active Directory.
 - Conflict resolution after reconnection, if workloads receive unexpected configurations that require manual intervention.
- The edge environment can have network-related constraints if the topology complies to the ISA-95 requirements. You can overcome such restraints by selecting a technology that offers connectivity across layers, such as [device hierarchies in Azure IoT Edge](#).
- If run-time configuration is decoupled from software releases, configuration changes need to be handled separately. To offer history and rollback features, you

need to store past configurations in a datastore in the cloud.

- A fault in a configuration, like a connectivity component configured to a nonexistent end-point, can break the workload. Therefore, it's important to treat configuration changes as you treat other deployment lifecycle events in the observability solution, so that the observability dashboards can help correlate system errors to configuration changes. For more information about observability, see [Cloud monitoring guide: Observability](#).
- Understand the roles that the cloud and edge datastores play in business continuity. If the cloud datastore is the single source of truth, then the edge workloads should be able to restore intended states by using automated processes.
- For resiliency, the edge datastore should act as an offline cache. This takes precedence over latency considerations.

When to use this pattern

Use this pattern when:

- There's a requirement to configure workloads outside of the software release cycle.
- Different people need to be able to read and update configurations.
- Configurations need to be available even if there's no connection to the cloud.

Example workloads:

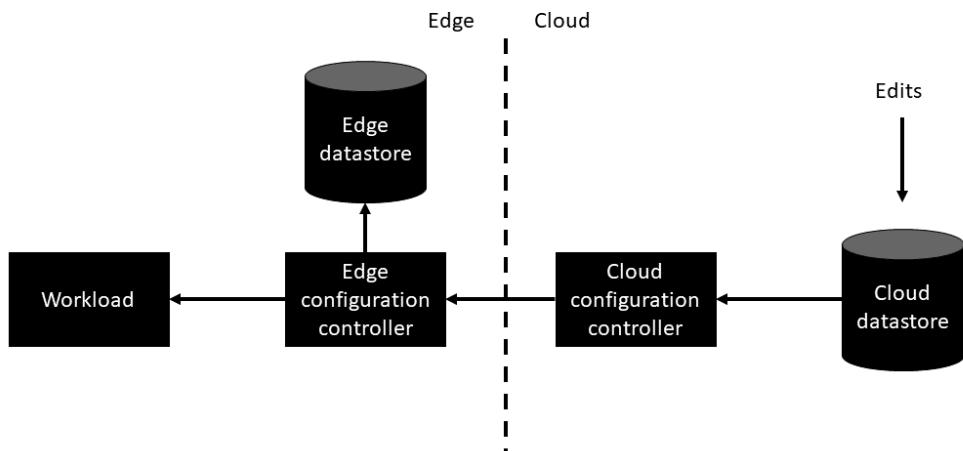
- Solutions that connect to assets on the shop floor for data ingestion—OPC Publisher, for example—and command and control
- Machine learning workloads for predictive maintenance
- Machine learning workloads that inspect in real-time for quality on the manufacturing line

Examples

The solution to configure edge workloads during run-time can be based on an external configuration controller or an internal configuration provider.

External configuration controller variation

External configuration controller



This variation has a configuration controller that's external to the workload. The role of the cloud configuration controller component is to push edits from the cloud datastore to the workload through the edge configuration controller. The edge also contains a datastore so that the system functions even when disconnected from the cloud.

With IoT Edge, the edge configuration controller can be implemented as a module, and the configurations can be applied with [module twins](#). The module twin has a size limit; if the configuration exceeds the limit, the solution can be [extended with Azure Blob Storage](#) or by chunking larger payloads over [direct methods](#).

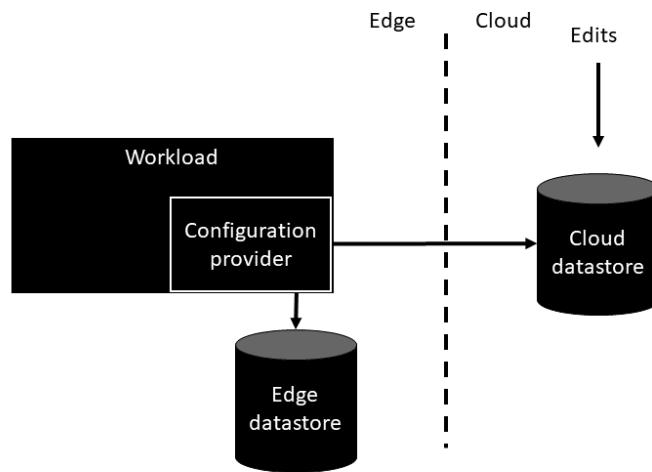
For an end-to-end example of the external configuration controller variation, see the [Connected factory signal pipeline](#).

The benefits of this variation are:

- The workload itself doesn't have to be aware of the configuration system. This capability is a requirement if the source code of the workload is not editable—for example, when using a module from the [Azure IoT Edge Marketplace](#).
- It's possible to change the configuration of multiple workloads at the same time by coordinating the changes via the cloud configuration controller.
- Additional validation can be implemented as part of the push pipeline—for example, to validate existence of endpoints at the edge before pushing the configuration to the workload.

Internal configuration provider variation

Internal configuration provider



In the internal configuration provider variation, the workload pulls configurations from a configuration provider. For an implementation example, see [Implement a custom configuration provider in .NET](#). That example uses C#, but other languages can be used.

In this variation, the workloads have unique identifiers so that the same source code running in different environments can have different configurations. One way to construct an identifier is to concatenate the hierarchical relationship of the workload to a top-level grouping such as a tenant. For IoT Edge, it could be a combination of Azure resource group, IoT hub name, IoT Edge device name, and module identifier. These values together form a unique identifier that work as a key in the datastores.

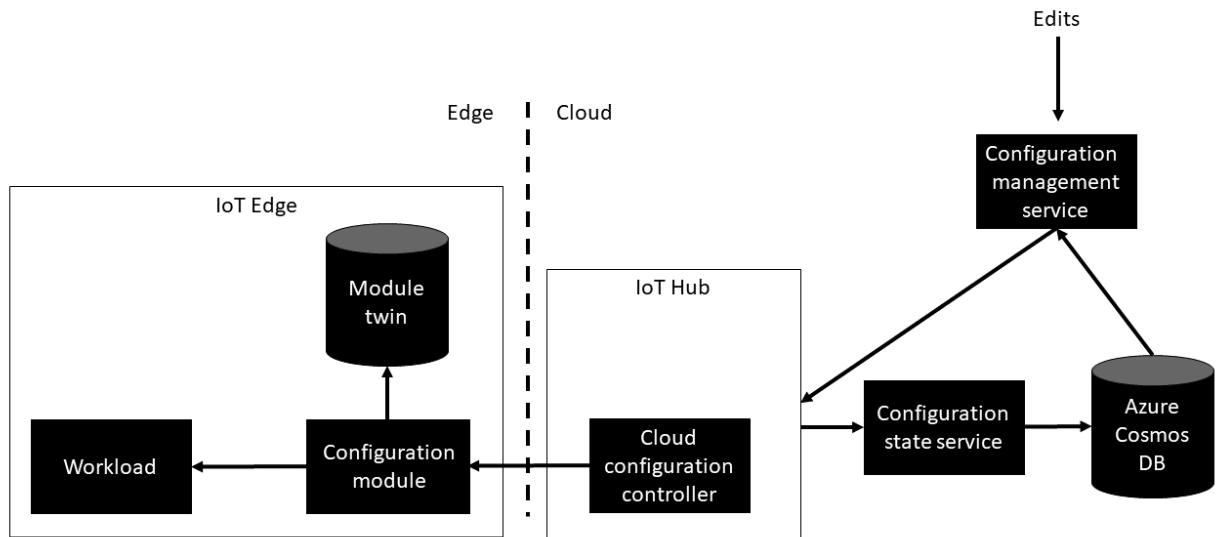
Although the module version can be added to the unique identifier, it's a common requirement to persist configurations across software updates. If the version is a part of the identifier, the old version of the configuration should be migrated forward with an additional implementation.

The benefits of this variation are:

- Other than the datastores, the solution doesn't require components, reducing complexity.
- Migration logic from incompatible old versions can be handled within the workload implementation.

Solutions based on IoT Edge

Solution architecture based on IoT Edge



The cloud component of the IoT Edge reference implementation consists of an IoT hub acting as the cloud configuration controller. The [Azure IoT Hub](#) module twin functionality propagates configuration changes and information about the currently applied configuration by using module twin desired and reported properties. The configuration management service acts as the source of the configurations. It can also be a user interface for managing configurations, a build system, and other tools used to author workload configurations.

An [Azure Cosmos DB](#) database stores all configurations. It can interact with multiple IoT hubs, and provides configuration history.

After an edge device indicates via reported properties that a configuration was applied, the configuration state service notes the event in the database instance.

When a new configuration is created in the configuration management service, it is stored in Azure Cosmos DB and the desired properties of the edge module are changed in the IoT hub where the device resides. The configuration is then transmitted by IoT Hub to the edge device. The edge module is expected to apply the configuration and report via the module twin the state of the configuration. The configuration state service then listens to twin change events, and upon detecting that a module reports a configuration state change, records the corresponding change in the Azure Cosmos DB database.

The edge component can use either the external configuration controller or internal configuration provider. In either implementation, the configuration is either transmitted in the module twin desired properties, or in case large configurations need to be transmitted, the module twin desired properties contain a URL to [Azure Blob Storage](#)

or to another service that can be used to retrieve the configuration. The module then signals in the module twin reported properties whether the new configuration was applied successfully and what configuration is currently applied.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Heather Camm](#) ↗ | Senior Program Manager

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- [Azure IoT Edge](#) ↗
- [What is Azure IoT Edge?](#)
- [Azure IoT Hub](#) ↗
- [IoT Concepts and Azure IoT Hub](#)
- [Azure Cosmos DB](#) ↗
- [Welcome to Azure Cosmos DB](#)
- [Azure Blob Storage](#) ↗
- [Introduction to Azure Blob storage](#)

Related resources

- [External Configuration Store pattern](#)

Event Sourcing pattern

Bookings

Instead of storing just the current state of the data in a domain, use an append-only store to record the full series of actions taken on that data. The store acts as the system of record and can be used to materialize the domain objects. This can simplify tasks in complex domains, by avoiding the need to synchronize the data model and the business domain, while improving performance, scalability, and responsiveness. It can also provide consistency for transactional data, and maintain full audit trails and history that can enable compensating actions.

Context and problem

Most applications work with data, and the typical approach is for the application to maintain the current state of the data by updating it as users work with it. For example, in the traditional create, read, update, and delete (CRUD) model a typical data process is to read data from the store, make some modifications to it, and update the current state of the data with the new values—often by using transactions that lock the data.

The CRUD approach has some limitations:

- CRUD systems perform update operations directly against a data store. These operations can slow down performance and responsiveness and can limit scalability, due to the processing overhead it requires.
- In a collaborative domain with many concurrent users, data update conflicts are more likely because the update operations take place on a single item of data.
- Unless there's another auditing mechanism that records the details of each operation in a separate log, history is lost.

Solution

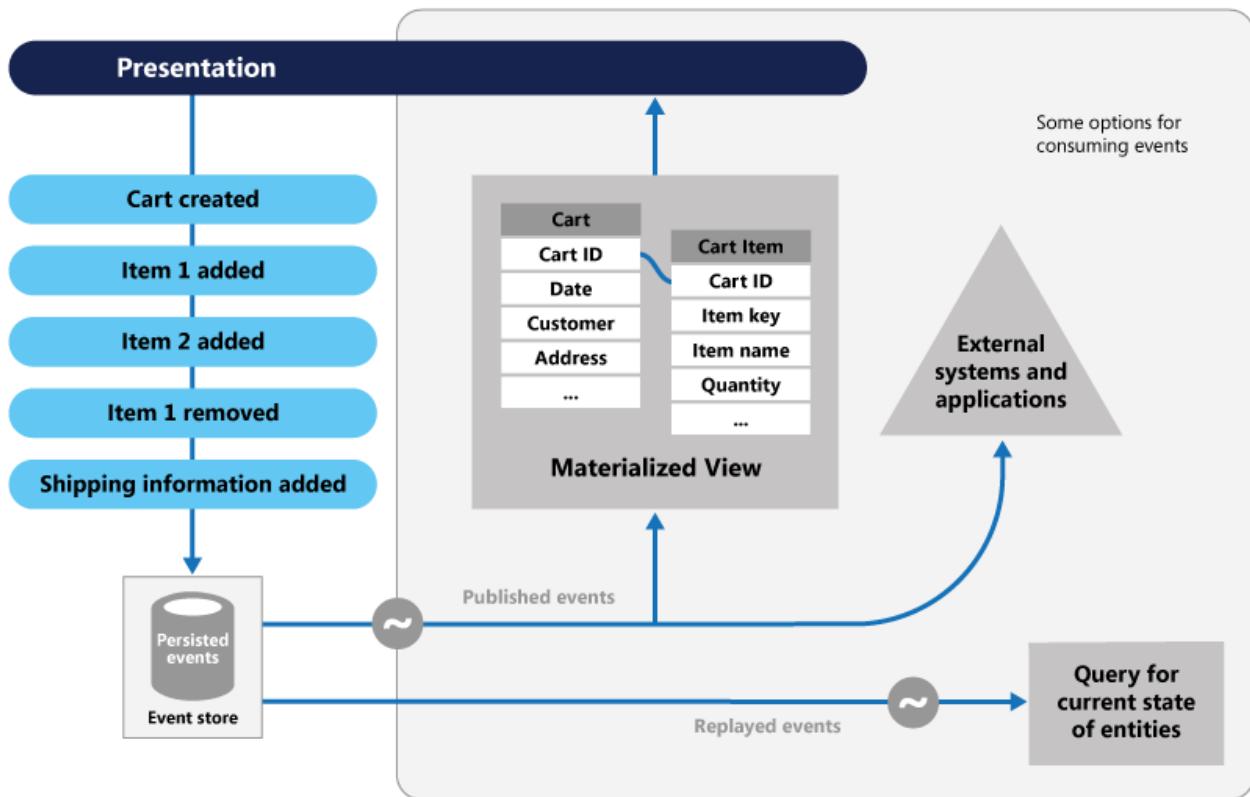
The Event Sourcing pattern defines an approach to handling operations on data that's driven by a sequence of events, each of which is recorded in an append-only store. Application code sends a series of events that imperatively describe each action that has occurred on the data to the event store, where they're persisted. Each event represents a set of changes to the data (such as `AddedItemToOrder`).

The events are persisted in an event store that acts as the system of record (the authoritative data source) about the current state of the data. The event store typically publishes these events so that consumers can be notified and can handle them if needed. Consumers could, for example, initiate tasks that apply the operations in the events to other systems, or perform any other associated action that's required to complete the operation. Notice that the application code that generates the events is decoupled from the systems that subscribe to the events.

Typical uses of the events published by the event store are to maintain materialized views of entities as actions in the application change them, and for integration with external systems. For example, a system can maintain a materialized view of all customer orders that's used to populate parts of the UI. The application adds new orders, adds or removes items on the order, and adds shipping information. The events that describe these changes can be handled and used to update the [materialized view](#).

At any point, it's possible for applications to read the history of events. You can then use it to materialize the current state of an entity by playing back and consuming all the events that are related to that entity. This process can occur on demand to materialize a domain object when handling a request. Or, the process occurs through a scheduled task so that the state of the entity can be stored as a materialized view, to support the presentation layer.

The figure shows an overview of the pattern, including some of the options for using the event stream such as creating a materialized view, integrating events with external applications and systems, and replaying events to create projections of the current state of specific entities.



The Event Sourcing pattern provides the following advantages:

- Events are immutable and can be stored using an append-only operation. The user interface, workflow, or process that initiated an event can continue, and tasks that handle the events can run in the background. This process, combined with the fact that there's no contention during the processing of transactions, can vastly improve performance and scalability for applications, especially for the presentation level or user interface.
- Events are simple objects that describe some action that occurred, together with any associated data that's required to describe the action represented by the event. Events don't directly update a data store. They're simply recorded for handling at the appropriate time. Using events can simplify implementation and management.
- Events typically have meaning for a domain expert, whereas object-relational impedance mismatch can make complex database tables hard to understand. Tables are artificial constructs that represent the current state of the system, not the events that occurred.
- Event sourcing can help prevent concurrent updates from causing conflicts because it avoids the requirement to directly update objects in the data store. However, the domain model must still be designed to protect itself from requests that might result in an inconsistent state.

- The append-only storage of events provides an audit trail that can be used to monitor actions taken against a data store. It can regenerate the current state as materialized views or projections by replaying the events at any time, and it can assist in testing and debugging the system. In addition, the requirement to use compensating events to cancel changes can provide a history of changes that were reversed. This capability wouldn't be the case if the model stored the current state. The list of events can also be used to analyze application performance and to detect user behavior trends. Or, it can be used to obtain other useful business information.
- The event store raises events, and tasks perform operations in response to those events. This decoupling of the tasks from the events provides flexibility and extensibility. Tasks know about the type of event and the event data, but not about the operation that triggered the event. In addition, multiple tasks can handle each event. This enables easy integration with other services and systems that only listen for new events raised by the event store. However, the event sourcing events tend to be very low level, and it might be necessary to generate specific integration events instead.

Event sourcing is commonly combined with the CQRS pattern by performing the data management tasks in response to the events, and by materializing views from the stored events.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

The system will only be eventually consistent when creating materialized views or generating projections of data by replaying events. There's some delay between an application adding events to the event store as the result of handling a request, the events being published, and the consumers of the events handling them. During this period, new events that describe further changes to entities might have arrived at the event store. The system should be designed to account for eventual consistency in these scenarios.

ⓘ Note

See the [Data Consistency Primer](#) for information about eventual consistency.

The event store is the permanent source of information, and so the event data should never be updated. The only way to update an entity to undo a change is to add a

compensating event to the event store. If the format (rather than the data) of the persisted events needs to change, perhaps during a migration, it can be difficult to combine existing events in the store with the new version. It might be necessary to iterate through all the events making changes so they're compliant with the new format, or add new events that use the new format. Consider using a version stamp on each version of the event schema to maintain both the old and the new event formats.

Multi-threaded applications and multiple instances of applications might be storing events in the event store. The consistency of events in the event store is vital, as is the order of events that affect a specific entity (the order that changes occur to an entity affects its current state). Adding a timestamp to every event can help to avoid issues. Another common practice is to annotate each event resulting from a request with an incremental identifier. If two actions attempt to add events for the same entity at the same time, the event store can reject an event that matches an existing entity identifier and event identifier.

There's no standard approach, or existing mechanisms such as SQL queries, for reading the events to obtain information. The only data that can be extracted is a stream of events using an event identifier as the criteria. The event ID typically maps to individual entities. The current state of an entity can be determined only by replaying all of the events that relate to it against the original state of that entity.

The length of each event stream affects managing and updating the system. If the streams are large, consider creating snapshots at specific intervals such as a specified number of events. The current state of the entity can be obtained from the snapshot and by replaying any events that occurred after that point in time. For more information about creating snapshots of data, see [Primary-Subordinate Snapshot Replication](#).

Even though event sourcing minimizes the chance of conflicting updates to the data, the application must still be able to deal with inconsistencies that result from eventual consistency and the lack of transactions. For example, an event that indicates a reduction in stock inventory might arrive in the data store while an order for that item is being placed. This situation results in a requirement to reconcile the two operations, either by advising the customer or by creating a back order.

Event publication might be *at least once*, and so consumers of the events must be idempotent. They must not reapply the update described in an event if the event is handled more than once. Multiple instances of a consumer can maintain and aggregate an entity's property, such as the total number of orders placed. Only one must succeed in incrementing the aggregate, when an order-placed event occurs. While this result isn't a key characteristic of event sourcing, it's the usual implementation decision.

The event storage selected needs to support the event load generated by your application.

Be mindful of scenarios where the processing of one event involves the creation of one or more new events since this can cause an infinite loop.

When to use this pattern

Use this pattern in the following scenarios:

- When you want to capture intent, purpose, or reason in the data. For example, changes to a customer entity can be captured as a series of specific event types, such as *Moved home*, *Closed account*, or *Deceased*.
- When it's vital to minimize or completely avoid the occurrence of conflicting updates to data.
- When you want to record events that occur, to replay them to restore the state of a system, to roll back changes, or to keep a history and audit log. For example, when a task involves multiple steps, you might need to execute actions to revert updates and then replay some steps to bring the data back into a consistent state.
- When you use events. It's a natural feature of the operation of the application, and it requires little extra development or implementation effort.
- When you need to decouple the process of inputting, or updating data from the tasks required to apply these actions. This change might be to improve UI performance, or to distribute events to other listeners that take action when the events occur. For example, you can integrate a payroll system with an expense submission website. The events that are raised by the event store in response to data updates made in the website would be consumed by both the website and the payroll system.
- When you want flexibility to be able to change the format of materialized models and entity data if requirements change, or—when used with CQRS—you need to adapt a read model or the views that expose the data.
- When used with CQRS, and eventual consistency is acceptable while a read model is updated, or the performance impact of rehydrating entities and data from an event stream is acceptable.

This pattern might not be useful in the following situations:

- Small or simple domains, systems that have little or no business logic, or nondomain systems that naturally work well with traditional CRUD data

management mechanisms.

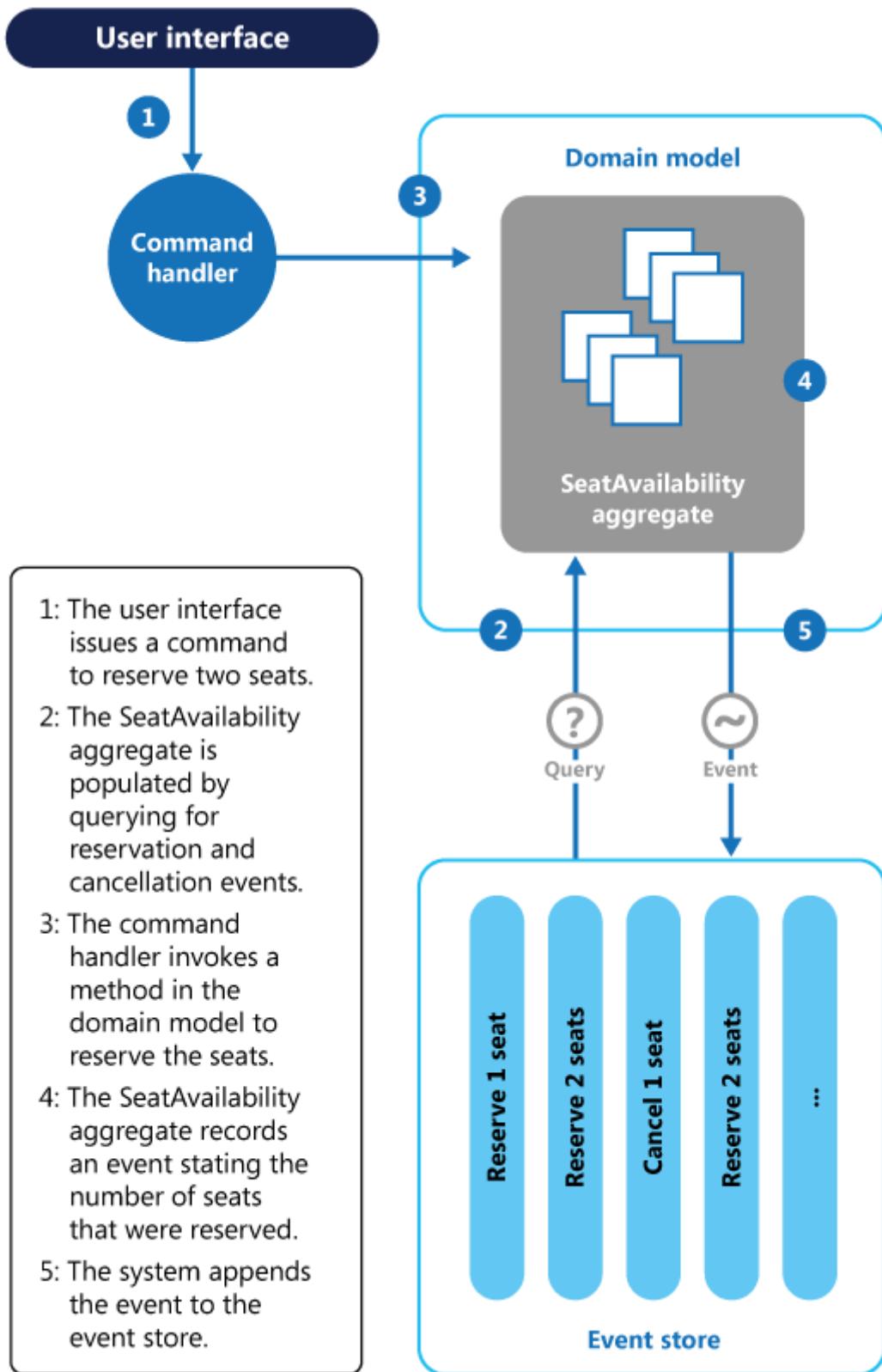
- Systems where consistency and real-time updates to the views of the data are required.
- Systems where audit trails, history, and capabilities to roll back and replay actions aren't required.
- Systems where there's only a low occurrence of conflicting updates to the underlying data. For example, systems that predominantly add data rather than updating it.

Example

A conference management system needs to track the number of completed bookings for a conference. This way it can check whether there are seats still available, when a potential attendee tries to make a booking. The system could store the total number of bookings for a conference in at least two ways:

- The system could store the information about the total number of bookings as a separate entity in a database that holds booking information. As bookings are made or canceled, the system could increment or decrement this number as appropriate. This approach is simple in theory, but can cause scalability issues if a large number of attendees are attempting to book seats during a short period of time. For example, in the last day or so prior to the booking period closing.
- The system could store information about bookings and cancellations as events held in an event store. It could then calculate the number of seats available by replaying these events. This approach can be more scalable due to the immutability of events. The system only needs to be able to read data from the event store, or append data to the event store. Event information about bookings and cancellations is never modified.

The following diagram illustrates how the seat reservation subsystem of the conference management system might be implemented using event sourcing.



The sequence of actions for reserving two seats is as follows:

1. The user interface issues a command to reserve seats for two attendees. The command is handled by a separate command handler. A piece of logic that is decoupled from the user interface and is responsible for handling requests posted as commands.
2. An aggregate containing information about all reservations for the conference is constructed by querying the events that describe bookings and cancellations. This

aggregate is called `SeatAvailability`, and is contained within a domain model that exposes methods for querying and modifying the data in the aggregate.

Some optimizations to consider are using snapshots (so that you don't need to query and replay the full list of events to obtain the current state of the aggregate), and maintaining a cached copy of the aggregate in memory.

3. The command handler invokes a method exposed by the domain model to make the reservations.
4. The `SeatAvailability` aggregate records an event containing the number of seats that were reserved. The next time the aggregate applies events, all the reservations will be used to compute how many seats remain.
5. The system appends the new event to the list of events in the event store.

If a user cancels a seat, the system follows a similar process except the command handler issues a command that generates a seat cancellation event and appends it to the event store.

In addition to providing more scope for scalability, using an event store also provides a complete history, or audit trail, of the bookings and cancellations for a conference. The events in the event store are the accurate record. There's no need to persist aggregates in any other way because the system can easily replay the events and restore the state to any point in time.

You can find more information about this example in [Introducing Event Sourcing](#).

Next steps

- [Object-relational impedance mismatch](#)
- [Data Consistency Primer](#). When you use event sourcing with a separate read store or materialized views, the read data won't be immediately consistent. Instead, the data will be only eventually consistent. This article summarizes the issues surrounding maintaining consistency over distributed data.
- [Data Partitioning Guidance](#). Data is often partitioned when you use event sourcing to improve scalability, reduce contention, and optimize performance. This article describes how to divide data into discrete partitions, and the issues that can arise.
- Martin Fowler's blog:
 - [Event Sourcing](#)

- [Snapshot on Martin Fowler's Enterprise Application Architecture website ↗](#)

Related resources

The following patterns and guidance might also be relevant when implementing this pattern:

- [Command and Query Responsibility Segregation \(CQRS\) pattern](#). The write store that provides the permanent source of information for a CQRS implementation is often based on an implementation of the Event Sourcing pattern. Describes how to segregate the operations that read data in an application from the operations that update data by using separate interfaces.
- [Materialized View pattern](#). The data store used in a system that's based on event sourcing is typically not well suited to efficient querying. Instead, a common approach is to generate prepopulated views of the data at regular intervals, or when the data changes.
- [Compensating Transaction pattern](#). The existing data in an event sourcing store isn't updated. Instead, new entries are added that transition the state of entities to the new values. To reverse a change, compensating entries are used because it isn't possible to reverse the previous change. Describes how to undo the work that was performed by a previous operation.

External Configuration Store pattern

Azure App Configuration Azure Blob Storage

Move configuration information out of the application deployment package to a centralized location. This can provide opportunities for easier management and control of configuration data, and for sharing configuration data across applications and application instances.

Context and problem

The majority of application runtime environments include configuration information that's held in files deployed with the application. In some cases, it's possible to edit these files to change the application behavior after it's been deployed. However, changes to the configuration require the application be redeployed, often resulting in unacceptable downtime and other administrative overhead.

Local configuration files also limit the configuration to a single application, but sometimes it would be useful to share configuration settings across multiple applications. Examples include database connection strings, UI theme information, or the URLs of queues and storage used by a related set of applications.

It's challenging to manage changes to local configurations across multiple running instances of the application, especially in a cloud-hosted scenario. It can result in instances using different configuration settings while the update is being deployed.

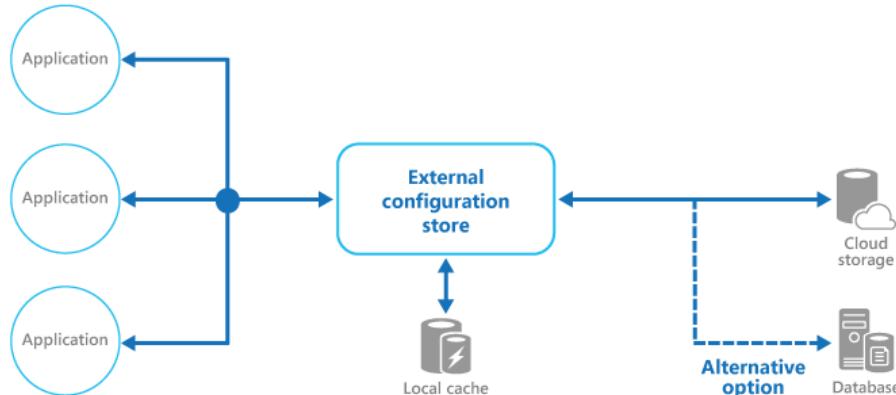
In addition, updates to applications and components might require changes to configuration schemas. Many configuration systems don't support different versions of configuration information.

Solution

Store the configuration information in external storage, and provide an interface that can be used to quickly and efficiently read and update configuration settings. The type of external store depends on the hosting and runtime environment of the application. In a cloud-hosted scenario it's typically a cloud-based storage service or dedicated configuration service, but could be a hosted database or other custom system.

The backing store you choose for configuration information should have an interface that provides consistent and easy-to-use access. It should expose the information in a correctly typed and structured format. The implementation might also need to authorize users' access in order to protect configuration data, and be flexible enough to allow storage of multiple versions of the configuration (such as development, staging, or production, including multiple release versions of each one).

Many built-in configuration systems read the data when the application starts up, and cache the data in memory to provide fast access and minimize the impact on application performance. Depending on the type of backing store used, and the latency of this store, it might be helpful to implement a caching mechanism within the external configuration store. For more information, see the [Caching Guidance](#). The figure illustrates an overview of the External Configuration Store pattern with optional local cache.



Issues and considerations

Consider the following points when deciding how to implement this pattern:

Choose a backing store that offers acceptable performance, high availability, robustness, and can be backed up as part of the application maintenance and administration process. In a cloud-hosted application, using a cloud storage mechanism or dedicated configuration platform service is usually a good choice to meet these requirements.

Design the schema of the backing store to allow flexibility in the types of information it can hold. Ensure that it provides for all configuration requirements such as typed data, collections of settings, multiple versions of settings, and any other features that the applications using it require. The schema should be easy to extend to support additional settings as requirements change.

Consider the physical capabilities of the backing store, how it relates to the way configuration information is stored, and the effects on performance. For example, storing an XML document containing configuration information will require either the configuration interface or the application to parse the document in order to read individual settings. It'll make updating a setting more complicated, though caching the settings can help to offset slower read performance.

Consider how the configuration interface will permit control of the scope and inheritance of configuration settings. For example, it might be a requirement to scope configuration settings at the organization, application, and the machine level. It might need to support delegation of control over access to different scopes, and to prevent or allow individual applications to override settings.

Ensure that the configuration interface can expose the configuration data in the required formats such as typed values, collections, key/value pairs, or property bags.

Consider how the configuration store interface will behave when settings contain errors, or don't exist in the backing store. It might be appropriate to return default settings and log errors. Also consider aspects such as the case sensitivity of configuration setting keys or names, the storage and handling of binary data, and the ways that null or empty values are handled.

Consider how to protect the configuration data to allow access to only the appropriate users and applications. This is likely a feature of the configuration store interface, but it's also necessary to ensure that the data in the backing store can't be accessed directly without the appropriate permission. Ensure strict separation between the permissions required to read and to write configuration data. Also consider whether you need to encrypt some or all of the configuration settings, and how this'll be implemented in the configuration store interface.

Centrally stored configurations, which change application behavior during runtime, are critically important and should be deployed, updated, and managed using the same mechanisms as deploying application code. For example, changes that can affect more than one application must be carried out using a full test and staged deployment approach to ensure that the change is appropriate for all applications that use this configuration. If an administrator edits a setting to update one application, it could adversely impact other applications that use the same setting.

If an application caches configuration information, the application needs to be alerted if the configuration changes. It might be possible to implement an expiration policy over cached configuration data so that this information is automatically refreshed periodically and any changes picked up (and acted on).

While caching configuration data can help address transient connectivity issues with the external configuration store at application runtime, this typically doesn't solve the problem if the external store is down when the application is first starting. Ensure your application deployment pipeline can provide the last known set of configuration values in a configuration file as a fallback if your application cannot retrieve live values when it starts.

When to use this pattern

This pattern is useful for:

- Configuration settings that are shared between multiple applications and application instances, or where a standard configuration must be enforced across multiple applications and application instances.
- A standard configuration system that doesn't support all of the required configuration settings, such as storing images or complex data types.
- As a complementary store for some of the settings for applications, perhaps allowing applications to override some or all of the centrally-stored settings.
- As a way to simplify administration of multiple applications, and optionally for monitoring use of configuration settings by logging some or all types of access to the configuration store.

Custom backing store example

In a Microsoft Azure hosted application, a possible choice for storing configuration information externally is to use Azure Storage. This is resilient, offers high performance, and is replicated three times with automatic failover to offer high availability. Azure Table storage provides a key/value store with the ability to use a flexible schema for the values. Azure Blob storage provides a hierarchical, container-based store that can hold any type of data in individually named blobs.

When implementing this pattern you'd be responsible for abstracting away Azure Blob storage and exposing your settings within your applications, including checking for updates at runtime and addressing how to respond to those.

The following example shows how a simplistic configuration store could be envisioned over Blob storage to store and expose configuration information. A `BlobSettingsStore` class could abstract Blob storage for holding configuration information, and implements a simple `ISettingsStore` interface.

```
C#  
  
public interface ISettingsStore  
{  
    Task<ETag> GetVersionAsync();  
    Task<Dictionary<string, string>> FindAllAsync();  
}
```

This interface defines methods for retrieving configuration settings held in the configuration store and includes a version number that can be used to detect whether any configuration settings have been modified recently. A `BlobSettingsStore` class could use the `ETag` property of the blob to implement versioning. The `ETag` property is updated automatically each time a blob is written.

By design, this simple illustration exposes all configuration settings as string values rather than typed values.

An `ExternalConfigurationManager` class could then provide a wrapper around a `BlobSettingsStore` instance. An application can use this class to retrieve configuration information. This class might use something like [Microsoft Reactive Extensions](#) to publish any changes made to the configuration while the system is running. It would also be responsible for implementing the [Cache-Aside pattern](#) for settings to provide added resiliency and performance.

Usage might look something like the following.

```
C#  
  
static void Main(string[] args)  
{  
    // Start monitoring configuration changes.  
    ExternalConfiguration.Instance.StartMonitor();  
  
    // Get a setting.  
    var setting = ExternalConfiguration.Instance.GetAppSetting("someSettingKey");  
    ...  
}
```

Using Azure App Configuration

While building a custom configuration store might be necessary in some situations, many applications can instead use [Azure App Configuration](#). Azure App Configuration supports [key-value pairs](#) that can be namespaced. The keys are typed and are individually versioned. Azure App Configuration also supports [point-in-time snapshots](#) of configuration so that you can easily inspect or even roll back to prior configuration values. Configuration values can be exported such that a copy of the configuration can ship with your application in case the service is unreachable when the application is starting.

Client libraries

Many of these features are exposed through client libraries which integrate with the application runtime to facilitate fetching and caching values, refreshing values on change, and even handling transient outages of App Configuration Service.

[Expand table](#)

Runtime	Client Library	Notes	Quickstart
.NET	Microsoft.Extensions.Configuration.AzureAppConfiguration	Provider for <code>Microsoft.Extensions.Configuration</code>	Quickstart
ASP.NET	Microsoft.Azure.AppConfiguration.AspNetCore	Provider for <code>Microsoft.Extensions.Configuration</code>	Quickstart

Runtime	Client Library	Notes	Quickstart
Azure Functions in .NET	Microsoft.Extensions.Configuration.AzureAppConfiguration	Used with Azure Function extensions to support configuration in <i>Startup.cs</i>	Quickstart
.NET Framework	Microsoft.Configuration.ConfigurationBuilders.AzureAppConfiguration	Configuration builder for <code>System.Configuration</code>	Quickstart
Java Spring	com.azure.spring > azure-spring-cloud-appconfiguration-config	Supports Spring Framework access via <code>ConfigurationProperties</code>	Quickstart
Python	azure.appconfiguration	Provides an <code>AzureAppConfigurationClient</code>	Quickstart
JavaScript/Node.js	@azure/app-configuration	Provides an <code>AppConfigurationClient</code>	Quickstart

In addition to client libraries, there are also an [Azure App Configuration Sync](#) GitHub Action and [Azure App Configuration Pull](#) & [Azure App Configuration Push](#) Azure DevOps tasks to integrate configuration steps into your build process.

Next steps

- See additional [App Configuration Samples](#)
- Learn how to [integrate Azure App Configuration with Kubernetes deployments](#)
- Learn how Azure App Configuration also can help [manage feature flags](#)

Federated Identity pattern

Microsoft Entra ID

Delegate authentication to an external identity provider. This can simplify development, minimize the requirement for user administration, and improve the user experience of the application.

Context and problem

Users typically need to work with multiple applications provided and hosted by different organizations they have a business relationship with. These users might be required to use specific (and different) credentials for each one. This can:

- **Cause a disjointed user experience.** Users often forget sign-in credentials when they have many different ones.
- **Expose security vulnerabilities.** When a user leaves the company the account must immediately be deprovisioned. It's easy to overlook this in large organizations.
- **Complicate user management.** Administrators must manage credentials for all of the users, and perform additional tasks such as providing password reminders.

Users typically prefer to use the same credentials for all these applications.

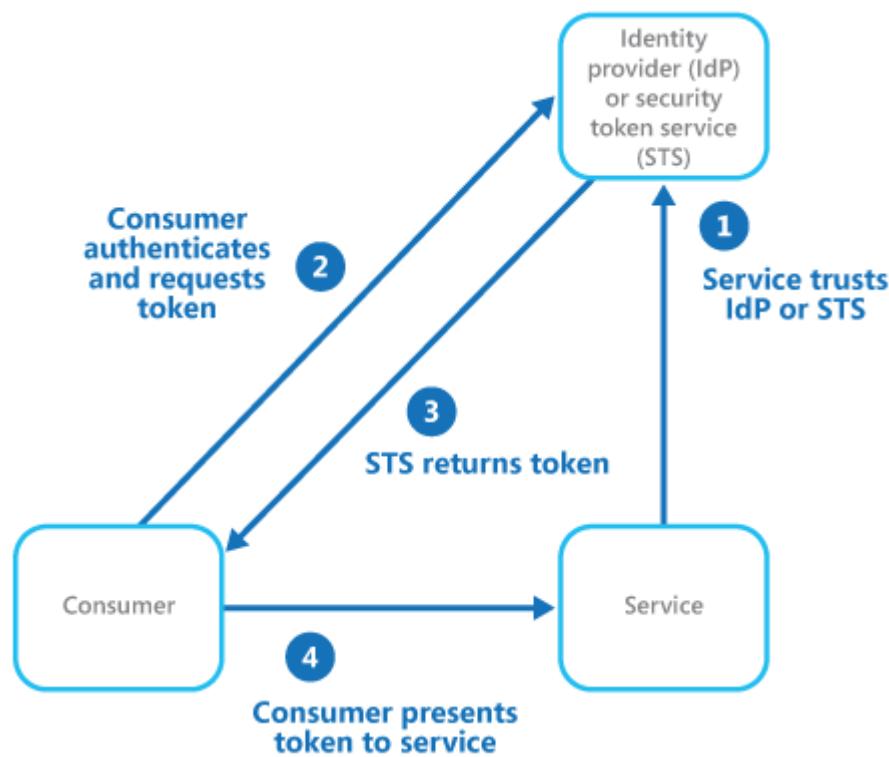
Solution

Implement an authentication mechanism that can use federated identity. Separate user authentication from the application code, and delegate authentication to a trusted identity provider. This can simplify development and allow users to authenticate using a wider range of identity providers (IdP) while minimizing the administrative overhead. It also allows you to clearly decouple authentication from authorization.

The trusted identity providers include corporate directories, on-premises federation services, other security token services (STS) provided by business partners, or social identity providers that can authenticate users who have, for example, a Microsoft, Google, Yahoo!, or Facebook account.

The figure illustrates the Federated Identity pattern when a client application needs to access a service that requires authentication. The authentication is performed by an IdP that works in concert with an STS. The IdP issues security tokens that provide

information about the authenticated user. This information, referred to as claims, includes the user's identity, and might also include other information such as role membership and more granular access rights.



This model is often called claims-based access control. Applications and services authorize access to features and functionality based on the claims contained in the token. The service that requires authentication must trust the IdP. The client application contacts the IdP that performs the authentication. If the authentication is successful, the IdP returns a token containing the claims that identify the user to the STS (note that the IdP and STS can be the same service). The STS can transform and augment the claims in the token based on predefined rules, before returning it to the client. The client application can then pass this token to the service as proof of its identity.

There might be additional security token services in the chain of trust. For example, in the scenario described later, an on-premises STS trusts another STS that is responsible for accessing an identity provider to authenticate the user. This approach is common in enterprise scenarios where there's an on-premises STS and directory.

Federated authentication provides a standards-based solution to the issue of trusting identities across diverse domains, and can support single sign-on. This type of authentication is becoming more common across all types of applications, especially cloud-hosted applications, because it supports single sign-on without requiring a direct network connection to identity providers. The user doesn't have to enter credentials for every application. This increases security because it prevents the creation of credentials

required to access many different applications, and it also hides the user's credentials from all but the original identity provider. Applications see just the authenticated identity information contained within the token.

Federated identity also has the major advantage that management of the identity and credentials is the responsibility of the identity provider. The application or service doesn't need to provide identity management features. In addition, in corporate scenarios, the corporate directory doesn't need to know about the user if it trusts the identity provider. This removes all the administrative overhead of managing the user identity within the directory.

Issues and considerations

Consider the following when designing applications that implement federated authentication:

- Authentication can be a single point of failure. If you deploy your application to multiple datacenters, consider deploying your identity management mechanism to the same datacenters to maintain application reliability and availability.
- Authentication tools make it possible to configure access control based on role claims contained in the authentication token. This is often referred to as role-based access control (RBAC), and it can allow a more granular level of control over access to features and resources.
- Unlike a corporate directory, claims-based authentication using social identity providers doesn't usually provide information about the authenticated user other than an email address, and perhaps a name. Some social identity providers, such as a Microsoft account, provide only a unique identifier. The application usually needs to maintain some information on registered users, and be able to match this information to the identifier contained in the claims in the token. Typically this is done through registration when the user first accesses the application, and information is then injected into the token as additional claims after each authentication.
- If there's more than one identity provider configured for the STS, STS must determine which identity provider the user should be redirected to for authentication. This process is called home realm discovery. The STS might be able to do this automatically based on an email address or user name that the user provides, a subdomain of the application that the user is accessing, the user's IP address scope, or on the contents of a cookie stored in the user's browser. For example, if the user entered an email address in the Microsoft domain, such as

user@live.com, the STS will redirect the user to the Microsoft account sign-in page. On later visits, the STS could use a cookie to indicate that the last sign in was with a Microsoft account. If automatic discovery can't determine the home realm, the STS will display a home realm discovery page that lists the trusted identity providers, and the user must select the one they want to use.

When to use this pattern

This pattern is useful for scenarios such as:

- **Single sign-on in the enterprise.** In this scenario you need to authenticate employees for corporate applications that are hosted in the cloud outside the corporate security boundary, without requiring them to sign in every time they visit an application. The user experience is the same as when using on-premises applications where they're authenticated when signing in to a corporate network, and from then on have access to all relevant applications without needing to sign in again.
- **Federated identity with multiple partners.** In this scenario you need to authenticate both corporate employees and business partners who don't have accounts in the corporate directory. This is common in business-to-business applications, applications that integrate with third-party services, and where companies with different IT systems have merged or shared resources.
- **Federated identity in SaaS applications.** In this scenario independent software vendors provide a ready-to-use service for multiple clients or tenants. Each tenant authenticates using a suitable identity provider. For example, business users will use their corporate credentials, while consumers and clients of the tenant will use their social identity credentials.

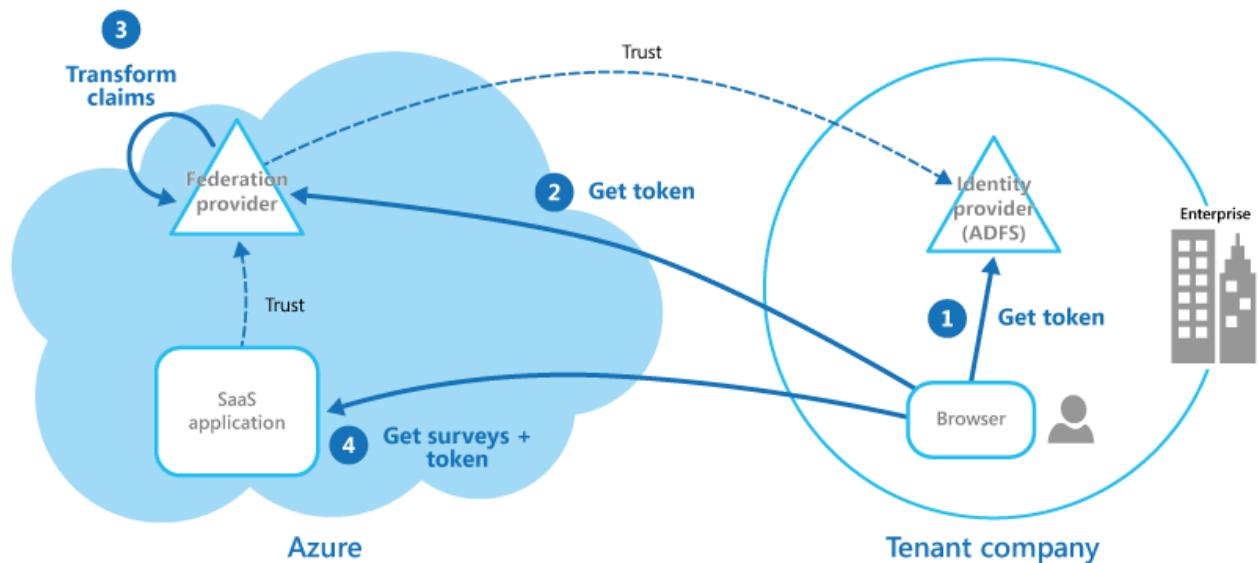
This pattern might not be useful in the following situations:

- All users of the application can be authenticated by one identity provider, and there's no requirement to authenticate using any other identity provider. This is typical in business applications that use a corporate directory (accessible within the application) for authentication, by using a VPN, or (in a cloud-hosted scenario) through a virtual network connection between the on-premises directory and the application.
- The application was originally built using a different authentication mechanism, perhaps with custom user stores, or doesn't have the capability to handle the negotiation standards used by claims-based technologies. Retrofitting claims-

based authentication and access control into existing applications can be complex, and probably not cost effective.

Example

An organization hosts a multi-tenant software as a service (SaaS) application in Microsoft Azure. The application includes a website that tenants can use to manage the application for their own users. The application allows tenants to access the website by using a federated identity that is generated by Active Directory Federation Services (AD FS) when a user is authenticated by that organization's own Active Directory.



The figure shows how tenants authenticate with their own identity provider (step 1), in this case AD FS. After successfully authenticating a tenant, AD FS issues a token. The client browser forwards this token to the SaaS application's federation provider, which trusts tokens issued by the tenant's AD FS, in order to get back a token that is valid for the SaaS federation provider (step 2). If necessary, the SaaS federation provider performs a transformation on the claims in the token into claims that the application recognizes (step 3) before returning the new token to the client browser. The application trusts tokens issued by the SaaS federation provider and uses the claims in the token to apply authorization rules (step 4).

Tenants won't need to remember separate credentials to access the application, and an administrator at the tenant's company can configure in its own AD FS the list of users that can access the application.

Next steps

- [Microsoft Entra ID](#)
- [Active Directory Domain Services](#)

- Active Directory Federation Services
- Multitenant Applications in Azure

Related resources

- Identity management for multitenant applications in Microsoft Azure
- Gatekeeper pattern
- Edge Workload Configuration pattern
- Compute Resource Consolidation pattern
- Gateway Offloading pattern

Gatekeeper pattern

Azure Dedicated Host

Protect applications and services by using a dedicated host instance to broker requests between clients and the application or service. The broker validates and sanitizes the requests, and can provide an additional layer of security and limit the system's attack surface.

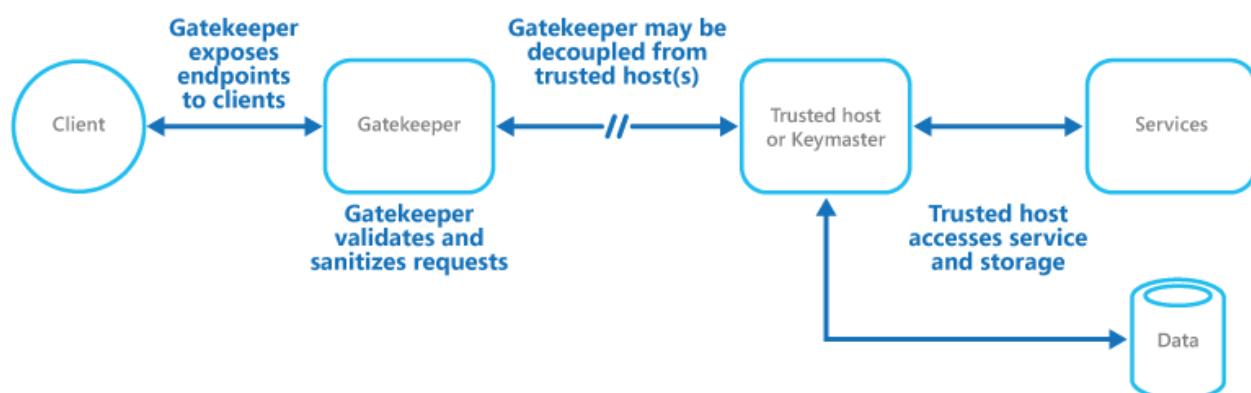
Context and problem

Cloud services expose endpoints that allow client applications to call their APIs. The code used to implement the APIs triggers or performs several tasks, including but not limited to authentication, authorization, parameter validation, and some or all request processing. The API code is likely to access storage and other services on behalf of the client.

If a malicious user compromises the system and gains access to the application's hosting environment, its security mechanisms and access to data and other services are exposed. As a result, the malicious user can gain unrestricted access to credentials, storage keys, sensitive information, and other services.

Solution

One solution to this problem is to decouple the code that implements public endpoints, from the code that processes requests and accesses storage. You can achieve decoupling by using a façade or a dedicated task that interacts with clients and then hands off the request—perhaps through a decoupled interface—to the hosts or tasks that handle the request. The figure provides a high-level overview of this pattern.



The gatekeeper pattern can be used to protect storage, or it can be used as a more comprehensive façade to protect all of the functions of the application. The important factors are:

- **Controlled validation.** The gatekeeper validates all requests, and rejects requests that don't meet validation requirements.
- **Limited risk and exposure.** The gatekeeper doesn't have access to the credentials or keys used by the trusted host to access storage and services. If the gatekeeper is compromised, the attacker doesn't get access to these credentials or keys.
- **Appropriate security.** The gatekeeper runs in a limited privilege mode, while the rest of the application runs in the full trust mode required to access storage and services. If the gatekeeper is compromised, it can't directly access the application services or data.

This pattern acts like a firewall in a typical network topography. It allows the gatekeeper to examine requests and make a decision about whether to pass the request on to the trusted host that performs the required tasks. This decision typically requires the gatekeeper to validate and sanitize the request content before passing it on to the trusted host.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- Ensure that the trusted hosts expose only internal or protected endpoints, used only by the gatekeeper. The trusted hosts shouldn't expose any external endpoints or interfaces.
- The gatekeeper must run in a limited privilege mode, which typically requires running the gatekeeper and the trusted host in separate hosted services or virtual machines.
- The gatekeeper shouldn't perform any processing related to the application or services or access any data. Its function is purely to validate and sanitize requests. The trusted hosts might need to perform additional request validation, but the gatekeeper should perform the core validation.
- Use a secure communication channel (HTTPS, SSL, or TLS) between the gatekeeper and the trusted hosts or tasks where possible. However, some hosting environments don't support HTTPS on internal endpoints.
- Adding the extra layer to implement the gatekeeper pattern will likely impact performance due to the additional processing and network communication required.

- The gatekeeper instance could be a single point of failure. To minimize the impact of a failure, consider deploying redundant instances and using an autoscaling mechanism to ensure capacity to maintain availability.

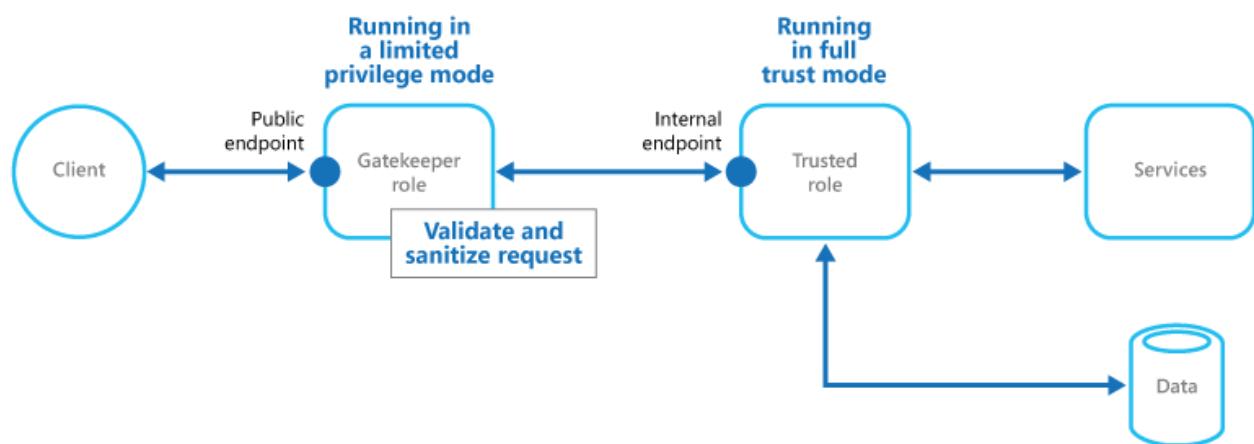
When to use this pattern

This pattern is helpful for applications that:

- handle sensitive information
- expose services that require a high degree of protection from malicious attacks
- perform mission-critical operations that can't be disrupted.
- require request validation be performed separately from the main tasks, or to centralize this validation to simplify maintenance and administration

Example

In a cloud-hosted scenario, this pattern can be implemented by decoupling the gatekeeper role or virtual machine, from the trusted roles and services in an application. The implementation can use an internal endpoint, a queue, or storage as an intermediate communication mechanism. The figure illustrates using an internal endpoint.



Related resources

The [Valet Key pattern](#) might also be relevant when implementing the Gatekeeper pattern. When communicating between the Gatekeeper and trusted roles, it's a good practice to enhance security by using keys or tokens that limit permissions for accessing resources. The pattern describes using a token or key that provides clients with restricted, direct access to a specific resource or service.

Gateway Aggregation pattern

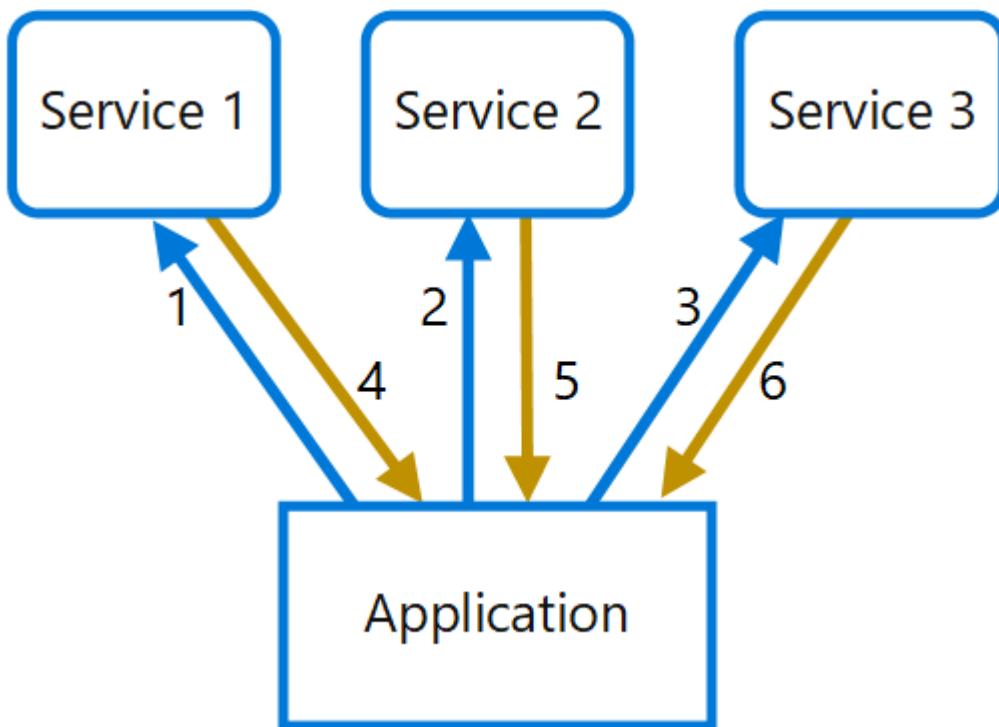
Azure Traffic Manager

Use a gateway to aggregate multiple individual requests into a single request. This pattern is useful when a client must make multiple calls to different backend systems to perform an operation.

Context and problem

To perform a single task, a client may have to make multiple calls to various backend services. An application that relies on many services to perform a task must expend resources on each request. When any new feature or service is added to the application, additional requests are needed, further increasing resource requirements and network calls. This chattiness between a client and a backend can adversely impact the performance and scale of the application. Microservice architectures have made this problem more common, as applications built around many smaller services naturally have a higher amount of cross-service calls.

In the following diagram, the client sends requests to each service (1,2,3). Each service processes the request and sends the response back to the application (4,5,6). Over a cellular network with typically high latency, using individual requests in this manner is inefficient and could result in broken connectivity or incomplete requests. While each request may be done in parallel, the application must send, wait, and process data for each request, all on separate connections, increasing the chance of failure.

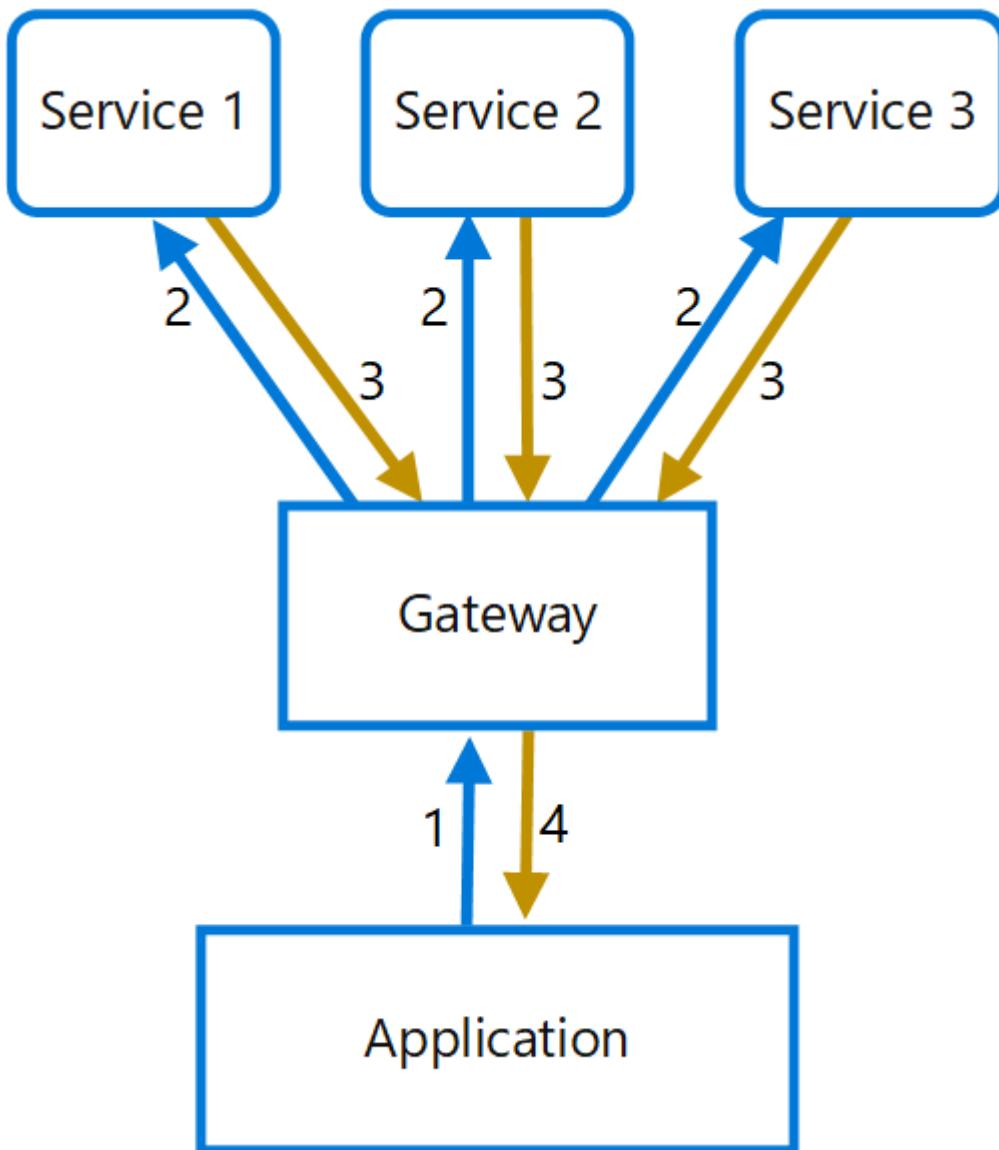


Solution

Use a gateway to reduce chattiness between the client and the services. The gateway receives client requests, dispatches requests to the various backend systems, and then aggregates the results and sends them back to the requesting client.

This pattern can reduce the number of requests that the application makes to backend services, and improve application performance over high-latency networks.

In the following diagram, the application sends a request to the gateway (1). The request contains a package of additional requests. The gateway decomposes these and processes each request by sending it to the relevant service (2). Each service returns a response to the gateway (3). The gateway combines the responses from each service and sends the response to the application (4). The application makes a single request and receives only a single response from the gateway.



Issues and considerations

- The gateway should not introduce service coupling across the backend services.
- The gateway should be located near the backend services to reduce latency as much as possible.
- The gateway service may introduce a single point of failure. Ensure the gateway is properly designed to meet your application's availability requirements.
- The gateway may introduce a bottleneck. Ensure the gateway has adequate performance to handle load and can be scaled to meet your anticipated growth.
- Perform load testing against the gateway to ensure you don't introduce cascading failures for services.
- Implement a resilient design, using techniques such as [bulkheads](#), [circuit breaking](#), [retry](#), and timeouts.
- If one or more service calls takes too long, it may be acceptable to timeout and return a partial set of data. Consider how your application will handle this scenario.

- Use asynchronous I/O to ensure that a delay at the backend doesn't cause performance issues in the application.
- Implement distributed tracing using correlation IDs to track each individual call.
- Monitor request metrics and response sizes.
- Consider returning cached data as a failover strategy to handle failures.
- Instead of building aggregation into the gateway, consider placing an aggregation service behind the gateway. Request aggregation will likely have different resource requirements than other services in the gateway and may impact the gateway's routing and offloading functionality.

When to use this pattern

Use this pattern when:

- A client needs to communicate with multiple backend services to perform an operation.
- The client may use networks with significant latency, such as cellular networks.

This pattern may not be suitable when:

- You want to reduce the number of calls between a client and a single service across multiple operations. In that scenario, it may be better to add a batch operation to the service.
- The client or application is located near the backend services and latency is not a significant factor.

Example

The following example illustrates how to create a simple a gateway aggregation NGINX service using Lua.

```
lua

worker_processes 4;

events {
    worker_connections 1024;
}

http {
    server {
        listen 80;

        location = /batch {
```

```

content_by_lua '
  ngx.req.read_body()

  -- read json body content
  local cjson = require "cjson"
  local batch = cjson.decode(ngx.req.get_body_data())["batch"]

  -- create capture_multi table
  local requests = {}
  for i, item in ipairs(batch) do
    table.insert(requests, {item.relative_url, { method =
  ngx.HTTP_GET}})
  end

  -- execute batch requests in parallel
  local results = {}
  local resps = { ngx.location.capture_multi(requests) }
  for i, res in ipairs(resps) do
    table.insert(results, {status = res.status, body =
  cjson.decode(res.body), header = res.header})
  end

  ngx.say(cjson.encode({results = results}))
';

}

location = /service1 {
  default_type application/json;
  echo '{"attr1":"val1"}';
}

location = /service2 {
  default_type application/json;
  echo '{"attr2":"val2"}';
}
}
}

```

Related resources

- [Backends for Frontends pattern](#)
- [Gateway Offloading pattern](#)
- [Gateway Routing pattern](#)

Gateway Offloading pattern

Azure Application Gateway

Offload shared or specialized service functionality to a gateway proxy. This pattern can simplify application development by moving shared service functionality, such as the use of SSL certificates, from other parts of the application into the gateway.

Context and problem

Some features are commonly used across multiple services, and these features require configuration, management, and maintenance. A shared or specialized service that is distributed with every application deployment increases the administrative overhead and increases the likelihood of deployment error. Any updates to a shared feature must be deployed across all services that share that feature.

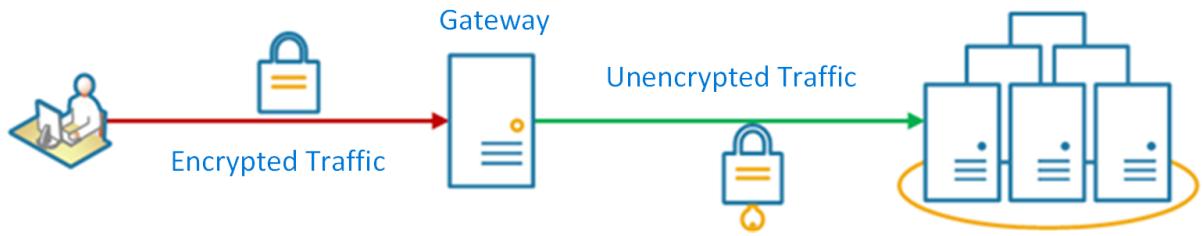
Properly handling security issues (token validation, encryption, SSL certificate management) and other complex tasks can require team members to have highly specialized skills. For example, a certificate needed by an application must be configured and deployed on all application instances. With each new deployment, the certificate must be managed to ensure that it does not expire. Any common certificate that is due to expire must be updated, tested, and verified on every application deployment.

Other common services such as authentication, authorization, logging, monitoring, or [throttling](#) can be difficult to implement and manage across a large number of deployments. It may be better to consolidate this type of functionality, in order to reduce overhead and the chance of errors.

Solution

Offload some features into a gateway, particularly cross-cutting concerns such as certificate management, authentication, SSL termination, monitoring, protocol translation, or throttling.

The following diagram shows a gateway that terminates inbound SSL connections. It requests data on behalf of the original requestor from any HTTP server upstream of the gateway.



Benefits of this pattern include:

- Simplify the development of services by removing the need to distribute and maintain supporting resources, such as web server certificates and configuration for secure websites. Simpler configuration results in easier management and scalability and makes service upgrades simpler.
- Allow dedicated teams to implement features that require specialized expertise, such as security. This allows your core team to focus on the application functionality, leaving these specialized but cross-cutting concerns to the relevant experts.
- Provide some consistency for request and response logging and monitoring. Even if a service is not correctly instrumented, the gateway can be configured to ensure a minimum level of monitoring and logging.

Issues and considerations

- Ensure the gateway is highly available and resilient to failure. Avoid single points of failure by running multiple instances of your gateway.
- Ensure the gateway is designed for the capacity and scaling requirements of your application and endpoints. Make sure the gateway does not become a bottleneck for the application and is sufficiently scalable.
- Only offload features that are used by the entire application, such as security or data transfer.
- Business logic should never be offloaded to the gateway.
- If you need to track transactions, consider generating correlation IDs for logging purposes.

When to use this pattern

Use this pattern when:

- An application deployment has a shared concern such as SSL certificates or encryption.
- A feature that is common across application deployments that may have different resource requirements, such as memory resources, storage capacity or network connections.
- You wish to move the responsibility for issues such as network security, throttling, or other network boundary concerns to a more specialized team.

This pattern may not be suitable if it introduces coupling across services.

Example

Using Nginx as the SSL offload appliance, the following configuration terminates an inbound SSL connection and distributes the connection to one of three upstream HTTP servers.

```
Console

upstream iis {
    server 10.3.0.10    max_fails=3    fail_timeout=15s;
    server 10.3.0.20    max_fails=3    fail_timeout=15s;
    server 10.3.0.30    max_fails=3    fail_timeout=15s;
}

server {
    listen 443;
    ssl on;
    ssl_certificate /etc/nginx/ssl/domain.cer;
    ssl_certificate_key /etc/nginx/ssl/domain.key;

    location / {
        set $targ iis;
        proxy_pass http://$targ;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Host $host;
    }
}
```

On Azure, this can be achieved by [setting up SSL termination on Application Gateway](#).

Related resources

- [Backends for Frontends pattern](#)
- [Gateway Aggregation pattern](#)

- Gateway Routing pattern

Gateway Routing pattern

Azure Application Gateway

Route requests to multiple services or multiple service instances using a single endpoint.

The pattern is useful when you want to:

- Expose multiple services on a single endpoint and route to the appropriate service based on the request
- Expose multiple instances of the same service on a single endpoint for load balancing or availability purposes
- Expose differing versions of the same service on a single endpoint and route traffic across the different versions

Context and problem

When a client needs to consume multiple services, multiple service instances or a combination of both, the client must be updated when services are added or removed. Consider the following scenarios.

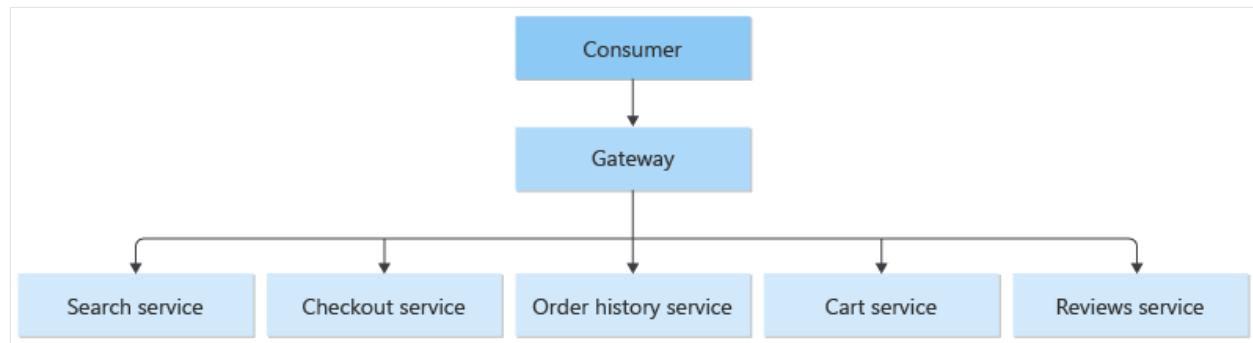
- **Multiple disparate services** - An e-commerce application might provide services such as search, reviews, cart, checkout, and order history. Each service has a different API that the client must interact with, and the client must know about each endpoint in order to connect to the services. If an API changes, the client must be updated as well. If you refactor a service into two or more separate services, the code must change in both the service and the client.
- **Multiple instances of the same service** - The system can require running multiple instances of the same service in the same or different regions. Running multiple instances can be done for load balancing purposes or to meet availability requirements. Each time an instance is spun up or down to match demand, the client must be updated.
- **Multiple versions of the same service** - As part of the deployment strategy, new versions of a service can be deployed along side existing versions. This is known as blue green deployments. In these scenarios, the client must be updated each time there are changes to the percentage of traffic being routed to the new version and existing endpoint.

Solution

Place a gateway in front of a set of applications, services, or deployments. Use application Layer 7 routing to route the request to the appropriate instances.

With this pattern, the client application only needs to know about a single endpoint and communicate with a single endpoint. The following illustrate how the Gateway Routing pattern addresses the three scenarios outlined in the context and problem section.

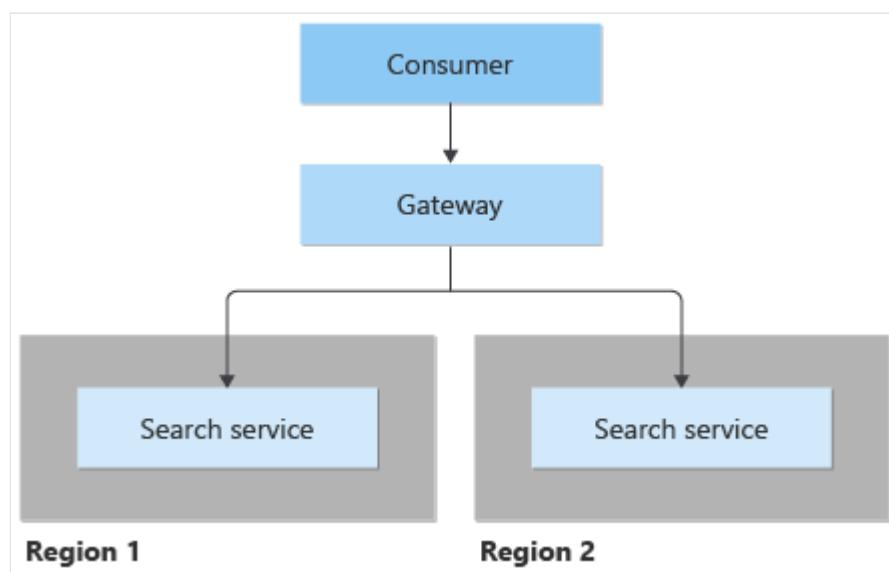
Multiple disparate services



The gateway routing pattern is useful in this scenario where a client is consuming multiple services. If a service is consolidated, decomposed or replaced, the client doesn't necessarily require updating. It can continue making requests to the gateway, and only the routing changes.

A gateway also lets you abstract backend services from the clients, allowing you to keep client calls simple while enabling changes in the backend services behind the gateway. Client calls can be routed to whatever service or services need to handle the expected client behavior, allowing you to add, split, and reorganize services behind the gateway without changing the client.

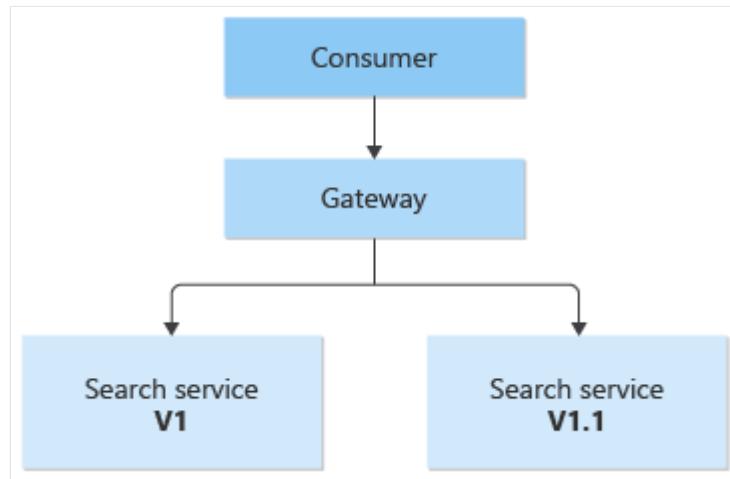
Multiple instances of the same service



Elasticity is key in cloud computing. Services can be spun up to meet increasing demand or spun down when demand is low to save money. The complexity of registering and unregistering service instances is encapsulated in the gateway. The client is unaware of an increase or decreases in the number of services.

Service instances can be deployed in a single or multiple regions. The [Geode pattern](#) details how a multi-region, active-active deployment can improve latency and increase availability of a service.

Multiple versions of the same service



This pattern can be used for deployments, by allowing you to manage how updates are rolled out to users. When a new version of your service is deployed, it can be deployed in parallel with the existing version. Routing lets you control what version of the service is presented to the clients, giving you the flexibility to use various release strategies, whether incremental, parallel, or complete rollouts of updates. Any issues discovered after the new service is deployed can be quickly reverted by making a configuration change at the gateway, without affecting clients.

Issues and considerations

- The gateway service can introduce a single point of failure. Ensure it's properly designed to meet your availability requirements. Consider resiliency and fault tolerance capabilities in the implementation.
- The gateway service can introduce a bottleneck. Ensure the gateway has adequate performance to handle load and can easily scale in line with your growth expectations.
- Perform load testing against the gateway to ensure you don't introduce cascading failures for services.
- Gateway routing is level 7. It can be based on IP, port, header, or URL.

- Gateway services can be global or regional. Azure Front Door is a global gateway, while Azure Application Gateway is regional. Use a global gateway if your solution requires multi-region deployments of services. Consider using Application Gateway if you have a regional workload that requires granular control how traffic is balanced. For example, you want to balance traffic between virtual machines.
- The gateway service is the public endpoint for services it sits in front of. Consider limiting public network access to the backend services, by making the services only accessible via the gateway or via a private virtual network.

When to use this pattern

Use this pattern when:

- A client needs to consume multiple services that can be accessed behind a gateway.
- You want to simplify client applications by using a single endpoint.
- You need to route requests from externally addressable endpoints to internal virtual endpoints, such as exposing ports on a VM to cluster virtual IP addresses.
- A client needs to consume services running in multiple regions for latency or availability benefits.
- A client needs to consume a variable number of service instances.
- You want to implement a deployment strategy where clients access multiple versions of the service at the same time.

This pattern may not be suitable when you have a simple application that uses only one or two services.

Example

Using Nginx as the router, the following is a simple example configuration file for a server that routes requests for applications residing on different virtual directories to different machines at the back end.

Console

```
server {
  listen 80;
  server_name domain.com;

  location /app1 {
    proxy_pass http://10.0.3.10:80;
  }
}
```

```
location /app2 {  
    proxy_pass http://10.0.3.20:80;  
}  
  
location /app3 {  
    proxy_pass http://10.0.3.30:80;  
}  
}
```

The following Azure services can be used to implement the gateway routing pattern:

- An [Application Gateway instance](#), which provides regional layer-7 routing.
- An [Azure Front Door instance](#), which provides global layer-7 routing.

Related resources

- [Backends for Frontends pattern](#)
- [Gateway Aggregation pattern](#)
- [Gateway Offloading pattern](#)

Geode pattern

Azure Cosmos DB

The Geode pattern involves deploying a collection of backend services into a set of **geographical nodes**, each of which can service any request for any client in any region. This pattern allows serving requests in an *active-active* style, improving latency and increasing availability by distributing request processing around the globe.



Context and problem

Many large-scale services have specific challenges around geo-availability and scale. Classic designs often *bring the data to the compute* by storing data in a remote SQL server that serves as the compute tier for that data, relying on scale-up for growth.

The classic approach may present a number of challenges:

- Network latency issues for users coming from the other side of the globe to connect to the hosting endpoint
- Traffic management for demand bursts that can overwhelm the services in a single region
- Cost-prohibitive complexity of deploying copies of app infrastructure into multiple regions for a 24x7 service

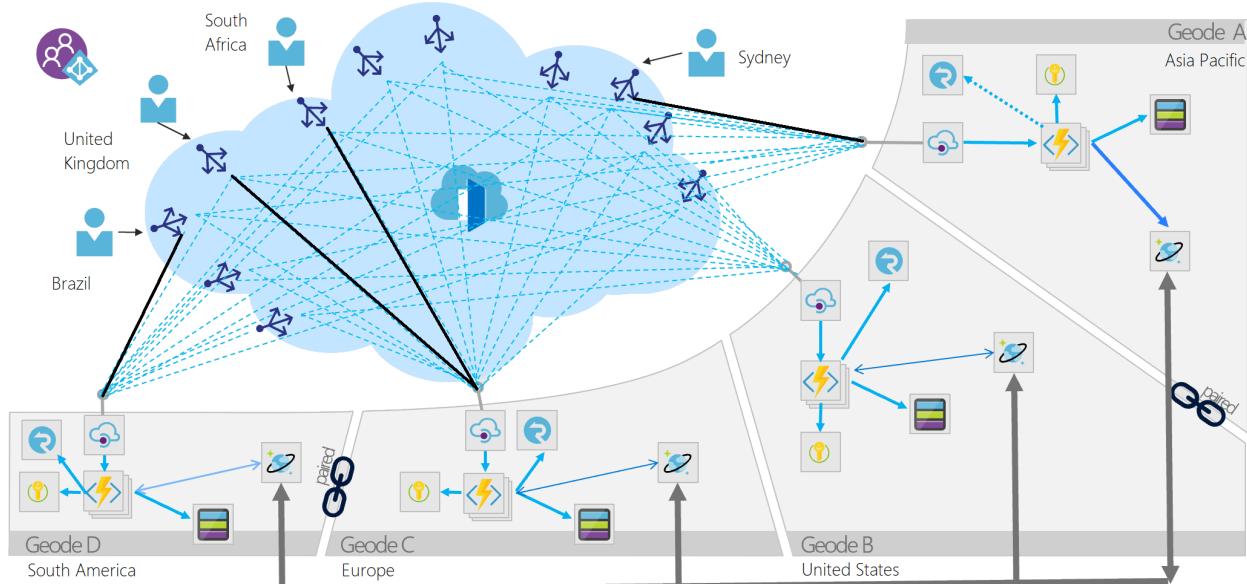
Modern cloud infrastructure has evolved to enable geographic load balancing of front-end services, while allowing for geographic replication of backend services. For availability and performance, getting data closer to the user is good. When data is geo-distributed across a far-flung user base, the geo-distributed datastores should also be

colocated with the compute resources that process the data. The geode pattern *brings the compute to the data*.

Solution

Deploy the service into a number of satellite deployments spread around the globe, each of which is called a *geode*. The geode pattern harnesses key features of Azure to route traffic via the shortest path to a nearby geode, which improves latency and performance. Each geode is behind a global load balancer, and uses a geo-replicated read-write service like [Azure Cosmos DB](#) to host the data plane, ensuring cross-geode data consistency. Data replication services ensure that data stores are identical across geodes, so *all* requests can be served from *all* geodes.

The key difference between a [deployment stamp](#) and a geode is that geodes never exist in isolation. There should always be more than one geode in a production platform.



Geodes have the following characteristics:

- Consist of a collection of disparate types of resources, often defined in a template.
- Have no dependencies outside of the geode footprint and are self-contained. No geode is dependent on another to operate, and if one dies, the others continue to operate.
- Are loosely coupled via an edge network and replication backplane. For example, you can use [Azure Traffic Manager](#) or [Azure Front Door](#) for fronting the geodes, while Azure Cosmos DB can act as the replication backplane. Geodes are not the same as clusters because they share a replication backplane, so the platform takes care of quorum issues.

The geode pattern occurs in big data architectures that use commodity hardware to process data colocated on the same machine, and MapReduce to consolidate results across machines. Another usage is near-edge compute, which brings compute closer to the intelligent edge of the network to reduce response time.

Services can use this pattern over dozens or hundreds of geodes. Furthermore, the resiliency of the whole solution increases with each added geode, since any geodes can take over if a regional outage takes one or more geodes offline.

It's also possible to augment local availability techniques, such as availability zones or paired regions, with the geode pattern for global availability. This increases complexity, but is useful if your architecture is underpinned by a storage engine such as blob storage that can only replicate to a paired region. You can deploy geodes into an intra-zone, zonal, or regional footprint, with a mind to regulatory or latency constraints on location.

Issues and considerations

Use the following techniques and technologies to implement this pattern:

- Modern DevOps practices and tools to produce and rapidly deploy identical geodes across a large number of regions or instances.
- Autoscaling to scale out compute and database throughput instances within a geode. Each geode individually scales out, within the common backplane constraints.
- A front-end service like Azure Front Door that does dynamic content acceleration, split TCP, and Anycast routing.
- A replicating data store like Azure Cosmos DB to control data consistency.
- Serverless technologies where possible, to reduce always-on deployment cost, especially when load is frequently rebalanced around the globe. This strategy allows for many geodes to be deployed with minimal additional investment. Serverless and consumption-based billing technologies reduce waste and cost from duplicate geo-distributed deployments.
- API Management is not required to implement the design pattern, but can be added to each geode that fronts the region's Azure Function App to provide a more robust API layer, enabling the implementation of additional functionality like rate limiting, for instance.

Consider the following points when deciding how to implement this pattern:

- Choose whether to process data locally in each region, or to distribute aggregations in a single geode and replicate the result across the globe. The [Azure](#)

[Cosmos DB change feed processor](#) offers this granular control using its *lease container* concept, and the *leasecollectionprefix* in the corresponding [Azure Functions binding](#). Each approach has distinct advantages and drawbacks.

- Geodes can work in tandem, using the Azure Cosmos DB change feed and a real-time communication platform like SignalR. Geodes can communicate with remote users via other geodes in a mesh pattern, without knowing or caring where the remote user is located.
- This design pattern implicitly decouples everything, resulting in an ultra-highly distributed and decoupled architecture. Consider how to track different components of the same request as they might execute asynchronously on different instances. A proper monitoring strategy is crucial. Both Azure Front Door and Azure Cosmos DB can be easily integrated with Log Analytics and Azure Functions should be deployed alongside Application Insights to provide a robust monitoring system at each component in the architecture.
- Distributed deployments have a greater number of secrets and ingress points that require proper security measures. Key Vault provides a secure layer for secret management and each layer within the API architecture should be properly secured so that the only ingress point for the API is the front-end service like Azure Front Door. The Azure Cosmos DB should restrict traffic to the Azure Function Apps, and the Function apps to Azure Front Door using Microsoft Entra ID or practices like IP restriction.
- Performance is drastically affected by the number of geodes that are deployed and the specific App Service Plans applied to the API technology in each geode. Deployment of additional geodes or movement towards premium tiers come with increased costs for the additional memory and compute, but do not do so on a per transaction basis. Consider load testing the API architecture once deployed and contrast increasing the numbers of geodes with increasing the pricing tier so that the most cost-efficient model is used for your needs.
- Determine the availability requirements for your data. Azure Cosmos DB has optional flags for enabling multi-region write, availability zones, and more. These increase the availability for the Azure Cosmos DB instance and creates a more resilient data layer, but come with additional costs.
- Azure offers a variety of load balancers that provide different functionalities for distribution of traffic. Use the [decision tree](#) to help select the right option for your API's front end.

When to use this pattern

Use this pattern:

- To implement a high-scale platform that has users distributed over a wide area.

- For any service that requires extreme availability and resilience characteristics, because services based on the geode pattern can survive the loss of multiple service regions at the same time.

This pattern might not be suitable for

- Architectures that have constraints so that all geodes can't be equal for data storage. For example, there may be data residency requirements, an application that needs to maintain temporary state for a particular session, or a heavy weighting of requests towards a single region. In this case, consider using [deployment stamps](#) in combination with a global routing plane that is aware of where a user's data sits, such as the traffic routing component described within the [deployment stamps pattern](#).
- Situations where there's no geographical distribution required. Instead, consider availability zones and paired regions for clustering.
- Situations where a legacy platform needs to be retrofitted. This pattern works for cloud-native development only, and can be difficult to retrofit.
- Simple architectures and requirements, where geo-redundancy and geo-distribution aren't required or advantageous.

Examples

- Windows Active Directory implements an early variant of this pattern. Multi-primary replication means all updates and requests can in theory be served from all serviceable nodes, but Flexible Single Master Operation (FSMO) roles mean that all geodes aren't equal.
- The [geode pattern accelerator](#) on GitHub showcases this design pattern in practice and is designed to help developers implement it with real-world APIs.
- A [QnA sample application](#) on GitHub showcases this design pattern in practice.

Health Endpoint Monitoring pattern

Azure App Service

Azure Front Door

Azure Monitor

Azure Traffic Manager

To verify that applications and services are performing correctly, you can use the Health Endpoint Monitoring pattern. This pattern specifies the use of functional checks in an application. External tools can access these checks at regular intervals through exposed endpoints.

Context and problem

It's a good practice to monitor web applications and back-end services. Monitoring helps ensure that applications and services are available and performing correctly. Business requirements often include monitoring.

It's sometimes more difficult to monitor cloud services than on-premises services. One reason is that you don't have full control of the hosting environment. Another is that the services typically depend on other services that platform vendors and others provide.

Many factors affect cloud-hosted applications. Examples include network latency, the performance and availability of the underlying compute and storage systems, and the network bandwidth between them. A service can fail entirely or partially due to any of these factors. To ensure a required level of availability, you must verify at regular intervals that your service performs correctly. Your service level agreement (SLA) might specify the level that you need to meet.

Solution

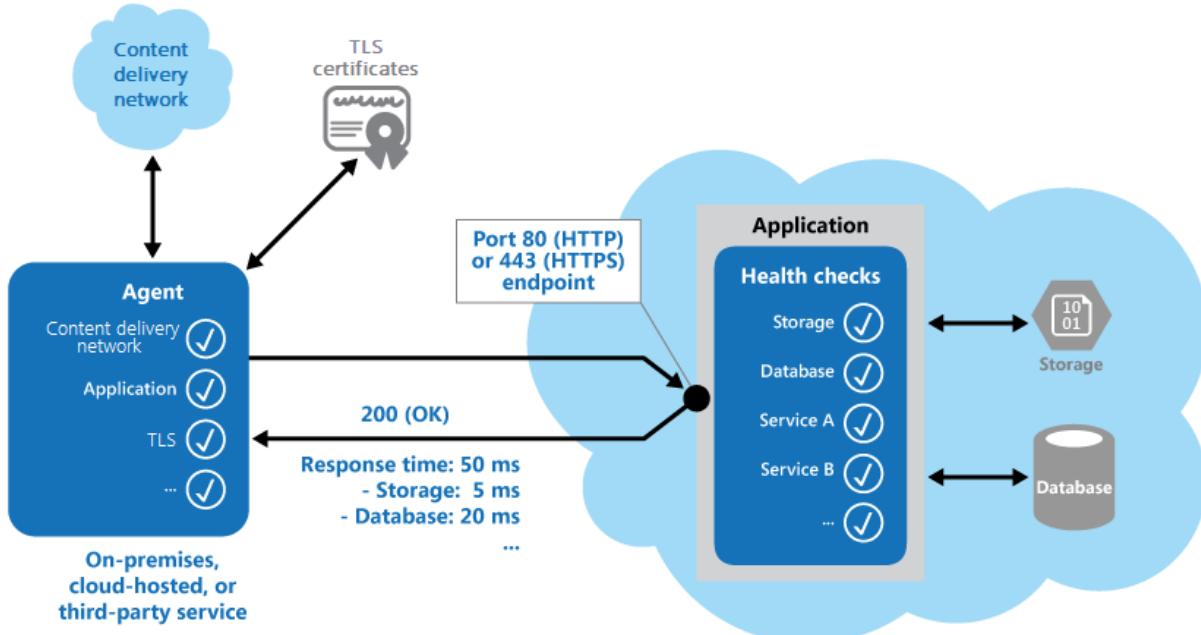
Implement health monitoring by sending requests to an endpoint on your application. The application should perform the necessary checks and then return an indication of its status.

A health monitoring check typically combines two factors:

- The checks (if any) that the application or service performs in response to the request to the health verification endpoint
- The analysis of the results by the tool or framework that performs the health verification check

The response code indicates the status of the application. Optionally, the response code also provides the status of components and services that the app uses. The monitoring tool or framework performs the latency or response time check.

The following figure provides an overview of the pattern.



The health monitoring code in the application might also run other checks to determine:

- The availability and response time of cloud storage or a database.
- The status of other resources or services that the application uses. These resources and services might be in the application or outside it.

Services and tools are available that monitor web applications by submitting a request to a configurable set of endpoints. These services and tools then evaluate the results against a set of configurable rules. It's relatively easy to create a service endpoint for the sole purpose of performing some functional tests on a system.

Typical checks that monitoring tools perform include:

- Validating the response code. For example, an HTTP response of 200 (OK) indicates that the application responded without error. The monitoring system might also check for other response codes to give more comprehensive results.
- Checking the content of the response to detect errors, even when the status code is 200 (OK). By checking the content, you can detect errors that affect only a section of the returned web page or service response. For example, you might check the title of a page or look for a specific phrase that indicates that the app returned the correct page.

- Measuring the response time. The value includes the network latency and the time that the application took to issue the request. An increasing value can indicate an emerging problem with the application or network.
- Checking resources or services that are located outside the application. An example is a content delivery network that the application uses to deliver content from global caches.
- Checking for the expiration of TLS certificates.
- Measuring the response time of a DNS lookup for the URL of the application. This check measures DNS latency and DNS failures.
- Validating the URL that a DNS lookup returns. By validating, you can ensure that entries are correct. You can also help prevent malicious request redirection that might result after an attack on your DNS server.

Where possible, it's also useful to run these checks from different on-premises or hosted locations and then compare response times. Ideally, you should monitor applications from locations that are close to customers. Then you get an accurate view of the performance from each location. This practice provides a more robust checking mechanism. The results can also help you make the following decisions:

- Where to deploy your application
- Whether to deploy it in more than one datacenter

To ensure that your application works correctly for all customers, run tests against all the service instances that customers use. For example, if customer storage is spread across more than one storage account, the monitoring process should check each account.

Issues and considerations

Consider the following points when you decide how to implement this pattern:

- Think about how to validate the response. For example, determine whether a 200 (OK) status code is sufficient to verify that the application is working correctly. Checking the status code is the minimum implementation of this pattern. A status code provides a basic measure of application availability. But a code supplies little information about the operations, trends, and possible upcoming issues in the application.
- Determine the number of endpoints to expose for an application. One approach is to expose at least one endpoint for the core services that the application uses and another for lower-priority services. With this approach, you can assign different levels of importance to each monitoring result. Also consider exposing extra endpoints. You can expose one for each core service to increase monitoring

granularity. For example, a health verification check might check the database, the storage, and an external geocoding service that an application uses. Each might require a different level of uptime and response time. The geocoding service or some other background task might be unavailable for a few minutes. But the application might still be healthy.

- Decide whether to use the same endpoint for monitoring and for general access. You can use the same endpoint for both but design a specific path for health verification checks. For example, you can use `/health` on the general access endpoint. With this approach, monitoring tools can run some functional tests in the application. Examples include registering a new user, signing in, and placing a test order. At the same time, you can also verify that the general access endpoint is available.
- Determine the type of information to collect in the service in response to monitoring requests. You also need to determine how to return this information. Most existing tools and frameworks look only at the HTTP status code that the endpoint returns. To return and validate additional information, you might have to create a custom monitoring utility or service.
- Figure out how much information to collect. Performing excessive processing during the check can overload the application and affect other users. The processing time might also exceed the timeout of the monitoring system. As a result, the system might mark the application as unavailable. Most applications include instrumentation such as error handlers and performance counters. These tools can log performance and detailed error information, which might be sufficient. Consider using this data instead of returning additional information from a health verification check.
- Consider caching the endpoint status. Running the health check frequently might be expensive. For example, if the health status is reported through a dashboard, you don't want every request to the dashboard to trigger a health check. Instead, periodically check the system health, and cache the status. Expose an endpoint that returns the cached status.
- Plan how to configure security for the monitoring endpoints. By configuring security, you can help protect the endpoints from public access, which might:
 - Expose the application to malicious attacks.
 - Risk the exposure of sensitive information.
 - Attract denial of service (DoS) attacks.

Typically, you configure security in the application configuration. Then you can update the settings easily without restarting the application. Consider using one or

more of the following techniques:

- Secure the endpoint by requiring authentication. If the monitoring service or tool supports authentication, you can use an authentication security key in the request header. You can also pass credentials with the request. When you use authentication, consider how to access your health check endpoints. As an example, [Azure App Service has a built-in health check](#) that integrates with App Service authentication and authorization features.
- Use an obscure or hidden endpoint. For example, expose the endpoint on a different IP address than the one that the default application URL uses. Configure the endpoint on a nonstandard HTTP port. Also, consider using a complex path to your test page. You can usually specify extra endpoint addresses and ports in the application configuration. If necessary, you can add entries for these endpoints to the DNS server. Then you avoid having to specify the IP address directly.
- Expose a method on an endpoint that accepts a parameter such as a key value or an operation mode value. When a request arrives, the code can run specific tests that depend on the value of the parameter. The code can return a 404 (Not Found) error if it doesn't recognize the parameter value. Make it possible to define parameter values in the application configuration.
- Use a separate endpoint that performs basic functional tests without compromising the operation of the application. With this approach, you can help reduce the impact of a DoS attack. Ideally, avoid using a test that might expose sensitive information. Sometimes you must return information that might be useful to an attacker. In this case, consider how to protect the endpoint and the data from unauthorized access. Relying on obscurity isn't enough. Consider also using an HTTPS connection and encrypting sensitive data, although this approach increases the load on the server.
- Decide how to ensure that the monitoring agent is performing correctly. One approach is to expose an endpoint that returns a value from the application configuration or a random value that you can use to test the agent. Also ensure that the monitoring system performs checks on itself. You can use a self-test or built-in test to prevent the monitoring system from issuing false positive results.

When to use this pattern

This pattern is useful for:

- Monitoring websites and web applications to verify availability.
- Monitoring websites and web applications to check for correct operation.
- Monitoring middle-tier or shared services to detect and isolate failures that can disrupt other applications.
- Complementing existing instrumentation in the application, such as performance counters and error handlers. Health verification checking doesn't replace application requirements for logging and auditing. Instrumentation can provide valuable information for an existing framework that monitors counters and error logs to detect failures or other issues. But instrumentation can't provide information if an application is unavailable.

Example

You can use the [ASP.NET health checks](#) middleware and libraries to report the health of app infrastructure components. This framework provides a way to report health checks in a consistent way. It implements many of the practices that this article describes. For instance, the ASP.NET health checks include external checks like database connectivity and specific concepts like liveness and readiness probes.

Several example implementations that use ASP.NET health checks are available on [GitHub](#).

Monitor endpoints in Azure-hosted applications

Options for monitoring endpoints in Azure applications include:

- Use the built-in monitoring features of Azure, such as [Azure Monitor](#).
- Use a third-party service or a framework like [Microsoft System Center Operations Manager](#).
- Create a custom utility or a service that runs on your own server or a hosted server.

Even though Azure provides comprehensive monitoring options, you can use additional services and tools to provide extra information. Application Insights, a feature of Monitor, is designed for development teams. This feature helps you understand how your app performs and how it's used. Application Insights monitors request rates, response times, failure rates, and dependency rates. It can help you determine whether external services are slowing you down.

The conditions that you can monitor depend on the hosting mechanism that you choose for your application. All options in this section support alert rules. An alert rule

uses a web endpoint that you specify in the settings for your service. This endpoint should respond in a timely way so that the alert system can detect that the application is operating correctly. For more information, see [Create a new alert rule](#).

If there's a major outage, client traffic should be routable to an application deployment that's available across other regions or zones. This situation is a good case for cross-premises connectivity and global load balancing. The choice depends on whether the application is internal or external facing. Services such as Azure Front Door, Azure Traffic Manager, or content delivery networks can route traffic across regions based on data that health probes provide.

[Traffic Manager](#) is a routing and load-balancing service. It can use a range of rules and settings to distribute requests to specific instances of your application. Besides routing requests, Traffic Manager can regularly ping a URL, port, and relative path. You specify the ping targets with the goal of determining which instances of your application are active and responding to requests. If Traffic Manager detects a status code of 200 (OK), it marks the application as available. Any other status code causes Traffic Manager to mark the application as offline. The Traffic Manager console displays the status of each application. You can configure each rule to reroute requests to other instances of the application that are responding.

Traffic Manager waits for a [certain amount of time](#) to receive a response from the monitoring URL. Make sure that your health verification code runs in this time. Allow for network latency for the round trip from Traffic Manager to your application and back again.

Next steps

The following guidance is useful for implementing this pattern:

- [Health monitoring guidance in microservices-based applications](#)
- [Monitoring application health for reliability](#), part of the Azure Well-Architected Framework
- [Create a new alert rule](#)

Related resources

- [External Configuration Store pattern](#)
- [Circuit Breaker pattern](#)
- [Gateway Routing pattern](#)
- [Gatekeeper pattern](#)

Index Table pattern

Azure Storage

Create indexes over the fields in data stores that are frequently referenced by queries. This pattern can improve query performance by allowing applications to more quickly locate the data to retrieve from a data store.

Context and problem

Many data stores organize the data for a collection of entities using the primary key. An application can use this key to locate and retrieve data. The figure shows an example of a data store holding customer information. The primary key is the Customer ID. The figure shows customer information organized by the primary key (Customer ID).

Primary Key (Customer ID)	Customer Data
1	LastName: Smith, Town: Redmond, ...
2	LastName: Jones, Town: Seattle, ...
3	LastName: Robinson, Town: Portland, ...
4	LastName: Brown, Town: Redmond, ...
5	LastName: Smith, Town: Chicago, ...
6	LastName: Green, Town: Redmond, ...
7	LastName: Clarke, Town: Portland, ...
8	LastName: Smith, Town: Redmond, ...
9	LastName: Jones, Town: Chicago, ...
...	...
1000	LastName: Clarke, Town: Chicago, ...
...	...

While the primary key is valuable for queries that fetch data based on the value of this key, an application might not be able to use the primary key if it needs to retrieve data based on some other field. In the customers example, an application can't use the Customer ID primary key to retrieve customers if it queries data solely by referencing the value of some other attribute, such as the town in which the customer is located. To perform a query such as this, the application might have to fetch and examine every customer record, which could be a slow process.

Many relational database management systems support secondary indexes. A secondary index is a separate data structure that's organized by one or more nonprimary (secondary) key fields, and it indicates where the data for each indexed value is stored. The items in a secondary index are typically sorted by the value of the secondary keys to enable fast lookup of data. These indexes are usually maintained automatically by the database management system.

You can create as many secondary indexes as you need to support the different queries that your application performs. For example, in a Customers table in a relational database where the Customer ID is the primary key, it's beneficial to add a secondary index over the town field if the application frequently looks up customers by the town where they reside.

However, although secondary indexes are common in relational systems, some NoSQL data stores used by cloud applications don't provide an equivalent feature.

Solution

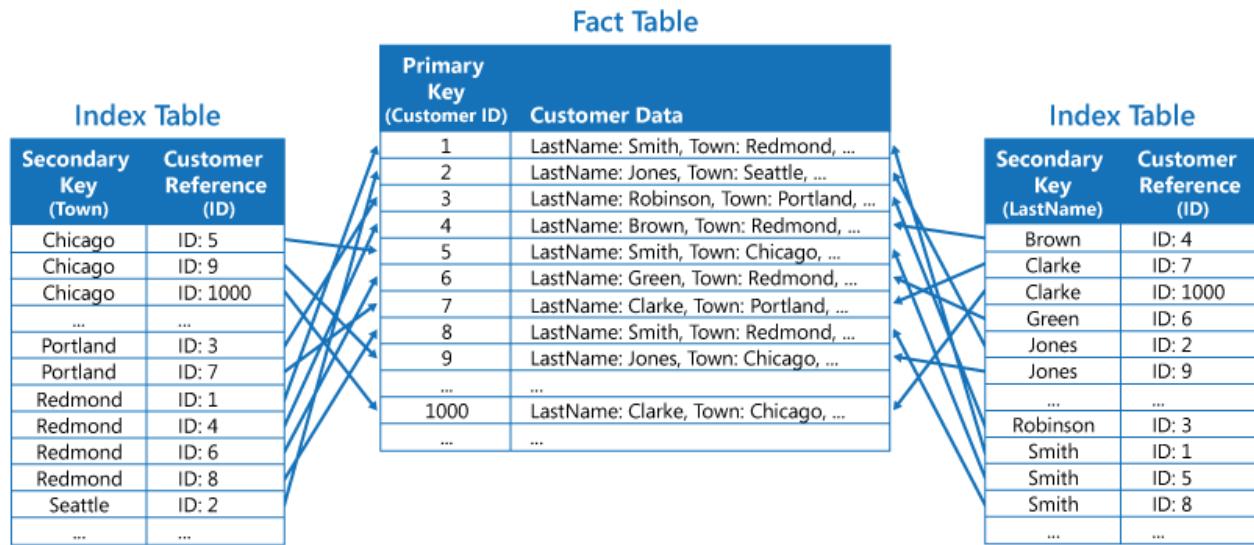
If the data store doesn't support secondary indexes, you can emulate them manually by creating your own index tables. An index table organizes the data by a specified key. Three strategies are commonly used for structuring an index table, depending on the number of secondary indexes that are required and the nature of the queries that an application performs.

The first strategy is to duplicate the data in each index table but organize it by different keys (complete denormalization). The next figure shows index tables that organize the same customer information by Town and LastName.

Secondary Key (Town)	Customer Data	Secondary Key (LastName)	Customer Data
Chicago	ID: 5, LastName: Smith, Town: Chicago, ...	Brown	ID: 4, LastName: Brown, Town: Redmond, ...
Chicago	ID: 9, LastName: Jones, Town: Chicago, ...	Clarke	ID: 7, LastName: Clarke, Town: Portland, ...
Chicago	ID: 1000, LastName: Clarke, Town: Chicago, ...	Clarke	ID: 1000, LastName: Clarke, Town: Chicago, ...
...	...	Green	ID: 6, LastName: Green, Town: Redmond, ...
Portland	ID: 3, LastName: Robinson, Town: Portland, ...	Jones	ID: 2, LastName: Jones, Town: Seattle, ...
Portland	ID: 7, LastName: Clarke, Town: Portland, ...	Jones	ID: 9, LastName: Jones, Town: Chicago, ...
Redmond	ID: 1, LastName: Smith, Town: Redmond,
Redmond	ID: 4, LastName: Brown, Town: Redmond, ...	Robinson	ID: 3, LastName: Robinson, Town: Portland, ...
Redmond	ID: 6, LastName: Green, Town: Redmond, ...	Smith	ID: 1, LastName: Smith, Town: Redmond, ...
Redmond	ID: 8, LastName: Smith, Town: Redmond, ...	Smith	ID: 5, LastName: Smith, Town: Chicago, ...
Seattle	ID: 2, LastName: Jones, Town: Seattle, ...	Smith	ID: 8, LastName: Smith, Town: Redmond, ...
...

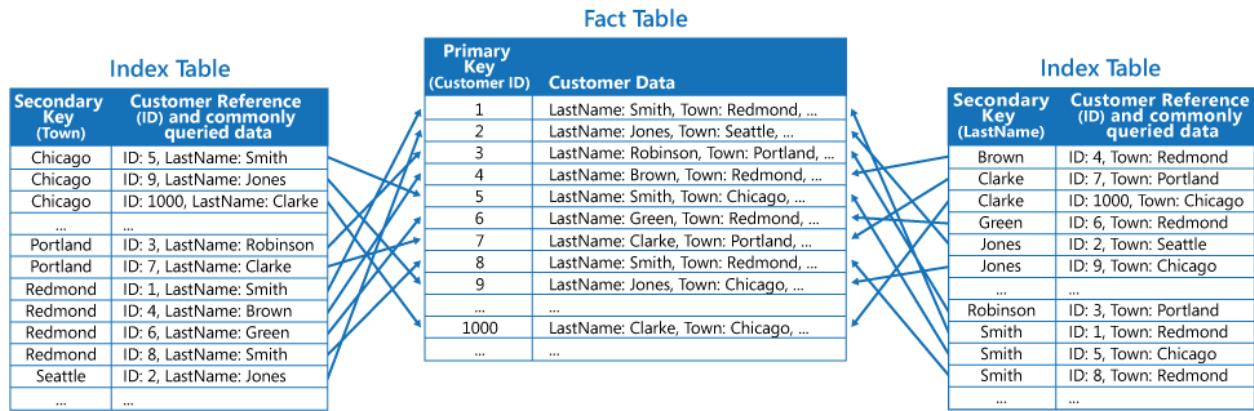
This strategy is appropriate if the data is relatively static compared to the number of times it's queried using each key. If the data is more dynamic, the processing overhead of maintaining each index table becomes too large for this approach to be useful. Also, if the volume of data is very large, the amount of space required to store the duplicate data is significant.

The second strategy is to create normalized index tables organized by different keys and reference the original data by using the primary key rather than duplicating it, as shown in the following figure. The original data is called a fact table.



This technique saves space and reduces the overhead of maintaining duplicate data. The disadvantage is that an application has to perform two lookup operations to find data using a secondary key. It has to find the primary key for the data in the index table, and then use the primary key to look up the data in the fact table.

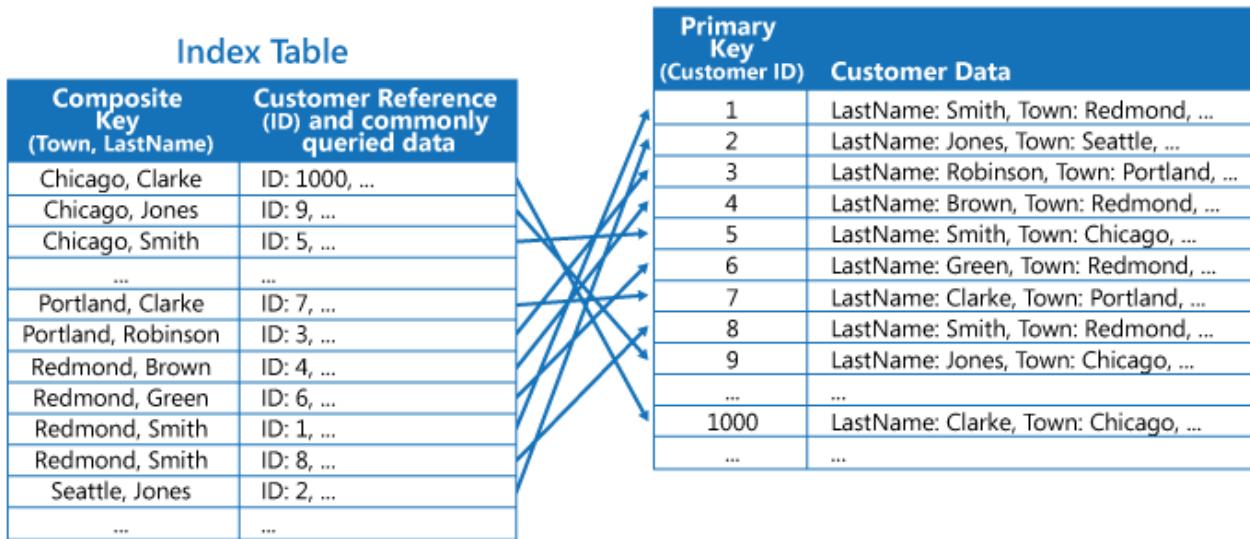
The third strategy is to create partially normalized index tables organized by different keys that duplicate frequently retrieved fields. Reference the fact table to access less frequently accessed fields. The next figure shows how commonly accessed data is duplicated in each index table.



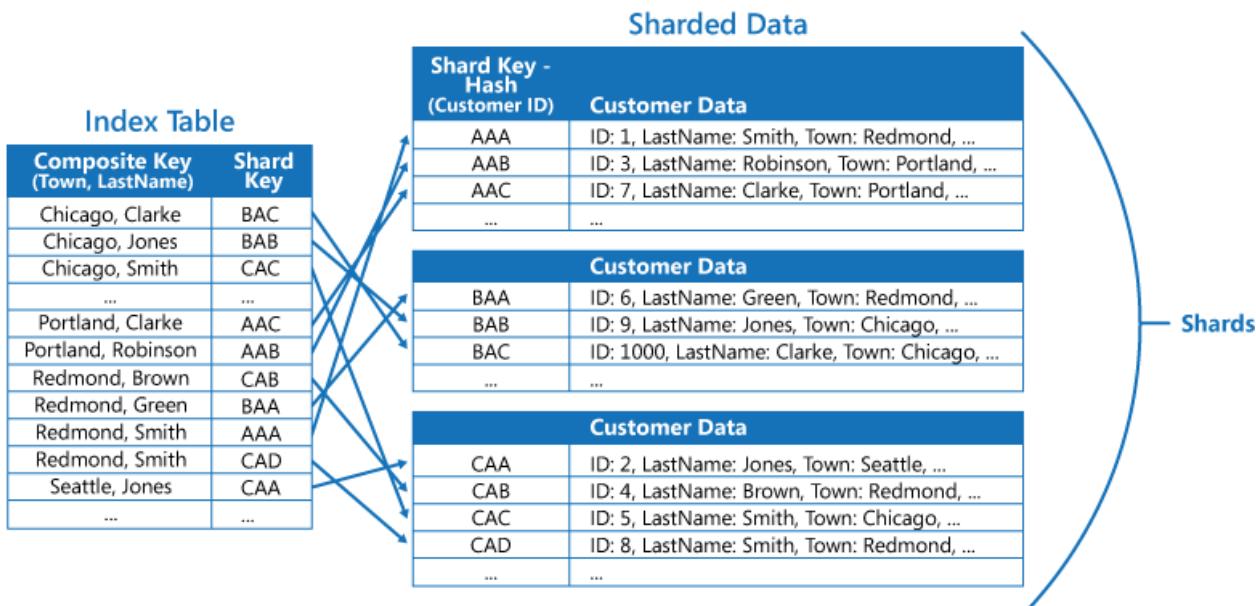
With this strategy, you can strike a balance between the first two approaches. The data for common queries can be retrieved quickly by using a single lookup, while the space and maintenance overhead isn't as significant as duplicating the entire data set.

If an application frequently queries data by specifying a combination of values (for example, "Find all customers that live in Redmond and that have a last name of Smith"), you could implement the keys to the items in the index table as a concatenation of the Town attribute and the LastName attribute. The next figure shows an index table based on composite keys. The keys are sorted by Town, and then by LastName for records that have the same value for Town.

Fact Table



Index tables can speed up query operations over sharded data, and are especially useful where the shard key is hashed. The next figure shows an example where the shard key is a hash of the Customer ID. The index table can organize data by the nonhashed value (Town and LastName), and provide the hashed shard key as the lookup data. This can save the application from repeatedly calculating hash keys (an expensive operation) if it needs to retrieve data that falls within a range, or it needs to fetch data in order of the nonhashed key. For example, a query such as "Find all customers that live in Redmond" can be quickly resolved by locating the matching items in the index table, where they're all stored in a contiguous block. Then, follow the references to the customer data using the shard keys stored in the index table.



Issues and considerations

Consider the following points when deciding how to implement this pattern:

- The overhead of maintaining secondary indexes can be significant. You must analyze and understand the queries that your application uses. Only create index tables when they're likely to be used regularly. Don't create speculative index tables to support queries that an application doesn't perform, or performs only occasionally.
- Duplicating data in an index table can add significant overhead in storage costs and the effort required to maintain multiple copies of data.
- Implementing an index table as a normalized structure that references the original data requires an application to perform two lookup operations to find data. The first operation searches the index table to retrieve the primary key, and the second uses the primary key to fetch the data.
- If a system incorporates a number of index tables over very large data sets, it can be difficult to maintain consistency between index tables and the original data. It might be possible to design the application around the eventual consistency model. For example, to insert, update, or delete data, an application could post a message to a queue and let a separate task perform the operation and maintain the index tables that reference this data asynchronously. For more information about implementing eventual consistency, see the [Data Consistency Primer](#).

Microsoft Azure storage tables support transactional updates for changes made to data held in the same partition (referred to as entity group transactions). If you can store the data for a fact table and one or more index tables in the same partition, you can use this feature to help ensure consistency.

- Index tables might themselves be partitioned or sharded.

When to use this pattern

Use this pattern to improve query performance when an application frequently needs to retrieve data by using a key other than the primary (or shard) key.

This pattern might not be useful when:

- Data is volatile. An index table can become out of date very quickly, making it ineffective or making the overhead of maintaining the index table greater than any savings made by using it.
- A field selected as the secondary key for an index table is nondiscriminating and can only have a small set of values (for example, gender).

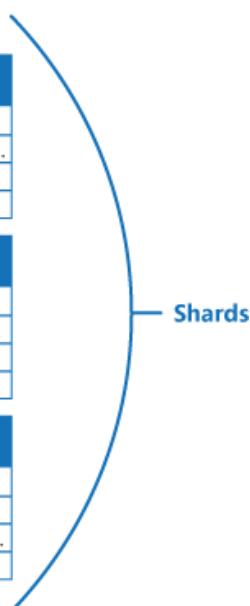
- The balance of the data values for a field selected as the secondary key for an index table are highly skewed. For example, if 90% of the records contain the same value in a field, then creating and maintaining an index table to look up data based on this field might create more overhead than scanning sequentially through the data. However, if queries very frequently target values that lie in the remaining 10%, this index can be useful. You should understand the queries that your application is performing, and how frequently they're performed.

Example

Azure storage tables provide a highly scalable key/value data store for applications running in the cloud. Applications store and retrieve data values by specifying a key. The data values can contain multiple fields, but the structure of a data item is opaque to table storage, which simply handles a data item as an array of bytes.

Azure storage tables also support sharding. The sharding key includes two elements, a partition key and a row key. Items that have the same partition key are stored in the same partition (shard), and the items are stored in row key order within a shard. Table storage is optimized for performing queries that fetch data falling within a contiguous range of row key values within a partition. If you're building cloud applications that store information in Azure tables, you should structure your data with this feature in mind.

For example, consider an application that stores information about movies. The application frequently queries movies by genre (action, documentary, historical, comedy, drama, and so on). You could create an Azure table with partitions for each genre by using the genre as the partition key, and specifying the movie name as the row key, as shown in the next figure.



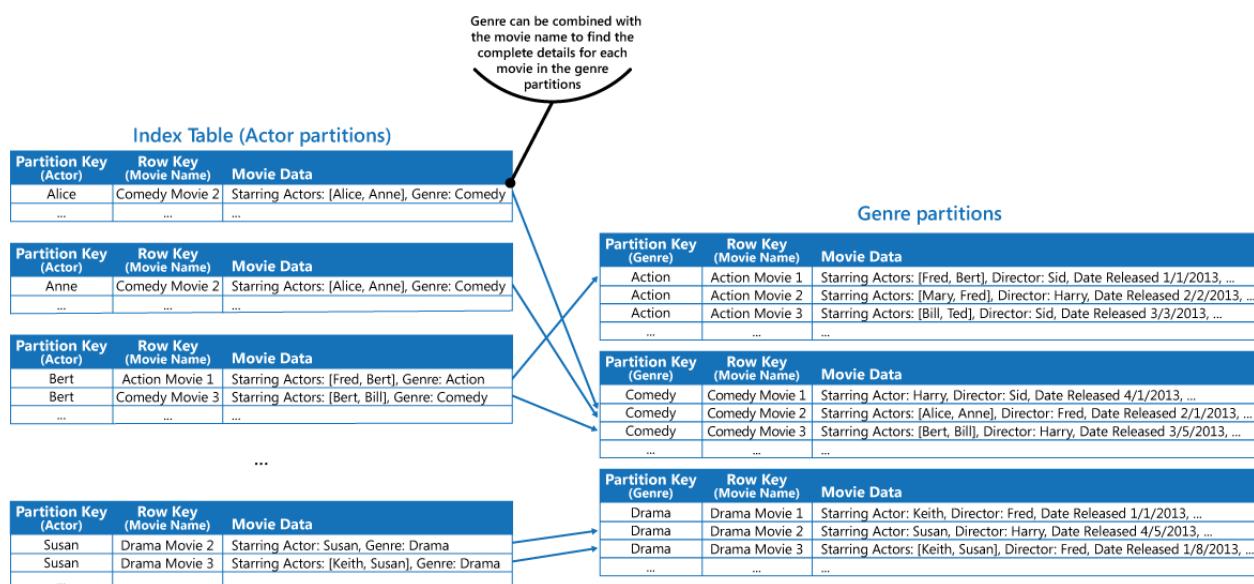
Partition Key (Genre)	Row Key (Movie Name)	Movie Data
Action	Action Movie 1	Starring Actors: [Fred, Bert], Director: Sid, Date Released 1/1/2013, ...
Action	Action Movie 2	Starring Actors: [Mary, Fred], Director: Harry, Date Released 2/2/2013, ...
Action	Action Movie 3	Starring Actors: [Bill, Ted], Director: Sid, Date Released 3/3/2013, ...
...

Partition Key (Genre)	Row Key (Movie Name)	Movie Data
Comedy	Comedy Movie 1	Starring Actor: Harry, Director: Sid, Date Released 4/1/2013, ...
Comedy	Comedy Movie 2	Starring Actors: [Alice, Anne], Director: Fred, Date Released 2/1/2013, ...
Comedy	Comedy Movie 3	Starring Actors: [Bert, Bill], Director: Harry, Date Released 3/5/2013, ...
...

Partition Key (Genre)	Row Key (Movie Name)	Movie Data
Drama	Drama Movie 1	Starring Actor: Keith, Director: Fred, Date Released 1/1/2013, ...
Drama	Drama Movie 2	Starring Actor: Susan, Director: Harry, Date Released 4/5/2013, ...
Drama	Drama Movie 3	Starring Actors: [Keith, Susan], Director: Fred, Date Released 1/8/2013, ...
...

This approach is less effective if the application also needs to query movies by starring actor. In this case, you can create a separate Azure table that acts as an index table. The partition key is the actor and the row key is the movie name. The data for each actor will be stored in separate partitions. If a movie stars more than one actor, the same movie will occur in multiple partitions.

You can duplicate the movie data in the values held by each partition by adopting the first approach described in the Solution section above. However, it's likely that each movie will be replicated several times (once for each actor), so it might be more efficient to partially denormalize the data to support the most common queries (such as the names of the other actors) and enable an application to retrieve any remaining details by including the partition key necessary to find the complete information in the genre partitions. This approach is described by the third option in the Solution section. The next figure shows this approach.



Next steps

- **Data Consistency Primer.** An index table must be maintained as the data that it indexes changes. In the cloud, it might not be possible or appropriate to perform operations that update an index as part of the same transaction that modifies the data. In that case, an eventually consistent approach is more suitable. Provides information on the issues surrounding eventual consistency.

Related resources

The following patterns might also be relevant when implementing this pattern:

- [Sharding pattern](#). The Index Table pattern is frequently used in conjunction with data partitioned by using shards. The Sharding pattern provides more information on how to divide a data store into a set of shards.
- [Materialized View pattern](#). Instead of indexing data to support queries that summarize data, it might be more appropriate to create a materialized view of the data. Describes how to support efficient summary queries by generating prepopulated views over data.

Leader Election pattern

Azure HDInsight

Coordinate the actions performed by a collection of collaborating instances in a distributed application by electing one instance as the leader that assumes responsibility for managing the others. This can help to ensure that instances don't conflict with each other, cause contention for shared resources, or inadvertently interfere with the work that other instances are performing.

Context and problem

A typical cloud application has many tasks acting in a coordinated manner. These tasks could all be instances running the same code and requiring access to the same resources, or they might be working together in parallel to perform the individual parts of a complex calculation.

The task instances might run separately for much of the time, but it might also be necessary to coordinate the actions of each instance to ensure that they don't conflict, cause contention for shared resources, or accidentally interfere with the work that other task instances are performing.

For example:

- In a cloud-based system that implements horizontal scaling, multiple instances of the same task could be running at the same time with each instance serving a different user. If these instances write to a shared resource, it's necessary to coordinate their actions to prevent each instance from overwriting the changes made by the others.
- If the tasks are performing individual elements of a complex calculation in parallel, the results need to be aggregated when they all complete.

The task instances are all peers, so there isn't a natural leader that can act as the coordinator or aggregator.

Solution

A single task instance should be elected to act as the leader, and this instance should coordinate the actions of the other subordinate task instances. If all of the task instances are running the same code, they are each capable of acting as the leader. Therefore, the

election process must be managed carefully to prevent two or more instances taking over the leader role at the same time.

The system must provide a robust mechanism for selecting the leader. This method has to cope with events such as network outages or process failures. In many solutions, the subordinate task instances monitor the leader through some type of heartbeat method, or by polling. If the designated leader terminates unexpectedly, or a network failure makes the leader unavailable to the subordinate task instances, it's necessary for them to elect a new leader.

There are several strategies for electing a leader among a set of tasks in a distributed environment, including:

- Selecting the task instance with the lowest-ranked instance or process ID.
- Racing to acquire a shared, distributed mutex. The first task instance that acquires the mutex is the leader. However, the system must ensure that, if the leader terminates or becomes disconnected from the rest of the system, the mutex is released to allow another task instance to become the leader.
- Implementing one of the common leader election algorithms such as the [Bully Algorithm](#) or the [Ring Algorithm](#). These algorithms assume that each candidate in the election has a unique ID, and that it can communicate with the other candidates reliably.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- The process of electing a leader should be resilient to transient and persistent failures.
- It must be possible to detect when the leader has failed or has become otherwise unavailable (such as due to a communications failure). How quickly detection is needed is system dependent. Some systems might be able to function for a short time without a leader, during which a transient fault might be fixed. In other cases, it might be necessary to detect leader failure immediately and trigger a new election.
- In a system that implements horizontal autoscaling, the leader could be terminated if the system scales back and shuts down some of the computing resources.
- Using a shared, distributed mutex introduces a dependency on the external service that provides the mutex. The service constitutes a single point of failure. If it becomes unavailable for any reason, the system won't be able to elect a leader.
- Using a single dedicated process as the leader is a straightforward approach. However, if the process fails there could be a significant delay while it's restarted.

The resulting latency can affect the performance and response times of other processes if they're waiting for the leader to coordinate an operation.

- Implementing one of the leader election algorithms manually provides the greatest flexibility for tuning and optimizing the code.

When to use this pattern

Use this pattern when the tasks in a distributed application, such as a cloud-hosted solution, need careful coordination and there's no natural leader.

Avoid making the leader a bottleneck in the system. The purpose of the leader is to coordinate the work of the subordinate tasks, and it doesn't necessarily have to participate in this work itself—although it should be able to do so if the task isn't elected as the leader.

This pattern might not be useful if:

- There's a natural leader or dedicated process that can always act as the leader. For example, it might be possible to implement a singleton process that coordinates the task instances. If this process fails or becomes unhealthy, the system can shut it down and restart it.
- The coordination between tasks can be achieved using a more lightweight method. For example, if several task instances simply need coordinated access to a shared resource, a better solution is to use optimistic or pessimistic locking to control access.
- A third-party solution is more appropriate. For example, the Microsoft Azure HDInsight service (based on Apache Hadoop) uses the services provided by Apache Zookeeper to coordinate the map and reduce tasks that collect and summarize data.

Example

The `DistributedMutex` project in the `LeaderElection` solution (a sample that demonstrates this pattern is available on [GitHub](#)) shows how to use a lease on an Azure Storage blob to provide a mechanism for implementing a shared, distributed mutex. This mutex can be used to elect a leader among a group of role instances in an Azure cloud service. The first role instance to acquire the lease is elected the leader, and remains the leader until it releases the lease or isn't able to renew the lease. Other role instances can continue to monitor the blob lease in case the leader is no longer available.

A blob lease is an exclusive write lock over a blob. A single blob can be the subject of only one lease at any point in time. A role instance can request a lease over a specified blob, and it'll be granted the lease if no other role instance holds a lease over the same blob. Otherwise the request will throw an exception.

To avoid a faulted role instance retaining the lease indefinitely, specify a lifetime for the lease. When this expires, the lease becomes available. However, while a role instance holds the lease it can request that the lease is renewed, and it'll be granted the lease for a further period of time. The role instance can continually repeat this process if it wants to retain the lease. For more information on how to lease a blob, see [Lease Blob \(REST API\)](#).

The `BlobDistributedMutex` class in the C# example below contains the `RunTaskWhenMutexAcquired` method that enables a role instance to attempt to acquire a lease over a specified blob. The details of the blob (the name, container, and storage account) are passed to the constructor in a `BlobSettings` object when the `BlobDistributedMutex` object is created (this object is a simple struct that is included in the sample code). The constructor also accepts a `Task` that references the code that the role instance should run if it successfully acquires the lease over the blob and is elected the leader. Note that the code that handles the low-level details of acquiring the lease is implemented in a separate helper class named `BlobLeaseManager`.

C#

```
public class BlobDistributedMutex
{
    ...
    private readonly BlobSettings blobSettings;
    private readonly Func< CancellationToken, Task> taskToRunWhenLeaseAcquired;
    ...

    public BlobDistributedMutex(BlobSettings blobSettings,
                                Func< CancellationToken, Task> taskToRunWhenLeaseAcquired)
    {
        this.blobSettings = blobSettings;
        this.taskToRunWhenLeaseAcquired = taskToRunWhenLeaseAcquired;
    }

    public async Task RunTaskWhenMutexAcquired(CancellationToken token)
    {
        var leaseManager = new BlobLeaseManager(blobSettings);
        await this.RunTaskWhenBlobLeaseAcquired(leaseManager, token);
    }
    ...
}
```

The `RunTaskWhenMutexAcquired` method in the code sample above invokes the `RunTaskWhenBlobLeaseAcquired` method shown in the following code sample to actually acquire the lease. The `RunTaskWhenBlobLeaseAcquired` method runs asynchronously. If the lease is successfully acquired, the role instance has been elected the leader. The purpose of the `taskToRunWhenLeaseAcquired` delegate is to perform the work that coordinates the other role instances. If the lease isn't acquired, another role instance has been elected as the leader and the current role instance remains a subordinate. Note that the `TryAcquireLeaseOrWait` method is a helper method that uses the `BlobLeaseManager` object to acquire the lease.

C#

```
private async Task RunTaskWhenBlobLeaseAcquired(
    BlobLeaseManager leaseManager, CancellationToken token)
{
    while (!token.IsCancellationRequested)
    {
        // Try to acquire the blob lease.
        // Otherwise wait for a short time before trying again.
        string leaseId = await this.TryAcquireLeaseOrWait(leaseManager,
token);

        if (!string.IsNullOrEmpty(leaseId))
        {
            // Create a new linked cancellation token source so that if either
            // original token is canceled or the lease can't be renewed, the
            // leader task can be canceled.
            using (var leaseCts =
                CancellationTokenSource.CreateLinkedTokenSource(new[] { token }))
            {
                // Run the leader task.
                var leaderTask =
this.taskToRunWhenLeaseAcquired.Invoke(leaseCts.Token);
                ...
            }
        }
    }
}
```

The task started by the leader also runs asynchronously. While this task is running, the `RunTaskWhenBlobLeaseAcquired` method shown in the following code sample periodically attempts to renew the lease. This helps to ensure that the role instance remains the leader. In the sample solution, the delay between renewal requests is less than the time specified for the duration of the lease in order to prevent another role instance from being elected the leader. If the renewal fails for any reason, the task is canceled.

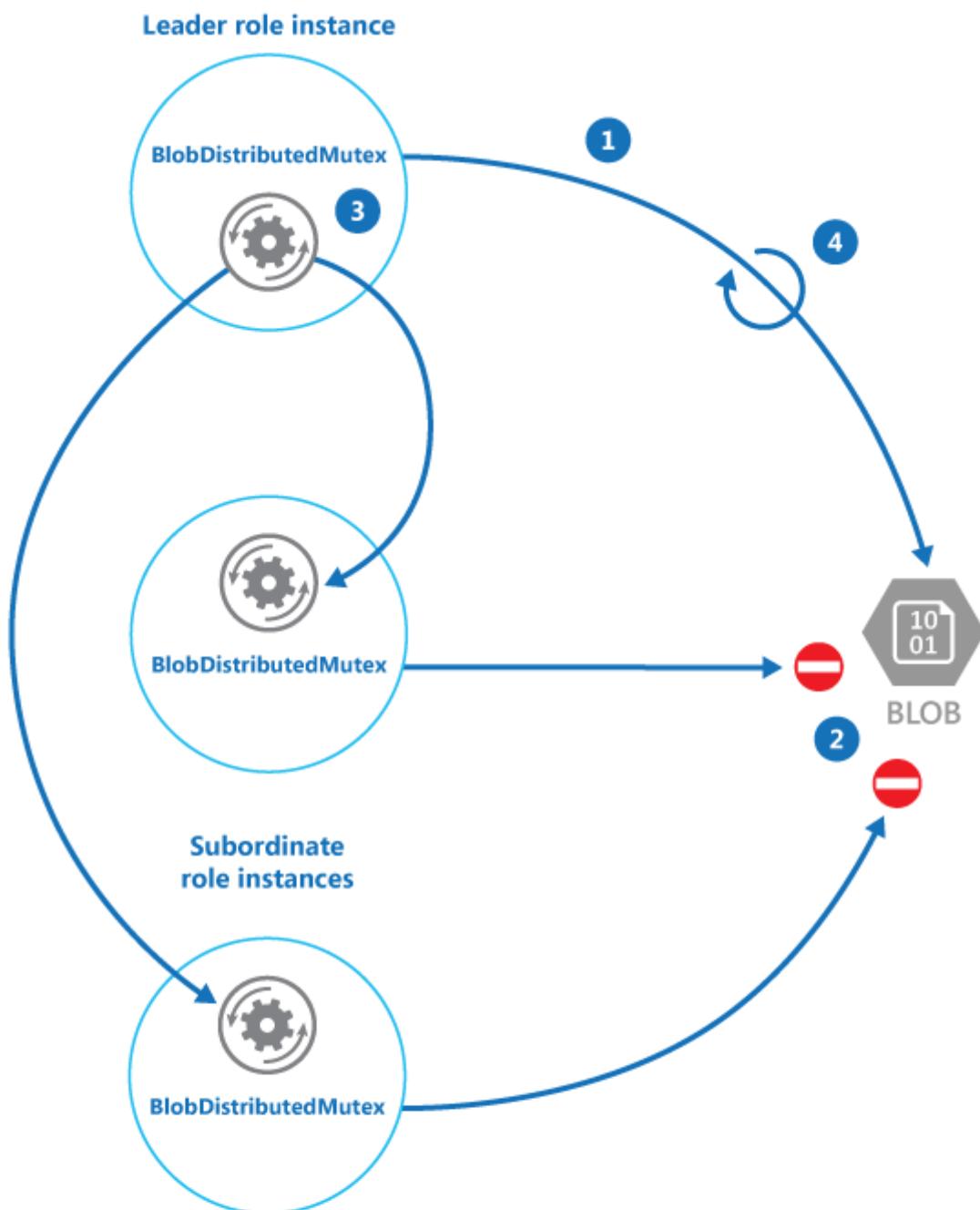
If the lease fails to be renewed or the task is canceled (possibly as a result of the role instance shutting down), the lease is released. At this point, this or another role instance might be elected as the leader. The code extract below shows this part of the process.

C#

```
private async Task RunTaskWhenBlobLeaseAcquired(
    BlobLeaseManager leaseManager, CancellationToken token)
{
    while (...)
    {
        ...
        if (...)
        {
            ...
            using (var leaseCts = ...)
            {
                ...
                // Keep renewing the lease in regular intervals.
                // If the lease can't be renewed, then the task completes.
                var renewLeaseTask =
                    this.KeepRenewingLease(leaseManager, leaseId, leaseCts.Token);

                // When any task completes (either the leader task itself or when
                // it
                // couldn't renew the lease) then cancel the other task.
                await CancelAllWhenAnyCompletes(leaderTask, renewLeaseTask,
                    leaseCts);
            }
        }
    }
    ...
}
```

The `KeepRenewingLease` method is another helper method that uses the `BlobLeaseManager` object to renew the lease. The `CancelAllWhenAnyCompletes` method cancels the tasks specified as the first two parameters. The following diagram illustrates using the `BlobDistributedMutex` class to elect a leader and run a task that coordinates operations.



- 1: A role instance calls the *RunTaskWhenMutexAcquired* method of a *BlobDistributedMutex* object and is granted the lease over the blob. The role instance is elected the leader.
- 2: Other role instances call the *RunTaskWhenMutexAcquired* method and are blocked.
- 3: The *RunTaskWhenMutexAcquired* method in the leader runs a task that coordinates the work of the subordinate role instances.
- 4: The *RunTaskWhenMutexAcquired* method in the leader periodically renews the lease.

The following code example shows how to use the `BlobDistributedMutex` class in a worker role. This code acquires a lease over a blob named `MyLeaderCoordinatorTask` in the lease's container in development storage, and specifies that the code defined in the `MyLeaderCoordinatorTask` method should run if the role instance is elected the leader.

C#

```
var settings = new
BlobSettings(CloudStorageAccount.DevelopmentStorageAccount,
    "leases", "MyLeaderCoordinatorTask");
var cts = new CancellationTokenSource();
var mutex = new BlobDistributedMutex(settings, MyLeaderCoordinatorTask);
mutex.RunTaskWhenMutexAcquired(this.cts.Token);
...

// Method that runs if the role instance is elected the leader
private static async Task MyLeaderCoordinatorTask(CancellationToken token)
{
    ...
}
```

Note the following points about the sample solution:

- The blob is a potential single point of failure. If the blob service becomes unavailable, or is inaccessible, the leader won't be able to renew the lease and no other role instance will be able to acquire the lease. In this case, no role instance will be able to act as the leader. However, the blob service is designed to be resilient, so complete failure of the blob service is considered to be extremely unlikely.
- If the task being performed by the leader stalls, the leader might continue to renew the lease, preventing any other role instance from acquiring the lease and taking over the leader role in order to coordinate tasks. In the real world, the health of the leader should be checked at frequent intervals.
- The election process is nondeterministic. You can't make any assumptions about which role instance will acquire the blob lease and become the leader.
- The blob used as the target of the blob lease shouldn't be used for any other purpose. If a role instance attempts to store data in this blob, this data won't be accessible unless the role instance is the leader and holds the blob lease.

Next steps

The following guidance might also be relevant when implementing this pattern:

- This pattern has a downloadable [sample application](#).
- **Autoscaling Guidance.** It's possible to start and stop instances of the task hosts as the load on the application varies. Autoscaling can help to maintain throughput and performance during times of peak processing.
- **Compute Partitioning Guidance.** This guidance describes how to allocate tasks to hosts in a cloud service in a way that helps to minimize running costs while

maintaining the scalability, performance, availability, and security of the service.

- The [Task-based Asynchronous](#) pattern.
- An example illustrating the [Bully Algorithm](#).
- An example illustrating the [Ring Algorithm](#).
- [Apache Curator](#) a client library for Apache ZooKeeper.
- The article [Lease Blob \(REST API\)](#) on MSDN.

Materialized View pattern

Azure Storage

Generate prepopulated views over the data in one or more data stores when the data isn't ideally formatted for required query operations. This can help support efficient querying and data extraction, and improve application performance.

Context and problem

When storing data, the priority for developers and data administrators is often focused on how the data is stored, as opposed to how it's read. The chosen storage format is usually closely related to the format of the data, requirements for managing data size and data integrity, and the kind of store in use. For example, when using NoSQL document store, the data is often represented as a series of aggregates, each containing all of the information for that entity.

However, this can have a negative effect on queries. When a query only needs a subset of the data from some entities, such as a summary of orders for several customers without all of the order details, it must extract all of the data for the relevant entities in order to obtain the required information.

Solution

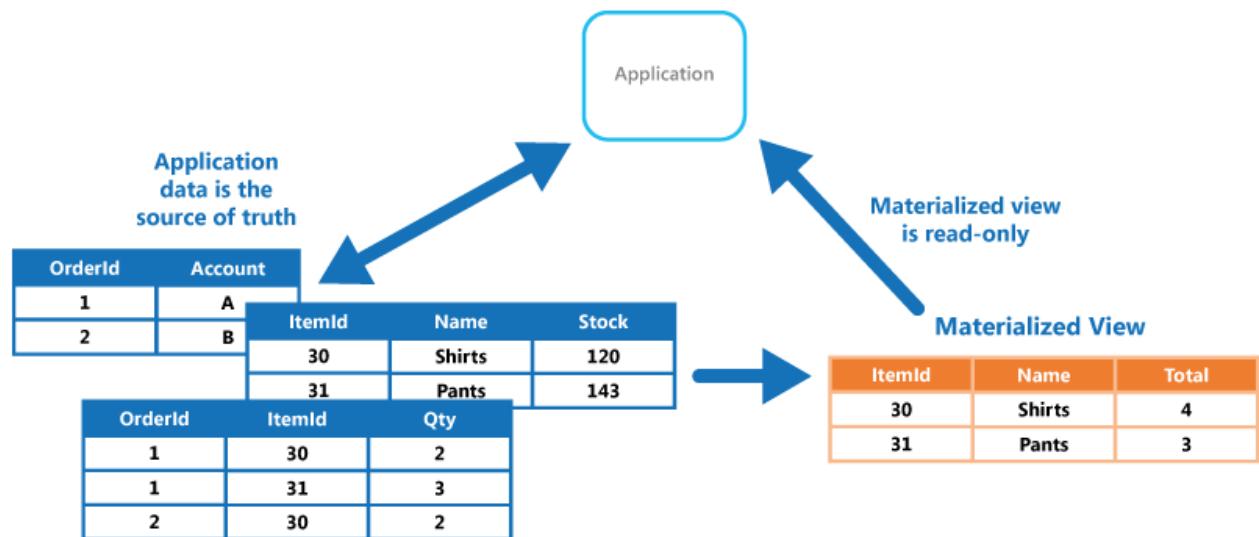
To support efficient querying, a common solution is to generate, in advance, a view that materializes the data in a format suited to the required results set. The Materialized View pattern describes generating prepopulated views of data in environments where the source data isn't in a suitable format for querying, where generating a suitable query is difficult, or where query performance is poor due to the nature of the data or the data store.

These materialized views, which only contain data required by a query, allow applications to quickly obtain the information they need. In addition to joining tables or combining data entities, materialized views can include the current values of calculated columns or data items, the results of combining values or executing transformations on the data items, and values specified as part of the query. A materialized view can even be optimized for just a single query.

A key point is that a materialized view and the data it contains is completely disposable because it can be entirely rebuilt from the source data stores. A materialized view is

never updated directly by an application, and so it's a specialized cache.

When the source data for the view changes, the view must be updated to include the new information. You can schedule this to happen automatically, or when the system detects a change to the original data. In some cases it might be necessary to regenerate the view manually. The figure shows an example of how the Materialized View pattern might be used.



Issues and considerations

Consider the following points when deciding how to implement this pattern:

How and when the view will be updated. Ideally it'll regenerate in response to an event indicating a change to the source data, although this can lead to excessive overhead if the source data changes rapidly. Alternatively, consider using a scheduled task, an external trigger, or a manual action to regenerate the view.

In some systems, such as when using the Event Sourcing pattern to maintain a store of only the events that modified the data, materialized views are necessary. Prepopulating views by examining all events to determine the current state might be the only way to obtain information from the event store. If you're not using Event Sourcing, you need to consider whether a materialized view is helpful or not. Materialized views tend to be specifically tailored to one, or a small number of queries. If many queries are used, materialized views can result in unacceptable storage capacity requirements and storage cost.

Consider the impact on data consistency when generating the view, and when updating the view if this occurs on a schedule. If the source data is changing at the point when the view is generated, the copy of the data in the view won't be fully consistent with the original data.

Consider where you'll store the view. The view doesn't have to be located in the same store or partition as the original data. It can be a subset from a few different partitions combined.

A view can be rebuilt if lost. Because of that, if the view is transient and is only used to improve query performance by reflecting the current state of the data, or to improve scalability, it can be stored in a cache or in a less reliable location.

When defining a materialized view, maximize its value by adding data items or columns to it based on computation or transformation of existing data items, on values passed in the query, or on combinations of these values when appropriate.

Where the storage mechanism supports it, consider indexing the materialized view to further increase performance. Most relational databases support indexing for views, as do big data solutions based on Apache Hadoop.

When to use this pattern

This pattern is useful when:

- Creating materialized views over data that's difficult to query directly, or where queries must be very complex to extract data that's stored in a normalized, semi-structured, or unstructured way.
- Creating temporary views that can dramatically improve query performance, or can act directly as source views or data transfer objects for the UI, for reporting, or for display.
- Supporting occasionally connected or disconnected scenarios where connection to the data store isn't always available. The view can be cached locally in this case.
- Simplifying queries and exposing data for experimentation in a way that doesn't require knowledge of the source data format. For example, by joining different tables in one or more databases, or one or more domains in NoSQL stores, and then formatting the data to fit its eventual use.
- Providing access to specific subsets of the source data that, for security or privacy reasons, shouldn't be generally accessible, open to modification, or fully exposed to users.
- Bridging different data stores, to take advantage of their individual capabilities. For example, using a cloud store that's efficient for writing as the reference data store, and a relational database that offers good query and read performance to hold the materialized views.
- When using microservices, you are recommended to keep them loosely coupled, including their data storage. Therefore, materialized views can help you consolidate data from your services. If materialized views are not appropriate in

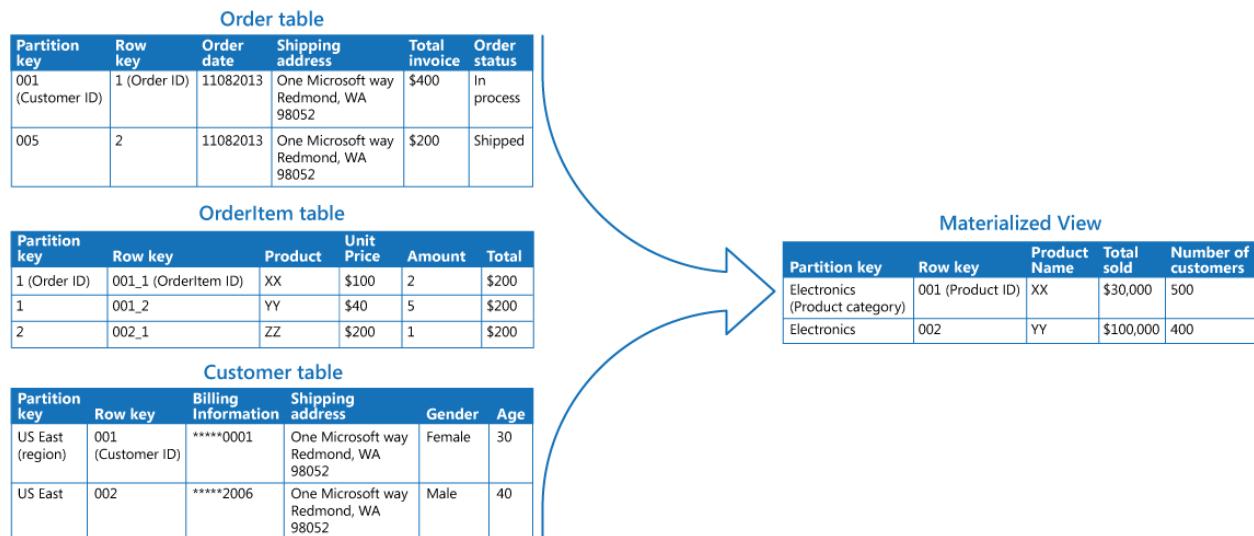
your microservices architecture or specific scenario, please consider having well-defined boundaries that align to [domain driven design \(DDD\)](#) and aggregate their data when requested.

This pattern isn't useful in the following situations:

- The source data is simple and easy to query.
- The source data changes very quickly, or can be accessed without using a view. In these cases, you should avoid the processing overhead of creating views.
- Consistency is a high priority. The views might not always be fully consistent with the original data.

Example

The following figure shows an example of using the Materialized View pattern to generate a summary of sales. Data in the Order, OrderItem, and Customer tables in separate partitions in an Azure storage account are combined to generate a view containing the total sales value for each product in the Electronics category, along with a count of the number of customers who made purchases of each item.



Creating this materialized view requires complex queries. However, by exposing the query result as a materialized view, users can easily obtain the results and use them directly or incorporate them in another query. The view is likely to be used in a reporting system or dashboard, and can be updated on a scheduled basis such as weekly.

Although this example uses Azure table storage, many relational database management systems also provide native support for materialized views.

Next steps

- [Data Consistency Primer](#). The summary information in a materialized view has to be maintained so that it reflects the underlying data values. As the data values change, it might not be practical to update the summary data in real time, and instead you'll have to adopt an eventually consistent approach. Summarizes the issues surrounding maintaining consistency over distributed data, and describes the benefits and tradeoffs of different consistency models.

Related resources

The following patterns might also be relevant when implementing this pattern:

- [Command and Query Responsibility Segregation \(CQRS\) pattern](#). Use to update the information in a materialized view by responding to events that occur when the underlying data values change.
- [Event Sourcing pattern](#). Use in conjunction with the CQRS pattern to maintain the information in a materialized view. When the data values are changed, the system can raise events that describe these changes and save them in an event store.
- [Index Table pattern](#). The data in a materialized view is typically organized by a primary key, but queries might need to retrieve information from this view by examining data in other fields. Use to create secondary indexes over data sets for data stores that don't support native secondary indexes.

Messaging Bridge pattern

Azure Service Bus

This article describes the Messaging Bridge pattern, which is a technique that you can use to integrate disparate systems that are built on top of different messaging infrastructures.

Context and problem

Many organizations and workloads can inadvertently have IT systems that use multiple messaging infrastructures like Microsoft Message Queueing (MSMQ), RabbitMQ, Azure Service Bus, and Amazon SQS. This problem can occur due to mergers, acquisitions, or due to extending current on-premises systems to cloud-hosted components for cost-effectiveness and the ease of maintenance.

Developers might address this challenge by modifying the systems being integrated to communicate by using HTTP-based web services. However, this approach has drawbacks, including:

- The systems must be modified by adding an HTTP client on one side and an HTTP request handler on the other. The systems must then be retested and redeployed.
- HTTP endpoints must be hosted, which adds complexity when you make web services secure and highly available.
- Frequent network connectivity problems that require custom-built retry mechanisms.

Solution

If the systems being integrated consist of components that communicate by exchanging messages, the Messaging Bridge pattern improves integration and mitigates drawbacks.

In this scenario, each system connects to one messaging infrastructure. To integrate across different messaging infrastructures, introduce a bridge component that connects to two or more messaging infrastructures at the same time. The bridge pulls messages from one and pushes them to the other without changing the payload.

The systems being integrated don't need to recognize the others or the bridge. The sender system is configured to send specific messages to a designated queue on its native messaging infrastructure. The bridge picks up those messages and forwards them

to another queue in a different messaging infrastructure where the receiver system picks them up.

Benefits

- The systems being integrated via the Messaging Bridge don't have to be modified. Ideally, the endpoints aren't aware that the messages are bridged.
- The integration is more reliable compared to the HTTP alternative due to the at-least-once message delivery mechanism guarantee.
- Migration scenarios can be more flexible. For example, endpoints can be migrated from one messaging infrastructure to another as the schedule permits instead of all at once.

Drawbacks

- Advanced features of one or both messaging technologies might not be available on the bridged route.
- The bridged route needs to consider both technologies' limitations. For example, the maximum message size might be 4 MB in MSMQ but only 64 KB in Azure Storage queues.

Issues and considerations

Consider the following points when implementing the Messaging Bridge pattern:

- If one of the integrated systems relies on distributed transactions, for example Microsoft Distributed Transaction Coordinator (DTC), for correctness, you must implement a deduplication mechanism in the bridge.
- If one of the systems being integrated doesn't use any messaging infrastructure and can't be modified, you can build the Messaging Bridge between the infrastructure that's used by the other system and a SQL Server-emulated queue. The legacy system can send messages by using the [change data capture](#) feature for SQL Server to push its changes to a dedicated queue table. The bridge can forward these messages to the actual messaging infrastructure.
- You can use a single queue in each messaging infrastructure, designated as the *bridging queue*. In this topology, configure the sending system to use that specific queue as the destination for message types that are sent to the other system. You can also use multiple pairs of queues in each messaging infrastructure, so the sender is unaware of the bridge. A *shadow queue* is created for each destination

queue in the destination system's messaging infrastructure. The bridge forwards messages between the shadow queues and their counterparts.

- In order to meet the desired availability service-level agreements (SLAs), you might need to scale out the Messaging Bridge by using the [Competing consumers](#) approach.
- Regular message-processing components use the [Retry pattern](#) to handle transient failures. The retry counter limit enables components to detect *poison* messages and remove them from the queue to unblock processing. The bridge might require a different retry policy to prevent falsely identifying messages as poison if an infrastructure failure occurs. You might use the [Circuit Breaker](#) pattern to pause forwarding.

When to use this pattern

Use the Messaging Bridge pattern when you need to:

- Integrate existing systems with minimal need for modification.
- Integrate legacy applications that can't use other messaging technologies.
- Extend existing on-premises applications with cloud-hosted components.
- Connect geo-distributed systems when the internet connection isn't stable.
- Migrate a single distributed system from one messaging infrastructure to another incrementally without the need to migrate the whole system in one effort.

This pattern might not be suitable if:

- At least one of the systems involved relies on a feature of one messaging infrastructure that isn't present in the other.
- Integration is synchronous in nature, and the initiating system requires immediate response.
- Integration has specific functional or nonfunctional requirements, such as security or privacy concerns.
- The volume of data for the integration exceeds the capacity of the messaging system or makes messaging an expensive solution to the problem.

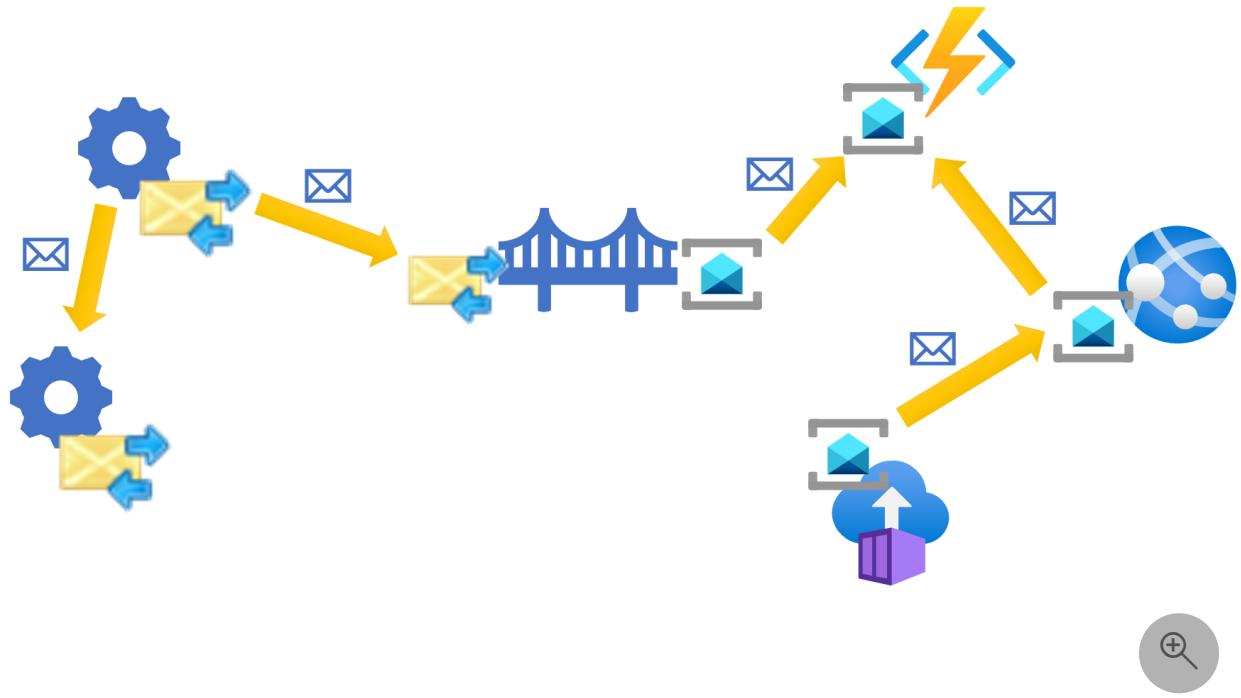
Example

There's an application written in a .NET framework for managing employee scheduling hosted on-premises. The application is well-structured with separate components communicating via MSMQ. The application works, and the workload team has no intention of rewriting it. A new consumer of the scheduling data needs to be built to

meet a business need, and the IT strategy calls for building new software as cloud-native applications to optimize the costs and delivery time.

The asynchronous queue-based architecture worked for the workload team in the past, so the team is going to use the same architectural approach but with the modern technology, Service Bus. The workload team doesn't want to introduce synchronous communication between the cloud and the on-premises deployment to mitigate the latency or unavailability of one affecting the other.

The team decides to use the Messaging Bridge pattern to connect the two systems. The pattern consists of two parts. One part receives messages from the existing MSMQ queue and forwards them to Service Bus. The other part takes messages from the Service Bus and forwards them to the existing MSMQ queue.



When the implementation team uses this approach, they utilize existing infrastructure in the existing application to integrate with the new components. The existing application isn't aware that the new components are hosted in Azure. Similarly, the new components communicate with the legacy application in the same way that they communicate between themselves, by sending Service Bus messages. The bridge forwards messages between the two systems.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Rob Bagby](#) | Principal Architecture Content Lead
- [Kyle Baley](#) | Software Engineer
- [Udi Dahan](#) | Founder & CEO of Particular Software
- [Chad Kittel](#) | Principal Software Engineer
- [Bryan Lamos](#) | Developer Relations
- [Szymon Pobiega](#) | Engineer

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- [Messaging Bridge pattern description](#) from the enterprise integration patterns community.
- Learn how to implement a [Messaging Bridge](#) in the Spring Java framework.
- [QPid bridge](#) can be used to bridge AMQP-enabled messaging technologies.
- The [NServiceBus Messaging Bridge](#) is a .NET implementation of a queue-to-queue bridge that supports a range of messaging infrastructures including MSMQ, Service Bus, and Azure Queue Storage.
- [NServiceBus.Router](#) is an open-source project that implements the Messaging Bridge pattern. It also allows bridging more than two technologies in a single instance and has advanced message-routing capabilities.

Related resources

- The [Competing Consumers](#) pattern ensures the implementation of the Messaging Bridge can handle the load.
- The [Retry](#) pattern allows the Messaging Bridge to handle transient failures.
- The [Circuit Breaker](#) pattern conserves resources when either side of the bridge experiences downtime.

Pipes and Filters pattern

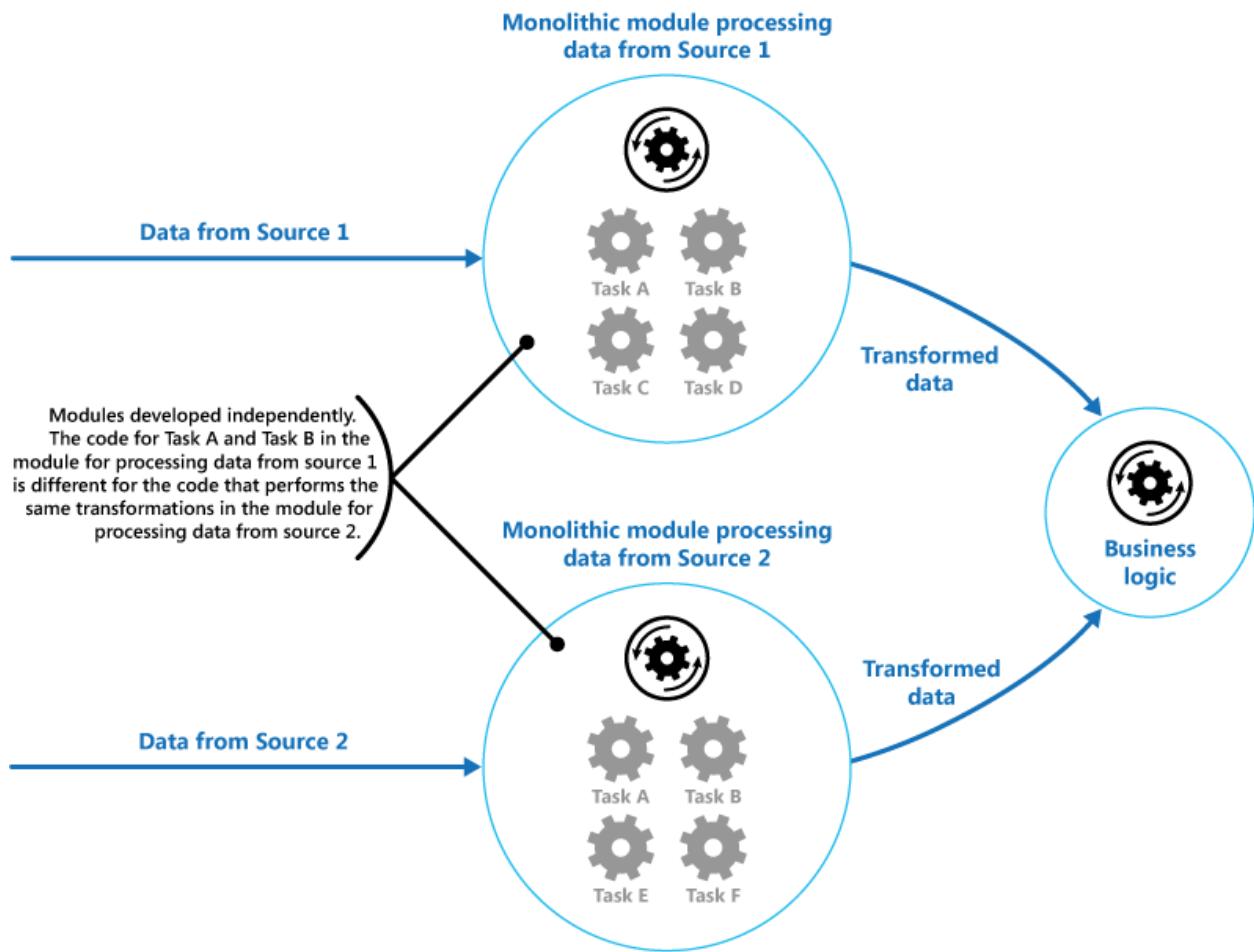
Azure DevOps

Decompose a task that performs complex processing into a series of separate elements that can be reused. Doing so can improve performance, scalability, and reusability by allowing task elements that perform the processing to be deployed and scaled independently.

Context and problem

An application can perform a variety of tasks that vary in complexity on the information it processes. A straightforward but inflexible approach to implementing an application is to perform this processing in a monolithic module. However, this approach is likely to reduce the opportunities for refactoring the code, optimizing it, or reusing it if parts of the same processing are required elsewhere in the application.

The following diagram illustrates the problems with processing data by using the monolithic approach. An application receives and processes data from two sources. The data from each source is processed by a separate module that performs a series of tasks to transform the data before passing the result to the business logic of the application.

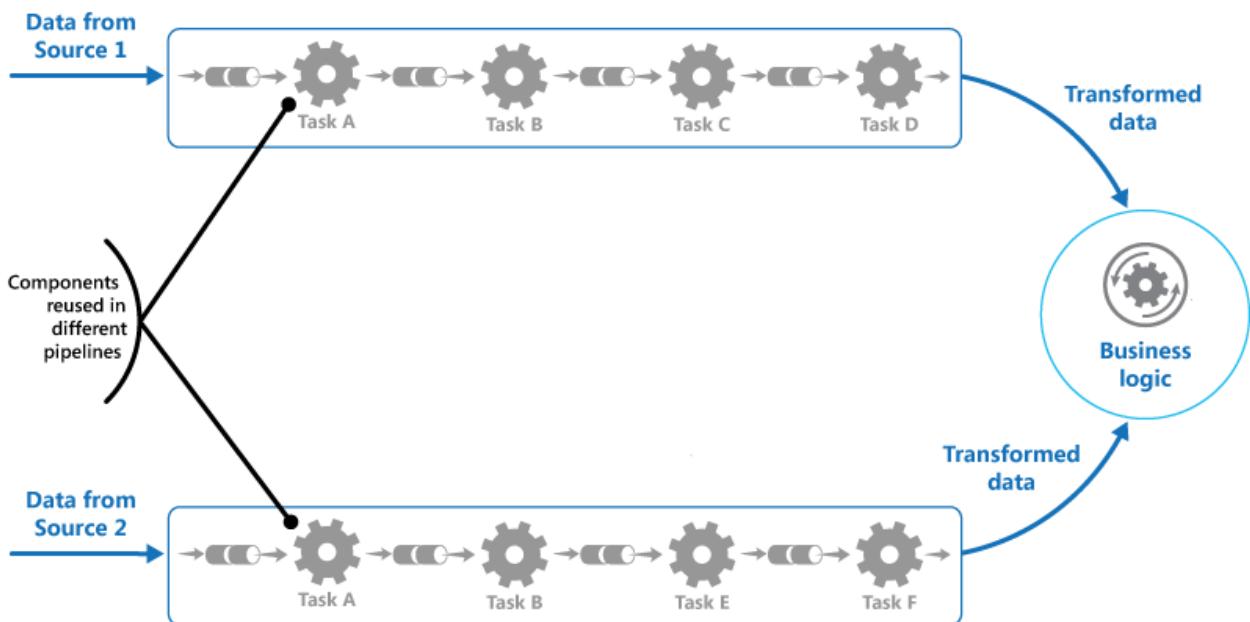


Some of the tasks that the monolithic modules perform are functionally similar, but the modules were designed separately. The code that implements the tasks is closely coupled in a module. Reuse and scalability weren't taken into account during development.

However, the processing tasks performed by each module, or the deployment requirements for each task, might change as business requirements are updated. Some tasks might be compute-intensive tasks that could benefit from running on powerful hardware. Other tasks might not require such expensive resources. Also, additional processing might be required in the future, or the order in which the tasks performed by the processing might change. A solution that addresses these problems and increases the possibilities for code reuse is required.

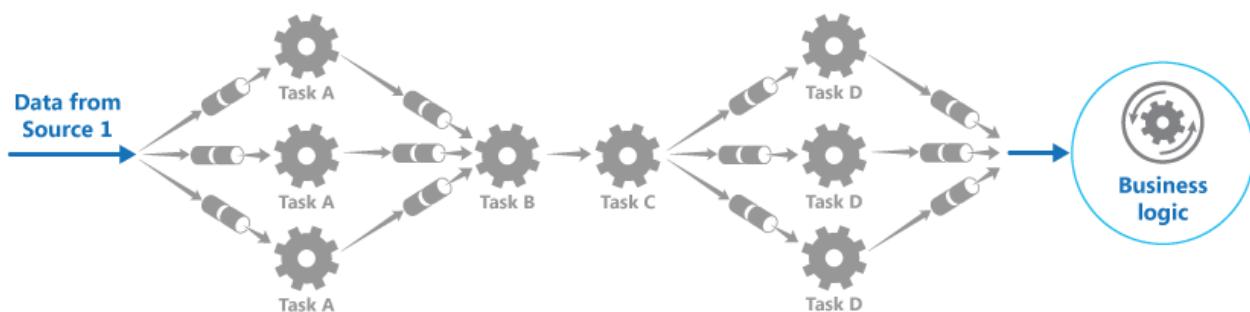
Solution

Break down the processing that's required for each stream into a set of separate components (or *filters*), each performing a single task. To achieve a standard format of the data that each component receives and sends, the filters can be combined in the pipeline. Doing so avoids code duplication and makes it easy to remove or replace components, or integrate additional components, if the processing requirements change. This diagram shows a solution that's implemented with pipes and filters:



The time it takes to process a single request depends on the speed of the slowest filters in the pipeline. One or more filters could be bottlenecks, especially if a high number of requests appear in a stream from a particular data source. A key advantage of the pipeline structure is that it provides opportunities for running parallel instances of slow filters, which enables the system to spread the load and improve throughput.

The filters that make up a pipeline can run on different machines, which enables them to be scaled independently and take advantage of the elasticity that many cloud environments provide. A filter that's computationally intensive can run on high-performance hardware, while other less-demanding filters can be hosted on less-expensive commodity hardware. The filters don't even need to be in the same datacenter or geographic location, so each element in a pipeline can run in an environment that's close to the resources it requires. This diagram shows an example applied to the pipeline for the data from Source 1:



If the input and output of a filter are structured as a stream, you can perform the processing for each filter in parallel. The first filter in the pipeline can start its work and output its results, which are passed directly to the next filter in the sequence before the first filter completes its work.

Another benefit of this model is the resiliency that it can provide. If a filter fails or the machine that it's running on is no longer available, the pipeline can reschedule the work

that the filter was doing and direct it to another instance of the component. Failure of a single filter doesn't necessarily result in failure of the entire pipeline.

Using the Pipes and Filters pattern together with the [Compensating Transaction pattern](#) is an alternative approach to implementing distributed transactions. You can break a distributed transaction into separate, compensable tasks, each of which can be implemented via a filter that also implements the Compensating Transaction pattern. You can implement the filters in a pipeline as separate hosted tasks that run close to the data that they maintain.

Issues and considerations

Consider the following points when you decide how to implement this pattern:

- **Complexity.** The increased flexibility that this pattern provides can also introduce complexity, especially if the filters in a pipeline are distributed across different servers.
- **Reliability.** Use an infrastructure that ensures that data flowing between filters in a pipeline won't be lost.
- **Idempotency.** If a filter in a pipeline fails after receiving a message and the work is rescheduled to another instance of the filter, part of the work might already be complete. If the work updates some aspect of the global state (like information stored in a database), a single update could be repeated. A similar issue might occur if a filter fails after it posts its results to the next filter in the pipeline but before indicating that it's completed its work successfully. In these cases, the same work could be repeated by another instance of the filter, causing the same results to be posted twice. This scenario could result in subsequent filters in the pipeline processing the same data twice. Therefore, filters in a pipeline should be designed to be idempotent. For more information, see [Idempotency Patterns ↗](#) on Jonathan Oliver's blog.
- **Repeated messages.** If a filter in a pipeline fails after it posts a message to the next stage of the pipeline, another instance of the filter might be run, and it would post a copy of the same message to the pipeline. This scenario could cause two instances of the same message to be passed to the next filter. To avoid this problem, the pipeline should detect and eliminate duplicate messages.

(!) Note

If you implement the pipeline by using message queues (like Azure Service Bus queues), the message queuing infrastructure might provide automatic duplicate message detection and removal.

- **Context and state.** In a pipeline, each filter essentially runs in isolation and shouldn't make any assumptions about how it was invoked. Therefore, each filter should be provided with sufficient context to perform its work. This context could include a significant amount of state information.

When to use this pattern

Use this pattern when:

- The processing required by an application can easily be broken down into a set of independent steps.
- The processing steps performed by an application have different scalability requirements.

ⓘ Note

You can group filters that should scale together in the same process. For more information, see the [Compute Resource Consolidation pattern](#).

- You require the flexibility to allow reordering of the processing steps that are performed by an application, or to allow the capability to add and remove steps.
- The system can benefit from distributing the processing for steps across different servers.
- You need a reliable solution that minimizes the effects of failure in a step while data is being processed.

This pattern might not be useful when:

- The processing steps performed by an application aren't independent, or they have to be performed together as part of a single transaction.
- The amount of context or state information that's required by a step makes this approach inefficient. You might be able to persist state information to a database, but don't use this strategy if the extra load on the database causes excessive contention.

Example

You can use a sequence of message queues to provide the infrastructure that's required to implement a pipeline. An initial message queue receives unprocessed messages. A component that's implemented as a filter task listens for a message on this queue, performs its work, and then posts the transformed message to the next queue in the sequence. Another filter task can listen for messages on this queue, process them, post the results to another queue, and so on, until the fully transformed data appears in the final message in the queue. This diagram illustrates a pipeline that uses message queues:



If you're building a solution on Azure, you can use Service Bus queues to provide a reliable and scalable queuing mechanism. The `ServiceBusPipeFilter` class shown in the following C# code demonstrates how you can implement a filter that receives input messages from a queue, processes the messages, and posts the results to another queue.

ⓘ Note

The `ServiceBusPipeFilter` class is defined in the `PipesAndFilters.Shared` project, which is available on [GitHub](#).

C#

```
public class ServiceBusPipeFilter
{
    ...
    private readonly string inQueuePath;
    private readonly string outQueuePath;
    ...
    private QueueClient inQueue;
    private QueueClient outQueue;
    ...

    public ServiceBusPipeFilter(..., string inQueuePath, string outQueuePath =
        null)
    {
        ...
        this.inQueuePath = inQueuePath;
        this.outQueuePath = outQueuePath;
    }
}
```

```
}

public void Start()
{
    ...
    // Create the outbound filter queue if it doesn't exist.
    ...
    this.outQueue = QueueClient.CreateFromConnectionString(...);

    ...
    // Create the inbound and outbound queue clients.
    this.inQueue = QueueClient.CreateFromConnectionString(...);
}

public void OnPipeFilterMessageAsync(
    Func<BrokeredMessage, Task<BrokeredMessage>> asyncFilterTask, ...)
{
    ...

    this.inQueue.OnMessageAsync(
        async (msg) =>
    {
        ...
        // Process the filter and send the output to the
        // next queue in the pipeline.
        var outMessage = await asyncFilterTask(msg);

        // Send the message from the filter processor
        // to the next queue in the pipeline.
        if (outQueue != null)
        {
            await outQueue.SendAsync(outMessage);
        }

        // Note: There's a chance that the same message could be sent twice
        // or that a message could be processed by an upstream or downstream
        // filter at the same time.
        // This would happen in a situation where processing of a message was
        // completed, it was sent to the next pipe/queue, and it then failed
        // to complete when using the PeekLock method.
        // In a real-world implementation, you should consider idempotent
        message
            // processing and concurrency.
        },
        options);
    }
}

public async Task Close(TimeSpan timespan)
{
    // Pause the processing threads.
    this.pauseProcessingEvent.Reset();

    // There's no clean approach for waiting for the threads to complete
    // the processing. This example simply stops any new processing, waits
    // for the existing thread to complete, closes the message pump,
}
```

```
// and finally returns.  
Thread.Sleep(timespan);  
  
    this.inQueue.Close();  
    ...  
}  
  
...  
}
```

The `Start` method in the `ServiceBusPipeFilter` class connects to a pair of input and output queues, and the `Close` method disconnects from the input queue. The `OnPipeFilterMessageAsync` method performs the actual processing of messages, and the `asyncFilterTask` parameter of this method specifies the processing to be performed. The `OnPipeFilterMessageAsync` method waits for incoming messages on the input queue, runs the code specified by the `asyncFilterTask` parameter over each message as it arrives, and posts the results to the output queue. The queues are specified by the constructor.

The sample solution implements filters in a set of worker roles. Each worker role can be scaled independently, depending on the complexity of the business processing that it performs or the resources that are required for processing. Additionally, multiple instances of each worker role can be run in parallel to improve throughput.

The following code shows an Azure worker role named `PipeFilterARoleEntry`, which is defined in the `PipeFilterA` project in the sample solution.

```
C#  
  
public class PipeFilterARoleEntry : RoleEntryPoint  
{  
    ...  
    private ServiceBusPipeFilter pipeFilterA;  
  
    public override bool OnStart()  
    {  
        ...  
        this.pipeFilterA = new ServiceBusPipeFilter(  
            ...,  
            Constants.QueueAPath,  
            Constants.QueueBPath);  
  
        this.pipeFilterA.Start();  
        ...  
    }  
  
    public override void Run()  
    {
```

```

    this.pipeFilterA.OnPipeFilterMessageAsync(async (msg) =>
{
    // Clone the message and update it.
    // Properties set by the broker (Deliver count, enqueue time, ...)
    // aren't cloned and must be copied over if required.
    var newMsg = msg.Clone();

    await Task.Delay(500); // DOING WORK

    Trace.TraceInformation("Filter A processed message:{0} at {1}",
        msg.MessageId, DateTime.UtcNow);

    newMsg.Properties.Add(Constants.FilterAMessageKey, "Complete");

    return newMsg;
});

...
}

...
}

```

This role contains a `ServiceBusPipeFilter` object. The `OnStart` method in the role connects to the queues that receive input messages and post output messages. (The names of the queues are defined in the `Constants` class.) The `Run` method invokes the `OnPipeFilterMessageAsync` method to perform processing on each message that's received. (In this example, the processing is simulated by waiting for a short time.) When processing is complete, a new message is constructed that contains the results (in this case, a custom property is added to the input message), and this message is posted to the output queue.

The sample code contains another worker role named `PipeFilterBRoleEntry`. It's in the `PipeFilterB` project. This role is similar to `PipeFilterARoleEntry`, but it performs different processing in the `Run` method. In the example solution, these two roles are combined to construct a pipeline. The output queue for the `PipeFilterARoleEntry` role is the input queue for the `PipeFilterBRoleEntry` role.

The sample solution also provides two other roles named `InitialSenderRoleEntry` (in the `InitialSender` project) and `FinalReceiverRoleEntry` (in the `FinalReceiver` project). The `InitialSenderRoleEntry` role provides the initial message in the pipeline. The `OnStart` method connects to a single queue, and the `Run` method posts a method to that queue. The queue is the input queue that's used by the `PipeFilterARoleEntry` role, so posting a message to it causes the message to be received and processed by the

`PipeFilterARoleEntry` role. The processed message then passes through the `PipeFilterBRoleEntry` role.

The input queue for the `FinalReceiveRoleEntry` role is the output queue for the `PipeFilterBRoleEntry` role. The `Run` method in the `FinalReceiveRoleEntry` role, shown in the following code, receives the message and performs some final processing. It then writes the values of the custom properties added by the filters in the pipeline to the trace output.

C#

```
public class FinalReceiverRoleEntry : RoleEntryPoint
{
    ...
    // Final queue/pipe in the pipeline to process data from.
    private ServiceBusPipeFilter queueFinal;

    public override bool OnStart()
    {
        ...
        // Set up the queue.
        this.queueFinal = new
ServiceBusPipeFilter(...,Constants.QueueFinalPath);
        this.queueFinal.Start();
        ...
    }

    public override void Run()
    {
        this.queueFinal.OnPipeFilterMessageAsync(
            async (msg) =>
        {
            await Task.Delay(500); // DOING WORK

            // The pipeline message was received.
            Trace.TraceInformation(
                "Pipeline Message Complete - FilterA:{0} FilterB:{1}",
                msg.Properties[Constants.FilterAMessageKey],
                msg.Properties[Constants.FilterBMessageKey]);

            return null;
        });
        ...
    }
    ...
}
```

Next steps

You might find the following resources helpful when you implement this pattern:

- [A sample that demonstrates this pattern, on GitHub ↗](#)
- [Idempotency patterns ↗](#), on Jonathan Oliver's blog

Related resources

The following patterns might also be relevant when you implement this pattern:

- [Competing Consumers pattern](#). A pipeline can contain multiple instances of one or more filters. This approach is useful for running parallel instances of slow filters. It enables the system to spread the load and improve throughput. Each instance of a filter competes for input with the other instances, but two instances of a filter shouldn't be able to process the same data. This article explains the approach.
- [Compute Resource Consolidation pattern](#). It might be possible to group filters that should scale together into a single process. This article provides more information about the benefits and tradeoffs of this strategy.
- [Compensating Transaction pattern](#). You can implement a filter as an operation that can be reversed, or that has a compensating operation that restores the state to a previous version if there's a failure. This article explains how you can implement this pattern to maintain or achieve eventual consistency.

Priority Queue pattern

Azure Service Bus

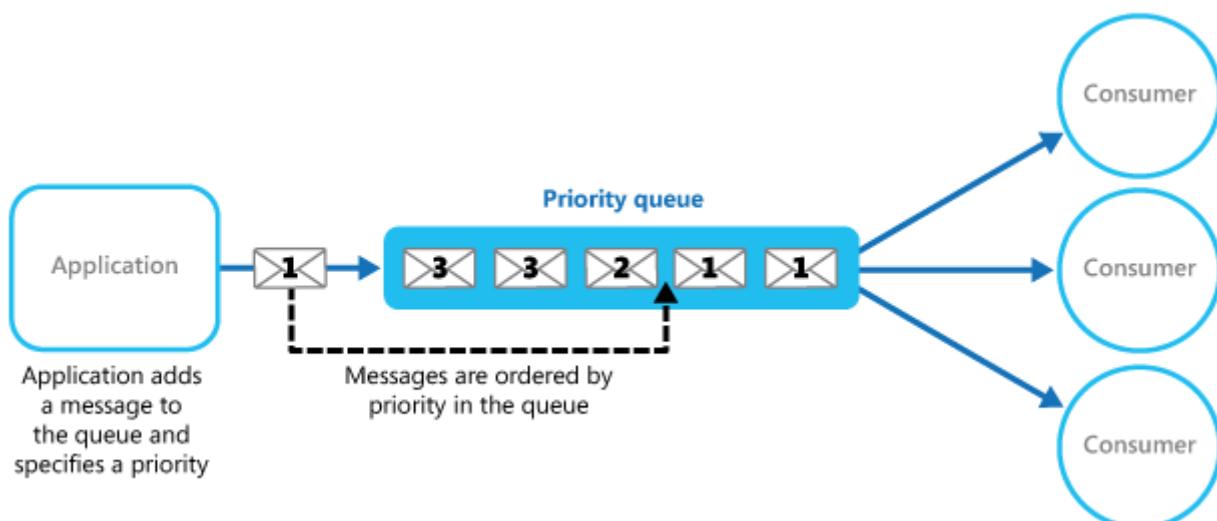
Prioritize requests sent to services so that requests with a higher priority are received and processed more quickly than those with a lower priority. This pattern is useful in applications that offer different service level guarantees to individual clients.

Context and problem

Applications can delegate specific tasks to other services, for example, to perform background processing or to integrate with other applications or services. In the cloud, a message queue is typically used to delegate tasks to background processing. In many cases, the order in which requests are received by a service isn't important. In some cases, though, it's necessary to prioritize specific requests. These requests should be processed earlier than lower priority requests that were previously sent by the application.

Solution

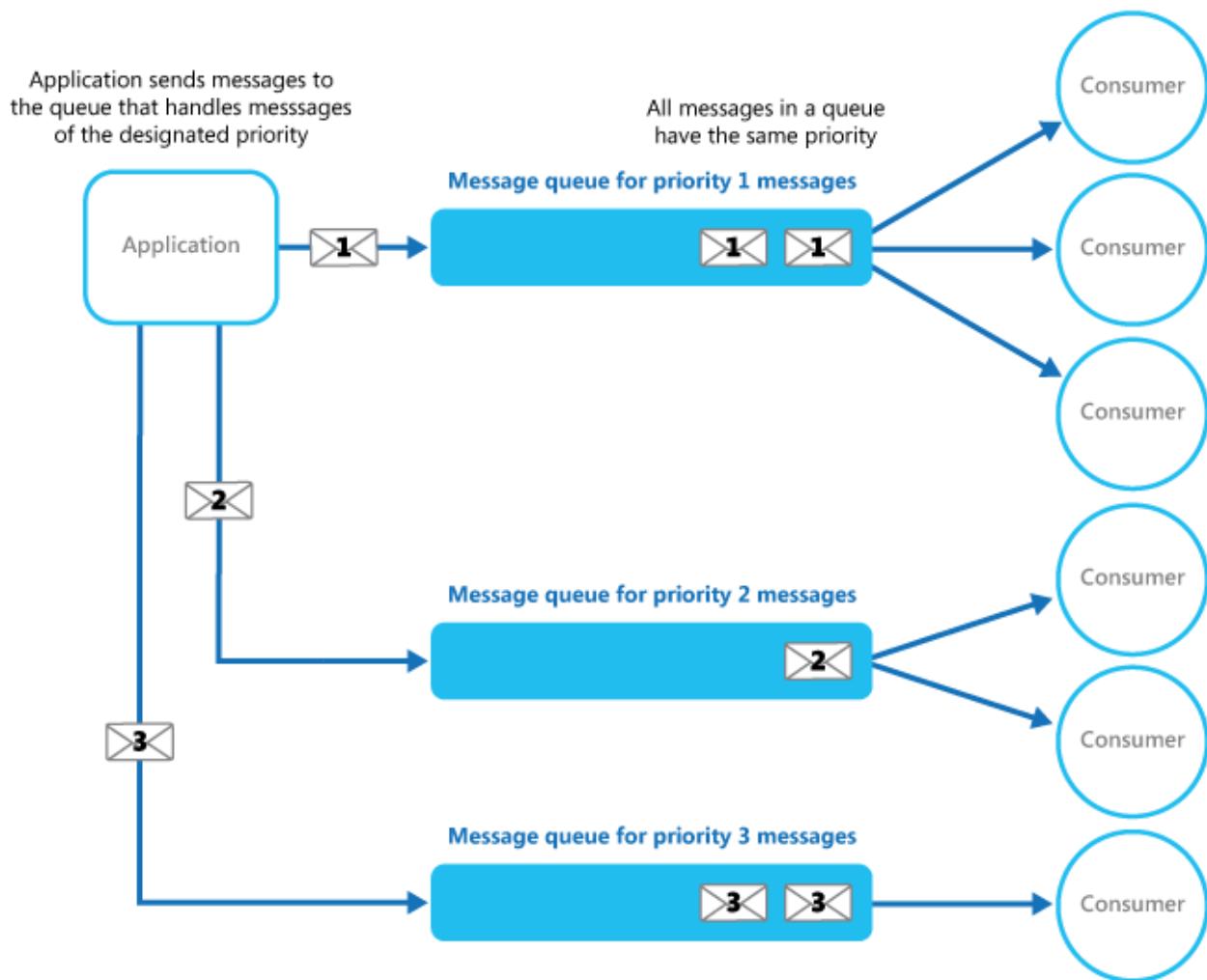
A queue usually is a first-in, first-out (FIFO) structure, and consumers typically receive messages in the same order that they're posted to the queue. However, some message queues support priority messaging. The application that's posting a message can assign a priority. The messages in the queue are automatically reordered so that those that have a higher priority are received before those that have a lower priority. This diagram illustrates the process:



(!) Note

Most message queue implementations support multiple consumers. (See the [Competing Consumers pattern](#).) The number of consumer processes can be scaled up and down based on demand.

In systems that don't support priority-based message queues, an alternative solution is to maintain a separate queue for each priority. The application is responsible for posting messages to the appropriate queue. Each queue can have a separate pool of consumers. Higher priority queues can have a larger pool of consumers that run on faster hardware than lower priority queues. This diagram illustrates the use of separate message queues for each priority:



A variation on this strategy is to implement a single pool of consumers that check for messages on high priority queues first, and only after that start to fetch messages from lower priority queues. There are some semantic differences between a solution that uses a single pool of consumer processes (either with a single queue that supports messages that have different priorities or with multiple queues that each handle messages of a single priority), and a solution that uses multiple queues with a separate pool for each queue.

In the single-pool approach, higher priority messages are always received and processed before lower priority messages. In theory, low priority messages could be continually superseded and might never be processed. In the multiple pool approach, lower priority messages are always processed, but not as quickly as higher priority messages (depending on the relative size of the pools and the resources that are available for them).

Using a priority-queuing mechanism can provide the following advantages:

- It allows applications to meet business requirements that require the prioritization of availability or performance, such as offering different levels of service to different groups of customers.
- It can help to minimize operational costs. If you use the single-queue approach, you can scale back the number of consumers if you need to. High priority messages are still processed first (although possibly more slowly), and lower priority messages might be delayed for longer. If you implement the multiple message queue approach with separate pools of consumers for each queue, you can reduce the pool of consumers for lower priority queues. You can even suspend processing for some very low priority queues by stopping all the consumers that listen for messages on those queues.
- The multiple message queue approach can help maximize application performance and scalability by partitioning messages based on processing requirements. For example, you can prioritize critical tasks so that they're handled by receivers that run immediately, and less important background tasks can be handled by receivers that are scheduled to run at times that are less busy.

Considerations

Consider the following points when you decide how to implement this pattern:

- Define the priorities in the context of the solution. For example, a *high priority* message could be defined as a message that should be processed within 10 seconds. Identify the requirements for handling high priority items, and the resources that need to be allocated to meet your criteria.
- Decide whether all high priority items must be processed before any lower priority items. If the messages are processed by a single pool of consumers, you have to provide a mechanism that can preempt and suspend a task that's handling a low priority message if a higher priority message enters the queue.

- In the multiple queue approach, when you use a single pool of consumer processes that listen on all queues rather than a dedicated consumer pool for each queue, the consumer must apply an algorithm that ensures it always services messages from higher priority queues before messages from lower priority queues.
- Monitor the processing speed on high and low priority queues to ensure that messages in those queues are processed at the expected rates.
- If you need to guarantee that low priority messages will be processed, implement the multiple message queue approach with multiple pools of consumers. Alternatively, in a queue that supports message prioritization, you can dynamically increase the priority of a queued message as it ages. However, this approach depends on the message queue providing this feature.
- The strategy of using separate queues based on message priority is recommended for systems that have a few well-defined priorities.
- The system can logically determine message priorities. For example, rather than having explicit high and low priority messages, you could designate messages as "paying customer" or "non-paying customer." Your system could then allocate more resources to processing messages from paying customers.
- There might be a financial and processing cost associated with checking a queue for a message. For instance, some commercial messaging systems charge a small fee each time a message is posted or retrieved, and each time a queue is queried for messages. This cost increases when you check multiple queues.
- You can dynamically adjust the size of a pool of consumers based on the length of the queue that the pool is servicing. For more information, see [Autoscaling guidance](#).

When to use this pattern

This pattern is useful in scenarios where:

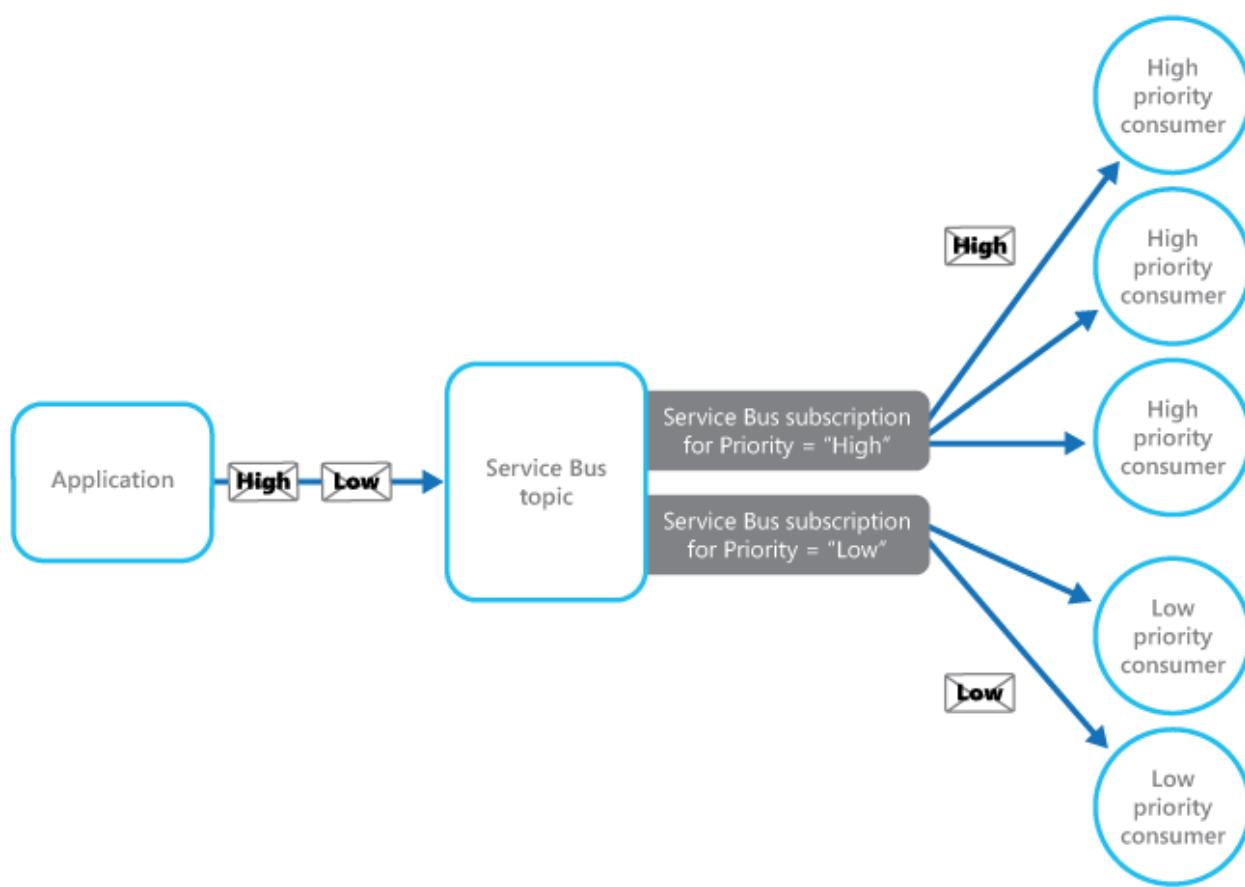
- The system must handle multiple tasks that have different priorities.
- Different users or tenants should be served with different priorities.

Example

Azure doesn't provide a queuing mechanism that natively supports automatic prioritization of messages via sorting. However, it does provide Azure Service Bus topics,

Service Bus subscriptions that support a queuing mechanism that provides message filtering, and a range of flexible capabilities that make Azure ideal for most priority-queue implementations.

An Azure solution can implement a Service Bus topic that an application can post messages to, just as it would post them to a queue. Messages can contain metadata in the form of application-defined custom properties. You can associate Service Bus subscriptions with the topic, and the subscriptions can filter messages based on their properties. When an application sends a message to a topic, the message is directed to the appropriate subscription, where a consumer can read it. Consumer processes can retrieve messages from a subscription by using the same semantics that they would use with a message queue. (A subscription is a logical queue.) This diagram shows how to implement a priority queue by using Service Bus topics and subscriptions:



In the preceding diagram, the application creates several messages and assigns a custom property called `Priority` in each message. `Priority` has a value of `High` or `Low`. The application posts these messages to a topic. The topic has two associated subscriptions that filter messages based on the `Priority` property. One subscription accepts messages with the `Priority` property set to `High`. The other accepts messages with the `Priority` property set to `Low`. A pool of consumers reads messages from each subscription. The high priority subscription has a larger pool, and these consumers might be running on more powerful computers that have more available resources than the computers for the low priority pool.

There's nothing special about the designation of high and low priority messages in this example. They're simply labels that are specified as properties in each message. They're used to direct messages to a specific subscription. If additional priorities are needed, it's relatively easy to create more subscriptions and pools of consumer processes to handle those priorities.

The PriorityQueue solution on [GitHub](#) is based on this approach. This solution contains Azure Function projects named `PriorityQueueConsumerHigh` and `PriorityQueueConsumerLow`. These Azure Function projects integrate with Service Bus via triggers and bindings. They connect to different subscriptions that are defined in `ServiceBusTrigger` and react to the incoming messages.

C#

```
public static class PriorityQueueConsumerHighFn
{
    [FunctionName("HighPriorityQueueConsumerFunction")]
    public static void Run(
        [ServiceBusTrigger("messages", "highPriority", Connection =
"ServiceBusConnection")]string highPriorityMessage,
        ILogger log)
    {
        log.LogInformation($"C# ServiceBus topic trigger function processed
message: {highPriorityMessage}");
    }
}
```

As an administrator, you can configure how many instances the functions on Azure App Service can scale out to. You can do that by configuring the **Enforce Scale Out Limit** option from the Azure portal, setting a maximum scale-out limit for each function. You typically need to have more instances of the `PriorityQueueConsumerHigh` function than the `PriorityQueueConsumerLow` function. This configuration ensures that high priority messages are read from the queue more quickly than low priority messages.

Another project, `PriorityQueueSender`, contains a time-triggered Azure function that's configured to run every 30 seconds. This function integrates with Service Bus via an output binding and sends batches of low and high priority messages to an `IAsyncCollector` object. When the function posts messages to the topic that's associated with the subscriptions used by the `PriorityQueueConsumerHigh` and `PriorityQueueConsumerLow` functions, it specifies the priority by using the `Priority` custom property, as shown here:

C#

```

public static class PriorityQueueSenderFn
{
    [FunctionName("PriorityQueueSenderFunction")]
    public static async Task Run(
        [TimerTrigger("0,30 * * * *")] TimerInfo myTimer,
        [ServiceBus("messages", Connection = "ServiceBusConnection")]
    IAsyncCollector<ServiceBusMessage> collector)
    {
        for (int i = 0; i < 10; i++)
        {
            var messageId = Guid.NewGuid().ToString();
            var lpMessage = new ServiceBusMessage() { MessageId = messageId };
            lpMessage.ApplicationProperties["Priority"] = Priority.Low;
            lpMessage.Body = BinaryData.FromString($"Low priority message
with Id: {messageId}");
            await collector.AddAsync(lpMessage);

            messageId = Guid.NewGuid().ToString();
            var hpMessage = new ServiceBusMessage() { MessageId = messageId };
            hpMessage.ApplicationProperties["Priority"] = Priority.High;
            hpMessage.Body = BinaryData.FromString($"High priority message
with Id: {messageId}");
            await collector.AddAsync(hpMessage);
        }
    }
}

```

Next steps

The following resources might be helpful to you when you implement this pattern:

- A sample that demonstrates this pattern, on [GitHub](#).
- [Asynchronous messaging primer](#). A consumer service that processes a request might need to send a reply to the instance of the application that posted the request. This article provides information about the strategies that you can use to implement request/response messaging.
- [Autoscaling guidance](#). You can sometimes scale the size of the pool of consumer processes that are handling a queue based on the length of the queue. This strategy can help you improve performance, especially for pools that handle high priority messages.

Related resources

The following patterns might be helpful to you when you implement this pattern:

- [Competing Consumers pattern](#). To increase the throughput of the queues, you can implement multiple consumers that listen on the same queue and process tasks in parallel. These consumers compete for messages, but only one should be able to process each message. This article provides more information on the benefits and disadvantages of implementing this approach.
- [Throttling pattern](#). You can implement throttling by using queues. You can use priority messaging to ensure that requests from critical applications, or applications that are run by high-value customers, are given priority over requests from less important applications.

Publisher-Subscriber pattern

Azure Event Grid Azure Event Hubs Azure Service Bus

Enable an application to announce events to multiple interested consumers asynchronously, without coupling the senders to the receivers.

Also called: Pub/sub messaging

Context and problem

In cloud-based and distributed applications, components of the system often need to provide information to other components as events happen.

Asynchronous messaging is an effective way to decouple senders from consumers, and avoid blocking the sender to wait for a response. However, using a dedicated message queue for each consumer does not effectively scale to many consumers. Also, some of the consumers might be interested in only a subset of the information. How can the sender announce events to all interested consumers without knowing their identities?

Solution

Introduce an asynchronous messaging subsystem that includes the following:

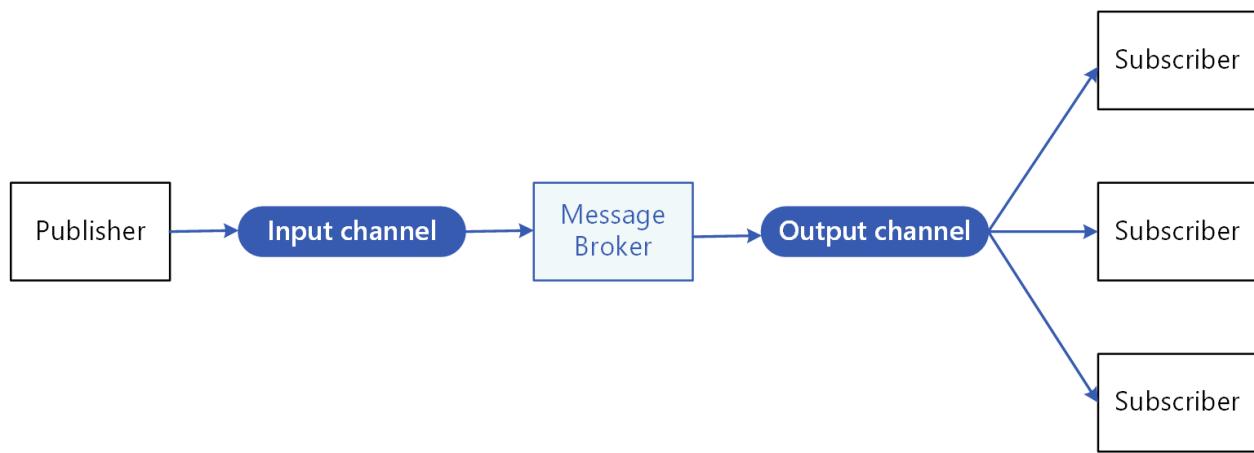
- An input messaging channel used by the sender. The sender packages events into messages, using a known message format, and sends these messages via the input channel. The sender in this pattern is also called the *publisher*.

(!) Note

A *message* is a packet of data. An *event* is a message that notifies other components about a change or an action that has taken place.

- One output messaging channel per consumer. The consumers are known as *subscribers*.
- A mechanism for copying each message from the input channel to the output channels for all subscribers interested in that message. This operation is typically handled by an intermediary such as a message broker or event bus.

The following diagram shows the logical components of this pattern:



Pub/sub messaging has the following benefits:

- It decouples subsystems that still need to communicate. Subsystems can be managed independently, and messages can be properly managed even if one or more receivers are offline.
- It increases scalability and improves responsiveness of the sender. The sender can quickly send a single message to the input channel, then return to its core processing responsibilities. The messaging infrastructure is responsible for ensuring messages are delivered to interested subscribers.
- It improves reliability. Asynchronous messaging helps applications continue to run smoothly under increased loads and handle intermittent failures more effectively.
- It allows for deferred or scheduled processing. Subscribers can wait to pick up messages until off-peak hours, or messages can be routed or processed according to a specific schedule.
- It enables simpler integration between systems using different platforms, programming languages, or communication protocols, as well as between on-premises systems and applications running in the cloud.
- It facilitates asynchronous workflows across an enterprise.
- It improves testability. Channels can be monitored and messages can be inspected or logged as part of an overall integration test strategy.
- It provides separation of concerns for your applications. Each application can focus on its core capabilities, while the messaging infrastructure handles everything required to reliably route messages to multiple consumers.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- **Existing technologies.** It is strongly recommended to use available messaging products and services that support a publish-subscribe model, rather than building your own. In Azure, consider using [Service Bus](#), [Event Hubs](#) or [Event Grid](#). Other technologies that can be used for pub/sub messaging include Redis, RabbitMQ, and Apache Kafka.
- **Subscription handling.** The messaging infrastructure must provide mechanisms that consumers can use to subscribe to or unsubscribe from available channels.
- **Security.** Connecting to any message channel must be restricted by security policy to prevent eavesdropping by unauthorized users or applications.
- **Subsets of messages.** Subscribers are usually only interested in subset of the messages distributed by a publisher. Messaging services often allow subscribers to narrow the set of messages received by:
 - **Topics.** Each topic has a dedicated output channel, and each consumer can subscribe to all relevant topics.
 - **Content filtering.** Messages are inspected and distributed based on the content of each message. Each subscriber can specify the content it is interested in.
- **Wildcard subscribers.** Consider allowing subscribers to subscribe to multiple topics via wildcards.
- **Bi-directional communication.** The channels in a publish-subscribe system are treated as unidirectional. If a specific subscriber needs to send acknowledgment or communicate status back to the publisher, consider using the [Request/Reply Pattern](#). This pattern uses one channel to send a message to the subscriber, and a separate reply channel for communicating back to the publisher.
- **Message ordering.** The order in which consumer instances receive messages isn't guaranteed, and doesn't necessarily reflect the order in which the messages were created. Design the system to ensure that message processing is idempotent to help eliminate any dependency on the order of message handling.
- **Message priority.** Some solutions may require that messages are processed in a specific order. The [Priority Queue pattern](#) provides a mechanism for ensuring specific messages are delivered before others.
- **Poison messages.** A malformed message, or a task that requires access to resources that aren't available, can cause a service instance to fail. The system should prevent such messages being returned to the queue. Instead, capture and store the details of these messages elsewhere so that they can be analyzed if

necessary. Some message brokers, like Azure Service Bus, support this via their [dead-letter queue functionality](#).

- **Repeated messages.** The same message might be sent more than once. For example, the sender might fail after posting a message. Then a new instance of the sender might start up and repeat the message. The messaging infrastructure should implement duplicate message detection and removal (also known as deduping) based on message IDs in order to provide at-most-once delivery of messages. Alternatively, if using messaging infrastructure which doesn't deduplicate messages, make sure the [message processing logic is idempotent](#).
- **Message expiration.** A message might have a limited lifetime. If it isn't processed within this period, it might no longer be relevant and should be discarded. A sender can specify an expiration time as part of the data in the message. A receiver can examine this information before deciding whether to perform the business logic associated with the message.
- **Message scheduling.** A message might be temporarily embargoed and should not be processed until a specific date and time. The message should not be available to a receiver until this time.

When to use this pattern

Use this pattern when:

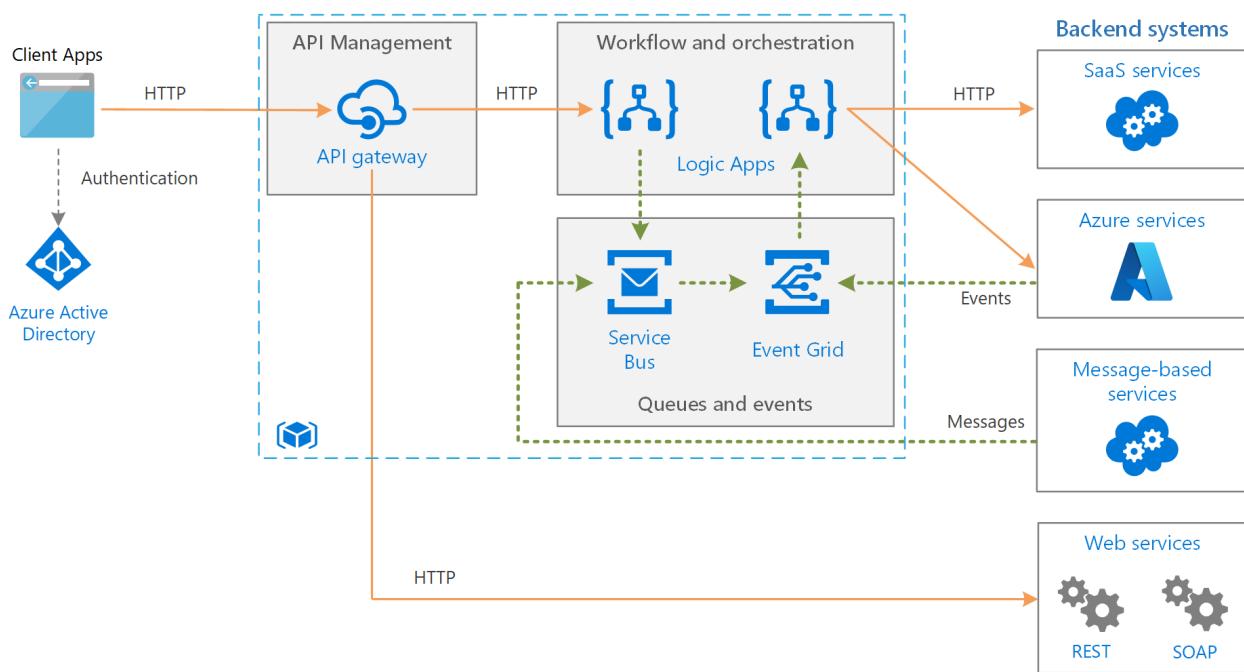
- An application needs to broadcast information to a significant number of consumers.
- An application needs to communicate with one or more independently-developed applications or services, which may use different platforms, programming languages, and communication protocols.
- An application can send information to consumers without requiring real-time responses from the consumers.
- The systems being integrated are designed to support an eventual consistency model for their data.
- An application needs to communicate information to multiple consumers, which may have different availability requirements or uptime schedules than the sender.

This pattern might not be useful when:

- An application has only a few consumers who need significantly different information from the producing application.
- An application requires near real-time interaction with consumers.

Example

The following diagram shows an enterprise integration architecture that uses Service Bus to coordinate workflows, and Event Grid to notify subsystems of events that occur. For more information, see [Enterprise integration on Azure using message queues and events](#).



Next steps

The following guidance might be relevant when implementing this pattern:

- [Choose between Azure services that deliver messages](#).
- [Asynchronous Messaging Primer](#). Message queues are an asynchronous communications mechanism. If a consumer service needs to send a reply to an application, it might be necessary to implement some form of response messaging. The Asynchronous Messaging Primer provides information on how to implement request/reply messaging using message queues.

The following patterns might be relevant when implementing this pattern:

- [Observer pattern](#). The Publish-Subscribe pattern builds on the Observer pattern by decoupling subjects from observers via asynchronous messaging.

- Message Broker pattern [↗](#). Many messaging subsystems that support a publish-subscribe model are implemented via a message broker.

This [blog post](#) [↗](#) describes different ways of handling messages that arrive out of order.

Related resources

- The [Event-driven architecture style](#) is an architecture style that uses pub/sub messaging.
- [Idempotent message processing](#)
- [Enterprise integration on Azure using message queues and events](#)

Queue-Based Load Leveling pattern

Azure Functions

Azure Service Bus

Use a queue that acts as a buffer between a task and a service it invokes in order to smooth intermittent heavy loads that can cause the service to fail or the task to time out. This can help to minimize the impact of peaks in demand on availability and responsiveness for both the task and the service.

Context and problem

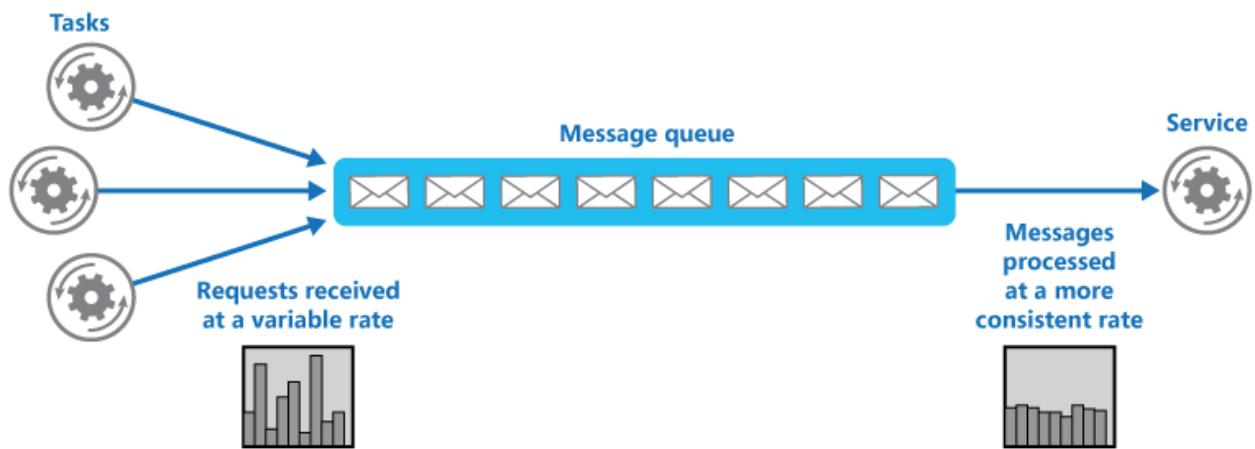
Many solutions in the cloud involve running tasks that invoke services. In this environment, if a service is subjected to intermittent heavy loads, it can cause performance or reliability issues.

A service could be part of the same solution as the tasks that use it, or it could be a third-party service providing access to frequently used resources such as a cache or a storage service. If the same service is used by a number of tasks running concurrently, it can be difficult to predict the volume of requests to the service at any time.

A service might experience peaks in demand that cause it to overload and be unable to respond to requests in a timely manner. Flooding a service with a large number of concurrent requests can also result in the service failing if it's unable to handle the contention these requests cause.

Solution

Refactor the solution and introduce a queue between the task and the service. The task and the service run asynchronously. The task posts a message containing the data required by the service to a queue. The queue acts as a buffer, storing the message until it's retrieved by the service. The service retrieves the messages from the queue and processes them. Requests from a number of tasks, which can be generated at a highly variable rate, can be passed to the service through the same message queue. This figure shows using a queue to level the load on a service.



The queue decouples the tasks from the service, and the service can handle the messages at its own pace regardless of the volume of requests from concurrent tasks. Additionally, there's no delay to a task if the service isn't available at the time it posts a message to the queue.

This pattern provides the following benefits:

- It can help to maximize availability because delays arising in services won't have an immediate and direct impact on the application, which can continue to post messages to the queue even when the service isn't available or isn't currently processing messages.
- It can help to maximize scalability because both the number of queues and the number of services can be varied to meet demand.
- It can help to control costs because the number of service instances deployed only have to be adequate to meet average load rather than the peak load.

Some services implement throttling when demand reaches a threshold beyond which the system could fail. Throttling can reduce the functionality available. You can implement load leveling with these services to ensure that this threshold isn't reached.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- It's necessary to implement application logic that controls the rate at which services handle messages to avoid overwhelming the target resource. Avoid passing spikes in demand to the next stage of the system. Test the system under load to ensure that it provides the required leveling, and adjust the number of queues and the number of service instances that handle messages to achieve this.

- Message queues are a one-way communication mechanism. If a task expects a reply from a service, it might be necessary to implement a mechanism that the service can use to send a response. For more information, see the [Asynchronous Messaging Primer](#).
- Be careful if you apply [autoscaling](#) to services that are listening for requests on the queue. This can result in increased contention for any resources that these services share and diminish the effectiveness of using the queue to level the load.
- Depending on the load of the service, you can run into a situation where you're effectively always trailing behind, where the system is always queuing up more requests than you're processing. The variability of incoming traffic to your application needs to be taken into consideration
- The pattern can lose information depending on the persistence of the Queue. If your queue crashes or drops information (due to system limits) there's a possibility that you don't have a guaranteed delivery. The behavior of your queue and system limits needs to be taken into consideration based on the needs of your solution.

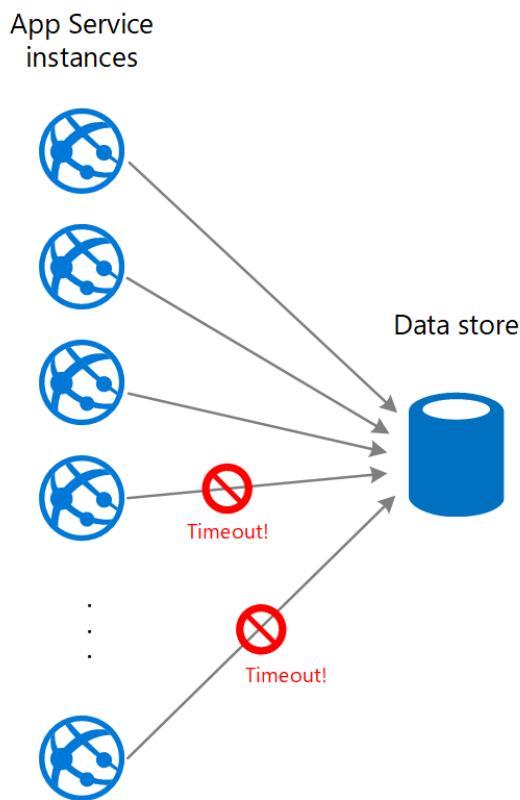
When to use this pattern

This pattern is useful to any application that uses services that are subject to overloading.

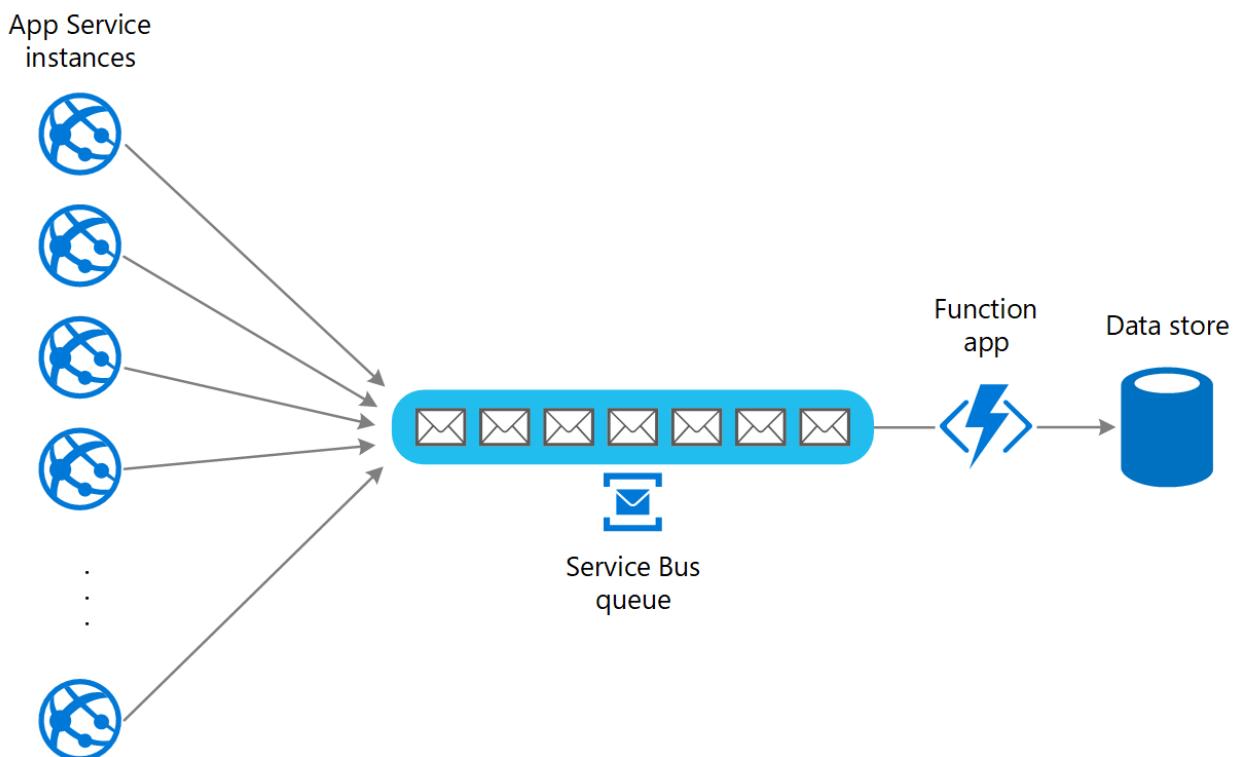
This pattern isn't useful if the application expects a response from the service with minimal latency.

Example

A web app writes data to an external data store. If a large number of instances of the web app run concurrently, the data store might be unable to respond to requests quickly enough, causing requests to time out, be throttled, or otherwise fail. The following diagram shows a data store being overwhelmed by a large number of concurrent requests from instances of an application.



To resolve this, you can use a queue to level the load between the application instances and the data store. An Azure Functions app reads the messages from the queue and performs the read/write requests to the data store. The application logic in the function app can control the rate at which it passes requests to the data store, to prevent the store from being overwhelmed. (Otherwise the function app will just re-introduce the same problem at the back end.)



Next steps

The following guidance might also be relevant when implementing this pattern:

- [Asynchronous Messaging Primer](#). Message queues are inherently asynchronous. It might be necessary to redesign the application logic in a task if it's adapted from communicating directly with a service to using a message queue. Similarly, it might be necessary to refactor a service to accept requests from a message queue. Alternatively, it might be possible to implement a proxy service, as described in the example.
- [Choose between Azure messaging services](#). Information about choosing a messaging and queuing mechanism in Azure applications.
- [Asynchronous message-based communication](#).

Related resources

- [Web-Queue-Worker architecture style](#). The web and worker are both stateless. Session state can be stored in a distributed cache. Any long-running work is done asynchronously by the worker. The worker can be triggered by messages on the queue, or run on a schedule for batch processing.

The following patterns might also be relevant when implementing this pattern:

- [Competing Consumers pattern](#). It might be possible to run multiple instances of a service, each acting as a message consumer from the load-leveling queue. You can use this approach to adjust the rate at which messages are received and passed to a service.
- [Throttling pattern](#). A simple way to implement throttling with a service is to use queue-based load leveling and route all requests to a service through a message queue. The service can process requests at a rate that ensures that resources required by the service aren't exhausted, and to reduce the amount of contention that could occur.

Rate Limiting pattern

Azure Service Bus

Azure Queue Storage

Azure Event Hubs

Many services use a [throttling pattern](#) to control the resources they consume, imposing limits on the rate at which other applications or services can access them. You can use a rate limiting pattern to help you avoid or minimize throttling errors related to these throttling limits and to help you more accurately predict throughput.

A rate limiting pattern is appropriate in many scenarios, but it is particularly helpful for large-scale repetitive automated tasks such as batch processing.

Context and problem

Performing large numbers of operations using a throttled service can result in increased traffic and throughput, as you'll need to both track rejected requests and then retry those operations. As the number of operations increases, a throttling limit may require multiple passes of resending data, resulting in a larger performance impact.

As an example, consider the following naive retry on error process for ingesting data into Azure Cosmos DB:

1. Your application needs to ingest 10,000 records into Azure Cosmos DB. Each record costs 10 Request Units (RUs) to ingest, requiring a total of 100,000 RUs to complete the job.
2. Your Azure Cosmos DB instance has 20,000 RUs provisioned capacity.
3. You send all 10,000 records to Azure Cosmos DB. 2,000 records are written successfully and 8,000 records are rejected.
4. You send the remaining 8,000 records to Azure Cosmos DB. 2,000 records are written successfully and 6,000 records are rejected.
5. You send the remaining 6,000 records to Azure Cosmos DB. 2,000 records are written successfully and 4,000 records are rejected.
6. You send the remaining 4,000 records to Azure Cosmos DB. 2,000 records are written successfully and 2,000 records are rejected.
7. You send the remaining 2,000 records to Azure Cosmos DB. All are written successfully.

The ingestion job completed successfully, but only after sending 30,000 records to Azure Cosmos DB even though the entire data set only consisted of 10,000 records.

There are additional factors to consider in the above example:

- Large numbers of errors can also result in additional work to log these errors and process the resulting log data. This naive approach will have handled 20,000 errors, and logging these errors may impose a processing, memory, or storage resource cost.
- Not knowing the throttling limits of the ingestion service, the naive approach has no way to set expectations for how long data processing will take. Rate limiting can allow you to calculate the time required for ingestion.

Solution

Rate limiting can reduce your traffic and potentially improve throughput by reducing the number of records sent to a service over a given period of time.

A service may throttle based on different metrics over time, such as:

- The number of operations (for example, 20 requests per second).
- The amount of data (for example, 2 GiB per minute).
- The relative cost of operations (for example, 20,000 RUs per second).

Regardless of the metric used for throttling, your rate limiting implementation will involve controlling the number and/or size of operations sent to the service over a specific time period, optimizing your use of the service while not exceeding its throttling capacity.

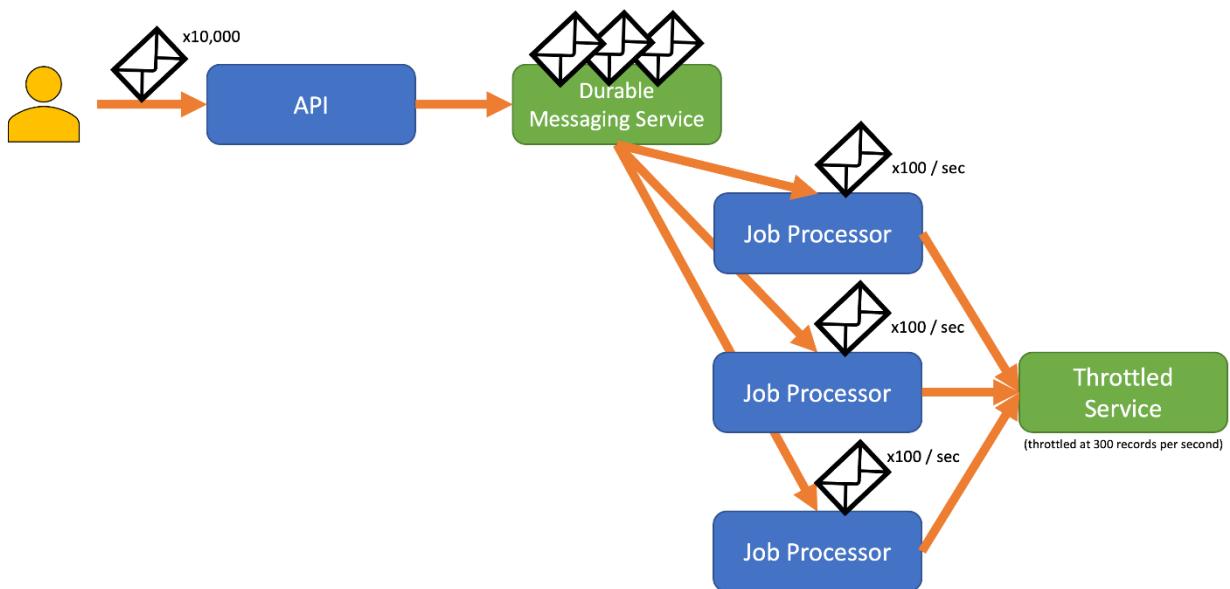
In scenarios where your APIs can handle requests faster than any throttled ingestion services allow, you'll need to manage how quickly you can use the service. However, only treating the throttling as a data rate mismatch problem, and simply buffering your ingestion requests until the throttled service can catch up, is risky. If your application crashes in this scenario, you risk losing any of this buffered data.

To avoid this risk, consider sending your records to a durable messaging system that *can* handle your full ingestion rate. (Services such as Azure Event Hubs can handle millions of operations per second). You can then use one or more job processors to read the records from the messaging system at a controlled rate that is within the throttled service's limits. Submitting records to the messaging system can save internal memory by allowing you to dequeue only the records that can be processed during a given time interval.

Azure provides several durable messaging services that you can use with this pattern, including:

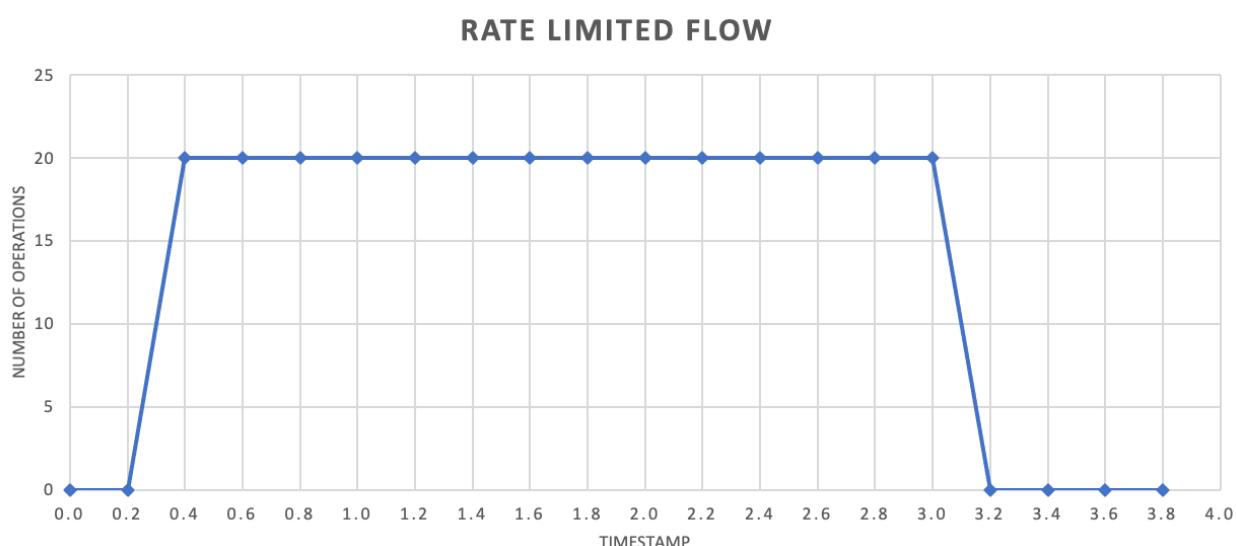
- [Azure Service Bus](#)
- [Azure Queue Storage](#)

- Azure Event Hubs ↗



When you're sending records, the time period you use for releasing records may be more granular than the period the service throttles on. Systems often set throttles based on timespans you can easily comprehend and work with. However, for the computer running a service, these timeframes may be very long compared to how fast it can process information. For instance, a system might throttle per second or per minute, but commonly the code is processing on the order of nanoseconds or milliseconds.

While not required, it's often recommended to send smaller amounts of records more frequently to improve throughput. So rather than trying to batch things up for a release once a second or once a minute, you can be more granular than that to keep your resource consumption (memory, CPU, network, etc.) flowing at a more even rate, preventing potential bottlenecks due to sudden bursts of requests. For example, if a service allows 100 operations per second, the implementation of a rate limiter may even out requests by releasing 20 operations every 200 milliseconds, as shown in the following graph.

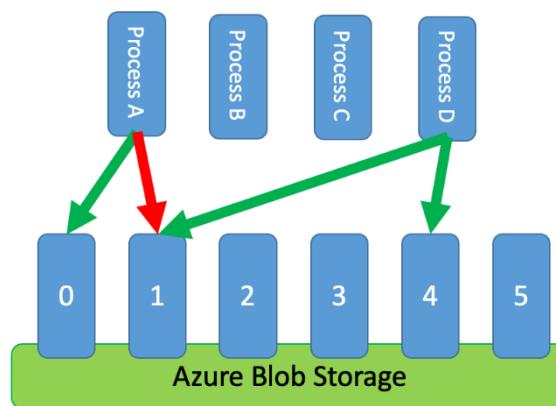


In addition, it's sometimes necessary for multiple uncoordinated processes to share a throttled service. To implement rate limiting in this scenario you can logically partition the service's capacity and then use a distributed mutual exclusion system to manage exclusive locks on those partitions. The uncoordinated processes can then compete for locks on those partitions whenever they need capacity. For each partition that a process holds a lock for, it's granted a certain amount of capacity.

For example, if the throttled system allows 500 requests per second, you might create 20 partitions worth 25 requests per second each. If a process needed to issue 100 requests, it might ask the distributed mutual exclusion system for four partitions. The system might grant two partitions for 10 seconds. The process would then rate limit to 50 requests per second, complete the task in two seconds, and then release the lock.

One way to implement this pattern would be to use Azure Storage. In this scenario, you create one 0-byte blob per logical partition in a container. Your applications can then obtain [exclusive leases](#) directly against those blobs for a short period of time (for example, 15 seconds). For every lease an application is granted, it will be able to use that partition's worth of capacity. The application then needs to track the lease time so that, when it expires, it can stop using the capacity it was granted. When implementing this pattern, you'll often want each process to attempt to lease a random partition when it needs capacity.

To further reduce latency, you might allocate a small amount of exclusive capacity for each process. A process would then only seek to obtain a lease on shared capacity if it needed to exceed its reserved capacity.



As an alternative to Azure Storage, you could also implement this kind of lease management system using technologies such as [Zookeeper](#) , [Consul](#) , [etcd](#) , [Redis/Redsync](#) , and others.

Issues and considerations

Consider the following when deciding how to implement this pattern:

- While the rate limiting pattern can reduce the number of throttling errors, your application will still need to properly handle any throttling errors that may occur.
- If your application has multiple workstreams that access the same throttled service, you'll need to integrate all of them into your rate limiting strategy. For instance, you might support bulk loading records into a database but also querying for records in that same database. You can manage capacity by ensuring all workstreams are gated through the same rate limiting mechanism. Alternatively, you might reserve separate pools of capacity for each workstream.
- The throttled service may be used in multiple applications. In some—but not all—cases it is possible to coordinate that usage (as shown above). If you start seeing a larger than expected number of throttling errors, this may be a sign of contention between applications accessing a service. If so, you may need to consider temporarily reducing the throughput imposed by your rate limiting mechanism until the usage from other applications lowers.

When to use this pattern

Use this pattern to:

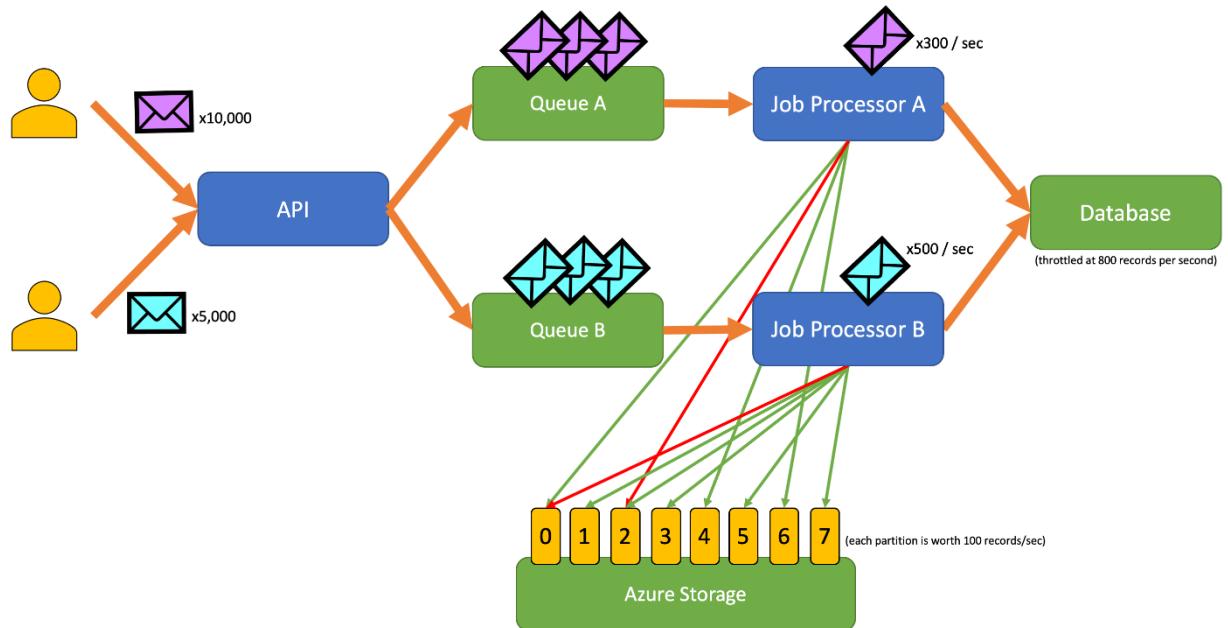
- Reduce throttling errors raised by a throttle-limited service.
- Reduce traffic compared to a naive retry on error approach.
- Reduce memory consumption by dequeuing records only when there is capacity to process them.

Example

The following example application allows users to submit records of various types to an API. There is a unique job processor for each record type that performs the following steps:

1. Validation
2. Enrichment
3. Insertion of the record into the database

All components of the application (API, job processor A, and job processor B) are separate processes that may be scaled independently. The processes do not directly communicate with one another.



This diagram incorporates the following workflow:

1. A user submits 10,000 records of type A to the API.
2. The API enqueues those 10,000 records in Queue A.
3. A user submits 5,000 records of type B to the API.
4. The API enqueues those 5,000 records in Queue B.
5. Job Processor A sees Queue A has records and tries to gain an exclusive lease on blob 2.
6. Job Processor B sees Queue B has records and tries to gain an exclusive lease on blob 2.
7. Job Processor A fails to obtain the lease.
8. Job Processor B obtains the lease on blob 2 for 15 seconds. It can now rate limit requests to the database at a rate of 100 per second.
9. Job Processor B dequeues 100 records from Queue B and writes them.
10. One second passes.
11. Job Processor A sees Queue A has more records and tries to gain an exclusive lease on blob 6.
12. Job Processor B sees Queue B has more records and tries to gain an exclusive lease on blob 3.
13. Job Processor A obtains the lease on blob 6 for 15 seconds. It can now rate limit requests to the database at a rate of 100 per second.
14. Job Processor B obtains the lease on blob 3 for 15 seconds. It can now rate limit requests to the database at a rate of 200 per second. (It also holds the lease for blob 2.)
15. Job Processor A dequeues 100 records from Queue A and writes them.
16. Job Processor B dequeues 200 records from Queue B and writes them.
17. One second passes.

18. Job Processor A sees Queue A has more records and tries to gain an exclusive lease on blob 0.
19. Job Processor B sees Queue B has more records and tries to gain an exclusive lease on blob 1.
20. Job Processor A obtains the lease on blob 0 for 15 seconds. It can now rate limit requests to the database at a rate of 200 per second. (It also holds the lease for blob 6.)
21. Job Processor B obtains the lease on blob 1 for 15 seconds. It can now rate limit requests to the database at a rate of 300 per second. (It also holds the lease for blobs 2 and 3.)
22. Job Processor A dequeues 200 records from Queue A and writes them.
23. Job Processor B dequeues 300 records from Queue B and writes them.
24. And so on...

After 15 seconds, one or both jobs still will not be completed. As the leases expire, a processor should also reduce the number of requests it dequeues and writes.



Implementations of this pattern are available in different programming languages:

- **Go** implementation is available on [GitHub ↗](#).
- **Java** implementation is available on [GitHub ↗](#).

Related resources

The following patterns and guidance might also be relevant when implementing this pattern:

- [Throttling](#). The rate limiting pattern discussed here is typically implemented in response to a service that is throttled.
- [Retry](#). When requests to throttled service result in throttling errors, it's generally appropriate to retry those after an appropriate interval.

[Queue-Based Load Leveling](#) is similar but differs from the Rate Limiting pattern in several key ways:

1. Rate limiting doesn't necessarily need to use queues to manage load, but it does need to make use of a durable messaging service. For example, a rate limiting pattern can make use of services like Apache Kafka or Azure Event Hubs.
2. The rate limiting pattern introduces the concept of a distributed mutual exclusion system on partitions, which allows you to manage capacity for multiple uncoordinated processes that communicate with the same throttled service.

3. A queue-based load leveling pattern is applicable anytime there is a performance mismatch between services or to improve resilience. This makes it a broader pattern than rate limiting, which is more specifically concerned with efficiently accessing a throttled service.

Retry pattern

Azure

Enable an application to handle transient failures when it tries to connect to a service or network resource, by transparently retrying a failed operation. This can improve the stability of the application.

Context and problem

An application that communicates with elements running in the cloud has to be sensitive to the transient faults that can occur in this environment. Faults include the momentary loss of network connectivity to components and services, the temporary unavailability of a service, or timeouts that occur when a service is busy.

These faults are typically self-correcting, and if the action that triggered a fault is repeated after a suitable delay it's likely to be successful. For example, a database service that's processing a large number of concurrent requests can implement a [throttling strategy](#) that temporarily rejects any further requests until its workload has eased. An application trying to access the database might fail to connect, but if it tries again after a delay it might succeed.

Solution

In the cloud, transient faults aren't uncommon and an application should be designed to handle them elegantly and transparently. This minimizes the effects faults can have on the business tasks the application is performing.

If an application detects a failure when it tries to send a request to a remote service, it can handle the failure using the following strategies:

- **Cancel.** If the fault indicates that the failure isn't transient or is unlikely to be successful if repeated, the application should cancel the operation and report an exception. For example, an authentication failure caused by providing invalid credentials is not likely to succeed no matter how many times it's attempted.
- **Retry.** If the specific fault reported is unusual or rare, it might have been caused by unusual circumstances such as a network packet becoming corrupted while it was being transmitted. In this case, the application could retry the failing request again

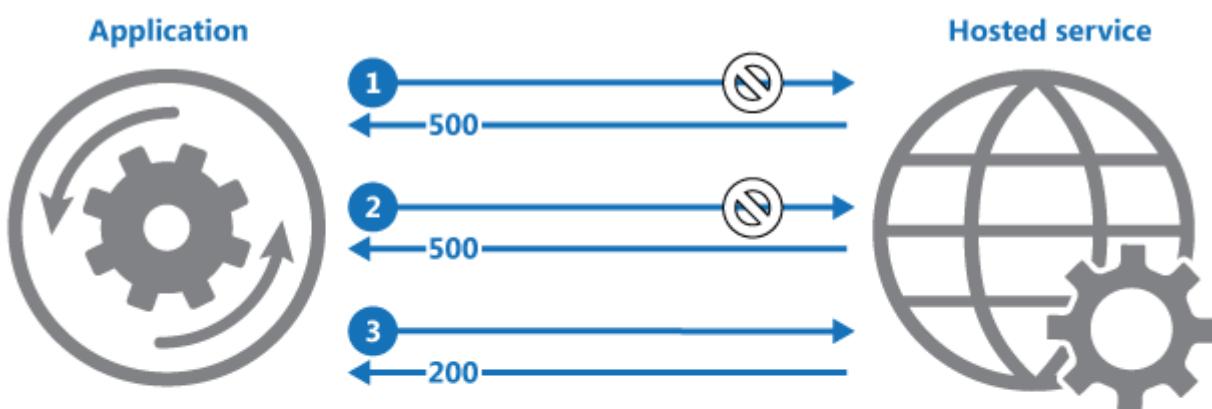
immediately because the same failure is unlikely to be repeated and the request will probably be successful.

- **Retry after delay.** If the fault is caused by one of the more commonplace connectivity or busy failures, the network or service might need a short period while the connectivity issues are corrected or the backlog of work is cleared. The application should wait for a suitable time before retrying the request.

For the more common transient failures, the period between retries should be chosen to spread requests from multiple instances of the application as evenly as possible. This reduces the chance of a busy service continuing to be overloaded. If many instances of an application are continually overwhelming a service with retry requests, it'll take the service longer to recover.

If the request still fails, the application can wait and make another attempt. If necessary, this process can be repeated with increasing delays between retry attempts, until some maximum number of requests have been attempted. The delay can be increased incrementally or exponentially, depending on the type of failure and the probability that it'll be corrected during this time.

The following diagram illustrates invoking an operation in a hosted service using this pattern. If the request is unsuccessful after a predefined number of attempts, the application should treat the fault as an exception and handle it accordingly.



- 1: Application invokes operation on hosted service. The request fails, and the service host responds with HTTP response code 500 (internal server error).
- 2: Application waits for a short interval and tries again. The request still fails with HTTP response code 500.
- 3: Application waits for a longer interval and tries again. The request succeeds with HTTP response code 200 (OK).

The application should wrap all attempts to access a remote service in code that implements a retry policy matching one of the strategies listed above. Requests sent to different services can be subject to different policies. Some vendors provide libraries

that implement retry policies, where the application can specify the maximum number of retries, the time between retry attempts, and other parameters.

An application should log the details of faults and failing operations. This information is useful to operators. That being said, in order to avoid flooding operators with alerts on operations where subsequently retried attempts were successful, it is best to log early failures as *informational entries* and only the failure of the last of the retry attempts as an actual error. Here is an [example of how this logging model would look like ↗](#).

If a service is frequently unavailable or busy, it's often because the service has exhausted its resources. You can reduce the frequency of these faults by scaling out the service. For example, if a database service is continually overloaded, it might be beneficial to partition the database and spread the load across multiple servers.

[Microsoft Entity Framework](#) provides facilities for retrying database operations. Also, most Azure services and client SDKs include a retry mechanism. For more information, see [Retry guidance for specific services](#).

Issues and considerations

You should consider the following points when deciding how to implement this pattern.

The retry policy should be tuned to match the business requirements of the application and the nature of the failure. For some noncritical operations, it's better to fail fast rather than retry several times and impact the throughput of the application. For example, in an interactive web application accessing a remote service, it's better to fail after a smaller number of retries with only a short delay between retry attempts, and display a suitable message to the user (for example, "please try again later"). For a batch application, it might be more appropriate to increase the number of retry attempts with an exponentially increasing delay between attempts.

An aggressive retry policy with minimal delay between attempts, and a large number of retries, could further degrade a busy service that's running close to or at capacity. This retry policy could also affect the responsiveness of the application if it's continually trying to perform a failing operation.

If a request still fails after a significant number of retries, it's better for the application to prevent further requests going to the same resource and simply report a failure immediately. When the period expires, the application can tentatively allow one or more requests through to see whether they're successful. For more details of this strategy, see the [Circuit Breaker pattern](#).

Consider whether the operation is idempotent. If so, it's inherently safe to retry. Otherwise, retries could cause the operation to be executed more than once, with unintended side effects. For example, a service might receive the request, process the request successfully, but fail to send a response. At that point, the retry logic might resend the request, assuming that the first request wasn't received.

A request to a service can fail for a variety of reasons raising different exceptions depending on the nature of the failure. Some exceptions indicate a failure that can be resolved quickly, while others indicate that the failure is longer lasting. It's useful for the retry policy to adjust the time between retry attempts based on the type of the exception.

Consider how retrying an operation that's part of a transaction will affect the overall transaction consistency. Fine tune the retry policy for transactional operations to maximize the chance of success and reduce the need to undo all the transaction steps.

Ensure that all retry code is fully tested against a variety of failure conditions. Check that it doesn't severely impact the performance or reliability of the application, cause excessive load on services and resources, or generate race conditions or bottlenecks.

Implement retry logic only where the full context of a failing operation is understood. For example, if a task that contains a retry policy invokes another task that also contains a retry policy, this extra layer of retries can add long delays to the processing. It might be better to configure the lower-level task to fail fast and report the reason for the failure back to the task that invoked it. This higher-level task can then handle the failure based on its own policy.

It's important to log all connectivity failures that cause a retry so that underlying problems with the application, services, or resources can be identified.

Investigate the faults that are most likely to occur for a service or a resource to discover if they're likely to be long lasting or terminal. If they are, it's better to handle the fault as an exception. The application can report or log the exception, and then try to continue either by invoking an alternative service (if one is available), or by offering degraded functionality. For more information on how to detect and handle long-lasting faults, see the [Circuit Breaker pattern](#).

When to use this pattern

Use this pattern when an application could experience transient faults as it interacts with a remote service or accesses a remote resource. These faults are expected to be short

lived, and repeating a request that has previously failed could succeed on a subsequent attempt.

This pattern might not be useful:

- When a fault is likely to be long lasting, because this can affect the responsiveness of an application. The application might be wasting time and resources trying to repeat a request that's likely to fail.
- For handling failures that aren't due to transient faults, such as internal exceptions caused by errors in the business logic of an application.
- As an alternative to addressing scalability issues in a system. If an application experiences frequent busy faults, it's often a sign that the service or resource being accessed should be scaled up.

Example

This example in C# illustrates an implementation of the Retry pattern. The `OperationWithBasicRetryAsync` method, shown below, invokes an external service asynchronously through the `TransientOperationAsync` method. The details of the `TransientOperationAsync` method will be specific to the service and are omitted from the sample code.

C#

```
private int retryCount = 3;
private readonly TimeSpan delay = TimeSpan.FromSeconds(5);

public async Task OperationWithBasicRetryAsync()
{
    int currentRetry = 0;

    for (;;)
    {
        try
        {
            // Call external service.
            await TransientOperationAsync();

            // Return or break.
            break;
        }
        catch (Exception ex)
        {
            Trace.TraceError("Operation Exception");

            currentRetry++;
        }
    }
}
```

```

    // Check if the exception thrown was a transient exception
    // based on the logic in the error detection strategy.
    // Determine whether to retry the operation, as well as how
    // long to wait, based on the retry strategy.
    if (currentRetry > this.retryCount || !IsTransient(ex))
    {
        // If this isn't a transient error or we shouldn't retry,
        // rethrow the exception.
        throw;
    }
}

// Wait to retry the operation.
// Consider calculating an exponential delay here and
// using a strategy best suited for the operation and fault.
await Task.Delay(delay);
}
}

// Async method that wraps a call to a remote service (details not shown).
private async Task TransientOperationAsync()
{
    ...
}

```

The statement that invokes this method is contained in a try/catch block wrapped in a for loop. The for loop exits if the call to the `TransientOperationAsync` method succeeds without throwing an exception. If the `TransientOperationAsync` method fails, the catch block examines the reason for the failure. If it's believed to be a transient error the code waits for a short delay before retrying the operation.

The for loop also tracks the number of times that the operation has been attempted, and if the code fails three times the exception is assumed to be more long lasting. If the exception isn't transient or it's long lasting, the catch handler throws an exception. This exception exits the for loop and should be caught by the code that invokes the `OperationWithBasicRetryAsync` method.

The `IsTransient` method, shown below, checks for a specific set of exceptions that are relevant to the environment the code is run in. The definition of a transient exception will vary according to the resources being accessed and the environment the operation is being performed in.

C#

```

private bool IsTransient(Exception ex)
{
    // Determine if the exception is transient.
    // In some cases this is as simple as checking the exception type, in
    // other

```

```

// cases it might be necessary to inspect other properties of the
exception.

if (ex is OperationTransientException)
    return true;

var webException = ex as WebException;
if (webException != null)
{
    // If the web exception contains one of the following status values
    // it might be transient.
    return new[] {WebExceptionStatus.ConnectionClosed,
                  WebExceptionStatus.Timeout,
                  WebExceptionStatus.RequestCanceled }.
                  Contains(webException.Status);
}

// Additional exception checking logic goes here.
return false;
}

```

Next steps

- Before writing custom retry logic, consider using a general framework such as [Polly](#) for .NET or [Resilience4j](#) for Java.
- When processing commands that change business data, be aware that retries can result in the action being performed twice, which could be problematic if that action is something like charging a customer's credit card. Using the Idempotence pattern described in [this blog post](#) can help deal with these situations.

Related resources

- [Reliable web app pattern](#) shows you how to apply the retry pattern to web applications converging on the cloud.
- For most Azure services, the client SDKs include built-in retry logic. For more information, see [Retry guidance for Azure services](#).
- [Circuit Breaker pattern](#). If a failure is expected to be more long lasting, it might be more appropriate to implement the Circuit Breaker pattern. Combining the Retry and Circuit Breaker patterns provides a comprehensive approach to handling faults.

Saga distributed transactions pattern

Azure

The Saga design pattern is a way to manage data consistency across microservices in distributed transaction scenarios. A saga is a sequence of transactions that updates each service and publishes a message or event to trigger the next transaction step. If a step fails, the saga executes compensating transactions that counteract the preceding transactions.

Context and problem

A *transaction* is a single unit of logic or work, sometimes made up of multiple operations. Within a transaction, an *event* is a state change that occurs to an entity, and a *command* encapsulates all information needed to perform an action or trigger a later event.

Transactions must be *atomic, consistent, isolated, and durable (ACID)*. Transactions within a single service are ACID, but cross-service data consistency requires a cross-service transaction management strategy.

In multiservices architectures:

- *Atomicity* is an indivisible and irreducible set of operations that must all occur or none occur.
- *Consistency* means the transaction brings the data only from one valid state to another valid state.
- *Isolation* guarantees that concurrent transactions produce the same data state that sequentially executed transactions would have produced.
- *Durability* ensures that committed transactions remain committed even in case of system failure or power outage.

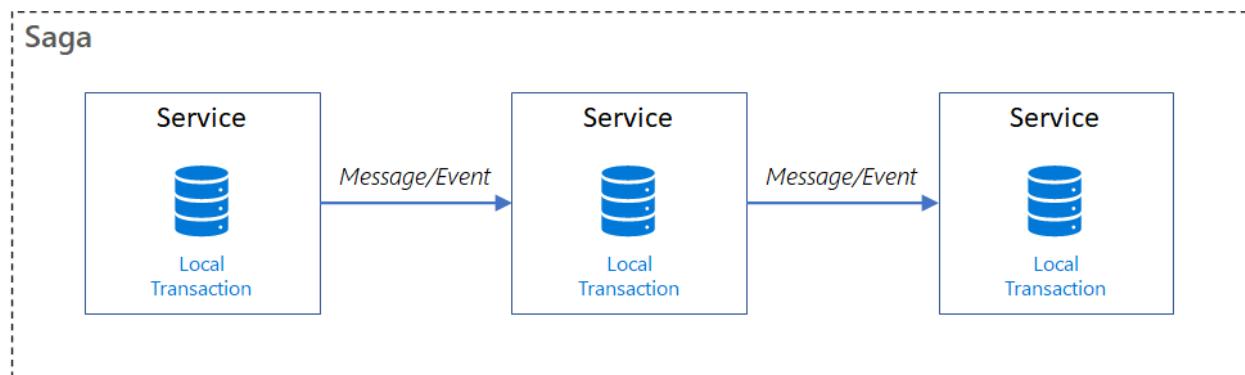
A [database-per-microservice](#) model provides many benefits for microservices architectures. Encapsulating domain data lets each service use its best data store type and schema, scale its own data store as necessary, and be insulated from other services' failures. However, ensuring data consistency across service-specific databases poses challenges.

Distributed transactions like the [two-phase commit \(2PC\)](#) protocol require all participants in a transaction to commit or roll back before the transaction can proceed. However some participant implementations, such as NoSQL databases and message brokering, don't support this model.

Another distributed transaction limitation is [interprocess communication \(IPC\)](#) synchronicity and availability. Operating system-provided IPC allows separate processes to share data. For distributed transactions to commit, all participating services must be available, potentially reducing overall system availability. Architectural implementations with IPC or transaction limitations are candidates for the Saga pattern.

Solution

The Saga pattern provides transaction management using a sequence of *local transactions*. A local transaction is the atomic work effort performed by a saga participant. Each local transaction updates the database and publishes a message or event to trigger the next local transaction in the saga. If a local transaction fails, the saga executes a series of *compensating transactions* that undo the changes that were made by the preceding local transactions.



In Saga patterns:

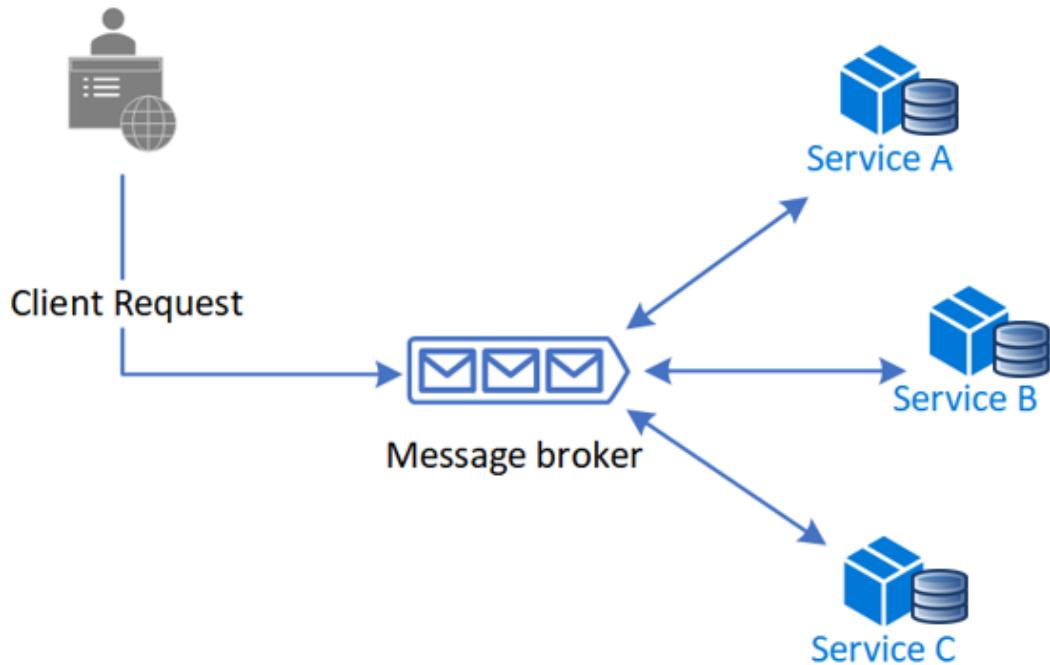
- *Compensable transactions* are transactions that can potentially be reversed by processing another transaction with the opposite effect.
- A *pivot transaction* is the go/no-go point in a saga. If the pivot transaction commits, the saga runs until completion. A pivot transaction can be a transaction that is neither compensable nor retryable, or it can be the last compensable transaction or the first retryable transaction in the saga.
- *Retryable transactions* are transactions that follow the pivot transaction and are guaranteed to succeed.

There are two common saga implementation approaches, *choreography* and *orchestration*. Each approach has its own set of challenges and technologies to

coordinate the workflow.

Choreography

Choreography is a way to coordinate sagas where participants exchange events without a centralized point of control. With choreography, each local transaction publishes domain events that trigger local transactions in other services.



Benefits

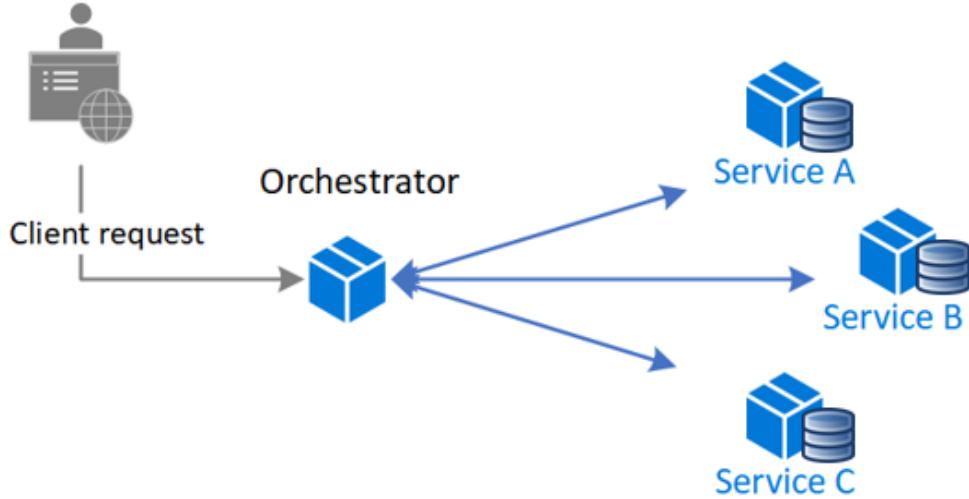
- Good for simple workflows that require few participants and don't need a coordination logic.
- Doesn't require additional service implementation and maintenance.
- Doesn't introduce a single point of failure, since the responsibilities are distributed across the saga participants.

Drawbacks

- Workflow can become confusing when adding new steps, as it's difficult to track which saga participants listen to which commands.
- There's a risk of cyclic dependency between saga participants because they have to consume each other's commands.
- Integration testing is difficult because all services must be running to simulate a transaction.

Orchestration

Orchestration is a way to coordinate sagas where a centralized controller tells the saga participants what local transactions to execute. The saga orchestrator handles all the transactions and tells the participants which operation to perform based on events. The orchestrator executes saga requests, stores and interprets the states of each task, and handles failure recovery with compensating transactions.



Benefits

- Good for complex workflows involving many participants or new participants added over time.
- Suitable when there is control over every participant in the process, and control over the flow of activities.
- Doesn't introduce cyclical dependencies, because the orchestrator unilaterally depends on the saga participants.
- Saga participants don't need to know about commands for other participants. Clear separation of concerns simplifies business logic.

Drawbacks

- Additional design complexity requires an implementation of a coordination logic.
- There's an additional point of failure, because the orchestrator manages the complete workflow.

Issues and considerations

Consider the following points when implementing the Saga pattern:

- The Saga pattern may initially be challenging, as it requires a new way of thinking on how to coordinate a transaction and maintain data consistency for a business

process spanning multiple microservices.

- The Saga pattern is particularly hard to debug, and the complexity grows as participants increase.
- Data can't be rolled back, because saga participants commit changes to their local databases.
- The implementation must be capable of handling a set of potential transient failures, and provide *idempotence* for reducing side-effects and ensuring data consistency. Idempotence means that the same operation can be repeated multiple times without changing the initial result. For more information, see the [guidance on ensuring idempotence when processing messages and updating state together](#).
- It's best to implement observability to monitor and track the saga workflow.
- The lack of participant data isolation imposes durability challenges. The saga implementation must include countermeasures to reduce anomalies.

The following anomalies can happen without proper measures:

- *Lost updates*, when one saga writes without reading changes made by another saga.
- *Dirty reads*, when a transaction or a saga reads updates made by a saga that has not yet completed those updates.
- *Fuzzy/nonrepeatable reads*, when different saga steps read different data because a data update occurs between the reads.

Suggested countermeasures to reduce or prevent anomalies include:

- *Semantic lock*, an application-level lock where a saga's compensable transaction uses a semaphore to indicate an update is in progress.
- *Commutative updates* that can be executed in any order and produce the same result.
- *Pessimistic view*: It's possible for one saga to read dirty data, while another saga is running a compensable transaction to roll back the operation. Pessimistic view reorders the saga so the underlying data updates in a retryable transaction, which eliminates the possibility of a dirty read.
- *Reread value* verifies that data is unchanged, and then updates the record. If the record has changed, the steps abort and the saga may restart.
- A *version file* records the operations on a record as they arrive, and then executes them in the correct order.
- *By value* uses each request's business risk to dynamically select the concurrency mechanism. Low-risk requests favor sagas, while high-risk requests favor distributed transactions.

When to use this pattern

Use the Saga pattern when you need to:

- Ensure data consistency in a distributed system without tight coupling.
- Roll back or compensate if one of the operations in the sequence fails.

The Saga pattern is less suitable for:

- Tightly coupled transactions.
- Compensating transactions that occur in earlier participants.
- Cyclic dependencies.

Example

[Orchestration-based Saga on Serverless](#) is a saga implementation reference using the orchestration approach that simulates a money transfer scenario with successful and failed workflows.

Next steps

- [Distributed data](#)
- Richardson, Chris. 2018: *Microservices Patterns*. Manning Publications.

Related resources

The following patterns might also be useful when implementing this pattern:

- [Choreography](#) has each component of the system participate in the decision-making process about the workflow of a business transaction, instead of relying on a central point of control.
- [Compensating transactions](#) undo work performed by a series of steps, and eventually define a consistent operation if one or more steps fail. Cloud-hosted applications that implement complex business processes and workflows often follow this *eventual consistency model*.
- [Retry](#) lets an application handle transient failures when it tries to connect to a service or network resource, by transparently retrying the failed operation. Retry can improve the stability of the application.
- [Circuit breaker](#) handles faults that take a variable amount of time to recover from, when connecting to a remote service or resource. Circuit breaker can improve the stability and resiliency of an application.

- [Health endpoint monitoring](#) implements functional checks in an application that external tools can access through exposed endpoints at regular intervals. Health endpoint monitoring can help verify that applications and services are performing correctly.

Scheduler Agent Supervisor pattern

Azure

Coordinate a set of distributed actions as a single operation. If any of the actions fail, try to handle the failures transparently, or else undo the work that was performed, so the entire operation succeeds or fails as a whole. This can add resiliency to a distributed system, by enabling it to recover and retry actions that fail due to transient exceptions, long-lasting faults, and process failures.

Context and problem

An application performs tasks that include a number of steps, some of which might invoke remote services or access remote resources. The individual steps might be independent of each other, but they are orchestrated by the application logic that implements the task.

Whenever possible, the application should ensure that the task runs to completion and resolve any failures that might occur when accessing remote services or resources. Failures can occur for many reasons. For example, the network might be down, communications could be interrupted, a remote service might be unresponsive or in an unstable state, or a remote resource might be temporarily inaccessible, perhaps due to resource constraints. In many cases the failures will be transient and can be handled by using the [Retry pattern](#).

If the application detects a more permanent fault it can't easily recover from, it must be able to restore the system to a consistent state and ensure integrity of the entire operation.

Solution

The Scheduler Agent Supervisor pattern defines the following actors. These actors orchestrate the steps to be performed as part of the overall task.

- The **Scheduler** arranges for the steps that make up the task to be executed and orchestrates their operation. These steps can be combined into a pipeline or workflow. The Scheduler is responsible for ensuring that the steps in this workflow are performed in the right order. As each step is performed, the Scheduler records

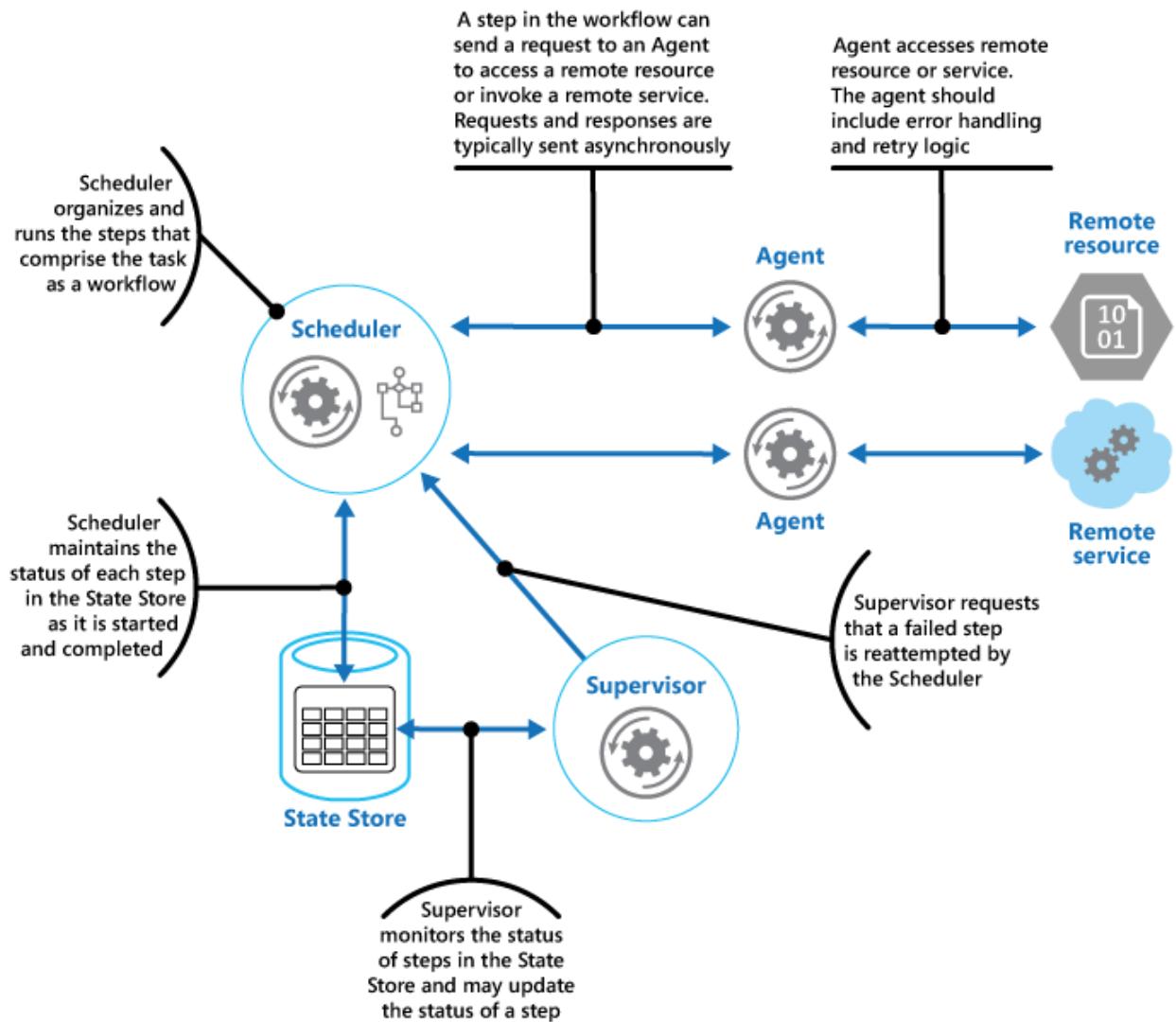
the state of the workflow, such as "step not yet started," "step running," or "step completed." The state information should also include an upper limit of the time allowed for the step to finish, called the complete-by time. If a step requires access to a remote service or resource, the Scheduler invokes the appropriate Agent, passing it the details of the work to be performed. The Scheduler typically communicates with an Agent using [asynchronous request/response messaging](#). This can be implemented using queues, although other distributed messaging technologies could be used instead.

The Scheduler performs a similar function to the Process Manager in the [Process Manager pattern](#). The actual workflow is typically defined and implemented by a workflow engine that's controlled by the Scheduler. This approach decouples the business logic in the workflow from the Scheduler.

- The **Agent** contains logic that encapsulates a call to a remote service, or access to a remote resource referenced by a step in a task. Each Agent typically wraps calls to a single service or resource, implementing the appropriate error handling and retry logic (subject to a timeout constraint, described later). When implementing retry logic, pass a stable identifier across all retry attempts so that the remote service can use it for any deduplication logic it may have. If the steps in the workflow being run by the Scheduler use several services and resources across different steps, each step might reference a different Agent (this is an implementation detail of the pattern).
- The **Supervisor** monitors the status of the steps in the task being performed by the Scheduler. It runs periodically (the frequency will be system-specific), and examines the status of steps maintained by the Scheduler. If it detects any that have timed out or failed, it arranges for the appropriate Agent to recover the step or execute the appropriate remedial action (this might involve modifying the status of a step). Note that the recovery or remedial actions are implemented by the Scheduler and Agents. The Supervisor should simply request that these actions be performed.

The Scheduler, Agent, and Supervisor are logical components and their physical implementation depends on the technology being used. For example, several logical agents might be implemented as part of a single web service.

The Scheduler maintains information about the progress of the task and the state of each step in a durable data store, called the state store. The Supervisor can use this information to help determine whether a step has failed. The figure illustrates the relationship between the Scheduler, the Agents, the Supervisor, and the state store.



ⓘ Note

This diagram shows a simplified version of the pattern. In a real implementation, there might be many instances of the Scheduler running concurrently, each a subset of tasks. Similarly, the system could run multiple instances of each Agent, or even multiple Supervisors. In this case, Supervisors must coordinate their work with each other carefully to ensure that they don't compete to recover the same failed steps and tasks. The [Leader Election pattern](#) provides one possible solution to this problem.

When the application is ready to run a task, it submits a request to the Scheduler. The Scheduler records initial state information about the task and its steps (for example, step not yet started) in the state store and then starts performing the operations defined by the workflow. As the Scheduler starts each step, it updates the information about the state of that step in the state store (for example, step running).

If a step references a remote service or resource, the Scheduler sends a message to the appropriate Agent. The message contains the information that the Agent needs to pass

to the service or access the resource, in addition to the complete-by time for the operation. If the Agent completes its operation successfully, it returns a response to the Scheduler. The Scheduler can then update the state information in the state store (for example, step completed) and perform the next step. This process continues until the entire task is complete.

An Agent can implement any retry logic that's necessary to perform its work. However, if the Agent doesn't complete its work before the complete-by period expires, the Scheduler will assume that the operation has failed. In this case, the Agent should stop its work and not try to return anything to the Scheduler (not even an error message), or try any form of recovery. The reason for this restriction is that, after a step has timed out or failed, another instance of the Agent might be scheduled to run the failing step (this process is described later).

If the Agent fails, the Scheduler won't receive a response. The pattern doesn't make a distinction between a step that has timed out and one that has genuinely failed.

If a step times out or fails, the state store will contain a record that indicates that the step is running, but the complete-by time will have passed. The Supervisor looks for steps like this and tries to recover them. One possible strategy is for the Supervisor to update the complete-by value to extend the time available to complete the step, and then send a message to the Scheduler identifying the step that has timed out. The Scheduler can then try to repeat this step. However, this design requires the tasks to be [idempotent](#)². The system should contain infrastructure to maintain consistency. For more information, see [Repeatable Infrastructure, Architect Azure applications for resiliency and availability](#), and [Resource consistency decision guide](#).

The Supervisor might need to prevent the same step from being retried if it continually fails or times out. To do this, the Supervisor could maintain a retry count for each step, along with the state information, in the state store. If this count exceeds a predefined threshold the Supervisor can adopt a strategy of waiting for an extended period before notifying the Scheduler that it should retry the step, in the expectation that the fault will be resolved during this period. Alternatively, the Supervisor can send a message to the Scheduler to request the entire task be undone by implementing a [Compensating Transaction pattern](#). This approach will depend on the Scheduler and Agents providing the information necessary to implement the compensating operations for each step that completed successfully.

It isn't the purpose of the Supervisor to monitor the Scheduler and Agents, and restart them if they fail. This aspect of the system should be handled by the infrastructure these components are running in. Similarly, the Supervisor shouldn't have knowledge of the actual business operations that the tasks being performed

by the Scheduler are running (including how to compensate should these tasks fail). This is the purpose of the workflow logic implemented by the Scheduler. The sole responsibility of the Supervisor is to determine whether a step has failed and arrange either for it to be repeated or for the entire task containing the failed step to be undone.

If the Scheduler is restarted after a failure, or the workflow being performed by the Scheduler terminates unexpectedly, the Scheduler should be able to determine the status of any inflight task that it was handling when it failed, and be prepared to resume this task from that point. The implementation details of this process are likely to be system-specific. If the task can't be recovered, it might be necessary to undo the work already performed by the task. This might also require implementing a [compensating transaction](#).

The key advantage of this pattern is that the system is resilient in the event of unexpected temporary or unrecoverable failures. The system can be constructed to be self-healing. For example, if an Agent or the Scheduler fails, a new one can be started and the Supervisor can arrange for a task to be resumed. If the Supervisor fails, another instance can be started and can take over from where the failure occurred. If the Supervisor is scheduled to run periodically, a new instance can be automatically started after a predefined interval. The state store can be replicated to reach an even greater degree of resiliency.

Issues and considerations

You should consider the following points when deciding how to implement this pattern:

- This pattern can be difficult to implement and requires thorough testing of each possible failure mode of the system.
- The recovery/retry logic implemented by the Scheduler is complex and dependent on state information held in the state store. It might also be necessary to record the information required to implement a compensating transaction in a durable data store.
- How often the Supervisor runs will be important. It should run often enough to prevent any failed steps from blocking an application for an extended period, but it shouldn't run so often that it becomes an overhead.
- The steps performed by an Agent could be run more than once. The logic that implements these steps should be idempotent.

When to use this pattern

Use this pattern when a process that runs in a distributed environment, such as the cloud, must be resilient to communications failure and/or operational failure.

This pattern might not be suitable for tasks that don't invoke remote services or access remote resources.

Example

A web application that implements an ecommerce system has been deployed on Microsoft Azure. Users can run this application to browse the available products and to place orders. The user interface runs as a web role, and the order processing elements of the application are implemented as a set of worker roles. Part of the order processing logic involves accessing a remote service, and this aspect of the system could be prone to transient or more long-lasting faults. For this reason, the designers used the Scheduler Agent Supervisor pattern to implement the order processing elements of the system.

When a customer places an order, the application constructs a message that describes the order and posts this message to a queue. A separate submission process, running in a worker role, retrieves the message, inserts the order details into the orders database, and creates a record for the order process in the state store. Note that the inserts into the orders database and the state store are performed as part of the same operation. The submission process is designed to ensure that both inserts complete together.

The state information that the submission process creates for the order includes:

- **OrderID**. The ID of the order in the orders database.
- **LockedBy**. The instance ID of the worker role handling the order. There might be multiple current instances of the worker role running the Scheduler, but each order should only be handled by a single instance.
- **CompleteBy**. The time the order should be processed by.
- **ProcessState**. The current state of the task handling the order. The possible states are:
 - **Pending**. The order has been created but processing hasn't yet been started.
 - **Processing**. The order is currently being processed.
 - **Processed**. The order has been processed successfully.
 - **Error**. The order processing has failed.

- **FailureCount**. The number of times that processing has been tried for the order.

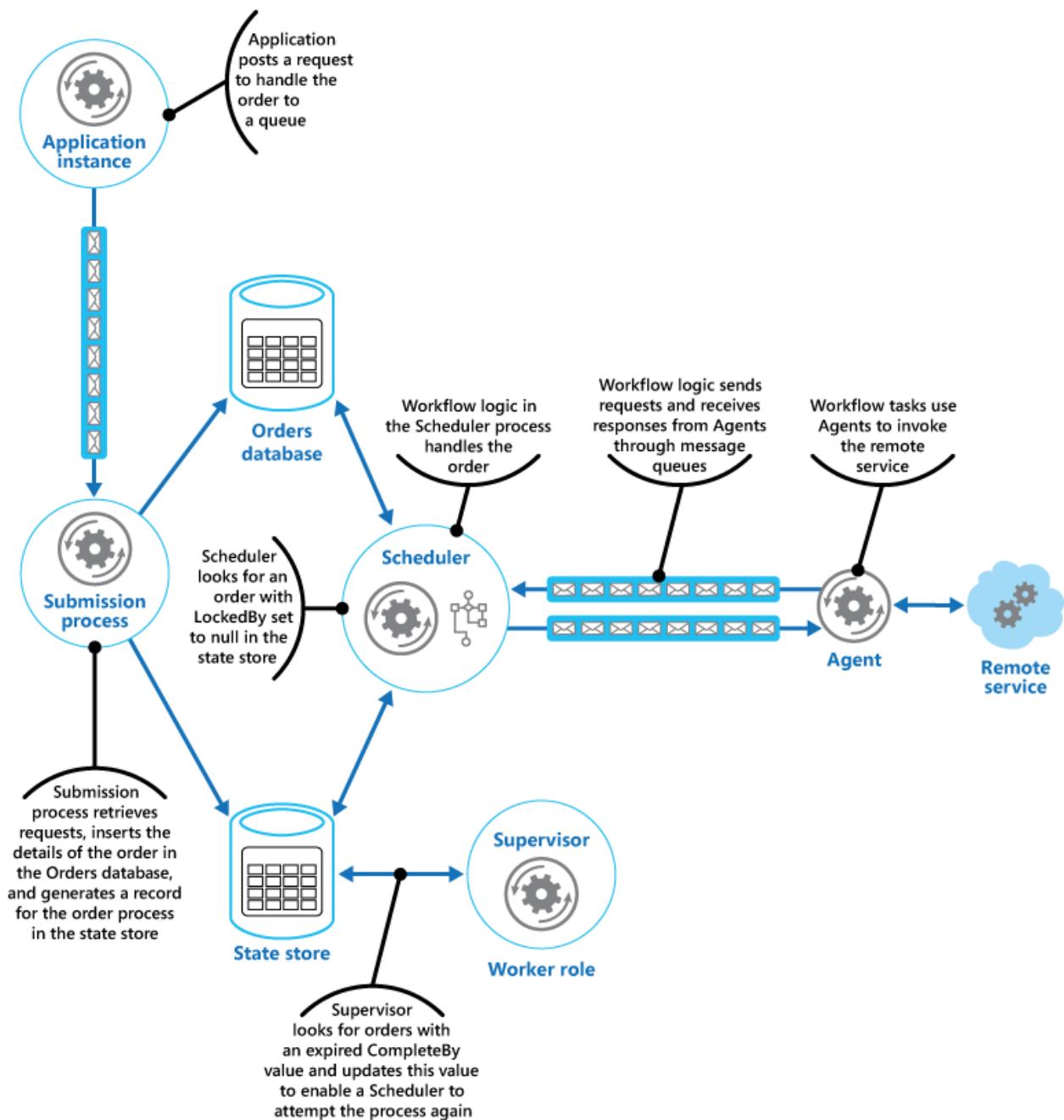
In this state information, the `OrderID` field is copied from the order ID of the new order. The `LockedBy` and `CompleteBy` fields are set to `null`, the `ProcessState` field is set to `Pending`, and the `FailureCount` field is set to 0.

Note

In this example, the order handling logic is relatively simple and only has a single step that invokes a remote service. In a more complex multistep scenario, the submission process would likely involve several steps, and so several records would be created in the state store — each one describing the state of an individual step.

The Scheduler also runs as part of a worker role and implements the business logic that handles the order. An instance of the Scheduler polling for new orders examines the state store for records where the `LockedBy` field is null and the `ProcessState` field is pending. When the Scheduler finds a new order, it immediately populates the `LockedBy` field with its own instance ID, sets the `CompleteBy` field to an appropriate time, and sets the `ProcessState` field to processing. The code is designed to be exclusive and atomic to ensure that two concurrent instances of the Scheduler can't try to handle the same order simultaneously.

The Scheduler then runs the business workflow to process the order asynchronously, passing it the value in the `OrderID` field from the state store. The workflow handling the order retrieves the details of the order from the orders database and performs its work. When a step in the order processing workflow needs to invoke the remote service, it uses an Agent. The workflow step communicates with the Agent using a pair of Azure Service Bus message queues acting as a request/response channel. The figure shows a high-level view of the solution.



The message sent to the Agent from a workflow step describes the order and includes the complete-by time. If the Agent receives a response from the remote service before the complete-by time expires, it posts a reply message on the Service Bus queue on which the workflow is listening. When the workflow step receives the valid reply message, it completes its processing and the Scheduler sets the `ProcessState` field of the order state to processed. At this point, the order processing has completed successfully.

If the complete-by time expires before the Agent receives a response from the remote service, the Agent simply halts its processing and terminates handling the order. Similarly, if the workflow handling the order exceeds the complete-by time, it also terminates. In both cases, the state of the order in the state store remains set to processing, but the complete-by time indicates that the time for processing the order has passed and the process is deemed to have failed. Note that if the Agent that's

accessing the remote service, or the workflow that's handling the order (or both) terminate unexpectedly, the information in the state store will again remain set to processing and eventually will have an expired complete-by value.

If the Agent detects an unrecoverable, nontransient fault while it's trying to contact the remote service, it can send an error response back to the workflow. The Scheduler can set the status of the order to error and raise an event that alerts an operator. The operator can then try to resolve the reason for the failure manually and resubmit the failed processing step.

The Supervisor periodically examines the state store looking for orders with an expired complete-by value. If the Supervisor finds a record, it increments the `FailureCount` field. If the failure count value is below a specified threshold value, the Supervisor resets the `LockedBy` field to null, updates the `CompleteBy` field with a new expiration time, and sets the `ProcessState` field to pending. An instance of the Scheduler can pick up this order and perform its processing as before. If the failure count value exceeds a specified threshold, the reason for the failure is assumed to be nontransient. The Supervisor sets the status of the order to error and raises an event that alerts an operator.

In this example, the Supervisor is implemented in a separate worker role. You can use a variety of strategies to arrange for the Supervisor task to be run, including using the Azure Scheduler service (not to be confused with the Scheduler component in this pattern). For more information about the Azure Scheduler service, visit the [Scheduler](#) page.

Although it isn't shown in this example, the Scheduler might need to keep the application that submitted the order informed about the progress and status of the order. The application and the Scheduler are isolated from each other to eliminate any dependencies between them. The application has no knowledge of which instance of the Scheduler is handling the order, and the Scheduler is unaware of which specific application instance posted the order.

To allow the order status to be reported, the application could use its own private response queue. The details of this response queue would be included as part of the request sent to the submission process, which would include this information in the state store. The Scheduler would then post messages to this queue indicating the status of the order (request received, order completed, order failed, and so on). It should include the order ID in these messages so they can be correlated with the original request by the application.

Next steps

The following guidance might also be relevant when implementing this pattern:

- [Asynchronous Messaging Primer](#). The components in the Scheduler Agent Supervisor pattern typically run decoupled from each other and communicate asynchronously. Describes some of the approaches that can be used to implement asynchronous communication based on message queues.
- [Reference 6: A Saga on Sagas](#). An example showing how the CQRS pattern uses a process manager (part of the CQRS Journey guidance).
- [Microsoft Azure Scheduler](#)

Related resources

The following patterns might also be relevant when implementing this pattern:

- [Retry pattern](#). An Agent can use this pattern to transparently retry an operation that accesses a remote service or resource that has previously failed. Use when the expectation is that the cause of the failure is transient and can be corrected.
- [Circuit Breaker pattern](#). An Agent can use this pattern to handle faults that take a variable amount of time to correct when connecting to a remote service or resource.
- [Compensating Transaction pattern](#). If the workflow being performed by a Scheduler can't be completed successfully, it might be necessary to undo any work it's previously performed. The Compensating Transaction pattern describes how this can be achieved for operations that follow the eventual consistency model. These types of operations are commonly implemented by a Scheduler that performs complex business processes and workflows.
- [Leader Election pattern](#). It might be necessary to coordinate the actions of multiple instances of a Supervisor to prevent them from attempting to recover the same failed process. The Leader Election pattern describes how to do this.
- [Cloud Architecture: The Scheduler-Agent-Supervisor pattern](#) on Clemens Vasters' blog
- [Process Manager pattern](#)

Sequential Convoy pattern

Azure Functions Azure Service Bus

Process a set of related messages in a defined order, without blocking processing of other groups of messages.

Context and problem

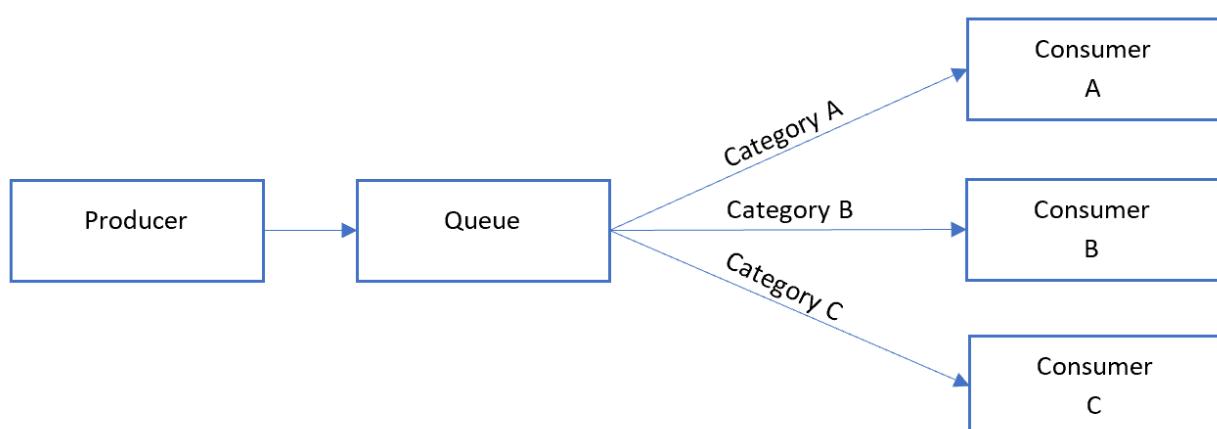
Applications often need to process a sequence of messages in the order they arrive, while still being able to scale out to handle increased load. In a distributed architecture, processing these messages in order is not straightforward, because the workers can scale independently and often pull messages independently, using a [Competing Consumers pattern](#).

For example, an order tracking system receives a ledger containing orders and the relevant operations on those orders. These operations could be to create an order, add a transaction to the order, modify a past transaction, or delete an order. In this system, operations must be performed in a first-in-first-out (FIFO) manner, but only at the order level. However, the initial queue receives a ledger containing transactions for many orders, which may be interleaved.

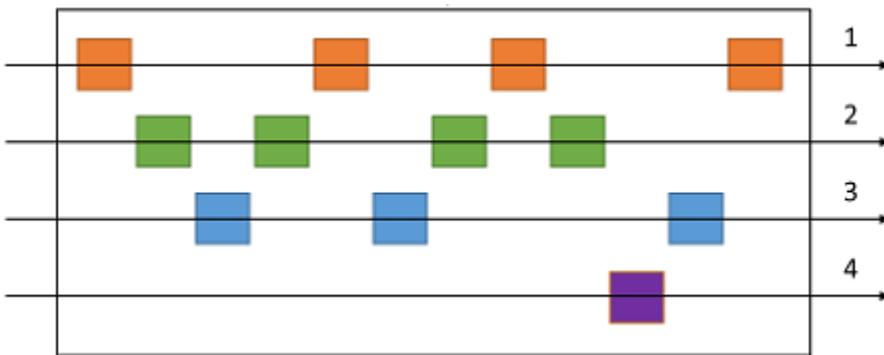
Solution

Push related messages into categories within the queuing system, and have the queue listeners lock and pull only from one category, one message at a time.

Here's what the general Sequential Convoy pattern looks like:



In the queue, messages for different categories may be interleaved, as shown in the following diagram:



Issues and considerations

Consider the following points when deciding how to implement this pattern:

- Category/scale unit. What property of your incoming messages can you scale out on? In the order tracking scenario, this property is the order ID.
- Throughput. What is your target message throughput? If it is very high, you may need to reconsider your FIFO requirements. For example, can you enforce a start/end message, sort by time, then send a batch for processing?
- Service capabilities. Does your choice of message bus allow for one-at-a-time processing of messages within a queue or category of a queue?
- Evolvability. How will you add a new category of message to the system? For example, suppose the ledger system described above is specific one customer. If you needed to onboard a new customer, could you have a set of ledger processors that distribute work per customer ID?
- It's possible that consumers might receive a message out of order, due to variable network latency when sending messages. Consider using sequence numbers to verify ordering. You might also include a special "end of sequence" flag in the last message of a transaction. Stream processing technologies such as Spark or Azure Stream Analytics can process messages in order within a time window.

When to use this pattern

Use this pattern when:

- You have messages that arrive in order and must be processed in the same order.
- Arriving messages are or can be "categorized" in such a way that the category becomes a unit of scale for the system.

This pattern might not be suitable for:

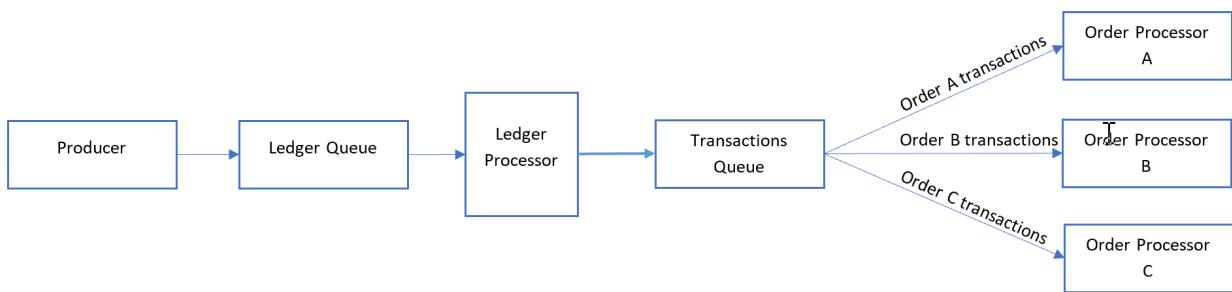
- Extremely high throughput scenarios (millions of messages/minute or second), as the FIFO requirement limits the scaling that can be done by the system.

Example

On Azure, this pattern can be implemented using Azure Service Bus [message sessions](#). For the consumers, you can use either Logic Apps with the [Service Bus peek-lock connector](#) or Azure Functions with the [Service Bus trigger](#).

For the previous order-tracking example, process each ledger message in the order it's received, and send each transaction to another queue where the category is set to the order ID. A transaction will never span multiple orders in this scenario, so consumers process each category in parallel but FIFO within the category.

The ledger processor fan outs the messages by de-batching the content of each message in the first queue:



The ledger processor takes care of:

1. Walking the ledger one transaction at a time.
 2. Setting the session ID of the message to match the order ID.
 3. Sending each ledger transaction to a secondary queue with the session ID set to the order ID.

The consumers listen to the secondary queue where they process all messages with matching order IDs *in order* from the queue. Consumers use [peek-lock](#) mode.

When considering scalability, the ledger queue is a primary bottleneck. Different transactions posted to the ledger could reference the same order ID. However, messages can fan out *after* the ledger to the number of orders in a serverless environment.

Next steps

The following information may be relevant when implementing this pattern:

- Message sessions: first in, first out (FIFO)
- Peek-Lock Message (Non-Destructive Read)
- In order delivery of correlated messages in Logic Apps by using Service Bus sessions (MSDN blog)

Sharding pattern

Azure Disk Storage

Divide a data store into a set of horizontal partitions or shards. This can improve scalability when storing and accessing large volumes of data.

Context and problem

A data store hosted by a single server might be subject to the following limitations:

- **Storage space.** A data store for a large-scale cloud application is expected to contain a huge volume of data that could increase significantly over time. A server typically provides only a finite amount of disk storage, but you can replace existing disks with larger ones, or add further disks to a machine as data volumes grow. However, the system will eventually reach a limit where it isn't possible to easily increase the storage capacity on a given server.
- **Computing resources.** A cloud application is required to support a large number of concurrent users, each of which run queries that retrieve information from the data store. A single server hosting the data store might not be able to provide the necessary computing power to support this load, resulting in extended response times for users and frequent failures as applications attempting to store and retrieve data time out. It might be possible to add memory or upgrade processors, but the system will reach a limit when it isn't possible to increase the compute resources any further.
- **Network bandwidth.** Ultimately, the performance of a data store running on a single server is governed by the rate the server can receive requests and send replies. It's possible that the volume of network traffic might exceed the capacity of the network used to connect to the server, resulting in failed requests.
- **Geography.** It might be necessary to store data generated by specific users in the same region as those users for legal, compliance, or performance reasons, or to reduce latency of data access. If the users are dispersed across different countries or regions, it might not be possible to store the entire data for the application in a single data store.

Scaling vertically by adding more disk capacity, processing power, memory, and network connections can postpone the effects of some of these limitations, but it's likely to only be a temporary solution. A commercial cloud application capable of supporting large

numbers of users and high volumes of data must be able to scale almost indefinitely, so vertical scaling isn't necessarily the best solution.

Solution

Divide the data store into horizontal partitions or shards. Each shard has the same schema, but holds its own distinct subset of the data. A shard is a data store in its own right (it can contain the data for many entities of different types), running on a server acting as a storage node.

This pattern has the following benefits:

- You can scale the system out by adding further shards running on additional storage nodes.
- A system can use off-the-shelf hardware rather than specialized and expensive computers for each storage node.
- You can reduce contention and improve performance by balancing the workload across shards.
- In the cloud, shards can be located physically close to the users that'll access the data.

When dividing a data store up into shards, decide which data should be placed in each shard. A shard typically contains items that fall within a specified range determined by one or more attributes of the data. These attributes form the shard key (sometimes referred to as the partition key). The shard key should be static. It shouldn't be based on data that might change.

Sharding physically organizes the data. When an application stores and retrieves data, the sharding logic directs the application to the appropriate shard. This sharding logic can be implemented as part of the data access code in the application, or it could be implemented by the data storage system if it transparently supports sharding.

Abstracting the physical location of the data in the sharding logic provides a high level of control over which shards contain which data. It also enables data to migrate between shards without reworking the business logic of an application if the data in the shards need to be redistributed later (for example, if the shards become unbalanced). The tradeoff is the additional data access overhead required in determining the location of each data item as it's retrieved.

To ensure optimal performance and scalability, it's important to split the data in a way that's appropriate for the types of queries that the application performs. In many cases,

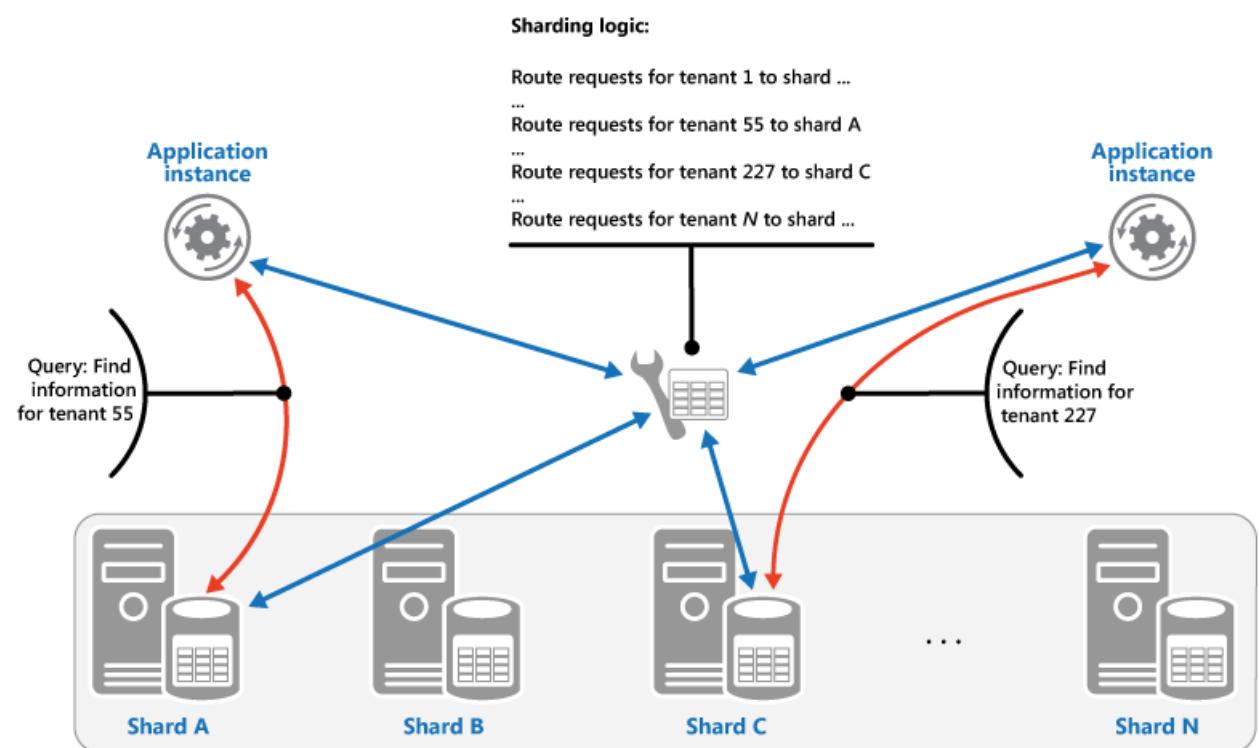
it's unlikely that the sharding scheme will exactly match the requirements of every query. For example, in a multi-tenant system an application might need to retrieve tenant data using the tenant ID, but it might also need to look up this data based on some other attribute such as the tenant's name or location. To handle these situations, implement a sharding strategy with a shard key that supports the most commonly performed queries.

If queries regularly retrieve data using a combination of attribute values, you can likely define a composite shard key by linking attributes together. Alternatively, use a pattern such as [Index Table](#) to provide fast lookup to data based on attributes that aren't covered by the shard key.

Sharding strategies

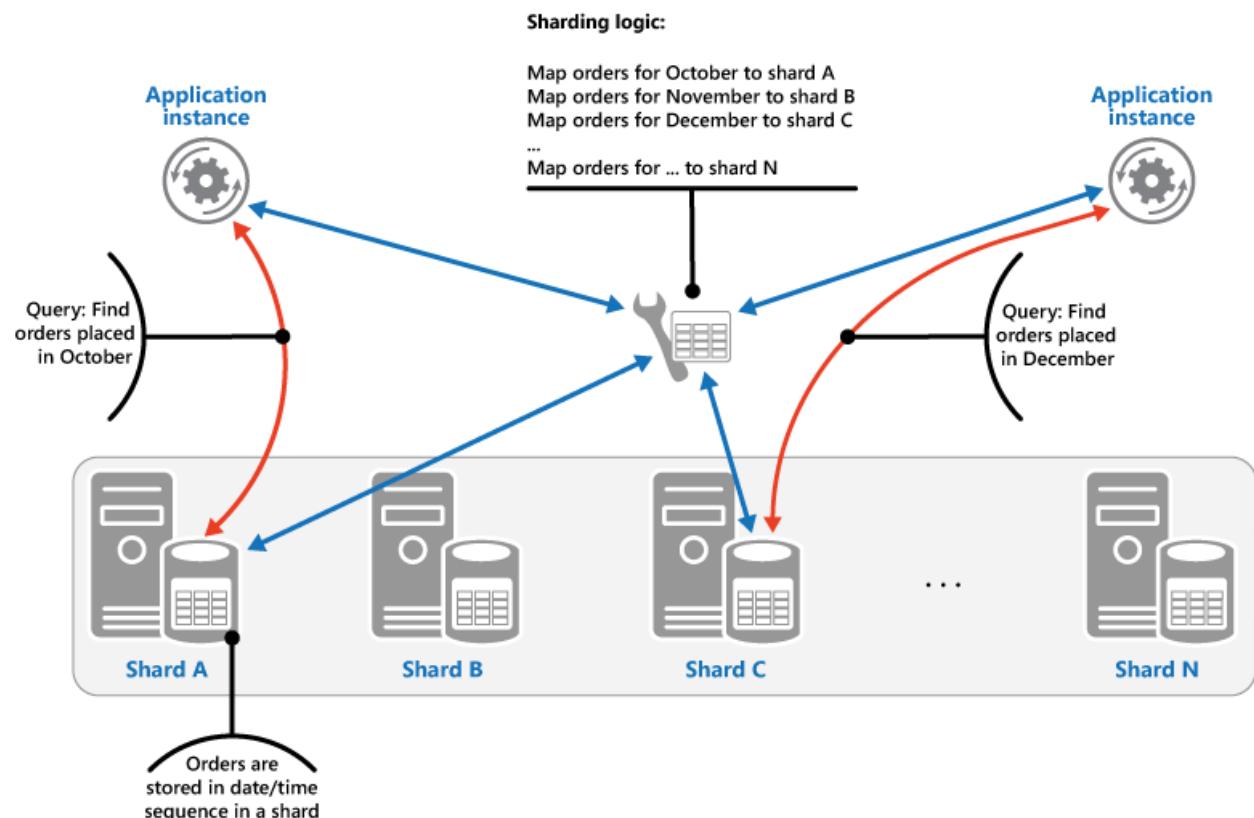
Three strategies are commonly used when selecting the shard key and deciding how to distribute data across shards. Note that there doesn't have to be a one-to-one correspondence between shards and the servers that host them—a single server can host multiple shards. The strategies are:

The Lookup strategy. In this strategy the sharding logic implements a map that routes a request for data to the shard that contains that data using the shard key. In a multi-tenant application all the data for a tenant might be stored together in a shard using the tenant ID as the shard key. Multiple tenants might share the same shard, but the data for a single tenant won't be spread across multiple shards. The figure illustrates sharding tenant data based on tenant IDs.



The mapping between the shard key and the physical storage can be based on physical shards where each shard key maps to a physical partition. Alternatively, a more flexible technique for rebalancing shards is virtual partitioning, where shard keys map to the same number of virtual shards, which in turn map to fewer physical partitions. In this approach, an application locates data using a shard key that refers to a virtual shard, and the system transparently maps virtual shards to physical partitions. The mapping between a virtual shard and a physical partition can change without requiring the application code be modified to use a different set of shard keys.

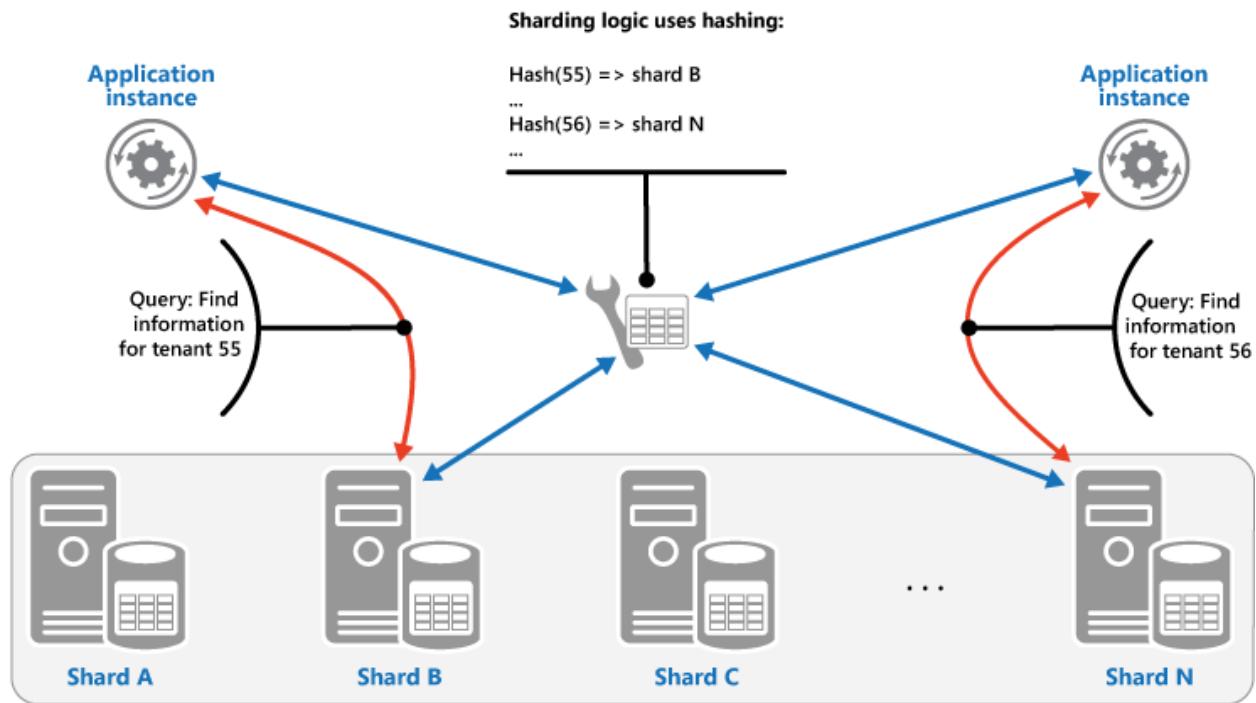
The Range strategy. This strategy groups related items together in the same shard, and orders them by shard key—the shard keys are sequential. It's useful for applications that frequently retrieve sets of items using range queries (queries that return a set of data items for a shard key that falls within a given range). For example, if an application regularly needs to find all orders placed in a given month, this data can be retrieved more quickly if all orders for a month are stored in date and time order in the same shard. If each order was stored in a different shard, they'd have to be fetched individually by performing a large number of point queries (queries that return a single data item). The next figure illustrates storing sequential sets (ranges) of data in shard.



In this example, the shard key is a composite key containing the order month as the most significant element, followed by the order day and the time. The data for orders is naturally sorted when new orders are created and added to a shard. Some data stores support two-part shard keys containing a partition key element that identifies the shard and a row key that uniquely identifies an item in the shard. Data is usually held in row

key order in the shard. Items that are subject to range queries and need to be grouped together can use a shard key that has the same value for the partition key but a unique value for the row key.

The Hash strategy. The purpose of this strategy is to reduce the chance of hotspots (shards that receive a disproportionate amount of load). It distributes the data across the shards in a way that achieves a balance between the size of each shard and the average load that each shard will encounter. The sharding logic computes the shard to store an item in based on a hash of one or more attributes of the data. The chosen hashing function should distribute data evenly across the shards, possibly by introducing some random element into the computation. The next figure illustrates sharding tenant data based on a hash of tenant IDs.



To understand the advantage of the Hash strategy over other sharding strategies, consider how a multi-tenant application that enrolls new tenants sequentially might assign the tenants to shards in the data store. When using the Range strategy, the data for tenants 1 to n will all be stored in shard A, the data for tenants n+1 to m will all be stored in shard B, and so on. If the most recently registered tenants are also the most active, most data activity will occur in a small number of shards, which could cause hotspots. In contrast, the Hash strategy allocates tenants to shards based on a hash of their tenant ID. This means that sequential tenants are most likely to be allocated to different shards, which will distribute the load across them. The previous figure shows this for tenants 55 and 56.

The three sharding strategies have the following advantages and considerations:

- **Lookup.** This offers more control over the way that shards are configured and used. Using virtual shards reduces the impact when rebalancing data because new physical partitions can be added to even out the workload. The mapping between a virtual shard and the physical partitions that implement the shard can be modified without affecting application code that uses a shard key to store and retrieve data. Looking up shard locations can impose an additional overhead.
- **Range.** This is easy to implement and works well with range queries because they can often fetch multiple data items from a single shard in a single operation. This strategy offers easier data management. For example, if users in the same region are in the same shard, updates can be scheduled in each time zone based on the local load and demand pattern. However, this strategy doesn't provide optimal balancing between shards. Rebalancing shards is difficult and might not resolve the problem of uneven load if the majority of activity is for adjacent shard keys.
- **Hash.** This strategy offers a better chance of more even data and load distribution. Request routing can be accomplished directly by using the hash function. There's no need to maintain a map. Note that computing the hash might impose an additional overhead. Also, rebalancing shards is difficult.

Most common sharding systems implement one of the approaches described above, but you should also consider the business requirements of your applications and their patterns of data usage. For example, in a multi-tenant application:

- You can shard data based on workload. You could segregate the data for highly volatile tenants in separate shards. The speed of data access for other tenants might be improved as a result.
- You can shard data based on the location of tenants. You can take the data for tenants in a specific geographic region offline for backup and maintenance during off-peak hours in that region, while the data for tenants in other regions remains online and accessible during their business hours.
- High-value tenants could be assigned their own private, high performing, lightly loaded shards, whereas lower-value tenants might be expected to share more densely-packed, busy shards.
- The data for tenants that need a high degree of data isolation and privacy can be stored on a completely separate server.

Scaling and data movement operations

Each of the sharding strategies implies different capabilities and levels of complexity for managing scale in, scale out, data movement, and maintaining state.

The Lookup strategy permits scaling and data movement operations to be carried out at the user level, either online or offline. The technique is to suspend some or all user activity (perhaps during off-peak periods), move the data to the new virtual partition or physical shard, change the mappings, invalidate or refresh any caches that hold this data, and then allow user activity to resume. Often this type of operation can be centrally managed. The Lookup strategy requires state to be highly cacheable and replica friendly.

The Range strategy imposes some limitations on scaling and data movement operations, which must typically be carried out when a part or all of the data store is offline because the data must be split and merged across the shards. Moving the data to rebalance shards might not resolve the problem of uneven load if the majority of activity is for adjacent shard keys or data identifiers that are within the same range. The Range strategy might also require some state to be maintained in order to map ranges to the physical partitions.

The Hash strategy makes scaling and data movement operations more complex because the partition keys are hashes of the shard keys or data identifiers. The new location of each shard must be determined from the hash function, or the function modified to provide the correct mappings. However, the Hash strategy doesn't require maintenance of state.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- Sharding is complementary to other forms of partitioning, such as vertical partitioning and functional partitioning. For example, a single shard can contain entities that have been partitioned vertically, and a functional partition can be implemented as multiple shards. For more information about partitioning, see the [Data Partitioning Guidance](#).
- Keep shards balanced so they all handle a similar volume of I/O. As data is inserted and deleted, it's necessary to periodically rebalance the shards to guarantee an even distribution and to reduce the chance of hotspots. Rebalancing can be an expensive operation. To reduce the necessity of rebalancing, plan for growth by ensuring that each shard contains sufficient free space to handle the expected volume of changes. You should also develop strategies and scripts you can use to quickly rebalance shards if this becomes necessary.

- Use stable data for the shard key. If the shard key changes, the corresponding data item might have to move between shards, increasing the amount of work performed by update operations. For this reason, avoid basing the shard key on potentially volatile information. Instead, look for attributes that are invariant or that naturally form a key.
- Ensure that shard keys are unique. For example, avoid using autoincrementing fields as the shard key. In some systems, autoincremented fields can't be coordinated across shards, possibly resulting in items in different shards having the same shard key.

Autoincremented values in other fields that are not shard keys can also cause problems. For example, if you use autoincremented fields to generate unique IDs, then two different items located in different shards might be assigned the same ID.

- It might not be possible to design a shard key that matches the requirements of every possible query against the data. Shard the data to support the most frequently performed queries, and if necessary create secondary index tables to support queries that retrieve data using criteria based on attributes that aren't part of the shard key. For more information, see the [Index Table pattern](#).
- Queries that access only a single shard are more efficient than those that retrieve data from multiple shards, so avoid implementing a sharding system that results in applications performing large numbers of queries that join data held in different shards. Remember that a single shard can contain the data for multiple types of entities. Consider denormalizing your data to keep related entities that are commonly queried together (such as the details of customers and the orders that they have placed) in the same shard to reduce the number of separate reads that an application performs.

If an entity in one shard references an entity stored in another shard, include the shard key for the second entity as part of the schema for the first entity. This can help to improve the performance of queries that reference related data across shards.

- If an application must perform queries that retrieve data from multiple shards, it might be possible to fetch this data by using parallel tasks. Examples include fan-out queries, where data from multiple shards is retrieved in parallel and then aggregated into a single result. However, this approach inevitably adds some complexity to the data access logic of a solution.

- For many applications, creating a larger number of small shards can be more efficient than having a small number of large shards because they can offer increased opportunities for load balancing. This can also be useful if you anticipate the need to migrate shards from one physical location to another. Moving a small shard is quicker than moving a large one.
- Make sure the resources available to each shard storage node are sufficient to handle the scalability requirements in terms of data size and throughput. For more information, see the section "Designing Partitions for Scalability" in the [Data Partitioning Guidance](#).
- Consider replicating reference data to all shards. If an operation that retrieves data from a shard also references static or slow-moving data as part of the same query, add this data to the shard. The application can then fetch all of the data for the query easily, without having to make an additional round trip to a separate data store.

If reference data held in multiple shards changes, the system must synchronize these changes across all shards. The system can experience a degree of inconsistency while this synchronization occurs. If you do this, you should design your applications to be able to handle it.

- It can be difficult to maintain referential integrity and consistency between shards, so you should minimize operations that affect data in multiple shards. If an application must modify data across shards, evaluate whether complete data consistency is actually required. Instead, a common approach in the cloud is to implement eventual consistency. The data in each partition is updated separately, and the application logic must take responsibility for ensuring that the updates all complete successfully, as well as handling the inconsistencies that can arise from querying data while an eventually consistent operation is running. For more information about implementing eventual consistency, see the [Data Consistency Primer](#).
- Configuring and managing a large number of shards can be a challenge. Tasks such as monitoring, backing up, checking for consistency, and logging or auditing must be accomplished on multiple shards and servers, possibly held in multiple locations. These tasks are likely to be implemented using scripts or other automation solutions, but that might not completely eliminate the additional administrative requirements.
- Shards can be geolocated so that the data that they contain is close to the instances of an application that use it. This approach can considerably improve

performance, but requires additional consideration for tasks that must access multiple shards in different locations.

When to use this pattern

Use this pattern when a data store is likely to need to scale beyond the resources available to a single storage node, or to improve performance by reducing contention in a data store.

ⓘ Note

The primary focus of sharding is to improve the performance and scalability of a system, but as a by-product it can also improve availability due to how the data is divided into separate partitions. A failure in one partition doesn't necessarily prevent an application from accessing data held in other partitions, and an operator can perform maintenance or recovery of one or more partitions without making the entire data for an application inaccessible. For more information, see the [Data Partitioning Guidance](#).

Example

The following example in C# uses a set of SQL Server databases acting as shards. Each database holds a subset of the data used by an application. The application retrieves data that's distributed across the shards using its own sharding logic (this is an example of a fan-out query). The details of the data that's located in each shard is returned by a method called `GetShards`. This method returns an enumerable list of `ShardInformation` objects, where the `ShardInformation` type contains an identifier for each shard and the SQL Server connection string that an application should use to connect to the shard (the connection strings aren't shown in the code example).

C#

```
private IEnumerable<ShardInformation> GetShards()
{
    // This retrieves the connection information from a shard store
    // (commonly a root database).
    return new[]
    {
        new ShardInformation
        {
            Id = 1,
            ConnectionString = ...
        }
    }
}
```

```

    },
    new ShardInformation
    {
        Id = 2,
        ConnectionString = ...
    }
);
}
}

```

The code below shows how the application uses the list of `ShardInformation` objects to perform a query that fetches data from each shard in parallel. The details of the query aren't shown, but in this example the data that's retrieved contains a string that could hold information such as the name of a customer if the shards contain the details of customers. The results are aggregated into a `ConcurrentBag` collection for processing by the application.

C#

```

// Retrieve the shards as a ShardInformation[] instance.
var shards = GetShards();

var results = new ConcurrentBag<string>();

// Execute the query against each shard in the shard list.
// This list would typically be retrieved from configuration
// or from a root/primary shard store.
Parallel.ForEach(shards, shard =>
{
    // NOTE: Transient fault handling isn't included,
    // but should be incorporated when used in a real world application.
    using (var con = new SqlConnection(shard.ConnectionString))
    {
        con.Open();
        var cmd = new SqlCommand("SELECT ... FROM ...", con);

        Trace.TraceInformation("Executing command against shard: {0}",
shard.Id);

        var reader = cmd.ExecuteReader();
        // Read the results in to a thread-safe data structure.
        while (reader.Read())
        {
            results.Add(reader.GetString(0));
        }
    }
});

Trace.TraceInformation("Fanout query complete - Record Count: {0}",
results.Count);

```

Next steps

The following guidance might also be relevant when implementing this pattern:

- [Data Consistency Primer](#). It might be necessary to maintain consistency for data distributed across different shards. Summarizes the issues surrounding maintaining consistency over distributed data, and describes the benefits and tradeoffs of different consistency models.
- [Data Partitioning Guidance](#). Sharding a data store can introduce a range of additional issues. Describes these issues in relation to partitioning data stores in the cloud to improve scalability, reduce contention, and optimize performance.

Related resources

The following patterns might also be relevant when implementing this pattern:

- [Index Table pattern](#). Sometimes it isn't possible to completely support queries just through the design of the shard key. Enables an application to quickly retrieve data from a large data store by specifying a key other than the shard key.
- [Materialized View pattern](#). To maintain the performance of some query operations, it's useful to create materialized views that aggregate and summarize data, especially if this summary data is based on information that's distributed across shards. Describes how to generate and populate these views.

Sidecar pattern

Azure

Deploy components of an application into a separate process or container to provide isolation and encapsulation. This pattern can also enable applications to be composed of heterogeneous components and technologies.

This pattern is named *Sidecar* because it resembles a sidecar attached to a motorcycle. In the pattern, the sidecar is attached to a parent application and provides supporting features for the application. The sidecar also shares the same lifecycle as the parent application, being created and retired alongside the parent. The sidecar pattern is sometimes referred to as the sidekick pattern and is a decomposition pattern.

Context and problem

Applications and services often require related functionality, such as monitoring, logging, configuration, and networking services. These peripheral tasks can be implemented as separate components or services.

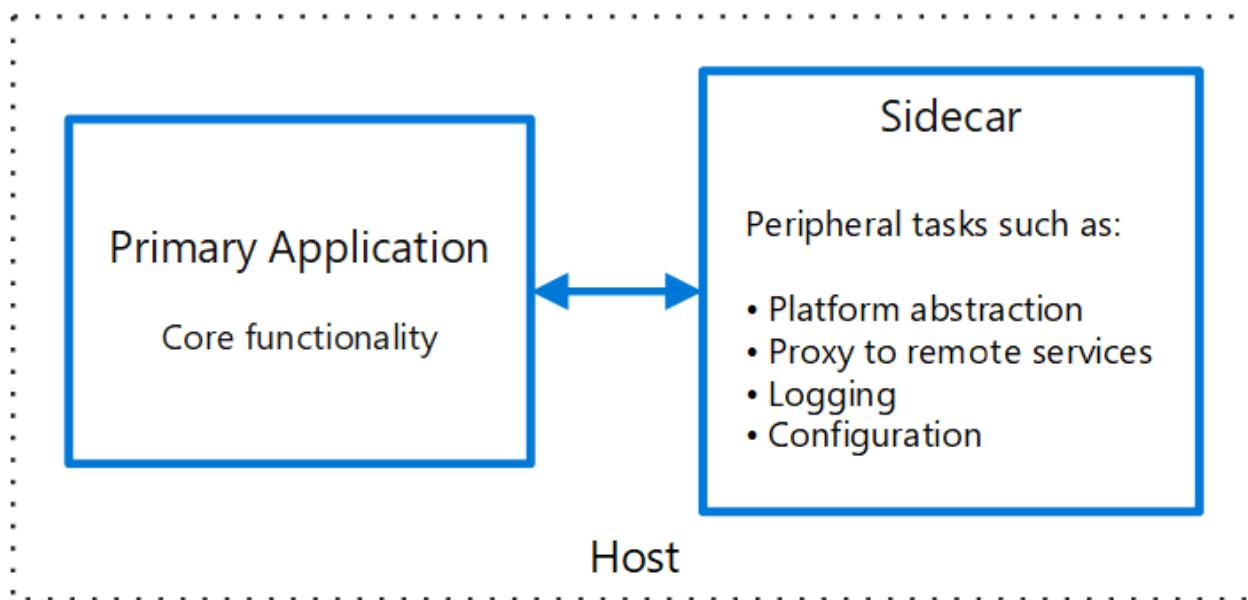
If they are tightly integrated into the application, they can run in the same process as the application, making efficient use of shared resources. However, this also means they are not well isolated, and an outage in one of these components can affect other components or the entire application. Also, they usually need to be implemented using the same language as the parent application. As a result, the component and the application have close interdependence on each other.

If the application is decomposed into services, then each service can be built using different languages and technologies. While this gives more flexibility, it means that each component has its own dependencies and requires language-specific libraries to access the underlying platform and any resources shared with the parent application. In addition, deploying these features as separate services can add latency to the application. Managing the code and dependencies for these language-specific interfaces can also add considerable complexity, especially for hosting, deployment, and management.

Solution

Co-locate a cohesive set of tasks with the primary application, but place them inside their own process or container, providing a homogeneous interface for platform services

across languages.



A sidecar service is not necessarily part of the application, but is connected to it. It goes wherever the parent application goes. Sidecars are supporting processes or services that are deployed with the primary application. On a motorcycle, the sidecar is attached to one motorcycle, and each motorcycle can have its own sidecar. In the same way, a sidecar service shares the fate of its parent application. For each instance of the application, an instance of the sidecar is deployed and hosted alongside it.

Advantages of using a sidecar pattern include:

- A sidecar is independent from its primary application in terms of runtime environment and programming language, so you don't need to develop one sidecar per language.
- The sidecar can access the same resources as the primary application. For example, a sidecar can monitor system resources used by both the sidecar and the primary application.
- Because of its proximity to the primary application, there's no significant latency when communicating between them.
- Even for applications that don't provide an extensibility mechanism, you can use a sidecar to extend functionality by attaching it as its own process in the same host or sub-container as the primary application.

The sidecar pattern is often used with containers and referred to as a sidecar container or sidekick container.

Issues and considerations

- Consider the deployment and packaging format you will use to deploy services, processes, or containers. Containers are particularly well suited to the sidecar pattern.
- When designing a sidecar service, carefully decide on the interprocess communication mechanism. Try to use language- or framework-agnostic technologies unless performance requirements make that impractical.
- Before putting functionality into a sidecar, consider whether it would work better as a separate service or a more traditional daemon.
- Also consider whether the functionality could be implemented as a library or using a traditional extension mechanism. Language-specific libraries may have a deeper level of integration and less network overhead.

When to use this pattern

Use this pattern when:

- Your primary application uses a heterogeneous set of languages and frameworks. A component located in a sidecar service can be consumed by applications written in different languages using different frameworks.
- A component is owned by a remote team or a different organization.
- A component or feature must be co-located on the same host as the application
- You need a service that shares the overall lifecycle of your main application, but can be independently updated.
- You need fine-grained control over resource limits for a particular resource or component. For example, you may want to restrict the amount of memory a specific component uses. You can deploy the component as a sidecar and manage memory usage independently of the main application.

This pattern may not be suitable:

- When interprocess communication needs to be optimized. Communication between a parent application and sidecar services includes some overhead, notably latency in the calls. This may not be an acceptable trade-off for chatty interfaces.
- For small applications where the resource cost of deploying a sidecar service for each instance is not worth the advantage of isolation.
- When the service needs to scale differently than or independently from the main applications. If so, it may be better to deploy the feature as a separate service.

Example

The sidecar pattern is applicable to many scenarios. Some common examples:

- Infrastructure API. The infrastructure development team creates a service that's deployed alongside each application, instead of a language-specific client library to access the infrastructure. The service is loaded as a sidecar and provides a common layer for infrastructure services, including logging, environment data, configuration store, discovery, health checks, and watchdog services. The sidecar also monitors the parent application's host environment and process (or container) and logs the information to a centralized service.
- Manage NGINX/HAProxy. Deploy NGINX with a sidecar service that monitors environment state, then updates the NGINX configuration file and recycles the process when a change in state is needed.
- Ambassador sidecar. Deploy an [ambassador](#) service as a sidecar. The application calls through the ambassador, which handles request logging, routing, circuit breaking, and other connectivity related features.
- Offload proxy. Place an NGINX proxy in front of a node.js service instance, to handle serving static file content for the service.

Related resources

- [Ambassador pattern](#)

Static Content Hosting pattern

Azure Storage

Deploy static content to a cloud-based storage service that can deliver them directly to the client. This can reduce the need for potentially expensive compute instances.

Context and problem

Web applications typically include some elements of static content. This static content might include HTML pages and other resources such as images and documents that are available to the client, either as part of an HTML page (such as inline images, style sheets, and client-side JavaScript files) or as separate downloads (such as PDF documents).

Although web servers are optimized for dynamic rendering and output caching, they still have to handle requests to download static content. This consumes processing cycles that could often be put to better use.

Solution

In most cloud hosting environments, you can put some of an application's resources and static pages in a storage service. The storage service can serve requests for these resources, reducing load on the compute resources that handle other web requests. The cost for cloud-hosted storage is typically much less than for compute instances.

When hosting some parts of an application in a storage service, the main considerations are related to deployment of the application and to securing resources that aren't intended to be available to anonymous users.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

- The hosted storage service must expose an HTTP endpoint that users can access to download the static resources. Some storage services also support HTTPS, so it's possible to host resources in storage services that require SSL.
- For maximum performance and availability, consider using a content delivery network (CDN) to cache the contents of the storage container in multiple

datacenters around the world. However, you'll likely have to pay for using the CDN.

- Storage accounts are often geo-replicated by default to provide resiliency against events that might affect a datacenter. This means that the IP address might change, but the URL will remain the same.
- When some content is located in a storage account and other content is in a hosted compute instance, it becomes more challenging to deploy and update the application. You might have to perform separate deployments, and version the application and content to manage it more easily—especially when the static content includes script files or UI components. However, if only static resources have to be updated, they can simply be uploaded to the storage account without needing to redeploy the application package.
- Storage services might not support the use of custom domain names. In this case it's necessary to specify the full URL of the resources in links because they'll be in a different domain from the dynamically-generated content containing the links.
- The storage containers must be configured for public read access, but it's vital to ensure that they aren't configured for public write access to prevent users being able to upload content.
- Consider using a valet key or token to control access to resources that shouldn't be available anonymously. See the [Valet Key pattern](#) for more information.

When to use this pattern

This pattern is useful for:

- Minimizing the hosting cost for websites and applications that contain some static resources.
- Minimizing the hosting cost for websites that consist of only static content and resources. Depending on the capabilities of the hosting provider's storage system, it might be possible to entirely host a fully static website in a storage account.
- Exposing static resources and content for applications running in other hosting environments or on-premises servers.
- Locating content in more than one geographical area using a content delivery network that caches the contents of the storage account in multiple datacenters around the world.

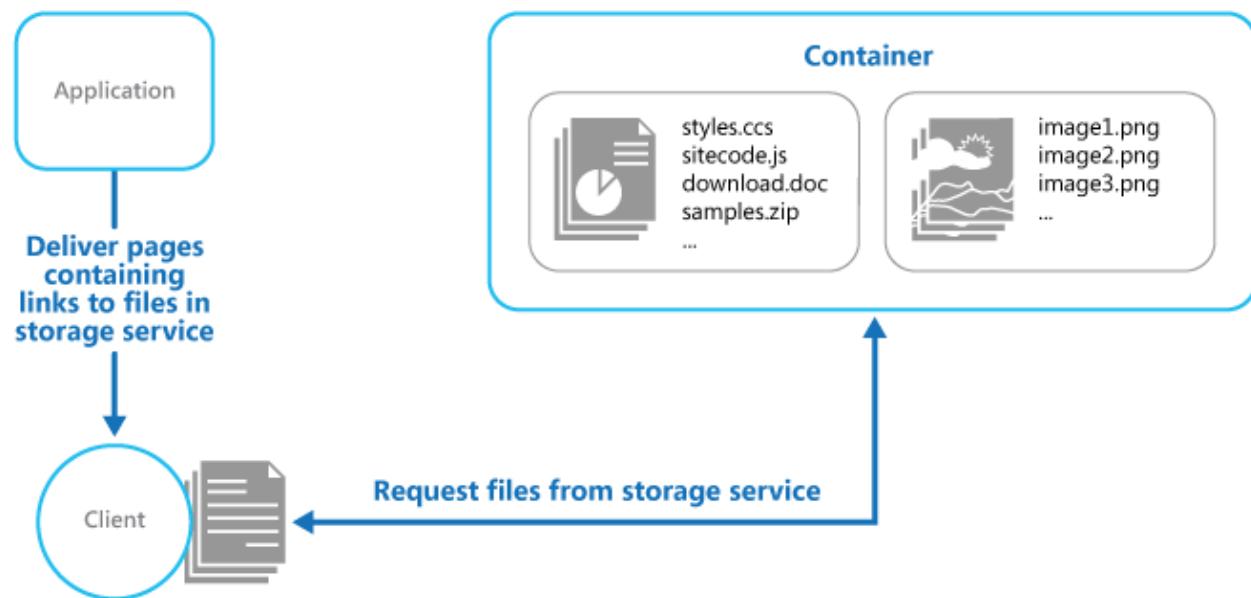
- Monitoring costs and bandwidth usage. Using a separate storage account for some or all of the static content allows the costs to be more easily separated from hosting and runtime costs.

This pattern might not be useful in the following situations:

- The application needs to perform some processing on the static content before delivering it to the client. For example, it might be necessary to add a timestamp to a document.
- The volume of static content is very small. The overhead of retrieving this content from separate storage can outweigh the cost benefit of separating it out from the compute resource.

Example

Azure Storage supports serving static content directly from a storage container. Files are served through anonymous access requests. By default, files have a URL in a subdomain of `core.windows.net`, such as `https://contoso.z4.web.core.windows.net/image.png`. You can configure a custom domain name, and use Azure CDN to access the files over HTTPS. For more information, see [Static website hosting in Azure Storage](#).



Static website hosting makes the files available for anonymous access. If you need to control who can access the files, you can store files in Azure blob storage and then generate [shared access signatures](#) to limit access.

The links in the pages delivered to the client must specify the full URL of the resource. If the resource is protected with a valet key, such as a shared access signature, this signature must be included in the URL.

A sample application that demonstrates using external storage for static resources is available on [GitHub](#). This sample uses configuration files to specify the storage account and container that holds the static content.

XML

```
<Setting name="StaticContent.StorageConnectionString"
         value="UseDevelopmentStorage=true" />
<Setting name="StaticContent.Container" value="static-content" />
```

The `Settings` class in the file `Settings.cs` of the `StaticContentHosting.Web` project contains methods to extract these values and build a string value containing the cloud storage account container URL.

C#

```
public class Settings
{
    public static string StaticContentStorageConnectionString {
        get
        {
            return RoleEnvironment.GetConfigurationSettingValue(
                "StaticContent.StorageConnectionString");
        }
    }

    public static string StaticContentContainer
    {
        get
        {
            return
RoleEnvironment.GetConfigurationSettingValue("StaticContent.Container");
        }
    }

    public static string StaticContentBaseUrl
    {
        get
        {
            var blobServiceClient = new
BlobServiceClient(StaticContentStorageConnectionString);

            return string.Format("{0}/{1}",
blobServiceClient.Uri.ToString().TrimEnd('/'),
StaticContentContainer.TrimStart('/'));
        }
    }
}
```

The `StaticContentUrlHtmlHelper` class in the file `StaticContentUrlHtmlHelper.cs` exposes a method named `StaticContentUrl` that generates a URL containing the path to the cloud storage account if the URL passed to it starts with the ASP.NET root path character (~).

C#

```
public static class StaticContentUrlHtmlHelper
{
    public static string StaticContentUrl(this HtmlHelper helper, string
contentPath)
    {
        if (contentPath.StartsWith("~/"))
        {
            contentPath = contentPath.Substring(1);
        }

        contentPath = string.Format("{0}/{1}",
Settings.StaticContentBaseUrl.TrimEnd('/'),
contentPath.TrimStart('/'));

        var url = new UrlHelper(helper.ViewContext.RequestContext);

        return url.Content(contentPath);
    }
}
```

The file `Index.cshtml` in the `Views\Home` folder contains an `img` element that uses the `StaticContentUrl` method to create the URL for its `src` attribute.

HTML

```

```

Next steps

- [Static Content Hosting sample](#). A sample application that demonstrates this pattern.

Related resources

- [Valet Key pattern](#). If the target resources aren't supposed to be available to anonymous users, use this pattern to restrict direct access.

- [Serverless web application on Azure](#). A reference architecture that uses static website hosting with Azure Functions to implement a serverless web app.

Strangler Fig pattern

Azure Migrate

Incrementally migrate a legacy system by gradually replacing specific pieces of functionality with new applications and services. As features from the legacy system are replaced, the new system eventually replaces all of the old system's features, strangling the old system and allowing you to decommission it.

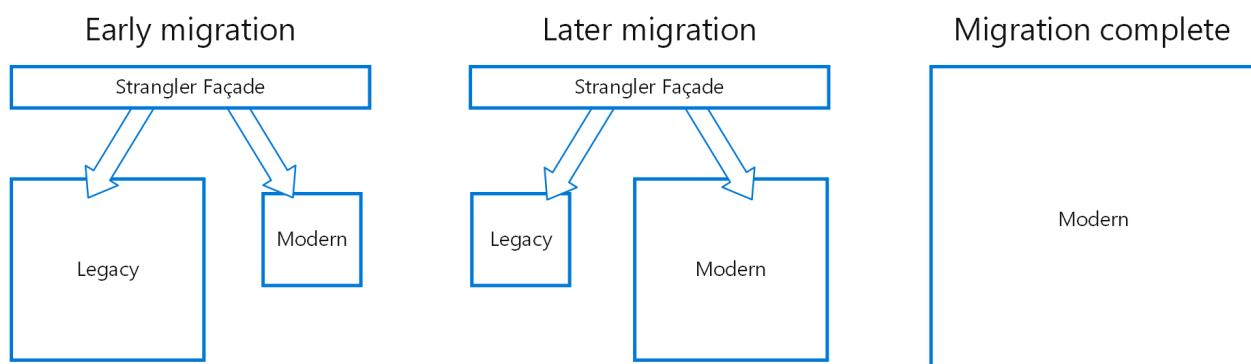
Context and problem

As systems age, the development tools, hosting technology, and even system architectures they were built on can become increasingly obsolete. As new features and functionality are added, the complexity of these applications can increase dramatically, making them harder to maintain or add new features to.

Completely replacing a complex system can be a huge undertaking. Often, you will need a gradual migration to a new system, while keeping the old system to handle features that haven't been migrated yet. However, running two separate versions of an application means that clients have to know where particular features are located. Every time a feature or service is migrated, clients need to be updated to point to the new location.

Solution

Incrementally replace specific pieces of functionality with new applications and services. Create a façade that intercepts requests going to the backend legacy system. The façade routes these requests either to the legacy application or the new services. Existing features can be migrated to the new system gradually, and consumers can continue using the same interface, unaware that any migration has taken place.



This pattern helps to minimize risk from the migration, and spread the development effort over time. With the façade safely routing users to the correct application, you can add functionality to the new system at whatever pace you like, while ensuring the legacy application continues to function. Over time, as features are migrated to the new system, the legacy system is eventually "strangled" and is no longer necessary. Once this process is complete, the legacy system can safely be retired.

Issues and considerations

- Consider how to handle services and data stores that are potentially used by both new and legacy systems. Make sure both can access these resources side-by-side.
- Structure new applications and services in a way that they can easily be intercepted and replaced in future strangler fig migrations.
- At some point, when the migration is complete, the strangler fig façade will either go away or evolve into an adaptor for legacy clients.
- Make sure the façade keeps up with the migration.
- Make sure the façade doesn't become a single point of failure or a performance bottleneck.

When to use this pattern

Use this pattern when gradually migrating a back-end application to a new architecture.

This pattern may not be suitable:

- When requests to the back-end system cannot be intercepted.
- For smaller systems where the complexity of wholesale replacement is low.

Next steps

- Martin Fowler's blog post on [StranglerFigApplication](#)

Related resources

- [Messaging Bridge pattern](#)

Throttling pattern

Azure

Control the consumption of resources used by an instance of an application, an individual tenant, or an entire service. This can allow the system to continue to function and meet service level agreements, even when an increase in demand places an extreme load on resources.

Context and problem

The load on a cloud application typically varies over time based on the number of active users or the types of activities they are performing. For example, more users are likely to be active during business hours, or the system might be required to perform computationally expensive analytics at the end of each month. There might also be sudden and unanticipated bursts in activity. If the processing requirements of the system exceed the capacity of the resources that are available, it'll suffer from poor performance and can even fail. If the system has to meet an agreed level of service, such failure could be unacceptable.

There're many strategies available for handling varying load in the cloud, depending on the business goals for the application. One strategy is to use [autoscaling](#) to match the provisioned resources to the user needs at any given time. This has the potential to consistently meet user demand, while optimizing running costs. However, while autoscaling can trigger the provisioning of additional resources, this provisioning isn't immediate. If demand grows quickly, there can be a window of time where there's a resource deficit.

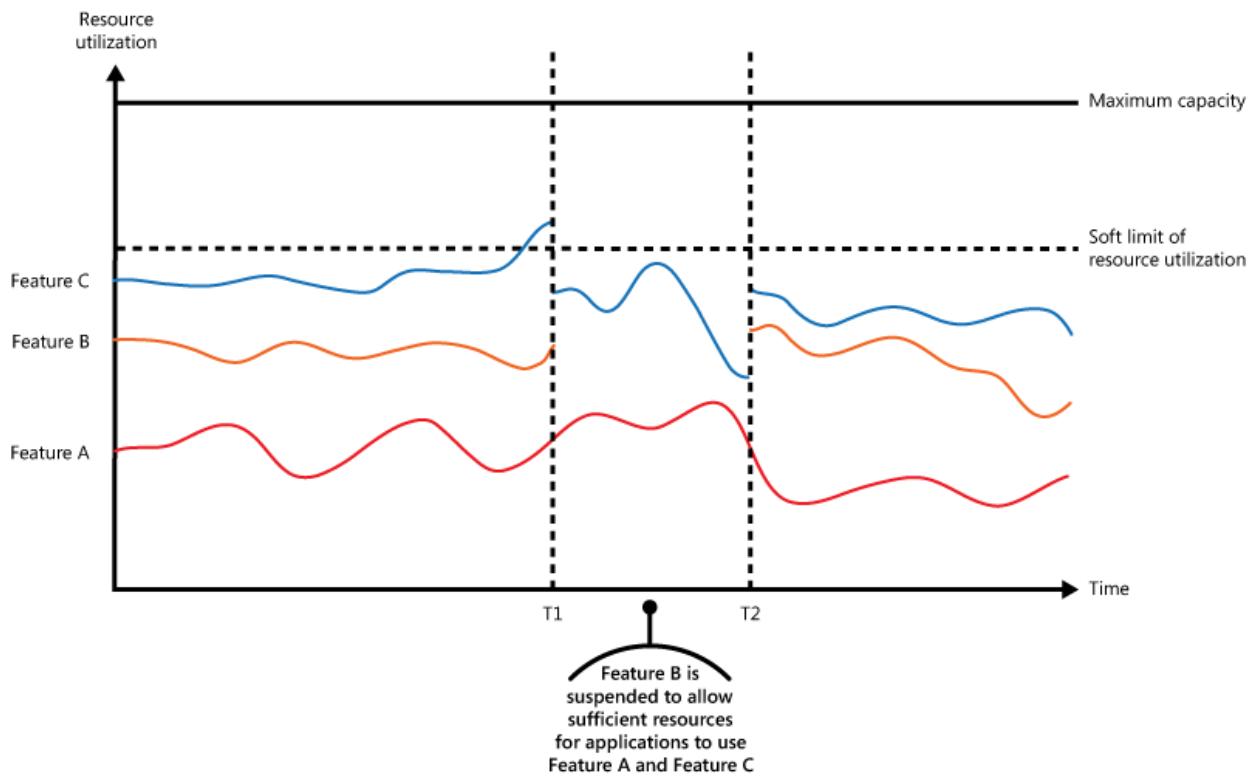
Solution

An alternative strategy to autoscaling is to allow applications to use resources only up to a limit, and then throttle them when this limit is reached. The system should monitor how it's using resources so that, when usage exceeds the threshold, it can throttle requests from one or more users. This will enable the system to continue functioning and meet any service level agreements (SLAs) that are in place. For more information on monitoring resource usage, see the [Instrumentation and Telemetry Guidance](#).

The system could implement several throttling strategies, including:

- Rejecting requests from an individual user who's already accessed system APIs more than n times per second over a given period of time. This requires the system to meter the use of resources for each tenant or user running an application. For more information, see the [Service Metering Guidance](#).
- Disabling or degrading the functionality of selected nonessential services so that essential services can run unimpeded with sufficient resources. For example, if the application is streaming video output, it could switch to a lower resolution.
- Using load leveling to smooth the volume of activity (this approach is covered in more detail by the [Queue-based Load Leveling pattern](#)). In a multi-tenant environment, this approach will reduce the performance for every tenant. If the system must support a mix of tenants with different SLAs, the work for high-value tenants might be performed immediately. Requests for other tenants can be held back, and handled when the backlog has eased. The [Priority Queue pattern](#) could be used to help implement this approach, as could exposing different endpoints for the different service levels/priorities.
- Deferring operations being performed on behalf of lower priority applications or tenants. These operations can be suspended or limited, with an exception generated to inform the tenant that the system is busy and that the operation should be retried later.
- You should be careful when integrating with some 3rd-party services that might become unavailable or return errors. Reduce the number of concurrent requests being processed so that the logs do not unnecessarily fill up with errors. You also avoid the costs that are associated with needlessly retrying the processing of requests that would fail because of that 3rd-party service. Then, when requests are processed successfully, go back to regular unthrottled request processing. One library that implements this functionality is [NServiceBus](#).

The figure shows an area graph for resource use (a combination of memory, CPU, bandwidth, and other factors) against time for applications that are making use of three features. A feature is an area of functionality, such as a component that performs a specific set of tasks, a piece of code that performs a complex calculation, or an element that provides a service such as an in-memory cache. These features are labeled A, B, and C.

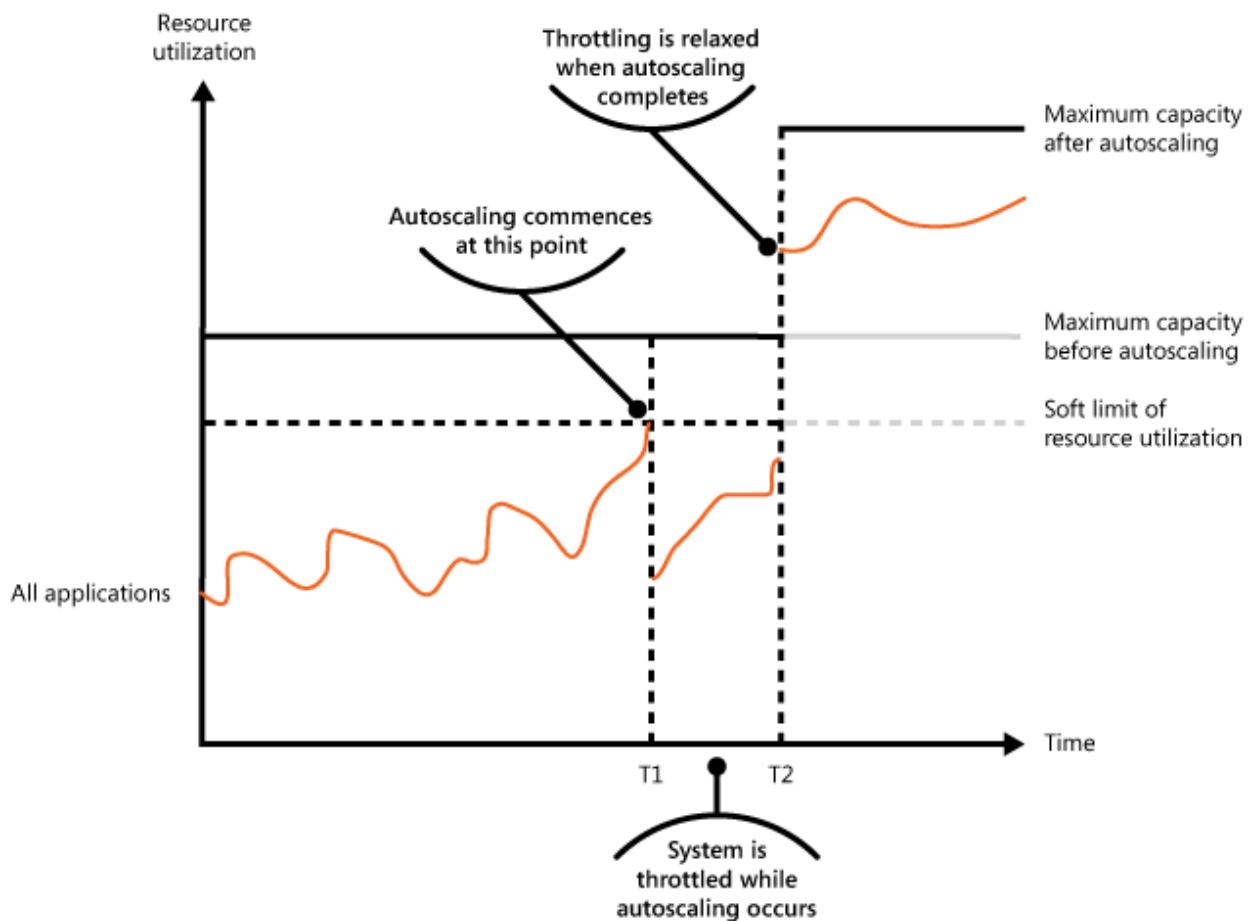


The area immediately below the line for a feature indicates the resources that are used by applications when they invoke this feature. For example, the area below the line for Feature A shows the resources used by applications that are making use of Feature A, and the area between the lines for Feature A and Feature B indicates the resources used by applications invoking Feature B. Aggregating the areas for each feature shows the total resource use of the system.

The previous figure illustrates the effects of deferring operations. Just prior to time T1, the total resources allocated to all applications using these features reach a threshold (the limit of resource use). At this point, the applications are in danger of exhausting the resources available. In this system, Feature B is less critical than Feature A or Feature C, so it's temporarily disabled and the resources that it was using are released. Between times T1 and T2, the applications using Feature A and Feature C continue running as normal. Eventually, the resource use of these two features diminishes to the point when, at time T2, there is sufficient capacity to enable Feature B again.

The autoscaling and throttling approaches can also be combined to help keep the applications responsive and within SLAs. If the demand is expected to remain high, throttling provides a temporary solution while the system scales out. At this point, the full functionality of the system can be restored.

The next figure shows an area graph of the overall resource use by all applications running in a system against time, and illustrates how throttling can be combined with autoscaling.



At time T1, the threshold specifying the soft limit of resource use is reached. At this point, the system can start to scale out. However, if the new resources don't become available quickly enough, then the existing resources might be exhausted and the system could fail. To prevent this from occurring, the system is temporarily throttled, as described earlier. When autoscaling has completed and the additional resources are available, throttling can be relaxed.

Issues and considerations

You should consider the following points when deciding how to implement this pattern:

- Throttling an application, and the strategy to use, is an architectural decision that impacts the entire design of a system. Throttling should be considered early in the application design process because it isn't easy to add once a system has been implemented.
- Throttling must be performed quickly. The system must be capable of detecting an increase in activity and react accordingly. The system must also be able to revert to its original state quickly after the load has eased. This requires that the appropriate performance data is continually captured and monitored.

- If a service needs to deny a user request temporarily, it should return a specific error code like 429 ("Too many requests") and 503 ("Server Too Busy") so the client application can understand that the reason for the refusal to serve a request is due to throttling.
- HTTP 429 indicates the calling application sent too many requests in a time window and exceeded a predetermined limit.
- HTTP 503 indicates the service isn't ready to handle the request. The common cause is that the service is experiencing temporary load spikes than expected.

The client application can wait for a period before retrying the request. A `Retry-After` HTTP header should be included, to support the client in choosing the retry strategy.

- Throttling can be used as a temporary measure while a system autoscales. In some cases it's better to simply throttle, rather than to scale, if a burst in activity is sudden and isn't expected to be long lived because scaling can add considerably to running costs.
- If throttling is being used as a temporary measure while a system autoscales, and if resource demands grow very quickly, the system might not be able to continue functioning—even when operating in a throttled mode. If this isn't acceptable, consider maintaining larger capacity reserves and configuring more aggressive autoscaling.
- Normalizing resource costs for different operations as they generally do not carry equal execution costs. For example, throttling limits might be lower for read operations and higher for write operations. Not considering the cost of an operation can result in exhausted capacity and exposing a potential attack vector.
- Dynamic configuration change of throttling behavior at runtime is desirable. If a system faces an abnormal load that the applied configuration cannot handle, throttling limits might need to increase or decrease to stabilize the system and keep up with the current traffic. Expensive, risky, and slow deployments are not desirable at this point. Using the [External Configuration Store pattern](#) throttling configuration is externalized and can be changed and applied without deployments.

When to use this pattern

Use this pattern:

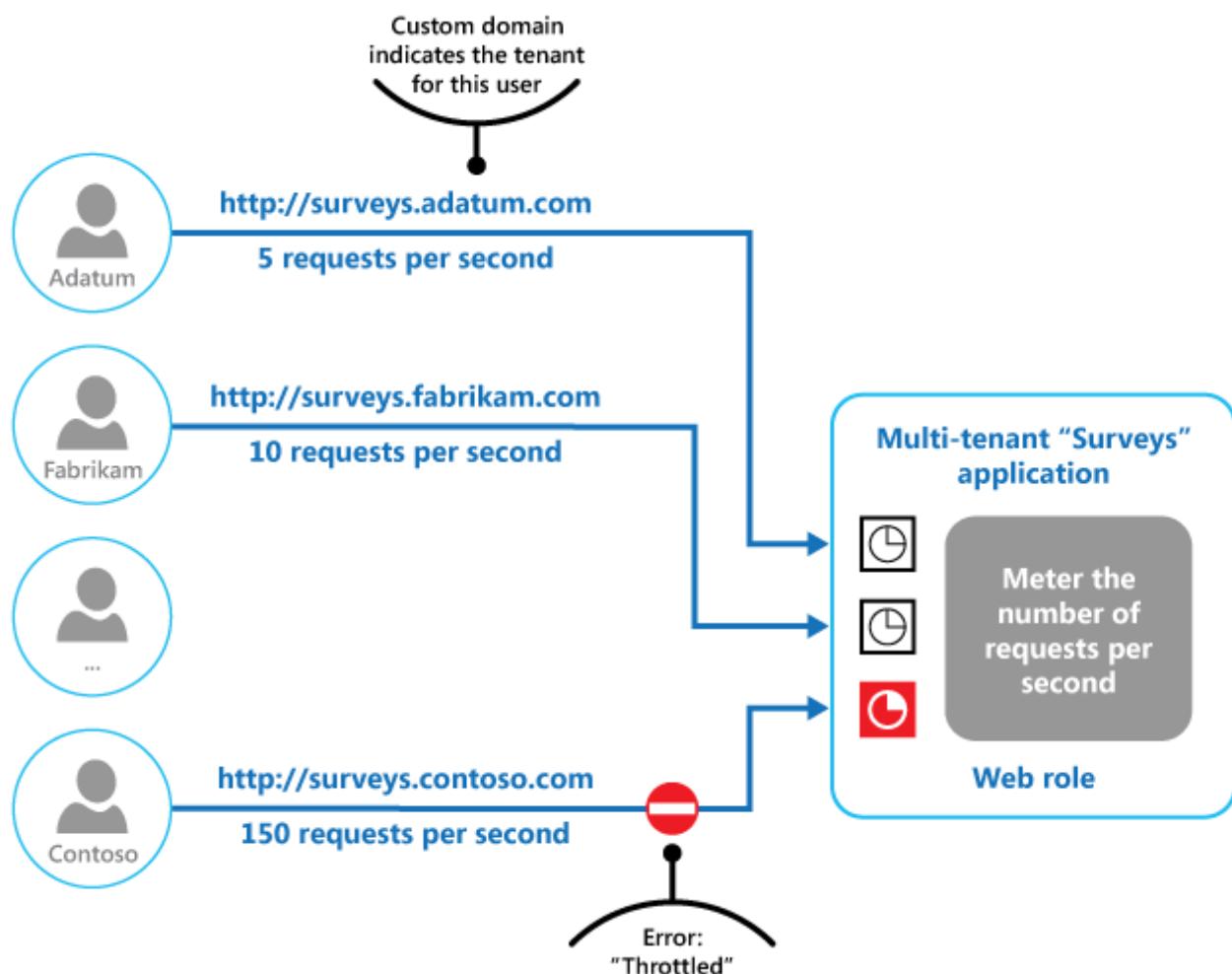
- To ensure that a system continues to meet service level agreements.

- To prevent a single tenant from monopolizing the resources provided by an application.
- To handle bursts in activity.
- To help cost-optimize a system by limiting the maximum resource levels needed to keep it functioning.

Example

The final figure illustrates how throttling can be implemented in a multi-tenant system. Users from each of the tenant organizations access a cloud-hosted application where they fill out and submit surveys. The application contains instrumentation that monitors the rate at which these users are submitting requests to the application.

In order to prevent the users from one tenant affecting the responsiveness and availability of the application for all other users, a limit is applied to the number of requests per second the users from any one tenant can submit. The application blocks requests that exceed this limit.



Next steps

The following guidance may also be relevant when implementing this pattern:

- [Instrumentation and Telemetry Guidance](#). Throttling depends on gathering information about how heavily a service is being used. Describes how to generate and capture custom monitoring information.
- [Service Metering Guidance](#). Describes how to meter the use of services in order to gain an understanding of how they are used. This information can be useful in determining how to throttle a service.
- [Autoscaling Guidance](#). Throttling can be used as an interim measure while a system autoscales, or to remove the need for a system to autoscale. Contains information on autoscaling strategies.

Related resources

The following patterns may also be relevant when implementing this pattern:

- [Queue-based Load Leveling pattern](#). Queue-based load leveling is a commonly used mechanism for implementing throttling. A queue can act as a buffer that helps to even out the rate at which requests sent by an application are delivered to a service.
- [Priority Queue pattern](#). A system can use priority queuing as part of its throttling strategy to maintain performance for critical or higher value applications, while reducing the performance of less important applications.
- [External Configuration Store pattern](#). Centralizing and externalizing the throttling policies provides the capability to change the configuration at runtime without the need for a redeployment. Services can subscribe to configuration changes, thereby applying the new configuration immediately, to stabilize a system.

Valet Key pattern

Azure Azure Storage

Use a token that provides clients with restricted direct access to a specific resource, in order to offload data transfer from the application. This is particularly useful in applications that use cloud-hosted storage systems or queues, and can minimize cost and maximize scalability and performance.

Context and problem

Client programs and web browsers often need to read and write files or data streams to and from an application's storage. Typically, the application will handle the movement of the data — either by fetching it from storage and streaming it to the client, or by reading the uploaded stream from the client and storing it in the data store. However, this approach absorbs valuable resources such as compute, memory, and bandwidth.

Data stores have the ability to handle upload and download of data directly, without requiring that the application perform any processing to move this data. But, this typically requires the client to have access to the security credentials for the store. This can be a useful technique to minimize data transfer costs and the requirement to scale out the application, and to maximize performance. It means, though, that the application is no longer able to manage the security of the data. After the client has a connection to the data store for direct access, the application can't act as the gatekeeper. It's no longer in control of the process and can't prevent subsequent uploads or downloads from the data store.

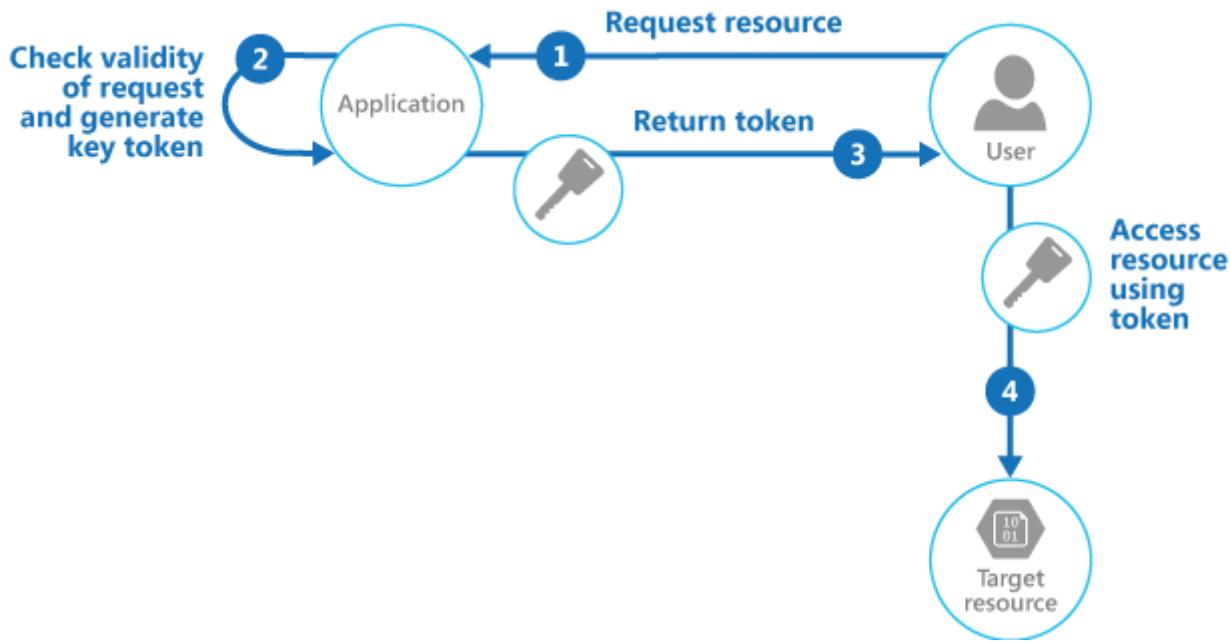
This isn't a realistic approach in distributed systems that need to serve untrusted clients. Instead, applications must be able to securely control access to data in a granular way, but still reduce the load on the server by setting up this connection and then allowing the client to communicate directly with the data store to perform the required read or write operations.

Solution

You need to resolve the problem of controlling access to a data store where the store can't manage authentication and authorization of clients. One typical solution is to restrict access to the data store's public connection and provide the client with a key or token that the data store can validate.

This key or token is usually referred to as a valet key. It provides time-limited access to specific resources and allows only predefined operations such as reading and writing to storage or queues, or uploading and downloading in a web browser. Applications can create and issue valet keys to client devices and web browsers quickly and easily, allowing clients to perform the required operations without requiring the application to directly handle the data transfer. This removes the processing overhead, and the impact on performance and scalability, from the application and the server.

The client uses this token to access a specific resource in the data store for only a specific period, and with specific restrictions on access permissions, as shown in the figure. After the specified period, the key becomes invalid and won't allow access to the resource.



It's also possible to configure a key that has other dependencies, such as the scope of the data. For example, depending on the data store capabilities, the key can specify a complete table in a data store, or only specific rows in a table. In cloud storage systems the key can specify a container, or just a specific item within a container.

The key can also be invalidated by the application. This is a useful approach if the client notifies the server that the data transfer operation is complete. The server can then invalidate that key to prevent further access.

Using this pattern can simplify managing access to resources because there's no requirement to create and authenticate a user, grant permissions, and then remove the user again. It also makes it easy to limit the location, the permission, and the validity period—all by simply generating a key at runtime. The important factors are to limit the validity period, and especially the location of the resource, as tightly as possible so that the recipient can only use it for the intended purpose.

Issues and considerations

Consider the following points when deciding how to implement this pattern:

Manage the validity status and period of the key. If leaked or compromised, the key effectively unlocks the target item and makes it available for malicious use during the validity period. A key can usually be revoked or disabled, depending on how it was issued. Server-side policies can be changed or, the server key it was signed with can be invalidated. Specify a short validity period to minimize the risk of allowing unauthorized operations to take place against the data store. However, if the validity period is too short, the client might not be able to complete the operation before the key expires. Allow authorized users to renew the key before the validity period expires if multiple accesses to the protected resource are required.

Control the level of access the key will provide. Typically, the key should allow the user to only perform the actions necessary to complete the operation, such as read-only access if the client shouldn't be able to upload data to the data store. For file uploads, it's common to specify a key that provides write-only permission, as well as the location and the validity period. It's critical to accurately specify the resource or the set of resources to which the key applies.

Consider how to control users' behavior. Implementing this pattern means some loss of control over the resources users are granted access to. The level of control that can be exerted is limited by the capabilities of the policies and permissions available for the service or the target data store. For example, it's usually not possible to create a key that limits the size of the data to be written to storage, or the number of times the key can be used to access a file. This can result in huge unexpected costs for data transfer, even when used by the intended client, and might be caused by an error in the code that causes repeated upload or download. To limit the number of times a file can be uploaded, where possible, force the client to notify the application when one operation has completed. For example, some data stores raise events the application code can use to monitor operations and control user behavior. However, it's hard to enforce quotas for individual users in a multi-tenant scenario where the same key is used by all the users from one tenant.

Validate, and optionally sanitize, all uploaded data. A malicious user that gains access to the key could upload data designed to compromise the system. Alternatively, authorized users might upload data that's invalid and, when processed, could result in an error or system failure. To protect against this, ensure that all uploaded data is validated and checked for malicious content before use.

Audit all operations. Many key-based mechanisms can log operations such as uploads, downloads, and failures. These logs can usually be incorporated into an audit process, and also used for billing if the user is charged based on file size or data volume. Use the logs to detect authentication failures that might be caused by issues with the key provider, or accidental removal of a stored access policy.

Deliver the key securely. It can be embedded in a URL that the user activates in a web page, or it can be used in a server redirection operation so that the download occurs automatically. Always use HTTPS to deliver the key over a secure channel.

Protect sensitive data in transit. Sensitive data delivered through the application will usually take place using TLS, and this should be enforced for clients accessing the data store directly.

Other issues to be aware of when implementing this pattern are:

- If the client doesn't, or can't, notify the server of completion of the operation, and the only limit is the expiration period of the key, the application won't be able to perform auditing operations such as counting the number of uploads or downloads, or preventing multiple uploads or downloads.
- The flexibility of key policies that can be generated might be limited. For example, some mechanisms only allow the use of a timed expiration period. Others aren't able to specify a sufficient granularity of read/write permissions.
- If the start time for the key or token validity period is specified, ensure that it's a little earlier than the current server time to allow for client clocks that might be slightly out of synchronization. The default, if not specified, is usually the current server time.
- The URL containing the key will be recorded in server log files. While the key will typically have expired before the log files are used for analysis, ensure that you limit access to them. If log data is transmitted to a monitoring system or stored in another location, consider implementing a delay to prevent leakage of keys until after their validity period has expired.
- If the client code runs in a web browser, the browser might need to support cross-origin resource sharing (CORS) to enable code that executes within the web browser to access data in a different domain from the one that served the page. Some older browsers and some data stores don't support CORS, and code that runs in these browsers might not be able to use a valet key to provide access to data in a different domain, such as a cloud storage account.

When to use this pattern

This pattern is useful for the following situations:

- To minimize resource loading and maximize performance and scalability. Using a valet key doesn't require the resource to be locked, no remote server call is required, there's no limit on the number of valet keys that can be issued, and it avoids a single point of failure resulting from performing the data transfer through the application code. Creating a valet key is typically a simple cryptographic operation of signing a string with a key.
- To minimize operational cost. Enabling direct access to stores and queues is resource and cost efficient, can result in fewer network round trips, and might allow for a reduction in the number of compute resources required.
- When clients regularly upload or download data, particularly where there's a large volume or when each operation involves large files.
- When the application has limited compute resources available, either due to hosting limitations or cost considerations. In this scenario, the pattern is even more helpful if there are many concurrent data uploads or downloads because it relieves the application from handling the data transfer.
- When the data is stored in a remote data store or a different datacenter. If the application was required to act as a gatekeeper, there might be a charge for the additional bandwidth of transferring the data between datacenters, or across public or private networks between the client and the application, and then between the application and the data store.

This pattern might not be useful in the following situations:

- If the application must perform some task on the data before it's stored or before it's sent to the client. For example, if the application needs to perform validation, log access success, or execute a transformation on the data. However, some data stores and clients are able to negotiate and carry out simple transformations such as compression and decompression (for example, a web browser can usually handle gzip formats).
- If the design of an existing application makes it difficult to incorporate the pattern. Using this pattern typically requires a different architectural approach for delivering and receiving data.
- If it's necessary to maintain audit trails or control the number of times a data transfer operation is executed, and the valet key mechanism in use doesn't support

notifications that the server can use to manage these operations.

- If it's necessary to limit the size of the data, especially during upload operations. The only solution to this is for the application to check the data size after the operation is complete, or check the size of uploads after a specified period or on a scheduled basis.

Example

Azure supports shared access signatures on Azure Storage for granular access control to data in blobs, tables, and queues, and for Service Bus queues and topics. A shared access signature token can be configured to provide specific access rights such as read, write, update, and delete to a specific table; a key range within a table; a queue; a blob; or a blob container. The validity can be a specified time period or with no time limit.

Azure shared access signatures also support server-stored access policies that can be associated with a specific resource such as a table or blob. This feature provides additional control and flexibility compared to application-generated shared access signature tokens, and should be used whenever possible. Settings defined in a server-stored policy can be changed and are reflected in the token without requiring a new token to be issued, but settings defined in the token can't be changed without issuing a new token. This approach also makes it possible to revoke a valid shared access signature token before it's expired.

Finally, creating shared access signatures can be done with a *user delegation key* instead of using the Storage account key. This method uses Microsoft Entra ID and supports Blob Storage only. It is the preferred solution when Blob Storage is the only service being accessed.

For more information, see [Grant limited access to Azure Storage resources using shared access signatures \(SAS\)](#).

The following code shows how to create a shared access signature token that's valid for five minutes. The `GetSharedAccessReferenceForUpload` method returns a shared access signatures token that can be used to upload a file to Azure Blob Storage.

C#

```
[ApiController]
public class SasController : ControllerBase
{
    private readonly string blobContainer = "valetkeysample";
    private readonly string blobEndpoint =
```

```

"https://<StorageAccountName>.blob.core.windows.net";
...
/// <summary>
/// Return a limited access key that allows the caller to upload a file
/// to this specific destination for about the next five minutes.
/// </summary>
private async Task<StorageEntitySas>
GetSharedAccessReferenceForUpload(string blobName)
{
    var blobServiceClient = new BlobServiceClient(new Uri(blobEndpoint), new
DefaultAzureCredential());

    var blobContainerClient =
blobServiceClient.GetBlobContainerClient(this.blobContainer);
    var blobClient = blobContainerClient.GetBlobClient(blobName);

    UserDelegationKey key = await
blobServiceClient.GetUserDelegationKeyAsync(DateTimeOffset.UtcNow,
DateTimeOffset.UtcNow.AddDays(1));

    var blobSasBuilder = new BlobSasBuilder
{
    BlobContainerName = blobClient.BlobContainerName,
    BlobName = blobClient.Name,
    Resource = "b",
    StartsOn = DateTimeOffset.UtcNow.AddMinutes(-5),
    ExpiresOn = DateTimeOffset.UtcNow.AddMinutes(5)
};
    blobSasBuilder.SetPermissions(BlobSasPermissions.Write);

    var storageSharedKeySignature = blobSasBuilder.ToSasQueryParameters(key,
blobServiceClient.AccountName).ToString();

    return new StorageEntitySas
{
    BlobUri = blobClient.Uri,
    Signature = storageSharedKeySignature
};
}

public class StorageEntitySas
{
    public string? Signature { get; internal set; }
    public Uri? BlobUri { get; internal set; }
}
}

```

The complete sample is available in the ValetKey solution available for download from [GitHub](#). The ValetKey.Web project in this solution contains a web application that includes the `SasController` class shown above. A sample client application that uses this web application to retrieve a shared access signatures key and upload a file to blob storage is available in the ValetKey.Client project.

Next steps

The following guidance might be relevant when implementing this pattern:

- A sample that demonstrates this pattern is available on [GitHub](#).
- [Grant limited access to Azure Storage resources using shared access signatures \(SAS\)](#)
- [Shared Access Signature Authentication with Service Bus](#)

Related resources

The following patterns might also be relevant when implementing this pattern:

- [Gatekeeper pattern](#). This pattern can be used in conjunction with the Valet Key pattern to protect applications and services by using a dedicated host instance that acts as a broker between clients and the application or service. The gatekeeper validates and sanitizes requests, and passes requests and data between the client and the application. Can provide an additional layer of security, and reduce the attack surface of the system.
- [Static Content Hosting pattern](#). Describes how to deploy static resources to a cloud-based storage service that can deliver these resources directly to the client to reduce the requirement for expensive compute instances. Where the resources aren't intended to be publicly available, the Valet Key pattern can be used to secure them.

Artificial intelligence (AI) architecture design

Article • 08/31/2023

Artificial intelligence (AI) is the capability of a computer to imitate intelligent human behavior. Through AI, machines can analyze images, comprehend speech, interact in natural ways, and make predictions using data.

Artificial Intelligence



Any technique that enables computers to mimic human intelligence. It includes *machine learning*

Machine Learning



A subset of AI that includes techniques that enable machines to improve at tasks with experience. It includes *deep learning*

Deep Learning



A subset of machine learning based on neural networks that permit a machine to train itself to perform a task.

AI concepts

Algorithm

An *algorithm* is a sequence of calculations and rules used to solve a problem or analyze a set of data. It is like a flow chart, with step-by-step instructions for questions to ask, but written in math and programming code. An algorithm may describe how to determine whether a pet is a cat, dog, fish, bird, or lizard. Another far more complicated algorithm may describe how to identify a written or spoken language, analyze its words, translate them into a different language, and then check the translation for accuracy.

Machine learning

Machine learning is an AI technique that uses mathematical algorithms to create predictive models. An algorithm is used to parse data fields and to "learn" from that data by using patterns found within it to generate models. Those models are then used to make informed predictions or decisions about new data.

The predictive models are validated against known data, measured by performance metrics selected for specific business scenarios, and then adjusted as needed. This process of learning and validation is called *training*. Through periodic retraining, ML models are improved over time.

- [What are the machine learning products at Microsoft?](#)

Deep learning

Deep learning is a type of ML that can determine for itself whether its predictions are accurate. It also uses algorithms to analyze data, but it does so on a larger scale than ML.

Deep learning uses artificial neural networks, which consist of multiple layers of algorithms. Each layer looks at the incoming data, performs its own specialized analysis, and produces an output that other layers can understand. This output is then passed to the next layer, where a different algorithm does its own analysis, and so on.

With many layers in each neural network-and sometimes using multiple neural networks-a machine can learn through its own data processing. This requires much more data and much more computing power than ML.

- [Deep learning versus machine learning](#)
- [Batch scoring of deep learning models on Azure](#)

Bots

A *bot* is an automated software program designed to perform a particular task. Think of it as a robot without a body. Early bots were comparatively simple, handling repetitive and voluminous tasks with relatively straightforward algorithmic logic. An example would be web crawlers used by search engines to automatically explore and catalog web content.

Bots have become much more sophisticated, using AI and other technologies to mimic human activity and decision-making, often while interacting directly with humans through text or even speech. Examples include bots that can take a dinner reservation,

chatbots (or conversational AI) that help with customer service interactions, and social bots that post breaking news or scientific data to social media sites.

Microsoft offers the Azure Bot Service, a managed service purpose-built for enterprise-grade bot development.

- [About Azure Bot Service](#)
- [Ten guidelines for responsible bots ↗](#)
- [Azure reference architecture: Enterprise-grade conversational bot](#)

Autonomous systems

Autonomous systems are part of an evolving new class that goes beyond basic automation. Instead of performing a specific task repeatedly with little or no variation (like bots do), autonomous systems bring intelligence to machines so they can adapt to changing environments to accomplish a desired goal.

Smart buildings use autonomous systems to automatically control operations like lighting, ventilation, air conditioning, and security. A more sophisticated example would be a self-directed robot exploring a collapsed mine shaft to thoroughly map its interior, determine which portions are structurally sound, analyze the air for breathability, and detect signs of trapped miners in need of rescue—all without a human monitoring in real time on the remote end.

- [Autonomous systems and solutions from Microsoft AI ↗](#)

General info on Microsoft AI

Learn more about Microsoft AI, and keep up-to-date with related news:

- [Microsoft AI School ↗](#)
- [Azure AI platform page ↗](#)
- [Microsoft AI platform page ↗](#)
- [Microsoft AI Blog ↗](#)
- [Microsoft AI on GitHub: Samples, reference architectures, and best practices ↗](#)
- [Azure Architecture Center](#)

High-level architectural types

Prebuilt AI

Prebuilt AI is exactly what it sounds like—off-the-shelf AI models, services, and APIs that are ready to use. These help you add intelligence to apps, websites, and flows without having to gather data and then build, train, and publish your own models.

One example of prebuilt AI might be a pretrained model that can be incorporated as is or used to provide a baseline for further custom training. Another example would be a cloud-based API service that can be called at will to process natural language in a desired fashion.

Azure Cognitive Services

[Cognitive Services](#) provide developers the opportunity to use prebuilt APIs and integration toolkits to create applications that can see, hear, speak, understand, and even begin to reason. The catalog of services within Cognitive Services can be categorized into five main pillars: Vision, Speech, Language, Web Search, and Decision/Recommendation.

- [Azure Cognitive Services documentation](#)
- [Try Azure Cognitive Services for free](#)
- [Choosing an Azure Cognitive Services technology](#)
- [Choosing a natural language processing technology in Azure](#)

Prebuilt AI models in AI Builder

AI Builder is a new capability in [Microsoft Power Platform](#) that provides a point-and-click interface for adding AI to your apps, even if you have no coding or data science skills. (Some features in AI Builder have not yet released for general availability and remain in preview status. For more information, refer to the [Feature availability by region](#) page.)

You can build and train your own models, but AI Builder also provides [select prebuilt AI models](#) that are ready for use right away. For example, you can add a component in Microsoft Power Apps based on a prebuilt model that recognizes contact information from business cards.

- [Power Apps on Azure](#)

- [AI Builder documentation](#)
- [AI model types in AI Builder](#)
- [Overview of prebuilt AI models in AI Builder](#)

Custom AI

Although prebuilt AI is useful (and increasingly flexible), the best way to get what you need from AI is probably to build a system yourself. This is obviously a very deep and complex subject, but let's look at some basic concepts beyond what we've just covered.

Code languages

The core concept of AI is the use of algorithms to analyze data and generate models to describe (or *score*) it in ways that are useful. Algorithms are written by developers and data scientists (and sometimes by other algorithms) using programming code. Two of the most popular programming languages for AI development are currently Python and R.

[Python](#) is a general-purpose, high-level programming language. It has a simple, easy-to-learn syntax that emphasizes readability. There is no compiling step. Python has a large standard library, but it also supports the ability to add modules and packages. This encourages modularity and lets you expand capabilities when needed. There is a large and growing ecosystem of AI and ML libraries for Python, including many that are readily available in Azure.

- [Python on Azure product home page](#)
- [Azure for Python developers](#)
- [Azure Machine Learning SDK for Python](#)
- [Introduction to machine learning with Python and Azure Notebooks](#)
- [scikit-learn.](#) An open-source ML library for Python
- [PyTorch.](#) An open-source Python library with a rich ecosystem that can be used for deep learning, computer vision, natural language processing, and more
- [TensorFlow.](#) An open-source symbolic math library also used for ML applications and neural networks

- [Tutorial: Apply machine learning models in Azure Functions with Python and TensorFlow](#)

R is a language and environment [for statistical computing and graphics](#). It can be used for everything from mapping broad social and marketing trends online to developing financial and climate models.

Microsoft has fully embraced the R programming language and provides many different options for R developers to run their code in Azure.

- [Use R interactively on Azure Machine Learning](#).
- [Tutorial: Create a logistic regression model in R with Azure Machine Learning](#)

Training

Training is core to machine learning. It is the iterative process of "teaching" an algorithm to create models, which are used to analyze data and then make accurate predictions from it. In practice, this process has three general phases: training, validation, and testing.

During the training phase, a quality set of known data is tagged so that individual fields are identifiable. The tagged data is fed to an algorithm configured to make a particular prediction. When finished, the algorithm outputs a model that describes the patterns it found as a set of parameters. During validation, fresh data is tagged and used to test the model. The algorithm is adjusted as needed and possibly put through more training. Finally, the testing phase uses real-world data without any tags or preselected targets. Assuming the model's results are accurate, it is considered ready for use and can be deployed.

- [Train models with Azure Machine Learning](#)

Hyperparameter tuning

Hyperparameters are data variables that govern the training process itself. They are configuration variables that control how the algorithm operates. Hyperparameters are thus typically set before model training begins and are not modified within the training process in the way that parameters are. Hyperparameter tuning involves running trials within the training task, assessing how well they are getting the job done, and then adjusting as needed. This process generates multiple models, each trained using different families of hyperparameters.

- [Tune hyperparameters for your model with Azure Machine Learning](#)

Model selection

The process of training and hyperparameter tuning produces numerous candidate models. These can have many different variances, including the effort needed to prepare the data, the flexibility of the model, the amount of processing time, and of course the degree of accuracy of its results. Choosing the best trained model for your needs and constraints is called *model selection*, but this is as much about preplanning before training as it is about choosing the one that works best.

Automated machine learning (AutoML)

Automated machine learning, also known as AutoML, is the process of automating the time-consuming, iterative tasks of machine learning model development. It can significantly reduce the time it takes to get production-ready ML models. Automated ML can assist with model selection, hyperparameter tuning, model training, and other tasks, without requiring extensive programming or domain knowledge.

- [What is automated machine learning?](#)

Scoring

Scoring is also called *prediction* and is the process of generating values based on a trained machine learning model, given some new input data. The values, or scores, that are created can represent predictions of future values, but they might also represent a likely category or outcome. The scoring process can generate many different types of values:

- A list of recommended items and a similarity score
- Numeric values, for time series models and regression models
- A probability value, indicating the likelihood that a new input belongs to some existing category
- The name of a category or cluster to which a new item is most similar
- A predicted class or outcome, for classification models

Batch scoring is when data is collected during some fixed period of time and then processed in a batch. This might include generating business reports or analyzing customer loyalty.

Real-time scoring is exactly that—scoring that is ongoing and performed as quickly as possible. The classic example is credit card fraud detection, but real-time scoring can

also be used in speech recognition, medical diagnoses, market analyses, and many other applications.

General info on custom AI on Azure

- Microsoft AI on GitHub: Samples, reference architectures, and best practices [↗](#)
- Azure Machine Learning SDK for Python
- Azure Machine Learning Python SDK notebooks [↗](#). A GitHub repo of example notebooks demonstrating the Azure Machine Learning Python SDK.
- Train R models using the Azure ML CLI (v2) [↗](#)

Azure AI platform offerings

Following is a breakdown of Azure technologies, platforms, and services you can use to develop AI solutions for your needs.

Azure Machine Learning

This is an enterprise-grade machine learning service to build and deploy models faster. Azure Machine Learning offers web interfaces and SDKs so you can quickly train and deploy your machine learning models and pipelines at scale. Use these capabilities with open-source Python frameworks, such as PyTorch, TensorFlow, and scikit-learn.

- What are the machine learning products at Microsoft?
- Azure Machine Learning product home page [↗](#)
- Azure Machine Learning documentation overview
- What is Azure Machine Learning? General orientation with links to many learning resources, SDKs, documentation, and more

Machine learning reference architectures for Azure

- Batch scoring of Python machine learning models on Azure
- Batch scoring of deep learning models on Azure
- Machine learning operationalization (MLOps) for Python models using Azure Machine Learning

- Batch scoring of R machine learning models on Azure
- Batch scoring of Spark machine learning models on Azure Databricks
- Enterprise-grade conversational bot
- Build a real-time recommendation API on Azure

Azure automated machine learning

Azure provides extensive support for automated ML. Developers can build models using a no-code UI or through a code-first notebooks experience.

- [Azure automated machine learning product home page ↗](#)
- [Azure automated ML infographic \(PDF\) ↗](#)
- [Tutorial: Create a classification model with automated ML in Azure Machine Learning](#)
- [Tutorial: Use automated machine learning to predict taxi fares](#)
- [Configure automated ML experiments in Python](#)
- [Use the CLI extension for Azure Machine Learning](#)
- [Automate machine learning activities with the Azure Machine Learning CLI](#)

Azure Cognitive Services

This is a comprehensive family of AI services and cognitive APIs to help you build intelligent apps. These domain-specific, pretrained AI models can be customized with your data.

- [Cognitive Services product home page ↗](#)
- [Azure Cognitive Services documentation](#)

Azure Cognitive Search

This is an AI-powered cloud search service for mobile and web app development. The service can search over private heterogenous content, with options for AI enrichment if your content is unstructured or unsearchable in raw form.

- [Azure Cognitive Search product home page ↗](#)

- [Getting started with AI enrichment](#)
- [Azure Cognitive Search documentation overview](#)
- [Choosing a natural language processing technology in Azure](#)
- [Quickstart: Create an Azure Cognitive Search cognitive skill set in the Azure portal](#)

Azure Bot Service

This is a purpose-built bot development environment with out-of-the-box templates to get started quickly.

- [Azure Bot Service product home page ↗](#)
- [Azure Bot Service documentation overview](#)
- [Azure reference architecture: Enterprise-grade conversational bot](#)
- [Microsoft Bot Framework ↗](#)
- [Microsoft Bot Framework SDK repo ↗](#)

Apache Spark on Azure

Apache Spark is a parallel processing framework that supports in-memory processing to boost the performance of big data analytic applications. Spark provides primitives for in-memory cluster computing. A Spark job can load and cache data into memory and query it repeatedly, which is much faster than disk-based applications, such as Hadoop.

[Apache Spark in Azure HDInsight](#) is the Microsoft implementation of Apache Spark in the cloud. Spark clusters in HDInsight are compatible with Azure Storage and Azure Data Lake Storage, so you can use HDInsight Spark clusters to process your data stored in Azure.

The Microsoft machine learning library for Apache Spark is [SynapseML ↗](#) (formerly known as MMLSpark). This open-source library adds many deep learning and data science tools, networking capabilities, and production-grade performance to the Spark ecosystem. [Learn more about SynapseML features and capabilities](#).

- [Azure HDInsight overview](#). Basic information about features, cluster architecture, and use cases, with pointers to quickstarts and tutorials.
- [Tutorial: Build an Apache Spark machine learning application in Azure HDInsight](#)

- Apache Spark best practices on HDInsight
- Configure HDInsight Apache Spark Cluster settings
- Machine learning on HDInsight
- GitHub repo for SynapseML: Microsoft machine learning library for Apache Spark ↗
- Create an Apache Spark machine learning pipeline on HDInsight

Azure Databricks Runtime for Machine Learning

Azure Databricks ↗ is an Apache Spark–based analytics platform with one-click setup, streamlined workflows, and an interactive workspace for collaboration between data scientists, engineers, and business analysts.

Databricks Runtime for Machine Learning (Databricks Runtime ML) lets you start a Databricks cluster with all of the libraries required for distributed training. It provides a ready-to-go environment for machine learning and data science. Plus, it contains multiple popular libraries, including TensorFlow, PyTorch, Keras, and XGBoost. It also supports distributed training using Horovod.

- Azure Databricks product home page ↗
- Azure Databricks documentation
- Machine learning capabilities in Azure Databricks
- How-to guide: Databricks Runtime for Machine Learning
- Batch scoring of Spark machine learning models on Azure Databricks
- Deep learning overview for Azure Databricks

Customer stories

Different industries are applying AI in innovative and inspiring ways. Following are a number of customer case studies and success stories:

- ASOS: Online retailer solves challenges with Azure Machine Learning service ↗
- KPMG helps financial institutions save millions in compliance costs with Azure Cognitive Services ↗
- Volkswagen: Machine translation speaks Volkswagen – in 40 languages ↗

- Buncee: NYC school empowers readers of all ages and abilities with Azure AI ↗
- InterSystems: Data platform company boosts healthcare IT by generating critical information at unprecedented speed ↗
- Zencity: Data-driven startup uses funding to help local governments support better quality of life for residents ↗
- Bosch uses IoT innovation to drive traffic safety improvements by helping drivers avoid serious accidents ↗
- Automation Anywhere: Robotic process automation platform developer enriches its software with Azure Cognitive Services ↗
- Wix deploys smart, scalable search across 150 million websites with Azure Cognitive Search ↗
- Asklepios Klinik Altona: Precision surgeries with Microsoft HoloLens 2 and 3D visualization ↗
- AXA Global P&C: Global insurance firm models complex natural disasters with cloud-based HPC ↗

Browse more AI customer stories ↗

Next steps

- To learn about the artificial intelligence development products available from Microsoft, refer to the [Microsoft AI platform](#) ↗ page.
- For training in how to develop AI solutions, refer to [Microsoft AI School](#) ↗.
- [Microsoft AI on GitHub](#): Samples, reference architectures, and best practices ↗ organizes the Microsoft open source AI-based repositories, providing tutorials and learning materials.

Analytics architecture design

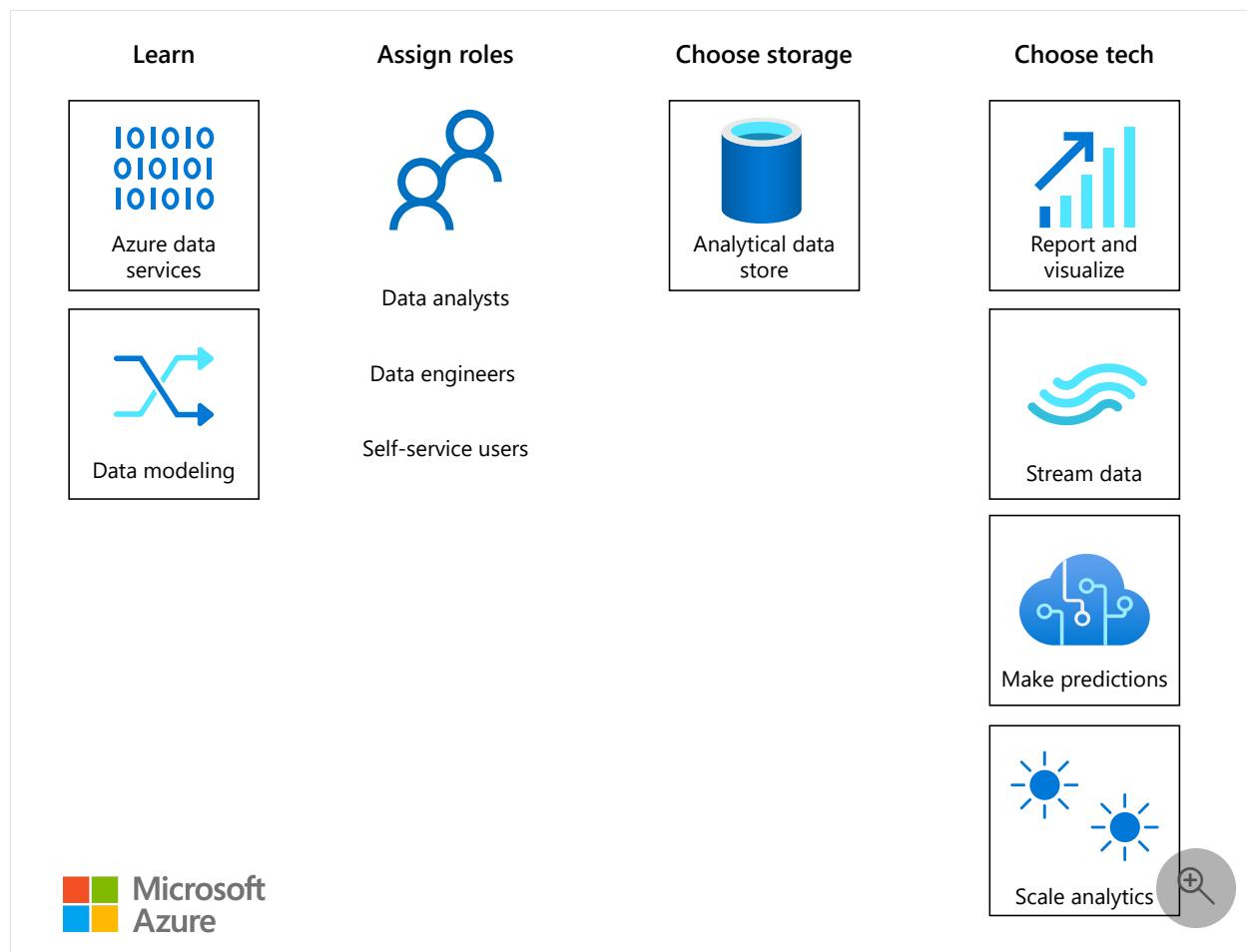
Azure Synapse Analytics

Power BI

With the exponential growth in data, organizations rely on the limitless compute, storage, and analytical power of Azure to scale, stream, predict, and see their data. Analytics solutions turn volumes of data into useful business intelligence (BI), such as reports and visualizations, and inventive artificial intelligence (AI), such as forecasts based on machine learning.

Whether your organization is just starting to evaluate cloud-based analytics tools or is looking to expand your current implementation, Azure provides many options. The workflow starts with learning about common approaches and aligning processes and roles around a cloud mindset.

Data can be processed in batches or in real-time, on-premises or in the cloud, but the goal of any analytics solution is to make use of data at scale. Increasingly, organizations want to create a single source of truth for all the relational and nonrelational data being generated by people, machines, and the Internet of Things (IoT). It's common to use a [big data architecture](#) or an [IoT architecture](#) to transform raw data into a structured form, then move it to an analytical data store. This store becomes the single source of truth that can power a multitude of insightful analytics solutions.



Download a [Visio file](#) of this architecture.

Learn about analytics on Azure

If you're new to analytics on Azure, the best place to learn more is with [Microsoft Learn](#), a free, online training platform. You'll find videos, tutorials, and hands-on learning for specific products and services, plus learning paths based on your job role, such as developer or data analyst.

- [Browse Azure data topics](#)
- [Explore Azure database and analytics services](#)

Organizational readiness

If your organization is new to the cloud, the [Cloud Adoption Framework](#) can help you get started. This collection of documentation and best practices offers proven guidance from Microsoft designed to accelerate your cloud adoption journey. It also lists [innovation tools to democratize data in Azure](#).

To help assure the quality of your analytics solution on Azure, we recommend following the [Azure Well-Architected Framework](#). It provides prescriptive guidance for organizations seeking architectural excellence and discusses how to design, provision, and monitor cost-optimized Azure solutions.

Path to production

Knowing how to [store your data](#) is one of the first decisions you need to make in your journey to analytics on Azure. Then you can choose the best [data analytics technology](#) for your scenario.

To get started, consider the following example implementations:

- [Analytics end-to-end with Azure Synapse](#)
- [Automated enterprise BI](#)
- [Data warehousing and analytics](#)
- [Geospatial data processing and analytics](#)
- [Stream processing with Azure Databricks](#)
- [Stream processing with Azure Stream Analytics](#)

Best practices

High-quality analytics start with robust, trustworthy data. At the highest level, [information security](#) practices help ensure that your data is protected in transit and at rest. Access to that data must also be trusted. Trustworthy data implies a design that implements:

- [Governance policies.](#)
- [Identity and access management.](#)
- [Network security controls.](#)
- [Data protection.](#)

At the platform level, the following [big data best practices](#) contribute to trustworthy analytics on Azure:

- Orchestrate data ingestion using a data workflow or pipeline solution such as those supported by Azure Data Factory or Oozie.

- Process data in place using a distributed data store, a big data approach that supports larger volumes of data and a greater variety of formats.
- Scrub sensitive data early as part of the ingestion workflow to avoid storing it in your data lake.
- Consider the total cost of the required Azure resources by balancing the per-unit cost of the compute nodes needed to the per-minute cost of using those nodes to complete a job.
- Create a data lake that combines storage for files in multiple formats, whether structured, semi-structured, or unstructured. At Microsoft, we use Azure Data Lake Storage Gen2 as our single source of truth. For example, see [BI solution architecture in the Center of Excellence](#).

Additional resources

Analytics is a broad category and covers a range of solutions. The following resources can help you discover more about Azure.

Hybrid

The vast majority of organizations need a hybrid approach to analytics because their data is hosted both on-premises and in the cloud. Organizations often [extend on-premises data solutions to the cloud](#). To connect environments, organizations must [choose a hybrid network architecture](#).

A hybrid approach might include mainframe and midrange systems as a data source for Azure solutions. For example, your organization may want to [modernize mainframe and midrange data](#) or provide [mainframe access to Azure databases](#).

Example solutions

Here are a few sample implementations of analytics on Azure to consider:

- [Big data analytics with Azure Data Explorer](#)
- [IoT analytics with Azure Data Explorer](#)
- [Real time analytics on big data architecture](#)
- [Modern analytics architecture with Azure Databricks](#)

- [Big data analytics with enterprise-grade security using Azure Synapse](#)
- [Browse more analytics examples in the Azure Architecture Center](#)

AWS or Google Cloud professionals

These articles can help you ramp up quickly by comparing Azure analytics options to other cloud services:

- [Analytics and big data in Database technologies on Azure and AWS](#)
- [Big data and analytics in Google Cloud to Azure services comparison](#)

Choose an analytical data store in Azure

Article • 10/09/2023

In a [big data](#) architecture, there is often a need for an analytical data store that serves processed data in a structured format that can be queried using analytical tools. Analytical data stores that support querying of both hot-path and cold-path data are collectively referred to as the serving layer, or data serving storage.

The serving layer deals with processed data from both the hot path and cold path. In the [lambda architecture](#), the serving layer is subdivided into a *speed serving* layer, which stores data that has been processed incrementally, and a *batch serving* layer, which contains the batch-processed output. The serving layer requires strong support for random reads with low latency. Data storage for the speed layer should also support random writes, because batch loading data into this store would introduce undesired delays. On the other hand, data storage for the batch layer does not need to support random writes, but batch writes instead.

There is no single best data management choice for all data storage tasks. Different data management solutions are optimized for different tasks. Most real-world cloud apps and big data processes have a variety of data storage requirements and often use a combination of data storage solutions.

What are your options when choosing an analytical data store?

There are several options for data serving storage in Azure, depending on your needs:

- [Azure Synapse Analytics](#)
- [Azure Synapse Spark pools](#)
- [Azure Databricks](#)
- [Azure Data Explorer](#)
- [Azure SQL Database](#)
- [SQL Server in Azure VM](#)
- [HBase/Phoenix on HDInsight](#)
- [Hive LLAP on HDInsight](#)
- [Azure Analysis Services](#)
- [Azure Cosmos DB](#)

These options provide various database models that are optimized for different types of tasks:

- [Key/value](#) databases hold a single serialized object for each key value. They're good for storing large volumes of data where you want to get one item for a given key value and you don't have to query based on other properties of the item.
- [Document](#) databases are key/value databases in which the values are *documents*. A "document" in this context is a collection of named fields and values. The database typically stores the data in a format such as XML, YAML, JSON, or BSON, but may use plain text.

Document databases can query on non-key fields and define secondary indexes to make querying more efficient. This makes a document database more suitable for applications that need to retrieve data based on criteria more complex than the value of the document key. For example, you could query on fields such as product ID, customer ID, or customer name.

- **Column store** databases are key/value data stores that store each column separately on disk. A *wide column store* database is a type of column store database that stores *column families*, not just single columns. For example, a census database might have a column family for a person's name (first, middle, last), a family for the person's address, and a family for the person's profile information (date of birth, gender). The database can store each column family in a separate partition, while keeping all the data for one person related to the same key. An application can read a single column family without reading through all of the data for an entity.
- **Graph** databases store information as a collection of objects and relationships. A graph database can efficiently perform queries that traverse the network of objects and the relationships between them. For example, the objects might be employees in a human resources database, and you might want to facilitate queries such as "find all employees who directly or indirectly work for Scott."
- Telemetry and time series databases are an append-only collection of objects. Telemetry databases efficiently index data in a variety of column stores and in-memory structures, making them the optimal choice for storing and analyzing vast quantities of telemetry and time series data.

Key selection criteria

To narrow the choices, start by answering these questions:

- Do you need serving storage that can serve as a hot path for your data? If yes, narrow your options to those that are optimized for a speed serving layer.
- Do you need massively parallel processing (MPP) support, where queries are automatically distributed across several processes or nodes? If yes, select an option that supports query scale-out.
- Do you prefer to use a relational data store? If so, narrow your options to those with a relational database model. However, note that some non-relational stores support SQL syntax for querying, and tools such as PolyBase can be used to query non-relational data stores.
- Do you collect time series data? Do you use append-only data?

Capability matrix

The following tables summarize the key differences in capabilities.

General capabilities

Capability	SQL Database	Azure Synapse SQL pool	Azure Synapse Spark pool	Azure Data Explorer	HBase/Phoenix on HDInsight	Hive LLAP on HDInsight	Azure Analysis Services	Azure Cosmos DB
Is managed service	Yes	Yes	Yes	Yes	Yes ¹	Yes ¹	Yes	Yes
Primary database model	Relational (column store format when using columnstore indexes)	Relational tables with column storage	Wide column store	Relational (column store), telemetry, and time series store	Wide column store	Hive/In-Memory	Tabular semantic models	Document store, graph, key-value store, wide column store
SQL language support	Yes	Yes	Yes	Yes	Yes (using Phoenix ² JDBC driver)	Yes	No	Yes
Optimized for speed serving layer	Yes ²	Yes ³	Yes	Yes	Yes	Yes	No	Yes

[1] With manual configuration and scaling.

[2] Using memory-optimized tables and hash or nonclustered indexes.

[3] Supported as an Azure Stream Analytics output.

Scalability capabilities

Capability	SQL Database	Azure Synapse SQL pool	Azure Synapse Spark pool	Azure Data Explorer	HBase/Phoenix on HDInsight	Hive LLAP on HDInsight	Azure Analysis Services	Azure Cosmos DB
Redundant regional servers for high availability	Yes	No	No	Yes	Yes	No	Yes	Yes
Supports query scale-out	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dynamic scalability (scale up)	Yes	Yes	Yes	Yes	No	No	Yes	Yes

Capability	SQL Database	Azure Synapse SQL pool	Azure Synapse Spark pool	Azure Data Explorer	HBase/Phoenix on HDInsight	Hive LLAP on HDInsight	Azure Analysis Services	Azure Cosmos DB
Supports in-memory caching of data	Yes	Yes	Yes	Yes	No	Yes	Yes	No

Security capabilities

Capability	SQL Database	Azure Synapse	Azure Data Explorer	HBase/Phoenix on HDInsight	Hive LLAP on HDInsight	Azure Analysis Services	Azure Cosmos DB
Authentication	SQL / Microsoft Entra ID	SQL / Microsoft Entra ID	Microsoft Entra ID	local / Microsoft Entra ID ¹	local / Microsoft Entra ID ¹	Microsoft Entra ID	database users / Microsoft Entra ID via access control (IAM)
Data encryption at rest	Yes ²	Yes ²	Yes	Yes ¹	Yes ¹	Yes	Yes
Row-level security	Yes	Yes ³	Yes	Yes ¹	Yes ¹	Yes	No
Supports firewalls	Yes	Yes	Yes	Yes ⁴	Yes ⁴	Yes	Yes
Dynamic data masking	Yes	Yes	Yes	Yes ¹	Yes	No	No

[1] Requires using a [domain-joined HDInsight cluster](#).

[2] Requires using transparent data encryption (TDE) to encrypt and decrypt your data at rest.

[3] Filter predicates only. See [Row-Level Security](#)

[4] When used within an Azure Virtual Network. See [Extend Azure HDInsight using an Azure Virtual Network](#).

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Zoiner Tejada](#) | CEO and Architect

Next steps

- [Analyze data in a relational data warehouse](#)
- [Create a single database - Azure SQL Database](#)
- [Create an Azure Databricks workspace](#)
- [Create Apache Spark cluster in Azure HDInsight using Azure portal](#)
- [Creating a Synapse workspace](#)
- [Explore Azure data services for modern analytics](#)
- [Explore Azure database and analytics services](#)
- [Query Azure Cosmos DB by using the API for NoSQL](#)

Related resources

- [Technology choices for Azure solutions](#)
- [Advanced analytics architecture](#)
- [Analyze operational data on MongoDB Atlas using Azure Synapse Analytics](#)
- [Non-relational data and NoSQL](#)

Choose a data analytics and reporting technology in Azure

Article • 03/14/2023

The goal of most big data solutions is to provide insights into the data through analysis and reporting. This can include preconfigured reports and visualizations, or interactive data exploration.

What are your options when choosing a data analytics technology?

There are several options for analysis, visualizations, and reporting in Azure, depending on your needs:

- [Power BI](#)
- [Jupyter Notebooks](#)
- [Zeppelin Notebooks](#)
- [Jupyter Notebooks in VS Code](#)

Power BI

[Power BI](#) is a suite of business analytics tools. It can connect to hundreds of data sources, and can be used for ad hoc analysis. See [this list](#) of the currently available data sources. Use [Power BI Embedded](#) to integrate Power BI within your own applications without requiring any additional licensing.

Organizations can use Power BI to produce reports and publish them to the organization. Everyone can create personalized dashboards, with governance and [security built in](#). Power BI uses [Azure Active Directory \(Azure AD\)](#) to authenticate users who log in to the Power BI service, and uses the Power BI login credentials whenever a user attempts to access resources that require authentication.

Jupyter Notebooks

[Jupyter Notebooks](#) provide a browser-based shell that lets data scientists create *notebook* files that contain Python, Scala, or R code and markdown text, making it an effective way to collaborate by sharing and documenting code and results in a single document.

Most varieties of HDInsight clusters, such as Spark or Hadoop, come [preconfigured with Jupyter notebooks](#) for interacting with data and submitting jobs for processing. Depending on the type of HDInsight cluster you are using, one or more kernels will be provided for interpreting and running your code. For example, Spark clusters on HDInsight provide Spark-related kernels that you can select from to execute Python or Scala code using the Spark engine.

Jupyter notebooks provide a great environment for analyzing, visualizing, and processing your data prior to building more advanced visualizations with a BI/reporting tool like Power BI.

Zeppelin Notebooks

[Zeppelin Notebooks](#) are another option for a browser-based shell, similar to Jupyter in functionality. Some HDInsight clusters come [preconfigured with Zeppelin notebooks](#). However, if you are using an [HDInsight Interactive Query](#) (Hive LLAP) cluster, [Zeppelin](#) is currently your only choice of notebook that you can use to run interactive Hive queries. Also, if you are using a [domain-joined HDInsight cluster](#), Zeppelin notebooks are the only type that enables you to assign different user logins to control access to notebooks and the underlying Hive tables.

Jupyter Notebooks in VS Code

VS Code is a free code editor and development platform that you can use locally or connected to remote compute. Combined with the Jupyter extension, it offers a full environment for Jupyter development that can be enhanced with additional language extensions. If you want a best-in-class, free Jupyter experience with the ability to leverage your compute of choice, this is a great option. Using VS Code, you can develop and run notebooks against remotes and containers. To make the transition easier from Azure Notebooks, we have made the container image available so it can be used with VS Code too.

Jupyter (formerly IPython Notebook) is an open-source project that lets you easily combine Markdown text and executable Python source code on one canvas called a notebook. Visual Studio Code supports working with Jupyter Notebooks natively, and through Python code files.

Key selection criteria

To narrow the choices, start by answering these questions:

- Do you need to connect to numerous data sources, providing a centralized place to create reports for data spread throughout your domain? If so, choose an option that allows you to connect to 100s of data sources.
- Do you want to embed dynamic visualizations in an external website or application? If so, choose an option that provides embedding capabilities.
- Do you want to design your visualizations and reports while offline? If yes, choose an option with offline capabilities.
- Do you need heavy processing power to train large or complex AI models or work with very large data sets? If yes, choose an option that can connect to a big data cluster.

Capability matrix

The following tables summarize the key differences in capabilities.

General capabilities

Capability	Power BI	Jupyter Notebooks	Zeppelin Notebooks	Jupyter Notebooks in VS Code
Connect to big data cluster for advanced processing	Yes	Yes	Yes	No
Managed service	Yes	Yes ¹	Yes ¹	Yes
Connect to 100s of data sources	Yes	No	No	No
Offline capabilities	Yes ²	No	No	No
Embedding capabilities	Yes	No	No	No
Automatic data refresh	Yes	No	No	No
Access to numerous open source packages	No	Yes ³	Yes ³	Yes ⁴
Data transformation/cleansing options	Power Query ² , R	40 languages, including Python, R, Julia, and Scala	20+ interpreters, including Python, JDBC, and R	Python, F#, R

Capability	Power BI	Jupyter Notebooks	Zeppelin Notebooks	Jupyter Notebooks in VS Code
Pricing	Free for Power BI Desktop (authoring), see pricing for hosting options	Free	Free	Free
Multiuser collaboration	Yes	Yes (through sharing or with a multiuser server like JupyterHub)	Yes	Yes (through sharing)

[1] When used as part of a managed HDInsight cluster.

[2] With the use of Power BI Desktop.

[2] You can search the [Maven repository](#) for community-contributed packages.

[3] Python packages can be installed using either pip or conda. R packages can be installed from CRAN or GitHub. Packages in F# can be installed via nuget.org using the [Paket dependency manager](#).

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Zoiner Tejada](#) | CEO and Architect

Next steps

- [Get started with Jupyter notebooks for Python](#)
- [Notebooks](#)
- [Run Azure Databricks Notebooks with Azure Data Factory](#)
- [Run Jupyter notebooks in your workspace](#)
- [What is Power BI?](#)

Related resources

- Advanced analytics architecture
- Data analysis and visualization in an Azure industrial IoT analytics solution
- Technology choices for Azure solutions

Choose a batch processing technology in Azure

Article • 10/09/2023

Big data solutions often use long-running batch jobs to filter, aggregate and otherwise prepare the data for analysis. Usually, these jobs involve reading source files from scalable storage (like HDFS, Azure Data Lake Store, and Azure Storage), processing them, and writing the output to new files in scalable storage.

The fundamental requirement of such batch processing engines is to scale out computations to handle a large volume of data. Unlike real-time processing, batch processing is expected to have latencies (the time between data ingestion and computing a result) that measure in minutes to hours.

Technology choices for batch processing

Azure Synapse Analytics

[Azure Synapse](#) is a distributed system designed to perform analytics on large data. It supports massive parallel processing (MPP), which makes it suitable for running high-performance analytics. Consider Azure Synapse when you have large amounts of data (more than 1 TB) and are running an analytics workload that will benefit from parallelism.

Azure Data Lake Analytics

[Data Lake Analytics](#) is an on-demand analytics job service. It's optimized for distributed processing of large data sets stored in Azure Data Lake Store.

- Languages: [U-SQL](#) (including Python, R, and C# extensions).
- Integrates with Azure Data Lake Store, Azure Storage blobs, Azure SQL Database, and Azure Synapse.
- Pricing model is per-job.

HDInsight

HDInsight is a managed Hadoop service. Use it to deploy and manage Hadoop clusters in Azure. For batch processing, you can use [Spark](#), [Hive](#), [Hive LLAP](#), [MapReduce](#).

- Languages: R, Python, Java, Scala, SQL
- Kerberos authentication with Active Directory, Apache Ranger-based access control
- Gives you complete control of the Hadoop cluster

Azure Databricks

[Azure Databricks](#) is an Apache Spark-based analytics platform. You can think of it as "Spark as a service." It's the easiest way to use Spark on the Azure platform.

- Languages: R, Python, Java, Scala, Spark SQL
- Fast cluster start times, autotermination, autoscaling.
- Manages the Spark cluster for you.
- Built-in integration with Azure Blob Storage, Azure Data Lake Storage (ADLS), Azure Synapse, and other services. See [Data Sources](#).
- User authentication with Microsoft Entra ID.
- Web-based [notebooks](#) for collaboration and data exploration.
- Supports [GPU-enabled clusters](#)

Key selection criteria

To narrow the choices, start by answering these questions:

- Do you want a managed service rather than managing your own servers?
- Do you want to author batch processing logic declaratively or imperatively?
- Will you perform batch processing in bursts? If yes, consider options that let you auto-terminate the cluster or whose pricing model is per batch job.
- Do you need to query relational data stores along with your batch processing, for example, to look up reference data? If yes, consider the options that enable the querying of external relational stores.

Capability matrix

The following tables summarize the key differences in capabilities.

General capabilities

Capability	Azure Data Lake Analytics	Azure Synapse	HDInsight	Azure Databricks
Is managed service	Yes	Yes	Yes ¹	Yes
Relational data store	Yes	Yes	No	No
Pricing model	Per batch job	By cluster hour	By cluster hour	Databricks Unit ² + cluster hour

[1] With manual configuration.

[2] A Databricks Unit (DBU) is a unit of processing capability per hour.

Capabilities

Capability	Azure Data Lake Analytics	Azure Synapse	HDInsight with Spark	HDInsight with Hive	HDInsight with Hive LLAP	Azure Databricks
Autoscaling	No	No	Yes	Yes	Yes	Yes
Scale-out granularity	Per job	Per cluster	Per cluster	Per cluster	Per cluster	Per cluster
In-memory caching of data	No	Yes	Yes	No	Yes	Yes
Query from external relational stores	Yes	No	Yes	No	No	Yes
Authentication	Microsoft Entra ID	SQL / Microsoft Entra ID	No	Microsoft Entra ID ¹	Microsoft Entra ID ¹	Microsoft Entra ID
Auditing	Yes	Yes	No	Yes ¹	Yes ¹	Yes
Row-level security	No	Yes ²	No	Yes ¹	Yes ¹	No
Supports firewalls	Yes	Yes	Yes	Yes ³	Yes ³	No
Dynamic data masking	No	Yes	No	Yes ¹	Yes ¹	No

[1] Requires using a domain-joined HDInsight cluster.

[2] Filter predicates only. See [Row-Level Security](#)

[3] Supported when [used within an Azure Virtual Network](#).

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Zoiner Tejada](#) | CEO and Architect

Next steps

- [Create a lake database in Azure Synapse Analytics](#)
- [Create an Azure Databricks workspace](#)
- [Explore Azure Databricks](#)
- [Get started with Azure Data Lake Analytics using the Azure portal](#)
- [Introduction to Azure Synapse Analytics](#)
- [What is Azure Databricks?](#)
- [What is Azure Synapse Analytics?](#)

Related resources

- [Analytics architecture design](#)
- [Choose an analytical data store in Azure](#)
- [Choose a data analytics technology in Azure](#)
- [Analytics end-to-end with Azure Synapse](#)
- [Batch processing](#)

Extract, transform, and load (ETL)

Azure Synapse Analytics

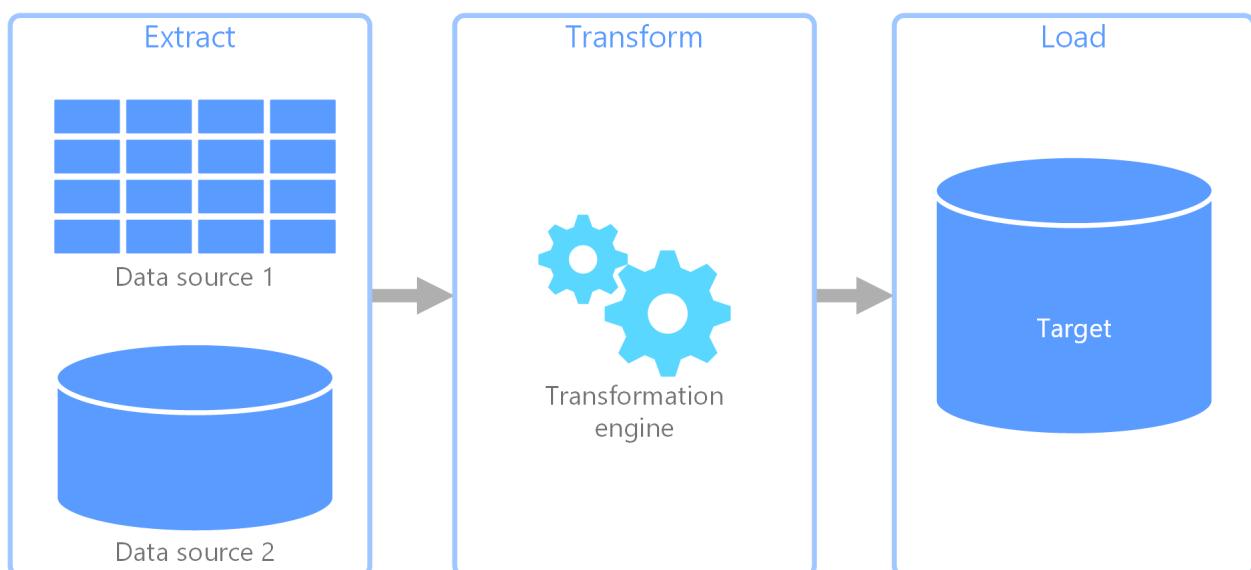
Azure Data Factory

A common problem that organizations face is how to gather data from multiple sources, in multiple formats. Then you'd need to move it to one or more data stores. The destination might not be the same type of data store as the source. Often the format is different, or the data needs to be shaped or cleaned before loading it into its final destination.

Various tools, services, and processes have been developed over the years to help address these challenges. No matter the process used, there's a common need to coordinate the work and apply some level of data transformation within the data pipeline. The following sections highlight the common methods used to perform these tasks.

Extract, transform, and load (ETL) process

Extract, transform, and load (ETL) is a data pipeline used to collect data from various sources. It then transforms the data according to business rules, and it loads the data into a destination data store. The transformation work in ETL takes place in a specialized engine, and it often involves using staging tables to temporarily hold data as it is being transformed and ultimately loaded to its destination.



The data transformation that takes place usually involves various operations, such as filtering, sorting, aggregating, joining data, cleaning data, deduplicating, and validating data.

Often, the three ETL phases are run in parallel to save time. For example, while data is being extracted, a transformation process could be working on data already received and prepare it for loading, and a loading process can begin working on the prepared data, rather than waiting for the entire extraction process to complete.

Relevant Azure service:

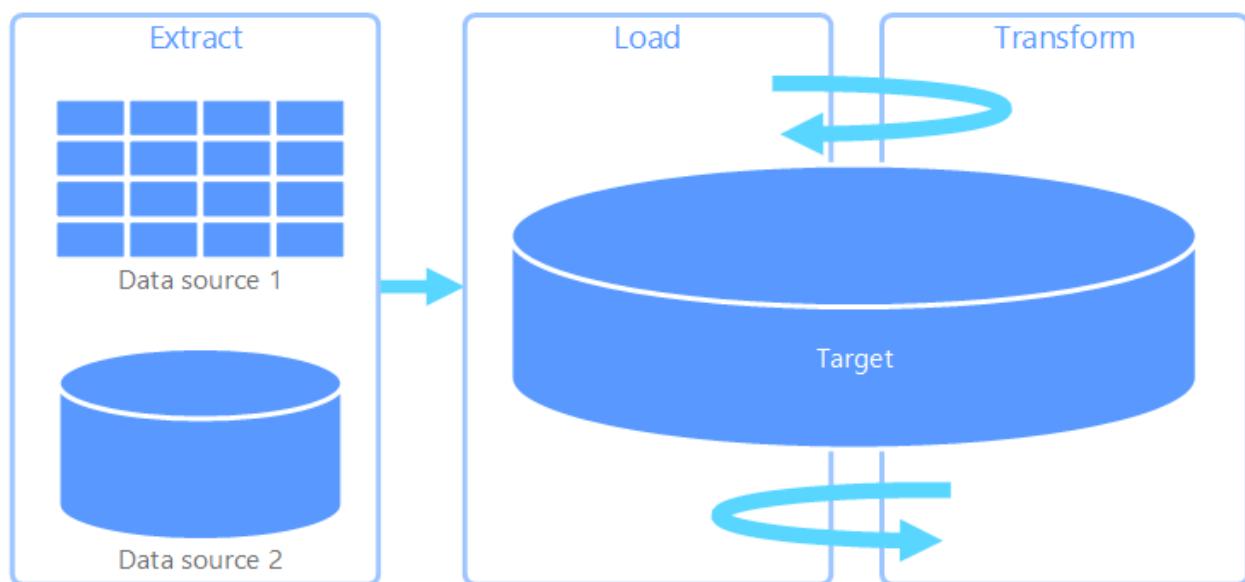
- [Azure Data Factory & Azure Synapse Pipelines](#)

Other tools:

- [SQL Server Integration Services \(SSIS\)](#)

Extract, load, and transform (ELT)

Extract, load, and transform (ELT) differs from ETL solely in where the transformation takes place. In the ELT pipeline, the transformation occurs in the target data store. Instead of using a separate transformation engine, the processing capabilities of the target data store are used to transform data. This simplifies the architecture by removing the transformation engine from the pipeline. Another benefit to this approach is that scaling the target data store also scales the ELT pipeline performance. However, ELT only works well when the target system is powerful enough to transform the data efficiently.



Typical use cases for ELT fall within the big data realm. For example, you might start by extracting all of the source data to flat files in scalable storage, such as a Hadoop distributed file system, an Azure blob store, or Azure Data Lake gen 2 (or a combination). Technologies, such as Spark, Hive, or Polybase, can then be used to query the source data. The key point with ELT is that the data store used to perform the transformation is the same data store where the data is ultimately consumed. This data

store reads directly from the scalable storage, instead of loading the data into its own proprietary storage. This approach skips the data copy step present in ETL, which often can be a time consuming operation for large data sets.

In practice, the target data store is a [data warehouse](#) using either a Hadoop cluster (using Hive or Spark) or a SQL dedicated pools on Azure Synapse Analytics. In general, a schema is overlaid on the flat file data at query time and stored as a table, enabling the data to be queried like any other table in the data store. These are referred to as external tables because the data does not reside in storage managed by the data store itself, but on some external scalable storage such as Azure data lake store or Azure blob storage.

The data store only manages the schema of the data and applies the schema on read. For example, a Hadoop cluster using Hive would describe a Hive table where the data source is effectively a path to a set of files in HDFS. In Azure Synapse, PolyBase can achieve the same result — creating a table against data stored externally to the database itself. Once the source data is loaded, the data present in the external tables can be processed using the capabilities of the data store. In big data scenarios, this means the data store must be capable of massively parallel processing (MPP), which breaks the data into smaller chunks and distributes processing of the chunks across multiple nodes in parallel.

The final phase of the ELT pipeline is typically to transform the source data into a final format that is more efficient for the types of queries that need to be supported. For example, the data may be partitioned. Also, ELT might use optimized storage formats like Parquet, which stores row-oriented data in a columnar fashion and provides optimized indexing.

Relevant Azure service:

- [SQL dedicated pools on Azure Synapse Analytics](#)
- [SQL Serverless pools on Azure Synapse Analytics](#)
- [HDInsight with Hive](#)
- [Azure Data Factory](#) ↗
- [Datamarts in Power BI](#)

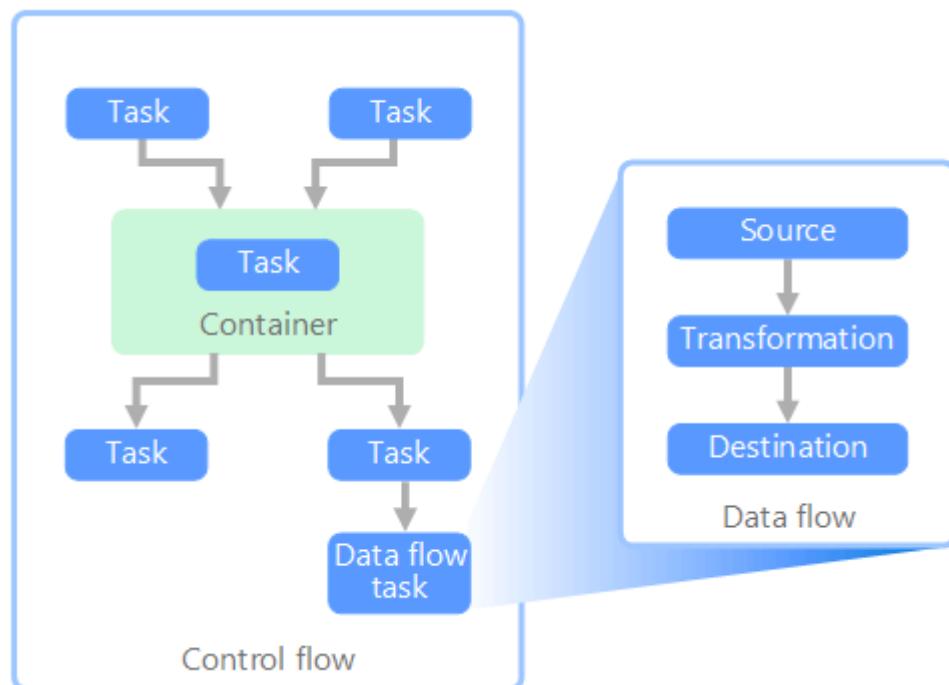
Other tools:

- [SQL Server Integration Services \(SSIS\)](#)

Data flow and control flow

In the context of data pipelines, the control flow ensures the orderly processing of a set of tasks. To enforce the correct processing order of these tasks, precedence constraints are used. You can think of these constraints as connectors in a workflow diagram, as shown in the image below. Each task has an outcome, such as success, failure, or completion. Any subsequent task does not initiate processing until its predecessor has completed with one of these outcomes.

Control flows execute data flows as a task. In a data flow task, data is extracted from a source, transformed, or loaded into a data store. The output of one data flow task can be the input to the next data flow task, and data flows can run in parallel. Unlike control flows, you cannot add constraints between tasks in a data flow. You can, however, add a data viewer to observe the data as it is processed by each task.



In the diagram above, there are several tasks within the control flow, one of which is a data flow task. One of the tasks is nested within a container. Containers can be used to provide structure to tasks, providing a unit of work. One such example is for repeating elements within a collection, such as files in a folder or database statements.

Relevant Azure service:

- [Azure Data Factory](#)

Other tools:

- [SQL Server Integration Services \(SSIS\)](#)

Technology choices

- Online Transaction Processing (OLTP) data stores
- Online Analytical Processing (OLAP) data stores
- Data warehouses
- Pipeline orchestration

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Raunak Jhawar](#) | Senior Cloud Architect
- [Zoiner Tejada](#) | CEO and Architect

Next steps

- Integrate data with Azure Data Factory or Azure Synapse Pipeline
- Introduction to Azure Synapse Analytics
- Orchestrate data movement and transformation in Azure Data Factory or Azure Synapse Pipeline

Related resources

The following reference architectures show end-to-end ELT pipelines on Azure:

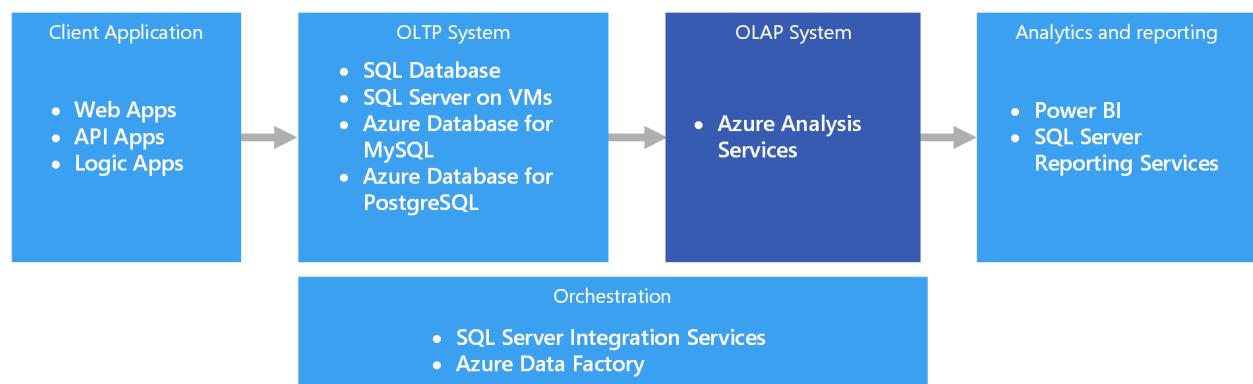
- [Enterprise BI in Azure with Azure Synapse](#)
- [Automated enterprise BI with Azure Synapse and Azure Data Factory](#)

Online analytical processing (OLAP)

Azure Analysis Services

Online analytical processing (OLAP) is a technology that organizes large business databases and supports complex analysis. It can be used to perform complex analytical queries without negatively affecting transactional systems.

The databases that a business uses to store all its transactions and records are called [online transaction processing \(OLTP\)](#) databases. These databases usually have records that are entered one at a time. Often they contain a great deal of information that is valuable to the organization. The databases that are used for OLTP, however, were not designed for analysis. Therefore, retrieving answers from these databases is costly in terms of time and effort. OLAP systems were designed to help extract this business intelligence information from the data in a highly performant way. This is because OLAP databases are optimized for heavy read, low write workloads.



Semantic modeling

A semantic data model is a conceptual model that describes the meaning of the data elements it contains. Organizations often have their own terms for things, sometimes with synonyms, or even different meanings for the same term. For example, an inventory database might track a piece of equipment with an asset ID and a serial number, but a sales database might refer to the serial number as the asset ID. There is no simple way to relate these values without a model that describes the relationship.

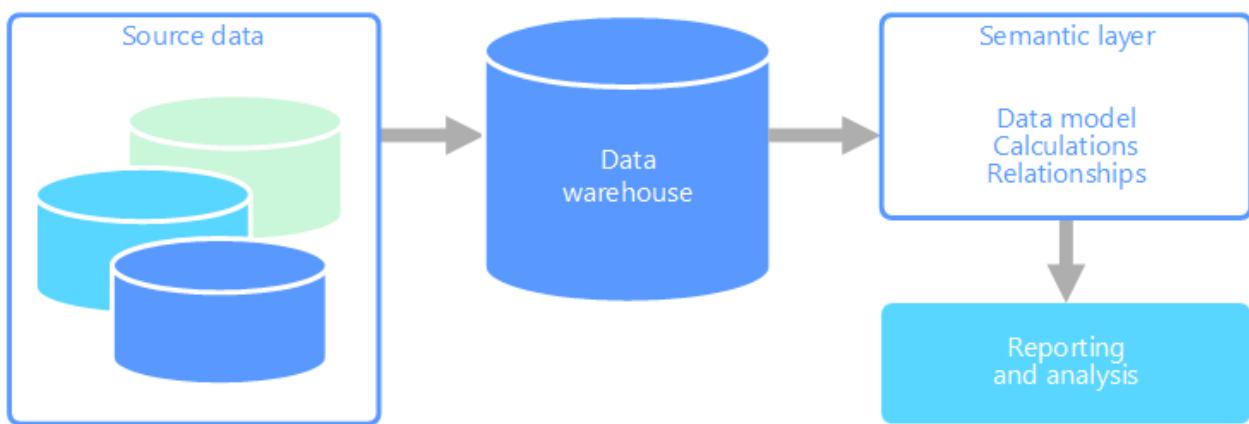
Semantic modeling provides a level of abstraction over the database schema, so that users don't need to know the underlying data structures. This makes it easier for end users to query data without performing aggregates and joins over the underlying

schema. Also, usually columns are renamed to more user-friendly names, so that the context and meaning of the data are more obvious.

Semantic modeling is predominately used for read-heavy scenarios, such as analytics and business intelligence (OLAP), as opposed to more write-heavy transactional data processing (OLTP). This is mostly due to the nature of a typical semantic layer:

- Aggregation behaviors are set so that reporting tools display them properly.
- Business logic and calculations are defined.
- Time-oriented calculations are included.
- Data is often integrated from multiple sources.

Traditionally, the semantic layer is placed over a data warehouse for these reasons.



There are two primary types of semantic models:

- **Tabular.** Uses relational modeling constructs (model, tables, columns). Internally, metadata is inherited from OLAP modeling constructs (cubes, dimensions, measures). Code and script use OLAP metadata.
- **Multidimensional.** Uses traditional OLAP modeling constructs (cubes, dimensions, measures).

Relevant Azure service:

- [Azure Analysis Services](#)

Example use case

An organization has data stored in a large database. It wants to make this data available to business users and customers to create their own reports and do some analysis. One option is just to give those users direct access to the database. However, there are several drawbacks to doing this, including managing security and controlling access. Also, the design of the database, including the names of tables and columns, may be hard for a user to understand. Users would need to know which tables to query, how

those tables should be joined, and other business logic that must be applied to get the correct results. Users would also need to know a query language like SQL even to get started. Typically this leads to multiple users reporting the same metrics but with different results.

Another option is to encapsulate all of the information that users need into a semantic model. The semantic model can be more easily queried by users with a reporting tool of their choice. The data provided by the semantic model is pulled from a data warehouse, ensuring that all users see a single version of the truth. The semantic model also provides friendly table and column names, relationships between tables, descriptions, calculations, and row-level security.

Typical traits of semantic modeling

Semantic modeling and analytical processing tends to have the following traits:

 [Expand table](#)

Requirement	Description
Schema	Schema on write, strongly enforced
Uses Transactions	No
Locking Strategy	None
Updateable	No (typically requires recomputing cube)
Appendable	No (typically requires recomputing cube)
Workload	Heavy reads, read-only
Indexing	Multidimensional indexing
Datum size	Small to medium sized
Model	Multidimensional

Requirement	Description
Data shape:	Cube or star/snowflake schema
Query flexibility	Highly flexible
Scale:	Large (10s-100s GBs)

When to use this solution

Consider OLAP in the following scenarios:

- You need to execute complex analytical and ad hoc queries rapidly, without negatively affecting your OLTP systems.
- You want to provide business users with a simple way to generate reports from your data
- You want to provide a number of aggregations that will allow users to get fast, consistent results.

OLAP is especially useful for applying aggregate calculations over large amounts of data. OLAP systems are optimized for read-heavy scenarios, such as analytics and business intelligence. OLAP allows users to segment multi-dimensional data into slices that can be viewed in two dimensions (such as a pivot table) or filter the data by specific values. This process is sometimes called "slicing and dicing" the data, and can be done regardless of whether the data is partitioned across several data sources. This helps users to find trends, spot patterns, and explore the data without having to know the details of traditional data analysis.

Semantic models can help business users abstract relationship complexities and make it easier to analyze data quickly.

Challenges

For all the benefits OLAP systems provide, they do produce a few challenges:

- Whereas data in OLTP systems is constantly updated through transactions flowing in from various sources, OLAP data stores are typically refreshed at a much slower intervals, depending on business needs. This means OLAP systems are better suited for strategic business decisions, rather than immediate responses to

changes. Also, some level of data cleansing and orchestration needs to be planned to keep the OLAP data stores up-to-date.

- Unlike traditional, normalized, relational tables found in OLTP systems, OLAP data models tend to be multidimensional. This makes it difficult or impossible to directly map to entity-relationship or object-oriented models, where each attribute is mapped to one column. Instead, OLAP systems typically use a star or snowflake schema in place of traditional normalization.

OLAP in Azure

In Azure, data held in OLTP systems such as Azure SQL Database is copied into the OLAP system, such as [Azure Analysis Services](#). Data exploration and visualization tools like [Power BI](#), Excel, and third-party options connect to Analysis Services servers and provide users with highly interactive and visually rich insights into the modeled data. The flow of data from OLTP data to OLAP is typically orchestrated using SQL Server Integration Services, which can be executed using [Azure Data Factory](#).

In Azure, all of the following data stores will meet the core requirements for OLAP:

- [SQL Server with Columnstore indexes](#)
- [Azure Analysis Services](#)
- [SQL Server Analysis Services \(SSAS\)](#)

SQL Server Analysis Services (SSAS) offers OLAP and data mining functionality for business intelligence applications. You can either install SSAS on local servers, or host within a virtual machine in Azure. Azure Analysis Services is a fully managed service that provides the same major features as SSAS. Azure Analysis Services supports connecting to [various data sources](#) in the cloud and on-premises in your organization.

Clustered Columnstore indexes are available in SQL Server 2014 and above, as well as Azure SQL Database, and are ideal for OLAP workloads. However, beginning with SQL Server 2016 (including Azure SQL Database), you can take advantage of hybrid transactional/analytic processing (HTAP) through the use of updateable nonclustered columnstore indexes. HTAP enables you to perform OLTP and OLAP processing on the same platform, which removes the need to store multiple copies of your data, and eliminates the need for distinct OLTP and OLAP systems. For more information, see [Get started with Columnstore for real-time operational analytics](#).

Key selection criteria

To narrow the choices, start by answering these questions:

- Do you want a managed service rather than managing your own servers?
- Do you require secure authentication using Microsoft Entra ID?
- Do you want to conduct real-time analytics? If so, narrow your options to those that support real-time analytics.

Real-time analytics in this context applies to a single data source, such as an enterprise resource planning (ERP) application, that will run both an operational and an analytics workload. If you need to integrate data from multiple sources, or require extreme analytics performance by using pre-aggregated data such as cubes, you might still require a separate data warehouse.

- Do you need to use pre-aggregated data, for example to provide semantic models that make analytics more business user friendly? If yes, choose an option that supports multidimensional cubes or tabular semantic models.

Providing aggregates can help users consistently calculate data aggregates. Pre-aggregated data can also provide a large performance boost when dealing with several columns across many rows. Data can be pre-aggregated in multidimensional cubes or tabular semantic models.

- Do you need to integrate data from several sources, beyond your OLTP data store? If so, consider options that easily integrate multiple data sources.

Capability matrix

The following tables summarize the key differences in capabilities.

General capabilities

[] Expand table

Capability	Azure Analysis Services	SQL Server Analysis Services	SQL Server with Columnstore Indexes	Azure SQL Database with Columnstore Indexes
Is managed service	Yes	No	No	Yes

Capability	Azure Analysis Services	SQL Server Analysis Services	SQL Server with Columnstore Indexes	Azure SQL Database with Columnstore Indexes
Supports multidimensional cubes	No	Yes	No	No
Supports tabular semantic models	Yes	Yes	No	No
Easily integrate multiple data sources	Yes	Yes	No ¹	No ¹
Supports real-time analytics	No	No	Yes	Yes
Requires process to copy data from source(s)	Yes	Yes	No	No
Microsoft Entra integration	Yes	No	No ²	Yes

[1] Although SQL Server and Azure SQL Database cannot be used to query from and integrate multiple external data sources, you can still build a pipeline that does this for you using [SSIS](#) or [Azure Data Factory](#). SQL Server hosted in an Azure VM has additional options, such as linked servers and [PolyBase](#). For more information, see [Pipeline orchestration, control flow, and data movement](#).

[2] Connecting to SQL Server running on an Azure Virtual Machine is not supported using a Microsoft Entra account. Use a domain Active Directory account instead.

Scalability Capabilities

Expand table

Capability	Azure Analysis Services	SQL Server Analysis Services	SQL Server with Columnstore Indexes	Azure SQL Database with Columnstore Indexes
Redundant regional servers for high availability	Yes	No	Yes	Yes
Supports query scale out	Yes	No	Yes	Yes
Dynamic scalability (scale up)	Yes	No	Yes	Yes

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Zoiner Tejada](#) | CEO and Architect

Next steps

- [Columnstore indexes: Overview](#)
- [Create an Analysis Services server](#)
- [What is Azure Data Factory?](#)
- [What is Power BI?](#)

Related resources

- [Big data architecture style](#)
- [Online analytical processing \(OLAP\)](#)

Choose a stream processing technology in Azure

Article • 02/02/2023

This article compares technology choices for real-time stream processing in Azure.

Real-time stream processing consumes messages from either queue or file-based storage, processes the messages, and forwards the result to another message queue, file store, or database. Processing may include querying, filtering, and aggregating messages. Stream processing engines must be able to consume endless streams of data and produce results with minimal latency. For more information, see [Real time processing](#).

What are your options when choosing a technology for real-time processing?

In Azure, all of the following data stores will meet the core requirements supporting real-time processing:

- [Azure Stream Analytics](#)
- [HDInsight with Spark Streaming](#)
- [Apache Spark in Azure Databricks](#)
- [HDInsight with Storm](#)
- [Azure Functions](#)
- [Azure App Service WebJobs](#)
- [Apache Kafka streams API](#)

Key Selection Criteria

For real-time processing scenarios, begin choosing the appropriate service for your needs by answering these questions:

- Do you prefer a declarative or imperative approach to authoring stream processing logic?
- Do you need built-in support for temporal processing or windowing?
- Does your data arrive in formats besides Avro, JSON, or CSV? If yes, consider options that support any format using custom code.

- Do you need to scale your processing beyond 1 GB/s? If yes, consider the options that scale with the cluster size.

Capability matrix

The following tables summarize the key differences in capabilities.

General capabilities

Capability	Azure Stream Analytics	HDInsight with Spark Streaming	Apache Spark in Azure Streaming	HDInsight with Storm	Azure Functions	Azure App Service WebJobs
Programmability	SQL, JavaScript	C#/F# ↗ , Java, Python, Scala	C#/F# ↗ , Java, Python, R, Scala	C#, Java	C#, F#, Java, Node.js, Python	C#, Java, Node.js, PHP, Python
Programming paradigm	Declarative	Mixture of declarative and imperative	Mixture of declarative and imperative	Imperative	Imperative	Imperative
Pricing model	Streaming units ↗	Per cluster hour	Databricks units ↗	Per cluster hour	Per function execution and resource consumption	Per app service plan hour

Integration capabilities

Capability	Azure Stream Analytics	HDInsight with Spark Streaming	Apache Spark in Azure Streaming	HDInsight with Storm	Azure Functions	Azure App Service WebJobs

Capability	Azure Stream Analytics	HDInsight with Spark	Apache Spark in Azure Streaming	HDInsight with Storm	Azure Functions	Azure App Service WebJobs
Inputs	Azure Event Hubs, Azure IoT Hub, Azure Blob storage/ADLS Gen2	Event Hubs, IoT Hub, Kafka, HDFS, Storage Blobs, Azure Data Lake Store	Event Hubs, IoT Hub, Kafka, HDFS, Storage Blobs, Azure Data Lake Store	Event Hubs, IoT Hub, Kafka, HDFS, Storage Blobs, Azure Data Lake Store	Supported bindings	Service Bus, Storage Queues, Storage Blobs, Event Hubs, WebHooks, Azure Cosmos DB, Files
Sinks	Azure Data Lake Storage Gen 1, Azure Data Explorer, Azure Database for PostgreSQL, Azure SQL Database, Azure Synapse Analytics, Blob storage and Azure Data Lake Gen 2, Azure Event Hubs, Power BI, Azure Table storage, Azure Service Bus queues, Azure Service Bus topics, Azure Cosmos DB, Azure Functions	HDFS, Kafka, Storage Blobs, Azure Data Lake Store, Store, Azure Cosmos DB	HDFS, Kafka, Storage Blobs, Azure Data Lake Store, Azure	Event Hubs, Service Bus, Kafka	Supported bindings	Service Bus, Storage Queues, Storage Blobs, Event Hubs, WebHooks, Azure Cosmos DB, Files

Processing capabilities

Capability	Azure Stream Analytics	HDInsight with Spark Streaming	Apache Spark in Azure Databricks	HDInsight with Storm	Azure Functions	Azure App Service WebJobs
Built-in temporal/windowing support	Yes	Yes	Yes	Yes	No	No
Input data formats	Avro, JSON or CSV, UTF-8 encoded	Any format using custom code	Any format using custom code	Any format using custom code	Any format using custom code	Any format using custom code
Scalability	Query partitions	Bounded by cluster size	Bounded by Databricks cluster scale configuration	Bounded by cluster size	Up to 200 function app instances processing in parallel	Bounded by app service plan capacity
Late arrival and out of order event handling support	Yes	Yes	Yes	Yes	No	No

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Zoiner Tejada](#) | CEO and Architect

Next steps

- [App Service overview](#)
- [Explore Azure Functions](#)
- [Get started with Azure Stream Analytics](#)
- [Perform advanced streaming data transformations](#)
- [Set up clusters in HDInsight](#)
- [Use Apache Spark in Azure Databricks](#)

Related resources

- Choose a real-time message ingestion technology
- Real time processing
- Stream processing with Azure Stream Analytics

DR for Azure Data Platform - Overview

Azure Synapse Analytics

Azure Machine Learning

Azure Cosmos DB

Azure Data Lake

Azure Event Hubs

Overview

This series provides an illustrative example of how an organization could design a disaster recovery (DR) strategy for an Azure enterprise Data platform.

- This series of articles complements the guidance provided by Microsoft's [Cloud Adoption Framework](#), Azure's [Well-Architected Framework](#) and [Business Continuity Management](#)

Azure provides a broad range of resiliency options that can provide service continuity in the event of a disaster. But higher service levels can introduce complexity and a cost premium. The trade-off of cost versus resiliency versus complexity is the key decision-making factor for most customers regarding DR.

While occasional point failures do happen across the Azure service, it should be noted that Microsoft Data Centers, and Azure Services have multiple layers of redundancy built-in. Any failure is normally limited in scope and is typically recovered within a matter of hours. Historically it's far more likely that a key service such as identity management experiences a service issue rather than an entire Azure region going offline.

It should also be acknowledged that cyber-attacks, particularly ransomware, now pose a tangible threat to any modern data ecosystem and can result in a data platform outage. While this is out-of-scope for this series, customers are advised to implement controls against such attacks as part of any data platform's security and resiliency design.

- Microsoft guidance on ransomware protection is available in Azure's [Cloud Fundamentals](#)

Scope

The scope of this article series includes:

- The service recovery of an Azure data platform from a physical disaster for an illustrative persona of the customer. This illustrative customer is:

- a mid-large organization with a defined operational support function, following an ITIL based service management methodology
- not cloud-native, with its core enterprise, shared services like access and authentication management and incident management remain on premises
- on the journey of cloud migration to Azure, enabled by automation
- The Azure data platform has implemented the following designs within the customer's Azure tenancy
 - [Enterprise Landing Zone](#) – Providing the platform foundation, including networking, monitoring, security etc.
 - [Azure Analytics Platform](#) ↗ - Providing the data components that support the various solutions and data products provided by the service
- This process will be executed by an Azure technical resource rather than a specialist Azure SME. As such, the resource(s) should have the following level of knowledge/skills
 - [Azure Fundamentals](#) – working knowledge of Azure, its core services, and data components
 - Working knowledge of Azure DevOps. Able to navigate source control and execute pipeline deployments
- This process describes the Failover process, from the primary to the secondary region

Out of scope

The following items are considered out-of-scope for this article series:

- The Fallback process, from the secondary region back to the primary region
- Any non-Azure applications, components, or systems – this includes but isn't limited to on-premises, other cloud vendors, third party web services etc.
- Recovery of any upstream services, such as on-premises networks, gateways, enterprise shared services, etc., that are prerequisites to this process
- Recovery of any downstream services, such as on-premises operational systems, third party reporting systems, data modeling or data science applications, etc., that are dependent on this process to recover their own services
- Data Loss scenarios, including recovery from [ransomware or similar data security incidents](#)
- Data Backup strategies and data restore plans
- Establishing the Root cause of a DR event
 - For Azure service/component incidents, Microsoft publishes a "Root Cause Analysis" within the [Status – History webpage](#) ↗

Key assumptions

The key assumptions for this DR worked example are

- The Organization follows an ITIL based service management methodology for operational support of the Azure data platform
- The Organization has an existing disaster recovery process as part of its service restoration framework for IT assets
- “[Infrastructure as Code](#)” (IaC) has been used to deploy the Azure data platform enabled by an automation service, such as Azure DevOps or similar
- Each solution hosted by the Azure data platform has completed a Business Impact Assessment or similar, providing clear service requirements for RPO, RTO and MTO

Next steps

Now that you have learned about the scenario at a high level, you can move on to learn about the [architecture](#) designed for the use case.

Related resources

- [DR for Azure Data Platform - Architecture](#)
- [DR for Azure Data Platform - Scenario details](#)
- [DR for Azure Data Platform - Recommendations](#)
- [DR for Azure Data Platform - Deploy this scenario](#)

DR for Azure Data Platform - Architecture

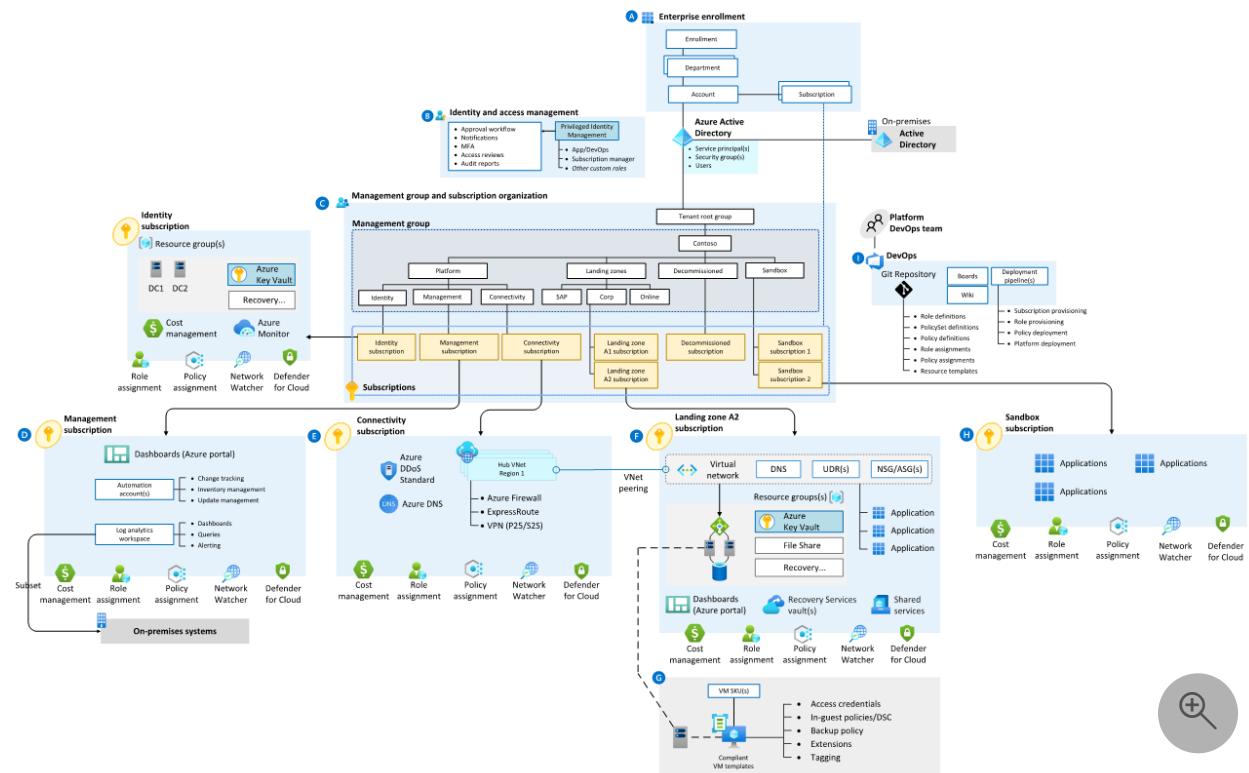
Azure Synapse Analytics Azure Machine Learning Azure Cosmos DB Azure Data Lake Azure Event Hubs

Use case definition

To support this worked example, the fictitious firm “Contoso” will be used with an Azure Data Platform based upon Microsoft Reference Architectures.

Data Service - Component View

Contoso has implemented the following foundational Azure structure, which is a subset of the [Enterprise Landing Zone](#).



The numbers in the following descriptions correspond to the preceding diagram above.

Contoso's Azure Foundations - Workflow

1. Enterprise Enrollment - Contoso's top parent enterprise enrollment within Azure reflecting its commercial agreement with Microsoft, its organizational account

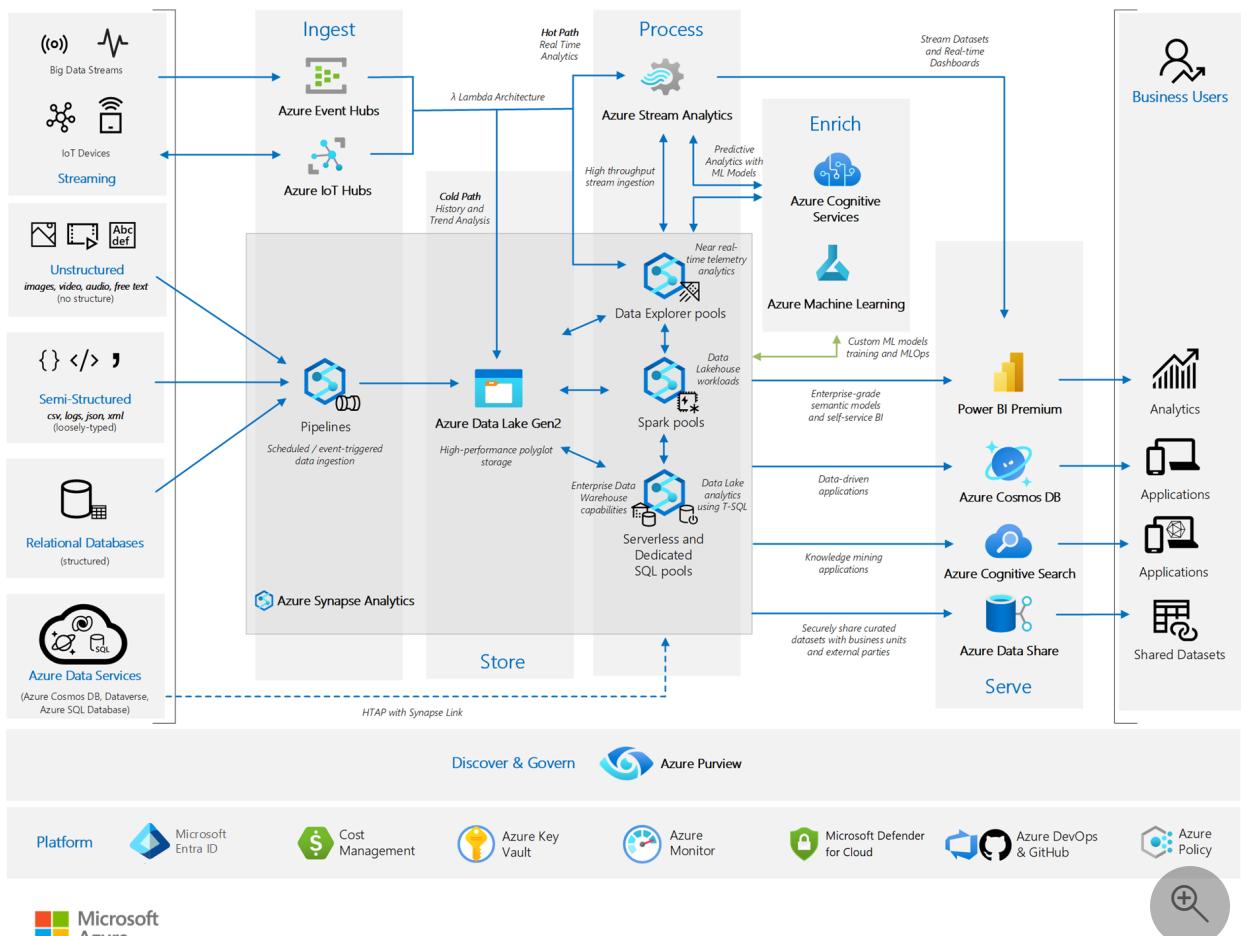
structure and available Azure subscriptions. It provides the billing foundation for subscriptions and how the digital estate is administered

2. **Identity and Access Management** – The components required to provide identity, authentication, resource access and authorization services across Contoso's Azure footprint
3. **Management Group and Subscription Organization** - A scalable group hierarchy aligned to the data platform's core capabilities, allowing operationalization at scale using centrally managed security and governance where workloads have clear separation. Management groups provide a governance scope above subscriptions
4. **Management Subscription** - A dedicated subscription for the various management level functions of required to support the data platform
5. **Connectivity Subscription** - A dedicated subscription for the connectivity functions of the data platform enabling it to identify named services, determine secure routing and communication across and between internal and external services
6. **Landing Zone Subscription** – One-to-many subscriptions for Azure native, online applications, internal and external facing workloads and resources
7. **DevOps Platform** - The DevOps Platform that supports the Azure foundation & Data Platform. This platform contains the code base source control repository and CI/CD pipelines enabling automated deployments of IaC

 **Note**

Many customers still retain a large IaaS footprint. To provide recovery capabilities across IaaS, the key component to be added is [Azure Site recovery](#). Site Recovery will orchestrate and automate the replication of Azure VMs between regions, on-premises virtual machines and physical servers to Azure, and on-premises machines to a secondary datacenter.

Within this foundational structure, Contoso has implemented the following elements to support its enterprise business intelligence needs, aligned to the guidance in [Analytics end-to-end with Azure Synapse](#).



Contoso's data platform

Contoso's Data Platform - Workflow

The workflow is read left to right, following the flow of data:

- **Data Sources** - The sources or types of data that the data platform can consume from
- **Ingest** - The Platform's capability to ingest data from various sources of varying structure and speed. This design reflects a [Lambda architecture](#)
- **Store** - The capability to securely store data at scale that has been ingested onto the platform
- **Process** - The Platform's capability to process data, making it "fit for purpose" for downstream processes like cleansing, standardizing and modeling. The pre-processing of data typically ensures that it's in a "position and a condition, ready for use"
- **Enrich** - The capability to enhance data processed on the platform via statistical, Machine Learning or other modeling techniques or prebuilt Azure AI Services
- **Serve** - The Platform's capability to shape and present data for downstream consumption
- **Data Consumers** - The individuals, applications or downstream processes that consume data from the platforms' various serving touchpoints

- **Discover and Govern** - The Platform's capabilities to govern the data it contains and ensure it's indexed, discoverable/searchable, well-described, with full lineage and is transparent to its end users and consuming processes.
- **Platform** - The foundation upon which the platform is built, that is, Contoso's Azure Foundations as described above.

ⓘ Note

For many customers, the conceptual level of the Data Platform reference architecture used will align, but the physical implementation may vary. For example, ELT (extract, load, transform) processes may be performed through **Azure Data Factory**, and data modeling by **Azure SQL server**. To address this concern, the Stateless vs Stateful section below will provide guidance.

For the Data Platform, Contoso has selected the lowest recommended production service tiers for all components and has chosen to adopt a “Redeploy on Disaster” DR strategy based upon an operating cost-minimization approach.

The following sections will provide a baseline understanding of the DR process and levers available to customers to uplift this posture.

Azure service and component view

The following tables present a breakdown of each Azure service and component used across the Contoso – Data platform, with options for DR uplift.

ⓘ Note

The sections below are organized by stateful vs stateless services

Stateful Foundational Components

- **Microsoft Entra ID including role entitlements**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: Premium P1
 - DR Uplift options: Microsoft Entra ID's resiliency is part of its SaaS offering
 - Notes
 - [Advancing service resilience in Microsoft Entra ID](#)

- **Azure Key Vault**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: N/A
 - DR Uplift options: N/A, Covered as part of the Azure Service
- **Recovery Services Vault**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: Default (GRS)
 - DR Uplift options: Enabling [Cross Region Restore](#) creates data restoration in the secondary, [paired region](#)
 - Notes
 - While LRS and ZRS are available, it requires configuration activities from the default setting
- **Azure DevOps**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: DevOps Services
 - DR Uplift options: DevOps [service and data resiliency](#) is part of its SaaS offering
 - Notes
 - DevOps Server as the on-premises offering will remain the customer's responsibility for disaster recovery
 - If third party services (SonarCloud, Jfrog Artifactory, Jenkins build servers for example) are used, they'll remain the customer's responsibility for recovery from a disaster
 - If IaaS VMs are used within the DevOps toolchain, they'll remain the customer's responsibility for recovery from a disaster

Stateless Foundational Components

- **Subscriptions**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: N/A
 - DR Uplift options: N/A, Covered as part of the Azure Service
- **Management Groups**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: N/A

- DR Uplift options: N/A, Covered as part of the Azure Service
- **Azure Monitor**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: N/A
 - DR Uplift options: N/A, Covered as part of the Azure Service
- **Cost Management**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: N/A
 - DR Uplift options: N/A, Covered as part of the Azure Service
- **Microsoft Defender for Cloud**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: N/A
 - DR Uplift options: N/A, Covered as part of the Azure Service
- **Azure DNS**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: Single Zone - Public
 - DR Uplift options: N/A, DNS is highly available by design
- **Network Watcher**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: N/A
 - DR Uplift options: N/A, Covered as part of the Azure Service
- **Virtual Networks, including Subnets, UDR & NSGs**
 - Component Recovery Responsibility: Contoso
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: N/A
 - DR Uplift options: [VNets can be replicated into the secondary, paired region](#)
- **Azure Firewall**
 - Component Recovery Responsibility: Contoso
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Standard

- DR Uplift options: Azure Firewall is [highly available by design](#) and can be created with [Availability Zones](#) for increased availability
- **Azure DDoS**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: DDoS Network Protection
 - DR Uplift options: N/A, covered as part of the Azure service
- **ExpressRoute Circuit**
 - Component Recovery Responsibility: Contoso, connectivity partner and Microsoft
 - Workload/Configuration Recovery Responsibility: Connectivity partner and Microsoft
 - Contoso SKU selection: Standard
 - DR Uplift options:
 - ExpressRoute can be uplifted to use [private peering](#), delivering a geo-redundant service
 - ExpressRoute also has [high availability designs](#) available
 - [Site-to-Site VPN connection](#) can be used as a backup for ExpressRoute
 - Notes
 - The ExpressRoute has [inbuilt redundancy](#), with each circuit consisting of two connections to two Microsoft Enterprise edge routers (MSEEs) at an ExpressRoute Location from the connectivity provider/client's network edge
 - [ExpressRoute premium](#) circuit will enable access to all Azure regions globally
- **VPN Gateway**
 - Component Recovery Responsibility: Contoso
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Single Zone - VpnGw1
 - DR Uplift options: A VPN Gateway can be deployed into an [Availability Zone](#) with the VpnGw#AZ SKUs to provide a [zone redundant service](#)
- **Azure Load Balancer**
 - Component Recovery Responsibility: Contoso
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Standard
 - DR Uplift options:
 - A Load Balancer can be configured for [Zone redundancy within a region with availability zones](#). If so, the data path will survive as long as one zone within the region remains healthy

- Depending on the primary region, a [cross-region load balancer](#) can be deployed for a highly available, cross regional deployment
- Notes
 - [Azure Traffic Manager](#) is a DNS-based traffic load balancer. This service supports the distribution of traffic for public-facing applications across the global Azure regions. This solution will provide protection from a regional outage within a high availability design

Stateful Data platform-specific services

- **Storage Account: Azure Data Lake Gen2**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: LRS
 - DR Uplift options: Storage Accounts have a broad range of [data redundancy](#) options from primary region redundancy up to secondary region redundancy
 - Notes
 - GRS is recommended to uplift redundancy, providing a copy of the data in the paired region
- **Azure Event Hubs**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Standard
 - DR Uplift options: An event hub namespace can be created with [availability zones](#) enabled. This resiliency can be extended to cover a full region outage with [Geo-disaster recovery](#)
 - Notes
 - By design, Event Hubs geo-disaster recovery doesn't replicate data, therefore there are several [considerations to keep in mind](#) for failover and fallback
- **Azure IoT Hubs**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Standard
 - DR Uplift options:
 - IoT Hub Resiliency can be uplifted by a [cross regional HA implementation](#)
 - Microsoft provides the following [guidance for HA/DR options](#)
 - Notes
 - IoT Hub provides Microsoft-Initiated Failover and Manual Failover by replicating data to the paired region for each IoT hub

- IoT Hub provides [Intra-Region HA](#) and will automatically use an availability zone if created in a [predefined set of Azure regions](#)
- **Azure Stream Analytics**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Standard
 - DR Uplift options: While Azure Stream Analytics is a fully managed PaaS offering, it doesn't provide automatic geo-failover. [Geo-redundancy](#) can be achieved by deploying identical Stream Analytics jobs in multiple Azure regions
- **Azure Machine Learning**
 - Component Recovery Responsibility: Contoso and Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: General Purpose, D Series instances
 - DR Uplift options:
 - Azure Machine Learning depends on multiple Azure services, some of which are [provisioned in the customer's subscription](#). As such, the customer remains responsible for the high-availability configuration of these services
 - Resiliency can be uplifted via a [multi-regional deployment](#)
 - Notes:
 - Azure Machine Learning itself doesn't [provide automatic failover or disaster recovery](#)
- **Power BI**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: Power BI Pro
 - DR Uplift options: N/A, Power BI's resiliency is part of its SaaS offering
 - Notes
 - Power BI resides in the Office365 tenancy, not that of Azure
 - [Power BI uses Azure Availability Zones](#) to protect Power BI reports, applications and data from data center failures
 - In the case of regional failure, Power BI will [failover to a new region](#), usually in the same geographical location, as noted in the [Microsoft Trust Center](#) ↗
- **Azure Cosmos DB**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: Single Region Write with Periodic backup
 - DR Uplift options:

- Single-region accounts may lose availability following a regional outage. Resiliency can be uplifted to a [single write region and at least a second \(read\) region and enable Service-Managed failover](#)
- It's [recommended](#) that Azure Cosmos accounts used for production workloads to enable automatic failover. In the absence of this configuration, the account will experience loss of write availability for all the duration of the write region outage, as manual failover won't succeed due to lack of region connectivity
- Notes
 - To protect against data loss in a region, Azure Cosmos DB provides two [different backup modes](#) - Periodic and Continuous
 - [Regional failovers](#) are detected and handled in the Azure Cosmos DB client. They don't require any changes from the application
 - The following guidance describes the [impact of a region outage based upon the Cosmos DB configuration](#)
- **Azure Data Share**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: N/A
 - DR Uplift options: Azure Data Share's resiliency can be uplifted by [HA deployment into a secondary region](#)
- **Microsoft Purview**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: N/A
 - DR Uplift options: N/A
 - Notes
 - As of Dec 2023, [Microsoft Purview doesn't support automated BCDR](#). Until that support is added, the customer is responsible for all backup and restore activities.

Stateless Data platform-specific services

- **Azure Synapse: Pipelines**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Computed Optimized Gen2
 - DR Uplift options: N/A, Synapse resiliency is part of its SaaS offering using the [automatic failover](#) feature

- Notes
 - If Self-Hosted Data Pipelines are used, they'll remain the customer's responsibility for recovery from a disaster
- **Azure Synapse: Data Explorer Pools**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Computed Optimized, Small (4 cores)
 - DR Uplift options: N/A, Synapse resiliency is part of its SaaS offering
 - Notes
 - Availability Zones are enabled by default for [Synapse Data Explorer](#) where available.
- **Azure Synapse: Spark Pools**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Computed Optimized, Small (4 cores)
 - DR Uplift options: N/A, Synapse resiliency is part of its SaaS offering
 - Notes
 - Currently, Azure Synapse Analytics only supports disaster recovery for [dedicated SQL pools](#) and [doesn't support it for Apache Spark pools](#) ↗
- **Azure Synapse: Serverless and Dedicated SQL Pools**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Contoso
 - Contoso SKU selection: Computed Optimized Gen2
 - DR Uplift options: N/A, Synapse resiliency is part of its SaaS offering
 - Notes
 - Azure Synapse Analytics [automatically takes snapshots](#) throughout the day to create restore points that are available for seven days
 - Azure Synapse Analytics performs a [standard geo-backup](#) once per day to a paired data center. The RPO for a geo-restore is 24 hours
 - If Self-Hosted Data Pipelines are used, they'll remain the customers responsibility recovery from a disaster
- **Azure AI services (formerly Cognitive Services)**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: Pay As You Go
 - DR Uplift options: N/A, the APIs for AI services are hosted by [Microsoft-managed data centers](#)
 - Notes

- If AI services has been deployed via customer deployed [Docker containers](#), recovery remains the responsibility of the customer
- **Azure AI Search (formerly Cognitive Search)**
 - Component Recovery Responsibility: Microsoft
 - Workload/Configuration Recovery Responsibility: Microsoft
 - Contoso SKU selection: Standard S1
 - DR Uplift options:
 - AI Search can be raised to an [HA design](#) by using replicas across [availability zones](#) and regions
 - [Multiple services in separate regions](#) can extend the resiliency further
 - Notes
 - In AI Search business continuity (and disaster recovery) is achieved through multiple AI Search services.
 - there's [no built-in mechanism for disaster recovery](#). If continuous service is required during a catastrophic failure, the recommendation is to have a second service in a different region, and implementing a geo-replication strategy to ensure indexes are fully redundant across all services

Stateful vs Stateless Components

The speed of innovation across the Microsoft product suite and Azure, in particular, means the component set that we've used for this worked example will quickly evolve. To future-proof against providing stale guidance and extend this guidance to components not explicitly covered in this document, the section below provides some instruction based upon the coarse-grain classification of state.

A component/service can be described as stateful if it's designed to remember preceding events or user interactions. Stateless means there's no record of previous interactions, and each interaction request has to be handled based entirely on information that comes with it.

For a DR scenario that calls for redeployment:

- Components/services that are "stateless", like Azure Functions and Azure Data Factory pipelines, can be redeployed from source control with at least a smoke test to validate availability before being introduced into the broader system
- Components/services that are "stateful", like Azure SQL database and storage accounts, require more attention
 - When procuring the component, a key decision will be selecting the data redundancy feature. This decision typically focuses on a trade-off between availability and durability with operating costs

- Datastores will also need a data backup strategy. The data redundancy functionality of the underlying storage mitigates this risk for some designs, while others, like SQL databases will need a separate backup process.
 - If necessary, the component can be redeployed from source control with a validated configuration via a smoke-test
 - A redeployed datastore must have its dataset rehydrated. Rehydration can be accomplished through data redundancy (when available) or a backup dataset. When rehydration has been completed, it must be validated for accuracy and completeness
 - Depending on the nature of the backup process, the backup datasets may require validation before being applied. Backup process corruption/error may result in an earlier backup being used in place of the latest version available
 - Any delta between the component date/timestamp and the current date should be addressed by re-executing or replaying the data ingestion processes from that point forward
 - Once the component's dataset is up to date, it can be introduced into the broader system

Other key services

This section contains HA/DR guidance for other key Azure Data components and services.

- Azure Databricks - DR guidance can be found in the [product documentation](#)
- Azure Analysis Services - HA guidance can be found in the [product documentation](#)
- Azure Database for MySQL
 - Flexible Server HA guidance can be found in the [product documentation](#)
 - Single Server HA guidance can be found in the [product documentation](#)
- SQL
 - SQL on Azure VMs guidance can be found in the [product documentation](#)
 - Azure SQL and Azure SQL Managed Instance guidance can be found in the [product documentation](#)

Next steps

Now that you've learned about the scenario's architecture, you can learn about the [scenario details](#)

Related resources

- DR for Azure Data Platform - Overview
- DR for Azure Data Platform - Scenario details
- DR for Azure Data Platform - Recommendations
- DR for Azure Data Platform - Deploy this scenario

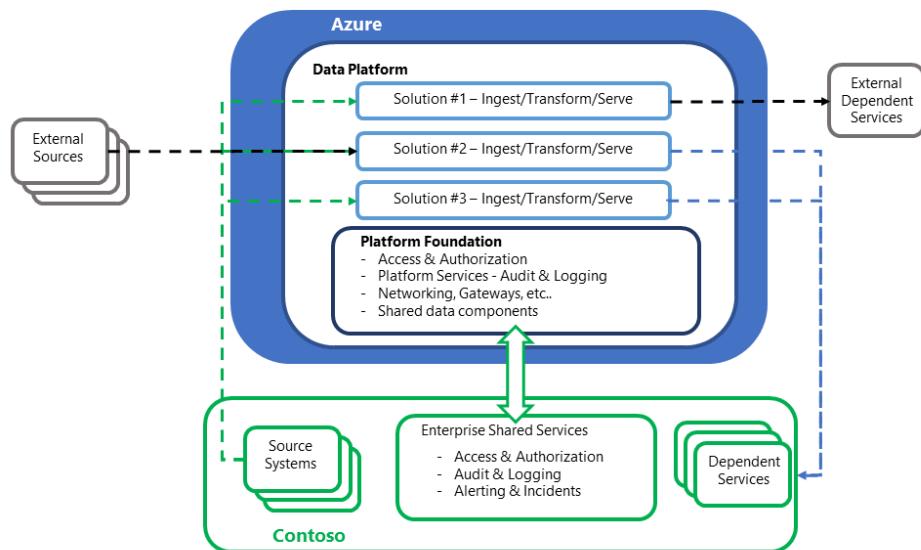
DR for Azure Data Platform - Scenario details

Azure Synapse Analytics Azure Machine Learning Azure Cosmos DB Azure Data Lake Azure Event Hubs

Data service topology

At a high-level the data service topology for Contoso's data platform can be illustrated as:

Simplified High-Level Topology



This logical diagram abstracts the key functions of the Contoso data ecosystem into a simplified, high-level view. This abstracted view supports the sections covering the scenario deployments, in line with the DR strategy selection and the segregation of responsibilities in a service recovery process.

DR Impact vs Customer Activity

The following sections present a breakdown of Contoso activity necessary across DR events of varying impacts.

Area: Foundational components

- Microsoft Entra ID including role entitlements
 - Contoso SKU selection: Premium P1

- DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Management Groups**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Subscriptions**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Azure Key Vault**
 - Contoso SKU selection: Standard
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Azure Monitor**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Microsoft Defender for Cloud**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Cost Management**
 - Contoso SKU selection: N/A
 - DR Impact

- Azure Data Center Failure: N/A
- Availability Zone Failure: N/A
- Azure Regional Failure: N/A
- **Azure DNS**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Network Watcher**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Recovery Services Vault**
 - Contoso SKU selection: Default (GRS)
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
 - Notes
 - [Cross Region Restore](#) will enable DR drills and the customer failing over to the secondary region
- **Virtual Networks, including Subnets, UDR & NSGs**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to redeploy the Foundation and Data platform VNets with their attached UDRs & NSGs into the secondary region
 - Notes
 - [Traffic Manager](#) can be used to geo-route traffic between regions that hold replica VNet structures. If they have the same address space, they can't be connected to the on-premises network, as it would cause routing issues. At the time of a disaster and loss of a VNet in one region, you can connect the

other VNet in the available region, with the matching address space to your on-premises network

- **Resource Groups**

- Contoso SKU selection: N/A
- DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to redeploy the Foundation and Data platform Resource groups into the secondary region
- Notes
 - This activity would be mitigated by implementing the "Warm Spare" strategy, having the network and resource group topology available in the secondary region

- **Azure Firewall**

- Contoso SKU selection: Standard
- DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: Contoso would need to validate availability and redeploy if necessary
 - Azure Regional Failure: Contoso would need to redeploy the Foundation Azure Firewalls into the secondary region
- Notes
 - Azure Firewall can be created with [Availability Zones](#) for increased availability
 - A "Warm Spare" strategy would mitigate this activity

- **Azure DDoS**

- Contoso SKU selection: Network Protection
- DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to create a [DDoS protection plan](#) for the Foundation's VNets within the secondary region

- **ExpressRoute – Circuit**

- Contoso SKU selection: Standard
- DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- Notes

- The physical circuit would remain the responsibility of Microsoft and the connectivity partner to recover
- **VPN Gateway**
 - Contoso SKU selection: VpnGw1
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: Contoso would need to validate availability and redeploy if necessary
 - Azure Regional Failure: Contoso would need to redeploy the Foundation VPN Gateways into the secondary region
 - Notes
 - VPN Gateways can be created with [Availability Zones](#) for increased availability
 - A "Warm Spare" strategy would mitigate this activity
- **Load Balancer**
 - Contoso SKU selection: Standard
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: Contoso would need to validate availability and redeploy if necessary
 - Azure Regional Failure: Contoso would need to redeploy the Foundation Load Balancers into the secondary region
 - Notes
 - Depending on the primary region, either a [zone redundant](#) or [cross-regional](#) design could be used to uplift this posture
- **Azure DevOps**
 - Contoso SKU selection: DevOps Services
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
 - Notes
 - DevOps Services is [built upon the Azure backbone](#) and uses [Azure blob storage with geo-replication](#) to ensure resiliency

Area: Data Platform components

- **Storage Account – Azure Data Lake Gen2**
 - Contoso SKU selection: LRS

- DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: Contoso would need to validate availability and redeploy if necessary
 - Azure Regional Failure: Contoso would need to redeploy the Data Platform Storage Accounts and rehydrate them with data in the secondary region
 - Notes
 - Storage Accounts have a broad range of [data redundancy](#) options from primary region redundancy up to secondary region redundancy
 - For Secondary region redundancy data is replicated to the [secondary region asynchronously](#). A failure that affects the primary region may result in data loss if the primary region can't be recovered. Azure Storage typically has an RPO of less than 15 minutes
 - In the case of a regional outage, Storage accounts which, are geo-redundant, would be available in the secondary region as LRS. Additional configuration would need to be applied to uplift these components in the secondary region to be geo-redundant
-
- **Azure Synapse - Pipelines**
 - Contoso SKU selection: Computed Optimized Gen2
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to deploy and [restore](#) the Data Platform Azure Synapse Analytics into the secondary region and redeploy the pipelines
 - Notes
 - Automatic restore points are [deleted after seven days](#)
 - [User-defined restore points](#) are available. Currently, there's a ceiling of 42 user-defined restore points that are automatically [deleted after seven days](#)
 - Synapse can also perform a DB restore in the local or remote region, and then immediately PAUSE the instance. This process will only incur storage costs – and have zero compute costs. This offers a way to keep a "live" DB copy at specific intervals
-
- **Azure Event Hubs**
 - Contoso SKU selection: Standard
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A

- Azure Regional Failure: Contoso would need to redeploy the Event Hubs instance into the secondary region
- Notes
 - When you use the Azure portal, zone redundancy via support for availability zones is [automatically enabled](#), this can be disabled via using the Azure CLI or PowerShell commands
 - This resiliency can be extended to cover a full region outage with [Geo-disaster recovery](#)
- **Azure IoT Hubs**
 - Contoso SKU selection: Standard
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to redeploy the IoT Hub into the secondary region
 - Notes
 - IoT Hub provides [Intra-Region HA](#) and will automatically use an availability zone if created in a [predefined set of Azure regions](#)
- **Azure Stream Analytics**
 - Contoso SKU selection: Standard
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to redeploy the IoT Hub into the secondary region
 - Notes
 - A key feature of Stream Analytics is its ability to recover from [Node failure](#)
- **Azure AI services (formerly Cognitive Services)**
 - Contoso SKU selection: Pay As You Go
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
- **Azure Machine Learning**
 - Contoso SKU selection: General Purpose – D Series instances
 - DR Impact
 - Azure Data Center Failure: Contoso would need to validate availability and redeploy if necessary

- Availability Zone Failure: Contoso would need to validate availability and redeploy if necessary
- Azure Regional Failure: Contoso would need to redeploy Machine Learning into the secondary region
- Notes
 - While the Machine Learning infrastructure is managed by Microsoft; the [associated resources are managed by the customer](#). Only Key Vault is highly available by default
 - Depending on the service criticality supported, Microsoft recommends a [multi-regional deployment](#)
- **Azure Synapse – Data Explorer Pools**
 - Contoso SKU selection: Computed Optimized, Small (4 cores)
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to redeploy Azure Synapse – Data Explorer Pools and pipelines into the secondary region
- **Azure Synapse – Spark Pools**
 - Contoso SKU selection: Compute Optimized Gen2
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to redeploy Azure Synapse – Spark Pools and pipelines into the secondary region
 - Notes
 - If an [external Hive metastore](#) is used, this will also need a recovery strategy in place
 - [Azure Site Recovery](#) can be used for a SQL Server metastore
 - A [MySQL](#) metastore would use the geo-restore feature or cross-regional read replicas
- **Azure Synapse – Serverless and Dedicated SQL Pools**
 - Contoso SKU selection: Compute Optimized Gen2
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso would need to deploy and [restore](#) the Data Platform Azure Synapse Analytics into the secondary region
 - Notes
 - Automatic restore points are [deleted after seven days](#)

- User-defined restore points are available. Currently, there's a ceiling of 42 user-defined restore points that are automatically [deleted after seven days](#)
- Synapse can also perform a DB restore in the local or remote region, and then immediately PAUSE the instance. This will only incur storage costs – and have zero compute costs. This solution offers a way to keep a "live" DB copy at specific intervals
- **Power BI**
 - Contoso SKU selection: Power BI Pro
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: N/A
 - Notes
 - The customer will [not need to do anything](#) if the outage is decided/declared by Power BI team
 - A Failed-over Power BI service instance [only supports read operations](#). Reports that use Direct Query or Live connect [won't work during a failover](#)
- **Azure Cosmos DB**
 - Contoso SKU selection: Single Region Write with Periodic backup
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: N/A
 - Azure Regional Failure: Contoso should monitor, ensuring there are [enough provisioned RUs](#) in the remaining regions to support read & write activities
 - Notes
 - [Single-region accounts may lose availability](#) following a regional outage. To ensure high availability of your Cosmos DB instance, configure it with a single write region and at least a second (read) region and enable Service-Managed failover
 - To avoid the loss of write availability, it advised that production workloads are configured with "enable service-managed failover", enabling automatic failover to [available regions](#)
- **Azure AI Search (formerly Cognitive Search)**
 - Contoso SKU selection: Standard S1
 - DR Impact
 - Azure Data Center Failure: Contoso would need to validate availability and redeploy if necessary
 - Availability Zone Failure: Contoso would need to validate availability and redeploy if necessary

- Azure Regional Failure: Contoso would need to redeploy AI Search into the secondary region
- Notes
 - there's [no built-in mechanism for disaster recovery](#)
 - Implementing multiple AI Search replicas across [availability zones](#) will address the data center outage risk
- **Azure Data Share**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: Contoso would need to validate availability and redeploy if necessary
 - Availability Zone Failure: Contoso would need to validate availability and redeploy if necessary
 - Azure Regional Failure: Contoso would need to redeploy the Data Share into the secondary region
 - Notes
 - Azure Data Share isn't currently supported by [Availability Zones](#)
 - Uplifting Data Share to a [HA deployment](#) will address each of these outage risks
- **Purview**
 - Contoso SKU selection: N/A
 - DR Impact
 - Azure Data Center Failure: N/A
 - Availability Zone Failure: Contoso would need to validate availability and redeploy if necessary
 - Azure Regional Failure: Contoso would need to deploy an instance of Purview into the secondary region
 - Notes
 - This activity would be mitigated by implementing the "Warm Spare" strategy, having a second instance of Azure Purview available in the secondary region
 - A "Warm Spare" approach has the following [key callouts](#):
 - The primary and secondary Azure Purview accounts can't be configured to the same Azure Data Factory, Azure Data Share and Synapse Analytics accounts, if applicable. As a result, the lineage from Azure Data Factory and Azure Data Share can't be seen in the secondary Azure Purview accounts
 - The integration runtimes are specific to an Azure Purview account. Hence, if scans must run in primary and secondary Azure Purview accounts in parallel, multiple self-hosted integration runtimes must be maintained

ⓘ Note

This section is intended as general guidance. The vendor's documentation on disaster recovery, redundancy and backup should be consulted for the correct approach for a new component/service under consideration.

"Azure Data Center Failure" covers the situation where the impacted region does not have **Availability Zones** offered.

If new/updated configuration or releases occurred at the point of the disaster event, these should be checked and redeployed (if necessary) as part of the work to bring the platform up to the current date.

Next steps

Now that you've learned about the scenario details, you can learn about [recommendations related to this scenario](#)

Related resources

- [DR for Azure Data Platform - Overview](#)
- [DR for Azure Data Platform - Architecture](#)
- [DR for Azure Data Platform - Deploy this scenario](#)
- [DR for Azure Data Platform - Summary](#)

DR for Azure Data Platform - Recommendations

Azure Synapse Analytics

Azure Machine Learning

Azure Cosmos DB

Azure Data Lake

Azure Event Hubs

Lessons learned

1. Ensure all the parties involved understand the difference between High Availability (HA) and Disaster Recovery (DR): a common pitfall is to confuse the two concepts and mismatch the solutions associated with them
2. Discuss with the business stakeholders about their expectations regarding the following aspects to define the Recovery Point Objectives (RPO) and Recovery Time Objectives (RTO):
 - a. How much downtime they can tolerate, keeping in mind that usually, the faster the recovery, the higher the cost
 - b. The type of incidents they want to be protected from, mentioning the related likelihood of such event. For example, the probability of a server going down is higher than a natural disaster that impacts all the datacenters across a region
 - c. What impact does the system being unavailable has on their business?
 - d. The OPEX budget for the solution moving forward
3. Consider what degraded service options your end-users can accept. These may include:
 - a. Still having access to visualization dashboards even without the most up-to-date data that is, if the ingestion pipelines don't work, end-users still have access to their data
 - b. Having read access but no write access
4. Your target RTO and RPO metrics can define what disaster recovery strategy you choose to implement:
 - a. Active/Active
 - b. Active/Passive
 - c. Active/Redeploy on disaster
 - d. Rely on Microsoft's SLA
5. Ensure you understand all the components that might impact the availability of your systems, such as:
 - a. Identity management
 - b. Networking topology
 - c. Secret/key management
 - d. Data sources

- e. Automation/job scheduler
 - f. Source repository and deployment pipelines (GitHub, Azure DevOps)
6. Early detection of outages is also a way to decrease RTO and RPO values significantly. Here are a few aspects that should be covered:
- a. Define what an outage is and how it maps to Microsoft's definition of an outage. The Microsoft definition is available on the [Azure Service Level Agreement](#) page at the product or service level.
 - b. An efficient monitoring and alerting system with accountable teams to review those metrics and alerts in a timely manner will help meet the goal
7. [Composite SLAs](#) mean that the more components you have in your architecture, the higher the probability of a failure. You could use composite SLA to define the chances of a component outage.
8. Regarding subscription design, the additional infrastructure for disaster recovery can be stored in the original subscription. PaaS services like ADLS Gen2 or Azure Data Factory typically have native features that allow fail over to secondary instances in other regions while staying contained in the original subscription. Some customers might want to consider having a dedicated resource group for resources used only in DR scenarios for cost purposes
- a. It should be noted that [subscription limits](#) may act as a constraint for this approach
 - b. Other constraints may include the design complexity and management controls to ensure the DR resource groups aren't used for BAU workflows
9. Design the DR workflow based on a solution's criticality and dependencies. For example, don't try to rebuild an Azure Analysis Services instance before your data warehouse is up and running, as it will trigger an error. Leave development labs later in the process, recover core enterprise solutions first
10. Try to identify recovery tasks that can be parallelized across solutions, reducing the total RTO
11. If Azure Data Factory is used within a solution, don't forget to include Self-Hosted integration runtimes in the scope. [Azure Site Recovery](#) is ideal for those machines
12. Manual operations should be automated as much as possible to avoid human errors, especially when under pressure. It's recommended to:
- a. Adopt resource provisioning through Bicep, ARM templates or PowerShell scripts
 - b. Adopt versioning of source code and resource configuration
 - c. Use CI/CD release pipelines rather than click-ops
13. As you have a plan for failover, you should consider procedures to fallback to the primary instances
14. Define clear indicators/metrics to validate that the failover has been success and solutions are up and running or that the situation is back to normal (also known as

primary functional)

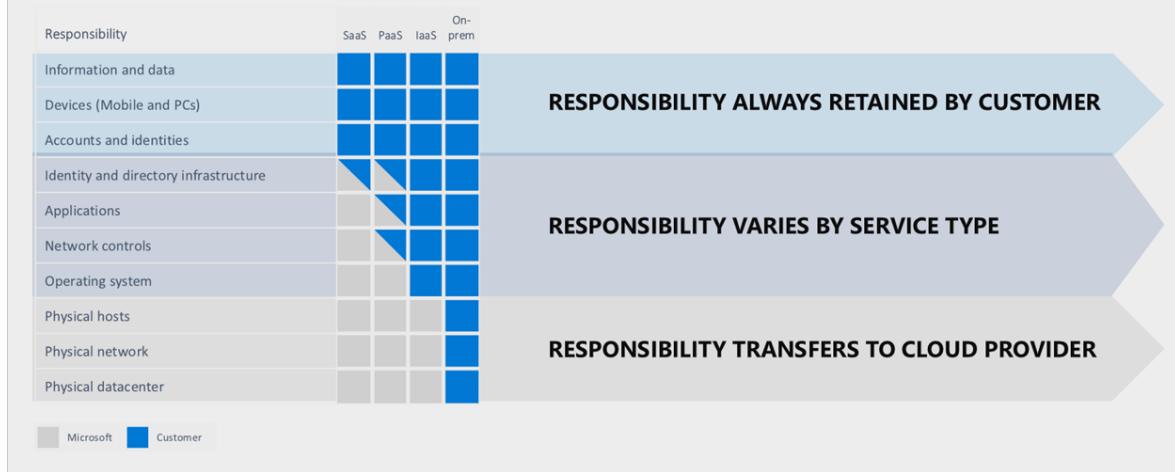
15. Decide if your SLAs should remain the same after a failover or if you allow for degraded service
 - a. This decision will greatly depend on the business service process being supported. For example, the failover for a room-booking system will look much different than a core operational system
16. An RTO/RPO definition should be based on specific user scenarios/solutions rather than at the infrastructure level. It will give you more granularity on what processes and components should be recovered first if there's an outage or disaster
17. Ensure you include capacity checks in the target region before moving forward with a failover: If there's a major disaster, be mindful that many customers will try to failover to the same paired region at the same time, which can cause delays or contention in provisioning the resources
 - a. If these risks are unacceptable, either an Active/Active or Active/Passive DR strategy should be considered
18. A Disaster Recovery plan should be created and maintained to document the recovery process and the action owners. Also, consider that people might be on leave, so be sure to include secondary contacts
19. Regular disaster recovery drills should be performed to validate the DR plan workflow, that it meets the required RTO/RPO, and to train the responsible teams
 - a. Data and configuration backups should also be regularly tested to ensure they are "fit for purpose" to support any recovery activities
20. Early collaboration with teams responsible for networking, identity, and resource provisioning will enable agreement on the most optimal solution regarding:
 - a. How to redirect users and traffic from your primary to your secondary site. Concepts such as DNS redirection or the use of specific tooling like [Azure Traffic Manager](#) can be evaluated
 - b. How to provide access and rights to the secondary site in a timely and secure manner
21. During a disaster, effective communication between the many parties involved is key to the efficient and rapid execution of the plan:
 - a. Decision makers
 - b. Incident response team
 - c. Impacted internal audience
 - d. External teams
22. Orchestration of the different resources at the right time will ensure efficiency in the disaster recovery plan execution

Considerations

Anti-patterns

- **Copy/Paste this article series** This article series is intended to provide guidance to customers looking for the next level of detail for an Azure-specific DR process. As such, it's based upon the generic Microsoft IP and reference architectures rather than any single customer-specific Azure implementation.
While the detail provided will help support a solid foundational understanding, customers must apply their own specific context, implementation, and requirements before obtaining a "fit for purpose" DR strategy and process.
- **Treating DR as a tech-only process** Business stakeholders play a critical role in defining the requirements for DR and completing the business validation steps required to confirm a service recovery. Ensuring that Business stakeholders are engaged across all DR activities will provide a DR process that is "fit for purpose", represents business value, and is executable.
- **"Set and forget" DR plans** Azure is constantly evolving, as are individual customer's use of various components and services. A "fit for purpose" DR process must evolve with them. Either via the SDLC process or periodic reviews, customers should regularly revisit their DR plan. The goal is to ensure the validity of the service recovery plan and that any deltas across components, services or solutions have been accounted for.
- **Paper-based assessments** While the end-to-end simulation of a DR event will be difficult across a modern data eco-system, efforts should be made to get as close as possible to a complete simulation across impacted components. Regularly scheduled drills will build the "muscle memory" required by the organization to be able to execute the DR plan with confidence.
- **Relying on Microsoft to do it all** Within the Microsoft Azure services, there's a clear [division of responsibility](#), anchored by the cloud service tier used:

Shared responsibility model



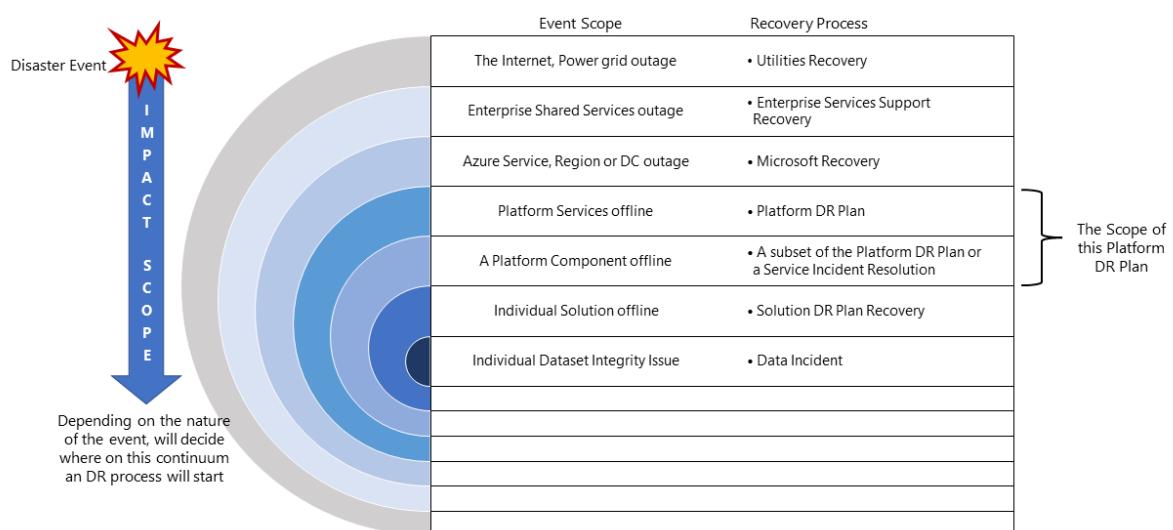
Even if a full [SaaS stack](#) is used, the customer will still retain the responsibility to ensure the accounts, identities, and data is correct/up-to-date, along with the devices used to interact with the Azure services.

Event scope and strategy

Disaster event scope

Different events will have a different scope of impact and, therefore, a different response. The following diagram illustrates this for a disaster event:

A Disaster Event Scope



Disaster strategy options

There are four high-level options for a [disaster recovery strategy](#):

- **Wait for Microsoft** - As the name suggests, the solution is offline until the complete recovery of services in the impacted region by Microsoft. Once recovered, the solution is validated by the customer and then brought up-to-date for service recovery
- **Redeploy on Disaster** - The solution is redeployed manually into an available region from scratch, post-disaster event
- **Warm Spare (Active/Passive)** - A secondary hosted solution is created in an alternate region, and components are deployed to guarantee minimal capacity; however, the components don't receive production traffic. The secondary services in the alternative region may be "turned off" or running at a lower performance level until such time as a DR event occurs
- **Hot Spare (Active/Active)** - The solution is hosted in an active/active setup across multiple regions. The secondary hosted solution receives, processes, and serves data as part of the larger system

DR strategy impacts

While the operating cost attributed to the higher levels of service resiliency often dominates the [Key Design Decision](#) (KDD) for a DR strategy. There are other important considerations.

ⓘ Note

Cost Optimization is one of the five pillars of architectural excellence with Azure's **Well-Architected Framework**. Its goal is to reduce unnecessary expenses and improve operational efficiencies

The DR scenario for this worked example is a complete Azure regional outage that directly impacts the primary region that hosts the Contoso Data Platform. For this

outage scenario, the relative impact on the four high-level DR Strategies are:

Strategy	Description	Impacts					
		Speed to Recovery	Complexity to Execute	Complexity to Implement	Impact to Customers	Above line OPEX Cost	
Wait for Microsoft	Wait for Microsoft to complete the recovery of services.	Orange	White	Green	Red	Green	
Redeploy on Disaster	The solution is redeployed from scratch, post the disaster event.	Red	Orange	Orange	Orange	Orange	
Warm Spare	A secondary hosted solution is created in an alternate region, and roles are deployed to guarantee minimal capacity; however, the roles don't receive production traffic.	Orange	Orange	Red	Green	Orange	
Hot Spare	The solution is hosted in an active/active setup across multiple regions, with both receiving, processing and serving data out.	Green	Green	Red	White	Red	

Classification Key

- **Recovery Time Objective (RTO)** – the expected elapsed time from the disaster event to platform service recovery
- **Complexity to Execute** – the complexity for the organization to execute the recovery activities
- **Complexity to Implement** - the complexity for the organization to implement the DR strategy
- **Impact to Customers** - the direct impact to customers of the data platform service from the DR strategy
- **Above line OPEX cost** - the extra cost expected from implementing this strategy like increased monthly billing for Azure for additional components and additional resources required to support

ⓘ Note

The above table should be read as a comparison between the options - a strategy that has a green indicator is better for that classification than another strategy with a yellow or red indicator

Next steps

Now that you've learned about the recommendations related to the scenario, you can learn how to [deploy this scenario](#)

Related resources

- [DR for Azure Data Platform - Overview](#)
- [DR for Azure Data Platform - Architecture](#)
- [DR for Azure Data Platform - Scenario details](#)
- [DR for Azure Data Platform - Summary](#)

DR for Azure Data Platform - Deploy this scenario

Azure Synapse Analytics

Azure Machine Learning

Azure Cosmos DB

Azure Data Lake

Azure Event Hubs

Customer activities required

Pre-incident

For Azure services

- Be familiar with [Azure Service Health](#) in the Azure portal. This page will act as the “one-stop shop” during an incident
- Consider the use of [Service Health alerts](#), which can be configured to automatically produce notifications when Azure incidents occur

For Power BI

- Be familiar with [Service Health](#) in the Microsoft 365 admin center. This page will act as the “one-stop shop” during an incident
- Consider the use of [Microsoft 365 Admin mobile app](#) to get automatic service incident alert notifications

During the incident

For Azure services

- [Azure Service Health](#) within their Azure management portal will provide the latest updates
 - If there are issues accessing Service Health, refer to the [Azure Status page](#)
 - If there are ever issues accessing the Status page, go to @AzureSupport on X (formerly Twitter)
- If impact/issues don’t match the incident (or persist after mitigation), then [contact support](#) to raise a service support ticket

For Power BI

- The [Service Health](#) page within their Microsoft 365 admin center will provide the latest updates

- If there are issues accessing Service Health, refer to the [Microsoft 365 status page](#)
- If impact/issues don't match the incident (or if issues persist after mitigation), you should raise a [service support ticket](#).

Post Microsoft recovery

See the sections below for this detail.

Post incident

For Azure Services

- Microsoft will publish a PIR to the [Azure portal - Service Health](#) for review

For Power BI

- Microsoft will publish a PIR to the [Microsoft 365 Admin - Service Health](#) for review

Wait for Microsoft process

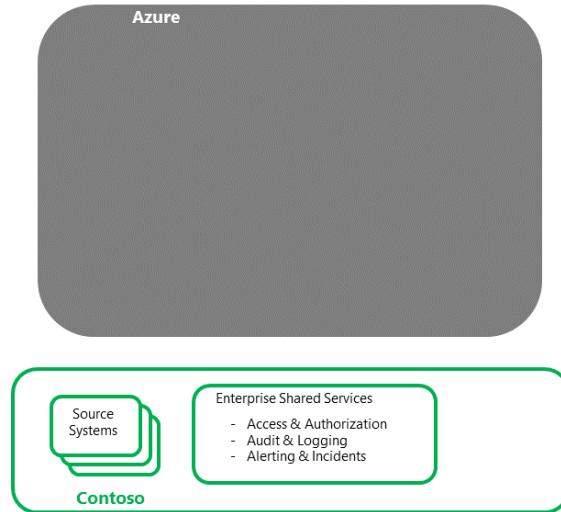
The “Wait for Microsoft” process is simply waiting for Microsoft to recover all components and services in the impacted, primary region. Once recovered, validate the binding of the data platform to enterprise shared or other services, the date of the dataset, and then execute the processes of bringing the system up to the current date.

Once this process has been completed, technical and business SME validation can be completed enabling the stakeholder approval for the service recovery.

Redeploy on disaster

For a “Redeploy on Disaster” strategy, the following high-level process flow can be described.

1. Recover Contoso – Enterprise Shared Services and source systems



- This step is a prerequisite to the recovery of the data platform
- This step would be completed by the various Contoso operational support groups responsible for the enterprise shared services and operational source systems

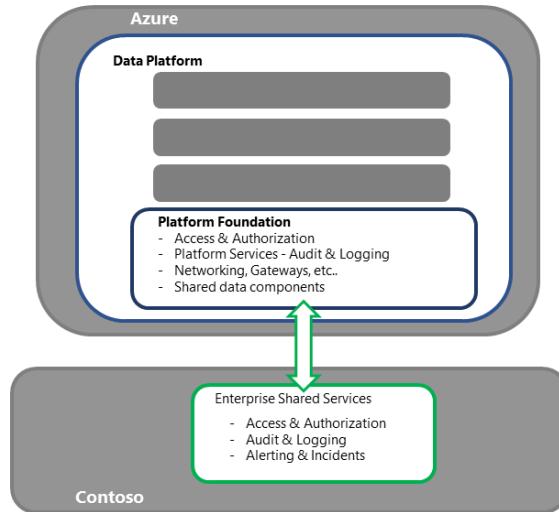
2. Recover Azure services Azure Services refers to the applications and services that make the Azure Cloud offering, are available within the secondary region for deployment.



Azure Services refers to the applications and services that make the Azure Cloud offering, are available within the secondary region for deployment.

- This step is a prerequisite to the recovery of data platform
- This step would be completed by Microsoft and other PaaS/SaaS partners

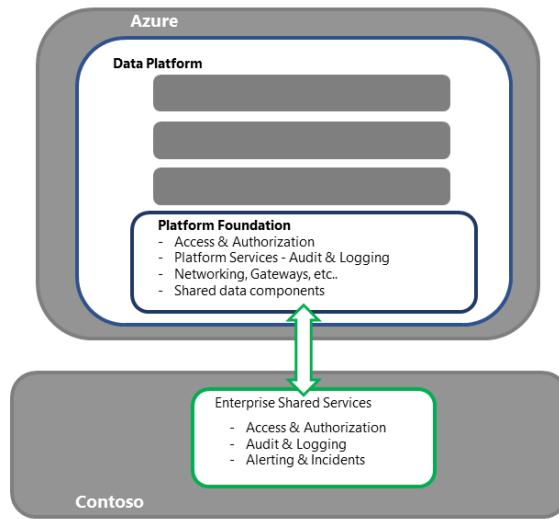
3. Recover the data platform foundation



- This step is the entry point for the Platform recovery activities
- For the Redeployment strategy, each required component/service would be procured and deployed into the secondary region
 - See the [Azure Service and Component Section](#) in this series for a detailed breakdown of the components and deployment strategies
- This process should also include activities like the binding to the enterprise shared services, ensuring connectivity to access/authentication, and validating that the log offloading is working, while also ensuring connectivity to both upstream and downstream processes
- Data/Processing should be confirmed. For example, validation of the timestamp of the recovered platform
 - If there are questions about data integrity, the decision could be made to roll back further in time before executing the new processing to bring the platform up to date
- Having a priority order for processes (based upon business impact) will help in orchestrating the recovery
- This step should be closed out by technical validation unless business users directly interact with the services. If there is direct access, there will need to be a business validation step
- Once validation has been completed, a handover to the individual solution teams to start their own DR recovery process happens
 - This handover should include confirmation of the current timestamp of the data/processes
 - If core enterprise data processes are going to be executed, the individual solutions should be made aware of this - inbound/outbound flows, for

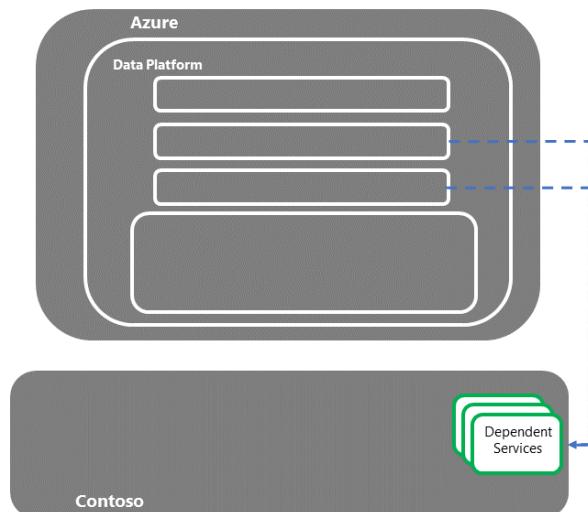
example

4. Recover the individual solutions hosted by the platform



- Each individual solution should have its own DR runbook. The runbooks should at least contain the nominated business stakeholders who will test and confirm that service recovery has been completed
- Depending on resource contention or priority, key solutions/workloads may be prioritized over others - core enterprise processes over ad hoc labs, for example
- Once the validation steps have been completed, a handover to the downstream solutions to start their DR recovery process happens

5. Handover to downstream, dependent systems



- Once the dependent services have been recovered, the E2E DR recovery process is complete

① Note

While it's theoretically possible to completely automate an E2E DR process, it's unlikely given the risk of the event vs. the cost of the SDLC activities required to cover the E2E process

6. **Fallback to the primary region** Fallback is the process of moving the data platform service and its data back to the primary region, once it's available for BAU.

Depending on the nature of the source systems and various data processes, fallback of the data platform could be done independently of other parts of the data eco-system.

Customers are advised to review their own data platform's dependencies (both upstream and downstream) to make the appropriate decision. The following section assumes an independent recovery of the data platform.

- Once all required components/services have become available in the primary region, customers would complete a smoke-test to validate the Microsoft recovery
- Component/Service configuration would be validated. Deltas would be addressed via redeployment from source control
- The system date in the primary region would be established across stateful components. The delta between the established date and the date/timestamp in the secondary region should be addressed by re-executing or replaying the data ingestion processes from that point forward
- With approval from both business and technical stakeholders, a fallback window would be selected. Ideally, during a lull in system activity and processing
- During the fallback, the primary region would be brought into sync with the secondary region, before the system was switched over
- After a period of a parallel run, the secondary region would be taken offline from the system
- The components in the secondary region would either be dropped or stripped back, depending on the DR strategy selected

Warm spare process

For a "Warm Spare" strategy, the high-level process flow is closely aligned to that of the "Redeploy on Disaster", the key difference being that components have already been

procured in the secondary region. This strategy eliminates the risk of resource contention from other organizations looking to complete their own DR in that region.

Hot spare process

The "Hot Spare" strategy means that the Platform services including PaaS and IaaS systems will persist despite the disaster event as the secondary systems run in tandem with the primary systems. As with the "Warm Spare" strategy, this strategy eliminates the risk of resource contention from other organizations looking to complete their own DR in that region.

Hot Spare customers would monitor the Microsoft recovery of components/services in the primary region. Once completed, customers would validate the primary region systems and complete the fallback to the primary region. This process would be similar to the DR Failover process that is, check the available codebase and data, redeploying as required.

ⓘ Note

A special note here should be made to ensure that any system metadata is consistent between the two regions.

- Once Fallback to the primary has been completed, the system load balancers can be updated to bring the primary region back into system topology. If available, a canary release approach can be used to incrementally switch the primary region on for the system.

DR plan structure

An effective DR plan presents a step-by-step guide for service recovery that can be executed by an Azure technical resource. As such, the following lists a proposed MVP structure for a DR Plan.

- Process Requirements
 - Any customer DR process-specific detail, such as the correct authorization required to start DR, and make key decisions about the recovery as necessary (including "definition of done"), service support DR ticketing reference, and war room details
 - Resource confirmation, including the DR lead and executor backup. All resources should be documented with primary and secondary contacts,

- escalation paths, and leave calendars. In critical DR situations, roster systems may need to be considered
- Laptop, power packs and/or backup power, network connectivity and mobile phone details for the DR executor, DR backup and any escalation points
 - The process to be followed if any of the process requirements aren't met
- Contact Listing
 - DR leadership and support groups
 - Business SMEs who will complete the test/review cycle for the technical recovery
 - Impacted Business Owners, including the service recovery approvers
 - Impacted Technical Owners, including the technical recovery approvers
 - SME support across all impacted areas, including key solutions hosted by the platform
 - Impact Downstream systems – operational support
 - Upstream Source systems – operational support
 - Enterprise shared services contacts. For example, access/authentication support, security monitoring and gateway support
 - Any external or third party vendors, including support contacts for cloud providers
 - Architecture design
 - Describe the end-end to E2E scenario detail, and attach all associated support documentation
 - Dependencies
 - List out all the component's relationships and dependencies
 - DR Prerequisites
 - Confirmation that upstream source systems are available as required
 - Elevated access across the stack has been granted to the DR executor resources
 - Azure services are available as required
 - The process to be followed if any of the prerequisites haven't been met
 - Technical Recovery - Step-by-Step instructions
 - Run order
 - Step description
 - Step prerequisite
 - Detailed process steps for each discrete action, including URL's
 - Validation instructions, including the evidence required
 - Expected time to complete each step, including contingency
 - The process to be followed if the step fails
 - The escalation points in the case of failure or SME support
 - Technical Recovery - Post requisites
 - Confirm the current date timestamp of the system across key components
 - Confirm the DR system URLs & IPs

- Prepare for the Business Stakeholder review process, including confirmation of systems access and the business SMEs completing the validation and approval
- Business Stakeholder Review and Approval
 - Business resource contact details
 - The Business validation steps as per the technical recovery above
 - The Evidence trail required from the Business approver signing off the recovery
- Recovery Post requisites
 - Handover to operational support to execute the data processes to bring the system up to date
 - Handover the downstream processes and solutions – confirming the date and connection details of the DR system
 - Confirm recovery process complete with the DR lead – confirming the evidence trail and completed runbook
 - Notify Security administration that elevated access privileges can be removed from the DR team

Callouts

- It's recommended to include system screenshots of each step process. These screenshots will help address the dependency on system SMEs to complete the tasks
 - To mitigate the risk from quickly evolving Cloud services, the DR plan should be regularly revisited, tested, and executed by resources with current knowledge of Azure and its services
- The technical recovery steps should reflect the priority of the component and solution to the organization. For example, core enterprise data flows are recovered before ad hoc data analysis labs
- The Technical recovery steps should follow the order of the workflows (typically left to right), once the foundation components/service like Key Vault have been recovered. This strategy will ensure upstream dependencies are available and components can be appropriately tested
- Once the step-by-step plan has been completed, a total time for activities with contingency should be obtained. If this total is over the agreed RTO, there are several options available:
 - Automate selected recovery processes (where possible)
 - Look for opportunities to run selected recovery steps in parallel (where possible). However, noting that this strategy may require additional DR executor resources.
 - Uplift key components to higher levels of service tiers such as PaaS, where Microsoft takes greater responsibility for service recovery activities

- Extend the RTO with stakeholders

DR testing

The nature of the Azure Cloud service offering results in constraints for any DR testing scenarios. Therefore, the guidance is to stand up a DR subscription with the data platform components as they would be available in the secondary region.

From this baseline, the DR plan runbook can be selectively executed, paying specific attention to the services and components that can be deployed and validated. This process will require a curated test dataset, enabling the confirmation of the technical and business validation checks as per the plan.

A DR plan should be tested regularly to not only ensure that it's up to date, but also to build "muscle memory" for the teams performing failover and recovery activities.

- Data and configuration backups should also be regularly tested to ensure they are "fit for purpose" to support any recovery activities.

The key area to focus on during a DR test is to ensure the prescriptive steps are still correct and the estimated timings are still relevant.

- If the instructions reflect the portal screens rather than code – the instructions should be validated at least every 12 months due to the cadence of change in cloud.

While the aspiration is to have a fully automated DR process, full automation may be unlikely due to the rarity of the event. Therefore, it's recommended to establish the recovery baseline with DSC IaC used to deliver the platform and then uplift as new projects build upon the baseline.

- Over time as components and services are extended, an NFR should be enforced, requiring the production deployment pipeline to be refactored to provide coverage for DR.

If your runbook timings exceed your RTO, there are several options:

- Extend the RTO with stakeholders
- Lower the time required for the recovery activities, via automation, running tasks in parallel or migration to higher cloud server tiers

Azure Chaos Studio

Azure Chaos Studio is a managed service for improving resilience by injecting faults into your Azure applications. Chaos Studio enables you to orchestrate fault injection on your Azure resources in a safe and controlled way, using experiments. See the product documentation for a description of the types of faults currently supported.

The current iteration of Chaos Studio only covers a subset of [Azure components and services](#). Until more fault libraries are added, Chaos Studio is a recommended approach for isolated resiliency testing rather than full system DR testing.

More information on Chaos studio can be found [here](#)

Azure Site Recovery

For IaaS components, Azure Site Recovery will protect most workloads running on a [supported VM or physical server](#)

There is strong guidance for:

- Executing an Azure VM Disaster Recovery Drill
- Executing a DR failover to a Secondary Region
- Executing a DR fallback to the Primary Region
- Enabling automation of a DR Plan

Related resources

- [Architecting for resiliency and availability](#)
- [Business continuity and disaster recovery](#)
- [Backup and disaster recovery for Azure applications](#)
 - [Recover from the loss of an Azure region](#)
- [Resiliency in Azure](#)
 - [Business continuity management in Azure](#)
- [Service Level Agreements Summary ↗](#)
 - [Azure Status ↗](#)
 - [Azure DevOps Status ↗](#)
- [Five Best Practices to Anticipate Failure ↗](#)

Next steps

Now that you've learned how to deploy the scenario, you can read a [summary](#) of the DR for Azure data platform series.

Related resources

- [DR for Azure Data Platform - Overview](#)
- [DR for Azure Data Platform - Architecture](#)
- [DR for Azure Data Platform - Scenario details](#)
- [DR for Azure Data Platform - Recommendations](#)

DR for Azure Data Platform - Summary

Azure Synapse Analytics Azure Machine Learning Azure Cosmos DB Azure Data Lake Azure Event Hubs

The articles in this series describe an approach for designing a disaster recovery (DR) strategy for an Azure-based data platform.

Every organization's data needs are different, but the guidance provided will act as starting point, enabling the design of a DR strategy that fits your business requirements.

Key terms glossary

 [Expand table](#)

Definition	Notes
Microsoft Entra ID	Microsoft Entra ID
ACL	Access Control Lists
ADLS	Azure Data Lake Storage
AKS	Azure Kubernetes Service
ARM	Azure Resource Manager
BAU	Business As Usual
BC	Business Continuity
BCDR	Business Continuity and Disaster

Definition	Notes	
	Recovery	
CI/CD	Continuous Integration & Continuous Deployment	Azure DevOps - CI/CD Overview
DR	Disaster Recovery	
DNS	Domain Name System	
DSC	Desired State Configuration	
E2E	End to End	
GRS	Geo-redundant storage	Storage Redundancy
HA	High Availability	
HSM	Hardware Security Module	Azure Managed HSM documentation
IAC	Infrastructure As Code	
IAAS	Infrastructure As A Service	What is IaaS? Infrastructure as a Service ↗
IOT	Internet of Things	
ITIL	Information Technology Infrastructure Library	

Definition	Notes	
KDD	Key Design Decision	
LRS	Locally Redundant Storage	
MSEE	Microsoft Enterprise Edge	
MTO	Maximum Tolerable Outage	The maximum acceptable amount of time that dependent business processes can tolerate the platform being unavailable. $MTO = RTO + WRT$
MVP	Minimum Viable Product	
NFR	Non-Functional Requirement	Requirements on a solution or system related to usability, scalability, security and accessibility rather than functionality.
NSG	Network Security Group	Azure network security groups overview
OPEX	Operational Expenditure	
PaaS	Platform As A Service	What is PaaS? Platform as a Service ↗
PIR	Post-Incident Review	
RA-GZRS	Read-Access Geo-Zone-Redundant Storage	Data redundancy - Azure Storage
RPO	Recovery Point Objective	The maximum acceptable amount of data/processing loss, measured in time.

Definition	Notes
RTO	Recovery Time Objective
	The maximum acceptable amount of time needed to recover the Data platform service and bring it back online.
SaaS	Software As A Service
	What is SaaS? Software as a Service ↗
SDLC	Software Development Lifecycle
SME	Subject Mater Expert
SLA	Service Level Agreement
SSO	Single sign-on
UDR	User Defined Route
	Azure virtual network traffic routing
VM	Virtual Machine
	Virtual Machines (VMs) for Linux and Windows ↗
VNET	Azure – Virtual Network
	Azure Virtual Network
WRT	Work Recovery Time
	The maximum acceptable amount of time that is required to update the platform data/processing from the recovery point to the current period, enabling the business/solutions to use the service as BAU.
ZRS	Zone-Redundant Storage
	Data redundancy - Azure Storage

Contributors

Microsoft maintains this article. The following contributors originally wrote it.

Principal authors:

- [Hajar Habjaoui](#) | Cloud Solution Architect
- [Justice Zisanhi](#) | Cloud Solution Architect
- [Scott Mckinnon](#) | Cloud Solution Architect

Other contributors:

- [Ananth Prakash](#) | Senior CSA Manager
- [Fabio Braga](#) | Chief Architect – Customer Success
- [Rolf Tesmer](#) | Senior Cloud Solution Architect

To see nonpublic LinkedIn profiles, sign in to LinkedIn.

Next steps

- [Cloud Adoption Framework](#)
- [Mission-critical workload](#)
- [Well-Architected Framework](#)

Related resources

- [DR for Azure Data Platform - Overview](#)
- [DR for Azure Data Platform - Architecture](#)
- [DR for Azure Data Platform - Scenario details](#)
- [DR for Azure Data Platform - Recommendations](#)

Actuarial risk analysis and financial modeling

Azure Blob Storage

Azure Data Factory

Azure Cosmos DB

Azure HDInsight

Azure Databricks

Over the last several years, insurers and companies that provide insurance-like products have seen several new regulations come into place. These new regulations have required more extensive financial modeling for insurers. The European Union enacted [Solvency II](#). This law requires insurers to demonstrate that they've done their proper analysis to validate that the insurer will be solvent at the end of the year. Insurers who provide variable annuities have to follow [Actuarial Guideline XLIII](#) with extensive analysis of asset and liability cash flows. All types of insurers, including those who distribute insurance-like products, will have to implement [International Financial Reporting Standard 17](#) (IFRS 17) by 2021. (IFRS stands for International Financing Reporting Standards.) Other regulations exist, depending on the jurisdictions the insurers operate in. These standards and regulations require actuaries to use compute-intensive techniques when modeling assets and liabilities. Much of the analysis will make use of stochastically generated scenario data over seriatim inputs of things like assets and liabilities. Beyond regulatory needs, actuaries do a fair amount of financial modeling and computation. They create the input tables for the models that generate the regulatory reports. Internal grids don't satisfy the computational needs, so actuaries are steadily moving to the cloud.

Actuaries move to the cloud to get more time to review, evaluate, and validate results. When regulators audit insurers, the actuaries need to be able to explain their results. The move to the cloud gives them access to the computing resources to run 20,000 hours of analysis in 24-120 hours of clock time, through the power of parallelization. To assist with this need for scale, many of the companies that create actuarial software provide solutions that allow calculations to run in Azure. Some of these solutions are built on technologies that run on-premises and on Azure, like the high-performance computing PowerShell solution, [HPC Pack](#). Other solutions are native to Azure and use [Azure Batch](#), [Virtual Machine Scale Sets](#), or a custom scaling solution.

In this article, we'll look at how actuarial developers can use Azure, coupled with modeling packages, to analyze risk. The article explains some of the Azure technologies that the modeling packages use to run at scale on Azure. You can use the same technology to do further analysis of your data. We look at the following items:

- Running larger models in less time, in Azure.
- Reporting on the results.

- Managing data retention.

Whether you service life, property and casualty, health, or other insurance, you need to create financial and risk models of your assets and liabilities. You can then adjust your investments and premiums so that you stay solvent as an insurer. IFRS 17 reporting adds changes to the models that the actuaries create, like calculating the contractual service margin (CSM), which changes how insurers manage their profit through time.

Running more in less time, in Azure

You believe in the promise of the cloud: it can run your financial and risk models faster and easier. For many insurers, a back of the envelope calculation shows the problem: they need years, or even decades, of sequential time to run these calculations from start to finish. You need technology to solve the runtime problem. Your strategies are:

- Data preparation: Some data changes slowly. Once a policy or service contract is in force, claims move at a predictable pace. You can prepare the data needed for model runs as it arrives, eliminating a need to plan much time for data cleansing and preparation. You may also use clustering to create stand-ins for seriatim data through weighted representations. Fewer records usually results in reduced computation time.
- Parallelization: If you need to do the same analysis to two or more items, you may be able to perform the analysis simultaneously.

Let's look at these items individually.

Data preparation

Data flows in from several different sources. You have semi-structured policy data in your books of business. You also have information about the insured people, companies, and the items that appear in various application forms. Economic Scenario Generators (ESGs) produce data in a variety of formats which may need translation to a form your model can use. Current data on values of assets also needs normalization. The stock market data, cash flow data on rentals, payment information on mortgages, and other asset data all need some preparation when you move from the source to the model. Finally, you should update any assumptions based on recent experience data. To speed up a model run, you prepare the data ahead of time. At run time you do any necessary updates to add in changes since the last scheduled update.

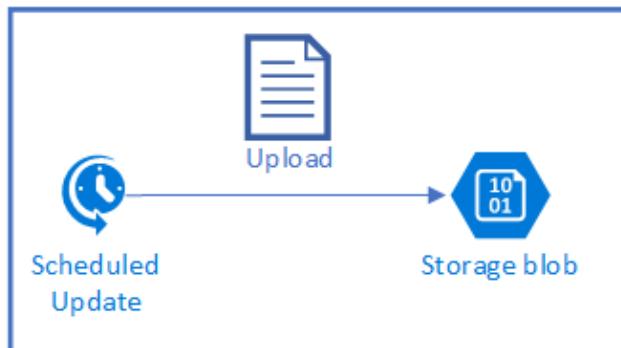
So, how do you prepare the data? Let's first look at the common bits and then look at how to work with the different ways data will appear. First, you want a mechanism to get all the changes since the last synchronization. That mechanism should include a value

that's sortable. For recent changes, that value should be greater than any previous change. The two most common two mechanisms are an ever-increasing ID field or a timestamp. If a record has an increasing ID key, but the rest of the record contains fields that can be updated, then you need to use something like a "last-modified" timestamp to find the changes. Once the records have been processed, record the sortable value of the last item updated. This value, probably a timestamp on a field called *lastModified*, becomes your watermark, used for subsequent queries on the data store. Data changes can be processed in many ways. Here are two common mechanisms that use minimal resources:

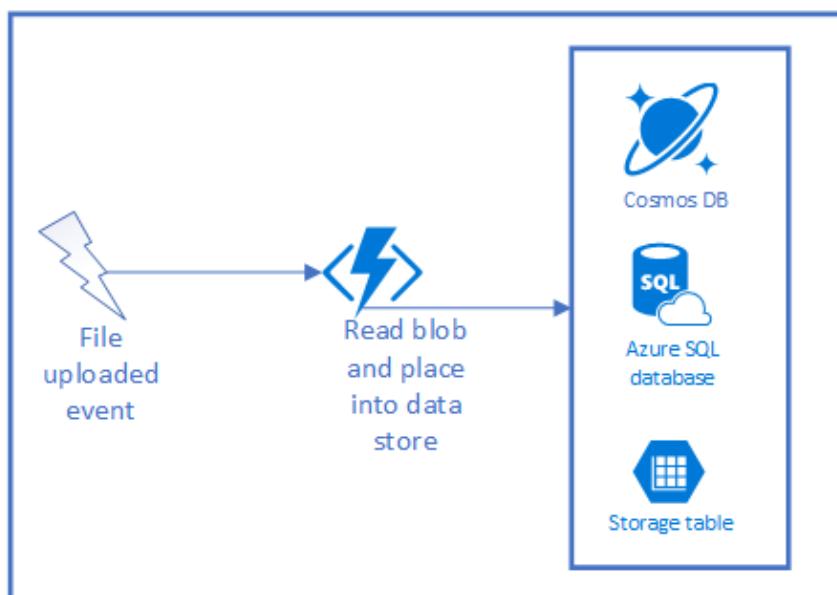
- If you have hundreds or thousands of changes to process, upload the data to blob storage. Use an event trigger in [Azure Data Factory](#) to process the changeset.
- If you have small sets of changes to process, or if you want to update your data as soon as a change happens, put each change into a queue message that's hosted by [Service Bus](#) or [Storage Queues](#). [This article](#) has a great explanation about the tradeoffs between the two queueing technologies. Once a message is in a queue, you can use a trigger in Azure Functions or Azure Data Factory to process the message.

The following figure illustrates a typical scenario. First, a scheduled job collects some set of data and places the file into storage. The scheduled job can be a CRON job running on premises, a [Scheduler task](#), [Logic App](#), or anything that runs on a timer. Once the file is uploaded, an [Azure Function](#) or [Data Factory](#) instance can be triggered to process the data. If the file can be processed in a short period of time, use a **Function**. If the processing is complex, requires AI or other complex scripting, you may find that [HDInsight](#), [Azure Databricks](#), or something custom works better. When done, the file winds up in a usable form as a new file or as records in a database.

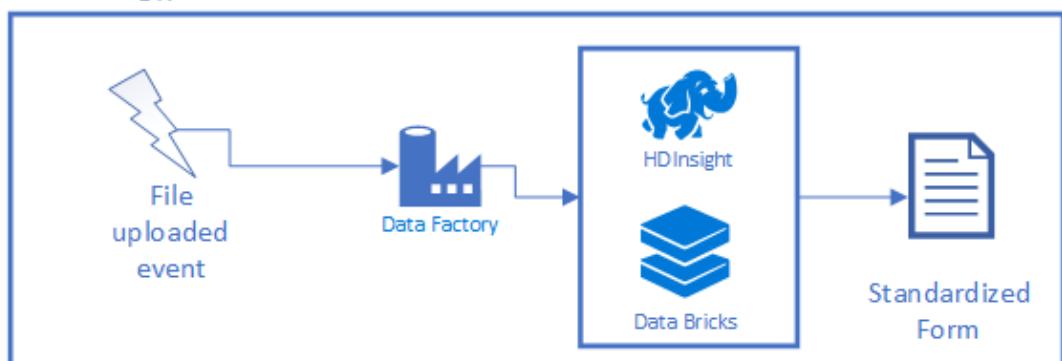
1



2



OR



Once the data is in Azure, you need to make it usable by the modeling application. You can write code to do custom transformations, run the items through **HDInsight** or Azure **Databricks** to ingest larger items, or copy the data into the right data sets. The use of big data tools can also help you do things like transform unstructured data into structured data as well as run any AI and machine learning over the received data. You can also host virtual machines, upload data straight to data sources from on-premises, call Azure Functions directly, and so on.

Later, the data needs to be consumed by your models. The way you do this depends largely on how the calculations need to access data. Some modeling systems require all data files to live on the node that runs the calculation. Others can make use of

databases like [Azure SQL Database](#), [MySQL](#), or [PostgreSQL](#). You can use a low-cost version of any of these items, and then scale up the performance during a modeling run. This gives you the price you need for everyday work. Plus it gives you the extra speed just when thousands of cores are requesting data. Normally, this data will be read-only during a modeling run. If your calculations occur across multiple regions, consider using [Azure Cosmos DB](#) or [Azure SQL geo-replication](#). Both provide mechanisms to automatically replicate data across regions with low latency. Your choice depends on the tools your developers know, how you've modeled your data, and the number of regions used for your modeling run.

Do spend some time thinking about where to store your data. Understand how many simultaneous requests for the same data will exist. Think about how you'll distribute the information:

- Does each computational node get its own copy?
- Is the copy shared through some high bandwidth location?

If you keep data centralized using Azure SQL, you'll likely keep the database at a lower-priced tier most of the time. If the data is only used during a modeling run and isn't updated very often, Azure customers will go so far as to back up the data and turn off their database instances between runs. The potential savings are large. Customers can also make use of [Azure SQL Elastic Pools](#). These are designed to control database costs, especially when you don't know which databases will be under a lot of load at different times. The elastic pools allow a collection of databases to use as much power as they need, then scale back once demand shifts elsewhere in the system.

You may need to disable data synchronization during a modeling run so that calculations later in the process are using the same data. If you use queuing, then disable the message processors but allow the queues to receive data.

You can also use the time before the run to generate economic scenarios, update actuarial assumptions, and generally update other static data. Let's look at economic scenario generation (ESG). The [Society of Actuaries](#) provides the [Academy Interest Rate Generator](#) (AIRG), an ESG which models U. S. Treasury yields. AIRG is prescribed for use in items like Valuation Manual 20 (VM-20) calculations. Other ESGs may model the stock market, mortgages, commodity prices, and so on.

Because your environment pre-processes the data, you can also run other pieces early. For example, you may have things which you model that use records to represent larger populations. One usually does this by clustering records. If the dataset is updated sporadically, such as once a day, one can reduce the record set to what will be used in the model as part of the ingestion process.

Let's look at a practical example. With IFRS-17, you need to group your contracts together such that the maximum distance between the start dates for any two contracts is less than one year. Let's assume that you do this the easy way and use the contract year as the grouping mechanism. This segmentation can be done while data is loaded into Azure by reading through the file and moving the records to the appropriate year groupings.

Focusing on data preparation reduces the time necessary to run the model components. By getting the data in early, you can save clock time for running your models.

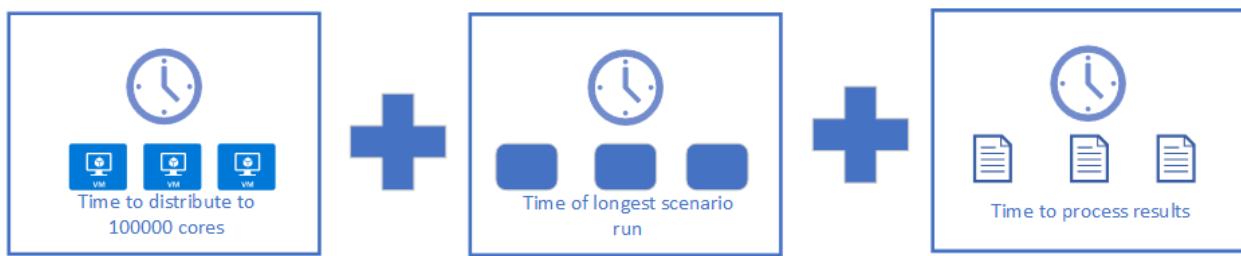
Parallelization

Proper parallelization of the steps can shrink clock execution time dramatically. This speedup happens by streamlining the pieces that you implement and knowing how to express your model in a way that allows for two or more activities to run simultaneously. The trick is to find balance between the size of the work request and the productivity of an individual node. If the task spends more time in setup and cleanup than it does in evaluation, you went too small. If the task is too large, execution time doesn't improve. You want the activity to be small enough to spread over multiple nodes and make a positive difference in elapsed execution time.

To get the most out of your system, you need to understand the workflow for your model and how the computations interact with the ability to scale out. Your software may have a notion of jobs, tasks, or something similar. Use that knowledge to design something that can split work up. If you have some custom steps in your model, design those to allow for inputs to be split into smaller groups for processing. This design is often referred to as a scatter-gather pattern.

- Scatter: split the inputs along natural lines and allow separate tasks to run.
- Gather: as the tasks complete, collect their outputs.

When splitting things, also know where the process needs to synchronize before moving forward. There are a few common places where people split things up. For nested stochastic runs, you may have a thousand outer loops with a set of inflection points that run inner loops of one hundred scenarios. Each outer loop can run simultaneously. You halt at an inflection point, then run the inner loops simultaneously, bring the information back to adjust the data for the outer loop, and go forward again. The following figure illustrates the workflow. Given enough compute, you can run the 100,000 inner loops on 100,000 cores, bringing processing time down to the sum of the following times:



Distribution will increase a bit depending on how that is done. It may be as simple as constructing a small job with the right parameters, or as complex as copying 100K files to the right places. Processing results can even be sped up if you can distribute the result aggregation using Apache Spark from Azure HDInsight, Azure Databricks, or your own deployment. For example, computing averages is a simple matter of remembering the number of items seen so far and the sum. Other computations may work better on a single machine with thousands of cores. For those, you can make use of GPU-enabled machines in Azure.

Most actuarial teams start this journey by moving their models to Azure. Then they collect timing data on the various steps in the process. Then they sort the clock time for each step from longest to shortest elapsed time. They won't look at total execution time since something may consume thousands of core hours but only 20 minutes elapsed time. For each of the longest running job steps the actuarial developers look for ways to decrease the elapsed time while getting the right results. This process repeats regularly. Some actuarial teams will set a target run time, let's say an overnight hedging analysis has a goal of running in under 8 hours. As soon as the time creeps over 8.25 hours, some part of the actuarial team will switch over to improve the time of the longest piece in the analysis. Once they get the time back under 7.5 hours, they switch back to development. The heuristics to go back and optimize vary among actuaries.

To run all this you have several options. Most actuarial software works with compute grids. Grids that work on-premises and on Azure use either [HPC Pack](#), a partner package, or something custom. Grids optimized for Azure will use [Virtual Machine Scale Sets](#), [Batch](#), or something custom. If you choose to use either Scale Sets or Batch, make sure to look at their support for low priority virtual machines (VMs) ([Scale Sets](#) low priority docs, [Batch](#) low priority docs). A low priority VM is a VM running on hardware that you can rent for a fraction of the normal price. The lower price is available because low priority VMs may be preempted when capacity demands it. If you have some flexibility in your time budget, the low priority VMs provide a great way to reduce the price of a modeling run.

If you need to coordinate the execution and deployment across many machines, perhaps with some machines running in different regions, you can take advantage of CycleCloud. CycleCloud costs nothing extra. It orchestrates data movement when necessary. This includes allocation, monitoring, and shut down of the machines. It can

even handle low priority machines, making sure expenses are contained. You can go so far as to describe the mix of machines you need. For example, maybe you need a class of machine but can run well on any version that has 2 or more cores. Cycle can allocate cores across those machine types.

Reporting on the results

Once the actuarial packages have run and produced their results, you'll have several regulator-ready reports. You'll also have a mountain of new data that you may want to analyze to generate insights not required by regulators or auditors. You may want to understand the profile of your best customers. Using insights, you can tell marketing what a low-cost customer looks like so that marketing and sales can find them faster. Likewise, you can use the data to discover which groups benefit the most from having the insurance. For example, you may discover that people who take advantage of an annual physical found out about early stage health issues earlier. This saves the insurance company time and money. You can use that data to drive behavior in your customer base.

To do this, you'll want access to plenty of data science tooling as well as some pieces for visualization. Depending on how much investigation you want to do, you can start with a [Data Science VM](#) which can be provisioned from the Azure Marketplace. These VMs have both Windows and Linux versions. Installed, you'll find Microsoft R Open, Microsoft Machine Learning Server, Anaconda, Jupyter, and other tools ready to go. Throw in a little R or Python to visualize the data and share insights with your colleagues.

If you need to do more analysis, you can use Apache data science tools like Spark, Hadoop, and others via either HDInsight or Databricks. Use these more for when the analysis needs to be done regularly and you want to automate the workflow. They're also useful for live analysis of large datasets.

Once you've found something that's interesting, you need to present the results. Many actuaries will start by taking the sample results and plugging them into Excel to create charts, graphs, and other visualizations. If you want something that also has a nice interface for drilling into the data, take a look at [Power BI](#). Power BI can make some nice visualizations, display the source data, and can allow for explaining the data to the reader through the addition of [ordered, annotated bookmarks](#).

Data retention

Much of the data you bring into the system needs to be preserved for future audits. Data retention requirements typically range from 7 to 10 years, but requirements vary.

Minimal retention involves:

- A snapshot of the original inputs to the model. This includes assets, liabilities, assumptions, ESGs, and other inputs.
- A snapshot of the final outputs. This includes any data used to create reports that are presented to regulatory bodies.
- Other important, intermediate results. An auditor will ask why your model came up with some result. You need to retain evidence about why the model made certain choices or came up with particular numbers. Many insurers will choose to keep the binaries used to produce the final outputs from the original inputs. Then, when questioned, they rerun the model to get a fresh copy of the intermediate results. If the outputs match, then the intermediate files should also contain the explanations they need.

During the model run, the actuaries use data delivery mechanisms that can handle the request load from the run. Once the run is complete and data is no longer needed, they preserve some of the data. At a minimum, an insurer should preserve the inputs and the runtime configuration for any reproducibility requirements. Databases are preserved to backups in Azure Blob Storage and servers are shut down. Data on high speed storage also moves to the less expensive Azure Blob Storage. Once in Blob Storage, you can choose the data tier used for each blob: hot, cool, or archive. Hot storage works well for frequently accessed files. Cool storage is optimized for infrequent data access. Archive storage is best for holding auditable files but the price savings comes at a latency cost: archived tier data latency is measured in hours. Read [Azure Blob storage: Hot, cool, and archive storage tiers](#) to fully understand the different storage tiers. You can manage data from creation through deletion with lifecycle management. URLs for blobs stay static, but where the blob is stored gets cheaper over time. This feature will save a lot of money and headaches for many users of Azure Storage. You can learn about the ins and outs in [Managing the Azure Blob Storage Lifecycle](#). The fact that you can automatically delete files is wonderful: it means you won't accidentally expand an audit by referring to a file that is out of scope because the file itself can be removed automatically.

Considerations

If the actuarial system you run has an on-premises grid implementation, that grid implementation will likely run on Azure too. Some vendors have specialized Azure implementations that run at hyperscale. As part of the move to Azure, move your internal tooling over also. Actuaries everywhere have been finding that their data science skills work well on their laptop or with a large environment. Look for things your team already does: maybe you have something that uses deep learning but takes hours or days to run on one GPU. Try running the same workload on a machine with four high-

end GPUs and look at the run times; odds are good you'll see significant speedups for things you already have.

As things improve, make sure that you also build out some data synchronization to feed the modeling data. A model run can't start until the data is ready. This may involve adding some effort so that you send only data which has changed. The actual approach depends on the data size. Updating a few MB maybe isn't a big deal but reducing the number of gigabyte uploads speeds things a lot.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Scott Seely](#) | Software Architect

Next steps

- R Developers: [Run a parallel R simulation with Azure Batch](#)
- ETL with Databricks: [Extract, transform, and load data using Azure Databricks](#)
- ETL with HDInsight: [Extract, transform, and load data using Apache Hive on Azure HDInsight](#)
- [Data Science VM How To \(Linux\)](#)
- [Data Science VM How To \(Windows\)](#)

Related resources

- [Risk grid computing in banking](#)
- [Risk grid computing solution](#)
- [Data management in banking](#)
- [A financial institution scenario for data mesh](#)
- [Big compute architecture style](#)

A financial institution scenario for data mesh

Article • 03/01/2023

This scenario is for customers that want to use cloud-scale analytics for scalability and *data mesh* architectures. It demonstrates a complex scenario with landing zones, data integrations, and data products.

Customer profile

A fictitious enterprise, Woodgrove Bank, is a large financial services company with a world-wide footprint. Woodgrove Bank's data is housed in on-premises and cloud deployment systems. Within the Woodgrove Bank architecture, there are several data warehouse systems for consolidated marketing and integrated reporting. This architecture includes several data lakes for ad hoc analytics and data discovery. Woodgrove Bank applications are interconnected via application integration patterns, which are mostly API-based or event-based.

The current situation

It's challenging for Woodgrove Bank to distribute data to different locations because of the complexity of data warehousing. Integrating new data is time consuming, and it's tempting to duplicate data. Woodgrove Bank finds it difficult to oversee the end-to-end data landscape because of point-to-point connectivity. The bank underestimated the demand for intensive data consumption. New use cases are introduced quickly, one after another. Data governance, such as data ownership and quality, and costs are hard to control. Keeping current with regulations is difficult because Woodgrove Bank doesn't know exactly where its data resides.

Architecture solution: Data mesh

Over the past several years, organizations have recognized that data is at the heart of everything. Data opens new efficiencies, drives innovation, unlocks new business models, and increases customer satisfaction. It's a top priority for companies to use data-driven methods, like data at scale.

Reaching a stage where the deeper value of data is accessible to all organization members is challenging. Legacy and tightly interconnected systems, centralized

monolithic platforms, and complex governance can be significant barriers to generating value out of data.

About data mesh

The concept of data mesh, a term coined by Zhamak Dehghani, encompasses data, technology, processes, and organization. Conceptually, it's an accessible approach to managing data where various domains use their own data. Data mesh challenges the idea of conventional centralization of data. Rather than looking at data as one huge repository, data mesh considers the decomposition of independent data products. This shift, from centralized to federated ownership, is backed by a modern and self-service data platform, which is typically designed by using cloud-native technologies.

When you break down the data mesh concept into building blocks, here are some key points to consider:

- **Data as a product:** Each (organizational) domain operates its data end to end. Accountability lies with the data owner within the domain. Pipelines become a first-class concern of the domains themselves.
- **Federated computational data governance:** To ensure that each data owner can trust the others and share its data products, an enterprise data governance body must be established. The governance body implements data quality, central visibility of data ownership, data access management, and data privacy policies.
- **Domain-oriented data ownership:** The enterprise should ideally define and model each data-domain node within the mesh by applying the principles of domain-oriented design.
- **Self-serve data platform:** A data mesh requires a self-serve data platform that allows users to remove the technical complexity and focus on their individual data use cases.

Cloud-scale analytics

Data-as-a-product thinking and a self-service platform model aren't new to Microsoft. Microsoft has observed best practices of distributed platforms, pipelines across domains, federated ownership, and self-explanatory data for many years.

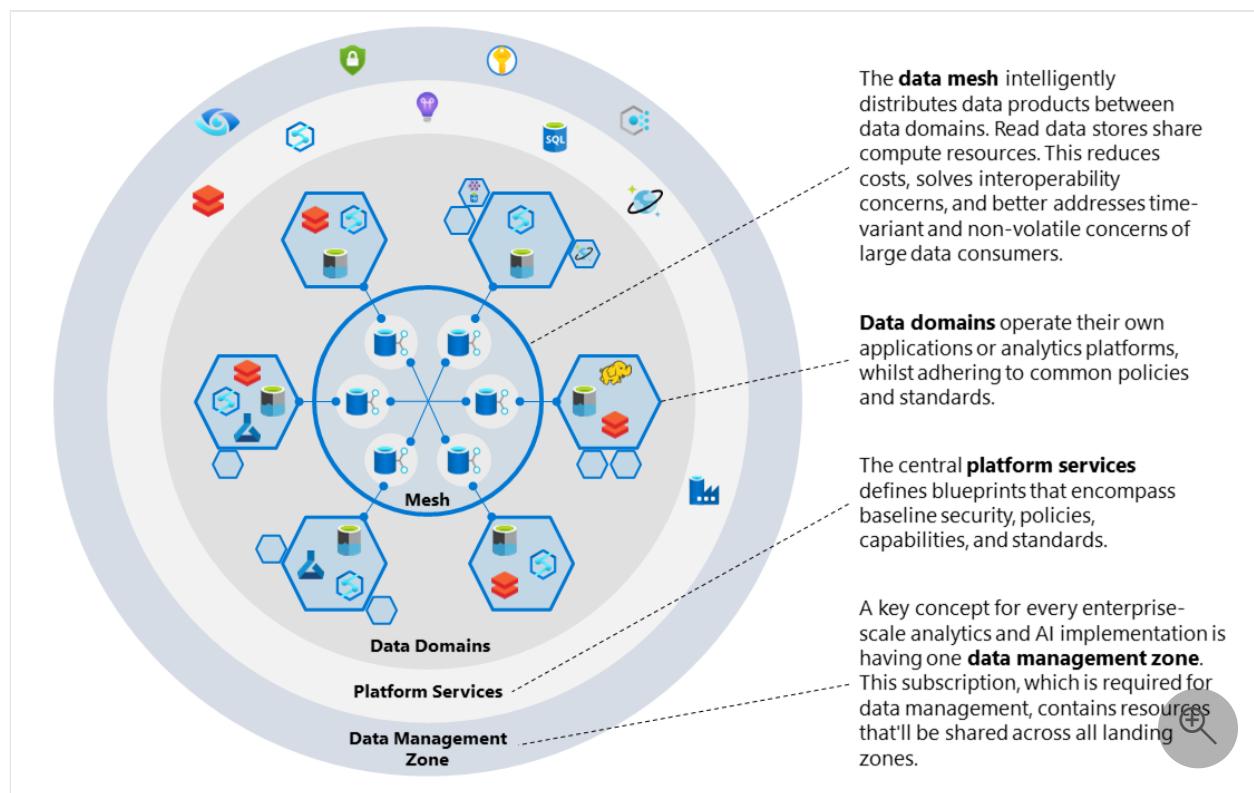
Woodgrove Bank can transition to data mesh by using cloud-scale analytics. Cloud-scale analytics is an open-source and prescriptive blueprint for designing and quickly deploying modern data platforms. It's coupled with Azure's best practices and design principles and is aligned with Azure Well-Architected Framework. Cloud-scale analytics

gives enterprises an 80 percent prescribed viewpoint, and the remaining 20 percent is customizable.

Cloud-scale analytics offers enterprises a strategic design path toward data mesh, and it can be used to quickly set up such an architecture. It offers a blueprint, including core data platform services for data management.

At the highest level, cloud-scale analytics use a data management capability, which is enabled through the data management landing zone. This zone is responsible for the federated data governance of an organization of the (self-service) platform, and the data domains that drive business value through data products. The benefit of this approach is that it removes technical complexity, while adhering to the same standards. It ensures that there's no proliferation of technology. It also allows enterprises to start modular, with a small footprint, and then grow over time.

The data management landing zone, as you can see in the following diagram, surrounds all data domains. It glues all domains together and provides the oversight that Woodgrove Bank is looking for.

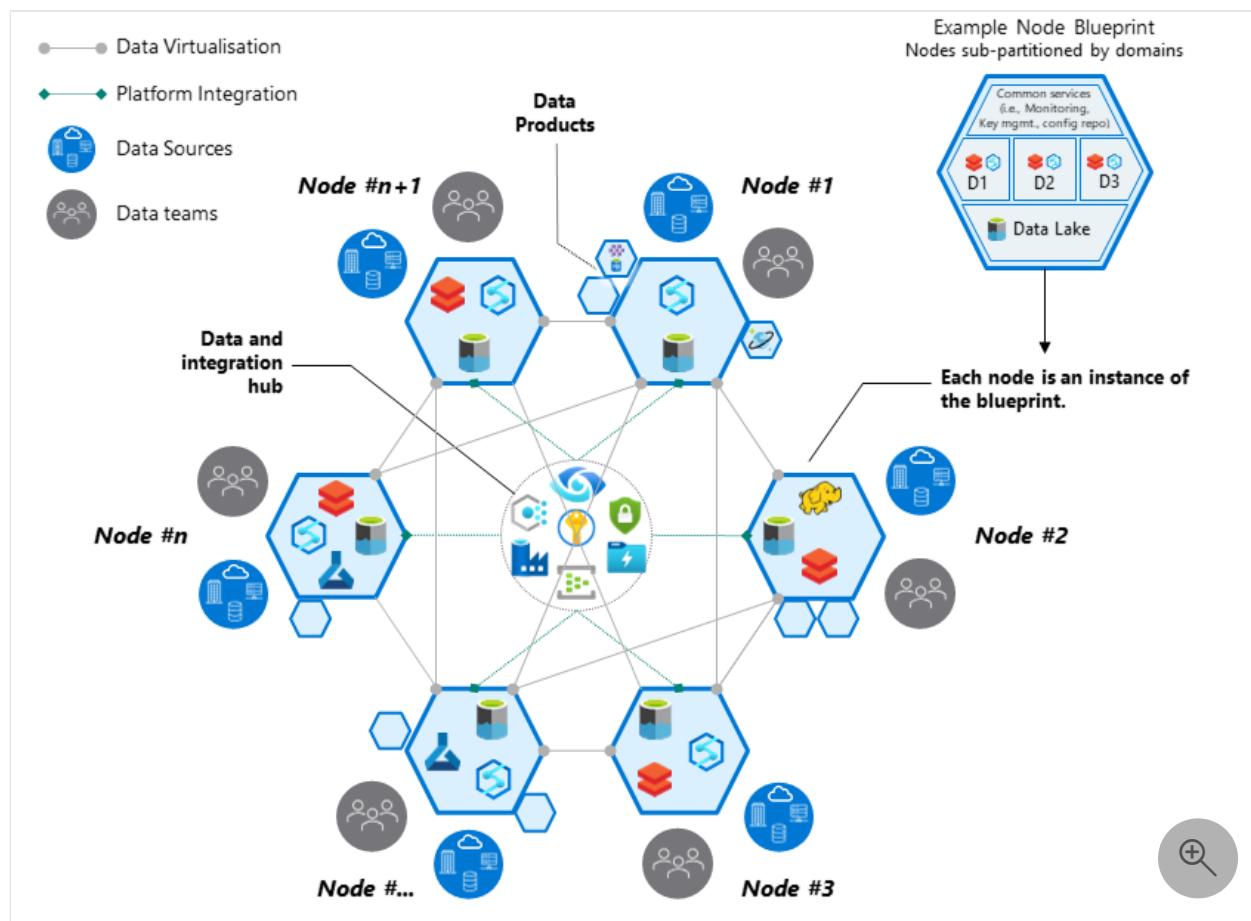


Cloud-scale analytics also advocates the application of consistent governance that uses a common architecture when data products are distributed. The framework allows direct communication between domains. It stays in control by placing an emphasis on central cataloging and classification to protect data and allow groups to discover data. It places an umbrella on top of your data estate.

Data domains

When you use cloud-scale analytics as a strategic path, you need to think of the decomposition of your architecture and the resulting granularity. Data mesh decomposes data by not following the borders of technologies. Instead, it applies the principles of domain-driven design (DDD), an approach to software development that involves complex systems for larger organizations. DDD is popular because of its effect on modern software and application development practices, such as microservices.

One of the patterns from domain-driven design is known as *bounded context*. Bounded contexts are used to set the logical boundaries of a domain's solution space to better manage complexity. It's important that teams understand which aspects, including data, they can change and which are shared dependencies to coordinate with others. Data mesh embraces bounded context. It uses this pattern to describe how organizations can coordinate around data domains and focus on delivering data as a product. Each data domain owns and operates multiple data products with its own technology stack, which is independent from the others.



Data products

When you zoom in on the inner architecture of such a data domain, you expect to find data products within it.

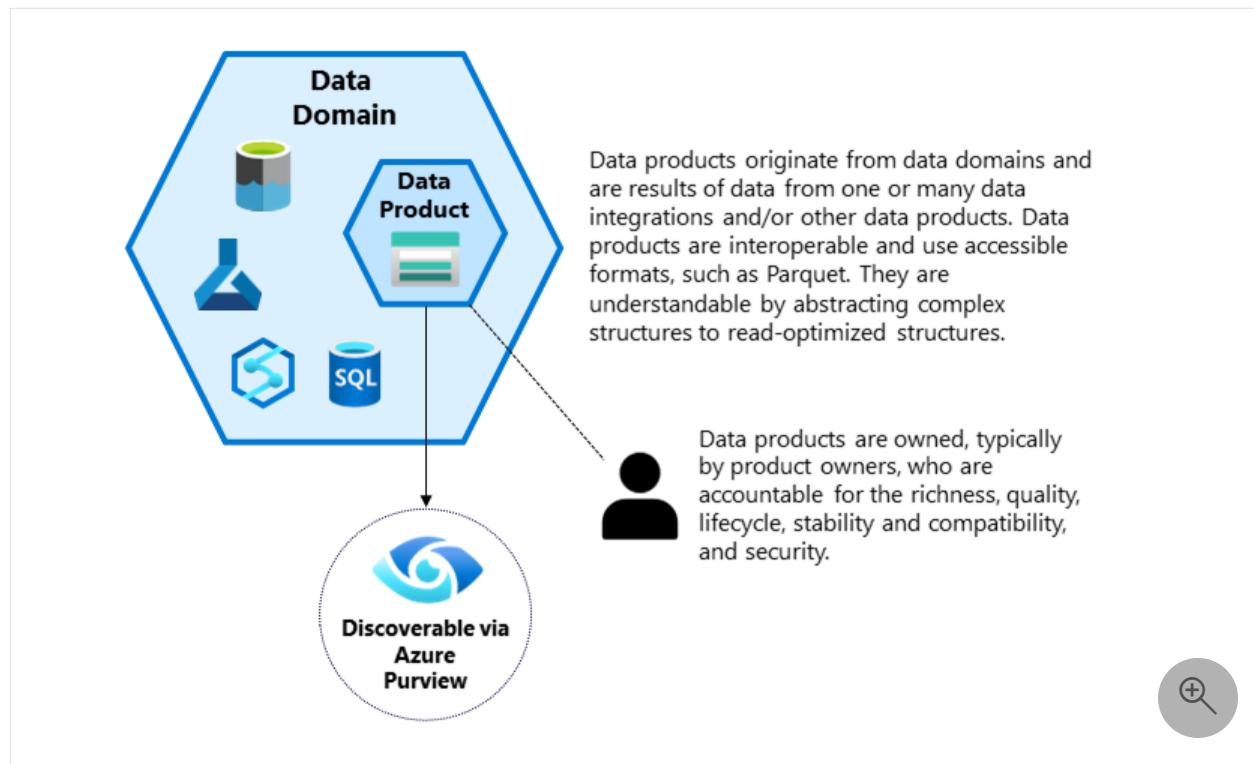
Data products fulfill a specific need within businesses that use data. Data products manage, organize, and make sense of the data across domains and then present the insights they've gained. A data product is a result of data from one or many data integrations or other data products. Data products are closely aligned with data domains and inherit the same constructed, formalized language. It's agreed upon by stakeholders and designers, and it serves the needs of the design. Each domain, which generates data, is responsible for making these data products available to the other domains.

To help quickly deliver data products, cloud-scale analytics offers templates for data distribution and integration patterns. The framework provides data batch, streaming, and analytics to address the needs of diverse consumers.

One great thing about cloud-scale analytics is how domains and data products are organized. Each data domain aligns with one data landing zone, which is a logical construct and a unit of scale in cloud-scale analytics architecture. It enables data retention and execution of data workloads, which generates insights and value. Each data product aligns with one resource group within the data landing zone, and all data landing zones and management zones align with subscriptions. This approach eases implementation and management.

All cloud-scale analytics templates inherit the same set of policies from the data management landing zone. The templates automatically deliver necessary metadata for data discoverability, governance, security, cost management, and operational excellence. You can quickly onboard new data domains without the need of complex onboarding, integrating, and testing.

The following diagram illustrates what a data product might look like:



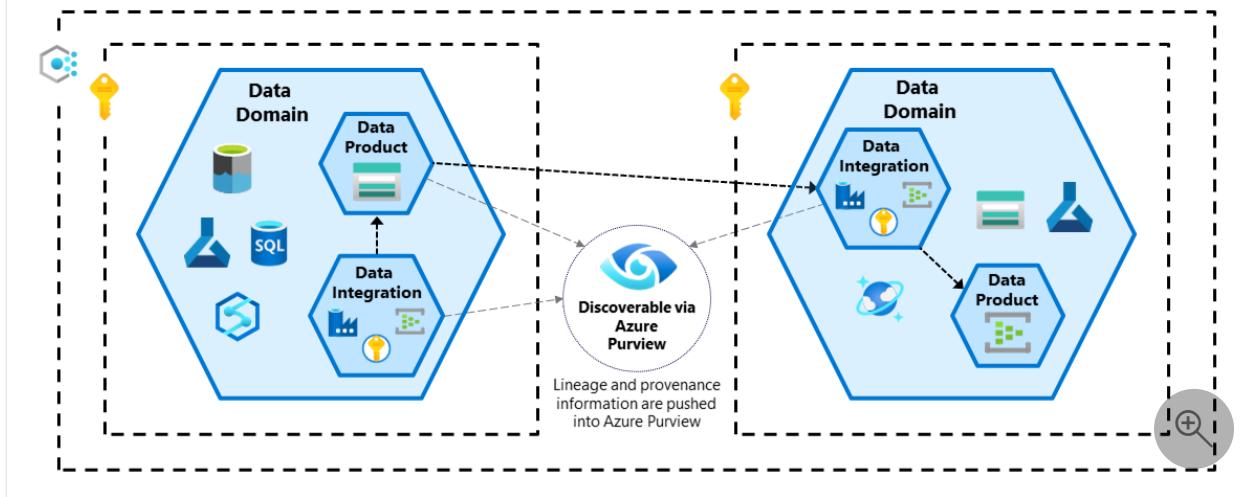
A pragmatic approach to building data products is to either align with the source, where the data originates, or with the consuming use case. In both cases, you need to provide an abstract view of the underlying (complex) application data model. You must try to hide the technical details and optimize for intensive data consumption. An Azure Synapse view or Parquet file, which logically groups data together, is an example of how a data product can be shared across various data domains.

Next, you need to work on data discoverability, provenance, usage, and lineage. A proven approach is to use a data governance service, like Azure Purview, to register all data. Data integration in cloud-scale analytics perfectly connects the dots because it allows building these data products as it simultaneously performs metadata registration.

By aligning data domains and Azure Purview collections, you automatically capture all data origin, lineage, data quality details, and consumption information from the individual domains. With this approach, you can connect multiple data domains and products to a centralized governance solution, which stores all the metadata from each environment. The benefit is that it centrally integrates all the metadata and makes it easily accessible to various consumers. You can extend this architecture to register new data products.

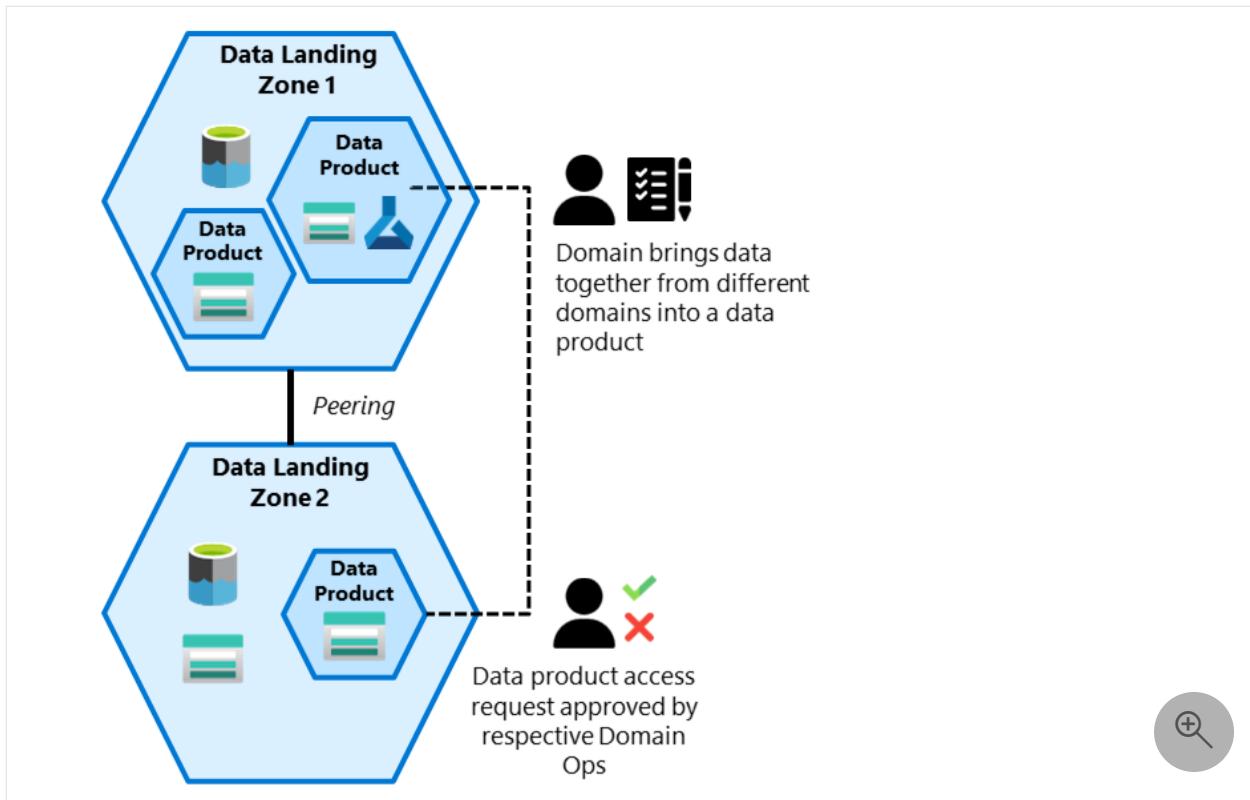
The following diagram illustrates a cross-domain data mesh architecture that uses cloud-scale analytics.

Data integration is used for data ingestion and data distribution between data domains. The paradigm shift is to do a shift left. Thus, data domains first build a 'data product' in the language of the domain. The underlying operational systems in this approach are abstracted away. Consuming domains never access these operational systems directly. It also means you can apply CQRS for consumers who don't require most accurate data.



The network design allows data products to be shared across domains by using minimal cost and eliminating a single point of failure and bandwidth limitations. To help ensure security, you can use the Microsoft *Zero Trust* security model. Cloud-scale analytics proposes the use of network isolation through private endpoints and private network communication, an identity-driven data access model that uses MIs, UMIs, and nested security groups, following the *principle of least privilege*.

You can use managed identities to ensure that a least privilege access model is followed. Applications and services in this model have limited access to data products. Azure policies, with the upcoming data policies, are used to enable self-service and enforce compliant resources within all data products, at scale. With this design, you can have uniform data access, while staying fully in control via centralized data governance and auditing.



Evolve toward the future

Cloud-scale analytics is designed with data mesh in mind. Cloud-scale analytics provides a proven approach by which organizations can share data across many data domains. This framework allows domains to have the autonomy to make choices and it governs the architecture by ring-fencing it with data management services.

When you're implementing data mesh, logically group and organize your domains. This approach requires an enterprise view and is likely a cultural shift for your organization. The shift requires you to federate data ownership among data domains and owners who are accountable for providing their data as products. It also requires teams to conform to centralized capabilities that are offered by the data management landing zone. This new approach might require individual teams to give up their current mandates, which are likely to generate resistance. You might have to make certain political choices and strike a balance between centralized and decentralized approaches.

You can scale a data mesh architecture by adding more landing zones to the architecture for individual domains. These landing zones use virtual network peering to connect to the data management landing zone and all other landing zones. This pattern allows you to share data products and resources across zones. When you split into separate zones, you can spread workloads across Azure subscriptions and resources. This approach allows you to implement the data mesh organically.

Learn more

Microsoft resources:

- [Data management landing zone template ↗](#)
- [Data landing zone template ↗](#)

Article by data mesh founder Zhamak Dehghani:

- [How to move beyond a monolithic data lake to a distributed data mesh ↗](#)

Azure security baseline for Data Factory

Article • 09/20/2023

This security baseline applies guidance from the [Microsoft cloud security benchmark version 1.0](#) to Data Factory. The Microsoft cloud security benchmark provides recommendations on how you can secure your cloud solutions on Azure. The content is grouped by the security controls defined by the Microsoft cloud security benchmark and the related guidance applicable to Data Factory.

You can monitor this security baseline and its recommendations using Microsoft Defender for Cloud. Azure Policy definitions will be listed in the Regulatory Compliance section of the Microsoft Defender for Cloud portal page.

When a feature has relevant Azure Policy Definitions, they are listed in this baseline to help you measure compliance with the Microsoft cloud security benchmark controls and recommendations. Some recommendations may require a paid Microsoft Defender plan to enable certain security scenarios.

ⓘ Note

Features not applicable to Data Factory have been excluded. To see how Data Factory completely maps to the Microsoft cloud security benchmark, see the [full Data Factory security baseline mapping file](#).

Security profile

The security profile summarizes high-impact behaviors of Data Factory, which may result in increased security considerations.

Service Behavior Attribute	Value
Product Category	Analytics, Integration
Customer can access HOST / OS	No Access
Service can be deployed into customer's virtual network	True
Stores customer content at rest	True

Network security

For more information, see the [Microsoft cloud security benchmark: Network security](#).

NS-1: Establish network segmentation boundaries

Features

Virtual Network Integration

Description: Service supports deployment into customer's private Virtual Network (VNet). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Azure-SSIS Integration runtime supports virtual network injection on the customer's virtual network. When creating an Azure-SSIS Integration Runtime (IR), you can join it with a virtual network. It will allow Azure Data Factory to create certain network resources, like an NSG and a load balancer. You can also provide your own static public IP address or have Azure Data Factory create one for you. Self-hosted integration runtime (IR) can be set up on IaaS VM within the customer's virtual network. The network traffic is also governed by the customer's NSG and firewall settings.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Join Azure-SSIS integration runtime to a virtual network](#)

Network Security Group Support

Description: Service network traffic respects Network Security Groups rule assignment on its subnets. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Azure-SSIS Integration runtime supports virtual network injection on the customer's virtual network. It abides by all NSG and firewall rules set by the customer in their virtual network. When creating an Azure-SSIS Integration Runtime (IR), you can join it with a virtual network. It will allow Azure Data Factory to create certain network

resources, like an NSG and a load balancer. You can also provide your own static public IP address or have Azure Data Factory create one for you. On the NSG that is automatically created by Azure Data Factory, Port 3389 is open to all traffic by default. Lock the port down to make sure that only your administrators have access.

Self-hosted integration runtime (IR) can be set up on IaaS VM within the customer's virtual network. The network traffic is also governed by the customer's NSG and firewall settings.

Based on your applications and enterprise segmentation strategy, restrict or allow traffic between internal resources based on your NSG rules. For specific, well-defined applications like a three-tier app, it can be a highly secure deny-by-default.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Join Azure-SSIS integration runtime to a virtual network](#)

NS-2: Secure cloud services with network controls

Features

Azure Private Link

Description: Service native IP filtering capability for filtering network traffic (not to be confused with NSG or Azure Firewall). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Additional Guidance: You can configure private endpoints in the Azure Data Factory Managed Virtual Network to connect to data stores privately.

Data Factory doesn't provide the capability to configure Virtual Network service endpoints.

When you create an Azure-SSIS Integration Runtime (IR), you can join it with a virtual network. It allows Azure Data Factory to create certain network resources, like an NSG and a load balancer. You can also provide your own static public IP address or have Azure Data Factory create one for you. On the NSG that is automatically created by Azure Data Factory, Port 3389 is open to all traffic by default. Lock the port down to

ensure that only your administrators have access. You can deploy Self-Hosted IRs on an on-premises machine or Azure VM inside a virtual network. Make sure that your virtual network subnet deployment has an NSG configured to allow only administrative access. Azure-SSIS IR disallows port 3389 outbound by default at the Windows Firewall Rule on each IR node for protection. You can secure your virtual network-configured resources by associating an NSG with the subnet and setting strict rules.

Reference: [Azure Private Link for Azure Data Factory](#)

Disable Public Network Access

Description: Service supports disabling public network access either through using service-level IP ACL filtering rule (not NSG or Azure Firewall) or using a 'Disable Public Network Access' toggle switch. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Disabling public network access is applicable only to Self-Hosted Integration Runtime (SHIR) and not Azure IR or SSIS IR. With SHIR, enabling private link data factory doesn't explicitly block the public access but customer can block public access manually.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Azure Private Link for Azure Data Factory](#)

Identity management

For more information, see the [Microsoft cloud security benchmark: Identity management](#).

IM-1: Use centralized identity and authentication system

Features

Azure AD Authentication Required for Data Plane Access

Description: Service supports using Azure AD authentication for data plane access.

[Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Data Factory can natively authenticate to the Azure services and resources that support Azure AD authentication.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Managed identity for Azure Data Factory](#)

Local Authentication Methods for Data Plane Access

Description: Local authentications methods supported for data plane access, such as a local username and password. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: You can use Windows authentication to access data sources from SSIS packages running on Azure-SSIS Integration Runtime (IR). Your data stores can be on premises, hosted on Azure Virtual Machines (VMs), or running in Azure as managed services. However, we recommend avoiding the usage of local authentication and using Azure AD wherever possible. Avoid the usage of local authentication methods or accounts, these should be disabled wherever possible. Instead use Azure AD to authenticate where possible.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Access data stores and file shares with Windows authentication from SSIS packages in Azure](#)

IM-3: Manage application identities securely and automatically

Features

Managed Identities

Description: Data plane actions support authentication using managed identities. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: By default, when creating data factory through Azure portal or PowerShell, managed identity will be created automatically. Using SDK or REST API, managed identity will be created only if user specifies “identity” keyword explicitly.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Managed identity for Azure Data Factory and Azure Synapse](#)

Service Principals

Description: Data plane supports authentication using service principals. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Data Factory allows you to use Managed identities, Service Principles to authenticate against data stores and compute that support AAD authentication.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Copy and transform data in Azure Data Lake Storage Gen2 using Azure Data Factory or Azure Synapse Analytics](#)

IM-7: Restrict resource access based on conditions

Features

Conditional Access for Data Plane

Description: Data plane access can be controlled using Azure AD Conditional Access Policies. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Conditional Access: Cloud apps, actions, and authentication context](#)

IM-8: Restrict the exposure of credential and secrets

Features

Service Credential and Secrets Support Integration and Storage in Azure Key Vault

Description: Data plane supports native use of Azure Key Vault for credential and secrets store. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: You can store credentials for data stores and computes in an Azure Key Vault. Azure Data Factory retrieves the credentials when executing an activity that uses the data store/compute.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Store credentials in Azure Key Vault](#)

Privileged access

For more information, see the [Microsoft cloud security benchmark: Privileged access](#).

PA-1: Separate and limit highly privileged/administrative users

Features

Local Admin Accounts

Description: Service has the concept of a local administrative account. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

PA-7: Follow just enough administration (least privilege) principle

Features

Azure RBAC for Data Plane

Description: Azure Role-Based Access Control (Azure RBAC) can be used to manage access to service's data plane actions. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Feature notes: Data Factory integrates with Azure RBAC to manage its resources. With RBAC, you manage Azure resource access through role assignments. You can assign roles to users, groups, service principals, and managed identities. Certain resources have pre-defined, built-in roles. You can inventory or query these roles through tools like Azure CLI, Azure PowerShell, or the Azure portal.

Limit the privileges you assign to resources through Azure RBAC to what the roles require. This practice complements the just-in-time (JIT) approach of Azure AD PIM. Review roles and assignments periodically.

Use built-in roles to give permissions. Only create custom roles when required.

You can create a custom role in Azure AD with more restrictive access to Data Factory.

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Roles and permissions for Azure Data Factory](#)

PA-8: Determine access process for cloud provider support

Features

Customer Lockbox

Description: Customer Lockbox can be used for Microsoft support access. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Customer Lockbox for Microsoft Azure](#)

Data protection

For more information, see the [Microsoft cloud security benchmark: Data protection](#).

DP-1: Discover, classify, and label sensitive data

Features

Sensitive Data Discovery and Classification

Description: Tools (such as Azure Purview or Azure Information Protection) can be used for data discovery and classification in the service. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Connect Data Factory to Microsoft Purview](#)

DP-2: Monitor anomalies and threats targeting sensitive data

Features

Data Leakage/Loss Prevention

Description: Service supports DLP solution to monitor sensitive data movement (in customer's content). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

DP-3: Encrypt sensitive data in transit

Features

Data in Transit Encryption

Description: Service supports data in-transit encryption for data plane. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Security considerations for data movement in Azure Data Factory](#)

DP-4: Enable data at rest encryption by default

Features

Data at Rest Encryption Using Platform Keys

Description: Data at-rest encryption using platform keys is supported, any customer content at rest is encrypted with these Microsoft managed keys. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Encrypt Azure Data Factory with customer-managed keys](#)

DP-5: Use customer-managed key option in data at rest encryption when required

Features

Data at Rest Encryption Using CMK

Description: Data at-rest encryption using customer-managed keys is supported for customer content stored by the service. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Encrypt Azure Data Factory with customer-managed keys](#)

DP-6: Use a secure key management process

Features

Key Management in Azure Key Vault

Description: The service supports Azure Key Vault integration for any customer keys, secrets, or certificates. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Store credentials in Azure Key Vault](#)

DP-7: Use a secure certificate management process

Features

Certificate Management in Azure Key Vault

Description: The service supports Azure Key Vault integration for any customer certificates. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Store credentials in Azure Key Vault](#)

Asset management

For more information, see the [Microsoft cloud security benchmark: Asset management](#).

AM-2: Use only approved services

Features

Azure Policy Support

Description: Service configurations can be monitored and enforced via Azure Policy.

[Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Use Azure Policy to audit and restrict which services users can provision in your environment. Use Azure Resource Graph to query for and discover resources within subscriptions. You can also use Azure Monitor to create rules to trigger alerts when they detect an unapproved service.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Azure Policy built-in definitions for Data Factory](#)

Logging and threat detection

For more information, see the [Microsoft cloud security benchmark: Logging and threat detection](#).

LT-1: Enable threat detection capabilities

Features

Microsoft Defender for Service / Product Offering

Description: Service has an offering-specific Microsoft Defender solution to monitor and alert on security issues. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Feature notes: Self-hosted IR (SHIR) running on Azure VMs and containers, uses Defender for establishing the secure configuration.

Configuration Guidance: This feature is not supported to secure this service.

LT-4: Enable logging for security investigation

Features

Azure Resource Logs

Description: Service produces resource logs that can provide enhanced service-specific metrics and logging. The customer can configure these resource logs and send them to their own data sink like a storage account or log analytics workspace. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Feature notes: Activity logs are available automatically. You can use activity logs to find errors when troubleshooting, or to monitor how users in your organization modified resources.

Use Microsoft Defender for Cloud and Azure Policy to enable resource logs and log data collecting.

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Diagnostic settings in Azure Monitor](#)

Backup and recovery

For more information, see the [Microsoft cloud security benchmark: Backup and recovery](#).

BR-1: Ensure regular automated backups

Features

Service Native Backup Capability

Description: Service supports its own native backup capability (if not using Azure Backup). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: To back up all code on Azure Data Factory, use source control functionality in Data Factory.

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Source control in Azure Data Factory](#)

Next steps

- See the [Microsoft cloud security benchmark overview](#)
- Learn more about [Azure security baselines](#)

Azure security baseline for Azure Databricks

Article • 09/20/2023

This security baseline applies guidance from the [Microsoft cloud security benchmark version 1.0](#) to Azure Databricks. The Microsoft cloud security benchmark provides recommendations on how you can secure your cloud solutions on Azure. The content is grouped by the security controls defined by the Microsoft cloud security benchmark and the related guidance applicable to Azure Databricks.

You can monitor this security baseline and its recommendations using Microsoft Defender for Cloud. Azure Policy definitions will be listed in the Regulatory Compliance section of the Microsoft Defender for Cloud portal page.

When a feature has relevant Azure Policy Definitions, they are listed in this baseline to help you measure compliance with the Microsoft cloud security benchmark controls and recommendations. Some recommendations may require a paid Microsoft Defender plan to enable certain security scenarios.

ⓘ Note

Features not applicable to Azure Databricks have been excluded. To see how Azure Databricks completely maps to the Microsoft cloud security benchmark, see the [full Azure Databricks security baseline mapping file](#).

Security profile

The security profile summarizes high-impact behaviors of Azure Databricks, which may result in increased security considerations.

Service Behavior Attribute	Value
Product Category	Analytics, Storage
Customer can access HOST / OS	No Access
Service can be deployed into customer's virtual network	True
Stores customer content at rest	True

Network security

For more information, see the [Microsoft cloud security benchmark: Network security](#).

NS-1: Establish network segmentation boundaries

Features

Virtual Network Integration

Description: Service supports deployment into customer's private Virtual Network (VNet). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: The default deployment of Azure Databricks is a fully managed service on Azure: all data plane resources, including a VNet that all clusters will be associated with, are deployed to a locked resource group. If you require network customization, however, you can deploy Azure Databricks data plane resources in your own virtual network (VNet injection), enabling you to implement custom network configurations. You can apply your own network security group (NSG) with custom rules to specific egress traffic restrictions.

Reference: [Databricks VNET Integration](#)

Network Security Group Support

Description: Service network traffic respects Network Security Groups rule assignment on its subnets. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Use network security groups (NSG) to restrict or monitor traffic by port, protocol, source IP address, or destination IP address. Create NSG rules to restrict your service's open ports (such as preventing management ports from being accessed from untrusted networks). Be aware that by default, NSGs deny all inbound traffic but allow traffic from virtual network and Azure Load Balancers.

Reference: [Network Security Group](#)

NS-2: Secure cloud services with network controls

Features

Azure Private Link

Description: Service native IP filtering capability for filtering network traffic (not to be confused with NSG or Azure Firewall). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

Disable Public Network Access

Description: Service supports disabling public network access either through using service-level IP ACL filtering rule (not NSG or Azure Firewall) or using a 'Disable Public Network Access' toggle switch. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Azure Databricks customers can use the IP access lists feature to define a set of approved IP addresses to prevent access from public IP or unapproved IP addresses.

Reference: [IP Access list in Databricks](#)

Identity management

For more information, see the [Microsoft cloud security benchmark: Identity management](#).

IM-1: Use centralized identity and authentication system

Features

Azure AD Authentication Required for Data Plane Access

Description: Service supports using Azure AD authentication for data plane access.

[Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

IM-3: Manage application identities securely and automatically

Features

Managed Identities

Description: Data plane actions support authentication using managed identities. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Feature notes: Azure Databricks is automatically set up to use Azure Active Directory (Azure AD) single sign-on to authenticate users. Users outside of your organization must complete the invitation process and be added to your Active Directory tenant before they are able to log in to Azure Databricks via single sign-on. You can implement SCIM to automate provisioning and de-provisioning users from workspaces.

[Understand single sign-on for Azure Databricks](#)

[How to use the SCIM APIs for Azure Databricks](#)

Configuration Guidance: This feature is not supported to secure this service.

Service Principals

Description: Data plane supports authentication using service principals. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: For services that don't support managed identities, use Azure Active Directory (Azure AD) to create a service principal with restricted permissions at the resource level. Configure service principals with certificate credentials and fall back to client secrets for authentication.

Reference: [Service principal in Databricks](#)

IM-7: Restrict resource access based on conditions

Features

Conditional Access for Data Plane

Description: Data plane access can be controlled using Azure AD Conditional Access Policies. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Feature notes: Additionally Azure Databricks supports IP access lists to make accessing the web application and the REST API more secure.

[IP access lists in Databricks](#)

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Conditional Access in Databricks](#)

IM-8: Restrict the exposure of credential and secrets

Features

Service Credential and Secrets Support Integration and Storage in Azure Key Vault

Description: Data plane supports native use of Azure Key Vault for credential and secrets store. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Azure Databricks also supports a secret scope stored in (backed by) an encrypted database owned and managed by Azure Databricks.

[Databricks-backed scopes](#)

Configuration Guidance: Ensure that secrets and credentials are stored in secure locations such as Azure Key Vault, instead of embedding them into code or configuration files.

Reference: [Key Vault Integration in Databricks](#)

Privileged access

For more information, see the [Microsoft cloud security benchmark: Privileged access](#).

PA-7: Follow just enough administration (least privilege) principle

Features

Azure RBAC for Data Plane

Description: Azure Role-Based Access Control (Azure RBAC) can be used to manage access to service's data plane actions. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Feature notes: You can use Azure Databricks SCIM APIs to manage users in an Azure Databricks workspace and grant administrative privileges to designated users.

[How to use the SCIM APIs](#)

In Azure Databricks, you can use access control lists (ACLs) to configure permission to access different workspace objects.

Access control in Databricks

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [How to manage access control in Azure Databricks](#)

PA-8: Determine access process for cloud provider support

Features

Customer Lockbox

Description: Customer Lockbox can be used for Microsoft support access. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: In support scenarios where Microsoft needs to access your data, use Customer Lockbox to review, then approve or reject each of Microsoft's data access requests.

Reference: [Customer Lockbox](#)

Data protection

For more information, see the [Microsoft cloud security benchmark: Data protection](#).

DP-3: Encrypt sensitive data in transit

Features

Data in Transit Encryption

Description: Service supports data in-transit encryption for data plane. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: By default, the data exchanged between worker nodes in a cluster is not encrypted. If your environment requires that data be encrypted at all times, you can create an init script that configures your clusters to encrypt traffic between worker nodes.

Configuration Guidance: Enable secure transfer in services where there is a native data in transit encryption feature built in. Enforce HTTPS on any web applications and services and ensure TLS v1.2 or later is used. Legacy versions such as SSL 3.0, TLS v1.0 should be disabled. For remote management of Virtual Machines, use SSH (for Linux) or RDP/TLS (for Windows) instead of an unencrypted protocol.

Reference: [Data in transit Encryption for Databricks](#)

DP-4: Enable data at rest encryption by default

Features

Data at Rest Encryption Using Platform Keys

Description: Data at-rest encryption using platform keys is supported, any customer content at rest is encrypted with these Microsoft managed keys. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Data at rest encryption using platform managed keys in Databricks](#)

DP-5: Use customer-managed key option in data at rest encryption when required

Features

Data at Rest Encryption Using CMK

Description: Data at-rest encryption using customer-managed keys is supported for customer content stored by the service. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Azure Databricks has two customer-managed key features for different types of data.

[Customer-managed keys for encryption](#)

Configuration Guidance: If required for regulatory compliance, define the use case and service scope where encryption using customer-managed keys are needed. Enable and implement data at rest encryption using customer-managed key for those services.

Reference: [Data at Rest Encryption Using CMK in Databricks](#)

DP-6: Use a secure key management process

Features

Key Management in Azure Key Vault

Description: The service supports Azure Key Vault integration for any customer keys, secrets, or certificates. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: Note, you cannot use an Azure Databricks personal access token or an Azure AD application token that belongs to a service principal.

[Avoid personal access token](#)

Configuration Guidance: Use Azure Key Vault to create and control the life cycle of your encryption keys, including key generation, distribution, and storage. Rotate and revoke your keys in Azure Key Vault and your service based on a defined schedule or when there is a key retirement or compromise. When there is a need to use customer-managed key (CMK) in the workload, service, or application level, ensure you follow the best practices for key management: Use a key hierarchy to generate a separate data encryption key (DEK) with your key encryption key (KEK) in your key vault. Ensure keys are registered with Azure Key Vault and referenced via key IDs from the service or application. If you need to bring your own key (BYOK) to the service (such as importing

HSM-protected keys from your on-premises HSMs into Azure Key Vault), follow recommended guidelines to perform initial key generation and key transfer.

Reference: [Key management in Databricks](#)

Asset management

For more information, see the [Microsoft cloud security benchmark: Asset management](#).

AM-2: Use only approved services

Features

Azure Policy Support

Description: Service configurations can be monitored and enforced via Azure Policy.
[Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Use Microsoft Defender for Cloud to configure Azure Policy to audit and enforce configurations of your Azure resources. Use Azure Monitor to create alerts when there is a configuration deviation detected on the resources. Use Azure Policy [deny] and [deploy if not exists] effects to enforce secure configuration across Azure resources.

Reference: [Databricks Azure Policy](#)

Logging and threat detection

For more information, see the [Microsoft cloud security benchmark: Logging and threat detection](#).

LT-1: Enable threat detection capabilities

Features

Microsoft Defender for Service / Product Offering

Description: Service has an offering-specific Microsoft Defender solution to monitor and alert on security issues. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

LT-4: Enable logging for security investigation

Features

Azure Resource Logs

Description: Service produces resource logs that can provide enhanced service-specific metrics and logging. The customer can configure these resource logs and send them to their own data sink like a storage account or log analytics workspace. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: For audit logging, Azure Databricks provides comprehensive end-to-end diagnostic logs of activities performed by Azure Databricks users, allowing your enterprise to monitor detailed Azure Databricks usage patterns.

Note: that Azure Databricks diagnostic logs require the Azure Databricks Premium Plan.

[How to enable Diagnostic Settings for Azure Activity Log](#)

[How to enable Diagnostic Settings for Azure Databricks](#)

Reference: [Resource logs in Databricks](#)

Posture and vulnerability management

For more information, see the [Microsoft cloud security benchmark: Posture and vulnerability management](#).

PV-3: Define and establish secure configurations for compute resources

Features

Other guidance for PV-3

When you create an Azure Databricks cluster, it spins up base VM images. User code is run within containers that are deployed on the VMs. Implement a third-party vulnerability management solution. If you have a vulnerability management platform subscription, you may use Azure Databricks initialization scripts, running in the containers on each of the nodes, to install vulnerability assessment agents on your Azure Databricks cluster nodes, and manage the nodes through the respective portal. Note that every third-party solution works differently.

[Databricks cluster node initialization scripts](#)

Backup and recovery

For more information, see the [Microsoft cloud security benchmark: Backup and recovery](#).

BR-1: Ensure regular automated backups

Features

Azure Backup

Description: The service can be backed up by the Azure Backup service. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

Service Native Backup Capability

Description: Service supports its own native backup capability (if not using Azure Backup). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: For your Azure Databricks data sources, ensure you have configured an appropriate level of data redundancy for your use case. For example, if using an Azure Storage account for your Azure Databricks data store, choose the appropriate redundancy option (LRS, ZRS, GRS, RA-GRS).

[Data sources for Azure Databricks](#)

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Regional disaster recovery for Azure Databricks clusters](#)

Next steps

- See the [Microsoft cloud security benchmark overview](#)
- Learn more about [Azure security baselines](#)

Azure security baseline for Azure Purview

Article • 09/20/2023

This security baseline applies guidance from the [Microsoft cloud security benchmark version 1.0](#) to Azure Purview. The Microsoft cloud security benchmark provides recommendations on how you can secure your cloud solutions on Azure. The content is grouped by the security controls defined by the Microsoft cloud security benchmark and the related guidance applicable to Azure Purview.

You can monitor this security baseline and its recommendations using Microsoft Defender for Cloud. Azure Policy definitions will be listed in the Regulatory Compliance section of the Microsoft Defender for Cloud portal page.

When a feature has relevant Azure Policy Definitions, they are listed in this baseline to help you measure compliance with the Microsoft cloud security benchmark controls and recommendations. Some recommendations may require a paid Microsoft Defender plan to enable certain security scenarios.

ⓘ Note

Features not applicable to Azure Purview have been excluded. To see how Azure Purview completely maps to the Microsoft cloud security benchmark, see the [full Azure Purview security baseline mapping file](#).

Security profile

The security profile summarizes high-impact behaviors of Azure Purview, which may result in increased security considerations.

Service Behavior Attribute	Value
Product Category	MGMT/Governance
Customer can access HOST / OS	No Access
Service can be deployed into customer's virtual network	True
Stores customer content at rest	True

Network security

For more information, see the [Microsoft cloud security benchmark: Network security](#).

NS-1: Establish network segmentation boundaries

Features

Virtual Network Integration

Description: Service supports deployment into customer's private Virtual Network (VNet). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Deploy the service into a virtual network. Assign private IPs to the resource (where applicable) unless there is a strong reason to assign public IPs directly to the resource.

Reference: [Microsoft Purview network architecture and best practices](#)

Network Security Group Support

Description: Service network traffic respects Network Security Groups rule assignment on its subnets. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Feature notes: This is only possible when private end points are implemented.

Configuration Guidance: Use network security groups (NSG) to restrict or monitor traffic by port, protocol, source IP address, or destination IP address. Create NSG rules to restrict your service's open ports (such as preventing management ports from being accessed from untrusted networks). Be aware that by default, NSGs deny all inbound traffic but allow traffic from virtual network and Azure Load Balancers.

Reference: [Connect to your Microsoft Purview and scan data sources privately and securely](#)

NS-2: Secure cloud services with network controls

Features

Azure Private Link

Description: Service native IP filtering capability for filtering network traffic (not to be confused with NSG or Azure Firewall). [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Deploy private endpoints for all Azure resources that support the Private Link feature, to establish a private access point for the resources.

Reference: [Connect privately and securely to your Microsoft Purview account](#)

Disable Public Network Access

Description: Service supports disabling public network access either through using service-level IP ACL filtering rule (not NSG or Azure Firewall) or using a 'Disable Public Network Access' toggle switch. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Disable public network access either using the service-level IP ACL filtering rule or a toggling switch for public network access.

Reference: [Firewalls to restrict public access](#)

Identity management

For more information, see the [Microsoft cloud security benchmark: Identity management](#).

IM-1: Use centralized identity and authentication system

Features

Azure AD Authentication Required for Data Plane Access

Description: Service supports using Azure AD authentication for data plane access.

[Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Use Azure Active Directory (Azure AD) as the default authentication method to control your data plane access.

Reference: [Tutorial: Use the REST APIs](#)

Local Authentication Methods for Data Plane Access

Description: Local authentications methods supported for data plane access, such as a local username and password. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

IM-3: Manage application identities securely and automatically

Features

Managed Identities

Description: Data plane actions support authentication using managed identities. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Use Azure managed identities instead of service principals when possible, which can authenticate to Azure services and resources that support Azure Active Directory (Azure AD) authentication. Managed identity credentials are fully

managed, rotated, and protected by the platform, avoiding hard-coded credentials in source code or configuration files.

Reference: [Create a user-assigned managed identity](#)

Service Principals

Description: Data plane supports authentication using service principals. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: There is no current Microsoft guidance for this feature configuration. Please review and determine if your organization wants to configure this security feature.

Reference: [Creating a service principal](#)

IM-7: Restrict resource access based on conditions

Features

Conditional Access for Data Plane

Description: Data plane access can be controlled using Azure AD Conditional Access Policies. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Define the applicable conditions and criteria for Azure Active Directory (Azure AD) conditional access in the workload. Consider common use cases such as blocking or granting access from specific locations, blocking risky sign-in behavior, or requiring organization-managed devices for specific applications.

Reference: [Conditional Access with Microsoft Purview](#)

IM-8: Restrict the exposure of credential and secrets

Features

Service Credential and Secrets Support Integration and Storage in Azure Key Vault

Description: Data plane supports native use of Azure Key Vault for credential and secrets store. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Ensure that secrets and credentials are stored in secure locations such as Azure Key Vault, instead of embedding them into code or configuration files.

Reference: [Credentials for source authentication in Microsoft Purview](#)

Privileged access

For more information, see the [Microsoft cloud security benchmark: Privileged access](#).

PA-1: Separate and limit highly privileged/administrative users

Features

Local Admin Accounts

Description: Service has the concept of a local administrative account. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

PA-7: Follow just enough administration (least privilege) principle

Features

Azure RBAC for Data Plane

Description: Azure Role-Based Access Control (Azure RBAC) can be used to manage access to service's data plane actions. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Use Azure role-based access control (Azure RBAC) to manage Azure resource access through built-in role assignments. Azure RBAC roles can be assigned to users, groups, service principals, and managed identities.

Reference: [Access control in the Microsoft Purview governance portal](#)

PA-8: Determine access process for cloud provider support

Features

Customer Lockbox

Description: Customer Lockbox can be used for Microsoft support access. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

Data protection

For more information, see the [Microsoft cloud security benchmark: Data protection](#).

DP-1: Discover, classify, and label sensitive data

Features

Sensitive Data Discovery and Classification

Description: Tools (such as Azure Purview or Azure Information Protection) can be used for data discovery and classification in the service. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Feature notes: The Microsoft Purview solution is the tool used for sensitive data recovery and classification.

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [What's available in the Microsoft Purview governance portal?](#)

DP-2: Monitor anomalies and threats targeting sensitive data

Features

Data Leakage/Loss Prevention

Description: Service supports DLP solution to monitor sensitive data movement (in customer's content). [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Learn about data loss prevention](#)

DP-3: Encrypt sensitive data in transit

Features

Data in Transit Encryption

Description: Service supports data in-transit encryption for data plane. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Transport Layer Security \(Encryption-in-transit\)](#)

DP-4: Enable data at rest encryption by default

Features

Data at Rest Encryption Using Platform Keys

Description: Data at-rest encryption using platform keys is supported, any customer content at rest is encrypted with these Microsoft managed keys. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	True	Microsoft

Configuration Guidance: No additional configurations are required as this is enabled on a default deployment.

Reference: [Transparent data encryption \(Encryption-at-rest\)](#)

DP-5: Use customer-managed key option in data at rest encryption when required

Features

Data at Rest Encryption Using CMK

Description: Data at-rest encryption using customer-managed keys is supported for customer content stored by the service. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

DP-6: Use a secure key management process

Features

Key Management in Azure Key Vault

Description: The service supports Azure Key Vault integration for any customer keys, secrets, or certificates. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Use Azure Key Vault to create and control the life cycle of your encryption keys, including key generation, distribution, and storage. Rotate and revoke your keys in Azure Key Vault and your service based on a defined schedule or when there is a key retirement or compromise. When there is a need to use customer-managed key (CMK) in the workload, service, or application level, ensure you follow the best practices for key management: Use a key hierarchy to generate a separate data encryption key (DEK) with your key encryption key (KEK) in your key vault. Ensure keys are registered with Azure Key Vault and referenced via key IDs from the service or application. If you need to bring your own key (BYOK) to the service (such as importing HSM-protected keys from your on-premises HSMs into Azure Key Vault), follow recommended guidelines to perform initial key generation and key transfer.

Reference: [Credentials for source authentication in Purview](#)

DP-7: Use a secure certificate management process

Features

Certificate Management in Azure Key Vault

Description: The service supports Azure Key Vault integration for any customer certificates. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

Asset management

For more information, see the [Microsoft cloud security benchmark: Asset management](#).

AM-2: Use only approved services

Features

Azure Policy Support

Description: Service configurations can be monitored and enforced via Azure Policy.

[Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Use Microsoft Defender for Cloud to configure Azure Policy to audit and enforce configurations of your Azure resources. Use Azure Monitor to create alerts when there is a configuration deviation detected on the resources. Use Azure Policy [deny] and [deploy if not exists] effects to enforce secure configuration across Azure resources.

Reference: [Azure Policy built-in policy definitions](#)

Logging and threat detection

For more information, see the [Microsoft cloud security benchmark: Logging and threat detection](#).

LT-1: Enable threat detection capabilities

Features

Microsoft Defender for Service / Product Offering

Description: Service has an offering-specific Microsoft Defender solution to monitor and alert on security issues. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

LT-4: Enable logging for security investigation

Features

Azure Resource Logs

Description: Service produces resource logs that can provide enhanced service-specific metrics and logging. The customer can configure these resource logs and send them to their own data sink like a storage account or log analytics workspace. [Learn more](#).

Supported	Enabled By Default	Configuration Responsibility
True	False	Customer

Configuration Guidance: Enable resource logs for the service. For example, Key Vault supports additional resource logs for actions that get a secret from a key vault or and Azure SQL has resource logs that track requests to a database. The content of resource logs varies by the Azure service and resource type.

Reference: [Microsoft Purview Metrics in Azure monitor](#)

Backup and recovery

For more information, see the [Microsoft cloud security benchmark: Backup and recovery](#).

BR-1: Ensure regular automated backups

Features

Azure Backup

Description: The service can be backed up by the Azure Backup service. [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

Service Native Backup Capability

Description: Service supports its own native backup capability (if not using Azure Backup). [Learn more.](#)

Supported	Enabled By Default	Configuration Responsibility
False	Not Applicable	Not Applicable

Configuration Guidance: This feature is not supported to secure this service.

Next steps

- See the [Microsoft cloud security benchmark overview](#)
- Learn more about [Azure security baselines](#)

Monitoring Azure Databricks

Article • 03/27/2023

Azure Databricks is a fast, powerful Apache Spark[↗]-based analytics service that makes it easy to rapidly develop and deploy big data analytics and artificial intelligence (AI) solutions. Many users take advantage of the simplicity of notebooks in their Azure Databricks solutions. For users that require more robust computing options, Azure Databricks supports the distributed execution of custom application code.

Monitoring is a critical part of any production-level solution, and Azure Databricks offers robust functionality for monitoring custom application metrics, streaming query events, and application log messages. Azure Databricks can send this monitoring data to different logging services.

The following articles show how to send monitoring data from Azure Databricks to [Azure Monitor](#), the monitoring data platform for Azure.

- [Send Azure Databricks application logs to Azure Monitor](#)
- [Use dashboards to visualize Azure Databricks metrics](#)
- [Troubleshoot performance bottlenecks](#)

The code library that accompanies these articles extends the core monitoring functionality of Azure Databricks to send Spark metrics, events, and logging information to Azure Monitor.

The audience for these articles and the accompanying code library are Apache Spark and Azure Databricks solution developers. The code must be built into Java Archive (JAR) files and then deployed to an Azure Databricks cluster. The code is a combination of [Scala[↗]](#) and Java, with a corresponding set of [Maven[↗]](#) project object model (POM) files to build the output JAR files. Understanding of Java, Scala, and Maven are recommended as prerequisites.

Next steps

Start by building the code library and deploying it to your Azure Databricks cluster.

[Send Azure Databricks application logs to Azure Monitor](#)

Related resources

- [Modern analytics architecture with Azure Databricks](#)

- Ingestion, ETL, and stream processing pipelines with Azure Databricks
- Data science and machine learning with Azure Databricks
- Orchestrate MLOps by using Azure Databricks

Send Azure Databricks application logs to Azure Monitor

Article • 05/25/2023

ⓘ Note

This article relies on an open source library hosted on GitHub at:

<https://github.com/mspnp/spark-monitoring> ↗.

The original library supports Azure Databricks Runtimes 10.x (Spark 3.2.x) and earlier.

Databricks has contributed an updated version to support Azure Databricks Runtimes 11.0 (Spark 3.3.x) and above on the `l4jv2` branch at:

<https://github.com/mspnp/spark-monitoring/tree/l4jv2> ↗.

Please note that the 11.0 release is not backwards compatible due to the different logging systems used in the Databricks Runtimes. Be sure to use the correct build for your Databricks Runtime. The library and GitHub repository are in maintenance mode. There are no plans for further releases, and issue support will be best-effort only. For any additional questions regarding the library or the roadmap for monitoring and logging of your Azure Databricks environments, please contact azure-spark-monitoring-help@databricks.com.

This article shows how to send application logs and metrics from Azure Databricks to a [Log Analytics workspace](#). It uses the [Azure Databricks Monitoring Library](#) ↗, which is available on GitHub.

Prerequisites

Configure your Azure Databricks cluster to use the monitoring library, as described in the [GitHub readme](#) ↗.

ⓘ Note

The monitoring library streams Apache Spark level events and Spark Structured Streaming metrics from your jobs to Azure Monitor. You don't need to make any changes to your application code for these events and metrics.

Send application metrics using Dropwizard

Spark uses a configurable metrics system based on the Dropwizard Metrics Library. For more information, see [Metrics](#) in the Spark documentation.

To send application metrics from Azure Databricks application code to Azure Monitor, follow these steps:

1. Build the `spark-listeners-loganalytics-1.0-SNAPSHOT.jar` JAR file as described in the [GitHub readme](#).
2. Create Dropwizard [gauges or counters](#) in your application code. You can use the `UserMetricsSystem` class defined in the monitoring library. The following example creates a counter named `counter1`.

Scala

```
import org.apache.spark.metrics.UserMetricsSystems
import org.apache.spark.sql.SparkSession

object StreamingQueryListenerSampleJob  {

  private final val METRICS_NAMESPACE = "samplejob"
  private final val COUNTER_NAME = "counter1"

  def main(args: Array[String]): Unit = {

    val spark = SparkSession
      .builder
      .getOrCreate

    val driverMetricsSystem = UserMetricsSystems
      .getMetricSystem(METRICS_NAMESPACE, builder => {
        builder.registerCounter(COUNTER_NAME)
      })

    driverMetricsSystem.counter(COUNTER_NAME).inc(5)
  }
}
```

The monitoring library includes a [sample application](#) that demonstrates how to use the `UserMetricsSystem` class.

Send application logs using Log4j

To send your Azure Databricks application logs to Azure Log Analytics using the [Log4j appender](#) in the library, follow these steps:

1. Build the **spark-listeners-1.0-SNAPSHOT.jar** and the **spark-listeners-loganalytics-1.0-SNAPSHOT.jar** JAR file as described in the [GitHub readme](#).
2. Create a **log4j.properties** [configuration file](#) for your application. Include the following configuration properties. Substitute your application package name and log level where indicated:

YAML

```
log4j.appender.A1=com.microsoft.pnp.logging.loganalytics.LogAnalyticsAppender
log4j.appender.A1.layout=com.microsoft.pnp.logging.JSONLayout
log4j.appender.A1.layout.LocationInfo=false
log4j.additivity.<your application package name>=false
log4j.logger.<your application package name>=<log level>, A1
```

You can find a sample configuration file [here](#).

3. In your application code, include the **spark-listeners-loganalytics** project, and import `com.microsoft.pnp.logging.Log4jConfiguration` to your application code.

Scala

```
import com.microsoft.pnp.logging.Log4jConfiguration
```

4. Configure Log4j using the **log4j.properties** file you created in step 3:

Scala

```
getClass.getResourceAsStream("<path to file in your JAR
file>/log4j.properties") {
    stream => {
        Log4jConfiguration.configure(stream)
    }
}
```

5. Add Apache Spark log messages at the appropriate level in your code as required. For example, use the `logDebug` method to send a debug log message. For more information, see [Logging](#) in the Spark documentation.

Scala

```
logTrace("Trace message")
logDebug("Debug message")
logInfo("Info message")
logWarning("Warning message")
logError("Error message")
```

ⓘ Note

If you're using the library and you have Apache Spark Notebooks, any logs that Spark generates during execution for the notebook automatically go to Log Analytics.

There is a limitation for Python to support custom logging messages using the Spark configured Log4j. Logs can only be sent from the driver node because executor nodes don't have access to the Java Virtual Machine from Python.

Run the sample application

The monitoring library includes a [sample application](#) that demonstrates how to send both application metrics and application logs to Azure Monitor. To run the sample:

1. Build the **spark-jobs** project in the monitoring library, as described in the [GitHub readme](#).
2. Navigate to your Databricks workspace and create a new job, as described [here](#).
3. In the job detail page, select **Set JAR**.
4. Upload the JAR file from `/src/spark-jobs/target/spark-jobs-1.0-SNAPSHOT.jar`.
5. For **Main class**, enter
`com.microsoft.pnp.samplejob.StreamingQueryListenerSampleJob`.
6. Select a cluster that is already configured to use the monitoring library. See [Configure Azure Databricks to send metrics to Azure Monitor](#).

When the job runs, you can view the application logs and metrics in your Log Analytics workspace.

Application logs appear under `SparkLoggingEvent_CL`:

Kusto

```
SparkLoggingEvent_CL | where logger_name_s contains "com.microsoft.pnp"
```

Application metrics appear under `SparkMetric_CL`:

Kusto

```
SparkMetric_CL | where name_s contains "rowcounter" | limit 50
```

 **Important**

After you verify the metrics appear, stop the sample application job.

Next steps

Deploy the performance monitoring dashboard that accompanies this code library to troubleshoot performance issues in your production Azure Databricks workloads.

[Use dashboards to visualize Azure Databricks metrics](#)

Related resources

- [Monitoring Azure Databricks](#)
- [Troubleshoot performance bottlenecks in Azure Databricks](#)
- [Modern analytics architecture with Azure Databricks](#)
- [Ingestion, ETL, and stream processing pipelines with Azure Databricks](#)

Use dashboards to visualize Azure Databricks metrics

Article • 05/11/2023

ⓘ Note

This article relies on an open source library hosted on GitHub at:

<https://github.com/mspnp/spark-monitoring> ↗.

The original library supports Azure Databricks Runtimes 10.x (Spark 3.2.x) and earlier.

Databricks has contributed an updated version to support Azure Databricks Runtimes 11.0 (Spark 3.3.x) and above on the `14jv2` branch at:

<https://github.com/mspnp/spark-monitoring/tree/14jv2> ↗.

Please note that the 11.0 release is not backwards compatible due to the different logging systems used in the Databricks Runtimes. Be sure to use the correct build for your Databricks Runtime. The library and GitHub repository are in maintenance mode. There are no plans for further releases, and issue support will be best-effort only. For any additional questions regarding the library or the roadmap for monitoring and logging of your Azure Databricks environments, please contact azure-spark-monitoring-help@databricks.com.

This article shows how to set up a Grafana dashboard to monitor Azure Databricks jobs for performance issues.

Azure Databricks is a fast, powerful, and collaborative [Apache Spark](#) ↗-based analytics service that makes it easy to rapidly develop and deploy big data analytics and artificial intelligence (AI) solutions. Monitoring is a critical component of operating Azure Databricks workloads in production. The first step is to gather metrics into a workspace for analysis. In Azure, the best solution for managing log data is [Azure Monitor](#). Azure Databricks does not natively support sending log data to Azure monitor, but a [library for this functionality](#) ↗ is available in [GitHub](#) ↗.

This library enables logging of Azure Databricks service metrics as well as Apache Spark structure streaming query event metrics. Once you've successfully deployed this library to an Azure Databricks cluster, you can further deploy a set of [Grafana](#) ↗ dashboards that you can deploy as part of your production environment.



Prerequisites

Configure your Azure Databricks cluster to use the monitoring library, as described in the [GitHub readme](#) ↗.

Deploy the Azure Log Analytics workspace

To deploy the Azure Log Analytics workspace, follow these steps:

1. Navigate to the `/perf-tools/deployment/loganalytics` directory.
2. Deploy the `logAnalyticsDeploy.json` Azure Resource Manager template. For more information about deploying Resource Manager templates, see [Deploy resources with Resource Manager templates and Azure CLI](#). The template has the following parameters:
 - **location:** The region where the Log Analytics workspace and dashboards are deployed.
 - **serviceTier:** The workspace pricing tier. See [here](#) for a list of valid values.
 - **dataRetention** (optional): The number of days the log data is retained in the Log Analytics workspace. The default value is 30 days. If the pricing tier is `Free`, the data retention must be seven days.
 - **workspaceName** (optional): A name for the workspace. If not specified, the template generates a name.

Azure CLI

```
az deployment group create --resource-group <resource-group-name> --template-file logAnalyticsDeploy.json --parameters location='East US' serviceTier='Standalone'
```

This template creates the workspace and also creates a set of predefined queries that are used by dashboard.

Deploy Grafana in a virtual machine

Grafana is an open source project you can deploy to visualize the time series metrics stored in your Azure Log Analytics workspace using the Grafana plugin for Azure Monitor. Grafana executes on a virtual machine (VM) and requires a storage account, virtual network, and other resources. To deploy a virtual machine with the bitnami-certified Grafana image and associated resources, follow these steps:

1. Use the Azure CLI to accept the Azure Marketplace image terms for Grafana.

Azure CLI

```
az vm image terms accept --publisher bitnami --offer grafana --plan default
```

2. Navigate to the `/spark-monitoring/perf-tools/deployment/grafana` directory in your local copy of the GitHub repo.
3. Deploy the `grafanaDeploy.json` Resource Manager template as follows:

Azure CLI

```
export DATA_SOURCE="https://raw.githubusercontent.com/mspnp/spark-monitoring/master/perf-tools/deployment/grafana/AzureDataSource.sh"
az deployment group create \
--resource-group <resource-group-name> \
--template-file grafanaDeploy.json \
--parameters adminPass='<vm password>' dataSource=$DATA_SOURCE
```

Once the deployment is complete, the bitnami image of Grafana is installed on the virtual machine.

Update the Grafana password

As part of the setup process, the Grafana installation script outputs a temporary password for the **admin** user. You need this temporary password to sign in. To obtain the temporary password, follow these steps:

1. Log in to the Azure portal.
2. Select the resource group where the resources were deployed.
3. Select the VM where Grafana was installed. If you used the default parameter name in the deployment template, the VM name is prefaced with **sparkmonitoring-vm-grafana**.
4. In the **Support + troubleshooting** section, click **Boot diagnostics** to open the boot diagnostics page.
5. Click **Serial log** on the boot diagnostics page.
6. Search for the following string: "Setting Bitnami application password to".
7. Copy the password to a safe location.

Next, change the Grafana administrator password by following these steps:

1. In the Azure portal, select the VM and click **Overview**.
2. Copy the public IP address.
3. Open a web browser and navigate to the following URL: `http://<IP address>:3000`.
4. At the Grafana login screen, enter **admin** for the user name, and use the Grafana password from the previous steps.
5. Once logged in, select **Configuration** (the gear icon).
6. Select **Server Admin**.
7. On the **Users** tab, select the **admin** login.
8. Update the password.

Create an Azure Monitor data source

1. Create a service principal that allows Grafana to manage access to your Log Analytics workspace. For more information, see [Create an Azure service principal with Azure CLI](#)

Azure CLI

```
az ad sp create-for-rbac --name http://<service principal name> \
--role "Log Analytics Reader" \
--scopes /subscriptions/mySubscriptionID
```

2. Note the values for appId, password, and tenant in the output from this command:

JSON

```
{  
  "appId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",  
  "displayName": "azure-cli-2019-03-27-00-33-39",  
  "name": "http://<service principal name>",  
  "password": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",  
  "tenant": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  
}
```

3. Log into Grafana as described earlier. Select **Configuration** (the gear icon) and then **Data Sources**.

4. In the **Data Sources** tab, click **Add data source**.

5. Select **Azure Monitor** as the data source type.

6. In the **Settings** section, enter a name for the data source in the **Name** textbox.

7. In the **Azure Monitor API Details** section, enter the following information:

- Subscription Id: Your Azure subscription ID.
- Tenant Id: The tenant ID from earlier.
- Client Id: The value of "appId" from earlier.
- Client Secret: The value of "password" from earlier.

8. In the **Azure Log Analytics API Details** section, check the **Same Details as Azure Monitor API** checkbox.

9. Click **Save & Test**. If the Log Analytics data source is correctly configured, a success message is displayed.

Create the dashboard

Create the dashboards in Grafana by following these steps:

1. Navigate to the `/perftools/dashboards/grafana` directory in your local copy of the GitHub repo.
2. Run the following script:

Bash

```
export WORKSPACE=<your Azure Log Analytics workspace ID>  
export LOGTYPE=SparkListenerEvent_CL  
  
sh DashGen.sh
```

The output from the script is a file named **SparkMonitoringDash.json**.

3. Return to the Grafana dashboard and select **Create** (the plus icon).
4. Select **Import**.
5. Click **Upload .json File**.
6. Select the **SparkMonitoringDash.json** file created in step 2.
7. In the **Options** section, under **ALA**, select the Azure Monitor data source created earlier.
8. Click **Import**.

Visualizations in the dashboards

Both the Azure Log Analytics and Grafana dashboards include a set of time-series visualizations. Each graph is time-series plot of metric data related to an Apache Spark [job](#), stages of the job, and tasks that make up each stage.

The visualizations are:

Job latency

This visualization shows execution latency for a job, which is a coarse view on the overall performance of a job. Displays the job execution duration from start to completion. Note that the job start time is not the same as the job submission time. Latency is represented as percentiles (10%, 30%, 50%, 90%) of job execution indexed by cluster ID and application ID.

Stage latency

The visualization shows the latency of each stage per cluster, per application, and per individual stage. This visualization is useful for identifying a particular stage that is running slowly.

Task latency

This visualization shows task execution latency. Latency is represented as a percentile of task execution per cluster, stage name, and application.

Sum Task Execution per host

This visualization shows the sum of task execution latency per host running on a cluster. Viewing task execution latency per host identifies hosts that have much higher overall task latency than other hosts. This may mean that tasks have been inefficiently or unevenly distributed to hosts.

Task metrics

This visualization shows a set of the execution metrics for a given task's execution. These metrics include the size and duration of a data shuffle, duration of serialization and deserialization operations, and others. For the full set of metrics, view the Log Analytics query for the panel. This visualization is useful for understanding the operations that make up a task and identifying resource consumption of each operation. Spikes in the graph represent costly operations that should be investigated.

Cluster throughput

This visualization is a high-level view of work items indexed by cluster and application to represent the amount of work done per cluster and application. It shows the number of jobs, tasks, and stages completed per cluster, application, and stage in one minute increments.

Streaming Throughput/Latency

This visualization is related to the metrics associated with a structured streaming query. The graph shows the number of input rows per second and the number of rows processed per second. The streaming metrics are also represented per application. These metrics are sent when the `OnQueryProgress` event is generated as the structured streaming query is processed and the visualization represents streaming latency as the amount of time, in milliseconds, taken to execute a query batch.

Resource consumption per executor

Next is a set of visualizations for the dashboard show the particular type of resource and how it is consumed per executor on each cluster. These visualizations help identify outliers in resource consumption per executor. For example, if the work allocation for a particular executor is skewed, resource consumption will be elevated in relation to other executors running on the cluster. This can be identified by spikes in the resource consumption for an executor.

Executor compute time metrics

Next is a set of visualizations for the dashboard that show the ratio of executor serialize time, deserialize time, CPU time, and Java virtual machine time to overall executor compute time. This demonstrates visually how much each of these four metrics is contributing to overall executor processing.

Shuffle metrics

The final set of visualizations shows the data shuffle metrics associated with a structured streaming query across all executors. These include shuffle bytes read, shuffle bytes written, shuffle memory, and disk usage in queries where the file system is used.

Next steps

[Troubleshoot performance bottlenecks](#)

Related resources

- [Monitoring Azure Databricks](#)
- [Send Azure Databricks application logs to Azure Monitor](#)
- [Modern analytics architecture with Azure Databricks](#)
- [Ingestion, ETL, and stream processing pipelines with Azure Databricks](#)

Troubleshoot performance bottlenecks in Azure Databricks

Article • 03/27/2023

ⓘ Note

This article relies on an open source library hosted on GitHub at:

<https://github.com/mspnp/spark-monitoring> ↗.

The original library supports Azure Databricks Runtimes 10.x (Spark 3.2.x) and earlier.

Databricks has contributed an updated version to support Azure Databricks Runtimes 11.0 (Spark 3.3.x) and above on the `l4jv2` branch at:

<https://github.com/mspnp/spark-monitoring/tree/l4jv2> ↗.

Please note that the 11.0 release is not backwards compatible due to the different logging systems used in the Databricks Runtimes. Be sure to use the correct build for your Databricks Runtime. The library and GitHub repository are in maintenance mode. There are no plans for further releases, and issue support will be best-effort only. For any additional questions regarding the library or the roadmap for monitoring and logging of your Azure Databricks environments, please contact azure-spark-monitoring-help@databricks.com.

This article describes how to use monitoring dashboards to find performance bottlenecks in Spark jobs on Azure Databricks.

Azure Databricks is an [Apache Spark](#) ↗–based analytics service that makes it easy to rapidly develop and deploy big data analytics. Monitoring and troubleshooting performance issues is a critical when operating production Azure Databricks workloads. To identify common performance issues, it's helpful to use monitoring visualizations based on telemetry data.

Prerequisites

To set up the Grafana dashboards shown in this article:

- Configure your Databricks cluster to send telemetry to a Log Analytics workspace, using the Azure Databricks Monitoring Library. For details, see the [GitHub readme](#) ↗.

- Deploy Grafana in a virtual machine. See [Use dashboards to visualize Azure Databricks metrics](#).

The Grafana dashboard that is deployed includes a set of time-series visualizations. Each graph is time-series plot of metrics related to an Apache Spark job, the stages of the job, and tasks that make up each stage.

Azure Databricks performance overview

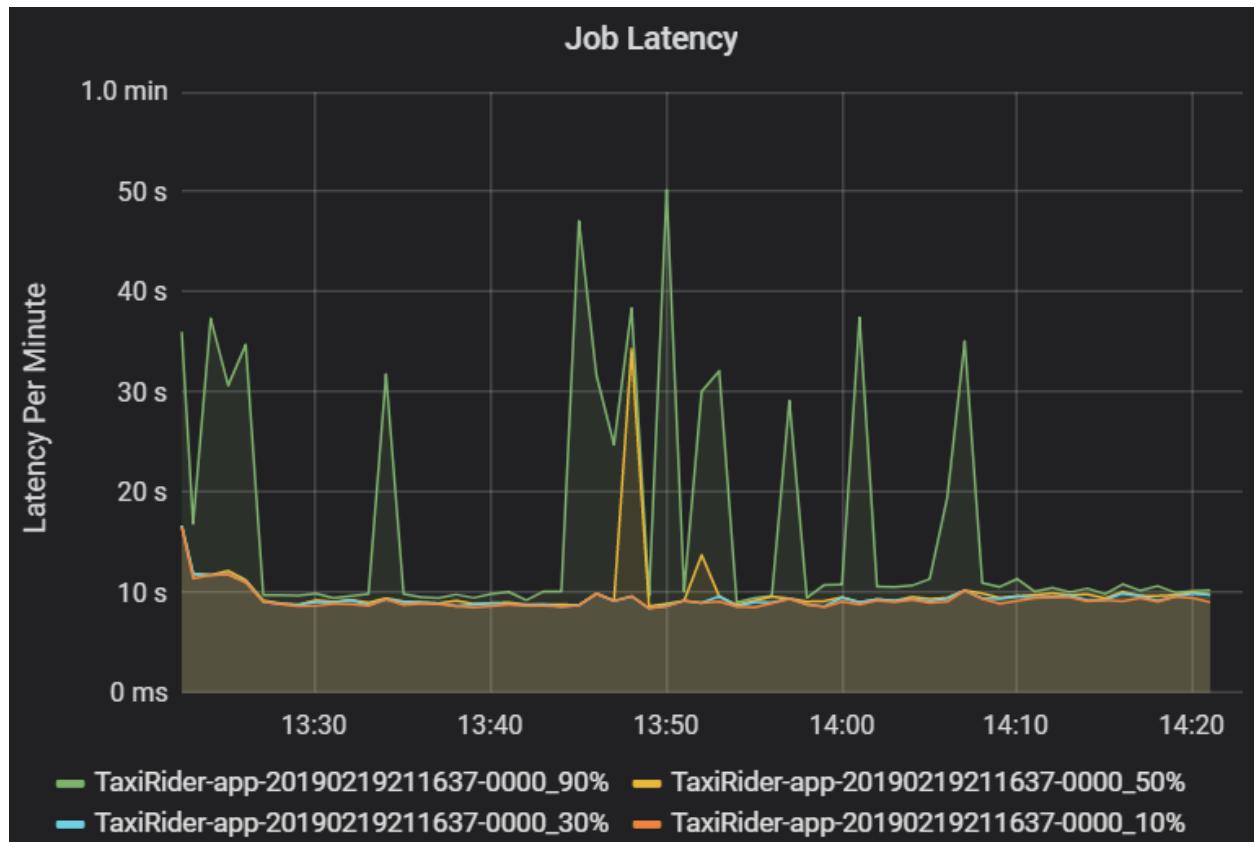
Azure Databricks is based on Apache Spark, a general-purpose distributed computing system. Application code, known as a **job**, executes on an Apache Spark cluster, coordinated by the cluster manager. In general, a job is the highest-level unit of computation. A job represents the complete operation performed by the Spark application. A typical operation includes reading data from a source, applying data transformations, and writing the results to storage or another destination.

Jobs are broken down into **stages**. The job advances through the stages sequentially, which means that later stages must wait for earlier stages to complete. Stages contain groups of identical **tasks** that can be executed in parallel on multiple nodes of the Spark cluster. Tasks are the most granular unit of execution taking place on a subset of the data.

The next sections describe some dashboard visualizations that are useful for performance troubleshooting.

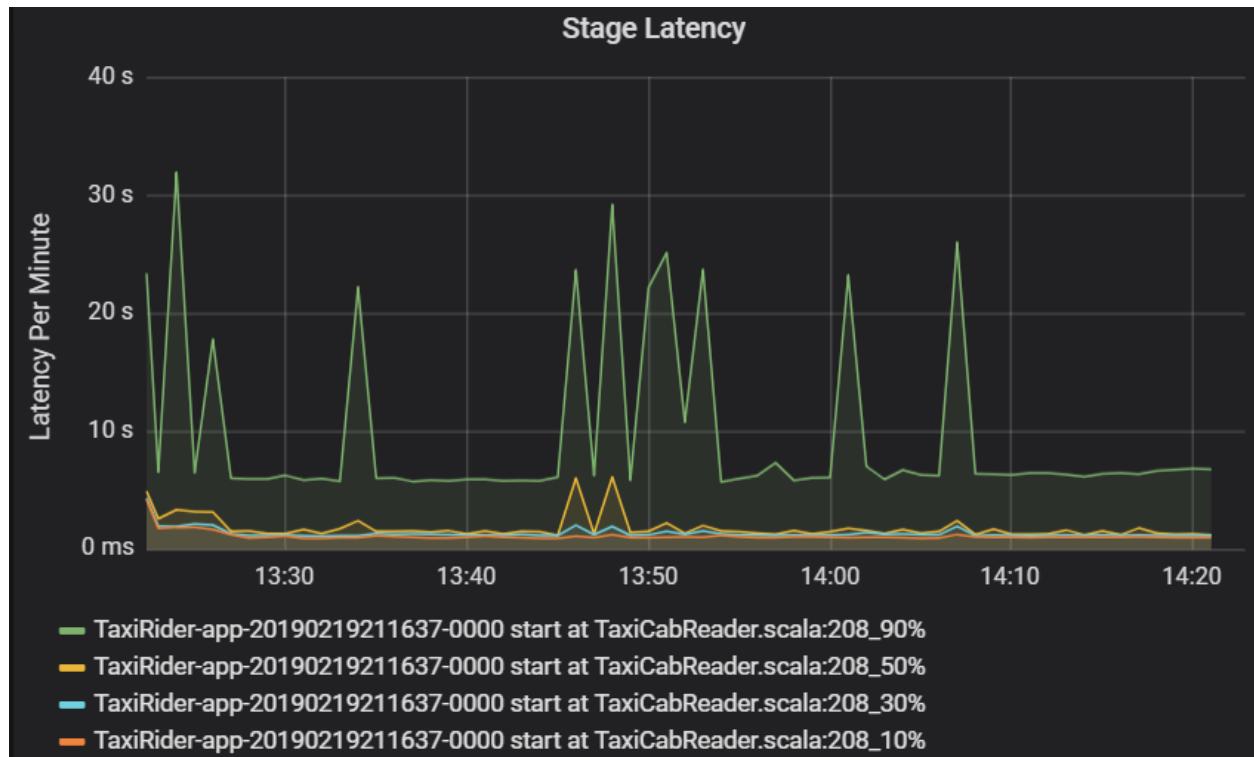
Job and stage latency

Job latency is the duration of a job execution from when it starts until it completes. It is shown as percentiles of a job execution per cluster and application ID, to allow the visualization of outliers. The following graph shows a job history where the 90th percentile reached 50 seconds, even though the 50th percentile was consistently around 10 seconds.

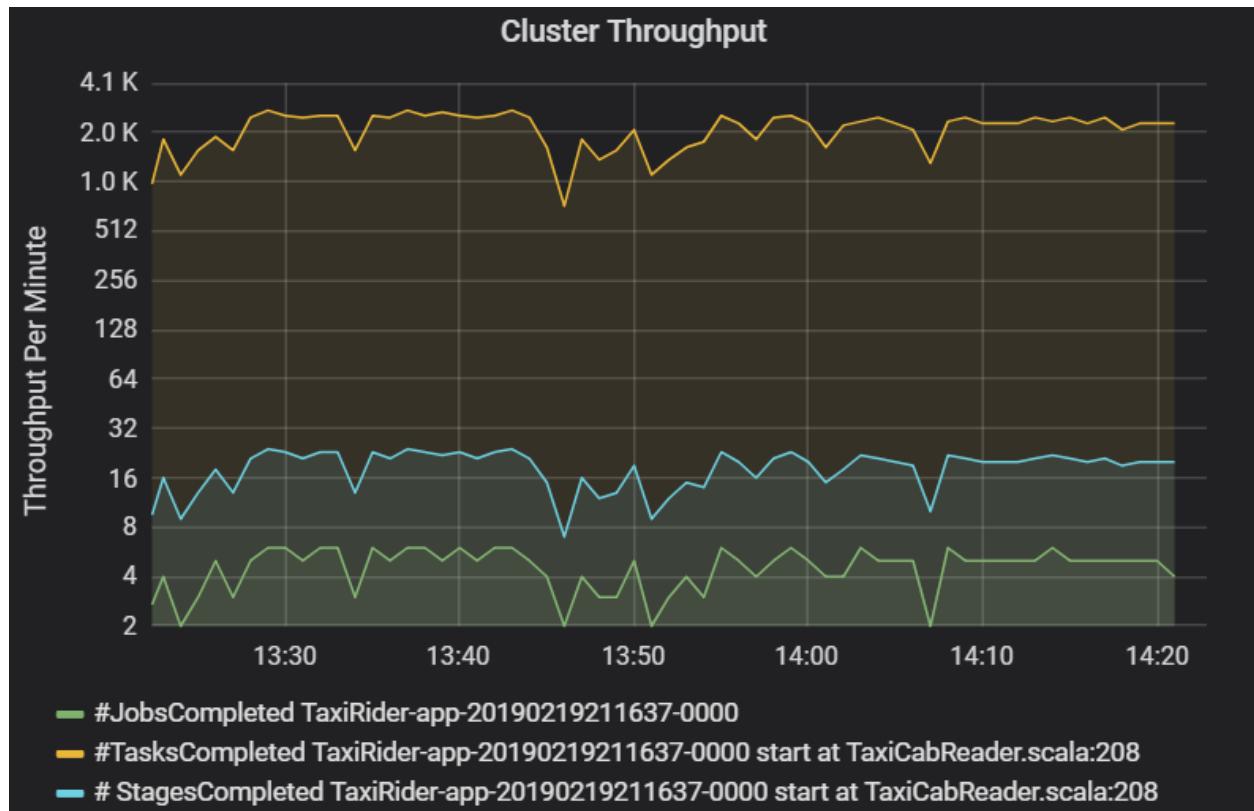


Investigate job execution by cluster and application, looking for spikes in latency. Once clusters and applications with high latency are identified, move on to investigate stage latency.

Stage latency is also shown as percentiles to allow the visualization of outliers. Stage latency is broken out by cluster, application, and stage name. Identify spikes in task latency in the graph to determine which tasks are holding back completion of the stage.

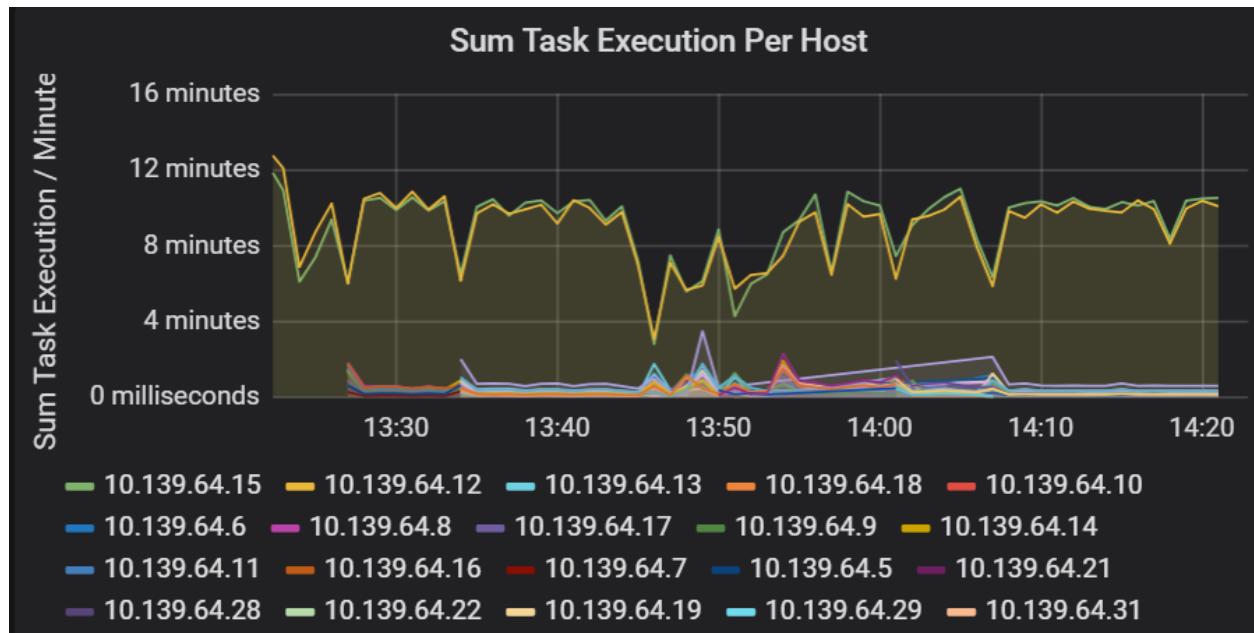


The cluster throughput graph shows the number of jobs, stages, and tasks completed per minute. This helps you to understand the workload in terms of the relative number of stages and tasks per job. Here you can see that the number of jobs per minute ranges between 2 and 6, while the number of stages is about 12 – 24 per minute.

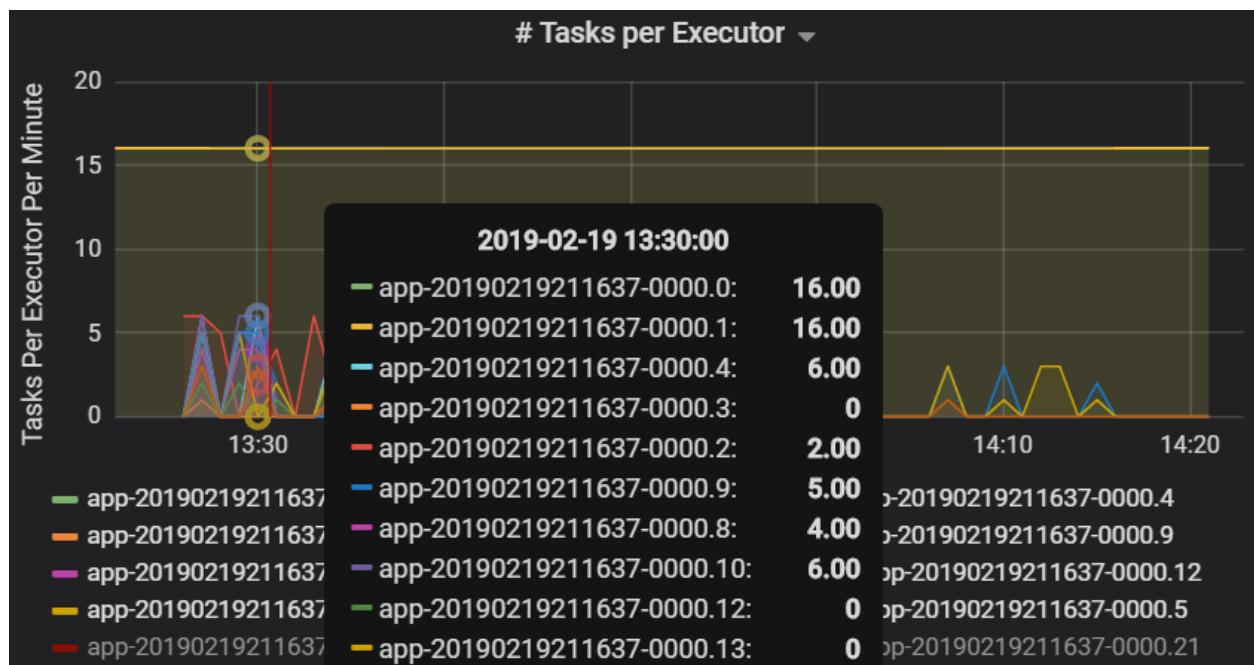


Sum of task execution latency

This visualization shows the sum of task execution latency per host running on a cluster. Use this graph to detect tasks that run slowly due to the host slowing down on a cluster, or a misallocation of tasks per executor. In the following graph, most of the hosts have a sum of about 30 seconds. However, two of the hosts have sums that hover around 10 minutes. Either the hosts are running slow or the number of tasks per executor is misallocated.



The number of tasks per executor shows that two executors are assigned a disproportionate number of tasks, causing a bottleneck.



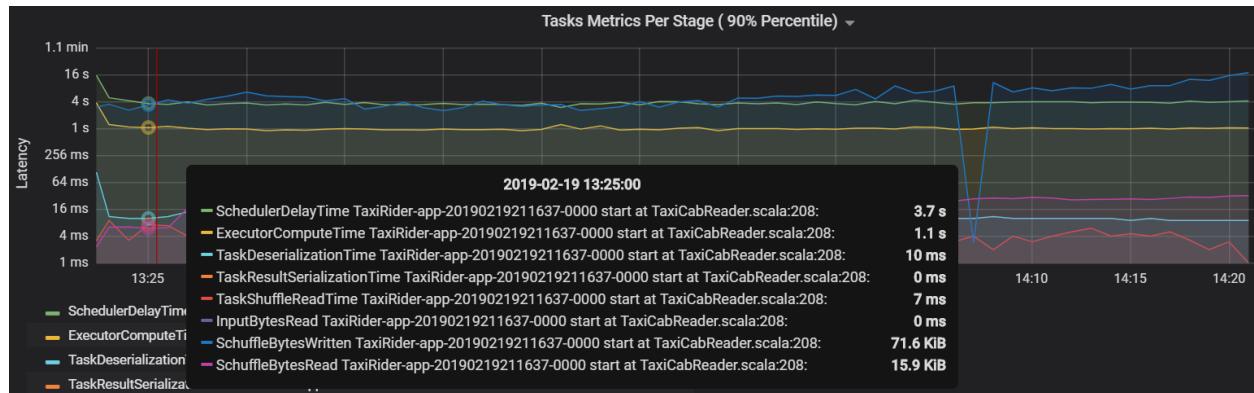
Task metrics per stage

The task metrics visualization gives the cost breakdown for a task execution. You can use it to see the relative time spent on tasks such as serialization and deserialization. This data might show opportunities to optimize — for example, by using [broadcast variables](#) to avoid shipping data. The task metrics also show the shuffle data size for a task, and the shuffle read and write times. If these values are high, it means that a lot of data is moving across the network.

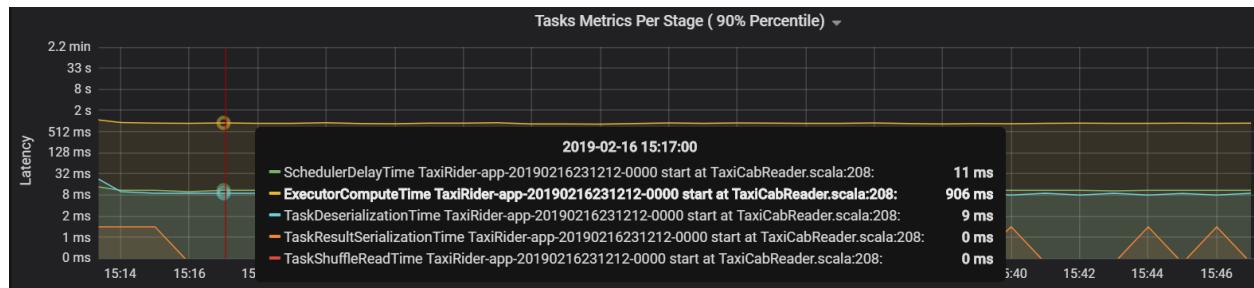
Another task metric is the scheduler delay, which measures how long it takes to schedule a task. Ideally, this value should be low compared to the executor compute

time, which is the time spent actually executing the task.

The following graph shows a scheduler delay time (3.7 s) that exceeds the executor compute time (1.1 s). That means more time is spent waiting for tasks to be scheduled than doing the actual work.



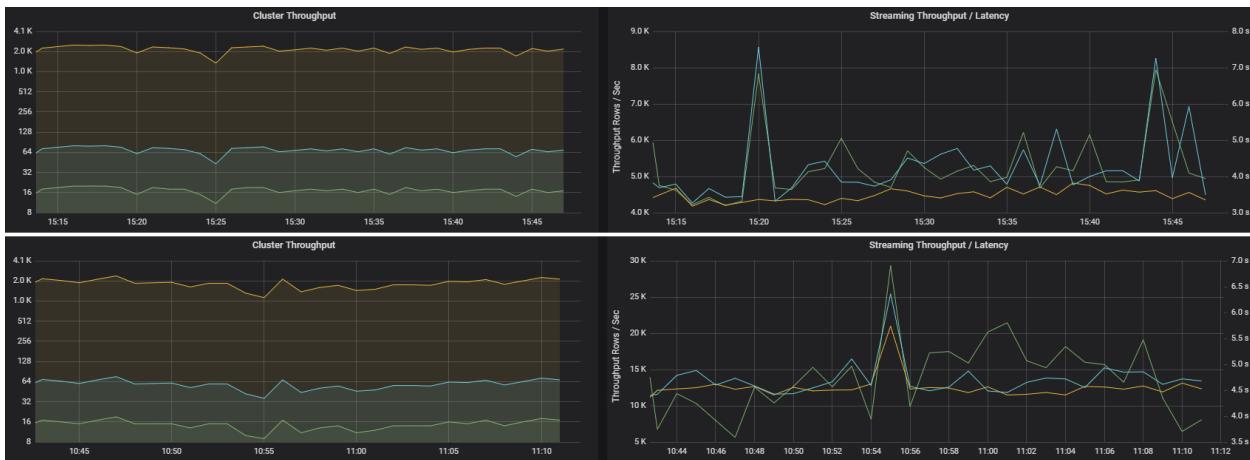
In this case, the problem was caused by having too many partitions, which caused a lot of overhead. Reducing the number of partitions lowered the scheduler delay time. The next graph shows that most of the time is spent executing the task.



Streaming throughput and latency

Streaming throughput is directly related to structured streaming. There are two important metrics associated with streaming throughput: Input rows per second and processed rows per second. If input rows per second outpaces processed rows per second, it means the stream processing system is falling behind. Also, if the input data comes from Event Hubs or Kafka, then input rows per second should keep up with the data ingestion rate at the front end.

Two jobs can have similar cluster throughput but very different streaming metrics. The following screenshot shows two different workloads. They are similar in terms of cluster throughput (jobs, stages, and tasks per minute). But the second run processes 12,000 rows/sec versus 4,000 rows/sec.



Streaming throughput is often a better business metric than cluster throughput, because it measures the number of data records that are processed.

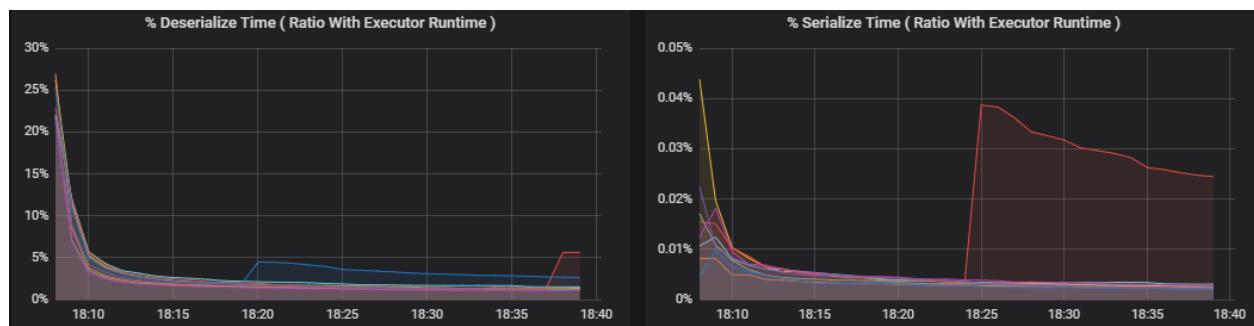
Resource consumption per executor

These metrics help to understand the work that each executor performs.

Percentage metrics measure how much time an executor spends on various things, expressed as a ratio of time spent versus the overall executor compute time. The metrics are:

- % Serialize time
- % Deserialize time
- % CPU executor time
- % JVM time

These visualizations show how much each of these metrics contributes to overall executor processing.



Shuffle metrics are metrics related to data shuffling across the executors.

- Shuffle I/O
- Shuffle memory
- File system usage
- Disk usage

Common performance bottlenecks

Two common performance bottlenecks in Spark are *task stragglers* and a *non-optimal shuffle partition count*.

Task stragglers

The stages in a job are executed sequentially, with earlier stages blocking later stages. If one task executes a shuffle partition more slowly than other tasks, all tasks in the cluster must wait for the slow task to catch up before the stage can end. This can happen for the following reasons:

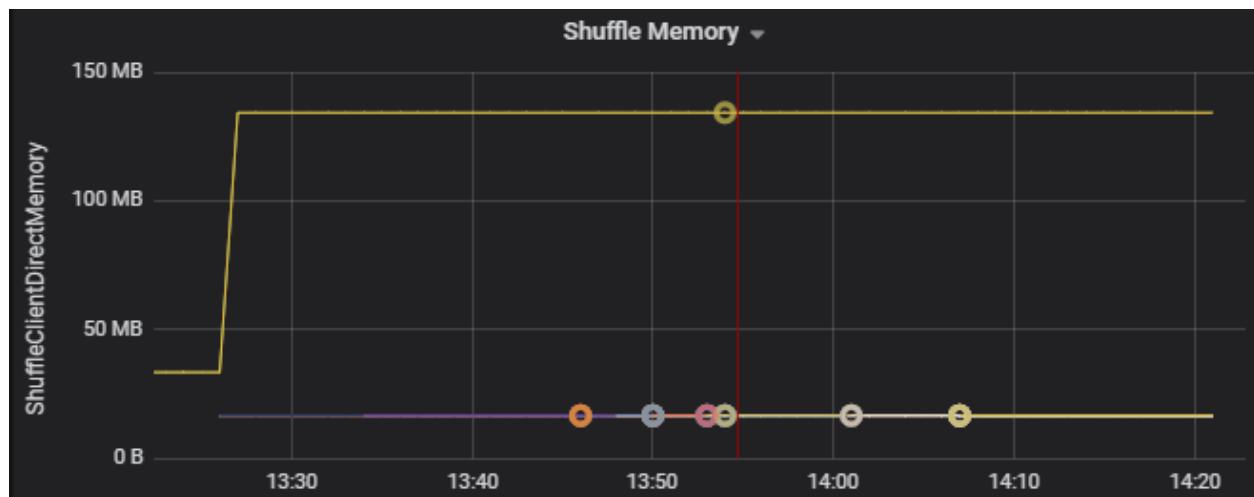
1. A host or group of hosts are running slow. Symptoms: High task, stage, or job latency and low cluster throughput. The summation of tasks latencies per host won't be evenly distributed. However, resource consumption will be evenly distributed across executors.
2. Tasks have an expensive aggregation to execute (data skewing). Symptoms: High task latency, high stage latency, high job latency, or low cluster throughput, but the summation of latencies per host is evenly distributed. Resource consumption will be evenly distributed across executors.
3. If partitions are of unequal size, a larger partition may cause unbalanced task execution (partition skewing). Symptoms: Executor resource consumption is high compared to other executors running on the cluster. All tasks running on that executor will run slow and hold the stage execution in the pipeline. Those stages are said to be *stage barriers*.

Non-optimal shuffle partition count

During a structured streaming query, the assignment of a task to an executor is a resource-intensive operation for the cluster. If the shuffle data isn't the optimal size, the amount of delay for a task will negatively impact throughput and latency. If there are too few partitions, the cores in the cluster will be underutilized which can result in processing inefficiency. Conversely, if there are too many partitions, there's a great deal of management overhead for a small number of tasks.

Use the resource consumption metrics to troubleshoot partition skewing and misallocation of executors on the cluster. If a partition is skewed, executor resources will be elevated in comparison to other executors running on the cluster.

For example, the following graph shows that the memory used by shuffling on the first two executors is 90X bigger than the other executors:



Next steps

- Monitoring Azure Databricks in an Azure Log Analytics Workspace [↗](#)
- Learning path: Build and operate machine learning solutions with Azure Databricks
- Azure Databricks documentation
- Azure Monitor overview

Related resources

- Monitoring Azure Databricks
- Send Azure Databricks application logs to Azure Monitor
- Use dashboards to visualize Azure Databricks metrics
- Modern analytics architecture with Azure Databricks
- Ingestion, ETL, and stream processing pipelines with Azure Databricks

Observability patterns and metrics for performance tuning

Azure Databricks

Azure Log Analytics

Azure Monitor

ⓘ Note

This article relies on an open source library hosted on GitHub at:
<https://github.com/mspnp/spark-monitoring>.

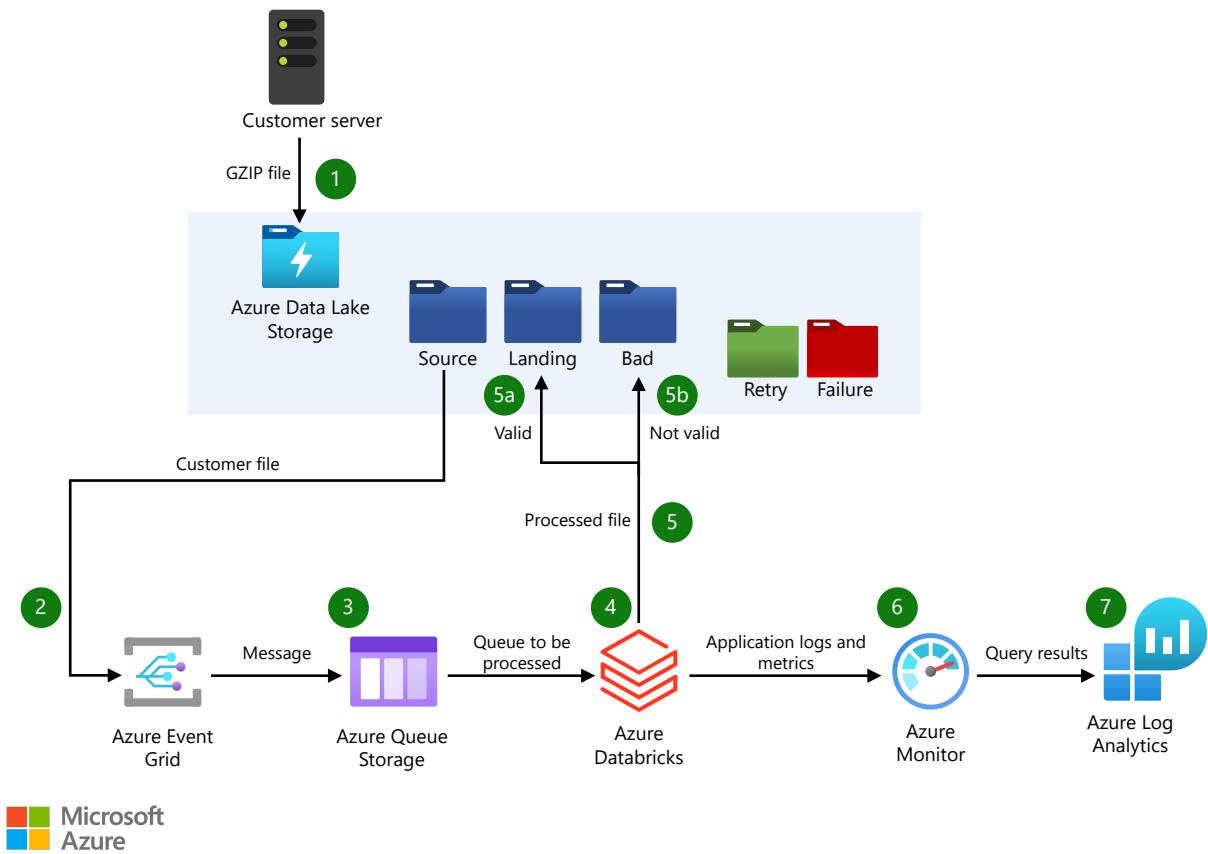
The original library supports Azure Databricks Runtimes 10.x (Spark 3.2.x) and earlier.

Databricks has contributed an updated version to support Azure Databricks Runtimes 11.0 (Spark 3.3.x) and above on the `14jv2` branch at:
<https://github.com/mspnp/spark-monitoring/tree/14jv2>.

Please note that the 11.0 release is not backwards compatible due to the different logging systems used in the Databricks Runtimes. Be sure to use the correct build for your Databricks Runtime. The library and GitHub repository are in maintenance mode. There are no plans for further releases, and issue support will be best-effort only. For any additional questions regarding the library or the roadmap for monitoring and logging of your Azure Databricks environments, please contact azure-spark-monitoring-help@databricks.com.

This solution demonstrates observability patterns and metrics to improve the processing performance of a big data system that uses Azure Databricks.

Architecture



Download a [Visio file](#) of this architecture.

Workflow

The solution involves the following steps:

1. The server sends a large GZIP file that's grouped by customer to the **Source** folder in Azure Data Lake Storage (ADLS).
2. ADLS then sends a successfully extracted customer file to Azure Event Grid, which turns the customer file data into several messages.
3. Azure Event Grid sends the messages to the Azure Queue Storage service, which stores them in a queue.
4. Azure Queue Storage sends the queue to the Azure Databricks data analytics platform for processing.
5. Azure Databricks unpacks and processes queue data into a processed file that it sends back to ADLS:
 - a. If the processed file is valid, it goes in the **Landing** folder.
 - b. Otherwise, the file goes in the **Bad** folder tree. Initially, the file goes in the **Retry** subfolder, and ADLS attempts customer file processing again (step 2). If a pair

of retry attempts still leads to Azure Databricks returning processed files that aren't valid, the processed file goes in the **Failure** subfolder.

6. As Azure Databricks unpacks and processes data in the previous step, it also sends application logs and metrics to Azure Monitor for storage.
7. An Azure Log Analytics workspace applies Kusto queries on the application logs and metrics from Azure Monitor for troubleshooting and deep diagnostics.

Components

- [Azure Data Lake Storage](#) is a set of capabilities dedicated to big data analytics.
- [Azure Event Grid](#) allows a developer to easily build applications with event-based architectures.
- [Azure Queue Storage](#) is a service for storing large numbers of messages. It allows access to messages from anywhere in the world through authenticated calls using HTTP or HTTPS. You can use queues to create a backlog of work to process asynchronously.
- [Azure Databricks](#) is a data analytics platform optimized for the Azure cloud platform. One of the two environments Azure Databricks offers for developing data-intensive applications is [Azure Databricks Workspace](#), an Apache Spark-based unified analytics engine for large-scale data processing.
- [Azure Monitor](#) collects and analyzes app telemetry, such as performance metrics and activity logs.
- [Azure Log Analytics](#) is a tool used to edit and run log queries with data.

Scenario details

Your development team can use observability patterns and metrics to find bottlenecks and improve the performance of a big data system. Your team has to do load testing of a high-volume stream of metrics on a high-scale application.

This scenario offers guidance for performance tuning. Since the scenario presents a performance challenge for logging per customer, it uses Azure Databricks, which can monitor these items robustly:

- Custom application metrics
- Streaming query events
- Application log messages

Azure Databricks can send this monitoring data to different logging services, such as Azure Log Analytics.

This scenario outlines the ingestion of a large set of data that has been grouped by customer and stored in a GZIP archive file. Detailed logs are unavailable from Azure Databricks outside of the real-time Apache Spark™ user interface, so your team needs a way to store all the data for each customer, and then benchmark and compare. With a large data scenario, it's important to find an optimal combination executor pool and virtual machine (VM) size for the fastest processing time. For this business scenario, the overall application relies on the speed of ingestion and querying requirements, so that system throughput doesn't degrade unexpectedly with increasing work volume. The scenario must guarantee that the system meets service-level agreements (SLAs) that are established with your customers.

Potential use cases

Scenarios that can benefit from this solution include:

- System health monitoring.
- Performance maintenance.
- Monitoring day-to-day system usage.
- Spotting trends that might cause future problems if unaddressed.

Considerations

These considerations implement the pillars of the Azure Well-Architected Framework, which is a set of guiding tenets that can be used to improve the quality of a workload.

For more information, see [Microsoft Azure Well-Architected Framework](#).

Keep these points in mind when considering this architecture:

- Azure Databricks can automatically allocate the computing resources necessary for a large job, which avoids problems that other solutions introduce. For example, with [Databricks-optimized autoscaling on Apache Spark](#), excessive provisioning may cause the suboptimal use of resources. Or you might not know the number of executors required for a job.
- A queue message in Azure Queue Storage can be up to 64 KB in size. A queue may contain millions of queue messages, up to the total capacity limit of a storage account.

Cost optimization

Cost optimization is about looking at ways to reduce unnecessary expenses and improve operational efficiencies. For more information, see [Overview of the cost optimization pillar](#).

Use the [Azure pricing calculator](#) to estimate the cost of implementing this solution.

Deploy this scenario

Note

The deployment steps described here apply only to Azure Databricks, Azure Monitor, and Azure Log Analytics. Deployment of the other components isn't covered in this article.

To get all the logs and information of the process, set up Azure Log Analytics and the Azure Databricks monitoring library. The monitoring library streams Apache Spark level events and Spark Structured Streaming metrics from your jobs to Azure Monitor. You don't need to make any changes to your application code for these events and metrics.

The steps to set up performance tuning for a big data system are as follows:

1. In the Azure portal, [create an Azure Databricks workspace](#). Copy and save the Azure subscription ID (a GUID), resource group name, Databricks workspace name, and workspace portal URL for later use.
2. In a web browser, go to the Databricks workspace URL and [generate a Databricks personal access token](#). Copy and save the token string that appears (which begins with `dapi` and a 32-character hexadecimal value) for later use.
3. Clone the [mspnp/spark-monitoring](#) GitHub repository onto your local computer. This repository has the source code for the following components:
 - The Azure Resource Manager (ARM) template for creating an Azure Log Analytics workspace, which also installs prebuilt queries for collecting Spark metrics
 - Azure Databricks monitoring libraries
 - The sample application for sending application metrics and application logs from Azure Databricks to Azure Monitor
4. Using the [Azure CLI](#) command for deploying an ARM template, [create an Azure Log Analytics workspace with prebuilt Spark metric queries](#). From the command

output, copy and save the generated name for the new Log Analytics workspace (in the format *spark-monitoring-<randomized-string>*).

5. In the Azure portal, copy and save your Log Analytics [workspace ID and key](#) for later use.
6. Install the Community Edition of [IntelliJ IDEA](#), an integrated development environment (IDE) that has built-in support for the [Java Development Kit](#) (JDK) and [Apache Maven](#). Add the [Scala plug-in](#).
7. Using IntelliJ IDEA, [build the Azure Databricks monitoring libraries](#). To do the actual build step, select **View > Tool Windows > Maven** to show the Maven tools window, and then select **Execute Maven Goal > mvn package**.
8. Using a [Python](#) package installation tool, install the [Azure Databricks CLI](#) and set up authentication with the Databricks personal access token you copied earlier.
9. [Configure the Azure Databricks workspace](#) by modifying the Databricks init script with the Databricks and Log Analytics values you copied earlier, and then using the Azure Databricks CLI to copy the init script and the Azure Databricks monitoring libraries to your Databricks workspace.
10. In your Databricks workspace portal, [create and configure an Azure Databricks cluster](#).
11. In IntelliJ IDEA, [build the sample application](#) using Maven. Then in your Databricks workspace portal, run the sample application to generate sample logs and metrics for Azure Monitor.
12. While the sample job is running in Azure Databricks, go to the Azure portal to view and query the event types (application logs and metrics) in the [Log Analytics interface](#):
 - a. Select **Tables > Custom Logs** to view the table schema for Spark listener events (**SparkListenerEvent_CL**), Spark logging events (**SparkLoggingEvent_CL**), and Spark metrics (**SparkMetric_CL**).
 - b. Select **Query explorer > Saved Queries > Spark Metrics** to view and run the queries that were added when you created the Log Analytics workspace.

Read more about viewing and running prebuilt and custom queries in the next section.

Query the logs and metrics in Azure Log Analytics

Access prebuilt queries

The prebuilt query names for retrieving Spark metrics are listed below.

- % CPU Time Per Executor
- % Deserialize Time Per Executor
- % JVM Time Per Executor
- % Serialize Time Per Executor
- Disk Bytes Spilled
- Error Traces (Bad Record Or Bad Files)
- File System Bytes Read Per Executor
- File System Bytes Write Per Executor
- Job Errors Per Job
- Job Latency Per Job (Batch Duration)
- Job Throughput
- Running Executors
- Shuffle Bytes Read
- Shuffle Bytes Read Per Executor
- Shuffle Bytes Read To Disk Per Executor
- Shuffle Client Direct Memory
- Shuffle Client Memory Per Executor
- Shuffle Disk Bytes Spilled Per Executor
- Shuffle Heap Memory Per Executor
- Shuffle Memory Bytes Spilled Per Executor
- Stage Latency Per Stage (Stage Duration)
- Stage Throughput Per Stage
- Streaming Errors Per Stream
- Streaming Latency Per Stream
- Streaming Throughput Input Rows/Sec
- Streaming Throughput Processed Rows/Sec
- Sum Task Execution Per Host
- Task Deserialization Time
- Task Errors Per Stage
- Task Executor Compute Time (Data Skew Time)
- Task Input Bytes Read
- Task Latency Per Stage (Tasks Duration)
- Task Result Serialization Time
- Task Scheduler Delay Latency
- Task Shuffle Bytes Read
- Task Shuffle Bytes Written
- Task Shuffle Read Time

- Task Shuffle Write Time
- Task Throughput (Sum Of Tasks Per Stage)
- Tasks Per Executor (Sum Of Tasks Per Executor)
- Tasks Per Stage

Write custom queries

You can also write your own queries in [Kusto Query Language \(KQL\)](#). Just select the top middle pane, which is editable, and customize the query to meet your needs.

The following two queries pull data from the Spark logging events:

Kusto

```
SparkLoggingEvent_CL | where logger_name_s contains "com.microsoft.pnp"
```

Kusto

```
SparkLoggingEvent_CL
| where TimeGenerated > ago(7d)
| project TimeGenerated, clusterName_s, logger_name_s
| summarize Count=count() by clusterName_s, logger_name_s,
bin(TimeGenerated, 1h)
```

And these two examples are queries on the Spark metrics log:

Kusto

```
SparkMetric_CL
| where name_s contains "executor.cpuTime"
| extend sname = split(name_s, ".")
| extend executor=strcat(sname[0], ".", sname[1])
| project TimeGenerated, cpuTime=count_d / 100000
```

Kusto

```
SparkMetric_CL
| where name_s contains "driver.jvm.total."
| where executorId_s == "driver"
| extend memUsed_GB = value_d / 1000000000
| project TimeGenerated, name_s, memUsed_GB
| summarize max(memUsed_GB) by tostring(name_s), bin(TimeGenerated, 1m)
```

Query terminology

The following table explains some of the terms that are used when you construct a query of application logs and metrics.

[\[+\] Expand table](#)

Term	ID	Remarks
Cluster_init	Application ID	
Queue	Run ID	One run ID equals multiple batches.
Batch	Batch ID	One batch equals two jobs.
Job	Job ID	One job equals two stages.
Stage	Stage ID	One stage has 100-200 task IDs depending on the task (read, shuffle, or write).
Tasks	Task ID	One task is assigned to one executor. One task is assigned to do a <code>partitionBy</code> for one partition. For about 200 customers, there should be 200 tasks.

The following sections contain the typical metrics used in this scenario for monitoring system throughput, Spark job running status, and system resources usage.

System throughput

[\[+\] Expand table](#)

Name	Measurement	Units
Stream throughput	Average input rate over average processed rate per minute	Rows per minute
Job duration	Average ended Spark job duration per minute	Duration(s) per minute
Job count	Average number of ended Spark jobs per	Number of jobs per

Name	Measurement	Units
	minute	minute
Stage duration	Average completed stages duration per minute	Duration(s) per minute
Stage count	Average number of completed stages per minute	Number of stages per minute
Task duration	Average finished tasks duration per minute	Duration(s) per minute
Task count	Average number of finished tasks per minute	Number of tasks per minute

Spark job running status

[\[+\] Expand table](#)

Name	Measurement	Units
Scheduler pool count	Number of distinct count of scheduler pools per minute (number of queues operating)	Number of scheduler pools
Number of running executors	Number of running executors per minute	Number of running executors
Error trace	All error logs with <code>Error</code> level and the corresponding tasks/stage ID (shown in <code>thread_name_s</code>)	

System resources usage

[\[+\] Expand table](#)

Name	Measurement	Units
Average CPU usage per executor/overall	Percent of CPU used per executor per minute	% per minute
Average used direct memory (MB) per host	Average used direct memory per executors per minute	MB per minute
Spilled memory per host	Average spilled memory per executor	MB per minute
Monitor data skew impact on duration	Measure range and difference of 70th-90th percentile and 90th-100th percentile in tasks duration	Net difference among 100%, 90%, and 70%; percentage difference among 100%, 90%, and 70%

Decide how to relate the customer input, which was combined into a GZIP archive file, to a particular Azure Databricks output file, since Azure Databricks handles the whole batch operation as a unit. Here, you apply granularity to the tracing. You also use custom metrics to trace one output file to the original input file.

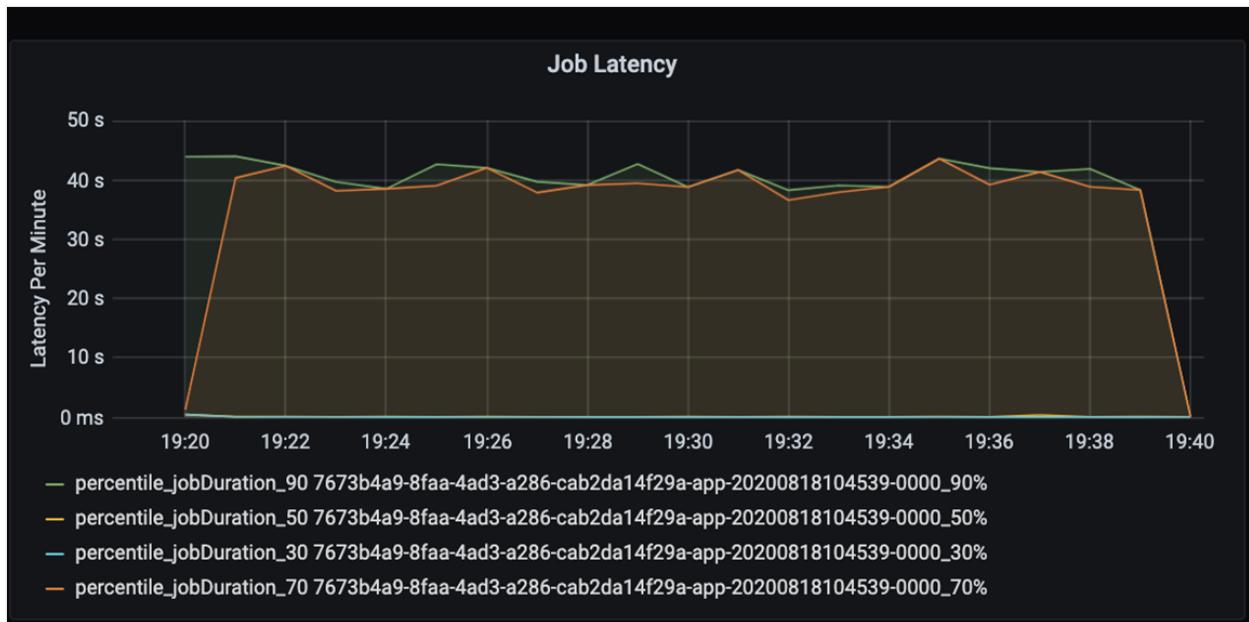
For more detailed definitions of each metric, see [Visualizations in the dashboards](#) on this website, or see the [Metrics](#) section in the Apache Spark documentation.

Assess performance tuning options

Baseline definition

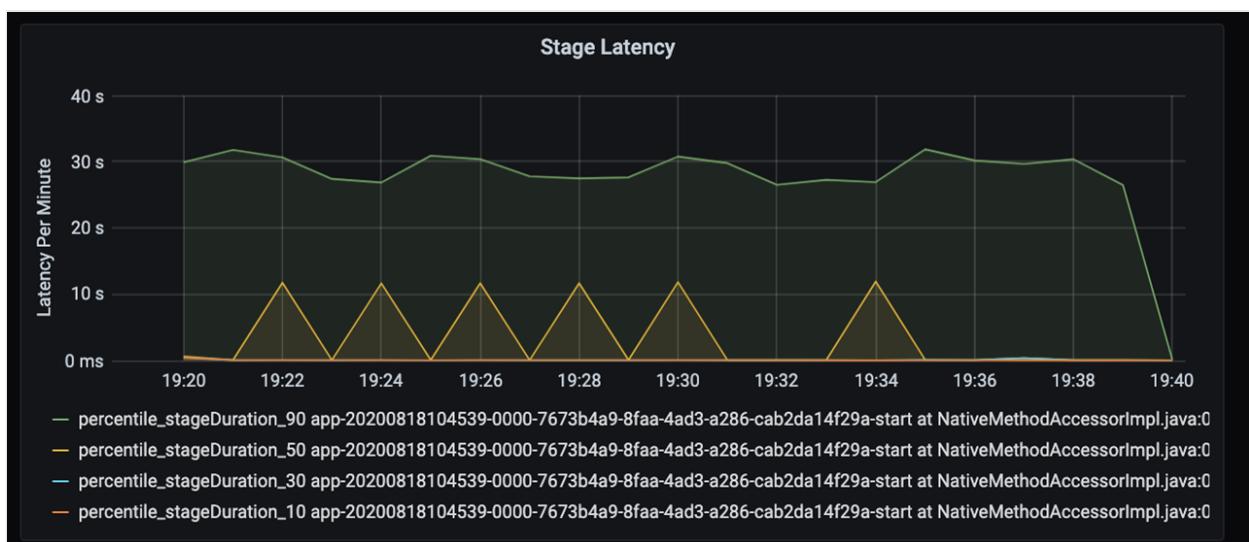
You and your development team should establish a baseline, so that you can compare future states of the application.

Measure the performance of your application quantitatively. In this scenario, the key metric is job latency, which is typical of most data preprocessing and ingestion. Attempt to accelerate the data processing time and focus on measuring latency, as in the chart below:



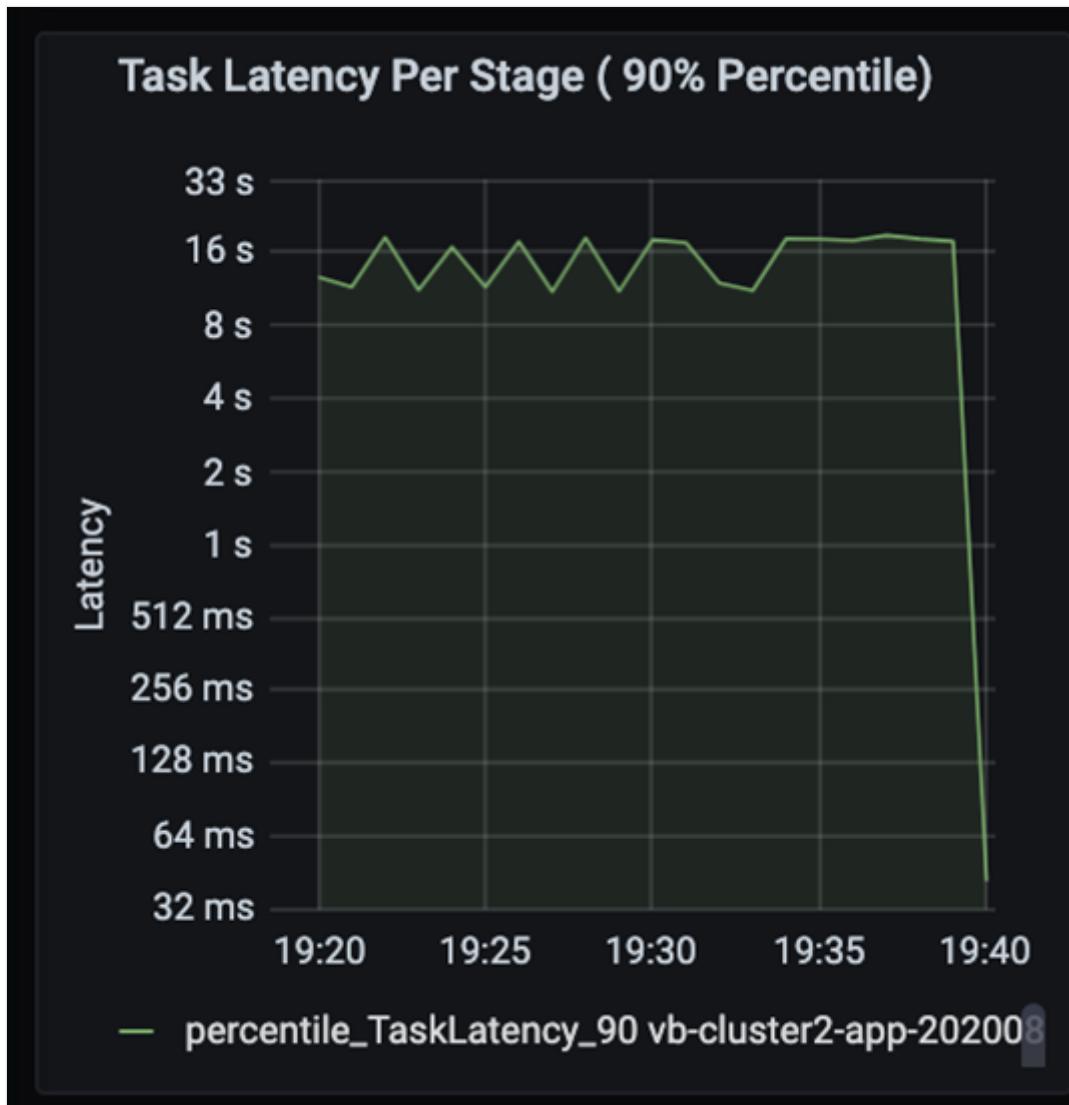
Measure the execution latency for a job: a coarse view on the overall job performance, and the job execution duration from start to completion (microbatch time). In the chart above, at the 19:30 mark, it takes about 40 seconds in duration to process the job.

If you look further into those 40 seconds, you see the data below for stages:



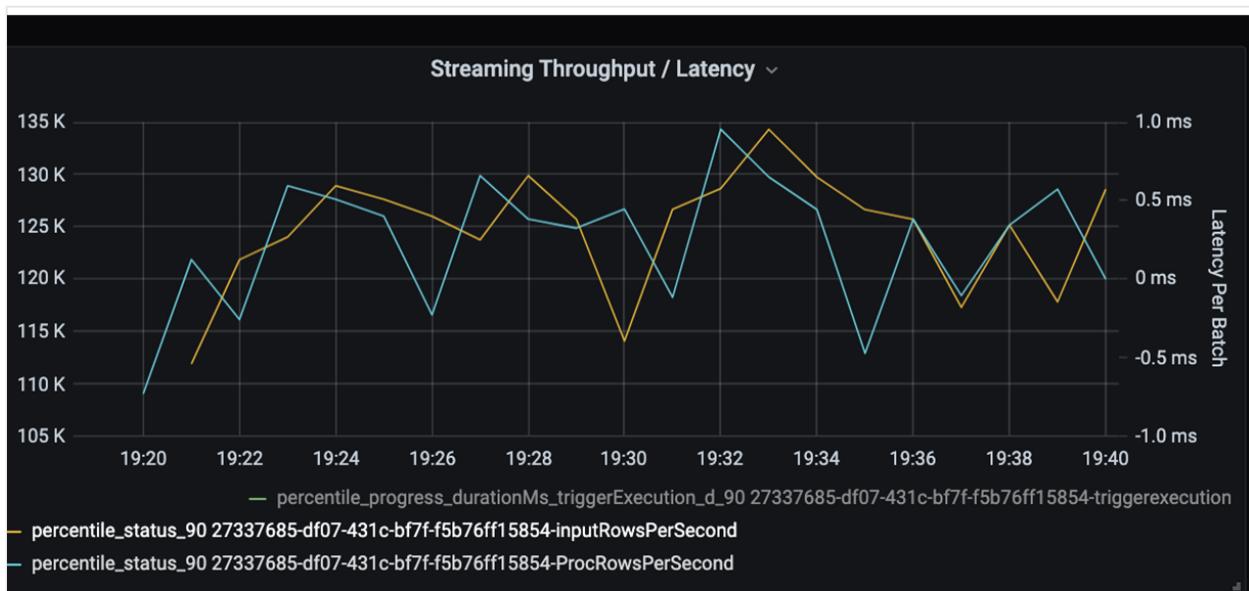
At the 19:30 mark, there are two stages: an orange stage of 10 seconds, and a green stage at 30 seconds. Monitor whether a stage spikes, because a spike indicates a delay in a stage.

Investigate when a certain stage is running slowly. In the partitioning scenario, there are typically at least two stages: one stage to read a file, and the other stage to shuffle, partition, and write the file. If you have high stage latency mostly in the writing stage, you might have a bottleneck problem during partitioning.



Observe the tasks as the stages in a job execute sequentially, with earlier stages blocking later stages. Within a stage, if one task executes a shuffle partition slower than other tasks, all tasks in the cluster must wait for the slower task to finish for the stage to complete. Tasks are then a way to monitor data skew and possible bottlenecks. In the chart above, you can see that all of the tasks are evenly distributed.

Now monitor the processing time. Because you have a streaming scenario, look at the streaming throughput.



In the streaming throughput/batch latency chart above, the orange line represents input rate (input rows per second). The blue line represents the processing rate (processed rows per second). At some points, the processing rate doesn't catch the input rate. The potential issue is that input files are piling up in the queue.

Because the processing rate doesn't match the input rate in the graph, look to improve the process rate to cover the input rate fully. One possible reason might be the imbalance of customer data in each partition key that leads to a bottleneck. For a next step and potential solution, take advantage of the scalability of Azure Databricks.

Partitioning investigation

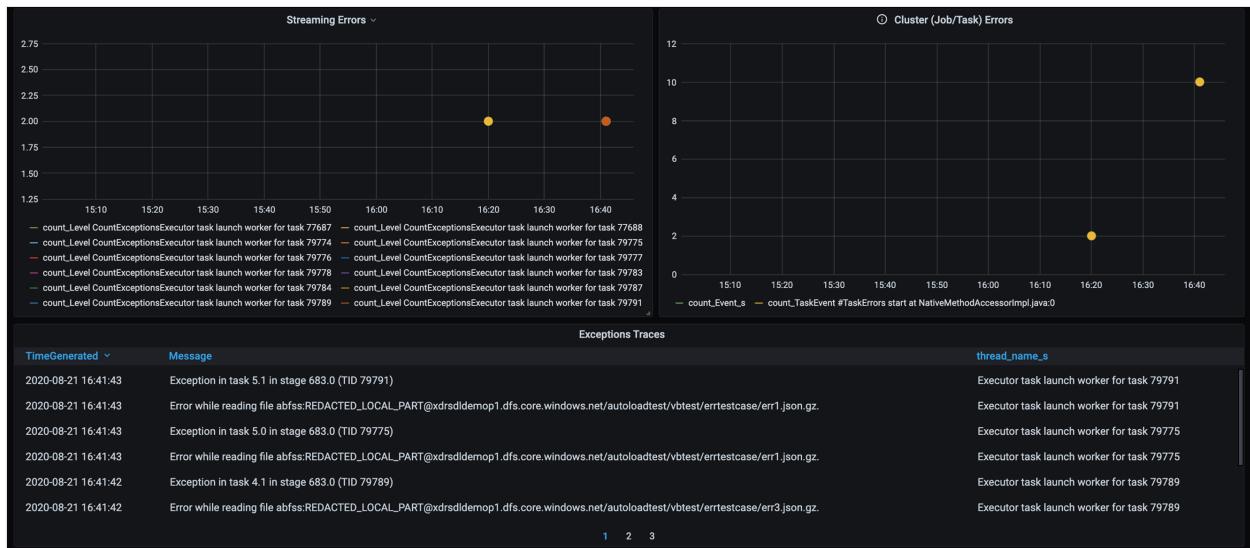
First, further identify the correct number of scaling executors that you need with Azure Databricks. Apply the rule of thumb of assigning each partition with a dedicated CPU in running executors. For instance, if you have 200 partition keys, the number of CPUs multiplied by the number of executors should equal 200. (For example, eight CPUs combined with 25 executors would be a good match.) With 200 partition keys, each executor can work only on one task, which reduces the chance of a bottleneck.

Because some slow partitions are in this scenario, investigate the high variance in tasks duration. Check for any spikes in task duration. One task handles one partition. If a task requires more time, the partition may be too large and cause a bottleneck.

Check Skew						
TimeGenerated	slice	max_TaskLatency	avg_TaskLatency	min_TaskLatency	max_Stage_Info_Number_of_Tasks_d	
2020-08-21 16:41:50	vb-cluster2-app-20200821054920-0...	2.8	0.71	0.012	200	
2020-08-21 16:41:35	vb-cluster2-app-20200821054920-0...	15	6.6	0.16	13	
2020-08-21 16:23:46	vb-cluster2-app-20200821054920-0...	28	5.8	0.012	200	
2020-08-21 16:23:35	vb-cluster2-app-20200821054920-0...	11	10	9.4	60	
2020-08-21 16:23:34	vb-cluster2-app-20200821054920-0...	0.057	0.030	0.014	60	
2020-08-21 16:23:34	vb-cluster2-app-20200821054920-0...	0.019	0.015	0.010	60	
2020-08-21 16:23:03	vb-cluster2-app-20200821054920-0...	30	7.8	0.0090	200	
2020-08-21 16:22:52	vb-cluster2-app-20200821054920-0...	12	11	9.7	100	
2020-08-21 16:22:52	vb-cluster2-app-20200821054920-0...	0.086	0.036	0.016	100	

Error tracing

Add a dashboard for error tracing so that you can spot customer-specific data failures. In data preprocessing, there are times when files are corrupted, and records within a file don't match the data schema. The following dashboard catches many bad files and bad records.



This dashboard displays the error count, error message, and task ID for debugging. In the message, you can easily trace the error back to the error file. There are several files in error while reading. You review the top timeline and investigate at the specific points in our graph (16:20 and 16:40).

Other bottlenecks

For more examples and guidance, see [Troubleshoot performance bottlenecks in Azure Databricks](#).

Performance tuning assessment summary

For this scenario, these metrics identified the following observations:

- In the stage latency chart, writing stages take most of the processing time.
- In the task latency chart, task latency is stable.
- In the streaming throughput chart, the output rate is lower than the input rate at some points.
- In the task's duration table, there's task variance because of imbalance of customer data.
- To get optimized performance in the partitioning stage, the number of scaling executors should match the number of partitions.
- There are tracing errors, such as bad files and bad records.

To diagnose these issues, you used the following metrics:

- Job latency
- Stage latency
- Task latency
- Streaming throughput
- Task duration (max, mean, min) per stage
- Error trace (count, message, task ID)

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [David McGhee](#) | Principal Program Manager

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- Read the [Log Analytics tutorial](#).
- [Monitoring Azure Databricks in an Azure Log Analytics workspace](#)
- [Deployment of Azure Log Analytics with Spark metrics](#)
- [Observability patterns](#)

Related resources

- [Send Azure Databricks application logs to Azure Monitor](#)
- [Use dashboards to visualize Azure Databricks metrics](#)

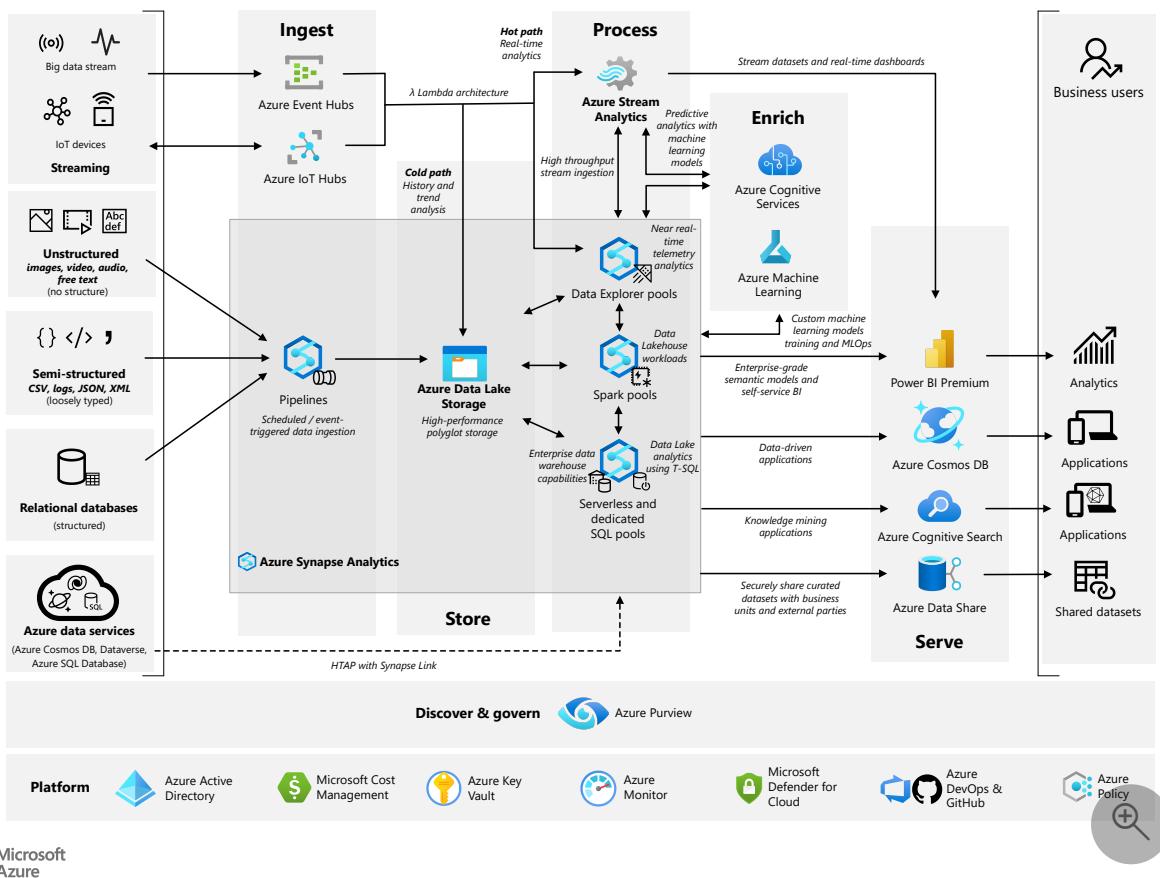
- Best practices for monitoring cloud applications
- Retry pattern

Analytics end-to-end with Azure Synapse

Azure Synapse Analytics Azure Cosmos DB Azure Data Factory Azure Databricks Azure Event Hubs

The solution described in this article combines a range of Azure services that will ingest, store, process, enrich, and serve data and insights from different sources (structured, semi-structured, unstructured, and streaming).

Architecture



Download a [Visio file](#) of this architecture.

ⓘ Note

- The services covered by this architecture are only a subset of a much larger family of Azure services. Similar outcomes can be achieved by using other services or features that are not covered by this design.

- Specific business requirements for your analytics use case could require the use of different services or features that are not considered in this design.

Dataflow

The analytics use cases covered by the architecture are illustrated by the different data sources on the left-hand side of the diagram. Data flows through the solution from the bottom up as follows:

Note

In the following sections, Azure Data Lake is used as the home for data throughout the various stages of the data lifecycle. Azure Data Lake is organized by different layers and containers as follows:

- The Raw layer is the landing area for data coming in from source systems. As the name implies, data in this layer is in raw, unfiltered, and unpurified form.
- In the next stage of the lifecycle, data moves to the Enriched layer where data is cleaned, filtered, and possibly transformed.
- Data then moves to the Curated layer, which is where consumer-ready data is maintained.

Please refer to the [Data lake zones and containers](#) documentation for a full review of Azure Data Lake layers and containers and their uses.

Azure data services, cloud native HTAP with Azure Cosmos DB and Dataverse

Process

1. [Azure Synapse Link for Azure Cosmos DB](#) and [Azure Synapse Link for Dataverse](#) enable you to run near real-time analytics over operational and business application data, by using the analytics engines that are available from your Azure Synapse workspace: [SQL Serverless](#) and [Spark Pools](#).
2. When using Azure Synapse Link for Azure Cosmos DB, use either a [SQL Serverless query](#) or a [Spark Pool notebook](#). You can access the [Azure Cosmos DB analytical store](#) and then combine datasets from your near real-time operational data with data from your data lake or from your data warehouse.

3. When using Azure Synapse Link for Dataverse, use either a [SQL Serverless query](#) or a [Spark Pool notebook](#). You can access the selected Dataverse tables and then combine datasets from your near real-time business applications data with data from your data lake or from your data warehouse.

Store

1. The resulting datasets from your [SQL Serverless queries](#) can be persisted in your data lake. If you are using [Spark notebooks](#), the resulting datasets can be persisted either in your data lake or data warehouse (SQL pool).

Serve

1. Load relevant data from the Azure Synapse SQL pool or data lake into [Power BI datasets](#) for data visualization and exploration. [Power BI models](#) implement a semantic model to simplify the analysis of business data and relationships. Business analysts use [Power BI](#) reports and dashboards to analyze data and derive business insights.
2. Data can also be securely shared to other business units or external trusted partners using [Azure Data Share](#). Data consumers have the freedom to choose what data format they want to use and also what compute engine is best to process the shared datasets.
3. Structured and unstructured data stored in your Synapse workspace can also be used to build [knowledge mining solutions](#) and use AI to uncover valuable business insights across different document types and formats including from Office documents, PDFs, images, audio, forms, and web pages.

Relational databases

Ingest

1. Use [Azure Synapse pipelines](#) to pull data from a wide variety of databases, both on-premises and in the cloud. Pipelines can be triggered based on a pre-defined schedule, in response to an event, or can be explicitly called via REST APIs.

Store

1. Within the Raw data lake layer, [organize your data lake](#) following the best practices around which layers to create, what folder structures to use in each layer and what

files format to use for each analytics scenario.

2. From the Azure Synapse pipeline, use a [Copy data activity](#) to stage the data copied from the relational databases into the [raw layer](#) of your [Azure Data Lake Store Gen 2](#) data lake. You can save the data in delimited text format or compressed as Parquet files.

Process

1. Use either [data flows](#), [SQL serverless queries](#), or [Spark notebooks](#) to validate, transform, and move the datasets from the Raw layer, through the Enriched layer and into your Curated layer in your data lake.
 - a. As part of your data transformations, you can invoke machine-training models from your [SQL pools using standard T-SQL](#) or Spark notebooks. These ML models can be used to enrich your datasets and generate further business insights. These machine-learning models can be consumed from [Azure Cognitive Services](#) or [custom ML models from Azure ML](#).

Serve

1. You can serve your final dataset directly from the data lake Curated layer or you can use Copy Data activity to ingest the final dataset into your SQL pool tables using the [COPY command](#) for fast ingestion.
2. Load relevant data from the Azure Synapse SQL pool or data lake into [Power BI datasets](#) for data visualization. [Power BI models](#) implement a semantic model to simplify the analysis of business data and relationships. Business analysts use [Power BI](#) reports and dashboards to analyze data and derive business insights.
3. Data can also be securely shared to other business units or external trusted partners using [Azure Data Share](#). Data consumers have the freedom to choose what data format they want to use and also what compute engine is best to process the shared datasets.
4. Structured and unstructured data stored in your Synapse workspace can also be used to build [knowledge mining solutions](#) and use AI to uncover valuable business insights across different document types and formats including from Office documents, PDFs, images, audio, forms, and web pages.

Semi-structured data sources

Ingest

1. Use [Azure Synapse pipelines](#) to pull data from a wide variety of semi-structured data sources, both on-premises and in the cloud. For example:

- Ingest data from file-based sources containing CSV or JSON files.
- Connect to No-SQL databases such as Azure Cosmos DB or MongoDB.
- Call REST APIs provided by SaaS applications that will function as your data source for the pipeline.

Store

1. Within the Raw data lake layer, [organize your data lake](#) following the best practices around which layers to create, what folder structures to use in each layer and what files format to use for each analytics scenario.
2. From the Azure Synapse pipeline, use a [Copy data activity](#) to stage the data copied from the semi-structured data sources into the [raw layer](#) of your [Azure Data Lake Store Gen 2](#) data lake. Save data to preserve the original format, as acquired from the data sources.

Process

1. For batch/micro-batch pipelines, use either [data flows](#), [SQL serverless queries](#) or [Spark notebooks](#) to validate, transform, and move your datasets into your Curated layer in your data lake. SQL Serverless queries expose underlying [CSV](#), [Parquet](#), or [JSON](#) files as external tables, so that they can be queried using T-SQL.
 - a. As part of your data transformations, you can invoke machine-learning models from your [SQL pools using standard T-SQL](#) or Spark notebooks. These ML models can be used to enrich your datasets and generate further business insights. These machine-learning models can be consumed from [Azure Cognitive Services](#) or [custom ML models from Azure ML](#).
2. For near real-time telemetry and time-series analytics scenarios, use [Data Explorer pools](#) to easily [ingest](#), consolidate, and correlate logs and IoT events data across multiple data sources. With Data Explorer pools, you can use [Kusto queries \(KQL\)](#) to perform [time-series analysis](#), [geospatial clustering](#), and machine learning enrichment.

Serve

1. You can serve your final dataset directly from the data lake Curated layer or you can use Copy Data activity to ingest the final dataset into your SQL pool tables using the [COPY command](#) for fast ingestion.
2. Load relevant data from the Azure Synapse [SQL pools](#), [Data Explorer pools](#), or a [data lake](#) into [Power BI datasets](#) for data visualization. [Power BI models](#) implement a semantic model to simplify the analysis of business data and relationships. Business analysts use [Power BI](#) reports and dashboards to analyze data and derive business insights.
3. Data can also be securely shared to other business units or external trusted partners using [Azure Data Share](#). Data consumers have the freedom to choose what data format they want to use and also what compute engine is best to process the shared datasets.
4. Structured and unstructured data stored in your Synapse workspace can also be used to build [knowledge mining solutions](#) and use AI to uncover valuable business insights across different document types and formats including from Office documents, PDFs, images, audio, forms, and web pages.

Non-structured data sources

Ingest

1. Use [Azure Synapse pipelines](#) to pull data from a wide variety of non-structured data sources, both on-premises and in the cloud. For example:
 - Ingest video, image, audio, or free text from file-based sources that contain the source files.
 - Call REST APIs provided by SaaS applications that will function as your data source for the pipeline.

Store

1. Within the Raw data lake layer, [organize your data lake](#) by following the best practices about which layers to create, what folder structures to use in each layer, and what files format to use for each analytics scenario.
2. From the Azure Synapse pipeline, use a [Copy data activity](#) to stage the data copied from the non-structured data sources into the [raw layer](#) of your [Azure Data Lake Store Gen 2](#) data lake. Save data by preserving the original format, as acquired from the data sources.

Process

1. Use [Spark notebooks](#) to validate, transform, enrich, and move your datasets from the Raw layer, through the Enriched layer and into your Curated layer in your data lake.
 - a. As part of your data transformations, you can invoke machine-learning models from your [SQL pools using standard T-SQL](#) or Spark notebooks. These ML models can be used to enrich your datasets and generate further business insights. These machine-learning models can be consumed from [Azure Cognitive Services](#) or [custom ML models from Azure ML](#).

Serve

1. You can serve your final dataset directly from the data lake Curated layer or you can use Copy Data activity to ingest the final dataset into your data warehouse tables using the [COPY command](#) for fast ingestion.
2. Load relevant data from the Azure Synapse SQL pool or data lake into [Power BI datasets](#) for data visualization. [Power BI models](#) implement a semantic model to simplify the analysis of business data and relationships.
3. Business analysts use [Power BI](#) reports and dashboards to analyze data and derive business insights.
4. Data can also be securely shared to other business units or external trusted partners using [Azure Data Share](#). Data consumers have the freedom to choose what data format they want to use and also what compute engine is best to process the shared datasets.
5. Structured and unstructured data stored in your Synapse workspace can also be used to build [knowledge mining solutions](#) and use AI to uncover valuable business insights across different document types and formats including from Office documents, PDFs, images, audio, forms, and web pages.

Streaming

Ingest

1. Use [Azure Event Hubs](#) or [Azure IoT Hubs](#) to ingest data streams generated by client applications or IoT devices. Event Hubs or IoT Hub will then ingest and store streaming data preserving the sequence of events received. Consumers can then connect to Event Hubs or IoT Hub endpoints and retrieve messages for processing.

Store

1. Within the Raw data lake layer, [organize your data lake](#) following the best practices around which layers to create, what folder structures to use in each layer and what files format to use for each analytics scenario.
2. Configure [Event Hubs Capture](#) or [IoT Hub Storage Endpoints](#) to save a copy of the events into the [Raw layer](#) of your [Azure Data Lake Store Gen 2](#) data lake. This feature implements the "Cold Path" of the [Lambda architecture pattern](#) and allows you to perform historical and trend analysis on the stream data saved in your data lake using [SQL Serverless queries](#) or [Spark notebooks](#) following the pattern for semi-structured data sources described above.

Process

1. For real-time insights, use a [Stream Analytics job](#) to implement the "Hot Path" of the [Lambda architecture pattern](#) and derive insights from the stream data in transit. Define at least one input for the data stream coming from your [Event Hubs](#) or [IoT Hub](#), one query to process the input data stream and one Power BI output to where the query results will be sent to.
 - a. As part of your data processing with Stream Analytics, you can invoke machine-learning models to enrich your stream datasets and drive business decisions based on the predictions generated. These machine-learning models can be consumed from Azure Cognitive Services or from [custom ML models in Azure Machine learning](#).
2. Use other Stream Analytics job outputs to send processed events to Azure Synapse [SQL pools](#) or [Data Explorer pools](#) for further analytics use cases.
3. For near real-time telemetry and time-series analytics scenarios, use [Data Explorer pools](#) to easily ingest IoT events directly from [Event Hubs](#) or [IoT Hubs](#). With Data Explorer pools, you can use [Kusto queries \(KQL\)](#) to perform [time-series analysis](#), [geospatial clustering](#), and machine learning enrichment.

Serve

1. Business analysts then use [Power BI real-time datasets and dashboard](#) capabilities to visualize the fast changing insights generated by your Stream Analytics query.
2. Data can also be securely shared to other business units or external trusted partners using [Azure Data Share](#). Data consumers have the freedom to choose

what data format they want to use and also what compute engine is best to process the shared datasets.

3. Structured and unstructured data stored in your Synapse workspace can also be used to build [knowledge mining solutions](#) and use AI to uncover valuable business insights across different document types and formats including from Office documents, PDFs, images, audio, forms and web pages.

Components

The following Azure services have been used in the architecture:

- [Azure Synapse Analytics](#)
- [Azure Data Lake Gen2](#)
- [Azure Cosmos DB](#)
- [Azure Cognitive Services](#)
- [Azure Machine Learning](#)
- [Azure Event Hubs](#)
- [Azure IoT Hub](#)
- [Azure Stream Analytics](#)
- [Microsoft Purview](#)
- [Azure Data Share](#)
- [Microsoft Power BI](#)
- [Microsoft Entra ID](#)
- [Azure Cost Management](#)
- [Azure Key Vault](#)
- [Azure Monitor](#)
- [Microsoft Defender for Cloud](#)
- [Azure DevOps](#)
- [Azure Policy](#)
- [GitHub](#)

Alternatives

- In the architecture above, Azure Synapse pipelines are responsible for data pipeline orchestration. [Azure Data Factory](#) pipelines also provide the same capabilities as described in this article.
- [Azure Databricks](#) can also be used as the compute engine used to process structured and unstructured data directly on the data lake.

- In the architecture above, Azure Stream Analytics is the service responsible for processing streaming data. Azure Synapse Spark pools and Azure Databricks can also be used to perform the same role through the execution of notebooks.
- [Azure HDInsight Kafka](#) clusters can also be used to ingest streaming data and provide the right level of performance and scalability required by large streaming workloads.
- You also can make use of [Azure Functions](#) to invoke Azure Cognitive Services or Azure Machine Learning custom ML models from an Azure Synapse pipeline.
- For comparisons of other alternatives, see:
 - [Choosing a data pipeline orchestration technology in Azure](#)
 - [Choosing a batch processing technology in Azure](#)
 - [Choosing an analytical data store in Azure](#)
 - [Choosing a data analytics technology in Azure](#)
 - [Choosing a stream processing technology in Azure](#)

Scenario details

This example scenario demonstrates how to use Azure Synapse Analytics with the extensive family of Azure Data Services to build a modern data platform that's capable of handling the most common data challenges in an organization.

Potential use cases

This approach can also be used to:

- Establish a [data product](#) architecture, which consists of a data warehouse for structured data and a data lake for semi-structured and unstructured data. You can choose to deploy a single data product for centralized environments or multiple data products for distributed environments such as Data Mesh. See more information about [Data Management and Data Landing Zones](#).
- Integrate relational data sources with other unstructured datasets, with the use of big data processing technologies.
- Use semantic modeling and powerful visualization tools for simpler data analysis.
- Share datasets within the organization or with trusted external partners.
- Implement knowledge mining solutions to extract valuable business information hidden in images, PDFs, documents, and so on.

Recommendations

Discover and govern

Data governance is a common challenge in large enterprise environments. On one hand, business analysts need to be able to discover and understand data assets that can help them solve business problems. On the other hand, Chief Data Officers want insights on privacy and security of business data.

Microsoft Purview

1. Use [Microsoft Purview for data discovery](#) and insights on your [data assets](#), [data classification](#), and [sensitivity](#), which covers the entire organizational data landscape.
2. Microsoft Purview can help you maintain a [business glossary](#) with the specific business terminology required for users to understand the semantics of what datasets mean and how they are meant to be used across the organization.
3. You can [register all your data sources](#) and organize them into [Collections](#), which also serves as a security boundary for your metadata.
4. Setup [regular scans](#) to automatically catalog and update relevant metadata about data assets in the organization. Microsoft Purview can also automatically add [data lineage](#) information based on information from Azure Data Factory or Azure Synapse pipelines.
5. [Data classification](#) and [data sensitivity](#) labels can be added automatically to your data assets based on pre-configured or customs rules applied during the regular scans.
6. Data governance professionals can use the reports and [insights](#) generated by Microsoft Purview to keep control over the entire data landscape and protect the organization against any security and privacy issues.

Platform services

In order to improve the quality of your Azure solutions, follow the recommendations and guidelines defined in the [Azure Well-Architected Framework](#) five pillars of architecture excellence: Cost Optimization, Operational Excellence, Performance Efficiency, Reliability, and Security.

Following these recommendations, the services below should be considered as part of the design:

1. [Microsoft Entra ID](#): identity services, single sign-on and multi-factor authentication across Azure workloads.
2. [Azure Cost Management](#): financial governance over your Azure workloads.
3. [Azure Key Vault](#): secure credential and certificate management. For example, [Azure Synapse Pipelines](#), [Azure Synapse Spark Pools](#) and [Azure ML](#) can retrieve credentials and certificates from Azure Key Vault used to securely access data stores.
4. [Azure Monitor](#): collect, analyze, and act on telemetry information of your Azure resources to proactively identify problems and maximize performance and reliability.
5. [Microsoft Defender for Cloud](#): strengthen and monitor the security posture of your Azure workloads.
6. [Azure DevOps](#) & [GitHub](#): implement DevOps practices to enforce automation and compliance to your workload development and deployment pipelines for Azure Synapse and Azure ML.
7. [Azure Policy](#): implement organizational standards and governance for resource consistency, regulatory compliance, security, cost, and management.

Considerations

These considerations implement the pillars of the Azure Well-Architected Framework, which is a set of guiding tenets that can be used to improve the quality of a workload. For more information, see [Microsoft Azure Well-Architected Framework](#).

The technologies in this architecture were chosen because each of them provides the necessary functionality to handle the most common data challenges in an organization. These services meet the requirements for scalability and availability, while helping them control costs. The services covered by this architecture are only a subset of a much larger family of Azure services. Similar outcomes can be achieved by using other services or features not covered by this design.

Specific business requirements for your analytics use cases may also ask for the use of different services or features not considered in this design.

Similar architecture can also be implemented for pre-production environments where you can develop and test your workloads. Consider the specific requirements for your workloads and the capabilities of each service for a cost-effective pre-production environment.

Cost optimization

Cost optimization is about looking at ways to reduce unnecessary expenses and improve operational efficiencies. For more information, see [Overview of the cost optimization pillar](#).

In general, use the [Azure pricing calculator](#) to estimate costs. The ideal individual pricing tier and the total overall cost of each service included in the architecture is dependent on the amount of data to be processed and stored and the acceptable performance level expected. Use the guide below to learn more about how each service is priced:

- [Azure Synapse Analytics](#) serverless architecture allows you to scale your compute and storage levels independently. Compute resources are charged based on usage, and you can scale or pause these resources on demand. Storage resources are billed per terabyte, so your costs will increase as you ingest more data.
- [Azure Data Lake Gen 2](#) is charged based on the amount of data stored and based on the number of transactions to read and write data.
- [Azure Event Hubs](#) and [Azure IoT Hubs](#) are charged based on the amount of compute resources required to process your message streams.
- [Azure Machine Learning](#) charges come from the amount of compute resources used to train and deploy your machine-learning models.
- [Cognitive Services](#) is charged based on the number of calls you make to the service APIs.
- [Microsoft Purview](#) is priced based on the number of data assets in the catalog and the amount of compute power required to scan them.
- [Azure Stream Analytics](#) is charged based on the amount of compute power required to process your stream queries.
- [Power BI](#) has different product options for different requirements. [Power BI Embedded](#) provides an Azure-based option for embedding Power BI functionality inside your applications. A Power BI Embedded instance is included in the pricing sample above.
- [Azure Cosmos DB](#) is priced based on the amount of storage and compute resources required by your databases.

Deploy this scenario

This deployment accelerator gives you the option to implement the entire reference architecture or choose what workloads you need for your analytics use case. You also have the option to select whether services are accessible via public endpoints or if they are to be accessed only via private endpoints.

Azure portal

Use the following button to deploy the reference using the Azure portal.



Deploy to Azure



For detailed information and additional deployment options, see the [deployment accelerator GitHub repo](#) with documentation and code used to define this solution.

Contributors

This article is being updated and maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Fabio Braga](#) | Principal MTC Technical Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- Review the guidelines defined in the [Azure data management and analytics scenario](#) for scalable analytics environment in Azure.
- Explore the [Data Engineer Learning Paths at Microsoft learn](#) for further training content and labs on the services involved in this reference architecture.
- Review the documentation and deploy the reference architecture using the [deployment accelerator available from GitHub](#).

Related resources

- [Big data analytics with enterprise-grade security using Azure Synapse](#)
- [High throughput stream ingestion to Azure Synapse](#)
- [Query a data lake or lakehouse by using Azure Synapse serverless](#)

Analyze operational data on MongoDB Atlas using Azure Synapse Analytics

Azure App Service Azure Data Lake Storage Azure Event Grid Azure Synapse Analytics Power BI

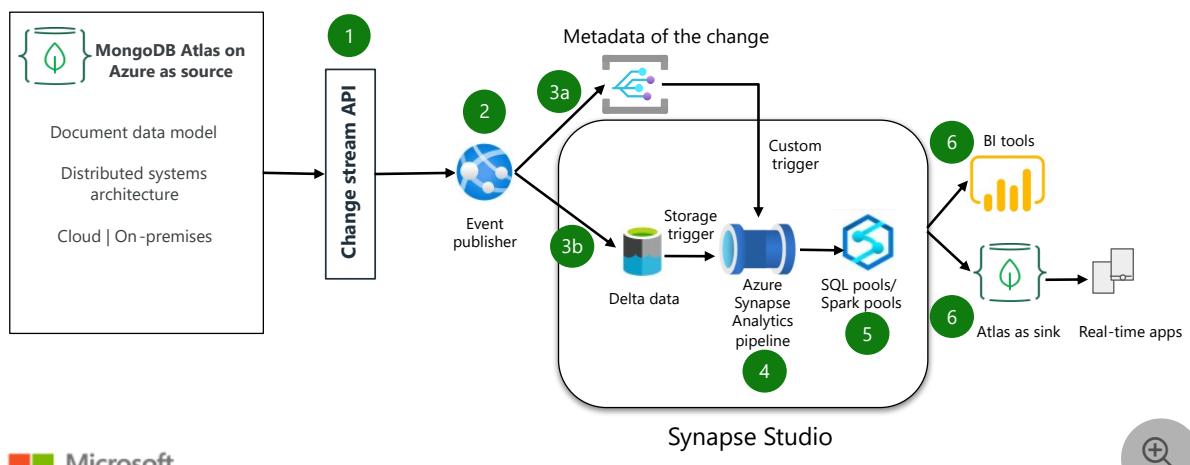
This article presents a solution for deriving insights from MongoDB Atlas operational data. The solution connects MongoDB Atlas to Azure Synapse Analytics. The connection makes it possible to transfer data in batches and in real time. The real-time approach keeps Azure Synapse Analytics dedicated SQL pools in sync with changes in the MongoDB Atlas data source.

Apache®, Apache Spark®, and the flame logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.

The MongoDB Atlas logo is a trademark of MongoDB. No endorsement is implied by the use of this mark.

Architecture

The following diagram shows how to sync MongoDB Atlas data to Azure Synapse Analytics in real time.



 Microsoft Azure



Download a [PowerPoint file](#) of all diagrams in this article.

Dataflow

The solution presents two options for triggering the pipelines that capture the real-time changes in the MongoDB Atlas operational data store (ODS) and sync the data. The following steps outline both options.

1. Changes occur in the operational and transactional data that's stored in MongoDB Atlas. The Mongo Atlas change stream APIs notify subscribed applications about the changes in real time.
2. A custom Azure App Service web app subscribes to the MongoDB change stream. There are two versions of the web app, *Event Grid* and *storage*, one for each version of the solution. Both app versions listen for changes that are caused by an insert, update, or delete operation in Atlas. When the apps detect a change, they write the changed document as a blob to Azure Data Lake Storage, which is integrated with Azure Synapse Analytics. The Event Grid version of the app also creates a new event in Azure Event Grid when it detects a change in Atlas.
3. Both versions of the solution trigger the Azure Synapse Analytics pipeline:
 - a. In the Event Grid version, a custom event-based trigger is configured in Azure Synapse Analytics. That trigger subscribes to the Event Grid topic that the web app publishes to. The new event on that topic activates the Azure Synapse Analytics trigger, which causes the Azure Synapse Analytics data pipeline to run.
 - b. In the storage version, a storage-based trigger is configured in Azure Synapse Analytics. When the new blob is detected in the integrated Data Lake Storage folder, that trigger is activated, which causes the Azure Synapse Analytics data pipeline to run.
4. In a copy activity, the Azure Synapse Analytics pipeline copies the full changed document from the Data Lake Storage blob to the dedicated SQL pool. This operation is configured to do an *upsert* on a selected column. If the column exists in the dedicated SQL pool, the upsert updates the column. If the column doesn't exist, the upsert inserts the column.
5. The dedicated SQL pool is the enterprise data warehousing feature that hosts the table that the data pipeline updates. The copy data activity of the pipeline keeps that table in sync with its corresponding Atlas collection.
6. Power BI reports and visualizations display current and near real-time analytics. The data also feeds into downstream applications. MongoDB Atlas functions as a sink by using an Azure Synapse Analytics data pipeline sink connector. Atlas then provides custom apps with the real-time data.

Components

- [MongoDB Atlas](#) is a database-as-a-service offering from MongoDB. This multicloud application data platform offers transactional processing, relevance-based search, real-time analytics, and mobile-to-cloud data synchronization. MongoDB also offers an on-premises solution, MongoDB Enterprise Advanced.
- [Change streams](#) in MongoDB Atlas give applications access to real-time data changes so that the apps can immediately react to those changes. The change streams provide a way for applications to receive notifications about changes to a particular collection, database, or entire deployment cluster.
- [App Service](#) and its Web Apps, Mobile Apps, and API Apps features provide a framework for building, deploying, and scaling web apps, mobile apps, and REST APIs. This solution uses web apps that are programmed in ASP.NET. The code is available on GitHub:
 - [Event Grid version](#)
 - [Storage version](#)
- [Azure Synapse Analytics](#) is the core service that this solution uses for data ingestion, processing, and analytics.
- [Data Lake Storage](#) provides capabilities for storing and processing data. As a data lake that's built on top of [Blob Storage](#), Data Lake Storage provides a scalable solution for managing large volumes of data from multiple, heterogeneous sources.
- [Azure Synapse Analytics pipelines](#) are used to perform extract, transform, and load (ETL) operations on data. Azure Data Factory provides a similar service, but you can create Azure Synapse Analytics pipelines within Synapse Studio. You can use multiple activities within the same pipeline. You can also create dependency endpoints to connect one activity with another activity in the pipeline.
- [Mapping data flows](#) are visually designed data transformations in Azure Synapse Analytics. Data flows provide a way for data engineers to develop data transformation logic without writing code. You can run the resulting data flows as activities within Azure Synapse Analytics pipelines that use scaled-out Apache Spark clusters. You can put data flow activities into operation by using existing Azure Synapse Analytics scheduling, control, flow, and monitoring capabilities.
- [Dedicated SQL pool](#) provides data warehousing capabilities for data after the data is processed and normalized. This feature of Azure Synapse Analytics was formerly

known as SQL Data Warehouse. Dedicated SQL pools make the refined data available to your end users and applications.

- [Azure Synapse Analytics triggers](#) provide an automated way to run pipelines. You can schedule these triggers. You can also set up event-based triggers, such as [storage event triggers](#) and [custom event triggers](#). The solution uses both types of event-based triggers.
- [Event Grid](#) is a highly scalable, serverless event broker. You can use Event Grid to deliver events to subscriber destinations.
- [Power BI](#) is a collection of software services and apps that display analytics information. In this solution, Power BI provides a way to use the processed data to perform advanced analysis and to derive insights.

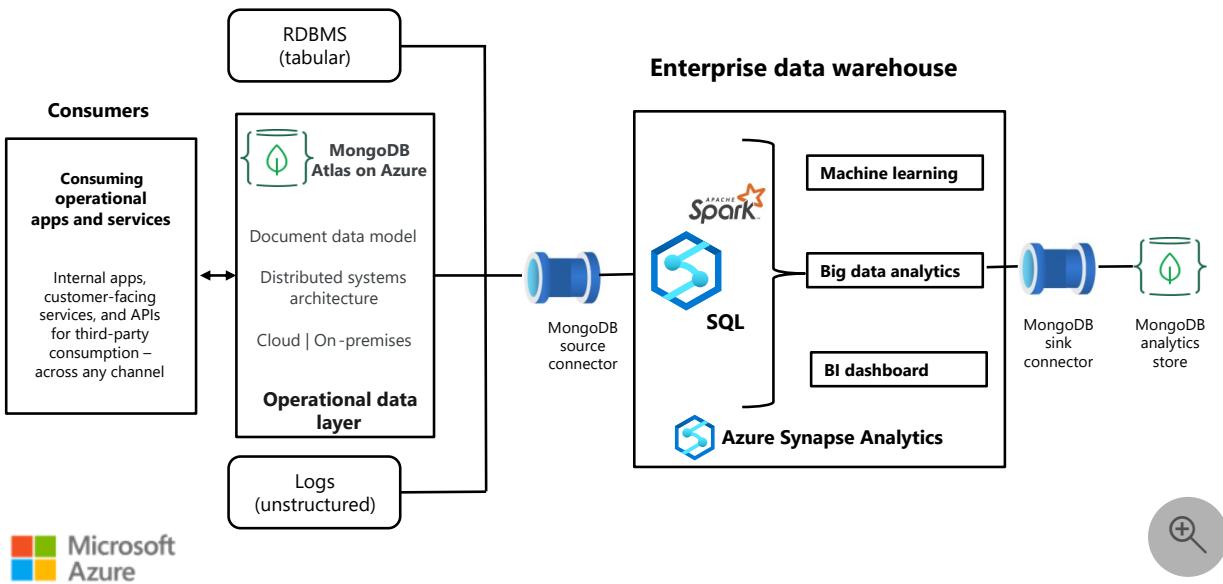
Scenario details

MongoDB Atlas serves as the operational data layer of many enterprise applications. This cloud database stores data from internal applications, customer-facing services, and third-party APIs from multiple channels. By using Azure Synapse Analytics pipelines, you can combine MongoDB Atlas data with relational data from other traditional applications and unstructured data from sources like logs.

Batch integration

In Azure Synapse Analytics, you can seamlessly integrate MongoDB on-premises instances and MongoDB Atlas as a source or sink resource. MongoDB is the only NoSQL database that has source and sink connectors for Azure Synapse Analytics and Data Factory.

With historical data, you can retrieve all the data at once. You can also retrieve data incrementally for specific periods by using a filter in batch mode. Then you can use SQL pools and Apache Spark pools in Azure Synapse Analytics to transform and analyze the data. If you need to store the analytics or query results in an analytics data store, you can use the sink resource in Azure Synapse Analytics.



For more information about how to set up and configure the connectors, see these resources:

- [Copy data from or to MongoDB Atlas using Azure Data Factory or Azure Synapse Analytics](#)
- [Copy data from or to MongoDB using Azure Data Factory or Azure Synapse Analytics](#)

The source connector provides a convenient way to run Azure Synapse Analytics on top of operational data that's stored in MongoDB or Atlas. After you use the source connector to retrieve data from Atlas, you can load the data into Data Lake Storage blob storage as a Parquet, Avro, JSON, text, or CSV file. You can then transform these files or join them with other files from other data sources in multi-database, multicloud, or hybrid cloud environments.

You can use the data that you retrieve from MongoDB Enterprise Advanced or MongoDB Atlas in the following scenarios:

- To retrieve all data from a particular date from MongoDB in a batch. You then load the data into Data Lake Storage. From there, you use a serverless SQL pool or Spark pool for analysis, or you copy the data into a dedicated SQL pool. After you retrieve this batch, you can apply changes to the data as they occur, as described in [Dataflow](#). A [Storage-CopyPipeline_mdb_synapse_ded_pool_RTS sample pipeline](#) is available as part of this solution. You can export the pipeline from GitHub for this one-time load purpose.
- To produce insights at a particular frequency, for instance, for a daily or hourly report. For this scenario, you schedule a pipeline to retrieve data on a regular basis before you run the analytics pipelines. You can use a MongoDB query to apply filter criteria and only retrieve a certain subset of data.

Real-time sync

Enterprises need insights that are based on real-time data, not stale data. A delay of a few hours in insight delivery can hold up the decision-making process and result in a loss of competitive advantage. This solution fuels critical decision making by propagating changes that occur in the MongoDB transactional database to the dedicated SQL pool in real time.

This solution has three parts, which the following sections describe.

Capture the MongoDB Atlas changes

The MongoDB change stream captures changes that occur in the database. The change stream APIs make information about changes available to App Service web apps that subscribe to the change stream. These apps write the changes to the Data Lake Storage blob storage.

Trigger a pipeline to propagate the changes to Azure Synapse Analytics

The solution presents two options for triggering an Azure Synapse Analytics pipeline after the blob is written to Data Lake Storage:

- A storage-based trigger. Use this option if you need real-time analytics, because the pipeline gets triggered as soon as the blob with the change is written. But this option might not be the preferred approach when you have a high volume of data changes. Azure Synapse Analytics limits the number of pipelines that can run concurrently. When you have a large number of data changes, you might hit that limit.
- An event-based custom trigger. This type of trigger has the advantage that it's outside Azure Synapse Analytics, so it's easier to control. The Event Grid version of the web app writes the changed data document to the blob storage. At the same time, the app creates a new Event Grid event. The data in the event contains the file name of the blob. The pipeline that the event triggers receives the file name as a parameter and then uses the file to update the dedicated SQL pool.

Propagate the changes to a dedicated SQL pool

An Azure Synapse Analytics pipeline propagates the changes to a dedicated SQL pool. The solution provides a *CopyPipeline_mdb_synapse_ded_pool_RTS* pipeline on GitHub

that copies the change in the blob from Data Lake Storage to the dedicated SQL pool. This pipeline is triggered by either the storage or Event Grid trigger.

Potential use cases

The use cases for this solution span many industries and areas:

- Retail
 - Building intelligence into product bundling and product promotion
 - Optimizing cold storage that uses IoT streaming
 - Optimizing inventory replenishment
 - Adding value to omnichannel distribution
- Banking and finance
 - Customizing customer financial services
 - Detecting potentially fraudulent transactions
- Telecommunications
 - Optimizing next-generation networks
 - Maximizing the value of edge networks
- Automotive
 - Optimizing parameterization of connected vehicles
 - Detecting anomalies in IoT communication in connected vehicles
- Manufacturing
 - Providing predictive maintenance for machinery
 - Optimizing storage and inventory management

Here are two specific examples:

- As this article describes earlier in [Batch integration](#), you can retrieve MongoDB data in a batch and then update the data as changes occur. This capability makes real-time insights possible for just-in-time decision making and conclusions. This functionality is useful for analytics of sensitive and critical information such as financial transactions and fraud detection data.
- As [Batch integration](#) also describes, you can schedule a pipeline to retrieve MongoDB data on a regular basis. This functionality is useful in retail scenarios such as updating inventory levels with daily sales data. In such cases, analytics reports and dashboards aren't of critical importance, and real-time analysis isn't worth the effort.

The following sections take a closer look at two retail industry use cases.

Product bundling

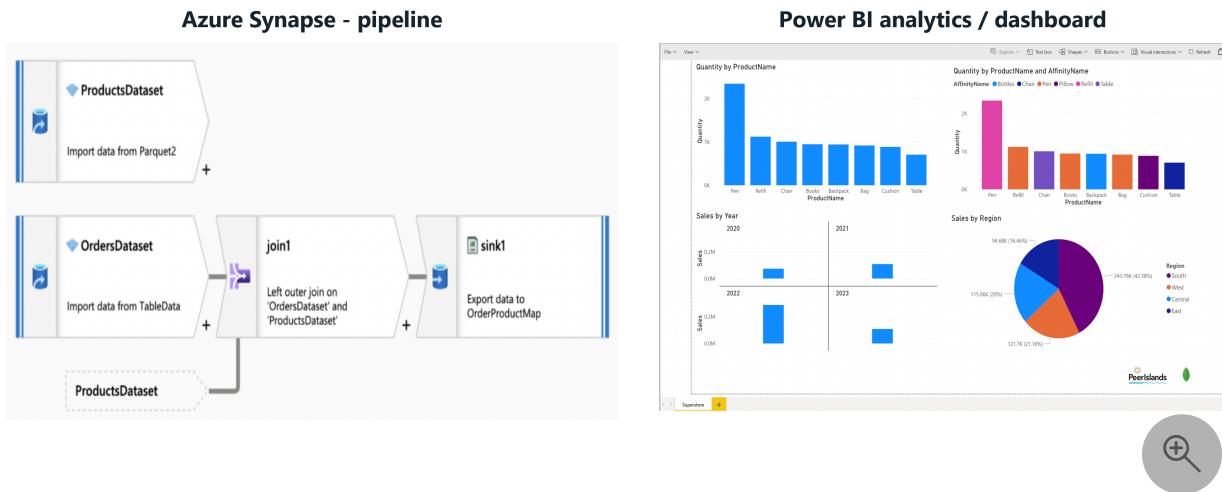
To promote the sale of a product, you can sell the product as part of a bundle together with other related products. The objective is to use sales pattern data to develop strategies for bundling a product into packages.

There are two sources of data:

- The product catalog data from MongoDB
- Sales data from Azure SQL

Both sets of data are migrated to an Azure Synapse Analytics dedicated SQL pool by using an Azure Synapse Analytics pipeline. Triggers and change data captures are used to achieve a near real-time data sync on top of the one-time migrated data.

The following Power BI charts show the affinity between the products and sales patterns. The affinity of the pen and ink-based refill is high. The sales data shows that the pen has a high sales volume in the specified area.



The analysis makes two suggestions for yielding better sales:

- Bundling the pen and ink-based refill
- Promoting the bundle in certain areas

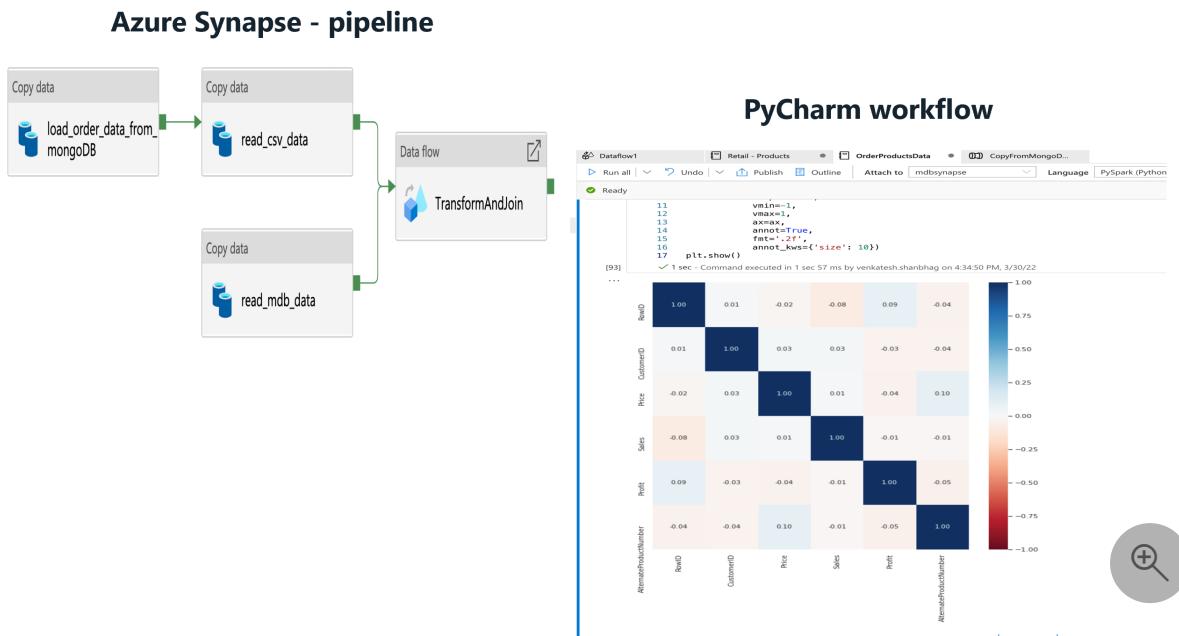
Product promotion

To promote the sale of a product, you can recommend the product to customers who are interested in related products. The objective is to use sales data and customer

buying pattern data to develop strategies for recommending a product to customers.

By using Azure Synapse Analytics, you can develop AI and machine learning models to determine which products to recommend to customers.

The following diagrams show the use of various types of data to create a model to determine alternate product recommendations. The data includes customer buying patterns, profits, product affinities, the sales volume of the products, and product catalog parameters.



If your model achieves high accuracy, it provides a list of products that you can recommend to the customer.

Considerations

These considerations implement the pillars of the Azure Well-Architected Framework, which is a set of guiding tenets that can be used to improve the quality of a workload. For more information, see [Microsoft Azure Well-Architected Framework](#).

Security

Security provides assurances against deliberate attacks and the abuse of your valuable data and systems. For more information, see [Overview of the security pillar](#).

For detailed information about the security requirements and controls of the Azure components in the solution, see the security section of each product's documentation.

Cost optimization

Cost optimization is about looking at ways to reduce unnecessary expenses and improve operational efficiencies. For more information, see [Overview of the cost optimization pillar](#).

- To estimate the cost of Azure products and configurations, use the [Azure pricing calculator](#).
- Azure helps you avoid unnecessary costs by identifying the correct number of resources for your needs, by analyzing spending over time, and by scaling to meet business needs without overspending. For example, you can pause the dedicated SQL pools when you don't expect any load. You can resume them later.
- You can replace App Service with Azure Functions. By orchestrating the functions within an Azure Synapse Analytics pipeline, you can reduce costs.
- To reduce the Spark cluster cost, choose the right data flow compute type. General and memory-optimized options are available. Also choose appropriate core count and time-to-live (TTL) values.
- To find out more about managing the costs of key solution components, see these resources:
 - [Plan and manage costs for Azure Synapse Analytics](#)
 - [Plan to manage costs for Azure Data Factory](#)

Performance efficiency

Performance efficiency is the ability of your workload to scale to meet the demands that are placed on it by users in an efficient manner. For more information, see [Performance efficiency pillar overview](#).

When there's a high volume of changes, running thousands of pipelines in Azure Synapse Analytics for every change in the collection can result in a backlog of queued pipelines. To improve performance in this scenario, consider the following approaches:

- Use the storage-based App Service code, which writes the JSON documents with the changes to Data Lake Storage. Don't link the storage-based trigger with the pipeline. Instead, use a scheduled trigger at a short interval, such as every two or five minutes. When the scheduled trigger runs, it takes all the files in the specified Data Lake Storage directory and updates the dedicated SQL pool for each of them.
- Modify the Event Grid App Service code. Program it to add a micro-batch of around 100 changes to the blob storage before it adds the new topic to the event with the metadata that includes the filename. With this modification, you trigger only one pipeline for one blob with the 100 changes. You can adjust the micro-batch size to suit your scenario. Use small micro-batches at a high frequency to

provide updates that are close to real time. Or use larger micro-batches at a lower frequency for delayed updates and reduced overhead.

For more information on improving the performance and scalability of Azure Synapse Analytics pipeline copy activity, see [Copy activity performance and scalability guide](#).

Deploy this scenario

For information about implementing this solution, see [Real-Time Sync Solution for MongoDB Atlas Integration with Synapse](#).

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- [Diana Annie Jenosh](#) | Senior Solutions Architect
- [Babu Srinivasan](#) | Senior Solutions Architect
- [Utsav Talwar](#) | Associate Solutions Architect

Other contributors:

- [Krishnakumar Rukmangathan](#) | Senior Program Manager
- [Sunil Sabat](#) | Principal Program Manager
- [Wee Hyong T.](#) | Principal Director
- [Paresh Saraf](#) | Technical Director

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

For more information about the solution, contact partners@mongodb.com.

For information about MongoDB, see these resources:

- [MongoDB](#)
- [MongoDB Atlas](#)
- [MongoDB horizontal use cases](#)
- [MongoDB industry-specific use cases](#)

For information about Azure solution components, see these resources:

- [What is Azure Synapse Analytics?](#)
- [Azure Synapse Analytics use cases ↗](#)
- [Azure Synapse Analytics industry-specific use cases ↗](#)
- [Azure Synapse Analytics connectors](#)
- [App Service overview](#)
- [What is Power BI? ↗](#)
- [Introduction to Azure Data Lake Storage Gen2](#)
- [What is Azure Event Grid?](#)

Related resources

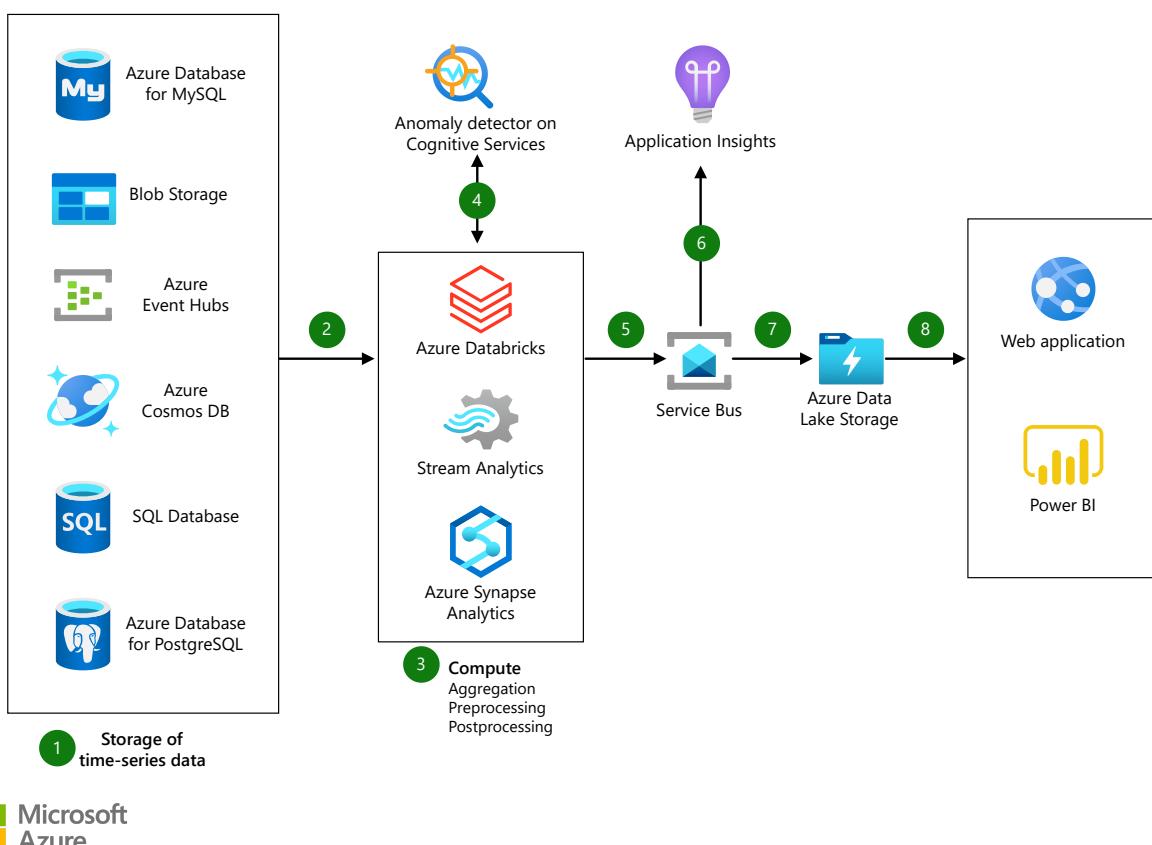
- [Enterprise business intelligence](#)
- [Automated enterprise BI](#)
- [Enterprise data warehouse](#)
- [Real-time analytics on big data architecture](#)
- [Use a speech-to-text transcription pipeline to analyze recorded conversations](#)

Anomaly detector process

Azure Databricks Azure Service Bus Azure Storage Accounts

This article presents an architecture for a near real-time implementation of an anomaly detection process.

Architecture



Download a [Visio file](#) of this architecture.

Dataflow

1. Time-series data can come from multiple sources, such as Azure Database for MySQL, Blob storage, Event Hubs, Azure Cosmos DB, SQL Database, and Azure Database for PostgreSQL.
2. Data is ingested into compute from various storage sources to be monitored by Anomaly Detector.
3. Databricks helps aggregate, sample, and compute the raw data to generate the time with the detected results. Databricks is capable of processing stream and

static data. Stream analytics and Azure Synapse can be alternatives based on the requirements.

4. The anomaly detector API detects anomalies and returns the results to compute.
5. The anomaly-related metadata is queued.
6. Application Insights picks the message from the message queue based on the anomaly-related metadata and sends an alert about the anomaly.
7. The results are stored in Azure Data Lake Service Gen2.
8. Web applications and Power BI can visualize the results of the anomaly detection.

Components

Key technologies used to implement this architecture:

- [Service Bus](#): Reliable cloud messaging as a service (MaaS) and simple hybrid integration.
- [Azure Databricks](#): Fast, easy, and collaborative Apache Spark-based analytics service.
- [Power BI](#): Interactive data visualization BI tools.
- [Storage Accounts](#): Durable, highly available, and massively scalable cloud storage.
- [Cognitive Services](#): Cloud-based services with REST APIs and client library SDKs available to help you build cognitive intelligence into your applications.
- [Logic Apps](#): Serverless platform for building enterprise workflows that integrate applications, data, and services. In this architecture, the logic apps are triggered by HTTP requests.
- [Azure Data Lake Storage Gen2](#): Azure Data Lake Storage Gen2 provides file system semantics, file-level security, and scale.
- [Application Insights](#): Application Insights is a feature of Azure Monitor that provides extensible application performance management (APM) and monitoring for live web apps.

Alternatives

- [Event Hubs with Kafka](#): An alternative to running your own Kafka cluster. This Event Hubs feature provides an endpoint that is compatible with Kafka APIs.
- [Azure Synapse Analytics](#): An analytics service that brings together enterprise data warehousing and big data analytics.
- [Azure Machine Learning](#): Build, train, deploy, and manage custom machine learning / anomaly detection models in a cloud-based environment.

Scenario details

The Azure Cognitive Services Anomaly Detector API enables you to monitor and detect abnormalities in your time series data without having to know machine learning. The algorithms of the API adapt by automatically identifying and applying the best-fitting models to your time series data, regardless of industry, scenario, or data volume. They determine boundaries for anomaly detection, expected values, and anomalous data points.

Potential use cases

Some areas that anomaly detection helps monitor:

- Bank fraud (finance industry)
- Structural defects (manufacturing industry)
- Medical problems (healthcare industry)

Considerations

These considerations implement the pillars of the Azure Well-Architected Framework, which is a set of guiding tenets that can be used to improve the quality of a workload. For more information, see [Microsoft Azure Well-Architected Framework](#).

Scalability

Most of the components used in this example scenario are managed services that will automatically scale.

For general guidance on designing scalable solutions, see the [performance efficiency checklist](#) in the Azure Architecture Center.

Security

Security provides assurances against deliberate attacks and the abuse of your valuable data and systems. For more information, see [Overview of the security pillar](#).

[Managed identities for Azure resources](#) are used to provide access to other resources internal to your account and then assigned to your Azure Functions. Allow those identities to access only requisite resources to ensure that nothing extra is exposed to your functions (and potentially to your customers).

For general guidance on designing secure solutions, see the [Azure Security Documentation](#).

Resiliency

All of the components in this scenario are managed, so at a regional level they're all resilient automatically.

For general guidance on designing resilient solutions, see [Designing resilient applications for Azure](#).

Cost optimization

Cost optimization is about looking at ways to reduce unnecessary expenses and improve operational efficiencies. For more information, see [Overview of the cost optimization pillar](#).

To explore the cost of running this scenario, see the pre-filled calculator with all of the services. To see how the pricing would change for your particular use case, change the appropriate variables to match your expected traffic / data volumes.

We've provided three sample cost profiles based on the amount of traffic (we assume all images are 100 kb in size):

- [Example calculator](#): this pricing example is a calculator with all services in this architecture, except Power BI and custom alerting solution.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Ashish Chauhan](#) | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- [Anomaly Detector API Documentation](#)
- [Interactive demo](#)

- Detect and visualize anomalies in your data with the Anomaly Detector API - Demo on Jupyter Notebook ↗
- Identify anomalies by routing data via IoT Hub to a built-in ML model in Azure Stream Analytics
- Recipe: Predictive maintenance with the Cognitive Services for Big Data
- Service Bus Documentation
- Azure Databricks Documentation
- Power BI Documentation
- Storage Documentation

Related resources

- Quality assurance
- Supply chain track and trace
- Introduction to predictive maintenance in manufacturing
- Predictive maintenance solution
- Predictive maintenance with the intelligent IoT Edge
- Stream processing with fully managed open-source data engines
- Connected factory hierarchy service
- Connected factory signal pipeline

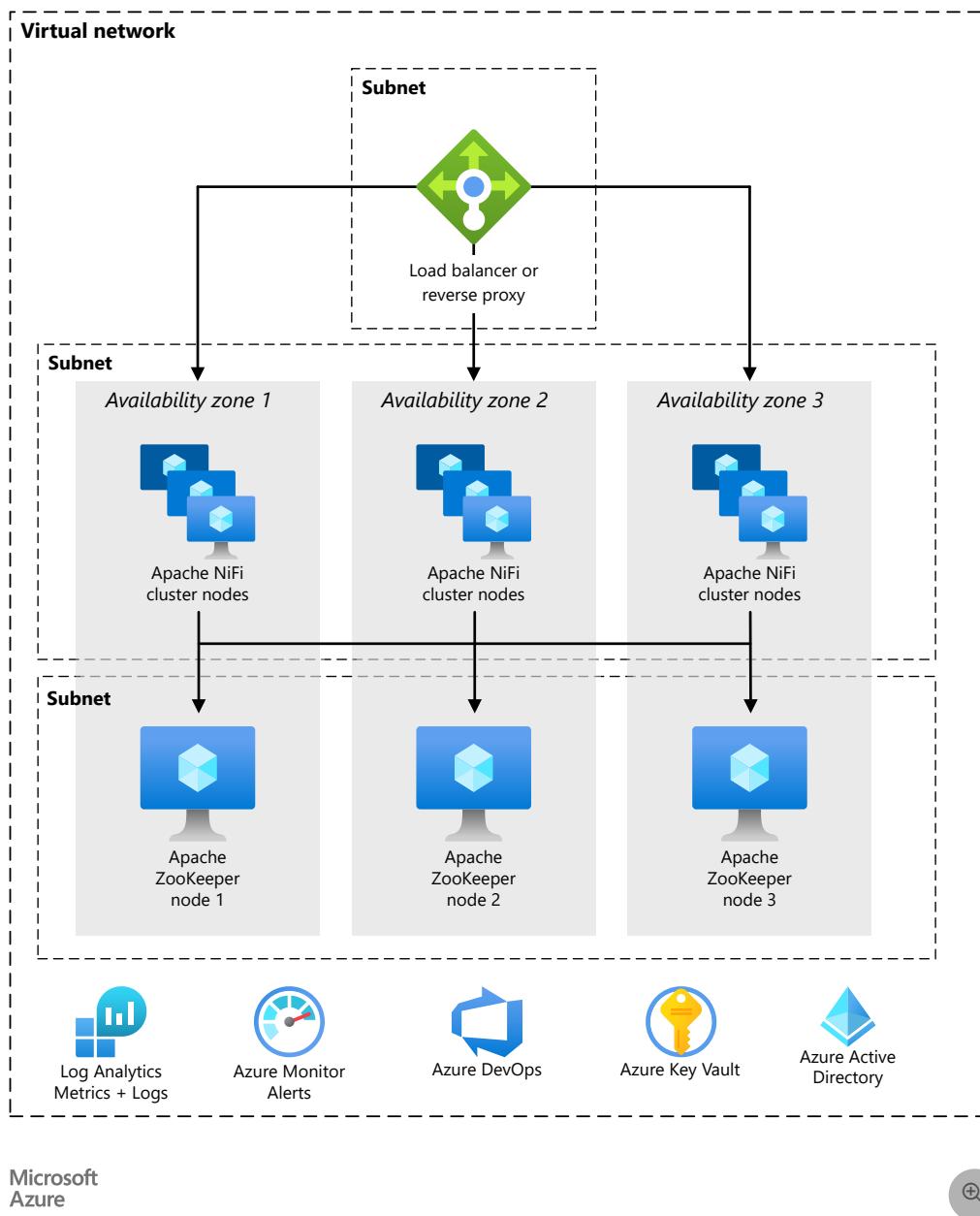
Apache NiFi on Azure

Azure Application Gateway | Azure Log Analytics | Azure Virtual Machines | Azure Virtual Network | Azure Virtual Machine Scale Sets

This example scenario shows how to run [Apache NiFi](#) on Azure. NiFi provides a system for processing and distributing data.

Apache®, Apache NiFi®, and NiFi® are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.

Architecture



Download a [Visio file](#) of this architecture.

Workflow

- The NiFi application runs on VMs in NiFi cluster nodes. The VMs are in a virtual machine scale set that the configuration deploys across availability zones.
- Apache ZooKeeper runs on VMs in a separate cluster. NiFi uses the ZooKeeper cluster for these purposes:
 - To choose a cluster coordinator node

- To coordinate the flow of data
- Azure Application Gateway provides layer-7 load balancing for the user interface that runs on the NiFi nodes.
- Monitor and its Log Analytics feature collect, analyze, and act on telemetry from the NiFi system. The telemetry includes NiFi system logs, system health metrics, and performance metrics.
- Azure Key Vault securely stores certificates and keys for the NiFi cluster.
- Microsoft Entra ID provides single sign-on and multifactor authentication.

Components

- [NiFi](#) provides a system for processing and distributing data.
- [ZooKeeper](#) is an open-source server that manages distributed systems.
- [Virtual Machines](#) is an infrastructure-as-a-service (IaaS) offer. You can use Virtual Machines to deploy on-demand, scalable computing resources. Virtual Machines provides the flexibility of virtualization but eliminates the maintenance demands of physical hardware.
- [Azure Virtual Machine Scale Sets](#) provide a way to manage a group of load-balanced VMs. The number of VM instances in a set can automatically increase or decrease in response to demand or a defined schedule.
- [Availability zones](#) are unique physical locations within an Azure region. These high-availability offerings protect applications and data from datacenter failures.
- [Application Gateway](#) is a load balancer that manages traffic to web applications.
- [Monitor](#) collects and analyzes data on environments and Azure resources. This data includes app telemetry, such as performance metrics and activity logs. For more information, see [Monitoring considerations](#) later in this article.
- [Log Analytics](#) is an Azure portal tool that runs queries on Monitor log data. Log Analytics also provides features for charting and statistically analyzing query results.
- [Azure DevOps Services](#) provides services, tools, and environments for managing coding projects and deployments.
- [Key Vault](#) securely stores and controls access to a system's secrets, such as API keys, passwords, certificates, and cryptographic keys.
- [Microsoft Entra ID](#) is a cloud-based identity service that controls access to Azure and other cloud apps.

Alternatives

- [Azure Data Factory](#) provides an alternative to this solution.
- Instead of Key Vault, you can use a comparable service to store system secrets.
- [Apache Airflow](#). See [how Airflow and NiFi are different](#).
- It is possible to use a supported enterprise NiFi alternative like [Cloudera Apache NiFi](#). The Cloudera offering is available through the [Azure Marketplace](#).

Scenario details

In this scenario, NiFi runs in a clustered configuration across Azure Virtual Machines in a scale set. But most of this article's recommendations also apply to scenarios that run NiFi in single-instance mode on a single virtual machine (VM). The best practices in this article demonstrate a scalable, high-availability, and secure deployment.

Potential use cases

NiFi works well for moving data and managing the flow of data:

- Connecting decoupled systems in the cloud
- Moving data in and out of Azure Storage and other data stores
- Integrating edge-to-cloud and hybrid-cloud applications with Azure IoT, Azure Stack, and Azure Kubernetes Service (AKS)

As a result, this solution applies to many areas:

- Modern data warehouses (MDWs) bring structured and unstructured data together at scale. They collect and store data from various sources, sinks, and formats. NiFi excels at ingesting data into Azure-based MDWs for the following reasons:
 - Over 200 processors are available for reading, writing, and manipulating data.
 - The system supports Storage services such as Azure Blob Storage, Azure Data Lake Storage, Azure Event Hubs, Azure Queue Storage, Azure Cosmos DB, and Azure Synapse Analytics.
 - Robust data provenance capabilities make it possible to implement compliant solutions. For information about capturing data provenance in the Log Analytics feature of Azure Monitor, see [Reporting considerations](#) later in this article.

- NiFi can run on a standalone basis on small-footprint devices. In such cases, NiFi makes it possible to process edge data and move that data to larger NiFi instances or clusters in the cloud. NiFi helps filter, transform, and prioritize edge data in motion, ensuring reliable and efficient data flows.
- Industrial IoT (IIoT) solutions manage the flow of data from the edge to the data center. That flow starts with data acquisition from industrial control systems and equipment. The data then moves to data management solutions and MDWs. NiFi offers capabilities that make it well suited for data acquisition and movement:
 - Edge data processing functionality
 - Support for protocols that IoT gateways and devices use
 - Integration with Event Hubs and Storage services

IoT applications in the areas of predictive maintenance and supply chain management can make use of this functionality.

Recommendations

Keep the following points in mind when you use this solution:

Recommended versions of NiFi

When you run this solution on Azure, we recommend using version 1.13.2+ of NiFi. You can run other versions, but they might require different configurations from the ones in this guide.

To install NiFi on Azure VMs, it's best to download the convenience binaries from the [NiFi downloads page](#). You can also build the binaries from [source code](#).

Recommended versions of ZooKeeper

For this example workload, we recommend using versions 3.5.5 and later or 3.6.x of ZooKeeper.

You can install ZooKeeper on Azure VMs by using official convenience binaries or source code. Both are available on the [Apache ZooKeeper releases page](#).

Considerations

These considerations implement the pillars of the Azure Well-Architected Framework, which is a set of guiding tenets that can be used to improve the quality of a workload. For more information, see [Microsoft Azure Well-Architected Framework](#).

For information on configuring NiFi, see the [Apache NiFi System Administrator's Guide](#). Also keep these considerations in mind when you implement this solution.

Cost optimization

Cost optimization is about looking at ways to reduce unnecessary expenses and improve operational efficiencies. For more information, see [Overview of the cost optimization pillar](#).

- Use the [Azure Pricing Calculator](#) to estimate the cost of the resources in this architecture.
- For an estimate that includes all the services in this architecture except the custom alerting solution, see this [sample cost profile](#).

VM considerations

The following sections provide a detailed outline of how to configure the NiFi VMs:

VM size

This table lists recommended VM sizes to start with. For most general-purpose data flows, Standard_D16s_v3 is best. But each data flow in NiFi has different requirements. Test your flow and resize as needed based on the flow's actual requirements.

Consider enabling accelerated networking on the VMs to increase network performance. For more information, see [Networking for Azure virtual machine scale sets](#).

VM size	vCPU	Memory in GB	Max uncached data disk throughput in I/O operations per second (IOPS) per MBps*	Max network interfaces (NICs) / Expected network bandwidth (Mbps)
Standard_D8s_v3	8	32	12,800/192	4/4,000
Standard_D16s_v3**	16	64	25,600/384	8/8,000
Standard_D32s_v3	32	128	51,200/768	8/16,000
Standard_M16m	16	437.5	10,000/250	8/4,000

* Disable data disk write caching for all data disks that you use on NiFi nodes.

** We recommend this SKU for most general-purpose data flows. Azure VM SKUs with similar vCPU and memory configurations should also be adequate.

VM operating system (OS)

We recommend running NiFi in Azure on one of the following guest operating systems:

- Ubuntu 18.04 LTS or later
- CentOS 7.9

To meet the specific requirements of your data flow, it's important to adjust several OS-level settings including:

- Maximum forked processes.
- Maximum file handles.
- The access time, `atime`.

After you adjust the OS to fit your expected use case, use Azure VM Image Builder to codify the generation of those tuned images. For guidance that's specific to NiFi, see [Configuration Best Practices](#) in the Apache NiFi System Administrator's Guide.

Storage

Store the various NiFi repositories on data disks and not on the OS disk for three main reasons:

- Flows often have high disk throughput requirements that a single disk can't meet.
- It's best to separate the NiFi disk operations from the OS disk operations.
- The repositories shouldn't be on temporary storage.

The following sections outline guidelines for configuring the data disks. These guidelines are specific to Azure. For more information on configuring the repositories, see [State Management](#) in the Apache NiFi System Administrator's Guide.

Data disk type and size

Consider these factors when you configure the data disks for NiFi:

- Disk type
- Disk size
- Total number of disks

Note

For up-to-date information on disk types, sizing, and pricing, see [Introduction to Azure managed disks](#).

The following table shows the types of managed disks that are currently available in Azure. You can use NiFi with any of these disk types. But for high-throughput data flows, we recommend Premium SSD.

 [Expand table](#)

	Ultra Disk (NVMe)	Premium SSD	Standard SSD	Standard HDD
Disk type	SSD	SSD	SSD	HDD
Max disk size	65,536 GB	32,767 GB	32,767 GB	32,767 GB
Max throughput	2,000 MiB/s	900 MiB/s	750 MiB/s	500 MiB/s
Max IOPS	160,000	20,000	6,000	2,000

Use at least three data disks to increase throughput of the data flow. For best practices for configuring the repositories on the disks, see [Repository configuration](#) later in this article.

The following table lists the relevant size and throughput numbers for each disk size and type.

[Expand table](#)

	Standard HDD S15	Standard HDD S20	Standard HDD S30	Standard SSD S15	Standard SSD S20	Standard SSD S30	Premium SSD P15	Premium SSD P20	Premium SSD P30
Disk size in GB	256	512	1,024	256	512	1,024	256	512	1,024
IOPS per disk	Up to 500	1,100	2,300	5,000					
Throughput per disk	Up to 60 MBps	125 MBps	150 MBps	200 MBps					

If your system hits VM limits, adding more disks might not increase throughput:

- IOPS and throughput limits depend on the size of the disk.
- The VM size that you choose places IOPS and throughput limits for the VM on all data disks.

For VM-level disk throughput limits, see [Sizes for Linux virtual machines in Azure](#).

VM disk caching

On Azure VMs, the Host Caching feature manages disk write caching. To increase throughput in data disks that you use for repositories, turn off disk write caching by setting Host Caching to `None`.

Repository configuration

The best practice guidelines for NiFi are to use a separate disk or disks for each of these repositories:

- Content
- FlowFile
- Provenance

This approach requires a minimum of three disks.

NiFi also supports application-level striping. This functionality increases the size or performance of the data repositories.

The following excerpt is from the `nifi.properties` configuration file. This configuration partitions and stripes the repositories across managed disks that are attached to the VMs:

config

```
nifi.provenance.repository.directory.stripes1=/mnt/disk1/ provenance_repository
nifi.provenance.repository.directory.stripes2=/mnt/disk2/ provenance_repository
nifi.provenance.repository.directory.stripes3=/mnt/disk3/ provenance_repository
nifi.content.repository.directory.stripes1=/mnt/disk4/ content_repository
```

```
nifi.content.repository.directory.stripe2=/mnt/disk5/ content_repository  
nifi.content.repository.directory.stripe3=/mnt/disk6/ content_repository  
nifi.flowfile.repository.directory=/mnt/disk7/ flowfile_repository
```

For more information about designing for high-performance storage, see [Azure premium storage: design for high performance](#).

Reporting

NiFi includes a provenance reporting task for the [Log Analytics](#) feature.

You can use this reporting task to offload provenance events to cost-effective, durable long-term storage. The Log Analytics feature provides a [query interface](#) for viewing and graphing the individual events. For more information on these queries, see [Log Analytics queries](#) later in this article.

You can also use this task with volatile, in-memory provenance storage. In many scenarios, you can then achieve a throughput increase. But this approach is risky if you need to preserve event data. Ensure that volatile storage meets your durability requirements for provenance events. For more information, see [Provenance Repository](#) in the Apache NiFi System Administrator's Guide.

Before using this process, create a log analytics workspace in your Azure subscription. It's best to set up the workspace in the same region as your workload.

To configure the provenance reporting task:

1. Open the controller settings in NiFi.
2. Select the reporting tasks menu.
3. Select **Create a new reporting task**.
4. Select **Azure Log Analytics Reporting Task**.

The following screenshot shows the properties menu for this reporting task:

Property	Value
Log Analytics Workspace Id	No value set
Log Analytics Custom Log Name	nifiprovenance
Log Analytics Workspace Key	No value set
Application ID	nifi
Instance ID	\${hostname(true)}
Job Name	nifi_reporting_job
Log Analytics URL Endpoint Format	https://{{0}}.ods.opinsights.azure.com/api/logs?api...
Event Type to Include	No value set
Event Type to Exclude	No value set
Component Type to Include	No value set
Component Type to Exclude	No value set
Component ID to Include	No value set
Component ID to Exclude	No value set
Component Name to Include	No value set

Two properties are required:

- The log analytics workspace ID
- The log analytics workspace key

You can find these values in the Azure portal by navigating to your Log Analytics workspace.

Other options are also available for customizing and filtering the provenance events that the system sends.

Security

Security provides assurances against deliberate attacks and the abuse of your valuable data and systems. For more information, see [Overview of the security pillar](#).

You can secure NiFi from an [authentication](#) and [authorization](#) point of view. You can also secure NiFi for all network communication including:

- Within the cluster.
- Between the cluster and ZooKeeper.

See the [Apache NiFi Administrators Guide](#) for instructions on turning on the following options:

- Kerberos
- Lightweight Directory Access Protocol (LDAP)
- Certificate-based authentication and authorization
- Two-way Secure Sockets Layer (SSL) for cluster communications

If you turn on ZooKeeper secure client access, configure NiFi by adding related properties to its `bootstrap.conf` configuration file. The following configuration entries provide an example:

```
config

java.arg.18=-Dzookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
java.arg.19=-Dzookeeper.client.secure=true
java.arg.20=-Dzookeeper.ssl.keyStore.location=/path/to/keystore.jks
java.arg.21=-Dzookeeper.ssl.keyStore.password=[KEYSTORE PASSWORD]
java.arg.22=-Dzookeeper.ssl.trustStore.location=/path/to/truststore.jks
java.arg.23=-Dzookeeper.ssl.trustStore.password=[TRUSTSTORE PASSWORD]
```

For general recommendations, see the [Linux security baseline](#).

Network security

When you implement this solution, keep in mind the following points about network security:

Network security groups

In Azure, you can use [network security groups](#) to restrict network traffic.

We recommend a *jumpbox* for connecting to the NiFi cluster for administrative tasks. Use this security-hardened VM with [just-in-time \(JIT\) access](#) or [Azure Bastion](#). Set up network security groups to control how you grant access to the jumpbox or Azure Bastion. You can achieve network isolation and control by using network security groups judiciously on the architecture's various subnets.

The following screenshot shows components in a typical virtual network. It contains a common subnet for the jumpbox, virtual machine scale set, and ZooKeeper VMs. This simplified network topology groups components into one subnet. Follow your organization's guidelines for separation of duties and network design.

Device	Type	Subnet
jumpbox-nic	Network interface	subnet-nifi
nifi-test-flow-vmss (instance 0)	Scale set instance	subnet-nifi
nifi-test-flow-vmss (instance 1)	Scale set instance	subnet-nifi
nifi-test-flow-vmss (instance 2)	Scale set instance	subnet-nifi
nifi-test-flow-zk-nic-0	Network interface	subnet-nifi
nifi-test-flow-zk-nic-1	Network interface	subnet-nifi
nifi-test-flow-zk-nic-2	Network interface	subnet-nifi

Outbound internet access consideration

NiFi in Azure doesn't need access to the public internet to run. If the data flow doesn't need internet access to retrieve data, improve the cluster's security by following these steps to disable outbound internet access:

1. Create an additional network security group rule in the virtual network.

2. Use these settings:

- Source: Any
- Destination: Internet
- Action: Deny

Outbound security rules						
Priority	Name	Port	Protocol	Source	Destination	Action
110	DenyInternet	Any	Any	VirtualNetwork	Internet	Allow

With this rule in place, you can still access some Azure services from the data flow if you configure a private endpoint in the virtual network. Use [Azure Private Link](#) for this purpose. This service provides a way for your traffic to travel on the Microsoft backbone network while not requiring any other external network access. NiFi currently supports Private Link for the Blob Storage and Data Lake Storage processors. If a network time protocol (NTP) server isn't available in your private network, allow outbound access to NTP. For detailed information, see [Time sync for Linux VMs in Azure](#).

Data protection

It's possible to operate NiFi unsecured, without wire encryption, identity and access management (IAM), or data encryption. But it's best to secure production and public-cloud deployments in these ways:

- Encrypting communication with Transport Layer Security (TLS)
- Using a supported authentication and authorization mechanism
- Encrypting data at rest

Azure Storage provides server-side transparent data encryption. But starting with the 1.13.2 release, NiFi doesn't configure wire encryption or IAM by default. This behavior might change in future releases.

The following sections show how to secure deployments in these ways:

- Enable wire encryption with TLS
- Configure authentication that's based on certificates or Microsoft Entra ID
- Manage encrypted storage on Azure

Disk encryption

To improve security, use Azure disk encryption. For a detailed procedure, see [Encrypt OS and attached data disks in a virtual machine scale set with the Azure CLI](#). That document also contains instructions on providing your own encryption key. The following steps outline a basic example for NiFi that work for most deployments:

1. To turn on disk encryption in an existing Key Vault instance, use the following Azure CLI command:

```
Azure CLI
az keyvault create --resource-group myResourceGroup --name myKeyVaultName --enabled-for-disk-encryption
```

2. Turn on encryption of the virtual machine scale set data disks with the following command:

```
Azure CLI
az vmss encryption enable --resource-group myResourceGroup --name myScaleSet --disk-encryption-keyvault myKeyVaultID
--volume-type DATA
```

3. You can optionally use a key encryption key (KEK). Use the following Azure CLI command to encrypt with a KEK:

```
Azure CLI
az vmss encryption enable --resource-group myResourceGroup --name myScaleSet \
--disk-encryption-keyvault myKeyVaultID \
--key-encryption-keyvault myKeyVaultID \
--key-encryption-key https://<mykeyvaultname>.vault.azure.net/keys/myKey/<version> \
--volume-type DATA
```

ⓘ Note

If you configured your virtual machine scale set for manual update mode, run the `update-instances` command. Include the version of the encryption key that you stored in Key Vault.

Encryption in transit

NiFi supports TLS 1.2 for encryption in transit. This protocol offers protection for user access to the UI. With clusters, the protocol protects communication between NiFi nodes. It can also protect communication with ZooKeeper. When you enable TLS, NiFi uses mutual TLS (mTLS) for mutual authentication for:

- Certificate-based client authentication if you configured this type of authentication.
- All intracluster communication.

To enable TLS, take the following steps:

1. Create a keystore and a truststore for client–server and intracluster communication and authentication.
2. Configure `$NIFI_HOME/conf/nifi.properties`. Set the following values:
 - Hostnames
 - Ports
 - Keystore properties
 - Truststore properties
 - Cluster and ZooKeeper security properties, if applicable
3. Configure authentication in `$NIFI_HOME/conf/authorizers.xml`, typically with an initial user that has certificate-based authentication or another option.
4. Optionally configure mTLS and a proxy read policy between NiFi and any proxies, load balancers, or external endpoints.

For a complete walkthrough, see [Securing NiFi with TLS](#) in the Apache project documentation.

ⓘ Note

As of version 1.13.2:

- NiFi doesn't enable TLS by default.
- There's no out-of-the-box support for anonymous and single user access for TLS-enabled NiFi instances.

To enable TLS for encryption in transit, configure a user group and policy provider for authentication and authorization in `$NIFI_HOME/conf/authorizers.xml`. For more information, see [Identity and access control](#) later in this article.

Certificates, keys, and keystores

To support TLS, generate certificates, store them in Java KeyStore and TrustStore, and distribute them across a NiFi cluster. There are two general options for certificates:

- Self-signed certificates
- Certificates that certified authorities (CAs) sign

With CA-signed certificates, it's best to use an intermediate CA to generate certificates for nodes in the cluster.

KeyStore and TrustStore are the key and certificate containers in the Java platform. KeyStore stores the private key and certificate of a node in the cluster. TrustStore stores one of the following types of certificates:

- All trusted certificates, for self-signed certificates in KeyStore
- A certificate from a CA, for CA-signed certificates in KeyStore

Keep the scalability of your NiFi cluster in mind when you choose a container. For instance, you might want to increase or decrease the number of nodes in a cluster in the future. Choose CA-signed certificates in KeyStore and one or more certificates from a CA in TrustStore in that case. With this option, there's no need to update the existing TrustStore in the existing nodes of the cluster. An existing TrustStore trusts and accepts certificates from these types of nodes:

- Nodes that you add to the cluster
- Nodes that replace other nodes in the cluster

NiFi configuration

To enable TLS for NiFi, use `$NIFI_HOME/conf/nifi.properties` to configure the properties in this table. Ensure that the following properties include the hostname that you use to access NiFi:

- `nifi.web.https.host` or `nifi.web.proxy.host`
- The host certificate's designated name or subject alternative names

Otherwise, a hostname verification failure or an HTTP HOST header verification failure might result, denying you access.

 [Expand table](#)

Property name	Description	Example values
<code>nifi.web.https.host</code>	Hostname or IP address to use for the UI and REST API. This value should be internally resolvable. We recommend not using a publicly accessible name.	<code>nifi.internal.cloudapp.net</code>
<code>nifi.web.https.port</code>	HTTPS port to use for the UI and REST API.	<code>9443</code> (default)
<code>nifi.web.proxy.host</code>	Comma-separated list of alternate hostnames that clients use to access the UI and REST API. This list typically includes any hostname that's specified as a subject alternative name (SAN) in the server certificate. The list can also include any hostname and port that a load balancer, proxy, or Kubernetes ingress controller uses.	<code>40.67.218.235, 40.67.218.235:443, nifi.westus2.cloudapp.com, nifi.westus2.cloudapp.com:443</code>
<code>nifi.security.keystore</code>	The path to a JKS or PKCS12 keystore that contains the certificate's private key.	<code>./conf/keystore.jks</code>
<code>nifi.security.keystoreType</code>	The keystore type.	<code>JKS</code> or <code>PKCS12</code>
<code>nifi.security.keystorePasswd</code>	The keystore password.	<code>08SitLBYPcZ7g/RpsqH+zM</code>
<code>nifi.security.keyPasswd</code>	(Optional) The password for the private key.	
<code>nifi.security.truststore</code>	The path to a JKS or PKCS12 truststore that contains certificates or CA certificates that authenticate trusted users and cluster nodes.	<code>./conf/truststore.jks</code>
<code>nifi.security.truststoreType</code>	The truststore type.	<code>JKS</code> or <code>PKCS12</code>
<code>nifi.security.truststorePasswd</code>	The truststore password.	<code>RJ1pGe6/TuN5fG+VnaEPI</code>
<code>nifi.cluster.protocol.is.secure</code>	The status of TLS for intracluster communication. If <code>nifi.cluster.is.node</code> is <code>true</code> , set this value to <code>true</code> to enable cluster TLS.	<code>true</code>
<code>nifi.remote.input.secure</code>	The status of TLS for site-to-site communication.	<code>true</code>

The following example shows how these properties appear in `$NIFI_HOME/conf/nifi.properties`. Note that the `nifi.web.http.host` and `nifi.web.http.port` values are blank.

config

```

nifi.remote.input.secure=true
nifi.web.http.host=
nifi.web.http.port=
nifi.web.https.host=nifi.internal.cloudapp.net

```

```

nifi.web.https.port=9443
nifi.web.proxy.host=40.67.218.235, 40.67.218.235:443, nifi.westus2.cloudapp.com, nifi.westus2.cloudapp.com:443
nifi.security.keystore=./conf/keystore.jks
nifi.security.keystoreType=JKS
nifi.security.keystorePasswd=08SitLBypCz7g/RpsqH+zM
nifi.security.keyPasswd=
nifi.security.truststore=./conf/truststore.jks
nifi.security.truststoreType=JKS
nifi.security.truststorePasswd=RJlpGe6/TuN5fG+VnaEPi8
nifi.cluster.protocol.is.secure=true

```

ZooKeeper configuration

For instructions on [enabling TLS in Apache ZooKeeper](#) for quorum communications and client access, see the [ZooKeeper Administrator's Guide](#). Only versions 3.5.5 or later support this functionality.

NiFi uses ZooKeeper for its zero-leader clustering and cluster coordination. Starting with version 1.13.0, NiFi supports secure client access to TLS-enabled instances of ZooKeeper. ZooKeeper stores cluster membership and cluster-scoped processor state in plain text. So it's important to use secure client access to ZooKeeper to authenticate ZooKeeper client requests. Also encrypt sensitive values in transit.

To enable TLS for NiFi client access to ZooKeeper, set the following properties in `$NIFI_HOME/conf/nifi.properties`. If you set `nifi.zookeeper.client.secure true` without configuring `nifi.zookeeper.security` properties, NiFi falls back to the keystore and truststore that you specify in `nifi.securityproperties`.

 [Expand table](#)

Property name	Description	Example values
<code>nifi.zookeeper.client.secure</code>	The status of client TLS when connecting to ZooKeeper.	<code>true</code>
<code>nifi.zookeeper.security.keystore</code>	The path to a JKS, PKCS12, or PEM keystore that contains the private key of the certificate that's presented to ZooKeeper for authentication.	<code>./conf/zookeeper.keystore.jks</code>
<code>nifi.zookeeper.security.keystoreType</code>	The keystore type.	<code>JKS</code> , <code>PKCS12</code> , <code>PEM</code> , or autodetect by extension
<code>nifi.zookeeper.security.keystorePasswd</code>	The keystore password.	<code>caB6ECKi03R/co+N+641rz</code>
<code>nifi.zookeeper.security.keyPasswd</code>	(Optional) The password for the private key.	
<code>nifi.zookeeper.security.truststore</code>	The path to a JKS, PKCS12, or PEM truststore that contains certificates or CA certificates that are used to authenticate ZooKeeper.	<code>./conf/zookeeper.truststore.jks</code>
<code>nifi.zookeeper.security.truststoreType</code>	The truststore type.	<code>JKS</code> , <code>PKCS12</code> , <code>PEM</code> , or autodetect by extension
<code>nifi.zookeeper.security.truststorePasswd</code>	The truststore password.	<code>qBdnLhsp+mKvV7wab/L4sv</code>
<code>nifi.zookeeper.connect.string</code>	The connection string to the ZooKeeper host or quorum. This string is a comma-separated list of <code>host:port</code> values. Typically the <code>secureClientPort</code> value isn't the same as the <code>clientPort</code> value. See your ZooKeeper configuration for the correct value.	<code>zookeeper1.internal.cloudapp.net:2281, zookeeper2.internal.cloudapp.net:2281, zookeeper3.internal.cloudapp.net:2281</code>

The following example shows how these properties appear in `$NIFI_HOME/conf/nifi.properties`:

config

```
nifi.zookeeper.client.secure=true
nifi.zookeeper.security.keystore=../conf/keystore.jks
nifi.zookeeper.security.keystoreType=JKS
nifi.zookeeper.security.keystorePasswd=caB6ECKi03R/co+N+64lrz
nifi.zookeeper.security.keyPasswd=
nifi.zookeeper.security.truststore=../conf/truststore.jks
nifi.zookeeper.security.truststoreType=JKS
nifi.zookeeper.security.truststorePasswd=qBdnLhsp+mKvV7wab/L4sv
nifi.zookeeper.connect.string=zookeeper1.internal.cloudapp.net:2281,zookeeper2.internal.cloudapp.net:2281,zookeeper3.internal.cloudapp.net:2281
```

For more information about securing ZooKeeper with TLS, see the [Apache NiFi Administration Guide](#).

Identity and access control

In NiFi, identity and access control is achieved through user authentication and authorization. For user authentication, NiFi has multiple options to choose from: Single User, LDAP, Kerberos, Security Assertion Markup Language (SAML), and OpenID Connect (OIDC). If you don't configure an option, NiFi uses client certificates to authenticate users over HTTPS.

If you're considering multifactor authentication, we recommend the combination of Microsoft Entra ID and [OIDC](#). Microsoft Entra ID supports cloud-native single sign-on (SSO) with OIDC. With this combination, users can take advantage of many enterprise security features:

- Logging and alerting on suspicious activities from user accounts
- Monitoring attempts to access deactivated credentials
- Alerting on unusual account sign-in behavior

For authorization, NiFi provides enforcement that's based on user, group, and access policies. NiFi provides this enforcement through UserGroupProviders and AccessPolicyProviders. By default, providers include File, LDAP, Shell, and Azure Graph-based UserGroupProviders. With [AzureGraphUserGroupProvider](#), you can source user groups from Microsoft Entra ID. You can then assign policies to these groups. For configuration instructions, see the [Apache NiFi Administration Guide](#).

AccessPolicyProviders that are based on files and Apache Ranger are currently available for managing and storing user and group policies. For detailed information, see the [Apache NiFi documentation](#) and [Apache Ranger documentation](#).

Application gateway

An application gateway provides a managed layer-7 load balancer for the NiFi interface. Configure the application gateway to use the virtual machine scale set of the NiFi nodes as its back-end pool.

For most NiFi installations, we recommend the following [Application Gateway](#) configuration:

- Tier: Standard
- SKU size: medium
- Instance count: two or more

Use a [health probe](#) to monitor the health of the web server on each node. Remove unhealthy nodes from the load balancer rotation. This approach makes it easier to view the user interface when the overall cluster is unhealthy. The browser only directs you to nodes that are currently healthy and responding to requests.

There are two key health probes to consider. Together they provide a regular heartbeat on the overall health of every node in the cluster. Configure the first health probe to point to the path `/NiFi`. This probe determines the health of the NiFi user interface on each node. Configure a second health probe for the path `/nifi-api/controller/cluster`. This probe indicates whether each node is currently healthy and joined to the overall cluster.

You have two options for configuring the application gateway's front-end IP address:

- With a public IP address
- With a private subnet IP address

Only include a public IP address if users need to access the UI over the public internet. If public internet access for users isn't required, access the load balancer front end from a jumpbox in the virtual network or through peering to your private network. If you configure the application gateway with a public IP address, we recommend enabling client certificate authentication for NiFi and enabling TLS for the NiFi UI. You can also use a network security group in the delegated application gateway subnet to limit source IP addresses.

Diagnostics and health monitoring

Within the diagnostics settings of Application Gateway, there's a configuration option for sending metrics and access logs. By using this option, you can send this information from the load balancer to various places:

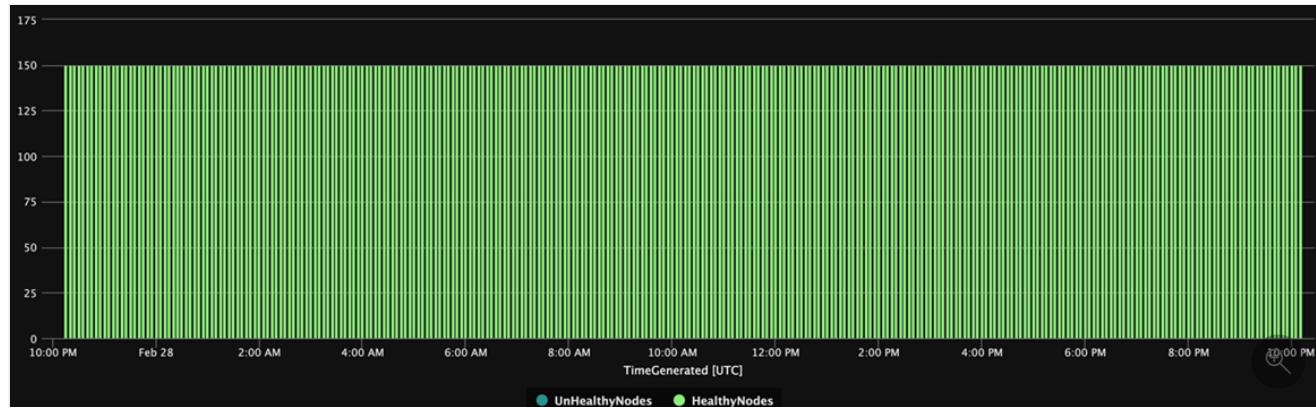
- A storage account
- Event Hubs
- A Log Analytics workspace

Turning on this setting is useful for debugging load-balancing issues and for gaining insight into the health of cluster nodes.

The following [Log Analytics](#) query shows cluster node health over time from an Application Gateway perspective. You can use a similar query to generate alerts or automated repair actions for unhealthy nodes.

```
Kusto
AzureDiagnostics
| summarize UnHealthyNodes = max(unHealthyHostCount_d), HealthyNodes = max(healthyHostCount_d) by bin(TimeGenerated, 5m)
| render timechart
```

The following chart of the query results shows a time view of the health of the cluster:



Availability

When you implement this solution, keep in mind the following points about availability:

Load balancer

Use a load balancer for the UI to increase UI availability during node downtime.

Separate VMs

To increase availability, deploy the ZooKeeper cluster on separate VMs from the VMs in the NiFi cluster. For more information on configuring ZooKeeper, see [State Management](#) in the Apache NiFi System Administrator's Guide.

Availability zones

Deploy both the NiFi virtual machine scale set and the ZooKeeper cluster in a cross-zone configuration to maximize availability. When communication between the nodes in the cluster crosses availability zones, it introduces a small amount of latency. But this latency typically has a minimal overall effect on the throughput of the cluster.

Virtual machine scale sets

We recommend deploying the NiFi nodes into a single virtual machine scale set that spans availability zones where available. For detailed information on using scale sets in this way, see [Create a virtual machine scale set that uses Availability Zones](#).

Monitoring

Multiple options are available for monitoring the health and performance of a NiFi cluster:

- Reporting tasks.
- [MonitoFi](#), a separate Microsoft-developed application. MonitoFi runs externally and monitors the cluster by using the NiFi API.

Reporting task-based monitoring

For monitoring, you can use a reporting task that you configure and run in NiFi. As [Diagnostics and health monitoring](#) discusses, Log Analytics provides a reporting task in the NiFi Azure bundle. You can use that reporting task to integrate the monitoring with Log Analytics and existing monitoring or logging systems.

Log Analytics queries

Sample queries in the following sections can help you get started. For an overview of how to query Log Analytics data, see [Azure Monitor log queries](#).

Log queries in Monitor and Log Analytics use a version of the [Kusto query language](#). But differences exist between log queries and Kusto queries. For more information, see [Kusto query overview](#).

For more structured learning, see these tutorials:

- [Get started with log queries in Azure Monitor](#)
- [Get started with Log Analytics in Azure Monitor](#)

Log Analytics reporting task

By default, NiFi sends metrics data to the `nifimetrics` table. But you can configure a different destination in the reporting task properties. The reporting task captures the following NiFi metrics:

 [Expand table](#)

Metric type	Metric name
NiFi Metrics	<code>FlowFilesReceived</code>
NiFi Metrics	<code>FlowFilesSent</code>
NiFi Metrics	<code>FlowFilesQueued</code>
NiFi Metrics	<code>BytesReceived</code>
NiFi Metrics	<code>BytesWritten</code>
NiFi Metrics	<code>BytesRead</code>
NiFi Metrics	<code>BytesSent</code>
NiFi Metrics	<code>BytesQueued</code>
Port status metrics	<code>InputCount</code>
Port status metrics	<code>InputBytes</code>
Connection status metrics	<code>QueuedCount</code>
Connection status metrics	<code>QueuedBytes</code>
Port status metrics	<code>OutputCount</code>
Port status metrics	<code>OutputBytes</code>

Metric type	Metric name
JVM Metrics	jvm.uptime
JVM Metrics	jvm.heap_used
JVM Metrics	jvm.heap_usage
JVM Metrics	jvm.non_heap_usage
JVM Metrics	jvm.thread_states.runnable
JVM Metrics	jvm.thread_states.blocked
JVM Metrics	jvm.thread_states.timed_waiting
JVM Metrics	jvm.thread_states.terminated
JVM Metrics	jvm.thread_count
JVM Metrics	jvm.daemon_thread_count
JVM Metrics	jvm.file_descriptor_usage
JVM Metrics	jvm.gc.runs jvm.gc.runs.g1_old_generation jvm.gc.runs.g1_young_generation
JVM Metrics	jvm.gc.time jvm.gc.time.g1_young_generation jvm.gc.time.g1_old_generation
JVM Metrics	jvm.buff_pool_direct_capacity
JVM Metrics	jvm.buff_pool_direct_count
JVM Metrics	jvm.buff_pool_direct_mem_used
JVM Metrics	jvm.buff_pool_mapped_capacity
JVM Metrics	jvm.buff_pool_mapped_count
JVM Metrics	jvm.buff_pool_mapped_mem_used
JVM Metrics	jvm.mem_pool_code_cache
JVM Metrics	jvm.mem_pool_compressed_class_space
JVM Metrics	jvm.mem_pool_g1_eden_space
JVM Metrics	jvm.mem_pool_g1_old_gen
JVM Metrics	jvm.mem_pool_g1_survivor_space
JVM Metrics	jvm.mem_pool_metaspace
JVM Metrics	jvm.thread_states.new
JVM Metrics	jvm.thread_states.waiting

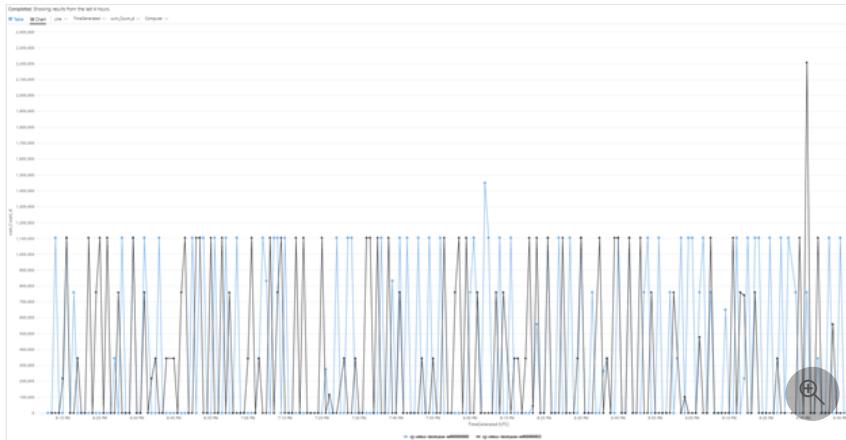
Metric type	Metric name
Processor Level metrics	BytesRead
Processor Level metrics	BytesWritten
Processor Level metrics	FlowFilesReceived
Processor Level metrics	FlowFilesSent

Here's a sample query for the `BytesQueued` metric of a cluster:

Kusto

```
let table_name = nifimetrics_CL;
let metric = "BytesQueued";
table_name
| where Name_s == metric
| where Computer contains {ComputerName}
| project TimeGenerated, Computer, ProcessGroupName_s, Count_d, Name_s
| summarize sum(Count_d) by bin(TimeGenerated, 1m), Computer, Name_s
| render timechart
```

That query produces a chart like the one in this screenshot:



⚠ Note

When you run NiFi on Azure, you're not limited to the Log Analytics reporting task. NiFi supports reporting tasks for many third-party monitoring technologies. For a list of supported reporting tasks, see the [Reporting Tasks](#) section of the [Apache NiFi Documentation index](#).

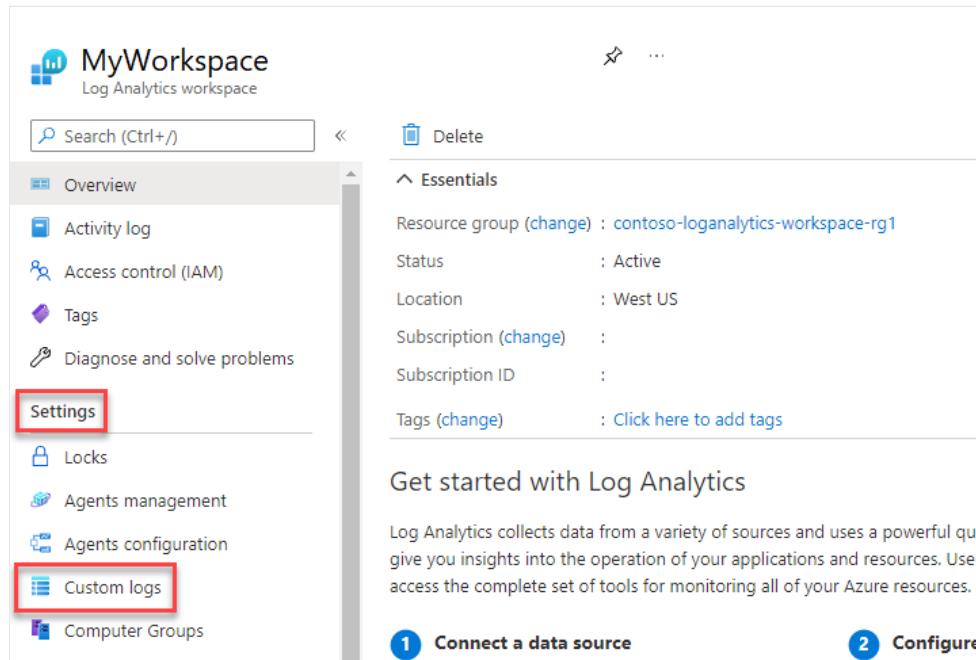
NiFi infrastructure monitoring

Besides the reporting task, install the [Log Analytics VM extension](#) on the NiFi and ZooKeeper nodes. This extension gathers logs, additional VM-level metrics, and metrics from ZooKeeper.

Custom logs for the NiFi app, user, bootstrap, and ZooKeeper

To capture more logs, follow these steps:

1. In the Azure portal, select **Log Analytics workspaces**, and then select your workspace.
2. Under **Settings**, select **Custom logs**.



MyWorkspace
Log Analytics workspace

Search (Ctrl+ /) Delete

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Locks

Agents management

Agents configuration

Custom logs

Computer Groups

Resource group (change) : contoso-loganalytics-workspace-rg1

Status : Active

Location : West US

Subscription (change) :

Subscription ID :

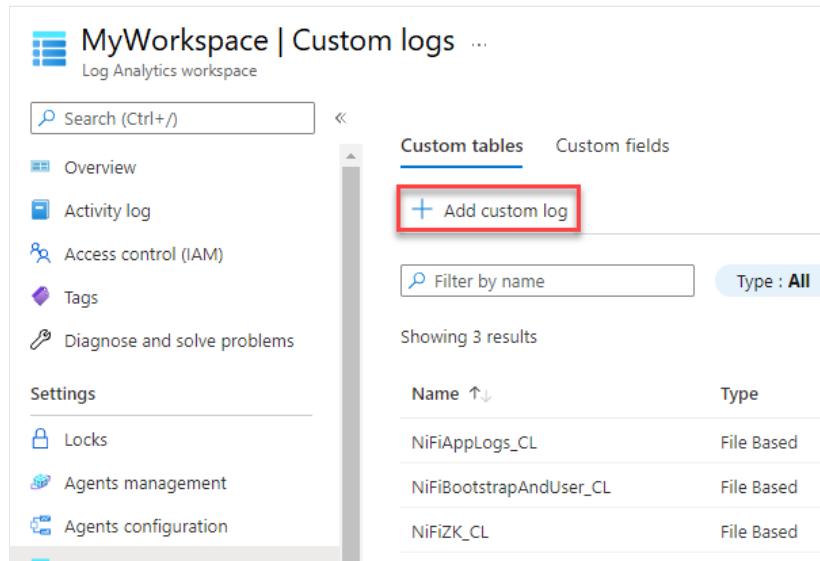
Tags (change) : [Click here to add tags](#)

Get started with Log Analytics

Log Analytics collects data from a variety of sources and uses a powerful query language to give you insights into the operation of your applications and resources. Use the Log Analytics workspace to access the complete set of tools for monitoring all of your Azure resources.

1 Connect a data source **2** Configure

3. Select Add custom log.



MyWorkspace | Custom logs

Search (Ctrl+ /)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Locks

Agents management

Agents configuration

Custom tables **Custom fields**

+ Add custom log

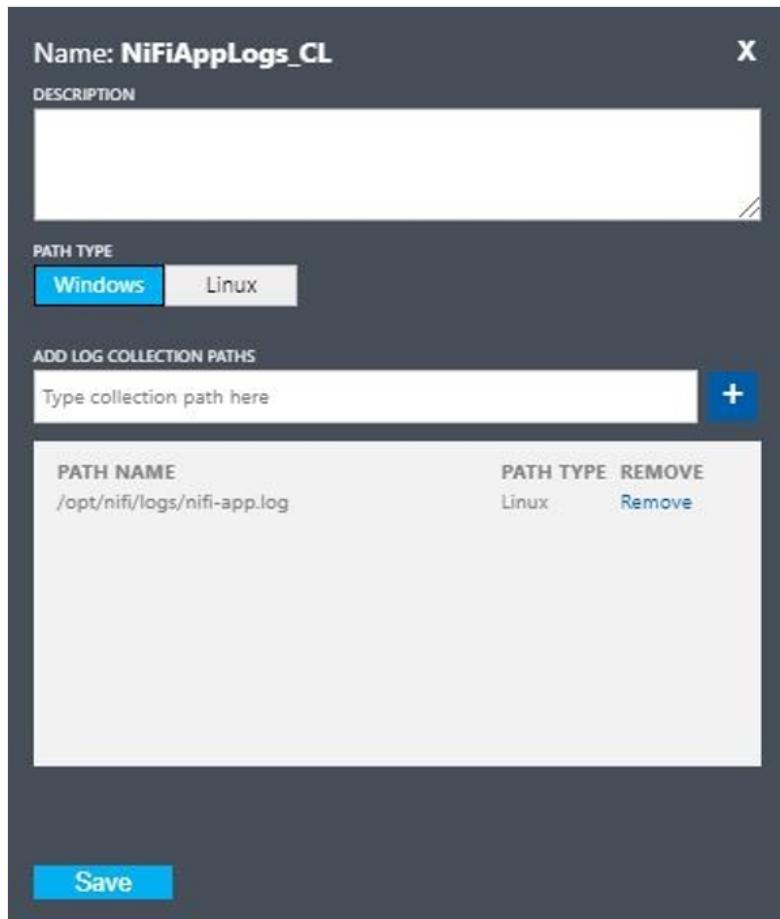
Filter by name Type : All

Showing 3 results

Name	Type
NiFiAppLogs_CL	File Based
NiFiBootstrapAndUser_CL	File Based
NiFiZK_CL	File Based

4. Set up a custom log with these values:

- Name: NiFiAppLogs
- Path type: Linux
- Path name: /opt/nifi/logs/nifi-app.log



5. Set up a custom log with these values:

- Name: NiFiBootstrapAndUser
- First path type: Linux
- First path name: /opt/nifi/logs/nifi-user.log
- Second path type: Linux
- Second path name: /opt/nifi/logs/nifi-bootstrap.log

Name: **NiFiBootstrapAndUser_CL**

DESCRIPTION

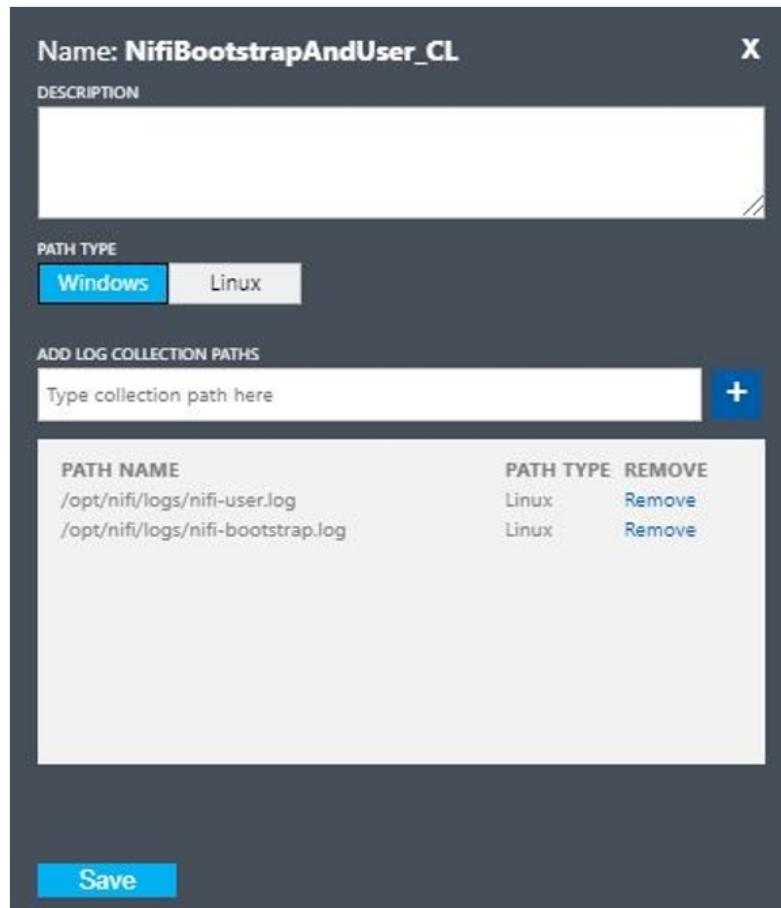
PATH TYPE

Windows

ADD LOG COLLECTION PATHS

Type collection path here

PATH NAME	PATH TYPE	REMOVE
/opt/nifi/logs/nifi-user.log	Linux	Remove
/opt/nifi/logs/nifi-bootstrap.log	Linux	Remove



6. Set up a custom log with these values:

- Name: **NiFiZK**
- Path type: **Linux**
- Path name: **/opt/zookeeper/logs/*.out**

Name: **NiFiZK_CL**

DESCRIPTION

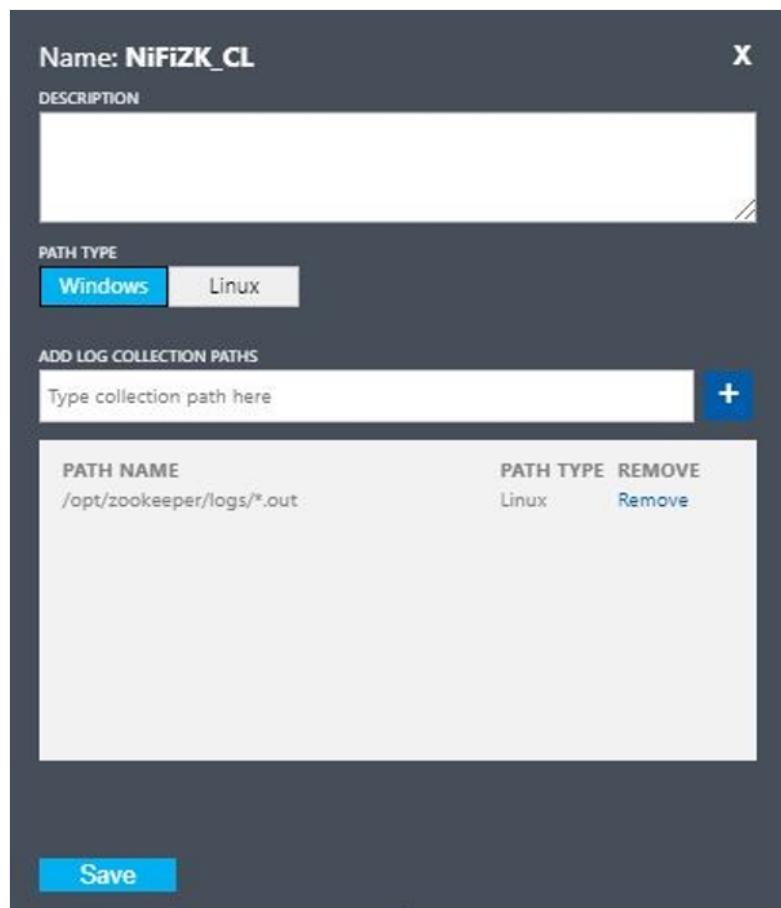
PATH TYPE

Windows

ADD LOG COLLECTION PATHS

Type collection path here

PATH NAME	PATH TYPE	REMOVE
/opt/zookeeper/logs/*.out	Linux	Remove



Here's a sample query of the `NiFiAppLogs` custom table that the first example created:

```
Kusto

NiFiAppLogs_CL
| where TimeGenerated > ago(24h)
| where Computer contains {ComputerName} and RawData contains "error"
| limit 10
```

That query produces results similar to the following results:

TimeGenerated [UTC]	Computer	RawData	Type	ResourceId
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,887 ERROR [Index Provenance Events-1] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,887 ERROR [Index Provenance Events-2] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,887 ERROR [Index Provenance Events-2] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,888 ERROR [Index Provenance Events-2] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,888 ERROR [Index Provenance Events-1] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,888 ERROR [Index Provenance Events-2] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,888 ERROR [Index Provenance Events-2] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,888 ERROR [Index Provenance Events-1] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20
2/27/2020, 10:31:06.000 PM	nifi-longhaul-nifi00000N	2020-02-27 04:51:41,888 ERROR [Index Provenance Events-2] o.a.n.p.i...	NiFiAppLogs_CL	/subscriptions/a5ab90eb-20

Infrastructure log configuration

You can use Monitor to monitor and manage VMs or physical computers. These resources can be in your local datacenter or other cloud environment. To set up this monitoring, deploy the Log Analytics agent. Configure the agent to report to a Log Analytics workspace. For more information, see [Log Analytics agent overview](#).

The following screenshot shows a sample agent configuration for NiFi VMs. The `Perf` table stores the collected data.

Here's a sample query for the NiFi app `Perf` logs:

```
Kusto

let cluster_name = {ComputerName};
// The hourly average of CPU usage across all computers.
Perf
| where Computer contains {ComputerName}
| where CounterName == "% Processor Time" and InstanceName == "_Total"
| where ObjectName == "Processor"
| summarize CPU_Time_Avg = avg(CounterValue) by bin(TimeGenerated, 30m), Computer
```

That query produces a report like the one in this screenshot:



Alerts

Use Monitor to create alerts on the health and performance of the NiFi cluster. Example alerts include:

- The total queue count has exceeded a threshold.
- The `BytesWritten` value is under an expected threshold.
- The `FlowFilesReceived` value is under a threshold.
- The cluster is unhealthy.

For more information on setting up alerts in Monitor, see [Overview of alerts in Microsoft Azure](#).

Configuration parameters

The following sections discuss recommended, non-default configurations for NiFi and its dependencies, including ZooKeeper and Java. These settings are suited for cluster sizes that are possible in the cloud. Set the properties in the following configuration files:

- `$NIFI_HOME/conf/nifi.properties`
- `$NIFI_HOME/conf/bootstrap.conf`
- `$ZOOKEEPER_HOME/conf/zoo.cfg`
- `$ZOOKEEPER_HOME/bin/zkEnv.sh`

For detailed information about available configuration properties and files, see the [Apache NiFi System Administrator's Guide](#) and [ZooKeeper Administrator's Guide](#).

NiFi

For an Azure deployment, consider adjusting properties in `$NIFI_HOME/conf/nifi.properties`. The following table lists the most important properties. For more recommendations and insights, see the [Apache NiFi mailing lists](#).

[Expand table](#)

Parameter	Description	Default	Recommendation
<code>nifi.cluster.node.connection.timeout</code>	How long to wait when opening a connection to other cluster nodes.	5 seconds	60 seconds
<code>nifi.cluster.node.read.timeout</code>	How long to wait for a response when making a request to other cluster nodes.	5 seconds	60 seconds
<code>nifi.cluster.protocol.heartbeat.interval</code>	How often to send heartbeats back to the cluster coordinator.	5 seconds	60 seconds
<code>nifi.cluster.node.max.concurrent.requests</code>	What level of parallelism to use when replicating HTTP calls like REST API calls to other cluster nodes.	100	500
<code>nifi.cluster.node.protocol.threads</code>	Initial thread pool size for inter-cluster/replicated communications.	10	50
<code>nifi.cluster.node.protocol.max.threads</code>	Maximum number of threads to use for inter-	50	75

Parameter	Description	Default	Recommendation
	cluster/replicated communications.		
<code>nifi.cluster.flow.election.max.candidates</code>	Number of nodes to use when deciding what the current flow is. This value short-circuits the vote at the specified number.	empty	75
<code>nifi.cluster.flow.election.max.wait.time</code>	How long to wait on nodes before deciding what the current flow is.	5 minutes	5 minutes

Cluster behavior

When you configure clusters, keep in mind the following points.

Timeout

To ensure the overall health of a cluster and its nodes, it can be beneficial to increase timeouts. This practice helps guarantee that failures don't result from transient network problems or high loads.

In a distributed system, the performance of individual systems varies. This variation includes network communications and latency, which usually affects inter-node, inter-cluster communication. The network infrastructure or the system itself can cause this variation. As a result, the probability of variation is very likely in large clusters of systems. In Java applications under load, pauses in garbage collection (GC) in the Java virtual machine (JVM) can also affect request response times.

Use properties in the following sections to configure timeouts to suit your system's needs:

`nifi.cluster.node.connection.timeout` and `nifi.cluster.node.read.timeout`

The `nifi.cluster.node.connection.timeout` property specifies how long to wait when opening a connection. The `nifi.cluster.node.read.timeout` property specifies how long to wait when receiving data between requests. The default value for each property is five seconds. These properties apply to node-to-node requests. Increasing these values helps alleviate several related problems:

- Being disconnected by the cluster coordinator because of heartbeat interruptions
- Failure to get the flow from the coordinator when joining the cluster
- Establishing site-to-site (S2S) and load-balancing communications

Unless your cluster has a very small scale set, such as three nodes or fewer, use values that are greater than the defaults.

`nifi.cluster.protocol.heartbeat.interval`

As part of the NiFi clustering strategy, each node emits a heartbeat to communicate its healthy status. By default, nodes send heartbeats every five seconds. If the cluster coordinator detects that eight heartbeats in a row from a node have failed, it disconnects the node. Increase the interval that's set in the `nifi.cluster.protocol.heartbeat.interval` property to help accommodate slow heartbeats and prevent the cluster from disconnecting nodes unnecessarily.

Concurrency

Use properties in the following sections to configure concurrency settings:

`nifi.cluster.node.protocol.threads` and `nifi.cluster.node.protocol.max.threads`

The `nifi.cluster.node.protocol.max.threads` property specifies the maximum number of threads to use for all-cluster communications such as S2S load balancing and UI aggregation. The default for this property is 50 threads. For large clusters, increase this value to account for the greater number of requests that these operations require.

The `nifi.cluster.node.protocol.threads` property determines the initial thread pool size. The default value is 10 threads. This value is a minimum. It grows as needed up to the maximum set in `nifi.cluster.node.protocol.max.threads`. Increase the `nifi.cluster.node.protocol.threads` value for clusters that use a large scale set at launch.

nifi.cluster.node.max.concurrent.requests

Many HTTP requests like REST API calls and UI calls need to be replicated to other nodes in the cluster. As the size of the cluster grows, an increasing number of requests get replicated. The `nifi.cluster.node.max.concurrent.requests` property limits the number of outstanding requests. Its value should exceed the expected cluster size. The default value is 100 concurrent requests. Unless you're running a small cluster of three or fewer nodes, prevent failed requests by increasing this value.

Flow election

Use properties in the following sections to configure flow election settings:

nifi.cluster.flow.election.max.candidates

NiFi uses zero-leader clustering, which means there isn't one specific authoritative node. As a result, nodes vote on which flow definition counts as the correct one. They also vote to decide which nodes join the cluster.

By default, the `nifi.cluster.flow.election.max.candidates` property is the maximum wait time that the `nifi.cluster.flow.election.max.wait.time` property specifies. When this value is too high, startup can be slow. The default value for `nifi.cluster.flow.election.max.wait.time` is five minutes. Set the maximum number of candidates to a non-empty value like `1` or greater to ensure that the wait is no longer than needed. If you set this property, assign it a value that corresponds to the cluster size or some majority fraction of the expected cluster size. For small, static clusters of 10 or fewer nodes, set this value to the number of nodes in the cluster.

nifi.cluster.flow.election.max.wait.time

In an elastic cloud environment, the time to provision hosts affects the application startup time. The `nifi.cluster.flow.election.max.wait.time` property determines how long NiFi waits before deciding on a flow. Make this value commensurate with the overall launch time of the cluster at its starting size. In initial testing, five minutes are more than adequate in all Azure regions with the recommended instance types. But you can increase this value if the time to provision regularly exceeds the default.

Java

We recommend using an [LTS release of Java](#). Of these releases, Java 11 is slightly preferable to Java 8 because Java 11 supports a faster garbage collection implementation. However, it's possible to have a high-performance NiFi deployment by using either release.

The following sections discuss common JVM configurations to use when running NiFi. Set JVM parameters in the bootstrap configuration file at `$NIFI_HOME/conf/bootstrap.conf`.

Garbage collector

If you're running Java 11, we recommend using the G1 garbage collector (G1GC) in most situations. G1GC has improved performance over ParallelGC because G1GC reduces the length of GC pauses. G1GC is the default in Java 11, but you can configure it explicitly by setting the following value in `bootstrap.conf`:

```
java.arg.13=-XX:+UseG1GC
```

If you're running Java 8, don't use G1GC. Use ParallelGC instead. There are deficiencies in the Java 8 implementation of G1GC that prevent you from using it with the recommended repository implementations. ParallelGC is slower than G1GC. But with ParallelGC, you can still have a high-performance NiFi deployment with Java 8.

Heap

A set of properties in the `bootstrap.conf` file determines the configuration of the NiFi JVM heap. For a standard flow, configure a 32-GB heap by using these settings:

```
config
```

```
java.arg.3=-Xmx32g
java.arg.2=-Xms32g
```

To choose the optimal heap size to apply to the JVM process, consider two factors:

- The characteristics of the data flow
- The way that NiFi uses memory in its processing

For detailed documentation, see [Apache NiFi in Depth](#).

Make the heap only as large as needed to fulfill the processing requirements. This approach minimizes the length of GC pauses. For general considerations for Java garbage collection, see the garbage collection tuning guide for your version of Java.

When adjusting JVM memory settings, consider these important factors:

- The number of *FlowFiles*, or NiFi data records, that are active in a given period. This number includes back-pressed or queued *FlowFiles*.
- The number of attributes that are defined in *FlowFiles*.
- The amount of memory that a processor requires to process a particular piece of content.
- The way that a processor processes data:
 - Streaming data
 - Using record-oriented processors
 - Holding all data in memory at once

These details are important. During processing, NiFi holds references and attributes for each *FlowFile* in memory. At peak performance, the amount of memory that the system uses is proportional to the number of live *FlowFiles* and all the attributes that they contain. This number includes queued *FlowFiles*. NiFi can swap to disk. But avoid this option because it hurts performance.

Also keep in mind basic object memory usage. Specifically, make your heap large enough to hold objects in memory. Consider these tips for configuring the memory settings:

- Run your flow with representative data and minimal back pressure by starting with the setting `-Xmx4G` and then increasing memory conservatively as needed.
- Run your flow with representative data and peak back pressure by starting with the setting `-Xmx4G` and then increasing cluster size conservatively as needed.
- Profile the application while the flow is running by using tools such as VisualVM and YourKit.
- If conservative increases in heap don't improve performance significantly, consider redesigning flows, processors, and other aspects of your system.

Additional JVM parameters

The following table lists additional JVM options. It also provides the values that worked best in initial testing. Tests observed GC activity and memory usage and used careful profiling.

[Expand table](#)

Parameter	Description	JVM default	Recommendation
<code>InitiatingHeapOccupancyPercent</code>	The amount of heap that's in use before a marking cycle is triggered.	45	35
<code>ParallelGCThreads</code>	The number of threads that GC uses. This value is capped to limit the overall effect on the system.	5/8 of the number of vCPUs	8
<code>ConcGCThreads</code>	The number of GC threads to run in parallel. This value is increased to account for capped <code>ParallelGCThreads</code> .	1/4 of the <code>ParallelGCThreads</code> value	4
<code>G1ReservePercent</code>	The percentage of reserve memory to keep free. This value is increased to avoid to-space exhaustion, which helps avoid full GC.	10	20
<code>UseStringDeduplication</code>	Whether to try to identify and de-duplicate references to identical strings. Turning on this feature can result in memory savings.	-	present

Configure these settings by adding the following entries to the NiFi `bootstrap.conf`:

```
config

java.arg.17=-XX:+UseStringDeduplication
java.arg.18=-XX:G1ReservePercent=20
java.arg.19=-XX:ParallelGCThreads=8
java.arg.20=-XX:ConcGCThreads=4
java.arg.21=-XX:InitiatingHeapOccupancyPercent=35
```

ZooKeeper

For improved fault tolerance, run ZooKeeper as a cluster. Take this approach even though most NiFi deployments put a relatively modest load on ZooKeeper. Turn on clustering for ZooKeeper explicitly. By default, ZooKeeper runs in single-server mode. For detailed information, see [Clustered \(Multi-Server\) Setup](#) in the ZooKeeper Administrator's Guide.

Except for the clustering settings, use default values for your ZooKeeper configuration.

If you have a large NiFi cluster, you might need to use a greater number of ZooKeeper servers. For smaller cluster sizes, smaller VM sizes and Standard SSD managed disks are sufficient.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- [Muazma Zahid](#) | Principal PM Manager

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

The material and recommendations in this document came from several sources:

- Experimentation
- Azure best practices
- NiFi community knowledge, best practices, and documentation

For more information, see the following resources:

- [Apache NiFi System Administrator's Guide](#)
- [Apache NiFi mailing lists](#)
- [Cloudera best practices for setting up a high-performance NiFi installation](#)
- [Azure premium storage: design for high performance](#)
- [Troubleshoot Azure virtual machine performance on Linux or Windows](#)

Related resources

- [Apache NiFi monitoring with MonitoFi](#)
- [Helm-based deployments for Apache NiFi](#)
- [Azure Data Explorer monitoring](#)
- [\[Hybrid ETL with Azure Data Factory\]\[Hybrid ETL with Azure Data Factory\]](#)
- [\[DataOps for the modern data warehouse\]\[DataOps for the modern data warehouse\]](#)
- [Data warehousing and analytics](#)

Automated enterprise BI

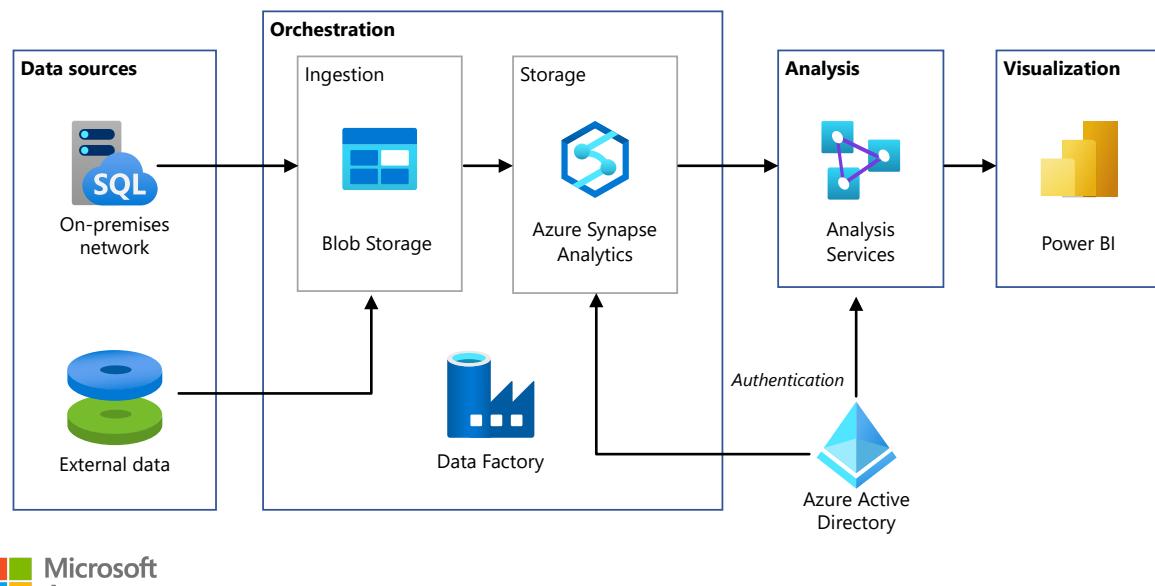
Microsoft Entra ID Azure Analysis Services Azure Blob Storage Azure Data Factory
Azure Synapse Analytics

💡 Solution ideas

This article is a solution idea. If you'd like us to expand the content with more information, such as potential use cases, alternative services, implementation considerations, or pricing guidance, let us know by providing [GitHub feedback](#).

This example is about how to perform incremental loading in an [extract, load, and transform \(ELT\)](#) pipeline. It uses Azure Data Factory to automate the ELT pipeline. The pipeline incrementally moves the latest OLTP data from an on-premises SQL Server database into Azure Synapse. Transactional data is transformed into a tabular model for analysis.

Architecture



Download a [Visio file](#) of this architecture.

This architecture builds on the one shown in [Enterprise BI with Azure Synapse](#), but adds some features that are important for enterprise data warehousing scenarios.

- Automation of the pipeline using Data Factory.

- Incremental loading.
- Integrating multiple data sources.
- Loading binary data such as geospatial data and images.

Workflow

The architecture consists of the following services and components.

Data sources

On-premises SQL Server. The source data is located in a SQL Server database on premises. To simulate the on-premises environment. The [Wide World Importers OLTP sample database](#) is used as the source database.

External data. A common scenario for data warehouses is to integrate multiple data sources. This reference architecture loads an external data set that contains city populations by year, and integrates it with the data from the OLTP database. You can use this data for insights such as: "Does sales growth in each region match or exceed population growth?"

Ingestion and data storage

Blob Storage. Blob storage is used as a staging area for the source data before loading it into Azure Synapse.

Azure Synapse. [Azure Synapse](#) is a distributed system designed to perform analytics on large data. It supports massive parallel processing (MPP), which makes it suitable for running high-performance analytics.

Azure Data Factory. [Data Factory](#) is a managed service that orchestrates and automates data movement and data transformation. In this architecture, it coordinates the various stages of the ELT process.

Analysis and reporting

Azure Analysis Services. [Analysis Services](#) is a fully managed service that provides data modeling capabilities. The semantic model is loaded into Analysis Services.

Power BI. Power BI is a suite of business analytics tools to analyze data for business insights. In this architecture, it queries the semantic model stored in Analysis Services.

Authentication

Microsoft Entra ID (Microsoft Entra ID) authenticates users who connect to the Analysis Services server through Power BI.

Data Factory can also use Microsoft Entra ID to authenticate to Azure Synapse, by using a service principal or Managed Service Identity (MSI).

Components

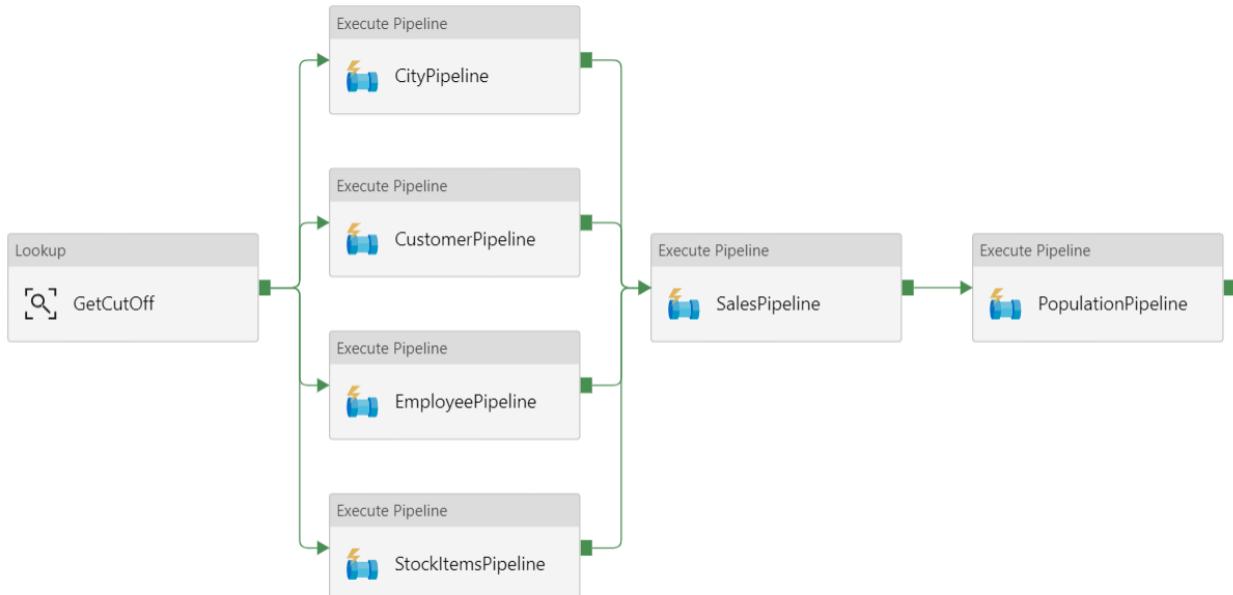
- [Azure Blob Storage](#)
- [Azure Synapse Analytics](#)
- [Azure Data Factory](#)
- [Azure Analysis Services](#)
- [Power BI](#)
- [Microsoft Entra ID](#)

Scenario details

Data pipeline

In [Azure Data Factory](#), a pipeline is a logical grouping of activities used to coordinate a task — in this case, loading and transforming data into Azure Synapse.

This reference architecture defines a parent pipeline that runs a sequence of child pipelines. Each child pipeline loads data into one or more data warehouse tables.



Recommendations

Incremental loading

When you run an automated ETL or ELT process, it's most efficient to load only the data that changed since the previous run. This is called an *incremental load*, as opposed to a full load that loads all the data. To perform an incremental load, you need a way to identify which data has changed. The most common approach is to use a *high water mark* value, which means tracking the latest value of some column in the source table, either a datetime column or a unique integer column.

Starting with SQL Server 2016, you can use [temporal tables](#). These are system-versioned tables that keep a full history of data changes. The database engine automatically records the history of every change in a separate history table. You can query the historical data by adding a FOR SYSTEM_TIME clause to a query. Internally, the database engine queries the history table, but this is transparent to the application.

Note

For earlier versions of SQL Server, you can use [Change Data Capture \(CDC\)](#). This approach is less convenient than temporal tables, because you have to query a separate change table, and changes are tracked by a log sequence number, rather than a timestamp.

Temporal tables are useful for dimension data, which can change over time. Fact tables usually represent an immutable transaction such as a sale, in which case keeping the system version history doesn't make sense. Instead, transactions usually have a column that represents the transaction date, which can be used as the watermark value. For example, in the Wide World Importers OLTP database, the Sales.Invoices and Sales.InvoiceLines tables have a `LastEditedWhen` field that defaults to `sysdatetime()`.

Here is the general flow for the ELT pipeline:

1. For each table in the source database, track the cutoff time when the last ELT job ran. Store this information in the data warehouse. (On initial setup, all times are set to '1-1-1900').
2. During the data export step, the cutoff time is passed as a parameter to a set of stored procedures in the source database. These stored procedures query for any records that were changed or created after the cutoff time. For the Sales fact table,