Table of Contents

Introduction to private Docker container registries

6/27/2017 • 2 min to read • Edit Online

Azure Container Registry is a managed Docker registry service based on the open-source Docker Registry 2.0. Create and maintain Azure container registries to store and manage your private Docker container images. Use container registries in Azure with your existing container development and deployment pipelines, and draw on the body of Docker community expertise.

For background about Docker and containers, see:

- Docker user guide

Use cases

Pull images from an Azure container registry to various deployment targets:

- **Scalable orchestration systems** that manage containerized applications across clusters of hosts, including DC/OS, Docker Swarm, and Kubernetes.
- **Azure services** that support building and running applications at scale, including Container Service, App Service, Batch, Service Fabric, and others.

Developers can also push to a container registry as part of a container development workflow. For example, target a container registry from a continuous integration and deployment tool such as Visual Studio Team Services or Jenkins.

Key concepts

- **Registry** - Create one or more container registries in your Azure subscription. Each registry is backed by a standard Azure storage account in the same location. Take advantage of local, network-close storage of your container images by creating a registry in the same Azure location as your deployments. A fully qualified registry name has the form `myregistry.azurecr.io`.

  You control access to a container registry using an Azure Active Directory-backed service principal or a provided admin account. Run the standard `docker login` command to authenticate with a registry.

- **Repository** - A registry contains one or more repositories, which are groups of container images. Azure Container Registry supports multilevel repository namespaces. This feature enables you to group collections of images related to a specific app, or a collection of apps to specific development or operational teams. For example:

  - `myregistry.azurecr.io/aspnetcore:1.0.1` represents a corporate-wide image
  - `myregistry.azurecr.io/warrantydept/dotnet-build` represents an image used to build .NET apps, shared across the warranty department
  - `myregistry.azrecr.io/warrantydept/customersubmissions/web` represents a web image, grouped in the customer submissions app, owned by the warranty department
- **Image** - Stored in a repository, each image is a read-only snapshot of a Docker container. Azure container registries can include both Windows and Linux images. You control image names for all your container deployments. Use standard Docker commands to push images into a repository, or pull an image from a repository.

- **Container** - A container defines a software application and its dependencies wrapped in a complete filesystem including code, runtime, system tools, and libraries. Run Docker containers based on Windows or

Linux images that you pull from a container registry. Containers running on a single machine share the operating system kernel. Docker containers are fully portable to all major Linux distros, Mac, and Windows.

Next steps

- Create a container registry using the Azure portal
- Create a container registry using the Azure CLI
- Push your first image using the Docker CLI
- To build a continuous integration and deployment workflow using Visual Studio Team Services, Azure Container Service, and Azure Container Registry, see this tutorial.
- If you want to set up your own Docker private registry in Azure (without a public endpoint), see Deploying Your Own Private Docker Registry on Azure.

Create a private Docker container registry using the Azure portal

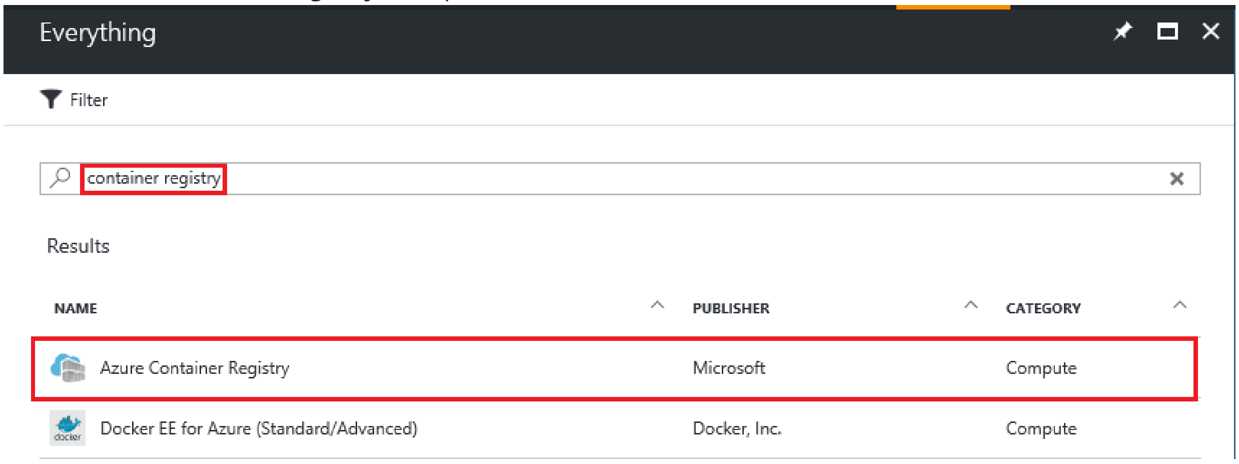
6/27/2017 • 2 min to read • Edit Online

Use the Azure portal to create a container registry and manage its settings. You can also create and manage container registries using the Azure CLI 2.0 commands or programmatically with the Container Registry REST API.

For background and concepts, see the overview.

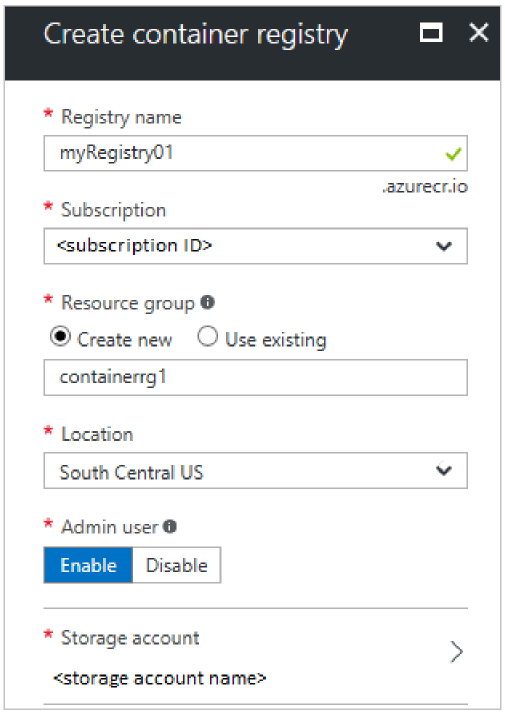
Create a container registry

1. In the Azure portal, click **+ New**.
2. Search the marketplace for **Azure container registry**.
3. Select **Azure Container Registry**, with publisher **Microsoft**.



NAME	PUBLISHER	CATEGORY
Azure Container Registry	Microsoft	Compute
Docker EE for Azure (Standard/Advanced)	Docker, Inc.	Compute

4. Click **Create**. The **Azure Container Registry** blade appears.



* Registry name: myRegistry01 .azurecr.io
* Subscription: <subscription ID>
* Resource group: [✓ Create new] Use existing
containerrg1
* Location: South Central US
* Admin user: [✓ Enable] Disable
* Storage account: <storage account name>

5. In the **Azure Container Registry** blade, enter the following information. Click **Create** when you are done.

a. **Registry name**: A globally unique top-level domain name for your specific registry. In this example, the registry name is *myRegistry01*, but substitute a unique name of your own. The name can contain only letters and numbers.

b. **Resource group**: Select an existing resource group or type the name for a new one.

c. **Location**: Select an Azure datacenter location where the service is available, such as **South Central US**.

d. **Admin user**: If you want, enable an admin user to access the registry. You can change this setting after creating the registry.

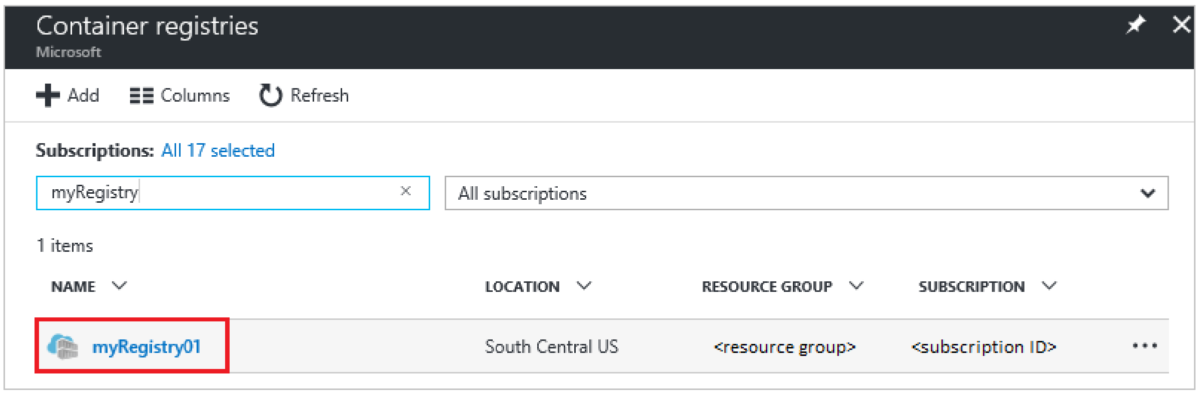
> **IMPORTANT**
> In addition to providing access through an admin user account, container registries support authentication backed by Azure Active Directory service principals. For more information and considerations, see Authenticate with a container registry.

e. **Storage account**: Use the default setting to create a storage account, or select an existing storage account in the same location. Currently Premium Storage is not supported.

Manage registry settings

After creating the registry, find the registry settings by starting at the **Container Registries** blade in the portal. For example, you might need the settings to log in to your registry, or you might want to enable or disable the admin user.

1. On the **Container Registries** blade, click the name of your registry.



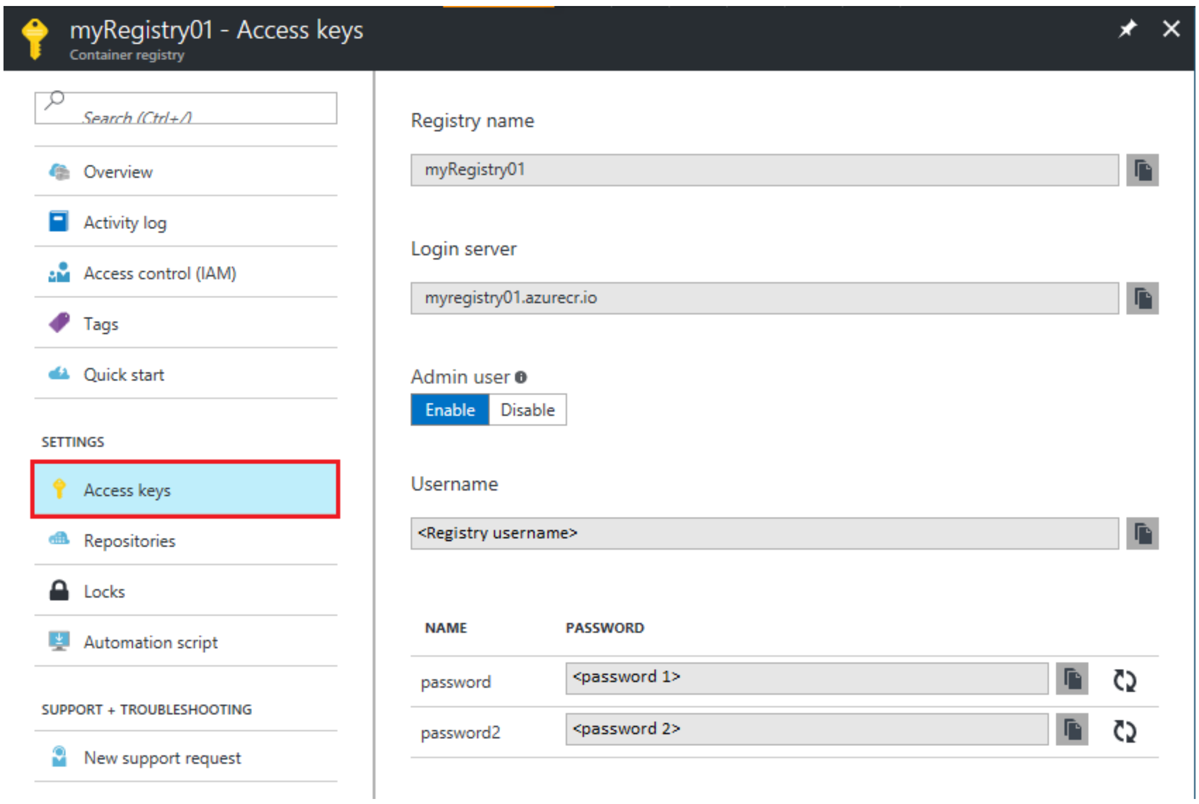
Subscriptions: All 17 selected

myRegistry | All subscriptions

1 items

NAME	LOCATION	RESOURCE GROUP	SUBSCRIPTION
myRegistry01	South Central US	<resource group>	<subscription ID>

2. To manage access settings, click **Access key**.



Registry name: myRegistry01

Login server: myregistry01.azurecr.io

Admin user: [✓ Enable] Disable

Username: <Registry username>

NAME	PASSWORD
password	<password 1>
password2	<password 2>

3. Note the following settings:

   - **Login server** - The fully qualified name you use to log in to the registry. In this example, it is `myregistry01.azurecr.io`.
   - **Admin user** - Toggle to enable or disable the registry's admin user account.
   - **Username** and **Password** - The credentials of the admin user account (if enabled) you can use to log in to the registry. You can optionally regenerate the passwords. Two passwords are created so that you can maintain connections to the registry by using one password while you regenerate the other password. To authenticate with a service principal instead, see Authenticate with a private Docker container registry.

Next steps

- Push your first image using the Docker CLI

Create a private Docker container registry using the Azure CLI 2.0

6/27/2017 • 3 min to read • Edit Online

Use commands in the Azure CLI 2.0 to create a container registry and manage its settings from your Linux, Mac, or Windows computer. You can also create and manage container registries using the Azure portal or programmatically with the Container Registry REST API.

- For background and concepts, see the overview
- For help on Container Registry CLI commands (`az acr` commands), pass the `-h` parameter to any command.

Prerequisites

- **Azure CLI 2.0**: To install and get started with the CLI 2.0, see the installation instructions. Log in to your Azure subscription by running `az login`. For more information, see Get started with the CLI 2.0.
- **Resource group**: Create a resource group before creating a container registry, or use an existing resource group. Make sure the resource group is in a location where the Container Registry service is available. To create a resource group using the CLI 2.0, see the CLI 2.0 reference.
- **Storage account** (optional): Create a standard Azure storage account to back the container registry in the same location. If you don't specify a storage account when creating a registry with `az acr create`, the command creates one for you. To create a storage account using the CLI 2.0, see the CLI 2.0 reference. Currently Premium Storage is not supported.
- **Service principal** (optional): When you create a registry with the CLI, by default it is not set up for access. Depending on your needs, you can assign an existing Azure Active Directory service principal to a registry (or create and assign a new one), or enable the registry's admin user account. See the sections later in this article. For more information about registry access, see Authenticate with the container registry.

Create a container registry

Run the `az acr create` command to create a container registry.

TIP

When you create a registry, specify a globally unique top-level domain name, containing only letters and numbers. The registry name in the examples is `myRegistry1`, but substitute a unique name of your own.

The following command uses the minimal parameters to create container registry `myRegistry1` in the resource group `myResourceGroup`, and using the *Basic* sku:

```
az acr create --name myRegistry1 --resource-group myResourceGroup --sku Basic
```

- `--storage-account-name` is optional. If not specified, a storage account is created with a name consisting of the registry name and a timestamp in the specified resource group.

When the registry is created, the output is similar to the following:

```
{
  "adminUserEnabled": false,
  "creationDate": "2017-06-06T18:36:29.124842+00:00",
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/myResourceGroup/providers/Microsoft.ContainerRegistry
/registries/myRegistry1",
  "location": "southcentralus",
  "loginServer": "myregistry1.azurecr.io",
  "name": "myRegistry1",
  "provisioningState": "Succeeded",
  "sku": {
    "name": "Basic",
    "tier": "Basic"
  },
  "storageAccount": {
    "name": "myregistry123456789"
  },
  "tags": {},
  "type": "Microsoft.ContainerRegistry/registries"
}
```

Take special note:

- `id` - Identifier for the registry in your subscription, which you need if you want to assign a service principal.
- `loginServer` - The fully qualified name you specify to log in to the registry. In this example, the name is `myregistry1.exp.azurecr.io` (all lowercase).

Assign a service principal

Use CLI 2.0 commands to assign an Azure Active Directory service principal to a registry. The service principal in these examples is assigned the Owner role, but you can assign other roles if you want.

Create a service principal and assign access to the registry

In the following command, a new service principal is assigned Owner role access to the registry identifier passed with the `--scopes` parameter. Specify a strong password with the `--password` parameter.

```
az ad sp create-for-rbac --scopes /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/myresourcegroup/providers/Microsoft.ContainerRegistry/registries/myregistry1 --
role Owner --password myPassword
```

Assign an existing service principal

If you already have a service principal and want to assign it Owner role access to the registry, run a command similar to the following example. You pass the service principal app ID using the `--assignee` parameter:

```
az role assignment create --scope /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/myresourcegroup/providers/Microsoft.ContainerRegistry/registries/myregistry1 --
role Owner --assignee myAppId
```

Manage admin credentials

An admin account is automatically created for each container registry and is disabled by default. The following examples show `az acr` CLI commands to manage the admin credentials for your container registry.

Obtain admin user credentials

```
az acr credential show -n myRegistry1
```

Enable admin user for an existing registry

```
az acr update -n myRegistry1 --admin-enabled true
```

Disable admin user for an existing registry

```
az acr update -n myRegistry1 --admin-enabled false
```

List images and tags

Use the `az acr` CLI commands to query the images and tags in a repository.

NOTE

Currently, Container Registry does not support the `docker search` command to query for images and tags.

List repositories

The following example lists the repositories in a registry, in JSON (JavaScript Object Notation) format:

```
az acr repository list -n myRegistry1 -o json
```

List tags

The following example lists the tags on the **samples/nginx** repository, in JSON format:

```
az acr repository show-tags -n myRegistry1 --repository samples/nginx -o json
```

Next steps

- Push your first image using the Docker CLI

Create a private Docker container registry using the Azure PowerShell

6/27/2017 • 2 min to read • Edit Online

Use commands in Azure PowerShell to create a container registry and manage its settings from your Windows computer. You can also create and manage container registries using the Azure portal, the Azure CLI, or programmatically with the Container Registry REST API.

- For background and concepts, see the overview
- For a full list of supported cmdlets, see Azure Container Registry Management Cmdlets.

Prerequisites

- **Azure PowerShell**: To install and get started with Azure PowerShell, see the installation instructions. Log in to your Azure subscription by running `Login-AzureRMAccount`. For more information, see Get started with Azure PowerShell.
- **Resource group**: Create a resource group before creating a container registry, or use an existing resource group. Make sure the resource group is in a location where the Container Registry service is available. To create a resource group using Azure PowerShell, see the PowerShell reference.
- **Storage account** (optional): Create a standard Azure storage account to back the container registry in the same location. If you don't specify a storage account when creating a registry with `New-AzureRMContainerRegistry`, the command creates one for you. To create a storage account using PowerShell, see the PowerShell reference. Currently Premium Storage is not supported.
- **Service principal** (optional): When you create a registry with PowerShell, by default it is not set up for access. Depending on your needs, you can assign an existing Azure Active Directory service principal to a registry or create and assign a new one. Alternatively, you can enable the registry's admin user account. See the sections later in this article. For more information about registry access, see Authenticate with the container registry.

Create a container registry

Run the `New-AzureRMContainerRegistry` command to create a container registry.

TIP

When you create a registry, specify a globally unique top-level domain name, containing only letters and numbers. The registry name in the examples is `MyRegistry`, but substitute a unique name of your own.

The following command uses the minimal parameters to create container registry `MyRegistry` in the resource group `MyResourceGroup` in the South Central US location:

```
$Registry = New-AzureRMContainerRegistry -ResourceGroupName "MyResourceGroup" -Name "MyRegistry"
```

- `-StorageAccountName` is optional. If not specified, a storage account is created with a name consisting of the registry name and a timestamp in the specified resource group.

Assign a service principal

Use PowerShell commands to assign an Azure Active Directory service principal to a registry. The service principal in these examples is assigned the Owner role, but you can assign other roles if you want.

Create a service principal

In the following command, a new service principal is created. Specify a strong password with the `-Password` parameter.

```
$ServicePrincipal = New-AzureRMADServicePrincipal -DisplayName ApplicationDisplayName -Password "MyPassword"
```

Assign a new or existing service principal

You can assign a new or an existing service principal to a registry. To assign it Owner role access to the registry, run a command similar to the following example:

```
New-AzureRMRoleAssignment -RoleDefinitionName Owner -ServicePrincipalName $ServicePrincipal.ApplicationId -
Scope $Registry.Id
```

Sign in to the registry with the service principal

After assigning the service principal to the registry, you can sign in using the following command:

```
docker login -u $ServicePrincipal.ApplicationId -p myPassword
```

Manage admin credentials

An admin account is automatically created for each container registry and is disabled by default. The following examples show PowerShell commands to manage the admin credentials for your container registry.

Obtain admin user credentials

```
Get-AzureRMContainerRegistryCredential -ResourceGroupName "MyResourceGroup" -Name "MyRegistry"
```

Enable admin user for an existing registry

```
Update-AzureRMContainerRegistry -ResourceGroupName "MyResourceGroup" -Name "MyRegistry" -EnableAdminUser
```

Disable admin user for an existing registry

```
Update-AzureRMContainerRegistry -ResourceGroupName "MyResourceGroup" -Name "MyRegistry" -DisableAdminUser
```

Next steps

- Push your first image using the Docker CLI

Push your first image to a private Docker container registry using the Docker CLI

6/27/2017 • 2 min to read • Edit Online

An Azure container registry stores and manages private Docker container images, similar to the way Docker Hub stores public Docker images. You use the Docker Command-Line Interface (Docker CLI) for login, push, pull, and other operations on your container registry.

For more background and concepts, see the overview

Prerequisites

- **Azure container registry** - Create a container registry in your Azure subscription. For example, use the Azure portal or the Azure CLI 2.0.
- **Docker CLI** - To set up your local computer as a Docker host and access the Docker CLI commands, install Docker Engine.

Log in to a registry

Run `docker login` to log in to your container registry with your registry credentials.

The following example passes the ID and password of an Azure Active Directory service principal. For example, you might have assigned a service principal to your registry for an automation scenario.

```
docker login myregistry.azurecr.io -u xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -p myPassword
```

TIP

Make sure to specify the fully qualified registry name (all lowercase). In this example, it is `myregistry.azurecr.io`.

Steps to pull and push an image

The follow example downloads the Nginx image from the public Docker Hub registry, tags it for your private Azure container registry, pushes it to your registry, then pulls it again.

1. Pull the Docker official image for Nginx

First pull the public Nginx image to your local computer.

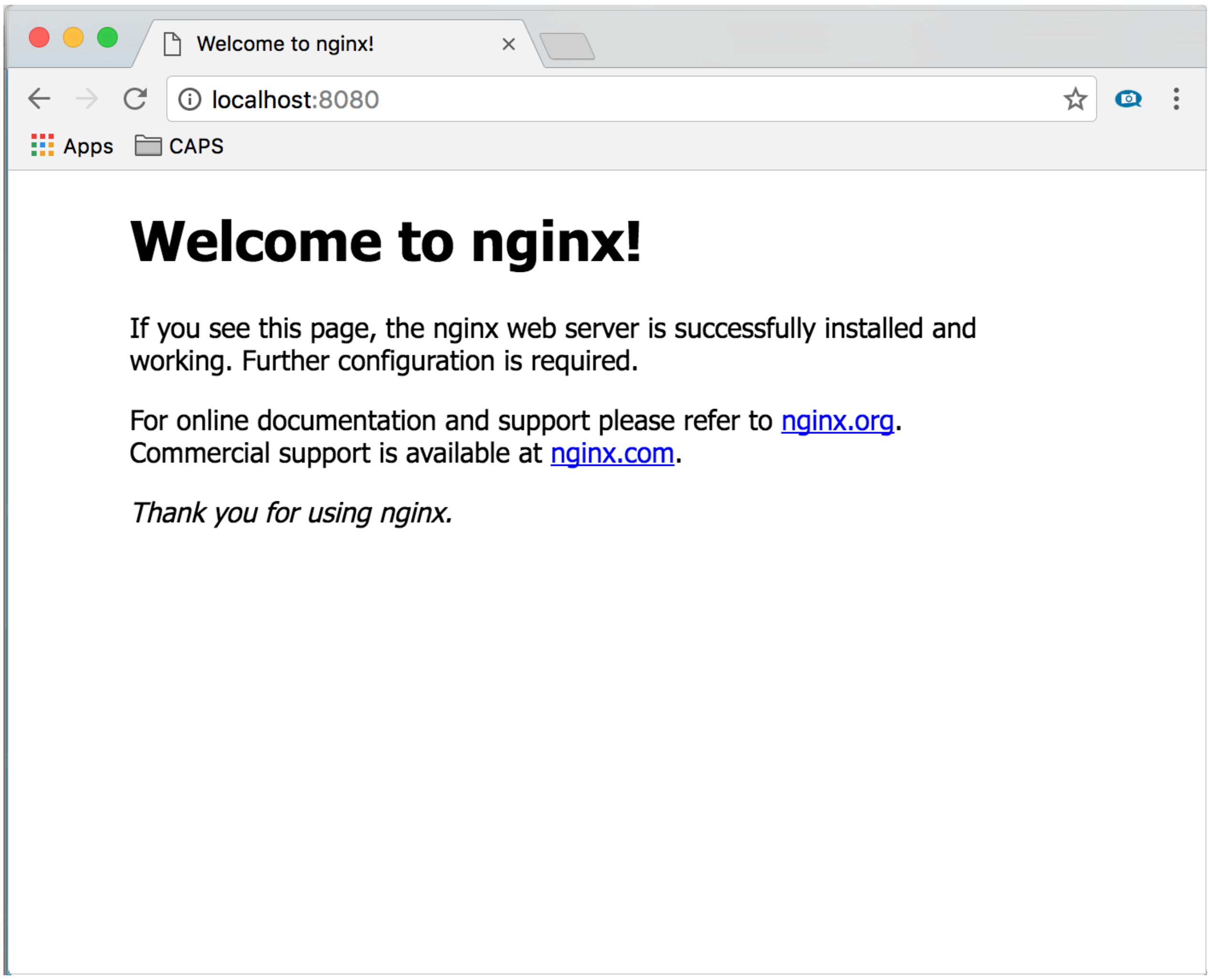
```
docker pull nginx
```

2. Start the Nginx container

The following command starts the local Nginx container interactively on port 8080, allowing you to see output from Nginx. It removes the running container once stopped.

```
docker run -it --rm -p 8080:80 nginx
```

Browse to http://localhost:8080 to view the running container. You see a screen similar to the following one.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

To stop the running container, press [CTRL]+[C].

3. Create an alias of the image in your registry

The following command creates an alias of the image, with a fully qualified path to your registry. This example specifies the `samples` namespace to avoid clutter in the root of the registry.

```
docker tag nginx myregistry.azurecr.io/samples/nginx
```

4. Push the image to your registry

```
docker push myregistry.azurecr.io/samples/nginx
```

5. Pull the image from your registry

```
docker pull myregistry.azurecr.io/samples/nginx
```

6. Start the Nginx container from your registry

```
docker run -it --rm -p 8080:80 myregistry.azurecr.io/samples/nginx
```

Browse to http://localhost:8080 to view the running container.

To stop the running container, press [CTRL]+[C].

7. (Optional) Remove the image

```
docker rmi myregistry.azurecr.io/samples/nginx
```

Concurrent Limits

In some scenarios, executing calls concurrently might result in errors. The following table contains the limits of concurrent calls with "Push" and "Pull" operations on Azure container registry:

OPERATION	LIMIT
PULL	Up to 10 concurrent pulls per registry
PUSH	Up to 5 concurrent pushes per registry

Next steps

Now that you know the basics, you are ready to start using your registry! For example, start deploying container images to an Azure Container Service cluster.

Azure container registry repositories

6/27/2017 • 1 min to read • Edit Online

Azure container registry allows you to store container images in repositories. By storing images in repositories, you can have groups of images (or version of images) in isolated environments. You can specify these repositories when you push images to your registry.

Prerequisites

- **Azure container registry** - Create a container registry in your Azure subscription. For example, use the Azure portal or the Azure CLI 2.0.
- **Docker CLI** - To set up your local computer as a Docker host and access the Docker CLI commands, install Docker Engine.
- **Pull an image** - Pull an image from the public Docker Hub registry, tag it, and push it to your registry. For guidance on how push and pull images, see Push Docker image to Azure private registry.

Viewing repositories in the Portal

Once you have pushed images to your container registry, you can see a list of the repositories hosting the images in the Azure portal.

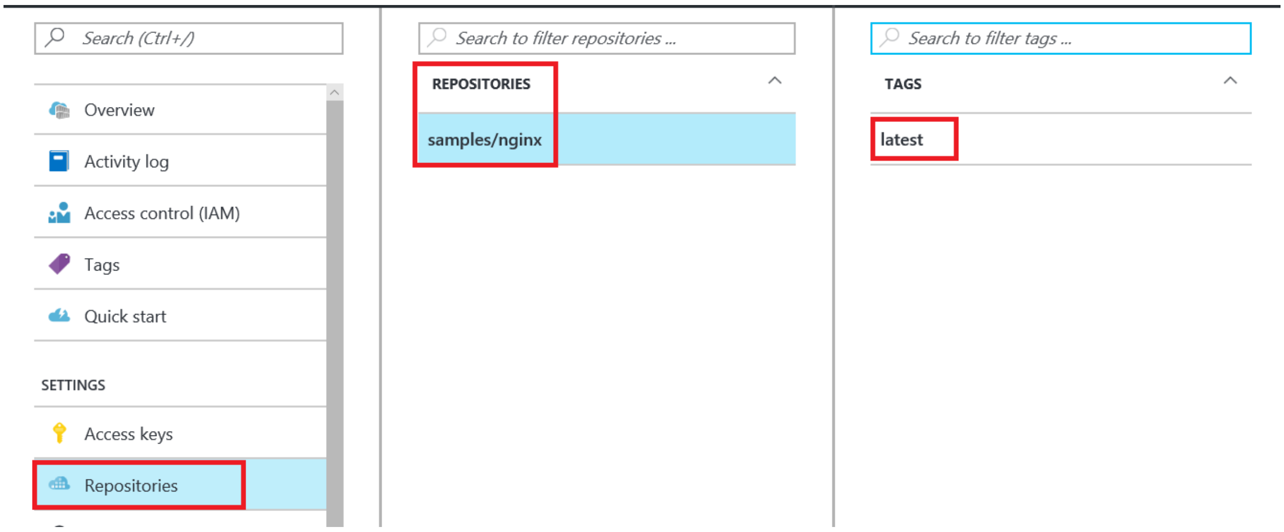
If you followed the steps in the Push Docker image to Azure private registry article, you should now have a Nginx image in your container registry. As part of the instructions, you should have specified a namespace for the image. In the example below, the command pushes the NGinx image to the "samples" repository:

```
docker push myregistry.azurecr.io/samples/nginx
```

Azure Container Registry supports multilevel repository namespaces. This feature enables you to group collections of images related to a specific app, or a collection of apps to specific development or operational teams. To read more about repositories in container registries, see Private Docker container registries in Azure.

To view the container registry repositories:

1. Log in to the Azure portal
2. On the **Azure Container Registry** blade, select the registry you wish to inspect
3. In the registry blade, click **Repositories** to see a list of all the repositories and their images
4. (Optional) Select a specific image to see tags



Next steps

Now that you know the basics, you are ready to start using your registry! For example, start deploying container images to an Azure Container Service cluster.

Use portal to create an Azure Active Directory application and service principal that can access resources

6/27/2017 • 6 min to read • Edit Online

When you have an application that needs to access or modify resources, you must set up an Azure Active Directory (AD) application and assign the required permissions to it. This approach is preferable to running the app under your own credentials because:

- You can assign permissions to the app identity that are different than your own permissions. Typically, these permissions are restricted to exactly what the app needs to do.
- You do not have to change the app's credentials if your responsibilities change.
- You can use a certificate to automate authentication when executing an unattended script.

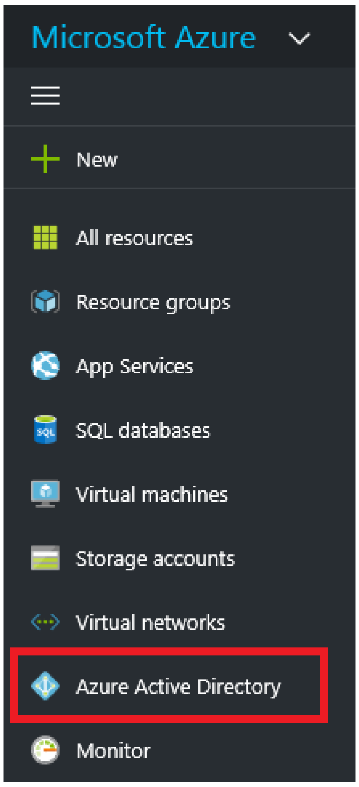
This topic shows you how to perform those steps through the portal. It focuses on a single-tenant application where the application is intended to run within only one organization. You typically use single-tenant applications for line-of-business applications that run within your organization.

Required permissions

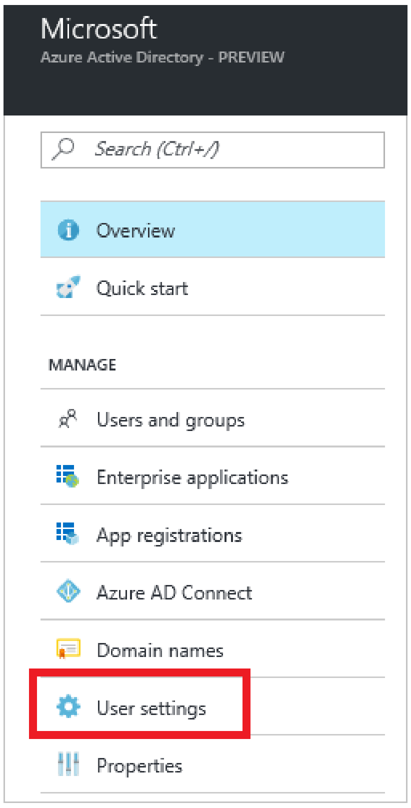
To complete this topic, you must have sufficient permissions to register an application with your Azure AD tenant, and assign the application to a role in your Azure subscription. Let's make sure you have the right permissions to perform those steps.

Check Azure Active Directory permissions

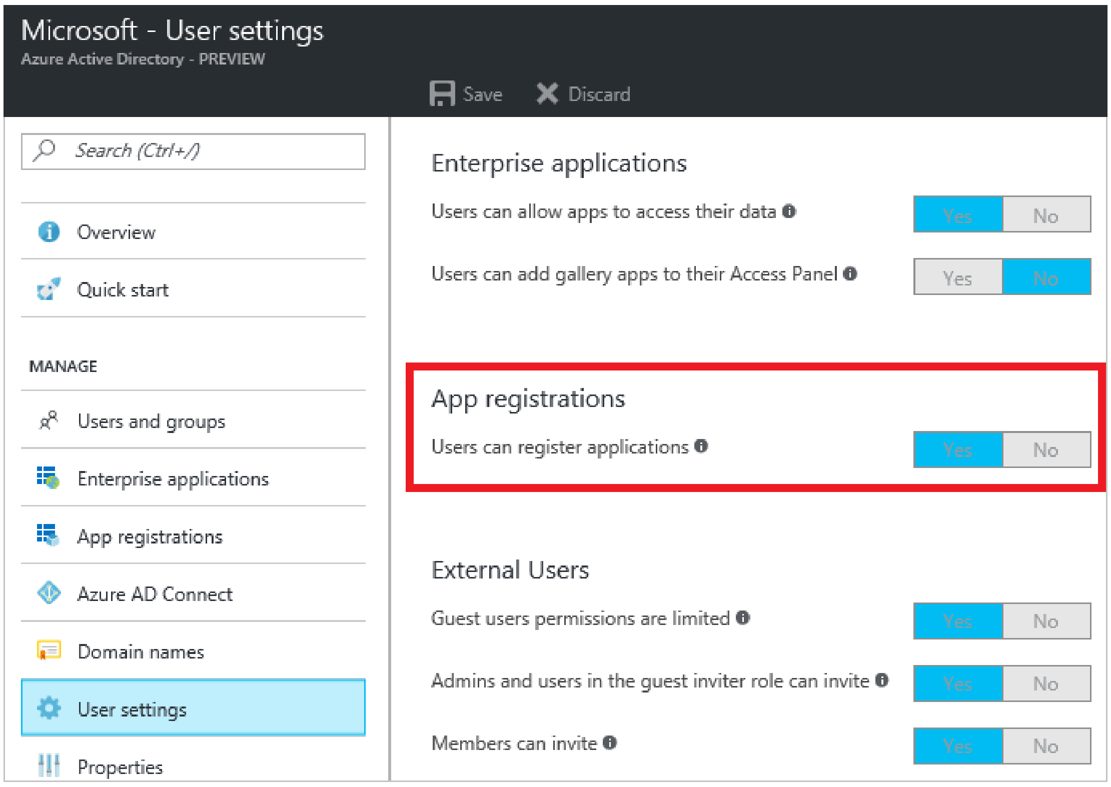
1. Log in to your Azure Account through the Azure portal.
2. Select **Azure Active Directory**.



3. In Azure Active Directory, select **User settings**.



4. Check the **App registrations** setting. If set to **Yes**, non-admin users can register AD apps. This setting means any user in the Azure AD tenant can register an app. You can proceed to Check Azure subscription permissions.



Enterprise applications

Users can allow apps to access their data [✓ Yes] No

Users can add gallery apps to their Access Panel Yes [✓ No]

App registrations

Users can register applications [✓ Yes] No

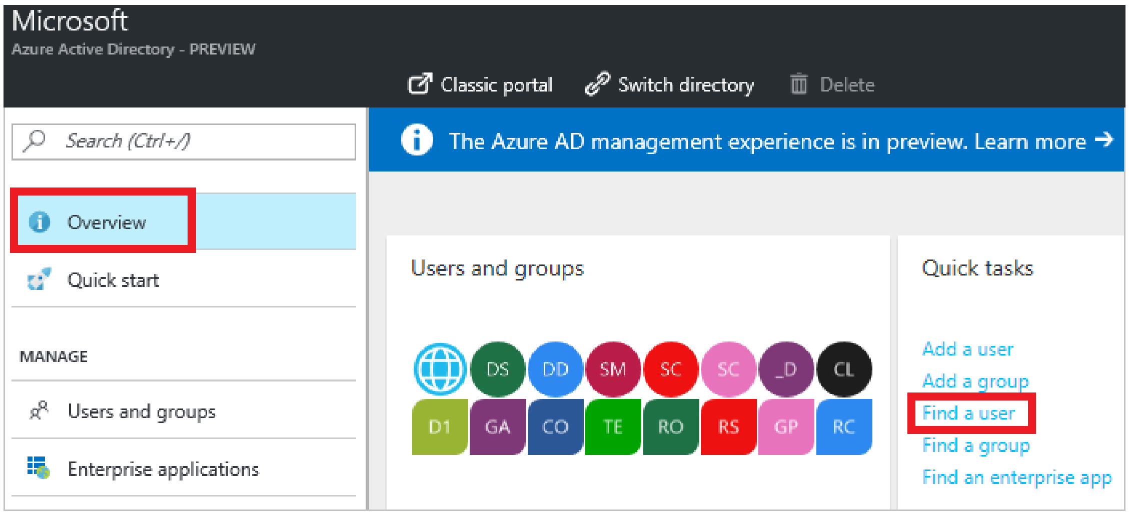
External Users

Guest users permissions are limited [✓ Yes] No

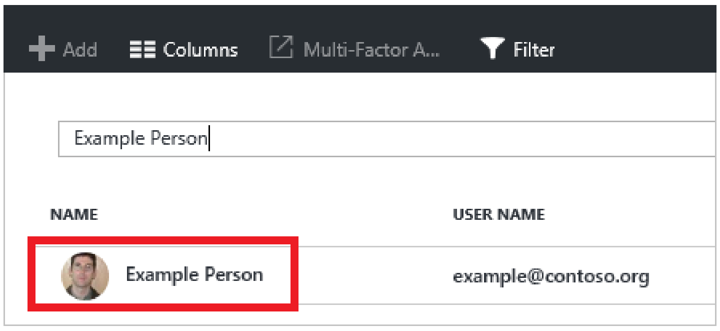
Admins and users in the guest inviter role can invite [✓ Yes] No

Members can invite [✓ Yes] No

5. If the app registrations setting is set to **No**, only admin users can register apps. You need to check whether your account is an admin for the Azure AD tenant. Select **Overview** and **Find a user** from Quick tasks.

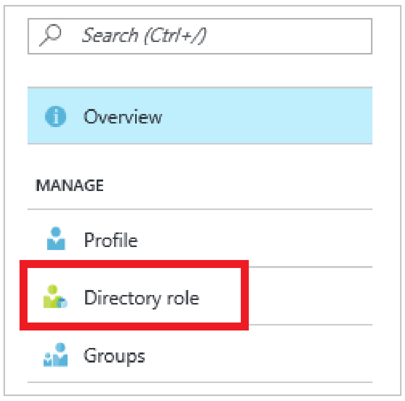


6. Search for your account, and select it when you find it.

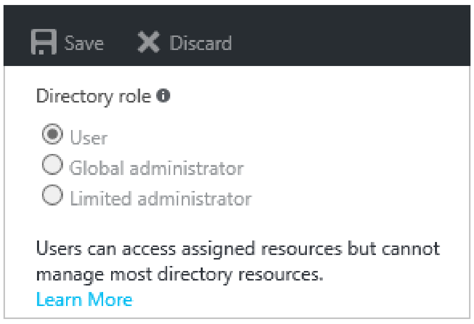


NAME	USER NAME
Example Person	example@contoso.org

7. For your account, select **Directory role**.



8. View your assigned directory role in Azure AD. If your account is assigned to the User role, but the app registration setting (from the preceding steps) is limited to admin users, ask your administrator to either assign you to an administrator role, or to enable users to register apps.



Directory role

[✓ User]
Global administrator
Limited administrator

Users can access assigned resources but cannot manage most directory resources.

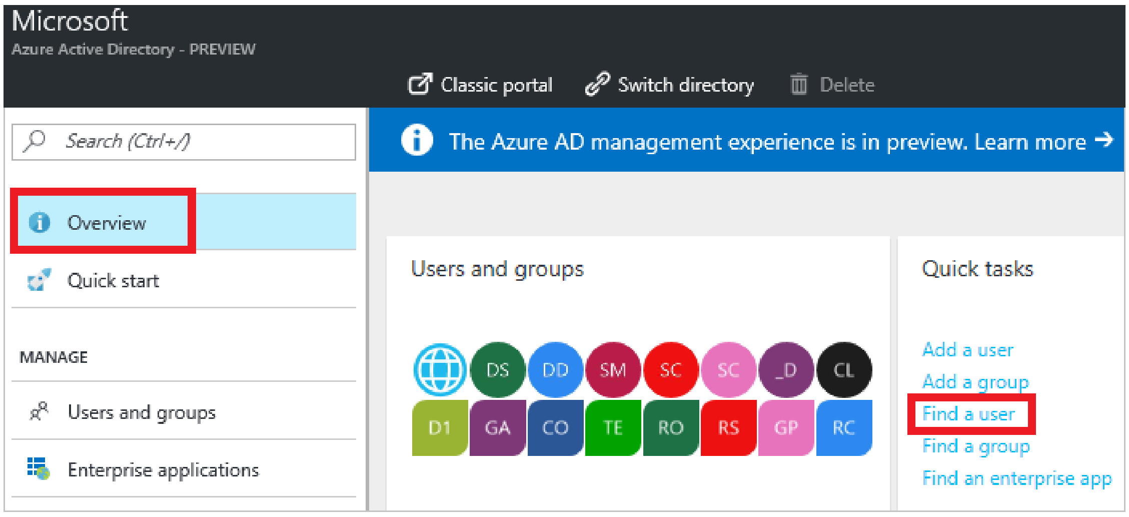
Check Azure subscription permissions

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

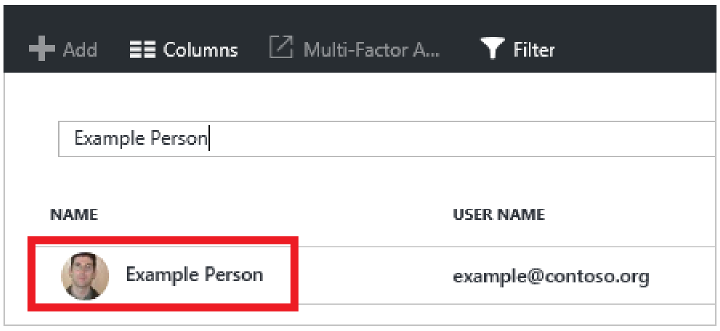
To check your subscription permissions:

1. If you are not already looking at your Azure AD account from the preceding steps, select **Azure Active Directory** from the left pane.

2. Find your Azure AD account. Select **Overview** and **Find a user** from Quick tasks.

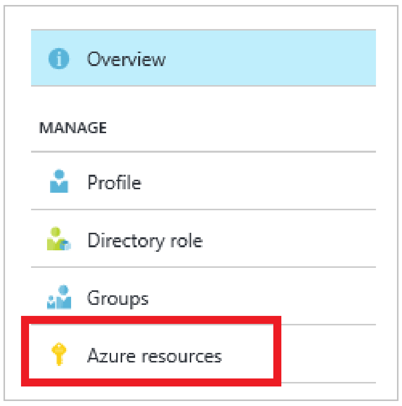


3. Search for your account, and select it when you find it.

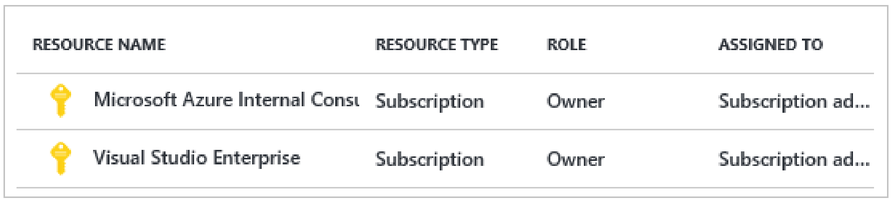


NAME	USER NAME
Example Person	example@contoso.org

4. Select **Azure resources**.



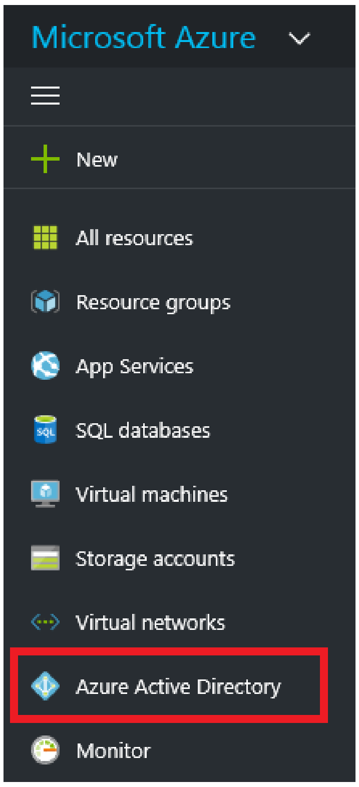
5. View your assigned roles, and determine if you have adequate permissions to assign an AD app to a role. If not, ask your subscription administrator to add you to User Access Administrator role. In the following image, the user is assigned to the Owner role for two subscriptions, which means that user has adequate permissions.



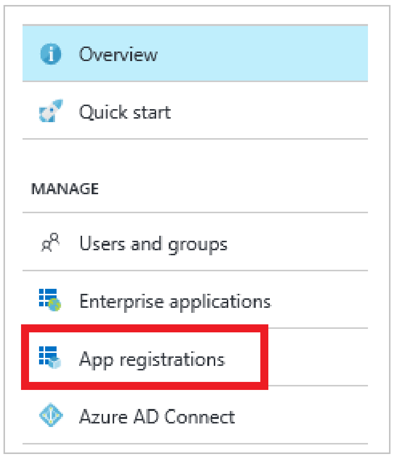
RESOURCE NAME	RESOURCE TYPE	ROLE	ASSIGNED TO
Microsoft Azure Internal Consu	Subscription	Owner	Subscription ad...
Visual Studio Enterprise	Subscription	Owner	Subscription ad...

Create an Azure Active Directory application

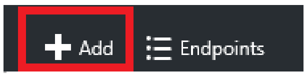
1. Log in to your Azure Account through the Azure portal.
2. Select **Azure Active Directory**.



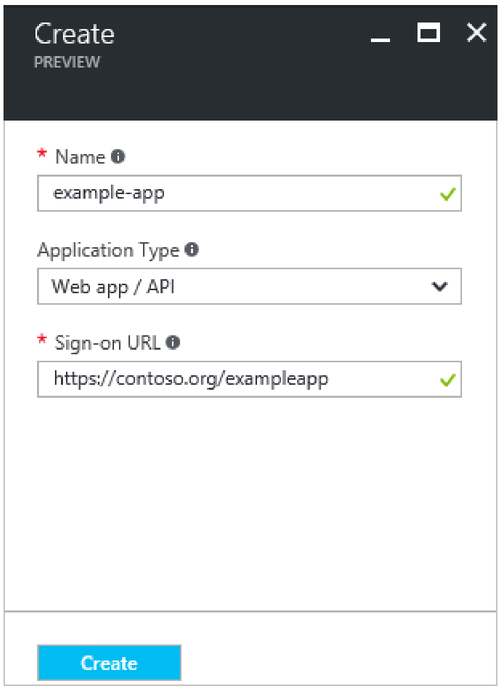
3. Select **App registrations**.



4. Select **Add**.



5. Provide a name and URL for the application. Select either **Web app / API** or **Native** for the type of application you want to create. After setting the values, select **Create**.


Create
PREVIEW

* Name
example-app

Application Type
Web app / API

* Sign-on URL
https://contoso.org/exampleapp

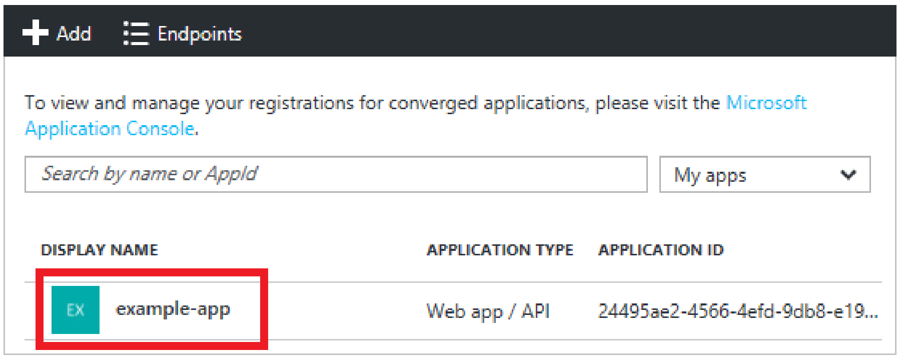
Create

You have created your application.

Get application ID and authentication key

When programmatically logging in, you need the ID for your application and an authentication key. To get those values, use the following steps:

1. From **App registrations** in Azure Active Directory, select your application.

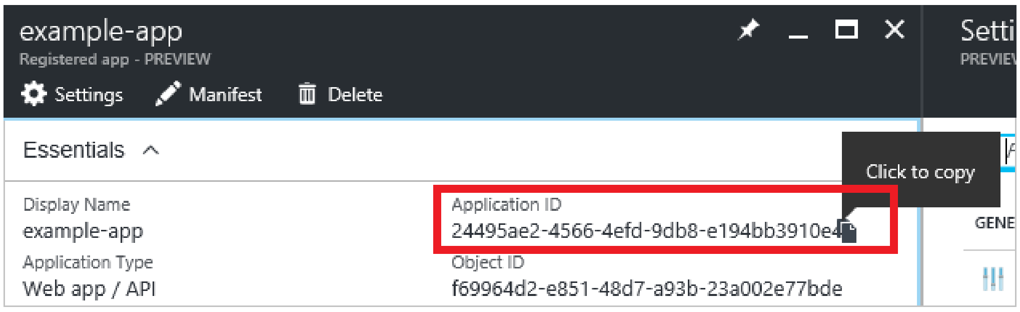

Add   Endpoints

To view and manage your registrations for converged applications, please visit the Microsoft Application Console.

Search by name or AppId   My apps

DISPLAY NAME	APPLICATION TYPE	APPLICATION ID
EX example-app	Web app / API	24495ae2-4566-4efd-9db8-e19...

2. Copy the **Application ID** and store it in your application code. The applications in the sample applications section refer to this value as the client id.

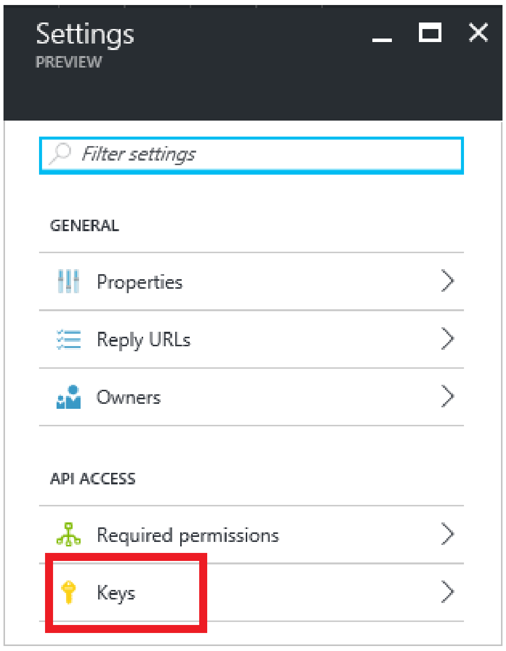

example-app
Registered app - PREVIEW
Settings   Manifest   Delete

Essentials

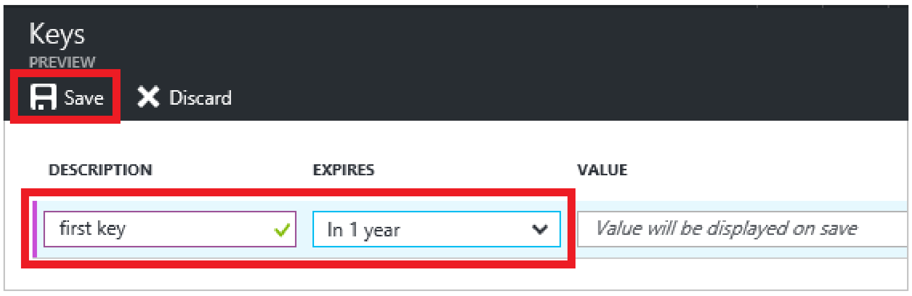
Click to copy

Display Name example-app	Application ID 24495ae2-4566-4efd-9db8-e194bb3910e4
Application Type Web app / API	Object ID f69964d2-e851-48d7-a93b-23a002e77bde

3. To generate an authentication key, select **Keys**.

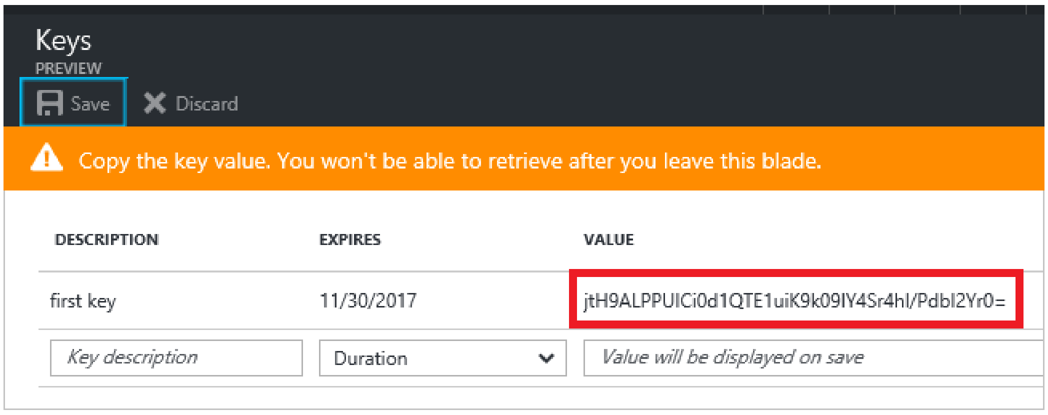


4. Provide a description of the key, and a duration for the key. When done, select **Save**.



DESCRIPTION	EXPIRES	VALUE
first key	In 1 year	Value will be displayed on save

After saving the key, the value of the key is displayed. Copy this value because you are not able to retrieve the key later. You provide the key value with the application ID to log in as the application. Store the key value where your application can retrieve it.



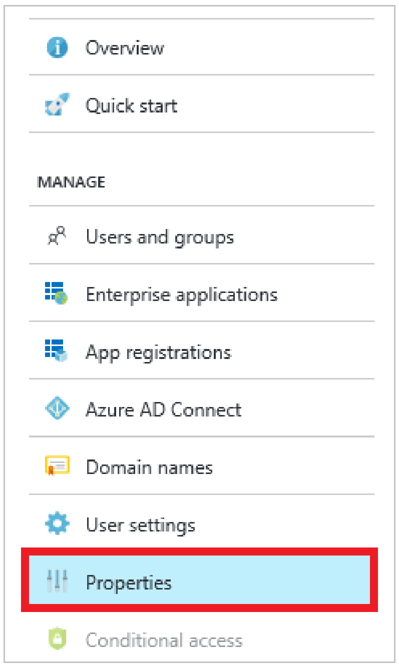
Copy the key value. You won't be able to retrieve after you leave this blade.

DESCRIPTION	EXPIRES	VALUE
first key	11/30/2017	jtH9ALPPUICi0d1QTE1uiK9k09lY4Sr4hl/PdbI2Yr0=
Key description	Duration	Value will be displayed on save

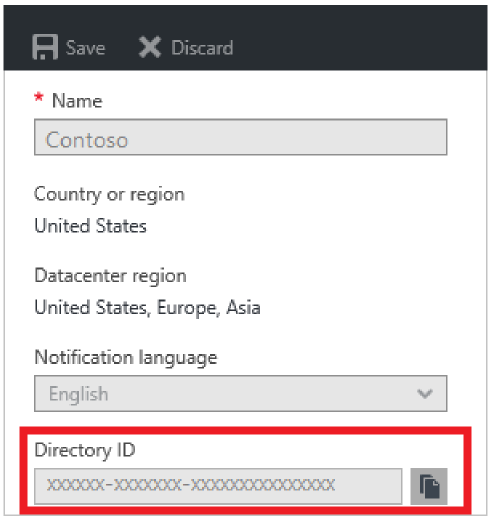
Get tenant ID

When programmatically logging in, you need to pass the tenant ID with your authentication request.

1. To get the tenant ID, select **Properties** for your Azure AD tenant.



2. Copy the **Directory ID**. This value is your tenant ID.



* Name
Contoso

Country or region
United States

Datacenter region
United States, Europe, Asia

Notification language
English

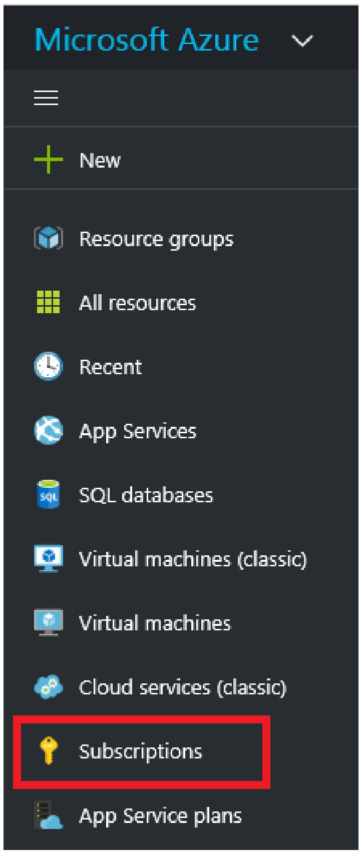
Directory ID
xxxxxx-xxxxxxx-xxxxxxxxxxxxxxx

Assign application to role

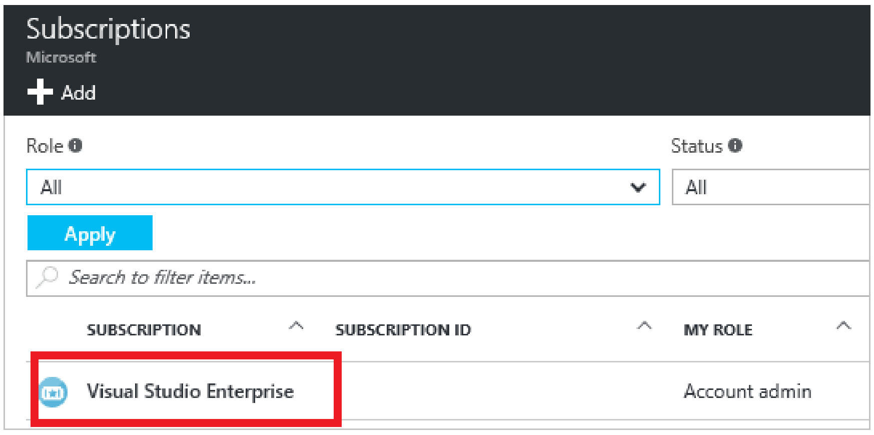
To access resources in your subscription, you must assign the application to a role. Decide which role represents the right permissions for the application. To learn about the available roles, see RBAC: Built in Roles.

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

1. Navigate to the level of scope you wish to assign the application to. For example, to assign a role at the subscription scope, select **Subscriptions**. You could instead select a resource group or resource.

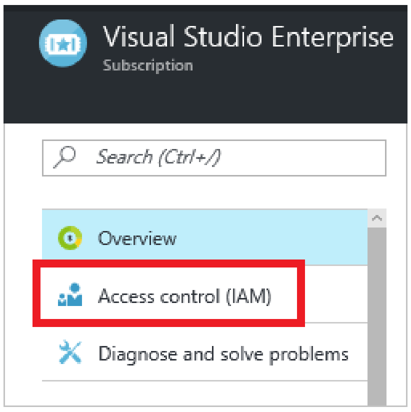


2. Select the particular subscription (resource group or resource) to assign the application to.

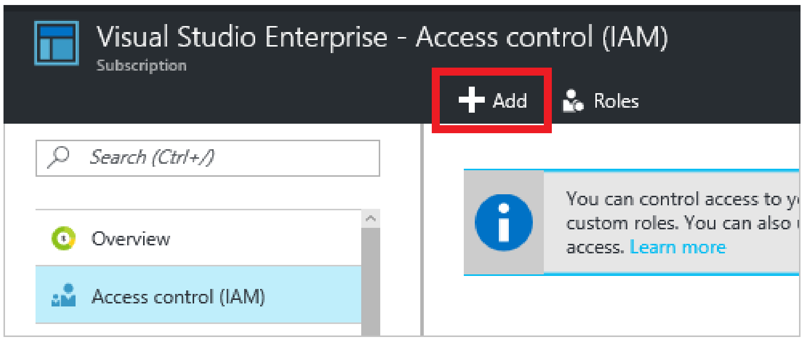


SUBSCRIPTION	SUBSCRIPTION ID	MY ROLE
Visual Studio Enterprise		Account admin

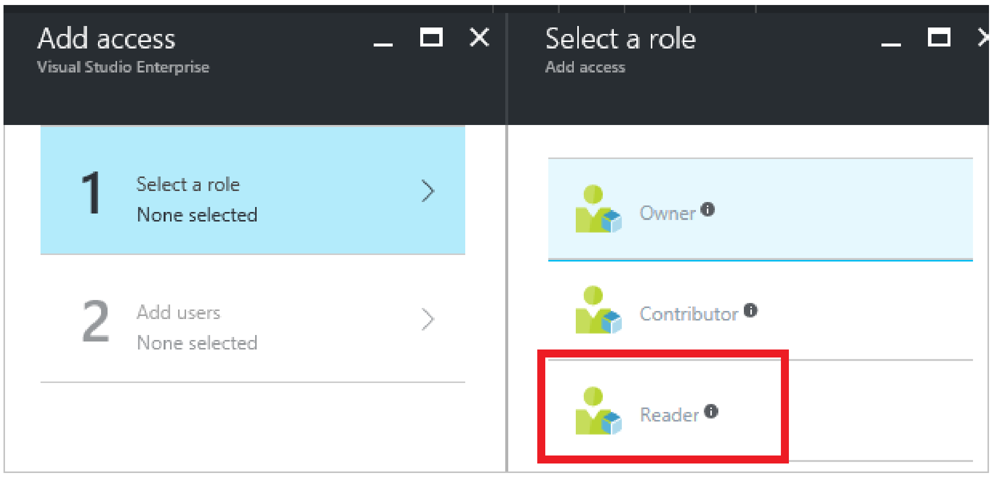
3. Select **Access Control (IAM)**.



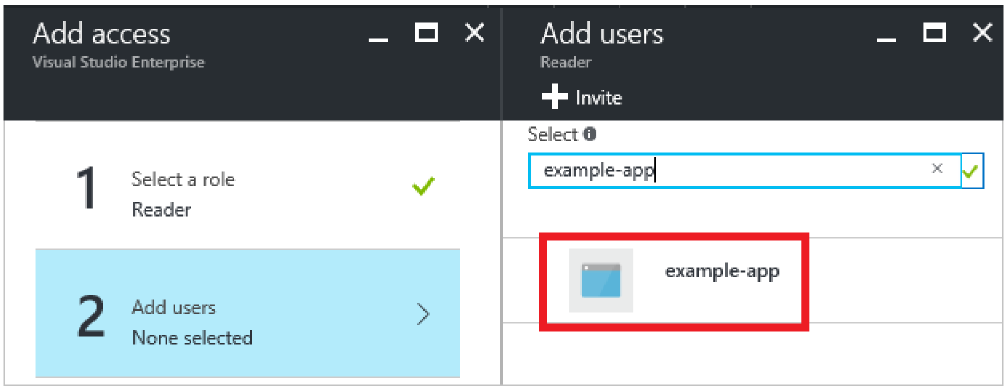
4. Select **Add**.


Visual Studio Enterprise - Access control (IAM)
Subscription
Add
Roles
Search (Ctrl+/)
Overview
Access control (IAM)
You can control access to y custom roles. You can also access. Learn more

5. Select the role you wish to assign to the application. The following image shows the **Reader** role.



6. Search for your application, and select it.



7. Select **OK** to finish assigning the role. You see your application in the list of users assigned to a role for that scope.

Log in as the application

Your application is now set up in Azure Active Directory. You have an ID and key to use for signing in as the application. The application is assigned to a role that gives it certain actions it can perform.

To log in through PowerShell, see Provide credentials through PowerShell.

To log in through Azure CLI, see Provide credentials through Azure CLI.

To get the access token for REST operations, see Create the request.

Look at the following sample applications to learn about logging in through application code.

Sample applications

The following sample applications show how to log in as the AD application.

.NET

- Deploy an SSH Enabled VM with a Template with .NET
- Manage Azure resources and resource groups with .NET

Java

- Getting Started with Resources - Deploy Using Azure Resource Manager Template - in Java
- Getting Started with Resources - Manage Resource Group - in Java

Python

- Deploy an SSH Enabled VM with a Template in Python
- Managing Azure Resource and Resource Groups with Python

Node.js

- Deploy an SSH Enabled VM with a Template in Node.js
- Manage Azure resources and resource groups with Node.js

Ruby

- Deploy an SSH Enabled VM with a Template in Ruby
- Managing Azure Resource and Resource Groups with Ruby

Next Steps

- To set up a multi-tenant application, see Developer's guide to authorization with the Azure Resource Manager API.
- To learn about specifying security policies, see Azure Role-based Access Control.
- For a list of available actions that can be granted or denied to users, see Azure Resource Manager Resource Provider operations.

Authenticate with a private Docker container registry

6/27/2017 • 2 min to read • Edit Online

To work with container images in an Azure container registry, you log in using the `docker login` command. You can log in using either an **Azure Active Directory service principal** or a registry-specific **admin account**. This article provides more detail about these identities.

Service principal

You can assign a service principal to your registry and use it for basic Docker authentication. Using a service principal is recommended for most scenarios. Provide the app ID and password of the service principal to the `docker login` command, as shown in the following example:

```
docker login myregistry.azurecr.io -u xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -p myPassword
```

Once logged in, Docker caches the credentials, so you don't need to remember the app ID.

TIP
If you want, you can regenerate the password of a service principal by running the `az ad sp reset-credentials` command.

Service principals allow role-based access to a registry. Available roles are:

- Reader (pull only access).
- Contributor (pull and push).
- Owner (pull, push, and assign roles to other users).

Anonymous access is not available on Azure Container Registries. For public images you can use Docker Hub.

You can assign multiple service principals to a registry, which allows you to define access for different users or applications. Service principals also enable "headless" connectivity to a registry in developer or DevOps scenarios such as the following examples:

- Container deployments from a registry to orchestration systems including DC/OS, Docker Swarm and Kubernetes. You can also pull container registries to related Azure services such as Container Service, App Service, Batch, Service Fabric, and others.

- Continuous integration and deployment solutions (such as Visual Studio Team Services or Jenkins) that build container images and push them to a registry.

Admin account

With each registry you create, an admin account gets created automatically. By default the account is disabled, but you can enable it and manage the credentials, for example through the portal or using the Azure CLI 2.0 commands. Each admin account is provided with two passwords, both of which can be regenerated. The two passwords allow you to maintain connections to the registry by using one password while you regenerate the other password. If the account is enabled, you can pass the user name and either password to the `docker login` command for basic authentication to the registry. For example:

```
docker login myregistry.azurecr.io -u myAdminName -p myPassword1
```

IMPORTANT

The admin account is designed for a single user to access the registry, mainly for test purposes. It is not recommended to share the admin account credentials among other users. All users appear as a single user to the registry. Changing or disabling this account disables registry access for all users who use the credentials.

Next steps

- Push your first image using the Docker CLI.
- For more information about authentication in the Container Registry preview, see the blog post.

Azure container registry repositories

6/27/2017 • 1 min to read • Edit Online

Azure Container Registries are compatible with a multitude of services and orchestrators. To make it easier to track the source services and agents from which ACR is used, we have started using the Docker header field in the Docker.config file.

Viewing repositories in the Portal

The ACR headers follow the format:

```
X-Meta-Source-Client: <cloud>/<service>/<optionalservicename>
```

- Cloud: Azure, Azure Stack, or other government or country-specific Azure clouds. Although Azure Stack and government clouds are not currently supported, this parameter enables future support.
- Service: name of the service.
- Optionalservicename: optional parameter for services with subservices, or to specify a SKU (ex: web apps correspond with Azure/app-service/web-apps).

Partner services and orchestrators are encouraged to use specific header values to help with our telemetry. Users can also modify the value passed to the header if they so desire.

The values we want ACR partners to use to populate the "X-Meta-Source-Client" field are below:

SERVICE NAME	HEADER
Azure Container Service	azure/compute/azure-container-service
App Service - Web Apps	azure/app-service/web-apps
App Service - Logic Apps	azure/app-service/logic-apps
Batch	azure/compute/batch
Cloud Console	azure/cloud-console
Functions	azure/compute/functions
Internet of Things - Hub	azure/iot/hub
HDInsight	azure/data/hdinsight
Jenkins	azure/jenkins
Machine Learning	azure/data/machile-learning
Service Fabric	azure/compute/service-fabric
VSTS	azure/vsts

Next steps

Learn more about registries and the supported services and orchestrators