

Microsoft Azure Digital Twins Tutorial

Disclaimer

The authors are not affiliated in any way with Microsoft and are not assuming any responsibility or liability for any errors or omissions in the content of this tutorial. The described information is subject to change and is provided with no guarantees of completeness, accuracy, usefulness, or timeliness. When using the described service, monitor your spending using the “Cost Management”.

1. Getting started

First, visit the official website for information on the basics of the service as well as the current pricing model. As of January 2023, Azure Digital Twins (ADT) is a paid service that is part of the Microsoft Azure platform, but there are two ways of using it for free, at least temporarily: newly created standard Azure accounts get \$200 of credit that can be used for ADT but must be spent within the first 30 days, while with student accounts, \$100 of credit are included per year.

Product overview: <https://azure.microsoft.com/en-us/products/digital-twins/>

Pricing: <https://azure.microsoft.com/en-us/pricing/details/digital-twins/>

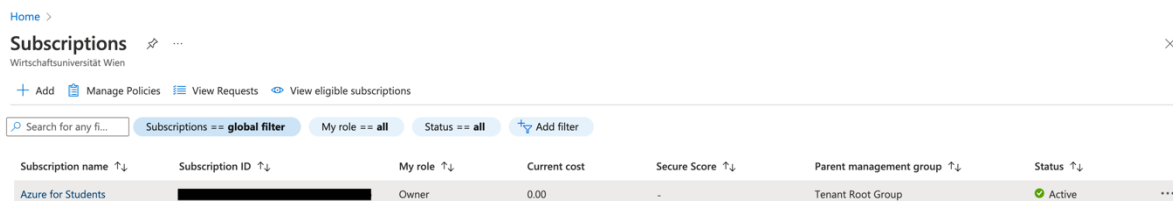
Standard account: <https://azure.microsoft.com/en-us/free/>

Student account: <https://azure.microsoft.com/en-us/free/students/>

2. Setting up an account

To start using Azure, click “Start free” on the “standard account” or “student account” page linked above and follow the steps: log in to an existing Microsoft account or create a new one, fill in some additional information about yourself or your company and complete the registration.

After you have set up Azure and are logged in, make sure that you have an active Azure subscription by navigating to the “Subscriptions” service in the Azure portal via the “Azure services” overview or the search field.



Subscription name ↑↓	Subscription ID ↑↓	My role ↑↓	Current cost	Secure Score ↑↓	Parent management group ↑↓	Status ↑↓
Azure for Students		Owner	0.00	-	Tenant Root Group	Active

Subscriptions: https://portal.azure.com/#view/Microsoft_Azure_Billing/SubscriptionsBlade

3. Setting up the Azure prerequisites

Navigate to the “Azure Digital Twins” service in the Azure portal via the “Azure services” overview or the search field. Click “Create”, which will enable you to create a new resource instance. Select the Azure subscription and a resource group (if you start from scratch, click “Create new” and enter a name for a new resource group), enter a name of for the new resource instance and select the region, which will affect the pricing (see the pricing link in step 1) as it determines on which servers your instance will be run. Check the box “Assign Azure Digital Twins Data Owner Role” to grant yourself full access to the resource instance. Leave all default settings in the other tabs as they are for a simple first trial run with ADT. However, especially the “Networking” and “Advanced” tabs can be relevant for security reasons as the respective settings allow access restriction and should be revisited later after carefully studying the relevant documentation on security and private endpoints.

Home > Azure Digital Twins >

Create Resource

Azure Digital Twins

* Basics * Networking Advanced Tags Review + create

Create an Azure Digital Twins instance to start building connected solutions that model the real world. [Learn more](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folder to organize and manage all your resources.

Subscription * Azure for Students

Resource group * (New) My-Resource-Group

[Create new](#)

Instance Details

Resource name * My-Resource-Instance

Region * West Central US

Grant access to resource

To manage the elements within an instance, a user needs access to Azure Digital Twins data plane APIs. Select the suggested role below to grant yourself full access to the data plane APIs. You can also use Access Control (IAM) to choose appropriate roles later. [Learn more](#)

☒ Assign Azure Digital Twins Data Owner Role

[Review + create](#) < Previous Next: Networking >

Create resource instance: <https://portal.azure.com/#create/Microsoft.DigitalTwins>

Security: <https://learn.microsoft.com/en-us/azure/digital-twins/concepts-security>

Private endpoints: <https://learn.microsoft.com/en-us/azure/digital-twins/how-to-enable-private-link>

4. Modeling the digital twin

Before creating the digital twin, you must define a model using the Digital Twin Definition Language (DTDL), which follows JSON syntax and is saved with a “.json” file extension. Use the type “property” to be able to save and query the state of the twin later. Semantic types allow you to specify the unit of measurement. The twinned object or process is modeled as an interface and can consist of multiple components, which are also modeled as interfaces. Relationships can also be defined.

```
data > {} process_model.json > ...
1  {
2    {
3      "@id": "dtmi:aluminum_factory:packaging;1",
4      "@type": "Interface",
5      "@context": "dtmi:dtidl:context;2",
6      "displayName": "Aluminum Packaging Process",
7      "contents": [
8        {
9          "name": "cncProximitySensor",
10         "@type": "Component",
11         "schema": "dtmi:aluminum_factory:cnc_proximity_sensor;1"
12       },
13       {
14         "name": "cncEngineSpeedSensor",
15         "@type": "Component",
16         "schema": "dtmi:aluminum_factory:cnc_engine_speed_sensor;1"
17       },
18       {
19         "name": "cncPressureSensor",
20         "@type": "Component",
21         "schema": "dtmi:aluminum_factory:cnc_pressure_sensor;1"
22       }
23     ]
24   },
25   {
26     "@id": "dtmi:aluminum_factory:cnc_proximity_sensor;1",
27     "@type": "Interface",
28     "@context": "dtmi:dtidl:context;2",
29     "displayName": "CNC Proximity Sensor",
30     "contents": [
31       {
32         "name": "proximity",
33         "@type": ["Property", "Distance"],
34         "unit": "millimetre",
35         "schema": "float"
36       }
37     ]
38   },
39   {
40     "@id": "dtmi:aluminum_factory:cnc_engine_speed_sensor;1",
41     "@type": "Interface",
42     "@context": "dtmi:dtidl:context;2",
43     "displayName": "CNC Engine Speed Sensor",
44     "contents": [
45       {
46         "name": "engineSpeed",
47         "@type": ["Property", "AngularVelocity"],
48         "unit": "revolutionPerMinute",
49         "schema": "integer"
50       }
51     ]
52   },
53   {
54     "@id": "dtmi:aluminum_factory:cnc_pressure_sensor;1",
55     "@type": "Interface",
56     "@context": "dtmi:dtidl:context;2",
57     "displayName": "CNC Pressure Sensor",
58     "contents": [
59       {
60         "name": "pressure",
61         "@type": ["Property", "Pressure"],
62         "unit": "bar",
63         "schema": "integer"
64       }
65     ]
66   }
67 }
```

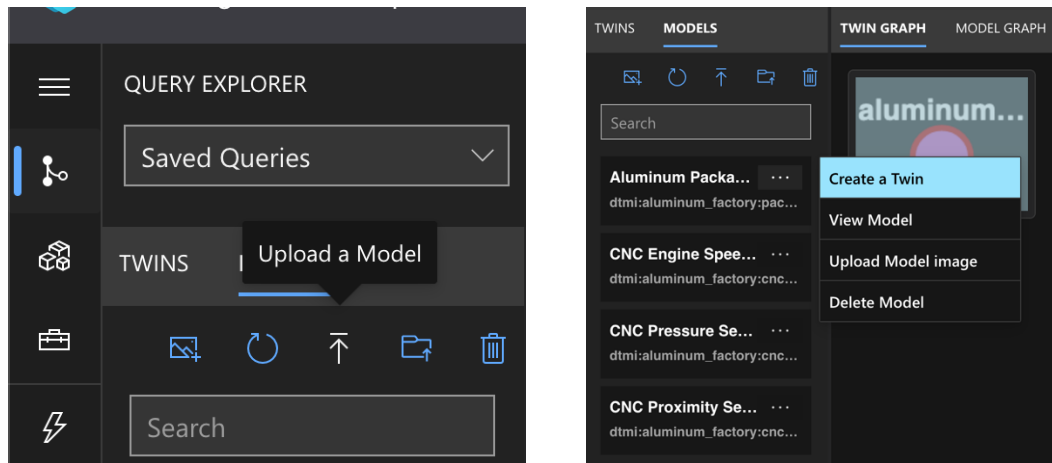
DTDL models: <https://learn.microsoft.com/en-us/azure/digital-twins/concepts-models>

Properties and telemetry: <https://learn.microsoft.com/en-us/azure/digital-twins/concepts-models#properties-and-telemetry>

Semantic types: <https://github.com/Azure/opendigitaltwins-dtdl/blob/master/DTDL/v2/dtdlv2.md#semantic-types>

5. Creating the digital twin

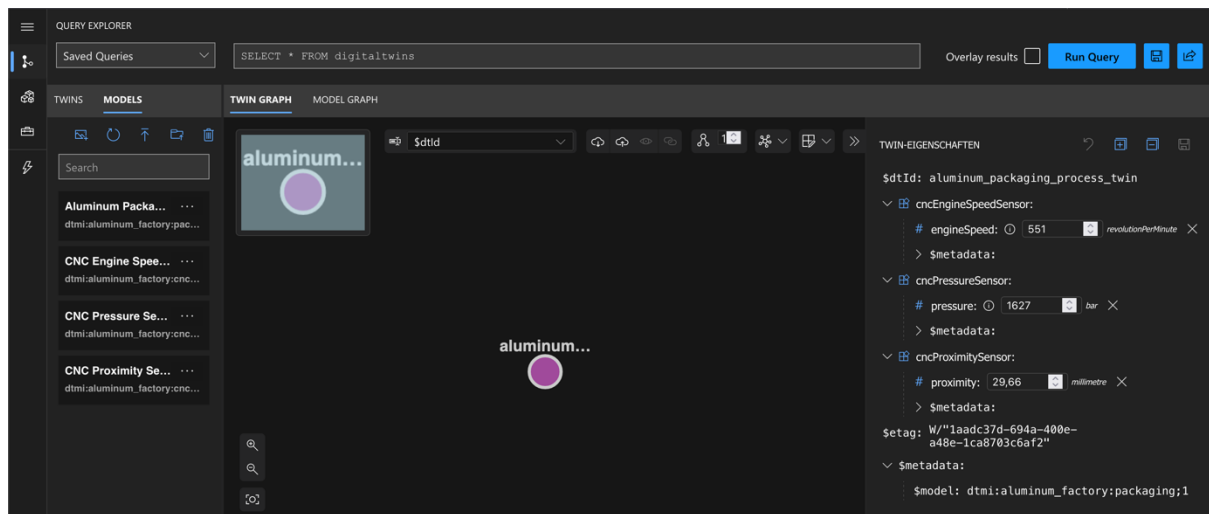
Again, navigate to the “Azure Digital Twins” service in the Azure portal via the “Azure services” overview or the search field. Click on the resource instance you created in step 3 and click “Open Azure Digital Twins Explorer (preview)”. The ADT explorer is a web-based tool that allows you to create, visualize and query the twin (on MacOS, it currently does not work in Safari, only in Chrome and Firefox). In the ADT explorer, navigate to the tab “Models”, click “Upload a Model”, and choose the DTDL (JSON) file on your machine. After the interfaces modeled in DTDL appear in the “Models” tab, click on the three dots (“More options”) of the main interface and click “Create a Twin”.



Azure Digital Twins explorer: <https://explorer.digitaltwins.azure.net/>

6. Querying the digital twin

To query the twin in the ADT explorer, click “Run Query” to run the pre-filled standard query “SELECT * FROM digitaltwins”.



Alternatively, you can use the official Python library to create, modify, update, query and delete twins. The `update_component()` method allows you to update a component by setting the value of a property of for the first time if none was set before (“op”: “add”) or by replacing an existing value (“op”: “replace”). To query the current state of a component, use the `get_component()` method.

```

twin_id = "aluminum_packaging_process_twin"

# Patches (= replacing or adding values for properties of components)

# Function to set engine speed
def set_engine_speed(value):
    component_1 = "cncEngineSpeedSensor"
    try:
        patch_1 = [
            {
                "op": "replace",
                "path": "/engineSpeed",
                "value": value
            }
        ]
        service_client.update_component(twin_id, component_1, patch_1)
    except:
        patch_1 = [
            {
                "op": "add",
                "path": "/engineSpeed",
                "value": value
            }
        ]
        service_client.update_component(twin_id, component_1, patch_1)

```

You don't have to include any hard-coded credentials in the code if you use the same configuration (except for the twin URL) as in the screenshot below and upload the scripts to the cloud shell in the Azure portal and execute them there using the command line interface: use the command “ls” to display the current directory and make sure the files were uploaded. Use the command “python” with the file name, e.g., “python integration.py”, to execute a script. Another way is executing the scripts in Visual Studio Code on a machine that has the Microsoft / Azure login credentials stored in its keychain. It's not completely clear how this works, but one verified way is using MacOS and

having the Azure login credentials stored in the MacOS / Safari keychain: upon executing the script in Visual Studio Code, you are asked to grant the app access to the keychain, which you confirm.

```
# Digital Twin Updates & Queries
import azure.identity as ident
import azure.digitaltwins.core as dt

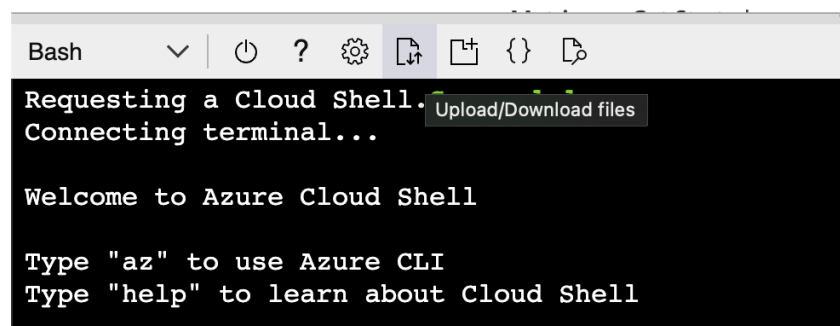
# URL to the Azure Digital Twin
twin_url = "https://RL-Resource-Instance.api.eus.digitaltwins.azure.net"

# DefaultAzureCredential supports different authentication mechanisms and determines the
# appropriate credential type based of the environment it is executing in.
# It attempts to use multiple credential types in an order until it finds a working credential.
# DefaultAzureCredential expects the following three environment variables:
# - AZURE_TENANT_ID: The tenant ID in Azure Active Directory
# - AZURE_CLIENT_ID: The application (client) ID registered in the AAD tenant
# - AZURE_CLIENT_SECRET: The client secret for the registered application

credential = ident.DefaultAzureCredential()
service_client = dt.DigitalTwinsClient(twin_url, credential)
twin_id = "aluminum_packaging_process_twin"
```



Cloud Shell



Azure Digital Twins Core client library for Python: <https://pypi.org/project/azure-digitaltwins-core/>