Microsoft Azure Cloud Services

Authorized & published by Summitworks Technologies Inc

SummitWorks™

GLOBAL SOLUTION ARCHITECTS

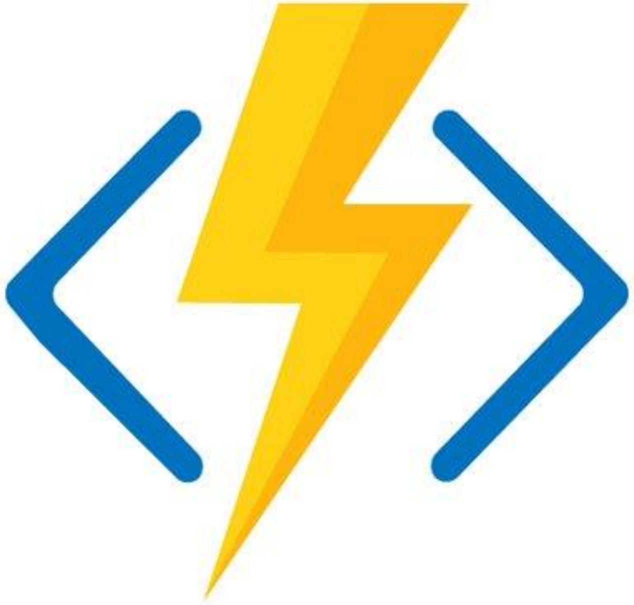
Agenda

- **Azure services**
  - Compute services
    - App Services
      - Function Apps
    - Azure Batch (for large-scale parallel and batch compute jobs)

App Services - Function Apps

Function Apps

Azure Functions is a solution for easily running small pieces of code, or "functions," in the cloud. You can write just the code you need for the problem at hand, without worrying about a whole application or the infrastructure to run it. Functions can make development even more productive, and you can use your development language of choice, such as *C#, F#, Node.js, Java, or PHP.* Pay only for the time your code runs and trust Azure to scale as needed. Azure Functions lets you develop serverless applications on Microsoft Azure.



Function Apps

Features

Choice of language - Write functions using your choice of C#, F#, or Javascript. See Supported languages for other options.

Pay-per-use pricing model - Pay only for the time spent running your code. See the Consumption hosting plan option in the pricing section.

Bring your own dependencies - Functions supports NuGet and NPM, so you can use your favorite libraries.

Integrated security - Protect HTTP-triggered functions with OAuth providers such as Azure Active Directory, Facebook, Google, Twitter, and Microsoft Account.

Function Apps

Features

Simplified integration - Easily leverage Azure services and software-as-a-service (SaaS) offerings. See the integrations section for some examples.

Flexible development - Code your functions right in the portal or set up continuous integration and deploy your code through GitHub, Azure DevOps Services, and other supported development tools.

Open-source - The Functions runtime is open-source and available on GitHub.

Function Apps

What can I do with Functions?

Functions is a great solution for processing data, integrating systems, working with the internet-of-things (IoT), and building simple APIs and microservices. Consider Functions for tasks like image or order processing, file maintenance, or for any tasks that you want to run on a schedule.

- **HTTPTrigger -** Trigger the execution of your code by using an HTTP request.

- **TimerTrigger -** Execute cleanup or other batch tasks on a predefined schedule.

- **CosmosDBTrigger -** Process Azure Cosmos DB documents when they are added or updated in collections in a NoSQL database.

- **BlobTrigger -** Process Azure Storage blobs when they are added to containers. You might use this function for image resizing.

Function Apps

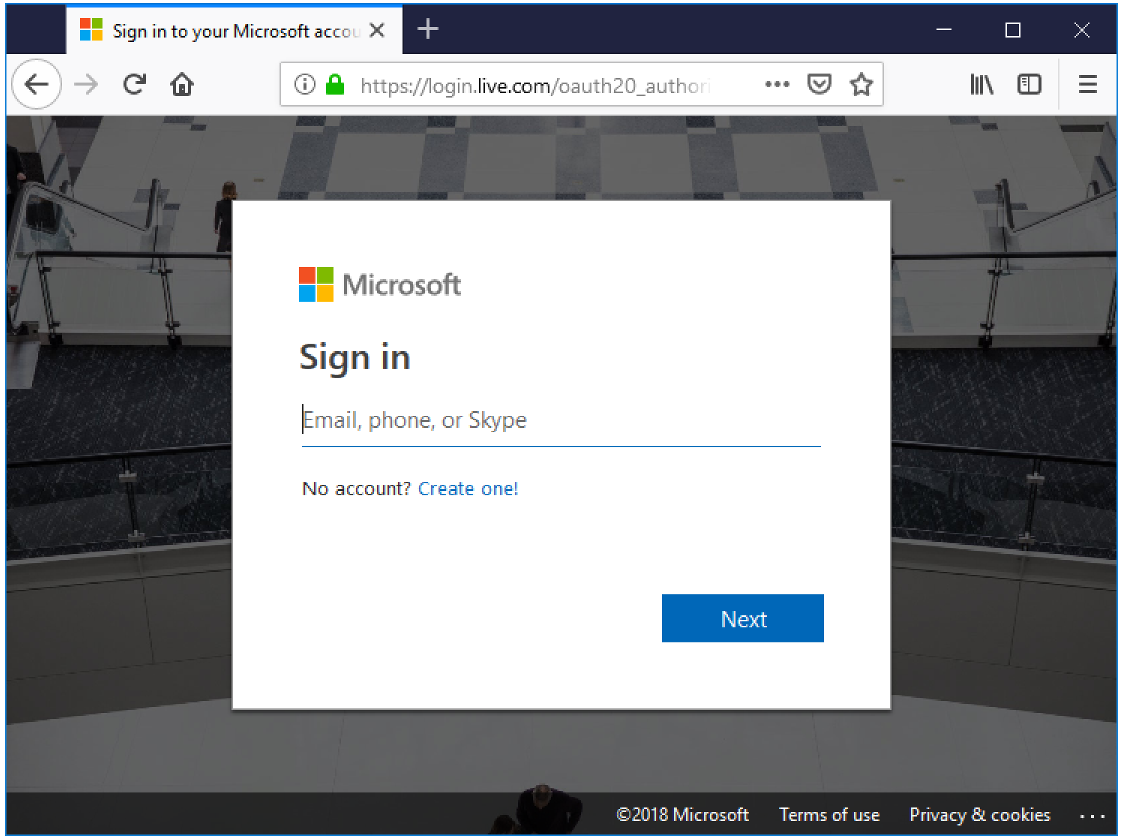
- **QueueTrigger -** Respond to messages as they arrive in an Azure Storage queue.
- **EventGridTrigger -** Respond to events delivered to a subscription in Azure Event Grid. Supports a subscription-based model for receiving events, which includes filtering. A good solution for building event-based architectures.
- **EventHubTrigger -** Respond to events delivered to an Azure Event Hub. Particularly useful in application instrumentation, user experience or workflow processing, and Internet of Things (IoT) scenarios.
- **ServiceBusQueueTrigger -** Connect your code to other Azure services or on-premises services by listening to message queues.
- **ServiceBusTopicTrigger -** Connect your code to other Azure services or on-premises services by subscribing to topics.

Create your first Azure function (portal)

Azure Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application. In this topic, learn how to use Functions to create a "hello world" function in the Azure portal.

Sign in to the Azure portal at

https://portal.azure.com.



Sign in to your Microsoft accou

https://login.live.com/oauth20_authori

Microsoft

Sign in

Email, phone, or Skype

No account? Create one!

Next

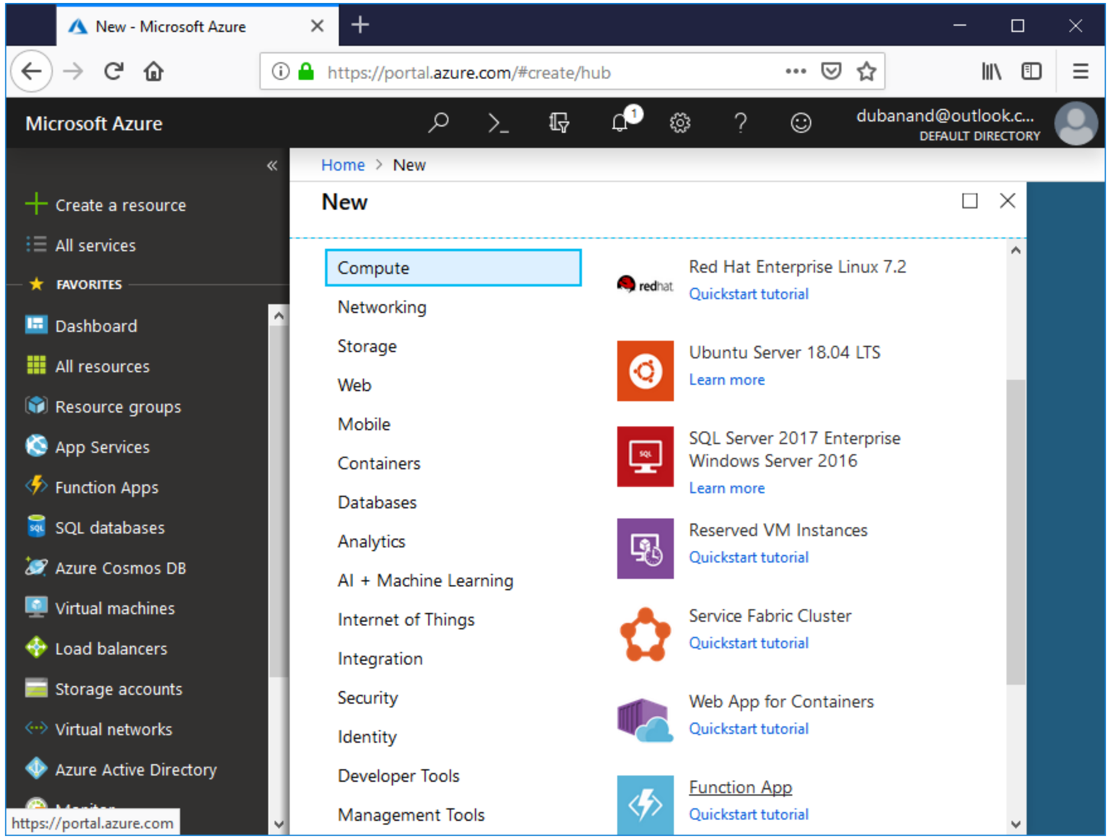
©2018 Microsoft Terms of use Privacy & cookies ...

Create your first Azure function (portal)

Create a function app

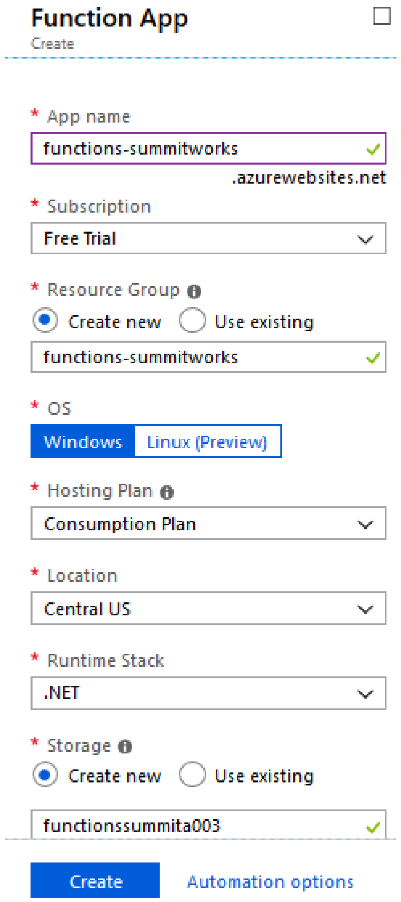
You must have a function app to host the execution of your functions. A function app lets you group functions as a logic unit for easier management, deployment, and sharing of resources.

1. Select the **New** button found on the upper left-hand corner of the Azure portal, then select ***Compute > Function App***.



Create your first Azure function (portal)

2. Use the function app settings as specified in the table below the image.



Function App
Create

* App name
functions-summitworks ✓
.azurewebsites.net

* Subscription
Free Trial

* Resource Group
● Create new ○ Use existing
functions-summitworks ✓

* OS
Windows | Linux (Preview)

* Hosting Plan
Consumption Plan

* Location
Central US

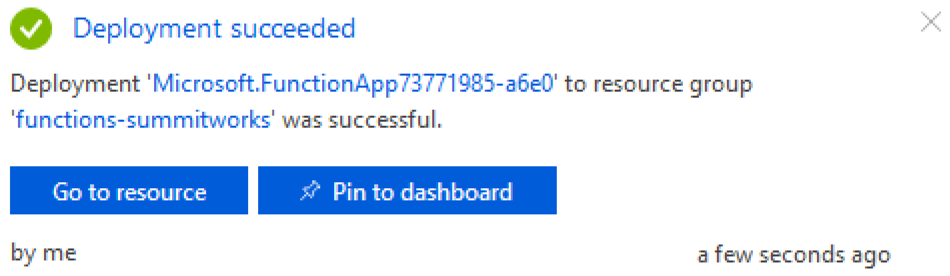
* Runtime Stack
.NET

* Storage
● Create new ○ Use existing
functionssummita003 ✓

3. Select **Create** to provision and deploy the function app

Create your first Azure function (portal)

4. Select the Notification icon in the upper-right corner of the portal and watch for the **Deployment succeeded** message.



Deployment succeeded

Deployment 'Microsoft.FunctionApp73771985-a6e0' to resource group 'functions-summitworks' was successful.

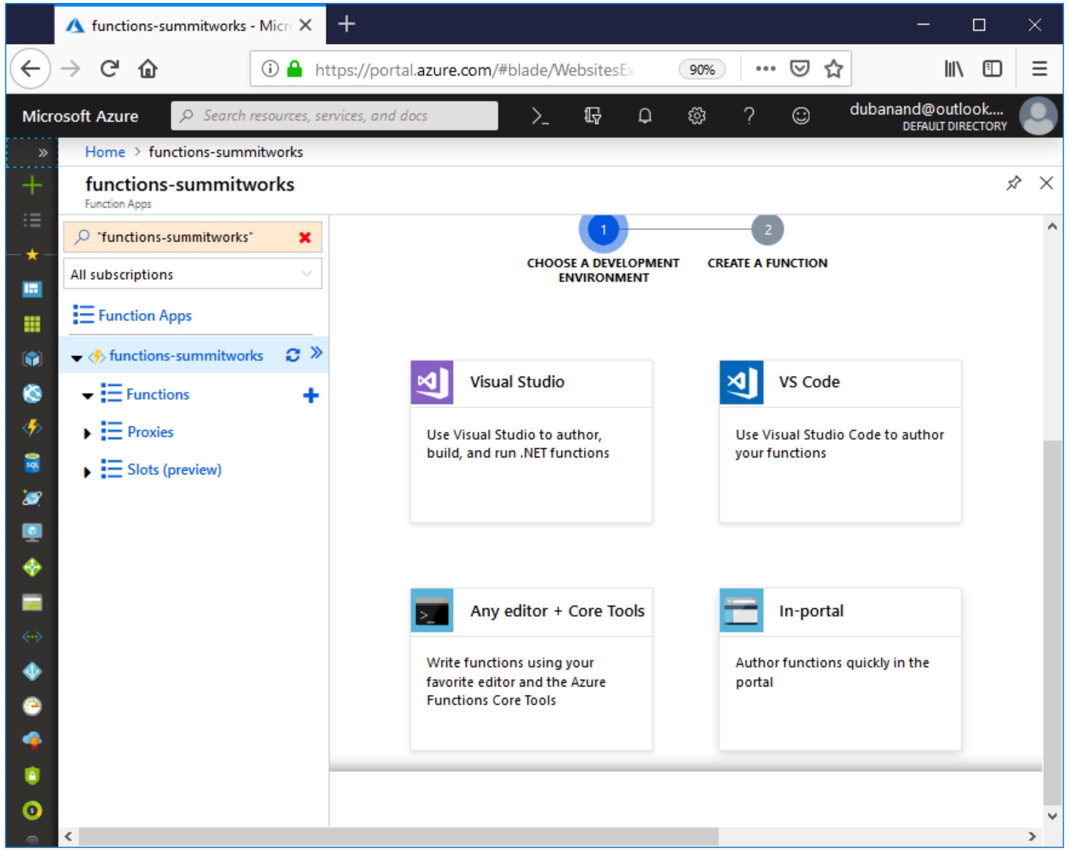
5. Select **Go to resource** to view your new function app.

Next, you create a function in the new function app.

Create your first Azure function (portal)

Create an HTTP triggered function

1. Expand your new function app, then select the **+** button next to **Functions**, choose **In-portal**, and select **Continue**.



Use Visual Studio to author, build, and run .NET functions

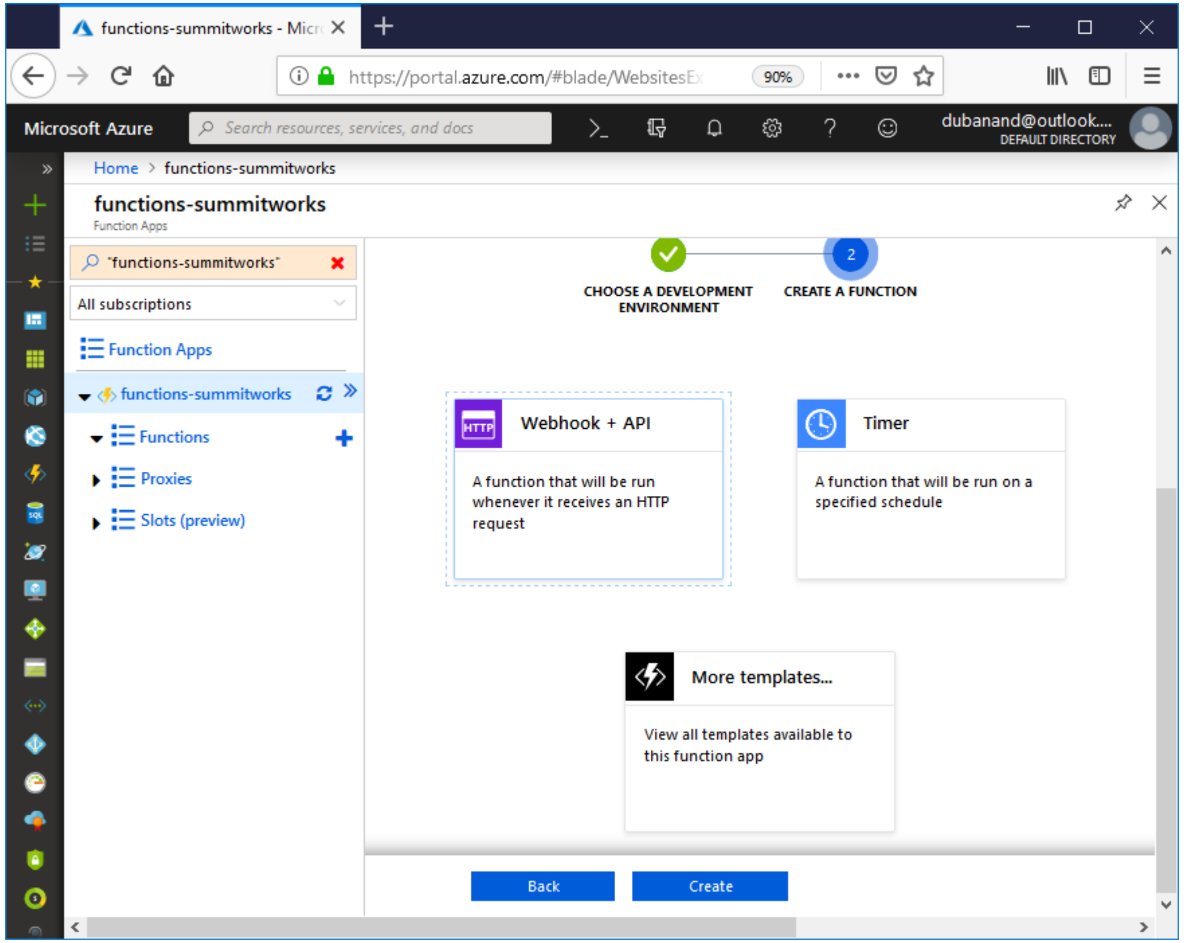
Use Visual Studio Code to author your functions

Write functions using your favorite editor and the Azure Functions Core Tools

Author functions quickly in the portal

Create your first Azure function (portal)

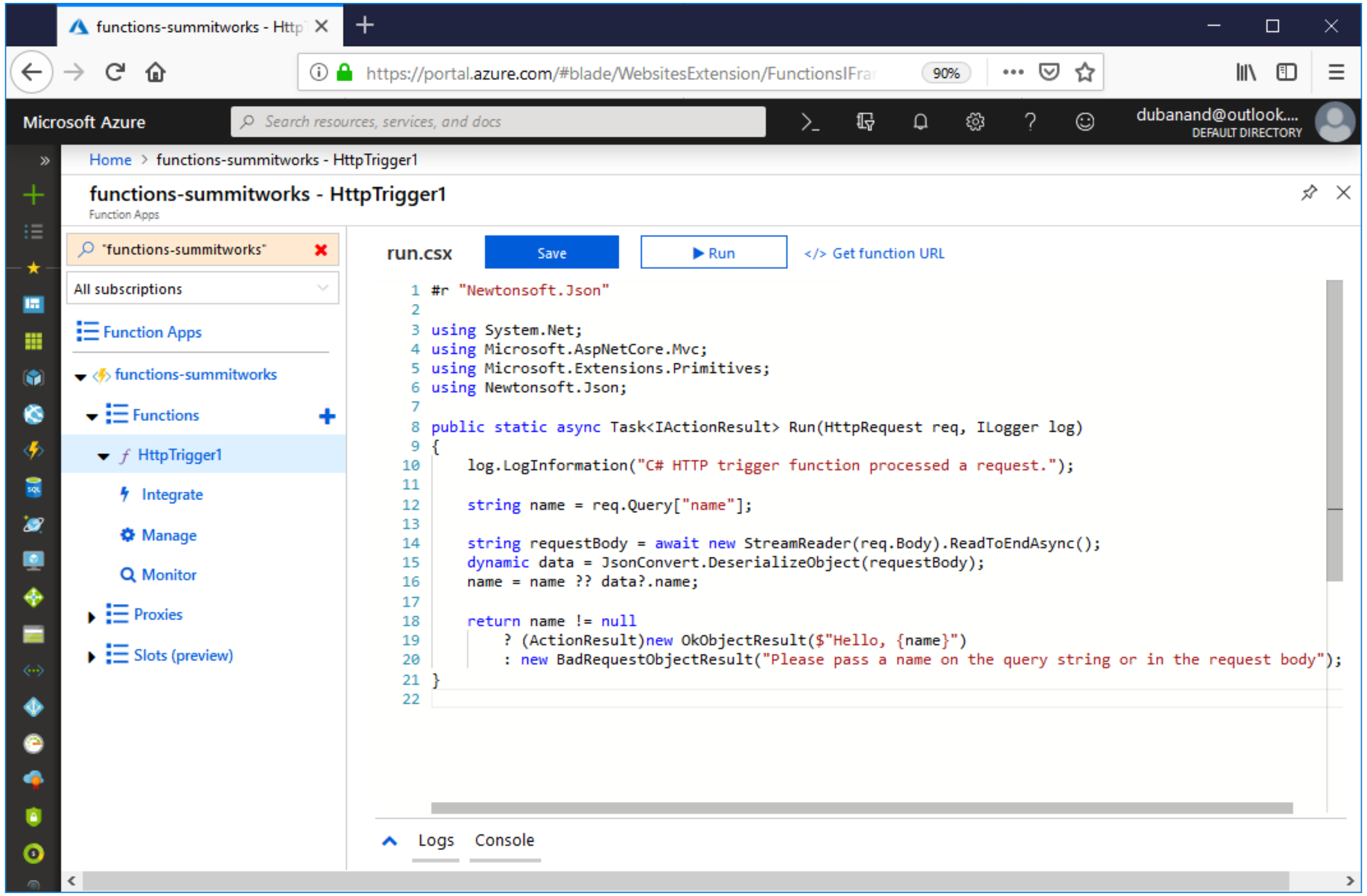
2. Choose **WebHook + API** and then select **Create**.


functions-summitworks - Micr
https://portal.azure.com/#blade/WebsitesEx
90%
Microsoft Azure
Search resources, services, and docs
dubanand@outlook....
DEFAULT DIRECTORY
Home > functions-summitworks
functions-summitworks
Function Apps
"functions-summitworks"
All subscriptions
Function Apps
functions-summitworks
Functions
Proxies
Slots (preview)
CHOOSE A DEVELOPMENT ENVIRONMENT
2
CREATE A FUNCTION
HTTP
Webhook + API
A function that will be run whenever it receives an HTTP request
Timer
A function that will be run on a specified schedule
More templates...
View all templates available to this function app
Back
Create

Create your first Azure function (portal)

A function is created using a language-specific template for an HTTP triggered function.

Now, you can run the new function by sending an HTTP request.

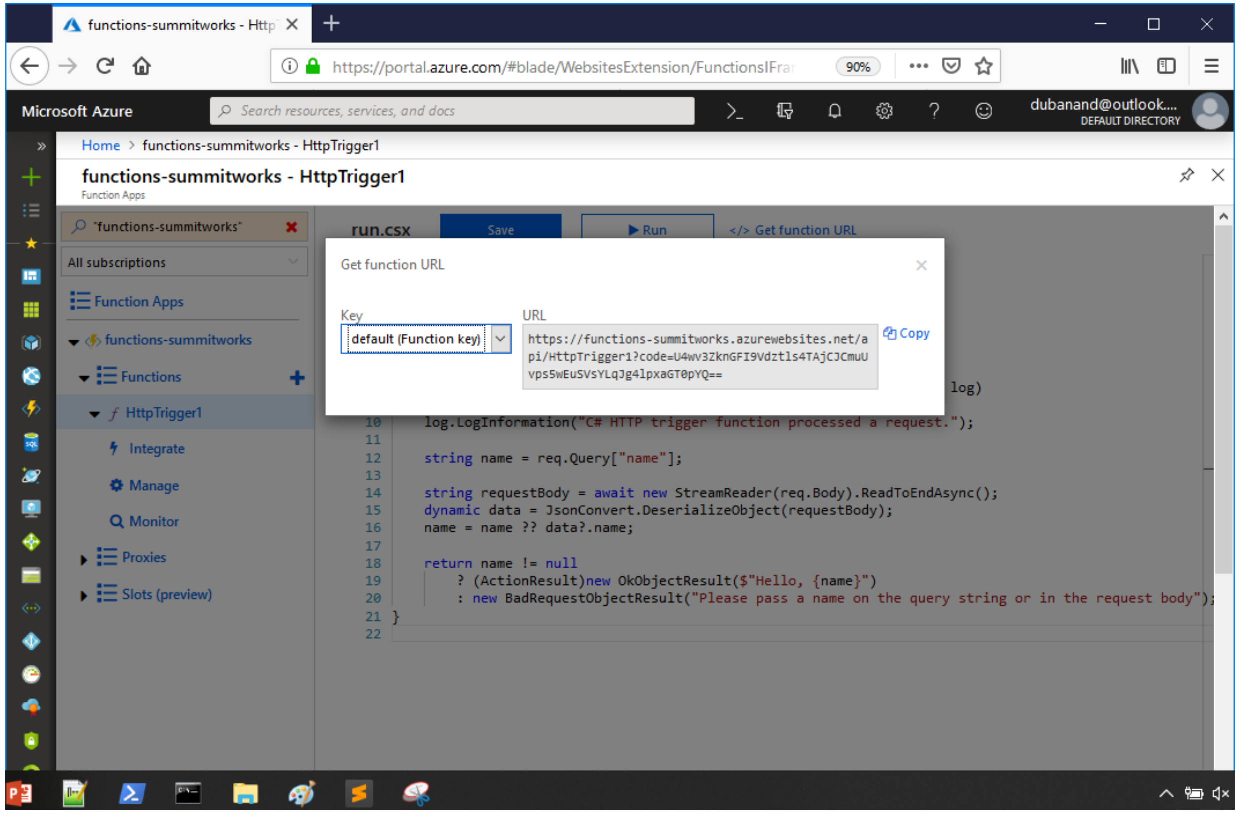


```
#r "Newtonsoft.Json"

using System.Net;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using Newtonsoft.Json;

public static async Task<IActionResult> Run(HttpRequest req, ILogger log)
{
    log.LogInformation("C# HTTP trigger function processed a request.");

    string name = req.Query["name"];

    string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
    dynamic data = JsonConvert.DeserializeObject(requestBody);
    name = name ?? data?.name;

    return name != null
        ? (ActionResult)new OkObjectResult($"Hello, {name}")
        : new BadRequestObjectResult("Please pass a name on the query string or in the request body");
}
```

Create your first Azure function (portal)

Test the function

In your new function, click </> Get function URL at the top right, select default (Function key), and then click Copy.



Get function URL

Key: default (Function key)

URL: https://functions-summitworks.azurewebsites.net/api/HttpTrigger1?code=U4wv3ZknGFI9Vdztls4TAjCJCmuUvps5wEuSVsYLqJg4lpxaGT0pYQ==

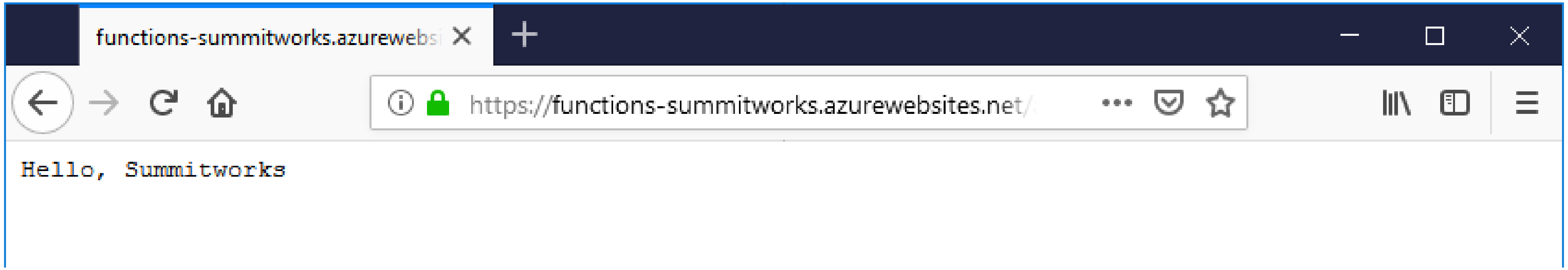
Copy

```
                                                          log)
    log.LogInformation("C# HTTP trigger function processed a request.");

    string name = req.Query["name"];

    string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
    dynamic data = JsonConvert.DeserializeObject(requestBody);
    name = name ?? data?.name;

    return name != null
        ? (ActionResult)new OkObjectResult($"Hello, {name}")
        : new BadRequestObjectResult("Please pass a name on the query string or in the request body");
}
```

Create your first Azure function (portal)

Paste the function URL into your browser's address bar. Add the query string value *&name=<yourname>* to the end of this URL and press the Enter key on your keyboard to execute the request. You should see the response returned by the function displayed in the browser.

The following example shows the response in the browser:

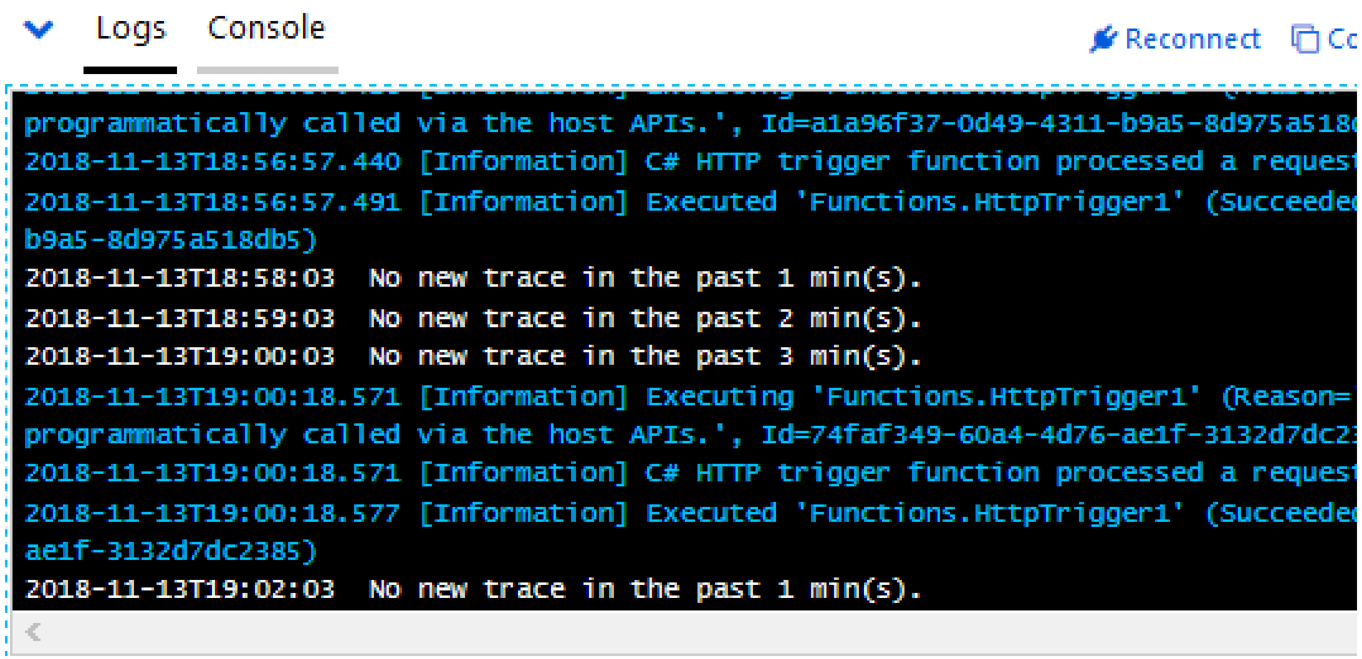

functions-summitworks.azurewebsi
https://functions-summitworks.azurewebsites.net/

```
Hello, Summitworks
```

The request URL includes a key that is required, by default, to access your function over HTTP.

Create your first Azure function (portal)

3. When your function runs, trace information is written to the logs. To see the trace output from the previous execution, return to your function in the portal and click the arrow at the bottom of the screen to expand the **Logs**.

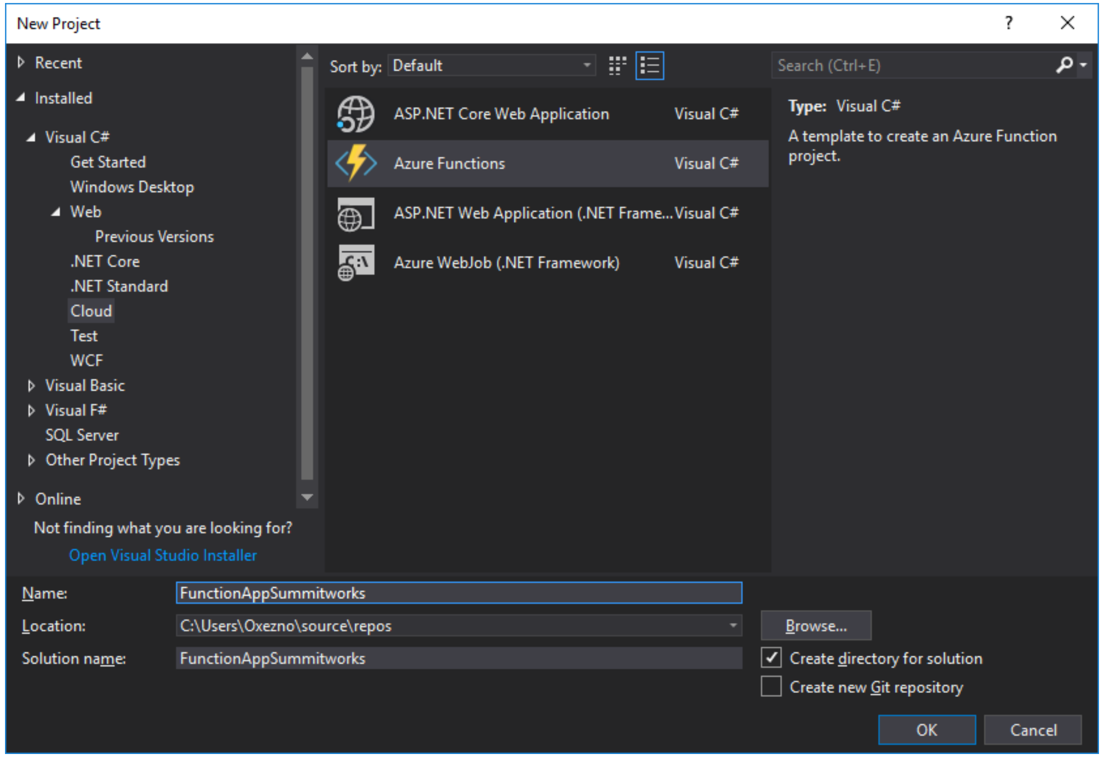


```
programmatically called via the host APIs.', Id=a1a96f37-0d49-4311-b9a5-8d975a518d
2018-11-13T18:56:57.440 [Information] C# HTTP trigger function processed a request
2018-11-13T18:56:57.491 [Information] Executed 'Functions.HttpTrigger1' (Succeeded
b9a5-8d975a518db5)
2018-11-13T18:58:03  No new trace in the past 1 min(s).
2018-11-13T18:59:03  No new trace in the past 2 min(s).
2018-11-13T19:00:03  No new trace in the past 3 min(s).
2018-11-13T19:00:18.571 [Information] Executing 'Functions.HttpTrigger1' (Reason=
programmatically called via the host APIs.', Id=74faf349-60a4-4d76-ae1f-3132d7dc23
2018-11-13T19:00:18.571 [Information] C# HTTP trigger function processed a request
2018-11-13T19:00:18.577 [Information] Executed 'Functions.HttpTrigger1' (Succeeded
ae1f-3132d7dc2385)
2018-11-13T19:02:03  No new trace in the past 1 min(s).
```

Create your first Azure function (VS)

1. In Visual Studio, select **New** > **Project** from the **File** menu.

2. In the **New Project** dialog, select **Installed**, expand ***Visual C# > Cloud***, select **Azure *Functions***, type a **Name** for your project, and click **OK**. The function app name must be valid as a C# namespace, so don't use underscores, hyphens, or any other nonalphanumeric characters.



New Project
?
Recent
Installed
Visual C#
Get Started
Windows Desktop
Web
Previous Versions
.NET Core
.NET Standard
Cloud
Test
WCF
Visual Basic
Visual F#
SQL Server
Other Project Types
Online
Not finding what you are looking for?
Open Visual Studio Installer
Search (Ctrl+E)
ASP.NET Core Web Application Visual C#
Azure Functions Visual C#
ASP.NET Web Application (.NET Frame... Visual C#
Azure WebJob (.NET Framework) Visual C#
Browse...
OK
Cancel

Sort by: Default

Type: Visual C#

A template to create an Azure Function project.

Name: FunctionAppSummitworks

Location: C:\Users\Oxezno\source\repos

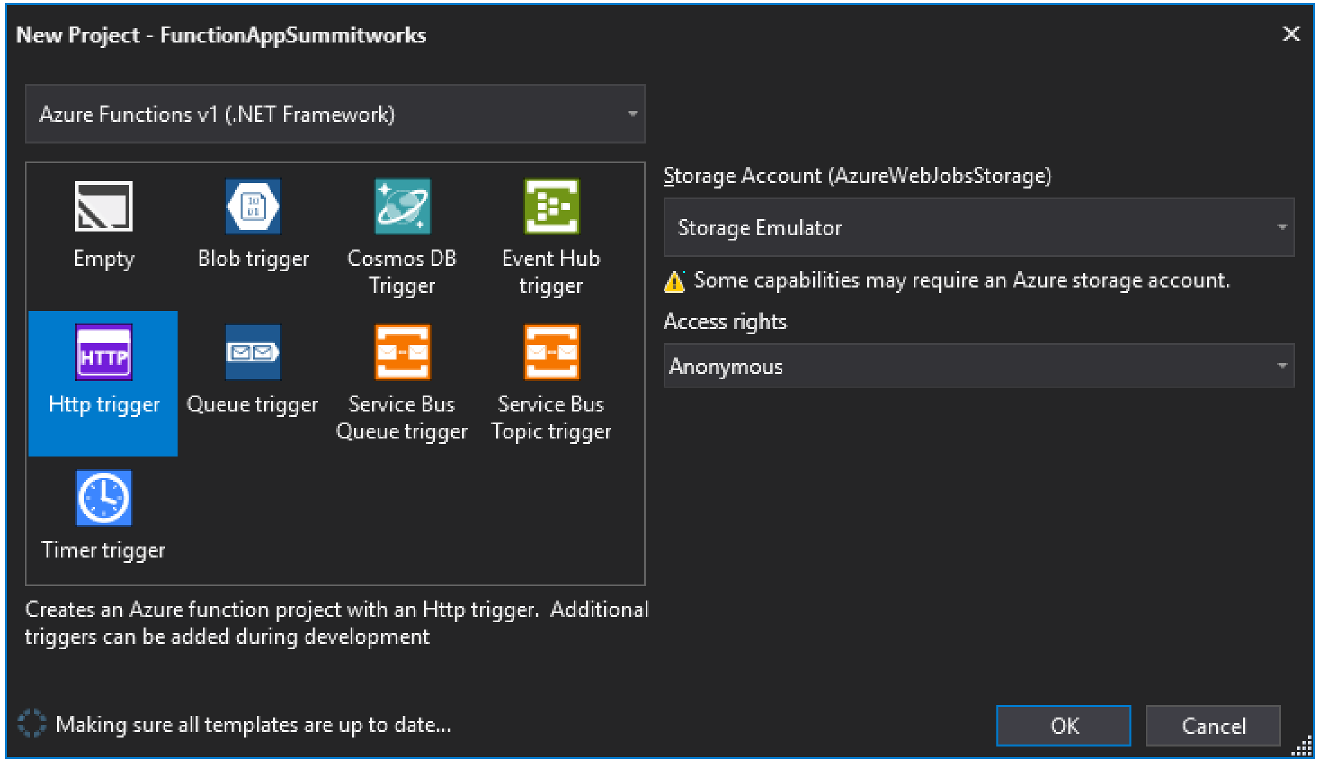
Solution name: FunctionAppSummitworks

☑ Create directory for solution

□ Create new Git repository

Create your first Azure function (VS)

3. Use the settings specified in the table that follows the image.



New Project - FunctionAppSummitworks

Azure Functions v1 (.NET Framework)

Creates an Azure function project with an Http trigger. Additional triggers can be added during development

Storage Account (AzureWebJobsStorage): Storage Emulator

⚠ Some capabilities may require an Azure storage account.

Access rights: Anonymous

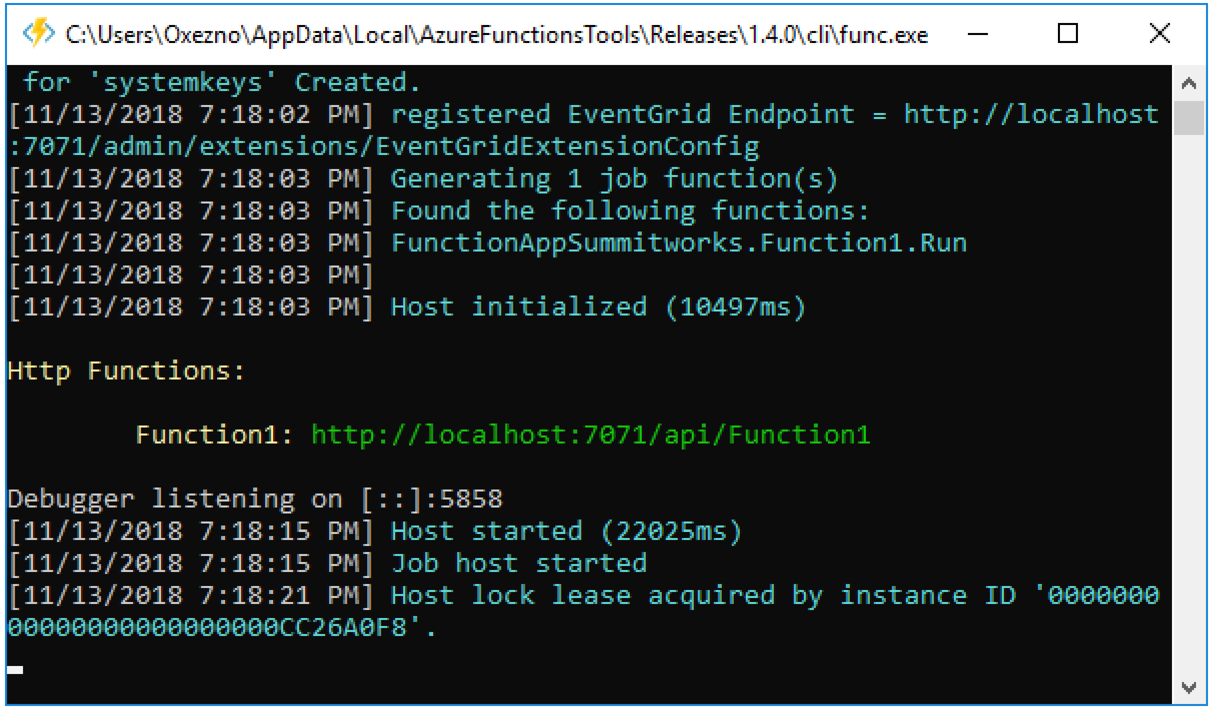
Making sure all templates are up to date...

Anonymous - The created function can be triggered by any client without providing a key. This authorization setting makes it easy to test your new function.

Create your first Azure function (VS)

Test the function locally

1. To test your function, press F5. If prompted, accept the request from Visual Studio to download and install Azure Functions Core (CLI) tools. You may also need to enable a firewall exception so that the tools can handle HTTP requests.
2. Copy the URL of your function from the Azure Functions runtime output.

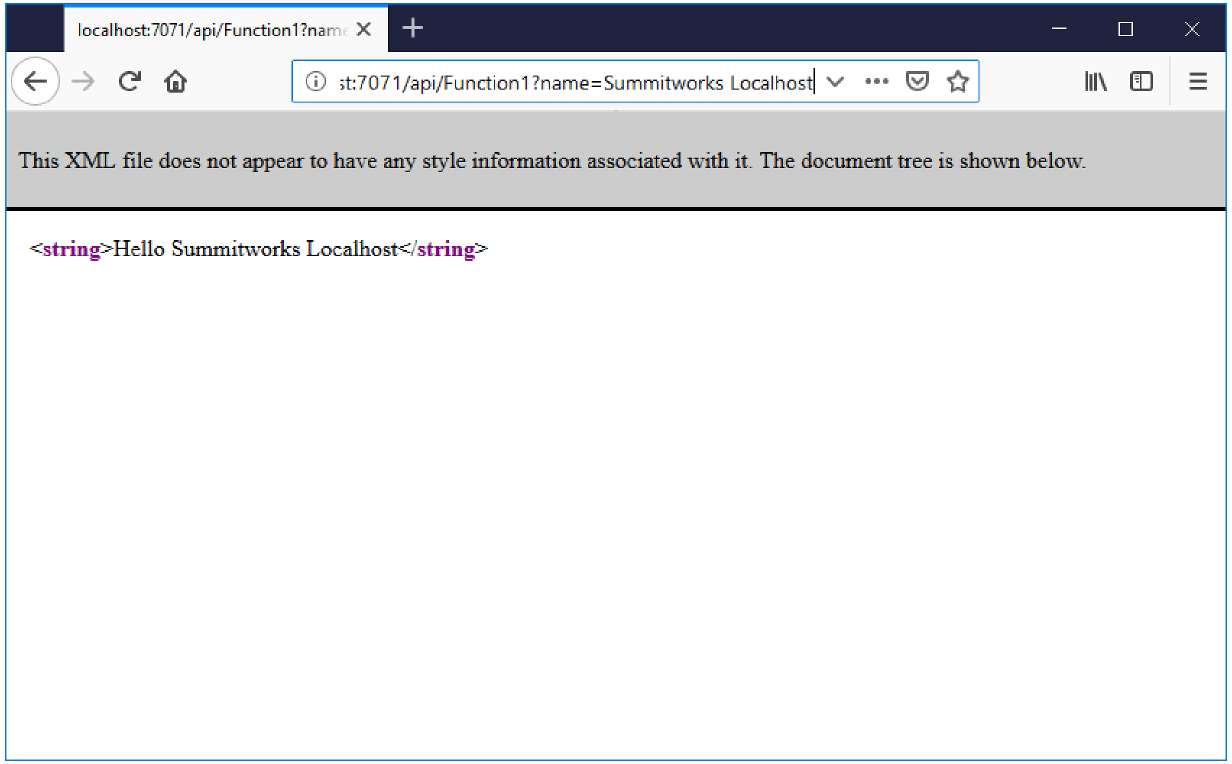


C:\Users\Oxezno\AppData\Local\AzureFunctionsTools\Releases\1.4.0\cli\func.exe

```
 for 'systemkeys' Created.
[11/13/2018 7:18:02 PM] registered EventGrid Endpoint = http://localhost
:7071/admin/extensions/EventGridExtensionConfig
[11/13/2018 7:18:03 PM] Generating 1 job function(s)
[11/13/2018 7:18:03 PM] Found the following functions:
[11/13/2018 7:18:03 PM] FunctionAppSummitworks.Function1.Run
[11/13/2018 7:18:03 PM]
[11/13/2018 7:18:03 PM] Host initialized (10497ms)

Http Functions:

        Function1: http://localhost:7071/api/Function1

Debugger listening on [::]:5858
[11/13/2018 7:18:15 PM] Host started (22025ms)
[11/13/2018 7:18:15 PM] Job host started
[11/13/2018 7:18:21 PM] Host lock lease acquired by instance ID '0000000
0000000000000000CC26A0F8'.
```

Create your first Azure function (VS)

3. Paste the URL for the HTTP request into your browser's address bar. Append the query string *?name=<YOUR_NAME>* to this URL and execute the request. The following shows the response in the browser to the local GET request returned by the function:


localhost:7071/api/Function1?name
st:7071/api/Function1?name=Summitworks Localhost

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<string>Hello Summitworks Localhost</string>
```

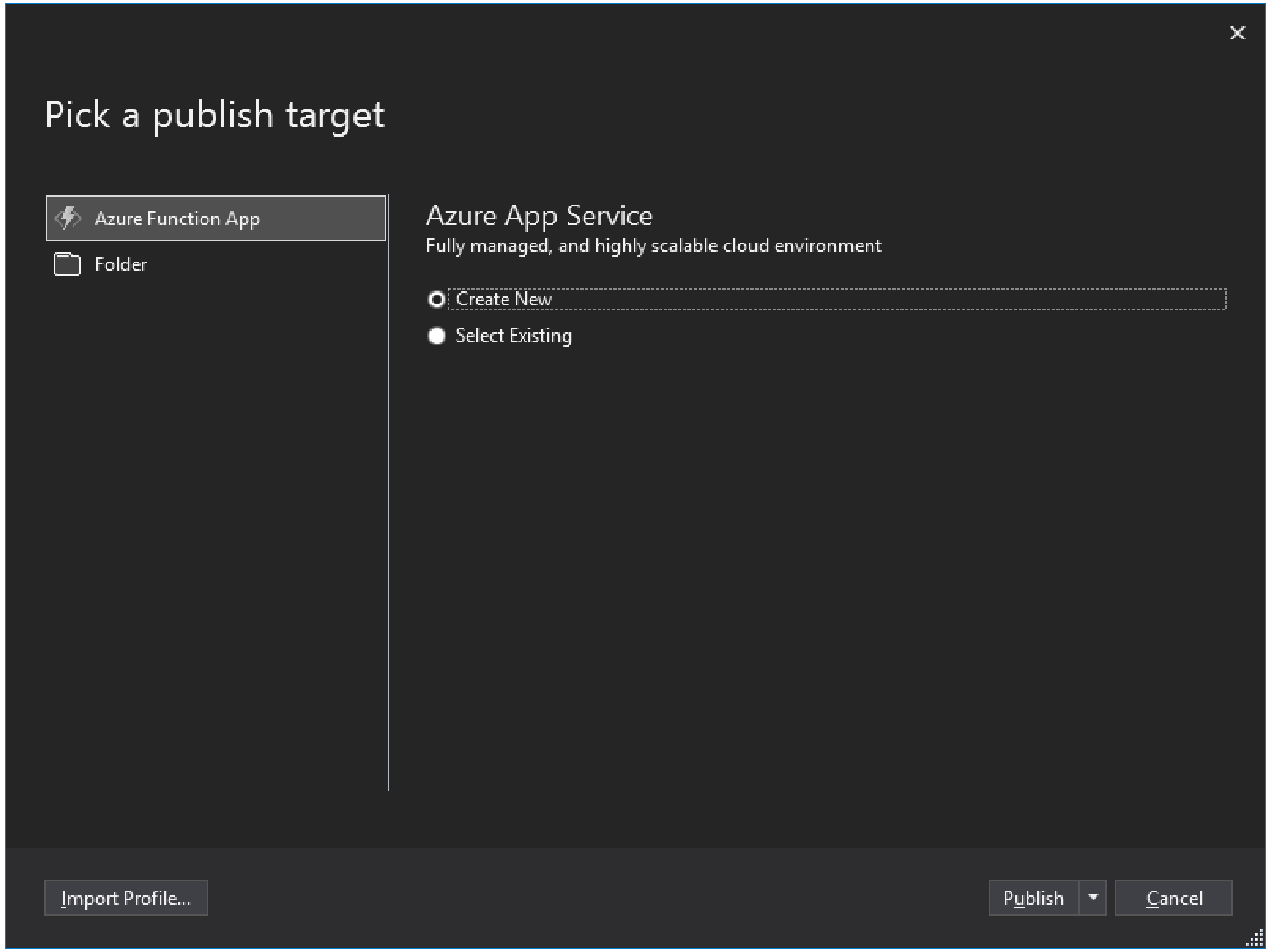
4. To stop debugging, press **Shift + F5**.

Create your first Azure function (VS)

Publish the project to Azure

1. In **Solution Explorer**, right-click the project and select **Publish**.

2. Select **Azure Function App**, choose **Create New**, and then select **Publish**.



Pick a publish target

Azure App Service

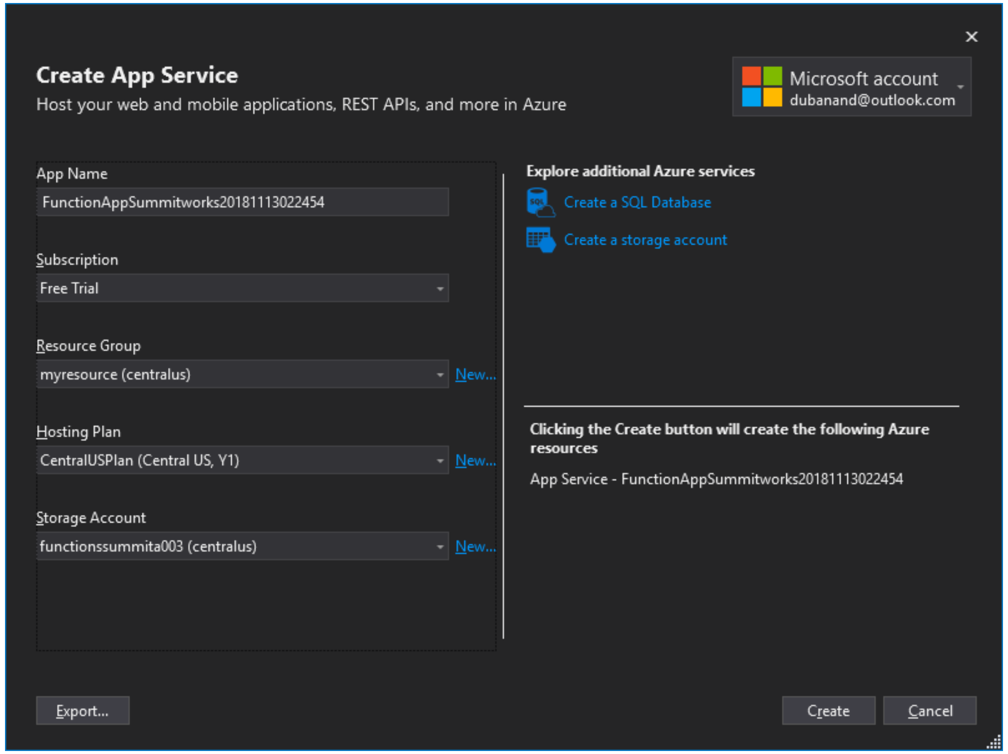
Fully managed, and highly scalable cloud environment

● Create New

○ Select Existing

Create your first Azure function (VS)

3. In the **Create App Service** dialog, use the **Hosting** settings as specified in the table below the image:



Create App Service

Host your web and mobile applications, REST APIs, and more in Azure

Microsoft account
dubanand@outlook.com

App Name: FunctionAppSummitworks20181113022454

Subscription: Free Trial

Resource Group: myresource (centralus)

Hosting Plan: CentralUSPlan (Central US, Y1)

Storage Account: functionssummita003 (centralus)

Explore additional Azure services

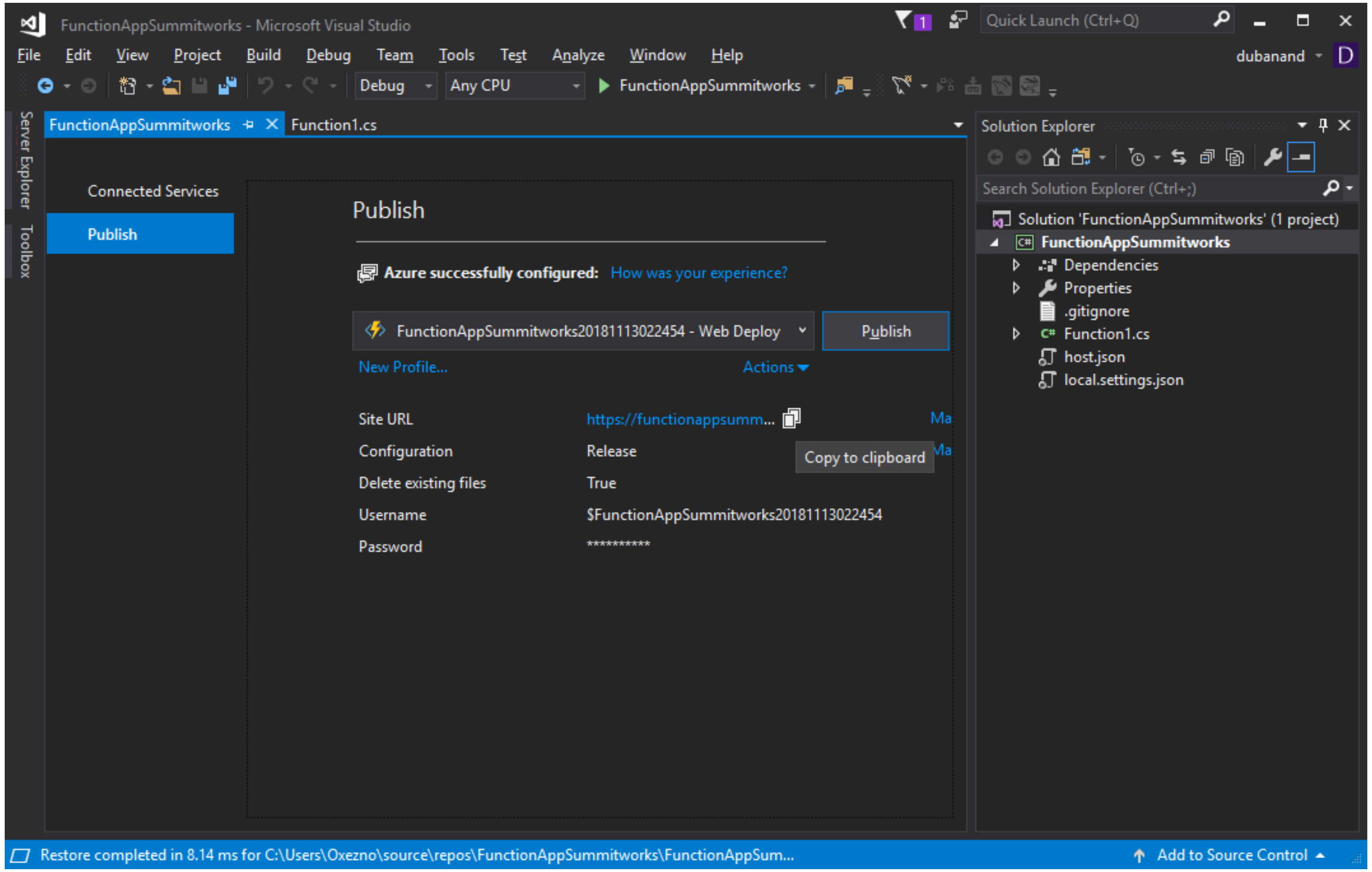
Clicking the Create button will create the following Azure resources

App Service - FunctionAppSummitworks20181113022454

4. Click **Create** to create a function app and related resources in Azure with these settings and deploy your function project code.

Create your first Azure function (VS)

5. After the deployment is complete, make a note of the **Site URL** value, which is the address of your function app in Azure.



Azure successfully configured: How was your experience?

FunctionAppSummitworks20181113022454 - Web Deploy

Site URL https://functionappsumm...

Configuration Release

Delete existing files True

Username $FunctionAppSummitworks20181113022454

Password **********

Restore completed in 8.14 ms for C:\Users\Oxezno\source\repos\FunctionAppSummitworks\FunctionAppSum...

Create your first Azure function (VS)

Test your function in Azure

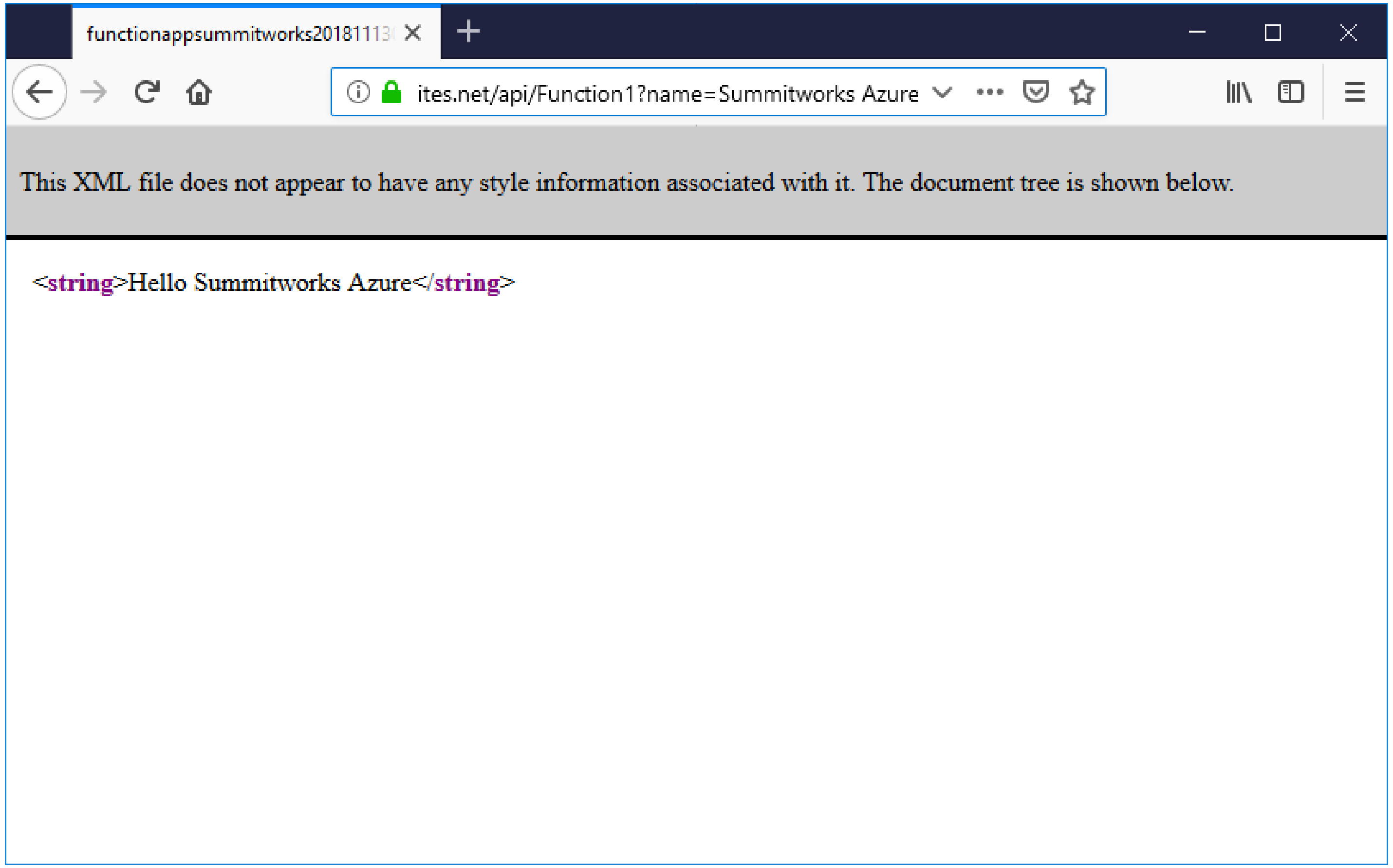
1. Copy the base URL of the function app from the Publish profile page. Replace the localhost:port portion of the URL you used when testing the function locally with the new base URL. As before, make sure to append the query string ?name=<YOUR_NAME> to this URL and execute the request.

The URL that calls your HTTP triggered function should be in the following format:

http://<APP_NAME>.azurewebsites.net/api/<FUNCTION_NAME>?name=<YOUR_NAME>

Create your first Azure function (VS)

2. Paste this new URL for the HTTP request into your browser's address bar. The following shows the response in the browser to the remote GET request returned by the function:



ites.net/api/Function1?name=Summitworks Azure

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<string>Hello Summitworks Azure</string>
```

Create your first Azure function (CLI)

Connect to Your Azure Account

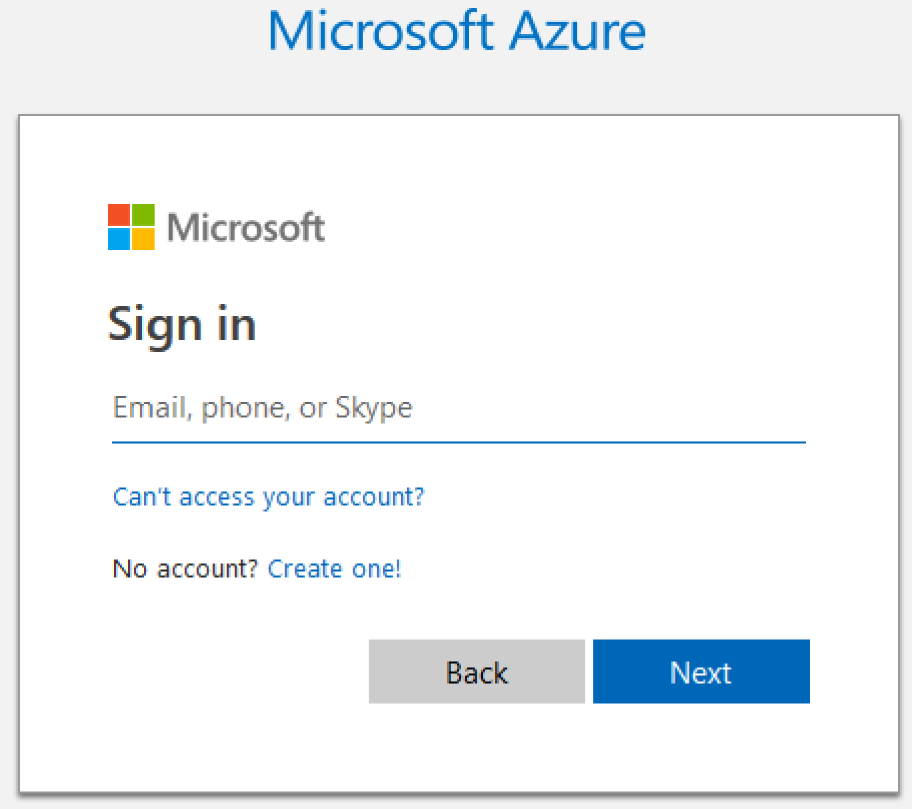
You can now run the Azure CLI with the az command from either Windows Command Prompt or PowerShell. PowerShell offers some tab completion features not available from Windows Command Prompt. To sign in, run the command:

```
az login
```

If the CLI can open your default browser, it will do so and load a sign-in page.

Otherwise, you need to open a browser page and follow the instructions on the command line to enter an authorization code after navigating to https://aka.ms/devicelogin in your browser.

Sign in with your account credentials in the browser.



Microsoft Azure

Microsoft

Sign in

Email, phone, or Skype

Can't access your account?

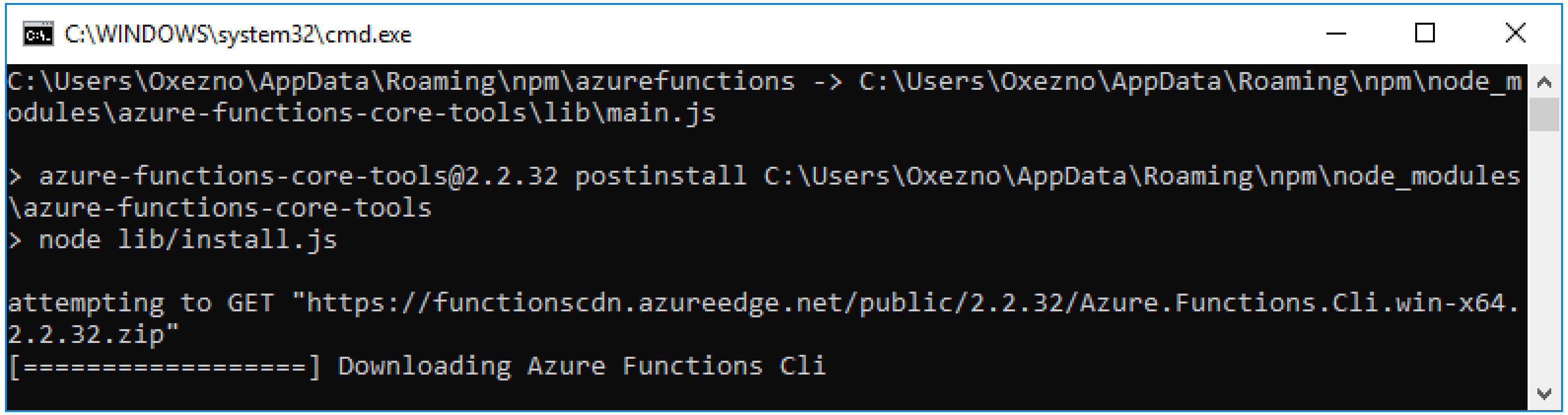
No account? Create one!

Create your first Azure function (CLI)

Install the Azure Functions Core Tools

Azure Functions Core Tools includes a version of the same runtime that powers Azure Functions runtime that you can run on your local development computer. It also provides commands to create functions, connect to Azure, and deploy function projects.

npm install -g azure-functions-core-tools



```
C:\Users\Oxezno\AppData\Roaming\npm\azurefunctions -> C:\Users\Oxezno\AppData\Roaming\npm\node_m
odules\azure-functions-core-tools\lib\main.js

> azure-functions-core-tools@2.2.32 postinstall C:\Users\Oxezno\AppData\Roaming\npm\node_modules
\azure-functions-core-tools
> node lib/install.js

attempting to GET "https://functionscdn.azureedge.net/public/2.2.32/Azure.Functions.Cli.win-x64.
2.2.32.zip"
[==================] Downloading Azure Functions Cli
```

Create your first Azure function (CLI)

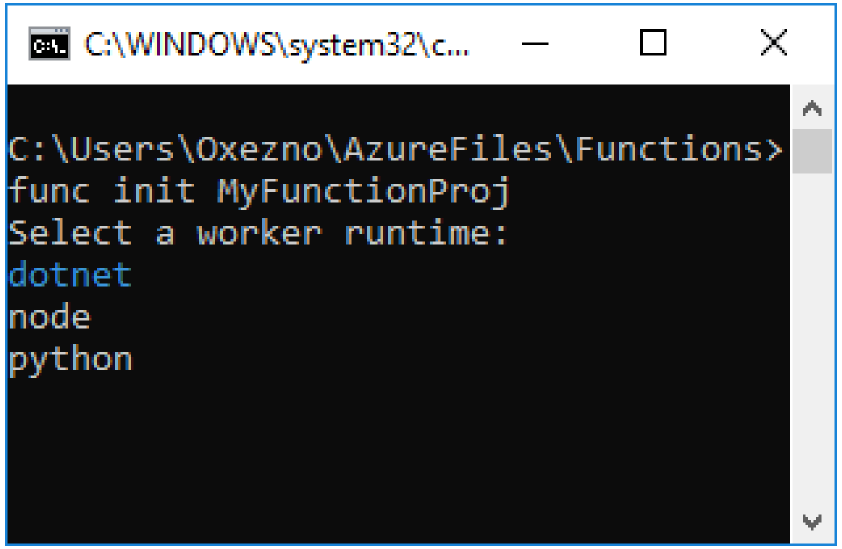
Run the following command from the command line to create a function app project in the MyFunctionProj folder of the current local directory. A GitHub repo is also created in MyFunctionProj.

func init MyFunctionProj

When prompted, select a worker runtime from the following language choices:

dotnet: creates a .NET class library project.

node: creates a JavaScript project.

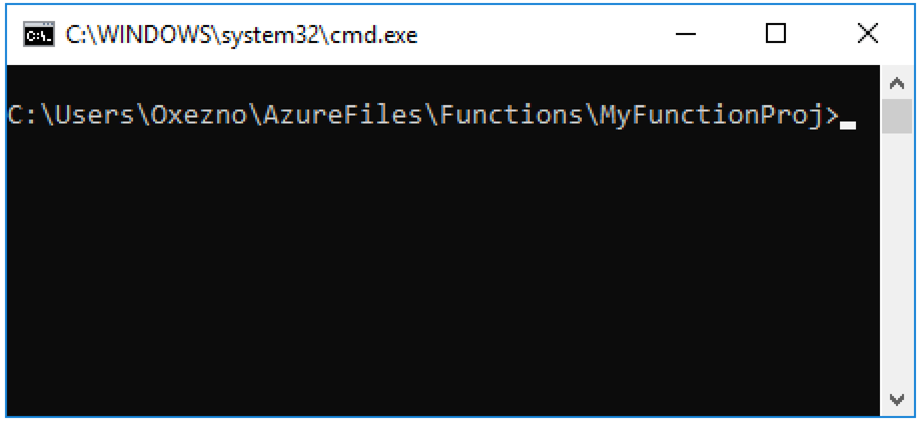


```
C:\Users\Oxezno\AzureFiles\Functions>
func init MyFunctionProj
Select a worker runtime:
dotnet
node
python
```

Create your first Azure function (CLI)

Use the following command to navigate to the new MyFunctionProj project folder.

- *cd MyFunctionProj*



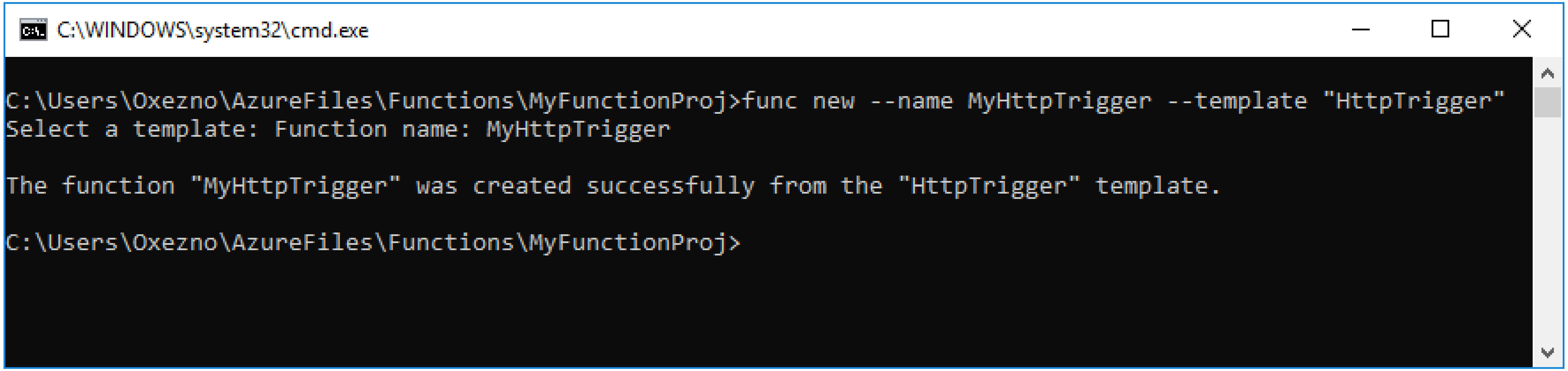
```
C:\Users\Oxezno\AzureFiles\Functions\MyFunctionProj>_
```

Create your first Azure function (CLI)

Create a function

The following command creates an HTTP-triggered function named MyHtpTrigger.

func new --name MyHttpTrigger --template "HttpTrigger"



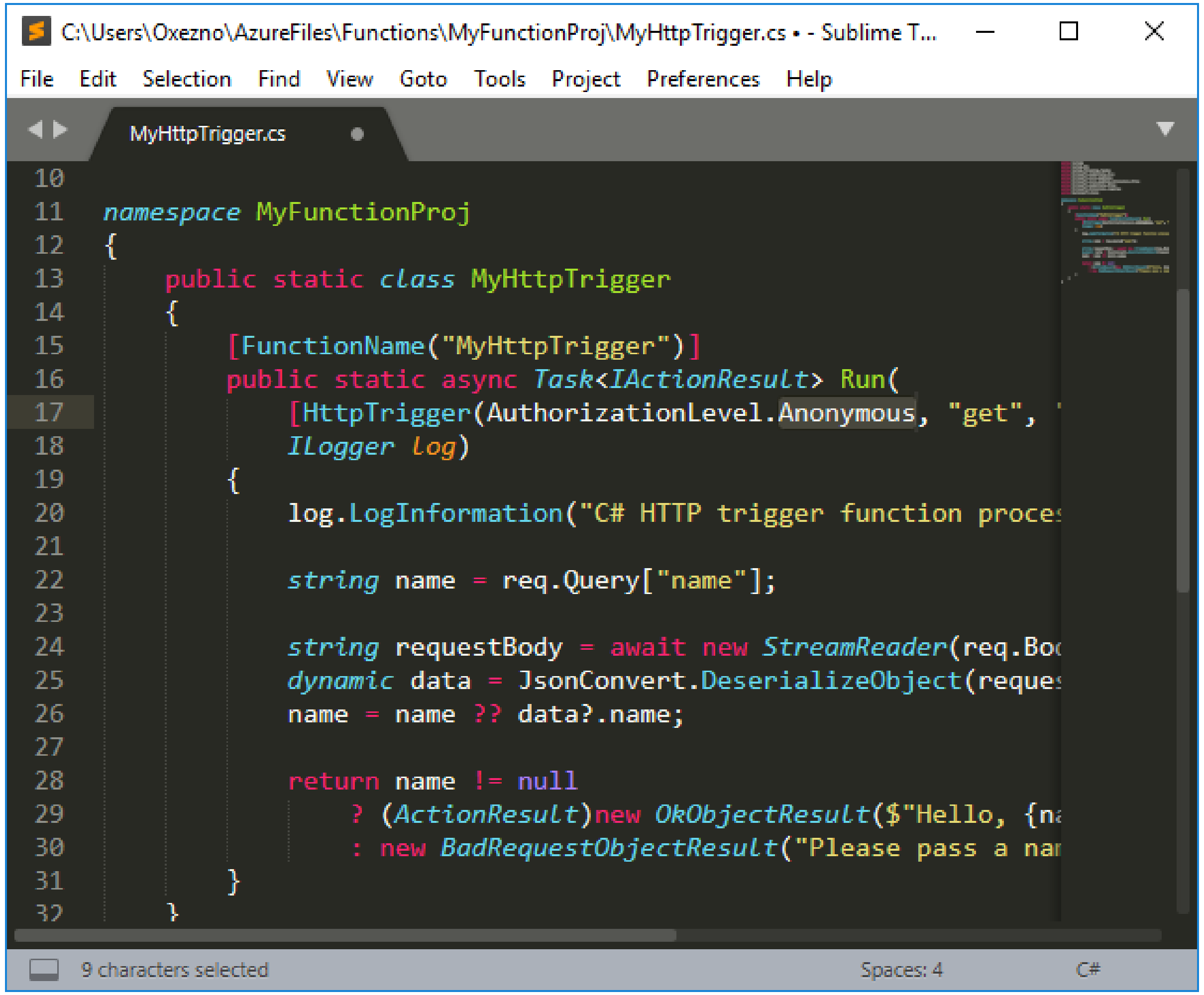
```
C:\Users\Oxezno\AzureFiles\Functions\MyFunctionProj>func new --name MyHttpTrigger --template "HttpTrigger"
Select a template: Function name: MyHttpTrigger

The function "MyHttpTrigger" was created successfully from the "HttpTrigger" template.

C:\Users\Oxezno\AzureFiles\Functions\MyFunctionProj>
```

Create your first Azure function (CLI)

Update the function

By default, the template creates a function that requires a function key when making requests. To make it easier to test the function in Azure, you need to update the function to allow anonymous access. The way that you make this change depends on your functions project language.

Open the MyHttpTrigger.cs code file that is your new function and update the AuthorizationLevel attribute in the function definition to a value of *anonymous* and save your changes.


C:\Users\Oxezno\AzureFiles\Functions\MyFunctionProj\MyHttpTrigger.cs • - Sublime T...
File Edit Selection Find View Goto Tools Project Preferences Help
MyHttpTrigger.cs

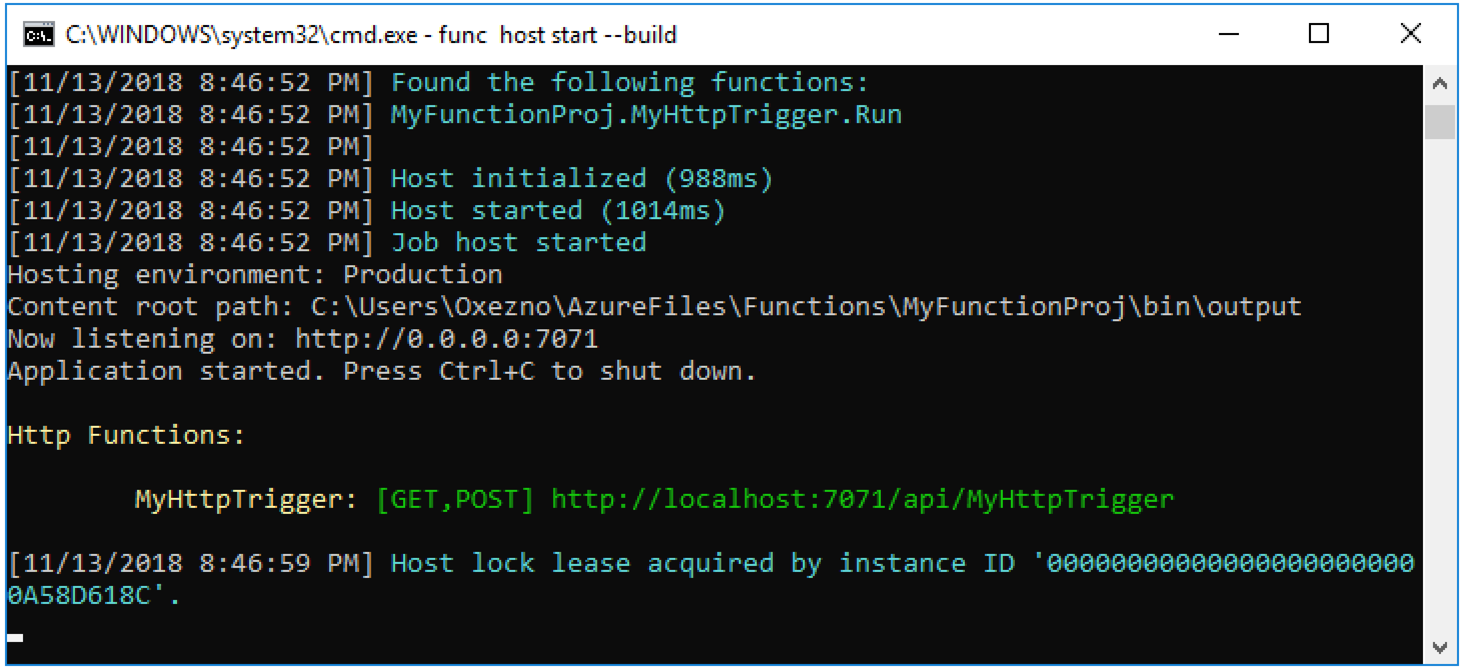
```

namespace MyFunctionProj
{
    public static class MyHttpTrigger
    {
        [FunctionName("MyHttpTrigger")]
        public static async Task<IActionResult> Run(
            [HttpTrigger(AuthorizationLevel.Anonymous, "get", '
            ILogger log)
        {
            log.LogInformation("C# HTTP trigger function proces

            string name = req.Query["name"];

            string requestBody = await new StreamReader(req.Bo
            dynamic data = JsonConvert.DeserializeObject(reque
            name = name ?? data?.name;

            return name != null
                ? (ActionResult)new OkObjectResult($"Hello, {n
                : new BadRequestObjectResult("Please pass a na
        }
    }
```

Create your first Azure function (CLI)

Run the function locally

The following command starts the function app. The app runs using the same Azure Functions runtime that is in Azure.

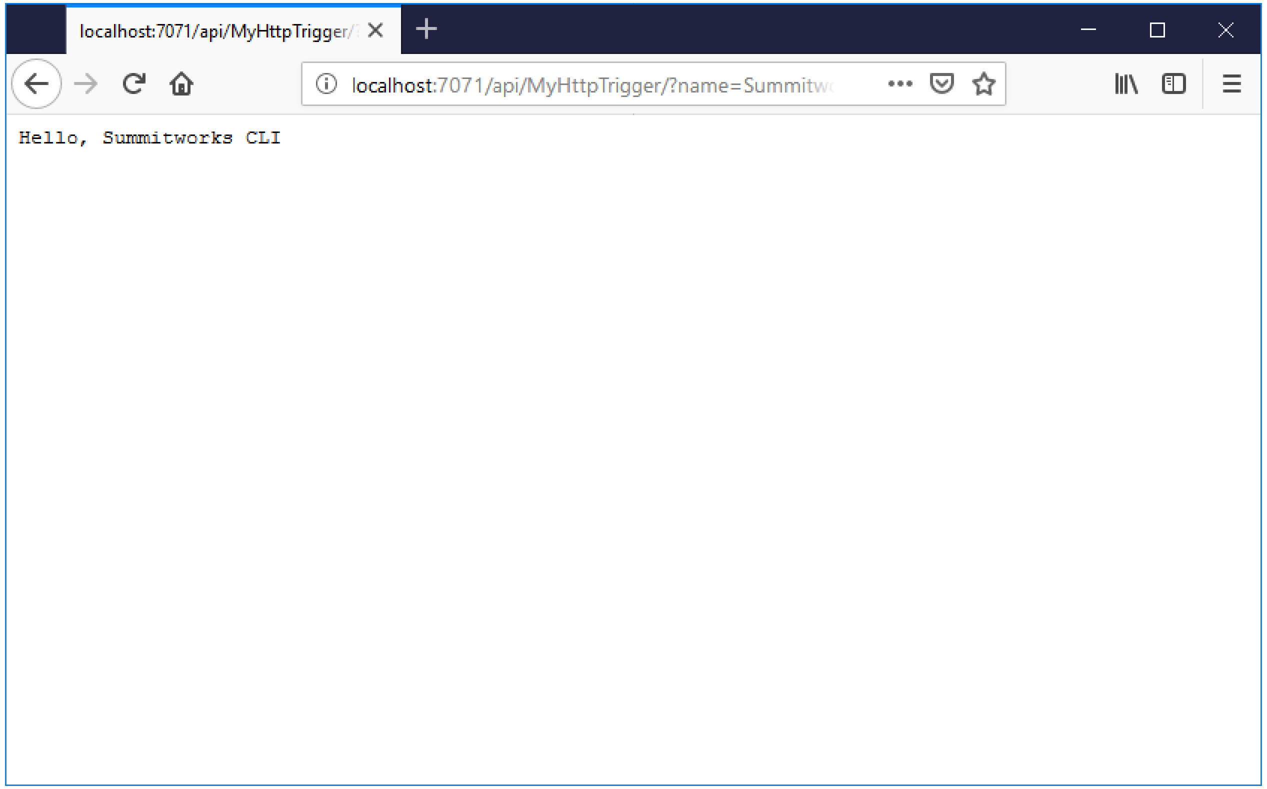
func host start --build



```
[11/13/2018 8:46:52 PM] Found the following functions:
[11/13/2018 8:46:52 PM] MyFunctionProj.MyHttpTrigger.Run
[11/13/2018 8:46:52 PM]
[11/13/2018 8:46:52 PM] Host initialized (988ms)
[11/13/2018 8:46:52 PM] Host started (1014ms)
[11/13/2018 8:46:52 PM] Job host started
Hosting environment: Production
Content root path: C:\Users\Oxezno\AzureFiles\Functions\MyFunctionProj\bin\output
Now listening on: http://0.0.0.0:7071
Application started. Press Ctrl+C to shut down.

Http Functions:

        MyHttpTrigger: [GET,POST] http://localhost:7071/api/MyHttpTrigger

[11/13/2018 8:46:59 PM] Host lock lease acquired by instance ID '00000000000000000000000
0A58D618C'.
```

Create your first Azure function (CLI)

Copy the URL of your HttpTrigger function from the runtime output and paste it into your browser's address bar. Append the query string ?name=<yourname> to this URL and execute the request. The following shows the response in the browser to the GET request returned by the local function:


localhost:7071/api/MyHttpTrigger/
localhost:7071/api/MyHttpTrigger/?name=Summitw

```
Hello, Summitworks CLI
```

Azure Compute Services - Batch

Azure Batch

Use Azure Batch to run large-scale parallel and high-performance computing (HPC) batch jobs efficiently in Azure. Azure Batch creates and manages a pool of compute nodes (virtual machines), installs the applications you want to run, and schedules jobs to run on the nodes. There is no cluster or job scheduler software to install, manage, or scale. Instead, you use Batch APIs and tools, command-line scripts, or the Azure portal to configure, manage, and monitor your jobs.

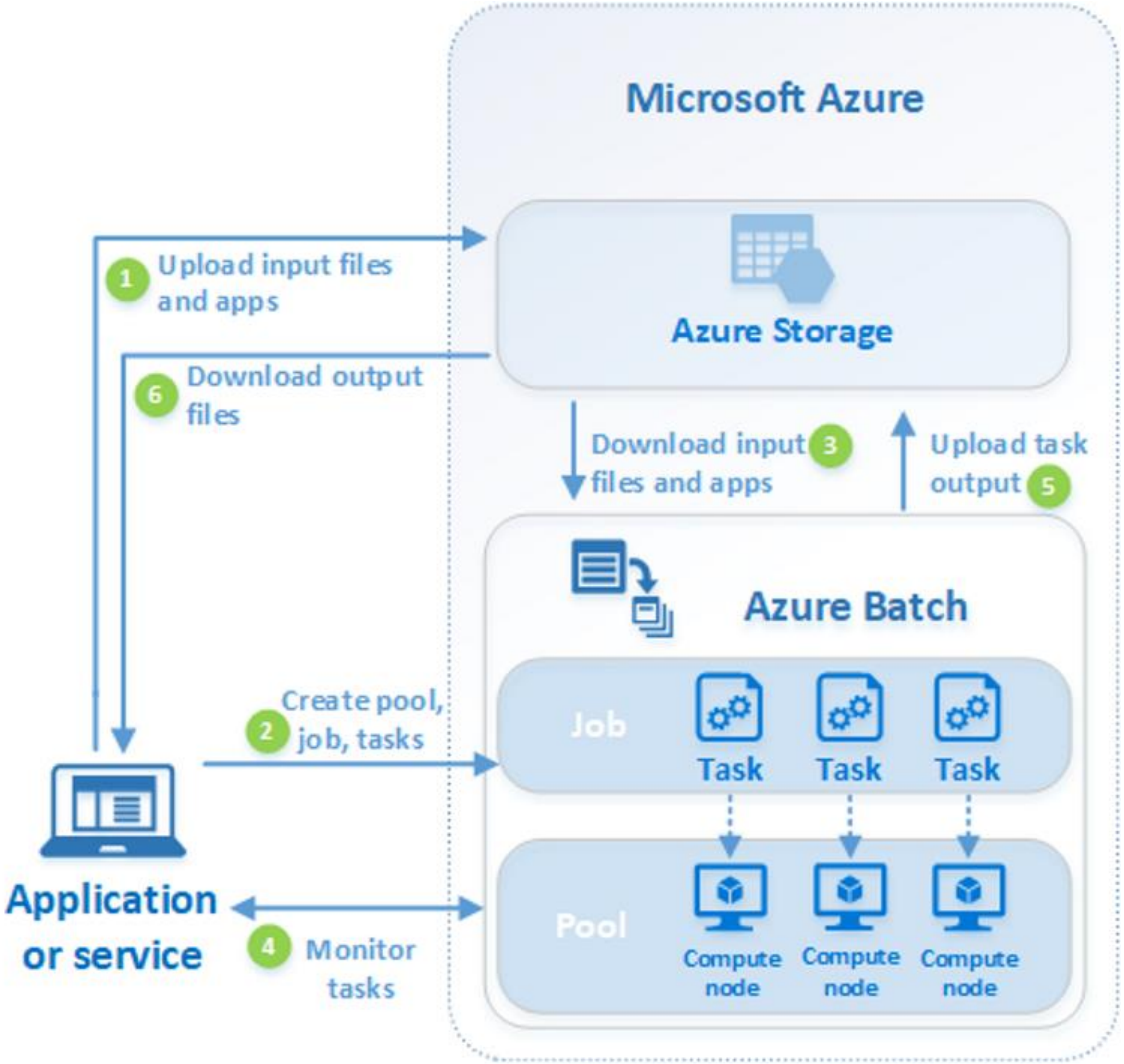
Developers can use Batch as a platform service to build SaaS applications or client apps where large-scale execution is required. For example, build a service with Batch to run a Monte Carlo risk simulation for a financial services company, or a service to process many images.

Azure Batch

How it works

A common scenario for Batch involves scaling out intrinsically parallel work, such as the rendering of images for 3D scenes, on a pool of compute nodes. This pool of compute nodes can be your "render farm" that provides tens, hundreds, or even thousands of cores to your rendering job.

The following diagram shows steps in a common Batch workflow, with a client application or hosted service using Batch to run a parallel workload.

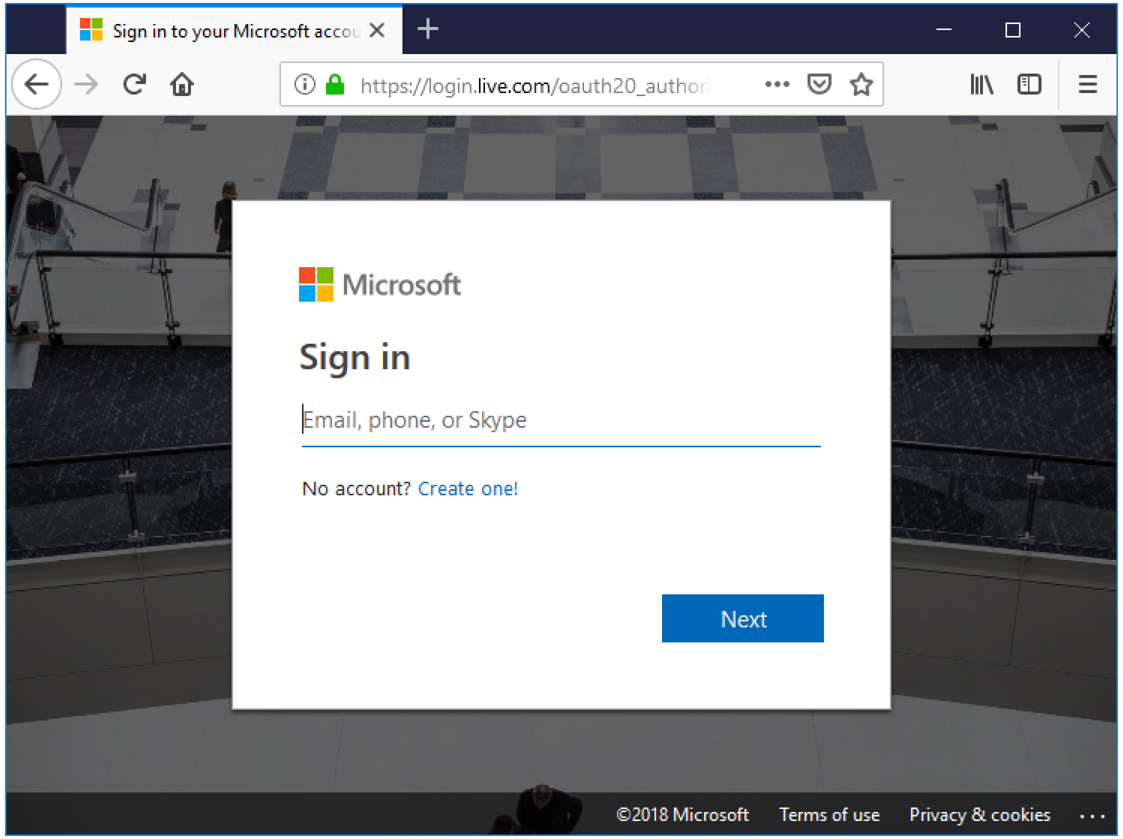


Run your first Batch job (portal)

Now lets understand how to use the Azure portal to create a Batch account, a *pool* of compute nodes (virtual machines), and a *job* that runs basic *tasks* on the pool.

Sign in to the Azure portal at

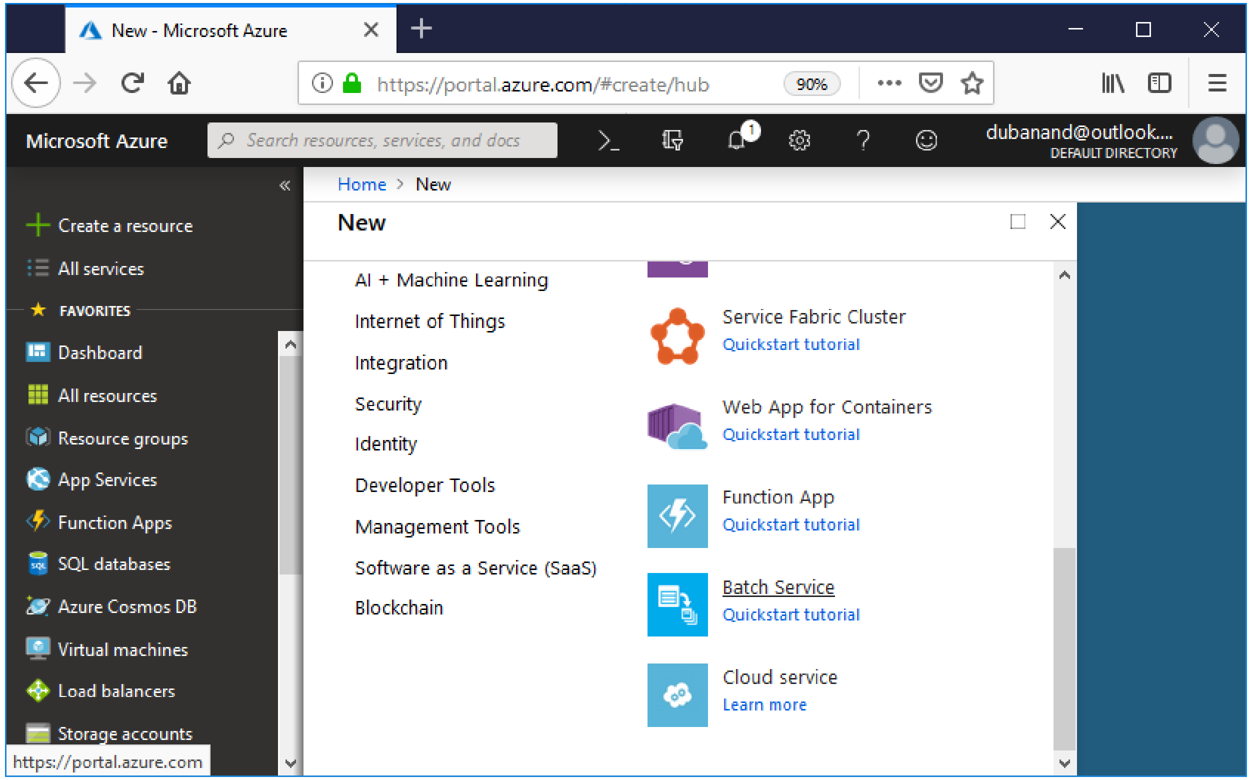
https://portal.azure.com.



Run your first Batch job (portal)

Create a Batch account

Follow these steps to create a sample Batch account for test purposes. You need a Batch account to create pools and jobs. As shown here, you can link an Azure storage account with the Batch account. Although not required for this quickstart, the storage account is useful to deploy applications and store input and output data for most real-world workloads.



1. Select ***Create a resource > Compute > Batch Service***.

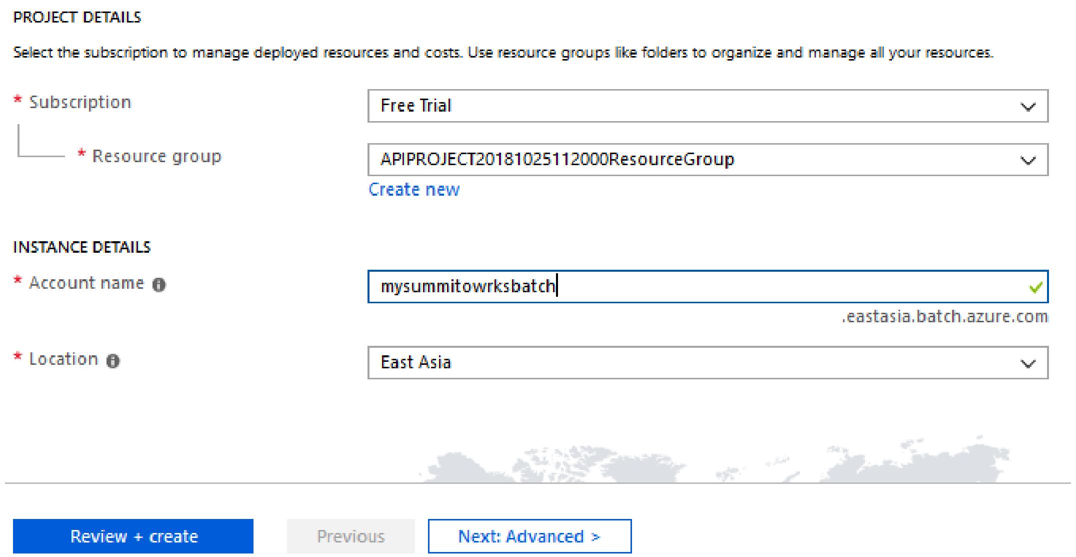
Run your first Batch job (portal)

2. Enter values for **Account name** and **Resource group**. The account name must be unique within the Azure **Location** selected, use only lowercase characters or numbers, and contain 3-24 characters.

3. In **Storage account**, select an existing storage account or create a new one.

4. Keep the defaults for remaining settings, and select **Create** to create the account.

When the **Deployment succeeded** message appears, go to the Batch account in the portal.



PROJECT DETAILS

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

* Subscription: Free Trial

* Resource group: APIPROJECT20181025112000ResourceGroup

Create new

INSTANCE DETAILS

* Account name: mysummitowrksbatch ✓

.eastasia.batch.azure.com

* Location: East Asia

Run your first Batch job (portal)

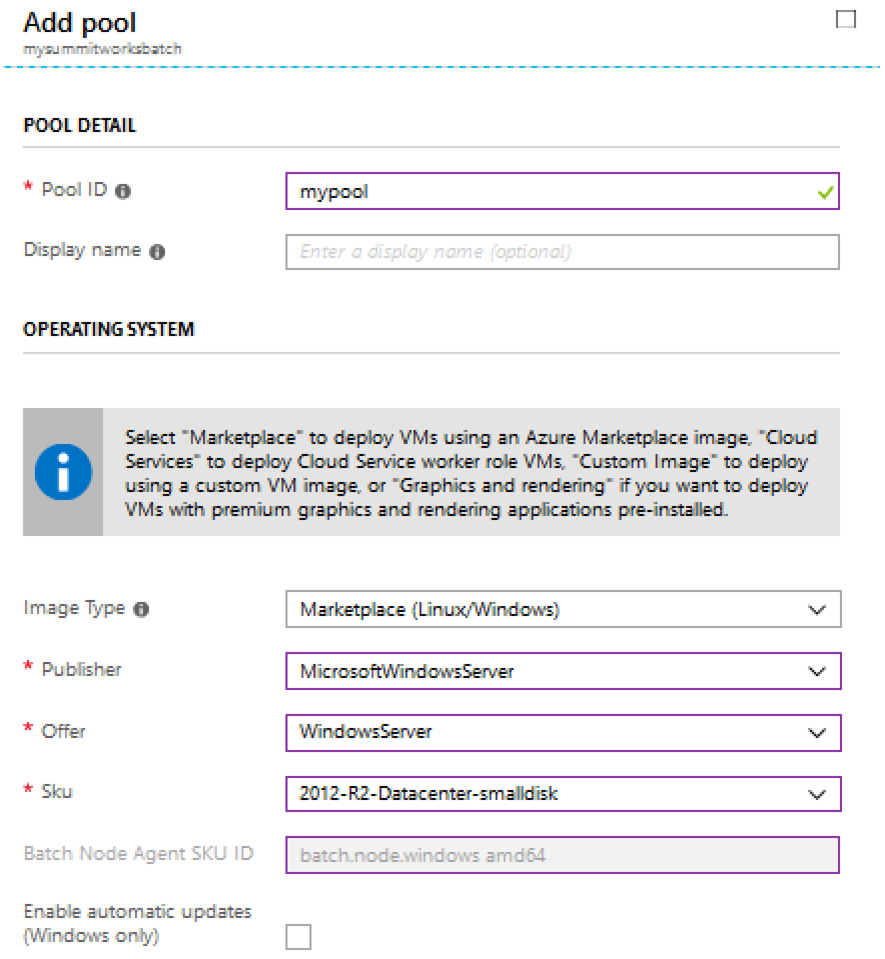
Create a pool of compute nodes

Now that you have a Batch account, create a sample pool of Windows compute nodes for test purposes. The pool for this quick example consists of 2 nodes running a Windows Server 2012 R2 image from the Azure Marketplace.

1. In the Batch account, select **Pools** > **Add**.

2. Enter a **Pool ID** called *mypool*.

3. In **Operating System**, select the following settings.



Add pool
mysummitworksbatch

POOL DETAIL

* Pool ID	mypool
Display name	Enter a display name (optional)

OPERATING SYSTEM

Select "Marketplace" to deploy VMs using an Azure Marketplace image, "Cloud Services" to deploy Cloud Service worker role VMs, "Custom Image" to deploy using a custom VM image, or "Graphics and rendering" if you want to deploy VMs with premium graphics and rendering applications pre-installed.

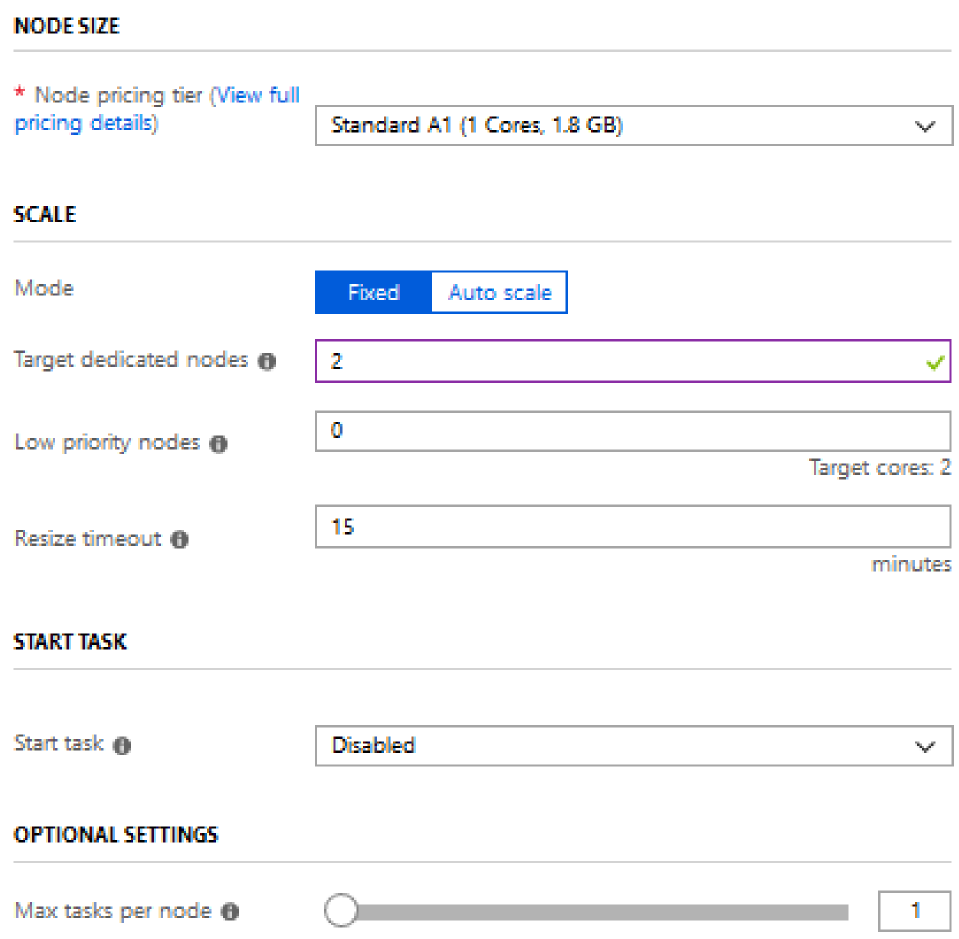
Image Type	Marketplace (Linux/Windows)
* Publisher	MicrosoftWindowsServer
* Offer	WindowsServer
* Sku	2012-R2-Datacenter-smalldisk
Batch Node Agent SKU ID	batch.node.windows amd64
Enable automatic updates (Windows only)	□

Run your first Batch job (portal)

4. Scroll down to enter **Node Size** and **Scale** settings. The suggested node size offers a good balance of performance versus cost for this quick example.

5. Keep the defaults for remaining settings, and select **OK** to create the pool.

Batch creates the pool immediately, but it takes a few minutes to allocate and start the compute nodes. During this time, the pool's **Allocation state** is **Resizing**. You can go ahead and create a job and tasks while the pool is resizing.



NODE SIZE

* Node pricing tier (View full pricing details): Standard A1 (1 Cores, 1.8 GB)

SCALE

Mode: [✓ Fixed] Auto scale

Target dedicated nodes: 2

Low priority nodes: 0

Target cores: 2

Resize timeout: 15 minutes

START TASK

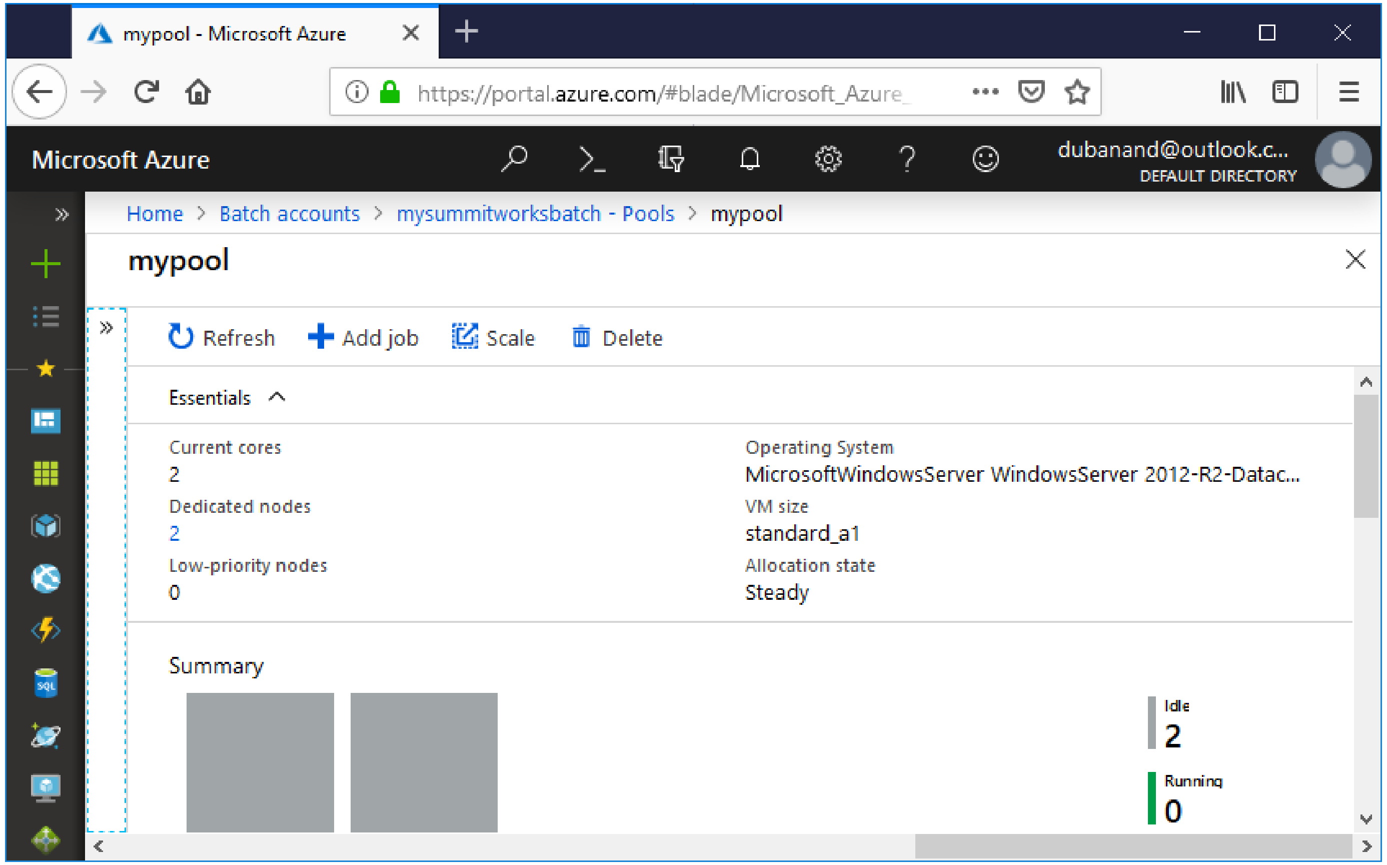
Start task: Disabled

OPTIONAL SETTINGS

Max tasks per node: 1

Run your first Batch job (portal)

After a few minutes, the state of the pool is **Steady**, and the nodes start. Select **Nodes** to check the state of the nodes. When a node's state is **Idle**, it is ready to run tasks.



Current cores: 2

Dedicated nodes: 2

Low-priority nodes: 0

Operating System: MicrosoftWindowsServer WindowsServer 2012-R2-Datac...

VM size: standard_a1

Allocation state: Steady

Idle: 2

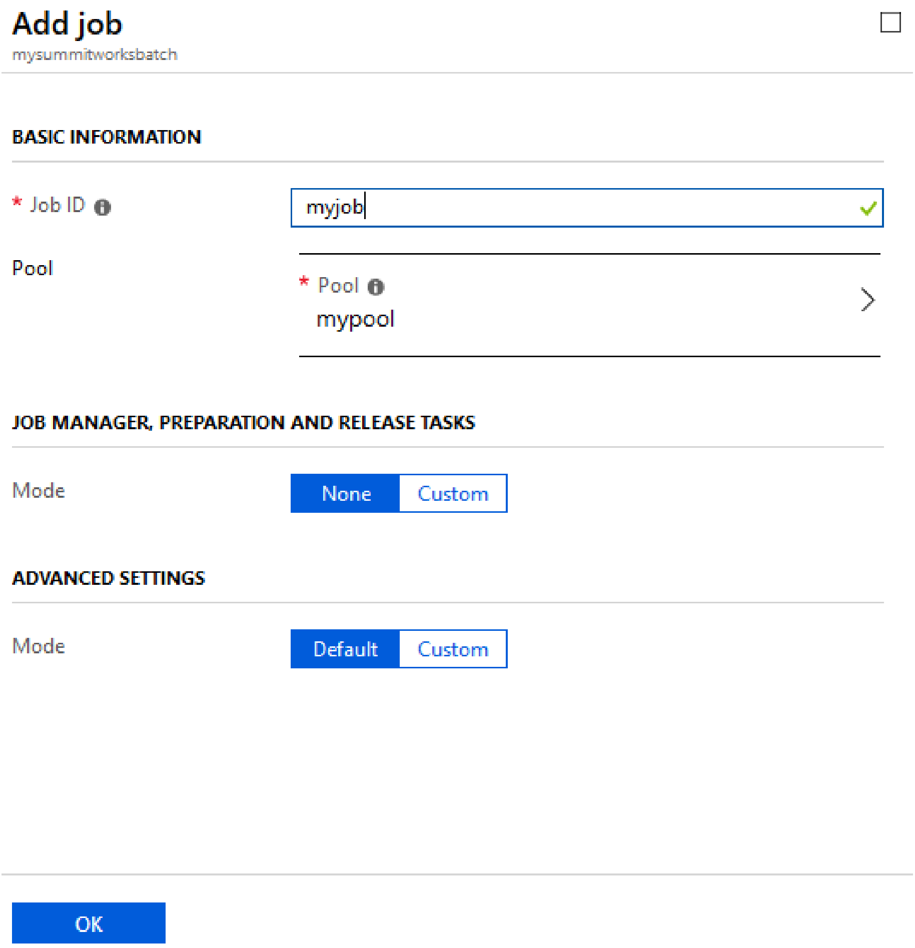
Running: 0

Run your first Batch job (portal)

Create a job

Now that you have a pool, create a job to run on it. A Batch job is a logical group for one or more tasks. A job includes settings common to the tasks, such as priority and the pool to run tasks on. Initially the job has no tasks.

1. In the Batch account view, select **Jobs** > **Add**.
2. Enter a **Job ID** called *myjob*. In **Pool**, select *mypool*. Keep the defaults for the remaining settings, and select **OK**.



Add job
mysummitworksbatch

BASIC INFORMATION

* Job ID: myjob

Pool: * Pool: mypool

JOB MANAGER, PREPARATION AND RELEASE TASKS

Mode: None | Custom

ADVANCED SETTINGS

Mode: Default | Custom

Run your first Batch job (portal)

Create tasks

Now create sample tasks to run in the job. Typically you create multiple tasks that Batch queues and distributes to run on the compute nodes. In this example, you create two identical tasks. Each task runs a command line to display the Batch environment variables on a compute node, and then waits 90 seconds.

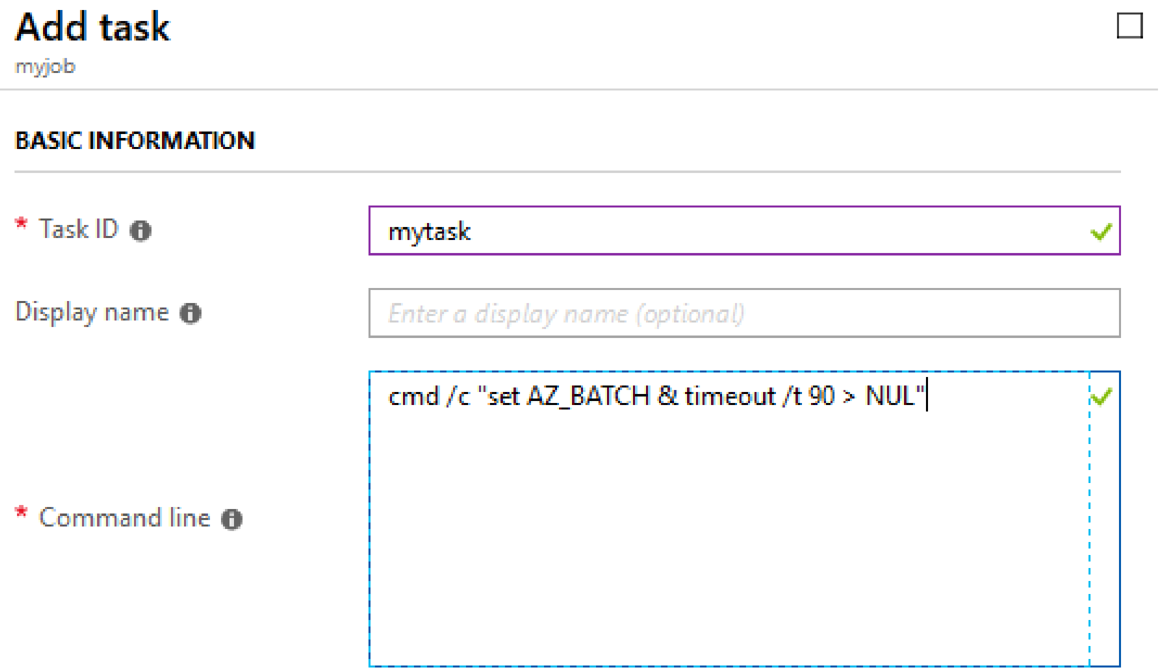
When you use Batch, the command line is where you specify your app or script. Batch provides a number of ways to deploy apps and scripts to compute nodes.

To create the first task:

1. Select **Add**.
2. Enter a **Task ID** called *mytask*.

Run your first Batch job (portal)

In Command line, enter *cmd /c "set AZ_BATCH & timeout /t 90 > NUL“*. Keep the defaults for the remaining settings, and select OK.



Add task

myjob

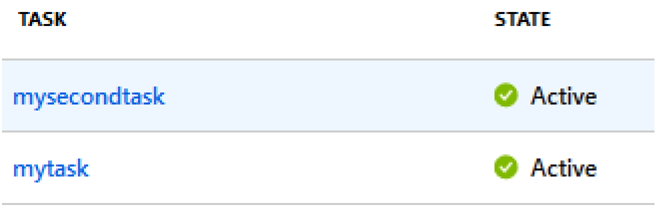
BASIC INFORMATION

* Task ID: mytask

Display name: Enter a display name (optional)

* Command line: cmd /c "set AZ_BATCH & timeout /t 90 > NUL"

After you create a task, Batch queues it to run on the pool. When a node is available to run it, the task runs.



TASK	STATE
mysecondtask	✓ Active
mytask	✓ Active

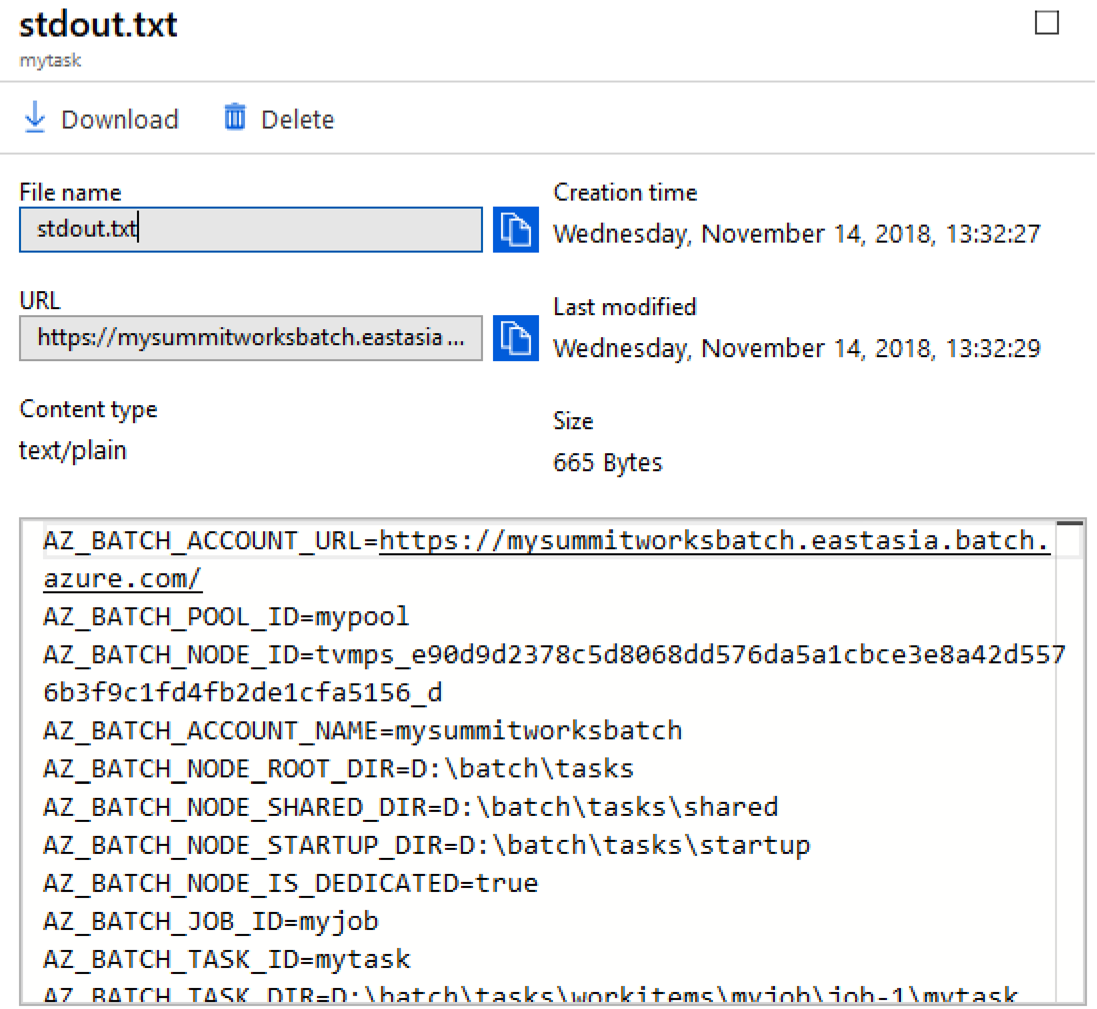
To create a second task, go back to step 1. Enter a different **Task ID**, but specify an identical command line. If the first task is still running, Batch starts the second task on the other node in the pool.

Run your first Batch job (portal)

View task output

The preceding task examples complete in a couple of minutes. To view the output of a completed task, select Files on node, and then select the file stdout.txt. This file shows the standard output of the task. The contents are similar to the following.

The contents show the Azure Batch environment variables that are set on the node. When you create your own Batch jobs and tasks, you can reference these environment variables in task command lines, and in the apps and scripts run by the command lines.



stdout.txt □
mytask

Download   Delete

File name: stdout.txt

Creation time: Wednesday, November 14, 2018, 13:32:27

URL: https://mysummitworksbatch.eastasia ...

Last modified: Wednesday, November 14, 2018, 13:32:29

Content type: text/plain

Size: 665 Bytes

```
AZ_BATCH_ACCOUNT_URL=https://mysummitworksbatch.eastasia.batch.
azure.com/
AZ_BATCH_POOL_ID=mypool
AZ_BATCH_NODE_ID=tvmps_e90d9d2378c5d8068dd576da5a1cbce3e8a42d557
6b3f9c1fd4fb2de1cfa5156_d
AZ_BATCH_ACCOUNT_NAME=mysummitworksbatch
AZ_BATCH_NODE_ROOT_DIR=D:\batch\tasks
AZ_BATCH_NODE_SHARED_DIR=D:\batch\tasks\shared
AZ_BATCH_NODE_STARTUP_DIR=D:\batch\tasks\startup
AZ_BATCH_NODE_IS_DEDICATED=true
AZ_BATCH_JOB_ID=myjob
AZ_BATCH_TASK_ID=mytask
AZ_BATCH_TASK_DIR=D:\batch\tasks\workitems\myjob\job-1\mytask
```

Run your first Batch job (VS)

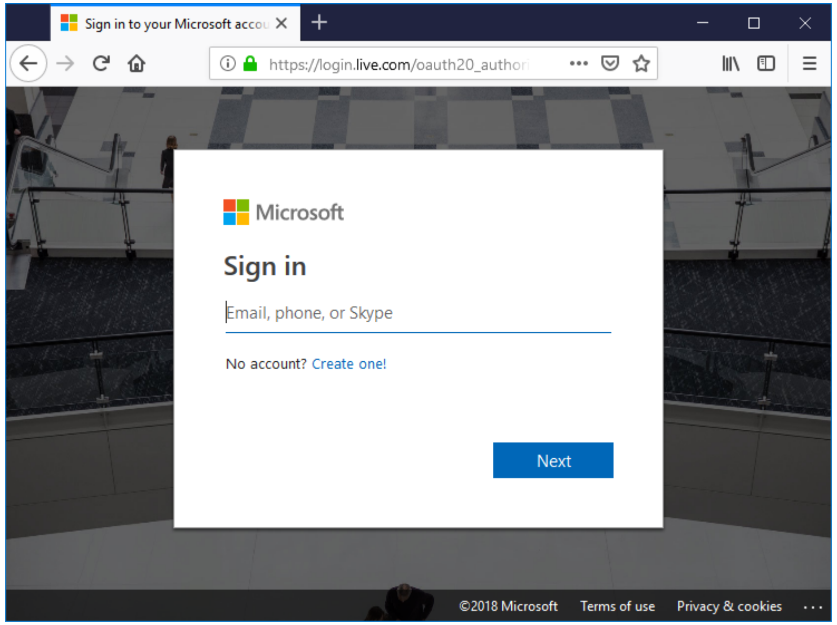
Sign in to the Azure portal at

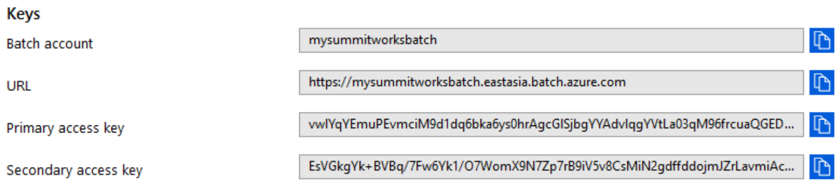
https://portal.azure.com.

Get account credentials

For this example, you need to provide credentials for your Batch and Storage accounts. A straightforward way to get the necessary credentials is in the Azure portal.

1. Click **All services** > **Batch accounts**, and then click the name of your Batch account.
2. To see the Batch credentials, click **Keys**. Copy the values of **Batch account**, **URL**, and **Primary access key** to a text editor.
3. To see the Storage account name and keys, click **Storage account**. Copy the values of **Storage account name** and **Key1** to a text editor.





Keys

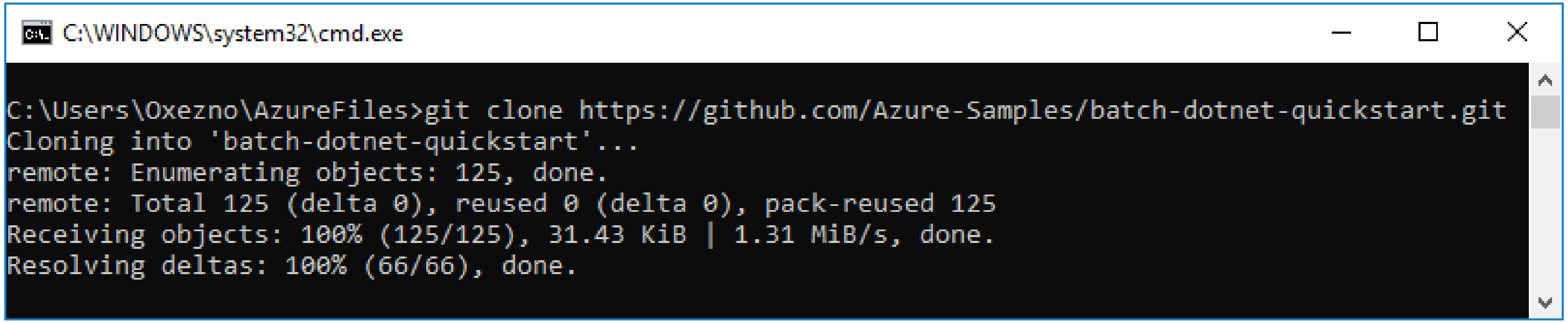
Batch account	mysummitworksbatch
URL	https://mysummitworksbatch.eastasia.batch.azure.com
Primary access key	vwlYqYEmuPEvmciM9d1dq6bka6ys0hrAgcGlSjbgYYAdvlqgYVtLa03qM96frcuaQGED...
Secondary access key	EsVGkgYk+BVBq/7Fw6Yk1/O7WomX9N7Zp7rB9iV5v8CsMiN2gdffddojmJZrLavmiAc...

Run your first Batch job (VS)

Download the sample

Download or clone the sample app from GitHub. To clone the sample app repo with a Git client, use the following command:

- *git clone https://github.com/Azure-Samples/batch-dotnet-quickstart.git*

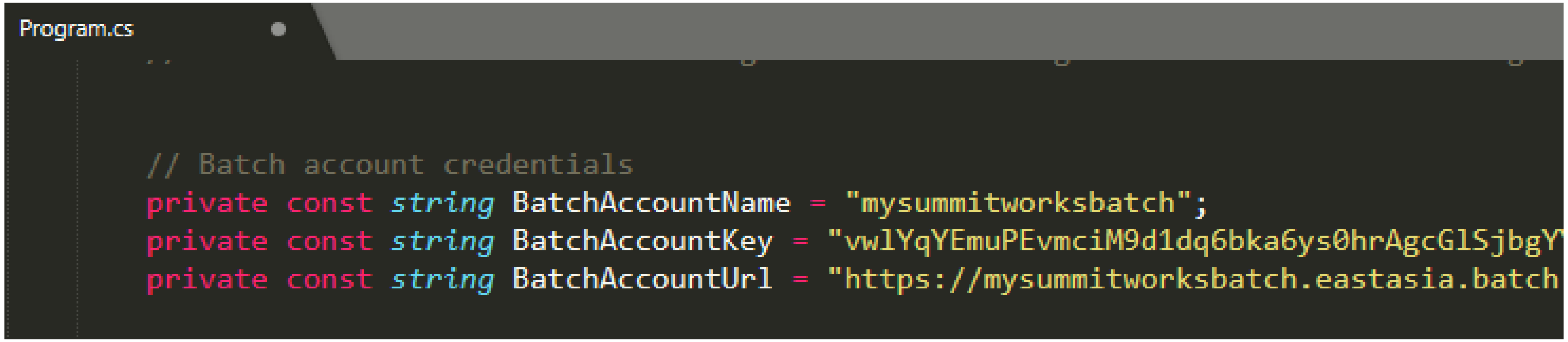


```
C:\Users\Oxezno\AzureFiles>git clone https://github.com/Azure-Samples/batch-dotnet-quickstart.git
Cloning into 'batch-dotnet-quickstart'...
remote: Enumerating objects: 125, done.
remote: Total 125 (delta 0), reused 0 (delta 0), pack-reused 125
Receiving objects: 100% (125/125), 31.43 KiB | 1.31 MiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Run your first Batch job (VS)

Navigate to the directory that contains the Visual Studio solution file *BatchDotNetQuickstart.sln*

Open the solution file in Visual Studio, and update the credential strings in *program.cs* with the values you obtained for your accounts. For example:



```
// Batch account credentials
private const string BatchAccountName = "mysummitworksbatch";
private const string BatchAccountKey = "vwlYqYEmuPEvmciM9d1dq6bka6ys0hrAgcGlSjbgY
private const string BatchAccountUrl = "https://mysummitworksbatch.eastasia.batch
```

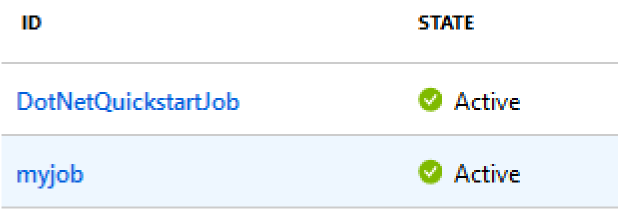
Run your first Batch job (VS)

Build and run the app

In Visual Studio:

- Right-click the solution in Solution Explorer, and click Build Solution.
- Confirm the restoration of any NuGet packages, if you're prompted. If you need to download missing packages, ensure the NuGet Package Manager is installed.

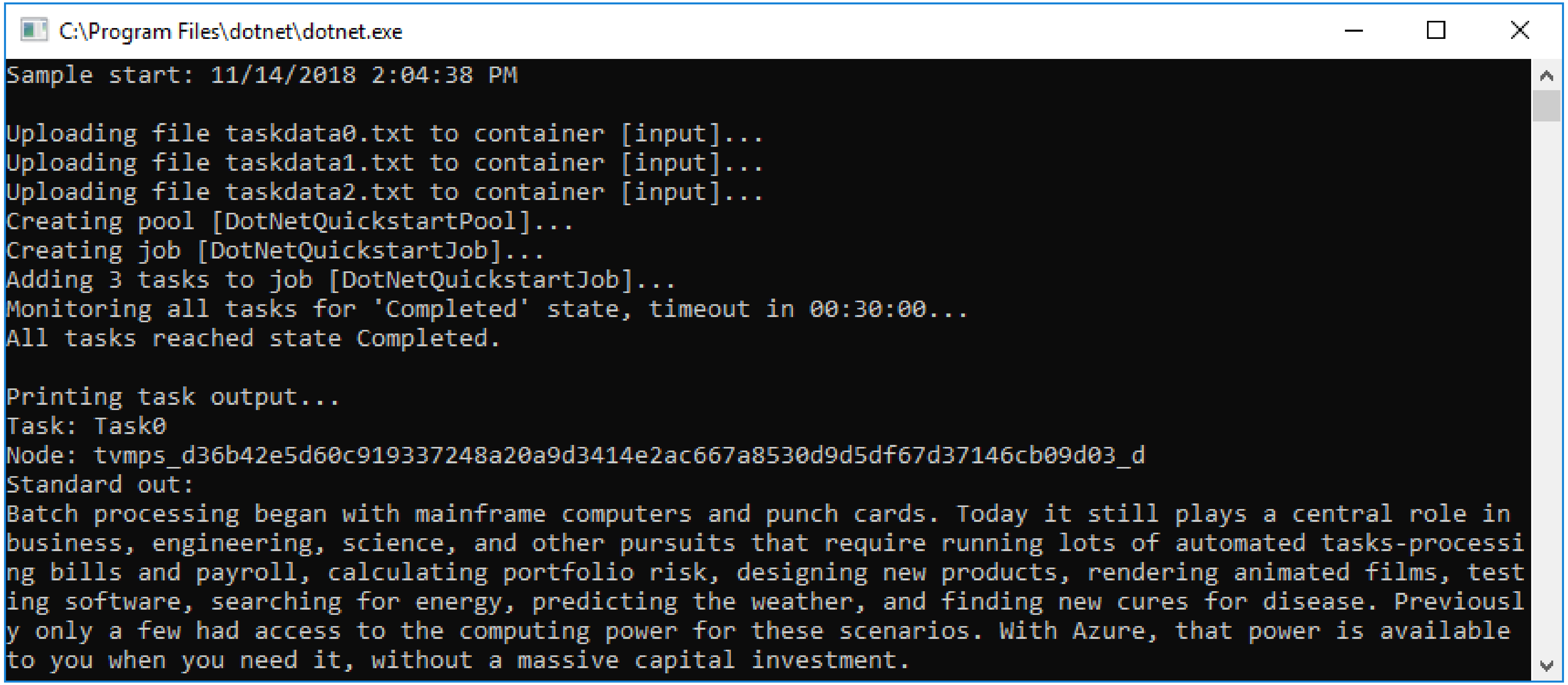
Then run it. When you run the sample application, the console output is similar to the following. During execution, you experience a pause at *Monitoring all tasks for 'Completed' state, timeout in 00:30:00...* while the pool's compute nodes are started. Tasks are queued to run as soon as the first compute node is running. Go to your Batch account in the Azure portal to monitor the pool, compute nodes, job, and tasks.



ID	STATE
DotNetQuickstartJob	✓ Active
myjob	✓ Active

Run your first Batch job (VS)

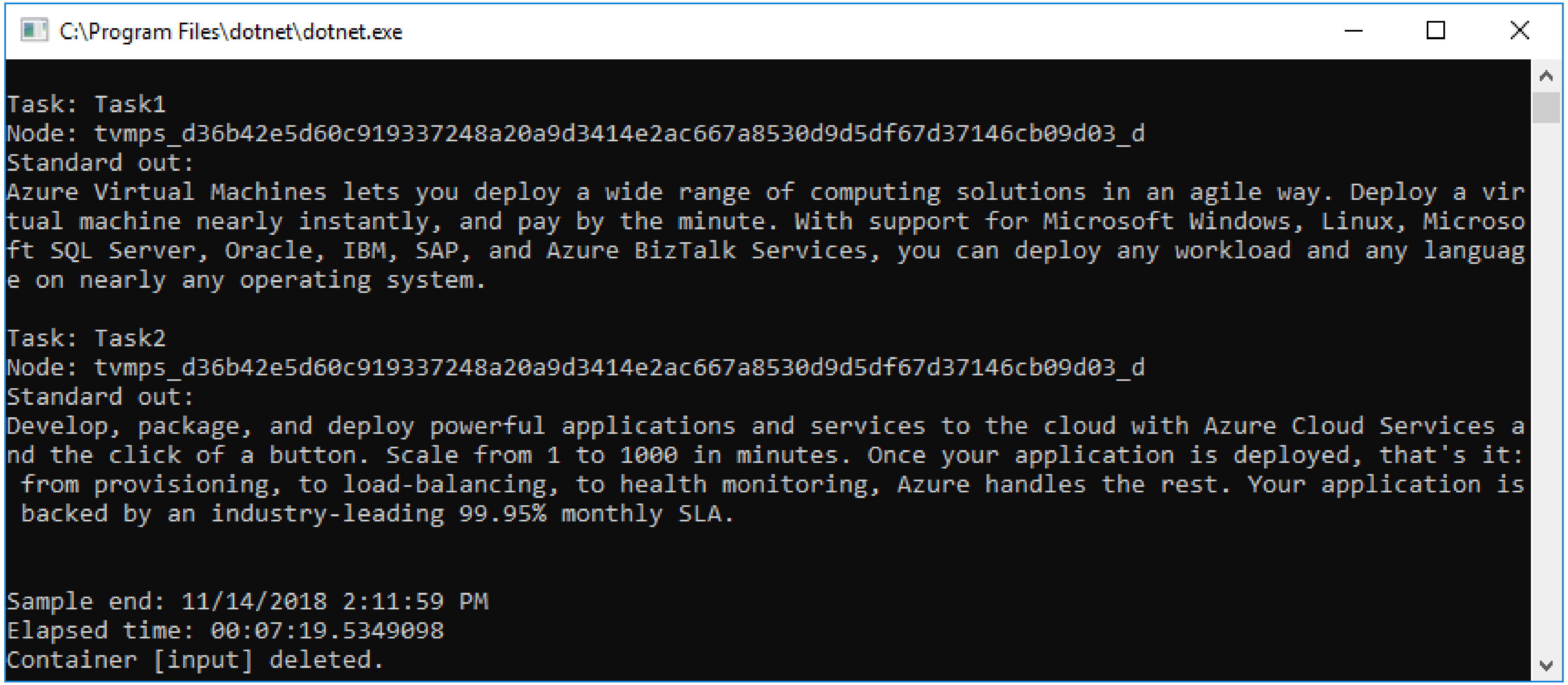
After tasks complete, you see output similar to the following for each task:



```
Sample start: 11/14/2018 2:04:38 PM

Uploading file taskdata0.txt to container [input]...
Uploading file taskdata1.txt to container [input]...
Uploading file taskdata2.txt to container [input]...
Creating pool [DotNetQuickstartPool]...
Creating job [DotNetQuickstartJob]...
Adding 3 tasks to job [DotNetQuickstartJob]...
Monitoring all tasks for 'Completed' state, timeout in 00:30:00...
All tasks reached state Completed.

Printing task output...
Task: Task0
Node: tvmps_d36b42e5d60c919337248a20a9d3414e2ac667a8530d9d5df67d37146cb09d03_d
Standard out:
Batch processing began with mainframe computers and punch cards. Today it still plays a central role in
business, engineering, science, and other pursuits that require running lots of automated tasks-processi
ng bills and payroll, calculating portfolio risk, designing new products, rendering animated films, test
ing software, searching for energy, predicting the weather, and finding new cures for disease. Previousl
y only a few had access to the computing power for these scenarios. With Azure, that power is available
to you when you need it, without a massive capital investment.
```

Run your first Batch job (VS)

Typical execution time is approximately 5 minutes when you run the application in its default configuration. Initial pool setup takes the most time. To run the job again, delete the job from the previous run and do not delete the pool. On a preconfigured pool, the job completes in a few seconds.



C:\Program Files\dotnet\dotnet.exe

```
Task: Task1
Node: tvmps_d36b42e5d60c919337248a20a9d3414e2ac667a8530d9d5df67d37146cb09d03_d
Standard out:
Azure Virtual Machines lets you deploy a wide range of computing solutions in an agile way. Deploy a vir
tual machine nearly instantly, and pay by the minute. With support for Microsoft Windows, Linux, Microso
ft SQL Server, Oracle, IBM, SAP, and Azure BizTalk Services, you can deploy any workload and any languag
e on nearly any operating system.

Task: Task2
Node: tvmps_d36b42e5d60c919337248a20a9d3414e2ac667a8530d9d5df67d37146cb09d03_d
Standard out:
Develop, package, and deploy powerful applications and services to the cloud with Azure Cloud Services a
nd the click of a button. Scale from 1 to 1000 in minutes. Once your application is deployed, that's it:
 from provisioning, to load-balancing, to health monitoring, Azure handles the rest. Your application is
 backed by an industry-leading 99.95% monthly SLA.


Sample end: 11/14/2018 2:11:59 PM
Elapsed time: 00:07:19.5349098
Container [input] deleted.
```

Run your first Batch job (VS)

The .NET app for this example does the following:

- Uploads three small text files to a blob container in your Azure storage account. These files are inputs for processing by Batch.
- Creates a pool of compute nodes running Windows Server.
- Creates a job and three tasks to run on the nodes. Each task processes one of the input files using a Windows command line.
- Displays files returned by the tasks.