

Development

Setting up the solution

We are starting right at the beginning. At first you'll need a solution to place the project in.

Navigate to a directory where you want the source to be in. For me it is

C:\Repos\fiveerr\jerrymartinez1. Open a console and type `dotnet new sln`, which will create a solution file. Open it. I am using Visual Studio 2022 Community Edition as the IDE of my choice.

Add a new Project.

Search for Azure Functions like so:

Hit Next and give a name. I'll choose `SoftDeleteDetection`. Hit Next.

Fill in the information like here. You want to use

- .NET6.0 (because it's the current version)
- Timer trigger (so that the function gets executed automatically)
- Azurite (because it's default)
- Docker if you want to (optional)
- Schedule is every 5 Minutes by default. I set it to every Monday at 9 AM. Here's a [cheat sheet](#)

Hit Create - The project will be created.

Rename the function and use file scoped namespaces

By default the function will be named `Function1`, which is not very meaningful. I change it to `NotifyOnSoftDeletedResourcesFunctions`. Sounds like a lot - but now everyone knows what this function is supposed to do without having to take a look inside.

I like to change filenames through the solution explorer - because then a dialog appears asking if the class name should also be changed.

Your class now looks like this:

```
using System;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Host;
using Microsoft.Extensions.Logging;

namespace SoftDeleteDetection
{
    public class NotifyOnSoftDeletedResourcesFunctions
    {
        [FunctionName("Function1")]
        public void Run([TimerTrigger("0 0 9 * * MON")]TimerInfo myTimer, IL
```

```

    {
        log.LogInformation($"C# Timer trigger function executed at: {DateTime.Now}");
    }
}

```

A class may contain multiple functions. That is why I pluralized the name - but this is just personal preference.

Change the name in the FunctionName-attribute and the method name Run as well to ScanAndNotify - again to have something meaningful.

```

public class NotifyOnSoftDeletedResourcesFunctions
{
    [FunctionName("ScanAndNotify")]
    public void ScanAndNotify([TimerTrigger("0 0 9 * * MON")]TimerInfo timerInfo)
    {
        log.LogInformation($"C# Timer trigger function executed at: {DateTime.Now}");
    }
}

```

With dotnet6.0 comes file scoped namespaces. We can switch to that by typing a semicolon behind namespace SoftDeleteDetection.

Our code now looks like this:

```

using System;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Host;
using Microsoft.Extensions.Logging;

namespace SoftDeleteDetection;

public class NotifyOnSoftDeletedResourcesFunctions
{
    [FunctionName("ScanAndNotify")]
    public void ScanAndNotify([TimerTrigger("0 0 9 * * MON")]TimerInfo timerInfo)
    {
        log.LogInformation($"C# Timer trigger function executed at: {DateTime.Now}");
    }
}

```

■ The cron expression from the start is now inside the TimerTrigger(...). Change it as you like.

Settings

We will need to store some settings in our application. For that we have the file `local.settings.json`

Content looks like this:

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "dotnet"
  }
}
```

We want to send an email, so let's do an example here for report receipts.

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "dotnet",
    "ReportReceipts": "kontakt@fistelmann.de;ichhabjanixbessereszutun@we
  }
}
```

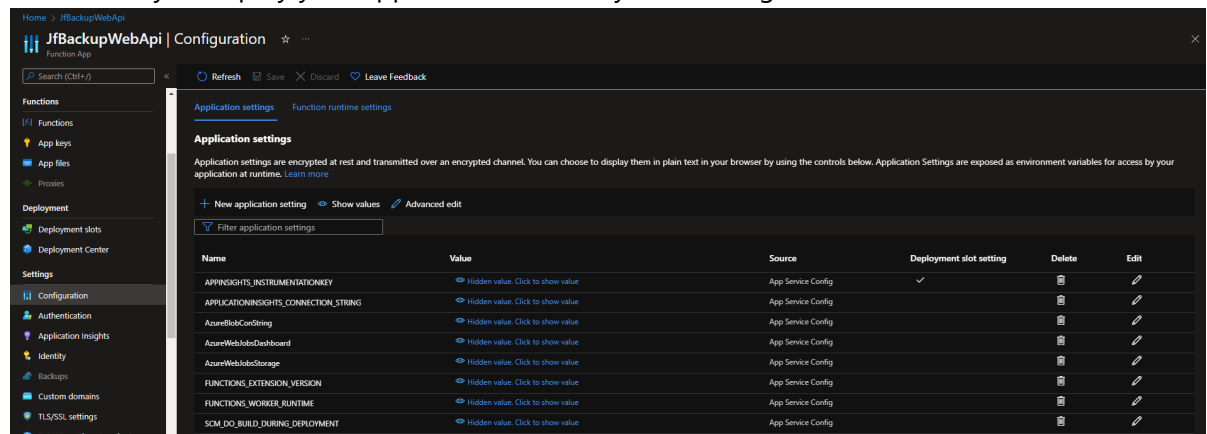


We can access the value inside our code like this:

```
string reportReceipts = Environment.GetEnvironmentVariable("ReportReceipts");
```



Later when you deploy your application to azure you set things like this:



In order to have an overview of the settings I tend to create a class with properties of settings names.

```
namespace SoftDeleteDetection.Models;

public static class Settings
{
    public const string ReportReceipts = "ReportReceipts";
}
```

If you do this, you retrieve a setting like this:

```
string reportReceipents = Environment.GetEnvironmentVariable(Settings.Report
```

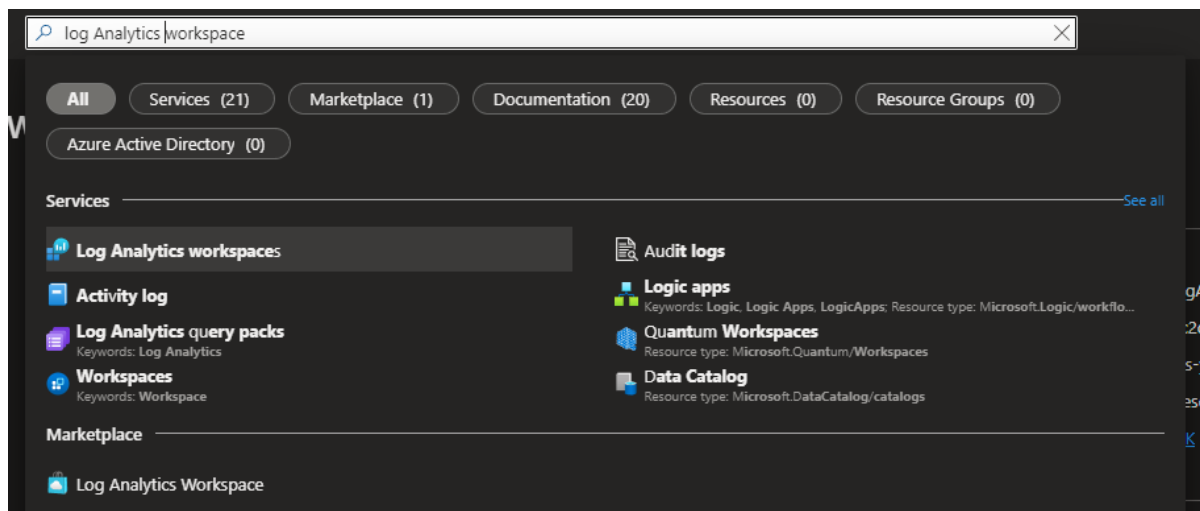
Setup Azure Monitor

We will use the [Azure Monitor](#) resource in order to get information about deletions of Resources and Azure Key Vault entries.

Azure Monitor Logs

For it to work properly we have to do some setup.

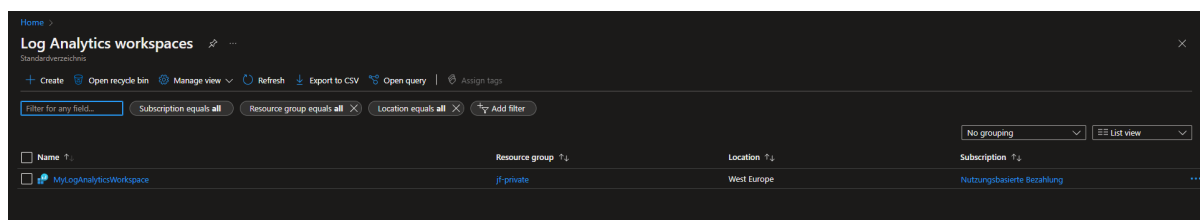
1. Create a log workspace
2. Set Diagnostic settings on resources



Create one if not existant.

Keep note of the workspace id as you will need it later.

<https://portal.azure.com/#view/HubsExtension/BrowseResource/resourceType/Microsoft.OperationalInsights%2Fworkspaces>



Now that we have a workspace we need to tell our resources to write logs there.

https://portal.azure.com/#view/Microsoft_Azure_Monitoring/AzureMonitoringBrowseBlade/~/~diagnosticsLogs

Home > Monitor

Monitor | Diagnostic settings

Microsoft

Search (Ctrl+/) Refresh Feedback

Subscription: Nutzungsbasierte Bezahlung Resource group: [No filter] Resource type: 0 selected

Select any of the resources to view diagnostic settings.

Name	Resource type	Resource group	Diagnostics status
AppCenterExport-TT2-Master	Application Insights	appcenter-export-TT2-Master	Not Available
BinanceNewsCrawler	Application Insights	appcenter-export-TT2-Master	Not Available
csb10033fff9c24fc55	Storage account	cloud-shell-storage-west europe	Disabled
blob	Storage account	cloud-shell-storage-west europe	Disabled
queue	Storage account	cloud-shell-storage-west europe	Disabled
table	Storage account	cloud-shell-storage-west europe	Disabled
file	Storage account	cloud-shell-storage-west europe	Disabled
BinanceNewsCrawlerInsights	Application Insights	jf-private	Not Available
JfBackupWebApi	Application Insights	jf-private	Not Available
jf-private-keyvault	Key vault	jf-private	Enabled
MyLogAnalyticsWorkspace	Log Analytics workspace	jf-private	Disabled
jfprivatestorage	Storage account	jf-private	Enabled
blob	Storage account	jf-private	Disabled
queue	Storage account	jf-private	Disabled
table	Storage account	jf-private	Disabled
file	Storage account	jf-private	Disabled
TranslationTT2Master	Cognitive Service	TT2Master	Disabled

We will do this for our Key Vault resource. Click on it and you will be redirected to a menu where you can set this up.

Microsoft Azure

Search resources, services, and docs (Ctrl+/)

Home > Monitor

Monitor | Diagnostic settings

Microsoft

Subscription: Nutzungsbasierte Bezahlung Resource group: jf-private Resource type: Key vaults Resource: jf-private-keyvault

Nutzungsbasierte Bezahlung > jf-private > jf-private-keyvault

Diagnostic settings are used to configure streaming export of platform logs and metrics for a resource to the destination of your choice. You may create up to five different diagnostic settings to send different logs and metrics to independent destinations. [Learn more about diagnostic settings](#)

Name	Storage account	Event hub	Log Analytics workspace	Partner solution	Edit setting
jf-private-keyvault-diagnostics	-	-	myloganalyticsworkspace	-	Edit setting

[Add diagnostic settings](#)

Click 'Add Diagnostic setting' above to configure the collection of the following data:

- AuditEvent
- AzurePolicyEvaluationDetails
- AllMetrics

Home > Monitor >

Diagnostic setting

Save Discard Delete Feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. [Learn more about the different log categories and contents of those logs](#)

Diagnostic setting name: KeyVaultAuditLogs

Logs

Category groups: ☒ audit ☐ allLogs

Categories: ☒ AuditEvent ☒ AzurePolicyEvaluationDetails

Metrics

☐ AllMetrics

Destination details

☒ Send to Log Analytics workspace

Subscription: Nutzungsbasierte Bezahlung

Log Analytics workspace: MyLogAnalyticsWorkspace (west europe)

☐ Archive to a storage account

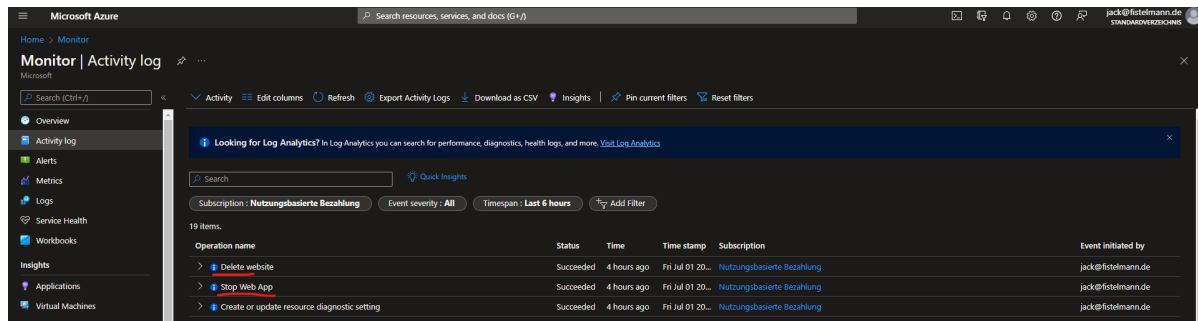
☐ Stream to an event hub

☐ Send to partner solution

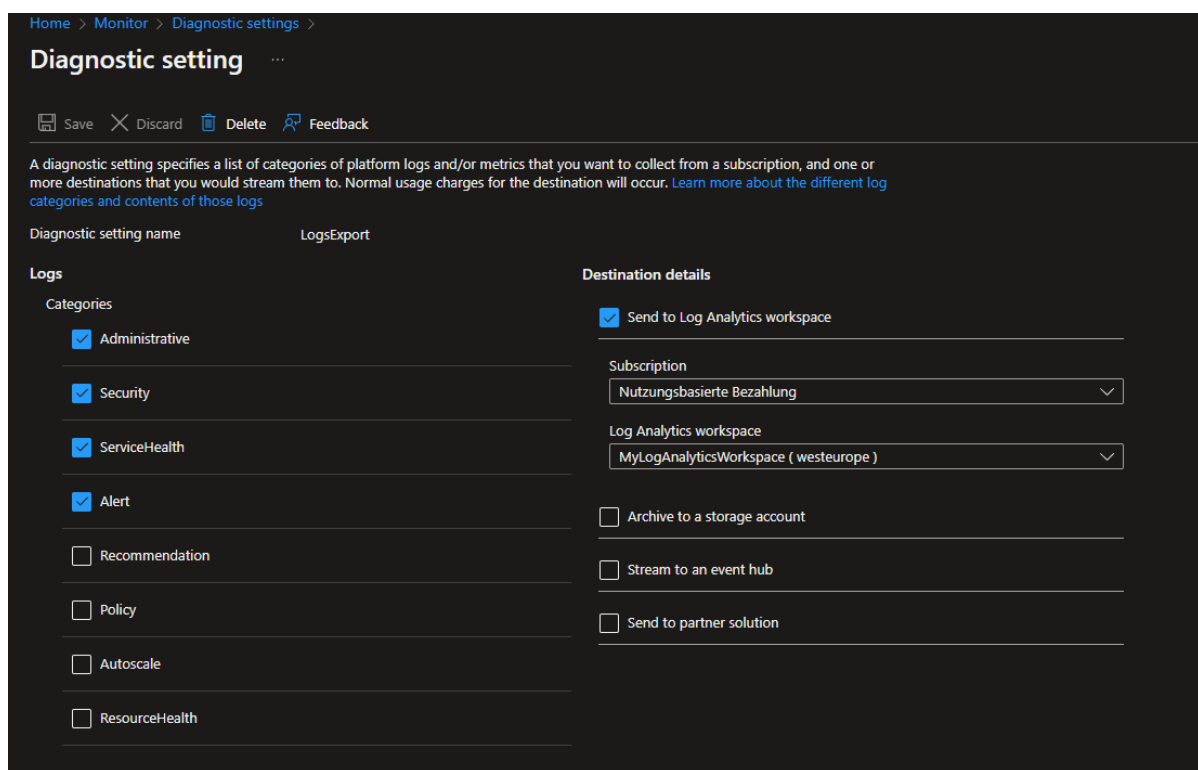
Now we will retrieve diagnostic log entries when something important happens to our key vault.

The next step is to get important log entries from our activity log.

https://portal.azure.com/#view/Microsoft_Azure_Monitoring/AzureMonitoringBrowseBlade/~/.activityLog



Click on Export Activity Logs and set up a Diagnostic setting just like you did for the key vault.



That's it. We don't need more than those two.

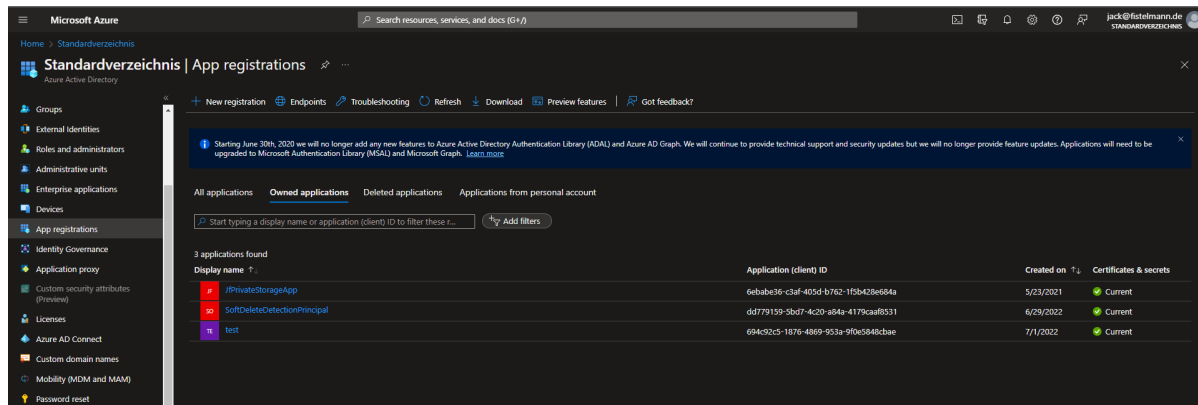
Access Azure Monitor

We are now able to perform Queries in the azure portal. But we want an azure function to query the resources for us. There are multiple ways to grant access. We will use client credentials for this, as the setup is pretty straight forward.

Go to the Active Directory resource:

https://portal.azure.com/#view/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/~/.Overview

And then to App Registrations

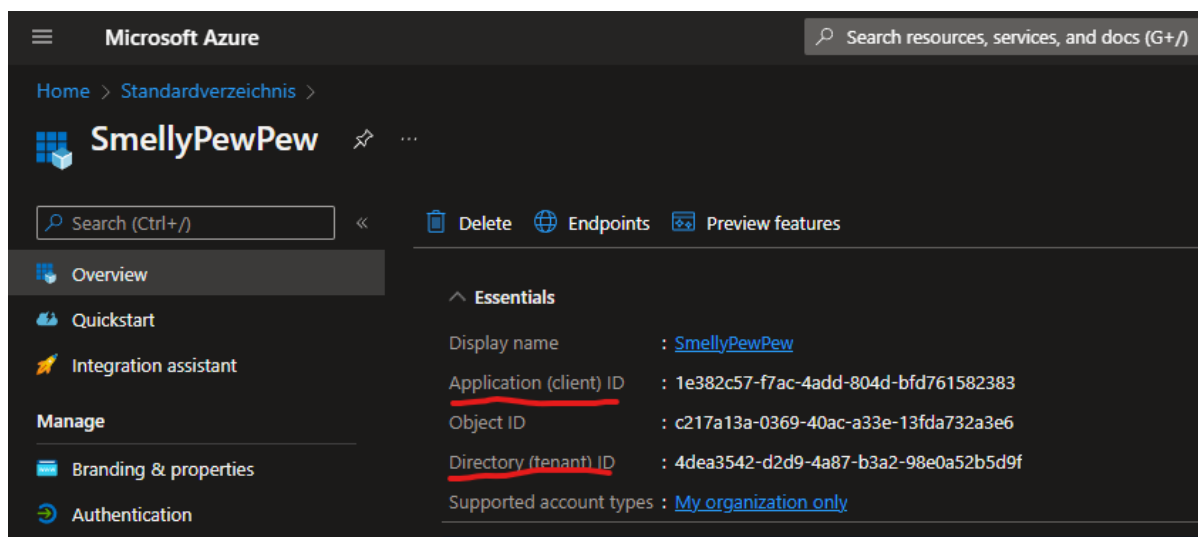


Click on + New registration to create a new one.

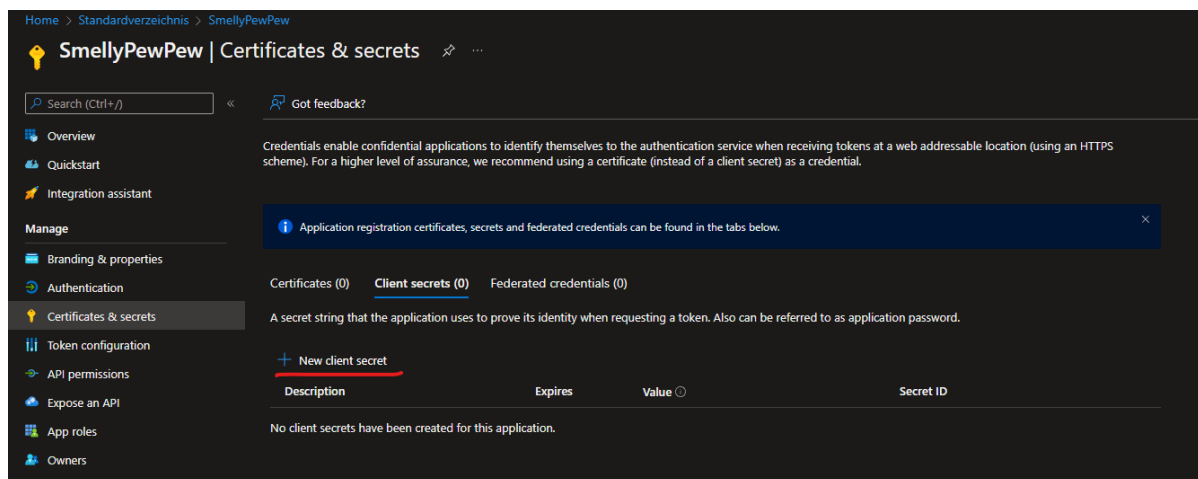
Give a meaningful name. Supported account types: Accounts in this organizational directory only (Standardverzeichnis only - Single tenant) (default) Redirect URI (optional): ignore this one.

Hit Register and you will be redirected to the new app registration.

Take a note of the Application (client) ID and the Directory (tenant) ID as you will need them later.

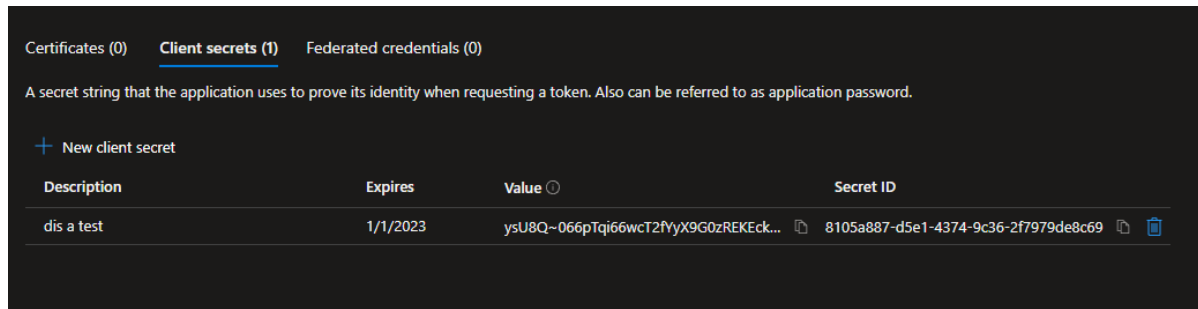


On the left menu, go to Certificates & secrets. Create a client secret.



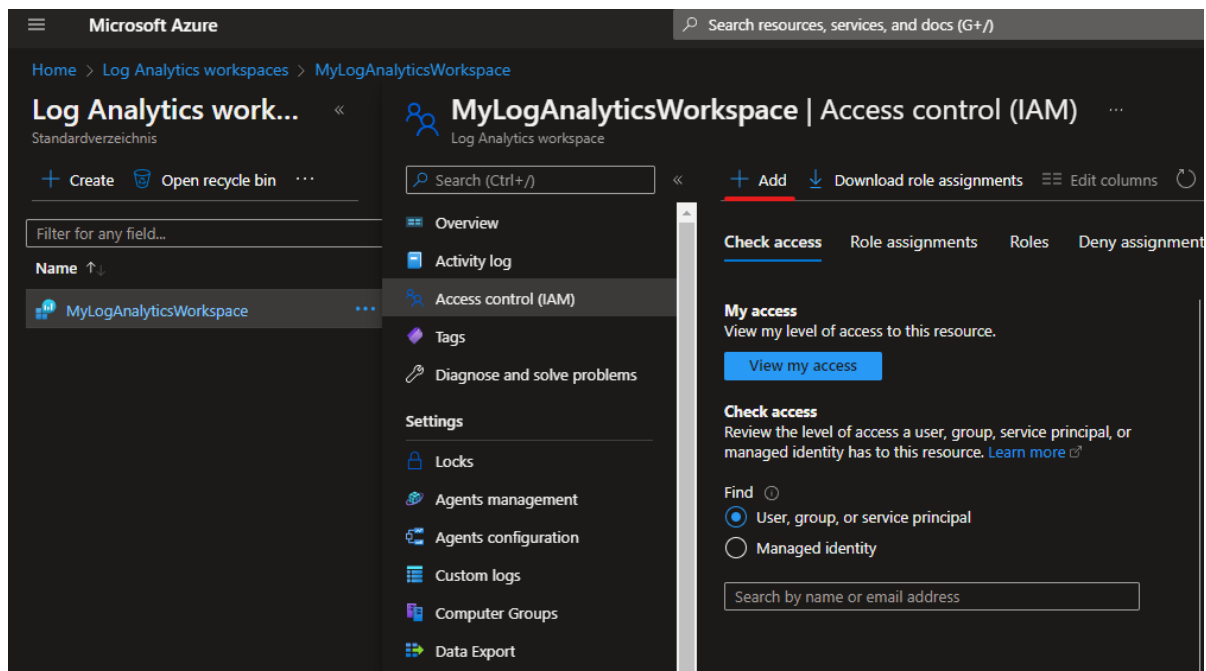
Enter a description and Expiry.

Remember that you need to refresh this from time to time. That is the nature of client secrets.



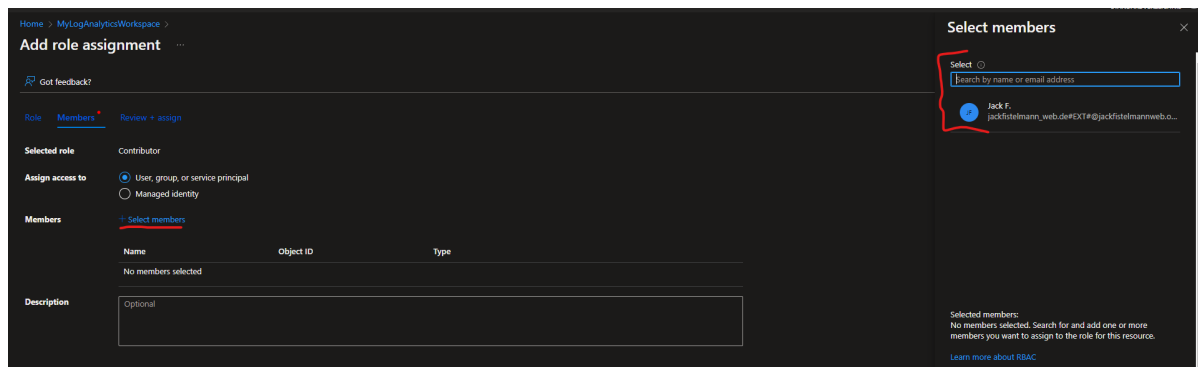
Keep note of the Value as you will need this later.

We now need to perform one final step. Giving this new app registration access to our Log Analytics Workspace.

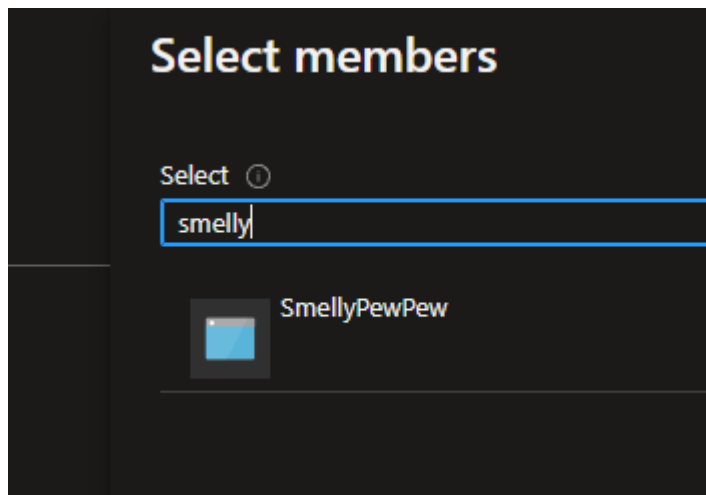


Click Add and Add role assignment.

Choose the Contributor role and hit next. Click on select members.



Now comes a pretty weird part. On the right side there is a text box. You need to type in the name of your recently created app resource in order for it to be shown.



Click on the item to select it and then the blue select button on the bottom right corner.

Add role assignment ...

[Got feedback?](#)

Role **Members** Review + assign

Selected role Contributor

Assign access to ☒ User, group, or service principal ☐ Managed identity

Members + Select members

Name	Object ID	Type
SmellyPewPew	0a2cbf77-687b-4529-a5f6-a8c0cc7856a6	App

Description Optional

Review + assign Previous Next

Hit review + assign.

Done.

Write the code

We will use three Nuget Packages.

- Azure.Identity (to use our client credentials and authorize)
- Azure.Monitor.Query (to query the logs)
- MailKit (to send emails)

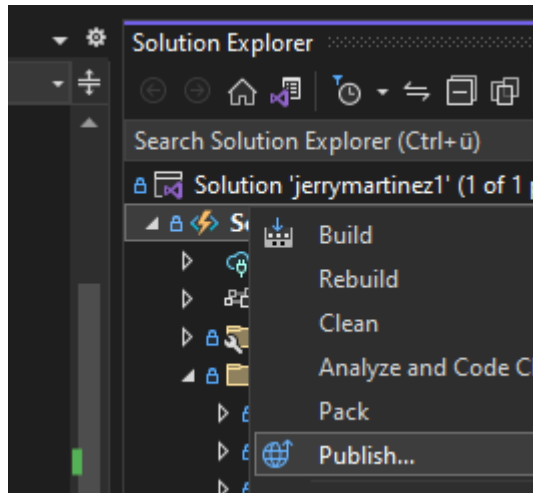
The code is very straight forward, so I will not document it here again.

Deploy Azure function

There are multiple ways to deploy an azure function.

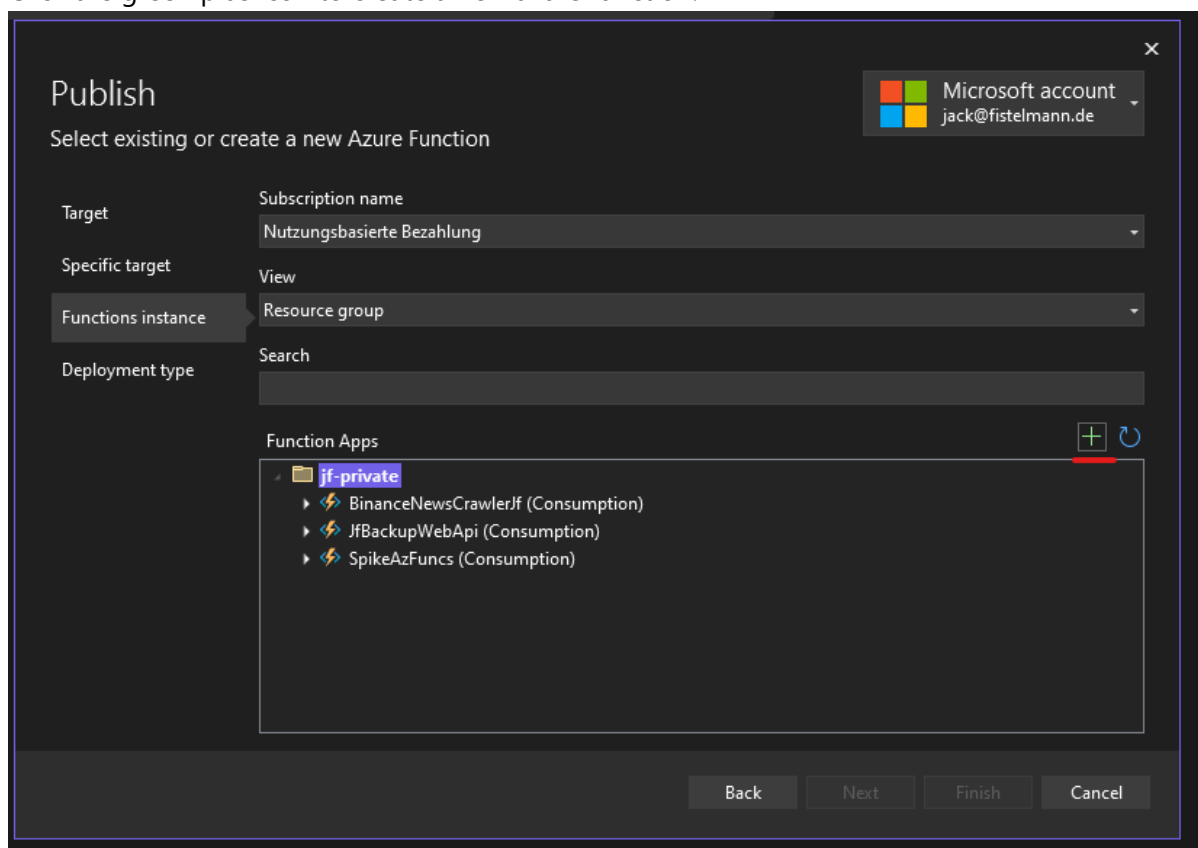
In this example I will deploy it from within Visual Studio.

From your Solution explorer, right click the project and hit publish.



Select Azure and Next. Select Azure Function App (Linux).

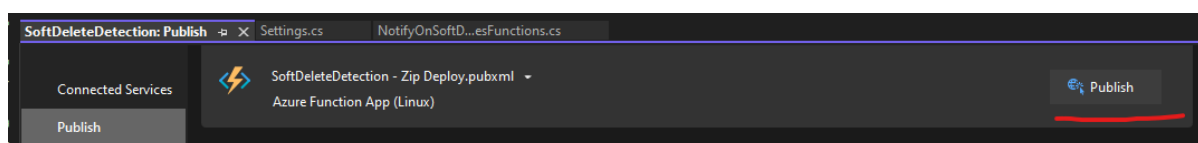
Click the green plus icon to create a new azure function.



Fill in everything as you need it and hit Create.

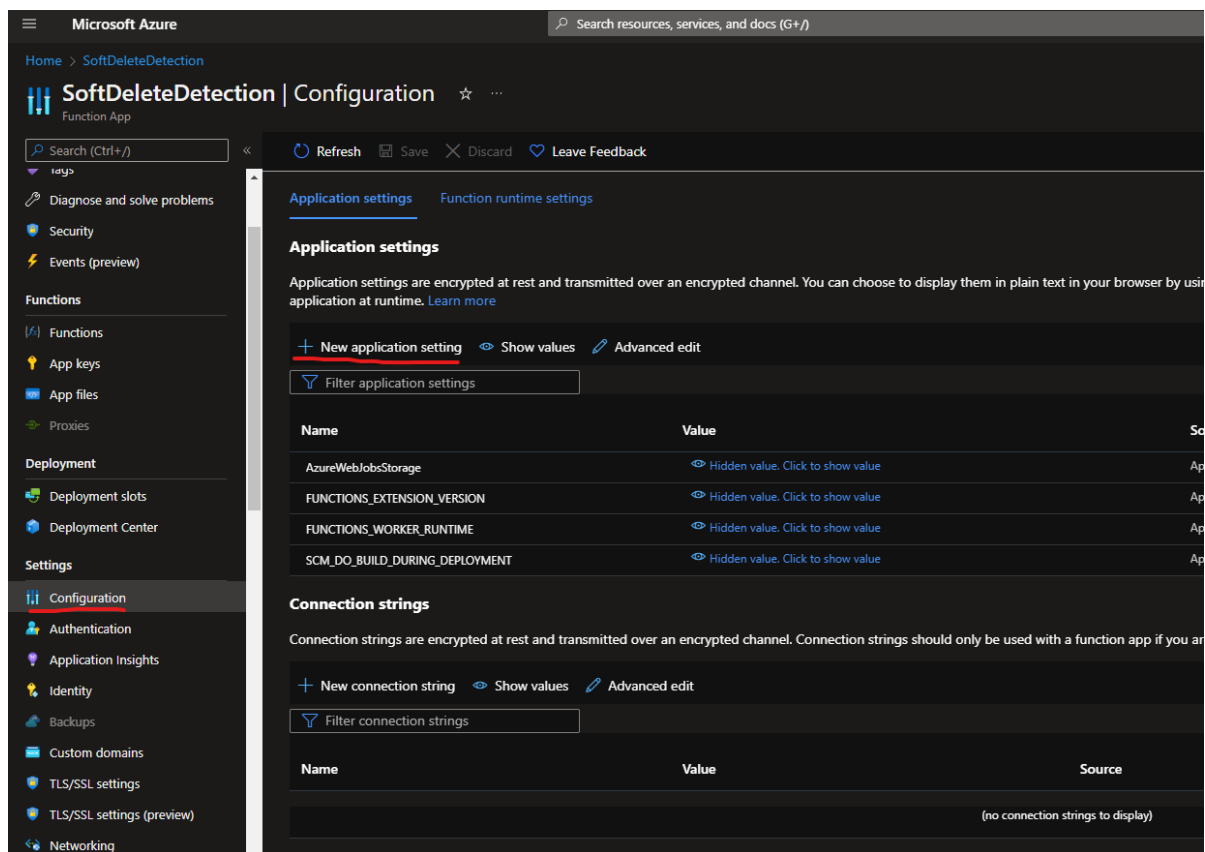
Hit Finish and Close.

Now we have a publish profile. Hit Publish and Visual Studio will deploy the app.



Configure Azure Function

The last thing we need to do is configuring the freshly published Azure Function. This will be done inside the azure portal.



Add the following settings:

Setting	Meaning
ReportReceipents	semicolon separated list of emails
SenderEmail	Email address from which the email should be sent
SenderName	Email Sender name
SmtpServer	smtp server host for email (outgoing)
SmtpPort	smtp server port (outgoing)
EnableSsl	whether to enable ssl or not
SmtpUsername	smtp server username
SmtpPassword	smtp server password
EmailSubject	email subject
LogWorkspaceld	log workspace id (noted before)
TenantId	app registration tenant id (noted before)
ClientID	app registration client id (noted before)
ClientSecret	app registration secret (noted before)