Module 16
Windows Service

Objective

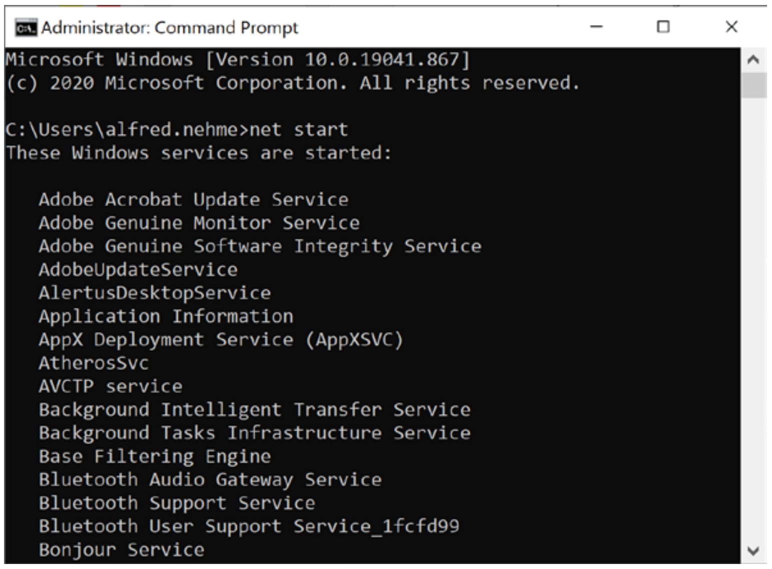
The objective of this module is to introduce you to the concept of a local **service**. Services exist in most operating systems such as Windows and Linux. In this module we will focus on Windows. We will use a service as part of Project 2 to build a “hybrid” cloud application. You should therefore finish this module before starting on Project 2.

Overview

A service is a program that runs in the background. It does not have a UI for a user to interact with and usually runs at all times (that is, from the moment you start your computer and until you shut it down – this is the default case, unless you can configure it to not start at boot time). A service can be configured to start automatically or manually. If configured to start automatically, the OS will start it when it boots.

The OS uses services to do many things in the background without the user interfering. Applications can also employ services to do any desirable operation that does not require user intervention. Let’s look at the services currently running on your Windows computer:

1. Open the Command window and type: `net start`
   This lists the services currently running on your PC.



Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\alfred.nehme>net start
These Windows services are started:

   Adobe Acrobat Update Service
   Adobe Genuine Monitor Service
   Adobe Genuine Software Integrity Service
   AdobeUpdateService
   AlertusDesktopService
   Application Information
   AppX Deployment Service (AppXSVC)
   AtherosSvc
   AVCTP service
   Background Intelligent Transfer Service
   Background Tasks Infrastructure Service
   Base Filtering Engine
   Bluetooth Audio Gateway Service
   Bluetooth Support Service
   Bluetooth User Support Service_1fcfd99
   Bonjour Service
```

They all don't have any UI to interact with and they were all started by the OS (you didn't manually start any of them).

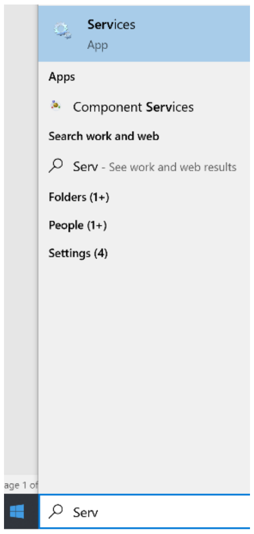
When you look at your computer and see that say 200 processes are currently running, you ask yourself: why 200 when I only started 2 or 3 applications?

The answer is that many of these processes are services running in the background that have no UI and do not need the user input and intervention.

2. Scroll through them. You will be able to tell from their names that some are part of the OS while others were installed by applications. Most of the time, the service name gives you some clue to what it does. For example:

   a. Microsoft Update Health Service: Likely has to do with Windows update. This is a service likely installed by the OS.
   b. Microsoft Defender Antivirus Service: The Microsoft Defender antivirus program. This is a service that also likely comes with the OS.
   c. Adobe Acrobat Update Service: Likely has to do with Adobe version update. This is a service installed by an application (Adobe Acrobat).

3. An alternative way to view services is to use the Services app. Press the Windows keyboard key and start typing `Services`. The Services app should come into view. Select it to open it.



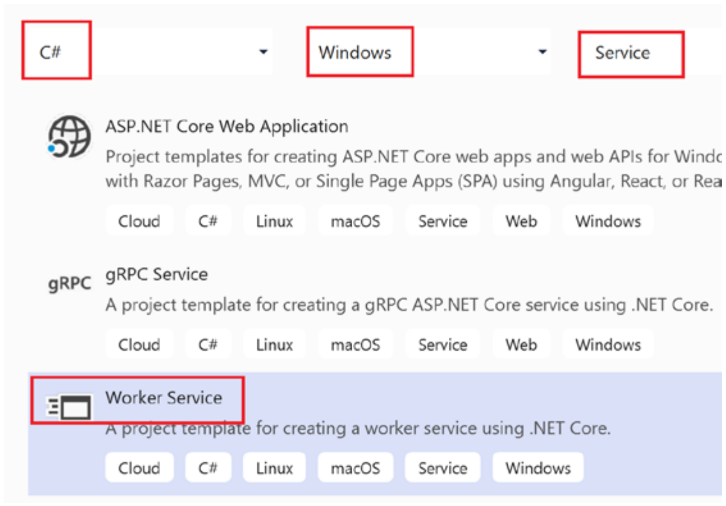
Services are ideal whenever your application needs to do something in the background without user interference. For example, assume you have an application that needs to check when a Bluetooth device is in range of your computer. You obviously want to do this in the background with no need for UI. A service is ideal for this and many other scenarios. In Project 2, you will use a Windows service to listen to messages that arrive to an AWS queue and act on these messages. The service and queue

together act as a glue to link the cloud with local resources (a “hybrid” architecture).
In this module, we will create a simple service just to learn how to create one and install it. In Project 2 we add meaningful code to make it do something interesting.

Create a Service

1. Open Visual Studio and create a blank solution. Name it `WinService`.

2. Add to the `WinService` solution a project using the project template **Worker Service**. Name the project `CS455SystemCheckService`.



C# | Windows | Service

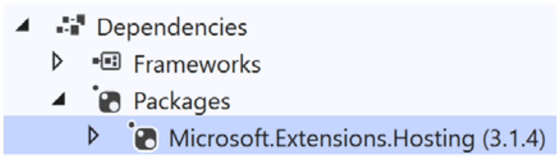
ASP.NET Core Web Application
Project templates for creating ASP.NET Core web apps and web APIs for Windc
with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or Rea
Cloud C# Linux macOS Service Web Windows

gRPC Service
A project template for creating a gRPC ASP.NET Core service using .NET Core.
Cloud C# Linux macOS Service Web Windows

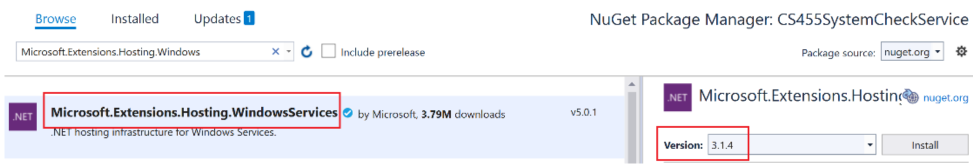
Worker Service
A project template for creating a worker service using .NET Core.
Cloud C# Linux macOS Service Windows

With this template you can deploy the resulting program as a Windows service or a Linux daemon (Linux daemons are the equivalent of Windows services). Daemon Wiki is found here. Read the first paragraph in it

3. Expand Dependencies – Packages and notice that the project template added a package named Microsoft.Extensions.Hosting (3.1.4)   (Note: you might have a different version)



You need to add another package named Microsoft.Extensions.Hosting.WindowsServices (3.1.4).


Browse
Installed
Updates 1
NuGet Package Manager: CS455SystemCheckService
Include prerelease
Package source: nuget.org
Install

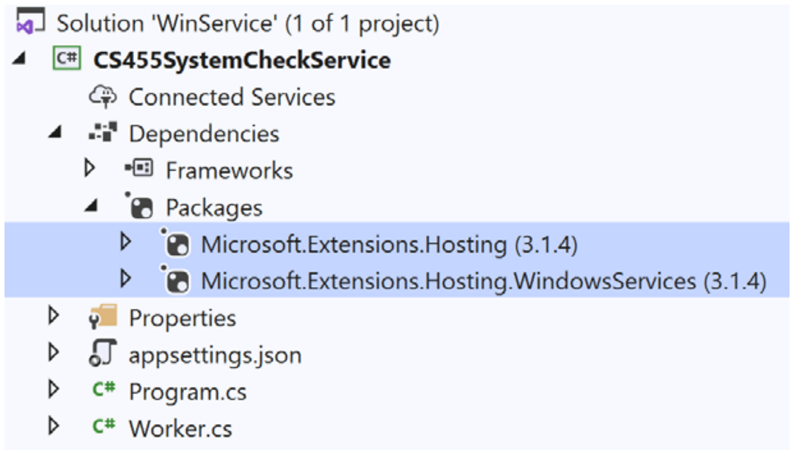
Microsoft.Extensions.Hosting.Windows

Microsoft.Extensions.Hosting.WindowsServices by Microsoft, 3.79M downloads v5.0.1
.NET hosting infrastructure for Windows Services.

Microsoft.Extensions.Hosting nuget.org

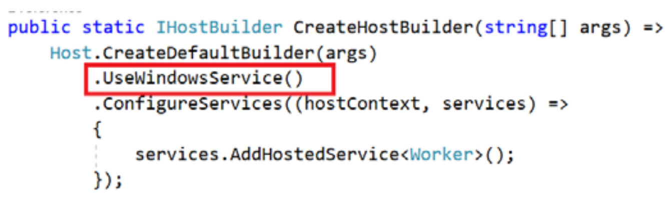
Version: 3.1.4

Your project should now look like this:



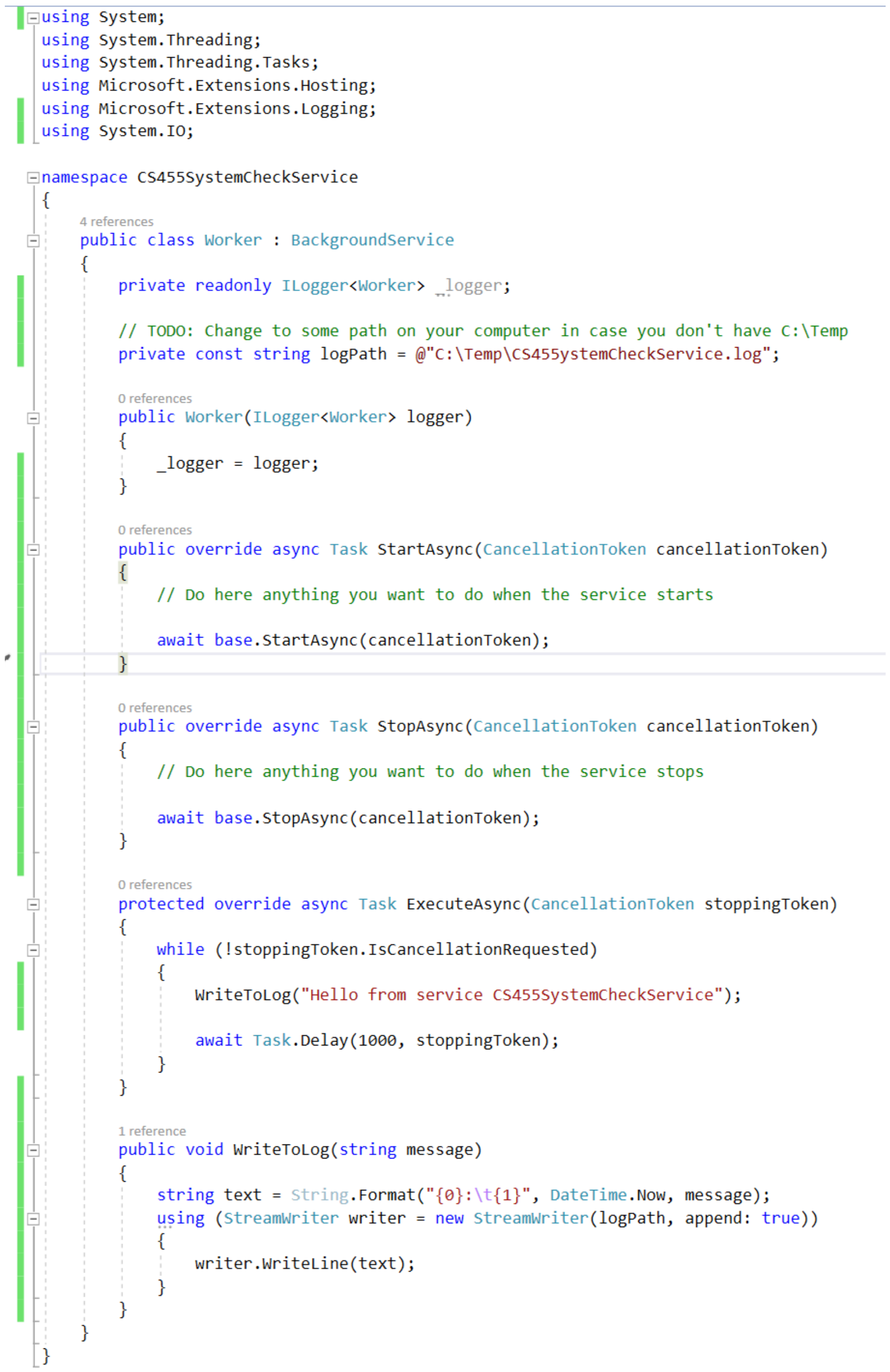
```
Solution 'WinService' (1 of 1 project)
◢ CS455SystemCheckService
    Connected Services
  ◢ Dependencies
    ▷ Frameworks
    ◢ Packages
      ▷ Microsoft.Extensions.Hosting (3.1.4)
      ▷ Microsoft.Extensions.Hosting.WindowsServices (3.1.4)
  ▷ Properties
  ▷ appsettings.json
  ▷ Program.cs
  ▷ Worker.cs
```

4. Open file `Program.cs` and change method `CreateHostBuilder` as shown below:



```
public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .UseWindowsService()
        .ConfigureServices((hostContext, services) =>
        {
            services.AddHostedService<Worker>();
        });
```

5. Open file Worker.cs and change it as shown below:



```
using System;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System.IO;

namespace CS455SystemCheckService
{
    4 references
    public class Worker : BackgroundService
    {
        private readonly ILogger<Worker> _logger;

        // TODO: Change to some path on your computer in case you don't have C:\Temp
        private const string logPath = @"C:\Temp\CS455ystemCheckService.log";

        0 references
        public Worker(ILogger<Worker> logger)
        {
            _logger = logger;
        }

        0 references
        public override async Task StartAsync(CancellationToken cancellationToken)
        {
            // Do here anything you want to do when the service starts

            await base.StartAsync(cancellationToken);
        }

        0 references
        public override async Task StopAsync(CancellationToken cancellationToken)
        {
            // Do here anything you want to do when the service stops

            await base.StopAsync(cancellationToken);
        }

        0 references
        protected override async Task ExecuteAsync(CancellationToken stoppingToken)
        {
            while (!stoppingToken.IsCancellationRequested)
            {
                WriteToLog("Hello from service CS455SystemCheckService");

                await Task.Delay(1000, stoppingToken);
            }
        }

        1 reference
        public void WriteToLog(string message)
        {
            string text = String.Format("{0}:\t{1}", DateTime.Now, message);
            using (StreamWriter writer = new StreamWriter(logPath, append: true))
            {
                writer.WriteLine(text);
            }
        }
    }
}
```

6. Save all files (menu File → Save All).
7. Build the project (Build → Rebuild Solution).

8. You now need to install the service. Open the command prompt and type

   ```
   sc.exe create CS455SystemCheckService start=auto
   binpath=path_to_your_service_exe
   ```

   (You need to change the orange part above).

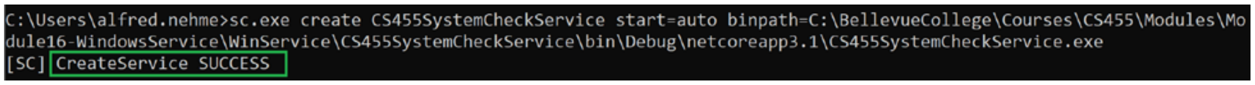
   For example, in my case my service executable path is:

   C:\BellevueCollege\Courses\CS455\Modules\Module16-WindowsService\WinService\CS455SystemCheckService\bin\Debug\netcoreapp3.1\ CS455SystemCheckService.exe

   And my command is:

   ```
   sc.exe create CS455SystemCheckService start=auto binpath=
   C:\BellevueCollege\Courses\CS455\Modules\Module16-
   WindowsService\WinService\CS455SystemCheckService\bin\Debug\netcoreapp3
   .1\ CS455SystemCheckService.exe
   ```

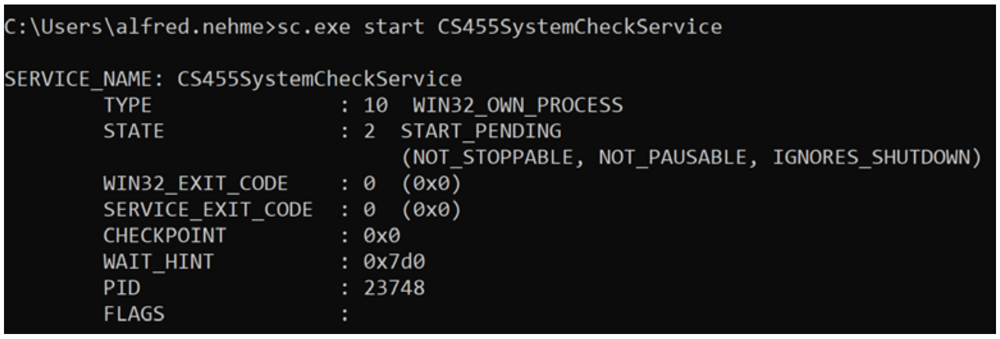
   If you see SUCCESS, then the service was correctly installed:

   ```
   C:\Users\alfred.nehme>sc.exe create CS455SystemCheckService start=auto binpath=C:\BellevueCollege\Courses\CS455\Modules\Mo
   dule16-WindowsService\WinService\CS455SystemCheckService\bin\Debug\netcoreapp3.1\CS455SystemCheckService.exe
   [SC] CreateService SUCCESS
   ```

9. Now that you installed the service, you need to start it. For this, use the command:

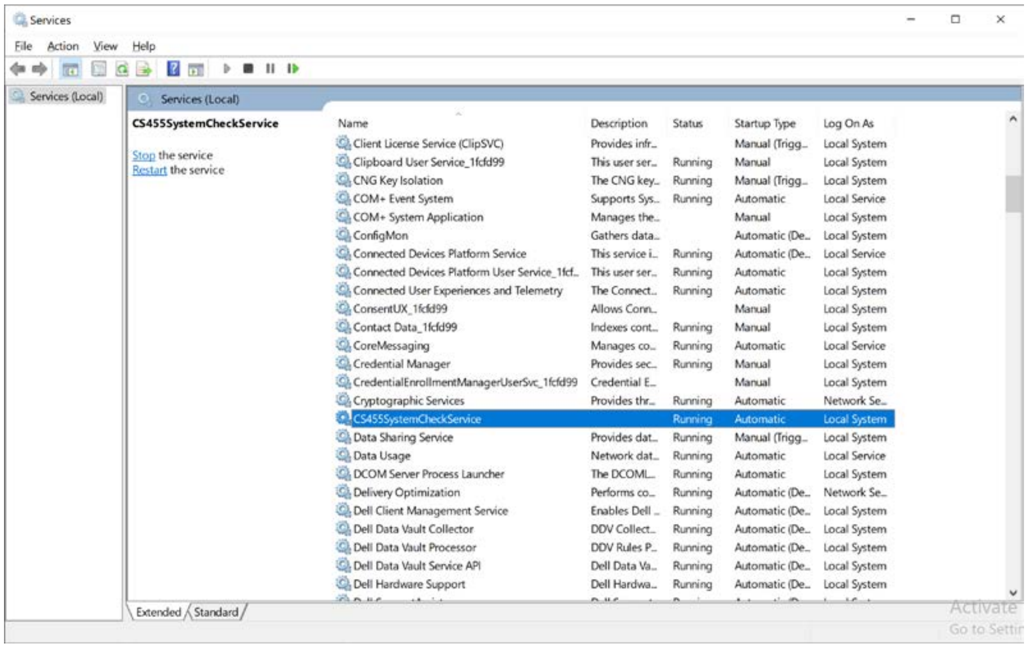
   ```
   sc.exe start CS455SystemCheckService
   ```

   ```
   C:\Users\alfred.nehme>sc.exe start CS455SystemCheckService

   SERVICE_NAME: CS455SystemCheckService
           TYPE               : 10  WIN32_OWN_PROCESS
           STATE              : 2  START_PENDING
                                   (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
           WIN32_EXIT_CODE    : 0  (0x0)
           SERVICE_EXIT_CODE  : 0  (0x0)
           CHECKPOINT         : 0x0
           WAIT_HINT          : 0x7d0
           PID                : 23748
           FLAGS              :
   ```

10. The service is now running in the background. Open the Services app (press the Windows keyboard button and start typing Services). You should see the service and its status should be "Running".



CS455SystemCheckService

Stop the service
Restart the service

Name	Description	Status	Startup Type	Log On As
Client License Service (ClipSVC)	Provides infr...		Manual (Trigg...	Local System
Clipboard User Service_1fcfd99	This user ser...	Running	Manual	Local System
CNG Key Isolation	The CNG key...	Running	Manual (Trigg...	Local System
COM+ Event System	Supports Sys...	Running	Automatic	Local Service
COM+ System Application	Manages the...		Manual	Local System
ConfigMon	Gathers data...		Automatic (De...	Local System
Connected Devices Platform Service	This service i...	Running	Automatic (De...	Local Service
Connected Devices Platform User Service_1fcf...	This user ser...	Running	Automatic	Local System
Connected User Experiences and Telemetry	The Connect...	Running	Automatic	Local System
ConsentUX_1fcfd99	Allows Conn...		Manual	Local System
Contact Data_1fcfd99	Indexes cont...	Running	Manual	Local System
CoreMessaging	Manages co...	Running	Automatic	Local Service
Credential Manager	Provides sec...	Running	Manual	Local System
CredentialEnrollmentManagerUserSvc_1fcfd99	Credential E...		Manual	Local System
Cryptographic Services	Provides thr...	Running	Automatic	Network Se...
CS455SystemCheckService		Running	Automatic	Local System
Data Sharing Service	Provides dat...	Running	Manual (Trigg...	Local System
Data Usage	Network dat...	Running	Automatic	Local Service
DCOM Server Process Launcher	The DCOML...	Running	Automatic	Local System
Delivery Optimization	Performs co...	Running	Automatic (De...	Network Se...
Dell Client Management Service	Enables Dell ...	Running	Automatic (De...	Local System
Dell Data Vault Collector	DDV Collect...	Running	Automatic (De...	Local System
Dell Data Vault Processor	DDV Rules P...	Running	Automatic (De...	Local System
Dell Data Vault Service API	Dell Data Va...	Running	Automatic (De...	Local System
Dell Hardware Support	Dell Hardwa...	Running	Automatic (De...	Local System

11. You should also see it listed if you run the command

```
net start
```

12. Now go to C:\Temp (or wherever you decided to save your log file) and open file CS455ystemCheckService.log.
You should see that the service is writing a message to it every second.

13. If you want to stop and delete the service, run these two commands:

```
sc.exe stop CS455SystemCheckService
sc.exe delete CS455SystemCheckService
```

Now if you go to the Services app and refresh the list (click F5 or menu Action → Refresh) you should see that `CS455SystemCheckService` disappeared.

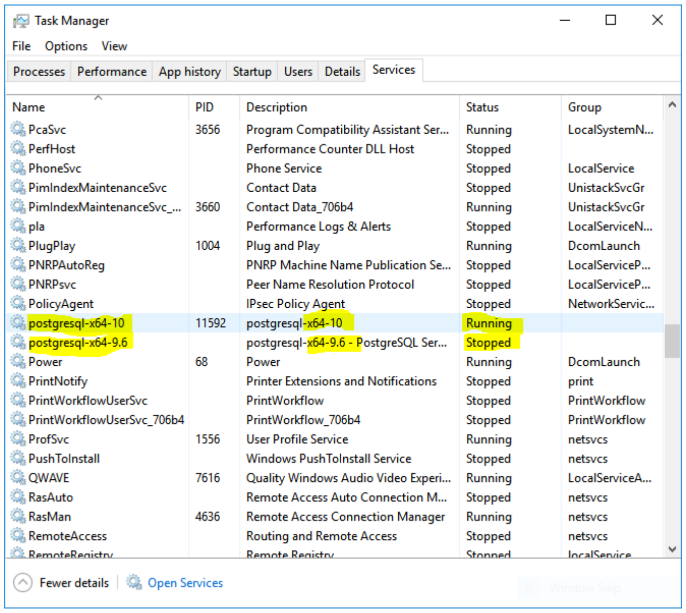
14. Note that we had to manually start the service using command `sc.exe start CS455SystemCheckService`. Do we have to keep doing that every time the computer starts? The answer is NO. The reason is because when we installed the service we specified `start=auto` (look at step 8). This flag tells the OS to start the service when it boots. Do this experiment.

    a. Stop the service but do not delete it. That is, run this command only:

    ```
    sc.exe stop CS455SystemCheckService
    ```

b. Go to the Services app, refresh it, and observe that the status of the service is stopped.
c. Now reboot your PC.
d. After the computer starts again, open the Services app. The status of `CS455SystemCheckService` should be Running. This means the OS started it for you.

15. Note that many applications you are familiar with (e.g., databases) are installed as a service. For example, if you install a database on your computer (e.g., PostgreSQL, etc.), the database will be operational when you start your computer and ready to listen to connections. You don't have to manually start it. The OS does it because when it was installed, the installer specified start=auto. For example, the snapshot below shows PostgreSQL installed as a service.


Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
Open Services

Name	PID	Description	Status	Group
PcaSvc	3656	Program Compatibility Assistant Ser...	Running	LocalSystemN...
PerfHost		Performance Counter DLL Host	Stopped	
PhoneSvc		Phone Service	Stopped	LocalService
PimIndexMaintenanceSvc		Contact Data	Stopped	UnistackSvcGr
PimIndexMaintenanceSvc_...	3660	Contact Data_706b4	Running	UnistackSvcGr
pla		Performance Logs & Alerts	Stopped	LocalServiceN...
PlugPlay	1004	Plug and Play	Running	DcomLaunch
PNRPAutoReg		PNRP Machine Name Publication Se...	Stopped	LocalServiceP...
PNRPsvc		Peer Name Resolution Protocol	Stopped	LocalServiceP...
PolicyAgent		IPsec Policy Agent	Stopped	NetworkServic...
postgresql-x64-10	11592	postgresql-x64-10	Running	
postgresql-x64-9.6		postgresql-x64-9.6 - PostgreSQL Ser...	Stopped	
Power	68	Power	Running	DcomLaunch
PrintNotify		Printer Extensions and Notifications	Stopped	print
PrintWorkflowUserSvc		PrintWorkflow	Stopped	PrintWorkflow
PrintWorkflowUserSvc_706b4		PrintWorkflow_706b4	Stopped	PrintWorkflow
ProfSvc	1556	User Profile Service	Running	netsvcs
PushToInstall		Windows PushToInstall Service	Stopped	netsvcs
QWAVE	7616	Quality Windows Audio Video Experi...	Running	LocalServiceA...
RasAuto		Remote Access Auto Connection M...	Stopped	netsvcs
RasMan	4636	Remote Access Connection Manager	Running	netsvcs
RemoteAccess		Routing and Remote Access	Stopped	netsvcs
RemoteRegistry		Remote Registry	Stopped	localService

When done with this module, delete the service (otherwise, it will keep writing to the log file forever)

Summary

What you should have learned from this module.

- You learned what a Windows service (or Linux daemon) is.
- You know how to create, install, start, stop, and delete a service.
- You observed that a service runs in the background without any user intervention.
- You learned that in a larger application, some components need to be running in the background at all times without any user intervention. Designing such applications to run as a service in Windows (or daemon in Linux) is one solution.

I Project 2, one component will be installed as a service. Use this module as a guide when creating it.

What to Submit:

There is nothing to submit for this module.