

- 1) Log into the Azure portal (<https://portal.azure.com>) and select 'Resource Groups' from the side bar. This will show you all resource groups that are currently active. Press the 'Add' button in the upper left corner to create a new resource group.

Enter a descriptive name for your resource group, select 'West Europe' as the location and create the resource group.

➔ All Azure resources have to reside in a resource group. A resource group acts as a logical container for your cloud resources and should contain resources with the same lifecycle. As an example, if you have web app A and web app B that both access the same database, you would have a resource group for A and B each and a resource group for the database as it has a different lifecycle than the web apps. Deleting a resource group will destroy all resources that are contained within.

- 2) Select 'App Services' from the side bar. As with Resource Groups you will be shown an overview of all currently existing web applications. Press the 'Add' button in the upper left corner to create a new web app.

Select the resource group that you have created in step 1. Give your application a name (this will be part of the url under which the application can be reached, so it has to be unique). Select 'Code' as Publish Method, .Net Core 2.1 as Runtime stack and 'West Europe' as Location.

Next you have to create an App Service Plan. Select 'Create new' and choose F1 as the pricing tier.

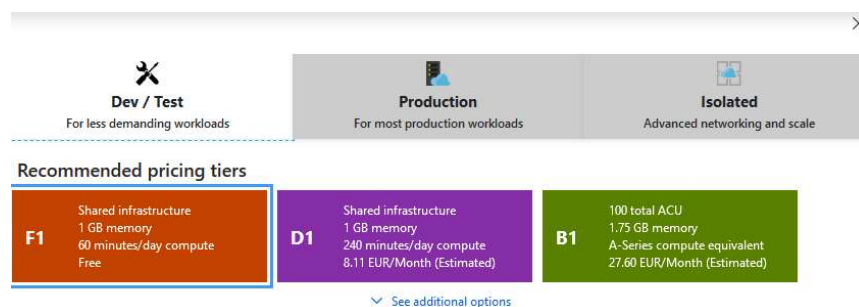


Figure 1

➔ All Web Apps have to run in an App Service Plan. It defines a set of compute resources that your web app will run on. It can be compared to a server farm in conventional hosting. You can have multiple web apps in a service plan and they will share the resources between them.


Your settings now should look like in Figure 2.


Basics Monitoring Tags Review and create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)


**PROJECT DETAILS**

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

\* Subscription  Visual Studio Enterprise

\* Resource Group  mp-cloud-workshop [Create new](#)

**INSTANCE DETAILS**

\* Name mp-cloud-workshop-tst  .azurewebsites.net

\* Publish [Code](#) [Docker Image](#)

\* Runtime stack .NET Core 2.1

\* Operating System [Linux](#) [Windows](#)

\* Location West Europe

**APP SERVICE PLAN**

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

\* Plan (New) mp-cloud-workshop-tst [Create new](#)

\* Sku and size **Free F1**  
Shared infrastructure, 1 GB memory  
[Change size](#)

[Review and create](#) [Next: Monitoring >](#)

Figure 2

Switch to the Monitoring tab and create a new Application Insights resource. We will use this later to monitor the application. You can now create the web app.

- 3) Open the 'Pipelines' tab in Azure DevOps and select 'Releases'. Create a new release pipeline by clicking '+New'. A new dialog, similar to the one in Figure 3 will open.

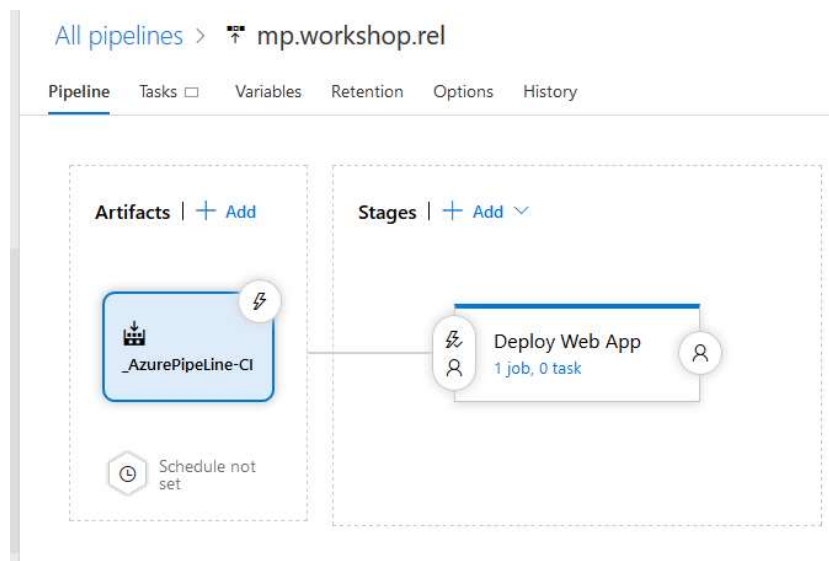


Figure 3

First, we have to tell Azure DevOps which artifacts we want to publish. Click on artifacts and select the build pipeline we have configured. We want to release a new version every time the web application is updated. Click on the lightning bolt in the 'Artifacts' section and enable the Continuous Deployment trigger. This will trigger the release pipeline every time a new build is available.

- 4) Next, we have to do the actual deployment of our web application to Azure. Click on the stage and select 'Azure App Service Deploy'. Select your Azure Subscription and authorize Azure DevOps to access it. Depending on if you have chosen Linux or Windows as Operating System when you created the Azure resources, select 'Web App on Windows' or 'Web App on Linux'. The App Service name has to be the name of the Azure Web App that you have created in Step 2. Save the release pipeline.
- 5) Create a new release by clicking the 'Create a release' button. After the deployment has finished, you can view your application by going to the Azure portal and selecting your Web App. The 'Overview' contains the URL under which your web application can be reached.
- 6) In the next step we will send monitoring data from our application to Application Insights. Open the web application and add the Microsoft.ApplicationInsights.AspNetCore and Microsoft.Extensions.Logging.ApplicationInsights Nuget packages. In the ConfigureServices method of the Startup class add services.AddApplicationInsightsTelemetry. This will add Telemetry Processors for e.g. CPU consumption to the app.

In the CreateWebHostBuilder method of the Program class add the following snippet:

```
WebHost.CreateDefaultBuilder(args)
    .ConfigureLogging((context, logBuilder) =>
    {

        logBuilder.AddApplicationInsights(context.Configuration["ApplicationInsights:InstrumentationKey"]);
    })
    .UseStartup<Startup>();
```

This will ensure that your application traces are also sent to Application Insights. Next the application needs to know to which Application Insights Instance the monitoring information has been sent. This is done by specifying the Instrumentation Key. Navigate to the Azure portal and select the Application Insights resource. In the 'Overview' tab you will find the Instrumentation Key. Copy it and add it to the appsettings.json file in your application.

```
"ApplicationInsights": {
  "InstrumentationKey": "<YourKeyHere>"
},
```

Navigate to the Home Controller and log a warning message.

- 7) Redeploy your app by committing your changes to your Azure DevOps repository.

- 8) Your application should now send telemetry to Application Insights. Open your Application Insights Resource in the Azure portal and select 'Live Metrics Stream' to observe the metrics send by your application in real-time.