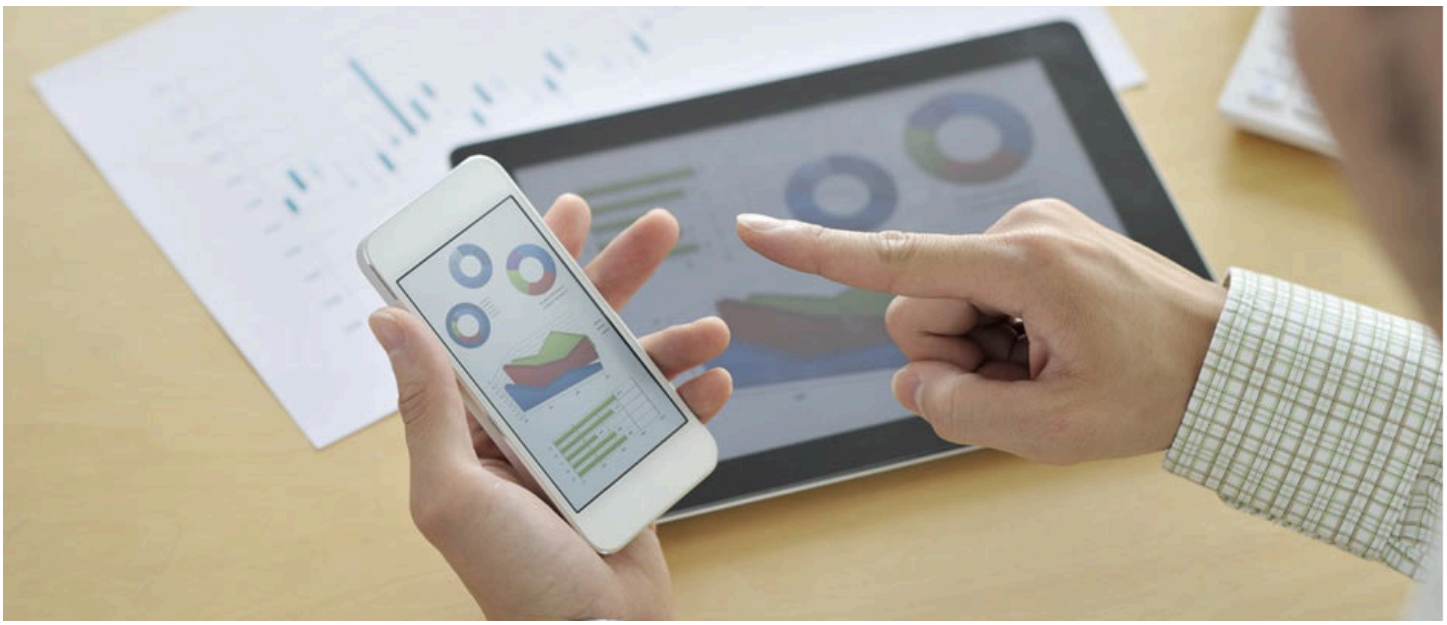
scientiamobile

"The first step in a great mobile experience"



WURFL MICROSERVICE FOR AZURE MARKETPLACE

Support

The ScientiaMobile Enterprise Support Portal is open to all WURFL users, both commercial license holders and evaluation users. It represents the combined knowledge base for the WURFL community. Commercial licensees are invited to post questions in the forum using the account to which their licenses are associated. This may mean faster handling of those posts by ScientiaMobile's personnel.

For commercial license holders, there are tiered support levels to address a variety of business support needs. After logging into your account, commercial licensees with support options can access the Enterprise Support portal to post tickets. These tickets will receive expedited attention.

To inquire about support plans, use our License Inquiry or our General Inquiry form.

Update Notifications

If you would like to be notified of our API updates, major data updates, and other technical changes, please subscribe to our ScientiaMobile Announcements list.

scientiamobile

www.scientiamobile.com

Tel +1.703.310.6650

E-mail: sales@scientiamobile.com

Copyright © 2026 ScientiaMobile, all rights reserved. WURFL Cloud, WURFL OnSite, WURFL and, InFuze WURFL InSight and respective logos are trademarks of ScientiaMobile. Apache is the trademark of the Apache Software Foundation. NGINX is the trademark of Nginx Software Inc. Varnish is the trademark of Varnish Software AB

WURFL Microservice for Azure Marketplace



Using WURFL Microservice for **Azure** (**Basic**, **Standard** or **Professional** Edition) will allow you to run your own WURFL-based device detection service in your hosting infrastructure by deploying familiar **Azure VM** images. If you are not familiar with Azure and the possibilities of the Azure Cloud infrastructure, **here is a good place to get started**.

WURFL Microservice Client API (WM Client)

Given an HTTP Request and a device capability name, the WURFL Client API will return the corresponding property value.

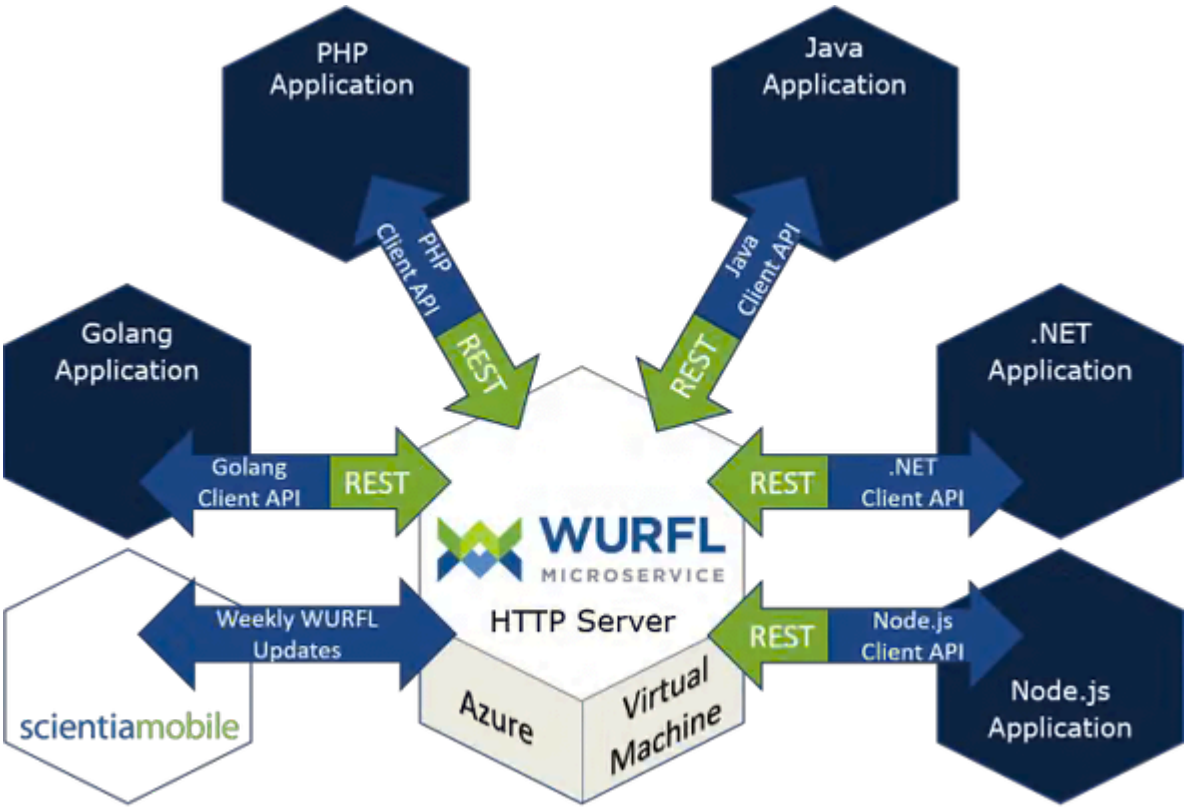
You can integrate the WURFL Microservice Standard, Basic or Pro Edition with major programming languages, including:

- .NET (C#)
- Java
- PHP
- Node.js
- Golang.
- Python
- Scala
- Rust

WURFL Microservice Architecture

WURFL Microservice Server (**WM Server**) is a service that exposes its functions to the WURFL Microservice Client API (**WM Client**). The Client API depends on the availability of the WM server to work.

The following diagram explains the overall architecture of WURFL Microservice as deployed through the Azure Marketplace. The Azure VM will start an instance of the WM Server. In addition to supporting the REST interface for WM Clients, the service will periodically query ScientiaMobile for updates to the WURFL Device Description Repository to include the profiles of newly released devices and updated API logic.



Note 1: *While the WM Client feels like a "standalone API" for most practical purposes, under the hood it requires interaction with the WURFL Microservice HTTP server (running in Azure) which introduces some latency (hugely mitigated by the built-in caching layer in the WM Client itself). For this reason, ScientiaMobile does not refer to the WURFL Microservice Client API (WM Client) as a "WURFL API" (that name is reserved for the* ***WURFL OnSite APIs*** *).*

Note 2: *The WM Server needs to have access to ScientiaMobile servers to obtain data and updates to the API logic. In high-security scenarios, access to outside internet-based services may be blocked. In those cases, your firewall may need to be configured to allow traffic between the WM Server and ScientiaMobile: outbound internet access on TCP port* `443` *and DNS is required.*

Note 3*: communication between the WM Client API and the WM Server happens through a REST API, but this is totally transparent to users of WURFL Microservice and ScientiaMobile acknowledges this aspect of the WURFL Microservice product for sake of transparency. Unless you have a very specific use-case(s), we strongly*

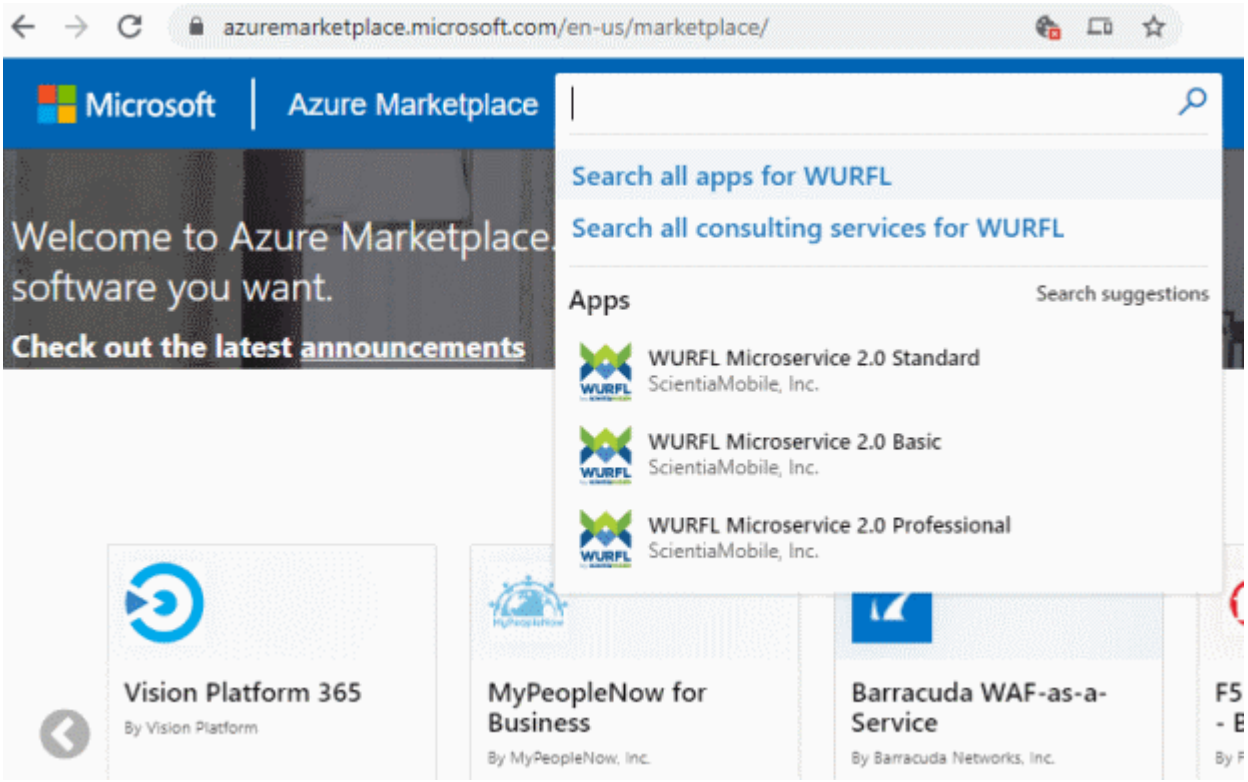
discourage you from utilizing the REST Interface directly. Not only does the WM Client provide a caching layer (which delivers much greater performance), but ScientiaMobile reserves the right to modify the internal REST API or even to switch to different non-REST protocols in future versions of the product without notice.

Deploying the Azure VM and Setting up the WM Client APIs

1. Go to the Azure Marketplace

Searching for WURFL in the searchbox at the top will bring up the three ScientiaMobile listings. Direct links are provided below for your convenience.

Select the right Virtual machine for your needs. The three VMs only vary based on the capability set each one of them supports (Capabilities = Device Properties) :



The three VMs were designed with different use-cases in mind. Choose the WURFL Microservice version that fits your needs. You can pick Basic, Standard or Professional.

Basic - web optimization use-case

WURFL Microservice for GO,JAVA,dotNET,NODE.JS,PHP - Basic Edition

Standard - Analytics use-case

WURFL Microservice for GO,JAVA,dotNET,NODE.JS,PHP - Standard Edition

Professional - Ad Tech / Event Streams use-case

WURFL Microservice for GO,JAVA,dotNET,NODE.JS,PHP - Professional Edition

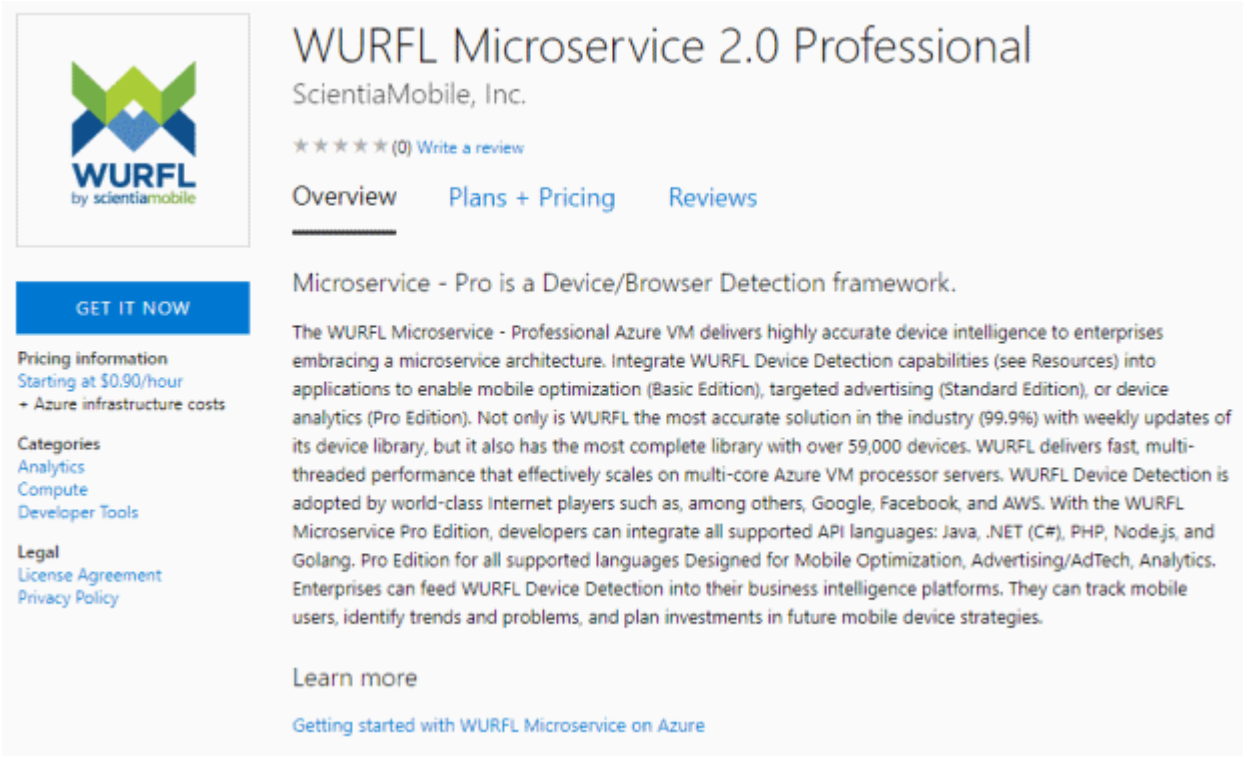
2. Select the Virtual machine instance you want to run your Azure VM on.

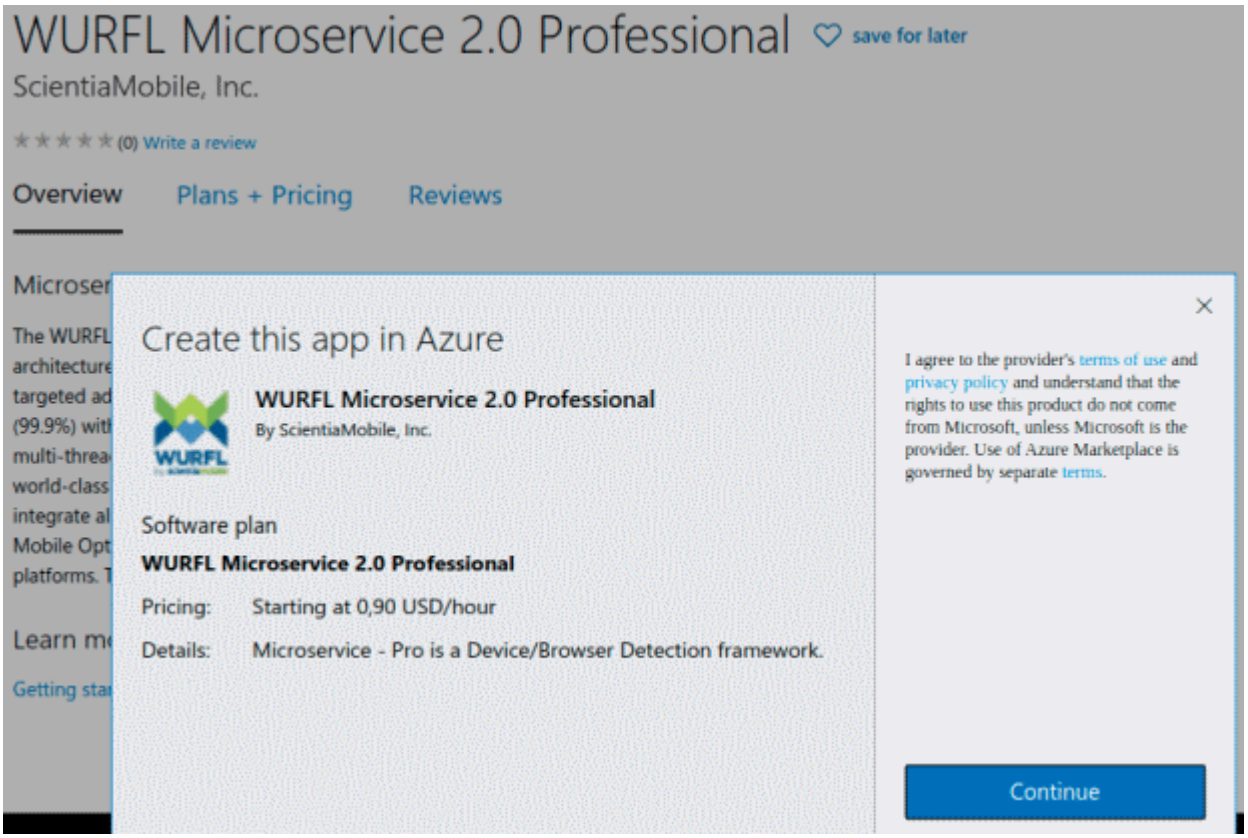
For the purposes of this document, we will pick the Professional edition, but the process is identical for the other versions (AKA "tiers"). Only the number of supported capabilities differs among VMs.

Capability Sets supported for each of the three tiers:

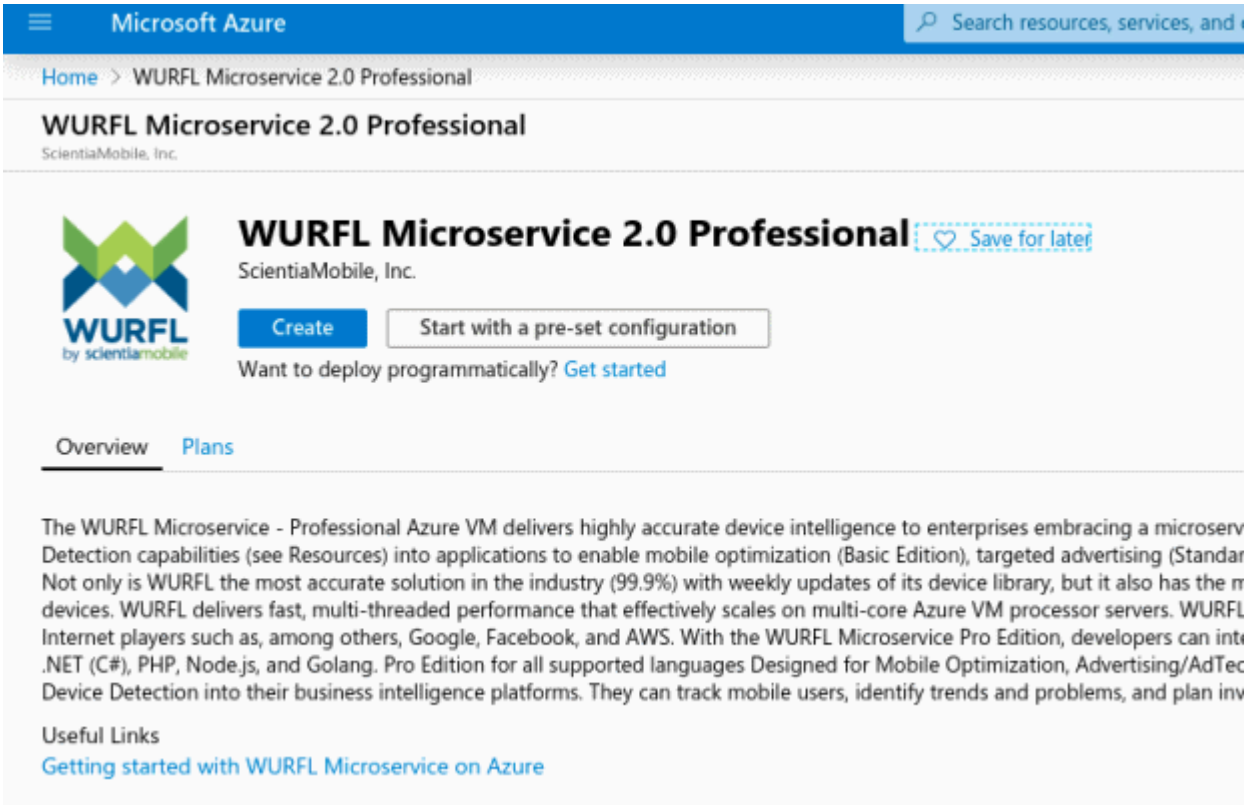
- Basic
- Standard
- Pro

Clicking the "Get it now" button and "Continue" will get the deployment process started.



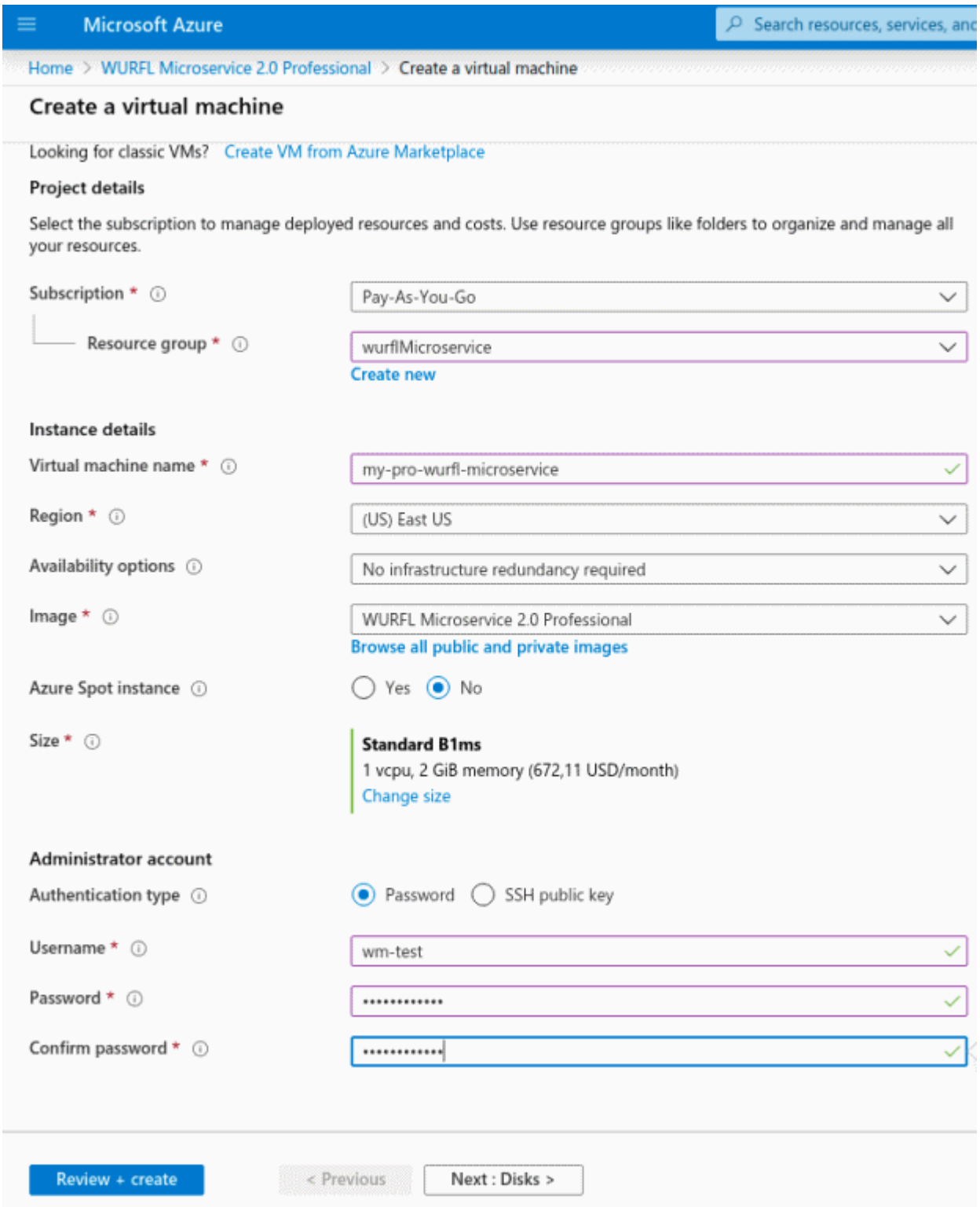


You are now ready to create your first Azure VM instance. In the creation page click on the blue “Create” button.

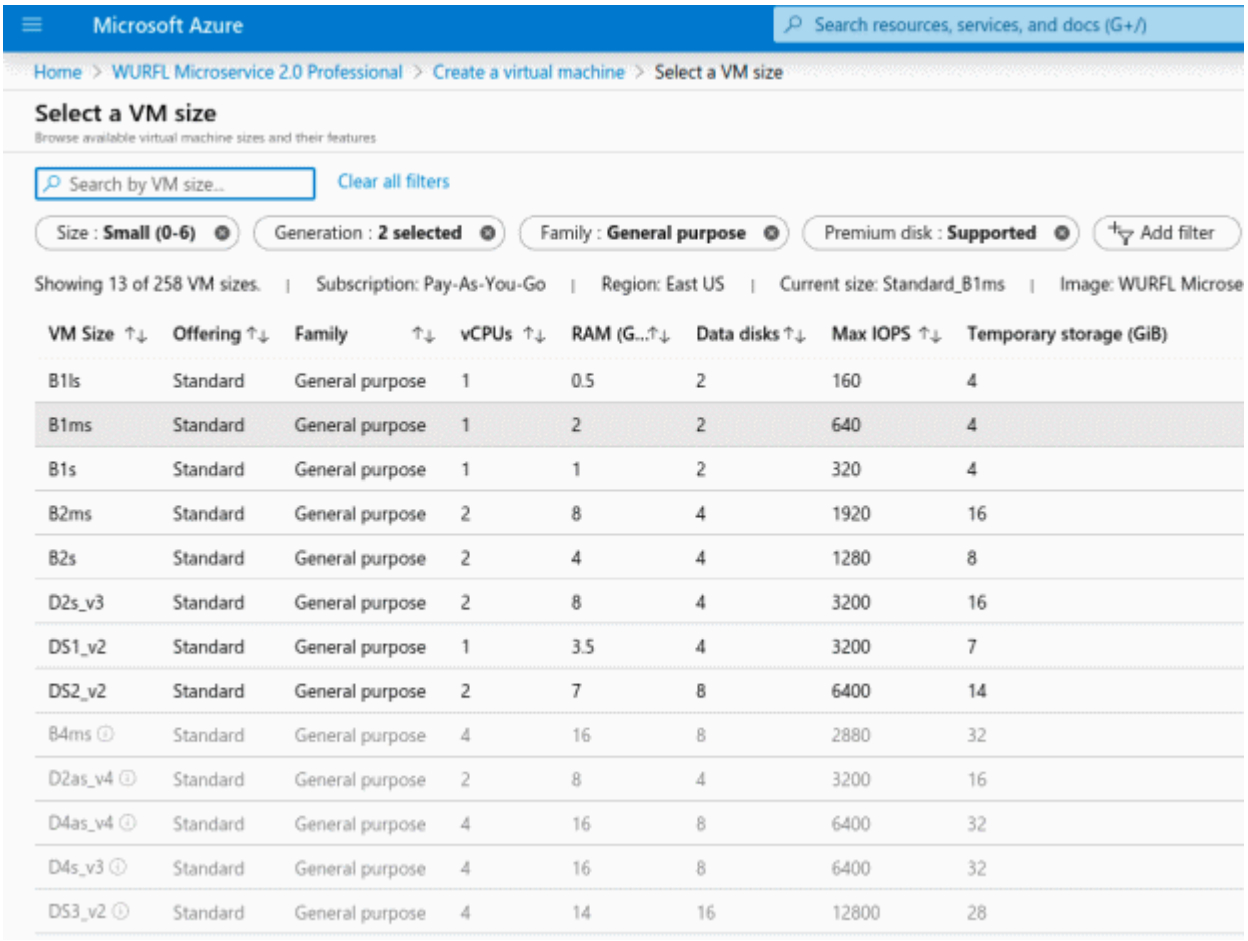


First you'll need to provide some data. Assuming you are familiar with the Azure platform, everything is self-explanatory. Click on the “Change size” link to select an instance type.

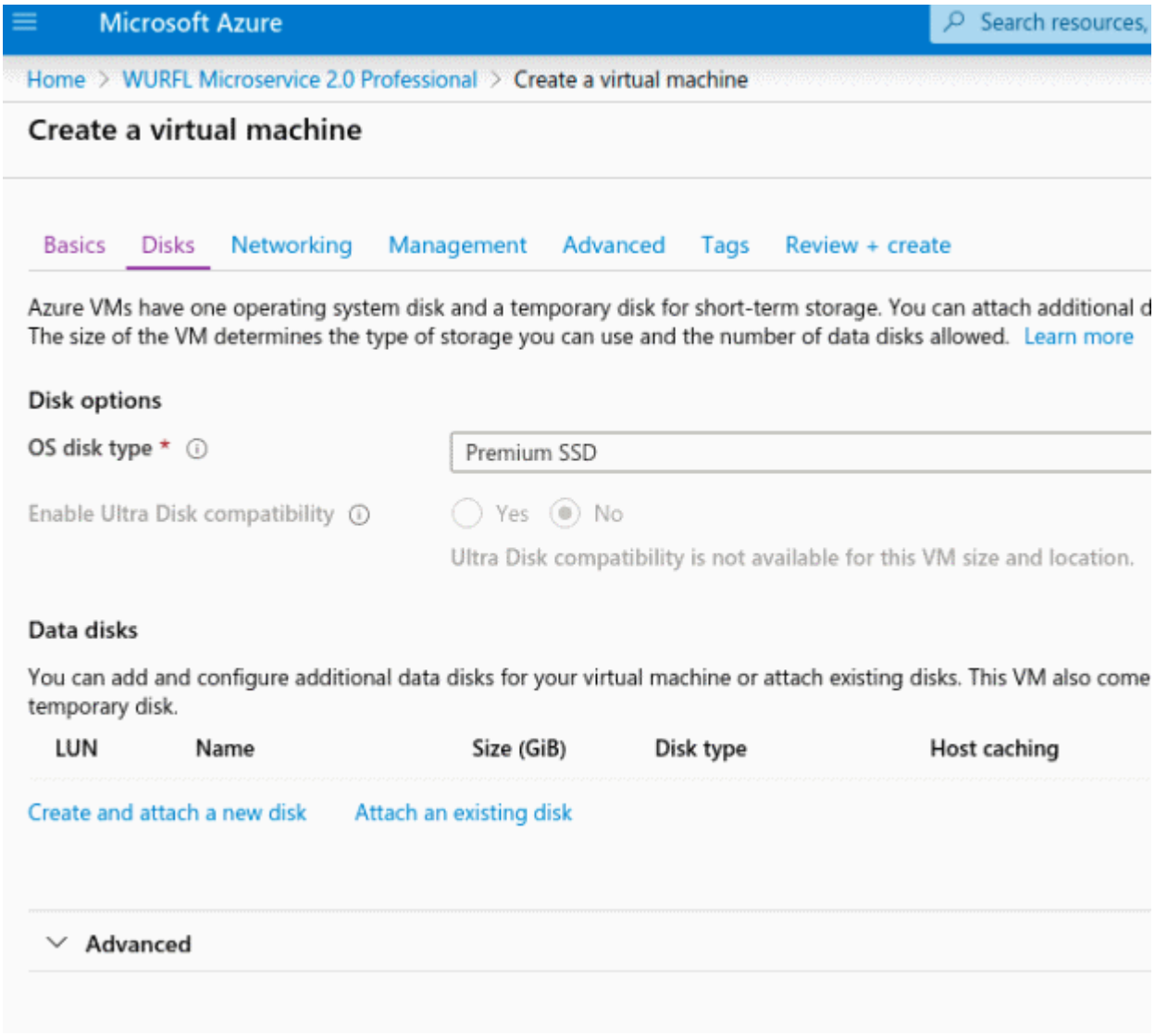
Note: *Which VM to choose depends greatly on how much traffic you need to process through device detection. As a rule of thumb, using* ***Standard_B1ms*** *(single CPU, 2GB Memory) will allow about 1000 lookups/sec. Actual throughput will be much higher for users of any of the WM Client APIs thanks to local caching. While* ***Standard_B1ms*** *is sufficient for development and testing, a larger instance such as* ***Standard_A2_v2*** *or larger may be needed for production depending on a variety of factors.*

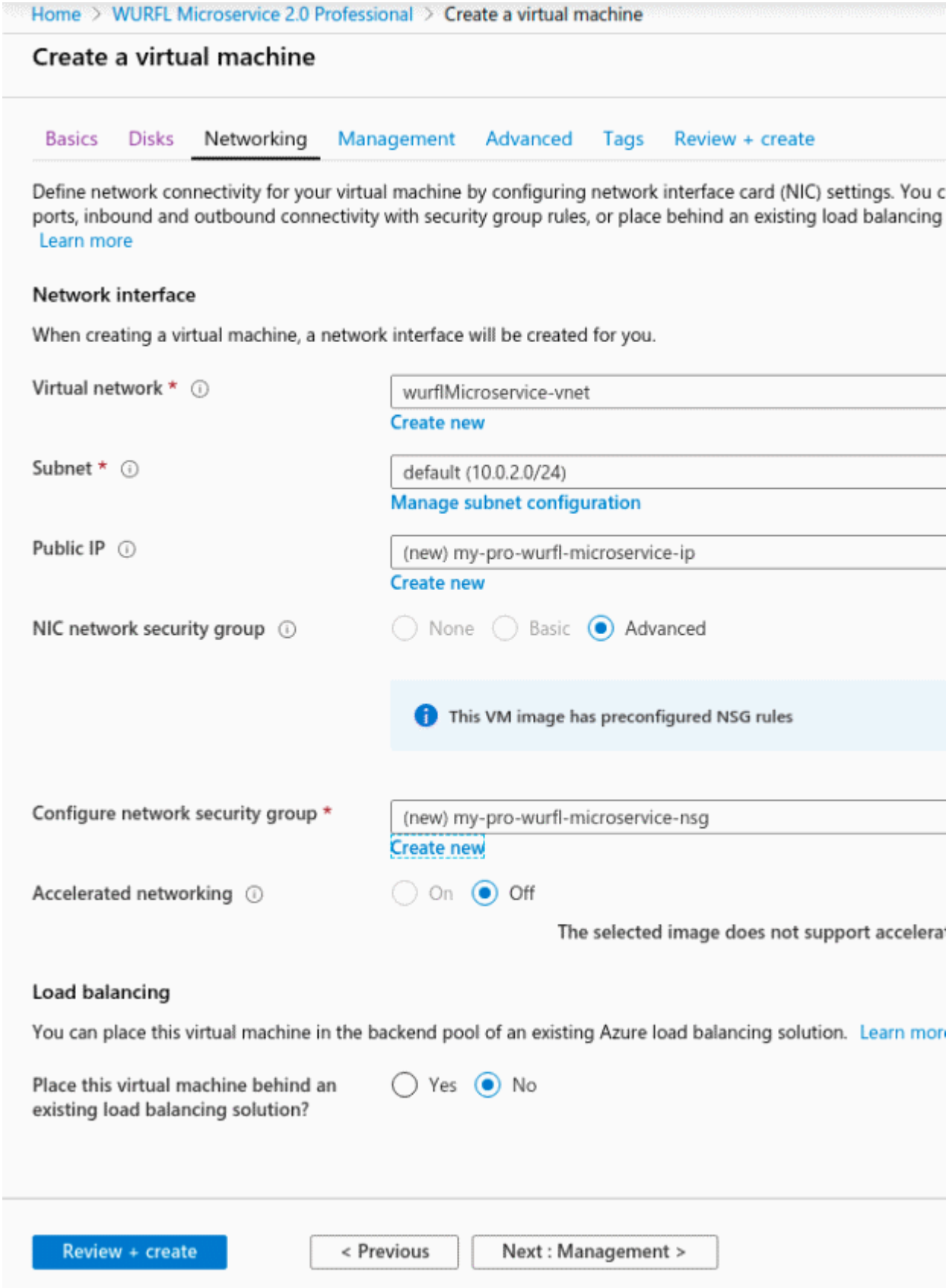


For the purpose of this guide, we selected the **B1ms** type (more information on Azure instance type sizes can be found here).

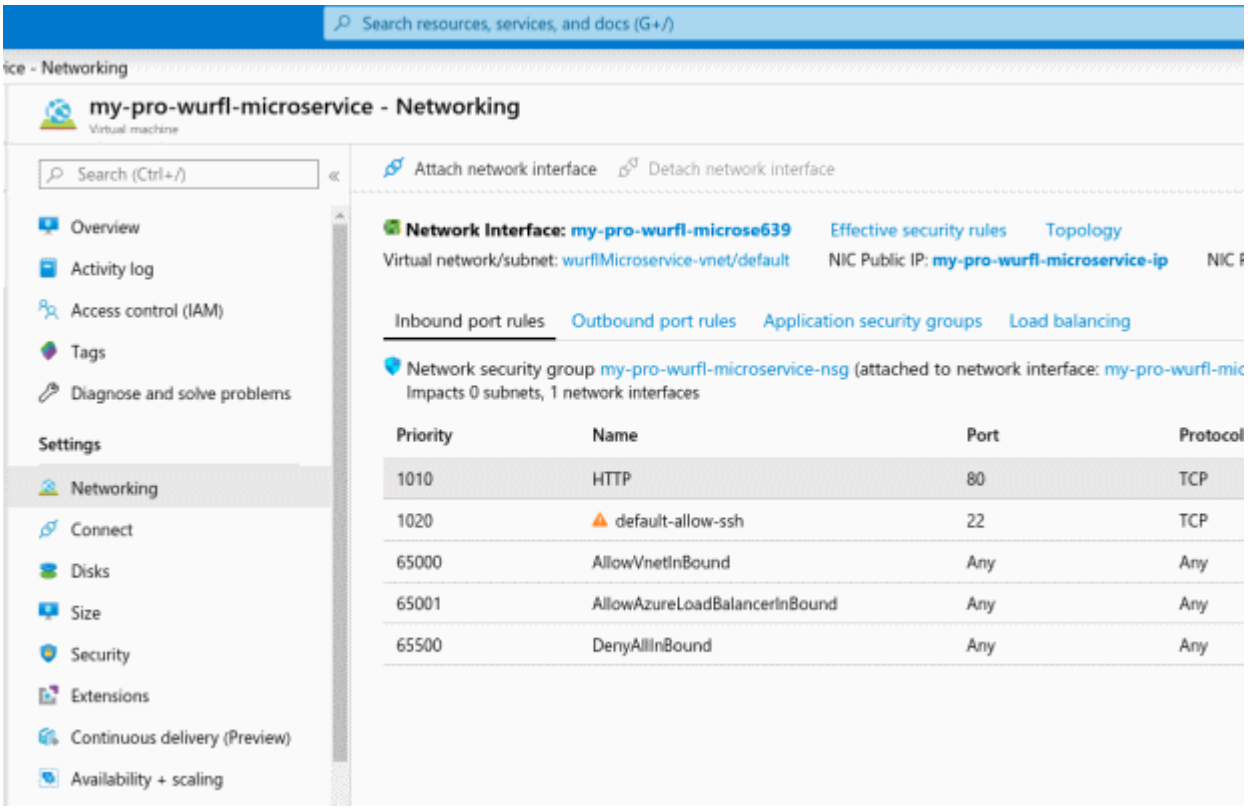


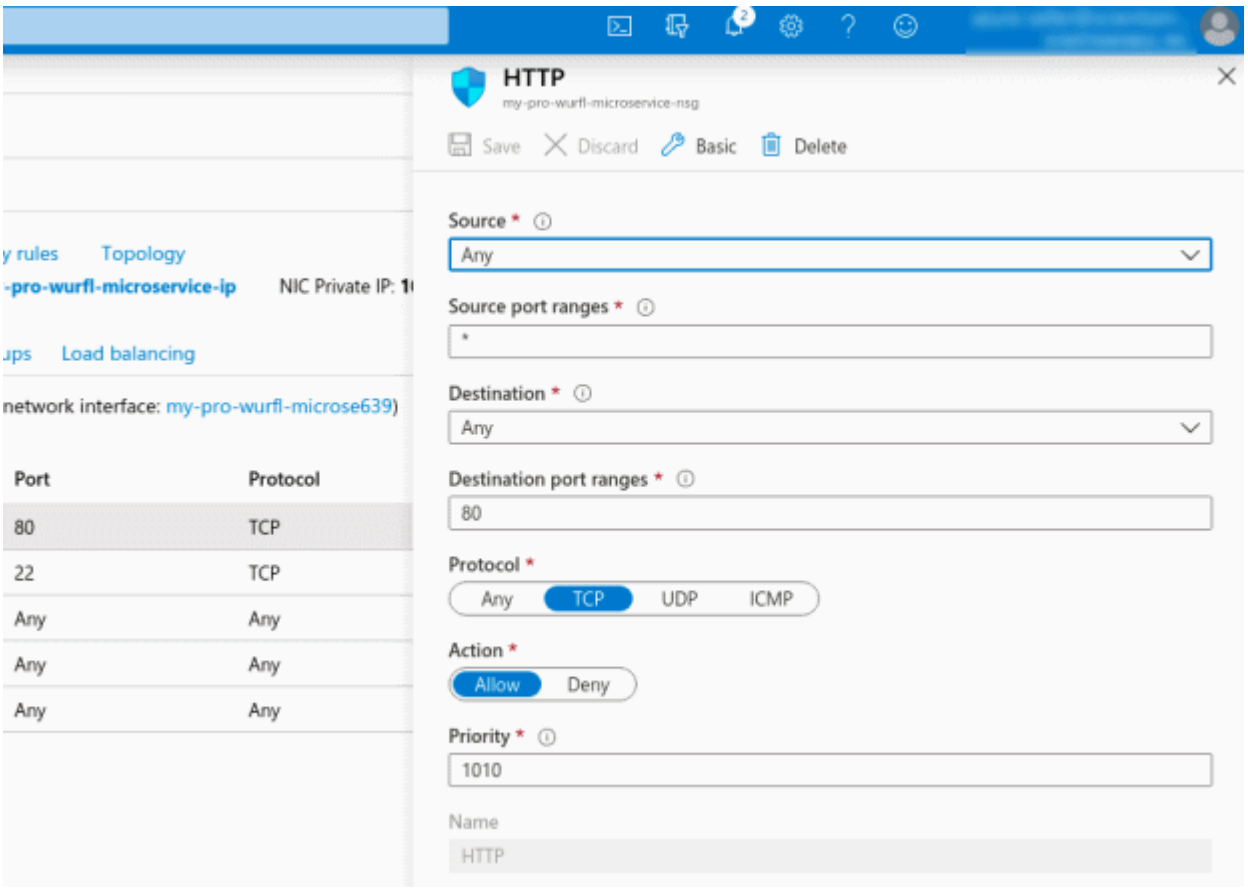
Choose your preferred disk type and size.


Microsoft Azure
Search resources,
Home > WURFL Microservice 2.0 Professional > Create a virtual machine
Create a virtual machine
Basics
Disks
Networking
Management
Advanced
Tags
Review + create
Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional d
The size of the VM determines the type of storage you can use and the number of data disks allowed. Learn more
Disk options
OS disk type *
Premium SSD
Enable Ultra Disk compatibility
Yes
No
Ultra Disk compatibility is not available for this VM size and location.
Data disks
You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also come
temporary disk.
LUN
Name
Size (GiB)
Disk type
Host caching
Create and attach a new disk
Attach an existing disk
Advanced



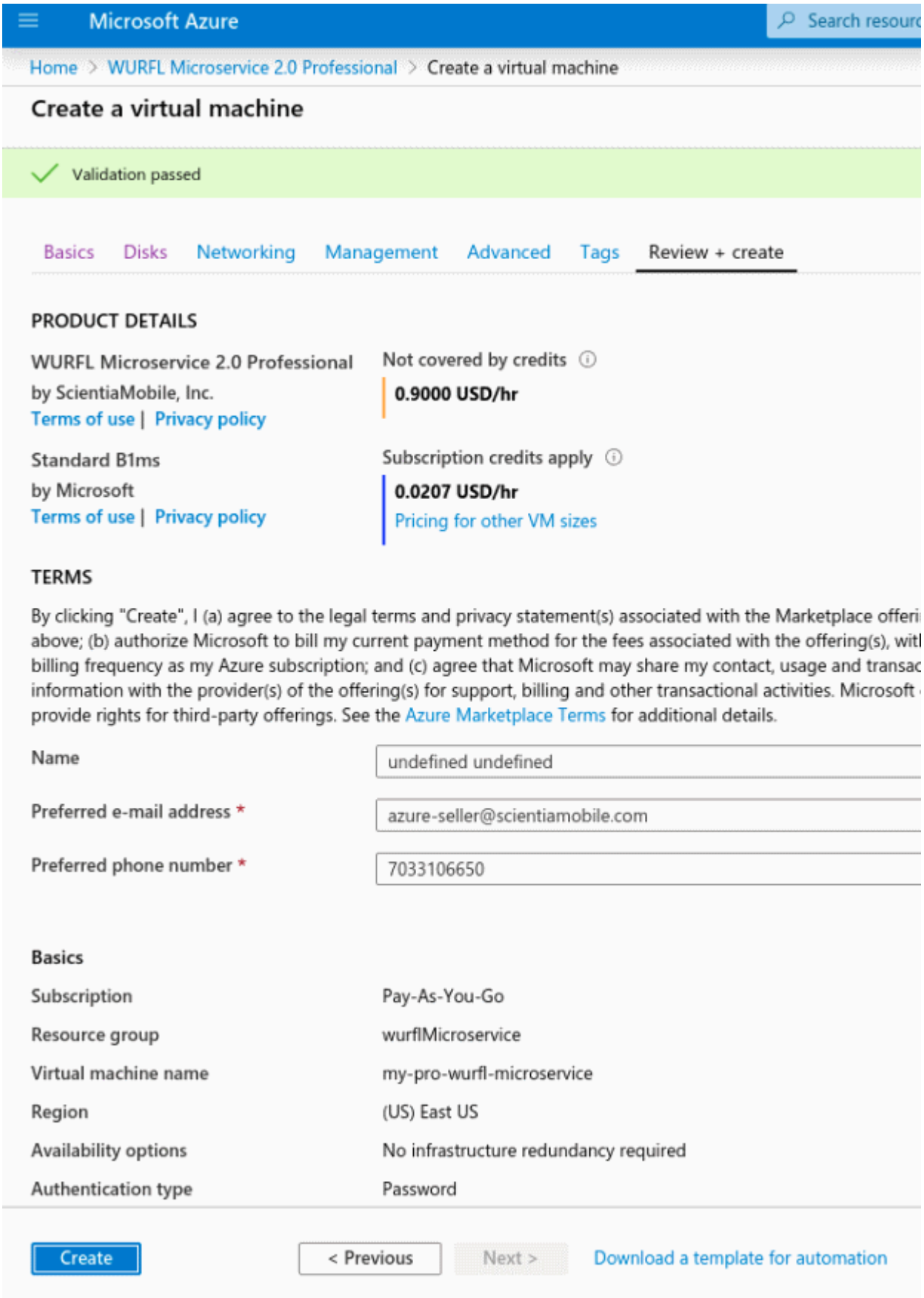
Security group configuration opens ports `80` and `22` by default. Port 80 is used to talk HTTP to the WURFL Microservice HTTP server, while port 22 allows SSH access to the VM instance.





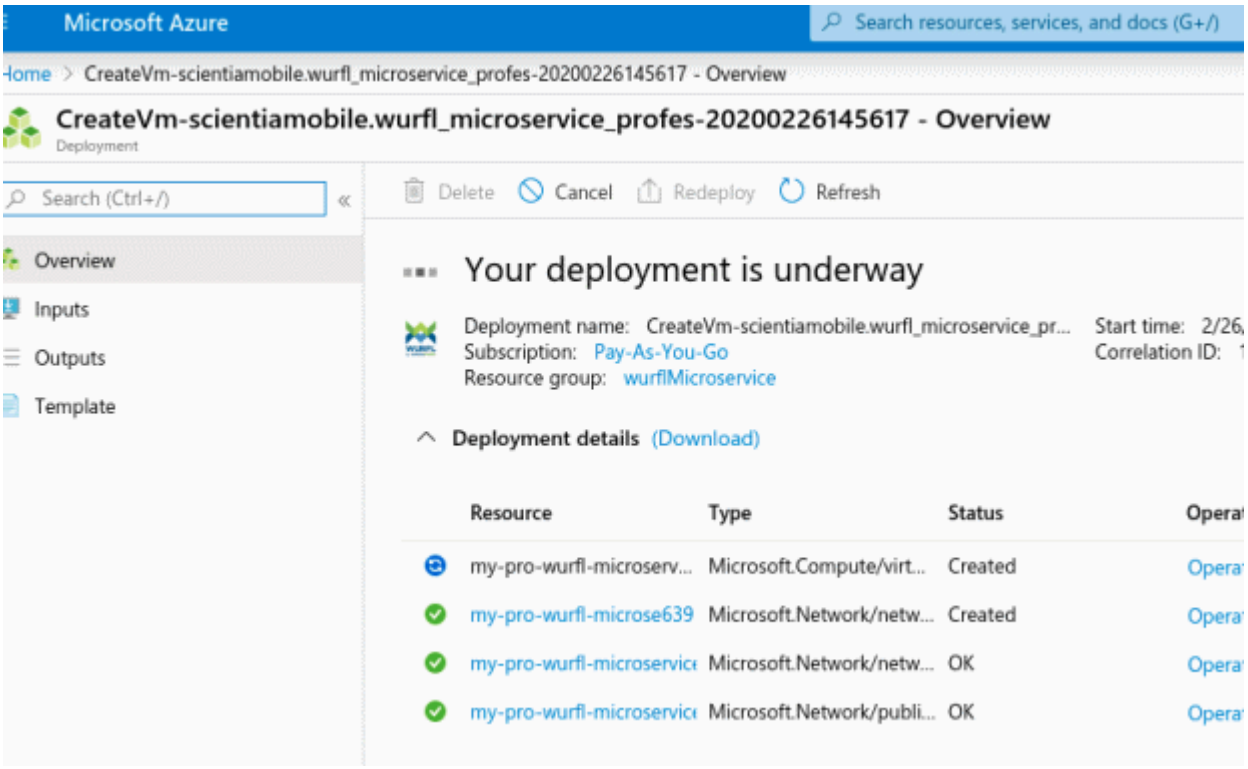
If for some reason, port 80 is not open by default, you will need to add it by clicking on the “Create new” link of “Configure security group” and “Inbound security rules” of Networking menu item (not shown).

It’s now time to review our instance by clicking on the “Review + Create” tab.

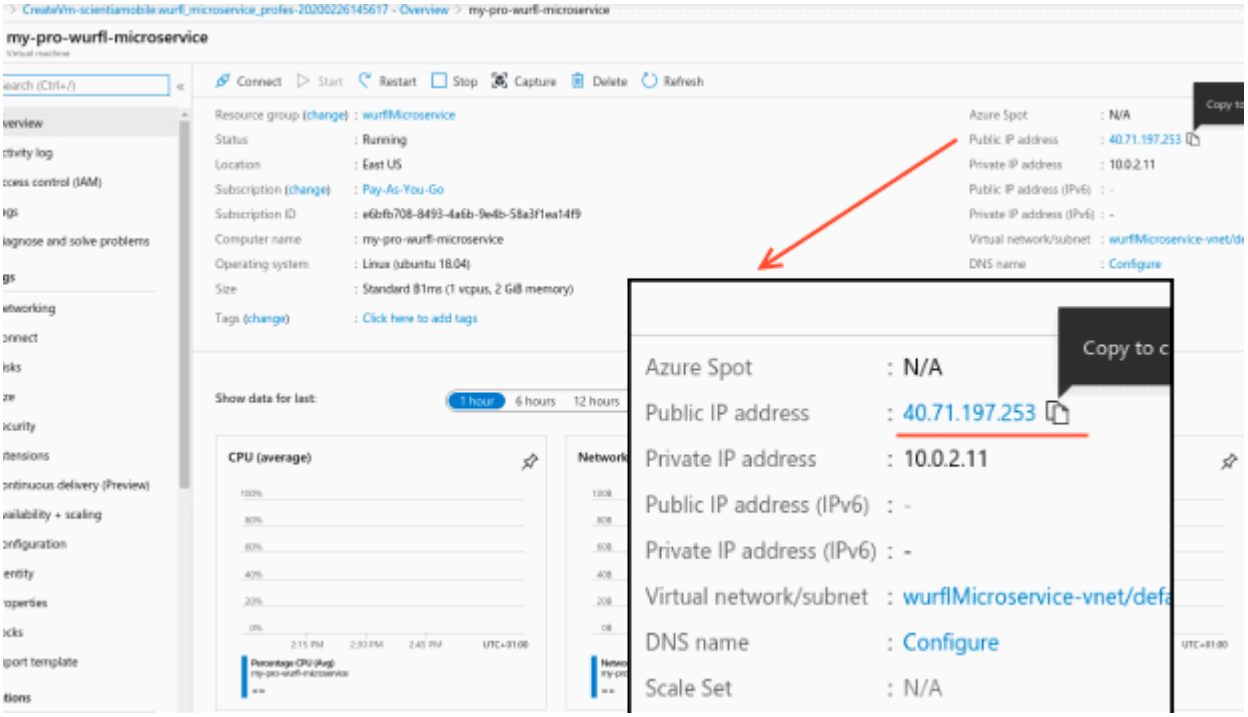


Review your configuration and, if everything looks correct, click on “Create”. The deployment of the new virtual machine starts.

This is the screen that you'll see while the deploy goes on.



When the deploy completes a blue button "Go to resource" will appear. Click on it to access the instance details.



Copy the IP address that you see in the "Public IP address" text field, open a browser and visit the `http://<instance-ip>/v2/status/json` URL address. You should see a small JSON object that vouches for the health of your installation.

Alternatively, you can use `curl` from a shell terminal: assuming your IP is `40.71.197.253`, the following command will confirm that the VM has been deployed successfully and that the WM Server is up and running.

```
$ curl  http://40.71.197.253/v2/status/json
{
    "lookup_request": 0,
    "lookup_useragent": 0,
    "lookup_device_id": 0,
    "make_model_requests": 0,
    "server_info_requests": 8,
    "v1_capabilities_requests": 0,
    "not_found_404": 0,
    "server_uptime": 1172
}
```

if your instance responds with something like this, everything looks good. Your WURFL Microservice server is up and running!

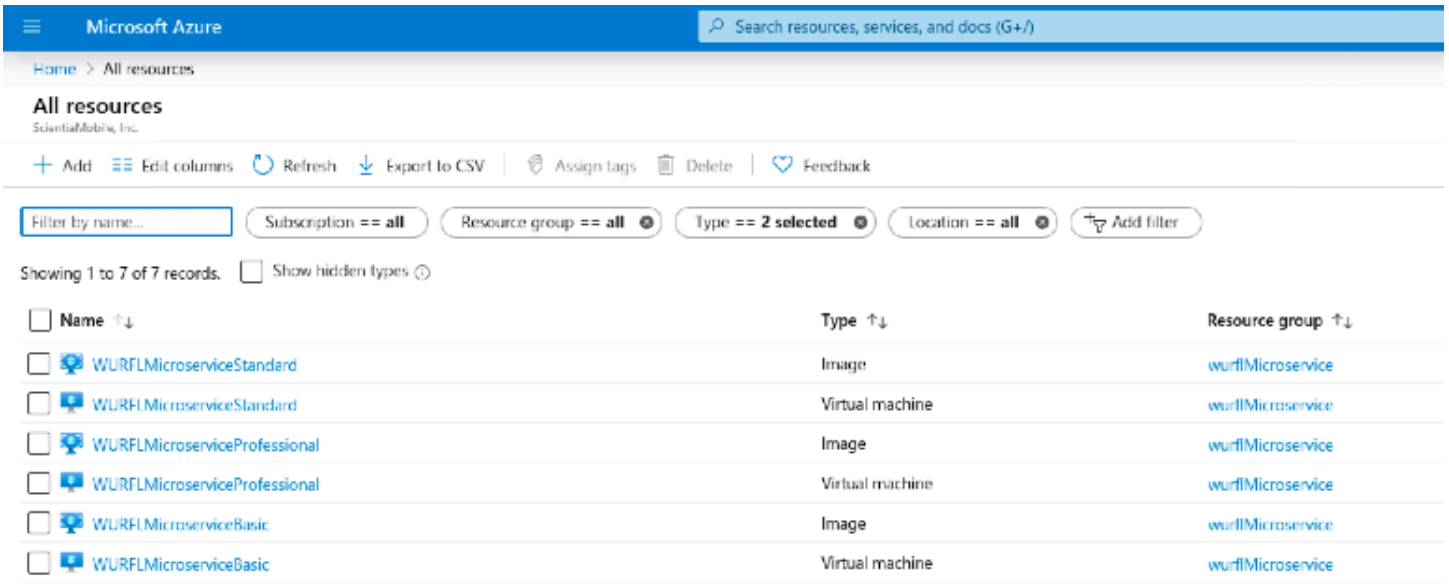
The WM Client API now has a WM server to talk to.

The WM Client API now has a WM server to talk to. More about Client APIs later.

3. Viewing the purchased VM in your Azure dashboard.

You can acces the newly-created instance from your Azure Dashboard.

You will need to know the Instance IP address (`<instance-ip>`) to correctly configure your WM Client API options (following steps). The address can be found in your Azure dashboard.



Note: The configuration presented here is the simplest possible way to obtain a running instance of WURFL Microservice within Azure. In a real production environment, things are likely to be more complex than this and may involve making multiple instances of the product with a load balancer to distribute traffic and achieve fault tolerance. Discussion of the Azure infrastructure is outside the scope of this document.

4. (Only for first-time WURFL Microservice Azure VMs users) Install the WM Client API package and import the API in your project to start using it in your application.

WM Client APIs are available for the following supported languages (regardless of the capability set/tier): Go (golang), Java, .NET, Node.js and PHP.

WM Client APIs for the respective languages are provided on the ScientiaMobile public github repository:

- dotNET : **https://github.com/WURFL/wurfl-microservice-client-dotnet**
- Golang : **https://github.com/WURFL/wurfl-microservice-client-golang**
- Java : **https://github.com/WURFL/wurfl-microservice-client-java**
- Node.js : **https://github.com/WURFL/wurfl-microservice-client-nodejs**
- PHP : **https://github.com/WURFL/wurfl-microservice-client-php**
- Python: **https://github.com/WURFL/wurfl-microservice-client-python**
- Rust: **https://github.com/WURFL/wurfl-microservice-client-rust**

You can find the documentation for each WM Client API here:

1. **Golang (GO Language)**
2. **Java**
3. **Microsoft .NET**
4. **PHP**
5. **Node.js**
6. **Python**
7. **Rust**

Maven Central, Nuget and NPM also carry the WM Client API libraries for Java, .NET and Node.js respectively.

© 2026 ScientiaMobile Inc.
All Rights Reserved.

NOTICE: All information contained herein is, and remains the property of ScientiaMobile Incorporated and its suppliers, if any. The intellectual and technical concepts contained herein are proprietary to ScientiaMobile Incorporated and its suppliers and may be covered by U.S. and Foreign Patents, patents in process, and are protected

by trade secret or copyright law. Dissemination of this information or reproduction of this material is strictly forbidden unless prior written permission is obtained from ScientiaMobile Incorporated.