Table of content

Web-application assignment

Tutorial with step-by-step instructions on how to deploy a simple web application, with notes on what services, tools and concepts are used and why they were chosen.

Pre-requisites: A developer-IDE with a terminal, GitHub and Azure accounts, preferably with SSH keys already set up, otherwise please refer to Module X first as it is recommended to have this already set up for safety reasons as well as easier management. Basic git knowledge, the user should know how to work with branches.

Module 1: Create the application

Start off by creating the backend application. It will be constructed in .NET, using the MVC (Model-View-Controller) pattern. Start your IDE of choice (the examples in this tutorial will use VS Code version 1.97.2).

1. Open the terminal and use the “+v” icon on the right to open up a new terminal, using the Bash language.
2. Navigate to the directory where you want your application folder to be, I recommend a shorter path near the root (I use C:/DevProj/ in a Windows environment). Within that folder, type this:

```
mkdir YourFolderName
cd YourFolderName
```

3. Now you have a folder created and navigated to it. Let us start the MVC project:

```
dotnet new mvc
dotnet new gitignore
```

4. Then git-initate the folder and add the files to Git staging area:

```
git init
git add .
```

git commit -m “Initiated webapp project folders and .net mvc structure”

Setting up a Git Repository

If you do not already have SSH keys set up, please refer to Module X. You have the option at this point to either create a repository using the graphical interface at github.com, grab the SSH key there and do a git remote add-command to connect your folder to your repository.

In this tutorial however, just like with the MVC project, we will not use graphical interfaces (as we are aiming to automate processes in the future), here we cover how to set up a repository using Bash.

First: Check if you have GitHub CLI installed:

```
gh --version
```

If it is not installed, you will see “bash: gh: command not found”. Check Module X for installation instructions!

Now, let us proceed to initialize the repo:

```
gh repo create YourRepoName --public --source=. --remote=origin --
push
```

Then just make sure you push up the initialized project. After this step I would recommend swapping to a development branch instead:

```
git push -u origin main
git checkout -b dev
```

Add your name to the landing (index) page:

This can be done in several ways, but the simplest would be:

Open Views/Home/index.cshtml and refactor the text to add your name at chosen location. The "ViewData" section is the header showing on top of the browser and the text segment below is html-code. <h1> is a header and <p> is a paragraph. See endnote example:[i]

Alterating the text like I did in the example above will result in something like this (see endnote).[ii] Another option could be to edit the html inside Views/Shared/_Layout.cshtml

Module 2: Provision a hosting environment

Like the index page name-adding, we have multiple ways to provision a hosting environment in Azure. The obvious one would be to use the Azure portal! However, if we want to take this one step further, to prepare for automate deployment, it would be better to use Azure CLI.

Taking this even one step further, we might find ourselves in a situation where we would want multiple similar setups, then a re-usable template would be great. For that purpose, we have ARM templates (and to extend that, tools like Bicep).

1. For this tutorial, we will not use ARM, but I will give a short introduction to it in Module X. Instead, we will provision a simple virtual machine (VM) using Azure CLI. If you haven't already, log in to your azure account (and choose your subscription):

```
az login
az account set --subscription "your-subscription-id"
```

2. Create a resource group (using TemplateRG as name and Sweden Central as location), then verify that the RG was created successfully (endnote pic):[iii]

```
az group create --name TemplateRG --location swedencentral
az group show --name TemplateRG --output table
```

3. Create an Ubuntu 22.04 VM, check status and grab its IP (see endnote)[iv]:

```
az vm create \
     --resource-group TemplateRG \
     --name MyVirtualMachineName \
     --image Ubuntu2204 \
     --admin-username template_admin \
     --generate-ssh-keys

az vm list --resource-group TemplateRG --output table

az vm list-ip-addresses --name MyVirtualMachineName --output table
```

Please note: Adjust RG- and admin-names as you choose, and Ubuntu-image after current version. This tutorial also expects you have to have set up the SSH as previously shown in this tutorial! If not, you may need to use a password instead. However, for security reasons, avoid exposing passwords in code, especially if your project is on GitHub. Use environmental variables, GitHub secrets, SSH keys, or hashed passwords instead.

4. Open a custom port. This assumes you have sudo rights!

By default, SSH uses port 22, which is commonly targeted by attacks. We will manually open a custom port (22042) for additional security. We open local SSH config:

```
sudo nano /etc/ssh/sshd_config
```

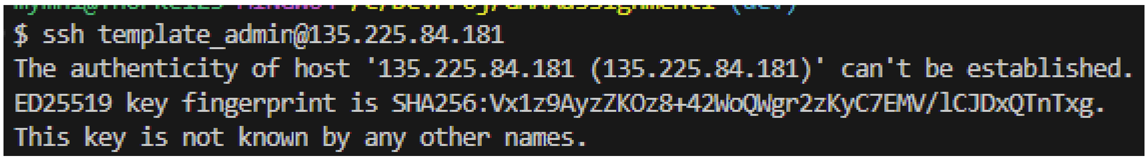
Find the line with #Port 22 (using arrow keys to navigate). Press Enter and type Port 22042. Save by pressing Ctrl + X , then Y and finally Enter to confirm changes. Restart SSH:

```
sudo systemctl restart ssh
```

5. Open the custom SSH port and then connect (to port 22!):

```
az vm open-port --resource-group TemplateRG --name
MyVirtualMachineName --port 22042
ssh template_admin@135.225.84.181
```

You will get this confirmation box (see below). Type "yes" and enter:



```
$ ssh template_admin@135.225.84.181
The authenticity of host '135.225.84.181 (135.225.84.181)' can't be established.
ED25519 key fingerprint is SHA256:Vx1z9AyzZKOz8+42WoQWgr2zKyC7EMV/lCJDxQTnTxg.
This key is not known by any other names.
```

6. Now you should be on the VM, make sure Azure allows 22042 access:

```
sudo nano /etc/ssh/sshd_config
```

Once again, add Port 22042 below the #Port22 line. Ctrl + X, Y and Enter.

```
sudo systemctl restart ssh
sudo ufw allow 22042/tcp
sudo ufw reload
exit
ssh -p 22042 template_admin@135.225.84.181
```

What you have done here is this: the 22042 port was open in the NSG (Network Security Group) and allowed by UFW (firewall, it is probably inactive, but you want that setting there anyway) but SSH is not configured to listen to 22042. So, we entered the VM and adjusted the SSH-configuration file. After that we can access the server with the ssh -p 22042 string.

7. Optional for the brave, but recommended for security:

Run the first command. Find your NSG name (mine was MyVirtualMachineNameNSG). Run the following commands. This turns off Port 22:

```
az network nsg list --resource-group TemplateRG --output table
az network nsg rule list --resource-group TemplateRG --nsg-name
MyVirtualMachineNameNSG --output table
az network nsg rule delete --resource-group TemplateRG --nsg-name
MyVirtualMachineNameNSG --name default-allow-ssh
```

Shit hits the fan: restart the RG

You can run this command to delete the resource group, in case you need to start over:

```
az group delete --resource-group TemplateRG
```

Make sure to adjust “TemplateRG” to whatever name you gave your RG.

Setting up a script to install Nginx:

1. Navigate to your root folder. Create a folder:

```
mkdir -p Scripts
```

2. Right click the Scripts folder in VS Code, choose New File and name it install_nginx.sh
3. Write this code in the install_nginx.sh file:

```
#!/bin/bash
set -e  # Stop the script if any command fails

echo "Updating package lists..."
sudo apt-get update -y

echo "Installing Nginx..."
sudo apt-get install nginx -y

echo "Starting and enabling Nginx..."
sudo systemctl start nginx
sudo systemctl enable nginx

echo "Nginx installation complete."
```

This will be a basic script just for installing, updating and starting Nginx. It has a basic error handler (set -e) to stop it if any command fails. It will be the foundational script we build on further down this tutorial.

We need to make this script executable, so we need to adjust the rights. For this we have chmod. We can either chose to use numbers like chmod 600 or chmod 400 to adjust rights, each number reflects a read/write/execute right for the user. But to make a script executable, we can also just use +x and the path, like this:

chmod +x Scripts/install_nginx.sh

Module 3: Configure the environment

Answer here

“Installera .NET runtime samt skapa en servicefil så att du kan starta din applikation som en service”

Module 4: Deploy the application

Answer here

“Driftsätt applikationen (deploy) i värdmiljön du skapat och konfigurerat i föregående uppgifter”

Module 5: Verify your solution

Answer here

“Verifiera att webapplikationen fungerar och att den kan köra i molnet, samt att den är nåbar från internet.

Ta en skärmdump från landningssidan med ditt namn på. Se till att adressraden i webbläsaren går att läsa tydligt. Klistra in skärmdumpen på första sidan i din inlämnade rapport.

Tänk på att hela rapporten skall vara i PDF-format innan den lämnas in.

Tips: I själva huvuddelen av rapporten behöver du bara redovisa intressanta delar av skärmdumpar eller kodavsnitt. Fullständig kod kan med fördel läggas allra sist i rapporten.”

Module X: In-depth guides

GitHub SSH key-installation instructions:

Here be dragons (TODO)

Azure SSH key-installation instructions:

Here be dragons (TODO)

GitHub CLI-installation instructions:

Install GitHub CLI using winget (Windows), homebrew (Mac) or whatever Linux distro you use. For Windows:

```
winget install --id GitHub.cli
```

One problem that could happen here, which I often see in Windows 11 Pro, is that the PATH often does not work. This means the system cannot find the file. For example, I can clearly see this: C:\Program Files\GitHub CLI\ has a gh.exe, but searching for it will return nothing!

You solve this by doing the following:

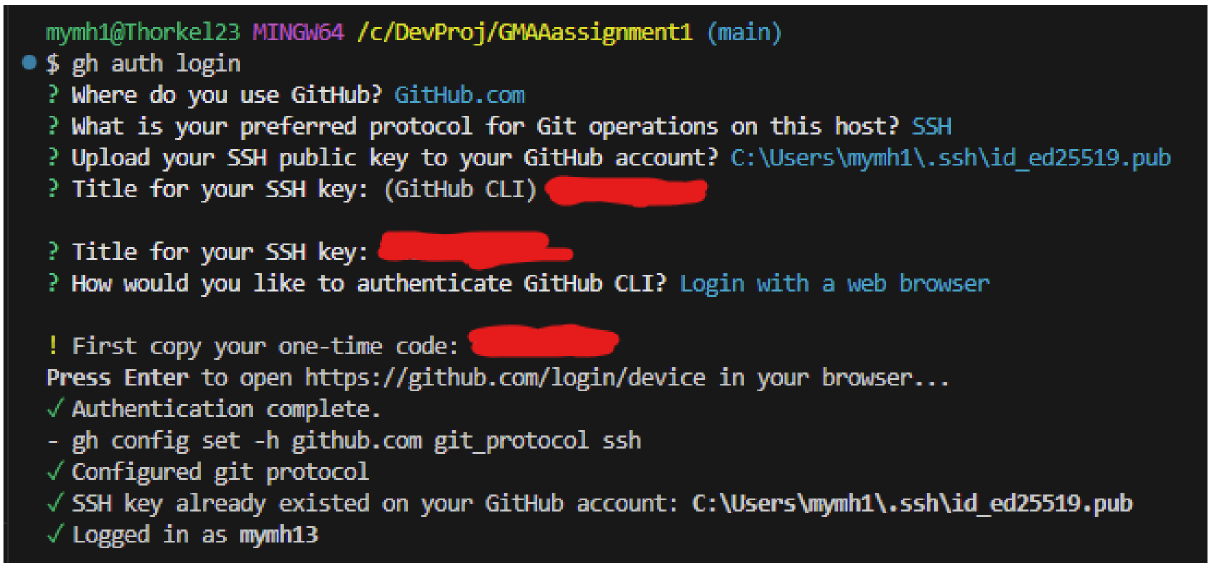
1. Win + R, type sysdm.cpl and Enter (see image in endnotes).[v]
2. Go to the Advanced tab. Click Environment Variables.
3. In the lower window (system variables), scroll down and select the Path variable.

4. Click Edit. In the new window, click New and add: C:\Program Files\GitHub CLI\
5. Click OK in all windows to close them, then close all your open terminals by typing "exit". Re-start a terminal and type "gh --version", it should be something like:

```
gh version 2.67.0 (2025-02-11)
https://github.com/cli/cli/releases/tag/v2.67.0
```

At this point you need to make sure gh is authorized to log into your GitHub account. Type the following and then watch the image below for choices:

gh auth login



```
mymh1@Thorkel23 MINGW64 /c/DevProj/GMAAassignment1 (main)
$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? C:\Users\mymh1\.ssh\id_ed25519.pub
? Title for your SSH key: (GitHub CLI) [redacted]

? Title for your SSH key: [redacted]
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: [redacted]
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ SSH key already existed on your GitHub account: C:\Users\mymh1\.ssh\id_ed25519.pub
✓ Logged in as mymh13
```

Title is your choice of title for the SSH key, choice of authentication is easier by just opening a web browser and accepting GH to access your account, otherwise you have to create a token and that is slightly more complicated. Follow the instructions and you should be set.

What is ARM (Azure Resource Manager)?

It is the deployment and management service for Azure. It can be accessed through Azure Portal (easy to use interface), Azure CLI (cross-platform and great for scripting and automation), PowerShell Modules (extended tooling but Windows centric), REST API (limitations to automation and complex) and the SDK (similar to API but less flexible).

ARM templates are written in JSON and are declarative, which means instead of having an ordered list of tasks for the system (imperative), we tell the system "this is what we want it to do" and the system will figure out how to get there. Since it is quite a potent tool, it can be challenging and complex to write bigger ARM templates. Thus, there are tools like Bicep. We will not expand into that or cover it in this template.

There is a section in Module X where I describe how to set up a basic ARM template.

Create a basic ARM template (reference material, see endnote):[vi]

1. Open the Azure Portal and navigate to Resource Groups.
2. Click "create a new resource group".

3. Chose a template resource group name like “TemplateRG”.
4. Chose a region you are likely to use like “Sweden Central”.
5. Click “review + create” but do not click Create!
6. Now click the “automation link” (see endnote picture).[vii]
7. On the Template page, we have two documents:
   - Template (defines the resources required to create the RG)
   - Parameters (values used in the template)
8. Click the Download button above the Template tab.
9. Extract the zip where you want your project.

At this point you can set the parameter values in your project in the Parameters json-file and use the Template to define resources required for the resource group (RG).

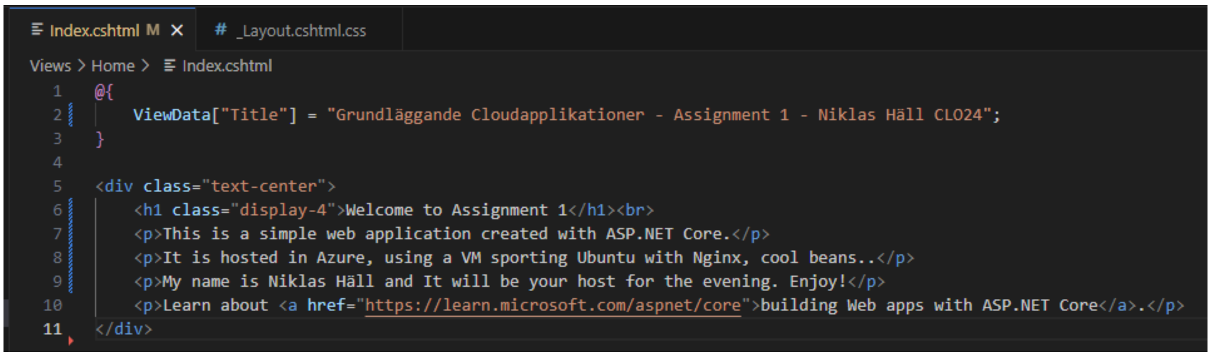
You can use this code to deploy the jsons through the Azure CLI:

```
az deployment sub create --location swedencentral --template-file
template.json --parameters parameters.json
```

What does this do? It creates a resource group, as that is what is the template we have initiated and the command we used. To create another ARM template we will have to adjust the Template (what are we creating) and the Parameters (values for the content).

Reference / Endnotes:

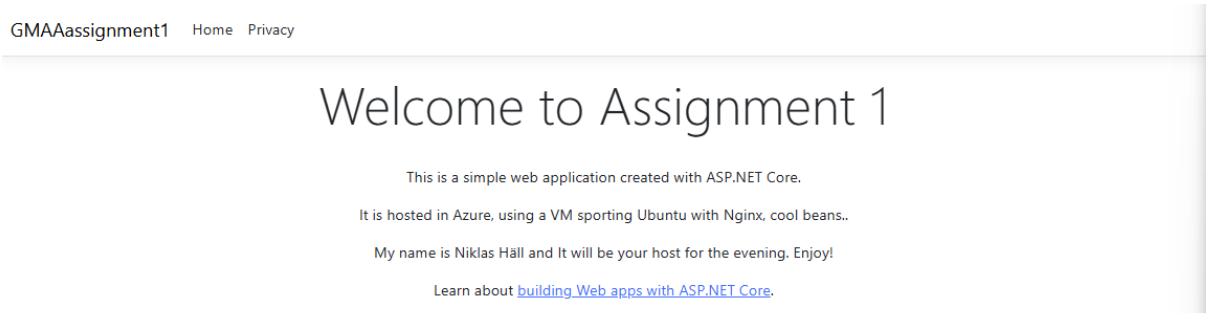
i



```
@{
    ViewData["Title"] = "Grundläggande Cloudapplikationer - Assignment 1 - Niklas Häll CLO24";
}

<div class="text-center">
    <h1 class="display-4">Welcome to Assignment 1</h1><br>
    <p>This is a simple web application created with ASP.NET Core.</p>
    <p>It is hosted in Azure, using a VM sporting Ubuntu with Nginx, cool beans..</p>
    <p>My name is Niklas Häll and It will be your host for the evening. Enjoy!</p>
    <p>Learn about <a href="https://learn.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>
```

ii



Welcome to Assignment 1

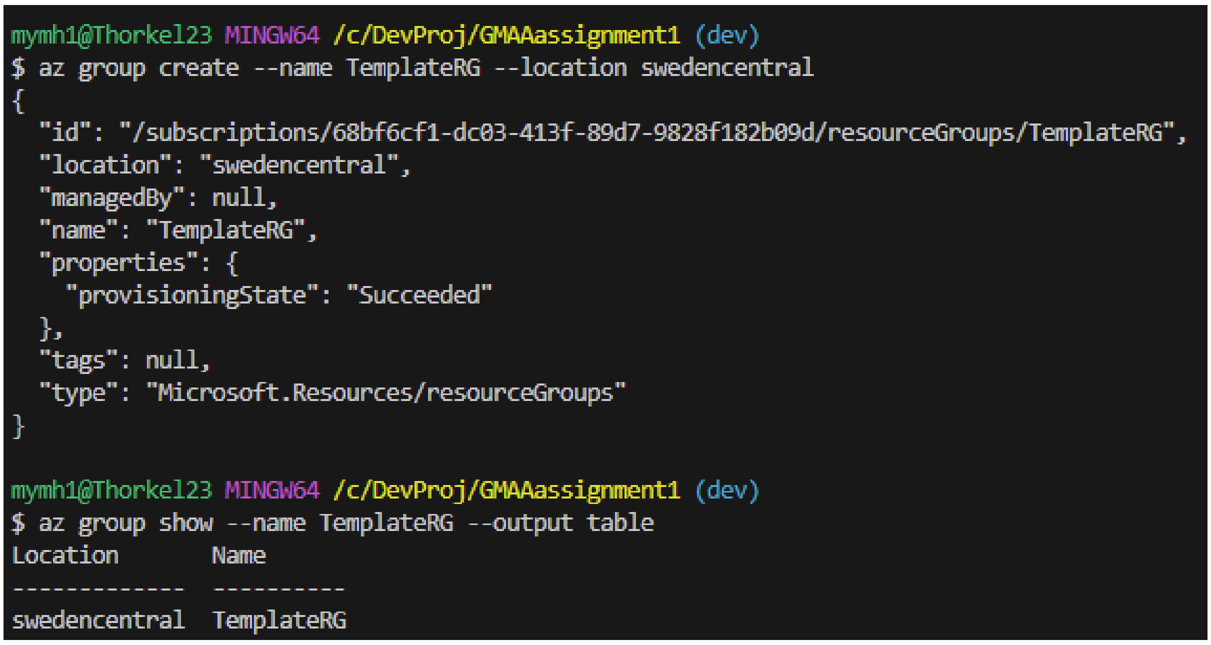
This is a simple web application created with ASP.NET Core.

It is hosted in Azure, using a VM sporting Ubuntu with Nginx, cool beans..

My name is Niklas Häll and It will be your host for the evening. Enjoy!

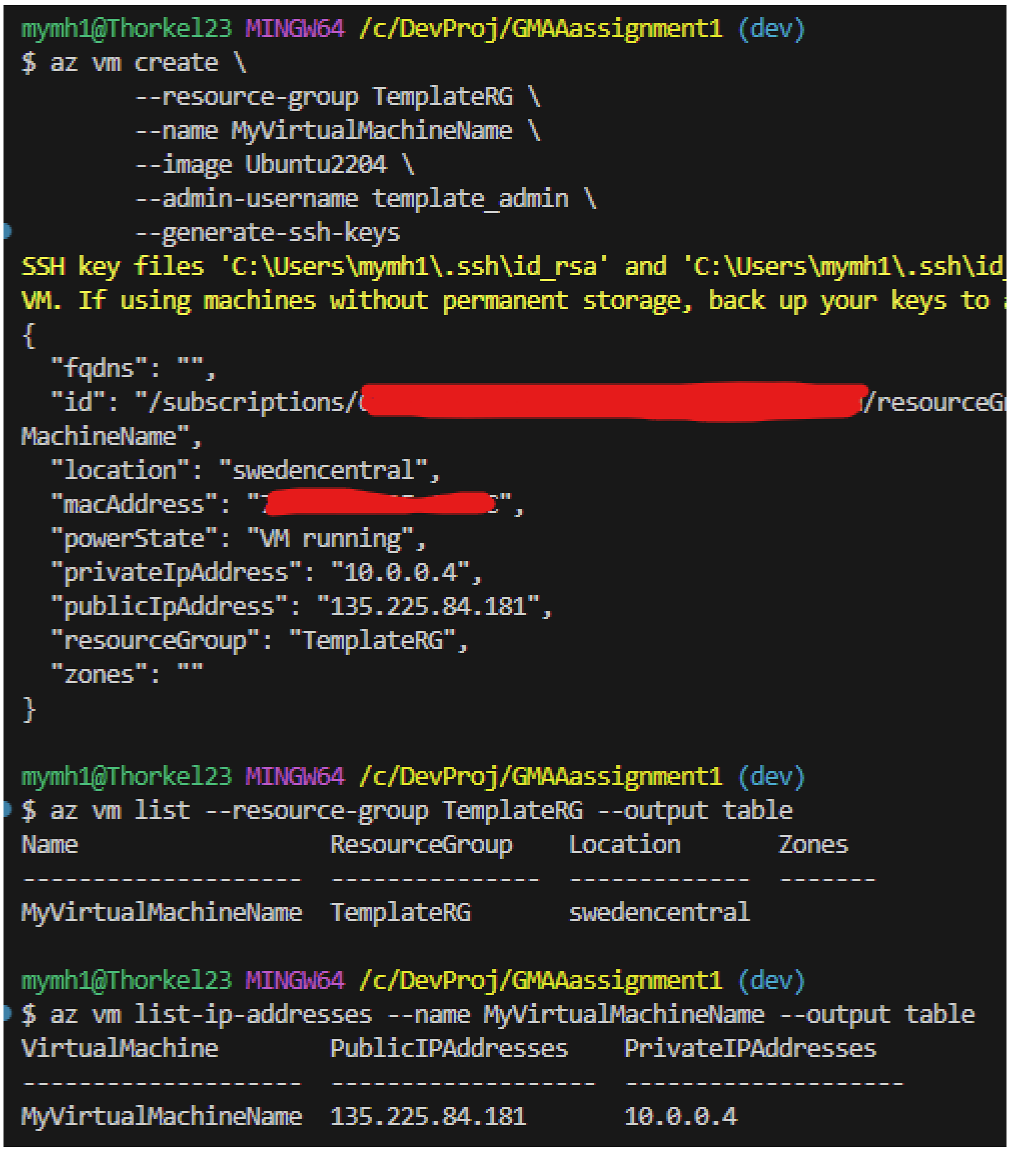
Learn about building Web apps with ASP.NET Core.

iii



```
mymh1@Thorkel23 MINGW64 /c/DevProj/GMAAassignment1 (dev)
$ az group create --name TemplateRG --location swedencentral
{
  "id": "/subscriptions/68bf6cf1-dc03-413f-89d7-9828f182b09d/resourceGroups/TemplateRG",
  "location": "swedencentral",
  "managedBy": null,
  "name": "TemplateRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}

mymh1@Thorkel23 MINGW64 /c/DevProj/GMAAassignment1 (dev)
$ az group show --name TemplateRG --output table
Location       Name
-------------  ----------
swedencentral  TemplateRG
```



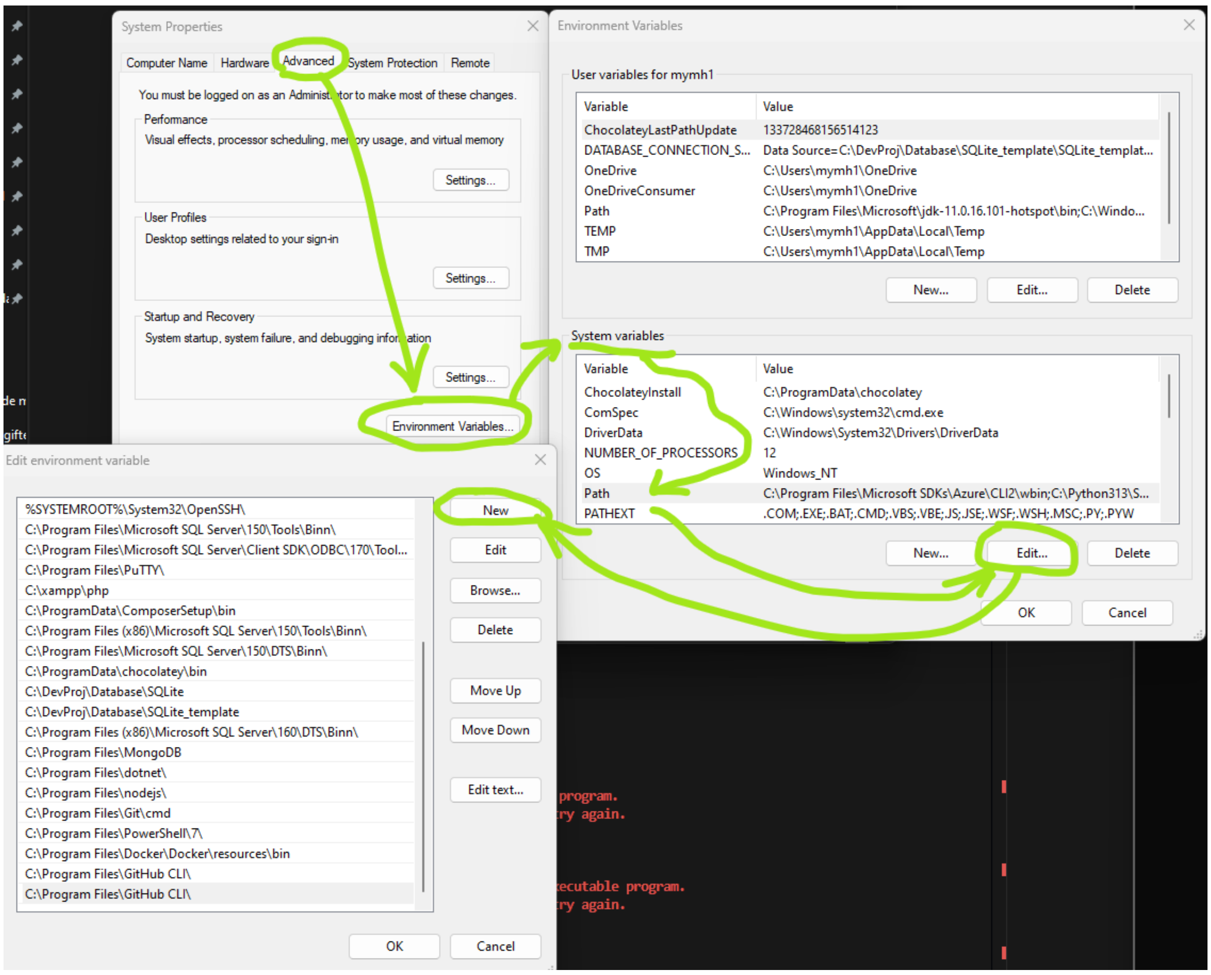
```
mymh1@Thorkel23 MINGW64 /c/DevProj/GMAAassignment1 (dev)
$ az vm create \
        --resource-group TemplateRG \
        --name MyVirtualMachineName \
        --image Ubuntu2204 \
        --admin-username template_admin \
        --generate-ssh-keys
SSH key files 'C:\Users\mymh1\.ssh\id_rsa' and 'C:\Users\mymh1\.ssh\id
VM. If using machines without permanent storage, back up your keys to 
{
  "fqdns": "",
  "id": "/subscriptions/[redacted]/resourceG
MachineName",
  "location": "swedencentral",
  "macAddress": "[redacted]",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "135.225.84.181",
  "resourceGroup": "TemplateRG",
  "zones": ""
}

mymh1@Thorkel23 MINGW64 /c/DevProj/GMAAassignment1 (dev)
$ az vm list --resource-group TemplateRG --output table
Name                  ResourceGroup    Location       Zones
--------------------  ---------------  -------------  -------
MyVirtualMachineName  TemplateRG       swedencentral

mymh1@Thorkel23 MINGW64 /c/DevProj/GMAAassignment1 (dev)
$ az vm list-ip-addresses --name MyVirtualMachineName --output table
VirtualMachine        PublicIPAddresses    PrivateIPAddresses
--------------------  -------------------  --------------------
MyVirtualMachineName  135.225.84.181       10.0.0.4
```

iv

v



System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your sign-in
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

Environment Variables

User variables for mymh1

Variable	Value
ChocolateyLastPathUpdate	133728468156514123
DATABASE_CONNECTION_S...	Data Source=C:\DevProj\Database\SQLite_template\SQLite_templat...
OneDrive	C:\Users\mymh1\OneDrive
OneDriveConsumer	C:\Users\mymh1\OneDrive
Path	C:\Program Files\Microsoft\jdk-11.0.16.101-hotspot\bin;C:\Windo...
TEMP	C:\Users\mymh1\AppData\Local\Temp
TMP	C:\Users\mymh1\AppData\Local\Temp

New... | Edit... | Delete

System variables

Variable	Value
ChocolateyInstall	C:\ProgramData\chocolatey
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	12
OS	Windows_NT
Path	C:\Program Files\Microsoft SDKs\Azure\CLI2\wbin;C:\Python313\S...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW

New... | Edit... | Delete

OK | Cancel

Edit environment variable

```
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Microsoft SQL Server\150\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tool...
C:\Program Files\PuTTY\
C:\xampp\php
C:\ProgramData\ComposerSetup\bin
C:\Program Files (x86)\Microsoft SQL Server\150\Tools\Binn\
C:\Program Files\Microsoft SQL Server\150\DTS\Binn\
C:\ProgramData\chocolatey\bin
C:\DevProj\Database\SQLite
C:\DevProj\Database\SQLite_template
C:\Program Files (x86)\Microsoft SQL Server\160\DTS\Binn\
C:\Program Files\MongoDB
C:\Program Files\dotnet\
C:\Program Files\nodejs\
C:\Program Files\Git\cmd
C:\Program Files\PowerShell\7\
C:\Program Files\Docker\Docker\resources\bin
C:\Program Files\GitHub CLI\
C:\Program Files\GitHub CLI\
```

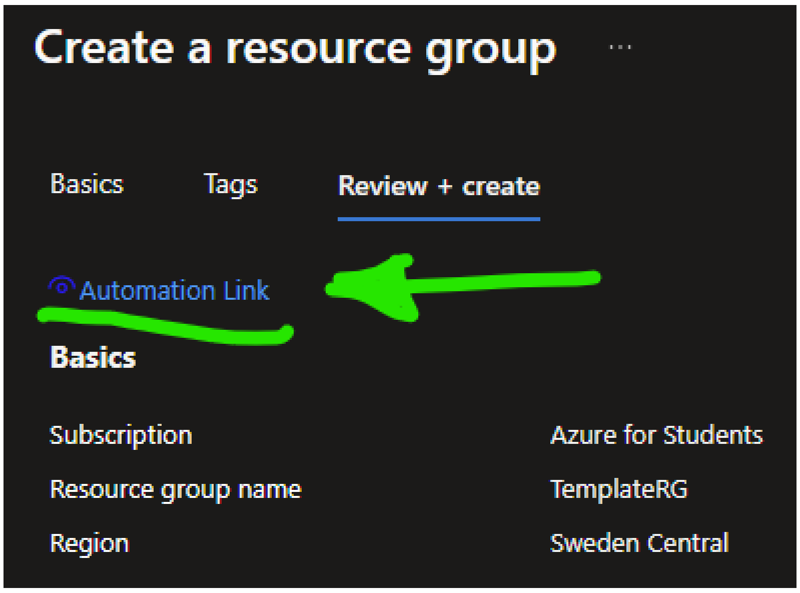
New | Edit | Browse... | Delete | Move Up | Move Down | Edit text...

OK | Cancel

```
program.
ry again.

ecutable program.
ry again.
```

vi https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/template-tutorial-create-first-template?tabs=azure-powershell



Create a resource group ...

Basics | Tags | Review + create

Automation Link

Basics

Subscription	Azure for Students
Resource group name	TemplateRG
Region	Sweden Central

vii