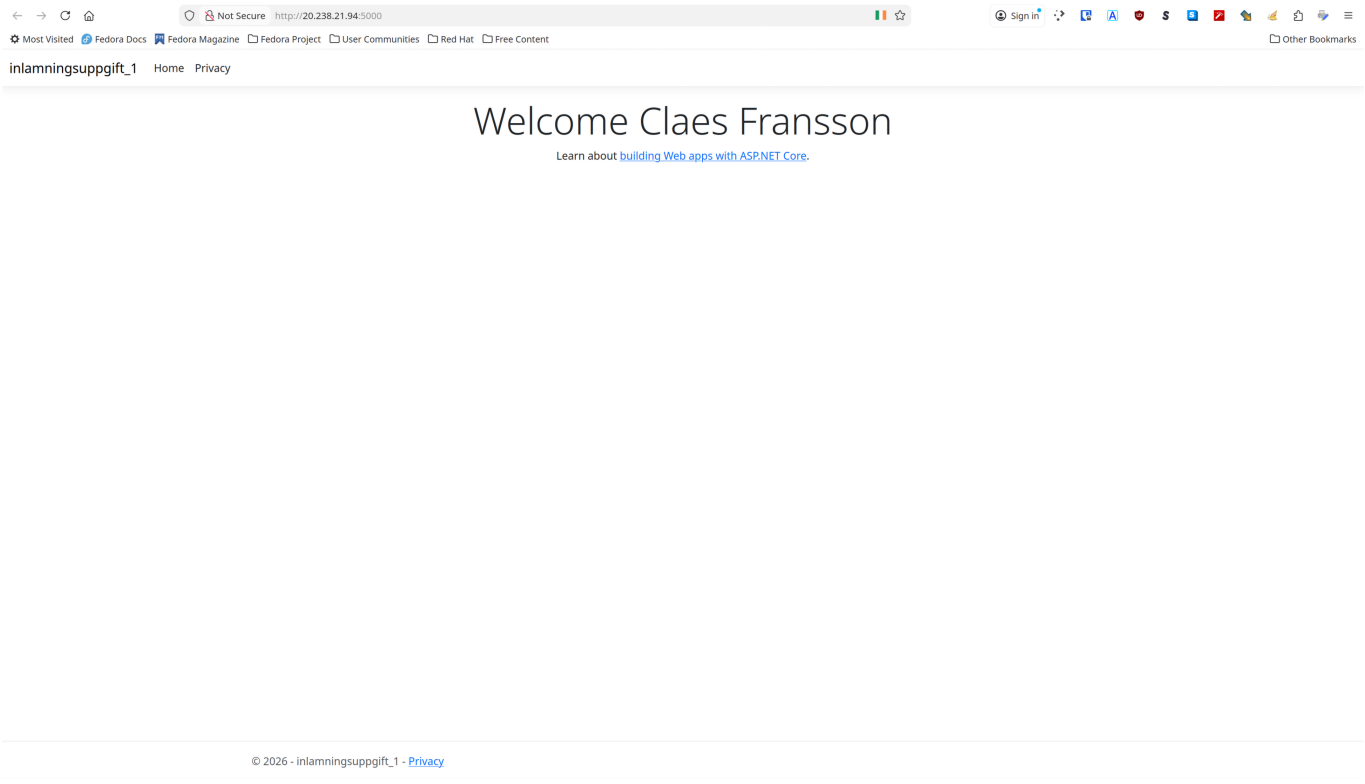
Claes Fransson

https://github.com/Claes1981/inlamningsuppgift_1



Tutorial

"Delmoment 1"

1. Create the web application locally by using the ASP.NET Core Web App .NET project template with `dotnet new mvc -n inlamningsuppgift_1`. It uses the Model-view-controller architectural pattern, which separation of concerns makes it easy to divide development of large complex applications between several developers*. It also uses the client-server model where the ASP.NET Core application is the server, and the browser, which renders Razor views, act as client.
2. Create a "src" directory in the project directory, and move everything into it.
3. Edit the `src/Views/Home/Index.cshtml` file and add your name after "Welcome" in the heading tag.
4. Open the link provided in the output in step 1, in a browser and verify that your name is visible on the home page.

"Delmoment 2"

1. Prepare the provision of a virtual machine in Azure by first copy the provision_vm.sh script to your local computer. This script uses Azure CLI for the provisioning. The *Standard_F1als_v7* in *northeurope*, zone *3* virtual machine size were the cheapest available last time I checked. With a script based provisioning method you apply Infrastructure as Code, which automatically gives you a documentation of exact (in principle) how the infrastructure is set up.
2. Run the script to execute the provisioning. If you previously have set up ssh keys on your computer, they will be used for authentication. That is more secure than e g exposing passwords in code, which could be read by unauthorized.
3. Verify that the virtual machine is accessible by confirming that you can log in to the public ip address of the machine, provided by the script output, e g `ssh azureuser@52.155.250.77`. By default port 22 are open to enable ssh access. The script also opens port 5000 since that port is used by the Kestrel web server on the virtual machine. You will use that port when later remotely connecting to your web app in the browser So far this solution only provide Infrastructure as a Service, since no application development tools are installed yet.

"Delmoment 3"

1. Prepare the configuration of the host environment by first copy the configuration.sh script to your local computer. It contains the Systemd service file as a here-document:

```
[Unit]
Description=.NET MVC Application
After=network.target

[Service]
Type=simple
User=dotnet-app
Group=dotnet-app
WorkingDirectory=/opt/dotnet-app
ExecStart=/usr/bin/dotnet /opt/dotnet-app/
```

```
inlamningsuppgift_1.dll
        Restart=always
        RestartSec=5
        Environment=ASPNETCORE_URLS=http://0.0.0.0:5000
        Environment=ASPNETCORE_ENVIRONMENT=Production

        [Install]
        WantedBy=multi-user.target
```

The `After=network.target` line makes sure that the service is not started unless the network is available. The `Restart=always` line makes Systemd restart the app if it crashes.
The configuration script also creates a dedicated application service user, which makes it possible to run the application as non-privileged user. That limits the potential damage if the application is compromised.

2. Configure the environment by running the script.
3. Verify that the .NET runtime environment was correctly installed by ssh into the machine again and run `dotnet --list-runtimes`.

"Delmoment 4"

1. Deploy your application you created in "Delmoment 1" by running the deployment.sh script on your computer. (Set the `LOCAL_APP_DIR` constant to your local `src` path.)
   The application files are transferred by the `deploy_application()` function, or more precicely, the `scp -r ./publish/* "$USERNAME@$ip:$INSTALL_DIR/"` command, and the application is started as a service by the `sudo systemctl start $SERVICE_NAME.service"` command of the `start_service()` function.
2. Verify that the service is running by ssh into the machine once again and run `sudo systemctl status dotnet-app.service`.

"Delmoment 5"

Verify that the application is accessible from the internet by opening, in a browser on your computer, the link with the public ip provided in the output of the `deployment.sh` script, and test that the web app is working as expected.