Deploying containers on Beanstalk

Install Docker.

Run the following command in Bash:

docker --version

You should see your installed Docker version printed.

Verify that the Docker daemon is running in the background (there should be a whale symbol on the system tray.)

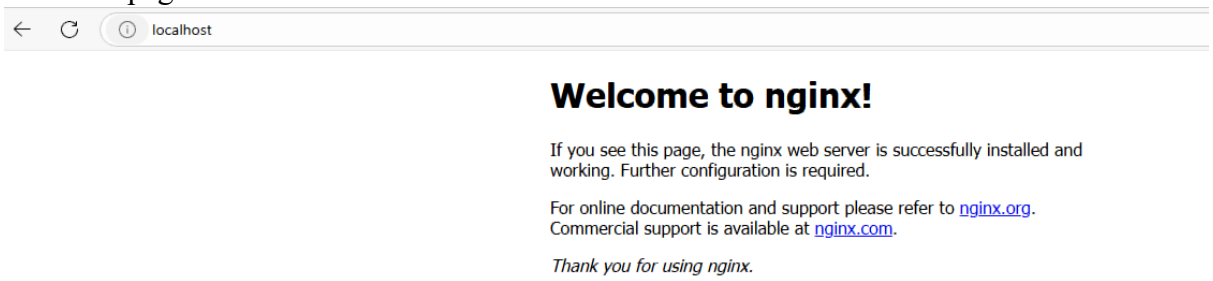
Now we are going to set up a container, open Bash and type:

docker run -d -p 80:80 nginx

```
Felhasznalo@DESKTOP-OPLV4ST MINGW64 ~
$ docker --version
Docker version 27.4.0, build bde2b89

Felhasznalo@DESKTOP-OPLV4ST MINGW64 ~
$ docker run -d -p 80:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
af302e5c37e9: Pull complete
207b812743af: Pull complete
841e383b441e: Pull complete
0256c04a8d84: Pull complete
38e992d287c5: Pull complete
9e9aab598f58: Pull complete
4de87b37f4ad: Pull complete
Digest: sha256:0a399eb16751829e1af26fea27b20c3ec28d7ab1fb72182879dcae1cca21206a
Status: Downloaded newer image for nginx:latest
d5bfd728d93fbbb34d925f736bc4aa898bb61576e1efda8d3c8862a1546413b7
```

Open your web browser and navigate to http://localhost. You should see the default Nginx welcome page.



Build your custom image

In this step, you're going to:

- Write a set of instructions that tells Docker how to build your custom container image.
- Get Docker to read your instructions and build the image.

- Create a folder and name it Compute. You can do this in your terminal by running the following commands:

mkdir Compute

Open the Compute folder by running this command:

cd /c/Users/Felhasznalo/Compute/

Create a new file in your **Compute** folder called Dockerfile (note that there is no file extension after the file name). You can do this in your terminal by running the command:

touch Dockerfile

Open the **Dockerfile** by running vim Dockerfile.

Add the following lines to your Dockerfile:

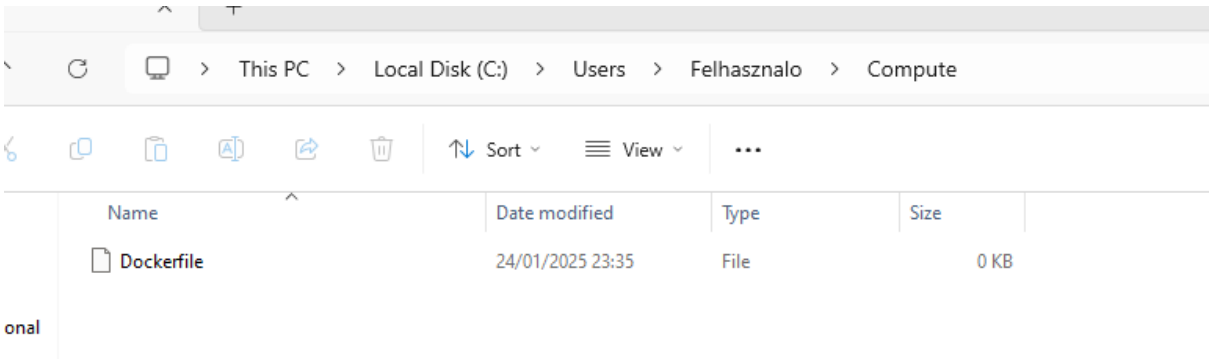
FROM nginx:latest

COPY index.html /usr/share/nginx/html/

EXPOSE 80

(switch to insert (i), ESC to go back to command mode, now save and quit (:wq))

Save the changes.



Creating the web page for your container

Now that we have your Dockerfile ready, what is the web page that your Nginx container should serve? We'll put together a simple HTML file for this project.

- Run these commands in your terminal to create a new file named index.html. This file should be in the same directory as your Dockerfile.

touch index.html

Open your **index.html** file and add this example HTML content:

- Note: if double clicking on the file automatically opens **index.html** in your browser, make sure to right click on index.html instead and open it with your text editor.
- Make sure you replace YOURNAME with your name.

<!doctype html>

```
<html>
  <head>
    <title>My Web App</title>
  </head>
  <body>
    <h1>Hello from YOURNAME's custom Docker image!</h1>
  </body>
</html>
```

Building the Image

Now, let's build the image!

- Head back to your local terminal, make sure you're still in the Compute directory, and run:

docker build -t my-web-app .

Run a container with your custom image

Image building done.

Next up is to run the container with the image we've built.

In this step, you're going to:

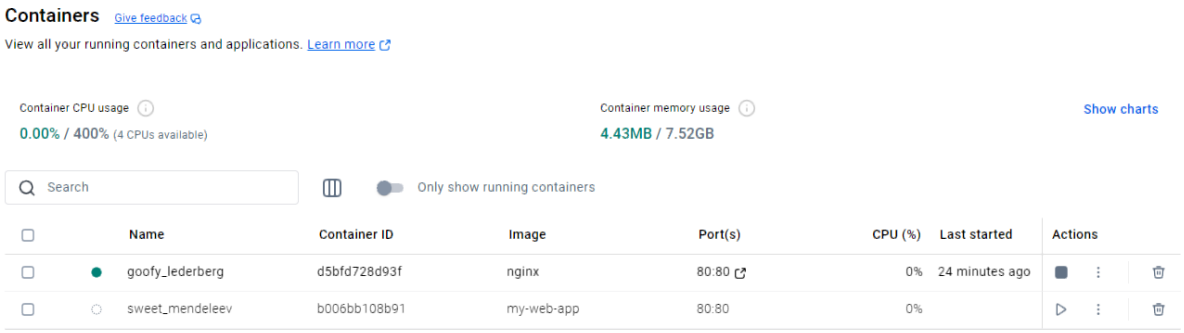
- Run your custom container image, so you can see your webpage in action.
- To run an image, use this command:

docker run -d -p 80:80 my-web-app

This error comes up when there's already a container using port 80, so the new container you're creating can't access it.

Let's figure out who's taking up port 80 right now.

- You can find the container that's running in port 80 by visiting **Docker Desktop** again. The container's icon should be green if it's running.
- This is the Nginx container you created when you first ran **docker run -d -p 80:80 nginx**. Culprit found.
- You can then stop the container by selecting the square stop icon. The container icon should turn grey.



Run docker run -d -p 80:80 my-web-app again, now it is working.

Open your web browser and navigate to http://localhost again. If we've done everything correctly, then we should see the content of our index.html file.

Log in to AWS with your IAM user

- Log in with your IAM Admin User.

Deploy your custom image to Elastic Beanstalk

In this step, we'll deploy your custom Docker image to AWS Elastic Beanstalk, a service that makes it easy for developers to deploy applications in the cloud.

In this step, you're going to:

- Set up your application in Elastic Beanstalk.
- Deploy your application and make it available to the world!

Creating an Elastic Beanstalk Application

- Head to the AWS Management Console and log in as your IAM user.
- Search for Elastic Beanstalk and click on the service.
- Click on **Create Application** in the Elastic Beanstalk homepage.

In the **Configure Environment** page:

- We'll leave the **Environment tier** as default.
- Enter NextWork App as the application name.
- We'll leave the **Environment information** section with the default values.
- Under the **Platform** section, select **Docker** as your Platform.
- The platform branch and version will be automatically selected for you too.

- In the **Application code** section, select **Upload your code.**
- Before we upload any code, head back to the **Compute** folder in your desktop and open up **index.html** with your text editor.
- Find the line that says <h1>Hello from YOURNAME's custom Docker image!</h1>
- Update index.html by **adding** a new line underneath the line you've found. For example:

```
<!doctype html>

<html>

  <head>

    <title>My Web App</title>

  </head>

  <body>

    <h1>Hello from YOURNAME's custom Docker image!</h1>

    <h1>

      If I can see this, it means Elastic Beanstalk has deployed an image with

      my work.

    </h1>

  </body>

</html>
```

Elastic Beanstalk needs your source code to be in a ZIP file. Create a ZIP file containing **Dockerfile** and **index.html**.

- Your files should be at the root level of the ZIP file, not inside a subfolder.
- To create your ZIP file, in your **Compute** folder, multi-select both your **Dockerfile** and **index.html**, then compress them together.

Elastic Beanstalk needs your source code to be in a ZIP file. Create a ZIP file containing **Dockerfile** and **index.html**.

- Your files should be at the root level of the ZIP file, not inside a subfolder.
- To create your ZIP file, in your **Compute** folder, multi-select both your **Dockerfile** and **index.html**, then compress them together.
- In the **Presets** section, choose **Single instance (Free tier eligible)**.
- Select **Next.**

In the **Configure service access** page:

- Select **Create and use new service role**. This creates a new IAM role that Elastic Beanstalk can use.

- Use the default service role name, which should look like **aws-elasticbeanstalk-service-role**.
- Ignore the EC2 key pair dropdown, since we don't need to access the EC2 instance directly for this project.
- Under EC2 instance profile, choose **ecsInstanceRole**.
- Select **Next.**

In the **Set up networking, database, and tags** page:

- Ignore the **Virtual Private Cloud** section - the default VPC looks good for this project!
- Under **Instance settings,** check **Activated** for the Public IP address option.
- Skip the rest of the page, we don't need to set up a database for our webpage.
- Select **Next.**

For the **Configure instance traffic and scaling** page:

- Under the **Instances** section, for your **Root volume type** select **General Purpose 3 (SSD)**.
- For the **Size** select 10 GB.
- Under **Instance metadata service (IMDS),** make sure IMDSv1 is **Deactivated** i.e. keep the Deactivated checkbox checked.

Since our app won't need access to other AWS services, we won't need IMDS.

- We don't need to configure anything else. Skip the security groups and Capacity sections.
- Select **Next.**

In the **Configure updates, monitoring, and logging** page:

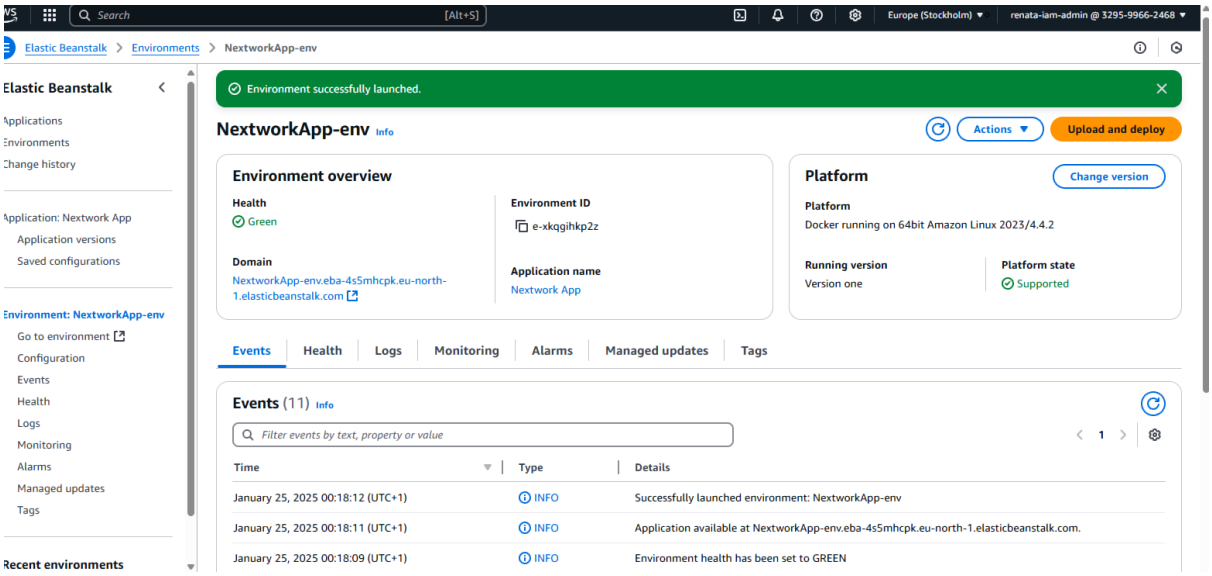
- Under the **Monitoring** section, select **Basic** for your System.
- **Make sure to select Basic for your System to keep your project free.**
- Scroll the **Managed platform updates** section.
- Make sure to **uncheck** the **Activated** checkbox for **Managed updates** option.

We don't need managed updates since we aren't keeping this application for long.

Now in the **Rolling updates and deployments** section, accept the default **All at once** deployment policy.

- Leave all other options, including the **Platform software** section as default.
- Select **Next.**
- For the final step, we'll just need to review our Elastic Beanstalk setup. This rounds up your selections across the last five pages.

- Click **Submit**. This starts the environment creation and application deployment process. This process can take several minutes...
- Environment launch successful!



- Click on the **Domain** link on the environment dashboard.
- The **Domain** link is showing you your updated HTML file, which tells us that Elastic Beanstalk has successfully deployed your ZIP file's contents as a container image.