Lab Assignment 5

Due: 4/22/2025, 11:59pm

Deploying with Containers on Virtual Machines

✸ Goal:

Bundle up your full-stack application (React + Express) as Docker containers, create a Virtual Machine (VM) on Amazon Web Services (AWS), and deploy your application on that instance. This hands-on activity will improve your understanding of DevOps pipelines, Dockerized deployment, and cloud infrastructure.

We are all aware of services such as Google App Engine / Vercel / Railway app and the million others that allow you to host an application as a service. These services are certainly a life saver when time is clutch or when you must deploy something quickly. We are deliberately using an EC2 instance (a Virtual Machine) and taking a barebones approach to understand how things work under the hood. The other options commonly used are using Elastic Container Service with Fargate.

Use LLMs to assist you in generating configuration files, troubleshooting, and exploring best practices. However, ensure the design and decisions originate from your own understanding.

Design and Learning

Learning Docker

Start by watching these videos about Docker. They talk about what docker is and how it gets in your development workflow. We don't yet want to get deep into how things work, but just on how to use these tools.

1/ The Coding Sloth

2/ Fireship

Other resources

Docker Getting Started | Docker Workshop – Docker Compose |

Slides on DevOps | If you are using NextJS, here are their docs. |

AWS EC2

Let's start with the first study course by AWS. This short course will make you familiar with creating a new virtual machine on the cloud. What is EC2

The gist: You are essentially speccing out a new laptop you are buying, but the laptop in this case is maintained by AWS. You get to access it over the internet via command line and network requests.

Linux is usually the preferred OS when deploying things in the cloud, but don't worry ... apart from a few basic commands, you won't need to interact with the terminal much. 😊

Let's finish this short AWS course to help you get started on how to create an instance.

>> AWS Skill Builder - Getting Started with EC2

🔢 Steps to Complete

1. **Ensure Your Application Works Locally**
   Your React and ExpressJS application should run correctly on your local machine.
2. **Install Docker on Your Local Machine**
   Follow official Docker installation instructions for your operating system.
3. **Add a Dockerfile to Your Application**
   a. Create Dockerfiles for both the frontend and backend.
   b. Ensure each container builds and runs independently.
4. **Add a docker-compose.yml File**
   a. Define services for both frontend and backend in case you want to keep them separate micro services.
   It's recommended to compile the frontend into plain old HTML/CSS/JS files and then use a single server to serve everything. But this is not required and take up much time.
   b. Ensure the application starts using: `docker compose up`
5. **Test the Application Locally Using Docker Compose**
   a. Validate all application features work correctly within containers.
6. **Create an AWS EC2 Instance**
   a. Use Amazon Linux 2 as the operating system.
   b. Choose appropriate instance type (e.g., t2.micro if eligible for free tier).

    c. Configure security groups to allow traffic on required ports (e.g., 22, 80, 443, 3000, 5000).
7. **Install Docker on the EC2 Instance**
    a. SSH into the EC2 instance.
    b. Use AWS documentation to install Docker. This is pretty quick!
8. **Clone Your GitHub Repository on the EC2 Instance**
    a. Use git clone to fetch your project onto the VM.
9. **Run the Docker Containers on EC2**
    a. Navigate to your project directory.
    b. Start the application using: `docker compose up –d`
    c. Verify the application is accessible via the EC2 public IP.
10. **(Optional) Enable HTTPS with Caddy or Nginx Proxy Manager**
    a. Set up HTTPS using Let's Encrypt certificates.
    b. Reverse proxy traffic to your Docker containers securely.

Final Deliverables

- A short (single paragraph!) written report (PDF or Markdown) summarizing the steps taken.
- Screenshots or logs showing successful deployment.
- GitHub repository link.
- Public IP or domain of the deployed application
- Summary of how you used LLMs during the assignment.

Appendix

We don't want to be experts in building software / writing docker files yet!
Idealy you should use npm build to generate static files from your frontend and have them deployed, but for now, feel free to use this Dockerfile as a reference

https://gist.github.com/ninadpchaudhari/fe99474e9178134b398718be006da926