

Ministry of Higher Education
and Scientific Research

*** * ***

University of Carthage

*** * ***

**National Institute of Applied
Sciences and Technology**



Summer Internship Report 2020/2021

**Software engineering
4th year**

Subject :

**Troubleshooting a production environment, updating a
Nodejs app to the latest version and containerizing it**

By : Sirine Achour

With:

The University of Michigan



Ministry of Higher Education
and Scientific Research

*** * ***

University of Carthage

*** * ***

**National Institute of Applied
Sciences and Technology**



Summer Internship Report

Software engineering
4th year

Subject :

**Troubleshooting a production environment and updating a
Nodejs app to the latest version and containerizing it**

By : Sirine Achour

With :

The University Of Michigan

<p><i>Under the supervision of: Dr. Bruce Maxim</i></p>	<p><i>Notice from the Internship Committee</i></p>
--	---

Acknowledgments

I would like to express my gratitude to the UM team for giving me this opportunity to work on interesting projects as well as for creating a stress free environment for me to learn and flourish.

I would also like to express my deepest gratitude to Mr. Jeremy York for his help and guidance as well as his due diligence throughout this entire internship.

Finally, I want to thank my family and friends for their constant support and encouragement.

Contents

Introduction	1
1 Introduction	1
2 Presenting the University of Michigan	1
3 Goals and specifications	2
3.1 Troubleshooting an AWS production instance	2
3.2 Updating the NodeJS app to the latest version of NodeJS	2
3.3 Migrating the NodeJS app from the AWS instance to the UM Container Service	2
Work	3
4 Internship journal	3
5 Realized work	4
5.1 Troubleshooting production site	4
5.2 Update site to the latest NodeJS version	4
5.2.1 Moving the app from BitBucket to GitHub	5
5.2.2 Finding extraneous files	5
5.2.3 Studying use of database tables in the app	6
5.2.4 Merging production and development code	7
5.2.5 Merging production and development databases	7
5.2.5.1 Database	7
5.2.5.2 Finding the data	7
5.2.5.3 Primary keys	7
5.2.5.4 Foreign keys	8
5.2.5.5 Generating the SQL queries	9
5.3 Containerizing the site	9

6	Future work	10
	Conclusion	11
7	Consolidation of skills	11
8	Conclusion	11

List of Figures

- 1 Logo of the University of Michigan 2
- 2 Gantt diagram of internship timeline 3
- 3 Translation Networks production site 4
- 4 List of files generated by automation script 6
- 5 Declaration of "collection_comments" table 8
- 6 Declaration of "collections" table 8
- 7 Contents of dockerfile 10

List of acronyms

- **UM** University of Michigan
- **Dev** Development
- **Prod** Production
- **App** Application
- **AWS** Amazon Web Services
- **EC2** Amazon Elastic Compute Cloud
- **SQL** Structured Query Language

1 Introduction

Within the scope of my university studies, I carried out a remote summer internship with The University of Michigan that lasted one month from 15/06/2021 to 15/07/2021.

This internship was an excellent opportunity for me to learn new technologies and attain new skills as well as put into use my competencies and help move along the UM research team projects

During my work with the UM research team, I was given a list of tasks that ranged from troubleshooting an AWS production instance, to analyzing and studying the use of database tables to containerizing a NodeJS website. All these tasks had one goal, which is updating and getting the Translation Networks website to work.

This website is a project that was contributed to by a team of art, history, and IT professors as well as librarians and other university students. It is basically a collection of digital tools that assist students in connecting ideas, creative activities, and sources. One of the main goals of the tools is to promote a better knowledge of translation.

2 Presenting the University of Michigan

The University of Michigan is an American public university that was founded in 1817 in Detroit. It is a highly ranked, highly respected university that is most known for being ranked number 1 in research volume among U.S. public research universities.

It offers different degrees for various majors such as doctoral degrees in the humanities, social sciences, and STEM fields (science, technology, engineering, and mathematics) as well as professional degrees in architecture, business, medicine, law, public policy, pharmacy, nursing, social work, public health, and dentistry.



Figure 1: Logo of the University of Michigan

3 Goals and specifications

I was asked to work on 3 different projects that revolve around the Translation Networks website:

3.1 Troubleshooting an AWS production instance

The Translation Networks site (translationnetworks.com), hosted on an AWS EC2 instance, has stopped working several months ago. This project involves getting the production instance to work again, as well as locating the root cause of the site going down.

3.2 Updating the NodeJS app to the latest version of NodeJS

The Translation Networks site is running on a very early version of NodeJS from 2015. This project is concerned with updating the newest NodeJS version as well as auditing files and removing extraneous files that are no longer (or were never) in use.

3.3 Migrating the NodeJS app from the AWS instance to the UM Container Service

This requires figuring out how to package the current NodeJS site into a container and run it on the UM container service which is a service providing a high availability, secure environment for hosting containerized applications and services.

4 Internship journal

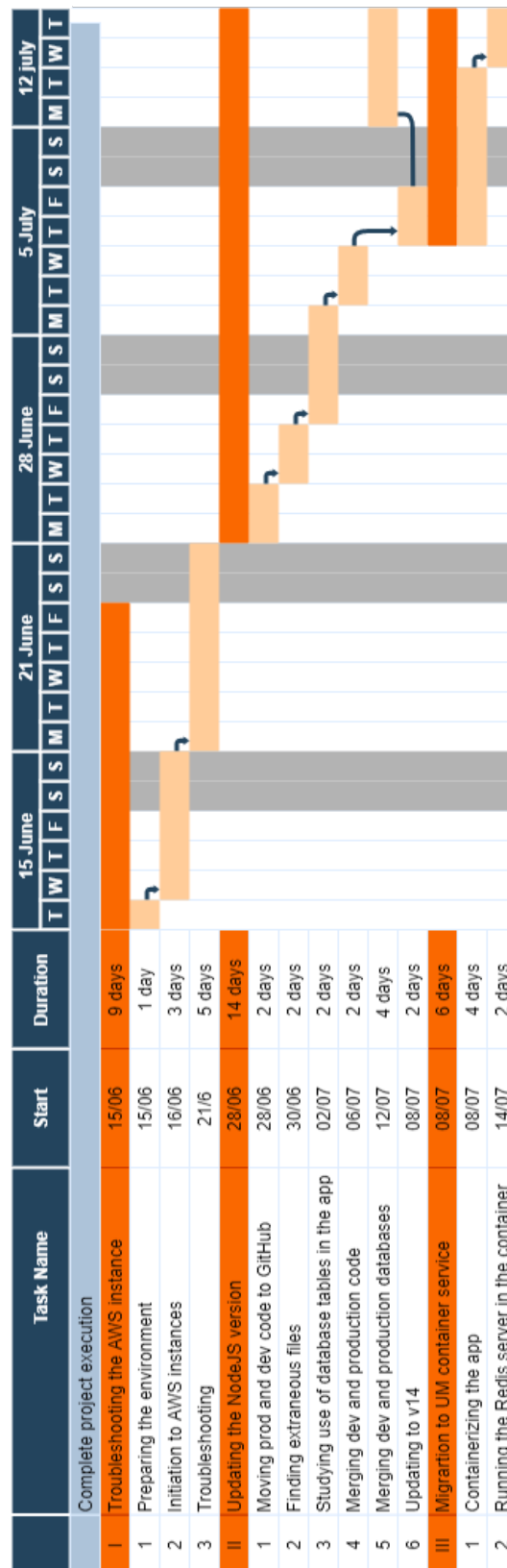


Figure 2: Gantt diagram of internship timeline

5 Realized work

5.1 Troubleshooting production site

The production site, running the AWS EC2 instance, had been down for a few months. After some digging, I found that firstly, the app's process hasn't even been running on the instance, so I simply ran it and I was surprised to find it working. Then after a short while, it stopped working again. So, I kept digging through the instance configurations and the site's code and I discovered that the site listens only on port 8080 and if reached using that port, it works just fine. I concluded, then, that the AWS instance is supposed to be forwarding the requests incoming on port 80 to port 8080, but this port forwarding wasn't set in place correctly. So, I did just that using iptables rules and problem solved.

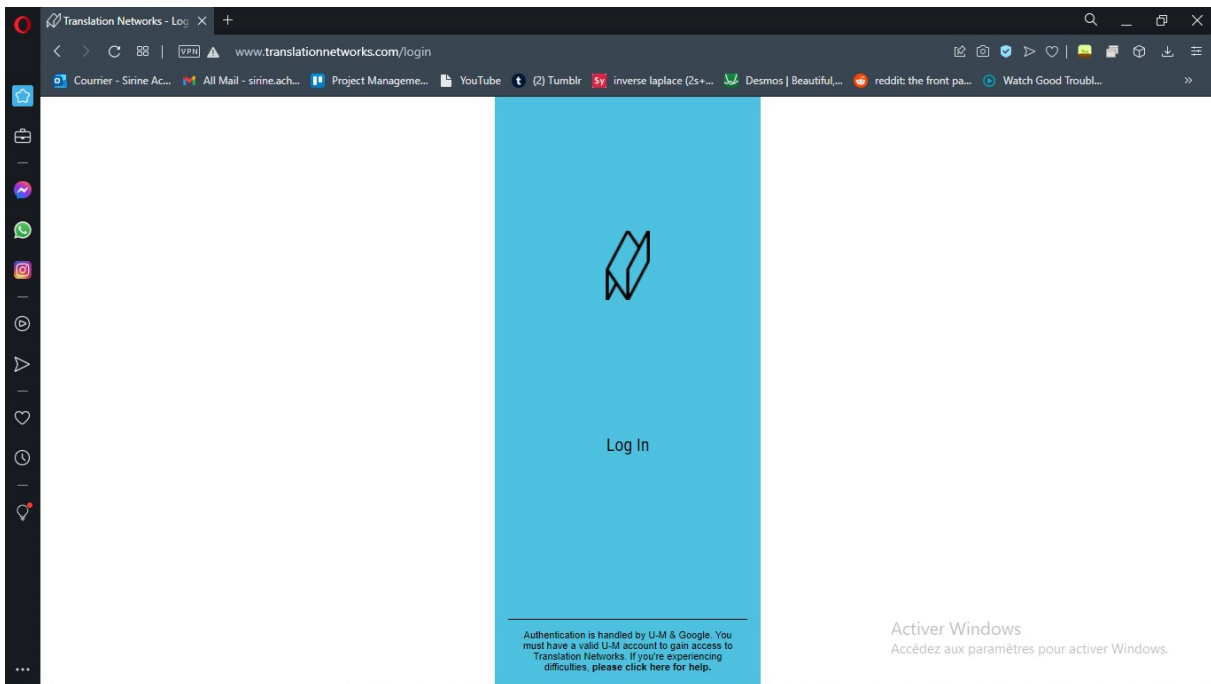


Figure 3: Translation Networks production site

5.2 Update site to the latest NodeJS version

This project had steps set in place, which I followed and finalized one by one :

5.2.1 Moving the app from BitBucket to GitHub

Since the owner of the BitBucket repository isn't someone on the team that I worked with, and since said repository is private, I started by cloning the code to my computer, I then, added an upstream remote URL leading to the new repository. Finally, I pushed to upstream. This was done so that the old commits from the BitBucket repository were still traceable from the new GitHub repository.

I made sure to push the production code to the "main" branch and the development code to the "dev" branch.

5.2.2 Finding extraneous files

This step is about finding the files in the app's code that are not being used and will not be of benefit for future development. I wrote a python script to automate this step. This script loops all files of the app and looks for any call for them in other files then generates the list of files that have never been called/imported by another file. Finally, I manually looped through this list to assess the benefit that each file might bring, then I finished by deleting the files that are of no use to the team.

```
TN-Nodejs-Production\Nathan Test.html
TN-Nodejs-Production\server.js
TN-Nodejs-Production\setupiptables.bash
TN-Nodejs-Production\util\1441749617616.jpg
TN-Nodejs-Production\util\cronjob.sh
TN-Nodejs-Production\util\glacierBackup.sh
TN-Nodejs-Production\views\collections\collection.jade
TN-Nodejs-Production\views\collections\collections.jade
TN-Nodejs-Production\views\explore\timeline.jade
TN-Nodejs-Production\views\explore\tree.jade
TN-Nodejs-Production\views\login\noaccess.html
TN-Nodejs-Production\views\misc\construction.jade
TN-Nodejs-Production\views\slates\color scheme.jpg
TN-Nodejs-Production\views\slates\icon-link.png
TN-Nodejs-Production\views\slates\icon-text.png
TN-Nodejs-Production\views\slates\textTool.png
TN-Nodejs-Production\views\slates\Translation Network Icons - 100%.png
TN-Nodejs-Production\views\slates\Translation Network Icons - Clone.png
TN-Nodejs-Production\views\slates\Translation Network Icons - Share.png
TN-Nodejs-Production\views\slates\Translation Network Icons - VIDEO.png
TN-Nodejs-Production\views\slates\Translation Network Icons - View.png
```

Figure 4: List of files generated by automation script

5.2.3 Studying use of database tables in the app

The point of this task is to judge the legitimacy of an existing excel spreadsheet that is supposed to contain a list of all of the database tables along with whether or not they have been used in the code. This spreadsheet is part of the documentation that the team is trying to maintain and update regularly for research and development purposes.

I started by determining what type of database was being used and found that it was a MySQL database. I, then, located the database on the server and sent its SQL dump to my computer. Next, I wrote a python script that finds all the database tables that have been referenced in the code and saves them to a text file. Then, I wrote another script that compares the contents of that file to the spreadsheet and generates 3 lists: a list of tables used in the code but are marked as "not used" in the spreadsheet, a list of tables that are "used" in the spreadsheet but never referenced in the code and finally a list of tables that are referenced in the code but not mentioned in the sheet.

5.2.4 Merging production and development code

This task was fairly simple. I merged the "dev" branch into "main" and resolved the resulting conflicts.

5.2.5 Merging production and development databases

I started by comparing the development and production databases and found that there are no differences in the schemas.

As for the data, I wrote a python script that finds the missing "INSERT INTO" SQL statements in each database dump and saves them to a file. When looking for queries that are in the development database but are missing from the production database, I found 2470 entries. However, when trying the other way around, I found 13826 queries. In conclusion, it is smart to pump the missing data into the production database rather than the development database.

After determining what I will be merging into, I decomposed the issue of merging the 2 databases into 5 main parts:

5.2.5.1 Database

I started by getting familiar with the database and its schema.

5.2.5.2 Finding the data

Problem : Finding which data exists in the development database but not in the production database.

Solution : I exported the 2 databases (schema + data). The python script, then, loops through both SQL files to find the "INSERT INTO" SQL statements that are part of the development database dump file but the production database's.

5.2.5.3 Primary keys

Problem : Some tables have auto-incremented primary keys. This means that when inserting a new entry in the table, we must pass it a NULL value for the primary key. MySQL will then automatically generate a viable value for that entry. So, if we were to insert an entry from the development database, into a table with the "AUTOINCREMENT" feature, we might run into the issue of having duplicate primary keys for different data. This will raise an error and abort the insertion of the new entry.

Solution : I went through the "CREATE TABLE" SQL statements and generated a list of tables having an auto-incremented primary key. I, then, collected all the new data that is being inserted into the tables in that list and set to NULL all the primary keys.

5.2.5.4 Foreign keys

Problem : The issue here is with the foreign keys pointing to the primary keys that we previously set to NULL.

Another issue is that in the current database schema, there are no actual foreign keys relations set in place. These relations are implied and can be logically concluded but are not recognized by MySQL. This makes the task of spotting the actual foreign keys very hard.

A prime example of this issue is displayed in the "collection_comments" and "comments" tables.

```
CREATE TABLE `collection_comments` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `collection` int(11) NOT NULL,  
  `user` varchar(16) NOT NULL,  
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  `text` varchar(4096) NOT NULL,  
  `status` enum('default','deleted') NOT NULL,  
  `type` enum('comment','note','wishlist') NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=36 DEFAULT CHARSET=utf8;
```

Figure 5: Declaration of "collection_comments" table

```
CREATE TABLE `collections` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `name` varchar(64) NOT NULL,  
  `description` varchar(4096) DEFAULT NULL,  
  `works` varchar(512) DEFAULT NULL,  
  `visibility` enum('public','class','private','instructors') NOT NULL,  
  `editable` enum('private','public','class','restricted','instructors') NOT NULL,  
  `owner` varchar(16) NOT NULL,  
  `notes` varchar(4096) DEFAULT NULL,  
  `sort_type` enum('none','alpha_asc','alpha_des','pub_asc','pub_des') NOT NULL,  
  `status` enum('default','deleted') NOT NULL,  
  `lastUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (`id`),  
  KEY `id` (`id`),  
  FULLTEXT KEY `name` (`name`,`description`,`works`,`owner`)  
) ENGINE=MyISAM AUTO_INCREMENT=106 DEFAULT CHARSET=utf8;
```

Figure 6: Declaration of "collections" table

It is obvious that the column "collection" in "collection_comments" is referencing a collections entry but the actual relation is not defined.

It becomes even harder to spot these relationships in the script when the column name doesn't match the referenced table name (example : "work_objects" is being referenced in "work_objects_log" but the column is named "work" rather than "work_object")

Solution : I kept track of the primary keys that we set to NULL then looped through the insert queries to find where these primary keys were referenced. I, then, saved the queries in a way that would allow us to trace the foreign key back to its corresponding primary key.

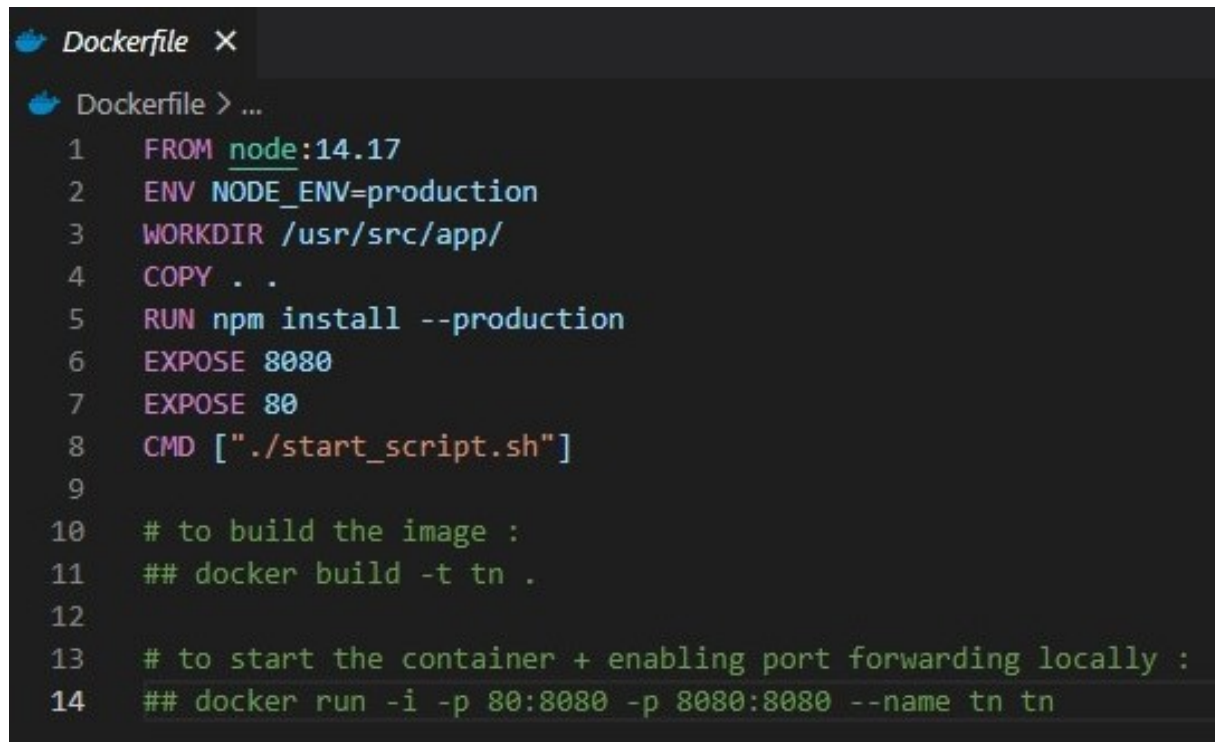
5.2.5.5 Generating the SQL queries

At this point, I had successfully generated a list of "INSERT INTO" queries that either do not have NULL primary keys or did not get referenced by other entries. The remaining issue here is to figure out how to order the remaining "INSERT INTO" queries and how to generate them with the newly assigned primary keys. My internship, unfortunately, ended before I can finish this task.

5.3 Containerizing the site

Before working on the dockerfile, I noticed that the project was lacking a "package.json" file (this file is necessary for automatically installing the dependencies) and that the "node_modules" folder contained some scripts that the old team developed. I moved those scripts to their own separate folder (called "js-scripts") since "node_modules" contents should be generated by "npm i" and should not be pushed to GitHub (meaning it shouldn't contain anything that isn't installed via npm).

After re-organizing the code, I wrote a dockerfile that exposes port 8080 since the app listens on that port and when running the container, port forwarding must be set up by linking both ports 80 and 8080 to port 80.



```
Dockerfile X
Dockerfile > ...
1 FROM node:14.17
2 ENV NODE_ENV=production
3 WORKDIR /usr/src/app/
4 COPY . .
5 RUN npm install --production
6 EXPOSE 8080
7 EXPOSE 80
8 CMD ["/start_script.sh"]
9
10 # to build the image :
11 ## docker build -t tn .
12
13 # to start the container + enabling port forwarding locally :
14 ## docker run -i -p 80:8080 -p 8080:8080 --name tn tn
```

Figure 7: Contents of dockerfile

The dockerfile uses the node:14.17 image, runs `npm i --production` to install the app's dependencies, and then calls out `start_script.sh`.

In this script, I started by installing the dependencies needed to run the Redis server (which is needed by the app to manage sessions). Then, I installed redis-server and ran it in the background. Finally, I ran the app in production mode.

I opted for running both the Redis server and the app in the same container (as opposed to writing up a docker-compose file and having Redis server on one container and the app on another and then linking them with a network) because had I chosen the other option, I would have had to change in the app's code where the Redis server is called using `localhost`. I thought it would be better and less confusing for the development to leave the app's code untouched.

6 Future work

During this month, I have managed to finish most of the work needed on the previously mentioned projects. All that is left to do is :

- Finalize the database merge script. (Project #2)
- Move the newly merged database to the container service. (Project #3)

- Deploy the container on the container service. (Project #3)

7 Consolidation of skills

I found that the skills that have been most useful to me during this internship are those below :

- Web development : This helped greatly in understanding how the old app was structured and in understanding the purpose of each file.
- Relational databases : This was necessary for understanding the database's schema and for the database merge task.
- Python
- Git
- Automation : This made a lot of tasks much easier to accomplish and saved me a lot of repetitive manual work.
- Problem solving
- Complexity of algorithms : This helped with creating optimal code for the automation scripts.

I also acquired new skills that will surely be of benefit to me in the future :

- Docker
- Familiarizing with AWS instances
- Troubleshooting network-related issues.

8 Conclusion

In this report, I attempted to give an account of the knowledge gained during this month of summer internship. This has been a wonderful learning experience as well as a great opportunity to apply the theoretical knowledge learned during my studies.

Thank you for reading.