ÇANKAYA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

CENG X00 (write “200” or “300”)

SUMMER INTERNSHIP REPORT

Name Last Name
Student ID

Company:
Name of the Company

Internship Period:
****/**/**** – **/**/******

STATEMENT OF NONPLAGIARISM

I hereby declare that all statements in this statement have been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last Name:
Date:
Signature:

ABSTRACT

During my internship at Divvy Drive, I worked primarily on Docker and Kubernetes. While I remembered Docker in the early days, I was later asked to containerize the WebApplication and WCF applications given to me by the company, first with Docker. For the first two weeks or so, I worked on preparing the Dockerfile and converting the two applications into a windows container. Over the next two weeks, I worked on the Kubernetes deployment. In the third week, I worked on kubernetes cluster installation and application deployment and set up the virtual machines. In the last week, I presented what I had done to the company as a presentation and made suggestions. I was already familiar with Docker in general, and this internship played a big role in learning and implementing Kubernetes logic.

TABLE OF CONTENTS

STATEMENT OF NONPLAGIARISM i
ABSTRACT ii
TABLE OF CONTENTS iii
LIST OF TABLES iv
LIST OF FIGURES v
1. INTRODUCTION 1
2. COMPANY INFORMATION 1
3. WORK DONE 2
3.1 X Project 2
3.1.1 Describing Code Work 5
3.1.2 Observations 5
3.2 Using References 6
3.3 Using Tables and Figures 6
3.4 Grading Criteria 7
4. CONCLUSION 9
REFERENCES 10
APPENDIX 11

LIST OF TABLES

Table 1. Technologies Used in the Report and Their Purposes 7

LIST OF FIGURES

Figure 1. Use Case of Multi Client-Server UDP Communication.. 7

1. INTRODUCTION

During my internship at Divvy Drive, I worked mostly in the DevOps department because I worked on Docker Containerization and Kubernetes deployment.

Divvy Drive is a company that offers cloud storage, data security, file sharing and backup services. The company's main areas of work focus on secure data storage, encryption, collaboration tools, automatic backup and enterprise solutions. It allows users to securely manage and share their data. The reason I chose this company was because I was curious about cloud technologies, but the company saw the phrase Docker in my CV and asked me to do containerization and distribution and present it. In short, I containerized the applications provided by the company and distributed them in the Kubernetes cluster and prepared a 70-page presentation about it. In this report, I will explain how and what I did.

2. COMPANY INFORMATION

Company Name: Divvy Drive
Address: Eskişehir Yolu 9. km No: 266 Tepe Prime Yaşam ve İş Merkezi B Blok No: 75 06800 Mustafa Kemal Mahallesi Çankaya - ANKARA - TÜRKİYE
Telephone Number: +90 (312) 287 70 01
Email Address: info@divvydrive.com

Divvy Drive is a technology company specializing in cloud storage, data security, and collaboration solutions. It provides secure file storage, encryption, and sharing services for both individuals and enterprises. The company focuses on high-level security standards, AI-driven file management, and seamless integration with enterprise workflows. Its infrastructure includes secure servers, encryption algorithms, web-based platforms, mobile applications, and API integrations. The organization consists of various departments such as R&D, Software Engineering, IT Security, and Customer Support, working together to enhance cloud-based solutions. I worked under the leadership of my internship supervisor Sercan Çayırcı. Sercan Çayırcı is a computer engineer who studied at Selçuk University in 2010-2015. He is software team lead in company, and also a full-stack developer. His e-mail is sercan_cyr@hotmail.com.

3. WORK DONE

In this internship, we will talk about the Docker and Kubernetes architecture of the two projects, their installation and the presentation I prepared. This section is divided into 2 main headings (Docker Containerization, Kubernetes Cluster) and 2 intermediate headings (WCF, Webapp) under each main heading.

3.1 Docker Containerization

Before starting the containerization process, at the very beginning of my internship, I made a few attempts to remember for a while to prepare for this process. In the first two days, I watched a few videos about Docker from the source I used to work with and tried a few .NET. Since I was working with Linux containers before this project, I had to learn the slight differences of Windows containers. Later in the process, the problems I experienced with Windows containers will be mentioned again. Afterwards, the company gave me two applications and I was asked to containerize these applications. These two applications will be explained separately.

3.1.1 WCF Service Project Containerization

First, I started working on the WCF project that was sent to me. I discussed the source codes and dependencies of the WCF (Windows Communication Foundation) Project. Since the WCF project is Windows dependent (because it only works in .NET Framework 4.x and under versions) and not suitable to run on the Linux kernel, I decided to use a Windows container. For this reason, I had the chance to learn about Windows containers, as we proceeded with different methods in Windows containers than in Linux containers.

After some research, I started with writing WCF Project's Dockerfile. As a result of this research, I tried to choose the base image I will use. When choosing a base image, I thought about choosing Windows Nano Server as the base image because I knew that the smaller the size, the easier it would be to distribute (this will be mentioned again in the kubernetes section), but I decided on Microsoft Windows-Server Core 2019 because the project required compilation and site creation. Afterwards, I wrote the site creation codes because IIS service and site were required for this project, but since I was not familiar with

IIS codes and services, it turned out to be incorrect and I had to do research. My process of writing the correct dockerfile and debugging took a little longer.

Afterwards, I debugged the dockerfile I wrote and took note of all the errors and solutions I received in the process. The errors I received while writing the Dockerfile, their reasons and solutions will be included in the final presentation I will create.

Then, I optimized the dockerfile for smaller image size and faster run. First, I separated the base-images that I compiled and ran using multi-stage build, and after compiling, I deleted the compilation image. In this way, the image size I created was significantly reduced. Afterwards, I continued the optimization by combining the RUN, COPY, ADD commands (so there will be fewer layers and your image size will be lower) and I reached the latest working and optimized dockerfile.

In the finale, I created an image from this dockerfile, ran the container, and tested that the final product was working. When I confirmed that it was working, I uploaded this image to DockerHub. You can pull eevah/basic_wcfserviceapplication:v1 to access this image. If I need to briefly explain the dockerfile code; First, it takes the Windows server core image. Then it creates the directory where this project will be copied. Then it creates the sites and opens the ports. Finally, it copies the application into it and creates the image.

3.1.2 WebApplication Project Containerization

In this project, like the other one, I had to first examine the project in order to write the dockerfile. As a result of my examination of the source codes, I determined that the project's ports and requirements were used and that the project used ASP .NET MVC. Since the project used .NET Framework 4.7.2, this project also had a Windows dependency and was not suitable for the Linux kernel.

After thoroughly understanding the project and its dependencies, I moved on to writing the dockerfile of this project. I was going to use the servercore base image in this project, but I decided to move it to a newer version and tried 2022 server core. To host this application, I used the IIS service that I had just learned and created the sites. I get some errors while creating this project, and I took notes of them all to use in my future presentation.

I did not make any obvious changes to the dockerfile except for some services, ports and base image changes used by the webapp. I created the image of this project, ran its container, and after making sure it was working properly, I uploaded it to dockerhub. You can also access the working image of this project by pulling eevah/webappl.net4.8:v1.

3.1.3 Creating Kubernetes Cluster

Afterward the containerization, the company asked me to distribute these applications by creating a Kubernetes cluster for next 2 weeks. Since I am not very familiar with Kubernetes, I first studied what the concept of Kubernetes is and how it is used. Since the distribution of both applications is the same, no extra topic will be opened.

First of all, as a result of my research, I realized that I needed to use a kubernetes cluster and this was the first thing I did. While I was setting up the virtual machines, I continued to work with Kubernetes. While creating these virtual machines, I thoroughly researched how much space and RAM I should use to be optimal.

In line with what I learned from my research, I learned that in order to set up my Kubernetes cluster, there must be at least one main and one worker node. The main node must be Ubuntu (since Kubernetes is Ubuntu based). Worker node can be Ubuntu or Windows, but since both of our applications will be distributed with Windows containers, I had to use Windows worker node. So I created two virtual machines and started the kubernetes cluster creation process. One of these virtual machines will be Ubuntu 22.04 (4GB RAM, 4 CPU and 20GB memory) and the other will be Windows Server 2022(8GB RAM, 2 CPU, 30GB memory) . I tried these virtual machines in both hyper-V and oracle virtualbox. I made sure that these virtual machines were on the same network and in bridge mode (if these conditions are not met, they do not work, I reinstalled them several times). To make sure of this, I pinged each other and checked. Then, the first thing I did was to set up my master node.

To create my master node (also called controll plane), I firstly close the swapoff. Then, IPv4 settings were made by following the instructions on Kubernetes' own website and checked it. Next, choosed and downloaded a container runtime to my Ubuntu virtual machine. For this, I chose containerd, which is the most used, I added the presentation which codes I used, got them from Docker's own site. Then, I ran some codes that were not on the site but were used to configure Kubernetes and noted them.. Afterwards, I installed and ran kubernetes connections such as Kubectl and Kubeadm one by one. Since this was one of the processes where I made the most mistakes, I touched on these a lot when creating a presentation at the end of the internship. I restarted my containerd service, checked it, got its IP adress and configured and started the main node of my Kubernetes cluster for the last time.

As the next step, I did research on creating a Windows worker node and started creating my worker node accordingly. After creating our worker node and installing windows server, I turned on Bridge mode and on the same network as the main node. Afterwards, I accessed Powershell and downloaded, installed and ran kubectl kubeadm and kubelet. Since we were using containerd in the main node, I needed to use containerd in this node as well, so I downloaded containerd, installed it and ran it. Additionally, I installed container and hyperV features. Since some of these steps gave errors, I had to rework the powershell codes. I went to the directory where we extracted containerd, found containerd and added it to my virtual machine as a service. I made my checks with the get-service code. I changed a lot of ways during the process of making them compatible and I received different errors, I noted all of these situations.

Afterwards, I needed to transfer some yaml files between the two machines. Since hyperV does not support copy-pasting, I connected the two machines to each other via TCP and sent the yaml files. As the last step of the Kubernetes cluster, I learned the IPs of both machines, pinged each other for the last time for control purposes, and then started the main node.

Finally, I bought tokens from the master node with commands. Using this and the master node up IP address, I added my worker node to the kubernetes cluster. Thus, my kubernetes cluster is ready.

3.1.4 Kubernetes Distribution

The things needed to deploy with Kubernetes and the cluster settings were researched and the lessons were followed. Some additional files were created in line with these lessons.

Deployment.yaml, service.yaml, secrets.yaml files were created for the applications using the images I pushed to Docker Hub. Then, using these manifest files, the applications were uploaded to Kubernetes with the kubectl command. Then -get commands were run for control. At some stages, I lost a lot of time in the Kubernetes section because my computer gave an insufficient RAM error, and I added them in my presentation.

Finally I containerized and distributed 2 applications with docker and kubernetes.

3.1.1 Describing Code Work

During my internship, I containerized two different applications using Docker and made them executable on Kubernetes. First, I created a Dockerfile for each application and managed its dependencies, and then uploaded these images to Docker Hub and enabled them to be used in the Kubernetes environment. I created a cluster on Kubernetes and defined separate Deployment and Service definitions for both applications. I ensured that applications ran reliably by using Kubernetes features such as autoscaling, load balancing, and fault tolerance. I ensured traffic was directed. I simply created and operated a distribution system that was more flexible and therefore more resource efficient.

3.1.2 Observations

During my internship, I had the opportunity to observe software engineering processes and methodologies used by the company. During the work process, tasks were determined by discussion in team meetings, and decisions were usually made by technical leaders or project managers. Workflows were carried out in accordance with the Agile methodology. Tasks were determined and distributed at the beginning of each sprint, and progress was monitored through daily stand-up meetings.

During the design process, architectural decisions were made through discussion among team members. Testing processes played a critical role to ensure the quality of applications.

The tools and technologies used within the company were preferred not only because they were popular, but also because they met a specific need and ensured that the system was scalable, secure and sustainable.

Observing these processes made me understand that not only technical knowledge but also skills such as intra-team communication, problem-solving ability and project management are important in software engineering. I realized that in my future career, I need to not only learn technologies, but also develop the necessary competencies to use these technologies efficiently and be a good team player.

3.2 Using References

When you are describing the work please keep in mind that you are not expected to teach a certain technology. The expected content is the thing YOU have done or noticed/observed during your internship. Please DO NOT borrow (plagiarize) any content from books, Wikipedia, or other websites directly and copy-paste it as if it is your own ideas. Not only will your internship be rejected, but also you will commit a punishable offense [1]. While directly copying from such resources is strictly disallowed you can refer to them and use material with proper CITATION. References should be provided in the IEEE citation format. Quotations in the text should be numbered like [1], [2]. You must provide at least a few references in your report. You can use books or articles as references [2]. While you can use websites and Wikipedia articles as well, please try to make sure that the information contained in them is accurate. Your references section should only contain sources you have used in your text and nothing else.

3.3 Using Tables and Figures

Tables and figures are a good way to explain a topic when used in correct places. For example, if you have programmed an application with a user interface, you can add some screenshots to describe the application. You should refer to the figure in the text (e.g., Figure 1 shows that). Always put a caption to your figure by right-clicking the image and selecting "Insert Caption." You can use the "Cross-Reference" feature of MS Word to reference your figure from the text (you can find it in the references tab in MS Word).

Table 1. Technologies Used in the Report and Their Purposes

Technology	Version	Purpose
Python	3.10	Data processing
Docker	24.0	Containerization of applications

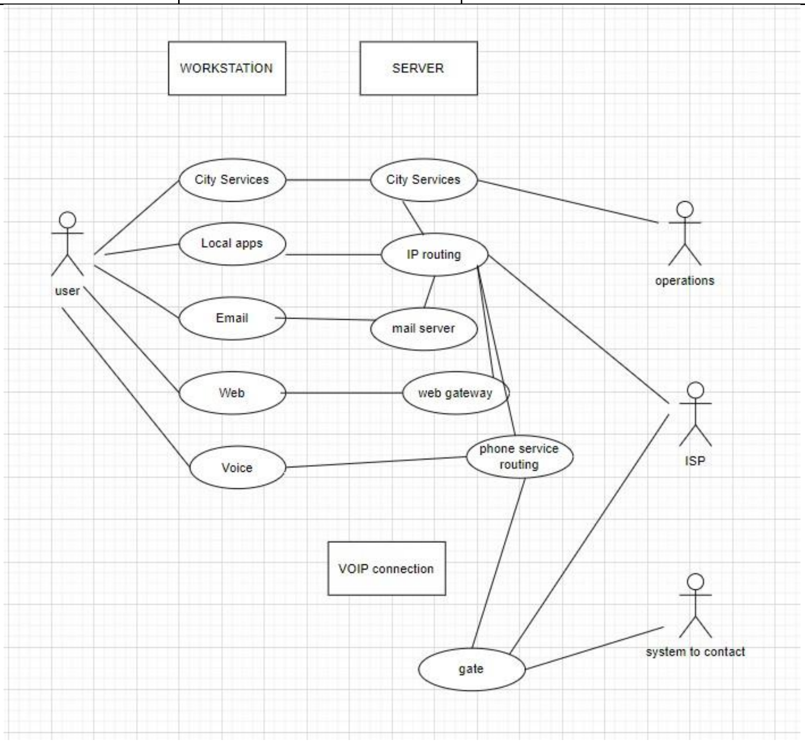

Figure 1. Use Case of Multi Client-Server UDP Communication

3.4 Grading Criteria

You should write your report according to the grading criteria. In your reports, we are expecting to see engineering qualifications. Consider the following 12 qualifications:

1. Demonstrates the ability to apply mathematics, science and engineering subjects to model and solve engineering problems.
2. Demonstrates the ability to identify, formulate and solve complex engineering problems.
3. Demonstrates the ability to select and apply appropriate analysis and modeling methods.
4. Have built a complex system, process, device, or product.
5. Have used information technologies effectively.
6. Demonstrated ability to select, devise or use modern techniques and tools.

7. Have conducted experiments, gathered data and interpreted results investigating an engineering problem.
8. Demonstrated good communication and presentation skills both orally and in writing.
9. Have independently researched and learned by educating him/herself.
10. Recognized professional and ethical responsibilities.
11. Observed and participated in business life practices such as project management, risk management and change management.
12. Demonstrated observations and knowledge about contemporary issues, global and societal effects of engineering practices.

You should write any evidence related to these criteria. Remember that you will get grades according to these criteria. You should note which pages you report relevant activities related to these qualifications so that you can write it to your "Intern Self-Evaluation Questionnaire." You should include the "Intern Self-Evaluation Questionnaire" and "Grade Form" at the end of your internship report.

4. CONCLUSION

Write a conclusion section where you summarize the work you have done. Restate your contribution, what you have learned, experienced, and acquired. Be specific in relating these to what you have learned at Çankaya University.

REFERENCES

[1] Çankaya University, “Çankaya Üniversitesi Ön Lisans ve Lisans Eğitim ve Öğretim Yönetmeliği,” 2024.

[2] R. L. Massengale, “Implementing an effective internship program,” Journal of Computing Sciences in Colleges, vol. 27, no. 5, pp. 24-31, 2012.

APPENDIX