Dockerizing web applications with Nginx unit

Kochartech

A Training Report

Submitted in partial fulfillment of the requirements for the award of degree of

Master of Computer Application

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

Submitted By:

Name of Student: Aniket Kumar

Registration Number: 12108348

To whom so ever it may concern

I, **Aniket Kumar, 12108348,** hereby declare that the work done by me on "**Dockerizing web applications with nginx**" from **August, 2022** to **December, 2022**, under the supervision of **Mr. Tarun Sharma**, **Associate Engineering Manager, Kochartech,** and **Mr. Parikshat Kumar Angra**, **Assisstant Professor,** Lovely professional University, Phagwara, Punjab, is a record of original work for the partial fulfillment of the requirements for the award of the degree, **Master of Computer Application.**

Aniket Kumar (12108348)

Signature of the student

Dated: 06-12-2022

Training Certificate from the organization

ACKNOWLDGEMENT

The internship opportunity I had with Kochartech was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me though this internship period.

Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to the Mr. Tarun Sharma, Associate Engineering Manager and Mr. Navsagardeep Singh, Junior Engineer who in spite of being extraordinarily busy with her/his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed organization and extending during the training.

I express my deepest thanks to Mr. Prikshat Kumar Angra, Assisstant Professor for taking part in useful decision & giving necessary advices and guidance and arranged all facilities to make life easier. I choose this moment to acknowledge his/her contribution gratefully.

Aniket Kumar

12108348

Date – 09-12-2022

Table of Contents

Sr. No.	Title	Page No.
1	Declaration	2
2	Training Certification from organization	3
3	Acknowledgement by student	4
4	Table of Contents	5
5	Introduction of the Company	6
6	Profile of the Problem	8
7	Existing System	10
8	Problem analysis	11
9	Software Requirement Analysis	13
10	Introduction of the Project Undertaken	14
11	Introduction to the Nginx unit	15
12	Installation of the nginx unit	17
13	Configuration of the nginx unit	18
14	Dockerizing a Java Application with nginx unit	19
15	Serving static files with nginx unit	24
16	Applying SSL certificate	26
17	Multilingual Apps in one image	27
18	Project Legacy	30
19	Offer Letter given by the company at the time of joining	31
20	Continuous Assessment (CA) for internship/ OJT	32
21	Bibliography/ References	33

Introduction of the Company

For over 15 years, KocharTech has been the industry catalyst in enhancing customer experiences through engagement. We specialize in digitally transforming business processes by designing disruptive solutions using AI and Machine Learning technologies.

Started in 2003, KocharTech has been powering some of the Fortune 500 companies, MNCs as well as Unicorn Start-ups across Asia, Africa, Europe and the Middle East. We have grown in strength and stature from less than 10 employees to 3500+ in over a decade.

From humble beginnings in Amritsar, Punjab, Kochar Tech has come a long way to establish itself as a leading Indian 'Digital Transformation' architect in AIoT, SaaS, and BPM solutions for over 20 years now. They now serve over 500 Million customers globally, surpassing 4 Billion transactions.

Creating world-class solutions that help make human lives better the company always wanted to work globally while staying local.

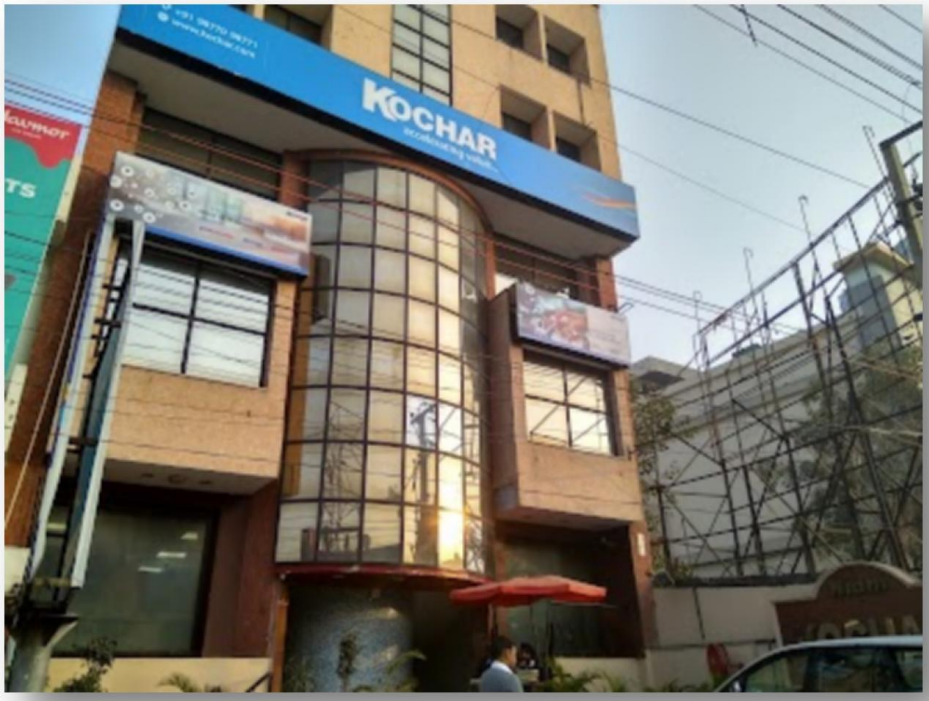
Over the last 20 years, a genuinely homegrown venture grew into an award-winning reputable global service provider delivering a myriad of tech products and managed services to Fortune 100 companies and MNCs across 25+ countries.

Their distinguished list of clientele includes Samsung, Etisalat, Vodafone, Singtel, Airtel, ICICI Bank, Emirates Transport, Cred, Cars24, Swiggy, Myntra, etc.

Today, enabling businesses to begin their 'Digital Transformation' journey with their intelligent, reliable, and scalable portfolio of digital engagement solutions that includes:

- MAXICUS – Next-generation Contact Center & Outsourcing Service Provider
- KNOWMAX - AI-based Knowledge Management Solutions

- LEAPMAX – Web & mobile platform to monitor & measure the productivity of the remote workforce.
- DEVICEMAX – Solutions for Lockdown, Remote Device Management
- IGZY – IoT Solutions & Managed Services for Enterprises and Consumers
- KOCHIVA - Upskilling Academy



With five state-of-the-art delivery centers in Amritsar, Gurgaon, Kolkata, Vadodara & Ludhiana, 5000+ employees, and a pan-India presence across central states and cities, KocharTech is exceptionally reliable and capable of providing local support to their clients anywhere in India.

Profile of the Problem

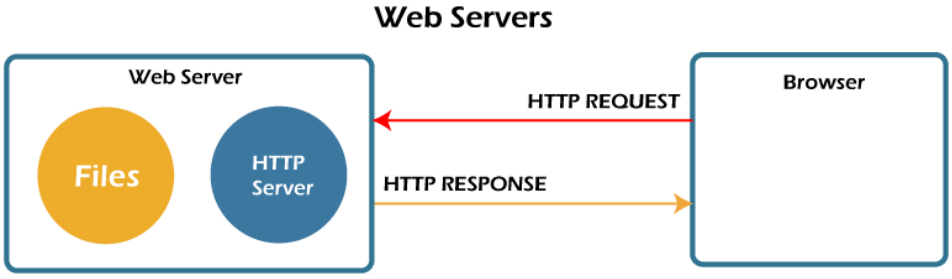
When a developer or a team of developers create any application, next thing is to deploy the application on a web server, so that it can be accessed by anyone. To achieve this, we need a web server, which can serve the application as well as handle static files, reverse proxying, applying ssl certificates, etc.

The term web server can refer to hardware or software, or both of them working together.

On the hardware side, a web server is a computer that stores web server software and a website's component files (for example, HTML documents, images, CSS stylesheets, and JavaScript files). A web server connects to the Internet and supports physical data interchange with other devices connected to the web.

On the software side, a web server includes several parts that control how web users access hosted files. At a minimum, this is an HTTP server. An HTTP server is software that understands URLs (web addresses) and HTTP (the protocol your browser uses to view webpages). An HTTP server can be accessed through the domain names of the websites it stores, and it delivers the content of these hosted websites to the end user's device.

In this project, the term web server refers to the software.



There are mainly two kinds of web servers – static and dynamic.

Static web servers refer to the servers, which serve only the static content i.e., the content is fixed and being shown as it is.

Dynamic web servers refer to the servers where the content of the page can be updated and altered.

Currently these are some popular web servers that are used to serve the applications on a server:

- **Apache HTTP Server**

  This web server is developed by Apache Software Foundation. It is an open-source, accessible web server available for almost all operating systems, including Windows, MACOS, Linux, FreeBSD, etc. Apache is one of the most popular web servers used around the globe.,
- **Microsoft Internet Information Services (IIS)**

  IIS is a high-performance web server that is developed by Microsoft only for Microsoft platforms. This webs server is tightly integrated with Microsoft operating system; therefore, it is not open-sourced.
- **Nginx**

  Nginx is an open-source web server commonly used by administrators as it supports light resource application and scalability.

Some of the problems in these web servers are:

- Not easy to configure.
- Not portable
- Server has to be taken down for maintenance

Existing System

Currently, most of the web applications are deployed on a Linux machine, which is a good thing, but the web servers they are using are old now.

Let's take Apache web server for example. Some of the Apache's main features are:

- Free and open-source
- Module-based architecture
- Easy configuration and customization
- Regular updates and security patches
- Big community of developers
- Compatibility with IPv6

Although Apache web server is also cross platform, easy to configure, but the configuration can not be modified or updated without restarting it, which makes the server down for a moment. If we try to customize the apache web server, new bugs and errors are invited.

Same goes for the nginx server. Nginx server is currently most popular and efficient web server used by most tech companies like Pinterest, Airbnb, Cloudflare and Zendesk. It has been called the new Apache for it's ability to scale more efficiently than other web servers. However, it is not an easy task to configure nginx web server.

Scope of the existing system

Currently, the nginx server is efficient enough if we:

- Do not want to further update our application
- No need to change the configuration
- No issues if server goes down while maintenance

But, if we want a reconfigurable web server with 100% uptime, nginx unit will be best option.

Problem analysis

The nginx Unit is a dynamic web application server, designed to run applications in multiple languages. Unit is lightweight, polyglot, and dynamically configured via API. The design of the server allows reconfiguration of specific application parameters as needed by the engineering or operations.

To deploy web applications, which are reconfigurable and updatable with 100% uptime, nginx unit can be used. If combined with docker, the server becomes portable, which means we can easily move our server from one system to another if needed.

Feasibility Study

The feasibility study of any system is mainly intended to study and analyse the proposed system and to decide whether the system under consideration will be viable or not after implementation.

Different types of feasibility study and the way we performed on our project are:

1. Technical Level

Since, this project is a technical project, feasibility at technical level is one of the most important factors of the feasibility study. The technologies, which we are using in this project docker and nginx unit. These technologies are easily available and can be used without any difficulties in setup. These tools are enough to display outputs in the required time.

2. Economical level

Economically, this project is very feasible as it is being developed at a very minimal cost. The main reason behind its low cost is that we are mostly using open source tools to develop it. The Operating System that we are using is Linux. Docker and nginx unit are also open source tools which are free to use. The client will have to bear some cost for the domain.

3. Operational level

At operational level, this project will provide a very easy to configure JSON file for users as well as hassle free setup for the client. No major training will be required for the user or the client. It will provide all the benefits of the present system as well as improved uptime of the server, hassle free updates and maintenance. This makes the project very feasible at operational level too.

Software Requirement Analysis

The requirement is nothing but a condition needed by the user to solve a problem or achieve an objective. It is the initial step in the development of a system. Software requirements specification lists out all the requirements stated by the user inconsistent manner. Great software can be created only from a great specification. Systems and software these days are so complex that to get on with the design before knowing what you are going to build is foolish and risky.

However, minimum requirements are listed below:

Hardware requirements

- Dual core processor
- 4GB RAM
- 128 GB HDD

Software requirements

- Linux OS
- Docker
- curl

Introduction of the Project Undertaken

- The main goal of the project is to containerize the web applications with the docker and serve them with nginx server.
- This will make the web applications portable, easy to deploy and will be running with 100% uptime.
- NGINX Unit is a dynamic web application server, designed to run applications in multiple languages. Unit is lightweight, polyglot, and dynamically configured via API. The design of the server allows reconfiguration of specific application parameters as needed by the engineering or operations.
- Unit is a lightweight and versatile open-source server that has three core capabilities:
    - it is a web server for static media assets,
    - an application server that runs code in multiple languages,
    - and an HTTP reverse proxy.
- It was built by nginx team members from scratch to be highly efficient and fully configurable at runtime

Introduction to the Nginx unit

- It is a universal tool that compresses several layers of the modern application stack into a potent, coherent solution with a focus on performance, low latency, and scalability.
- It is intended as a building block for any web architecture regardless of its complexity, from enterprise-scale deployments to your pet's homepage.
- Unit's native RESTful JSON API enables dynamic updates with zero interruptions and flexible configuration, while its out-of-the-box productivity reliably scales to production-grade workloads.
- The latest version is 1.28.0, released on September 13, 2022.

Benefits

- **Flexibility**
  - the configuration is managed in a separate JSON file, which is mounted with the container.
  - The benefit for this approach is we can change the configuration on the go, without rebuilding the image.
  - Apps in multiple languages and language versions run side by side.
- **Performance**
  - Requests are asynchronously processed in threads with efficient event loops (epoll, kqueue)
  - Syscalls and data copy operations are kept to a necessary minimum.
  - 10,000 inactive HTTP keep-alive connections take up only a few MBs of memory.

- Router and app processes rely on low-latency IPC built with lock-free queues over shared memory.
- Multithreaded request processing is supported for Java, Perl, Python, and Ruby apps.

Installation of the nginx unit

Nginx unit is available as docker image, which is portable and easy to setup. For every language module, there is a different image and also a minimal image for customizations.

Apart from the docker image, there are 3 files which is needed to run an application in a docker container with nginx unit.

- Dockerfile - for pulling unit image and additional setup
- docker-compose.yml - for building the image, mounting the volumes and setting network mode.
- config.json - configuration file for unit, tells language of the application, path to the executable file, port to run, any environments, routings if multiple applications.

Steps

- Add relevant files files to the working directory, for example: /www/
- Create a config.json file containing necessary configurations.
- Create a Dockerfile if necessary and a docker-compose.yml, in which, mention the language specific image tag, mount the volumes and the config file, set the network mode to host and give log path.
- Start the docker container.
- The configuration should be applied automatically from the config.json file.

Configuration of the nginx unit

Nginx unit provides APIs for configuration which accepts JSON data and makes it easier to configure, and the best thing is that server is always running with almost 0 downtime.

It is secured as it can only be accessed from inside the container.

API Paths available in nginx unit are:

- /config
- /certificates
- /control
- /status

Config

- The /config section of the control API handles Unit's general configuration with entities such as listeners, routes, applications, or upstreams.
- To accept requests, add a listener object in the config/listeners API section; the object's name can be:
    - A unique IP socket: 127.0.0.1:80, [::1]:8080,
    - Or, a wildcard that matches any host IPs on the port: *:80
- Unit dispatches the requests it receives to destinations referenced by listeners.
- Available listener options:
    - pass
        - Destination to which the listener passes incoming requests.
        - The value is a variable pointing to a configuration entity.
    - tls: Object; defines SSL/TLS settings.
    - forwarded: Object; configures client IP address and protocol replacement.

Sample Config File

```
{
    "listeners": {
        "*:80": {
```

```
            "pass": "applications/java_app"
        }
    },


    "applications": {
        "java_app": {
            "type": "java",
            "webapp": "sample.war",
            "working_directory": "/www/"
        }
    },
    "access_log": "/var/log/unit/access_log"
}
```

Updating the config

Just make a PUT request via curl with the updated config file and the Reconfiguration will be done successfully if everything is fine.

```
curl -X PUT --data-binary @<path/to/config/file> --unix-socket
/var/run/control.unit.sock http://localhost/config
```

Querying the config

```
curl --unix-socket /var/run/control.unit.sock http://localhost/config/
```

API Manipulation

To address the control API or parts of it, control socket should be queried over HTTP.

URI path segments of the requests to the API must be names of its JSON object members or indexes of its JSON array elements.

API can be manipulated with the following HTTP methods:

- GET Returns the entity at the request URI as a JSON value in the HTTP response body.
- POST Updates the array at the request URI, appending the JSON value from the HTTP request body.
- PUT Replaces the entity at the request URI and returns status message in the HTTP response body.
- DELETE Deletes the entity at the request URI and returns status message in the HTTP response body.

Dockerizing a Java Application with nginx unit

To dockerize a Java application with nginx unit, I have taken a sample war file of a java application. For this purpose, I have created a docker-compose.yml file, which contains these things:

- The docker image of nginx unit with java tag.
- Working directory.
- Mounting the volumes of the application directory, config file and logs.
- Setting network mode to host.

docker-compose.yml file

```
version: "3.6"

services:

  unit:

    image: nginx/unit:1.26.1-jsc11

    container_name: java_app

    working_dir: /www/

    volumes:

    - ${PWD}:/www/

    - ${PWD}/docker-entrypoint.sh:/usr/local/bin/docker-entrypoint.sh

    - ${PWD}/config.json:/docker-entrypoint.d/config.json

    - ${PWD}/logs/:/var/log/unit/

    network_mode: host

    restart: unless-stopped
```

config.json file

```
{

    "listeners": {
```

```
        "*:80": {
            "pass": "applications/java_app"
        }
    },


    "applications": {
        "java_app": {
            "type": "java",
            "webapp": "sample.war",
            "working_directory": "/www/"
        }
    },
    "access_log": "/var/log/unit/access_log"
}
```

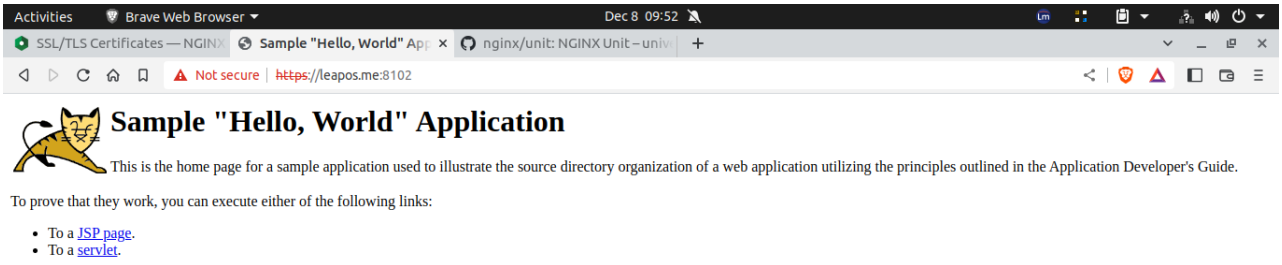
Now, we have a docker-compose.yml file, config.json and a sample.war file.

To build and start the docker container containing the nginx unit:

```
docker-compose up -d
```

After running the above command, the image will be pulled and configurations will be applied and the server will be up within a moment.

Output:


Activities
Brave Web Browser
Dec 8 09:52
SSL/TLS Certificates — NGINX
Sample "Hello, World" Ap
nginx/unit: NGINX Unit – univ
Not secure
https://leapos.me:8102
Sample "Hello, World" Application
This is the home page for a sample application used to illustrate the source directory organization of a web application utilizing the principles outlined in the Application Developer's Guide.
To prove that they work, you can execute either of the following links:
To a JSP page.
To a servlet.

Serving static files with nginx unit

Although nginx unit is a dynamic web server, it can serve static files tool.

To serve the static files, list the file paths in the *share* option of a route step action.

To serve the files, unit must be able to access them.

Options available for static files:

- share (required)
  - a variable - string or array of strings,
  - listing file paths that are tried until a file is found.
  - When no file is found, fallback is used if set
- index
  - Filename to be tried if share is a directory.
  - When no file is found, fallback is used if set.
  - The default is index.html
- fallback
  - used if the request can't be served by share or index
- types
  - Array of MIME type patterns
  - used to filter the shared files.
- chroot
  - Directory pathname for path restrictions
  - Linux kernel version should be 5.6+
- follow_symlinks, traverse_mounts
  - Booleans, turn on and off symbolic link and mount point resolution respectively;
  - If chroot is set, they only affect the insides of chroot.
  - The default for both options is true

Sample config file for serving static files

```
{

    "listeners": {
```

```
        "*:80": {
            "pass": "routes"
        }
    },


    "routes": [
        {
            "action": {
                "share": "/www/static/$uri"
            }
        }
    ]
}
```

Applying SSL certificate

Uploading ssl certificates and updating them is also an easy task in nginx unit. We just have to do a curl request and the certificate chain is uploaded to the server.

- The */certificates* section of the control API handles TLS certificates that are used with Unit's listeners.
- To set up SSL/TLS for a listener:
    - upload a .pem file with the certificate chain and private key to Unit.
    - and name the uploaded bundle in the listener's configuration

Steps

- First, create a .pem file with your certificate chain and private key: cat cert.pem ca.pem key.pem > bundle.pem
- Upload the resulting bundle file to Unit's certificate storage under a suitable name (in this case, bundle):

```
curl -X PUT --data-binary @bundle.pem --unix-socket
/var/run/control.unit.sock http://localhost/certificates/bundle
```

- Now, reference the uploaded certificate in config file under listener.
    - Add certificate name inside tls.
    - Ex: `{ "pass": "applications/php", "tls": {"certificate": "bundle1"} }`
- Lastly, update the config using curl.

Multilanguage Apps in one image

Nginx unit also provides a feature to deploy multilingual apps, *i.e.*, applications developed in multiple languages. For example, one module of an application is developed in Java and other module in Python, nginx unit can deploy them in a single container. We just have to update the config.json and make sure that the required language modules are installed in the docker image.

We have to use the following Dockerfile template that starts with the minimal Unit image based on Debian 11 and installs official language module packages:

Dockerfile

```
FROM nginx/unit:1.28.0-minimal

# We take a minimal Unit image and install language-specific modules.


# First, we install the required tooling and add Unit's repo.

RUN apt update && apt install -y curl apt-transport-https gnupg2 lsb-
release  \

        debian-archive-keyring
\

    &&  curl -o /usr/share/keyrings/nginx-keyring.gpg
\

           https://unit.nginx.org/keys/nginx-keyring.gpg
\

    && echo "deb [signed-by=/usr/share/keyrings/nginx-keyring.gpg]
\

           https://packages.nginx.org/unit/debian/ `lsb_release -cs`
unit"    \

           > /etc/apt/sources.list.d/unit.list
```

```
# Next, we install the necessary language module packages and perform
cleanup.

RUN apt update && apt install -y
\

        unit-jsc11 unit-perl unit-php unit-python2.7 unit-python3.9
unit-ruby \

    && apt remove -y curl apt-transport-https gnupg2 lsb-release
\

           debian-archive-keyring
\

    && apt autoremove --purge -y
\

    && rm -rf /var/lib/apt/lists/* /etc/apt/sources.list.d/*.list
```

Config.json file

```
{

    "listeners": {

        "*:8100": {

            "pass": "applications/php"

        },

        "*:8101": {

            "pass": "applications/java"

       },

       "*:8102": {

            "pass": "routes"

       }
```

```
    },

    "applications": {
        "php": {
            "type": "php",
            "root": "/www/"
        },
        "java": {
            "type": "java",
            "webapp": "sample.war",
            "working_directory": "/www/"
        }
    },

    "routes": [
       {
           "action": {
              "share": "/www/dist/Frontend/"
           }
      }
    ],
    "access_log": "/var/log/unit/access_log",
}
```

Project Legacy

Nginx unit is currently new in the market. Hence, there are still many bugs to be fixed and new features to be added. It will take some time to reach more number of users.

In our case, we have deployed a Java application developed with JSP. But when I tried to deploy a Spring Boot project which used Java version 1.8, it was not able to deploy. For applications built on Java version 11 or newer, there was not any problem.

Thus, nginx unit does not support legacy applications, which means we can only use it for newer applications of specific languages – Go, PHP, Python, Java, Node.js, Rupy and Perl.

Future Scope

This project has very much to do in future and if used efficiently, will be a revolutionary product. Currently, if an application has to be updated, the server needs to be taken down, but in case of nginx unit, the server is always up. This makes it unique in it's segment.

Offer Letter given by the company at the time of joining

kochartech
Accelerating Value

Date: 20th May'2022
Name: Aniket Kumar

Subject: **Internship Program**

Dear Aniket Kumar

With reference to your application, we would like to congratulate you on being selected for IT Internship with Kochar Infotech Limited. Your training is scheduled to start effective from 1st June'2022 for a period of 1 year at Nidhi-2, Lawrnce road, Amritsar. You will be given stipend of Rs 30,000 per month.

Your performance will be reviewed on monthly basis and after completion of your internship, If Company's internal team finds you suitable, you will be entitled to CTC of 10 Lakh per annum.

Your internship will include training/orientation and focus primarily on learning and gaining a deeper understanding of concepts. You shall perform all such duties as may be delegated to you by the Company and comply with all such directions as may from time to time assigned or given to you. This internship program will not create employee employer relationship between you and company during the period of internship.

Your internship will come to end without any prior notice, in case your performance and conduct is not found satisfactory.

You are expected to be sincere, diligent and regular in your internship and maintain appropriate decorum. In case of breach of any mutually agreed terms, company may choose to call off the internship program at its own will without any prior intimation or reason thereof.

Contact Person: Raman Sharma
Contact Number: 8968704010

For **Kochar Infotech Limited**

Authorized Signatory

Corporate Office : Kochar Infotech Limited, 76-B, Udyog Vihar, Phase-IV, Gurgaon – 122 001, Haryana(India) ◆ CIN : U80302PB1995PLC017324
Registered Office : Nidhi-2, Lawrence Road, Amritsar - 143 001 ◆ +91-98770 98771
info@kochartech.com
www.kochartech.com

Continuous Assessment (CA) for internship/ OJT

CONTINUOUS ASSESSMENT (CA) for INTERNSHIP/OJT

(By external Supervisor from organization)

Name of the student Aniket Kumar Registration Number 12108348

Internship Project Title (if/any): Dockerizing Web Applications with nginx unit.

Name of Organization &Address: Kochartech
Nidhi-2, Lawrence Road, Amritsar - 143001

Name of External Internship in-charge (with mobile number):
Tarun Sharma Contact No: 8588836190

S.No.	Criteria	Marks Obtained	Maximum Marks
1	Student conduct during internship	9	10
2	Punctuality and Enthusiasm	20	20
3	Technical Skill & Knowledge	15	20
4	Performance	45	50
	Total	89.	**100**

Date 06/12/2022. Authorized Signatory For Kochar Infotech Limited. Auth. Signatory

Name Tarun Jain. Designation Director - IT & Innovation.

For Kochar Infotech Limited
Company Seal
Auth. Signatory

Bibliography/ References

[1]. Nginx unit documentation - https://unit.nginx.org/

[2]. Docker documentation - https://docs.docker.com/

[3]. Nginx unit dockerhub - https://hub.docker.com/r/nginx/unit

[4]. Nginx unit github repository - https://github.com/nginx/unit