

VictoMoto: Client-Side, Server-Side and Docker

1st Given Edward Patch

Software Engineer (of BSc Year 3)

Cloud Computing

University of Wales Trinity St. Davids (of Dr. Tim Bashford and Mike Dacey)

Swansea, Wales

Student ID: 1801492

CONTENTS

I	Introduction	1
II	Deploying Applications	1
III	Virtualisation, Virtual Machines and Containers	2
IV	Development Process	2
IV-A	Tools for Development	2
IV-B	Website Design	2
IV-C	Web Application Server	2
IV-D	Web Application Service	3
V	Conclusion	3
VI	Appendices	3
VII	Terminology	3
VIII	Reference List	3

Abstract—An eCommerce web application using a node client and node service demonstrating database connections with AJAX. The web application displays a few examples to describe AJAX communication from client-side to server-side and PostgreSQL database functions and how they will work in an eCommerce web environment. Docker containers within the paper show how web applications are set up in a virtual environment with ease, allowing the business to change to different clouds.

Index Terms—AJAX, Client, Service, Docker

I. INTRODUCTION

Within this paper, an eCommerce web application to feature the motorcycle dealership ‘VictoMoto’ that needs a web application to showcase the products and increase stocks. The web application requires a product list, basket and a way to handle enquiries as it is not common for dealerships to sell motorcycles online.

This paper covers how the deployment of the web application in a Docker Container using docker-compose works and explains the formalities of the container functions. Virtualisation, Virtual Machines and Containers are then compared to form an understanding of the differences of each technology and the benefits and drawbacks.

The development process of the VictoMoto web application will describe any techniques, problems and solutions the

developer experienced on the way. Both client and web service convey what took place within the making of VictoMoto’s web application.

II. DEPLOYING APPLICATIONS

Deploying all applications to the cloud may seem beneficial. However, it can cost time and management. Cloud hosting costs vary based on the application’s services and can prove very costly from traditional web services. For example, if the website had HTML content with some JavaScript client-side functionality, a website host like GoDaddy would be more efficient than cloud hosting. Web applications enable the developer to create a more dynamic approach to embed the application within the web browser, TV, Mobile and Console applications or offline software, for example, Electron Framework. This type of application would suit an application that requires multiple services like the web service, web server, and any extra databases or even heavy-duty applications require more resources on the server-side to render the page.

Creating backup scripts that deploy an application from a Docker container or other methods is possible. A few reasons to set up automation deployments are as follows:

- A script that pings the web application detects any high ping or no ping; it may be best to redeploy the application under another location as Amazon AWS with automation offers different servers, and pricing varies.
- Cloud automation offers tools to only deploy under conditions specific to the condition of the application if it passes things like unit tests, versioning, versioning.
- A suggestion of automation, dependent on the services and purpose of the application, is a script that checks popular cloud companies like Microsoft Azure, Google Cloud and Amazon AWS, is to check the pricing and automate the redeployment to the cheapest service at the time based on the usage. An excellent example is a service with higher traffic on certain seasons than others.

The flexibility of cloud services means no ties with hosting companies and paying for what the web application uses rather than a fixed number. This feature is ideal for ‘testing the water’ for an idea that may or may not work and save money based on traffic and runtime

III. VIRTUALISATION, VIRTUAL MACHINES AND CONTAINERS

Virtualisation enables the system to run on top of another system. This process is slower than using the existing system. For example, Windows Sub Linux (WSL) is a slower experience than running a Linux system. However, in some cases, Virtualisation does pay off. The idea of WSL is to create a virtual environment to allow the compatibility of applications such as Docker,

Virtual Machines are similar to Virtualisation, except rather than offering a lightweight virtual environment, the virtual machine runs any desired operating system. Both methods share physical memory with the host operating system. However, both Graphical User Interface (GUI) and Command Line Interface (CLI) live within the virtual machine. Whereas, Virtualisation relies on a CLI.

Containers is a handy environment as with Docker installed on a Linux operating system, a Docker script creates containers that install different services and software to run the software. This idea means a complicated web application can move from cloud to cloud with a simple execution. Running services on a Virtual Machine or from a Virtualisation software is possible, but compatibility may change dependent on the Linux that is running.

Each method provides meaningful methods. A few examples that existing software already promotes to the average user:-

- Virtualisation - ChromeOS uses a virtual environment to install Linux. This way gives the user the ability to install Linux applications rather than just native ChromeOS applications. **Note** - Windows 11 has a virtual environment of both Linux and Android, meaning that Linux and Android applications are compatible within the Windows 11 operating system.
- Virtual Machine - Bluestacks is software that enables an Android operating system. Many individuals use this software to install Android applications to their Windows operating system.
- Containers - Microsoft Azure Container Registry is an alternative to Docker containers.

IV. DEVELOPMENT PROCESS

VictoMoto Web Application Note: Deployment of VictoMoto includes a database SQL script and a staff login of **username: adminWM** and **password: notApassword101**.

Never store user credentials without a form of encryption!

Ports: Web Server: 1500 — Web Service: 3000

A. Tools for Development

The following tools to create this product are as recorded:-

ChromeOS - Debian 10 (Virtual Environment), Docker, Docker Compose, Postman, NodeJS and Visual Studio IDE.

B. Website Design

The website design uses Bootstrap 4.5 and a custom CSS template. An image of the view displays the design of HTML and CSS before executing JavaScript code. The subtle animations, Bootstrap containers and styling techniques stand throughout the VictoMoto web application. Functionality for things like Application Views, the Staff Login navigation and active feature for the navbar relies on the JavaScript element to function dynamically rather than statically.

Figure 1 shows the custom stylesheet and how the view works.

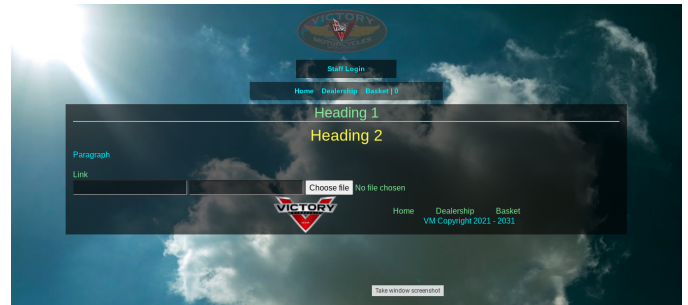


Fig. 1. View of the HTML pages

C. Web Application Server

Node modules required to run the service:-

CORS, Express, Express-Validator, Nodemon, PG and PG-Format.

File structure of the server, limited to depth of two:-

```
- public
  - assets
  - forms
  basket.html
  dealership.html
  index.html
- src
  - Cache
  - Tools
  navbar-sel.js
  runtime.js
- views
  - admin
  - consumer
index.js
```

Under 'src/runtime.js' contains logic that keeps track of the navigational, login and basket statuses. This script runs on each page as the user navigates through.

View active navigation work by comparing the page title to the *nav-link* ID. If there is a similarity, then it would set the *active* class to the current page. This logic under 'src/navbar-sel.js'. Each page includes the JavaScript view that corresponds to the page. Rather than having a one-page website, instead of routing, this method calls JavaScript

allowing the structure to stay loaded. Content is added within the JavaScript functionality found in the 'views/' directory.

Login view contacts the web service to query the database to log on. When the user signs in, the staff member can click on the Dealership page, which changes the view from what the consumer sees to an administration place to add, remove, or see the consumer view on the Dealership view. This view demonstrates how views are more useful with dynamic touches than constant static influences. To store the Login state, the *sessionStorage* in-built function is available.

Home view requests product data from the web service to list the products in a category view. If the user clicks an item, an introductory display will show, listing each product. From here, the consumer can add or remove items to VictoMoto's Enquiry Basket. The text will change from 'Add to Basket' to 'Remove from Basket', vice versa, dependent on what is in the Basket, whilst updating the Basket navigation text with the runtime script.

To keep track of the Basket status of what was added or removed from the basket, certain information like ID, make, and model was passed to the *sessionStorage* to save pointless calls to the database rather than storing an ID in the cache and recalling another query.

Note: Inquiry code follows this style:-

d - Dealership ID, **i** - Bike ID, **m** - make and **o** - model.

Basket view has three important features that benefits Vic-toMoto. These are a inquiry code which uses *d-imo* layout.

D. Web Application Service

Node modules required to run the client-side:-

CORS, Express, Express-Validator, Nodemon, PG and PG-Format.

File structure of the server, limited to depth of two:-

```
- src
    dbcomms.js
    queryhandler.js
    service.js
```

The server's purpose is to await communication from the service, before executing a series of commands then returning the web service information back, in this case, of a form of JSON. The server has five routes.

- Method: GET — Route: / — Action: Returns Welcome.
- Method: POST — Route: /vm-slogin — Asynchronous Action:
Queries 'tblStaff' table and checks if the results is equal exactly to one. If it is then it will return the JSON 'name: firstName + lastName, session: true'. Otherwise it returns, '"session": false'.
- Method: POST — Route: /query/vm — Action: Lists products.
- Method: DELETE — Route: /query/vm-remove — Action: Removes Products.
- Method: POST — Route: /query/vm-insert — Action: Inserts Products.

V. CONCLUSION

After setting up the web application, it is essential to note that the application is ready to run on any system or cloud service with Docker support. Use of Virtualisation of ChromeOS Linux Environment and Containers with Docker from the tools described. NodeJS tools were installed to the virtual environment within ChromeOS to test the product in a development state to create an ideal test environment. A series of tests were conducted on both Windows and ChromeOS systems with different monitor resolutions to ensure the website worked effectively during the testing period.

ACKNOWLEDGMENT

The logo is from the Victory Motorcycles brand-the company's based in Medina, Minnesota, United States, [Victory Motorcycle](#).

Disclaimer: Inspiration from Victory Motorcycles helps create this web application for education purposes and intentions only.

VI. APPENDICES

Video Link

VII. TERMINOLOGY

List of terminologies used in this document:-

- AJAX - Asynchronous JavaScript and XML.
- WSL - Windows Sub Linux.
- GUI - Graphical User Interface.
- CLI - Command-Line Interface.
- HTML - HyperText Transfer Protocol.
- CSS - Cascading StyleSheets.
- IDE - Integrated Development Environment.
- JSON - JavaScript Object Notation.

VIII. REFERENCE LIST

- [1] M. Portnoy, *Virtualization essentials*, second edition. ed. Indianapolis, Indiana: Sybex, 2016.
- [2] G. A. A. Santana, *Data center virtualization fundamentals*, ser. Data centre virtualization fundamentals. Indianapolis, Ind: Cisco Press, 2014.
- [3] M. Skulysh and O. Klimovych, "Approach to virtualization of Evolved Packet Core Network Functions," in *The Experience of Designing and Application of CAD Systems in Microelectronics*, Feb. 2015, pp. 193–195.
- [4] linkedincom and Posey Brien, "Hosting Virtual Machines in a Multicloud Environment." Carpinteria, CA, 2018, book Title: Hosting Virtual Machines in a Multicloud Environment.
- [5] Savill John, *Mastering Windows Server 2016 Hyper-V*. Indianapolis, Indiana: Sybex, 2017, book Title: Mastering Windows Server 2016 Hyper-V.
- [6] I. Kim, T. Kim, and Y. I. Eom, "NHVM: Design and Implementation of Linux Server Virtual Machine Using Hybrid Virtualization Technology," in *2010 International Conference on Computational Science and Its Applications*, Mar. 2010, pp. 171–175.
- [7] R. Anand, S. Sarswathi, and R. Regan, "Security issues in virtualization environment," in *2012 International Conference*

on Radar, Communication and Computing (ICRCC), Dec. 2012, pp. 254–256.

- [8] Matthias Karl and Kane Sean P, *Docker: up & running : shipping reliable containers in poduction.*, second edition. ed. Beijing: O'Reilly, 2018, book Title: Docker : up & running : shipping reliable containers in poduction.
- [9] N. Zhao, V. Tarasov, H. Albahar, A. Anwar, L. Rupprecht, D. Skourtis, A. K. Paul, K. Chen, and A. R. Butt, "Large-Scale Analysis of Docker Images and Performance Implications for Container Storage Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 918–930, Apr. 2021, conference Name: IEEE Transactions on Parallel and Distributed Systems.
- [10] C.-M. Iurian, I.-A. Ivanciu, and V. Dobrota, "Couchbase Server in Microsoft Azure Cloud: A Docker Container Approach," in *2020 International Symposium on Electronics and Telecommunications (ISETC)*, Nov. 2020, pp. 1–4, iSSN: 2475-7861.
- [11] J. Ma, B. An, D. Cao, and X. Chen, "Comparing Container-Based Microservices and Workspace as a Service: Which One to Choose?" in *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, Mar. 2018, pp. 240–245.
- [12] D. Balouek-Thomert, E. Caron, P. Gallard, and L. Lefèvre, "Nuge: A container-based cloud computing service federation," *Concurrency and computation*, vol. 29, no. 11, pp. e4049–n/a, 2017, place: Chichester, UK Publisher: John Wiley & Sons, Ltd.
- [13] P. Dutta, T. Mukherjee, V. G. Hegde, and S. Gujar, "C-Cloud: A Cost-Efficient Reliable Cloud of Surplus Computing Resources," in *2014 IEEE 7th International Conference on Cloud Computing*, Jun. 2014, pp. 986–987, iSSN: 2159-6190.
- [14] R. Rauscher and R. Acharya, "Virtual Machine Placement in Predictable Computing Clouds," in *2014 IEEE 7th International Conference on Cloud Computing*, Jun. 2014, pp. 975–976, iSSN: 2159-6190.
- [15] D. Reis, B. Piedade, F. F. Correia, J. P. Dias, and A. Aguiar, "Developing Docker and Docker-Compose Specifications: A Developers' Survey," *IEEE Access*, vol. 10, pp. 2318–2329, 2022, conference Name: IEEE Access.
- [16] A. Krasnov, R. R. Maiti, and D. M. Wilborne, "Data Storage Security in Docker," in *2020 SoutheastCon*, Mar. 2020, pp. 1–1, iSSN: 1558-058X.
- [17] M. Al-Rakhani, M. Alsahli, M. M. Hassan, A. Alamri, A. Guerrieri, and G. Fortino, "Cost Efficient Edge Intelligence Framework Using Docker Containers," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2018, pp. 800–807.