

# Cloud Web Server with Automation

A detailed report on hosting a web server in a container on a cloud with build automation

Student:

Srikanth Shilesh Pasam (10387794)

Teacher:

Obinna Izima

Course:

Network and System Administration (B9IS121) CA 1

# Index

<i>Introduction .....</i>	<i>4</i>
<i>Background.....</i>	<i>4</i>
Virtualization .....	4
Cloud Computing .....	5
Containers.....	6
Automation .....	6
DNS .....	7
<i>Project Overview.....</i>	<i>7</i>
<i>Technical Description .....</i>	<i>8</i>
Azure VM .....	8
Docker Container .....	9
Jenkins.....	10
GitHub .....	11
Domain Registration .....	11
Route 53 .....	11
<i>Testing.....</i>	<i>12</i>
<i>Conclusion.....</i>	<i>12</i>
<i>Proof of Concept.....</i>	<i>13</i>
<i>Reflection.....</i>	<i>17</i>
<i>Bibliography .....</i>	<i>18</i>
<i>Appendix 1 .....</i>	<i>19</i>
<i>Appendix 2 .....</i>	<i>20</i>

# List of Illustrations

<i>Fig.</i>	<i>Description</i>	<i>Page no.</i>
1	<i>The Project Overview Flow Chart</i>	7
2	<i>List of Technologies Used</i>	8
3	<i>Setting up VM on Azure</i>	13
4	<i>Establishing Connection with VM from Terminal</i>	13
5	<i>Configured Apache httpd Server</i>	14
6	<i>Installed Jenkins and Mapped it to Port 8080</i>	14
7	<i>Jenkins successfully builds source code files for the first time after being pushed into GitHub</i>	14
8	<i>Website accessed using IP address after the first build by Jenkins</i>	15
9	<i>Jenkins able to automatically build after every update of the website</i>	15
10	<i>Successfully purchased a domain</i>	15
11	<i>Linked the domain name to the IP address of the VM</i>	16
12	<i>Website being accessed using the domain name</i>	16

# Introduction

The project aims to deploy a web site on a server in the cloud with build automation support. The concept of virtualization used by running a server in the cloud with a container installed in it to host the web files. Along with this container on the server, there is also an automation tool. The automation tool will enable in keeping the website updated on the server by syncing with the source code management application. A domain was purchased to let users access the website, and the domain servers configured to route the internet traffic to the website hosted remotely in the cloud.

The project's design integrates multiple networking and system administration technologies from different service providers to create one interdependent system. The primary reason for this approach is to demonstrate the skills developed, which can only be highlighted by manually configuring each system to work in an interdependent manner rather than using a pre-built or configured system.

## Background

This project utilizes technologies based on virtualization- cloud and containers, automation, and DNS. These systems work together to host the website on a cloud server, which stays updated automatically and is accessible to the users from across the world through the domain name. Let us understand the various technologies used in more detail before heading into the technical specifications of it.

### Virtualization

Virtualization is the process in which a system's hardware resources get divided into multiple instances, with each instance running its version of the environment. The Hypervisor is the one responsible for this. A Hypervisor divides the systems hardware or operating system from the resources running on it. The Hypervisor manages the allocation of resources like memory, data access, storage, networking, and security to each of the instances running within it. In this way, a single system or server can have multiple virtual machines running within it. Virtualization technology lets utilize systems resources to a very high degree of efficiency (*What is virtualization?*, no date).

Virtualization itself has five categories based on specific needs. They are:

- Data Virtualization
- Desktop Virtualization
- Server Virtualization
- Operating System Virtualization
- Network Virtualization

This project uses the concept of Server Virtualization. In this, each server has multiple virtual machines running within it, each with its operating system and hardware resources allocated to it. So, using virtualization, we can host the web files on a virtual machine in a server instead of locally, thus reducing cost and energy.

## Cloud Computing

Cloud computing is the process of utilizing other computer hardware and software resources for our needs. Cloud computing services include computing resources and data storage. There are three ways of using them. The flexibility and responsibility of each service differ. They are:

- IaaS – Infrastructure as a Service
- PaaS – Platform as a Service
- SaaS – Software as a Service

The IaaS service provides the highest flexibility where we maintain the server's hardware and software as we deem necessary. The service provider provides only the equipment and infrastructure.

The PaaS service provides the server along with its operating system and maintenance. We can focus on developing, testing, and deploying our application.

The SaaS service only lets us work with software in the cloud. The service provider manages everything else.

This project utilizes IaaS service from the cloud service provider. This service lets us create a virtual machine on the cloud server. We can choose the operating system to run on this virtual machine, the memory required for computation, data storage, and network security features.

Using a virtual machine on the cloud to act as the web server has the following benefits:

- Scalability – Depending on the requirements and demand, we can scale up or scale down the computational power and data storage capacity of the virtual machine or increase or decrease the number of virtual machines almost instantaneously.

- Cost-effectiveness – The most significant advantage of using cloud services is cost-effectiveness. We do not need to invest in purchasing, maintaining, or updating the server resources. All this is taken care of by the service provider.
- Reliability – Cloud offers features like data recovery, disaster management, and data replication to keep the data safe.
- Security – Cloud provides with physical and digital security of the resources.
- Global – Using clouds redundant data availability feature, we can make the website globally available with low response times for the users.

*(Benefits of cloud computing - Learn, no date)*

## Containers

Containers are a standardized package of software that can run an application. The container has all the files and packages necessary to run the application. This application can then be run on any machine within the container without having to worry about setting up the necessary environment for it again.

Containers are similar to virtual machines except where a virtual machine virtualizes the hardware, containers virtualize the operating system of the machine (Chamberlain, 2018).

Containers benefit this project in the following ways:

- They are very portable, meaning; the website developed on the local system using a container will function the same even when deployed in the cloud.
- Containers are lightweight, considering that they only have the bare minimum resources necessary in order to run the application. Being highly portable makes them very efficient in running even on a virtual machine with low compute power and storage.
- Since containers are efficient and lightweight, we can scale them quickly and efficiently as and when required.

## Automation

Build Automation refers to the process of automating specific tasks we need to do. It usually includes compiling source code into binary code and running tests, which, if successful, will then lead to packaging and even deployment of the code (Bergmann, 2011 p.1).

Automation tools help in keeping the website updated on the server as we keep rolling out updates for it. It reduces the hassle of manually updating the source code files in the webserver each time we make changes, thus saving time and letting us focus on application development.

## DNS

Domain Name System is the process that translates web address names into their unique IP addresses. Thirteen dedicated servers do this process called root servers, which are around the world (*IANA — Root Servers*, no date).

DNS is an essential part of the TCP/IP protocol based on which the internet functions. On entering a website name, the browser looks up for its corresponding IP address. If the IP address is unavailable within the system, then the request is forwarded to a resolver. If the address is unavailable here then, the request gets sent to one of the root servers. From here, the root server directs it to the TLD – Top Level Domain server based on the domain of the requested web address. Finally, the Authoritative Name Servers gets the request. They are the final level and store all information on a particular domain. The Authoritative Name Servers respond with the IP address of the requested website (Mockapetris, 1987).

DNS servers help route the website traffic from across the world to the webserver hosted in the cloud. DNS makes it easier for users to access the website just by knowing the domain name rather than the IP address of the webserver.

## Project Overview

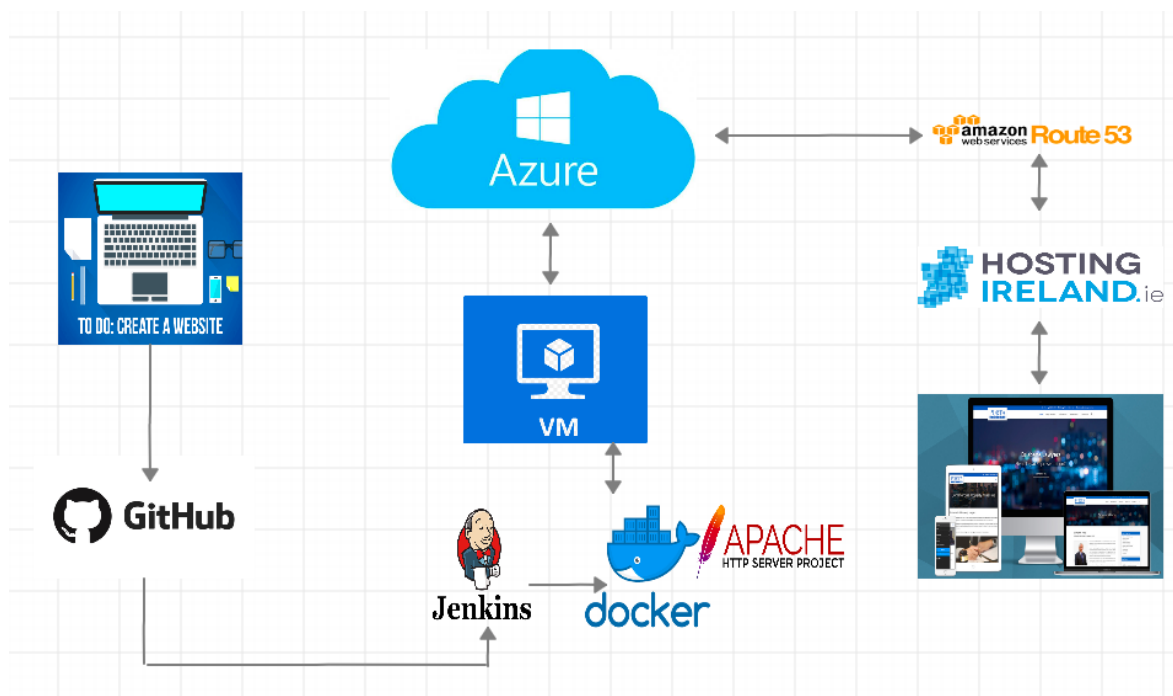


Fig.1 - The Project Overview Flow Chart

# Technical Description

The below table lists the service providers and the corresponding technology used from them for demonstrating the proof of concept:

No.	Service Provider	Technology	Version/Category
1	Cloud Platform	Azure	Virtual Machine
2	Container	Docker	Apache httpd
3	Build Automation	Jenkins	Version 2.190.2 (LTS)
4	Source Code Management	GitHub	
5	Domain Registrar	Hosting Ireland	' .com ' domain
6	DNS Name Servers	AWS	Route 53

*Fig.2 – List of Technologies Used*

## Azure VM

We start building the system by creating a Virtual Machine in the Microsoft Azure Cloud platform. We can opt for Standard B1ms, which has one virtual CPU and 2GB of memory. The B-series of virtual machines offer base-level machines that can burst CPU performance levels by 100% when required. They are on the lower end of the scale in terms of computing power and are ideal for small scale applications like low capacity web servers, proof of concepts, and development environments (Ayshakeen, no date).

In this VM, we choose Ubuntu 18.04 (LTS) OS. The region I selected was West Europe to reduce network latency and did not choose any data redundancy option. There are two security options related to how we can access and authenticate with the system, SSH key and private password. We will use the password authentication option.

We also need to enable ports HTTP (80), HTTPS (443), SSH (22), and TCP/IP (8080) on the VM.

Once the system is running, it was assigned a private and public IP address by Azure. We need to make the public IP address static so that it remains constant even after rebooting the VM. The static IP makes it easier for configuring the other systems in this project.

We can now access the VM using the terminal on our PC or using the Cloud Shell feature on Azure. In order to access the VM, we need to enter the following command in the terminal:

```
(base) Srikanths-MacBook-Air:~ srikanthshileshpasam$ ssh srikanthshileshpasam@104.40.231.66
```

The command is of the format: username @ public IP address of the VM.



## Docker Container

We now have to install Docker Container on the VM. We can do this using the CLI after logging in to the VM.

```
srikanthshileshpasam@MyUbuntuVM:~$ sudo apt install docker.io
```

The sudo keyword gives admin privileges over the Ubuntu OS. The apt keyword means the Advanced Packaging Tool, which lets us install new software packages.

After installing Docker, we need to modify it to start during system startup automatically using the command:

```
srikanthshileshpasam@MyUbuntuVM:~$ sudo systemctl start docker
srikanthshileshpasam@MyUbuntuVM:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
srikanthshileshpasam@MyUbuntuVM:~$
```

Systemctl is the keyword for controlling the system settings. First, we start the Docker service, followed by enabling it to run at startup automatically (*Post-installation steps for Linux*, 2019).

The next step I did was to install the Docker image httpd, which is the official Docker image for the Apache HTTP server. We need to run the command:

```
$ docker pull httpd
```

The command installed the Apache server Docker image in the VM (*httpd - Docker Hub*, no date).

Finally, we need to configure port 80 of the VM to be accessed by the Apache server in Docker by:

```
srikanthshileshpasam@MyUbuntuVM:~$ sudo docker run -p 80:80 httpd
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Wed Nov 13 17:53:32.389655 2019] [mpm_event:notice] [pid 1:tid 140324990891136] AH00489: Apache/2.4.41 (Unix) configured -- resuming normal operations
[Wed Nov 13 17:53:32.392253 2019] [core:notice] [pid 1:tid 140324990891136] AH00094: Command line: 'httpd -D FOREGROUND'
```

The 80:80 means I have connected my VM's port 80 to the Docker containers port 80, which runs the Apache. We can check if this works by visiting the IP address of the VM on a browser. On doing so, we can see a message on the browser which read 'It Works!'. The Apache server sent this message from the VM.

## Jenkins

We now move on to setting up a build automation tool in the VM. We can use Jenkins for this as it is an open-source automation server. The download command:

```
srikanthshileshpasam@MyUbuntuVM:~$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
OK
```

This command added the key to the repository package where Jenkins is available. Followed by that we will add the packages repository address to the VM's list using:

```
srikanthshileshpasam@MyUbuntuVM:~$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

The command enables the Jenkins packages on the VM to be updated automatically with the latest features.

```
srikanthshileshpasam@MyUbuntuVM:~$ sudo apt install jenkins
```

By this command, we can install Jenkins (*Debian Repository for Jenkins*, no date).

Now we will start the application on the VM by:

```
srikanthshileshpasam@MyUbuntuVM:~$ sudo systemctl start jenkins
srikanthshileshpasam@MyUbuntuVM:~$
```

Next, we will configure the Jenkins application. For this, open a web browser and enter the VM's IP address followed by the port 8080 like this:

`http://104.40.231.66:8080`

The address will lead to the page where we need to unlock Jenkins on the VM. We can access the password in the VM at the folder location mentioned. We can use the nano command to open the file like this:

```
srikanthshileshpasam@MyUbuntuVM:~$ cd /var/lib/jenkins
srikanthshileshpasam@MyUbuntuVM:/var/lib/jenkins$ ls
config.xml          jenkins.telemetry.Correlator.xml  nodes          secrets
hudson.model.UpdateCenter.xml  jobs          plugins        updates
identity.key.enc      logs          secret.key     userContent
jenkins.install.UpgradeWizard.state  nodeMonitors.xml  secret.key.not-so-secret  users
srikanthshileshpasam@MyUbuntuVM:/var/lib/jenkins$ sudo nano secrets/initialAdminPassword
```

After submitting the password and unlocking Jenkins successfully, we will install it with the suggested plugins, create an account and save the instance configuration of the Jenkins URL.

The next part is to create a job in Jenkins. The 'Create Job' is the place where we can configure Jenkins to pull the source code from the GitHub and build it in the Apache container. We need to select Git under the source code management section and paste the URL of the folder location of the source code files. Select 'GitHub Hook' as the build trigger.

Finally, under the 'Build' tab 'Execute Shell', we will give the code (see Appendix 1) for it to build a new container after deleting the old one every time a new trigger occurs. Once built, the source code files are then moved from the Jenkins folder to the Apache container folder.

## GitHub

We need to allow GitHub to let Jenkins access the source code files. For this, we can go to the settings tab of the source code folder and clicked on the 'Webhooks' category. We will add a new webhook by providing the URL of port Jenkins is available on the VM. Once any commits take place in the repository, Jenkins will build the files after getting triggered using the webhook from GitHub. We can check the build status on the Jenkins homepage.

## Domain Registration

The website is successfully hosted on the webserver now, but it is difficult for users to reach it as it only has an IP address and a lengthy DNS name given by Azure. To make it easier for users, we will need a more straightforward domain name for the website. For this, we can buy a domain with a domain registrar. The website purchased for this project is: srikanthshileshpasam.com

## Route 53

The final stage of the project is to link the domain name to the webserver using DNS. The DNS servers will help route the traffic of the website to the webserver in the VM. We can use Route 53 of AWS for this. In Route 53 section, we need to select 'Create Hosted Zone' and enter the domain name. Now, Route 53 provides a list of their name servers. Here we will add two more record sheets, one with the name tag left blank and the other with a '\*'. In both of them, we need to fill the value space with the IP address of the VM.

Finally, we will copy the AWS name servers and head over to the domain registrar site. Under the 'Name Servers' tab, paste the list of AWS name servers.

With this, we have configured all the systems successfully.

# Testing

To test if the project is working, we need to visit the domain purchased to check if the website is loading in it. We can cross-check the build automation functionality by pushing changes to the Git repository locally and verify if they are reflecting in the website on the internet.

## Conclusion

To briefly summarize the functioning of the system, we have developed a website on the PC and pushed the updates to a Git repository. The Git webhooks these files to the Jenkins tool.

We have a virtual machine in Azure. In this VM, we installed a Docker container, and Jenkins. The Docker container has an image of the Apache httpd server. Apache server connects using port 80, whereas Jenkins connects with port 8080 of the VM. Whenever Jenkins receives a push from GitHub, it triggers a build. Jenkins is automated to delete the existing container hosting the web files and create a new one with the updated files.

Users from anywhere in the world can access the website. After the users enter the website domain name, the network traffic flows to the domain registrar. Here, the domain registrar redirects the traffic to the AWS name servers. The AWS name servers route the traffic to the IP of the VM in Azure, and the web traffic arrives at its destination successfully.

# Proof of Concept

The following screenshots provide the proof of concept of the system developed:

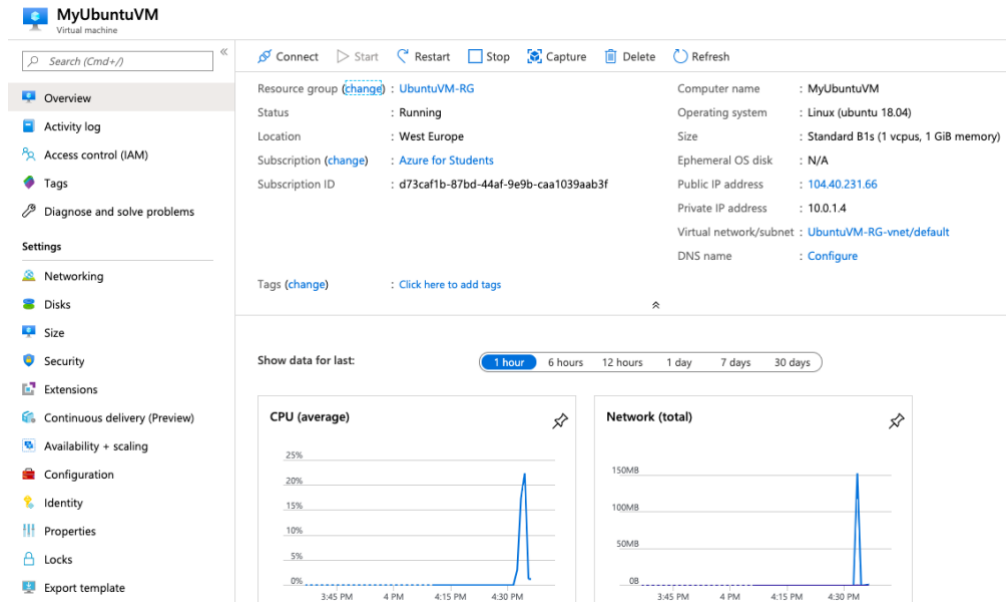


Fig.3 - Setting up VM on Azure

```
(base) Srikanths-MacBook-Air:~ srikanthshileshpasam$ ssh srikanthshileshpasam@104.40.231.66
The authenticity of host '104.40.231.66 (104.40.231.66)' can't be established.
ECDSA key fingerprint is SHA256:jjiuBrDntt1NQrCm4RGGbo70TBdH6CzGqsoAzma8eQU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '104.40.231.66' (ECDSA) to the list of known hosts.
srikanthshileshpasam@104.40.231.66's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-1025-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Nov 13 16:48:53 UTC 2019

System load:  0.0      Processes:    113
Usage of /:   5.5% of 28.90GB   Users logged in:  0
Memory usage: 46%      IP address for eth0: 10.0.1.4
Swap usage:   0%

0 packages can be updated.
0 updates are security updates.

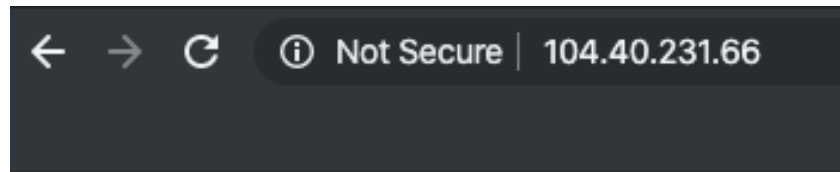
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

srikanthshileshpasam@MyUbuntuVM:~$
```

Fig.4 - Established connection with VM from Terminal



# It works!

Fig.5 - Configured Apache httpd Server

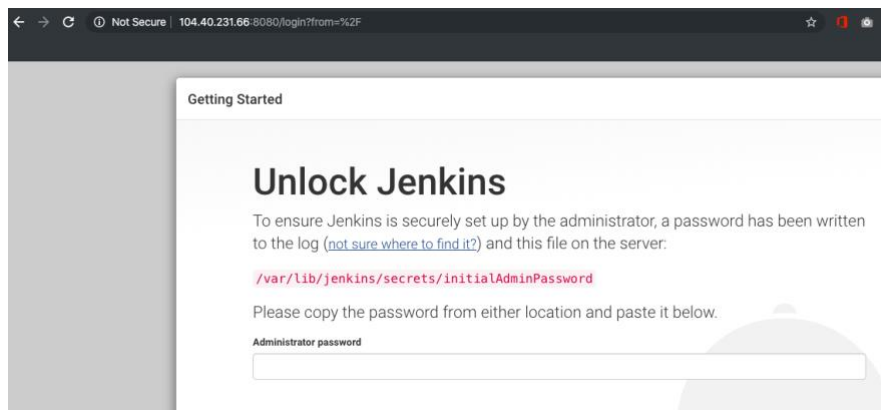


Fig.6 - Installed Jenkins and mapped it to port 8080

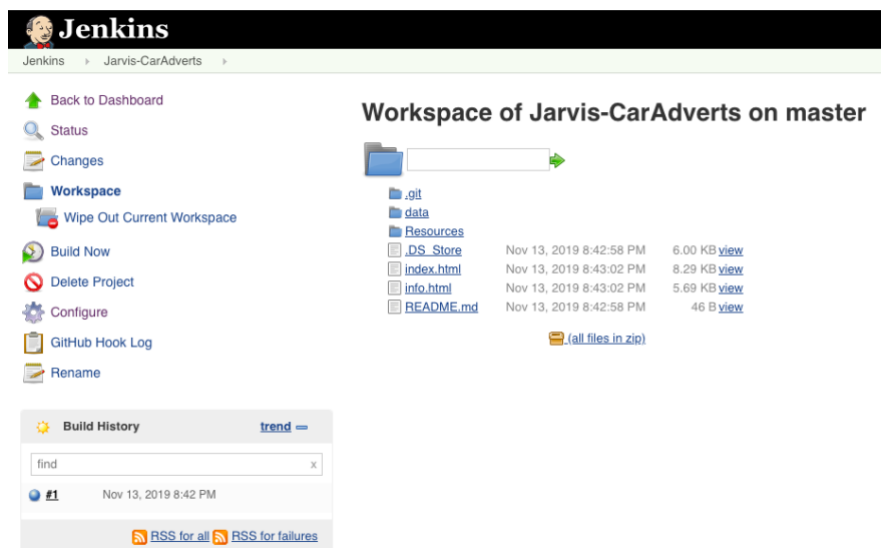


Fig.7 - Jenkins successfully builds source code files for the first time after being pushed into GitHub

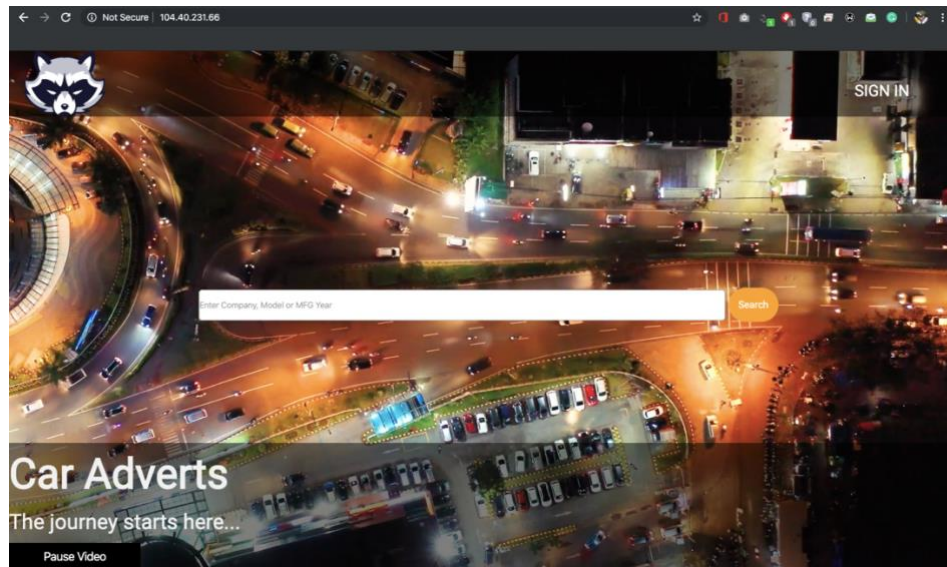


Fig.8 - Website accessed using IP address after the first build by Jenkins

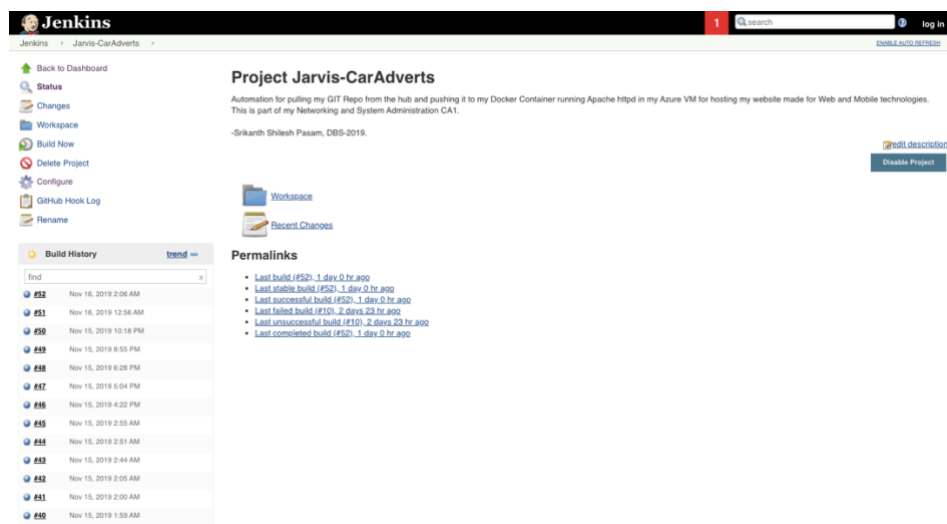


Fig.9 - Jenkins able to automatically build after every update of the website

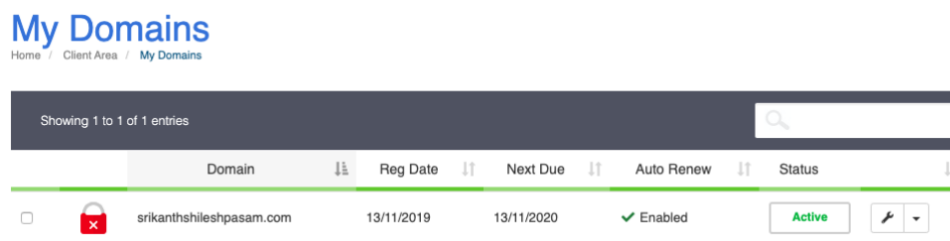











Fig.10 - Successfully purchased a domain




**CHECK-HOST**      


IP: 37.228.244.179 Country:  Ireland (Munster, Limerick)

Info Ping HTTP TCP port UDP port DNS

**IP and website location: srikanthshileshpasam.com**  

**DB-IP (05.11.2019)**

IP address	<b>104.40.231.66</b>
Host name	104.40.231.66
IP range	104.40.128.0-104.40.255.255 <b>CIDR</b>
ISP	Microsoft Corporation
Organization	Microsoft Corporation
Country	 <b>Netherlands (NL)</b>
Region	North Holland
City	Amsterdam
Time zone	Europe/Amsterdam, GMT+0100
Local time	22:25:23 (CET) / 2019.11.14
Postal Code	1012

 Leaflet | © OpenStreetMap contributors

Powered by **DB-IP**

Fig.11 - Linked the domain name to the IP address of the VM.

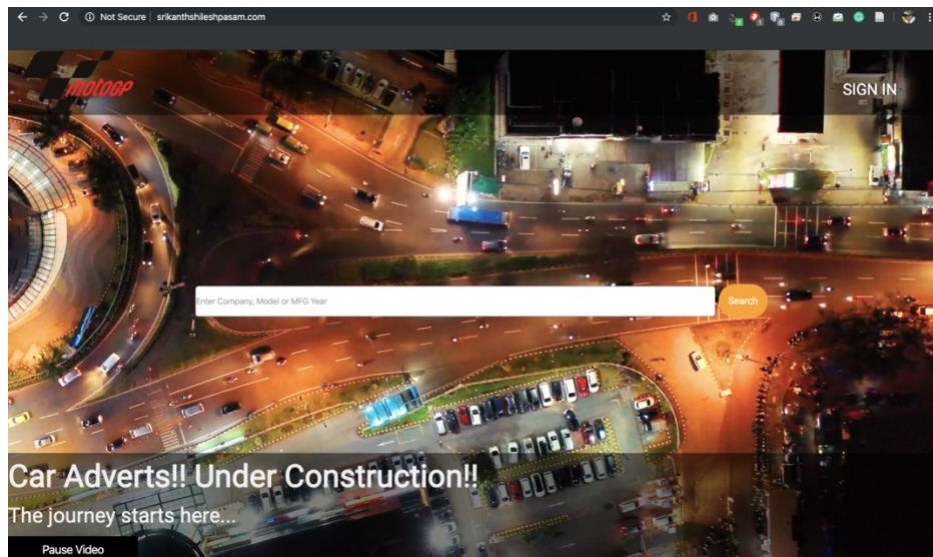


Fig.12 - Website being accessed using the domain name



# Reflection

This assignment has been one of the most challenging projects for me in my life. I come from a background with absolutely no experience in the field of IT, be it in academics or professional career. In order for me to start working on this project, I had to study multiple different technologies. The depth of each of these technologies was vast, and my time was limited. I had spent all the time I had in the past month to learn as much as possible. My external learning resources ranged from Udemy courses to Azure Learn to the documentation pages of the technologies on their websites. Initially, everything seemed too complicated and time-consuming. I felt like there was no progress made, and the submission date fast approaching. I kept persevering and pushing myself, and slowly but steadily, I started to make progress. In the time that followed, I started developing a more profound understanding of the concepts and was able to set vision and goals for this project. With careful planning and commitment, I started to achieve my goals and was able to bring my project to fruition. As I finish up my report, I am left with a satisfying sense of accomplishment.

# Bibliography

Ayshakeen (no date) *B-series Azure Windows VM sizes*. Available at: <https://docs.microsoft.com/en-us/azure/virtual-machines/windows/b-series-burstable> (Accessed: 16 November 2019).

*Benefits of cloud computing - Learn* (no date). Available at: <https://docs.microsoft.com/en-us/learn/modules/principles-cloud-computing/3-benefits-of-cloud-computing> (Accessed: 16 November 2019).

Bergmann, S. (2011) *Integrating PHP Projects with Jenkins*. O'Reilly Media, Inc.

Chamberlain, D. (2018) 'Containers vs. Virtual Machines (VMs): What's the Difference?', *NetApp Blog*, 16 March. Available at: <https://blog.netapp.com/blogs/containers-vs-vms/> (Accessed: 16 November 2019).

*Debian Repository for Jenkins* (no date). Available at: <https://pkg.jenkins.io/debian-stable/> (Accessed: 17 November 2019).

*httpd - Docker Hub* (no date). Available at: [https://hub.docker.com/\\_/httpd](https://hub.docker.com/_/httpd) (Accessed: 16 November 2019).

*IANA — Root Servers* (no date). Available at: <https://www.iana.org/domains/root/servers> (Accessed: 16 November 2019).

Mockapetris, P. V. (1987) *Domain names - concepts and facilities*. RFC 1034. ietf.org. Available at: <https://tools.ietf.org/html/rfc1034> (Accessed: 16 November 2019).

*Post-installation steps for Linux* (2019) *Docker Documentation*. Available at: <https://docs.docker.com/engine/installation/linux/linux-postinstall/> (Accessed: 16 November 2019).

*What is virtualization?* (no date). Available at: <https://www.redhat.com/en/topics/virtualization/what-is-virtualization> (Accessed: 16 November 2019).

# Appendix 1

Jenkins execute shell build code:

```
#!/bin/bash
```

```
if [ "$(docker inspect -f '{{.State.Running}}' apache) = true" ]; then
```

```
    docker rm -f apache
```

```
fi
```

```
docker build -t httpd:latest .
```

```
docker run --name apache -d -v /var/lib/jenkins/workspace/Jarvis-  
CarAdverts:/usr/local/apache2/htdocs/ -p 80:80 httpd:latest
```

# Appendix 2

*GitHub repository link for all the CLI codes used:*

<https://github.com/srikanthshileshpasam/Networking-and-System-Administration>