**Script**

**Risk Assessment**

With this project one of the largest risks to overcome was the lack of knowledge on the tools in which the application we’re deploying was built.

Something to be aware of in any project is the scope. Our second biggest concern was a short time frame in which to complete our deployment. As of such we planned for a minimum viable product to build to and make sure is working before attempting anything larger.

A problem with any projects that have elements that are connected to the internet with IPs is the risk of CSRF attacks. To connect to our virtual machines, key pairs have been set up through AWS.

DDOS attacks are also a problem for any project that is connected to the internet, however, as this isn’t a public project and none of us are sharing the IPs to our vms this shouldn’t be an issue.

**Cost of the Project**

We have found that in order to have the processing power to deploy the application we would require t2.small vms. This of course being outside of the free tier eligibility.

A t2.small machine costs 1.7 pence an hour. That is 41 pence a day leading to roughly £12.24 a month, depending on the length of the month.

As we would require 4 vms to be able to deploy the application this cost would rise to £48.96 a month.

**AWS**

We used AWS to host our virtual machines as a secure and easy place to run the application.

Before we could set up the infrastructure to automatically deploy the application, we first installed and ran the application manually on the vms. This was so we understood what needed to be done. The dependencies and the deployment order that needed to take place.

Hosting on AWS was also useful so all of us could access, edit and upload files to and from the vms.

**Jenkins Build**

Jenkins is used to automatically update the entire application over its many different vm deployments. It does so using a Jenkinsfile.

The Jenkinsfile will run again every time a new commit is made to the main branch of the git repo. Meaning the updates are automatically applied to the application deployment as and when they are added.

Using Jenkins like this allows for easier, automated deployment and version control. You can easily see what is different and roll back should the need arise.

**Ansible**

Ansible is a tool used to automate the deployment of work throughout the entire infrastructure.

It is an invaluable tool for a system admin as it allows a user to deal with; application deployment, configuration management, intra-service orchestration and basically anything else a system admin would need to use on a daily basis.

The tool we used the most from ansible in this project were its playbook.yaml files. These allowed us to set up vms with the software they required to run. This allowed for our main automotive process in establishing the vms. This also means if a vm ever needs updating or changing, doing so in the playbook file will automate this in conjunction with Jenkins.

**Terraform**

Terraform is a tool that is used to control infrastructure on a cloud service provider.

Using Terraform means that an environment will remain up to date. Because of the structure of terraform you are using infrastructure as a code, meaning that in order to use it you will be acting on something that is versioned and maintained.

Terraform is commonly used for multi-tier applications, software demos, disposable environments and multi-cloud deployments.

Our main use for terraform was to use it to create and configure the vms that we needed in aws.

**Docker**

Docker is a container management software that is used to create images and containers.

We use Dockerfiles to automate and streamline this process.

In this project we have used Docker to create and use images that are necessary to create and run the application. Using Dockerfiles to help automate doing so.

**Docker Swarm**

Docker swarm is a tool that allows users to create multiple containers across many different hosts, manager and workers.

Doing this means that when an update is applied it does not take down every version of the application that is running meaning there is never any downtime.

We have used docker swarm for this purpose, So that we may update parts of the code and not have to take down the application to do so. This also future proofs the project in case the creator of the application updates it, we can apply it will no downtime.

**NGINX**

NGINX is a tool that is used for; web serving, reverse proxying, caching and load balancing.

We have used it as a load balancer in order to access our different docker containers in our swarm. It automatically sends the user to an available container.