|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Description | Evaluation | Likelihood | Impact Level | Responsibility | Response | Control Measures |
| Amazon Server failures of problems | There is a slight, but extant, chance that Amazon cloud servers may go down or run into errors, effecting the EC2s and bringing the application offline. | Remote – AWS has had service outages but a whole AWS data centre has never gone down. | Severe – directly impacts the digital architecture that hosts and runs the application. | AWS | AWS servers are designed so that customers can recover data storied in any building in case an outage does occur.  As developers we could re-create or re-start the architecture or switch to another Cloud services provider such as Google or Microsoft. | CI Pipeline with Jenkins linked to Git VCM to readily test, build and deploy the project on new digital architecture if required. |
| Insecure Infrastructure | If access to the application is not managed and passwords and secure environment variables are not hidden then the application will remain open for tampering by public users. | Low – application contains no secure data and is of no meaningful interest to third parties. Security levels themselves have varied as the project progressed with initial environment variables and security groups being less secure and then tightened as we figured out the processes and learned how to improve security. | Low to Severe –unwanted access can have potentially very serious implications, including corruption of the entire application or loss of user data but the nature of the data in use makes the potential for harm relatively small. | Developers | Depends on the nature of the security breach – temporarily taking down the EC2 and investigating the nature of the harm is a good first step.  In a situation where the app continuing to run is more critical then the evaluation of security could potentially be done with temporarily highly restricted access for many users. Passwords and secure credentials should be changed and, if possible, made more secure. | Following best practices including VPCs, using a private subnet for the database server, security groups forbidding general access and following the policy of least privilege help restrict access and keep the project secure. |
| Excessive traffic or load causing application issues | If there are many users or demanding use cases for the application, then the virtual machine may be unable to function effectively under high demand. | Very low – the nature of the project makes high traffic unlikely and the processes it runs are not especially demanding – python in particular avoids some of the system-resource demands of other languages such as Java. | Significant – can cause app to fail to run as required. | Developers | The size and resources of the VM’s sued for the machine could be increased or load-balancing added via AWS to allow the application to scale as needed. | Choosing a VM with adequate system resources available and deploying the application through Docker-Swarm minimize the chances of excessive load causing issues. |
| Digital Architecture design decisions interfering with functionality and deployment. | AWS configuration options, such as CIDR blocks for Subnets and VPCs may interfere with the functionality of tools later used to containerise and deploy the project. | Certain (i.e. Risk Occurred) – network configuration impacted ability to use docker-stack deployment leading to a dns error denying database connection with the backend. | Moderate – can cause configuration issues or require the digital architecture to be restructured/recreated. | Developers | Recreating the EC2 with different VPC and Subnet CIDR configuration would resolve the issue in the specific case.  More generally, re-configuring either specific tools or the digital architecture in general should resolve conflicts. | Ensuring as little possible conflict as possible by carefully creating resources and potentially first checking which tools a project will use and how they run can help eliminate this issue.  It is important to also choose tools which are compatible with the nature of the architecture you choose to create – emphasising the importance of the planning and research stage to avoid potential issues. |
| Insufficient/poor documentation | Poor documentation can squander one of the main benefits of using VM’s and containerisation – the ‘it works on my machine’ issue. If the process is poorly explained and described then, while the automation may hold, the application is difficult to examine except by its creators. | Low | Low – potential issues can be worked through by experienced eyes looking at the code, however in more complex and longer applications this problem can scale to Severe. | Developers | If the developers are around to provide or create more documentation or explanations the problem is mitigated. | Writing clean, approachable code and creating documentation and plans as we progress helps ensure that there is sufficient information to easily use the application.  There is also an installation script included to ensure the application can be run and all required resources obtained easily. |
|  |  |  |  |  |  |  |