**Services used for migrating from GCP to AWS:**

<https://aws.amazon.com/pricing/>

**Service line**-

1. AWS EC2 t2 micro instances <https://aws.amazon.com/ec2/pricing/>
2. AWS Lamba that charges number of requests x duration of time it takes <https://aws.amazon.com/lambda/pricing/>
3. AWS P3 Sagemaker for deep learning models ($3.06/hour)
4. AWS Elastic Beanstalk <https://aws.amazon.com/elasticbeanstalk/pricing/>
5. AWS Sagemaker for basic ML models
6. Option for GPU <https://gpu.land/>
7. Use Azure Machine Learning (ML) <https://azure.microsoft.com/en-us/services/machine-learning/> and use pytorch for free 12 months trial ( let the AI team know to create the Azure accounts to get access to the GPU needed for data loading and training
8. AWS S3 for storage
9. Used to store the Artifact (Angular build output) in order to deploy the application from Github. Then we use the Artifact from the s3 service and push it out to ec2 for deployment.
10. AWS RDS which is the big query replacement
11. Single RDS service with 13 tables
12. Contact users
13. Deactivated healthcare providers
14. Deactivated patients
15. Healthcare provider authentication
16. Healthcare provider register
17. Job application
18. Job categories
19. Job openings
20. New users
21. Patients
22. Reset password tokens
23. Token schema
24. Verified User
25. Storing patient data that runs through the application
26. Contain provider data stored
27. Job application users
28. AWS Cloudtrail
29. Monitoring the logs
30. Usage and time of the transaction
31. Create Reports
32. Helpful for HIPAA compliance

**EC2 setting up the website:**

1. Once deployed, setup nginx as the reverse proxy. Currently setup in /etc/nginx
2. Mention the site link in the location parameter with key as proxy\_pass
3. This will set the http up
4. Now for https, place certificates in a folder
5. Add the location of the folder in nginx conf in listen key
6. Whenever a deployment happens, restart nginx using the command mentioned in the start\_script.sh
7. An elastic ip is set up to make the public ip static, otherwise it varies dynamically and can’t be set up in dns
8. Before all this, CodeDeploy IAM role has to be attached to EC2, Security group with inbound rules allowing access from the port required and other ports has also to be mentioned

**EC2 logging:**

1. For **cloudwatch**, awslogs service is required on ec2
2. The configurational changes have to be modified in /etc/awslogs/awslogs.conf
3. Place the location of the log file wanting to be monitored
4. We have /home/ec2-user/app/web-application/node/combined.outerr.log in place right now
5. The logs as well as the console outputs are appended into the log file
6. These are filtered out from the metric filters present in the Cloud management and queries are saved. Currently INFO and ERROR filters are in place
7. Dashboards are associated with these queries and saved to a EC-2 dashboard

**Beanstalk:**

1. We are using that container that stores our application within AWS.

**S3:**

1. Using it to temporarily saving the github artifact so that we can merge the new code and deploy into production

**Cloudwatch:**

What needs to be done:

1. Convert the logging into pdf format so that the provider organization can download the reports for their team and all patient transactions.

**CodeDeploy:**

Code deploy currently is copying the app and its required files and compiling them on the ec2 server. The location where it will deploy the code is being set in the appspec.yml file that is in the root directory of the app. It is also being stored on the github in the main directory.

Files:

Source: /

Destination: /home/ec2-user/app

This location needs to be updated in order to change the location of the codebase. But what it might entail on the permission side and what needs to be updated further on the codedeploy deployment package that needs to be investigated further.