Name: Seif Elbayomi

ID: B00856100

Capstone Project Report

The Azure Board:

https://seaforth.research.cs.dal.ca/CSCI%202690

GitHub Link:

https://github.com/nihalhk/finalAssignment

Team Meetings:

- **Planning Meeting**
  - https://dalu.sharepoint.com/:v:/t/CSCI2690Fall2021-TeamC-2690Superstore/Ebi4CD7IyrhIpkkhhUTEHGgBr0TMGBCG3flH42Suasq5nw?e=r1xM1j
- **Standup Meeting**
  - https://dalu.sharepoint.com/:v:/t/CSCI2690Fall2021-TeamC-2690Superstore/EbomOj4Yk_dAkpySbJ7rfU0BEKyDUARXT88zk1hX8BzYjw?e=Vr1fPo
- **Review Meeting**
  - **https://dalu.sharepoint.com/:v:/t/CSCI2690Fall2021-TeamC-2690Superstore/Ec3NLcNFdTdEqdHzJxtP3KEBX_n_GsGqBmcKWtGussuRWA?e=xK9PdP**
- **Retrospective Meeting**
  - **https://dalu.sharepoint.com/:v:/t/CSCI2690Fall2021-TeamC-2690Superstore/EQgVZgItKLRKmaqQdqxVE-MBQdn4dVpS_v8AqmdakYySBQ?e=Q2gMYf**

Process Description:

Firstly, I started my iteration by implementing the action items from the previous retrospect meeting. I started the report from the beginning of the iteration so that I can reflect all the changes on a real time manner. My teammate Nihal created the repo on day 1 of the iteration so that we can start working earlier on the project. I spent some time familiarizing myself with working with the repo and branches and I made research on both. I identified the risk and the risk treatment policy from the beginning of the iteration (All of that was reflected on the azure board in a real time manner).

Then I had a planning meeting with my team to plan how we are going to work on the project, and we created tasks for each requirement and assigned them to each team member and we made estimates on the time each task will take.

I firstly started researching each item on its own. First, I read a lot about what a container is and what an API is then what a GKE is and then what Azure is. Then I researched what are the benefits of GKE and what are the benefits of Azure each on their own. After that I researched what is ISTIO and what is AKS.

After that, I made some intensive research on comparing Azure with GKE and determining the pros and cons of each one of them to make the decision. I finally decided that the client should move to GKE due to the reasons mentioned in the answer of the first question below in the report.

After making the decision, I started researching on the services needed to run on GCP to run the APIs of the client and I listed them in the answer of the second question below.

Then I started researching the resources needed to do this migration smoothly. I also read an example of GitLab migrating from Azure to GCP and how their preparation and process itself was. (The process is illustrated in the answer to question 3 below)

Finally, I researched a deployment pattern I should use to minimize the impact on my clients. After intensive research from some strategies and referring to the GitLab migration example. I made a step-by-step strategy to make the migration process as smooth as possible and also made some suggestions that needs evaluation by the clients on the risks of these suggestions and whether he intends to take these risks or not.

Answering the Questions:

1- Should we move to GKE? why?

Yes, we should move to GKE.

Because of the following:

- Google, no doubt, presents a better offering when it comes to allowed quotas on its managed Kubernetes service GKE. With GKE, you can run up to 50 clusters /zones in addition to 50 regional clusters. AKS also allows you to run 100 clusters/regions.
  GKE allows 5000 nodes per cluster and 1000 nodes per node pools. GKE doesn't document how many node pools it allows. However, you are restricted to 110 pods/ nodes.

  Comparatively, AKS quota limits are a little conservative. It allows just 1000 nodes per cluster and 100 nodes per node pool. It allows up to 10 node pools. However, it has a larger pod limit; AKS users can run 250 pods/ nodes.

- GKE has the upper hand over AKS when it comes to upgrades. GKE is a lot less hassle as it allows automatic upgrades to its control plane and worker nodes. If you're on AKS, you probably have to manually upgrade the control plane and worker nodes every time, which slows the process down.
- Microsoft's AKS does support open-source Ubuntu in addition to Windows Server. When it comes to operating system support, GKE supports Container Optimized OS in addition to Ubuntu and Windows Server.
- GKE may have an edge with the number of added features over AKS

2- What are the list of services that we need to activate on GCP?

For running the client's APIs on GCP we need to do the following:

i. Creating a Google account
ii. Creating a Google project
iii. Discovering APIs library
iv. Enabling APIs
v. Enabling billing
vi. Getting application credentials
vii. Using application credentials
viii. Build the applications

For servicing multiple APIs, we need to do the following:

i. Verify domain name and ownership of the domain

ii. Deploying an API on a subdomain (Set the host field , Deploy the Endpoints configuration, Deploy the API backend.)

3- What resources do I need to do this migration?

To do this migration we need to do the following:

Configure the source and destination for migration

Set up GCP as a migration destination. Before beginning a migration to Google Cloud, you must create Identity and Access Management permissions.

Set up a VPN for communication between the migration source and destination. Plan for and create a secure connection between Azure and Google Cloud. One way to do this is to follow this guide.

Configure network access. See Network Access Requirements for detailed information on firewall, routing, and network tag considerations for your Migrate for Compute Engine deployment.

Create Azure credentials in the Migrate for Compute Engine Manager that will enable Migrate for Compute Engine to connect to Azure.

Set up the Migrate for Compute Engine Manager. The Migrate for Compute Engine Manager provides a web UI and controls migration operations from Google Cloud.

Create Cloud Details on your Migrate for Compute Engine Manager using your Azure credentials.

Set up Cloud Extensions. After configuring the Migrate for Compute Engine Manager, create Cloud Extensions for your migration.

Prepare your Linux VMs. If you are migrating Linux VMs, install the Migrate for Compute Engine package to reconfigure them for Google Cloud.

Migrate VMs

Migrate a wave of VMs. Migrate for Compute Engine organizes groups of VMs into waves. When you understand the dependencies of your applications, create runbooks that contain groups of VMs, then begin your migration.

Migration process of VMs:

During migration of an instance from Azure to Google Cloud, Migrate for Compute Engine does the following:

Stops the source VM in Azure.

Creates the Migrate for Compute Engine VM Importer at Azure.

Takes a snapshot from the source VM disk.

Creates a data disk from source VM snapshots and attaches it to the Migrate for Compute Engine importer.

Creates an instance in Google Cloud.

Streams data from the importer to the Google Cloud Cloud Extension.

When migration is complete, Migrate for Compute Engine terminates the importer and resources are cleaned up.

At the end of the process, the original Azure instance remains intact and powered off.

Migrating with waves

You can migrate VMs from Azure to Google Cloud with Wave Migrations

4- What deployment pattern should I use to minimize the impact on my clients?

We should use a sequential approach to migrate to GKE to have the least impact on the client.

The process should be as follows:

1- Prepare teams for a Google Cloud migration
This can be achieved by proper training for all the teams to ensure those teams are taking full advantage of all the Google tools and services they might need to optimize app performance and take full advantage of the cloud.

2- Use monitoring and automation to safeguard performance
NAPM solutions such as Accedian's Skylight are ideal for offering a holistic view of the network and delivering proactive alerts for correcting issues before they impact app and SD-WAN performance.
Automation is another important tip for a successful, optimized app migration. Critical jobs like deployments, exchanges and configuration updates may all be easily automated. It reduces the chance of manual errors and prevents valuable human workers from wasting time and money on repetitive tasks.

3- Take advantage of Google's managed service

A few examples include App Engine, which offers serverless web hosting, and CloudSQL for MySQL, which replaces MySQL cluster management. GCP also provides AutoML for tagging and classifying images with the option to deploy workloads on GKE instead of managing Kubernetes clusters.

4- Scale resources to optimize app and network performance

As GCP enables horizontal scaling, which allows users to elastically add or remove VMs, cluster nodes, and database instances. Vertical scaling allows more resources to be added to existing instances without the need for any additional physical infrastructure.

Again, a strong NAPM offering is beneficial when making changes to the network, providing real-time monitoring and alerts about app and network performance. This enables administrators to make changes proactively before they have a negative impact on network operations.

5- Use GCP app migration to reduce costs

Google's migration tools help with in-cloud testing and validation during migration, which can help spot performance issues early. Workloads also can be migrated in phases (known as "migration waves") for a more orderly migration.

By using GCP's built-in migration tools, many of the typical "gotchas" of app migration can be avoided or resolved quickly. When used in tandem with a NAPM solution like Skylight, performance issues can even be eliminated during the move to GCP.

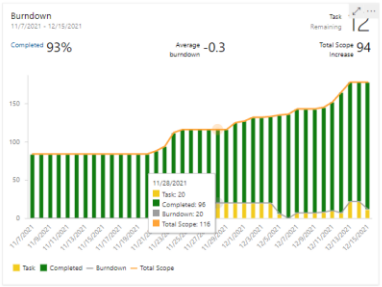
Migrating apps and workloads to the Google Compute Platform clearly offer major benefits to enterprises. By taking advantage of the above tips, businesses can optimize their app migration and glean the most value from GCP.

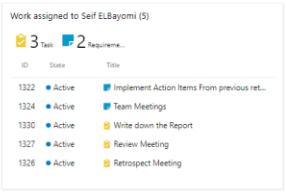
App optimization during a cloud migration is simply not possible without close monitoring and management of network and application performance. That's why Google actually recommends that its enterprise customers deploy a monitoring and alerting system to ensure applications and networks achieve and maintain optimal performance both during and after a cloud migration.

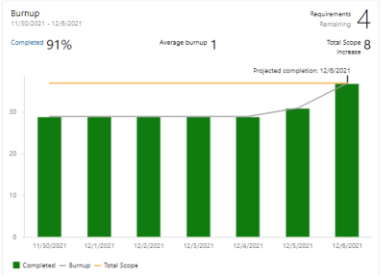
Risk Register and Risk Treatment Plan Policy:

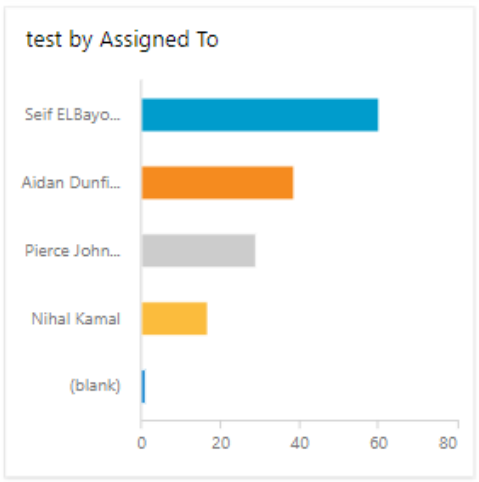
ID	Description	Likelihood of risk occurring	Impact of risk	Owner	Risk Treatment Plan Policy	Date Raised
1	Loss of control over data and process	4	5	Project Manager	Restrict access manually. Encrypt data to help with security.	[enter date]
2	Inadequate Preventive Security Checks	3	4	Project Manager	Outsource someone with expertise in cloud migration	[enter date]
3	Additional latency	2	2	Project Manager	Improve network localization and optimization. Track traffic flow by segment Interconnect multiple clouds. Outsource the Internet at the edge. Engage business partners and ecosystems for data sharing and digital commerce	[enter date]
4	Infrastructure vulnerabilities	3	3	IT Team	Perform a cloud readiness assessment With IT team	[enter date]
5	Incompatibility with current configuration	5	2	Project Manager	Hire IT to fix tech debt, review the legacy architecture, make comprehensive documentation, and measure interdependent parts.	[enter date]

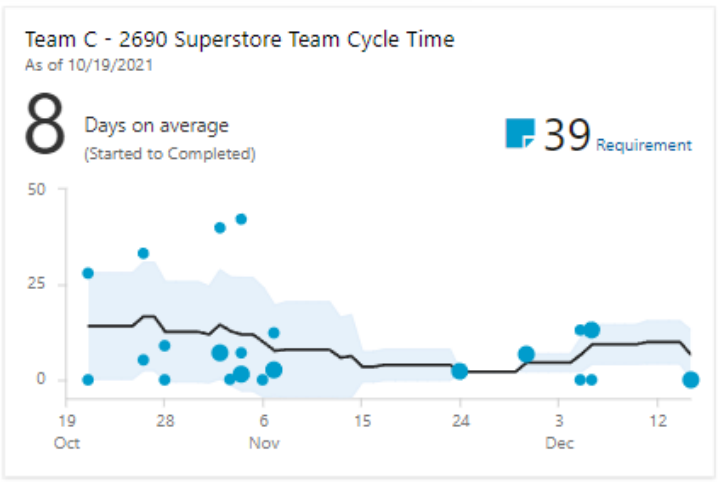
Screenshots:

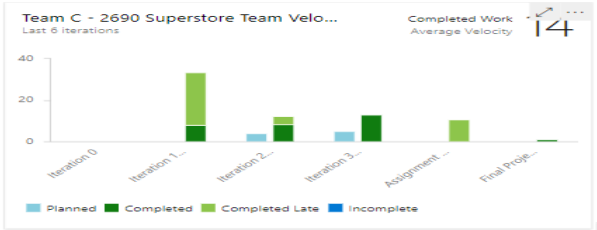


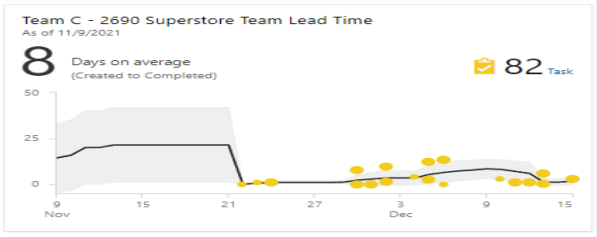


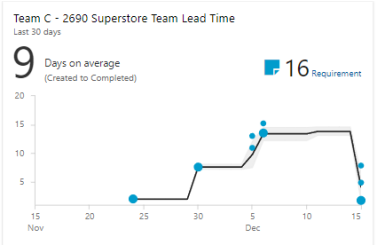


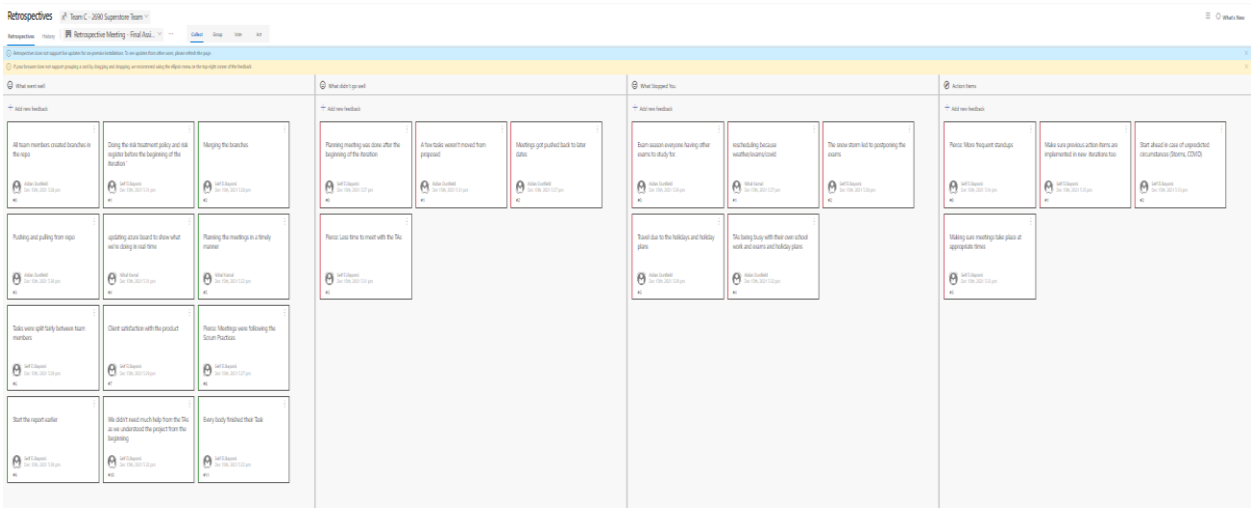











Retrospectives
Team C - 2690 Superstore Team
Retrospectives
History
Retrospective Meeting - Final Assi...
Collect
Group
Vote
Act
What's New
What went well
Add new feedback
All team members created branches in the repo
Doing the risk treatment policy and risk register before the beginning of the iteration
Merging the branches
Pushing and pulling from repo
updating azure board to show what we're doing in real-time
Planning the meetings in a timely manner
Tasks were split fairly between team members
Client satisfaction with the product
Pierce: Meetings were following the Scrum Practices.
Start the report earlier
We didn't need much help from the TAs as we understood the project from the beginning
Every body finished their Task
What didn't go well
Add new feedback
Planning meeting was done after the beginning of the iteration
A few tasks weren't moved from proposed
Meetings got pushed back to later dates
Pierce: Less time to meet with the TAs
What Stopped You
Add new feedback
Exam season everyone having other exams to study for.
rescheduling because weather/exams/covid
The snow storm led to postponing the exams
Travel due to the holidays and holiday plans
TAs being busy with their own school work and exams and holiday plans
Action Items
Add new feedback
Pierce: More frequent standups
Make sure previous action items are implemented in ever iterations foo
Start ahead in case of unpredicted circumstances (Storms, COVID
Making sure meetings take place at appropriate times