- **Describe how you met your menu item requirements: you will list the services you selected, and provide a comparison of alternative services, explaining why you chose the services in your system over the alternatives.**

Service I used	Alternative	Description/reason
Compute		
Lambda		• Do not worry about infrastructure • Automated scaling • Highly cost-effective compare to other options (pay by per milliseconds)
	EC2	• EC2 is suitable for continuous running applications. • I do not have any long running application. • Instead, I'm using lambdas to call via API gateway as for short lived tasks such as storing data to db, encrypting-decrypting session token etc. when required • Cost is exponentially reduced by using lambda as I pay for minimal memory as well as storage Scaling mechanism needs to be manually setup for EC2
	Elastic Beanstalk	• This option is suitable for fully functional applications running for long time. • Whereas, I have divided my application into smaller parts such that each part doesn't require much computing power and work independently and can complete the processing under 29 seconds
AWS Step Functions		• To store user_id and related access_token received from Fyers API, I have used step function. Inside step function, first lambda is responsible for encrypting the access_token so security of access_token can be increased. • Second lambda function is storing it into db for future requirements. • As a future enhancement, if I want to store encrypted password or any other parameter (doing some preprocessing), I just have to add another step to this step function • I can add a security notification (YOU HAVE RECENTLY LOGGED IN) option as well which can work as a parallel flow when user sign in
	Combination of lambdas calling each other	• However, the step function is internally doing this only, it is cleaner way to do this rather than this option because it reduces coupling. • Another lambda function doesn't have to worry about how the data is coming and from where. Both lambda function performs a single task. First one is doing encryption of access_token and another is storing it in DB.

Service I used	Alternative	Description/reason
Storage		
S3		• Highly Scalable and durable (11 9s of availability) object-based storage • Security parameters are high (each file can be protected with different access policy) – In future this option can help to increase security by setting up policies such that specific user has access to specific files • For my application, I'm using S3 to store code files of lambda – in future I can add a security policy that allow specific lambda to access specific file from S3 • Provide cost effective option (S3 glacier) to archive data
	Elastic Block Store (EBS)	• It is mainly used as a persistent block-level storage for EC2 as a primary storage • It is designed for low-latency data access such as transactional database • It is costlier than S3
	Elastic File System	• It is suitable as EC2 instance (which I'm not using) storage • It supports Network File System protocol with high availability • EFS is designed for applications that require file-based access to data, such as databases, analytics, and media processing • It is the costlier than EBS
DynamoDB		• Provides Consistence performance with Automatic backup, restore options. It has automatic scalable and built-in tools for data analytics • I have used DynamoDB for 2 tables (users and track_list) where users table contains user_id and encrypted (with KMS key) access_token and track_list contains user_id and stock to track with its target price
	Neptune	• It supports Graph Query Languages, mainly used for graph databases which requires high-performance graph queries and analysis • It is focused on ML tasks such as detecting fraud patterns or predictions • Graph database is mainly used for vertices and edge-based systems which is networking heavy i.e., which has highly corelated data. Thus, it is not suitable for my use case as all users and their track list should be independently managed
	AppSync	• AppSync is used for gathering data from multiple sources (database or non-database) and act as a single network request for the application • It is also a GraphQL based database management service (not a database!) • It works on Pub/Sub model where client can subscribe to specific data and AppSync publishes an update when that data is changed
	Timestream	• It is used for storing time-series data • It is highly effective for the storing recent data in highly efficient storage and moving historical data into a cost-optimized tire according to custom user policies • Not useful for my case as I'm not using Time-series data

Service I used	Alternative	Description/reason
	Aurora	• Aurora is suitable for relational database and it mainly supports MySQL and PostgreSQL • It is cost heavy than DynamoDB • For my use case, I wanted to be open for unstructured data to analyze in the future to generate tips based on current user track lists
	Athena	• Athena is not a database rather it provides analytical service that can run optimized SQL queries on S3 or any other databases. • Not suitable for my use case as I need the database to store the users' data
	IoT Analytics	• Mainly used for IoT device analytics using MATLAB or Octave scripts • Not going near to IoT devices so not relevant to my use case
Network		
AWS API Gateway		• AWS's only easy to configure option to create REST API that interacts with Lambda • Easy to build secure REST API or WebSocket APIs • Easily monitored and cost-effective scalable option with performance
	Virtual Private Cloud (VPC)	• It can't create APIs so not suitable for calling Lambda • It is mainly used for protecting the networks such that no outside device can access unrequired resources
	CloudFront	• CloudFront works as a first level security for API • It can be added in front of API Gateway to enhance the security with traffic encryption and access controls, and use AWS Shield Standard to defend against DDoS attacks at no additional charge
	EventBridge	• To build event driven apps, EventBridge is used • It also supports scheduled event • My architecture is not dependent on events thus EventBridge
General		
Secrets Manager		• Used Secret manager to store API Keys of Fyers API • This secret can be secured by using policy such that only required services can access to the secret • Secret can be encrypted using custom KMS keys • These secrets can be easily replicated across regions
	Systems Manager Parameter Store	• It is used to store parameters of the application • It doesn't encrypt the data being stored • Less secure than secrets manager but cheaper option • Since API key is confidential credential, I have used Secrets Manager to store the key
Key Management Service (KMS)		• Provide AWS generated key to encrypt and decrypt data • Custom key can be added if one doesn't want AWS generated key • I have used AWS generated symmetric key to encrypt access_token retrieved from Fyers API to store in db and while retrieving the token, I'm decrypting the access_token with the same key to hit the Fyers API for fetching data

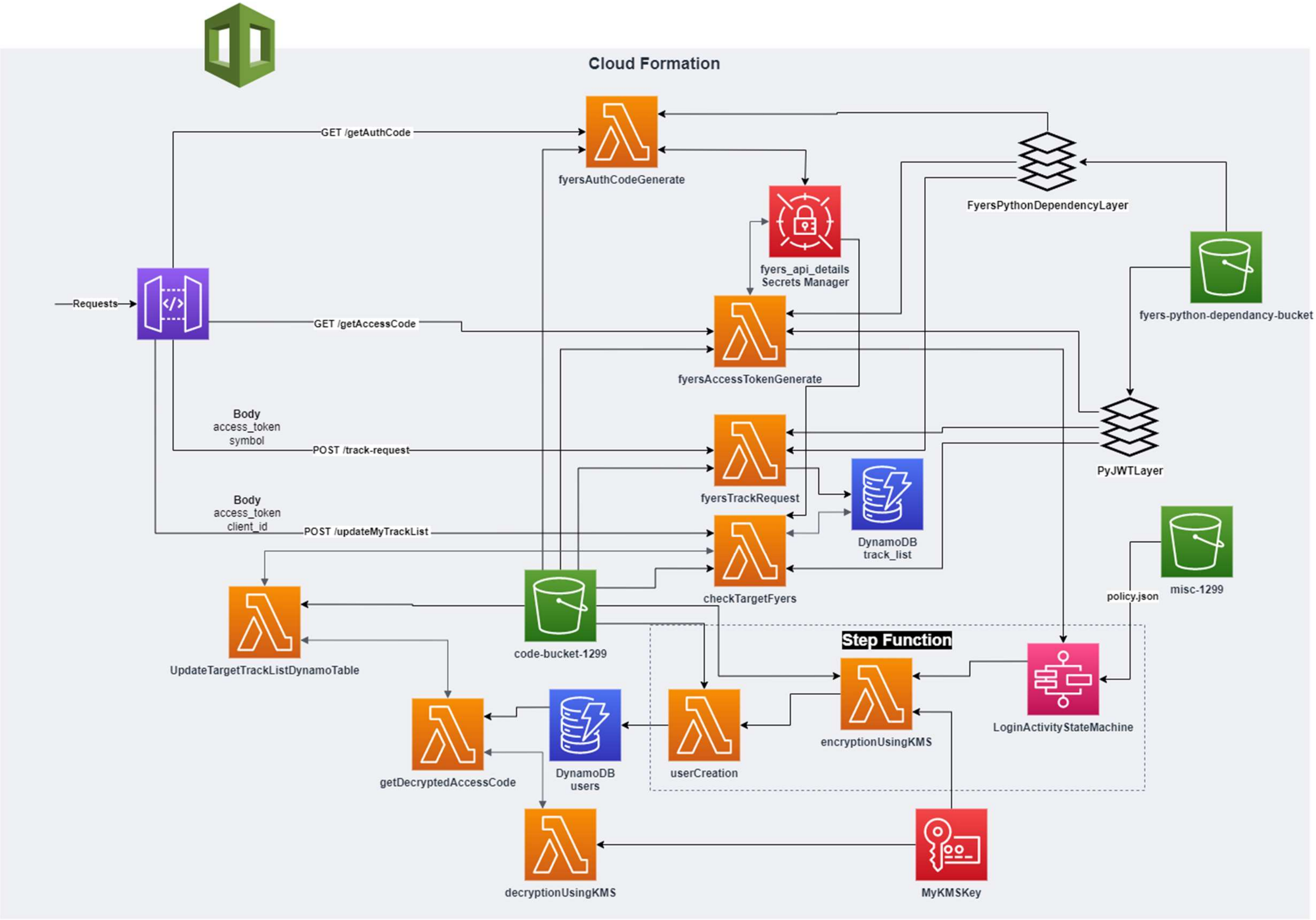
- **Describe your deployment model. Why did you choose this deployment model?**

  I have used AWS public cloud throughout, although it is Vocareum managed but I doubt that there are any special measures taken from Vocareum side. Thus, I'm using the **Public cloud** deployment model.
  I want to make my API available publicly instead of specific group of people. All the IT resources are managed by AWS and I'm trusting AWS to deliver on service level agreements defined on their privacy policy.

- **Describe your delivery model. Why did you choose this delivery model?**

  I have used Function as a Service (FaaS) as a delivery model.
  I have exposed all the required endpoints using API gateway which when called activate a lambda that does the computation, coordinates with required services and return response.
  There is no infrastructure was configured by me. My code will only run when needed and nothing will be idle and waiting for something to happen.

- **Describe your final architecture:**



  - **How do all of the cloud mechanisms fit together to deliver your application?**

All computations happen in lambda.
API Gateway will get the request and route it to appropriate lambda function.
Lambda function will coordinate with all other services like KMS, Secret manager, State Machine etc. as required.

  - **Where is data stored?**

All the Lambda code files, CloudFormation files, Lambda Layer zip file are stored in S3 bucket.
User data is stored in DynamoDB.
API secrets are stored in secret manager
Encryption decryption key is store in KMS

  - **What programming languages did you use (and why) and what parts of your application required code?**

To provision the whole architecture without clicking any button from the console I used Cloud Formation. In which YAML is used for cloud formation. It is cleaner than JSON.

All computation that happens in lambda requires code. I used python for lambda code as it provides libraries.

- **How is your system deployed to the cloud?**

1. To create lambda dependency layers, use create_layer_from_requirements.bat file which will create a layer zip file from requirements.txt file.
2. Provision happens via cloud formation file.
3. First S3 CloudFormation file should be loaded via AWS CloudFormation console.
4. Once all buckets are provisioned upload dependency zip files into dependency bucket
5. Upload State machine policy into misc bucket.
6. Upload lambda code zip files into code bucket.
7. Run CloudFormation file that will provision all other services.
8. Now API is up and running use postman collection attached in GitLab to test it out.
   (YOU NEED FYERS ACCOUNT TO LOGIN)

- **If your system does not match an architecture taught in the course, you will describe why that is, and whether your choices are wise or potentially flawed.**

I have used Rapid Provisioning advance Architecture to provision architecture using CloudFormation which provisions all the resources in minutes.
I have used Dynamic Scalability Architecture. Because lambda can scale up and down autonomously. Also, DynamoDB as well as S3 buckets are highly scalable.

- **How does your application architecture keep data secure at all layers? If it does not, if there are vulnerabilities, please explain where your data is vulnerable and how you could address these vulnerabilities with further work.**

After going through the security measures, I understood that my data is not secured as it should be.

1. AWS provides various mechanism to secure the data. I have used Secret manager to store the Fyers API's credentials and it works for AWS services but I'm providing it hardcoded in CloudFormation file itself. Anyone who has access to CloudFormation file can access these keys.
2. I'm using KMS key to encrypt session token (access token) provided by Fyers API (after user sign in happens) before storing in the database. That way I can ensure that even if my table got compromised, access token will not get compromised.

For both the above cases, I could have assigned more strict policy such that only required lambda should have access to Secret Manager or DynamoDB table and KMS keys should be accessible to required resources. I could have used rotate key features to make the key more secure. I could have used Asymmetric key while creating the KMS key as it is more secure than symmetric.

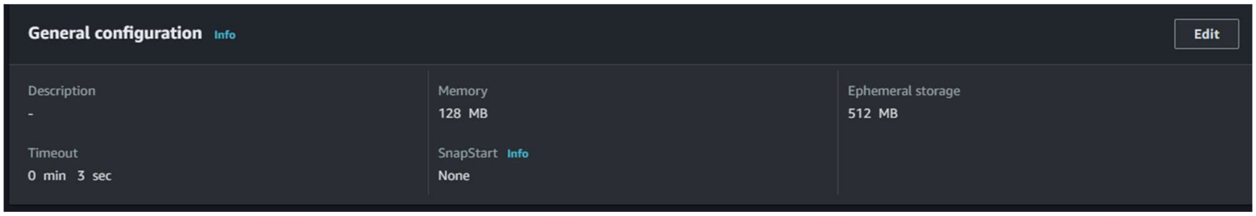
  - **Which security mechanisms are used to achieve the data security described in the previous question? List them, and explain any choices you made for each mechanism (technology you used, algorithm, cloud provider service, etc.)**

1. Encryption: encrypting access token via KMS keys
2. Hashing: These keys are getting hased while encrypted. Also, Secret manager hashes the secret.

- **What would your organization have to purchase to reproduce your architecture in a private cloud while providing relatively the same level of availability as your cloud implementation? Try to give a rough estimate of what it would cost, don't worry if you are far off. These systems are complicated and you don't know all the exact equipment and software you would need to purchase. Just explore and try your best to figure out the combination of software and hardware you would need to buy to reproduce your app on-premise.**

  To provision the whole architecture comprised of cloud formation, API gateway, lambda layers, step function, KMS key, secret manager only software(code) is required and some storage to store this software (maximum storage estimate 1 GB) as well as minor processing CPU (maximum 1 GB).
  I'm assuming that hardware virtualization possible in private environment.
  16 GB DDR4 RAM = $57.60 [1]
  Thus, 1 GB DDR4 RAM = $57.60/16 = $3.6
  1 TB SSD = $62.99 [2]
  Thus, 1 GB SSD = $62.99/1024 = $0.06

One lambda function takes following resources to work:



General configuration

Description
-

Memory
128 MB

Ephemeral storage
512 MB

Timeout
0 min 3 sec

SnapStart
None

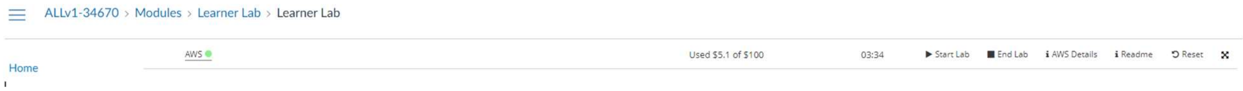
Memory: 128 MB = $3.6/ (1024/128) = 3.6/8 = $0.45
Storage: 512 MB = $0.06/2 = $0.03
Total Lambda Hardware Cost = $0.48

To keep all these things intact there will be required a Rack (or cabinet) = $148.98 + GST (GST can be redeemable as business expense so not counting it in estimate)

Since these servers will be running 24x7 it will require a cooling system as well = $114.29

Service	Required Hardware	Calculation	Cost
Computing power required to run all the software to acts like below services	Processing Power (RAM) – 16 GB	$57.60	57.60
cloud formation	SSD (storage) – 1 GB Processing Power (RAM) – 1 GB	3.6 + 0.06 = $3.66	3.66
API gateway			
lambda layers			
step function			
KMS key			
secret manager			
S3 buckets storage (2)	SSD (storage) – 1 TB (total)	$62.99	62.99
DynamoDB storage	SSD (storage) – 1 TB (total)	$62.99	62.99
Lambda	9 * Lambda Hardware Cost	9 * $0.48	4.32
Total Hardware Cost			**191.56**
Cabinet		$148.98	148.98
Cooling System		$114.29	114.29
Total With Everything			**454.83**

If we only count essential hardware, it will cost around $200
With all the required accessories to accommodate this hardware it will cost around $500
Compare to this AWS is providing all this for only 1% price ($5) as shown in below image



Used $5.1 of $100

- **Which cloud mechanism would be most important for you to add monitoring to in order to make sure costs do not escalate out of budget unexpectedly? In other words, which of your cloud mechanisms has the most potential to cost the most money?**

I would add monitoring to Lambda and DynamoDB as these are the resources costing most money. Because I use them frequently throughout my API, I have to keep tap on its usage.
Secret manager is costlier service but I'm calling it very few times and only for 1 secret so it should not impact the budget overall.

- **How would your application evolve if you were to continue development? What features might you add next and which cloud mechanisms would you use to implement those features?**

I want to add the login notification as a security measure. For that I can use Simple Email Service (SES). I can add this step in existing step function so whenever a user login happens, a notification should go to that particular user.
I want to create a UI for my API; I can containerize the UI which can be hosted on Elastic Container Registry and deployed on Elastic Container Service.

I can use Virtual Private Cloud (VPC) to increase security so only intended services should know about each other.
I can introduce policies such that only allowed services use a particular service.

REFERENCES

[1] “Crucial RAM 16GB DDR4 3200MHz CL22 (or 2933MHz or 2666MHz) Desktop Memory CT16G4DFRA32A : Amazon.ca: Electronics,” *www.amazon.ca*. https://www.amazon.ca/Crucial-16GB-Desktop-Memory-CT16G4DFRA32A/dp/B08C56GZGK/ref=sr_1_1?hvadid=588886769183&hvdev=c&hvlocphy=9000087&hvnetw=g&hvqmt=e&hvrand=18001826403199633 74&hvtargid=kwd-298139575183&hydadcr=24949_13553992&keywords=16gb+memory+ram&qid=1681068276&sr=8-1 (accessed Apr. 09, 2023).

[2] “Silicon Power 1TB NVMe M.2 PCIe Gen3x4 2280 TLC R/W up to 2,200/1,600MB/s SSD (SU001TBP34A60M28CA) : Amazon.ca: Electronics,” *www.amazon.ca*. https://www.amazon.ca/Silicon-Power-Gen3x4-600MB-SU001TBP34A60M28CA/dp/B08WZ5F86P/ref=sr_1_1?hvadid=596266197596&hvdev=c&hvlocphy=9000087&hvnetw=g&hvqmt=e&hvrand=7172530748087888187&hvtargid=kwd-302276912559&hydadcr=5743_13218199&keywords=1tb%2Bm%2B2&qid=1681068392&sr=8-1&th=1 (accessed Apr. 09, 2023).