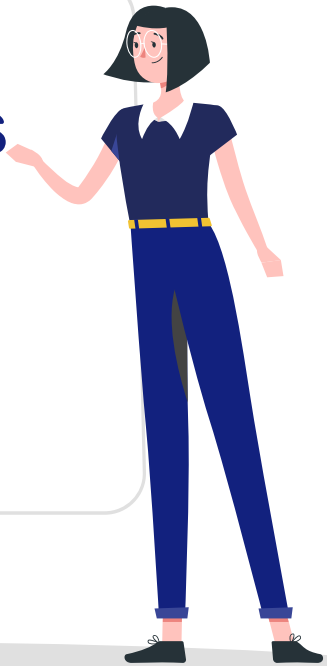
Ascenda

Loyalty Program API

**Austin Woon, Lee Sherman, Wong Javier,
Tan Leonard, Tan Brennan, Khoo Ernest**

Business Needs & Solution Overview



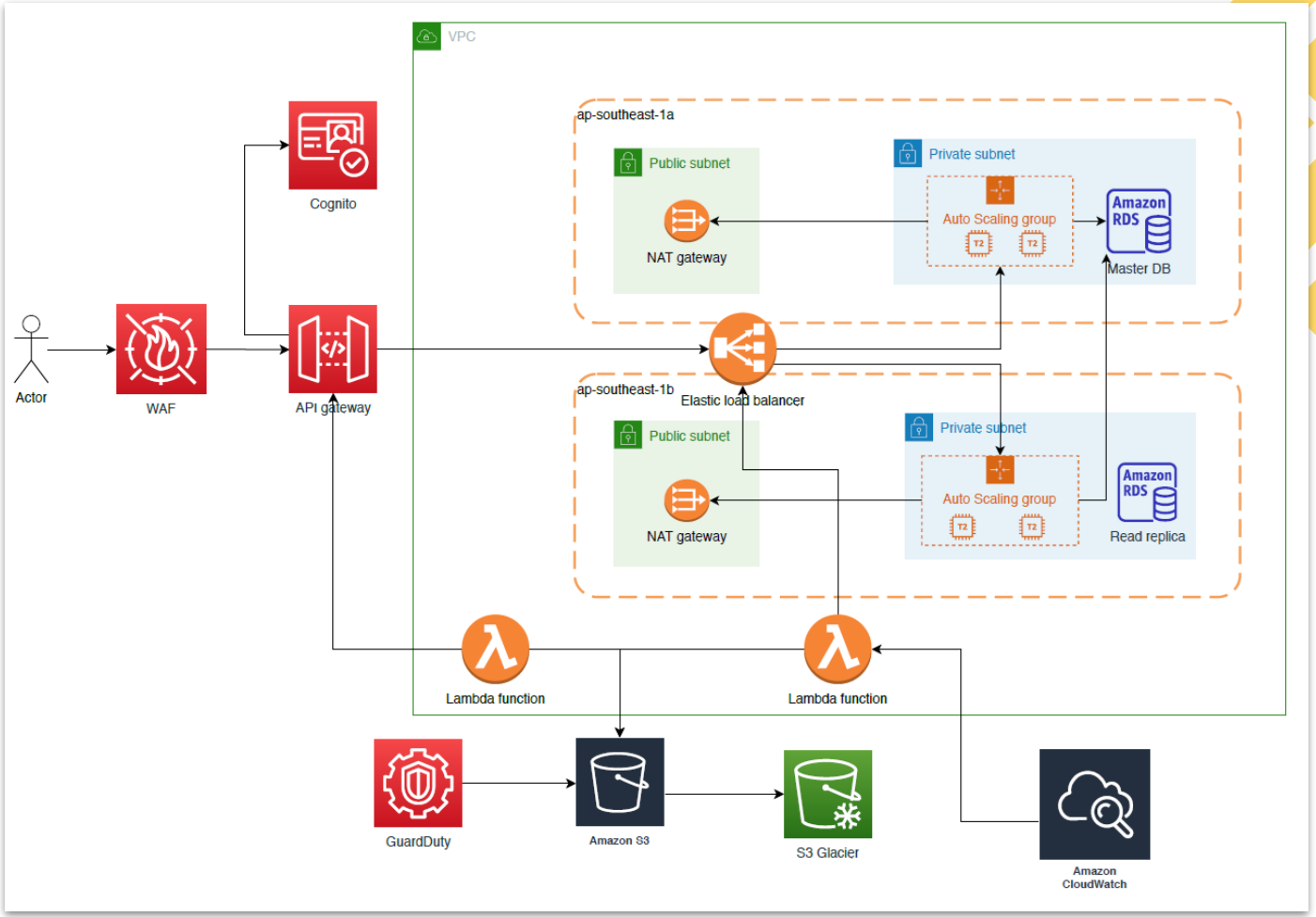
Business Needs

Loyalty partners are customers of Ascendas and require a solution that will facilitate easy and seamless loyalty currency transfers between loyalty programs

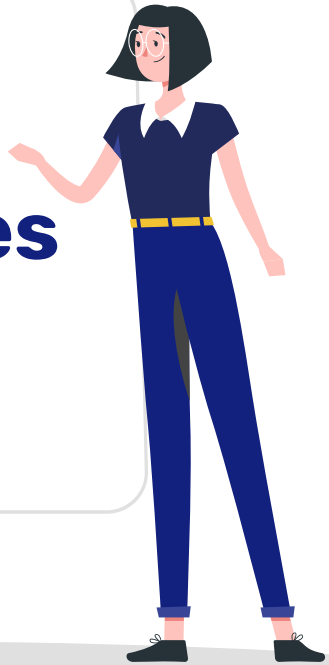
Ascendas must provide its customers with an API that will

- **Display information of loyalty programs on the bank's frontend**
- **Perform loyalty membership validation**
- **Accept & process accrual information for loyalty programs on behalf of banks**
- **Return transaction details when complete and allow querying of transfer details**
- **Perform fulfilment with actual loyalty program, integrating with their specific transfer formats**

Solution Overview



Product Features



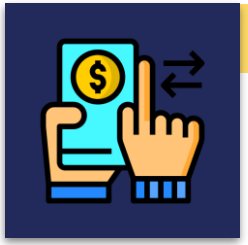
Features Requested

Loyalty Program API

Single source of truth for latest information regarding loyalty programs

Membership Validation

Validate Memberships of Users belonging to loyalty programs



Accrual Requests

Process *new accrual requests* from Bank

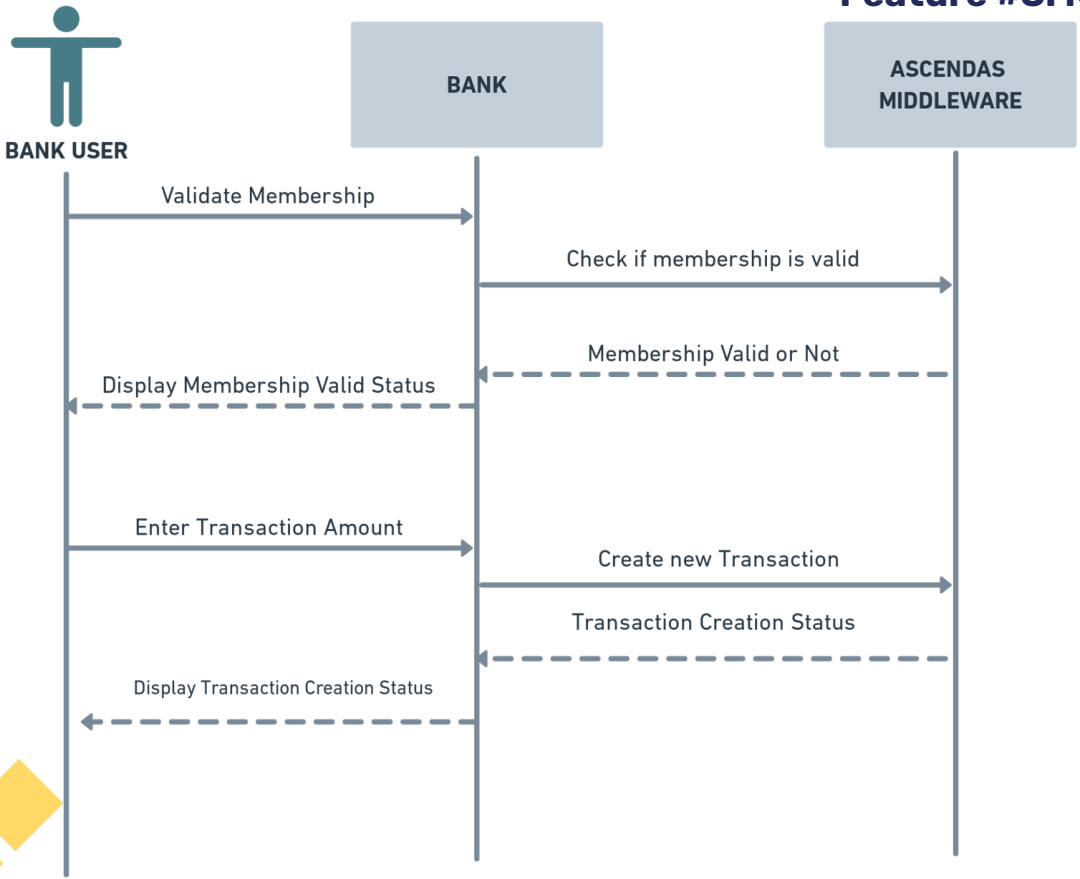
Transaction Enquiry

Allow banks to *poll for transaction updates* periodically

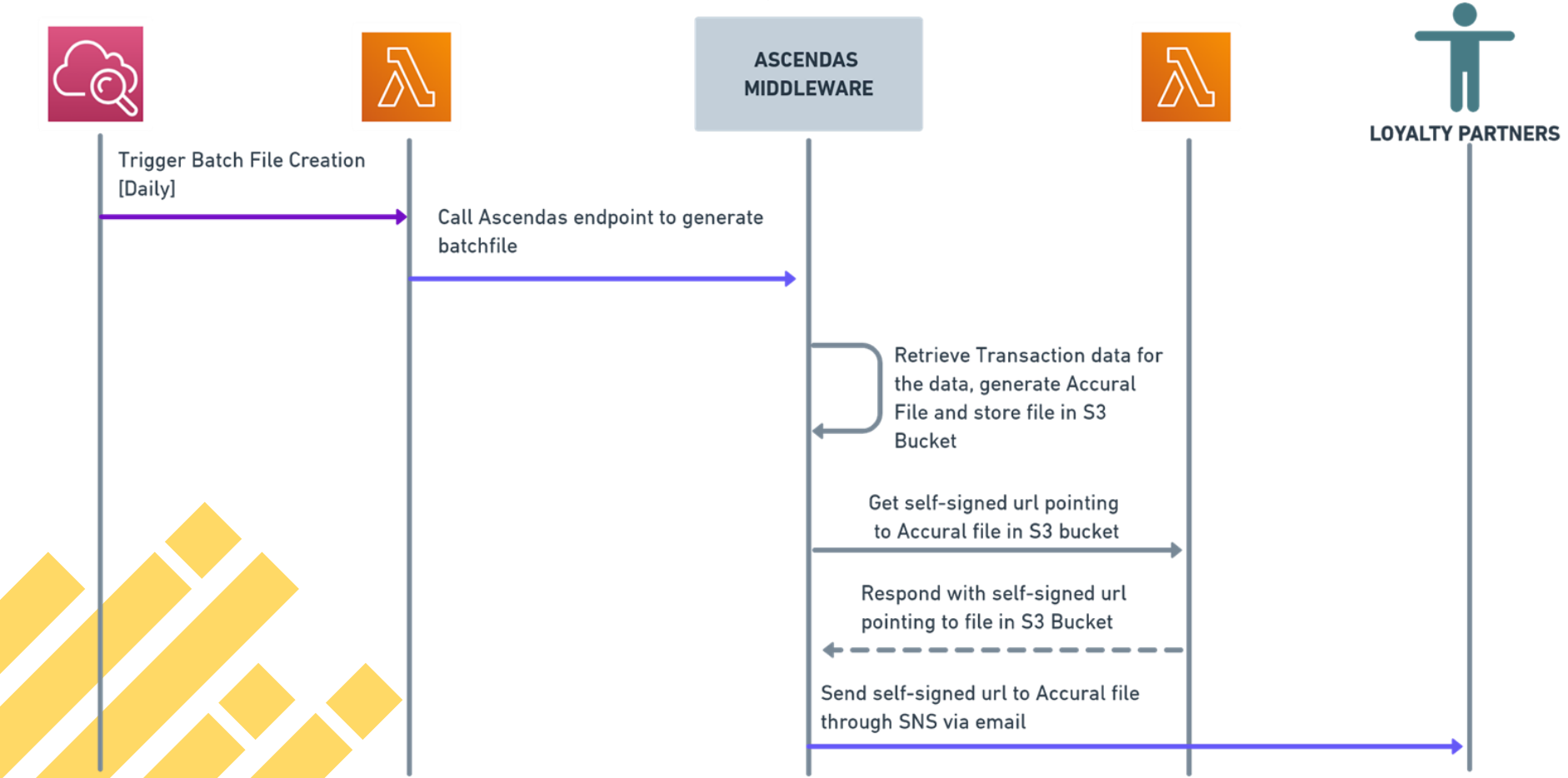
Transfer Fulfilments

Handle *transfer fulfilments* between Loyalty Partners and Banks

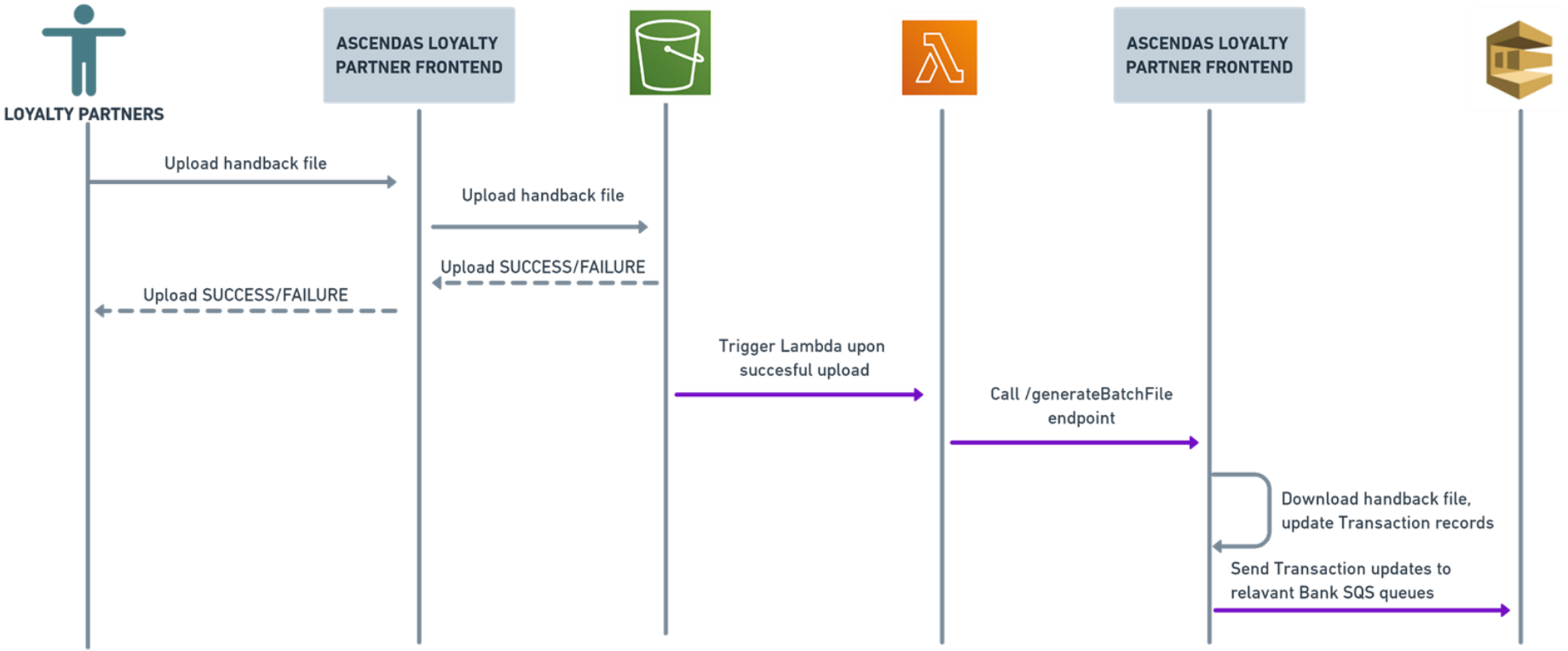
Feature #1: Loyalty Program API
Feature #2: Membership Validation
Feature #3: New Accrual Request



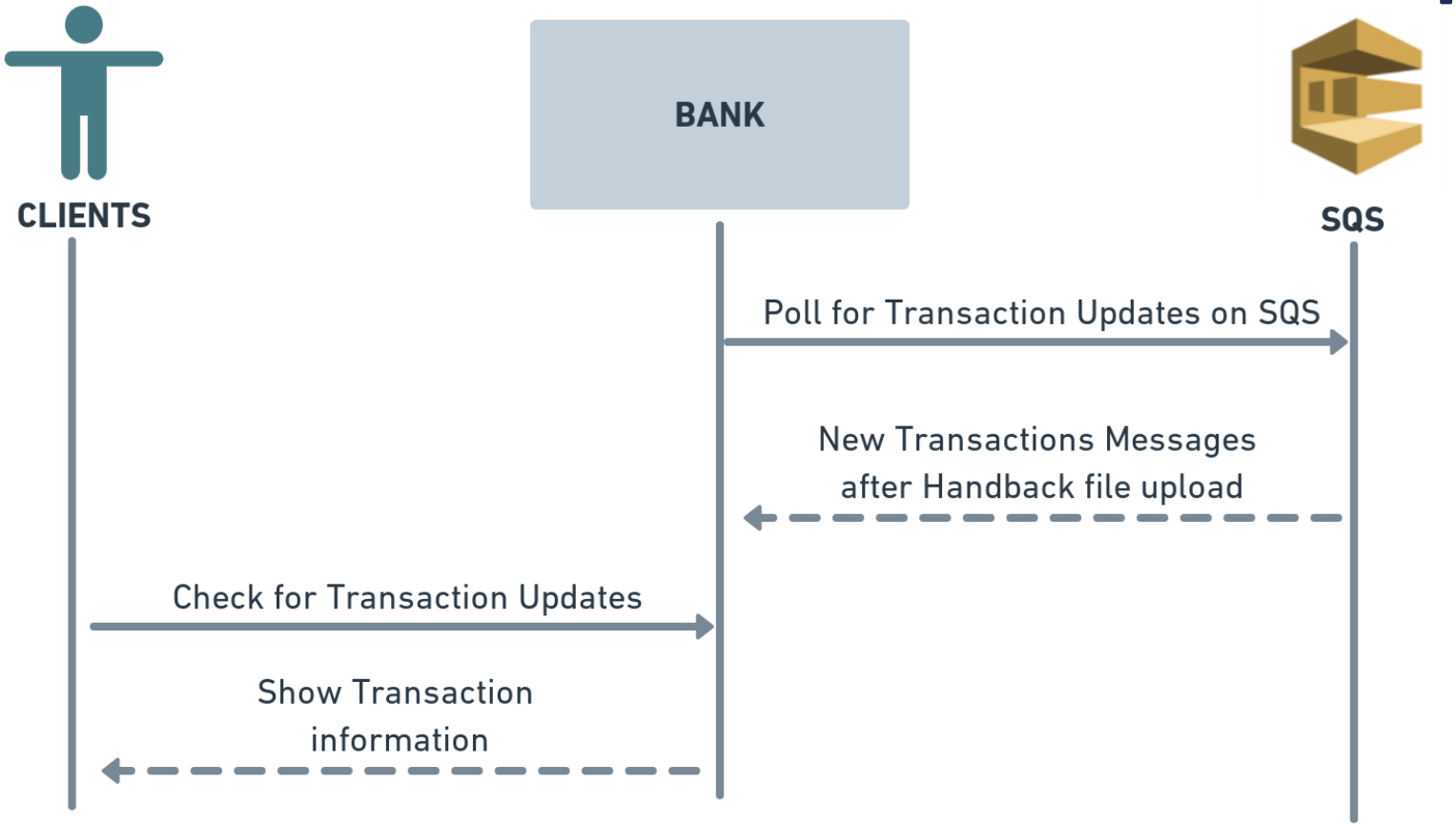
Feature #5: Transaction Fulfilments

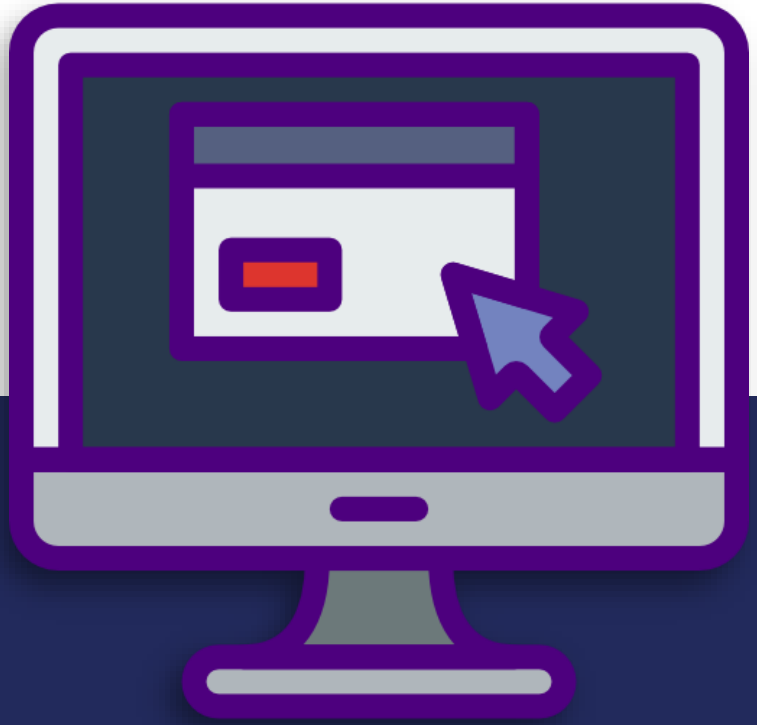


Feature #5: Transaction Fulfilments



Feature #4: Transaction Enquiry





PRODUCT DEMO

Maintainability

Development Process - Pipeline

Github Actions

Lint Check for Syntax Errors

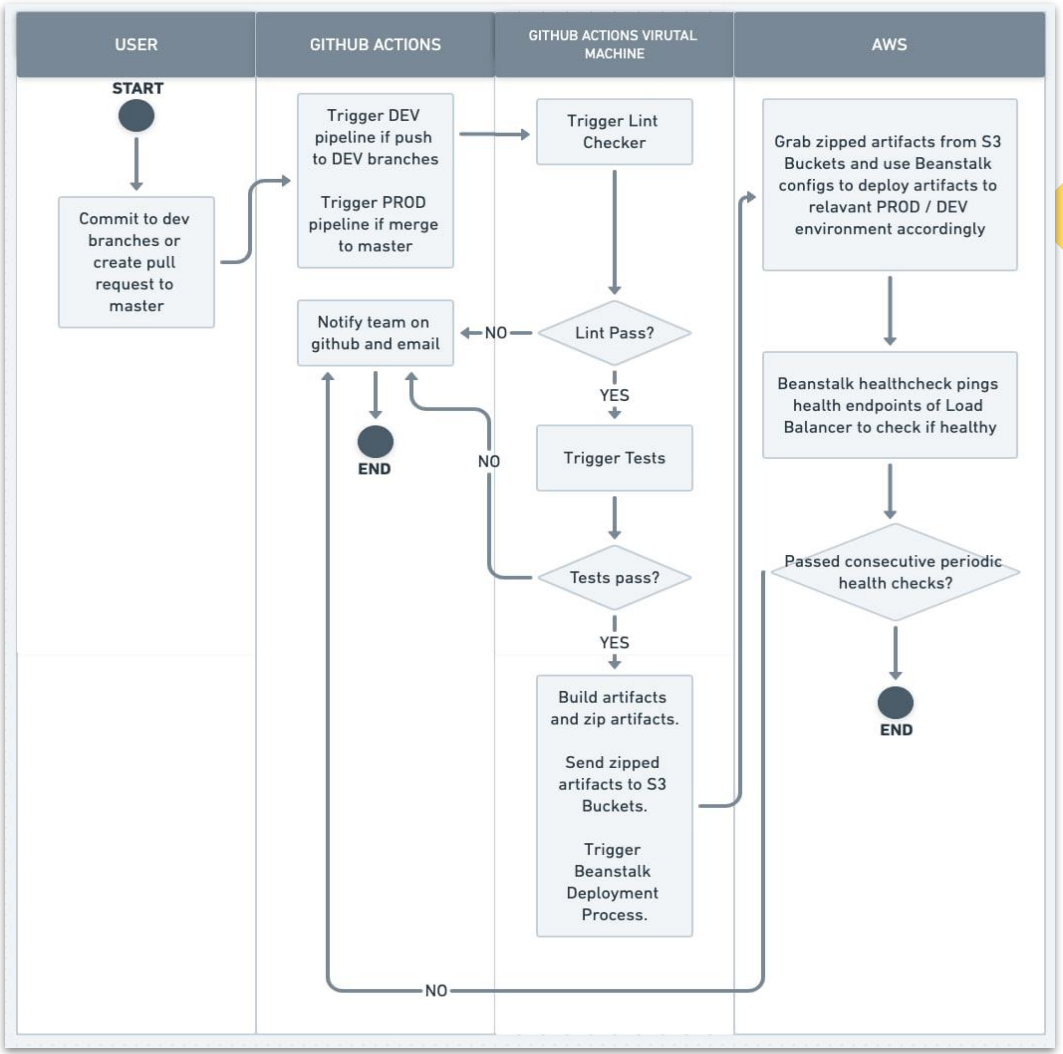
SUCCESS

Run all test cases

SUCCESS

aws

Deploy on AWS


USER
GITHUB ACTIONS
GITHUB ACTIONS VIRUTAL MACHINE
AWS
START
Commit to dev branches or create pull request to master
Trigger DEV pipeline if push to DEV branches
Trigger PROD pipeline if merge to master
Trigger Lint Checker
Lint Pass?
NO
Notify team on github and email
END
YES
Trigger Tests
NO
Tests pass?
YES
Build artifacts and zip artifacts.
Send zipped artifacts to S3 Buckets.
Trigger Beanstalk Deployment Process.
NO
Grab zipped artifacts from S3 Buckets and use Beanstalk configs to deploy artifacts to relavant PROD / DEV environment accordingly
Beanstalk healthcheck pings health endpoints of Load Balancer to check if healthy
Passed consecutive periodic health checks?
END

PIPELINE CODE WALKTHROUGH & DEMO

Maintainability Design and Implementation

HashiCorp Terraform

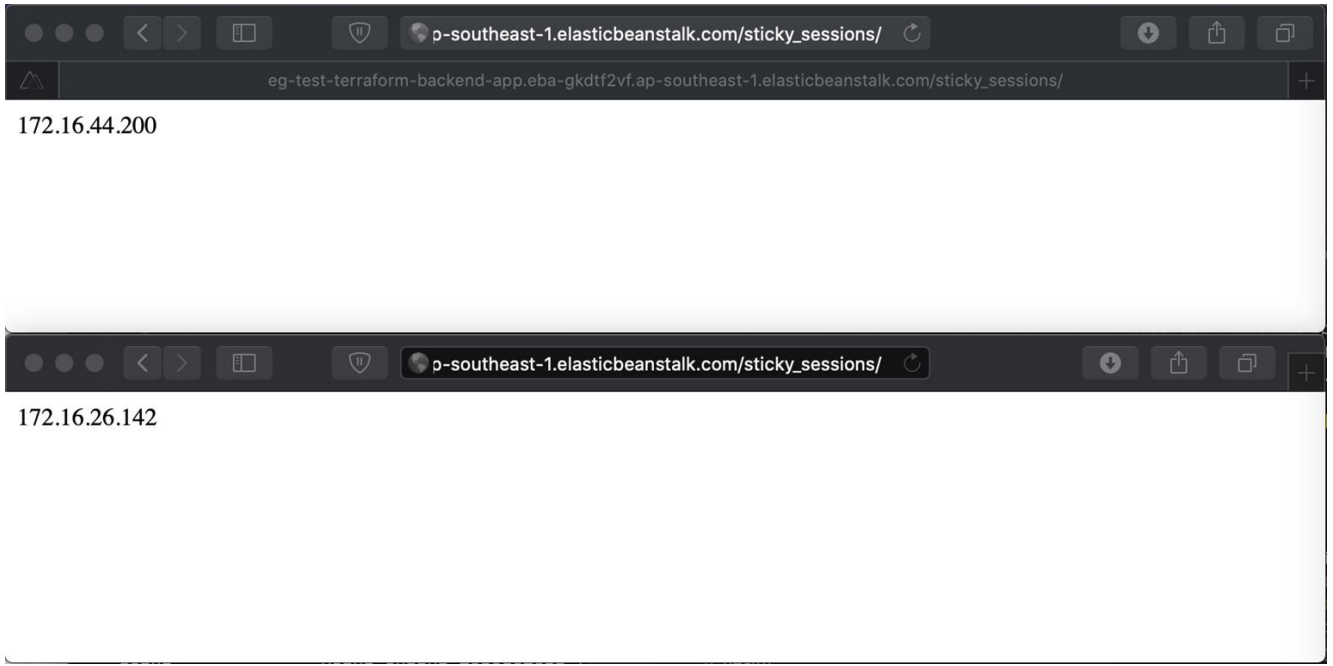
Availability

Availability Design and Implementation - Instance and DB

- EC2 instance availability
- RDS availability
- Demo time

Availability Design and Implementation - Sticky Sessions

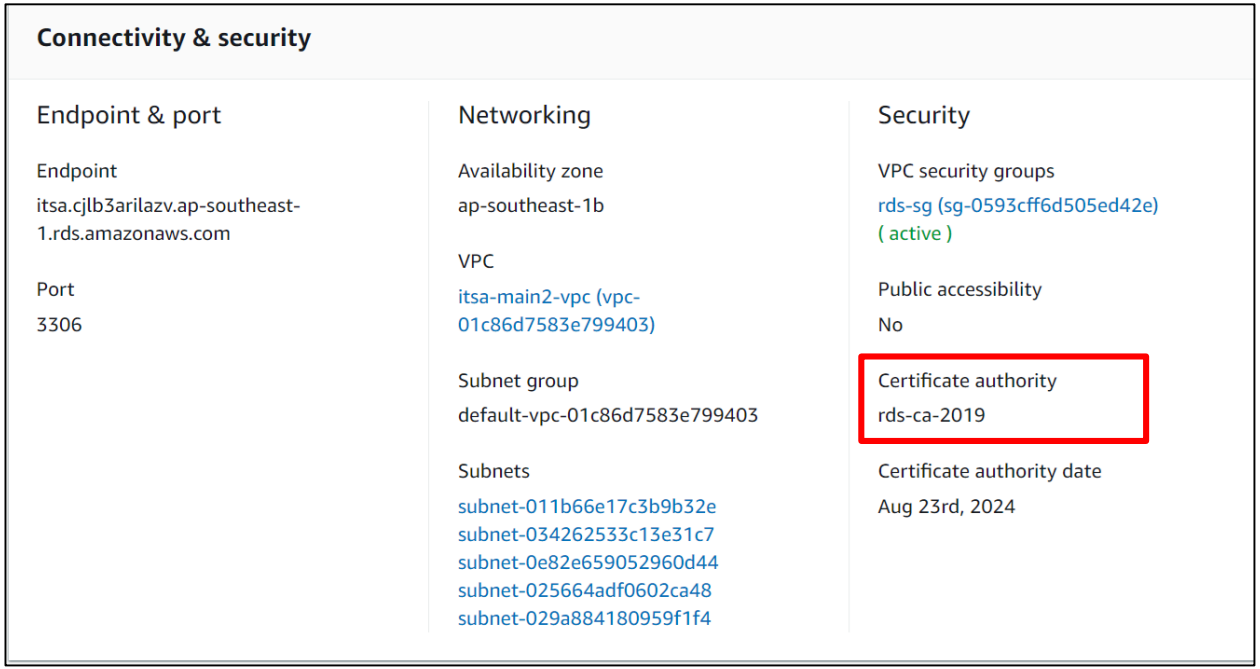
- We did it
- Not very applicable in our scenario



Security

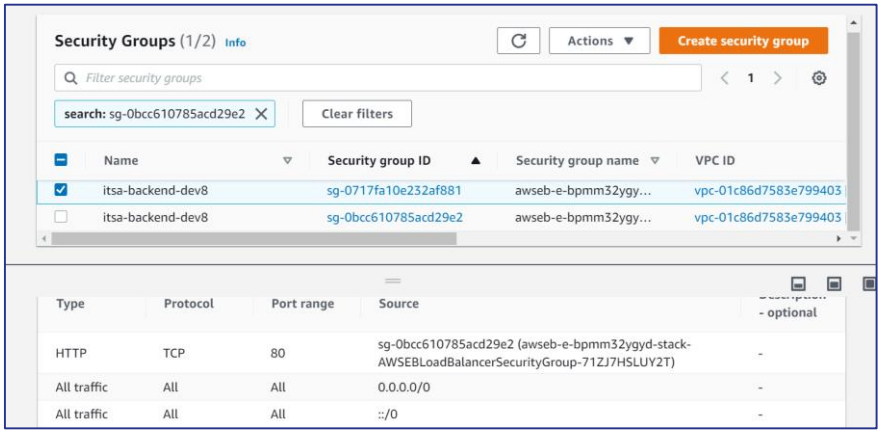
Security Design and Implementation

Feature 1: Ensure application data is sent over an encrypted network tunnel

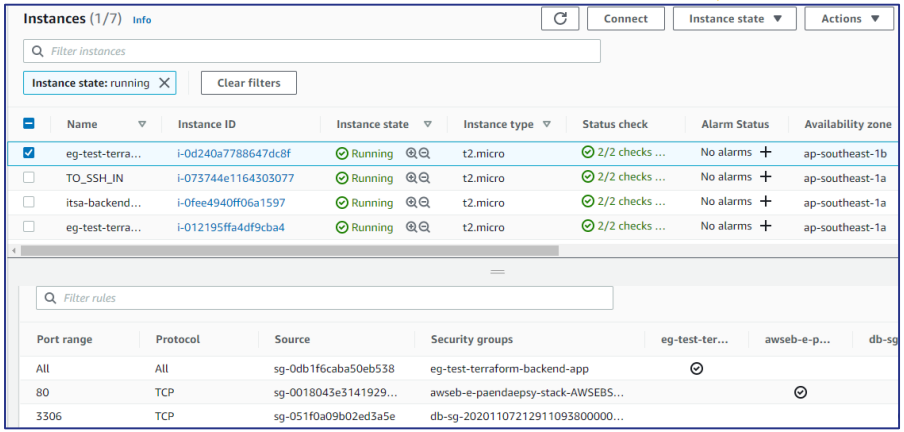


Security Design and Implementation

Feature 2: Ensuring proper access control between components



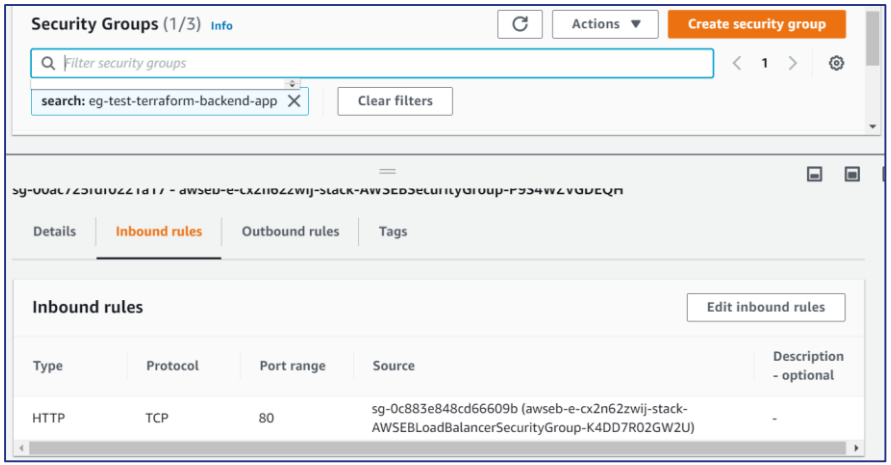
ELB Security Groups



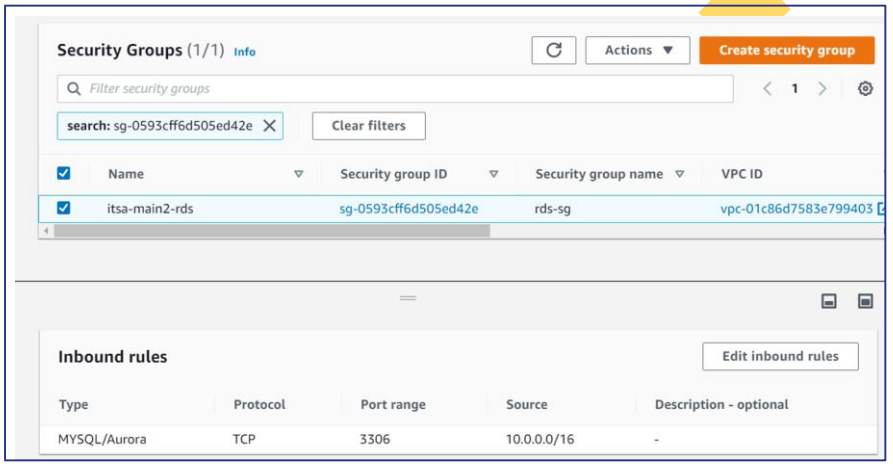
EC2 Security Groups

Security Design and Implementation

Feature 2: Ensuring proper access control between components



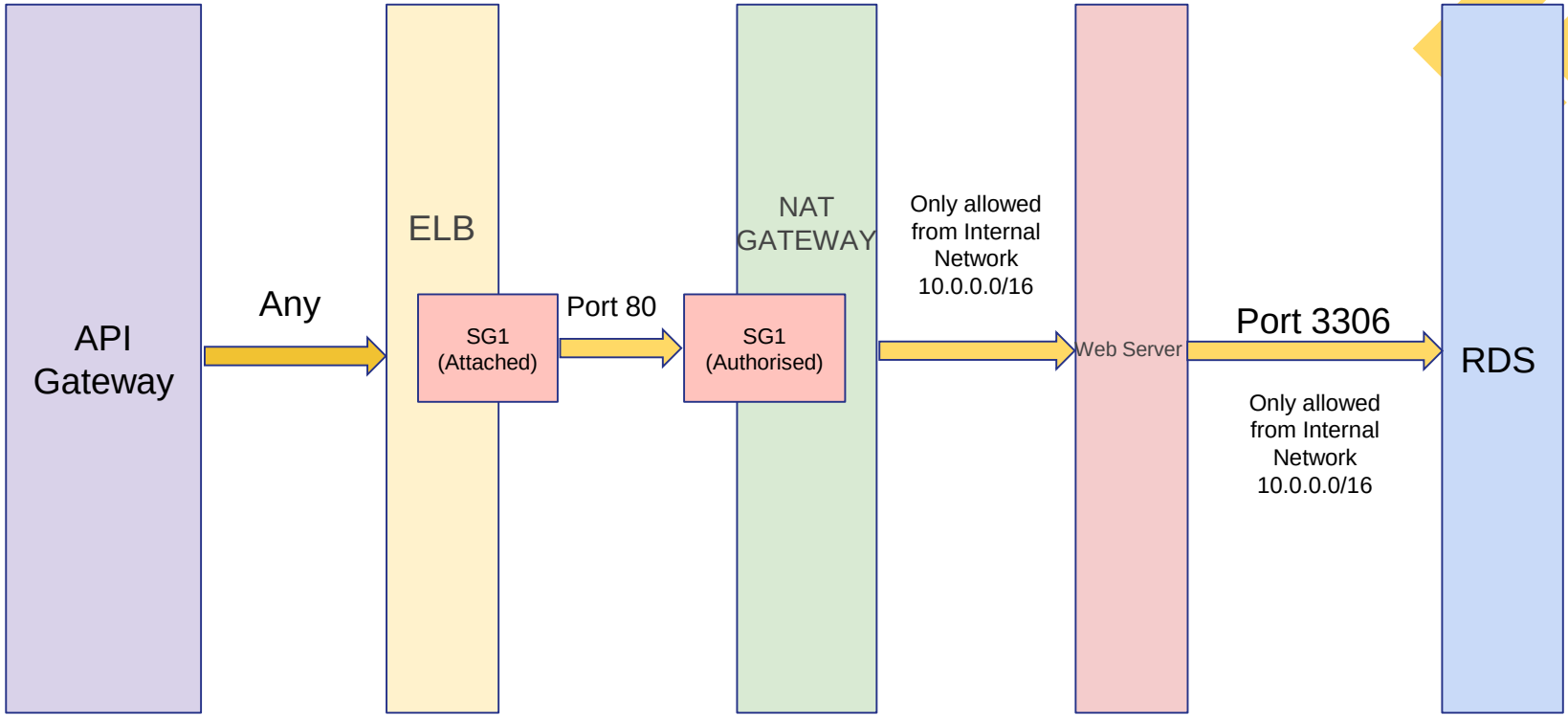
Security Group Referencing



RDS Security Group

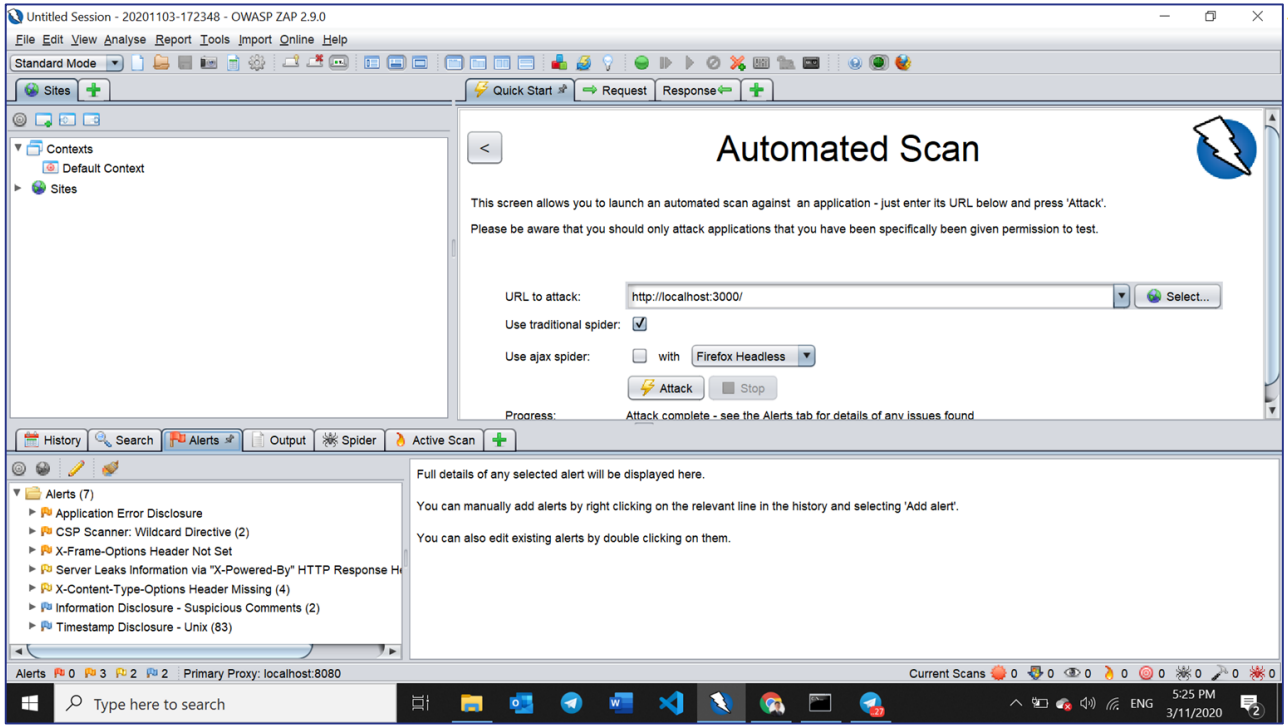
Security Design and Implementation

Feature 2: Ensuring proper access control between components



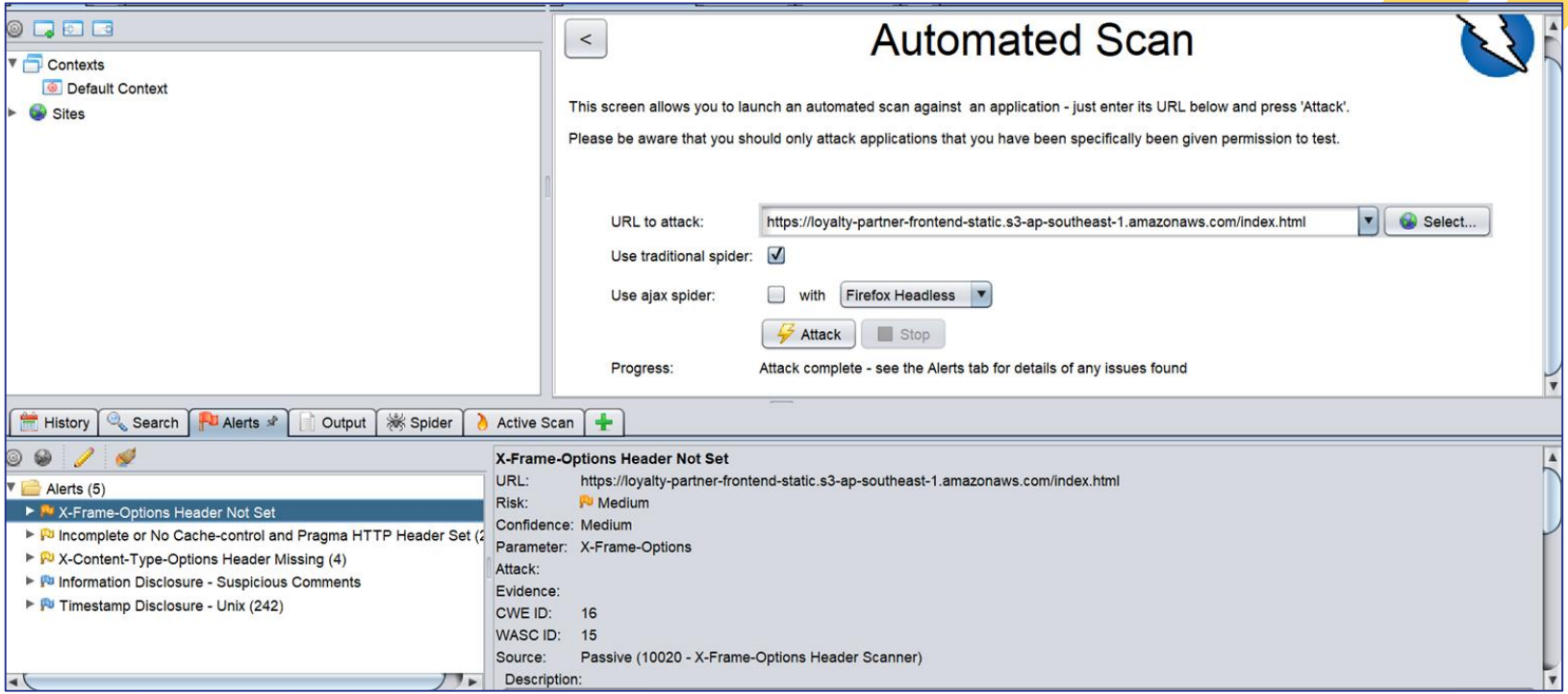
Security Design and Implementation

Feature 3: Penetration Testing using OWASP Zap (Localhost)



Security Design and Implementation

Feature 3: Penetration Testing using OWASP Zap (Beanstalk)



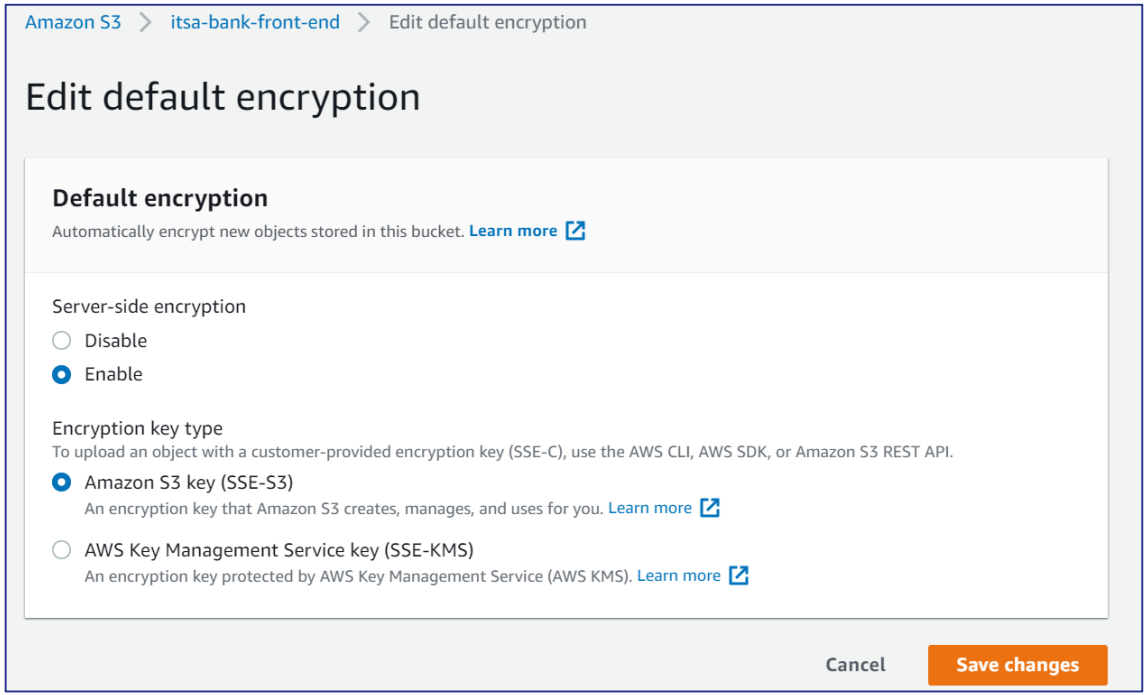
Security Design and Implementation

Feature 4: 4 Key Vulnerabilities

Asset	Potential Threat/ Vulnerability Pair	Potential Mitigation Controls
Batch/Accrual files stored in Amazon S3	Malicious form of information exposure / Unencrypted data files	Enable server side encryption in us using S3 managed keys (AES-256) or Customer master keys from KMS.

Security Design and Implementation

Feature 4: 4 Key Vulnerabilities



Security Design and Implementation

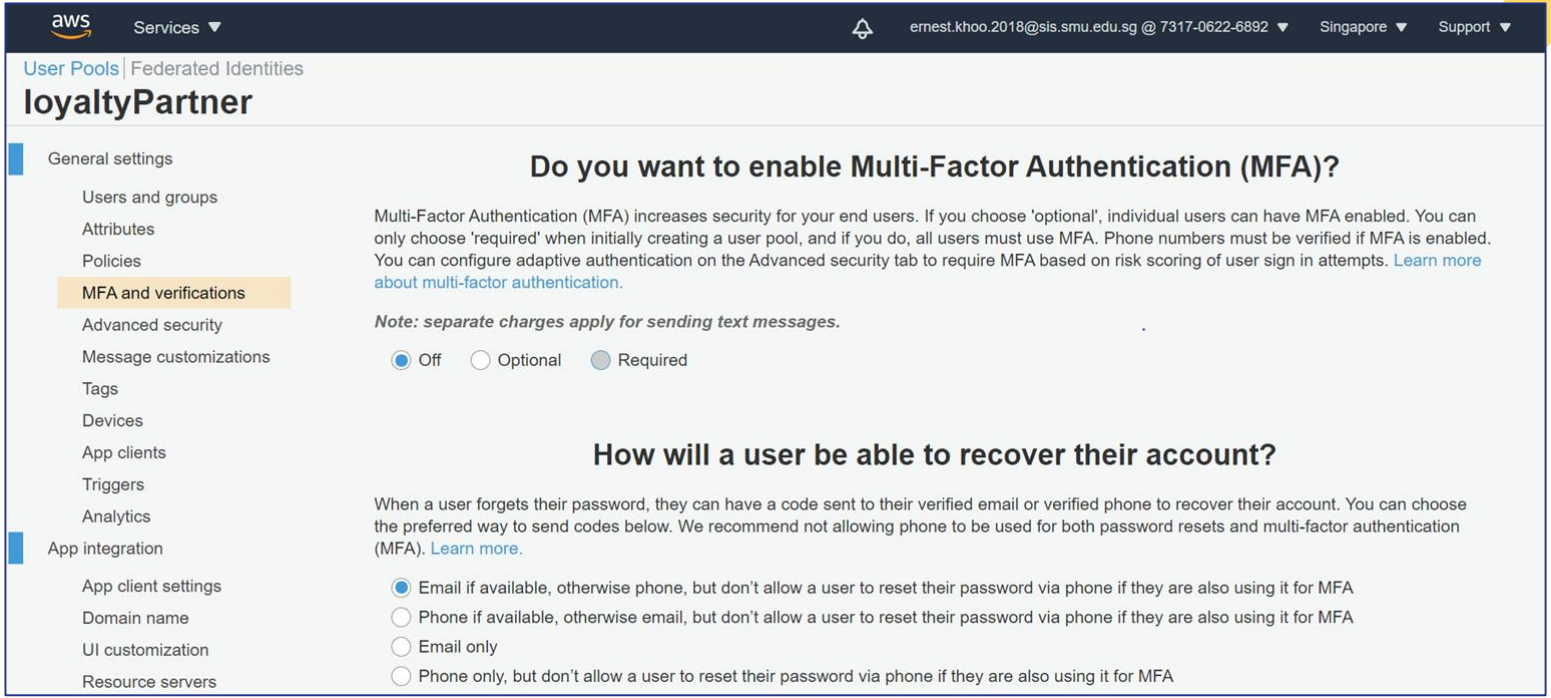
Feature 4: 4 Key Vulnerabilities

Asset	Potential Threat/ Vulnerability Pair	Potential Mitigation Controls
Loyalty Partners' Credentials	Server side request forgery (SSRF) / Token based authentication	Add multi-factor authentication (MFA) to the user pool in cognito to protect the identity of your users

AWS Cognito

Security Design and Implementation

Feature 4: 4 Key Vulnerabilities



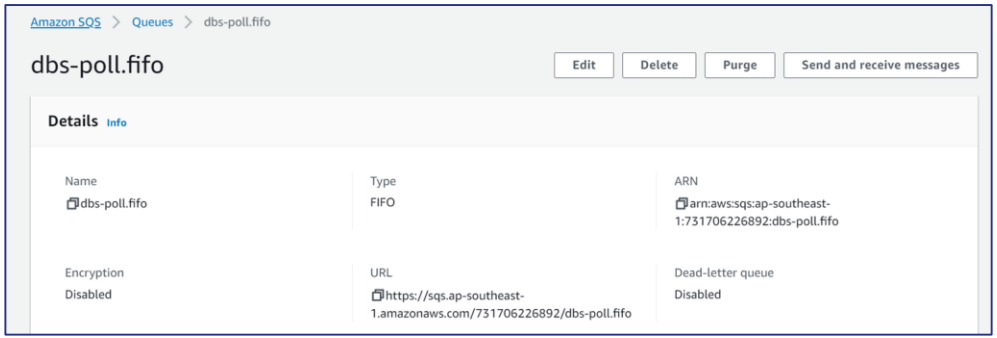
Security Design and Implementation

Feature 4: 4 Key Vulnerabilities

Asset	Potential Threat/ Vulnerability Pair	Potential Mitigation Controls
Ascenda's Database	One possible attack would be a malicious form of information exposure that could exploit the data in the database if it is too openly exposed.	We enforced a permission of least privilege where each specific bank is authorised to only poll from a specific SQS queue using IAM credentials. **[Implemented through SQS & IAM]**

Security Design and Implementation

Feature 4: 4 Key Vulnerabilities



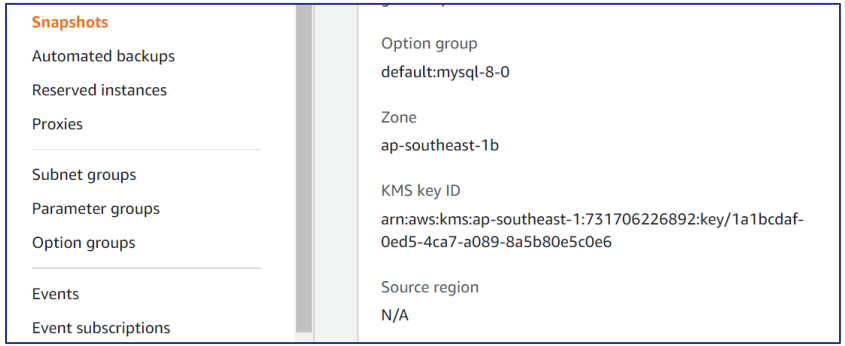
Security Design and Implementation

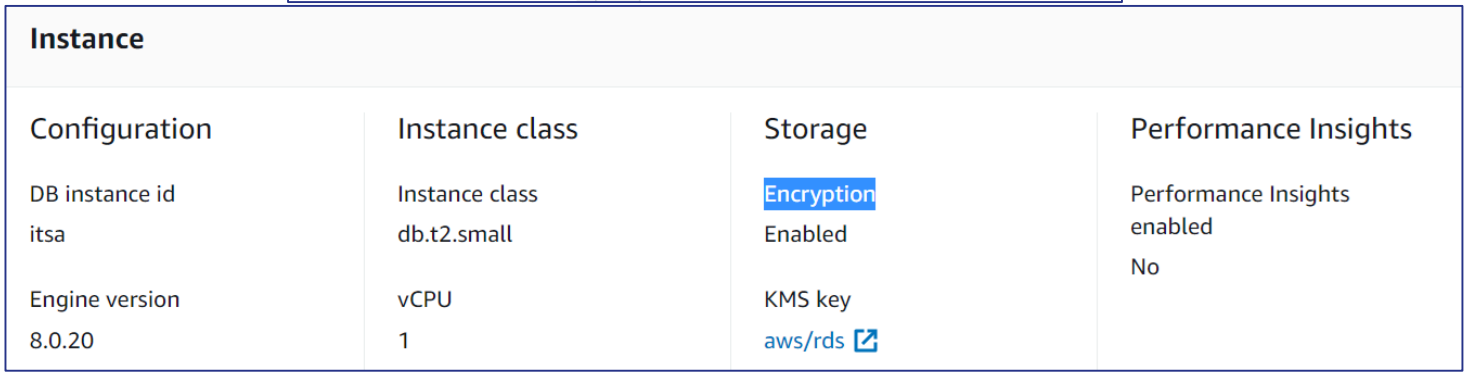
Feature 4: 4 Key Vulnerabilities

Asset	Potential Threat/ Vulnerability Pair	Potential Mitigation Controls
Users' Credentials, Membership & Transaction information	One possible attack would be a malicious form of information exposure that could exploit the unencrypted RDS data and snapshots.	Ensure RDS data and snapshots are encrypted. This ensures data at rest and snapshots are encrypted.

Security Design and Implementation

Feature 4: 4 Key Vulnerabilities





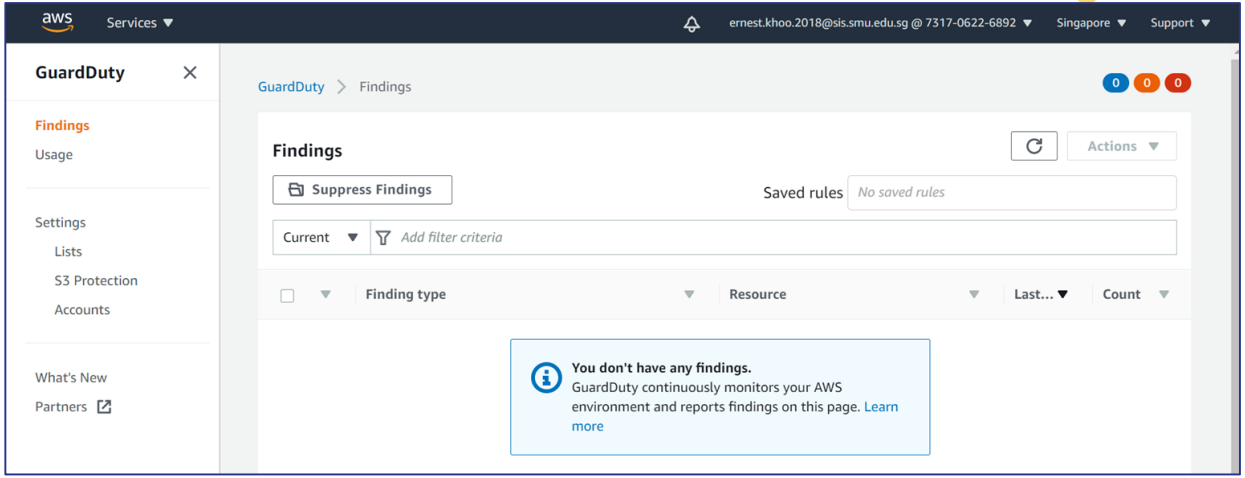
Security Design and Implementation

Feature 5: Other Security Designs: AWS WAF

Amazon IP reputation list This group contains rules that are based on Amazon threat intelligence. This is useful if you would like to block sources associated with bots or other threats.	25	Add to web ACL Set rules action to count
Anonymous IP list This group contains rules that allow you to block requests from services that allow obfuscation of viewer identity. This can include request originating from VPN, proxies, Tor nodes, and hosting providers. This is useful if you want to filter out viewers that may be trying to hide their identity from your application.	50	Add to web ACL
Core rule set Contains rules that are generally applicable to web applications. This provides protection against exploitation of a wide range of vulnerabilities, including those described in OWASP publications.	700	Add to web ACL Set rules action to count
Linux operating system Contains rules that block request patterns associated with exploitation of vulnerabilities specific to Linux, including LFI attacks. This can help prevent attacks that expose file contents or execute code for which the attacker should not have had access.	200	Add to web ACL Set rules action to count
PHP application Contains rules that block request patterns associated with exploiting vulnerabilities specific to the use of the PHP, including injection of unsafe PHP functions. This can help prevent exploits that allow an attacker to remotely execute code or commands.	100	Add to web ACL
POSIX operating system Contains rules that block request patterns associated with exploiting vulnerabilities specific to POSIX/POSIX-like OS, including LFI attacks. This can help prevent attacks that expose file contents or execute code for which access should not been allowed.	100	Add to web ACL
SQL database Contains rules that allow you to block request patterns associated with exploitation of SQL databases, like SQL injection attacks. This can help prevent remote injection of unauthorized queries.	200	Add to web ACL Set rules action to count

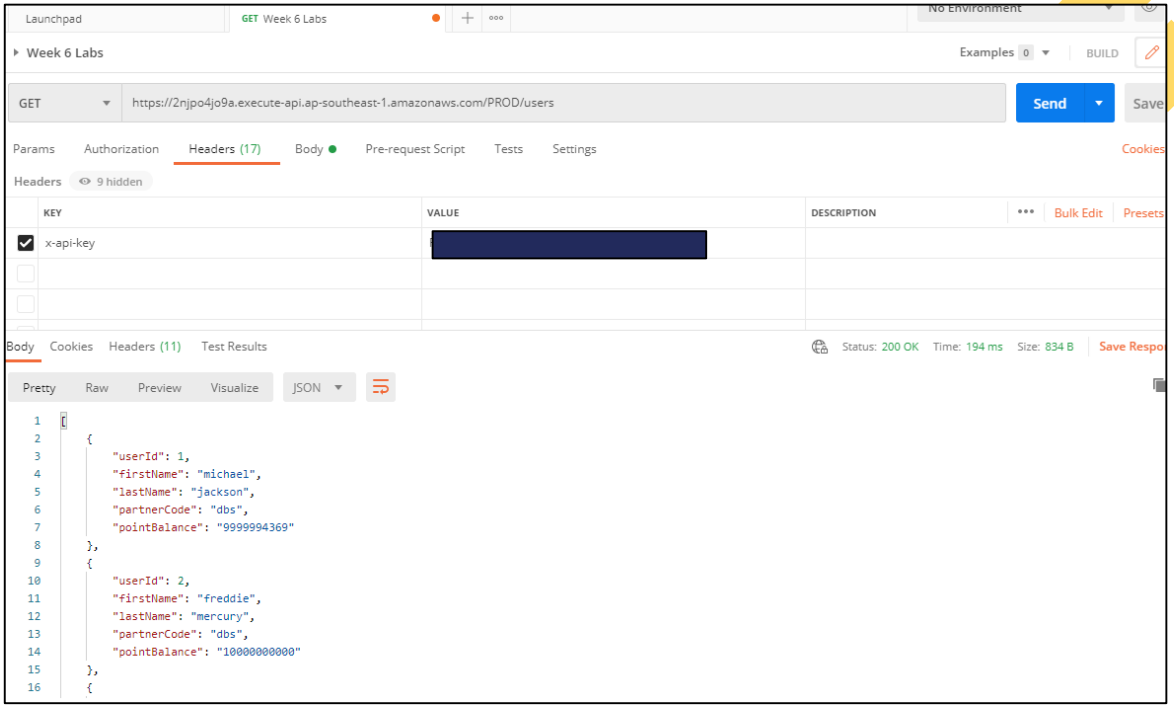
Security Design and Implementation

Feature 5: Other Security Designs: Amazon GuardDuty



Security Design and Implementation

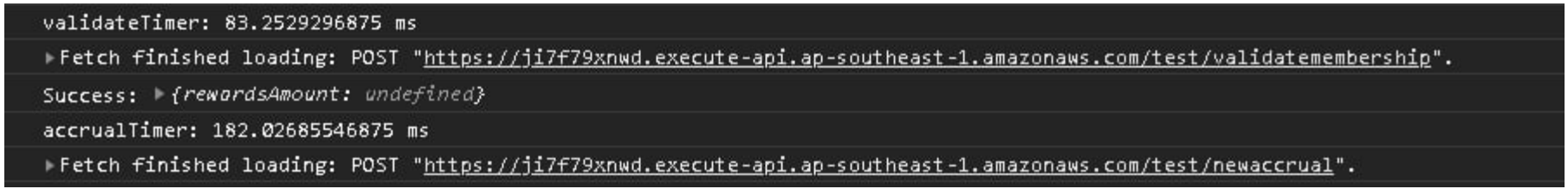
Feature 5: Other Security Designs: API Gateway



Performance

Performance Design and Implementation

Global Performance: Ensure requests time < 200ms

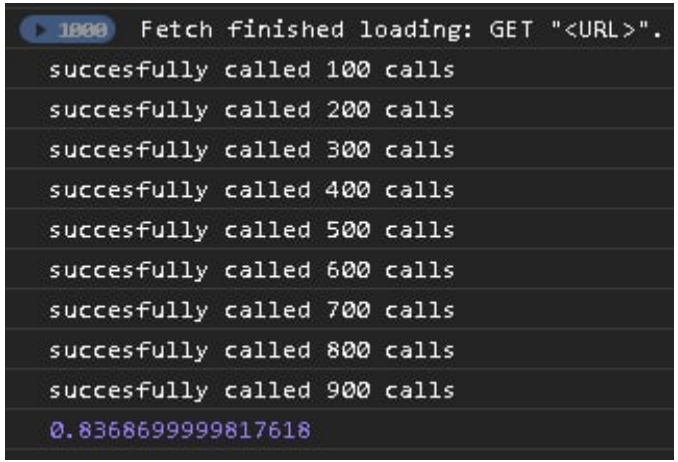


Cache Implementation

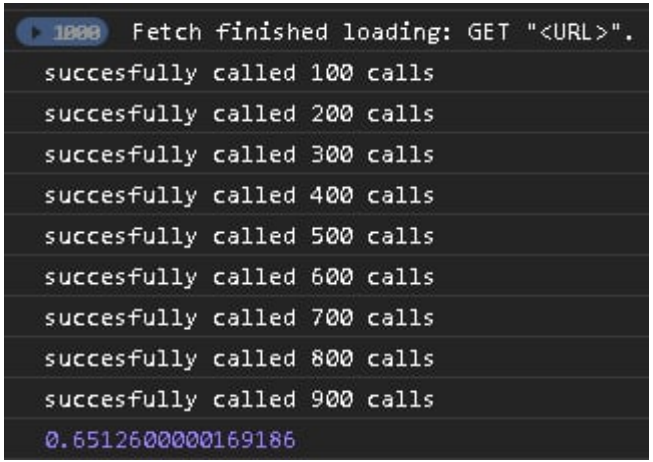
With API Gateway caching, we can reduce the number of calls and also improve the latency of requests to our API

Cache Implementation

Without Cache

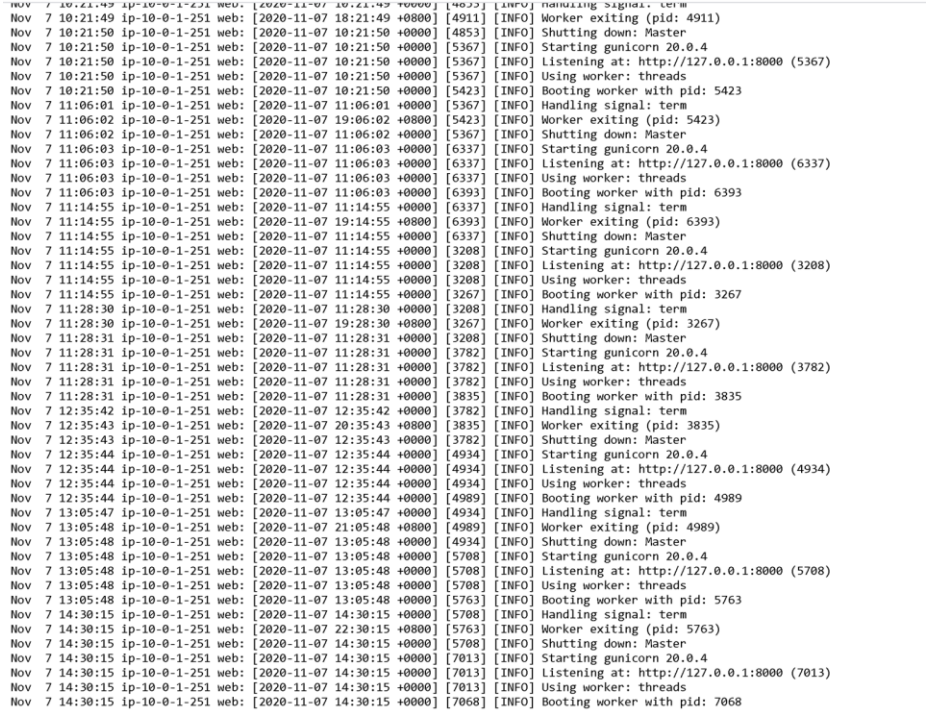


With Cache



Average time taken was calculated from 1000 requests

Parallel Execution



```
Nov  7 10:21:49 ip-10-0-1-251 web: [2020-11-07 18:21:49 +0800] [4911] [INFO] Worker exiting (pid: 4911)
Nov  7 10:21:50 ip-10-0-1-251 web: [2020-11-07 10:21:50 +0000] [4853] [INFO] Shutting down: Master
Nov  7 10:21:50 ip-10-0-1-251 web: [2020-11-07 10:21:50 +0000] [5367] [INFO] Starting gunicorn 20.0.4
Nov  7 10:21:50 ip-10-0-1-251 web: [2020-11-07 10:21:50 +0000] [5367] [INFO] Listening at: http://127.0.0.1:8000 (5367)
Nov  7 10:21:50 ip-10-0-1-251 web: [2020-11-07 10:21:50 +0000] [5367] [INFO] Using worker: threads
Nov  7 10:21:50 ip-10-0-1-251 web: [2020-11-07 10:21:50 +0000] [5423] [INFO] Booting worker with pid: 5423
Nov  7 11:06:01 ip-10-0-1-251 web: [2020-11-07 11:06:01 +0000] [5367] [INFO] Handling signal: term
Nov  7 11:06:02 ip-10-0-1-251 web: [2020-11-07 19:06:02 +0800] [5423] [INFO] Worker exiting (pid: 5423)
Nov  7 11:06:02 ip-10-0-1-251 web: [2020-11-07 11:06:02 +0000] [5367] [INFO] Shutting down: Master
Nov  7 11:06:03 ip-10-0-1-251 web: [2020-11-07 11:06:03 +0000] [6337] [INFO] Starting gunicorn 20.0.4
Nov  7 11:06:03 ip-10-0-1-251 web: [2020-11-07 11:06:03 +0000] [6337] [INFO] Listening at: http://127.0.0.1:8000 (6337)
Nov  7 11:06:03 ip-10-0-1-251 web: [2020-11-07 11:06:03 +0000] [6337] [INFO] Using worker: threads
Nov  7 11:06:03 ip-10-0-1-251 web: [2020-11-07 11:06:03 +0000] [6393] [INFO] Booting worker with pid: 6393
Nov  7 11:14:55 ip-10-0-1-251 web: [2020-11-07 11:14:55 +0000] [6337] [INFO] Handling signal: term
Nov  7 11:14:55 ip-10-0-1-251 web: [2020-11-07 19:14:55 +0800] [6393] [INFO] Worker exiting (pid: 6393)
Nov  7 11:14:55 ip-10-0-1-251 web: [2020-11-07 11:14:55 +0000] [6337] [INFO] Shutting down: Master
Nov  7 11:14:55 ip-10-0-1-251 web: [2020-11-07 11:14:55 +0000] [3208] [INFO] Starting gunicorn 20.0.4
Nov  7 11:14:55 ip-10-0-1-251 web: [2020-11-07 11:14:55 +0000] [3208] [INFO] Listening at: http://127.0.0.1:8000 (3208)
Nov  7 11:14:55 ip-10-0-1-251 web: [2020-11-07 11:14:55 +0000] [3208] [INFO] Using worker: threads
Nov  7 11:14:55 ip-10-0-1-251 web: [2020-11-07 11:14:55 +0000] [3267] [INFO] Booting worker with pid: 3267
Nov  7 11:28:30 ip-10-0-1-251 web: [2020-11-07 11:28:30 +0000] [3208] [INFO] Handling signal: term
Nov  7 11:28:30 ip-10-0-1-251 web: [2020-11-07 19:28:30 +0800] [3267] [INFO] Worker exiting (pid: 3267)
Nov  7 11:28:31 ip-10-0-1-251 web: [2020-11-07 11:28:31 +0000] [3208] [INFO] Shutting down: Master
Nov  7 11:28:31 ip-10-0-1-251 web: [2020-11-07 11:28:31 +0000] [3782] [INFO] Starting gunicorn 20.0.4
Nov  7 11:28:31 ip-10-0-1-251 web: [2020-11-07 11:28:31 +0000] [3782] [INFO] Listening at: http://127.0.0.1:8000 (3782)
Nov  7 11:28:31 ip-10-0-1-251 web: [2020-11-07 11:28:31 +0000] [3782] [INFO] Using worker: threads
Nov  7 11:28:31 ip-10-0-1-251 web: [2020-11-07 11:28:31 +0000] [3835] [INFO] Booting worker with pid: 3835
Nov  7 12:35:42 ip-10-0-1-251 web: [2020-11-07 12:35:42 +0000] [3782] [INFO] Handling signal: term
Nov  7 12:35:43 ip-10-0-1-251 web: [2020-11-07 20:35:43 +0800] [3835] [INFO] Worker exiting (pid: 3835)
Nov  7 12:35:43 ip-10-0-1-251 web: [2020-11-07 12:35:43 +0000] [3782] [INFO] Shutting down: Master
Nov  7 12:35:44 ip-10-0-1-251 web: [2020-11-07 12:35:44 +0000] [4934] [INFO] Starting gunicorn 20.0.4
Nov  7 12:35:44 ip-10-0-1-251 web: [2020-11-07 12:35:44 +0000] [4934] [INFO] Listening at: http://127.0.0.1:8000 (4934)
Nov  7 12:35:44 ip-10-0-1-251 web: [2020-11-07 12:35:44 +0000] [4934] [INFO] Using worker: threads
Nov  7 12:35:44 ip-10-0-1-251 web: [2020-11-07 12:35:44 +0000] [4989] [INFO] Booting worker with pid: 4989
Nov  7 13:05:47 ip-10-0-1-251 web: [2020-11-07 13:05:47 +0000] [4934] [INFO] Handling signal: term
Nov  7 13:05:48 ip-10-0-1-251 web: [2020-11-07 21:05:48 +0800] [4989] [INFO] Worker exiting (pid: 4989)
Nov  7 13:05:48 ip-10-0-1-251 web: [2020-11-07 13:05:48 +0000] [4934] [INFO] Shutting down: Master
Nov  7 13:05:48 ip-10-0-1-251 web: [2020-11-07 13:05:48 +0000] [5708] [INFO] Starting gunicorn 20.0.4
Nov  7 13:05:48 ip-10-0-1-251 web: [2020-11-07 13:05:48 +0000] [5708] [INFO] Listening at: http://127.0.0.1:8000 (5708)
Nov  7 13:05:48 ip-10-0-1-251 web: [2020-11-07 13:05:48 +0000] [5708] [INFO] Using worker: threads
Nov  7 13:05:48 ip-10-0-1-251 web: [2020-11-07 13:05:48 +0000] [5763] [INFO] Booting worker with pid: 5763
Nov  7 14:30:15 ip-10-0-1-251 web: [2020-11-07 14:30:15 +0000] [5708] [INFO] Handling signal: term
Nov  7 14:30:15 ip-10-0-1-251 web: [2020-11-07 22:30:15 +0800] [5763] [INFO] Worker exiting (pid: 5763)
Nov  7 14:30:15 ip-10-0-1-251 web: [2020-11-07 14:30:15 +0000] [5708] [INFO] Shutting down: Master
Nov  7 14:30:15 ip-10-0-1-251 web: [2020-11-07 14:30:15 +0000] [7013] [INFO] Starting gunicorn 20.0.4
Nov  7 14:30:15 ip-10-0-1-251 web: [2020-11-07 14:30:15 +0000] [7013] [INFO] Listening at: http://127.0.0.1:8000 (7013)
Nov  7 14:30:15 ip-10-0-1-251 web: [2020-11-07 14:30:15 +0000] [7013] [INFO] Using worker: threads
Nov  7 14:30:15 ip-10-0-1-251 web: [2020-11-07 14:30:15 +0000] [7068] [INFO] Booting worker with pid: 7068
```

1. Implemented gunicorn workers
2. Handle multiple requests in parallel
3. Improve our performance drastically

Thanks!

Do You Have Any Questions?