

Florida International University
Knight Foundation School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Project Title: AWS Frontier

Team Members: Saul Castillo, George Marerro, Shadman Noor

Product Owner(s): Masoud Sadjadi

Mentor(s): Masoud Sadjadi

Instructor: Masoud Sadjadi
Final Deliverable AWS Frontier

The MIT License (MIT)
Copyright (c) 2016 Florida International University

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Abstract

The main goal of this project was to be able to create a Nextcloud AWS application that was both secure and useful for users. AWS as a service already provides many resources for a user, however, they do not provide a centralized location in which a user is able to track all of the resources they have created on their account. This can be done with AWS Frontier as we utilized AWS API to communicate with AWS and provide a monitoring panel displaying a multitude of resources on an account. Along with this, we have made the application secure by using decryption/encryption on unique user keys. All in all, maintaining security while providing concise, useful information was the goal for our project.

Table of Contents

INTRODUCTION	5
CURRENT SYSTEM	
.....5 PURPOSE OF NEW SYSTEM	5
USER STORIES	
IMPLEMENTED USER STORIES	7
PENDING USER STORIES	10
PROJECT PLAN	
HARDWARE AND SOFTWARE RESOURCES	12
SPRINTS PLAN	
13 Sprint 1	

[illegible]

38

Current System

The current system that AWS Frontier contains is a Web application hosted on NextCloud, one that is extremely customizable and easy to read for future modifications. It includes a visually appealing User interface that separates each resource into its own container for better visualization and readability. Along with this, in order to retrieve said resources from an account, the application communicates with AWS through AWS API. Using this we were able to list resources including but not limited to EC2 Instances, RDS Databases, and IAM Roles. While working with AWS, we wanted to maintain security of user information. Therefore, we implemented a Encryption/decryption system for the access and secret keys that a user must input into the application in order to have the information retrieved from their AWS account.

Purpose of New System

The primary goal throughout the semester was to create a stable AWS application from the ground up and provide a solid foundation for future developers to enhance the vision that we have created with AWS Frontier. The applications readability was very important for us as it is key for others to understand what we had coded and what the user interface consists of. Overall, we were focused on setting up future capstone students with a fine blueprint to be able to be successful in improving the application while also making a currently reliable and useful application.

Date Page 5 of 22

Final Deliverable AWS Frontier

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the AWS Frontier project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

- Review the documentation for Nextcloud and AWS.
- Look over AWS documentation and Quickstart framework.
- Setup AWS environment.
- Get the development environment set up and deploy a stack on AWS using Quickstart.
- Coordinate the area to work on the environment as a group, as well as gather appropriate documentation beforehand.
- Get to know Nextcloud features.
- Look for possible implementations of the AWS API on Nextcloud.
- Research and implement AWS resources that could be displayed on Nextcloud monitoring application. Parse API logs.

- Parse useful information from the API logs and attempt to add field in settings for AWS API keys. Look for possible implementations of the AWS API on Nextcloud.
- Look for possible implementations of the AWS API on Nextcloud.
- Research and implement more AWS resources that could be displayed on Nextcloud monitoring application.
- Give the front end of the application more structure.
- Add more features to the Nextcloud app.
- Design a better UI and make the experience for a user feel modernized.
- Improve the look of the containers and the content within them.
- Update UI from list of possible options.

Pending User Stories

- Organize the parsed contents of the cloudtrail events.
- Add an option in the sidebar to hide unwanted features
- Add option to expand and minimize selected feature containers

Date Page 6 of 22
Final Deliverable AWS Frontier

Project Plan

This section describes the planning that went into the realization of this project. This project incorporated agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

Software	Languages
<ul style="list-style-type: none"> • Docker • Visual Studio Code • Ubuntu • Github • Nextcloud • AWS 	<ul style="list-style-type: none"> • HTML • JavaScript • PHP • CSS

Date Page 7 of 22
Final Deliverable AWS Frontier

Sprints Plan

Sprint 1

In this sprint, our team aimed to enhance our understanding and setup of essential tools. We reviewed Nextcloud's documentation to familiarize ourselves with its features and capabilities. Additionally, we planned to dive into AWS documentation, focusing on the Quickstart framework to streamline our development process. Setting up the AWS environment is also a key objective, ensuring we have a solid foundation for deploying solutions effectively. Our collective effort was directed towards establishing a robust and efficient workflow for future development tasks.

Sprint 2

Focused on setting up and optimizing our development environment. Our main objective was to establish the environment and deploy a stack on AWS using the Quickstart tool. We also coordinated as a team to work efficiently in this environment, making sure to

gather all necessary documentation in advance. Additionally, we dedicated time to familiarize ourselves with the features of Nextcloud, enhancing our overall understanding and capabilities in the development process.

Sprint 3

We set up the development environment and deployed an EC2 instance on AWS. Our next focus was exploring potential implementations of the AWS API within Nextcloud. Finally, we worked on integrating the AWS API with Nextcloud, aiming to create a seamless connection between the two platforms for more efficient workflows.

Sprint 4

Focused on integrating AWS resources with our Nextcloud monitoring application. Our primary goal was researching and implementing AWS resources, specifically parsing API logs for useful data. We also aimed to enhance the settings in our application by adding a field for AWS API keys, ensuring efficient data parsing from the API logs. Additionally, we worked on expanding the functions available in our AWS API and Nextcloud integration, striving for a more comprehensive and functional system.

Date Page 8 of 22
Final Deliverable AWS Frontier

Sprint 5

Focused on enhancing our Nextcloud monitoring application and its interface. Our primary aim was to research and implement additional AWS resources for better monitoring capabilities. We also concentrated on giving more structure to the front end of our application, improving its usability and design. Additionally, we dedicated efforts to augment security features, ensuring a more robust and secure application.

Sprint 6

Concentrated on modernizing and enhancing the user interface of our application. Our main goal was to design a more user-friendly and contemporary UI. We also worked on improving the appearance of containers and their content, focusing on aesthetic and functional upgrades. Additionally, we implemented updates to the UI from a list of potential options, ensuring a more intuitive and engaging user experience.

Sprint 7

The final sprint mainly consisted of writing documentation, commenting on the code for better usability in the future, ensuring that its titles and logos within nextcloud were presentation-ready, and creating the demo videos, presentations, and installation guides.

SYSTEM DESIGN

Architectural Overview

The "AWS Frontier" application, designed to monitor AWS resources and hosted on Nextcloud, exhibits a multi-faceted architectural approach combining both microservices and client-server models.

1. Microservices Architecture

Our application leverages a microservices architecture, particularly evident in how it interacts with various AWS services. Each AWS component (like CloudWatch, IAM, EC2, SQS, RDS, Lambda, and Cost Explorer) is treated as an independent microservice. This design choice offers several advantages:

Scalability: Each service can be scaled independently, allowing for efficient resource allocation. **Resilience:** Services are isolated, so potential issues in one service don't cascade to others.

Date Page 9 of 22

Final Deliverable AWS Frontier

Flexibility: We can update or modify individual services without significant impact on the entire application.

2. Client-Server Architecture

The application's structure follows the client-server model, divided into two main parts:

Front-end (Client): The front-end is built using mymaintemplate.php. It serves as the client, presenting the user interface and interacting with the back-end via JavaScript. **Back-end (Server):** The mainscript.js forms the back-end. It operates as the server, managing business logic, processing data, and communicating with AWS services.

Subsystem Decomposition

The "AWS Frontier" system is decomposed into several subsystems, each handling specific functionalities:

Nextcloud Environment: It acts as the hosting platform, running on a Docker container with an Ubuntu base. This subsystem is essential for local development and testing.

Front-end Subsystem: This includes HTML, CSS, and JavaScript files. It handles the presentation layer and user interactions.

Back-end Subsystem: Comprising the core of the application logic, it manages API calls to AWS services, data processing, and encryption tasks.

AWS Services Subsystem: This is a collection of AWS resources and services the application monitors and interacts with, forming the core of the monitoring capabilities.

Design Patterns Utilized

In developing "AWS Frontier," we incorporated several design patterns to enhance modularity, code reusability, and system robustness:

Model-View-Controller (MVC): Our front-end and back-end architecture follows the MVC pattern. The front-end (mymaintemplate.php) acts as the View, the back-end (mainscript.js) serves as the Controller, and AWS services represent the Model. This separation ensures a clear distinction between user interface, business logic, and data.

Singleton Pattern: To manage AWS SDK instances efficiently, we employed the Singleton pattern. It ensures a single, shared instance of the AWS SDK is created and used throughout the application, thus optimizing resource usage and consistency.

Strategy Pattern: For flexible interaction with various AWS services, the Strategy pattern is applied in the back-end. It allows dynamic selection and execution of different AWS service calls, enabling the application to adapt to various AWS APIs seamlessly.

Date Page 10 of 22

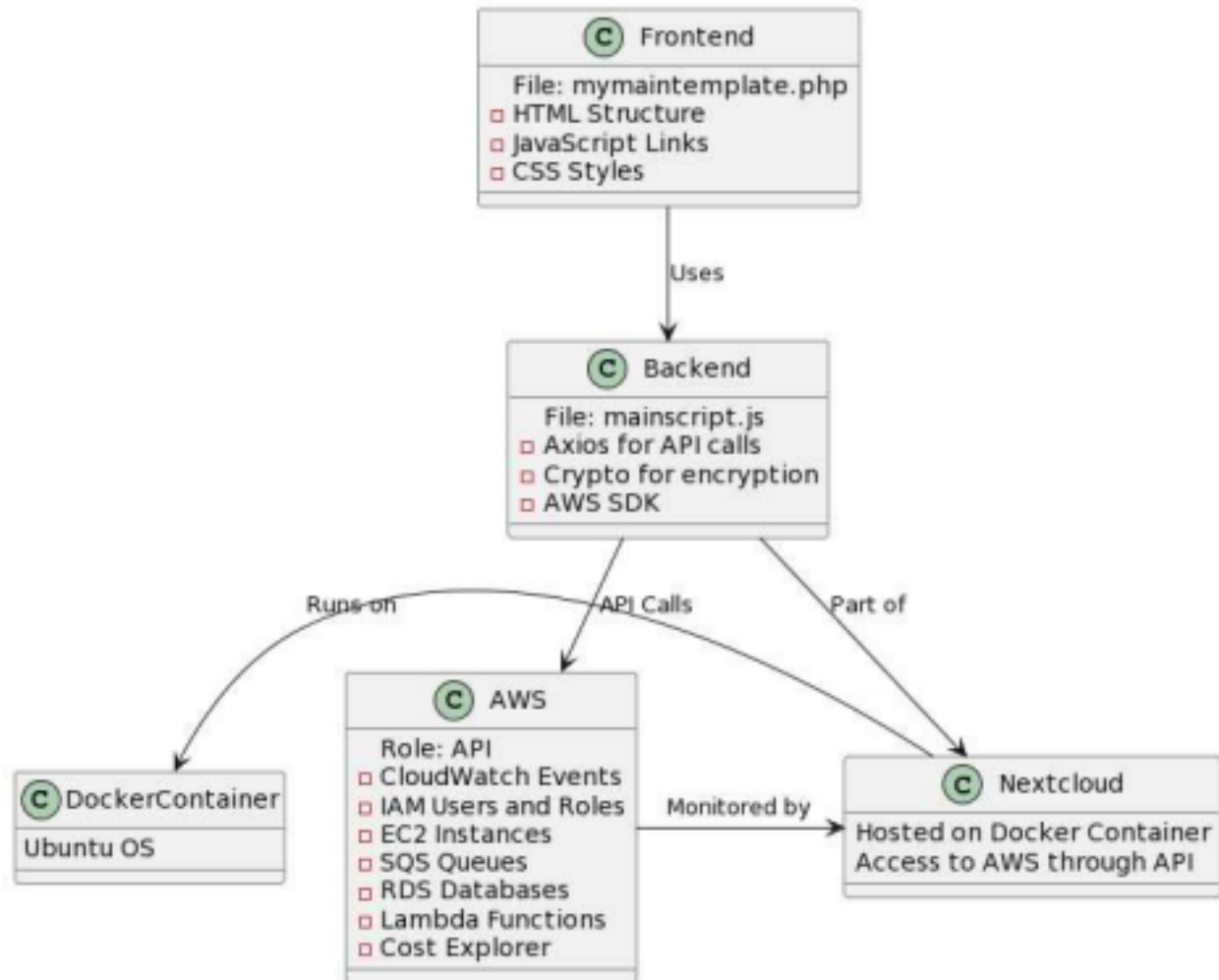
Final Deliverable AWS Frontier

Conclusion

The design decisions in "AWS Frontier" revolve around creating a scalable, flexible, and efficient monitoring tool for AWS resources. By combining microservices and

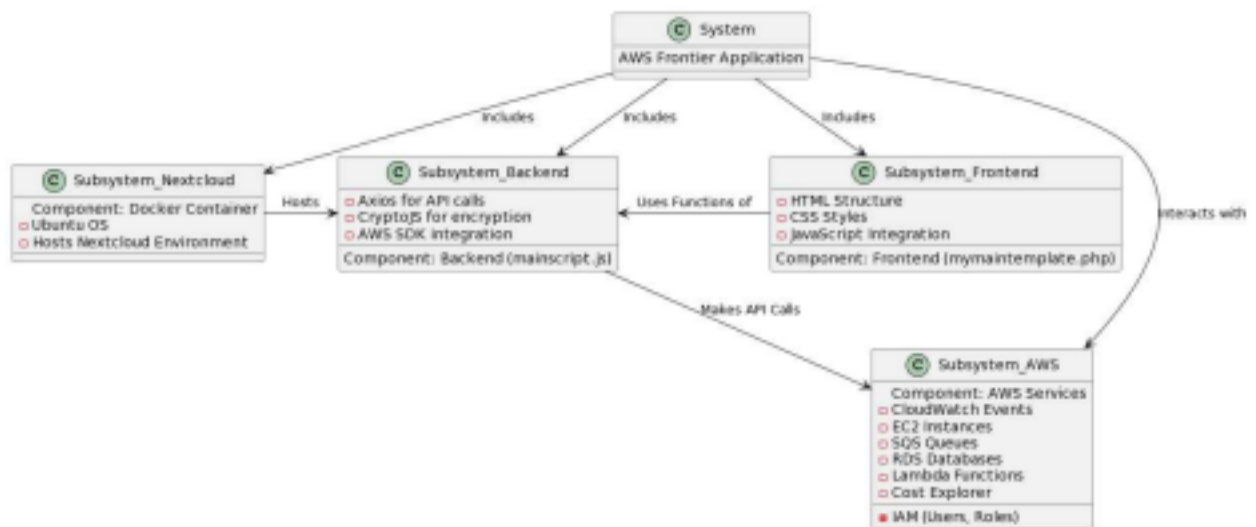
client-server architectures, along with strategic design patterns, the application achieves a high degree of modularity, robustness, and maintainability, crucial for a dynamic cloud environment.

Architectural Patterns

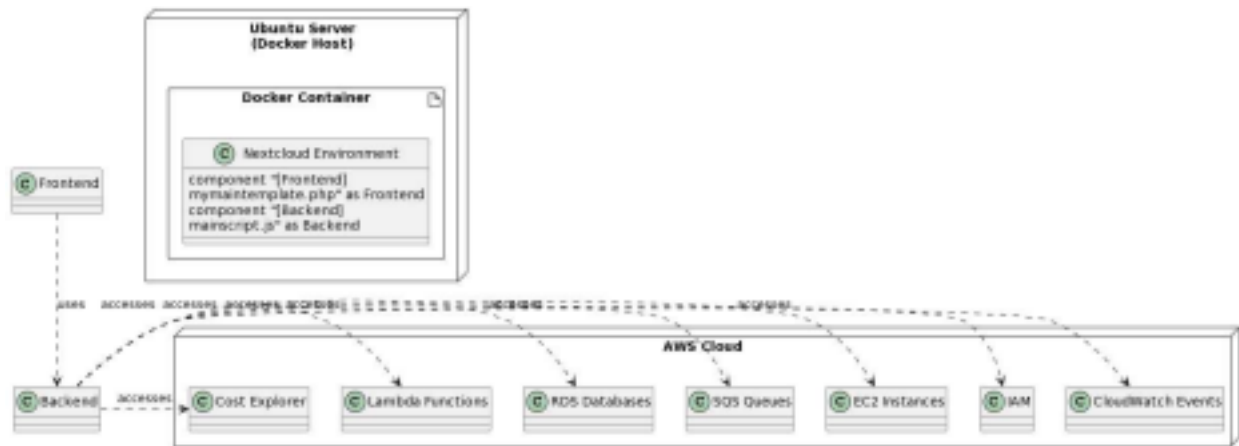


Date Page 11 of 22
Final Deliverable AWS Frontier

System and Subsystem Decomposition



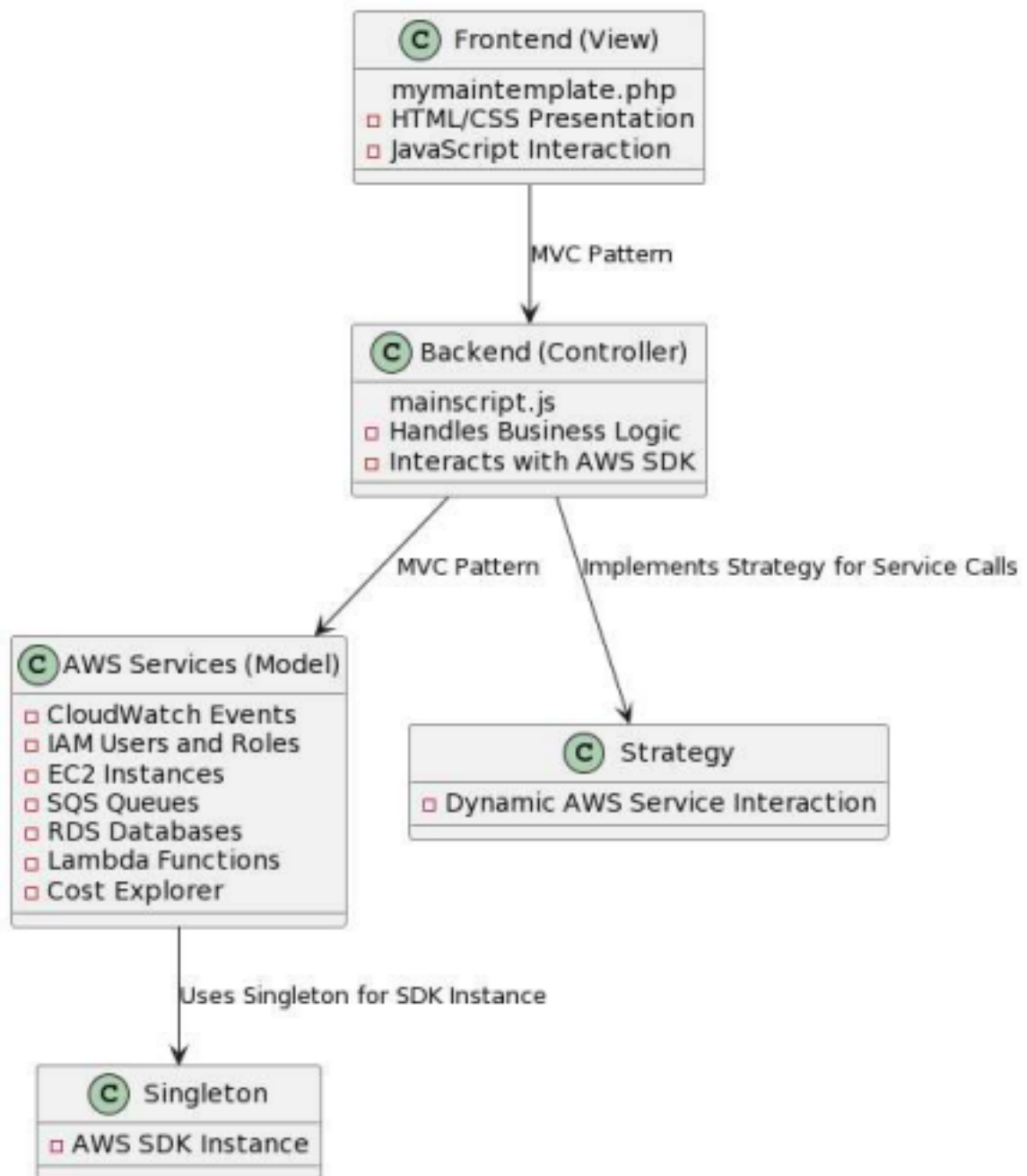
Deployment Diagram



Date Page 12 of 22

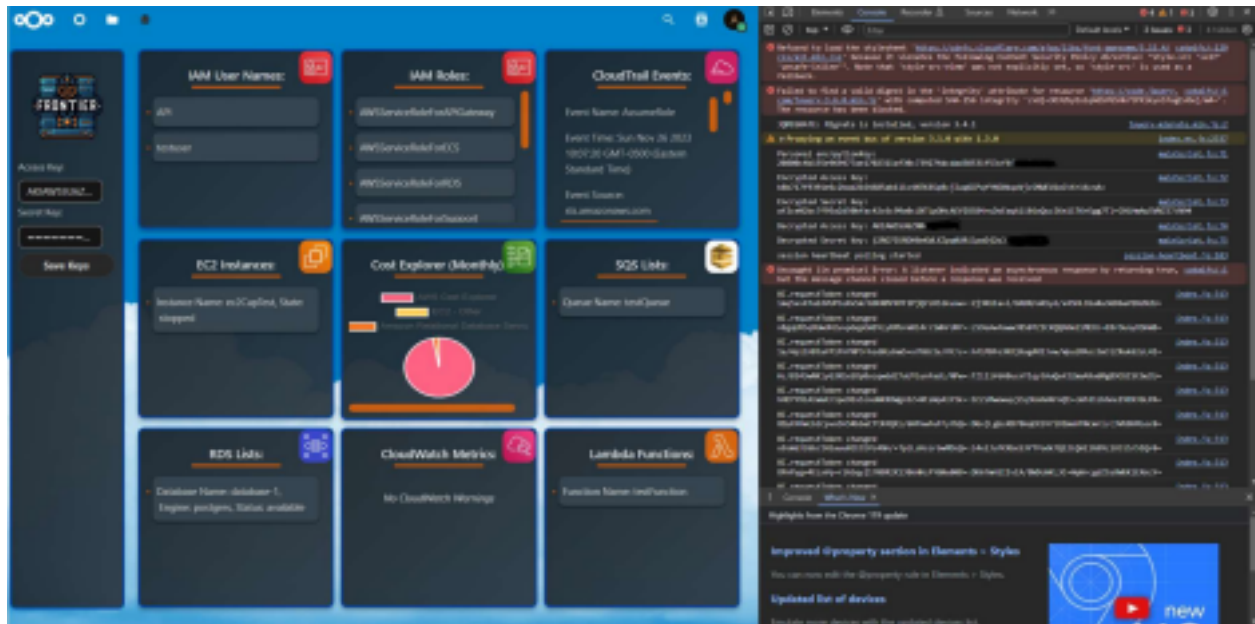
Final Deliverable AWS Frontier

Design Patterns



Date Page 13 of 22
Final Deliverable AWS Frontier

SYSTEM VALIDATION



The encryption for the Access keys was validated through checkpoints setup in the Javascript code. We can see in the console that the keys are properly encrypted and decrypted (This code is for development/illustrative purposes only, the deployed version does not have these checkpoints as it is a security hazard)

GLOSSARY

Microservices Architecture: An architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities. Each service is independently deployable and scalable.

Client-Server Architecture: A network architecture where a system is divided into two parts: a client that requests data and a server that responds to client requests. This model is a central part of web services and applications.

Nextcloud: An open-source software platform for file hosting and sharing, often used to create and manage cloud storage services.

Docker Container: A lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, runtime, system tools, libraries, and settings.

Ubuntu: An open-source operating system based on Linux, commonly used for cloud and server applications.

MVC (Model-View-Controller): A software architectural pattern for implementing user interfaces. It divides the application into three interconnected components to separate internal representations of information from the ways information is presented to and accepted from the user.

Singleton Pattern: A software design pattern that restricts the instantiation of a class to one "single" instance. This is useful when exactly one object is needed to coordinate actions across the system.

Strategy Pattern: A behavioral software design pattern that enables selecting an algorithm's behavior at runtime. It defines a family of algorithms, encapsulates each one, and makes them interchangeable.

AWS (Amazon Web Services): A subsidiary of Amazon providing on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis.

CloudWatch: A monitoring and observability service built for DevOps engineers, developers, site reliability engineers (SREs), and IT managers.

IAM (Identity and Access Management): A web service that helps securely control access to AWS resources, allowing the creation and management of AWS users and groups, and use of permissions to allow and deny their access to AWS resources.

EC2 (Elastic Compute Cloud): A part of Amazon's cloud-computing platform, AWS, that allows users to rent virtual computers on which to run their own computer applications.

SQS (Simple Queue Service): A distributed message queuing service introduced by Amazon.com in late 2004. It supports programmatic sending of messages via web service applications as a way to communicate over the Internet.

RDS (Relational Database Service): A distributed relational database service by Amazon Web Services (AWS). It is a web service running "in the cloud" designed to simplify the setup, operation, and scaling of a relational database for use in applications.

Lambda: An event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services. It is a computing service that runs code in response to events and automatically manages the computing resources required by that code.

Cost Explorer: A tool provided by AWS that enables tracking of AWS spending and usage patterns. It allows for detailed analysis of costs and usage data.

CBC (Cipher Block Chaining): A mode of operation for a block cipher, one of several methods of encrypting text. In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphertext block is dependent on all plaintext blocks processed up to that point.

IV (Initialization Vector): In cryptography, an initialization vector (IV) is an arbitrary number that can be used along with a secret key for data encryption. This number, also

called a nonce, is employed only once in any session. The IV ensures that if the same data is encrypted multiple times with the same key, the ciphertext is different each time.

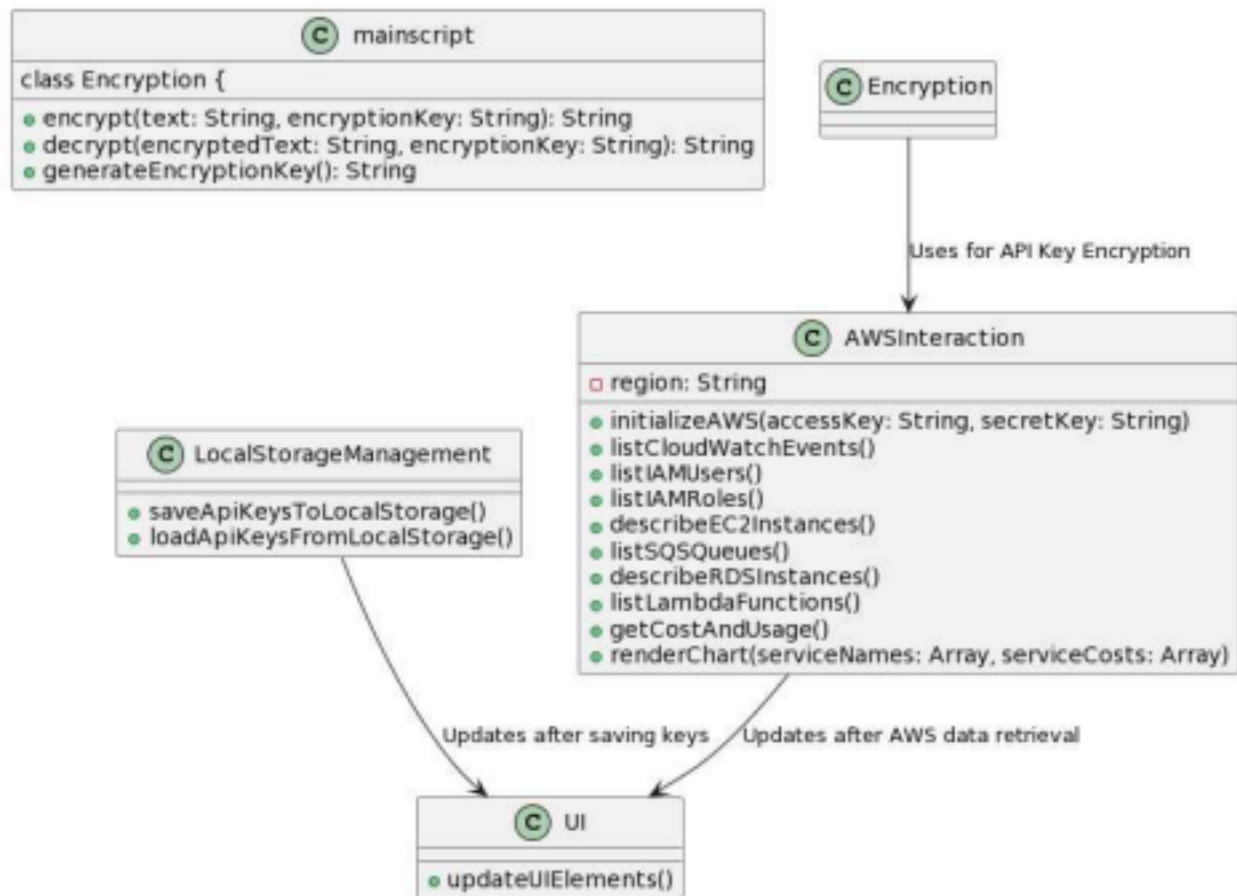
AES (Advanced Encryption Standard): A symmetric encryption algorithm widely used across the globe. AES operates on fixed-size blocks of data and uses the same key for both encrypting

Date Page 16 of 22
Final Deliverable AWS Frontier

and decrypting the data. In the context of your application, AES is used for encrypting and decrypting API keys.

APPENDIX

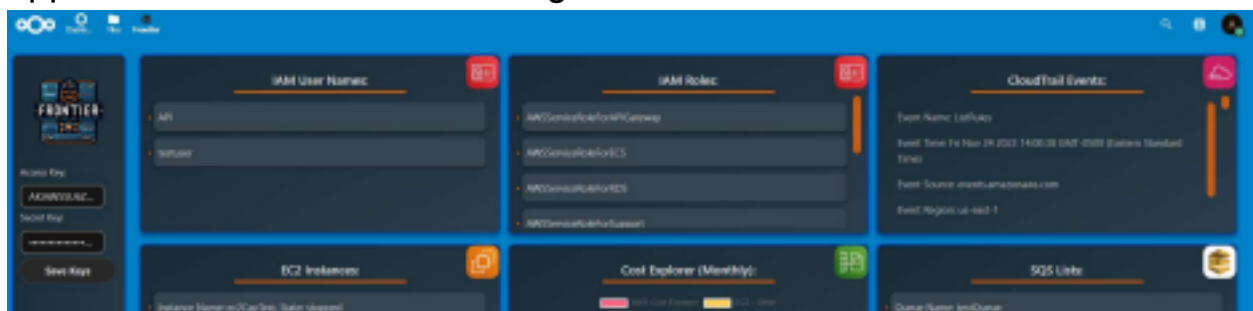
Appendix A - UML Diagrams

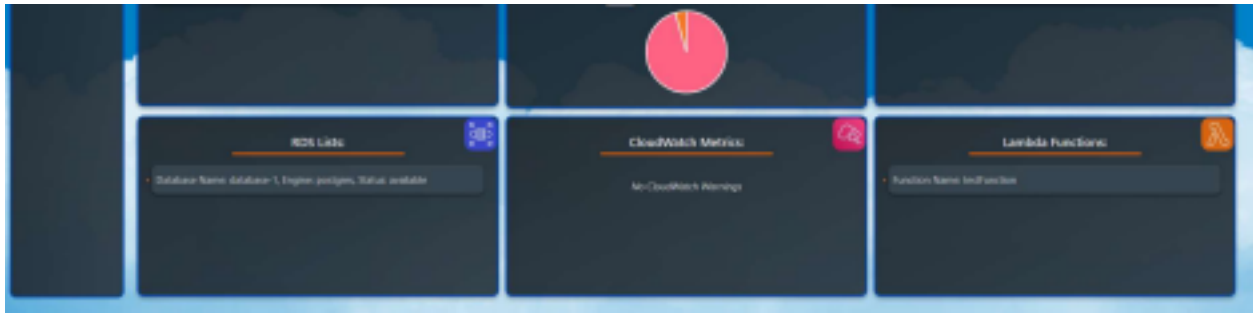


Date Page 18 of 22

Final Deliverable AWS Frontier

Appendix B - User Interface Design





The Logo seen at the top left of the page was made with DALL-E 2, a generative artificial intelligence from OpenAI.

Appendix C - Sprint Review Reports

Sprint 1

Our team successfully enhanced our understanding and setup of essential tools. We completed a thorough review of Nextcloud's documentation, gaining familiarity with its features and capabilities. Our dive into AWS documentation, focusing on the Quickstart framework, has streamlined our development process. We successfully set up the AWS environment, laying a solid foundation for deploying solutions effectively. These efforts have collectively established a robust and efficient workflow for our future development tasks.

Sprint 2

We focused on setting up and optimizing our development environment and achieved our main objective of establishing this environment. We ensured that all necessary documentation was

Date Page 19 of 22
Final Deliverable AWS Frontier

gathered. Additionally, we dedicated time to familiarizing ourselves with Nextcloud's features, enhancing our overall development process.

Sprint 3

The team set up the development environment and deployed an EC2 instance on AWS, achieving a significant milestone. We explored and identified potential implementations of the AWS API within Nextcloud. The integration of the AWS API

with Nextcloud was completed, creating a connection between the two platforms for more efficient workflows.

Sprint 4

Our focus on integrating AWS resources with our Nextcloud monitoring application was fruitful. We researched and implemented AWS resources, particularly parsing API logs for useful data. The application settings were enhanced by adding a field for AWS API keys, facilitating efficient data parsing. Moreover, we expanded the functions available in our AWS API and Nextcloud integration, achieving a more comprehensive and functional system.

Sprint 5

We concentrated on enhancing our Nextcloud monitoring application and its interface. Additional AWS resources were researched and implemented, improving monitoring capabilities. The front end of our application now has more structure, which has improved its usability and design. We also augmented security features, ensuring a robust and secure application.

Sprint 6

Successfully designed a more user-friendly and contemporary UI. The appearance of containers and their content was improved, focusing on both aesthetic and functional upgrades. We also implemented updates to the UI from a list of potential options.

Sprint 7

Finalized documentation and code commenting for future usability. We ensured that titles and logos within Nextcloud were presentation-ready. Additionally, we created demo videos, presentations, and installation guides, effectively preparing our project for presentation and ease of use.

documents

1. The first step for usage and modification of the app is to set up the development environment for Nextcloud. Begin by opening the tutorial below:
"Tutorial: Setting up a Development Environment."
<https://cloud.nextcloud.com/s/iyNGp8ryWxc7Efa?path=%2F1%20Setting%20up%20a%20development%20environment>
2. Choose the tutorial for your operating system and follow it to completion
3. Locate Your App Folder: First, navigate to the nextcloud-docker-dev folder. Inside it, enter the workspace/server/apps-extra directory.
4. Transfer the app into Nextcloud: Place the entire "catgifs" folder found in the "code" section of the final deliverable into the apps-extra directory.
5. Access Nextcloud: Open your web browser and log in to your Nextcloud instance.
6. Navigate to Apps Section
7. Click on your profile picture or avatar, located in the top right corner of the Nextcloud interface.
8. From the dropdown menu, select "Apps." This will take you to the 'Apps' section of Nextcloud.
9. In the 'Apps' section, look for "Frontier" under the 'Your apps' category.
10. To activate AWS Frontier, click on the "Enable" button. Click this, followed by the "Enable" button to proceed.
11. Confirmation: Once enabled, your app should now be visible in the top menu bar of Nextcloud, identifiable by a small AWS Frontier logo.
12. Once the app and environment have been setup, follow our AWS User Manual guide to learn how to add the necessary permissions and get the AWS API key.
https://www.youtube.com/embed/PTPvW1U90j8?si=qmsZVv_KtXnuktrA Date

REFERENCES

"API Reference." AWS Lambda, Amazon Web Services, 2023,
https://docs.aws.amazon.com/lambda/latest/dg/API_Reference.html. 27 November 2023.



"This image was created with the assistance of DALL-E 2." OpenAI, 2023,
<https://help.openai.com/en/articles/6640875-how-should-i-credit-dall-e-in-my-work>. 27 November 2023.

"Tutorial: Setting up a Development Environment." Nextcloud,
<https://cloud.nextcloud.com/s/iyNGp8ryWxc7Efa?path=%2F1%20Setting%20up%20a%20development%20environment>. 22 November 2023.

"Tutorial: Developing a Simple Interface-Only App." Nextcloud,
<https://cloud.nextcloud.com/s/iyNGp8ryWxc7Efa?path=%2F3%20Developing%20a%20simple%20interface-only%20app>. 7 September 2023.

