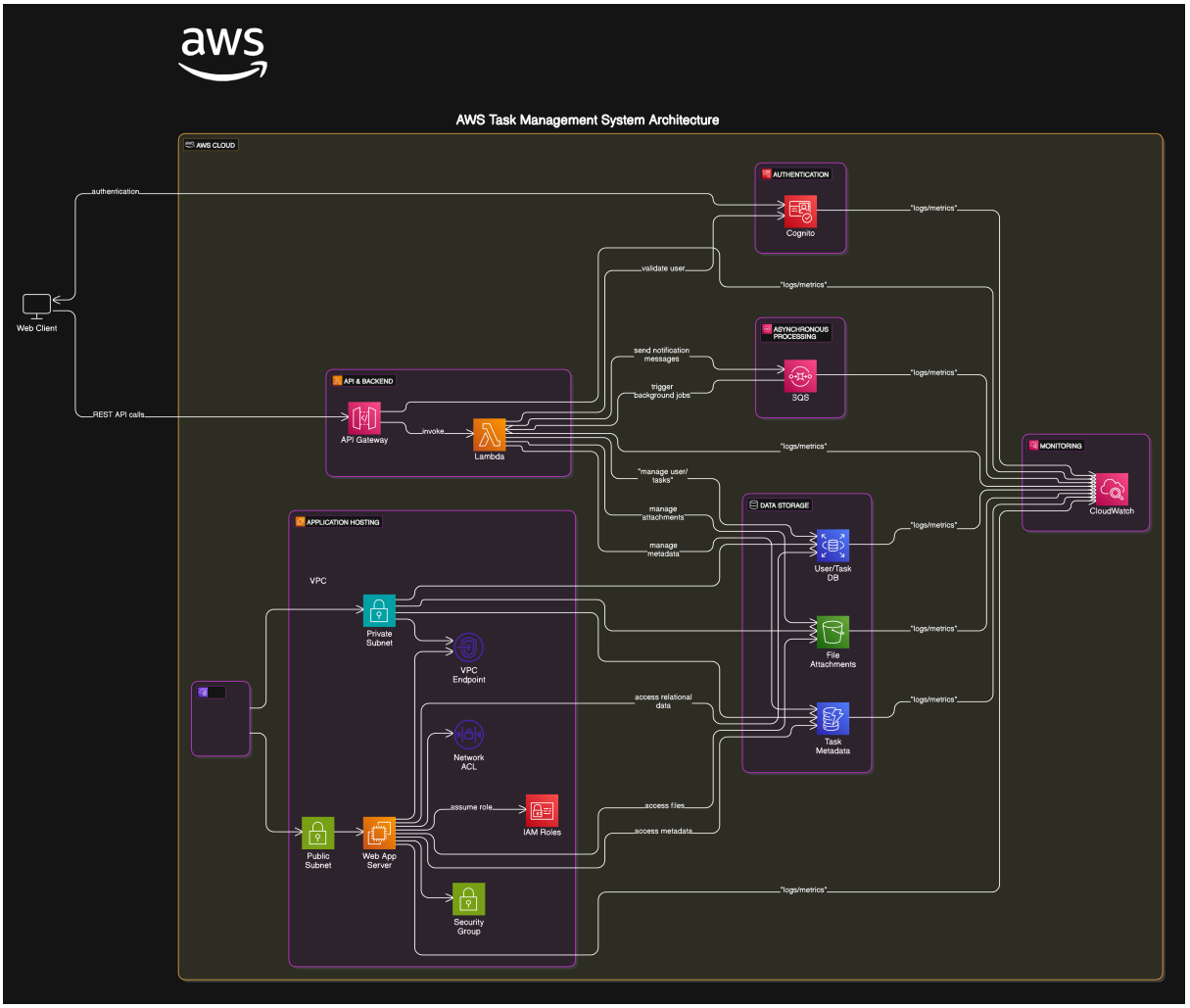
AWS Documentation

Team name:
Cloud_AWS_TEAM_101

Ali Amr ID 10000652 T 6
Mostafa Abuzahra 10001994 T 14
Mohamed gad 10001181 T 19
Mamdouh Mahfouz 10001816 T 20
Adham Hisham 10000480 T 20
Mohammed Emad 10006645 T 16
Mohammed Tarek 10002243 T16

Architecture Diagram:



Overview

The system is designed to manage tasks efficiently with the following core functionalities:

- **User Authentication and Management**
- **Task Management** (Create, update, delete tasks)
- **File Attachments** (File storage and management)
- **Notifications** (Send notifications for task updates)
- **Asynchronous Processing** (Handle background jobs)

The diagram shows the interconnection between AWS services that ensure smooth task management.

Key Components and Their Interactions

1. **AWS Cloud** (Top-Level):
    - The entire architecture resides in **AWS Cloud**, where various services are interconnected to manage and process tasks efficiently.
2. **User Authentication**:
    - **AWS Cognito** handles user sign-up, sign-in, and secure authentication.
    - The system uses Cognito to authenticate users and validate their access.
3. **API & Backend**:
    - **API Gateway** exposes the API endpoints for creating, updating, deleting, and managing tasks. It serves as the entry point for HTTP(S) requests from the web client.
    - **AWS Lambda** functions act as the backend logic processor for various tasks, including:
        - Handling API requests.
        - Performing operations like task management, updating metadata, and sending notifications.
    - The **Lambda** functions are triggered by API Gateway.
4. **Application Hosting**:
    - **VPC (Virtual Private Cloud)**: The hosting of the web application is done within a secure **VPC**.
    - The **Private Subnet** hosts resources that don’t require public access.
    - The **Public Subnet** hosts the **Web App Server**, which communicates with the backend services.
    - The web application connects with the backend API Gateway through the VPC.
    - **IAM Roles** are configured to provide secure access to the required AWS services for the application.
5. **Data Storage**:
    - **User/Task DB**: Relational data such as user profiles and task metadata is stored in **Amazon RDS**.

- **File Attachments**: Files uploaded by users are stored in **Amazon S3**, providing secure and scalable file storage.
- **Task Metadata**: Task details and other metadata are stored in **Amazon DynamoDB** for fast access and performance.

6. **Asynchronous Processing**:
   - **Amazon SQS (Simple Queue Service)** is used for managing background jobs such as sending notifications.
   - This enables the system to process messages and trigger actions asynchronously without affecting the main task flow.
7. **Monitoring**:
   - **AWS CloudWatch** monitors all activities and metrics for each service. This includes monitoring Lambda functions, API Gateway, EC2 instances, and other AWS resources.
   - It tracks errors, performance, resource utilization, and generates logs to ensure system health and performance.
   - CloudWatch provides dashboards for real-time visualization of service performance, helping administrators manage the system effectively.
8. **External Access**:
   - Users interact with the system via the **Web Client** (built using **Next.js** and **TypeScript**).
   - The **Web Client** communicates with the backend API through REST API calls via **API Gateway**.
   - The front-end and back-end components are hosted on AWS and interact with services such as **Lambda**, **Cognito**, **DynamoDB**, **S3**, and **CloudWatch**.

Flow of Operation

1. **User Authentication**:
   - A user accesses the system via the web client.
   - The user signs up or logs in via **Cognito**.
2. **Task Creation**:

- After successful authentication, the user can create tasks.
- The task information is sent to **API Gateway**, which triggers the corresponding **Lambda function** for task creation.
- The task data is stored in **DynamoDB** and **RDS** for structured and unstructured data.
- Files (if attached) are uploaded to **S3** for storage.

3. **Notifications**:
   - After task updates, **SQS** handles asynchronous notifications (e.g., via email) to keep users informed about task status changes.
4. **Asynchronous Processing**:
   - Operations such as notification delivery are managed through **SQS** queues, ensuring they don't block the main task processing flow.
5. **Monitoring**:
   - **CloudWatch** continuously tracks the performance of all services, ensuring system health.
   - Dashboards and alarms are set up to alert the team of any issues with the system (e.g., high error rates, low performance, etc.).

Setup Guide: Step-by-Step Instructions for Deploying the Application on AWS

1. Prerequisites

Before starting the deployment, ensure you have the following tools installed:

- **AWS CLI**: Install and configure the AWS CLI with your credentials.
- **AWS CDK**: Install AWS CDK globally on your machine:

  ```
  npm install -g aws-cdk
  ```

- **Node.js**: Required for running the frontend with Next.js.
  - Download Node.js from Node.js official website.

- **Python 3.6 or higher**: Required for deploying the backend infrastructure using AWS CDK.
  - Verify the installation with:

    ```
    python --version
    ```

- **Git**: To clone the repository.

2. Clone the Repository

Clone the project repository to your local machine:

```
git clone <repository-url>
cd <repository-folder>
```

3. Set Up AWS CDK

1. **Install Dependencies**:
   - Navigate to the **backend** directory and install the necessary dependencies:

     ```
     cd project
     npm install
     ```

   - For **Python dependencies** required for CDK:

     ```
     python -m pip install -r requirements.txt
     ```

   - For **AWS CDK** in Python:

     ```
     python -m pip install aws-cdk-lib constructs
     ```

2. **Bootstrap the AWS Environment**:

   Before deploying resources with CDK, you must bootstrap your AWS environment:

   ```
   cdk bootstrap
   ```

4. Deploy the Backend (CDK Infrastructure)

The backend infrastructure, including the following services, will be deployed using AWS CDK:

- **Lambda** functions
- **API Gateway**
- **Cognito** for user authentication
- **RDS** for relational data
- **DynamoDB** for non-relational data
- **S3** for file storage
- **CloudWatch** for monitoring
- **SQS** for background job handling

To deploy the backend infrastructure:

```
cdk deploy
```

5. Deploy the Frontend (Next.js Application)

Once the backend is deployed, the frontend can be deployed either via **AWS Amplify** or **Amazon S3**.

1. **Install Frontend Dependencies**:

   In the **frontend** directory, install the necessary dependencies:

   ```
   npm install
   ```

2. **Run the Frontend Locally**:

   To test the frontend locally, run:

   ```
   npm run dev
   ```

3. **Deploy to AWS**:

   To deploy the frontend to AWS, use **AWS Amplify** or **S3** for static hosting:

   - **AWS Amplify**:

Follow the instructions in the AWS Amplify documentation.

- **Amazon S3**:

  For hosting on S3, build the frontend and upload it to an S3 bucket:

  ```
  npm run build
  aws s3 sync ./out s3://<your-s3-bucket-name> --delete
  ```

6. Post-Deployment Configuration

After deployment, ensure that the following are correctly configured:

- **API URLs**: Set the appropriate **API URL** in your frontend application's environment variables (`NEXT_PUBLIC_API_BASE_URL`).
- **AWS Cognito**: Ensure user authentication works by testing the login and sign-up flows.
- **SQS and Lambda**: Verify that notifications and background jobs are functioning correctly.
- **CloudWatch**: Ensure that all metrics and logs are being captured.

User Manual: Instructions for Using the Task Management System

1. Introduction

The Task Management System allows users to create, update, and manage tasks efficiently. It integrates with **AWS** for secure authentication, file storage, and real-time notifications. The system is hosted using **AWS services** such as **Cognito**, **Lambda**, **API Gateway**, **S3**, **DynamoDB**, **CloudWatch**, and **SQS**.

2. Sign Up and Login

1. **Sign Up**:
   - Navigate to the **Sign Up** page.
   - Fill in the required details (email, password, etc.) to create an account.
   - Once signed up, you will be redirected to the **Task Dashboard**.

2. **Login**:
    - If you already have an account, go to the **Login** page.
    - Enter your credentials to log in securely via **Amazon Cognito**.
    - After login, you will be redirected to the **Task Dashboard**.

3. Managing Tasks

1. **Create a Task**:
    - Go to the **Create Task** page.
    - Fill in the task details:
        - **Task Title**
        - **Description**
        - **Priority**
        - **Due Date**
    - Optionally, attach files to the task by clicking the **Upload** button.
    - Click **Create Task** to save the task.
2. **Edit a Task**:
    - From the **Task Dashboard**, select a task you wish to update.
    - Click on the **Edit** button, modify the task details, and save the changes.
3. **Delete a Task**:
    - From the **Task Dashboard**, click on the task you want to delete.
    - Click the **Delete** button to remove the task from the system.
4. **View Task Details**:
    - Click on any task in the dashboard to view its detailed information (title, description, priority, due date, attachments).

4. File Attachments

1. **Uploading Files**:
    - When creating or editing a task, click the **Upload** button to attach files (PDFs, images, etc.).

- Files are stored in **Amazon S3** securely.

2. **Viewing Attachments**:
    - After creating a task with an attachment, you can click on the attachment link to view or download the file.

5. Notifications

1. **Task Notifications**:
    - When a task is updated (status change, priority change, etc.), you will receive a notification.
    - Notifications are sent via **email**, triggered by **AWS Lambda** and processed by **SQS**.

6. Monitoring and Logs

1. **CloudWatch Monitoring**:
    - All system performance, errors, and logs are tracked in **CloudWatch**.
    - As an admin or user, you can view the system's status and metrics in real-time.
2. **Troubleshooting**:
    - If you experience issues, check **CloudWatch Logs** for error messages or reach out to support.

7. Logout

To securely log out from the system:

- Click on the **Logout** button in the user menu.

Conclusion

This **Task Management System** offers an intuitive and secure platform for task management, leveraging **AWS Cloud Services** for performance, security, and scalability. By following the setup guide, you can deploy the system on AWS and start managing tasks seamlessly.