SRE Coding Exercise: Deploy and Monitor a Serverless Application

Objective

Design, deploy, and monitor a serverless web application using AWS Lambda, API Gateway, S3, CloudFront, and RDS (PostgreSQL). Set up observability and logging with Sentry, Splunk, and CloudWatch.

The exercise will be evaluated by Lebogang Mabala and Rodrigo Borges. Please share a link to your GitHub for the completed code. Please send the link to lebogang@mightybyte.us rodrigo@mightybyte.us and dan@mightybyte.us

---

Requirements

1. Serverless Application:

- Create a simple REST API with two endpoints:
  - `POST /tasks`: Accepts a JSON payload and stores task data in an RDS PostgreSQL database.
  - `GET /tasks`: Fetches all tasks from the database.
- Use **AWS Lambda** to handle the backend logic.
- Deploy the API using **AWS API Gateway**.

2. Static Frontend:

- Create a basic static web app (HTML + JavaScript) to interact with the API.
- Host the static site on **AWS S3** and serve it via **AWS CloudFront** for caching and global distribution.

3. Database:

- Set up an **RDS PostgreSQL** instance.
- Use a table with the following schema:

```
CREATE TABLE tasks (
  id SERIAL PRIMARY KEY,
  description TEXT NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4. Observability:

- Configure **Sentry** for error monitoring and reporting.
- Set up **Splunk** to aggregate logs from API Gateway, Lambda, and RDS logs.

- Use **CloudWatch** to monitor the Lambda function and create a dashboard to display key metrics:
  - Request count
  - Error count
  - Average response time

5. Deployment Automation:

- Write an **Infrastructure as Code (IaC)** template (e.g., AWS CloudFormation or Terraform) to provision the AWS resources.
- Include environment variables for configuration (e.g., database credentials, Sentry DSN).

6. Documentation:

- Provide a README with:
  - Setup instructions
  - A sample `curl` command to test the API
  - Links to Sentry, Splunk, and CloudWatch dashboards

Evaluation Criteria

1. **Functionality:**
   - Are the API endpoints functional and correctly interacting with the database?
2. **Architecture:**
   - Is the application leveraging AWS services efficiently (e.g., serverless architecture, CloudFront caching)?
3. **Monitoring and Observability:**
   - Is error monitoring integrated with Sentry?
   - Are logs effectively aggregated in Splunk?
   - Are CloudWatch metrics and dashboards useful for diagnosing issues?
4. **Code Quality:**
   - Is the IaC template well-structured and reusable?
   - Are the Lambda functions modular and maintainable?
5. **Documentation:**
   - Are the setup instructions clear?
   - Is it easy to replicate the deployment?

Bonus Challenges

- Implement CI/CD using GitHub Actions, AWS CodePipeline, or another tool of choice.
- Add a **custom CloudWatch alarm** to alert when error rates exceed a threshold.

- Use **AWS Secrets Manager** for managing database credentials securely.
- Integrate a basic unit test for the Lambda functions.
