Absolutely, Bilal — here is your **complete and detailed phase-by-phase breakdown** of your Oceanic Energy Solutions (OES) cloud implementation, written in a clear and professional format you can directly use in your **portfolio report**.

**🌊 Oceanic Energy Solutions (OES) – Cloud Infrastructure Implementation**

This implementation enables real-time collection, secure storage, processing, and visualization of marine buoy data using AWS services. The system supports data ingestion, monitoring, alerting, authentication, and CI/CD deployment — all built with AWS Free Tier services.

**✅ PHASE 1: DynamoDB – Time-Series Sensor Data Storage**

**🔹 What:**

Created a **DynamoDB** table named BuoyData to persist structured data from autonomous marine buoys.

**🔧 Implementation:**

* Table Name: BuoyData
* Partition Key: buoy\_id (String)
* Sort Key: timestamp (ISO8601 UTC timestamp as String)
* Data Fields: temperature, wave\_height

**🎯 Why:**

* Allows fast, low-latency storage and retrieval of buoy sensor data.
* Suitable for time-series data, enabling easy trend analysis.
* Fully managed and cost-efficient via AWS Free Tier.

**✅ PHASE 2: S3 Bucket – Raw Data Logging + Static Frontend Hosting**

**🔹 What:**

Created an **S3** bucket: oes-buoy-storage-bilal for:

* Hosting your HTML/JS dashboard frontend
* Storing raw buoy data logs in JSON format

**🔧 Implementation:**

* Enabled **static website hosting** on the bucket
* Used put\_object to store logs from Lambda
* Bucket policy secured for CloudFront-only access

**🎯 Why:**

* S3 provides reliable, scalable, and cost-effective storage.
* Simplifies frontend hosting without needing a web server.
* Supports long-term archival and potential analytics via Athena.

**✅ PHASE 3: IAM – Secure Access Control with Least Privilege**

**🔹 What:**

Created an IAM Role (lambda-oes-role) for Lambda functions with least-privilege policies.

**🔧 Policy Permissions:**

* dynamodb:PutItem, dynamodb:Scan
* s3:PutObject
* sns:Publish
* logs:\*

**🎯 Why:**

* Follows AWS best practices (Principle of Least Privilege)
* Ensures Lambda can only access required AWS resources
* Prevents accidental or malicious over-permissioned access

**✅ PHASE 4: Lambda Functions – Backend Processing Logic**

**🔹 Function 1: storeBuoyData**

* Triggered via **POST** from API Gateway
* Stores incoming data into **DynamoDB**
* Saves the full JSON log into **S3**
* Sends **SNS alert** if wave\_height > 3.0

**🔹 Function 2: getData**

* Triggered via **GET** from API Gateway
* Retrieves and returns buoy data from DynamoDB

**🎯 Why:**

* Enables a **serverless backend** that auto-scales
* Ensures clear separation of concerns (store vs. fetch)
* Integrates cleanly with security and monitoring layers

**✅ PHASE 5: API Gateway – Secure API Layer**

**🔹 What:**

Set up a RESTful API using **API Gateway**.

**🔧 Routes:**

* /sendData → POST → triggers storeBuoyData
* /getData → GET → triggers getData

**🎯 Why:**

* Exposes Lambda functions as HTTPS endpoints
* Allows secure interaction with both buoys and the frontend dashboard
* Simplifies scaling and access control using usage plans and authorizers

**✅ PHASE 6: Frontend Dashboard – S3 Hosted Monitoring Interface**

**🔹 What:**

A single-page **HTML + Chart.js** dashboard, deployed to your S3 bucket (served via CloudFront).

**🔧 Features:**

* Fetches data from /getData
* Displays:
  + 📈 Wave Height Over Time
  + 🌡️ Temperature Over Time
  + ⚠️ Critical Alert if wave height > 3.0m
* Includes:
  + Refresh button
  + Animated visual cards
  + Login/logout buttons (Cognito integrated)

**🎯 Why:**

* Provides real-time visibility for stakeholders
* Lightweight, modern UI with mobile responsiveness
* Helps decision-makers and scientists monitor environmental data

**✅ PHASE 7: SNS Alerts – Email Notification on Critical Events**

**🔹 What:**

Integrated **Amazon SNS** for real-time critical alerts.

**🔧 Setup:**

* Topic: BuoyAlerts
* Subscription: bilal.naseer2k18@gmail.com
* Triggered by Lambda if wave height > 3.0

**🎯 Why:**

* Enables proactive monitoring
* Sends real-time emails when hazardous wave conditions are detected
* Can be expanded to multiple subscribers or SMS

**✅ PHASE 8: API Key + Usage Plan – Rate Limiting and Access Control**

**🔹 What:**

Secured API endpoints using **API Keys and Usage Plans**.

**🔧 Configuration:**

* API Key: OESFrontendKey
* Usage Plan: OESUsagePlan
* Associated to both sendData and getData methods

**🎯 Why:**

* Prevents abuse of public API endpoints
* Adds quota management and throttling
* Works alongside JWT Auth or in fallback situations

**✅ PHASE 9: CloudWatch Logs – Monitoring and Debugging**

**🔹 What:**

Enabled **CloudWatch Logs** for both Lambda functions.

**🔧 Capabilities:**

* Logs each invocation
* Captures errors, warnings, and output
* Can be used to build dashboards or set alarms

**🔜 What’s Next:**

* Add **X-Ray tracing** for full request flow
* Create **CloudWatch Dashboard** for latency, invocations, error rate
* Set **alarms** for unusual patterns

**🎯 Why:**

* Helps debug real-time issues
* Provides observability and operational visibility
* Supports continuous improvement

**✅ PHASE 10: CI/CD with GitHub – Automated Frontend Deployment**

**🔹 What:**

Set up a **GitHub-based CI/CD pipeline** to auto-deploy frontend changes to S3.

**🔧 Implementation:**

* Frontend repo hosted on GitHub
* Auto-push new index.html, style.css, script.js to S3 on commit (manual or via GitHub Actions)

**🎯 Why:**

* Streamlines deployment
* Ensures latest version is always online
* Encourages version control and repeatability

**✅ PHASE 11: CloudFront – HTTPS + Cognito Login Redirect Support**

**🔹 What:**

Configured **CloudFront** distribution in front of your S3 bucket.

**🔧 Reason:**

* Cognito requires redirect URIs to be domain-based (not raw S3 URLs)
* Enabled https://d25xenu5by9uag.cloudfront.net as your **login/logout redirect domain**

**🎯 Why:**

* Enables Cognito login/logout flow to complete securely
* Improves CDN performance and SSL handling
* Protects your S3 bucket using origin access policies

**✅ PHASE 12: Cognito JWT Authentication – Login + Logout**

**🔹 What:**

Configured **Amazon Cognito User Pool** for secure frontend login.

**🔧 Features:**

* Login button redirects to Cognito-hosted UI
* id\_token extracted from URL
* Secure logout redirects via CloudFront
* Used sessionStorage for token handling (not localStorage)

**🎯 Why:**

* Adds user authentication without server complexity
* Keeps your data protected from unauthorized access
* JWT-based auth enables secure curl API calls

**✅ PHASE 13: SessionStorage Token Handling – Secure Testing**

**🔹 What:**

Instead of saving tokens in localStorage, used sessionStorage.

**🔧 Benefits:**

* More secure (clears on tab close)
* Enables you to extract token easily for testing with curl

**🎯 Why:**

* Helps secure the frontend
* Prevents long-term storage of sensitive credentials
* Eases development and debugging

**🧾 Summary Table – Full System Overview**

| **AWS Service** | **Role in System** |
| --- | --- |
| **DynamoDB** | Stores buoy sensor readings |
| **S3** | Hosts frontend and logs raw data |
| **Lambda** | Serverless compute to process and retrieve data |
| **API Gateway** | Exposes secure HTTP endpoints |
| **IAM** | Controls access between services |
| **SNS** | Sends alert emails for critical wave events |
| **Cognito** | Provides secure login + token auth |
| **CloudFront** | Enables HTTPS + Cognito redirect support |
| **CloudWatch** | Captures logs and supports monitoring |
| **Chart.js** | Visualizes data in browser dashboard |
| **GitHub CI/CD** | Automates frontend deployment to S3 |

**PHASE 14: ✅ AWS VPC + Budget Implementation Summary (OES Project)**

**🔐 Virtual Private Cloud (VPC) Implementation**

**To secure the cloud infrastructure for Oceanic Energy Solutions (OES), a custom AWS Virtual Private Cloud (VPC) was created using the "VPC and more" workflow. This setup included:**

* **Two private subnets across separate availability zones for high availability**
* **Two public subnets for potential future use**
* **A dedicated Internet Gateway for public routing**
* **Private route tables assigned only to Lambda functions via subnets**

**To maintain a secure, serverless architecture, both storeBuoyData and getBuoyData Lambda functions were explicitly attached to the private subnets in this VPC. These Lambdas interact with DynamoDB and S3, which are accessible via configured VPC Gateway Endpoints — ensuring no public internet routing is used.**

**In addition, an Interface Endpoint for SNS was created so Lambdas could publish wave-height alerts securely within the VPC. The networking configuration was confirmed through successful end-to-end testing, proving full isolation from the internet while maintaining secure access to required AWS services.**

**This architecture aligns with the AWS Well-Architected Framework (Security + Reliability) and supports Zero Trust principles through network segmentation and least-privilege routing.**

**PHASE 15:💸 Billing & Cost Management Integration**

**To control costs and avoid exceeding the AWS Free Tier, a Zero Spend Budget was implemented using AWS Budgets. This budget template was configured to:**

* **Monitor all services across the account**
* **Trigger an email alert when monthly costs exceed $0.01**
* **Notify the project owner at the earliest sign of any billable usage**

**This lightweight but highly effective budget integration ensures that the project remains sustainable and financially predictable — a key requirement in both academic and production-grade deployments.**

**The setup reflects a strong understanding of cost-aware architecture, one of the five pillars of AWS Well-Architected design.**

**🧠 Final Note**

**By combining a private, segmented network with proactive cost monitoring, this solution demonstrates a secure, scalable, and financially responsible cloud implementation suitable for real-world applications in data-sensitive industries like oceanic monitoring.**

Enter Data:

curl -X POST "https://3gwhstp17h.execute-api.eu-north-1.amazonaws.com/dev/sendData" -H "Content-Type: application/json" -H "Authorization: Bearer [TOKEN ID] " \

-d '{

"buoy\_id": "buoy-901",

"temperature": 21.9,

"wave\_height": 3.4

}'

--------------------

curl -X POST "https://3gwhstp17h.execute-api.eu-north-1.amazonaws.com/dev/sendData" -H "Content-Type: application/json" -H "Authorization: Bearer eyJraWQiOiI4cWFSNkV5Uld5VXFcLzdqUUpkS2l3NG9cL3dpZzBcL0pleVVvMlwvWWxqZ29ZTT0iLCJhbGciOiJSUzI1NiJ9..HLWiNutF-ZOWMwR2ecHdAQ0GZcCxS\_0nRNE4Ow7PielblhdWxv3gqpcKp8HcOVSHsMeGINnRiln6UYk2Hy2KehtNFxVfSrj\_qApflgRoa6KxuW7htkBlC2JwcZ7T82aZ1MBhLugT3zOQN8TOaHr127Gze4u9TG9vP6q7mREjxJOjG8tKIeGEJ\_Hx89-WODO2iKUqcxRH\_1XRNZPkM0ICNtiVjJZYXr5PbPlfzlMknzxeQm1nRXETAfIehmLE3KPcLvhhV90cwq6Ptm7sxKReOIt3yc5V9srchhwB53hxX40a-XmSj-5pIjnUJ8CUUyoRthL4reIxlvOHSAl6qf96zg " \ -d '{

"buoy\_id": "buoy-901",

"temperature": 30.4 ,

"wave\_height": 4.1

}'

curl -X POST "https://3gwhstp17h.execute-api.eu-north-1.amazonaws.com/dev/sendData" -H "Content-Type: application/json" -H "Authorization: Bearer eyJraWQiOiI4cWFSNkV5Uld5VXFcLzdqUUpkS2l3NG9cL3dpZzBcL0pleVVvMlwvWWxqZ29ZTT0iLCJhbGciOiJSUzI1NiJ9..HLWiNutF-ZOWMwR2ecHdAQ0GZcCxS\_0nRNE4Ow7PielblhdWxv3gqpcKp8HcOVSHsMeGINnRiln6UYk2Hy2KehtNFxVfSrj\_qApflgRoa6KxuW7htkBlC2JwcZ7T82aZ1MBhLugT3zOQN8TOaHr127Gze4u9TG9vP6q7mREjxJOjG8tKIeGEJ\_Hx89-WODO2iKUqcxRH\_1XRNZPkM0ICNtiVjJZYXr5PbPlfzlMknzxeQm1nRXETAfIehmLE3KPcLvhhV90cwq6Ptm7sxKReOIt3yc5V9srchhwB53hxX40a-XmSj-5pIjnUJ8CUUyoRthL4reIxlvOHSAl6qf96zg " \ -d '{

"buoy\_id": "buoy-901",

"temperature": 30.4 ,

"wave\_height": 4.1

}'