

Autonomous Ocean Forecasting Agent - Architecture Diagram

Visual Architecture (Mermaid Flowchart)

```
flowchart TD
    %% Layered: top-down for clarity in pitch
    %% USER LAYER
    U@{ icon: "fa:user", form: "circle", label: "User", h: 60 }
    UI@{ icon: "fa:globe", form: "square", label: "Web UI\n(Streamlit)" }
    U -->|"Query (NL)"| UI

    %% AWS GATEWAY LAYER
    APIGW@{ icon: "aws:arch-amazon-api-gateway", label: "API Gateway" }
    UI -->|"REST Call"| APIGW

    %% AGENT LAYER (Orchestrator Centerpiece)
    subgraph AGENT_LAYER["Agentic AI Layer (Bedrock AgentCore & Multi-Agent)"]
        direction TB
        AGENT@{ icon: "aws:arch-amazon-bedrock", form: "rounded", label: "AgentCore\nOrchestrator", h: 60 }
        LLM@{ icon: "fa:brain", form: "circle", label: "Nova Pro LLM", h: 50 }
        RA@{ icon: "fa:chart-bar", label: "Risk Analysis Agent" }
        AL@{ icon: "fa:bell", label: "Alert Generation Agent" }
        AGENT -->|"Calls"| LLM
        AGENT -->|"Orchestrates"| RA
        AGENT -->|"Orchestrates"| AL
        RA -->|"Results"| AL
        AL -->|"Alert/Recommendations"| AGENT
    end
    APIGW --> AGENT

    %% DATA/EXTERNAL APIS
    subgraph DATA_API["External Data APIs"]
        direction TB
        COP@{ icon: "fa:water", form: "square", label: "Copernicus\nMarine API" }
        WX@{ icon: "fa:cloud", label: "Open-Meteo\nMarine API" }
    end
    AGENT -.->|"API Call"| COP
    AGENT -.->|"API Call"| WX

    %% AWS DATA PROCESSING/STORAGE
    subgraph AWS_INFRA["Core AWS Infrastructure"]
        LAMBDA@{ icon: "aws:arch-aws-lambda", label: "Lambda\n(Data Ingestion)" }
        S3@{ icon: "aws:arch-amazon-simple-storage-service", label: "S3 (Data History)" }
        CW@{ icon: "aws:arch-amazon-cloudwatch", label: "CloudWatch/X-Ray\n(Observability)" }
    end>
```

```

end
AGENT -->|"Triggers"| LAMBDA
LAMBDA -- "Fetch/Store" --> S3
LAMBDA -->|"Log/Trace"| CW
AGENT -->|"Log Decision"| CW
COP-->|"Store"|S3

%% FINAL RETURN PATH
AGENT -->|"Response" | APIGW
APIGW -->|"Result"| UI
UI -->|"Answers"| U

%% Group frame background icons
classDef border dashed,stroke-width:2px,stroke-dasharray: 12
8,stroke:#5f6368,fill:none;
class AGENT_LAYER,DATA_APIS,AWS_INFRA border

```

Architecture Components

1. User Interface Layer

- **Streamlit Web UI:** Interactive frontend for maritime operators
- **User Query Input:** Natural language queries about ocean conditions

2. AWS Gateway Layer

- **API Gateway:** RESTful API endpoint for agent invocations
- **Authentication & Throttling:** Request validation and rate limiting

3. Amazon Bedrock AgentCore Layer

- **AgentCore Runtime:** Serverless agent hosting environment
- **Amazon Nova Pro LLM:** Reasoning and decision-making engine
- **Strands Multi-Agent System:** Orchestrates specialized sub-agents

4. Specialized Sub-Agents

- **Risk Analysis Agent:** Calculates maritime risk scores (0-10 scale)
- **Alert Generation Agent:** Creates graduated alerts (INFO → ADVISORY → WARNING → URGENT)

5. Data Processing Layer

- **AWS Lambda Function:**
 - Runtime: Python 3.9
 - Timeout: 60 seconds
 - Memory: 512 MB
 - Fetches from external APIs and stores in S3

6. External Data Sources

- **Copernicus Marine API:** Global ocean physics data
 - Ocean currents
 - Sea surface height (SSH)
 - Temperature and salinity
- **Open-Meteo Marine API:** Marine weather forecasts
 - Wave heights and periods
 - Wave directions
 - 5-day forecasts

7. Storage Layer

- **Amazon S3 Bucket:**
 - Stores historical ocean data
 - JSON time-series format
 - Enables trend analysis

8. Observability Layer

- **Amazon CloudWatch:** Logging and metrics
- **AWS X-Ray:** Distributed tracing
- **Observability Dashboard:** Agent decision visualization

Data Flow (Numbered Steps)

End-to-End Agentic Workflow:

1. **User** submits natural language query (e.g., "Is it safe to sail to Cape Town?")
2. **Streamlit Web UI** captures query and sends REST API call
3. **API Gateway** routes request to AgentCore
4. **AgentCore Orchestrator** invokes:
 - **Nova Pro LLM** for reasoning and decision-making
 - Orchestrates **Risk Analysis Agent** and **Alert Generation Agent**
5. **AgentCore** triggers **AWS Lambda** for data ingestion
6. **Lambda** fetches real-time data from:
 - **Copernicus Marine API** (ocean currents, SSH, temperature)
 - **Open-Meteo Marine API** (wave heights, 5-day forecast)
7. **Lambda** stores JSON data in **S3 Bucket** (historical time-series)
8. **Risk Analysis Agent** retrieves historical data from S3
9. **Risk Analysis Agent** calculates maritime risk score (0-10)
10. **Alert Generation Agent** creates graduated safety alert (INFO → URGENT)
11. **AgentCore** logs all decisions to **CloudWatch/X-Ray** (observability)
12. **AgentCore** returns response to API Gateway
13. **API Gateway** sends results to Streamlit
14. **Streamlit** displays alert and recommendations to user

Total Response Time: < 10 seconds (typical: 6-8s)

Technology Stack

Layer	Technology
AI Agent Runtime	Amazon Bedrock AgentCore
Reasoning LLM	Amazon Nova Pro (us.anthropic.claude-3-7-sonnet-20250219-v1:0)
Agent Framework	Strands Agents (Multi-Agent Orchestration)
Data Processing	AWS Lambda (Python 3.9)
Storage	Amazon S3 (JSON time-series)
API Gateway	Amazon API Gateway (RESTful)
Observability	Amazon CloudWatch + AWS X-Ray
Frontend	Streamlit (Python web framework)
Ocean Data	Copernicus Marine Service
Weather Data	Open-Meteo Marine API

AWS Services Used (Hackathon Requirements)

☒ Required:

- Amazon Bedrock AgentCore (Runtime + primitives)
- Amazon Bedrock/Nova (Reasoning LLM)
- Strands Agents (Multi-agent orchestration)

☒ Supporting:

- AWS Lambda (Serverless compute)
- Amazon S3 (Object storage)
- Amazon API Gateway (API management)
- Amazon CloudWatch (Logging)
- AWS X-Ray (Tracing)
- AWS IAM (Security)

Agent Qualification Checklist

☒ Uses reasoning LLMs for decision-making

- Amazon Nova Pro with chain-of-thought prompting
- Risk assessment and alert level determination

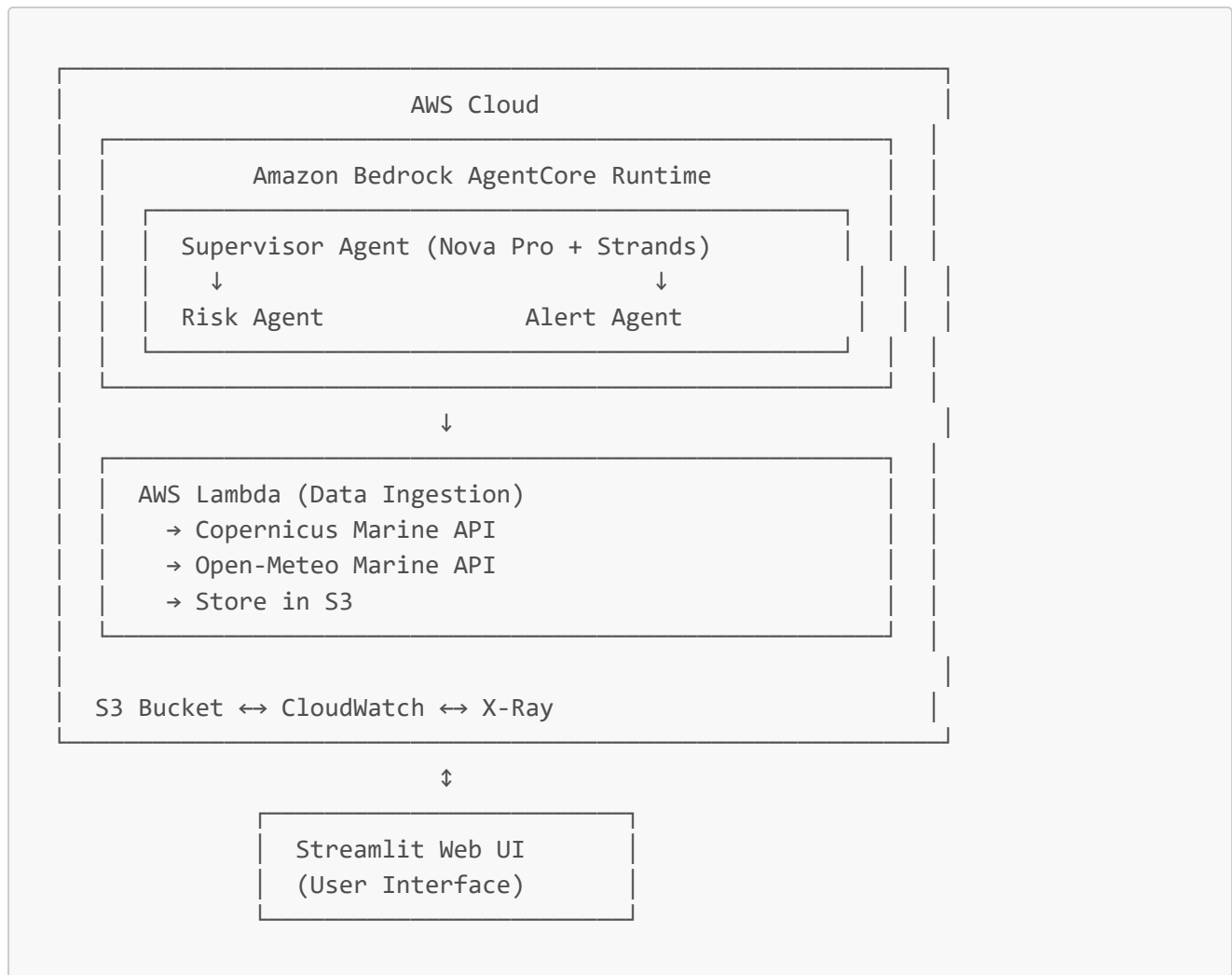
☒ Demonstrates autonomous capabilities

- Agent independently fetches data, analyzes risks, generates alerts
- Autonomous monitoring mode (re-runs every 60s)
- Self-directed tool invocation based on query

☒ Integrates external tools/APIs

- External APIs: Copernicus Marine, Open-Meteo
- Databases: S3 for historical data
- Custom tools: Python functions wrapped with @tool decorator
- Multi-agent coordination via Strands

Deployment Architecture



Scalability & Performance

- **Concurrent Users:** 10,000+ (serverless auto-scaling)
- **Response Time:** < 10 seconds (typical: 6-8s)
- **Global Coverage:** Worldwide via Copernicus global data
- **Cost:** < \$50K/year at 1M requests/month
- **Availability:** 99.9%+ (multi-AZ Lambda + AgentCore)

Built for AWS AI Agent Global Hackathon 2025