**1. Components Where CloudWatch Can Be Used to Check Application Logs**

AWS CloudWatch can be configured to collect logs from the following components in the architecture:

* **AWS ECR/ECS (Containerized Services)**
  + Log application logs from services running in ECS using **CloudWatch Logs**.
  + Use **AWS FireLens** for enhanced log routing.
* **AWS RDS (MS SQL)**
  + Enable **CloudWatch Logs for RDS** to monitor database logs, slow queries, and error logs.
* **AWS MSK (Managed Kafka)**
  + Configure **CloudWatch Metrics and Logs** for broker health and topic activity.
* **API Gateway**
  + Enable **CloudWatch Logs for API Gateway** to capture request/response logs and errors.
* **AWS Lambda (if used internally)**
  + Lambda logs execution details automatically to **CloudWatch Logs**.
* **AWS ALB (Application Load Balancer)**
  + ALB access logs can be stored in **S3** and monitored via CloudWatch.
* **Reg Procesor, Rep Procesor, Eli Service, Inputor, CSV Data Loader**
  + Since these services are deployed on ECS, enable **CloudWatch Logs agent** for centralized log storage.

**2. CloudWatch Configurations for Observability**

To monitor the health and performance of services in CloudWatch, configure the following:

**A. ECS Services (Reg Procesor, Rep Procesor, Eli Service, Inputor, etc.)**

* **Enable ECS CloudWatch Container Insights** for metrics like:
  + CPU and Memory Utilization
  + Container Restarts and Failures
  + Network Traffic

**B. AWS RDS (MS SQL)**

* Enable **Enhanced Monitoring** and collect:
  + Database CPU, memory, disk I/O metrics
  + Query execution performance
  + Connections and deadlocks

**C. API Gateway**

* Enable **CloudWatch Metrics and Logs** for:
  + Request latency
  + 4XX and 5XX error rates
  + Integration latency

**D. AWS MSK (Kafka)**

* Enable **MSK CloudWatch Metrics** to track:
  + Broker CPU and Memory Utilization
  + Topic-specific message rates
  + Consumer lag

**E. AWS S3 (For Data Storage & Processing)**

* Use **S3 Event Notifications** to track:
  + File uploads/downloads
  + Object access patterns

**F. AWS ALB (Load Balancer)**

* Enable **ALB CloudWatch Metrics** to monitor:
  + Request count per target
  + Latency (average response time)
  + 4XX and 5XX errors

**G. CloudWatch Alarms**

* Set alarms for:
  + ECS task failures
  + High API Gateway response times
  + RDS connection spikes
  + Kafka consumer lag threshold breach

**How to configure AWS CloudWatch Logs and Metrics for all the components mentioned in Steps 1 & 2.**

**Step 1: Configuring AWS CloudWatch Logs for Application Logs**

AWS CloudWatch Logs allow you to collect, monitor, and analyze logs generated by various AWS services like ECS, RDS, MSK, API Gateway, and more.

**1.1 Enabling CloudWatch Logs for ECS Services (Reg Procesor, Rep Procesor, Eli Service, Inputor, etc.)**

ECS (Elastic Container Service) can be configured to send logs to CloudWatch using **awslogs driver**.

**Steps to Enable CloudWatch Logs for ECS Tasks:**

1. **Open AWS Console** → Navigate to **ECS**.
2. Click on **Task Definitions** → Select the ECS Task for your application.
3. Click **Create New Revision**.
4. Scroll to **Container Definitions** and click **Edit**.
5. Under **Log Configuration**, choose:
   * **Log Driver**: awslogs
   * **Log group**: /ecs/your-service-name
   * **Region**: Select your AWS Region
   * **Stream prefix**: ecs
6. Click **Update** → Click **Create**.
7. Deploy the new revision by updating your ECS Service:
   * Go to **ECS Cluster** → **Service** → **Update** → Choose new revision.
   * Click **Deploy**.

✅ **Logs from ECS tasks will now be available in CloudWatch Logs under /ecs/your-service-name.**

**1.2 Enabling CloudWatch Logs for RDS (MS SQL)**

AWS RDS provides options to log slow queries, errors, and general logs.

**Steps to Enable RDS CloudWatch Logs:**

1. **Open AWS Console** → Navigate to **RDS**.
2. Click on **Databases** → Select your **RDS Instance**.
3. Scroll to the **Log Exports** section.
4. Select logs to export:
   * ✅ **Audit logs**
   * ✅ **Error logs**
   * ✅ **Slow query logs**
5. Click **Modify** → **Apply Changes**.
6. Open **CloudWatch Logs** → Check for /aws/rds/instance-name/logs.

✅ **Your RDS logs will now be available in CloudWatch Logs.**

**1.3 Enabling CloudWatch Logs for AWS MSK (Kafka)**

MSK (Managed Kafka) logs are not enabled by default.

**Steps to Enable CloudWatch Logs for MSK:**

1. **Open AWS Console** → Navigate to **MSK**.
2. Click **Clusters** → Select your Kafka Cluster.
3. Under **Monitoring**, enable:
   * **CloudWatch Logs**: Broker logs
   * **CloudWatch Metrics**: Kafka topic metrics
4. Click **Save**.

✅ **Kafka logs will now be sent to CloudWatch Logs under /aws/msk/broker-logs.**

**1.4 Enabling CloudWatch Logs for API Gateway**

API Gateway provides logs for requests, responses, and errors.

**Steps to Enable CloudWatch Logs for API Gateway:**

1. **Open AWS Console** → Navigate to **API Gateway**.
2. Select your API → Click **Stages**.
3. Select a **Stage** (e.g., Prod).
4. Under **Logs/Tracing**, enable:
   * ✅ **Enable CloudWatch Logs**
   * ✅ **Log full requests and responses**
5. Click **Deploy API**.

✅ **API Gateway logs will be stored in CloudWatch Logs under /aws/apigateway/your-api-name.**

**1.5 Enabling CloudWatch Logs for ALB (Application Load Balancer)**

ALB logs are stored in **S3** but can be monitored via CloudWatch.

**Steps to Enable ALB Logs:**

1. **Open AWS Console** → Navigate to **EC2** → Click **Load Balancers**.
2. Select your ALB → Click **Attributes**.
3. Enable **Access Logs**.
4. Set **S3 Bucket Name** → Save.
5. Configure an **AWS Lambda function** to stream logs from S3 to CloudWatch.

✅ **ALB logs will now be available for monitoring in CloudWatch.**

**Step 2: Configuring AWS CloudWatch Metrics for Observability**

CloudWatch Metrics help track application performance and health.

**2.1 Enabling CloudWatch Metrics for ECS Services**

CloudWatch **Container Insights** provides detailed ECS metrics.

**Steps to Enable Container Insights for ECS:**

1. **Open AWS Console** → Navigate to **ECS**.
2. Click **Clusters** → Select your ECS Cluster.
3. Click **Update Cluster Settings**.
4. Enable **Container Insights** → Save.
5. Open **CloudWatch Console** → Navigate to **Metrics**.
6. Under ECS/ContainerInsights, check:
   * CPUUtilization
   * MemoryUtilization
   * Task Count

✅ **CloudWatch now provides ECS CPU, memory, and network usage.**

**2.2 Enabling CloudWatch Metrics for RDS**

RDS **Enhanced Monitoring** provides real-time database metrics.

**Steps to Enable Enhanced Monitoring for RDS:**

1. **Open AWS Console** → Navigate to **RDS**.
2. Click **Databases** → Select your **RDS Instance**.
3. Scroll to **Monitoring** → Click **Modify**.
4. Enable **Enhanced Monitoring**.
5. Set **Granularity** to 1s or 5s (for real-time monitoring).
6. Click **Save**.

✅ **CloudWatch now provides CPU, memory, and connection metrics for RDS.**

**2.3 Enabling CloudWatch Metrics for AWS MSK (Kafka)**

Kafka requires **CloudWatch Metrics** for monitoring brokers and topics.

**Steps to Enable CloudWatch Metrics for Kafka:**

1. **Open AWS Console** → Navigate to **MSK**.
2. Click **Clusters** → Select your Kafka Cluster.
3. Under **Monitoring**, enable:
   * ✅ **Broker CPU and Memory Utilization**
   * ✅ **Topic Message Rates**
   * ✅ **Consumer Lag Metrics**
4. Click **Save**.

✅ **Kafka metrics will now be available in CloudWatch Metrics under /aws/msk/metrics.**

**2.4 Enabling CloudWatch Alarms for Health Monitoring**

Set up **CloudWatch Alarms** for real-time notifications.

**Steps to Create CloudWatch Alarms:**

1. **Open AWS Console** → Navigate to **CloudWatch**.
2. Click **Alarms** → **Create Alarm**.
3. Select a metric:
   * ✅ **ECS Task Failures** (ECS/ContainerInsights)
   * ✅ **High API Response Times** (API Gateway Metrics)
   * ✅ **RDS Connection Spikes** (RDS Metrics)
   * ✅ **Kafka Consumer Lag** (MSK Metrics)
4. Set **Threshold** (e.g., **CPU Utilization > 80%**).
5. Select an **SNS Notification** to send alerts.
6. Click **Create Alarm**.

✅ **CloudWatch will now send alerts for critical service issues.**

**2.5 Enabling CloudWatch Dashboards for Visualization**

To track multiple metrics, create a **CloudWatch Dashboard**.

**Steps to Create a CloudWatch Dashboard:**

1. **Open AWS Console** → Navigate to **CloudWatch**.
2. Click **Dashboards** → **Create Dashboard**.
3. Select a **Widget Type**:
   * ✅ **ECS CPU & Memory Usage**
   * ✅ **API Gateway Request Latency**
   * ✅ **RDS Query Performance**
   * ✅ **Kafka Broker Utilization**
4. Click **Save Dashboard**.

✅ **You now have a real-time CloudWatch dashboard for AWS observability.**

**Conclusion**

By following these steps, you have: ✅ **Configured CloudWatch Logs for ECS, RDS, Kafka, API Gateway, and ALB.**  
✅ **Enabled CloudWatch Metrics for ECS, RDS, Kafka, and API Gateway.**  
✅ **Created CloudWatch Alarms to monitor service health.**  
✅ **Set up a CloudWatch Dashboard for observability.**

**1.1 Capturing Java Spring Boot Application Logs in CloudWatch and Viewing Them**

In **AWS ECS**, Java Spring Boot applications typically generate logs using **Logback** or **Log4j**. These logs need to be captured and sent to **AWS CloudWatch Logs** for centralized monitoring and debugging. Below is a detailed step-by-step guide on how to **configure, capture, and view** these logs.

**Step 1: Configure Java Spring Boot Application for Logging**

To send logs to CloudWatch, the Java Spring Boot application must be configured correctly.

**Option 1: Logback Configuration (Recommended)**

If your Spring Boot application uses **Logback**, add a CloudWatch appender in your logback.xml file:

xml

CopyEdit

<configuration>

<appender name="CLOUDWATCH" class="com.kdgregory.logback.aws.CloudWatchAppender">

<region>us-east-1</region> <!-- Change to your AWS region -->

<logGroup>/ecs/your-service-name</logGroup>

<logStream>{hostname}</logStream>

<autoCreate>true</autoCreate>

<rotationMode>interval</rotationMode>

<rotationInterval>60</rotationInterval>

</appender>

<root level="INFO">

<appender-ref ref="CLOUDWATCH" />

</root>

</configuration>

**Alternative:** If using **Log4j**, a similar CloudWatch Appender can be configured in log4j2.xml.

**Step 2: Enable CloudWatch Logs for ECS Services**

Once the application is logging properly, configure **AWS ECS** to capture and forward logs.

**2.1: Update ECS Task Definition**

1. **Go to AWS ECS Console** → Click **Task Definitions** → Select your **Task**.
2. Click **Create New Revision** → Scroll to **Container Definitions**.
3. Click **Edit** on the **Spring Boot container**.
4. Scroll to **Log Configuration** → Set:
   * **Log Driver**: awslogs
   * **awslogs-group**: /ecs/your-service-name
   * **awslogs-region**: <your-region>
   * **awslogs-stream-prefix**: ecs
5. Click **Update**, then **Create** a new revision.

**Step 3: Deploy the Updated ECS Service**

1. Go to **ECS Console** → Click on **Clusters**.
2. Select your **Cluster** → Click on **Services**.
3. Select your **Spring Boot ECS Service** → Click **Update**.
4. Choose the **latest Task Definition Revision**.
5. Click **Deploy**.

**Step 4: View Logs in CloudWatch**

Once the service is running, you can view logs in **CloudWatch**.

**Option 1: Using AWS Console**

1. Go to **AWS CloudWatch Console**.
2. Click **Logs** → Choose **Log Groups**.
3. Find /ecs/your-service-name and click on it.
4. Select the log stream (e.g., ecs/task-id) to view logs.

**Option 2: Using AWS CLI**

Run the following AWS CLI command to retrieve logs:

aws logs tail --follow /ecs/your-service-name --region <your-region>

**Step 5: Set Up Log Retention and Alerts**

**5.1: Configure Log Retention**

By default, logs are stored indefinitely. Set retention:

1. Go to **CloudWatch Logs** → **Log Groups**.
2. Select /ecs/your-service-name.
3. Click **Edit Retention** → Choose a period (e.g., 30 days).

**5.2: Set Up CloudWatch Alerts for Errors**

1. Open **CloudWatch Console** → Go to **Logs Insights**.
2. Run a query to filter error logs:

fields @timestamp, @message

| filter @message like "ERROR"

1. Click **Create Alarm** → Set **threshold conditions**.
2. Choose an **SNS topic** to receive email notifications.

**Summary**

✅ Java Spring Boot logs are configured using **Logback/Log4j2**.  
✅ Logs are sent to **AWS CloudWatch** via the ECS Task Definition.  
✅ Logs can be viewed in **CloudWatch Console** or via **AWS CLI**.  
✅ Alerts can be set up for **error monitoring**.

**Below is a detailed guide for enabling AWS CloudWatch Logs and Metrics for the following AWS services:**

1. **Amazon RDS (Relational Database Service)**
2. **Amazon MSK (Managed Streaming for Apache Kafka)**
3. **Amazon API Gateway**
4. **AWS Lambda**
5. **Application Load Balancer (ALB)**

Each section includes steps on **how logs are captured, how to configure CloudWatch integration, and how to view logs and metrics**.

**1. Amazon RDS (Relational Database Service)**

**1.1 How RDS Logs Are Captured**

Amazon RDS supports logging at different levels:

* **Error logs**: System errors, startup/shutdown events.
* **Slow query logs**: Queries that exceed a set threshold.
* **General logs**: All SQL statements executed.

**1.2 Enable CloudWatch Logs for RDS**

1. **Go to AWS Console** → **Amazon RDS**.
2. Select the **RDS instance**.
3. Click **Modify** → Scroll to **Database logs exports**.
4. Select:
   * ✅ error log
   * ✅ slow query log
   * ✅ general log
5. Click **Continue**, then **Apply Changes**.

**1.3 View RDS Logs in CloudWatch**

1. Open **CloudWatch Console** → Go to **Log Groups**.
2. Find /aws/rds/instance-name/error and click it.
3. Click **Log Streams** to view logs.

**2. Amazon MSK (Managed Streaming for Apache Kafka)**

**2.1 How MSK Logs Are Captured**

MSK generates logs for:

* **Broker logs**: Kafka broker activity.
* **Topic logs**: Messages sent/received.
* **Zookeeper logs**: Coordination logs.

**2.2 Enable CloudWatch Logs for MSK**

1. **Go to AWS Console** → **MSK Clusters**.
2. Click **Create Cluster** or select an existing cluster.
3. Scroll to **Monitoring**.
4. Enable:
   * ✅ **CloudWatch Logs** (kafka\_broker, zookeeper)
   * ✅ **Enhanced Monitoring** for broker metrics.
5. Click **Save Changes**.

**2.3 View MSK Logs in CloudWatch**

1. Open **CloudWatch Console** → **Log Groups**.
2. Look for /aws/msk/cluster-name and select it.
3. Click on **Log Streams** to view logs.

**3. API Gateway**

**3.1 How API Gateway Logs Are Captured**

API Gateway captures:

* **Execution logs**: Request/response flow.
* **Access logs**: Client details, status codes.

**3.2 Enable CloudWatch Logs for API Gateway**

1. Open **AWS API Gateway Console**.
2. Select your **API** → Click **Stages**.
3. Select the **Stage** (e.g., dev, prod).
4. Enable logging:
   * ✅ **Enable CloudWatch Logs**
   * ✅ **Enable Execution Logging** → Set to INFO
   * ✅ **Enable Access Logging** → Set format:

{

"requestId": "$context.requestId",

"ip": "$context.identity.sourceIp",

"method": "$context.httpMethod",

"path": "$context.path",

"status": "$context.status"

}

1. Click **Deploy API**.

**3.3 View API Gateway Logs in CloudWatch**

1. Open **CloudWatch Console** → **Log Groups**.
2. Look for /aws/api-gateway/api-name and click it.
3. View execution logs for request debugging.

**4. AWS Lambda**

**4.1 How Lambda Logs Are Captured**

AWS Lambda automatically logs:

* **Function invocation logs**.
* **Execution errors, stack traces**.

**4.2 Enable CloudWatch Logs for Lambda**

1. Open **AWS Lambda Console**.
2. Click your **Lambda function**.
3. Scroll to **Monitoring** → Click **View Logs in CloudWatch**.
4. Add permissions:
   * Go to **IAM** → Attach the AWSLambdaBasicExecutionRole policy.

**4.3 View Lambda Logs in CloudWatch**

1. Open **CloudWatch Console** → **Log Groups**.
2. Look for /aws/lambda/function-name and click it.
3. View logs for function executions.

**5. Application Load Balancer (ALB)**

**5.1 How ALB Logs Are Captured**

ALB captures:

* **Access logs**: Request IP, response time.
* **Error logs**: Failed connections, timeout errors.

**5.2 Enable CloudWatch Logs for ALB**

1. Open **AWS Console** → **EC2** → **Load Balancers**.
2. Select your **ALB** → Click **Attributes**.
3. Enable **Access Logs**:
   * ✅ **Enable Logging**
   * ✅ Set **S3 Bucket** to store logs.
   * ✅ Set **Prefix** (e.g., alb-logs/).
4. Click **Save Changes**.

**5.3 View ALB Logs in CloudWatch**

ALB logs are stored in **S3**. To send them to **CloudWatch**:

1. Create a **Lambda function** to transfer logs from **S3** to **CloudWatch**.
2. Use this Python script in Lambda:

import boto3

import gzip

import json

from io import BytesIO

def lambda\_handler(event, context):

s3 = boto3.client('s3')

logs\_client = boto3.client('logs')

for record in event['Records']:

bucket = record['s3']['bucket']['name']

key = record['s3']['object']['key']

obj = s3.get\_object(Bucket=bucket, Key=key)

log\_data = gzip.GzipFile(fileobj=BytesIO(obj['Body'].read())).read().decode('utf-8')

logs\_client.put\_log\_events(

logGroupName='/aws/elb/access-logs',

logStreamName='alb-log-stream',

logEvents=[{'timestamp': int(event['eventTime']), 'message': log\_data}]

)

1. Configure an **S3 Event Notification** to trigger this Lambda function.

**Updated Summary Table: CloudWatch Integration for AWS Services**

| **AWS Service** | **Logs Captured** | **CloudWatch Integration Steps** |
| --- | --- | --- |
| **ECS - Reg Procesor** | Application logs, task logs | Enable CloudWatch Logs in ECS Task Definition, use Fluent Bit or AWS Logs Driver |
| **ECS - Rep Procesor** | Application logs, task logs | Enable CloudWatch Logs in ECS Task Definition, use Fluent Bit or AWS Logs Driver |
| **ECS - Eli Service** | Application logs, task logs | Enable CloudWatch Logs in ECS Task Definition, use Fluent Bit or AWS Logs Driver |
| **ECS - Inputor** | Application logs, task logs | Enable CloudWatch Logs in ECS Task Definition, use Fluent Bit or AWS Logs Driver |
| **RDS** | Error, slow query, general logs | Enable in RDS settings, view in CloudWatch |
| **MSK** | Broker, topic, Zookeeper logs | Enable in MSK settings, view in CloudWatch |
| **API Gateway** | Execution, access logs | Enable in API Stage settings, use custom format |
| **Lambda** | Invocation logs, errors | Automatically enabled, view in CloudWatch |
| **ALB** | Access, error logs | Store in S3, transfer logs via Lambda |

**Updated Summary Table for Observability:**

| **AWS Service** | **Metrics Captured** | **CloudWatch Configuration Steps** |
| --- | --- | --- |
| **ECS - Reg Procesor** | CPU, Memory, Task Count, Network IO, Application Metrics | Enable CloudWatch Container Insights, use ECS service metrics |
| **ECS - Rep Procesor** | CPU, Memory, Task Count, Network IO, Application Metrics | Enable CloudWatch Container Insights, use ECS service metrics |
| **ECS - Eli Service** | CPU, Memory, Task Count, Network IO, Application Metrics | Enable CloudWatch Container Insights, use ECS service metrics |
| **ECS - Inputor** | CPU, Memory, Task Count, Network IO, Application Metrics | Enable CloudWatch Container Insights, use ECS service metrics |
| **RDS (MS SQL)** | Connections, Queries per second, CPU, Freeable Memory, Latency | Enable Enhanced Monitoring, configure alarms |
| **MSK (Kafka)** | Broker Health, Active Connections, Messages in/out, Consumer Lag | Enable MSK Monitoring, configure CloudWatch alarms |
| **API Gateway** | Request Count, Latency, Error Rate, Integration Latency | Enable detailed metrics in API Gateway settings |
| **Lambda** | Invocation Count, Duration, Error Rate, Throttles | Enable Lambda Insights for detailed observability |
| **ALB (Application Load Balancer)** | Request Count, Response Time, HTTP Errors, Target Response Time | Enable ALB access logs and metrics in CloudWatch |

This table outlines the observability metrics and required configurations for each AWS service to integrate with **Amazon CloudWatch** effectively.