CloudWatch Calculator Documentation

1. Project Structure

```
nodejstrial/
├── app.js            # Main calculator application
├── logger.js         # CloudWatch logging configuration
├── test-cloudwatch.js  # Test script for logging
├── package.json      # Project dependencies
└── node_modules/     # Installed packages
```

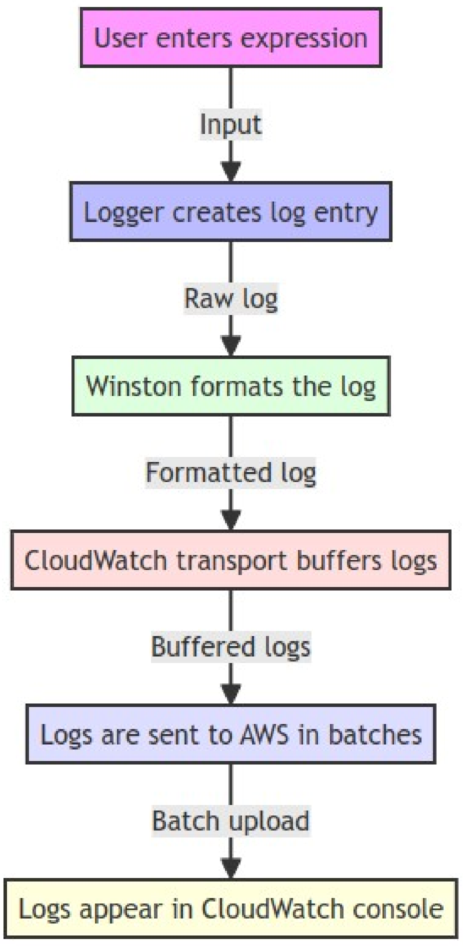
2. How It Works

2.1 Logger Setup (logger.js)

- Uses `winston` as the logging framework
- `winston-cloudwatch` sends logs to AWS CloudWatch
- AWS credentials are used to authenticate with AWS services
- Creates a log group and stream in CloudWatch
- Formats logs with timestamp and metadata

2.2 Calculator Application (app.js)

- Takes mathematical expressions as input
- Logs various events:
    - Application start/stop
    - User input
    - Calculation attempts
    - Success/failure results
- Includes error handling with logging



3. Flow of Data

1. User enters expression
2. Logger creates log entry
3. Winston formats the log
4. CloudWatch transport buffers logs
5. Logs are sent to AWS in batches
6. Logs appear in CloudWatch console

4. AWS Services Used

- **CloudWatch Logs**: Stores application logs
- **IAM**: Manages permissions for logging
- **AWS SDK**: Communicates with AWS services

5. Cost Management ⚠

5.1 How to Stop Incurring Charges

```
# 1. Delete Log Groups in AWS Console
- Go to CloudWatch console
- Select "Log groups" from left sidebar
- Find "calculator-app-logs"
- Select it and click "Delete"

# 2. Or use AWS CLI
aws logs delete-log-group --log-group-name calculator-app-logs
```

5.2 View Current Usage

```
# Check log group size
aws logs describe-log-groups \
  --log-group-name-prefix calculator-app-logs
```

5.3 Cost Reduction Tips

Delete logs when done testing

1. Set shorter retention periods
2. Use log filters to store only important logs
3. Consider local logging for development

6. Local Testing Without AWS

Replace logger.js with local-only logging:

```
const winston = require('winston');

const logger = winston.createLogger({
  transports: [
    new winston.transports.Console(),
    new winston.transports.File({ filename: 'calculator.log' })
  ]
});

module.exports = logger;
```

7. AWS Pricing Information (as of 2024)

- Data ingestion: $0.50 per GB
- Data storage: $0.03 per GB per month
- Free Tier:
  - 5GB of ingestion
  - 5GB of storage
  - 3 dashboards per month

8. Safety Checklist

- ✅ Never commit AWS credentials to code
- ✅ Delete unused log groups
- ✅ Monitor AWS billing dashboard
- ✅ Set up billing alerts
- ✅ Use local logging for development

9. Best Practices

1. Use environment variables for credentials
2. Implement log rotation
3. Log appropriate level of detail
4. Handle errors gracefully
5. Include relevant metadata in logs

10. Troubleshooting

1. Check AWS credentials are set
2. Verify IAM permissions
3. Confirm correct region
4. Look for error messages in console
5. Check network connectivity

11. How to Turn Everything Off

Step 1: Delete AWS Resources

```
# Delete log group
aws logs delete-log-group --log-group-name calculator-app-logs

# Verify deletion
aws logs describe-log-groups --log-group-name-prefix calculator-app-logs
```

Step 2: Switch to Local Logging

1. **Remove CloudWatch dependencies**

   ```
   npm uninstall winston-cloudwatch aws-sdk
   ```

   Update logger.js to local-only version (as shown in section 6)

Step 3: Clean Up Credentials

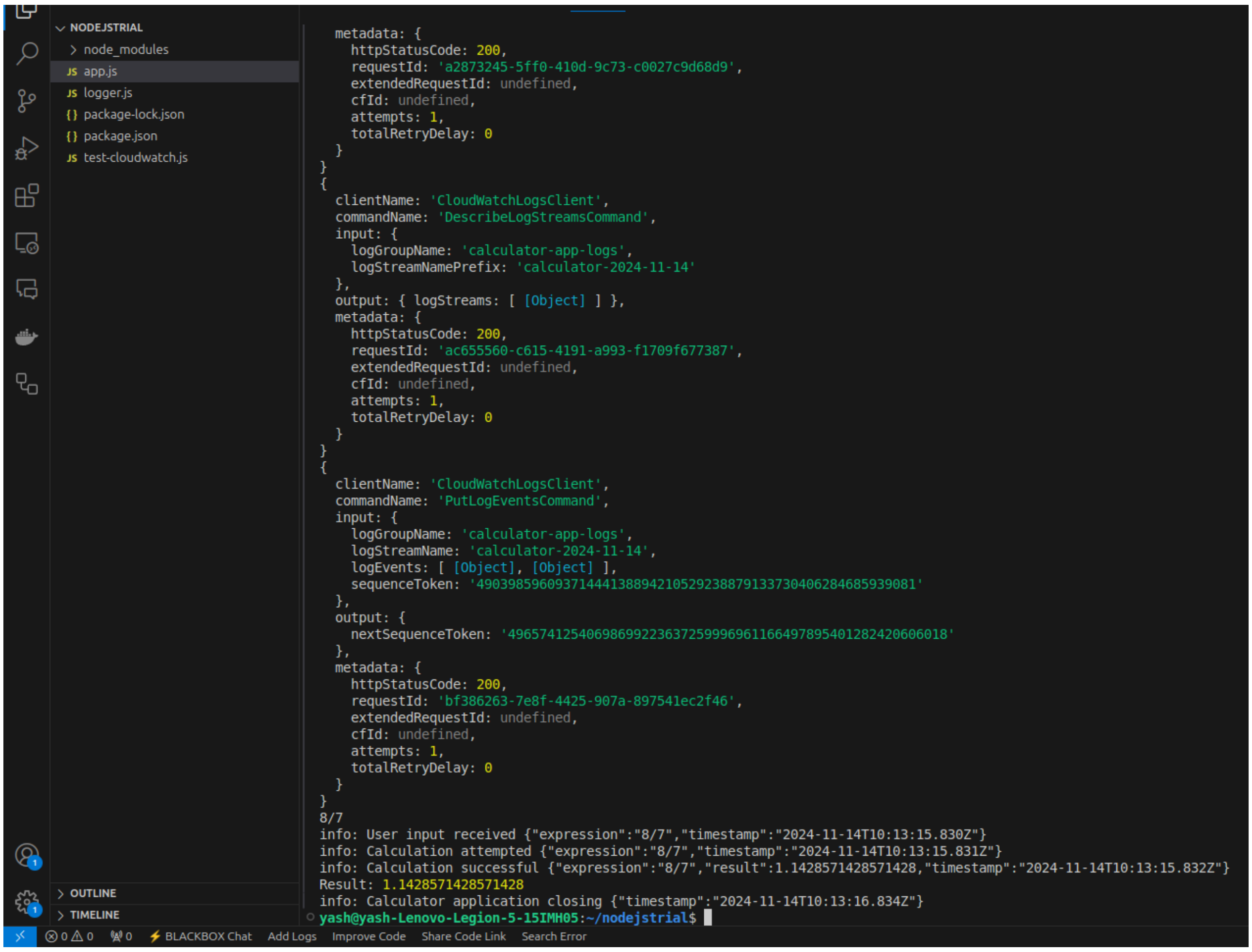
```
# Clear environment variables
unset AWS_ACCESS_KEY_ID
unset AWS_SECRET_ACCESS_KEY
unset AWS_REGION
```

12. Monitor AWS Costs

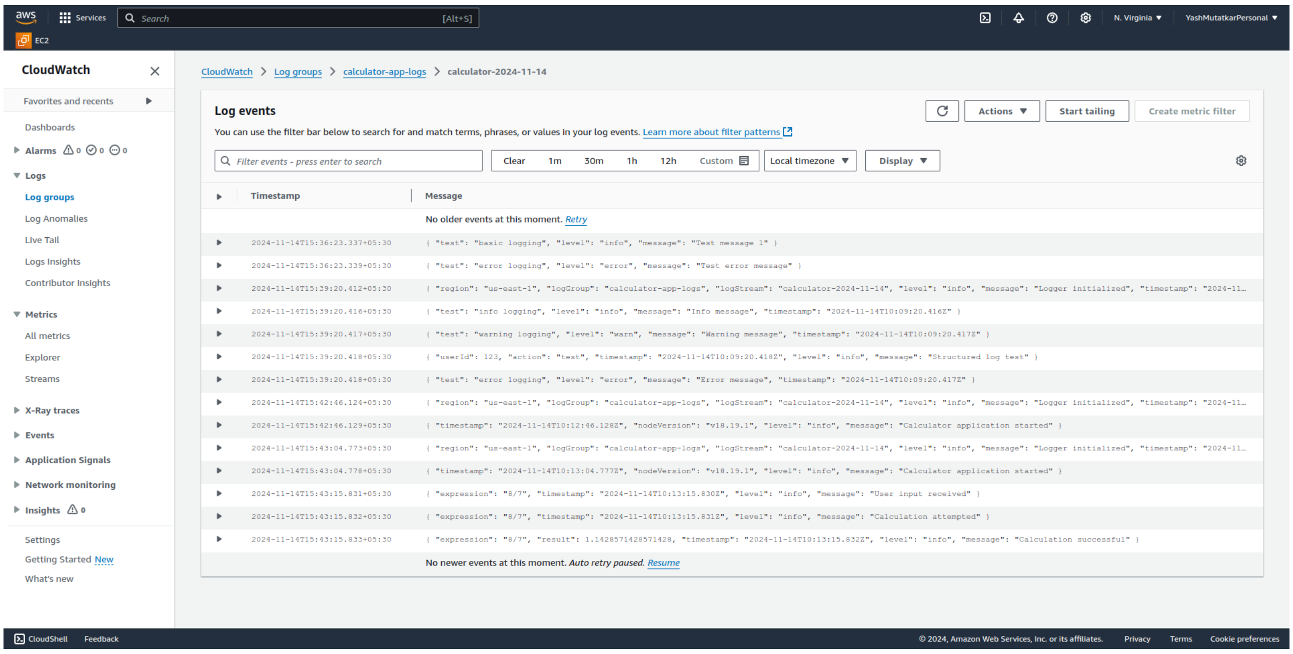
1. Visit AWS Billing Dashboard
2. Check CloudWatch usage
3. Set up billing alerts:
    - Go to Billing Dashboard
    - Create budget
    - Set alert threshold (e.g., $5)

Remember: Always delete test resources and monitor your AWS billing dashboard to avoid unexpected charges!

Outputs



```
    metadata: {
      httpStatusCode: 200,
      requestId: 'a2873245-5ff0-410d-9c73-c0027c9d68d9',
      extendedRequestId: undefined,
      cfId: undefined,
      attempts: 1,
      totalRetryDelay: 0
    }
  }
  {
    clientName: 'CloudWatchLogsClient',
    commandName: 'DescribeLogStreamsCommand',
    input: {
      logGroupName: 'calculator-app-logs',
      logStreamNamePrefix: 'calculator-2024-11-14'
    },
    output: { logStreams: [ [Object] ] },
    metadata: {
      httpStatusCode: 200,
      requestId: 'ac655560-c615-4191-a993-f1709f677387',
      extendedRequestId: undefined,
      cfId: undefined,
      attempts: 1,
      totalRetryDelay: 0
    }
  }
  {
    clientName: 'CloudWatchLogsClient',
    commandName: 'PutLogEventsCommand',
    input: {
      logGroupName: 'calculator-app-logs',
      logStreamName: 'calculator-2024-11-14',
      logEvents: [ [Object], [Object] ],
      sequenceToken: '49039859609371444138894210529238879133730406284685939081'
    },
    output: {
      nextSequenceToken: '49657412540698699223637259996961166497895401282420606018'
    },
    metadata: {
      httpStatusCode: 200,
      requestId: 'bf386263-7e8f-4425-907a-897541ec2f46',
      extendedRequestId: undefined,
      cfId: undefined,
      attempts: 1,
      totalRetryDelay: 0
    }
  }
  8/7
  info: User input received {"expression":"8/7","timestamp":"2024-11-14T10:13:15.830Z"}
  info: Calculation attempted {"expression":"8/7","timestamp":"2024-11-14T10:13:15.831Z"}
  info: Calculation successful {"expression":"8/7","result":1.1428571428571428,"timestamp":"2024-11-14T10:13:15.832Z"}
  Result: 1.1428571428571428
  info: Calculator application closing {"timestamp":"2024-11-14T10:13:16.834Z"}
○ yash@yash-Lenovo-Legion-5-15IMH05:~/nodejstrial$ 
```



Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                        Message
                                 No older events at this moment. Retry
2024-11-14T15:36:23.337+05:30    { "test": "basic logging", "level": "info", "message": "Test message 1" }
2024-11-14T15:36:23.339+05:30    { "test": "error logging", "level": "error", "message": "Test error message" }
2024-11-14T15:39:20.412+05:30    { "region": "us-east-1", "logGroup": "calculator-app-logs", "logStream": "calculator-2024-11-14", "level": "info", "message": "Logger initialized", "timestamp": "2024-11…
2024-11-14T15:39:20.416+05:30    { "test": "info logging", "level": "info", "message": "Info message", "timestamp": "2024-11-14T10:09:20.416Z" }
2024-11-14T15:39:20.417+05:30    { "test": "warning logging", "level": "warn", "message": "Warning message", "timestamp": "2024-11-14T10:09:20.417Z" }
2024-11-14T15:39:20.418+05:30    { "userId": 123, "action": "test", "timestamp": "2024-11-14T10:09:20.418Z", "level": "info", "message": "Structured log test" }
2024-11-14T15:39:20.418+05:30    { "test": "error logging", "level": "error", "message": "Error message", "timestamp": "2024-11-14T10:09:20.417Z" }
2024-11-14T15:42:46.124+05:30    { "region": "us-east-1", "logGroup": "calculator-app-logs", "logStream": "calculator-2024-11-14", "level": "info", "message": "Logger initialized", "timestamp": "2024-11…
2024-11-14T15:42:46.129+05:30    { "timestamp": "2024-11-14T10:12:46.128Z", "nodeVersion": "v18.19.1", "level": "info", "message": "Calculator application started" }
2024-11-14T15:43:04.773+05:30    { "region": "us-east-1", "logGroup": "calculator-app-logs", "logStream": "calculator-2024-11-14", "level": "info", "message": "Logger initialized", "timestamp": "2024-11…
2024-11-14T15:43:04.778+05:30    { "timestamp": "2024-11-14T10:13:04.777Z", "nodeVersion": "v18.19.1", "level": "info", "message": "Calculator application started" }
2024-11-14T15:43:15.831+05:30    { "expression": "8/7", "timestamp": "2024-11-14T10:13:15.830Z", "level": "info", "message": "User input received" }
2024-11-14T15:43:15.832+05:30    { "expression": "8/7", "timestamp": "2024-11-14T10:13:15.831Z", "level": "info", "message": "Calculation attempted" }
2024-11-14T15:43:15.833+05:30    { "expression": "8/7", "result": 1.1428571428571428, "timestamp": "2024-11-14T10:13:15.832Z", "level": "info", "message": "Calculation successful" }
                                 No newer events at this moment. Auto retry paused. Resume
```

© 2024, Amazon Web Services, Inc. or its affiliates.