Monitoring Memory and Multithreading with AWS, PostgreSQL, and Pytho

Python-Level Monitoring

1. psutil - Monitor memory, CPU, and thread usage.

   Example:

```
import psutil

process = psutil.Process()

print(f"Memory usage: {process.memory_info().rss / 1024 ** 2:.2f} MB")

print(f"Threads: {process.num_threads()}")
```

2. threading - Use for tracking active threads manually.

3. tracemalloc - Track memory allocation and detect memory leaks.

4. memory_profiler - Use @profile to measure memory usage line-by-line.

PostgreSQL Monitoring

1. Use pg_stat_activity to monitor active queries and connections.

   Example:

```
SELECT pid, state, query, wait_event_type, backend_start FROM pg_stat_activity;
```

2. Use psycopg2 or SQLAlchemy in Python to fetch and analyze this data programmatically.

AWS Monitoring

1. Amazon CloudWatch - Monitor EC2, RDS memory/CPU/network.

   Example using boto3 and psutil:

```
import boto3

import psutil

cloudwatch = boto3.client('cloudwatch')

cloudwatch.put_metric_data(

    Namespace='MyApp',

    MetricData=[
```

```
            {
                'MetricName': 'MemoryUsageMB',
                'Value': psutil.Process().memory_info().rss / 1024 ** 2,
                'Unit': 'Megabytes'
            }
        ]
    )
```

2. AWS X-Ray - Use to trace distributed/multithreaded applications (optional).

Best Practices

- Use psutil to monitor the Python process memory and threads.
- Push custom metrics to CloudWatch using boto3.
- Track thread activity using Python's threading/multiprocessing module.
- Use pg_stat_activity to audit database usage and load.
- Combine Python, PostgreSQL, and AWS tools for full-stack observability.