Real-Time Systems + AWS: .NET Engineer Cheat Sheet

Ingestion & Buffering

- Use `TcpListener` + `async/await` for scalable socket server.
- Buffer incoming messages with `Channel<T>` (bounded) or `BlockingCollection<T>`.
- Use `BoundedChannelFullMode.Wait/DropOldest` to control overflow.
- Log or persist dropped data for replay if needed.

Real-Time Pipeline to AWS

- Send messages to Amazon Kinesis or MSK (Kafka).
- Use Lambda or Kinesis Firehose to stream data to S3.
- Use partitioned S3 paths (e.g., /device/yyyy/MM/dd/) for efficient queries.
- Downstream: Athena, Glue, Redshift Spectrum.

Secure Communication

- Use TLS 1.2+ for all data in transit.
- Use VPC endpoints for S3/Kinesis/DynamoDB.
- Use IAM roles for ECS/EC2 (no long-term secrets).
- Encrypt data at rest with SSE-KMS or client-side encryption.

Anomaly Detection & Alerts

- Run anomaly detection in Lambda or ECS consumers.
- Route flagged events to SNS for alerts.
- Use PagerDuty or webhook for escalation.
- Store anomaly records in DynamoDB or S3 for auditing.

Scaling & Resilience

- Use Auto Scaling for ECS/Fargate consumers.
- Track buffer fill levels and instrument lag via CloudWatch.
- Use exponential backoff + circuit breakers on processing failures.
- Store raw data to S3 before transformation for recovery.

Patterns Summary

- Strategy Pattern: Handle per-instrument logic.
- Factory Pattern: Instantiate protocol or parser based on config.
- Command Pattern: Encapsulate instrument actions.
- Decorator Pattern: Add logging/metrics to instrument communication.