CloudWatch Logs & Metric Filtering

CloudWatch Logs Basics

- **Log Groups & Log Streams:**

Log Group: A collection of log streams sharing retention, access, and monitoring settings.
Log Stream: A time-ordered sequence of log events from a single source.

Metric Filters

- **Purpose:** Extract numerical metrics from raw log data.
- **Key Elements:**

Default Value: Ensures continuous data (e.g., reporting 0 when no matches are found).
Dimensions: Key-value pairs that further define the metric (e.g., service name, region).
Filter Pattern: A syntax to match terms in logs (single term, multiple terms, optional terms, exact phrases, include/exclude).

- **Usage Examples:**

Count all events: Leaving the filter pattern blank to count every log event.
HTTP 404 Errors: Using a pattern to match logs with status code 404.

Live Tail

- **Purpose:** View and analyze real-time log data to quickly identify issues.
- **Features:** Filtering, highlighting specified terms, and a live event stream.

Logs Insights

- **Interactive Querying:** Use a SQL-like language to filter, aggregate, and visualize log data.
- **Common Commands:**

fields: Select specific fields (e.g., timestamp, message).
filter: Search logs with conditions (case-sensitive).
stats: Aggregate data (e.g., count events in 5-minute bins).

Anomaly Detection & Pattern Analysis

- **Anomaly Detection:** Uses machine learning to flag deviations (e.g., spikes in error rates).
- **Pattern Recognition:** Compresses large sets of log events into recurring patterns for easier analysis.

- **Setup Considerations:** Training period (typically past 2 weeks) and a default anomaly visibility of 21 days.

Troubleshooting with VPC Flow Logs

- **Purpose:** Capture and analyze IP traffic flow data between network interfaces.
- **Usage:**

Diagnose security group or network ACL issues.
Review traffic details (source, destination, ports, protocols) for troubleshooting connectivity issues.

Application Monitoring with CloudWatch

CloudWatch Application Insights

- **Purpose:** Automatically identify, monitor, and troubleshoot key application components.
- **Key Concepts:**

Component: A logical grouping (auto-grouped, standalone, or custom) of similar resources (e.g., a web application's load balancer and EC2 instances).
Onboarding Options:
Resource Group-based: Monitor specific resource groups.
Account-based: Faster onboarding to monitor all resources.

- **Features:**

Automated dashboards, alarm generation, OpsCenter integration for operational work items.

CloudWatch Synthetics

- **Purpose:** Create canaries (synthetic transactions) that simulate user actions to monitor application availability, performance, and functionality.
- **Types of Canaries:**

Heartbeat Monitoring: Checks if a page loads (simple uptime tests).
API Canary: Tests API read/write functionality.
Broken Link Checker: Validates internal and external links.
GUI Workflow Builder & Visual Monitoring: Simulate multi-step workflows and compare screenshots to detect UI regressions.

- **Configuration & Troubleshooting:**

Canaries are Lambda functions using Node.js or Python.
Setup involves specifying endpoints, schedules, and alarms based on success rates, durations, or failure counts.

For VPC canaries, ensure proper IAM role permissions and network configurations (e.g., private subnets).

CloudWatch RUM (Real User Monitoring)

- **Purpose:** Collect and analyze client-side performance data from actual user sessions.
- **Data Collected:**

Page load times, client-side errors, navigation events, and Web Vitals (Largest Contentful Paint, First Input Delay, Cumulative Layout Shift).

- **Implementation:**

A JavaScript snippet is generated to capture data and send it to CloudWatch.
Data retention is typically 30 days (extendable via CloudWatch Logs).

- **Usage:**

Provides insights into user experience across browsers, devices, and geographies.
Data can be visualized on dashboards and used for alarming.

Application Signals

- **Purpose:** Gain a unified view of application health by automatically collecting key metrics (e.g., latency, availability, fault rates) and mapping application dependencies.
- **Features:**

Automatic discovery of services and dependencies via integrations with CloudWatch RUM, Synthetics, and AWS Service Catalog AppRegistry.
Enables setting and monitoring Service Level Objectives (SLOs) with defined Service Level Indicators (SLIs).
Visualizes application topology with a Service Map, making it easier to identify performance bottlenecks and single points of failure.

Putting It All Together

- **From Logs to Insights:**

Use **metric filters** and **Live Tail** to continuously monitor log data.
Query logs with **Logs Insights** to perform deeper analysis and troubleshooting.
Enable **anomaly detection** for proactive issue identification.

- **From Synthetic Testing to Real User Data:**

Deploy **CloudWatch Synthetics canaries** to simulate user journeys and validate endpoint performance.
Use **CloudWatch RUM** to capture real user experiences and fine-tune performance metrics.

- **Comprehensive Application Monitoring:**

CloudWatch Application Insights provides a high-level view of application health and operational issues.
Application Signals bring together metrics, traces, and topology mapping for an integrated view of service performance and dependencies.