Logging Overview

Logs are essential for monitoring and troubleshooting. In AWS, application logs from EC2 instances can be categorized into different types, e.g., Event, Server, Access logs. To collect and store logs for a Linux-based web app on EC2, the appropriate solution is to use custom logs for Amazon CloudWatch. **Custom logs** are **application-specific logs** generated by software running on EC2, such as web server logs. You can configure CloudWatch to capture these logs, where they can be stored, monitored, and for monitoring and analysis.

VPC Flow Logs capture IP **traffic to and from network interfaces** in a VPC. These logs can be integrated with services like Amazon Data Firehose for further processing and analysis. VPC Flow Logs can be delivered directly to Amazon Data Firehose, as they are an example **of vended logs**.

Amazon **CloudWatch Logs** organizes logs into **log groups** and **log streams**. A log stream is a sequence of **log events** from a single source, like an EC2 instance or Lambda function. A log stream represents all the log events from a common source, which helps organize and retrieve logs efficiently.

Amazon CloudWatch **pricing for vended logs**, such as VPC Flow Logs or ELB access logs, is based on a **per-gigabyte volume discount model**. As the volume of these logs increases beyond a certain threshold, the cost per gigabyte decreases, providing a cost-effective solution for storing large amounts of AWS service logs.

Determining a Monitoring Strategy

When creating a monitoring plan, consider how and to whom **notifications** should be sent. **Emails** and **dashboards** are suitable for individuals who need to be informed of **non-urgent events** that do not require immediate decisions or actions. These passive forms of communication provide updates without demanding instant attention, making them ideal for keeping stakeholders informed.

Establishing a Baseline with Amazon EC2 Metrics

By default, Amazon EC2 instances provide **basic monitoring** with **5-minute granularity**. To set up **1-minute** monitoring of incoming network traffic, the developer should enable **detailed monitoring** for the EC2 instance. This increases the metric collection frequency to 1-minute intervals, providing more granular data for performance analysis.

Developers can check for spikes in CPU utilization on an Amazon EC2 instance during the previous week in the Amazon EC2 or Amazon CloudWatch consoles. Both consoles display basic performance metrics like CPU utilization, allowing developers to view performance trends over time.

Monitoring with the CloudWatch Agent

Logs stored locally on an EC2 instance are lost if the instance is terminated or fails. To preserve application logs, a developer can **install an Amazon CloudWatch agent** on the EC2 instance and **publish custom application logs to CloudWatch Logs**. This ensures that even if the instance fails, the logs remain accessible in CloudWatch, preventing important log data loss.

To collect application logs for a custom app running on an **Amazon EC2** instance, a developer should **install the Amazon CloudWatch agent** on the instance. This agent enables the collection and **pushing of custom application logs to CloudWatch Logs**, facilitating centralized log management and analysis.

The *CloudWatchAgentServerPolicy* is an **AWS managed policy** that provides the necessary **permissions** for the CloudWatch agent **to send logs** from an EC2 instance to Amazon CloudWatch.

The CloudWatch agent uses a **JSON configuration file** to **define the metrics and logs it should collect** on an EC2 instance. This file allows for customized monitoring based on the application's needs.

If a company's security policy prohibits installing a CloudWatch agent on their on-premises servers, the cloud engineer should recommend migrating the workload to Amazon EC2. This will allow CloudWatch to natively monitor **basic metrics** such as metrics for bytes sent in and out on network interfaces connected to a workload, without an agent, complying with security policies while enabling the desired monitoring.

While Amazon EC2 instances provide **standard metrics** like CPU utilization by default, they **do not monitor memory usage**. To collect memory usage metrics, the Amazon CloudWatch agent must be installed on the instances.

Logging Activities with CloudTrail

To determine when and who changed an Amazon S3 bucket policy, use AWS **CloudTrail logs**. The *userIdentity* field in CloudTrail logs the IAM identity of the user who made the API request, allowing you to identify the responsible party.

Keeping Your Applications Healthy

To receive **automated alerts** about issues affecting an application running on Amazon EC2 instances, the developer should **add the application to Amazon CloudWatch Application Insights**. CloudWatch Application Insights **automaticallydetects problems** and creates alerts with minimal setup, helping to monitor application health efficiently without extensive configuration.

Serverless Observability

To gain more detailed performance metrics for AWS Lambda functions, a developer can install the **Lambda Insights** extension. Lambda Insights provides **additional metrics to monitor Lambda behavior**, offering deeper insights into function performance, resource usage, and potential issues.

CloudWatch Container Insights

When running a **containerized application on an Amazon ECS cluster**, a cloud engineer can use AWS CLI to enable **CloudWatch Container Insights**. This gathers performance metrics like CPU and memory usage across the container environment, providing insights into the application's performance within the cluster.

CloudWatch Metric Streams

A cloud engineer can efficiently collect and process a **large volume of metrics** from an application hosted on Amazon EC2 by creating an **Amazon CloudWatch metric stream** to send the metrics **to an AWS or third-party analytics service**. Metric streams enable near real-time delivery of metrics, facilitating seamless integration and timely data processing without complex setups.

AWS X-Ray

When an event-driven app with interconnected microservices experiences issues, using **AWS X-Ray** is an efficient way to troubleshoot. X-Ray makes it easier to **trace requests** and **pinpoint bottlenecks** or errors in the microservices environment. The **service map** visualizes the application architecture, allowing the developer to identify where the issues are occurring.

CloudWatch Dashboards

To **securely share** an Amazon **CloudWatch dashboard** with a third-party, the administrator should share it with the third-party's **email address**. This limits access and prevents unauthorized exposure of sensitive monitoring data.

CloudWatch Synthetics and CloudWatch RUM

Amazon **CloudWatch Synthetics** allows you to monitor your applications and APIs by creating **canaries** - synthetic **monitoring scripts** that **simulate real user actions**. Canaries can automatically test your webpages and APIs for issues like error messages, broken links, and slow performance and can simulate interactions like clicking buttons, filling out forms, and navigating through your application to ensure it works as expected.

Amazon CloudWatch RUM provides a solution by instrumenting web applications to **capture detailed user interaction data** in real-time. It collects metrics like page load times, errors, and user actions, giving developers visibility into how problems affect the **actual end-user experience**. Rather than relying only on limited testing, CloudWatch RUM gives developers a window into how their application is performing for customers.

Amazon CloudWatch Evidently

Amazon **CloudWatch Evidently** enables developers to **safely experiment with new web and mobile application features**, by testing them on specific user segments. It allows deploying features to a **subset of users**, monitoring their impact on performance and engagement metrics, and **easily rolling back changes** that negatively affect the user experience.

High-Resolution Metrics, Alarms, and Composite Alarms

To gain **more granular visibility** into the performance of a mission-critical application running on Amazon EC2, the operations administrator can **enable high-resolution metrics** through the Amazon CloudWatch agent. By default, the agent sends basic metrics to CloudWatch at a 1-minute frequency. To get higher-resolution data, the administrator can configure the agent to send metrics at **intervals as short as 1 second**.

Key Concepts to Understand

- What is CloudWatch RUM? What is it used for?
- What are custom logs for CloudWatch?
- What are Vended logs?
- What is Log stream? Where does a log stream originate?
- What is a 429 error? What does it mean?
- What is the frequency (minutes/seconds) for Default, Detailed and High-frequency monitoring?
- Why you should install the CloudWatch agent and publish logs?
- Why should you set up dashboards and emails when using CloudWatch?
- What is detailed monitoring? What is it's frequency? Why would you use it?
- What is the IAM role **CloudWatchAgentServerPolic**y? What is it used for?
- Understand why you would create a CloudWatch file on an EC2 instance
- What is AWS CloudTrail? What is it used for?
- What is the **userIdentity** field? What is it used for?
- If you have an On-Prem server that you can't download the CloudWatch agent to.......what is another option you could choose in order to get AWS CloudWatch metrics?
- What type of metric is the memory metric? True or False: Memory metric is include as a default metric w/ CloudWatch
- What are the default metrics that come with CloudWatch? True or False: Default metrics are 15 min frequency?
- What is Lambda Insights? What is it used for?
- What is CloudWatch Container Insights? What is it used for?

- What is CloudWatch Application insights? What is it used for?
- What is X-Ray? What is it used for?
- What is a CloudWatch metric stream?
- What is CloudWatch synthetics? What is a canary?