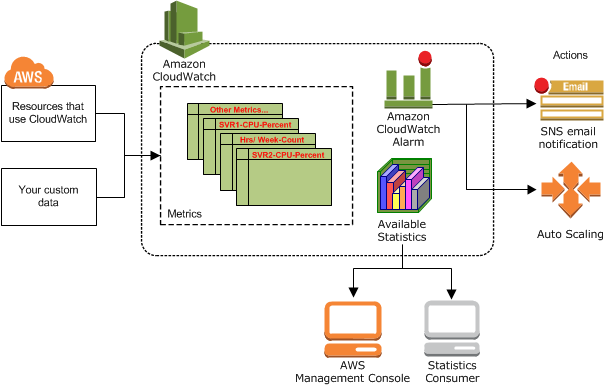
AWS CloudWatch

# AWS CloudWatch

* AWS CloudWatch monitors AWS resources and applications in real-time.
* CloudWatch can be used to collect and track metrics, which are the variables to be measured for resources and applications.
* CloudWatch alarms can be configured
  + to send notifications or
  + to automatically make changes to the resources based on defined rules
* In addition to monitoring the built-in metrics that come with AWS, custom metrics can also be monitored
* CloudWatch provides system-wide visibility into resource utilization, application performance, and operational health.
* By default, **CloudWatch stores the log data indefinitely**, and the retention can be changed for each log group at any time
* **CloudWatch Alarm history is stored for only 14 days**

#### **Required Mainly for SysOps Associate & DevOps Professional Exam**

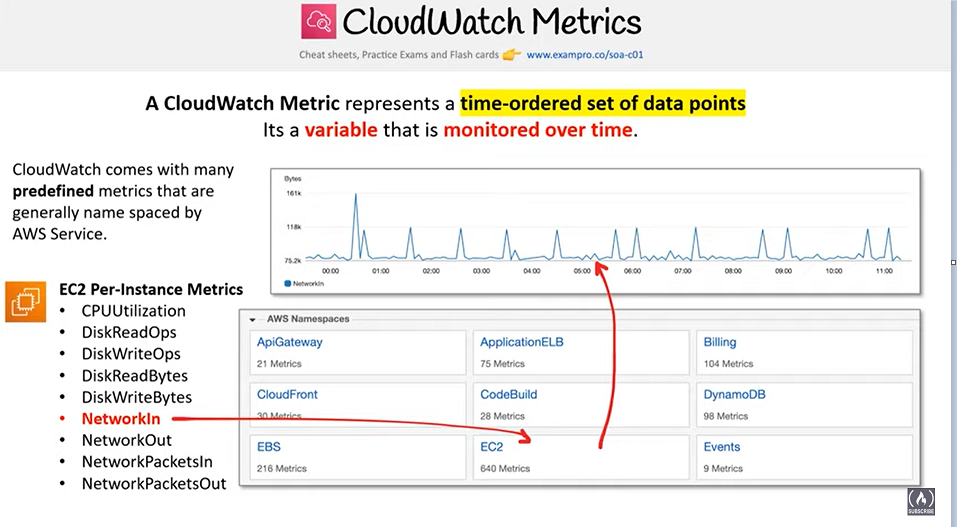
## CloudWatch Architecture



* CloudWatch collects various metrics from various resources
* These metrics, as statistics, are available to the user through Console, CLI
* CloudWatch allows creation of alarms with defined rules
  + to perform actions to auto scaling or stop, start, or terminate instances
  + to send notifications using SNS actions on your behalf

## CloudWatch Concepts

### Metrics



* Metric is the fundamental concept in CloudWatch.
* Uniquely defined by a name, a namespace, and one or more dimensions
* Represents a time-ordered set of data points published to CloudWatch.
* Each data point has a time stamp, and (optionally) a unit of measure
* Data points can be either custom metrics or metrics from other  
  services in AWS.
* Statistics can be retrieved about those data points as an ordered set of time-series data that occur within a specified time window.
* When the statistics are requested, the returned data stream is identified by namespace, metric name, dimension, and (optionally) the unit.
* **Metrics exist only in the region in which they are created**
* **CloudWatch stores the metric data for two weeks**
* **Metrics cannot be deleted, but they automatically expire in 14 days if no new data is published to them.**
* **NOTE: From**[**Nov 2016 AWS provides Extended Metrics Retention**](https://aws.amazon.com/blogs/aws/amazon-cloudwatch-update-extended-metrics-retention-user-interface-update/)
  + **One minute** data points are available for **15 days**.
  + **Five minute** data points are available for **63 days**.
  + **One hour** data points are available for 455 days (**15 months).**

### Namespaces

* CloudWatch namespaces are containers for metrics.
* Metrics in different namespaces are isolated from each other, so that metrics from different applications are not mistakenly aggregated into the same statistics.
* AWS namespaces all follow the convention AWS/<service>, for e.g. AWS/EC2 and AWS/ELB
* Namespace names must be fewer than 256 characters in length.
* **There is no default namespace. Each data element put into CloudWatch must specify a namespace**

### Dimensions

* A dimension is a name/value pair that uniquely identifies a metric.
* Every metric has specific characteristics that describe it, and you can think of dimensions as categories for those characteristics.
* Dimensions helps design a structure for the statistics plan.
* Dimensions are part of the unique identifier for a metric, whenever a unique name pair is added to one of the metrics, a new metric is created
* Dimensions can be used to filter result sets that CloudWatch query returns
* A metric can be assigned up to **ten dimensions to a metric**.

### Time Stamps

* Each metric data point must be marked with a time stamp to identify the data point on a time series
* **Time stamp can be up to two weeks in the past** and up to **two hours into the future**.
* If no time stamp is provided, CloudWatch creates a time stamp based on the time the data element was received.
* All times reflect the UTC time zone when statistics are retrieved

### Units

* Units represent the statistic’s unit of measure for e.g. count, bytes, % etc

### Statistics

* Statistics are metric data aggregations over specified periods of time (i.e. Maximum, Minimum, Average, Sum, Pxx, )
* Aggregations are made using the namespace, metric name, dimensions, and the data point unit of measure, within the specified time period

### Periods

* Period is the length of time associated with a specific statistic.
* Each statistic represents an aggregation of the metrics data collected for a specified period of time.
* Although periods are expressed in seconds, the minimum granularity for a period is one minute.

### Aggregation

* CloudWatch aggregates statistics according to the period length specified in calls to GetMetricStatistics.
* Multiple data points can be published with the same or similar time stamps. CloudWatch aggregates them by period length when the statistics about those data points are requested.
* Aggregated statistics are only available when using detailed monitoring.
* Instances that use basic monitoring are not included in the aggregates
* **CloudWatch does not aggregate data across regions.**

### Alarms

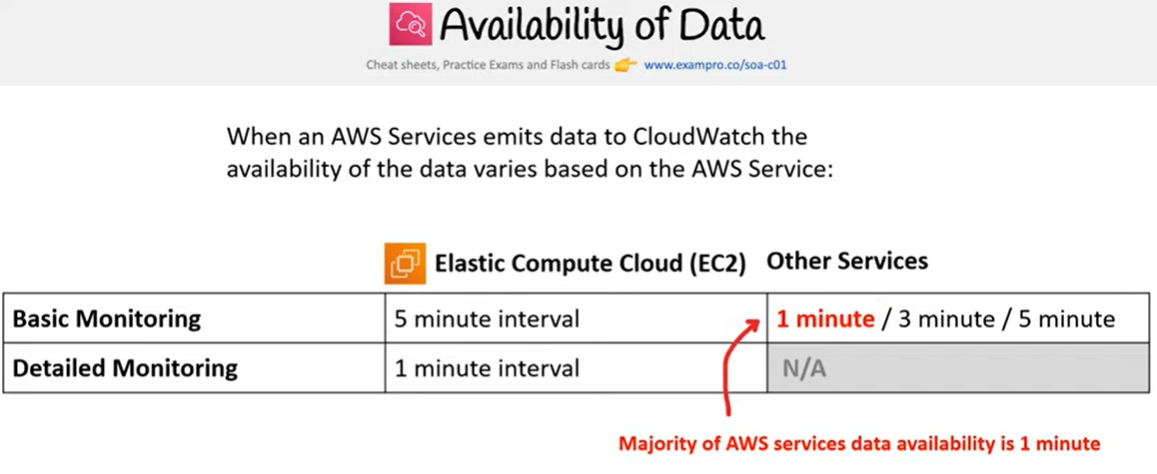
* Alarms can automatically initiate actions on behalf of the user, based on specified parameters
* Alarm watches a single metric over a specified time period, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods.
* Alarms invoke actions for sustained state changes only i.e. the state must have changed and been maintained for a specified number of periods
* Action can be a
  + SNS notification
  + Auto Scaling policies
  + EC2 action – stop or terminate EC2 instances
* After an alarm invokes an action due to a change in state, its subsequent behavior depends on the type of action associated with the alarm.
  + For Auto Scaling policy notifications, the alarm continues to invoke the action for every period that the alarm remains in the new state.
  + For SNS notifications, no additional actions are invoked.
* An alarm has three possible states:
  + **OK**—The metric is within the defined threshold
  + **ALARM**—The metric is outside of the defined threshold
  + **INSUFFICIENT\_DATA**—Alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state
* **Alarms exist only in the region in which they are created.**
* **Alarm actions must reside in the same region as the alarm**
* **Alarm history is available for the last 14 days.**
* **Alarm can be tested by setting it to any state using the SetAlarmState API (mon-set-alarm-state command). This temporary state change lasts only until the next alarm comparison occurs.**
* **Alarms can be disabled and enabled using the DisableAlarmActions and EnableAlarmActions APIs (mon-disable-alarm-actions and mon-enable-alarm-actions commands).**

### Regions

* **CloudWatch does not aggregate data across regions. Therefore, metrics are completely separate between regions.**

## Custom Metrics

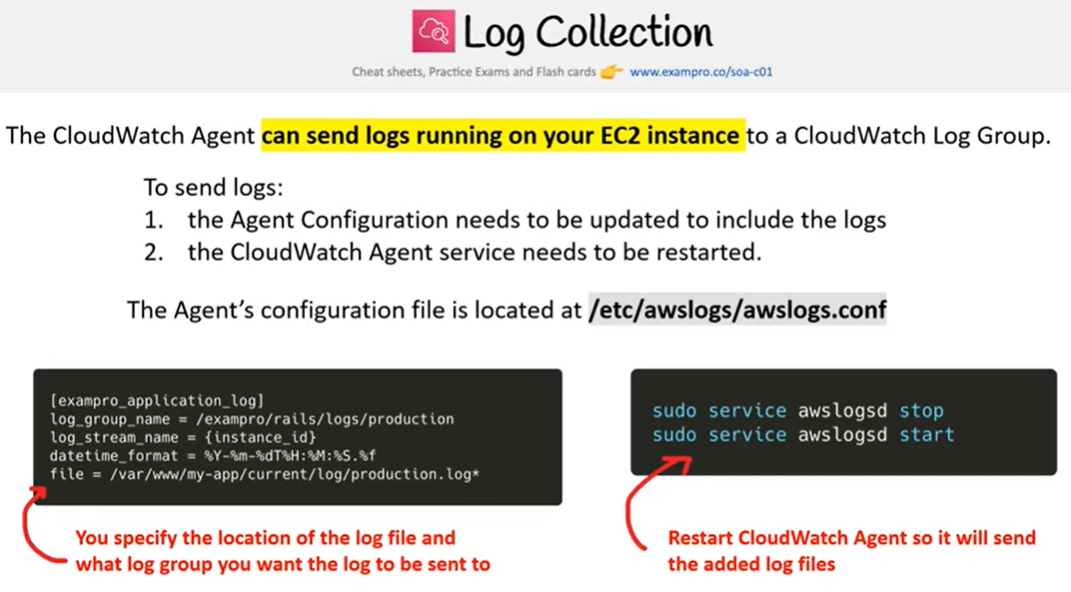
* **CloudWatch allows publishing custom metrics with put-metric-data CLI command (or its Query API equivalent PutMetricData)**
* **CloudWatch creates a new metric if put-metric-data is called with a new metric name,  else it associates the data with the specified existing metric**
* **put-metric-data command can only publish one data point per call**
* CloudWatch stores data about a metric as a series of data points and each data point has an associated time stamp
* Creating a new metric using the put-metric-data command, can take up to two minutes before statistics can be retrieved on the new metric using the get-metric-statistics command and can take up to fifteen minutes before the new metric appears in the list of metrics retrieved using the list-metrics command.
* CloudWatch allows publishing
  + **Single data point**
    - Data points can be published with time stamps as granular as one-thousandth of a second, CloudWatch aggregates the data to a minimum granularity of one minute
    - CloudWatch records the average (sum of all items divided by number of items) of the values received for every 1-minute period, as well as number of samples, maximum value, and minimum value for the same time period
    - CloudWatch uses one-minute boundaries when aggregating data points
  + **Aggregated set of data points called a statistics set**
    - Data can also be aggregated before being published to CloudWatch
    - Aggregating data minimizes the number of calls reducing it to a single call per minute with the statistic set of data
    - Statistics include Sum, Average, Minimum, Maximum, SampleCount
* **If the application produces data that is more sporadic and have periods that have no associated data, either a the value zero (0) or no value at all can be published**
* **However, it can be helpful to publish zero instead of no value**
  + **to monitor the health of your application for e.g. alarm can be configured to notify if no metrics published every 5 minutes**
  + **to track the total number of data points**
  + **to have statistics such as minimum and average to include data points with the value 0**.

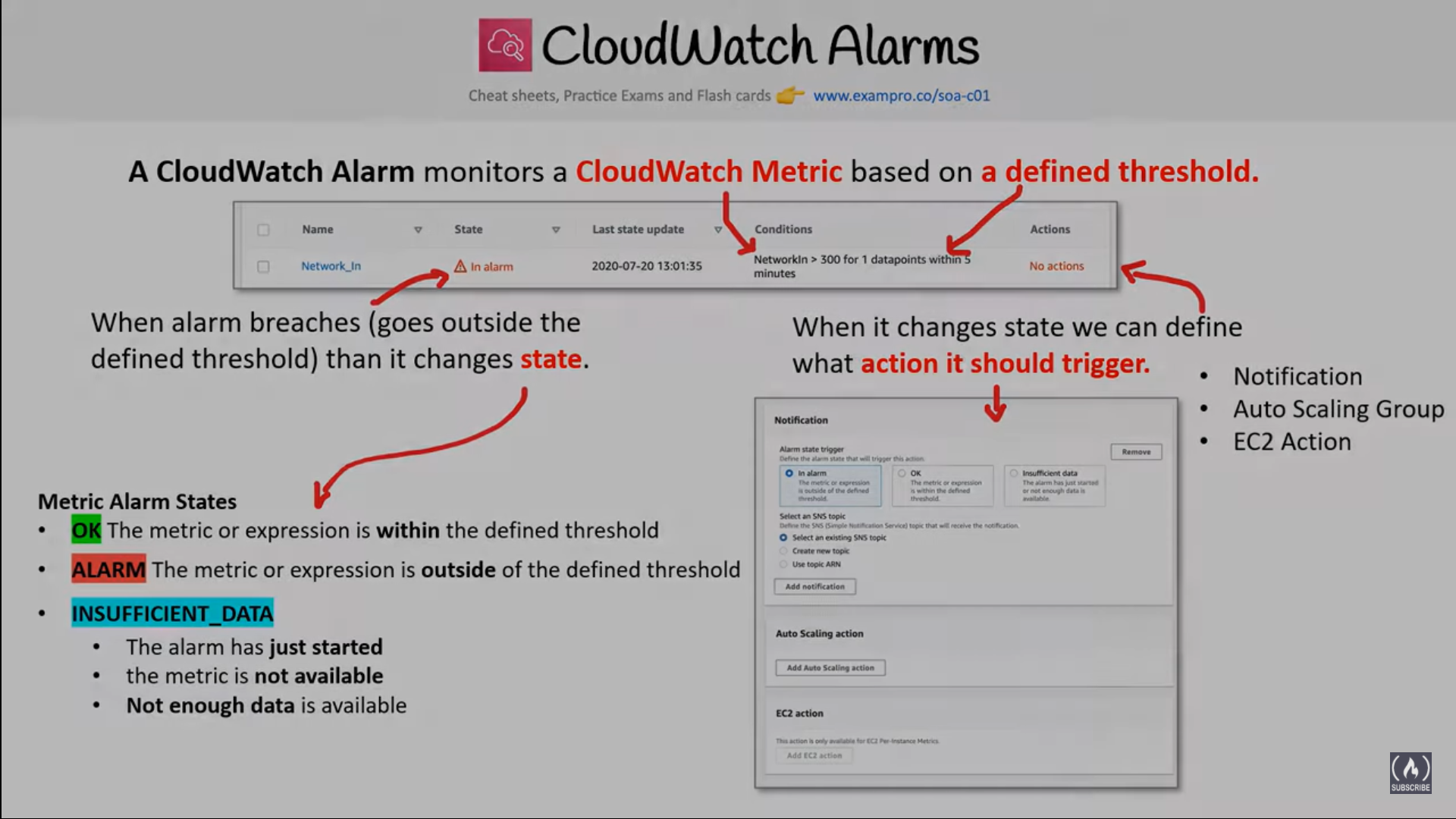


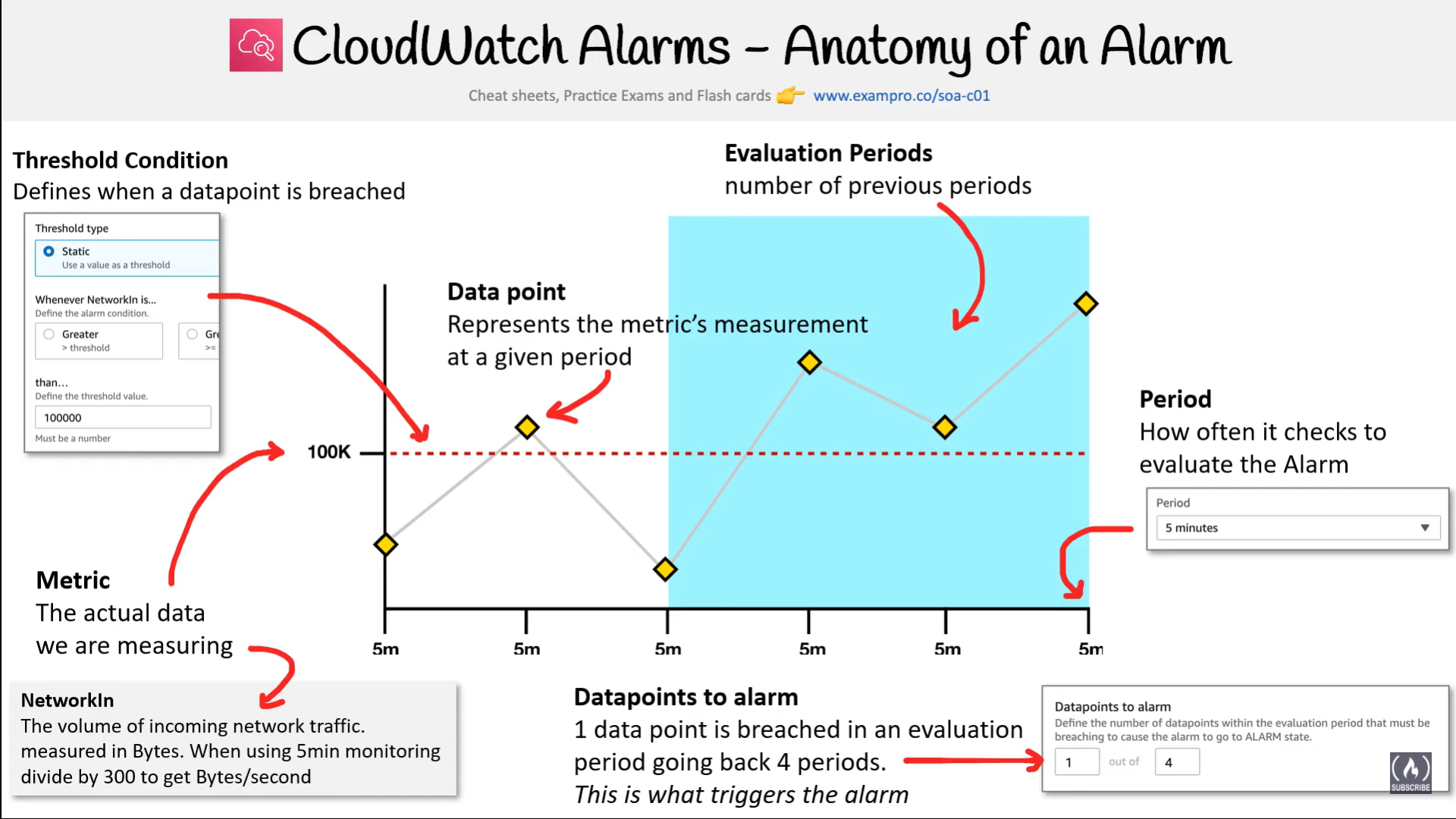


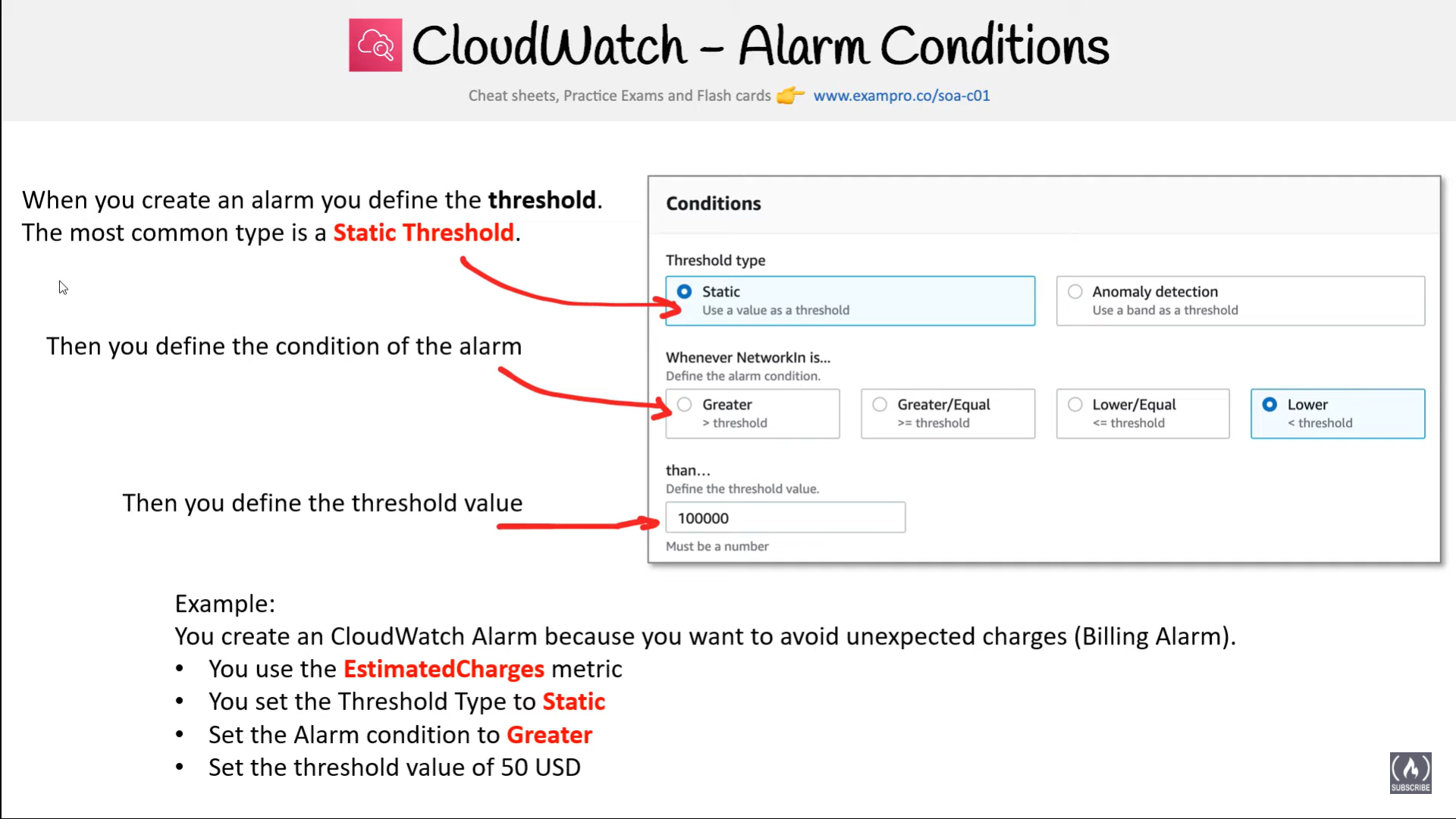


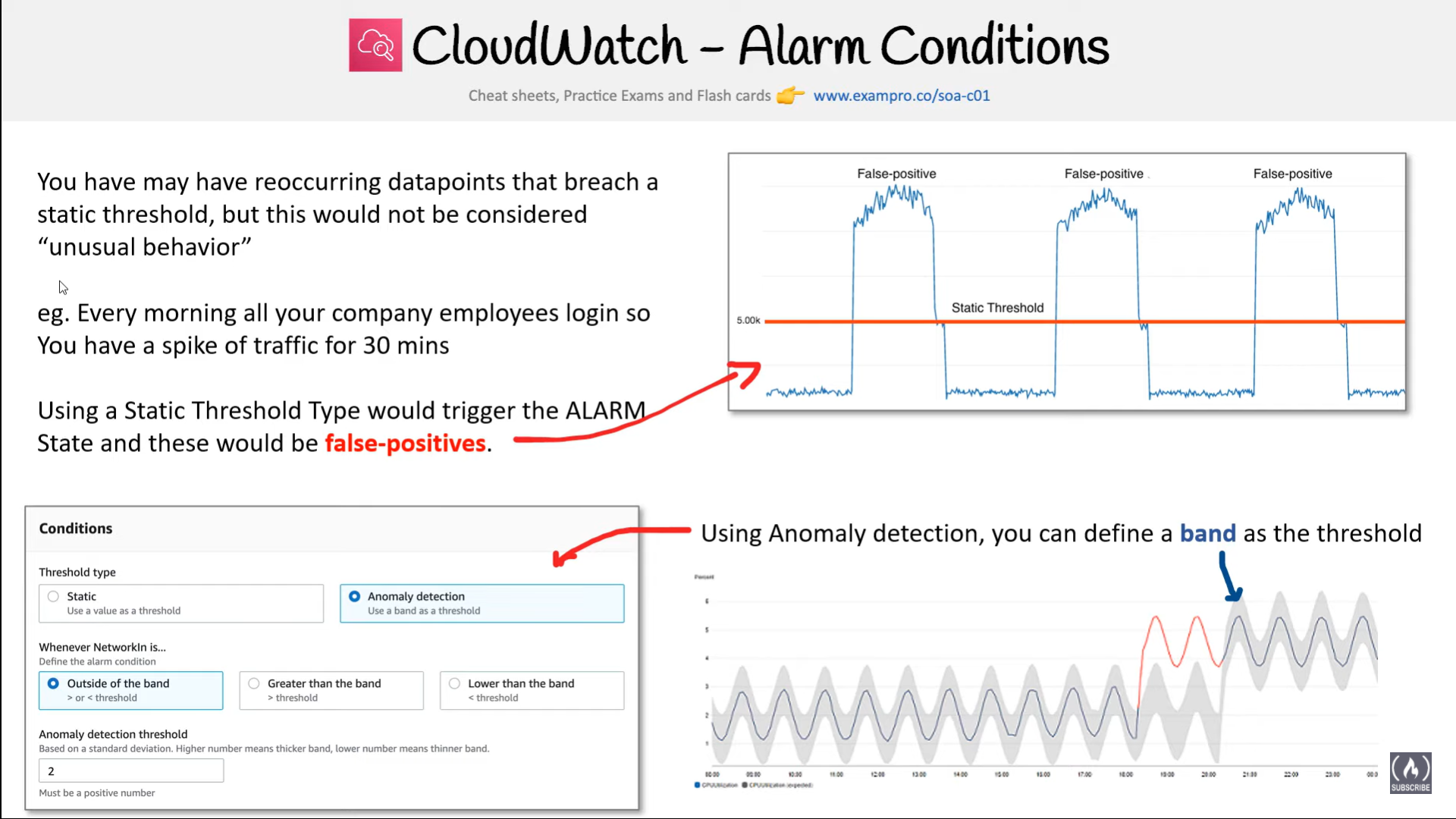




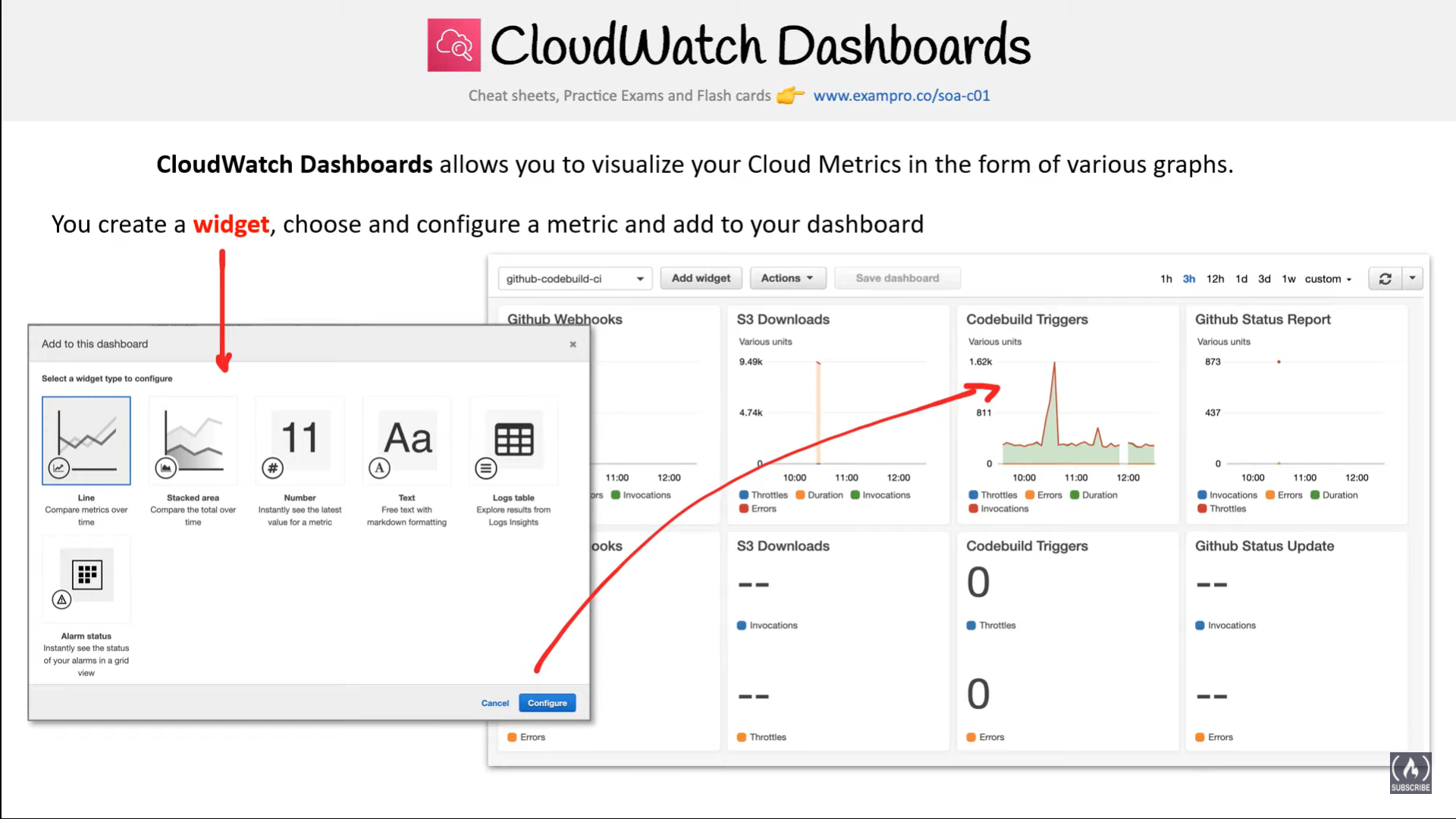












## Accessing CloudWatch

* CloudWatch can be accessed using
  + AWS CloudWatch console
  + CloudWatch CLI
  + AWS CLI
  + CloudWatch API
  + AWS SDKs

## AWS Certification Exam Practice Questions

1. A company needs to monitor the read and write IOPs metrics for their AWS MySQL RDS instance and send real-time alerts to their operations team. Which AWS services can accomplish this? Choose 2 answers
   1. Amazon Simple Email Service (Cannot be integrated with CloudWatch directly)
   2. **Amazon CloudWatch**
   3. Amazon Simple Queue Service
   4. Amazon Route 53
   5. **Amazon Simple Notification Service**
2. A customer needs to capture all client connection information from their load balancer every five minutes. The company wants to use this data for analyzing traffic patterns and troubleshooting their applications. Which of the following options meets the customer requirements?
   1. Enable AWS CloudTrail for the load balancer.
   2. **Enable access logs on the load balancer.** (Refer [link](https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/access-log-collection.html))
   3. Install the Amazon CloudWatch Logs agent on the load balancer.
   4. Enable Amazon CloudWatch metrics on the load balancer (does not provide Client connection information)
3. A user is running a batch process on EBS backed EC2 instances. The batch process starts a few instances to process Hadoop Map reduce jobs, which can run between 50 – 600 minutes or sometimes for more time. The user wants to configure that the instance gets terminated only when the process is completed. How can the user configure this with CloudWatch?
   1. **Setup the CloudWatch action to terminate the instance when the CPU utilization is less than 5%**
   2. Setup the CloudWatch with Auto Scaling to terminate all the instances
   3. Setup a job which terminates all instances after 600 minutes
   4. It is not possible to terminate instances automatically
4. A user has two EC2 instances running in two separate regions. The user is running an internal memory management tool, which captures the data and sends it to CloudWatch in US East, using a CLI with the same namespace and metric. Which of the below mentioned options is true with respect to the above statement?
   1. The setup will not work as CloudWatch cannot receive data across regions
   2. **CloudWatch will receive and aggregate the data based on the namespace and metric**
   3. CloudWatch will give an error since the data will conflict due to two sources
   4. CloudWatch will take the data of the server, which sends the data first
5. A user is sending the data to CloudWatch using the CloudWatch API. The user is sending data 90 minutes in the future. What will CloudWatch do in this case?
   1. **CloudWatch will accept the data (**With Amazon CloudWatch, The user can send the  
      data using CLI but the time has to be in the UTC format. If the user does not provide the time, CloudWatch will  
      take the data received time in the UTC timezone. The time stamp sent by the user can be up to two weeks in  
      the past and up to two hours into the future.)
   2. It is not possible to send data of the future
   3. It is not possible to send the data manually to CloudWatch
   4. The user cannot send data for more than 60 minutes in the future
6. A user is having data generated randomly based on a certain event. The user wants to upload that data to CloudWatch. It may happen that event may not have data generated for some period due to randomness. Which of the below mentioned options is a recommended option for this case?
   1. For the period when there is no data, the user should not send the data at all
   2. For the period when there is no data the user should send a blank value
   3. **For the period when there is no data the user should send the value as 0**(Refer [User Guide)](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/publishingMetrics.html#publishingZero)
   4. The user must upload the data to CloudWatch as having no data for some period will cause an error at CloudWatch monitoring
7. A user has a weighing plant. The user measures the weight of some goods every 5 minutes and sends data to AWS CloudWatch for monitoring and tracking. Which of the below mentioned parameters is mandatory for the user to include in the request list?
   1. Value
   2. **Namespace (**refer [put-metric request](http://docs.aws.amazon.com/cli/latest/reference/cloudwatch/put-metric-data.html)**)**
   3. Metric Name
   4. Timezone
8. A user has a refrigerator plant. The user is measuring the temperature of the plant every 15 minutes. If the user wants to send the data to CloudWatch to view the data visually, which of the below mentioned statements is true with respect to the information given above?
   1. **The user needs to use AWS CLI or API to upload the data**
   2. The user can use the AWS Import Export facility to import data to CloudWatch
   3. The user will upload data from the AWS console
   4. The user cannot upload data to CloudWatch since it is not an AWS service metric
9. A user has launched an EC2 instance. The user is planning to setup the CloudWatch alarm. Which of the below mentioned actions is not supported by the CloudWatch alarm?
   1. **Notify the Auto Scaling launch config to scale up**
   2. Send an SMS using SNS
   3. Notify the Auto Scaling group to scale down
   4. Stop the EC2 instance
10. A user has a refrigerator plant. The user is measuring the temperature of the plant every 15 minutes. If the user wants to send the data to CloudWatch to view the data visually, which of the below mentioned statements is true with respect to the information given above?
    1. **The user needs to use AWS CLI or API to upload the data**
    2. The user can use the AWS Import Export facility to import data to CloudWatch
    3. The user will upload data from the AWS console
    4. The user cannot upload data to CloudWatch since it is not an AWS service metric
11. A user is trying to aggregate all the CloudWatch metric data of the last 1 week. Which of the below mentioned statistics is not available for the user as a part of data aggregation?
    1. **Aggregate**
    2. Sum
    3. Sample data
    4. Average
12. A user has setup a CloudWatch alarm on an EC2 action when the CPU utilization is above 75%. The alarm sends a notification to SNS on the alarm state. If the user wants to simulate the alarm action how can he achieve this?
    1. Run activities on the CPU such that its utilization reaches above 75%
    2. From the AWS console change the state to ‘Alarm’
    3. **The user can set the alarm state to ‘Alarm’ using CLI**
    4. Run the SNS action manually
13. A user is publishing custom metrics to CloudWatch. Which of the below mentioned statements will help the user understand the functionality better?
    1. The user can use the CloudWatch Import tool
    2. **The user should be able to see the data in the console after around 15 minutes**
    3. If the user is uploading the custom data, the user must supply the namespace, timezone, and metric name as part of the command
    4. The user can view as well as upload data using the console, CLI and APIs
14. An application that you are managing has EC2 instances and DynamoDB tables deployed to several AWS Regions. In order to monitor the performance of the application globally, you would like to see two graphs 1) Avg CPU Utilization across all EC2 instances and 2) Number of Throttled Requests for all DynamoDB tables. How can you accomplish this? **[PROFESSIONAL]**
    1. Tag your resources with the application name, and select the tag name as the dimension in the CloudWatch Management console to view the respective graphs (CloudWatch metrics are regional)
    2. **Use the CloudWatch CLI tools to pull the respective metrics from each regional endpoint. Aggregate the data offline & store it for graphing in CloudWatch.**
    3. Add SNMP traps to each instance and DynamoDB table. Leverage a central monitoring server to capture data from each instance and table. Put the aggregate data into CloudWatch for graphing (Can’t add SNMP traps to DynamoDB as it is a managed service)
    4. Add a CloudWatch agent to each instance and attach one to each DynamoDB table. When configuring the agent set the appropriate application name & view the graphs in CloudWatch. (Can’t add agents to DynamoDB as it is a managed service)
15. You have set up Individual AWS accounts for each project. You have been asked to make sure your AWS Infrastructure costs do not exceed the budget set per project for each month. Which of the following approaches can help ensure that you do not exceed the budget each month? **[PROFESSIONAL]**
    1. Consolidate your accounts so you have a single bill for all accounts and projects (Consolidation will not help limit per account)
    2. Set up auto scaling with CloudWatch alarms using SNS to notify you when you are running too many Instances in a given account (many instances do not directly map to cost and would not give exact cost)
    3. Set up CloudWatch billing alerts for all AWS resources used by each project, with a notification occurring when the amount for each resource tagged to a particular project matches the budget allocated to the project. (as each project already has a account, no need for resource tagging)
    4. **Set up CloudWatch billing alerts for all AWS resources used by each account, with email notifications when it hits 50%. 80% and 90% of its budgeted monthly spend**
16. You meet once per month with your operations team to review the past month’s data. During the meeting, you realize that 3 weeks ago, your monitoring system which pings over HTTP from outside AWS recorded a large spike in latency on your 3-tier web service API. You use DynamoDB for the database layer, ELB, EBS, and EC2 for the business logic tier, and SQS, ELB, and EC2 for the presentation layer. Which of the following techniques will NOT help you figure out what happened?
    1. Check your CloudTrail log history around the spike’s time for any API calls that caused slowness.
    2. **Review CloudWatch Metrics graphs to determine which component(s) slowed the system down.**(Metrics data was available for 2 weeks before, however it has been extended now)
    3. Review your ELB access logs in S3 to see if any ELBs in your system saw the latency.
    4. Analyze your logs to detect bursts in traffic at that time.
17. You have a high security requirement for your AWS accounts. What is the most rapid and sophisticated setup you can use to react to AWS API calls to your account?
    1. Subscription to AWS Config via an SNS Topic. Use a Lambda Function to perform in-flight analysis and reactivity to changes as they occur.
    2. Global AWS CloudTrail setup delivering to S3 with an SNS subscription to the deliver notifications, pushing into a Lambda, which inserts records into an ELK stack for analysis.
    3. Use a CloudWatch Rule ScheduleExpression to periodically analyze IAM credential logs. Push the deltas for events into an ELK stack and perform ad-hoc analysis there.
    4. **CloudWatch Events Rules, which trigger based on all AWS API calls, submitting all events to an AWS Kinesis Stream for arbitrary downstream analysis.**(CloudWatch Events allow subscription to AWS API calls, and direction of these events into Kinesis Streams. This allows a unified, near real-time stream for all API calls, which can be analyzed with any tool(s). Refer [link](http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/EventTypes.html#api_event_type))
18. To monitor API calls against our AWS account by different users and entities, we can use \_\_\_\_ to create a history of calls in bulk for later review, and use \_\_\_\_ for reacting to AWS API calls in real-time.
    1. AWS Config; AWS Inspector
    2. AWS CloudTrail; AWS Config
    3. **AWS CloudTrail; CloudWatch Events**(CloudTrail is a batch API call collection service, CloudWatch Events enables real-time monitoring of calls through the Rules object interface. Refer [link](https://aws.amazon.com/whitepapers/security-at-scale-governance-in-aws/))
    4. AWS Config; AWS Lambda
19. You are hired as the new head of operations for a SaaS company. Your CTO has asked you to make debugging any part of your entire operation simpler and as fast as possible. She complains that she has no idea what is going on in the complex, service-oriented architecture, because the developers just log to disk, and it’s very hard to find errors in logs on so many services. How can you best meet this requirement and satisfy your CTO? **[PROFESSIONAL]**
    1. Copy all log files into AWS S3 using a cron job on each instance. Use an S3 Notification Configuration on the <code>PutBucket</code> event and publish events to AWS Lambda. Use the Lambda to analyze logs as soon as they come in and flag issues. (is not fast in search and introduces delay)
    2. Begin using CloudWatch Logs on every service. Stream all Log Groups into S3 objects. Use AWS EMR cluster jobs to perform adhoc MapReduce analysis and write new queries when needed. (is not fast in search and introduces delay)
    3. Copy all log files into AWS S3 using a cron job on each instance. Use an S3 Notification Configuration on the <code>PutBucket</code> event and publish events to AWS Kinesis. Use Apache Spark on AWS EMR to perform at-scale stream processing queries on the log chunks and flag issues. (is not fast in search and introduces delay)
    4. **Begin using CloudWatch Logs on every service. Stream all Log Groups into an AWS Elasticsearch Service Domain running Kibana 4 and perform log analysis on a search cluster.**(ELK – Elasticsearch, Kibana stack is designed specifically for real-time, ad-hoc log analysis and aggregation)
20. Your EC2-Based Multi-tier application includes a monitoring instance that periodically makes application -level read only requests of various application components and if any of those fail more than three times 30 seconds calls CloudWatch to fire an alarm, and the alarm notifies your operations team by email and SMS of a possible application health problem. However, you also need to watch the watcher -the monitoring instance itself – and be notified if it becomes unhealthy. Which of the following is a simple way to achieve that goal? **[PROFESSIONAL]**
    1. Run another monitoring instance that pings the monitoring instance and fires a could watch alarm mat notifies your operations team should the primary monitoring instance become unhealthy.
    2. **Set a CloudWatch alarm based on EC2 system and instance status checks and have the alarm notify your operations team of any detected problem with the monitoring instance.**
    3. Set a CloudWatch alarm based on the CPU utilization of the monitoring instance and nave the alarm notify your operations team if C r the CPU usage exceeds 50% few more than one minute: then have your monitoring application go into a CPU-bound loop should it Detect any application problems.
    4. Have the monitoring instances post messages to an SOS queue and then dequeue those messages on another instance should the queue cease to have new messages, the second instance should first terminate the original monitoring instance start another backup monitoring instance and assume (the role of the previous monitoring instance and beginning adding messages to the SQS queue.