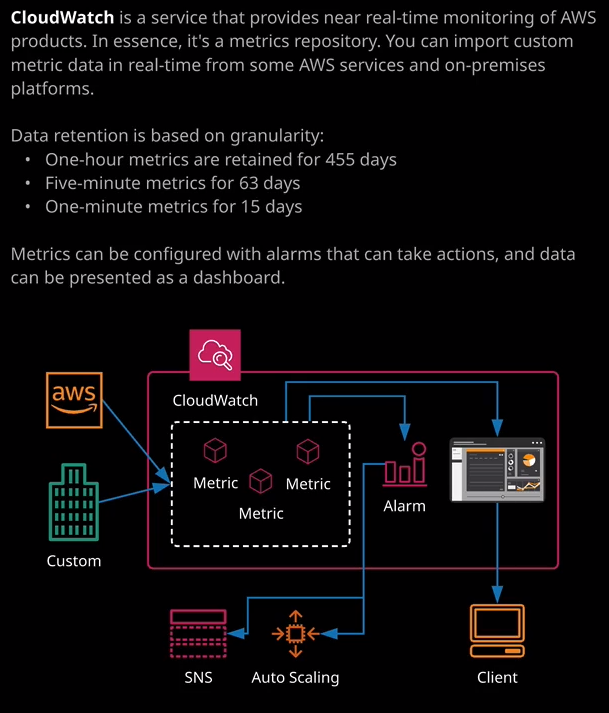
Welcome back and welcome to this lesson where I want to introduce CloudWatch. Now **CloudWatch is a core product in AWS. In many ways, it's as important as IAM. CloudWatch provides a full suite of metric collection and monitoring functionality, and it's used in pretty much all AWS products**. When you click a monitoring tab on anything inside AWS, it's CloudWatch data which get shown. With CloudWatch, the architecture is simple AWS products and any custom sources of data that you configure send all of their data or their data points into CloudWatch. They get grouped up together into metrics, and then we can take actions on that data. We can generate alarms. We can use it to create monitoring dashboards. But essentially the sole function of CloudWatch is to act as a repository for metric data that is its key piece of functionality.

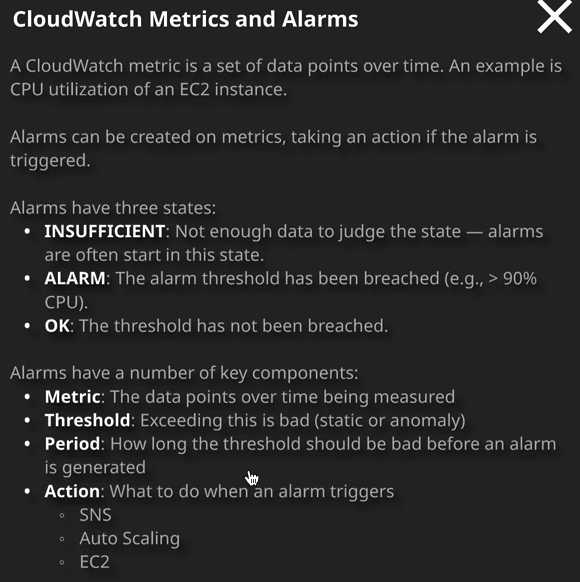


Now I'll make sure that this link is in the lesson description, but this is a pretty comprehensive list of the services that publish data into CloudWatch, and most of them publish a pretty comprehensive set of metrics. So most of the elements of these services, everything you can monitor, is published directly into CloudWatch. Now, as well as AWS services, you can publish data directly into CloudWatch using the APIs or the CloudWatch agent. Now you'll get some experience of using the CloudWatch agent in a hands-on lab that's at the end of this topic. So I won't be covering the CloudWatch agent in detail in this lesson but don't worry, you will be getting some practical exposure to it.

Now one of the key things to understand for the exam is what data CloudWatch monitors by default and what it doesn't. So let me move across to the CloudWatch console. So I want you to imagine an EC2 instance that's running on an EC2 host. By **default, anything that you can visualize as being external to the EC2 instance or visible from the EC2 host is going to be logged into CloudWatch. So things like network usage, CPU usage, they arrive by default. They're injected into CloudWatch by these AWS services. The things that you don't see by default with CloudWatch specifically for EC2 are things like internal metrics. So the memory usage of individual processes or overall memory utilization. Those things are not captured by default, and that's what the CloudWatch agent provides.** It sits internally on these EC2 instances and injects this data into CloudWatch and in my opinion, you should install this agent by default because it gives you access to a lot more rich data than just using the default metrics that the service logs.

So this is the overall architecture of CloudWatch. We already know that we've got AWS services injecting data into CloudWatch, but we've also got all the custom sources. Now, this could be an application that you code as a developer, which is using API calls to put data directly into CloudWatch or it could be the CloudWatch agent that sitting on either on-premises servers or EC2 instances that's injecting that data on your behalf. Essentially, you want to think of **CloudWatch as a repository service for metric data. Now, at this point I want to talk about exactly what I mean by a metric**.

So CloudWatch itself works with metrics. A metric is essentially a time ordered set of data points, so if we go into metrics and go on to EC2 and then let's say we go on to per instance metrics. So for a particular EC2 instance, we've got a metric CPU utilization and we've got some data. So this is in UTC that this has monitored a number of points of CPU utilization for this EC2 instance. **So a metric is essentially a time ordered set of data points.** Now, each of these is a data point, so a data point is an actual value for a specific point in time. If you imagine an EC2 instance and let's say that you're reporting the CPU utilization every five minutes. Well each one of those five minutes would be a data point, and it would be a data point for the metric of CPU utilization on that instance. So the data point is the point in time value, and the metric is a collection of all of those data points. So when I talk about metrics, think of CPU utilization of a particular EC2 instance but it could also be the number of write operations for a particular EBS volume, it could be the consumed read capacity units for a particular DynamoDB table, they're all examples of metrics.   
Now the frequency of data that's being injected into CloudWatch depends on the service, so certain services such as EC2 might, by default, deliver CPU utilization every five minutes. Certain services you might be able to configure to do a lot more regular reporting, maybe once every minute. What **CloudWatch does is it aggregates data the older that it is**. So you could access to a 60 second frequency for three hours. After three hours, that's aggregated to every 60 seconds, and that's available for 15 days. After 15 days, that's aggregated to every five minutes, and that's available for 63 days. After 63 days, it's aggregated to hourly data, and that's available for 455 days, which is 15 months. **So essentially the older data gets, the less granularity there is. But that's logical because generally, super detailed data only matters in the short term. Over long term, you're looking for trends**. Now feel free on your own account to have a look around the various different metrics that are available. As you use services inside an AWS account, you'll see additional metrics pop up. So the first time a data point is added for a particular metric, you'll see it in the list of available metrics. You won't see them all. They only appear when you use them.



Now metrics are actually grouped into what's known as namespaces. So one example namespace would be AWS/EC2. Think of these as grouping. So inside these you've got lots of different available metrics. So the namespace is essentially just a container or a grouping. Now as well as just allowing you to visualize your data, CloudWatch also allows you to take actions on metrics. You can create alarms on them, and you saw this earlier when I demo'd auto scaling groups and it's part of the rules that controlled whether an auto scaling group would add or remove instances, I created alarms for CPU utilization. So when you create an alarm, you define a threshold for that alarm. So let's go to one example that I've already got created in my account, and that's a billing alarm. Everyone should have a billing alarm in their account. This alarm is created with a threshold and what this defines is if estimated charges, which were in dollars are greater than 10 for one or more data points within six hours, then go into alarm state. So thresholds define what is good and what is bad, and you can actually define thresholds in two ways. The historic way uses static. So you define a condition if this value is more or less than a certain value for a certain number of periods then alarm. But one of the features that's currently in preview and by the time you watch this lesson it may have gone to full release is that you can actually allow CloudWatch to perform anomaly detection. So, for example, with CPU utilization, you could enable, on a per metric basis, that CloudWatch analyze all of the data for CPU utilization and define a normal level. So the level of CPU utilization on this graph is the one that's in dark blue and the detected normal range is what's indicated in light grey. What this would mean is if the CPU utilization went outside of this gray area then it would flag as an anomaly and the alarm would go into an alarm state. So alarms can either use static thresholds or in the future you'll be able to define anomaly detection. So essentially, when you're creating an alarm, you specify the metric, you specify the threshold, you specified the number of periods that that threshold would need to be breached and if that occurs, the alarm goes into the alarm state. If that doesn't occur, the alarm is in an OK state. So what this is indicating right now is this AWS account has an expected bill of more than $10. So this is always an alarm I create on any brand new AWS account that if the monthly expected charge goes over $10 then it goes into an alarm state and it sends me an email. So for an alarm, you can define an action. It can either be simple. It can notify via email or as you saw with the auto scaling groups, they could actually perform certain actions, and that's important to understand for the exam. With that being said, that is everything that I wanted to cover in this lesson. I want to keep it relatively brief because you'll only need to be aware of the high level architecture for the exam and you'll be tending to use CloudWatch data for other things across your AWS account. So go ahead mark this lesson as complete and when you're ready, you can join me in the next.