MANAGEMENT AND GOVERNANCE
IN
AWS

CloudWatch and Metrics list

Amazon CloudWatch is a monitoring and observability service that provides insights into AWS resources and applications using easily readable dashboards. It collects and tracks metrics, logs, and events, allowing us to monitor performance, detect anomalies, and set alarms for operational issues. CloudWatch creates its dashboards and performs its operations based on something called Metrics, numerical data points that represent the performance of our systems.

Said metrics help us track key performance indicators (KPIs) such as CPU usage, memory utilization, and request counts, enabling us to analyze and optimize the performance of our applications and infrastructure. CloudWatch has three metrics setup by default and another five custom metrics which can be enabled as per our requirement, both of which are worth remembering and noting, especially for cloud administrators.

The three default metrics in CloudWatch are as follows:

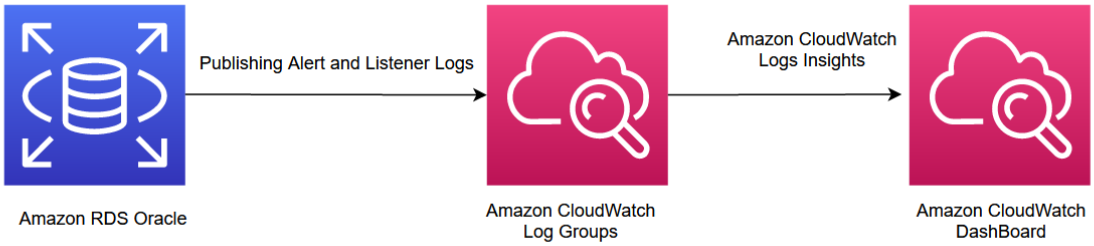
- **CPU Utilization of an EC2 instance**
- **Disk Reads activity of an EC2 instance**
- **Network packets out of an EC2 instance**

While the five custom metrics readily available for setup are as follows:

- **Memory utilization**
- **Disk swap utilization**
- **Disk space utilization**
- **Page file utilization**
- **Log collection**

CloudWatch Logs Streams

As mentioned before, CloudWatch can be used to generate and aggregate logs in a central location using a sub-offering from CloudWatch called CloudWatch Logs, which allows us to organize logs pertaining to all of your systems, applications, and AWS resources all in a single, highly scalable service. A diagram of how CloudWatch logs can be integrated with existing AWS infrastructure is illustrated below using the example of RDS databases:



Amazon CloudWatch Logs Streams is an extension of CloudWatch Logs, and can be seen as Kinesis for logs, allowing sequences of log events from the same source to be streamed in **near real-time** into other services such as S3 or Elasticsearch (Amazon OpenSearch), allowing us to stay up to date on the happenings of our resources quite conveniently, a particularly useful tool for cloud administrators who can often mitigate and solve infrastructure disasters immediately. In fact, we can use the CloudWatch API alongside CloudWatch Log streams to better manage our AWS environment, creating a lambda function to say, re-establish the connection between our RDS and EC2 instance, trigerring it whenever either faces an issue and has to restart.

CloudTrail

Since AWS environments are so large and complicated, it is often helpful for us to be able to track the actions that are undertaken by users, services and agents in relation to each other. AWS CloudTrail provides us with the ability to do just that, recording said actions as events within it. CloudTrail is always automatically enabled on every AWS account on the point of creation and helps us perform **operational and risk auditing, governance and compliance** operations on our AWS account.

The above description might make it sound like CloudTrail serves many of the same functions as CloudWatch but such beliefs are not to be heeded. This is because CloudTrail is best understood as a tracking tool, allowing us to keep a breadcrumb trail of all the minute actions that were performed within our AWS environment while AWS CloudWatch is mainly a monitoring tool, used to observe the current status of various AWS resources.

TLDR;

CloudTrail ⇒ Tracking

CloudWatch ⇒ Monitoring

Cost and Usage Report + Cost Explorer

Now, we have discussed a whole host of services and how they can be used to spin up complex architectures and create a variety of novel applications but it is also worth remembering that there is no such thing as a free lunch, and it is not very hard to imagine companies utilizing AWS racking up (sometimes unnecessarily) huge bills on cloud infrastructure. Well, AWS also graciously provides us with services for financial management, allowing us to better understand the costs associated with our infrastructure. In this section, we will be very briefly discussing two of those services: **Cost and Usage Report (CUR)** and **Cost Explorer**.

Both services are quite different in nature and serve both different use cases and potentially even different user bases. Cost and Usage Report provides us with the most comprehensive set of cost and usage data available for a user, organized as a detailed billing report. Said billing reports can be then published to a S3 bucket of our choosing.

Cost and Usage Report allows us to create reports that break down our costs by by the hour, day, or month, by product or product resource, or by tags that we define ourselves as a Comma-Separated Values (CSV) file, and can be viewed in any spreadsheet software of our choice.

AWS Cost Explorer on the other hand is a more technically limited service, in the sense that it cannot provide us with the same level of granular detail about the costs being incurred by the different resources used in AWS, has a 24 hour lag

when displaying monthly data, and cannot give us in-depth hourly data beyond the last 14 days, a capability that CUR can easily meet.

However, Cost Explorer is a more accessible service, relying on visualizations to present the cost and usage data in a more easily readable and understandable format usually in the form of graphs, charts and tables. Said visualizations can after all be understood by anyone with basic statistical literacy regardless of their technical familiarity with AWS, especially when forecasting and creating graphs of forecasted future costs, a feature made available to us by AWS Cost Explorer.

A table comparing the two services has been made available for the ease of the reader below:

	AWS Cost and Usage Report	**AWS Cost Explorer**
Description	A comprehensive, spreadsheet-like report of your billing data for a Payer AWS account	Highly visual cost graphs and tables showing a relatively high-level view of your costs and usage data for a specific Payer account in AWS
Unique functionality	Details historical cost and usage data and sends it to an Amazon S3 bucket for further analysis and longer retention	Not only offers historical records but also creates forecasts and savings recommendations
Data fields	Multiple line items. Also supports Cost Categories and Cost Allocation Tags	Up to 18 filters and groupings
Format	CSV and Parquet	CSV
Cost data duration	Hourly, daily, and monthly	Hourly (up to 14 days), daily, and monthly
Pricing	Free, but standard Amazon S3 charges apply	Free, although querying cost and usage data via the Cost Explorer API costs $0.01 per paginated request

AWS Config

As an AWS environment grows larger in size and starts integrating more and more components and services within itself, managing the different settings and configurations of all the different resources does become quite a bit of a hassle. In order to make said tasks easier, Amazon launched the AWS Config service.

AWS Config (short for AWS Configurations) is a service that allows us to assess, audit and evaluate the configurations (and configuration changes) of different AWS resources within our account, presenting the information to us in a relatively consistent format.

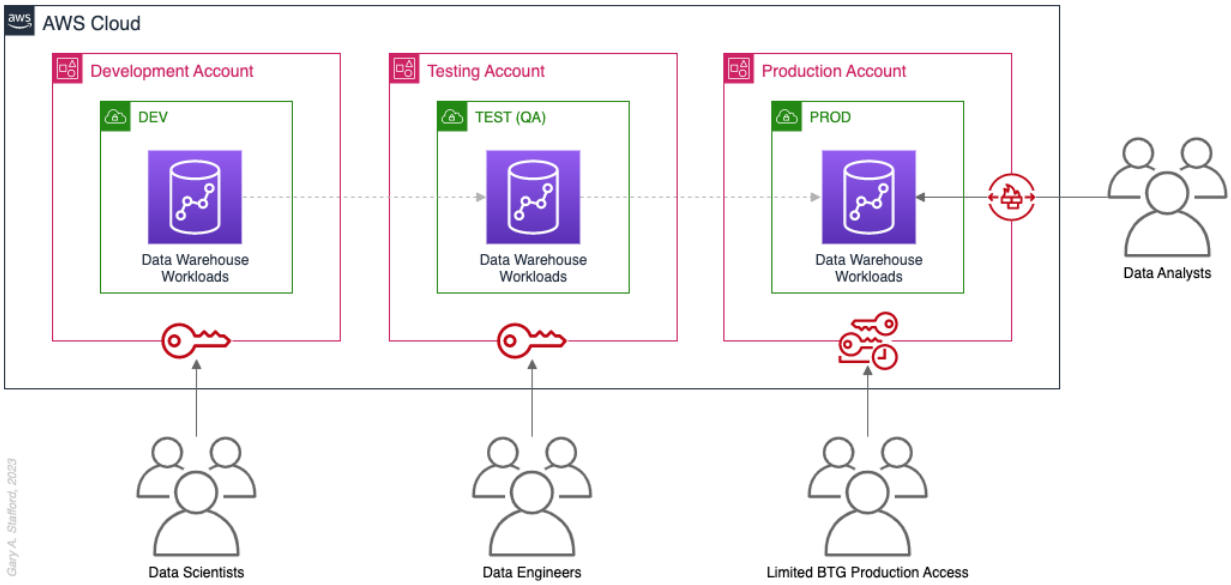
It also allows us to govern over our resources in other ways by providing us with ways to define rules to for example, detect improperly tagged resources or check if a certain setting has been enabled. In fact, the rules defined by us, (called AWS Config custom rules) are often used as invocation sources for Lambda functions, an example of which will be provided in a later section.

Also, AWS Config is automatically utilized by AWS Control Tower for its functioning, a service used when managing an AWS environment that has resources spread across multiple accounts. AWS Config Resources automatically created by AWS Control tower can be easily identified using the `aws-control-tower` tag alongside the `managed-by-control-tower` value. Certain sub-features of the AWS Config service also aid in maintaining the security of AWS resources and will be touched upon in a future section.

AWS Organizations and PrincipalOrg Id

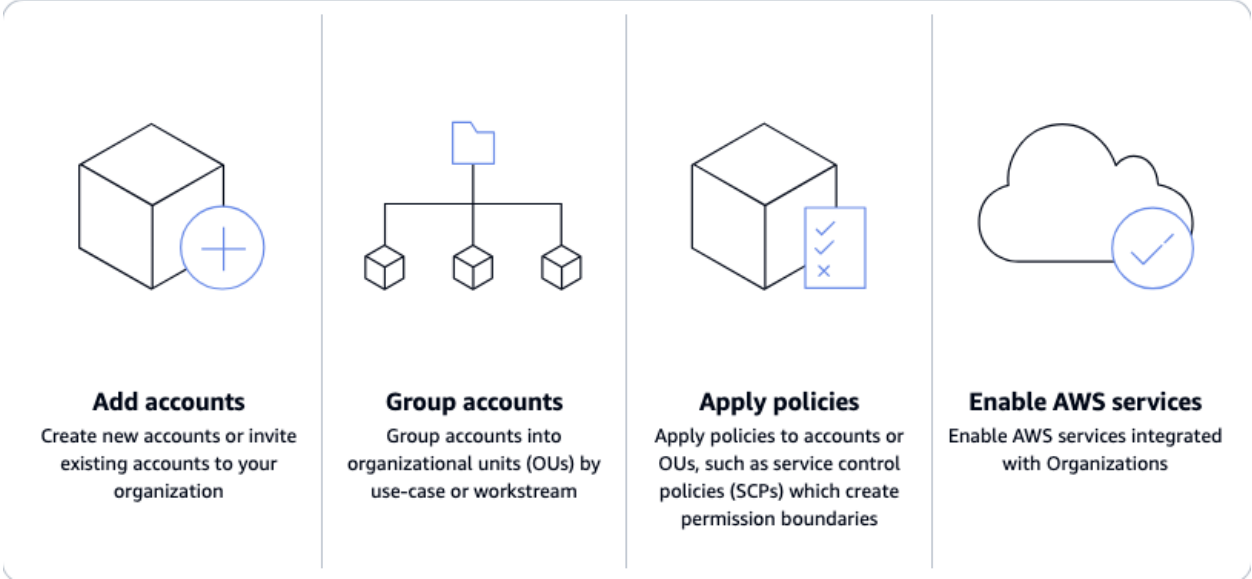
Now, multi-account AWS environments were mentioned previously once before but an explanation as to what they were and how they work was never explained. Multi-account AWS environments are, simply put, architectures that utilize multiple accounts to manage resources, users and workloads within an organization. This is often done for security and categorization reasons, allowing us to insulate workloads from one another and prevent unwanted users poking around in places where they do not belong, causing undesirable consequences.

An example of an environment which utilizes three different accounts for three different workloads, using separate accounts for development, testing and production, is given below:



Source: AWS

In order to more easily create and manage accounts that utilize a shared set of resources, AWS created a service called **AWS Organizations**. It works by grouping accounts into something called Organizational Units (or OUs), to which policies can be assigned/added to. These policies define the OUs permission boundary, i.e. which services, instances and other general AWS resources that they have access to, dictating what the accounts can and cannot do. An infographic presenting the mentioned capabilities of AWS Organizations is given below:



Source: AWS

Finally, OUs can help give some much needed structure to complex architectures and help conveniently locate, isolate and differentiate similar sets of resources from one another. This is because resources being governed by an OU can be quite easily identified by the presence of the Principal Org ID, whose value is used to uniquely identify and distinguish OUs from one another. In contrast, resources that do not belong to any OU have no such tag, making them easy to identify as well.

AWS Backup

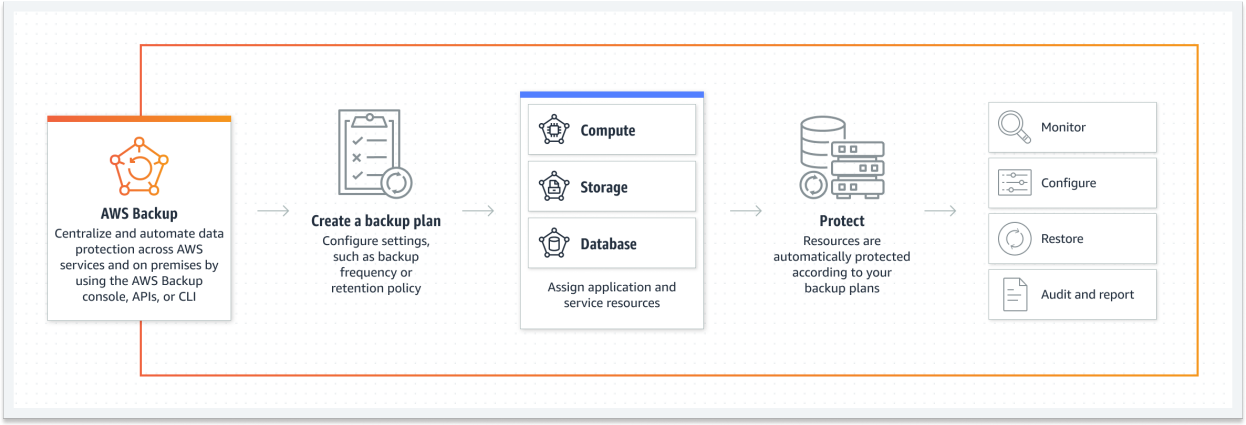
No business is completely immune from the loss of information. Breakdown of hardware, malware, hacking, unwanted intrusion and accidental deletion are just some of the ways in which data relating to business operations may be lost. Having backups of said data therefore may be a desirable thing for many businesses and is in fact, a necessity by law in many countries.

In the early days of cloud computing, companies had to manually script backup jobs, store data offsite, and frequently monitor backup operations, tasks that are prone to error. As cloud usage grew, these processes became an increasingly significant (and also often increasingly tedious) task. It was in this environment that AWS first launched **AWS Backup** in January 2019, offering an easy-to-use fully managed backup service that centralizes and automates the backup of data stored within a group of select AWS services which are listed below:

- **Amazon EC2** / **Amazon EBS**
- **Amazon S3**
- **Amazon RDS** (including all DB engines like **Aurora** and **DynamoDB**)
- **Amazon DocumentDB** / **Amazon Neptune**
- **Amazon EFS** / **Amazon FSx** (including Lustre and Windows File Server)
- **AWS Storage Gateway** (Volume Gateway)
- **Amazon Redshift**
- **Amazon Timestream**
- **AWS CloudFormation**
- **VMWare Cloud on AWS**

Note that AWS Backup is limited to the above set of services, and data stored in any other AWS offering must either be manually backed up or backed up using a third party tool.

AWS Backup also allows us to create data backup plans for the mentioned services, define backup policies for the data within them, automate backup scheduling, manage backup retention periods and store said backups across both multiple AWS regions and multiple AWS accounts. An infographic on how AWS Backup works is given below:



Source: AWS

The service definitely simplifies a myriad of data protection tasks and helps companies ensure the resilience of their data, meet regulatory compliances and perform recovery tasks without the need for custom scripting and/or manual intervention and has led to AWS Backup becoming the somewhat default option for backing up data related to AWS-specific resources. Implementations of the **Backup and Restore** method of disaster recovery mentioned in the Databases section for example, are often based on the AWS Backup service.

Systems Manager Patch Manager vs Run Command

AWS Systems Manager Run Command allows you to remotely execute commands on multiple EC2 instances, on-premises servers, and virtual machines (VMs) at scale. It simplifies operational tasks by eliminating the need for manual SSH or RDP access, making it ideal for automating software installations, configuration changes, and administrative tasks across your infrastructure.

Patch Manager, also part of AWS Systems Manager, automates the process of patching operating systems and applications across your instances. It ensures that your systems remain up-to-date with the latest security patches and updates, thereby enhancing security and compliance while reducing manual effort in managing patch deployments. Together, these tools streamline management tasks and improve operational efficiency within your AWS environment.

TLDR;

Systems Manager Patch Manager is used to perform OS level patches on the EC2 instances.

On the other hand, System manager run command is used to run third party commands and software updates and the like.

CloudFormation and Creation Policy

We have talked so far about the different types of resources and architectures in AWS, but we have not really delved deeply into the many novel ways in which our cloud infrastructure can be created and managed. Two ways in which resources can be created, configured and managed within AWS are the AWS Management Console and AWS CLI, both of which have been discussed in an earlier background section. However, there is another method: **CloudFormation**.

CloudFormation is an **IaC(Infrastructure as Code)** service which allows us to automate the provisioning of AWS resources using code. This is done by writing a **CloudFormation Template**, a single JSON file that describes all the AWS resources that need to be created and the relations between them, non-technical readers may imagine this file as a sort of code version of the architecture diagrams that have been sprinkled throughout the book. Said templates can then be fed into CloudFormation, spinning up and configuring the described infrastructure on our behalf, saving us the hassle of individually creating and configuring AWS resources and figure out what's dependent on what, immensely simplifying the process of deploying and managing infrastructure on AWS.

CloudFormation also has an attribute called **CreationPolicy** which can be used to setup certain third party software and ensure certain components are running before the resources mentioned in the template are spun up. This is because the resources utilizing the CreationPolicy attribute may be forcefully restrained from reaching the ***"Completed"*** stage until AWS CloudFormation verifies that certain conditions such as a specified number of success signals or timeout period has been met.