AWS Cloud Fundamentals

Develop Intelligence

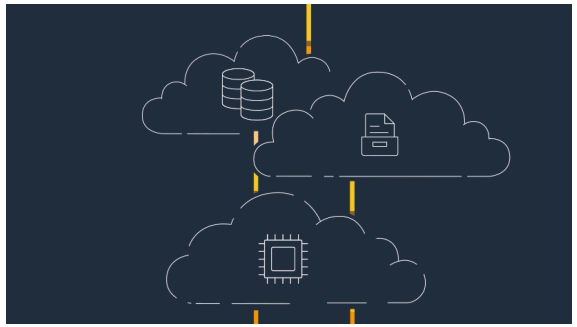
Kerry Nakayama

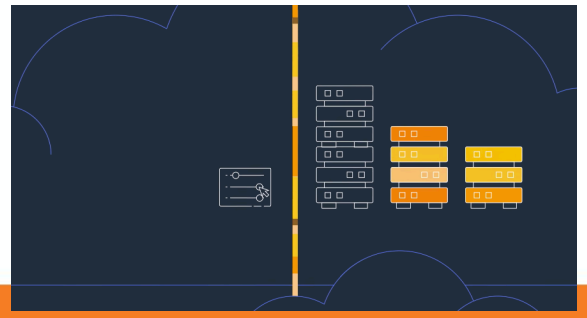
Objectives

- Cloud computing and AWS
  - Microservices
  - Regions
  - Users, Roles, & Groups
  - Virtual machines
- Networking VPC
  - Provisioners
- Server vs Serverless
  - Traditional vs cloud warehouse
- Types of Databases
  - SQL Solutions
- Storage
  - Cloud storage
  - Systems
- ELT
  - Managing infrastructure
- Snowflake infrastructure
  - Shared Nothing
  - MPP
  - Elasticity
  - Automation

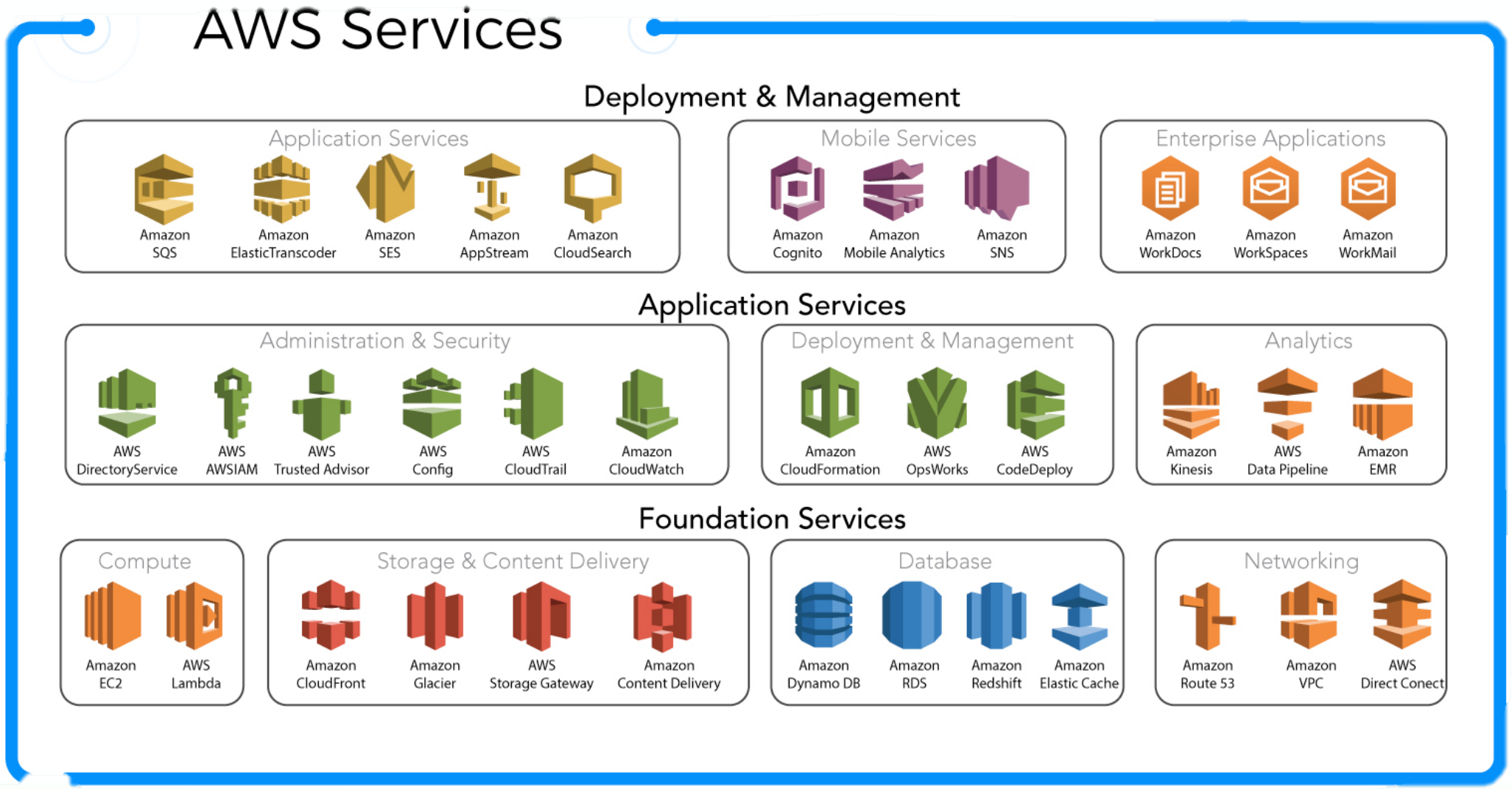
What is Cloud Computing

- Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing
  - No need for physical data centers or servers
  - Access on an as-needed basis





AWS Services



AWS Regions

- AWS serves over a million active customers in more than 190 countries
- An AWS Region is a physical location in the world where AWS has multiple Availability Zones
- AWS provides you with the flexibility to place instances and store data within multiple geographic regions

Users, Roles, Groups

- **USERS:** This is the entity that you create in the IAM management console that represents an individual

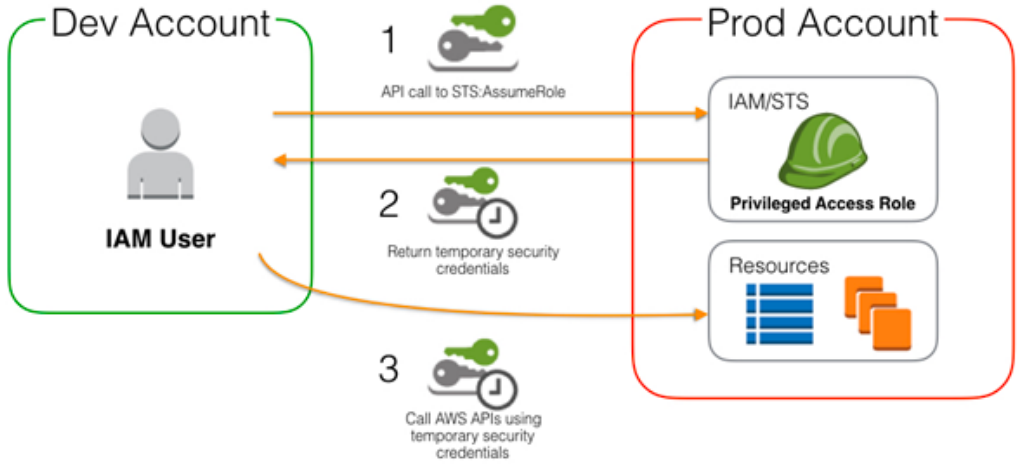
- **GROUPS:** This is a collection of IAM USERS.
  - You would use GROUPS to grant/demove permissions for multiple users at the same time (think “HR” or “DEVS” or “ACCOUNTING”)

- **ROLES:** Roles are a set of permissions that can be assumed by **USERS** or **GROUPS** (of users) that give access to AWS functions.

- Credentials can be temporary

IAM User and Roles

- You can use AWS Identity and Access Management (IAM) roles to grant access to databases among other items
- An IAM *role* is an IAM identity that you can create in your account that has specific permissions but can be assigned to multiple people
- You generally have two ways to use a role: interactively in the IAM console, or programmatically with the AWS CLI

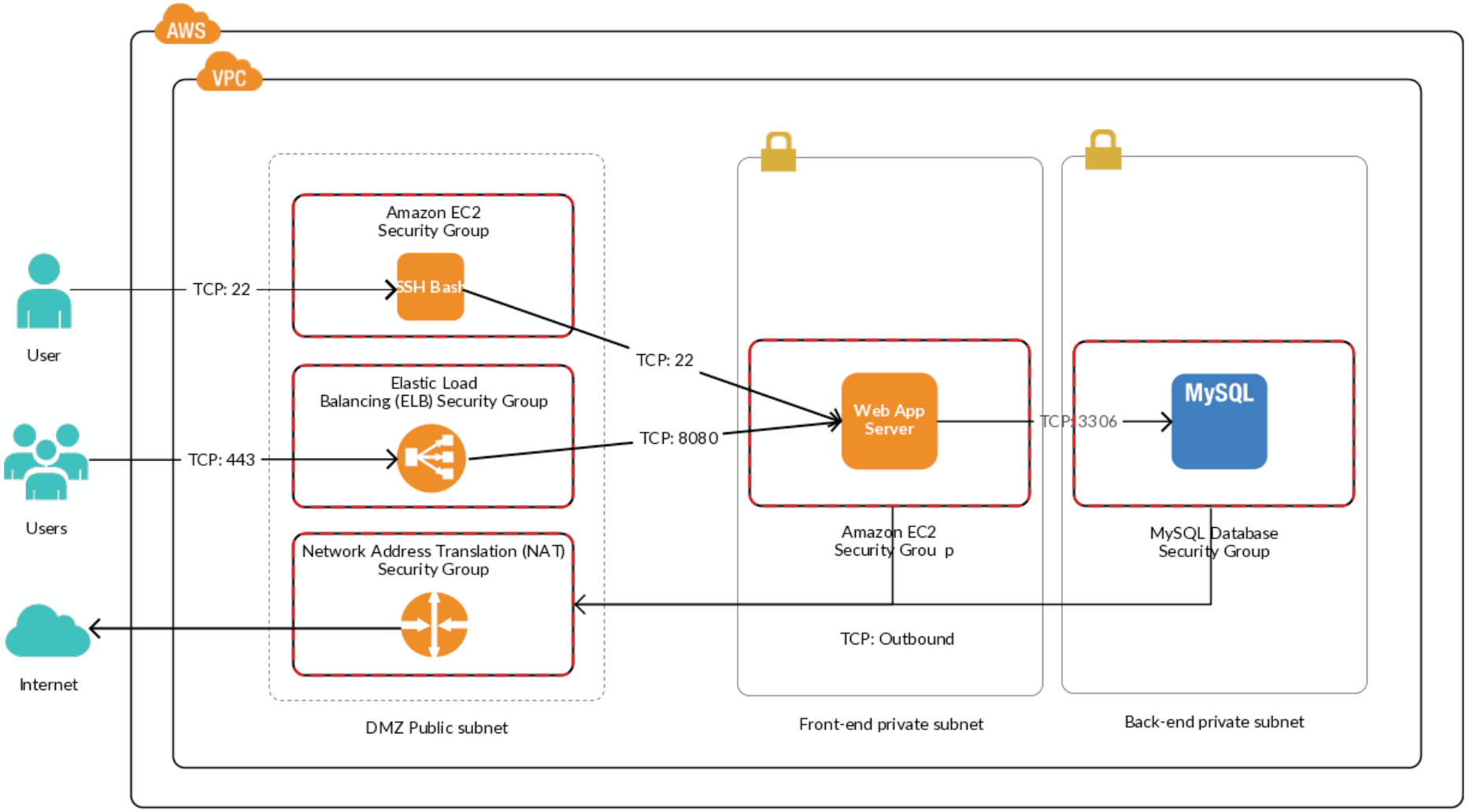


- complex uses of roles, such as granting access to applications and services, or federated external users, you can call the AssumeRole API

Creating and Managing Virtual Machines with EC2

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud
  - EC2 eliminates your need to invest in hardware up front
  - Launches *instances* which are virtual computing environments
  - Storage volumes for temp data that's deleted when you terminate your instance
    - known as instance store volumes
  - Persistent storage volumes for data using Amazon Elastic Block Store (Amazon EBS)
  - Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs)

Networking (Virtual Private Cloud)



Reference: https://cdn.amazonblogs.com/security_awsblog

Networking (Virtual Private Cloud)

- When creating any sort of architecture you'll probably want various elements to be able to talk to each other within a network.
- INTRAnet versus INTERnet (as an example)
- With cloud architecture it's not uncommon to deploy all elements of a single application into a single network to allow each element to talk to each other.
- You can open up the network to the outside world using ports; so you might allow internet traffic into your cloud through port 443 and port 80 BUT…
- You would prevent access onto 3306 (port for mysql). NOTHING talks to the Database from outside the cloud!

Provisioners

- Old, ancient system:
  - Single ”production” server that handled all traffic
  - Physically existed on site
  - Physically had to add memory
- Newer system:
  - Virtual machines to handle traffic
  - Networking multiple machines together
  - If virtual machine in network breaks, remove it, repair it, return it to network
- Newest system:
  - Create Virtual machines “as needed”
  - ALL MACHINES MUST BE THE SAME THOUGH (or else responses could change)

Server vs Serverless Architecture

Server vs Serverless Architecture

Develop Intelligence

Server	Serverless
Predictive tasks: For tasks that use constant or predictive compute, it may be more cost-effective to use server-based billing	**Server management:** Because there is no backend infrastructure to be responsible for, liability is reduced and there is no system administration
Testing and debugging: Debugging is less complicated because there is visibility into backend processes and the application is not broken up into separate, smaller functions	**Scalability:** With a serverless architecture, you don't have to think twice about provisioning infrastructure because of its ability to automatically scale with traffic volumes
Overall control: The company owns and manages the infrastructure, which provides full control over all aspects of the application	**Application flexibility:** You can migrate individual application features or partial workloads to run on serverless as on-demand events
Compliance and security: Having full control over the infrastructure allows for full visibility, which may be required for compliance and security standards	**Cost:** The cost of hiring backend infrastructure engineers goes down, along with operational costs
Legacy applications: Existing applications may not have the flexibility of decoupling individual parts and may be better suited to migrate to a server-based architecture	**Startup friendly:** The serverless architecture pay-as-you-go model allows you to build an environment nearly for free and ease into the market without dealing with huge bills for minimum traffic

Traditional vs Cloud Data Warehousing

- The biggest difference between Traditional and Cloud Data Warehousing has to do with the fundamental impermanence of the infrastructure.
- With traditional data warehousing you create, for example, a MYSQL database that must remain in pristine condition
  - We need to make sure that data going in does not crash the system
  - All data needs to be permanent and cleaned
- In cloud infrastructure you can destroy and re-create databases at will (again- think of wizards provisioning things out of nowhere)
- Instead of bug hunting- in cloud infrastructure we just destroy and recreate

Type of Databases

Relational

Relational databases store data with a defined schema. These databases are commonly used for transactional and traditional applications.

Key-value

Key-value databases are optimized to store and retrieve key-value pairs in large volumes and in milliseconds, without the performance overhead and scale limitations of relational databases.

Document

Document databases are designed to store data as documents. Data is typically represented as a readable document.

Graph

Graph databases are used for applications that need to enable users to query and navigate relationships between highly connected datasets.

In-memory

In-memory databases are used for read-heavy and compute-intensive applications that require low-latency access to data.

SQL Solutions

Develop Intelligence

Live data

Amazon Athena

Symmetric Multi-processing

Microsoft SQL Server

ORACLE

Massive Parallel Processing

Amazon Redshift

snowflake

Google BigQuery

Storage

Cloud storage is the fundamental building block of cloud architecture

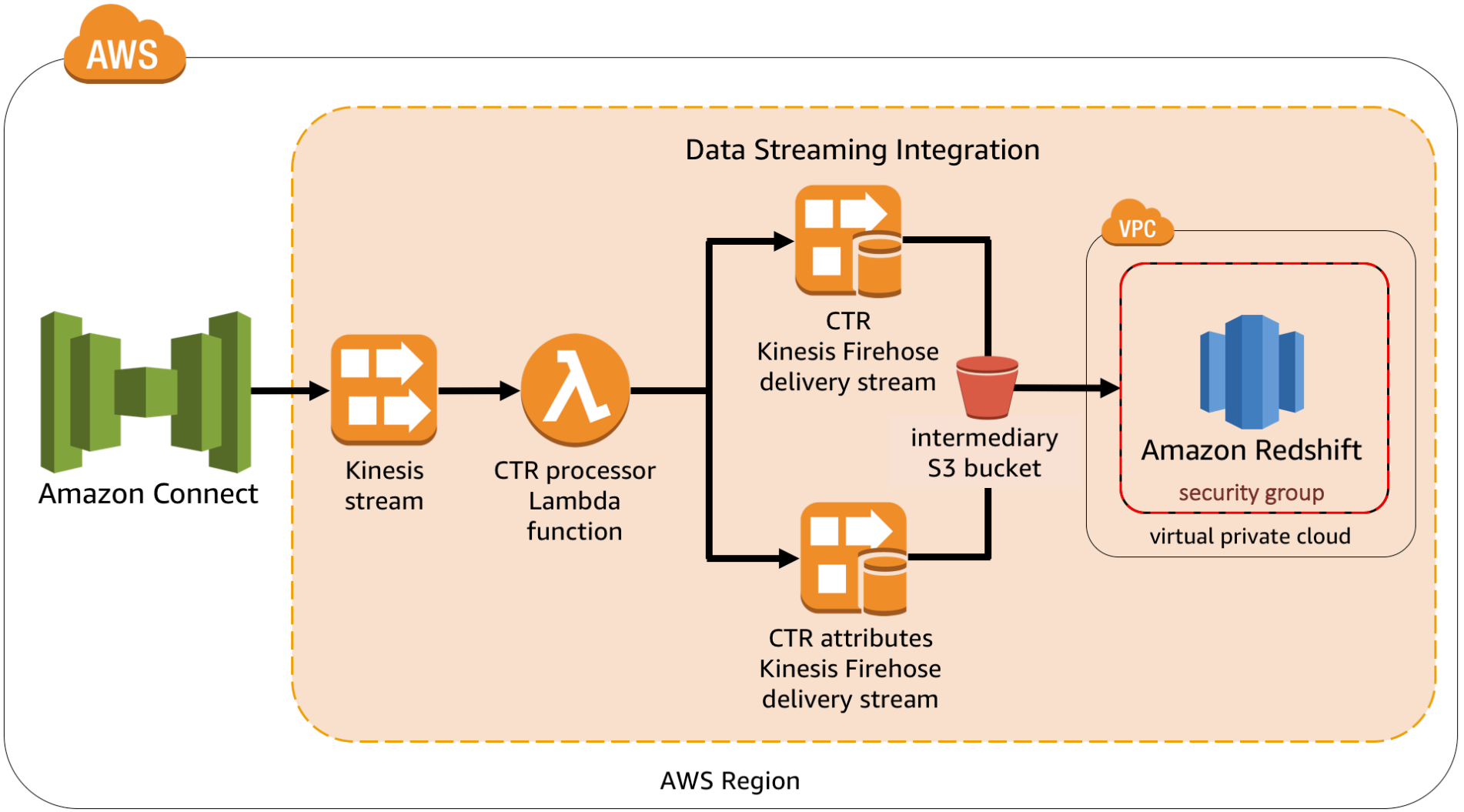
- S3 bucket
  - Directories that exist in the cloud
  - "/Documents" or "/Downloads" folders
  - Hold huge amounts of data
  - Safe, Reliable, Dependable way to store data
- Data architecture
  - S3 is a fundamental component
  - Different types if S3 buckets
  - Plays its part for data retention

Moving Data in the Cloud

Homogeneous migrations, where you migrate between same database engines, may require the use of native database tools to migrate database elements

Heterogeneous migrations, where you migrate between different database engines, require the use of the Schema Conversion Tool to first translate your database schema to the new platform

Cloud Storage System

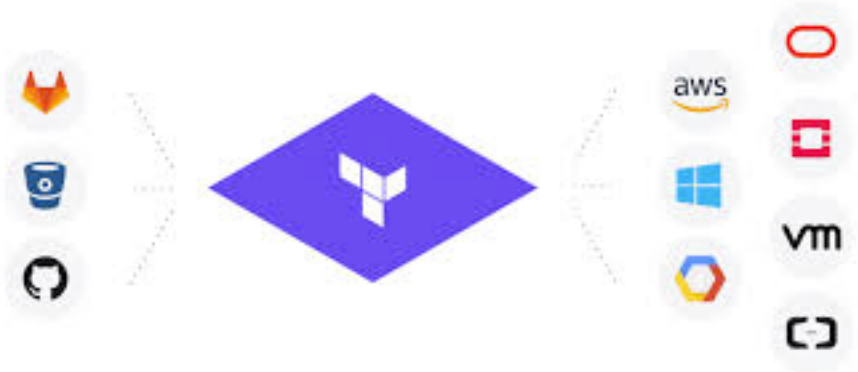


Transforming Data

- In any ETL process we'll want to transform data en route to the database.
- As data is flowing through the ETL AWS has several options for transformations, including:
  - Lambdas:
    - Every time you get a piece of data into your AWS system a LAMBDA will run a single, small piece of code that will transform the structure of that data before passing it on to the next "step" in your ETL process
    - Lambdas do NOT need to live on servers to run- AWS runs them virtually. You do not need to worry about a server.
  - IN-STREAM transformations:
    - With Streaming (read- EXTREMELY FAST MOVING) data AWS can actually have the "pipeline" that the data is going through do some transformation before sending it along to the next step in the process

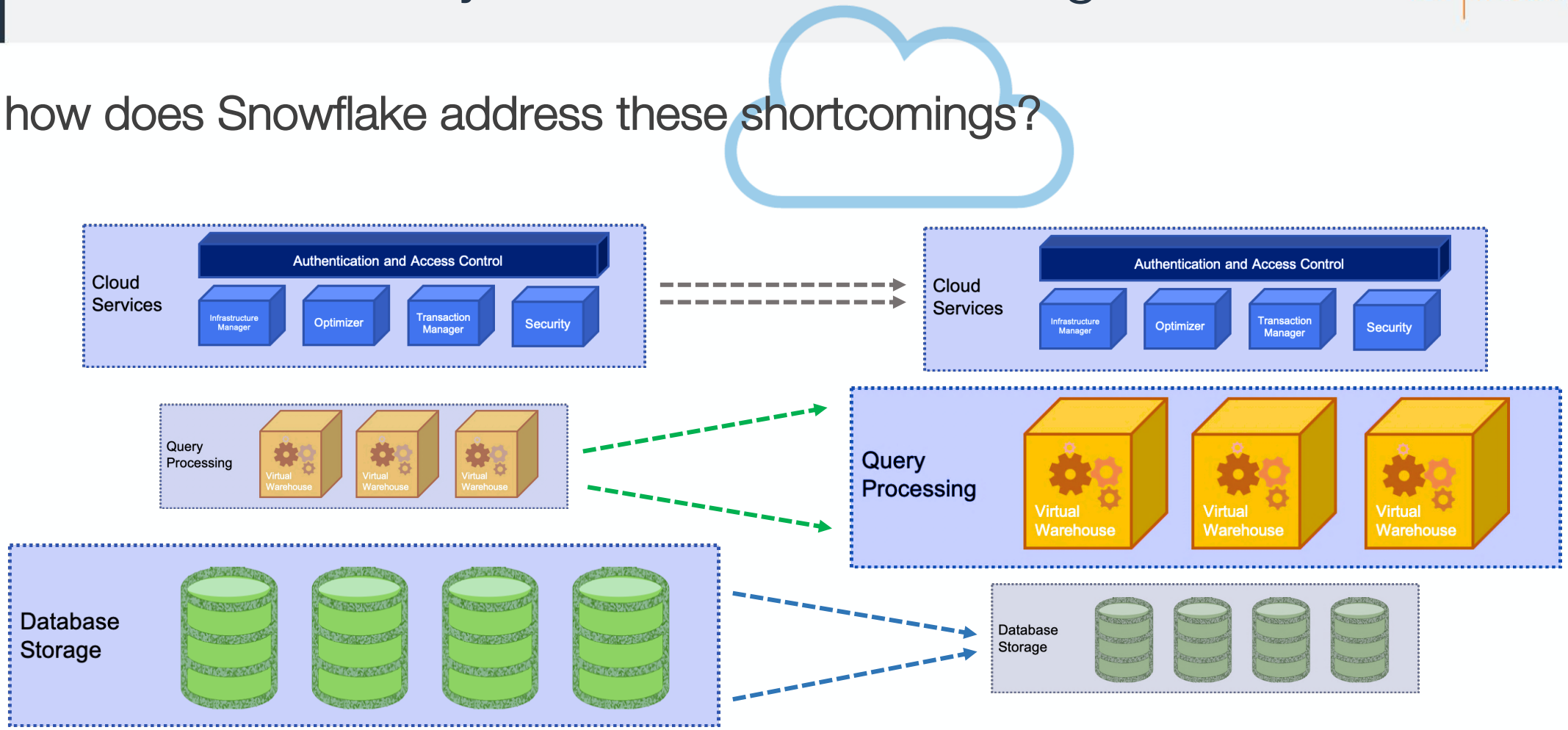
Managing Infrastructure (Terraform)

- Terraform is basically a way of putting your entire infrastructure into CODE form and making all changes and deployments through that code.
- Want a new Virtual Private Cloud? Write it in Terraform
- Want a new Lambda? Write it in Terraform
- Terraform is a provisioner that works for Azure, GCloud AND AWS and ensures that your infrastructure can be re-deployed quickly in case of emergency!



Snowflake is a hybrid of Shared Nothing and Disk

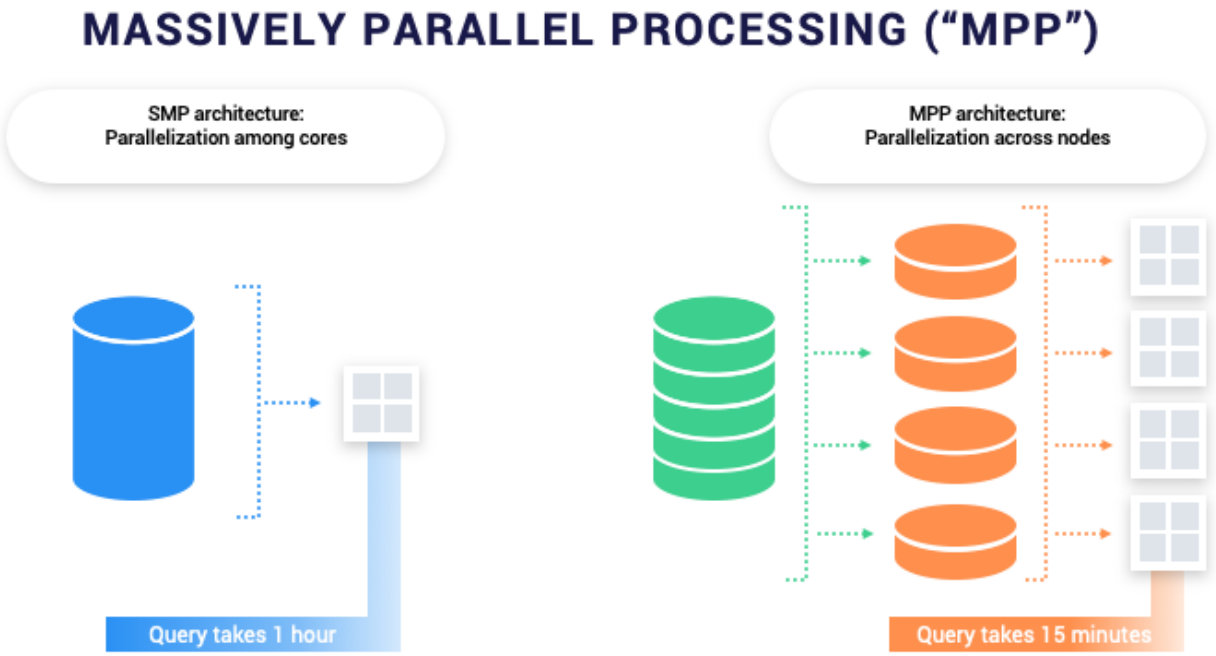
So how does Snowflake address these shortcomings?



Snowflake is Elastic!

Massive Parallel Processing

- Snowflake *is* a *shared-nothing* or *massive parallel processing* system.
  - Imagine if you will, for a minute, a series of workers, each of whom have their own memory, resources, and processors assigned to a task.
  - We can split the task among these workers thereby speeding up the task immeasurably.



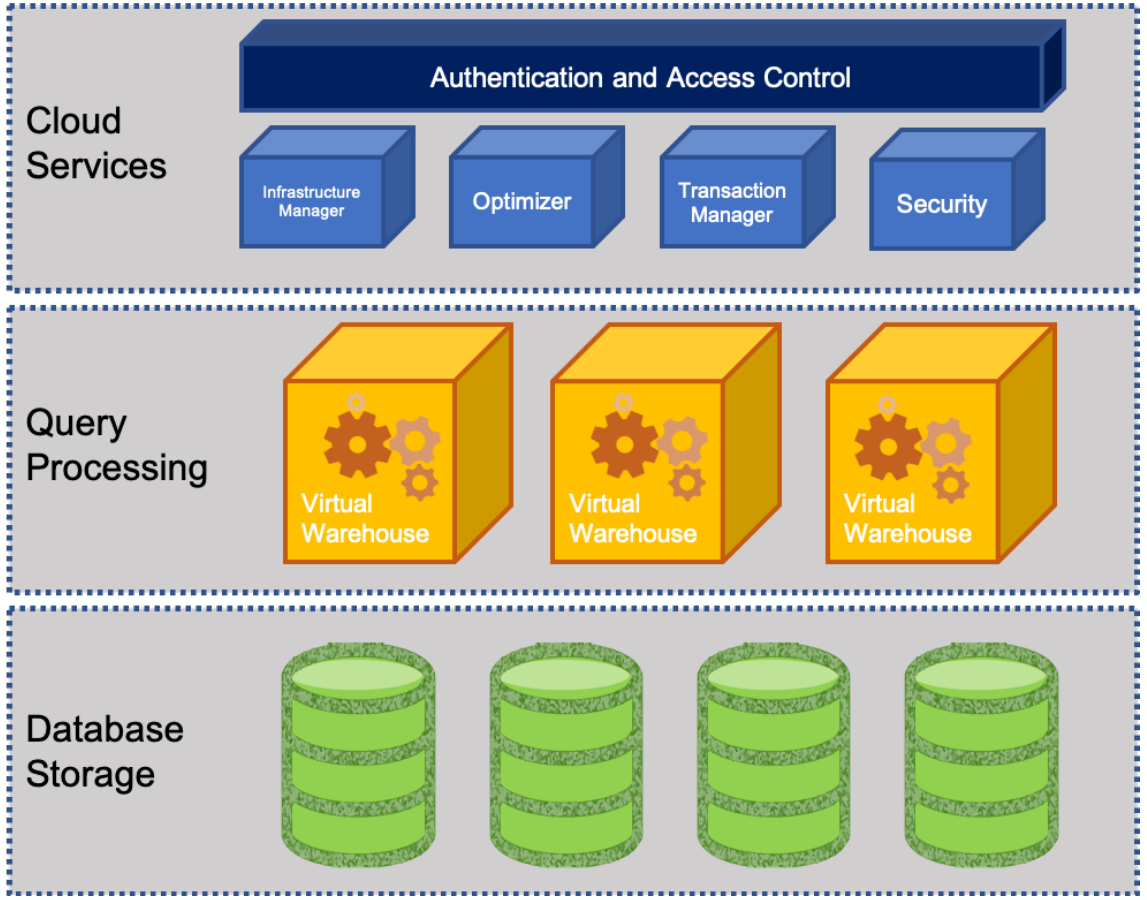
"MPP is the coordinated processing of a program by multiple processors that work on different parts of the program, with each processor using its own operating system and memory"

What does Snowflake offer?

Let's divide data warehouses into **three parts:**

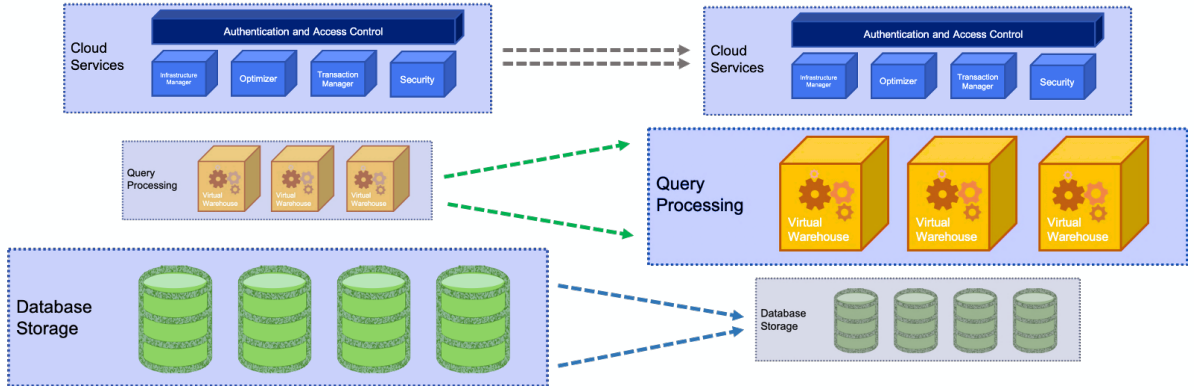
- **Storage:** for storing data and
- **Computing:** which is what allows us to do all the things we do in structured SQL databases
- **Services:** Which are some of the things like pipelines, etc.

Snowflake offers us the opportunity to independently scale the storage, services, and the computing. Why is this important?

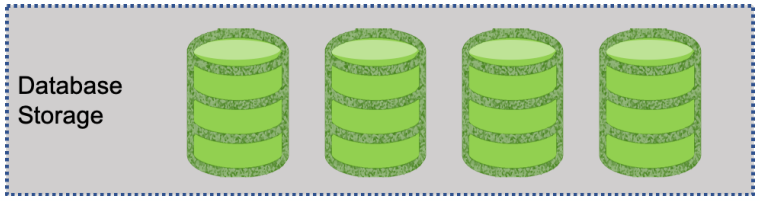


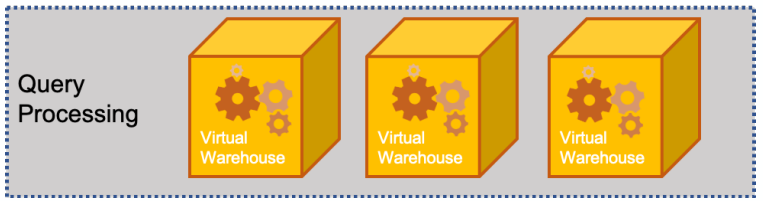
Snowflake Elasticity

- Snowflake has a novel concept to deal with elasticity:
  - This means that we can deal with the DATA STORAGE, COMPUTE and USERS (Services) layer without effecting the other ones



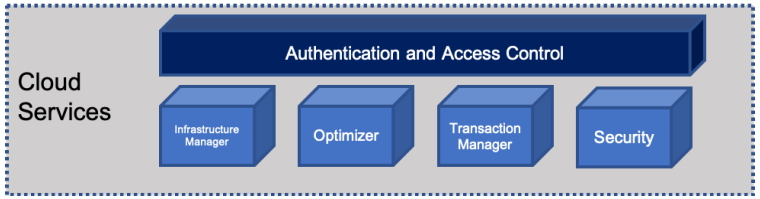
DATA





Compute

Services



Snowflake Automation

- **Data distribution:**
  - Managed automatically by Snowflake based on usage.
  - Snowflake automatically manages how data is distributed in the virtual warehouse
    - Other systems rely on a static partitioning scheme based on a distribution algorithm or key chosen by the user at load time
  - Data is automatically redistributed based on usage to minimize data shuffling and maximize performance.
- **Loading Data:**
  - Complex ETL data pipelines are no longer needed to prepare data for loading. Snowflake natively supports and optimizes diverse data, both structured and semi-structured, while making that data accessible via SQL.

Confused? ASK QUESTIONS

Break & Lab2