Distributed Systems
CS6421
Cloud Application Architectures

Prof. Roozbeh Haghnazar

Slides Credit:

Prof. Tim Wood and Prof. Roozbeh Haghnazar

Includes material adapted from Van Steen and Tanenbaum's Distributed Systems book

Last Time…

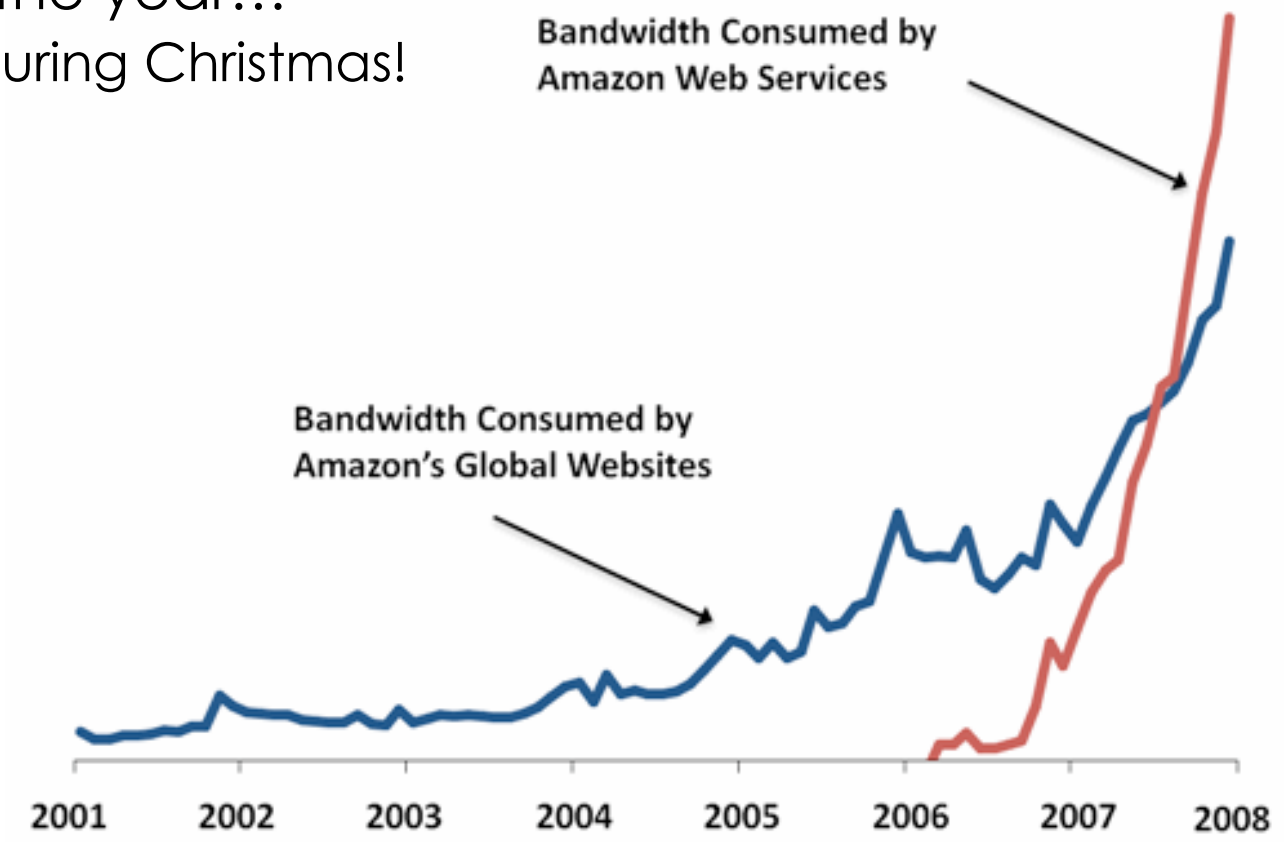
- Performance in Dist. Systems
  - Introduction
  - Performance metrics
  - Models

This Time…

- Cloud Application Architectures
  - IaaS vs PaaS
  - Multi-Tier
  - Microservices
  - Serverless

Amazon's Cloud

- Amazon built its cloud platform so that other people could pay for its infrastructure during the rest of the year…
  - Only needed peak capacity during Christmas!
- Now its cloud users are far bigger than its own sites



Types of Cloud Services

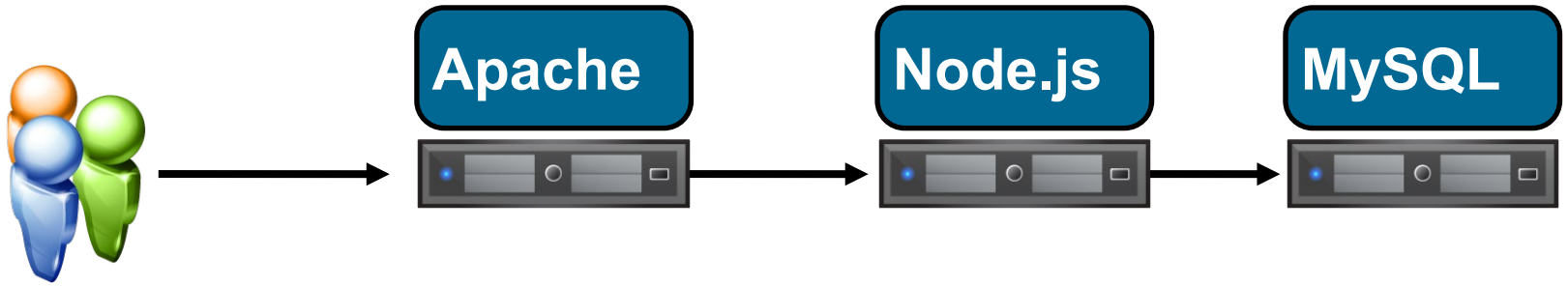
- Infrastructure as a Service (**IaaS**)
  - Rent VMs, Containers, physical servers, disks, etc. by the hour
  - Examples: EC2, EBS, S3
  - Benefits?
  - Limitations?
- Other options: Function as a Service, Software as a Service, Resource as a Service…
- Platform as a Service (**PaaS**)
  - Cloud provides a software layer on top of its resources
  - Exposes a programming API for users to develop cloud-based apps
  - Cloud provider manages all underlying resources (autoscaling)
  - Examples: Beanstalk, Lambda, EMR
  - Benefits?
  - Limitations?

Prof. Tim Wood & Prof. Roozbeh Haghnazar

Multi-Tier Applications

Multi-Tier Web Applications

- Traditionally composed of 3 components
- Separation of duties:
  - **Front-end web server** for static content (Apache, lighttpd, nginx)
  - **Application (API) tier** for dynamic logic (PHP, Tomcat, node.js)
  - **Database back-end** holds state (MySQL, MongoDB, Postgres)
- Why divide up in this way?



STATEFUL VS STATELESS

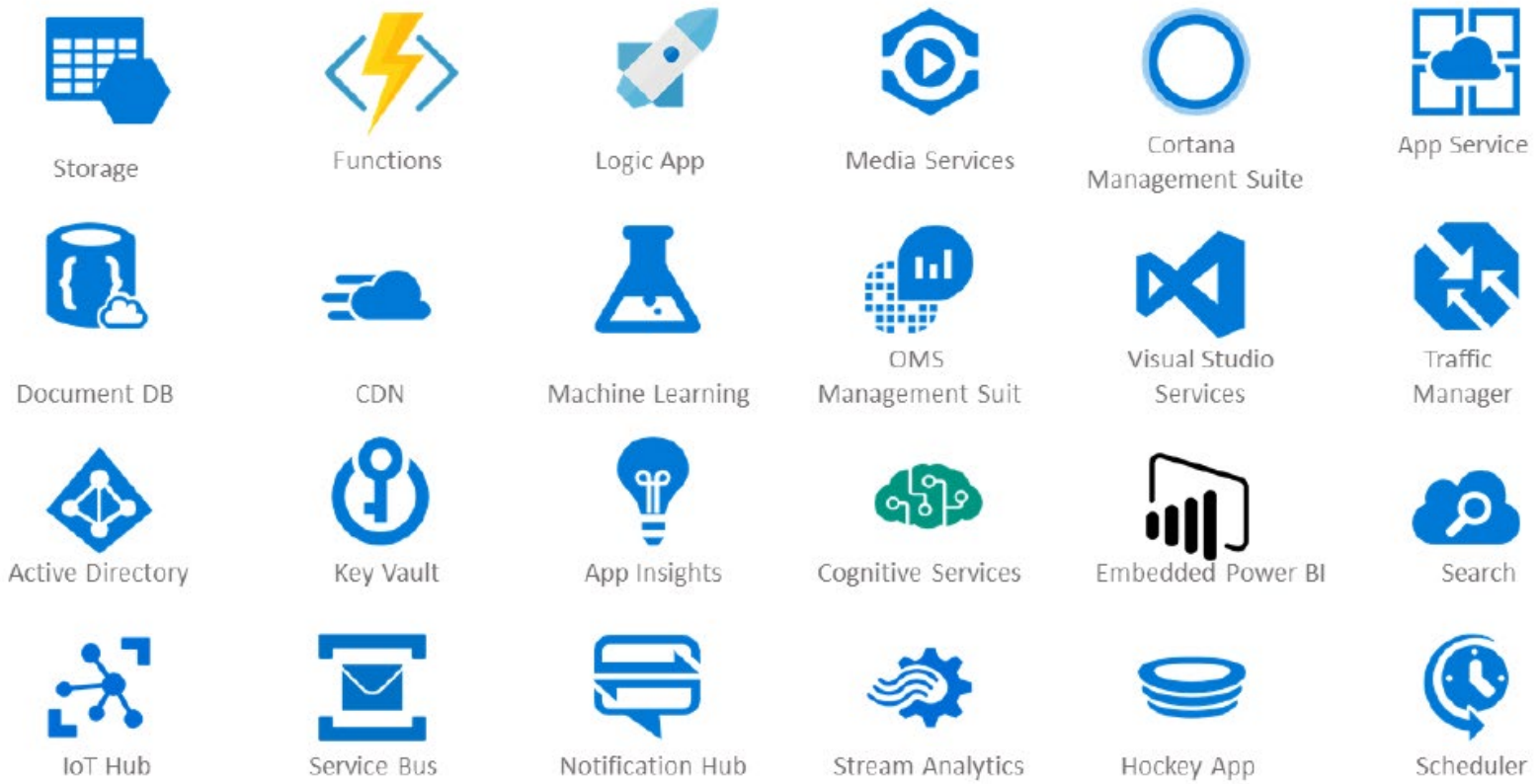
- The multi-tier architecture is based largely around whether a tier needs to worry about state
- Front-end - totally **stateless**
  - There is no data that must be maintained by the server to handle subsequent requests
- Application tier - maintains **per-connection state**
  - There is some temporary data related to each user, e.g., my shopping cart
  - May not be critical for reliability - might just store in memory
- Database tier - global state
  - Maintains the global data that application tier might need
  - Persists state and ensures it is consistent

USAGE EXAMPLES OF SERVERLESS

- Web Application (Weather stations)

- Chatbots (Event-Driven Architecture for chatbot)

Serverless in Azure



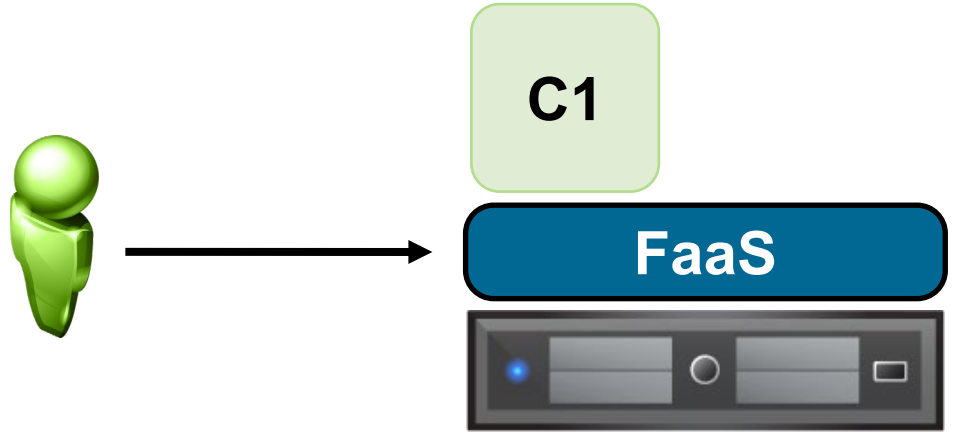
SERVERLESS STARTUP

- Function as a Service platfoms (Lambda, GCF, Azure Functions, etc)
  - Define a stateless "function" to execute for each request
  - A container will be instantiated to handle the first request
  - The same container will be used until it times out or is killed

No workload means no resources being used!

SERVERLESS STARTUP

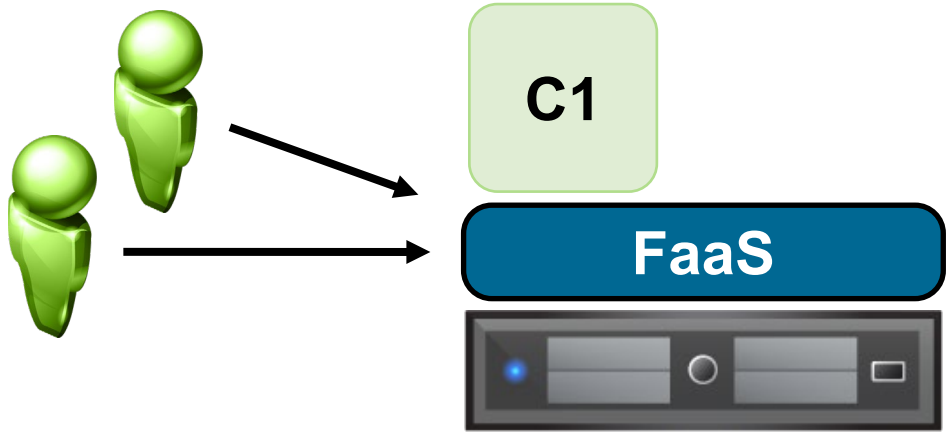
- Function as a Service platfoms (Lambda, GCF, Azure Functions, etc)
  - Define a stateless "function" to execute for each request
  - A container will be instantiated to handle the first request
  - The same container will be used until it times out or is killed



Request arrives, start green container

SERVERLESS STARTUP

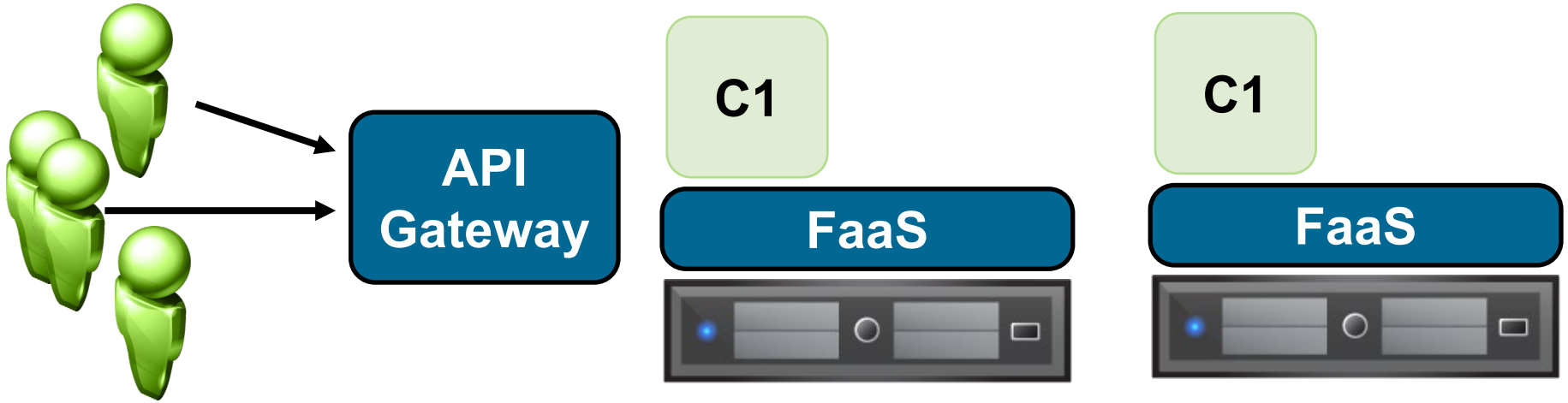
- Function as a Service platfoms (Lambda, GCF, Azure Functions, etc)
  - Define a stateless "function" to execute for each request
  - A container will be instantiated to handle the first request
  - The same container will be used until it times out or is killed



Reuse that container for subsequent requests

SERVERLESS STARTUP

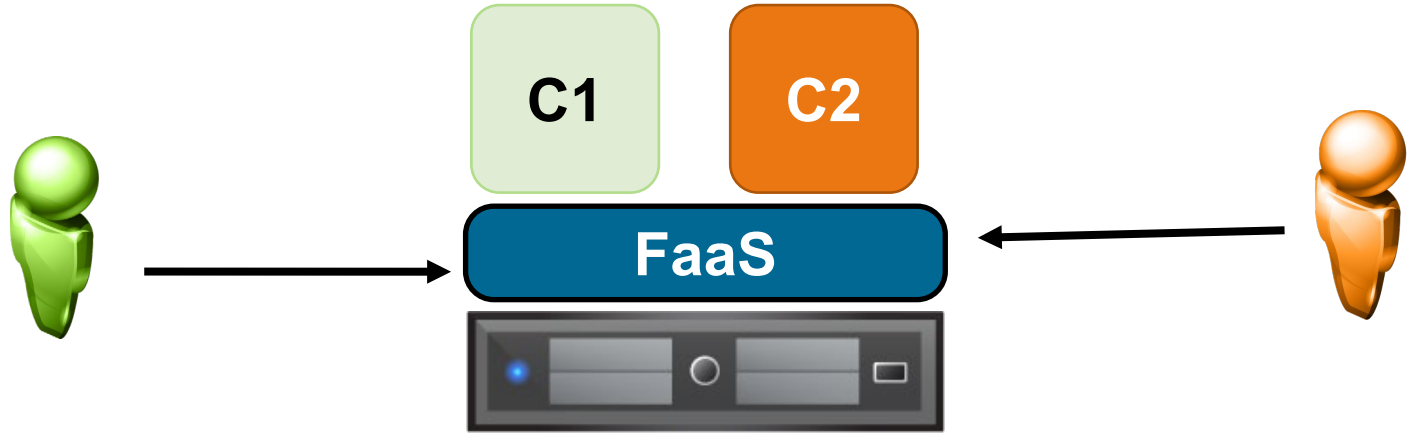
- Function as a Service platfoms (Lambda, GCF, Azure Functions, etc)
  - Define a stateless "function" to execute for each request
  - A container will be instantiated to handle the first request
  - The same container will be used until it times out or is killed



Add more replicas if workload exceeds capacity

SERVERLESS STARTUP

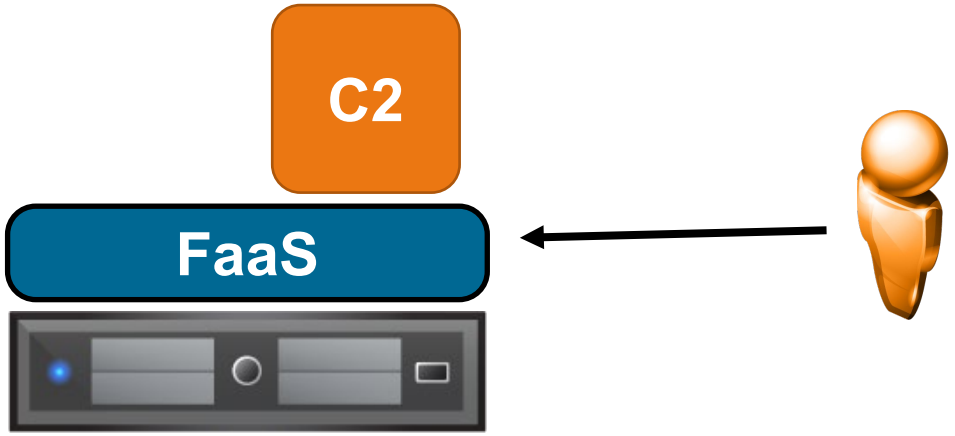
- Function as a Service platfoms (Lambda, GCF, Azure Functions, etc)
  - Define a stateless "function" to execute for each request
  - A container will be instantiated to handle the first request
  - The same container will be used until it times out or is killed



Start new container if user needs a different function

SERVERLESS STARTUP

- Function as a Service platfoms (Lambda, GCF, Azure Functions, etc)
  - Define a stateless "function" to execute for each request
  - A container will be instantiated to handle the first request
  - The same container will be used until it times out or is killed



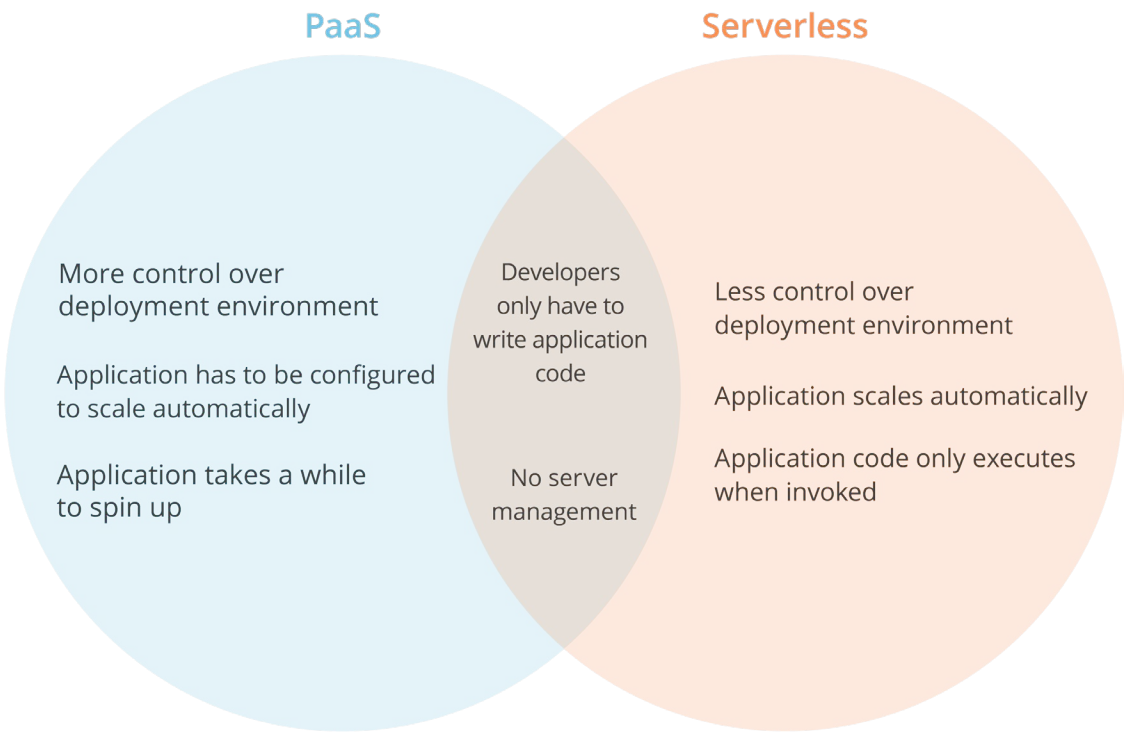
Stop old containers once not in use

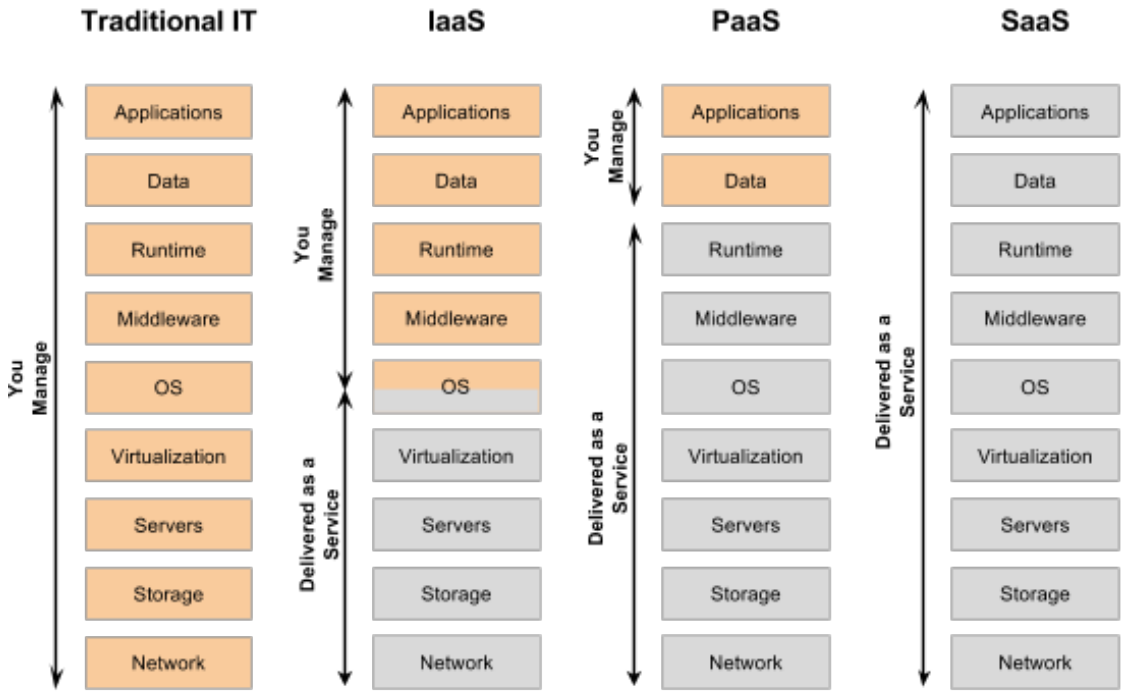
SERVERLESS PROS/CONS

- Benefits:
  - Simple for developer when auto scaling up
  - Pay for exactly what we use (at second granularity)
  - Efficient use of resources (auto scale up and down based on requests)
  - don't worry about reliability/server management at all
- Drawbacks:
  - Limited functionality (stateless, limited programming model)
  - High latency for first request to each container (**10x slower and 5x resource cost**)
  - Some container layer overheads plus the lambda gateway and routing overheads
  - Potentially higher and unpredictable costs
  - Difficult to debug / monitor behavior
  - Security

Auto scaling up
--
Stateless

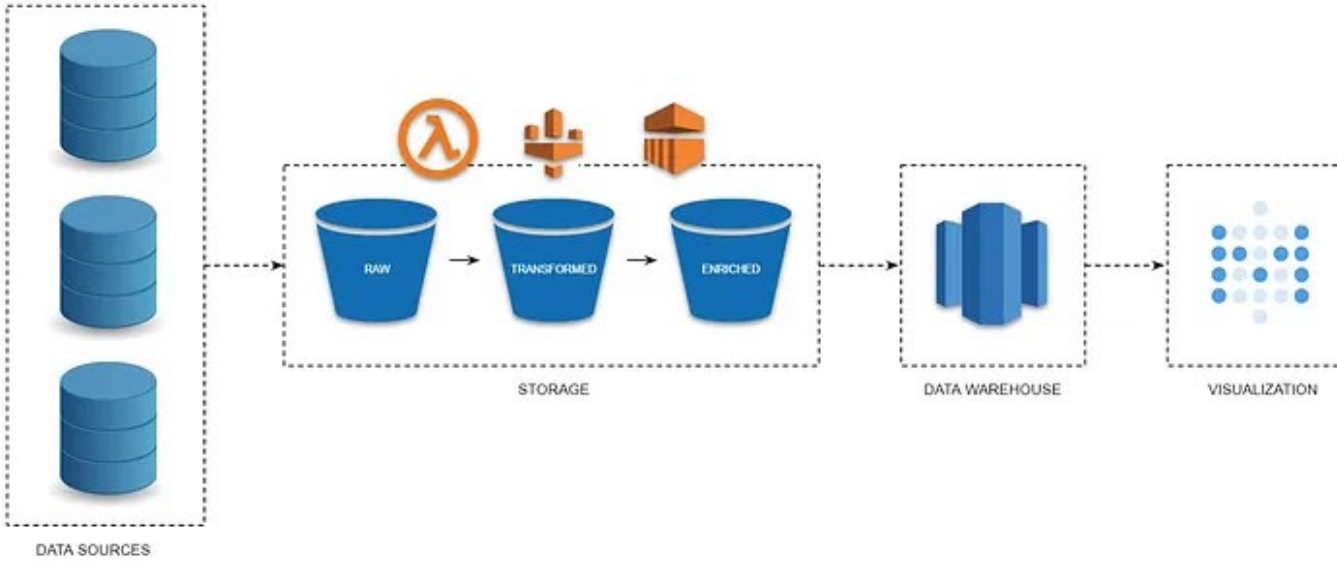
SERVERLESS VS PAAS?



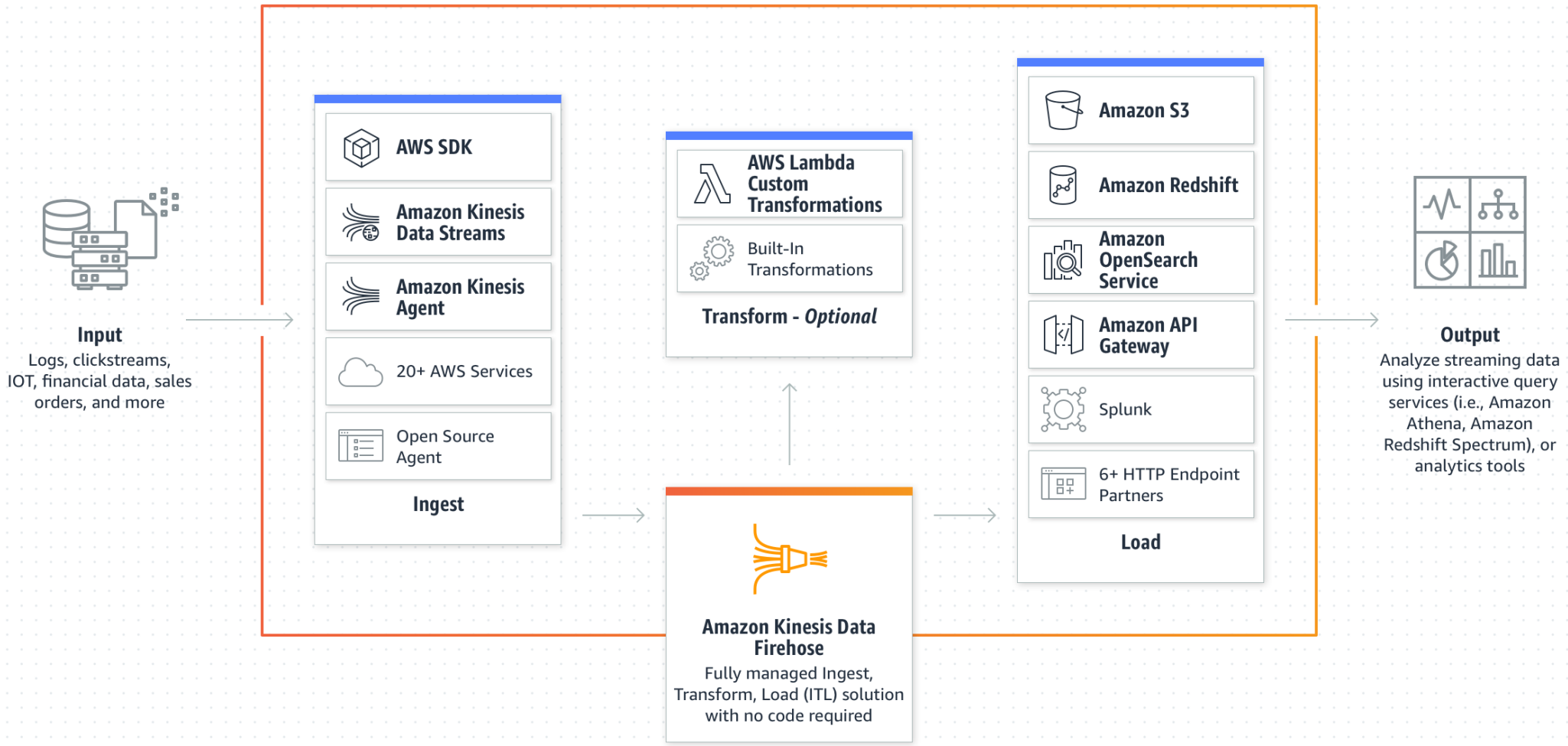


AWS BIGDATA PIPELINE

- **Data Nodes -** The section of input data for a task or the location where output data is to be collected.
- **Activities -** A description of work to perform on a program using a computational means and typically input and output data nodes.
- **Preconditions -** A conditional statement that must be true before action can run.
- **Scheduling Pipelines -** Marks the timing of a planned event, such as when an action runs.
- **Resources -** The computational resource that performs the work that a pipeline defines.
- **Actions -** An action that is triggered when specified conditions are met, such as the failure of an activity.



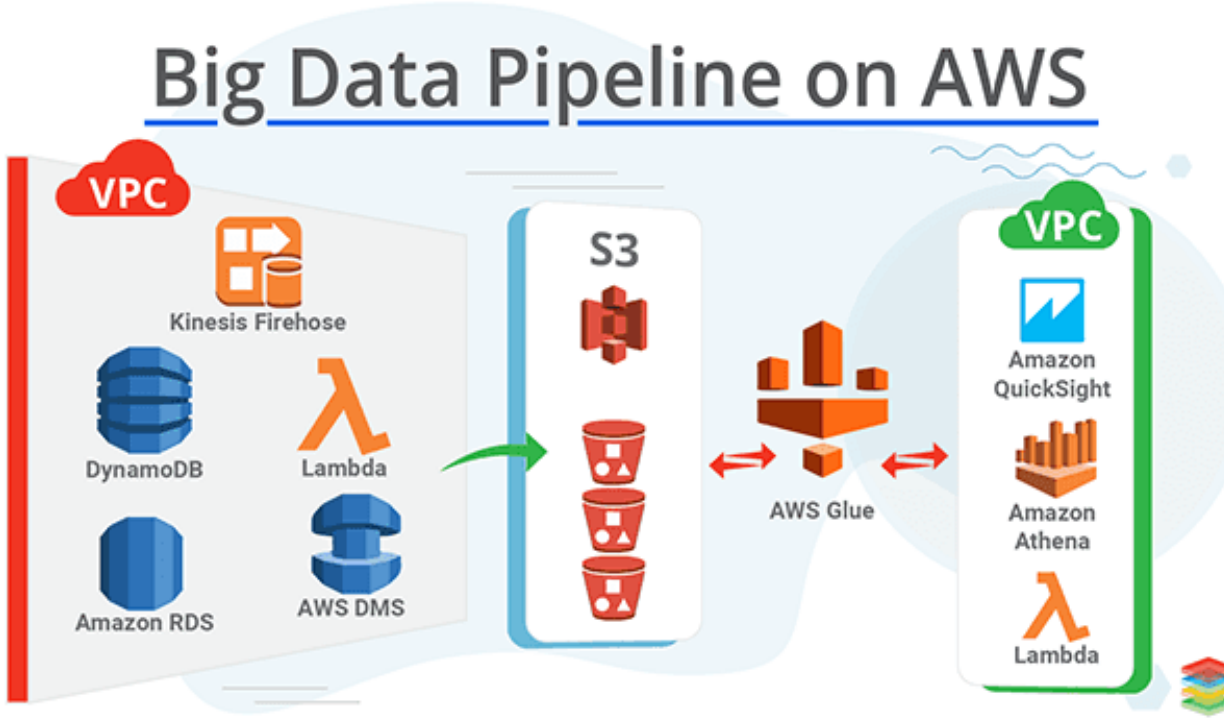
AWS BIGDATA PIPELINE



AWS BIGDATA PIPELINE

AWS Storage Services

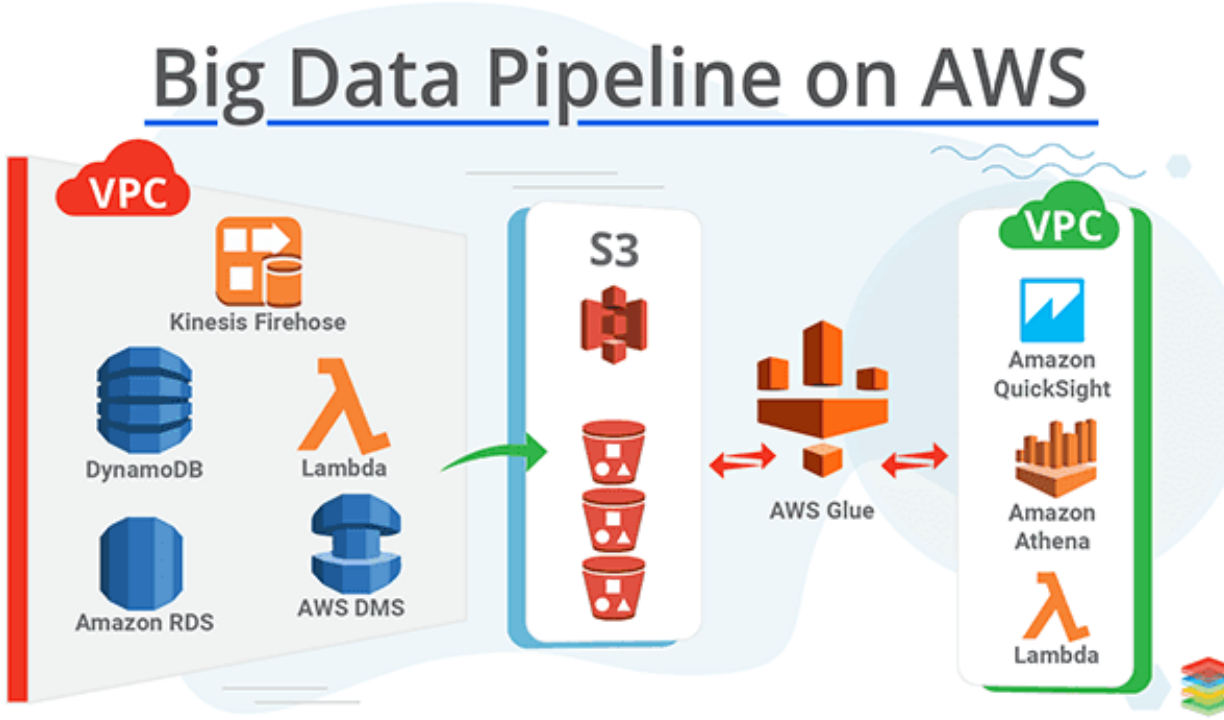
- **Amazon DynamoDB -** Fully managed NoSQL database with fast performance.
- **Amazon RDS -** It is a fully managed relational database that can accommodate large datasets. Has numerous options for the database you want, e.g., AWS aurora, Postgres, MsSQL, MariaDB.
- **Amazon Redshift -** Fully managed petabyte-scale Data Warehouse.
- **Amazon S3 -** Low-cost highly-scalable object storage.



AWS BIGDATA PIPELINE

AWS Compute Services

- **Amazon EC2 -** Service for scalable servers in AWS data center, can be used to build various types of software services.
- **Amazon EMR -** Service for distributed storage and compute over big data, using frameworks such as Hadoop and Apache Spark



AWS BIGDATA PIPELINE

What is AWS Glue?

- AWS Glue is a serverless ETL job service. While using this, we don't have to worry about setting up and managing the underlying infrastructure for running the ETL job.

How AWS GLUE Works?

- AWS glue has three main components -Data Catalog
- ETL Engine
- Scheduler