CLOUD COMPUTING

CS 346: Application Development

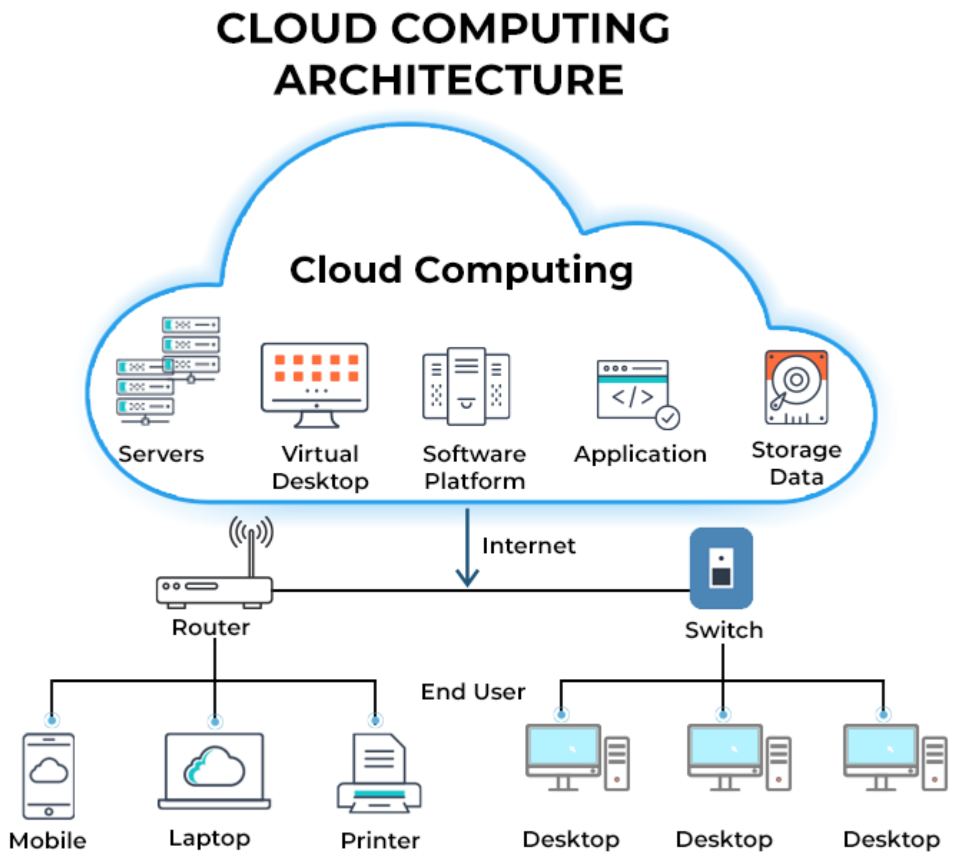
“WHY BUILD WHEN YOU CAN RENT?”

Although you can host and manage your own service and/or database, it’s more common to use large-scale platforms to host it.

Rosenberg & Mateos (2014) define cloud computing as

> "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (https://en.wikipedia.org/wiki/Cloud_computing).

Cloud computing as a concept has been around since the 1960s, but it has only become popular since around 2007, when Amazon launched its Elastic Compute Cloud (EC2) service.



Spiceworks 2024.

What are the essential characteristics that make this approach to valuable?

•A **range of preconfigured services** including application hosting, and data storage.

• **Pooled computing resources** available to any subscribing users.

• **Virtualized computing resources** to maximize hardware utilization.

• **Elastic scaling** up or down according to need.

• **Automated creation** of new virtual machines or deletion of existing ones.

• Resource usage **billed only as used.**

ADVANTAGES TO A VENDOR

1. **Cost.** You don't need to purchase and setup hardware, but can instead allocate services as you need them. This allows you to just pay for what you need, and scale up if demand increases.
2. **Flexibility**: You can allocate just the services that you need, and add different services as your needs change.
3. **Scalability**: You can add more resources on-demand, as needed. Systems can even do this automatically, e.g. as more users login to your application, more servers are launched automatically to meet demand.
4. **Efficiency**: These systems are extremely large-scale and finely tuned to provide excellent performance. You will likely not have the resources to design and build something comparable yourself.

SERVICES PROVIDED

Compute: Virtual machines, containers, serverless computing. Commonly, you can deploy code the cloud, and the cloud will run it for you. This is often called "serverless" computing.

Storage: Object storage, databases, file storage. Typically, some type of scalable database is provided, as well as storage for files and other binary objects. This will often be a NoSQL database, since these are easier to scale.

Networking: Content delivery networks, load balancing, virtual private networks. In the context of users accessing your services or databases, you will need to manage network traffic. This can be done through load balancing, or by using a content delivery network to cache data closer to users.

Security: Identity and access management, encryption, firewalls. These services will all have sophisticated user management and security features. You can control who has access to your services, and how they can interact with them.

Advanced Features: Machine learning, Big Data, Internet of Things. Platforms are increasingly offering advanced services, such as machine learning, that you can leverage.

PICK A VENDOR!

Today, there are many cloud solutions, each offering similar functionality, including Amazon Web Services (AWS), Google Cloud (GCP), Google Firebase, and Microsoft Azure.

The *main questions* you should ask when choosing a cloud provider are:

1. **What services do you need?** Different providers offer different services, and you should choose the one that best fits your needs. In this course, you will need database support, and user authentication. You may also need compute i.e. the ability to deploy a service to the cloud.
2. **What platform are you developing for**? Some services are better suited to mobile applications, others to web applications. Firebase, for instance, is a great choice for mobile applications, but doesn't have official support for desktop (Kotlin Multiplatform) - at least at the time of writing.

HIGHLIGHTS: AMAZON WEB SERVICES (AWS)

AWS consists of approximately 300 web services, each with distinct capabilities. Popular services include Storage (S3), Compute (EC2, Lambda), Networking (Route53), Security (IAM), Big Data (DynamoDB).

The AWS SDK for Kotlin is meant to provide idiomatic Kotlin functionality for accessing AWS services. See the AWS SDK for Kotlin Overview video.

Reasons to use AWS?

- It's very popular, and extremely capable (300+ services will cover anything you need!).
- Works on Android and desktop applications.

Reason to avoid AWS?

- Complexity. Can be expensive.

HIGHLIGHTS: FIREBASE

Firebase is a mobile and web application development platform, built on top of Google Cloud (GCP). If offers several services, including authentication, a real-time database, object storage and push messaging.

See What is Firebase and how to use it.

Reasons to use Firebase?

- Works *extremely* well for Android projects; Google solution for a Google OS.
- Free tier to get started.

Reason to avoid Firebase?

- May not work on desktop; no official SDK.
- Document database (vs. Supabase relational).
- Can quickly get very expensive (rate-based usage!)

HIHGLIGHTS: SUPABASE

Supabase is an open-source Firebase alternative. “Start your project with a Postgres database, Authentication, instant APIs, Edge Functions, Realtime subscriptions, Storage, and Vector embeddings.” Provides similar functionality to Firebase, but advantages!

Reasons to use Supabase?

- Works with both Android and Desktop apps. See Supabase Kotlin SDK documentation.
- PostgreSQL relational database (compared to document DB in Firebase).
- Cheap. Free tier to get started.

Reason to avoid Supabase?

- None…