

[Log In](#)

[Join](#)

[Back To Module Home](#)

# Basic Building Blocks for Modern System Design

0% completed

## Introduction to Building Blocks

## Domain Name System

## Load balancers

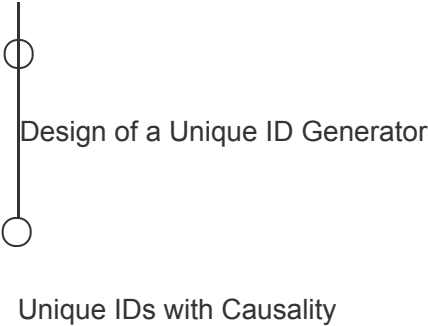
## Cache

## Databases

## Key-value Store

## Content Delivery Network (CDN)

## Sequencer



**Distributed Monitoring**

**Distributed Cache**

**Distributed Messaging Queue**

**Pub-sub**

**Rate Limiter**

**Blob Store**

**Distributed Search**

**Distributed Task Scheduler**

**Sharded Counters**

**Conclusion**

Mark Module as Completed

# System Design: Sequencer

Understand the basics of designing a sequencer.

## We'll cover the following

- Motivation
- How do we design a sequencer?

## Motivation#

There can be millions of events happening per second in a large distributed system. Commenting on a post on Facebook, sharing a Tweet, and posting a picture on Instagram are just a few examples of such events. We need a mechanism to distinguish these events from each other. One such mechanism is the assignment of globally unique IDs to each of these events.

Assigning a primary key to an entry in a database is another use case of a unique ID. Usually, the auto-increment feature in databases fulfills this requirement. However, that feature won't work for a distributed database, where different nodes independently generate the identifiers. For this case, we need a unique ID generator that acts as a primary key in a distributed setting—for example, a horizontally-sharded table.

A unique ID helps us identify the flow of an event in the logs and is useful for debugging. A real-world example of unique ID usage is Facebook's end-to-end performance tracing and analysis system, Canopy. **Canopy** uses TraceID to identify an event uniquely across the execution path that may potentially perform hundreds of microservices to fulfill one user request.

Assigning a unique TraceID to each event

## How do we design a sequencer?#

We've divided the sequencer's comprehensive design into the following two lessons:

1. **Design of a Unique ID Generator:** After enlisting the requirements of the design, we discuss three ways to generate unique IDs: using UUID, using a database, and using a range handler.
2. **Unique IDs with Causality:** In this lesson, we incorporate an additional factor of time in the generation of IDs and explain the process by taking causality into consideration.

**Unique IDs** are important for identifying events and objects within a distributed system. However, designing a unique ID generator within a distributed system is challenging. In the next lesson, let's look at the requirements for a distributed unique ID generation system.

**Back**

Quiz on CDN's Design

**Next**

Design of a Unique ID Generator

Mark as Completed

[Report an Issue](#)