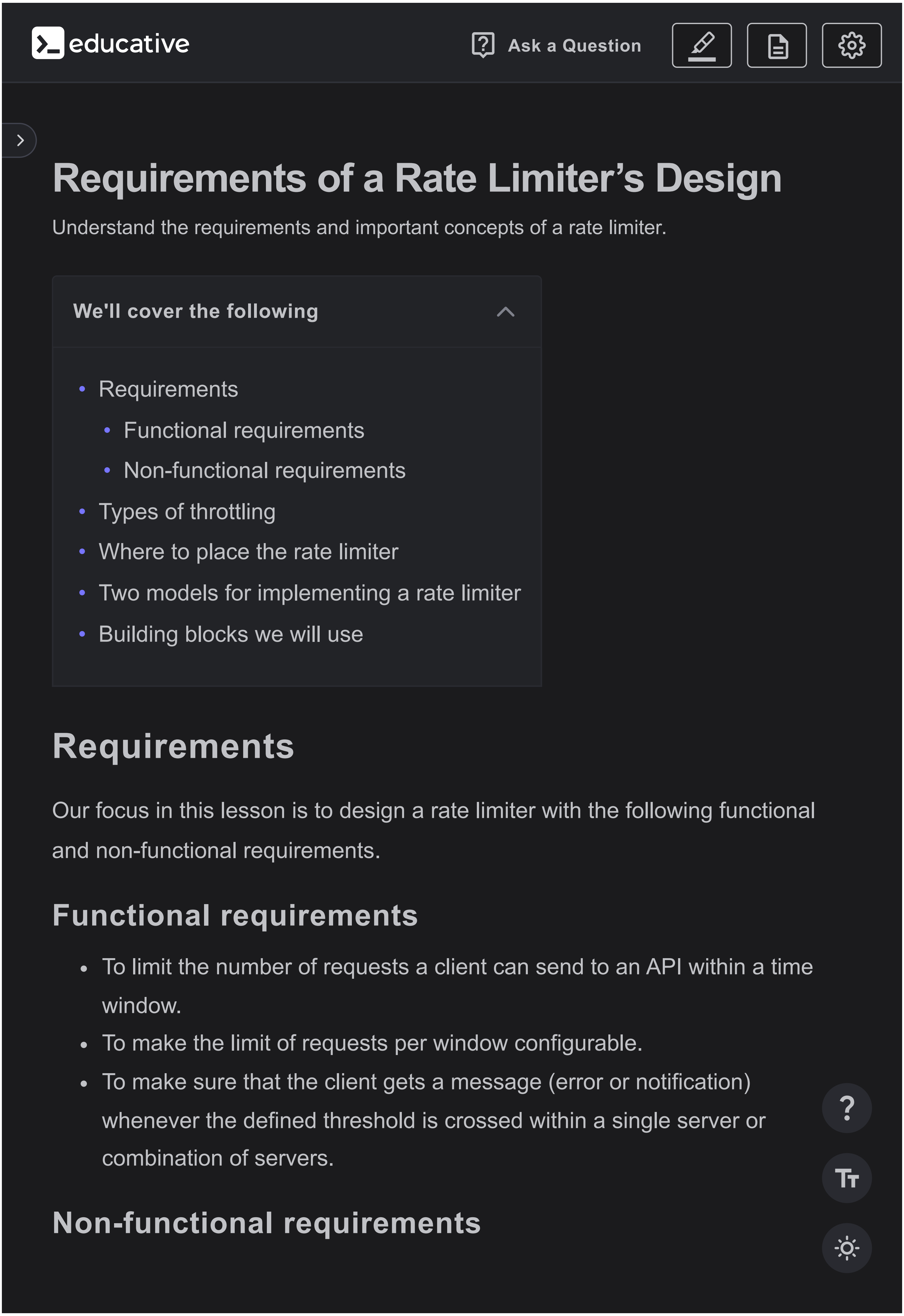


Requirements of a Rate Limiter’s Design

Understand the requirements and important concepts of a rate limiter.

We'll cover the following

Requirements

Our focus in this lesson is to design a rate limiter with the following functional and non-functional requirements.

Functional requirements

- To limit the number of requests a client can send to an API within a time window.
- To make the limit of requests per window configurable.
- To make sure that the client gets a message (error or notification) whenever the defined threshold is crossed within a single server or combination of servers.

Non-functional requirements

- **Availability:** Essentially, the rate limiter protects our system. Therefore, it should be highly available.
- **Low latency:** Because all API requests pass through the rate limiter, it should work with a minimum latency without affecting the user experience.
- **Scalability:** Our design should be highly scalable. It should be able to rate limit an increasing number of clients’ requests over time.

Types of throttling

A rate limiter can perform three types of throttling.

1. **Hard throttling:** This type of throttling puts a hard limit on the number of API requests. So, whenever a request exceeds the limit, it is discarded.
2. **Soft throttling:** Under soft throttling, the number of requests can exceed the predefined limit by a certain percentage. For example, if our system has a predefined limit of 500 messages per minute with a 5% exceed in the limit, we can let the client send 525 requests per minute.
3. **Elastic or dynamic throttling:** In this throttling, the number of requests can cross the predefined limit if the system has excess resources available. However, there is no specific percentage defined for the upper limit. For example, if our system allows 500 requests per minute, it can let the user send more than 500 requests when free resources are available.

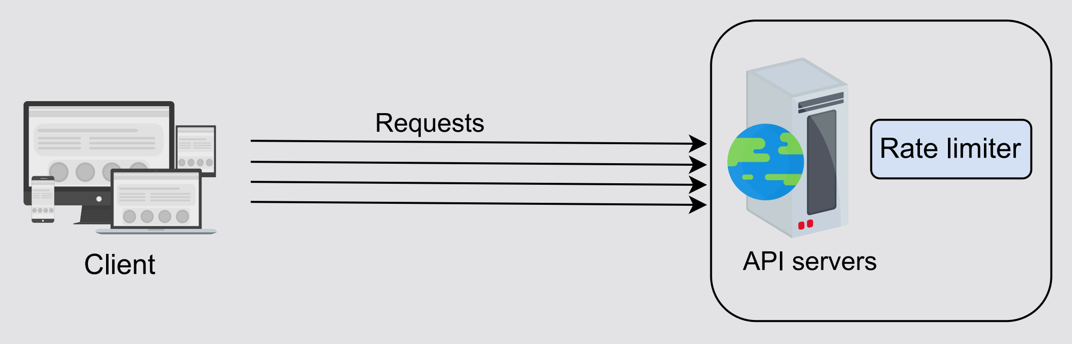
Throttling at the Operating System (OS) level

Where to place the rate limiter

There are three different ways to place the rate limiter.

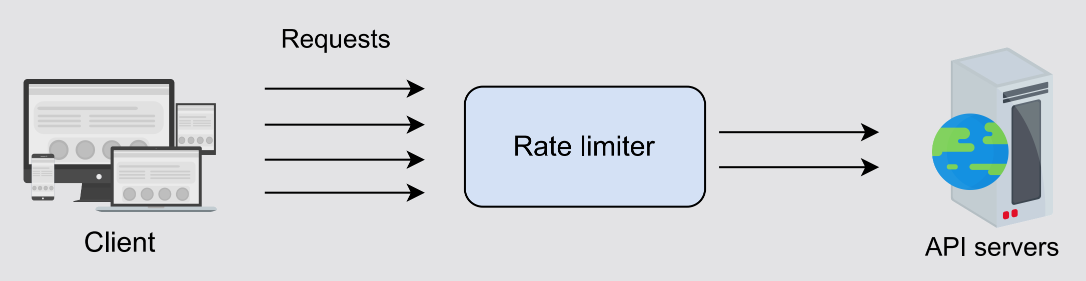
1. **On the client side:** It is easy to place the rate limiter on the client side. However, this strategy is not safe because it can easily be tampered with by malicious activity. Moreover, the configuration on the client side is also difficult to apply in this approach.

2. **On the server side:** As shown in the following figure, the rate limiter is placed on the server-side. In this approach, a server receives a request that is passed through the rate limiter that resides on the server.



Rate limiter placed at the server side

3. **As middleware:** In this strategy, the rate limiter acts as middleware, throttling requests to API servers as shown in the following figure.



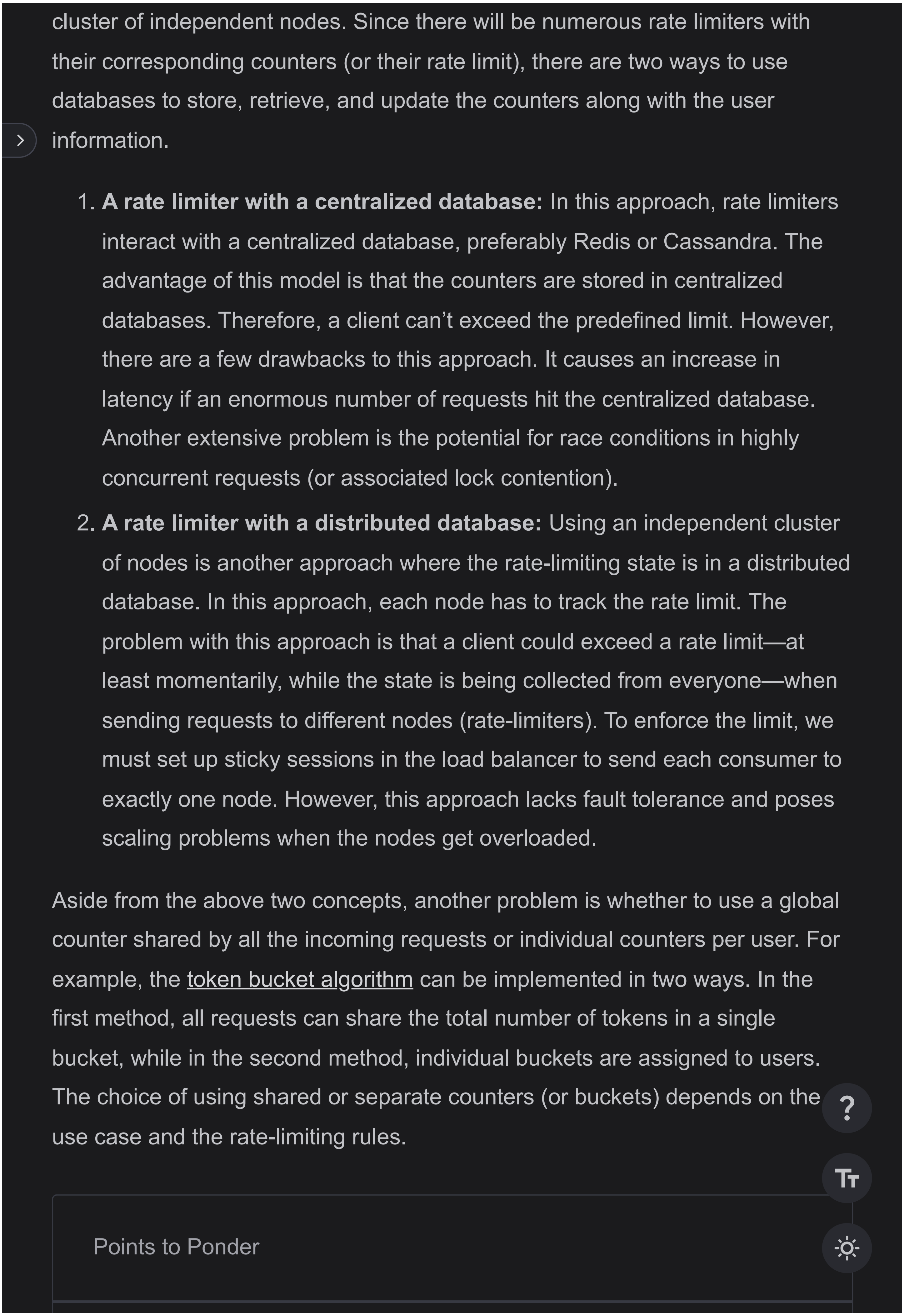
Rate limiter as middleware

Placing a rate limiter is dependent on a number of factors and is a subjective decision, based on the organization’s technology stack, engineering resources, priorities, plan, goals, and so on.

> **Note:** Many modern services use APIs to provide their functionality to the clients. API endpoints can be a good vantage point to rate limit the incoming client traffic because all traffic passes through them.

Two models for implementing a rate limiter

One rate limiter might not be enough to handle enormous traffic to support millions of users. Therefore, a better option is to use multiple rate limiters as a



cluster of independent nodes. Since there will be numerous rate limiters with their corresponding counters (or their rate limit), there are two ways to use databases to store, retrieve, and update the counters along with the user information.

1. **A rate limiter with a centralized database:** In this approach, rate limiters interact with a centralized database, preferably Redis or Cassandra. The advantage of this model is that the counters are stored in centralized databases. Therefore, a client can't exceed the predefined limit. However, there are a few drawbacks to this approach. It causes an increase in latency if an enormous number of requests hit the centralized database. Another extensive problem is the potential for race conditions in highly concurrent requests (or associated lock contention).
2. **A rate limiter with a distributed database:** Using an independent cluster of nodes is another approach where the rate-limiting state is in a distributed database. In this approach, each node has to track the rate limit. The problem with this approach is that a client could exceed a rate limit—at least momentarily, while the state is being collected from everyone—when sending requests to different nodes (rate-limiters). To enforce the limit, we must set up sticky sessions in the load balancer to send each consumer to exactly one node. However, this approach lacks fault tolerance and poses scaling problems when the nodes get overloaded.

Aside from the above two concepts, another problem is whether to use a global counter shared by all the incoming requests or individual counters per user. For example, the token bucket algorithm can be implemented in two ways. In the first method, all requests can share the total number of tokens in a single bucket, while in the second method, individual buckets are assigned to users. The choice of using shared or separate counters (or buckets) depends on the use case and the rate-limiting rules.

Points to Ponder

Question 1

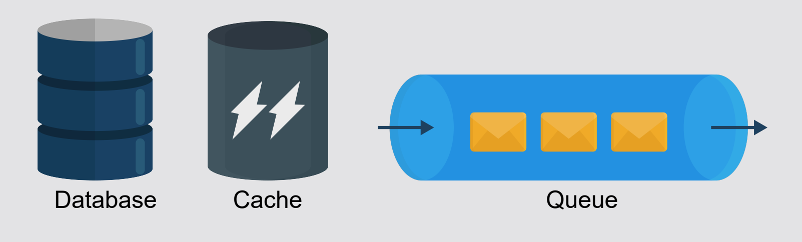
Can a load balancer be used as a rate limiter?

Show Answer

1 of 2

Building blocks we will use

The design of the rate limiter utilizes the following building blocks that we discussed in the initial chapters.



Building blocks in the design of a rate limiter

- **Databases** are used to store rules defined by a service provider and metadata of users using the service.
- **Caches** are used to cache the rules and users’ data for frequent access.
- **Queues** are essential for holding the incoming requests that are allowed by the rate limiter.

In the next lesson, we’ll focus on a high-level and detailed design of a rate limiter based on the requirements discussed in this lesson.

← Back
System Design: The Rate Limiter

☑ Mark As Completed

Next
Design of a Rate Limiter