**What are some basic examples here?**

Build a rate limiter in java

What are the different token bucket algo?

The Sliding Window Counter algorithm is a more efficient variation of the Sliding Window Log algorithm. It is a hybrid that combines the fixed window counter and sliding window log. Instead of maintaining a log of request timestamps, it calculates the weighted counter for the previous time window. When a new request arrives, the counter is adjusted based on the weight, and the request is allowed if the total is below the limit.

Let’s walk through a concrete example. In this example, the rate limiter allows a maximum of 7 requests per minute. There are 5 requests in the previous minute and 3 in the current minute. For a new request that arrives at 30% (18 seconds) into the current minute, the number of requests in the rolling window is calculated using the following formula:

Requests in current window + requests in the previous window \* overlap percentage of the rolling window and previous window

Using this formula, we get 3 + 5 \* 0.7% = 6.5 requests. Depending on the use case, the number can either be rounded up or down. In our example, it is rounded down to 6.

[[A diagram of a building

Description automatically generated](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F4c21343a-94f7-409d-b024-fb4b3a6e1ce3_1376x734.png)](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F4c21343a-94f7-409d-b024-fb4b3a6e1ce3_1376x734.png" \t "_blank)

Since the rate limiter in this example allows 7 requests per minute, the current request can go through.

There is another way to implement this algorithm that is more complicated. Instead of computing a weighted counter for the previous window, it uses a counter for each time slot within the window.  We will not discuss this other implementation here.

**Pros and cons here**

The Sliding Window Counter algorithm has its pros and cons. It smooths out spikes in traffic because the limit includes the average rate of the previous window in the calculation. It also reduces storage and processing requirements compared to the Sliding Window Log algorithm.

However, it may still allow bursts of requests to slip through. It is an approximation of the actual rate because it assumes requests in the previous window are evenly distributed. This problem may not be as bad as it seems. According to [experiments done by Cloudflare](https://blog.cloudflare.com/counting-things-a-lot-of-different-things/), only 0.003% of requests are incorrectly allowed or rate limited among 400 million requests.

**Token bucket here**

The Token Bucket algorithm is widely used for rate limiting. It is simple, well understood and commonly used by large tech companies. Both [Amazon](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html) and [Stripe](https://stripe.com/blog/rate-limiters) use this algorithm to throttle their API requests.

The Token Bucket algorithm uses a "bucket" to hold tokens. The tokens represent the allowed number of requests.

The bucket is initially filled with tokens, and tokens are added at a fixed rate over time. When a request arrives, it consumes a token from the bucket, and the request is allowed if there are enough tokens.

Let’s walk through an example. Here, the token bucket capacity is 4. The refiller puts 2 tokens into the bucket every second. Once the bucket is full, extra tokens will overflow.

[[A diagram of a refilling process

Description automatically generated](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fed30d402-1533-4637-bbb7-27bdfda16f04_1600x798.png)](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fed30d402-1533-4637-bbb7-27bdfda16f04_1600x798.png" \t "_blank)

Each request consumes one token. When a request arrives, we check if there are enough tokens in the bucket.  
 note that the requets consumes the token her e  
  
**The diagram below explains how it works.**

If there are enough tokens, we take one token out for each request, and the request goes through. If there are not enough tokens, the request is dropped.

[[A diagram of a process flow

Description automatically generated](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F11d02ea3-7ff9-4420-adde-0015b236780b_1600x1435.png)](https://substackcdn.com/image/fetch/f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2F11d02ea3-7ff9-4420-adde-0015b236780b_1600x1435.png" \t "_blank)

The following diagram illustrates how token consumption, refill, and rate limiting logic work. In this example, the token bucket size is 4, and the refill rate is 4 per 1 minute.

A screenshot of a screenshot of a cell phone

Description automatically generated

The token bucket algorithm takes two parameters:

* Bucket size: the maximum number of tokens allowed in the bucket
* Refill rate: number of tokens put into the bucket every second

How many buckets do we need? This depends on the rate-limiting rules. Here are a few examples.

**1.Different bucket for differnet API endpoint:**

It is usually necessary to have different buckets for different API endpoints. For instance, if a user is allowed to make 1 post per second, add 150 friends per day, and like 5 posts per second, 3 buckets are required for each user.

2. Throttle based on IP Address  
  
If we need to throttle requests based on IP addresses, each IP address requires a bucket.

3. Global bucket

* If the system allows a maximum of 10,000 requests per second, it makes sense to have a global bucket shared by all requests.

This algorithm allows for a smooth distribution of requests and can handle bursts of requests up to the bucket's capacity. It is memory efficient and relatively easy to implement.

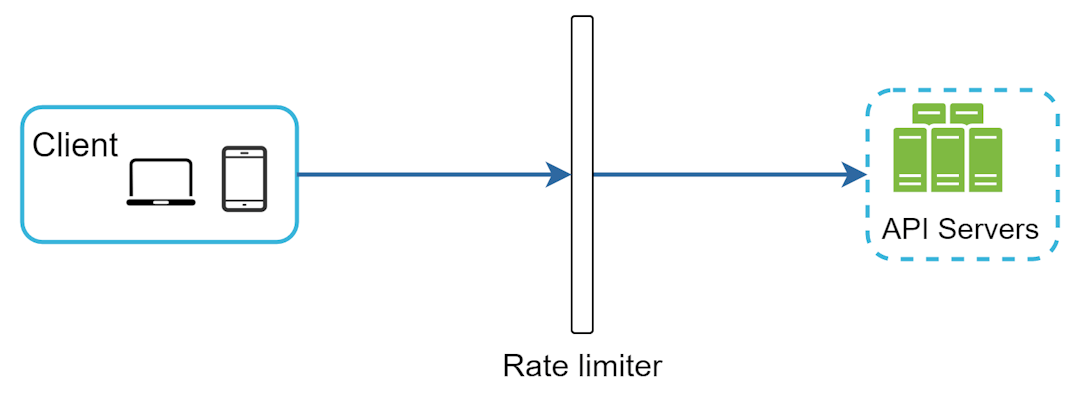
**Leaky bucket**

Leaky bucket here

Here are a few examples:

* A user can write no more than 2 posts per second.
* You can create a maximum of 10 accounts per day from the same IP address.
* You can claim rewards no more than 5 times per week from the same device.

And then



And the above should look like that

1. API gateway here would support all this here,

**Method 1: Using token bucket algorithm here**

A screenshot of a computer screen

Description automatically generated

. Both Amazon [5] and Stripe [6] use this algorithm to throttle their API requests.

**How many buckets do we need?**

It is usually necessary to have different buckets for different API endpoints. For instance, if a user is allowed to make 1 post per second, add 150 friends per day, and like 5 posts per second, 3 buckets are required for each user.

* A token bucket is a container that has pre-defined capacity. Tokens are put in the bucket at preset rates periodically. Once the bucket is full, no more tokens are added. As shown in Figure 4, the token bucket capacity is 4. The refiller puts 2 tokens into the bucket every second. Once the bucket is full, extra tokens will overflow.

**A tokening alogirhtm here**

1. Each request consumes 1 token, When a request arrives, we check if there are enough tokens in the bucket. Figure 5 explains how it works.
2. f there are enough tokens, we take one token out for each request, and the request goes through.
3. If there are not enough tokens, the request is dropped.

**What are the parameters here?**

* Bucket size: the maximum number of tokens allowed in the bucket
* Refill rate: number of tokens put into the bucket every second

**What’s the pros and cons here?**

Pros:

* The algorithm is easy to implement.
* Memory efficient.
* Token bucket allows a burst of traffic for short periods. A request can go through as long as there are tokens left.

Cons:

* Two parameters in the algorithm are bucket size and token refill rate. However, it might be challenging to tune them properly.

**Leaking bucket algorithm?**

1. And then here

A screenshot of a video game

Description automatically generated

**What’s the token bucket algorithm?**