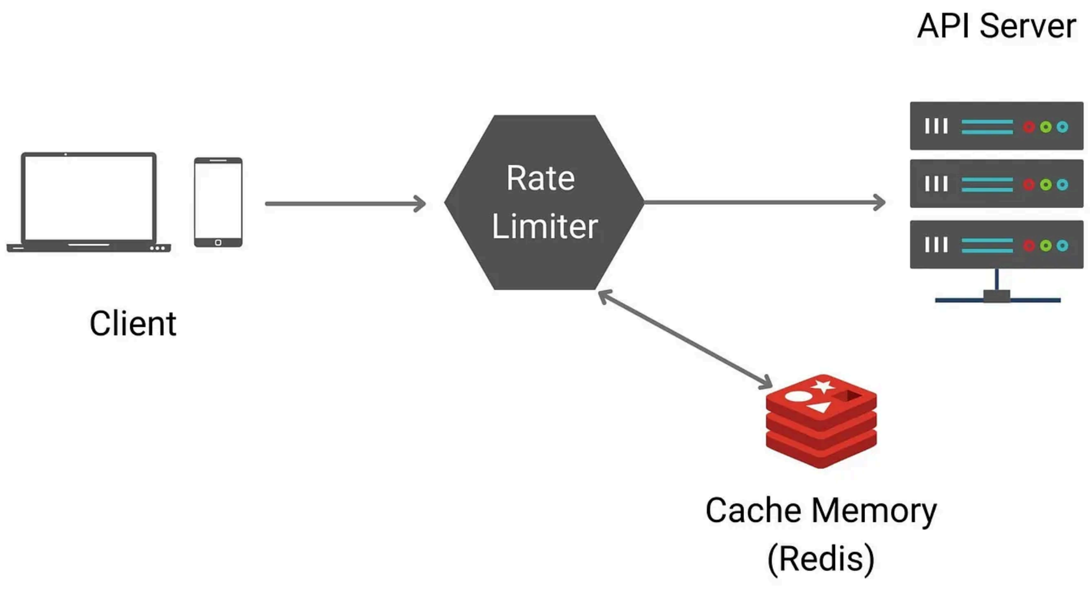
RATE LIMITING PATTERN

3. **Configure FastAPI with `fastapi-limiter`:**

python Copy code

```python
from fastapi import FastAPI, Depends
from fastapi_limiter import FastAPILimiter
from fastapi_limiter.depends import RateLimiter

app = FastAPI()

@app.on_event("startup")
async def startup():
    # Initialize FastAPILimiter with Redis connection
    await FastAPILimiter.init("redis://localhost:6379")

@app.get("/items", dependencies=[Depends(RateLimiter(times=5, seconds=60))])
async def get_items():
    return {"message": "This endpoint is rate-limited to 5 requests per minute'
```

4. **Explanation**:
   - `times`: Number of allowed requests.
   - `seconds`: Time window for the rate limit.
   - Redis is used to track request counts and timestamps.
5. **Run the App**: Start the app and test the endpoint. After 5 requests within a minute, additional requests will receive a `429 Too Many Requests` response.



source: FASTAPI rate limiting

https://medium.com/@viswanathan.arjun/develop-your-own-api-rate-limiter-in-fastapi-part-ii-e49917d91a04

Summary

- **FastAPI**: Ideal for lightweight, application-level rate limiting with user-specific logic. Requires external tools like Redis for distributed setups.
- **NGINX**: Strong at the edge or as an API gateway, with efficient performance and scalability but less flexibility.
- **Istio**: Best for complex service meshes needing fine-grained, distributed, and dynamic rate-limiting capabilities.

Here's a comparison table of **FastAPI**, **NGINX**, and **Istio** rate-limiting capabilities:

<table picture>

Best Practices:

1. **Use Distributed Rate Limiting**:
   - Use Redis or another centralized store to ensure rate limits are consistent across multiple application instances.
2. **Implement Exponential Backoff**:
   - Return `Retry-After` headers to guide clients on when to retry.
3. **Secure Redis Connection**:
   - Use authentication and encryption for Redis in production.
4. **Monitor and Log Limits**:
   - Log rate-limiting events to track abuse or misconfiguration.

This allows you to build robust, scalable rate-limiting mechanisms with FastAPI.

Best Practices with NGINX:

1. **Use `ngx_http_limit_req_module`**:
   - Leverage the `limit_req` module for token-bucket-style rate limiting.

Example configuration:
nginx
Copy code
```
http {

    limit_req_zone $binary_remote_addr zone=addr:10m rate=10r/s;

    server {

        location /api/ {

            limit_req zone=addr burst=20 nodelay;

            proxy_pass http://backend;

        }

    }

}
```

-
    - **Security Tip**: Use `$binary_remote_addr` (hashed IP) to avoid leaking raw IPs in memory.

Best Practices with Istio:

1. **Use Istio's Rate Limiting Policies**:
    - Leverage Istio's **EnvoyFilter** or the `QuotaSpec` configuration for rate limiting.

Example using EnvoyFilter:
yaml
Copy code

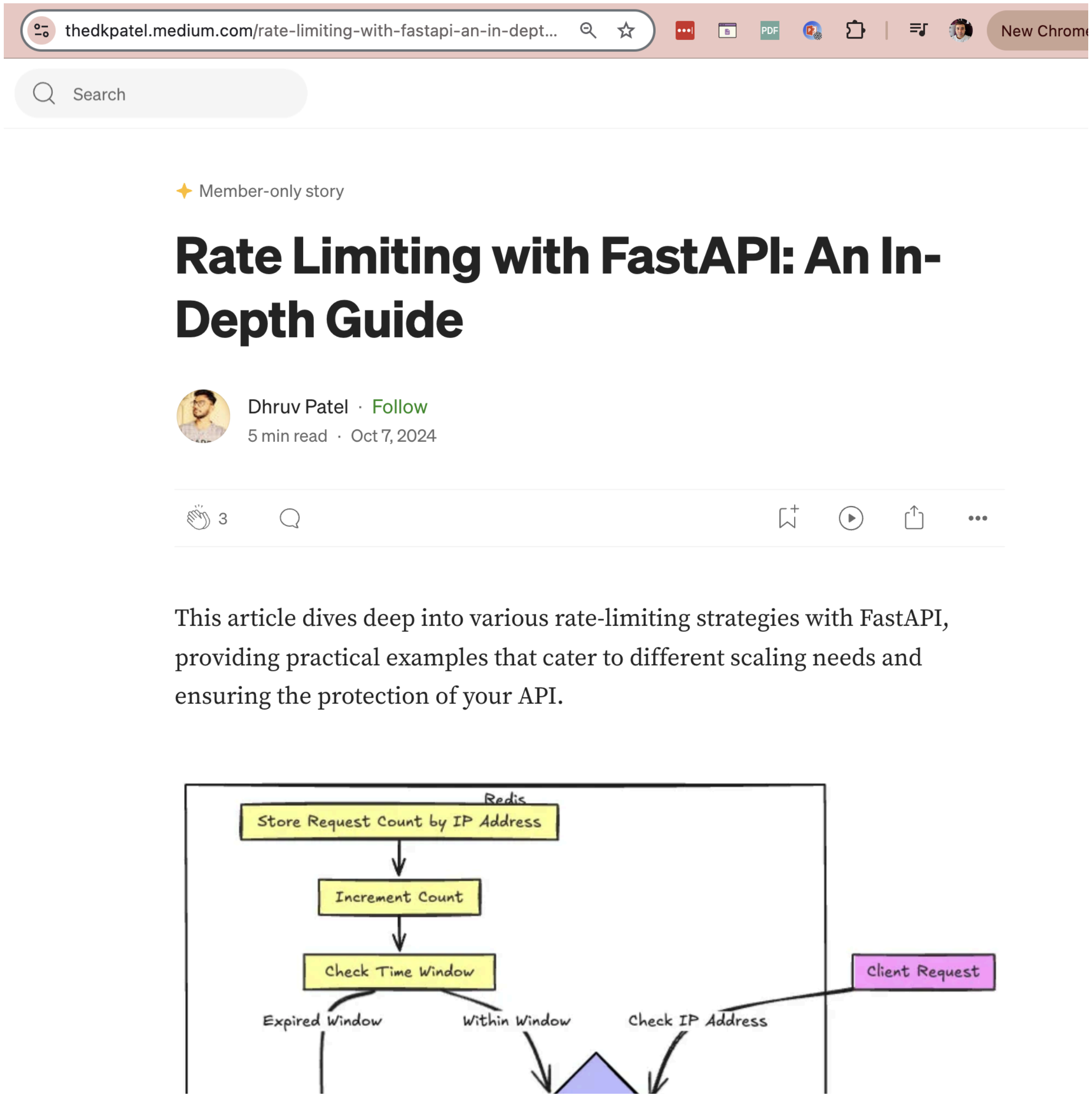
```
apiVersion: networking.istio.io/v1alpha3

kind: EnvoyFilter

metadata:

  name: rate-limit

  namespace: istio-system

spec:

  configPatches:

  - applyTo: HTTP_FILTER

    match:

      context: SIDECAR_INBOUND

    patch:

      operation: INSERT_BEFORE

      value:

        name: envoy.filters.http.local_ratelimit

        typed_config:
          "@type":
type.googleapis.com/envoy.extensions.filters.http.local_ratelimit
.v3.LocalRateLimit
          stat_prefix: http_local_rate_limiter
          token_bucket:
            max_tokens: 10
            tokens_per_fill: 10
            fill_interval: 1s
```

    ○

2. **Distributed Rate Limiting**:
    - Use Istio with external systems like Redis or gRPC rate-limiting services for distributed enforcement.
    - For more scalability, configure a **Rate Limit Service** (RLS) to centralize policies.

This pattern dives deep into various rate-limiting strategies with FastAPI, providing practical examples that cater to different scaling needs and ensuring the protection of your API.

Rate limiting is a technique used to control the number of requests a client can make to an API within a certain timeframe. This prevents abuse, mitigates the impact of Denial of Service (DoS) attacks, and ensures fair usage of system resources. FastAPI, known for its speed and ease of use, can be equipped with rate limiting using various approaches, making it a robust option for building secure and efficient APIs.


thedkpatel.medium.com/rate-limiting-with-fastapi-an-in-dept...

Search

Member-only story

Rate Limiting with FastAPI: An In-Depth Guide

Dhruv Patel · Follow
5 min read · Oct 7, 2024

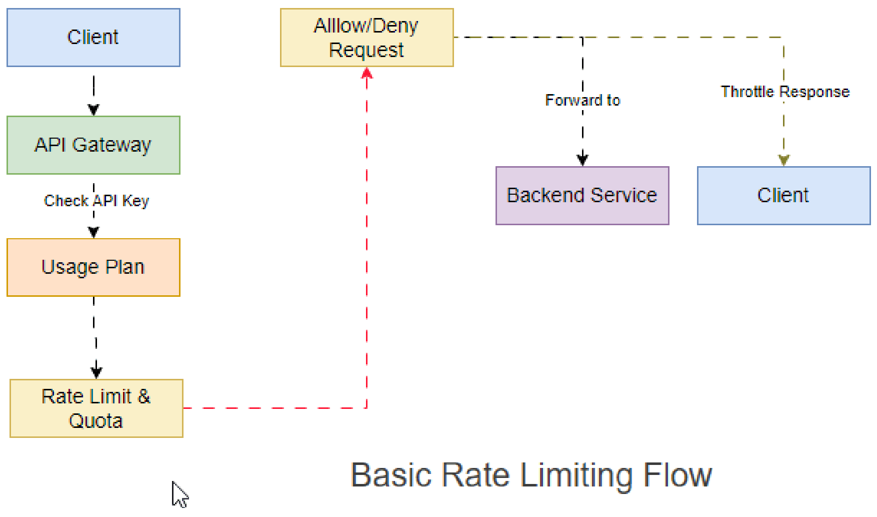
3

This article dives deep into various rate-limiting strategies with FastAPI, providing practical examples that cater to different scaling needs and ensuring the protection of your API.

https://thedkpatel.medium.com/rate-limiting-with-fastapi-an-in-depth-guide-c4d64a776b83

FastAPI, a modern web framework for building APIs with Python, offers a robust platform for developing high-performance web applications. Managing the rate at which clients can access your API endpoints is essential to prevent abuse, manage server load, and ensure fair usage of resources. It is a best practice not just for your externally exposed applications but also for internal applications.

https://thelearningfellow.medium.com/quick-and-easy-rate-limiting-for-fastapi-6be34e8102a7



Basic Rate Limiting Flow

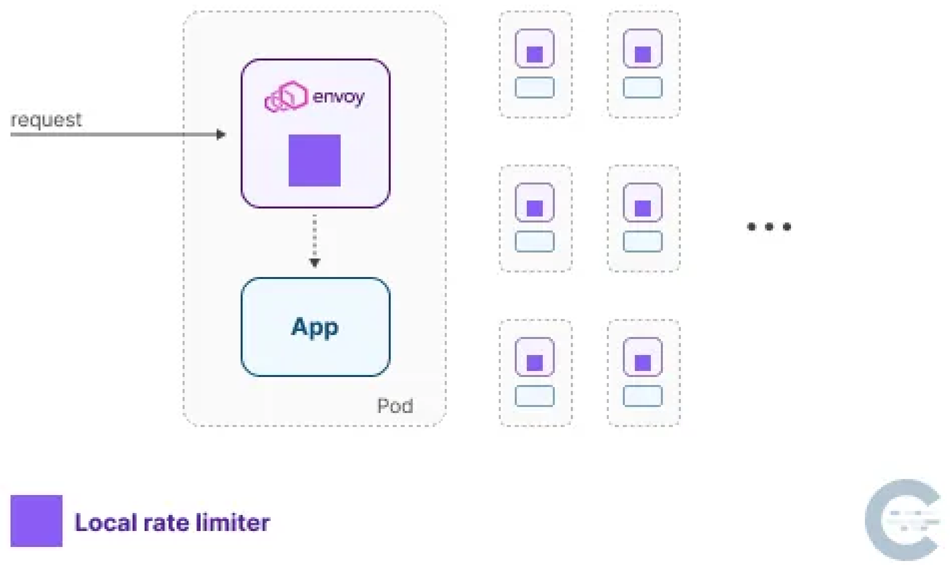
A rate limit is a restriction on the number of requests that a user or client can make to a server within a specified time period. The rate limit can be applied to requests based on various criteria, such as the user’s IP address, authentication credentials, or the requested URL.

https://medium.com/@rajeshpillai/api-rate-limiting-2542c2a90b38

istio rate limiting

https://medium.com/@ishujeetpanjeta/mastering-istio-rate-limiting-for-efficient-traffic-management-ff5acbcde7b3

https://istio.io/latest/docs/tasks/policy-enforcement/rate-limit/

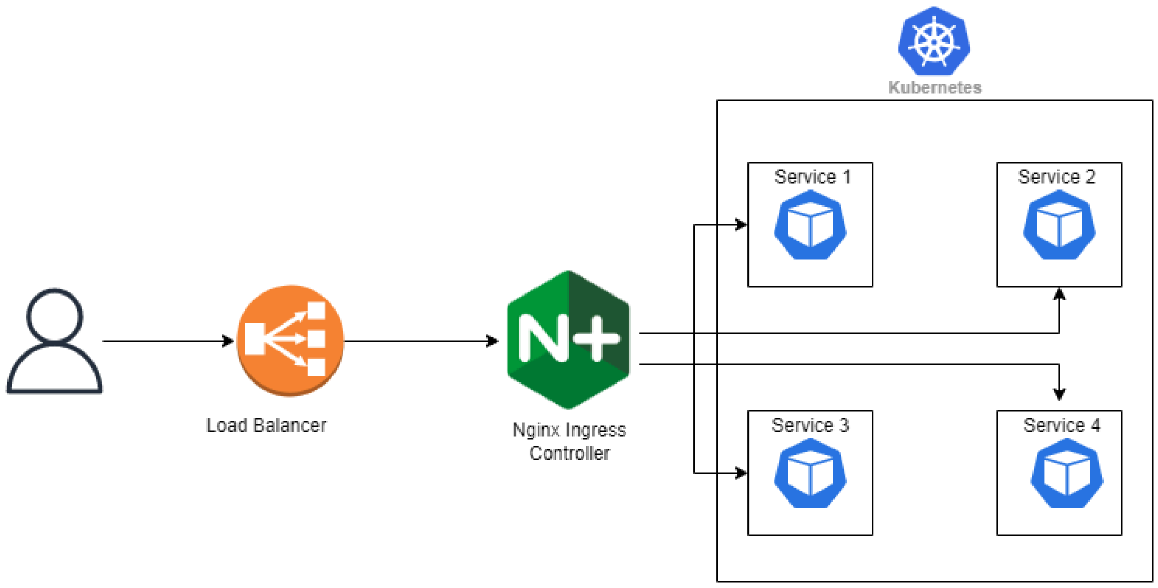


NGINX rate limiting

https://blog.nginx.org/blog/rate-limiting-nginx

https://rednafi.com/go/rate_limiting_via_nginx/

https://docs.nginx.com/nginx/admin-guide/security-controls/controlling-access-proxied-http/



https://nidhiashtikar.medium.com/implementation-of-an-ingress-controller-using-nginx-6b034ec23eac