A forward proxy is a server that sits between user devices and the internet. A forward proxy is commonly used for: - Protect clients - Avoid browsing restrictions - Block access to certain content

A reverse proxy is a server that accepts a request from the client, forwards the request to web servers, and returns the results to the client as if the proxy server had processed the request. A reverse proxy is good for: - Protect servers - Load balancing - Cache static contents - Encrypt and decrypt SSL communication

A diagram of a network

Description automatically generated

### Forward Proxy() vs Reverse Proxy (LB):

<https://www.youtube.com/watch?v=JedHPaRtfkY>

<https://www.youtube.com/watch?v=HrG0MHkSsCA>

A **proxy server** acts as an intermediary between a client and a server. There are two main types of proxy servers: **forward proxies** and **reverse proxies**.

**1. Forward Proxy:**

* A **forward proxy** is positioned between the client and the internet (or target server). It acts on behalf of the client by forwarding requests to the destination server and then sending the response back to the client.
* **Use cases:**
  + **Privacy and anonymity**: Forward proxies can hide the client's IP address from the destination server.
  + **Content filtering**: Organizations can use forward proxies to control access to certain websites or content.
  + **Caching**: Forward proxies can cache data to improve response time for frequently accessed resources.
  + **Access to restricted content**: Clients can use a forward proxy to access websites restricted in their location.

**How it works:**

* The client makes a request to the forward proxy.
* The proxy relays the request to the target server.
* The server sends the response to the proxy, which then forwards it to the client.

**2. Reverse Proxy:**

* A **reverse proxy** is positioned between the server and the client. It receives client requests and forwards them to one or more servers behind it. The client sees the reverse proxy as the actual server.
* **Use cases:**
  + **Load balancing**: Distributing incoming client requests across multiple servers to balance the load and improve reliability.
  + **Security**: The reverse proxy can protect servers from direct exposure to the internet, providing an additional security layer.
  + **SSL termination**: A reverse proxy can manage SSL certificates and encryption, reducing the load on backend servers.
  + **Caching**: It can cache responses to reduce the load on backend servers for frequently requested data.

**How it works:**

* The client sends a request to the reverse proxy.
* The proxy forwards the request to the appropriate backend server.
* The backend server sends the response to the proxy, which forwards it to the client.

**Comparison:**

| **Feature** | **Forward Proxy** | **Reverse Proxy** |
| --- | --- | --- |
| Acts on behalf of | Client | Server |
| Client visibility | Client knows it's interacting with a proxy | Client believes it's interacting with a server |
| Use cases | Anonymity, access control, caching | Load balancing, security, caching, SSL termination |

Both proxies serve as intermediaries but serve different purposes based on where they're placed in the communication flow.

### Reverse proxy-Load balancing API Gateway is example

### DK-Blocking Linked in and FB from company premises in forward proxy.

A **load balancing URL** is an example of a **reverse proxy**.

Here's why:

* A **reverse proxy** sits in front of a group of backend servers and distributes incoming client requests across them. This is how load balancing works: when a client makes a request to the load balancer (which is acting as a reverse proxy), the load balancer forwards the request to one of the backend servers, balancing the load.

In this scenario:

* The **client** interacts with the load balancer, which it believes to be the server.
* The **reverse proxy (load balancer)** then forwards the requests to the appropriate backend server (according to the load-balancing strategy).

Examples of reverse proxies used for load balancing:

* **NGINX** or **HAProxy** configured for load balancing.
* **AWS Elastic Load Balancer (ELB)** or **Google Cloud Load Balancer**.

So, a load balancer is a classic example of a **reverse proxy**.

An **API Gateway** is an example of a **reverse proxy**.

**Here's why:**

* An API Gateway typically sits between clients (e.g., web or mobile apps) and backend services. It receives API requests from clients and routes them to the appropriate backend services, often handling tasks such as request routing, authentication, rate limiting, and response aggregation.

In this role:

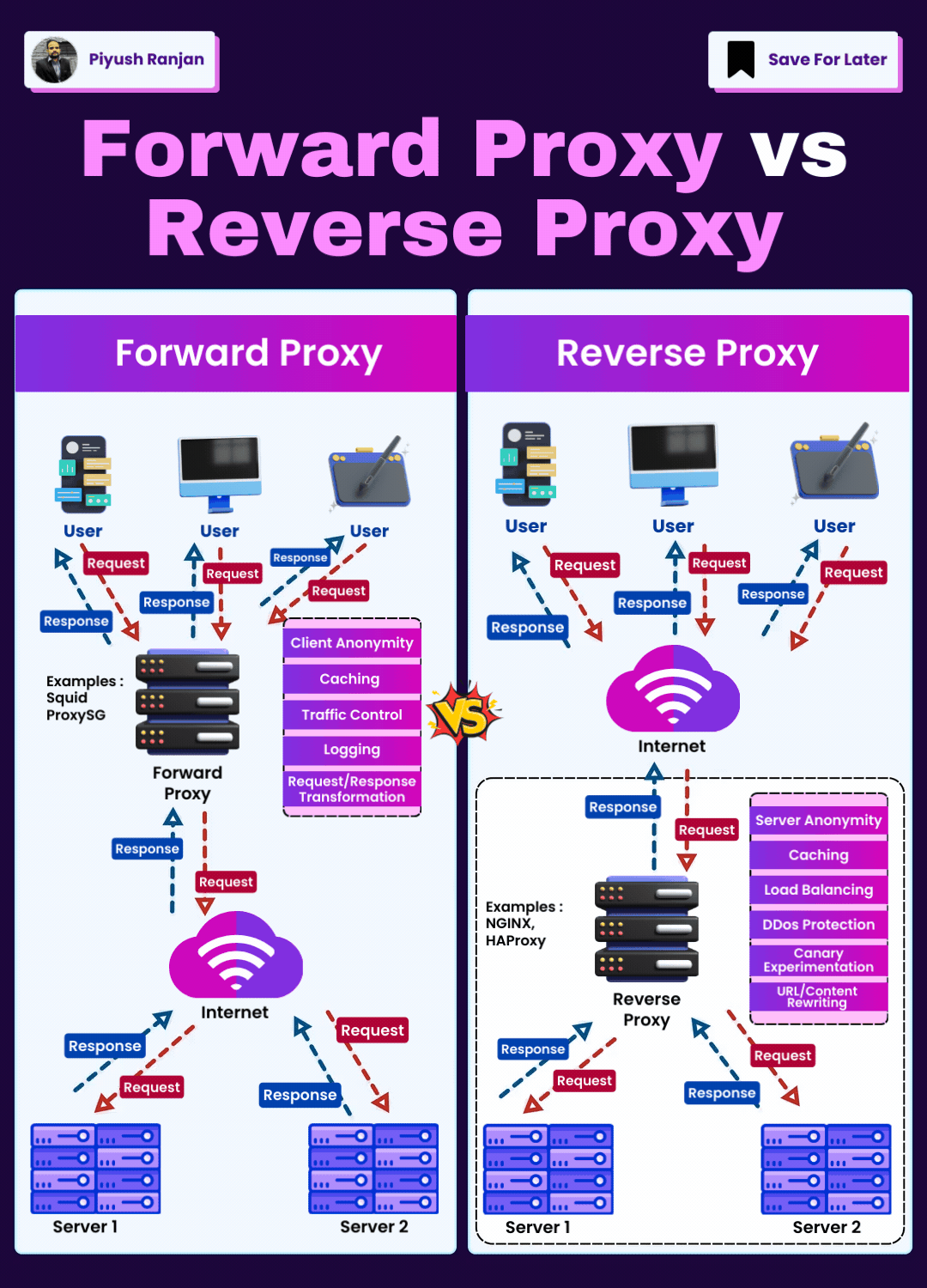
* The **client** sends requests to the API Gateway (which is the reverse proxy), and the gateway forwards the requests to one or more backend services (similar to how a reverse proxy forwards requests to backend servers).
* The API Gateway can also handle **load balancing**, **security** (e.g., managing authentication and authorization), **caching**, and **traffic shaping**—all features typically associated with a reverse proxy.

**Key similarities between API Gateway and Reverse Proxy:**

* Both sit in front of the backend services and manage incoming traffic.
* Both can distribute requests to different servers or services, adding a layer of abstraction between the client and the backend.

In summary, an **API Gateway** functions as a **reverse proxy** with additional API management features.

Understanding the differences between Forward Proxy and Reverse Proxy is essential for optimizing network performance and security. Here’s a quick overview:  
  
Forward Proxy:  
1. Client Anonymity: Hides client details from the server.  
2. Caching: Speeds up response times by storing frequently accessed data.  
3. Traffic Control: Manages and controls outbound traffic.  
4. Logging: Tracks client requests for monitoring and analysis.  
5. Request/Response Transformation: Alters requests and responses for various needs.  
6. Examples: Squid, ProxySG.  
  
Reverse Proxy:  
1. Server Anonymity: Conceals server details from clients.  
2. Caching: Enhances performance by caching server responses.  
3. Load Balancing: Distributes incoming requests across multiple servers to ensure optimal performance.  
4. DDoS Protection: Shields servers from distributed denial-of-service attacks.  
5. Canary Experimentation: Enables testing of new features with a subset of users.  
6. URL/Content Rewriting: Modifies URLs and content dynamically.  
7. Examples: NGINX, HAProxy.



### Real example of forward proxy in java web application

In a Java web application, a **forward proxy** is typically used to route outgoing requests from the client to external servers, such as when the application needs to access external APIs, download resources, or handle internet traffic through a proxy for security or monitoring purposes.

Here’s a real example of using a forward proxy in a Java web application with **HttpURLConnection** and **Apache HttpClient**:

**Example 1: Using HttpURLConnection**

In this example, a forward proxy is set up for outgoing HTTP requests.

A computer screen shot of a program code

Description automatically generated

In this code:

* The proxy settings are defined using the Proxy object.
* The HttpURLConnection uses the forward proxy to send a request to the target URL (www.example.com).

**Example 2: Using Apache HttpClient**

You can use **Apache HttpClient**, a popular library for making HTTP requests in Java, with a forward proxy.

A screen shot of a computer program

Description automatically generated

**Explanation:**

* In this example, HttpHost is used to define the forward proxy with the hostname and port of the proxy server.
* The HttpClients.custom() method is used to create an HttpClient instance that is configured to use the proxy.
* When the request is executed, it is routed through the forward proxy to the target URL.

**Real-World Use Case:**

Forward proxies are useful in scenarios where a Java web application:

* Needs to route all its outgoing internet requests through a corporate proxy for security reasons.
* Is hosted in an environment where internet access is controlled (e.g., data centers or behind firewalls), and only certain IP addresses (the proxy) are allowed to make requests.
* Requires anonymizing outgoing requests, masking the internal IP address, or bypassing geographic restrictions.

This kind of setup is often seen in corporate environments or when an application needs to interact with external services through a proxy for monitoring or filtering purposes.

### What id forward proxy and reverse proxy

ChatGPT

**ChatGPT**

Forward and reverse proxies are intermediary servers that sit between clients and servers, facilitating communication between them. They serve different purposes and operate in distinct ways:

1. **Forward Proxy:**
   * **Purpose:** A forward proxy acts on behalf of clients to access resources from the internet or other external networks.
   * **Client-Side Proxy:** When a client makes a request to access a resource, such as a website or service, it sends the request to the forward proxy server instead of directly contacting the destination server.
   * **Anonymity and Security:** Forward proxies can provide anonymity and security for clients by masking their IP addresses and filtering outbound traffic. They can also enforce access controls and policies to regulate internet usage within an organization.
   * **Caching:** Forward proxies can cache frequently accessed content, reducing bandwidth usage and improving performance for subsequent requests.
   * **Example:** In a corporate network, a forward proxy may be deployed to control and monitor internet access for employees. When an employee requests access to a website, the request is first sent to the forward proxy server, which then retrieves the requested content from the internet on behalf of the client.
2. **Reverse Proxy:**
   * **Purpose:** A reverse proxy sits in front of servers and acts as a gateway for incoming client requests, directing traffic to the appropriate backend servers based on various criteria.
   * **Server-Side Proxy:** When a client sends a request to access a service hosted on a server, the request is received by the reverse proxy instead of directly reaching the backend server.
   * **Load Balancing:** Reverse proxies often perform load balancing by distributing incoming client requests across multiple backend servers. This helps distribute the workload evenly and ensures high availability and scalability of the service.
   * **Security and SSL Termination:** Reverse proxies can enhance security by serving as a barrier between clients and backend servers, protecting the servers from direct exposure to the internet. They can also handle SSL/TLS encryption and decryption, offloading this processing burden from the backend servers.
   * **Content Caching and Compression:** Reverse proxies can cache static content and compress responses, improving performance by reducing the load on backend servers and speeding up content delivery to clients.
   * **Example:** In a web application deployment, a reverse proxy may be used to distribute incoming HTTP requests across multiple web servers, handle SSL termination, and cache static assets such as images and CSS files. Clients interact with the reverse proxy, which forwards requests to the appropriate backend server based on factors like load balancing algorithms or request routing rules.

In summary, forward proxies are used by clients to access resources on external networks, while reverse proxies are deployed by server operators to manage incoming client requests and optimize backend server infrastructure. Both types of proxies play crucial roles in enhancing security, performance, and scalability in networked environments.

A diagram of a computer network

Description automatically generated