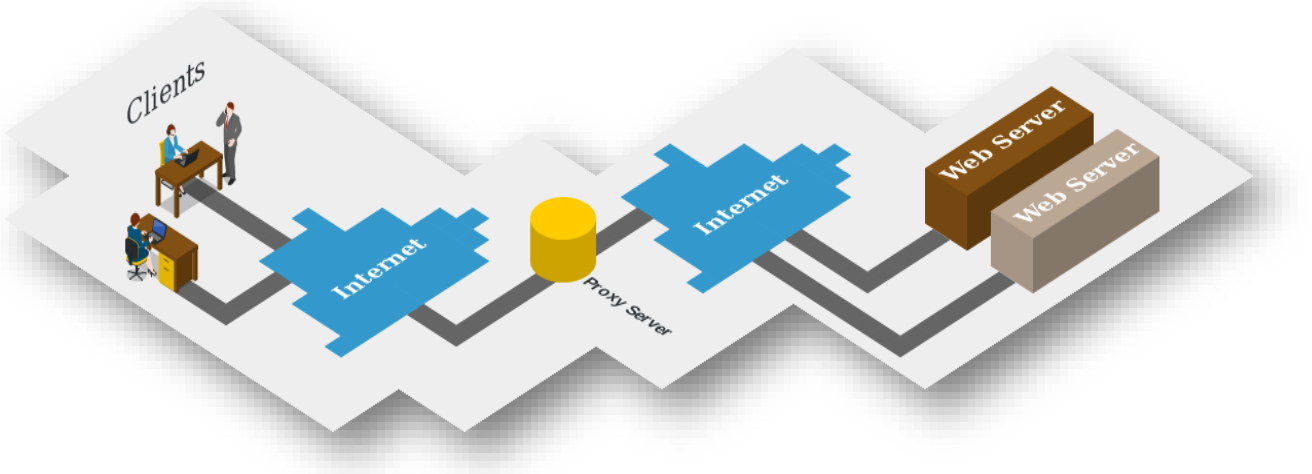
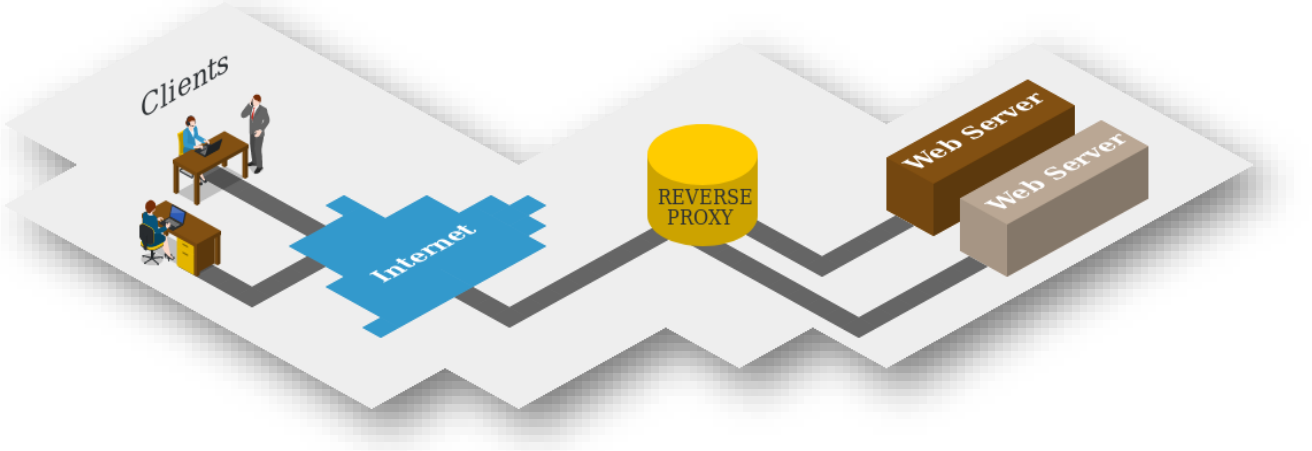
NGINX Reverse Proxy

پروکسی سرور یک سرور واسط است که تمامی درخواست ها را به مقاصد مختلف هدایت می کند. سرورهای مقصد، درخواست کننده ها را به صورت مستقیم نمی بینند .پس از دریافت یک درخواست، سرور پروکسی یک پورت به آن اختصاص می دهد و سپس درخواست را به مقصد ارسال می‌کند. پس از دریافت پاسخ، جواب درخواست از طریق همان اتصال قبلی به درخواست کننده ارسال می شود .



بر خلاف پروکسی سرور، یک پروکسی معکوس در کنار وب سرورها قرار می گیرد .درخواست کننده ها تمامی وب سرورها و سرویس های پشت پروکسی معکوس را به شکل یک سرور می بینند و توانایی تفکیک آنها را از هم ندارند.



پروکسی معکوس درخواست ها را به یکی از سرورها ارسال می کند .پاسخ دریافتی به گونه ای ارسال می شود که هیچ تغییری در آن صورت نپذیرد با این اختلاف که درخواست کننده متوجه ماهیت سرور پاسخ دهنده نخواهد شد.

کاربرد های آن:

۱- تقسیم بار (Load balancing)

پروکسی معکوس را می توان بعنوان یک تقسیم کننده بار در مقابل سرورها قرار داد؛ به گونه ای که حجم درخواست ها بین چندین سرور تقسیم شود .این امر باعث افزایش سرعت و قابلیت پاسخگویی و همچنین اطمینان از عدم اشغال کامل یک سرور می شود .اگر یکی از سرورها پاسخگو نباشد، بار آن بین سرورهای باقیمانده تقسیم می گردد .

۲- بهبود سرعت وب (Web acceleration)

پروکسی معکوس می تواند در خواست ها را فشرده کرده تا بار ترافیکی کاهش یابد .علاوه بر فشرده سازی برای کاهش بار می توان از قابلیت cache در پروکسی های معکوس هم استفاده نمود .

۳- امنیت و ناشناس بودن (Security and anonymity)

بحث امنیت و ناشناس ماندن یکی از اساسی ترین دلایل راه اندازی یک Proxy Server در شبکه است حالا اگر Reverse Proxy را بخواهیم به همین منظور در شبکه راه اندازی کنیم این نوید را به ما می دهد که یک مهاجم نتواند با دسترسی مستقیم به شبکه داخلی ما از ساختار و توپولوژی شبکه ما آگاه شود زیرا مهاجم با تعدادی سرور رو به رو است که توسط یک سرور که همان Reverse Proxy Server است درخواست هایش به آن سرور ها در شبکه داخلی هدایت می شود .که این خود عاملی برای ناشناس ماندن سرور ها و ساختار شبکه ما می شود. رمزنگاری SSL معمولا بر روی وب سرورها انجام نمی شود؛ می توان از پروکسی معکوس برای رمزنگاری تمامی درخواست ها استفاده کرد .با استفاده از این امکان، نیازی به تنظیم نمودن SSL بر روی هر سرور به صورت جداگانه نیست .

کانفیگ و راه اندازی اولیه:

پس از نصب و تنظیمات اولیه nginx و firewall سیستم، وارد فایل nginx.conf در آدرس اشاره شده می‌شویم:

etc/nginx/nginx.conf

در فایل کانفیگ باید تغییرات اشاره شده انجام شود تا پروکسی معکوس ایجاد شود و کار کند :

```
server {
    listen 80;<----- پورت دلخواه
    listen [::]:80; <--------- پورت دلخواه
    server_name  example.ettesal.local  example.com ; <------ هر دامینی که داریم
    location  / {
       proxy_pass http://192.168.1.196:80/; آدرس سرور جواب دهنده یا دامنه جایگزین انتخاب شده
       proxy_pass http://videocake.example.com
 }
    location /app {
          proxy_pass http://videocake.example.com/app/;
    }
}
```

با استفاده از این کد های اشاره شده می‌توان درخواست های دامنه اصلی را به دامنه دیگری منتقل کرد و یا حتی می‌توان با ایجاد یک وب سرور Apache و اشاره کردن به آن در فایل کانفیگ Nginx درخواست های سمت Back end را که شامل زبان PHP است را در Nginx با پروکسی معکوس پشتیبانی کرد و دسترسی به وب سرور Apache را از بیرون شبکه محدود کرد به طوری که دیگران سرور پاسخ دهنده را Nginx می‌بینند.

استفاده به عنوان load balancer :

ابتدا سرور های که بار باید بین آنها تقسیم شود را در همان فایل کانفیگ اصلی Nginx تعریف می‌کنیم:

```
upstream backend {
        server 192.168.1.196;
        server 192.168.1.197;
        server 192.168.1.198;
}
```

حال می‌توان در قسمت location که قبلا به آن اشاره شد استفاده کرد.

```
server {
    listen 80;
    location / {
      proxy_pass http://backend;
    }
```

استفاده از رمزنگاری SSL :

ابتدا فایل های certification را به صورت زیر تعریف می کنیم و سپس مشخصات سرور را ذکر می‌کنیم.

```
http {
    ssl_certificate     /root/certs/example.com/example.com.crt;
    ssl_certificate_key /root/certs/example.com/example.com.key;
    ssl_ciphers         EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH;
    ssl_protocols       TLSv1.1 TLSv1.2;
server {
    listen              192.168.1.196:443 ssl;
    server_name         example.com  www.example.com;
    root                /var/www/example1.com;
   }
}
```