۱)

زمانی استفاده میشود که شما در بازکردن یک سایت و یا هنگام ارتباط با یک سیستم در اینترنت مشکل دارید که با استفاده از این دستور میتوانید متوجه شوید که مشکل کجاست و این دستور میتواند نقشه مسیر ارتباطی از کامپیوتر شما تا وب سروری که وب سایت مد نظر شما در ان است را نشان دهد.این دستور در ویندوز و سیستم عامل های دیگر نیز قابل استفاده است و میتوان گفت که این دستور یک ابزار مفید برای عیب یابی و فهمیدن مشکل در اتصال به شبکه اینترنت است درست مثل ping.

این دستور شاملpacket loss &high latencyمیباشد

این دستور زمانی که به وب سایت متصل میشود ترافیک ارسالی از مسیرها و واسطه های مختلفی عبور میکند و به مقصد میرسد و نیز میتوانیم با این دستور میزان تاخیربه وجود امده در هر توقف را مشاهده کنیم اگر در رسیدن به وب سایت مشکل دارید اما مطمعن هستید که وب سایت به درستی کار میکند پس این مسیر مشکل دارد که با این دستور میتوان متوجه شد که مشکل کجاست

از نظر فنی این دستور یک ترتیب متوالی از بسته ها را با استفاده از پروتوکل icmpارسال میکند و هرکدام از این بسته ها مقداری رابررسی میکند و دارای زمان مشخص است اگر زمان هر بسته به صفر برسد روتر موردنظر رابرگشت داده و. پیغام خطا نمایش داده میشود و با ارسال بسته ها به این شیوهtracerouteمطمعن میشود که هر روتر در مسیرفعال است یا خیر.

این دستور را میتوان درcmdاجرا کردو بعد از باز کردن ان مقابل ان ادرس مقصد را وارد کنید. برای مثال اسکرین شات این دستور برای آدرس گوگل آمده است:

```
Tracing route to google.com [172.217.18.142]
over a maximum of 30 hops:

  1     1 ms    <1 ms     1 ms  192.168.1.1
  2     7 ms     7 ms     6 ms  85.15.16.164
  3     *        6 ms     6 ms  172.18.205.49
  4    38 ms   100 ms   100 ms  172.18.205.209
  5     9 ms     7 ms     7 ms  172.16.0.181
  6    10 ms     9 ms    10 ms  172.18.196.102
  7    10 ms     8 ms     9 ms  10.10.53.225
  8    98 ms   100 ms    99 ms  10.21.212.10
  9    12 ms     8 ms     7 ms  10.21.21.10
 10    36 ms    34 ms    34 ms  134.0.220.186
 11     *        *        *     Request timed out.
 12     *        *        *     Request timed out.
 13    37 ms    36 ms    38 ms  134.0.217.233
 14    37 ms    36 ms    36 ms  213.202.4.171
 15    37 ms    36 ms    36 ms  216.239.41.109
 16    65 ms   100 ms   100 ms  172.253.51.137
 17    38 ms    35 ms    36 ms  mct01s09-in-f14.1e100.net [172.217.18.142]

Trace complete.
```

در ویندوز باید از لغت tracertو در مک و لینوکس tracerouteاستفاده کرد

در خروجی این دستور اولین خط بیانگر روتر خود شما است و خط بعدی روترisp است و همینطور که به سمت پایین میروید بیانگر روترهای دورتر و موجود در این مسیر است. برای مثال ما اگر این دستور را برای سایت دیجیکالا نیز انجام دهیم دو خط اول و دوم یکسان هستند چون خط اول روتر خود ما و خط دوم روتر سرویس دهنده اینترنت ما که شاتل است میباشد.


قالب هر خط در این دستور بصورت زیر می باشد:

Hop RTT۱ RTT۲ RTT۳ Domain Or IP Address

عبارت : Hop هرگاه که یک پکت از یک روتر عبور می کند نشان دهنده یک Hop می باشد.

عبارات RTT۱ RTT۲ RTT۳ : زمان رفت و برگشتی است که یک پکت تا Hop مورد نظر طی می کند. اگر زمانی شما بجای مشاهده زمان مورد نظر علامت * را مشاهده نمودید، بدانید که هیچگونه پاسخی دریافت نکرده اید.

عبارت Domain Name یا IP Address : اگر Domain Name در دسترس باشد شما قادر خواهید بود که مکان روتر را متوجه شوید در غیر اینصورت فقط آدرس IP روتر مورد نظر به شما نشان داده می شود

۲)

یک api عبارت است از مجموعه ای از قواعد که از طریق ان اپلیکشن های مختلف با هم ارتباط برقرار میکنند و یعنی دو موجودیت مختلف را به همدیگر ربط میدهد و نیز میتوانید داده ها را با فرمت مناسب به خروجی بفرستند و ان را برگشت دهد و لغتrestبه معنی نمایش اطلاعات برای کاربران از راهی که خوانایی بالایی داشته باشدو یک پروتوکل نیست و این تنها یک راه حل و یا یک سبک معماری برای نوشتنapiاستو restیک روش معماری است وrestful مفسری برای ان است مثلا اگر شما یک سرور دارید و قسمتbackend یکrest apiاست ولی اگر کاربر از قسمتclient side یک درخواست برای استفاده ازapiبکند کاربرشماrestful خواهد بود

وRESTFULشامل ۴ عملیات است

۱. دریافت داده از یک فرمت مناسب
۲. ایجاد داده جدید
۳. تغییرو بروزرسانی داده
۴. حذف کردن داده

وrest مبتی برhttp است  و هر کدام از عملیات بالا حاوی متدhttp منحصر به خودشان است

متدی برای دریافت اطلاعاتgetاست

متدی برای ایجاد دادهpost است

متدی برای بروزرسانی و ایجادتغییرات درداده‌putاست

وdelete متدی برای حذف است

و این ۴ عملیت داده های ما را مدیریت میکنند

RESTیک رابط برای مدیریت درخواست‌ها و ارتباط با بانک اطلاعاتی دارد که می‌شود در جدول زیر آن را به صورت کلی مشاهده کرد:

| | | | | |
|---|---|---|---|---|
| HTTP | POST | GET | PUT | DELETE |
| SQL | INSERT | SELECT | UPDATE | DELETE |
| CRUD | CREATE | READ | UPDATE | DELETE |

تمام درخواست‌ها، حاوی جواب‌های منحصر به فرد خودشان هستند. این جواب‌ها از طریق یکسری کد ارائه می‌شوند که تعداد آن‌ها بسیار زیاد است. اما می‌شود آن‌ها را در ۵ کلاس دسته بندی کرد. عدد اول هر کلاس نمایانگر پیغامی است که زیرکلاس‌های آن ارائه می‌دهند:

۱;xx - informational

۲;xx - success

۳;xx - redirection

۴;xx - client error

۵.xx - server error

باید توجه کرد که اگر در طراحیrestful دقت نکنید هزینه های جانبی زیادی دارد که اکثر ان ها مربوط به هزینه اعمال تغییرات است  برای طراحی بهتر باید بدانیم که طراحی یک سرویس خوب  علاوه بر داشتن فن به هنر نیز نیازمند است و وب سرویس باید به صورت درست و به جا از استانداردهای وب استفاده کند و نیز برنامه نویس به راحتی بتواند از ان استفاده کند و ساده باشد و به کارایی و انعطاف پذیر بودن دقت شود و برای اینکه مطمئن شوید restful برای شما مناسب است یا خیر باید موارد زیر را در نظر بگیرید:

- ✓ کلاینت-سرور: این محدودیت بیان میکند که سرور و کلاینت باید جداگانه در نظر گرفته شوند.
- ✓ محدودیت :Statelessاین محدودیت یعنی اینکه restful مستقل از حالت است و همه درخواست ها جداگانه بررسی میشوند و هر درخواست شامل اطلاعات کافی برای پردازش کامل به صورت تنهایی میباشد
- ✓ کش: چونکه مستقل از حالت است پس داده های قابل کش را باید ذخیره کند
- ✓ اینترفیس واحد

باید تمامی محدودیت ها و نکات مورد توجه قرار گیرد.

۳)

وب سرویس یک سیستم نرم افزاری برای پشتیبانی از تعامل دستگاه و ماشین در شبکه است. یا به عبارت دیگر یک سیستم از سیستم دیگر سرویس میگیرد. وب سرویس ها اجزای یک نرم افزار هستند و از طریق پروتکل ارتباط برقرار میکنند. باید دقت داشت که وب سرویس ها توسط اپلیکیشن های دیگر قابل استفاده هستند و XML و HTTP پلت فرم اولیه وب سرویس ها هستند. وب سرویس ها دو کلاس کلی دارند:

۱. سازگار به REST
۲. وب سرویس arbitrary

در معماری وب سرویس سه نقش اساسی وجود دارد:

۱. ارائه دهنده خدمت: این نقش خدمات وب را ارائه میدهد. ارائه دهنده خدمات این سرویس را پیاده سازی کرده و آن را در دسترس قرار میدهد.
۲. درخواست کننده خدمت: که این نقش مصرف کننده وب سرویس است. این کار با استفاده از باز کردن اتصال شبکه و ارسال درخواست XML برای استفاده از وب سرویس موجود انجام میشود.
۳. ثبت خدمات: این نقش یک فهرست خدمات ارائه میدهد. این نقش مکانی در اختیار قرار میدهد تا توسعه دهندگان بتوانند خدمات جدید ایجاد و توسعه دهند و استفاده کننده گان بتوانند خدمات مورد نیاز خود را پیدا کنند.

NGINX یک وب سرویس است که در حال حاضر بیش از ۲۵٪ دامنه های فعال را میزبانی میکند. نقش آن متعادل کردن کردن بارگزاری load balancer وب سرور و HTTP Cash است. این وب سرویس از همان ابتدا برروی استفاده بهینه از رم و کارایی بیشتر تمرکز کرد. از این وب سرور به عنوان پروکسی معکوس با وب سرور Apache استفاده کرد که در نتیجه آن شما قدرت آپاچی و سرعت انجین ایکس را خواهید داشت.

قابلیت های NGNIX در ادامه آمده است:

- ✓ قابلیت پشتیبانی بیش از ۱۰۰۰۰ اتصال همزمان و مصرف رم بسیار پایین
- ✓ پشتیبانی از SSL
- ✓ سازگار با IPv۶
- ✓ قابلیت Load Balancing
- ✓ و...

باید دقت کرد که هدف NGINX و آپاچی باهم متفاوت است. هدف اصلی NGINX ایجاد سریعترین وب سرور در دنیای اینترنت بود. NGINX به طور مداوم Apache و سایر وب سرورها را در معیارهای اندازه گیری عملکرد وب سرور به چالش می‌کشاند. رویکرد آپاچی فرآیندمحور است و برای هر درخواست یک ترد ایجاد میکند در حالی که معماری انجین ایکس مبتنی بر رویداد است. علت اصلی این امر توانایی در پردازش چندین درخواست در یک ترد است.

Apache یک نرم افزار وب سرور منبع باز و رایگان است که حدود ۴۶ درصد از وب سایت های سراسر جهان را پشتیبانی می کند. نام رسمی وب سرور آپاچی Apache HTTP Server است و توسط Apache Software توسعه یافته است. صاحبان وب سایت ها با استفاده از آپاچی به بازدیدکنندگان سایت خود محتوای مورد نظرشان را ارائه میدهند. همه ما Apache را یک وب سرور می نامیم، آپاچی یک سرور فیزیکی

نیست، بلکه یک نرم افزار است که بر روی سرور اجرا می شود. کار اصلی آن ایجاد ارتباط بین مرورگر بازدید کننده صفحه(کروم یا...) و سرور است . هنگامی که شما میخواهید یک صفحه را بازدید کنید درخواستی از سمت مرورگر شما به سرور میرود و آپاچی با تمام فایل هایی که درخواست شده است به آن پاسخ میدهد. آپاچی ساختاری مبتنی بر ماژول دارد و ماژول هایی برای امنیت، ذخیره سازی، تایید اعتبار رمز عبود و ... در آن وجود دارد.

Apache Tomcat یک نرم افزار اپن سورس است که توسط Apache ساخته شده است. هدف ایجاد این وب سرویس فراهم آوردن اجرای servletها است. حال باید توجه داشت که به زبان ساده servlet یک فناوری سمت سرور است که به رسیدگی به درخواست http و پاسخ مشتری کمک می کند. مطالعات نشان داده اند که بیش از ۶۰ درصد برنامه های جاوا از این وب سرویس استفاده میکنند. این وب سرویس برای اپلیکیشن های تحت وبی است که به زبان جاوا نوشته شده اند که تمامی ویژگی های JavaEE را نیاز ندارند اما به یک ابزار مطمئن نیاز دارند. از دیدگاه دیگری آپاچی تامکت وظیفه دارد یک محیط runtime برای servlet فراهم کند که در این محیط فرد میتواند کد جاوا را اجرا کند. حجم کمی دارد یعنی از منابع کمی استفاده میکند و از SSL پشتیبانی میکند. تامکت وظایف زیر را دارد:

۱. به همه درخواست های دریافتی از مشتری پاسخ دهد.
۲. کلاس های مربوط به servlet های مربوطه را با استفاده از نقشه برداری servlet (از فایل web.xml) بارگیری کند تا درخواست های مشتری ورودی دریافت شود.
۳. کلاس servlet را اجرا کند.
۴. در آخر کلاس servlet را خالی کند.

Netty فریم وورک سرور کلاینت است که پیاده سازی آسان اپلیکیشن های شبکه همانند پروتکل سرور و کلاینت را ممکن میسازد. وظیفه اصلی آن ساده سازی برنامه نویسی شبکه مانند TCP و UDP است. از پروتکل SSL پشتیبانی میکند.

در حقیقت نتی non-blocking I/O کلاینت-سرور فریم وورک است که برای توسعه اپلیکیشن های شبکه جاوایی مورد استفاده قرار میگیرد

(۴

مرورگرها و دنیای وب به طوری طراحی شده اند که قانونی به نام same-origin policy را اجرا می کنند. این قانون می گوید مقصد درخواست های ما باید با منبعی که به آن هستیم یکی باشد. زمانی مقصد درخواست و منبع آن یکی نیست که:

- ✓ به یک دامنه و زیر دامنه دیگر متصل شویم.
- ✓ به یک پروتکل دیگر متصل شویم.
- ✓ به یک پورت دیگر متصل شویم.

به طور مثال تا زمانی که به وب سایت mydomain.com متصل باشیم هیچ اشکالی ندارد که به منابع آن دسترسی داشته باشیم (مثلا mydomain.com/doc۱) اما نمی توانیم به منابع وب سایت دیگری مانند anotherdomain.com دسترسی داشته باشیم. دلیل وجود این قانون جلوگیری از هک شدن است؛ برای مثال فرض کنید هکری لینک مخربی را برای شما ارسال کرده باشد. شما با کلیک روی این لینک به وب سایت هکر منتقل می شوید که در پشت صحنه یک آی فریم از وب سایت بانکی شما باز میشود و هکر با استفاده از کوکی هایی که در مرورگر شما است، شما را وارد حسابتان می کند. طبیعتا این مسئله یک رخنه امنیتی بزرگ است و نباید مجاز باشد. قانون same-originبرای جلوگیری از چنین حملاتی ایجاد شده است و می گوید که دسترسی به resource ها باید از همان origin ای باشد که به آن متصل هستیم

کورس دو بخش دارد؛ کورس در سمت کلاینت و کورس در سمت سرور.

کورس در سمت کلاینت:

منظور ما از سمت کلاینت همان front-end یا مرورگر است. زمانی که وارد دنیای مرورگرها و front-end می شویم باید بدانید که قانون same-origin policy فقط مربوط به اسکریپت ها می شود اما مرورگر ها آن را بسط دادند تا شامل درخواست های جاوا اسکریپتی نیز بشود بنابراین به صورت پیش فرض ما فقط می توانیم به منابعی دسترسی داشته باشیم که از same-origin (از سایتی که در آن هستیم) باشند. مساله اینجاست که ما بعضی اوقات میخواهیم ارتباط با منابع دیگر برقرار کنیم که اینجا سیاست same origin به مشکل میخورد. برای حل این مساله CORS معرفی شد که مخفف Cross-origin resource sharing یا به اشتراک گذاری منابع از چند سورس مختلف است.

User agent ها (مثلا یک مرورگر یا کتابخانه axios یا هر agent دیگری که از آن برای اتصال به یک سایت استفاده شده است) می توانند از مکانیسم خاصی به نام CORS استفاده کنند تا نوع خاصی از درخواست های cross-origin را ارسال کنند. این عملیات با استفاده از تنظیم header های خاصی در درخواست HTTP شما انجام می شود. زمانی که می خواهیم یک درخواست Cross-origin (درخواستی به منبع دیگر) را ارسال کنیم، مرورگر شما یک header خاص به نام Origin را به درخواست HTTP اضافه می کند. مقدار این header برابر با منبعی است که درخواست از آن ارسال شده است. سرور این درخواست را دریافت کرده و header هایش را بررسی می کند تا بداند هر درخواست از کجا آمده است.

کورس در سمت سرور:

اگر شما توسعه دهنده سرور هستید باید header های مجموعه Access-Control را به پاسخ های سرور خود اضافه کنید. مرورگرها بر اساس مقدار این دسته از header ها می توانند مشخص کنند که چه نوع درخواست های cross-origin ای مجاز هستند. در صورتی که Access-Control در header شما نباشد، تمام درخواست های cross-origin بلوکه خواهند شد. تعداد header های مربوط به Access-Control بسیار زیاد است اما مرورگر فقط یک header خاص را برای مجاز دانستن درخواست های cross-origin می خواهد و آن هم Access-Control-Allow-Origin می باشد. مقداری که به این header می دهید مشخص خواهد کرد که چه مقصد هایی می توانند به این منبع (سرور ما) درخواست ارسال کنند. به طور مثال اگر قرار است وب سایت mydomain.com به سرور و وب سایت ما (مثلا https://api.mywebsite.com) دسترسی داشته باشد باید آن را به header ذکر شده پاس بدهیم. حالا اگر درخواستی از سمت https://mywebsite.com ارسال شود می تواند به منابع موجود در https://api.mywebsite.com نیز دسترسی داشته باشد. از این به بعد مرورگر مقدار موجود در Access-Control-Allow-Origin را با مقصد درخواست شما مقایسه می کند و اگر این دو یکی نباشند اجازه ارسال درخواست را نخواهید داشت.

۵)

اگر کانفیگ های کورس به درستی نصب نشده باشند، کنسول browser خطای Cross-Origin Request Blocked نمایش داده میشود که برای اصلاح آن باید مطمئن شد که API هدرهای مناسب ارسال میکند. برای حل این مساله با سرچی در اینترنت متوجه میشویم که اکثر سایت ها توصیه میکنند عبارت Access-Control-Allow-Origin: * را به هدر خود اضافه کنید که در نتیجه آن همه به API شما میتوانند دسترسی داشته باشند و این خود یک حفره امنیتی است. در راه درست برای آن وجود دارد:

۱. از مدیر API بخواهید که موارد را بررسی کرده و هدر ها را اصلاح کند
۲. یک middleware بسازید. چون که کورس از طریق هدر ارتباط برقرار میکند بنابراین شما میتوانید یک جزئی شبیه پروکسی ایجاد کنید که کار آن ایجاد درخواست و گرفتن پاسخ از API است. این هدرها را به بالای کد اضافه کرده و به UI خود ارسال کنید بنابراین شما مستقیماً به API متصل نمیشوید.

۶)

زمانی که شما درخواستی در وب میفرستید این درخواست ابتدا به پروکسی سرور میرود و پروکسی سرور از سمت شما درخواست را به فضای اینترنت میفرستد، پاسخ ها را جمع آوری میکند و به شما پاسخ میدهد و اطلاعات دریافت شده که معمولاً صفحات وب هستند به شما میفرستد که در نتیجه این کار امنیت و حریم شخصی شما حفظ میگردد؛ به این فرآیند پروکسی کردن درخواست میگویند. در حقیقت هنگام بازدید یک سایت شما مستقیماً به وب سایت اطلاعاتی در مورد خودتون میفرستید، اطلاعاتی از قبیل کامپیوتر مورد استفاده، موقعیت مکانی، آدرس آی پی و... در پاسخ به این اطلاعات ارسالی منابع وب سایت(مانند عکس ها، متن ها) به شما نمایش داده میشوند برخلاف این حالت در ارتباط از طریق پروکسی کردن درخواست پروکسی امکان تغییر و پنهان کردن اطلاعات شما را دارد همچنین میتواند اطلاعاتی دریافتی از وب سایت را فیلتر کند به همین خاطر یکی از کاربردهای پروکسی برای محدود کردن دسترسی کودکان به اینترنت است.

سرور پروکسی، به عنوان یک دروازه ای بین شما و دنیای اینترنت می باشد. این سرور، به عنوان یک سرور واسطه، کاربران را از وبسایت هایی که در حال مرور آن ها هستند، جدا می کند. Proxy Server ها سطوح مختلفی از عملکرد، امنیت و حریم خصوصی را بسته به نوع مصرف، نیاز و یا خط مشی های سازمان ارائه می دهند.

اگر از proxy server استفاده می کنید، ترافیک اینترنت از طریق آن (سرور پروکسی) در مسیر خود به طرف آدرس درخواستی شما، جریان می یابد. سپس درخواست از طریق همان سرور پراکسی بر می گردد (در مورد این قانون، استثناهایی وجود دارد)، سپس سرور پراکسی، داده های دریافت شده از وبسایت ها را برای شما، ارسال می کند. یک Proxy Server در واقع، رایانه ای متصل به اینترنت است که آدرس آی پی مخصوص به خودش را دارد و کامپیوتر شما از آن اطلاع دارد. زمانی که شما یک درخواست برای وب ارسال می کنید، این درخواست، ابتدا به سرور پروکسی می رود. سپس سرور پروکسی درخواست شما را از طریق وب انجام می دهد، پاسخ را از سرور وب جمع آوری می کند و داده های وب پیج را برای شما ارسال می کند و شما می توانید آن پیج را در برازر خود ببینید. زمانی که یک سرور پروکسی، وب درخواستی را برای شما ارسال می کند، می تواند تغییراتی را در داده های ارسال شده ایجاد کند، با این وجود، هنوز هم اطلاعاتی که انتظارش را دارید را دریافت می کنید. یک proxy server می تواند آدرس آی پی شما را تغییر دهد، بنابراین سرور وب نمی داند که شما دقیقا در کجای دنیا قرار دارید. همچنین می تواند داده های شما را رمزگذاری کند، بنابراین، اطلاعات شما در فرایند ارسال، قابل خواندن نمی باشند. و در آخر، یک پروکسی سرور می تواند دسترسی به برخی صفحات پیج را بر اساس IP address مسدود کند.

۷)

کلمه ی reserve proxy دقیقا برعکس proxyاست یعنی درخواست های کلاینت ها را از محیط نت دریافت میکند و به سرور های موردنظر شبکه داخلی هدایت میکند که به انinbound proxy گفته میشود زیرا درخواست ها را از سمت بیرون شبکه داخلی دریافت میکند

سروری است که پشت فایروال قرار میگیرد ودرخواست ها را از شبکه اینترنت دریافت کرده و ان ها را به سرور های موجود درشبکه داخلی ارسال میکند و این سرورهای موجود در شبکه داخلی به صورت مستقیم از شبکه نت قابل دسترسی نمیباشند و کاربران درخواست کننده سرویس از طرف نت از طرف reserve proxy احراز هویت میشوند

۱. استفاده از این سه فایده داارد:. مورد اولload balancikg است که با استفاده از این است که درخواست هایی که به سمت وب سرور موجود درشبکه داخلی هدایت میشوند توسط reserve proxyدریافت میشوند و درخواست به سمت سروری کهloadکاری کمتری دارد ارسال میشود و کارایی شبکه ما نیز بالاتر میرود

۲. Web acceleration : سرور های در نقش Reverse Proxy به خوبی انجام Caching اطلاعات می توانند داده های ورودی و خروجی را نیز فشرده سازی کنند که این کار سرعت جریان ترافیک ورودی و خروجی شبکه را بین کلاینت و سرور افزایش میدهد. همچنین اینکه می توانند عمل رمزنگاری اطلاعات را بوسیله پروتکل SSL انجام دهند و در نتیجه بار کاری حاصل از انجام عمل رمزنگاری از روی وب سرور ها برداشته می شود.

۳. 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

پروکسی معمولی از کلاینت محافظت میکند در حالی که رزرو پروکسی از سرور محافظت میکند. شما از طریق این پروکسی میتوانید حمله های به سرور خود را کم کنید. همانند پروکسی مستقیم که یک نقطه برای دسترسی و کنترل فراهم میکند این پروکسی نیز همین کار را میکند. شما میتوانید ترافیک و درخواست های ورودی به سرور را کنترل کنید. این پروکسی ها به عنوان Load Balancer نیز عمل میکنند. در حقیقت در اینجا درخواست های ورودی را به صورت کلاستر دسته بندی میکند و به سرورهای مختلف که یک سرویس را ارائه میدهند ارسال میکند یعنی اگر تعداد درخواست های ورودی به سرور زیاد است از این پروکسی باید استفاده کرد.