

**مستندات نهایی پروژه مقایسه و remote access در node red**

استاد مربوطه :

دکتر مینا ملک زاده

گردآوری شده توسط :

پوریا آزاد و شایان نیکروش

خرداد

1402

## فهرست

[فهرست 2](#_Toc138622136)

[نصب و راه اندازی node-red 3](#_Toc138622137)

[connection-oriented و connection-less در TCP 6](#_Toc138622138)

[connection-oriented و connection-less در UDP 10](#_Toc138622139)

[Reliable in TCP 12](#_Toc138622140)

[Reliable in UDP 17](#_Toc138622141)

[پیاده سازی Acknowledge 21](#_Toc138622142)

[تست سرعت TCP و UDP 25](#_Toc138622143)

[Fragmentation for TCP 29](#_Toc138622144)

[Remote 36](#_Toc138622145)

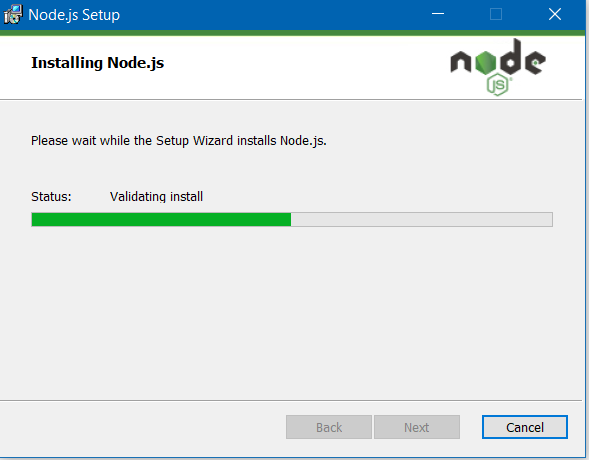
## نصب و راه اندازی node-red

1. نصب Node Js

A screenshot of a computer

Description automatically generated

ابتدا باید نسخه LTS Node Js رو از سایت رسمی دانلود کنیم و نصب کنیم

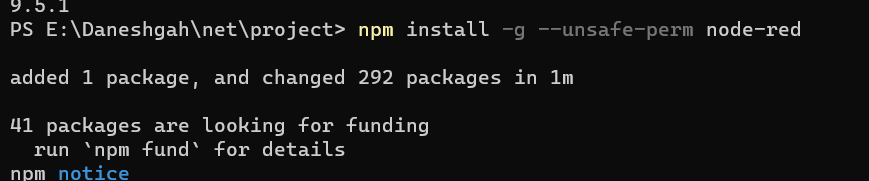


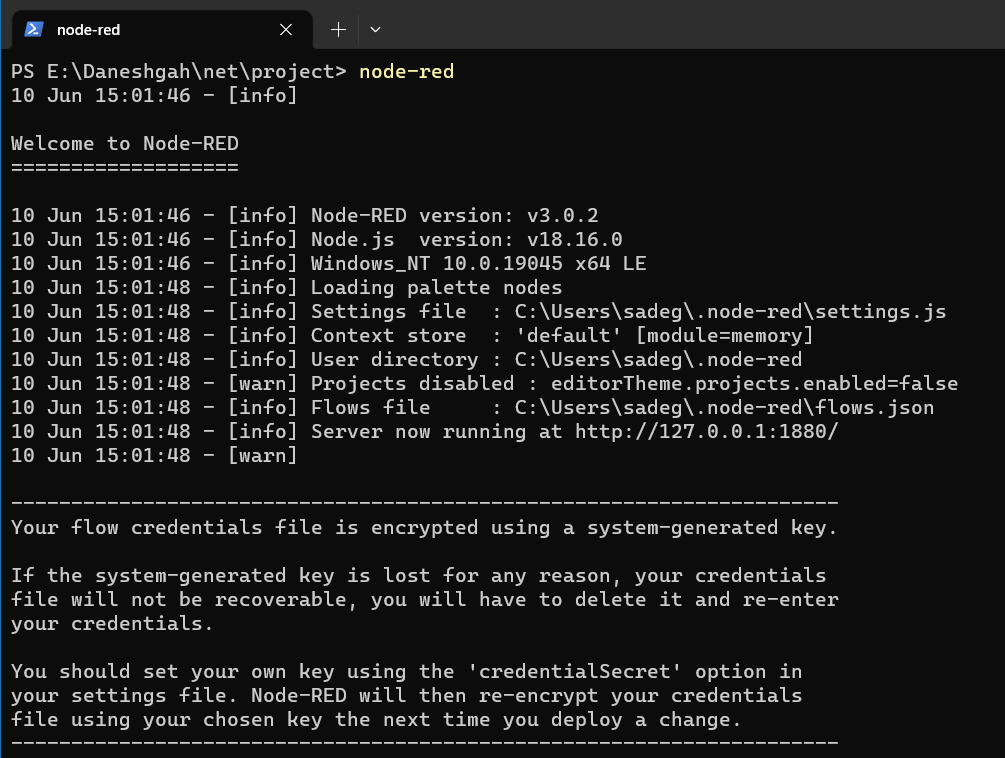
اگه به درستی انجام شده بود با کامند های زیر در power shell باید ورژن node js رو بده

A picture containing text, screenshot, font, black

Description automatically generated

حالا با وارد کردن کامند زیر میتونیم node-red رو نصب کنیم



در اخر با وارد کردن node-red ، node-red بر روی لوکال هاست اجرا میشه

نکته) تاوقتی که میخواهیم با node-red کار کنیم نباید power shell یا CMD رو ببندیم

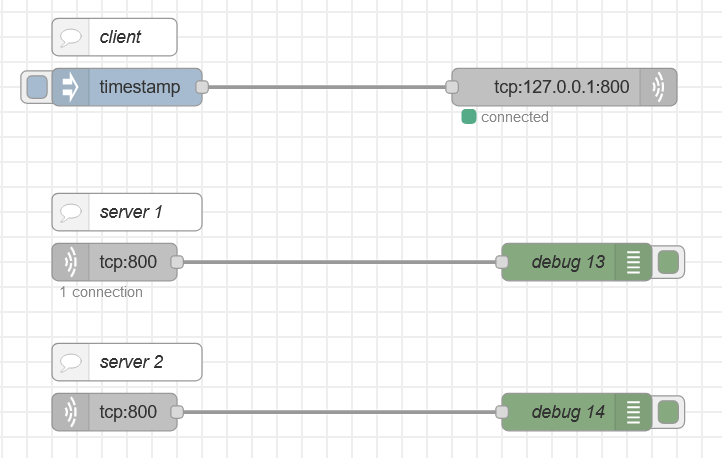
## ***connection-oriented و connection-less در TCP***

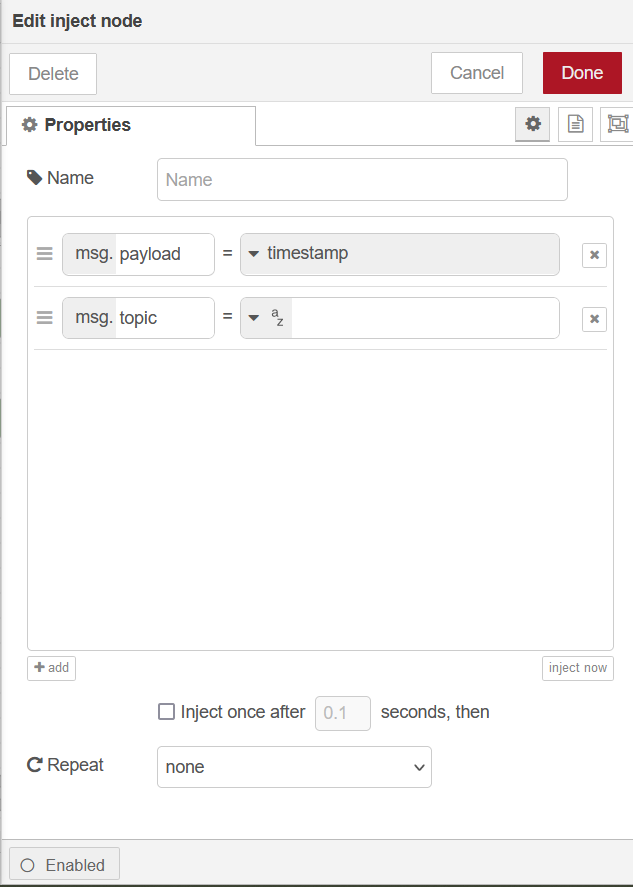
Connection-oriented و Connection-less دو روش مختلف برای ارسال داده‌ها در شبکه‌های کامپیوتری هستند.

در Connection-oriented، ابتدا یک ارتباط بین دو دستگاه برقرار می‌شود. این ارتباط توسط یک پروتکل مانند TCP مدیریت می‌شود و اطمینان حاصل می‌شود که داده‌ها به ترتیب درستی و بدون از دست دادن به مقصد می‌رسند. در این روش، قبل از ارسال داده‌ها، یک دستور handshake بین دو دستگاه انجام می‌شود تا اطمینان حاصل شود که ارتباط بین دو دستگاه برقرار شده است. این روش برای ارسال داده‌های حساس و مهم مانند پرداخت‌های آنلاین و انتقال فایل‌های بزرگ به کار می‌رود.

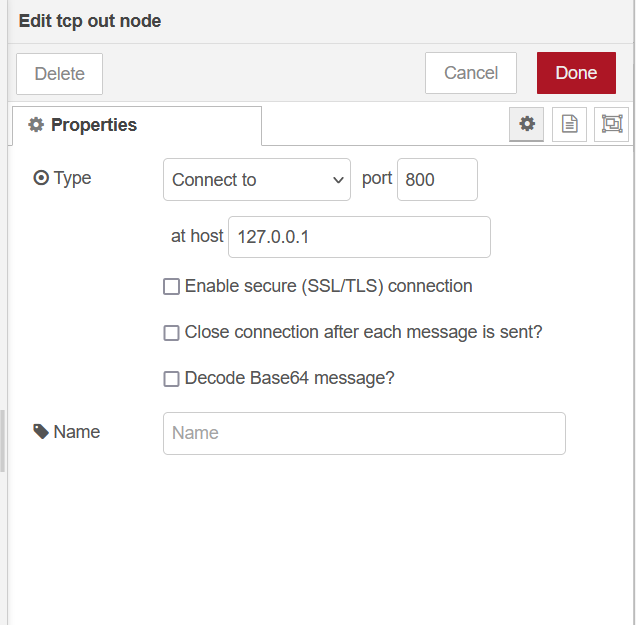
در Connection-less ، هیچ ارتباطی بین دو دستگاه برقرار نمی‌شود و داده‌ها بدون هیچ گونه تضمینی برای رسیدن به مقصد ارسال می‌شوند. در این روش، داده‌ها به صورت پیام‌های مستقل ارسال می‌شوند و هیچ گونه تضمینی برای رسیدن به مقصد وجود ندارد. این روش برای ارسال داده‌هایی که نیاز به سرعت و کارایی دارند مانند تصاویر و ویدئوها به کار می‌رود.

برای مقایسه نحوه کار connection-oriented و connection-less ابتدا چارت زیر را برای tcp پیاده سازی میکنیم

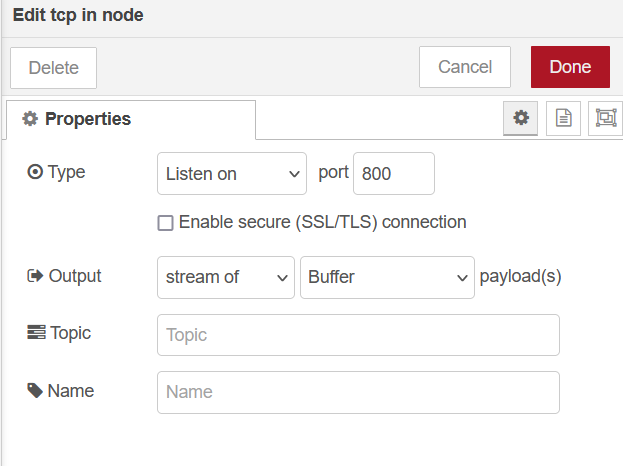




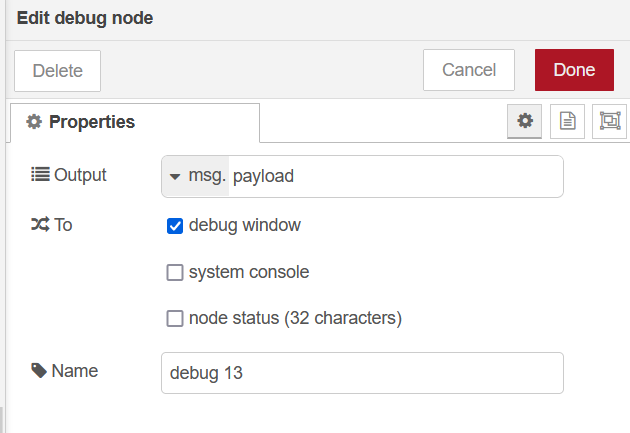
تنظیمات inject node



تنظیمات tcp out

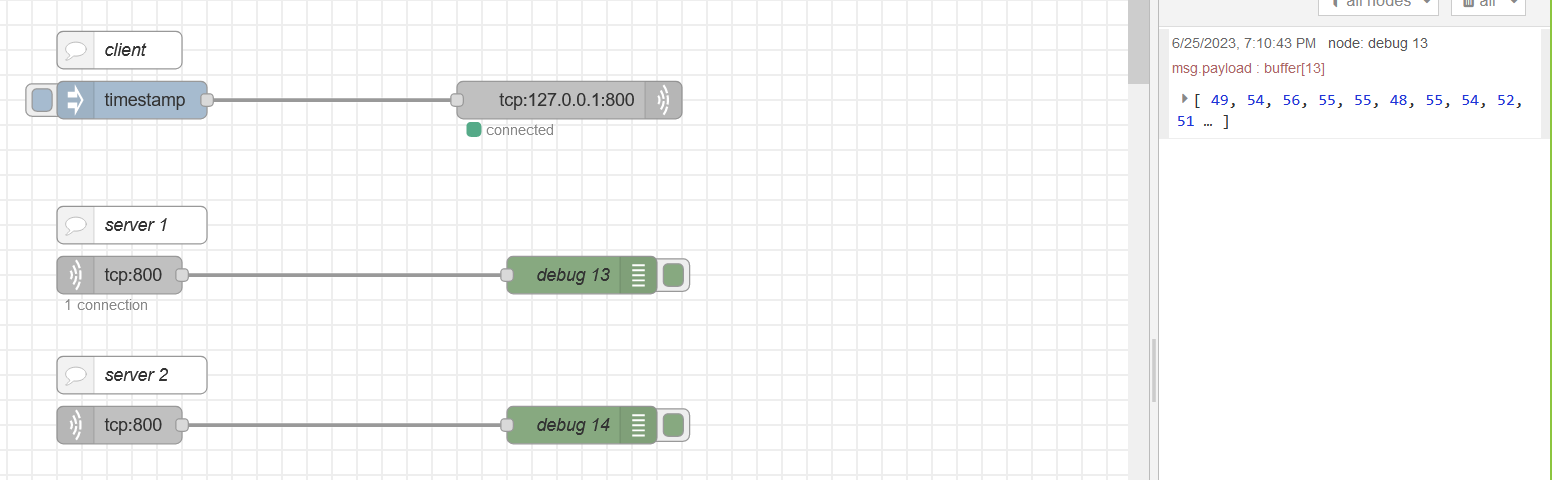


تنظیمات tcp in



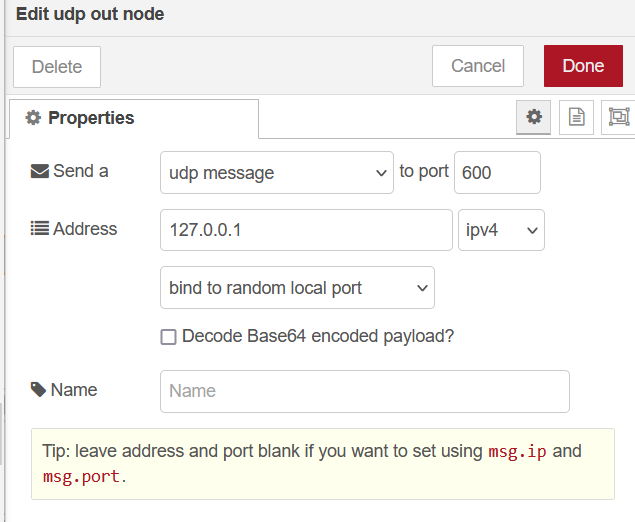
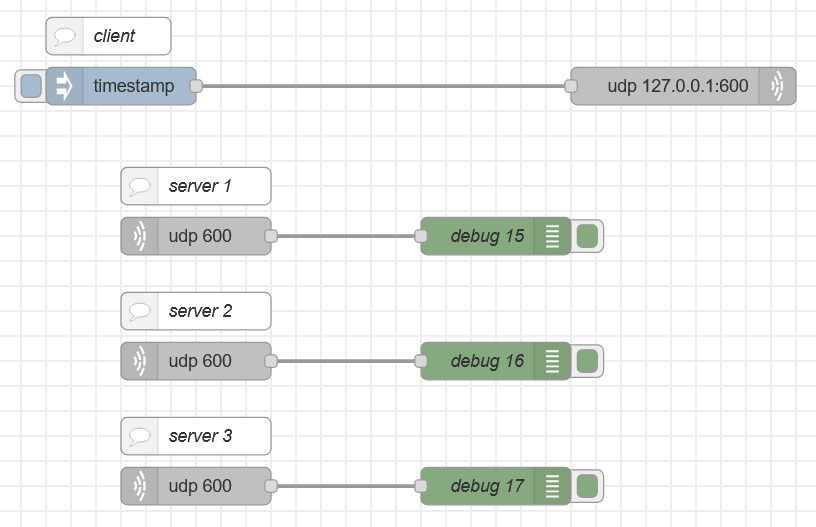
تنظیمات debug node

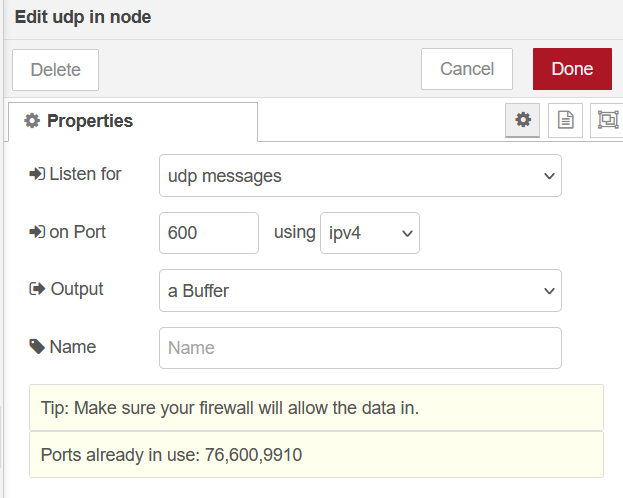
حال بعد از ارسال یک timestamp میبینیم که فقط به یکی از server 1 ها متصل شده است



## ***connection-oriented و connection-less در UDP***

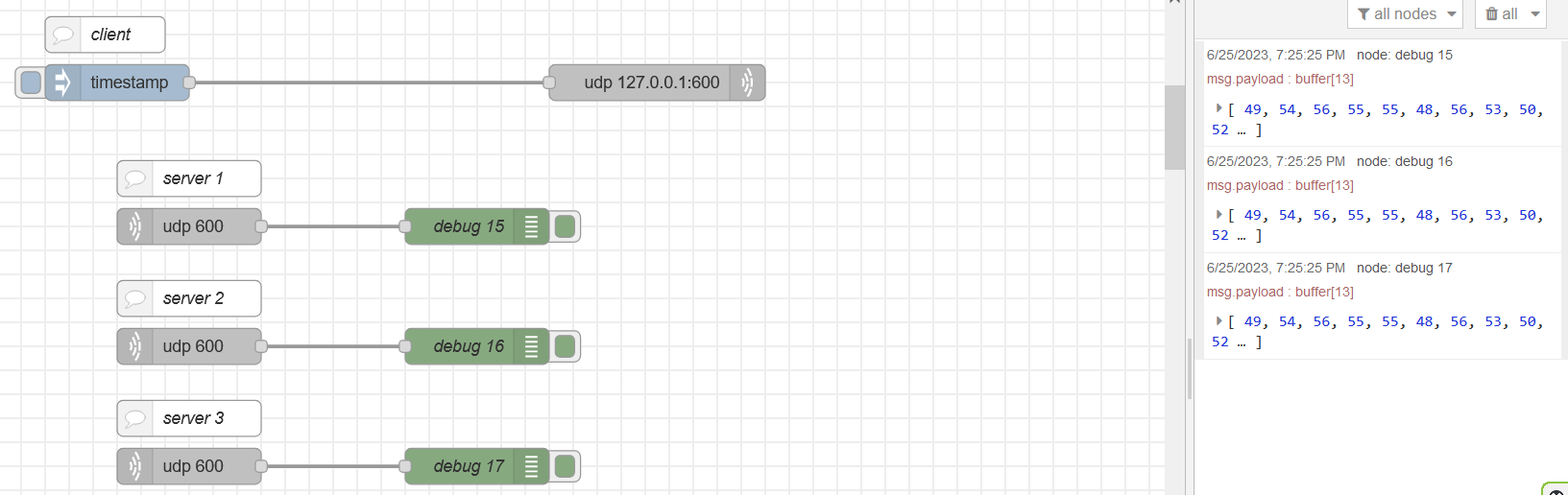
برای مقایسه نحوه کار connection-oriented و connection-less در udpچارت زیر را پیاده سازی میکنیم

تنظیمات udp out



تنظیمات udp in

دو udp in دیگر به شکل بالا پیاده سازی میکنیم



در اینجا میبینیم که client زمانی که یک timestamp برای پورتی خاص میفرستیم سه سرور جواب ما را میدهند و ارتباط چک نشده است به عبارتی هر سروری که بتواند جواب ما را میدهد اگر هم چند سرور بتوانند همه شان جواب مارا میدهد

## ***Reliable in TCP***

مفهوم Reliable به معنای قابل اطمینان بودن یک پروتکل یا خدمت است. در شبکه‌های کامپیوتری، reliable بودن یک پروتکل به معنای تضمین این است که داده‌ها به صورت کامل، در ترتیب درست و بدون خطا از یک محل به مقصد منتقل شوند.

برای داشتن یک پروتکل reliable، به چند ویژگی اساسی نیاز است:

1- تأیید دریافت: هنگامی که یک بسته داده ارسال می شود، باید تأیید دریافت آن بسته از سمت مقصد دریافت شود. این اطمینان می دهد که بسته داده به درستی به مقصد رسیده است.

2- دریافت داده ها به ترتیب درست: اگر بسته های داده به ترتیب درست دریافت نشوند، ممکن است باعث ایجاد خطا و اشتباه در داده های دریافت شده شود.

3- بازبینی خطا: در صورت بروز خطا در ارسال و دریافت داده ها، باید قابلیت بازبینی و اصلاح خطا وجود داشته باشد تا داده های درست دریافت شوند.

4- کنترل جریان: باید از طریق کنترل جریان اطمینان حاصل شود که مقصد قادر به پردازش و دریافت داده ها است.

5- تحمل از از دست دادن بسته ها: در شبکه های پرترافیک، ممکن است بسته ها گم شوند. پروتکل باید قابلیت تحمل از از دست دادن بسته ها را داشته باشد و بازسازی داده ها را به درستی انجام دهد.

با توجه به این ویژگی ها، پروتکل TCP یک پروتکل reliable است که برای ارسال داده ها در شبکه استفاده می شود. این پروتکل اطمینان حاصل می کند که داده ها به ترتیب درستی دریافت می شوند و هر بسته داده ای که ارسال می شود دریافت شده است.

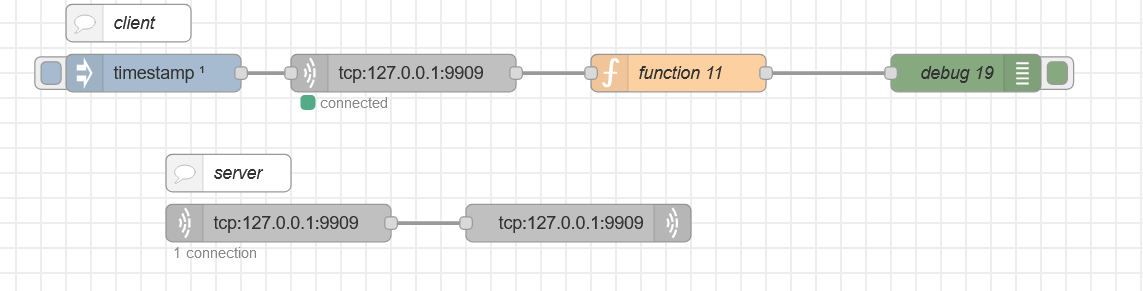
UDP یک پروتکل ناقص (unreliable) است که برای انتقال داده ها در شبکه استفاده می شود. در UDP، داده ها به دو صورت بدون تأیید دریافت (unacknowledged) و بدون ترتیب (unordered) ارسال می شوند، به عبارت دیگر، این پروتکل به صورت ناقص و سریع عمل می کند.

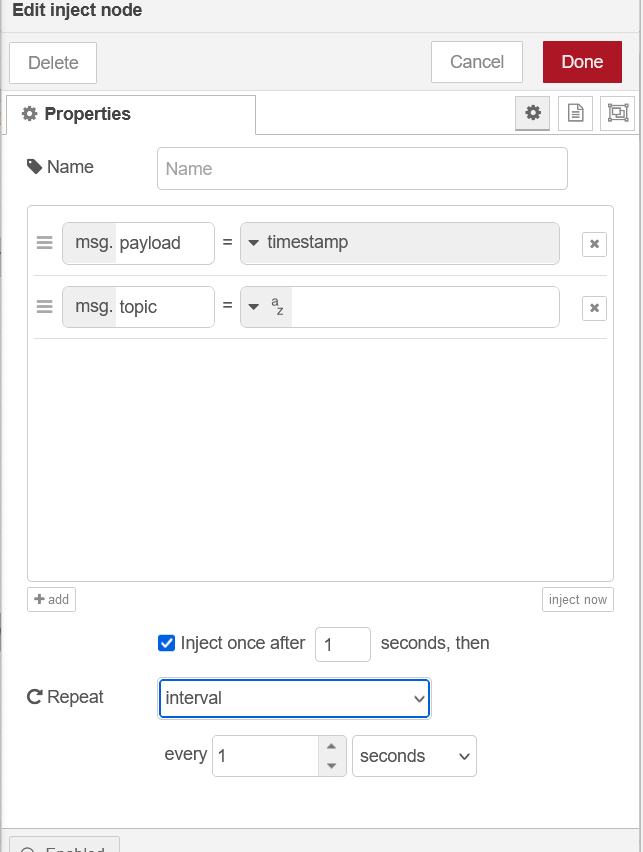
از این رو، در UDP، برخلاف TCP، هیچ گونه مکانیزم تأیید دریافت (ACK) یا بازبینی خطا (error checking) وجود ندارد. به همین دلیل، UDP به عنوان یک پروتکل unreliable شناخته می شود.

با این حال، در برخی از کاربردهای شبکه، ممکن است که نیاز به سرعت بالا و حجم کم داده ها باشد و مهم نباشد که بعضی از داده ها گم شوند. به عنوان مثال، در برخی از بازی های آنلاین، قابلیت تحمل از از دست دادن برخی از بسته های داده به عنوان یک ویژگی مطلوب شناخته می شود.

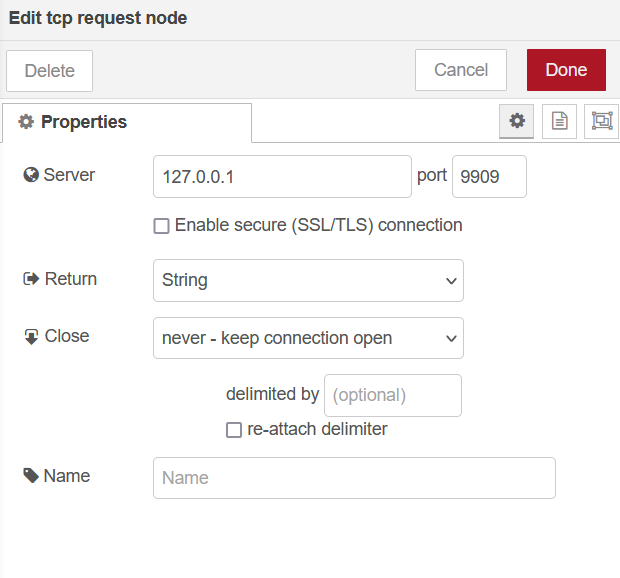
در نتیجه، برای کاربردهایی که نیاز به سرعت بالا و حجم کم داده ها دارند و از اهمیت کمتری برخوردارند، می توان از UDP به جای TCP استفاده کرد. اما برای کاربردهایی که دقت و اطمینان از رسیدن داده ها بسیار مهم هستند، بهتر است از پروتکل reliable مانند TCP استفاده شود.

برای پیاده سازی ابتدا چارت زیر را برای tcp پیاده میکنیم

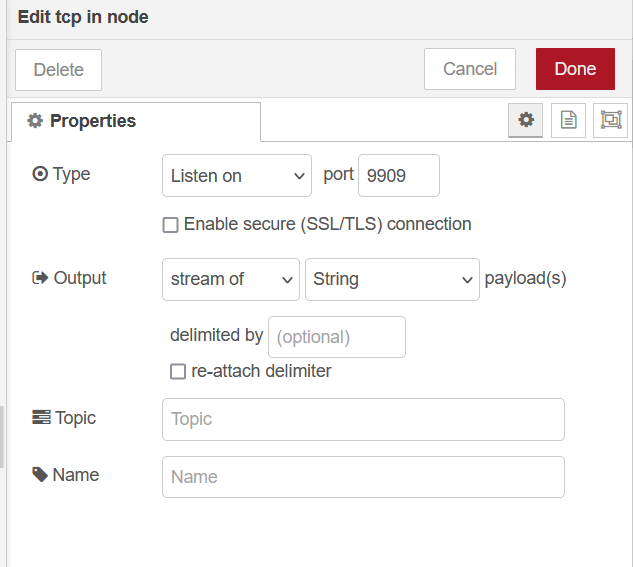




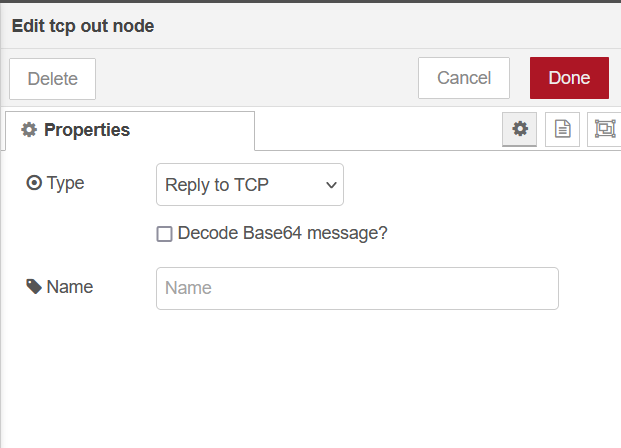
تنظیمات inject



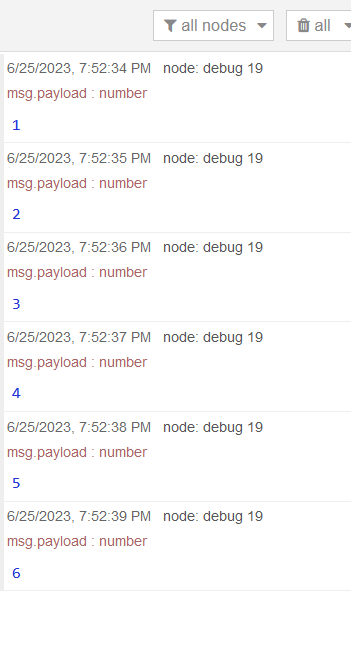
تنظیمات tcp request



تنظیمات tcp in

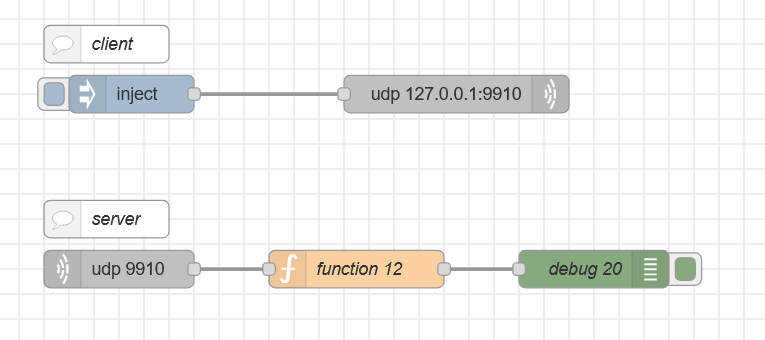


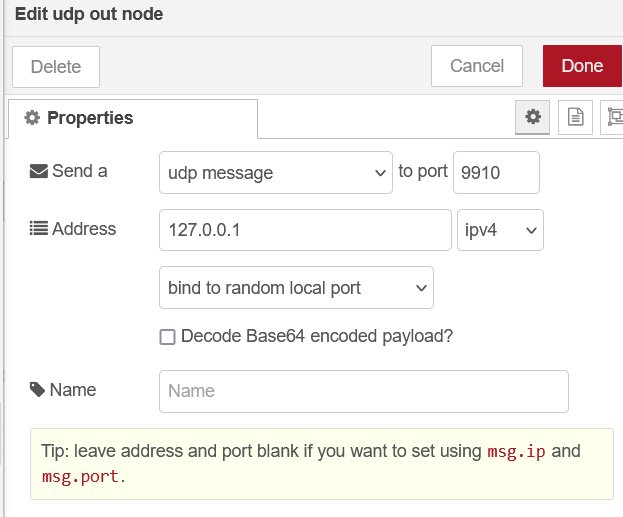
تنظیمات tcp out



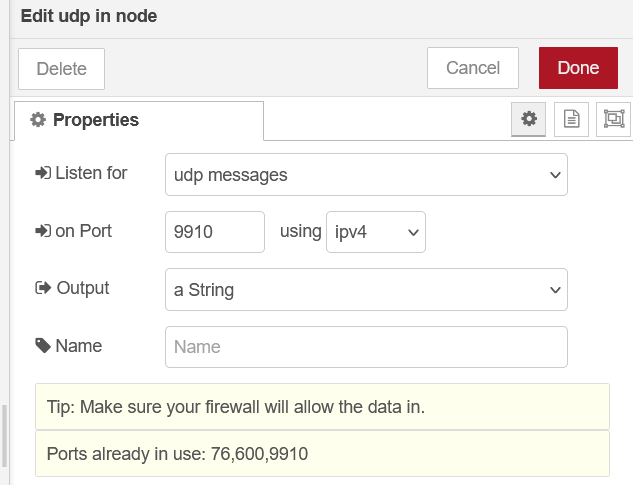
در خروجی شماره پاکت ها را بترتیب میبینیم

## ***Reliable in UDP***

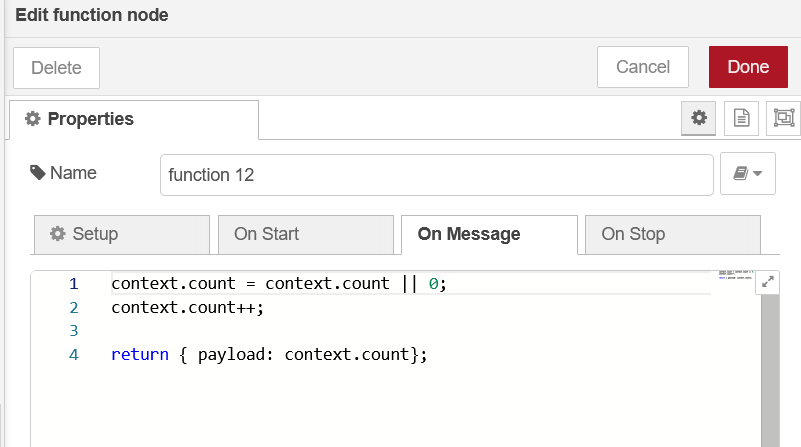
در مرحله بعد چارت udp را بصورت زیر پیاده سازی میکنیم



تنظیمات udp out



تنظیمات udp in



تنظیمات function



به ظاهر که دو پروتکل tcp و udp به خوبی عمل میکنند و هیچ پاکت گمشده ای نداریم چون در لوکال هاست خودمان این را اجرا میکنیم اما اگر در محیط اینترنت این را اجرا کنیم چون پروتکل udp conection-oriented نیست پس ممکن است پاکت هایی به مقصد نرسند و در نتیجه tcp reliable هست

## 

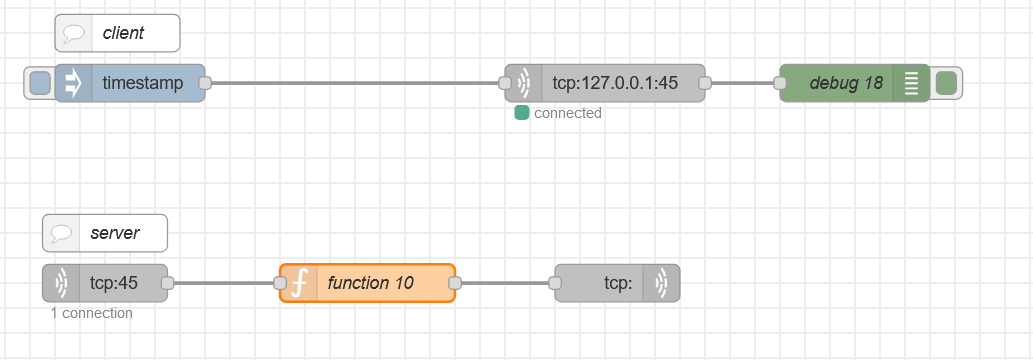
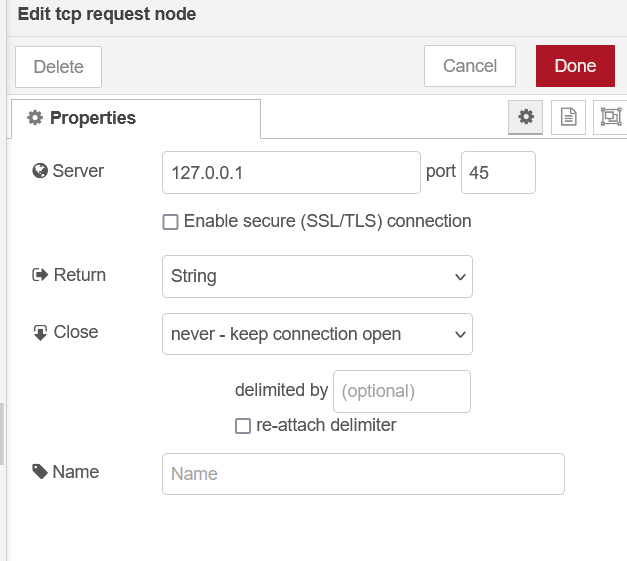
## پیاده سازی Acknowledge

در پروتکل TCP، برای اطمینان از دریافت درست و کامل داده‌ها، از مکانیزم تأیید دریافت (Acknowledgment یا به اختصار ACK) استفاده می‌شود. هر بسته‌ای که ارسال می‌شود، باید توسط دستگاه دریافت کننده تأیید دریافت شود. این تأیید دریافت، به فرستنده برای ادامه ارسال داده‌ها، اطمینان می‌دهد که بسته‌های قبلی به درستی دریافت شده‌اند و در صورت بروز خطا در ارسال بسته‌ها، فرستنده می‌تواند بسته‌های از دست رفته را مجدداً ارسال کند.

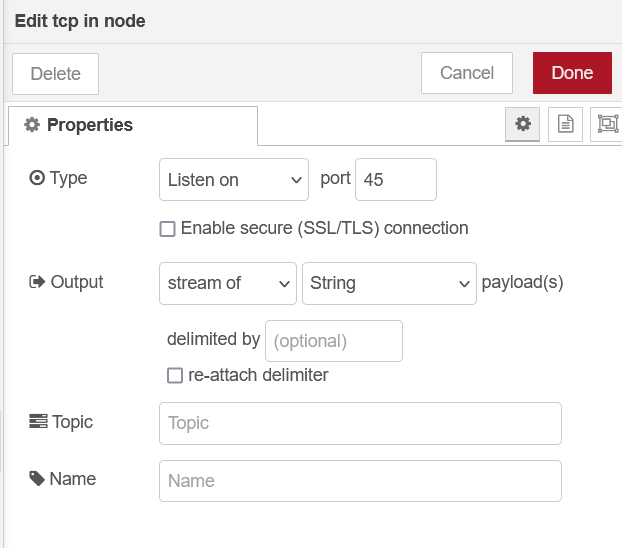
مکانیزم ACK در پروتکل TCP به این صورت عمل می‌کند که هر بسته‌ای که دریافت می‌شود، با شماره ترتیبی (Sequence number) بسته‌ها که در بسته‌ها قرار دارد، به فرستنده پاسخ داده می‌شود. در واقع، در هر بسته ACK که به فرستنده ارسال می‌شود، شماره ترتیبی آخرین بسته‌ای که به درستی دریافت شده، قرار دارد. به عنوان مثال، اگر دستگاه دریافت کننده بسته‌های با شماره ترتیبی 1، 2 و 3 را دریافت کرده باشد، باید یک ACK با شماره ترتیبی 3 به فرستنده ارسال کند تا فرستنده بداند که بسته‌های 1 تا 3 درست و کامل دریافت شده‌اند و می‌تواند بسته‌های بعدی را ارسال کند.

به طور کلی، مکانیزم ACK در پروتکل TCP برای اطمینان از دریافت درست و کامل داده‌ها، و همچنین کنترل خطا و بازیابی خطا در ارسال داده‌ها، بسیار مهم است.

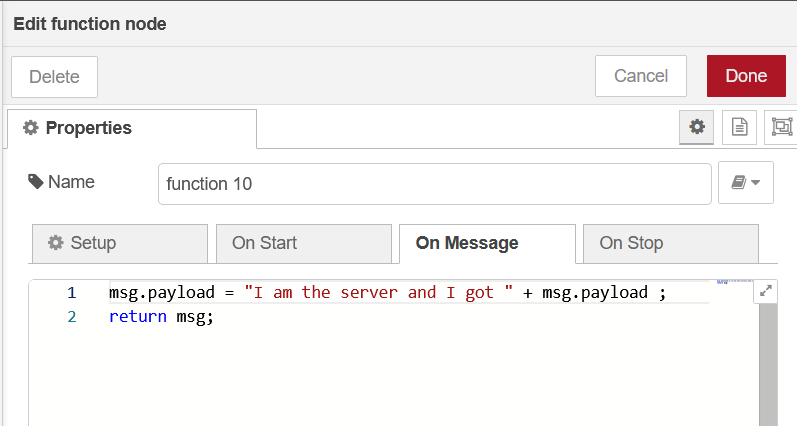
برای پیاده سازی acknowledge بدین صورت عمل میکنیم



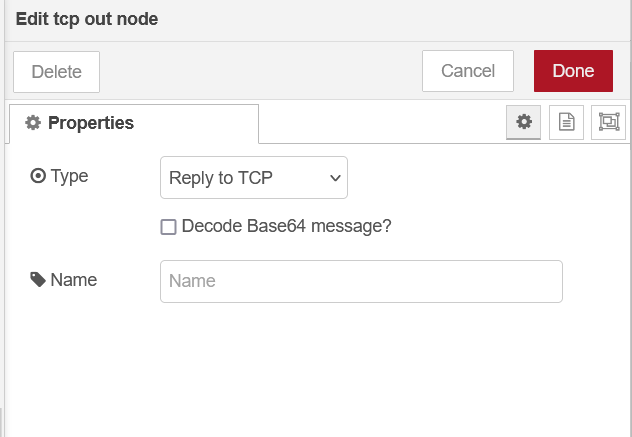
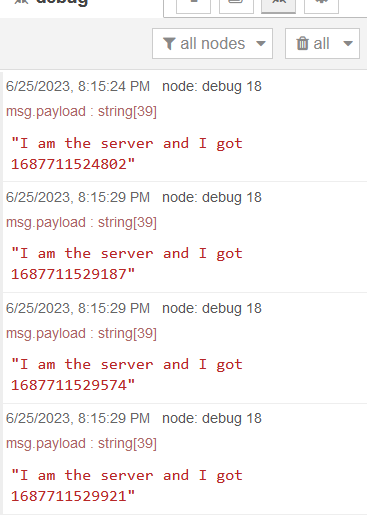
تنظیمات tcp request



تنظیمات tcp in



تنظیمات function

تنظیمات tcp out

و بدین صورت بعد از هر دریافت پیام از طرف سرور یک تایید به همراه timestamp دریافتی را میگیریم

## ***تست سرعت TCP و UDP***

علت تفاوت سرعت ارسال داده در پروتکل TCP و UDP، به دو عامل اصلی بستگی دارد:

1- مکانیزم تأیید دریافت:

در پروتکل TCP، برای هر بسته‌ای که ارسال می‌شود، باید تأیید دریافت آن بسته نیز به فرستنده ارسال شود. این مکانیزم باعث افزایش تعداد بارهای ارتباطی بین دو دستگاه و کاهش سرعت ارسال داده می‌شود. در مقابل، در پروتکل UDP، بدون تأیید دریافت، بسته‌ها به صورت پیوسته و بدون انتظار برای تأیید دریافت بسته‌ها به سمت مقصد ارسال می‌شوند و بنابراین سرعت بیشتری دارند.

2- کنترل خطا:

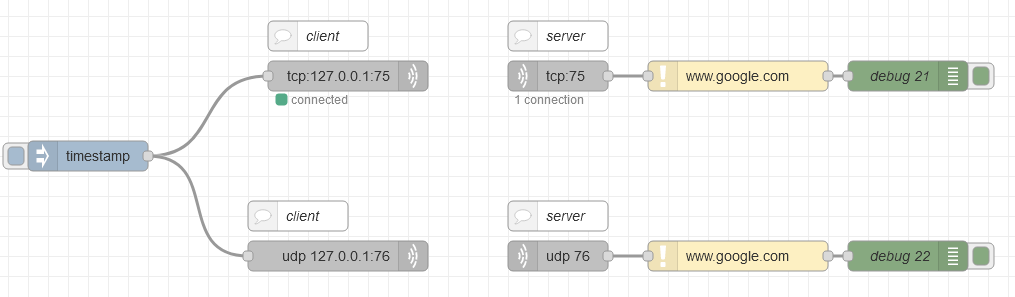
در پروتکل TCP، برای کنترل خطا در ارسال داده، از یک مکانیزم به نام روش بازیابی خطا (error recovery) استفاده می‌شود. در این روش، اگر بسته‌ای در ارتباط از دست برود، فرستنده مجدداً آن بسته را ارسال می‌کند. این کار باعث افزایش تعداد بارهای ارتباطی می‌شود و در نتیجه سرعت ارسال داده کاهش پیدا می‌کند. در پروتکل UDP، به دلیل عدم وجود مکانیزم بازیابی خطا، در صورتی که بسته‌ای در ارتباط از دست برود یا به صورت ناقص دریافت شود، این مشکل را نمی‌توان به صورت خودکار برطرف کرد و باید در سطح برنامه‌نویسی این مشکل را مدیریت کرد. در نتیجه، پروتکل UDP برای داده‌هایی که سرعت و کیفیت در انتقال آن‌ها مهم است، مانند داده‌های صوتی و تصویری، مناسب است.

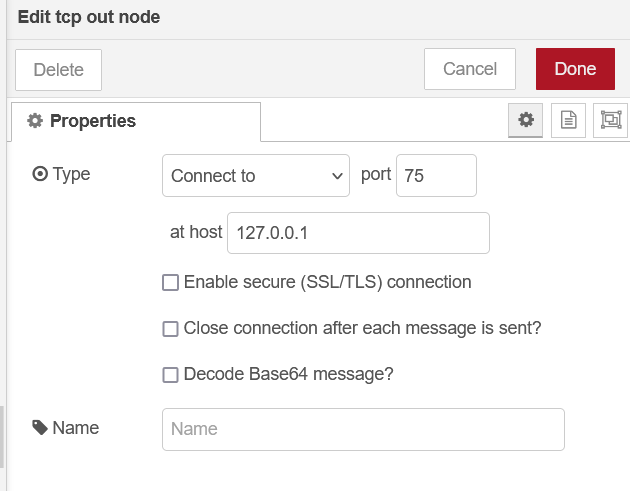
برای تست سرعت این دو پروتکل کافی است که از adv ping برای اینکار استفاده نماییم

ابتدا با استفاده از دستور زیر در cmd یک Palette نصب میکنیم

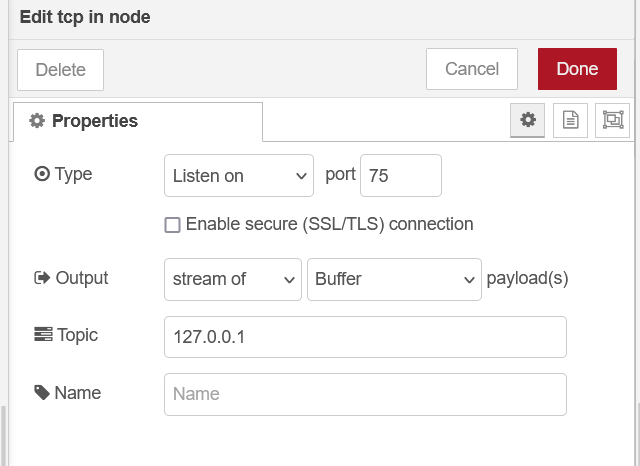
npm install node-red-contrib-advanced-ping

حال کافیست چارت زیر را پیاده سازی کنیم

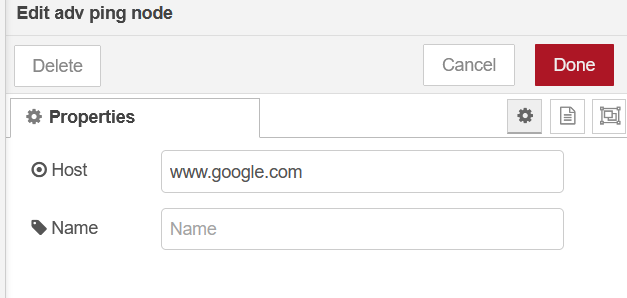




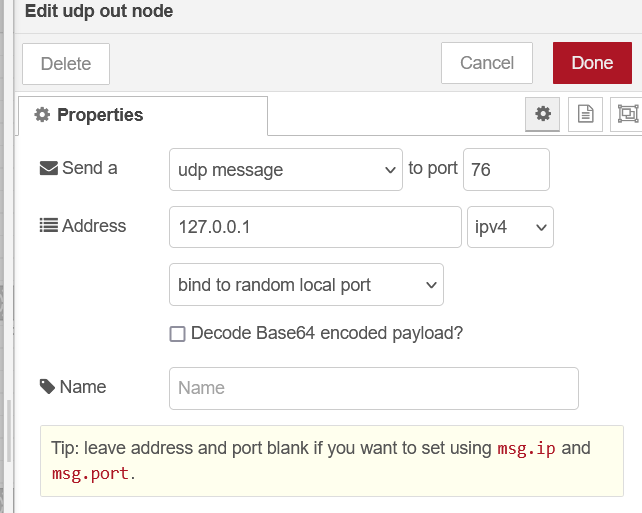
تنظیمات tcp out



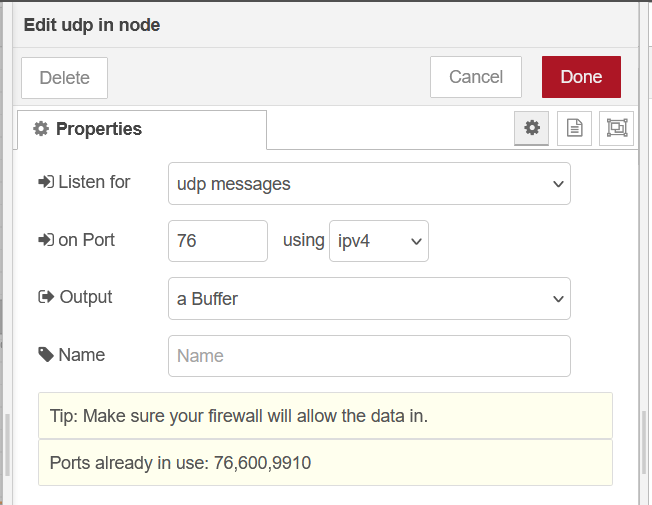
تنظیمات tcp in



تنظیمات adv ping

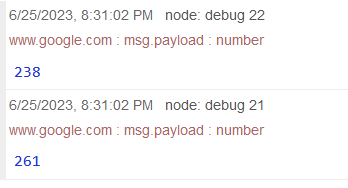


تنظیمات udp out



تنظیمات udp in

برای تنظیمات adv ping هم مثل قبلی عمل میکنیم



حال میبینیم که پینگ پایینی که مربوط به tcp ماست کندتر در آمد در صورتی که بالایی که مربوط به udp است کمتر در آمد و در نتیجه udp سریعتر است

## ***Fragmentation for TCP***

فرگمنتاسیون یک فرایند در شبکه‌های کامپیوتری است که در آن داده‌ها به بخش‌های کوچکتر، به نام فرگمنت‌ها، تقسیم می‌شوند تا بتوانند بهتر در شبکه منتقل شوند.

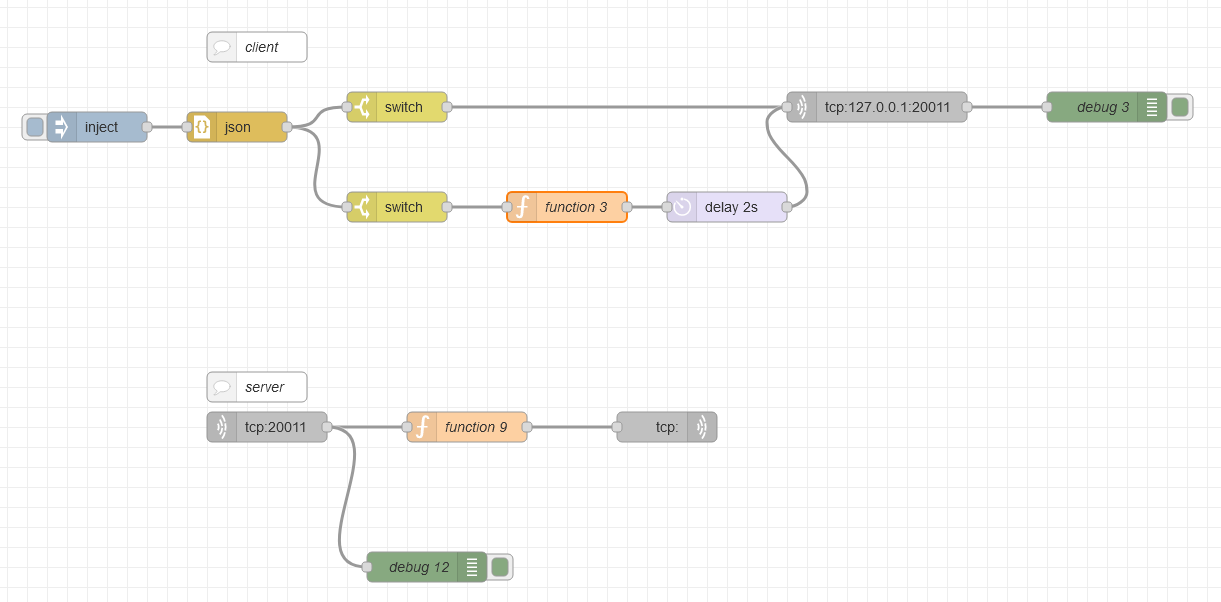
در پروتکل TCP، داده‌های ارسالی به بخش‌های کوچکتر تقسیم شده و به صورت جداگانه از هم به سمت مقصد ارسال می‌شوند. این فرایند به دلیل محدودیت اندازه بسته‌های شبکه و برخی محدودیت‌های فیزیکی شبکه، مانند اندازه حداکثر فریم در لایه دو، انجام می‌شود.

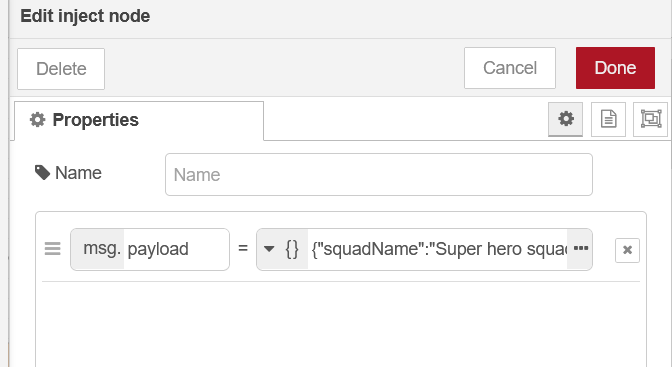
با استفاده از فرگمنتاسیون، بسته‌های بزرگ تر، مانند فایل‌های بزرگ، می‌توانند به چند بخش کوچکتر تقسیم شوند و به صورت جداگانه از هم ارسال شوند. همچنین، به کمک فرگمنتاسیون می‌توان بسته‌هایی که در شبکه از بین می‌روند یا به صورت ناقص دریافت می‌شوند، دوباره ساخته و از سمت مقصد دریافت کرد.

فرایند فرگمنتاسیون در TCP به این صورت انجام می‌شود که ابتدا بسته‌هایی به اندازه مشخصی به نام Maximum Segment Size (MSS) تقسیم می‌شوند. MSS بیانگر حداکثر اندازه داده‌ای است که می‌توان در یک بسته TCP ارسال کرد، بدون این که بسته‌ای بزرگ‌تر از حد مجاز شبکه ایجاد شود. سپس، هر یک از بسته‌های ایجاد شده به یک Header جدید از پروتکل TCP اضافه می‌شود که شامل اطلاعاتی مانند شماره دنباله بایت، شماره پنجره گیرنده و شماره تأیید است. در نهایت، هر بسته با یک شماره دنباله بایت و شماره تأیید مشخص به سمت مقصد ارسال می‌شود.

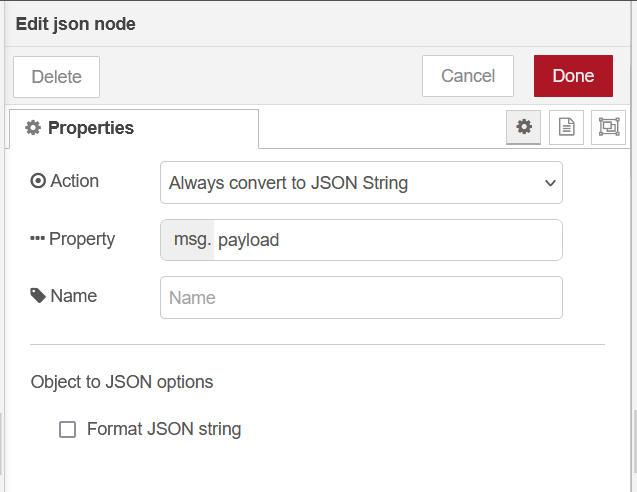
در سمت مقصد، بسته‌های دریافت شده توسط پروتکل TCP مجدداً ترکیب می‌شوند تا داده اصلی بازسازی شود. این کار با استفاده از شماره دنباله بایت و شماره تأیید پروتکل TCP انجام می‌شود. در صورتی که یک بسته در شبکه از بین رود یا به صورت ناقص دریافت شود، پروتکل TCP تلاش می‌کند برای دوباره ارسال آن بسته تا داده‌ها به طور کامل و سالم دریافت شوند.

برای پیاده سازی فرگمنتیشن tcp در نود رد میتوانیم بدین صورت عمل کنیم که ابتدا با استفاده از قطعه سوییچ اندازه قطعه را مقایسه میکنیم اگر درست بود که آن را مستقیم ارسال میکنیم و اگر از یک حدی بیشتر بود آن را به تابعی میفرستیم که در آن جا اندازه پکت رو به بخش های کوچک تری تقسیم میکند و آن را به خروجی ارسال میکند چارت زیر نحوه پیاده سازی آن است

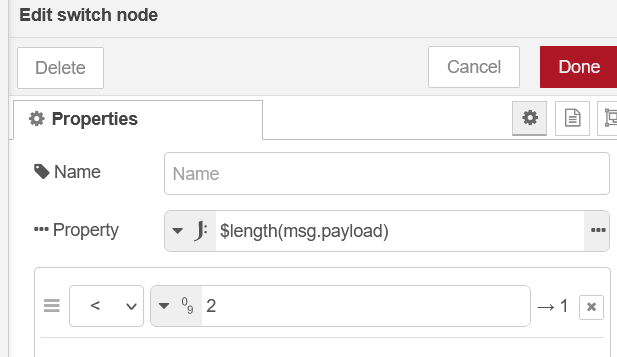




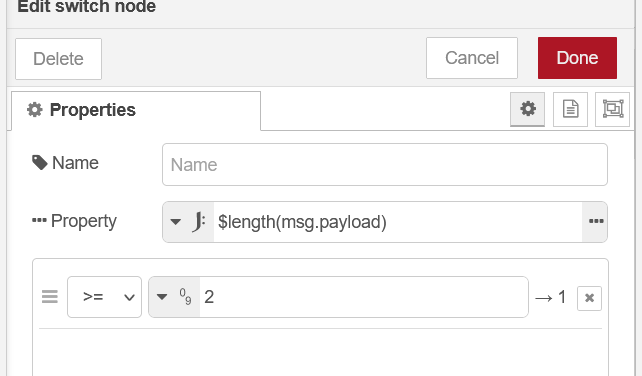
تنظیمات inject که در payload آن یک فایل json نوشته ایم



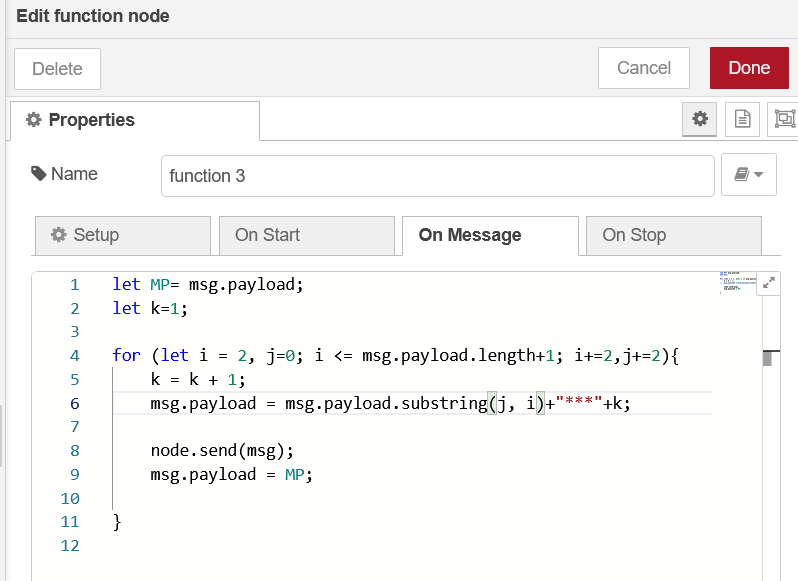
تنظیمات json



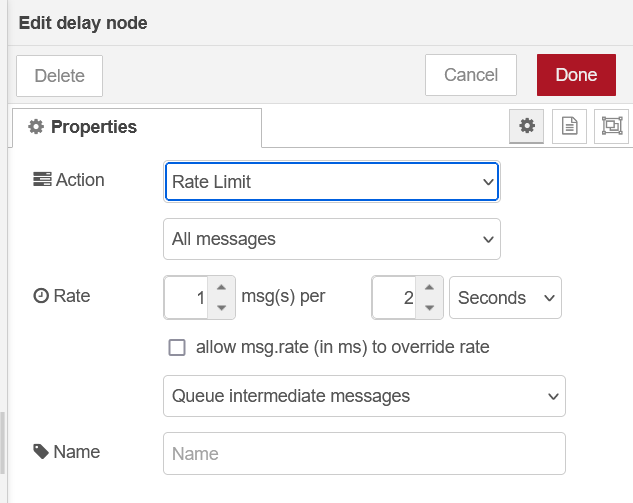
تنظیمات سوییچ بالایی



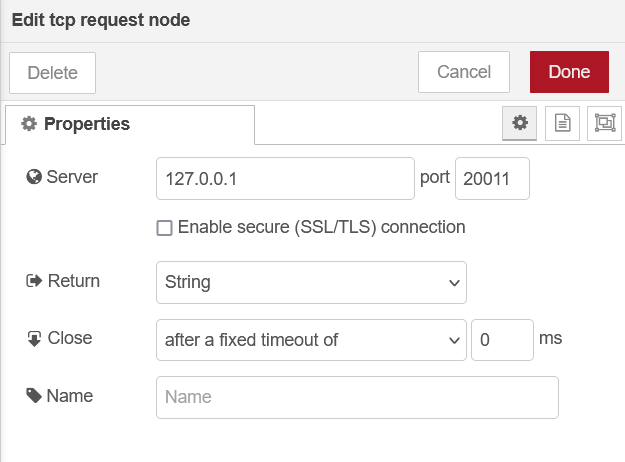
تنظیمات سوییچ پایینی



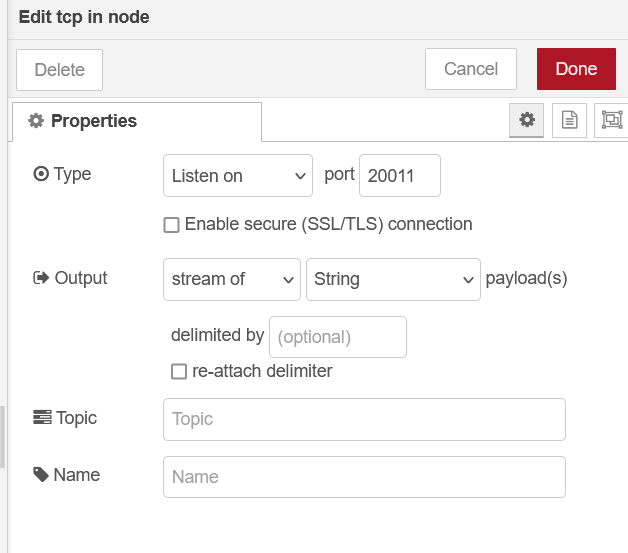
تنظیمات function



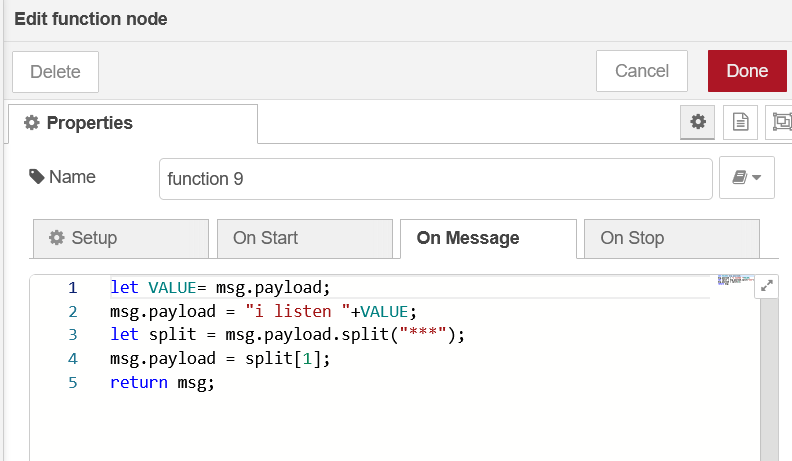
تنظیمات delay



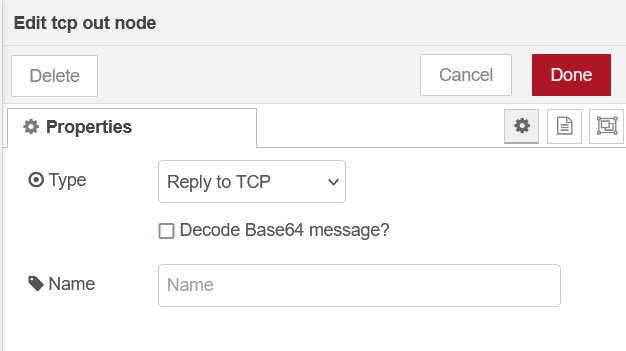
تنظیمات tcp request



تنظیمات tcp in



تنظیمات function



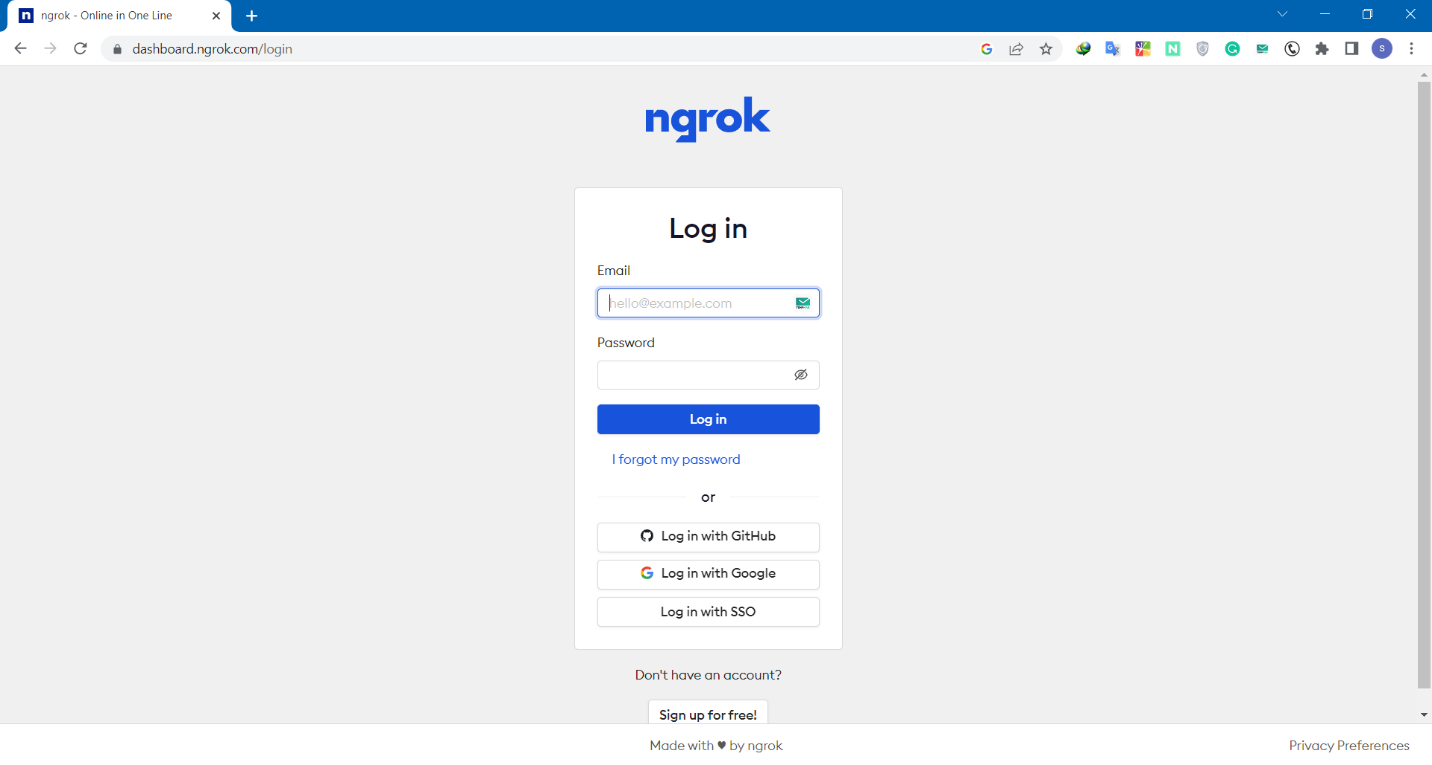
تنظیمات tcp out

نتیجه:

## Remote

برای اینکه node-red رو به صورت remote فعال کنیم از سرویس ngrok استفاده میکنیم که مراحل فعال سازیش به صورت زیر هست

ایتدا باید در سایت ngrok ، log in کنیم



میشه از اکانت github یا google برای log in شدن استفاده کرد

حالا فایل نصبی ngrok رو از سایتش دانلود میکنیم

A screenshot of a computer

Description automatically generated



بعد باید فایل نصبی رو به پوشه C/Window/System32 منتقل کنیم

A screenshot of a computer

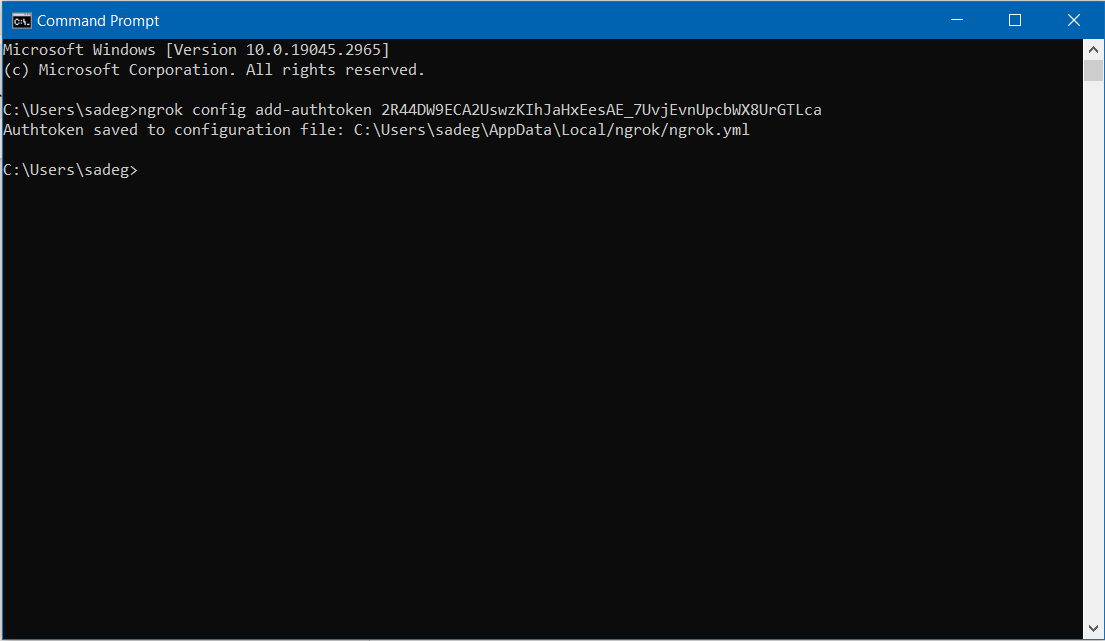
Description automatically generated

حالا باید با استفاده از key داده ،ngrok رو config کنیم

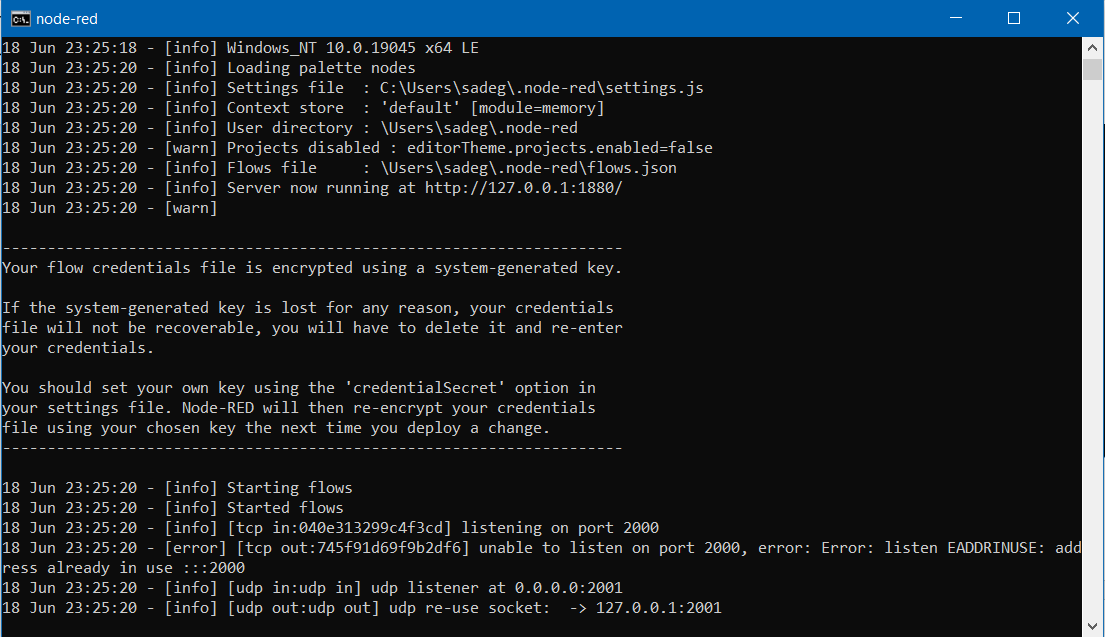
A screenshot of a computer

Description automatically generated

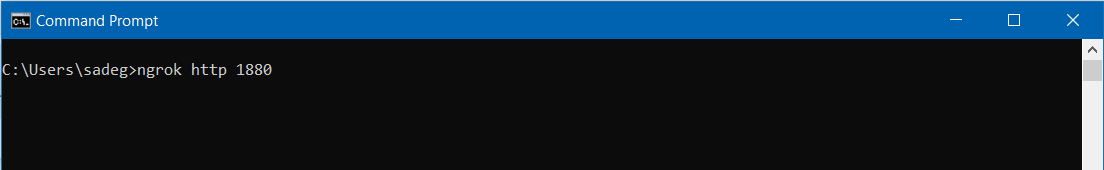
برای config کردن ngrok ، cmd در ویندوز یا Terminal در لینوکس باز میکنیم و کد مشخص شده رو paste میکنیم



حالا باید ابتدا node-red رو فعال کنیم



بعد با دستور ngrok http 1880 ، port forwarding انجام میدیم



نکته) 1880 پورت که node-red روش فعال هست

A picture containing text, screenshot, software, multimedia software

Description automatically generated

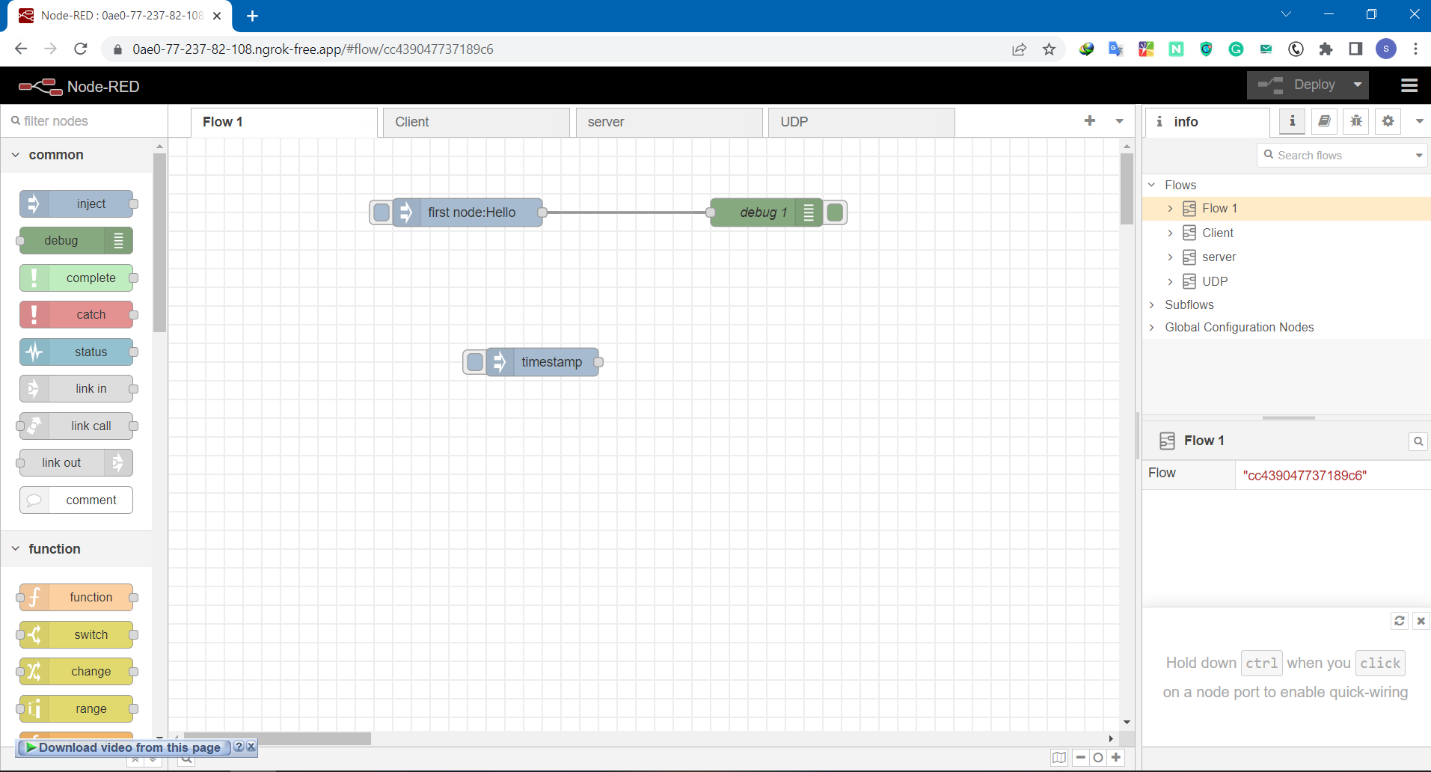


حالا به صورت remote با url مشخص شده میتونیم به node-red متصل شیم

A screenshot of a computer

Description automatically generated

این صفحه باز میشه و با زدن vist site میتونیم به صفحه node-red مورد نظر رو مشاهده بکنیم



پایان