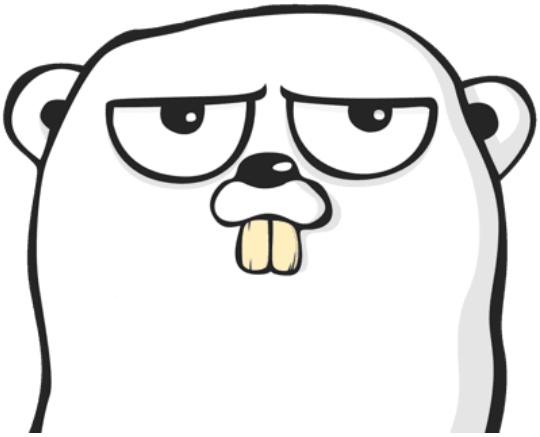


Sep 22, 2019 12 min read

My story of storing chat logs

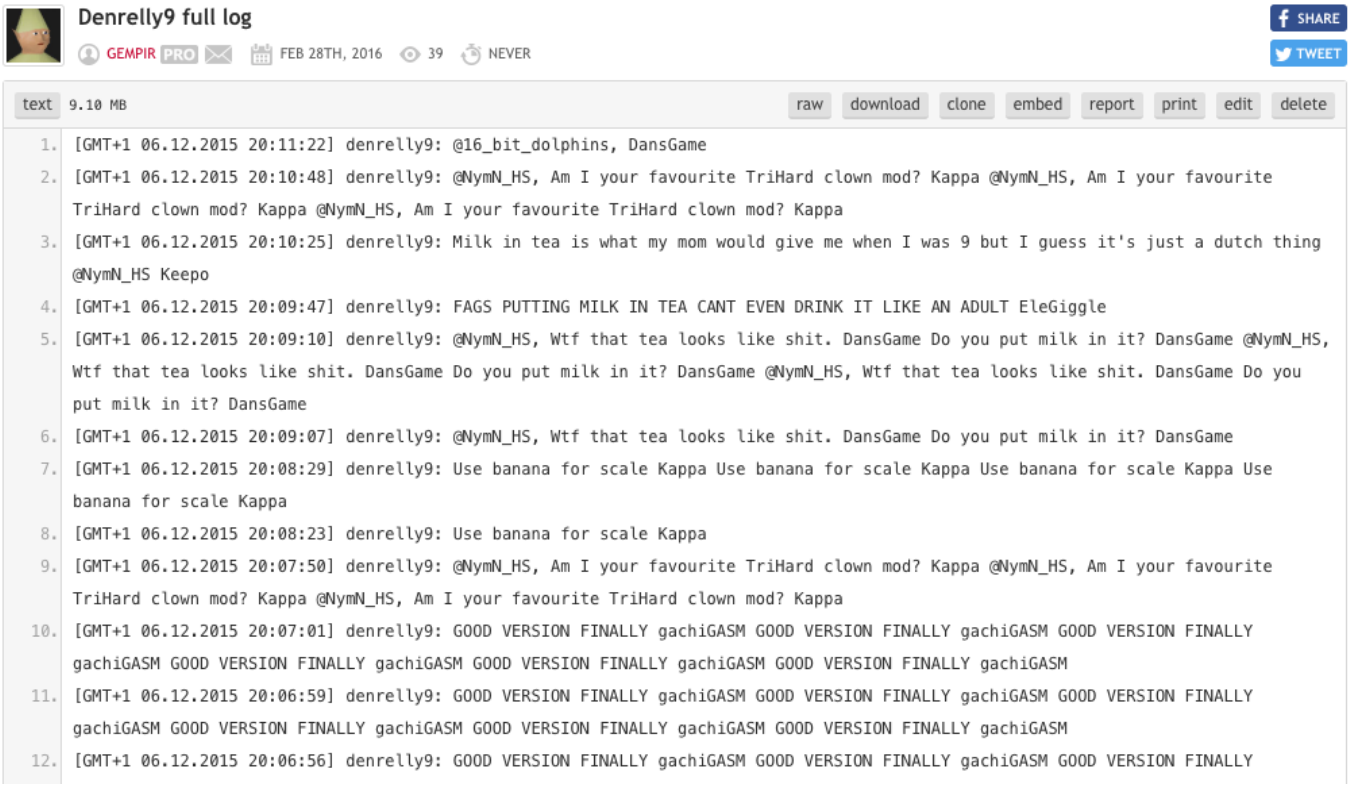
A few years ago I was constantly looking for fun side projects. I hadn't found that one project that could entertain me for long though. Until I started to play with IRC on twitch. IRC itself wasn't really that common anymore, but an up and coming streaming site named twitch.tv was using IRC to run the chat used by thousands of viewers. I started writing on this bot with simple features like the current time or repeating what I would say if I issue the command to do so. I quickly realized that this wasn't going anywhere since lots of big bots already supported most of the feature I offered on mine and a lot more including online dashboard etc.

What was missing though was a good chat log solution for smaller channels. Logs are so important for purposes of moderation or are just nice to go back to if you wanna go back to that link you posted a few days ago. Basically there were a very few chat log solutions back then. The only one I knew at the time was OverRustleLogs. In 2018 OverRustleLogs are still going strong and doing a great job, but the problem was they only logged big channels. For a streamers with just a few hundred viewers OverRustleLogs just wasn't available.

The first bot

So I thought maybe that's my chance, make my bot log the chat and impress the streamer, I was watching twitch.tv/nmplol, hes a pretty funny dude check him out if you like. I had never hosted for a real audience before besides some tiny personal portfolios. Twitch could expose my content to a lot of users very quickly, since when I link appears in chat many people might click it at once. Back then I thought hosting it myself would be annoying and too much data transferred per month. So I got the bright idea to use other sites for my logs. I would still store the whole logs locally but upload them on demand to pastebin, I even got a pro account for that purpose and also asked pastebin support for better ratelimits and they happily said yes since I wasn't doing anything bad.

If you care how this looked in terms of code here it is: https://github.com/gempir/gempbot/tree/365110d602a09d3e3cf92a42544870cfb6109a4a a lot has developed over time in that repository and you can tell by the commit count, I looked back a few years to find a point of time where I used the pastebin method, the rest of the code also does other random stuff chatbots usually do, but this story is not about that.



Please excuse the language in the log, twitch chat can be a crazy place. At the same time I started adding features to gempbot like saving all Oddshot links to another log. Oddshots were basically the first type of clips. Oddshots made the whole clipping meta even possible thanks to an easy to use browser extension, but this blog post isn't about clips.

The pastebin method was enough at the time because only a few people were requesting logs and I was only hosting it for 1 streamer basically.

Getting a server

So before I started hitting my extended ratelimits on pastebin I decided I needed my own server to host the logs from. I started out with a DigitalOcean droplet.

MEMORY	VCPUS	TRANSFER	SSD DISK	PRICE
1 GB	1 vCPU	1 TB	25 GB	$5/mo $0.007/hr

This setup ran my previous bot as a log collector and a newly developed application written in go to host the logs. gempAPI was one of my first projects in Go. If I recall correctly and it's where I started to enjoy Go more and more. I used redis to log the last message for every user that typed somewhere in some chat, which then gempAPI returned on demand. This setup ran pretty solid for a long time.

But I kept asking myself if I can't find a better solution. A real database was very appealing to me. It would allow me to query logs not just by username, I could do a lot more. Mysql was my first choice because it's what I use professionally and I used the most for private projects. I stopped using DigitalOcean somewhere in that time, because I realized that the price/performance ratio wasn't very good. Scaleway was a great alternative with a modern UI and relatively cheap.

```
func parseMessage(msg string) {
        if !strings.Contains(msg, ".tmi.twitch.tv PRIVMSG ") {
                return
        }

        fulluser := userrp.FindString(msg)
        userirc := strings.Split(fulluser, "!")
        username := userirc[0][1:len(userirc[0])]
        split2 := strings.Split(msg, ".tmi.twitch.tv PRIVMSG ")
        split3 := channelrp.FindString(split2[1])
        channel := split3[0 : len(split3)-2]
        split4 := strings.Split(split2[1], split3)
        message := split4[1]
        message = actionrp1.ReplaceAllLiteralString(message, "")
        message = actionrp2.ReplaceAllLiteralString(message, "")
        timestamp := time.Now().Format("2006-01-2 15:04:05")

        saveMessage(channel, username, message, timestamp)
}

func saveMessage(channel, username, message, timestamp string) {
        _, err := db.Exec("INSERT INTO gempLog (channel, username, message, timestamp) VALUES (?, ?, ?, ?)", channel, username, message, timestamp)
        checkErr(err)
}
```

This was my code back then to save a message. The parsing code was a big chaotic, but I think it's a lot better than importing an entire irc library that probably still won't support all the twitch irc features I needed. Mysql was cool at first. I could write queries that could filter everything I ever wanted. But already after a few days my messages table was getting big. I started to hit more than 1 million rows with just a few channels I was logging.

The Problem here is all this is running on a 500mb RAM Server which isn't even solely there for mysql, it runs the bot and API as well. Mysql can scale easily to billions of rows, but I was indexing basically 3 columns (channel, username and timestamp) which made RAM usage just go crazy, obviously.

I realized I can't afford a server big enough to handle that kind of scale. I thought of a lot of different variants I could use mysql, like creating a table for every channel or daily tables etc. I went through a lot of options but in the end I decided switching back to files was the better move.

Birth of go-twitch-irc

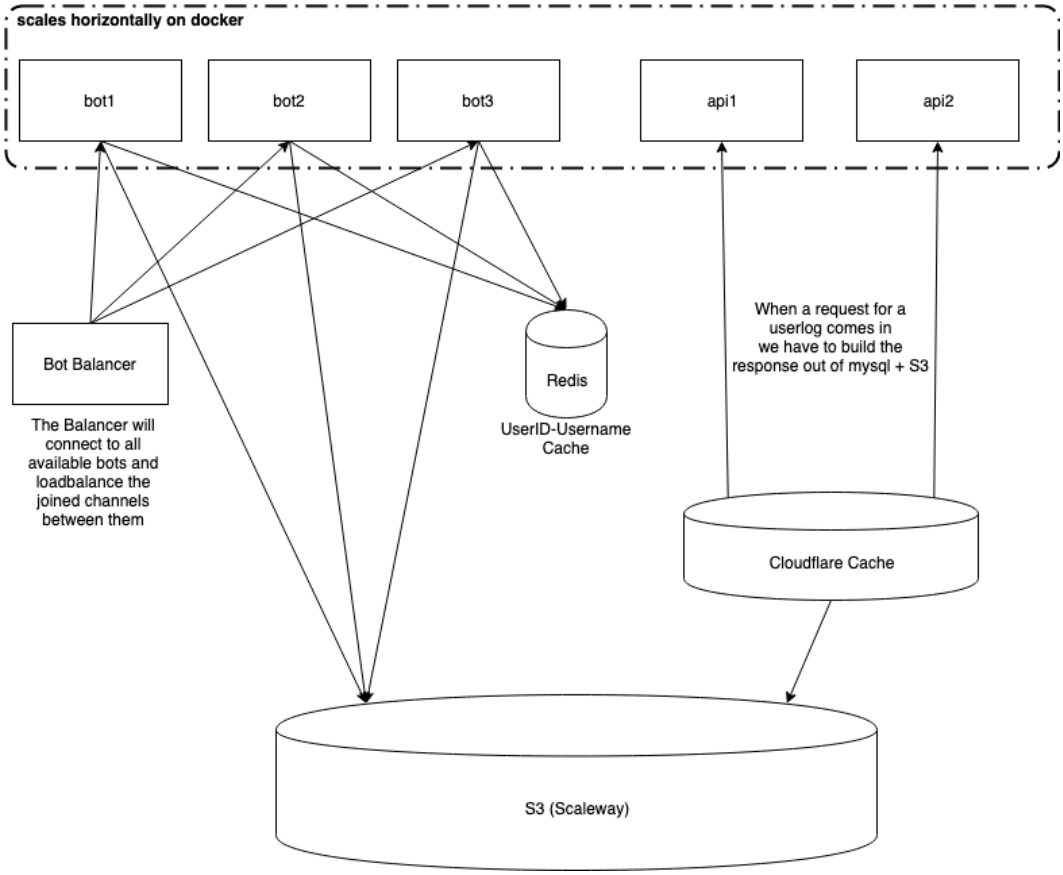
Another thing that bothered me again and again is parsing irc messages. I had many projects that needed to interact with twitch chat so I kept copying the same parse function again and again. I didn't find any good libraries in Go for parsing twitch irc messages. So over the past years and a lot of cool contributors the library go-twitch-irc has gotten a pretty nice feature set.

https://github.com/gempir/go-twitch-irc

One of my goals with this project was 100% test coverage, because I think in case of a parser this is really important and not an awfully hard goal to reach. Besides parsing this library also handle connections and reconnections etc. Thanks to it it's a lot less annoying to write a new project now.

Logs.tv

I always wanted to run a distributed system just to get experience how to run something like that. Specifically I wanted something that scaled horizontally meaning I could add instances as much as I want to improve performance. Since I scratched the Database for now I was looking for another alternative. For my distributed system I thought I needed a distributed filesystem of sorts. Amazon offers S3 which is basically just a storage service with a nice API. And at this time Scaleway was starting out with a beta of their own version of S3. Perfect timing for me. I dreamed big and bought a new domain for my ideas. With that logs.tv was born.



I sketched up a plan for my new system with horizontally scaled bot which write the logs to S3 and then API apps which abstract S3. S3 isn't ideal for frequent access so I still needed hot storage. Mysql was again my choice because now the storage would be limited to a month or so or however I want to store stuff in the hot storage before moving it to cold storage in S3. I couldn't find the original graphic which included mysql. Although I liked my plan, I quickly realized the system to archive mysql to S3 would be pretty complicated and I would always have weird "downtimes" when I was moving data between storages. If you care to look at the code of logstv it's open source like almost all my projects.

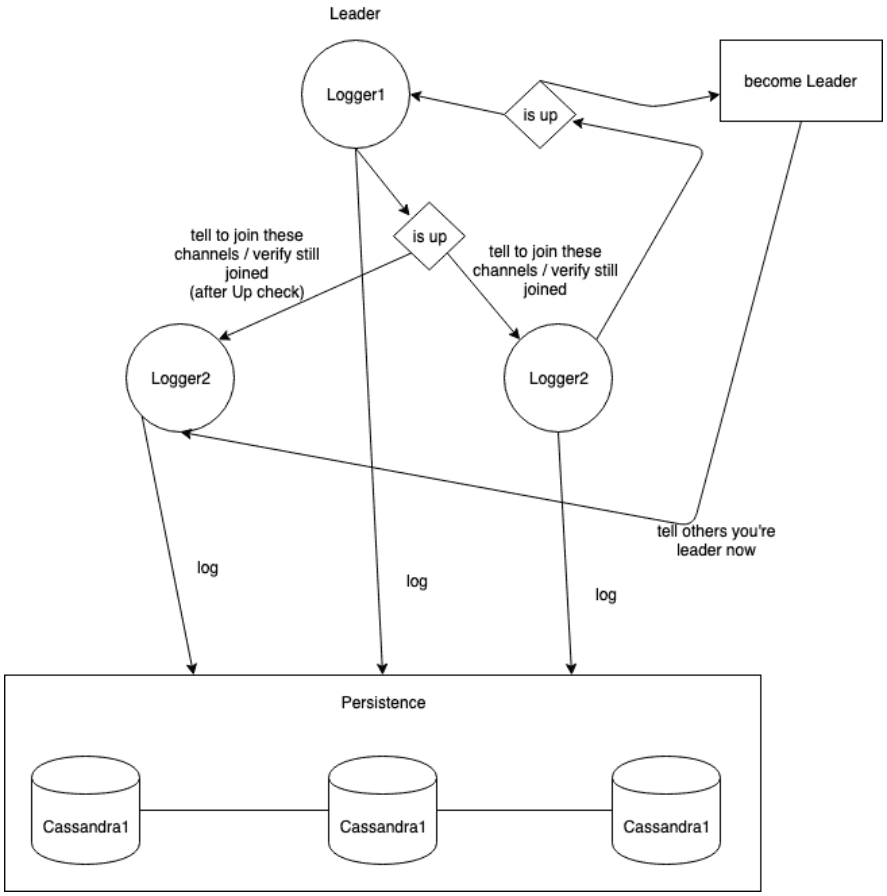
github.com/gempir/logstv

I gave up on S3 and mysql because I thought the log archiving would be too complicated and I also wouldn't have the time to develop all this. This is just a private side project. I still wanted to achieve my dream of a distributed system, but I needed a different storage system that wasn't as complicated as my hot/cold storage system. Mysql wasn't made for the amount of data in my mind and I thought I needed another type of database.

I read a very good article from discord at that time.

https://blog.discordapp.com/how-discord-stores-billions-of-messages-7fa6ec7ee4c7

Basically the TLDR for me was use Cassandra. Without any experience I installed cassandra and started hacking.



I also started thinking how my bots possibly could work in parallel. Now if you know anything about consensus algorithms you might have heard of raft. I was trying to build my own implementation of raft for my bots but I realized pretty quickly I was out of my depth. So I put this aside for now and switched my focus on the storage part of logstv.

Like I said I never touched Cassandra before and for a beginner Cassandra can look very similar to a normal relational database like mysql. The thing about Cassandra is you tailor your tables to the queries you will run on it. I wanted 3 things.

- Channel logs
- User per Channel logs
- User logs

Getting those would be incredible. But I couldn't really do all these things with just 1 table. The primary key is what is so important in Cassandra. It decides on which node data sits on Cassandra etc. So instead of storing logs in just 1 table I created 3 tables with the same data but differently stored.

```
var queries = []string{
        `CREATE  KEYSPACE IF NOT EXISTS logstv
        WITH REPLICATION = {
                'class' : 'SimpleStrategy',
                'replication_factor' : 1
        };`,
        `CREATE TABLE IF NOT EXISTS logstv.messages (
                timeuuid timeuuid,
                channelid bigint,
                userid bigint,
                message text,
                PRIMARY KEY ((channelid, userid), timeuuid)
        ) WITH CLUSTERING ORDER BY (timeuuid ASC);`,
        `CREATE TABLE IF NOT EXISTS logstv.channel_messages (
                timeuuid timeuuid,
                channelid bigint,
                message text,
                PRIMARY KEY ((channelid), timeuuid)
        ) WITH CLUSTERING ORDER BY (timeuuid ASC);`,
        `CREATE TABLE IF NOT EXISTS logstv.user_messages (
                timeuuid timeuuid,
                userid bigint,
                message text,
                PRIMARY KEY ((userid), timeuuid)
        ) WITH CLUSTERING ORDER BY (timeuuid ASC);`,
        `CREATE TABLE IF NOT EXISTS logstv.channels (
                userid bigint,
                username text,
                PRIMARY KEY (userid, username)
        );`,
        `CREATE INDEX IF NOT EXISTS channels_username_index ON logstv.channels (username)`,
        `CREATE TABLE IF NOT EXISTS logstv.users (
                userid bigint,
                username text,
                PRIMARY KEY (userid, username)
        );`,
        `CREATE INDEX IF NOT EXISTS users_username_index ON logstv.users (username)`,
}
```

This setup was running actually in production for a few weeks. It felt amazing. Because it was fast and data access was so flexible. If you wanna look at more code here is the project.

https://github.com/gempir/logstv-cassandra

I wanted this to work, but after like 1 week of logging I was already up to 1 GB of data. I already ran compression on my tables but it wasn't good enough. I couldn't handle that kind of data. I was logging the past 3 years with less than 10 GB with my text file solution. So again, I fell back to text files.

Justlog

My current solution for production was a very basic version of gempbot with a pretty weak API all written in Go. I decided to rewrite gempbot and make it a lot more user friendly so others could use the new gempbot as well if they wanted to log something. A new name felt appropriate and so justlog was born.

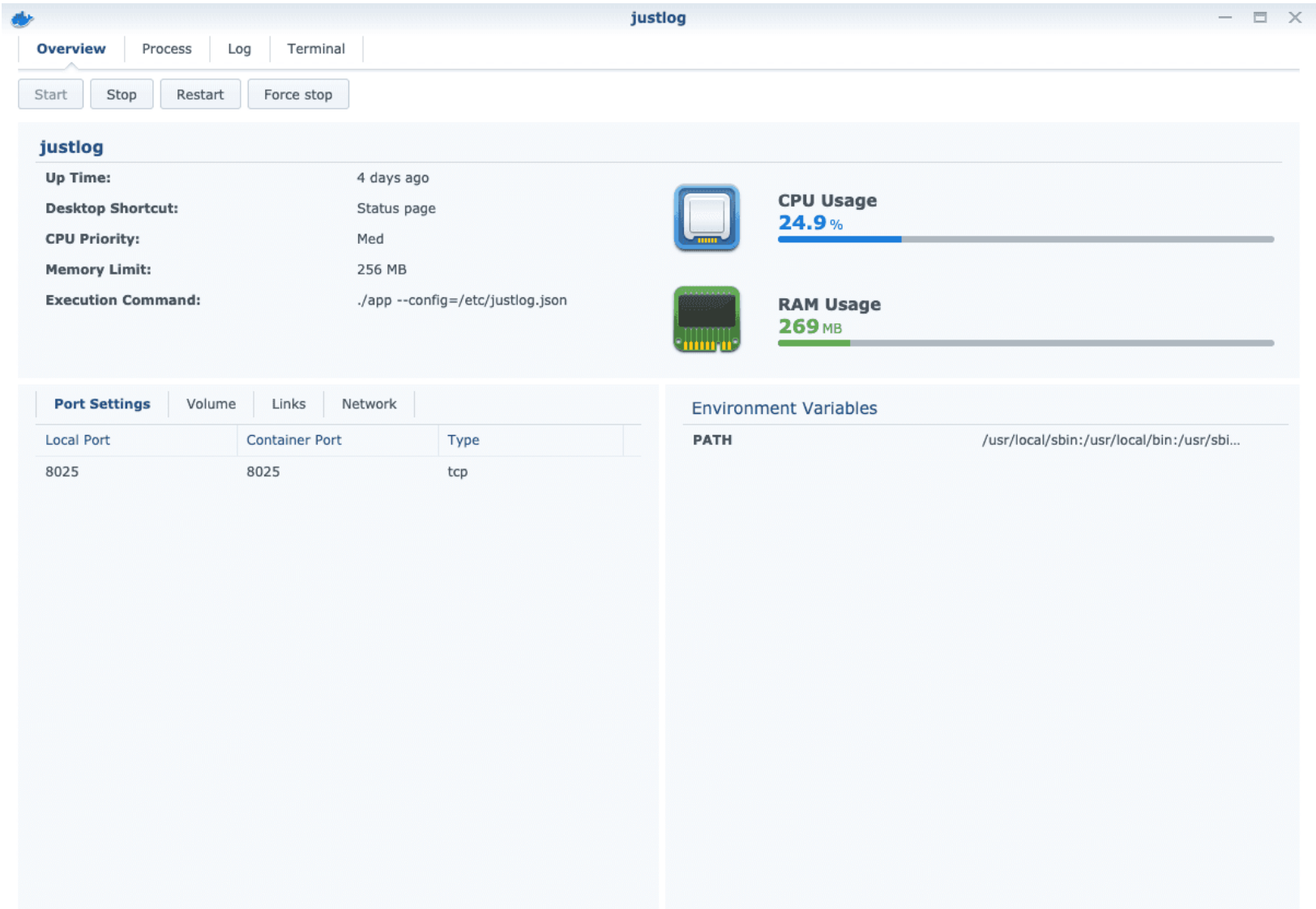
https://github.com/gempir/justlog

Justlog is written in Go, uses go-twitch-irc for the connection and parsing of twitch chat, it uses my favorite http framework in go, echo, and thankfully it has a real API documentation with the help of swaggo. Checkout the documentation here: api.gempir.com.

Justlog is configured with a simple json file and uses a userid based system to log to disk to avoid any problems with renames. I stopped logging messages as simple messages and instead now log the raw IRC message. This gives the advantage of having all the data like emotes, timestamps or other tags from every message even old ones, so when we add new features to the API it can also offer those features for old messages.

At first I ran justlog on a scaleway server with around 50 GB disk space, which was more than enough, because thanks to the power of gzip I was barely going over 13 GB of logs eventhough I was storing 3 years worth of logs from nymn's and forsen's channel.

I wanted to save myself some money on the 5.99 € per month I was paying for the server and I just bought a new NAS for my home. The (Synology DiskStation DS218+) (https://www.synology.com/en-global/products/DS218+) is what justlog currently runs on. It has support for docker which made it really attractive to me. I can monitor justlog from a nice webUI and the logs are saved on a NAS built for storage.



Lessons Learned

I learned a few lessons through all those past years.

- Go is fun, I wanna keep using it
- Textfiles + Gzip is unbeatable in terms of storage efficiency
- Distributed systems are fun but hard to maintain and a lot of work
- Docker can also be useful for small private projects
- Open Source is awesome

I have a big habit to not to stick to whatever I am currently obsessed with. I can switch pretty quickly, but I think that's fun and makes me learn a lot. For now I'm pretty happy with the current solution and will keep improving justlog. I have friends who already host their own justlog to log other channels, which I think is pretty good that they think my software is good enough. I will say though Cassandra was the most fun working with and I hope I can one day return to Cassandra maybe not for logging but for other purposes maybe even professionally.

3 Comments - *powered by utteranc.es*

GeeksGrimoire commented on Jan 18, 2022

I have some questions on this tool. Not sure if you run the URL https://vtlogs.moe/ that utilizes your tool described here or not, but I am a twitch mod & Discord Admin for TechyCutie and we've found this tool very helpful for cross-referencing people in chat but would love if we could have TechyCutie added to this for chat logging and if you're not responsible for this URL, how can I use this tool and set it up for Logging for TechyCutie?

Thank you,
GeeksGrimoire | JC-Dragon#0001

gempir commented on Jan 18, 2022 Owner

@GeeksGrimoire I don't have anything to do with that instance. You'll have to contact the owner if you want your channel to be added.

Otherwise

I recommend looking at the readme and running justlog in docker. https://github.com/gempir/justlog

GeeksGrimoire commented on Jan 18, 2022

@gempir Thank you for your quick response. I will see if I can track down the person who is the owner of that instance.

Write Preview

Sign in to comment

Styling with Markdown is supported

Sign in with GitHub