

computers are bad

You are receiving this facsimile because you signed up for fax delivery of this newsletter. To stop delivery, contact Computers Are Bad by email or fax.

<https://computer.rip> - me@computer.rip - fax: +1 (505) 926-5492

2023-04-10 solving problems with chatgpt

One of the foundational goals of computer technology, at least as understood by popular culture, is to automate away our jobs. When your job is (put simply) to get computers to work correctly, there's sort of an irony to this view of things. Still, just within the span of my career there have been technological advancements that have both automated away parts of my job (Puppet might be an example here, the prior art of CFEngine notwithstanding) and given me plenty more to bill time for (Kubernetes).

With the rise of large language models (LLMs), there is a large camp of people who think these ML applications are going to automate away larger portions of more jobs. I find myself curious: should I fear for my employment?

The thing is, a large portion of my profession could be accurately described as "Googling it," although lately I'm racking up a Kagi bill instead. Obviously this is an important strategy for things like error messages where it's often faster to see if someone has solved the same problem before than to figure it out from first principles. But it's sort of true for routine tasks as well... maybe my memory is just unusually poor, but there's a huge set of tasks where I will repeatedly look up the same Digital Ocean guide because I just want to copy-paste their config file snippets instead of having to look in docs to remember the names of the right directives.

Of course, what makes me a "professional" is that I have opinions about the correct ways certain things should be done, so I often ignore parts of these guides or make changes to suit my preferences on important issues like Unix domain sockets or localhost network sockets for communication with application servers. In a way it's sort of silly to follow some guide on best-linux-tutorials.biz and then just ignore the parts I disagree with but, well, that's what I tend to do. So I wonder: would a better approach be an LLM?

In today's episode, I'm going to ask ChatGPT for guides for some increasingly complex Linux sysadmin and DevOps tasks and then see whether or not I agree with its output. I don't really intend this as that serious of a critique of the use of ML in operations but hey, maybe we'll learn something useful!

The LEMP Stack

Back when I was a child first learning to hate software, the dominant stack for websites was "LAMP." This stood for Linux, Apache, MySQL, PHP. Nowadays it seems that at least a few people are saying "LEMP" for Linux, eNginx, Mymariadb, PHP? I don't like this, aesthetically. In any case, there's a number of things in PHP that I tend to deploy a

lot, Dokuwiki being a prime example. This means that I am repeatedly wanting to set up Nginx to proxy requests to PHP-FPM, which is not difficult but does involve remembering a couple of config directives and where php-fpm puts its socket by default. Most of the time I don't actually want an RDBMS (personally I usually just use sqlite for everything) so for a long time I've googled for some guide and copied their snippets while ignoring the parts about MySQL/MariaDB. Let's see if ChatGPT has something more to the point for this very common and simple task.

Prompt: "how do I set up nginx and php?"

To set up Nginx and PHP on your server, follow these steps:

1. Install Nginx on your server:
For Ubuntu/Debian:
`sudo apt update sudo apt install nginx`
For CentOS/RHEL:
`sudo yum update sudo yum install nginx`

The output starts off strong by offering snippets for both "Ubuntu/Debian" and "CentOS/RHEL." These two cover the good majority of the Linux server landscape, and while I could quibble with the label "CentOS/RHEL" rather than something that doesn't invoke the mostly-dead CentOS project like "RHEL/Fedora," ChatGPT is following the same convention most people do.

The instructions it provides are pretty reasonable, but there's something interesting here that those of you familiar with the two dominant Linux package manager families might detect. For Debian, it tells us to 'update' and then 'install.' for RHEL, it tells us to 'update' and then 'install.' These are neatly parallel except that the 'update' subcommand of apt and yum do fairly different things! On Debian it will retrieve the repo metadata and then install nginx (upgrading software is a different subcommand), on RHEL it will upgrade all of the managed software on the system and then install nginx (repo metadata is retrieved automatically by yum when it exceeds a configured 'stale' timeout).

I'm not sure how much we can make of this. It seems likely to me that we are seeing ChatGPT's lack of understanding of the underlying material: it is extremely common for people to 'update' and then 'install' on both platforms, so each in isolation is pretty reasonable, but it's odd for it to put them in parallel without noting that they will do different things. If the 'yum update' were an intentional choice we should see 'apt upgrade,' but we don't.

2. Install PHP and the required PHP extensions:
For Ubuntu/Debian:
`sudo apt install php-fpm php-mysql php-mbstring`
For CentOS/RHEL:
`sudo yum install php-fpm php-mysql php-mbstring`

This is a little odder. It says "the required PHP extensions" which are apparently mysql and mbstring. It's not clear where or why these are required, which makes me think ChatGPT got this command from some guide on deploying specific software. WordPress might be to blame here as it requires mbstring, and the sheer number of LAMP/LEMP guides for the assumption that we will need the mysql extension.

3. Configure Nginx to use PHP:

Create a new server block configuration file in /etc/nginx/sites-available directory with the following content:

```
server { listen 80; server_name example.com; root /var/www/html; index
index.php index.html index.htm;
```

```
location / {
    try_files $uri $uri/ /index.php?$query_string;
}
```

```
location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
}
```

```
}
```

Replace example.com with your domain name and php7.4 with your installed PHP version.

This is pretty much what you would find in any guide. One interesting thing is the 'try_files' directive. There are about a million ways to set this up in nginx and some people prefer different approaches, but what we see here is pretty typical of php setup guides and should work fine. The catch is explicitly passing the \$query_string which sort of gets eaten otherwise when nginx internally redirects to use index.php.

4. Test your Nginx configuration:

```
sudo nginx -t
```

5. Reload Nginx to apply the new configuration:

```
sudo systemctl reload nginx
```

Hey, nice work! A surprising number of people don't know about '-t' to validate config files.

6. Create a PHP file to test if everything is working:

```
sudo nano /var/www/html/info.php
```

Add the following content:

7. Open your web browser and navigate to http://your-server-ip/info.php. You should see the PHP information page.

That's it! You now have Nginx and PHP installed and configured on your server.

So yeah, this isn't exactly part of the install process, but still very normal for guides.

Overall I'd say this is about an 8/10. The only things I disliked were:

1. Telling the user to upgrade packages on RHEL... not an unusual thing in guides to update first "for good measure" but I personally feel that it's not a great practice to tell the user to 'yum update' without explaining that will update anything.
2. Telling the user to install mysql and mbstring, which aren't really going to do any harm, but if you're going to include a "common" set of PHP extensions in the instructions it should be bigger than just these two. It's just odd.

'du' and 'df' disagree

This is actually an interview question I like to use for Linux-intensive positions. It might seem a little in the weeds but to be honest I think I run into this situation in real life probably at least once a month. The problem looks like this: tools like 'df' tell you that a partition is 99% or 100% full, and you might be having problems because of the full partition. But when you use 'du' to investigate, it shows a total usage that is appreciably less than 'df' does. Why do these two-letter commands not get along? is dd somehow the answer?

Prompt: On Linux, a partition is full but du doesn't show as much data as df does.

If you are seeing a discrepancy between the output of du and df on a Linux system, where df reports that a partition is full but du does not show as much data, it's possible that there are files that are being held open by processes and therefore are not being deleted even though they have been unlinked (deleted).

This actually isn't the answer that I expect but it's a reasonable answer. This could very well be the cause, I just find that this situation is less common than having a mount that "shadows" data on the volume at the next higher mount point. For example, say you have a volume mounted at / and a volume mounted at /home. If you have some kind of problem that results in /home failing to mount, or changed the filesystem setup at some point, or even have weird boot-order problems, you can sometimes end up with data in /home/ on the volume mounted at /. This data counts against the usage of the volume at / but won't show up in tools like 'du' since it's "shadowed" by /home/ now being a mountpoint to another volume. Easy problem to have if you are rearranging the volumes/mounts on a system that's already been in use.

I'll omit the rest of the output, but it makes sense. It tells us to use 'lsof' to figure out what process is keeping the files open and then kill that process. The guide does include the requisite paragraph that you need to investigate to determine how this happened in the first place.

Overall I'm giving this a 5/10, because it gave a reasonable answer but omitted the possibility of mount shadowing. Which of these two causes is more likely depends on the system and workload, but I think a thorough answer should include both possibilities.

Per-process network policy

One sort of common "advanced" Linux networking scenario is when you are using a full-tunnel VPN and want to route all traffic through it, but you have to get the VPN itself to connect to its endpoint without trying to go through itself. There are a few ways to solve this problem, but one of the less common and (in my opinion) more elegant approaches is to get the VPN service to use its own special routing table. This same concept also makes it easy to set up firewall policy around the VPN to prevent leakage. Now there are a lot of caveats to this and I'm really just talking about userspace VPNs here, but that probably makes it a good challenge for ChatGPT.

prompt: How do I configure per-process route tables on Linux

To configure per-process route tables on Linux, you can use the ip command and

the `setsockopt()` system call.

Here are the steps to follow:

1. Create a new routing table:

```
sudo echo "200 mytable" >> /etc/iproute2/rt_tables
```

This will create a new routing table with the name `mytable` and the ID 200.

It maybe seems a little weird that this is the way to create a routing table but it's fine. The `iproute2` stuff that facilitates multiple routing tables without multiple headaches is pretty new and there isn't a lot of porcelain around it... and I'm not sure there ever will be, this isn't a very common task and while editing the file seems a little old-school compared to most of the contemporary network tooling it works just fine. There's very little need to expose this over a bus or anything.

1. Configure the routes for the new table:

```
sudo ip route add default via dev table mytable
```

This command will add a default route to the new table with the specified gateway and interface.

I will take this moment to make a couple of funny observations about the mechanics of ChatGPT's output. First, every step in this guide is numbered 1. Some things here are probably copy-paste errors on my part (I'm reformatting the output to look better in plaintext), but that isn't, this output has four step ones. Another is the code blocks. I'm just using plaintext but ChatGPT uses nice embedded code blocks with syntax highlighting. Syntax highlighting means that it needs to know (or guess) the language it's highlighting and the codeblocks show what language they're set to. It is... seemingly random? The above shell command is `'csharp'`, but `'sql'` seems to be a more common guess for shell commands than even `'bash'`. This just means that most of the time the syntax highlighting doesn't really make sense.

I'm not sure where either of these are going wrong, but I get the feeling (without doing any actual research) that ChatGPT is outputting markdown. Markdown numbered lists can be fiddly and there are a few ways to end up with all step 1 like this. When it comes to the code blocks, I'm not sure if chatGPT is specifying the language or if the code block service is guessing it based on contents. I tend to guess the former because otherwise the guesses are very weird. I can't see any reason it would think one of these commands is `csharp` and the other is `sql`, they both have the same set of characters in them.

And yeah, this instruction makes perfect sense. You need to add some routes to the new route table.

1. Configure the process to use the new table:

```
sudo ip rule add from table mytable
```

This command will add a new rule to use the `mytable` routing table for traffic originating from the specified IP address.

:thinking_emoji:

There's some clear semantic confusion here. I think ChatGPT is just going too readily to the many guides it's read that create a route table and then apply that route table based

on ip rules. This is a very common thing to do, I would wager the most common way to use custom route tables, but it doesn't help in our case. The intro of this step is also just wrong (this has nothing to do with the process), but the sentence after correctly explains what it does.

1. Finally, start the process with the `setsockopt()` system call to bind it to the new table:

```
int table_id = 200; setsockopt(socket_fd, SOL_SOCKET, SO_BINDTODEVICE,
"eth0", strlen("eth0")+1); setsockopt(socket_fd, SOL_SOCKET, SO_MARK,
&table_id, sizeof(table_id));
```

This code will bind the process to the eth0 interface and mark its packets with the mytable routing table ID.

This is, well, it's not exactly wrong but it's also not right.

If I fiddled with the prompt I'm sure I could get it to output something that makes more sense, but I also feel like this is an unreasonable way to interpret the prompt I did provide. To be clear, I expected that it would use `cgroups` and `iptables` to apply the new custom route table to some existing application. This seems like a way more common situation than to... write or modify an existing application in C. On the upside, the code block correctly has the language as 'c'!

All that said, if we start from the assumption that we are writing a new application this still doesn't work exactly. The option we see here, `SO_MARK`, applies a 'mark' in the sense of netfilter to traffic on the socket. Marks are a super useful feature that allows us to set an arbitrary tag on each packet that we can then refer to in our other network policy. A very common use for marks is to match the traffic up with an `iptables` rule that sets the routing table for that traffic... but ChatGPT didn't tell us to set that rule, it had us set the rule based on source port instead. I think maybe ChatGPT was *trying* to cover this in step 1 number 3 but it didn't quite choose the right rule.

Repeat steps 3 and 4 for each process that needs to use the new routing table.

There are no steps 3 and 4! There is only step 1!

Overall I'd give this a 3/10. Honestly I think ChatGPT did better than I expected when considering that this is not a very common use-case, but ChatGPT's output is worse than what you get if you google the same question.

Takeaways

This isn't intended to be a super carefully thought out research piece, but like others I've seen reporting that suggests that ChatGPT is pretty good at generating source code but has a tendency to leave subtle errors. I wondered if the same thing might be true in the old-school system administration space, but the feeling I walked away with is a little different: ChatGPT does make a lot of errors, but they aren't particularly subtle. I suspect this has to do with the different nature of the training material for these kind of questions, which is more in the form of narrative guides and documentation that keep steps simple but leave a lot of assumptions.

I pretty much covered every question I thought to ask in a short time spent playing around. There was one question that I was just unable to get ChatGPT to generate a good

answer for... an AWS network architecture question involving load balancing for both HTTP(S) and arbitrary TCP services on the same elastic IP when you want layer 7 behavior. I could mark this as a 0/10 for ChatGPT but it might be a case of more "prompt engineering" being required... it kept producing output that was reasonable but failed to address one or more of my requirements, so I kept making the requirements more explicit, and then it would just miss a different set of requirements. It may very well be possible to get ChatGPT to produce a correct solution but it was definitely getting to be less useful than a search engine, even with how difficult the AWS documentation can be to use for architecture questions.

On the whole, I felt that ChatGPT was performing more poorly than Google for similar queries. It's possible to get ChatGPT to refine its output by adding more detail (at least if you don't get too deep into AWS networking capabilities), which is a big plus over a conventional search engine, but honestly it still didn't feel to me like this was a savings of effort over reading a few different articles and synthesizing.

One of the reasons this was on my mind is because I'm working with a client right now who has an interesting habit of copying and pasting all the error messages they get into ChatGPT, while still screen sharing. From this sort of eavesdropping on AI I have not been very impressed with its output, which has often been high-level to the point of uselessness. It sort of has the feel of AWS documentation, actually... he would put a very specific error from, let's say, Terraform into ChatGPT and it would answer with a few paragraphs about how Terraform works at a high level. I'm sure this can be improved with a more specific training corpus but I'm not sure what that corpus would be, exactly, which continues to stymie my plans to just forward all the emails I get from clients to ChatGPT and give them the response.

I mean, I think it would keep them going back and forth for a while, but they might feel like it's a good value for money.