How To: Enable x11 Forwarding with SSH

In the last article, I explained how to enable SSH. In today’s article, we’re going to learn how to forward GUI application windows with SSH. x11 forwarding is easy and beneficial.

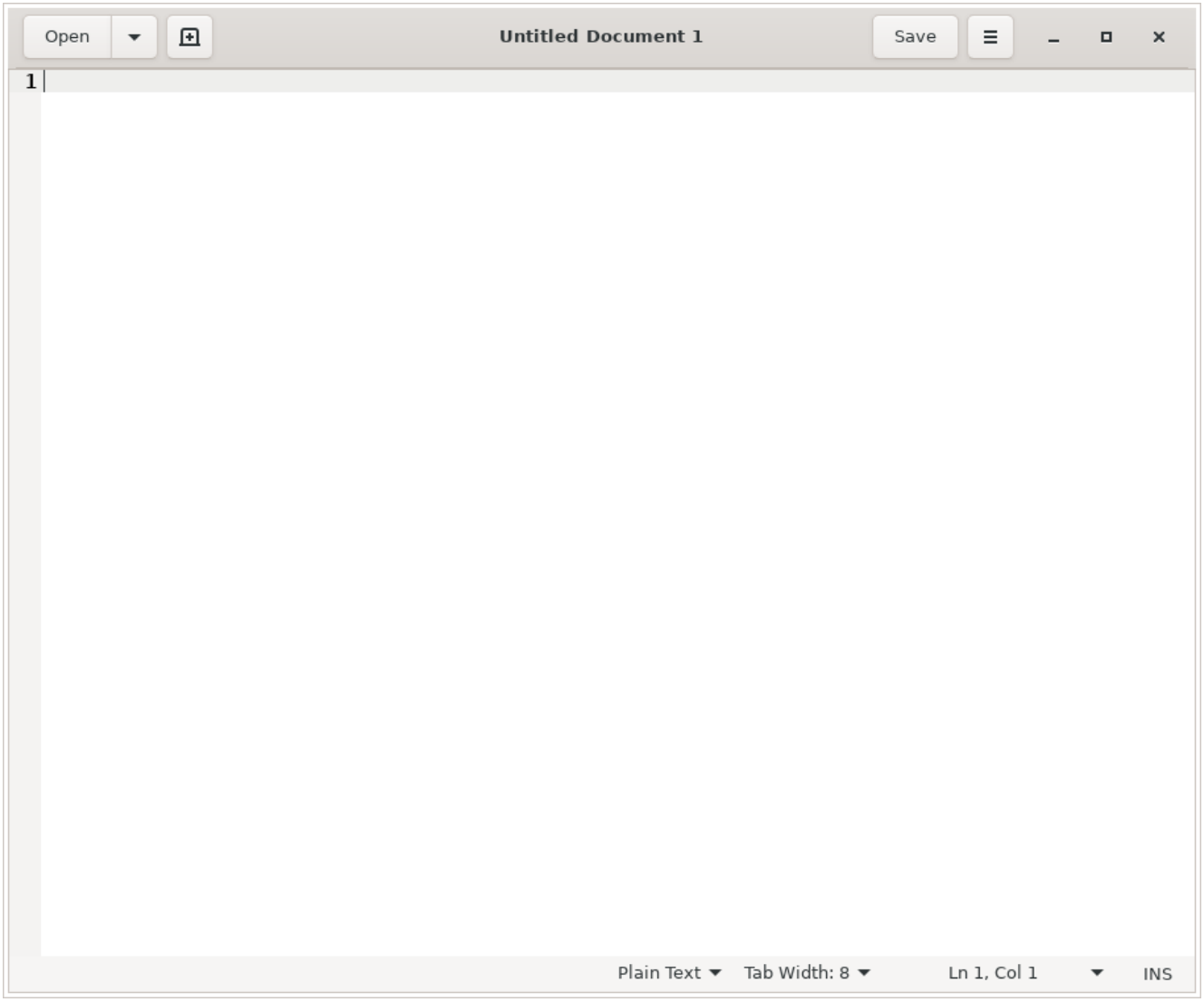
Just to quickly clear up a misconception, x11 forwarding works just fine with Wayland. Way back in the earliest days, it was agreed that it should retain backwards compatibility with x11 forwarding.

Enable x11 Forwarding With SSH

What is this strange thing, this x11 forwarding?

Well, when you’re connected to another computer via SSH you can use the terminal to control the computer. That’s great, but what if you want to use a GUI application? Sure, you could set up some sort of remote desktop application, such as VNC. Or, alternatively, you can just forward graphic applications over SSH. It’s remarkably easy!

Perhaps a picture is in order. Check this:



This GEdit is actually running on my laptop, forwarded to this desktop.

That’s right. That’s running on my laptop. I’ve just forwarded the GUI application to this computer. If I write something and save it, it’d be saved on the computer that I’m connected to and not the computer that I’m using.

Amusingly, I used this just earlier today. I had to move a complex password to my laptop and I was being lazy. See? It comes in handier than you might think. Okay, I could have easily used nano, but I wanted to make sure that I’d configured x11 forwarding properly and get a screenshot.

So, how do you do it?

Well, first you need to crack open your terminal. To do that,

you just press CTRL + ALT + T on your keyboard and your default terminal emulator will open.

Now, in said terminal, I want you to run the following command:

[code]sudo nano /etc/ssh/sshd_config[/code]

Once you have that open, you just need to remove the appropriate asterisk (uncomment it out) for the right line. Look for the line that says:

[code]#X11Forwarding yes[/code]

And change it to:

[code]X11Forwarding yes[/code]

Then save the file by pressing CTRL + X, then Y, and then ENTER.

Next up, you'll need to restart the SSH service and that's done with:

[code]systemctl restart sshd[/code]

And that's it. You can now use x11 forwarding over SSH. To do so, you just need to add the -X switch.

[code]ssh user@host_name.local -X[/code]

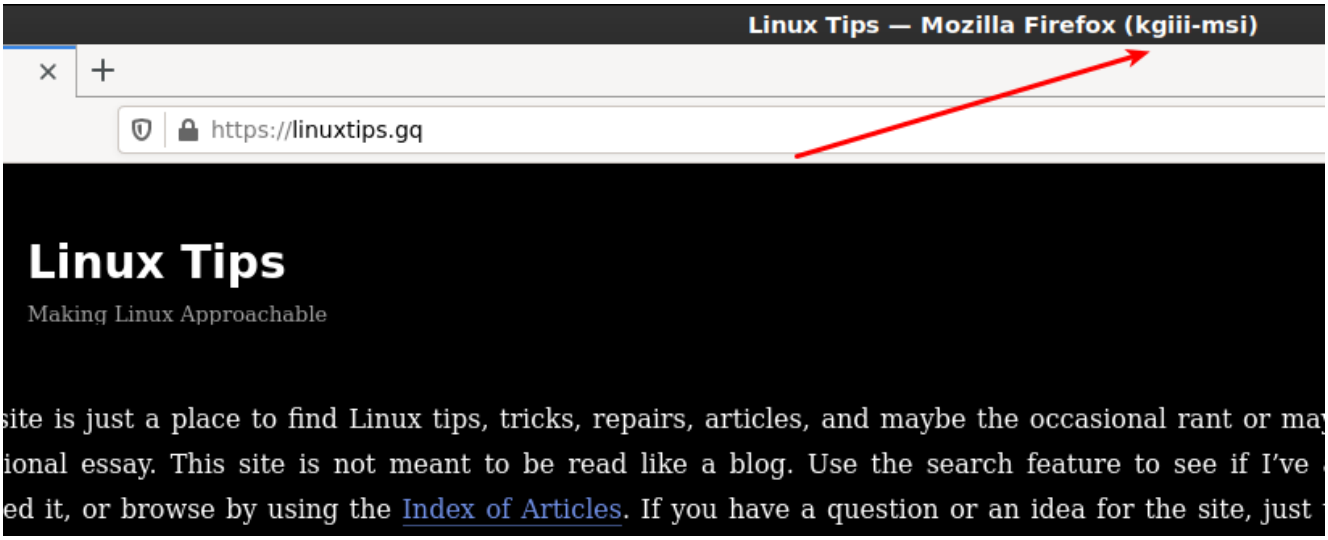
To try to make sense of that, if I were to do this connecting to the new MSI laptop, then my command would look just like:

[code]ssh kgiii@kgiii-msi.local -X[/code]

You can also use the IP address, instead of the hostname, just like we discussed in the previous article about SSH. To do that, it looks like this:

[code]ssh user@ip_address -X[/code]

Once you're there, just go ahead and start an application. For example, open gedit by typing just that. You may find some applications won't work, often due to ownership and permissions, but you'll find many that do. If you find one that doesn't work, you can always check any errors thrown and work to resolve the issue.



See? Note the carefully drawn arrow that shows where it was forwarded from. Tada!

That's an example of Firefox forwarded over SSH using x11 forwarding and you may notice the washed out look. I haven't really dug into it, but I am reasonably confident that it's because of compression. I've never needed to dig into that and, amazingly enough, I don't know everything and don't see any reason to invest time learning about it any further. You get what I know, not what I am able to learn! (You're welcome!)

Anyhow, there you have it. One more article in the books and one more bit of knowledge plastered across the internet. If you found the article useful, you could use that rating button. I kinda use those ratings to decide what to write. You can also sign up for the newsletter. I had to remove some @gmx.com email address because they simply don't let my emails through. (I've never sent a single unsolicited message, it's just a horrible ccTLD and it gets filtered often.) Sign up

again with a different email address. Thanks for reading!