How To Install and Configure VNC Remote Access for the GNOME Desktop on CentOS 7

PostedNovember 25, 2014 243.3kviews [APPLICATIONS](https://www.digitalocean.com/community/tags/applications?type=tutorials) [CENTOS](https://www.digitalocean.com/community/tags/centos?type=tutorials)

Introduction

**VNC** or Virtual Network Computing is a platform-independent protocol that enables users to connect to a remote computer system and use its resources from a Graphical User Interface (GUI).

It's like remote controlling an application: the client computer's keystrokes or mouse clicks are transmitted over the network to the remote computer. VNC also allows clipboard sharing between both computers. If you come from a Microsoft Windows server background, VNC is much like the Remote Desktop Service, except it's also available for OS X, Linux, and other operating systems.

Like everything else in the networking world, VNC is based on the client server model: VNC server runs on a remote computer — your Droplet — which serves incoming client requests.

**Goals**

In this tutorial we will learn how to install and configure a VNC server on CentOS 7. We will install the TigerVNC server which is freely available from the [TigerVNC GitHub repository](https://github.com/TigerVNC/tigervnc/releases).

To demonstrate how VNC works, we will also install the GNOME desktop on your CentOS server. We will create two user accounts and configure VNC access for them. We will then test their connectivity to the remote desktop, and finally, learn how to secure the remote connection through an SSH tunnel.

**Prerequisites**

The commands, packages, and files shown in this tutorial were tested on a minimal installation of CentOS 7. We would recommend the following:

* **Distro**: CentOS 7, 64-bit
* **Resource Requirements**: A Droplet with 2 GB RAM
* To follow this tutorial, you should use a sudo user. To understand how sudo privileges work, you can refer to [this DigitalOcean tutorial](https://www.digitalocean.com/community/tutorials/how-to-edit-the-sudoers-file-on-ubuntu-and-centos)

**Warning:** You should not run any commands, queries, or configurations from this tutorial on a production Linux server. This could result in security issues and downtime.

Step 1 — Creating Two User Accounts

First, we will create two user accounts. These accounts will remotely connect to our CentOS 7 server from VNC clients.

* joevnc
* janevnc

Run the following command to add a user account for **joevnc**:

sudo useradd -c "User Joe Configured for VNC Access" joevnc

Then run the passwd command to change **joevnc**'s password:

sudo passwd joevnc

The output will ask us for new password. Once supplied, the account will be ready for login:

Changing password for user joevnc.

New password:

Retype new password:

passwd: all authentication tokens updated successfully.

Next, create an account for **janevnc**:

sudo useradd -c "User Jane Configured for VNC Access" janevnc

Set the password for **janevnc**:

sudo passwd janevnc

Step 2 — Installing GNOME Desktop

Now we will install GNOME desktop. GNOME is a collaborative effort: it's a collection of free and open source software that makes up a very popular desktop environment. There are other desktop environments like KDE, but GNOME is more popular. Our VNC users will use GNOME to interact with the server from its desktop:

sudo yum groupinstall -y "GNOME Desktop"

Depending on the speed of your network, this can take a few minutes.

Once the package group is installed, reboot the server:

sudo reboot

**Troubleshooting — Server Stuck at Boot Phase**

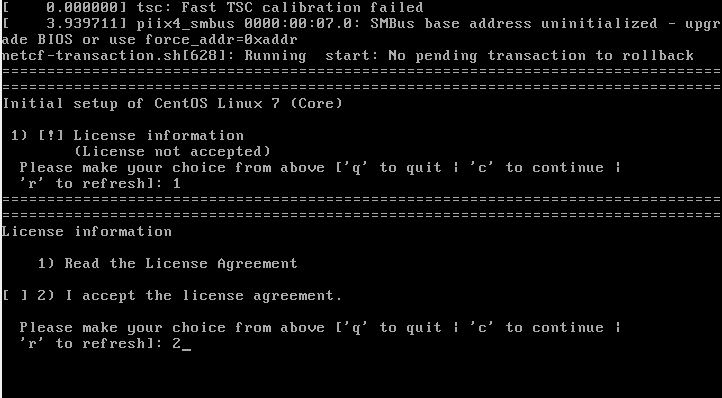
Depending on how your server has been set up, when the machine starts up it may remain in the boot phase showing a message like this:

Initial setup of CentOS Linux 7 (core)

1) [!] License information (Licence not accepted)

Please make your choice from above ['q' to quit | 'c' to continue | 'r' to refresh]:

To get past this, press **1** (license read), then **2** (accept licence), and then **C** (to continue). You may have to press **C** two or more times. The image below shows this:



If you don't see this error and the boot process is smooth, all the better – you can move on to the next step.

Step 3 — Installing TigerVNC Server

TigerVNC is the software that will allow us to make a remote desktop connection.

Install the Tiger VNC server:

sudo yum install -y tigervnc-server

This should show output like the following:

Loaded plugins: fastestmirror, langpacks

Loading mirror speeds from cached hostfile

. . .

Running transaction

Installing : tigervnc-server-1.2.80-0.30.20130314svn5065.el7.x86\_64 1/1

Verifying : tigervnc-server-1.2.80-0.30.20130314svn5065.el7.x86\_64 1/1

Installed:

tigervnc-server.x86\_64 0:1.2.80-0.30.20130314svn5065.el7

Complete!

Now we have VNC server and the GNOME desktop installed. We have also created two user accounts for connecting through VNC.

Step 4 — Configuring VNC Service for Two Clients

VNC server doesn't start automatically when it's first installed. To check this, run the following command:

sudo systemctl status vncserver@:.service

The output will be like this:

vncserver@:.service - Remote desktop service (VNC)

Loaded: loaded (/usr/lib/systemd/system/vncserver@.service; disabled)

Active: inactive (dead)

You can also run this command:

sudo systemctl is-enabled vncserver@.service

This should show output like this:

disabled

So why is it disabled? That's because each user will start a separate instance of the VNC service daemon. In other words, VNC doesn't run as one single process that serves every user request. Each user connecting via VNC will have to start a new instance of the daemon (or the system administrator can automate this).

CentOS 7 uses the systemd daemon to initiate other services. Each service that natively runs under systemd has a *service unit file* that's placed under the /lib/systemd/system directory by the yum installer. Processes that get started automatically at boot time have a link to this service unit file placed in the /etc/systemd/system/ directory.

In our case, a generic service unit file was created in the /lib/systemd/system/ directory, but no link was made under /etc/systemd/system/. To test this, run the following commands:

sudo ls -l /lib/systemd/system/vnc\*

You should see:

-rw-r--r--. 1 root root 1744 Jun 10 16:15 /lib/systemd/system/vncserver@.service

Then check under /etc/systemd/system/:

sudo ls -l /etc/systemd/system/\*.wants/vnc\*

This one doesn't exist:

ls: cannot access /etc/systemd/system/\*.wants/vnc\*: No such file or directory

So, the first step is to start two new instances of VNC server for our two users. To do this, we will need to make two copies of the generic VNC service unit file under /etc/system/system. In the code snippet below, you're making two copies with two different names:

sudo cp /lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver@:4.service

sudo cp /lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver@:5.service

So why did we add two numbers (along with the colon) in the copied file names?

Again, that comes back to the concept of individual VNC services. VNC by itself runs on port 5900. Since each user will run their own VNC server, each user will have to connect via a separate port. The addition of a number in the file name tells VNC to run that service as a *sub-port* of 5900. So in our case, **joevnc**'s VNC service will run on port 5904 (5900 + 4) and **janevnc**'s will run on 5905 (5900 + 5).

Next edit the service unit file for each client. Open the /etc/systemd/system/vncserver@:4.service file with the **vi** editor:

sudo vi /etc/systemd/system/vncserver@:4.service

A look at the "Quick HowTo" section tells us we have already completed the first step. Now we need to go through the remaining steps. The comments also tell us that VNC is a non-trusted connection. We will talk about this later.

For now, edit the [Service] section of the file, replacing instances of <USER> with joevnc. Also, add the -geometry 1280x1024 clause at the end of the ExecStart parameter. This just tells VNC the screen size it should start in. You will modify two lines in total. Here's what the edited file should look like (note that the entire file is not shown):

# The vncserver service unit file

#

# Quick HowTo:

# 1. Copy this file to /etc/systemd/system/vncserver@:<display>.service

# 2. Edit <USER> and vncserver parameters appropriately

# ("runuser -l <USER> -c /usr/bin/vncserver %i -arg1 -arg2")

# 3. Run `systemctl daemon-reload`

# 4. Run `systemctl enable vncserver@:<display>.service`

#

. . .

[Unit]

Description=Remote desktop service (VNC)

After=syslog.target network.target

[Service]

Type=forking

# Clean any existing files in /tmp/.X11-unix environment

ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'

ExecStart=/sbin/runuser -l joevnc -c "/usr/bin/vncserver %i -geometry 1280x1024"

PIDFile=/home/joevnc/.vnc/%H%i.pid

ExecStop=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'

[Install]

WantedBy=multi-user.target

Save the file and exit vi.

Similarly, open the /etc/systemd/system/vncserver@:5.service file in vi and make the changes for user **janevnc**:

sudo vi /etc/systemd/system/vncserver@:5.service

Here's just the [Service] section with the changes marked:

[Service]

Type=forking

# Clean any existing files in /tmp/.X11-unix environment

ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'

ExecStart=/sbin/runuser -l janevnc -c "/usr/bin/vncserver %i -geometry 1280x1024"

PIDFile=/home/janevnc/.vnc/%H%i.pid

ExecStop=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'

Next, run the following commands to reload the systemd daemon and also to make sure VNC starts up for two users at boot time.

sudo systemctl daemon-reload

Enable the first server instance:

sudo systemctl enable vncserver@:4.service

Output:

ln -s '/etc/systemd/system/vncserver@:4.service' '/etc/systemd/system/multi-user.target.wants/vncserver@:4.service'

Enable the second server instance:

sudo systemctl enable vncserver@:5.service

Output:

ln -s '/etc/systemd/system/vncserver@:5.service' '/etc/systemd/system/multi-user.target.wants/vncserver@:5.service'

Now you've configured two VNC server instances.

Step 5 — Configuring Your Firewall

Next, we will need to configure the firewall to allow VNC traffic through ports **5904** and **5905** only. CentOS 7 uses Dynamic Firewall through the firewalld daemon; the service doesn't need to restart for changes to take effect.

The firewalld service should start automatically at system boot time, but it's always good to check:

sudo firewall-cmd --state

This should show:

running

If the state is "not running" for any reason, execute the following command to make sure it's running:

sudo systemctl start firewalld

Now add the rules for ports 5904 and 5905:

sudo firewall-cmd --permanent --zone=public --add-port=5904-5905/tcp

Output:

success

Reload the firewall:

sudo firewall-cmd --reload

Output:

success

Step 6 — Setting VNC Passwords

We are one step away from seeing VNC in action. In this step, the users will need to set their **VNC passwords**. These are *not* the users' Linux passwords, but the passwords to log in to the VNC sessions.

Open another terminal connection to the CentOS 7 server, and this time log in as **joevnc**.

ssh joevnc@your\_server\_ip

Execute the following command:

vncserver

As shown in the output below, the server will ask **joevnc** to set up a VNC password. After typing in the password, the program also shows a number of files being created in the user's home directory:

You will require a password to access your desktops.

Password:

Verify:

xauth: file /home/joevnc/.Xauthority does not exist

New 'localhost.localdomain:1 (joevnc)' desktop is localhost.localdomain:1

Creating default startup script /home/joevnc/.vnc/xstartup

Starting applications specified in /home/joevnc/.vnc/xstartup

Log file is /home/joevnc/.vnc/localhost.localdomain:1.log

Let's look at the line New 'localhost.localdomain:1 (joevnc)' desktop is localhost.localdomain:1. **localhost.localdomain** was the server name in our example; in your case it could be different. Note the number after the server name: (**1**, separated by a colon). It's not the number in **joevnc**'s service unit file (which was **4**). That's because this is the *display number* **joevnc**'s session will run on in this server, not the port number of the service (5904) itself.

Next open a new terminal session and log in as **janevnc**. Here as well, start the VNC server and set a password for **janevnc**:

vncserver

You should see similar output showing that **janevnc**'s session will run on display **2**.

Finally, reload the services from the **main terminal session**:

sudo systemctl daemon-reload

sudo systemctl restart vncserver@:4.service

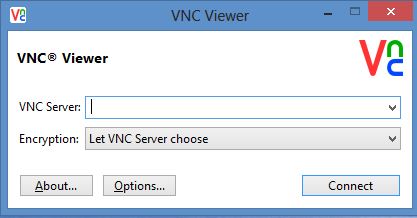
sudo systemctl restart vncserver@:5.service

Step 7 — Connecting to Remote Desktops with a VNC Client

For this tutorial, we will assume users **joevnc** and **janevnc** are trying to connect to the CentOS 7 server from their Windows computers.

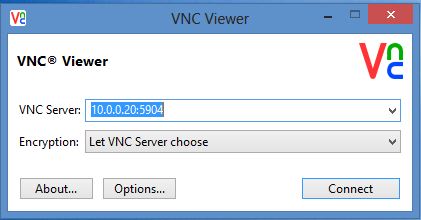
They will each need a VNC client for Windows to log into the remote desktop. This client is just like a terminal client like PuTTY, except it shows graphical output. There are various VNC client available, but the one we will use is RealVNC, available [here](https://www.realvnc.com/download/viewer/). VNC Viewer for Mac OS X is available for download on the same page, and the Mac version is fairly similar to the Windows one.

When VNC Viewer is started, it shows a dialogue box like this:

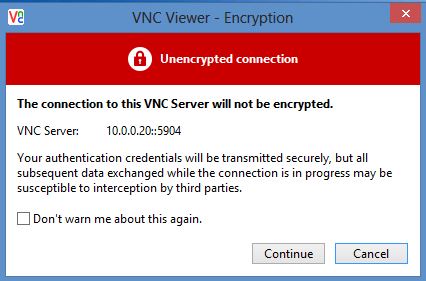


In the **VNC Server** field, add the IP address of your CentOS 7 server. Specify the port number 5904 after the server's IP, separate by a colon (**:**). We used 5904 because that's the VNC service port for **joevnc**.

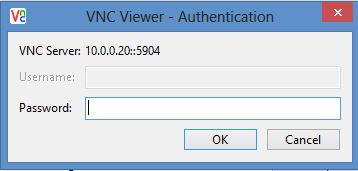
We have also decided to let VNC Viewer choose the encryption method. This option will only encrypt the password sent across the network. Any subsequent communication with the server will be unencrypted. (We'll set up a secure SSH tunnel in the final step.)



In fact, a warning message shows just that:

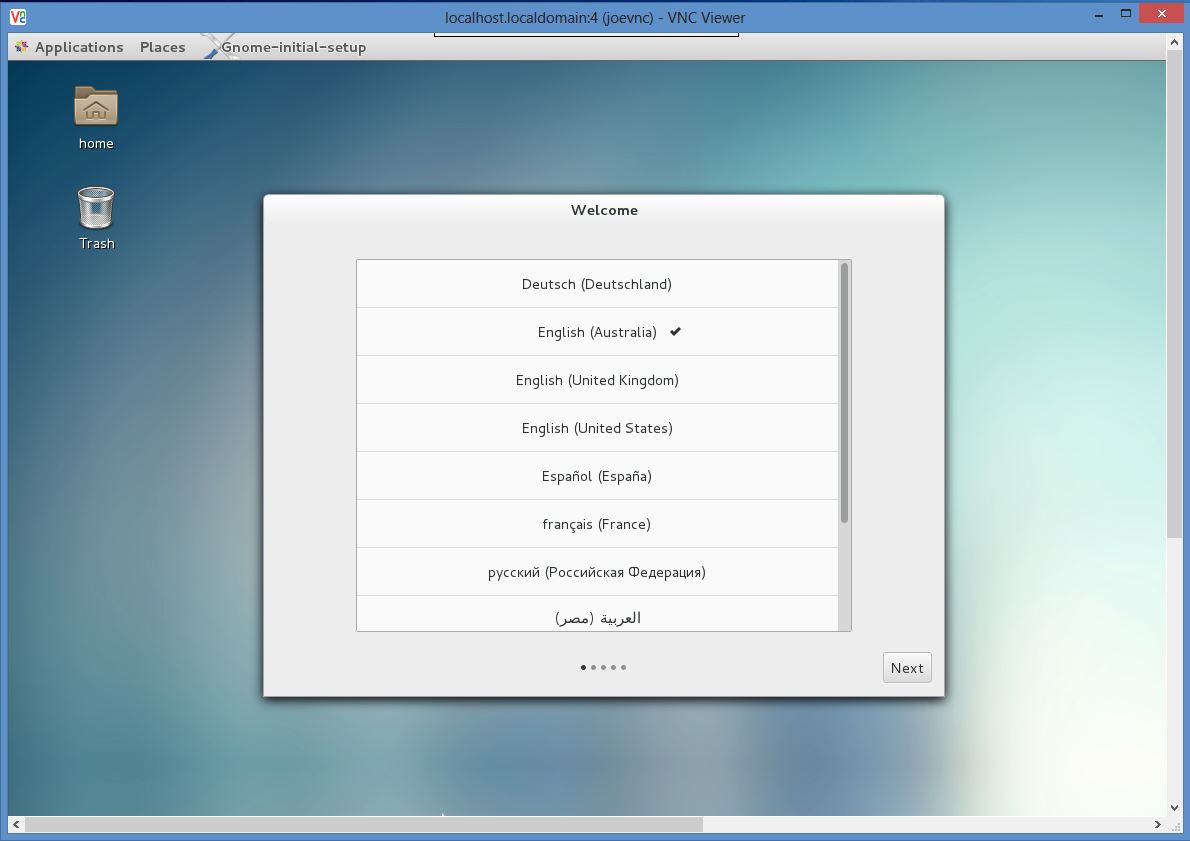


Accept the warning for now. A password prompt is displayed:



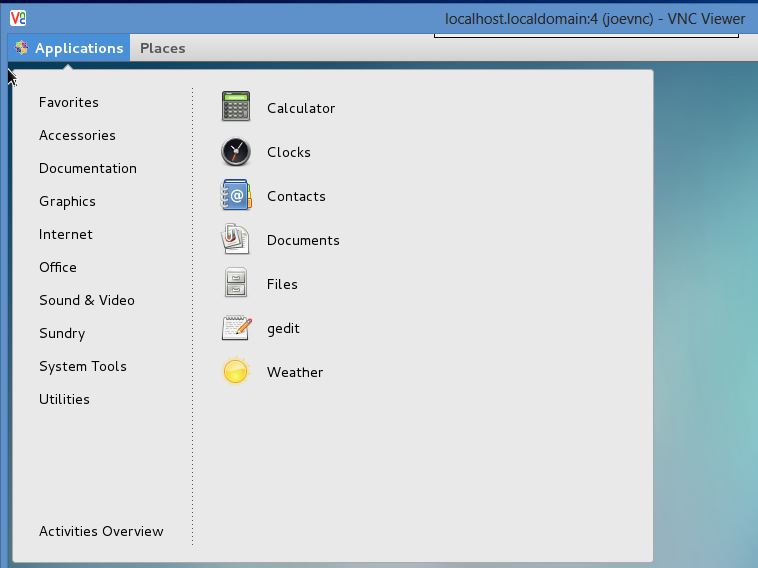
Enter **joevnc**'s VNC password that you set earlier.

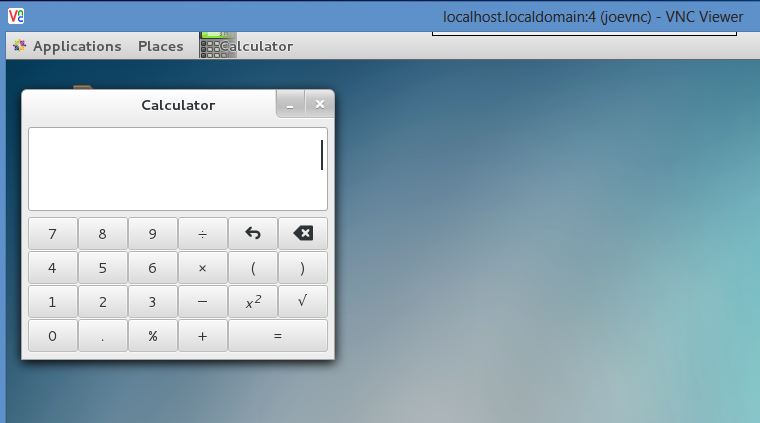
A new window opens showing the GNOME desktop for our remote CentOS server:



Accept the default welcome message.

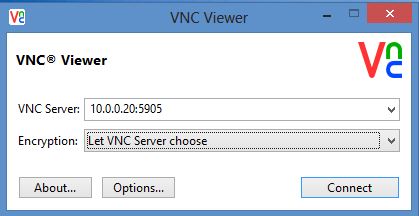
Now **joevnc** can start a graphical tool like the GNOME calculator:





You can leave this desktop connection open.

Now **janevnc** can also start another VNC session with the CentOS server. The IP address is the same, and the port is 5905:



When **janevnc** logs in via VNC Viewer, an empty desktop with a welcome message is shown, just like it was shown for **joevnc**. In other words, the two users are not sharing the desktop instances. **joevnc**'s desktop should still be showing the calculator.

To close the remote desktop session, simply closing the window will do. However, this doesn't stop the user's VNC service in the background on the server. If the service is not stopped or restarted and the machine had no reboots, the same desktop session would be presented at the next logon.

Close the VNC Viewer windows for **joevnc** and **janevnc**. Close their terminal sessions, too. From the main terminal window, check to see if the VNC services are still running:

sudo systemctl status vncserver@:4.service

The output shows that the remote desktop is still running:

vncserver@:4.service - Remote desktop service (VNC)

Loaded: loaded (/etc/systemd/system/vncserver@:4.service; enabled)

Active: active (running) since Sat 2014-11-01 12:06:49 EST; 58min ago

Process: 2014 ExecStart=/sbin/runuser -l joevnc -c /usr/bin/vncserver %i -geometry 1280x1024 (code=exited, status=0/SUCCESS)

. . .

Check the second service:

sudo systemctl status vncserver@:5.service

This one is running, too:

vncserver@:5.service - Remote desktop service (VNC)

Loaded: loaded (/etc/systemd/system/vncserver@:5.service; enabled)

Active: active (running) since Sat 2014-11-01 12:42:56 EST; 22min ago

Process: 3748 ExecStart=/sbin/runuser -l janevnc -c /usr/bin/vncserver %i -geometry 1280x1024 (code=exited, status=0/SUCCESS)

. . .

If you wanted to log back into **joevnc**'s desktop at this point, you'd see the same calculator app open.

This presents some interesting challenges for system administrators. If you have a number of users connecting to the server via VNC, you may want to devise some way to stop their VNC services when no longer needed. This may save some valuable system resources.

**Troubleshooting — VNC Service Crashes**

As you test and play around with VNC, you may sometimes find the service has crashed and is unrecoverable. When you try to check the status:

sudo systemctl status vncserver@:4.service

This long error message may come up:

vncserver@:4.service - Remote desktop service (VNC)

Loaded: loaded (/etc/systemd/system/vncserver@:4.service; enabled)

Active: failed (Result: exit-code) since Fri 2014-11-07 00:02:38 EST; 2min 20s ago

Process: 2221 ExecStart=/sbin/runuser -l joevnc -c /usr/bin/vncserver %i -geometry 1280x1024 (code=exited, status=2)

Process: 1257 ExecStartPre=/bin/sh -c /usr/bin/vncserver -kill %i > /dev/null 2>&1 || : (code=exited, status=0/SUCCESS)

Trying to start the service doesn't work:

sudo systemctl start vncserver@:4.service

Failed startup:

Job for vncserver@:4.service failed. See 'systemctl status vncserver@:4.service' and 'journalctl -xn' for details.

Usually the reason is simple enough. Check /var/log/messages:

sudo tail /var/log/messages

The related error will look like this:

Nov 7 00:08:36 localhost runuser: Warning: localhost.localdomain:4 is taken because of /tmp/.X11-unix/X4

Nov 7 00:08:36 localhost runuser: Remove this file if there is no X server localhost.localdomain:4

Nov 7 00:08:36 localhost runuser: A VNC server is already running as :4

Nov 7 00:08:36 localhost systemd: vncserver@:4.service: control process exited, code=exited status=2

Nov 7 00:08:36 localhost systemd: Failed to start Remote desktop service (VNC).

Nov 7 00:08:36 localhost systemd: Unit vncserver@:4.service entered failed state.

Nov 7 00:08:36 localhost systemd: Failed to mark scope session-c3.scope as abandoned : Stale file handle

The remedy is to delete the file under /tmp folder:

sudo rm -i /tmp/.X11-unix/X4

Output:

rm: remove socket ‘/tmp/.X11-unix/X4’? y

Then start the VNC service:

sudo systemctl start vncserver@:4.service

**General Troubleshooting**

Although relatively rare, you may encounter other errors when working with VNC. For example, your remote desktop screen can go blank or hang, the session might crash with a cryptic error message, VNC Viewer may not connect properly or transmit commands to the GUI to launch applications, etc.

We recommend checking the /var/log/messages file to get a better understanding. At times you may need to reboot your server, or in extreme cases recreate the VNC service.

System resources can also be a culprit; you may have to add extra RAM to your Droplet, etc.

Step 8 — Securing VNC Sessions through SSH Tunneling

So far both **joevnc** and **janevnc** have been accessing their remote desktops through unencrypted channels. As we saw before, VNC Viewer warns us about this at connection time; only the password is encrypted as the sessions begins. Any subsequent network traffic and data transfer is open for anyone to intercept in the middle.

**About SSH Tunnelling**

This is where Secure Shell (SSH) sessions can help. With SSH, VNC can run within the context of an SSH encrypted session. This is known as *tunnelling*. In effect, VNC traffic piggybacks on the SSH protocol, resulting in all of its communication with the server being encrypted. It's called *tunnelling* because SSH is providing wraparound protection over VNC and VNC is running as if in a tunnel within SSH. SSH tunnelling can be used for other protocols like POP, X, or IMAP as well.

SSH tunnelling works with *port forwarding* which is basically a means of translating access from one particular port to a different port on another machine. With port forwarding, when a client application connects to Port A running on machine A, it's transparently forwarded to port B running on machine B. The client application is unaware of this translation and thinks it's connecting to the original port. Port forwarding is one of the features of SSH protocol.

For more detailed information about SSH tunneling, read [this tutorial](https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-tunneling-on-a-vps).

In this tutorial we have configured VNC to run on ports 5904 (for **joevnc**) and 5905 (for **janevnc**).

With port forwarding, we can set our local VNC client to connect to port **5900** on the local client computer, and this can be mapped to port **5905** on the remote server. This is example is for **janevnc**'s connection, but you could easily follow the same steps for any other clients.

When the VNC client application starts, it can be pointed to port **5900** on **localhost**, and our port forwarding will transparently transport it to port **5905** on the remote server.

**Note:** You'll have to start an SSH section **each time** to make the connection secure.

**OS X**

On your Mac, open **Terminal**.

Enter the following connection information, being sure to replace your\_server\_ip with your remote server's IP address:

ssh -L 5900:your\_server\_ip:5905 janevnc@your\_server\_ip -N

Enter **janevnc**'s UNIX password. The connection will appear to hang; you can keep it running for as long as you use the remote desktop.

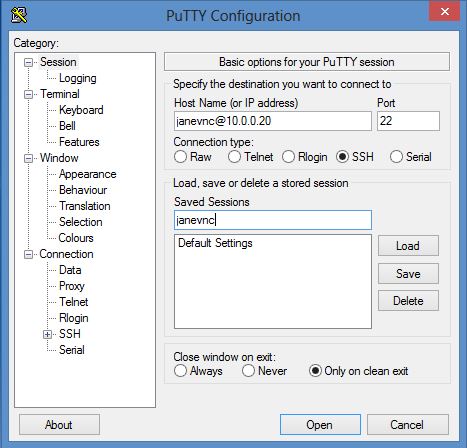
Now skip ahead to the VNC Viewer instructions.

**Windows**

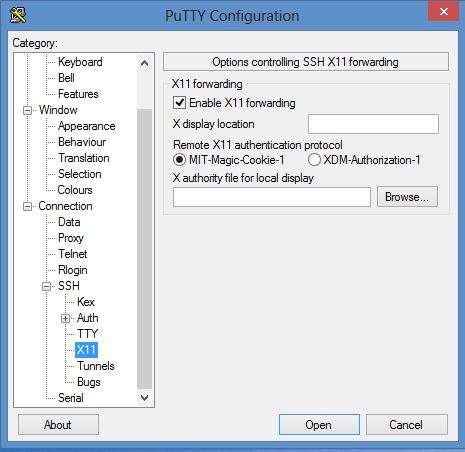
For securing **janevnc**'s VNC session, we will assume the local Windows computer has PuTTY installed. PuTTY is free and can be downloaded from [here](http://www.putty.org/).

If **janevnc**'s VNC and terminal sessions are not closed already, close them now.

Start PuTTY. In the session screen, ensure you specify the server IP address and give a descriptive name to the connection, then click the **Save** button to save the connection details. Note how we have specified username@your\_server\_ip in the **Hostname** field:



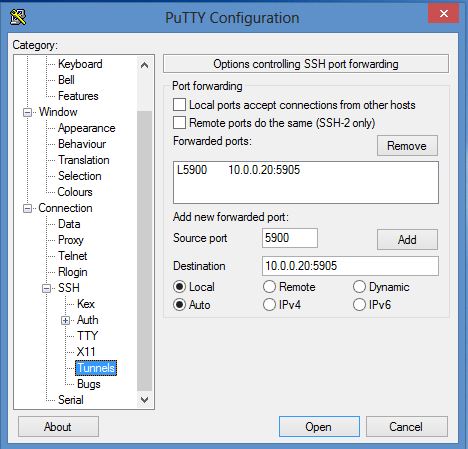
Next, expand the **SSH** menu item in the left navigation pane, and select the **X11** item. This shows the X11 forwarding properties for the session. Ensure the checkbox for **Enable X11 forwarding** is checked. This ensures that SSH encrypts X Windows traffic that flows between the server and client:



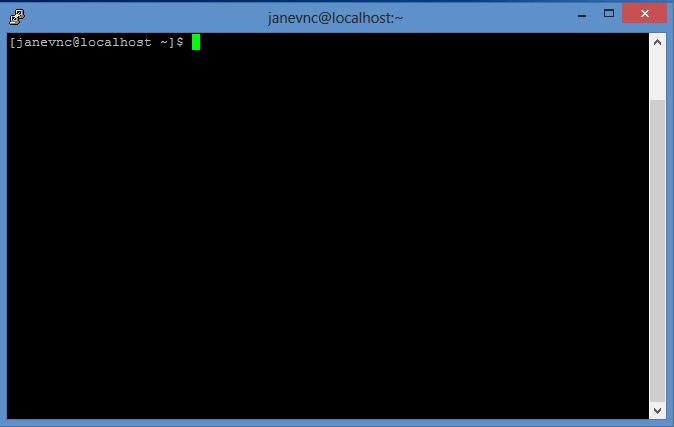
Finally, select **SSH > Tunnels**. Type 5900 in the **Source** port field. In the **Destination** field, specify your server's name or IP address, followed by a colon and the VNC port number for the intended user. In our case, we have specified your\_server\_ip:5905.

Alternately, you could use port **5902**. The **2** in this case would be the display number for **janevnc**(remember the message displayed when **janevnc** ran the vncserver command).

Click the **Add** button and the mapping will be added under **Forwarded ports**. This is where we are adding port forwarding for the SSH session; when the user connects to localhost at port **5900**, the connection will be automatically tunnelled through SSH to the remote server's port **5905**.

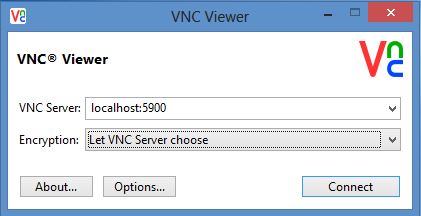


Go back to the **Sessions** items and save the session for **janevnc**. Click the **Open** button and a new terminal session will open for **janevnc**. Log in as **janevnc** with the appropriate UNIX password:



**VNC Viewer**

Next start VNC Viewer again. This time, in the **VNC Server** address, type <^> and let VNC server choose the encryption method:

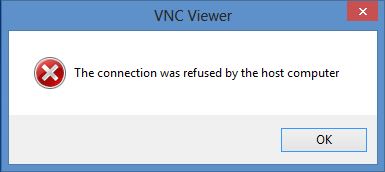


Click the **Connect** button.

You will still get the dialogue box warning you about an unencrypted session, but this time you can safely ignore it. VNC Viewer doesn't know about the port it's being forwarded to (this was set in the SSH session just started) and assumes you are trying to connect to the local machine.

Accepting this warning will show the familiar password prompt. Enter **janevnc**'s VNC password to access the remote desktop.

So how do you know the session was encrypted? If you think about it, we had set port forwarding in the SSH session. If an SSH session wasn't established, port forwarding wouldn't have worked. In fact, if you close the terminal window and log out of the PuTTY session then try to connect with VNC Viewer alone, a connection attempt to localhost:5900 would show the following error message:



So, if the localhost:5900 connection works, you can be confident that the connection is encrypted.

Remember that you will want to establish the SSH connection first every time you use VNC, to make sure your connection is always encrypted.

**Conclusion**

Accessing your CentOS Linux system from a GUI front end can make system administration much simpler. You can connect from any client operating system and don't have to depend on web-based hosting control panels. VNC has a much smaller footprint compared to most control panels.

Although we have shown how two ordinary users can connect with their VNC clients, that's hardly practical in serious production environments. In reality, users will have customized applications or browsers for accessing the server. Running a number of VNC services for each user also creates an unnecessary burden on system resources, not to mention the inherent risks associated with it.

If you decide to install and run VNC on your production Linux server, we strongly recommend using it for administrative purposes only.