How to Install WildFly JAVA Application Server on Ubuntu 22.04

March 28, 2024 by Hitesh Jethva

WildFly, formerly JBoss, is a free and open-source application server that helps you build and deploy Java web applications. It's written in Java and compliant with Java EE (Enterprise Edition) specifications. WildFly was created by JBoss but is now developed by RedHat. It is a powerful, production-ready, modular, and lightweight application server that provides all necessary tools and features to run and deploy Java web applications.

WildFly is a cross-platform application server. It provides tools such as EJBs, JPA, Servlets, JAX-RS, Batch, and security for your Java applications.

This guide will help you install WildFly on the Ubuntu 22.04 server. You will manually install WildFly from the Binary package on the Ubuntu system.

Prerequisites

To complete this tutorial, you will need a **Ubuntu 22.04** Server. This server should have a non-root user with root/administrator privileges. Additionally, it's recommended that you enable the "**UFW**" firewall on your Ubuntu system.

Installing Java OpenJDK

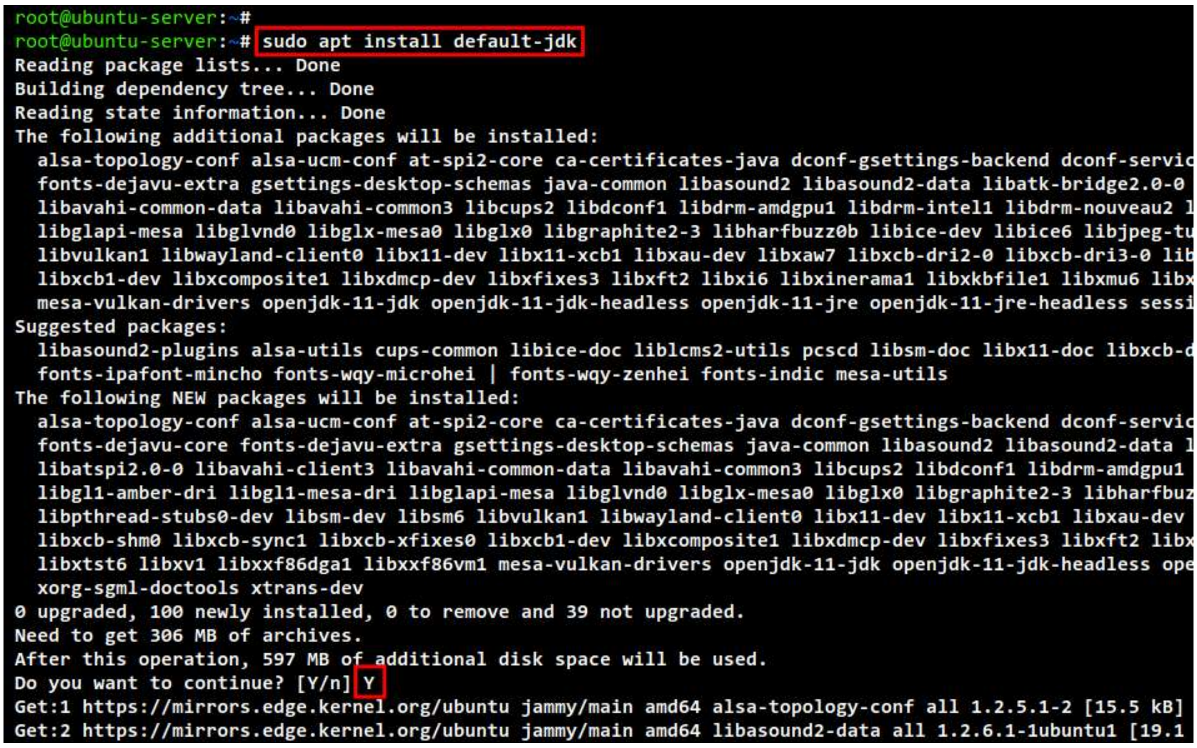
Before starting to install WildFly, you will install the Java OpenJDK first on your Ubuntu server. At the time of this writing, the latest version of WildFly required at least Java v8. The default Ubuntu APT repository provides Java v11, which is suitable for the latest version of WildFly installation.

Run the "*apt*" command below to update and refresh your Ubuntu package index.

```
sudo apt update
```

To install Java OpenJDK, run the "*apt install*" command below. You will be prompted to confirm the installation, input **Y**, and press **ENTER** to continue, and the Java OpenJDK installation will begin.

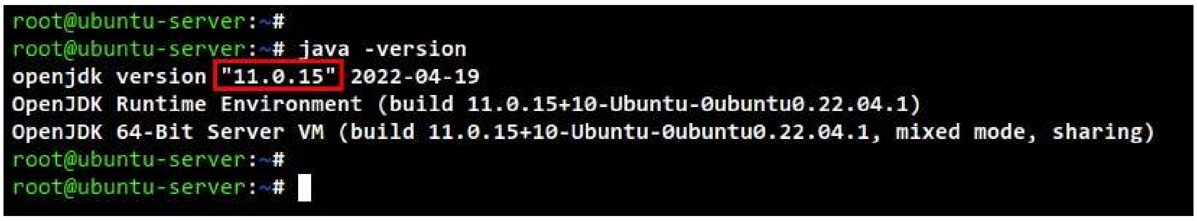
```
sudo apt install default-jdk
```



```
root@ubuntu-server:~#
root@ubuntu-server:~# sudo apt install default-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-servic
  fonts-dejavu-extra gsettings-desktop-schemas java-common libasound2 libasound2-data libatk-bridge2.0-0
  libavahi-common-data libavahi-common3 libcups2 libdconf1 libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 l
  libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev libice6 libjpeg-tu
  libvulkan1 libwayland-client0 libx11-dev libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 lib
  libxcb1-dev libxcomposite1 libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libx
  mesa-vulkan-drivers openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless sessi
Suggested packages:
  libasound2-plugins alsa-utils cups-common libice-doc liblcms2-utils pcscd libsm-doc libx11-doc libxcb-d
  fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic mesa-utils
The following NEW packages will be installed:
  alsa-topology-conf alsa-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-servic
  fonts-dejavu-core fonts-dejavu-extra gsettings-desktop-schemas java-common libasound2 libasound2-data l
  libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libdconf1 libdrm-amdgpu1
  libgl1-amber-dri libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuz
  libpthread-stubs0-dev libsm-dev libsm6 libvulkan1 libwayland-client0 libx11-dev libx11-xcb1 libxau-dev
  libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcomposite1 libxdmcp-dev libxfixes3 libxft2 libx
  libxtst6 libxv1 libxxf86dga1 libxxf86vm1 mesa-vulkan-drivers openjdk-11-jdk openjdk-11-jdk-headless ope
  xorg-sgml-doctools xtrans-dev
0 upgraded, 100 newly installed, 0 to remove and 39 not upgraded.
Need to get 306 MB of archives.
After this operation, 597 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://mirrors.edge.kernel.org/ubuntu jammy/main amd64 alsa-topology-conf all 1.2.5.1-2 [15.5 kB]
Get:2 https://mirrors.edge.kernel.org/ubuntu jammy/main amd64 libasound2-data all 1.2.6.1-1ubuntu1 [19.1
```

Now that Java OpenJDK is installed, check and verify the Java version using the following command. You should see that **Java OpenJDK 11** is installed on your Ubuntu 22.04 server.

```
java -version
```



```
root@ubuntu-server:~#
root@ubuntu-server:~# java -version
openjdk version "11.0.15" 2022-04-19
OpenJDK Runtime Environment (build 11.0.15+10-Ubuntu-0ubuntu0.22.04.1)
OpenJDK 64-Bit Server VM (build 11.0.15+10-Ubuntu-0ubuntu0.22.04.1, mixed mode, sharing)
root@ubuntu-server:~#
root@ubuntu-server:~#
```

Installing WildFly from Binary Files

After installing Java, you will install WildFly manually on your Ubuntu server. The manual WildFly installation requires you to set up a new system user, download the WildFLy Binary Package, and set up the configuration directory and systemd service for WildFly.

To create a new system user, run the "*useradd*" command below.

In the following example, the new system user will be named "**wildfly**" with the base home directory "*/opt/wildfly*". This directory also will be used as the main installation directory for WildFly.

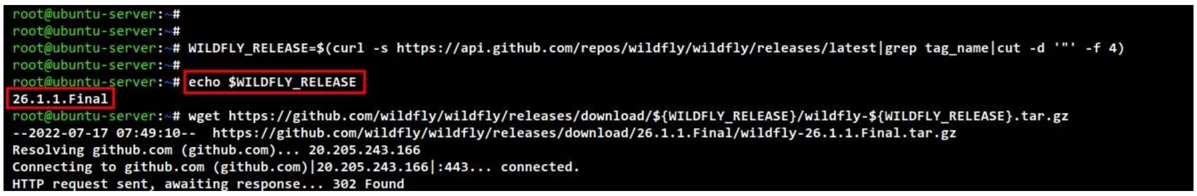
```
sudo useradd -r -d /opt/wildfly -s /usr/sbin/nologin wildfly
```

Before downloading WildFly binary package, run the following command to create a new temporary environment variable "**WILDFLY_RELEASE**". This will assign the environment variable "**WILDFLY_RELEASE**" with the latest release of the WildFly version. At the time of this writing, the latest version of WildFly is **v26.x**.

```
WILDFLY_RELEASE=$(curl -s
https://api.github.com/repos/wildfly/wildfly/releases/latest|grep tag_name|cut -
d '"' -f 4)
echo $WILDFLY_RELEASE
```

To download the WildFly Binary Package, run the "*wget*" command below. You should see the new file in your current directory.

```
wget
https://github.com/wildfly/wildfly/releases/download/${WILDFLY_RELEASE}/wildfly-
${WILDFLY_RELEASE}.tar.gz
```



```
root@ubuntu-server:~#
root@ubuntu-server:~#
root@ubuntu-server:~# WILDFLY_RELEASE=$(curl -s https://api.github.com/repos/wildfly/wildfly/releases/latest|grep tag_name|cut -d '"' -f 4)
root@ubuntu-server:~#
root@ubuntu-server:~# echo $WILDFLY_RELEASE
26.1.1.Final
root@ubuntu-server:~# wget https://github.com/wildfly/wildfly/releases/download/${WILDFLY_RELEASE}/wildfly-${WILDFLY_RELEASE}.tar.gz
--2022-07-17 07:49:10--  https://github.com/wildfly/wildfly/releases/download/26.1.1.Final/wildfly-26.1.1.Final.tar.gz
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
```

Next, extract the WildFLy package using the "*tar*" command below. Then, move the new extracted WildFly directory to *"/opt/wildfly"*.

```
tar -xvf wildfly-${WILDFLY_RELEASE}.tar.gz
sudo mv wildfly-${WILDFLY_RELEASE} /opt/wildfly
```

Set the correct and appropriate ownership of the WldFly installation directory to the user "**wildfly**" using the below command.

```
sudo chown -R wildfly:wildfly /opt/wildfly
```

After that, create a new default configuration directory "*/etc/wildfly*" using the following command. Then, copy the example of the WildFly configuration file into it.

```
sudo mkdir -p /etc/wildfly/
sudo cp sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.conf
/etc/wildfly/
```

Edit the default WildFly config file "*/etc/wildfly/wildfly.conf*" using the below command.

```
sudo nano /etc/wildfly/wildfly.conf
```

You can change the "**WILDFLY_BIND**" address to your local server IP address. Also, in this demonstration, you will be running WildFly in the "**standalone**" mode with the config file "**standalone.xml**".

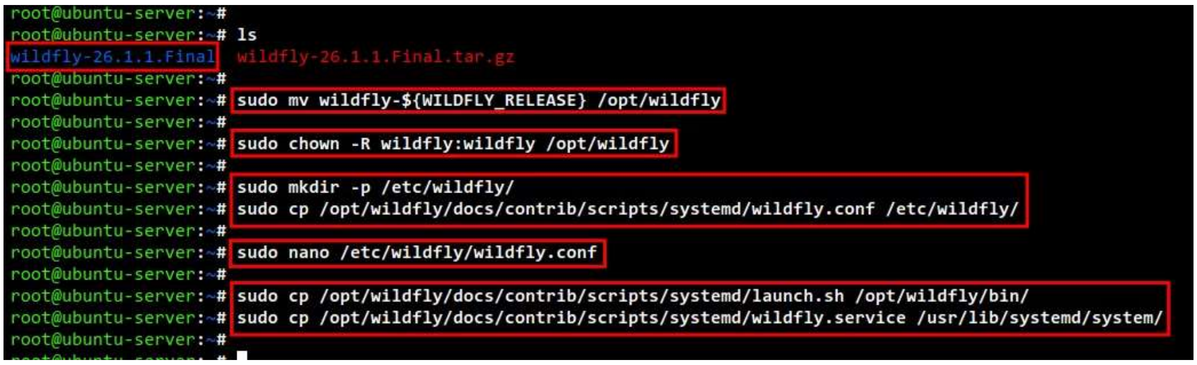
```
# The configuration you want to run
WILDFLY_CONFIG=standalone.xml
# The mode you want to run
WILDFLY_MODE=standalone

# The address to bind to
WILDFLY_BIND=192.168.5.10
```

Save and close the file when you are done.

Next, copy the example of WildFly start script "*/opt/wildfly/docs/contrib/scripts/systemd/launch.sh*" to the directory "*/opt/wildfly/bin/*". Then, copy the WildFly systemd service file "*/opt/wildfly/docs/contrib/scripts/systemd/wildfly.service*" to the directory "*/usr/lib/systemd/system/*".

```
sudo cp /opt/wildfly/docs/contrib/scripts/systemd/launch.sh /opt/wildfly/bin/
sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.service
/usr/lib/systemd/system/
```



```
root@ubuntu-server:~#
root@ubuntu-server:~# ls
wildfly-26.1.1.Final  wildfly-26.1.1.Final.tar.gz
root@ubuntu-server:~#
root@ubuntu-server:~# sudo mv wildfly-${WILDFLY_RELEASE} /opt/wildfly
root@ubuntu-server:~#
root@ubuntu-server:~# sudo chown -R wildfly:wildfly /opt/wildfly
root@ubuntu-server:~#
root@ubuntu-server:~# sudo mkdir -p /etc/wildfly/
root@ubuntu-server:~# sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.conf /etc/wildfly/
root@ubuntu-server:~#
root@ubuntu-server:~# sudo nano /etc/wildfly/wildfly.conf
root@ubuntu-server:~#
root@ubuntu-server:~# sudo cp /opt/wildfly/docs/contrib/scripts/systemd/launch.sh /opt/wildfly/bin/
root@ubuntu-server:~# sudo cp /opt/wildfly/docs/contrib/scripts/systemd/wildfly.service /usr/lib/systemd/system/
root@ubuntu-server:~#
```

Additionally, you will also need to create a new directory "*/var/run/wildfly*" that will be used for storing the PID (Process ID) file of the WildFly server. Also, be sure to change the correct ownership of that directory to the user "**wildfly**".

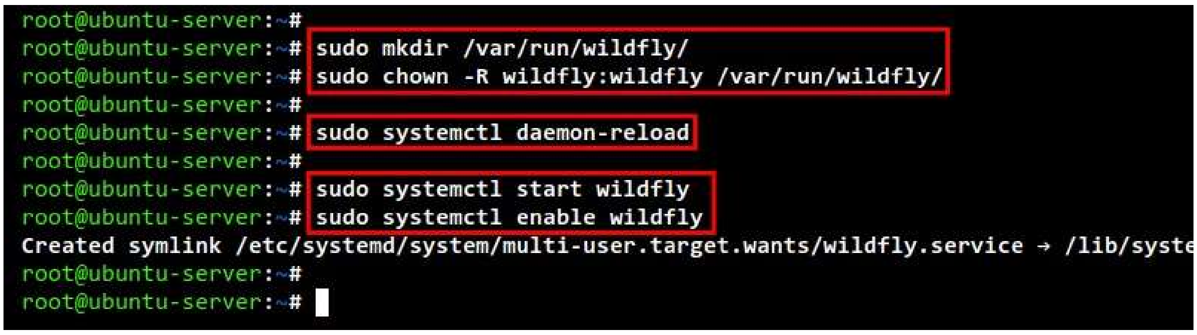
```
sudo mkdir /var/run/wildfly/
sudo chown -R wildfly:wildfly /var/run/wildfly/
```

Now run the "*systemctl*" command below to reload the systemd manager and apply the new WildFly service file.

```
sudo systemctl daemon-reload
```

After that, start and enable the "*wildfly*" service using the following "*systemctl*" command. And the "*wildfly*" service will be up and running, and it will be run automatically at system boot.

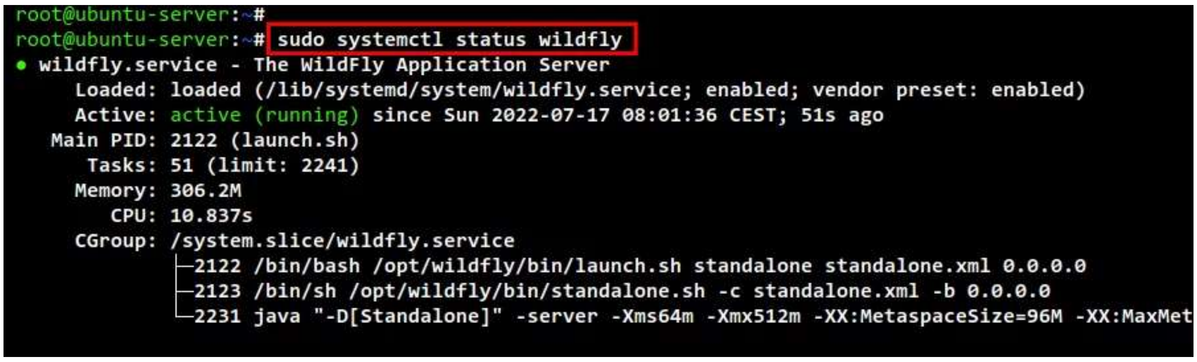
```
sudo systemctl start wildfly
sudo systemctl enable wildfly
```



```
root@ubuntu-server:~#
root@ubuntu-server:~# sudo mkdir /var/run/wildfly/
root@ubuntu-server:~# sudo chown -R wildfly:wildfly /var/run/wildfly/
root@ubuntu-server:~#
root@ubuntu-server:~# sudo systemctl daemon-reload
root@ubuntu-server:~#
root@ubuntu-server:~# sudo systemctl start wildfly
root@ubuntu-server:~# sudo systemctl enable wildfly
Created symlink /etc/systemd/system/multi-user.target.wants/wildfly.service → /lib/syste
root@ubuntu-server:~#
root@ubuntu-server:~#
```

To check and verify the "*wildfly*" service status, run the below command. And you should see the "wildfly" service is "*enabled*" and the current status is 'running".

```
sudo systemctl status wildfly
```

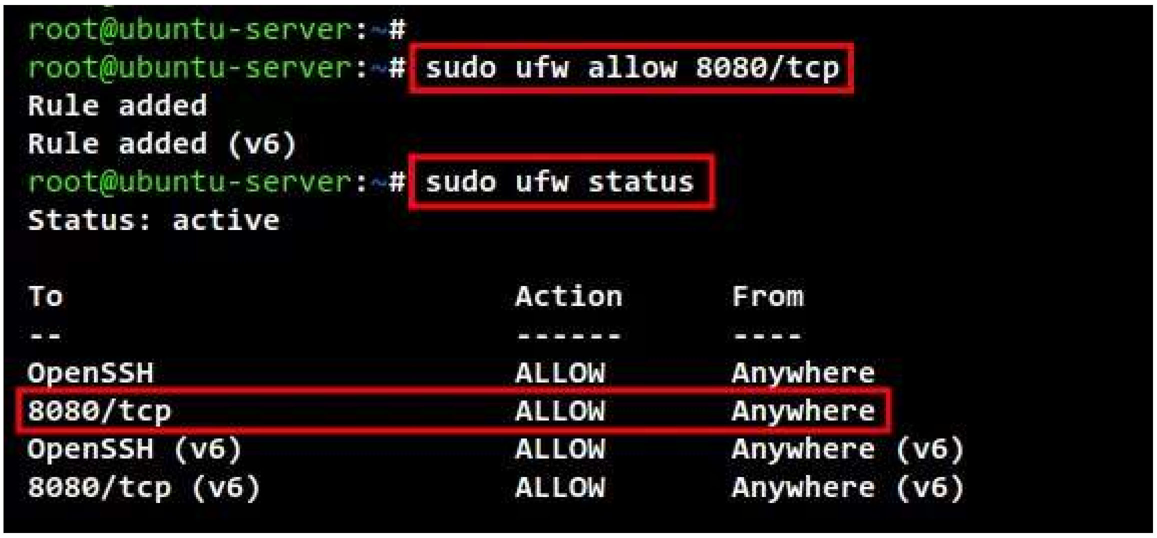


```
root@ubuntu-server:~#
root@ubuntu-server:~# sudo systemctl status wildfly
● wildfly.service - The WildFly Application Server
     Loaded: loaded (/lib/systemd/system/wildfly.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-07-17 08:01:36 CEST; 51s ago
   Main PID: 2122 (launch.sh)
      Tasks: 51 (limit: 2241)
     Memory: 306.2M
        CPU: 10.837s
     CGroup: /system.slice/wildfly.service
             ├─2122 /bin/bash /opt/wildfly/bin/launch.sh standalone standalone.xml 0.0.0.0
             ├─2123 /bin/sh /opt/wildfly/bin/standalone.sh -c standalone.xml -b 0.0.0.0
             └─2231 java "-D[Standalone]" -server -Xms64m -Xmx512m -XX:MetaspaceSize=96M -XX:MaxMet
```

By default, WildFly runs on the TCP port "8080." You must open that port if you have the "UFW" firewall enabled on your server.

Run the "*ufw*" command below to open the TCP port "*8080*" for WildFly. Then, check and verify the UFW firewall status. You should see that port "8080" has been added.

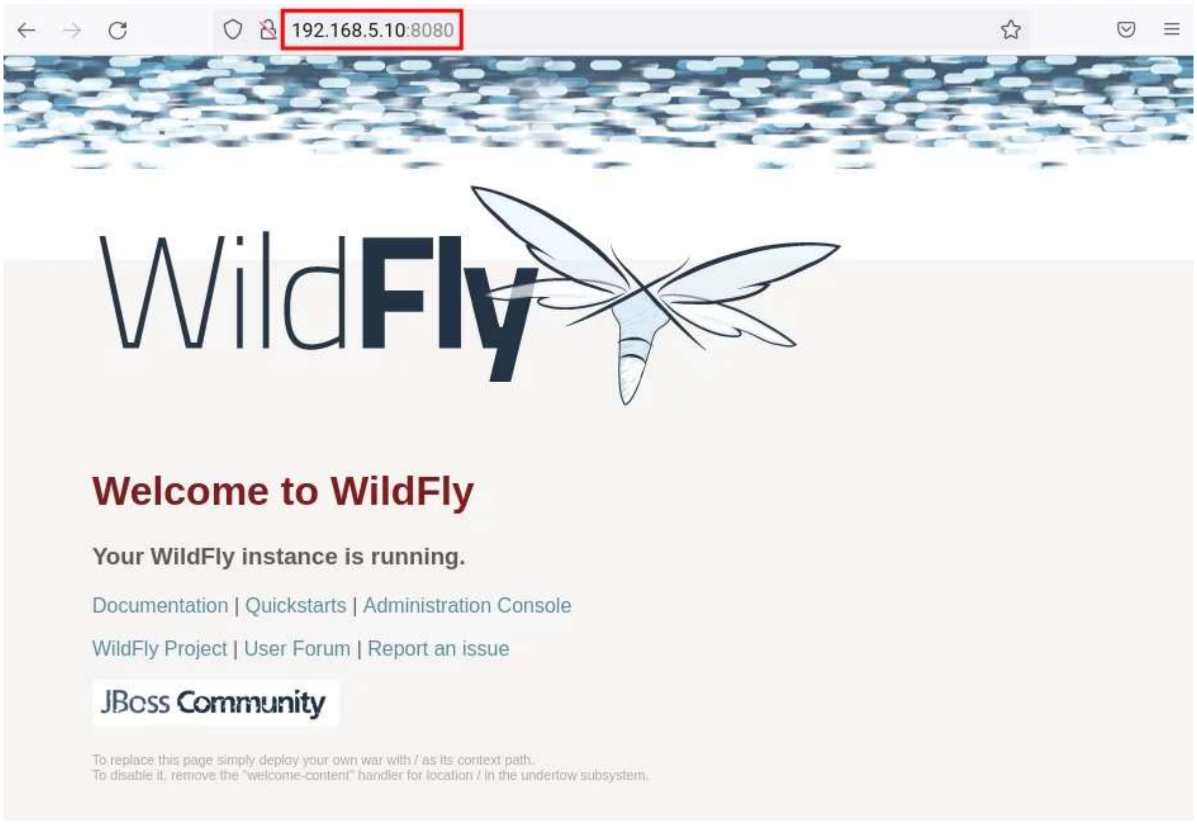
```
sudo ufw allow 8080/tcp
sudo ufw status
```



```
root@ubuntu-server:~#
root@ubuntu-server:~# sudo ufw allow 8080/tcp
Rule added
Rule added (v6)
root@ubuntu-server:~# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
```

Open your web browser and visit the server IP address with the port "8080" (e.g., http:/192.168.5.10:8080/). You will see WildFly's default index.html page.


192.168.5.10:8080

Welcome to WildFly

Your WildFly instance is running.

Documentation | Quickstarts | Administration Console

WildFly Project | User Forum | Report an issue

JBoss Community

To replace this page simply deploy your own war with / as its context path.
To disable it, remove the "welcome-content" handler for location / in the undertow subsystem.

Setting Up Administration User WildFly

The WildFly administration is disabled on the default installation. In this demo, you will enable the WildFly Administration dashboard that allows you to set up application deployment, runtime configuration, access control, and other WildFly settings.

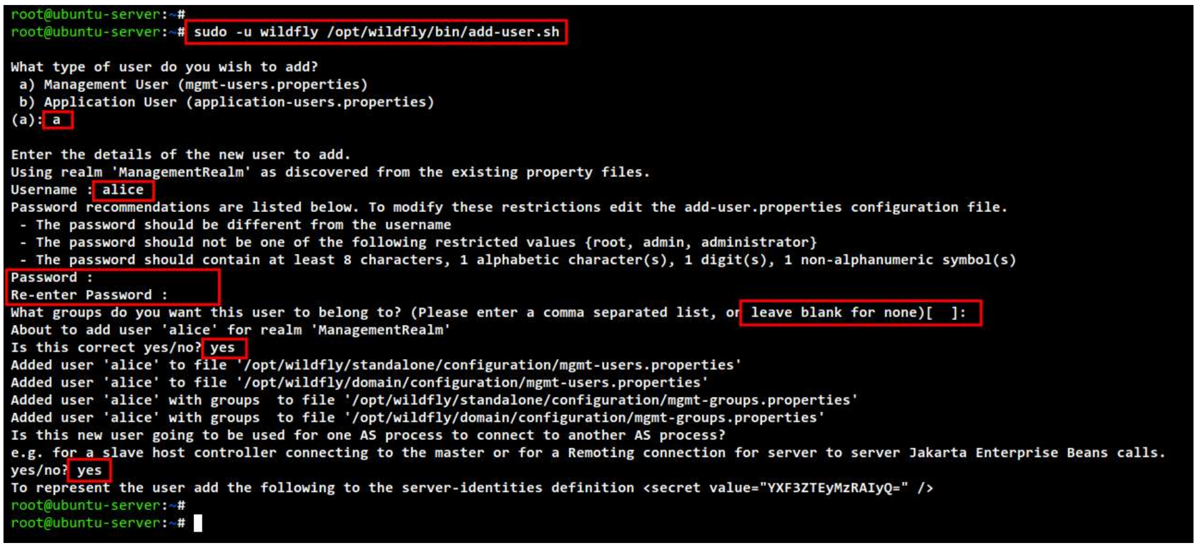
You will set up a new admin user for WildFly before you start configuring and enabling the WildFly administration.

Run the WildFly add user script *"/opt/wildfly/bin/add-user.sh"* as "**wildfly**" user below.

```
sudo -u wildfly /opt/wildfly/bin/add-user.sh
```

Now you will be prompted for the admin user configuration below:

- Input "**a**" to set up the WildFly "**Management User**".
- Input the username for the WildFly administrator. In this demo, we will use the admin user "**alice**".
- Input the password for the new admin user and repeat. Be sure to use the strong password as recommended by the script.
- Leave the group configuration as default and press **ENTER**.
- Input "**yes**" to confirm the new WildFly admin configurations.
- Lastly, input "**yes**" again to enable the new user to be used to identify one WildFly process or you can just input "no" for this settings.



```
root@ubuntu-server:~#
root@ubuntu-server:~# sudo -u wildfly /opt/wildfly/bin/add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : alice
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
 - The password should be different from the username
 - The password should not be one of the following restricted values {root, admin, administrator}
 - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]:
About to add user 'alice' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'alice' to file '/opt/wildfly/standalone/configuration/mgmt-users.properties'
Added user 'alice' to file '/opt/wildfly/domain/configuration/mgmt-users.properties'
Added user 'alice' with groups  to file '/opt/wildfly/standalone/configuration/mgmt-groups.properties'
Added user 'alice' with groups  to file '/opt/wildfly/domain/configuration/mgmt-groups.properties'
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server Jakarta Enterprise Beans calls.
yes/no? yes
To represent the user add the following to the server-identities definition <secret value="YXF3ZTEyMzRAIyQ=" />
root@ubuntu-server:~#
root@ubuntu-server:~#
```

Enabling WildFly Administration Console

If you created the WildFly administrator user, now you will be setting up the WildFly Administration console via the configuration file "*/etc/wildfly/wildfly.conf*".

Edit the config file "*/etc/wildfly/wildfly.conf*" using the following command.

```
sudo nano /etc/wildfly/wildfly.conf
```

Add the following configuration to the file. This will create a new environment variable for the WildFly service named "**WILDFLY_CONSOLE_BIND**", which will run the WildFly administration console on the internal IP address "**192.168.5.10**".

```
# IP Address for WildFly Console
WILDFLY_CONSOLE_BIND=192.168.5.10
```

Save and close the file when you are done.

Next, edit the WildFly start script *"/opt/wildfly/bin/launch.sh*" using the following command.

```
sudo nano /opt/wildfly/bin/launch.sh
```

Change the start command as below. Add option "*-bmanagement $4*" to enable the WildFly administration console.

```
if [[ "$1" == "domain" ]]; then
    $WILDFLY_HOME/bin/domain.sh -c $2 -b $3 -bmanagement $4
else
    $WILDFLY_HOME/bin/standalone.sh -c $2 -b $3 -bmanagement $4
fi
```

Save and close the file when you are done.

Now edit the WildFly systemd service file "*/usr/lib/systemd/system/wildfly.service*" using the following command.

```
sudo nano /usr/lib/systemd/system/wildfly.service
```

Add the environment variable "**$WILDFLY_CONSOLE_BIND**" as an additional command option on the "*ExecStart*" as the following.

```
ExecStart=/opt/wildfly/bin/launch.sh $WILDFLY_MODE $WILDFLY_CONFIG $WILDFLY_BIND
$WILDFLY_CONSOLE_BIND
```

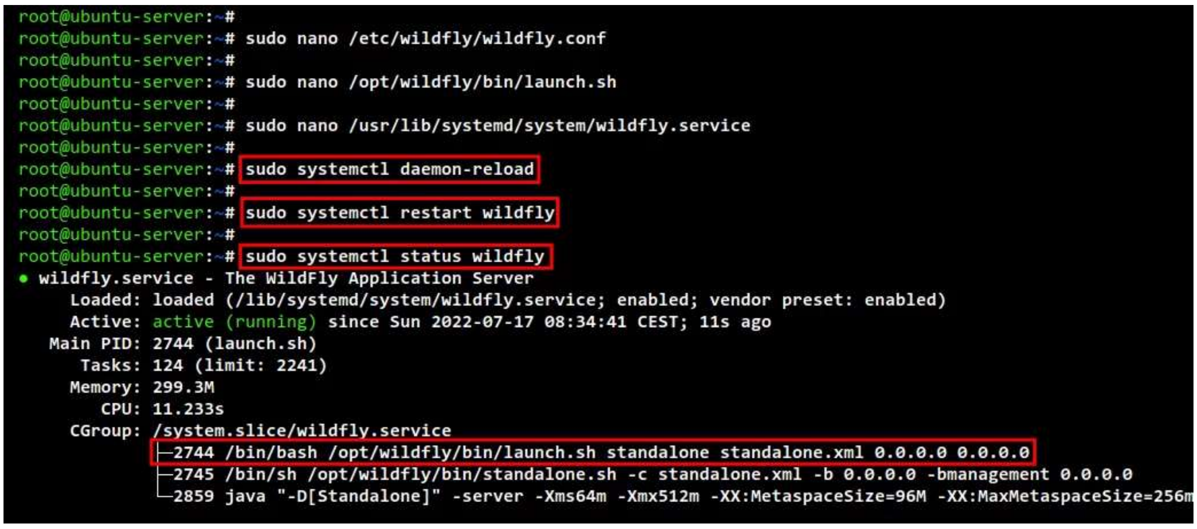
Save and close the file when you are done.

After editing the WildFly systemd service file, run the "*systemctl*" command below to reload the systemd manager. Then, restart the "*wildfly*" service to apply new changes and enable the administration console.

```
sudo systemctl daemon-reload
sudo systemctl restart wildfly
```

Now check and verify the "*wildfly*" service using the "*systemctl*" command below. And you should get the "*wildfly*" service is "**running**". On the start script "*launch.sh*" and "*standalone.sh*" you will see additional options for enabling the Administration console.

```
sudo systemctl status wildfly
```



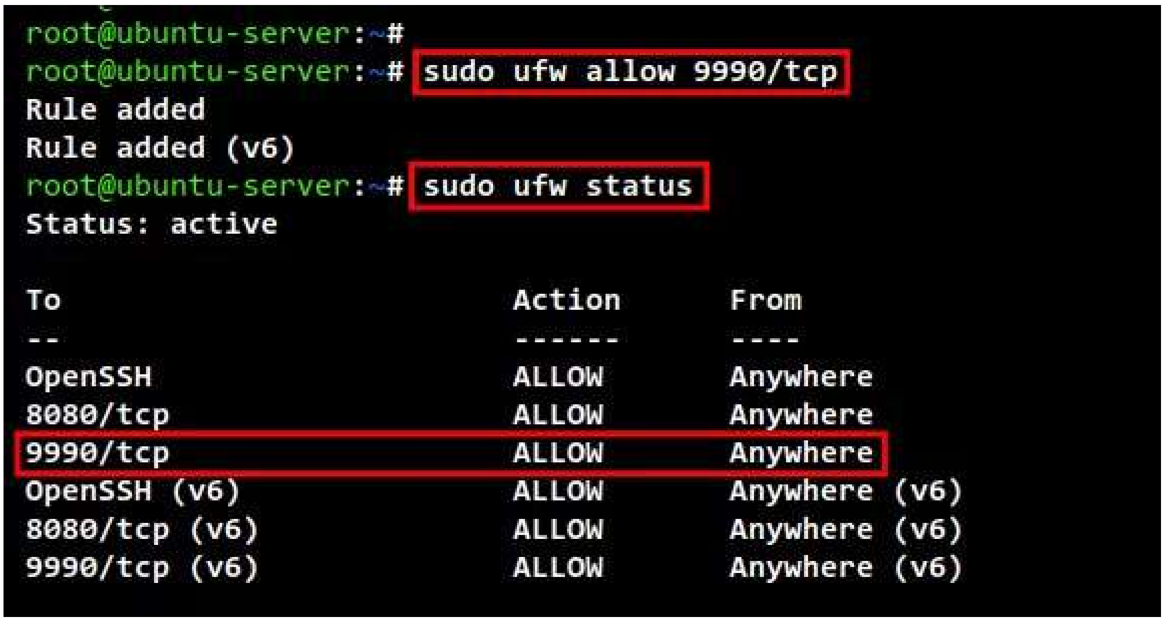
```
root@ubuntu-server:~#
root@ubuntu-server:~# sudo nano /etc/wildfly/wildfly.conf
root@ubuntu-server:~#
root@ubuntu-server:~# sudo nano /opt/wildfly/bin/launch.sh
root@ubuntu-server:~#
root@ubuntu-server:~# sudo nano /usr/lib/systemd/system/wildfly.service
root@ubuntu-server:~#
root@ubuntu-server:~# sudo systemctl daemon-reload
root@ubuntu-server:~#
root@ubuntu-server:~# sudo systemctl restart wildfly
root@ubuntu-server:~#
root@ubuntu-server:~# sudo systemctl status wildfly
● wildfly.service - The WildFly Application Server
     Loaded: loaded (/lib/systemd/system/wildfly.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-07-17 08:34:41 CEST; 11s ago
   Main PID: 2744 (launch.sh)
      Tasks: 124 (limit: 2241)
     Memory: 299.3M
        CPU: 11.233s
     CGroup: /system.slice/wildfly.service
             ├─2744 /bin/bash /opt/wildfly/bin/launch.sh standalone standalone.xml 0.0.0.0 0.0.0.0
             ├─2745 /bin/sh /opt/wildfly/bin/standalone.sh -c standalone.xml -b 0.0.0.0 -bmanagement 0.0.0.0
             └─2859 java "-D[Standalone]" -server -Xms64m -Xmx512m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m
```

The default WildFly administration is running on the TCP port "*9990*". To enable access to it, you will need to add the UFW firewall rule.

Run the "*ufw*" command below to add the UFW firewall rule for opening the WildFly administration console port "*9990*". Then, check and verify the UFW firewall status.

```
sudo ufw allow 9990/tcp
sudo ufw status
```

You will see an output similar to the following screenshot.

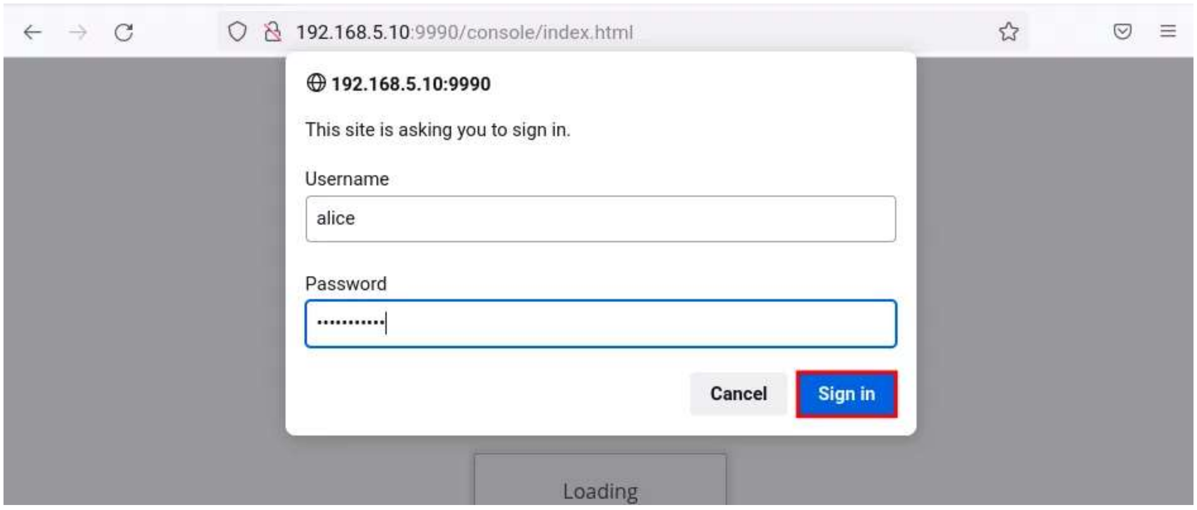


```
root@ubuntu-server:~#
root@ubuntu-server:~# sudo ufw allow 9990/tcp
Rule added
Rule added (v6)
root@ubuntu-server:~# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
8080/tcp                   ALLOW       Anywhere
9990/tcp                   ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
8080/tcp (v6)              ALLOW       Anywhere (v6)
9990/tcp (v6)              ALLOW       Anywhere (v6)
```

Lastly, back to your web browser and visit the server IP address followed by the port "*9990*" (i.e: http://192.168.5.10:9990/). And you will be prompted for the WildFly admin user and password.

Input your WildFly admin user and password, then click the "**Sign In**" button.



192.168.5.10:9990/console/index.html

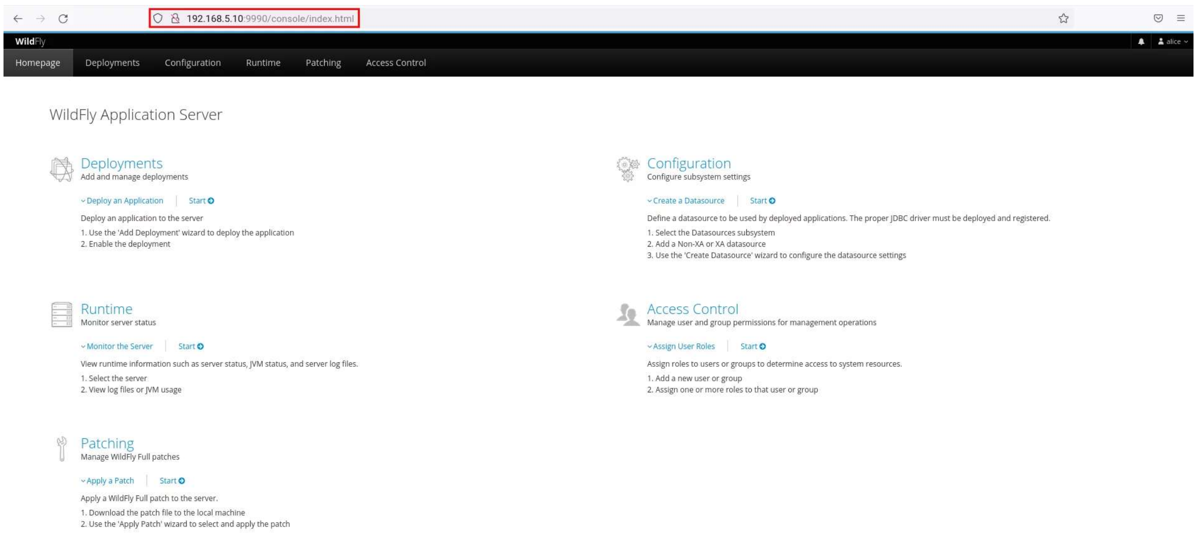
192.168.5.10:9990

This site is asking you to sign in.

Username: alice

Password: ••••••••••••

If your user and password are correct, you will see the WildFly Administration Console page below.



192.168.5.10:9990/console/index.html

WildFly Application Server

Deployments
Add and manage deployments
Deploy an Application | Start
Deploy an application to the server
1. Use the 'Add Deployment' wizard to deploy the application
2. Enable the deployment

Configuration
Configure subsystem settings
Create a Datasource | Start
Define a datasource to be used by deployed applications. The proper JDBC driver must be deployed and registered.
1. Select the Datasources subsystem
2. Add a Non-XA or XA datasource
3. Use the 'Create Datasource' wizard to configure the datasource settings

Runtime
Monitor server status
Monitor the Server | Start
View runtime information such as server status, JVM status, and server log files.
1. Select the server
2. View log files or JVM usage

Access Control
Manage user and group permissions for management operations
Assign User Roles | Start
Assign roles to users or groups to determine access to system resources.
1. Add a new user or group
2. Assign one or more roles to that user or group

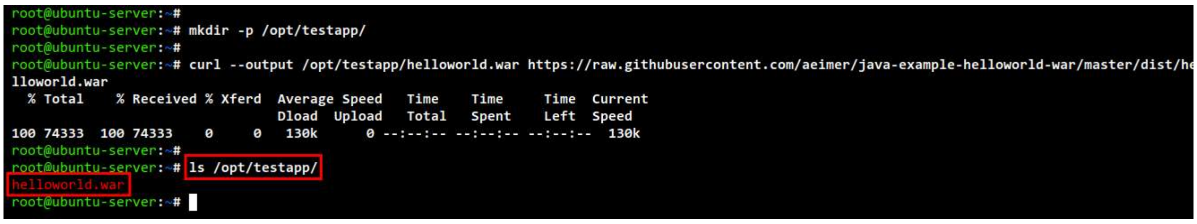
Patching
Manage WildFly Full patches
Apply a Patch | Start
Apply a WildFly Full patch to the server.
1. Download the patch file to the local machine
2. Use the 'Apply Patch' wizard to select and apply the patch

Deploying Application to WildFly

Now you will deploy the "Hello World" example application to WildFly. You can deploy an application from the WildFly administration console or using the command-line interface mode.

Create a new directory *"/opt/testapp"* and download the example application "*helloworld.war*" using the "*curl*" command below.

```
mkdir -p /opt/testapp/
curl --output /opt/testapp/helloworld.war
https://raw.githubusercontent.com/aeimer/java-example-helloworld-
war/master/dist/helloworld.war
```



```
root@ubuntu-server:~#
root@ubuntu-server:~# mkdir -p /opt/testapp/
root@ubuntu-server:~#
root@ubuntu-server:~# curl --output /opt/testapp/helloworld.war https://raw.githubusercontent.com/aeimer/java-example-helloworld-war/master/dist/he
lloworld.war
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 74333  100 74333    0     0   130k      0 --:--:-- --:--:-- --:--:--  130k
root@ubuntu-server:~#
root@ubuntu-server:~# ls /opt/testapp/
helloworld.war
root@ubuntu-server:~#
```

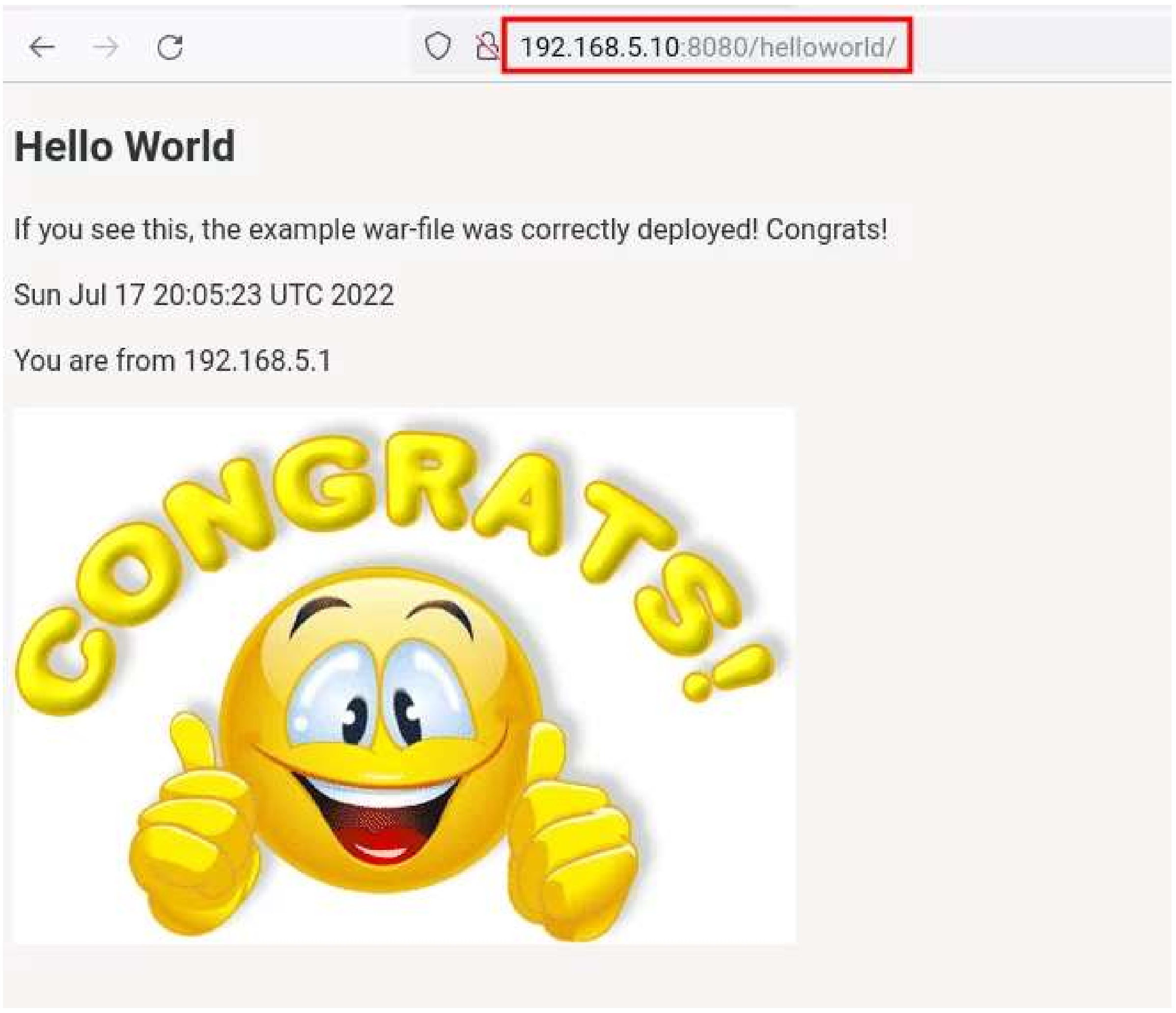
Next, run the script "*jboss-cli.sh*" to connect to your WildFly administration from the command line.

```
sudo -u wildfly /opt/wildfly/bin/jboss-cli.sh --connect
```

After you connected to the WildFly shell, run the following command to deploy the example application "*/opt/testapp/helloworld.wa*r" to WildFly.

```
deploy /opt/testapp/helloworld.war
```

Lastly, back to your web browser and visit your server IP address with the following path "*/helloworld*" (i.e: http://192.168.5.10:8080/helloworld/). And you should get the example application "*helloworld.war*".



Hello World

If you see this, the example war-file was correctly deployed! Congrats!

Sun Jul 17 20:05:23 UTC 2022

You are from 192.168.5.1

Conclusion

Congratulation! You have now successfully installed the WildFly on Ubuntu 22.04 server using the manual installation from Binary Package. You have also created the administrator user for WildFly and enabled the web-based Administration Console for WildFly. In the end, you have also learned how to deploy an application to WildFly from the command-line interface.