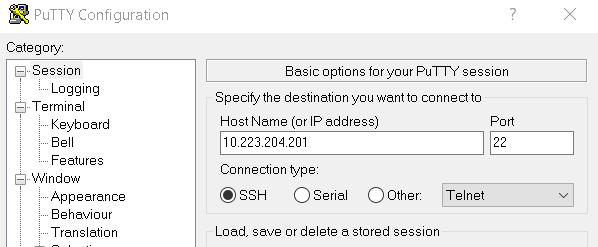
**Parking senzor: P02-NI**

IP adresa: 10.223.204.201

Port: 7800



Slika 1. otvaranje sesije kroz PuTTy alat

Analiza mrežnog prometa provedene je pomoću:

*sudo tcpdump -i ens192 udp -v port 7800*

# Stvaranje komunikacije između socket-a i servera

Komunikacija sa IoT uređajem postignuta preko python socket paketa.

**(Skripta: ParkingSensor.py)**

<https://wiki.python.org/moin/UdpCommunication>

IP adresu i port servera pohranjene su u varijable localIP i localPort. Uspostavljena je veza između PC-a i senzora preko socket-a kojeg smo nazvali UDPServerSocket.

*UDPServerSocket.bind( localIP, localPort )*

Nakon uspješnog povezivanja sa serverom skripta izbacuje poruku da je uspostavljena veza sa serverom te IP adresu i port servera.

Senzor se okine prolaskom metalne ploče preko njega te šalje UDP paket na naš PC.

Nakon primljene poruke na PC šaljemo senzoru povratnu informaciju senzoru ACK u binarnom obliku koji je zadan formatom zadanog socket interfaca.

Od senzora nakon toga primamo informacije vezane o stanju senzora. Dekodiramo poruku te iz nje pohranjujemo korisne informacije u zasebne varijable.

# Stvaranje baze podataka:

## Instalacija PostgreSQL

*sudo yum install @postgresql //skidanje i instalacija paketa*

*sudo postgresql-setup –initdb //inicijalizacija database cluster-a*

*sudo passwd postgres //Postavljanje lozinke*

*sudo systemctl start postgresql //pokretanje PostgreSQL-a*

*sudo systemctl enable postgresql //Automatsko pokretanje na boot-u*

*sudo -i -u postgres psql //Spajanje na bazu podataka kao user postgres*

*\c - komanda za selektiranje baze podataka*

*\q - izlaz is psql-a*

Dodatne informacije:

<https://www.cyberciti.biz/faq/install-and-setup-postgresql-on-rhel-8/>

Bazu podataka stvorena je u PostgreSQL-u

*CREATE TABLE Car (*

*Command varchar(2),*

*RFQualityCode varchar(2),*

*IMEIID varchar(15),*

*DataId varchar(4),*

*CarStatusValue varchar(2),*

*SensorValue varchar(4),*

*TemperatureValue varchar(4),*

*RetryCount varchar(2),*

*TransitSequenceNumber varchar(2),*

*date varchar(255)*

*);*

## Povezivanje programa s bazom podataka

Spajanje s bazom podataka ostvareno je pomoću psycopg2 modula

## Instalacija python3-psycopg2.x86\_64

*sudo dnf makecache*

*sudo dnf install python3-psycopg2.x86\_64*

Nakon instalacije modula spojili smo ga s bazom podataka pomoću funkcije *psycopg2.connect()*, kojoj smo kao argumente predali ime baze podataka te korisničko ime i lozinku pripadnog korisnika.

Nakon početnog spajanja s bazom uspostavili smo “cursor” koji će raditi sve konkretne promjene na bazi, to smo postigli funkcijom *connection.cursor()*.

Kako bi poslali pripadne podatke u bazu, moramo koristiti funkciju *cursor.execute()* kojoj kao argumente predajemo SQL naredbu koja se mora izvesti i varijable koje ćemo ubacivati u bazu, u našem programu to izgleda ovako :

*cursor.execute(“””INSERT INTO car (command, rfqualitycode, imeiid, dataid, carstatusvalue, sensorvalue, temperaturevalue, retrycount, transitsequencenumber,date)*

*VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s,%s)""",*

*(command, rf, device\_id, report\_id, car\_status, sensor\_value, temperature, retry\_count, transit\_seq\_no,dt\_string,))*.

Nakon svih promjena na bazi podataka moramo napisati *connection.commit()* kako bi se te promjene primijenile na bazu.

# Grafana

## Instalacija Grafane:

sudo yum install grafana //instalacija paketa

sudo systemctl daemon-reload

sudo systemctl start grafana-server

sudo systemctl enable grafana-server

## Expose Grafane na lokalnom računalu:

Grafanu s virtualnog stroja smo povezali s lokalnim uređajem tako što smo promijenili datoteku “defaults.ini” u direktoriju */usr/share/grafana/conf*

U datoteku smo postavili varijablu:

* http\_addr na 0.0.0.0
* *http\_port* na 3000

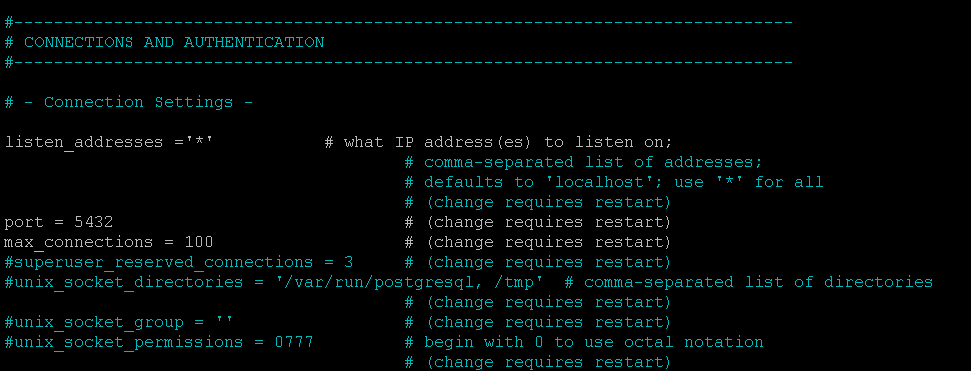
Direktorij smo pronašli pomoću bash naredbe find

*find / -type d –name ‘grafana’*

## Expose PostgreSQL-a na lokalnom računalu:

Kako bi povezali PostgreSQL koji je na virtualnom stroju s lokalnim računalom, mijenjali smo datoteku postgresql.conf u direktoriju /var/lib/pgsql/data/

Varijablu *listen\_addresses* postavili smo na *\** kako bi vratila sve adrese, a *port* je postavljen na *5432*



Slika 2. prikaz postgresql.conf

Restartat postgresql service !!

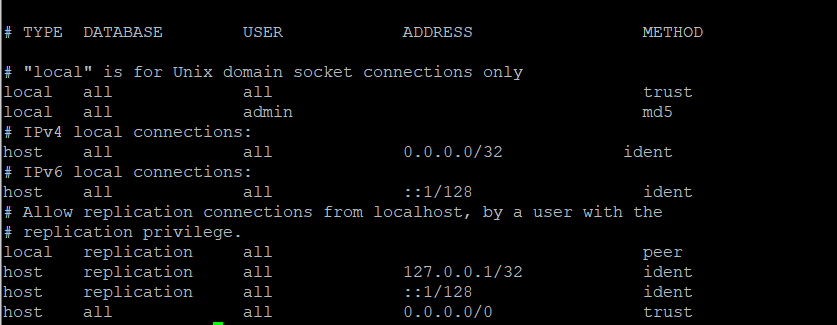
## Komunikacija baze podataka s programom i lokalnim računalom

Kod povezivanja baze podataka s programom i lokalnim računalom, imali smo problem zbog postavki vatrozida s bazom podataka.

*Morali smo promijeniti postavke u datoteci pg\_hba.conf u direktoriju /var/lib/pgsql/data/*

Direktorij u kojem je datoteka pronašli smo pomoću bash naredbe find:

find / -type f –name ‘pg\_hba.conf’



Slika 3. prikaz pg\_hba.conf

Pomoću trust metode u prvom unosu, uspjeli smo propustiti pristup bazi podataka s vanjskog PC te pomoću postavljanja adrese na 0.0.0.0

Restartat postgresql service !!

# Postavke vatrozida

## Izlistavanje exposeanih portova:

*sudo firewall-cmd --list-ports*

ili

*netstat -lntu*

## Exposeanje port-a lokalnom računalu za Grafanu

*sudo firewall-cmd --add-port=3000/tcp --permanent*

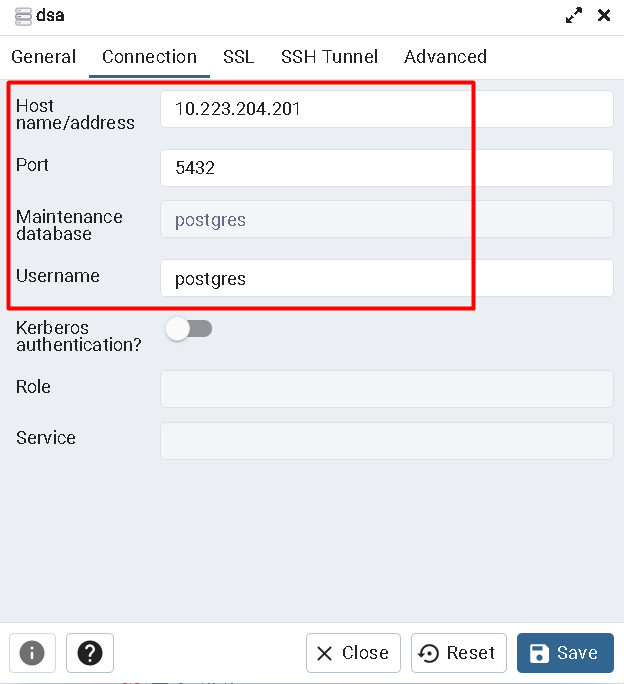
## Exposeanje port-a lokalnom računalu za PostgreSQL

*sudo firewall-cmd --add-port=5432/tcp --permanent*

## Reload firewall-a

sudo firewall-cmd –-reload

# Spajanje na bazu podataka preko lokalnog računala:



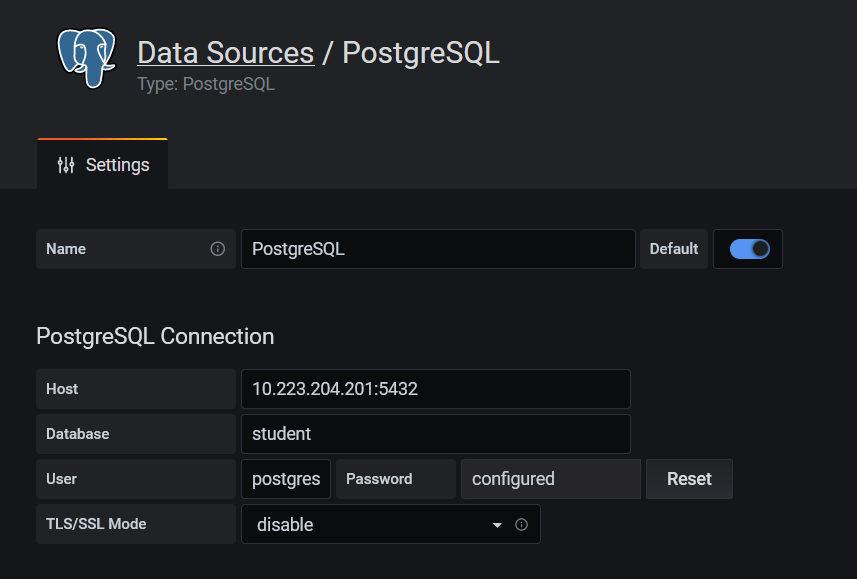
Slika 4. postavke za spajanje na bazu podataka preko PgAdmin alata

Dodatne informacije i postavke:

[**https://docs.bitnami.com/aws/apps/noalyss/administration/configure-pgadmin/**](https://docs.bitnami.com/aws/apps/noalyss/administration/configure-pgadmin/)

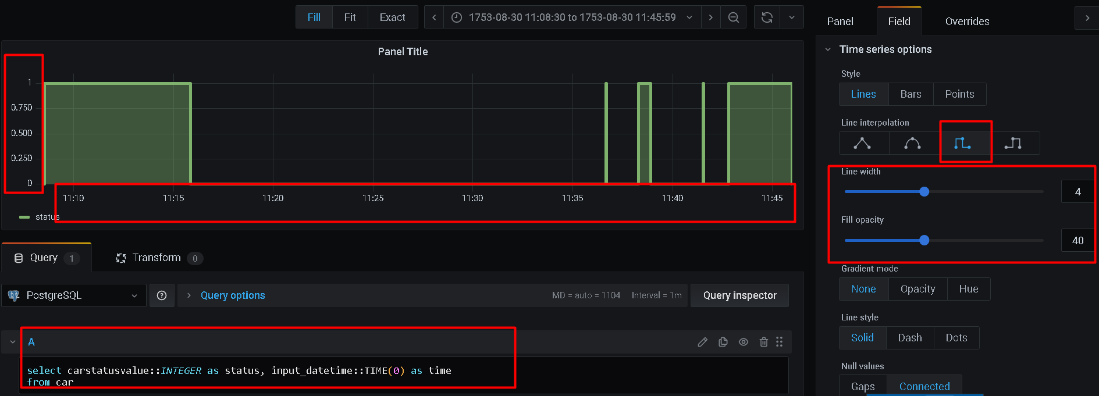
[**https://www.groovypost.com/howto/open-ports-in-linux/**](https://www.groovypost.com/howto/open-ports-in-linux/)

# Konfiguracija Data Source-a Grafane:



Slika 5. Unos postavki za spajanje PostgreSQL-a i Grafane

Spajanje na grafanu preko <http://10.223.204.201:3000/> na lokalnom računalu



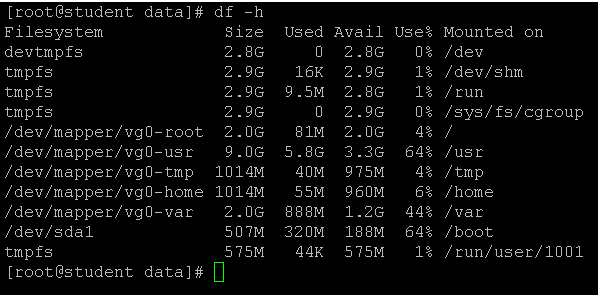
Slika 6. postavljanje prikaza podataka na grafani

# Instalacija grafičkog sučelja na RHEL8

## Proširivanje diskova

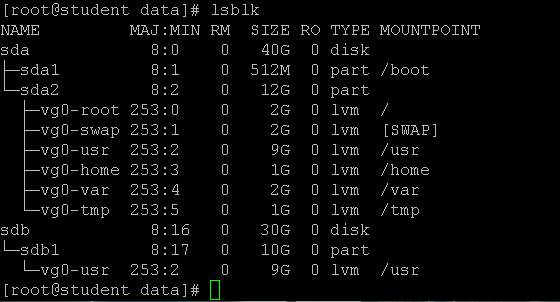
*/dev/mapper/vg0-usr mora imati barem 6GB*

*df -h*



Slika 7.

*lsblk*



Slika 8.

Stvaranje nove particije preko fdisk

*fdisk /dev/sdb*

*n //add partition*

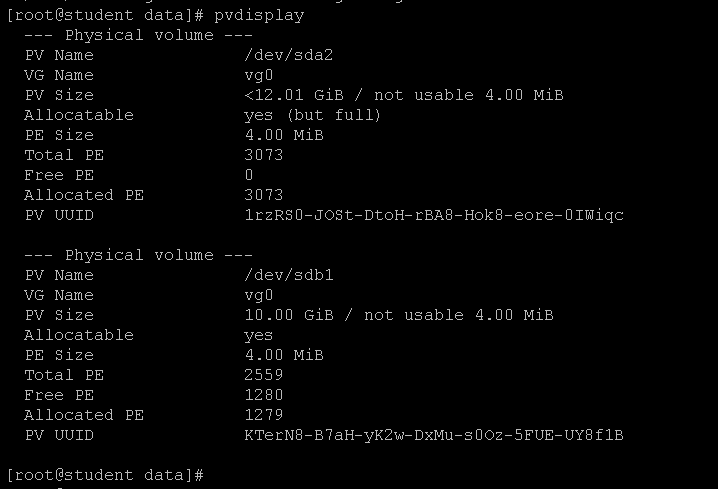
*+9GB*

*w //spremi promjene*

Prikaz physical volumse poslije stvaranje particije:

Pvcreate /dev/sdb1

*pvdisplay*

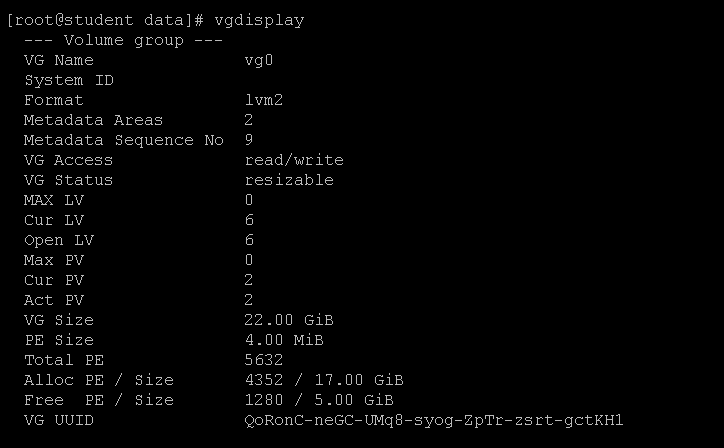


Slika 9.

Assign physical volume to volume group:

*vgdisplay*

Vgextend vg0 /dev/sdb1

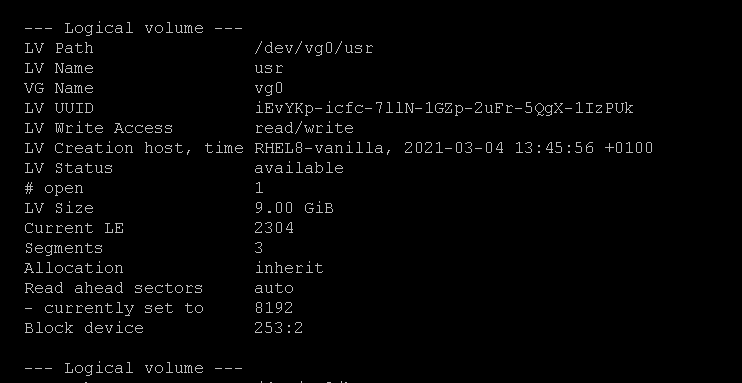


Slika 10.

Extend logical volume

*lvdisplay*

Lvextend –L+5G /dev/vg0/usr



Slika 11.

Još je potrebno commitat change na fizičke diskove:

*xfs\_growfs /dev/mapper/vg0\_usr*

Ili

*resize2fs /dev/mapper/vg0\_usr*

Ovisno o načinu kodiranja.

Skidanje paketa grafičkog sučelja:

*dnf groupinstall "Server with GUI"*

*systemctl set-default graphical*

*reboot //Nakon reboota pokrenuto je grafičko sučelje*

Dodatci

## ParkingSensor.py

*import socket*

*import psycopg2*

*from datetime import datetime*

*import time*

*localIP = "10.223.204.201"*

*localPort = 7800*

*bufferSize = 1024*

*connection = psycopg2.connect("dbname=student user=postgres password=123")*

*cursor = connection.cursor()*

*UDPServerSocket = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_DGRAM)*

*UDPServerSocket.bind((localIP, localPort))*

*print("Uspostavljena veza sa serverom.")*

*print("IP adresa: ", localIP)*

*print("Port: ", localPort)*

*while True:*

*message, addr = UDPServerSocket.recvfrom((bufferSize))*

*print("Primljena poruka: {0}".format(message))*

*UDPServerSocket.sendto(b"ACK", addr)*

*message = message.decode('ascii')*

*command = message[0:2]*

*print(command)*

*if (command == "02"):*

*ts = time.time()*

*timestamp = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y %H:%M:%S')*

*rf = message[2:4]*

*device\_id = message[4:19]*

*report\_id = message[19:23]*

*car\_status = message[23:25]*

*sensor\_value = message[25:29]*

*temperature = message[29:33]*

*retry\_count = message[33:35]*

*transit\_seq\_no = message[35:37]*

*cursor.execute("""INSERT INTO car (command, rfqualitycode, imeiid, dataid, carstatusvalue, sensorvalue, temperaturevalue, retrycount, transitsequencenumber,input\_datetime)*

*VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s,%s)""", (command, rf, device\_id, report\_id, car\_status, sensor\_value, temperature, retry\_count, transit\_seq\_no,timestamp,))*

*if(car\_status == "00"):*

*print("Mjesto je slobodno.")*

*else:*

*print("Mjesto je zauzeto!")*

*connection.commit()*

*else:*

*print("Nije primljena tražena poruka!")*