**Postgresql in Heroku**

To add a database in heroku, we need to be in the resources. From there we add the following Heroku Postgres add-on🡺



That’s it our database is set to be used! Now we can seed our database with existing data. So for this we need to export .sql file from our database by using the following command 🡺

pg\_dump -U user -d db\_name -h 127.0.0.1 > dump.sql

We can exclude the host part if the db is in local. Here this command created dump.sql automagically.

**Note that** to use psql commands we need to be in the following directory D:\PostgreSQL\14\bin

Now since, we have .sql file, we can seed our database which is located on Heroku. To do this we use the following commands🡺

heroku pg:psql –app app-name < .sql-file-path

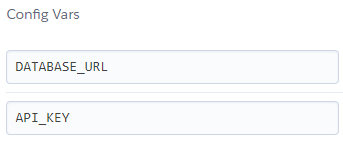
heroku pg:psql –app runecube < C:\\Users\\User\\OneDrive\\”Рабочий стол”\\restore.sql

Keep in mind that when we use packet manager console of nuget we can add migrations and update our database.

Our connection string:

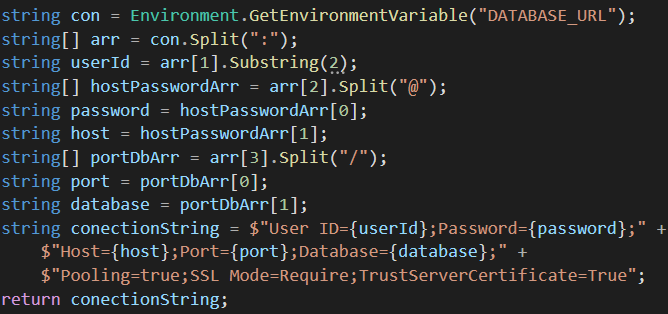
User id=username; Password=password; Host=hostAddress; Port=port; Database=db; Pooling=true;SSL Mode=Require; TrustServerCertificate =True

Note that we need to take our secret keys (api keys, connection strings) as enviromental variables from Heroku. We can also specify our config vars 🡺



From DATABASE\_UR variable we get the following string 🡺 postgres://username:password@host:port/database

As you can see it is different from what we have to send. So we filter it and generate our connection string from it



We also need to add some extra info to our connection string such as Pooling, SSL Mode and etc.

We can also connect to our heroku postgres using heroku cli which is provided for us in the database credentials section 🡺

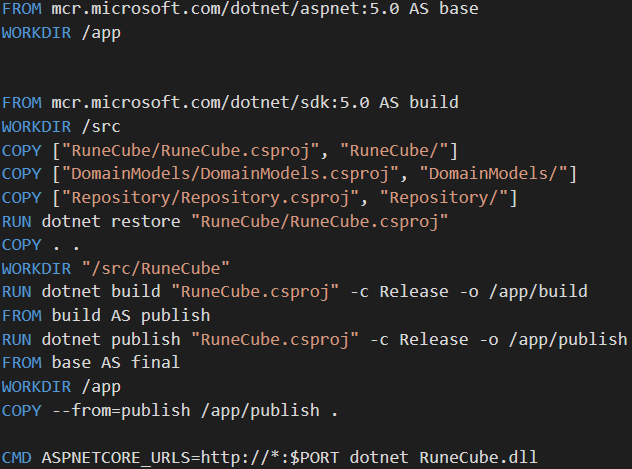


**Dockerfile for Heroku**

Since heroku itself configures our port we cannot expose it manually. Instead we need to add the following cmd at the end of our dockerfile which basically takes the port from the environmental variable which is set by heroku.

CMD ASPNETCORE\_URLS=http://\*:$PORT dotnet RuneCube.dll

In this example, we are using docker-compose.



**Publishing app to Heroku**

Once we build our image we need to tag our image as following 🡺

registry.heroku.com/app-name/web After this we push this image to docker hub from which we release to heroku🡺

docker build –t my-image registry.heroku.com/runecube/web

docker push registry.heroku.com/runecube/web

heroku container:release web --app runecube