**Készítette**: Dr. Subecz Zoltán

Neumann János Egyetem, GAMF Informatikai és Műszaki Kar, Informatika tanszék

[subecz.zoltan@gamf.uni-neumann.hu](mailto:subecz.zoltan@gamf.uni-neumann.hu)

Java-Spring-Megjelenítés az Interneten-Microsoft Azure-al

[Bevezetés 2](#_Toc113481769)

[1. feladat Egyszerű alkalmazás készítése - Java kód feltöltése és futtatása az Azure felhőben 3](#_Toc113481770)

[Alkalmazás előkészítése - Feltöltése a GitHub-ra 3](#_Toc113481771)

[App service létrehozás 3](#_Toc113481772)

[Feltöltés az Azure-ra: letöltés GIT-ről, Build, Deploy 4](#_Toc113481773)

[Azure PostgreSQL adatbázis készítése + használata grafikus programmal 8](#_Toc113481774)

[Azure adatbázis elérése grafikus adatbáziskezelő programmal saját gépről – pgAdmin, Azure Data Studio 10](#_Toc113481775)

[2. feladat Java JDBC-vel elérni saját gépről Azure PostgreSQL adatbázist 12](#_Toc113481776)

[Java project készítése – Tábla feltöltése + CRUD alkalamzás készítése az Azure adatbázisba helyi gépről Java-ban – JDBC-vel 12](#_Toc113481777)

[Kapcsolat kialakítása az adatbázissal Java JDBC-vel helyi gépről 13](#_Toc113481778)

[CRUD alkalmazás készítése (Create, Read, Update, Delete) 15](#_Toc113481779)

[3. feladat - Alkalmazás adatbázissal – Java kód feltöltése és futtatása az Azure felhőben 19](#_Toc113481780)

[Alkalmazás elkészítése és futtatása helyi gépen – az Azure adatbázis elérésével 19](#_Toc113481781)

[Alkalmazás feltöltése a GitHub-ra 20](#_Toc113481782)

[App service létrehozás 20](#_Toc113481783)

[Feltöltés az Azure-ra: letöltés GIT-ről, Build, Deploy 21](#_Toc113481784)

# Bevezetés

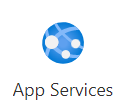
"Microsoft Azure for Students": ha valaki rendelkezik Neumann-os (Microsoft 365 – Office 365) e-mail címmel, azok tudnak regisztrálni erre az oldalra és ingyen igénybe tudják venni ezt a szolgáltatást.

<https://azure.microsoft.com/en-us/free/students/>

Nem kell bankszámlaszámot megadni,

100 USD kreditet ad 1 évig és sok ingyenes szolgáltatás van rajta.

**Nyelvbeállítás**: jobb felső sarok:  => View account

 =>  => 

<https://azure.microsoft.com/hu-hu/services/app-service/#overview>

Az **Azure App Service** lehetővé teszi az alkalmazások gyorsabb létrehozását egy egyedülálló felhőszolgáltatással, amellyel gyorsan és egyszerűen készíthet nagyvállalati használatra kész webes és mobilalkalmazásokat bármilyen platformhoz vagy eszközhöz, és méretezhető, megbízható felhőalapú infrastruktúrán helyezheti azokat üzembe.

<https://docs.microsoft.com/en-us/azure/developer/java/>

# 1. feladat Egyszerű alkalmazás készítése - Java kód feltöltése és futtatása az Azure felhőben

<https://docs.microsoft.com/en-us/azure/app-service/quickstart-java?pivots=platform-windows-development-environment-maven&tabs=javase>

alapján, kissé módosítva

## Alkalmazás előkészítése - Feltöltése a GitHub-ra

A gyakorlaton tanult első Spring Boot alkalmazást (készítjük el és) fogjuk felmásolni a Heroku-ra: **első Egyszerű webalkalmazás**

Készítsük el az alkalmazást, vagy használjuk fel a **Megoldasok.zip** fájlban lévő megoldást.

Ellenőrizzük le: az alkalmazás működjön a helyi gépen!

**system.properties: Java verzió megadása**

Azure-ban lehet, hogy nem kötelező (más szolgáltatónál meg kell adni)

pom.xml fájlban nézzük meg a Java verziót:

<java.version>**11**</java.version>

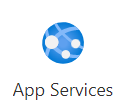
Készítsünk a Projekt mappájában egy **system.properties** fájlt ezzel a tartalommal:

java.runtime.version=11

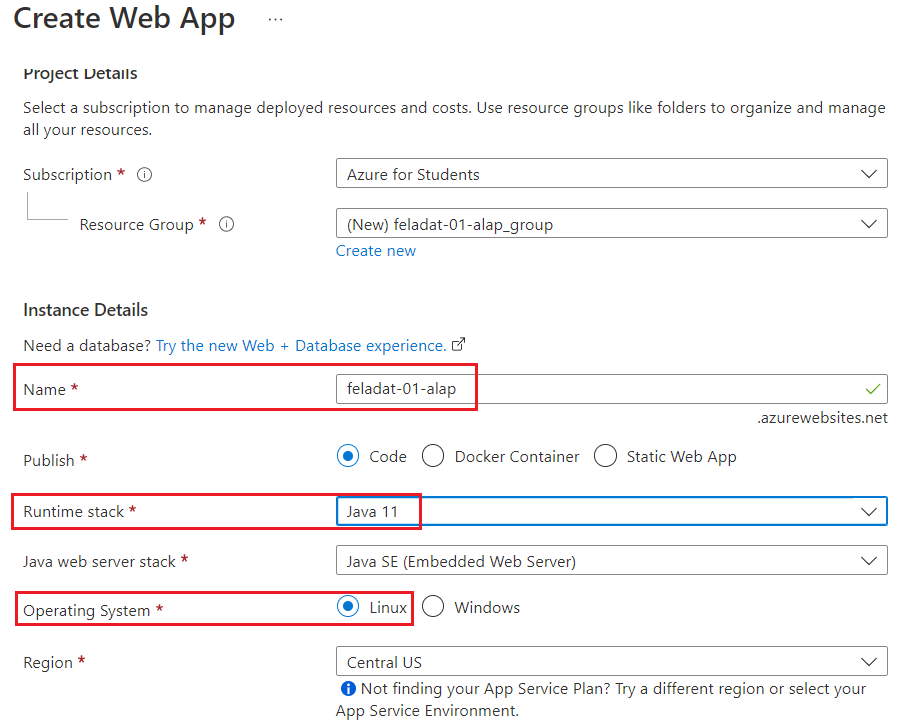
**GITHUB-GITLAB használat.docx** fájlban leírtak alapján töltsük fel a projektet a GitHub-ra

pl. <https://github.com/suzo777/JavaSpringBoot1>

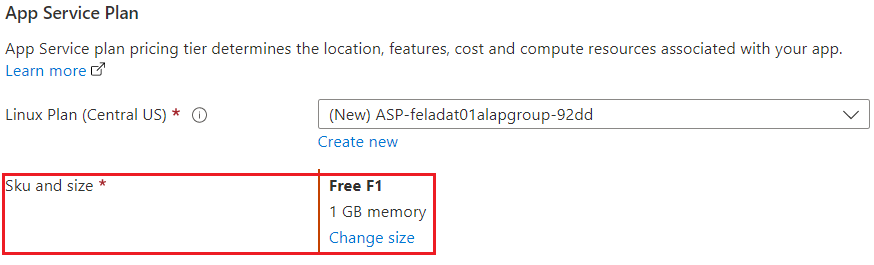
## App service létrehozás

 => 

**Name**: feladat-01-alap Egyedinek kell lenni az Interneten: <https://feladat-01-alap.azurewebsites.net>



Regions: <https://azure.microsoft.com/en-us/global-infrastructure/services/?products=all&regions=all>



**Free F1**: ez az ingyenes lehetőség

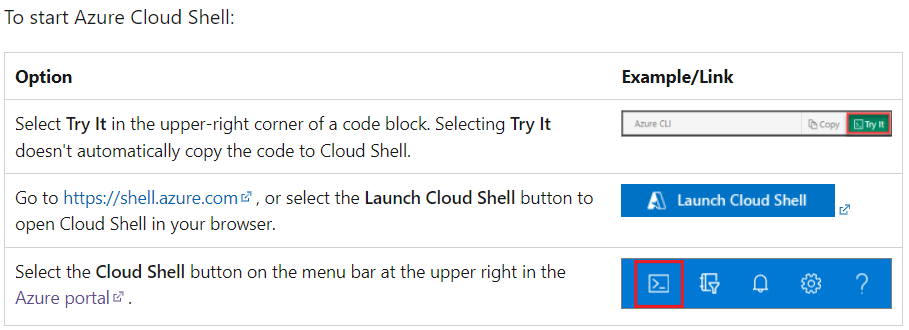


**Java 11 szerepe**

Majd látjuk a Cloud shell-ben, hogy ott jelenleg a Java 11 van telepítve, az használható. Mivel a fodításhoz és a telepítéshez azt fogjuk használni, ezért felesleges itt magasabb verziószámot beállítani.

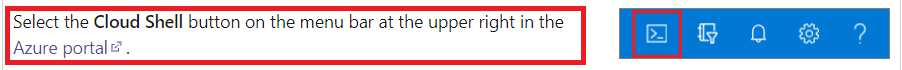
## Feltöltés az Azure-ra: letöltés GIT-ről, Build, Deploy

**1 - Use Azure Cloud Shell**

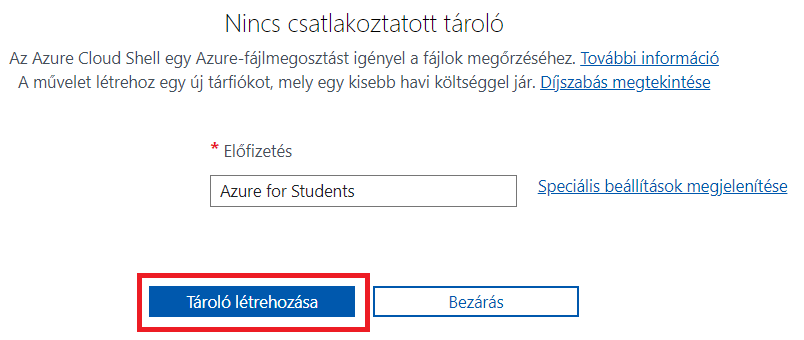


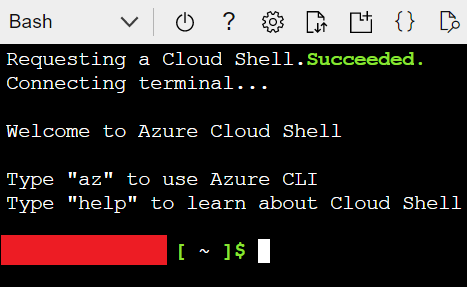
<https://shell.azure.com/>

vagy: (oldalon a felső sorban)



=> Bash





**Néhány utasítás a Cloud Shell-be-ben:**

ls: mappa tartalmának kiíratása

cd: mappaváltás

rm: fájl törlés

rm –r: mappa törlés

 Fájl feltöltés/letöltés

 Fájlkezelő, fájlszerkesztő megnyitás

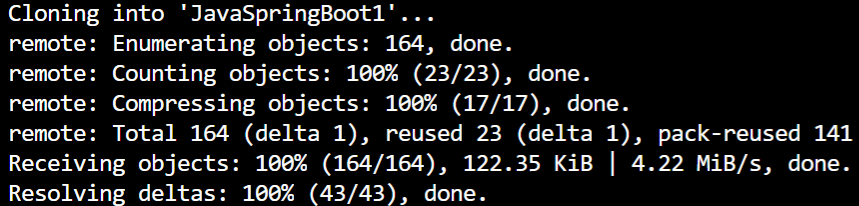
**2 - Create a Java app**

Az elkészített Java alkalmazás letöltése a GIT-ről:

Bemásolni a Cloud Shell-be:

git clone [https://github.com/\*\*\*\*\*/JavaSpringBoot1](https://github.com/*****/JavaSpringBoot1)

Ctrl+Shift+V +Enter



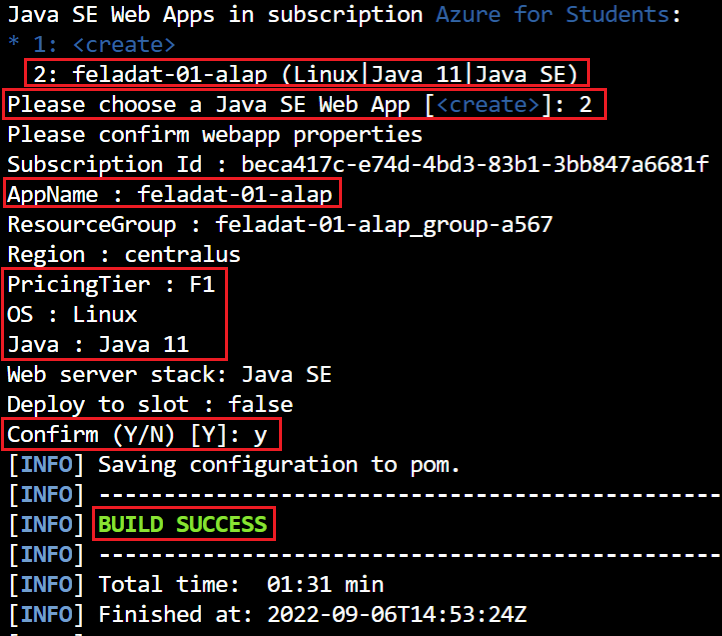
**ls**: mappát létrehozta

**mappába belépni:**

cd JavaSpringBoot1

mvn com.microsoft.azure:azure-webapp-maven-plugin:2.5.0:config

Ctrl+Shift+V +Enter



AppName : feladat-01-alap

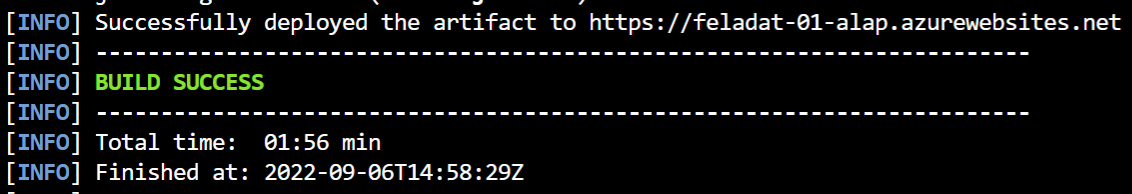
mvn package azure-webapp:deploy

Ctrl+Shift+V +Enter

Kiír közben egy figyelmeztetést:



de sikeresen feltelepítette:



**Böngészőbe:**

<https://feladat-01-alap.azurewebsites.net/>



**Nincs "/" útvonal megadva! Lásd: DemoApplication.java**

De így jó:

https://feladat-01-alap.azurewebsites.net/feladat1a



=> DemoApplication.java

@GetMapping("/feladat1a")

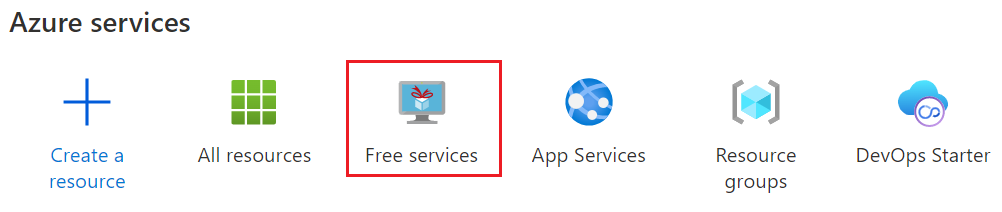
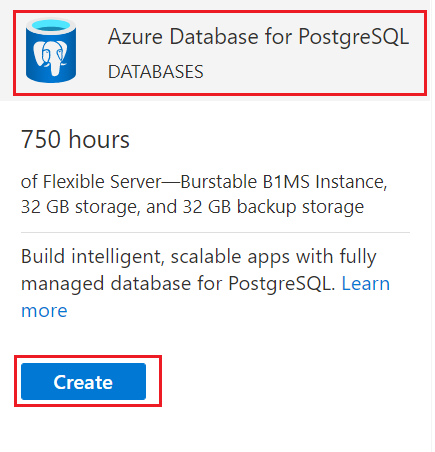
@ResponseBody

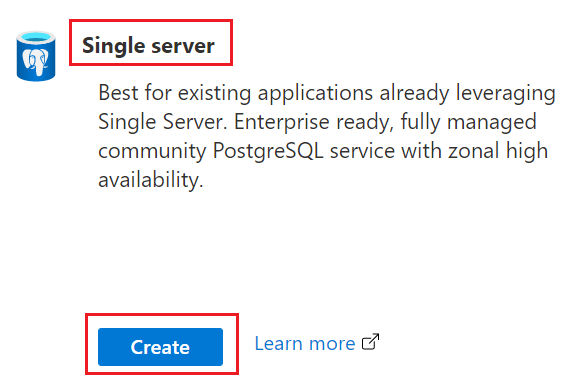
public String kiir1a() {

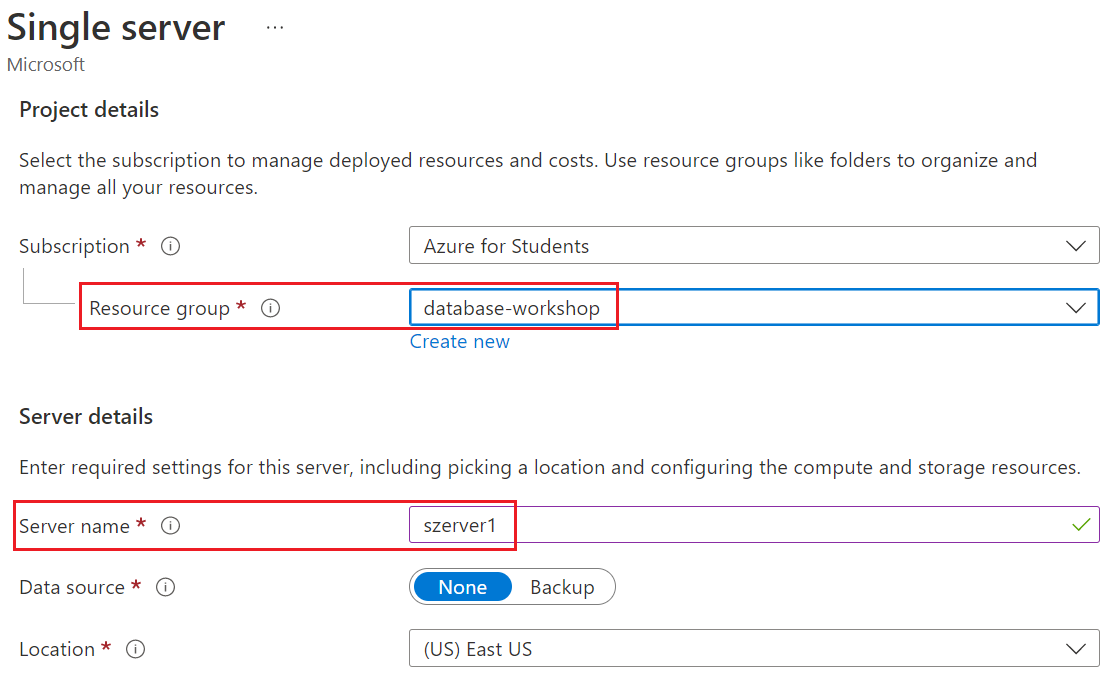
return "<h1>Hello world!</h1>";

}

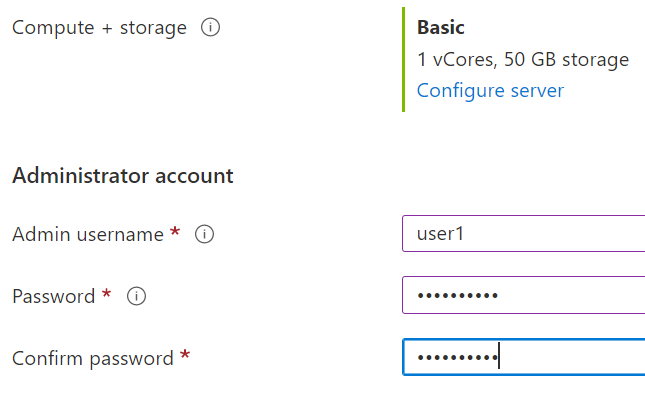
# Azure PostgreSQL adatbázis készítése + használata grafikus programmal



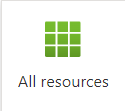


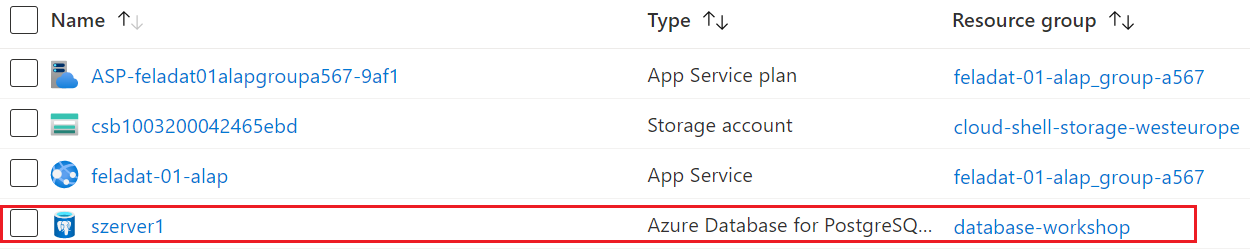
=> Configure server: a legegyszerűbbet és legolcsóbbat választani!



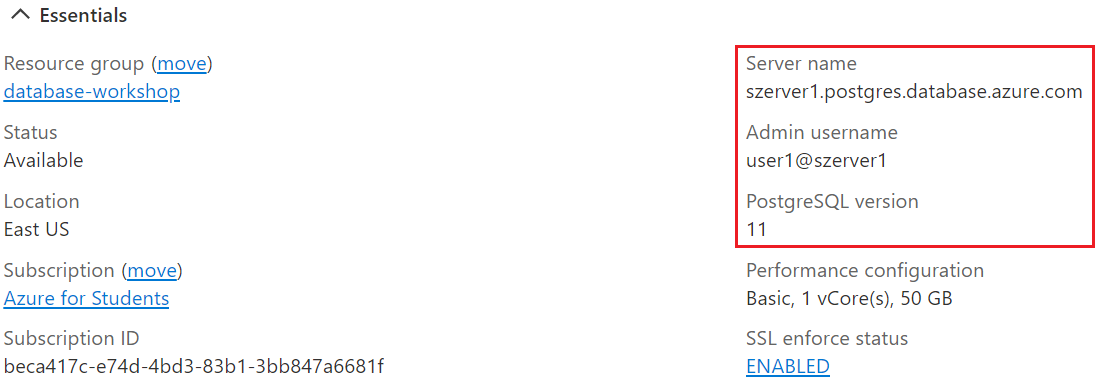
password: Proba12345



Az adatbázis megjelenik az  csoportban:

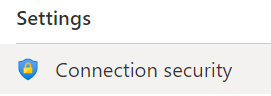


Főbb adatok leolvashatóak:



**A szerver hozzáférésének engedélyezése – Firewall rules**

Azure Database for PostgreSQL instances are secured by default. They have a firewall that doesn't allow any incoming connection. To be able to use your database, you need to add a firewall rule that will allow the local IP address to access the database server.



Meg kell adni, hogy honnan lehet elérni az adatbázist. Megadjuk, hogy belső programok elérhessék (pl. egy Java program, amit majd feltelepítünk), és megadjuk a saját gépünk IP címét. Saját gépről is el akarjuk érni:

- pl. a következő példában egy Java programmal

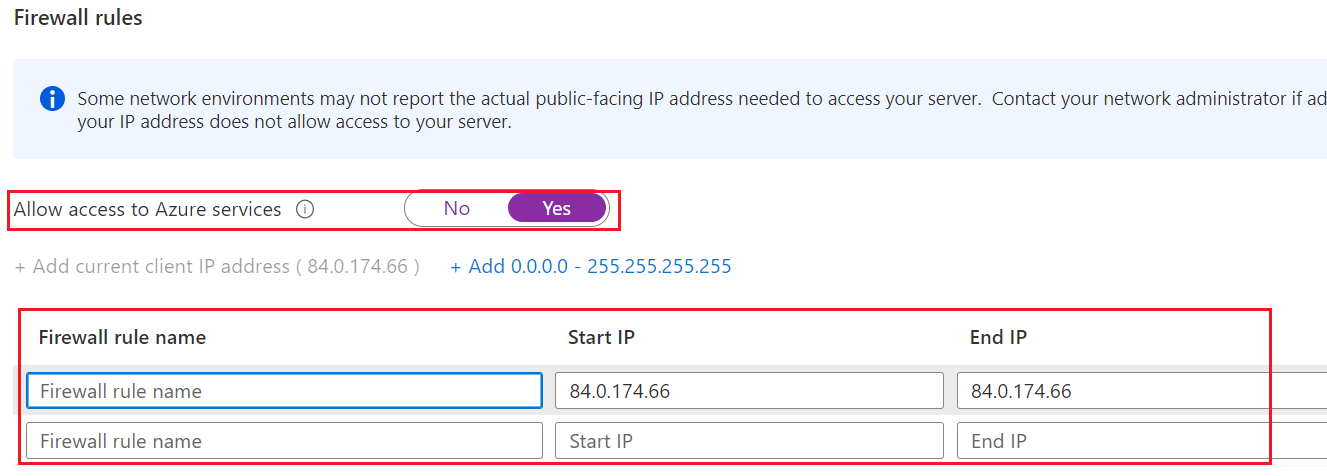
- grafikus adatbáziskezelő programmal. pl. pgAdmin vagy Azure Data Studio (lásd később)

**Saját IP cím lekérdezése:pl.:**

<http://whatismyip.akamai.com/>

<https://www.whatismyip.com/>

pl.: 84.0.174.66



**Save**

Allow access to **Azure services = Yes**: így tudjuk belső programokkal elérni az adatbázist.

## Azure adatbázis elérése grafikus adatbáziskezelő programmal saját gépről – pgAdmin, Azure Data Studio

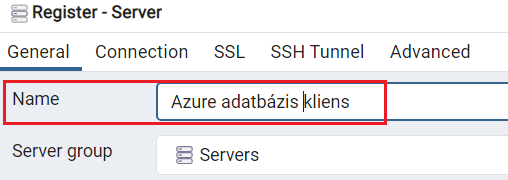
Több jó program is van erre pl. pgAdmin vagy Azure Data Studio

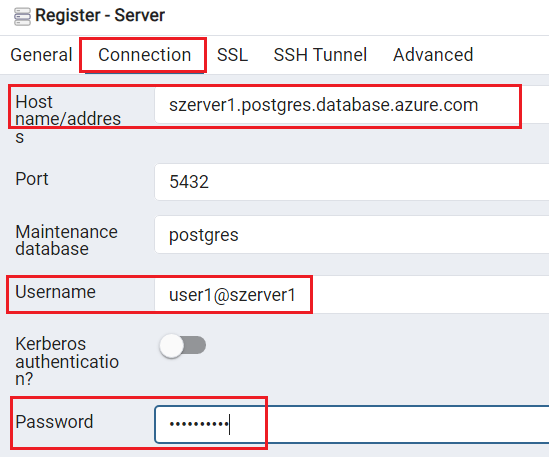
**pgAdmin – Postgresql adatbázisokhoz**

<https://www.pgadmin.org/download/>

Telepítés, indítás után:

Add new server =>





Password: Proba12345

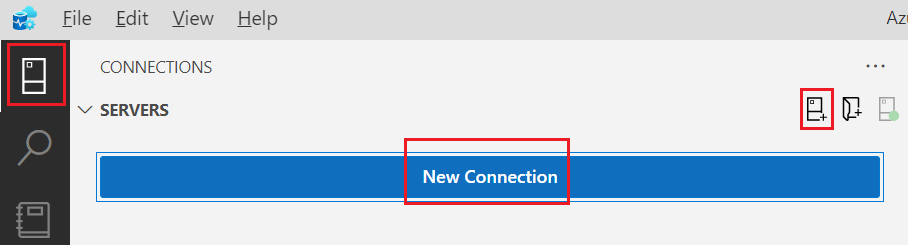
**Azure Data Studio – Azure adatbázisokhoz**

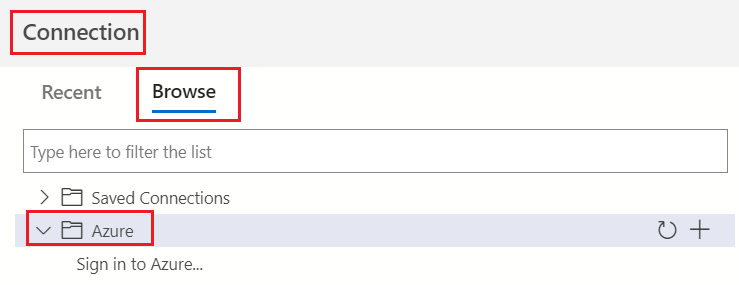
<https://azure.microsoft.com/en-us/services/developer-tools/data-studio/#overview>

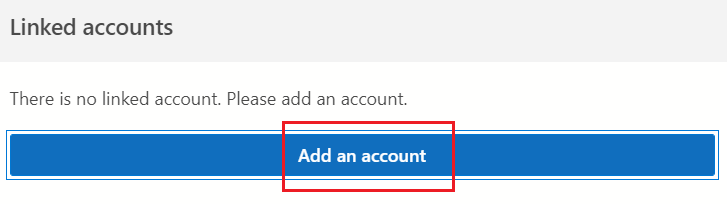
=> Download now

<https://docs.microsoft.com/en-us/sql/azure-data-studio/download-azure-data-studio?view=sql-server-ver16>

User installer letöltése, telepítése

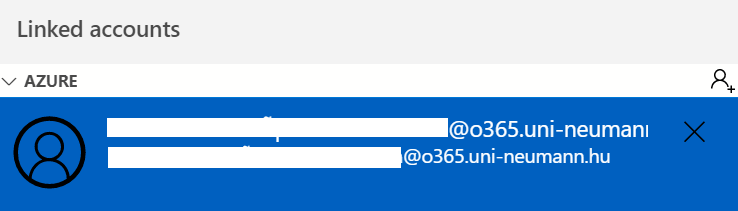




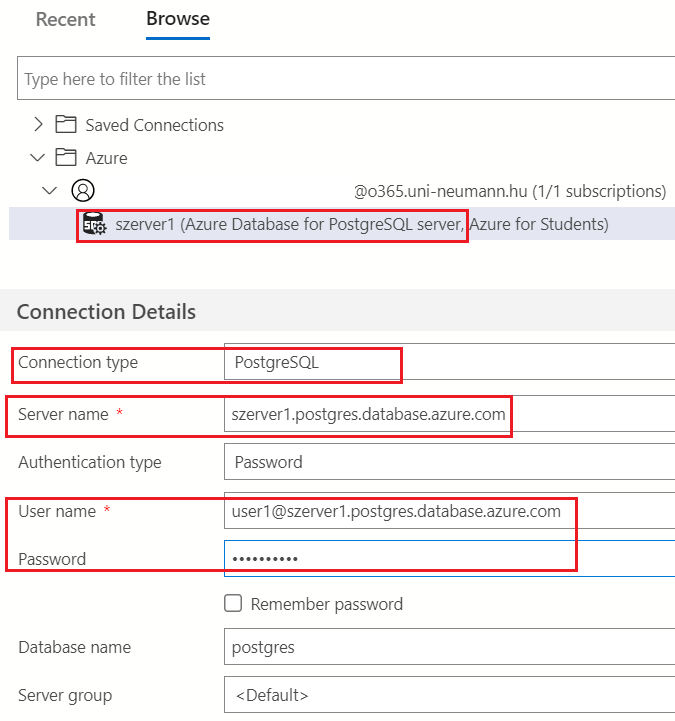


Megnyíló böngészőben be kell jelentkezni, ahogy az Azure-ba bejelentkeztünk.

Sikeres bejelentkezés után megjelenik az Azure Data Studio-ban a kapcsolat, ami már használható.



A folyamat közben feltelepíti a PostgreSQL extension-t



Bármelyiket is választjuk, **hozzunk létre egy adatbázist pl. database1 néven** pl. SQL utasítással:

Create database database1

# 2. feladat Java JDBC-vel elérni saját gépről Azure PostgreSQL adatbázist

**Use Java and JDBC with Azure Database for PostgreSQL**

<https://docs.microsoft.com/en-us/azure/postgresql/single-server/connect-java>

alapján, kissé módosítva

Ehhez nem kell létrehozni új App Service-t.

## Java project készítése – Tábla feltöltése + CRUD alkalamzás készítése az Azure adatbázisba helyi gépről Java-ban – JDBC-vel

**IntelliJ-ben (más IDE is jó) készítsünk egy új Java projektet.**

Készítsünk egy mappát a projektnek pl.: c:\Java-Azure-1

New project / Jave, Maven,

Name pl.: Azure1

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>demo</name>

<properties>

<java.version>1.8</java.version>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.2.12</version>

</dependency>

</dependencies>

</project>

<java.version>11</java.version> is jó

Ha pirossal bejelöl dolgokat: File menü / Invalidate Cache

### Kapcsolat kialakítása az adatbázissal Java JDBC-vel helyi gépről

**Prepare a configuration file to connect to Azure Database for PostgreSQL**

src/main/resources/**application.properties**

url=jdbc:postgresql://szerver1.postgres.database.azure.com:5432/database1?ssl=true&sslmode=require

user=user1@szerver1

password=Proba12345

Az szerverév, adatbázisnév és jelszó, ahogy beállítottuk.

**Connect to the database**

src\main\resources\**schema.sql**

DROP TABLE IF EXISTS todo;

CREATE TABLE todo (id SERIAL PRIMARY KEY, description VARCHAR(255), details VARCHAR(4096), done BOOLEAN);

Hozzuk létre a **com.example.demo** csomagot:

File menü / New / Package

src/main/java/com/example/demo/**DemoApplication**.java

package com.example.demo;

import java.sql.\*;

import java.util.\*;

import java.util.logging.Logger;

public class DemoApplication {

private static final Logger log;

static {

System.setProperty("java.util.logging.SimpleFormatter.format", "[%4$-7s] %5$s %n");

log =Logger.getLogger(DemoApplication.class.getName());

}

public static void main(String[] args) throws Exception {

log.info("Loading application properties");

Properties properties = new Properties(); properties.load(DemoApplication.class.getClassLoader().getResourceAsStream("application.properties"));

log.info("Connecting to the database");

Connection connection = DriverManager.getConnection(properties.getProperty("url"), properties);

log.info("Database connection test: " + connection.getCatalog());

log.info("Create database schema");

Scanner scanner = new Scanner(DemoApplication.class.getClassLoader().getResourceAsStream("schema.sql"));

Statement statement = connection.createStatement();

while (scanner.hasNextLine()) {

statement.execute(scanner.nextLine());

}

/\* Ezt egyelőre kikommentezzük. Majd később használni fogjuk:

Todo todo = new Todo(1L, "configuration", "congratulations, you have set up JDBC correctly!", true);

insertData(todo, connection);

todo = readData(connection);

todo.setDetails("congratulations, you have updated data!");

updateData(todo, connection);

deleteData(todo, connection);

\*/

log.info("Closing database connection");

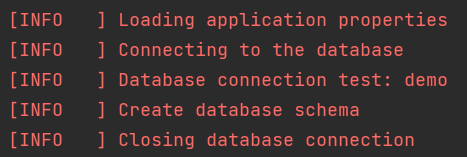
connection.close();

}

}

In this file, you can see that we commented methods to insert, read, update and delete data: we will code those methods in the rest of this article, and you will be able to uncomment them one after each other.

**Futtatás:**



**Create a domain class**

src/main/java/com/example/demo/**Todo**.java

package com.example.demo;

public class Todo {

private Long id;

private String description;

private String details;

private boolean done;

public Todo() {

}

public Todo(Long id, String description, String details, boolean done) {

this.id = id;

this.description = description;

this.details = details;

this.done = done;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public String getDetails() {

return details;

}

public void setDetails(String details) {

this.details = details;

}

public boolean isDone() {

return done;

}

public void setDone(boolean done) {

this.done = done;

}

@Override

public String toString() {

return "Todo{" +

"id=" + id +

", description='" + description + '\'' +

", details='" + details + '\'' +

", done=" + done +

'}';

}

}

### CRUD alkalmazás készítése (Create, Read, Update, Delete)

**Insert data into Azure Database for PostgreSQL**

src/main/java/com/example/demo/**DemoApplication**.java

**fájlhoz adjuk hozzá a következő metódust:**

private static void **insertData**(Todo todo, Connection connection) throws SQLException {

log.info("Insert data");

PreparedStatement insertStatement = connection

.prepareStatement("INSERT INTO todo (id, description, details, done) VALUES (?, ?, ?, ?);");

insertStatement.setLong(1, todo.getId());

insertStatement.setString(2, todo.getDescription());

insertStatement.setString(3, todo.getDetails());

insertStatement.setBoolean(4, todo.isDone());

insertStatement.executeUpdate();

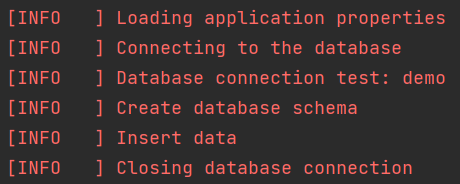
}

**Kommentezzük ki a következő utasításokat a DemoApplication osztályban:**

Todo todo = new Todo(1L, "configuration", "congratulations, you have set up JDBC correctly!", true);

insertData(todo, connection);

**Futtatás**



**Reading data from Azure Database for PostgreSQL**

src/main/java/com/example/demo/**DemoApplication**.java

**fájlhoz adjuk hozzá a következő metódust:**

private static Todo **readData**(Connection connection) throws SQLException {

log.info("Read data");

PreparedStatement readStatement = connection.prepareStatement("SELECT \* FROM todo;");

ResultSet resultSet = readStatement.executeQuery();

if (!resultSet.next()) {

log.info("There is no data in the database!");

return null;

}

Todo todo = new Todo();

todo.setId(resultSet.getLong("id"));

todo.setDescription(resultSet.getString("description"));

todo.setDetails(resultSet.getString("details"));

todo.setDone(resultSet.getBoolean("done"));

log.info("Data read from the database: " + todo.toString());

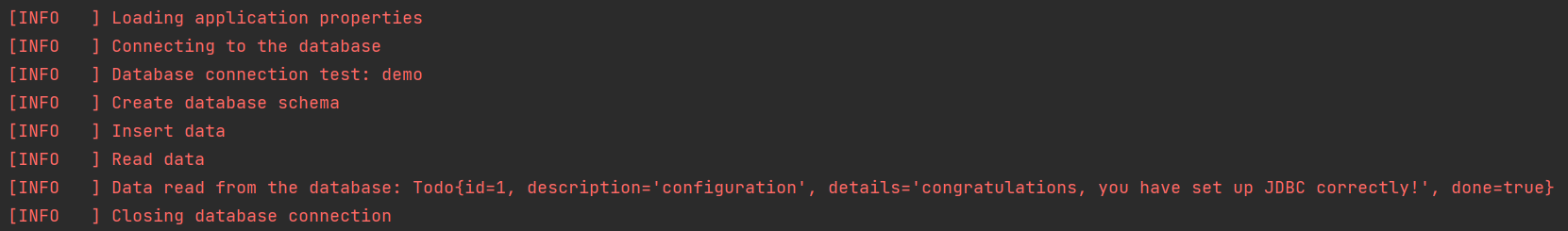
return todo;

}

**Kommentezzük ki a következő utasítást a DemoApplication osztályban:**

todo = readData(connection);

**Futtatás**



**Updating data in Azure Database for PostgreSQL**

src/main/java/com/example/demo/**DemoApplication**.java

**fájlhoz adjuk hozzá a következő metódust:**

private static void **updateData**(Todo todo, Connection connection) throws SQLException {

log.info("Update data");

PreparedStatement updateStatement = connection

.prepareStatement("UPDATE todo SET description = ?, details = ?, done = ? WHERE id = ?;");

updateStatement.setString(1, todo.getDescription());

updateStatement.setString(2, todo.getDetails());

updateStatement.setBoolean(3, todo.isDone());

updateStatement.setLong(4, todo.getId());

updateStatement.executeUpdate();

readData(connection);

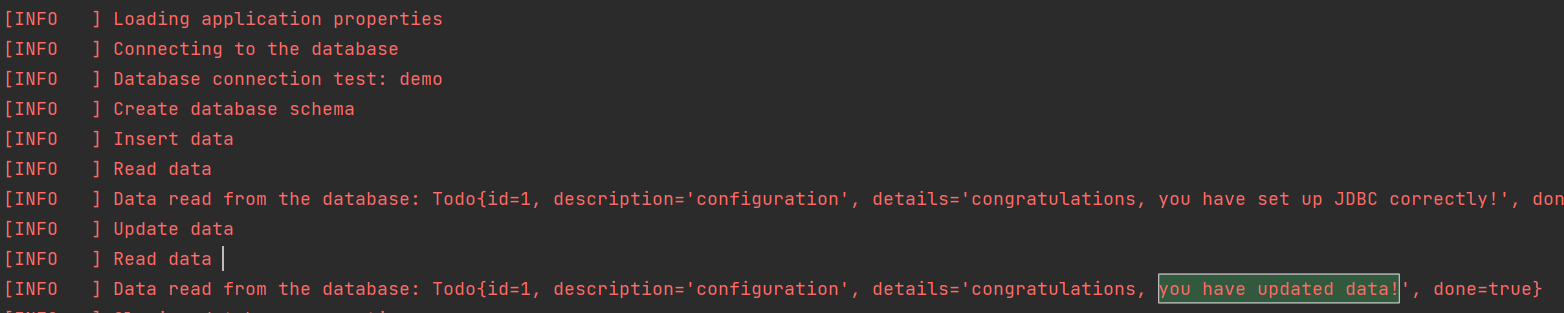
}

**Kommentezzük ki a következő utasításokat a DemoApplication osztályban:**

todo.setDetails("congratulations, you have updated data!");

updateData(todo, connection);

**Futtatás**



**Deleting data in Azure Database for PostgreSQL**

src/main/java/com/example/demo/**DemoApplication**.java

**fájlhoz adjuk hozzá a következő metódust:**

private static void **deleteData**(Todo todo, Connection connection) throws SQLException {

log.info("Delete data");

PreparedStatement deleteStatement = connection.prepareStatement("DELETE FROM todo WHERE id = ?;");

deleteStatement.setLong(1, todo.getId());

deleteStatement.executeUpdate();

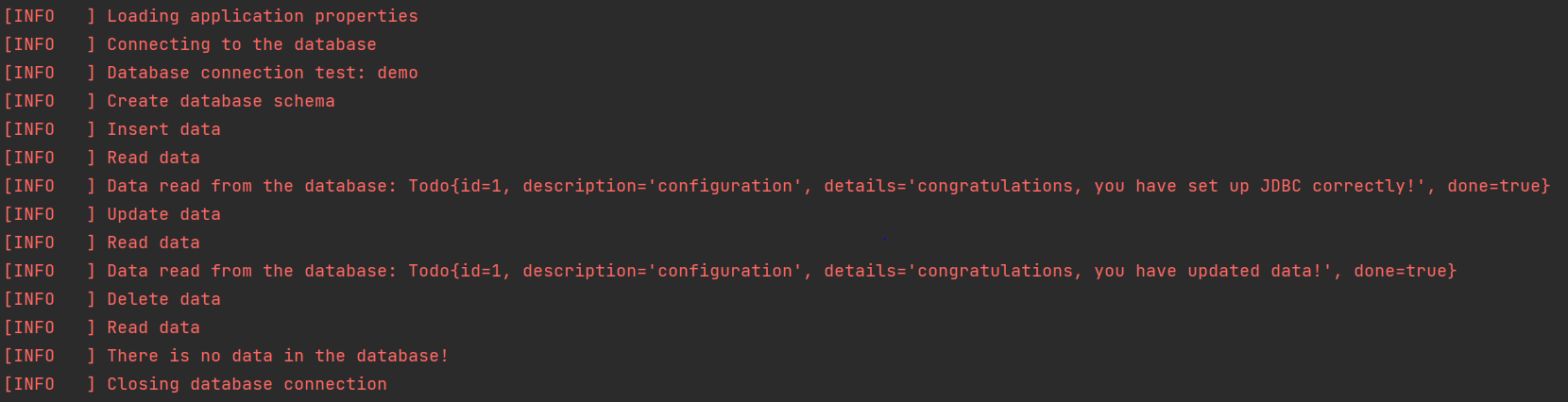
readData(connection);

}

**Kommentezzük ki a következő utasítást a DemoApplication osztályban:**

deleteData(todo, connection);

**Futtatás**



# 3. feladat - Alkalmazás adatbázissal – Java kód feltöltése és futtatása az Azure felhőben

A gyakrolaton tanult első adatbázisos példát használjuk itt fel:

Spring-Boot-Adatbázis-JPA - Hibernate-MySQL-CRUD alkalmazás

## Alkalmazás elkészítése és futtatása helyi gépen – az Azure adatbázis elérésével

Készítsük el a **dolgozo** táblát az Azure Postgresql adatbázisban. pl. pgAdmin vagy Azure Data Studio alkalmazásokkal.

CREATE TABLE dolgozo (

id SERIAL PRIMARY KEY,

nev varchar(45) NOT NULL,

cim varchar(70) NOT NULL,

kor int NOT NULL);

INSERT INTO dolgozo (nev,cim,kor) VALUES

('Tóth István','Debrecen',35),

('Nagy Éva','Szeged',20),

('Horváth Péter','Kecskemét',23);

Készítsük el az alkalmazást, vagy használjuk fel a **Megoldasok.zip** fájlban lévő megoldást.

Ellenőrizzük le: az alkalmazás működjön a helyi gépen!

**system.properties: Java verzió megadása**

Azure-ban lehet, hogy nem kötelező (más szolgáltatónál meg kell adni)

pom.xml fájlban nézzük meg a Java verziót:

<java.version>**11**</java.version>

Készítsünk a Projekt mappájában egy **system.properties** fájlt ezzel a tartalommal:

java.runtime.version=11

**pom.xml módosítása**

kivenni:

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

adjuk hozzá:

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

</dependency>

**src\main\resources\application.properties módosítása**

server.port=8888

spring.datasource.url=jdbc:postgresql://szerver1.postgres.database.azure.com:5432/database1?ssl=true&sslmode=require

spring.datasource.username=user1@szerver1

spring.datasource.password=Proba12345

spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.hibernate.ddl-auto=update

**Jól működik minden funkció!!!**

**Itt még helyi gépen fut:**

<http://localhost:8888/>

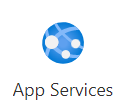


## Alkalmazás feltöltése a GitHub-ra

**GITHUB-GITLAB használat.docx** fájlban leírtak alapján töltsük fel a projektet a GitHub-ra

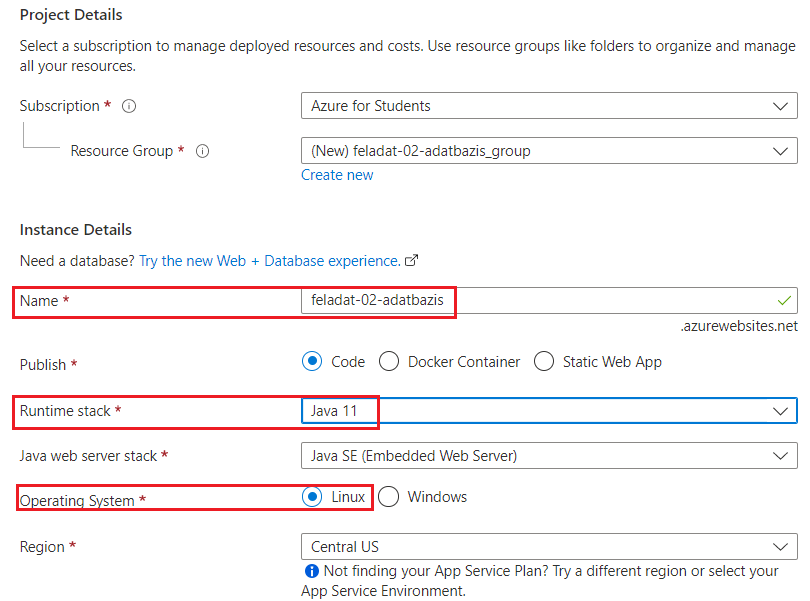
pl. <https://github.com/suzo777/JavaSpringBoot2>

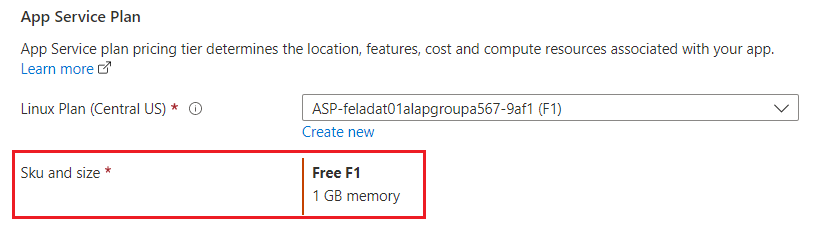
## App service létrehozás

 => 

**Name**: **feladat-02-adatbazis** Egyedinek kell lenni az Interneten:

<https://feladat-02-adatbazis.azurewebsites.net>







## Feltöltés az Azure-ra: letöltés GIT-ről, Build, Deploy

<https://shell.azure.com/>

pl.

git clone <https://github.com/suzo777/JavaSpringBoot2>

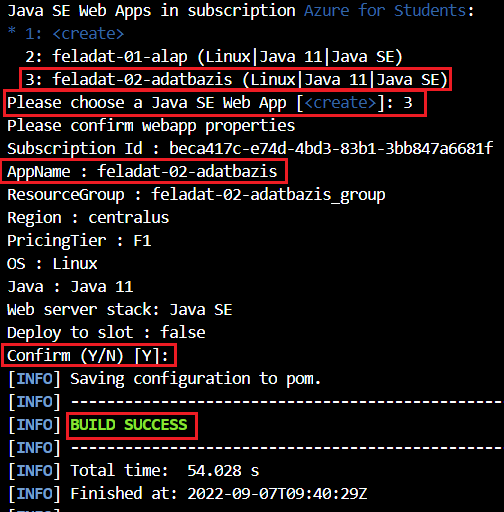
Ctrl+Shift+V +Enter

**mappába belépni:**

cd JavaSpringBoot2

mvn com.microsoft.azure:azure-webapp-maven-plugin:2.5.0:config

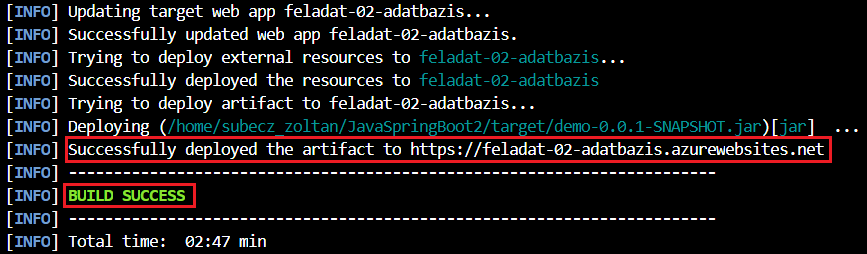
Ctrl+Shift+V +Enter



AppName : feladat-02-adatbazis

mvn package azure-webapp:deploy

Ctrl+Shift+V +Enter



**Minden funkció jól működik:**

Még van benne egy minimális hiba az első új dolgozó hozzáadása után: üres táblát mutat.

Ezt oldják meg gyakorló feladatként

<https://feladat-02-adatbazis.azurewebsites.net/>

