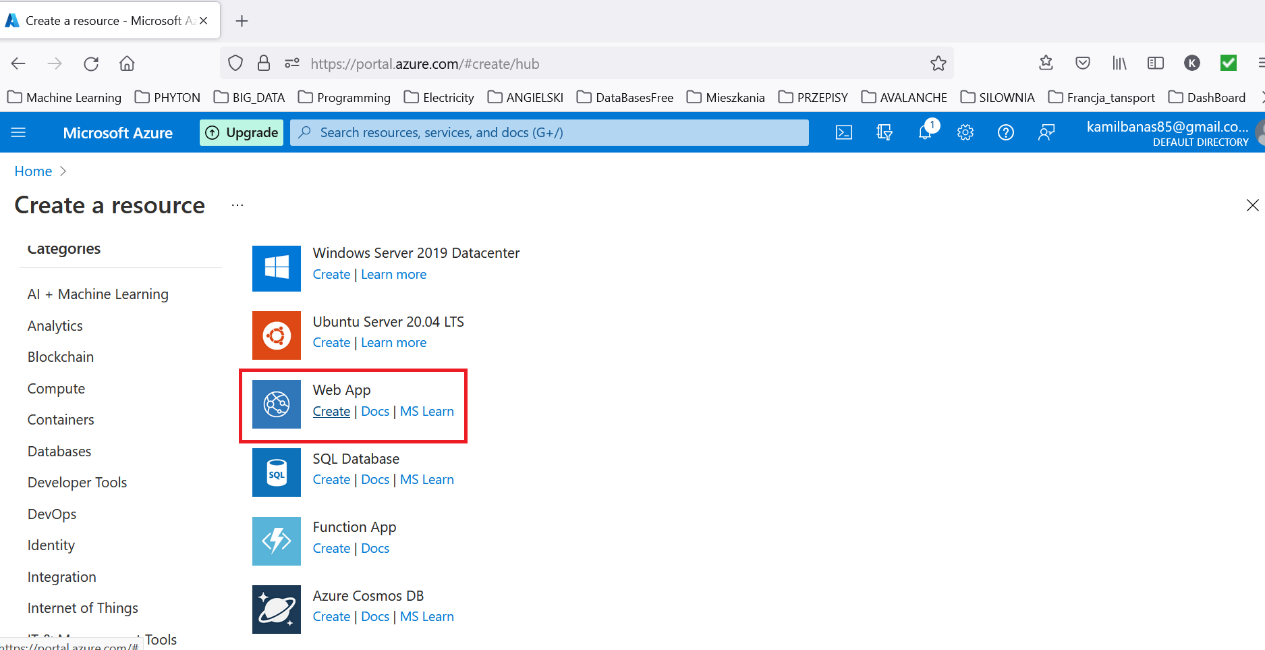
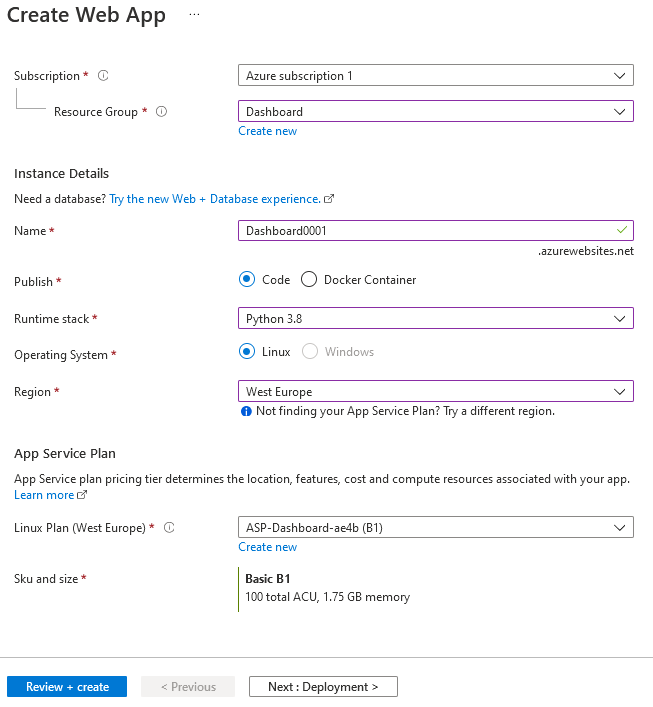
# Create Dash App

Create Recourses -> Create App



Set up:

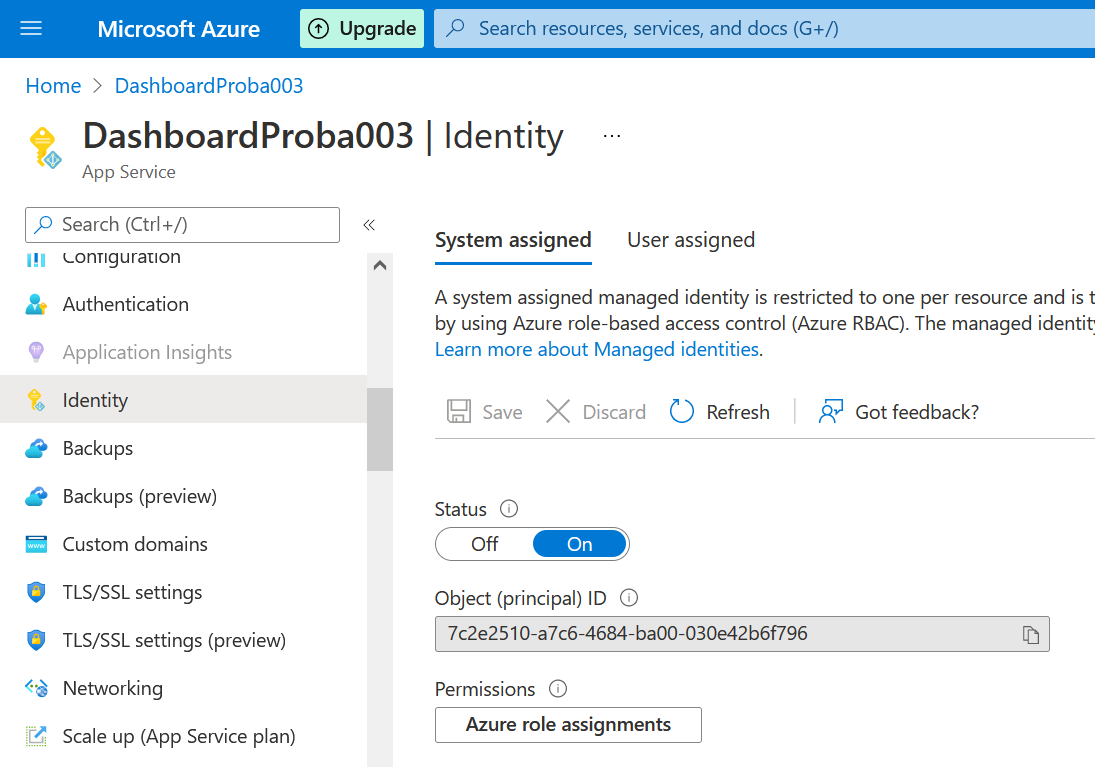
* Subscription
* Resources group
* Name of App
* Deployment from code or doceker
* If code: runtime stack -> Phyton 3.8
* Region
* Capacity of server…



**Review + Create**

**Enable Managed Identity:**

This enables add App Service as user of data base, so it can read data from DB.

****

## Example of app code

import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import pyodbc as pyodbc

import pandas as pd

import plotly.express as px

from logzero import logger

#%%

### The environment variables "MSI\_ENDPOINT" and "MSI\_SECRET" are created by

### the App Service when you turn on managed identities

# https://github.com/HediHargam/AzureMSI-connect-webapp-to-sqldb/blob/main/sql\_db\_conn.py

# import os

# import requests

# import pyodbc

# import struct

# import pandas as pd

# from logzero import logger

# def get\_token\_with\_msi(resource\_uri: str):

# "Retrieve token in the right format (binary structure) for SQL connection with SQL\_COPT\_SS\_ACCES\_TOKEN attribute"

#

# identity\_endpoint = os.environ["IDENTITY\_ENDPOINT"]

# identity\_header = os.environ["IDENTITY\_HEADER"]

#

# token\_auth\_uri = (

# f"{identity\_endpoint}?resource={resource\_uri}&api-version=2019-08-01"

# )

# head\_msi = {"X-IDENTITY-HEADER": identity\_header}

#

# resp = requests.get(token\_auth\_uri, headers=head\_msi)

# access\_token = bytes(resp.json()["access\_token"], "utf-8")

# exptoken = b""

# for i in access\_token:

# exptoken += bytes({i})

# exptoken += bytes(1)

# struct\_token = struct.pack("=i", len(exptoken)) + exptoken

#

# return struct\_token

def get\_data\_from\_sql\_db(

server: str, database: str, driver: str, query: str

) -> pd.DataFrame:

""""Get data from SQL database with MSI Authentication and return df from executed query"""

# token = get\_token\_with\_msi(resource\_uri="https://ossrdbms-aad.database.windows.net") # May be useful for PostrgresSQL

with pyodbc.connect(

"Driver="

+ driver

+ ";Server="

+ server

+ ";PORT=1433;Database="

+ database

+ ";Authentication=ActiveDirectoryMsi",

) as conn:

logger.info("Successful connection to database")

with conn.cursor() as cursor:

cursor.execute(query).fetchall()

data = pd.read\_sql(query, conn)

return data

#%%

sqlQuery = """

SELECT \*

FROM [SalesLT].[Product]

"""

driver = '{ODBC Driver 17 for SQL Server}'

server = 'server--01.database.windows.net'

database = 'Baza01'

# username = 'KamilBanas85'

# password = 'Francja2022!'

DF = get\_data\_from\_sql\_db( server = server,

database = database,

driver = driver,

query = sqlQuery,

)

DF = DF[['ProductID','ProductModelID']]

#%%

dash\_app = dash.Dash(\_\_name\_\_)

app = dash\_app.server

dash\_app.layout = html.Div(children=[

html.H1(children='Hello Dash'),

dcc.Graph(id='example-graph',

figure = px.scatter(DF, x="ProductID", y="ProductModelID")

)

])

if \_\_name\_\_ == '\_\_main\_\_':

dash\_app.run\_server(debug=True)

## Deplyment from local git

cmd -> conda prompt

select directory of files -> F: -> cd F:\Dash\Dash\_Proba\_05\_\_data\_from\_DB\_\_idenity

In cmd:

# In spyder find phyton exe file

import sys

sys.executable -> C:\\Users\\kamil\\anaconda3\\python.exe

# create and set up enviromental variables (to install only required libraries and crete ‘requirments’ file with these libreries)

python -m pip install --user virtualenv or pip install virtualenv

C:\\Users\\kamil\\anaconda3\\python.exe -m venv venvName

.\venvName\Scripts\activate

# install required libreries

pip install dash

pip install plotly

pip install pandas

pip install pyodbc

pip install logzero

# create requirment file

pip freeze > requirements.txt

# include text file to exclude enviomental variable file: file name: .gitignore with text: venvName/

# test app

C:\\Users\\kamil\\anaconda3\\python.exe app.py

# web can be find: <http://localhost:8050>

# make commit

git init

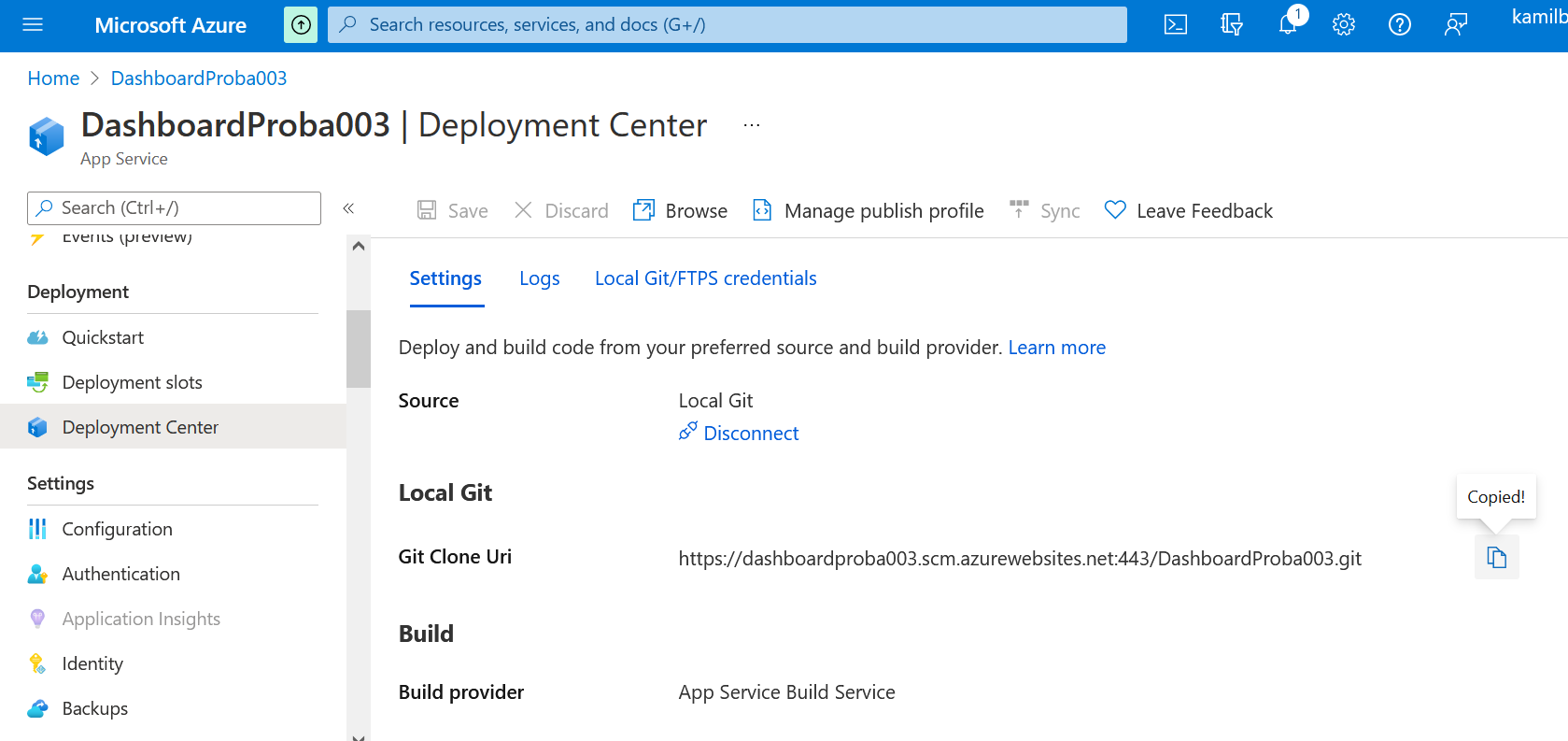
git add .

git commit –m “menneged idenity”

# set up remote repo with scheme: git remote add NameOfRemoteRepo UrlToRemoteRepo

git remote add azure <https://dashboardproba003.scm.azurewebsites.net:443/DashboardProba003.git>

where url can be find in ‘deplyment center’:



# push commit to rempte repo

git push NameOfRemoteRepo master

Other git usefull comands:

# change remote repo (f.e.when new app is created and url change): git remote set-url RepoName NewUrl

git remote set-url azure <https://dashboardfromlocalgit01.scm.azurewebsites.net:443/DashboardFromLocalGit01.git>

# Synchronize local repo with master repo (take old commits from remote to local): git push RemoteRepoName master

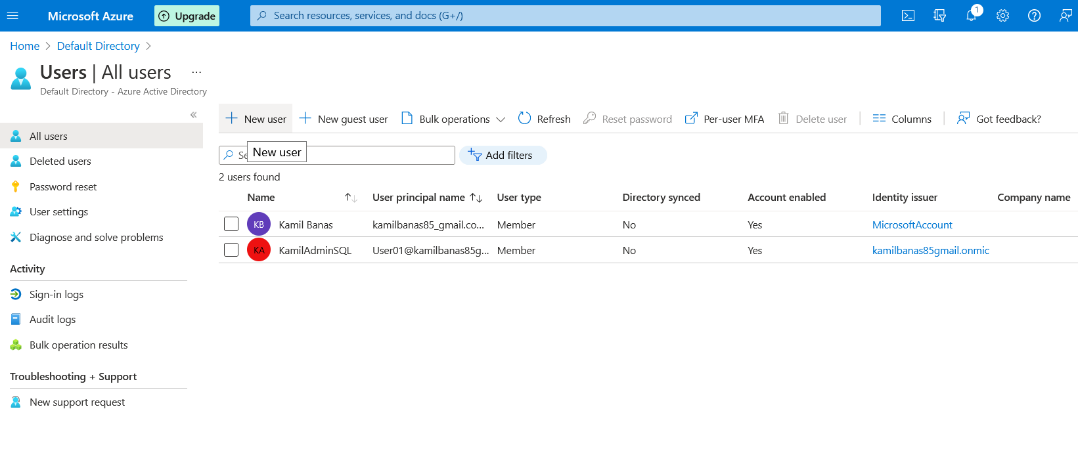
git push azure master

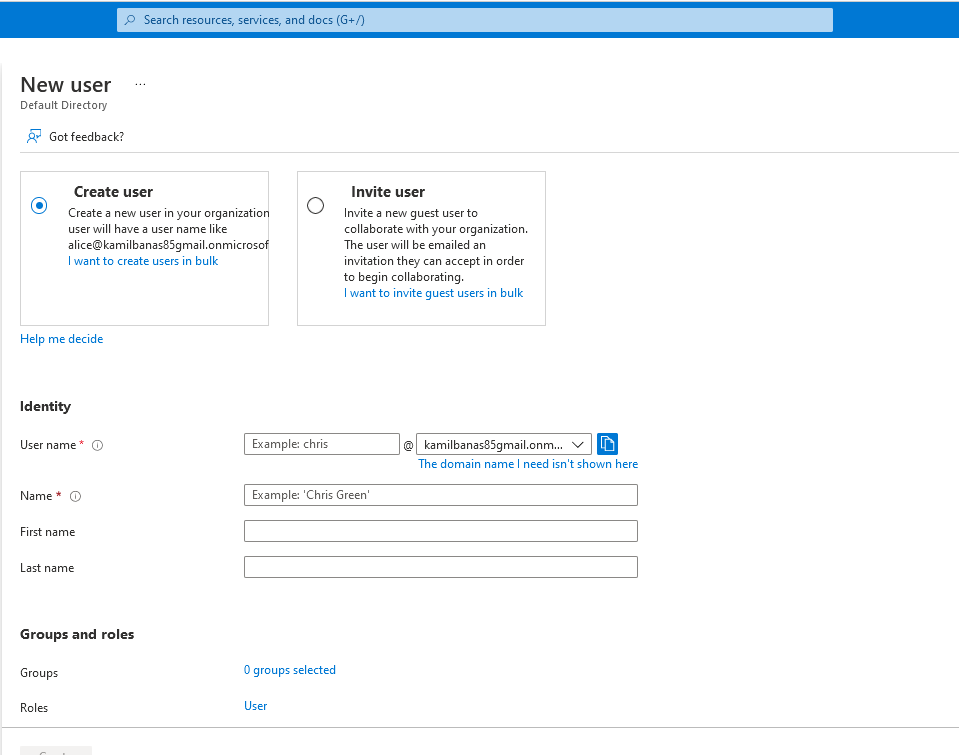
# Create SQL DataBase

## Create User – Admin DB

To create Active Directory admin of SQL Database it is require to create user which is not a global Admin ?

So create user in Active Directory:





## Create SQL Data Base

<https://docs.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?tabs=azure-portal>

Set up:

* Resource group
* DataBase name
* Server -> Create New (if doesn’t exist) ->

Set up:

* Server Name
* Location
* Autentication -> Both AD and SQL (AD – user created, SQL – login and password)
* Network Connectivity -> Public endpoint
* Allow Azure services and resources to access this server -> On
* Backup storage redundancy -> Local Redundant ?
* Data source -> Sample !! If Example !!

**Create**

After Creation:

* Add permition to App Service Managed Identity to read data from SQL (Logged as Admin !):

CREATE USER [user@sqlity.com] FROM EXTERNAL PROVIDER;

ALTER ROLE db\_datareader ADD MEMBER [user@sqlity.com];

ALTER ROLE db\_datawriter ADD MEMBER [user@sqlity.com];

ALTER ROLE db\_ddladmin ADD MEMBER [user@sqlity.com];

or group:

ALTER ROLE db\_datareader ADD MEMBER [AzureADGroupName];

GO

Check Users:

select name as username,

create\_date,

modify\_date,

type\_desc as type,

authentication\_type\_desc as authentication\_type

from sys.database\_principals

where type not in ('A', 'G', 'R', 'X')

and sid is not null

and name != 'guest'

order by username;