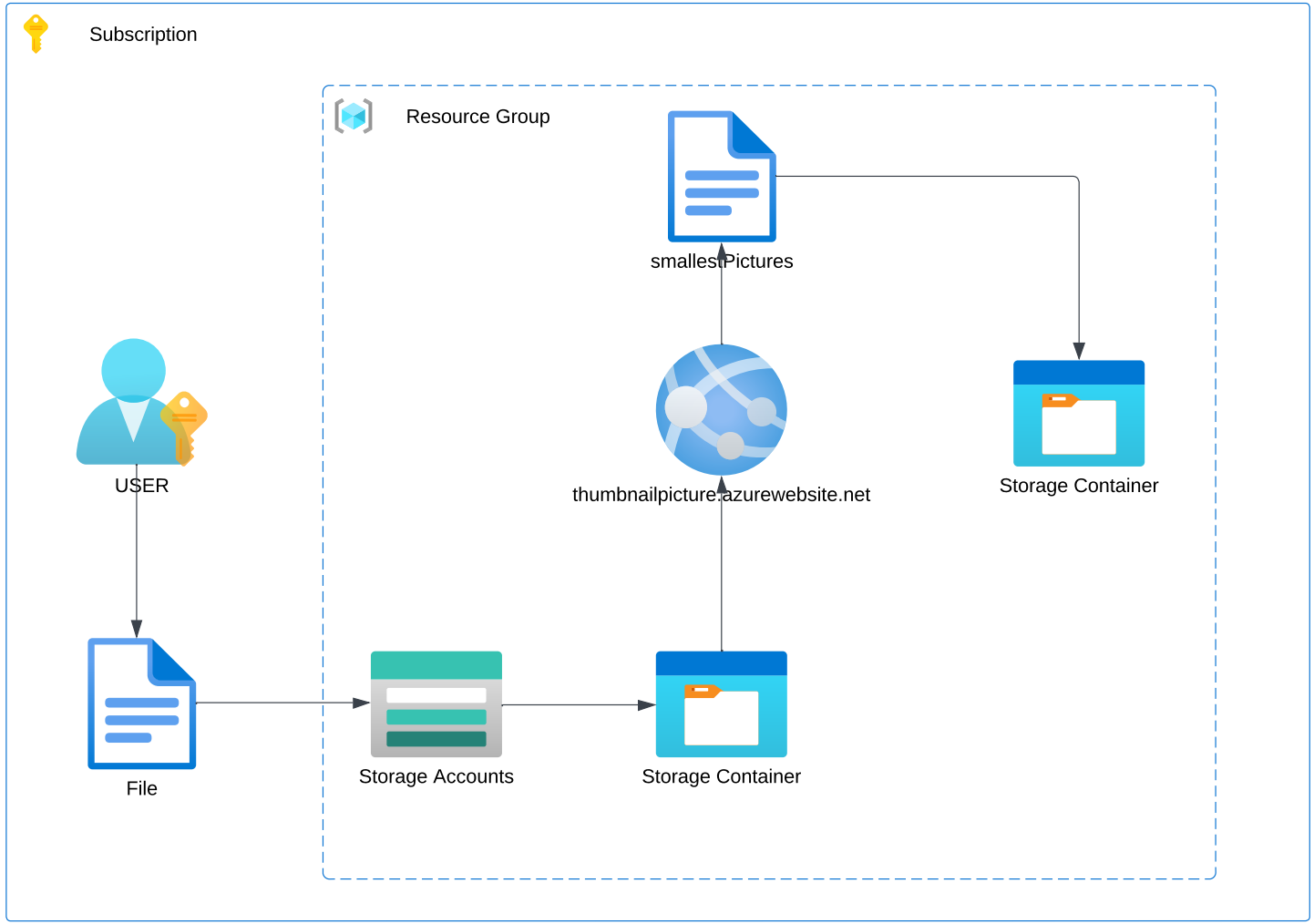
THUMBNAIL GENERATOR

Strenghts: This architecture was choosen due to it is easy to configurate. The net paymet is pay as you go and it is cheaper if one is not used at all. Within Python, the PIL library helps to develop the requirement in an easily way.

Weaknesses: If the knowlede about the architecture is not much enough, it is neccesary a guide or follow the documation within azure. On the other hand, if the amount of usage increase when files are load to become smaller, could get some extra charges. The more it is used the more it is paid.

1. The user load a image file to to the Storage Container which is in a Storage Account.
2. Once the file is in the input container called "myblob" the app service, which is runing within Azure cloud by Microsoft, takes the file an process it through a Thumbnail generator.
3. Once the image file has been reached the size, it is pulled in another Storage container within Azure.



Python File

Function_app.py

```
import azure.functions as func
import logging
from PIL import Image
from io import BytesIO

app = func.FunctionApp()

@app.blob_trigger(arg_name="myblob", path="original/{name}",
                  connection="AzureWebJobsStorage")
@app.blob_output(arg_name="outputBlob",path="newpicture/{name}",connection="AzureWebJobsStorage")
def smallestPictures(myblob: func.InputStream, outputBlob:func.Out[func.InputStream]):
    logging.info(f"Python blob trigger function processed blob"
             f"Name: {myblob.name}"
             f"Blob Size: {myblob.length} bytes")

    image = Image.open(myblob)

    thumbnail_size = (128,128)

    image.thumbnail(thumbnail_size)

    thumb_stream = BytesIO()

    image.save(thumb_stream,format='JPEG')
    thumb_stream.seek(0)

    outputBlob.set(thumb_stream)
```