Python API Pool Resource Exercise

1. Objective

The main goal of this exercise is to write Restful API which allow users to manage pools and their resources by using standards http methods for RESTFUL

2. API Definitions

Build RESTFUL API that support the following

2.1. **Pool**

It is a container that contains a list of (resources) belongs to that object and each pool could have the following details:

1. ID (Unique ID): Id for the pool
2. Name: Name of the pool
3. Resources: List of resources associated with it

This model should support the following API endpoints:

A. **Create Pool :** Create pool object

```
Url: /api/pools/{id}
HTTP Method: PUT

Note: ID must be unique and provided as part of the URL

Payload

{
   "name":"test_pool",
   "resources":[
      "1.1.1.1",
      "2.2.2.2",
      "3.3.3.3"
   ]
}
```

```
Response 201 OK

{
   "name":"test_pool",
   "resources":[
      "1.1.1.1",
      "2.2.2.2",
      "3.3.3.3"
   ]
}
```

B. **Get Pool:** This endpoint should be able to get pool with all it resources associated with it and include status for each resource (whether it is **released** or **allocated**)

```
Url: /api/pools/{id}
HTTP Method: GET


Response 200 OK

{
   "name":"test_pool",
   "resources":[
      {
       "id": "f331c635-6cd1-4f5a-9d00-b36642ce9b07",
       "ip_address": "1.1.1.1",
       "status": "ALLOCATED"
      },
      {
       "id": "9fa703d9-8f87-44be-8e8e-62ae3d48a5b8",
       "ip_address": "2.2.2.2",
       "status": "RELEASED"
      },
      {
       "id": "12f68b8f-98b3-43da-bee4-aed9fcf1c410",
       "ip_address": "3.3.3.3",
       "status": "RELEASED"
      }
   ]
}
```

C. **List Pool:** This endpoint should be able to List all pools with all their resources associated with them and include status for each resource (whether **released** or **allocated**)

```
Url: /api/pools
HTTP Method: GET
Response 200 OK
{
   "items":[
      {
         "name":"test_pool",
         "resources":[
            {
               "id":"f331c635-6cd1-4f5a-9d00-b36642ce9b07",
               "ip_address":"1.1.1.1",
               "status":"ALLOCATED"
            },
            {
               "id":"9fa703d9-8f87-44be-8e8e-62ae3d48a5b8",
               "ip_address":"2.2.2.2",
               "status":"RELEASED"
            },
         ]
      },
      {
         "name":"pool2",
         "resources":[
            {
               "id":"42c80e0e-8037-42e3-8385-df072b725595",
               "ip_address":"4.4.4.4",
               "status":"RELEASED"
            },
            {
               "id":"b6caac22-5190-4e7d-bed9-01f4882c06c2",
               "ip_address":"5.5.5.5",
               "status":"RELEASED"
            },
         ]
      }
   ]
}
```

D. **Delete Pool :** This should delete pool with all the it resources. Note: It should not allow to delete pool if its already has some at least one resource is allocated and should return **409 status code**

```
Url: /api/pools/{id}
HTTP Method: Delete

Successful Response 204

Error Responses

409 Conflict When it is not allowed to remove pool
404 Resource is not found
```

2.2. Resource

It is an address object that belongs to a certain pool resource and it should support the following actions

A. **Allocate resource:** This method should be able to allocate new resource that belong to its pool and mark that object as **ALLOCATED**

```
Url: /api/pools/{id}/allocate
HTTP Method: PUT

Payload
{
 "id":"42c80e0e-8037-42e3-8385-df072b725595"
}

Response 200 OK
{
 "id":"42c80e0e-8037-42e3-8385-df072b725595",
 "ip_address":"4.4.4.4",
 "status":"ALLOCATED"
}
```

B. **Release resource:** This method should be able to release resource to its original pool and marked it as **RELEASED**

```
Url: /api/pools/{id}/release
HTTP Method: PUT

Payload
{
 "id":"42c80e0e-8037-42e3-8385-df072b725595"
}
Response 200 OK
{
 "id":"42c80e0e-8037-42e3-8385-df072b725595",
 "ip_address":"4.4.4.4",
 "status":"RELEASED"
}
```

C. **Add new resource to existing pool:** This method should be able to add/create a new resource to certain pool

```
Url: /api/pools/{id}/resource/add
HTTP Method: POST

Payload
{
 "ip_address":"7.7.7.7"
}
Response 200 OK
{
 "id":"00416893-7a22-4abf-94f6-292e3fb08bbb",
 "ip_address":"7.7.7.7",
 "status":"RELEASED"
}
```

D. **Delete resource from existing pool:** This method should be able to remove a resource from certain pool

```
Url: /api/pools/{id}/resource/remove/{resource_id}
HTTP Method: DELETE

Successful Response 204

404 Resource is not found
```

E. **Get Resource:** This method should be able to get info about current resource

```
Url: /api/pools/{id}/resource/{resource_id}
HTTP Method: GET

Response 200 OK
{
 "id":"00416893-7a22-4abf-94f6-292e3fb08bbb",
 "ip_address":"7.7.7.7",
 "status":"RELEASED",
 "pool_name":"pool-1",
}

404 Resource is not found
```

Note: in the API mentioned above there is {id} & {resource_id} these are represent the following:

1. Id : Related to pool
2. Resource_id: Related to resource

3. Requirements

This exercise should be implemented using the Python programming language (2.x or 3.x) and Flask framework and need to take care of the following:

1. Implemented using Python best practices
2. Implemented using PEP8 standard
3. Implemented the API as illustrated in the API definitions section
4. Implement and add unit tests for the code
5. Run code against PEP8 to make sure it is written as expected
6. The code should be hosted in private GitHub repository
7. Your GitHub repository should include README about the task and how to run it
8. **No need to have a user management functionality but the code should be written in a way that can be added later on**