Υποδομές Cloud

Διάλεξη 5η

Θωμάς Καμαλάκης

Χαροκόπειο Πανεπιστήμιο Αθηνών

Χειμερινό Εξάμηνο, 2024

Περιεχόμενα

1 Proxmox API: Εισαγωγή

2 Παραμετροποίηση VM

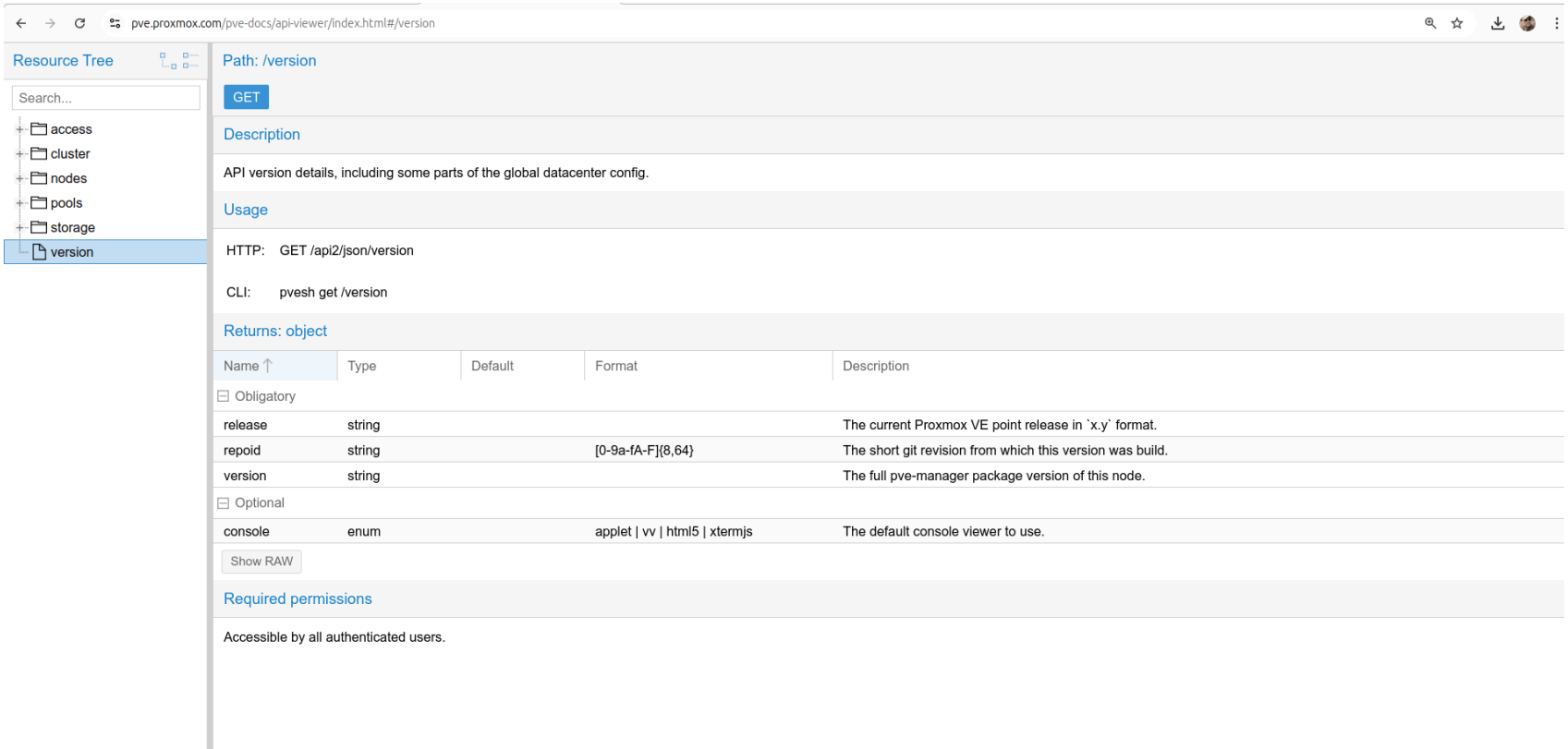
3 Permissions

4 Ένα ολοκληρωμένο παράδειγμα

Εισαγωγικά

- Proxmox έχει ένα Application Programming Interface που επιτρέπει τη διαχείριση και την αλληλεπίδραση με την πλατφόρμα.
- Διαχείριση Πόρων: Το API δίνει τη δυνατότητα δημιουργίας, διαγραφής και τροποποίησης εικονικών μηχανών, κτλ.
- Παρακολούθηση Συστήματος: Παρέχει πρόσβαση σε δεδομένα που αφορούν τη χρήση των πόρων.
- Αυτοματισμός: Μέσω του API, οι χρήστες μπορούν να αυτοματοποιούν διαδικασίες, όπως η δημιουργία εικονικών μηχανών.

Documentation



Python requests

- Υποστήριξη HTTP μεθόδων: Υποστηρίζει όλες τις βασικές μεθόδους (GET, POST, PUT, DELETE κτλ).
- Παρέχει εύκολους τρόπους για να στείλει κανείς παραμέτρους (query parameters), δεδομένα από φόρμες και JSON payload.
- Διαχειρίζεται αυτόματα τα cookies και τις συνεδρίες, διευκολύνοντας τη διατήρηση των δεδομένων συνεδρίας μεταξύ αιτημάτων.
- Επιτρέπει την προσαρμογή των HTTP headers για τον έλεγχο της συμπεριφοράς του αιτήματος.
- Παρέχει χειρισμούς για σφάλματα, όπως προβλήματα σύνδεσης ή αποτυχίες απαντήσεων HTTP (status codes).
- SSL/TLS: Επιτρέπει την ασφαλή επικοινωνία μέσω HTTPS και προσφέρει δυνατότητες επαλήθευσης πιστοποιητικών.

Hello world

```python
import requests

secret = 'fef0676b-0307-4c0a-988f-5b5e7b9194ee'
tokenid = 'admin'
url = 'https://10.100.52.11:8006/api2/json/version/'

headers = {
    'Authorization' : 'PVEAPIToken=thkam@HUA!%s=%s' %(tokenid, secret)

}

r = requests.get(url, headers=headers, verify=False)
print(r.text)
```

```
thomas@kirklaptop:~/proxmox$ python3 -i helloworld.py
/usr/lib/python3/dist-packages/urllib3/connectionpool.py:1100: InsecureRequestWarning: Unverified HTTPS request is being made to host '10.100.52.11'. Adding certificate verification is st
rongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
{"data":{"release":"8.2","repoid":"9355359cd7afbae4","version":"8.2.2"}}
```

Οι χρήστες του Proxmox

```python
import requests

secret = '87dc4dcb-bd9c-4257-be71-b63676cfa0db'
tokenid = 'admin'
url = 'https://10.100.52.11:8006/api2/json/access/users'

headers = {
    'Authorization' : 'PVEAPIToken=root@pam!%s=%s' %(tokenid, secret)

}

r = requests.get(url, headers=headers, verify=False)
users = r.json()['data']
print(users)
for u in users:
    print(u)
```

```
[{'enable': 1, 'realm-type': 'ldap', 'userid': 'alexdem@HUA', 'expire': 0}, {'expire': 0, 'enable': 1, 'email': 'thkam@hua.gr', 'real
m-type': 'pam', 'userid': 'root@pam'}, {'expire': 0, 'enable': 1, 'realm-type': 'ldap', 'userid': 'thkam@HUA'}]
{'enable': 1, 'realm-type': 'ldap', 'userid': 'alexdem@HUA', 'expire': 0}
{'expire': 0, 'enable': 1, 'email': 'thkam@hua.gr', 'realm-type': 'pam', 'userid': 'root@pam'}
{'expire': 0, 'enable': 1, 'realm-type': 'ldap', 'userid': 'thkam@HUA'}
```

Τα nodes του Proxmox

proxmox > getnodes.py > ...

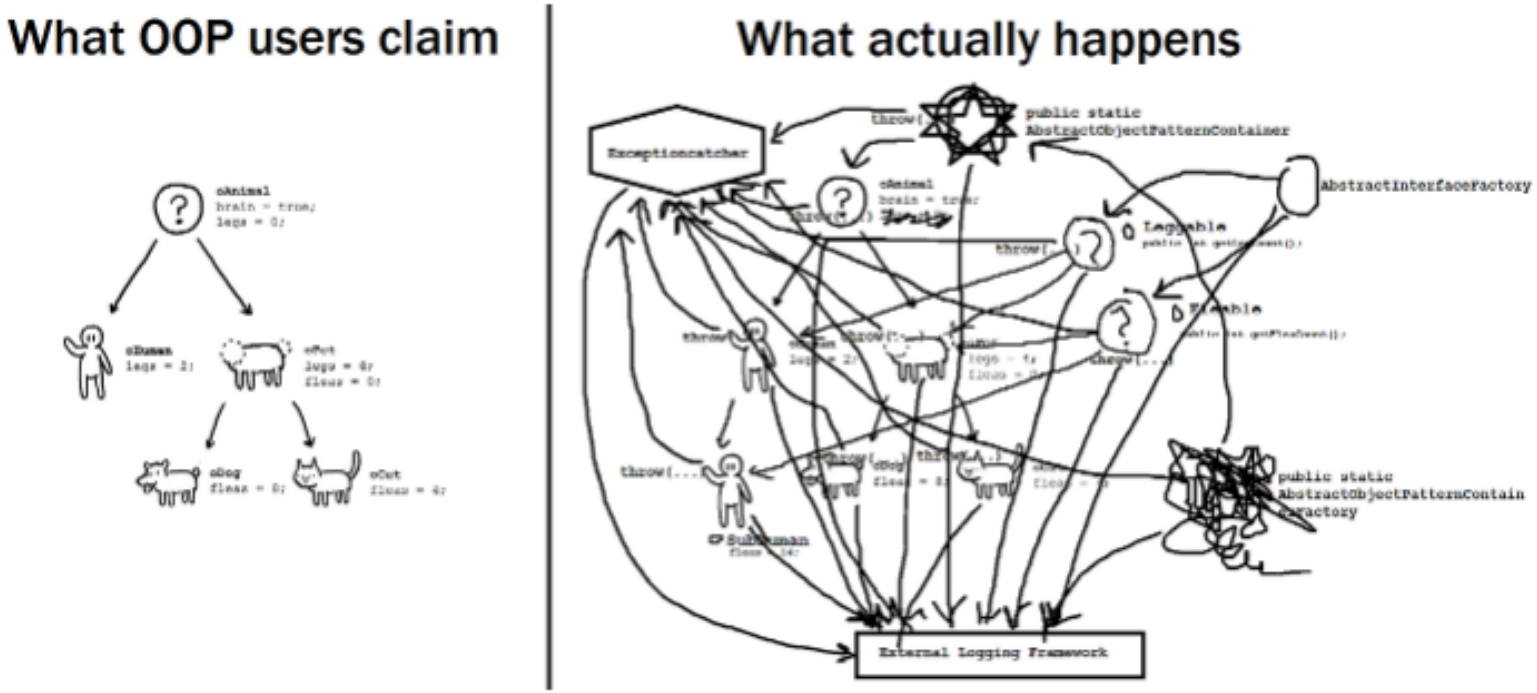
```python
import requests

secret = '87dc4dcb-bd9c-4257-be71-b63676cfa0db'
tokenid = 'admin'
url = 'https://10.100.52.11:8006/api2/json/nodes'

headers = {
    'Authorization' : 'PVEAPIToken=root@pam!%s=%s' %(tokenid, secret)

}

r = requests.get(url, headers=headers, verify=False)
nodes = r.json()['data']
for n in nodes:
    print(n)
```

```
{'node': 'edprox2', 'id': 'node/edprox2', 'maxcpu': 4, 'mem': 3625586688, 'ssl_fingerprint': '3F:78:9D:31:58:C3:7A:BA:65:78:1E:B1:93
A0:59:1F:81:3D:ED:8F:72:A7:E3:F4:29:3C:D2:FC:BA:CC:21:B9', 'level': '', 'status': 'online', 'cpu': 0.0439897698209719, 'maxmem': 167
3074944, 'disk': 3323527168, 'type': 'node', 'uptime': 174830, 'maxdisk': 13560061952}
{'mem': 3485306880, 'maxcpu': 4, 'ssl_fingerprint': '70:FA:E1:BB:5A:FB:09:BF:42:55:6A:BD:31:03:0C:77:57:E2:42:48:C1:93:76:1B:97:8B:9
:5D:16:45:3B:42', 'level': '', 'id': 'node/edprox1', 'node': 'edprox1', 'cpu': 0.151314110742536, 'maxmem': 16773074944, 'status': '
nline', 'disk': 3353194496, 'type': 'node', 'maxdisk': 13560061952, 'uptime': 174814}
{'maxmem': 16773074944, 'cpu': 0.0335151987529228, 'status': 'online', 'ssl_fingerprint': 'A8:53:9B:F4:A2:7C:DF:28:38:4F:CD:8E:7C:87
8C:E6:DD:8C:C3:10:F9:EE:89:9B:A5:75:06:1C:3B:2C:51:26', 'level': '', 'mem': 1746661376, 'maxcpu': 4, 'node': 'exprod3', 'id': 'node/
xprod3', 'maxdisk': 13560061952, 'uptime': 174812, 'type': 'node', 'disk': 3321516032}
```

Zen of Python

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

OOP


What OOP users claim
What actually happens
ExceptionCatcher
public static
AbstractObjectPatternContainer
AbstractInterfaceFactory
External Logging Framework

Proxmox API wrapper

```python
import requests

SECRET = '87dc4dcb-bd9c-4257-be71-b63676cfa0db'
TOKENID = 'admin'
URL = 'https://10.100.52.11:8006/api2/json/'

class api:

    def __init__(self,
                 secret = SECRET,
                 tokenid = TOKENID,
                 url = URL,
                 verify = False):

        self.secret = secret
        self.tokenid = tokenid
        self.url = url
        self.verify = verify

        self.headers = {
            'Authorization' : 'PVEAPIToken=root@pam!%s=%s' %(self. tokenid, self.secret)
        }

    def get(self, suburl):
        url = self.url + suburl
        print(url)
        return requests.get(url,
                            headers = self.headers,
                            verify = self.verify).json()['data']
```

Proxmox API wrapper

```
def get_nodes(self):
    return self.get('nodes')

def get_users(self):
    return self.get('users')

def get_node_vms(self, nodeid):
    suburl = 'nodes/{node}/qemu'.format(node=nodeid)
    print(suburl)
    return self.get(suburl)

def get_vms(self):
    nodes = self.get_nodes()
    nodeids = [n['node'] for n in nodes]
    vms = []
    for nodeid in nodeids:
        vms += self.get_node_vms(nodeid)

    return vms
```

Proxmox API wrapper

```
[{'mem': 1148218680, 'name': 'replica1', 'vmid': 102, 'uptime': 176766, 'diskwrite': 0, 'disk': 0, 'diskread': 0, 'netout': 88809, 'status'
 'running', 'cpu': 0.0234405581090273, 'pid': 15253, 'cpus': 1, 'maxdisk': 21474836480, 'maxmem': 2147483648, 'netin': 64555332}, {'status'
 'stopped', 'netout': 0, 'diskread': 0, 'diskwrite': 0, 'disk': 0, 'uptime': 0, 'name': 'vm3', 'vmid': 103, 'mem': 0, 'netin': 0, 'maxmem':
2147483648, 'cpus': 1, 'maxdisk': 34359738368, 'cpu': 0}, {'disk': 0, 'status': 'stopped', 'cpus': 1, 'vmid': 101, 'maxdisk': 1073741824, 'r
etout': 0, 'name': 'VM 101', 'uptime': 0, 'mem': 0, 'netin': 0, 'diskwrite': 0, 'maxmem': 536870912, 'cpu': 0, 'diskread': 0}, {'status': 's
topped', 'disk': 0, 'cpus': 1, 'vmid': 105, 'netout': 0, 'maxdisk': 21474836480, 'name': 'rf1', 'uptime': 0, 'mem': 0, 'netin': 0, 'maxmem'
 2147483648, 'diskwrite': 0, 'cpu': 0, 'diskread': 0}, {'disk': 0, 'status': 'running', 'cpus': 1, 'vmid': 100, 'netout': 133833, 'maxdisk'
 21474836480, 'name': 'ubuntutemplate', 'pid': 2092, 'uptime': 180919, 'mem': 625599248, 'netin': 67210677, 'diskwrite': 0, 'maxmem': 214748
3648, 'cpu': 0.0153081670369794, 'diskread': 0}, {'uptime': 81140, 'netin': 30948046, 'mem': 34214575, 'diskread': 0, 'cpu': 0.345160518183
33, 'maxmem': 2147483648, 'diskwrite': 0, 'cpus': 1, 'status': 'running', 'disk': 0, 'netout': 7512608, 'maxdisk': 1073741824, 'vmid': 104,
'name': 'VM 104', 'pid': 499754}]
>>> len(vms)
6
>>> [v['name'] for v in vms]
['replica1', 'vm3', 'VM 101', 'rf1', 'ubuntutemplate', 'VM 104']
```

Proxmox API wrapper

```
[{'mem': 1148218680, 'name': 'replica1', 'vmid': 102, 'uptime': 176766, 'diskwrite': 0, 'disk': 0, 'diskread': 0, 'netout': 88809, 'status'
 'running', 'cpu': 0.0234405581090273, 'pid': 15253, 'cpus': 1, 'maxdisk': 21474836480, 'maxmem': 2147483648, 'netin': 64555332}, {'status'
 'stopped', 'netout': 0, 'diskread': 0, 'diskwrite': 0, 'disk': 0, 'uptime': 0, 'name': 'vm3', 'vmid': 103, 'mem': 0, 'netin': 0, 'maxmem':
2147483648, 'cpus': 1, 'maxdisk': 34359738368, 'cpu': 0}, {'disk': 0, 'status': 'stopped', 'cpus': 1, 'vmid': 101, 'maxdisk': 1073741824, 'r
etout': 0, 'name': 'VM 101', 'uptime': 0, 'mem': 0, 'netin': 0, 'diskwrite': 0, 'maxmem': 536870912, 'cpu': 0, 'diskread': 0}, {'status': 's
topped', 'disk': 0, 'cpus': 1, 'vmid': 105, 'netout': 0, 'maxdisk': 21474836480, 'name': 'rf1', 'uptime': 0, 'mem': 0, 'netin': 0, 'maxmem'
 2147483648, 'diskwrite': 0, 'cpu': 0, 'diskread': 0}, {'disk': 0, 'status': 'running', 'cpus': 1, 'vmid': 100, 'netout': 133833, 'maxdisk'
 21474836480, 'name': 'ubuntutemplate', 'pid': 2092, 'uptime': 180919, 'mem': 625599248, 'netin': 67210677, 'diskwrite': 0, 'maxmem': 214748
3648, 'cpu': 0.0153081670369794, 'diskread': 0}, {'uptime': 81140, 'netin': 30948046, 'mem': 34214575, 'diskread': 0, 'cpu': 0.345160518183
33, 'maxmem': 2147483648, 'diskwrite': 0, 'cpus': 1, 'status': 'running', 'disk': 0, 'netout': 7512608, 'maxdisk': 1073741824, 'vmid': 104,
'name': 'VM 104', 'pid': 499754}]
>>> len(vms)
6
>>> [v['name'] for v in vms]
['replica1', 'vm3', 'VM 101', 'rf1', 'ubuntutemplate', 'VM 104']
```

Proxmox API wrapper

```
    def get_vm_params(self, nodeid, vmid):
        suburl = 'nodes/{node}/qemu/{vmid}/config'.format(node=nodeid,
                                                          vmid=vmid)
        return self.get(suburl)
```

```
{'cores': 1, 'cpu': 'x86-64-v2-AES', 'sockets': 1, 'ostype': 'l26', 'scsi0': 'ednfs:100/vm-100-disk-
.qcow2,iothread=1,size=20G', 'net0': 'virtio=BC:24:11:6C:C8:FB,bridge=vmbr0,firewall=1', 'digest': '
fe88b46b4d3e433e747a1fd08ef8f96adaa027e', 'name': 'ubuntutemplate', 'memory': '2048', 'ide2': 'ednfs
iso/ubuntu2404.iso,media=cdrom,size=2708862K', 'numa': 0, 'scsihw': 'virtio-scsi-single', 'agent': '
', 'meta': 'creation-qemu=8.1.5,ctime=1728496735', 'boot': 'order=scsi0;ide2;net0', 'smbios1': 'uuid
54195ce9-1ecc-40e3-88dc-c57bfd7e3d8a', 'vmgenid': '4f7c22dc-f292-458b-8da1-16d1a4cf211c'}
```

Δημιουργία VM

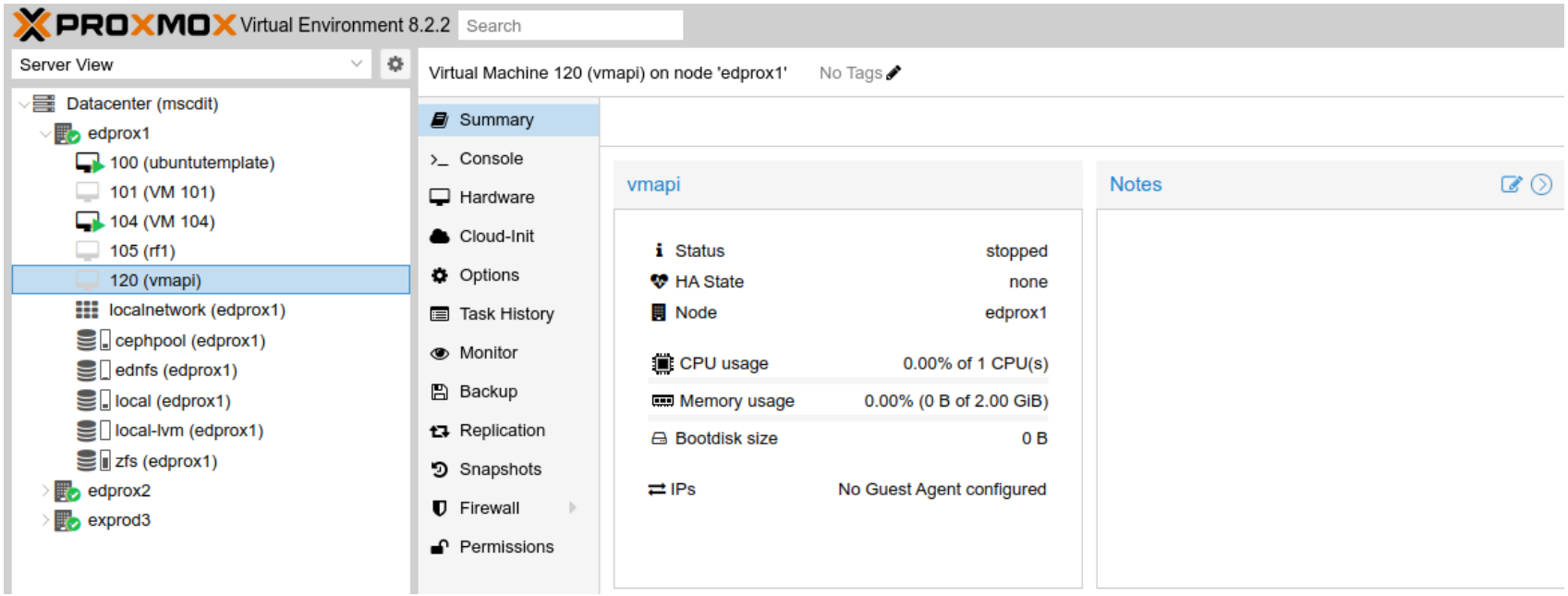
```
    def post(self, suburl, params):
        url = self.url + suburl
        return requests.post(url,
                             json = params,
                             headers = self.headers,
                             verify = self.verify)

    def get_nodes(self):
```

```
    def create_vm(self, nodeid, **kwargs):
        suburl = 'nodes/{node}/qemu'.format(node=nodeid)
        return self.post(suburl, params=kwargs)
```

Δημιουργία VM

```python
import requests
from proxmoxapi import api


a = api()
params = {'cores': 1,
          'vmid' : 120,
          'cpu': 'x86-64-v2-AES',
          'sockets': 1,
          'net0': 'virtio=BC:24:11:6C:C8:FF,bridge=vmbr0,firewall=1',
          'name': 'vmapi',
          'memory': '2048',
          'ide2': 'ednfs:iso/ubuntu2404.iso,media=cdrom,size=2708862K',
         }

a.create_vm('edprox1', **params)
```

Δημιουργία VM



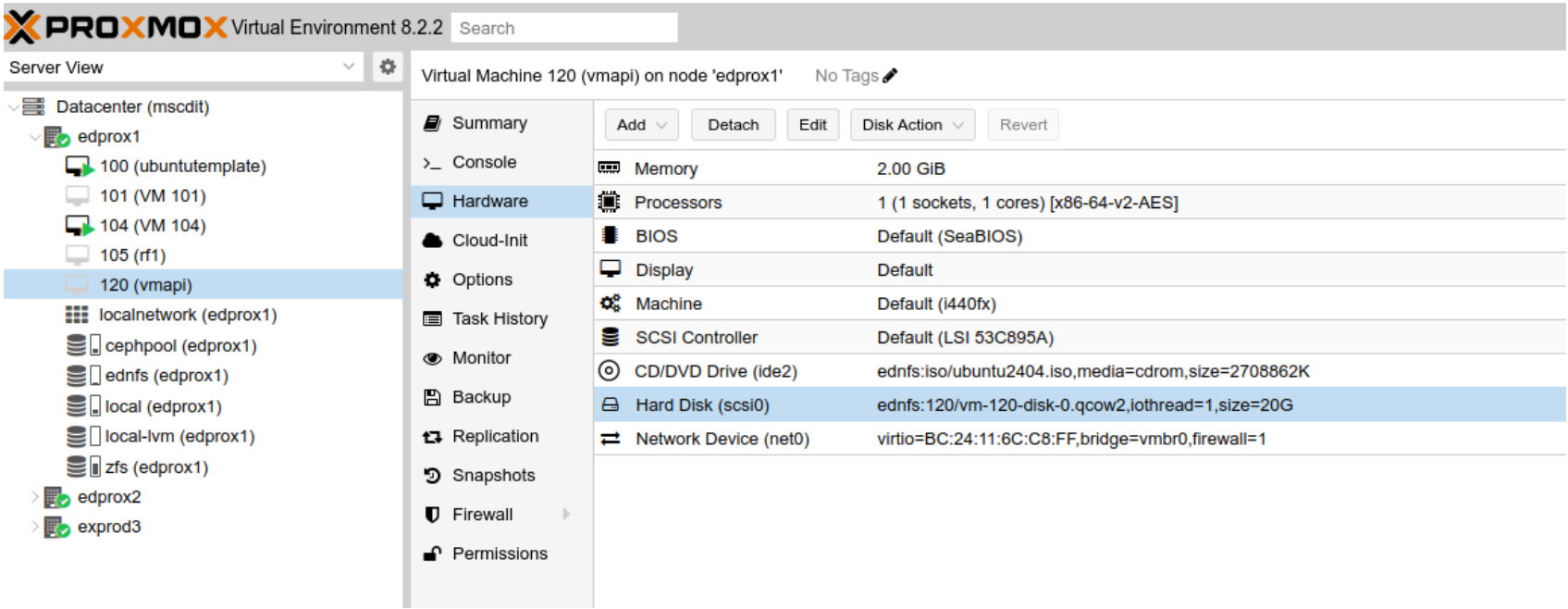
Δημιουργία Δίσκων

```
def create_disk(self, nodeid, storageid, **kwargs):
    suburl = 'nodes/{node}/storage/{storage}/content'.format(node=nodeid,
                                                             storage=storageid)
    return self.post(suburl, params=kwargs)
```

Δημιουργία Δίσκων

```python
import requests
from proxmoxapi import api


a = api()

params = {'filename' : 'vm-120-disk-0.qcow2',
          'size' : '20G',
          'vmid' : 120
         }

r = a.create_disk('edprox1', 'ednfs', **params)

params = {'cores': 1,
          'vmid' : 120,
          'cpu': 'x86-64-v2-AES',
          'sockets': 1,
          'net0': 'virtio=BC:24:11:6C:C8:FF,bridge=vmbr0,firewall=1',
          'name': 'vmapi',
          'memory': '2048',
          'ide2': 'ednfs:iso/ubuntu2404.iso,media=cdrom,size=2708862K',
          'scsi0': 'ednfs:120/vm-120-disk-0.qcow2,iothread=1,size=20G'
         }

a.create_vm('edprox1', **params)
```

Δημιουργία Δίσκων

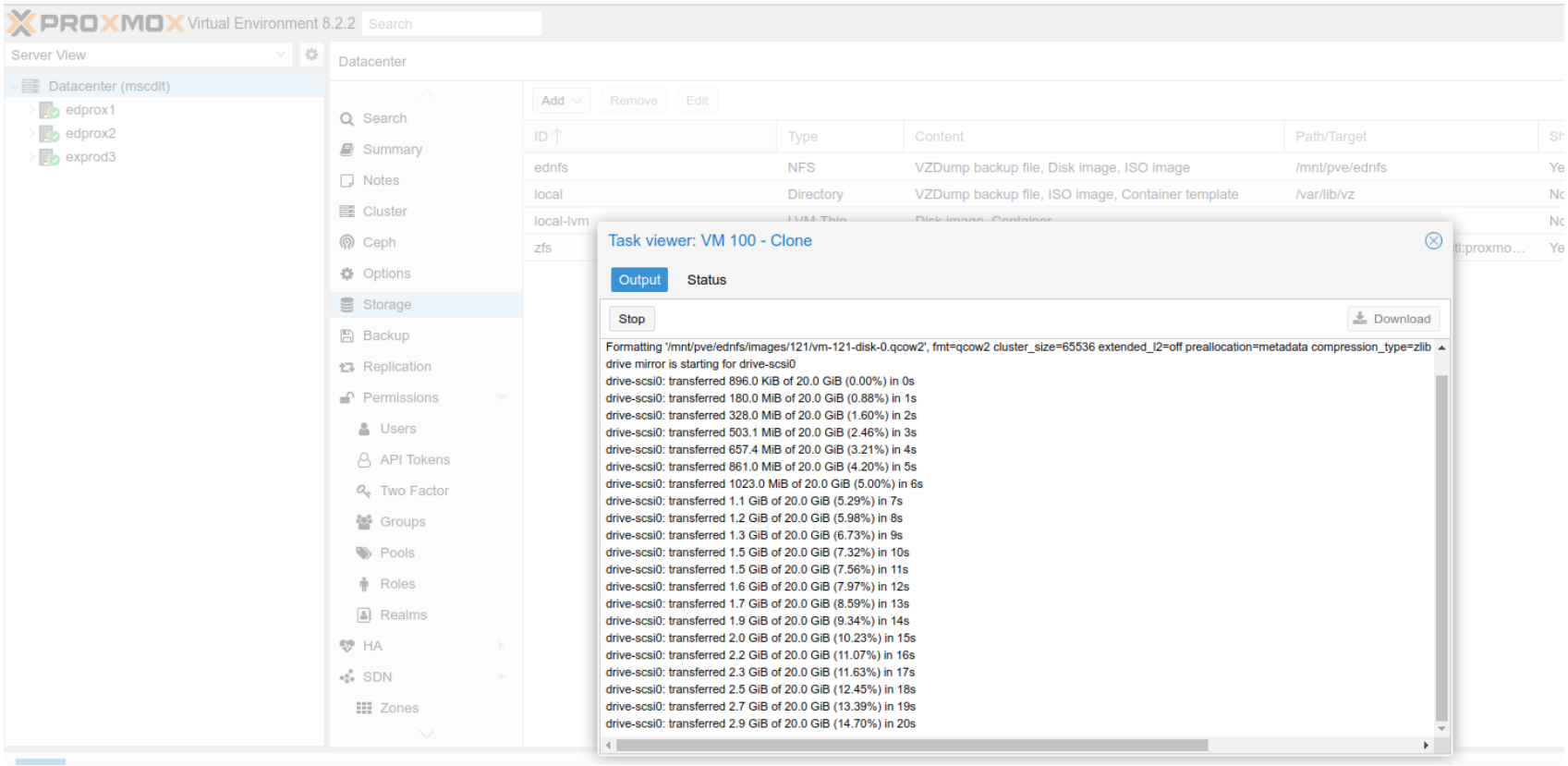


VM cloning

```
    def clone_vm(self, nodeid, vmid, newid):
        suburl = 'nodes/{node}/qemu/{vmid}/clone'.format(node=nodeid,
                                                         vmid=vmid)
        params = {'newid' : newid }
        return self.post(suburl, params=params)
```

```
import requests
from proxmoxapi import api


a = api()
r = a.clone_vm('edprox1', 100, 121)
```

VM cloning


PROXMOX Virtual Environment 8.2.2
Search
Server View
Datacenter
Datacenter (mscdit)
edprox1
edprox2
exprod3
Search
Summary
Notes
Cluster
Ceph
Options
Storage
Backup
Replication
Permissions
Users
API Tokens
Two Factor
Groups
Pools
Roles
Realms
HA
SDN
Zones
Add
Remove
Edit
ID
Type
Content
Path/Target
ednfs
NFS
VZDump backup file, Disk image, ISO image
/mnt/pve/ednfs
local
Directory
VZDump backup file, ISO image, Container template
/var/lib/vz
local-lvm
zfs
Task viewer: VM 100 - Clone
Output
Status
Stop
Download
Formatting '/mnt/pve/ednfs/images/121/vm-121-disk-0.qcow2', fmt=qcow2 cluster_size=65536 extended_l2=off preallocation=metadata compression_type=zlib
drive mirror is starting for drive-scsi0
drive-scsi0: transferred 896.0 KiB of 20.0 GiB (0.00%) in 0s
drive-scsi0: transferred 180.0 MiB of 20.0 GiB (0.88%) in 1s
drive-scsi0: transferred 328.0 MiB of 20.0 GiB (1.60%) in 2s
drive-scsi0: transferred 503.1 MiB of 20.0 GiB (2.46%) in 3s
drive-scsi0: transferred 657.4 MiB of 20.0 GiB (3.21%) in 4s
drive-scsi0: transferred 861.0 MiB of 20.0 GiB (4.20%) in 5s
drive-scsi0: transferred 1023.0 MiB of 20.0 GiB (5.00%) in 6s
drive-scsi0: transferred 1.1 GiB of 20.0 GiB (5.29%) in 7s
drive-scsi0: transferred 1.2 GiB of 20.0 GiB (5.98%) in 8s
drive-scsi0: transferred 1.3 GiB of 20.0 GiB (6.73%) in 9s
drive-scsi0: transferred 1.5 GiB of 20.0 GiB (7.32%) in 10s
drive-scsi0: transferred 1.5 GiB of 20.0 GiB (7.56%) in 11s
drive-scsi0: transferred 1.6 GiB of 20.0 GiB (7.97%) in 12s
drive-scsi0: transferred 1.7 GiB of 20.0 GiB (8.59%) in 13s
drive-scsi0: transferred 1.9 GiB of 20.0 GiB (9.34%) in 14s
drive-scsi0: transferred 2.0 GiB of 20.0 GiB (10.23%) in 15s
drive-scsi0: transferred 2.2 GiB of 20.0 GiB (11.07%) in 16s
drive-scsi0: transferred 2.3 GiB of 20.0 GiB (11.63%) in 17s
drive-scsi0: transferred 2.5 GiB of 20.0 GiB (12.45%) in 18s
drive-scsi0: transferred 2.7 GiB of 20.0 GiB (13.39%) in 19s
drive-scsi0: transferred 2.9 GiB of 20.0 GiB (14.70%) in 20s

VM config

```
class api:

    def __init__(self,
                 secret = SECRET,
                 tokenid = TOKENID,
                 url = URL,
                 verify = False,
                 check_interval = CHECK_INTERVAL,
                 timeout = TIMEOUT):

        self.secret = secret
        self.tokenid = tokenid
        self.url = url
        self.verify = verify
        self.timeout = timeout
        self.check_interval = check_interval

        self.headers = {
            'Authorization' : 'PVEAPIToken=root@pam!%s=%s' %(self. tokenid, self.secre
        }
```

VM config

```
def elapsed(t0):
    t1 = datetime.datetime.now()
    return (t1-t0).total_seconds()
```

```
  def get_task_status(self, nodeid, upid):
      suburl = 'nodes/{node}/tasks/{upid}/status'.format(node=nodeid,
                                                         upid=upid)
      return self.get(suburl)
```

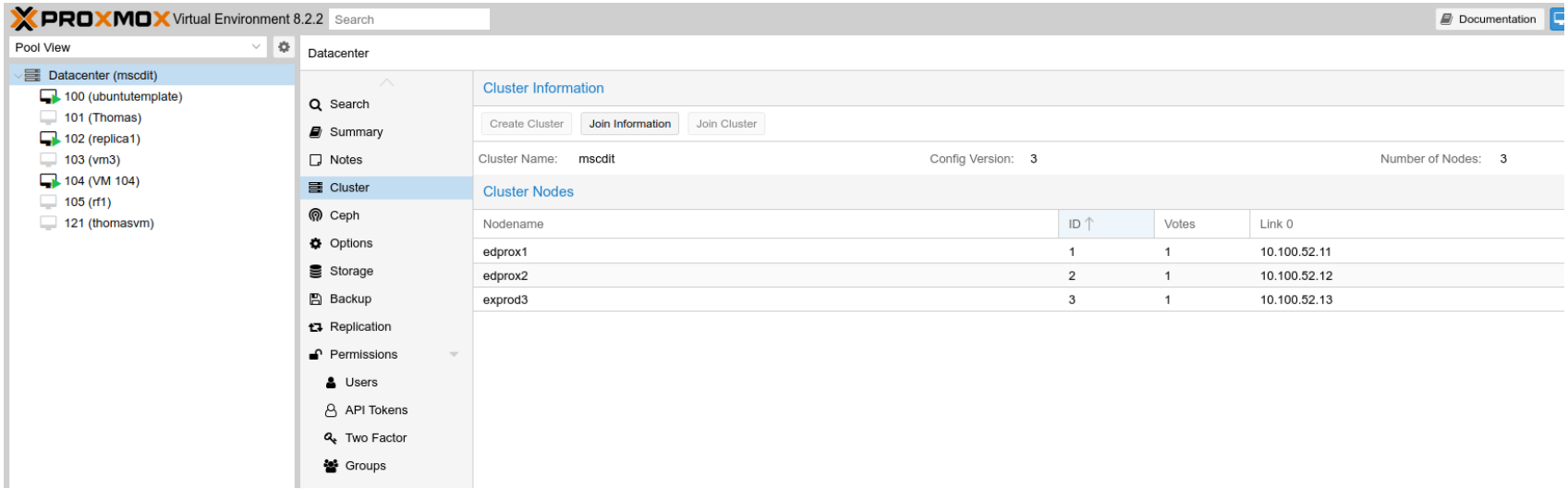
VM config

```
def change_vm_config(self, nodeid, vmid, **params):
    suburl = 'nodes/{node}/qemu/{vmid}/config'.format(node=nodeid,
                                                      vmid=vmid)
    return self.post(suburl, params = params)

def wait_for_task(self, nodeid, upid, timeout = None, echodot=True):
    t0 = datetime.datetime.now()

    if not timeout:
        timeout=self.timeout

    while elapsed(t0) < timeout:
        s = self.get_task_status(nodeid, upid)
        if s['status'] == 'stopped':
            print()
            return elapsed(t0)
        time.sleep(self.check_interval)
        if echodot:
            print('.', end='', flush=True)

    return -1
```

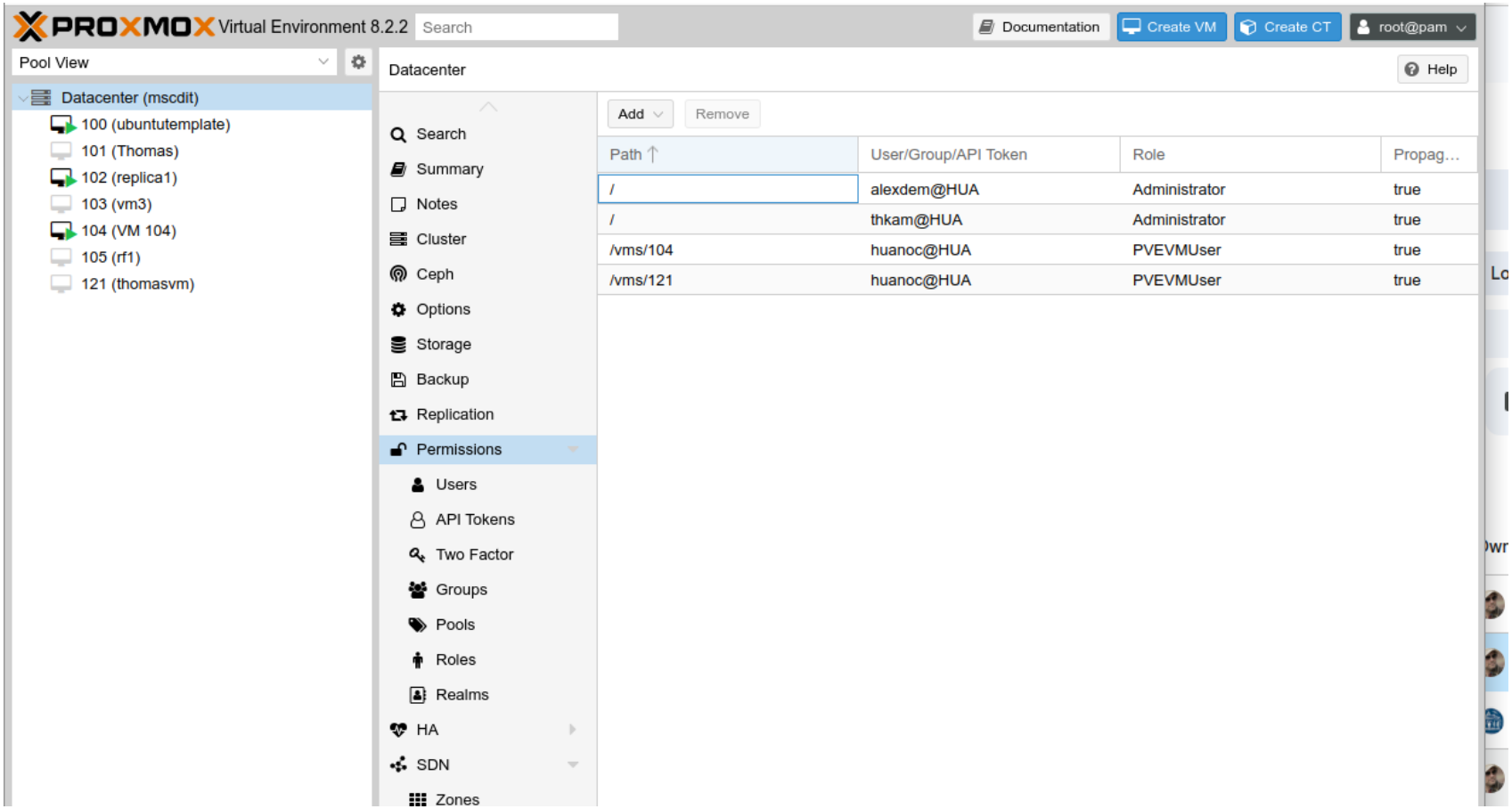
VM config

```
import requests
from proxmoxapi import api


a = api()
upid = a.clone_vm('edprox1', 100, 121)
t = a.wait_for_task('edprox1', upid, timeout = 500)
a.change_vm_config('edprox1', 121, name='thomasvm')
```

VM config


PROXMOX Virtual Environment 8.2.2
Search
Documentation
Pool View
Datacenter
Datacenter (mscdit)
100 (ubuntutemplate)
101 (Thomas)
102 (replica1)
103 (vm3)
104 (VM 104)
105 (rf1)
121 (thomasvm)
Search
Summary
Notes
Cluster
Ceph
Options
Storage
Backup
Replication
Permissions
Users
API Tokens
Two Factor
Groups
Cluster Information
Create Cluster
Join Information
Join Cluster
Cluster Name: mscdit
Config Version: 3
Number of Nodes: 3
Cluster Nodes
Nodename
ID
Votes
Link 0
edprox1 1 1 10.100.52.11
edprox2 2 1 10.100.52.12
exprod3 3 1 10.100.52.13

ACL: Access control lists



ACL: Access control lists

```
def put(self, suburl, params):
    url = self.url + suburl
    print(url)
    r = requests.put(url,
                     json = params,
                     headers = self.headers,
                     verify = self.verify)
    print(r.status_code)
    print(r.text)
    return r.json()['data']
```

ACL: Access control lists

```python
def get_acl(self):
    suburl = 'access/acl'
    return self.get(suburl)

def set_vm_user(self, userid, vmid):
    suburl = 'access/acl'
    path = '/vms/{vmid}'.format(vmid=vmid)
    params = {
        'roles' : ['PVEVMUser'],
        'users' : [ userid ],
        'path' : path,
        'propagate' : 1
    }
    return self.put(suburl, params=params)
```

ACL: Access control lists

```python
import requests
from proxmoxapi import api


a = api()
l = a.get_acl()
print(l)
```

```
thomas@kirklaptop:~/proxmox$ python3 -i getacls.py
[{'path': '/', 'type': 'user', 'ugid': 'alexdem@HUA', 'roleid': 'Administrator', 'propagate': 1}, {'path
: '/', 'type': 'user', 'ugid': 'thkam@HUA', 'propagate': 1, 'roleid': 'Administrator'}, {'type': 'user',
'path': '/vms/104', 'propagate': 1, 'roleid': 'PVEVMUser', 'ugid': 'huanoc@HUA'}, {'path': '/vms/121', '
ype': 'user', 'ugid': 'huanoc@HUA', 'roleid': 'PVEVMUser', 'propagate': 1}]
```

Δημιουργία πολλαπλών VM

- Θέλουμε να διαβάζουμε μία λίστα από χρήστες από ένα αρχείο .csv
- Για κάθε ένα χρήστη θέλουμε αν δεν υπάρχει να τον δημιουργούμε
- Στη συνέχεια να κάνουμε clone το VM 100 και να το ονομάζουμε με ένα συγκεκριμένο όνομα.
- Έπειτα να δίνουμε πρόσβαση στο χρήστη σε αυτό το VM.

Δημιουργία πολλαπλών VM

```python
def create_user(self, userid, **params):
    suburl = 'access/users'
    return self.post(suburl, params = params)

def get_users(self):
    suburl = 'access/users'
    return self.get(suburl)

def get_user(self, userid):
    suburl = 'access/users/{userid}'.format(userid=userid)
    return self.get(suburl)

def user_exists(self, userid):
    users = [ u['userid'] for u in self.get_users() ]
    return userid in users

def get_acl(self):
    suburl = 'access/acl'
    return self.get(suburl)
```

Δημιουργία πολλαπλών VM

```python
from proxmoxapi import api
import csv

CSV_FILE = 'users.csv'
VM_TEMPLATE = 100
NODE = 'edprox1'

def get_list(filename = CSV_FILE):
    with open(filename,'r') as f:
        reader = csv.DictReader(f , delimiter = ';')
        vm_list = list(reader)

    return vm_list

def clone_vm(user, newvmid, node = NODE, vmid = VM_TEMPLATE, name = None):
    a = api()
    t = a.clone_vm(node, vmid, newvmid)
    print(t)
    a.wait_for_task(node, t)
    if not a.user_exists(user):
        a.create_user(user)

    a.set_vm_user(user, newvmid)
    a.change_vm_config(node, newvmid, name=name)

vms = get_list()
for vm in vms:
    print(vm)
    clone_vm(vm['user'], vm['vmid'], name = vm['vm'])
```