ETCD LAB

A distributed, reliable key-value store for the most critical data of a distributed system.
Homepage: https://etcd.io/

Key features:

- Simple: well-defined, user-facing API (gRPC)
- Secure: automatic TLS with optional client cert authentication
- Fast: benchmarked 10,000 writes/sec
- Reliable: properly distributed using Raft

There are two major use cases: concurrency control in the distributed system and application configuration store. For example, CoreOS Container Linux uses etcd to achieve a global semaphore to avoid that all nodes in the cluster rebooting at the same time. Also, Kubernetes use etcd for their configuration store.

During this lab we will be using etcd3 python client.
Homepage: https://pypi.org/project/etcd3/

Etcd credentials are shared on the slack channel: https://join.slack.com/t/ibm-agh-labs/shared_invite/zt-e8xfjgtd-8IDWmn912qPOflbM1yk6~Q

Please copy & paste them into the cell below:

```
etcdCreds = {
  "connection": {
    "cli": {
      "arguments": [
        [
          "--cacert=45dc1d70-521a-11e9-8c84-3e25686eb210",
          "--endpoints=https://afc2bd38-f85c-4387-b5fc-
f4642c7fcf7b.bc28ac43cf10402584b5f01db462d330.databases.appdomain.cloud:311
90",
          "--
user=ibm_cloud_f59f3a7b_7578_4cf8_ba20_6df3b352ab46:230064666d4fe6d81f7c53a
2c364fb60fa079773e8f9adbc163cb0b2e3c58142"
        ]
      ],
      "bin": "etcdctl",
      "certificate": {
        "certificate_base64":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURIVENDQWdXZ0F3SUJBZ0lVVmlhMWZrWE
lsTXhGY2lob3lncWg2Yit6N0pNd0RRWUpLb1pJaHZjTkFRUwKQlFBd0hqRWNNQm9HQTFVRUF3d
1RTVUpOSUVOc2IzVmtJRVJoZEdGaVlYTmxjekFlRncweE9ERXdNVEV4TkRRNApOVEZhRncweU9E
RXdNRGd4TkRRNE5URmFNQjR4SERBYUJnTlZCQU1NRTBsQ1RTQkRiRzkxWkNCRVlYUmhZbUZ6Clp
YTXdnZ0VpTUEwR0NTcUdTSWIzRFFFQkFRVUFBNElCRHdBd2dnRUtBb0lCQVFESkYxMlNjbTJGUm
pQb2N1bmYKbmNkUkFMZDhJRlpiWDhpbDM3MDZ4UEV2b3ZpMTRHNGVIRWZuT1JRY2g3VElPR212R
WxITVllbUtFT3Z3K0VZUApmOXpqU1IxNFVBOXJYeHVaQmgvZDlRa2pjTkw2YmMvbUNUOXpYbmpz
dC9qRzJSbHdmRU1lZnVIQWp1T3c4bkJuCllQeFpiNm1ycVN6T2FtSmpnVVp6c1RMeHRId21yWkx
uOGhlZnhITlBrdGFVMUtFZzNQRkJxaWpDMG9uWFpnOGMKanpZVVVXNkpBOWZZcWJBL1YxMkFsT3
AvUXhKUVVoZlB5YXozN0FEdGpJRkYybkxVMjBicWdyNWhqTjA4SjZQUwpnUk5hNXc2T1N1RGZiZ
```

```
2M4V3Z3THZzbDQvM281akFVSHp2OHJMaWF6d2VPYzlTcDBKd3JHdUJuYTFPYm9mbHU5ClM5SS9B
Z01CQUFHalV6QlJNQjBHQTFVZERnUVdCQlJGejFFckZFSU1CcmFDNndiQjNNMHpuYm1IMmpBZkJ
nTlYKSFNNRUdEQVdnQlJGejFFckZFSU1CcmFDNndiQjNNMHpuYm1IMmpBUEJnTlZIUk1CQWY4RU
JUQURBUUgvTUEwRwpDU3FHU0liM0RRRUJDd1VBQTRJQkFRQ2t4NVJzbk9PMWg0dFJxRzh3R21ub
1EwOHNValpsRXQvc2tmR0pBL2RhClUveEZMMndhNjljTTdNR1VMRitoeXZYSEJScnVOTCtJM1RO
SmtVUEFxMnNhakZqWEtCeVdrb0JYYnRyc2ZKckkKQWhjZnlzN29tdjJmb0pHVGxJY0FybnBCL0p
1bEZITmM1YXQzVk1rSTlidEh3ZUlYNFE1QmdlVlU5cjdDdlArSgpWRjF0YWxSUVpKandyeVhsWG
JvQ0c0MTU2TUtwTDIwMUwyV1dqazBydlBVWnRKcjhmTmd6M24wb0x5MFZ0Zm93Ck1yUFh4THk5T
lBqOGlzT3V0ckxEMjlJWTJBMFY0UmxjSXhTMEw3c1ZPeTB6RDZwbXpNTVFNRC81aWZ1SVg2YnEK
bEplZzV4akt2TytwbElLTWhPU1F5dTRUME1NeTZmY2t3TVpPK0liR3JDZHIKLS0tLS1FTkQgQ0V
SVElGSUNBVEUtLS0tLQo=",
        "name": "45dc1d70-521a-11e9-8c84-3e25686eb210"
      },
      "composed": [
        "ETCDCTL_API=3 etcdctl --cacert=45dc1d70-521a-11e9-8c84-
3e25686eb210 --endpoints=https://afc2bd38-f85c-4387-b5fc-
f4642c7fcf7b.bc28ac43cf10402584b5f01db462d330.databases.appdomain.cloud:311
90 --
user=ibm_cloud_f59f3a7b_7578_4cf8_ba20_6df3b352ab46:230064666d4fe6d81f7c53a
2c364fb60fa079773e8f9adbc163cb0b2e3c58142"
      ],
      "environment": {
        "ETCDCTL_API": "3"
      },
      "type": "cli"
    },
    "grpc": {
      "authentication": {
        "method": "direct",
        "password":
"230064666d4fe6d81f7c53a2c364fb60fa079773e8f9adbc163cb0b2e3c58142",
        "username": "ibm_cloud_f59f3a7b_7578_4cf8_ba20_6df3b352ab46"
      },
      "certificate": {
        "certificate_base64":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURIVENDQWdXZ0F3SUJBZ0lVVmlhMWZrWE
lsTXhGY2lob3lncWg2Yit6N0pNd0RRWUpLb1pJaHZjTkFRUwKQlFBd0hqRWNNQm9HQTFVRUF3d
1RTVUpOSUVOc2IzVmtJRVJoZEdGaVlYTmxjekFlRncweE9ERXdNVEV4TkRRNApOVEZhRncweU9E
RXdNRGd4TkRRNE5URmFNQjR4SERBYUJnTlZCQU1NRTBsQ1RTQkRiRzkxWkNCRVlYUmhZbUZ6Clp
YTXdnZ0VpTUEwR0NTcUdTSWIzRFFFQkFRVUFBNElCRHdBd2dnRUtBb0lCQVFESkYxMlNjbTJGUm
pQb2N1bmYKbmNkUkFMZDhJRlpiWDhpbDM3MDZ4UEV2b3ZpMTRHNGVIRWZuT1JRY2g3VElPR212R
WxITVllbUtFT3Z3K0VZUApmOXpqU1IxNFVBOXJYeHVaQmgvZDlRa2pjTkw2YmMvbUNUOXpYbmpz
dC9qRzJSbHdmRU1lZnVIQWp1T3c4bkJuCllQeFpiNm1ycVN6T2FtSmpnVVp6c1RMeHRId21yWkx
uOGhlZnhITlBrdGFVMUtFZzNQRkJxaWpDMG9uWFpnOGMKanpZVVVXNkpBOWZZcWJBL1YxMkFsT3
AvUXhKUVVoZlB5YXozN0FEdGpJRkYybkxVMjBicWdyNWhqTjA4SjZQUwpnUk5hNXc2T1N1RGZiZ
2M4V3Z3THZzbDQvM281akFVSHp2OHJMaWF6d2VPYzlTcDBKd3JHdUJuYTFPYm9mbHU5ClM5SS9B
Z01CQUFHalV6QlJNQjBHQTFVZERnUVdCQlJGejFFckZFSU1CcmFDNndiQjNNMHpuYm1IMmpBZkJ
nTlYKSFNNRUdEQVdnQlJGejFFckZFSU1CcmFDNndiQjNNMHpuYm1IMmpBUEJnTlZIUk1CQWY4RU
JUQURBUUgvTUEwRwpDU3FHU0liM0RRRUJDd1VBQTRJQkFRQ2t4NVJzbk9PMWg0dFJxRzh3R21ub
1EwOHNValpsRXQvc2tmR0pBL2RhClUveEZMMndhNjljTTdNR1VMRitoeXZYSEJScnVOTCtJM1RO
SmtVUEFxMnNhakZqWEtCeVdrb0JYYnRyc2ZKckkKQWhjZnlzN29tdjJmb0pHVGxJY0FybnBCL0p
1bEZITmM1YXQzVk1rSTlidEh3ZUlYNFE1QmdlVlU5cjdDdlArSgpWRjF0YWxSUVpKandyeVhsWG
JvQ0c0MTU2TUtwTDIwMUwyV1dqazBydlBVWnRKcjhmTmd6M24wb0x5MFZ0Zm93Ck1yUFh4THk5T
lBqOGlzT3V0ckxEMjlJWTJBMFY0UmxjSXhTMEw3c1ZPeTB6RDZwbXpNTVFNRC81aWZ1SVg2YnEK
```

```
bEplZzV4akt2TytwbElLTWhPU1F5dTRUME1NeTZmY2t3TVpPK0liR3JDZHIKLS0tLS1FTkQgQ0V
SVElGSUNBVEUtLS0tLQo=",
        "name": "45dc1d70-521a-11e9-8c84-3e25686eb210"
      },
      "composed": [

"https://ibm_cloud_f59f3a7b_7578_4cf8_ba20_6df3b352ab46:230064666d4fe6d81f7
c53a2c364fb60fa079773e8f9adbc163cb0b2e3c58142@afc2bd38-f85c-4387-b5fc-
f4642c7fcf7b.bc28ac43cf10402584b5f01db462d330.databases.appdomain.cloud:311
90"
      ],
      "hosts": [
        {
          "hostname": "afc2bd38-f85c-4387-b5fc-
f4642c7fcf7b.bc28ac43cf10402584b5f01db462d330.databases.appdomain.cloud",
          "port": 31190
        }
      ],
      "path": "",
      "query_options": {},
      "scheme": "https",
      "type": "uri"
    }
  },
  "instance_administration_api": {
    "deployment_id": "crn:v1:bluemix:public:databases-for-etcd:eu-
de:a/a34b4e9ea7ab66770e048caf83277971:afc2bd38-f85c-4387-b5fc-
f4642c7fcf7b::",
    "instance_id": "crn:v1:bluemix:public:databases-for-etcd:eu-
de:a/a34b4e9ea7ab66770e048caf83277971:afc2bd38-f85c-4387-b5fc-
f4642c7fcf7b::",
    "root": "https://api.eu-de.databases.cloud.ibm.com/v4/ibm"
  }
}  # copy and paste etcd credentials provided on the Slack channel here
```

```
!pip install etcd3
```

```
Collecting etcd3
[?25l  Downloading
https://files.pythonhosted.org/packages/9c/eb/6d1ef4d6a3e8b74e45c502cbd3ea6
c5c6c786d003829db9369c2530f5e3f/etcd3-0.12.0.tar.gz (63kB)
[K     |████████████████████████████████| 71kB 7.7MB/s eta 0:00:011
[?25hCollecting grpcio>=1.27.1 (from etcd3)
[?25l  Downloading
https://files.pythonhosted.org/packages/cd/04/2b67f0a3645481235d5547891fd0e
45e384f1ae5676788f24a7c8735b4e9/grpcio-1.29.0-cp36-cp36m-
manylinux2010_x86_64.whl (3.0MB)
[K     |████████████████████████████████| 3.0MB 10.1MB/s eta 0:00:01
[?25hRequirement already satisfied: protobuf>=3.6.1 in
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages (from etcd3) (3.6.1)
Requirement already satisfied: six>=1.12.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from etcd3) (1.12.0)
Collecting tenacity>=6.1.0 (from etcd3)
  Downloading
https://files.pythonhosted.org/packages/b5/05/ff089032442058bd3386f9cd991cd
88ccac81dca1494d78751621ee35e62/tenacity-6.2.0-py2.py3-none-any.whl
Requirement already satisfied: setuptools in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from protobuf>=3.6.1-
>etcd3) (40.8.0)
Building wheels for collected packages: etcd3
  Building wheel for etcd3 (setup.py) ... □[?25ldone
□[?25h  Stored in directory:
/home/dsxuser/.cache/pip/wheels/a8/36/b5/cabe849e7cb6e1c273ca48946b825d6f6f
5271017c8497d7ea
Successfully built etcd3
□[31mERROR: tensorflow 1.13.1 requires tensorboard<1.14.0,>=1.13.0, which
is not installed.□[0m
Installing collected packages: grpcio, tenacity, etcd3
  Found existing installation: grpcio 1.16.1
    Uninstalling grpcio-1.16.1:
      Successfully uninstalled grpcio-1.16.1
Successfully installed etcd3-0.12.0 grpcio-1.29.0 tenacity-6.2.0
```

How to connect to etcd using certyficate (part 1: prepare file with certificate)

```
import base64
import tempfile

etcdHost = etcdCreds["connection"]["grpc"]["hosts"][0]["hostname"]
etcdPort = etcdCreds["connection"]["grpc"]["hosts"][0]["port"]
etcdUser = etcdCreds["connection"]["grpc"]["authentication"]["username"]
etcdPasswd = etcdCreds["connection"]["grpc"]["authentication"]["password"]
etcdCertBase64 = etcdCreds["connection"]["grpc"]["certificate"]
["certificate_base64"]

etcdCertDecoded = base64.b64decode(etcdCertBase64)
etcdCertPath = "{}/{}.cert".format(tempfile.gettempdir(), etcdUser)

with open(etcdCertPath, 'wb') as f:
    f.write(etcdCertDecoded)

print(etcdCertPath)
```

```
/home/dsxuser/.tmp/ibm_cloud_f59f3a7b_7578_4cf8_ba20_6df3b352ab46.cert
```

Short Lab description

During the lab we will simulate system that keeps track of logged users

- All users will be stored under parent key (path): /logged_users
- Each user will be represented by key value pair
    - key /logged_users/name_of_the_user
    - value hostname of the machine (e.g. name_of_the_user-hostname)

How to connect to etcd using certyficate (part 2: create client)

```
import etcd3

etcd = etcd3.client(
    host=etcdHost,
    port=etcdPort,
    user=etcdUser,
    password=etcdPasswd,
    ca_cert=etcdCertPath
)

cfgRoot='Krzysztof-Hardek/logged_users'
```

Task 1 : Fetch username and hostname

define two variables

- username name of the logged user (tip: use getpass library)
- hostname hostname of your mcomputer (tip: use socket library)

```
import getpass
import socket

username = 'Krzysztof-Hardek'
hostname = socket.gethostname()

userKey='{}/{}'.format(cfgRoot, username)
userKey, '->', hostname
```

```
('Krzysztof-Hardek/logged_users/Krzysztof-Hardek',
 '->',
 'notebook-conda2py368ee4a748cf994ce5aadc2e5f85017d0a-6b449dsvb89')
```

Task 2 : Register number of users

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

for all names in table fixedUsers register the appropriate key value pairs

```
registered_users = etcd.get_prefix(cfgRoot)

for value, meta in registered_users:
    print(meta.key, value)
```

```
b'Krzysztof-Hardek/logged_users/Adam' b'hostname-Adam'
b'Krzysztof-Hardek/logged_users/Borys' b'hostname-Borys'
b'Krzysztof-Hardek/logged_users/Cezary' b'hostname-Cezary'
b'Krzysztof-Hardek/logged_users/Damian' b'hostname-Damian'
b'Krzysztof-Hardek/logged_users/Emil' b'hostname-Emil'
b'Krzysztof-Hardek/logged_users/Filip' b'hostname-Filip'
b'Krzysztof-Hardek/logged_users/Gustaw' b'hostname-Gustaw'
b'Krzysztof-Hardek/logged_users/Henryk' b'hostname-Henryk'
b'Krzysztof-Hardek/logged_users/Ignacy' b'hostname-Ignacy'
b'Krzysztof-Hardek/logged_users/Jacek' b'hostname-Jacek'
b'Krzysztof-Hardek/logged_users/Kamil' b'hostname-Kamil'
b'Krzysztof-Hardek/logged_users/Leon' b'hostname-Leon'
b'Krzysztof-Hardek/logged_users/Marek' b'hostname-Marek'
b'Krzysztof-Hardek/logged_users/Norbert' b'hostname-Norbert'
b'Krzysztof-Hardek/logged_users/Oskar' b'hostname-Oskar'
b'Krzysztof-Hardek/logged_users/Patryk' b'hostname-Patryk'
b'Krzysztof-Hardek/logged_users/Rafa\xc5\x82' b'hostname-Rafa\xc5\x82'
b'Krzysztof-Hardek/logged_users/Stefan' b'hostname-Stefan'
b'Krzysztof-Hardek/logged_users/Tadeusz' b'hostname-Tadeusz'
```

```
fixedUsers = [
    'Adam',
    'Borys',
    'Cezary',
    'Damian',
    'Emil',
    'Filip',
    'Gustaw',
    'Henryk',
    'Ignacy',
    'Jacek',
    'Kamil',
    'Leon',
    'Marek',
    'Norbert',
    'Oskar',
    'Patryk',
    'Rafał',
    'Stefan',
    'Tadeusz'
]

for user in fixedUsers:
```

```
    etcd.put(f'{cfgRoot}/{user}', user)
```

Task 3: List all users

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

List all registered user (tip: use common prefix)

```
registered_users = etcd.get_prefix(f'{cfgRoot}')

for value, meta in registered_users:
    print(meta.key, value)
```

```
b'Krzysztof-Hardek/logged_users/Adam' b'Adam'
b'Krzysztof-Hardek/logged_users/Borys' b'Borys'
b'Krzysztof-Hardek/logged_users/Cezary' b'Cezary'
b'Krzysztof-Hardek/logged_users/Damian' b'Damian'
b'Krzysztof-Hardek/logged_users/Emil' b'Emil'
b'Krzysztof-Hardek/logged_users/Filip' b'Filip'
b'Krzysztof-Hardek/logged_users/Gustaw' b'Gustaw'
b'Krzysztof-Hardek/logged_users/Henryk' b'Henryk'
b'Krzysztof-Hardek/logged_users/Ignacy' b'Ignacy'
b'Krzysztof-Hardek/logged_users/Jacek' b'Jacek'
b'Krzysztof-Hardek/logged_users/Kamil' b'Kamil'
b'Krzysztof-Hardek/logged_users/Leon' b'Leon'
b'Krzysztof-Hardek/logged_users/Marek' b'Marek'
b'Krzysztof-Hardek/logged_users/Norbert' b'Norbert'
b'Krzysztof-Hardek/logged_users/Oskar' b'Oskar'
b'Krzysztof-Hardek/logged_users/Patryk' b'Patryk'
b'Krzysztof-Hardek/logged_users/Rafa\xc5\x82' b'Rafa\xc5\x82'
b'Krzysztof-Hardek/logged_users/Stefan' b'Stefan'
b'Krzysztof-Hardek/logged_users/Tadeusz' b'Tadeusz'
```

Task 4 : Same as Task2, but use transaction

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

for all names in table fixedUsers register the appropriate key value pairs, use transaction to make it a single request
(Have you noticed any difference in execution time?)

```
success  = [
    etcd.transactions.put(f'{cfgRoot}/{user}', f'hostname-{user}') for user
in fixedUsers
]
```

```
etcd.transaction(
    compare=[
        etcd.transactions.version(cfgRoot) == 1 # intentional failure,
should be 0
    ],
    success=success,
    failure=[
        etcd.transactions.put(f'{cfgRoot}/error', 'condition failed')
    ]
)
```

```
(False, [response_put {
    header {
      revision: 360989
    }
  }])
```

Task 5 : Get single key (e.g. status of transaction)

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

Check the key you are modifying in on-failure handler in previous task

```
etcd.get(f'{cfgRoot}/error')
```

```
(b'condition failed', <etcd3.client.KVMetadata at 0x7fda55a1bc88>)
```

Task 6 : Get range of Keys (Emil -> Oskar)

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

- Get range of keys
- Is it inclusive / exclusive? **exclusive**
- Sort the resposne descending
- Sort the resposne descending by value not by key

```
result = etcd.get_range(f'{cfgRoot}/Emil', f'{cfgRoot}/Oskar')

for value, meta in result:
    print(meta.key, value)

print('------------------------------------------------------')

result = etcd.get_range(f'{cfgRoot}/Emil', f'{cfgRoot}/Oskar',
```

```
sort_order='descend')

for value, meta in result:
    print(meta.key, value)

print('-----------------------------------------------------------')

result = etcd.get_range(f'{cfgRoot}/Emil', f'{cfgRoot}/Oskar',
sort_order='descend', sort_target='value')

for value, meta in result:
    print(meta.key, value)
```

```
b'Krzysztof-Hardek/logged_users/Emil' b'hostname-Emil'
b'Krzysztof-Hardek/logged_users/Filip' b'hostname-Filip'
b'Krzysztof-Hardek/logged_users/Gustaw' b'hostname-Gustaw'
b'Krzysztof-Hardek/logged_users/Henryk' b'hostname-Henryk'
b'Krzysztof-Hardek/logged_users/Ignacy' b'hostname-Ignacy'
b'Krzysztof-Hardek/logged_users/Jacek' b'hostname-Jacek'
b'Krzysztof-Hardek/logged_users/Kamil' b'hostname-Kamil'
b'Krzysztof-Hardek/logged_users/Leon' b'hostname-Leon'
b'Krzysztof-Hardek/logged_users/Marek' b'hostname-Marek'
b'Krzysztof-Hardek/logged_users/Norbert' b'hostname-Norbert'
-----------------------------------------------------------
b'Krzysztof-Hardek/logged_users/Norbert' b'hostname-Norbert'
b'Krzysztof-Hardek/logged_users/Marek' b'hostname-Marek'
b'Krzysztof-Hardek/logged_users/Leon' b'hostname-Leon'
b'Krzysztof-Hardek/logged_users/Kamil' b'hostname-Kamil'
b'Krzysztof-Hardek/logged_users/Jacek' b'hostname-Jacek'
b'Krzysztof-Hardek/logged_users/Ignacy' b'hostname-Ignacy'
b'Krzysztof-Hardek/logged_users/Henryk' b'hostname-Henryk'
b'Krzysztof-Hardek/logged_users/Gustaw' b'hostname-Gustaw'
b'Krzysztof-Hardek/logged_users/Filip' b'hostname-Filip'
b'Krzysztof-Hardek/logged_users/Emil' b'hostname-Emil'
-----------------------------------------------------------
b'Krzysztof-Hardek/logged_users/Norbert' b'hostname-Norbert'
b'Krzysztof-Hardek/logged_users/Marek' b'hostname-Marek'
b'Krzysztof-Hardek/logged_users/Leon' b'hostname-Leon'
b'Krzysztof-Hardek/logged_users/Kamil' b'hostname-Kamil'
b'Krzysztof-Hardek/logged_users/Jacek' b'hostname-Jacek'
b'Krzysztof-Hardek/logged_users/Ignacy' b'hostname-Ignacy'
b'Krzysztof-Hardek/logged_users/Henryk' b'hostname-Henryk'
b'Krzysztof-Hardek/logged_users/Gustaw' b'hostname-Gustaw'
b'Krzysztof-Hardek/logged_users/Filip' b'hostname-Filip'
b'Krzysztof-Hardek/logged_users/Emil' b'hostname-Emil'
```

Task 7: Atomic Replace

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

Do it a few times, check if value has been replaced depending on condition. **it has**

```
etcd.replace(f'{cfgRoot}/Norbert', 'hostname-Norbert', 'hostname-Norbert2')

r = etcd.get(f'{cfgRoot}/Norbert')
print(r)

etcd.replace(f'{cfgRoot}/Norbert', 'hostname-Norbert', 'hostname-Norbert3')

r = etcd.get(f'{cfgRoot}/Norbert')
print(r)

etcd.replace(f'{cfgRoot}/Norbert', 'hostname-Norbert2', 'hostname-Norbert')

r = etcd.get(f'{cfgRoot}/Norbert')
print(r)
```

```
(b'hostname-Norbert2', <etcd3.client.KVMetadata object at 0x7fda54162588>)
(b'hostname-Norbert2', <etcd3.client.KVMetadata object at 0x7fda541625f8>)
(b'hostname-Norbert', <etcd3.client.KVMetadata object at 0x7fda541625c0>)
```

Task 8 : Create lease - use it to create expiring key

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

You can create a key that will be for limited time add user that will expire after a few seconds

Tip: Use lease

```
import time

lease = etcd.lease(ttl=5)

etcd.put(f'{cfgRoot}/Nowy', 'hostname-Nowy', lease=lease)

print(etcd.get(f'{cfgRoot}/Nowy'))

time.sleep(6)

print(etcd.get(f'{cfgRoot}/Nowy'))
```

```
(b'hostname-Nowy', <etcd3.client.KVMetadata object at 0x7fda54162358>)
(None, None)
```

Task 9 : Create key that will expire after you close the connection to etcd

Tip: use threading library to refresh your lease

```
import threading

def refresh_lease(lease):

    while lease.refresh():
        global stop_thread
        if stop_thread:
            break

        time.sleep(2)

lease = etcd.lease(ttl=4)

stop_thread = False
refresh_thread = threading.Thread(target=refresh_lease, args=(lease,))

etcd.put(f'{cfgRoot}/Nowy4', 'hostname-Nowy4', lease=lease)

refresh_thread.start()

print(etcd.get(f'{cfgRoot}/Nowy4'))

time.sleep(5)
print(etcd.get(f'{cfgRoot}/Nowy4'))

time.sleep(5)
print(etcd.get(f'{cfgRoot}/Nowy4'))

stop_thread = True
refresh_thread.join()

time.sleep(5)
print(etcd.get(f'{cfgRoot}/Nowy4'))
```

```
(b'hostname-Nowy4', <etcd3.client.KVMetadata object at 0x7fda540c11d0>)
(b'hostname-Nowy4', <etcd3.client.KVMetadata object at 0x7fda540c11d0>)
(b'hostname-Nowy4', <etcd3.client.KVMetadata object at 0x7fda540c11d0>)
(None, None)
```

Task 10: Use lock to protect section of code

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

```
import time

with etcd.lock('lock-1', ttl=10) as lock:
```

```
    print('asd')
    print(lock.is_acquired())
    lock.acquire()
    print('cde')
    print('asdasd')
    lock.release()
```

```
asd
True
cde
asdasd
```

Task 11: Watch key

etcd3 api: https://python-etcd3.readthedocs.io/en/latest/usage.html

This cell will lock this notebook on waiting
After running it create a new notebook and try to add new user

```
def etcd_call(cb):
    print(cb)

etcd.add_watch_callback(key='/lease/watch/friends', callback=etcd_call)
```

```

```

```
notebook-conda2py368ee4a748cf994ce5aadc2e5f85017d0a-6b449dsvb89
```

```

```