CSE 6331   Cloud Computing
Spring 2017, © DL, UTA, 2017

Programming Assignment 1
Introduction to Cloud Storage
Due: January 25, In Blackboard

Description:

One of the most common uses of “Clouds”, is shared or backup storage.
A number of services provide free (limited) storage, and several provide an easy to use, comfortable interface, such as a folder (subdirectory) on your desktop where you may drop file to be automatically backed up (to the cloud service and retrieved – or even shared between users), or web-based interface.
Several of these services: Dropbox, Sugarsync, Skydrive, Googledrive, and iCloud offer free storage and most importantly – an API where you may program utilities to use these storage services.
You will create a utility to provide “Storage as a Service” which will:
Securely store and retrieve files to a cloud service provider.
You should offer (at least) these services to a user:

Store a copy of a local file, encrypted, on your (Cloud) service.

Retrieve (and decrypt) a remote file

List all files on your service

Delete a remote file

Limit the maximum sizes of remote files to 1 MB, or less, each

and no more than 10 MB, in total.

Provide an easy to use, “local” interface (web, menu, or commands)

(Handle cases such as file existing on service, file too large, and similar)

Please, submit through Blackboard. If necessary, email ONLY to the class account.
All work must be your own, or from a group.
You must submit this lab, working (or partially) by the due date. The subject should clearly state the lab number.
You may (optionally) demonstrate this lab, working (or partially) to the GTA before the due date.
Your program should be well commented and documented, make sure the first few lines of your program contain your name, this course number, and the lab number.
Your comments should reflect your design and issues in your implementation.
Your design and implementation should address error conditions.

All work must be your own (or group), you may reference web sites, books, or my code but You MUST site the references.

Cloud Assignment 1 – IBM Bluemix data store, retrieve and basic cryptography

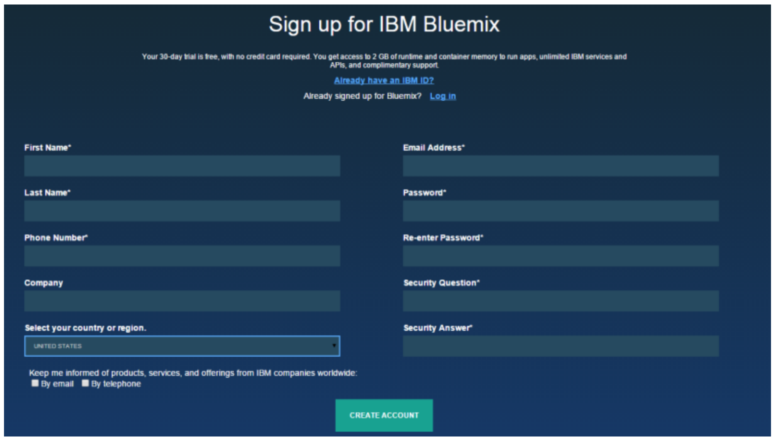
The following is one possible method, this illustrates the basic steps/issues, but there are tutorials on the IBM

Bluemix site that are generally of interest and provide much more detailed information.

You need: Python 2.x (https://www.python.org/), IBM bluemix account, pip installed on your computer.

Prerequisites to install:
```
pip install python-swiftclient
pip install python-keystoneclient
pip install urllib3 certifipyopenssl
```

1. Create an IBM Bluemix account at the www.**ibm**.com/**Bluemix**
   Then click on the Signup button and then register at the following website
   https://console.ng.bluemix.net/registration/



Sign up for IBM Bluemix

Your 30-day trial is free, with no credit card required. You get access to 2 GB of runtime and container memory to run apps, unlimited IBM services and APIs, and complimentary support.

Already have an IBM ID?

Already signed up for Bluemix? Log in

First Name*

Email Address*

Last Name*

Password*

Phone Number*

Re-enter Password*

Company

Security Question*

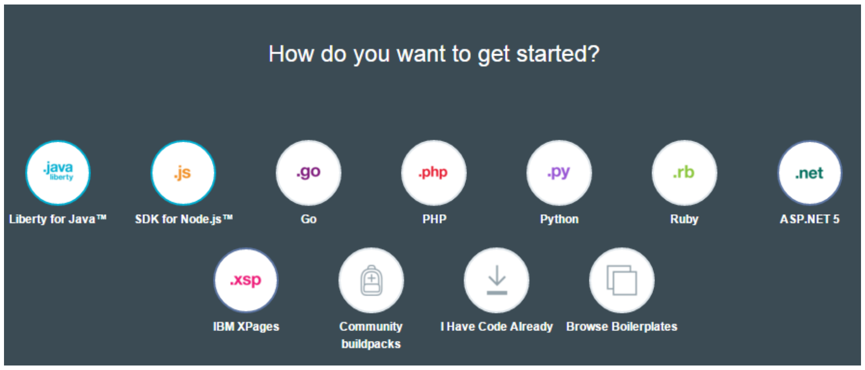
Select your country or region.
UNITED STATES

Security Answer*

Keep me informed of products, services, and offerings from IBM companies worldwide:
By email By telephone

CREATE ACCOUNT

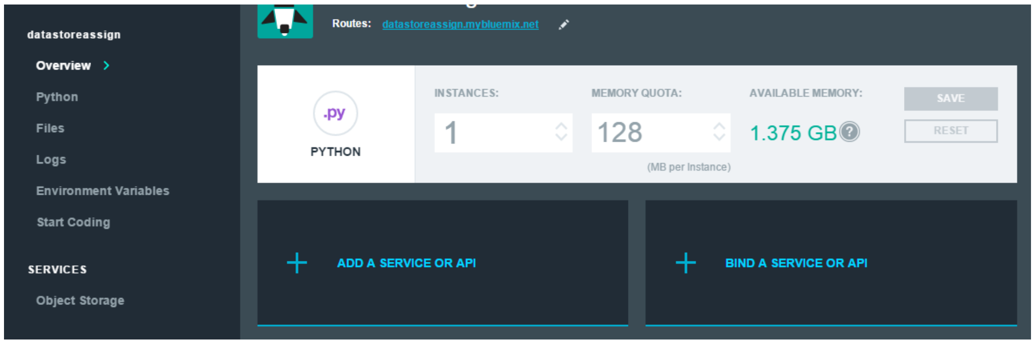
2. Click on dashboard then click on Cloud Foundry App then choose Web app.
   Now choose the starting point as python on this page.



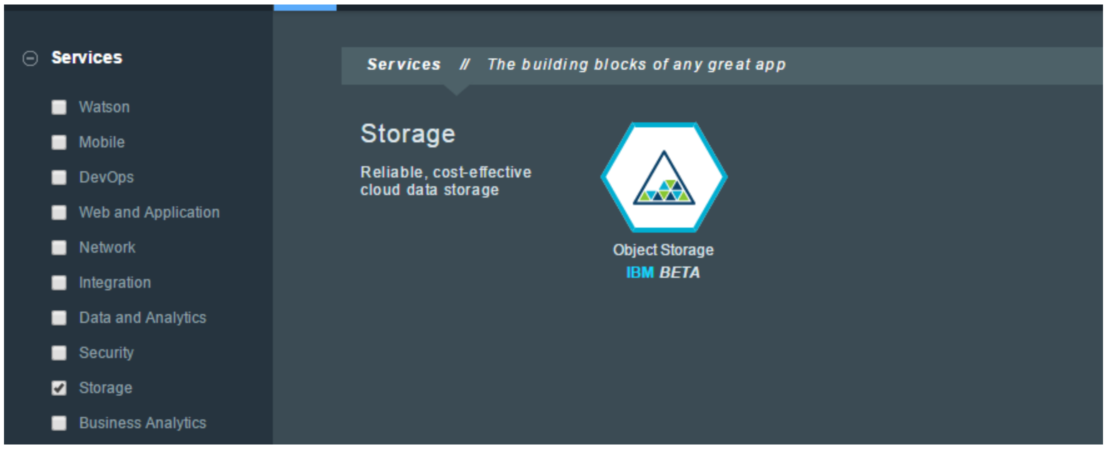
How do you want to get started?

3. Provide app name and host in the prompt. Click on the application title to go to the overview page.
4. Please ensure Cloud Foundry command line interface tool is installed. Follow the instruction in the link below.
   https://www.ng.bluemix.net/docs/starters/install_cli.html

5. Click on the overview of the app and then click on Add a service or API

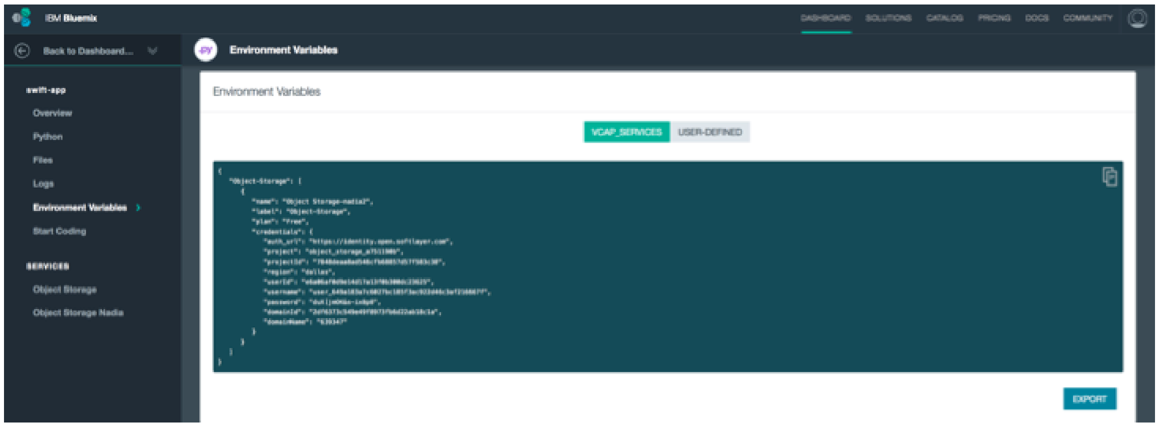


6. Under services -> select storage and then click on object storage IBM BETA



7. Select plan as free and then click on create with the default values.
   To learn more about data store refer to this page
   https://www.ng.bluemix.net/docs/services/ObjectStorage/index.html

8. Connecting to V3 and authenticating through VCAP credentials.
   On the left side panel select 'EnvironmentVariables' . These are credentials associated with your Object Storage Instance, which your bound python application will utilize to connect.



```
{
  "Object-Storage": [
    {
      "name": "Object Storage-nadia2",
      "label": "Object-Storage",
      "plan": "Free",
      "credentials": {
        "auth_url": "https://identity.open.softlayer.com",
        "project": "object_storage_[illegible]",
        "projectId": "[illegible]",
        "region": "dallas",
        "userId": "[illegible]",
        "username": "user_[illegible]",
        "password": "[illegible]",
        "domainId": "[illegible]",
        "domainName": "[illegible]"
      }
    }
  ]
}
```

Connect to your app using the following in the python code
In the below code append auth_url + '/V3'

```
#IBM bluemix app Environment variables of the app.
auth_url =  #authorization URL[redacted]
password = #password
project_id = #project id
user_id = #user id
region_name = 'dallas' #region name
conn = swiftclient.Connection(key=password,
authurl=auth_url,
auth_version='3',
os_options={"project_id": project_id,
            "user_id": user_id,
            "region_name": region_name})
```

9. Swift is an OpenStack object store project that offers cloud storage software so that you can store and retrieve data with a simple API.
   Using pip install python-swiftclient – This is a python client for the Swift API. There's a Python API (the swiftclient module), and a command-line script (swift).
   python-keystoneclient – To authenticate from your local machine to Bluemix

This is a client for the OpenStack Identity API, implemented by the Keystone team; it contains a Python API (thekeystoneclient module) for OpenStack's Identity Service.

10. Now you (locally) can create a new container to the datastore using
    conn.put_container(<container name>)

    Storage containers are similar to folders (directories) which hold the objects. (buckets in Amazon Web Services)

    You can put the files (upload) in the container using the following code

    ```
    conn.put_object(container_name,
    file_name,
    contents= <file content>,
    content_type= <file type>)
    ```

    Similarly you can download the file with the following local code.
    conn.get_object(<container name>,<file name>)

    You can list objects in a container, and print out each object name, the file size, and last modified date using -

    ```
    for container in conn.get_account()[1]:
       for data in conn.get_container(container['name'])[1]:
          print 'object: {0}\t size: {1}\t date: {2}'.format(data['name'], data['bytes'], data['last_modified'])
    ```

    Print the contents of the conn.get_account() to see what this function returns.

    You can delete the object from a container using
    conn.delete_object(<container_name>, <file_name>)

    You can get more examples at the following:
    https://www.ibm.com/developerworks/community/blogs/

11. python-gnupg is a Python package for encrypting and decrypting strings or files using GNU Privacy Guard (GnuPG or GPG). GPG is an open source alternative to Pretty Good Privacy (PGP). A popular use of GPG and PGP is encrypting email. For more information, see the python-gnupg documentation.

    Read more about GnuPG at https://www.gnupg.org/

    This creates a GPG key. This also creates the gpghome directory if it does not exist.

    ```
    #create an instance of gnupg and pass homedirectory to store public
    # and private keyring files as well as trust database.
    gpg = gnupg.GPG(gnupghome='Path to the gnupg/.gnupg')

    #generate RSA keys of 1024 length and Passphrase which is secret key.
    input_data = gpg.gen_key_input(key_type="RSA", key_length=1024,
        passphrase='xxxxx')
    key = gpg.gen_key(input_data)
    ```

    ```
    #Open the file in the selected path and encrypt the data using public key
    with open(outFile,'rb') as f:
         status = gpg.encrypt_file(f)
    ```

    ```
    #decrypt the file data that was downloaded from the IBM Bluemix
    decrypted_data = gpg.decrypt(fdata, passphrase='my passphrase')
    ```

12. Combine steps 10 and 11 to encrypt the file and then upload on to IBM Bluemix and also allow downloading the file and then decrypting.