



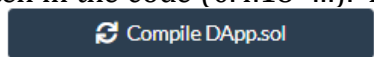



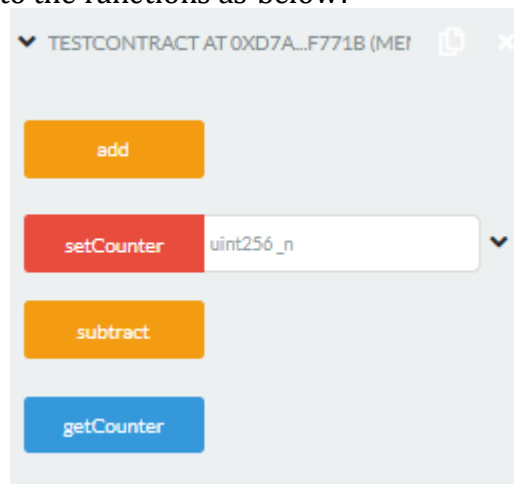


Lab 6

Create Simple DApp and Pub/sub System

[Create Simple DApp]

1. In this lab, we will test a simple DApp using Remix.
 - a. Download DApp.sol
 - b. Access to Remix <https://remix.ethereum.org>
 - c. Add Workspaces in the FILE EXPLORERS on the screen. (Click  in the left menu if you cannot see it.)
 - d. Select  and upload DApp.sol to the folder (Select  to upload the file.)
 - e. Click  in the left menu. Select the compiler version as the same version written in the code (0.4.18+...). Then, compile the program by clicking  button.
2. The program basically has four functions. add(), subtract(), setCounter() and getCounter(). Test the function as follows:
 - a. Click  in the left menu, then  button. It will deploy the contract in the simulator.
 - b. The deployed contract will be displayed as

 - c. Click ">", then the contract will be expanded with some buttons which corresponding to the functions as below:




- d. You now can test your functions by clicking buttons. Note that you can check the change of the counter value by clicking "getCounter" button.

[Create a Simple Real-time App Using Pub/Sub System]

In this part, you will simulate a simple real-time app for multi-device communication using a publish/subscribe system. You will 1) create an application, 2) test the app in the virtual environment, 3) connect your devices to the app and 4) add a simple function to the app. This lab task needs access to the Internet and Ubuntu VM. You need to run python code in Ubuntu VM.

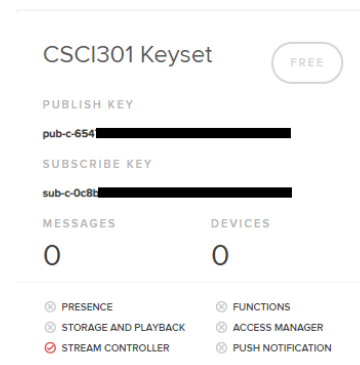
1. (Create an account in Pubnub) Create your account on www.pubnub.com. PubNub is a company providing a middleware solution for real-time communication between devices. The free version of the account is powerful enough for this task.
2. (Create your application) After you create an account, sign in to the site and select *Try for Free* on the screen.

- 1) On your home screen, select APPS  Apps on the left menu and create a new application by clicking “CREATE NEW APP” button.



- 2) Type the application name (e.g., CSCI301) and create apps.

- 3) Select the app you just created. Then you will move to the keyset of your application. One keyset is already created by default. Select the created keyset.



3. Scroll down the screen. You can see two tabs “Configuration” and “Get Started”. Select the “Get Started” tab and select Python language. Execute the python code that appeared on the webpage by following the steps below:

- 1) On your Ubuntu VM, install *pubnub package* for python using “pip3” by executing `$ pip3 install 'pubnub>=6.0.1'`

[Note] if “pip” is not installed in your computer, you can install it using the command `$ sudo apt-get install python-pip`

- 2) Create a file *Lab6.py*, open *Lab6.py* using any text editor (e.g., `$gedit Lab6.py`) and copy the python code from the webpage to the file and set your 'mySubscribeKey', 'myPublishKey' and 'myUniqueUUID'. You can get those keys from the keyset you just created. “myUniqueUUID”

is for your device. You can use any unique ID for this. (e.g., "ClientPy01").

4. Go back to the PubNub webpage and select Debug Console on the left menu. **[Note]** In advance of selecting the debug console, one of the keysets must be selected. Otherwise, clients will not be created.

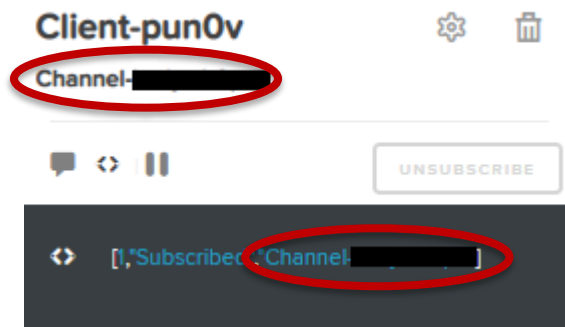
- 1) Add one more client by clicking "CREATE CLIENT".



- 2) Send a message by replacing "Enter Message Here" to "temp: 21". Check that all clients can receive the message on the screen.



- 3) Copy the channel name for further tasks. The text highlighted by the circles is the channel name.



5. Go back to your code in Ubuntu VM. Duplicate *Lab6.py* into *Lab6_pub.py*.
 - 1) You can set the channel name, which currently is 'my_channel' in the code by replacing it with the name you just copied. Try to run code (i.e., execute *python3 Lab6.py* in your Ubuntu terminal) and also check the consoles in Task 4.
 - 2) You also try to send a message from the console in Task 4 and observe that the message is delivered in your terminal.
 - 3) You create *Lab6_pub.py*, which publishes the message. Delete unwanted codes so that it only has a publish function. Change the channel name in `pubnub.publish().channels()` as your channel name copied in Task 4.

```
pubnub.publish()\
    .channel("/Your channelname")\
    .message({"sender": pnconfig.uuid, "content": "Hello From Python SDK"})\
    .pn_async(my_publish_callback)
```

- 4) Open two new terminals in Ubuntu and execute *Lab6.py* in one terminal and *Lab6_pub.py* in the other terminal, respectively. Check the clients who subscribe to the same channel on the consoles and the



terminal running *Lab6.py*. You can see that all clients receive the same message almost at the same time.

6. Modify *Lab6_pub.py* to take the message from the command. You can use `sys` package (https://www.tutorialspoint.com/python/python_command_line_arguments.htm). For example, CSCI301 will be stored in `sys.argv[1]` if you execute the following command:

```
$ python3 Lab6_pub.py CSCI301
```
7. Modify *Lab6.py* to save a subscribed message in a file when it subscribes to a message. You can achieve this by adding some codes in `message(self, pubnub, message)` of `MySubscribeCallback(SubscribeCallBack)` class.