Publish/Subscribe Getting Started Guide

Demo URL: http://pubnub.github.io/api-guide-with-tictactoe/

Pub/Sub In a Nutshell

PubNub utilizes a Publish/Subscribe model for realtime data streaming and device signaling which lets you establish and maintain persistent socket connections to any device and push data to global audiences in less than ¼ of a second.

You can publish messages to any given channel, and subscribing clients receive only messages associated with that channel. The message payload can be any JSON data including numbers, strings, arrays, and objects.

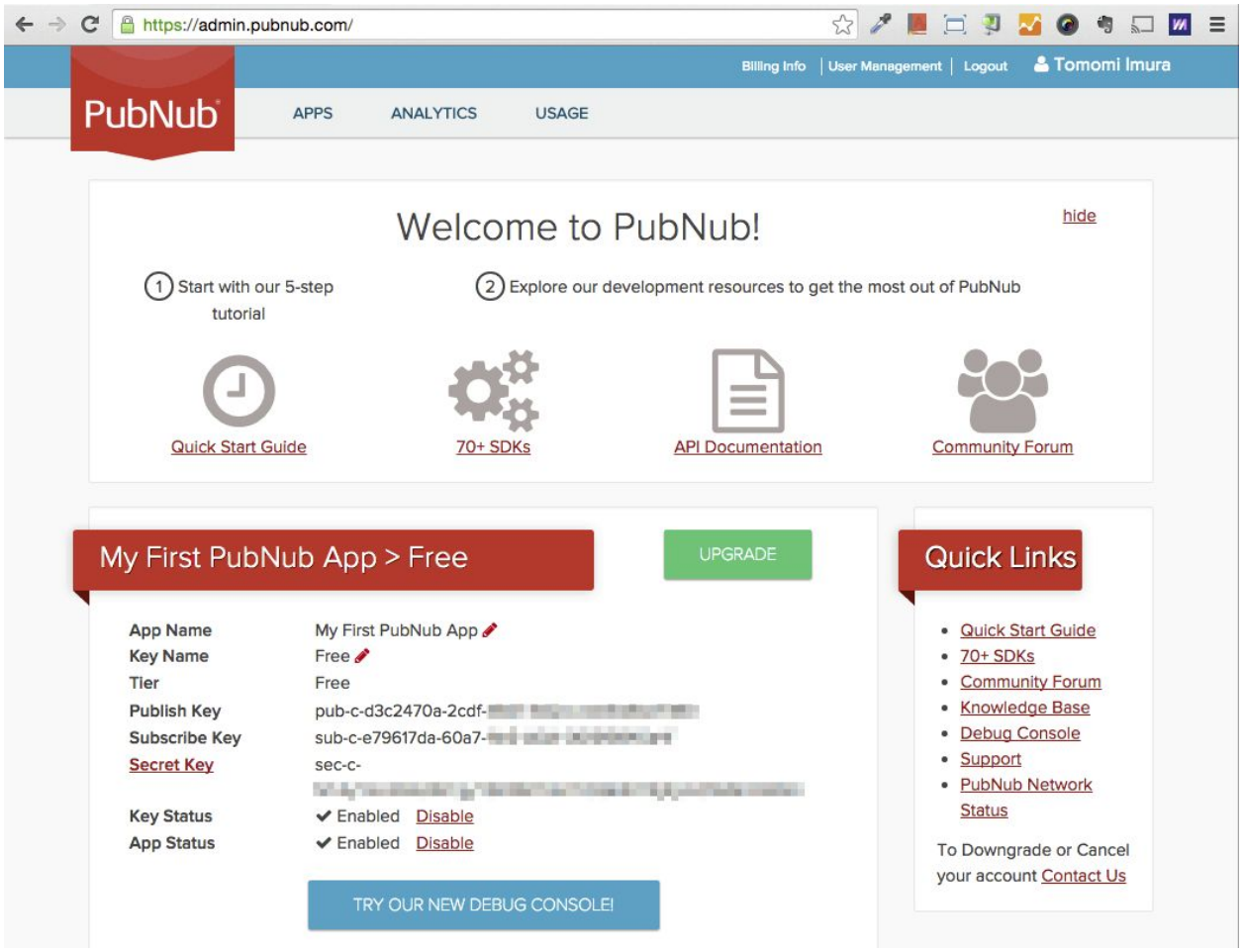
Pub/Sub Use Cases and Scenarios

- Chat rooms: Sending and receiving messages
- Locations and Connected cars: dispatching taxi cabs
- Smart sensors: Receiving data from a sensor for data visualizations
- Health: Monitoring heart rate from a patient's wearable device
- Multiplayer gamings
- Interactive media: audience-participating voting system
- And many many more...

Setting up Your PubNub Account

To build an application that leverages the PubNub Data Stream, you need to sign up for your account to obtain API keys.

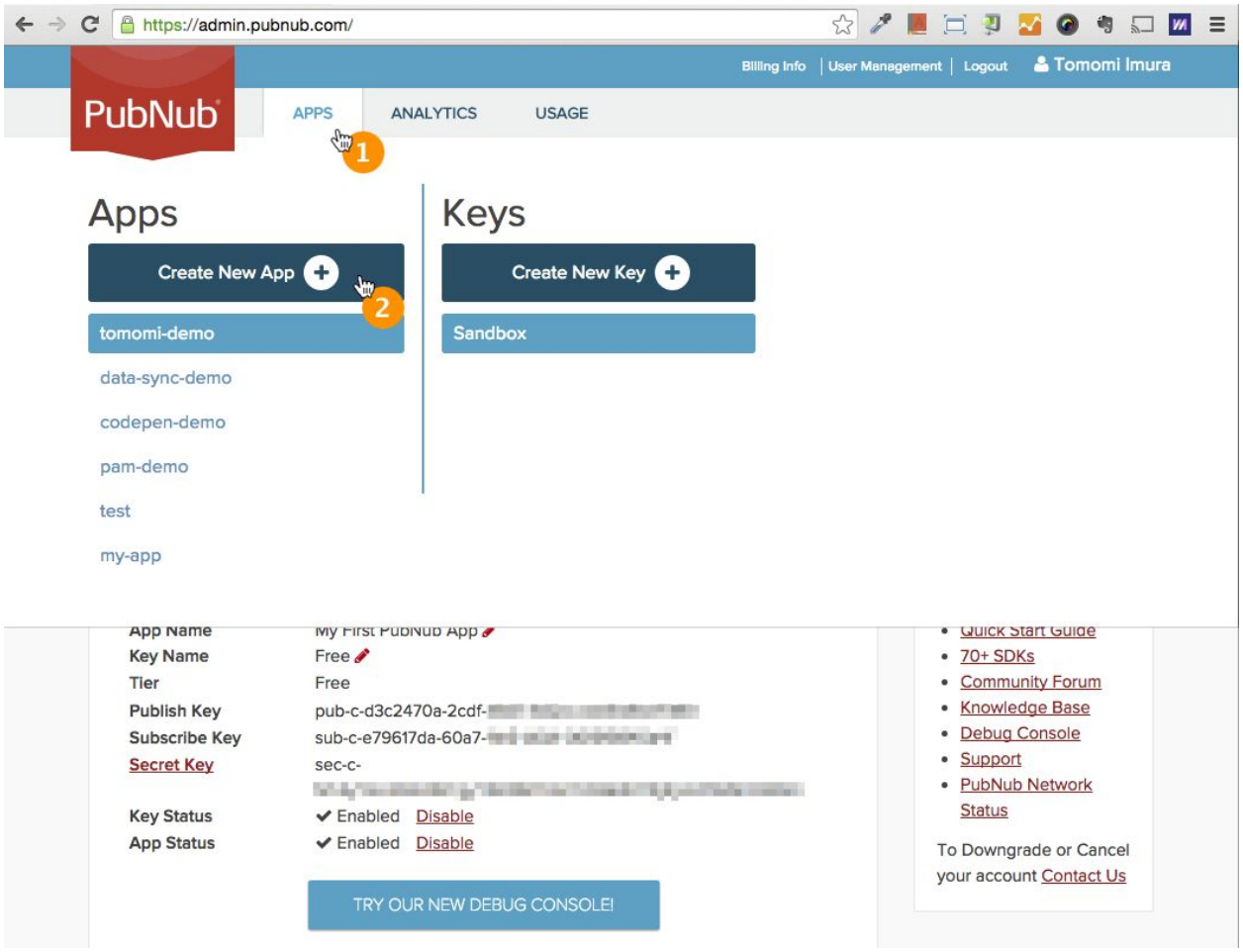
Once you have successfully signed up you will be taken to the admin page.



You can edit the app name by clicking the red pencil icon next to the “My First PubNub App” at App Name, if you wish.

You should have your *Publish Key* and *Subscribe Key* under your app. (You do not need the *Secret Key* for now. You will only need this when you are using Access Manager APIs).

You can add additional apps by clicking the **APPS** tab on the top menu bar, then clicking **Create New App**.



You will get a new set of Publish and Subscribe Keys each time you create a new app.

Getting Started with the API

To get started with PubNub JavaScript APIs, you need to include the latest (or desired) version of `pubnub.js` from CDN or locally.

```
<script src="http://cdn.pubnub.com/pubnub-3.x.x.min.js"></script>
```

You can find the latest version on the JavaScript SDK Getting Started page.

Initializing

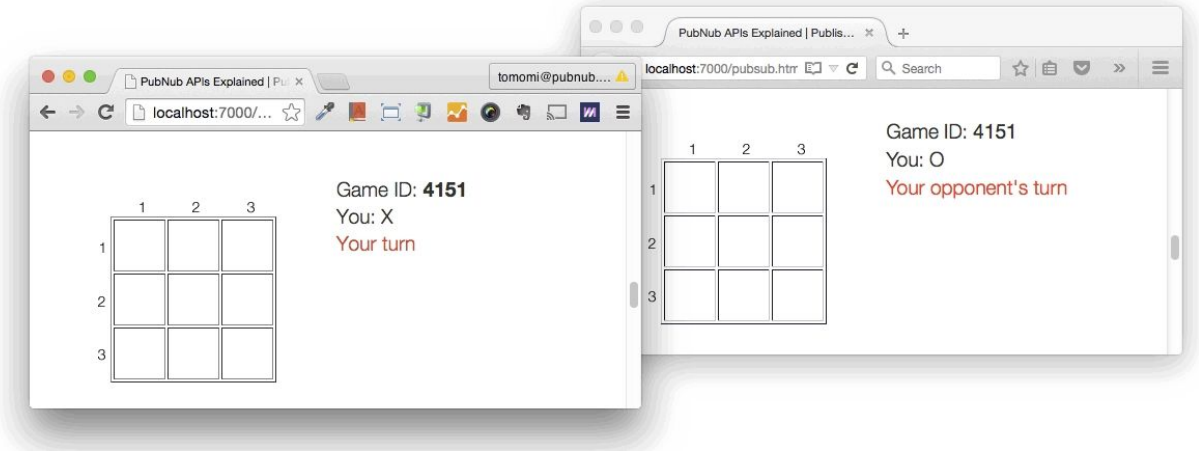
First things first: initialize a PubNub instance.

```
var pubnub = PUBNUB({
    subscribe_key: 'your_sub_key', // always required
    publish_key: 'your_pub_key'    // only required if publishing
});
```

Publishing and Subscribing

Take a look at the Tic-Tac-Toe demo. This is a simple multi-player game using PubNub's pub/sub APIs.

Each time player X click one of the cells, the move is published to PubNub data stream. The opponent, player O receives the data by subscribing to the channel and receiving the message that was published. The game's user-interface on both browser windows is updated, as well as the game status is updated upon each subscribe call.



Publishing

When a player makes a move, publish the player name (either X or O) and the position that indicates the coordinate of row and column (e.g. 1-2, 3-3, etc.) :

```
pubnub.publish({
      channel: 'tic-tac-toe',
      message: {
          player: 'X',
```

```
            position: '2-2'
        },
        callback: function(m){
            console.log(m);
        }
});
```

The callback is called when the operation is successful.

Channel names are UTF-8 compatible. Do not use any of the following characters:

- comma: ,
- slash: /
- backslash: \
- period: .
- asterisks: *
- colon: :

Also keep in mind that when you name the channel you want to keep it short enough not to take up the payload size. The maximum size of each message, including the channel name, is 32KB.

While you are playing the Tic-Tac-Toe demo, observe how each move is being published in the **Publish/Subscribe Explained** section on the bottom of the demo page.

Subscribing

All players subscribe to the channel get the published data from the stream, each time a player publishes their move.

In this scenario, the player X made a move (and the position is 2-2) and data is published to the channel, and right after the data is published, both X and O receive the data as they are subscribed to the channel.

The game status and the UI is updated on both ends, when the success callback is called:

```
pubnub.subscribe({
    channel: 'tic-tac-toe',
    connect: play,
    callback: function(m) {
        checkGameStatus(m);
        updateUI(m);
```

```
    },
    error: function(err) {
        console.log(err);
    }
});
```

There is an optional `connect` callback in the subscribe method- this is called the first time the connection to PubNub is established. In this scenario, the new game is being set up (defined in your `play` function) at the callback.

While you are playing the Tic-Tac-Toe demo, observe how each move is being subscribed at **Publish/Subscribe Explained** section on the bottom of the demo page.

Unsubscribing

Although this feature is not used in this Tic Tac Toe demo app, you can unsubscribe to stop receiving messages from a given channel if you wish:

```
pubnub.unsubscribe({
   channel : 'some_channel',
});
```