Certified

Developer - Associate

lab

lab title

# Creating a Low Cost Sync Database for JavaScript Applications with AWS V1.00

Course title

AWS Certified Developer Associate

# Table of Contents

## Contents

## About the Lab

These lab notes are to support the instructional videos on Creating a Low Cost Sync Database for JavaScript

Applications with AWS in the BackSpace AWS Certified Developer course.

This tutorial will focus on using S3 with the AWS Javascript SDK for Browser. This could be used with apps that access files on an S3 bucket and also require user information to be stored. I have chosen Facebook as the identity provider but a similar process applies for Amazon and Google. I would recommend only using one identity provider for your app so that your users don't produce multiple S3 buckets.

Please refer to the AWS JavaScript SDK documentation at:

http://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/S3.html

**Please note that AWS services change on a weekly basis and it is extremely important you check the version number on this document to ensure you have the lastest version with any updates or corrections.**

# Creating an Amazon S3 bucket to Host a Static Website

**In this section we will use the S3 service to create a bucket for hosting a static website. We will not be using it with a domain name to simplify the lab (this will be explained in the S3 lab). We will then enable Cross Origin Resource Sharing (CORS) which allows AWS resources to be requested from another domain outside the domain from which the resources originated, without being blocked by the browser.**

Select the S3 Console.

Create a bucket and give it a unique name such as lab.your.domain.name

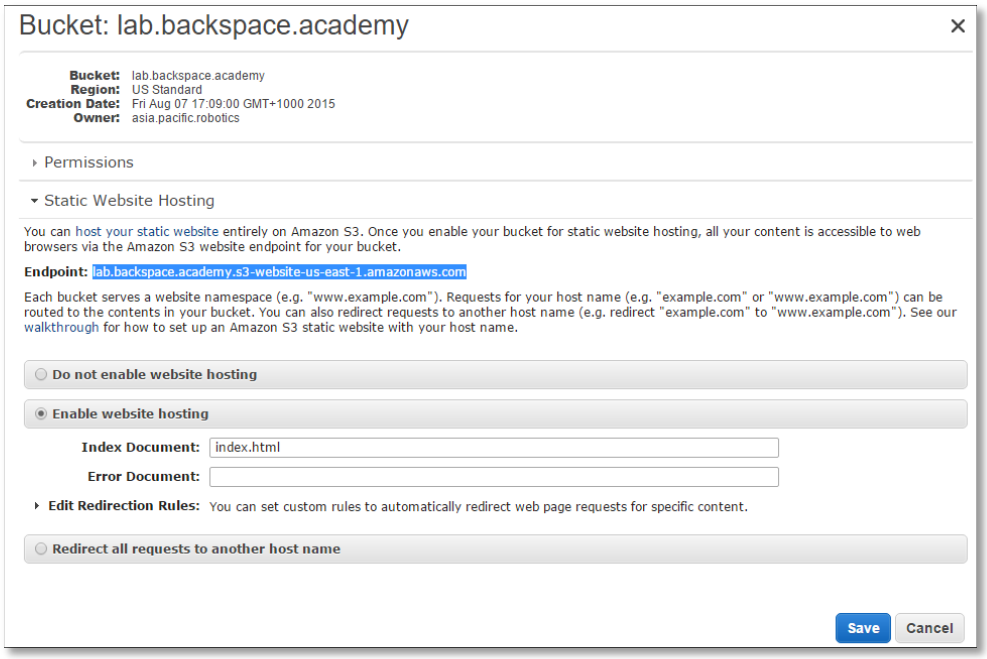
Select the bucket and click "Properties"

Expand "Static Website Hosting"

Check "Enable website hosting"

Enter "index.html" for the index document.

Take note of the endpoint for the website as we will need this later when registering with Facebook.


Bucket: lab.backspace.academy

| | |
|---|---|
| **Bucket:** | lab.backspace.academy |
| **Region:** | US Standard |
| **Creation Date:** | Fri Aug 07 17:09:00 GMT+1000 2015 |
| **Owner:** | asia.pacific.robotics |

Permissions

Static Website Hosting

You can host your static website entirely on Amazon S3. Once you enable your bucket for static website hosting, all your content is accessible to web browsers via the Amazon S3 website endpoint for your bucket.

**Endpoint:** lab.backspace.academy.s3-website-us-east-1.amazonaws.com

Each bucket serves a website namespace (e.g. "www.example.com"). Requests for your host name (e.g. "example.com" or "www.example.com") can be routed to the contents in your bucket. You can also redirect requests to another host name (e.g. redirect "example.com" to "www.example.com"). See our walkthrough for how to set up an Amazon S3 static website with your host name.

Do not enable website hosting [✓ Enable website hosting]

**Index Document:** index.html

**Error Document:**

**Edit Redirection Rules:** You can set custom rules to automatically redirect web page requests for specific content.

Redirect all requests to another host name

Click “Save”.

Click properties again.

Click to expand permissions.

Select “Add CORS Configuration”

Replace with the following XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01">
   <CORSRule>
      <AllowedOrigin>*</AllowedOrigin>
      <AllowedMethod>GET</AllowedMethod>
      <AllowedMethod>PUT</AllowedMethod>
      <AllowedMethod>POST</AllowedMethod>
      <AllowedMethod>DELETE</AllowedMethod>
      <AllowedHeader>*</AllowedHeader>
   </CORSRule>
</CORSConfiguration>
```

Click “Close”

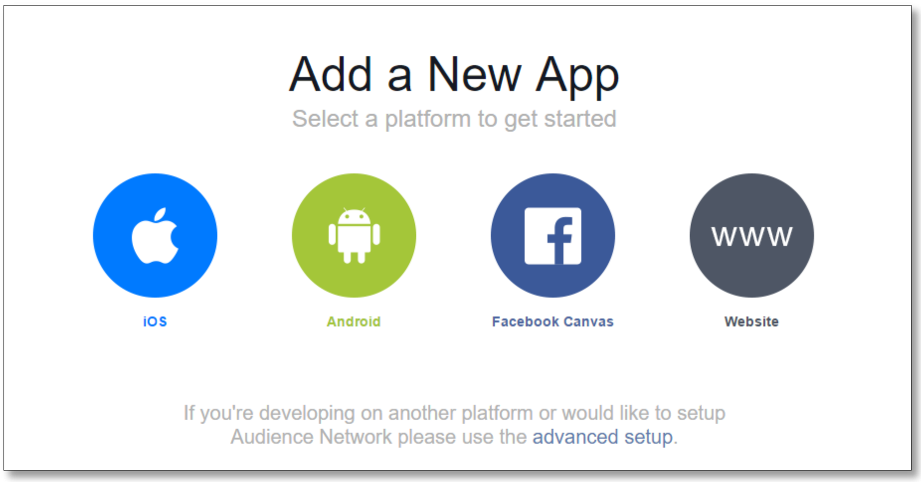
Click “Save”

# Creating a Facebook App and Getting the App ID

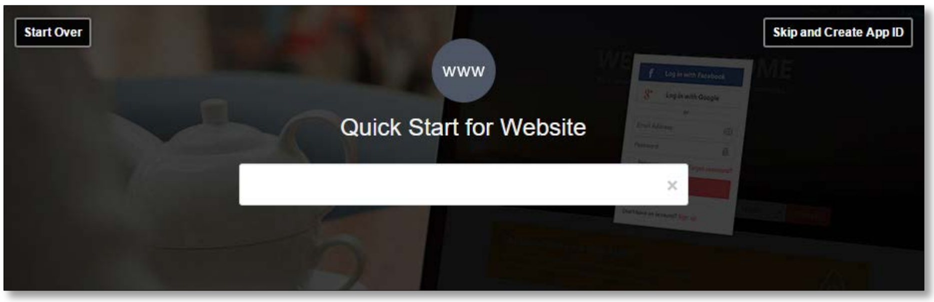
**In this section we will create a Facebook App for our web application and get the App ID for our application to reference.**

Go to https://developers.facebook.com/apps/.

Select “My Apps” then click “Add a new App”


Add a New App
Select a platform to get started
iOS
Android
Facebook Canvas
Website
If you're developing on another platform or would like to setup Audience Network please use the advanced setup.

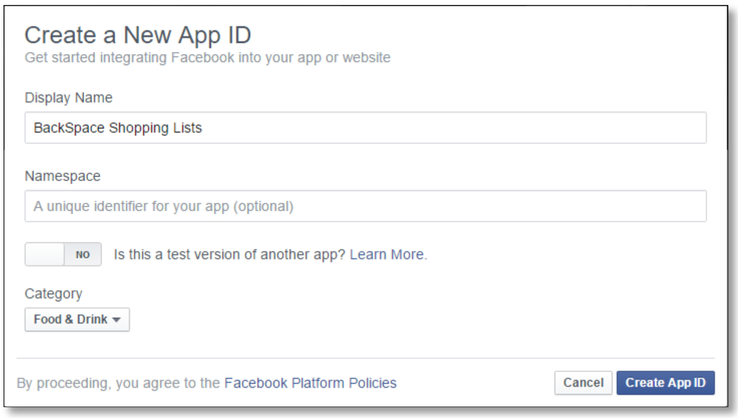
Select “Website”



Click “Skip and Create App ID”

Give your app a unique name.

Select “Food and Drink” for category.



Create a New App ID
Get started integrating Facebook into your app or website

Display Name: BackSpace Shopping Lists

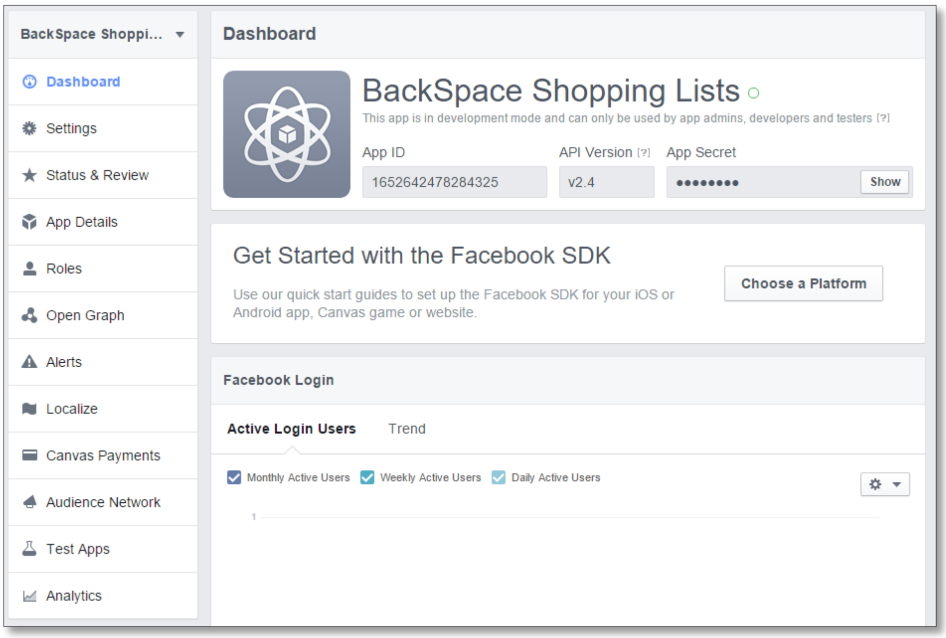
Namespace: A unique identifier for your app (optional)

NO Is this a test version of another app? Learn More.

Category: Food & Drink

By proceeding, you agree to the Facebook Platform Policies

Click "Create App ID"



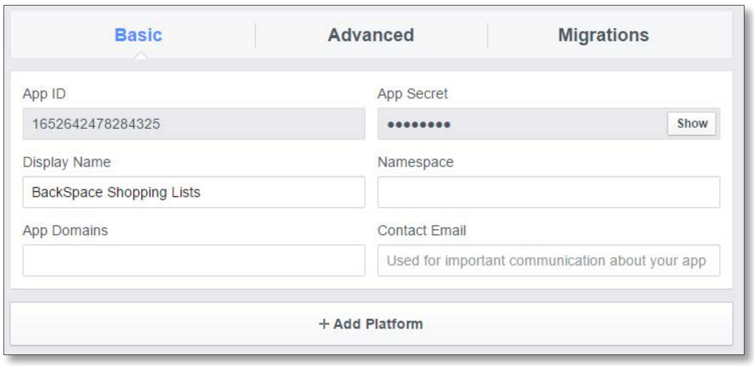
Dashboard

BackSpace Shopping Lists
This app is in development mode and can only be used by app admins, developers and testers [?]

App ID: 1652642478284325
API Version [?]: v2.4
App Secret: ●●●●●●●● Show

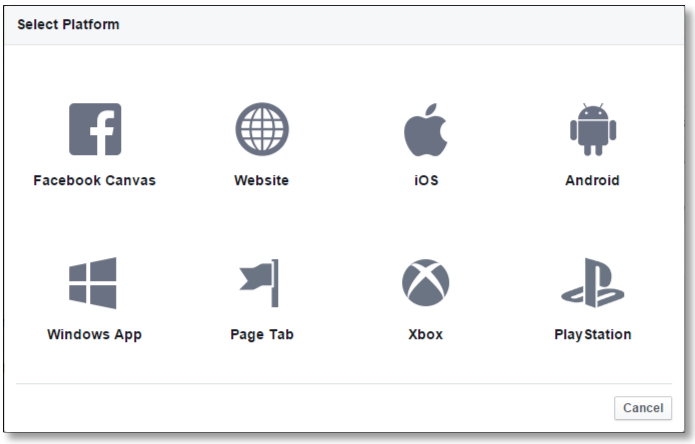
Get Started with the Facebook SDK
Use our quick start guides to set up the Facebook SDK for your iOS or Android app, Canvas game or website.

Click on "Settings"



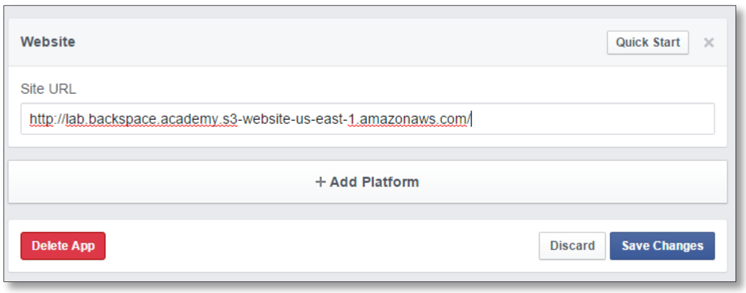
App ID: 1652642478284325
App Secret: ●●●●●●●● Show
Display Name: BackSpace Shopping Lists
Namespace:
App Domains:
Contact Email: Used for important communication about your app

Click on "Add Platform"



Click on “Website”

Input the URL of the site you created previously.



Site URL

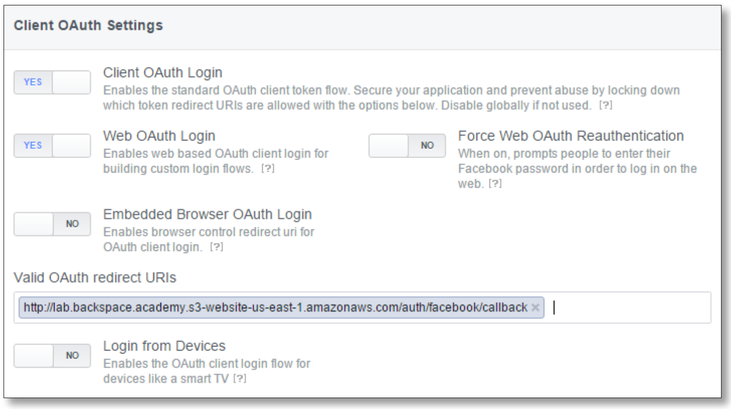
http://lab.backspace.academy.s3-website-us-east-1.amazonaws.com/

Click “Save Changes”

Click on the “Advanced” Tab

Scroll down to Client OAuth Settings

Enter redirect URL as YOUR_WEBSITE_URL/auth/facebook/callback



Client OAuth Settings

YES Client OAuth Login
Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]

YES Web OAuth Login
Enables web based OAuth client login for building custom login flows. [?]

NO Force Web OAuth Reauthentication
When on, prompts people to enter their Facebook password in order to log in on the web. [?]

NO Embedded Browser OAuth Login
Enables browser control redirect uri for OAuth client login. [?]

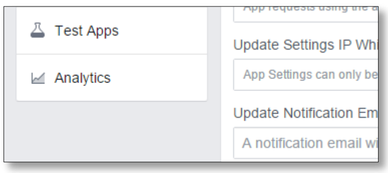
Valid OAuth redirect URIs

http://lab.backspace.academy.s3-website-us-east-1.amazonaws.com/auth/facebook/callback

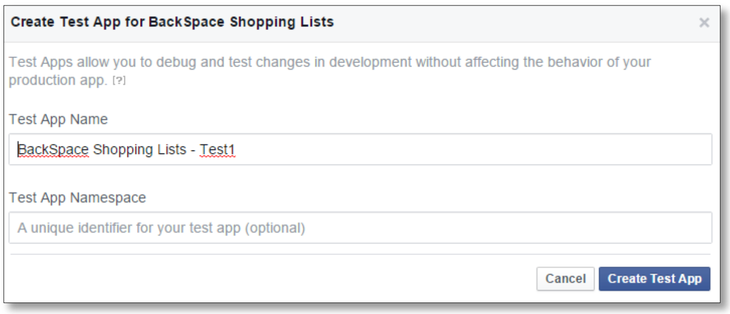
NO Login from Devices
Enables the OAuth client login flow for devices like a smart TV [?]

Click “Save Changes”

Click “Test Apps”



Click “Create a Test App”


**Create Test App for BackSpace Shopping Lists**

Test Apps allow you to debug and test changes in development without affecting the behavior of your production app. [?]

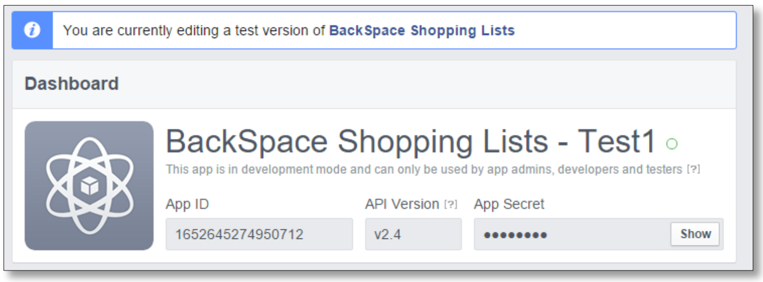
Test App Name

BackSpace Shopping Lists - Test1

Test App Namespace

A unique identifier for your test app (optional)

Click “Create a Test App”


You are currently editing a test version of **BackSpace Shopping Lists**

**Dashboard**

BackSpace Shopping Lists - Test1

This app is in development mode and can only be used by app admins, developers and testers [?]

| App ID | API Version [?] | App Secret |
|---|---|---|
| 1652645274950712 | v2.4 | •••••••• |

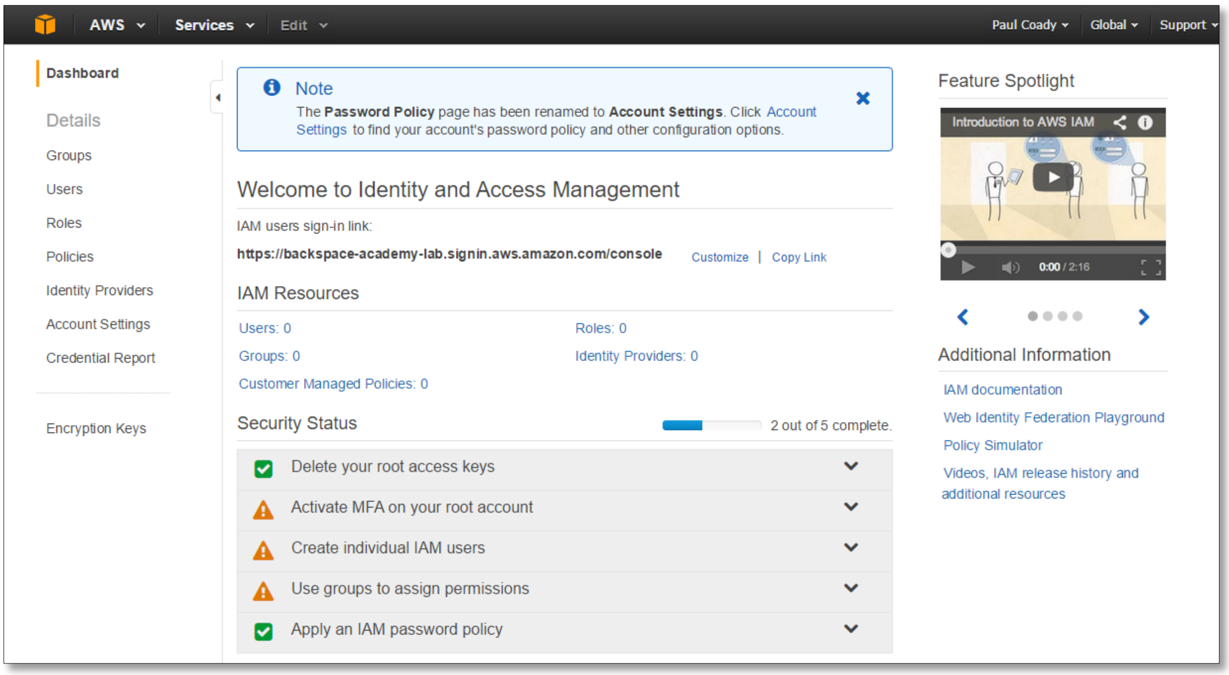
Take note of your test “App ID”, we will need it later.

When you change from test to production you will use the main app ID but for testing we will use the test App ID.

# Creating an IAM Role to Assign Users Logged in through Facebook

**In this section we will use the Identity and Access Management (IAM) service to create a policy for creating federated users. We will then create a Facebook app and use this for identification of users. User access will be restricted to an S3 bucket in their Facebook ID.**

Select the IAM Console



Note

The **Password Policy** page has been renamed to **Account Settings**. Click Account Settings to find your account's password policy and other configuration options.

Welcome to Identity and Access Management

IAM users sign-in link:

https://backspace-academy-lab.signin.aws.amazon.com/console

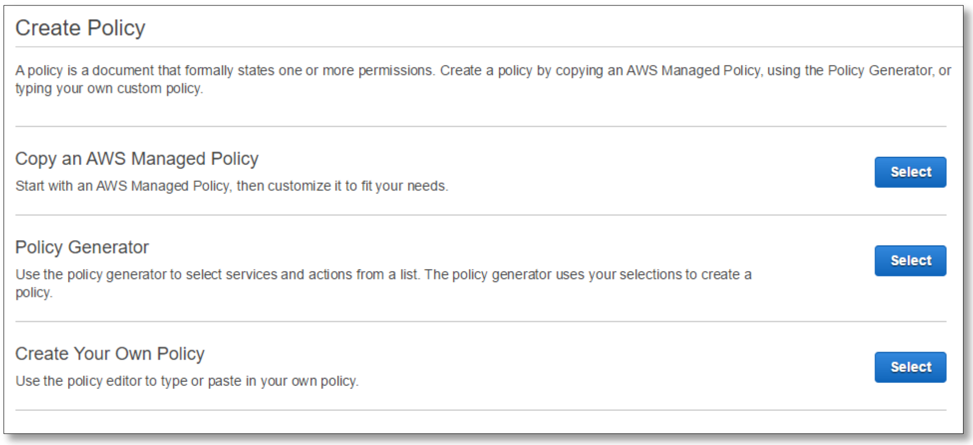
IAM Resources

| | |
|---|---|
| Users: 0 | Roles: 0 |
| Groups: 0 | Identity Providers: 0 |
| Customer Managed Policies: 0 | |

Security Status — 2 out of 5 complete.

- ✓ Delete your root access keys
- ⚠ Activate MFA on your root account
- ⚠ Create individual IAM users
- ⚠ Use groups to assign permissions
- ✓ Apply an IAM password policy

Click “Policies” then “Get Started”. Click “Create Policy”.



Create Policy

A policy is a document that formally states one or more permissions. Create a policy by copying an AWS Managed Policy, using the Policy Generator, or typing your own custom policy.

Copy an AWS Managed Policy

Start with an AWS Managed Policy, then customize it to fit your needs.

Policy Generator

Use the policy generator to select services and actions from a list. The policy generator uses your selections to create a policy.

Create Your Own Policy

Use the policy editor to type or paste in your own policy.

Select “Create Your Own Policy”.

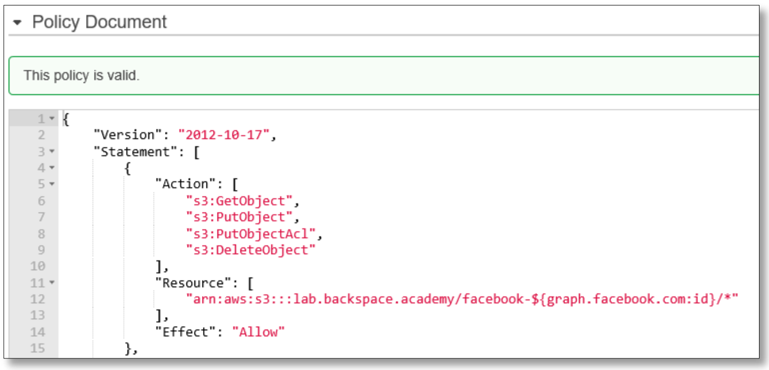
Call the policy “BackSpaceAcademyFacebook”

Give it a description.

Add the following to Policy Document. Replace YourBucketName with the bucket you created (bucket name only, not the website URL). **It is very important to ensure that the "Version": "2012-10-17" statement is included otherwise the policy variables will not be recognised**:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::YourBucketName/facebook-${graph.facebook.com:id}/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::YourBucketName"
            ],
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": "facebook-${graph.facebook.com:id}"
                }
            }
        }
    ]
}
```

Click “Validate Policy” to check for errors



Policy Document

This policy is valid.

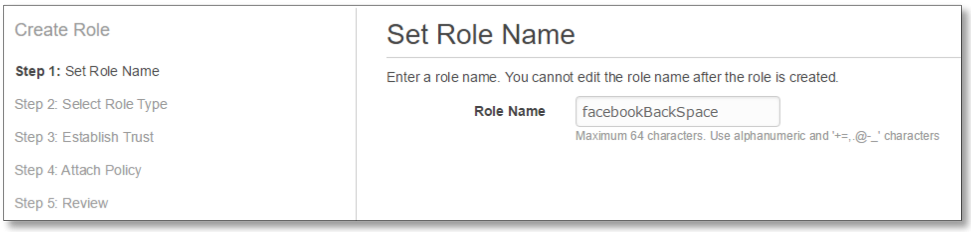
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::lab.backspace.academy/facebook-${graph.facebook.com:id}/*"
            ],
            "Effect": "Allow"
        },
```

Click “Create Policy”

Go to “Roles”

Click “Create Role”

Call the role “facebookBackSpace”



Create Role

**Step 1:** Set Role Name

Step 2: Select Role Type

Step 3: Establish Trust

Step 4: Attach Policy

Step 5: Review

Set Role Name

Enter a role name. You cannot edit the role name after the role is created.

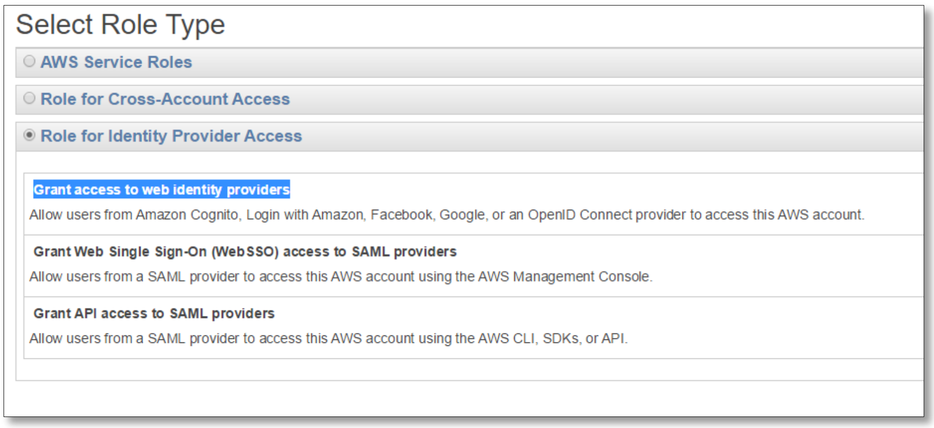
**Role Name** facebookBackSpace

Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters

Click “Next Step”

Click “Role for Identity Provider Access”

Select “Grant access to web identity providers”



Select Role Type

○ AWS Service Roles

○ Role for Cross-Account Access

● Role for Identity Provider Access

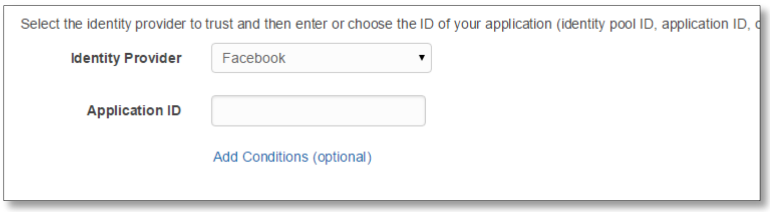
Grant access to web identity providers
Allow users from Amazon Cognito, Login with Amazon, Facebook, Google, or an OpenID Connect provider to access this AWS account.

Grant Web Single Sign-On (WebSSO) access to SAML providers
Allow users from a SAML provider to access this AWS account using the AWS Management Console.

Grant API access to SAML providers
Allow users from a SAML provider to access this AWS account using the AWS CLI, SDKs, or API.

Enter Facebook as the Identity Provider.

Enter you Facebook App ID.

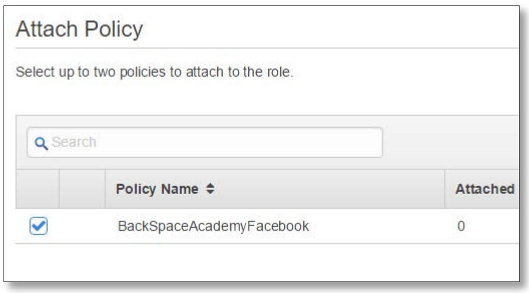


Select the identity provider to trust and then enter or choose the ID of your application (identity pool ID, application ID,

| | |
|---|---|
| Identity Provider | Facebook |
| Application ID | |

Click “Next Step” for “Verify Role Trust”

Attach the “BackSpaceAcademyFacebook” policy.



Attach Policy

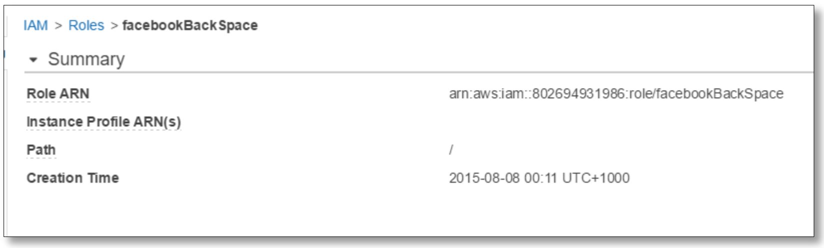
Select up to two policies to attach to the role.

| | Policy Name | Attached |
|---|---|---|
| ✓ | BackSpaceAcademyFacebook | 0 |

Click “Next Step”

Click “Create Role”

Click on the new role and take note of the Role ARN (we will need this later)



IAM > Roles > facebookBackSpace

Summary

| | |
|---|---|
| Role ARN | arn:aws:iam::802694931986:role/facebookBackSpace |
| Instance Profile ARN(s) | |
| Path | / |
| Creation Time | 2015-08-08 00:11 UTC+1000 |

# Creating an Application Using Facebook Login and Local Storage

**In this section we will create an application using Federated login with Facebook to provide access to AWS S3 resources. We will save data to local browser storage.**

Download the following files to a folder (select “Raw” then save) from
https://gist.github.com/pcoady/5633ac8d8f77e4a898f3

- index.html
- app.js
- app.css

Download the following assets for the app to the folder also:

http://cdn.backspace.academy/public/classroom/aws-cd-a/IAM-S3-lab/supermarket.jpg

http://cdn.backspace.academy/public/classroom/aws-cd-a/IAM-S3-lab/favicon.ico

Open up Windows File Explorer and right click on the folder and select “Git Init Here”.

You will now see a .git folder created inside your folder.

Open Atom IDE.

Open the main folder (not the .git folder).

You will now see your project in the side menu.

Select “Packages” – “Git Plus” – “Add All Commit”.

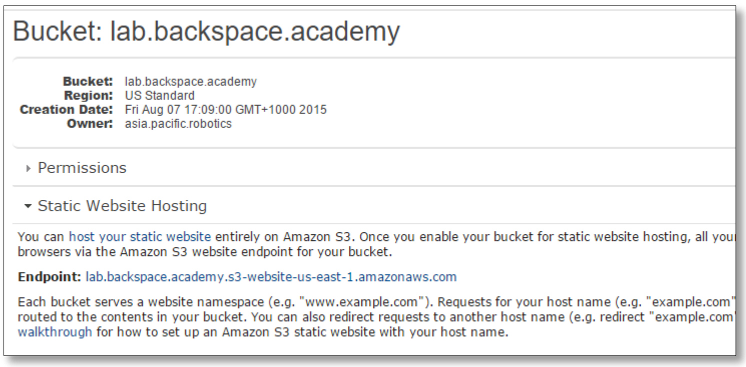
Type in a commit message then save and close COMMIT_EDITMSG.

You have now done your first commit. From now on commit regularly after changes have been made.

Open the S3 console.

Upload the project files to your bucket (make sure they are all upload as public).

Now point your browser to the bucket website endpoint.



Bucket: lab.backspace.academy

| | |
|---|---|
| **Bucket:** | lab.backspace.academy |
| **Region:** | US Standard |
| **Creation Date:** | Fri Aug 07 17:09:00 GMT+1000 2015 |
| **Owner:** | asia.pacific.robotics |

Permissions

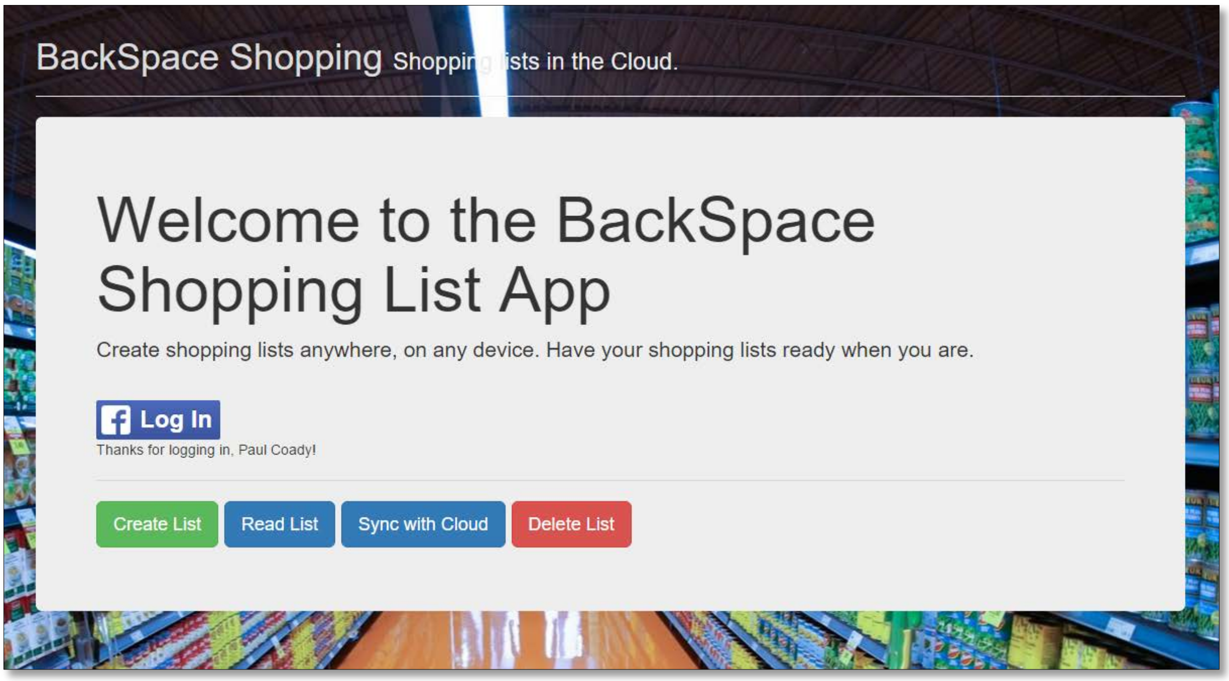
Static Website Hosting

You can host your static website entirely on Amazon S3. Once you enable your bucket for static website hosting, all you
browsers via the Amazon S3 website endpoint for your bucket.

**Endpoint:** lab.backspace.academy.s3-website-us-east-1.amazonaws.com

Each bucket serves a website namespace (e.g. "www.example.com"). Requests for your host name (e.g. "example.com"
routed to the contents in your bucket. You can also redirect requests to another host name (e.g. redirect "example.com
walkthrough for how to set up an Amazon S3 static website with your host name.

Sign in with Facebook.



BackSpace Shopping Shopping lists in the Cloud.

Welcome to the BackSpace Shopping List App

Create shopping lists anywhere, on any device. Have your shopping lists ready when you are.

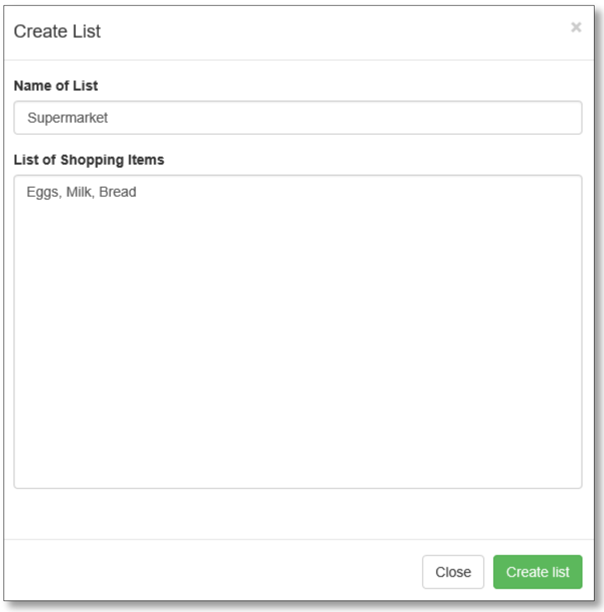
Thanks for logging in, Paul Coady!

Press F12 to open the Developer Tools.

Go back to the browser and click ”CreateList”

Create a new shopping list and click “Create List”

Now click close.


Create List

**Name of List**

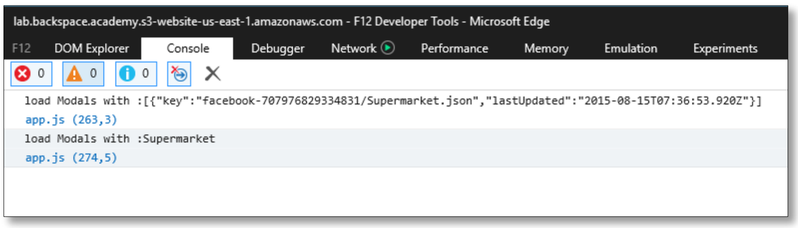
Supermarket

**List of Shopping Items**

Eggs, Milk, Bread

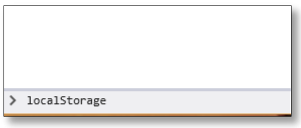
A shopping list has been created and stored in persistent local storage. You will get a notification 'Shopping list successfully created'.

If you check the developer tools console you will see an object has been created and has been added to the modal screen dropdown menus.


lab.backspace.academy.s3-website-us-east-1.amazonaws.com - F12 Developer Tools - Microsoft Edge
F12
DOM Explorer
Console
Debugger
Network
Performance
Memory
Emulation
Experiments
0
0
0

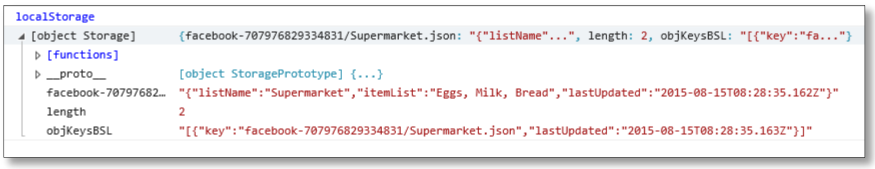
```
load Modals with :[{"key":"facebook-707976829334831/Supermarket.json","lastUpdated":"2015-08-15T07:36:53.920Z"}]
app.js (263,3)
load Modals with :Supermarket
app.js (274,5)
```

Type “localStorage” at the bottom of the console to see the contents of the device local storage.



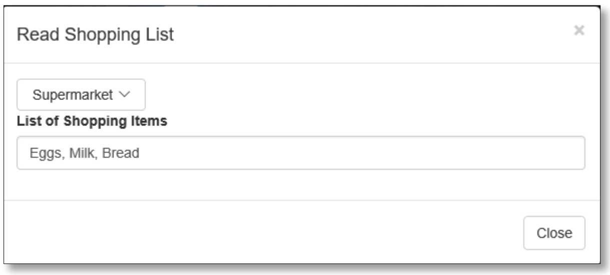
```
> localStorage
```

Click to expand the objects we created.



```
localStorage
◢ [object Storage]      {facebook-707976829334831/Supermarket.json: "{"listName"...", length: 2, objKeysBSL: "[{"key":"fa..."}
  ▷ [functions]
  ▷ __proto__           [object StoragePrototype] {...}
    facebook-70797682…  "{"listName":"Supermarket","itemList":"Eggs, Milk, Bread","lastUpdated":"2015-08-15T08:28:35.162Z"}"
    length              2
    objKeysBSL          "[{"key":"facebook-707976829334831/Supermarket.json","lastUpdated":"2015-08-15T08:28:35.163Z"}]"
```

Now let’s read the shopping list. Click “Read List” and select the list to read its contents.



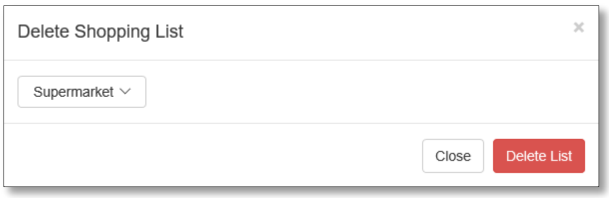
List of Shopping Items

Eggs, Milk, Bread

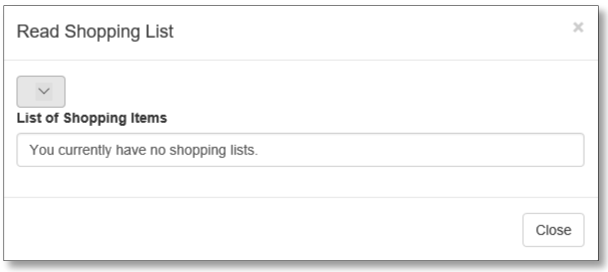
Now let’s delete the list.

Click “Delete List”

Select the list and click “Delete List”



Now go back to read lists to see it has been deleted.



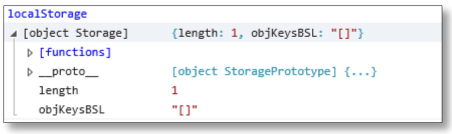
List of Shopping Items

You currently have no shopping lists.

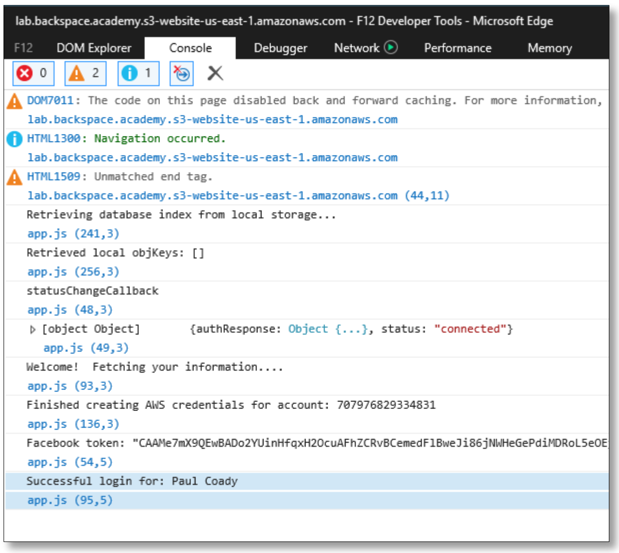
Click “Close”

Now refresh the browser screen and look at the developer tools console again.

Check localStorage again to see the object hase been deleted and the objects index is now empty.



```
localStorage
◢ [object Storage]      {length: 1, objKeysBSL: "[]"}
  ▷ [functions]
  ▷ __proto__           [object StoragePrototype] {...}
    length              1
    objKeysBSL          "[]"
```


lab.backspace.academy.s3-website-us-east-1.amazonaws.com - F12 Developer Tools - Microsoft Edge
F12
DOM Explorer
Console
Debugger
Network
Performance
Memory
0
2
1

```
DOM7011: The code on this page disabled back and forward caching. For more information,
lab.backspace.academy.s3-website-us-east-1.amazonaws.com
HTML1300: Navigation occurred.
lab.backspace.academy.s3-website-us-east-1.amazonaws.com
HTML1509: Unmatched end tag.
lab.backspace.academy.s3-website-us-east-1.amazonaws.com (44,11)
Retrieving database index from local storage...
app.js (241,3)
Retrieved local objKeys: []
app.js (256,3)
statusChangeCallback
app.js (48,3)
[object Object]       {authResponse: Object {...}, status: "connected"}
  app.js (49,3)
Welcome!  Fetching your information....
app.js (93,3)
Finished creating AWS credentials for account: 707976829334831
app.js (136,3)
Facebook token: "CAAMe7mX9QEwBADo2YUinHfqxH2OcuAFhZCRvBCemedF1BweJi86jNWHeGePdiMDRoL5eOE
app.js (54,5)
Successful login for: Paul Coady
app.js (95,5)
```

Here you can see the application has:

- retrieved the empty object index from local storage;
- downloaded Facebook token and user information; and
- created temporary AWS credentials after receiving  the Facebook token.

Our basic application is ready to be developed into a cloud application. We have AWS temporary credentials for federated users and can use these to access our S3 bucket and sync with local storage. This way we can synchronise data across all devices based on the most recent “lastUpdated” entry of the object.

# Storing your App Data in Amazon S3

**In this section we will use our temporary AWS credentials to upload and download JSON files to S3. We will then look at synchronising local data on devices to the cloud data based on the most current "lastUpdated" entry.**

Open Atom IDE.

Change the createShoppingList function in app.js to:

```
function createShoppingList(){
  if (validLogin()){
    if (!userSync){
      alert('Please sync database before creating new lists.');
      return;
    }
    var listName = $('#inputCreateName')[0].value;
    if (listName === ''){
      growl('danger', 'List Name Error', 'Please enter a valid name for your shopping list!');
      return;
    }
    $('#btnCreate').attr("disabled", "disabled");
    showSpinner();
    data = {
      listName: listName,
      itemList: $('#inputCreateList')[0].value,
      lastUpdated: new Date()
    };
    var key = prefix + '/' +  listName + '.json';
    saveLocalStorage(data,key);
    var keyDetails =     {
      'key': key,
      'lastUpdated': new Date()
    }
    objKeys.push(keyDetails);
    saveLocalStorage(objKeys,'objKeysBSL');
    console.log('Creating shopping list in the Cloud: ' + key)
    saveS3(data,key);
    saveS3(objKeys, (prefix + '/' + 'objKeysBSL.json'));
    growl('success', 'List Created', 'Shopping list successfully created in Cloud');
    hideSpinner();
    $('#btnCreate').removeAttr("disabled");
    loadModals();
  }
}
```

Now create the saveS3 function:

```
function saveS3(data,key){
  if (validLogin()){
    //Object key will be facebook-USERID#/FILE_NAME
    var temp = JSON.stringify(data);
    var params = {
        Bucket: bucketName,
        Key: key,
        ContentType: 'json',
        Body: temp
    };
    console.log('Saving data to S3 at: '+ JSON.stringify(key))
    S3.putObject(params, function (err, data) {
        if (err) {
            console.log('Error saving to cloud: ' + err);
            growl('danger','Error.','Unable to save data to cloud.');
        } else {
          growl('success','Finished','Data saved to cloud.');
        }
    });
 }
}
```

Also change syncShoppingList to:

```
function syncShoppingList(){
  userSync = true;
}
```

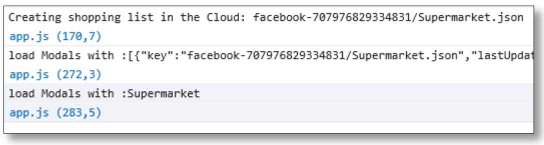
Upload to S3 again (make sure it is public).

Open your browser to the app url

Open the developer tools.

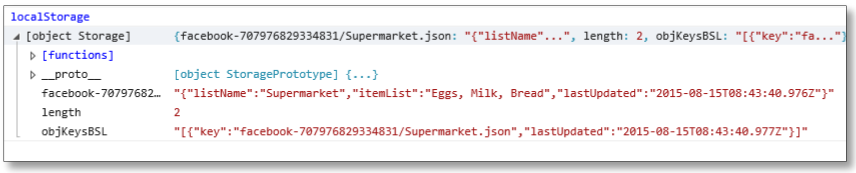
Create another shopping list in the app.

Now check the console to see the object created in S3.



```
Creating shopping list in the Cloud: facebook-707976829334831/Supermarket.json
app.js (170,7)
load Modals with :[{"key":"facebook-707976829334831/Supermarket.json","lastUpdat
app.js (272,3)
load Modals with :Supermarket
app.js (283,5)
```

Now check localStorage in dev tools

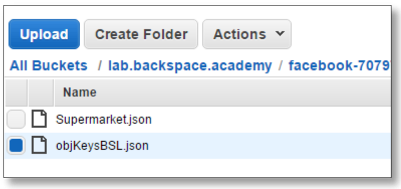


```
localStorage
◢ [object Storage]      {facebook-707976829334831/Supermarket.json: "{"listName"...", length: 2, objKeysBSL: "[{"key":"fa..."}
  ▷ [functions]
  ▷ __proto__           [object StoragePrototype] {...}
    facebook-70797682…  "{"listName":"Supermarket","itemList":"Eggs, Milk, Bread","lastUpdated":"2015-08-15T08:43:40.976Z"}"
    length              2
    objKeysBSL          "[{"key":"facebook-707976829334831/Supermarket.json","lastUpdated":"2015-08-15T08:43:40.977Z"}]"
```

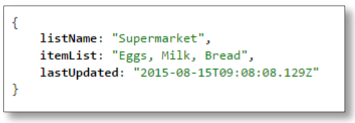
Now open the AWS S3 console.

Open the newly created facebook folder.

You will see our JSON index file and object file created.

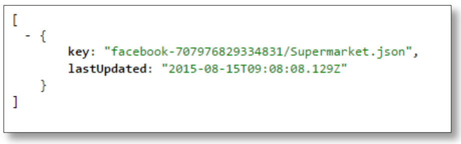


Open Supermarket.json to see its contents



```
{
    listName: "Supermarket",
    itemList: "Eggs, Milk, Bread",
    lastUpdated: "2015-08-15T09:08:08.129Z"
}
```

Open objKeysBSL.json to see the reference to the object we created. Any new shopping lists will be added to this list.



```
[
  - {
        key: "facebook-707976829334831/Supermarket.json",
        lastUpdated: "2015-08-15T09:08:08.129Z"
    }
]
```

We have successfully creates a shopping list that is replicated in S3.

Now go back to Atom IDE.

Change deleteShoppingList to:

```
function deleteShoppingList(shoppingList){
  if (objKeys.length>0){
    $('#btnDelete').attr("disabled", "disabled");
    var key = objKeys[shoppingList].key;
    deleteLocalStorage(key);
    objKeys.splice(shoppingList, 1);
    saveLocalStorage(objKeys,'objKeysBSL');
    if (validLogin()){
      console.log('Deleting shopping list from S3: ' + key)
      deleteS3(key);
      saveS3(objKeys, (prefix + '/' + 'objKeysBSL.json'));
    }
    loadModals();
    growl('success','Deleted.','Shopping list successfully deleted');
    $('#btnDelete').removeAttr("disabled");
  }
  else{
    growl('danger','Error.','You have no shopping lists to delete!');
  }
}
```

Now update the deleteS3 function:

```
function deleteS3(key){
  var params = {
    Bucket: bucketName,
    Key: key
  };
  S3.deleteObject(params, function(err, data) {
    if (err) {
      console.log(err, err.stack);
      growl('danger','Error','Data not deleted from cloud.');
    }
    else {
      growl('success','Finished','Data deleted from cloud.');
    }
  });
}
```

Open the S3 console.

Delete the objects in the Facebook folder.

Upload to S3 again in the main folder, not the Facebook folder (make sure it is public).

Clear localStorage in dev tools.

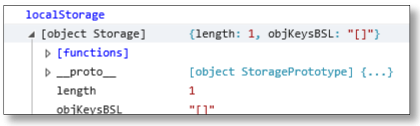
Open your browser to the app url

Create a new shopping list.

Check it has been created in localStorage and S3.

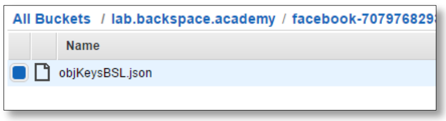
Click “Delete list” and delete the list.

Open the developer tools to see the list has been deleted and the object index is empty.



```
localStorage
◢ [object Storage]        {length: 1, objKeysBSL: "[]"}
  ▷ [functions]
  ▷ __proto__             [object StoragePrototype] {...}
    length                1
    objKeysBSL            "[]"
```

Open the S3 console to see the list has been also deleted and the object index is empty (make sure you click the refresh button).



All Buckets / lab.backspace.academy / facebook-70797682

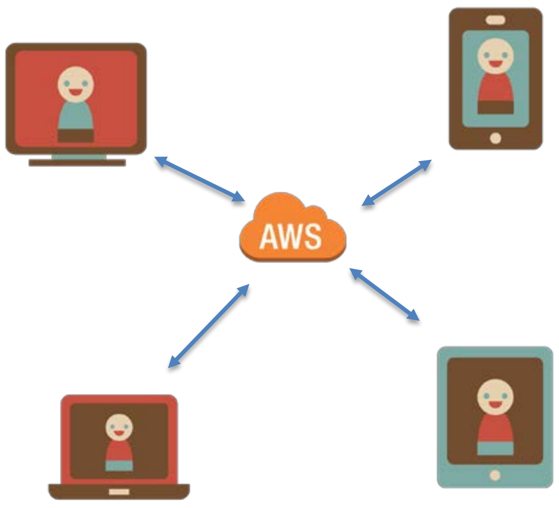
| | | Name |
|---|---|---|
| 🟦 | | objKeysBSL.json |

We have successfully created and replicated an object on local storage and S3. We have also successfully deleted the object from both local storage and S3.

Next we will create a sync operation that will sync the device with the current index on the cloud. If the list on the cloud is newer local storage will be updated and vice versa.

# Synchronising files on a Device to Amazon S3

**In this section we will create a sync operation that will sync the device with the current index on the cloud. If the list on the cloud is newer local storage will be updated and vice versa.**



Open Atom IDE

Change syncShoppingList to:

```
function syncShoppingList(){
  if (validLogin()){
    var temp = [];
    var temp2 = [];
    var params = {
      Bucket: bucketName,
      Key: (prefix + '/' + 'objKeysBSL.json')
    };
    showSpinner();
    // Get objKeys index from S3
    S3.getObject(params, function(err, data) {
      if (err){
        console.log(err, err.stack);
        // No shopping lists in cloud, check local storage
        checkLocal();
      }
      else  {
        temp = JSON.parse(data.Body);
        if (temp.length > 0) {
          console.log('Downloaded Object keys: ' + JSON.stringify(temp))
          saveLocalStorage(temp,'objKeysBSL')
          // Start the sync
          syncS3Local(temp,false);
        }
        else {
          // No shopping lists in cloud, check local storage
          checkLocal();
        }
      }
    });
  }
  else{
    growl('warning','Facebook Login','You are not logged in with Facebook');
  }
}
```

Change checkLocal to:

```
function checkLocal(){
  // Check local storage for objects index
  console.log('Could not find lists index in cloud.');
  var temp = readLocalStorage('objKeysBSL');
  if ((typeof temp !== 'undefined')&&(temp.length > 0)) {
    console.log('Found local lists index.')
    saveS3(temp, (prefix + '/' + 'objKeysBSL.json'))
    // Start the sync
    syncS3Local(temp, true);
  }
  else{
    // No shopping lists local either
    hideSpinner();
    console.log('Could not find lists index in local storage.');
    growl('warning','Finished','Nothing to sync');
    userSync = true;
  }
}
```

Change syncS3Local to:

```
function syncS3Local(objKeyS3, emptyS3){
  console.log('Starting to sync cloud to local storage...');
  var tempKey, tempLocal, TempS3, dateLocal, dateS3, params;
  async.each(objKeyS3, function (value) {
    tempKey = value.key;
    // Get object lastUpdated date from localStorage
    tempLocal = readLocalStorage(value.key);
    if(typeof tempLocal === 'undefined'){
      dateLocal = 0;
    }
    else{
      dateLocal = new Date(tempLocal.lastUpdated);
    }
    dateS3 = new Date(value.lastUpdated);
    if (dateLocal < dateS3){
      // Local storage is old update
      // Get object from S3
      params = {
                   Bucket: bucketName,
                   Key: value.key
                 };
                 S3.getObject(params, function(err, data) {
        if (err){
          console.log('Error: Could not sync ' + value.key);
          console.log(err, err.stack);
        }
        else  {
          TempS3 = JSON.parse(data.Body);
          // Save to local storage
          console.log('Synced: ' + value.key);
          saveLocalStorage(TempS3,value.key);
        }
      });
    }
    else if ((dateLocal > dateS3)||(emptyS3)){
      // Local storage is newer update
      // Save local object to S3
      var temp = JSON.stringify(tempLocal);
      params = {
          Bucket: bucketName,
          Key: value.key,
          ContentType: 'json',
          Body: temp
      };
      console.log('Saving data to S3 at: '+ JSON.stringify(value.key));
      S3.putObject(params, function (err, data) {
          if (err) {
              console.log('Error saving to cloud: ' + err);
              growl('danger','Error.','Unable to save data to cloud.');
          } else {
            growl('success','Finished','Data saved to cloud.');
          }
      });
    }
  }, function (err) {
    if (err) console.error('Async.js error: ' + err.message);
  });
  loadModals();
  hideSpinner();
  growl('success','Finished','Sync local database with cloud.');
  userSync = true;
}
```

Now save and upload to S3

Now run the application and sync to the cloud.

No run the command localStorage.clear() to clear the local storage

No run the sync again and check the local storage has been restored.