**Tweetaspike: Hands-On Lab Exercises**

**GitHub URL**: github.com/aerospike-edu/tweetaspike-workshop

**Clone URLs**:

* **SSH**: git@github.com:aerospike-edu/tweetaspike-workshop.git
* **HTTPS**: github.com/aerospike-edu/tweetaspike-workshop.git

**Lab 1 Overview**

In this lab, we will focus on writing code to:

* Connect to Aerospike Cluster
* Write User Record
* Read User Record

**Lab 1 Exercises**

In the project, locate **/lib/controllers/aerospike\_config.js** and update \*\***aerospikeCluster**\*\* with your IP address

In the project, locate **/lib/controllers/api.js** and add code to:

1. Connect to Aerospike Cluster

* Lab:1:TODO:1
* Lab:1:TODO:2

1. Write User Record - In *exports.createUser()*

* Lab:1:TODO:3
* Lab:1:TODO:4
* Lab:1:TODO:5

1. Read User Record - In *exports.checkUsername()*
   * Lab:1:TODO:6
   * Lab:1:TODO:7

**Lab 1 Answers**

In File: /lib/controllers/api.js

**1) Connect to Aerospike Cluster**

**// Lab 1:TODO 1: Connect to the Aerospike cluster**

***client.connect****(function (response) {*

*if ( response.code === 0) {*

*// handle success*

*console.log("\nConnection to Aerospike cluster succeeded!\n");*

*}*

*else {*

*// handle failure*

*// console.log(response);*

*console.log("\nConnection to Aerospike cluster failed!\n");*

*process.exit(0);*

*}*

*});*

**// Lab 1:TODO 2: Close connection to Aerospike cluster**

*if (client != null) {*

***client.close();***

*}*

In File: /lib/controllers/api.js

**2) Write User Record**

Code In: exports.createUser()

**// Lab 1:TODO 3: Create Aerospike key object**

*var key =* ***aerospike.key****(aerospikeDBParams.dbName,aerospikeDBParams.usersTable,params.uid);*

**// Lab 1:TODO 4: Create an object representing user record using parameters passed in**

*var userRecord = {uid: params.uid, username: params.username, password: params.password, auth: params.auth, tweetCount: 0};*

**// Lab 1:TODO 5: Write user record in the Aerospike database using put operation. If the put operation succeeds, return status 'Ok.' Otherwise return error**

***client.put****(key, userRecord, function(err, rec, meta) {*

*// Check for errors*

*if ( err.code === aerospike.status.AEROSPIKE\_OK ) {*

*// The record was successfully created.*

*// console.log(rec, meta);*

*res.json({status : 'Ok'});*

*}*

*else {*

*// An error occurred*

*console.error('createUser error: ', err);*

*res.json({status: err});*

*}*

*});*

In File: /lib/controllers/api.js

**3) Read User Record**

Code In: exports.checkUsername()

**// Lab 1:TODO 6: Create Aerospike key object**

*var key =* ***aerospike.key****(aerospikeDBParams.dbName,aerospikeDBParams.usersTable,params.username);*

**// Lab 1:TODO 7: Read user record from the Aerospike database using get operation. If the get operation succeeds, return status 'Ok' along with uid and auth bins from the record. Otherwise return error.**

***client.get****(key, function(err, rec, meta) {*

*// Check for errors*

*if ( err.code === aerospike.status.AEROSPIKE\_OK ) {*

*// The record was successfully read.*

*// console.log(rec);*

*res.json({status : 'Ok', uid : rec.uid, auth: rec.auth});*

*}*

*else {*

*// An error occurred*

*console.error('checkUsername error:', err);*

*res.json({status : 'Invalid Username'});*

*}*

*});*

**Run Application**

In the terminal window, browse to the application folder and run command **node server.** Then browse to [**http://localhost:9000**](http://localhost:9000)

**Lab 2 Overview**

In this lab, we will focus on writing code to:

* Create Secondary Index
* Execute Query Based On Secondary Index
* Process Result

**Lab 2 Exercises**

In the project, locate **/lib/controllers/aerospike\_config.js** and update \*\***aerospikeCluster**\*\* with your IP address

In the project, locate **/lib/controllers/api.js** and then in **exports.retrieveTweetsUsingIndex(),** add code to:

1. Create Secondary Index

* Lab:2:TODO:1
* Lab:2:TODO:2

1. Execute Query Based On Secondary Index

* Lab:2:TODO:3
* Lab:2:TODO:4
* Lab:2:TODO:5

1. Process Result

* Lab:2:TODO:6

**Lab 2 Answers**

In File: /lib/controllers/api.js

**1) Create Secondary Index**

Code In: exports.retrieveTweetsUsingIndex()

**// Lab 2:TODO 1: Setup Secondary Index options object with namespace, set, bin to be indexed and index name**

*var options = {*

*ns: aerospikeDBParams.dbName,*

*set: aerospikeDBParams.tweetsTable,*

*bin : 'username',*

*index: 'username\_index'*

*};*

**// Lab 2:TODO 2: Create Secondary Index based on the options object created above. NOTE: In production environment, this should be done via aql**

***client.createStringIndex****(options, function(err) {*

*if ( err.code === aerospike.status.AEROSPIKE\_OK ) {*

*// index created*

*} else {*

*// An error occurred*

*console.error('createIndexOnUsername error: ', err);*

*}*

*});*

In File: /lib/controllers/api.js

**2) Execute Query Based On Secondary Index**

Code In: exports.retrieveTweetsUsingIndex()

**// Lab 2:TODO 3: Setup query statement with namespace, set, filter predicate and bins to retrieve**

*var statement = {filters:[aerospike.filter.equal('username', uid)], select: ['key','tweet','ts']};*

**// Lab 2:TODO 4: Create query object with namespace, set and statement created above**

*var query =* ***client.query****(aerospikeDBParams.dbName,aerospikeDBParams.tweetsTable, statement);*

**// Lab 2:TODO 5: Execute query**

*var stream =* ***query.execute();***

In File: /lib/controllers/api.js

**3) Process Result**

Code In: exports.retrieveTweetsUsingIndex()

**// Lab 2:TODO 6: Add 'data', 'error' and 'end' event handlers to process query result**

*var tweets = {};*

***stream.on****(****'data'****, function(record) {*

*// console.log(record.bins.tweet);*

*tweets[record.bins.ts] = {tweet: record.bins.tweet, key: record.bins.key, uid: uid, ts: record.bins.ts};*

*});*

***stream.on****(****'error'****, function(err) {*

*console.log('retrieveTweetsUsingIndex Error: ',err);*

*res.json({status : err});*

*});*

***stream.on****(****'end'****, function() {*

*console.log('retrieveTweetsUsingIndex:\n', tweets);*

*res.json({status : 'Ok', tweets: \_this.sortTweets(tweets,uid)});*

*});*

**Run Application**

In the terminal window, browse to the application folder and run command **node server.** Then browse to [**http://localhost:9000**](http://localhost:9000)

**Lab 3 Overview**

In this lab, we will focus on writing code to:

* Setup Server “tweet” Event Listener
  + This event is triggered from connected clients when a new “tweet” gets created
* Emit Server “broadcast” Event
  + This event notifies connected clients of new tweets
* Emit Client “tweet” Event
  + This event is triggered when a new “tweet” gets created
* Setup Client “broadcast” Event Listener
  + This event is triggered from server when a new “tweet” gets created by a connected client

**Lab 3 Exercises**

In the project, locate **/lib/controllers/aerospike\_config.js** and update \*\***aerospikeCluster**\*\* with your IP address

In the project, locate **server.js** and **/app/scripts/controllers/main.js** and add code to:

1. Setup Server “tweet” Listener - In *server.js*

* Lab:3:TODO:1

1. Emit Server “broadcast” Event – In *server.js*

* Lab:3:TODO:2

1. Emit Client “tweet” Event – In *main.js*

* Lab:3:TODO:3

1. Setup Client “broadcast” Event Listener - In *main.js*

* Lab:3:TODO:4

**Lab 3 Answers**

In File: server.js

**// Lab 3:TODO 1: Setup Server “tweet” Event Listener**

***socket.on****(****'tweet'****, function (data) {*

*//use io.sockets.emit to send a message to all clients with the message category of 'broadcast' -*

*//which is our key to know on the client-side that a new message is being broadcasted*

**// Lab 3:TODO 2: Emit Server “broadcast” Event**

***io.sockets.emit****(****'broadcast'****, data);*

*});*

In File: /app/scripts/controllers/main.js

**// Lab 3:TODO 3: Emit Client “tweet” Event**

Code In: $scope.createTweet()

***socket.emit****(****'tweet'****,{uid: uid, tweet: tweetObject.tweet, realTweet: true});*

**// Lab 3:TODO 4: Setup Client “broadcast” Event Listener**

***$scope.$on****(****'socket:broadcast'****, function (event,data) {*

*if ((data.realTweet || $scope.alertStatus) && $scope.uid != data.uid)*

*{*

*var tweetObject = {key: data.uid+':'+($scope.myFollowingTweets.length+1),tweet: data.tweet, ts: (new Date).getTime(), uid: data.uid};*

*$scope.myFollowingTweets.unshift(tweetObject);*

*}*

*});*

**Run Application**

In the terminal window, browse to the application folder and run command **node server.** Then browse to [**http://localhost:9000**](http://localhost:9000)