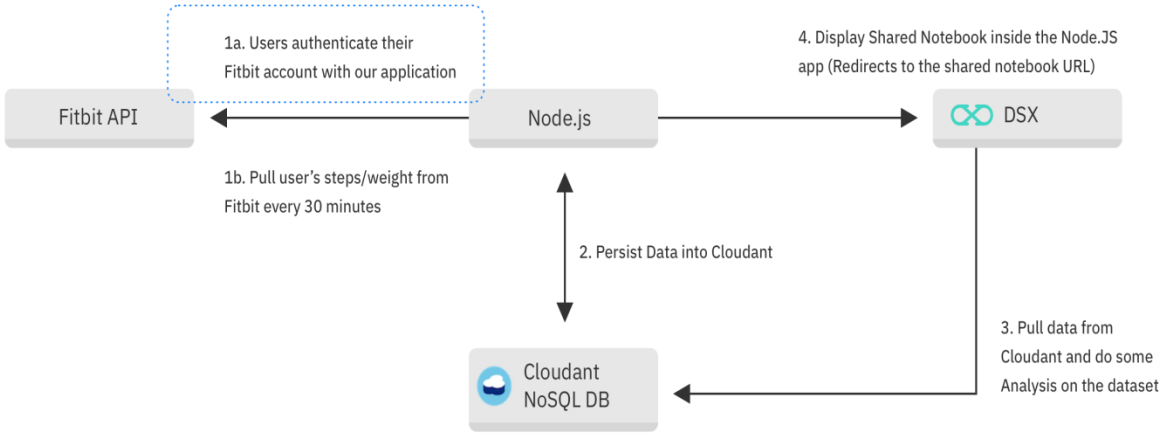
TRAIN ON IBM MODEL

Basic Fitness App Architecture:



Step 1: Blue mix

Create a Blue mix trial account.

The other services used to create the app will all be provisioned and managed within the Blue mix environment. We recommend that you use the standard plan to see ultimate performance benefits.

Step 2: Node.js

Once logged into your Blue mix account, provision the Node.js Cloud Foundry app.

This is how we'll host your app.

Step 3: Cloud ant

Next, provision the IBM Cloud ant Nasal database.

Cloud ant will be used to store persistent data, such as user steps and relative weight.

Step 2: Node.js

Once logged into your Blue mix account, provision the Node.js Cloud Foundry app.
This is how we'll host your app.

Step 3: Cloud ant

Next, provision the IBM Cloud ant Nasal database.
Cloud ant will be used to store persistent data, such as user steps and relative weight.

Setup your .envy file

```
CLOUDANT_ACCOUNT=YOUR_CLOUDANT_URL
CLOUDANT_API_KEY=YOUR_CLOUDANT_API_KEY
CLOUDANT_PASSWORD=YOUR_CLOUDANT_PASSWORD
FITBIT_CLIENT_ID=YOUR_FITBIT_CLIENT_ID
FITBIT_CLIENT_SECRET=YOUR_FITBIT_CLIENT_SECRET
FITBIT_VERIFICATION_CODE=YOUR_FITBIT_SUBSCRIPTION_VERIFICATION_CODE
```

Register Fit bit Users

In order for us to pull data from Fit bit, we first need to setup a callback endpoint for users to register with our application and a subscription to notify when users data needs to be updated. You can view the Fit bit's documentation at Fitbit OAuth. Once you've created your SDK for Node.js app, you should have a FQDN from Blue mix. (https://<unique_hostname>.mybluemix.net).
Next, we will need to create an express server with an endpoint to handle the Oath callback from Fit bit and subscription notifications.

app.js

```
 1import express from "express";
 2import body Parser from "body-parser";
 3import path from "path";
 4import cent from "cent";
 5import { logger } from "./logger";
 6
 7import home from "./controllers/home";
 8import fit bit from "./controllers/fit bit";
 9import registered from "./controllers/registered";
10
11let app = new express();
12
13// Middleware to handle application/son and applicant/x-www-form-urlencoded
14app. use(bodyParser.json());
15app. use(bodyParser.urlencoded({ extended: false }));
16app. set("views", path. Join(__dirham, "public"));
17
18app. set("view engine","ejs");
19
20app. use(express. Static(__dirham + "/public"));
21// Middleware routes
22app. use("/", home);
23app. use("/fit bit", fit bit);
24app. use("/registered", registered);
25
26const append = cfenv.getAppEnv();
27const PORT = appEnv.port || 3000;
28
29
```

```
30// start server on the specified port and binding host
31app. listen(PORT, "0.0.0.0", function() {
32    logger.log("info", "server starting on " + appEnv.url);
33});
```

Add Persistence Using Cloud ant

When a user registers with our application, we want to make sure we are persisting their access token and refresh token so we can pull the user's data later on. To do this, we will use Cloud ant on IBM Blue mix. But before we can write code to insert data into Cloud ant, we need to create the databases.

Launch the Cloud ant dashboard and create a weight, steps, and users database. (If you are unsure on how to create a database, you can view the documentation here.) You will also want to create an index to optimize query lookup, which can also be found in the documentation. We will now create a wrapper around the Cloud ant client to handle persisting steps, weights, and user's data into our Cloud ant database.

cloudant.js