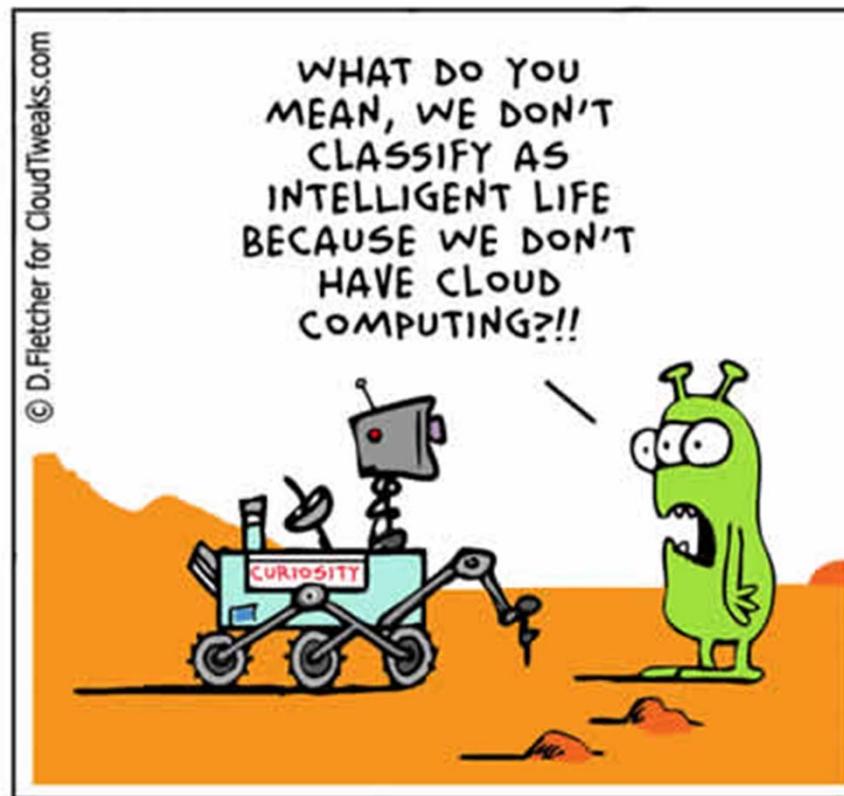


P02: IBM Cloud Services (PaaS, SaaS) - Lab



IBM Data Science Experience

Outline

- Objective
- Bluemix
 - High-Level Architecture
 - Bluemix Cloud Foundry
 - Watch video – Demo: Getting Started with Node.js on Bluemix 2016
- Examples
 - Web Application to Query in-Cloud Database
 - Web Application with Watson Personality Insights Service
 - In-cloud Data Analytics using Simple SQL and R Scripting
- Summary

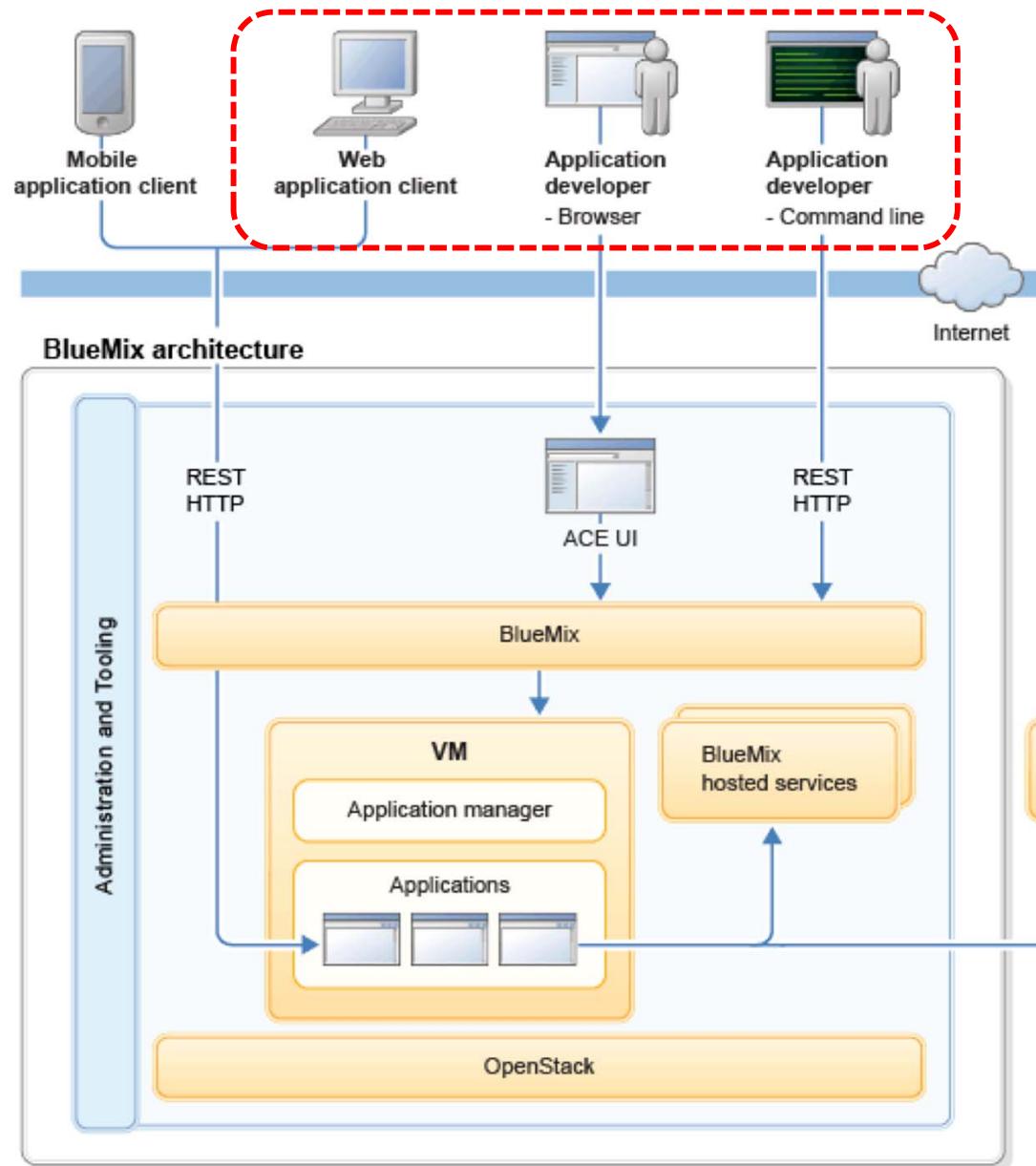
Objective

- develop a simple SaaS application using IBM Bluemix PaaS and SaaS services
- Example 1: A web server application for querying an in-cloud database
 - setting up an in-cloud database SaaS (Db2) and load data from the local computer
 - develop a simple SaaS web server application (using Node.js boilerplate PaaS) and linking with the in-cloud database SaaS
- Example 2: A web server application for generating personality insights using cognitive analysis SaaS
 - perform cognitive analysis using Watson service from a Node.js web server application and retrieve results
- Example 3: Using SaaS services for data analytics on cloud
 - use IBM Db2 Data Warehousing and DSX SaaS services for in-cloud data analytics with SQL and R scripting languages

Milestones

1. Successful login
2. Install CLI
3. Load web app
4. Load Watson web app
5. Run SQL query
6. Run R script and visualize data

Bluemix High-level Architecture



Application developer interacts with systems through:

1. Command line interface
2. Browser interface

Bluemix Cloud Foundry

- Users focus on application code, i.e., don't have to worry about the OS and infrastructure layers
- Cloud Foundry is an open source PaaS
 - Corporate users: SAS, Cisco, Rakuten, Baidu, SAP Verizon, ..
 - Leverages on the broad community
 - Push, extend and manage applications using
 - a command line tool – Bluemix CLI implements Cloud Foundry operations
 - eclipse plugin
 - DevOps ...
 - Addresses PaaS lack of cross-compatibility among different cloud providers

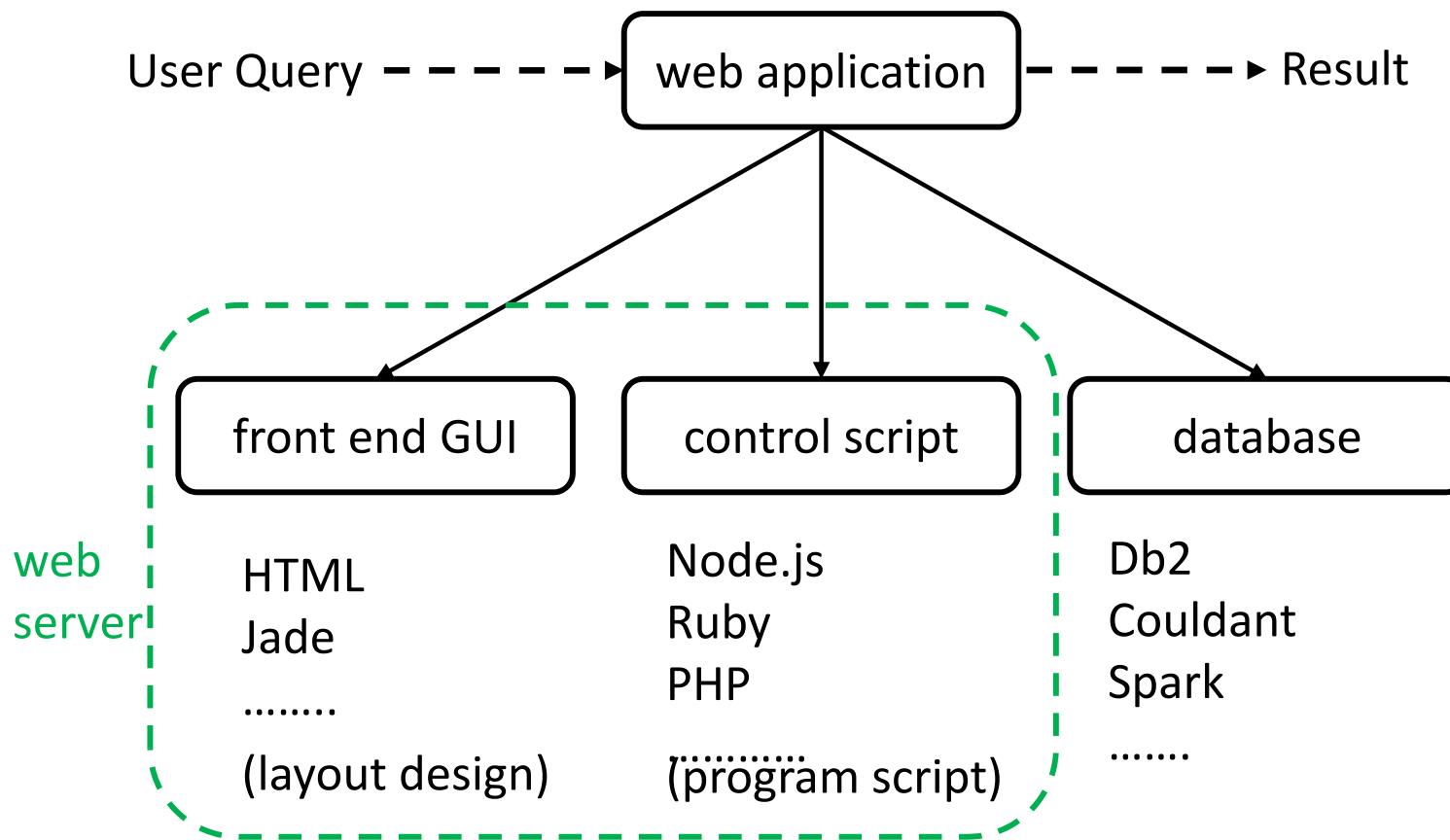
Video

[Demo: Getting Started with Node.js on Bluemix 2016](#)

DEVELOPING A WEB APPLICATION

- Objective: set up a web server application using to retrieve data from an in-cloud database and display on a browser
- Example: Web application to retrieve and display economic data of countries stored in a IBM Db2 database, running on a cloud-based web server using node.js boilerplate
- Services can be added on top of the boilerplate runtime (eg. database service)
- Data file: World bank economic statistics

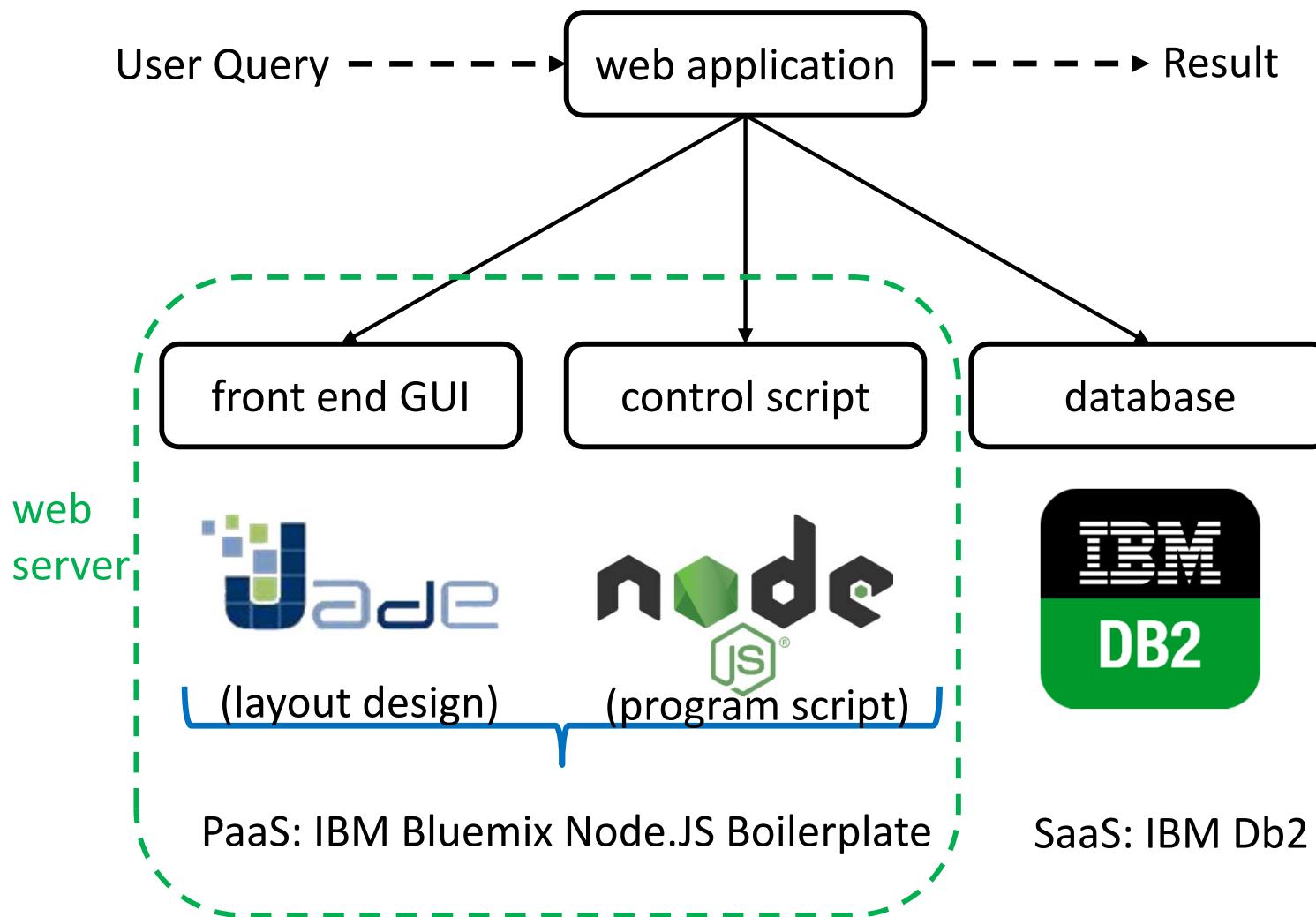
Developing a Web Application



Different technologies are offered by IBM Bluemix cloud

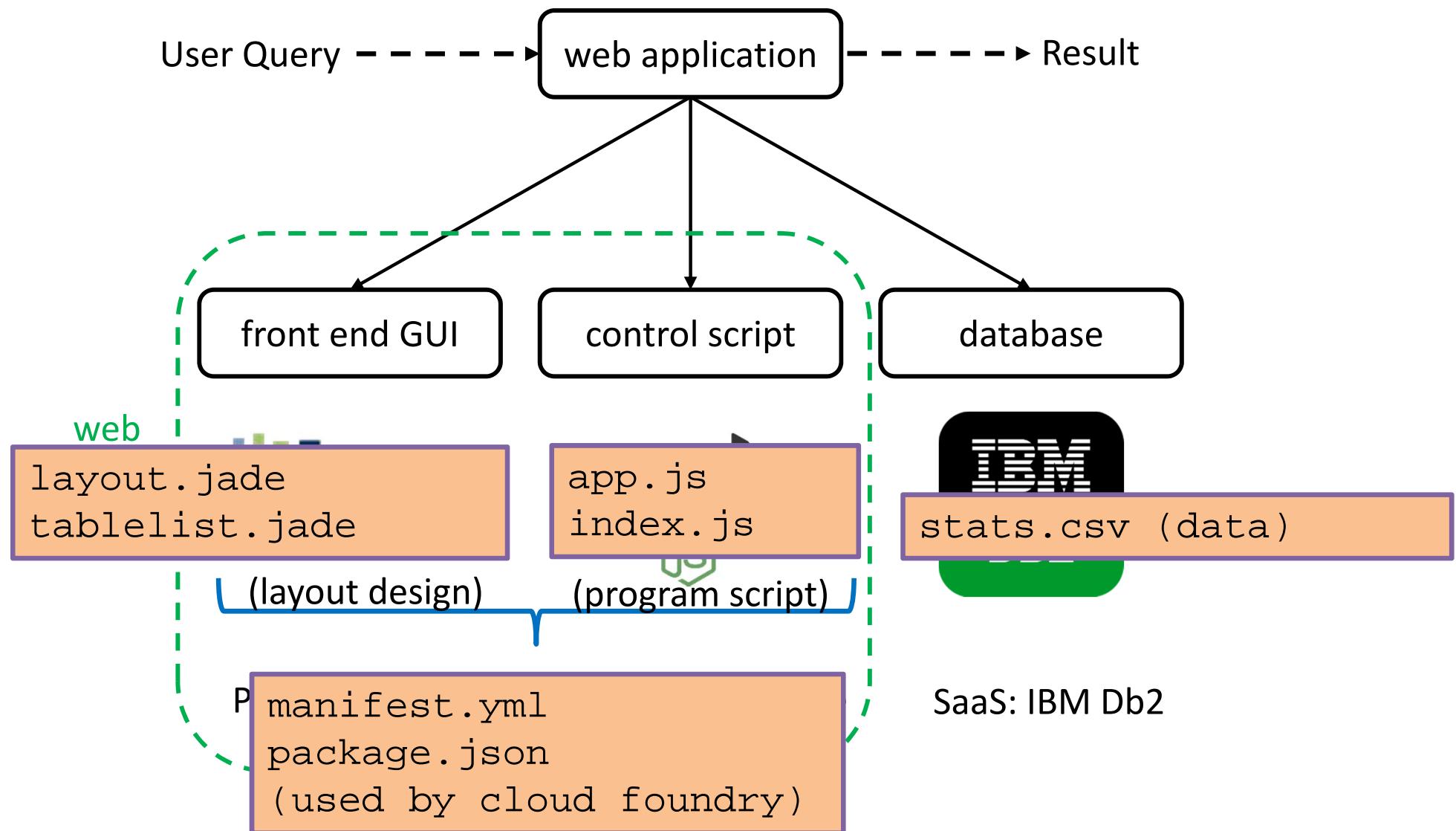
Developing a Web Application

What we are going to use for Example 1:

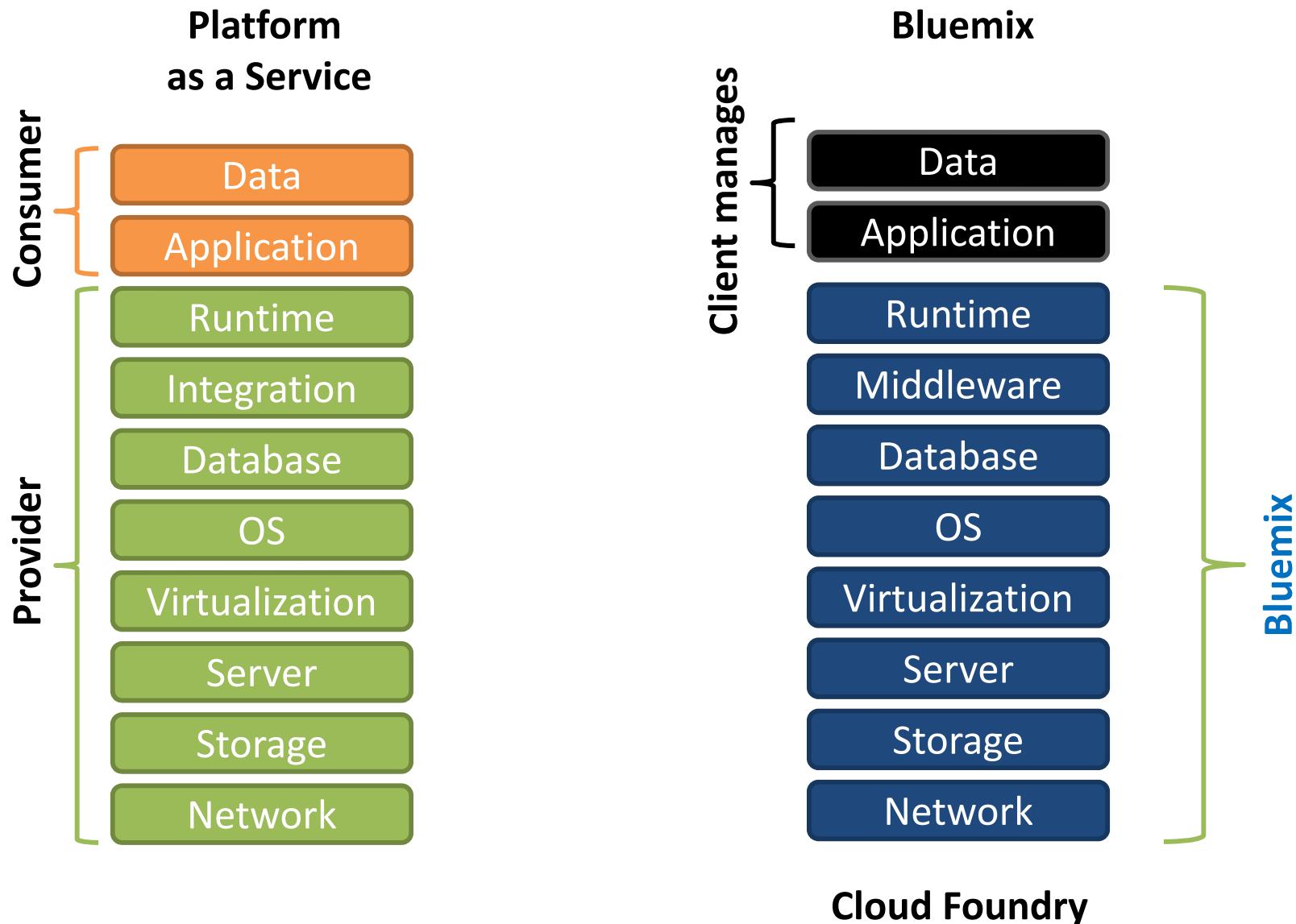


Developing a Web Application (relevant files)

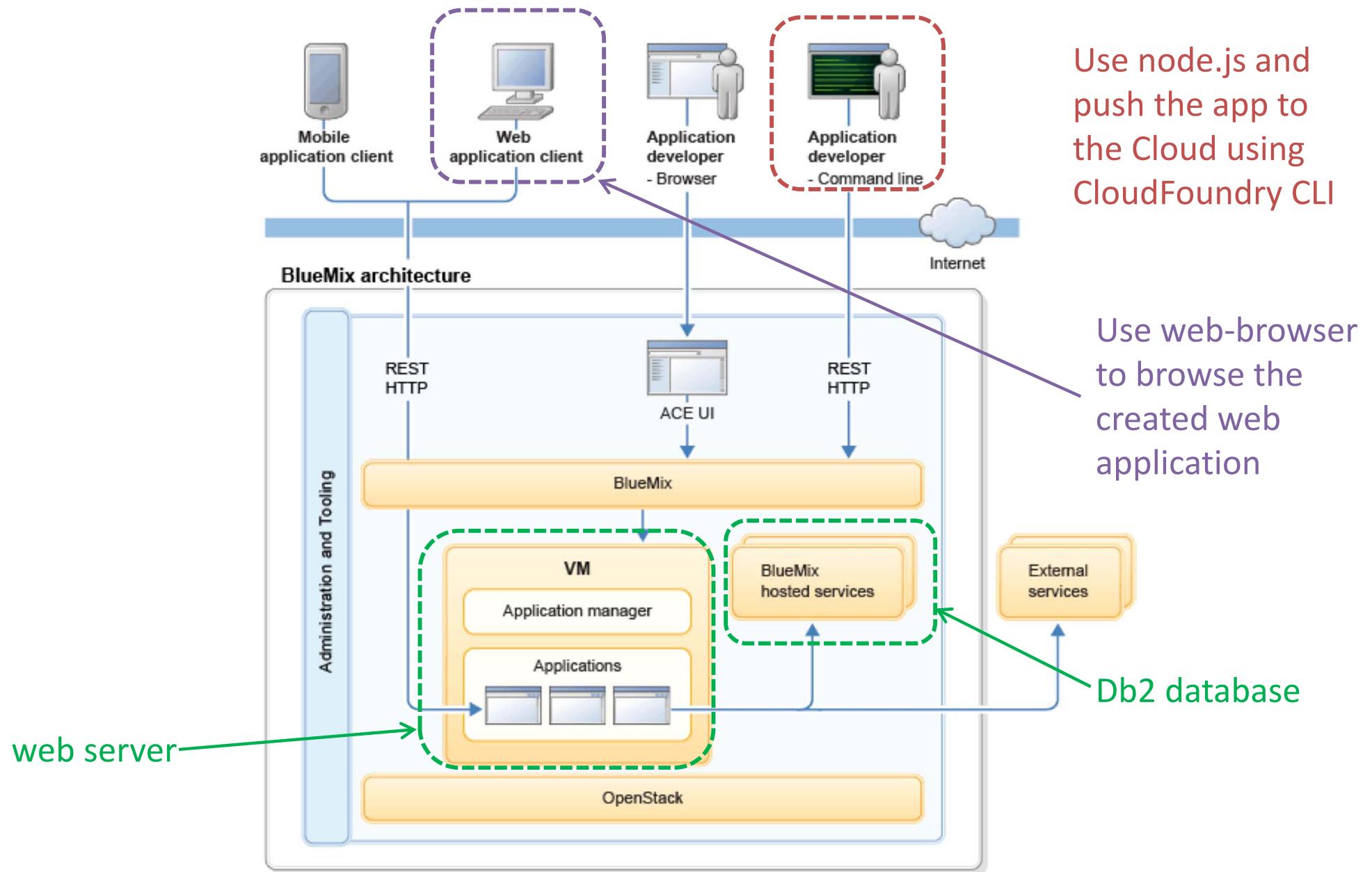
What we are going to use for Example 1:



What is PaaS?



Web Application Architecture



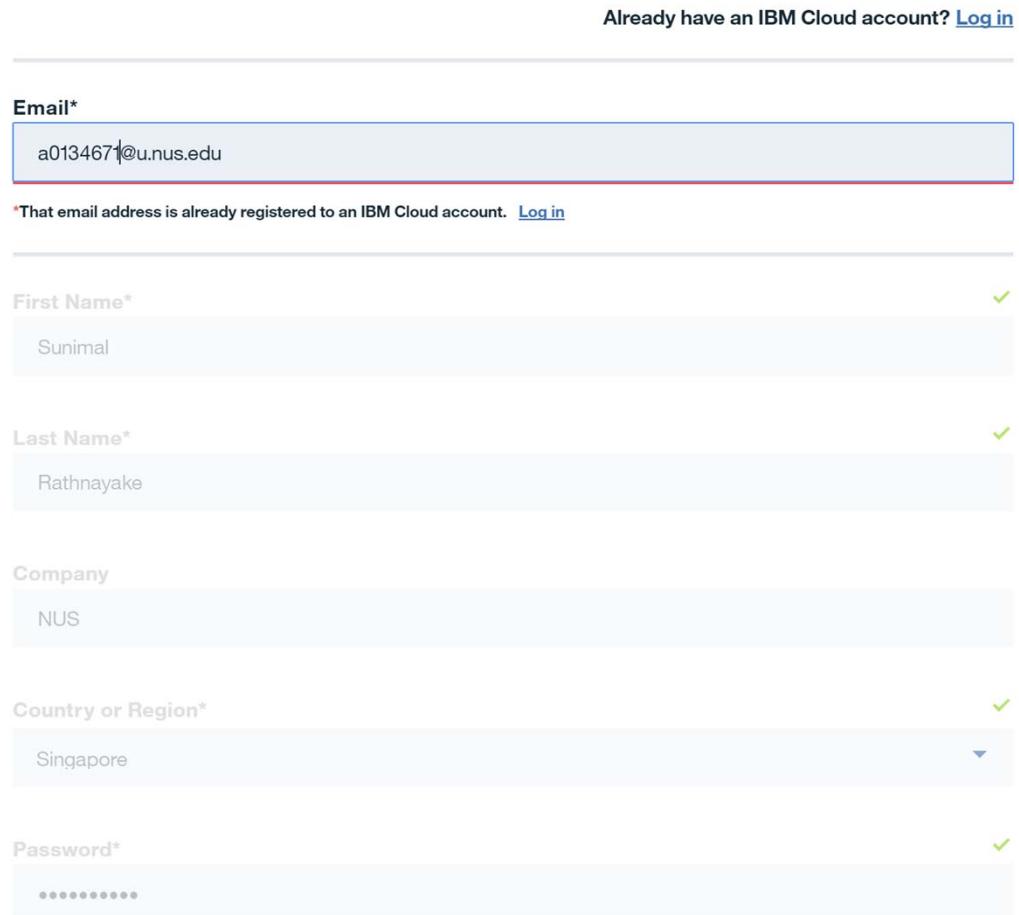
Main Steps: Web Application using Node.js

1. Sign up for IBM Bluemix
2. Select Db2 for Transactions from the catalog – only necessary if the application needs a database
3. Upload data to the newly created Db2 database create necessary tables
4. Modify/review data fields and get information necessary for remote access
5. Select Cloud Foundry Node.js app from the catalog – this is needed for every new application
6. Provide a hostname that serves as the web application URL – hostname.mybluemix.net
7. Install the Bluemix CLI (command line interface) client on your local machine - this is required if you have not done so previously
8. Add a connection to the database created in steps 2-4
9. To start a new web application, download the code (index.js, manifest.yml, app.js, package.json,..) onto your local directory
10. Modify manifest.yml with database host name
11. Deploy/upload application on Bluemix. Using the Bluemix CLI, connect the Bluemix client to the selected Bluemix region (server)
12. Using the Bluemix CLI, login to Bluemix and, upload your application to Bluemix!
13. To access the web application, enter your host URL (in step 5)

Step 1: Signing up

- IBM Cloud sign up
<https://console.bluemix.net/>

- Fill in the form with your details.
Make sure to use the NUS school email.



The screenshot shows the IBM Cloud sign-up form. At the top right, there is a link to log in if you already have an account. The form fields are as follows:

- Email***: a0134671@u.nus.edu - A red border surrounds this field, and a small error message below it states: "That email address is already registered to an IBM Cloud account. [Log in](#)".
- First Name***: Sunimal - This field has a green checkmark to its right.
- Last Name***: Rathnayake - This field has a green checkmark to its right.
- Company**: NUS - This field has a green checkmark to its right.
- Country or Region***: Singapore - This field has a green checkmark to its right.
- Password***: A series of six dots representing a password.

Milestone 1: Successful Login

- Did you successfully log in to IBM Bluemix?
- Let's wait until everyone is done!
- In the meantime:
 - Explore the array of services offered in Bluemix

IBM Cloud Platform – Bluemix

The screenshot shows the IBM Cloud Platform - Bluemix dashboard at <https://console.bluemix.net/dashboard/apps/>. The dashboard is titled "Dashboard" and includes filters for "REGION US South", "CLOUD FOUNDRY ORG a0134671@u.nus.edu", and "CLOUD FOUNDRY SPACE dev". A search bar and a "Create resource" button are also present.

Cloud Foundry Services (3/10 Used)

Name	Service Offering	Plan
a0134671-db	Db2 Warehouse	Entry
DSX	Data Science Experience	Lite
Spark-lm	Apache Spark	Lite

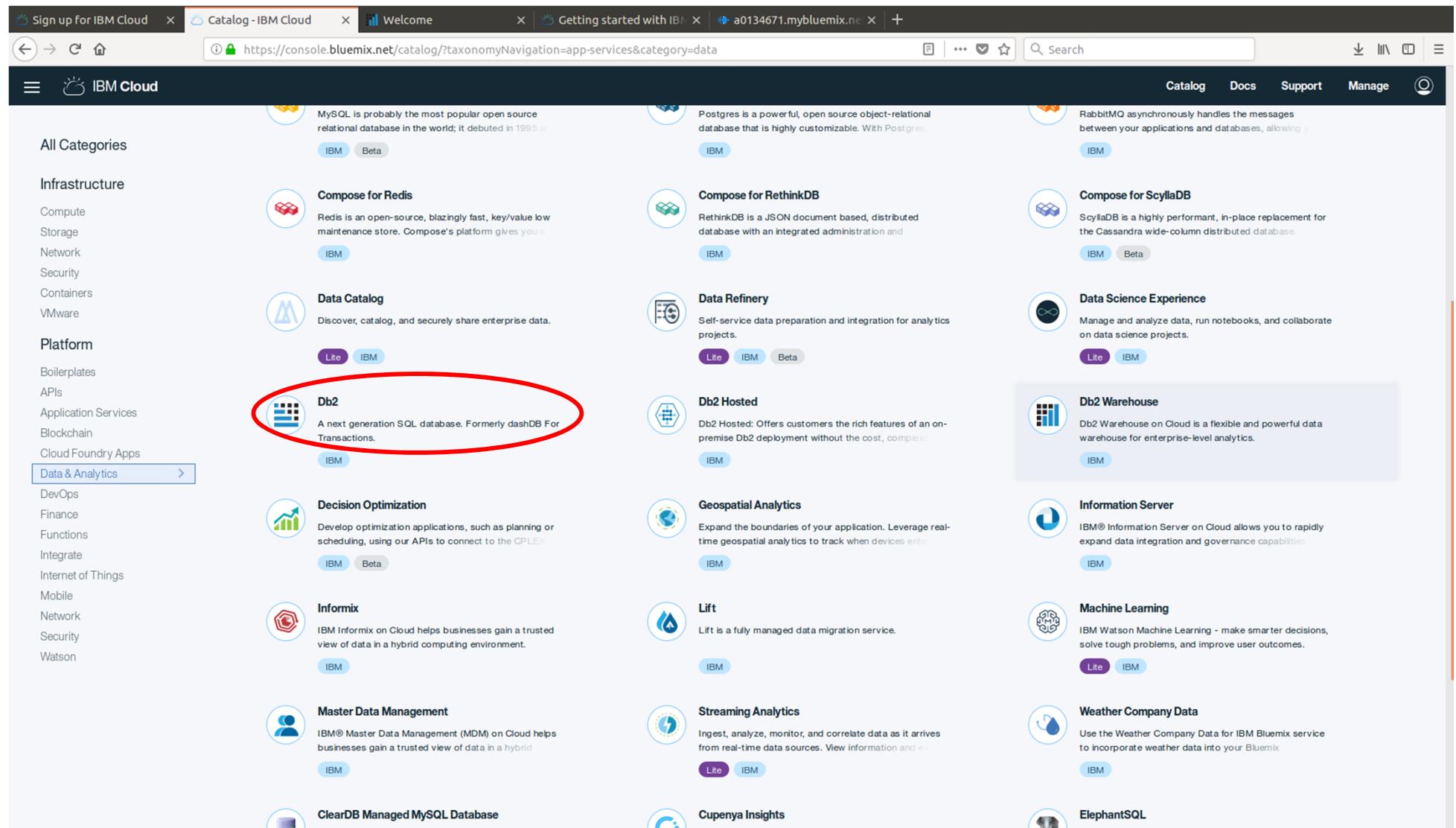
Services (1)

Name	Resource Group	Service Offering	Plan	Details
cloud-object-storage-nb	Default	Cloud Object Storage	Lite	Provisioned

How to instantiate Bluemix services?

- Two ways to instantiate a Bluemix service
 1. Using the web browser interface (Bluemix web console)
 - eg. select Db2 on the web interface and do the setup through the web browser.
 - created service can be later connected with other application such as web applications
 - [we demonstrate this in Example 1 \(Db2 service\)](#)
 2. Using the command line interface (IBM Cloud CLI)
 - eg. issue a command through the CLI to instantiate a service (managed by cloud foundry software)
 - [we demonstrate this in Example 2 \(Watson personality insights service\)](#)

Step 2: Select Db2 from Catalog



The screenshot shows the IBM Cloud Catalog interface. On the left, there's a sidebar with categories like All Categories, Infrastructure, Platform, and Data & Analytics (which is currently selected). The main area lists various database services. One service, 'Db2', is circled in red. Other visible services include MySQL, PostgreSQL, RabbitMQ, Compose for Redis, Compose for RethinkDB, Compose for ScyllaDB, Data Catalog, Data Refinery, Data Science Experience, Db2 Hosted, Db2 Warehouse, Decision Optimization, Geospatial Analytics, Information Server, Machine Learning, Master Data Management, Streaming Analytics, Weather Company Data, ClearDB Managed MySQL Database, Copenya Insights, and ElephantSQL. Each service has a brief description, an icon, and status indicators (IBM, Lite, Beta).

Step 2: Db2 Setup

The screenshot shows the IBM Cloud service catalog interface for creating a Db2 Warehouse. The top navigation bar includes tabs for 'Sign up for IBM Cloud', 'Mail - sunimalr@u.nus.edu', and 'Db2 Warehouse - IBM Cloud'. The main content area is titled 'Db2 Warehouse'.

Service name: Db2 Warehouse-y9

Choose a region/location to deploy in: US South

Choose an organization: a0134671@u.nus.edu

Choose a space: dev

Features

- Fully managed, safe, and secure**
Your Db2 Warehouse on Cloud instance is managed, monitored, encrypted and backed-up by IBM, so you can focus on gaining insights from your data instead of administering it.
- Highly Available (HA) with MPP**
Your data workloads, queries and dashboards will all continue to run, even during unexpected failure.
- Compatible with Netezza and Oracle**
Most of your Netezza and Oracle workloads will migrate seamlessly to Db2 Warehouse on Cloud. For the select few edge cases, we offer free tooling to help you make a smooth transition.
- Related Db2 cloud products**
Check out Db2 on Cloud, the enterprise-class, high-performance transactional data store.
- Built for the hybrid cloud**
Our unified architecture enables a range of hybrid use cases. Easily move data between your on-premises data stores and Db2 Warehouse on Cloud. Access data & run SQL queries across multiple heterogeneous data sources with IBM Fluid Query.

Images

Click an image to enlarge and view screen captures, slides, or videos. Screen caps show the user interface for the service after it has been provisioned.

Need Help? [Contact IBM Cloud Sales](#)

Estimate Monthly Cost [Cost Calculator](#)

Create

Step 2: Open Db2 Console

The screenshot shows the IBM Cloud service details page for a Db2 Warehouse on Cloud service named 'a0134671-db'. The service is located in the US South region, associated with the org 'a0134671@u.nus.edu' and space 'dev'. A red oval highlights the 'OPEN' button, which is used to access the Db2 console. Below the main section, there are sections for 'Data Movement' and 'Connect Your Applications', and a 'Where to Start' section with 'Learn' and 'Open' options.

Manage / Data & Analytics / a0134671-db

Location: US South Org: a0134671@u.nus.edu Space: dev

Db2 Warehouse on Cloud

When you open the console, you can connect to the service, upload your data, and run analytics from the cloud.

Data Movement
Upload locally from your computer, or set up remote jobs from various sources such as Soltlayer Swift, IBM Cloudant, or Amazon S3.

Connect Your Applications
After you have your data in place, you can connect your business intelligence or analytics-focused applications, and start running queries.

Where to Start

Learn
Open

Learn what you can do with Db2 Warehouse on Cloud
Open the console to get started with Db2 Warehouse on Cloud today!

Step 3: Upload data

- First download stats.csv data file from the course webpage, then upload to Db2

The image displays two side-by-side screenshots of the IBM Cloud Services dashboard, specifically the Db2 service details page.

Left Screenshot: This shows the main dashboard for the Db2 service. It features a timeline at the top with dates from 1/20 to 1/25. Below the timeline is a table with columns: SOURCE, FILENAME, TARGET, REQUESTED BY, ROWS LOADED, and ROWS REJECTED. A message "You have no recent load jobs" is displayed. A red oval highlights the "Load data" button at the bottom of the table.

Right Screenshot: This shows the "Load new" wizard. The first step, "Source", is active. It includes a "selection" section with a "Browse files" button, which is circled in red. The interface has tabs for Source, Target, Define, and Finalize.

Step 4: Select Scheme, Table Name and, Review Table

- Create a new table “STATS”

The image displays two side-by-side screenshots of the IBM DB2 Warehouse on Cloud interface, specifically the 'Select a load target' step. Both screenshots show a 'Schema' dropdown menu with several options listed. In the left screenshot, the option 'DASH6649' is highlighted with a red circle. In the right screenshot, the option 'EDSTATS' is highlighted with a red circle. The 'Table' dropdown menu below the schemas also lists several options, with 'EDSTATS' being the selected choice in both cases.

To connect database with another app, we need to note the “scheme” and the “tables” name.

Step 4: Select Scheme and Table Name

- Preview
 - You can change the column names if necessary
- Review settings
- Click Begin Load

Note the table name.
We need this in next steps

You are loading the file *EdStats-Country.csv* into **DASH6649.EDSTATS**

	COUNTRY CODE	SHORT NAME	TABLE NAME	LONG NAME	2-ALPHA CODE	CURRENCY UNIT	SPECIAL NOTES
1	AFG	Afghanistan	Afghanistan	Islamic State of Afghanistan	AF	Afghan afghani	Fiscal year end: March 20; reporting period for national accounts data: FY (from 2013 are CY)
2	ALB	Albania	Albania	Republic of Albania	AL	Albanian lek	
3	DZA	Algeria	Algeria	s Democratic Republic of Algeria	DZ	Algerian dinar	
4	ASM	American Samoa	American Samoa	American Samoa	AS	U.S. dollar	
5	ADO	Andorra	Andorra	Principality of Andorra	AD	Euro	
6	AGO	Angola	Angola	s Republic of Angola	AO	Angolan kwanza	April 2013 database update: Based on IMF data, national accounts data were revised for 2000
7	ATG	Antigua and Barbuda	Antigua and Barbuda	Antigua and Barbuda	AG	East Caribbean dollar	April 2012 database update: Based on official government statistics, national accounts data w.

Step 5: Select Cloud Foundry Node.js

- Go back to catalog and select SDK for Node.js under CloudFoundry apps

The screenshot shows the IBM Cloud Catalog interface. On the left, there's a sidebar with 'All Categories' and sections for Infrastructure (Compute, Storage, Network, Security, Containers, VMware), Platform (Boilerplates, APIs, Application Services, Blockchain), and Cloud Foundry Apps. The 'Cloud Foundry Apps' section is currently selected, indicated by a blue border around its link. The main content area displays various runtime environments: Liberty for Java™, SDK for Node.js™ (which is circled in red), Runtime for Swift, XPages, Go, PHP, Python, Ruby, and Tomcat. Each entry includes a brief description and an 'IBM' badge.

Step 6: Provide required app details

Use a unique app name and hostname.
I use my NUSNET id.

The screenshot shows the 'Create a Cloud Foundry App' page for the 'SDK for Node.js™' service. The 'App name:' field contains 'a0134671-app' and the 'Host name:' field contains 'a0134671', both of which are circled in red with arrows pointing to them from the text above. The 'Domain:' dropdown is set to 'mybluemix.net'. Deployment settings include 'US South' region, organization 'a0134671@u.nus.edu', and space 'dev'. The 'Pricing Plans' section shows a single plan: 'Default', which runs one or more apps free for 30 days (375 GB-hours free) at a cost of \$0.0735 USD/GB-Hour. A note states: 'This is a service plan for the IBM Bluemix Platform runtime.' Navigation links at the bottom include 'Need Help?', 'Contact IBM Cloud Sales', 'Estimate Monthly Cost', 'Cost Calculator', '\$0.0735 USD/GB-Hour', and a 'Create' button.

SDK for Node.js™

Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides enhanced performance, security, and serviceability.

Lite IBM

[View Docs](#)

VERSION 3.x
TYPE Application
REGION Sydney, Germany, US East, US South, United Kingdom

App name: a0134671-app

Host name: a0134671

Domain: mybluemix.net

Choose a region/location to deploy in: US South

Choose an organization: a0134671@u.nus.edu

Choose a space: dev

Pricing Plans

Monthly prices shown are for country or region: [Singapore](#)

PLAN	FEATURES	PRICING
✓ Default	Run one or more apps free for 30 days (375 GB-hours free).	\$0.0735 USD/GB-Hour

This is a service plan for the IBM Bluemix Platform runtime.

Need Help?
[Contact IBM Cloud Sales](#)

Estimate Monthly Cost
[Cost Calculator](#)

\$0.0735 USD/GB-Hour

Create

Step 7 : Install Bluemix CLI

The screenshot shows the IBM Cloud Application Details page for an app named 'a0134671-app'. The app is listed as 'Running'. The page provides instructions for managing the app via the command line interface. A prominent blue button labeled 'Download Bluemix Command Line Interface' is circled in red at the top of the instructions section. Below the button, there is a restriction note about Cygwin and a list of steps for getting started.

Cloud Foundry apps /

a0134671-app • Running [Visit App URL](#)

Org: a0134671@u.nus.edu Location: US South Space: dev

Download, modify, and redeploy your Cloud Foundry app with the command line interface

Last Updated: 2017-06-31 | [Edit in GitHub](#)

Use IBM Cloud command line interface to download, modify, and redeploy your Cloud Foundry applications and service instances.

Before you begin, download and install the IBM Cloud command line interface.

[Download Bluemix Command Line Interface](#)

Restriction: The command line tool is not supported by Cygwin. Use the tool in a command line window other than the Cygwin command line window.

After you install the command line interface, you can get started:

- ① Change to the directory where your code is located.
\$ cd *your_new_directory*
- ② Make changes to your app code as you see fit. For example, if you are using a IBM® Cloud sample application and your app contains the `src/main/webapp/index.html` file, you can modify it and edit "Thanks for creating ..." to say something new. Ensure the app runs locally before you deploy it back to IBM Cloud.

Take note of the `manifest.yml` file. When deploying your app back to IBM Cloud, this file is used to determine your application's URL, memory allocation, number of instances, and other crucial parameters. You can [read more about the manifest file](#) in the Cloud Foundry documentation.

Also pay attention to the `README.md` file, which contains details such as build instructions if applicable.

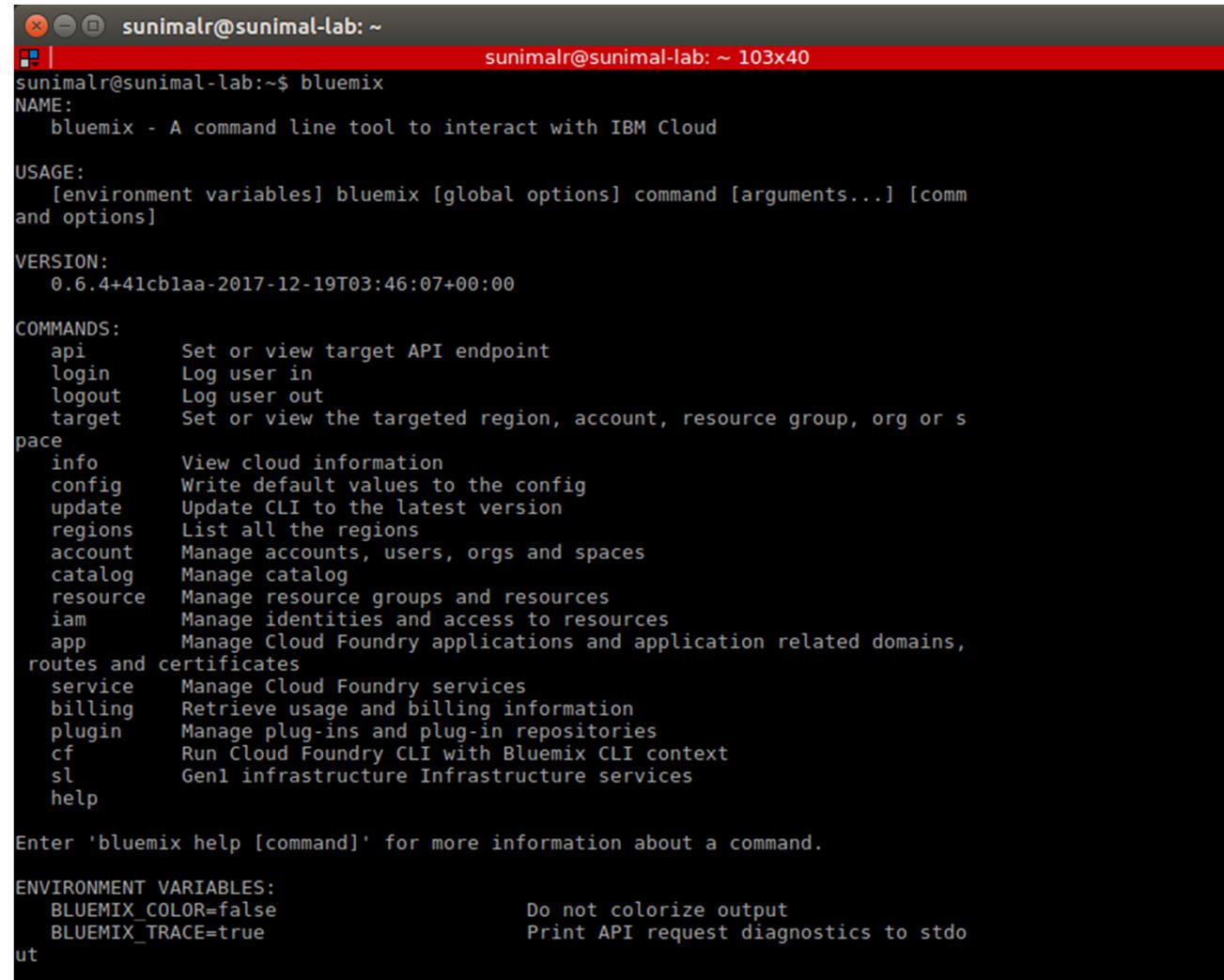
Note: If your application is a Liberty app, you must build it before redeploying.

Step 7 : Install Bluemix CLI

The screenshot shows a web browser window with the URL https://console.bluemix.net/docs/cli/reference/bluemix_cli/get_started.html#getting-started. The page title is "Getting started with IBM Cloud CLI". The left sidebar has sections for "CLI and Dev Tools", "LEARN", "HOW TO", and "REFERENCE". The "HOW TO" section contains links for "Download and install IBM Cloud CLI", "Extending IBM Cloud CLI with plug-ins", "Logging in IBM Cloud", and "Debugging". The "REFERENCE" section lists "IBM Cloud (bx) commands", "IBM Cloud CLI Installer all versions", "IBM Cloud developer tools commands", "IBM Cloud infrastructure (bluemix cli) commands", "Cloud Foundry (cf) commands", "IBM Cloud CLI plug-ins", "Cloud Foundry CLI plug-ins", and "Recommended development environments". A search bar at the top right contains the text "cloud-cli". The main content area has a heading "Getting started with IBM Cloud CLI" and a sub-section "To get started with IBM Cloud CLI:". Step 1 is "Select the installer of your OS to download" with links for Mac OS X 64 bit, Windows 64 bit, and Linux 64 bit. Step 2 is "Run the installer" with instructions for macOS and windows. Step 3 is "Target an API endpoint and login to IBM Cloud" with a terminal screenshot showing the command \$ bx. A red circle highlights the download links for Mac OS X, Windows, and Linux.

Step 7 : Install Bluemix CLI

- Open command line and enter ‘bluemix’ and see if installation is a success. (as below)



```
sunimalr@sunimal-lab:~$ bluemix
NAME:
  bluemix - A command line tool to interact with IBM Cloud

USAGE:
  [environment variables] bluemix [global options] command [arguments...] [comm
and options]

VERSION:
  0.6.4+41cbla-2017-12-19T03:46:07+00:00

COMMANDS:
  api      Set or view target API endpoint
  login    Log user in
  logout   Log user out
  target   Set or view the targeted region, account, resource group, org or s
pace
  info     View cloud information
  config   Write default values to the config
  update   Update CLI to the latest version
  regions  List all the regions
  account  Manage accounts, users, orgs and spaces
  catalog  Manage catalog
  resource Manage resource groups and resources
  iam      Manage identities and access to resources
  app      Manage Cloud Foundry applications and application related domains,
routes and certificates
  service  Manage Cloud Foundry services
  billing  Retrieve usage and billing information
  plugin   Manage plug-ins and plug-in repositories
  cf       Run Cloud Foundry CLI with Bluemix CLI context
  sl       Gen1 infrastructure Infrastructure services
  help     Help on commands

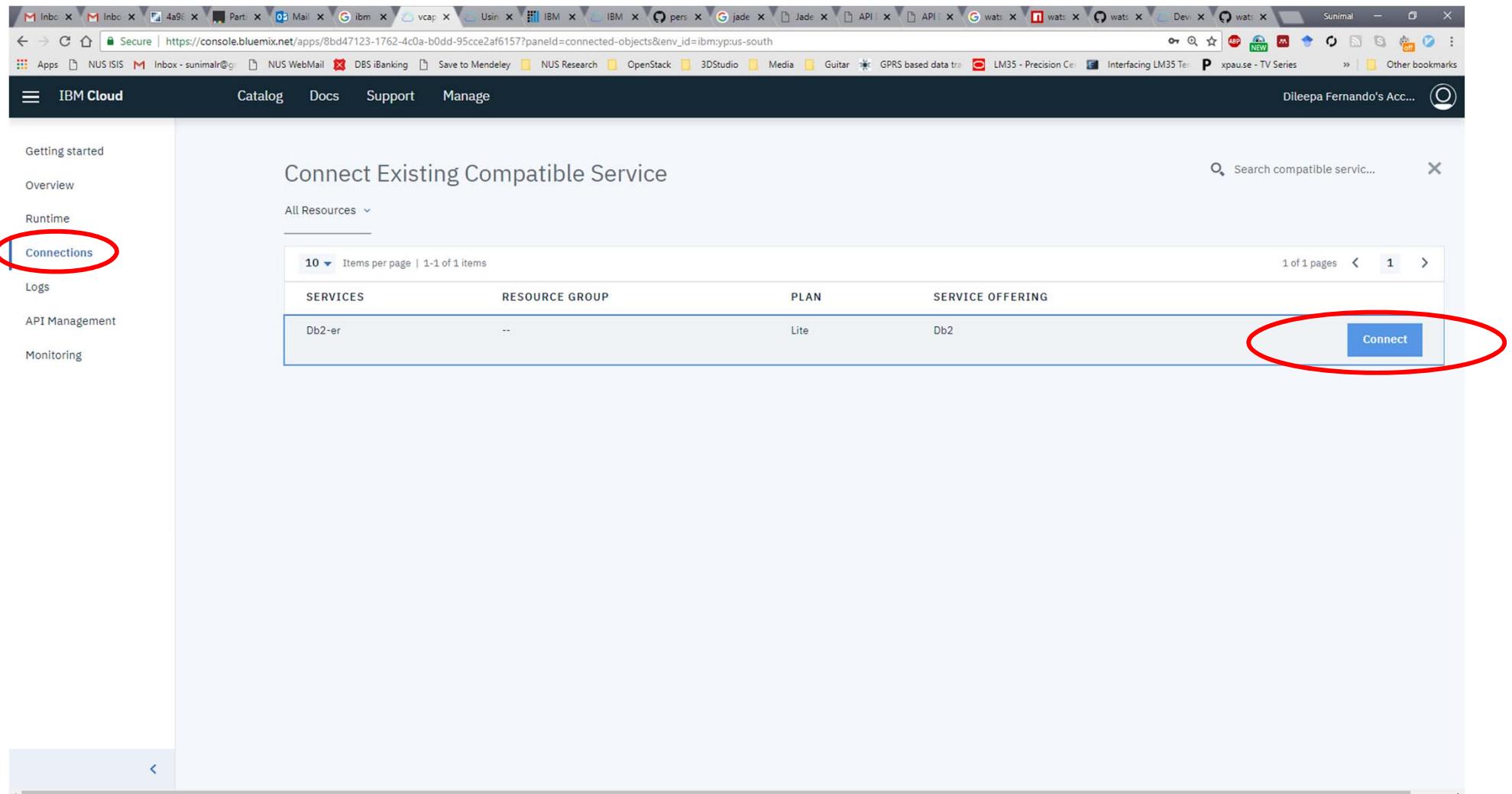
Enter 'bluemix help [command]' for more information about a command.

ENVIRONMENT VARIABLES:
  BLUEMIX_COLOR=false          Do not colorize output
  BLUEMIX_TRACE=true           Print API request diagnostics to stdo
ut
```

Milestone 2: Install CLI

- Did you successfully install Bluemix CLI?
- Bluemix CLI implements Cloud Foundry services.
- Let's wait until everyone is done!
- In the meantime:
 - Read more about cloud foundry at:
<http://searchcloudcomputing.techtarget.com/definition/Cloud-Foundry>

Step 8: Connect the web app to Db2 database



The screenshot shows the IBM Cloud console interface. On the left, a sidebar menu is open with the following items: Getting started, Overview, Runtime, **Connections** (which is circled in red), Logs, API Management, and Monitoring. The main content area is titled "Connect Existing Compatible Service". It displays a table with one item: "Db2-er" under SERVICES, "--" under RESOURCE GROUP, Lite under PLAN, and Db2 under SERVICE OFFERING. A blue "Connect" button is located at the end of this row, which is also circled in red. The top navigation bar shows the URL as https://console.bluemix.net/apps/8bd47123-1762-4c0a-b0dd-95cce2af6157?panelId=connected-objects&env_id=ibm:ypus-south.

SERVICES	RESOURCE GROUP	PLAN	SERVICE OFFERING
Db2-er	--	Lite	Db2

Step 8: Note the environment variables

Under runtime tab, you can see VCAP_SERVICES environment variables. This information is needed for external connections.

The screenshot shows the IBM Cloud console interface. On the left, a sidebar menu is visible with options: Getting started, Overview, **Runtime** (which is circled in red), Connections, Logs, API Management, and Monitoring. The main content area displays a Cloud Foundry app named "ibm-db-hands-on". The app's status is shown as "awake but will sleep after 10 more days of development inactivity". Below the status, it shows the Org: a0134674@u.nus.edu, Location: US South, and Space: dev. There are three tabs at the top of the app details: Memory and instances, Environment variables (which is selected and highlighted in blue), and SSH. A large red oval highlights the "Environment variables" section. Within this section, the VCAP_SERVICES variable is expanded, showing its contents. The VCAP_SERVICES JSON object contains entries for "dashDB For Transactions" and "dashDB For Analytics". The "dashDB For Transactions" entry includes fields like "hostname", "password", "https_url", "port", "ssldsn", "host", "dbcurl", "uri", "db", "dsn", "username", and "ssljdbcurl". The "dashDB For Analytics" entry includes fields like "label", "plan", "site", "name", and "tags". At the bottom right of the "Environment variables" section, there is an "Export" button. The footer of the page indicates "User defined".

```
VCAP_SERVICES
{
  "dashDB For Transactions": [
    {
      "credentials": {
        "hostname": "dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net",
        "password": "6d32@xxjg2mcf60",
        "https_url": "https://dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net",
        "port": 50000,
        "ssldsn": "DATABASE=BLUDB;HOSTNAME=dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net;PORT=50001;PROTOCOL=TCPPIP;UID=idz12073;PWD=6d32@xxjg2mcf60;Security=SSL",
        "host": "dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net",
        "dbcurl": "jdbc:db2://dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net:50000/BLUDB",
        "uri": "db2://idz12073:6d32%40xxjg2mcf60r@dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net:50000/BLUDB",
        "db": "BLUDB",
        "dsn": "DATABASE=BLUDB;HOSTNAME=dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net;PORT=50000;PROTOCOL=TCPPIP;UID=idz12073;PWD=6d32@xxjg2mcf60;",
        "username": "idz12073",
        "ssljdbcurl": "jdbc:db2://dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net:50001/BLUDB:sslConnection=true"
      },
      "syslog_drain_url": null,
      "volume_mounts": [],
      "label": "dashDB For Transactions",
      "plan": "standard",
      "site": "us-east",
      "name": "Db2_Standard",
      "tags": [
        "big_data",
        "ibm_created"
      ]
    }
  ],
  "dashDB For Analytics": [
    {
      "label": "dashDB For Analytics"
    }
  ]
}
```

Step 9: Download Node.js content files

- Download the db-app.zip file from the course web page to your local computer and extract
- File structure
 - app.js – web application core
 - manifest.yml – deployment parameters for Cloud Foundry
 - package.json – app dependencies
 - routes/index.js – functions to connect to Database
- Take note of manifest.yml and package.json which sets deployment parameters and resolve dependencies when deploying applications via Cloud Foundry and app.js file which sets up server variables and initiates the web server

List of data and code with location (Example 1)

- Data files
 - stats.csv (downloaded at above Step3)
- Source files(db-app.zip file downloadable from course webpage)
 - db-app
 - views
 - layout.jade layout design
 - tablelist.jade
 - routes
 - index.js control logic
 - app.js
 - manifest.yml
 - package.json dependencies and app properties

app.js

```
var express = require('express');
var routes = require('./routes');
var http = require('http');
var path = require('path');
var ibmdb = require('ibm_db');
require('cf-deployment-tracker-client').track();

var app = express();

// all environments
app.set('port', process.env.PORT || 3000);
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
app.use(express/favicon());
app.use(express.logger('dev'));
app.use(express.json());
app.use(express.urlencoded());
app.use(express.methodOverride());
app.use(express.cookieParser('your secret here'));
app.use(express.session());
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));
var db2;
var hasConnect = false;

// development only
if ('development' == app.get('env')) {
  app.use(express.errorHandler());
}

if (process.env.VCAP_SERVICES) {
  var env = JSON.parse(process.env.VCAP_SERVICES);
  if (env['dashDB']) {
    hasConnect = true;
    db2 = env['dashDB'][0].credentials;
  }
}

if ( hasConnect == false ) {

  db2 = {
    db: "BLUDB",
    hostname: "xxxx",
    port: 50000,
    username: "xxxx",
    password: "xxxx"
  };
}

var connString = "DRIVER={DB2};DATABASE=" + db2.db + ";UID=" + db2.username + ";PWD=" + db2.password + ";HOSTNAME=" + db2.hostname + ";port=" + db2.port;

app.get('/', routes.listSysTables(ibmdb,connString));

http.createServer(app).listen(app.get('port'), function(){
  console.log('Express server listening on port ' + app.get('port'));
});
```

package.json

```
{  
  "engines": {  
    "node": "^4"  
  },  
  "name": "db2dbnodesample",  
  "version": "0.0.1",  
  "description": "A sample node app for connecting to db2 service on Bluemix",  
  "dependencies": {  
    "cf-deployment-tracker-client": "^0.0.8",  
    "express": "^3.5.3",  
    "ibm_db": "^2.0.0",  
    "jade": "^1.11.0"  
  },  
  "scripts": {  
    "start": "node app.js"  
  },  
}
```

manifest.yml

```
---
declared-services:
  dashDB-nodesample:
    label: "dashDB For Transactions"
    plan: Lite
applications:
# replace the host variable below with your own unique one, as this
one can be already taken
- name: ibm-db-hands-on
  memory: 64M
  instances: 1
  path: .
  host: sunimal
  framework: node
  command: node app.js
  buildpack: sdk-for-nodejs
  services:
    - Db2-er
```

index.js

```
exports.listSysTables = function(ibmdb, connString) {
    return function(req, res) {

        ibmdb.open(connString, function(err, conn) {
            if (err) {
                res.send("error occurred " + err.message);
            }
            else {
                conn.query("SELECT * FROM LDZ12073.STATS FETCH FIRST 10 ROWS ONLY", function(err, tables, moreResultSets) {

                    if (!err) {
                        res.render('tablelist', {
                            "tablelist" : tables,
                            "tableName" : "10 rows from the STATS table",
                            "message": "Congratulations. Your connection to Db2 is successful."
                        });
                    }
                    else {
                        res.send("error occurred " + err.message);
                    }
                }/*
                Close the connection to the database
                param 1: The callback function to execute on completion of close function.
                */
                conn.close(function(){
                    console.log("Connection Closed");
                });
            }
        });
    }
}
```

Note: index.js is located in the ‘routes’ folder

layout.jade

```
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    block content
```

tablelist.jade

```
extends layout

block content
  h1=message
  h2=tableName

  form#list-form
    div#main
      table#list
        thead
          tr
            for hdr in Object.keys(tablelist[0])
              th #{hdr}
        tbody
          if tablelist && tablelist.length > 0
            each row in tablelist
              tr
                each col in row
                  td #{col}
```

Note: located in the ‘views’ folder

Step 10: Modify index.js

Use the **schema.table** name from step 4

```
exports.listSysTables = function(ibmdb, connString) {
    return function(req, res) {

        ibmdb.open(connString, function(err, conn) {
            if (err) {
                res.send("error occurred " + err.message);
            }
            else {
                conn.query("SELECT * FROM LDZ12073.STATS FETCH FIRST 10 ROWS ONLY", function(err, tables, moreResultSets) {

                    if (!err) {
                        res.render('tablelist', {
                            "tablelist" : tables,
                            "tableName" : "10 rows from the STATS table",
                            "message": "Congratulations. Your connection to Db2 is successful."
                        });
                    }
                    else {
                        res.send("error occurred " + err.message);
                    }
                    /*
                    Close the connection to the database
                    param 1: The callback function to execute on completion of close function.
                    */
                    conn.close(function(){
                        console.log("Connection Closed");
                    });
                });
            }
        });
    }
}
```

Note: index.js is located in the ‘routes’ folder

Step 10: Modify manifest File

```
---  
declared-services:  
  dashDB-nodesample:  
    label: "dashDB For Transactions"  
    plan: Lite  
applications:  
# replace the host variable below with your own unique one, as this  
one can be already taken  
- name: your app name here this is the maximum allowed RAM for the app. When you  
  memory: 64M ← create the app in web console (in step 2 and 3), you may be  
  instances: 1          asked to select memory size. Use that number here.  
  path: .  
  host: your host name here  
  framework: node  
  command: node app.js  
  buildpack: sdk-for-nodejs  
services:  
- name for the database (eg: db-<your nusnet id>) - we did this in  
  step 2
```

Steps 11, 12: Cloud Foundry and Bluemix

- Open command line and change directory into the folder with code downloaded in the previous step
- Use the Bluemix CLI to connect to the Bluemix API (Depending on the region you are logged on to. ‘ng’ is for US South)
 - bluemix api <https://api.ng.bluemix.net>
- Login to Bluemix
 - >`bluemix login -u <bluemix_id> -o <organization> -s <space>`
 - use the IBM ID and password
- upload your application to Bluemix
 - `bluemix app push <appname>`

Step 13: View App

- Go to your favorite browser and enter the URL:

http://your_host_name.mybluemix.net

The screenshot shows a web browser displaying the results of a SQL query against a database named 'STATS'. The query is as follows:

```
SELECT * FROM STATS;
```

The results are presented in a table with the following columns:

Country_Code	Short_Name	Full_Name	Long_Name	alpha_3	Currency_1	Special_Notes	Region	Income_Group	International_memberships	WR_2_code	National_accounts_finst_year	National_accounts_refinance_year	SNA_price_valuation	Lending_category	Other_group		
AFG	Afghanistan	Afghanistan	Afghanistan	AF	Afghani	Afghanistan are covered from the AfP and AfR datasets from the Central Statistics Organization members for inclusion of the regional economy.	South Asia	Low income		AF	7002103				Value added at basic prices (VAB)	IDA	HIPC
ALB	Albania	Albania	Republic of Albania	AL	Albanian lek		Europe & Central Asia	Upper middle income		AL	Original constant price 1995				Value added at basic prices (VAB)	IDA	HIPC
DZA	Algeria	Algeria	People's Democratic Republic of Algeria	DZ	Algerian dinar		Europe & Central Asia	Upper middle income		DZ	1989				Value added at basic prices (VAB)	IDA	HIPC
ASM	American Samoa	American Samoa	American Samoa	AS	U.S. dollar		East Asia & Pacific	Upper middle & Pacific income		AS					Value added at market prices (VMP)	IDA	HIPC
ADO	Andorra	Andorra	Principality of Andorra	AD	Euro	April 2014 database updated based on IMF data, national accounts data were revised	Europe & Central Asia	High income: nonOECD		AD	1999				Value added at market prices (VMP)	IDA	HIPC
AOO	Angola	Angola	People's Republic of AO	AO	Angolan kwanza		Sub-Saharan Africa	Upper middle income		AO	7002				Value added at market prices (VMP)	IDA	HIPC

<http://sunimal.mybluemix.net>

- You should see the contents of the database on the web browser

Milestone 3: Load Web App

- Did you successfully load the web app?
- Let's wait until everyone is done!
- In the meantime:
 - Read more about core node.js files at:
<https://docs.cloudfoundry.org/buildpacks/node/node-tips.html>

WEB APPLICATION USING WATSON ANALYTICS

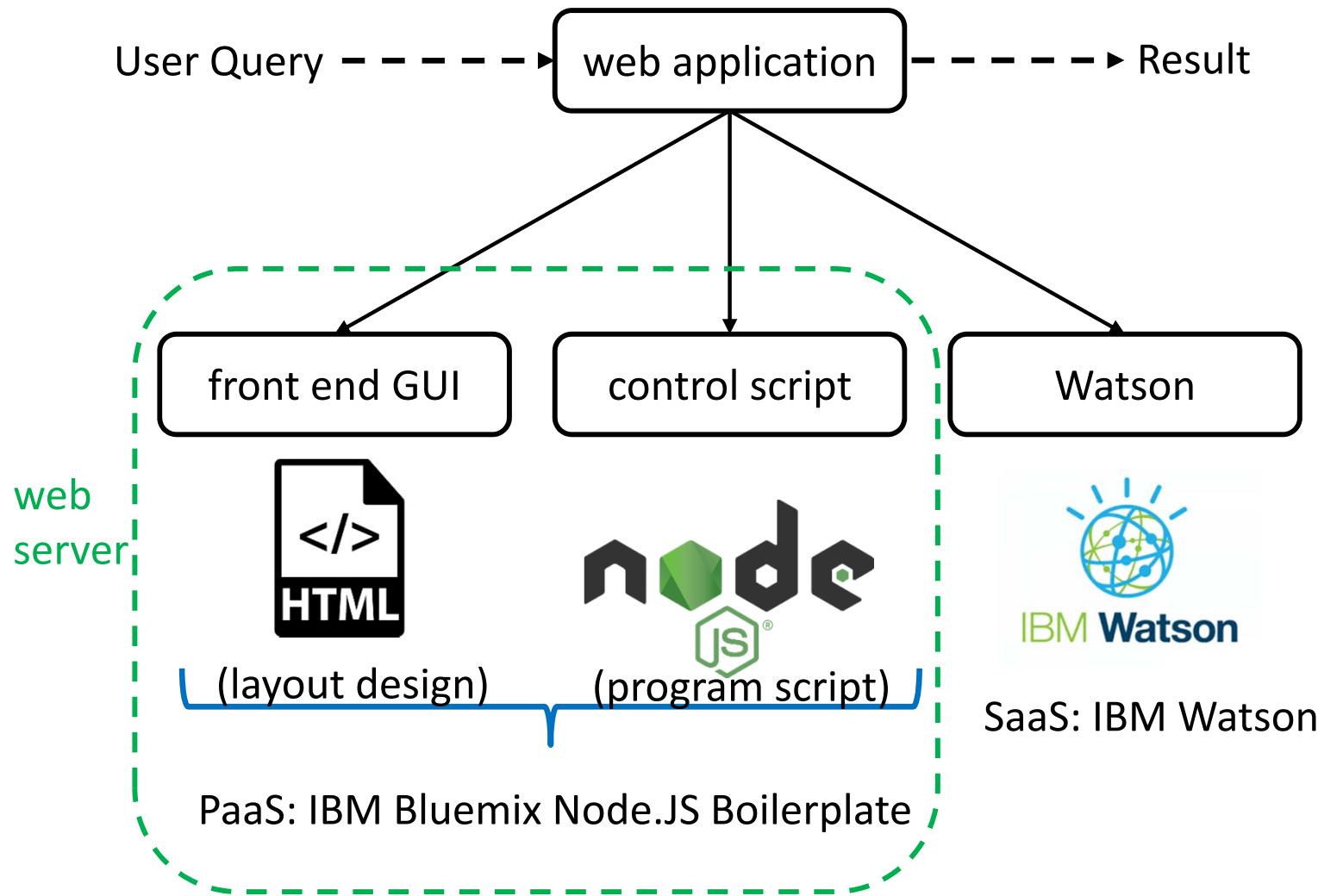
- Watson is a cognitive analysis service powered by different AI techniques
- Watson could be used for different cognitive tasks such as
 - summarize a text to filter important points
 - analyze a person's personality based on their resume
 - build interactive chat-bots to communicate with clients and answer their questions
 -
- Watson comes with a comprehensive API that helps integrate it with other IBM tools and services such as database, machine learning, web applications, among others

Watson Analytics

- Objective: use SaaS service (Watson) to get personality insights from a piece of text
- Example: a web server application to upload a part of a text and call IBM Watson services to get personality insights based on the uploaded text
- Data: part of a text from US former president Barak Obama's speech

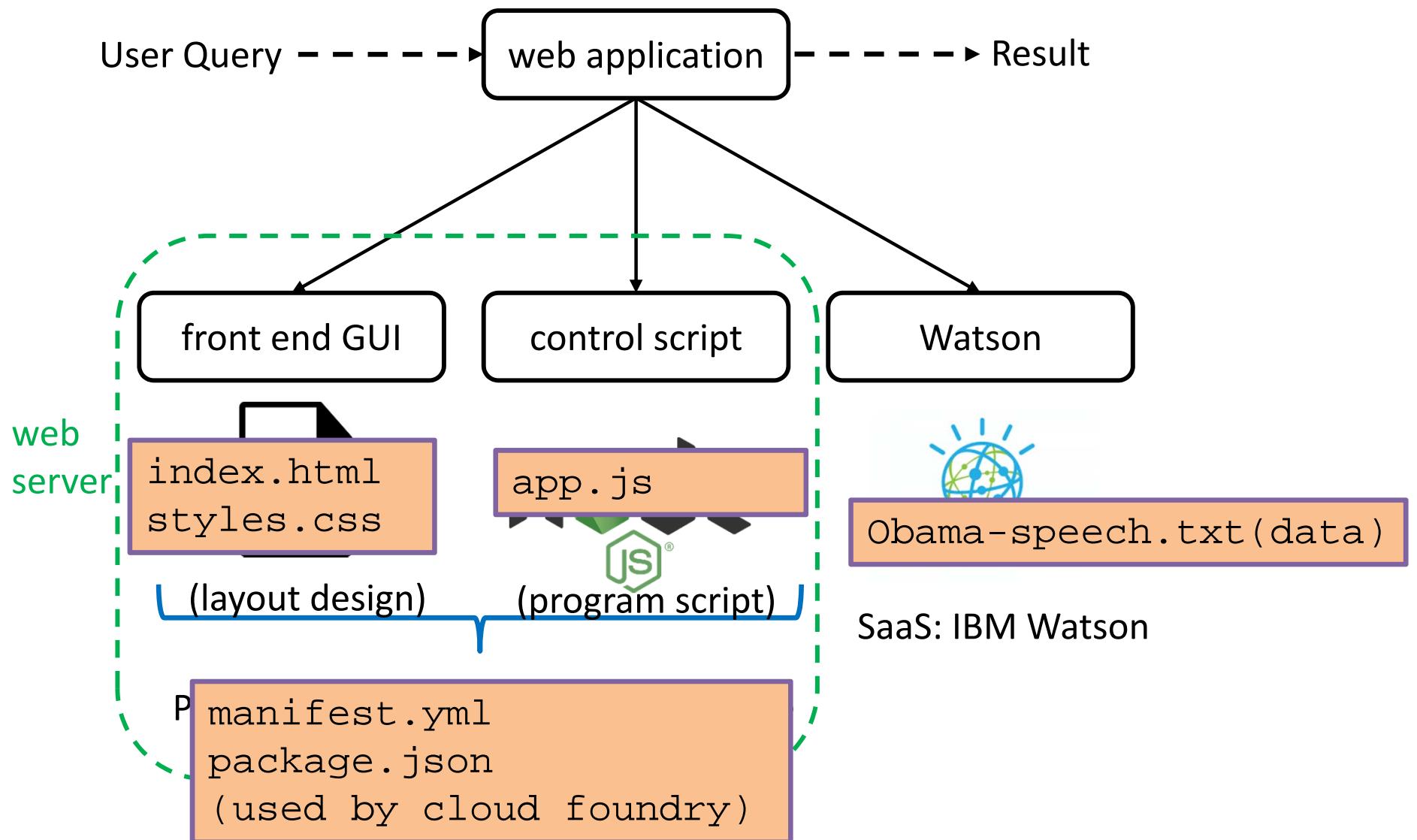
Developing a Web Application

What we are going to use for Example 2



Developing a Web Application

What we are going to use for Example 2



Watson Analytics: Main Steps in Setting up Web Server Application

1. Sign up for IBM Bluemix and go to the Catalog
2. Select Cloud Foundry Node.js app from the catalog – this is needed for every new application
3. Provide a hostname that serves as the web application URL – hostname.mybluemix.net
4. Install the Bluemix CLI (command line interface) client on your local machine - this is required only if you have not done so previously. We did this in Example 1.
5. To start a new web application, download the code (index.js, manifest.yml, app.js, package.json,...) onto your local directory from course web site
6. Modify manifest.yml (and other source code files as necessary)
7. Using Bluemix CLI, login to Bluemix
8. Create Watson personality insights service using CLI
9. Using the Bluemix CLI, upload your application to Bluemix!
10. To access the web application, enter your host URL (in step 3)

In Example 2 we demonstrate how to create a service with the CLI without using the web browser interface (step 8)

Step 2: Select Cloud Foundry Node.js

- Go back to catalog and select SDK for Node.js under CloudFoundry apps

The screenshot shows the IBM Cloud Catalog interface. On the left, there's a sidebar with 'All Categories' and sections for Infrastructure (Compute, Storage, Network, Security, Containers, VMware), Platform (Boilerplates, APIs, Application Services, Blockchain), and Cloud Foundry Apps. The 'Cloud Foundry Apps' section is currently selected, indicated by a blue border around its link. The main content area displays various runtime environments: Liberty for Java™, SDK for Node.js™ (which is circled in red), Runtime for Swift, XPages, Go, PHP, Python, Ruby, and Tomcat. Each entry includes a brief description and an 'IBM' badge.

Step 3: Provide required app details

Use a unique app name and hostname.
(this should be different from the hostname you gave in Example 1)

Sign up for IBM Cloud SDK for Node.js™ - IBM + https://console.bluemix.net/catalog/startersdks-for-nodejs?taxonomynavigation=apps Catalog Docs Support Manage

IBM Cloud View all Create a Cloud Foundry App

SDK for Node.js™

Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides enhanced performance, security, and serviceability.

Lite IBM

[View Docs](#)

VERSION 3.x
TYPE Application
REGION Sydney, Germany, US East, US South, United Kingdom

App name: a0134671-app

Host name: a0134671

Domain: mybluemix.net

Choose a region/location to deploy in: US South Choose an organization: a0134671@u.nus.edu Choose a space: dev

Pricing Plans Monthly prices shown are for country or region: Singapore

PLAN	FEATURES	PRICING
✓ Default	Run one or more apps free for 30 days (375 GB-hours free).	\$0.0735 USD/GB-Hour

This is a service plan for the IBM Bluemix Platform runtime.

Need Help? [Contact IBM Cloud Sales](#) Estimate Monthly Cost [Cost Calculator](#) \$0.0735 USD/GB-Hour [Create](#)

Step 5: Download Node.js content files

- Download the watson-app.zip file from the course web page to your local computer and extract it
- File structure
 - app.js – web application core
 - manifest.yml – deployment parameters for Cloud Foundry
 - package.json – app dependencies
 - public/index.html – web layout
- Take note of manifest.yml and package.json which sets deployment parameters and resolve dependencies when deploying applications via Cloud Foundry and app.js file which sets up server variables and initiates the web server

List of data and code with location (Example 2)

- Data files (watson-data.zip downloadable from course webpage)
 - watson-data
 - obama-speech.txt
- Source files (watson-app.zip downloadable from course webpage)
 - watson-app
 - public
 - stylesheets
 - style.css layout design
 - index.html
 - routes
 - app.js control logic
 - manifest.yml dependencies and app properties
 - package.json

app.js

```
/*eslint-env node*/
//-----
// node.js starter application for Bluemix
//-----

// This application uses express as its web server
// for more info, see: http://expressjs.com
var express = require('express');

// cfenv provides access to your Cloud Foundry environment
// for more info, see: https://www.npmjs.com/package/cfenv
var cfenv = require('cfenv');

// create a new express server
var app = express();

// serve the files out of ./public as our main files
app.use(express.static(__dirname + '/public'));

// get the app environment from Cloud Foundry
var appEnv = cfenv.getAppEnv();

// start server on the specified port and binding host
app.listen(appEnv.port, '0.0.0.0', function() {
    // print a message when the server starts listening
    console.log("server starting on " + appEnv.url);
});

var watson = require('watson-developer-cloud'),
    multer = require('multer');

var creds = appEnv.getServiceCreds('personality-insights-tutorial');
creds.version = 'v2';
var personalityInsights = watson.personality_insights(creds);

var uploading = multer({
    storage: multer.memoryStorage()
});

app.set('json spaces', 4);

app.post('/upload', uploading.single('file'), function (request, response) {
    console.log("file");
    var txtFile = request.file.buffer.toString();
    console.log(request.file.buffer);

    personalityInsights.profile({
        text: txtFile },
        function (error, result) {
            if (error) {
                response.send(error);
            } else {
                response.json(result);
            }
        });
    });

//----Deployment Tracker-----
require('cf-deployment-tracker-client').track();
```

Open the file in a
text editor to clearly
see/modify the
content

package.json

```
{  
  "name": "NodejsStarterApp",  
  "version": "0.0.1",  
  "description": "A sample nodejs app for Bluemix",  
  "scripts": {  
    "start": "node app.js"  
  },  
  "dependencies": {  
    "body-parser": "^1.14.2",  
    "cfenv": "1.0.x",  
    "express": "4.12.x",  
    "multer": "^1.1.0",  
    "watson-developer-cloud": "1.x",  
    "cf-deployment-tracker-client": "*"  
  },  
  "engines": {  
    "node": "0.12.x"  
  },  
  "repository": {  
    "type": "git",  
    "url": "https://github.com/IBM-Bluemix/personality-insights-nodejs-tutorial.git"  
  }  
}
```

public/index.html

```
<!DOCTYPE html>
<html>

<head>
  <title>NodeJS Watson Application</title>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge" >
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet" href="stylesheets/style.css">
</head>

<body>
  <form action="/upload" method="POST"
enctype="multipart/form-data">
    Select a txt file to upload:
    <input type="file" name="file">
    <input type="submit" value="Upload File">
  </form>

</body>

</html>
```

Step 6: Modify manifest.yml

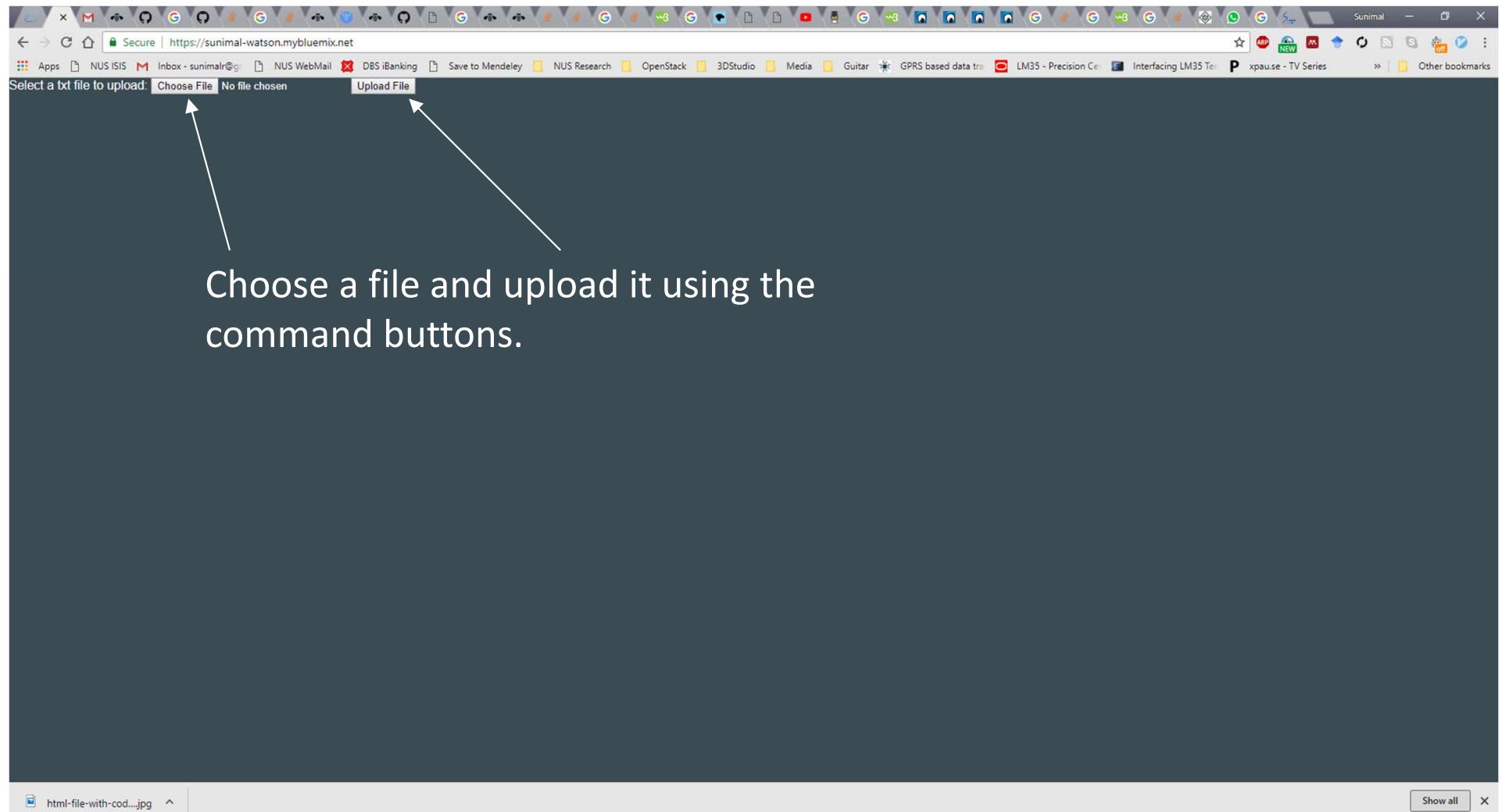
```
---  
declared-services:  
  personality-insights-tutorial:  
    label: personality_insights  
    plan: lite  
applications:      this is the maximum allowed RAM for the app. When  
- path: .  
  memory: 64M          you create the app in web console (in step 2 and 3),  
  instances: 1          you may be asked to select memory size. Use that  
  domain: mybluemix.net number here.  
  name: sunimal-watson  
  host: sunimal-watson  
  disk_quota: 1024M  
services:  
- personality-insights-tutorial
```

use the hostname and app name you selected

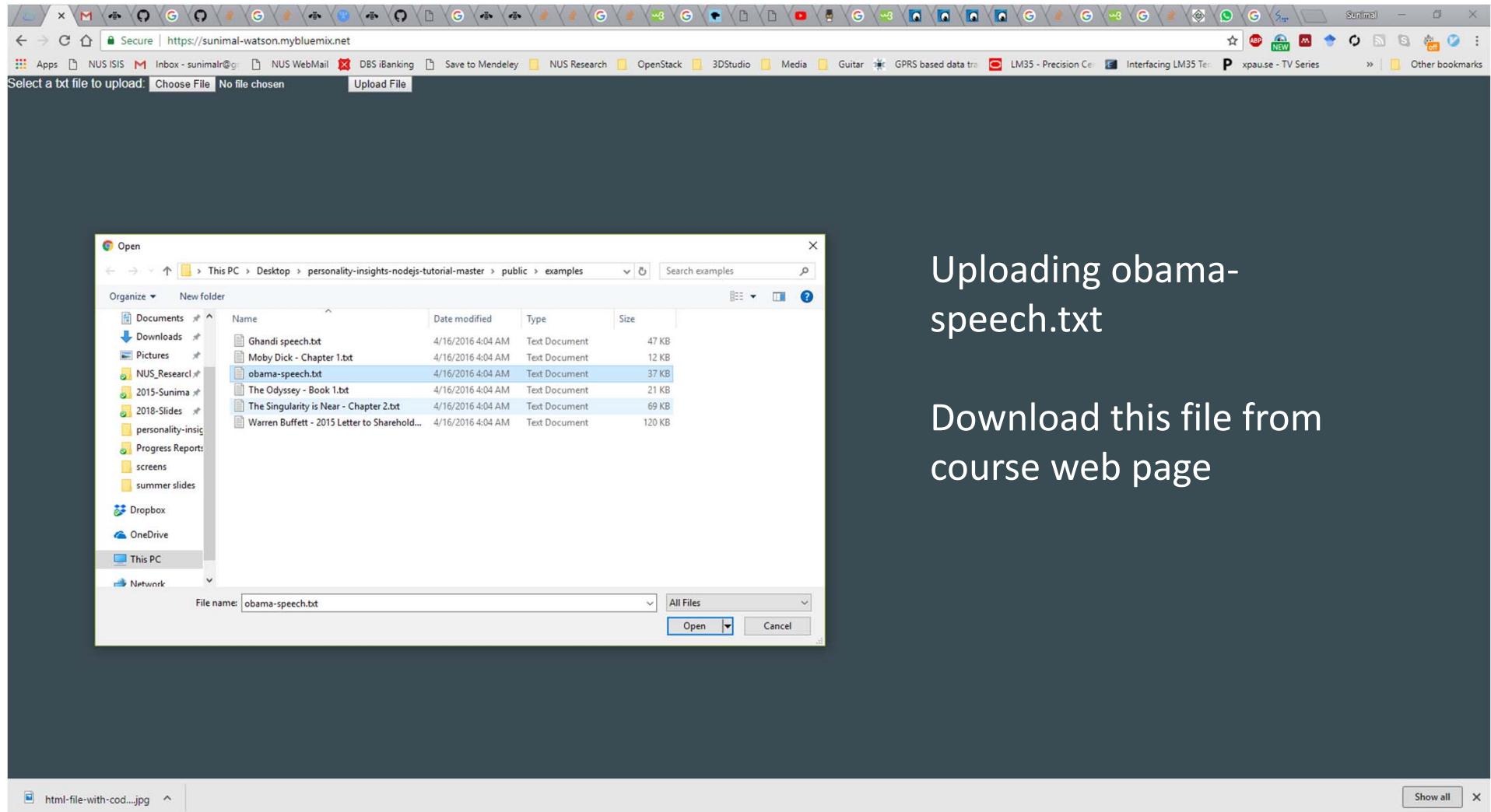
Step 7-9: Bluemix CLI Steps

- Open command line and change directory into the folder with code downloaded in the previous step
- Use the Bluemix CLI to connect to the Bluemix API (Depending on the region you are logged on to. 'ng' is for US South)
 - [bluemix api https://api.ng.bluemix.net](https://api.ng.bluemix.net)
- Login to Bluemix
 - [bluemix login -u <bluemix_id> -o <organization> -s <space>](#)
 - use the IBM ID and password
- Create the Personality Insights service in Bluemix
 - [bluemix create-service personality_insights tiered personality-insights-tutorial](#)
- upload your application to Bluemix
 - [bluemix app push <appname>](#)

Step 10: View Result



Step 10: View Result

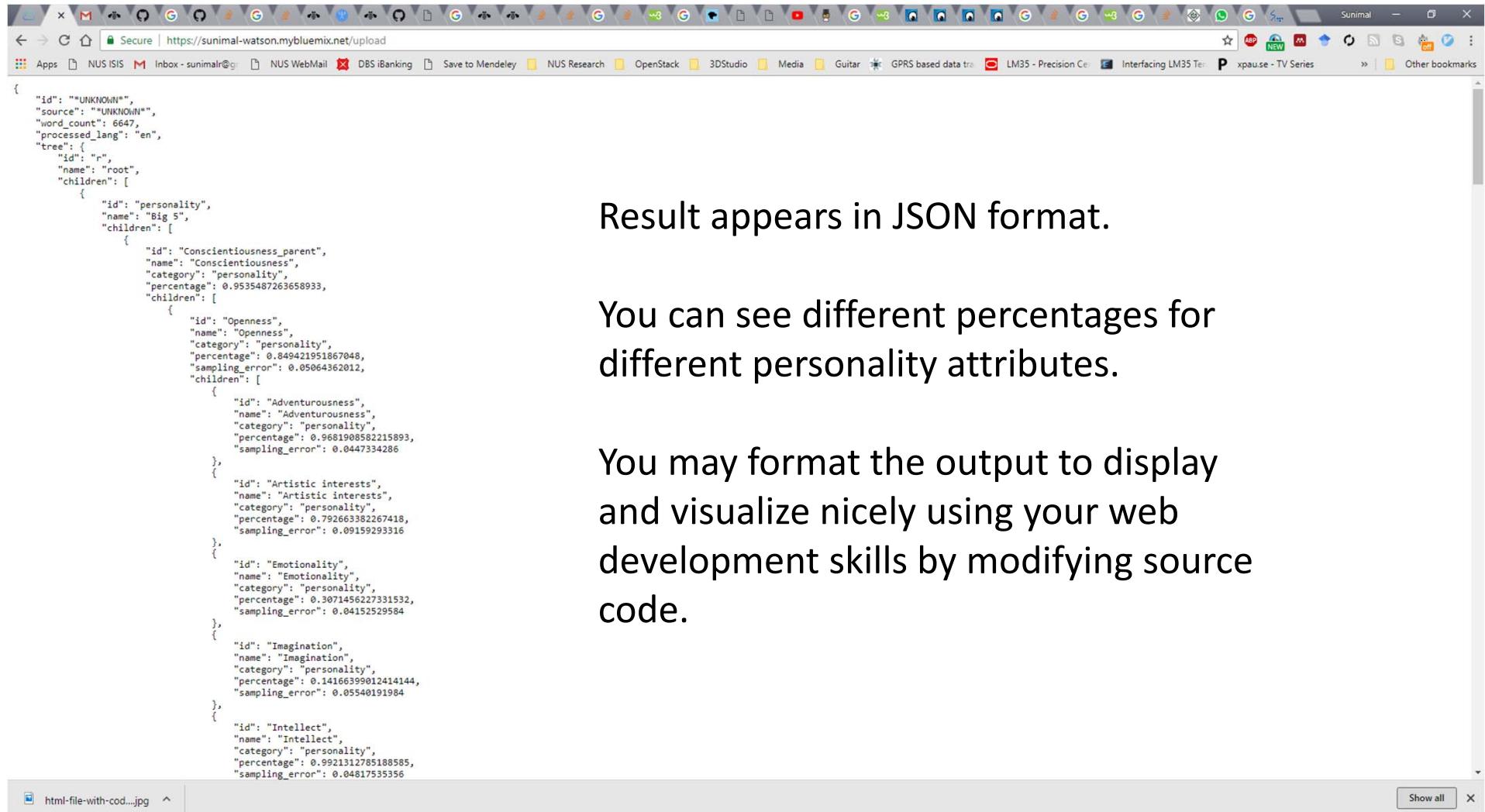


A screenshot of a web browser window titled "Secure | https://sunimal-watson.mybluemix.net". The address bar shows the URL. Below the address bar, there is a toolbar with various icons and a bookmarks bar. A message "Select a txt file to upload:" is displayed above a file input field. The file input field has two buttons: "Choose File" and "Upload File". A file selection dialog box is overlaid on the browser window. The dialog box is titled "Open" and shows a list of files in a folder structure. The folder path is "This PC > Desktop > personality-insights-nodejs-tutorial-master > public > examples". The list includes several text files: "Ghandi speech.txt", "Moby Dick - Chapter 1.txt", "obama-speech.txt" (which is selected), "The Odyssey - Book 1.txt", "The Singularity is Near - Chapter 2.txt", and "Warren Buffett - 2015 Letter to Shareholder...". The "obama-speech.txt" file is highlighted with a blue selection bar. At the bottom of the dialog box, there is a "File name:" dropdown set to "obama-speech.txt", an "Open" button, and a "Cancel" button. The background of the browser window shows the uploaded file list: "html-file-with-cod...jpg". To the right of the dialog box, there is text: "Uploading obama-speech.txt" and "Download this file from course web page".

Uploading obama-speech.txt

Download this file from course web page

Step 10: View Result



The screenshot shows a Google Chrome browser window with the URL <https://sunimal-watson.mybluemix.net/upload>. The page displays a large block of JSON data representing the personality analysis results. The JSON structure starts with an object containing fields like 'id', 'source', 'word_count', 'processed_lang', and 'tree'. The 'tree' field is a complex nested object representing the hierarchical structure of personality traits, with levels for categories like 'personality' and specific traits like 'Openness', 'Adventurousness', etc., each with their respective percentages and sampling errors.

```
{
  "id": "UNKNOWN",
  "source": "UNKNOWN",
  "word_count": 6647,
  "processed_lang": "en",
  "tree": {
    "id": "r",
    "name": "root",
    "children": [
      {
        "id": "personality",
        "name": "Big 5",
        "children": [
          {
            "id": "Conscientiousness_parent",
            "name": "Conscientiousness",
            "category": "personality",
            "percentage": 0.9535487263658933,
            "children": [
              {
                "id": "Openness",
                "name": "Openness",
                "category": "personality",
                "percentage": 0.849421951867048,
                "sampling_error": 0.05064362012,
                "children": [
                  {
                    "id": "Adventurousness",
                    "name": "Adventurousness",
                    "category": "personality",
                    "percentage": 0.9681908582215893,
                    "sampling_error": 0.0447334286
                  },
                  {
                    "id": "Artistic interests",
                    "name": "Artistic interests",
                    "category": "personality",
                    "percentage": 0.792663382267418,
                    "sampling_error": 0.09159293316
                  },
                  {
                    "id": "Emotionality",
                    "name": "Emotionality",
                    "category": "personality",
                    "percentage": 0.3071456227331532,
                    "sampling_error": 0.04152529584
                  },
                  {
                    "id": "Imagination",
                    "name": "Imagination",
                    "category": "personality",
                    "percentage": 0.141663399012414144,
                    "sampling_error": 0.05540191984
                  },
                  {
                    "id": "Intellect",
                    "name": "Intellect",
                    "category": "personality",
                    "percentage": 0.9921312785188585,
                    "sampling_error": 0.04817535356
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Result appears in JSON format.

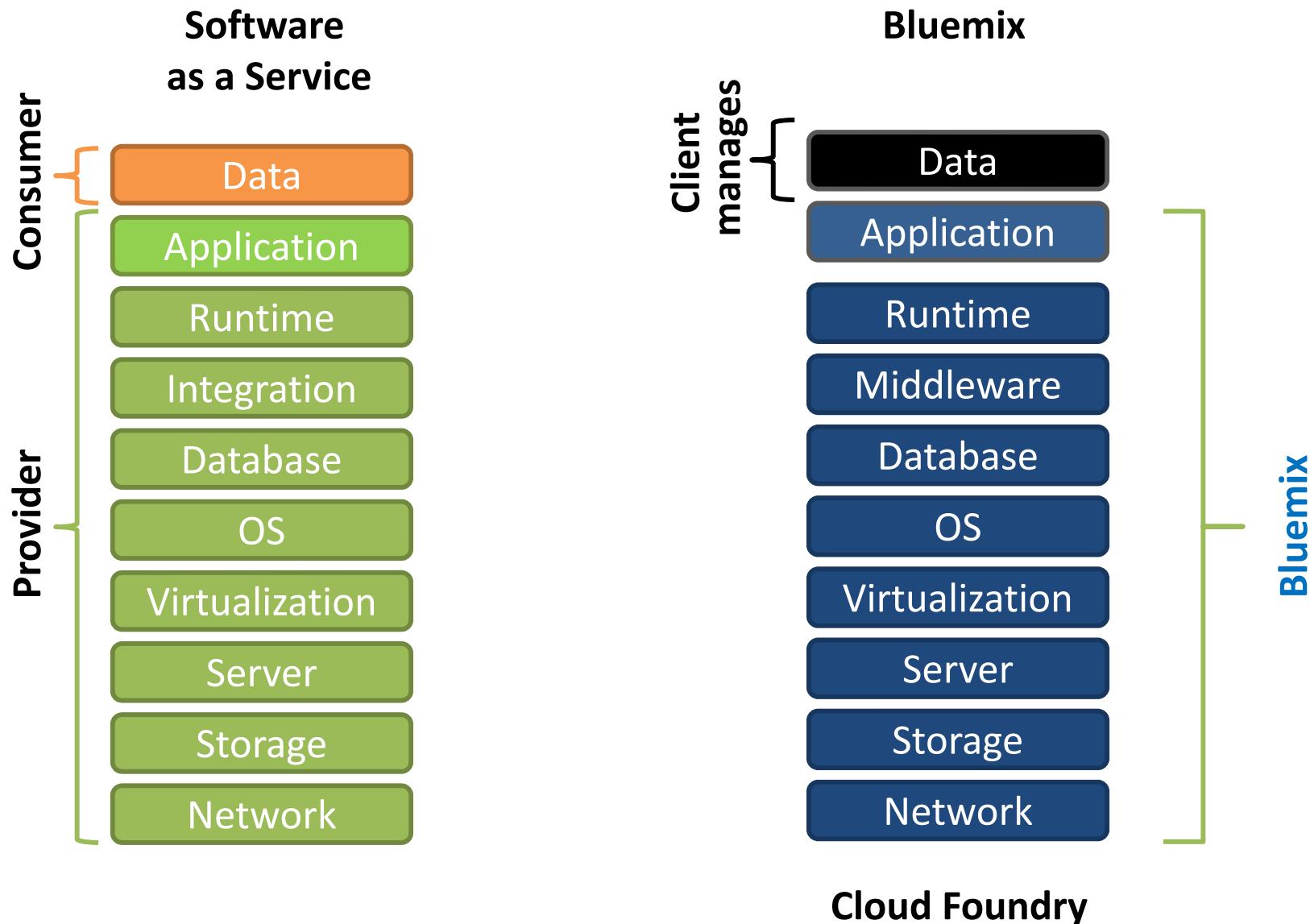
You can see different percentages for different personality attributes.

You may format the output to display and visualize nicely using your web development skills by modifying source code.

Milestone 4: Load Watson Web App

- Did you successfully load the web app?
- Let's wait until everyone is done!
- In the meantime:
 - Learn more about available services under IBM Watson on the Bluemix console

What is SaaS?



EXAMPLE 3 QUERYING WITH SQL SCRIPTING

- SQL is a querying language used for accessing and querying databases
- IBM Cloud offers in cloud SQL language support with Db2 (e.g. IBM Db2 Warehousing)

Objective: Storing and querying data with a database query language

Example: Load economic data of countries to a an IBM Db2 database, and, run simple SQL queries on-cloud on the stored dataset

Data: Economic data of countries from United Nations (stats.csv)

Example 3-Part 1: Querying with SQL Scripting

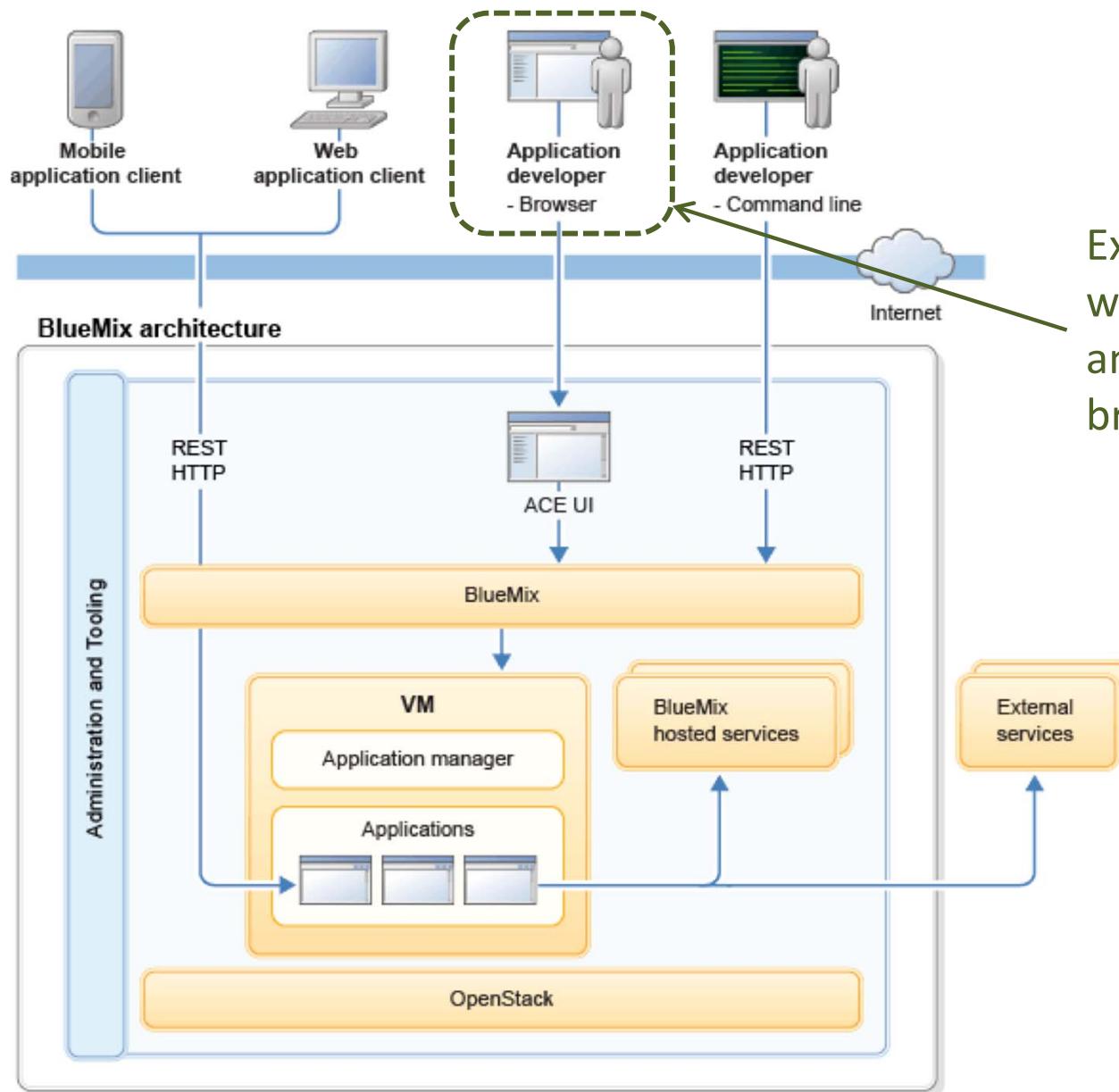
- SQL is a querying language used for accessing and querying databases
- IBM Cloud offers in cloud SQL language support with Db2 (e.g. IBM Db2 Warehousing)

Objective: Storing and querying data with a database query language

Example: Load economic data of countries to a an IBM Db2 database, and, run simple SQL queries on-cloud on the stored dataset

Data: Economic data of countries from United Nations (stats.csv)

SaaS: Analytics using DB2 Service



Extend SaaS to write SQL query and R code in web-browser

Main Steps: Analytics using DB2 Warehouse

Example 2-Part1:
SQL Scripting

1. Login to Bluemix and go to Catalog
2. Select **DB2 Warehouse** under Data & Analytics and setup
3. Open DB2 console
4. Upload data (CSV or Excel) from local computer
5. Review upload data on the cloud
6. Run SQL
7. Start an 'IBM Data Science Experience - DSX' account
8. Create and setup new DSX project
9. Connect DSX to Db2 database
10. Create a 'R' notebook
11. Insert R data frame
12. Write R script to analyze data
13. Output the plot

Example 2-Part2:
R Scripting

Step 1: Select Db2 Warehouse

The screenshot shows the IBM Cloud Catalog interface. The top navigation bar includes links for Catalog, Docs, Support, Manage, and a feedback button. A banner at the top states: "Try the best of the Catalog for free with no time restrictions with Lite plans. The Lite filter is enabled. Remove the filter to see the full Catalog." On the left, a sidebar lists categories: All Categories, Infrastructure, Compute, Storage, Network, Security, Containers, VMware, Platform, Data & Analytics (selected), APIs, Application Services, Blockchain, Cloud Foundry Apps, DevOps, Finance, Functions, Integrate, Internet of Things, Mobile, Security, Storage, and Watson. The main content area displays various database services with their descriptions and status (e.g., IBM, Beta). The 'Db2 Warehouse' service is circled in red.

Service	Description	Status
Compose for MySQL	MySQL is a fast, easy-to-use, and flexible RDBMS.	IBM
Compose for PostgreSQL	Postgres is a powerful, open source object-relational database that is highly customizable.	IBM
Compose for RabbitMQ	RabbitMQ asynchronously handles the messages between your applications and databases.	IBM
Compose for Redis	Redis is an open-source, blazingly fast, low maintenance key/value store.	IBM
Compose for RethinkDB	RethinkDB is a JSON document based, distributed database with an integrated administration and exploration console.	IBM
Compose for ScyllaDB	ScyllaDB is a highly performant, in-place replacement for the Cassandra wide-column distributed database.	IBM
Db2	A next generation SQL database. Formerly dashDB For Transactions.	Lite IBM
Db2 Hosted	Db2 Hosted: Offers customers the rich features of an on-premise Db2 deployment without the cost, complexity, and management overhead.	IBM
Db2 Warehouse	Db2 Warehouse on Cloud is a flexible and powerful data warehouse for enterprise-level analytics.	IBM
Decision Optimization	Develop optimization applications, such as planning or scheduling, using our APIs to connect to the CPLEX.	IBM
Geospatial Analytics	Expand the boundaries of your application. Leverage real-time geospatial analytics to track when devices enter.	IBM
IBM Cognos Dashboard Embedded	Bring data to life directly from your application with this powerful and easy-to-use visualization service.	Lite IBM
Information Server	IBM® Information Server on Cloud allows you to rapidly expand data integration and governance capabilities.	IBM
Informix	IBM Informix on Cloud helps businesses gain a trusted view of data in a hybrid computing environment.	IBM
Lift CLI	Migrate your data quickly, easily and securely from your on-premises data source to an IBM Cloud data property.	IBM
Master Data Management	IBM® Master Data Management (MDM) on Cloud helps businesses gain a trusted view of data in a hybrid com.	IBM
SQL Query	Analyze data in Object Storage with ANSI SQL.	IBM
Streaming Analytics	Leverage IBM Streams to ingest, analyze, monitor, and correlate data as it arrives from real-time data sources.	Lite IBM

Step 2: Db2 Warehouse Setup

The screenshot shows the IBM Cloud catalog interface for creating a new Db2 Warehouse service. The top navigation bar includes tabs for 'Sign up for IBM Cloud', 'Mail - sunimalr@u.nus.edu', and 'Db2 Warehouse - IBM Cloud'. The main content area is titled 'Db2 Warehouse'.

Service name: Db2 Warehouse-y9

Choose a region/location to deploy in: US South

Choose an organization: a0134671@u.nus.edu

Choose a space: dev

Features

- Fully managed, safe, and secure**
Your Db2 Warehouse on Cloud instance is managed, monitored, encrypted and backed-up by IBM, so you can focus on gaining insights from your data instead of administering it.
- Highly Available (HA) with MPP**
Your data workloads, queries and dashboards will all continue to run, even during unexpected failure.
- Compatible with Netezza and Oracle**
Most of your Netezza and Oracle workloads will migrate seamlessly to Db2 Warehouse on Cloud. For the select few edge cases, we offer free tooling to help you make a smooth transition.
- Related Db2 cloud products**
Check out Db2 on Cloud, the enterprise-class, high-performance transactional data store.
- Built for the hybrid cloud**
Our unified architecture enables a range of hybrid use cases. Easily move data between your on-premises data stores and Db2 Warehouse on Cloud. Access data & run SQL queries across multiple heterogeneous data sources with IBM Fluid Query.

Images

Click an image to enlarge and view screen captures, slides, or videos. Screen caps show the user interface for the service after it has been provisioned.

Need Help? [Contact IBM Cloud Sales](#)

Estimate Monthly Cost [Cost Calculator](#)

Create

Step 3: Open Db2 Warehouse Console

The screenshot shows the IBM Cloud interface for managing a service. The left sidebar shows 'Manage' selected. The main area displays a service named 'a0134671-db' located in 'US South' with 'Org: a0134671@u.nus.edu' and 'Space: dev'. A red oval highlights the 'OPEN' button in a teal box under the heading 'Db2 Warehouse on Cloud'. Below it, there's a section about connecting applications and a 'Where to Start' section with 'Learn' and 'Open' options.

When you open the console, you can connect to the service, upload your data, and run analytics from the cloud.

Data Movement
Upload locally from your computer, or set up remote jobs from various sources such as Soltlayer Swift, IBM Cloudant, or Amazon S3.

Connect Your Applications
After you have your data in place, you can connect your business intelligence or analytics-focused applications, and start running queries.

Where to Start

Learn
Learn what you can do with Db2 Warehouse on Cloud

Open
Open the console to get started with Db2 Warehouse on Cloud today!

Step 4: Upload data

- First download stats.csv data file from the course webpage, then upload to Db2

The image displays two side-by-side screenshots of the IBM Cloud Services dashboard, specifically the Db2 service details page.

Left Screenshot: This shows the main dashboard for the Db2 service. It features a timeline at the top with dates from 1/20 to 1/25. Below the timeline is a table with columns: SOURCE, FILENAME, TARGET, REQUESTED BY, ROWS LOADED, and ROWS REJECTED. A message "You have no recent load jobs" is displayed. A red oval highlights the "Load data" button at the bottom of the table.

Right Screenshot: This shows the "Load new" wizard. The first step, "Source", is active. It includes a "selection" section with a "Browse files" button, which is circled in red. The interface has tabs for Source, Target, Define, and Finalize.

Step 5: Select Scheme, Table Name and, Review Table

- Create a new table “EDSTATS”

The image displays two side-by-side screenshots of the IBM DB2 Warehouse on Cloud interface, specifically the 'Select a load target' step. Both screenshots show a 'Schema' dropdown menu and a 'Table' dropdown menu.

Left Screenshot (Schema Selection): The 'Schema' dropdown is open, showing several options: DASH6649, GOSALES, GOSALESDW, GOSALESHR, GOSALESMR, GOSALESR, and SAMPLES. The option 'DASH6649' is highlighted with a red circle. The 'Table' dropdown below it is empty, showing the message 'No entries found'.

Right Screenshot (Table Selection): The 'Table' dropdown is open, showing several options: DASH6649, EDSTATS, ENHORSCHMA, GOSALES, GOSALESDW, GOSALESHR, GOSALESMR, GOSALESR, and SAMPLES. The option 'EDSTATS' is highlighted with a red circle. The 'Schema' dropdown below it is empty, showing the message 'No entries found'.

Step 5: Select Scheme and Table Name

- Preview
 - You can change the column names if necessary
- Review settings
- Click Begin Load

Note the table name.
We need this in next steps

You are loading the file *EdStats-Country.csv* into **DASH6649.EDSTATS**

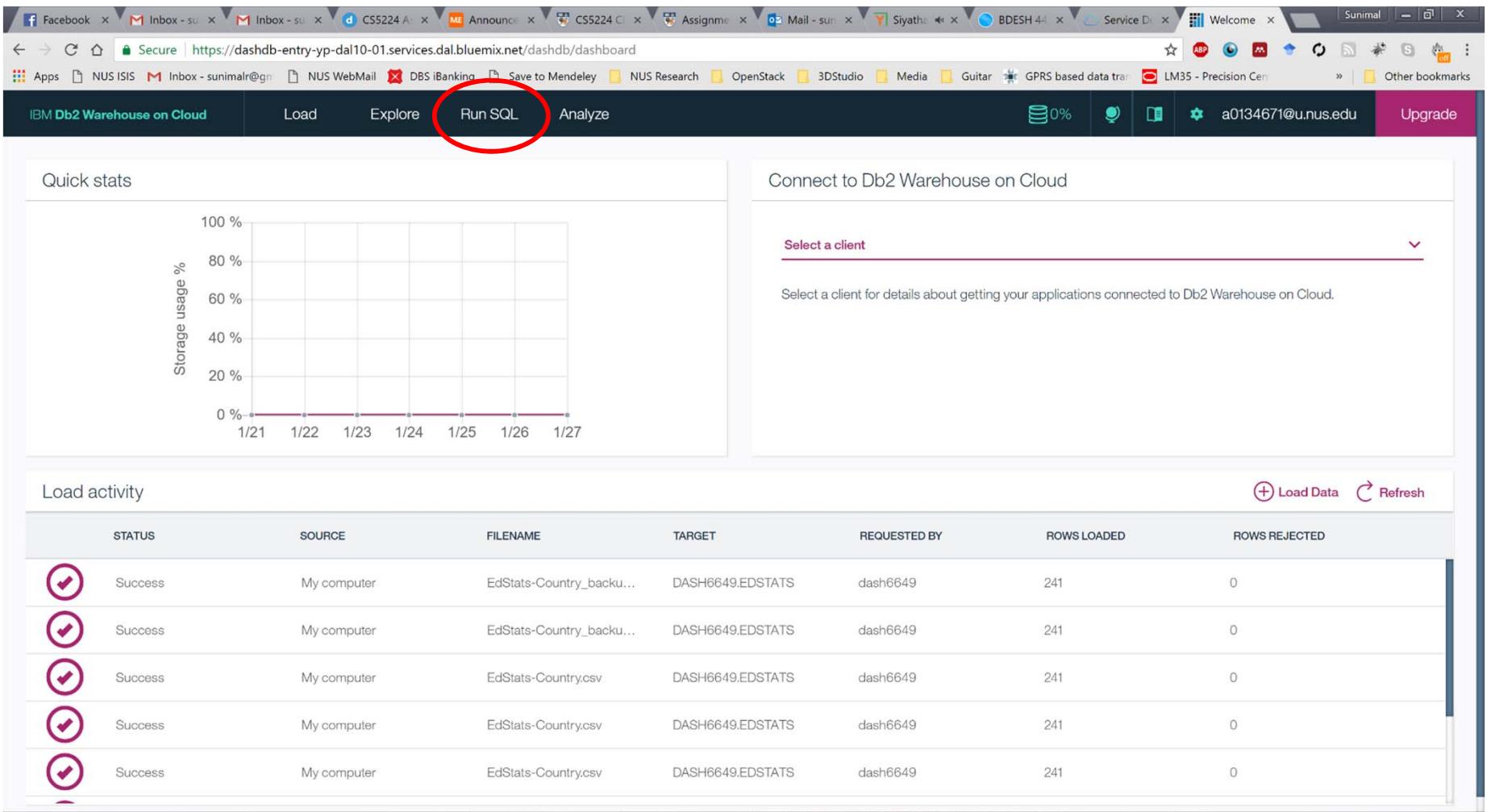
	COUNTRY CODE	SHORT NAME	TABLE NAME	LONG NAME	2-ALPHA CODE	CURRENCY UNIT	SPECIAL NOTES
1	AFG	Afghanistan	Afghanistan	Islamic State of Afghanistan	AF	Afghan afghani	Fiscal year end: March 20; reporting period for national accounts data: FY (from 2013 are CY)
2	ALB	Albania	Albania	Republic of Albania	AL	Albanian lek	
3	DZA	Algeria	Algeria	s Democratic Republic of Algeria	DZ	Algerian dinar	
4	ASM	American Samoa	American Samoa	American Samoa	AS	U.S. dollar	
5	ADO	Andorra	Andorra	Principality of Andorra	AD	Euro	
6	AGO	Angola	Angola	s Republic of Angola	AO	Angolan kwanza	April 2013 database update: Based on IMF data, national accounts data were revised for 2000
7	ATG	Antigua and Barbuda	Antigua and Barbuda	Antigua and Barbuda	AG	East Caribbean dollar	April 2012 database update: Based on official government statistics, national accounts data w.

Step 6: Run SQL Query

- Few things to note
 - SQL syntaxes slightly change across implementations (mysql, mssql, db2)
 - While loading file (step 5), the field names are shown in ALL CAPS, but this is inaccurate
 - To see correct column names, go to ‘Explore’ tab

Step 6: Run SQL Query

- Click “Run SQL” to enter the SQL console



The screenshot shows the IBM Db2 Warehouse on Cloud dashboard. At the top, there is a navigation bar with tabs: "IBM Db2 Warehouse on Cloud" (highlighted in blue), "Load", "Explore", "Run SQL" (circled in red), and "Analyze". Below the navigation bar, there are two main sections: "Quick stats" and "Load activity". The "Quick stats" section contains a chart titled "Storage usage %" showing a flat line at 0% usage from January 21 to January 27. The "Load activity" section displays a table of five successful load operations from "My computer" to "DASH6649.EDSTATS", each with 241 rows loaded and 0 rejected. A "Connect to Db2 Warehouse on Cloud" sidebar is also visible.

STATUS	SOURCE	FILENAME	TARGET	REQUESTED BY	ROWS LOADED	ROWS REJECTED
Success	My computer	EdStats-Country_backu...	DASH6649.EDSTATS	dash6649	241	0
Success	My computer	EdStats-Country_backu...	DASH6649.EDSTATS	dash6649	241	0
Success	My computer	EdStats-Country.csv	DASH6649.EDSTATS	dash6649	241	0
Success	My computer	EdStats-Country.csv	DASH6649.EDSTATS	dash6649	241	0
Success	My computer	EdStats-Country.csv	DASH6649.EDSTATS	dash6649	241	0

Step 6: Run SQL Query

- Find out countries that fall under the low income group and has their country code starting with S

```
SELECT  
"Country_Code","Long_Name","Currency_Unit","Income_Group","System_of  
_trade"  
FROM <INSERT Database table name here eg. DASH6649.EDSTATS>  
WHERE "Income_Group" LIKE 'Low%' AND "Country_Code" LIKE 'S%';
```

Database name should be set as required by referring to the database tables.

Step 6: Run SQL Query

- Click “Run All” and get the result

The screenshot shows the IBM Db2 Warehouse on Cloud interface. At the top, there's a toolbar with various icons and tabs like 'Load', 'Explore', 'Run SQL', and 'Analyze'. Below the toolbar is a dark header bar with the text 'IBM Db2 Warehouse on Cloud' and a user profile. The main area has two main sections: 'SQL editor' on the left containing some sample SQL code, and 'Results' on the right displaying a table of data. A red circle highlights the 'Run All' button in the toolbar.

SQL editor:

```
1 -- Enter SQL statements below or Load a SQL script into the editor from the toolbar.
2 SELECT "Country_Code", "Long_Name", "Currency_Unit", "Income_Group", "System_of_trade"
3 FROM DASH6649.EDSTATS
4 WHERE "Income_Group" LIKE 'Low%' AND "Country_Code" LIKE 'S%';
```

Results:

	COUNTRY_CODE	LONG_NAME	CURRENCY_UNIT	INCOME_GROUP	S
1	SLV	Republic of El Salvador	U.S. dollar	Lower middle income	Sp
2	SLB	Solomon Islands	Solomon Islands dollar	Lower middle income	Sp
3	SOM	Somali Democratic Republic	Somali shilling	Low income	

Milestone 5: Run SQL Query

- Did you successfully run the SQL?
- Let's wait until everyone is done!
- In the meantime:
 - Explore more about Db2 at:
<https://www.youtube.com/watch?v=zm4-VZoIYV0>

Example 3-Part 2: Analytics with R scripting

- R is a scripting language used for statistical computing including big data analysis, and data mining
- IBM Cloud offers R language support with IBM Data Science Experience service also known as DSX service
- Data loaded in an existing on-cloud database can be analyzed in a DSX project after properly linking the database to the DSX project

Objective: Perform simple analytics and visualization of data on cloud

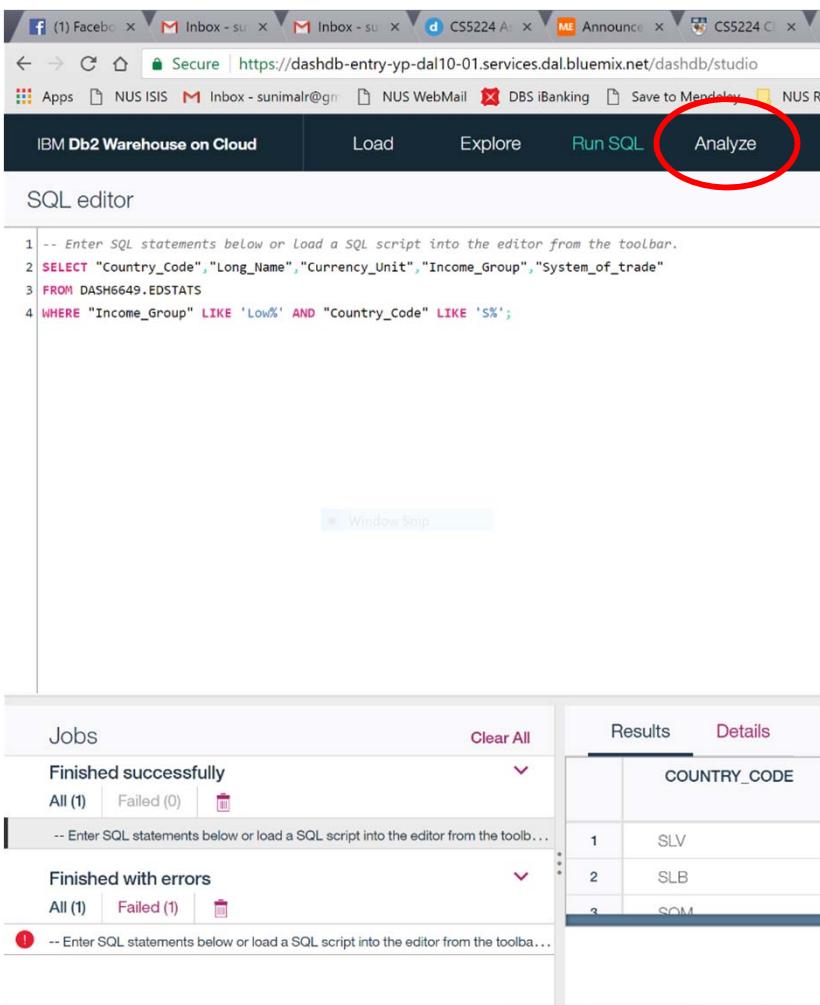
Example: Load economic data of countries to a an IBM Db2 database, and, execute simple R scripts for filtering data and visualization with plots

Data: Economic data of countries from United Nations (stats.csv)

This is a continuation of Example 3-Part 1

Step 7: Setup DSX

- Click on “Analyze” in the Db2 console
- Click “Launch DSX”

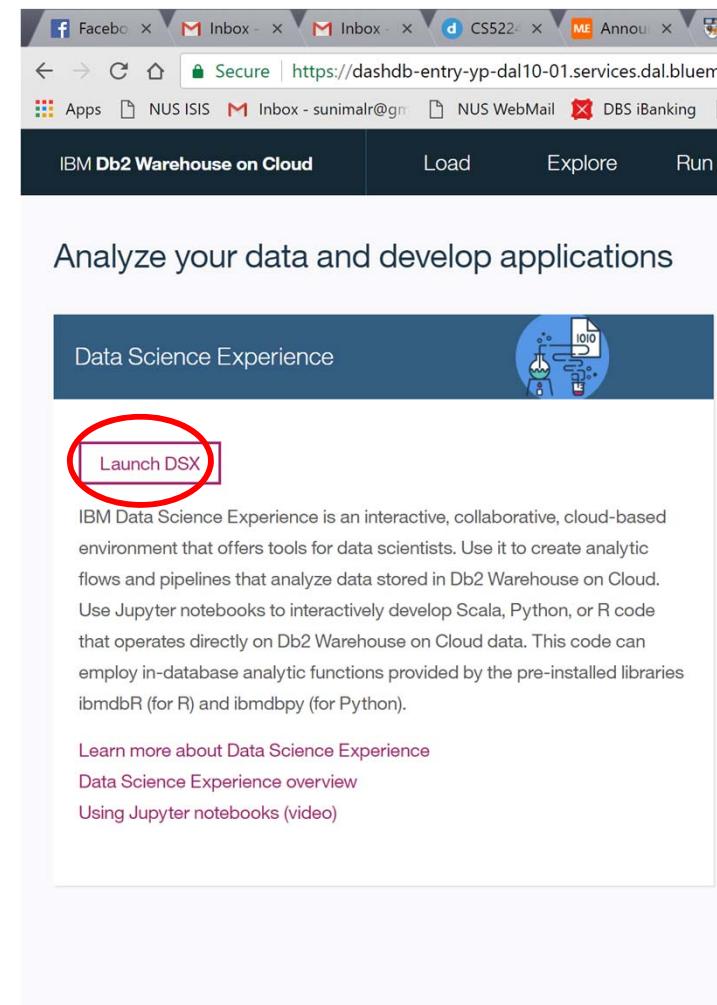


The screenshot shows the 'Analyze' interface of the IBM Db2 Warehouse on Cloud. At the top, there's a navigation bar with tabs: 'Load', 'Explore', 'Run SQL', and 'Analyze'. The 'Analyze' tab is highlighted with a red circle. Below the navigation bar is a 'SQL editor' section containing a snippet of SQL code:

```
1 -- Enter SQL statements below or Load a SQL script into the editor from the toolbar.
2 SELECT "Country_Code", "Long_Name", "Currency_Unit", "Income_Group", "System_of_trade"
3 FROM DASH6649.EDSTATS
4 WHERE "Income_Group" LIKE 'Low%' AND "Country_Code" LIKE 'S%';
```

At the bottom left, there's a 'Jobs' section showing two categories: 'Finished successfully' and 'Finished with errors'. The 'Results' tab is selected in the main content area, displaying a table with three rows:

COUNTRY_CODE
SLV
SLB
SOM



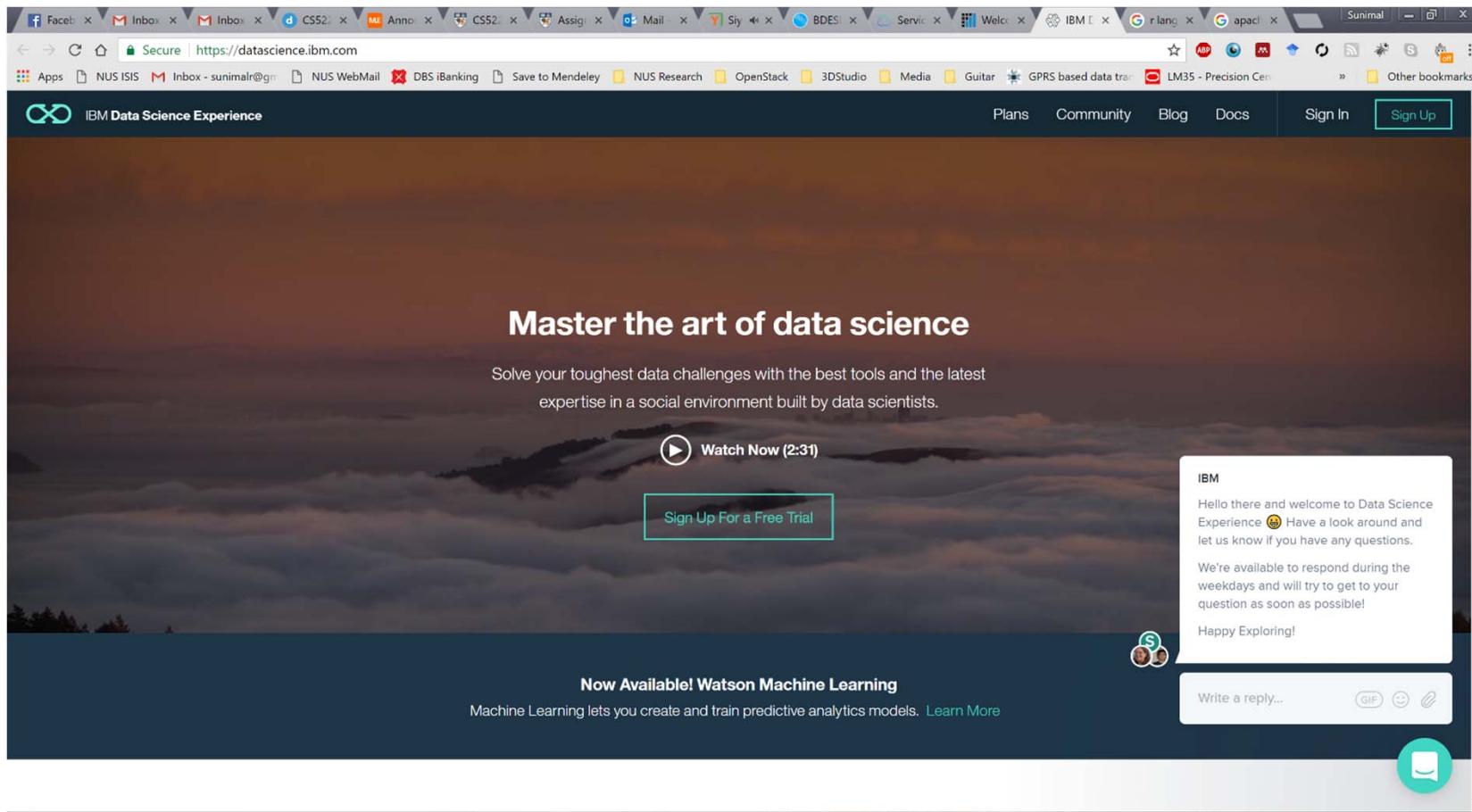
The screenshot shows the Data Science Experience landing page. The title 'Analyze your data and develop applications' is at the top. Below it is a section titled 'Data Science Experience' featuring an icon of laboratory glassware. A large button labeled 'Launch DSX' is circled in red. To the right of the button, there's a descriptive text block:

IBM Data Science Experience is an interactive, collaborative, cloud-based environment that offers tools for data scientists. Use it to create analytic flows and pipelines that analyze data stored in Db2 Warehouse on Cloud. Use Jupyter notebooks to interactively develop Scala, Python, or R code that operates directly on Db2 Warehouse on Cloud data. This code can employ in-database analytic functions provided by the pre-installed libraries ibmRdb (for R) and ibmRdbpy (for Python).

Below the text are three links:
[Learn more about Data Science Experience](#)
[Data Science Experience overview](#)
[Using Jupyter notebooks \(video\)](#)

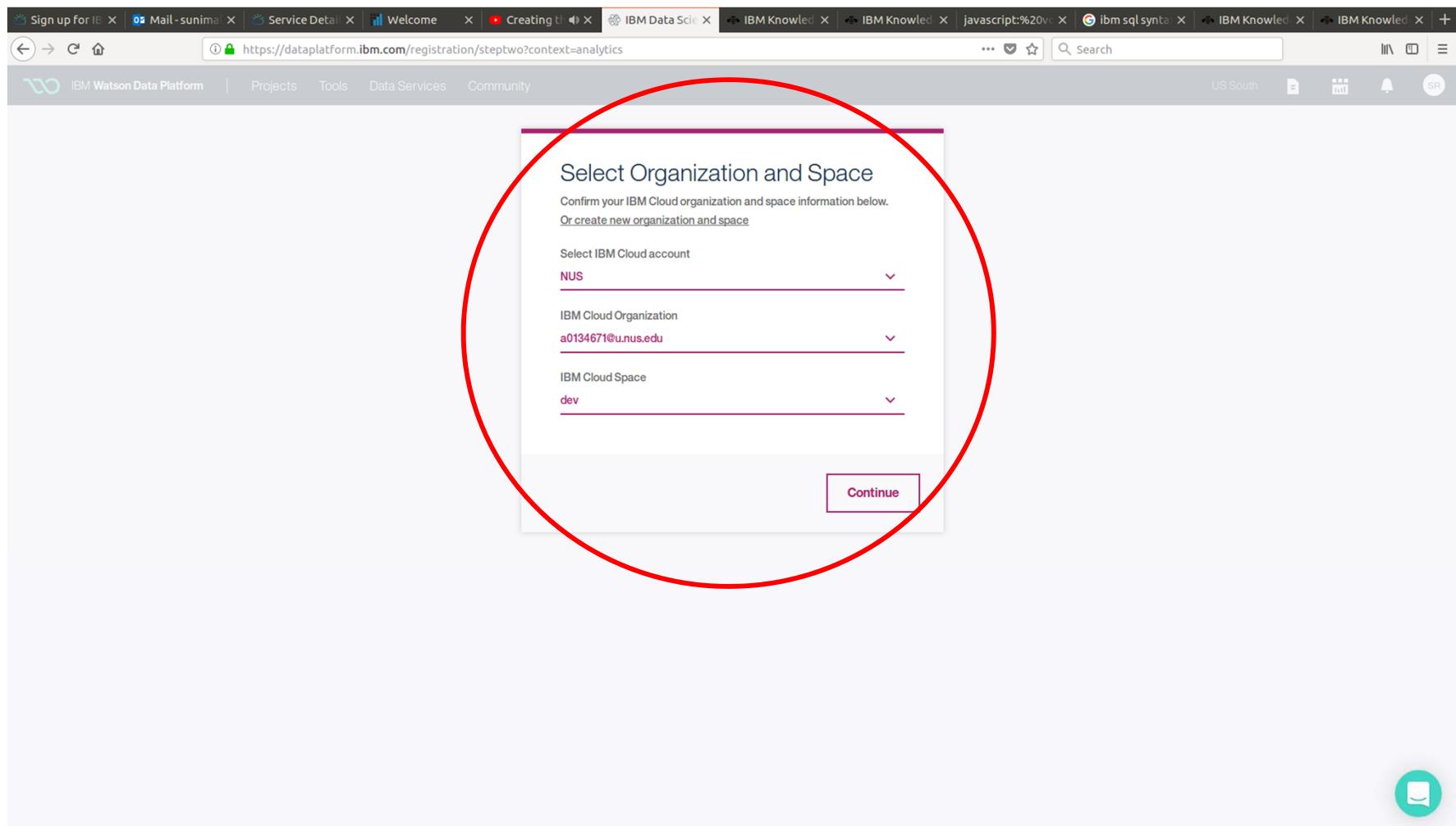
Step 7: Setup DSX

- Log in to DSX (You may be asked to sign up the first time)



Step 7: Setup DSX

- Select the relevant account details from drop down menus



Step 8: Create new DSX project

- Give a unique name for the DSX project
- Select storage type – IBM Cloud Storage
- Add a “Spark” service as the compute engine – *Spark is a fast and general cluster computing system for Big Data. It provides high-level APIs in Scala, Java, Python, and R, and an optimized engine that supports general computation graphs for data analysis.*

The screenshot shows the 'New project' page in the IBM Data Science Experience. The URL in the browser is <https://dataplatform.ibm.com/projects/new-project?context=analytics>.

Define project details

Name: csf224-a0134671
Description: Project description
Choose project options: Restrict who can be a collaborator (checkbox)

Define storage

① Select storage type:
 Object Storage (Swift API) IBM Cloud Object Storage (circled with red oval)

② Add: Add an object storage instance and then return to this page and click Refresh.

③ Refresh

Define compute engine

① Select Spark service:
 Spark service Add (circled with red oval)
Add IBM Analytics or Apache Spark, then return to this page and click Refresh.

② Refresh

Step 8: Create new DSX project

- Once the project is setup successfully, you should arrive at this portal

The screenshot shows the IBM Data Science Experience (DSX) project overview page for a project named "cs5224-a0134671".

Project Summary:

- Name:** cs5224-a0134671
- Last Updated:** Jan 25 2018
- Assets:** 0
- Bookmarks:** 0
- Collaborators:** 1

Date created: Jan 25 2018

Description: CS5224 Demo

Storage: 0% of 5 GB used

Collaborators: Sunimal Rathnayake (Admin)

Bookmarks: You currently have 0 bookmarks

Recent activity: A dashed box indicates where alerts related to the project will show when it is active.

Step 9: Connect to Db2 database

- We need to connect DSX project to our previously created Db2 database

The image consists of two side-by-side screenshots of the IBM Data Science Experience (DSX) interface.

Left Screenshot: Shows the DSX dashboard. On the left, there is a sidebar with various options: 'Connected data', 'Notebook', **Connection** (which is circled in red), 'Data asset', 'Model BETA', 'Streams flow BETA', 'SPSS Modeler flow BETA', and 'Environment'. Below the sidebar, it says '0 Bookmarks'. A note at the bottom left says 'A project will show here when a project is active.'

Right Screenshot: Shows the 'New connection' page. At the top, it says 'Your service instances in IBM Cloud'. Below this, there is a list of services. One item, 'a0134671-db IBM Db2 Warehouse on Cloud', is circled in red. The page also includes sections for 'IBM services' and 'Third-party services' with various other options like Cloud Object Storage, IBM Db2, etc.

If there are multiple databases in your IBM Cloud account, select the correct one that we created with EDSTATS csv file.

Step 9: Connect to Db2 database

- Give a name to data connection. Other fields should have been filled automatically.

The screenshot shows the 'New connection' configuration page in the IBM Data Science Experience. The URL in the browser is https://dataplatform.ibm.com/data/connections-v2/new?project_id=494ec190-2fcf-441f-84f9-9969bbe8880f&context=analytics.

Connection overview

Name	a0134671-db
Description	Connection created from an IBM Cloud instance

Connection details

Hostname or IP Address *	dashdb-entry-yp-dal10-01.services.dal.bluemix.net
Database *	BLUDB
Username *	dash6649
Password *	*****

Secure Gateway

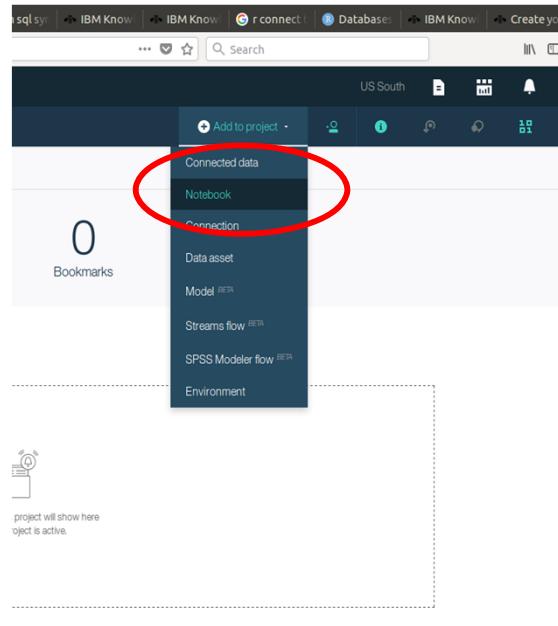
Use a secure gateway

Enter information for the selected data source

Create 1

Step 10: Create a new R Notebook

- We need to connect DSX project to our previously created Db2 database



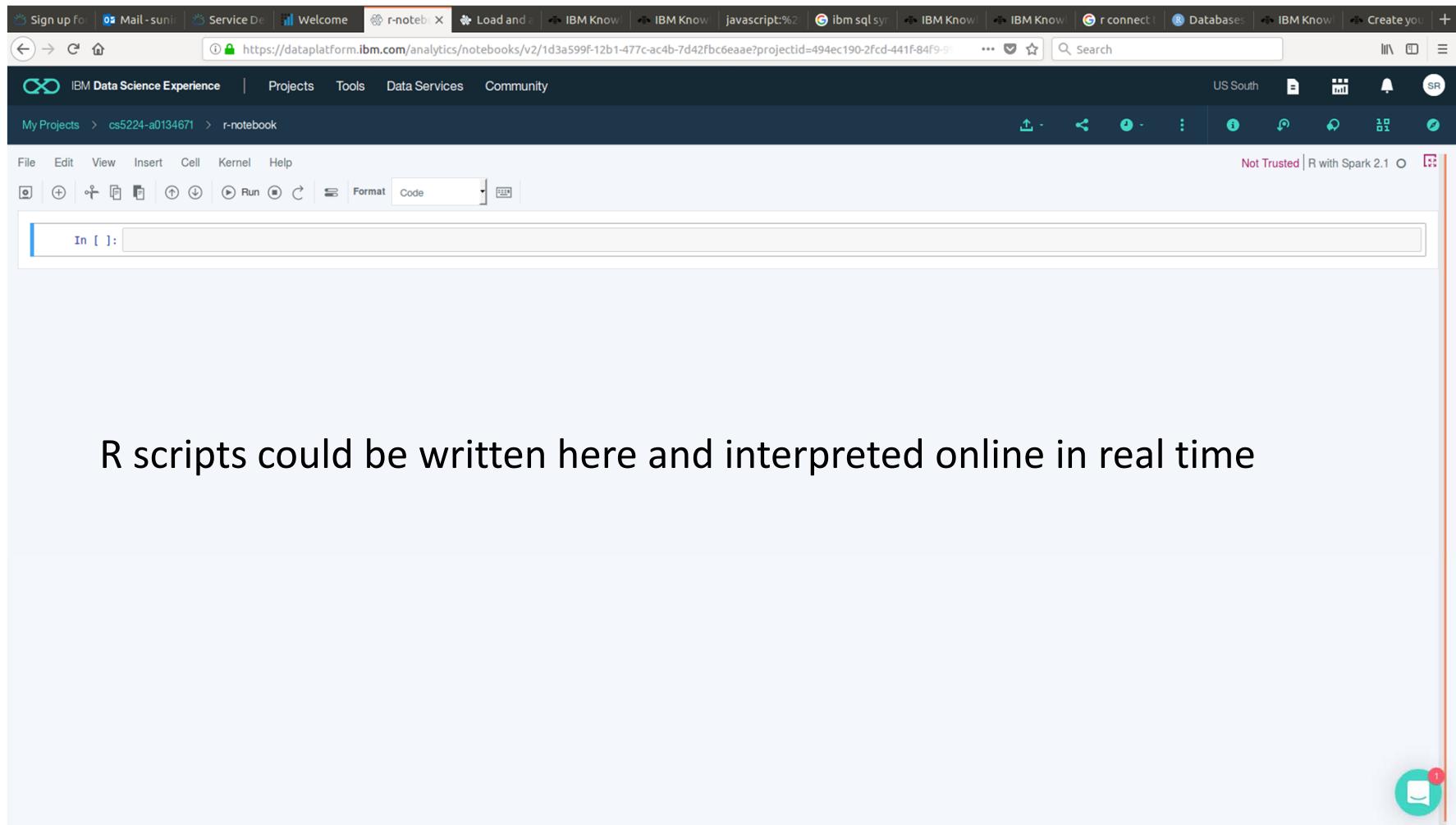
Make sure that you select R

A screenshot of the 'New notebook' creation form. It has tabs for 'Blank', 'From file', and 'From URL'. The 'Name*' field contains 'r-notebook' (with 40 characters remaining). The 'Description' field contains 'Analytics in R demo' (with 481 characters remaining). Under 'Language*', the 'R' option is selected (indicated by a red circle). Under 'Spark version*', the '2.1' option is selected. Under 'Spark service*', 'Spark-IM' is chosen. A note at the bottom says 'Associate this notebook with the Spark Service of your choice.'

Notebook is an online interpreter that allows scripting in different languages like R, Python etc.

Step 10: Create a new R Notebook

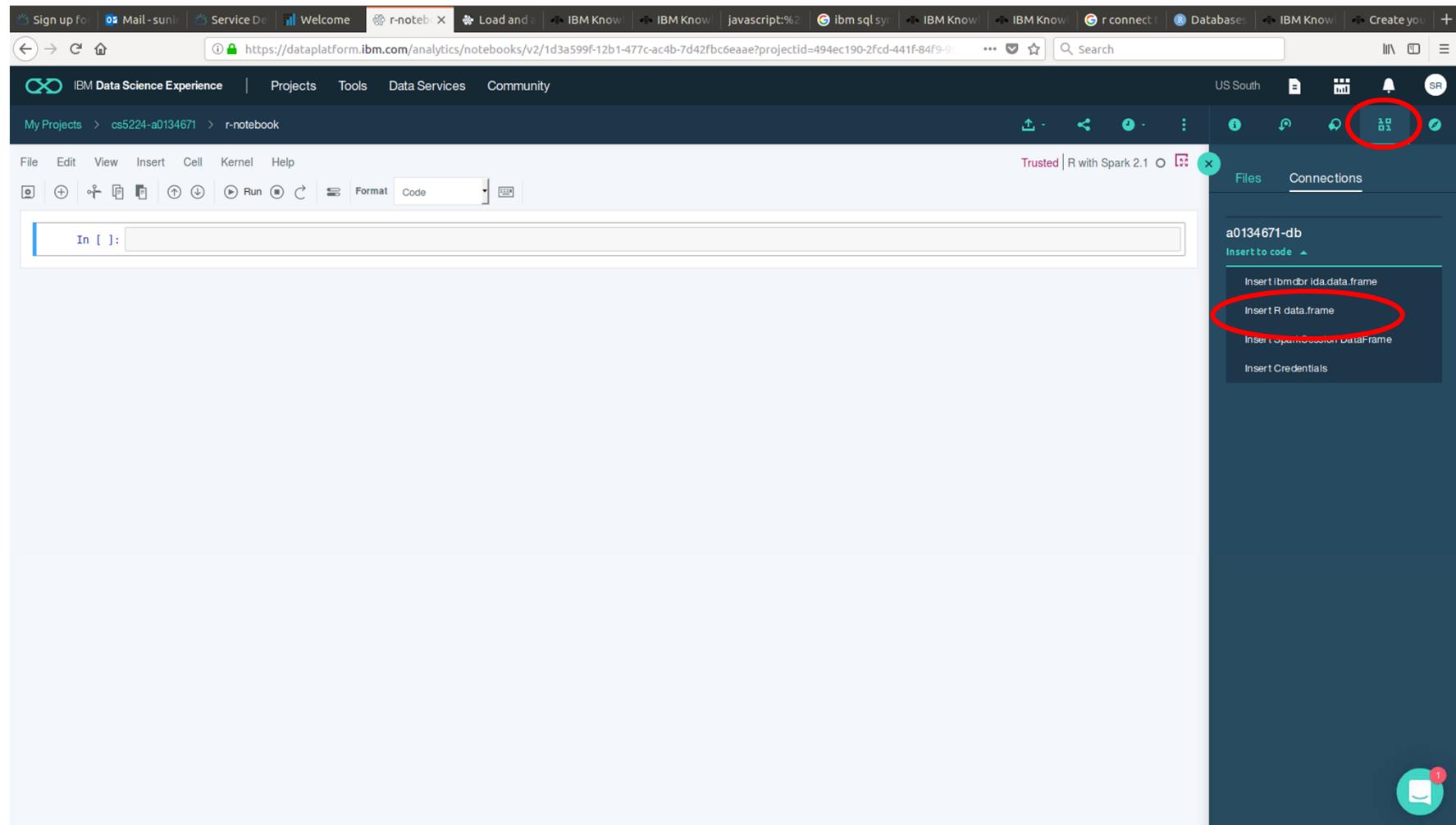
- Created R notebook should look like this



R scripts could be written here and interpreted online in real time

Step 11: Insert R data frame

- Insert a R data frame from the data connection.



Step 11: Insert R data frame

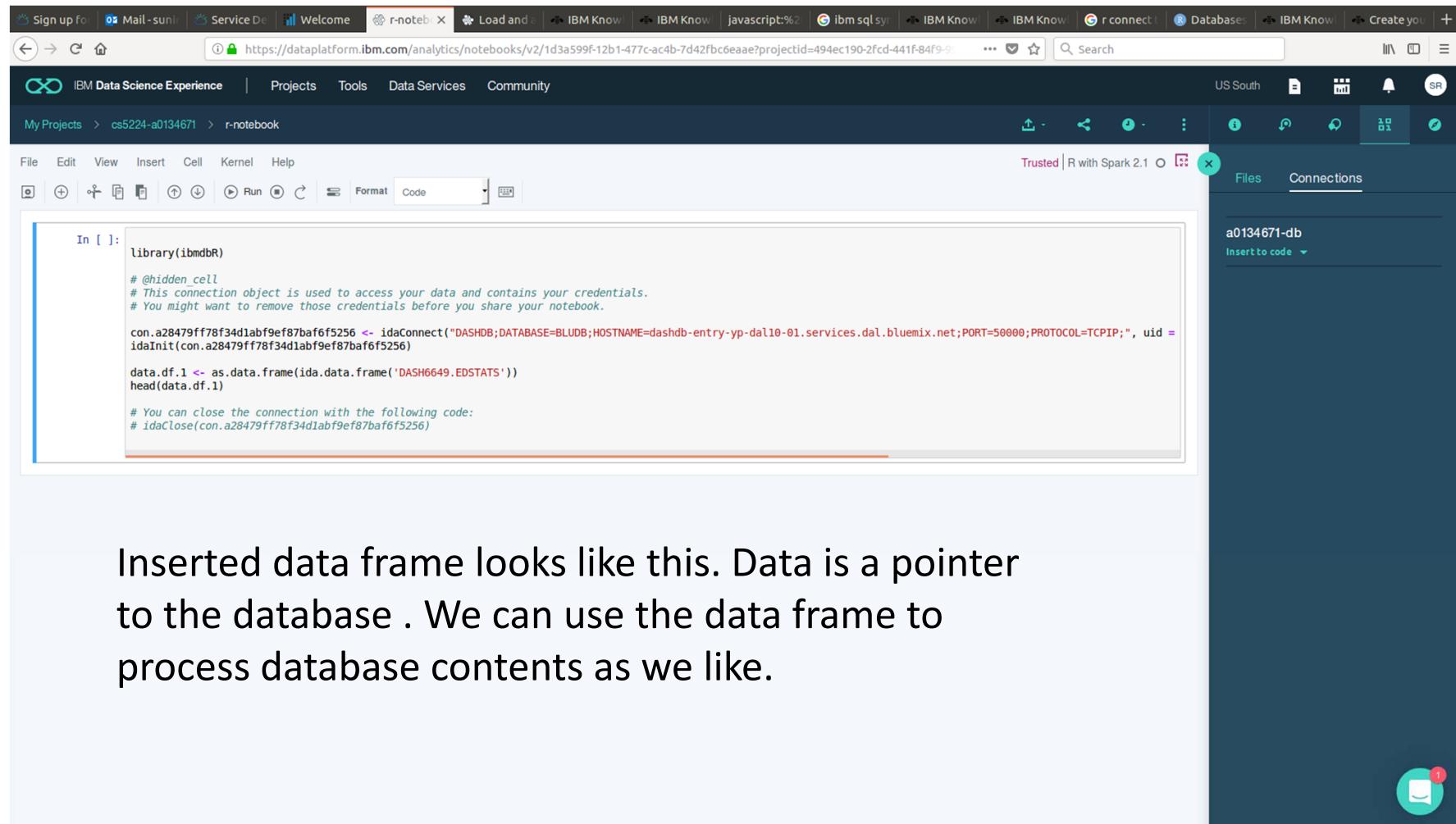
- Insert a R data frame from the data connection.

The screenshot shows the IBM Data Science Experience R-notebook interface. The main area is a code editor with a single cell labeled "In []". To the right is a sidebar titled "Connections" which lists a connection named "a0134671-db". Under this connection, there are dropdown menus for "schema:" (set to "DASH6649") and "Table:" (set to "EDSTATS"). A red circle highlights the "Insert Code" button at the bottom of the sidebar. The top navigation bar includes links for "Sign up for", "Mail", "Service Desk", "Welcome", "r-notebook", "Load and", "IBM Know", "IBM Know", "javascript%", "ibm sql", "IBM Know", "IBM Know", "r connect", "Database", "IBM Know", and "Create your". The URL in the address bar is <https://dataplatform.ibm.com/analytics/notebooks/v2/1d3a599f-12b1-477c-ac4b-7d42fb6eaae?projectId=494ec190-2fc4-441f-84f9-9>.

Select the data scheme name and table name (EDSTATS) from the drop down lists and click “Insert Code”

Step 11: Insert R data frame

- Insert a R data frame from the data connection.



The screenshot shows an R notebook in the IBM Data Science Experience environment. The code cell contains R code to connect to a database and create a data frame:

```
In [ ]:
library(ibmdbR)

# @hidden_cell
# This connection object is used to access your data and contains your credentials.
# You might want to remove those credentials before you share your notebook.

con.a28479ff78f34d1abf9ef87baf6f5256 <- idaConnect("DASHDB;DATABASE=BLUDB;HOSTNAME=dashdb-entry-yp-dal10-01.services.dal.bluemix.net;PORT=50000;PROTOCOL=TCPIP;", uid =
idaInit(con.a28479ff78f34d1abf9ef87baf6f5256)

data.df.1 <- as.data.frame(ida.data.frame('DASH6649.EDSTATS'))
head(data.df.1)

# You can close the connection with the following code:
# idaClose(con.a28479ff78f34d1abf9ef87baf6f5256)
```

The notebook interface includes a toolbar, a file menu, and a sidebar with a 'Connections' tab showing a database connection named 'a0134671-db'.

Inserted data frame looks like this. Data is a pointer to the database . We can use the data frame to process database contents as we like.

Step 11: Insert R data frame

- Insert a R data frame from the data connection.

The screenshot shows an R-notebook interface within the IBM Data Science Experience. A red arrow points from the text "Click 'Run' to execute the script" down to the "Run" button in the toolbar above the code editor. The code editor contains R script code for connecting to a database and creating a data frame. The right sidebar shows a database connection named "a0134671-db".

```
library(ibmdbR)

# @hidden_cell
# This connection object is used to access your data and contains your credentials.
# You might want to remove those credentials before you share your notebook.

con.a28479ff78f34d1abf9f87baf6f5256 <- idaConnect("DASHDB;DATABASE=BLUDB;HOSTNAME=dashdb-entry-yp-dal10-01.services.dal.bluemix.net;PORT=50000;PROTOCOL=TCPIP;", uid = idaInit(con.a28479ff78f34d1abf9ef87baf6f5256)

data.df.1 <- as.data.frame(ida.data.frame('DASH6649.EDSTATS'))
head(data.df.1)

# You can close the connection with the following code:
# idaClose(con.a28479ff78f34d1abf9ef87baf6f5256)
```

Click "Run" to execute the script

Inserted data frame looks like this. Data is a pointer to the database . We can use the data frame to process database contents as we like.

Step 12: Write R script to Plot Data

- # Load the plotting package
 - `library(ggplot2)`
- # Plot the count of countries across different income groups
 - #Classify the levels and label them: Create a factor

```
data_frame1<- as.data.frame(ida.data.frame('DASH6649.EDSTATS'))[ ,c('Currency_Unit', 'Income_Group', 'Region', 'Short_Name')]
```

```
data_frame1$"Income_Group" <- factor(data_frame1$"Income_Group",levels=c("Low income","Lower middle income","Upper middle income","High income: nonOECD","High income: OECD"), labels=c("LI","LMI","UMI","HIN","HIO"))
```
 - #Count the income groups

```
counts <- table(data_frame1$"Income_Group")
```
 - #Plot counts versus the labelled income groups

```
barplot(counts, main="Income Distribution",xlab="Income Group")
```

Step 12: Write R script to Plot Data

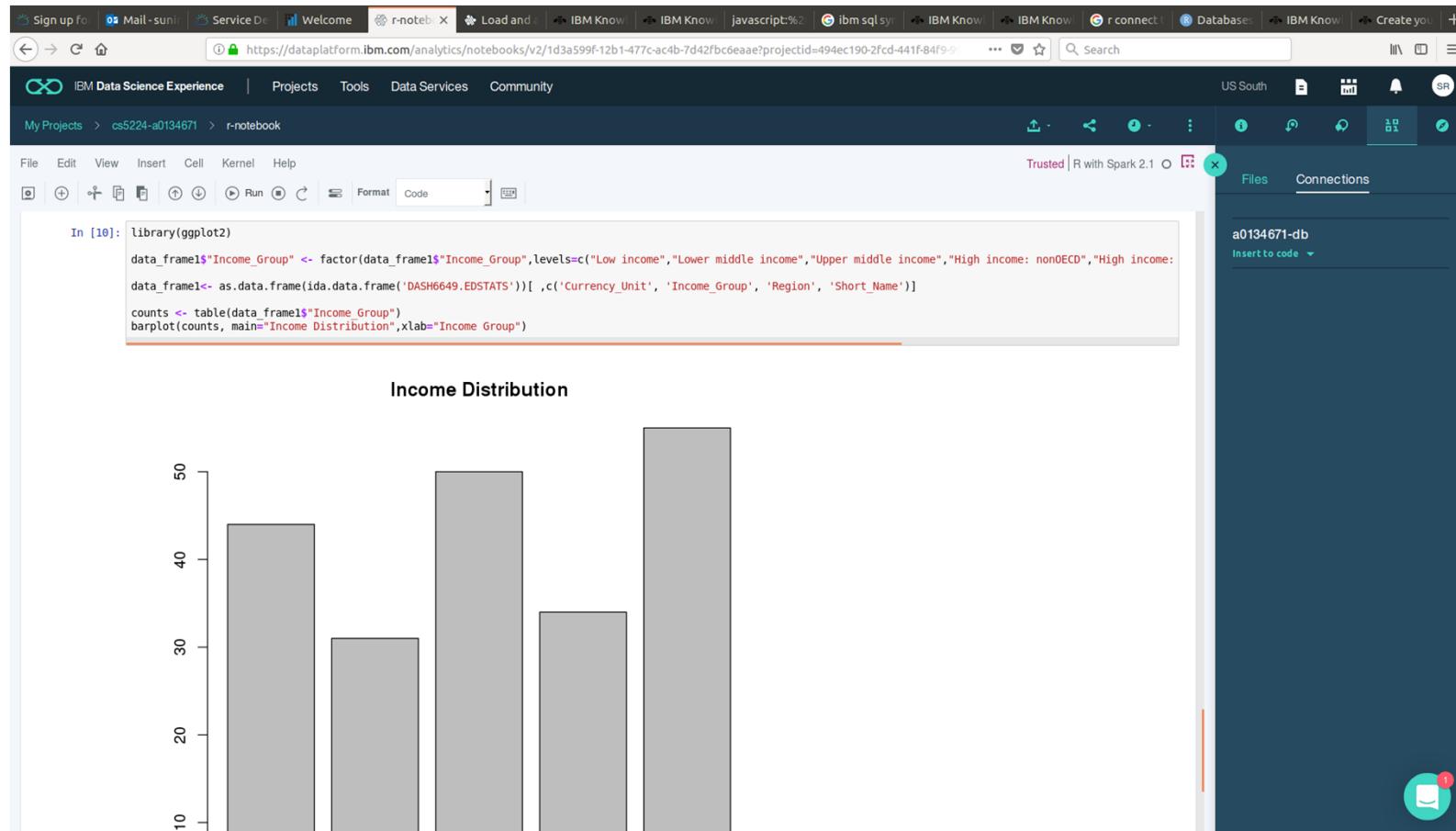
```
library(ggplot2)

data_frame1<- as.data.frame(ida.data.frame('DASH6649.EDSTATS'))[ ,c('Currency_Unit', 'Income_Group', 'Region', 'Short_Name')]

data_frame1$"Income_Group" <-
factor(data_frame1$"Income_Group",levels=c("Low income","Lower middle income","Upper middle income","High income: nonOECD","High income: OECD"),
labels=c("LI","LMI","UMI","HIN","HIO"))

counts <- table(data_frame1$"Income_Group")
barplot(counts, main="Income Distribution",xlab="Income Group")
```

Step 13: Output the Plot



Milestone 6: Visualize data

- Did you successfully visualize the data retrieved from the database?
- Congratulations! This is the last step of the hands on activity. You may refer to the references section at the end of the presentation slides to learn more or search on the web.
- **Next step: Lab Assessment**

Summary

- hands-on with BlueMix
- simple web server application using node.js and Db2
- integrating IBM Watson personality insights service with a node.js web application
- simple SQL query using Db2 Warehouse SaaS
- simple R script to plot results of data on Db2 and DSX

References

- [NodeJS-DashDB example](#)
- Demo: Getting Started with [Node.js](#) on Bluemix 2016- https://www.youtube.com/watch?v=sHhNoV-S_I&list=PLJxa6lsF8C5qFbRinR2ZaEZ3AiIFTdLqx (13 mins)
- [IBM Bluemix The Cloud Platform for Creating and Delivering Applications](#), IBM Redbooks, 2015.
- Bluemix users: <https://www.ibm.com/cloud-computing/bluemix/case-studies>
- Watson SDK help: <https://github.com/watson-developer-cloud/node-sdk>
- Watson startup guide: <https://console.bluemix.net/docs/services/watson/developing-nodejs.html#developing-a-watson-application-in-node-js>
- IBM dashDB:
<http://www-01.ibm.com/support/knowledgecenter/SS6NHC/com.ibm.swg.im.dashdb.kc.doc/welcome.html>
- SQL Query: <http://www.w3schools.com/sql/>
- R Language:
<http://cran.r-project.org/doc/manuals/r-release/R-intro.html>
- Plotting: <http://docs.ggplot2.org/current>
- IBM DSX Getting Started Guide: <https://datascience.ibm.com/docs/content/getting-started/get-started.html>
- IBM DSX Tutorials: <https://github.com/IBMDatascience/dsx-tutorials>