Build a DB2 CLI console to manage SQLDB databases

Use Ruby, Tmate, and Dropbox to create a lightweight utility app on IBM Bluemix

Manage your SQLDB databases with ease, using an application you can quickly build and deploy on the IBM cloud platform, Bluemix.

Manage your SQLDB databases with ease, using an application you can quickly build and deploy on the IBM cloud platform, Bluemix. The application illustrated in this tutorial lets you open a console in an app running on the IBM Bluemix cloud platform, and run familiar DB2 commands, such as `db2 describe table mytable`, and `db2 list tables for all` against your SQLDB. You can also export your SQLDB database and store a backup in the cloud, and even use a backup image to clone a different SQLDB instance.

All these capabilities are available in the DB2climate application, and I'll show you how to build it.

"DB2climate is a fast, lightweight utility that helps you easily manage your SQLDB databases."

DB2climate is a CLI console app based on Dan Higham's Tmate server buildpack, that helps you manage your Bluemix SQLDB service. It opens an SSH connection to the application warden container, which has the DB2 runtime client, the Dropbox CLI, and the Dropbox API client preloaded and configured. You can therefore run all supported DB2 commands against your SQLDBs and move data in and out of Bluemix using Dropbox cloud storage.

Get the code

In short, DB2climate is a fast, lightweight utility that helps you manage your SQLDB databases with ease. It also allows you to perform tasks not currently available with SQLDB's built-in web-based Managed Database Console, such as the following:
1. Import/load a file with IXF format to a database
2. Export tables using a SQLDB console
3. Migrate data between SQLDB instances using a backup image
4. Add non-ephemeral disk space to Bluemix
5. Sync data to SQLDB

What you'll need to build your app

- A Bluemix account so you can use the SQL Database service.
- The Cloud Foundry cf command line tool version v6.5.1 or later.
- A terminal program with SSH capability, such as Babun or PuTTY.
- A Dropbox account.
- A Dropbox API app. (Once your API app is successfully created, note the App Key and App secret.)
- Some familiarity with DB2 runtime commands.

Step 1. Deploy DB2climate to Bluemix

1. Log in to Bluemix.
   
   usage: cf login [-a API_URL]
   example: cf login -a https://api.ng.bluemix.net

2. Create an instance of the SQL Database service.
   
   usage: cf create-service SERVICE PLAN SERVICE_INSTANCE_NAME
   example: cf create-service sqldb sqldb_small SQLDB_001

3. Create a git clone of this repository.
   
   git clone https://hub.jazz.net/git/felixf/bluemix-db2-cli-mate

4. Navigate to the cloned application directory.
   
   cd bluemix-db2-cli-mate

5. From the cloned DB2climate app directory, push the app using the --no-start flag so you can bind your SQLDB service before starting it.
   
   usage: cf push APP [--no-start]
   example: cf push db2climate --no-start

6. Bind the SQLDB service to the new app.
   
   usage: cf bind-service APP SERVICE_INSTANCE_NAME
   example: cf bind-service db2climate SQLDB_001

7. Start the app.
   
   usage: cf start APP
   example: cf start db2climate

8. The start process involves downloading and installing a few packages, including Ruby runtime, DB2 v9.7 runtime client, Dropbox Linux client, Dropbox API client, and a repackaged Tmate server. The complete package should take just a few minutes to install and be operational.
   Note that when Bluemix first shows the application is running, initialization may still in progress. A complete initialized application should be around 360MB in size. Anything above that suggests that the application is not ready. Wait for 10 seconds and run the following command to verify:
   
   cf app db2climate
That's it! DB2climate has now successfully deployed to the Bluemix cloud.

**Step 2. SSH into the DB2climate app container**

1. Issue the following command:
   
   ```
   cf files db2climate logs/tmate.log
   ```

2. The last four lines of `tmate.log` should look like this:

   ![Tmate log output]

3. The connection URL is in the last line of the log. The two 1000 numbers are the height and width of your terminal, followed by the connection URL.

   If you are using a Windows PC, you will need to have a Cygwin-like terminal program. I use the Babun windows shell, which you can download from [https://github.com/babun/babun](https://github.com/babun/babun).  

   **Note:** Before attempting to make an SSH connection for the first time, make sure you run the `ssh-keygen` program in the home directory; otherwise, you'll get a connection denied error.
4. After running the `ssh-keygen` program, you can SSH into the app container.

5. If you prefer to use PuTTY, please refer to this blog on generating private keys and make connections.

### Step 3. Set up Dropbox as non-ephemeral storage

By default, there is no persistent storage available for DB2climate, so all changes written to disk are lost if the application is stopped or restarted. Therefore, we need to add some kind of non-ephemeral storage for our database backups and exported tables.

In this tutorial, I show you how to use Dropbox as your persistent storage. This is not the only solution, but it's a quick way to get the app working. The advantage of using Dropbox API over a Dropbox Linux client is that Dropbox API does not synchronize with cloud storage automatically, so it is safe to use your existing Dropbox account.

1. DB2climate comes with a copy of Dropbox core API client installed. To avoid having to enter Dropbox API keys each time DB2climate is restarted, you can store the Dropbox API keys securely inside the SQLDB bound to the DB2climate app.
   Run the `set_dbox_api.rb` script and follow the prompt to enter the API key and API secret. Next, select the appropriate access type. The access type must match the permission type of your API app when the API app was created.
2. Having stored the Dropbox API keys to the database, we can start using the Dropbox API program.

Run the `dropbox-api` command. Copy/paste the URL displayed on the screen to a browser. Log on to your Dropbox account and click **Allow**. After you have done that, return to your SSH console and press **Enter** to finish the setup.

Congratulations, you have successfully linked your app container to your Dropbox account!

3. Run `dropbox.api help` to get a list of available commands. For instance, to upload the `local.tar` file from the container to the Dropbox root directory, issue the following command:

```
$   dropbox-api put local.tar dropbox: />
```

To list all the files in your Dropbox account, issue:

```
$   dropbox-api ls
```

**Step 4. Use DB2climate as a DB2 client**

You can run all supported DB2 CLI commands against your SQLDBs after using SSH to access the warden container. All TCPIP nodes and databases will have been automatically cataloged for you.
In addition, in order to avoid having users refer to the `VCAP_SERVICE` file for database credentials when running DB2 commands, the following scripts help you easily manage your databases.

**General-purpose scripts**

- **Script Name: ls_db.rb**
  Purpose: List all databases bound to the app and display their credentials.
  
  Syntax: `$ ls_db.rb`
  Example: `ls_db.rb`

- **Script Name: catalog_db.rb**
  Purpose: Catalog the TCPIP node and database.
  
  Syntax: `$ catalog_db.rb db_name` (where `db_name` is the instance name of your created SQLDB service)
  Example: `$ catalog_db.rb SQLDB_001`

  **Note:** This script is executed automatically on your SSH connection to set up your database environment, so you don't need to run it separately.

- **Script Name: connect_db.rb**
  Purpose: Connect to the database.
  
  Syntax: `$ connect_db.rb db_name` (where `db_name` is the instance name of your created SQLDB service)
  Example: `$ connect_db.rb SQLDB_001`

- **Script Name: gen_dbSchema.rb**
  Purpose: Generate a database schema.
  
  Syntax: `$ gen_dbSchema.rb [database_db_name] schema_name`
  Example 1: `$ gen_dbSchema.rb BX` (single db instance bound to the app)
  Example 2: `$ gen_dbSchema.rb SQLDB_002 BX` (multi db instances bound to the app)

- **Script Name: cr_sample_tables.rb**
  Purpose: Create and populate two sample tables for testing export and import scripts.
  
  Syntax: `$ cr_sample_tables.rb db_Name`
  Example: `$ cr_sample_tables.rb SQLDB_001`

- **Script Name: ls_tables.rb**
  Purpose: List all tables of a given schema name.
  
  Syntax: `$ ls_tables.rb [db_name] schema_name`
  Example 1: `$ ls_tables.rb BX` (single db instance bound to the app)
  Example 2: `$ ls_tables.rb SQLDB_002 BX` (multi db instances bound to the app)

**Export and import table scripts**

- **Script Name: export_table.rb**
  Purpose: Export a table in supported format (.ixf or .del).
Output file: the name of exported table with export format as extension
syntax: $ export_table.rb [db_name] schema_name table_Name export_format
Example 1: $ export_table.rb BX country ixf (single db instance bound to the app)
Example 2: $ export_table.rb SQLDB_002 BX CITY del (multi db instances bound to the app)

• Script Name: import_table.rb
Purpose: Import a table in supported format (.ixf or .del) with all supported import modes.
syntax: $ import_table.rb [db_name] schema_name table_file_name [import_mode]
Supported Import mode are: 1=insert(default), 2=insert_update, 3=replace, 4=replace_create (ixf only), 5=create (ixf only)
Example 1: $ import_table.rb BX country.ixf (single db instance bound to the app with default insert mode)
Example 2: $ import_table.rb SQLDB_002 BX country.ixf 2 (multi db instances bound to the app with insert_update mode)

Backup and restore database scripts

Unfortunately, we are not able to run the built-in `db2 backup` or `db2 restore` utilities with DB2climate. This is because the `db2 get authorizations` command "tells all," so the default user does not have enough authority to perform backup and restore.

```
vmcap@182b95hdli:~$ db2 get authorizations

Administrative Authorizations for Current User

Direct SYSDM authority= NO
Direct SYSCtrl authority = NO
Direct SYSCNTM authority = NO
Direct DBADM authority = YES
Direct CREATETAB authority = YES
Direct BINDADD authority = YES
Direct CONNECT authority = YES
Direct CREATE NOT_FENCED authority = YES
Direct IMPLICIT_SCHEMA authority = YES
Direct LOAD authority = YES
Direct QUIESC_TYPE authority = YES
Direct CREATE_EXTERNAL_ROUTINE authority = YES
Direct SYSPRM authority= NO
```

The DB2 backup/restore utility requires an instance-level authority such as SYSDM, SYSCtrl, or SYSCNTM. The default user, however, has only DBADM authority, so any attempts to run backup and restore will result in the following error:
The workaround I recommend is to use the `db2move` command, which I regularly use to migrate databases from one OS to another. This command exports all tables in IXF format so they can be imported back into a new database. It is fully automated and extremely easy to use. Please note that `db2move` does not come with the DB2 runtime client. It is available only on server editions, but you can download DB2 Express-C free of charge.

The Bluemix application container restricts size to a maximum of 2GB, so installing a full copy of DB2 Express-C server would use up too much container space. Luckily, we need just three files to get the `db2move` utility up and running: `db2move`, `db2common.bnd`, and `db2move.bnd`.

Put `db2move` into the `sqlib/bin` directory and the two `.bnd` files into the `sqlib/bnd` directory. This will give you a fully working `db2move` utility.

**Note:** The `db2move` utility has already been packaged with the DB2climate application, so you don’t need to perform the above tasks for this sample app.

Here are the backup and restore database scripts:

- **Script Name:** `export_db.rb`  
  **Purpose:** Export all database tables in `.ixf` format using the `db2move export` command.  
  **Output file:** `db_name` with a `.tar.gz` extension in the current directory.  
  **Syntax:** `$ export_db.rb db_name`  
  **Example:** `$ export_db.rb SQLDB_001`  
  **Sample output file:** `$ SQLDB_001.tar.gz`

- **Script Name:** `import_db.rb`  
  **Purpose:** Import all database tables using the `db2move import` command. Use it to restore to an existing DB or to clone a new one.  
  **Syntax:** `$ import_db.rb db_name db_backup_file_name`  
  **Example:** `$ import_db.rb SQLDB_001 SQLDB_001.tar.gz`

**DB2climate tips and known issues**

**Tips**

- Bluemix allows multiple apps to bind to the same service, and vice versa. Therefore, DB2climate can be used as a utility app to bind to the same SQLDB service to which your
primary app binds. As the next figure shows, we have bound the SQLDB_001 instance to both the Bluemix101-FF and the DB2climate apps. In this example, Bluemix101-FF is our primary app, which uses the SQLDB_001 as its back-end database, whereas DB2climate is our utility app that is used to help us manage the DB instance.

• **Important:** When you have finished using the terminal, shut down your terminal program by clicking the X icon and confirm **OK**. This leaves your Tmate session running so you can reconnect to it using the same Tmate credentials. If you type `exit` at the Tmate terminal, it will shut down the Tmate server and you will have to restart the DB2climate application. This will generate a new Tmate credential. On top of that, you will have to set up the Dropbox account again and re-catalog all the databases.

**Known issues**

• If you unbind your SQLDB instance from the DB2climate app and then later rebind it, you will no longer be able to run some of the DB2 commands successfully. While some commands such as `db2 get dbm cfg` and `db2 get authorization` continue to work, other
commands like `db2 list tables` and `db2 select * from ...` will yield errors, as shown:

```
<table>
<thead>
<tr>
<th>db_name</th>
<th>db_label</th>
<th>host</th>
<th>username</th>
<th>password</th>
<th>database_id</th>
<th>db_port</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLDB_001</td>
<td>SQLDB</td>
<td>23.246.23.46</td>
<td>scctupf</td>
<td>2xyswvit80t7</td>
<td>I_088342</td>
<td>50000</td>
</tr>
<tr>
<td>DB2_TEST</td>
<td>SQLDB</td>
<td>192.135.204.188</td>
<td>sppwbgds</td>
<td>z5gxpqhjepmo</td>
<td>I_353945</td>
<td>50000</td>
</tr>
</tbody>
</table>
```

Because of the above restriction, you should not unbind your SQLDB instance from the DB2climate application at any time. If you accidentally do, the only way to restore your ability to manage your SQLDB instance via DB2climate is to delete your SQLDB instance, re-create a new instance, and restore the database from your Dropbox cloud storage backup.

- Occasionally, when you deploy or start the DB2climate app, the Tmate server might not initialize properly, and as a result the Tmate URL may not be revealed. If that happens, simply stop and restart the DB2climate app. That should fix the issue.

**Conclusion**

The DB2climate app that you can build using the instructions in this tutorial is a lightweight utility app to help you manage your SQLDB databases. It allows you to SSH into your application container, from which you can run your familiar DB2 commands using the DB2 runtime client.

DB2climate also provides some additional capabilities that are currently missing from the default Bluemix SQLDB web management console, such as import/export tables in IXF format, and backup/restore databases to different instances of SQLDB databases. In addition, DB2climate employs Dropbox to provide persistent storage on Bluemix.

**RELATED TOPICS:**  
- DB2 CLI (command line interface)  
- SQLDB database  
- Ruby
