Hello World:

Best practices for your Db2-based cloud app

Dr. Henrik Loeser / hloeser@de.ibm.com / @data_henrik

Agenda

- Db2 Software Development Kits
- Cloud Deployment Models
- Local development
- Security Recommendations and Best Practices
- Performance and Monitoring
- Summary

Db2 SDKs

...ADO.NET Basic C C++ C# CLI Cobol Fortran Java JDBC Node.js ODBC Perl PHP Python R REXX Ruby Scala...

IBM Db2 and Programming Languages (1|3)

- Db2 SDKs for many programming languages, hosted on GitHub: https://github.com/ibmdb
- Supports Node.js and Python
- Specific Python SDKs for DBI, Django, SQLAlchemy, …
- Other packages part of IBM data server clients

IBM Db2 and Programming Languages (2|3)

Node.js:

```
// open connection
var conn=ibmdb.openSync(dsn);
// insert values
var data=conn.querySync("insert into
                events (shortname, location, ...)
                values(?,?, ...)", eventValues);
// close connection
conn.closeSync();
```

IBM Db2 and Programming Languages (3|3)

Python:

```
// open connection
conn = ibm_db.connect(ssldsn, "", "")

// prepare and execute statement
logStmt = ibm_db.prepare(conn,
          "insert into systemlog values(?,?,?,?)")
res=ibm_db.execute(logStmt, ...,logtext))
```

IBM Db2 REST APIs (1|2)

- Db2 offers REST APIs for administration and data access
  - https://cloud.ibm.com/apidocs/db2-warehouse-on-cloud
  - https://cloud.ibm.com/apidocs/db2-on-cloud
- Manage access and users, (up)load data
- Database objects, monitoring, ...
- Submit SQL jobs and retrieve results

IBM Db2 REST APIs (2|2)

Sample code:

```
def sendSQLJob(baseURI, token, commands):
    url     = baseURI+"/sql_jobs"
    headers = { "Authorization" : "Bearer "+token, "accept": "application/json" }
    data    = {"commands": commands, "limit":1000, "separator":";", "stop_on_error":"no"}
    response  = requests.post( url, json=data, headers=headers )
    return response.json()

def getSQLJob(baseURI, token, jobid):
    url = baseURI+"/sql_jobs/"+jobid
    headers = { "authorization" : "Bearer "+token, "accept": "application/json" }
    response = requests.get(url, headers=headers)
    return response.json()

...
sqlJob=sendSQLJob(credentials["baseURI"],db2Token["token"],"select * from mytable")
...
jobres=getSQLJob(credentials["baseURI"],db2Token["token"], sqlJob["id"])
```

Db2 programming for the cloud

Cloud Deployment Options

Cloud Deployment Models (IBM Cloud)

- Machine: bare metal or virtual machine or ...
- Container: Docker containers on Kubernetes
- PaaS: Cloud Foundry
- FaaS: Cloud Functions

Similar options with other cloud providers.

Serverless /FaaS / IBM Cloud Functions

- Everything is a function
- no permanently running server => **serverless**
- IBM Cloud Functions / Apache OpenWhisk
- supports Node.js, Python, Docker, ...
- Benefits: Cost model, programming model, existing packages
- Good for infrequently used / executed code

Db2 options on IBM Cloud (1|3)

- Deploy your own Db2 (bare metal / VM / container)
  - DIY, everything
- Db2 Hosted
  - “do it yourself” (DIY) administration
  - Not tailored towards small projects
- Db2 Warehouse on Cloud (“dashDB”)
- Db2 on Cloud (“dashDB for Transactions”)

Db2 options on IBM Cloud (2|3)

- Db2 on Cloud (“dashDB for Transactions”)
  - “regular” Db2
  - Offers free Lite plan, great for (small) testing, requires manual renewal after 30 days
  - Many payable plans and options

Db2 options on IBM Cloud (3|3)

- Db2 Warehouse on Cloud (“dashDB”)
  - Db2 with BLU, columnar table organization by default
  - SMP and MPP options plans
  - ~~Entry plan with free quota~~ (discontinued this September)
    => picked for some of my cloud tutorials
    => add “organize by row” to CREATE TABLE statements
    Tutorials going to switch to plan "Flex One" or "Db2 on Cloud Lite"

Db2 Programming for the Cloud

- Your app and components may be exposed to the “outside”
- App components likely are not co-located
- Security and performance best practices
- Credentials, dynamic configuration, use sharing between services
- Reduce network traffic, limit columns and rows, apply filters, server-side predicates

→ see section with Tips & Tricks

Setting up a local dev environment

Local development

Connect your local tools

- Db2 on Cloud provides access credentials
  - Username, password
  - Hostname, port
  - SSL information (sometimes enforced)
- Use your regular local tools to connect to the cloud-based database

Catalog the cloud environment

- Use CATALOG to make cloud environment available to local Db2 environment

- **catalog tcpip node** cloudenv remote 50.97.xx.xxx server 50000
- **catalog db** clddb1 at node cloudenv
- **connect** to clddb1 user user0815

Federate to the cloud (1|2)

- Connect to cloud database from within local database:
  - **create wrapper** drda
  - **create server** mycloud type db2/cs version 10.5 wrapper drda authorization user0815 password "mypwd" options (dbname 'CLDDB1')
  - **create user mapping** for hloeser server mycloud options (remote_authid 'user0815', remote_password 'mypwd')

Federate to the cloud (2|2)

- Now everything is ready:
  - ```
    select distinct tabschema
    from mycloud.syscat.tables
    ```
  - ```
    insert into thelocaltable
    select * from mycloud.user0815.thecloudtable
    where col1 > 15
    and col2 = '2019-09-01'
    ```

Security

The secret is in the credentials!

Cloud Security – T&T (1|4)

- Your app and its components are accessible from the Internet
- Understand separation of duties and admin tasks, shared responsibility in *aaS contracts
- Parts of a solution ([micro]services) in shared environments
- Understand roles: user vs. developer vs. admin vs. auditor ...
- Use of service IDs
- Public vs. private endpoints

Cloud Security – T&T (2|4)

- State of the art: Automatic binding of credentials
- Avoid manual configuration and copying credentials

- Same principle, different terms
  - Cloud Functions: service bindings, access via runtime env
  - Cloud Foundry: service binding, VCAP environment
  - Kubernetes: secrets, either as env or file system

Cloud Security – T&T (3|4)

- user ID vs. **service ID**: separation of person and function
- **API key and access token** vs. username / password

- prepared vs. dynamic SQL statements
- BYOK (bring your own key) & data encryption
- SSL connection, certificate manager
- security traces and audits

Cloud Security – T&T (4|4)

- Rotate credentials
- Rename old and recreate new credentials
- Update service bindings / secrets
- Rotate service credentials without app downtime
  - Kubernetes offers "kubectl rollout restart"
  - Cloud Foundry has "cf v3-zdt-restart"
  - Cloud Functions allows to rebind a service

Application Performance

Quick, quick...!

App Performance – T&T (1|3)

- Cloud = shared infrastructure
  - Your services / solution parts are not co-located
  - network, CPU priority, ...
- cut down network traffic
  - what type of data is needed where and for what?
  - apply predicates
  - filter out unnecessary columns

App Performance – T&T (2|3)

- cut down CPU cycles
  - avoid data and type conversions
  - follow recommendations
  - is sorting and aggregation needed? Where is it done?
  - client-side vs. server-side processing

App Performance – T&T (3|3)

- old Db2 tricks
  - OPTIMIZE FOR n ROWS
  - FETCH FIRST n ROWS ONLY
  - FOR READ ONLY / FOR UPDATE ONLY
  - multiple statement calls vs. stored procedure
  - indexes

Summary

- Cloud offers new capabilities for Db2-based apps
  - Db2: from DIY to fully managed
  - App: FaaS, Cloud Foundry, Kubernetes, VM, etc.
- Use existing tools to work with cloud-based Db2
- Understand security and performance aspects

→ It is easy to get started

Check out these tutorials with Db2

- Tutorial: Build a database-driven Slackbot
  https://cloud.ibm.com/docs/tutorials?topic=solution-tutorials-slack-chatbot-database-watson
- Tutorial: SQL Database for Cloud data
  https://cloud.ibm.com/docs/tutorials?topic=solution-tutorials-sql-database#sql-database
- Tutorial: Combining serverless and Cloud Foundry for data retrieval and analytics
  https://cloud.ibm.com/docs/tutorials?topic=solution-tutorials-serverless-github-traffic-analytics#serverless-github-traffic-analytics