Number 1 – Brightside & REXX Demo

- This Demo has been developed as a sample of how Brightside can be used with REXX as the language for scripting.

- There is a REXX program '**REXXN1**' that gives us the #1 Song and Artist in UK at a certain date. The program needs a parameter in the format '**aaaammdd**'.

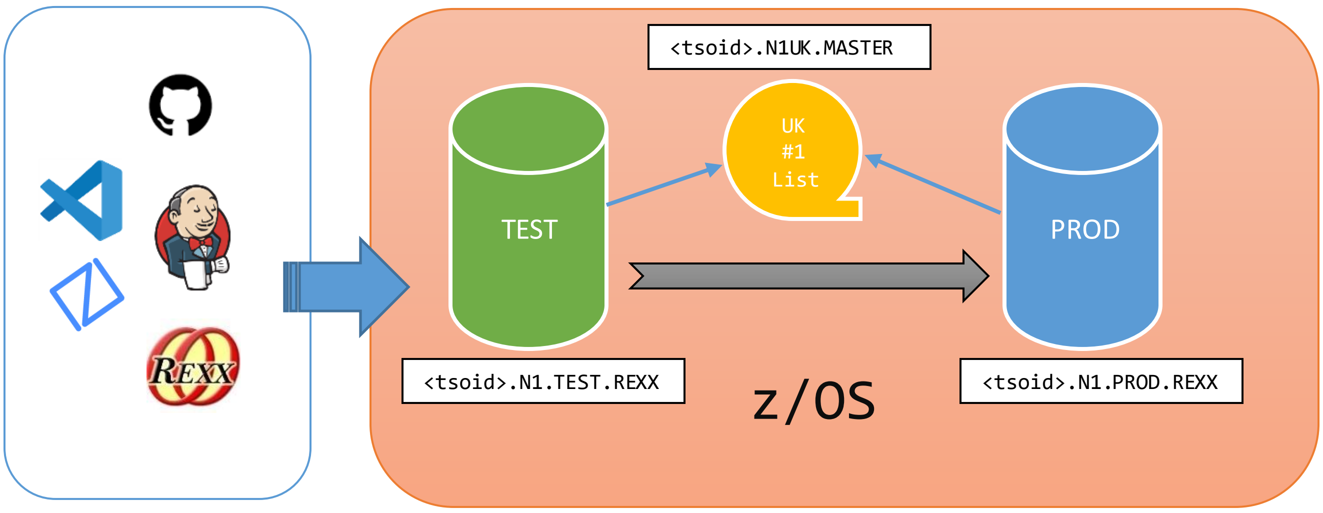
- As a REXX program, one way to execute it could be from any command line:

```
===> TSO EX '<tsoid>.N1.PROD.REXX(REXXN1)' '19650422'
```

```
-------------------------------
On 1965/04/23 #1 in UK was:
-------------------------------
UK_Title :  TICKET TO RIDE
UK_Artist:  BEATLES
```

- The goal is to have a Jenkins Pipeline defined in order to be able to modify the application, run tests and if everything goes ok, install in production. The **REXXN1** program is in the \cntl\\ directory.

- Application structure:



- All code is stored in GitHub: 'https://github.com/drb1972/Number1_Share.git'. The repository is not public, so you need to ask me to share it proportioning me your GitHub user.

- **ooRexx** in needed to execute the scripts. Open Object REXX will be used as the scripting language instead JavaScript. It is free and can be downloaded for several operating systems. Can be downloaded from https://sourceforge.net/projects/oorexx/files/oorexx/4.2.0/

- Depending on your laptop/VM settings and how are Brightside and the rest of the tools installed, you may or may not going to execute the scripts in your user profile. It could be necessary to have already installed the necessary resources:

- In the system variables it is necessary to add: C:\Program Files\ooRexx to the PATH (or the folder where you have installed it) .Brightside needs be in the system path too (not just the user profile path).
- Bright zosmf-profile and tso-profile need to be defined at system level depending agaim in tour environment, not the user one.
- Check that your zosmf profile password is the current one or it will be revoked for several logon tries.

- **Important:** Edit “start.rexx” and write your TSOID in “**TSOID=’xxxxxxxx’**”. The purpose of this is because the sample will be created under you tso userid HLQ.

- **‘Start.rexx’** is the task runner, and it has some steps including several commands and some logic, but only one at a time will be executed (same as Gulp). These steps are:

  1. ALLOC: if the library structure doesn’t exists, it is automatically created.
     a. `<tsoid>.N1.TEST.REXX`
     b. `<tsoid>.N1UK.MASTER`
  2. UPLOAD:
     a. **\cntl\Rexxn1.rex** is uploaded to PDS: `<tsoid>.N1.TEST.REXX(REXXN1)`
     b. **listUKMaster.txt** is uploaded to PS: `<tsoid>.N1UK.MASTER`.
  3. TESTT_UK: A couple of test are done to keep within the Pipeline. If any test does not end as expected, Jenkins will stop the Building.
  4. INSTALL:
     a. If `<tsoid>.N1.PROD.REXX` doesn’t exists. It is created.
     b. If `<tsoid>.N1.PROD.REXX.BACKUP` doesn’t exists. It is created.
     c. Copy REXXN from `<tsoid>.N1.`**`PROD`**`.REXX` to `<tsoid>.N1.PROD.REXX.`**`BACKUP`** (not in the first Install).
     d. Copy REXXN from `<tsoid>.N1.`**`TEST`**`.REXX` to `<tsoid>.N1.`**`PROD`**`.REXX`
  5. TESTP_UK: A couple of test are done. If tests does not end as expected, an automatic BackOut is executed to restore the last REXXN1 version (not the first time).

- **GitHub:**

To share the code with you:

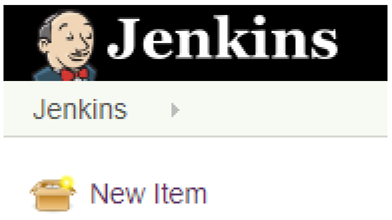
1. I need your GitHub user in order to add you as a Collaborator. You will receive an invitation to join Number1_Share repository.
2. Once you can see <my-user>/Number1_Share in your GitHub you have to “FORK” it by clicking here:

3. Once is forked, you will have your own repository to test the ‘Number1’ application and Pipeline as: <your-GitHub-User>/Number1_Share
4. Clone it in your projects folder from the CLI:

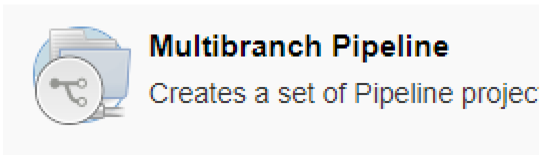
   git clone ‘https:// <your-GitHub-User>/Number1_Share’

- **Jenkins:**

1. New item:

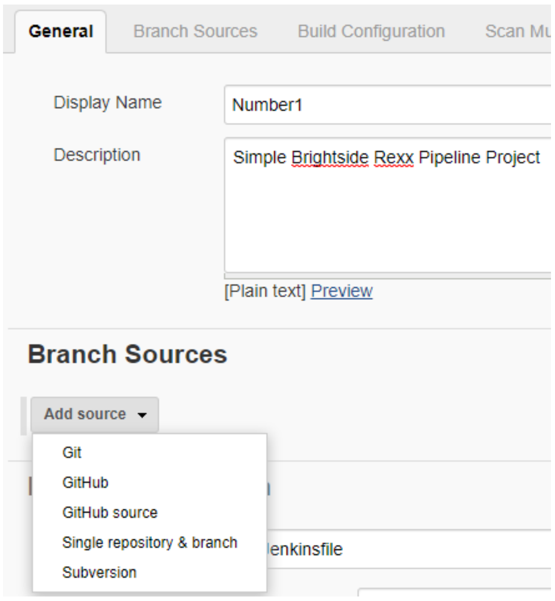


2. Enter a Name (I named Number1), select Multibranch Pipeline and click on the ‘ok’ button.



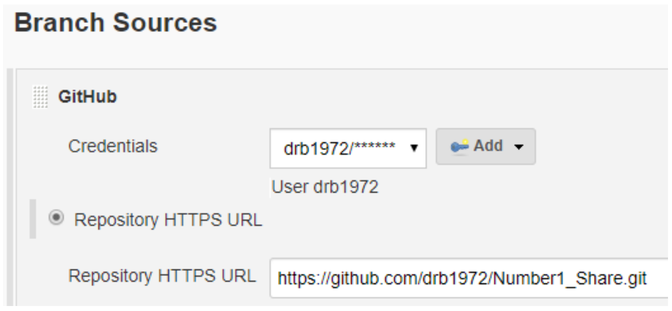
3. Select **GitHub** in the ‘Add source’ drop down menu and paste your user GitHub url:

   3.1. https://github.com/<your-github-user>/Number1_Share.git. Description is optional. Click on the ‘Save’ button.



Display Name: Number1

Description: Simple Brightside Rexx Pipeline Project



Credentials: drb1972/******

User drb1972

● Repository HTTPS URL

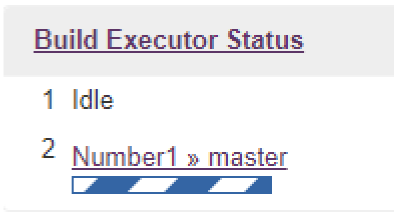
Repository HTTPS URL: https://github.com/drb1972/Number1_Share.git

3.2. Validate your GitHub credentials to have access. (At this moment the Number1_Share GitHub repository is private, and as far as I know if I make it public, anyone could push changes. I haven't figured out yet how to share it 'Read only').

3.3. Save.

4. Jenkins will check now the branches and run the Build Executor.

4.1. You can click on Number1 >> master and you will see the first run of the project

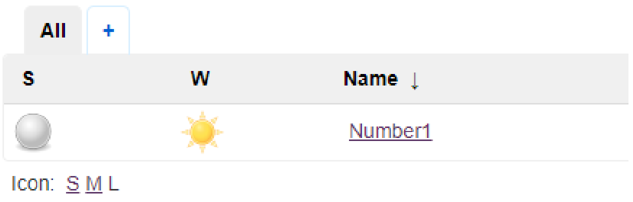


Build Executor Status

1 Idle

2 Number1 » master

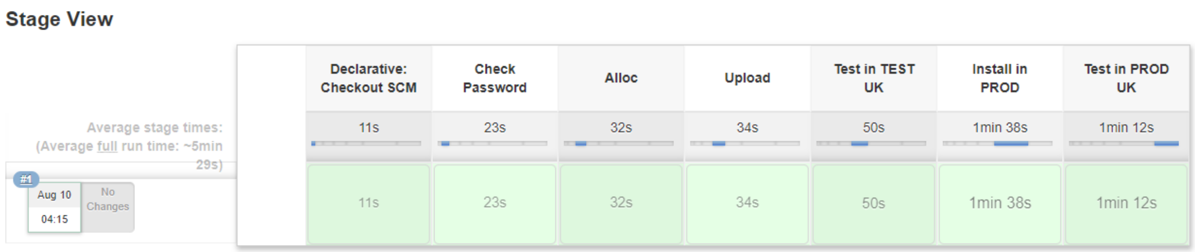
5. In the main page, select Number1 project:



S	W	Name ↓
●	☀	Number1

6. Select 'Build now':

- Hopefully you will succeed in the Pipeline process:



Stage View

Average stage times:
(Average full run time: ~5min 29s)

#1 Aug 10 04:15 No Changes

	Declarative: Checkout SCM	Check Password	Alloc	Upload	Test in TEST UK	Install in PROD	Test in PROD UK
Average	11s	23s	32s	34s	50s	1min 38s	1min 12s
#1	11s	23s	32s	34s	50s	1min 38s	1min 12s

6.1. These are the standard outputs you can see in the Jenkins Pipeline steps log:

6.1.1. Step - Check Password: This output should be always the same. I begin with this step because if I am working with a zosmf profile that has an old tso password and I issue 3 Brightside commands against the mainframe, my password will be revoked. It has happened to me already ☺

```
+ rexx start pass
Correct TSO access.
```

6.1.2. Step – Alloc

First execution:

```
+ rexx start alloc
========================================
>> ALLOC - Create libraries
========================================
Creating  RODDI01.N1.TEST.REXX
Data set created successfully.
Creating  RODDI01.N1UK.MASTER
Data set created successfully.
```

Subsequent executions:

```
+ rexx start alloc
========================================
>> ALLOC - Create libraries
========================================
RODDI01.N1.TEST.REXX Already exists
RODDI01.N1UK.MASTER Already exist
```

6.1.3. Step – Upload: This output should be always the same.

```
+ rexx start upload
========================================
>> UPLOAD - Upload libraries
========================================
Uploading RODDI01.N1.TEST.REXX
Uploading RODDI01.N1UK.MASTER
```

6.1.4. Step – Test in TEST UK: This output should be always the same.

```
+ rexx start testt_uk
========================================
>> TESTT_UK - TESTING in TEST
========================================
========================================
Beginning Test1...
========================================
Expected result for 19600826:
UK_Title : APACHE
UK_Artist: SHADOWS
========================================
Result ==> UK_Title :    APACHE
Result ==> UK_Artist:    SHADOWS
TEST1  OK
========================================
Beginning Test2...
========================================
Expected result for 18600826:
NULL value
========================================
NULL value
TEST2  OK
Tests in TEST OK
```

6.1.5. Step – Install in PROD:

First execution:

```
+ rexx start install
========================================
>> INSTALL - Install in PROD
========================================
Creating  RODDI01.N1.PROD.REXX
Data set created successfully.
Creating  RODDI01.N1.PROD.REXX.BACKUP
Data set created successfully.
Copying REXXN1 from RODDI01.N1.TEST.REXX to RODDI01.N1.PROD.REXX
Successful Install
```

Subsequent executions:

```
+ rexx start install
========================================
>> INSTALL - Install in PROD
========================================
RODDI01.N1.PROD.REXX Already exists
RODDI01.N1.PROD.REXX.BACKUP Already exists
Copying REXXN1 from RODDI01.N1.PROD.REXX to
RODDI01.N1.PROD.REXX.BACKUP
Successful BackUp
Copying REXXN1 from RODDI01.N1.TEST.REXX to
RODDI01.N1.PROD.REXX
Successful Install
```

6.1.6. Step – Test in PROD UK: This output should be always the same.

```
+ rexx start testp_uk
========================================
>> TESTP_UK - TESTING in PROD
========================================
========================================
Beginning Test1...
========================================
Expected result for 19600826:
UK_Title : APACHE
UK_Artist: SHADOWS
========================================
Result ==> UK_Title :    APACHE
Result ==> UK_Artist:    SHADOWS
TEST1  OK
========================================
Beginning Test2...
========================================
Expected result for 18600826:
NULL value
========================================
NULL value
TEST2  OK
========================================
Tests in PROD OK
```

- After the first installation in PROD, you can execute REXXN1. Just type ‘**rexx testn1’** in the CLI and you will be prompted for a date. <u>But before executing</u> it, edit the script ‘**testn1.rex’** and set your tsoid in user = 'xxxxxxxx'.

- The same way as Gulp, you can execute in the CLI one of the stages of the Pipeline by executing '**rexx start <step>**'. You can see the name of the steps in the Jenkinsfile right behind each description.

- It is also possible to define Gulp tasks in combination with the REXX scripts. This gives us the flexibility of choosing the most suitable for each purpose.

- This Demo has been created also to share ideas that could be extrapolated to any project. For example: It is included a simple REXX script called '**commit_n1.rex'** that easy and group some commands. When executed:

    1. Look for files that have been changed since the last commit and look for 'source-code' files (in this case, the ones with the extension = .rex). A automatically edits these files, no matter in what directory they are, and overwrites the timestamp just before it is uploaded to GitHub, therefore we can always see when was the source code sent to compile, or whatever: The code is updated with the following format at the top of the program:

```
/* timestamp - 2019-08-10T05:15:34.232000 */
```

    2. Sets the GitHub configuration to the GitHub user I am using with this repository. I do this because I have more than 1 GitHub account linked with different projects. Change the user to yours by editing the script.
    3. Commit the files that have been changed
    4. PUSH into your default GitHub repository

    To execute it: '**rexx commit_n1'** in the project folder.

    (I have some utilities like this in a folder added to the path of my windows system, so I can execute any in all projects just as command lines. In mainframe it would be scripts in my alloc F SYSPROC library).

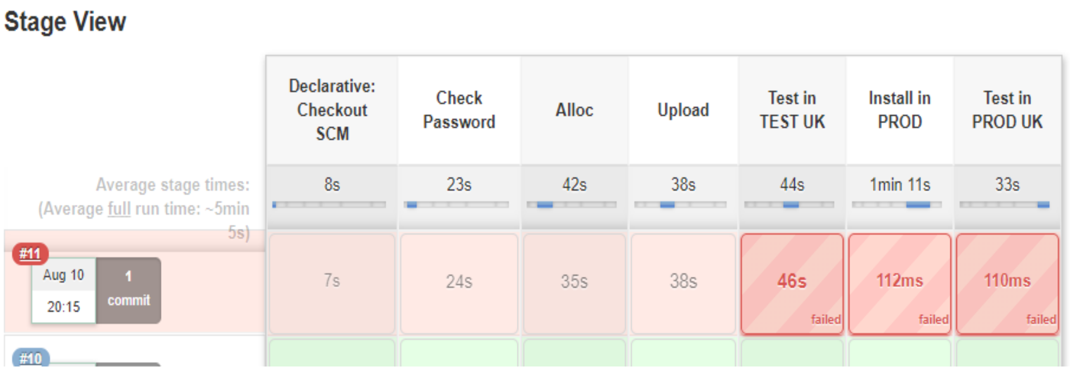
- Practice 1: Let's make the tests in TEST fail. Jenkins should stop the execution of the Pipeline and not allow the program go into production PROD. To do that we will do the following:

    1. Edit the 'listUKMaster.txt' file in the Workspace and change line #15. This line is the one used in our test script, expecting 'APACHE'. We will modify it to 'APACH':

```
25/08/1960;APACHE;SHADOWS;5 -> 25/08/1960;APACH ;SHADOWS;5
```

    2. Execute in the CLI: 'rexx commit_n1'

    3. In Jenkins start a Build. It should fail in the step 'Test in TEST UK' and stop the installation:



Stage View

	Declarative: Checkout SCM	Check Password	Alloc	Upload	Test in TEST UK	Install in PROD	Test in PROD UK
Average stage times: (Average full run time: ~5min 5s)	8s	23s	42s	38s	44s	1min 11s	33s
#11 Aug 10 20:15 1 commit	7s	24s	35s	38s	46s failed	112ms failed	110ms failed
#10							

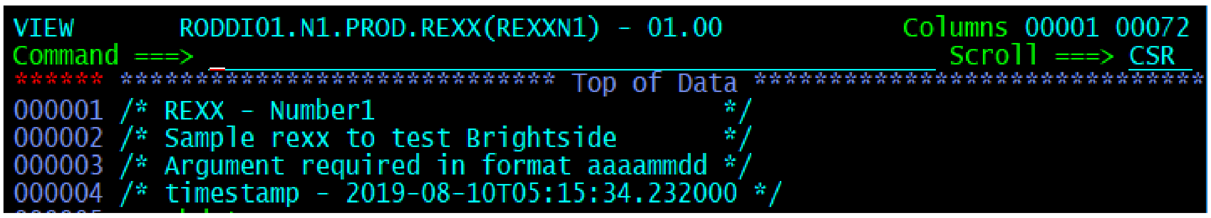
Taking a look at the 'Test in TEST UK' log:

```
+ rexx start testt_uk
========================================
>> TESTT_UK - TESTING in TEST
========================================
========================================
Beginning Test1...
========================================
Expected result for 19600826:
UK_Title : APACHE
UK_Artist: SHADOWS
========================================
Result ==> UK_Title :     APACH
Result ==> UK_Artist:     SHADOWS
TEST1  KO
========================================
Beginning Test2...
========================================
Expected result for 18600826:
NULL value
========================================
NULL value
TEST2  OK
Tests in TEST not OK
```

4. Now we will edit again the listUKMaster.txt and will restore the 'E' in APACHE in line 15.

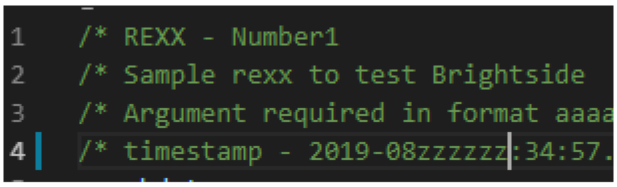
- Practice 2: Now we are going to make the program fail after it has passed the tests in TEST and have been installed in production PROD. The script executes a couple of tests right after the program has been installed and if any fails, there is a BackOut procedure that will restore the latest version of the program that was running in production. To provoke this situation we are going to modify the testing script in production:

    1. Take a look at the timestamp in the current production program REXXN1 in the mainframe: `<tsoid>.N1.PROD.REXX(REXXN1).` Mine shows 05:15:34.



```
VIEW       RODDI01.N1.PROD.REXX(REXXN1) - 01.00           Columns 00001 00072
Command ===> _                                               Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 /* REXX - Number1                     */
000002 /* Sample rexx to test Brightside     */
000003 /* Argument required in format aaaammdd */
000004 /* timestamp - 2019-08-10T05:15:34.232000 */
```

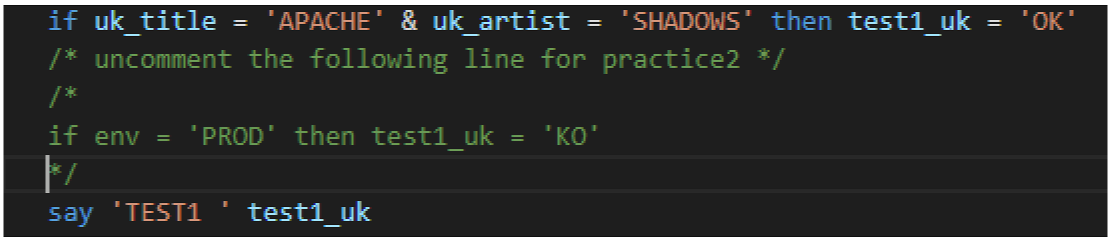
2. Let's change something in the main program (**rexxn1.rex** in the **\cntl** directory), adding a comment or overwriting the timestamp, so when you will use the 'commit_n1' the program will be updated with the time of the PUSH to GitHub.



```
/* REXX - Number1
/* Sample rexx to test Brightside
/* Argument required in format aaaa
/* timestamp - 2019-08zzzzzz:34:57.
```

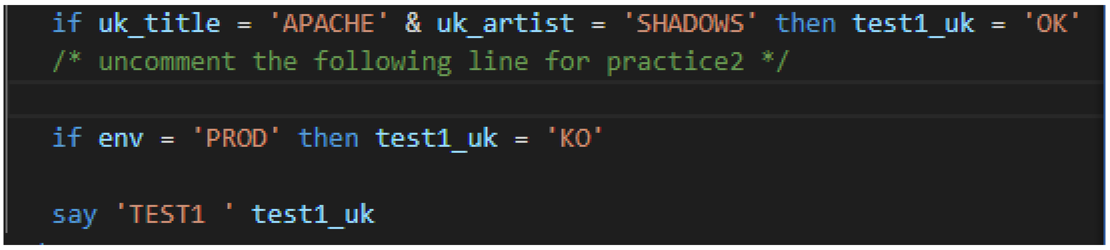
3. Edit the **start.rex** script, find the following paragraph and modify it uncommenting the code:

Before:



```
if uk_title = 'APACHE' & uk_artist = 'SHADOWS' then test1_uk = 'OK'
/* uncomment the following line for practice2 */
/*
if env = 'PROD' then test1_uk = 'KO'
*/
say 'TEST1 ' test1_uk
```

After:

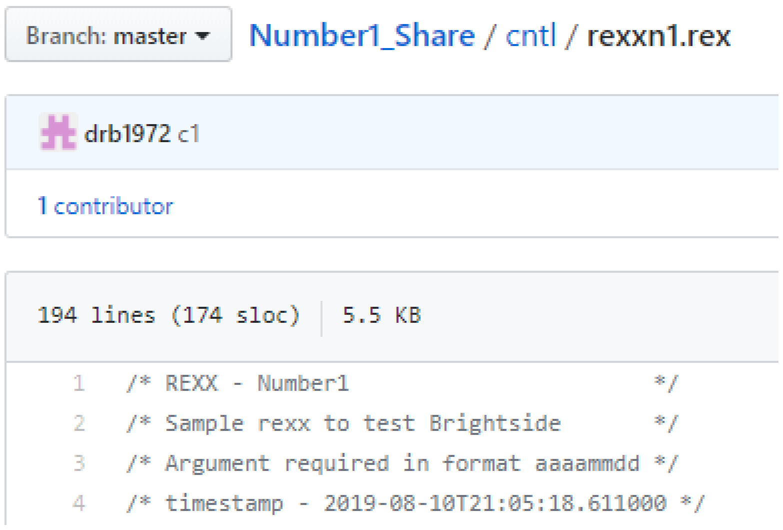


```
if uk_title = 'APACHE' & uk_artist = 'SHADOWS' then test1_uk = 'OK'
/* uncomment the following line for practice2 */

if env = 'PROD' then test1_uk = 'KO'

say 'TEST1 ' test1_uk
```

4. Execute '**rexx commit_n1**'. This will update in GitHub the script and the program **rexxn1.rex** with the new timestamp. After issuing the command, you can access to the source code in GitHub to check the new timestamp (21:05:18):



Branch: master ▾ Number1_Share / cntl / rexxn1.rex

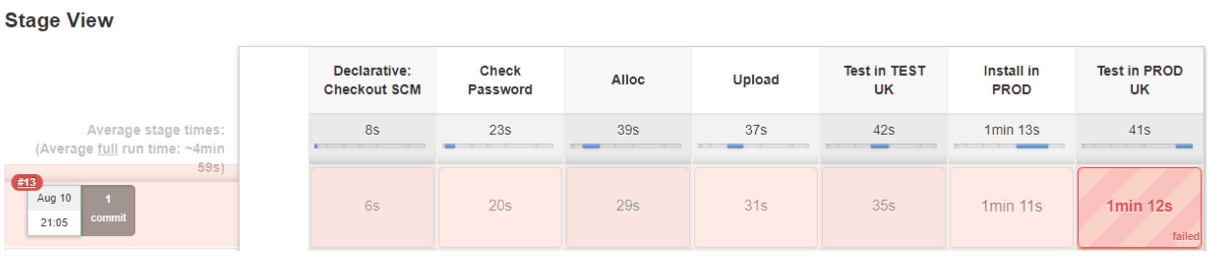
drb1972 c1

1 contributor

194 lines (174 sloc) | 5.5 KB

```
/* REXX - Number1                   */
/* Sample rexx to test Brightside   */
/* Argument required in format aaaammdd */
/* timestamp - 2019-08-10T21:05:18.611000 */
```

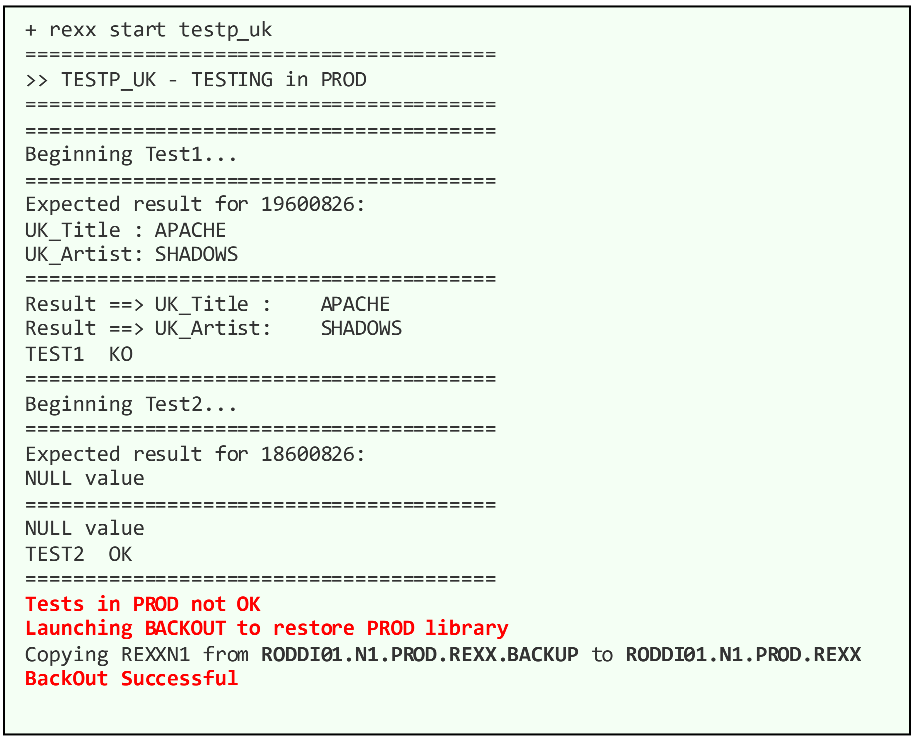
5. In Jenkins, Build now. The Pipeline should fail in the last step 'Test in PROD UK' and restore the last version in Production just replaced in the 'Install in PROD' step.



Stage View

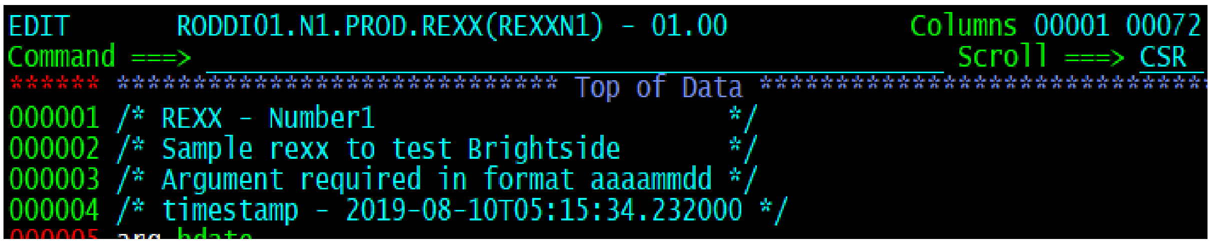
	Declarative: Checkout SCM	Check Password	Alloc	Upload	Test in TEST UK	Install in PROD	Test in PROD UK
Average stage times: (Average full run time: ~4min 59s)	8s	23s	39s	37s	42s	1min 13s	41s
#13 Aug 10 21:05 1 commit	6s	20s	29s	31s	35s	1min 11s	1min 12s failed

The log for the 'Test in PROD UK' Shows:



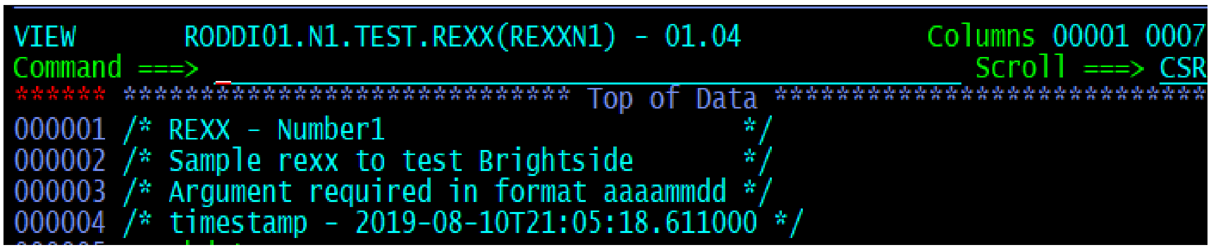
```
+ rexx start testp_uk
=======================================
>> TESTP_UK - TESTING in PROD
=======================================
=======================================
Beginning Test1...
=======================================
Expected result for 19600826:
UK_Title : APACHE
UK_Artist: SHADOWS
=======================================
Result ==> UK_Title :     APACHE
Result ==> UK_Artist:     SHADOWS
TEST1  KO
=======================================
Beginning Test2...
=======================================
Expected result for 18600826:
NULL value
=======================================
NULL value
TEST2  OK
=======================================
Tests in PROD not OK
Launching BACKOUT to restore PROD library
Copying REXXN1 from RODDI01.N1.PROD.REXX.BACKUP to RODDI01.N1.PROD.REXX
BackOut Successful
```

PROD library after the BackOut: The last functional version.



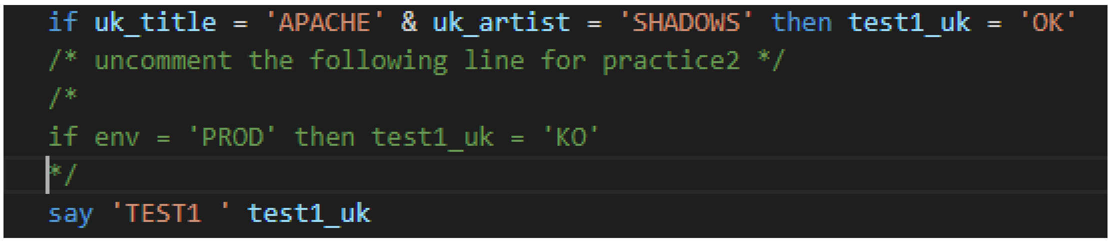
```
EDIT       RODDI01.N1.PROD.REXX(REXXN1) - 01.00            Columns 00001 00072
Command ===>                                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 /* REXX - Number1                        */
000002 /* Sample rexx to test Brightside        */
000003 /* Argument required in format aaaammdd */
000004 /* timestamp - 2019-08-10T05:15:34.232000 */
000005 arg bdate
```

Test still has the new version that needs to be modified after figuring out what the problem was in production. In the real world, more and different tests should be added to the Pipeline in order to avoid wrong installations in PROD. It should have failed in the TEST tests.



```
VIEW       RODDI01.N1.TEST.REXX(REXXN1) - 01.04            Columns 00001 0007
Command ===> _                                                 Scroll ===> CSR
****** ***************************** Top of Data ****************************
000001 /* REXX - Number1                         */
000002 /* Sample rexx to test Brightside         */
000003 /* Argument required in format aaaammdd */
000004 /* timestamp - 2019-08-10T21:05:18.611000 */
```

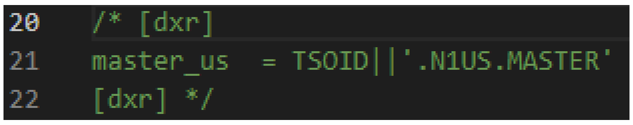
6. Edit the '**start.rex**' script, and comment again the lines we have used to cheat in this practice and let the script behave correctly:



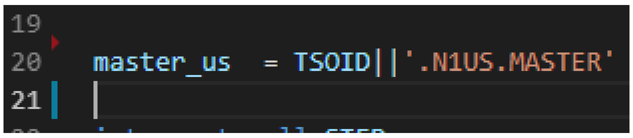
```
if uk_title = 'APACHE' & uk_artist = 'SHADOWS' then test1_uk = 'OK'
/* uncomment the following line for practice2 */
/*
if env = 'PROD' then test1_uk = 'KO'
*/
say 'TEST1 ' test1_uk
```

- Practice 3: Modify the program 'REXXN1'. There is a request to enhance our program. There is another #1 list songs & artists from the US. Now we want our program to give us both #1 in UK and in the US. We will use the provided file 'listUSMaster.txt'. To do so:

1. Edit the task runner '**start.rex**' and find the paragraphs commented with this symbols '[dxr]' and delete these lines (there are 6 occurrences):



```
/* [dxr]
master_us   = TSOID||'.N1US.MASTER'
[dxr] */
```

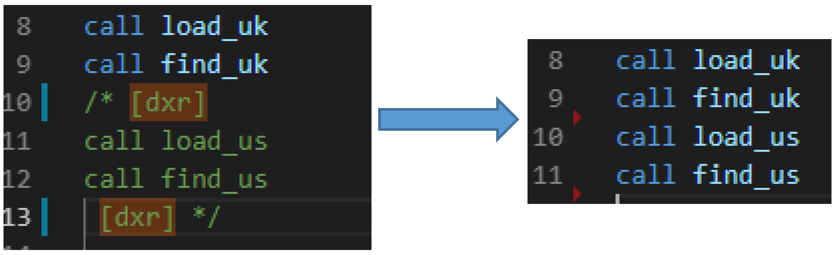


```

master_us   = TSOID||'.N1US.MASTER'

```

2. Edit '**rexxn1.rex**' in the \cntl directory and delete the lines with '[dxr]':



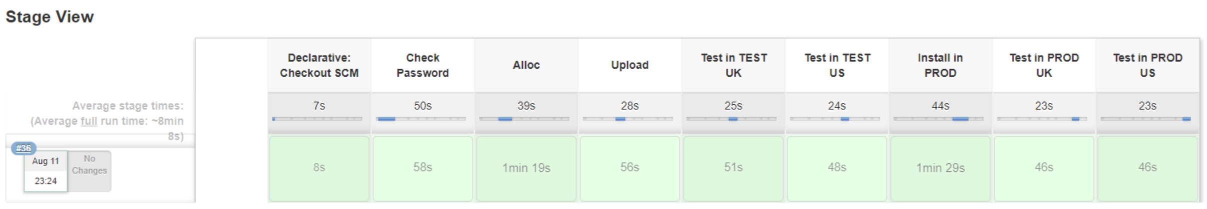
```
call load_uk
call find_uk
/* [dxr]
call load_us
call find_us
[dxr] */
```

```
call load_uk
call find_uk
call load_us
call find_us
```

3. Edit the Jenkinsfile. Uncomment the lines that are commented.

4. Issue the 'rexx commit_n1' command.

5. In Jenkins, build a Pipeline. New steps have been added:



Stage View

	Declarative: Checkout SCM	Check Password	Alloc	Upload	Test in TEST UK	Test in TEST US	Install in PROD	Test in PROD UK	Test in PROD US
Average stage times: (Average full run time: ~8min 8s)	7s	50s	39s	28s	25s	24s	44s	23s	23s
#36 Aug 11 23:24 No Changes	8s	58s	1min 19s	56s	51s	48s	1min 29s	46s	46s

We can see now the following messages in the steps logs after the modifications:

Alloc: The first time after the modification the new Masterfile for US is created:

```
+ rexx start alloc
=======================================
>> ALLOC - Create libraries
=======================================
RODDI01.N1.TEST.REXX Already exists
RODDI01.N1UK.MASTER Already exists
Creating  RODDI01.N1US.MASTER
Data set created successfully.
```

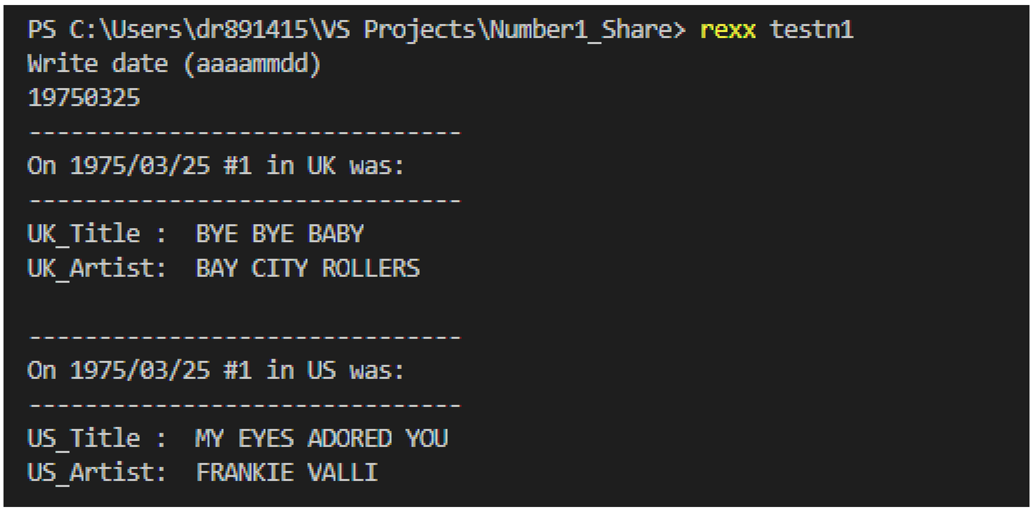
Upload: Upload the lists and the modified program:

```
+ rexx start upload
=======================================
>> UPLOAD - Upload libraries
=======================================
Uploading RODDI01.N1.TEST.REXX
Uploading RODDI01.N1UK.MASTER
Uploading RODDI01.N1US.MASTER
```

Test in TEST US: This are a couple of new tests added to check the US file processing. Same for PROD

```
+ rexx start testt_us
=======================================
>> TESTT_US - TESTING in TEST
=======================================
=======================================
Beginning Test1...
=======================================
Expected result for 19600826:
US_Title : IT'S NOW OR NEVER
US_Artist: ELVIS PRESLEY
=======================================
Result ==> US_Title :    IT'S NOW OR NEVER
Result ==> US_Artist:    ELVIS PRESLEY
TEST1  OK
=======================================
Beggining Test2...
=======================================
Expected result for 18600826:
NULL value
=======================================
NULL value
TEST2  OK
Tests in TEST OK
```

6. Now you can test from the CLI the date you want: ‘rexx testn1’



```
PS C:\Users\dr891415\VS Projects\Number1_Share> rexx testn1
Write date (aaaammdd)
19750325
-------------------------------
On 1975/03/25 #1 in UK was:
-------------------------------
UK_Title :  BYE BYE BABY
UK_Artist:  BAY CITY ROLLERS

-------------------------------
On 1975/03/25 #1 in US was:
-------------------------------
US_Title :  MY EYES ADORED YOU
US_Artist:  FRANKIE VALLI
```

7. You can also test directly with the Bright command:
`bright tso issue command "ex 'roddi01.n1.prod.rexx(rexxn1)' '19750325'"`

Thanks for your time. Please, feel free to send me your feedback, ideas, enhancements, ...

Diego