TECHNISCHE UNIVERSITÄT DRESDEN

Report

Title	Automatic Code Extraction & check for Scone Docs
Student Name	Prateek Bhatnagar
Duration	01.06.2019-31.10.2019
Course	M.Sc (DSE)

1. Abstract 3
2. Introduction 3
3. Problem 3
3.1 Definition 3
3.2 High level Solution 3
3.3 Automation Strategy in General 4
3.3.1 Phase1 4
3.3.2 Phase2 4
4. Conclusion 5
5. Future work 5
6. Mentors 5
7. Appendix A 6
7.1 Python code 6
7.2 Output for MainFolder 6
7.3 Output for testSummary 6
7.4 Content for files 6
8. Appendix B 7

1. Abstract

This Report is about 'Automatic Code Extraction and Check for SCONE docs'. Automatic code extraction means extracting code and its respective output from markdown files available for Scone Docs.

2. Introduction

Scone docs include code examples from various languages like C, C++, Java, Python, etc. These examples are organized in the form of a markdown (.md) file which is kept at respective locations at GitHub. For reference following link is used and worked upon https://github.com/doflink/sconedocs.Whenever some changes are done in scone library or code one needs to execute all these tests manually to ensure validation and verification for Scone project.

Automation of these tests is essential to increase efficiency in the long run and hence a prototype is designed in 'Python' to ease this.

3. Problem

3.1 Definition

The main problem with such automation is that each test is randomized which means there is no fixed pattern that is followed in each code example and every language has different syntax and semantics. To address the above problem microservices-based Docker containers are used wherein each code example is run inside a specific container.
These code examples are organized in the form of markdown files. To summarize, this report is trying to address the following challenges:-

1. A fixed pattern should be followed for each code example.
2. Code should be extracted from the markdown file and its respective output
3. A docker container is to be maintained for each code example for its execution.

3.2 High level Solution

Markdown file is a simple text file with some dialects of the markdown language. It contains symbols which help to format the text. To differentiate between code and output following dialects were used here:-

	Code	Output
Start Symbol	```bash	```
End Symbol	```	```

Anything in between start and end symbols is either code or output. After extracting the code, the actual execution is done and the output of this actual execution is compared with the expected output extracted from the markdown file. If the files don't contain anything between the respective symbols then those files are not executed for tests as they don't have bash code and are simply ignored.

3.3 Automation Strategy in General

Automation Strategy is divided into two Phases

1. Phase 1-Preparation of the actual output
2. Phase 2-Comparison of Actual Output(Docker Execution logs) Vs Expected Output (Extracted from .md file)

3.3.1 Phase1

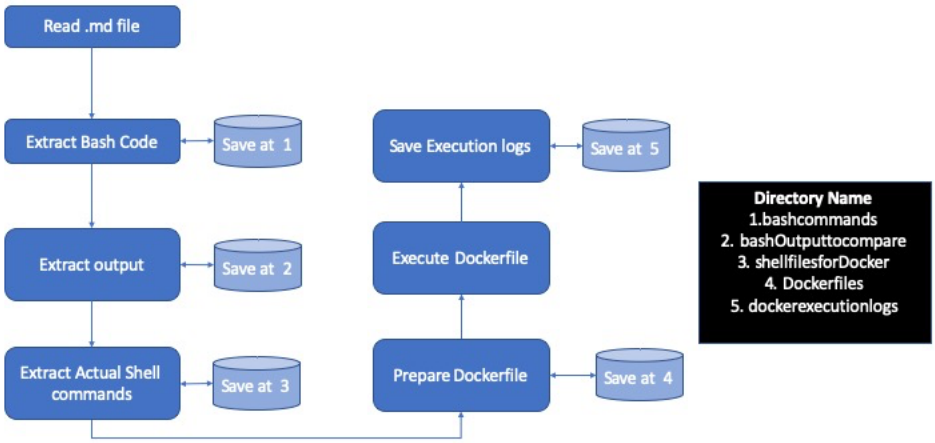

Figure 1

1. Markdown file (.md) file is read line by line through file operations in python.
2. Bash code is extracted and saved in a file at Directory 'bashcommands' i.e. for example, C.md file is read and C.sh file is created and saved at 'bashcommands'.
3. The next step output of the code is extracted and saved at Directory 'bashOutputtocompare'.
4. The file in step2 contains bash commands which include docker commands also like "docker pull xxxx". But docker pull cannot be executed as it as hence dockerfile is created out of docker commands and actual bash commands are saved in another ready to execute bash script at directory 'shellfilesforDocker'.
5. Dockerfile is generated at directory 'Dockerfiles' and respective bash script is configured to run on starting the container.
6. Dockerfile is actually executed.
7. Execution logs generated out of execution at step6 are saved at 'dockerexecutionlogs'.

In phase1 the actual output is prepared for comparison.

3.3.2 Phase2

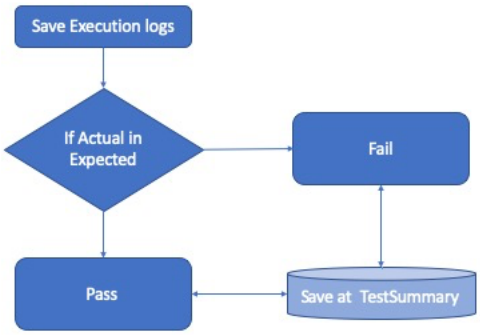

Figure 2

1. The actual output is looked line by line in the expected output (extracted in phase1 saved at directory 'bashOutputtocompare').
2. If the line is not found, the test is marked as fail and logs are written in xxx. fail file at directory 'TestSummary' and 'No of Test Failed' count is increased by 1 in xxx.summary file.
3. If a line is found, the test is marked as pass and logs are written in xxx.pass file at directory 'TestSummary' and 'No of Test Passed' count is increased by 1 in xxx.summary file.

In phase2, the test result is generated with a separate pass and fail files. If no test failed then xxx.fail file is not generated.

4. Conclusion

Earlier all the execution of all the tests was manual, but now with the automation prototype in place, these tests can be formatted and executed automatically.

5. Future work

There is future work that can be done in this prototype as follows:-

1. Formatting all the tests (sconedocs) in one agreed format.
2. Implementation of the prototype in Rust.
3. Integration with DevOps pipeline (E.g Jenkins) for automatic execution of all the test cases on new build trigger.

6. Mentors

This work has been completed under the guidance of:-

1. Dr. Le Quoc Do (do.le_quoc@tu-dresden.de)
2. Prof Christof Fetzer(christof.fetzer@tu-dresden.de)

7. Appendix A

7.1 Python code

https://github.com/prbh695a/Scone

7.2 Output for MainFolder

```
prateekb@edr06753 Scone % ls -l
-rw-r--r--   1 prateekb  staff   394 Oct 14 20:44 README.md
drwxr-xr-x  51 prateekb  staff  1632 Sep 24 13:43 bashCommands
drwxr-xr-x  51 prateekb  staff  1632 Oct  8 14:47 bashOutputtocompare
drwxr-xr-x   8 prateekb  staff   256 Oct  8 15:08 dockerexecutionlogs
drwxr-xr-x  17 prateekb  staff   544 Oct  8 13:48 dockerfiles
-rw-r--r--@  1 prateekb  staff  9889 Oct 14 19:22 scone.py
drwxr-xr-x  13 prateekb  staff   416 Sep 23 12:59 sconedocs
drwxr-xr-x  51 prateekb  staff  1632 Oct  8 16:06 shellfilesforDocker
drwxr-xr-x  18 prateekb  staff   576 Oct  9 10:58 testSummary
```

7.3 Output for testSummary

```
prateekb@edr06753 testSummary % ls -lrt
total 120
-rw-r--r--  1 prateekb  staff   144 Sep 23 14:55 C.pass
-rw-r--r--  1 prateekb  staff   198 Sep 23 14:55 C.summary
-rw-r--r--  1 prateekb  staff     0 Sep 24 13:38 R.pass
-rw-r--r--  1 prateekb  staff  5375 Sep 24 13:38 R.fail
```

7.4 Content for files

```
prateekb@edr06753 testSummary % cat Cplusplus.fail
Expected output not found in the file as === export SCONE_KERNEL=0
Expected output not found in the file as === export SCONE_STACK=81920
Expected output not found in the file as === export SCONE_SGXBOUNDS=no
Expected output not found in the file as === export SCONE_VARYS=no
Expected output not found in the file as === Configure options: --enable-shared --enable-debug
--prefix=/mnt/ssd/franz/subtree-scone2/built/cross-compiler/x86_64-linux-musl

prateekb@edr06753 testSummary % cat C.pass
Expected output found as ===fib(23)= 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987,
1597, 2584, 4181, 6765, 10946, 17711, 28657

prateekb@edr06753 testSummary % cat C.summary
======================Test Summary======================
NumberOfTestcases : 1
NumberOfTestcasesPassed : 1
NumberOfTestcasesFailed : 0
======================Test Summary======================
prateekb@edr06753 testSummary %
```

8. Appendix B

Few files are changed in the scone docs folder to show how the prototype works exactly, but every file should be formatted exactly in the same manner to make this prototype work. Here are the actual summary and remarks about the same.

S.NO	Sconedocs	Status	Remarks
1	c	Running	
2	c++	Running	
3	Go	Need to Reformat	Curl command is not found
4	java	Need to Reformat	Error while parsing
5	fortan	Running	
6	MrEnclave	Need to Reformat	Scone image not found
7	Nodejs	Need to Reformat	running loop forever at 3000
8	Python	Need to Reformat	Some Error with '('
9	R	Need to Reformat	Error running images
10	Rust	Need to Reformat	Error running images
11	SCONE_CAS	Need to Reformat	No Docker pull command present
12	SCONE_CLI	Running	<Tab> is giving some issue
13	SCONE_Compose	Running	HA Proxy etc
14	SCONE_Curated_Images	Need to Reformat	Image test
15	SCONE_Dockerfile		docker file example
16	SCONE_EE2EE	Need to Reformat	Swarm
17	SCONE_ENV	NA	Memory Related Changes
18	SCONE_FileShield	Need to Reformat	Docker pull missing
19	SCONE_GENERATE_IMAGE	Need to Reformat	Docker pull missing
20	SCONE_HOST	Need to Reformat	Docker pull missing
21	SCONE_HOST_SETUP	Need to Reformat	Docker pull missing
22	SCONE_OpenStack	Need to Reformat	Empty
23	SCONE_Publications	NA	
24	SCONE_SERVICE	Need to Reformat	Docker pull missing
25	SCONE_STACK	Need to Reformat	Docker pull missing
26	SCONE_SWARM	Need to Reformat	Docker pull missing
27	SCONE_Swarm_Example	Need to Reformat	Docker pull missing
28	SCONE_TUTORIAL	Need to Reformat	Docker pull missing
29	SCONE_VOLUME	Need to Reformat	Docker pull missing
30	SCONE_toolchain	Need to Reformat	Docker pull missing
31	aboutScone	NA	
32	advantages	NA	
33	appsecurity	NA	

34	background	NA	
35	blender	NA	
36	checkconsistency	NA	
37	configuration	NA	
38	dapps	NA	
39	dockerfileexample	NA	
40	dockerinstall	NA	
41	faq	NA	
42	firstcontainer	Need to Reformat	
43	glossary	NA	
44	groupcacheUseCase	Need to Reformat	
45	hardwaremode	Need to Reformat	
46	hostexample	Need to Reformat	
47	iexec_install	NA	
48	index	NA	
49	installation	NA	
50	memory_dump	Need to Reformat	
51	microcode	NA	
52	multistagebuild	Need to Reformat	
53	news	NA	
54	outline	NA	
55	performance	NA	
56	pypyscone	NA	
57	pyspark	NA	
58	requirements.txt	NA	
59	scone-use-case-overview	NA	
60	scone_file_shield	NA	
61	scone_session	NA	
62	sconeinstall	NA	
63	screencast.txt	Need to Reformat	
64	secure_remote_execution	NA	
65	sgxinstall	NA	
66	ssh	NA	
67	technical_summary	NA	
68	tensorflow	Need to Reformat	
69	tensorflowlite	Need to Reformat	
70	usecases	NA	
71	vault	NA	