Shell Scripts: How Businesses Can Utilize Shell Scripts to Automatize Database Tasks

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Darnell Khay

Spring, 2022

Rosanne Vrugtman, Department of Computer Science

Briana Morrison, Department of Computer Science

Shell Scripts: How Businesses Can Utilize Shell Scripts to Automatize Database Tasks

CS4991 Capstone Report, 2022
Darnell Khay
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
Dk3ctu@virginia.edu

Abstract
A multinational financial service company with data engineering teams based in Charlotte, NC, encountered a challenge to efficiently migrate their applications and respective projects by the end of 2022. The DevOps team at this company developed a shell program that would supposedly improve the migration; however, it only created further issues that hindered the process instead. The proposed solution was to implement another shell program that supports the initial program by preparing the setup required, establishing a user interface, validating input and managing application information. Developers across several scrum teams that had to migrate their applications found their migration time reduced from hours to minutes by using both programs together. The secondary program could be applicable to any other business that has to migrate applications in a similar manner. With proper maintenance of the application information in the script, the helper can be used as a general template for similar purposes. Testing should also always be done before executing the script in case there are applications that do not belong within a team anymore.

1. Introduction

There is a task that must be completed within several weeks. It involves hundreds of applications and each one needs to be updated the same way. One option is to manually go through each one and update them according to current guidelines. Is this solution feasible given the time frame? The method is possible, but it is not efficient.

The data engineering teams that were assigned this task had over 20 applications. Each application included projects of different stages, averaging around six projects per application. The migration process involved updating a file for each project to have a recent version number so that testing and deploying the applications would run without error. All of this was to be done on the platform GitHub. The DevOps team created a shell program to assist the process. However, it resulted in bugs and reworks every week along with limited documentation. This was a significant issue that the data engineering team faced and had to fix as soon as possible. I worked alongside with the data engineering teams proposed a solution to develop another shell program that would be able to assist the initial program. The new shell program was developed successfully and also increased the efficiency of the migration process.

2. Related Works

Shell scripts and the benefits they provide for automating tasks have been demonstrated across multiple accounts. Yu and Todd (2018) include a suggestion for a programming course that teaches students scripting fundamentals. More importantly, it emphasizes the various fields of work that can benefit greatly from the automation that shell and other scripting languages may provide. One of these fields includes SQL databases. The advocacy of shell scripting from this source supports the solution chosen for our project.

Another source that agrees with the benefits of shell scripting is Bipin (2022). It gives credit to mainframe programming languages such as Python and Java, but also declares that shell scripting is still very useful especially for day-to-day tasks.

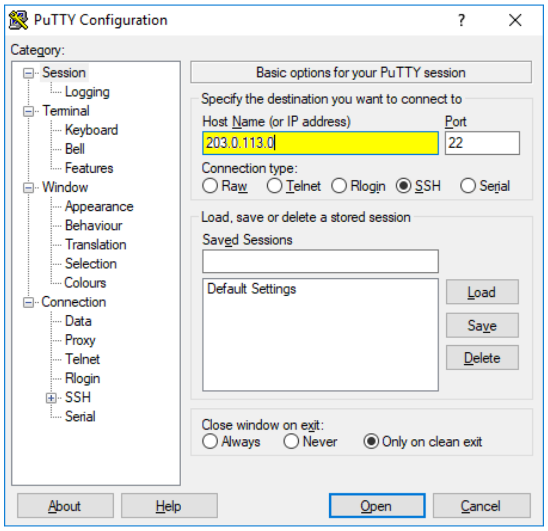
3. Program Design

Senior engineers got together with the intern, who proposed the solution, to discuss the design and implications of this secondary shell program. The following subsections describe each part of the process.

3.1 Development Server

The program was developed among one of the many servers hosted by the company. One of the development servers was chosen because a new program was going to be created. Other servers were designated for system integration and user acceptance testing only, as well as production deployment. The server was connected through a Linux terminal emulator using a software called PuTTY, which enables remote secure connections by requiring proper authentication when a host name is being connected. All development and minor testing was done through PuTTY.

As shown in Figure 1, the interface of PuTTY on bootup. Before connecting to a server, it requires a host name or IP address to connect to and the port number which typically is set to 22. Refer to Figure 2 to view a simulated Linux shell window. Any valid Linux command can be executed within this window.



Specify the destination you want to connect to

Host Name (or IP address): 203.0.113.0

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Close window on exit: Always Never [✓ Only on clean exit]

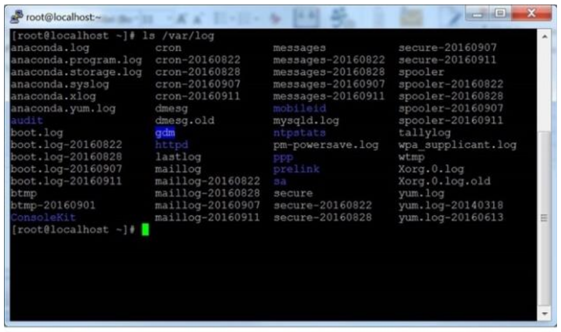
Figure 1: Example interface of PuTTY on bootup

3.2 Requirements

The secondary shell program had to be unique from the initial. Therefore, several requirements had to be incorporated into the program. The program needed some form of user interface and a way for a user to input some response on execution. There also had to be input validation to ensure that any input used further into the program is acceptable. The program also required storage of the database applications in a static structure, something that can only be modified when editing the source code, not through external means. Last, error handling was to be implemented to catch an issue to prevent the program from crashing later on.

3.3 Automation

The main design of this secondary program was to automate the migration process. The requirements all together allowed this to happen. There was now no need to manually check if a user's input was valid and based on the input, the initial script would execute an appropriate number of times on its own without needing to be manually run over and over. The program will also automatically exit based on error encounters so that updating incorrect application information does not occur.



```
[root@localhost ~]# ls /var/log
anaconda.log          cron             messages            secure-20160907
anaconda.program.log  cron-20160822    messages-20160822   secure-20160911
anaconda.storage.log  cron-20160828    messages-20160828   spooler
anaconda.syslog       cron-20160907    messages-20160907   spooler-20160822
anaconda.xlog         cron-20160911    messages-20160911   spooler-20160828
anaconda.yum.log      dmesg            mobileid            spooler-20160907
audit                 dmesg.old        mysqld.log          spooler-20160911
boot.log              gdm              ntpstats            tallylog
boot.log-20160822     httpd            pm-powersave.log    wpa_supplicant.log
boot.log-20160828     lastlog          ppp                 wtmp
boot.log-20160907     maillog          prelink             Xorg.0.log
boot.log-20160911     maillog-20160822 sa                  Xorg.0.log.old
btmp                  maillog-20160828 secure              yum.log
btmp-20160901         maillog-20160907 secure-20160822     yum.log-20140318
ConsoleKit            maillog-20160911 secure-20160828     yum.log-20160613
[root@localhost ~]#
```

Figure 2: A sample terminal window after connection with server is established

3.4 Documentation

As a 10-week intern, I also had to provide documentation before I had finished my internship. For that reason, documentation was also part of the design of the program in order to provide a reference for the program's functions. The documentation teaches other engineers how each part of the program works in case further development is needed in the future.

3.5 Challenges

I received no guidance other than the requirements established. The main goal that the senior engineers wanted me to achieve was automation. However, with limited meetups and constant occurrences of staff being out of office, I had difficulty requesting assistance to make sure the features implemented appeared correct.

4. Results

Upon completion of the secondary shell program and documentation, I tested the program to simulate the increase in efficiency that migration process now has. Running the secondary shell program reduced runtime of the initial shell program from two hours to under ten minutes. I presented the demonstration to the senior engineers to prove this efficiency. With the demonstration, all features of the requirements were also tested and proven to function properly with no bugs found. The program was delivered to other senior engineers with instructions from the documentation to teach them the purpose and usage of this program.

5. Conclusion

A handful of other tasks needed to be done by the end of 2022 and this task in particular was one of the most tedious by far. However, ever since the development of the additional shell program, the workload has decreased for the data engineers involved in this migration process. Additional improvements are being discovered through each usage of the program as the process comes to an end.

6. Future Work

The program has plenty of features that could be added, such as more flexibility to other databases that exist within the business. Further testing could also be done to ensure stronger quality in case this program goes into production for client use. Overall, the automation from the shell program I implemented is evidently effective in the business however, I would encourage other students and researchers to consider utilizing shell scripts for their own needs in other fields of work that may be relevant.

References

[1] Cai, Y. and Arney, T. 2018. Scripting for Administration, Automation and Security. In *Proceedings of the 19th Annual SIG*

Conference on Information Technology Education (SIGITE ’18), Association for Computing Machinery, New York, NY, USA, 147. DOI:https://doi.org/10.1145/3241815.3241829

[2] Patwardhan, B. 2022. Focus: Shell Scripting is Still Going Strong. *Open Source for You*. Retrieved September 22, 2022 from https://www.proquest.com/advancedtechaerospace/docview/2655026351/citation/6741DC78962F45DDPQ/1