Phocas DevOps Technical Challenge

Matthew Kent

Files Provided:

PhocasDevOpsChallenge-MattKent

CheckForUpdate (.exe to run solution for Challenge 1)

DirectorySetup (.cmd file to build required directory or copy over folders provided)

VisualStudio2019ProjectFiles (Project Solution)

ProgramCode (Copy of the code from ProjectFiles for quicker access to view)

TerraFormManager (Used for changing program versions)

\*\* This Word doc \*\*

# Challenge #1

Code written in Visual Studio 2019 using C#, which also creates custom .BAT files to run CMD commands to tackle the privilege issues with dealing with Program Files. I chose this as I knew I’d be able to work with extracting a string for the download link needed, while having limited experience with C# allowed some knowledge on creating the necessary tool.

I mainly have experience in PHP/HTML/CSS/JS/Python, however after taking a C# class during my degree it definitely felt like it would provide the best desired outcome for the problem statement out of my main scripting/programming languages that I’m comfortable with.

While this may have been possible with a CMD/Powershell combo, this seemed like the more logical way to go about it. By being able to place variables inside the created .BAT files, this allowed for more custom commands when creating specific file paths for backups (Challenge #3) or moving files to specific directories. While also making use of libraries like HTML Agility Pack to parse the required download link and find the newest version of Terraform.

**Logic Flow / Rough Pseudo Code**

Program Start

Create Directories if not already created

If (Temp VersionInfo.txt does not exist)

MessageBox(No version history, YesNo to update)

If (result == Yes)

Create .BAT file to copy file over from Program Files

Launch .BAT file

If (result == No)

Close Program (Can’t function or make decisions without knowing the current version)

Set CurrentVersion String from Temp\VersionInfo.txt

Load the terraform downloads page into a HtmlDoc using the HTMLAgilityPack

Search through the documents for all <li></li>, then loop through till it matches the string used for the Windows 10 64 bit .zip file download ("windows\_amd64.zip")

Split the string to gather the newVersion string (1.0.7 etc), and the new download link.

If (currentVersion does not equal newVersion)

MessageBox(Update Available, YesNo to update)

If (result == Yes)

Create new dir for the new version to download to with the newVersion string included in the name

Download new version to new dir

Updated VersionInfo.txt inside the Temp folder

Set strings for copying/overwriting Program Files VersionInfo.txt to new updated copy, Unzipping the new .zip, Moving the new .zip file, Moving the new folder to Program Files once unzipped.

Create and run new .BAT file to execute

Unzip file while in \Temp to reduce privilege issues

Move folder over to Program Files with .zip and unzipped .exe

Delete current .exe inside Program Files\terraform

Move new .exe to replace the deleted .exe

New MessageBox to show update has succeeded, with new Version number

If (result == No)

MessageBox(Terraform ‘versionNumber’ is already up to date)

# Challenge #2

Use the Windows Task Scheduler to run the .exe weekly, makes use of a great tool that’s available on Windows machines.

Create the task as TerraformUpdater

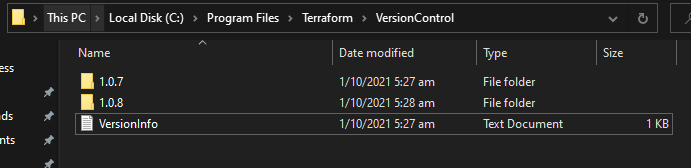
Trigger it at the required time (Monday 9am Weekly)

Create ‘Action’ of starting the Updater.exe

# Challenge #3

Terraform Manager

* Allows for backup to previous versions with ease, if they are still in the VersionControl
* The folder management and program help deal with rolling back to previous versions
* As no additional packages are needed it can run as a single .exe
* I would include the update functionality, however it’s not stable enough to be used yet.
* Allows for ‘re-applying’ a version that is in the version control directory.

Showing the program can see both available versions, each version file contains their respective .zip file.

