The art of the shutdown – A PowerCLI adventure

One of the most talked about processes in VMware environments is the process of shutting down. While we strive to create "always on" systems, the reality is that we will at some point have to shut down a vSphere environment for one reason or another. Whether it is a planned event, or if we have to be prepared to react to a power loss situation we can leverage one of the greatest tools we have available for us: PowerCLI.

Imagine if you will (I always think of Rod Serling when I start with that), a VMware infrastructure environment which is about to lose power. There are a number of ways to deal with this situation. A typical scenario for a physical server environment is a UPS is attached to your building power and upon losing building power the UPS takes action based on battery status. At some point it is programmed to send a shutdown signal to UPS agents installed on your Windows or *nix servers.

Here is the catch. VMware ESXi, while based on the *nix kernel, is not an OS which is managed the same way. There are no native methods to add components such as UPS agents and non-core packages. One alternative method is, we can take is to have a server which receives the signal from the UPS via an agent, and then runs a PowerCLI shutdown script to make the shutdown as clean as possible.

Don't put all of your hopes in the cleanest shutdown of course. The idea for this is to quickly bring as many resources down as possible before a hard stop because of power loss. The script is built to try to do VM tools initiated shutdown and if it does not respond in time then we issue a power off command.

The way that I've built the script was to follow the natural flow that I though I would follow if I was interactively working on the issue. Whether right or wrong, this is how I choose to write all my scripts. Since I don't have a programming background (which may be obvious by some of my code) I work in flowchart logic and write all my scripts that way to start with.

My datacenter design has a virtualized vCenter server which adds an extra layer of complexity because I use the vCenter to initiate most of the shutdown so it is key that it is the last guest to be shut down.

The logic I follow is this:

1. Find out which cluster I am working on and turn DRS to manual – this stops the VM guests from piling onto the available hosts as we shut them down which creates massive traffic and I/O to the environment thus slowing the whole process down
2. Find which host is running my virtual vCenter instance so that I can shut that host down last
3. Enumerate the guests on each host and for each we will check for VM Tools, issue VM Tools shutdown command, wait, issue Power Off command
4. Once all guests are off on each hosts, issue a power down command to the host server
5. Once all hosts are down, run the same process on the last host which is handling our virtualized vCenter server

There are numerous challenges with this type of script. They include a lot of situational variables like the behaviour of specific guests and hosts which could slow down the process or potentially halt

the script altogether in the middle of this very important process.

One of the greatest challenges is how to run the script in a “test mode”. While most of the commands have the -WhatIf parameter available, the host shutdown option is done using this command:

```
$vCenterHost | ForEach { Get-View $_.ID } | ForEach {
$_.ShutdownHost_Task($TRUE)}
```

Trust me…there is no good way to WhatIf that one. So the workaround is that you can issue command line arguments to the script to tell the commands whether to run in test or real mode and then wrap each command in an If statement.

The trade off for running lots of If statements throughout the script is performance. But because I wanted to be able to run the script in both modes it seemed like a worthwhile trade. Another challenge, is that it lengthens the script quite a bit to add in all of the logic for checking each command for run mode.

So now is the real test. I’m opening it up to the community for advice. I am sure of one thing which is that there are literally dozens of ways to achieve this result. I want to have people bring back any comments, suggestions and tips on how to optimize this script and the process in general.

Key things I know are points of contention:

- This design shuts down each host synchronously so there is a risk that a problem in the process will leave the remaining hosts online and subject to a hard power off
- Should there really be a test mode?
- Redundancy in the loop structure (btw I’m not a programmer as you may have noticed)

So without further adue, here is the link to the script file. Just download and rename to .ps1 and you are ready to go!

https://discoposse.com/media/myscripts/ShutdownDataCenter.ps1.txt

Big thanks to @mwpreston who gave me the inspiration for the script (http://blog.mwpreston.net/shares/) and for the tip to add the inventory of VM guests which are in a PoweredOn state to be able to do a programmatic power on. The CSV file created in the shutdown script can be used as a source to power on the devices in a steady power up using this script:

```
$vms = Import-CSV PowereredOnVMGuests.csv
foreach ($vm in $vms)
{
   Start-VM $vm.Name
   sleep 5
}
```

I’m looking forward to hearing from you all with any feedback or tips