Updated Sep 27, 2023

Milestone 4.1 - Automation with pyvmomi

💡Any systems administration activity that is characterized by a good deal of pointing and clicking, selection of options, typing of free text such as IP addresses, names of VMs and arbitrary commands is prone to error. Automation makes systems administration tasks more consistent and easily repeatable. VSphere has a vast array of automation options including ansible, powercli, the REST API, Terraform and likely others. We are going to use python3 in conjunction with the pyvmomi library to learn about automation of enterprise virtualization tasks.

You will need to brush off your python skills.

Resources

The following pyvmomi samples are great and provide simple driver programs to invoke some common vsphere operations. You are encouraged to import and use functions from the tools directory in your own programs.

- https://github.com/vmware/pyvmomi-community-samples
  - These samples have all the vsphere examples you will need to complete your program. You will just need to integrate it and modify as you see fit.
- This is a great tutorial for making a basic python program with a menu.
- Demonstration of interactive login using pyvmomi
- Instructor Demonstration (without the code review)
- Another awesome resource

Prerequisites

The following tech journal excerpt shows a method to install the dependencies for pyvmomi and visual studio code on mgmt1. If you've not enabled a method to get to mgmt01 full screen with copy/paste it is time to do so. See early suggestions on Chrome Remote Desktop.

Updated Sep 27, 2023

pyvmomi

installation

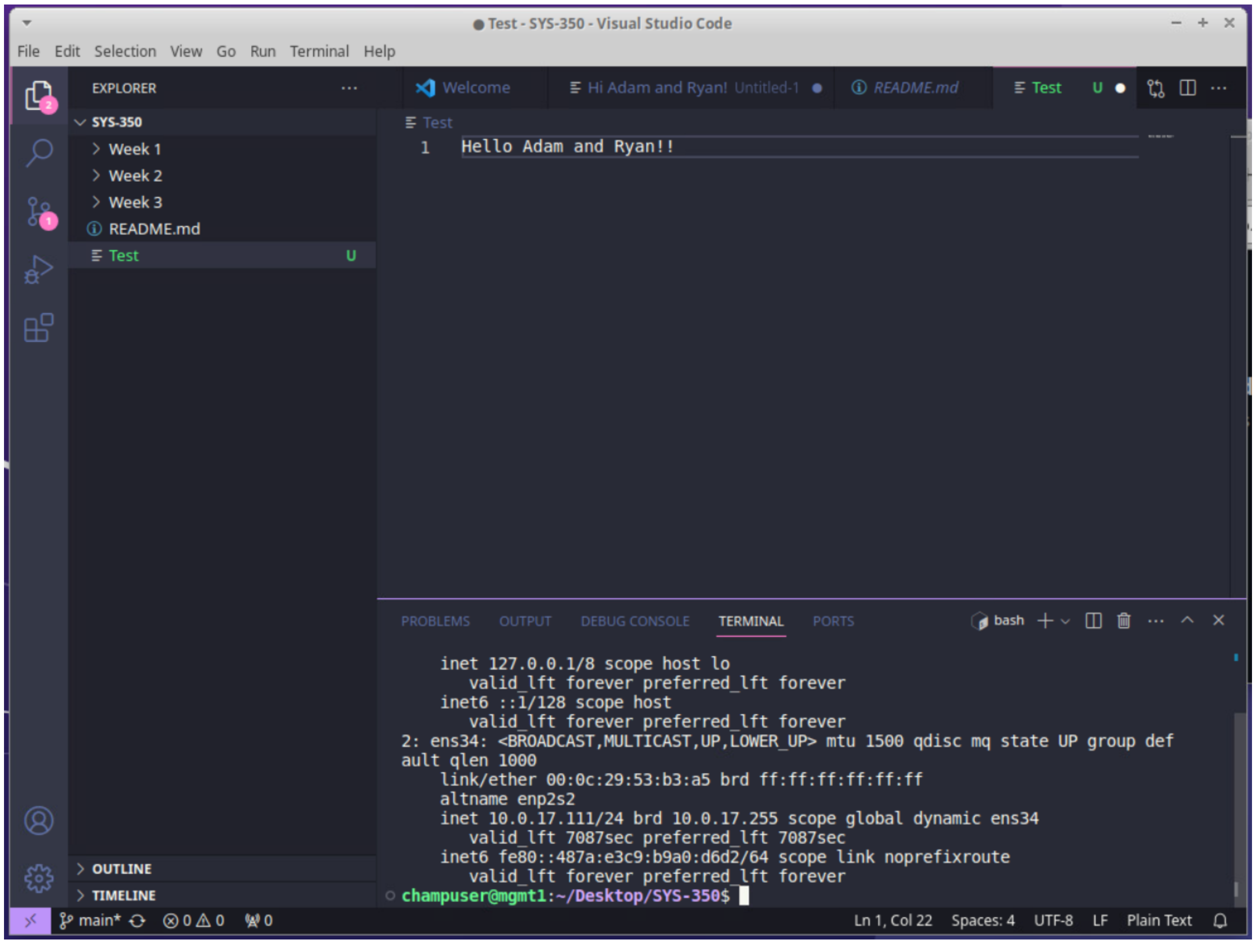
```
sudo apt update
sudo apt install python3-pip
#install as logged in user
pip3 install wheel
pip3 install pyvmomi
pip3 install pyvim
```

vscode

```
sudo snap install code --classic
```

Updated Sep 27, 2023

`Deliverable 1.  Figure out how to clone your course git repository. Open the base directory in visual studio code(1), and cover in your video demo a view similar to the one below that also shows a terminal with your mgmt1 IP address(2)`


Test - SYS-350 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
SYS-350
Week 1
Week 2
Week 3
README.md
Test U
Welcome
Hi Adam and Ryan! Untitled-1
README.md
Test U
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash
main* ⊗ 0 ⚠ 0 0
Ln 1, Col 22 Spaces: 4 UTF-8 LF Plain Text

```
Hello Adam and Ryan!!
```

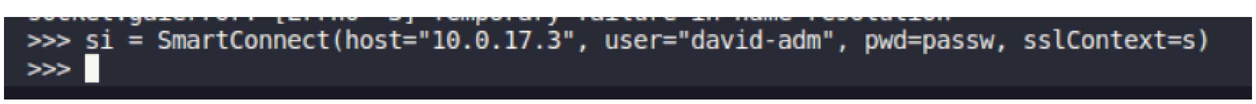
```
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group def
ault qlen 1000
    link/ether 00:0c:29:53:b3:a5 brd ff:ff:ff:ff:ff:ff
    altname enp2s2
    inet 10.0.17.111/24 brd 10.0.17.255 scope global dynamic ens34
       valid_lft 7087sec preferred_lft 7087sec
    inet6 fe80::487a:e3c9:b9a0:d6d2/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
champuser@mgmt1:~/Desktop/SYS-350$
```

An interactive session

Watch the following video. Extend this interactive example into a python program that has the required features.

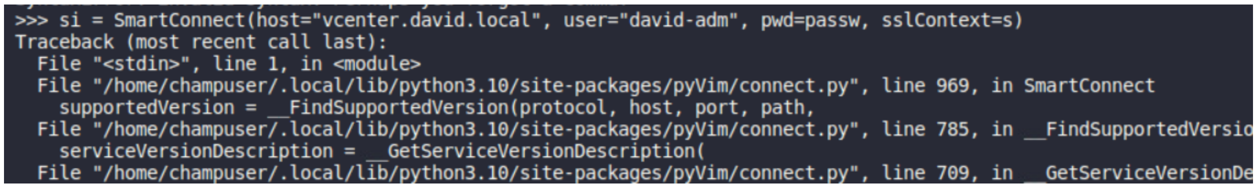
`Deliverable 2.  Demo  your own interactive session with vcenter via pyvmomi.  Print out an element of the aboutInfo object.  Similar to the one below.`

Note: SmartConnect function is cut off a bit in screenshot - but need to use the passw variable and sslContext parameters. Review docs on the pyvim SmartConnect function!

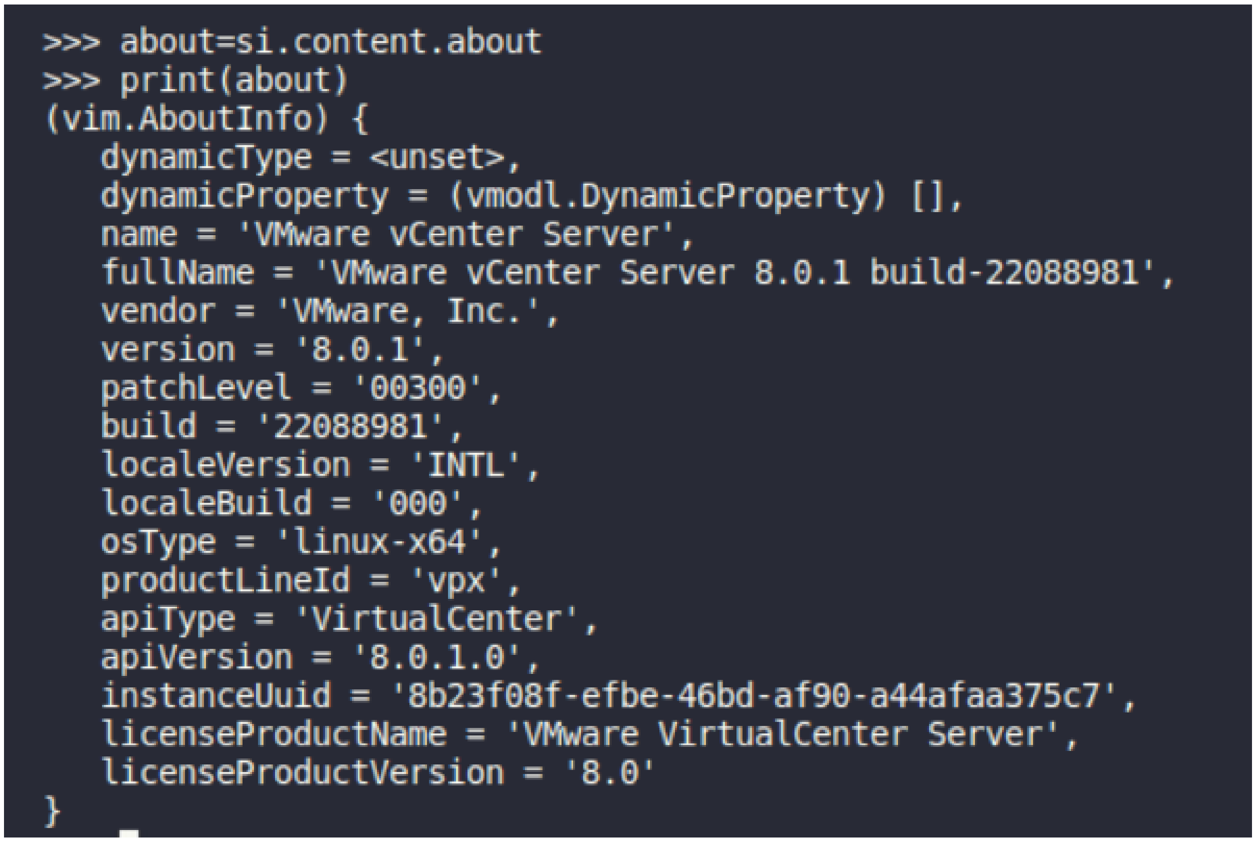


```
>>> si = SmartConnect(host="10.0.17.3", user="david-adm", pwd=passw, sslContext=s)
>>> 
```

Updated Sep 27, 2023



```
>>> si = SmartConnect(host="vcenter.david.local", user="david-adm", pwd=passw, sslContext=s)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/champuser/.local/lib/python3.10/site-packages/pyVim/connect.py", line 969, in SmartConnect
    supportedVersion = __FindSupportedVersion(protocol, host, port, path,
  File "/home/champuser/.local/lib/python3.10/site-packages/pyVim/connect.py", line 785, in __FindSupportedVersio
    serviceVersionDescription = __GetServiceVersionDescription(
  File "/home/champuser/.local/lib/python3.10/site-packages/pyVim/connect.py", line 709, in __GetServiceVersionDe
```



```
>>> about=si.content.about
>>> print(about)
(vim.AboutInfo) {
   dynamicType = <unset>,
   dynamicProperty = (vmodl.DynamicProperty) [],
   name = 'VMware vCenter Server',
   fullName = 'VMware vCenter Server 8.0.1 build-22088981',
   vendor = 'VMware, Inc.',
   version = '8.0.1',
   patchLevel = '00300',
   build = '22088981',
   localeVersion = 'INTL',
   localeBuild = '000',
   osType = 'linux-x64',
   productLineId = 'vpx',
   apiType = 'VirtualCenter',
   apiVersion = '8.0.1.0',
   instanceUuid = '8b23f08f-efbe-46bd-af90-a44afaa375c7',
   licenseProductName = 'VMware VirtualCenter Server',
   licenseProductVersion = '8.0'
}
```

Updated Sep 27, 2023

Your Python Program

Python
```python
# Import necessary modules
import getpass, warnings, ssl
from pyVim.connect import SmartConnect
from pyVmomi import vim
warnings.filterwarnings("ignore", category=DeprecationWarning)

# Define a class for vCenter connection
class VCenterConnection:
  def __init__(self, creds_filename):
    # Initialize the class with credentials and SmartConnect instance
    self.creds = self.readCreds(creds_filename)
    self.si = None

  # Function to read credentials from a file
  def readCreds(self, filename):
    try:
      with open(filename, 'r') as file:
        creds = {}
        for line in file:
          key, value = line.strip().split('=')
          creds[key.strip()] = value.strip()
        return creds
    except Exception as e:
      print(f"Error reading credentials: {e}")
            return None

    # Function to login to vCenter
    def login(self):
        if self.creds is None:
            return
        passwd = getpass.getpass()
        try:
            # Establish an SSL connection to vCenter
            s = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
            s.verify_mode = ssl.CERT_NONE
            self.si = SmartConnect(host=self.creds.get("hostname"),
user=self.creds.get("username"), pwd=passwd, sslContext=s)
    except Exception as e:
      print(f"Error connecting to vCenter: {e}")

    # Function to display session information
    def sessionInfo(self):
```

Updated Sep 27, 2023

```
        p = self.si.content.sessionManager.currentSession
        print("\nSession Info\n")
    print("-----------------------------------------------------------------")
    print("Session Key: " + p.key)
    print("Session User Agent: " + p.userAgent)
    print("Session User/Domain: " + p.userName)
    print("Session Server: " + self.si.content.about.name)

  # Function to retrieve and display VM information
  def vmInfo(self):
    search = input("Search for a VM: ")
    print()

    for child in self.si.content.rootFolder.childEntity:
      dc = child
      vmfolder = dc.vmFolder
      vmlist = vmfolder.childEntity

    for vm in vmlist:
      if vm.name !=
"%2fvmfs%2fvolumes%2f64f20cb8-49d66acc-314c-3cecef4663da%2fpfsense%2fpfsense.vm
x":
        if search in vm.name:
          print("Name: " + vm.name)
          print("Power State: " + vm.runtime.powerState)
          print("Number of CPUs: " + str(vm.config.hardware.numCPU))
          print("Memory in GB: " + str(vm.config.hardware.memoryMB / 1024))
          print()

          # Retrieve and display VM's IP address
          ip_address = None
          try:
            for guest in vm.guest.net:
              if guest.ipConfig.ipAddress:
                ip_address = guest.ipConfig.ipAddress[0].ipAddress
                break
          except AttributeError:
            print("IP Address: Not available (VMware Tools may not be
installed)")

          if ip_address:
            print("IP Address: " + ip_address)
          else:
            continue
```

Updated Sep 27, 2023

```
  # Function to display the main menu
  def display_menu(self):
    while True:
      print("\nMenu:")
      print("1. Display Session Info")
      print("2. Search for VMs")
      print("7. Exit")

      choice = input("Enter your choice (1-7): ")

      if choice == '1':
        self.sessionInfo()
      elif choice == '2':
        self.vmInfo()
      elif choice == '7':
        print("Goodbye!")
        if self.si is not None:
          # Terminate the current session when exiting

self.si.content.sessionManager.TerminateSession(self.si.content.sessionManager.
currentSession.key)
        break
      else:
        print("Invalid choice. Please enter a valid option (1-7).")

# Entry point of the script
if __name__ == "__main__":
  # Create an instance of VCenterConnection with the credential file
  vcenter = VCenterConnection("creds.txt")
  # Perform the login
  vcenter.login()
  # Display the main menu
  vcenter.display_menu()
```

Updated Sep 27, 2023

Deliverable 4: Tech Journaling - Technical (Demo)
Briefly demonstrate your documentation and code for this milestone. The demo provided allows us to easily navigate your journal's technical components. The journal is easy to read, there are screenshots where appropriate and formatted commands when it would be useful to copy paste. You link to repository content rather than formatting lengthy commands in markdown. Content that is suitable to multiple courses should be elevated and linked to rather than hidden in your course specific wiki.

Deliverable 5: Tech Journaling - Reflection (Demo)
You discuss the task in general terms and you highlight where you went off course, where you burned time and any breakthroughs you had.