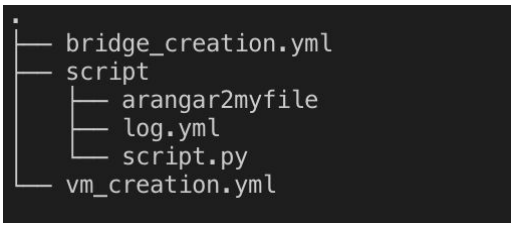
README

Solution 4: Ansible

1. Directory structure:



   a. Main playbooks:
      i. vm_creation.yml
      ii. bridge_creation.yml
   b. Inventory:
      i. arangar2myfile
2. *Execution:*
   a. ansible-playboook vm_creation.yml -vvv -i <inventory_filename>
   b. ansible-playboook bridge_creation.yml -vvv -K -i <inventory_filename>
   (-K option will ask for password as privilege escalation is enabled for ovs-vsctl task)
   NOTE:
   - Fails if bridge already created, idempotency check could be added.
   - Template file location is with respect to host machine setup.

   c. ansible-playbook log.yml -vvv -K -i <inventory_filename>
   NOTE:
   - Inventory contains ssh details with respect to the host machine only.

Solution 5: Python Libvirt API

We are submitting 3 scripts:

1. **info.py**
   a. *Execution*:
      i. python info.py
   b. Displays the Host and Guest VMs information
2. **usage.py**
   a. *Execution*:
      i. python usage.py CPU
      ii. python usage.py MEM
   b. Sorts the VMs according to either CPU or MEM
   c. Further, if CPU is given as the argument:
      i. Asks for a threshold value and writes alert messages for VMs having CPU usage higher than that.
      ii. Creates / Appends to alert.txt in the same folder
      iii. Format: (Vm name, time stamp, CPU usage)

d. NOTE:
    i. it only takes one argument (CPU/MEM)
    ii. If no arguments are given, throws an exception and asks to give either of the arguments

3. bonus.py

a. Execution:
    i. python bonus.py CPU
    ii. python bonus.py MEM
b. Sorts the VMs according to the moving averages of either CPU or MEM
c. Assumptions:
    i. We ask for the following from user (int):
        1. Polling interval
        2. Window size
        3. Total time
    ii. The script computes the CPU/MEM usage at every polling interval until the total time which results in a list of usages for the respective VMs.
    iii. Further, we compute the moving averages with the given values of the window size.
    iv. At this point, we have moving averages of the usage for each VM in the following format:

'ssujalVM1':[22.257417715065312,22.25741674710963,22.257415779153945,22.2574147955]

'ssujalVM2':[34.9457963944386,34.945795572627226,34.9457945140138,34.9457934554003]

    v. Finally, we print out the sorting order for every polling interval according to its respective value of the VM
