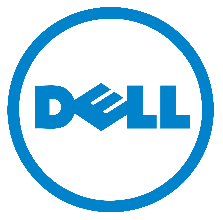
BROCADE VEPC Automation scripts

Design Reference



Date: Sep 14, 2015

Contents

[Revision History: 2](#_Toc430013583)

[1 Functionality of Scripts 3](#_Toc430013584)

[1.1 vEPC\_Deploy.py 3](#_Toc430013585)

[1.2 scale\_up.py 7](#_Toc430013586)

[1.3 scale\_down.py 9](#_Toc430013587)

[1.4 vEPC\_termination.py 10](#_Toc430013588)

# Revision History:

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Description** |
| 1.0 | Sep 14, 2015 | Abdul Rehman |  |

# Functionality of Scripts

All the scripts generate logs during execution. To see those logs, go into the /vEPC/logs/ directory. It contains a total of 8 files, 2 for each script; one file is for activity log i.e. the total flow of script in which it deploys VCM components and the second is error log in case if any error occurs during the execution of script.

1. deploy\_date\_time.log and deploy\_error\_date\_time.log
2. scale\_up\_date\_time.log and scale\_up\_error\_date\_time.log
3. scale\_down\_date\_time.log and scale\_down\_error\_date\_time.log
4. terminate\_date\_time.log and terminate\_error\_date\_time.log

## vEPC\_Deploy.py

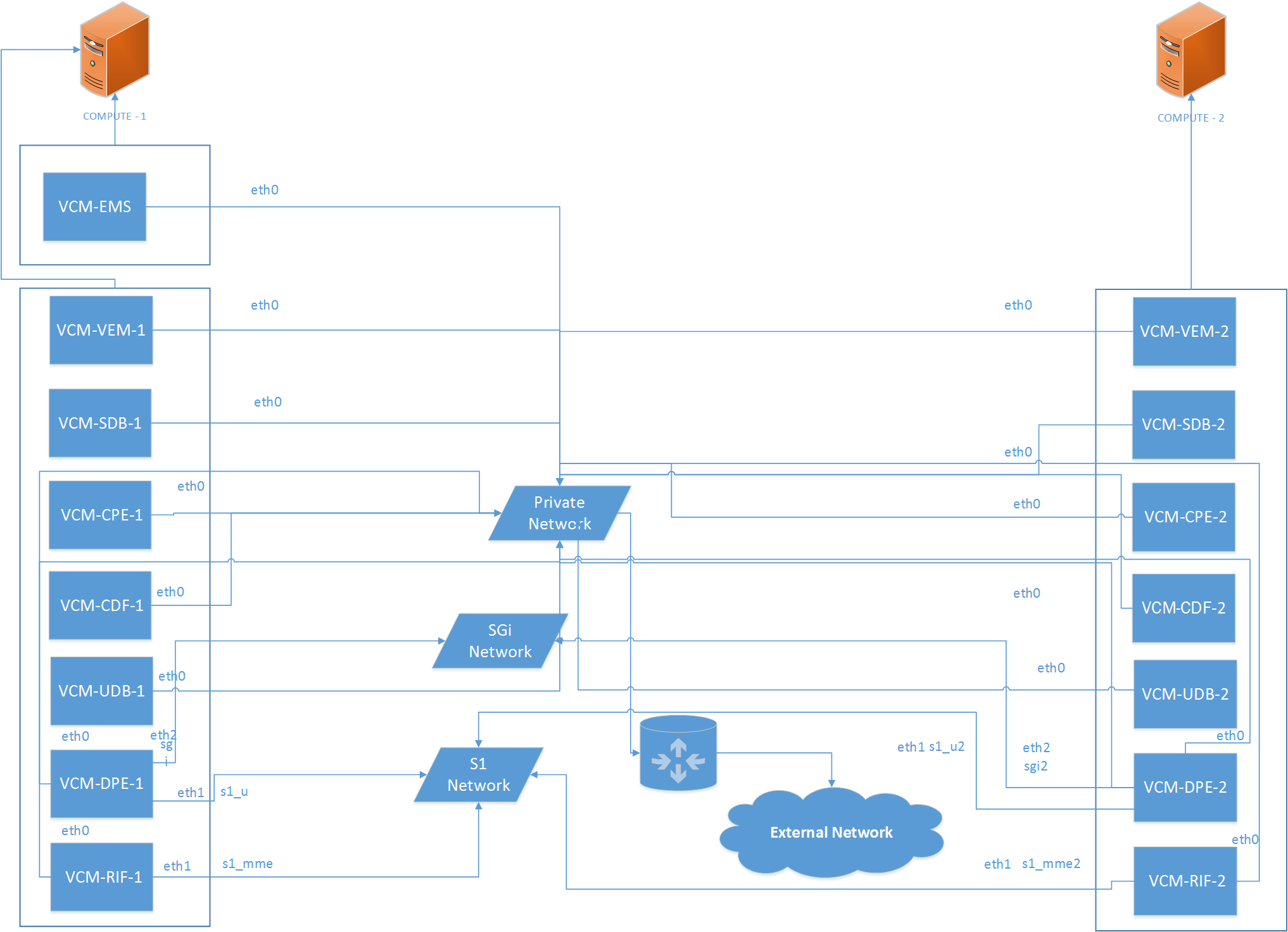


Figure : Complete VCM deployment along with network connectivity in High Availability

VCM components are deployed in high availability and each component requires the creation of 7 instances, with different requirements of ports and networks attached to it.

When this script runs, it gets the required parameters from creds.txt file located in vEPC directory and add them to the configurations.json file.

Script then checks for the required resources on compute nodes for initial deploy and if not available required and available resources values are stored in activity log file vEPC\_deploy\_date\_time.log file and script gives warning to the user if it still wants to continue and based on the input continues or terminates.

Glance images of VCM and EMS are then created if they don’t already exist. After that script checks if components of vEPC exist and if there are any, it exits with a prompt to first run the vEPC termination script.

If no vEPC components exists, script creates Availability Zones with name Compute 1 and Compute 2 if not already exist and assign 1 compute node to each zone. After that it creates networks S1 and SGi and assign port s1\_u, s1\_mme, s1\_u2 and s1\_mme2 to network S1 and port sgi, sgi2 to network SGi.

Then script starts deploying VCM-1 components on Compute 1, i.e. the creation of 7 instances with different ports and networks attached to it. Instances are created in the following order and names:

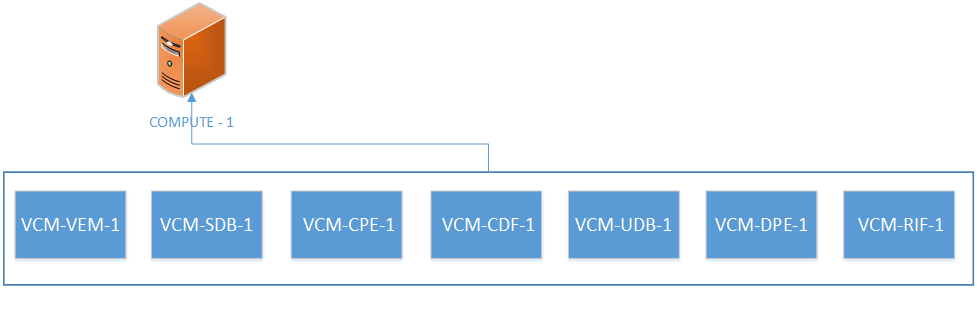


Figure 2: VCM-1 Deployment Flow

Instance DPE-1 is connected to the port s1\_u of S1 network and sgi port of SGi network while RIF-1 is connected with port s1\_mme of S1 network, both of these instances along with all the other 5 instances will also be connected to the private network of Openstack; name of which will be provided by user in creds.txt file.

After instances creation, it tries to ping all the 7 instances to check whether they booted up or not. If any instance didn’t boots up, script waits for a short time and again tries to check if it booted up. It keeps on waiting and trying again until timeout occurs and script terminates if the instance is taking too long to boot as there may be a problem with the networking of Openstack or its deployment because of which instance are not accessible via floating IPs.

When all the instances boots up, it uses SSH to access each instance and changes the hostnames of all instances. After changing the hostnames, all the 7 instances are rebooted and scripts waits for some time to allow instances to boot up before running the deploy script.

Now script again accesses the instances via SSH and runs the deploy script with instance ID 1. For VEM, UDB and RIF, some additional required configurations are also made. After running the deploy script in instances, validate deploy is executed to check initial deploy script executed successfully and VNFC service is ready to run.

It then updates the ports s1\_u, s1\_mme and sgi to allow traffic from IP addresses from S1 and SGi networks.

After allowing IP addresses, VCM services on VEM-1 and SDB-1 are started and validated using validate deploy script and if there isn’t any error, script outputs that VCM services are running on VEM-1. When the services successfully runs on VEM, VCM configuration file is executed via SSH in VEM-1 and output is shown at command line during the execution of script.

Services on rest of VCM components CPE-1, CDF-1, UDB-1, DPE-1 and RIF-1 are started and output from each instance is shown sequentially on CLI.

Then LTE provisioning script is executed on UDB-1 and output is shown on the CLI.

After the creation of VCM-1 components and starting of services on all its instances, VCM-2 instances are been deployed on Compute 2.

Deployment of VCM-2 components is similar to VCM-1 with slight changes in instances names, ports assignment and starting the services.

7 VCM-2 instances are created in following names and order:

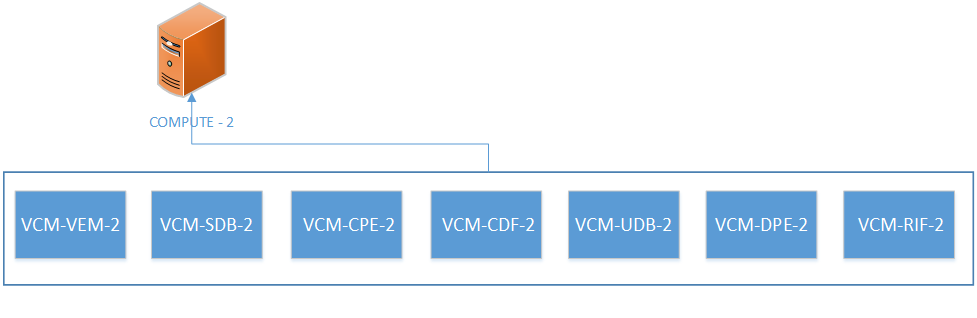


Figure 3: VCM2 Deployment and Network Connectivity Flow

Instance DPE-2 is connected to the port s1\_u2 of S1 network and sgi2 port of SGi network while RIF-2 is connected with port s1\_mme2 of S1 network, both of these instances along with all the other 5 instances will also be connected to the private network of Openstack name of which is provided in creds.txt file.

After instance creation, it tries to ping all the 7 instances to check whether they booted up or not. If any instance didn’t boots up, script waits for a short time and again tries to check if it booted up. It keeps on waiting and trying again until timeout occurs and script terminates if the instance is taking too long to boot as there may be a problem with the networking of Openstack or its deployment because of which instance are not accessible via floating IPs.

When all the instances boots up, it uses SSH to access each instance and changes the hostnames of all instances. After changing the hostnames, all the 7 instances are rebooted and scripts waits for some time to allow instances to boot before running the deploy script.

Now script again accesses the instances via SSH and runs the deploy script with instance ID 2. For VEM, UDB and RIF, some additional required configurations are also made. After running the deploy script in instances, validate deploy is executed to check initial deploy script executed successfully and VNFC service is ready to run.

It then updates the ports s1\_u, s1\_mme and sgi to allow traffic from IP addresses from S1 and SGi networks.

After allowing IP addresses, VCM services on VEM-2 and SDB-2 are started and validated using validate deploy script and if there isn’t any error, script outputs that VCM services is running on VEM-2. When the services successfully runs on VEM-2, VCM configuration file is executed via SSH in VEM-2 and output is shown at command line during the execution of script.

Services on rest of VCM components CPE-2, CDF-2, UDB-2, DPE-2 and RIF-2 are started and output from each instance is shown sequentially on CLI. Then LTE provisioning script is executed on UDB-2 and output is shown on the CLI.

In the end after successful deployment of VCM-1 and VCM-2 components, script checks for the EMS instance and if it doesn’t exit, instance for EMS is created on Compute 1.

Hostname file for EMS is created based on its IP address assigned to it and it is then copied into the EMS to change the hostname and EMS is rebooted to allow host-name changes to take effect.

After the EMS boots up, service is started and output from the instance is shown on CLI and script exits with showing the web URL to start EMS GUI.

## scale\_up.py

Instances required during execution of scale-up process are SDB, CPE and DPE and this script deploys the instances for unit value i.e. each time the script runs, it scales up VCM components for one value.

The script selects the compute nodes in round robin i.e. when the script runs, it changes the availability zone for the deployment of VCM instances for the first time zone will be Compute 1, for the second time zone will be Compute 2 and so on. This is to ensure that load on both compute nodes remain balanced.

Scale-up script deploys 3 new instances SDB, CPE and DPE every time it’s executed. When the script runs for the first time after the vEPC initial deployment, it starts deploying instance with names VCM-SDB-3, VCM-CPE-3 and VCM-DPE-3. After first execution, script checks the last scale-up value and deploys new instances after incrementing that value by 1.

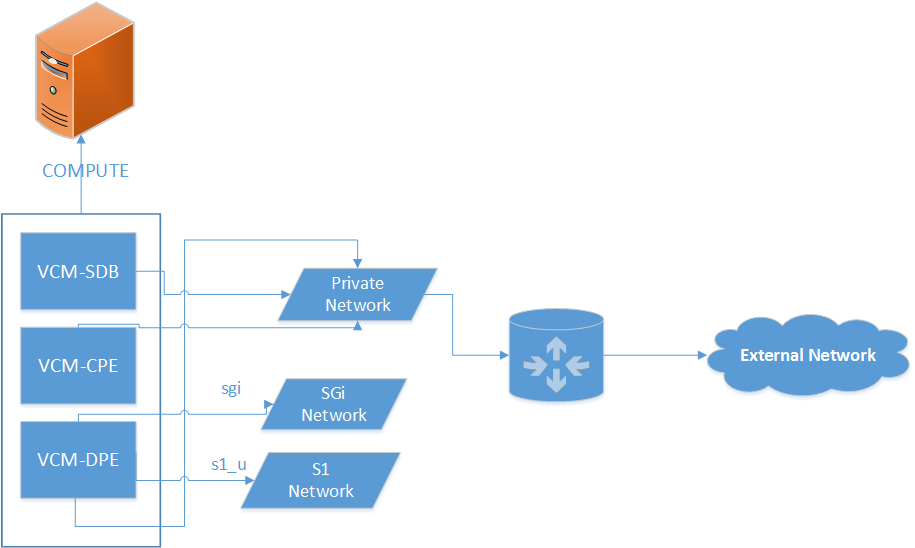


Figure 4: vEPC Scale-up process

After the instances are created, ping is performed to each instance to check if it is booted or not and waits for any instance if it’s taking time to boot. Hostname files are created based on the scale-up value and after all instances are up, hostname configuration is done on each instance and it’s restarted for the changes to take effect.

Deploy script is then executed and each set is given instance id which is based on scale-up value i.e. ID will be 3 for all SDB-3, CPE-3 and DPE-3 and id will be 4 for all SDB-4, CPE-4 and DPE-4. Validate deploy script is also executed after executing deploy script on each instance to check if running of deploy script was successful or not.

In the end VCM services are started on all the instances created during the scale up process.

## scale\_down.py

This script first prompts the user if it wants to scale down and if the user enters ‘yes’, this script scales-down the VCM instances to a unit value i.e. only the highest most instances based on the scale-up value will be scaled-down.

E.g. if the highest value of scaled-up instances is 3, script will terminate the instance with names:

* VCM-SDB-3
* VCM-CPE-3
* VCM-DPE-3

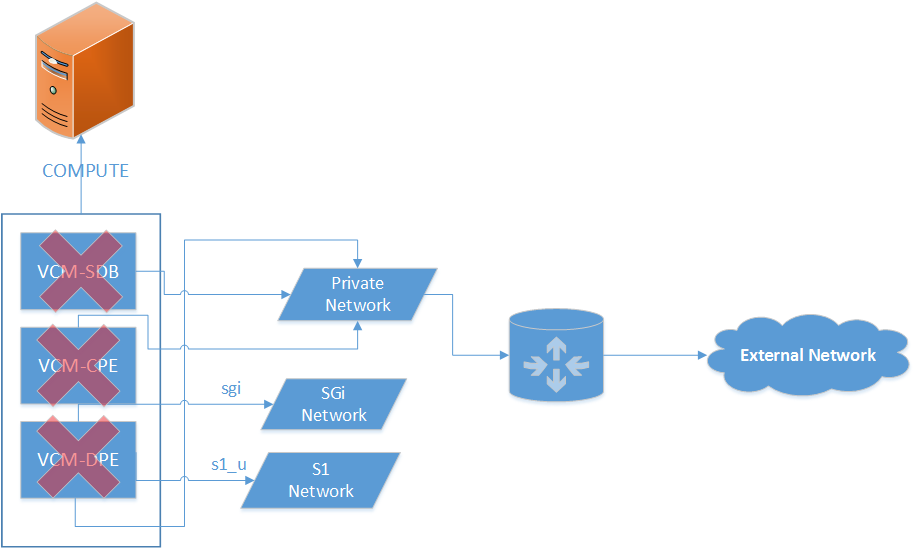


Figure 5: vEPC Scale-down process

If user wants to scale down all the scaled-up instances, it should keep on executing scale-down script until it says that no scale-up instances exist.

## vEPC\_termination.py

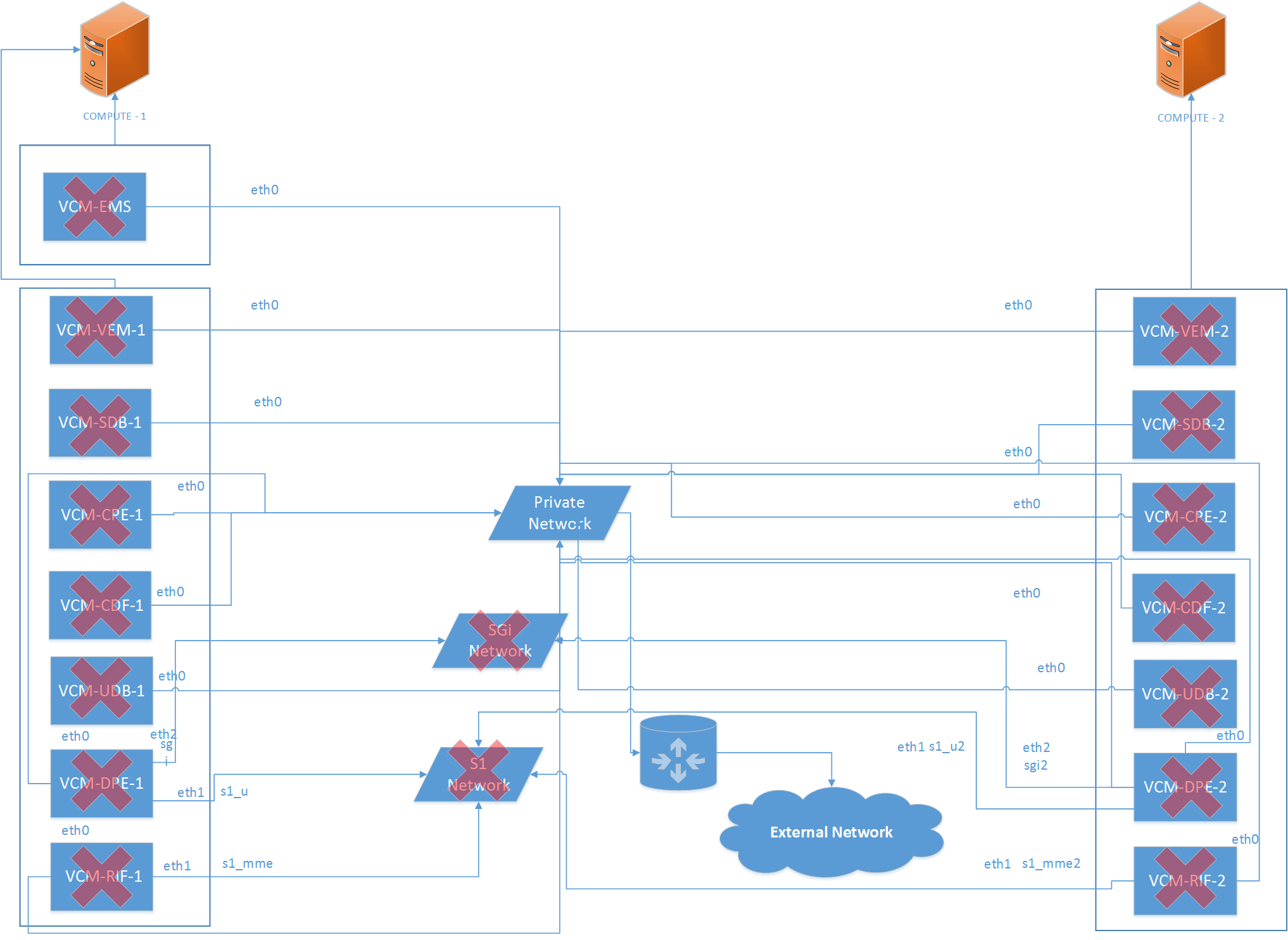


Figure : Complete vEPC Termination

Note: Before running the termination script, make sure that you have scaled-down all VCM instances otherwise script will exit showing the message of running the scale-down script until no vEPC scale-up instances exist.

When this script executes, it prompts the user if it’s sure to terminate vEPC and if user enters ‘yes’, first the script checks for any scaled-up instance and if any scale-up instance exist script terminates with a message of running scale-down script first.

If no scale-up instances exist, vEPC termination process gets started.

It terminates VCM-1 and VCM-2 components in parallel and after complete deletion, it asks the user if it wants to delete the EMS and based on the input performs the required action. Then the script deletes the S1 and SGi networks along with all their ports.

User is then prompted about the deletion of aggregate groups and opted to enter ‘yes/no’ and performs the action on the basis of input.

In the end, script prompts user if it also wants to delete the VCM and EMS images of glance and performs the action on the basis of input.