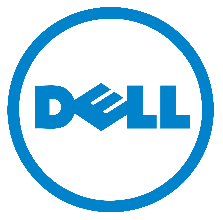
**BROCADE VEPC Automation scripts**

**Design Reference**



**Copyright © 2015 Dell Inc. All rights reserved.** This product is protected by U.S. and international copyright and intellectual property laws. Dell™ and the Dell logo are trademarks of Dell Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

**Table of Contents**

[**Revision History:** 3](#_Toc431809109)

[**Table of Acronyms** 4](#_Toc431809110)

[**1** **Purpose** 5](#_Toc431809111)

[**2** **Scope** 5](#_Toc431809112)

[**3** **Functionality of Scripts** 5](#_Toc431809113)

[**3.1** **vEPC\_Deploy.py** 6](#_Toc431809114)

[**3.2** **scale\_up.py** 11](#_Toc431809115)

[**3.3** **scale\_down.py** 13](#_Toc431809116)

[**3.4** **vEPC\_termination.py** 14](#_Toc431809117)

**Table of Figures**

[Figure 1: Complete VCM deployment along with network connectivity in High Availability 6](#_Toc431391573)

[Figure 2: VCM-1 Deployment Flow 7](#_Toc431391574)

[Figure 3: VCM2 Deployment and Network Connectivity Flow 9](#_Toc431391575)

[Figure 4: vEPC Scale-up process 11](#_Toc431391576)

[Figure 5: vEPC Scale-down process 13](#_Toc431391577)

[Figure 6: Complete vEPC Termination 14](#_Toc431391578)

# **Revision History:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Description** |
| 1.0 | Sep 14, 2015 | Abdul Rehman | Initial Version |
| 1.1 | Sep 22, 2015 | Abdul Rehman | Updated Version based on Jermaine’s comments |

# **Table of Acronyms**

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| CPE | Control Plane Engine |
| CDF | Charging Data Function |
| CLI | Command Line Interface |
| DPE | Data Path Engine |
| EPC | Evolved Packet Core |
| EMS | Element Management System |
| GUI | Graphical User Interface |
| IP | Internet Protocol |
| RAN | Radio Access Network |
| RIF | RAN Interface |
| SDB | Session Database |
| SSH | Secure Shell |
| UDB | User Database |
| URL | Uniform Resource Locator |
| VEPC | Virtual Evolved Packet Core |
| VCM | Virtual Core for Mobile, a Product by Brocade. |
| VEM | VCM Element Management |

# **Purpose**

This document explains the working of Brocade vEPC automation scripts in detail. It describes the process followed in deploying Brocade VCM using scripts i.e. number of instances to be deployed, in which order they will be deployed, which compute node is used for active VCM component and standby VCM component, type of configurations to be done and other related information.

# **Scope**

The document is intended for users who need to understand the detailed process followed to deploy Brocade vEPC using automation scripts. It doesn’t explain the working of Open stack APIs rather it explains the steps followed to deploy vEPC.

# **Functionality of Scripts**

Each script generate logs during execution. To view the logs, go to 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**

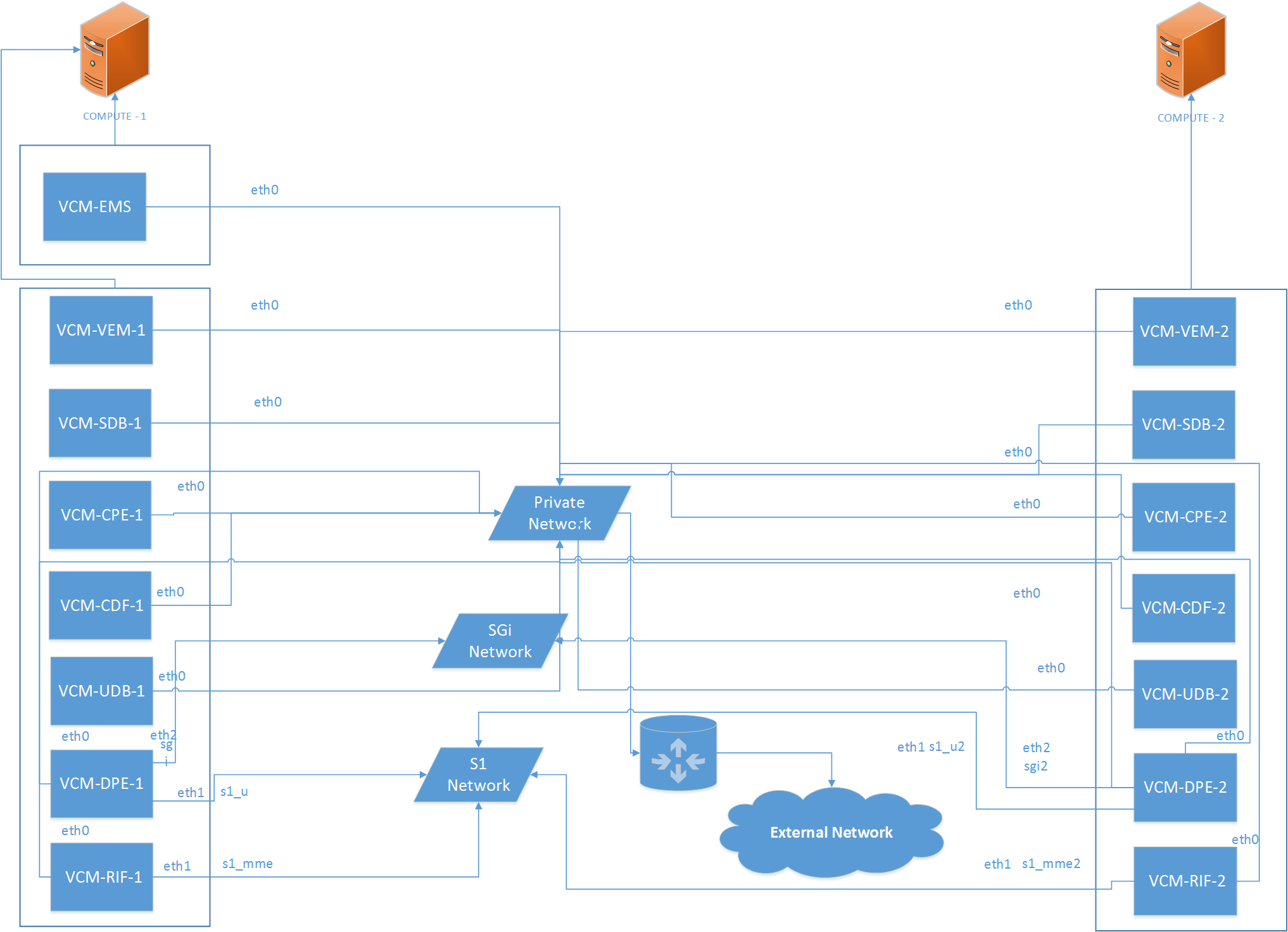


Figure 1: Complete VCM deployment along with network connectivity in High Availability

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

Before running the scripts you need to follow the steps mentioned in “vEPC Automation Scripts User Guide”. The guide shows how to edit *creds.txt file*. When this script runs, it gets the required parameters from c*reds.txt* file located in vEPC directory. The script then adds them to the *configurations.json* file which is used for all configurations including authentication for Openstack APIs.

Script then checks for the required resources on compute nodes for initial deployment. If the resources are not available they are stored in activity log file *vEPC\_deploy\_date\_time.log.* The script shows a warning message to the user if he still wants to continue. Based on the input script continues or terminates.

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

**VCM-1 deployment:**

If vEPC components are not present, script creates Availability Zones with name Compute 1 and Compute 2 and assign compute node 1 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.

The script then starts deploying VCM-1 components on Compute 1, i.e. it creates seven instances with different ports and networks. Instances are created in the following order and names:

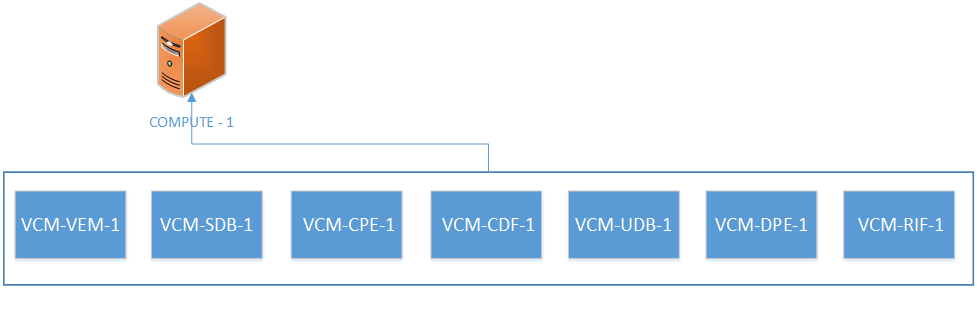


Figure 2: VCM-1 Deployment Flow

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 five 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 are created, it tries to ping all instances to check whether they boot up or not. If any instance fails to boot up, script waits for five seconds and again tries to check until it boots up.

The script terminates after two minutes if the instance takes too long to boot up as there may be a problem with the networking of Openstack or its deployment because of which instance(s) are not accessible via floating IPs.

When all instances boot up, it uses SSH to access each instance and changes the hostnames of all instances. After changing the hostnames, all seven instances are rebooted. The script then waits for ten seconds to allow instances to boot up before running the deployment script.

Now script again accesses the instances via SSH and runs the deployment script with instance ID 1. Some additional required configurations are also made For VEM, UDB and RIF. After that *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 initiated and validated using *validate deploy* script. If there isn’t any error, script produces the VCM services that are running on VEM-1. When services successfully run 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 initiated 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 VCM-1 components are created and services are initiated on all instances, VCM-2 instances are deployed on Compute 2.

**VCM-2 Deployment:**

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

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

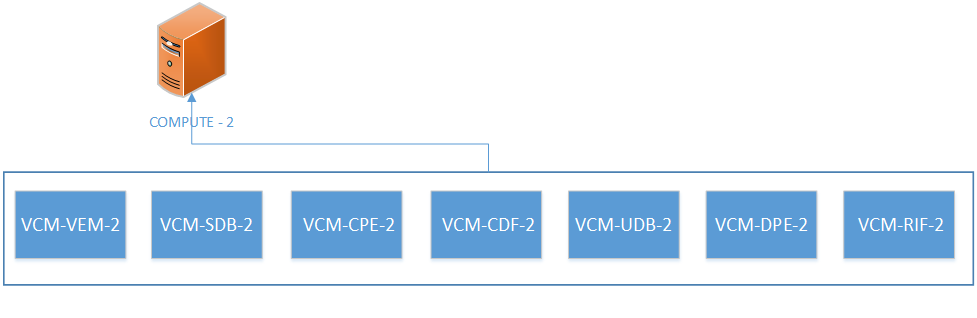


Figure 3: VCM2 Deployment and Network Connectivity Flow

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 are connected to the private network of Openstack name of which is provided in creds.txt file.

After instances are created, it tries to ping all instances to check whether they boot up or not. If any instance fails to boot up, script waits for five seconds and again tries to check until it boots up.

The script terminates after two minutes if the instance takes too long to boot up as there may be a problem with the networking of Openstack or its deployment because of which instance(s) are not accessible via floating IPs.

When all instances boot up, it uses SSH to access each instance and changes the hostnames of all instances. After changing the hostnames, all seven instances are rebooted. The script then waits for ten seconds to allow instances to boot up before running the deployment 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.

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 example it is Compute 1 for first time zone, Compute 2 for the second 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 is 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.

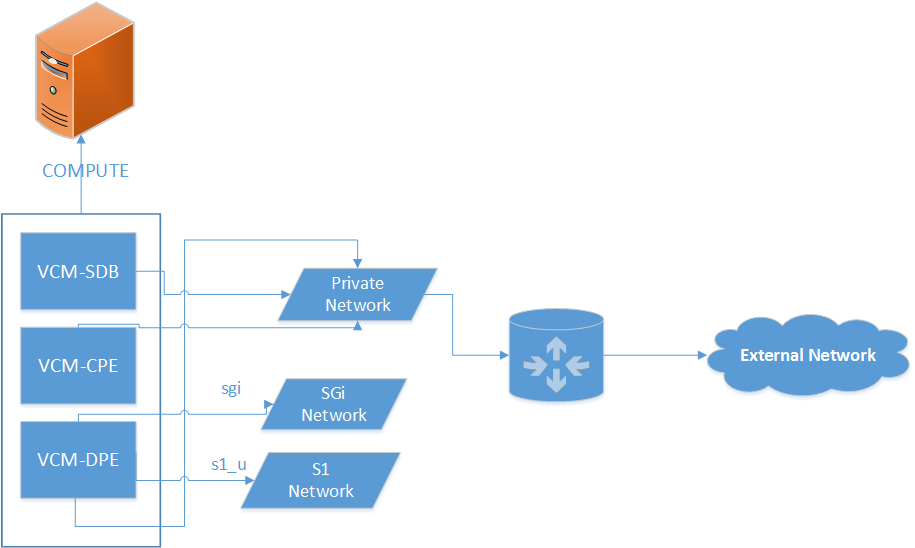


Figure 4: vEPC Scale-up process

After the instances are created, ping is performed on each instance to check if it boots or not. It waits for instance if it’s taking time. 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 is 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 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 he wants to scale down.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

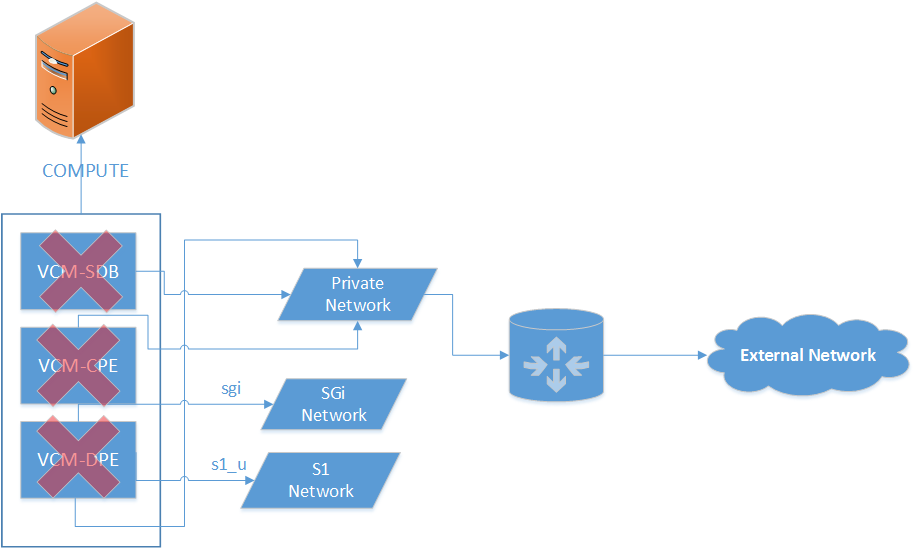


Figure 5: vEPC Scale-down process

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**

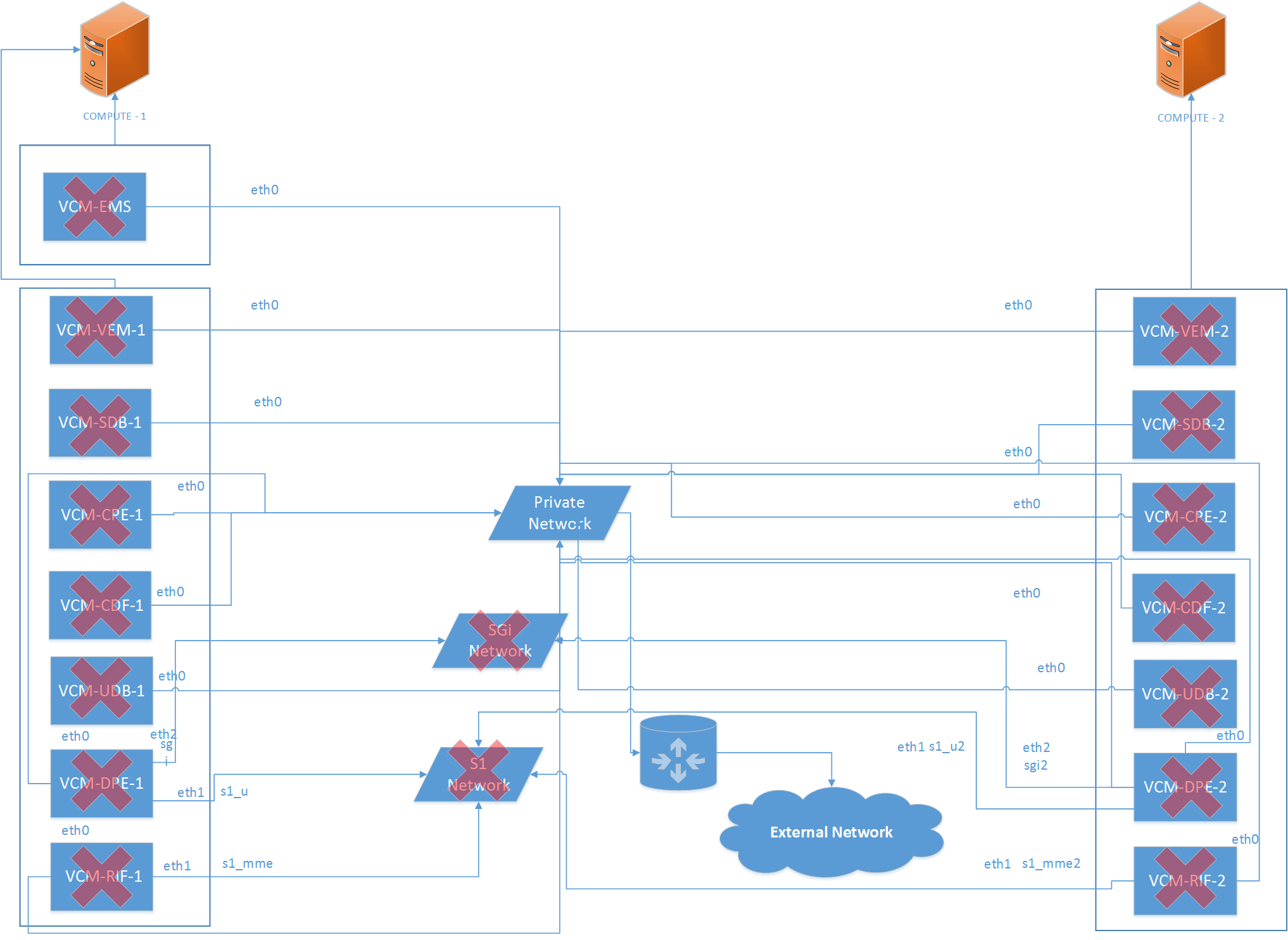


Figure 6: Complete vEPC Termination

***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 the terminate script executes, it prompts the user if he is sure to terminate vEPC.If user enters ‘yes’, first the script checks for any scale-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. Selecting yes will revert back to the default aggregate group i.e. “nova” and all compute nodes will be that group.

In the end, script prompts the user if it also wants to delete the glance images of VCM and EMS and performs the action on the basis of input. Creating glance images takes 4-6 minutes, so if you have plan to deploy again in the future and want to save your time then select “No” or if you don’t have any specific plan then select “Yes”.