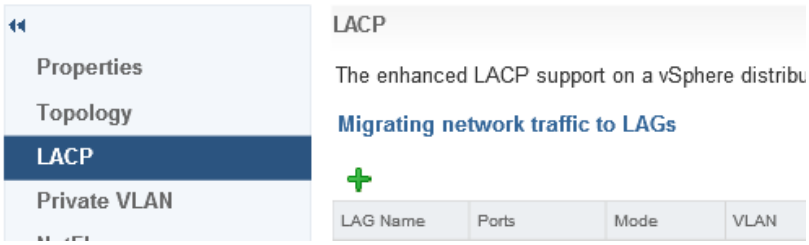
Creating Distributed switches with LACP/LAG

Whoever is working with LACP on switches and LAG on vSphere hosts must have noticed that the New-VDSwitch cmdlet doesn't give you the option to create the LAG on your VDSwitch. There is also no new-Lag or set-LagOnVdswitch cmdlet. If you have a small environment it is not a big deal to use the WebClient for creating the LAG and migrate to it.



But when you have to build large environments with multiple switches or if you have to rebuild your environment multiple times for test purposes it is a burden it cannot be automated using PowerShell.

That's why I was happy to find (thanks to a colleague) the website of LagerHaus (https://www.lagerhaus128.ch/?p=982). This man has made a custom PowerShell module to create new switches with the option to create a LAG. It did almost the right job for me but I had to add the option to use the right load balancing method. As you might know, it is best to set it the same on the both (physical switch and distributed switch) sides. Note that the Load balancing setting on the LAG will override the load balancing method set in de PortGroups.

This is what I created of the LagerHaus custom PowerShell module:

```
#Requires -Version 4.0

<#
.SYNOPSIS
    Quickly creates my default vSphere distributed switch
.DESCRIPTION
    Quickly creates a vSphere distributed switch with a LACP group containing
    a configurable number of ports and, if needed, some port groups
.PARAMETER VIServer
    Specifies the IP address or the DNS name of the vSphere server to which you
    want to connect. You can also specify a server by providing its IPv6
    address enclosed in square brackets, for example [fe80::56ff:feb0:74bd].
.PARAMETER Credential
    Specifies a PSCredential object that contains credentials for
    authenticating with the server.
.PARAMETER Datacenter
    Specifies the name of the datacenter you want to create the vDs within.
.PARAMETER VdsLinkDiscoveryProtocol
    Specifies the discovery protocol type of the vSphere distributed switch
    that you want to create. This parameter accepts CDP and LLDP values. If
    you do not set a value for this parameter, the CDP is
    used.
.PARAMETER VdsLinkDiscoveryProtocolOperation
    Specifies the link discovery protocol operation for the vSphere distributed
    switch that you want to create. This parameter accepts Advertise, Listen,
    Both, and Disabled values. If you do not set a value for this parameter,
    the Both is used.
.PARAMETER VdsNumUplinkPorts
    Specifies the number of uplink ports on the vSphere distributed switch that
    you want to create.
.PARAMETER VdsVersion
    Specifies the version of the vSphere distributed switch that you want to
    create. This parameter accepts 4.0, 4.1.0, 5.0.0, 5.1.0, 5.5.0, and 6.0.0
    values. You cannot specify a version that is incompatible with the version
    of the vCenter Server system you are connected to.
.PARAMETER VdsName
    Specifies a name for the new vSphere distributed switch that you want to
    create.
.PARAMETER LagName
    Specifies the name of the ling aggregation group you want to create.
.PARAMETER LagUplinkNum
    Specifies number of uplinks to be used.
.PARAMETER LagMode
    Specifies the mode for the ling aggregation group that you want to create.
    This parameter accepts active or passive values. If you do not set a value
    for this parameter, active is used.
.NOTES
    Author: kurator | lagerhaus128.ch
    Revision 0:
        Initial revision
Author: K. Zwerver
    Revision 1:
        Added support of load balancing method
.EXAMPLE
    Create-VDSwitch -VIServer vc01.fexlab.local -Credential (Get-Credential) -Datacenter
'datacenter01' -VdsLinkDiscoveryProtocol CDP -VdsLinkDiscoveryProtocolOperation Both -
VdsNumUplinkPorts 2 -VdsVersion 6.0.0 -VdsName 'vDs3' -LagName 'lag0' -LagUplinkNum 4 -
LagMode active -PortGroups
@(@{'Suffix'='Mgmt';'NumPorts'=4;'VlanId'=702},@{'Suffix'='ActiveDirectory';'NumPorts'=4
;'VlanId'=814})
#>

#continue on next page
```

```
function Create-VDSwitch {
    param(
        [Parameter(Mandatory=$true)][string]$VIServer,

[Parameter(Mandatory=$true)][System.Management.Automation.PSCredential]$Credentia
l,
        [Parameter(Mandatory=$true)][string]$Datacenter,

[Parameter(Mandatory=$false)][ValidateSet('CDP','LLDP','Unknown')][string]$VdsLin
kDiscoveryProtocol = 'CDP',

[Parameter(Mandatory=$false)][ValidateSet('Advertise','Both','Disabled','Listen',
'Unknown')][string]$VdsLinkDiscoveryProtocolOperation = 'Both',
        [Parameter(Mandatory=$false)][int]$VdsNumUplinkPorts = 2,

[Parameter(Mandatory=$false)][ValidateSet('4.0','4.1.0','5.0.0','5.1.0','5.5.0','
6.0.0','6.5.0')][string]$VdsVersion = '6.0.0',
        [Parameter(Mandatory=$false)][string]$VdsName = 'vDSwitch0',
        [Parameter(Mandatory=$false)][string]$LagName = 'lag0',
        [Parameter(Mandatory=$false)][int]$LagUplinkNum = 4,

[Parameter(Mandatory=$false)][ValidateSet('active','passive')][string]$LagMode =
'active',

       [Parameter(Mandatory=$false)][ValidateSet('srcDestMac','srcDestIP','srcPor
tId','srcTcpUdpPort','srcDestIpTcpUdpPort','srcDestIpVlan','destIp','destIpTcpUdp
Port','destIpVlan','destIpTcpUdpPortVlan','srcdestIpTcpUdpPortVlan','srcdestTcpUd
pPort','destMac','srcMac','destTcpUdpPort','srcIP','srcIpTcpUdpPort','srcIPVlan',
'srcIpTcpUdpPortVlan')][string]$LagLoadBal = 'srcDestMac',
        [Parameter(Mandatory=$false)][array]$PortGroups =
@(@{'Suffix'='Mgmt';'NumPorts'=4;'VlanId'=702})
    )
    Try{
        Import-Module VMware.VimAutomation.Core
        Import-Module VMware.VimAutomation.Vds
        $VIServerObject = Connect-VIServer -Server $VIServer -Credential
$Credential -NotDefault
        $VDSwitch = New-VDSwitch -LinkDiscoveryProtocol $VdsLinkDiscoveryProtocol
-LinkDiscoveryProtocolOperation $VdsLinkDiscoveryProtocolOperation -
NumUplinkPorts $VdsNumUplinkPorts -Version $VdsVersion -Name $VdsName -Server
$VIServerObject -Location (Get-Datacenter -Name $Datacenter -Server
$VIServerObject)
        $VDSwitch.ExtensionData.EnableNetworkResourceManagement($true)
        $VDSwitch.ExtensionData.UpdateViewData()
        $VDSwitch.ExtensionData.ReconfigureDvs((New-Object
VMware.Vim.VMwareDVSConfigSpec -Property @{
            'ConfigVersion'=($VDSwitch.ExtensionData.Config.ConfigVersion);
            'NetworkResourceControlVersion' = 'version3';
            'LacpApiVersion' ='multipleLag';
        }))
        $VDSwitch.ExtensionData.UpdateDVSLacpGroupConfig((New-Object
VMware.Vim.VMwareDvsLacpGroupSpec -Property @{
            'LacpGroupConfig' = New-Object VMware.Vim.VMwareDvsLacpGroupConfig -
Property @{
                'Name' = $LagName;
                'UplinkNum' = $LagUplinkNum;
                'Mode' = $LagMode;
                            'LoadbalanceAlgorithm'=$LagLoadBal;
            };
            'Operation' = 'Add'
        }))
         Remove-Variable VDSwitch,VIServerObject
    }Catch{
        Write-Error $_.Exception.Message
    }
}
Export-ModuleMember -Function Create-VDSwitch
```

In our switch create script I added the following lines:

```
# Import modules
Import-Module .\Create-VDSwitch.psm1
```

And to create the switch:

```
Create-VDSwitch -VIServer $vcenter -Credential $credential -Datacenter $datacenter
-VdsLinkDiscoveryProtocol LLDP -VdsLinkDiscoveryProtocolOperation Both -
VdsNumUplinkPorts 2 -VdsName $switch -LagName LAG-$switch -LagUplinkNum 2 -LagMode
passive -LagLoadBal $loadbalanceAlgoritme
```

And to create the PortGroup (for example a port-group for vMotion):

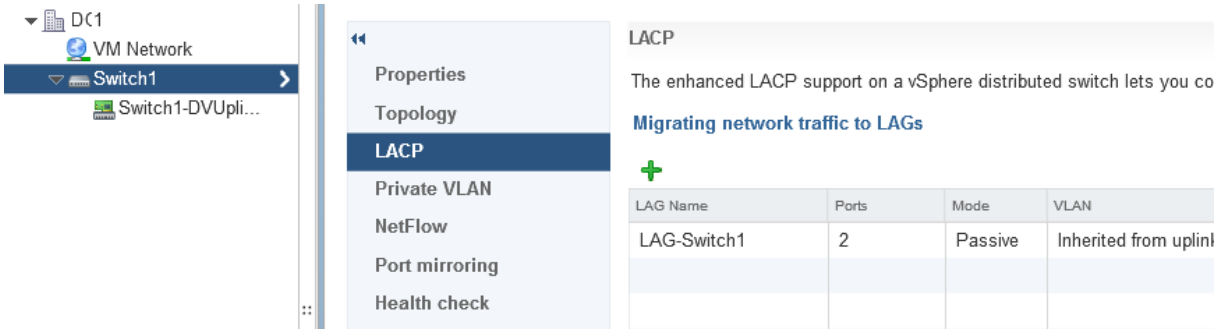
```
New-VDPortgroup -VDSwitch $switch -Name vMotion-$clustername -VlanId $vmotionvlan
|Get-VDUplinkTeamingPolicy | set-VDUplinkTeamingPolicy -ActiveUplinkPort LAG-
$switch1 -UnusedUplinkPort  @('dvUplink1','dvUplink2')
```

Lagerhaus made the variables $vcenter and $cred mandatory. You can of course edit this but I left it default here so this one line of code will work (after filling in the variables).

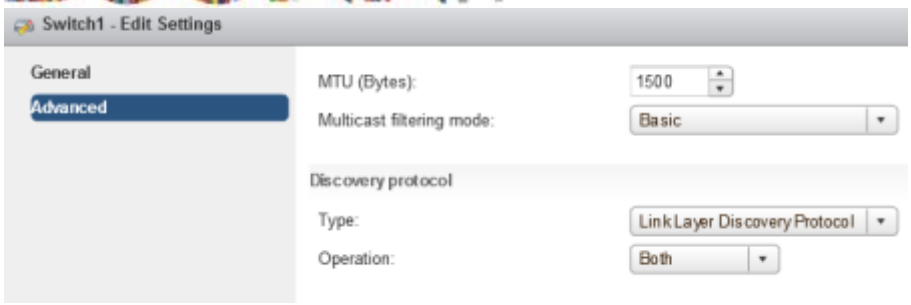
It works pretty straightforward. If you fill in the variables you will get something like this:

```
$cred = Get-Credential -Message "vCenter username en password"

Create-VDSwitch -VIServer vcenter.domain.com -Credential $cred -Datacenter DC1 -
VdsLinkDiscoveryProtocol LLDP -VdsLinkDiscoveryProtocolOperation Both -
VdsNumUplinkPorts 2 -VdsName Switch1 -LagName LAG-Switch1 -LagUplinkNum 2 -LagMode
passive -LagLoadBal srcDestMac

New-VDPortgroup -VDSwitch Switch1 -Name vMotion-TestCluster -VlanId 60 | Get-
VDUplinkTeamingPolicy | set-VDUplinkTeamingPolicy -ActiveUplinkPort LAG- Switch1 -
UnusedUplinkPort  @('dvUplink1','dvUplink2')
```

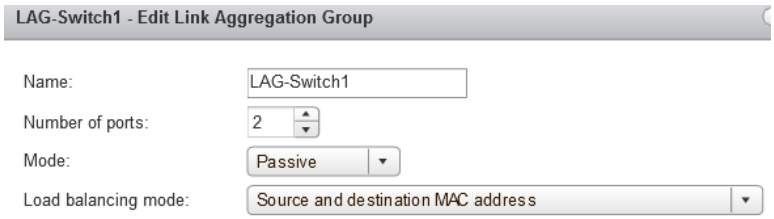
This will create a Distributed switch with the name Switch1 in DC1 with 2 uplinks and a LAG with the name LAG_Switch1 with 2 uplinks:



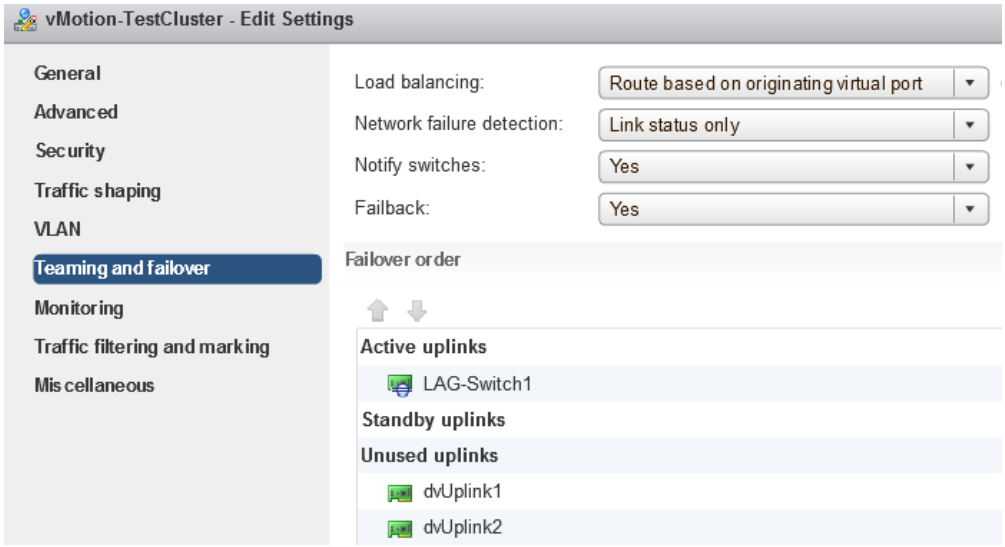
with LLDP as discovery protocol, working in both directions:



The LAG does not actively negotiate (you should set the switch to active) the port-channel and the load balancing method is Source and destination MAC address:



And the PortGroup with the LAG as the only active uplink:



Migrating to LAG with PowerShell cmdlets

After creating distributed switches with LAGs, you might want to start using them. First you have to add your hosts to your distributed switches.

Default a newly added hosts has its network connections on a standard switch, called vSwitch0. Before migrating to the LAGs, the hosts need to be connected with the right interface to the distributed switches. To do this in PowerShell you can use something like this:

```
Add-VDSwitchVMHost -VDSwitch $switch -VMHost $host
$vmnic0 = Get-vmhostnetworkadapter -VMHost $host -Physical -Name vmnic0
$vmnic2 = Get-vmhostnetworkadapter -VMHost $host -Physical -Name vmnic2
Add-VDSwitchPhysicalNetworkAdapter -VMHostPhysicalNic $vmnic0 -DistributedSwitch $switch
-Confirm:$false
Add-VDSwitchPhysicalNetworkAdapter -VMHostPhysicalNic $vmnic2 -DistributedSwitch $switch
-VirtualNicPortgroup $portgroup -VMHostVirtualNic $vmk0 -Confirm:$false
```

The vSphere WebClient provides a nice migration to LAG wizard. But using the wizard again and again in large environments costs a lot of time. So I created a script that does the trick. I found a blogpost which I used as a starting point: https://fdo-workspace.blogspot.nl/2017/01/vsphere-v55-lacp-configuration-using.html This script migrates only 1 switch and 1 host at a time and 1 switch uplink. I made modifications so that the script works for multiple switches and multiple hosts based on an input file. I use the input file for all the configure scripts so I do not have to fill in the hosts and IP's every time. The script finds the vmnics used on the switches itself and migrates them to the LAG.

Note: If you have set the LAG on your management port-group config to the only active uplink, the host will be disconnected if you just add your vmnics to the management PortGroup when adding the host to the distributed switch. For the other port-groups it is not a problem if they are not used yet. For the management port-group it is better to set de port-group in migration mode: dvUplinks as active uplinks and the LAG as Standby uplink, until you're finished migrating. If you read this too late, don't worry. When a host disconnects as the result of a change in the network configuration it will revert the change and connect again.

An example of an input file:

```
host,mgmtIP,vMotionIP,VSANIP
host1.domain.com,192.168.1.1,192.168.10.1,192.168.20.1
host2.domain.com,192.168.1.2,192.168.10.2,192.168.20.2
```

The script migrates the used vmnics to the LAG. It finally sets the LAG as active and dvUplink1 and 2 as unused for the management PortGroup. When I created the remaining port groups I already set the LAG as the only active uplink. For the management PortGroup I did set the LAG as standby uplink and the dvUplinks as active so the hosts remain connected (see previous note). But after the migration it is not necessary anymore and has to be modified.

```
# Template for migrating to LAG (previously created) Script voor VMware ESXi 6.0
host(s)
#
# K. Zwerver, 28 juli 2017: first setup
#
# This script does some spliting of arguments. It depence on your naming convention
how you need to use it.

# CONFIGURATION VARIABLES #

# Input File
$fn = "C:\Scripts\input.txt" # Input file with host info

# vCenter and cluster
$vcenter = "vcenter.domain.com"
$clustername = "TestCluster1"
$clustershort = "TCLS1"

# Switches #
$vds1 = "$clustershort-Switch1"
$vds2 = "$clustershort-Switch2"
$vds3 = "$clustershort-Switch3"

# Prefetch for Uplinkportgroup leave "" if not used
$upg = "UPLG-"
$allvds = "$vds1","$vds2","$vds3"

# Managment portgroup
$MgmtPortGroup = "$clustername-mgmt"

# Login
$cred = Get-Credential -Message "vCenter Credentials user@domain"
Connect-VIServer -Server $vcenter -Credential $cred #Connect to vCenter

# Execute script for all distributed switches and hosts
foreach ($vds in $allvds){
        import-csv $fn | %{
                $switch = Get-VDSwitch -Name $vds
                $current = $_;
                $esxname = $current.host      # Value from the file under "host".
                $esx = get-vmhost -Name $esxname
                $shortesx= $esx.name.split('.')[0] #remove domain name

                # Get connected nics from switch and add the 'vmnic' part to
variable
                $temp = $esx.ExtensionData.config.network.ProxySwitch | where
{$_.dvsname -match $vds}
                $nics = $temp.pnic
                $nic1,$nic2 = $nics.split(' ')
                $vmnic1 = $nic1.split('-')[2]
                $vmnic2 = $nic2.split('-')[2]

                # Split switchname to the only part needed
                $SwitchShortName=$vds.Split('-')[1]

                # Get uplink ports
                $uplinkport0 =
$esx.extensiondata.Config.Network.ProxySwitch.hostlag.uplinkport | where {$_.value
-match $SwitchShortName+"-0"}
                $uplinkport1 =
$esx.extensiondata.Config.Network.ProxySwitch.hostlag.uplinkport | where {$_.value
-match $SwitchShortName+"-1"}

                # Get UplinkPortGroup
                $nameDvUplink= $upg+$clustername+"-"+$SwitchShortName
                $uplinkPortGroupKey = $(get-view -viewtype
DistributedVirtualPortgroup -property name,key -filter @{name=$nameDvUplink}).key

                # Build variables for vmnic - uplinkport nic binding
                $pnicspec0 = New-Object
VMware.Vim.DistributedVirtualSwitchHostMemberPnicSpec
                $pnicspec0.PnicDevice = $vmnic1
                $pnicspec0.uplinkPortKey = $uplinkport0.Key
                $pnicspec0.uplinkPortgroupKey = "$uplinkPortGroupKey"

                $pnicspec1 = New-Object
VMware.Vim.DistributedVirtualSwitchHostMemberPnicSpec
                $pnicspec1.PnicDevice = $vmnic2
                $pnicspec1.uplinkPortKey = $uplinkport1.Key
                $pnicspec1.uplinkPortgroupKey = "$uplinkPortGroupKey"
```

```
                # Build variable for host to edit
                $hostspec = New-Object
VMware.Vim.DistributedVirtualSwitchHostMemberConfigSpec
                $hostspec.Host = $esx.ExtensionData.Config.Host
                $hostspec.Operation = 'edit'
                $hostspec.Backing = New-Object
VMware.Vim.DistributedVirtualSwitchHostMemberPnicBacking
                $hostspec.Backing.PnicSpec = New-Object
VMware.Vim.DistributedVirtualSwitchHostMemberPnicSpec[] (2) #number of uplinks
                $hostspec.Backing.PnicSpec[0]= $pnicspec0
                $hostspec.Backing.PnicSpec[1]= $pnicspec1

                # Build variable with all the created values
                $dvspec = New-Object VMware.Vim.DVSConfigSpec
                $dvspec.ConfigVersion = $switch.ExtensionData.Config.ConfigVersion
                $dvspec.Host = $hostspec

                # Run ReconfigureDVS Task with all the values and give some feedback
                $task = $switch.ExtensionData.ReconfigureDvs_Task($dvspec)
                Write-Host "Modified switch "$vds" on host "$esx "task "$task

                # Wait 5 sec before continu because only one edit at a time is
allowed and the script is faster than the action is finished
                Start-Sleep -s 5

                # Clear variables
                Remove-Variable task,switch
                #}
        }
}

# Set DVlinks unused and LAG active on the management switch
Get-VDSwitch -Name $vds2 | Get-VDPortgroup -Name $MgmtPortGroup |Get-
VDUplinkTeamingPolicy | set-VDUplinkTeamingPolicy -ActiveUplinkPort LAG-$vds2 -
UnusedUplinkPort  @('dvUplink1','dvUplink2')

Disconnect-VIServer -Confirm:$false
```

When finished the LACP's need to be created on the switches. An example on a nexus switch:

```
interface port-channel10

  description vSphereHost1
  switchport mode trunk
  switchport trunk native vlan 28
  switchport trunk allowed vlan 28-40
  spanning-tree port type edge trunk
  vpc 10

int eth 1/1
channel-group 10 mode active
```

Some consideration when moving to LAG:

Create the LAG's first. When they are all set create the port-channels (with VPC's if you use Cisco Nexus switches for example) on the switches. The ports will go down up so it is disruptive. But the port-channel will be formed nicely. If you go the other way around, first creating the port-channels on your switches and then planning to create the LAG's on your vSphere hosts, the ports on the switches will be suspended (assuming again using VPC) because the VPC packets will not go through the hosts.

I assume the use of VPC (Virtual PortChannel, not virtual private cloud) because in modern network designs it is common to connect the EUD to multiple switches. VPC gives you the ability to create a port-channel on two switches that will form one port-channel.

Migration to LAG

If you want a non-disruptive migration to LAG you should use the migration wizard in the WebClient:

Or script the steps if you don't want to do this by hand. But also modify the port-channels on the switch during the progress. These are the steps to follow (there are plenty of other blogs on the internet on how to migrate so I will just give a brief description of the method I used and worked the best for me):

1. Create the LAG on the distributed switch without members (you can use a part of the script for that)
2. Create the Port-channel (with VPC) on the physical switches without members (create a port-channel for every vSphere host and every distributed switch)
3. Add the LAG to the standby adapters on the port groups config in de teaming and failover configuration (all port groups on the distributed switch)
4. Add 1 network card to the LAG on the vSphere hosts
5. On the physical switch add the ports connected to the nics you just added to the LAG, to the corresponding port-channels (see fig. 1)
6. Modify all the port-groups on the distributed switch and set the LAG as the primary adapter in the teaming and failover config. Set the DvUplinks unused.
7. Add the other network card to the LAG on the vSphere host
8. On the physical switch add the second ports connected to the nics you just added to the LAG, to the corresponding port-channels
9. You're finished

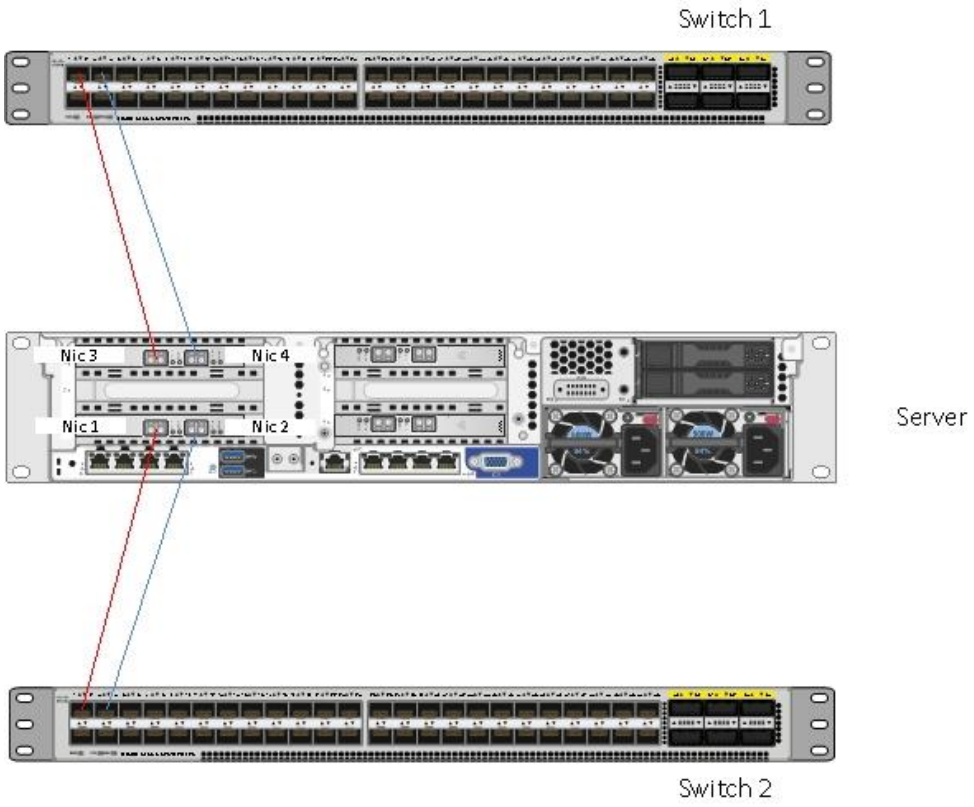


Fig. 1 Port-channel to host layout: The ports 1 on both switches are in the same port-channel (red lines) and the ports 3 on both switches are in a different port-channel (blue lines). Nics 1 and 3 on the host are on the same distributed switch and correspondent to the first port-channel (red). Nics 2 and 4 correspondents to the second port-channel (blue).