Overview and preparation

This section includes the following topics:

- Overview of InfoScale solutions in cloud environments
- InfoScale agents for monitoring resources in cloud environments
- InfoScale FSS feature for storage sharing in cloud environments
- InfoScale non-FSS feature for storage sharing in cloud environments
- About SmartIO in AWS environments
- Preparing for InfoScale installations in cloud environments
- Installing the AWS CLI package
- VPC security groups example

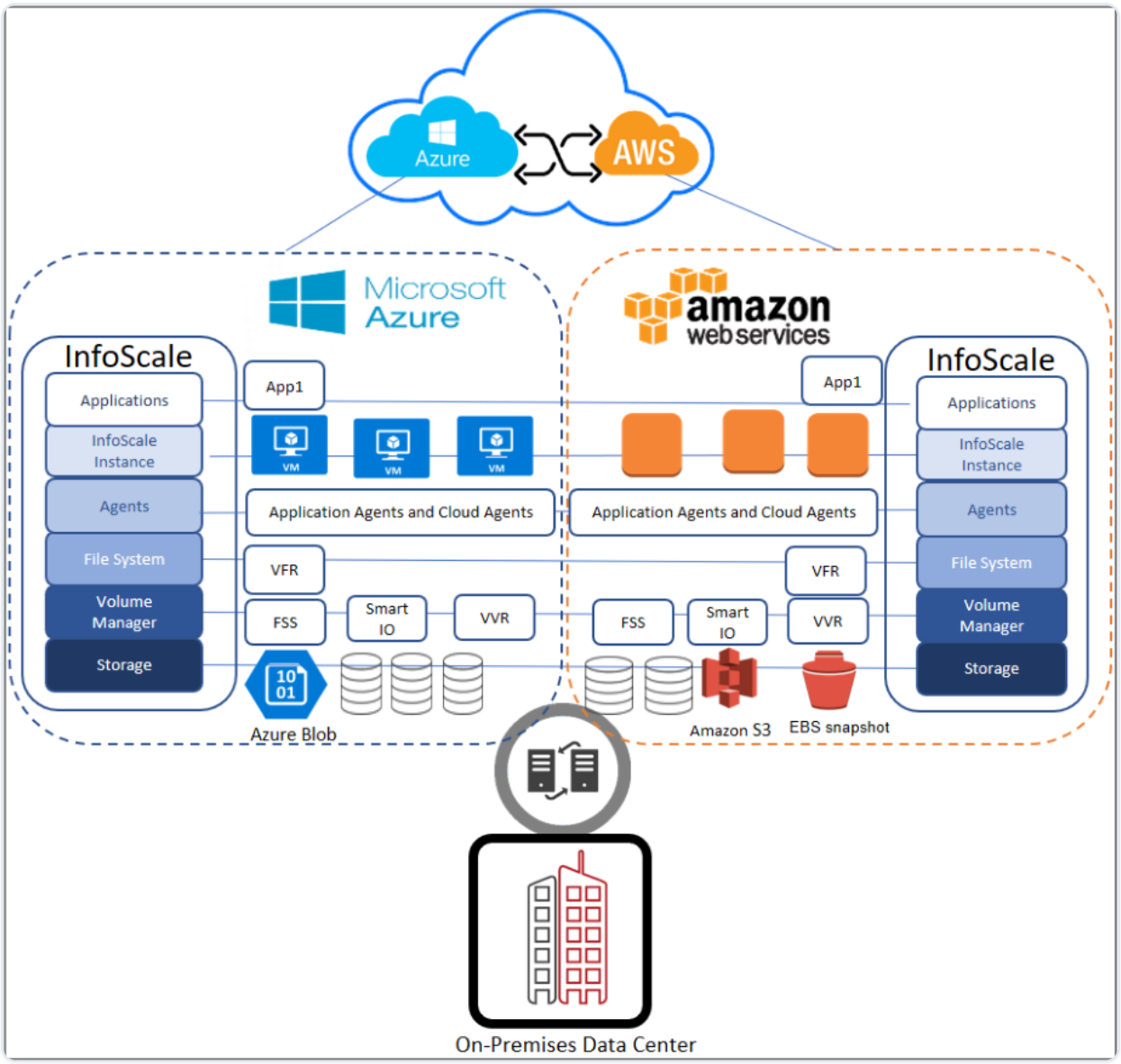
Overview of InfoScale solutions in cloud environments

The Acrtera InfoScale® product suite helps you make your applications highly available and resilient to disruptions. The applications may reside in your on-premises data centers or in public, private, or hybrid cloud environments. With InfoScale, you can combine cost-effective platform independence with consistent high application performance and availability. Acrtera supports InfoScale configurations in the Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) cloud environments.

Acrtera InfoScale Storage components let you manage various kinds of storage, which includes SAN, local flash, SSD, DAS, and cloud S3 targets. Acrtera InfoScale Availability agents provide monitoring and failover capabilities for the networking, storage, and replication resources that are associated with an application in the cloud.

Using these components you can configure an application for:

- Migration from on-premises to cloud or from cloud to cloud
- High availability or disaster recovery within a cloud
- Disaster recovery from on-premises to cloud or across clouds



Additionally, the InfoScale Operations Manager application migration wizard lets you automate the migration of your workload across OS platforms into the public cloud.

InfoScale agents for monitoring resources in cloud environments

InfoScale provides agents to identify the various resources that are used by applications in cloud environments. The agents monitor and manage the resource states to provide HA and DR for the configured applications.

Table: InfoScale agents for AWS

NAME	PURPOSE
AWSIP	Manages the private IP, elastic IP, and overlay IP resources in AWS.
AWSRoute53	Updates and monitors the host name
	to IP address mapping with the Amazon Route 53 DNS web service.
EBSVol	Attaches and detaches EBS volumes to and from Amazon EC2 instances, and monitors their state when attached to an instance.

Table: InfoScale agents for Azure

NAME	PURPOSE
AzureIP	Manages the private IP, public IP, and overlay IP resources in Azure.
AzureDNSZone	Provides DNS-based traffic routing and failover\:
	- Monitors and updates the host name to resource record mapping.
	- Performs the mapping for the Azure DNS domain during failover of nodes across subnets or regions.
AzureDisk	Attaches managed and unmanaged data disks
	to an Azure VM of the same or of a different resource group, monitors their state, and detaches them.

Table: InfoScale agents for GCP

NAME	PURPOSE
GoogleIP	Manages the private IP and the overlay IP resources in GCP.
GoogleDisk	Attaches persistent disks
	to an GCP VM instance of the same or of a different resource group, monitors their state, and detaches them.

For details on each of these agents, refer to the following documents:

- *Cluster Server Bundled Agents Reference Guide - Linux*
- *Cluster Server Bundled Agents Reference Guide - Windows*

InfoScale FSS feature for storage sharing in cloud environments

InfoScale supports flexible storage sharing (FSS) in cloud environments for a cluster that is located within the same region. The nodes in the cluster may be located within the same zone or across zones (Availability Zone in case of AWS and user-defined site in case of Azure). FSS leverages cloud block storage to provide shared storage capability.

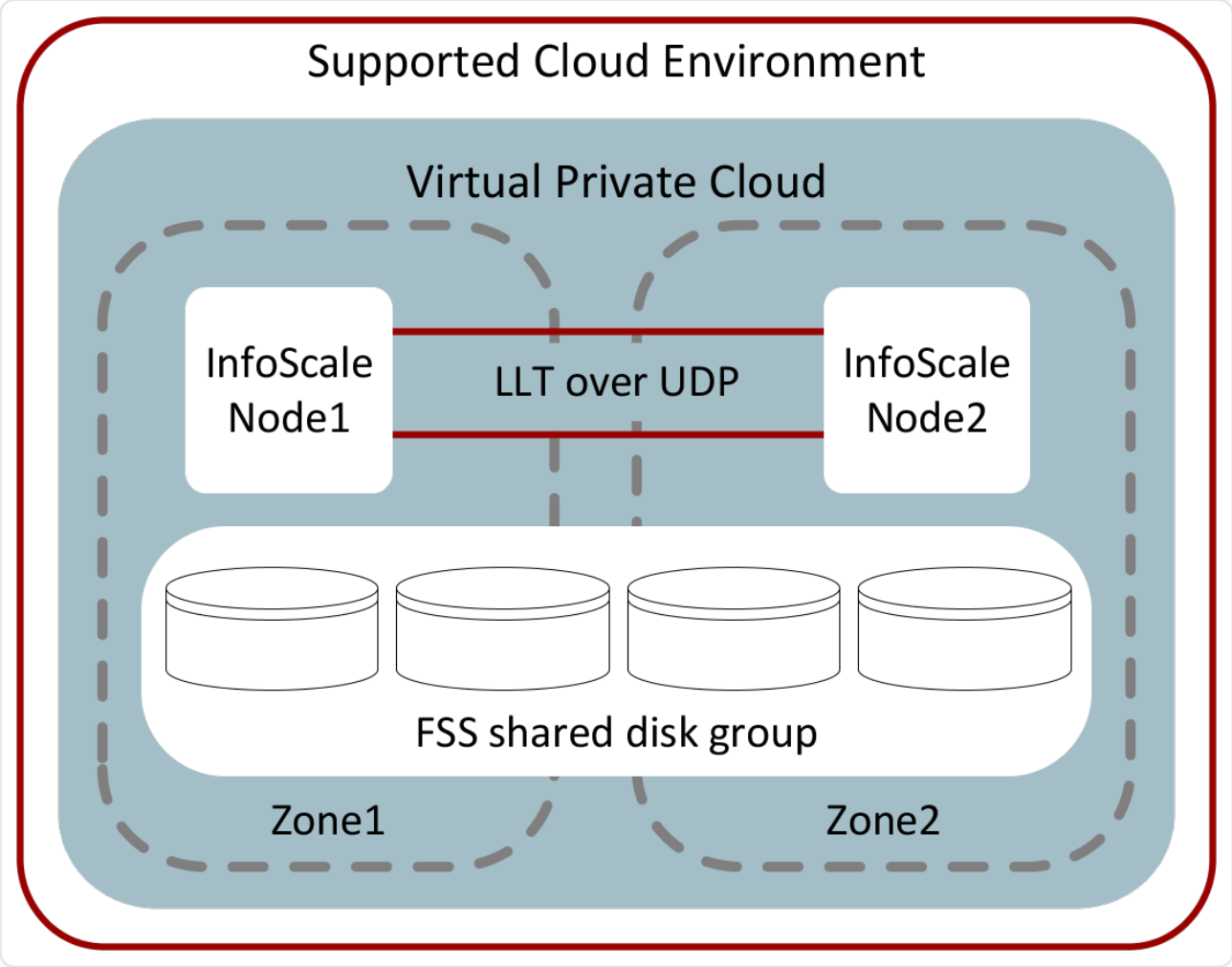
Storage devices that are under Volume Manager (VxVM) control are prefixed with the private IP address of the node. You can override the default behavior with the `vxdctl set hostprefix` command. For details, see the *Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

VxVM provides the ability to grow the existing storage by growing the LUN, which is termed as Dynamic LUN Expansion (DLE). You can leverage DLE to grow the storage in cloud environments. To use the DLE feature, you invoke the `vxdisk resize` command with the `length` option from the master node in an FSS configuration. Alternatively, you can use the Management Server console of InfoScale Operations Manager to resize a disk, which internally invokes this command. The command intelligently detects the remote disks and gets the required protocol executed to complete the LUN expansion; you do not need to specify any additional options. No explicit master switching is required. For details on growing the existing storage by growing the LUN, refer to the *Storage Foundation Administrator's Guide - Linux*.

Note: Do not resize an LVM disk or any other disk on which the OS is installed.

In cloud environments, FSS in campus cluster configurations can be used as a disaster recovery mechanism, across data centers within a single region. For example, in AWS, nodes within an AZ can be configured as one of the campus cluster site, while the nodes in another AZ can be configured as the second site. For details, see the *InfoScale Disaster Recovery Implementation Guide - Linux*.

Figure: Typical FSS configuration in a supported cloud environment



Note: (Azure only) By default, in addition to the storage disks that you have attached, every virtual machine that is provisioned contains a temporary resource disk. Do not use the temporary resource disk as a data disk. A temporary resource disk is an ephemeral storage that must not be used for persistent data. The disk may change after a machine is redeployed or restarted, and the data is lost. SeeAbout identifying a temporary resource disk - Linux. For details on how Azure uses a temporary disk, see the Microsoft Azure documentation. **Note:** (Azure only) In case your configuration uses Premium SSD v2 disks,

> remember that they are shared across all the VMs in a zone, by default. Such disks can be unintentionally attached to non-InfoScale-cluster VMs, without any error or warning. To restrict the disk access to InfoScale cluster nodes and thus prevent unauthorized use, set theDisk.MaxSharesproperty of the Azure disk to1. **Note:** (GCP only) When VCS is stopped and started on VM instances, or after a node restarts, the import and recovery operations on FSS diskgroups may take longer than expected. The master cannot import a diskgroup until all the nodes have joined the cluster. Some nodes may join their cluster with some delay. In that case, a diskgroup import operation takes longer than expected to succeed. Even if the master initially fails to import the diskgroup due to such a delay, the operation completes successfully later on retry.

> Considerations for LLT

In cloud environments, FSS is supported with LLT over reliable UDP configurations only. Acrtera recommends that you configure LLT over UDP for both, FSS and non-FSS clusters in the cloud.

Cloud-based networks are relatively slow and have high latency as compared to physical networks.

To achieve better LLT performance in such high-latency cloud networks, set the following tunable values before you start LLT or the LLT services:

- `set-flow window:10`
- `set-flow highwater:10000`
- `set-flow lowwater:8000`
- `set-flow rporthighwater:10000`
- `set-flow rportlowwater:8000`
- `set-flow ackval:5`
- `set-flow linkburst:32`

For details on the usage of these tunables, refer to the *Cluster Server Administrator's Guide - Linux*.

InfoScale non-FSS feature for storage sharing in cloud environments

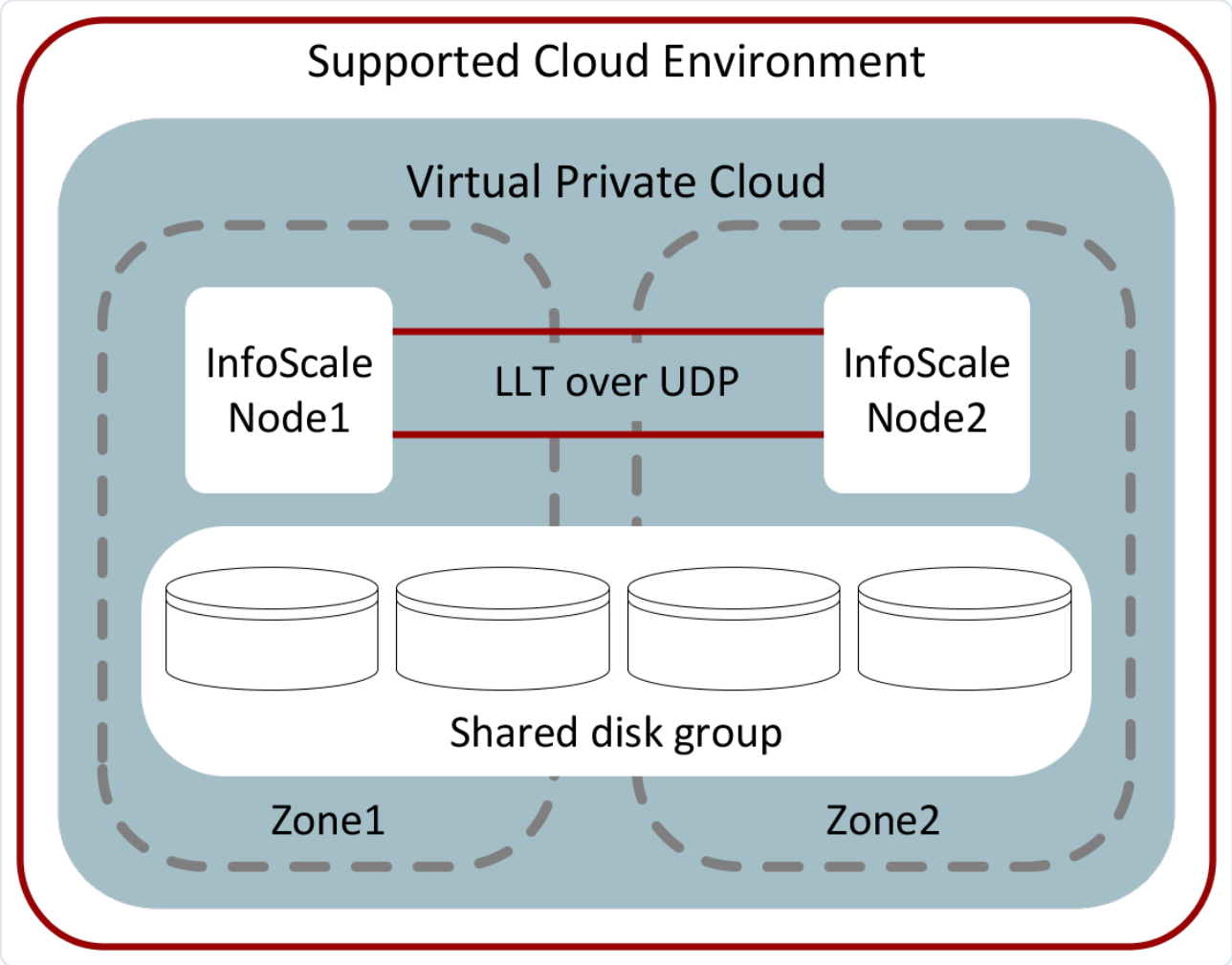
The nodes in the cluster may be located within the same zone or across zones (Availability Zone in case of AWS and user-defined site in case of Azure). Storage devices that are under VxVM control

are prefixed with the private IP address of the node. You can override the default behavior with the `vxdctl set hostpfirex` command. For details, see the *Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

Considerations for LLT

Acrtera recommends that you configure LLT over UDP for both, FSS and non-FSS clusters in the cloud. InfoScale use LLT only for messages. Performance tuning is not required for LLT configuration on non-FSS environment.

Figure: Typical non-FSS configuration in a supported cloud environment



Cloud-based networks are relatively slow and have high latency as compared to physical networks.

To achieve better LLT performance in high-latency cloud networks, set the following tunable values before you start LLT or the LLT services:

- `set-flow window:10`
- `set-flow highwater:10000`
- `set-flow lowwater:8000`

- `set-flow rporthighwater:10000`
- `set-flow rportlowwater:8000`
- `set-flow ackval:5`
- `set-flow linkburst:32`

Disable the LLT adaptive window in Azure and in GCP as follows:

- `/etc/sysconfig/llt LLT_ENABLE_AWINDOW=0`

The configuration that the following /etc/llttab file represents for Node 1 has links crossing IP routers. Notice that IP addresses are shown for each link on each peer node.

Availability zone do not support ARP(Address Resolution Protocol) protocol. So, need to disable LLT heartbeat over ARP and broadcast in the link command of the /etc/llttab file.

- `set-node Node1`
- `set-cluster 1`
- `link link1 udp - udp 50000 - 192.1.3.1 -`
- `link link2 udp - udp 50001 - 192.1.4.1 -`

set address of each link for all peer nodes in the cluster

format: set-addr node-id link tag-name address

Example:

- `set-addr 0 link1 192.1.1.1`
- `set-addr 0 link2 192.1.2.1`
- `set-addr 2 link1 192.1.5.2`
- `set-addr 2 link2 192.1.6.2`

disable LLT broadcasts

- `set-bcasthb 0`
- `set-arp 0`

The /etc/llttab file on Node 0 resembles:

- `set-node Node0`
- `set-cluster 1`

About SmartIO in AWS environments

In AWS environments, SmartIO can be configured on Amazon instance store SSDs for better performance. Instance Store SSDs are directly attached block device storage in the AWS cloud. As in an on-premises clustered environment, the SmartIO cache is local to each node in the cluster. For details on the SmartIO feature and its usage, see the *InfoScale SmartIO for Solid-State Drives Solutions Guide - Linux*.

Preparing for InfoScale installations in cloud environments

> **Note:** InfoScale installations in cloud environments are supported on the Microsoft Windows, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, CentOS, and Oracle Enterprise Linux platforms.

> Preparing for InfoScale product installations in cloud involve the following tasks:

1. Launch the instances on which you plan to install the InfoScale products.
2. Create the required network interfaces on all the instances and restart the network services.
3. Where required, add swap files on the instances.
4. Make sure that instances have public connectivity through public IP, elastic IP, the NAT gateway, or over VPN.

In case of Linux systems, this connectivity is also required so that any missing platform RPMs can be installed.

1. Copy the appropriate InfoScale installable files on one or all of the instances.
2. To allow remote installation of an InfoScale product:
    - On Linux systems, enable root login over SSH on all the instances and set a password for the root user.
    - On Windows systems, set the necessary administrative user and group permissions.
3. (AWS only) Ensure that the AWS CLI is installed on each InfoScale node.

See Installing the AWS CLI package.

1. Ensure that the security groups in your VPC are properly configured and all the UDP ports for the LLT high priority links are enabled in the VPC and the VM firewall.

For example, you must enable network ports 50000 to 50006 for 2 high priority links, 50000 to 50007 for 3 high priority links, and so on up to 8 high priority links. These examples are based on the default port number 50000. If the default port number in your environment are different, use the corresponding port range. You can find the default port number mentioned in `/etc/lltab`.

1. (GCP only) When you add multiple interfaces to a VM instance in the Google Cloud you need to set up the routing mechanism that is necessary for your platform.

    - On Linux systems, enable policy-based routing. For details, refer to the Google documentation at:

create-use-multiple-interfaces

- On Windows systems, add persistent routes. For example:

-	VM1	VM2
Subnet	10.208.8.0/22	10.209.4.0/22
Add persistent route	`route add 10.208.4.0 mask 255.255.252.0 10.208.8.1 -p`	`route add 10.208.8.0 mask 255.255.252.0 10.208.4.1 -p`

Ensure that you add persistent routes for every interface on every VM instance.

After the preparatory tasks are complete, run the appropriate installer for your platform on each compute instance. Follow the installer prompts to complete the installation. For details, refer to the *InfoScale Installation Guide - Linux* or *InfoScale Installation and Upgrade Guide - Windows*.

On AWS, you can also deploy InfoScale and create the required configurations using the CloudFormation Template (CFT) and the corresponding Amazon Machine Images (AMIs).

B07CYGD14V?qid=1542877554194&sr=0-3

More Information

VPC security groups example

Installing the AWS CLI package

To install the AWS CLI package

1. Enter the following commands sequentially:

```
$ curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
```

```
$ unzip awscli-bundle.zip
```

```
$ sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
```

1. Create the Identity and Access Management (IAM) role.

For details, refer to the AWS documentation about Identity and Access Management.

1. Configure the AWS CLI.

```
$ aws configure
```

1. Verify that the CLIs are working properly.

```
$ aws ec2 describe-instances
```

VPC security groups example

A security group acts as a virtual firewall for the instance to control inbound and outbound traffic. Therefore, you must make sure that in cluster configuration, ports are enabled in the security groups so that the VMs can communicate with each others.

Consider an example with two VMs that uses two subnets:

Subnet 1: 172.17.1.0/24

Subnet 2: 172.17.2.0/24

With this setup, the following table describes the inbound rules for a security group.

```
Inbound rules


 Type                  Protocol     Port range          Source
```

```
ALL UDP                 UDP         0-65535             172.17.1.0/24

All UDP                 UDP         0-65535             172.17.2.0/24
```

This table indicates that the inbound traffic is allowed between the instances that are assigned to the security group.

Configurations for Amazon Web Services - Linux

This section includes the following topics:

- Replication configurations in AWS - Linux
- HA and DR configurations in AWS - Linux

Replication configurations in AWS - Linux

The steps for setting up replication in AWS depends on where your primary data center and your secondary data center are located.

Refer to the topic that is appropriate for your setup:

- See Replication from on-premises to AWS - Linux.
- See Replication across AZs within an AWS region - Linux.
- See Replication across AWS regions - Linux.
- See Replication across multiple AWS AZs and regions (campus cluster) - Linux.

Replication from on-premises to AWS - Linux

In this scenario, data is replicated from an on-premises data center to a cloud data center.

Sample configuration

This sample configuration assumes a clustered environment that uses CVR for replication.

- The primary site is configured on premise.
- The secondary site is configured on the cloud.
- Private IP addresses used for replication in standalone environments OR

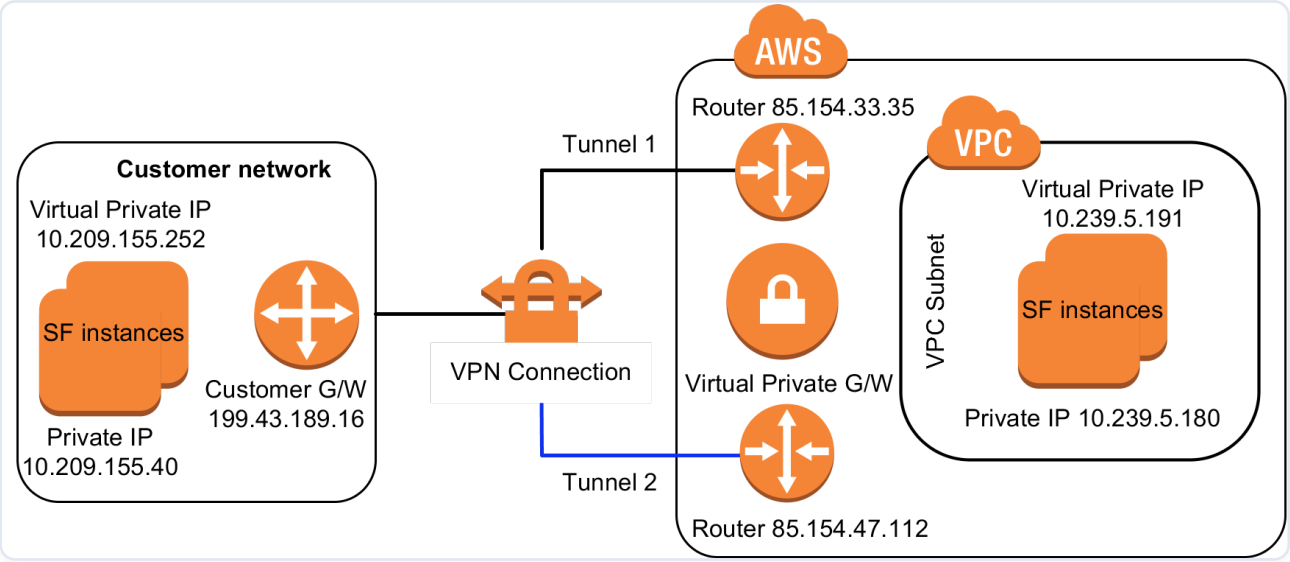
Virtual private IP addresses used for replication in clustered environments.

- A customer gateway on premise and a virtual private gateway on the cloud

- 2 VPN tunnels connecting the customer gateway and the virtual private gateway. This provides increased availability for the Amazon VPC service. If there's a device failure within AWS, the VPN connection automatically fails over to the second tunnel to provide uninterrupted access.
- 2 routers connecting the customer and cloud networks.

The following graphic illustrates the configuration.

Figure: Sample replication configuration from on-premises to AWS



Prerequisites

Ensure that the following requirements are met before you proceed with the configuration:

- Open ports for the communication between on premise and AWS cloud clusters in those subnets where the primary and secondary clusters are configured.

Make sure that in cluster configuration, ports are enabled in the security groups so that the VMs can communicate with each others.

- The virtual private IP addresses must be plumbed on both master nodes.
- The virtual private IP addresses must be configured within the subnet.

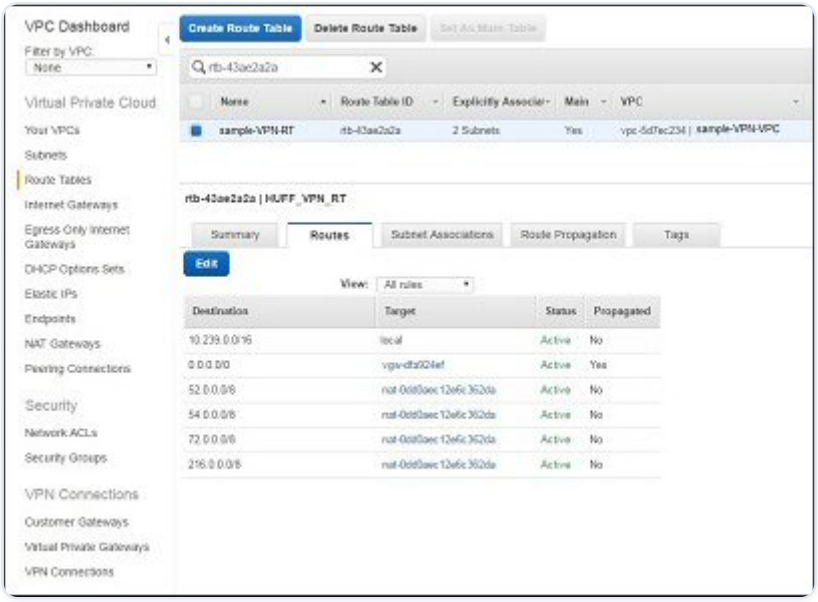
Setting up replication

The following procedure lists high-level tasks to configure replication from an on-premises data center to a cloud data center in a clustered environment that uses CVR.

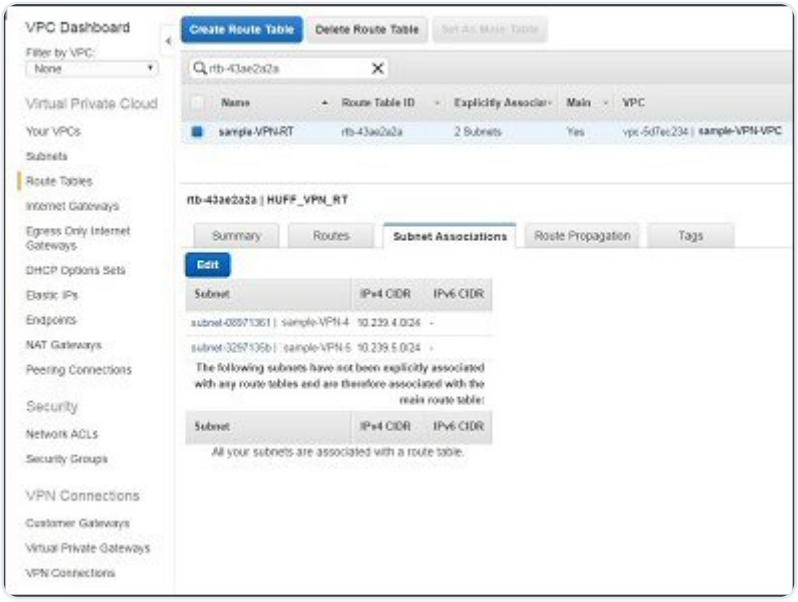
To set up replication from on-premises to AWS

1. Create a VPC with a valid CIDR block, for example 10.239.0.0/16.
2. Add a subnet in the Availability Zone.

3. Create the EC2 instance.
4. Configure the virtual private gateway and attach it to the VPC.
5. Configure the customer gateway.
6. Create route table entries.

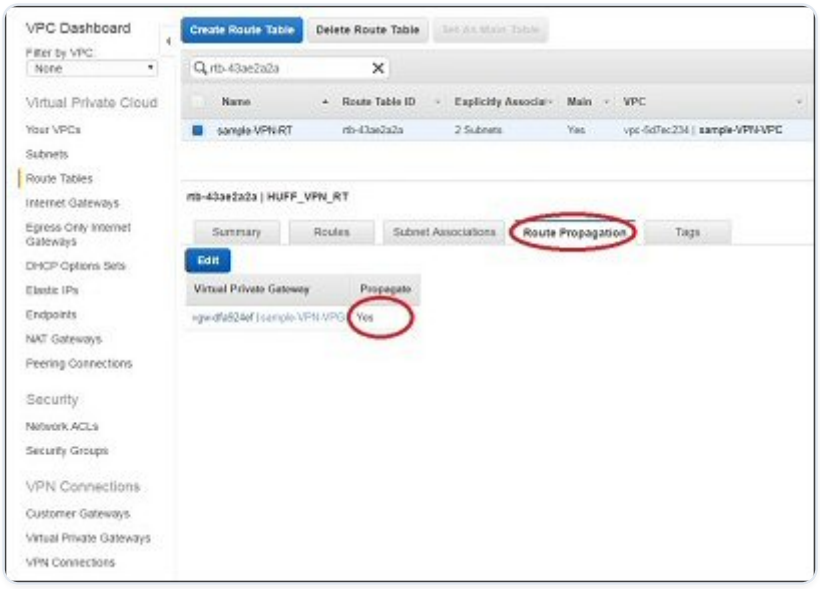


7. Associate the subnet with the route table.

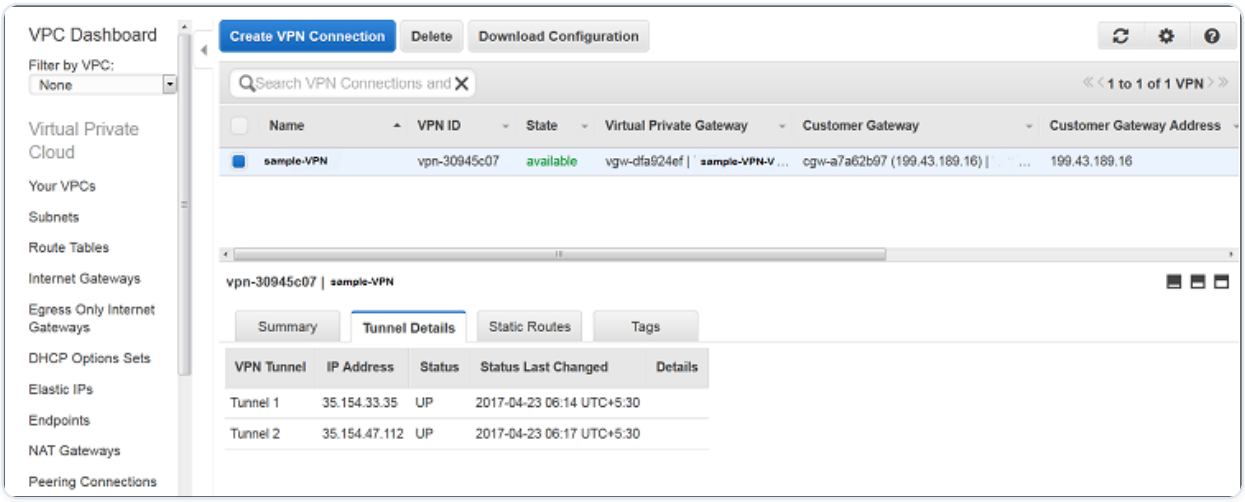


8. Enable route propagation to automatically propagate the routes to the table.

On the Route Propagation tab in the details pane, choose Edit, select the virtual private gateway that you created.



1. Create a VPN connection.
2. Download the VPN configuration file.
3. Create a VPN tunnel between the customer network and the cloud network.



4. Set up replication between the on-premise and cloud instances.

For instructions about setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the status of replication.

```
\# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

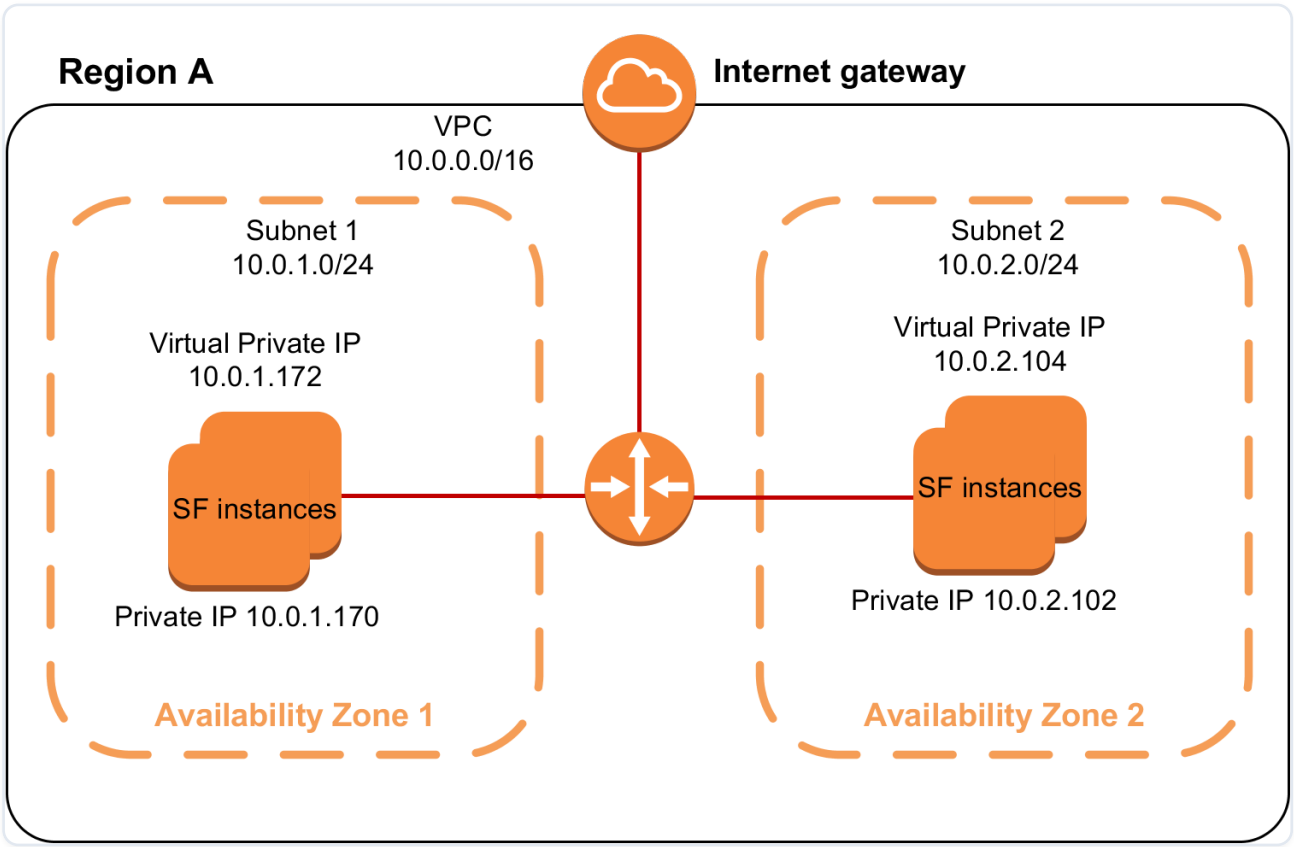
Replication across AZs within an AWS region - Linux

In this scenario, replication is set up across Availability Zones within the same region.

- A VPC with a valid CIDR block (for example, 10.0.0.0/16) that spans two Availability Zones (AZs) within the same region.
- The primary instance is configured in AZ1 and the secondary instance is configured in AZ2.
- InfoScale instances in the primary site and secondary site subnets.
- An Internet gateway to allow access to the Internet.
- Private IP addresses used for replication in standalone environments OR

Virtual private IP addresses used for replication in clustered environments.

Figure: Replication across AWS AZs



Setting up replication across AZs within the same region

Perform the steps in the following procedure to set up replication across AZs within the same region.

To set up replication in cloud environments

1. Create a VPC with a valid CIDR block, for example, 10.0.0.0/16.

2. Create the internet gateway and attach it to the VPC.
3. Modify the VPC route table such that the two instances across availability zones can communicate with each other using private IP addresses.
4. Create two subnets - one subnet for the primary site in AZ1 and the second subnet for the secondary site in AZ2 with valid CIDR range, for example 10.0.1.0/24 and 10.0.2.0/24 respectively.
5. Launch the EC2 instances in the primary and secondary subnets. Install InfoScale on the instances.

   > **Note:** For standalone environments, private IP addresses are used for replication. For clustered environments, virtual private IP addresses are used for replication.

6. Set up the AWSIP agent resource to manage the virtual private IP address (VPIP) configuration.

For detailed instructions, refer to the *Cluster Server Administrator's Guide - Linux*.

1. Verify connectivity between the virtual private IP addresses of the instances.

   ```
   \# ping PIP
   ```

   ```
   \# ping VPIP
   ```

2. Set up replication between the instances using private IP address or virtual private IP address.

For instructions on setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the status of replication.

   ```
   \# vradmin -g dg_name repstatus rvg_name
   ```

   Ensure that the status shows:

   ```
   Replication status: replicating (connected)
   ```

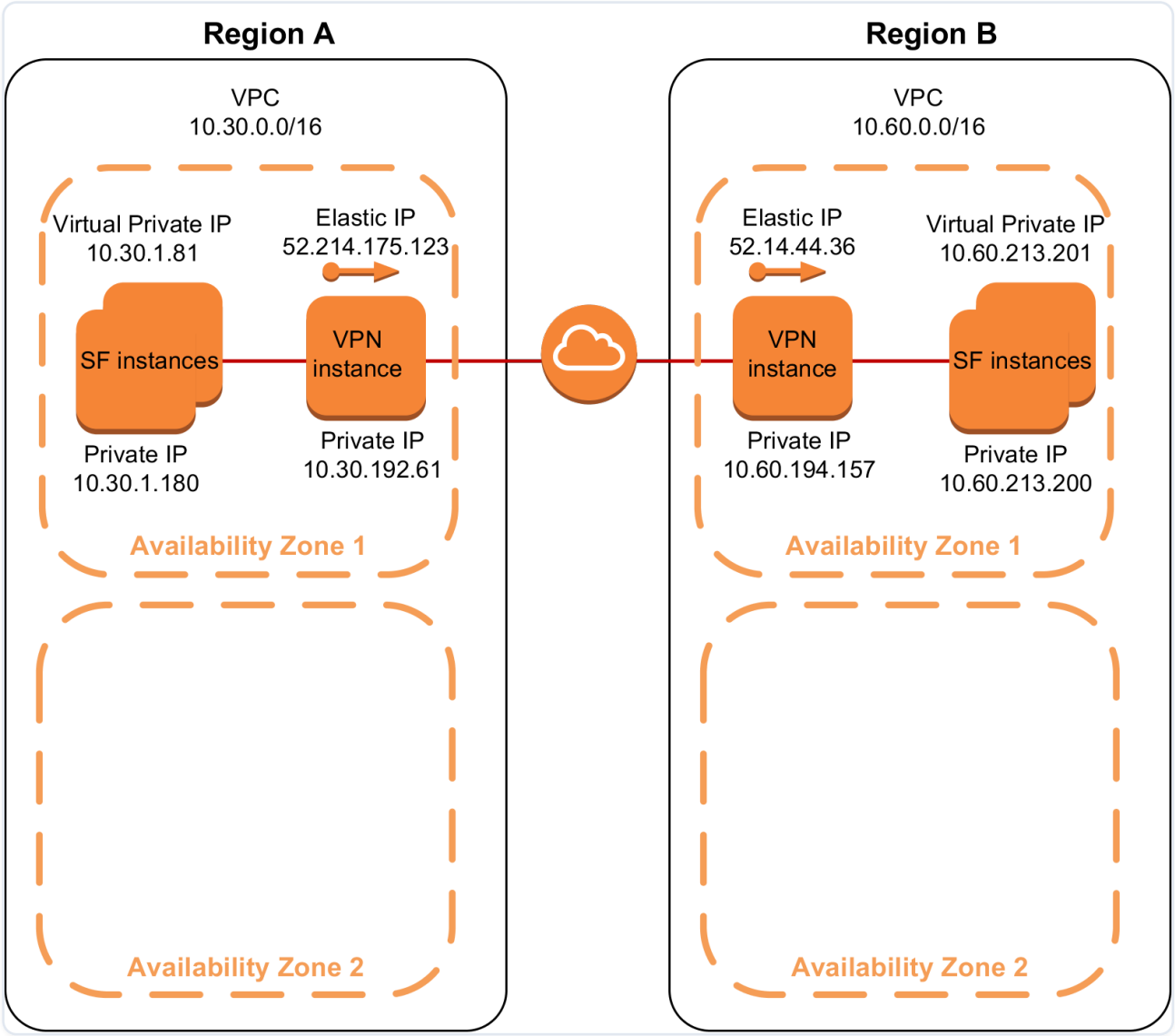
Replication across AWS regions - Linux

In this scenario, replication is set up across Availability Zones configured in different regions. The configuration uses software VPN `Openswan` to connect the VPCs across different regions.

- Two VPCs with valid CIDR blocks (for example, 10.30.0.0/16 and 10.60.0.0/16 respectively) that are located in two different regions.
- The primary instance belongs to AZ1 of region A and the secondary instance belongs to AZ1 of region B.
- InfoScale instances in each AZ.
- A VPN tunnel is established between two VPCs in two different regions using software VPN. Here, we have used OpenSwan software VPN. This is a secure IPSec tunnel.
- Elastic IP addresses (EIP) to connect the two VPN instances
- Private IP addresses used for replication in standalone environments OR

Virtual private IP addresses used for replication in clustered environments.

Figure: Replication across AWS regions



Setting up replication across regions

Perform the steps in the following procedure to set up replication across regions.

To set up replication across regions

1. Create two VPCs with valid CIDR blocks in different regions, for example, 10.30.0.0/16 and 10.60.0.0/16 respectively.
2. Create the primary site EC2 instance.
3. Create the primary site VPN instance, which belongs to the same VPC as that of the primary EC2 instance.
4. Modify the route table on the primary site. Ensure that the route table entry directs the secondary site traffic through the primary site VPN instance.
5. Create the secondary site EC2 instance.

6. Create the secondary site VPN instance, which belongs to the same VPC as that of the secondary EC2 instance.
7. Modify the route table on the secondary site. Ensure that the route table entry directs the primary site traffic through the secondary site VPN instance.
8. Set up connectivity across regions using software VPN. The sample configuration uses Openswan.

Perform the following steps:

- Install the Openswan packages on the primary and secondary VPN instances.
- Configure the `/etc/ipsec.conf` and `/etc/ipsec.secrets` files.

> **Note:** The/etc/ipsec.conffile contains information about the private IP address of the VPN instance, the subnet range of the left subnet, elastic IP address of the destination VPN, the subnet range of the destination right subnet. The/etc/ipsec.secretsfile contains the secret key. This key must be the same on both VPN sites.

- Restart the IPSec service.

```
\# service ipsec restart

-  Add the IPSec connection.

\# ipsec auto --add vpc2vpcConnection

\# ipsec auto --up vpc2vpcConnection
```

- Enable IPSec forwarding. \# sysctl -w net.ipv4.ip_forward=1

```
1.  Set up replication between the instances using the private IP
address or virtual private IP address.



For instructions on setting up replication, refer to the *InfoScale
Replication Administrator's Guide - Linux*.
```

```
1.  Verify the status of replication.


```bash

\# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

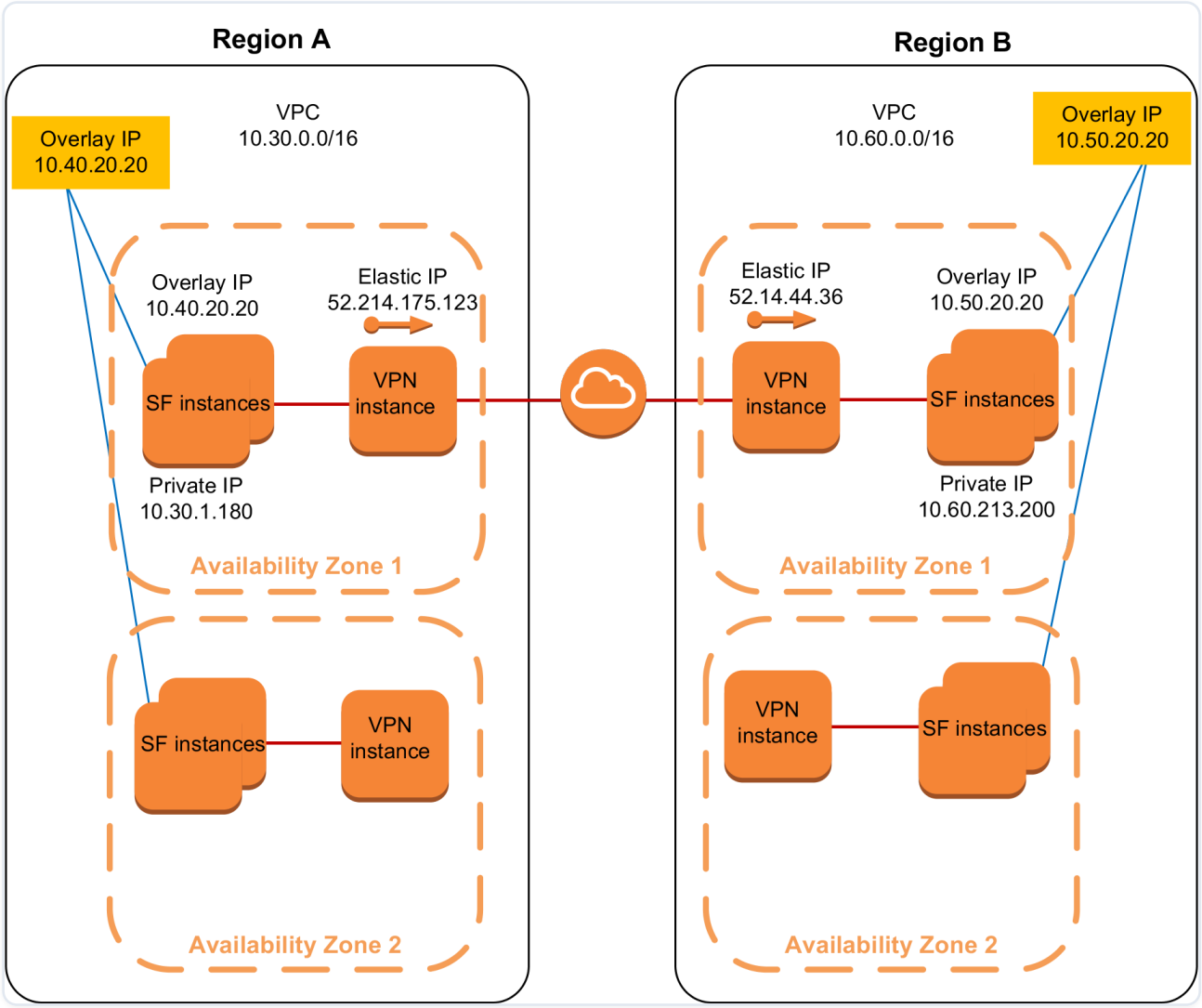
Replication across multiple AWS AZs and regions (campus cluster) - Linux

In this scenario, data is replicated across multiple Availability Zones and regions. The configuration uses software VPN `Openswan` to connect the VPCs across different regions.

- Two VPCs with valid CIDR blocks (for example, 10.30.0.0/16 and 10.60.0.0/16 respectively) that are located in two different regions.
- The primary instance belongs to AZ1 of region 1 and the secondary instance belongs to AZ2 of region 2.
- InfoScale instances in each AZ.
- The primary Instance communicates with the secondary Instance using VPN instances at both ends.
- A VPN tunnel to secure communication between the instances across VPCs.
- Elastic IP addresses (EIP) to connect the two VPN instances
- Private IP addresses used for replication in standalone environments OR

Overlay IP addresses used for replication in clustered environments.

Figure: Replication across multiple AWS AZs and regions
```



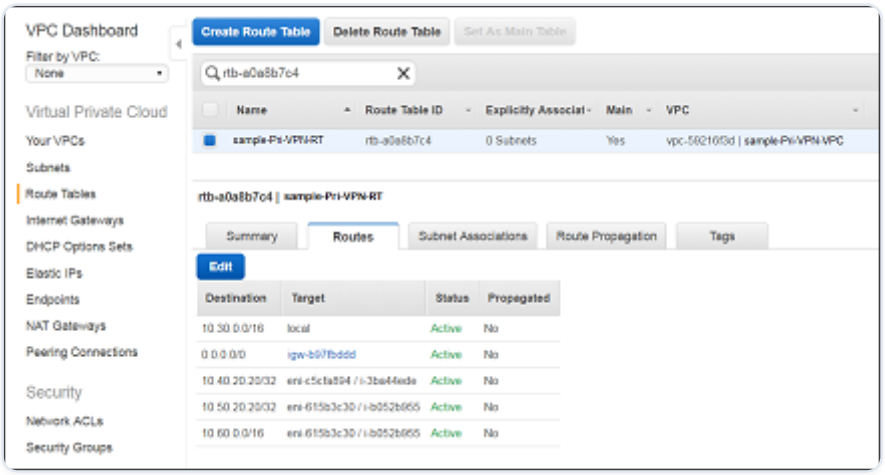
Setting up replication across multiple Availability Zones and regions (campus cluster)

Perform the steps in the following procedure to set up replication across regions.

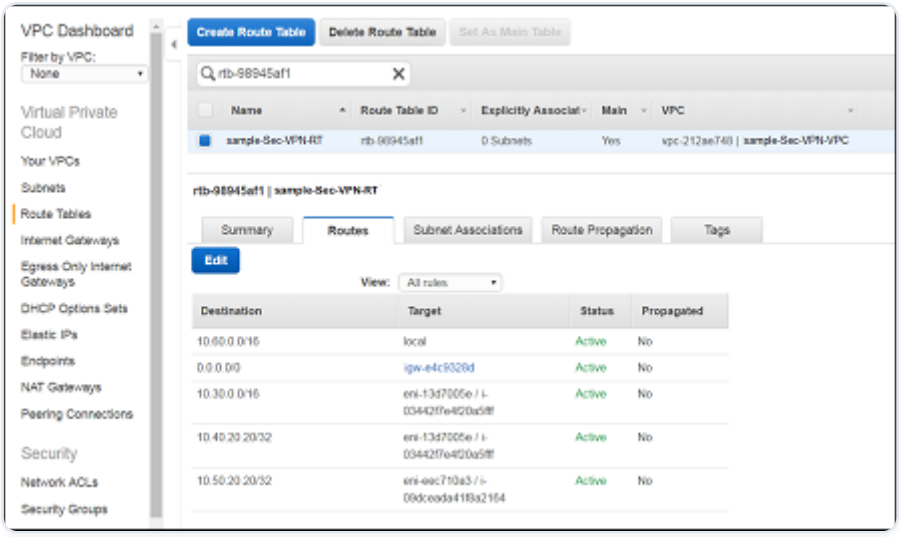
To set up replication across regions

1. Create two VPCs with valid CIDR blocks, for example, 10.30.0.0/16 and 10.60.0.0/16 respectively.
2. Create the primary site EC2 instances in the respective Availability Zones of the region.
3. Create the primary site VPN instances in the respective Availability Zones of the region. The VPN instances belong to the same VPC as that of the primary EC2 instance.
4. Choose a valid overlay IP address as the replication IP address for the primary site. The overlay IP address is a private IP address outside of the primary site's VPC CIDR block. Plumb the overlay IP address on the master node of the primary site cluster.
5. Modify the route table on the primary site to include the overlay IP address. Ensure that the route table entry directs any traffic destined for the primary site to be routed through the

secondary VPN instance and the traffic destined for the secondary site overlay IP to be routed through the secondary InfoScale instance.



6. Create the secondary site EC2 instances in the respective Availability Zones of the second region.
7. Create the secondary site VPN instances in the respective Availability Zones of the second region. The VPN instances belong to the same VPC as that of the secondary EC2 instance.
8. Choose a valid overlay IP address as the replication IP address for the secondary site. The overlay IP address is a private IP address outside of the secondary site's VPC CIDR block. Plumb the overlay IP address on the master node of the secondary site cluster.
9. Modify the route table on the secondary site. Ensure that the route table entries direct traffic destined for the secondary site to be routed through the primary VPN instance and the traffic destined for the primary site overlay IP to be routed through the primary InfoScale instance.



10. Set up connectivity across regions using software VPN. The sample configuration uses Openswan.

Perform the following steps:

- Install the Openswan packages on the primary and secondary VPN instances.
- Configure the `/etc/ipsec.conf` and `/etc/ipsec.secrets` files.

> **Note:** The/etc/ipsec.conffile contains information about the private IP address of the VPN instance, the subnet range of the left subnet, elastic IP address of the destination VPN, the subnet range of the destination right subnet. The/etc/ipsec.secretsfile contains the secret key. This key must be the same on both VPN sites.

- Restart the IPSec service.

```
\# service ipsec restart

-  Add the IPSec connection.

\# ipsec auto -add vpc2vpcConnection

\# ipsec auto -up vpc2vpcConnection
```

- Enable IPSec forwarding. \# sysctl -w net.ipv4.ip_forward=1

```
1.  Modify the ``ipsec.conf`` file to add the overlay IP address for
both primary and secondary site VPN instances.



    ![image](/content/dam/infoscale/us/en/infoscale/media/img/
image_9ecc7248fd03.png)



1.  Verify whether or not the master nodes on the primary and secondary
site can reach each other using the overlay IP address.

1.  Set up replication between the primary and secondary sites.



For instructions on setting up replication, refer to the *InfoScale
```

```
Replication Administrator's Guide - Linux*.



1.  Verify the status of replication.



```bash

\# vradmin -g dg_name repstatus rvg_name
```

Ensure that the RLINK is in CONNECT state and the replication status shows:

```
Replication status: replicating (connected)
```

HA and DR configurations in AWS - Linux

InfoScale Enterprise lets you configure applications for high availability (HA) and disaster recovery (DR) in Amazon Web Services (AWS) cloud.

For more information on the supported use cases, read the following topics:

- See Failover within a subnet of an AWS AZ using virtual private IP - Linux.
- See Failover across AWS subnets using overlay IP - Linux.
- See Public access to InfoScale cluster nodes in AWS using elastic IP - Linux.
- See DR from on-premises to AWS and across AWS regions or VPCs - Linux.

InfoScale Enterprise provides the AWSIP agent and the Route53 agent to manage the network resources for cluster communication, failover, and failback. The agents monitor and manage the IP resources within cluster nodes in the AWS cloud or cluster nodes that are spread across on-premises and AWS. For more information on the agents, see the *InfoScale Cluster Server Bundled Agents Reference Guide - Linux* for your operating system.

Note: Acrtera recommends that you configure three coordination point servers (CP servers) in three different availability zones (AZs) to manage I/O fencing.
```

> Application data is data is stored on the Elastic Block Store (EBS) volumes that are attached to Elastic Compute Cloud (EC2) instances. In case of clusters with two or more nodes, the Flexible Storage Sharing (FSS) feature enables network sharing of the EBS volumes. Thus, application data can be shared between the cluster nodes. For more information on FSS, refer to the *InfoScale Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

InfoScale Enterprise lets you configure replication between EC2 instances, which is further used to support various HA and DR scenarios for applications in AWS.

For more information on the supported replication configurations in AWS, refer to the following documents:

- *InfoScale Replication Administrator's Guide - Linux*
- *InfoScale Disaster Recovery Implementation Guide - Linux*

Failover within a subnet of an AWS AZ using virtual private IP - Linux

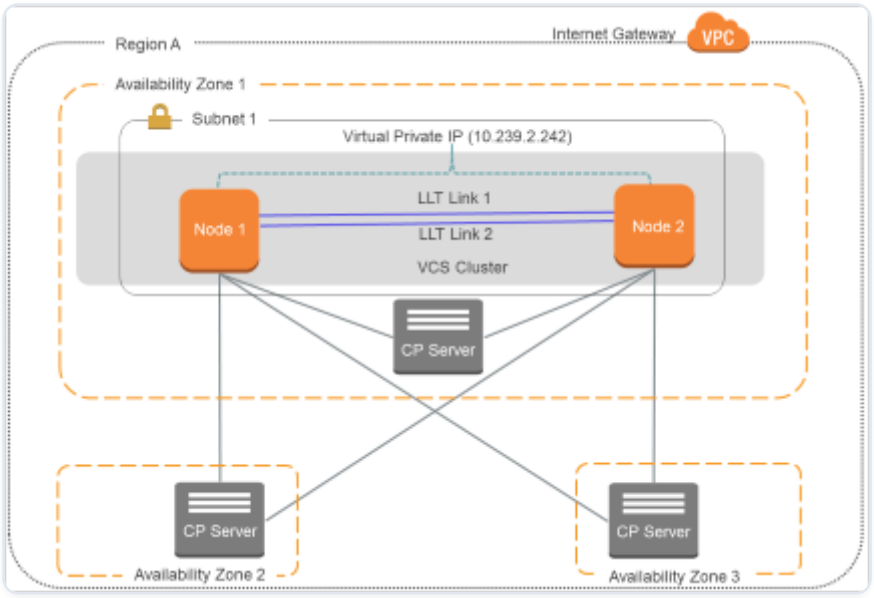
InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - within the same subnet of an AZ.

The following information is required:

- The virtual private IP (secondary private IP) address to be failed over
- The device to which the IP should be plumbed
- The directory in which the AWS CLI is installed; this input is not required if it is provided in the `PATH` environment variable

Sample configuration with private IP

The following graphic depicts a sample failover configuration within the same subnet using a virtual private IP:



The sample configuration includes the following elements:

- A virtual private cloud (VPC) is configured in Region A of the AWS cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node1 and Node2, which are EC2 instances.
- Both the cluster nodes exist in the same subnet.
- A virtual private IP is configured, which is failed over from one node to the other as part of the failover or the failback operations.

Sample service group configuration with private IP

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group appnetworkSG (

     SystemList = { ip-172-34-20-109 = 0, ip-172-34-30-231 = 1 }

     AutoStartList = { ip-172-34-20-109, ip-172-34-30-231 }

  )



     AWSIP AwsIp_Res (

          PrivateIP = "10.239.2.242"
```

```
            Device = eth0
            AWSBinDir = "/usr/local/bin"
            )


      IP Ip_Res (
            Device = eth0
            Address = "10.239.2.242"
            NetMask = "255.255.252.0"
            )
      NIC Nic_Res (
            Device = eth0
            )


AwsIp_Res requires Ip_Res
Ip_Res requires Nic_Res
```

Failover across AWS subnets using overlay IP - Linux

InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - between different subnets in the same AZ or in different AZs.

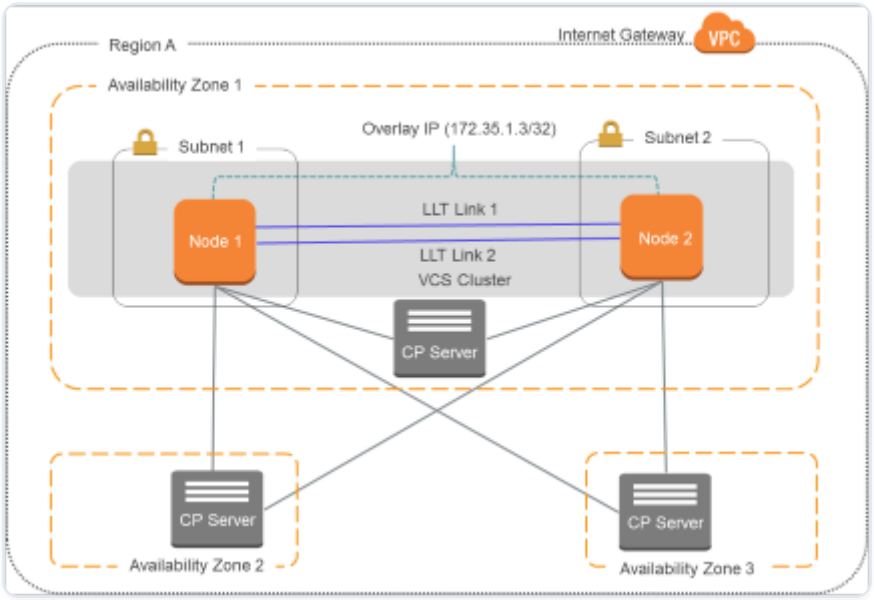
The following information is required:

- The overlay IP address to be used for failover
- The device to which the IP should be plumbed
- The directory in which the AWS CLI is installed; this input is not required if it is provided in the PATH environment variable

AWS does not allow the private IP of one subnet to be failed over to a different subnet. To overcome this limitation, InfoScale Enterprise provides an overlay IP, which is defined at the VPC level, so that it can be used across subnets.

Sample configuration with overlay IP for failover across subnets in the same AZ

The following graphic depicts a sample failover configuration across subnets within the same AZ using an overlay IP:



The sample configuration includes the following elements:

- A virtual private cloud (VPC) is configured in Region A of the AWS cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node 1 and Node 2, which are EC2 instances.

Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.

- An overlay IP is configured, which allows the private IP of a node to be failed over from one subnet to another in an AZ as part of the failover or the failback operations.

Sample service group configuration with overlay IP for failover across subnets in the same AZ

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group appnetworkSG (

     SystemList = { ip-172-34-20-109 = 0, ip-172-34-30-231 = 1 }

     AutoStartList = { ip-172-34-20-109, ip-172-34-30-231 }
```

```
    )


    AWSIP AwsIp_Res (
        OverlayIP = "172.35.1.3/32"
        Device = eth0
        AWSBinDir = "/usr/local/bin"
        )

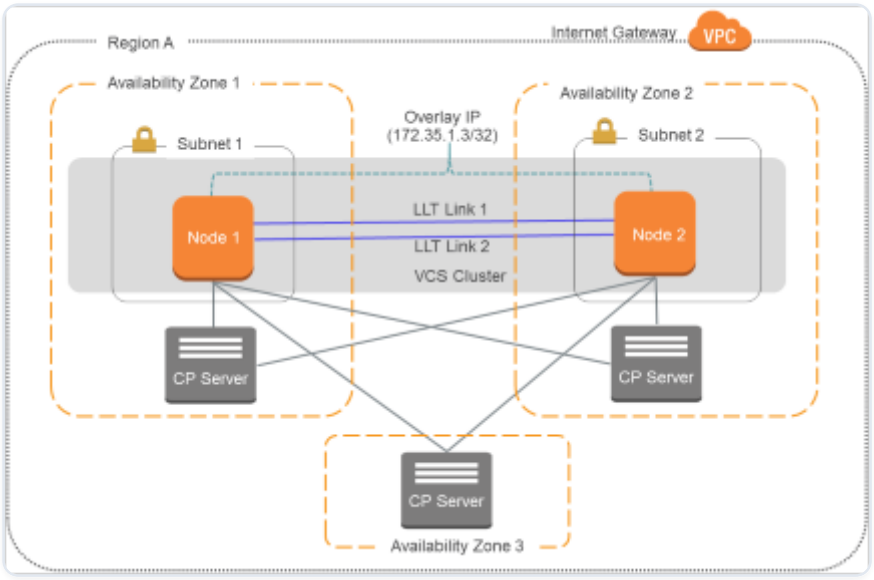

    IP Ip_Res (
        Device = eth0
        Address = "172.35.1.3"
        NetMask = "255.255.255.255"
        )


    NIC Nic_Res (
        Device = eth0
        )


AwsIp_Res requires Ip_Res
Ip_Res requires Nic_Res
```

Sample configuration with overlay IP for failover across subnets in different AZs

The following graphic depicts a sample failover configuration across subnets in different AZs using an overlay IP:



Sample service group configuration with overlay IP for failover across subnets in different AZs

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group appnetworkSG (

     SystemList = { ip-172-34-20-109 = 0, ip-172-34-30-231 = 1 }

     AutoStartList = { ip-172-34-20-109, ip-172-34-30-231 }

     )



     AWSIP AwsIp_Res (

          OverlayIP = "172.35.1.3/32"

          Device = eth0

          AWSBinDir = "/usr/local/bin"

          )
```

```
     IP Ip_Res (
          Device = eth0
          Address = "172.35.1.3"
          NetMask = "255.255.255.255"
          )

     NIC Nic_Res (
          Device = eth0
          )

AwsIp_Res requires Ip_Res
Ip_Res requires Nic_Res
```

Public access to InfoScale cluster nodes in AWS using elastic IP - Linux

To allow public access to an InfoScale cluster node or to an application configured for HA or DR in AWS, specify the IP to be used in the `ElasticIP` attribute for the AWSIP resource. For example, if you have an application that needs to be highly available and to be accessible globally, you can use the `ElasticIP` attribute of the AWSIP agent to ensure both.

Sample service group configuration with elastic IP

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group appnetworkSG (
     SystemList = { ip-172-34-20-109 = 0, ip-172-34-30-231 = 1 }
     AutoStartList = { ip-172-34-20-109, ip-172-34-30-231 }
```

```
    )


    AWSIP AwsIp_Res (
        PrivateIP @ ip-172-34-20-109 ="172.34.20.110"
        PrivateIP @ ip-172-34-30-231 ="172.34.30.220"
        ElasticIP = "52.3.20.17"
        Device = eth0
        AWSBinDir = "/usr/local/bin"
        )


    IP Ip_Res (
        Device = eth0
        Address @ ip-172-34-20-109 ="172.34.20.110"
        Address @ ip-172-34-30-231 ="172.34.30.220"
        NetMask = "255.255.240.0"
        )


    NIC Nic_Res (
        Device = eth0
        )


AwsIp_Res requires Ip_Res
```

```
Ip_Res requires Nic_Res
```

DR from on-premises to AWS and across AWS regions or VPCs - Linux

InfoScale Enterprise lets you use the global cluster option (GCO) for DR configurations. You can use a DR configuration to fail over applications across different regions or VPCs in AWS. The cluster nodes can be in the same AZ or in different AZs.

The following information is required:

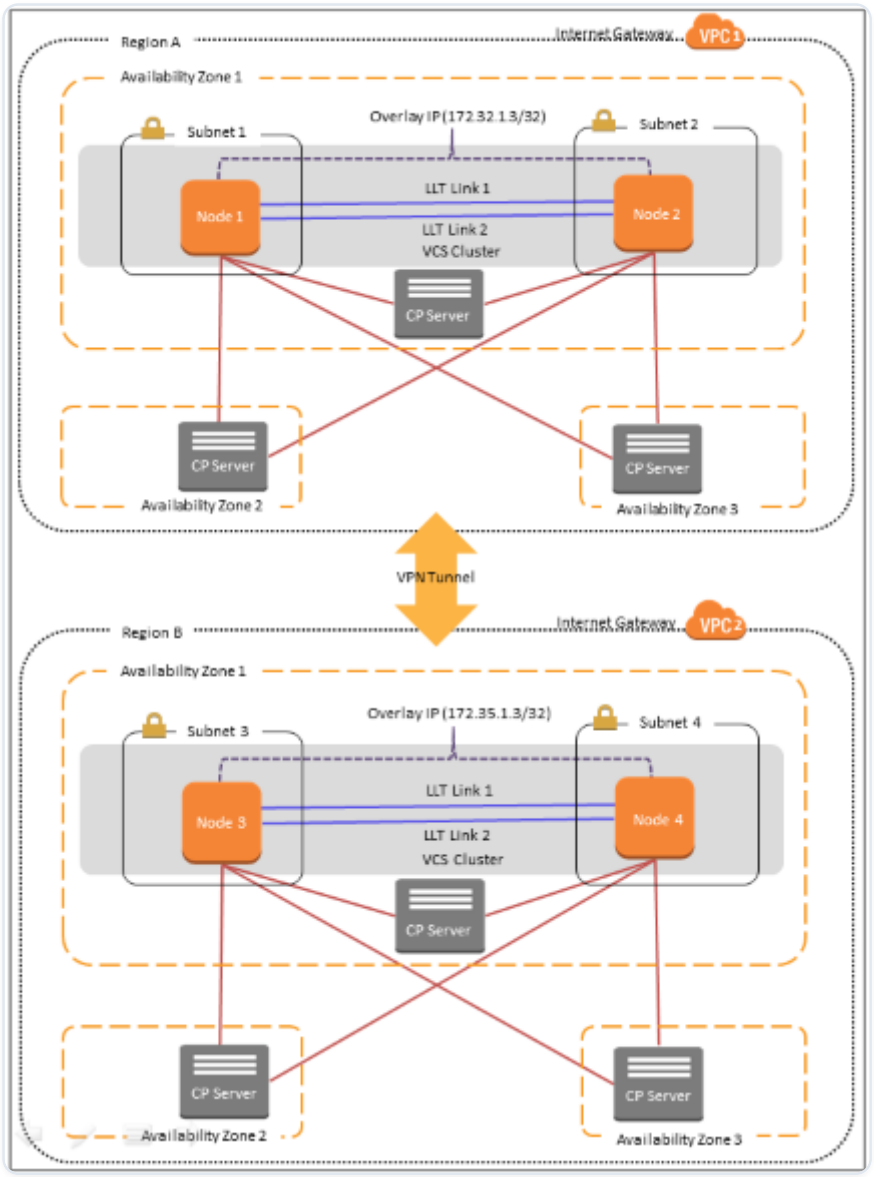
- VPN tunnel information between regions or VPCs
- The IP address to be used for cross-cluster communication:
    - Virtual private IP for cluster the nodes that exist in the same subnet
    - Overlay IP for cluster the nodes that exist in different subnets

You can also use GCO to configure applications for DR from an on-premises site to AWS.

Note: If you use an Amazon VPN tunnel in a global cluster configuration between an on-premises site and AWS, the cluster nodes in the cloud must be in the same subnet.

Sample configuration for DR across regions or VPCs

The following graphic depicts a sample DR configuration across AWS regions:



The sample configuration includes the following elements:

- VPN tunnel between Region A and Region B
- The primary site has the following elements:
  - A virtual private cloud, VPC 1, is configured in Region A of the AWS cloud.
  - An application is configured for HA using an InfoScale cluster that comprises two nodes, Node 1 and Node 2, which are EC2 instances.

Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.

- The overlay IP allows the private IP of a node to be fail over from one subnet to another in an AZ during failover or failback.
- The secondary site has the following elements:
- A virtual private cloud, VPC 2, is configured in Region B of the AWS cloud.

- The same application is configured for HA on Node 3 and Node 4, which exist in Subnet 3 and Subnet 4 respectively.
- The overlay IP allows the private IP of a node to fail over from one subnet to another in an AZ.

Sample service group configuration for GCO across regions

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) at the primary site (Region A):

```
include "types.cf"



cluster sitever (

     ClusterAddress = "172.32.1.2"

  SecureClus = 1

     )



remotecluster sitecal (

     ClusterAddress = "172.35.1.2"

     ConnectTimeout = 3000

     SocketTimeout = 3000

     )



heartbeat Icmp (

     ClusterList = { sitecal }

     Arguments @sitecal = { "172.35.1.2" }

     )
```

```
system ip-172-31-21-156 (

     )


system ip-172-31-61-106 (

     )


group ClusterService (

     SystemList = { ip-172-31-21-156 = 0, ip-172-31-61-106 = 1 }

  AutoStartList = { ip-172-31-21-156, ip-172-31-61-106 }

     OnlineRetryLimit = 3

     OnlineRetryInterval = 120

     )


    AWSIP Aws_Ipres (

         OverlayIP = "172.32.1.2/32"

         Device = eth0

         AWSBinDir = "/usr/local/bin"

         )


    Application wac (

         StartProgram = "/opt/VRTSvcs/bin/wacstart -secure"
```

```
        StopProgram = "/opt/VRTSvcs/bin/wacstop"
        MonitorProcesses = { "/opt/VRTSvcs/bin/wac -secure" }
        RestartLimit = 3
        )


    IP Ipres (
        Device = eth0
        Address = "172.32.1.2"
        NetMask = "255.255.255.0"
        )


    NIC gconic (
        Device = eth0
        )


    Aws_Ipres requires Ipres
    Ipres requires gconic
    wac requires Ipres
```

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) at the secondary site (Region B):

```
include "types.cf"



cluster sitecal (

        ClusterAddress = "172.35.1.2"

        SecureClus = 1

        )



remotecluster sitever (

        ClusterAddress = "172.32.1.2"

        ConnectTimeout = 3000

        SocketTimeout = 3000

        )



heartbeat Icmp (

        ClusterList = { sitever }

        Arguments @sitever = { "172.32.1.2" }

        )



system ip-172-34-20-109 (

        )



system ip-172-34-30-231 (
```

```
        )


group ClusterService (
        SystemList = { ip-172-34-20-109 = 0, ip-172-34-30-231 = 1 }
        AutoStartList = { ip-172-34-20-109, ip-172-34-30-231 }
        OnlineRetryLimit = 3
        OnlineRetryInterval = 120
        )


AWSIP Aws_Ipres (
        OverlayIP = "172.35.1.2/32"
        Device = eth0
        AWSBinDir = "/usr/local/bin"
        )


Application wac (
        StartProgram = "/opt/VRTSvcs/bin/wacstart -secure"
        StopProgram = "/opt/VRTSvcs/bin/wacstop"
        MonitorProcesses = { "/opt/VRTSvcs/bin/wac -secure" }
        RestartLimit = 3
        )
```

```
IP Ipres (
        Device = eth0
        Address = "172.35.1.2"
        NetMask = "255.255.255.0"
        )

NIC gconic (
        Device = eth0
        )

Aws_Ipres requires Ipres
Ipres requires gconic
wac requires Ipres
```

Configurations for Amazon Web Services - Windows

This section includes the following topics:

- Replication configurations in AWS - Windows
- HA and DR configurations in AWS - Windows

Replication configurations in AWS - Windows

The procedure for setting up replication in AWS depends on where your primary data center and your secondary data center are located.

Follow the procedure that is appropriate for your setup:

- See Replication from on-premises to AWS - Windows.
- See Replication across AZs in an AWS region - Windows.
- See Replication across AWS regions - Windows.

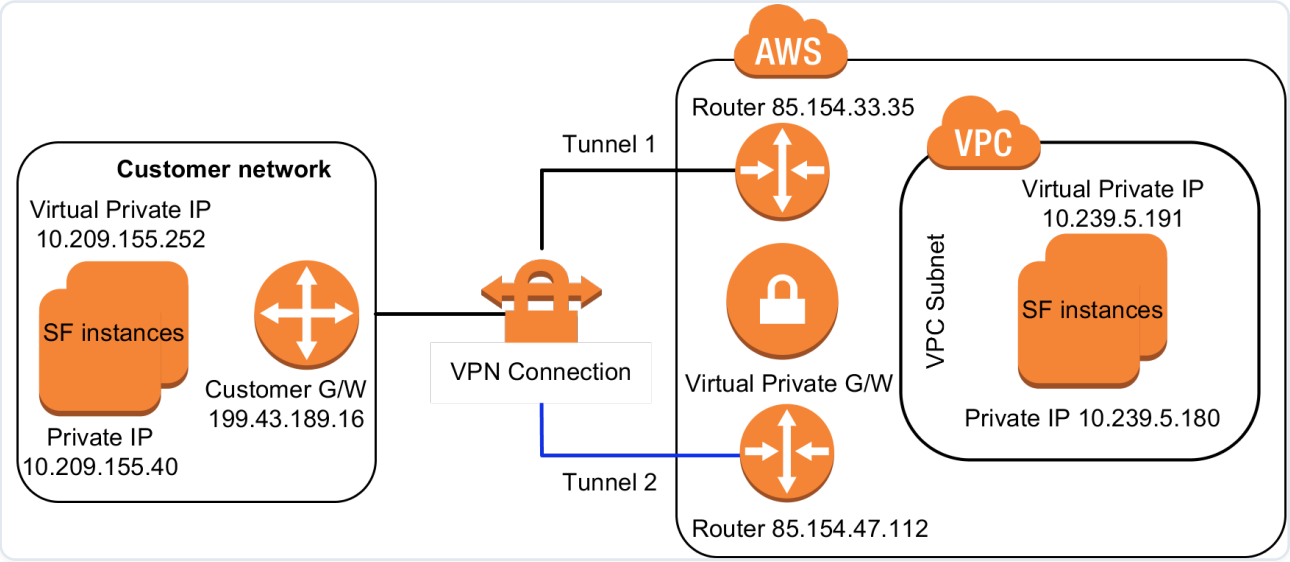
Replication from on-premises to AWS - Windows

In this scenario, data is replicated from an on-premises data center to an on-cloud data center.

Sample configuration

The following graphic illustrates the configuration.

Figure: Sample replication configuration from on-premises to AWS



Prerequisites

Ensure that the following requirements are met before you proceed with the configuration:

- Required ports are open for communication between an on-premise data center and an on-cloud data center.

For details, refer to the *InfoScale Installation and Upgrade Guide - Windows*.

- The virtual private IP addresses are plumbed on both the nodes.
- The virtual private IP addresses are configured within the subnet.

Setting up replication

The following procedure lists high-level tasks to configure replication from an on-premises data center to a cloud data center.

To set up replication from on-premises to AWS

1. Using AWS portal, create a Virtual Private Cloud (VPC) with a valid CIDR block, for example 10.239.0.0/16.
2. Create a subnet in the VPC created and assign it to an Availability Zone.
3. Create an EC2 instance and associate it with the VPC created.
4. Configure a virtual private gateway and associate it to the VPC.
5. Configure a gateway in the on-premise data center.
6. Create route table entries.
7. Associate the subnet with the route table.
8. Enable route propagation to automatically propagate the routes to the table.

On the Route Propagation tab in the details pane, choose Edit, and select the virtual private gateway that you created.

1. Create a VPN connection.
2. Download the VPN configuration file.
3. Create a VPN tunnel between the on-premise network and the on-cloud network.
4. Install the appropriate InfoScale product on EC2 instances in both the data centers.
5. Create VxVM disk groups, VxVM volumes, Replicated Volume Group (RVG), and RLinks.
6. Set up replication between the on-premise and on-cloud instances.
7. Verify the status of replication.

   ```
   \# vradmin -g dg_name repstatus rvg_name
   ```

   Ensure that the replication status shows:

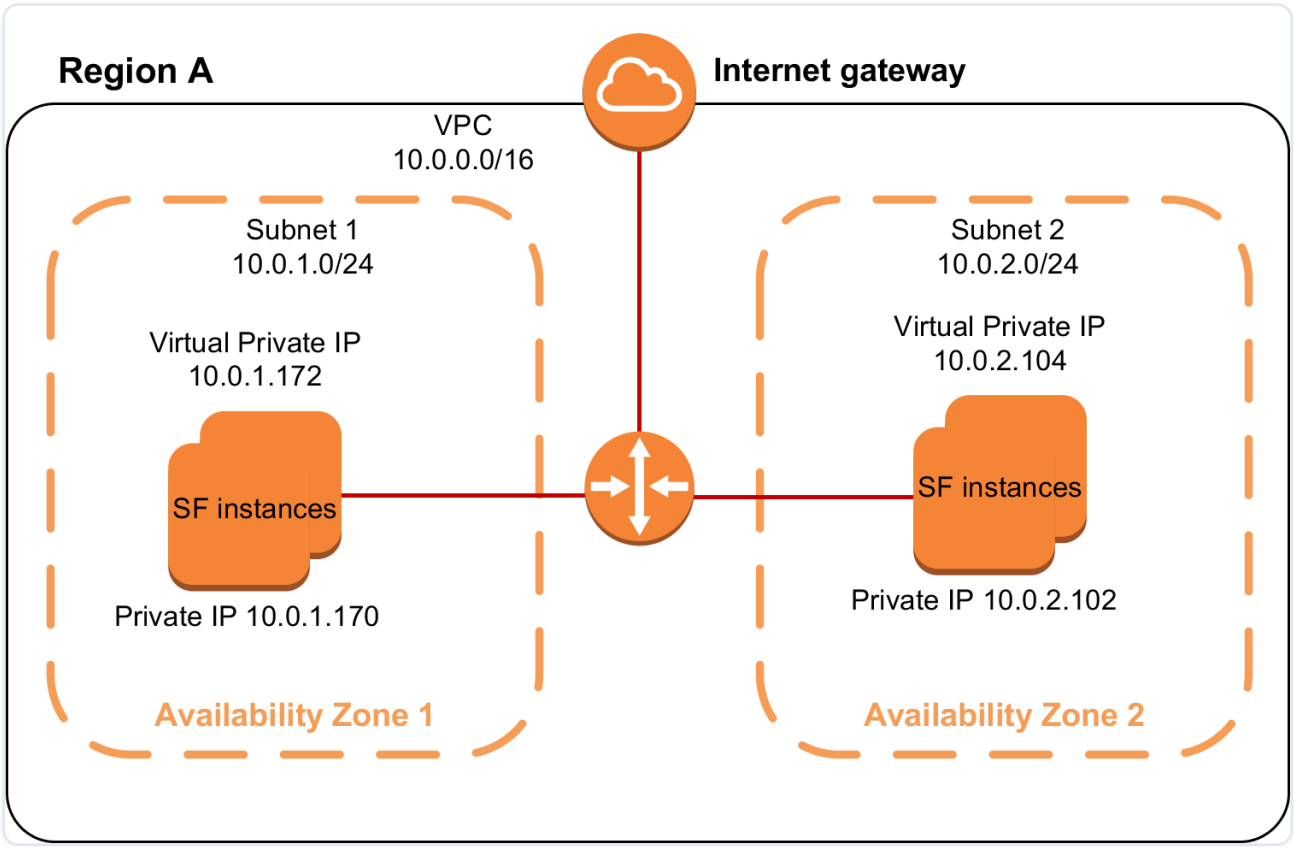
   ```
   Replication status: replicating (connected)
   ```

Replication across AZs in an AWS region - Windows

In this scenario, data is replicated between two on-cloud data centers that are located in different Availability Zones within a same region.

The following diagram illustrates a sample configuration for setting up replication between data centers that are located in different Availability Zones within a same region.

Figure: Replication across Availability Zones



Setting up replication across AZs within the same region

Perform the steps in the following procedure to set up replication across AZs within the same region.

To set up replication across AZs within the same region

1. Create a VPC with a valid CIDR block, for example, 10.0.0.0/16.
2. Create the internet gateway and attach it to the VPC.
3. Modify the VPC route table such that the two instances across availability zones can communicate with each other using private IP addresses.
4. Create two subnets - one subnet for the primary site in AZ1 and the second subnet for the secondary site in AZ2 with valid CIDR range, for example 10.0.1.0/24 and 10.0.2.0/24 respectively.
5. Launch the EC2 instances in the primary and secondary subnets. Install InfoScale on the instances.
6. Verify connectivity between the virtual private IP addresses of the instances.

   ```
   \# ping PIP
   ```

   a. Install the appropriate InfoScale product on EC2 instances in both the data centers.

b. Create VxVM disk groups, VxVM volumes, Replicated Volume Group (RVG), and RLinks.
c. Set up replication between the instances using private IP address or virtual private IP address.
d. Verify the status of replication.

```
\# vradmin -g dg_name repstatus rvg_name
```

Ensure that the status shows:

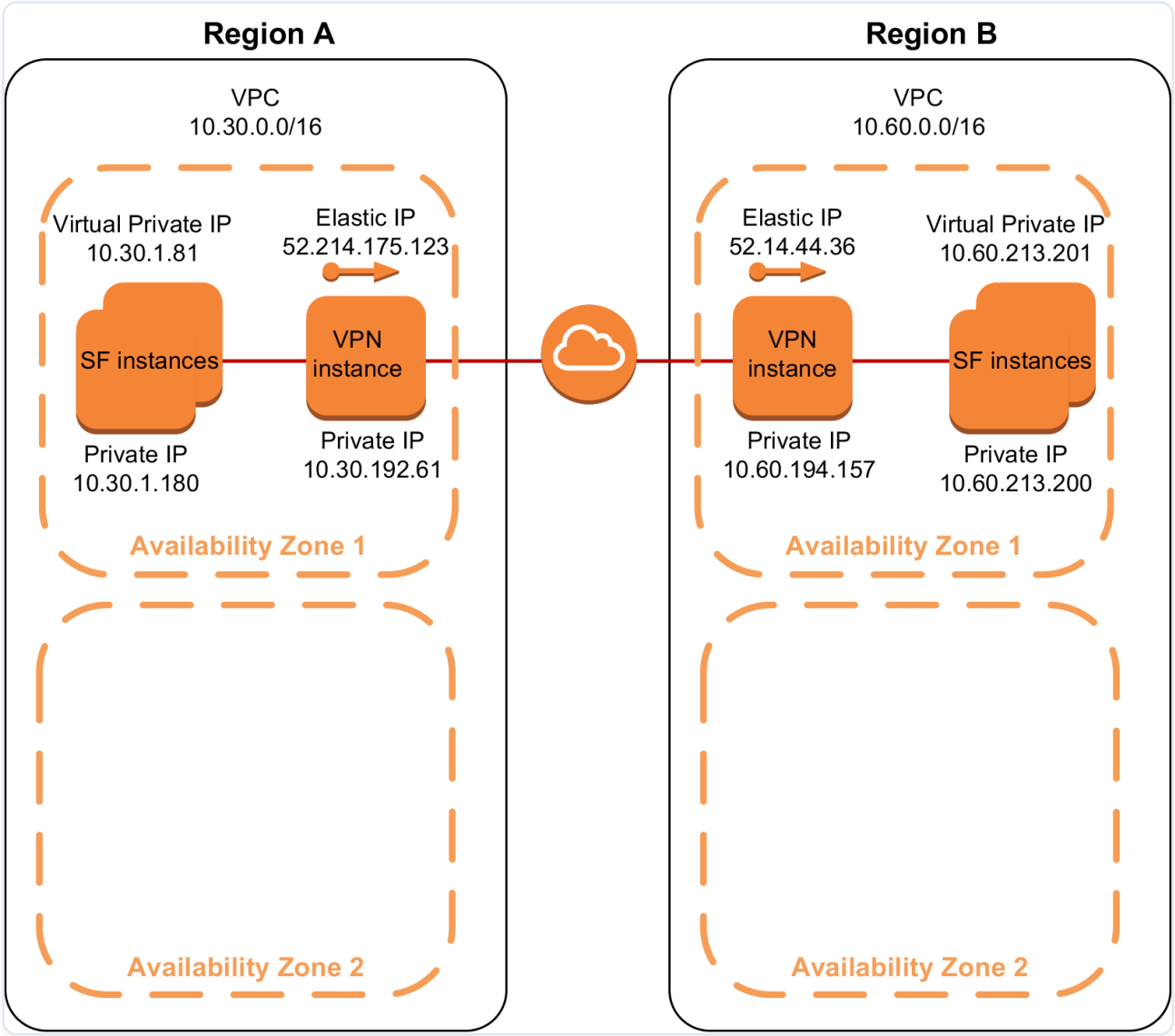
```
Replication status: replicating (connected)
```

Replication across AWS regions - Windows

In this scenario, replication is set up across Availability Zones configured in different regions. The configuration uses software VPN `Openswan/OpenVPN` to connect the VPCs across different regions.

The following diagram illustrates a sample configuration for setting up replication across regions.

Figure: Replication across regions



Setting up replication across regions

Perform the steps in the following procedure to set up replication across regions.

To set up replication across regions

1. Create two VPCs with valid CIDR blocks in different regions, for example, 10.30.0.0/16 and 10.60.0.0/16 respectively.
2. Create a primary site EC2 instance.
3. Create a primary site VPN instance, which belongs to the same VPC as that of the primary EC2 instance.
4. Modify the route table on the primary site. Ensure that the route table entry directs the secondary site traffic through the primary site VPN instance.
5. Create a secondary site EC2 instance.

6. Create a secondary site VPN instance, which belongs to the same VPC as that of the secondary EC2 instance.
7. Modify the route table on the secondary site. Ensure that the route table entry directs the primary site traffic through the secondary site VPN instance.
8. Set up connectivity across regions using software VPN.
9. Install the appropriate InfoScale product on the EC2 instances in both the data centers.
10. Create VxVM disk groups, VxVM volumes, Replicated Volume Group (RVG), and RLinks.
11. Set up replication between the instances using the private IP address or virtual private IP address.
12. Verify the status of replication.

    ```
    \# vradmin -g dg_name repstatus rvg_name
    ```

    Ensure that the replication status shows:

    ```
    Replication status: replicating (connected)
    ```

HA and DR configurations in AWS - Windows

InfoScale Enterprise lets you configure applications for high availability (HA) and disaster recovery (DR) in Amazon Web Services (AWS) cloud.

For more information on the supported use cases, read the following topics:

- See Failover within a subnet of an AWS AZ using virtual private IP - Windows.
- See Failover across AWS subnets using overlay IP - Windows.
- See Public access to InfoScale cluster nodes in AWS using Elastic IP - Windows.
- See DR from on-premises to AWS and across AWS regions or VPCs - Windows.
- See DR from on-premises to AWS - Windows.

InfoScale provides the AWSIP agent and the Route53 agent to manage the network resources for cluster communication, failover, and failback. The agents monitor and manage the IP resources within cluster nodes in the AWS cloud or cluster nodes that are spread across on-premises and AWS. For more information on the agents, see the *Cluster Server Bundled Agents Reference Guide* for your operating system.

Application data is data is stored on the Elastic Block Store (EBS) volumes that are attached to Elastic Compute Cloud (EC2) instances. InfoScale lets you configure replication between EC2 instances, which is further used to support various HA and DR scenarios for applications in AWS.

For more information on the supported replication configurations in AWS, see the following documents:

- *Cluster Server Administrator's Guide - Windows*
- *Volume Replicator Administrator's Guide - Windows*

InfoScale provides configuration wizards that you can use to create, modify, and delete file share, MSMQ, Application, and SQL Server HA configurations.

For details about the usage of the Configuration Wizard, refer to the following documents as appropriate:

- *Cluster Server Administrator's Guide - Windows*
- *Cluster Server Bundled Agents Reference Guide - Windows*
- *Storage Foundation and High Availability Solutions, Solutions Guide - Windows*

For details about the usage of the SQL Server Configuration Wizard, refer to the following documents as appropriate:

- *Cluster Server Implementation Guide for Microsoft SQL Server - Windows*
- *Storage Foundation and High Availability Solutions HA and DR Solutions Guide for Microsoft SQL Server - Windows*

EBS Multi-Attach feature support with InfoScale Enterprise in AWS cloud

InfoScale Enterprise now supports EBS Multi-Attached volumes. Amazon EBS Multi-Attach enables you to attach a single Provisioned IOPS SSD volume to multiple instances that are in the same Availability Zone.

You can now use the InfoScale Storage agents to utilize the shared disks in the AWS cloud environment.

> **Note:** AWS shared disk support is limited to InfoScale Enterprise and InfoScale Storage solution.

InfoScale service group configuration wizards support for EBS Multi-Attach

You can use the InfoScale application service group configuration wizards on Windows platform to configure high availability for various applications such as SQL Server and file shares. The service groups automatically configures the required InfoScale AWS agents for managing the application and the network resources in the AWS cloud.

Refer to the *Cluster Server Bundled Agents Reference Guide - Windows* for more information about AWS agents.

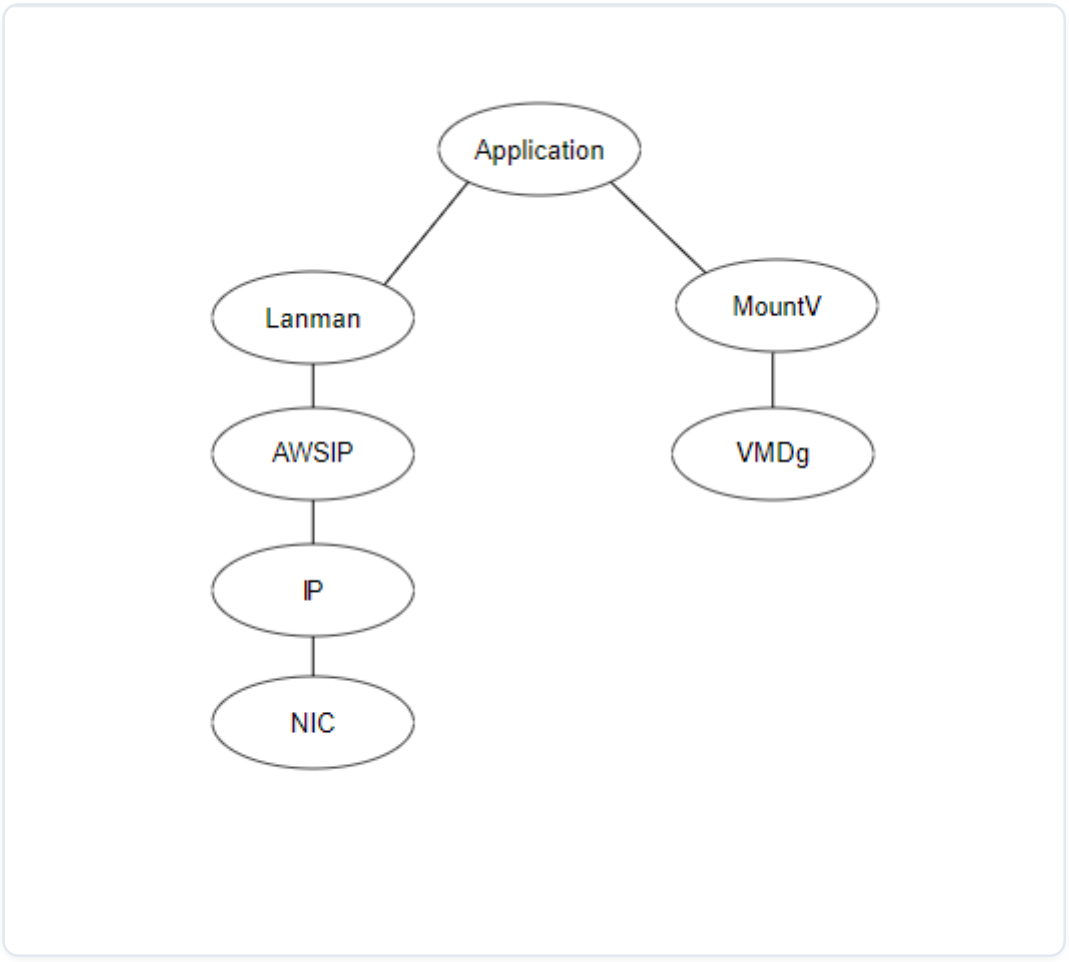
The following limitations are applicable when you use these wizards on Windows:

- If the cluster nodes span across different subnets, the wizards configure the AWSIP agent resources using the private IP address (PrivateIP attribute) instead of using the overlay IP address (OverlayIP attribute).

In such a case, you must run the wizard until the end and complete the service group configuration. After the service group is configured, you must manually modify the service group and replace the PrivateIP entry with the OverlayIP attribute and the overlap IP address.

Sample service group dependency for AWS multi-attach disk

Figure:



Failover within a subnet of an AWS AZ using virtual private IP - Windows

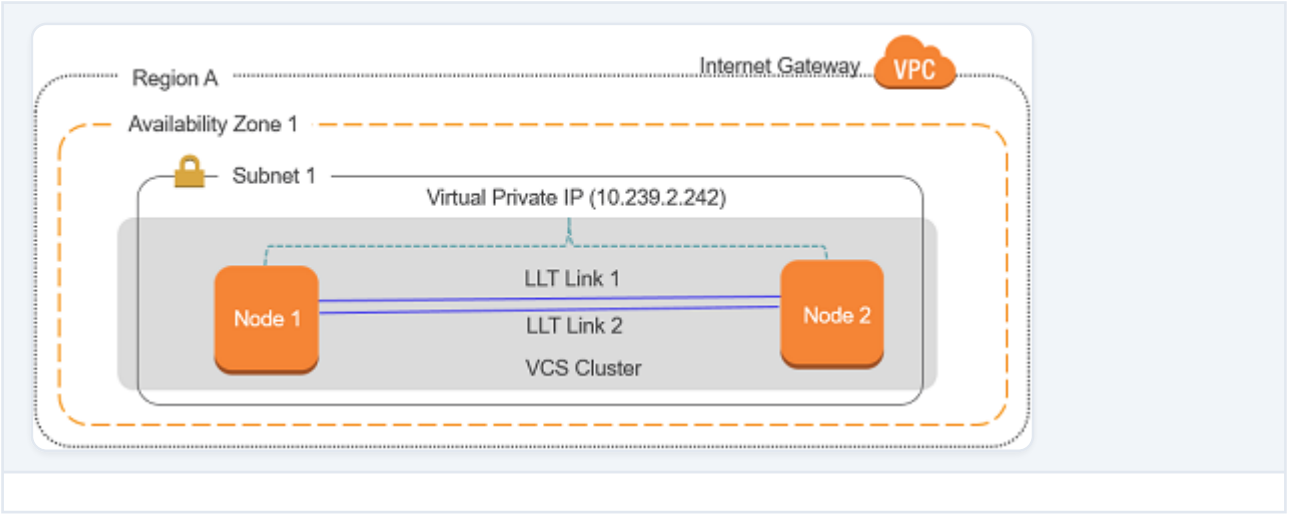
InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - within the same subnet of an AZ.

The following information is required:

- The virtual private IP (secondary private IP) address to be failed over
- The device to which the IP should be plumbed
- The directory in which the AWS CLI is installed; this input is not required if it is provided in the `PATH` environment variable

Sample configuration with private IP

The following graphic depicts a sample failover configuration within the same subnet using a virtual private IP:



The sample configuration includes the following elements:

- A virtual private cloud (VPC) is configured in Region A of the AWS cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node1 and Node2, which are EC2 instances.
- Both the cluster nodes exist in the same subnet.
- A virtual private IP is configured, which is failed over from one node to the other as part of the failover or the failback operations.

Sample service group configuration with private IP

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group AWSIPGrp (

        SystemList = { WIN-38PNEVJSR2K = 0 , WIN-39PNEVJSR2K = 1  }

        AutoStartList = { WIN-38PNEVJSR2K, WIN-39PNEVJSR2K }

        )



        AWSIP awsip (

                PrivateIP = "10.239.3.96"

                Device@WIN-38PNEVJSR2K  = 12-7F-CE-5B-E2-6E
```

```
                Device@WIN-39PNEVJSR2K = 12-7F-CE-5B-E2-6F


                )


        IP ipres (
                Address = "10.239.3.96"
                SubNetMask = "255.255.254.0"
                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"
                MACAddress @WIN-39PNEVJSR2K = "12:7F:CE:5B:E2:6F"
                )


        NIC nicres (
                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"
                MACAddress @WIN-39PNEVJSR2K = "12:7F:CE:5B:E2:6F"
                )


ipres requires nicres
awsip requires ipres
```

Failover across AWS subnets using overlay IP - Windows

InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - between different subnets in the same AZ or in different AZs.

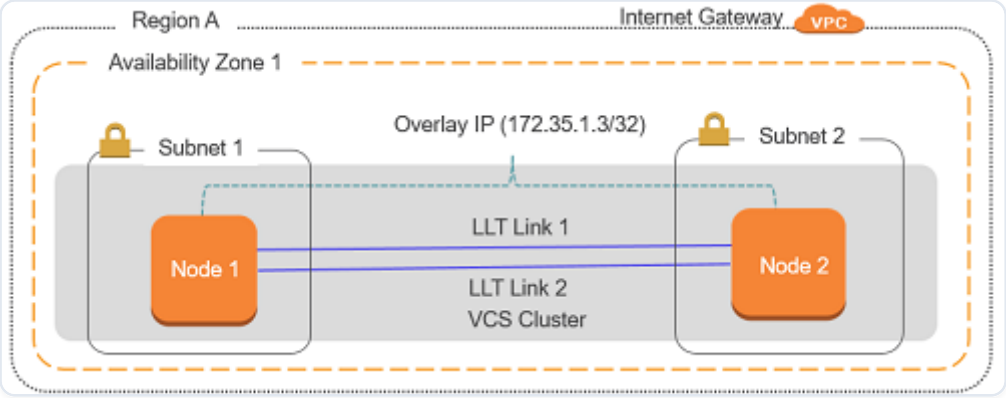
The following information is required:

- The overlay IP address to be used for failover
- The device to which the IP should be plumbed
- The directory in which the AWS CLI is installed; this input is not required if it is provided in the PATH environment variable

AWS does not allow the private IP of one subnet to be failed over to a different subnet. To overcome this limitation, InfoScale Enterprise provides an overlay IP, which is defined at the VPC level, so that it can be used across subnets.

Sample configuration with overlay IP for failover across subnets in the same AZ

The following graphic depicts a sample failover configuration across subnets within the same AZ using an overlay IP:



The sample configuration includes the following elements:

- A virtual private cloud (VPC) is configured in Region A of the AWS cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node 1 and Node 2, which are EC2 instances.

Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.

- An overlay IP is configured, which allows the private IP of a node to be failed over from one subnet to another in an AZ as part of the failover or the failback operations.

Sample service group configuration with overlay IP for failover across subnets in the same AZ

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group AWSIPGrp (

        SystemList = { WIN-38PNEVJSR2K = 0 , WIN-39PNEVJSR2K = 1  }
```

```
        AutoStartList = { WIN-38PNEVJSR2K, WIN-39PNEVJSR2K }
        )


        AWSIP overlay (
                OverlayIP = "172.16.8.55/32"
                Device @WIN-38PNEVJSR2K = 12-7F-CE-5B-E2-6E
                Device@WIN-39PNEVJSR2K = 12-7F-CE-5B-E2-6F
                RouteTableIds = { rtb-c5272ca3, rtb-fb97ac9d }
                )
        IP ipres (
                Address= "172.16.8.55"
                SubNetMask = "255.255.254.0"
                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"
                MACAddress @WIN-39PNEVJSR2K = "12:7F:CE:5B:E2:6F"
                )

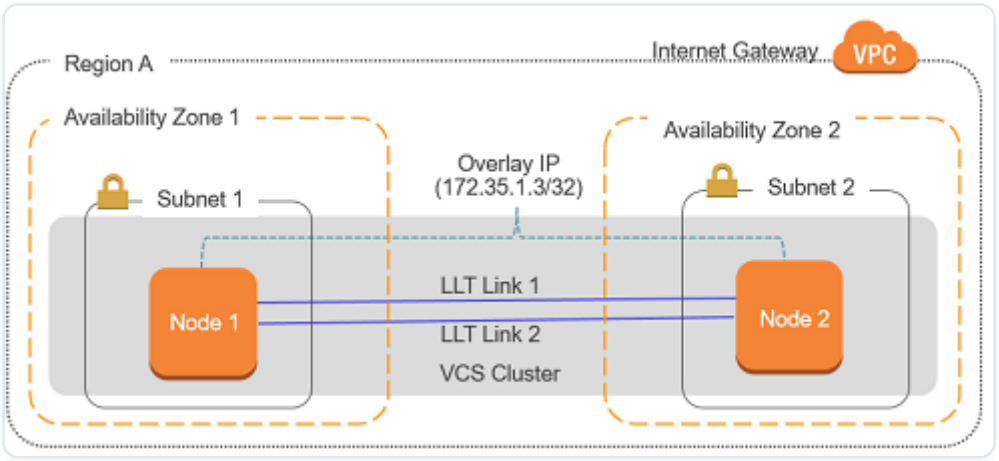

        NIC nicres (
                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"
                MACAddress @WIN-39PNEVJSR2K = "12:7F:CE:5B:E2:6F"
                )


ipres requires nicres
```

```
overlayip requires ipres
```

Sample configuration with overlay IP for failover across subnets in different AZs

The following graphic depicts a sample failover configuration across subnets in different AZs using an overlay IP:



Sample service group configuration with overlay IP for failover across subnets in different AZs

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group AWSIPGrp (

        SystemList = { WIN-38PNEVJSR2K = 0 , WIN-39PNEVJSR2K = 1  }

        AutoStartList = { WIN-38PNEVJSR2K, WIN-39PNEVJSR2K }

        )


        AWSIP overlay (

                OverlayIP = "172.16.8.55/32"

                Device @WIN-38PNEVJSR2K = 12-7F-CE-5B-E2-6E

                Device@WIN-39PNEVJSR2K = 12-7F-CE-5B-E2-6F

                RouteTableIds = { rtb-c5272ca3, rtb-fb97ac9d }
```

```
                )

        IP ipres (

                Address = "172.16.8.55"

                SubNetMask = "255.255.254.0"

                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"

                MACAddress @WIN-39PNEVJSR2K = "12:7F:CE:5B:E2:6F"

                )


        NIC nicres (

                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"

                MACAddress @WIN-39PNEVJSR2K = "12:7F:CE:5B:E2:6F"

                )


ipres requires nicres

overlayip requires ipres
```

Public access to InfoScale cluster nodes in AWS using Elastic IP - Windows

To allow public access to an InfoScale cluster node or to an application configured for HA or DR in AWS, specify the IP to be used in the `ElasticIP` attribute for the AWSIP resource. For example, if you have an application that needs to be highly available and to be accessible globally, you can use the `ElasticIP` attribute of the AWSIP agent to ensure both.

Sample service group configuration with Elastic IP

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group AWSIPGrp (

        SystemList = { WIN-38PNEVJSR2K = 0 , WIN-39PNEVJSR2K = 1  }

        AutoStartList = { WIN-38PNEVJSR2K, WIN-39PNEVJSR2K }

        )



        AWSIP elasticIP (

                PrivateIP = "10.239.3.95"

                ElasticIP = "34.193.196.156"

                Device @WIN-38PNEVJSR2K = " 12-7F-CE-5B-E2-6E"


  Device@WIN-39PNEVJSR2K = 12-7F-CE-5B-E2-6F



                )                  )

        IP ipres (

                Address = "10.239.3.95"

                SubNetMask = "255.255.254.0"

                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"

                MACAddress @WIN-39PNEVJSR2K = "12:7F:CE:5B:E2:6F"

                )



        NIC nicres (

                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"
```

```
                    MACAddress @WIN-39PNEVJSR2K = "12:7F:CE:5B:E2:6F"
                    )


ipres requires nicres

elasticIP requires ipres
```

DR from on-premises to AWS and across AWS regions or VPCs - Windows

InfoScale Enterprise lets you use the global cluster option (GCO) for DR configurations. You can use a DR configuration to fail over applications across different regions or VPCs in AWS. The cluster nodes can be in the same AZ or in different AZs.

The following information is required:

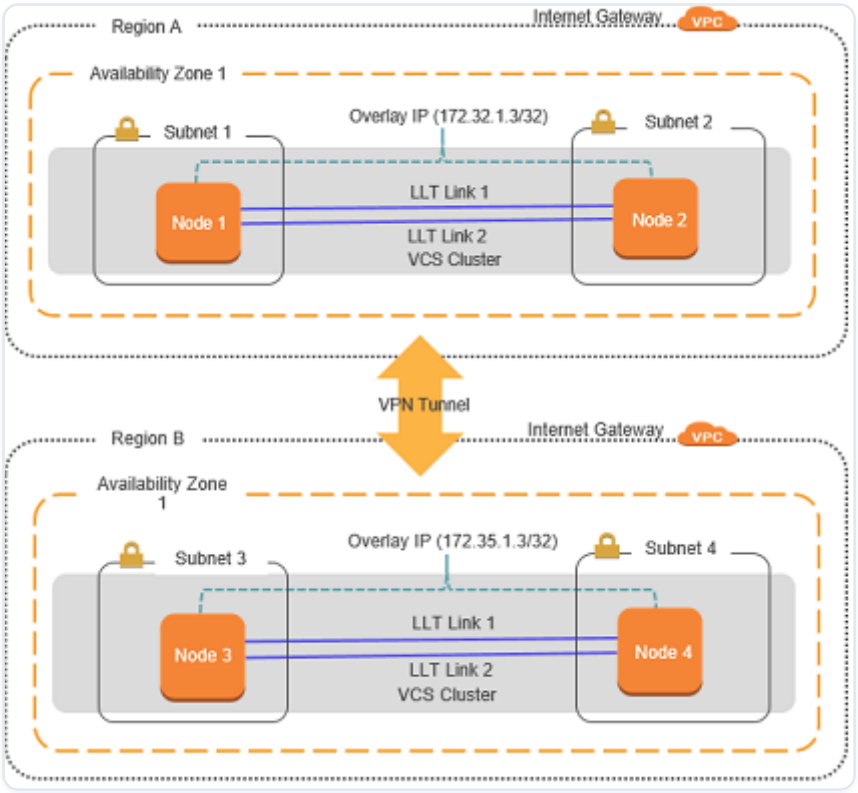
- VPN tunnel information between regions or VPCs
- The IP address to be used for cross-cluster communication:
    - Virtual private IP for cluster the nodes that exist in the same subnet
    - Overlay IP for cluster the nodes that exist in different subnets

You can also use GCO to configure applications for DR from an on-premises site to AWS.

> **Note:** If you use an Amazon VPN tunnel in a global cluster configuration between an on-premises site and AWS, the cluster nodes in the cloud must be in the same subnet.

Sample configuration for DR across regions or VPCs

The following graphic depicts a sample DR configuration across AWS regions:



The sample configuration includes the following elements:

- VPN tunnel between Region A and Region B
- The primary site has the following elements:
  - A virtual private cloud, VPC 1, is configured in Region A of the AWS cloud.
  - An application is configured for HA using an InfoScale cluster that comprises two nodes, Node 1 and Node 2, which are EC2 instances.

Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.

- The overlay IP allows the private IP of a node to be fail over from one subnet to another in an AZ during failover or failback.
- The secondary site has the following elements:
- A virtual private cloud, VPC 2, is configured in Region B of the AWS cloud.
- The same application is configured for HA on Node 3 and Node 4, which exist in Subnet 3 and Subnet 4 respectively.
- The overlay IP allows the private IP of a node to fail over from one subnet to another in an AZ.

Sample service group configuration for GCO across regions

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) at the primary site (Region A):

```
include "types.cf"



cluster sitever (

     ClusterAddress = "172.32.1.2"

  SecureClus = 1

     )



remotecluster sitecal (

     ClusterAddress = "172.35.1.2"

     ConnectTimeout = 3000

     SocketTimeout = 3000

     )



heartbeat Icmp (

     ClusterList = { sitecal }

     Arguments @sitecal = { "172.35.1.2" }

     )



system ip-172-31-21-156 (

     )



system ip-172-31-61-106 (
```

```
    )


group ClusterService (
    SystemList = { ip-172-31-21-156 = 0, ip-172-31-61-106 = 1 }
  AutoStartList = { ip-172-31-21-156, ip-172-31-61-106 }
    OnlineRetryLimit = 3
    OnlineRetryInterval = 120
    )


   AWSIP Aws_Ipres (
        OverlayIP = "172.32.1.2/32"
        Device = eth0
        AWSBinDir = "/usr/local/bin"
        )


   Application wac (
        StartProgram = "/opt/VRTSvcs/bin/wacstart -secure"
        StopProgram = "/opt/VRTSvcs/bin/wacstop"
        MonitorProcesses = { "/opt/VRTSvcs/bin/wac -secure" }
        RestartLimit = 3
        )
```

```
    IP Ipres (
         Device = eth0
         Address = "172.32.1.2"
         NetMask = "255.255.255.0"
         )

    NIC gconic (
         Device = eth0
         )

    Aws_Ipres requires Ipres
    Ipres requires gconic
    wac requires Ipres
```

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) at the secondary site (Region B):

```
include "types.cf"

cluster sitecal (
        ClusterAddress = "172.35.1.2"
        SecureClus = 1
        )
```

```
remotecluster sitever (
        ClusterAddress = "172.32.1.2"
        ConnectTimeout = 3000
        SocketTimeout = 3000
        )

heartbeat Icmp (
        ClusterList = { sitever }
        Arguments @sitever = { "172.32.1.2" }
        )

system ip-172-34-20-109 (
        )

system ip-172-34-30-231 (
        )

group ClusterService (
        SystemList = { ip-172-34-20-109 = 0, ip-172-34-30-231 = 1 }
        AutoStartList = { ip-172-34-20-109, ip-172-34-30-231 }
        OnlineRetryLimit = 3
        OnlineRetryInterval = 120
```

```
        )


AWSIP Aws_Ipres (
        OverlayIP = "172.35.1.2/32"
        Device = eth0
        AWSBinDir = "/usr/local/bin"
        )


Application wac (
        StartProgram = "/opt/VRTSvcs/bin/wacstart -secure"
        StopProgram = "/opt/VRTSvcs/bin/wacstop"
        MonitorProcesses = { "/opt/VRTSvcs/bin/wac -secure" }
        RestartLimit = 3
        )


IP Ipres (
        Device = eth0
        Address = "172.35.1.2"
        NetMask = "255.255.255.0"
        )


NIC gconic (
```

```
        Device = eth0

        )



Aws_Ipres requires Ipres

Ipres requires gconic

wac requires Ipres
```

DR from on-premises to AWS - Windows

InfoScale Enterprise lets you use the global cluster option (GCO) for DR configurations. You can use a DR configuration to fail over applications from on-premise cluster to AWS.

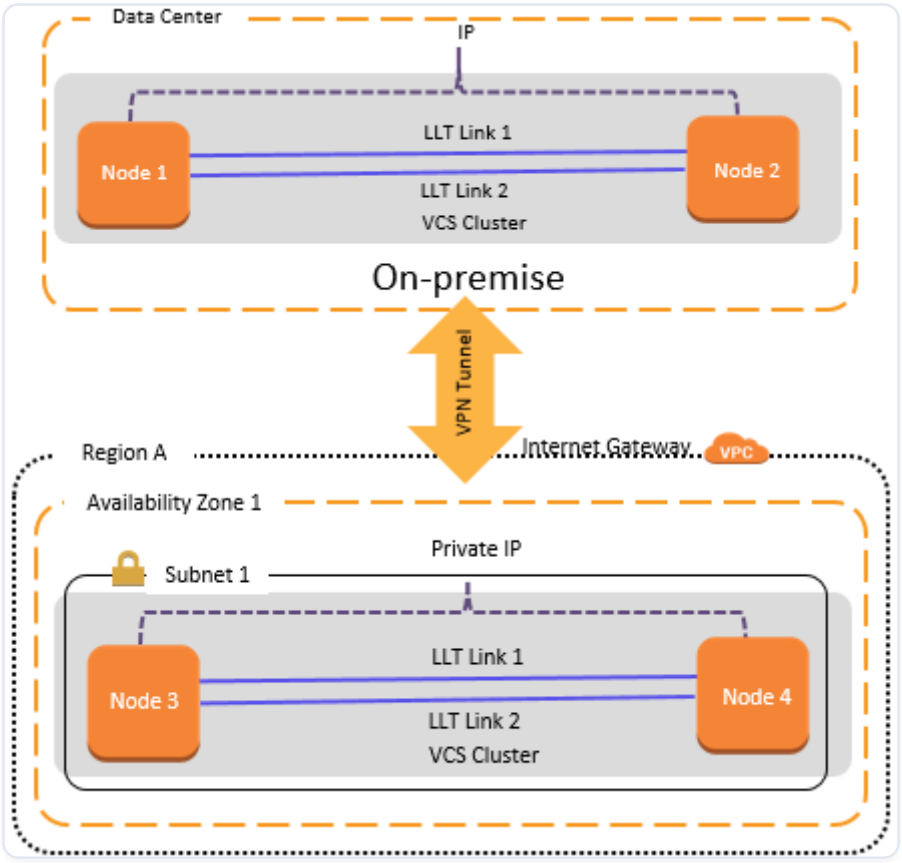
The following information is required:

- VPN tunnel information between regions or VPCs
- The Virtual private IP address to be used for cross-cluster communication.

> **Note:** If you use an Amazon VPN tunnel in a global cluster configuration between an on-premises site and AWS, the cluster nodes in the cloud must be in the same subnet.

Sample configuration for DR across regions or VPCs

The following graphic depicts a sample DR configuration from on-premise cluster to AWS cloud:



The sample configuration includes the following elements:

- VPN tunnel between on-premise data center and Region A
- The primary site has the following elements:
  - Cluster nodes in the same subnet
  - Virtual private IP for cross-cluster communication
- The secondary site has the following elements:
  - A virtual private cloud, VPC, is configured in Region Aof the AWS cloud
  - The same application is configured for HA on Node 3 and Node 4, which exist in the same subnet

Sample service group configuration for GCO from on-premise to cloud

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) for a single node:

```
include "types.cf"
```

```
cluster vvr_aws (
          ClusterAddress = "10.239.3.96"
          SecureClus = 1
          )


remotecluster vvr_cloud (
          ClusterAddress = "10.209.57.98"
          ConnectTimeout = 30000
          SocketTimeout = 30000
          )


heartbeat Icmp (
          ClusterList = { vvr_cloud }
          AYARetryLimit = 10
          Arguments @vvr_cloud = { "10.209.57.98" }
          )


system WIN-38PNEVJSR2K (
          )


group ClusterService (
          SystemList = { WIN-38PNEVJSR2K = 0 }
```

```
        AutoStartList = { WIN-38PNEVJSR2K }

        )



        AWSIP gcoawsip (

                PrivateIP = "10.239.3.96"

                Device = 12-7F-CE-5B-E2-6E

                )



        IP csg_ip (

                Address = "10.239.3.96"

                SubNetMask = "255.255.254.0"

                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"

                )



        NIC csg_nic (

                MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"

                )



        Process wac (

                StartProgram @WIN-38PNEVJSR2K = "\"C:\\Program Files\\


               Veritas\\Cluster Server\\bin\\wac.exe\""
```

```
                    StopProgram @WIN-38PNEVJSR2K = "\"C:\\Program Files\\

                  Veritas\\Cluster Server\\bin\\wacstop.exe\""

                    MonitorProgram @WIN-38PNEVJSR2K = "\"C:\\Program Files\
\

                Veritas\\Cluster Server\\bin\\wacmonitor.exe\""

                     )


           csg_ip requires csg_nic

           wac requires gcoawsip

           wac requires csg_ip


group fileshare (

           SystemList = { WIN-38PNEVJSR2K = 0 }

           ClusterList = { vvr_cloud = 1, vvr_aws = 0 }

           Authority = 1

           ClusterFailOverPolicy = Auto

           )


           AWSIP vvrawsip (

                    PrivateIP = "10.239.3.97"

                    Device @WIN-38PNEVJSR2K = 12-7F-CE-5B-E2-6E
```

```
        )


FileShare fileshare-FileShare (
        PathName = "\\"
        ShareName = testshare
        LanmanResName = fileshare-Lanman
        MountResName = mounvres
        UserPermissions = { Everyone = FULL_CONTROL }
        )


IP fileshare-IP (
        Address = "10.239.3.97"
        SubNetMask = "255.255.254.0"
        MACAddress @WIN-38PNEVJSR2K = 12-7F-CE-5B-E2-6E
        )


Lanman fileshare-Lanman (
        VirtualName = FILESHARE-DHAWA
        IPResName = fileshare-IP
        )


MountV mounvres (
```

```
            MountPath = "e:"
            VolumeName = datavol
            VMDGResName = vmnsdg
            )


        NIC fileshare_NIC (
            MACAddress @WIN-38PNEVJSR2K = "12:7F:CE:5B:E2:6E"
            )


        RVGPrimary rvgprimary_1 (
            RvgResourceName = vvrsg-VvrRvg
            AutoResync = 1
            )


        requires group vvrsg online local hard
        vvrawsip requires fileshare_NIC
        fileshare-FileShare requires mounvres
        fileshare-FileShare requires fileshare-Lanman
        fileshare-IP requires vvrawsip
        fileshare-Lanman requires fileshare-IP
        mounvres requires rvgprimary_1
```

```
group vvrsg (
        SystemList = { WIN-38PNEVJSR2K = 0 }
        AutoStartList = { WIN-38PNEVJSR2K }
        )


        Proxy proxy_ip (
                TargetResName = csg_ip
                )


        Proxy proxy_nic (
                TargetResName = csg_nic
                )


        VMNSDg vmnsdg (
                DiskGroupName = vvrdg
                DGGuid = 03d725b1-fca3-49b2-b722-38984835e6b0
                )


        VvrRvg vvrsg-VvrRvg (
                RVG = rvg
                VMDgResName = vmnsdg
                IPResName = proxy_ip
```

```
                )


        vvrsg-VvrRvg requires proxy_nic

        vvrsg-VvrRvg requires proxy_ip

        vvrsg-VvrRvg requires vmnsdg
```

Configurations for Microsoft Azure - Linux

This section includes the following topics:

- Replication configurations in Azure - Linux
- HA and DR configurations in Azure - Linux

Replication configurations in Azure - Linux

The steps for setting up replication in Azure depends on where your primary data center and your secondary data center are located.

Follow the procedure that is appropriate for your setup:

- See Replication from on-premises to Azure - Linux.
- See Replication within an Azure region - Linux.
- See Replication across Azure regions - Linux.
- See Replication across multiple Azure sites and regions (campus cluster) - Linux.

Additionally, as part of each of these procedures, you need to identify a temporary resource disk to be used as swap space.

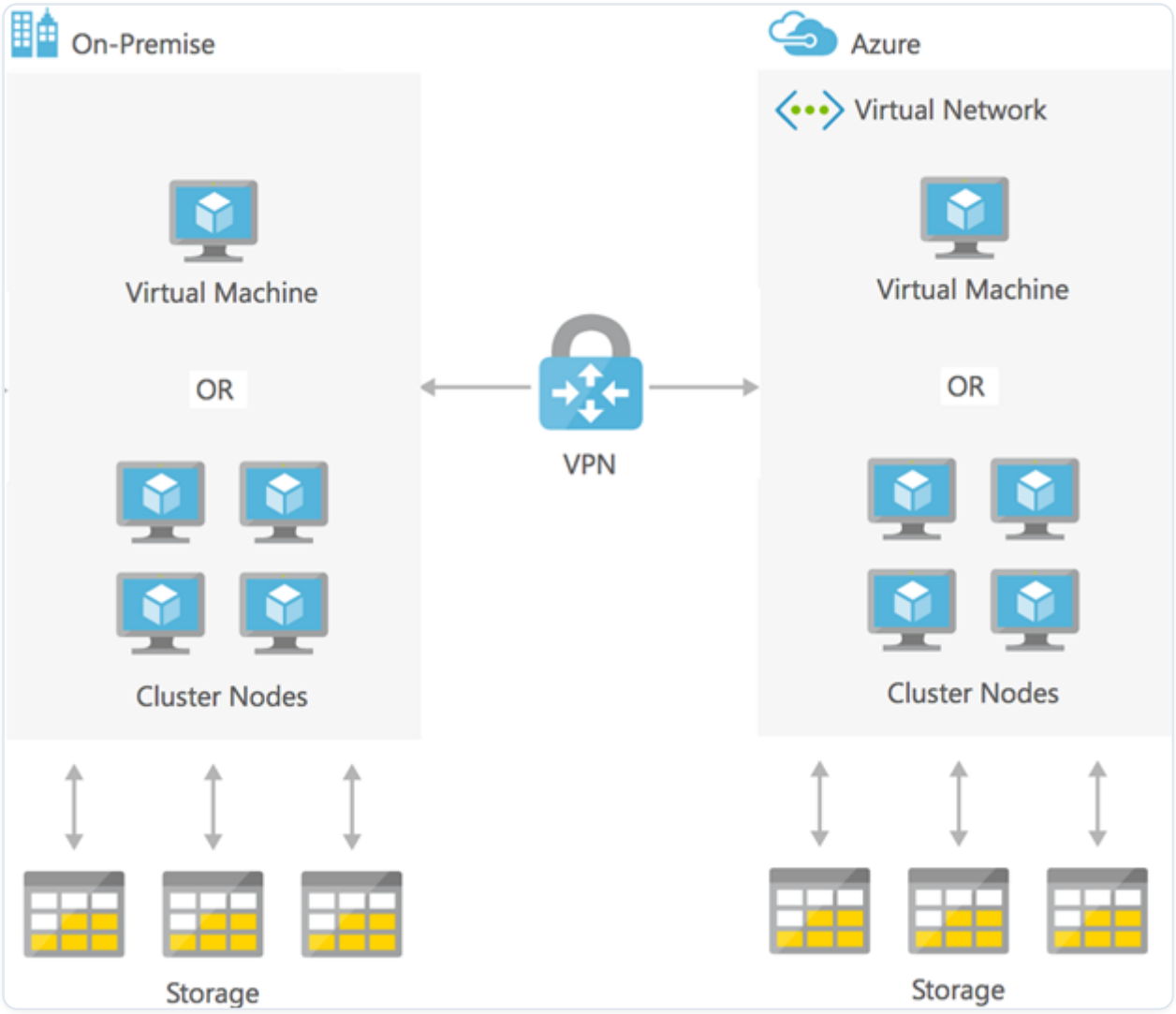
See About identifying a temporary resource disk - Linux.

Replication from on-premises to Azure - Linux

The following diagram illustrates the sample configuration for setting up replication between an on-premise data center to Azure cloud (on-cloud data center):

> **Note:** For ease of use, the machines, whether virtual or physical are commonly mentioned as virtual machines. Your on-premise data center may include physical machines instead of the virtual machines. In any case, the steps to set up replication from an on-premise data center to an on-cloud data center remain the same for physical and well as virtual machines.

> Figure: Sample configuration for setting up replication between an on-premise data center to on-cloud data center



About setting up replication between an on-premise data center to on-cloud data center

Replication between an on-premise data center to on-cloud data center involves the following high-level steps:

1. Prepare the setup at on-premise data center
2. Prepare the setup at on-cloud data center
3. Establish a tunnel from on-premise data center to cloud data center
4. Deploy setup

The following sections provide details about performing each of these steps.

Preparing the setup at on-premise data center

Perform the following steps to prepare the setup at on-premise data center:

1. Enable the ports that are used for inbound and outbound communication.

For a list of required ports and services, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Create a subnet and a local VPN gateway.
2. Note the address space that is allotted for the subnet and the public IP address that is allotted for the local VPN gateway.

Preparing the setup at on-cloud data center

Perform the following steps to prepare the setup at on-cloud data center:

1. Using Microsoft Azure portal, create a resource group.
2. Create a VNet in the resource group created and specify an IP address space for the VNet.

The IP address range must be diff on the on-premise subnet and on the on-cloud subnet.

1. Create a gateway subnet.
2. Create a VPN gateway and associate it with the created VNet.

Note the public IP address that is allotted for the on-cloud VPN gateway.

1. Create a local network gateway.

When you create the local network gateway, you must provide the on-premise subnet IP address range and the public IP address of the on-premise local VPN gateway.

1. Establish a tunnel from on-cloud to on-premise network.

To establish the tunnel, create a connection of type **Site-to-Site (IPSec)** and choose the on-cloud VPN gateway and the local network gateway.

1. Provide a shared key (alpha-numeric key).

A shared key is a pass-phrase. This pass-phrase is required when you establish a tunnel from on-premise data center to cloud data center.

Establishing a tunnel from on-premise data center to on-cloud data center

To establish a tunnel from on-premise data center to on-cloud data center, use the following parameters:

- Public IP address that is allotted for the on-cloud VPN gateway

- Shared key (alpha-numeric key) that was provided while establishing a tunnel from on-cloud data center to on-premise data center
- On-cloud VPN gateway configuration type (Policy based or Route based)

Deploying the setup

To deploy the setup (in both the data centers)

1. Create virtual machines in the subnets created.
2. Provision storage.
3. Install the appropriate InfoScale product.
4. Create VxVM disk groups, VxVM volumes, Storage Replicator Log (SRL), Replicated Volume Group (RVG), and RLinks.

For details, refer to the *InfoScale Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

> **Note:** In Azure environment, by default, in addition to the storage disks that you have attached, every virtual machine that is provisioned contains a temporary resource disk, that serves as an ephemeral storage. Do not use the temporary resource as a data disk (VxVM disk) to store persistent data. The disk may change after a machine is redeployed or is restarted, and the data will be lost. For more information about how Azure uses a temporary disk, see Microsoft documentation. For details about how to identify a temporary resource disk: SeeAbout identifying a temporary resource disk - Linux.

1. Flush the iptables on both the virtual machines.

```
# iptable -F
```

1. Set up replication between the virtual machines using the private IP address or the virtual IP address.

For details on setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

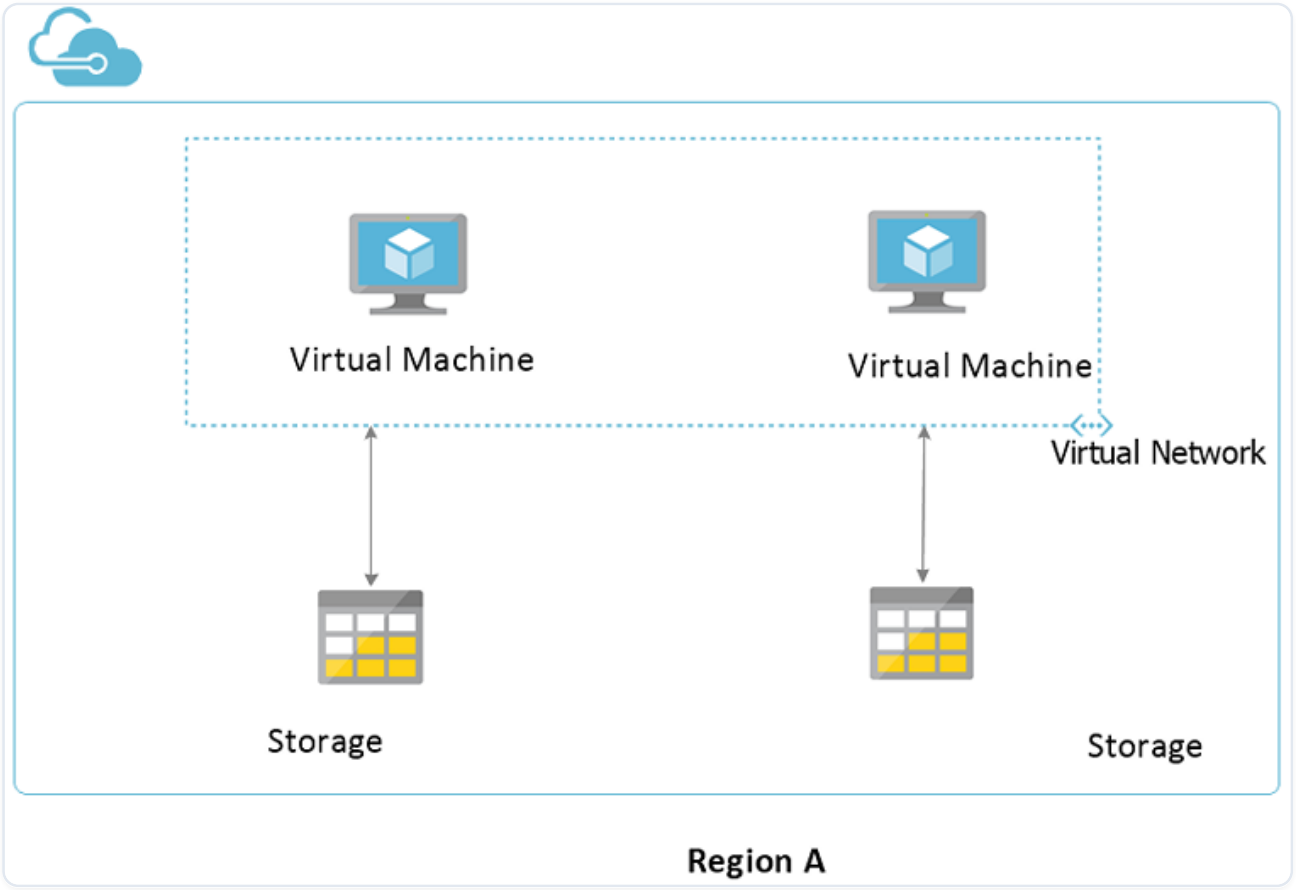
```
Replication status: replicating (connected)
```

Replication within an Azure region - Linux

In an Azure cloud environment, in a single region, you can provision your setup across virtual networks (VNets) or within a VNet.

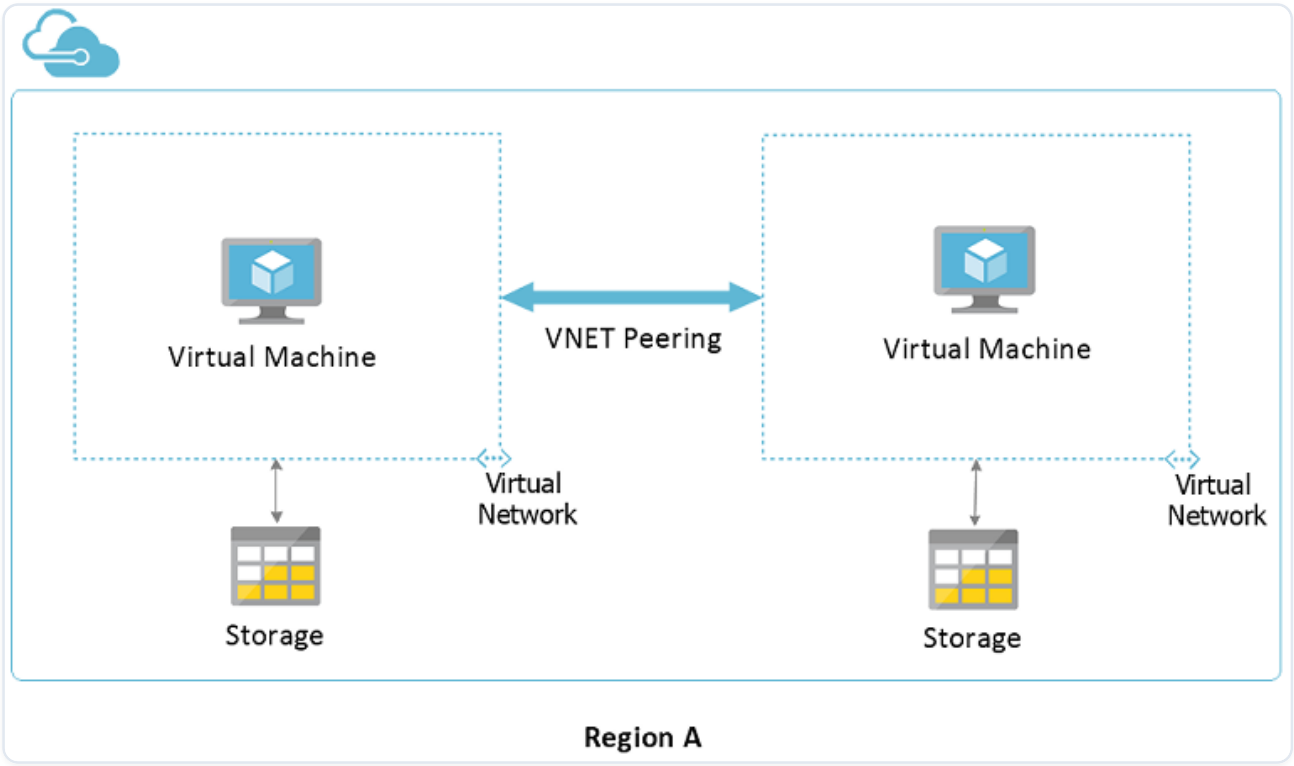
The following diagram illustrates the sample configuration for setting up replication between the same VNet:

Figure: Sample configuration for setting up replication between same VNet



The following diagram illustrates the sample configuration for setting up replication across VNets:

Figure: Sample configuration for setting up replication across VNets



Setting up replication in the same VNet

Perform the following steps to set up replication in the same VNet, within the same region

1. Enable the ports that are used for inbound and outbound communication.

For a list of required ports and services, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Using Microsoft Azure portal, create a VNet and specify an IP address space for the VNet.
2. Create a subnet in the VNet created.

For details about creating a VNet, specifying an IP address space, and creating a subnet, refer to Microsoft documentation.

1. Create two virtual machines within the subnets and provision storage.
2. Install the appropriate InfoScale product on both the virtual machines.
3. Create VxVM disk groups, VxVM volumes, Storage Replicator Log (SRL), Replicated Volume Group (RVG), and RLinks.

For details, refer to the *InfoScale Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

> **Note:** In Azure environment, by default, in addition to the storage disks that you have attached, every virtual machine that is provisioned contains a temporary resource disk, that

> serves as an ephemeral storage. Do not use the temporary resource as a data disk (VxVM disk) to store persistent data. The disk may change after a machine is redeployed or is restarted, and the data will be lost. For more information about how Azure uses a temporary disk, see Microsoft documentation. For details about how to identify a temporary resource disk: SeeAbout identifying a temporary resource disk - Linux.

1. Flush the iptables on both the virtual machines.

```
# iptable -F
```

1. Set up replication between the virtual machines using the private IP address or the virtual IP address.

For details on setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

Setting up replication across VNet

Perform the following steps to set up replication across the VNets, within the same region

1. Using Microsoft Azure portal, create two VNets and specify an IP address space for each VNet.
2. Set up VNet Peering between the two VNets.
3. Create a subnet in each VNet.
4. Create a virtual machine in each subnet and provision storage.
5. Install the appropriate InfoScale product on both the virtual machines.
6. Create VxVM disk groups, VxVM volumes, Storage Replicator Log (SRL), Replicated Volume Group (RVG), and RLinks.

For details, refer to the *InfoScale Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

> **Note:** In Azure environment, by default, in addition to the storage disks that you have attached, every virtual machine that is provisioned contains a temporary resource disk, that serves as an ephemeral storage. Do not use the temporary resource as a data disk (VxVM disk) to store persistent data. The disk may change after a machine is redeployed or is restarted, and the data will be lost. For more information about how Azure uses a temporary disk, see Microsoft documentation. For details about how to identify a temporary resource disk: SeeAbout identifying a temporary resource disk - Linux.

1. Flush the iptables on both the virtual machines.

```
# iptable -F
```

1. Set up replication between the virtual machines using the private IP address or the virtual IP address.

For details on setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

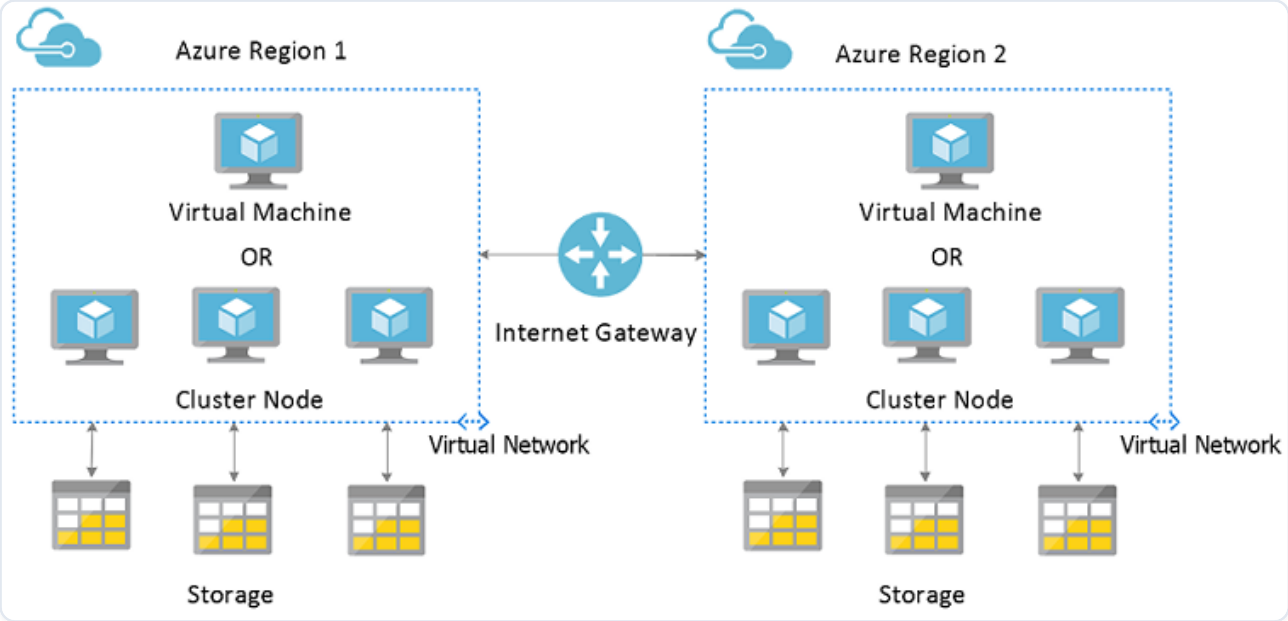
Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

Replication across Azure regions - Linux

The following diagram illustrates the sample configuration for setting up replication across regions:

Figure: Sample configuration for setting up replication across regions



Setting up replication across regions

Perform the following steps to set up replication across region

1. Enable the ports that are used for inbound and outbound communication.

For a list of required ports and services, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Using Azure portal, create a resource group (RG) in both the regions.
2. Create a VNet in each region and specify a non-overlapping IP address space for each VNet.
3. Create a subnet and a gateway subnet under the VNets created in both the regions.
4. Create a virtual network gateway in both the regions.
5. Establish a connection in between the two virtual network gateways.
6. Create virtual machines in the subnets and provision storage.
7. Install the appropriate InfoScale product.
8. Create VxVM disk groups, VxVM volumes, Storage Replicator Log (SRL), Replicated Volume Group (RVG), and RLinks.

For details, refer to the *InfoScale Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*

> **Note:** In Azure environment, by default, in addition to the storage disks that you have attached, every virtual machine that is provisioned contains a temporary resource disk, that serves as an ephemeral storage. Do not use the temporary resource as a data disk (VxVM

> disk) to store persistent data. The disk may change after a machine is redeployed or is restarted, and the data will be lost. For more information about how Azure uses a temporary disk, see Microsoft documentation. For details about how to identify a temporary resource disk: SeeAbout identifying a temporary resource disk - Linux.

1. Flush the iptables on both the virtual machines.

```
# iptable -F
```

1. Set up replication using the private IP address or the virtual IP address.

For details on setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

Replication across multiple Azure sites and regions (campus cluster) - Linux

To set up replication across regions, in a campus cluster, perform the following steps at both the regions:

1. Enable the ports that are used for inbound and outbound communication.

For a list of required ports and services, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Using Azure portal, create a resource group (RG) in both the regions.
2. Create a VNet in each region and specify a non-overlapping IP address space for each VNet.
3. Create a subnet and a gateway subnet under the VNets created in both the regions.
4. Create a virtual network gateway in both the regions.
5. Establish a connection in between the two virtual network gateways.
6. Create virtual machines in the subnets and provision storage.
7. Install the appropriate InfoScale product.

8. Create FSS disk groups, VxVM volumes, Storage Replicator Log (SRL), Replicated Volume Group (RVG), and RLinks.

For details, refer to the *InfoScale Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

> **Note:** In Azure environment, by default, in addition to the storage disks that you have attached, every virtual machine that is provisioned contains a temporary resource disk, that serves as an ephemeral storage. Do not use the temporary resource as a data disk (VxVM disk) to store persistent data. The disk may change after a machine is redeployed or is restarted, and the data will be lost. For more information about how Azure uses a temporary disk, see Microsoft documentation. For details about how to identify a temporary resource disk: SeeAbout identifying a temporary resource disk - Linux.

1. Create clusters in both the regions.
2. Flush the iptables on all the cluster nodes.

```
# iptable -F
```

1. Set up replication using the private IP address or the virtual IP address.

For details on setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

About identifying a temporary resource disk - Linux

Typically, a temporary resource disk is named as /dev/sdb and is mounted at /mnt. However, the location may change depending on whether or not it is utilized for swap space or is unmounted by a user.

To identify whether or not swap space is configured and a disk utilized for swap space

1. Identify the swap space configuration and check the swap file:

```
# swapon
```

Following is the sample output of this command:

```
NAME TYPE SIZE USED PRIO /mnt/resource/swapfile file 2G 0B -1
```

Where, /mnt/resource is the default location where a temporary disk is mounted when used for swap space.

1. Identify the disk used for swap space:

```
# mount | grep "/mnt/resource"
```

Following is the sample output of this command:

```
/dev/sdb on /mnt/resource type filesystem
```

Where /dev/sdb is the temporary disk.

1. Identify a VxVM disk that corresponds to a temporary disk.

```
# vxdisk -e list | grep sdb (If sdb is the OS device name for temporary disk found in the earlier step)
```

Following is the sample output for the command:

```
10-0-15-6_disk_490 auto:none - - online invalid sdb -
```

HA and DR configurations in Azure - Linux

InfoScale Enterprise lets you configure applications for high availability (HA) and disaster recovery (DR) in Azure cloud environment.

For more information on the supported use cases, read the following topics:

- See Failover within an Azure subnet using private IP - Linux.
- See Failover across Azure subnets using overlay IP - Linux.
- See Public access to cluster nodes in Azure using public IP - Linux.
- See DR from on-premises to Azure and across Azure regions or VNets - Linux.

InfoScale Enterprise provides the following agents to manage Azure resources:

- The AzureIP agent and the AzureDNSZone agent manage the network resources for cluster communication, failover, and failback. These agents monitor and manage the IP resources within cluster nodes in the Azure cloud.
- The following options are available for handling authentication for Azure subscription:
    - You can configure the AzureAuth agent to use the service principal credentials for authentication. The AzureIP, AzureDNSZone, and the AzureDisk agents then depend on the AzureAuth agent for handling the Azure related authentication.
    - You can also use the Azure user-assigned managed identities for authentication. You can configure the `ManagedIdentityClientID` attribute for the AzureIP, AzureDNSZone, and the AzureDisk agents to use managed identities for the Azure related authentication.

In this case, these agents no longer depend on the AzureAuth agent; you do not have to configure the AzureAuth agent resources in the cluster.

> **Note:** Acrtera recommends that you configure the agents to use managed identities, instead of the AzureAuth agent, for authenticating the Azure subscription.

> For more information about the agents, refer to the *InfoScale Cluster Server Bundled Agents Reference Guide - Linux.*

- In an Azure environment, application data is stored on the Azure data disks that are attached to the Azure virtual machines. The AzureDisk agent provides high availability of the Azure disks during an application failover. The agent supports high availability for managed and unmanaged disks.

    > **Note:** Azure recommends that you configure Azure VMs in the same Availability Set to ensure that at least one node from the cluster is always operational.

InfoScale lets you configure replication between Azure virtual machines, which is further used to support various HA and DR scenarios for applications in Azure cloud.

For more information on the supported replication configurations in Azure, refer to the following documents:

- *InfoScale Replication Administrator's Guide - Linux*
- *InfoScale Disaster Recovery Implementation Guide - Linux*

Failover within an Azure subnet using private IP - Linux

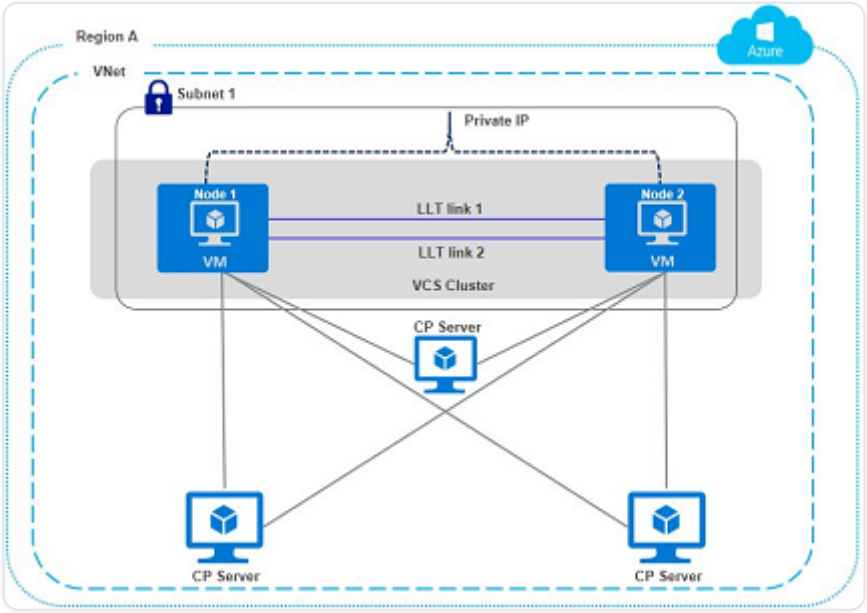
InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - within the same subnet in the same VNet.

The following information is required:

- A private IP (secondary private IP) address to be failed over
- The device to which the IP should be plumbed

Sample configuration with private IP

The following graphic depicts a sample failover configuration within the same subnet using a private IP:



The sample configuration includes the following elements:

- A Azure virtual network (VNet) is configured in Region A of the Azure cloud
- An application is configured for HA using a cluster that comprises two nodes, Node1 and Node2, which are Azure virtual machines
- Both the cluster nodes exist in the same subnet
- A private IP is configured, which is failed over from one node to the other as part of the failover or the failback operations

Sample service group configuration with private IP

The following snippet is a service group configuration from a sample configuration file (main.cf):

Sample configuration using AzureAuth agent

```
group AzureAuthGrp (

        SystemList = { azureVM1 = 0, azureVM2 = 1 }

        Parallel = 1

        )



        AzureAuth azurauth (

                SubscriptionId = 6940a326-abg6-40dd-b628-c1e9bbdf1d63

                ClientId = 8c891a8c-xyz2-473b-bigc-035bd50fb896

                SecretKey =
gsiOssRooSnoJuiOhmOumShuNoiQioNsjQlqHovUosQsrMt

                TenantId = 96dcasae-0448-4308-b503-6667d61dd0e3

                )



        Phantom phres (

                )



group AzurePrivateIP (

        SystemList = { azureVM1 = 0, azureVM2 = 1 }

        AutoStartList = { azureVM1, azureVM2 }

                )



               AzureIP azureipres (
```

```
                PrivateIP = "10.3.3.100"
                NICDevice = eth0
                VMResourceGroup = ShilRG
                AzureAuthResName = azurauth
                )


        IP ipres (
                Device = eth0
                Address = "10.3.3.100"
                NetMask = "255.255.255.0"
                )


        NIC nicres (
                Device = eth0
                )


        ipres requires azureipres
        ipres requires nicres
```

Sample configuration using a managed identity

```
group AzurePrivateIP (
SystemList = { azureVM1 = 0, azureVM2 = 1 }
```

```
AutoStartList = { azureVM1, azureVM2 }

)



AzureIP azureipres (

PrivateIP = "10.3.3.100"

NICDevice = eth0

VMResourceGroup = ShilRG

ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)



IP ipres (

Device = eth0

Address = "10.3.3.100"

NetMask = "255.255.255.0"

)



NIC nicres (

Device = eth0

)



ipres requires azureipres
```

```
ipres requires nicres
```

Failover across Azure subnets using overlay IP - Linux

InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - between different subnets in the same VNet.

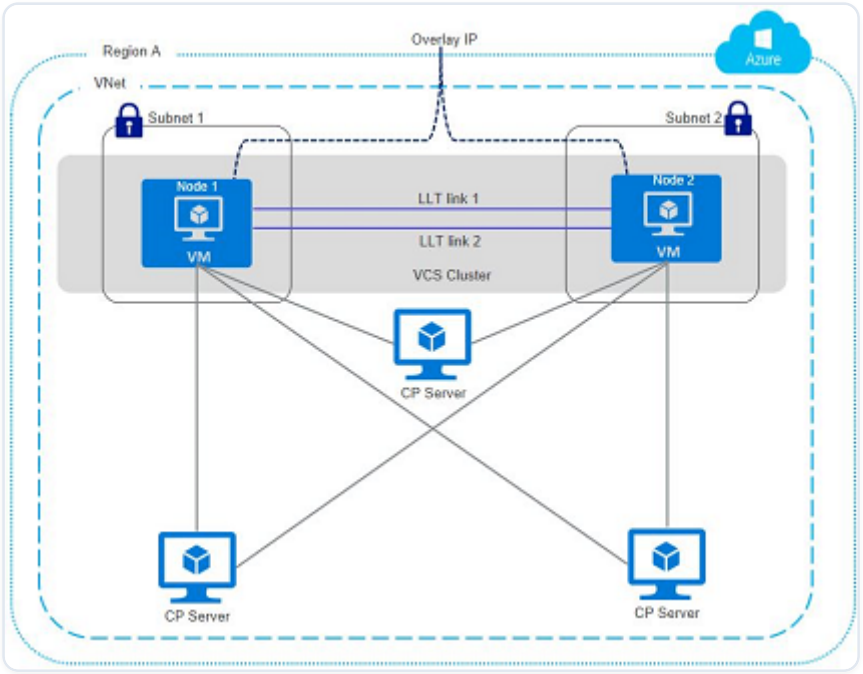
The following information is required:

- The IP address outside VNet to be used for failover
- The device to which the IP should be plumbed

Azure does not allow the private IP of one subnet to be failed over to a different subnet. To overcome this limitation, provide an overlay IP, which is outside the VNet level, so that it can be used across subnets.

Sample configuration with Overlay IP for failover across subnets in the same VNet

The following graphic depicts a sample failover configuration across subnets within the same VNet using an overlay IP:



The sample configuration includes the following elements:

- A Azure virtual network (VNet) is configured in Region A of the Azure cloud
- An application is configured for HA using a cluster that comprises two nodes, Node1 and Node2, which are Azure virtual machines

- Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.
- An overlay IP is configured to allow redirecting IP address traffic to another cluster node belonging to different subnet within the same VNet as a part of the failover or the failback operations

Sample service group configuration with overlay IP for failover across subnets in the same VNet

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

Sample configuration using AzureAuth agent

```
group AzureAuthGrp (

        SystemList = { azureVM1 = 0, azureVM2 = 1 }

        Parallel = 1

        )



        AzureAuth azurauth (

                SubscriptionId = 6940a326-abg6-40dd-b628-c1e9bbdf1d63

                ClientId = 8c891a8c-xyz2-473b-bigc-035bd50fb896

                SecretKey =
gsiOssRooSpsPotQkmOmmQpuNoiQioNsjQlqHovUosQsrMt

                TenantId = 96dcasae-0448-4308-b503-6667d61dd0e3

                )



        Phantom phres (

                )
```

```
group AzureOverlayIP (
        SystemList = { azureVM1 = 0, azureVM2 = 1 }
        AutoStartList = { azureVM1, azureVM2 }
        )


        AzureIP overlayipres (
                NICDevice = eth0
                VMResourceGroup = ShilRG
                OverlayIP = "192.168.99.99"
                RouteTableResourceIds = {
                     "/subscriptions/6940a326-abg6-40dd-b628-c1e9bbdf1d63/

          resourceGroups/ShilRG/providers/Microsoft.Network/

          routeTables/shilroute",
                     "/subscriptions/6940a326-abg6-40dd-b628-c1e9bbdf1d63/

          resourceGroups/SHILRG1/providers/Microsoft.Network/

                          routeTables/shilroute_eastUS2" }
                AzureAuthResName = azurauth
                )


        IP ipres (
```

```
                Device = eth0
                Address = "192.168.99.99"
                NetMask = "255.255.255.255"
                )

        NIC nicres (
                Device = eth0
                )

        ipres requires overlayipres
        ipres requires nicres
```

Sample configuration using a managed identity

```
group AzureOverlayIP (
SystemList = { azureVM1 = 0, azureVM2 = 1 }
AutoStartList = { azureVM1, azureVM2 }
)

AzureIP overlayipres (
NICDevice = eth0
VMResourceGroup = ShilRG
OverlayIP = "192.168.99.99"
```

```
RouteTableResourceIds = {

"/subscriptions/6940a326-abg6-40dd-b628-c1e9bbdf1d63/

resourceGroups/ShilRG/providers/Microsoft.Network/

routeTables/shilroute",

"/subscriptions/6940a326-abg6-40dd-b628-c1e9bbdf1d63/

resourceGroups/SHILRG1/providers/Microsoft.Network/

routeTables/shilroute_eastUS2" }

ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)



IP ipres (

Device = eth0

Address = "192.168.99.99"

NetMask = "255.255.255.255"

)



NIC nicres (

Device = eth0

)

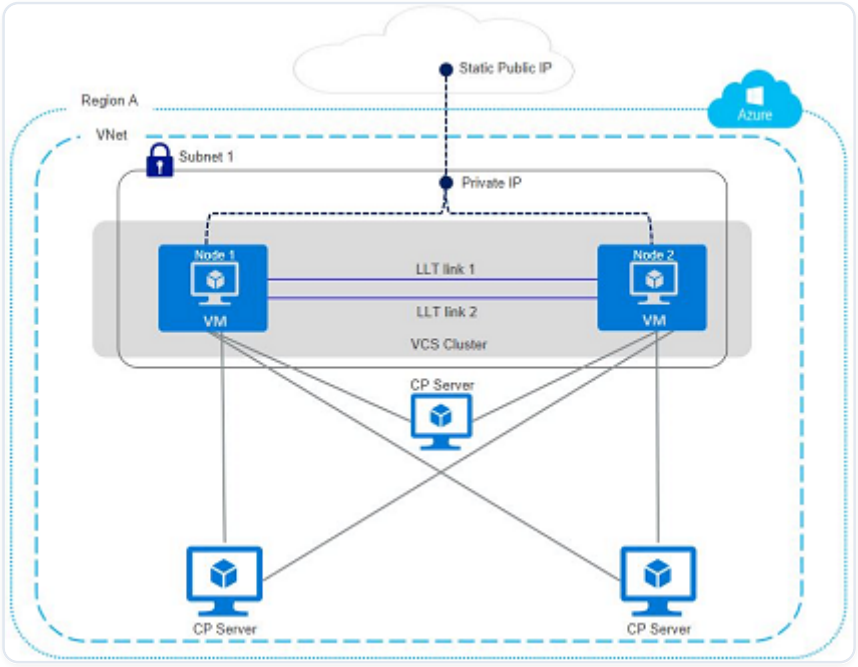


ipres requires overlayipres
```

```
ipres requires nicres
```

Public access to cluster nodes in Azure using public IP - Linux

To allow public access to a cluster node or to an application configured for HA in Azure environment, specify the IP in the PublicIP attribute for the AzureIP resource. This IP is used to map the Public IP address to a secondary private IP address. For example, if you have an application that needs to be highly available and to be accessible globally, you can use the PublicIP attribute of the AzureIP agent to ensure both.



Sample service group configuration with Public IP

The following snippet is a service group configuration from a sample configuration file (main.cf):

Sample configuration using AzureAuth agent

```
group AzureAuthGrp (

                        SystemList = { azureVM1 = 0, azureVM2 = 1 }

                        Parallel = 1

                        )



AzureAuth azurauth (
```

```
                    SubscriptionId = 6940a326-abg6-40dd-b628-c1e9bbdf1d63
                    ClientId = 8c891a8c-xyz2-473b-bigc-035bd50fb896
                    SecretKey = gsiOssRooSpsPotQkmOmiQioNsjQlqHovUosQsrMt
                    TenantId = 96dcasae-0448-4308-b503-6667d61dd0e3
                    )
                Phantom phres (
                )

group AzurePrivateIP (
                SystemList = { azureVM1 = 0, azureVM2 = 1 }
                AutoStartList = { azureVM1, azureVM2 }
                )
                AzureIP azureipres (
                PrivateIP = "10.3.3.100"
                NICDevice = eth0
             PublicIP = "40.71.178.90"
                AzureAuthResName = azurauth
                )
IP ipres (
            Device = eth0
            Address = "10.3.3.100"
            NetMask = "255.255.255.0"
```

```
            )
NIC nicres (
            Device = eth0
            )
            ipres requires azureipres
            ipres requires nicres
```

Sample configuration using a managed identity

```
group AzurePrivateIP (
SystemList = { azureVM1 = 0, azureVM2 = 1 }
AutoStartList = { azureVM1, azureVM2 }
)


AzureIP azureipres (
PrivateIP = "10.3.3.100"
NICDevice = eth0
PublicIP = "40.71.178.90"
ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022
)


IP ipres (
Device = eth0
```

```
Address = "10.3.3.100"

NetMask = "255.255.255.0"

)



NIC nicres (

Device = eth0

)



ipres requires azureipres

ipres requires nicres
```

DR from on-premises to Azure and across Azure regions or VNets - Linux

VCS lets you use the global cluster option (GCO) for DR configurations. You can use a DR configuration to fail over applications across different regions or VNets in Azure or between an on-premises site and Azure.

The following is required for on-premise to cloud DR using VPN tunneling:

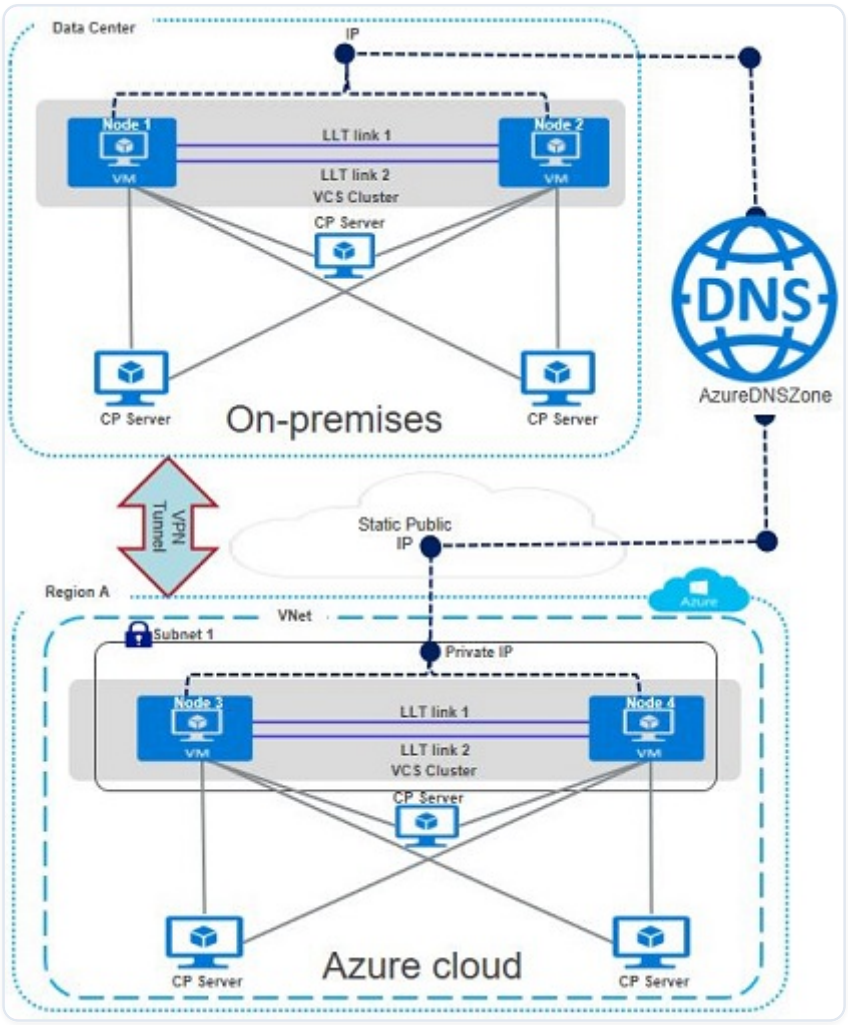
- Prepare the setup at on-premise data center
- Prepare the setup at on-cloud data center
- Establish a VPN tunnel from on-premise data center to cloud data center
- Virtual private IP for cluster nodes that exist in the same subnet. The IP address is used for cross-cluster communication

The following is required for region to region DR using VNet peering:

- Prepare the setup at both the data centers in the regions
- Establish a VNet peering from one region to the other region
- Virtual private IP for cluster nodes that exist in the same subnet. The IP address is used for cross-cluster communication

> **Note:** If you use an VPN tunnel between an on-premises site and Azure or you use VNet peering between Azure regions, the cluster nodes in the cloud must be in the same subnet.

Sample configuration with private IP



The sample configuration includes the following elements:

- VPN tunnel between on-premise data center and Region A
- The primary site has the following elements:
  - Cluster nodes in the same subnet
  - Fencing is configured using CP servers or disk-based I/O fencing
  - Virtual private IP for cross-cluster communication
- The secondary site has the following elements:
  - A VNet is configured in Region A of the Azure cloud

- The same application is configured for HA on Node 3 and Node 4, which exist in Subnet
- Fencing is configured using CP servers
- Virtual private IP for cross-cluster communication

Sample service group configuration with private IP

The following snippet is a service group configuration from a sample configuration file (main.cf):

Sample configuration using AzureAuth agent

```
cluster shil-sles11-clus1-eastus (

        ClusterAddress = "10.3.3.100"

        SecureClus = 1

        )



remotecluster shil-sles11-clus2-eastus2 (

        ClusterAddress = "10.5.0.5"

        )



heartbeat Icmp (

        ClusterList = { shil-sles11-clus2-eastus2 }

        Arguments @shil-sles11-clus2-eastus2 = { "10.5.0.5" }

        )



system azureVM1 (

        )
```

```
system azureVM2 (

        )



group AzureAuthGrp (

        SystemList = { azureVM1 = 0, azureVM2 = 1 }

        Parallel = 1

        )



        AzureAuth azurauth (

                SubscriptionId = 6940a326-abg6-40dd-b628-c1e9bbdf1d63

                ClientId = 8c891a8c-xyz2-473b-bigc-035bd50fb896

                SecretKey =
gsiOssRooSpsPotQkmOmmShuNoiQioNsjQlqHovUosQsrMt

                TenantId = 96dcasae-0448-4308-b503-6667d61dd0e3

                )



        Phantom phres (

                )



group ClusterService (

        SystemList = { azureVM1 = 0, azureVM2 = 1 }

        AutoStartList = { azureVM1, azureVM2 }
```

```
        OnlineRetryLimit = 3
        OnlineRetryInterval = 120
        )


        Application wac (
                StartProgram = "/opt/VRTSvcs/bin/wacstart -secure"
                StopProgram = "/opt/VRTSvcs/bin/wacstop"
                MonitorProcesses = { "/opt/VRTSvcs/bin/wac -secure" }
                RestartLimit = 3
                )


        AzureIP azureipres (
                PrivateIP = "10.3.3.100"
                NICDevice = eth0
                VMResourceGroup = ShilRG
                AzureAuthResName = azurauth
                )


        IP gcoip (
                Device = eth0
                Address = "10.3.3.100"
                NetMask = "255.255.255.0"
```

```
                )


        NIC gconic (
                Device = eth0
                )


        gcoip requires azureipres
        gcoip requires gconic
        wac requires gcoip


 group VVR (
        SystemList = { azureVM1 = 0, azureVM2 = 1 }
        AutoStartList = { azureVM1, azureVM2 }
        )


        AzureDisk diskres (
                DiskIds = {
                        "/subscriptions/6940a326-abg6-40dd-b628-
c1e9bbdf1d63/

                                resourceGroups/SHILRG/providers/
Microsoft.Compute/

                                disks/azureDisk1_shilvm2-sles11" }
```

```
                VMResourceGroup = ShilRG
                AzureAuthResName = azurauth
                )

        AzureDisk diskres1 (
                DiskIds = {
                        "/subscriptions/6940a326-abg6-40dd-b628-
c1e9bbdf1d63/
                                resourceGroups/SHILRG/providers/
Microsoft.Compute/
                                disks/azureDisk1_shilvm2-sles11_1" }
                VMResourceGroup = ShilRG
                AzureAuthResName = azurauth
                )

        AzureDisk diskres3 (
                DiskIds = {
                        "/subscriptions/6940a326-abg6-40dd-b628-
c1e9bbdf1d63/
                                resourceGroups/SHILRG/providers/
Microsoft.Compute/
                                disks/azureDisk1_shilvm2-sles11_2" }
```

```
        VMResourceGroup = ShilRG
        AzureAuthResName = azurauth
        )


AzureIP azureipres_vvr (
        PrivateIP = "10.3.3.200"
        NICDevice = eth0
        VMResourceGroup = ShilRG
        AzureAuthResName = azurauth
        )


AzureIP azureipres_vvr1 (
        PrivateIP = "10.3.3.201"
        NICDevice = eth0
        VMResourceGroup = ShilRG
        AzureAuthResName = azurauth
        )


DiskGroup dgres (
        DiskGroup = vvrdg
        )
```

```
        IP ip_vvr (
                Device = eth0
                Address = "10.3.3.200"
                NetMask = "255.255.255.0"
                )


        NIC nic_vvr (
                Device = eth0
                )


        RVG rvgres (
                RVG = rvg
                DiskGroup = vvrdg
                )


        azureipres_vvr requires ip_vvr
        dgres requires diskres
        dgres requires diskres1
        ip_vvr requires nic_vvr
        rvgres requires azureipres_vvr
        rvgres requires dgres
```

```
group datagrp (
        SystemList = { azureVM1 = 0, azureVM2 = 1 }
        ClusterList = { shil-sles11-clus1-eastus = 0,
                 shil-sles11-clus2-eastus2 = 1 }
        Authority = 1
        )
Application sample_app (
User = "root"
StartProgram = "/data/sample_app start"
StopProgram = "/data/sample_app stop"
PidFiles = { "/var/lock/sample_app/app.pid" }
MonitorProcesses = { "sample_app" }
)
        Mount mountres (
                MountPoint = "/data"
                BlockDevice = "/dev/vx/dsk/vvrdg/vol1"
                FSType = vxfs
                FsckOpt = "-y"
                )

        RVGPrimary rvgprimary (
                RvgResourceName = rvgres
```

```
                AutoResync = 1

                )



        requires group VVR online local hard

        mountres requires rvgprimary

       sample_app requires  mountres
```

Sample configuration using a managed identity

```
cluster shil-sles11-clus1-eastus (

ClusterAddress = "10.3.3.100"

SecureClus = 1

)



remotecluster shil-sles11-clus2-eastus2 (

ClusterAddress = "10.5.0.5"

)



heartbeat Icmp (

ClusterList = { shil-sles11-clus2-eastus2 }

Arguments @shil-sles11-clus2-eastus2 = { "10.5.0.5" }

)
```

```
system azureVM1 (

)



system azureVM2 (

)



group ClusterService (

SystemList = { azureVM1 = 0, azureVM2 = 1 }

AutoStartList = { azureVM1, azureVM2 }

OnlineRetryLimit = 3

OnlineRetryInterval = 120

)



    Application wac (

    StartProgram = "/opt/VRTSvcs/bin/wacstart -secure"

    StopProgram = "/opt/VRTSvcs/bin/wacstop"

    MonitorProcesses = { "/opt/VRTSvcs/bin/wac -secure" }

    RestartLimit = 3

    )



    AzureIP azureipres (

    PrivateIP = "10.3.3.100"
```

```
    NICDevice = eth0
    VMResourceGroup = ShilRG
    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022
    )

    IP gcoip (
    Device = eth0
    Address = "10.3.3.100"
    NetMask = "255.255.255.0"
    )

    NIC gconic (
    Device = eth0
    )

    gcoip requires azureipres
    gcoip requires gconic
    wac requires gcoip

group VVR (
SystemList = { azureVM1 = 0, azureVM2 = 1 }
AutoStartList = { azureVM1, azureVM2 }
```

```
)


    AzureDisk diskres (
    DiskIds = {
    "/subscriptions/6940a326-abg6-40dd-b628-c1e9bbdf1d63/
    resourceGroups/SHILRG/providers/Microsoft.Compute/
    disks/azureDisk1_shilvm2-sles11" }
    VMResourceGroup = ShilRG
    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022
)


    AzureDisk diskres1 (
    DiskIds = {
    "/subscriptions/6940a326-abg6-40dd-b628-c1e9bbdf1d63/
    resourceGroups/SHILRG/providers/Microsoft.Compute/
    disks/azureDisk1_shilvm2-sles11_1" }
    VMResourceGroup = ShilRG
    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022
    )


    AzureDisk diskres3 (
    DiskIds = {
```

```
    "/subscriptions/6940a326-abg6-40dd-b628-c1e9bbdf1d63/
    resourceGroups/SHILRG/providers/Microsoft.Compute/
    disks/azureDisk1_shilvm2-sles11_2" }
    VMResourceGroup = ShilRG
    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022
    )

    AzureIP azureipres_vvr (
    PrivateIP = "10.3.3.200"
    NICDevice = eth0
    VMResourceGroup = ShilRG
    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022
    )

    AzureIP azureipres_vvr1 (
    PrivateIP = "10.3.3.201"
    NICDevice = eth0
    VMResourceGroup = ShilRG
    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022
    )

    DiskGroup dgres (
```

```
DiskGroup = vvrdg

)



IP ip_vvr (

Device = eth0

Address = "10.3.3.200"

NetMask = "255.255.255.0"

)



NIC nic_vvr (

Device = eth0

)



RVG rvgres (

RVG = rvg

DiskGroup = vvrdg

)



azureipres_vvr requires ip_vvr

dgres requires diskres

dgres requires diskres1

ip_vvr requires nic_vvr
```

```
    rvgres requires azureipres_vvr

    rvgres requires dgres



group datagrp (

SystemList = { azureVM1 = 0, azureVM2 = 1 }

ClusterList = { shil-sles11-clus1-eastus = 0,

shil-sles11-clus2-eastus2 = 1 }

Authority = 1

)



Application sample_app (

User = "root"

StartProgram = "/data/sample_app start"

StopProgram = "/data/sample_app stop"

PidFiles = { "/var/lock/sample_app/app.pid" }

MonitorProcesses = { "sample_app" }

)



    Mount mountres (

    MountPoint = "/data"

    BlockDevice = "/dev/vx/dsk/vvrdg/vol1"

    FSType = vxfs
```

```
    FsckOpt = "-y"

    )



    RVGPrimary rvgprimary (

    RvgResourceName = rvgres

    AutoResync = 1

    )



    requires group VVR online local hard

    mountres requires rvgprimary

    sample_app requires mountres
```

Configurations for Microsoft Azure - Windows

This section includes the following topics:

- Replication configurations in Azure - Windows
- HA and DR configurations in Azure - Windows

Replication configurations in Azure - Windows

The procedure for setting up replication in Azure depends on where your primary data center and your secondary data center are located.

Follow the procedure that is appropriate for your setup:

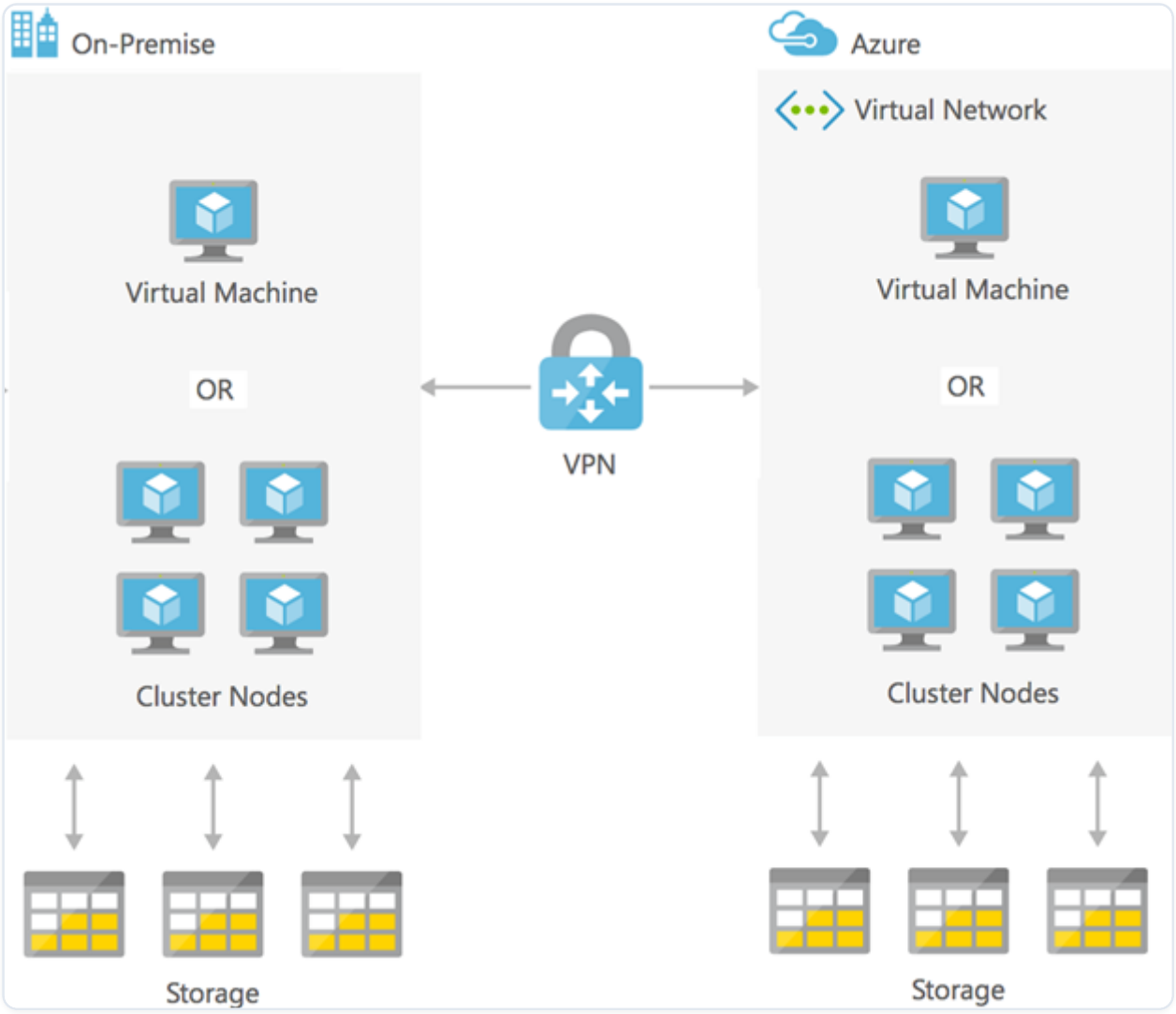
- See Replication from on-premises to Azure - Windows.
- See Replication within an Azure region - Windows.
- See Replication across Azure regions - Windows.

Replication from on-premises to Azure - Windows

The following diagram illustrates the sample configuration for setting up replication between an on-premises data center to an Azure data center:

> **Note:** For the ease of use, the machines, whether virtual or physical, are commonly mentioned as virtual machines. Your on-premises data center may include physical machines instead of the virtual machines. In any case, the procedure to set up replication from an on-premises data center to a cloud data center remain the same for physical and well as virtual machines.

> Figure: Sample configuration for setting up replication between an on-premises data center to cloud data center



About setting up replication between an on-premise data center to on-cloud data center

Replication between an on-premise data center to on-cloud data center involves the following high-level steps:

1. Prepare the setup at on-premise data center
2. Prepare the setup at on-cloud data center
3. Establish a tunnel from on-premise data center to on-cloud data center
4. Deploy setup

The following sections provide details about performing each of these steps.

Preparing the setup at on-premise data center

To prepare the setup at on-premise data center

1. Enable the ports that are used for inbound and outbound communication.

For details, refer to the *InfoScale Installation and Upgrade Guide - Windows*.

1. Create a subnet and a local VPN gateway.
2. Note the address space that is allotted for the subnet and the public IP address that is allotted for the local VPN gateway.

Preparing the setup at on-cloud data center

To prepare the setup at on-cloud data center

1. Using Microsoft Azure portal, create a resource group.
2. Create a VNet in the resource group created and specify an IP address space for the VNet.

The IP address range must be diff on the on-premise subnet and on the on-cloud subnet.

1. Create a gateway subnet.
2. Create a VPN gateway and associate it with the created VNet.

Note the public IP address that is allotted for the on-cloud VPN gateway.

1. Create a local network gateway.

When you create the local network gateway, you must provide the on-premise subnet IP address range and the public IP address of the on-premise local VPN gateway.

1. Establish a tunnel from on-cloud to on-premise network.

To establish the tunnel, create a connection of type **Site-to-Site (IPSec)** and choose the on-cloud VPN gateway and the local network gateway.

1. Provide a shared key (alpha-numeric key).

A shared key is a pass-phrase. This pass-phrase is required when you establish a tunnel from on-premise data center to cloud data center.

Establishing a tunnel from on-premise data center to on-cloud data center

To establish a tunnel from on-premise data center to on-cloud data center, use the following parameters:

- Public IP address that is allotted for the on-cloud VPN gateway
- Shared key (alpha-numeric key) that was provided while establishing a tunnel from on-cloud data center to on-premise data center
- On-cloud VPN gateway configuration type (Policy based or Route based)

Deploying the setup

To deploy the setup (in both the data centers)

1. Create virtual machines in the subnets created.
2. Provision storage.
3. Install the appropriate InfoScale product.
4. Create VxVM disk groups, VxVM volumes, Replicated Volume Group (RVG), and RLinks.
5. Flush the iptables on both the virtual machines.

```
# iptable -F
```

1. Set up replication between the virtual machines using the private IP address or the virtual IP address.
2. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

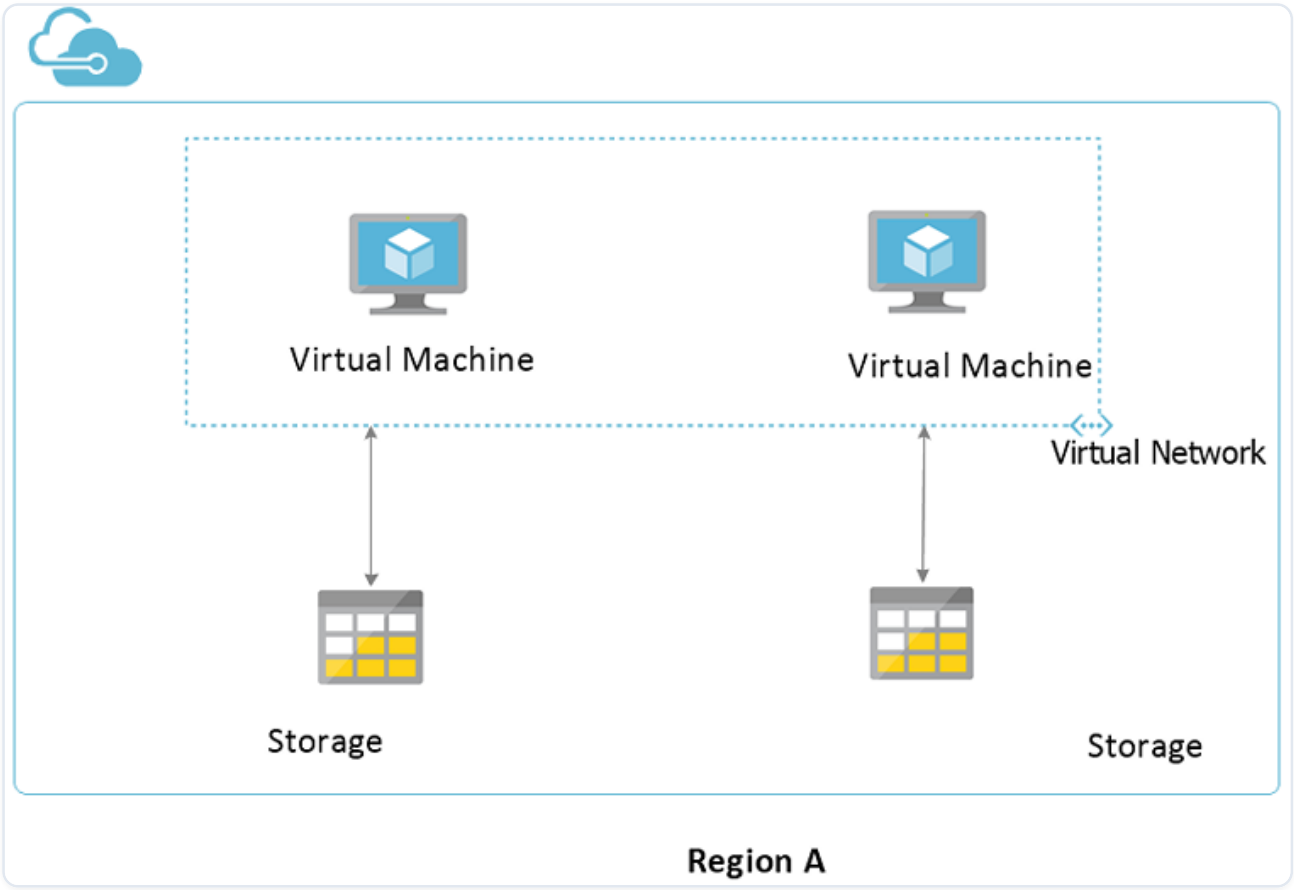
```
Replication status: replicating (connected)
```

Replication within an Azure region - Windows

In an Azure cloud environment, in a single region, you can provision your setup across virtual networks (VNets) or within a VNet.

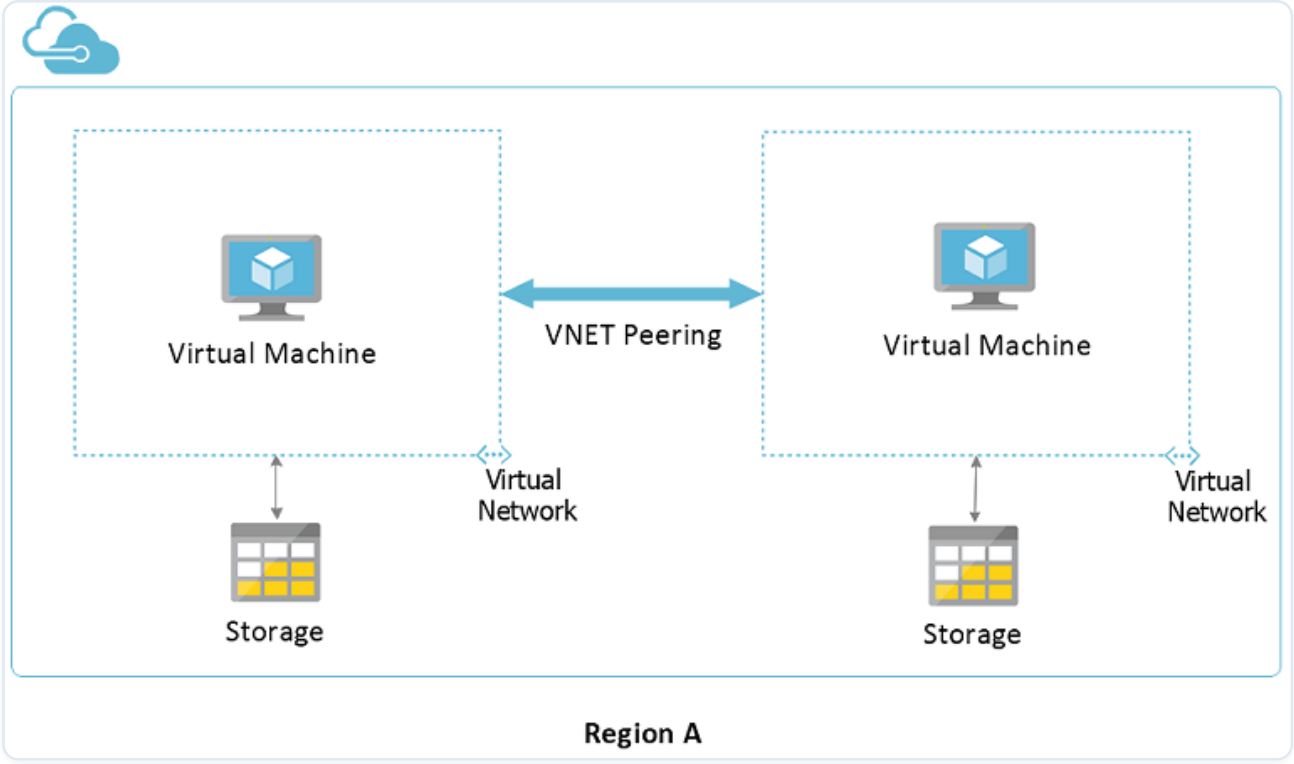
The following diagram illustrates the sample configuration for setting up replication between the same VNet:

Figure: Sample configuration for setting up replication between same VNet



The following diagram illustrates the sample configuration for setting up replication across VNets:

Figure: Sample configuration for setting up replication across VNets



Setting up replication in the same VNet

To set up replication in the same VNet within the same region

1. Enable the ports that are used for inbound and outbound communication.

For details, refer to the *InfoScale Installation and Upgrade Guide - Windows*.

1. Using Microsoft Azure portal, create a VNet and specify an IP address space for the VNet.
2. Create a subnet in the VNet created.

For details about creating a VNet, specifying an IP address space, and creating a subnet, refer to Microsoft documentation.

1. Create two virtual machines within the subnets and provision storage.
2. Install the appropriate InfoScale product on both the virtual machines.
3. Create VxVM disk groups, VxVM volumes, Storage Replicator Log (SRL), Replicated Volume Group (RVG), and RLinks.
4. Flush the iptables on both the virtual machines.

```
# iptable -F
```

1. Set up replication between the virtual machines using the private IP address or the virtual IP address.
2. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

Setting up replication across VNet

To set up replication across the VNets, within the same region

1. Using Microsoft Azure portal, create two VNets and specify an IP address space for each VNet.
2. Set up VNet Peering between the two VNets.
3. Create a subnet in each VNet.
4. Create a virtual machine in each subnet and provision storage.
5. Install the appropriate InfoScale product on both the virtual machines.
6. Create VxVM disk groups, VxVM volumes, Storage Replicator Log (SRL), Replicated Volume Group (RVG), and RLinks.
7. Flush the iptables on both the virtual machines.

```
# iptable -F
```

1. Set up replication between the virtual machines using the private IP address or the virtual IP address.
2. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

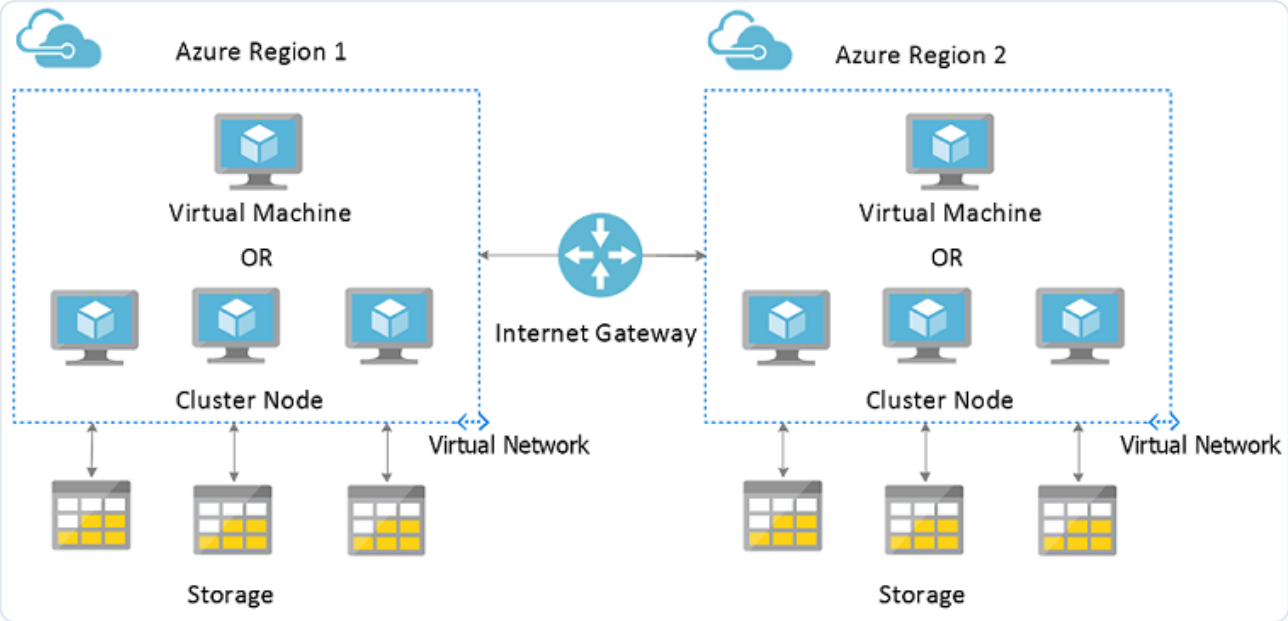
Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

Replication across Azure regions - Windows

The following diagram illustrates the sample configuration for setting up replication across regions:

Figure: Sample configuration for setting up replication across regions



Setting up replication across regions

To set up replication across region

1. Enable the ports that are used for inbound and outbound communication.

For details, refer to the *InfoScale Installation and Upgrade Guide - Windows*.

1. Using Azure portal, create a resource group (RG) in both the regions.
2. Create a VNet in each region and specify a non-overlapping IP address space for each VNet.
3. Create a subnet and a gateway subnet under the VNets created in both the regions.

4. Create a virtual network gateway in both the regions.
5. Establish a connection in between the two virtual network gateways.
6. Create virtual machines in the subnets and provision storage.
7. Install the InfoScale product.
8. Create VxVM disk groups, VxVM volumes, Replicated Volume Group (RVG), and RLinks.
9. Flush the iptables on both the virtual machines.

```
# iptable -F
```

1. Set up replication using the private IP address or the virtual IP address.
2. Verify the replication status.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

HA and DR configurations in Azure - Windows

InfoScale Enterprise lets you configure applications for high availability (HA) and disaster recovery (DR) in Azure cloud environment.

For more information on the supported use cases, read the following topics:

- See Failover within an Azure subnet using private IP - Windows.
- See Failover across Azure subnets using overlay IP - Windows.
- See Public access to cluster nodes in Azure using public IP - Windows.
- See DR from on-premises to Azure and across Azure regions or VNets - Windows.

InfoScale Enterprise provides the following agents to manage Azure resources:

- The AzureIP agent and the AzureDNSZone agent manage the network resources for cluster communication, failover, and failback. These agents monitor and manage the IP resources within cluster nodes in the Azure cloud.
- The following options are available for handling authentication for Azure subscription:

- You can configure the AzureAuth agent to use the service principal credentials for authentication. The AzureIP, AzureDNSZone, and the AzureDisk agents then depend on the AzureAuth agent for handling the Azure related authentication.
- You can also use the Azure user-assigned managed identities for authentication. You can configure the `ManagedIdentityClientID` attribute for the AzureIP, AzureDNSZone, and the AzureDisk agents to use managed identities for the Azure related authentication.

In this case, these agents no longer depend on the AzureAuth agent; you do not have to configure the AzureAuth agent resources in the cluster.

> **Note:** Acrtera recommends that you configure the agents to use managed identities, instead of the AzureAuth agent, for authenticating the Azure subscription.

- In an Azure environment, application data is stored on the Azure data disks that are attached to the Azure virtual machines. The AzureDisk agent provides high availability of the Azure disks during an application failover. This agent supports high availability for managed and unmanaged disks.

For more information about the agents, refer to the *Cluster Server Bundled Agents Reference Guide - Windows.*

> **Note:** Azure recommends that you configure Azure VMs in the same Availability Set to ensure that at least one node from the cluster is always operational.

> VCS lets you configure replication between Azure virtual machines, which is further used to support various HA and DR scenarios for applications in Azure cloud.

For more information on the supported replication configurations in Azure, refer to the following documents:

- *Cluster Server Administrator's Guide - Windows*
- *Volume Replicator Administrator's Guide - Windows*

Shared disk support in Azure cloud and InfoScale service group configuration using wizards

InfoScale supports shared disks in an Azure cloud environment. You can use the InfoScale application service group configuration wizards on Windows platform to configure high availability

for various applications such as SQL Server and file shares. The service groups automatically configure the required InfoScale Azure agents for managing the application and the network resources in the Azure cloud.

The following limitations are applicable when you use these wizards on Windows:

- If the cluster nodes span across different subnets, the wizards configure the AzureIP agent resources using the private IP address (PrivateIP attribute) instead of using the overlay IP address (OverlayIP attribute).

In such a case, you must run the wizard until the end and complete the service group configuration. After the service group is configured, you must manually modify the service group and replace the PrivateIP entry with the OverlayIP attribute and the overlap IP address.

- When you create new service groups in a shared storage environment, the wizards configure the Azure agents with managed identities for authenticating the Azure subscription. You cannot configure the AzureAuth agent to use service principal credentials for managing Azure authentication.

If required, you can manually modify the service group later and add the AzureAuth agent resources and dependencies. However, Acrtera recommends that you use managed identities for Azure authentication.

Refer to the *Cluster Server Bundled Agents Reference Guide - Windows* for more information about Azure agents and user-assigned managed identities.

Failover within an Azure subnet using private IP - Windows

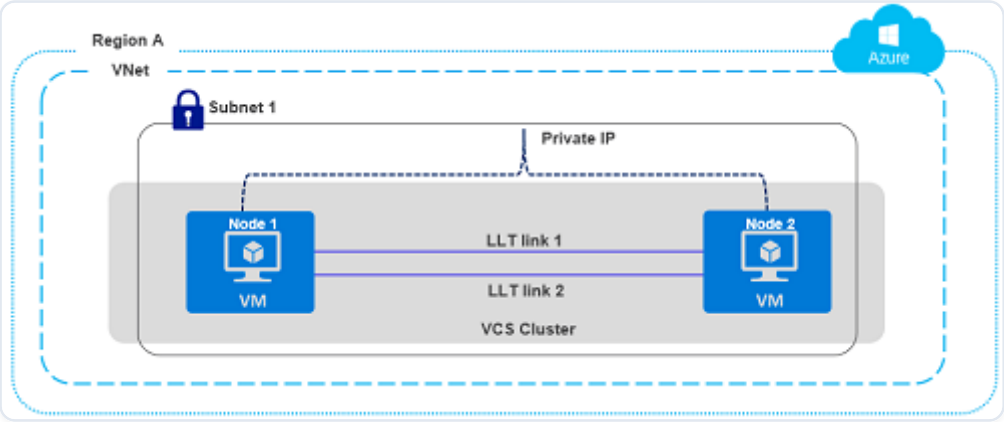
InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - within the same subnet in the same VNet.

The following information is required:

- A private IP (secondary private IP) address to be failed over
- The device to which the IP should be plumbed

Sample configuration with private IP

The following graphic depicts a sample failover configuration within the same subnet using a private IP:



The sample configuration includes the following elements:

- A Azure virtual network (VNet) is configured in Region A of the Azure cloud
- An application is configured for HA using a cluster that comprises two nodes, Node1 and Node2, which are Azure virtual machines
- Both the cluster nodes exist in the same subnet
- A private IP is configured, which is failed over from one node to the other as part of the failover or the failback operations

Sample service group configuration with private IP

The following snippet is a service group configuration from a sample configuration file (main.cf):

Sample configuration using AzureAuth agent

```
group AzureAuthGrp (

    SystemList = { AzureVM1 = 0, AzureVM2 = 1 }

    Parallel = 1

    )



    AzureAuth azureAuth (

        SubscriptionId = 640a326-fga6-90gh-b616-c1e9bb

        ClientId = e8d899-d32a-47d04-8986-be739104d

        SecretKey = fntPgnUnhTprQrqTRonSlpRhnGrrNklFngLs
```

```
        TenantId = 9fjkabae-2348-4308-b503-6667d61
        )

        Phantom phres (
        )

group AzureIPGrp (
    SystemList = { AzureVM1 = 0, AzureVM2 = 1 }
    )

        IP IP_res (
            Address = "10.1.5.67"
            SubNetMask = "255.255.255.0"
            MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
            MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
            )

        NIC NIC_res (
            MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
            MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
            )
```

```
        AzureIP AzureIP_res (
            PrivateIP = "10.1.5.42"
            MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
            MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
            AzureAuthResName = azureAuth
            )


    IP_res requires AzureIP_res
    AzureIP_res requires NIC_res
```

Sample configuration using a managed identity

```
group AzureIPGrp (
SystemList = { AzureVM1 = 0, AzureVM2 = 1 }
)


    IP IP_res (
    Address = "10.1.5.67"
    SubNetMask = "255.255.255.0"
    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
    )
```

```
    NIC NIC_res (
    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
    )

    AzureIP AzureIP_res (
    PrivateIP = "10.1.5.42"
    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022
    )

IP_res requires AzureIP_res
AzureIP_res requires NIC_res
```

Failover across Azure subnets using overlay IP - Windows

InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - between different subnets in the same VNet.

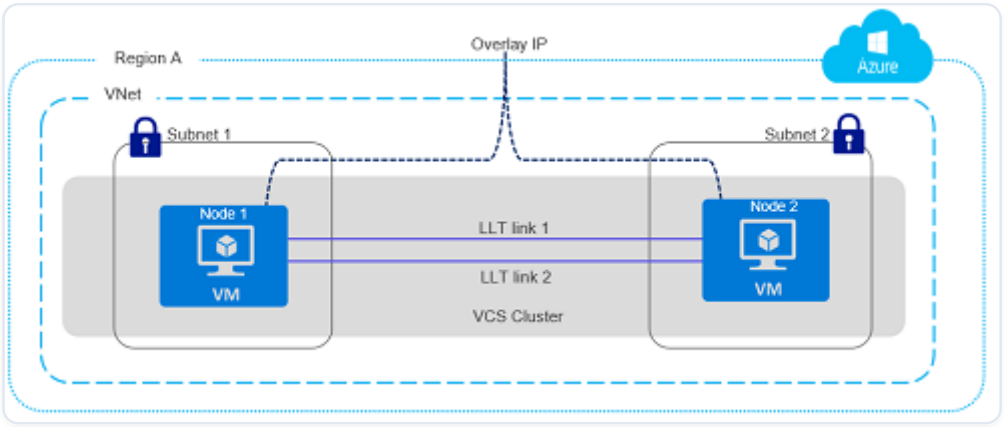
The following information is required:

- The IP address outside VNet to be used for failover
- The device to which the IP should be plumbed

Azure does not allow the private IP of one subnet to be failed over to a different subnet. To overcome this limitation, provide an overlay IP, which is outside the VNet level, so that it can be used across subnets.

Sample configuration with Overlay IP for failover across subnets in the same VNet

The following graphic depicts a sample failover configuration across subnets within the same VNet using an overlay IP:



The sample configuration includes the following elements:

- A Azure virtual network (VNet) is configured in Region A of the Azure cloud
- An application is configured for HA using a cluster that comprises two nodes, Node1 and Node2, which are Azure virtual machines
- Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.
- An overlay IP is configured to allow redirecting IP address traffic to another cluster node belonging to different subnet within the same VNet as a part of the failover or the failback operations

Sample service group configuration with overlay IP for failover across subnets in the same VNet

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

Sample configuration using AzureAuth agent

```
group AzureAuthGrp (

                SystemList = { AzureVM1 = 0, AzureVM2 = 1 }

                Parallel = 1

                )
```

```
                AzureAuth azureAuth (
                        SubscriptionId = 640a326-fga6-90gh-b616-c1e9bb
                        ClientId = e8d899-d32a-47d04-8986-be739104d
                        SecretKey = fntPgnUnhTprQrqpiNtrItpRhnGrrNklFngLs
                        TenantId = 9fjkabae-2348-4308-b503-6667d61
                                )

                Phantom phres (
                                )

group AzureOverlayIPGrp (
                SystemList = { AzureVM1 = 0, AzureVM2 = 1 }
                )

                IP IP_res (
                                Address = "192.168.3.88"
                                SubNetMask = "255.255.255.0"
                                MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
                                MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
                                )

                NIC NIC_res (
```

```
                                MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
                                MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
                                )


                AzureIP AzureOverlayIPres (
                        MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
                        MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
                        OverlayIP = "192.168.3.88"
                        RouteTableResourceIds = {
                            "/subscriptions/640a326-fga6-90gh-b616-
c1e9bb/
                                                    resourceGroups/TestRG/
providers/Microsoft.Network/
                                                    routeTables/
testRoute_eastUS2",
                            "/subscriptions/640a326-fga6-90gh-b616-
c1e9bb/
                                                resourceGroups/TestRG2/
providers/Microsoft.Network/
                                                routeTables/route1" }
                             AzureAuthResName = azureAuth
                                )
```

```
                IP_res requires AzureOverlayIPres

                AzureOverlayIPres requires NIC_res
```

Sample configuration using a managed identity

```
group AzureOverlayIPGrp (

SystemList = { AzureVM1 = 0, AzureVM2 = 1 }

)



    IP IP_res (

    Address = "192.168.3.88"

    SubNetMask = "255.255.255.0"

    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

    )



    NIC NIC_res (

    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

    )



    AzureIP AzureOverlayIPres (

    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
```

```
    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

    OverlayIP = "192.168.3.88"

    RouteTableResourceIds = {

    "/subscriptions/640a326-fga6-90gh-b616-c1e9bb/

    resourceGroups/TestRG/providers/Microsoft.Network/

    routeTables/testRoute_eastUS2",

    "/subscriptions/640a326-fga6-90gh-b616-c1e9bb/

    resourceGroups/TestRG2/providers/Microsoft.Network/

    routeTables/route1" }

    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

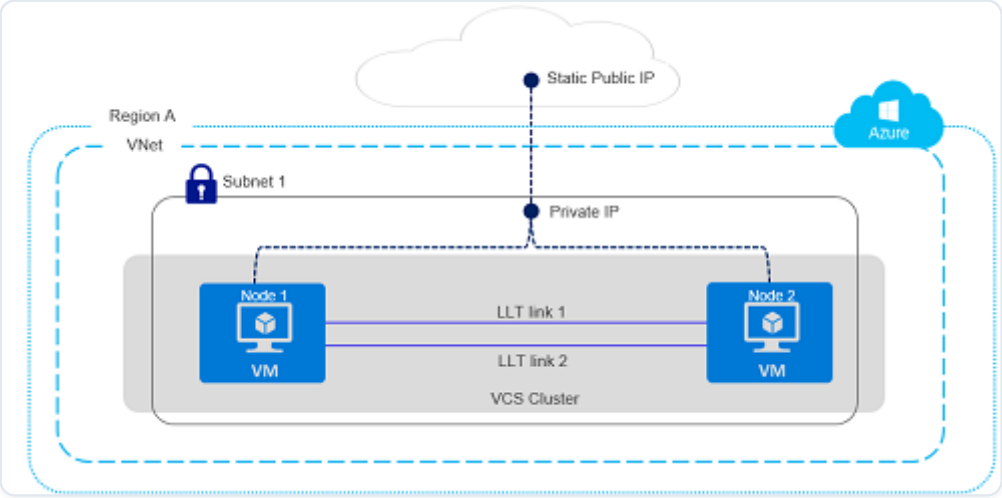
    )


IP_res requires AzureOverlayIPres

AzureOverlayIPres requires NIC_res
```

Public access to cluster nodes in Azure using public IP - Windows

To allow public access to a cluster node or to an application configured for HA in Azure environment, specify the IP in the PublicIP attribute for the AzureIP resource. This IP is used to map the Public IP address to a secondary private IP address. For example, if you have an application that needs to be highly available and to be accessible globally, you can use the PublicIP attribute of the AzureIP agent to ensure both.



Sample service group configuration with Public IP

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

Sample configuration using AzureAuth agent

```
group AzureAuthGrp (

                SystemList = { AzureVM1 = 0, AzureVM2 = 1 }

                Parallel = 1

                )



                AzureAuth azureAuth (

                    SubscriptionId = 640a326-fga6-90gh-b616-c1e9bb

                    ClientId = e8d899-d32a-47d04-8986-be739104d

                    SecretKey = fnlhPhrQpiNtrItpRhnGrrNklFngLs

                    TenantId = 9fjkabae-2348-4308-b503-6667d61

                                )



                Phantom phres (
```

```
                                )


group AzurePublicIPGrp (
                SystemList = { AzureVM1 = 0, AzureVM2 = 1 }
                )


                IP IP_res (
                     Address = "10.1.5.67"
                     SubNetMask = "255.255.255.0"
                     MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
                     MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
                          )


                NIC NIC_res (
                     MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
                     MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
                            )


                AzureIP AzurePublicIPres (
                     PrivateIP = "10.1.5.42"
                     MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
                     MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
```

```
                    PublicIP = "52.165.223.2"

                    AzureAuthResName = azureAuth

                            )



                IP_res requires AzurePublicIPres

                AzurePublicIPres requires NIC_res
```

Sample configuration using a managed identity

```
group AzurePublicIPGrp (

SystemList = { AzureVM1 = 0, AzureVM2 = 1 }

)



    IP IP_res (

    Address = "10.1.5.67"

    SubNetMask = "255.255.255.0"

    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

    )



    NIC NIC_res (

    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
```

```
    )


    AzureIP AzurePublicIPres (

    PrivateIP = "10.1.5.42"

    MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

    MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

    PublicIP = "52.165.223.2"

    ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

    )


IP_res requires AzurePublicIPres

AzurePublicIPres requires NIC_res
```

DR from on-premises to Azure and across Azure regions or VNets - Windows

VCS lets you use the global cluster option (GCO) for DR configurations. You can use a DR configuration to fail over applications across different regions or VNets in Azure or between an on-premises site and Azure.

The following is required for on-premise to cloud DR using VPN tunneling:

- Prepare the setup at on-premise data center
- Prepare the setup at on-cloud data center
- Establish a VPN tunnel from on-premise data center to cloud data center
- Virtual private IP for cluster nodes that exist in the same subnet. The IP address is used for cross-cluster communication

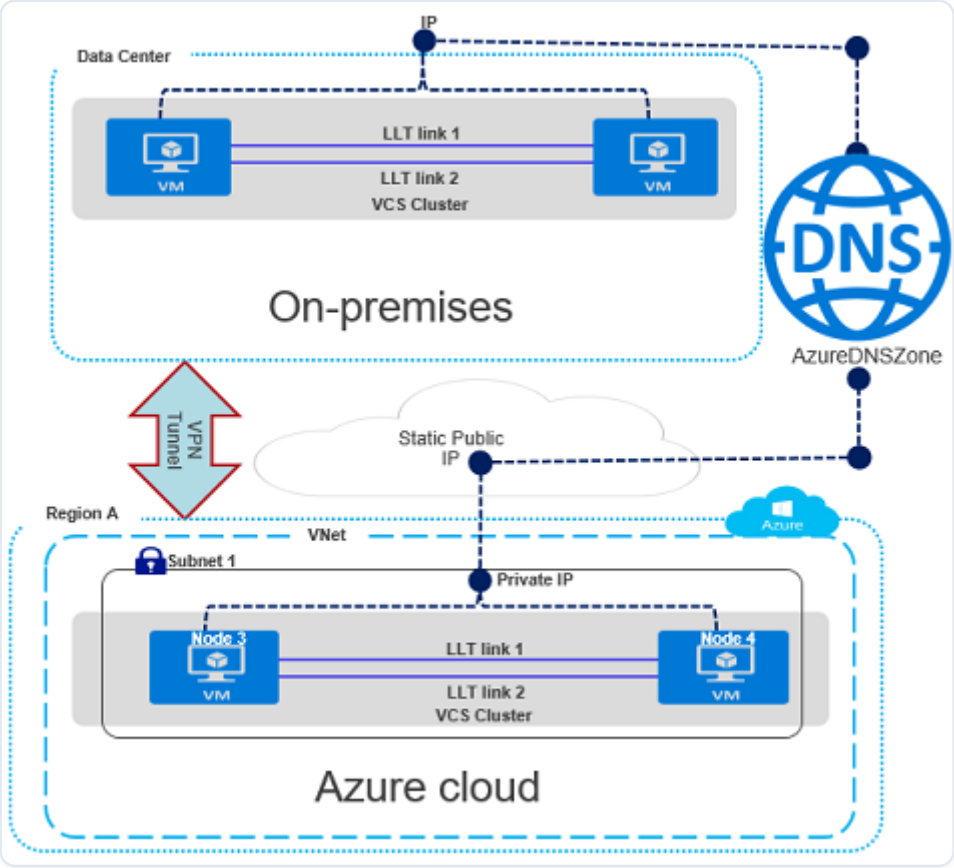
The following is required for region to region DR using VNet peering:

- Prepare the setup at both the data centers in the regions

- Establish a VNet peering from one region to the other region
- Virtual private IP for cluster nodes that exist in the same subnet. The IP address is used for cross-cluster communication

> **Note:** If you use an VPN tunnel between an on-premises site and Azure or you use VNet peering between Azure regions, the cluster nodes in the cloud must be in the same subnet.

Sample configuration with private IP



The sample configuration includes the following elements:

- VPN tunnel between on-premise data center and Region A
- The primary site has the following elements:
  - Cluster nodes in the same subnet
  - Virtual private IP for cross-cluster communication
- The secondary site has the following elements:
  - A VNet is configured in Region A of the Azure cloud

- The same application is configured for HA on Node 3 and Node 4, which exist in Subnet
- Virtual private IP for cross-cluster communication

Sample service group configuration with private IP

The following snippet is a service group configuration from a sample configuration file (main.cf):

Sample configuration using AzureAuth agent

```
cluster vcs_cluster1 (

    ClusterAddress = "10.3.3.100"

    SecureClus = 1

)



remotecluster vcs_cluster2 (

    ClusterAddress = "10.5.0.5"

)



heartbeat Icmp (

    ClusterList = { vcs_cluster2 }

    Arguments @vcs_cluster2 = { "10.5.0.5" }

)



system AzureVM1(

)

system AzureVM2(
```

```
)


group AzureAuthGrp (
    SystemList = { AzureVM1= 0, AzureVM2= 1 }
    Parallel = 1
)


    AzureAuth azurauth (
        SubscriptionId = 6940a326-abc6-40dd-b717-c1e9bcdf1d63
        ClientId = 8c891a8c-ghi2-473b-bbec-035ac50fb896
        SecretKey = gsiOssRooSpShuNoiQioNsjQlqHovUosQsrMt
        TenantId = 96dcased-0448-4308-b505-6789d61dd0e3
    )


    Phantom phres (
    )


group ClusterService (
    SystemList = { AzureVM1= 0, AzureVM2= 1 }
    AutoStartList = { AzureVM1, AzureVM2}
    OnlineRetryLimit = 3
    OnlineRetryInterval = 120
```

```
)


    Process wac (
        StartProgram @ AzureVM1 = "\"C:\\Program Files\\Veritas\\
                            Cluster Server\\bin\\wac.exe\""
        StartProgram @ AzureVM2 = "\"C:\\Program Files\\Veritas\\
                            Cluster Server\\bin\\wac.exe\""
        StopProgram @ AzureVM1 = "\"C:\\Program Files\\Veritas\\
                            Cluster Server\\bin\\wacstop.exe\""
        StopProgram @ AzureVM2 = "\"C:\\Program Files\\Veritas\\
                            Cluster Server\\bin\\wacstop.exe\""
        MonitorProgram @ AzureVM1 = "\"C:\\Program Files\\Veritas\\
                            Cluster Server\\bin\\wacmonitor.exe\""
        MonitorProgram @ AzureVM2 = "\"C:\\Program Files\\Veritas\\
                            Cluster Server\\bin\\wacmonitor.exe\""


    AzureIP azureipres (
        PrivateIP = "10.3.3.100"
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
AzureAuthResName = azurauth
    )
```

```
    IP gcoip (
        Address = "10.3.3.100"
        SubNetMask = "255.255.255.0"
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

    )

    NIC gconic (
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
    )

    gcoip requires azureipres
    gcoip requires gconic
    wac requires gcoip

group VVR (
    SystemList = { AzureVM1= 0, AzureVM2= 1 }
    AutoStartList = { AzureVM1, AzureVM2}
)
```

```
AzureDisk disk_res (
    DiskIds = {
        "/subscriptions/6940a326-abc6-40dd-b717-c1e9bcdf1d63/
     resourceGroups/TestRG/providers/Microsoft.Compute/
     disks/AzureDisk1"
    }
    VMResourceGroup = TestRG
    AzureAuthResName = azurauth
)

AzureDisk disk_res1 (
    DiskIds = {
        "/subscriptions/6940a326-abc6-40dd-b717-c1e9bcdf1d63/
       resourceGroups/TestRG/providers/Microsoft.Compute/
       disks/AzureDisk2"
        }
    VMResourceGroup = TestRG
    AzureAuthResName = azurauth
)

AzureDisk disk_res3 (
```

```
        DiskIds = {
            "/subscriptions/6940a326-abc6-40dd-b717-c1e9bcdf1d63/
         resourceGroups/TestRG/providers/Microsoft.Compute/
         disks/AzureDisk3"
            }
        VMResourceGroup = TestRG
        AzureAuthResName = azurauth
    )


    AzureIP azureipres_vvr (
        PrivateIP = "10.3.3.200"
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
        AzureAuthResName = azurauth
    )


    AzureIP azureipres_vvr1 (
        PrivateIP = "10.3.3.201"
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
        AzureAuthResName = azurauth
    )
```

```
    IP ip_vvr (
        Address = "10.3.3.200"
        SubNetMask = "255.255.255.0"
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
    )

    NIC nic_vvr (
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
    )

    VMNSDg dgres (
        DiskGroupName = vvrdg
        DGGuid = 6bee34d7-056d-4b01-ae63-397099792471
        )

    VvrRvg rvgres (
        RVG = rvg
        VMDgResName = dgres
        IPResName = ip_vvr
```

```
        )


    ip_vvr requires azureipres_vvr

    dgres requires disk_res

    dgres requires disk_res1

    ip_vvr requires nic_vvr

    rvgres requires ip_vvr

    rvgres requires dgres


group SG_FS (

    SystemList = { AzureVM1= 0, AzureVM2= 1 }

    ClusterList = { vcs_cluster1 = 0,    vcs_cluster2 = 1 }

    Authority = 1

    )


    FileShare SG_FS-FileShare (

        PathName = "\\PerfLogs"

        ShareName = PerfLogs

        LanmanResName = SG_FS-Lanman

        MountResName = SG_FS-MountV

        UserPermissions = { "VCSDOMAIN\\Administrator" = FULL_CONTROL }

        ShareSubdirectories = 1
```

```
        )


    AzureIP SG_FS-AzureIP (
        PrivateIP = "10.3.3.300"
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
        AzureAuthResName = azurauth
    )


IP SG_FS-IP (
        Address = "10.3.3.300"
        SubNetMask = "255.255.255.0"
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
        )


    Lanman SG_FS-Lanman (
        VirtualName = AzureVirtualFS
        IPResName = SG_FS-IP
        DNSUpdateRequired = 1
        ADUpdateRequired = 1
        DNSCriticalForOnline = 1
```

```
        ADCriticalForOnline = 1
        )


    MountV SG_FS-MountV (
        MountPath = "G:"
        VolumeName = volume1
        VMDGResName = dgres
        )


    NIC SG_FS-NIC (
MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
        )


    RVGPrimary SG_FS-RVGPrimary (
        RvgResourceName = rvgres
        )


    requires group VVR online local hard
    SG_FS-FileShare requires SG_FS-MountV
    SG_FS-FileShare requires SG_FS-Lanman
    SG_FS-IP requires SG_FS-NIC
```

```
    SG_FS-IP requires SG_FS-AzureIP

    SG_FS-Lanman requires SG_FS-IP

    SG_FS-MountV requires SG_FS-RVGPrimary
```

Sample configuration using a managed identity

```
cluster vcs_cluster1 (

ClusterAddress = "10.3.3.100"

SecureClus = 1

)

remotecluster vcs_cluster2 (

ClusterAddress = "10.5.0.5"

)

heartbeat Icmp (

ClusterList = { vcs_cluster2 }

Arguments @vcs_cluster2 = { "10.5.0.5" }

)

system AzureVM1(

)

system AzureVM2(

)

group ClusterService (

SystemList = { AzureVM1= 0, AzureVM2= 1 }
```

```
AutoStartList = { AzureVM1, AzureVM2}

OnlineRetryLimit = 3

OnlineRetryInterval = 120

)

Process wac (

StartProgram @ AzureVM1 = "\"C:\\Program Files\\Veritas\\

Cluster Server\\bin\\wac.exe\""

StartProgram @ AzureVM2 = "\"C:\\Program Files\\Veritas\\

Cluster Server\\bin\\wac.exe\""

StopProgram @ AzureVM1 = "\"C:\\Program Files\\Veritas\\

Cluster Server\\bin\\wacstop.exe\""

StopProgram @ AzureVM2 = "\"C:\\Program Files\\Veritas\\

Cluster Server\\bin\\wacstop.exe\""

MonitorProgram @ AzureVM1 = "\"C:\\Program Files\\Veritas\\

Cluster Server\\bin\\wacmonitor.exe\""

MonitorProgram @ AzureVM2 = "\"C:\\Program Files\\Veritas\\

Cluster Server\\bin\\wacmonitor.exe\""

)

AzureIP azureipres (

PrivateIP = "10.3.3.100"

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC
```

```
ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)

IP gcoip (

Address = "10.3.3.100"

SubNetMask = "255.255.255.0"

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

)

NIC gconic (

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

)

gcoip requires azureipres

gcoip requires gconic

wac requires gcoip

group VVR (

SystemList = { AzureVM1= 0, AzureVM2= 1 }

AutoStartList = { AzureVM1, AzureVM2}

)

AzureDisk disk_res (

DiskIds = {

"/subscriptions/6940a326-abc6-40dd-b717-c1e9bcdf1d63/
```

```
resourceGroups/TestRG/providers/Microsoft.Compute/

disks/AzureDisk1"

}

VMResourceGroup = TestRG

ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)

AzureDisk disk_res1 (

DiskIds = {

"/subscriptions/6940a326-abc6-40dd-b717-c1e9bcdf1d63/

resourceGroups/TestRG/providers/Microsoft.Compute/

disks/AzureDisk2"

}

VMResourceGroup = TestRG

ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)

AzureDisk disk_res3 (

DiskIds = {

"/subscriptions/6940a326-abc6-40dd-b717-c1e9bcdf1d63/

resourceGroups/TestRG/providers/Microsoft.Compute/

disks/AzureDisk3"

}

VMResourceGroup = TestRG
```

```
ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)

AzureIP azureipres_vvr (

PrivateIP = "10.3.3.200"

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)

AzureIP azureipres_vvr1 (

PrivateIP = "10.3.3.201"

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)

IP ip_vvr (

Address = "10.3.3.200"

SubNetMask = "255.255.255.0"

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

)

NIC nic_vvr (

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0
```

```
MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

)

VMNSDg dgres (

DiskGroupName = vvrdg

DGGuid = 6bee34d7-056d-4b01-ae63-397099792471

)

VvrRvg rvgres (

RVG = rvg

VMDgResName = dgres

IPResName = ip_vvr

)

ip_vvr requires azureipres_vvr

dgres requires disk_res

dgres requires disk_res1

ip_vvr requires nic_vvr

rvgres requires ip_vvr

rvgres requires dgres

group SG_FS (

SystemList = { AzureVM1= 0, AzureVM2= 1 }

ClusterList = { vcs_cluster1 = 0, vcs_cluster2 = 1 }

Authority = 1

)
```

```
FileShare SG_FS-FileShare (

PathName = "\\PerfLogs"

ShareName = PerfLogs

LanmanResName = SG_FS-Lanman

MountResName = SG_FS-MountV

UserPermissions = { "VCSDOMAIN\\Administrator" = FULL_CONTROL }

ShareSubdirectories = 1

)

AzureIP SG_FS-AzureIP (

PrivateIP = "10.3.3.300"

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

ManagedIdentityClientID = 1da89bd2-9735-4266-b920-27c23b98f022

)

IP SG_FS-IP (

Address = "10.3.3.300"

SubNetMask = "255.255.255.0"

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

)

Lanman SG_FS-Lanman (

VirtualName = AzureVirtualFS
```

```
IPResName = SG_FS-IP

DNSUpdateRequired = 1

ADUpdateRequired = 1

DNSCriticalForOnline = 1

ADCriticalForOnline = 1

)

MountV SG_FS-MountV (

MountPath = "G:"

VolumeName = volume1

VMDGResName = dgres

)

NIC SG_FS-NIC (

MACAddress @AzureVM1 = 00-0D-3A-91-73-A0

MACAddress @AzureVM2 = 00-0D-3A-92-03-DC

)

RVGPrimary SG_FS-RVGPrimary (

RvgResourceName = rvgres

)

requires group VVR online local hard

SG_FS-FileShare requires SG_FS-MountV

SG_FS-FileShare requires SG_FS-Lanman

SG_FS-IP requires SG_FS-NIC
```

```
SG_FS-IP requires SG_FS-AzureIP

SG_FS-Lanman requires SG_FS-IP

SG_FS-MountV requires SG_FS-RVGPrimary
```

Configurations for Google Cloud Platform- Linux

This section includes the following topics:

- Replication configurations in GCP - Linux
- HA and DR configurations in GCP - Linux

Replication configurations in GCP - Linux

The procedure for setting up replication in GCP depends on where your primary data center and your secondary data center are located.

Refer to the topic that is appropriate for your setup:

- See Replication across GCP regions - Linux.
- See Replication across multiple GCP zones and regions (campus cluster) - Linux.

Replication across GCP regions - Linux

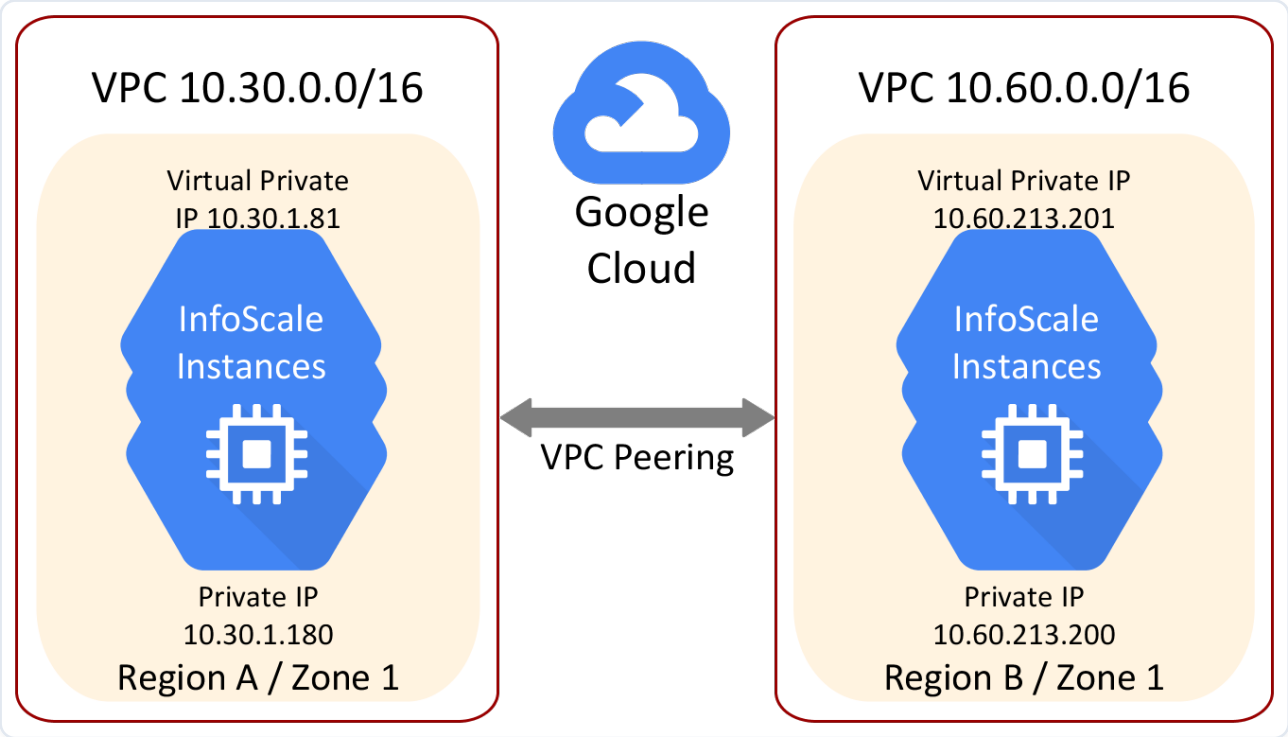
In this scenario, replication is set up across zones that are configured in different regions and VPC peering is used to connect the networks.

- Two VPCs with valid CIDR blocks (for example, 10.30.0.0/16 and 10.60.0.0/16 respectively) that are located in two different regions.
- The primary instance belongs to Zone1 of region A and the secondary instance belongs to Zone1 of region B.
- InfoScale instances in each zone.
- Set up VPC peering between the two regions.
- Elastic IP addresses (EIP) to connect the two VPN instances
- Private IP addresses used for replication in standalone environments

OR

Virtual private IP addresses used for replication in clustered environments

Figure: Replication across GCP regions



Setting up replication across regions

Perform the steps in the following procedure to set up replication across regions.

To set up replication across regions

1. Create two VPCs with valid CIDR blocks in different regions, for example, 10.30.0.0/16 and 10.60.0.0/16 respectively.
2. Create the primary site VM instance.
3. Create the secondary site VM instance.
4. Set up replication between the instances using the private IP address or virtual private IP address.

For instructions on setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the status of replication.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

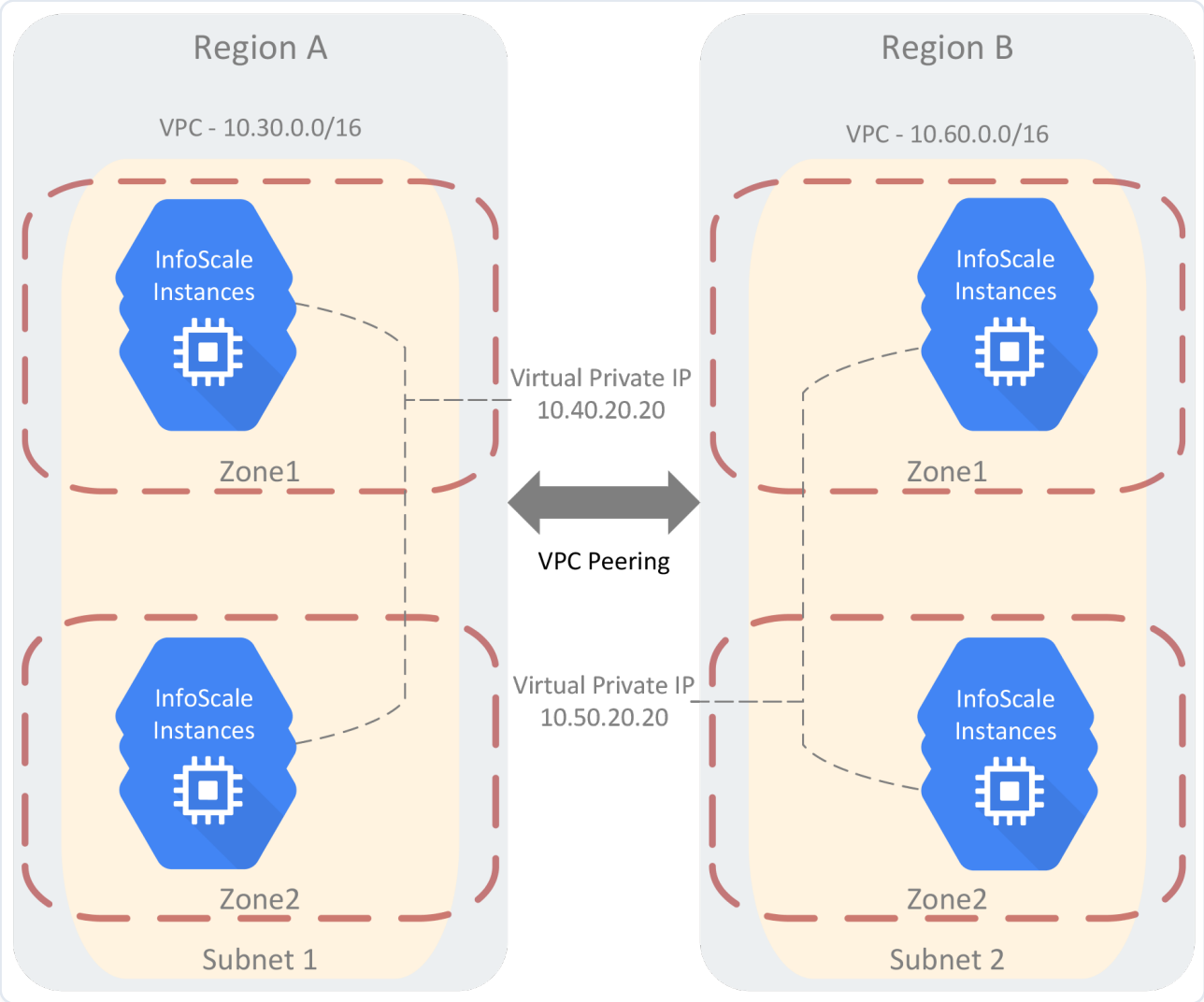
Replication across multiple GCP zones and regions (campus cluster) - Linux

In this scenario, data is replicated across multiple zones and regions.

- Two VPCs with valid CIDR blocks (for example, 10.30.0.0/16 and 10.60.0.0/16 respectively) that are located in two different regions.
- The primary instance belongs to Zone1 of region 1 and the secondary instance belongs to Zone2 of region 2.
- InfoScale instances in each zone.
- The primary Instance communicates with the secondary Instance using VPN instances at both ends.
- VPC peering is used to secure communication between the instances across the VPC networks.
- Elastic IP addresses (EIP) to connect the two VPN instances
- Private IP addresses used for replication in standalone environments OR

Overlay IP addresses used for replication in clustered environments.

Figure: Replication across multiple GCP zones and regions



Setting up replication across multiple zones and regions (campus cluster)

Perform the steps in the following procedure to set up replication across regions.

To set up replication across regions

1. Create two VPCs with valid CIDR blocks, for example, 10.30.0.0/16 and 10.60.0.0/16 respectively.
2. Create the primary site VM instances in the respective zones of the region.
3. Choose a valid virtual private IP address as the replication IP address for the primary site. The virtual private IP address is an unused IP in that subnet. Plumb the virtual private IP address on the master node of the primary site cluster.
4. Create the secondary site VM instances in the respective zones of the second region.
5. Choose a valid virtual private IP address as the replication IP address for the secondary site. The virtual private IP address is an unused IP in that subnet. Plumb the overlay IP address on the master node of the secondary site cluster.

6. After you set up VPC peering, verify that the master nodes at the primary and the secondary sites can reach each other by using the virtual private IP address.
7. Set up replication between the primary and secondary sites.

For instructions on setting up replication, refer to the *InfoScale Replication Administrator's Guide - Linux*.

1. Verify the status of replication.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the RLINK is in CONNECT state and the replication status shows:

```
Replication status: replicating (connected)
```

HA and DR configurations in GCP - Linux

InfoScale Enterprise lets you configure applications for high availability (HA) and disaster recovery (DR) in a Google Cloud Platform (GCP) environment.

For more information on the supported use cases, read the following topics:

- See Failover within a subnet of a GCP zone using virtual private IP - Linux.
- See Failover across GCP subnets using overlay IP - Linux.
- See DR across GCP regions or VPC networks - Linux.
- See Shared storage within a GCP zone or across GCP zones - Linux.

InfoScale Enterprise provides the GCPIP agent to manage the network resources for cluster communication, failover, and failback. The agents monitors and manages the IP resources of cluster nodes within the Google cloud or cluster nodes that are spread across on-premises and the Google Cloud. For details on the agents, see the *Cluster Server Bundled Agents Reference Guide - Linux*.

> **Note:** Acrtera recommends that you configure three coordination point servers (CP servers) in three different zones to manage I/O fencing.

> Application data is data is stored on the zonal persistent disks that are attached to VM instances. In case of clusters with two or more nodes, the Flexible Storage Sharing (FSS) feature enables network sharing of the zonal persistent disks. Thus, application data can be

> shared between the cluster nodes. For details on FSS, see the *Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

InfoScale Enterprise lets you configure replication between VM instances, which is further used to support various HA and DR scenarios for applications in GCP.

For details on the supported replication configurations in GCP:

See Replication configurations in GCP - Linux.

Failover within a subnet of a GCP zone using virtual private IP - Linux

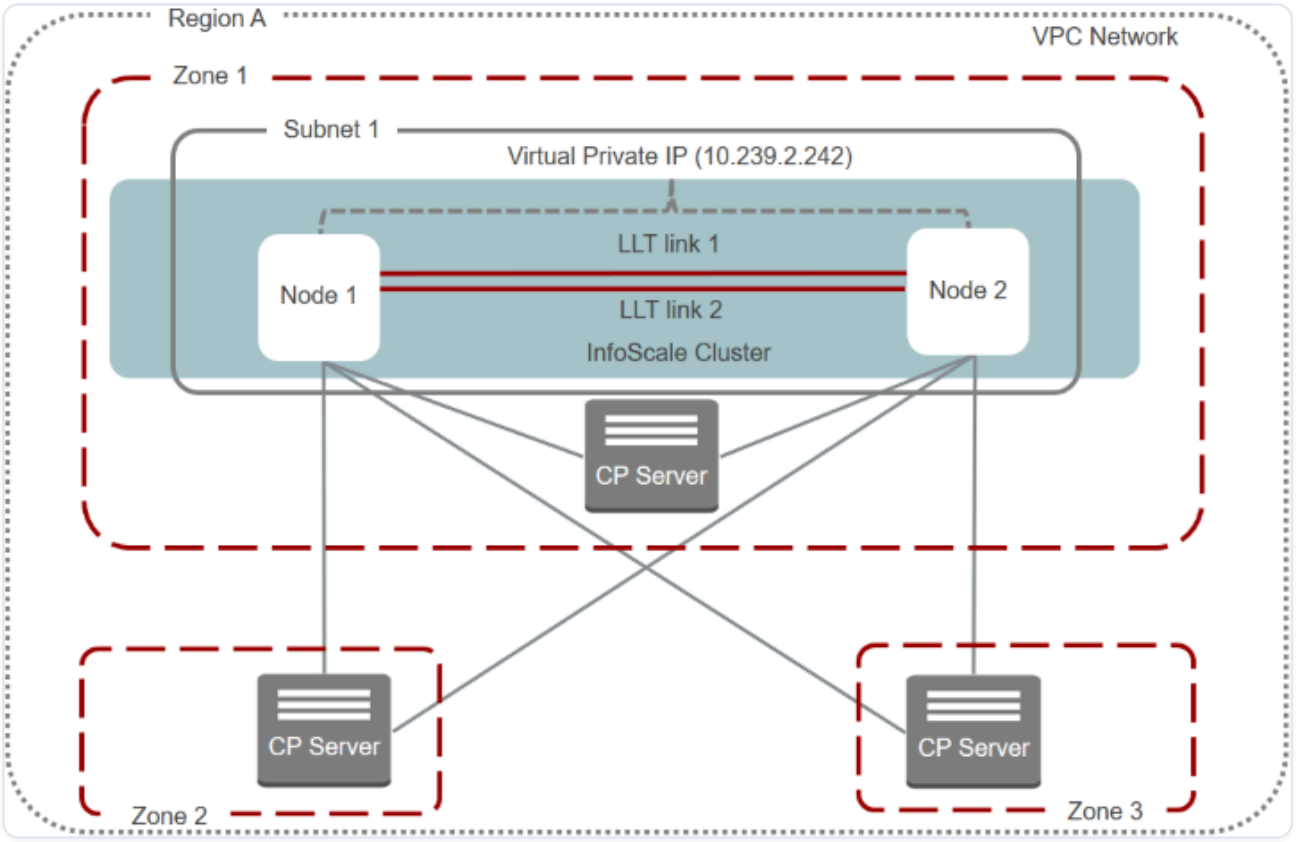
InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - within the same subnet of a zone.

The following information is required:

- The virtual private IP (secondary private IP) address to be failed over
- The device to which the IP should be plumbed
- The `google-api-python-client` GCP Python module. For details, see the *Bundled Agents Reference Guide - Linux*.

Sample configuration with private IP

The following graphic depicts a sample failover configuration within the same subnet using a virtual private IP:



The sample configuration includes the following elements:

- A VPC network is configured in Region A of the Google cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node1 and Node2, which are VM instances.
- Both the cluster nodes exist in the same subnet.
- A virtual private IP is configured, which is failed over from one node to the other as part of the failover or the failback operations.

Sample service group configuration with private IP

The following snippet is a service group configuration from a sample VCS configuration file (main.cf):

```
group oragrp (

    SystemList = { cloud-vm1 = 0, cloud-vm2 = 1 }

        )



    GoogleIP gipres (
```

```
        Device = eth0
        PrivateIP = "10.209.1.20"
        )

    IP physipres (
        Device = eth0
        Address = "10.209.1.20"
        NetMask = "255.255.255.255"
        )

    NIC nicres (
        Device = eth0
        )

gipres requires physipres
physipres requires nicres
```

Failover across GCP subnets using overlay IP - Linux

InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - between different subnets in the same zone or in different zones.

The following information is required:

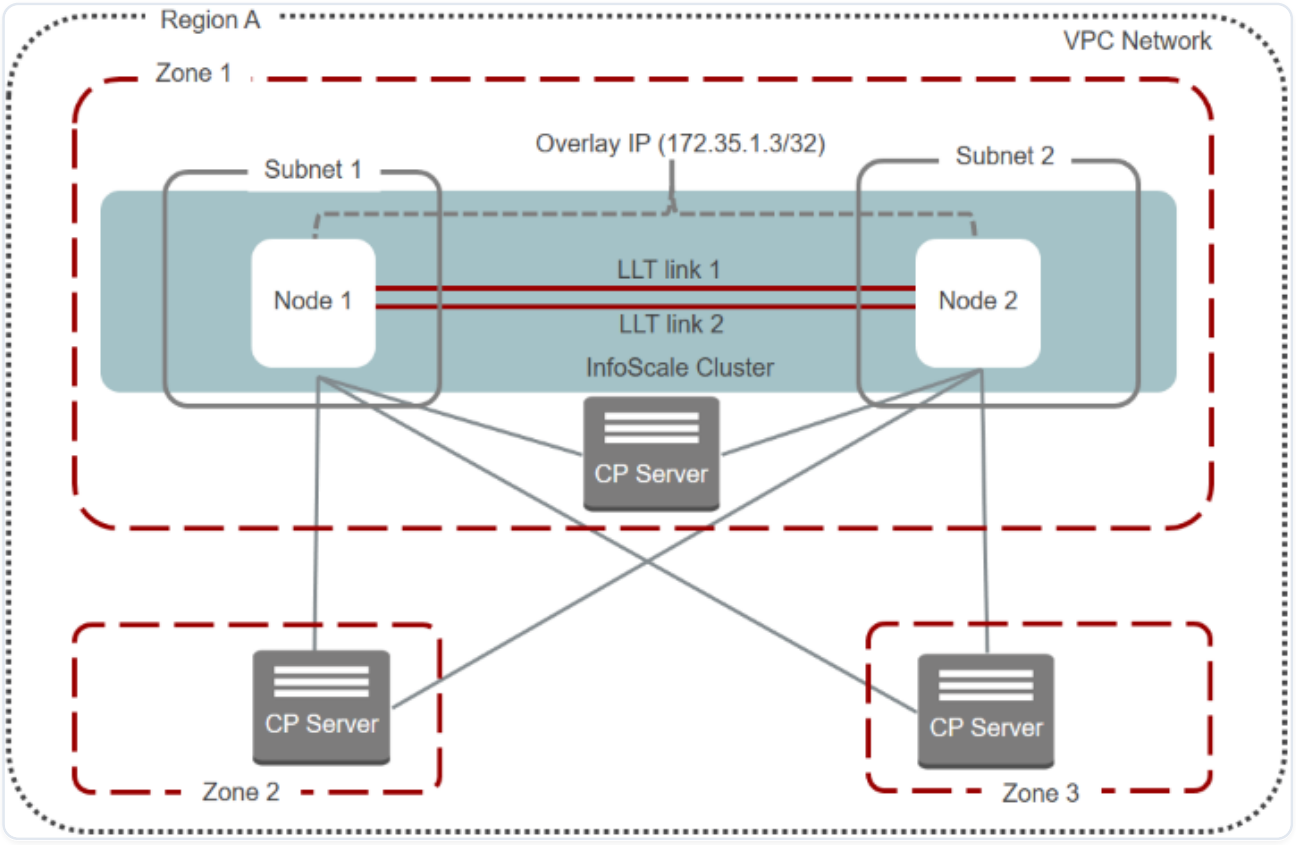
- The overlay IP address to be used for failover
- The device to which the IP should be plumbed

- The `google-api-python-client` GCP Python module. For details, see the *Bundled Agents Reference Guide - Linux*.

The private IP of one subnet cannot be failed over to a different subnet. InfoScale Enterprise provides an overlay IP, which can be used across subnets.

Sample configuration with overlay IP for failover across subnets in the same zone

The following graphic depicts a sample failover configuration across subnets within the same zone using an overlay IP:



The sample configuration includes the following elements:

- A VPC network is configured in Region A of the Google cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node 1 and Node 2, which are VM instances.

Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.

- An overlay IP is configured outside the CIDR block of the VPC network to which the cluster nodes belong. This IP is used to fail over from one subnet to another in a zone as part of the failover or the failback operations.

Sample service group configuration with overlay IP for failover across subnets in the same zone

The following snippet is a service group configuration from a sample VCS configuration file (`main.cf`):

```
group oragrp (

    SystemList = { cloud-vm1 = 0, cloud-vm2 = 1 }

    )



    GoogleIP gipres (

        Device = eth0

        OverlayIP = "192.168.10.10"

        )



    IP physipres (

        Device = eth0

        Address = "192.168.10.10"

        NetMask = "255.255.255.255"

        )



    NIC nicres (

        Device = eth0

        )

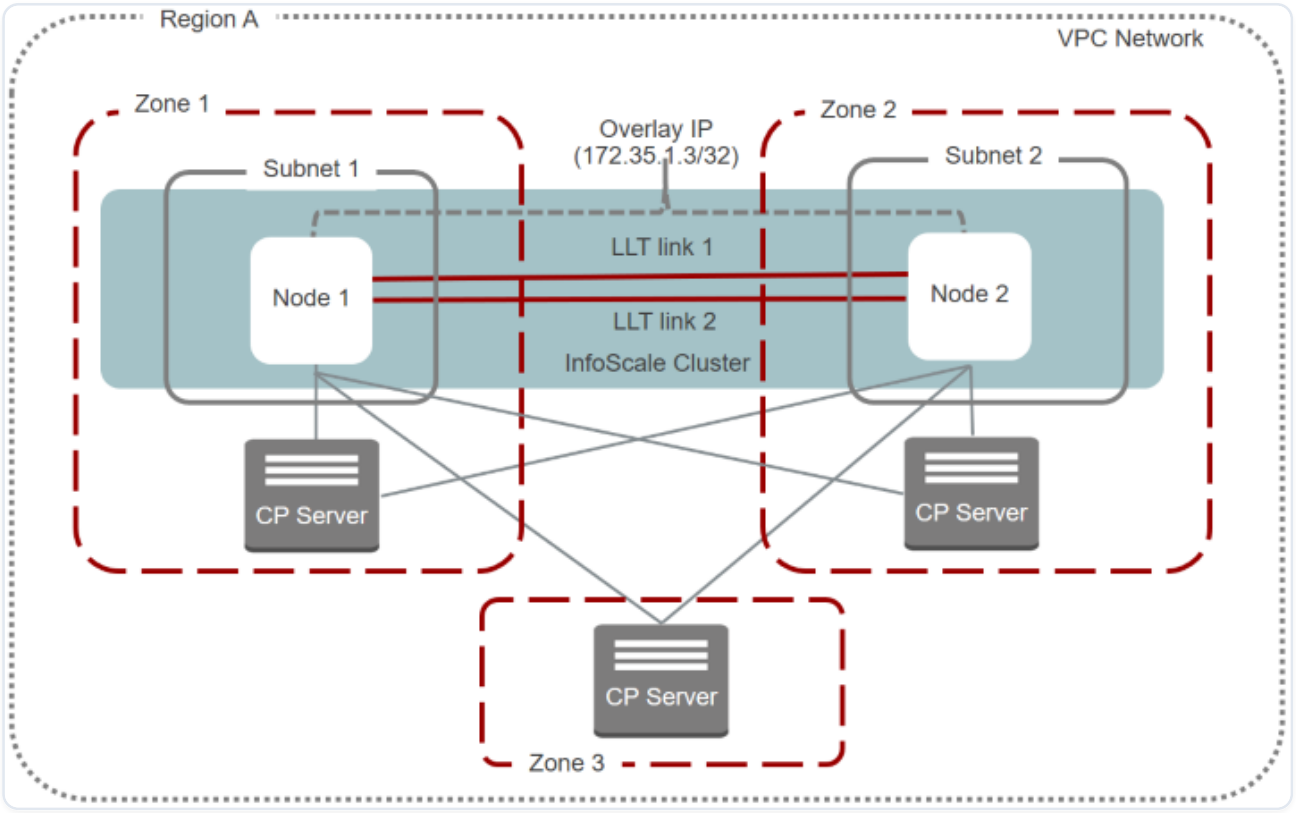


gipres requires physipres
```

```
physipres requires nicres
```

Sample configuration with overlay IP for failover across subnets in different zones

The following graphic depicts a sample failover configuration across subnets in different zones using an overlay IP:



Sample service group configuration with overlay IP for failover across subnets in different zones

The following snippet is a service group configuration from a sample VCS configuration file (`main.cf`):

```
group oragrp (

    SystemList = { cloud-vm1 = 0, cloud-vm2 = 1 }

    )



    GoogleIP gipres (

        Device = eth0
```

```
        OverlayIP = "192.168.10.10"
        )


    IP physipres (
        Device = eth0
        Address = "192.168.10.10"
        NetMask = "255.255.255.255"
        )


    NIC nicres (
        Device = eth0
        )


gipres requires physipres
physipres requires nicres
```

DR across GCP regions or VPC networks - Linux

InfoScale Enterprise lets you use the global cluster option (GCO) for DR configurations. You can use a DR configuration to fail over applications across different regions or VPC networks in the Google Cloud. The cluster nodes can be in the same zone or in different zones.

The following information is required:

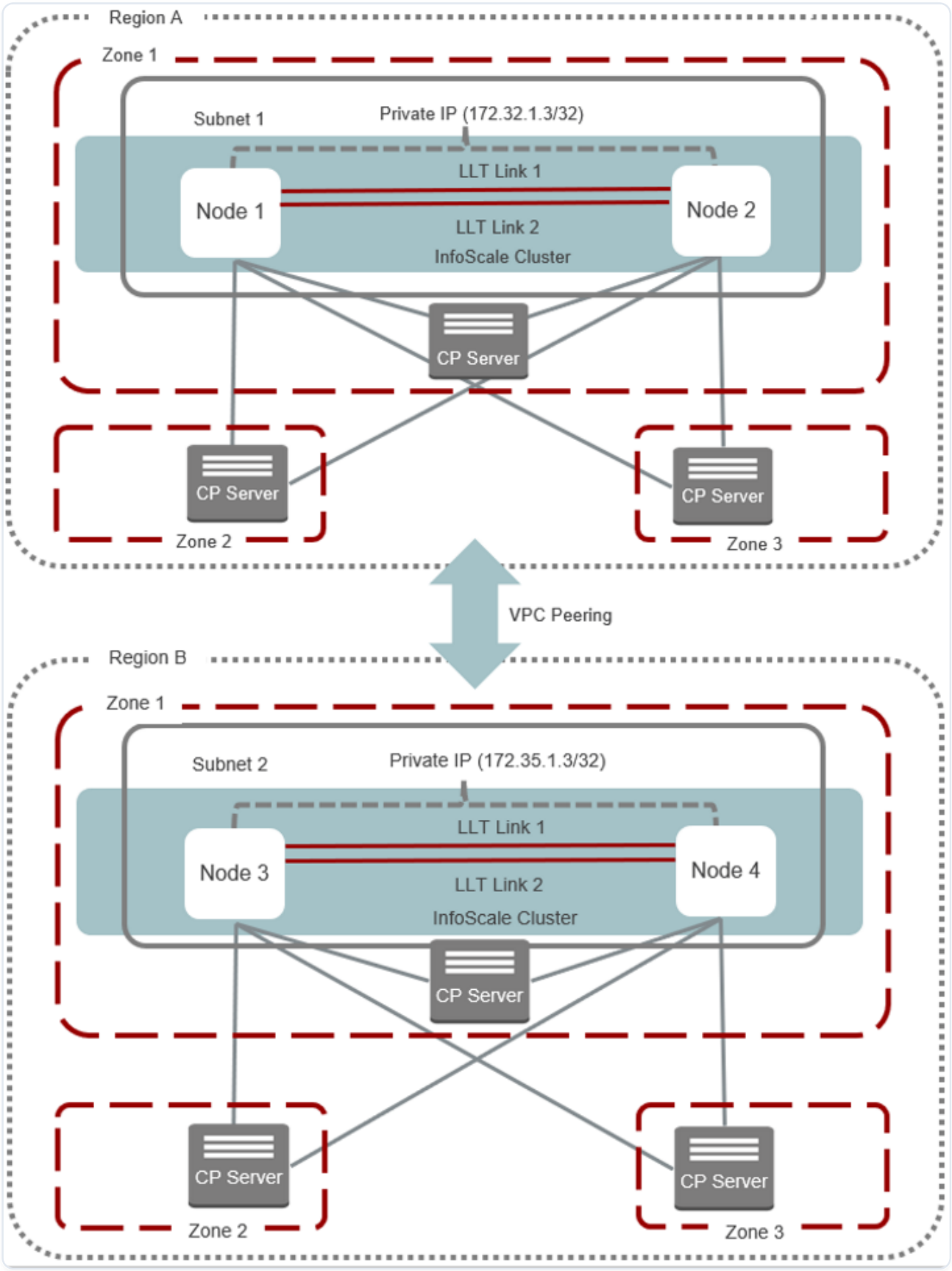
- VPC peering between regions or VPC networks
- The IP address to be used for cross-cluster communication:
    - Virtual private IP for cluster the nodes that exist in the same subnet

    - Overlay IP for cluster the nodes that exist in different subnets

- The `google-api-python-client` GCP Python module. For details, see the *Bundled Agents Reference Guide - Linux*.

Sample configuration for DR across regions or VPCs

The following graphic depicts a sample DR configuration across GCP regions:



The sample configuration includes the following elements:

- VPC network peering between Region A and Region B
- The primary site has the following elements:

- A VPC network, VPC1, is configured in Region A of the Google cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node 1 and Node 2, which are VM instances.

Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.

- The virtual private IP of a node to be failed over in a subnet within the same zone or across zones.
- The secondary site has the following elements:
- A VPC network, VPC2, is configured in Region B of the Google cloud.
- The same application is configured for HA on Node 3 and Node 4, which exist in Subnet 3 and Subnet 4 respectively.
- The virtual private IP of a node to be failed over in a subnet within the same zone or across zones.

Sample service group configuration for GCO across regions

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) at the primary site (Region A):

```
include "types.cf"



cluster pri (

    ClusterAddress = "10.209.0.11"

    UseFence = SCSI3

    HacliUserLevel = COMMANDROOT

    )



remotecluster sec (

    ClusterAddress = "10.247.0.11"

    )
```

```
heartbeat Icmp (
    ClusterList = { sec }
    Arguments @sec = { "10.247.0.11" }
    )

system cloud-vm1 (
    )

system cloud-vm2 (
    )

system cloud-vm3 (
    )

group ClusterService (
    SystemList = { cloud-vm1 = 0, cloud-vm2 = 1, cloud-vm3 = 2 }
    AutoStartList = { cloud-vm1, cloud-vm2, cloud-vm3 }
    OnlineRetryLimit = 3
    OnlineRetryInterval = 120
    )
```

```
Application wac (
    StartProgram = "/opt/VRTSvcs/bin/wacstart"
    StopProgram = "/opt/VRTSvcs/bin/wacstop"
    MonitorProcesses = { "/opt/VRTSvcs/bin/wac" }
    RestartLimit = 3
    )

GoogleIP vipres1 (
    PrivateIP = "10.209.0.11"
    Device = eth0
    )

IP pipres1 (
    Device = eth0
    Address = "10.209.0.11"
    NetMask = "255.255.255.255"
    )

NIC GoogleIP_Nic1 (
    Device = eth0
    )
```

```
pipres1 requires GoogleIP_Nic1

vipres1 requires pipres1

wac requires pipres1



// resource dependency tree

//

//      group ClusterService

//      {

//      GoogleIP vipres1

//          {

//          IP pipres1

//              {

//              NIC GoogleIP_Nic1

//              }

//          }

//      Application wac

//          {

//          IP pipres1

//              {

//              NIC GoogleIP_Nic1

//              }

//          }
```

```
    //      }
```

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) at the secondary site (Region B):

```
include "types.cf"


cluster sec (

    ClusterAddress = "10.247.0.11"

    HacliUserLevel = COMMANDROOT

    )


remotecluster pri (

    ClusterAddress = "10.209.0.11"

    )


heartbeat Icmp (

    ClusterList = { pri }

    Arguments @pri = { "10.209.0.11" }

    )


system rahul-pri-cloud-vm1 (

    )
```

```
group ClusterService (
    SystemList = { rahul-pri-cloud-vm1 = 0 }
    AutoStartList = { rahul-pri-cloud-vm1 }
    OnlineRetryLimit = 3
    OnlineRetryInterval = 120
    )


    Application wac (
        StartProgram = "/opt/VRTSvcs/bin/wacstart"
        StopProgram = "/opt/VRTSvcs/bin/wacstop"
        MonitorProcesses = { "/opt/VRTSvcs/bin/wac" }
        RestartLimit = 3
        )


    GoogleIP vipres1 (
        PrivateIP = "10.247.0.11"
        Device = eth0
        )


    IP pipres1 (
        Device = eth0
        Address = "10.247.0.11"
```

```
        NetMask = "255.255.255.255"
        )


    NIC GoogleIP_Nic1 (
        Device = eth0
        )


    pipres1 requires GoogleIP_Nic1
    vipres1 requires pipres1
    wac requires pipres1


    // resource dependency tree
    //
    //      group ClusterService
    //      {
    //      GoogleIP vipres1
    //          {
    //          IP pipres1
    //              {
    //              NIC GoogleIP_Nic1
    //              }
    //          }
```

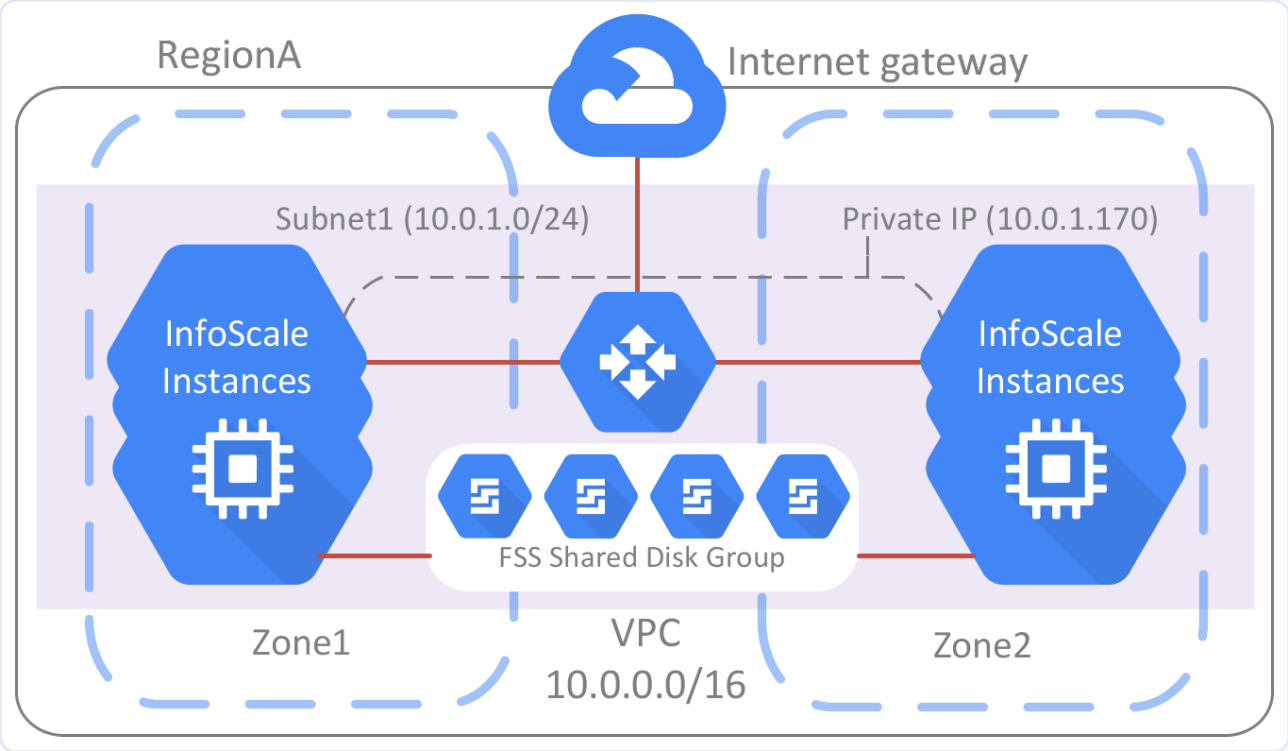
```
//      Application wac
//          {
//          IP pipres1
//              {
//              NIC GoogleIP_Nic1
//              }
//          }
//      }
```

Shared storage within a GCP zone or across GCP zones - Linux

In this scenario, FSS is set up across zones within the same region.

- A VPC network with a valid CIDR block (for example, 10.0.0.0/16) is created, which spans two zones within the same region.
- A cluster is configured within a zone or across zones.
- The primary instance is configured in Zone1 and the secondary instance is configured in Zone2.
- InfoScale instances in the primary site and secondary site subnets.

Figure: FSS across GCP zones



Setting up FSS across zones within the same region

Perform the steps in the following procedure to set up FSS across zones within the same region.

To set up replication in cloud environments

1. Create a VPC network with a valid CIDR block, for example, 10.0.0.0/16.
2. Create a subnet across the two zones, with valid CIDR range, for example, 10.0.1.0/24.
3. Launch the VM instances, and install InfoScale on those instances.
4. Set up the GoogleIP agent resource to manage the private IP address configuration.

See HA and DR configurations in GCP - Linux.

1. Set up a cluster between the instances using the private IP address.

For instructions, refer to the *Storage Foundation Cluster File System High Availability Administrator's Guide - Linux*.

1. Verify that the cluster is up by using the following command:

```
# /etc/vx/bin/vxclustadm -nidmap
```

Configurations for Google Cloud Platform - Windows

This section includes the following topics:

- Replication configurations in GCP - Windows
- HA and DR configurations in GCP - Windows

Replication configurations in GCP - Windows

The procedure for setting up replication in GCP depends on where your primary data center and your secondary data center are located.

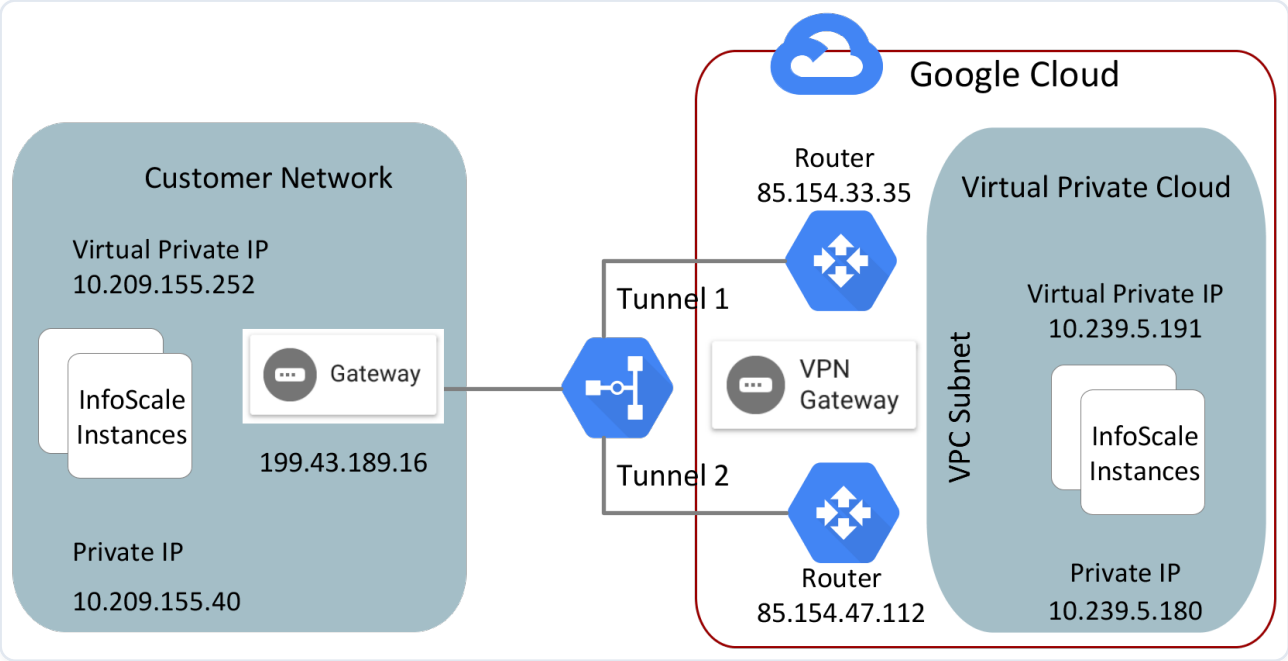
Follow the procedure that is appropriate for your setup:

- See Replication from on-premises to GCP - Windows.
- See Replication across zones in a GCP region - Windows.
- See Replication across GCP regions - Windows.

Replication from on-premises to GCP - Windows

In this scenario, replication is configured from an on-premises data center to a Google cloud data center.

Figure: Sample replication configuration from on-premises to Google cloud



Prerequisites

Ensure that the following requirements are met before you proceed with the configuration:

- The required ports are open for communication between an on-premise data center and an on-cloud data center.

For details, refer to the *InfoScale Installation and Upgrade Guide - Windows*.

- The virtual private IP addresses are plumbed on both the nodes.
- The virtual private IP addresses are configured within the subnet.

Setting up replication

The following procedure lists high-level tasks to set up replication for this scenario.

To set up replication from an on-premises data center to a Google cloud data center

1. Using the GCP Console, create a VPC network with a valid CIDR block, for example, 10.239.0.0/16.
2. Create a subnet in the VPC network and assign it to a zone.
3. Create an VM instance and associate it with the VPC network.
4. Configure a virtual private gateway and associate it to the VPC.
5. Configure a gateway in the on-premises data center.
6. Create a VPN tunnel between the on-premises network and the Google cloud network.
7. Install the appropriate InfoScale product on the VM instances in both the data centers.

8. Create VxVM disk groups, VxVM volumes, Replicated Volume Group (RVG), and RLinks.
9. Set up replication between the on-premises and on-cloud instances.
10. Verify the status of replication.

```
# vradmin -g dg_name repstatus rvg_name
```

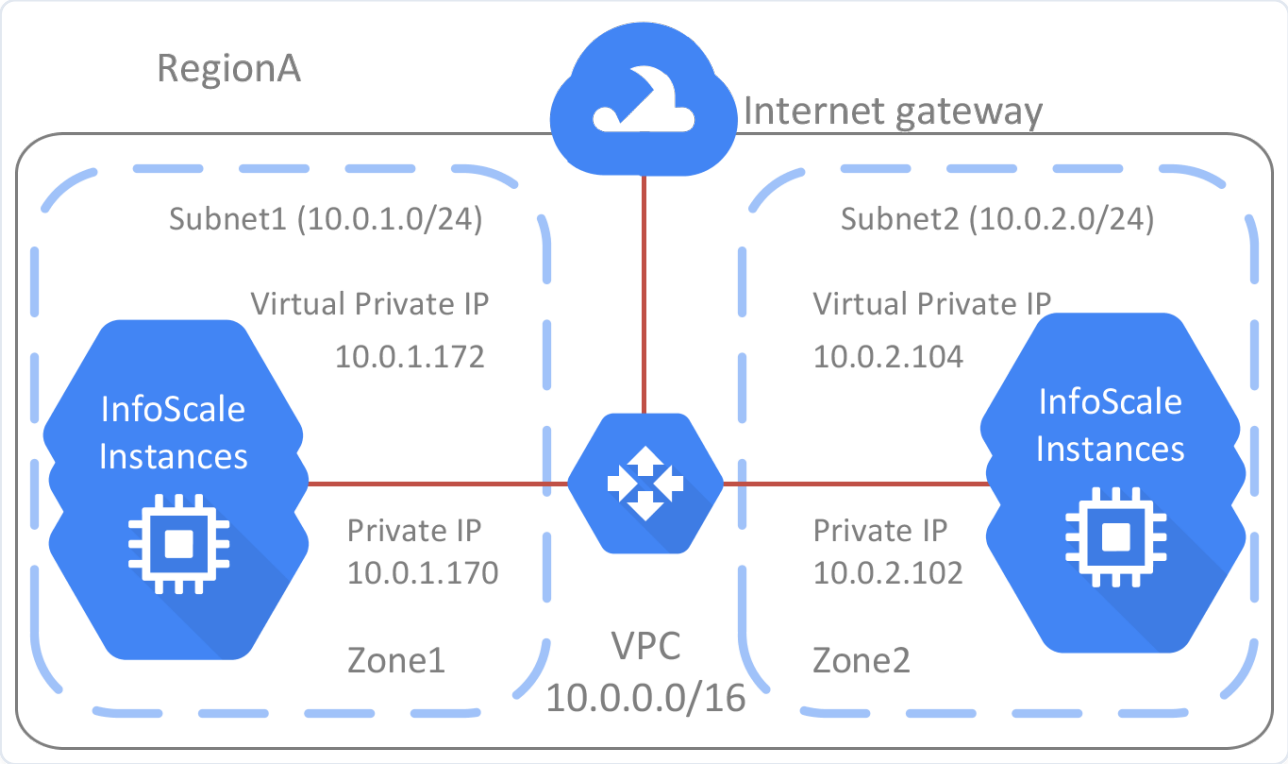
Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

Replication across zones in a GCP region - Windows

In this scenario, replication is configured between two Google cloud data centers that are located in different zones within the same region.

Figure: Sample replication configuration across zones within a Google cloud region



Setting up replication

The following procedure lists high-level tasks to set up replication for this scenario.

To set up replication across zones within the same Google cloud region

1. Create a VPC network with a valid CIDR block, for example, 10.0.0.0/16.

2. Create two subnets - one subnet for the primary site in Zone1 and the second subnet for the secondary site in Zone2 with valid CIDR range, for example, 10.0.1.0/24 and 10.0.2.0/24 respectively.
3. Launch the VM instances in the primary and secondary subnets, and install InfoScale on the instances.
4. Verify connectivity between the virtual private IP addresses of the instances.

```
\# ping PIP
```

   a. Create VxVM disk groups, VxVM volumes, Replicated Volume Group (RVG), and RLinks.
   b. Set up replication between the instances using private IP address or virtual private IP address.
   c. Verify the status of replication.

`# vradmin -g dg_name repstatus rvg_name`

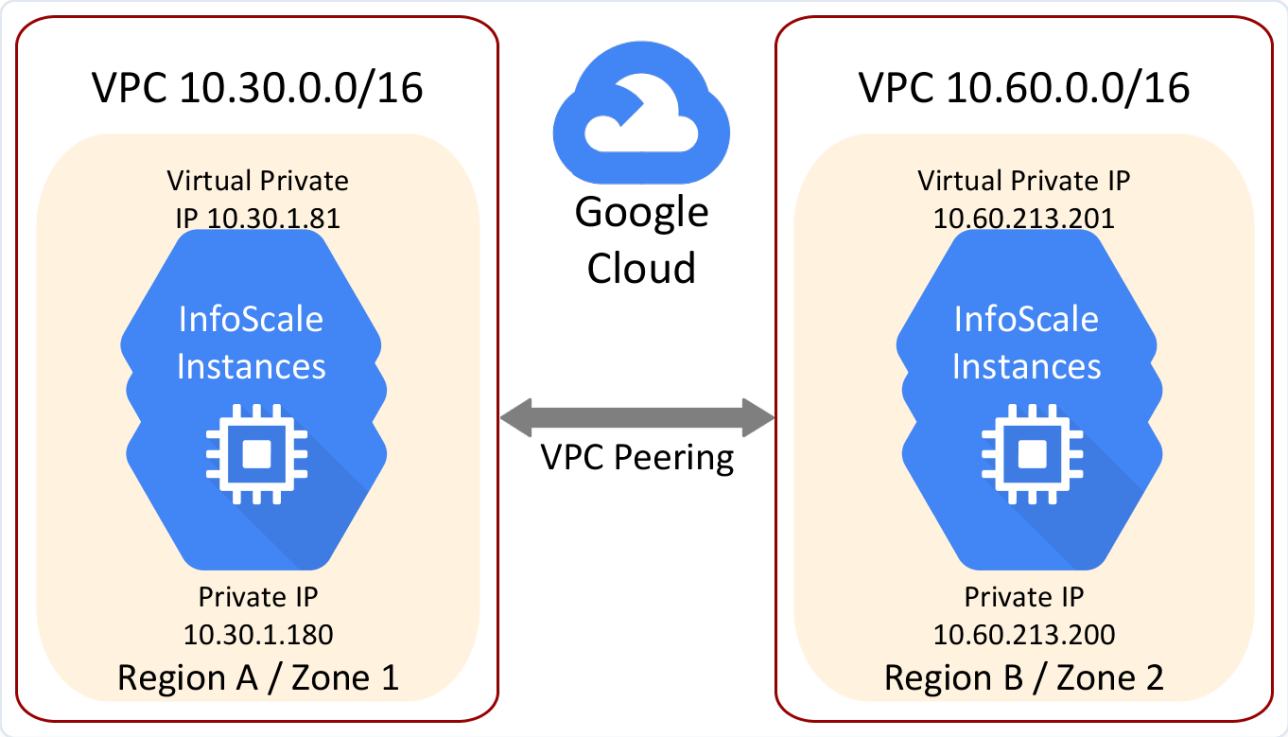
Ensure that the status shows:

```
Replication status: replicating (connected)
```

Replication across GCP regions - Windows

In this scenario, replication is configured across zones configured in different regions.

Figure: Sample replication configuration across Google cloud regions



Setting up replication

The following procedure lists high-level tasks to set up replication for this scenario.

To set up replication across Google cloud regions

1. Create two VPC networks with valid non-overlapping CIDR blocks - one subnet for the primary site in Region1 and the second for the secondary site in Region2 with valid CIDR range, for example, 10.30.0.0/16 and 10.60.0.0/16 respectively.
2. Create a primary site VM instance.
3. Create a secondary site VM instance.
4. Peer the two VPC networks.
5. Install the appropriate InfoScale product on the VM instances in both the data centers.
6. Create VxVM disk groups, VxVM volumes, Replicated Volume Group (RVG), and RLinks.
7. Set up replication between the instances using the private IP address or virtual private IP address.
8. Verify the status of replication.

```
# vradmin -g dg_name repstatus rvg_name
```

Ensure that the replication status shows:

```
Replication status: replicating (connected)
```

HA and DR configurations in GCP - Windows

InfoScale Enterprise lets you configure applications for high availability (HA) and disaster recovery (DR) in Google Cloud Platform (GCP) environment.

For more information on the supported use cases, read the following topics:

- See Failover within a subnet of a GCP zone using virtual private IP - Windows.
- See Failover across GCP subnets using overlay IP - Windows.
- See DR across GCP regions or VPC networks - Windows.

InfoScale provides the GCPIP agent to manage the network resources for cluster communication, failover, and failback. The agent monitors and manages the IP resources of cluster nodes within the Google cloud or cluster nodes that are spread across on-premises and the Google cloud. For details on the agents, see the *Cluster Server Bundled Agents Reference Guide - Windows*.

Application data is stored on persistent disks that are attached to VM instances.

InfoScale lets you configure replication between VM instances, which is further used to support various HA and DR scenarios for applications in GCP.

For details on the supported replication configurations in GCP:

See Replication configurations in GCP - Windows.

InfoScale provides configuration wizards that you can use to create, modify, and delete file share and SQL Server HA configurations.

For details about the usage of the File Share Configuration Wizard, refer to the following documents as appropriate:

- *Cluster Server Administrator's Guide - Windows*
- *Storage Foundation and High Availability Solutions, Solutions Guide - Windows*

For details about the usage of the SQL Server Configuration Wizard, refer to the following documents as appropriate:

- *Cluster Server Implementation Guide for Microsoft SQL Server - Windows*
- *Storage Foundation and High Availability Solutions HA and DR Solutions Guide for Microsoft SQL Server - Windows*

Considerations for using InfoScale Enterprise with shared disks in the GCP cloud

InfoScale Enterprise supports shared disks in a GCP cloud environment. You can configure InfoScale disk groups on shared storage disks in the GCP cloud.

Consider the following before you configure InfoScale Enterprise in the GCP cloud:

- If using shared disks in your InfoScale configuration, ensure that the SCSI3 reservation is enabled.

Run the following command on each node:

```
# vxclus usesystembus ON
```

Failover within a subnet of a GCP zone using virtual private IP - Windows

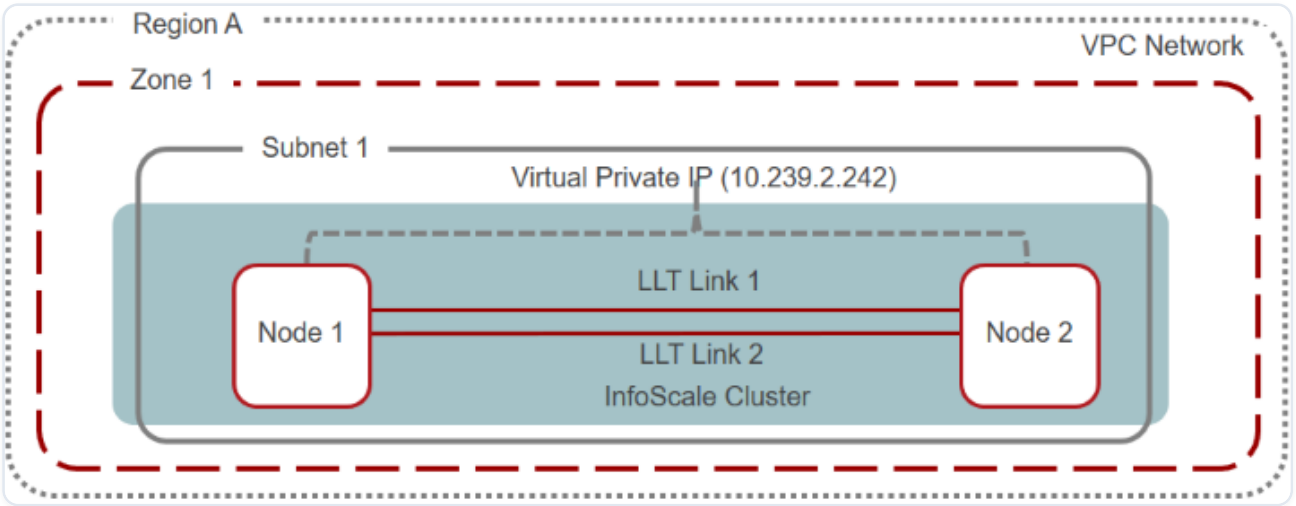
InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - within the same subnet of a zone.

The following information is required:

- The virtual private IP (secondary private IP) address to be failed over
- The device to which the IP should be plumbed
- The `google-api-python-client` GCP Python module. For details, see the *Bundled Agents Reference Guide - Windows*.

Sample configuration with private IP

The following graphic depicts a sample failover configuration within the same subnet using a virtual private IP:



The sample configuration includes the following elements:

- A VPC network is configured in Region A of the Google cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node1 and Node2, which are VM instances.
- Both the cluster nodes exist in the same subnet.
- A virtual private IP is configured, which is failed over from one node to the other as part of the failover or the failback operations.

Sample service group configuration with private IP

The following snippet is a service group configuration from a sample VCS configuration file (`main.cf`):

```
GoogleIP appsg_GoogleIP (

        PrivateIP = "10.208.8.21"

        MACAddress @cloudvm1 = "42:01:0A:D0:08:06"

        MACAddress @cloudvm2 = "42:01:0A:D0:08:05"

)



IP appsg_ip (

        Address = "10.208.8.21"

        SubNetMask = "255.255.252.0"

        MACAddress @cloudvm1 = "42:01:0A:D0:08:06"

        MACAddress @cloudvm2 = "42:01:0A:D0:08:05"

)



NIC appsg_nic (
```

```
        MACAddress @cloudvm1 = "42:01:0A:D0:08:06"
        MACAddress @cloudvm2 = "42:01:0A:D0:08:05"
)


appsg_GoogleIP requires appsg_ip
appsg_ip requires appsg_nic
```

Failover across GCP subnets using overlay IP - Windows

InfoScale clusters let you fail over IPs - and thereby, the application configured for HA - between different subnets in the same zone or in different zones.

The following information is required:

- The overlay IP address to be used for failover

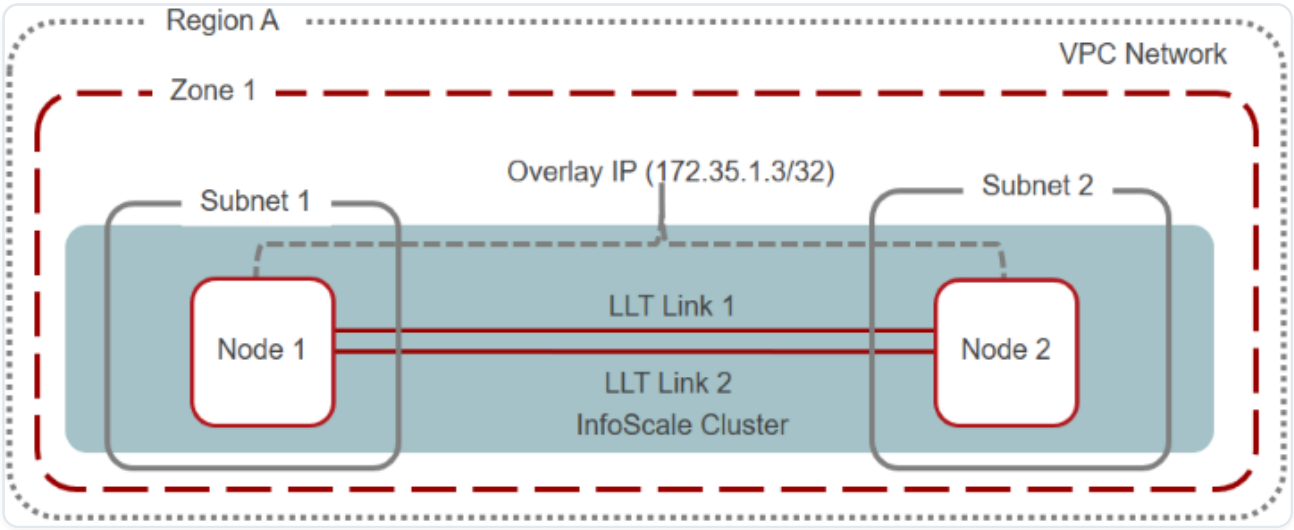
(The overlay IP address must be defined outside the CIDR block to which the cluster nodes belong.)

- The device to which the IP should be plumbed
- The `google-api-python-client` GCP Python module. For details, see the *Bundled Agents Reference Guide - Windows*.

The private IP of one subnet cannot be failed over to a different subnet. InfoScale Enterprise provides an overlay IP, which can be used across subnets.

Sample configuration with overlay IP for failover across subnets in the same zone

The following graphic depicts a sample failover configuration across subnets within the same zone using an overlay IP:



The sample configuration includes the following elements:

- A VPC network is configured in Region A of the Google cloud.
- An application is configured for HA using an InfoScale cluster that comprises two nodes, Node 1 and Node 2, which are VM instances.

Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.

- An overlay IP is configured to allow the redirection of IP address traffic to another cluster node that belongs to a different subnet during failover or failback. The overlay IP is configured outside the CIDR block of the VPC network to which the nodes belong.

Sample service group configuration with overlay IP for failover across subnets in the same zone

The following snippet is a service group configuration from a sample VCS configuration file (`main.cf`):

```
group app_network_sg (

    SystemList = { cloudvm1 = 0, cloudvm2 = 1 }

    AutoStartList = { cloudvm1, cloudvm2 }

    )



    GoogleIP appsg_GoogleIP (

           OverlayIP = "192.168.9.3"
```

```
            MACAddress @cloudvm1 = "42:01:0A:D0:08:06"
            MACAddress @cloudvm2 = "42:01:0A:D0:08:05"
    )

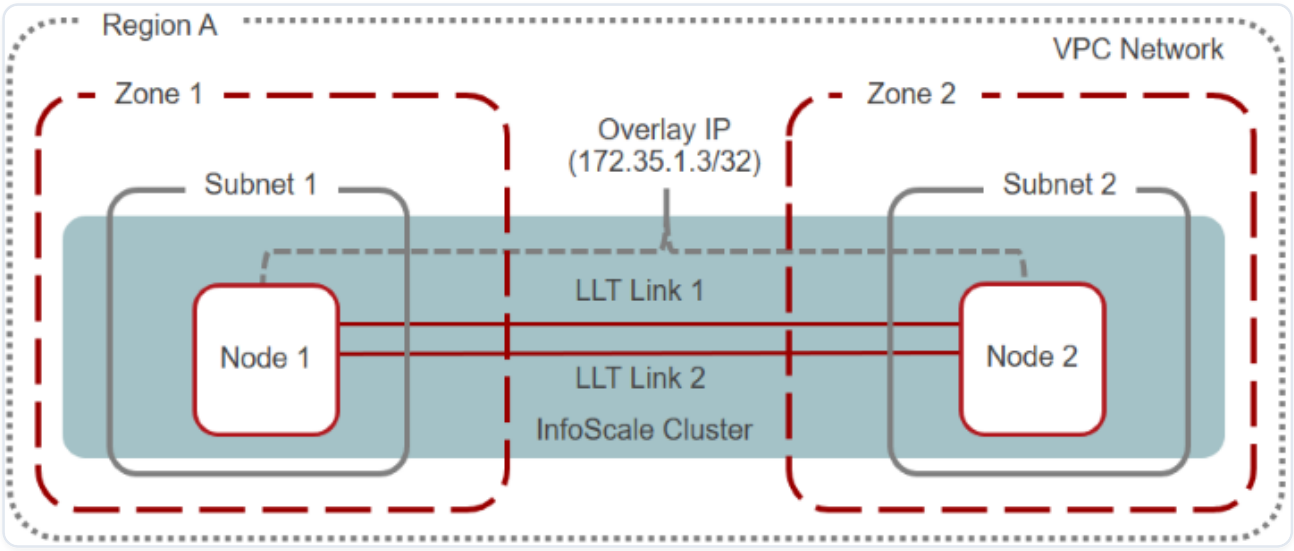
    IP appsg_ip (
            Address = "192.168.9.3"
            SubNetMask = "255.255.252.0"
            MACAddress @cloudvm1 = "42:01:0A:D0:08:06"
            MACAddress @cloudvm2 = "42:01:0A:D0:08:05"
    )

    NIC appsg_nic (
            MACAddress @cloudvm1 = "42:01:0A:D0:08:06"
            MACAddress @cloudvm2 = "42:01:0A:D0:08:05"
    )

appsg_GoogleIP requires appsg_ip
appsg_ip requires appsg_nic
```

Sample configuration with overlay IP for failover across subnets in different zones

The following graphic depicts a sample failover configuration across subnets in different zones using an overlay IP:



Sample service group configuration with overlay IP for failover across subnets in different zones

The following snippet is a service group configuration from a sample VCS configuration file (`main.cf`):

```
group app_network_sg (

    SystemList = { cloudvm1 = 0, cloudvm2 = 1 }

    AutoStartList = { cloudvm1, cloudvm2 }

    )



    GoogleIP appsg_GoogleIP (

            OverlayIP = "192.168.9.3"

            MACAddress @cloudvm1 = "42:01:0A:D0:08:06"

            MACAddress @cloudvm2 = "42:01:0A:D0:08:05"

    )



    IP appsg_ip (

            Address = "192.168.9.3"
```

```
            SubNetMask = "255.255.252.0"

            MACAddress @cloudvm1 = "42:01:0A:D0:08:06"

            MACAddress @cloudvm2 = "42:01:0A:D0:08:05"

    )



    NIC appsg_nic (

            MACAddress @cloudvm1 = "42:01:0A:D0:08:06"

            MACAddress @cloudvm2 = "42:01:0A:D0:08:05"

    )



appsg_GoogleIP requires appsg_ip

appsg_ip requires appsg_nic
```

DR across GCP regions or VPC networks - Windows

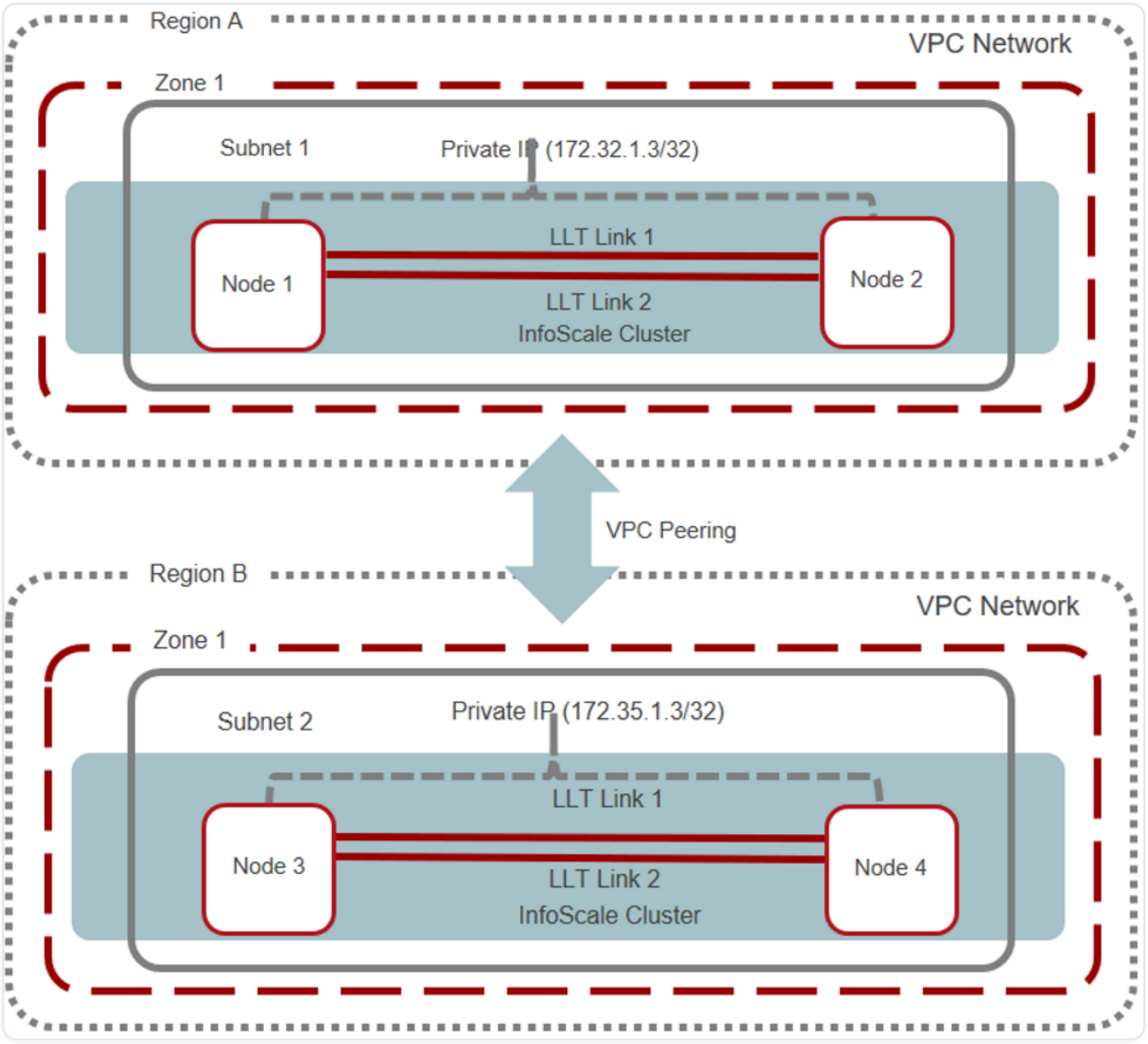
InfoScale Enterprise lets you use the global cluster option (GCO) for DR configurations. You can use a DR configuration to fail over applications across different regions or VPC networks in GCP. The cluster nodes can be in the same zone or in different zones.

The following information is required:

- VPC network peering between regions
- The IP address to be used for cross-cluster communication:
  - Virtual private IP for cluster the nodes that exist in the same subnet
  - Overlay IP for cluster the nodes that exist in different subnets
- The `google-api-python-client` GCP Python module. For details, see the *Bundled Agents Reference Guide - Windows*.

Sample configuration for DR across regions or VPC networks

The following graphic depicts a sample DR configuration across GCP regions:



The sample configuration includes the following elements:

- VPN tunnel between Region A and Region B
- The primary site has the following elements:
  - A VPC network, VPC1, is configured in RegionA of the Google cloud.
  - An application is configured for HA using an InfoScale cluster that comprises two nodes, Node 1 and Node 2, which are VM instances.

Node 1 exists in Subnet 1 and Node 2 exists in Subnet 2.

- An overlay IP is configured to allow the redirection of IP address traffic to another cluster node that belongs to a different subnet during failover or failback.
- The secondary site has the following elements:

- A VPC network, VPC2, is configured in RegionB of the Google cloud.
- The same application is configured for HA on Node 3 and Node 4, which exist in Subnet 3 and Subnet 4 respectively.
- The overlay IP allows the private IP of a node to fail over from one subnet to another in a zone.

Sample service group configuration for GCO across regions

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) at the primary site (Region A):

```
include "types.cf"



cluster democls (

    UserNames = { "ADMINISTRATOR@NEWGCPDC" = "" }

    ClusterAddress = "10.208.8.21"

    Administrators = { "ADMINISTRATOR@NEWGCPDC" }

    SecureClus = 1

    )



remotecluster DRNODE (

    ClusterAddress = "10.208.4.23"

    )



heartbeat Icmp (

    ClusterList = { DRNODE }

    Arguments @DRNODE = { "10.208.4.23" }
```

```
    )


system cloudvm1 (

    )


system cloudvm2 (

    )


group ClusterService (

    SystemList = { cloudvm1 = 0, cloudvm2 = 1 }

    AutoStartList = { cloudvm1, cloudvm2 }

    )


    GoogleIP csg_GoogleIP (

        PrivateIP = "10.208.8.21"

        MACAddress @cloudvm1 = "42:01:0A:D0:08:06"

        MACAddress @cloudvm2 = "42:01:0A:D0:08:05"

        )


    IP csg_ip (

        Address = "10.208.8.21"

        SubNetMask = "255.255.252.0"
```

```
        MACAddress @cloudvm1 = "42:01:0A:D0:08:06"
        MACAddress @cloudvm2 = "42:01:0A:D0:08:05"
        )


    NIC csg_nic (
        MACAddress @cloudvm1 = "42:01:0A:D0:08:06"
        MACAddress @cloudvm2 = "42:01:0A:D0:08:05"
        )


    Process wac (
        StartProgram @cloudvm1 = "\"C:\\Program Files\\
          Veritas\\Cluster Server\\bin\\wac.exe\""
        StartProgram @cloudvm2 = "\"C:\\Program Files\\
          Veritas\\Cluster Server\\bin\\wac.exe\""
        StopProgram @cloudvm1 = "\"C:\\Program Files\\
          Veritas\\Cluster Server\\bin\\wacstop.exe\""
        StopProgram @cloudvm2 = "\"C:\\Program Files\\
          Veritas\\Cluster Server\\bin\\wacstop.exe\""
        MonitorProgram @cloudvm1 = "\"C:\\Program Files\\
          Veritas\\Cluster Server\\bin\\wacmonitor.exe\""
        MonitorProgram @cloudvm2 = "\"C:\\Program Files\\
          Veritas\\Cluster Server\\bin\\wacmonitor.exe\""
```

```
        )


    csg_GoogleIP requires csg_ip

    csg_ip requires csg_nic

    wac requires csg_GoogleIP
```

The following snippet is a service group configuration from a sample VCS configuration file (main.cf) at the secondary site (Region B):

```
include "types.cf"


cluster DRNODE (

    UserNames = { "ADMINISTRATOR@NEWGCPDC" = "" }

    ClusterAddress = "10.208.4.23"

    Administrators = { "ADMINISTRATOR@NEWGCPDC" }

    SecureClus = 1

    )


remotecluster democls (

    ClusterAddress = "10.208.8.21"

    )


heartbeat Icmp (

    ClusterList = { democls }
```

```
    Arguments @democls = { "10.208.8.21" }
    )


system cloudvm3 (
    )


group ClusterService (
    SystemList = { cloudvm3 = 0 }
    AutoStartList = { cloudvm3 }
    )


    GoogleIP csg_GoogleIP (
        PrivateIP = "10.208.4.23"
        MACAddress = 42-01-0A-D0-04-04
    )


    IP csg_ip (
        Address = "10.208.4.23"
        SubNetMask = "255.255.252.0"
        MACAddress @cloudvm3 = "42:01:0A:D0:04:04"
    )
```

```
NIC csg_nic (
    MACAddress @cloudvm3 = "42:01:0A:D0:04:04"
)

Process wac (
    StartProgram @cloudvm3 = "\"C:\\Program Files\\
      Veritas\\Cluster Server\\bin\\wac.exe\""
    StopProgram @cloudvm3 = "\"C:\\Program Files\\
      Veritas\\Cluster Server\\bin\\wacstop.exe\""
    MonitorProgram @cloudvm3 = "\"C:\\Program Files\\
      Veritas\\Cluster Server\\bin\\wacmonitor.exe\""
)

csg_GoogleIP requires csg_ip
csg_ip requires csg_nic
wac requires csg_GoogleIP
```

Replication to and across cloud environments

This section includes the following topics:

- Data replication in supported cloud environments
- Supported replication scenarios
- Setting up replication across AWS and Azure environments

Data replication in supported cloud environments

To offer scalable, cost-effective disaster recovery (DR) options for your business, Acrtera provides the following technologies in tandem with cloud services:

- Flexible Storage Sharing (FSS) and Volume Replicator (VVR) for Linux environments

Using FSS, a technology that enables seamless cluster-wide sharing of local storage over the network, you can form clusters across zones within a region. Using VVR, you can replicate data across zones and regions for both standalone and clustered systems.

- VVR for Windows environments

You can replicate data across zones and across regions.

> **Note:** In these topics about replication,zoneis used as a common term to refer to an Availability Zone (AZ) in AWS, a VNet or a user-defined site in Azure, or a zone in Google cloud.

> With the help of these technologies, you can:

- Leverage the cloud as a DR site.
- Replicate application data from your on-premises data center to the cloud or within the cloud.
- Avoid the infrastructural costs that are needed to maintain a second physical site.

Data is always available, even if a failure occurs in a node, or a storage device, or a zone.

Supported replication scenarios

InfoScale provides the Flexible Storage Sharing (FSS) feature (on Linux) and the VVR component (on Linux and Windows) to replicate data in any of the following scenarios:

- Replication from on-premises to AWS, Azure, or Google Cloud

In this scenario, you configure replication between an on-premises data center and an AWS, an Azure, or a Google Cloud data center. The on-premises data center is the primary site and the cloud data center is the secondary site.

- Replication within a region

In this scenario, you configure replication between two cloud data centers that are located within a region. The data centers may exist either in the same zone or in different zones.

In case of Linux environments, if a network zone fails, InfoScale replicates data from one zone to the other by using VVR or Cluster Volume Replication (CVR).

In case of Windows environments, if a network zone fails, InfoScale replicates data from one zone to the other by using VVR.

- Replication across regions

In this scenario, you configure replication between two cloud data centers that are located in different regions. The primary site is in one region and the secondary site is in another region.

In Linux environments, you can protect your data from a zonal failure by using CVR to configure a cluster that spans across zones.

- Replication across clouds

In this scenario, you configure replication between an Azure cloud and an AWS cloud. One of the clouds is the primary site and the other is the DR site.

Setting up replication across AWS and Azure environments

Setting up replication across clouds is similar to setting up replication from an on-premises (primary) data center to a cloud (secondary) data center. Consider one of the clouds as the primary data center and the other as the secondary data center.

The high-level process is as follows:

1. Prepare the deployment setup in each cloud.

See Replication from on-premises to AWS - Linux.

See Replication from on-premises to Azure - Linux.

See Replication from on-premises to AWS - Windows.

See Replication from on-premises to Azure - Windows.

1. Set up a communication channel between the clouds. For example, you can set up a virtual network gateway or you can configure a VPN-based connection.
2. Install the appropriate InfoScale product on the compute instances of the primary data and the secondary data center.

For details on installation in Linux environments, refer to the *InfoScale Installation Guide - Linux*.

For details on installation in Windows environments, refer to the *InfoScale Installation and Upgrade Guide - Windows*.

1. Configure replication using the VVR option between the InfoScale nodes to which your source and target volumes are attached.

For details on configuring replication in Linux environments, refer to the *InfoScale Replication Administrator's Guide - Linux*.

For details on configuring replication in Windows environments, refer to the *InfoScale Volume Replicator Administrator's Guide* \- Windows.

Migrating files to the cloud using Cloud Connectors

This section includes the following topics:

- About cloud connectors
- About InfoScale support for cloud connectors
- How InfoScale migrates data using cloud connectors
- Limitations for file-level tiering
- About operations with Amazon Glacier
- Migrating data from on-premise to cloud storage
- Reclaiming object storage space
- Removing a cloud volume
- Examining in-cloud storage usage
- Sample policy file
- Replication support with cloud tiering

InfoScale® 9.1 Solutions in Cloud Environments - v9.1

About cloud connectors

Cloud connectors let you use cloud storage as a tier to manage your storage needs with agility and flexibility. You can build a hybrid storage environment that seamlessly integrates local on-premises storage with cloud storage.

With cloud connectors, you can migrate data from on-premises storage to cloud storage. Expensive DAS and SAN storage can thus be freed up for mission-critical data or high-performance applications while less frequently used data can be moved to the cloud tier.

About InfoScale support for cloud connectors

InfoScale supports Amazon S3, Amazon Glacier, Amazon GovCloud, Azure BLOB, and Google Cloud object storage.

Note: Only file-level tiering is supported for Amazon Glacier.

The S3 connector works with any storage that can be accessed using the standard S3 protocol, signature version 2 and 4. It is supported on systems running RHEL and SLES versions that are supported in this release.

Azure BLOB connectors work with Microsoft Azure BLOB object storage. These connectors are supported on systems running RHEL versions that are supported in this release.

Google connectors work with Google Cloud object storage. These connectors are supported on systems running Linux versions that are supported in this release.

InfoScale supports the of use cloud connectors to migrate data in the following scenarios:

- From an on-premises data center to a cloud data center
- From a cloud data center to an on-premises data center
- Across data centers in two different supported cloud environments

(for example, from an AWS cloud data center to an Azure cloud data center)

The procedure to configure migration of data is similar in all these scenarios.

See Migrating data from on-premise to cloud storage.

You use the `policy.xml` file to specify the source and the destination information that is applicable to your scenario.

See Sample policy file.

How InfoScale migrates data using cloud connectors

InfoScale supports block-level and file-level data migration to cloud. It uses the capabilities of VxFS multi-volume file system and VxVM volume sets to provide a stable infrastructure to connect on-premises and cloud environments. The cloud storage is represented as a volume in a disk group. A single file system is created over a set of volumes comprising a group of local volumes and cloud volumes. The multi-volume file system is mounted on the same mount point that was in use by the existing data volume. This configuration allows applications to seamlessly access data even after data is moved to the cloud.

A cloud volume is a regular VxVM volume with `cloud` attribute enabled. Based on the attribute setting, VxVM decides whether storage must be provisioned locally or in the cloud. The cloud volume is not limited by locally available storage capacity. Each cloud volume is associated with a target storage unit called buckets or containers (the terminology varies with the vendor). The local and cloud volumes are assigned tiers across which files can be migrated. All I/O requests on cloud volumes are managed by the `vxcloudd` daemon.

The migration is based on the policy file, `policy.xml`, which defines the storage placement policies for your data. The policy file can be customized to suit your needs. The migration can include regular files; not empty directories or symbolic links.

In case of block-level migration, data is stored in the cloud volume in blocks of fixed sizes, each block representing an object. The object can hold data from different files. The object size corresponds to the block size of the file system on the local volume. For example, a file of 10 KB stored on the local volume with a file system block size of 2k will be written to the cloud volume in 2k block sizes as 5 distinct objects.

In case of file-level migration, a single file is broken in to blocks of 64 MB and each block is stored as a single object. A single file can have one or more objects. For example, a file of 10 KB will be written in a single block as a single object. However, a file of 100MB will be broken to 2 blocks of 64 MB and 36 MB, and will be written as two objects.

In both the migration types, each volume in the volume set is assigned one of the following user-defined placement classes:

LOCAL	INDICATES THAT THE VOLUME IS A REGULAR IN-PLACE VOLUME.
CLOUD	Indicates that the volume is an off-site cloud volume with the attribute `vxcloud=on` or `fscloud=on` .

Note: For all the supported cloud connector types except Amazon Glacier, the cloud upload and retrieval operations are synchronous. SeeAbout operations with Amazon Glacier.

File movement across the local and cloud tiers is defined by the policies assigned to and enforced on the file systems.

For details on the policy file, placement policies, and rules, see the *Administering SmartTier* chapter in the *Storage Foundation Administrator's Guide - Linux*.

Figure: How InfoScale migrates data using cloud connectors illustrates the process.

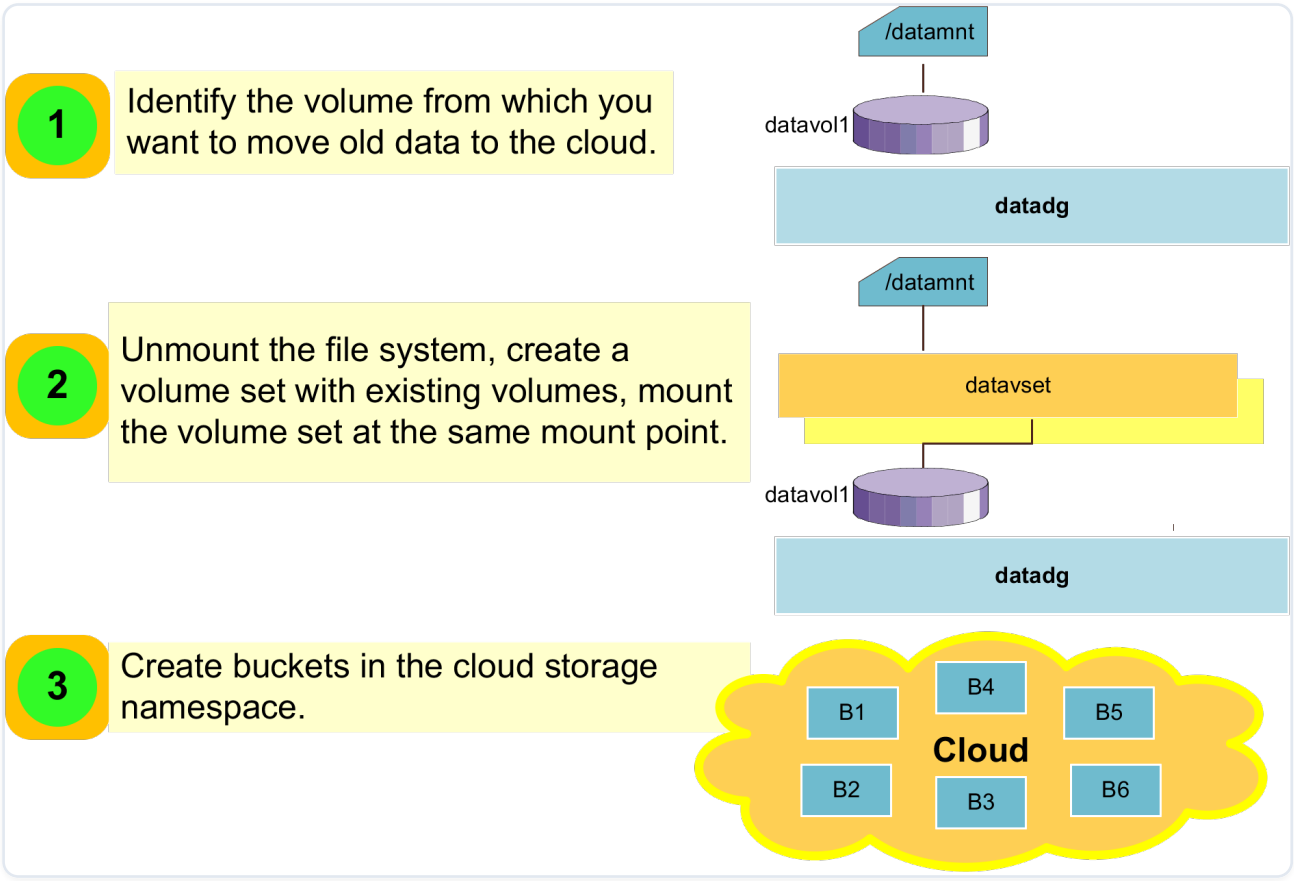
The illustration assumes the following storage placement policies.

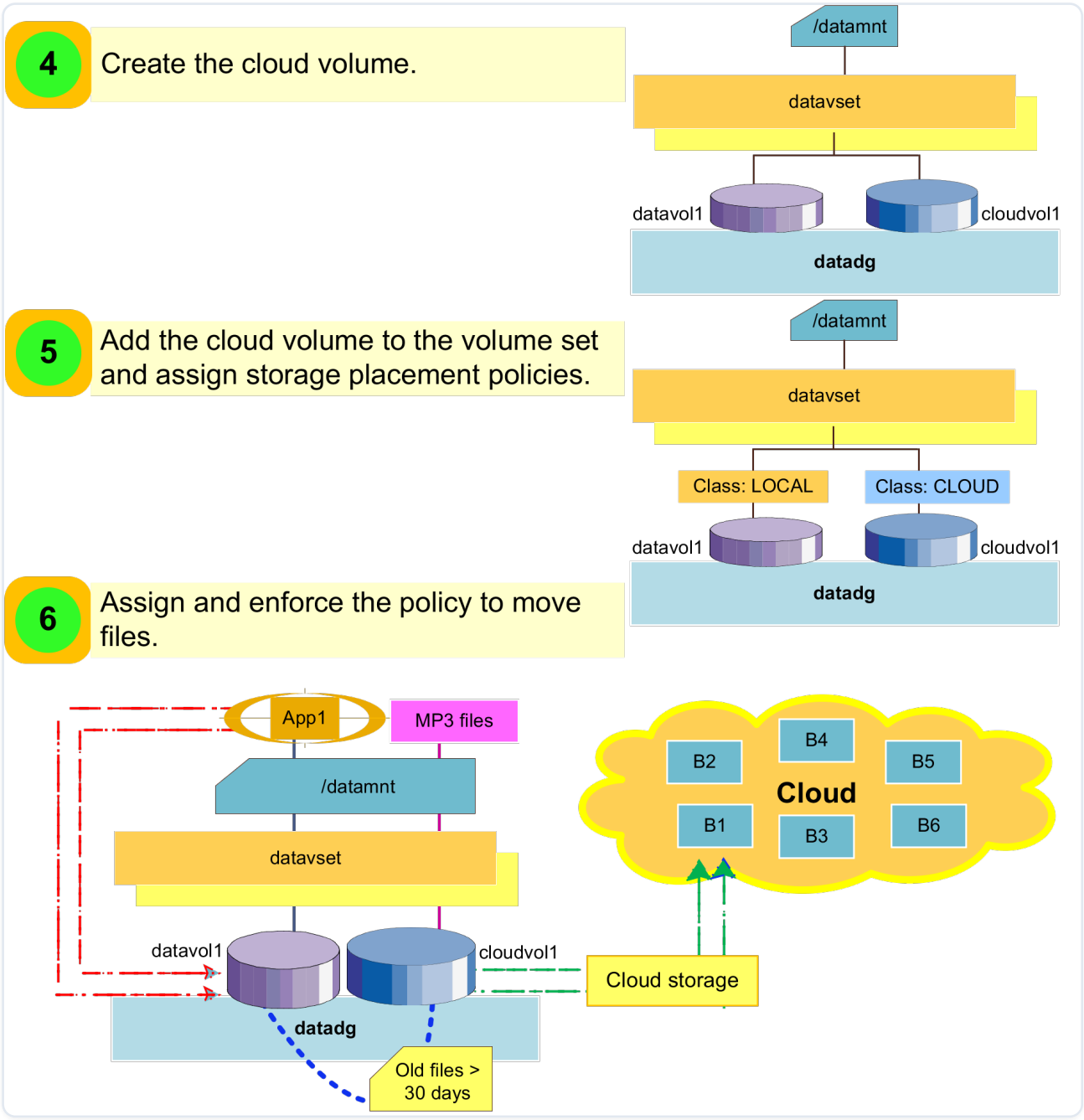
- MP3 files are stored directly on cloud volumes.

(Applicable to block-level migration only, but not in case of file-level migration.)

- Data is moved to the cloud if not accessed for 30 days.

Figure: How InfoScale migrates data using cloud connectors





Limitations for file-level tiering

InfoScale supports file-level tiering in cloud environments with the following limitations:

- A file can be present either in the cloud or on-premises, but not at both locations simultaneously. The filesystem may have a few files on-premises and few in the cloud, but at any given time, a single file can exist only at one location.
- During migration, access to files is blocked. If any related applications try to access the files under migration, they are blocked until the migration is complete.
- Migration of files from the cloud to on-premises is not supported if a size-based policy is specified.

- Write operations on files in the cloud are blocked. Only buffered read is allowed on files in the cloud.
- If you want to remove a cloud volume from MVFS, you must first move all the files that exist on the cloud volume to some other volume or to another cloud.
- Migration of compressed files is not allowed.
- Migration of memory-mapped files is not allowed.
- Migration of files that contain shared extents, like deduplicated files, is not allowed.
- The `vxcloudusage` utility scans all the system Inodes and their block maps, which may be time consuming. Therefore, Acrtera recommends that you do not run this utility frequently. Additionally, when Inodes and block maps are scanned, system caches may get recycled, which may indirectly affect the overall performance of the system. Therefore, Acrtera recommends that you do not run this utility during active production hours.
- When the Acrtera Volume Replicator (VVR) option is configured, SmartIO caching cannot be used on the target site.

The limitations for the command-line support are as follows:

- The following commands are not supported for files in the cloud:
  - `FSCACHE`
  - `SETEXT`
  - `VXCOMPRESS`
  - `VXFILESNAP`
  - `TRUNCATE`
- The following file system commands are not supported with cloud volumes:
  - `FSCDSADM`
  - `FSCDSTASK`
  - `FSCKPTADM`
- Thin reclamation is not supported for file systems on cloud devices.
- If a checkpoint is present, you cannot add a cloud volume to the filesystem.
- The fragmentation index is always zero for files in the cloud.
- Direct write operations on cloud devices are not allowed. However, the policy is invalid if it specifies file creation on a cloud volume.

About operations with Amazon Glacier

For all the supported cloud types except Glacier, the upload and retrieval operations are synchronous. For Glacier, the retrieval operation is asynchronous and the upload operation is synchronous.

Upload to Glacier

An archive is a base unit of storage, similar to objects in Amazon S3 or Azure BLOB. A vault is a container for storing archives, similar to buckets in S3. An archive is created for each on-premises file, which means that there is a 1:1 mapping of archives and files. When a file is moved to Glacier, all the data in the file is written as one archive in a Glacier vault.

Download or retrieval of an archive from Glacier

Retrieving an archive from Glacier is an asynchronous operation in which you first initiate a job, and then download the output after the job completes. When you initiate a job, Glacier returns a job ID in the response, and executes the job asynchronously. A new `fscloudd` daemon is added to poll for initiated jobs, and to check whether the job initialization is complete. When a retrieval from the Glacier tier is initiated, the `fscloudd` daemon is started internally, if it is not running already.

InfoScale supports the following types of retrievals from Glacier:

- Expedited retrievals, which are typically made available within 1 - 5 minutes.
- Standard retrievals, which typically complete within 3 - 5 hours. You can access any of your archives within several hours. When a retrieval option is not specified in a request, this option is used by default.
- Bulk retrievals, which typically complete within 5 - 12 hours. This retrieval option costs the least, and you can use it to retrieve large amounts of data, even petabytes, in a day.

To specify a retrieval type, you provide flags in the `policy.xml` file.

In the following example, `tier2` is a Glacier tier, which indicates relocation from Glacier.

```
<FROM Flags="expedited_retrieval">
  <SOURCE>
    <CLASS>tier2</CLASS>
  </SOURCE>
```

```
</FROM>
```

Flags for the other supported retrieval types are `standard_retrieval` and `bulk_retrieval`.

Relocation from Glacier is done asynchronously, and you can track it by using the `fsppadm status` command. For example:

```
\#/opt/VRTS/bin/fsppadm status /mnt1 tier2

UX:vxfs fsppadm: INFO: V-3-20000: fsppadm:

Download Processing for 5 files is remaining.



\#/opt/VRTS/bin/fsppadm status /mnt1 tier2

UX:vxfs fsppadm: INFO: V-3-20000: fsppadm:

Download Processing for this tier is not in progress.
```

Migrating data from on-premise to cloud storage

Make sure that the data you want to migrate includes regular files, not empty directories or symbolic links.

To migrate data from on-premise to cloud storage

1. Create the policy file `policy.xml`.

See the *Administering SmartTier* chapter in the *Storage Foundation Administrator's Guide - Linux*.

For a sample policy file:

See Sample policy file.

1. Create a volume set with existing volumes.

> **Note:** Unmount the file system before creating the volume set. After creating the volume set, mount it at the same mount point.

```
\# umount mount_path_of_data_volume

\# vxvset -g dg_name make vset_name local_data_volume

\# mount -t vxfs /dev/vx/dsk/dg_name/vset_name \

mount_path_of_data_volume
```

a. Create buckets/containers in the cloud storage namespace. See the related cloud vendor documentation for instructions.

b. Create the cloud volume.

For block-level data migration:

```
\# vxassist -g dg_name make cloudvol_name size vxcloud=on
```

For file-level data migration:

```
\# vxassist -g dg_name make cloudvol_name size fscloud=on
```

1. Configure the Cloud target.

CONNECTOR	COMMAND
S3	lang=bash
	\# vxcloud -g diskgroup_name \
	addcloud cloudvol_name host=host_address \
	bucket=bucket_name access_key=access_key \
	type=S3 secret_key=secret_key \
	where, secret_key and access_key are credentials to access the vendor cloud services.

CONNECTOR	COMMAND
	By default, `https` is set to `true` and `sig_version` is set to `v4` .
Glacier	lang=bash
	\# vxcloud -g diskgroup_name \
	addcloud vol_name host=host_address \
	bucket=vault_name access_key=access_key \
	secret_key=secret_key type=GLACIER
	where, secret_key and access_key are credentials to access the vendor cloud services.
	Note: Only file-level tiering is supported with Amazon Glacier.
BLOB	lang=bash
	\# vxcloud -g diskgroup_name \
	addcloud cloudvol_name \
	host=host_address bucket=bucket_name \
	access_key=access_key type=BLOB \
	endpoint=account_name
	where, access_key is the credential to access the vendor cloud services and endpoint is the storage account name of the user.
	By default, `https` is set to `true` .

CONNECTOR	COMMAND
Google Cloud	lang=bash
	\# vxcloud -g diskgroup_name \
	addcloud cloudvol_name host=host_address \
	bucket=bucket_name type=GOOGLE \
	google_config=config.json_file_path
	where, `config.json` is a file that contains the private key, `project_id` , and `client_email` values for the Google service account. Download this file in the JSON format from the Service Accounts tab of the GCP Console.
	By default, `https` is set to `true` .

> **Note:** Acrtera recommends that you associate each cloud volume with a separate bucket.

2. Add the cloud volume to the volume set:

```
\# vxvset -g dg_name addvol vset_name cloudvol_name
```

   a. Verify that the cloud volume is appropriately tagged.

```
\# fsvoladm queryflags dataonly mount_path_of_data_volume cloudvol_name
```

3. Assign placement classes to the local and cloud volumes.

```
\# vxassist -g dg_name settag local_datavol_name \

vxfs.placement_class.LOCAL

vxassist -g dg_name settag cloudvol_name \
```

```
vxfs.placement_class.CLOUD
```

a. Assign the policy to the file systems.

```
\# fsppadm assign mount_path_of_data_volume policy.xml
```

4. View an analysis report of the data transfer.

```
\# fsppadm analyze mount_path_of_data_volume
```

a. Enforce the policy to move the data between the local volume and the cloud volume.

> **Note:** You can create a cron job to schedule the migration of old data onto the cloud volume.

```
\# fsppadm enforce mount_path_of_data_volume
```

5. Verify the location of files on the local and cloud volumes:

```
\# fsmap -a list_file

\# fsmap -a /data1/*

Volume    Extent Type     File Offset     Extent Size     File

localvol         Data               0         1048576     /data1/
reports-2016-03

cloudvol         Data               0         1048576     /data1/
reports-2016-04
```

a. Check the free space and used space across volumes in the volume set using

```
\# fsvoladm list mount_path_of_data_volume

\# fsvoladm list /data1
```

```
devid    size                 used                 avail
 name

0        2097152              356360               1740792
 localvol

1        10737418240          40                   10737418200
 cloudvol
```

Reclaiming object storage space

You can remove old, unused files from cloud storage and reclaim unused spaces.

Run the following command to reclaim the storage:

```
\# /opt/VRTS/bin/fsadm -R mount_path_of_data_volume
```

> **Note:** Thefsadm -Roption is not supported on file systems when file-level cloud tiering is enabled.

Removing a cloud volume

At any given point in time, you may want to stop migrating your data and remove the created cloud volume. Depending on your requirement, you may or may not migrate the data back on premise.

To remove a cloud volume

1. Remove the volume from the file system.

`# fsvoladm remove mountpoint volname`.

1. Remove the volume from the volume set.

`# vxvset [-g diskgroup] [-f] [-c "ch_rmopt"] [-o index] rmvol volset {volumename | index]`

1. Remove target.

`# vxcloud -g diskgroup_name rmcloud cloudvol_name`

Examining in-cloud storage usage

After you migrate or reclaim data from cloud, you may want to check the storage usage. The *vxcloudusage* utility enables you to check the storage capacity consumed. The utility allows you to check the storage capacity consumed at the following levels:

- cloud vendor-specific usage
- cloud bucket-level usage
- on-premise device usage of VxFS

The usage data includes details like number of files on cloud, number of objects created on cloud, and the total cloud storage usage in KBs.

To check the in-cloud storage usage:

```
#/opt/VRTS/bin/vxcloudusage``vxcloudusage -[b] mountpoint
```

To check the cloud vendor-specific usage:

```
# /opt/VRTS/bin/vxcloudusage /mnt1
```

For example,

```
CLOUD                    FILES          OBJECTS              SIZE(KB)

Onpremise                31105                0                 21

s3.amazonaws.com             5               20              10240
```

To check bucket-level usage:

```
# /opt/VRTS/bin/vxcloudusage -b /mnt1
```

```
BUCKET/VOLT          CLOUD                  FILES        OBJECTS
   SIZE(KB)

Onpremise            Onpremise              31105              0
     21

optimusprime-test    s3.amazonaws.com           4             16
   8192
```

```
optimusprime2        s3.amazonaws.com          1              4
   2048
```

Sample policy file

The following sample policy file defines the following policies:

- Files which are not accessed for more than 30 days are moved to the cloud tier.
- MP3 files are stored on the cloud tier.

(Applicable to block-level migration only, but not in case of file-level migration.)

- All other files are created on the local tier.

```
<?xml version="1.0"?>

<!DOCTYPE PLACEMENT_POLICY SYSTEM "/opt/VRTSvxfs/etc/
placement_policy.dtd">

<PLACEMENT_POLICY Version="5.0" Name="2tier_write_policy">

    <RULE Flags="data" Name="Regular-Files-Rule">

        <COMMENT>

        Files which have not been accessed within the past 5 seconds

        and which are on tier2 will be moved to LOCAL tier i.e. tier1

        </COMMENT>

        <SELECT Flags="Data">

            <PATTERN> * </PATTERN>

        </SELECT>

        <CREATE>

            <ON>
```

```
            <DESTINATION>
                <CLASS> tier1 </CLASS>
            </DESTINATION>
        </ON>
    </CREATE>
    <RELOCATE>
        <FROM>
            <SOURCE>
                <CLASS> tier1 </CLASS>
            </SOURCE>
        </FROM>
        <TO>
            <DESTINATION>
                <CLASS> tier2 </CLASS>
            </DESTINATION>
        </TO>
        <WHEN>
            <ACCAGE Units="days">
                <MIN Flags="gt"> 30 </MIN>
            </ACCAGE>
        </WHEN>
    </RELOCATE>
```

```
    </RULE>

</PLACEMENT_POLICY>
```

Replication support with cloud tiering

The VVR component of InfoScale supports synchronous and asynchronous block-level replication with cloud tiering.

Usage: To use this functionality, add the local volume and the corresponding cloud volume to the volume set that is configured for VVR replication.

Prerequisite: The cloud volumes on the primary and the secondary site must be connected to the same cloud connector and bucket.

For details on configuring replication with VVR, refer to the *InfoScale Replication Administrator's Guide*.

Configuration for Load Balancer for AWS and Azure - Linux

This section includes the following topics:

- Load Balancer for AWS and Azure on Linux

Load Balancer for AWS and Azure on Linux

A Load Balancer lets you distribute traffic to your backend virtual machines. A Load Balancer is a fully managed service that provides high availability for your applications. With Standard Load Balancer, you can scale your applications and create highly available services. Load Balancer supports both inbound scenarios and outbound scenarios. A Load Balancer provides low latency and high throughput and scales up to millions of flows for all TCP applications as well as UDP applications.

Sample configuration

```
group azure-lb-grp (
        SystemList = { ip-10-10-10-8 = 0, ip-10-10-10-9 = 1 }
        )


        NIC nicres (
                Device = eth0
                )


        IP ipres (
                Device = eth0
                Address = "10.10.10.10"
```

```
                NetMask = "255.255.255.0"
                )


        Process port_app (
                PathName = "/port_probe.sh"
                Arguments = "port_app 5000"
                PidFile = "/var/VRTSvcs/lock/volatile/port_app_pid"
                )


        ipres requires nicres
        ipres requires port_app
```

The contents of the `port_probe.sh` file are as follows:

```
cat /port_probe.sh
\#!/bin/bash
res=$1
port=$2
pid_file="/var/VRTSvcs/lock/volatile/${res}_pid"
NC="/usr/bin/nc"


old_pid=`ps -eo pid,args  | grep -w "nc"  | grep -w $port$
| awk '{ print $1 }`
```

```
if [ -n $old_pid ]; then

        kill -9 $old_pid

fi



$NC -l -k $port &

pid=$!

echo $pid > $pid_file



exit 0
```

Troubleshooting issues in cloud deployments

This section includes the following topics:

- In an Azure environment, you may encounter failures when exporting non-shared disks for FSS

In an Azure environment, you may encounter failures when exporting non-shared disks for FSS

For Flexible Storage Sharing (FSS), you must first export all the non-shared disks for network sharing.

If your deployment setup involves JBODs, you may encounter the following failures while exporting the non-shared disks:

- The disk export operation fails as described in the following example:

```
# vxdisk export DiskName
```

```
VxVM vxdisk ERROR V-5-1-531 Device DiskName: export failed:

Disk not supported for FSS operations
```

- The `checkfss` command fails as described in the following example:

```
# vxddladm checkfss DiskName
```

```
VxVM vxddladm INFO V-5-1-18714 DiskName is not a valid disk

for FSS operation
```

This issue occurs if a JBOD definition is not added to the disks.

Workaround:

Before exporting a disk for network sharing, add a JBOD definition on the disk.

> **Note:** You can add a JBOD definition to a disk only if its SCSI inquiry supports a unique serial number. If the SCSI inquiry is unable to locate a unique serial number, you cannot export the disk for network sharing.

> To add a JBOD definition to a disk, perform the following steps:

1. Run a query on the standard disk pages (0, 0x80, and 0x83) to find the available unique serial number.

For page 0:

```
# /etc/vx/diag.d/vxscsiinq -d /dev/vx/rdmp/DiskName
```

For page 0x80:

```
# /etc/vx/diag.d/vxscsiinq -d -e 1 -p 0x80 /dev/vx/rdmp/DiskName
```

For page 0x83:

```
# /etc/vx/diag.d/vxscsiinq -d -e 1 -p 0x83 /dev/vx/rdmp/DiskName
```

Sample output of the command for page number 0x83 that contains the unique serial number:

```
------- Identifier Descriptor 1 -------

ID type             : 0x1 (T10 vendor ID based)

Protocol Identifier : 0x0

Code set            : 0x1

PIV                 : 0x0

Association         : 0x0

Length              : 0x18

Data                : 4d5346542020202069c0ae2f82ab294b834866ff...

        /dev/vx/rdmp/DiskName: Raw data size 32

Bytes:  0 -  7  0x00  0x83  0x00  0x1c  0x01  0x01  0x00  0x18 ........
```

```
Bytes:  8 - 15  0x4d  0x53  0x46  0x54  0x20  0x20  0x20  0x20 MSFT

Bytes: 16 - 23  0x69  0xc0  0xae  0x2f  0x82  0xab  0x29  0x4b i../..)K

Bytes: 24 - 31  0x83  0x48  0x66  0xff  0x1d  0xd3  0xf5  0xcb .Hf.....
```

1. Note the following values from the command output in step 1:

   - opcode = 0x12 (18)

This is a standard opcode.

- pagecode = page number that contains the unique serial number

For example, 083.

- offset = byte offset where the serial number starts

For example, in the output here, the offset value is 8.

- length = length of the unique serial number that is provided in the Length field

For example, 24 or 0x18.

1. Add the JBOD definition.

```
# vxddladm addjbod vid=vendorid serialnum=opcode/pagecode/offset/length
```

For example:

```
# vxddladm addjbod vid=MSFT serialnum=18/0x83/8/24
```

1. Scan the disks.

```
# vxdisk scandisks
```

1. Verify whether the JBOD definition has been added sucessfully.

```
# vxddladm checkfss DiskName
```

The command output displays a confirmation message.