VOR Stream on Azure: Horizontal Scaling Guide



FRG | Financial Risk Group

Revision History

Version	Date	Description
v0.1	10/15/2026	Initial draft

Contents

Overview 3
How Horizontal Scaling Works 3
Definitions 3
Architecture 4
Sizing 4
Network Requirements 5
How-To 6
Task A: Prepare the Azure Infrastructure 6
Step 1: Create the VMs 6
Step 2: Let the Midtier Start and Stop Compute VMs 6
Step 3: Set Up Shared Storage 6
Task B: Deploy VOR Stream 7
Step 1: Write the Inventory 7
Step 2: Start Every VM 8
Step 3: Run the Playbook 8
Step 4: Deallocate the Compute VMs 9
Task C: Verify Scaling 9
Task D: Upgrade or Redeploy 9
Task E: Add a Compute VM 9
Task F: Remove a Compute VM 9

Overview

Purpose: Explains how to deploy VOR Stream on Microsoft Azure with a pool of compute VMs that VOR Stream starts and stops on demand, so compute is only paid for while runs are executing.

Intended Audience: The infrastructure and platform team building the Azure environment and running the VOR Stream deployment. It assumes familiarity with Azure and with Ansible.

Scope: Covers what VOR Stream needs from Azure for horizontal scaling. Virtual networks, VM builds, backups and monitoring follow your own standards.

Note: *This guide assumes you already know the standard VOR Stream infrastructure and network requirements. It builds on them with what you need to run VOR Stream with horizontal scaling on Azure.*

How Horizontal Scaling Works

- The VOR Stream Midtier starts a compute VM when a run is submitted and deallocates it when the run ends.
- Each run executes on **one** compute VM, and each compute VM executes **one run at a time**.
- More compute VMs means more runs can execute at the same time. It does not make a single run faster; for that, use a larger VM size.
- Runs are **not queued**. If every compute VM is busy, a new run fails instead of waiting, so provide as many compute VMs as the number of runs you expect at the same time.
- Users run `vor` CLI commands on the Midtier VM, not on a compute VM. Compute VMs are deallocated whenever no run needs them, so they are not available for interactive use.

Definitions

Table 1 - Definitions

Term	Definition
Midtier	The VOR Stream service (`super`) that runs processes and starts and stops compute VMs.
Compute VM	A VM in the `[compute]` inventory group that executes runs. The only tier that scales out.
Run	One execution of a VOR Stream process, submitted from the workbench or CLI.
Playpen	A VOR Stream project directory. Must be on shared storage visible to the Midtier and every compute VM.
Deallocate	Stopping an Azure VM so that its compute is no longer billed. Disks are kept.
IMDS	Azure Instance Metadata Service (`169.254.169.254`), which tells a VM its own name and resource ID.

Term	Definition
Consul	VOR Stream's service registry. Compute VMs are registered in it during deployment.

Architecture

The VOR Stream inventory (`hosts.ini`) assigns VMs to **host groups**. Each group stands for a VOR Stream service, and the playbook installs that service on every VM listed under it. A VM can belong to several groups, so services that are sized alike can share a VM; the Web VM in this guide, for example, is listed under `django` and `web`.

Only the compute tier scales out. Every other service runs once, or as a cluster for availability, regardless of how many compute VMs you have.

Table 2 - Host groups

Ansible group	Count	Purpose
`compute`	One per concurrent run	Runs VOR Stream processes. The only group that scales out; must be dedicated VMs.
`super`, `sdk`	One	The Midtier. `super` manages playpens and jobs and starts, stops and dispatches runs to the compute VMs; `sdk` is the SDK API (gRPC), which runs call for configuration, secrets and other services. Must **not** also be in `[compute]`.
`rabbitmq`	One, or a cluster	Message broker that passes data between the nodes of a run.
`django`, `web`	One	The application: `django` is the backend API and `web` the browser workbench.
`postgres`	One	The application database used by Django. An external PostgreSQL or SQL Server database can be used instead.
`vault`, `consul`	One, or a cluster	`vault` stores secrets and issues credentials; `consul` provides service discovery and shared configuration.

Shared storage, such as an NFS server, is not part of the inventory. It holds the playpens and is mounted on the Midtier and every compute VM (Task A, Step 3).

Sizing

Recommended starting sizes. Adjust them once you have observed your own workloads.

Table 3 - Recommended Azure VM sizes

Tier	Small	Large
Compute	`Standard_F4als_v7` to `Standard_F16als_v7`, `Standard_D8as_v7`, `Standard_D16as_v7`	`Standard_D32as_v7`, `Standard_D64as_v7`, `Standard_F64als_v7`
Midtier (`super`, `sdk`)	`Standard_D4as_v7`	`Standard_D8as_v7`
RabbitMQ	`Standard_E2as_v7`	`Standard_E4as_v7`, `Standard_E8as_v7`
Web (`django`, `web`, `postgres`)	`Standard_F2als_v7`	`Standard_D4as_v7`, `Standard_D8as_v7`, `Standard_F8als_v7`
Consul / Vault	`Standard_B2ls_v2`, `Standard_F1als_v7`	`Standard_D2as_v7`, `Standard_D4as_v7`

- Avoid burstable (B-series) sizes for compute and RabbitMQ; runs are sustained load.
- Builds and runs execute on the compute VMs, not the Midtier.
- The v7 sizes are not offered in every Azure region. Check that they are available in your region before you deploy.
- When you create a VM with a v7 size, choose a **Gen2** RHEL image. Azure rejects images that are not supported for these sizes.

Network Requirements

These are the flows that horizontal scaling adds: traffic to and from the compute VMs, the shared storage, and the Midtier's calls to Azure. If you restrict traffic with network security groups or firewalls, allow them in addition to your standard VOR Stream rules. Ports are the defaults; if you override one in the inventory, use your value.

Table 4 - Network flows for horizontal scaling

From	To	Port	Purpose
Midtier	Compute VMs	TCP 23234	Runs (SSH)
Midtier, compute VMs	NFS server	TCP 2049	Shared storage, if you use an NFS server (Task A, Step 3)
Midtier	Azure Resource Manager (`management.azure.com`)	HTTPS 443	Start and deallocate compute VMs

Compute VMs also need the same access to RabbitMQ, Vault, Consul and the Midtier as any VOR Stream host.

The Midtier reaches each compute VM by the hostname the VM reported during deployment (its `ansible_fqdn`), so the Midtier must be able to resolve those names. With Azure-provided

DNS this works by default. With custom DNS servers, `ansible_fqdn` is whatever your DNS returns for each VM, so make sure those names resolve from the Midtier.

How-To

Complete the tasks in order. Task A prepares Azure; Task B deploys VOR Stream; Task C confirms scaling works. Tasks D to F are for later changes.

Task A: Prepare the Azure Infrastructure

Step 1: Create the VMs

- Operating system: Red Hat Enterprise Linux 8, 9 or 10.
- Create compute VMs with **Regular** priority. Spot VMs can be evicted mid-run and may not have capacity when the Midtier needs to start them.
- Keep the Midtier on its own VM, separate from the compute VMs.

Note: *During deployment, the playbook reads each compute VM's name and resource ID from IMDS (`169.254.169.254`), and at runtime the Midtier gets its managed identity token from it. It is reachable by default; do not block it.*

Step 2: Let the Midtier Start and Stop Compute VMs

Create a custom role with these actions:

```
Microsoft.Compute/virtualMachines/start/action
Microsoft.Compute/virtualMachines/deallocate/action
Microsoft.Compute/virtualMachines/read
```

- Assign the role to a **managed identity** on the Midtier VM, scoped to the compute VMs or the resource group that holds them.
- Only the Midtier VM needs this role. Compute VMs need no Azure permissions.

Step 3: Set Up Shared Storage

Put playpen directories on shared storage and mount it at the **same path** on the Midtier and every compute VM. This guide uses an NFS server VM as an example; you are free to use any other shared storage instead, set up according to its own requirements. The examples below use `/opt/playpens`; use the path your users will create playpens under.

Service user: The playbook creates the `vrisk` user and group without fixed IDs, so the IDs can differ between VMs, and NFS checks numeric IDs, not names. Before running the playbook, create them with the same IDs on the Midtier, every compute VM and the NFS server. Use any ID that is free on all of them; the playbook keeps an existing user.

```
sudo groupadd -g 1001 vrisk
sudo useradd -u 1001 -g 1001 vrisk
```

NFS server: Export the playpen directory to the virtual network. Replace `10.0.0.0/16` with your subnet. If firewalld is running on the NFS server, the last two commands allow NFS through it.

```
sudo dnf install -y nfs-utils
sudo mkdir -p /export/playpens
sudo chown vrisk:vrisk /export/playpens
sudo chmod 2775 /export/playpens
echo '/export/playpens 10.0.0.0/16(rw,sync,no_subtree_check)' | sudo tee -a
/etc/exports
sudo systemctl enable --now nfs-server
sudo exportfs -ra
sudo firewall-cmd --permanent --add-service=nfs
sudo firewall-cmd --reload
```

Midtier and compute VMs: Mount the share. The last command checks that `vrisk` can write to it.

```
sudo dnf install -y nfs-utils
sudo mkdir -p /opt/playpens
echo 'nfs-server.example.com:/export/playpens /opt/playpens nfs defaults,nofail 0
0' | sudo tee -a /etc/fstab
sudo systemctl daemon-reload
sudo mount -a
sudo -u vrisk touch /opt/playpens/.write-test && sudo -u vrisk rm
/opt/playpens/.write-test
```

Note: *If your processes write files outside the playpen that are needed after the run, such as reports users download, put that directory on the shared storage as well and mount it on every VM that reads it.*

Task B: Deploy VOR Stream

Step 1: Write the Inventory

Start from `sample_hosts.ini`, provided with this guide. It contains only the host groups and variables related to horizontal scaling. Merge its groups and variables into the `hosts.ini` that ships in the root of the installation package and add the rest of your deployment settings there. List every compute VM under `[compute]` and put the Midtier on its own VM.

```
# Sample VOR Stream inventory for a horizontally scaled Azure deployment.
# Only the host groups and variables related to horizontal scaling are shown.
# Replace the example hostnames, then merge these groups and variables into the
# hosts.ini that ships with the installation package (keep its existing settings).

[all:vars]
# With Azure-provided DNS, ansible_fqdn resolves to <vm-
name>.internal.cloudapp.net,
# so set these names explicitly. rabbitmq_host applies only to a single
# RabbitMQ broker; leave it out for a cluster.
web_server_name=web.example.com
django_server_name=web.example.com
rabbitmq_host=rabbitmq.example.com

# One line per compute VM; each VM runs one run at a time.
# Do not list the Midtier VM here.
[compute]
```

```
compute01.example.com
compute02.example.com
compute03.example.com

# The Midtier starts and stops the compute VMs.
[super]
midtier.example.com

[sdk]
midtier.example.com

[django]
web.example.com

[web]
web.example.com

[postgres]
db.example.com

[rabbitmq]
rabbitmq.example.com

[vault]
hashi.example.com

[consul]
hashi.example.com

[consul_ui]
web.example.com

[compute:vars]
# Port the Midtier uses to reach the compute VMs (default 23234).
# If you change it, open the new port in the network rules.
# compute_port=23234
```

Note: *A VM listed in both `[compute]` and `[super]` is registered as `on-prem`: the Midtier never starts or stops it, and it accepts any number of runs at once. The shipped `hosts.ini` puts every compute VM in `[super]` and `[sdk]` through its `[super:children]` and `[sdk:children]` sections.* ***Delete the [super:children] and [sdk:children] lines and the compute line under each****, or scaling is silently disabled; keep `[super:vars]` and `[sdk:vars]`. Keep the Midtier out of `[compute]`.*

Step 2: Start Every VM

Ensure every VM in the inventory is running before you run the playbook.

Step 3: Run the Playbook

The Ansible control host needs Python 3.10 or later. On RHEL 8 or 9, install Python 3.11:

```
sudo dnf install python3.11 python3.11-pip
```

Then, from the root of the extracted VOR Stream installation package, install Ansible and run the playbook. Replace `azureuser` with the VM admin user, which needs sudo on every VM.

```
./setup-ansible-offline.sh
source venv/bin/activate
ansible-playbook -i hosts.ini playbook.yaml -u azureuser
```

Step 4: Deallocate the Compute VMs

Deallocate them with **Stop** in the Azure portal. The Midtier starts them when runs need them. Do not shut them down from inside the operating system: the VM then shows as **Stopped** and is still billed until a run next uses it.

Task C: Verify Scaling

1. In the Consul UI, open Services > compute and confirm each compute:<fqdn> instance shows provider = azure.
2. Submit a run from the VOR Stream workbench.
3. In the Azure portal, one compute VM should move to **Running** within a few minutes.
4. When the run completes, that VM should return to **Stopped (deallocated)**.
5. To check concurrency, submit as many runs at once as you have compute VMs. Each should start on a different VM.

Task D: Upgrade or Redeploy

Ensure every VM is running before you run the playbook, so each one is updated and the compute VMs are re-registered. Deallocate the compute VMs afterwards.

Task E: Add a Compute VM

1. Create the VM to the same specification as the existing compute VMs.
2. Make sure the Midtier's role assignment covers it (automatic if the role is scoped to the resource group).
3. Mount the shared storage as in Task A, Step 3.
4. Add it to `[compute]`, ensure every VM is running, and re-run the playbook.

Task F: Remove a Compute VM

Compute registrations are made from the Midtier and are not cleaned up automatically. Remove the VM from `[compute]`, then deregister it on the Midtier VM as `vrisk`. The service ID is `compute:` followed by the VM's FQDN as the VM reported it during deployment, which can differ from its inventory name.

```
sudo -iu vrisk
consul services deregister -id compute:<compute-vm-fqdn>
```

Delete the VM only after it has been deregistered. While a deleted VM is still registered, every run fails.