Learn more about HA

Last updated by | Hamza Aqel | Jan 10, 2023 at 7:29 AM PST

Contents

Public Documentation:

You can review our public documentation High availability concepts in Azure Database for PostgreSQL - Flexible Server.

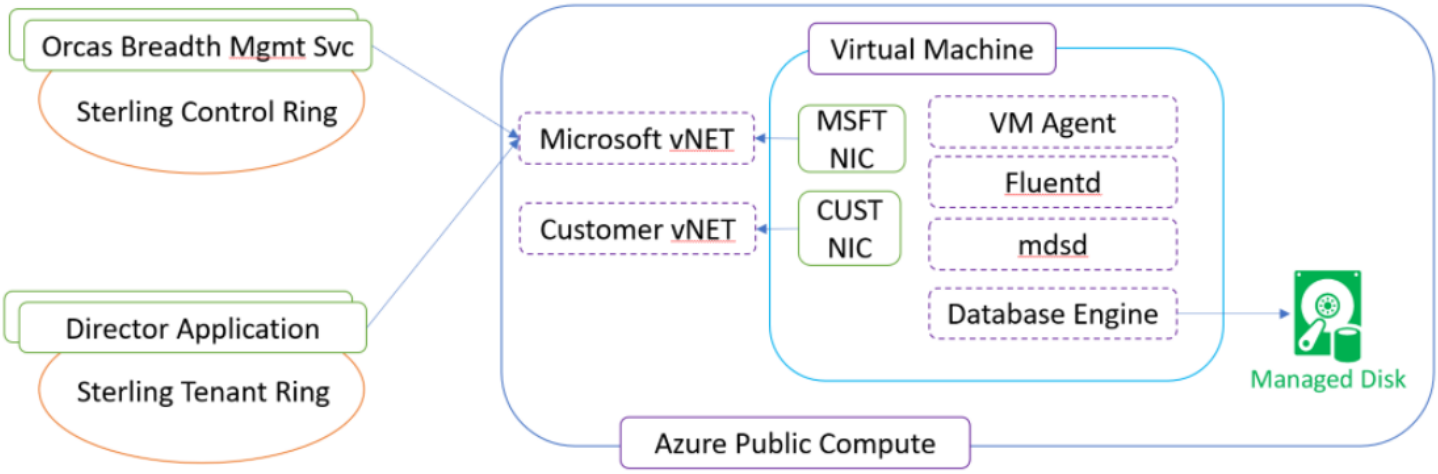
Internal notes only:

Orcas Breadth: High Availability (HA)

This document discusses the availability characteristics and the High Availability offering options for Orcas Breadth service. It doesn't cover disaster recovery (DR/BCDR) scenarios.

Orcas breadth design overview

The below diagram provides a high-level overview of the breadth service and the major components of the system.



Availability

Availability of the Orcas Breadth service can be perceived and defined across 2 broad set of scenarios. Provisioning and Connectivity. This document focuses on the Connectivity aspect i.e. the availability of a provisioned DB server. And for context, availability SLA in Azure SQL DB and Orcas is calculated using the customer perceived/observed connectivity – a server is deemed unavailable if there were >0 failed connection attempts in a 30 sec window and 0 successful connections during the same period.

In Orcas Breadth, once the server is provisioned, the Orcas control plane does not play any part in the data plane operations, i.e. the connectivity workflow doesn't have any dependencies on the control plane and the factors that impact the connectivity to the DB engine are confined to the DB engine host. Further, within the DB engine host except for the DB engine container that is hosting the DB engine no other component is involved in the connectivity workflow and their availability or the lack thereof has no impact on the ability to connect to the DB engine. Essentially,

1. Virtual Machine that is hosting the DB engine
2. DB engine process itself (we will refer to this as the DB container) and
3. remote storage that is used by the DB engine are the items that will determine the availability of the DB engine for customer connections.

Assuming Linux VM keeps 99.98~99.99% uptime, plain vanilla Orcas breadth Linux VMs will get uptime of **99.9832% ~ 99.9916%.**

Events that can impact the availability of the VM and the DB engine can be broken down into the following two buckets:

1. **Customer initiated** including maintenance events within managed maintenance window (MMW) a. Planned upgrades/updates for the DB engine.
    1. Patches
    2. SLO updates
    3. Configuration changes like server parameters that require restarts

    b. Planned upgrades/updates for updating the OS c. Planned upgrades/updates to the host OS
2. **Unplanned** unavailability

a. Unplanned down time of the DB engine.

1. Crashes
2. High resource usage

b. Unplanned VM downtime, see above for the analysis and the breakdown

Unplanned DB Engine down time

A customer will not be able to log into the DB engine when the DB engine crashes. In Orcas breadth, the DB engine is used as is (with some changes like telemetry plumbing for GDPR scenarios in PG etc.). This removes the wide range of crashes that have been observed in the Orcas service that were caused by mostly the fault points in the integration with the compute (SQLPAL) and storage abstractions (SBS). For crashes in the DB engine that happen due to other reasons (bugs etc.), the docker container is restarted to mitigate the process crash. That leaves us with the case of persistent and consistent crashes in a loop (is this feasible in PG/MySQL). <Discuss the detection and mitigation for the consistent crash>

Planned updates to the guest OS

Planned updates to the DB Engine

Unplanned VM downtime

HA Options

There are a set of core primitive operations that are critical to High Availability topology. These include:

1. Detection of the failure
2. Mitigation for the failure by switching over a. Compute b. Storage c. Connections [Networking]

Detection of the failure

Can be done by

1. An external monitor (like Director, which pings the VM and the DB for health)
2. An internal monitor (like Docker Health Status check, which connects and executes a query against the DB)
3. A triangulated signal from the internal and external monitors

Compute Switchover

1. Rebuild a Compute VM.
2. A set of spare VMs (not necessarily at 1:1 ratio but a small percent of the total nodes)
3. Offload compute to a service like ACI or Service Fabric Mesh

Storage Switchover

1. Managed Disks - Attach/detach
2. PFS - Multi attach, switch primary (determine how this will work for the spare nodes case)

Connection Switchover

In all the options below connection switch over will be done via updating the DNS record. We can set the TTL for DNS record to be low, so the connections will get routed to new VM after the failover. If customer is using Public IP then we will associate same with new VM so no need to update the DNS record however in case of private IP this will need the update.

Standalone VM

Offer standalone VM SKU along with SKU with AZ option where customer will pay more for warm compute.

- Director will detect the failure of 'primary' compute and failover to the secondary compute.
- Storage options will have impact on the down time for customer

1. Managed disk – Detach attach times would cause unavailability in the order of 2-3 minutes.
2. PFS – This will be quick since warm compute can read from PFS at the same time as primary compute is writing to it. So the only time is time to detect failure and trigger 'failover'.