Instance Pools

Last updated by | Vitor Tomaz | Nov 16, 2022 at 12:58 PM PST

Azure SQL Instance pools is now in public preview.

Take a look at our Azure update blog post here: https://azure.microsoft.com/en-in/updates/azure-sql-database-instance-pools-are-now-in-preview/

For those that'd like an overview on Instance pools OneNote Instance pools and understanding it for on-call (Web view) which covers:

- Customer experience (PowerShell / Portal)
- Monitoring and debugging
- XTS (instancePools.xts)
- SOPs and TSGs
- Useful queries and tools

For those like that would like a video covering all of the above:

https://msit.microsoftstream.com/video/7df4adc1-8030-42f7-8adb-6c528836ef5e

For a detailed look at instance pool architecture:

https://docs.microsoft.com/en-us/azure/sql-database/sql-database-instance-pools

And for a PowerShell how to:

https://docs.microsoft.com/en-us/azure/sql-database/sql-database-instance-pools-how-to

Instance Pools mainly changes the way how *customers would plan resources and provision instances*, while instance functionality will remain unchanged

Allows customers to plan and purchase capacity at level of instance group (or instance pool), rather than at the individual instance level. This offering will also allow customers to create instances of the smallest possible sizes and migrate their on-premise workloads without database consolidation effort.

You can consider a pool as resource container that customers pre-allocates before instance provisioning in order to unlock following key features:

- Ability to provision small instances (2 vCores for example)
- Ability to pack more databases per single vCore (higher density of DB packaging than in case of individual instances)
- Short and predictable provisioning time (in range of minutes)

The main goal that pools are targeting are to **enable massive instance migration of the smallest size** and to **eliminate need for database consolidation**

The entry bar (in terms of the minimal size and price) that Managed Instance introduced is rather high for many customers: 8 vCores in Public Preview and 4 vCores in GA.

During previews we've seen that many enterprise customers have long tail of Tier 2/ Tier 3 applications running on separated SQL instances with a low compute allocation (1 -2 CPUs, or with even higher virtualization).

For these customers database consolidation (moving DBs from multiple instances on-prem to a single MI in Azure) today is the only option. Some customers are reluctant to do database consolidation due to the following reasons:

- It requires careful capacity planning (which DBs are good candidates to be placed on the same instance). There's no good tool support for this scenario.
- It involves security considerations (which DBs are safe to be hosted on the same instance)
- It requires extra work of instance-level data consolidation (logins, credentials, jobs, etc.) with possible reconciliation and app changes (in case of overlapping instance metadata)
- It requires additional investment in resource governance to ensure that individual database perform expectedly

Majority of the customers prefer simple 1:1 migration (on-prem SQL instance -> SQL MI of the same size) because it eliminates work above.

Below are the main assumptions about optimal customer scenario:

- Customer is ready to perform instance migration to SQL Azure at scale (number of instances >> 1), for non-critical and apps
- Instances are rather smaller in terms of the compute size (< 4 cores)
- There's no concern with placing all instances in the same subnet
- Customer is willing to allocate upfront total amount of resources ("a bundle of resources") required to host a pool of instances.
- Resources allocation and deallocation is done in increments that are bigger than minimal individual instance size.

For example, minimal unit of allocation is 8 vCores, while minimal instance size is 1 vCores.

- Customer is willing to pay for a resource bundle regardless of its utilization (while expected utilization is always high)
- Applications utilizing instances conform to static resource usage patterns.

This model in general will give customer more transparency and control over the resources that are used to host managed instances. Also, it will improve provisioning experience in terms of its predictability: cluster creation or expansion is expectedly longer, while later instance creation very fast (few minutes each).

Basically with Instance Pools the customer can BUY UPFRONT a capacity and distribute to many Azure SQL DB Managed Instances.

Instance Pool is determined by its name, hardware generation and the targeted service tier: General Purpose, Business Critical or Hyperscale.

Instance Poo is bound to customer subnet as target of deployment and boundary of networking isolation and security.

Instance Pool can be grown up to the level of biggest virtual machine available for MI provisioning.

If there are free resources, IP can be scaled down to the level of smallest available VM for MI with favorable GM on selected HW Gen.

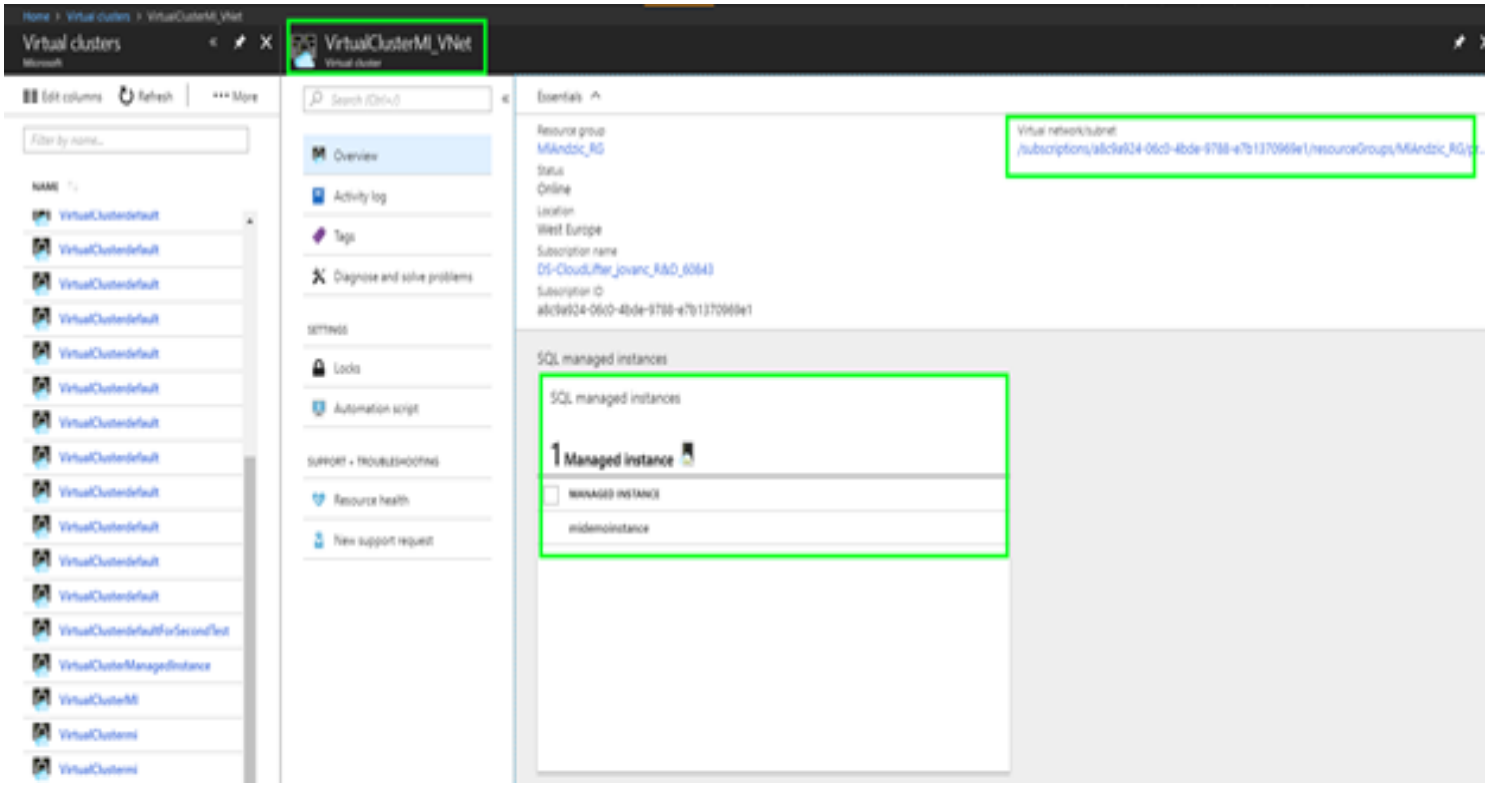
In a subnet customer will be able to place multiple IRPs of same/different size and service tier, although MVP is to enable creation of one IP per service tier first.

Instances created in different IP within same subnet can connect to each other without any additional configuration.

From customer perspective, IP can be seen also as is a “hint” to the provisioning service which underlying VM family to use when deploy PaaS (paradigm from AWS RDS provisioning where VM selection is first deployment step).

Provisioning guarantees instance placement affinity to a specific IP. Instances in deployed in different IRPs have independent HA configuration

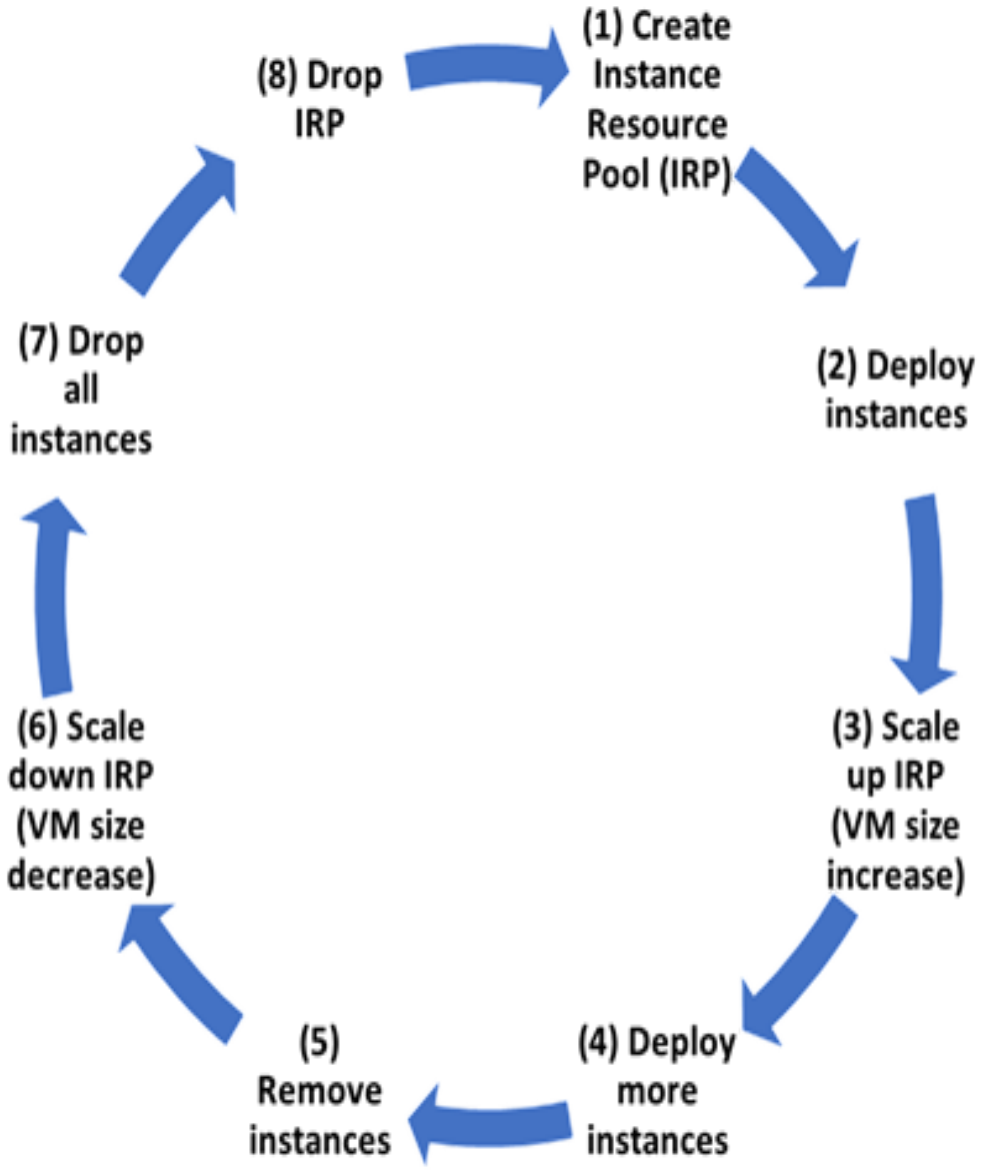
Virtual cluster (VC): it’s an underlying object created by SQL MI provisioning workflow to allow managed instance “injection” in customer VNet / subnet. It represents resources of all managed instances that are placed in the same subnet. Costco expands resources allocation model, with ability for the customers to create IP first. Hence we want to expand information returned about VCs through REST API, to include created IRPs and underlying instances.



The difference in the VC lifecycle between “regular” and Costco model is show below:

Regular deployment model	Costco model	
First instance deployment → virtual cluster build out → **first instance deployment finished (start of billing)** → subsequent instance creation	IP deployment → **virtual cluster build out (start of billing)** → first instance creation → subsequent instance creation	

Here is high-level lifecycle diagram for IP and Instances:



How good have you found this content?