1&1 IONOS Enterprise Cloud

Fully automated high availability setup for gateway and database cluster

This script is meant to create a full featured setup with gateways and database servers in a hot-standby mode in a virtual data center of the Enterprise Cloud of 1&1 IONOS.

In order to prove the usability of such a setup and to make it more generally comparable you may load the productive database MusicBrainz.

Overview of the basic steps

(For the impatient: read this section and you can start.)

1. At first a virtual data center (VDC) in the Enterprise Cloud will be created and three customer IP addresses will be reserved. This is necessary to make sure that an address does not get lost after a reboot of the virtual machine (VM).
2. The next step is to create a bundle of two VMs which will secure the internet connectivity with a virtual IP which could be taken over by the standby VM in case of a failure of the master VM.
3. While creating the gateway VMs, the failover group is created. This is important to guarantee that the whole internet traffic is sent to the active VM where the external failover IP is assigned to.
4. Then a VM as management instance is created to be an example for the reachability of the internal network. The first aim of the project was to demonstrate just that. This VM will be the management host which uses SaltStack to fulfill the next steps of installation. It is also possible to limit the setup at this point, hereby you would have a VDC to test the failover group and virtual IP feature externally and within the private network.
   (For a more detailed description of the failover configuration see the corresponding article series ‘The Failover Cloud Solution Part 1 to 3’   by 1&1 IONOS Professional Services Consultant Thomas Vogel.)

If this is all you require, you just start the script with a location option. Only if explicitly wanted and and further options are specified the next two parts come into place:

5. Another big step is the setup of two PostgreSQL instances in a master-standby configuration. The cluster will be reachable via a virtual private IP handled by PGPool which also handles the requests to the database. The internal virtual IP is reachable via port forwarding served by the gateway VMs. For testing and comparison different underlying file systems could be chosen.
6. As last task a working MusicBrainz database is loaded into the setup.

In order to start download the package, unpack it, change into the created subdirectory and execute HA-GateAndDatabase.sh. Starting the script without options shows the usage of the script:

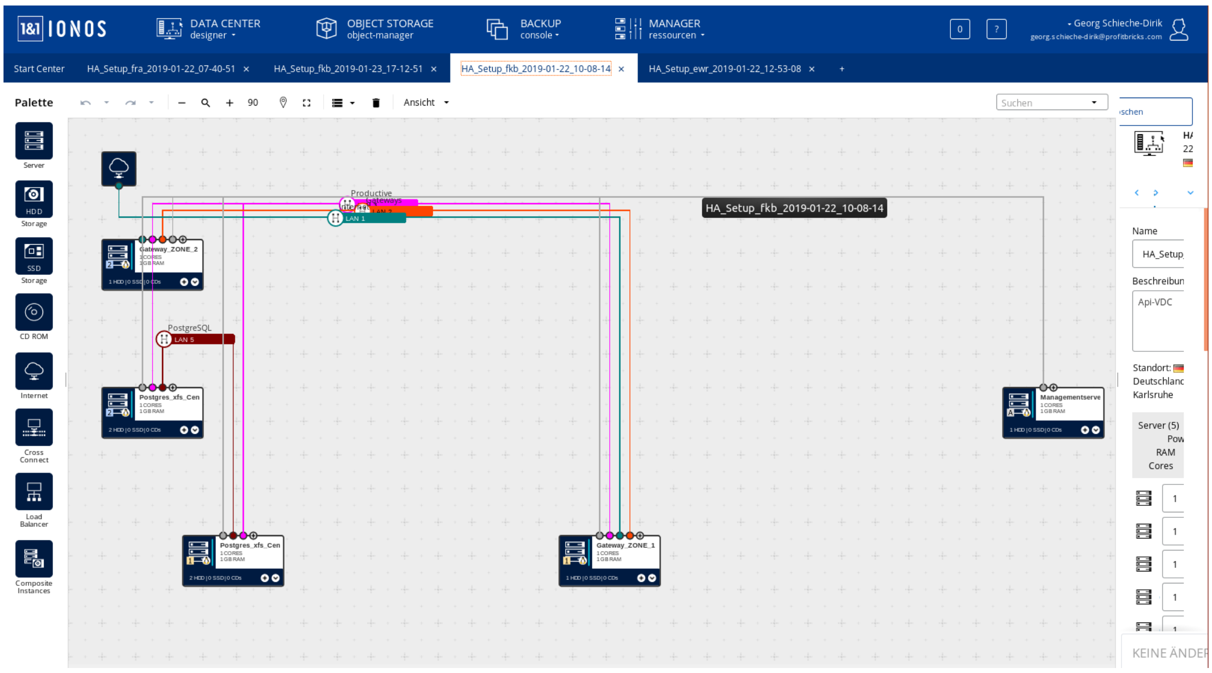
```
<snip>
Usage: ./HA-GateAndDatabase.sh -l location [-d] [-D] [-M] [-s] [-i] [-f
filesystem] [-S number]

Location can be fra (Frankfurt), fkb (Karlsruhe), ewr (Newark) and las (Las
Vegas)

-d activate debug mode
-D install the database cluster
-M install the MusicBrainz database
   The filesystem for the DB would be expanded to at least 300 GB
-s use SSD for database volumes
-i use Intel for database VM cores
-f filesystem
   The filesystem could be ext4 (default), xfs, btrfs, zfs
-S size of database volume in GB

Just "./HA-GateAndDatabase.sh -l location" installs only the gateway HA setup
with the management host behind it.

The user credentials need to be placed in a file named
/home/gschiechedirik/ionos/.config in the form 'user@domain.tdl:password'
(without apostrophes).

With the "-d" option, your Data Center Designer password will be shown on the
console!

</snip>
```

With the command
`’./HA-GateAndDatabase.sh -l fra‘`
a simple HA-setup located in our data center in Frankfrut is created. For checking a failover group and learning its functionality this is all you need.

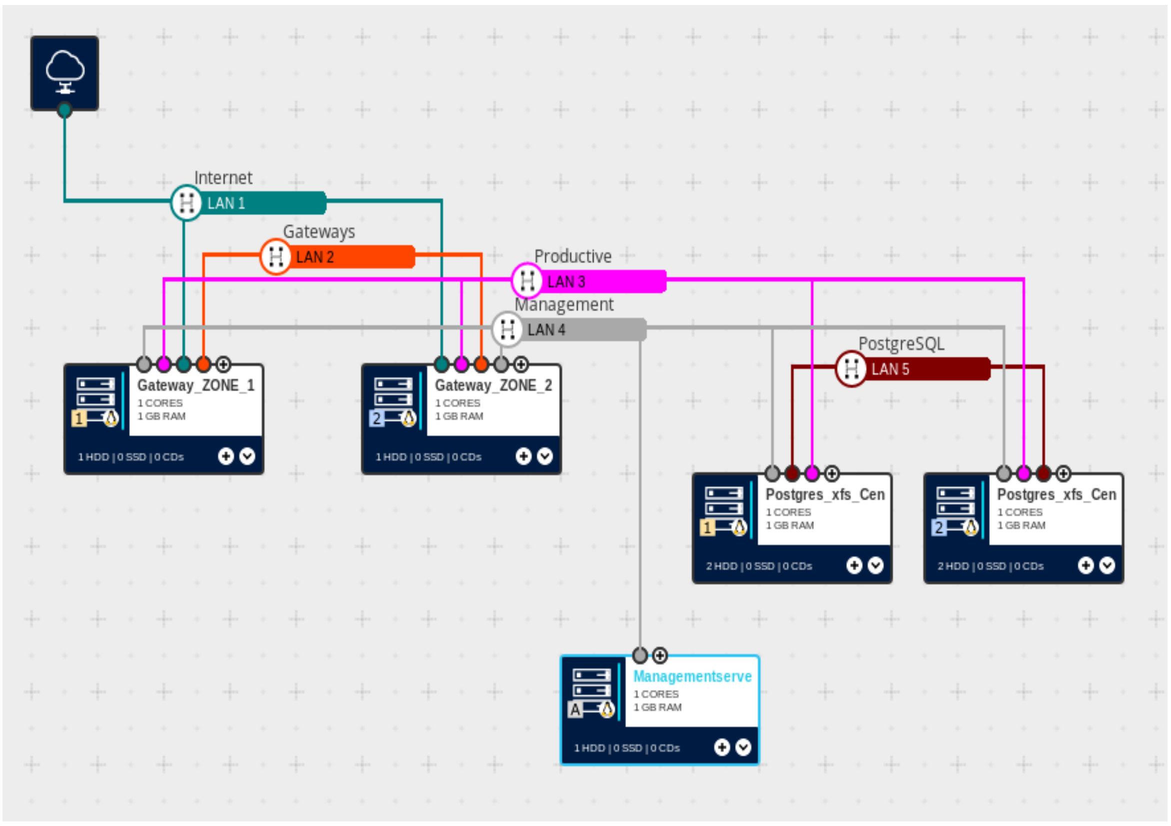
If you like to test the database cluster as well you may use
`’./HA-GateAndDatabase.sh -l fra -D -M -f xfs‘`

IP-failover setup – Data Center Designer View

When the script is finished the virtual data center first looks like this.

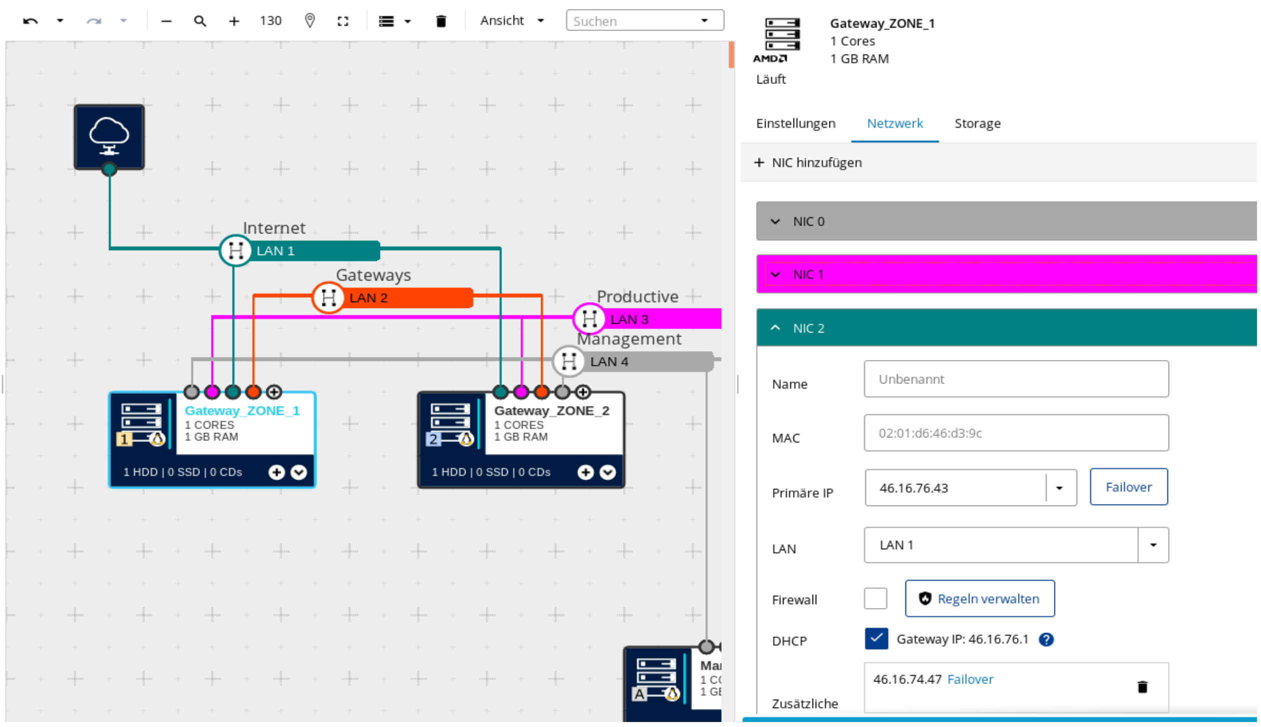


After moving the VMs the structure is visible.



The failover part is configured for the VMs Gateway_ZONE_1 and Gateway_ZONE_2
On the right side the external NIC configuration of Gateway_ZONE_1 is shown.

The

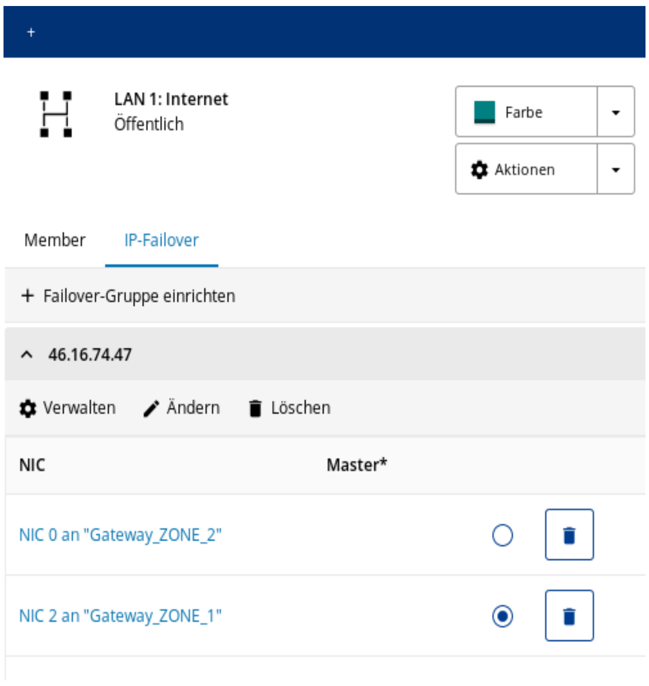


Gateway_ZONE_1
1 Cores
1 GB RAM
Läuft

Name	Unbenannt
MAC	02:01:d6:46:d3:9c
Primäre IP	46.16.76.43
LAN	LAN 1
Firewall	☐
DHCP	✓ Gateway IP: 46.16.76.1
Zusätzliche	46.16.74.47 Failover

primary external IP is 46.16.76.43. **The virtual IP is the additional one 46.16.74.47. This IP is used for the failover.**

Here you see the config windows for the failover group.



LAN 1: Internet
Öffentlich

46.16.74.47

NIC	Master*
NIC 0 an "Gateway_ZONE_2"	○
NIC 2 an "Gateway_ZONE_1"	●

Reasons for this project:

1. First off all I wanted to get to know the DCD API. Every call is made with curl. That way I am always able to check the functionality of our programming interface. There are several other tools or SDKs which do this job but if there is something not working as expected it is never certain if the API has an issue or the used software. With shell commands the user has the option to address the API without any layer in between.
2. If the script is started with the debug option ’-d‘, then every call is shown explicitly. This makes it easy to find the right syntax for every call and to check if all used functions are working as expected. It is easy to use the steps as template for other projects.
3. The Virtual Router Redundancy Protocol (VRRP) is widely used for a HA setup with a virtual IP, especially in our infrastructure. So it makes sense to use it for this setup. After invoking the script the whole setup is done using shell commands. The necessary firewall settings are also made.
4. In the management VM SaltStack is installed. This tool is used to manage great networks of computers and, therefore, one candidate for the configuration of the database VMs and the installation of all necessary applications.
5. The PostgreSQL database is configured as master-standby cluster. The standby is kept in sync via streaming replication. The standby can handle read queries and can take over the master role in case that the master stops working. If the former master is restarted and gets online again, the Salt configuration let become it the new standby (while the formers standby keeps the master role). As file system you can choose between ext4, btrfs, xfs and zfs.
6. The distribution of read and write requests for the database and the handling of the virtual IP for the productive interface is managed by PGPool. This is also configured by SaltStack.
7. As 'complete productive database' example the MusicBrainz (see https://musicbrainz.org/doc/About/Data_License) database is installed. This installation is also done by SaltStack. The part of importing MusicBrainz takes something between four and five hours. Only after that (!) the whole failover setup should be tested. The failback of one instance also takes at least one hour to bring the database in sync again.
8. A simple test of the setup is to kill the master instance and to look if the failover works as expected. Other possibilities are given as command execution output.

Having this done we have a highly available database setup you can use to experience and test new software, its performance on our platform and new configurations.

Assembling the aforesaid in one single script and fulfilling all those demands described with one command execution will result in considerable spare time for thorough preparation.