

PostgreSQL deployment in Azure

OpenEarth is dedicated to automate deployment of the OpenEarth stack in Azure, please assist us at <https://github.com/openearth-stack>.

The first component of the OpenEarth stack we focus on is PostgreSQL plus the PostGIS extension.

Managed service

Azure offers Postgres as a managed service for app developers, a DataBase as a service (DBaaS)

<https://azure.microsoft.com/en-us/services/postgresql/>

Of course you can also use this fully functional Linux Data Science Virtual Machine which includes Postgres.

<https://azure.microsoft.com/en-us/marketplace/partners/microsoft-ads/linux-data-science-vm/>

or a Postgres cluster

<https://azure.microsoft.com/en-us/marketplace/partners/bitnami/postgres-clusterdefault/>

However, because the OpenEarth partners have on-premise facilities (data centres for models, a fleet, etc.) the architecture for azure deployment has to be fully hybrid.

Due to this requirement we cannot always use the DBaaS. Therefore we also offer an option using other technologies.

Hybrid

For a hybrid offering there are several options: build (automatically) from scratch, or use a container.

There are detailed instructions on how to deploy a PG database in Azure using only a Linux server:

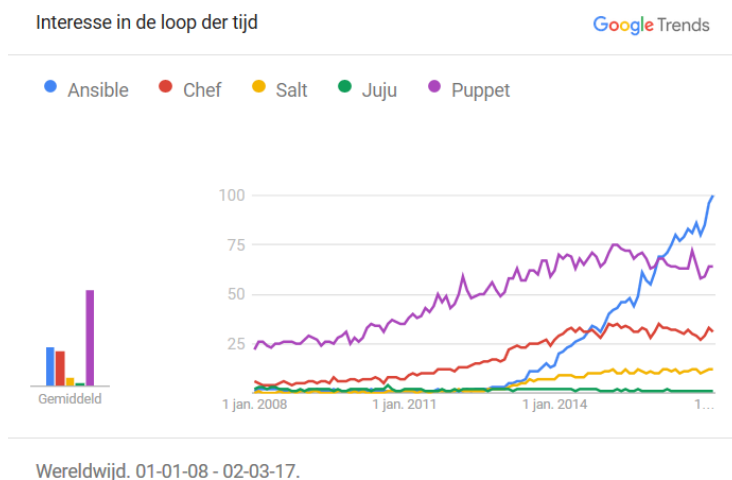
<https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-postgresql-install/>

Following all these instructions might take a few hours, due to human processing time as well as due to virtual machine spin-up time.

The next step is to automate the human processing time into a script, with our choice [Ansible and Docker](#).

The most popular scripting tool for deployment at this moment is Ansible. As of oct 2016 Ansible has taken over Puppet as the most popular tool

https://www.google.com/trends/explore?date=2008-01-01%202017-03-01&q=%2Fm%2F0k0vzjb,%2Fm%2F05zxlz3,%2Fm%2F0hn8c6s,%2Fm%2F0h68cj_,%2Fm%2F03d3cj&hl=en-US



In addition to being the most popular, Ansible has some advantages over other the previous generation of tools such as Puppet, notably the ease to start using it. You can find many blogs that compare Ansible to other system configuration tools, like

<http://www.slideshare.net/jbminn/docker-ansiblemakechefpuppetunnecessaryminnihan-34984161>

We hope this convinces you to use Ansible. Here is a high-level overview of Ansible:

<https://sysadmindcasts.com/episodes/43-19-minutes-with-ansible-part-1-4>

After you automated the human processing time, the virtual machine spin-up time can be reduced by means of deploying containers. There are two ways to do this. First, there are classic tools to deploy full virtual machines, such as virtualbox or vmware. Vagrant is wrapper around these tools to automate them. However, very popular nowadays is Docker to launch mini-containers fast. The differences between Docker and Vagrant are explained here

<http://devo.ps/blog/vagrant-docker-and-ansible-wtf/>

We prefer to combine Docker with Ansible: use Ansible scripting to make a Docker container.

<https://www.ansible.com/docker>

<https://oliverveits.wordpress.com/2015/11/09/it-automation-a-hello-world-example-using-ansible-on-docker/>

This can be automated using

<https://docs.docker.com/docker-hub/builds/>

blocked URL

The Ansible scripts can be tested with [Travis](#).

Vagrant might still come in very handy to launch the Ansible Management Node.

<https://sysadmindcasts.com/episodes/45-learning-ansible-with-vagrant-part-2-4>

Some are sceptical of making a container/boxes of a database, but there are strong argument in favour of it

<http://stackoverflow.com/questions/25047986/does-it-make-sense-to-dockerize-containerize-databases>

However, be aware that Docker containers are transient, so make sure you configure it to store your data *outside* the Docker container.

Note that fully functional Postgres Docker containers

https://hub.docker.com/_/postgres/

<https://azure.microsoft.com/en-us/marketplace/partners/docker/postgres-arm/> with <https://azure.microsoft.com/en-us/documentation/templates/postgresql-standalone-server-ubuntu/>

as well as Vagrant boxes (<http://www.pgdevbox.com/#>) are already available for download. However, these do not contain the geospatial PostGIS extension which is essential for OpenEarth applications. Hence it makes sense to script a Docker container ourselves.

A huge advantage of Ansible is also that it comes with tools to also automate database tasks, notably to set proper authorization. And perhaps to insert a data model.

<http://blog.2ndquadrant.com/ansible-loves-postgresql/>

http://docs.ansible.com/ansible/postgresql_db_module.html

Please note that you need to deploy tools like Barman for back-up and recovery of your database.

<http://www.pgbarman.org/>

For DataBase administration (DBA) purposes, we recommend to deploy the python webtool PGAdmin 4 on the same Linux server.

<https://www.pgadmin.org/download/pip4.php>

For

The next step in automation is to use docker-swarm to spin up a database with associated pgadmin and geoserver.

<https://hub.docker.com/r/openearth/>