Computational
Health
Informatics

Leibniz
Universität
Hannover

Refining the installation method of GNU Health

Gerald Wiese

GNU Health CON 2022

About me

- Gerald Wiese, Leibniz University Hannover, Germany
- Master thesis: Automatic deployment of an open source hospital information system for use in research and teaching (2020/21)
- Joined GNU Health community meanwhile
- Employed at Computational Health Informatics:
  - Lab: Linux System Administration
  - Exercise: Fundamentals of Medical Informatics
  - Supervising Theses
  - Research

1) Automation using Ansible

2) How to improve the installation strategy and documentation of GNU Health?

Ansible

- Term from Science Fiction (first use of Ursula K. Le Guin): Fictional devices communicating faster than speed of light [Deimeke21]

- Automation tool combining configuration management, server deployment & ad-hoc task execution [Geerling22]

- „Be the easiest IT automation system to use, ever.“
- Founded 2012 by Michael DeHaan, sponsored by Red Hat
- 5000+ contributing users, GPLv3 [Ansible]

Ansible

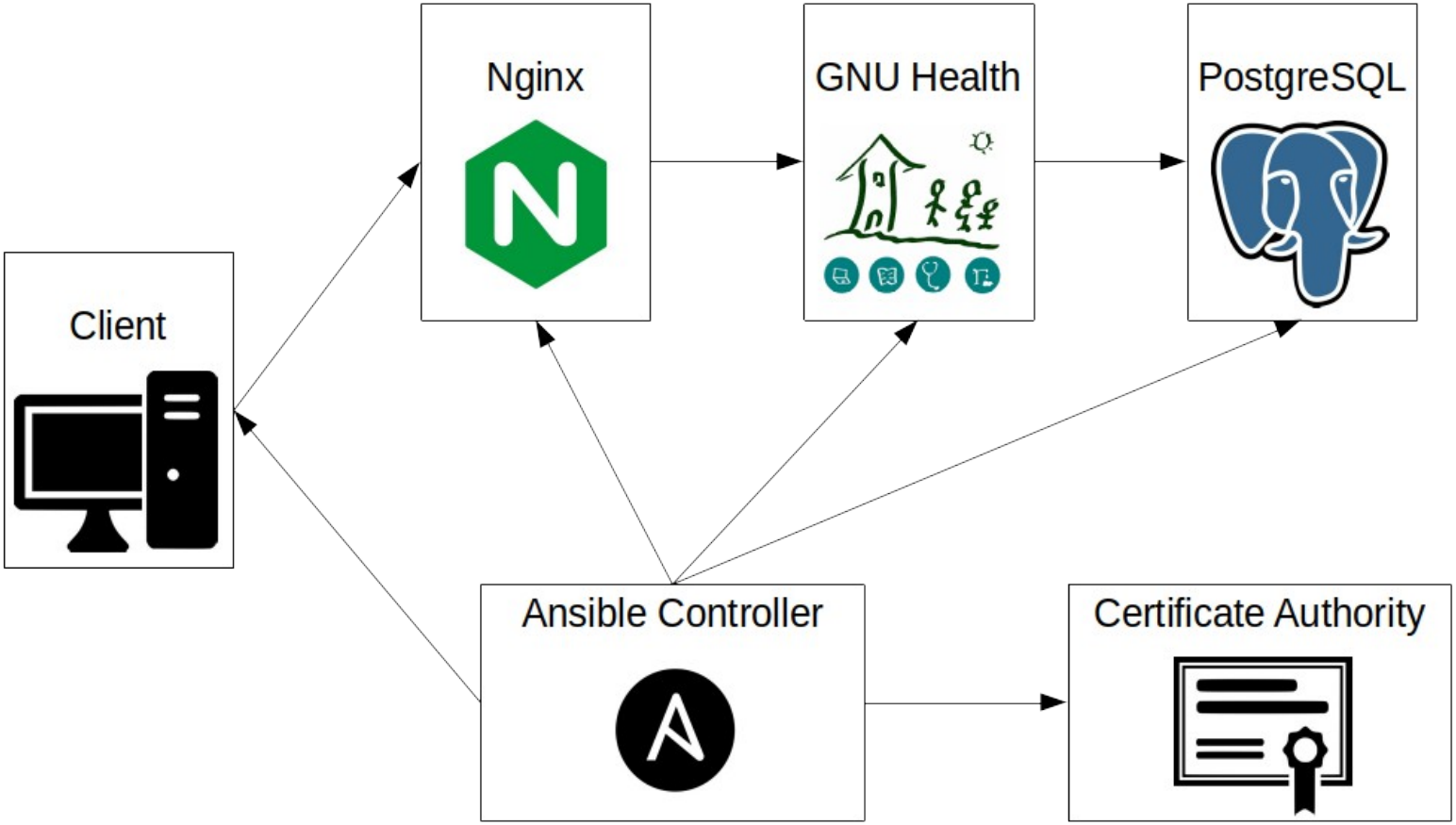
- Aim:
  - Clear – Simple API & YAML syntax
  - Fast – Learning & installation
  - Complete – Fulfill many needs all in one
  - Efficient – Lightweight & extensible
  - Secure – SSH, avoid additional daemons/ports
- Idempotence
- Reproducability
- Scalability [Geerling22]

Ansible

- Written in Python
- Managing target systems using SSH & Python
- >3.000 modules
- Easy to start & suited for complex scenarios
- Components:
  - Inventories – Machines & configuration
  - Playbook – Sequences of tasks for execution
  - Modules – Every task calls one module
  - Roles – Modularize complex projects

  [Deimeke21]

GNU Health – Deployment automation



Ansible Repository

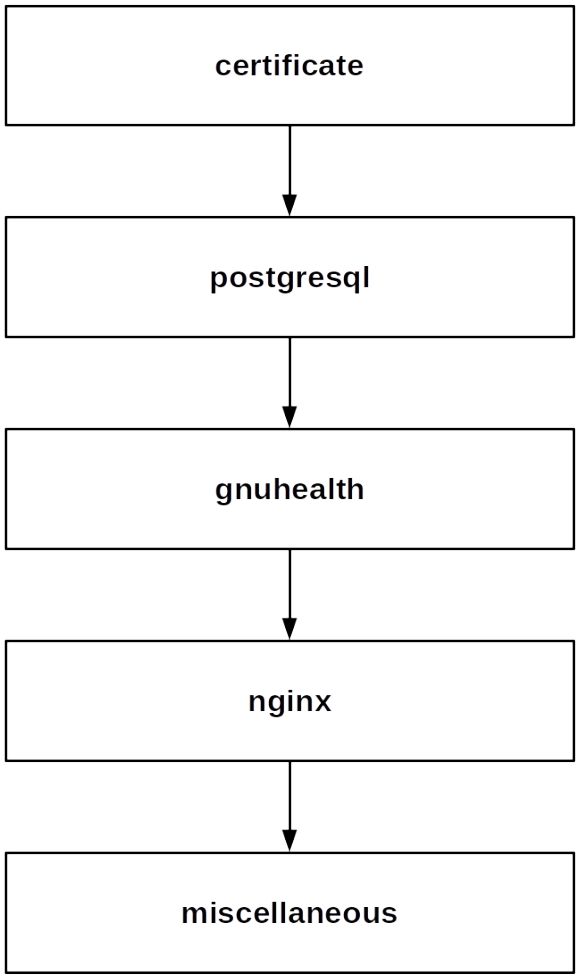
- For GNU Health & Orthanc:
  - Ubuntu, Debian & openSUSE Leap
  - Servers all-in-one or distributed
  - Different connection options: UNIX Socket, TCP, TLS
  - One-liner installation possible

- GitLab Pages documentation
- Vagrant: Create Virtual Machines using VirtualBox & Ansible
- Integration tests using Molecule & GitLab CI

https://gitlab.com/geraldwiese/gnuhealth-automatic-deployment

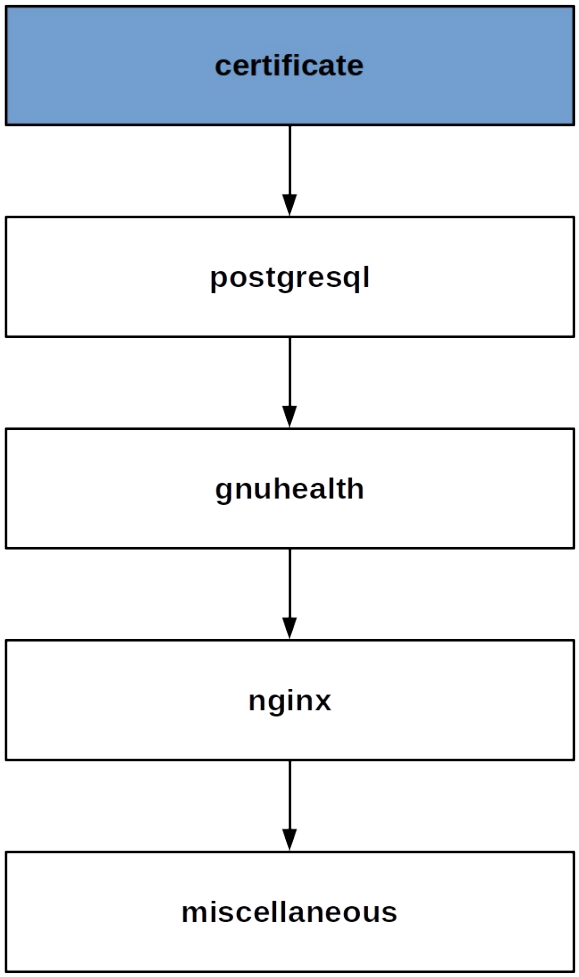
GNU Health - Deployment

- Ansible playblook calls multiple roles
- Configuration in inventories:
  - dev
  - test
  - prod
  - (vagrant)
- Encrypt credentials using Ansible-Vault



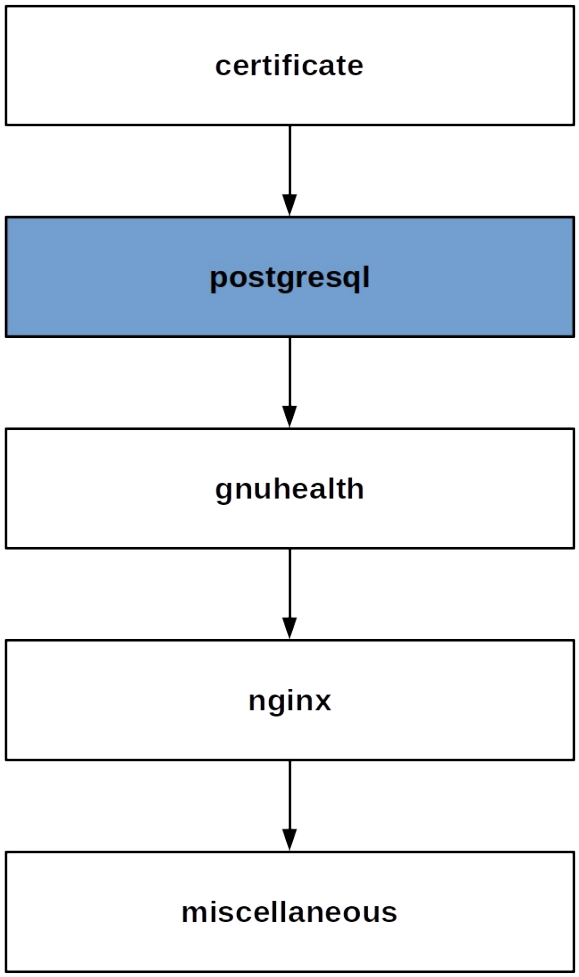
Digital certificates

- Needed for realizing TLS/HTTPS
- Three different options:
  - Set path of existing one
  - Generate using Let‘s Encrypt (only tested on Ubuntu yet)
  - Create Certificate Authority for signing server certificates



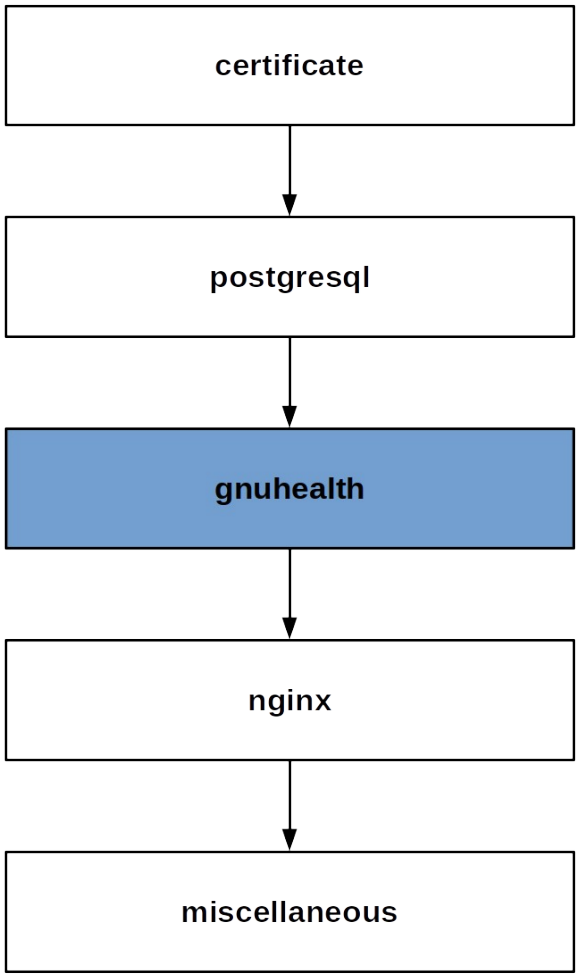
PostgreSQL

- Create user & database
- Allow remote connections if needed but limit access in pg_hba.conf: *hostssl health,template1 gnuhealth \ 10.0.0.10/32 scram-sha-256*
- Ensure correct permissions & paths for certificate & key



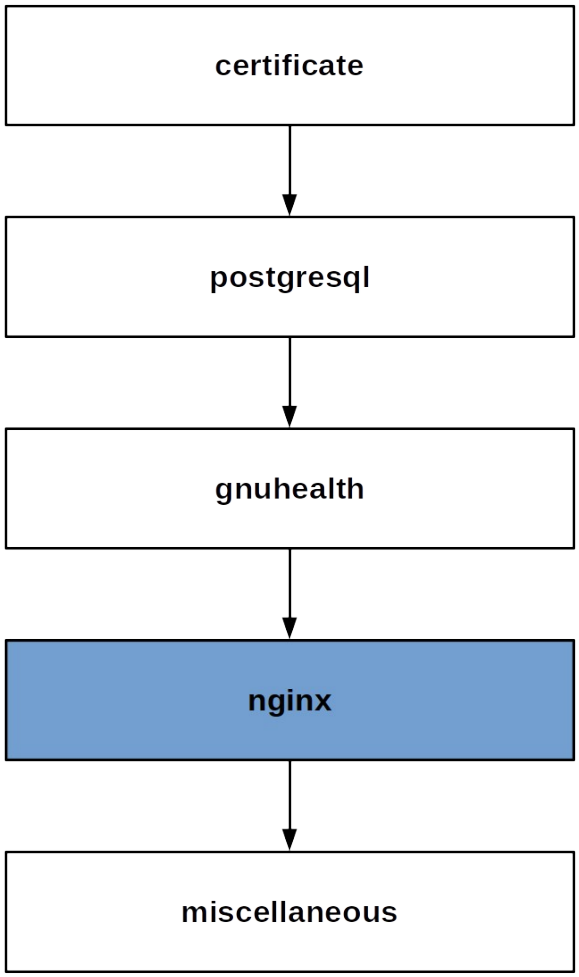
GNU Health

- Create user *gnuhealth* without login
- Install GNU Health & uWSGI using pip and virtual environment
- Set PostgreSQL connection string
- Manage certificates if needed
- Create directories
- Distribute config files
- Create & enable systemd service



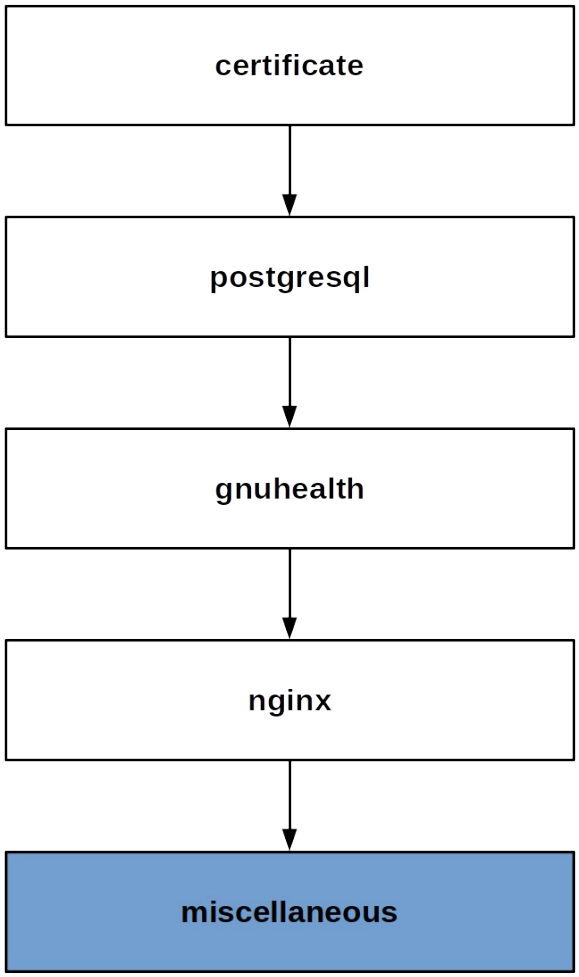
Nginx

- Install Nginx for HTTPS access
- Handle certificates
- Connection to GNU Health application:
  - UNIX socket
  - TCP socket & uWSGI
  - HTTPS
- Disable TLS<1.2
- Set recommended SSL ciphers
- Set header *X-Real-IP*



Miscellaneous

- Fail2Ban: Temporary ban of IPs
- Set timezone
- Unattended Upgrades
- sSMTP configuration for existing mail
- E-Mail notifications on service inactivity



GNU Health Client

- Install using pip or zypper
- Create desktop entry if installed by pip
- Add Crypto plugin
- Profile in /etc/skel → Copied when creating OS users
- Trust CA → encrypted communication
- Possible to install on any number of systems

Examples – Test instance

- Preferrably use a Virtual Machine (VM)
- After installing requirements & cloning git repository:

$ ansible-playbook playbooks/gnuhealth.yml -i inventories/dev -c local -e my_user=`whoami` -K

$ ansible-playbook playbooks/desktop.yml -i inventories/dev -c local -e my_user=`whoami` -e ghcl_trust_certs=true -K

- This results in having a local client & server including Nginx & PostgreSQL

Examples – Vagrant & VirtualBox

- Requirements include Vagrant & VirtualBox
- After cloning, navigate in vagrant directory and generate VMs:
  $ cd gnuhealth-automatic-deployment/vagrant/gnuhealth_split
  $ vagrant up

- Client, Nginx, GNU Health & PostgreSQL each on a single VM
- Click „Show“ in VirtualBox or „vagrant ssh {gui,web,app,db}“
- Operating system choice in Vagrantfile


https://geraldwiese.gitlab.io/gnuhealth-automatic-deployment/examples.html

Tests using Molecule & GitLab CI

- Ensure config templates are up to date
- Linting
- Run Ansible roles without errors
- Idempotency
- Test connectivity / integration using Proteus
- Different operating systems
- Split vs. all-in-one
- *.gitlab-ci.yml*:
  - Possible to trigger after every commit
  - Install requirements and run tests
  - Deploy GitLab Pages documentation

Modular Expandability

- GNU Health is not the only software you can put between Nginx & PostgreSQL

  → Orthanc already realized

- Can be used as a base for deploying any Python app having a WSGI function

- Changing config templates, adding operating systems or functionality should be easy

2) How to improve the installation strategy and documentation of GNU Health?

Installation based on PyPI

- Package *gnuhealth-all-modules* contains all modules & dependencies
- Works on any operating system (OS)
- Patches independent from OS maintainers
- Upgrades are simple
- Avoid installing Python packages system wide:
  - Potential version conflicts
  - Python 3.6 End Of Life on openSUSE Leap

→ Virtual Python Environment

https://pypi.org/project/gnuhealth-all-modules/

uWSGI

- GNU Health/Trytond uses builtin *werkzeug* by default

  Warning:

  Do not use the development server when deploying to production. It is intended for use only during local development. It is not designed to be particularly efficient, stable, or secure. [werkzeug]

- uWSGI offers many configuration options & functionalities for productive use [Debian] [Bloomberg]

Separate systems

- Document the separation into web server, application & database:
  - How to realize remote access?
  - How to minimize accessibility?
- Document how to use TLS for all those subsystems
- Three connection methods between Nginx ↔ uWSGI/GNU Health ↔ PostgreSQL:
  - Local, UNIX socket
  - Remote TCP, TLS/HTTPS
  - Remote TCP, unencrypted
- Basis for redundancy

Benefit from Ansible

- Reproducability:
  - While actually performing actions you are already documenting them
  - In order to set up your environment again, change the hostname or IP address & that‘s it

    → Improve automation & cooperation
- Keep complex systems as overseeable as possible
- Numerous modules & roles to (re)use

Future work

- Redundancy
- Optimize modularity
- Backup/restore based on separation/redundancy
- PyPI & updated documentation for 4.2?
- Collaboration for deployments in the field?
- More ideas?

- Contact:
  Gerald Wiese
  wiese@gnuhealth.org

References

- [Deimeke21] Dirk Deimeke et al., Linux-Server Das umfassende Handbuch, 2021
- [Geerling22] Jeff Geerling, Ansible for DevOps, 2022
- [Ansible] https://github.com/ansible/ansible
- [werkzeug] https://werkzeug.palletsprojects.com/en/2.2.x/serving/
- [Debian] https://packages.debian.org/unstable/tryton-server-uwsgi
- [Bloomberg] https://www.bloomberg.com/company/stories/configuring-uwsgi-production-deployment/