

JUMPING JIVE



730884 – JUMPING JIVE – H2020-INFRADEV-
2016-2017/H2020-INFRADEV-2016-1

Evaluation of software packages (for system monitoring)
Written by A. Neidhardt

Deliverable 8.4

Submission date: 07.08.2017

Content

1	Overview	3
2	Evaluation criteria.....	5
3	Used software packages for system monitoring and control of VLBI	7
3.1	VLBI network monitoring	8
3.1.1	"MoniCA" or "openMoniCA" (Australia Telescope National Facility and AuScope geodetic VLBI telescopes)	8
3.1.2	ZABBIX & SysMon (Wettzell Observatory)	11
3.1.3	Telegraf - InfluxDB - Grafana (TIG) (NASA FS)	14
3.1.4	Radboud Radio Lab VLBI monitor (EVN, mm-VLBI)	18
3.1.5	Monitoring and Control Infrastructure MCI (MIT Haystack Observatory)	20
3.1.6	Industrial monitoring tools (Nagios, Zabbix, etc.)	22
3.1.7	SKA Telescope Manager (SKA)	30
3.1.8	Individual Linux scripts	32
3.1.9	Goddard Mission Services Evolution Center (GMSEC) Architecture	34
3.2	VLBI remote control, operation and automation	36
3.2.1	e-RemoteCtrl (Wettzell, O'Higgins, AuScope, partly NyAlesund and Hartebeesthoek, tested for TIGO in Chile)	37
3.2.2	Dynamic Observation (AuScope tests)	42
3.2.3	Remote control and monitoring of e-VLBI experiments at JIVE ERIC... ..	46
3.2.4	SSH/VPN-tunnels to or VNC/Remote-desktop of the NASA Field System PC	50
4	Final evaluation statement of the software packages.....	52
5	Appendix: Zabbix SysMon	54
6	Appendix: openMoniCA (on Ubuntu 16.04 LTS)	122
7	Appendix: Telegraf - InfluxDB - Grafana (TIG) for the NASA Field System	125

1 Overview

In the past, monitoring of elements of a VLBI array during observations used to be virtually impossible. Equipment failures, human mistakes or other mishaps would often only be noted during or even after correlation by inspecting the data, which might take place months after the actual observations. The advent of e-VLBI, transferring data in real time to the correlator, brought a considerable improvement in the communications between stations and correlator. However, VLBI observations are still mostly done using recorders, during sessions that hardly have any central overview of the network as a whole. This will be unavoidably the case for the proposed global VLBI, involving remote stations all over the world.

During the EC-funded NEXPreS project, a remote control and monitoring system was developed and deployed at a number of geodetic stations. This task will evaluate this product and other existing monitoring systems to find a common ground in order to ensure interoperability. It will adapt existing software for integration into a central infrastructure and set up web-based access techniques. The final product will be a central, web-based monitoring system, usable for both astronomical and geodetic VLBI. This system will be accessible to all involved, correlator and stations, and will serve to continuously monitor and assess the status of the VLBI network, enabling automated warnings in case of failures and providing the information needed to continuously improve the performance of the network. Such a system will also be of great value when helping with the commissioning of new VLBI telescopes, in particular in areas of the world where radio astronomy is not well-established yet. (Grant agreement)

The tasks of the work package can be separated into two main branches for the centralized coordination (see Fig. 1.1:

- Remote monitoring: receiving status, health and quality data from all relevant components to detect error situations and to raise alarms
- Remote access: secure and safe access to the relevant components to repair error states or to control individual processes

While a safe and secure remote access was demonstrated during the FP7 project NEXPreS using e-RemoteCtrl, the software was just established at a few sites, e.g. Wettzell, O'Higgins, Hobart, Yarragadee, Kathrine, and a few test sites like Hartebeesthoek and Ny Alesund. Restrictions in network managements at the different sites and administrative questions made it difficult to support and enable the active fetching of data and commanding of equipment even if a role-based, tunneled, encrypted authentication was implemented.

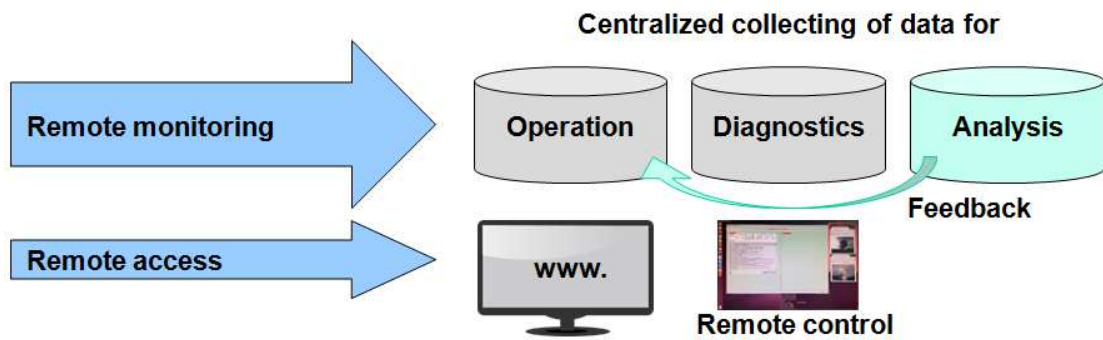


Fig. 1.1: Active remote access and passive monitoring of data for operation, diagnostics and analysis.

The passive monitoring where sites can send their data independently on predefined channels to a centralized data center might have better acceptance than active access. Therefore, the main focus is on monitoring whole networks of VLBI telescopes if sessions should be checked by a centralized service. Open techniques and web-based methods like browsers with common web pages are the key feature in modern and mobile tools.

Such a monitoring can provide data and feedback for operation of an observing session at a complete network of telescopes. It also can support diagnostics after error situations to detect at least the start time of an erroneous situation. It might also give information which equipment fails how and in which way. Finally, collected data sets like system temperatures or clock offsets are important for the analysis centers, which can give feedback to the operator. As faster such analysis data can be used for quality checks, as faster feedback can be given.

Because monitoring of health states is one central part of the work package, the analysis and comparison in the first working period leading to this deliverable concentrating on system monitoring software which should then be installed and used at centralized data centers. The report just touches currently existing software and techniques for remote control to describe the current situation.

2 Evaluation criteria

The evaluation also separates between monitoring environments and control & operation software. The evaluation itself is a qualitative estimation based on an individual representation and converted into quantitative numbers.

The criteria for monitoring software are:

- Industrial importance
- Open source
- Use for VLBI
- User community
- Developer community/ Size of developer team
- Readiness for production and release policy
- Community support
- Documentation
- Platform support (Linux for central server; also others for data provider)
- Simplicity to install
- Simplicity to use for non-specialists
- Simplicity to extend with own parts
- Supported techniques/protocols for the use in VLBI
- Web based access possibilities
- Usability for distributed VLBI networks
- Coherent design and use of programming languages
- State-of-the-art techniques
- Access control
- Maintainability and possibility of migration
- Future relevance
- Individual notes

Each item can be rated with a number from 0 to 5, where 5 is the best rating and 0 the worst.

The criteria for control and operation software are:

- Industrial importance
- Open source
- Use for VLBI
- User community
- Developer community/ Size of developer team
- Readiness for production and release policy
- Community support
- Documentation
- Platform support (Linux for central server; also others for data provider)

- Simplicity to install
- Simplicity to use for non-specialists
- Simplicity to extend with own parts
- Supported techniques/protocols for the use in VLBI
- Web based access possibilities
- Usability for distributed VLBI networks
- Coherent design and use of programming languages
- State-of-the-art techniques
- Access control and network security
- User control management
- Local supervisory
- Safety implementations
- Maintainability and possibility of migration
- Official certification or product certification
- Future relevance
- Individual notes

Control and operation aspects have a reduced priority due to found issues with network security and safety at the sites. This is taken into account that monitoring tools are evaluated with a maximum priority of 5 while control and operation software has a maximum priority of 3. Therefore, each control and operation item can be rated with a number from 0 to 3, where 3 is the best rating and 0 the worst.

Each software package is briefly described and then evaluated using the criteria above. If some parts of the control and operation programs support monitoring tasks, they are additionally rated. The additional numbers can be found in parentheses.

The end of the report compiles all ratings to an ordered table which is the suggestion for integration into a central infrastructure. Highest rated software is best suited for a first implementation. But ratings are just an individual snapshot and might be rated differently by other experts. They should be adapted if other aspects from the operation center or sites should be taken into account during the implementation.

3 Used software packages for system monitoring and control of VLBI

There are different software packages already used at the different telescope sites. The following sections are a collection and short description of the already used software. It does not claim completeness but is carefully collected with the experience of the past years working for the VLBI communities.

3.1 VLBI network monitoring

3.1.1 "MoniCA" or "openMoniCA" (Australia Telescope National Facility and AuScope geodetic VLBI telescopes)

MoniCA is a Java-based graphical application for viewing real-time and archival monitor data. It was developed for the Australia Telescope National Facility by CSIRO (David Brodrick). It is an open-source project and can be downloaded from Google-code.

Some features are:

(cited from <http://code.google.com/p/open-monica/>)

- *Generic support for industry standard protocols such as Modbus, SNMP and EPICS.*
- *Combine data in arbitrary ways to create aggregate monitor points.*
- *Alarm management system, with notifications, alarm acknowledgement, etc.*
- *Control points allow operation of remote devices to be implemented.*
- *Optionally archive data to disk, using a compressed format or a MySQL database.*
- *The GUI client can plot any combination of monitor points and display real-time values in tables.*
- *Client fully tested on Linux, Windows, Mac and Solaris operating systems.*
- *Javascript-based web browser client included in the distribution.*
- *Server has a ZeroC Ice interface and can be accessed natively from a number of different programming languages.*
- *Server has a simple ASCII socket as an alternate way to provide data to external programs.*
- *Used at research facilities with tens of thousands of real-time monitor points.*
- *Easily export data for analysis in spreadsheets or take graphical screen-shots.*

openMoniCA is already in use at telescopes, which will not switch to another system but might offer data using already existing access points, where access rights are required. The documentation is based on some files in the installation directory and limited Wiki pages. The community is limited and the developer community consists of one person. The existing setups demonstrate the support of a centralized

data collection in a distributed network at least with two hierarchy levels: the remote, distributed sensors and the central data collection.

Links:

Source code:

<http://code.google.com/p/open-monica/>

<https://github.com/davidbrodrick/open-monica>

Description:

<https://www.narrabri.atnf.csiro.au/monitor/monica/>

Wiki (from the AuScope network):

<http://auscope.phys.utas.edu.au/opswiki/doku.php?id=software:openmonica>

Evaluation:

Qualifier	Priority (0-5)
Industrial importance	0
Open source	5
Use for VLBI	5
User community	2
Developer community/ Size of developer team	2
Readiness for production and release policy	5
Community support	3
Documentation	2
Platform support (Linux for central server; also others for data provider)	4
Simplicity to install	5
Simplicity to use for non-specialists	5
Simplicity to extend with own parts	4
Supported techniques/protocols for the use in VLBI	4
Web based access possibilities	2
Usability for distributed VLBI networks	5
Coherent design and use of programming languages	5
State-of-the-art techniques	4
Access control	5
Maintainability and possibility of migration	4
Future relevance	3
Individual notes:	2
<ul style="list-style-type: none"> - openMoniCA will continue to play a role in the ATNF network; - It might be replaced by Telegraf-InfluxDB-Grafana (TIG) within the geodetic AuScope network in the coming future - It does not really play a big role outside of these networks 	
Priority	3,62

3.1.2 ZABBIX & SysMon (Wettzell Observatory)

Developers at the Wettzell observatory started to develop SysMon because a central data archive for operation and analysis data was required. The development started in the year 2009. M. Schönberger wrote a centralized monitoring server fulfilling the requirements of the systems at the Wettzell observatory. It used the idl2rp.pl generator (idl2rpc.pl is a Perl script written at Wettzell for a local communication middleware) to automatically create SunRPC communication interfaces for the data transfer.

Upcoming requests and requirements at other telescopes, like at the MIT Haystack Observatory, made it necessary to find a more common solution. The Monitoring and Control Infrastructure (MCI) group was founded, which wrote a white paper defining the goals and implementations of a centralized monitoring system. In 2014, the IVS Task Force on Seamless Auxiliary Data Archives was founded to realize central monitoring.

While the Wettzell SysMon widely followed the suggestions from the white paper, administrative decisions at the MIT Haystack observatory led to another development which focused on the end-points of the new VGOS telescopes. The central monitoring is now under the responsibility of the NASA Field System (see TIG). SysMon was continued locally and is now in use again for the Wettzell observatory.

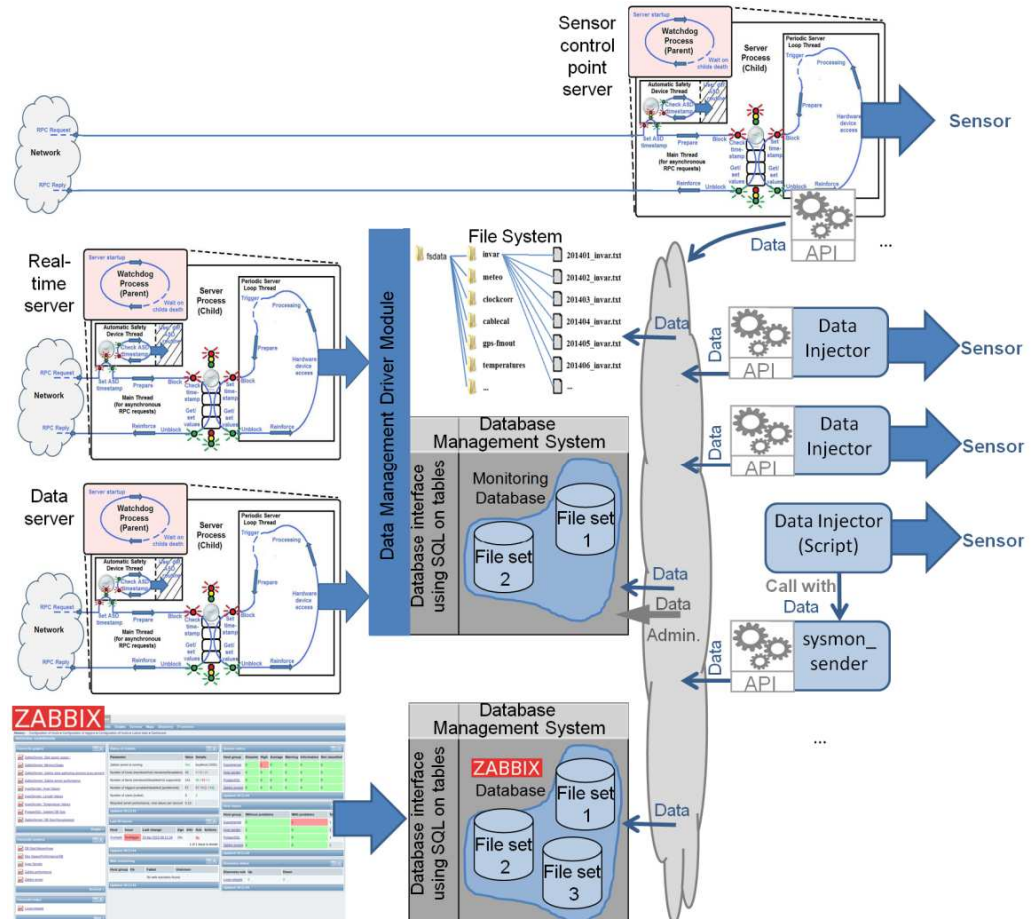
The Wettzell SysMon is an API to collect monitoring data. It offers a C++ programming interface and a sender program and manages all administrative things behind. SysMon uses PostgreSQL as database for current and mid-term values and local file trees to store historic data sets. It does not implement any graphical user interface. One main aspect is to use the professional, industrial tools for that. Because Zabbix found a brought acceptance at the Wettzell observatory, it is used to present the collected data. Therefore, SysMon was extended to support Zabbix, while both systems are in principle independent. Currently, SysMon is used for analysis and historic data while operational and diagnostic data are mainly injected to Zabbix.

The software is open-source using the GNU Lesser license, but currently only on a local SVN repository of the Wettzell observatory.

There is another communication channel using SCP-copied data files with a structure defined by the simple_structured_conf of the Wettzell observatory. A thread is already integrated into the server of the VLBI

remote control software “e-RemoteCtrl” (see later in the remote control section). The files are read periodically and integrated into SysMon/Zabbix. It currently supports the main data taken from the NASA Field System.

The design is the following:



Evaluation:

Qualifier	Priority (0-5)*
Industrial importance	5 (0)
Open source	5 (5)
Use for VLBI	5 (5)
User community	5 (2)
Developer community/ Size of developer team	5 (2)
Readiness for production and release policy	5 (5)
Community support	5 (2)
Documentation	3 (2)
Platform support (Linux for central server; also others for data provider)	5 (2)
Simplicity to install	5 (5)
Simplicity to use for non-specialists	5 (5)
Simplicity to extend with own parts	5 (5)
Supported techniques/protocols for the use in VLBI	5 (3)
Web based access possibilities	5 (0)
Usability for distributed VLBI networks	5 (5)
Coherent design and use of programming languages	5 (5)
State-of-the-art techniques	5 (5)
Access control	5 (0)
Maintainability and possibility of migration	5 (5)
Future relevance	5 (3)
Individual notes:	5 (5)
<ul style="list-style-type: none"> - SysMon is a helpful utility - See more about Zabbix in the following sections - All developers are part of the Wettzell observatory, which leads to short integration and change times 	
Priority	4,90 (3,38)

* Evaluation of SysMon is in parenthesis; the highest value from the Zabbix values and SysMon values is taken for the final prioritization number.

3.1.3 Telegraf - InfluxDB - Grafana (TIG) (NASA FS)

A first presentation about the use of TIG for VLBI was given by David Horsley (NASA, GSFC, NVI) during the IVS TOW 2017. It is the preferred system by the NASA Field System development group. Therefore, it will play an extended role especially for all telescopes using the NASA Field System to run their VLBI observations.

Illustrations can be found here: <https://grafana.com/grafana>

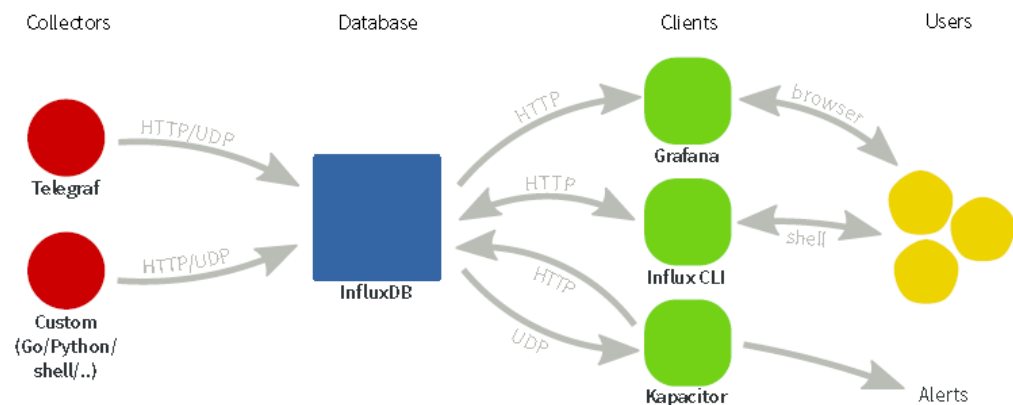
The overview from the TOW 2017 can be found here

<http://www.haystack.mit.edu/workshop/TOW2017/files/Seminars/horsley-tig-notes-2017.pdf>, which is copied into this deliverable report. It is written by David Horsley. The following is taken from these notes (with special permission by David Horsley):

An up-to-date version of the documentation for the NASA Field System can be found in the FS online documentation on <https://lupus.gsfc.nasa.gov/fs/docs/monitoring/> (a user account and password is required)

The Telegraf, InfluxDB and Grafana (TIG) provide a system for collecting, storing, processing, and visualizing time-series data. The three component are loosely coupled together and each swapped for an alternative package. The purpose of this document give an overview of these tools use in VLBI operations and to guide a user through the installation process. The reader is expected to be competent with a Linux OS.

The role of components are as follows:



Telegraf collects data from different sources. Telegraf runs on every computer where you want to collect statistics. Telegraf includes plugins for collecting data on things such as:

- *disk usage and load*
- *system load*
- *network load and performance*
- *process statistics*
- *system sensors*

The VLBI branch, provided in the FS repository, contains plugins for:

- *The Field System (log, schedule, some RDBE data)*
- *Modbus Antennas (Currently Patriot 12m of the AuScope/GGAO generation)*
- *MET4 meteorological system via metserver*
- *RDBE multicast*

InfluxDB is a time-series database. It offerers high-performance compression and retrieval for this type of data. It also has functions for processing and manipulating the data. It is similar to relational databases you may be familiar with, but is far more efficient at handling time-series data. While InfluxDB has an SQL-like query language, it is distinct and it is best to consider it as a new system.

Like an SQL type database, InfluxDB method of getting data is a push model. This means the clients, the programs with the data, initiate the connection and write to the database. If you require a fetch model, you must write your own collector program. Telegraf fill this role for some purposes.

The load on the system it runs on can be fairly high, depending on the number of points you are monitoring. For this reason, it is worth doing some testing and tuning if you wish to run it on your FS PC. If you can, it is best to run the database server on a separate machine.

The third component Grafana provides the graphical user interface. It allows you to plot historical data, and build (near) real-time dashboards for any metrics that are being written to the database. Grafana should be run on a computer that can access InfluxDB server(s) and the computer(s) you want to monitor from. Grafana runs a web server and you connect to it via your web browser. I have found Google Chrome to give superior performance for Grafana.

Each project is open-source with paid support. Grafana.net² provide premium support for Grafana and InfluxData³ provide the same for Telegraf and InfluxDB. InfluxData also maintain the other open-source

packages Chronograf (similar to Grafana), and Kapacitor (used for alerts and data processing). I will not cover these here, only because I have do not have much experience with them, however both look promising. InfluxData also maintain a commercial version of InfluxDB with cluster support and admin tools aimed at larger scales.

Discussions during the TOW 2017 showed that there is already a plugin to support data exchange between Zabbix and Grafana. Investigations on that are required.

Acknowledgement: Special thanks to David Horsley for his permission to include the TIG documentation and for his willingness of future support to integrate TIG.

Evaluation:

Qualifier	Priority (0-5)
Industrial importance	0
Open source	5
Use for VLBI	5
User community*	5
Developer community/ Size of developer team	3
Readiness for production and release policy	5
Community support	5
Documentation	4
Platform support (Linux for central server; also others for data provider)	5
Simplicity to install	4
Simplicity to use for non-specialists	3
Simplicity to extend with own parts	5
Supported techniques/protocols for the use in VLBI	5
Web based access possibilities	5
Usability for distributed VLBI networks	5
Coherent design and use of programming languages	5
State-of-the-art techniques	5
Access control	4
Maintainability and possibility of migration	4
Future relevance	5
Individual notes:	5
<ul style="list-style-type: none"> - US software components are used - Preferred system for the NASA FS - Quasi-industrial products are combined - Possibility to request data using SQL with clear structures - Might be ideal for data analysis - The high data-rates are not really required for the VLBI-data 	
Priority	4,38

* User community is rated with view to the future situation of the use of the NASA Field System at the VLBI sites

3.1.4 Radboud Radio Lab VLBI monitor (EVN, mm-VLBI)

The VLBI Monitor code is open source since Feb. 2nd, 2016 and was developed for the Event Horizon Telescope, which operates mm-frequency observations with radio telescopes world-wide. The server is available at https://gitlab.science.ru.nl/radiolab/vlbi_monitor under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. All (example) clients are available at https://gitlab.science.ru.nl/radiolab/vlbi_monitor_client under the terms of the MIT License (for compatibility with closed source software at the observatories). The software is independently developed by Dr. Pim Schellart, Radboud Radio Lab, Department of Astrophysics, Research Institute for Mathematics, Astrophysics and Particle Physics (IMAPP), Radboud University, Nijmegen, The Netherlands (<http://www.astro.ru.nl/~pschella>).

Issues and feature requests can be filed at: https://gitlab.science.ru.nl/radiolab/vlbi_monitor/issues for the server and https://gitlab.science.ru.nl/radiolab/vlbi_monitor_client/issues for the (example) clients.

A more elaborate description of the design of the VLBI monitor is available at https://gitlab.science.ru.nl/radiolab/vlbi_monitor/blob/master/DESIGN.md, where this information was taken from.

It supports an individual real-time data monitoring, enables communication and log-file exchange, and stores the data for later analysis. The stations have to write their own programs and after that they can send the collected values using the clients. Used techniques are HTTP, JSON RPC 2.0/BSON, Go programming language for the server with Mgo package for MongoDB, MongoDB for the database.

Evaluation:

Qualifier	Priority (0-5)
Industrial importance	0
Open source	5
Use for VLBI	5
User community	1
Developer community/ Size of developer team	2
Readiness for production and release policy	3
Community support	2
Documentation	2
Platform support (Linux for central server; also others for data provider)	2
Simplicity to install	4
Simplicity to use for non-specialists	3
Simplicity to extend with own parts	3
Supported techniques/protocols for the use in VLBI	3
Web based access possibilities	5
Usability for distributed VLBI networks	4
Coherent design and use of programming languages	2
State-of-the-art techniques	5
Access control	3
Maintainability and possibility of migration	2
Future relevance	2
Individual notes:	2
<ul style="list-style-type: none"> - Individual project for the EHT project - Single developer - Does not play a wider role in other communities - Fulfills requirements also for other VLBI tasks - Might be interesting if data from the mm-VLBI sites should be integrated 	
Priority	2,86

3.1.5 Monitoring and Control Infrastructure MCI (MIT Haystack Observatory)

Upcoming requests and requirements at other telescopes, like at the MIT Haystack Observatory, made it necessary to find a more common solution. The Monitoring and Control Infrastructure (MCI) group was founded, which wrote a white paper defining the goals and implementations of a centralized monitoring system (see also Zabbix/SysMon section).

While the Wettzell SysMon widely followed the suggestions from the white paper, administrative decisions at the MIT Haystack observatory led to another development which focused on the end-points of the new VGOS telescopes.

MIT MCI is a very individual development for the US-VGOS antennas using Python programs. The focus is more on the specific hardware. The central monitoring is now under the responsibility of the NASA Field System (see TIG). There is no information or documentation available for a more common use. It seems that TIG will be the central part also for MCI.

Evaluation:

Qualifier	Priority (0-5)
Industrial importance	0
Open source	1
Use for VLBI	5
User community	2
Developer community/ Size of developer team	1
Readiness for production and release policy	2
Community support	0
Documentation	1
Platform support (Linux for central server; also others for data provider)	1
Simplicity to install	3
Simplicity to use for non-specialists	2
Simplicity to extend with own parts	1
Supported techniques/protocols for the use in VLBI	1
Web based access possibilities	0
Usability for distributed VLBI networks	4
Coherent design and use of programming languages	3
State-of-the-art techniques	3
Access control	0
Maintainability and possibility of migration	3
Future relevance	1
Individual notes:	1
<ul style="list-style-type: none"> - Individual project for the US-VGOS MIT project - Not scalable - TIG will play the role of the central monitoring and MCI just offers programs to offer data of individual hardware 	
Priority	1,67

3.1.6 Industrial monitoring tools (Nagios, Zabbix, etc.)

Originally evaluated by M. Schönberger (BKG) for Wettzell Observatory and adapted, edited and extended by A. Neidhardt

Network monitoring systems

There are plenty of network monitoring tools available with GPL, LGPL or BSD licenses.

General links and documents

http://en.wikipedia.org/wiki/Network_monitoring
http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems
<http://workaround.org/try-zabbix>
http://staff.science.uva.nl/~jblom/gigaport/tools/monitor_tools.html
<http://www.thegeekstuff.com/2009/09/top-5-best-network-monitoring-tools/>
<http://www.findbestopensource.com/product/pandorafms>
<http://sixrevisions.com/tools/10-free-server-network-monitoring-tools-that-kick-ass/>
<http://ftp.heanet.ie/disk1/people.ee.ethz.ch/%257Eoetiker/webtools/rrdtool-1.0.x/rrdworld/cacti.html>

MRTG and RRDTool

MRTG and RRDTool are data logging and graphing systems for time series. They generate HTML pages which can be accessed using web browsers.

RRDTool (written by Tobi Oetiker) has been developed on the basis of MRTG to improve its limitations, mostly in the areas of performance and graphing flexibility.

It can be easily integrated in shell scripts, perl, python, ruby, lua or tcl applications.

Many web graphing tools use it (cacti, munin ...).

Drraw

Simple Web interface to generate interactive graphs of rrd database files. It does not care how the .rrd files get their data.

Torus

Cacti like web frontend for RRDTool.

Cacti

Installation on Debian Squeeze failed because of errors while connecting to MySQL database as "admin". Limited on PHP, Databases RRDTools and MySQL.

Advantages:

- Data input method with possibility to import several values per command
- CSV data export function

Disadvantages:

- MySQL database is only supported
- Needs RRDTools

Data input methods and data queries:

Cacti uses an internal data input method to send data to Cacti. You have to write a script or program which prints the value or values which you want to inject.

Cacti gathers information which values should be grabbed using data queries. It enables to react on e.g. interface changes. It is necessary to define a trigger to inform Cacti about changes so that the data queries are triggered.

Spine, the fast polling mechanism using multiple threads:

There is a faster polling mechanism written in C, called Spine. It uses multiple threads to fetch data.

Conclusion:

It is a nice tool to generate pretty graphs. There are many plug-ins and themes. Unfortunately, only one polling task performs all queries. The polling process is started as a “cron-job”. There is no possibility to define different polling time intervals. A suggestion to enable different time settings is to run more instances of Cacti. Therefore, it might be a good solution to monitor atomic clocks or the meteorology data.

Collectd

It is a very lightweight daemon written in C. It runs on POSIX machines and its ideal for embedded systems.

Collectd cannot create graphics. It can write to RRDTool using a plug-in.

Cricket

Cricket is MRTG with a configuration tree instead of a configuration file. It is a very “thin” tool (needs only 2MB disk space) for graphing time series with RRDtool.

It needs no database.

Icinga

It is a fork of Nagios with modern web interface and additional database connectors (mySQL, Oracle, PostgreSQL) and an API to integrate numerous extensions without complicated modification of the Icinga core.

Munin

Munin is a flexible and powerful solution to generate graphs for web presentation.

- written in Perl
- using RRDtool
- CGI/fast-CGI
- since v1.4 dynamic template using javascript
- Uses Munin-nodes to collect data on multiple machines
- Very active community

Nagios

Local tests of Nagios showed that it is powerful, flexible, highly scalable and very reliable, but very hard to install and configure. Tested were some SNMP devices, where the configuration showed the difficult configuration quite well. The web interface is also quite bulky.

This is also confirmed on different web pages, e.g. the OMD-web page: The standard installation of Nagios shows a disappointing behavior due to installation and configuration. They suggest using Check_MK and their Open Monitoring Distribution (OMD).

Only the core version of Nagios is open-source licensed. The main part is commercial. But Nagios is one of the most used distributions for industrial environments.

OMD - Open source Monitoring Distribution in combination with Nagios

OMD is a composition of open source monitoring tools around Nagios. Running of several instances is supported. Therefore, user accounts are required and created.

commands	
omd create projectname	creates the omd instance and user projectname on host
omd start muc	starts the muc instance

commands	
sudo passwd muc	set password for user muc
su muc	login as user muc
omd stop	stops the omd instance of logged in user

More information (in German): http://mathias-kettner.de/nagios_schulung_fortgeschritten_check_mk.html

Observium

Observium is a PHP/MySQL-based Network Observation and Monitoring System (NOMS) which collects data from devices using SNMP. The data are presented with a web interface. It makes heavy use of the RRDtool (written by Tobi Oetiker) package. Observium has a number of simple core design goals driving its development: minimum interaction, maximum automation and maximum accessibility of information. These design goals have resulted in a slightly unconventional monitoring system with almost no individually customizable settings per device. Almost everything that can be monitored is automatically discovered.

This automatic discovery reduces the flexibility to adapt it to individual tasks.

OpenNMS

OpenNMS is a free and open-source enterprise network monitoring and management platform. It is developed by The OpenNMS Group with a user/developer community behind.

OpenNMS is a distributed, scalable, monitoring and management platform for all network aspects. Currently the focus is on Fault and Performance Management.

(more: <https://en.wikipedia.org/wiki/OpenNMS>)

OpenNMS is currently also one of the most favored monitoring systems for industrial environments. It is maybe worth to keep an eye on it.

Opsview

- Uses the Nagios-core.
- Supports Plugins

- Graphing tools are built-in (which replace deprecated programs of the RRDtool)
- Reporting is an enterprise feature

Shinken

Shinken is Nagios compatible and for big server scenarios.

Pandora FMS (Pandora Flexible Monitoring System)

Pandora FMS is especially designed to monitor computer networks. The big advantages are the monitoring of status and performance of operating systems, servers, applications, firewalls, databases, routers, etc.

Similar to Zabbix it uses agents on the different computers and can be used to do remote monitoring. The Web interface is PHP and the supported databases are MySQL, Oracle, or PostgreSQL.

Pandora is open source with several commercial parts.

Zabbix

Zabbix is an open-source software written by Alexei Vladishev. It is designed for the monitoring of networks, applications, services, servers, and network hardware. Zabbix can use the databases MySQL, PostgreSQL, SQLite, Oracle, or IBM DB2. It is written in C and PHP.

It uses adaptable agents on the different computers for Linux and Windows systems. The agents can be extended with individual monitoring values/items, see (in German) http://lab4.org/wiki/Zabbix_Agent_erweitern. It also supports SNMP (using the snmp_builder to read MIB lists and to create IOD items), TCP, ICMP, IPMI, JMX, SSH, Telnet. Scripts can use a special sender program to add individual monitoring values, called items. An extended HTTP API allows to fetch graphs or to change and use existing hosts. A special feature is the use of maps. They allow it to draw the logical monitoring structures showing the monitoring hosts as graphical icons. There are already several templates for graphical web pages especially for the monitoring of IT components.

Found advantages are:

- Implementation: server and agent in C, web frontend in PHP
- Supports all basic databases like PostgreSQL, SQLite, MySQL, Oracle and IBM DB2
- Integrated design to get many things out of the box
- Amazing graphing functionality

A good overview can be found here (in German):

http://lab4.org/wiki/Zabbix_Handbuch_Inhaltsverzeichnis

Earlier versions need much system resources.

Some links:

[http://www.realpowerwork.com/fileadmin/documents/linux/Zabbix%201.8.x%20Installation%20\(Ubuntu_Server_8.04\).htm](http://www.realpowerwork.com/fileadmin/documents/linux/Zabbix%201.8.x%20Installation%20(Ubuntu_Server_8.04).htm)

<http://lin4.de/blog/category/zabbix>

<http://www.pro-linux.de/artikel/2/1149/systemueberwachung-mit-zabbix-teil-1.html>

Conclusion

(by M. Schönberger, adapted by A. Neidhardt)

After installing and running the software, Zabbix is a handy, simple, intuitive, integrated network monitoring software. A big advantage is the extended PHP user interface (the Zabbix frontend) for monitoring, reporting, graphing, administrating. This makes it easy to setup standard network monitoring agents with items like CPU load, SNMP queries, etc., even for beginners. This is a big benefit compared to others, like NAGIOS, which uses extended configuration text files. All settings done in the frontend can be converted to template files which can then be integrated into other Zabbix environments.

Other sources for a comparison:

A nice overview can also be found on the Wikipedia pages

(https://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems). The following table is a copy of just open-source products.

While industry mostly uses Nagios, OpenNMS, Zabbix, Cacti, or Pandora FMS, the open-source systems with the most supported features and the largest communities are Cacti, OpenNMS and Zabbix. In comparison they are mostly similar with more or less advantages. Because of the existing experiences at Wetzell using Zabbix, Zabbix is the preferred environment with fastest chance of success because of missing learning curves.

Name	IP SLA Reports	Logical Grouping	Trending	Trend Prediction	Auto Discovery	Agentless	SNMP	Syslog	Plugins	Triggers / Alerts	WebApp	Distributed Monitoring	Inventory	Platform	Data Storage Method	License	Maps	Access Control	IPv6	Latest release date	Latest release version
Argus - The all seeing	Yes	Yes	Yes	Yes	No	Supported	Yes	Yes	Yes	Yes	Viewing, Acknowledging, Reporting	Yes	Unknown	Perl	Flat file, Berkeley DB	Artistic License	No	Yes	Yes	2013-02	3.7
Cacti	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Full Control	Yes	Yes	PHP	RRDtool, MySQL	GPL	Plugin	Yes	Yes	2017-01-29	1.0.0
Check_MK	Via plugin	Yes	Yes	Yes	Via plugin	Supported	Yes	Yes	Yes	Yes	Full Control	Yes	Yes	C, Python	RRDtool	GPL	Plugin	Yes	Yes	2016-11-24	1.2.6p14
collectd	No	No	No	No	Push model, multi-ast possible	Supported	Yes	Yes	Yes	Yes	Viewing	Yes	No	C	RRDtool	GPLv2	No	Apache ACL	Yes	2016-10-07	5.6.1
FreeNATS	Yes	Yes	No	No	Yes	No	No	Via plugin	Yes	In PHP Code	Full Control	No	No	PHP	MySQL	GPL	No	Yes	Unknown	2011-07-09	1.13.3b
Ganglia	No	Yes	Yes	No	Via gmond check in	No	Via plugin	No	Yes	No	Viewing	Yes	Unknown	C, PHP	RRDtool	BSD	Yes	No	Unknown	2016-06-14	3.7.2
Icinga	Via plugin	Yes	Yes	No	Via plugin	Supported	Via plugin	Via plugin	Yes	Yes	Full Control	Yes	Via plugin	C[]	MySQL, PostgreSQL, Oracle Database	GPL		Yes	Yes	2017-03-29	2.6.3
Monitortix	No	No	Yes	Yes	No	No	Yes	No	Yes	Yes	Viewing	Yes	Unknown	Perl	RRDtool	GPL	Unknown	Yes	Yes	2016-10-14	3.9.0
Munin	No	Yes	Yes	Yes	No	No	Yes	No	Yes	Partial	Viewing	Via nodes	Unknown	Perl	RRDtool	GPL	Unknown	Unknown	Yes	2016-08-19	2.0.26
Nagios	Via plugin	Yes	Yes	No	Via plugin	Supported	Via plugin	Via plugin	Yes	Yes	Yes	Yes	Via plugin	C	Flat file, SQL (via ndoutils), MySQL (via Nconf)	GPL	Yes	Yes	Yes[]	2017-02-23	4.3.1[]
NexXMS	No	Yes	Yes	Yes	Yes	Supported	Yes	Yes	Yes	Yes	Full Control	Yes	Yes	C++ Java	MySQL, MS SQL, Oracle, PostgreSQL, SQLite	GPL	Yes	Yes	No	2017-04-03	2.1-M3
Octopussy	No	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Full control	No	Yes	Perl, ASP	MySQL	GPL	No	Yes	Unknown	2014-04-15	1.0.14
OpenNMS	Yes	Yes	Yes	Supported	Yes	Supported	Yes	Yes	Yes	Yes	Full Control	Yes	Yes	Java	JRobin / RRDTool / Apache Cassandra, PostgreSQL	AGPLv3	Yes	Yes	Yes	2017-06-08	20.0.0
Performance Co-Pilot	No	Yes	Yes	No	Yes	Optional, Limited	Yes	Yes	Yes	Yes	Viewing	Yes	Yes	C Perl, Python, POSIX, MingW	Flat file	GPL, LGPL	No	Yes	Yes	2016-11-15	3.11.6
Shinken	Via plugin	Yes	Yes	No	Yes	Supported	Via plugin	Via plugin	Yes	Yes	Viewing, Acknowledging, Reporting	Yes	Via plugin	Python	Flat file, MySQL, Oracle, Graphite, Sslite, MongoDB	AGPL	Yes	Yes	Yes	2016-03-10	2.4.3
Xymon/Hobbit	Yes	Yes	Yes	No	Via Plugin	Via Plugin	Via Plugin	No	Yes	Yes	Viewing, Acknowledging, Reporting	Yes	Via Plugin	C, Shell	Flat file, RRDTool, MySQL via plugin	GPL	Via Plugin	Apache ACL	No	2016-03-24	4.3.27
Zabbix	Yes	Yes	Yes	Yes	Yes	Supported	Yes	Yes	Yes	Yes	Full Control	Yes	Yes	C, PHP	Oracle, MySQL, PostgreSQL, IBM DB2, SQLite	GPL	Yes	Yes	Yes	2017-04-20	3.2.5
Name	IP SLA Reports	Logical Grouping	Trending	Trend Prediction	Auto Discovery	Agentless	SNMP	Syslog	Plugins	Triggers / Alerts	WebApp	Distributed Monitoring	Inventory	Platform	Data Storage Method	License	Maps	Access Control	IPv6	Latest release date	Latest release version

Reference: https://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems



Evaluation:

Qualifier	Priority (0-5)
Industrial importance	5
Open source	5
Use for VLBI	5
User community	5
Developer community/ Size of developer team	5
Readiness for production and release policy	5
Community support	5
Documentation	3
Platform support (Linux for central server; also others for data provider)	5
Simplicity to install	5
Simplicity to use for non-specialists	5
Simplicity to extend with own parts	4
Supported techniques/protocols for the use in VLBI	5
Web based access possibilities	5
Usability for distributed VLBI networks	5
Coherent design and use of programming languages	5
State-of-the-art techniques	5
Access control	5
Maintainability and possibility of migration	4
Future relevance	5
Individual notes:	5
<ul style="list-style-type: none"> - Industrial monitoring tools and environments have a huge community and support; - The use of professional environments is quite valuable - Disadvantages are the close connection to an individual system, specializations on network and IT, and overloaded features - Experiences at the Wettzell observatory show that Zabbix fulfilled all needs in a local environment and can be used for hierarchically distributed systems as well 	
Priority to integrate (Zabbix)	4,81

3.1.7 SKA Telescope Manager (SKA)

SKA develops a completely individual system: see

<http://skatelescope.org/tm/>

But nothing else can be found about already existing parts.

Evaluation:

Qualifier	Priority (0-5)*
Industrial importance	0
Open source	3
Use for VLBI	2
User community	2
Developer community/ Size of developer team	2
Readiness for production and release policy	0
Community support	0
Documentation	0
Platform support (Linux for central server; also others for data provider)	1
Simplicity to install	0
Simplicity to use for non-specialists	0
Simplicity to extend with own parts	0
Supported techniques/protocols for the use in VLBI	0
Web based access possibilities	1
Usability for distributed VLBI networks	3
Coherent design and use of programming languages	2
State-of-the-art techniques	3
Access control	0
Maintainability and possibility of migration	2
Future relevance	3
Individual notes:	0
<ul style="list-style-type: none"> - There is no information available - The licensing policy is not clear, probably open-source - Might play a future role 	
Priority	1,14

* Priorities based on guessing from workshop participations.

3.1.8 Individual Linux scripts

There are several individual monitoring scripts and programs (like dboss and Mark6 recording systems or mk6_scan_plotting.sh at the Wettzell observatory, and so on), which are written to do individual monitoring jobs.

As these programs are so specific to individual tasks at the observatories, a general evaluation is impossible. Therefore, the integration to a central monitoring of a complete VLBI network is not suggested. Nevertheless, the scripts can be extended to support monitoring equipment, e.g. to send out data to specific systems.

Evaluation:

Qualifier	Priority (0-5)
Industrial importance	0
Open source	1
Use for VLBI	5
User community	1
Developer community/ Size of developer team	1
Readiness for production and release policy	0
Community support	0
Documentation	0
Platform support (Linux for central server; also others for data provider)	1
Simplicity to install	1
Simplicity to use for non-specialists	0
Simplicity to extend with own parts	1
Supported techniques/protocols for the use in VLBI	0
Web based access possibilities	0
Usability for distributed VLBI networks	1
Coherent design and use of programming languages	1
State-of-the-art techniques	2
Access control	0
Maintainability and possibility of migration	1
Future relevance	2
Individual notes:	0
<ul style="list-style-type: none"> - There is usually no information available - Very individual - No general use for a central monitoring 	
Priority	0,86

3.1.9 Goddard Mission Services Evolution Center (GMSEC) Architecture

See

<https://gmsec.gsfc.nasa.gov/GMSEC%20FAQs%202014%2004%2023.htm>

Download from: <https://opensource.gsfc.nasa.gov/>

The Goddard Mission Services Evolution Center (GMSEC) middleware or architecture is currently under development by the NASA, Goddard Space Flight Center and will be used by different NASA projects especially for satellite missions. GMSEC is a solution which is applicable to current and future NASA missions, using standardized interfaces, a middleware which implements a software bus, and a flexible design for the integration of individual hardware components.

The GMSEC architecture is no monitoring system in the classic sense but supports aspects of real-time operations (remote control and monitoring) using individual components, like GMSEC Message Visualization/Storage, GMSEC System Display, GMSEC System Agent, GMSEC Alert Notification System Router, and so on.

It currently plays no role for VLBI operations.

Evaluation:

Qualifier	Priority (0-5)*
Industrial importance	0
Open source	5
Use for VLBI	0
User community	3
Developer community/ Size of developer team	2
Readiness for production and release policy	3
Community support	2
Documentation	1
Platform support (Linux for central server; also others for data provider)	2
Simplicity to install	2
Simplicity to use for non-specialists	0
Simplicity to extend with own parts	2
Supported techniques/protocols for the use in VLBI	1
Web based access possibilities	0
Usability for distributed VLBI networks	1
Coherent design and use of programming languages	4
State-of-the-art techniques	5
Access control	2
Maintainability and possibility of migration	2
Future relevance	2
Individual notes:	1
<ul style="list-style-type: none"> - Focus on satellite missions - General attempt but difficult to evaluate for other fields - Might play a future role 	
Priority	1,90

* Priorities based on guessing using web page contents.

3.2 VLBI remote control, operation and automation

While monitoring of current health, analytic and operational states with reporting and notification is one aspect of the work package, actively interacting and manipulating of session observations is the other aspect.

Developing of remote control capabilities was one main aspect of the WP5 of the EU-FP7-funded NEXPreS project. The resulting software “e-RemoteCtrl” is currently the only existing software which deals with several aspects of controlling radio telescopes from remote including security aspects. The software requires the NASA Field System as software which processes observation sessions and controls hardware.

Other attempts deal with individual tasks, like operating real-time e-VLBI sessions or adapting networks dynamically by using continuously optimized schedules.

Already during the developments, also later in the integration phase, and while discussions in board and task force meetings, direct and active control is not wished by most telescope operating companies or institutes. Especially, NASA sites will not give any permit to actively interact with NASA systems. This restrictions lead to a reduced benefit of active control.

Another aspect is the “passive” control. Dynamic observation strategies at the AuScope network follow such a concept, where the sites fetch updated schedules, prepare and run them. Feedback about system states is forwarded to a central coordination site, where the next schedule is prepared. Using this attempt, the sites decided to follow new instructions, which are limited to the capabilities of schedule descriptions implicitly changing the system behavior and do not allow a direct active change at the system which explicitly changes the behavior.

The following sections are a collection of found control strategies and software techniques used for VLBI.

3.2.1 e-RemoteCtrl (Wettzell, O'Higgins, AuScope, partly NyAlesund and Hartebeesthoek, tested for TIGO in Chile)

The work for e-RemoteCtrl started in the year 2008/2009 at the Wettzell observatory. The Wettzell observatory operated three telescopes at three different locations around the world: the 20m Wettzell antenna in Germany, the 9m antenna in O'Higgins in the Antarctica, and the 6m antenna of the TIGO system in Concepción, Chile. Two more antennas were planned at this time, located at the Wettzell observatory to follow the VGOS standards. All antennas should be operated from one operator desk. 2010 to 2013, e-RemoteCtrl was extended during the FP7 project NEXPreS to integrate security mechanisms and a role-based operation. Authentication mechanisms and encrypted data exchange were implemented. Different static and dynamic user roles allow it to restrict the controllable tasks. A one month working journey as University Associate in the School of Land and Food, School of Mathematics and Physics at the University of Tasmania, Hobart in the year 2014 allowed another extended adaption of the software to implement solutions for the needs at the VLBI sites. During the last two years, only bug-fixing was done.

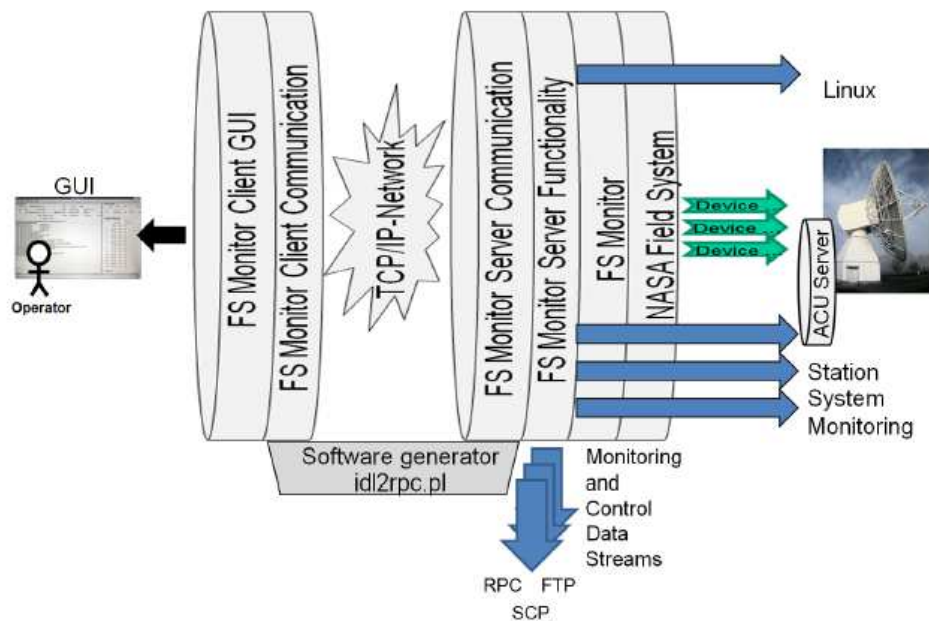
The software is written in C/C++ under the GNU Lesser license and available in the version of 2014/2015 using wxWidgets 2.8. The software currently requires Linux as operating system. It is currently under regular use by the Wettzell observatory and the AuScope antennas. The release change of wxWidgets as framework for the graphical user interface to version 3.0 makes it necessary to write a new release of e-RemoteCtrl. This influences all main parts of the graphical user interface because graphical actions must be handled as events and the parallel processing is done more restrictively. The upgrade is planned for the second half of 2017. Additional plans will extend the communication with HTTP functionalities.

Information can be found here:

http://www.haystack.mit.edu/workshop/TOW2017/files/Seminars/remote_operation_notes.pdf

or in the book "Applied Computer Science for GGOS Observatories" (see <http://www.springer.com/de/book/9783319401379>)

The design principle is shown in the following figure:



The NASA Field System is used to run the regular sessions and control the specific devices needed. A software layer “FS Monitor” is a software module to read data from the shared memory and send commands to the Field System. This module is included to a server which uses a generated communication interface on the basis of SunRPC. The interface generation is done with the Wettzell “idl2rpc.pl” generator. The server also reads additional information from other systems, filters and prepares it, and is additionally able to publish the values and status information on other streaming channels.

The client consists of a graphical user interface using wxWidgets as GUI framework and the generated client component for the communication to the server. It requests all selected status information, presents them in a classic “Field System”-like and more graphical mode, as well as it does the error and alert notification. The remote operator can enter commands, which are then sent to the Field System PC and processed by the Field System.

The whole communication can be encrypted using SSH tunnels. The basic authentication is also done using SSH. The further user-based access control uses the regular Linux mechanisms and special “rbash” shells. Finally, the role-based authentication identifying the final operator rights is part of the e-RemoteCtrl server.

e-RemoteCtrl is currently the only complete remote control software for the NASA Field System. It will be integrated to the NASA Field System software branch in the version using wxWidgets 3.0 in the year 2017.

The server of e-RemoteCtrl directly supports the sending of data streams to the Wettzell SysMon/Zabbix environment. Regularly and on each state change, the server copies a file with all status information from the NASA Field System to the SysMon/Zabbix server using SCP. This behavior can be configured with the settings in the server configuration file.

Evaluation:

Qualifier	Priority (0-3)
Industrial importance	0
Open source	3
Use for VLBI	3
User community	2
Developer community/ Size of developer team	1
Readiness for production and release policy	3
Community support	2
Documentation	3
Platform support (Linux for central server; also others for data provider)	1
Simplicity to install	3
Simplicity to use for non-specialists	3
Simplicity to extend with own parts	2
Supported techniques/protocols for the use in VLBI	3
Web based access possibilities	2 (3)
Usability for distributed VLBI networks	2 (3)
Coherent design and use of programming languages	3
State-of-the-art techniques	3
Access control and network security	3
User control management	3
Local supervisory	3
Safety implementations	1
Maintainability and possibility of migration	1 (3)
Official certification or product certification	0
Future relevance	2
Individual notes:	2 (3)
<ul style="list-style-type: none"> - Limited by site restrictions - Well tested - Complete system partly developed during FP7 project - Open-source software - Currently the only complete remote control software for the NASA Field System - Server supports data propagation of status information to SysMon/Zabbix → server might be quite interesting 	
Priority	2,16 (2,36*)

* Evaluation of the server of e-RemoteCtrl.

3.2.2 Dynamic Observation (AuScope tests)

A first report can be found here: ftp://ivscc.gsfc.nasa.gov/pub/general-meeting/2016/pdf/017_lovell_etal.pdf

Another report about one year of dynamic observation was given during the IVS EVGA meeting at Gothenborg, Sweden.

Future VLBI observations of the International VLBI Service for Geodesy and Astrometry (IVS) use the extended network of the new VGOS telescope. The regular and almost continuous observations increase in cost and labor. Centralized, automated and adaptive scheduling and operation might be one of the key features. A first attempt and test setup is done by the AuScope network in Australia and Tasmania using a Dynamic Observing concept. Dynamic Observation trials were carried out in 2016.

Each antenna facility is treated like a dynamic resource, which can dynamically notify its availability to a centralized scheduling and correlating facility. This operation center creates schedules of 15 minutes duration taking into account which resource is available and what schedule is best for the final VLBI observation network. Different to the remote control philosophy, where a control center takes over the real control of the different antennas dealing with all the security and safety issues, the dynamic observing approach keeps the complete control local while allowing stations to participate in a centrally coordinated and optimized observing program. The following information is taken from the proceedings paper referred to above.

The main part of the operation center is an installation of the VieVs scheduler using Octave instead of Matlab. It dynamically produces the schedule for the coming next 15 minutes (which is the minimum time interval to obtain tropospheric solutions), taking into account if a telescope is available or not, which sources were observed in the previous interval, and information about cable wrap and antenna positions. Optimizations of the sky coverage over 24 hours are currently not included. All relevant information about each schedule interval is published on a web server. Finally, the software generates one VEX file the past 24 hours of observations for the correlator.

Each antenna site uses a script/program “dysched.pl/mondump” to send status data to the operation center and to retrieve the new schedules. Additionally, the operators use a graphical application to notify the availability of the antenna or cancel it while the session is

processed. The status information is also displayed on the web server. Tests used the antennas at Kathrine, Yarragadee, Hobart, and partly Hartebeesthoek.

Finally, the data was shipped to the operation center at the University of Tasmania in Hobart. An installation of the DiFX software was used to correlate 24 hour intervals using the prepared VEX file.

The Dynamic Observation idea is a nice attempt to optimize networks after antennas report failure states or drop-outs. The real advantage for the analysis is not completely clear yet. Dynamic Observation also does not reduce the manpower required to run the whole VLBI network, because the control and therefore the shifts are in the responsibility of the antenna sites. This can only be reduced if automation is increased, which will again open the topic of safety.

Evaluation:

Qualifier	Priority (0-3)
Industrial importance	0
Open source	3
Use for VLBI	3
User community	1
Developer community/ Size of developer team	1
Readiness for production and release policy	3
Community support	1
Documentation	1
Platform support (Linux for central server; also others for data provider)	1
Simplicity to install	3
Simplicity to use for non-specialists	3
Simplicity to extend with own parts	2
Supported techniques/protocols for the use in VLBI	2
Web based access possibilities	3
Usability for distributed VLBI networks	3
Coherent design and use of programming languages	2
State-of-the-art techniques	2
Access control and network security	3
User control management	0
Local supervisory	3
Safety implementations	0
Maintainability and possibility of migration	3
Official certification or product certification	0
Future relevance	2
Individual notes:	1 (3)
<ul style="list-style-type: none"> - Interesting attempt - Central operation might be extended with solutions for e-VLBI control by JIVE to command e-VLBI equipment - The big advantage is not completely clear - Data exchange of monitoring data is solved with "dysched.pl/mondump" → programs might be useful to retrieve monitoring data 	
Priority	1,84 (1,92*)

* Evaluation of the programs which are extensions to the NASA Field System.

3.2.3 Remote control and monitoring of e-VLBI experiments at JIVE ERIC

By Harro Verkouter (JIVE ERIC)

e-VLBI is a real-time observing mode of the European VLBI Network (EVN). It is fundamentally different from disk based observing and correlation, which both are run as independent batch jobs, typically weeks or more apart. During an e-VLBI run the correlator requires direct access to specific equipment at the stations for real-time control and monitoring.

This section briefly lists the control- and monitoring tools which are used to perform a real-time observation.

The 'runjob' control program at the correlator

The main GUI program to run a correlator job is a Python/Qt script which accommodates e-VLBI. Its responsibilities when running a real-time correlation are:

- allocating correlator compute nodes and correlator input nodes. At incoming real-time data rates of 1024Mbps per station or higher, resource planning becomes an issue because of hardware limitations
- configuring and starting the JIVE Software Correlator (SFXC) according to the results of the resource planning
- configuring a station's data sending equipment dynamically such that data streams can be routed or downsized based on available hardware or network performance. Currently supported data sending equipment types are Mark5 recording units and HAT-Lab's FiLa10G (see <http://www.hat-lab.com/hatlab/>). The Mark5 recording system is a hard-disk based VLBI recording system jointly developed by MIT Haystack Observatory and Conduant Corporation of Boulder, Colorado. It is based around generic server hardware combined with in-house developed hardware to interface to the specific VLBI hardware. The server runs a standard Linux operating system, allowing for development of control software in a programmer friendly environment whilst offloading the data intensive tasks as much to dedicated hardware as possible. Both devices, Mark5 and FiLa10G, use a different command set as well as a different protocol.
- starting and stopping of the actual data transfers from the stations to JIVE

Control software of the Mark5 recording unit and the correlator

One of the pieces of equipment being partially controlled from 'runjob' at the correlator is the Mark5 recording unit at a station. It is the device transforming a stations' raw VLBI data into a stream of ethernet packets to the correlator. The existing control software of the recording unit turned out to be unable to support the e-VLBI observing mode due to various issues. This drove JIVE to develop replacement control software, which became known as 'jive5ab' and is completely written in C++. The control interface to the Mark5 unit is standardized through the VSI/S protocol (see http://www.vlbi.org/vsi/docs/2003_02_13_vsi-s_final_rev_1.pdf) - a simple ASCII based protocol over a TCP/IPv4 socket connection.

The jive5ab control software has built-in performance monitoring on all transfers. A second monitoring feature is that jive5ab enables a client to extract data from a running transfer without interrupting it. This allows for detailed inspection of the contents of the data stream without stopping the observation.

jive5ab is also used on the correlator input nodes to read data from the network. These are generic Linux machines which are part of the SFXC cluster but have been outfitted with a 10Gbps connection to the external network. If data is sent to JIVE using the VLBI Transport Protocol (VTP), detailed network statistics like packet loss and/or reordering can be collected from each jive5ab instance. Data extraction without interruption can also be used on these nodes to inspect the received data stream.

Monitoring

During an e-VLBI observing run no automated monitoring takes place. SFXC's own diagnostic tools, the real-time weight- and fringe plot, are the main monitoring points, actively being observed by a human operator. The fringe display shows the integrated fringe amplitude for all baselines to one reference station and is published on the web (see <http://services.jive.nl/sfxc/fringe.html>). The weight display shows the fraction of good data per station. The former is a direct proxy for whether a station is set up correctly (correct time stamps, frequency, pointing) whereas the latter is a direct proxy for a station's data transfer quality. Irregularities in the weight plot or loss of fringe indicate a malfunction.

A variety of smaller tools were developed which can be used to diagnose the cause of malfunction.

- 'controlEvlbi' A Perl/Tk GUI program to review or modify the per-station network parameters, e.g. control/data IPv4 addresses and protocol information. It has a limited diagnostic tool built in: the IPv4 address(es) from the database can be 'ping'ed' to see if they are reachable.
- 'controlmk5/'controlstation': These Perl/Tk GUI programs can, from a central location, send VSI/S commands or queries to Mark5 units at JIVE ('controlmk5') or at the remote stations ('controlstation'). The 'controlstation' program was phased out and 'controlmk5' was replaced by a more modern, Python/Qt based, version. These programs connect to the Mark5 control software using TCP/IPv4 connections and parse the reply/ies into GUI styles or updates.
- 'Transfer monitoring': A small Python/Qt utility which can be launched from 'runjob'. For each participating station it allows to sample the sending data rate at the station, the received data rate at the correlator input node, its packet loss/reorder statistics or the data rate into the correlator. It can also run a canned sequence of extracting some data from the incoming data stream and decode time stamps found therein to check for correctness.
- 'vlbish': A Python based command-line interface (CLI) utility similar in functionality to the GUI based 'controlmk5' combined with 'controlstation'. It being a CLI allows for more advanced command/query distribution: aliases of groups-of-machines can be created to quickly send a the same command to many stations. 'vlbish' links into the database at JIVE for resolving symbolic names (e.g. a station's two letter code) into IPv4 address for ease of use. It has many other useful features like periodically executing monitoring queries, executing of batch scripts or communicate with the FiLa10G, which communicates via a non-VSI/S compliant protocol. The GUI lacks this functionality.

Evaluation:

Qualifier	Priority (0-3)
Industrial importance	0
Open source	1
Use for VLBI	3
User community	3
Developer community/ Size of developer team	1
Readiness for production and release policy	3
Community support	1
Documentation	1
Platform support (Linux for central server; also others for data provider)	1
Simplicity to install	3
Simplicity to use for non-specialists	3
Simplicity to extend with own parts	1
Supported techniques/protocols for the use in VLBI	2
Web based access possibilities	1
Usability for distributed VLBI networks	3
Coherent design and use of programming languages	2
State-of-the-art techniques	2
Access control and network security	2
User control management	0
Local supervisory	3
Safety implementations	0
Maintainability and possibility of migration	2
Official certification or product certification	0
Future relevance	2
Individual notes:	1(3)
<ul style="list-style-type: none"> - Environment already established and finished - Complete, independent system - No need to transfer it to another server - Monitoring data might be interesting to be included to a monitoring data system 	
Priority	1,64 (1,72*)

* Use of the monitoring parts.

3.2.4 SSH/VPN-tunnels to or VNC/Remote-desktop of the NASA Field System PC

The simplest and always existing mechanisms to remotely access computers use standard mechanism like VNC, Remote Desktops, SSH, or VPN.

Virtual Network Computing (VNC) or Remote Desktops connections are communication methods for a remote access to the desktop of a PC. They are available for all operating systems and copy the complete keyboard-video-mouse interaction from the local desktop to a remote one. The user sees the desktop and can interact in the same way like sitting in front of the local PC. These mechanisms are quite helpful to directly interact with a PC and programs on the PC. They require a working X-server or graphical window system. External KVM-extenders with network access use VNC or Remote Desktops as well. This setup is quite helpful for rebooting or booting scenarios. They are very resource hungry because the complete graphical information is transferred via the network.

SSH- or VPN-tunnels do not require a graphical window system. It is just necessary that the PC, which should be accessed, runs the specific SSH or VPN server. Remote users can use according clients to connect to specific communication ports and interact with the remote PC using a command shell or tunnel other ports. Especially the tunneling of ports is a suitable way to access other applications offering communication ports. Using VPN, the client PC is connected like it would be in the same network as the remote PC, which should be controlled. All open communication ports of the remote PC can directly be accessed without any additional mechanisms. SSH and VPN just work for communication-related applications, e.g. to request data or to access file systems.

All of these techniques are more or less individual mechanisms and do not really offer security for centralized control. They allow comprehensive access to complete networks or computers and require good connectivity, so that network and security restrictions are central questions.

Evaluation:

Qualifier	Priority (0-3)
Industrial importance	3
Open source	0
Use for VLBI	2
User community	3
Developer community/ Size of developer team	1
Readiness for production and release policy	3
Community support	1
Documentation	2
Platform support (Linux for central server; also others for data provider)	3
Simplicity to install	3
Simplicity to use for non-specialists	3
Simplicity to extend with own parts	0
Supported techniques/protocols for the use in VLBI	0
Web based access possibilities	1
Usability for distributed VLBI networks	1
Coherent design and use of programming languages	2
State-of-the-art techniques	2
Access control and network security	1
User control management	0
Local supervisory	0
Safety implementations	0
Maintainability and possibility of migration	3
Official certification or product certification	0
Future relevance	2
Individual notes:	1
<ul style="list-style-type: none"> - Manually established SSH/VPN-tunnels are always individual solutions - VNC/Remote-Desktops are quite helpful for direct interactions - External KVM-Switches with VNC are helpful for rebooting scenarios - No safety mechanisms ("freezing" KVM is not managed automatically) - No real solution for remote control tasks 	
Priority	1,48

4 Final evaluation statement of the software packages

Software package	Current Max. Priority	Current Priority
Monitoring Environments		
(ZABBIX &) SysMon (Wettzell Observatory)	5	4,90
Industrial monitoring tools (Nagios, Zabbix, etc.) → Zabbix	5	4,81
Telegraf - InfluxDB - Grafana (TIG) (NASA FS)	5	4,38
MoniCA" or "openMoniCA" (Australia Telescope National Facility and AuScope geodetic VLBI telescopes)	5	3,62
Radboud Radio Lab VLBI monitor (EVN, mm-VLBI)	5	2,86
Goddard Mission Services Evolution Center (GMSEC) Architecture	5	1,90
Monitoring and Control Infrastructure MCI (MIT Haystack Observatory)	5	1,67
SKA Telescope Manager (SKA)	5	1,14
Individual Linux scripts	5	0,86
Control and Operation Environments		
e-RemoteCtrl (Wettzell, O'Higgins, AuScope, partly NyAlesund and Hartebeesthoek, tested for TIGO in Chil�)	3	2,36
⇒ data sending server		
e-RemoteCtrl (Wettzell, O'Higgins, AuScope, partly NyAlesund and Hartebeesthoek, tested for TIGO in Chil�)	3	2,16
⇒ complete		
Dynamic Observation (AuScope tests)	3	1,92
⇒ data exchange server		
Dynamic Observation (AuScope tests)	3	1,84
⇒ complete		
Remote control/monitoring of e-VLBI experiments from JIVE	3	1,72
⇒ data exchange		
Remote control/monitoring of e-VLBI experiments from JIVE	3	1,64
⇒ complete		
SSH/VPN-tunnels to or VNC/Remote-desktop of the NASA Field System PC	3	1,48

Legend:

Priority	0-1	1-2	2-3	3-4	4-5
Color					

The priority numbers are qualitative estimations converted to quantitative numbers and based on experience, international relevance and workshop appearances. It is also a current view and might change while the project is ongoing. Therefore, an essential aspect must be taken into account: the interoperability, which is evaluated under the item "Maintainability and possibility of migration". A pure quantitative statement is difficult due to the different characteristics and purposes of the software packages.

Because of the security and safety issues of remote control and operation found after the NEXPreS project using the e-RemoteCtrl software, main focus is laid on having a central data stock with the VLBI network status and health information. Therefore, the main part of the infrastructure consists of monitoring tools.

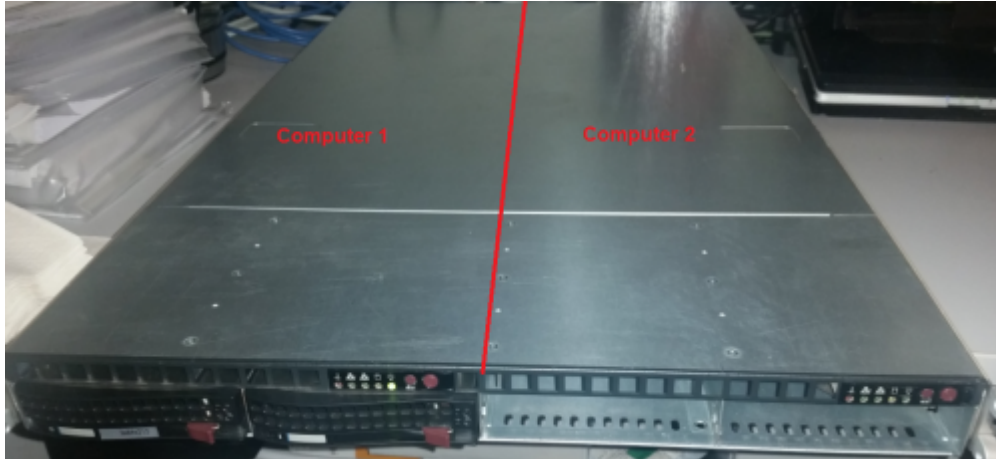
5 Appendix: Zabbix SysMon

The following section is taken from the internal Wettzell notes about how to setup the SysMon/Zabbix system for Wettzell and for a central monitoring environment as a test case.

VLBI SysMon Node

1) Used machines

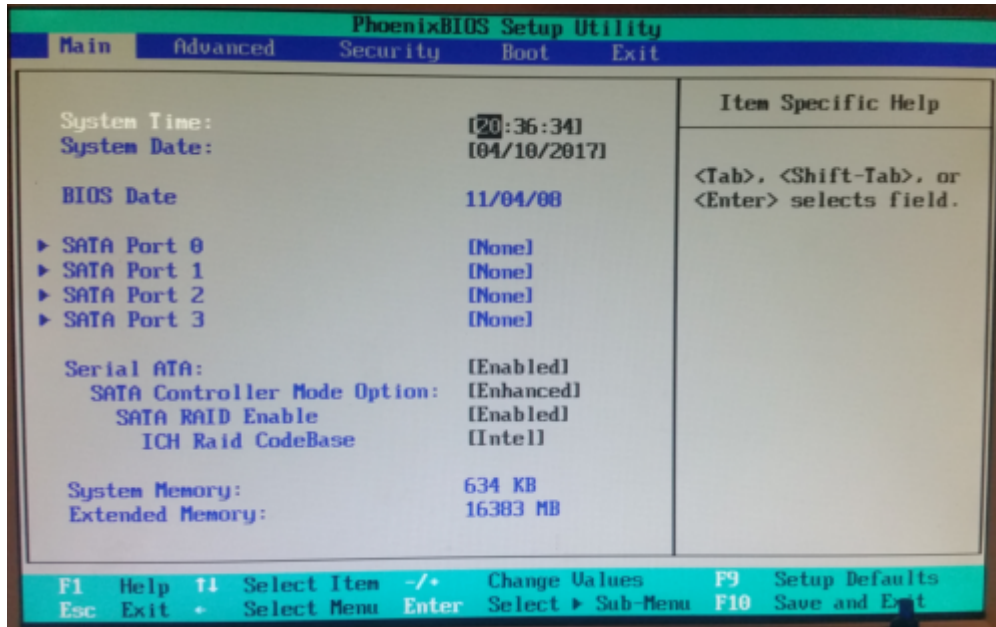
- The used machines are: [Supermicro X7DWT/X7DWT-INF/X7DWT-INF+](#) with two boards in one one-height slot
- [User Manual](#)



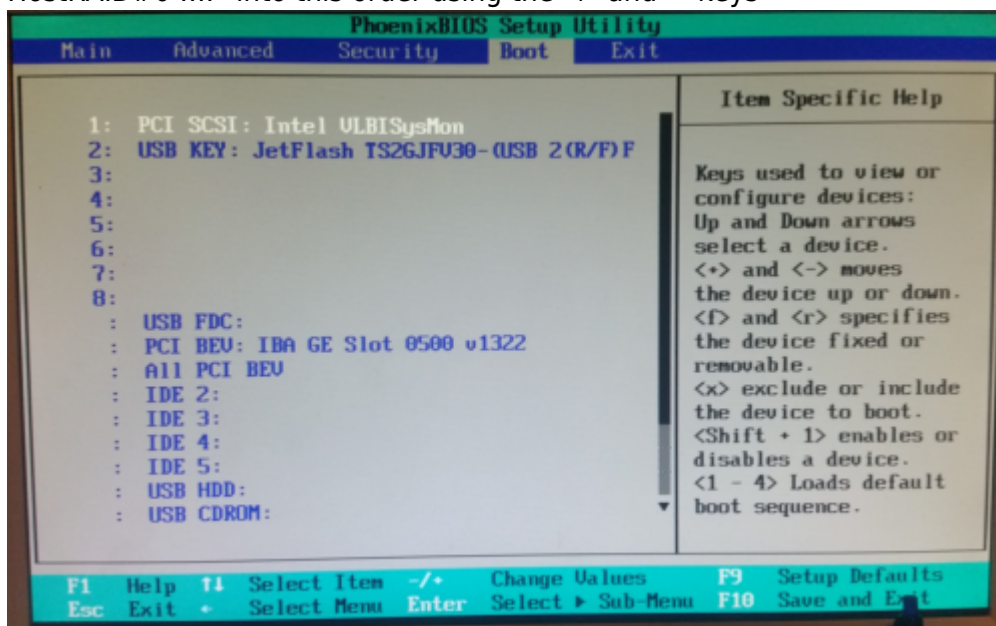
- Both computers must be configured in the same way
 - The first is for the internal management (local telescopes)
 - The second is for the external management (other telescopes etc.)

2) Setup the BIOS

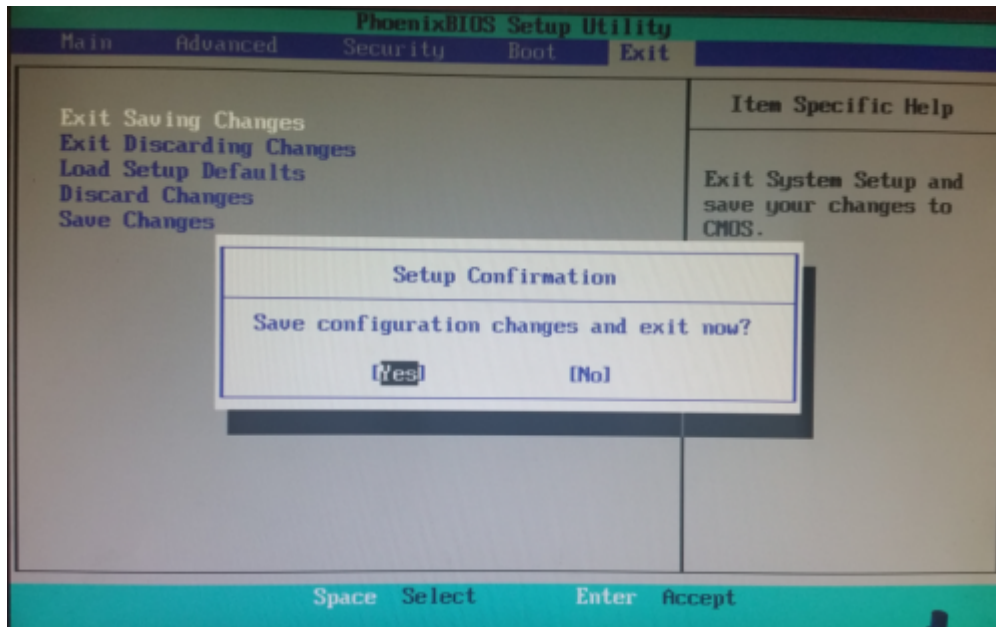
- Open BIOS by pressing “DEL” after startup of the computer
- **Attention: an English keyboard style is used for the following configuration!!!**
- Activate the RAID system in the BIOS
 - See [User manual](#)
 - Enable SATA in the “Main” screen of the BIOS system:
 - “Serial ATA: Enabled”
 - “SATA Controller Mode Option: Enhanced”
 - “SATA RAID Enable: Enabled”
 - “ICH Raid CodeBase: Intel” (we use an Intel ESB2 RAID controller)



- Set the right boot order
 - Change into "Boot" screen of the BIOS system using the arrow keys
 - Push "USB KEY" (maybe also "USB FDC" or other USB devices) and " PCI SCSI: HostRAID#0" into this order using the '+' and '-' keys



- Exit and save the configuration with all changes using the "Exit" screen



3) Configure the RAID 0

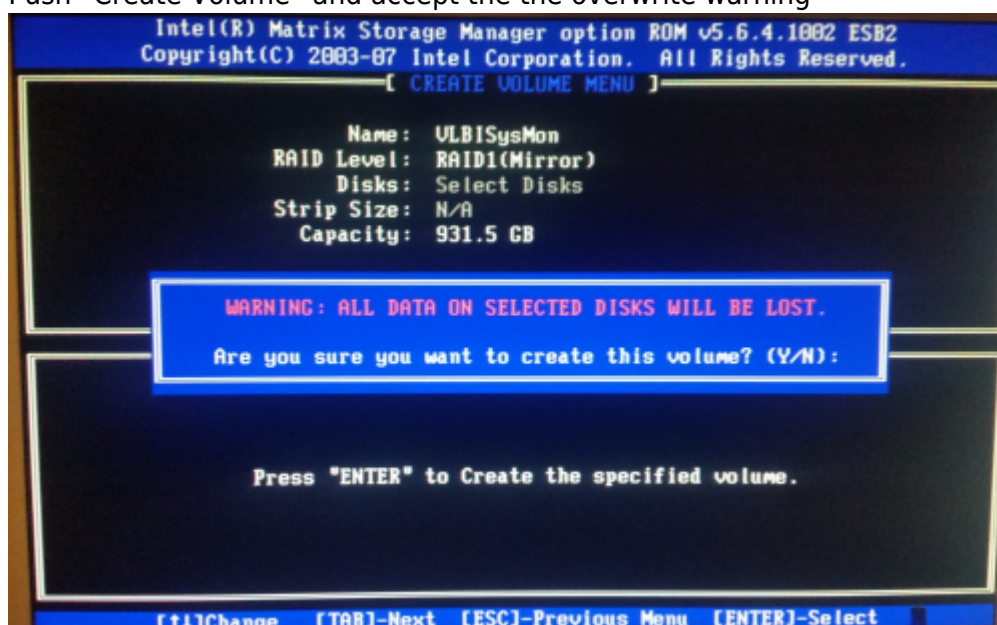
- We use two HDDs with 1TB each as RAID 1 (mirror set) for redundancy on an Intel ESB2 RAID controller.
- **Attention: an English keyboard style is used for the following configuration!!!**
- Open the Intel RAID Configuration Utility using Ctrl+'I'
- Activate the creation of a raid



- Create a mirror set (RAID1) with the name "VLBISysMon" on the existing disks



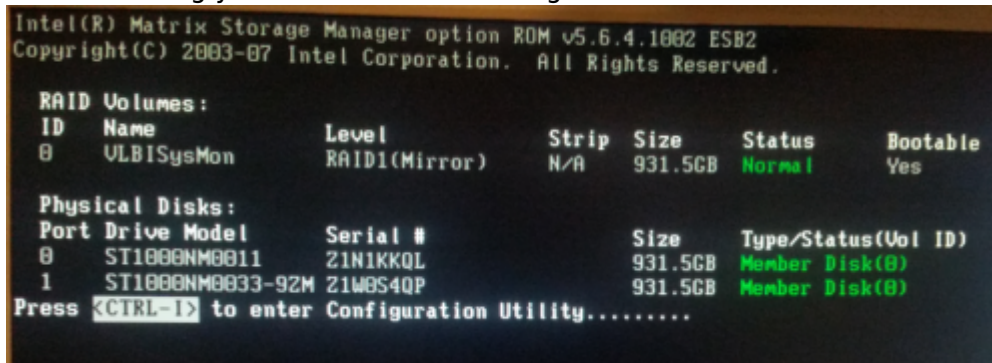
- Push "Create Volume" and accept the the overwrite warning



- Exit the configuration utility



- After rebooting you should see something like this



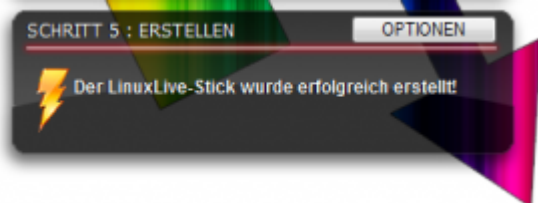
4) Download Ubuntu and install the ISO on a datastick

Methode Windows PC and LinuxLive USB Creator

- Download Ubuntu from <https://www.ubuntu.com/download/desktop> on a separate machine, e.g. a Windows PC
- You will get an ISO-image of the installation
- Download "**LinuxLive USB Creator**" from <https://www.heise.de/download/product/linuxlive-usb-creator-90060> (do not use UNetbootin because it has some failures with 64-bit Linux/Ubuntu systems; see [http://askubuntu.com/questions/544419/cant-run-a-fresh-install-of-ubuntu-14-10-shows-kernel-p](http://askubuntu.com/questions/544419/cant-run-a-fresh-install-of-ubuntu-14-10-shows-kernel-panic) anick)
- Install LinuxLive USB Creator by double click on the installer program and follow the installation instructions
- Start the program LinuxLive USB Creator and create a Linux USB-stick (a detailed instruction can be found here: <https://www.lidux.de/anleitungen/37-ubuntu-1210-usb-stick-installieren-creator>)
 - Select the USB-stick on which the image should be installed
 - Select the ISO image of the Linux system
 - Select no "PERSISTENZ"
 - Select the formatting of the stick with FAT32
 - Click on the flash sign to start the installation



- The installation is finished after you see:



- Close the program and dismount the USB-stick

Methode copying the Ubuntu image on a stick

Ubuntu CD and DVD images can now be written directly to a USB stick, which is a very easy way to make a bootable USB stick.

- Simply choose a CD or DVD image that will fit on your USB stick and copy it on a stick with no partitions.

```
#sudo cp ubuntu-17.04-desktop-i386.iso /dev/sdc
```

- Further informations:

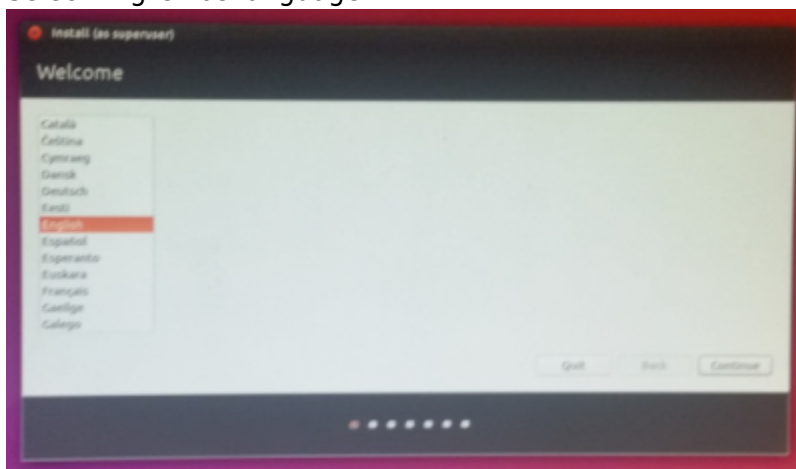
<https://help.ubuntu.com/16.04/installation-guide/amd64/ch04s03.html#usb-copy-iso-hybrid>

5) Install Ubuntu on the SysMon machine

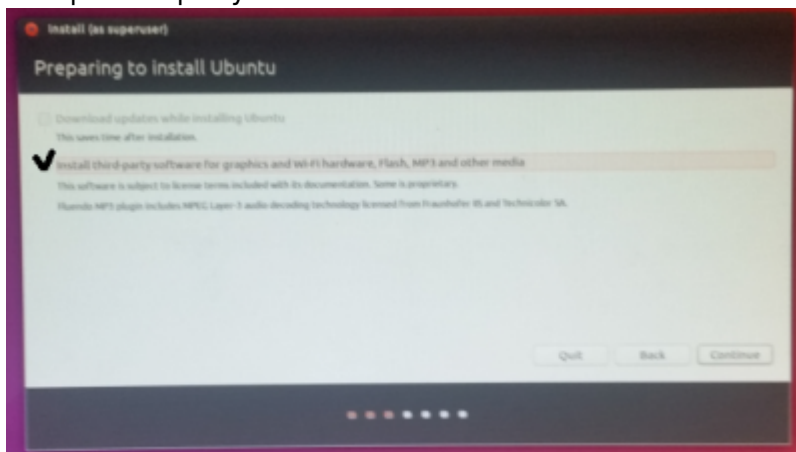
- Insert the stick into a free USB slot
- To connect keyboard and mouse, you need a USB hub, because Supermicro X7DWT just has two USB ports



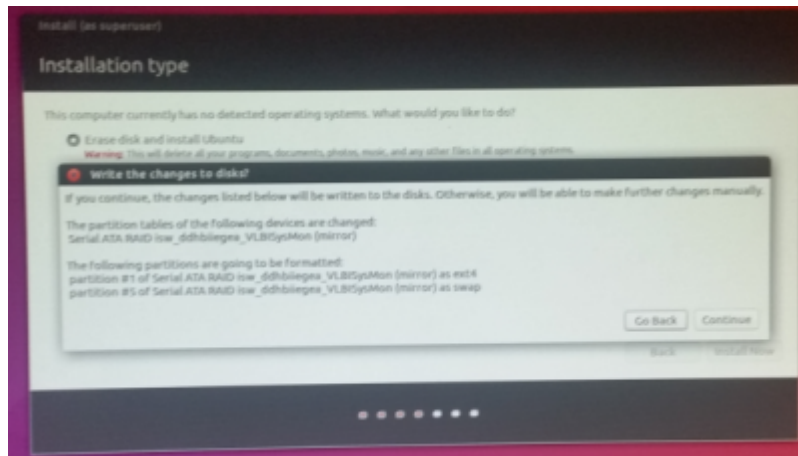
- Start the PC and select "Install Linux" when prompted
- Select English as language



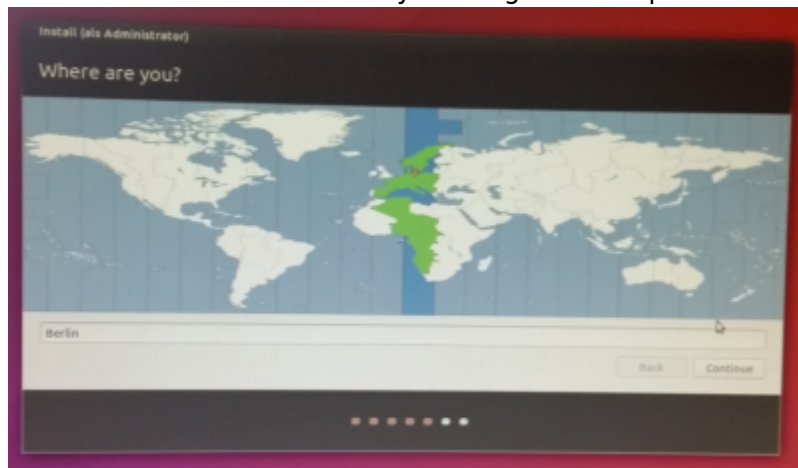
- Accept third-party software



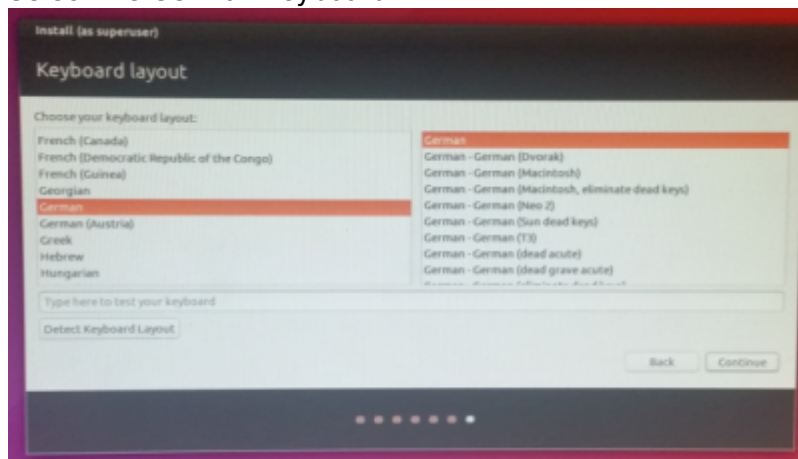
- Select the standard setting "Erase disk and install Ubuntu". Accept the partitioning request.



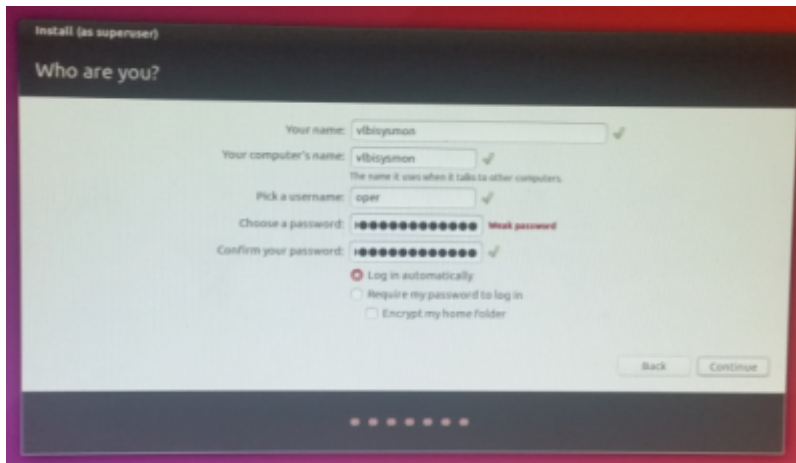
- Select "Berlin" as timezone by clicking onto the position of Berlin on the map



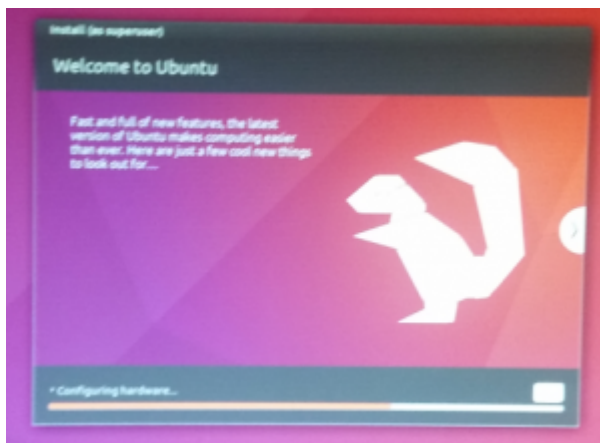
- Select the German keyboard



- Create a the personalization with computer name and your name as "vlbisysmon" and a user "oper" with a dedicated password and select the auto-login.



- Then the installation starts



- Reboot the system and keep the USB-stick in the USB-slot
- Reboot with the live system on the stick (Start Linux from the stick)
- Follow the installation of the GRUB bootloader on the RAID (see https://wiki.ubuntuusers.de/GRUB_2/Reparatur/ (German)) for a standard desktop system
 - Open a terminal (search "term" in the programs)
 - Become root

```
sudo su
```

- Mount the RAID to /mnt

- ```
mount /dev/mapper/isw_ciiaeibbja_VLBISysMon1 /mnt
```

  
(isw stands for Intel Raid Controller; "ciiaeibbja" can be a different string)

- You can check the RAID (other checks can be found here: <https://www.pilgermaske.org/2013/05/dmraid-mainboard-raid-unter-linux-einrichten/> (German))

- ```
lsblk
```

- Mount the required directories for the GRUB installation

- ```
sudo mount -o bind /dev /mnt/dev
sudo mount -o bind /sys /mnt/sys
sudo mount -t proc /proc /mnt/proc
cp /proc/mounts /mnt/etc/mtab
```

- Change into root environment of the installed system on the RAID

- `chroot /mnt /bin/bash`

- Install GRUB (**Attention: the RAID must be used and not a partition on the RAID; this is defined by the device path without an ending number ⇒ not ...\_VLBISysMon1, but ...\_VLBISysMon**)

- `grub-install /dev/mapper/isw_ciiaeibbja_VLBISysMon`

```
root@ubuntu:~# grub-install /dev/mapper/isw_dgdgghheaf_VLBISysMon
Installing for i386-pc platform.
Installation finished. No error reported.
```

- Update GRUB

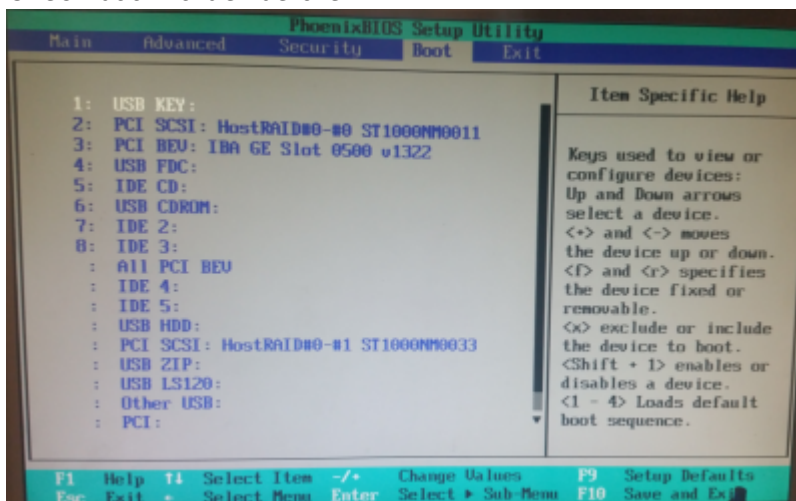
- `update-grub`

```
root@ubuntu:~# update-grub
Generating grub configuration file ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT is set is no longer supported.
Found linux image: /boot/vmlinuz-4.8.0-36-generic
Found initrd image: /boot/initrd.img-4.8.0-36-generic
Found mentest86+ image: /boot/mentest86+.elf
Found mentest86+ image: /boot/mentest86+.bin
done
```

- Exit the changed root privileges

- `exit`

- Reboot the system and extract the USB-stick, so that the system boots from the harddrives. Check bootz order before.



- Open a terminal and become root again

- `sudo su`

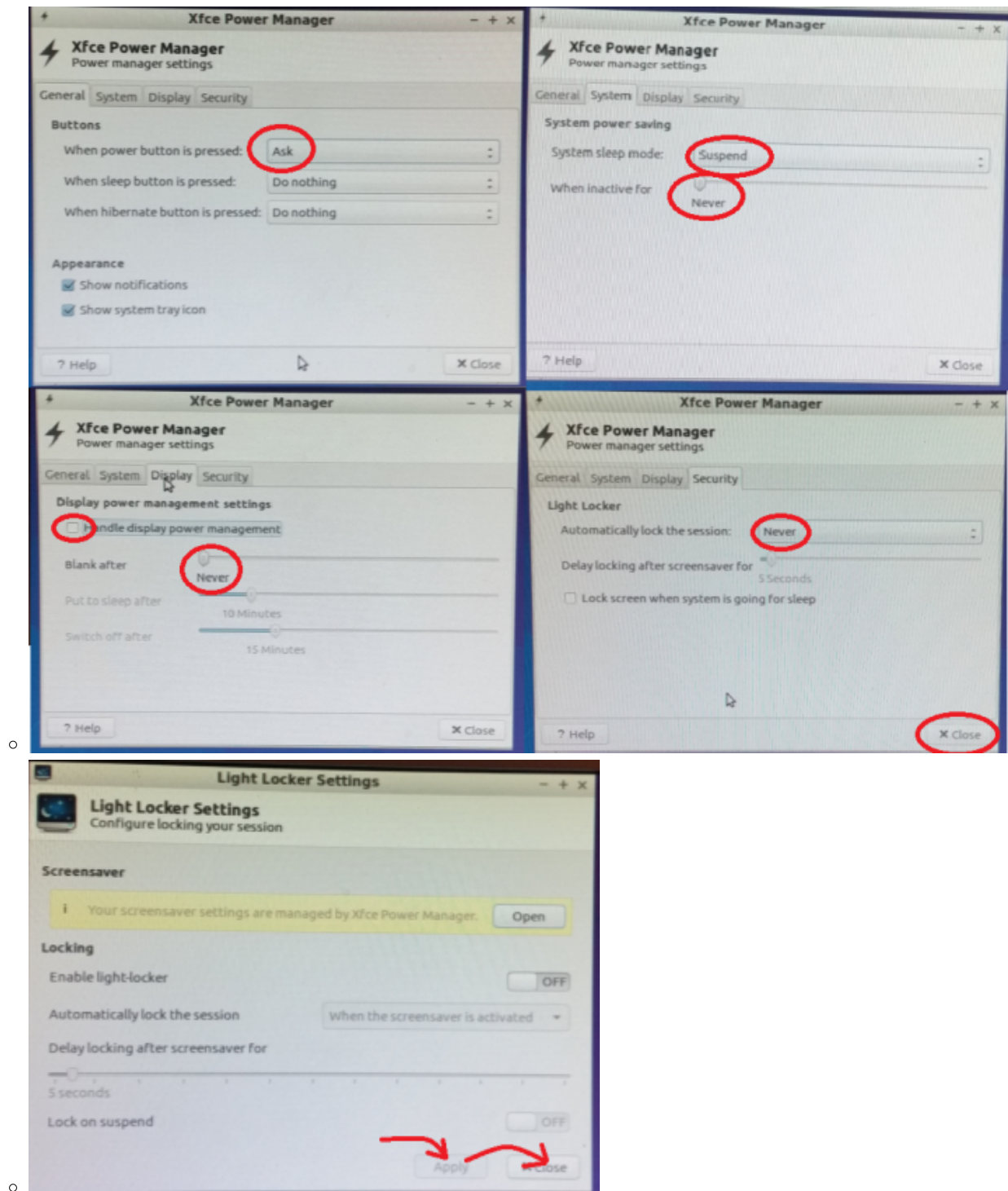
- If necessary, set a APT-proxy with "`cat > /etc/apt/apt.conf`", where the following must be entered (finish with "`Ctrl+C`")

- `Acquire::http::Proxy "http://gate-w.wettzell.ifag.de:8000";`

- Update package information
- `apt-get update`
- **Downgrade the desktop environment from “Unity” to a lightweight one, e.g. “LXDE (Lightweight X11 Desktop Environment)”**, (it is still possible to change the environment after log-out and clicking onto the Ubuntu logo over the user login)
  - with “ *sudo apt-get install lubuntu-desktop* ”
  - and set it as default environment:
    - check which environments are available with “ *ls /usr/share/xsessions/* ” and if *Lubuntu.desktop* exists and
    - edit the default settings file with “ *vi /usr/share/lightdm/lightdm.conf.d/50-lubuntu.conf* ” as root and change it to
      - `[SeatDefaults]`  
`user-session=Lubuntu`
- Reboot
- (Maybe it is necessary to change “update-apt-xapi”-settings, which updates the software database regularly and takes a lot of CPU time)
- **Set all parameters in the screensaver** in the menu **Start menu → Preferences → Light Locker Settings** and follow the instruction in the images below

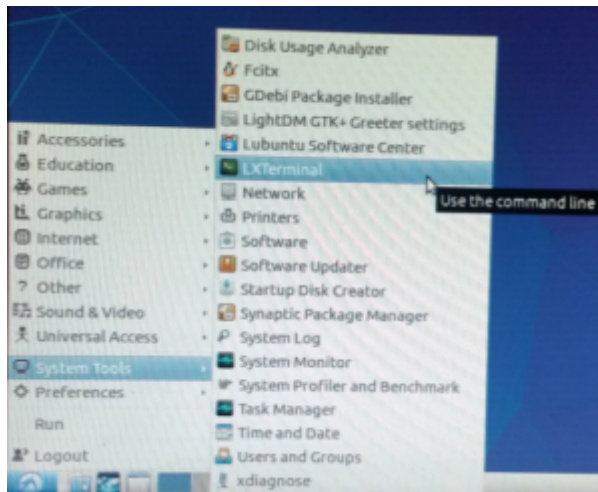




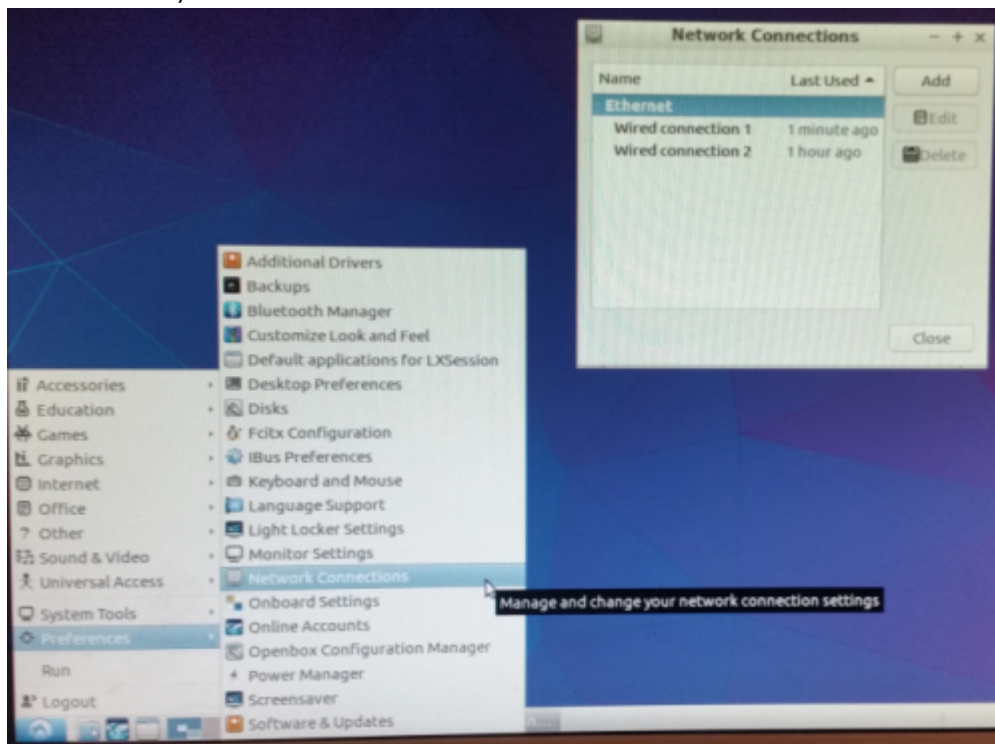


- **Set the network**

- Open a terminal ("LXTerminal") using the start menu

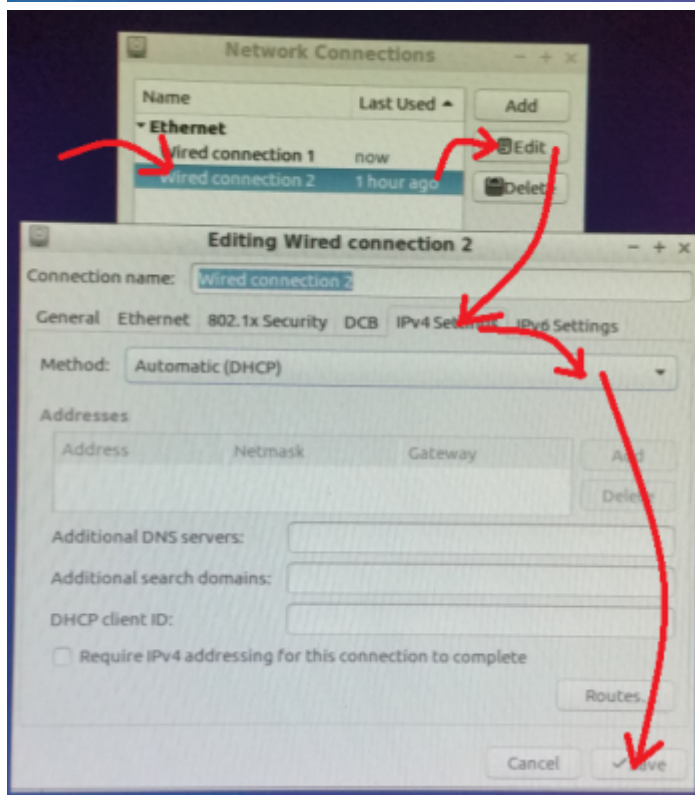
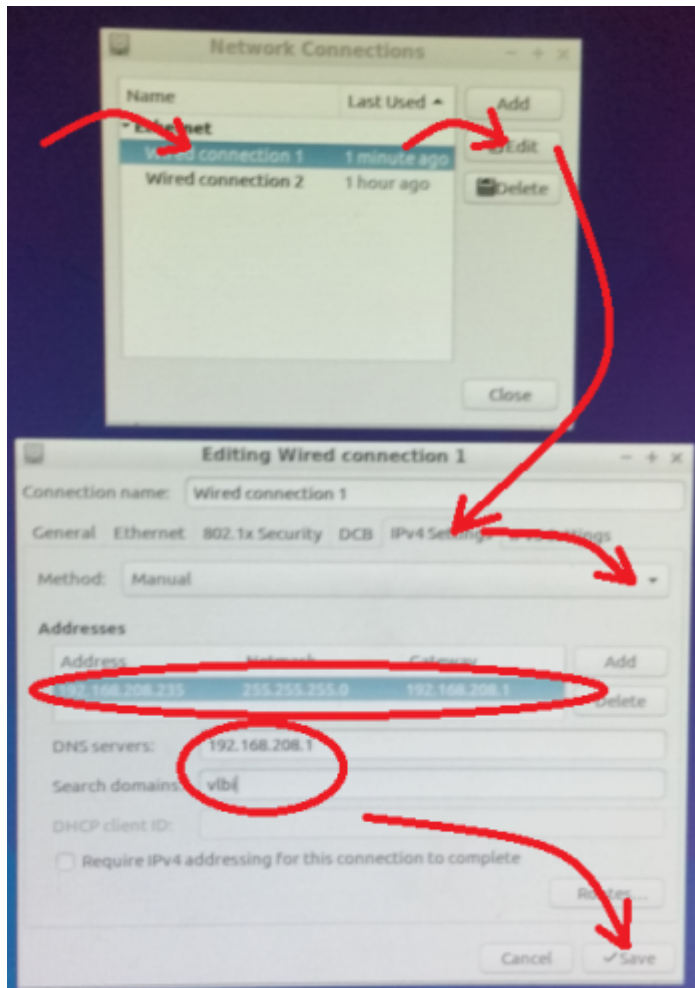


- 
- Become root with "`sudo su`"
- Set hostname if not already correct: "`vi /etc/hostname`" and set it to "vlbisyson"
- Open the "Network Connections" dialog (under the LXDE (Lightweight X11 Desktop Environment) it is in the **Start menu** → **Preferences** → **Network Connections**)

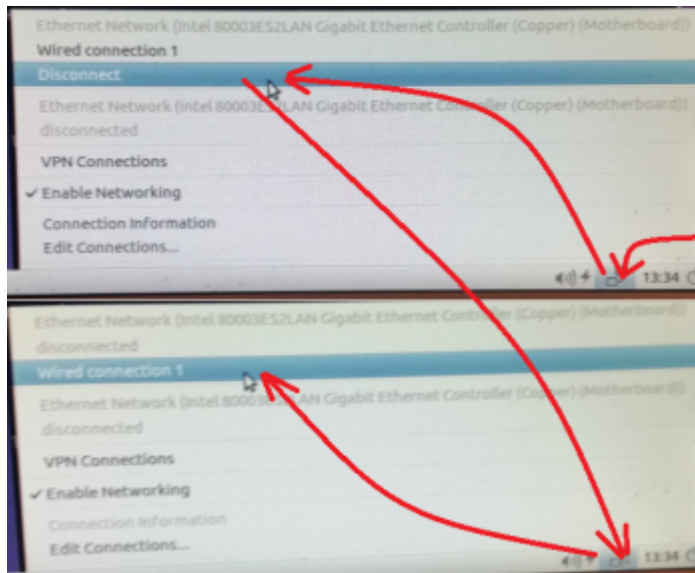


- 
- Each server has two network interfaces. The first one gets a static IP setting with an fixed IP from the IP-table (see [IP-addresses of the "vlbi" network](#)) and the second gets a DHCP setting (this can be let as it is in the standard installation)





- Disconnect the wired connection and connect again



- Check the correct settings using *"ifconfig"* in a terminal

```

root@vlbisyson: /home/oper
File Edit Tabs Help
TX packets:5949 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:594678 (594.6 KB) TX bytes:594678 (594.6 KB)

root@vlbisyson:/home/oper# ifconfig
enp5s0f0 Link encap:Ethernet HWaddr 00:30:c6:62:d6
inet addr:192.168.208.235 Bcast:192.168.208.255 Mask:255.255.255.0
inet6 addr: fe80::c9c8:6eb8:204a:1897/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:253278 errors:0 dropped:0 overruns:0 frame:0
TX packets:126280 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:349824597 (349.8 MB) TX bytes:8961190 (8.9 MB)
Interrupt:27 Memory:d8020000-d8040000

```

- **Set clock to UTC (GMT+0)**

- First set the time and timezone in the desktop with **Start menu → System tools → Time and Date**
- Set hardware clock to UTC (as user "root"): *"vi /etc/default/rcS"* and set line *"UTC=yes"*
- Run *"timedatectl set-local-rtc 0"*
- Set localtime to GMT+0: *"rm /etc/localtime"* and *"ln -s /usr/share/zoneinfo/Etc/GMT+0 /etc/localtime"*
- Check it with *"timedatectl"*. You should see something like this:

```

root@vlbisyson:/home/oper# timedatectl
Local time: Do 2017-04-13 12:04:29 GMT
Universal time: Do 2017-04-13 12:04:29 UTC
RTC time: Do 2017-04-13 12:04:29
Time zone: Etc/GMT+0 (GMT, +0000)
Network time on: yes
NTP synchronized: yes
RTC in local TZ: no

```

- **Activate NTP**

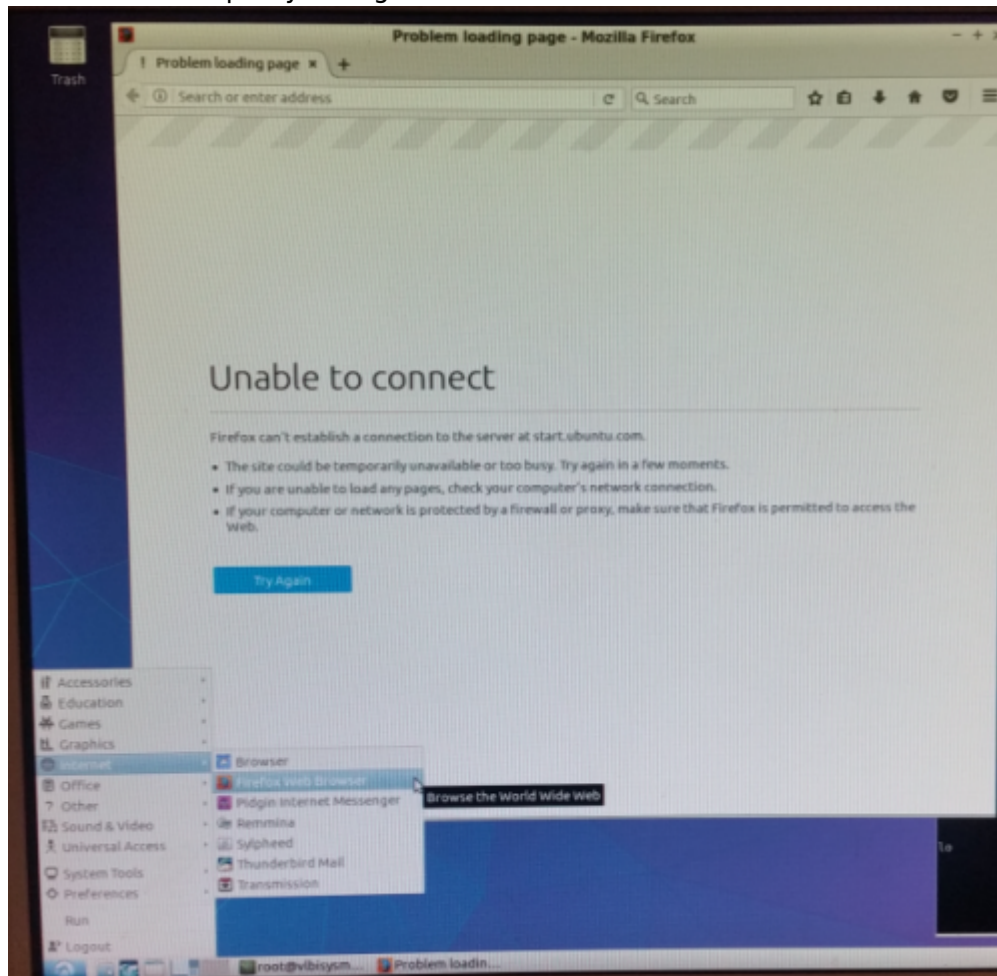
- *"apt-get install ntp"*
- *"apt-get install ntpdate"*
- Set local NTP servers for *"ntpdate"*: *"vi /etc/default/ntpdate"*
  - Set line *"NTPSERVERS= '192.168.208.4 192.168.208.5'"* (delete the existing NTPSERVERS line)
- Set local NTP servers for *"ntpd"*: *"vi /etc/ntp.conf"*
  - Set line *"server 192.168.208.4"*
  - Set line *"server 192.168.208.5"*
  - Set all existing server lines as comments (starting '#')
  - Set all existing pool lines as comments (starting '#')
- Set current time once
  - *" /etc/init.d/ntp stop"*

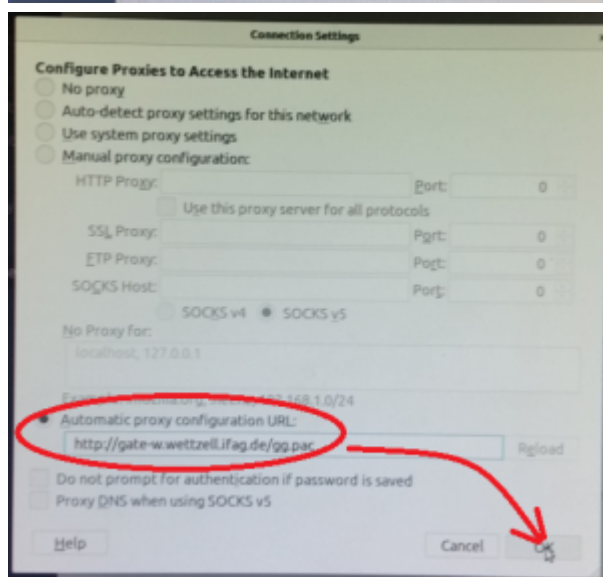
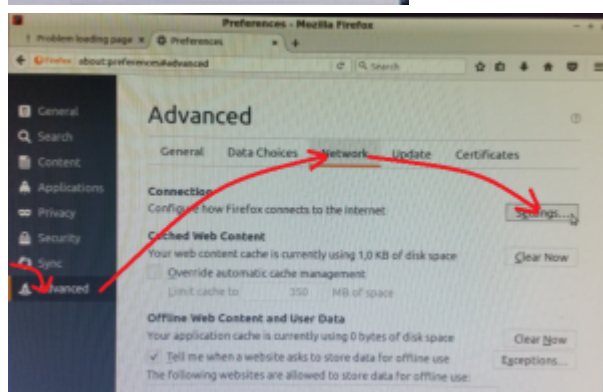
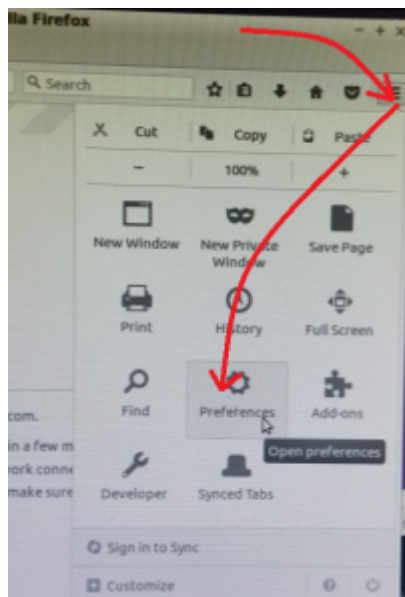
- `"ntpdate -s 192.168.208.4"`
- `" /etc/init.d/ntp start"`
- Check NTP status
  - `" ntpq -p"`

## 6) Costomize Linux software for system monitoring

### 6.1) Firefox browser

- Add "Automatic proxy configuration URL" in the Firefox internet browser





- If another browser should also be used, do the same setting there.

## 6.2) SSH server

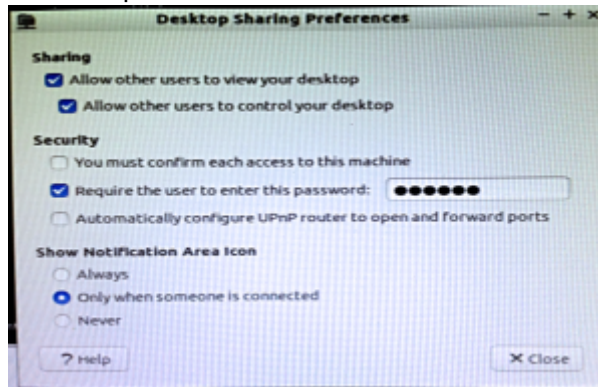
- Install a SSH server with `apt-get install ssh` (or as minimum `apt-get install openssh-server`)
- **Hint: Getting X11 forwarding through ssh working after running su**
  - Run `xauth list $DISPLAY` to get the cookie of the SSH connection, e.g. `somehost.somedomain:10 mit-magic-cookie-1 4d22408a71a55b41ccd1657d377923ae`
  - Change user with `sudo su`



- Run `"xauth add «<cookie» ", e.g. "xauth add somehost.somedomain:10 mit-magic-cookie-1 4d22408a71a55b41ccd1657d377923ae "` to add the forwarding cookie to the new user

### 6.3) Vino VNC server

- Configure the "Desktop Sharing Preferences" by calling `"vino-preferences"` as user `"oper"` (define a VNC password: here `"+oper!"`)



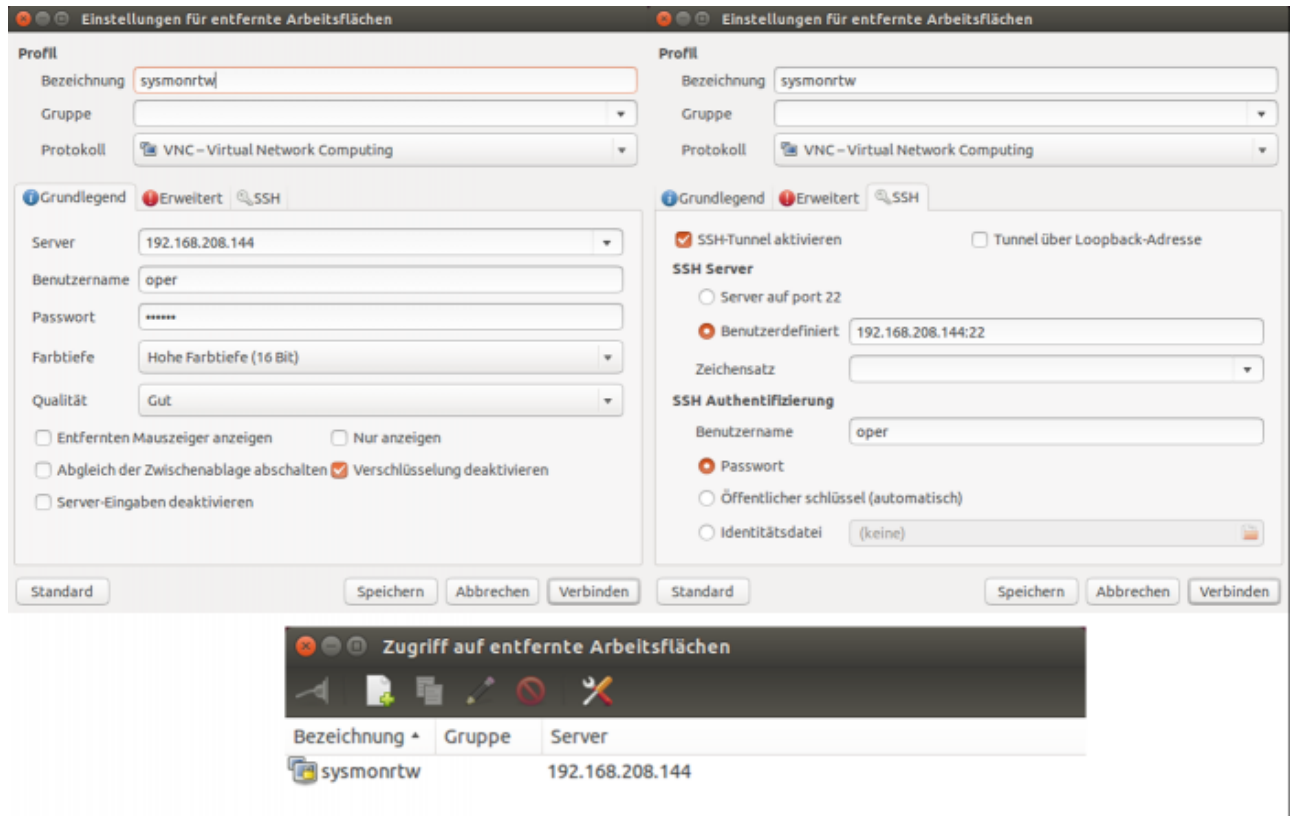
- Disable encryption, to easily allow the access with all VNC clients, with `"gsettings set org.gnome.Vino require-encryption false"`
- Create a new directory (if not yet available) as user `"oper"`: `"mkdir /home/oper/Software "` and `"mkdir /home/oper/Software/vino_vnc "`
- Change into the new directory with `"cd /home/oper/Software/vino_vnc "`
- Create a start script `"vinovnc.sh"` with an editor in the new folder and add the following content:

- ```
#!/bin/bash
/usr/lib/vino/vino-server > /dev/null 2> /dev/null &
```

- Change the access rights of the new script with `"chmod 744 ./vinovnc.sh "`
- Create a desktop starter file with `"vi /home/oper/.config/autostart/vinovnc.desktop "` and add the following context (it can also be created with the program `"lxshortcut"`):

- ```
[Desktop Entry]
Type=Application
Name=Vino VNC server
Comment=Automatic start of the VINO VNC server
Exec=/home/oper/Software/vino_vnc/vinovnc.sh
Terminal=false
```

- Test the automatic start: log-out and -on again, which should start the application (test it with `"ps ax | grep vino"`)
- The VNC Ports are: **5800** and **5900**
- An example configuration of a remote VNC client can look like the following setup for the Ubuntu `"Remmina Remote Desktop Client"` (similar settings can also be used for other VNC clients, like `"Real VNC"` under windows or `xvnc4viewer` under Linux; just if a tunneling is required, it must be set manually, using a separate SSH client)



## 6.4) Editor geany

- Install geany using the command *"apt-get install geany"*

## 6.5) GNU g++ compiler

- Install g++ using the command *"apt-get install g++"*

## 6.6) Subversion

- Install Subversion as root with *"apt-get install subversion"*

## 6.7) PostgreSQL 9.5

- *"apt-get install postgresql-9.5"*
- The PostgreSQL database is then at *" /var/lib/postgresql/9.5/main"*
- The PostgreSQL configuration is then at *" /etc/postgresql/9.5/main/postgresql.conf"* (to find the current location of the configuration file use *"ps ax | grep postgres"*, which prints the complete calling arguments of the server including the *"config\_file"* parameter, e.g. *" /usr/lib/postgresql/9.5/bin/postgres -D /var/lib/postgresql/9.5/main -c config\_file=/etc/postgresql/9.5/main/postgresql.conf"*)
- Enable remote access
  - *"vi /etc/postgresql/9.5/main/postgresql.conf"* and enable *"listen\_addresses = 'localhost'"* and *"port = 5432"*
  - *"vi /etc/postgresql/9.5/main/pg\_hba.conf"* and enable *"host all all 127.0.0.1/32 trust"*

```

◦ # Database administrative login by UNIX sockets
local all postgres trust
TYPE DATABASE USER CIDR-ADDRESS METHOD
"local" is for Unix domain socket connections only
local all all trust
IPv4 local connections:
host all all 127.0.0.1/32 trust
IPv6 local connections:
host all all ::1/128 trust
Zabbix database access
local zabbix zabbix md5

```

- Restart PostgreSQL with “ `/etc/init.d/postgresql stop`” and “ `/etc/init.d/postgresql start`” (“ `/etc/init.d/postgresql-8.4 restart`” may not work correctly)
- Test the connectivity with “ `psql -h 127.0.0.1 -p 5432 postgres postgres`” (quit with Ctrl-D)
- For the programming [simple\\_psqlquery](#) can be used
- Further documentation can be found on <http://www.postgresql.org/docs/9.5/static/index.html>
- Install the PostgreSQL library for the compiler using “`apt-get install libpq-dev`”

## 6.8) Wettzell System Monitoring Software (SysMon)

- The software can be found on the Wettzell Subversion repository  
<http://xsamba.wtz/svn/vlbi/trunk/code/vlbisysmon/>
  - Create a directory “Software” in the home directory of the user oper with “`mkdir /home/oper/Software`”
  - Change into the new directory and fetch the SysMon source with the Subversion command “`svn co http://xsamba.wtz/svn/vlbi/trunk/code/vlbisysmon/`”
  - Connect to PostgreSQL using “ `psql -h 127.0.0.1 -p 5432 postgres postgres`” (quit with Ctrl-D)
  - Create role and database:
    - “`CREATE ROLE sysmon ENCRYPTED PASSWORD '+sysmon!' SUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT LOGIN CONNECTION LIMIT 100;`”
    - “`CREATE DATABASE sysmon WITH OWNER=sysmon;`”
  - Test the connectivity to the new database with “ `psql -h 127.0.0.1 -p 5432 sysmon sysmon`” (quit with Ctrl-D)
  - Change into directory of Wettzell SysMon software and build the individual components which you want to use
- ```

cd /home/oper/Software/vlbisysmon/main/sysmon_sender/make
make build
cd /home/oper/Software/vlbisysmon/main/sysmon_backup/make
make build

```

6.8) Apache web server

- Install Apache2 as root with “`apt-get install apache2`”

6.9) PHP

- `"apt-get install php libapache2-mod-php php-mcrypt"`

6.10) automake

- `"apt-get install automake"`

6.11) Zabbix

- A basic manual can be found here: <https://www.zabbix.com/documentation/2.2/manual>
- Install the Zabbix software using
 - `"apt-get install zabbix-server-pgsql "`
 - `"apt-get install zabbix-agent "`
 - `"apt-get install zabbix-frontend-php "`
- Configure the server with `"geany /etc/zabbix/zabbix_server.conf "`

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=zabbix
```

- Create the zabbix database after connecting with `"psql -h 127.0.0.1 -p 5432 postgres postgres"` (quit with Ctrl-D)

```
CREATE USER zabbix WITH PASSWORD 'zabbix';
CREATE DATABASE zabbix OWNER zabbix;
```

- Download the Zabbix sources which fit to the Zabbix installation of the operating system: e.g. for Ubuntu 16.04. LTS it is Zabbix 2.4.7 (to check, start `"zabbix_server"` with `"DebugLevel=3"` in the configuration file `" /etc/zabbix/zabbix_server.conf "` and read the log file at `" /var/log/zabbix-server/zabbix_server.log "`, which is also defined in the configuration file of the server):

- `zabbix_3.2.4.orig.tar.gz`
- or download from <http://www.zabbix.com/download.php> to the directory `/home/oper/Software/` and extract the package with `"tar -zxvf zabbix_3.2.4.orig.tar.gz"`

Package	Distribution	Version	Architecture	Download	Documentation
Zabbix 3.2	Red Hat Enterprise Linux CentOS Oracle Linux	7	x86_64	Download	
		6	i386	Download	
		5	x86_64	Download	
			i386	Download	
	Debian	7 (Wheezy), 8 (Jessie)	i386	Download	
			amd64	Download	
	Ubuntu	14.04 LTS (Trusty), 16.04 (Xenial Xerus)	i386	Download	
			amd64	Download	

- Change into folder `/home/oper/Software/zabbix-3.2.4/database/postgresql` and run (see https://www.zabbix.com/documentation/3.2/manual/appendix/install/db_scripts)
 - ```
psql -h 127.0.0.1 -p 5432 -U zabbix zabbix < schema.sql
stop here if you are creating database for Zabbix proxy
psql -h 127.0.0.1 -p 5432 -U zabbix zabbix < images.sql
psql -h 127.0.0.1 -p 5432 -U zabbix zabbix < data.sql
```
- Restart Zabbix server process
  - `" /etc/init.d/zabbix-server stop "`
  - `" /etc/init.d/zabbix-server start "`
- Configure PHP with `"geany /etc/php/7.0/apache2/php.ini "` and restart the Apache2 server with `" /etc/init.d/apache2 stop "` and `" /etc/init.d/apache2 start "`
  - ```
[Date]
; Defines the default timezone used by the date functions
date.timezone = Europe/Berlin
max_execution_time = 600
post_max_size = 32M
memory_limit = 256M
mbstring.func_overload = 0
upload_max_filesize = 16M
max_input_time = 600
```
- Create Web front-end as root
 - `" cd /var/www "`
 - `" mv /var/www/html/ /var/www/html_original "`
 - `" chown -R www-data:www-data /var/www/html_original "`
 - `" mkdir html "`
 - `" cp -R /home/oper/Software/zabbix-3.2.4/frontend/php/* ./html/ "`
 - `" chown -R www-data:www-data /var/www/html "`
 - Restart the Apache2 server with `" /etc/init.d/apache2 stop "` and `" /etc/init.d/apache2 start "`
 - Open a browser and connect to <http://127.0.0.1> and follow the instructions (if the configuration file cannot be saved automatically, then download it and save it at `/var/www/html/conf/`).




[Welcome](#)
[Check of pre-requisites](#)
[Configure DB connection](#)
[Zabbix server details](#)
[Pre-installation summary](#)
[Install](#)

Welcome to Zabbix 3.2

[Back](#) [Next step](#)

Licensed under [GPL v2](#)

Zabbix 3.2.4. © 2001–2017, [Zabbix SIA](#)



[Welcome](#)
[Check of pre-requisites](#)
[Configure DB connection](#)
[Zabbix server details](#)
[Pre-installation summary](#)
[Install](#)


Check of pre-requisites

	Current value	Required	
PHP version	7.0.15-Ubuntu0.10.04.4	5.4.0	OK
PHP option "memory_limit"	256M	128M	OK
PHP option "post_max_size"	32M	16M	OK
PHP option "upload_max_filesize"	16M	2M	OK
PHP option "max_execution_time"	600	300	OK
PHP option "max_input_time"	600	300	OK
PHP option "date.timezone"	Europe/Berlin		OK
PHP databases support	PostgreSQL		OK
PHP bcmath	on		OK
PHP mbstring	on		OK
PHP option "mbstring.func_overload"	off	off	OK

[Back](#) [Next step](#)

Licensed under [GPL v2](#)

Zabbix 3.2.4. © 2001–2017, [Zabbix SIA](#)



Configure DB connection

Please create database manually, and set the configuration parameters for connection to this database. Press "Next step" button when done.

Database type

Database host

Database port 0 - use default port


Database name

User

Password

Licensed under [GPL v2](#)

Zabbix 3.2.4. © 2001–2017, [Zabbix SIA](#)



Zabbix server details

Please enter the host name or host IP address and port number of the Zabbix server, as well as the name of the installation (optional).

Host

Port

Name

Licensed under [GPL v2](#)

Zabbix 3.2.4. © 2001–2017, [Zabbix SIA](#)

ZABBIX

Welcome

Check of pre-requisites

Configure DB connection

Zabbix server details

Pre-installation summary

Install

Pre-installation summary

Please check configuration parameters. If all is correct, press "Next step" button, or "Back" button to change configuration parameters.

Database type	PostgreSQL
Database server	localhost
Database port	default
Database name	zabbix
Database user	zabbix
Database password	*****
Database schema	
Zabbix server	localhost
Zabbix server port	10051
Zabbix server name	

Back

Next step

Licensed under [GPL v2](#)

Zabbix 3.2.4. © 2001–2017, Zabbix SIA

ZABBIX

Welcome

Check of pre-requisites

Configure DB connection

Zabbix server details

Pre-installation summary

Install

Install

Congratulations! You have successfully installed Zabbix frontend.

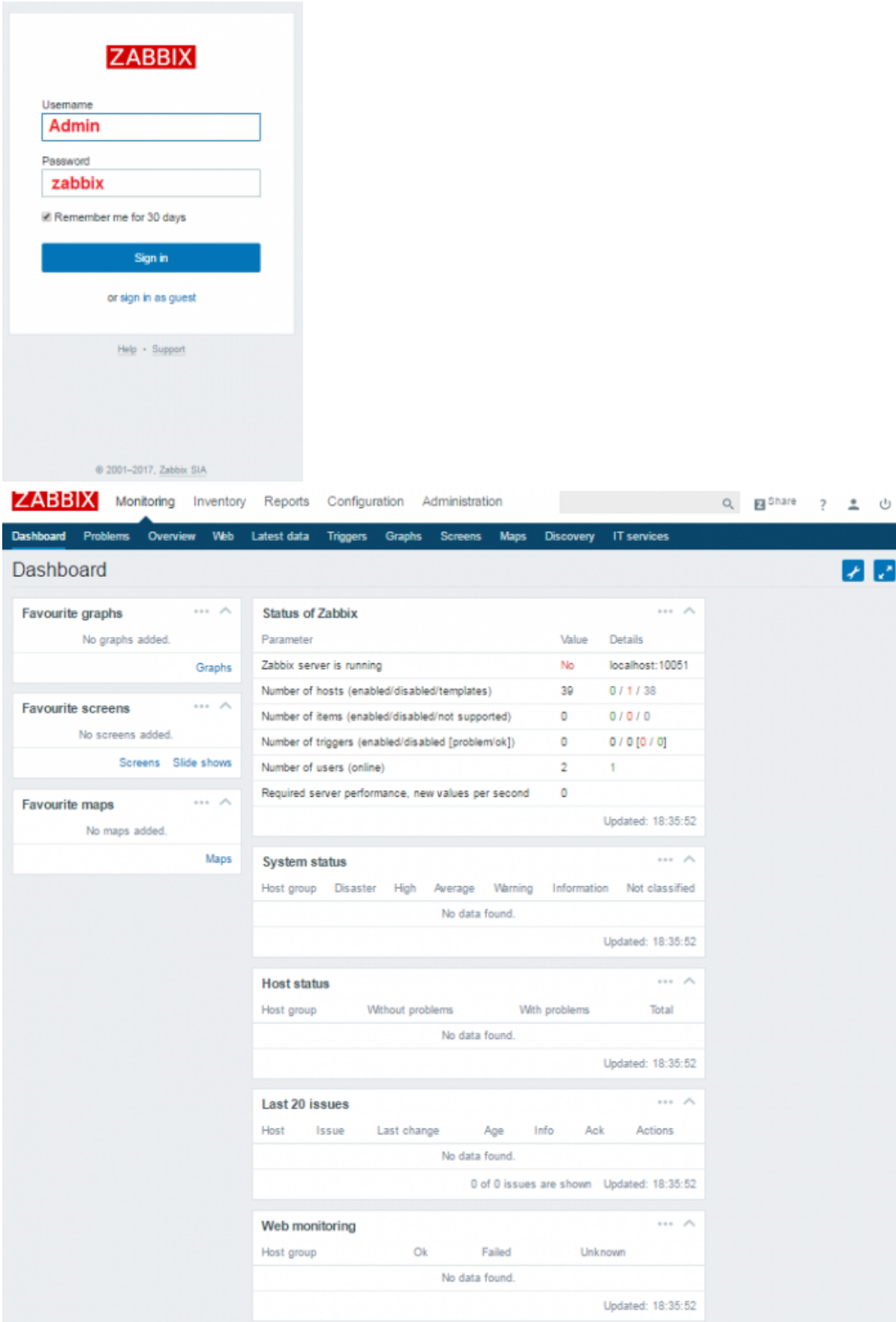
Configuration file "/var/www/html/conf/zabbix.conf.php" created.

Back

Finish

Licensed under [GPL v2](#)

Zabbix 3.2.4. © 2001–2017, Zabbix SIA



The image shows two screenshots of the Zabbix web interface. The top screenshot is the login page, featuring the Zabbix logo, a 'Sign in' button, and a 'Remember me for 30 days' checkbox. The bottom screenshot is the Zabbix dashboard, displaying various monitoring metrics and system status.

Dashboard Overview:

- Favourite graphs:** No graphs added.
- Favourite screens:** No screens added.
- Favourite maps:** No maps added.
- Status of Zabbix:**

Parameter	Value	Details
Zabbix server is running	No	localhost:10051
Number of hosts (enabled/disabled/templates)	39	0 / 1 / 38
Number of items (enabled/disabled/not supported)	0	0 / 0 / 0
Number of triggers (enabled/disabled [problem/ok])	0	0 / 0 [0 / 0]
Number of users (online)	2	1
Required server performance, new values per second	0	
- System status:**

Host group	Disaster	High	Average	Warning	Information	Not classified
No data found.						
- Host status:**

Host group	Without problems	With problems	Total
No data found.			
- Last 20 issues:**

Host	Issue	Last change	Age	Info	Ack	Actions
No data found.						
- Web monitoring:**

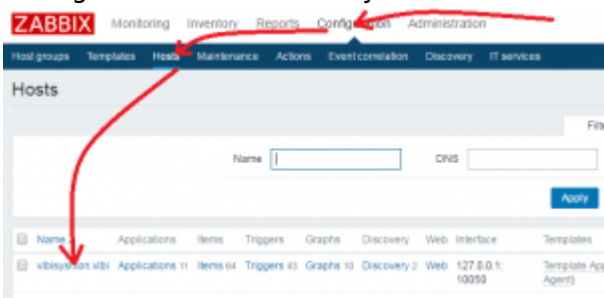
Host group	Ok	Failed	Unknown
No data found.			

- Prepare manual Zabbix installation

- `sudo apt-get install libsnmp-dev`

- Update the "zabbix_server" and "zabbix_agent" to the latest version
 - Change into directory " /home/oper/Software/zabbix-3.2.4/ "

- Run a configuration
 - `./configure --enable-server --enable-agent --with-postgresql --with-net-snmp`
- Build the server and agent
- `make`
- Copy server and agent to /usr/sbin
- ```
mv /usr/sbin/zabbix_server /usr/sbin/zabbix_server_2.4.7
cp /home/oper/Software/zabbix-3.2.4/src/zabbix_server/zabbix_server /usr/sbin/zabbix_server
mv /usr/sbin/zabbix_agentd /usr/sbin/zabbix_agentd_2.4.7
cp /home/oper/Software/zabbix-3.2.4/src/zabbix_agent/zabbix_agentd /usr/sbin/zabbix_agentd
```
- Create a soft-link to the original configuration files
- ```
ln -s /etc/zabbix/zabbix_server.conf /usr/local/etc/zabbix_server.conf
ln -s /etc/zabbix/zabbix_agentd.conf /usr/local/etc/zabbix_agentd.conf
```
- Stop agent and server
- ```
/etc/init.d/zabbix-server stop
/etc/init.d/zabbix-server start
/etc/init.d/zabbix-agent stop
/etc/init.d/zabbix-agent start
```
- Change server name using "geany /etc/zabbix/zabbix\_agentd.conf "
- `Hostname=vlbisyson.vlbi`
- Change hostname to "vlbisyson.vlbi" also in the Web interface



The screenshot shows the Zabbix web interface. The top navigation bar includes links for Monitoring, Inventory, Reports, Configuration, and Administration. The 'Configuration' tab is active, and the 'Hosts' sub-tab is selected. The main content area shows the configuration for a host named 'vblbiysmon.vlbi'. The host is enabled and has a status of '29K'. It is associated with 11 applications, 64 items, 43 triggers, 10 graphs, 2 discovery rules, and 0 web scenarios. The configuration form includes fields for Host name, Visible name, Groups (In groups and Other groups), New group, Agent interfaces (IP address, DNS name, Connect to, Port, Default), SNMP interfaces, JMX interfaces, IPMI interfaces, Description, Monitored by proxy, and Enabled. The 'Update' button is highlighted.

## 7) Create an HTTP file archive

- Create a directory in the web space of the already existing Apache server to store historic monitoring data there as files
- ```
mkdir /var/www/html/monitoring_archive
chown -R www-data:www-data /var/www/html/monitoring_archive
chmod -R 777 /var/www/html/monitoring_archive
```
- The structure of the archive can be individual but suggested is a folder structure in the following way: monitoring control point ID, year, month, individual day file, e.g.


```
TTW1Dewar
|-> 2017
|   |-> 01
|       |-> 20170101TTW1Dewar.txt
|       |-> 20170102TTW1Dewar.txt
|       |-> 20170103TTW1Dewar.txt
|       |-> 20170104TTW1Dewar.txt
|       |-> ...
|   |-> 02
|   |-> 03
|   |-> 04
|   |-> ...
|-> 2018
```

```
| -> 2019  
| -> ...  
Meteo  
...
```

- Each program for the individual monitoring control point must take care on the individual structure itself.

From:

<http://wiki.wtz/> - **Geodetic Observatory - Wiki**

Permanent link:

http://wiki.wtz/doku.php?id=vlbi:sysmon:000_vlbi_sysmon_node



Last update: **2017/05/12 13:37**

Install and configure the monitoring of a NASA FS PC (operational and diagnostic data)

The general data from the NASA FS PC are **operational or diagnostic data** which do not need to be saved for data VLBI analysis centers. Operational and diagnostic data are just managed in the Zabbix system and the data history is limited to one or two weeks (max. a month) to reduce data volume on the system monitoring PC.

1) Install a Zabbix agentd on the NASA FS PC

- The following description is just for NASA FS PCs which are in the same network as the Zabbix Server PC doing the monitoring.
- Download Zabbix sources as described in [0\) SysMon Node VLBI](#)
- Unpack sources e.g. into a folder /usr2/prog/Software/

- ```
tar -xzvf zabbix_3.2.4.orig.tar.gz
```

- Change into newly created directory, run the configuration utility and build the agent

- ```
cd /usr2/prog/Software/zabbix-3.2.4
./configure --enable-agent
make
```

- Make a safety copy of the original configuration file

- ```
cp /usr2/prog/Software/zabbix-3.2.4/config/zabbix_agentd.conf
/usr2/prog/Software/zabbix-3.2.4/config/zabbix_agentd.conf_orig
```

- Edit the configuration file and change the values to the following (Hostname should be the name for the individual system which is also used later in the Zabbix front end; the IP addresses of the server must be those from the real Zabbix server PC)

- ```
DebugLevel=3
Server=192.168.208.235
ServerActive=192.168.208.235
Hostname=fsttw1.vlbi
```

- Become root rights and do the following steps

- ```
su
cp /usr2/prog/Software/zabbix-3.2.4/conf/zabbix_agentd.conf
/usr/local/etc/zabbix_agentd.conf
cp /usr2/prog/Software/zabbix-3.2.4/src/zabbix_agent/zabbix_agentd
/usr/sbin/zabbix_agentd
```

```
groupadd zabbix
useradd -g zabbix zabbix
```

- Test the start of the Zabbix agent

- ```
/usr2/prog/Software/zabbix-3.2.4/src/zabbix_agent/zabbix_agentd -c /usr2/prog/Software/zabbix-3.2.4/conf/zabbix_agentd.conf
```

- Create a startscript in /etc/init.d/ e.g. with the name zabbix_agentd or include it to another start script. To start the zabbix_agentd it must contain this:

- ```
#!/bin/sh
BEGIN INIT INFO
Provides: zabbix_agentd
Required-Start:
Required-Stop:
Default-Start: 2 3 4 5
Default-Stop: 0 1 6
Short-Description: Start zabbix_agent for PC monitoring
Description: Start zabbix_agent for PC monitoring
END INIT INFO

case $1 in
start)
 su daemon -c /usr/bin/zabbix_agentd
 ;;
stop)
 kill `cat /tmp/zabbix_agentd.pid`
 ;;
restart)
 $0 stop
 sleep 2
 $0 start
 ;;
*)
 echo "usage: $0 [start|stop|restart]"
 ;;
esac
```

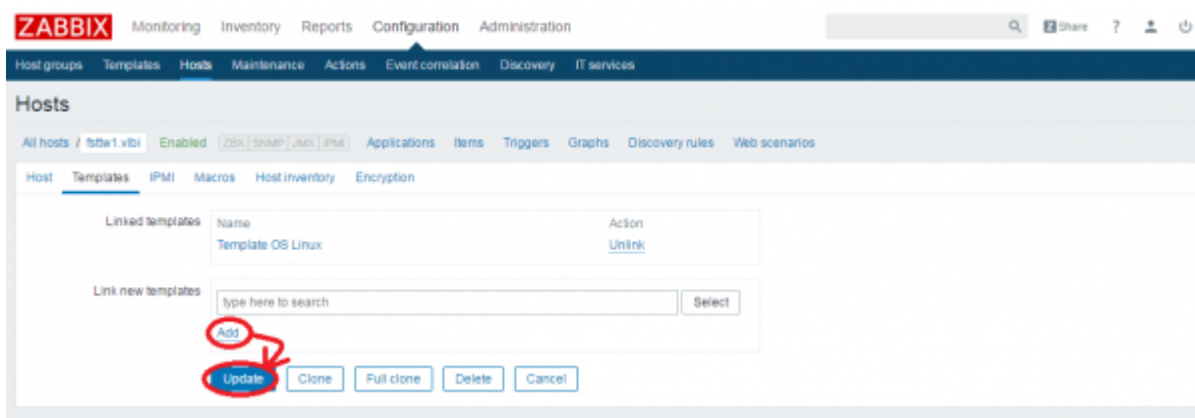
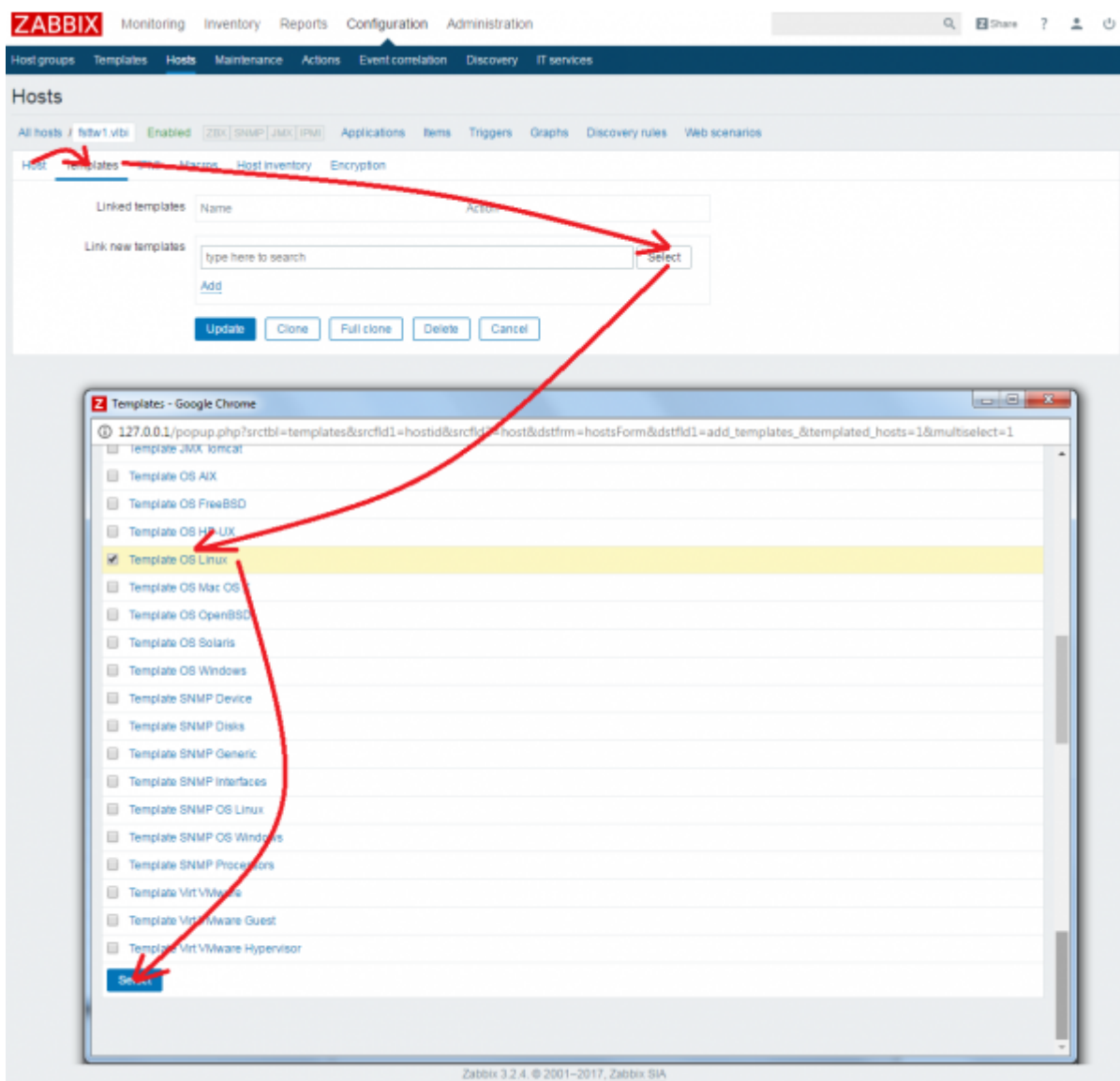
## 2) Activate monitoring on the Zabbix server

- Create new host

The top screenshot shows the Zabbix web interface at the 'Hosts' page. A red arrow points to the 'Configuration' menu in the top navigation bar. The 'Hosts' table shows one host: 'fshw1.vlbi' with IP 127.0.0.1, status 'Enabled', and agent 'Zabbix Agent'.

The bottom screenshot shows the 'Host configuration' form for 'fshw1.vlbi'. The 'New group' field is set to 'NASAFeldSystems'. The 'Agent interfaces' section shows an IP address of 192.168.208.13 on port 10050. The 'Description' field contains 'NASA Field System PC of the Radio Telescope TTW!'. The 'Enabled' checkbox is checked.

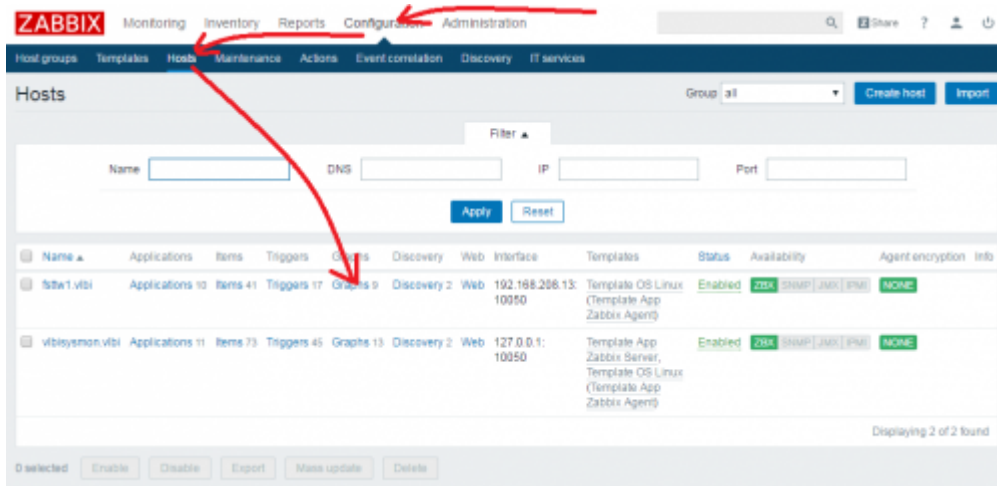
- Set templates



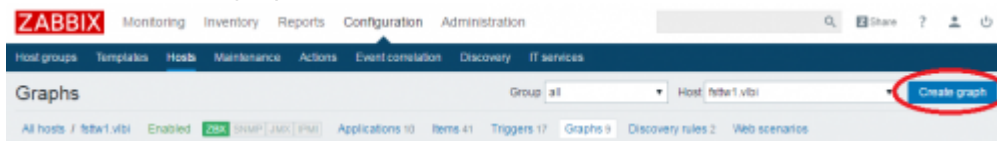
- Wait a few minutes and the collected data should appear from the FSPC

### 3) Customize the data presentation/graph for the NASA FS PC needs

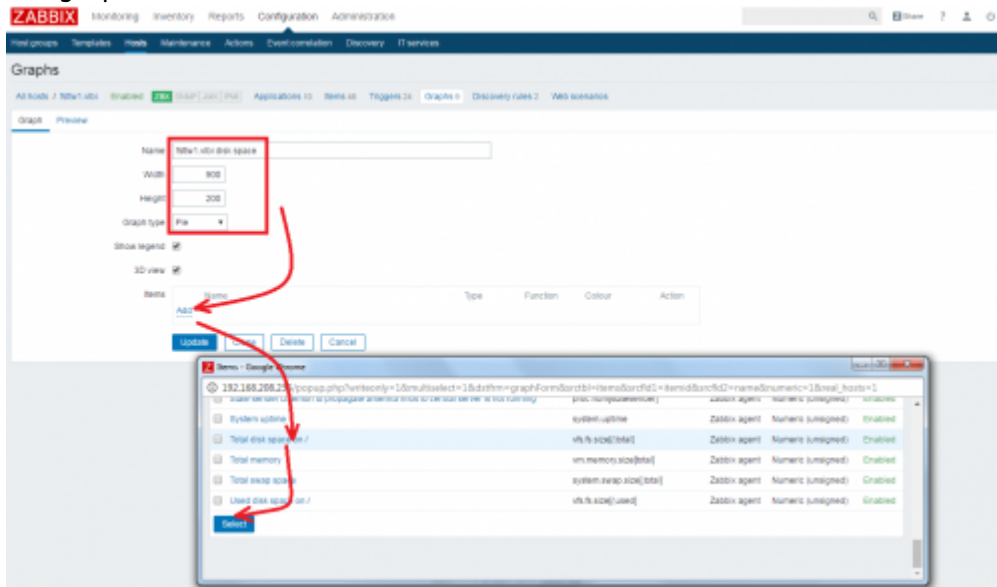
- Create new graph for disk space (which is an already existing item in the template for hosts and therefore already collected by the Zabbix agent)
- Select the host for the FS PC and "Graphs"



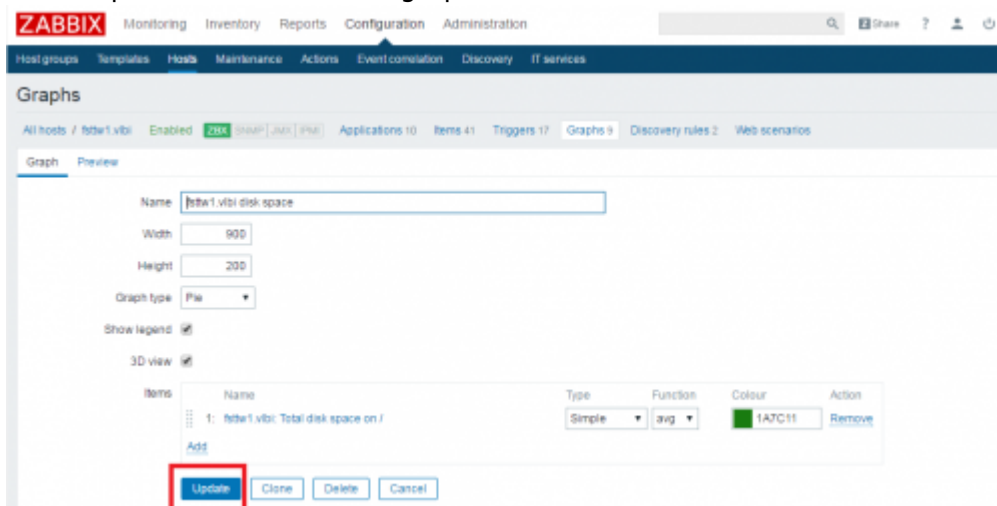
- Click on "Create graph"



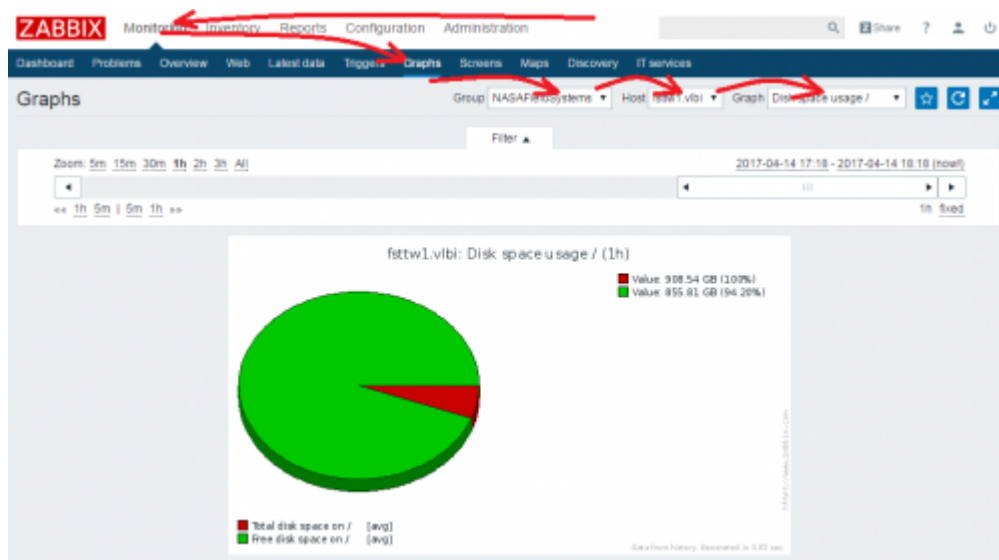
- Enter a name, the graph sizes, the graph type (here "Pie" chart) and add the item monitored in the graph



- Push "Update" to create new graph



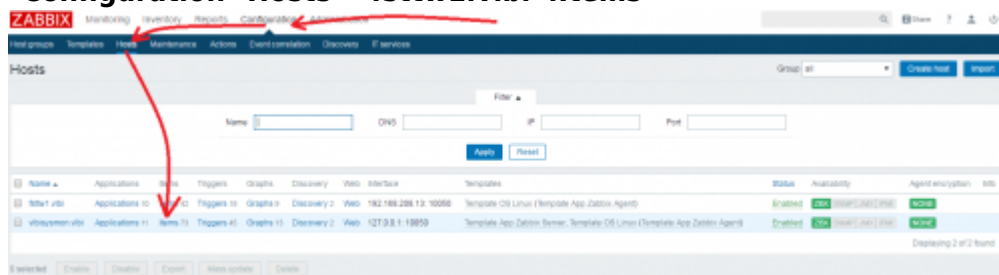
- Check the output of the new graph by selecting "Monitoring→Graphs→Group "NASAFIELDSystems"→Host "fsttw1.vlbi"→ Graph "Disk space usage /"



- This steps can now be done for all already collected monitoring data from the host “fsttw1.vlbi” or also for further computers from which data should be collected with a Zabbix agent host.

#### 4) Add additional, individual monitoring items collected by Zabbix agent

- Create a new item (= monitoring sensor value) for the host “fsttw1.vlbi” by selecting **“Configuration→Hosts→“fsttw1.vlbi”:Items”**



- Click on the button “Create item”



- One important item is the information if a process is running. Zabbix agent already supports such additional checks using specific keys. The key “proc.num” (number of process found for an individual name) is used for process checks. Enter a name for the item, e.g. to check if the Antenna Control Daemon (acud) is running, select the right key (e.g. proc.num(acud)), enter an update interval in seconds, enter the time for historic data storage and for trend storage, assign an application (e.g. Processes”), and push “Add” to create the new item for the host “fsttw1.vlbi”.

**Check if process running using the returned number of active processes with name "acud"**

- Wait a minute and check if the data arrive for the new item using **"Monitoring→Select→Hosts "fsttw1.vlbi"→Processes"** and check the column "Last value", which should show the number of processes found for the individual process name (for regular processes you should see 1 if it is running or 0 if it is not running; for "idl2rpc.pl" processes using the Wettzell Communication Middleware you should see 2 if it is running or 0 if it is not running).

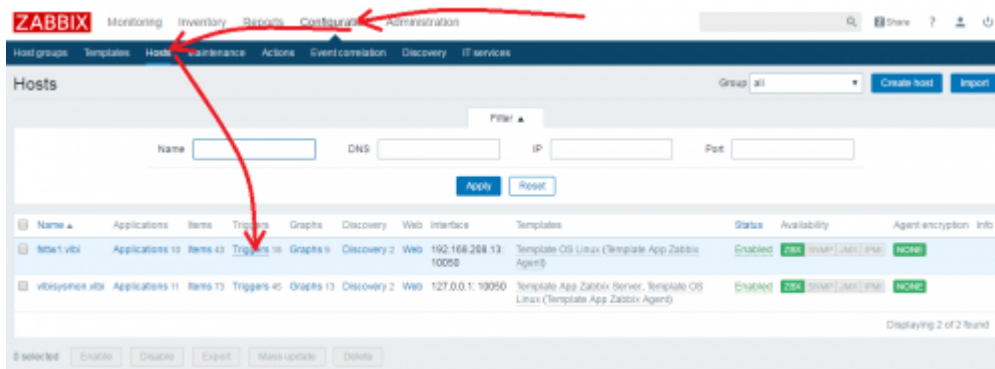
| Name                        | Last check          | Last value | Change |
|-----------------------------|---------------------|------------|--------|
| ACU Daemon is running       | 2017-04-16 00:42:47 | 2          |        |
| NASA Field System Running   | 2017-04-16 00:42:46 | 1          |        |
| Number of processes         | 2017-04-16 00:42:04 | 211        | +3     |
| Number of running processes | 2017-04-16 00:42:06 | 1          |        |

**Number of found processes**

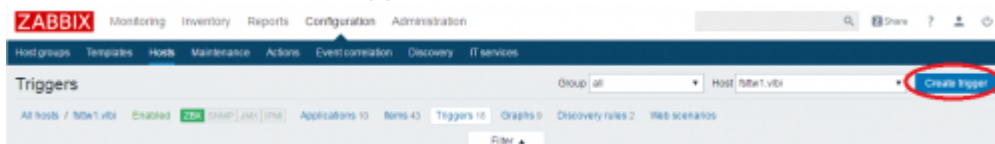
## 5) Add additional, individual trigger to detect alert levels

- Create a new trigger for alert levels for the host "fsttw1.vlbi" using **"Configuration→Hosts→"fsttw1.vlbi":Triggers"**

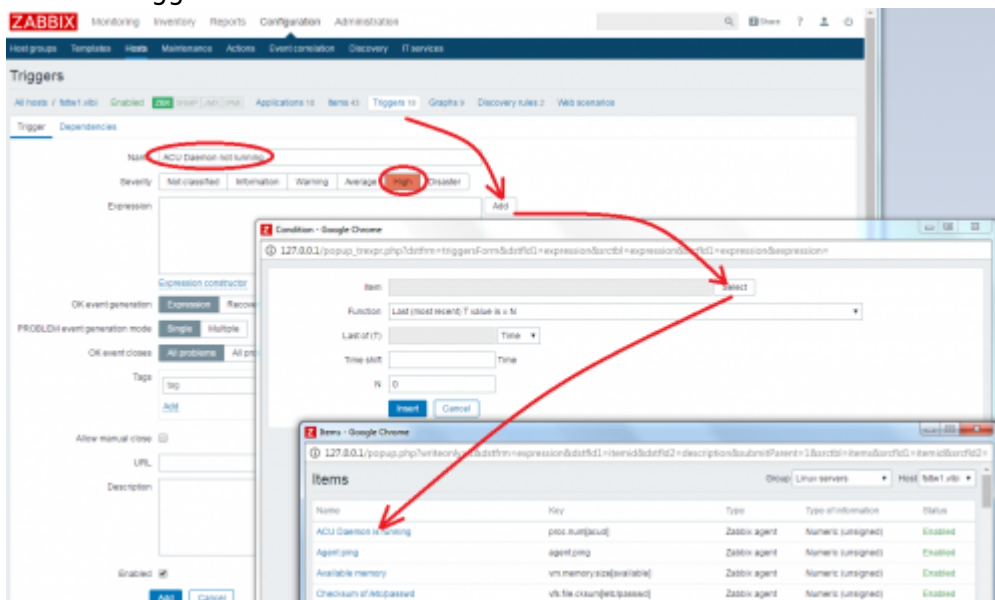




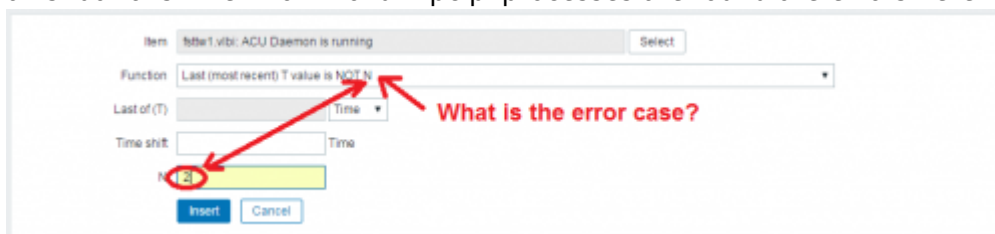
- Click the button "Create trigger"



- Enter a name for the trigger (e.g. "ACU Daemon is not running") and a severity (alarm level) and add an expression by pushing the button "Add". Select the item which should be checked with the trigger.



- Enter a function, which is the IF-statement to check which state will lead into an error state, e.g. all situations when not two id2rpc.pl processes are found are errors here.



- To test the expression, it is necessary to open the Expression constructor and to click on "Test" for the activated expression.



The top screenshot shows the Zabbix 'Triggers' configuration page. The trigger name is 'ACU Daemon not running'. The severity is set to 'High'. The expression is '{#hw1.vlib.proc.num[acud].last()}<=2'. A red arrow points to the 'Expression constructor' link.

The bottom screenshot shows the same trigger configuration page. The 'Test' button is highlighted with a red arrow.

- The expression can then be tested against different values. The result of the expression will be shown in the “result” frame.

The 'Test' dialog box shows the expression '{#hw1.vlib.proc.num[acud].last()}' and the result type 'Numeric (integer 64bit)'. The 'Value' field is set to '2'. The 'Result' section shows the expression 'A {#hw1.vlib.proc.num[acud].last()}<=2' and the result 'FALSE'. A red arrow points from the 'Value' field to the 'Result' section. A red text box on the right says 'Everything ok? Trigger is FALSE else Trigger is TRUE'.

- Maybe also click “Allow manual close”, which means that an operator can reset the error state manually, and click “Update” to create the new trigger.

**ZABBIX** Monitoring Inventory Reports Configuration Administration

Host groups Templates Hosts Maintenance Actions Event constitution Discovery IT services

**Triggers**

All hosts / vlbi1.vbi Enabled 200 10000 10000 Applications 10 Items 43 Triggers 10 Graphs 9 Discovery rules 2 Web scenarios

**Trigger** Dependencies

Name: ACU Daemon not running

Severity: Not classified Information Warning Average **High** Disaster

Expression: [vlbi1.vbi.proc.num(acud)]last()=2

And Or Replace

A

Target: Expression: A [vlbi1.vbi.proc.num(acud)]last()=2 Action: Info Remove

Close expression constructor

OK event generation: Expression Recovery expression None

PROBLEM event generation mode: Single Multiple

OK event closes: All problems All problems if tag values match

Tags: tag value Remove Add

Allow manual close: ☒

URI:

Description:

Enabled ☒ Update Close Delete Cancel

- The items are then tested and the host will show an error state if the trigger fires.

## 6) Create a screen to show all important information about the NASA FS PC

- Individual screens can be used to show system states and graphs. The most important states for the NASA FS PC are e.g. disk space, CPU load, maybe memory, network load.
- Create a new screen using **"Monitoring→Screens→Creat screen"**

**ZABBIX** Monitoring Inventory Reports Configuration Administration

Dashboard Problems Overview Web Latest data Triggers Graphs **Screens** Maps Discovery IT services

**Screens**

Screens Create screen Import

Filter

Name

Apply Reset

- Define a name for the screen and the dimensions as number of rows and columns and push "Add".

**ZABBIX** Monitoring Inventory Reports Configuration Administration

Dashboard Problems Overview Web Latest data Triggers Graphs **Screens** Maps Discovery IT services

**Screens**

Screen Sharing

Owner: Admin (Zabbix Administrator) Select

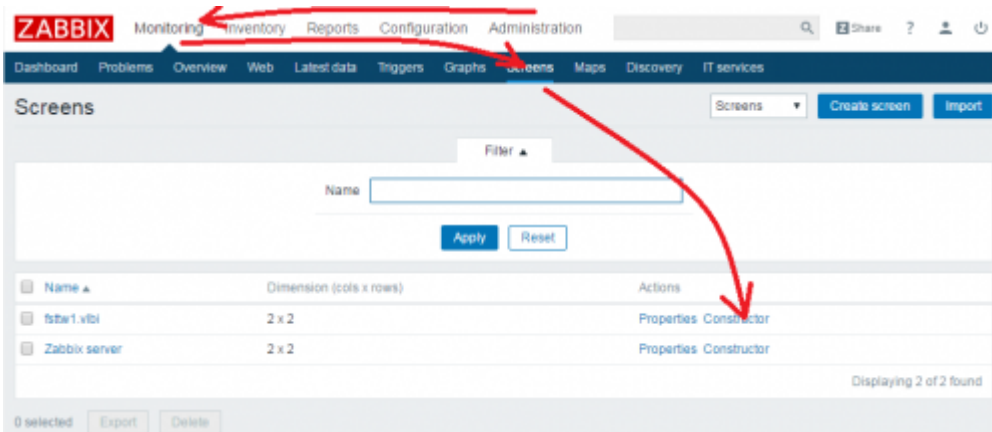
Name: vlbi1.vbi

Columns: 2

Rows: 2

Add Cancel

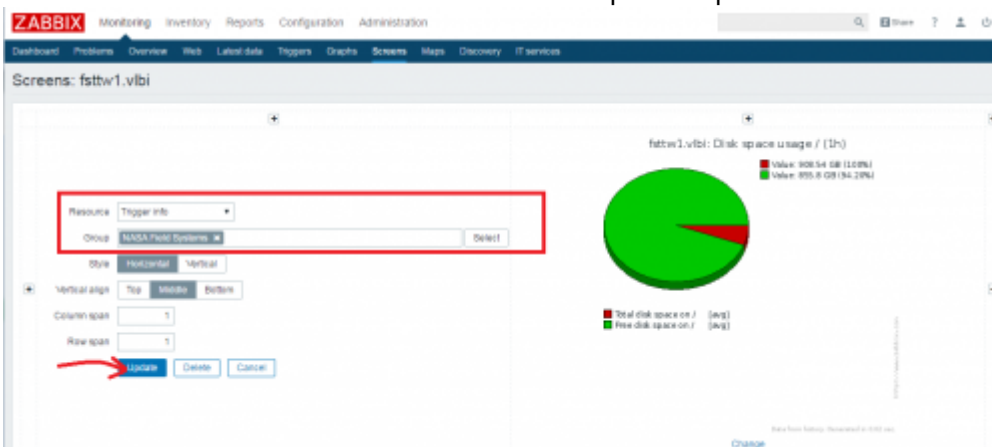
- Open the "Constructor" of the screen



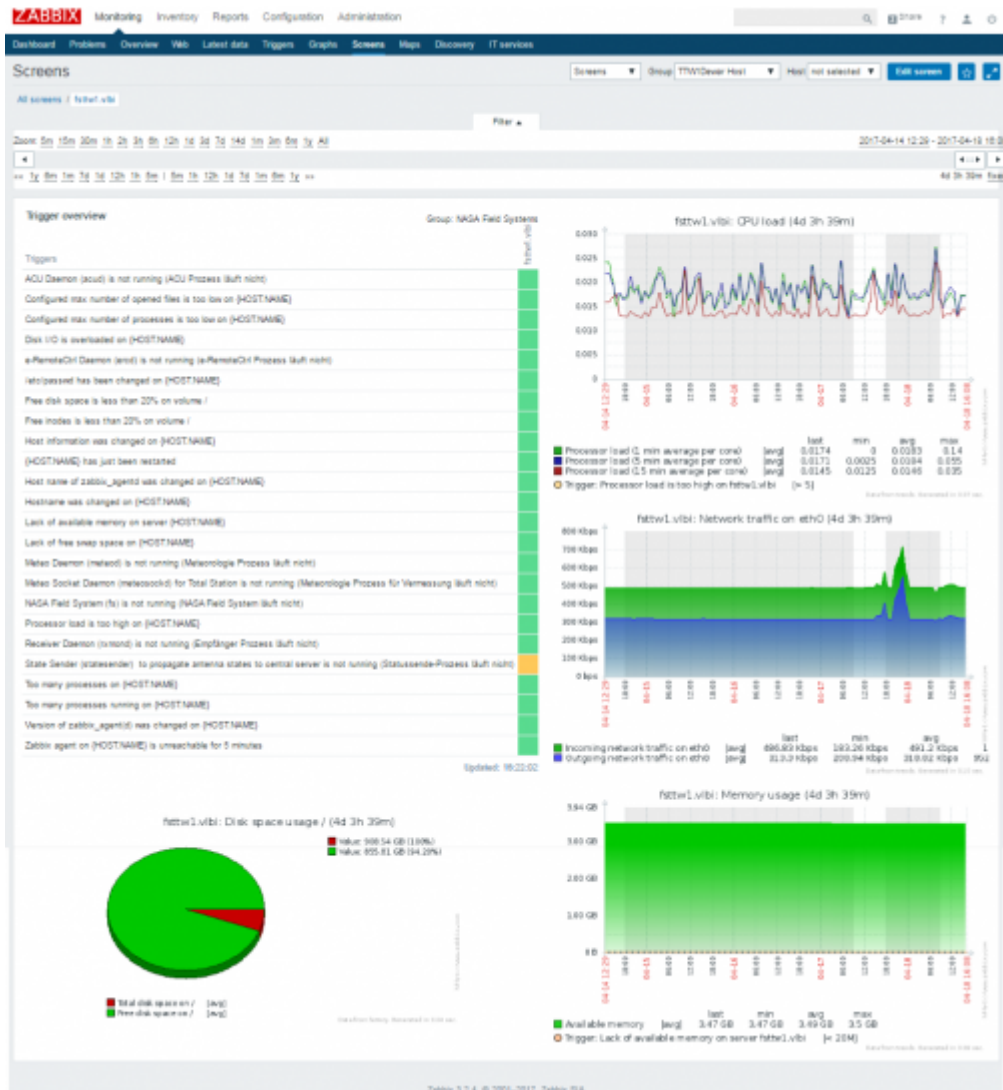
- Push the “Change” link for each of the individual fields on the screen.



- Select what should be shown in this field and push “Update” or “Add”.



- If all fields are filled with parameters, the screen can look like this:



## 7) Create an overview system map for the NASA FS PC needs

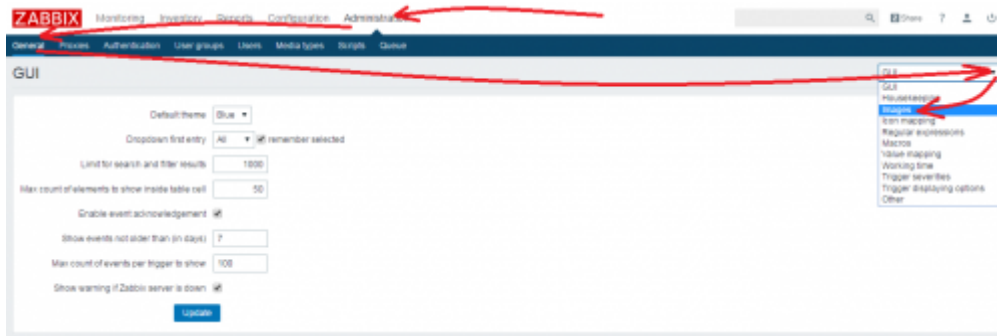
Maps are used to show the logic structure of system processes. They are a hierarchical structure for VLBI systems. The complete antenna system (e.g. TTW1) is one upper layer. It is split into sublayers of maps according to the individual parts, e.g. Control, Antenna, Infrastructure, IT, and so on. The structure can be individually.

### 7.1) Create own icons for individual processes

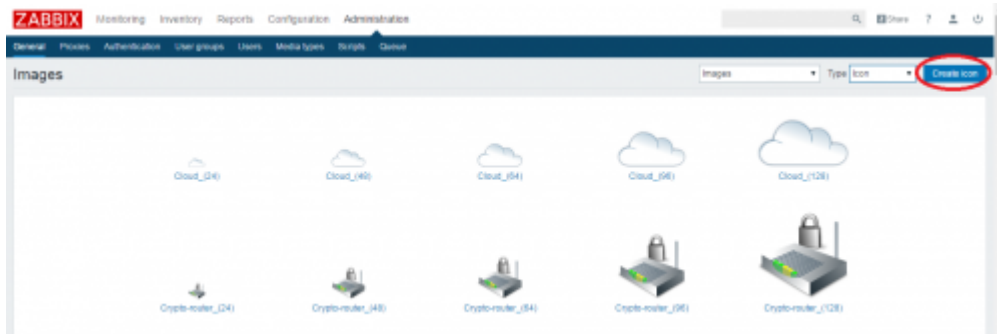
- To adapt to individual processes, individual icons for elements in the map can be created an load to Zabbix. New icons can be created with MS Powerpoint  
Sample icon in MS Powerpoint  
or graphic tools. It is important that different images (e.g. PNG format) are always created for

the following width sizes in pixel: 24, 48, 64, 96, 128 pixels. These different sizes are used to increase a highlighted icon if an alert level is fired.

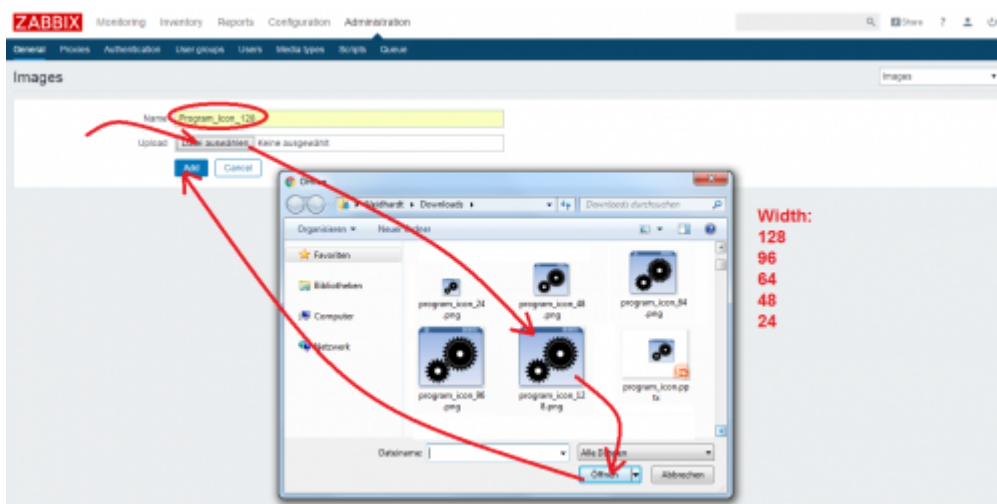
- A new icon can be added using **“Administration→general→Images”**



- Push the button “Create icon” to install a new icon set.



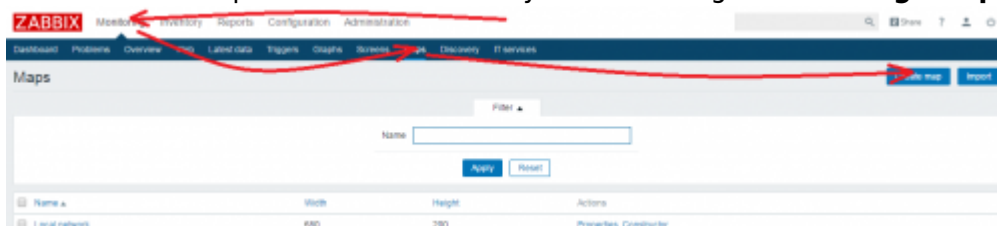
- Select the icon image file in the upload dialog and give a name for the icon, e.g. the icon type in combination with the size.



- Do this step for all individual icons

## 7.2) Create a new map for the NASA field system PC

- Create a new map for the NASA Field System PC using **“Monitoring→Maps→Create map”**



- Enter an owner. Also other users can be defined here to be owner of the map. Define a name for the map, e.g. “NASA Field System PC”, the dimensions of the map and check the checkboxes. Minimum trigger severity should be “Warning”, so that warnings and higher alerts are highlighted. Finally, press the button “Add” to create the map.

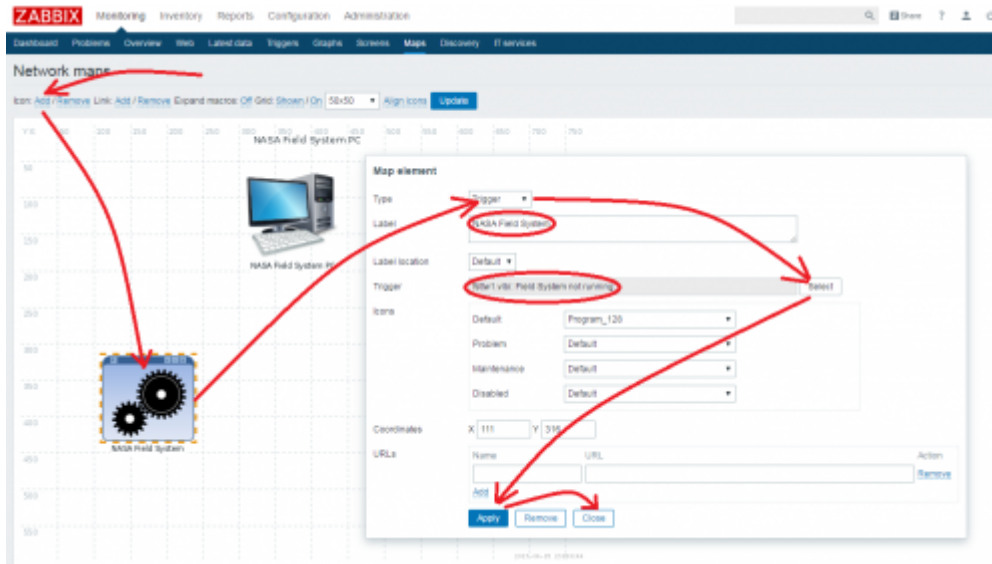
- Construct the new map to set icons and network connections clicking on the "Constructor" of the the newly created map.

| Name                 | Width | Height | Actions                |
|----------------------|-------|--------|------------------------|
| Local network        | 580   | 200    | Properties Constructor |
| NASA Field System PC | 800   | 600    | Properties Constructor |

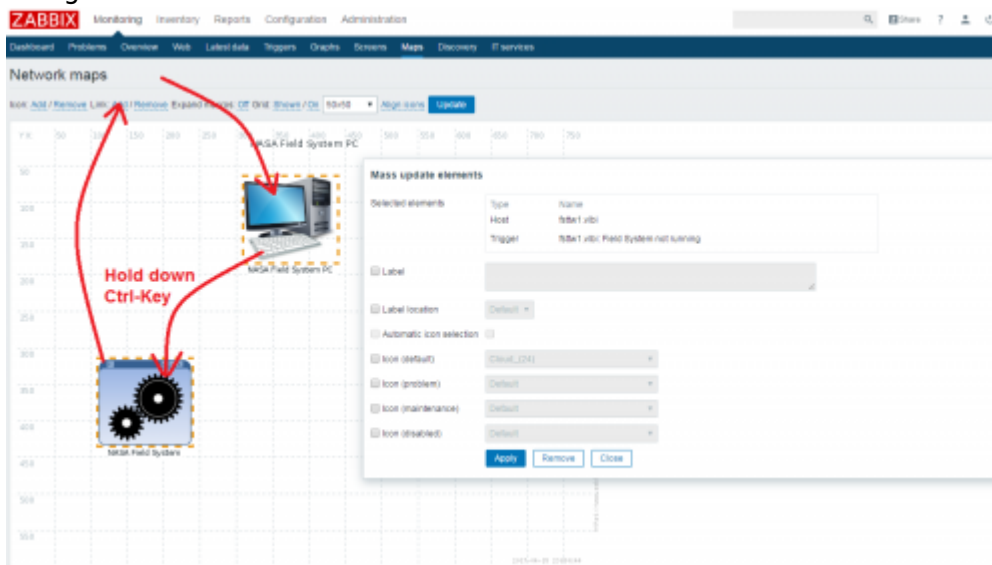
- Add a new icon and pull it to a position on the map. Click on the new icon and fill out the form which opens. Select the type (e.g. Host or Trigger or ...), set the label, assign a host, define the different sizes of the icons for different alert situations and add an individual screen link to the URLs (use the internal references shown in the image below). "Apply" and "Close" the dialog.

- Add further icons, e.g. for triggers

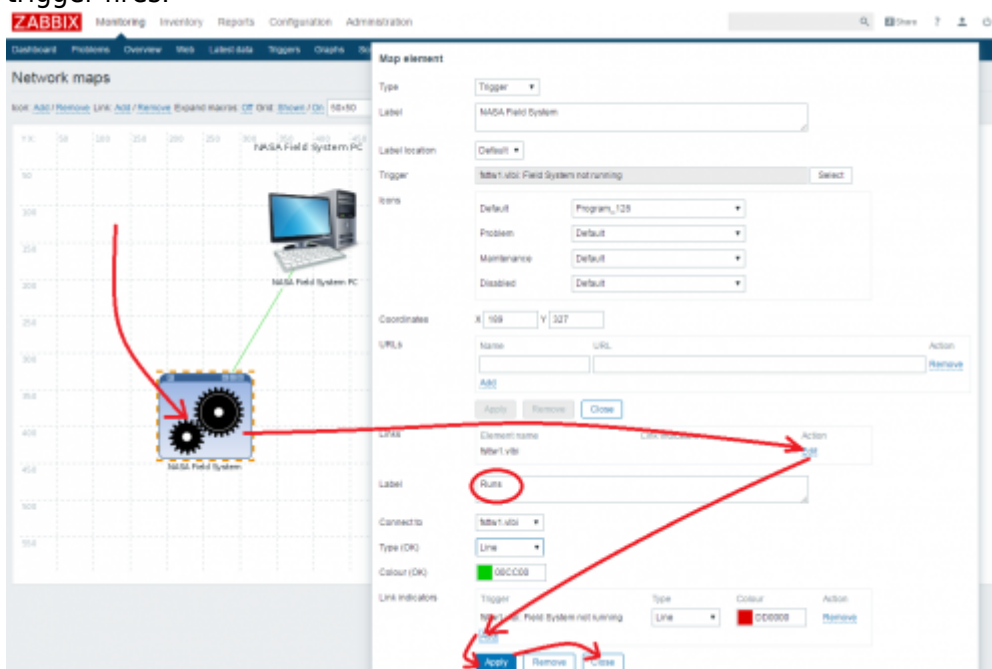




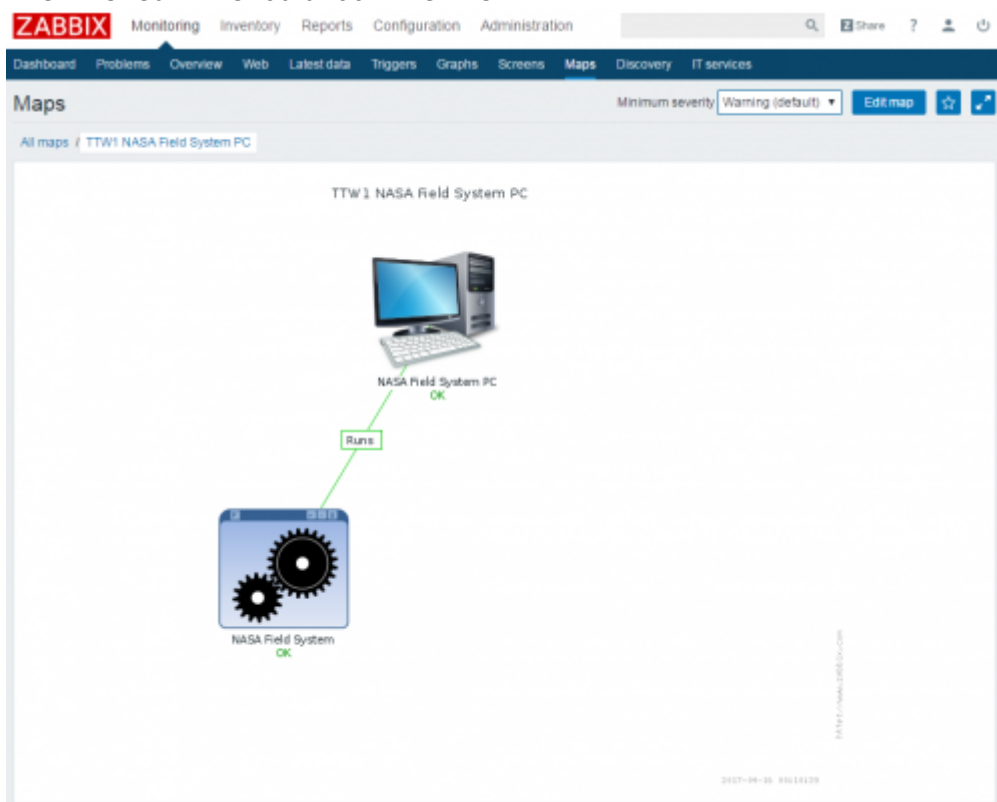
- Link the individual icons. Click on one icon, hold down the Ctrl-key and click on another icon. Using "Add" of a "Link" to connect both icons.



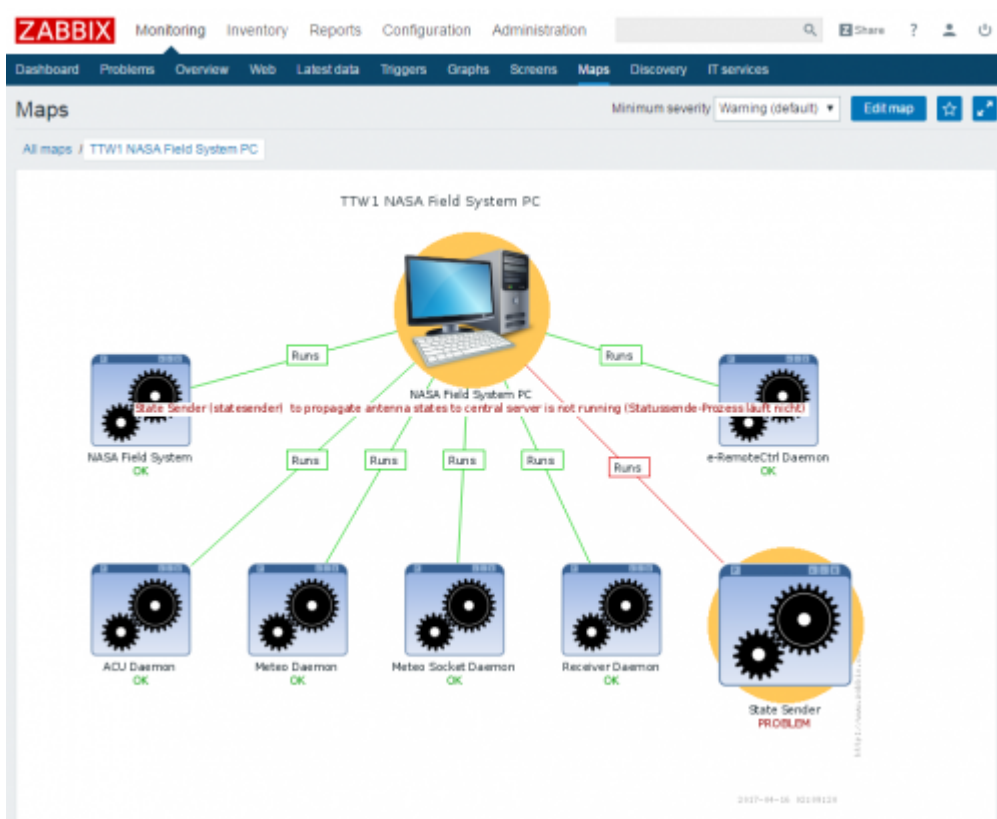
- Label the newly created link and connect it to a trigger, so that the line will be colored when the trigger fires.



- The finished link should look like this:



- If several system parameters are connected and set, the map with a warning severity looks like this:





From:

<http://wiki.wtz/> - **Geodetic Observatory - Wiki**

Permanent link:

[http://wiki.wtz/doku.php?id=vlbi:sysmon:001\\_zabbix\\_add\\_monitoring\\_for\\_nasafspc](http://wiki.wtz/doku.php?id=vlbi:sysmon:001_zabbix_add_monitoring_for_nasafspc)



Last update: **2017/04/18 18:04**

# Install and configure the monitoring of an SNMP device (operational and diagnostic data)

The ...

## 1) Prepare the server and agent for SNMP

### 1.1) SNMP-traps

- Zabbix is not able to receive SNMP-traps directly. It needs help from other tools:

```
sudo apt-get install snmp snmpd snmptt snmptrapd snmp-mibs-downloader
```

- Default is to run the agent snmpd, we need the snmptrapd running. Change configuration file **/etc/default/snmpd**

```
SNMPDRUN=no
```

In file **/etc/default/snmptrapd** change following line:

```
TRAPDRUN=yes
```

- To get readable traps, trapper daemon must pass the received traps to trap translate daemon. Therefore edit configuration file **/etc/snmp/snmptrapd.conf**

```
traphandle default /usr/sbin/snmptt
disableAuthorization yes
```

- To get zabbix conform messages edit following lines in file **/etc/snmp/snmptt.ini**

```
mode = standalone
translate_log_trap_oid = 2
net_snmp_perl_enable = 1
date_time_format = %H:%M:%S %Y/%m/%d
log_file = /tmp/zabbix_traps.tmp
log_system_enable = 1
mibs_environment = ALL
```

- Make backup of original file /etc/snmp/snmptt.conf and change the file to only containing following two lines:

```
EVENT general .* "General event" Normal
FORMAT ZBXTRAP $aA $ar severity:$s Fn+*
```

- After editing the config files we have to restart the services

Till Ubuntu 14.04:

```
service snmpd restart && service snmpd restart
```

Since Ubuntu 16.04:

```
systemctl restart snmpd.service && systemctl restart snmptrapd.service &&
systemctl restart snmpd.service
```

- Configure and restart zabbix-server

```
sudo nano /usr/local/etc/zabbix_server.conf
```

Delete # before following lines

```
SNMPTrapperFile=/tmp/zabbix_traps.tmp
StartSNMPTrapper=1
```

Restart server:

```
systemctl restart zabbix-server.service
```

- Configure logrotate for /tmp/zabbix\_traps.tmp

To prevent the trapper-file from growing to big we can use the linux tool logrotate. It should be already installed. Create configuration file

```
sudo nano /etc/logrotate.d/zabbix_traps
```

and insert following lines


```
/tmp/zabbix_traps.tmp {
 daily
 size 10M
 compress
 notifempty
 missingok
 maxage 365
 rotate 1
}
```

## 1.2) Configure SNMP monitoring of UPS with zabbix

- Configure SNMP for Twin- and RTW-UPS

Open web interface of UPS (<http://192.168.208.240>) in a web browser and log in. Go to Configuration→SNMP and set at least 192.168.208.235 (sysmonvlbi.vlbi - the vlbi zabbix server) as SNMP Trap Receiver. Set community as well.

**Adapter is rebooting now !**  
Saving logfiles, configuration and battery capacity to flash memory...



**CS121 Status**  
System & Network Status  
Device Status  
Device Status Graphic  
Device Functions

**Configuration**  
UPS Model & System  
Network & Security  
LED Display  
RAS Configuration  
Scheduled Actions  
SNMP  
Firmware  
Time/Server  
GPRS  
Sensor Manager  
Events / Alarms  
Save Configuration

**Logfiles**

**Web Links**

### SNMP Settings

| SNMP Communities |           |            | SNMP Trap Receivers |           |
|------------------|-----------|------------|---------------------|-----------|
| Address          | Community | Permission | Address             | Community |
| 1 0.0.0.0        |           | Read only  | 1 192.168.208.235   | public    |
| 2 0.0.0.0        |           | Read only  | 2 192.168.208.236   | public    |
| 3 0.0.0.0        |           | Read only  | 3 0.0.0.0           |           |
| 4 0.0.0.0        |           | Read only  | 4 0.0.0.0           |           |
| 5 0.0.0.0        |           | Read only  | 5 0.0.0.0           |           |
| 6 0.0.0.0        |           | Read only  | 6 0.0.0.0           |           |
| 7 0.0.0.0        |           | Read only  | 7 0.0.0.0           |           |
| 8 0.0.0.0        |           | Read only  | 8 0.0.0.0           |           |
| 9 0.0.0.0        |           | Read only  | 9 0.0.0.0           |           |
| 10 0.0.0.0       |           | Read only  | 10 0.0.0.0          |           |

Please note: Any IP has SNMP access unless you define IP addresses here

**Test SNMP Traps**  
You can send a powerfail trap and a power restored trap to the receivers defined below.  
Please note: To test newly added receivers, you must save the configuration and reboot the CS121 first.

192.168.208.235 public  
192.168.208.236 public  
192.168.208.235 public

Click Apply and Configuration→Save Configuration. Save and reboot. Do the same for usvantrtw.vlbi (<http://192.168.208.241>)

- Add hosts and Link with the snmp ups templates

Login to web interface of Zabbix on <http://192.168.208.235> Go to Configuration→Hosts. At right upper corner click Create host.

**ZABBIX** Monitoring Inventory Reports Configuration Administration

Host groups Templates **Hosts** Maintenance Actions Event correlation Discovery IT services

## Hosts

All hosts / **usvanrtw.vlbi** Enabled ZBX **SNMP** JMX IPMI Applications 2 Items 9 Triggers 7 Graphs Discovery rules Web scenarios

Host Templates IPMI Macros Host inventory Encryption

Host name

Visible name

Groups In groups

UPS

Other groups

Discovered hosts  
Hypervisors  
Linux servers  
Marel UPS  
NASA Field Systems  
Templates\_GOW  
Templates\_ZabbixExamples  
Templates\_ZabbixExamples  
TTW1  
TTW1Dewar Host

New group

Agent interfaces

IP address DNS name Connect to Port Default

Add

SNMP interfaces

192.168.208.241

IP DNS 161

Remove

☒ Use bulk requests

Add

JMX interfaces

Add

IPMI interfaces

Add

Description

UPS in cellar of RTW control building.

Monitored by proxy

(no proxy)

Enabled

☒

Update

Clone

Full clone

Delete

Cancel

Insert host name and group UPS. Add a SNMP interface. The agent interface can be removed. Insert description and click Update.

From:  
<http://wiki.wtz/> - **Geodetic Observatory - Wiki**

Permanent link:  
[http://wiki.wtz.doku.php?id=vlbi:sysmon:001\\_zabbix\\_add\\_monitoring\\_for\\_snmp](http://wiki.wtz.doku.php?id=vlbi:sysmon:001_zabbix_add_monitoring_for_snmp)

Last update: **2017/06/22 17:28**

# Install and configure the monitoring of a SysMon node (analysis data)

The data from other equipments at location of the telescope are interesting for analysis and not only for operations. Therefore, they are **analysis data** which are saved to be used by the VLBI analysis centers. Analysis data are managed by SysMon and in the Zabbix system. There is a web archive containing the data history over years in individual formats on the system monitoring PC.

The management of the analysis data requires a suitable installation of the Wettzell System Monitoring Suite (SysMon): see [0\) SysMon Node VLBI](#).

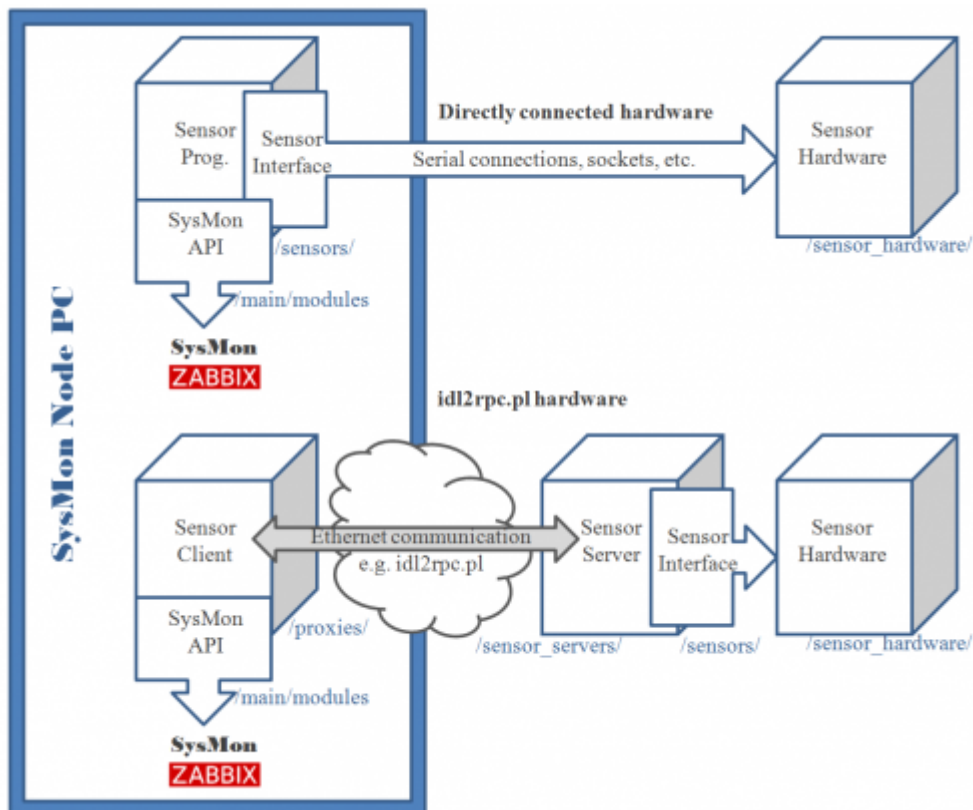
- The SysMon suite is maintained by the Wettzell observatory (see <http://xsamba.wtz/svn/vlbi/trunk/code/vlbisysmon/>) and consists of the following parts, while for own programs only the folder “main” is essential.



- - **main:** contains all central modules and programs of the current SysMon suite, e.g.
    - **modules:** with all the C/C++ modules of the API
    - **sysmon\_backup:** a program which can be used to backup content from the SysMon database into files
    - **sysmon\_sender:** a program which can be used in scripts and other programs to register sensors and to send data to SysMon using system calls
  - **proxies:** Proxies are predefined clients, which fetch data from already existing sensor servers (mainly for Wettzell observatory needs in the form of idl2rpc.pl clients)
  - **sensor hardware:** Programs which run on sensor hardware like microcontrollers
  - **sensor servers:** Server programs (mainly for Wettzell observatory needs in the form of idl2rpc.pl servers) which fetch data from sensors, process them and offer them to clients.
  - **sensor:** Modules and test programs which realize the interface to a sensor hardware
- The SysMon node PCs use the following software components (
 

PPT-Images collection

 ):



- The SysMon sender or SysMon API must always be used on the PC where SysMon and Zabbix is running. A remote data injection is not enabled. Therefore, proxies can be used.

## 1) Create an own C/C++ program to send in data of a dedicated sensor control point or use a script calling "sysmon\_senderc"

- All Wettzell programs are located in the SysMon project. Other external programs just need the files from directory "main" of the SysMon suite.
- A simple program just consists of the following scheme:

```

.....
#include "mcidb_access.hpp"
.....

int main (int argc, char * argv[])
{
 /// Variables
 unsigned short usError = 0;
 mcidb_access CSysMonAPI; /// CSysMonAPI = object to access the
 Wettzell System Monitoring SysMon

 /// Check program arguments where a configuration file should be
 handed over somehow
 ...

 /// Register new sensor control point using a predefined
 configuration file

```

```

 if (CSysMonAPI.usRegisterSensors(argv[1]))
 {
 std::cout << "[ERROR] Cannot register sensors\n";
 usError = 2;
 goto FinishProgram;
 }

 while (true)
 {
 double dPressureMBar = 0.0; /// dPressureMBar = pressure
value of a dewar
 std::stringstream ssValue; /// ssValue = conversion of
double values to strings
 ...

 /// Open connection to sensor hardware
 ...

 /// Read data from sensor, e.g a pressure value of the dewar
"dPressureMBar"
 ...

 /// Close connection to sensor hardware
 ...

 /// Send data to the system monitoring SysMon, e.g. a pressure
value of the dewar "dPressureMBar"
 ssValue << std::fixed << std::setprecision(7) << dPressureMBar
* 10000; // =0.0001000*10^-4 mbar
 if (CSysMonAPI.usSendSingleData ("TTW1Dewar_Pressure",
ssValue.str()))
 {
 std::cerr << "[ERROR] Cannot send TTW1Dewar_Pressure to
SysMon\n";
 }
 ssValue.str("");
 ...

 /// Manage own timing interval
 sleep (60);
 }

FinishProgram:
 /// Error processing
 ...

 return 0;
}

```

- Scripts can also just call the program "sysmon\_senderc" which is located at



"/home/oper/Software/vlbisysmon/main/sysmon\_sender/bin/ " using one of the following arguments:

- `sysmon_send <-option>`  
where <-option> is:
  - I configfile.conf (register)
  - R configfile.conf (deregister)
  - D configfile.conf (delete)
  - d configfile.conf netsensorid (delete netsensorid)
  - S configfile.conf (register, update, write, send)
  - s configfile.conf netsensorid value [alarmlevel] (write, send)
  - f configfile.conf datafile.txt (write, send)

## 2) Register the sensor control point at SysMon

- It is necessary to create a standardized SysMon configuration file which describe the new sensor control point with its sensors to register it in the SysMon system.
- A basic explanation of the configuration can be found in the Monitoring and Control Infrastructure Whitepaper:  
20120323mciworkingdocument.pdf
- An example of such a configuration looks like the following file "dewarproxyc.conf" describing the dewar values of the Wn antenna at Wettzell:

- ```
<MCISensorControlPoint>
    ControlPointID           = TTW1Dewar
    ControlPointType         = Proxie
    ControlPointPort         = 52700
    <MCISensorProprietarySettings>
        <Communication>
#           <IDL2RPCServer> # not yet implemented
#           SocketConnect    = 1
#Flag for Connection
#           IPAddress         = 192.168.208.13      #IP
Address
#           Port              = 52666
#PortNumber
#           </IDL2RPCServer>
            <IPServer>
                SocketConnect    = 1                #Flag
for Connection
                IPAddress         = 192.168.208.13    #IP
Address
                Port              = 52666
#PortNumber
            </IPServer>
            <SerialConnection>
                SerialConnect     = 0                #Flag
for Connection
```

```

        tty                                = /dev/ttyS0
#Serial Port tty
        FDtty                              = 3                                #File
Descriptor for tty
        </SerialConnect>
        <Settings>
            Timeout                          = 3                                #For
both connections (sec)
        </Settings>
        </Communication>
        <WriteSensorData>
            FilePath                          =
/var/www/html/monitoring_archive            #Path of FileArchive
            WriteTime                          = 5
#Write every 1,2,3,4,5,6,10,12,15,20,30 or 60 minutes
        </WriteSensorData>
        <WriteZabbixData>
            Execution                          = yes
#yes/no Create file and send to zabbix
            FilePath                          = /tmp                                #Path
of zabbix_file.txt
            FileName                          = zabbix_file.txt
            ServiceFilePath                    = /usr/bin/zabbix_sender            #for
the shell-command
        </WriteZabbixData>
        <CreateZabbixImportTemplates>
            Execution                          = yes
            FilePath                          =
/home/oper/Software/vlbisysmon/proxies/rxmon/xml            #Path of
zabbix_import_template and _host.txt
            FileNameHostTemplate              = zabbix_import_host.xml
        </CreateZabbixImportTemplates>
    </MCISensorProprietarySettings>
    <MCIZabbixConnection>
        Host                                = 127.0.0.1
        Port                                = 5432
        DBName                              = zabbix
        UserName                            = zabbix
        Pwd                                 = zabbix
        Timeout                             = 3
    </MCIZabbixConnection>
    <MCIDBConnection>
        Host                                = 127.0.0.1
        Port                                = 5432
        DBName                              = sysmon
        UserName                            = sysmon
        Pwd                                 = +sysmon!
        Timeout                             = 3
    </MCIDBConnection>
    <MCIBackupSettings>
        Execution                            = yes

```

```
#yes/no Create backupfile and delete from database tables
    ArchiveDays                      = 30
#Interval of backups (days)
    BackupPath                      = /tmp                                #Path
for TableBackups
    ServiceBackupPath              =
/home/oper/Software/vlbisysmon/main/sysmon_backup/bin/sysmon_backupc
#Path for the module sysmon_backup
    </MCIBackupSettings>
    <MCISensor>
        SensorID                    = TTW1Dewar_TempAmbient
        SensorName                  = TempAmbient
        SensorUnit                  = deg C
        SensorManufacturer          = X
        SensorModel                 = X
        SensorPosition              = X
        SensorUpdateInterval        = X
        SensorResolution            = X
        SensorDataAvailabilityTime  = X
        SensorMinLimit              = X
        SensorMaxLimit              = X
        SensorMinWarningLimit       = 5
        SensorMaxWarningLimit       = 35
        SensorMinAlertLimit         = 0
        SensorMaxAlertLimit         = 40
        SensorFlagProvider          = yes
        SensorFlagConsumer          = no
        SensorFlagCommandable       = no
        SensorFlagManageable        = no
        SensorDataArchiveDirectory = X
        SensorPropArchiveDirectory =
    </MCISensor>
    <MCISensor>
        SensorID                    = TTW1Dewar_TempFirstStage
        SensorName                  = TempFirstStage
        SensorUnit                  = K
        SensorManufacturer          = X
        SensorModel                 = X
        SensorPosition              = X
        SensorUpdateInterval        = X
        SensorResolution            = X
        SensorDataAvailabilityTime  = X
        SensorMinLimit              = X
        SensorMaxLimit              = X
        SensorMinWarningLimit       = 0
        SensorMaxWarningLimit       = 30
        SensorMinAlertLimit         = 0
        SensorMaxAlertLimit         = 32
        SensorFlagProvider          = yes
```

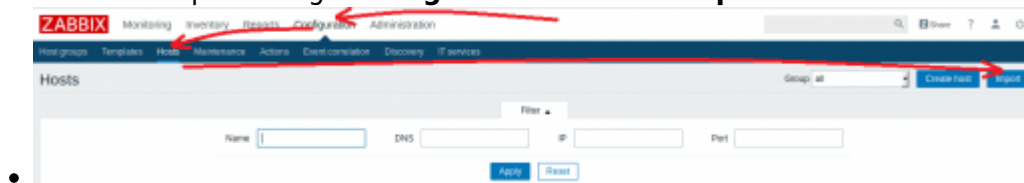
```
SensorFlagConsumer          = no
SensorFlagCommandable       = no
SensorFlagManageable        = no
SensorDataArchiveDirectory  = X
SensorPropArchiveDirectory  =
</MCISensor>
<MCISensor>
  SensorID                  = TTW1Dewar_TempSecondStage
  SensorName                = TempSecondStage
  SensorUnit                = K
  SensorManufacturer        = X
  SensorModel               = X
  SensorPosition            = X
  SensorUpdateInterval      = X
  SensorResolution          = X
  SensorDataAvailabilityTime = X
  SensorMinLimit            = X
  SensorMaxLimit            = X
  SensorMinWarningLimit     = 0
  SensorMaxWarningLimit     = 10
  SensorMinAlertLimit       = 0
  SensorMaxAlertLimit       = 12
  SensorFlagProvider        = yes
  SensorFlagConsumer        = no
  SensorFlagCommandable     = no
  SensorFlagManageable      = no
  SensorDataArchiveDirectory = X
  SensorPropArchiveDirectory =
</MCISensor>
<MCISensor>
  SensorID                  = TTW1Dewar_Pressure
  SensorName                = Pressure
  SensorUnit                = e-4 mbar
  SensorManufacturer        = X
  SensorModel               = X
  SensorPosition            = X
  SensorUpdateInterval      = X
  SensorResolution          = X
  SensorDataAvailabilityTime = X
  SensorMinLimit            = X
  SensorMaxLimit            = X
  SensorMinWarningLimit     = 0
  SensorMaxWarningLimit     = 4
  SensorMinAlertLimit       = 0
  SensorMaxAlertLimit       = 10
  SensorFlagProvider        = yes
  SensorFlagConsumer        = no
  SensorFlagCommandable     = no
  SensorFlagManageable      = no
  SensorDataArchiveDirectory = X
  SensorPropArchiveDirectory =
```

```
</MCISensor>
</MCISensorControlPoint>
```

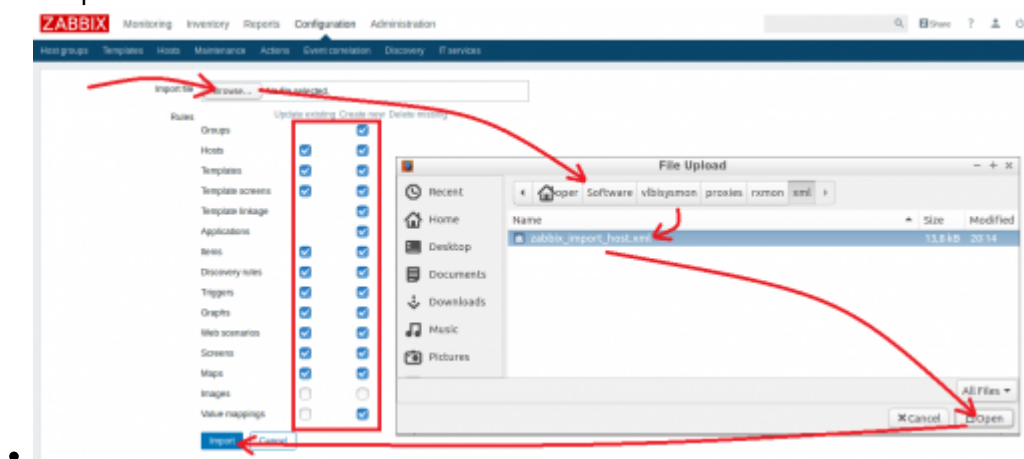
- Register the new sensor control point:
 - The registration is processed using the function "*CSysMonAPI.usRegisterSensors(argv[1])*" in the above program
 - Another possibility to register the sensors is to use the "*sysmon_senderc*" program, which can be found here "*/home/oper/Software/vlbisysmon/main/sysmon_sender/bin/*" in the described installation. Call the program with the following parameter:
 - `sysmon_senderc -R dewarproxyc.conf`

3) Import sensor control point template as host to Zabbix

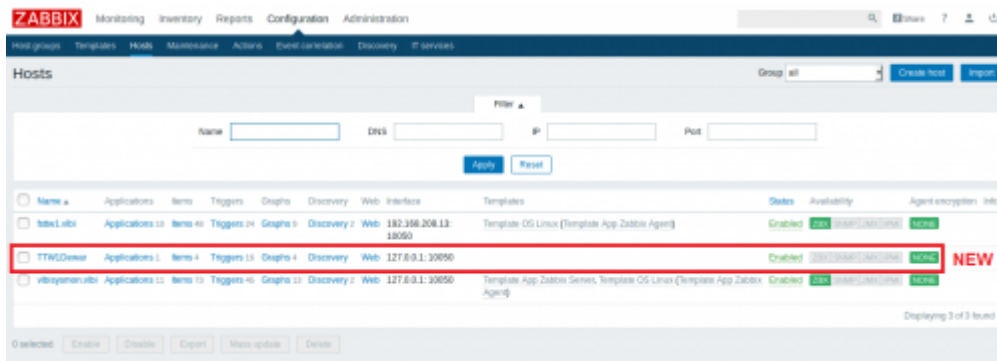
- The registration creates a separate XML-file if the configuration file contains a valid "*<CreateZabbixImportTemplates>*" block.
- The generated XML-file describes the complete host settings for the new sensor control point and can directly be imported to Zabbix. The import can be done using the web interface of Zabbix.
- Start the import using "**Configuration→Hosts→Import**"



- Select the rules for "Update existing", "Create new" and "Delete missing" and browse to your file in the "File Upload" window. Select the newly create XML template file, click "Open" and then "Import".



- After the import you should find a new host with some items, triggers, graphs, etc. in the list of hosts



4) Send in data and check the arrival

- There are two possibilities to send data:
 - Start your C or C++ program (or proxy)


```

** Open connection to rxmon server at 192.168.208.13:52666
** Read dewar values from rxmon server at 192.168.208.13:52666
500deg C
info from server: "processed: 1; failed: 0; total: 1; seconds spent: 0.000047"
sent: 1; skipped: 0; total: 1
27.000K
info from server: "processed: 1; failed: 0; total: 1; seconds spent: 0.000044"
sent: 1; skipped: 0; total: 1
9.000K
info from server: "processed: 1; failed: 0; total: 1; seconds spent: 0.000028"
sent: 1; skipped: 0; total: 1
1.591000010^-4 mbar
info from server: "processed: 1; failed: 0; total: 1; seconds spent: 0.000051"
sent: 1; skipped: 0; total: 1
** Close connection to rxmon server.
          
```
 - or run the program "sysmon_senderc" in the directory , which can be found here `"/home/oper/Software/vlbisysmon/main/sysmon_sender/bin/"` using:


```

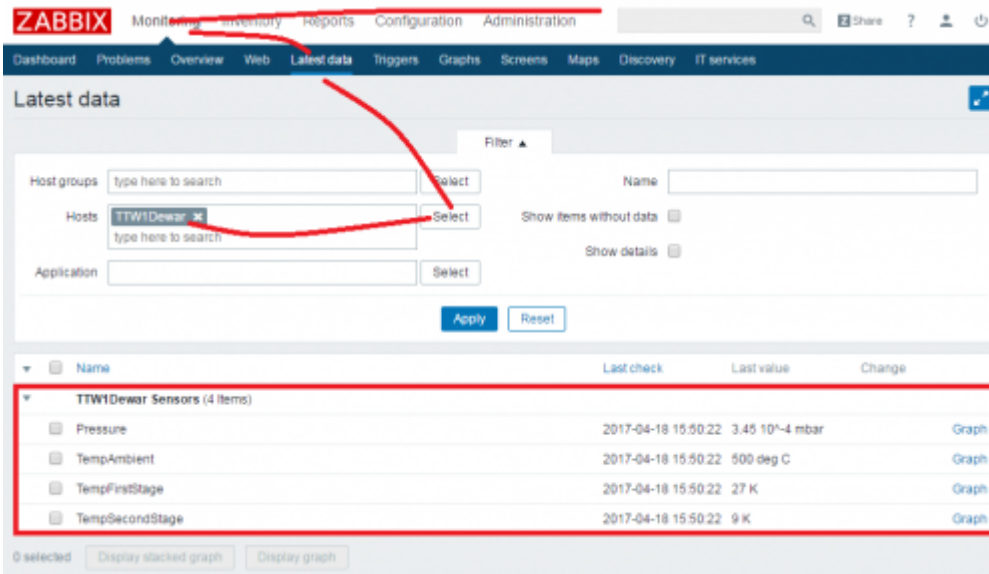
sysmon_senderc -s dewarproxyc.conf TTW1Dewar_TempAmbient
23.0
          
```

Arguments:

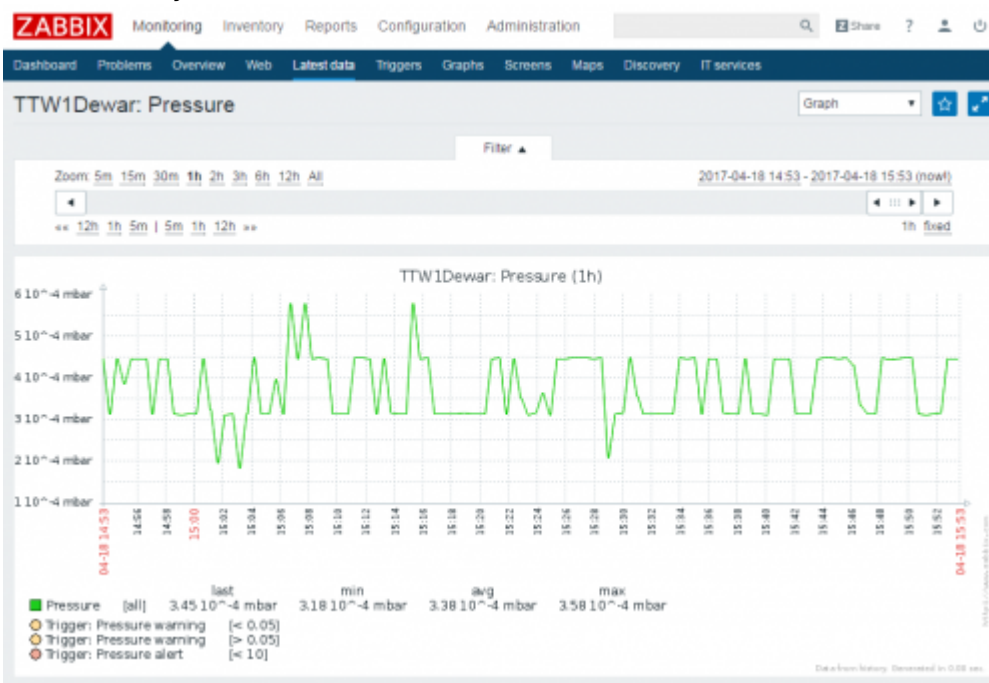
=====

-s dewarproxyc.conf => the configuration file used
 TTW1Dewar_TempAmbient => SensorID
 23.0 => New value

- Check if the data arrive in Zabbix.



- You can directly click on “Graph” link behind each received new value to show the data history of the already received data.



- Administrators can also check if the data arrive in the sysmon databases using “psql” program of PostgreSQL.

```
psql -h 127.0.0.1 -p 5432 sysmon sysmon
select * from mcicurrentvalues;
```

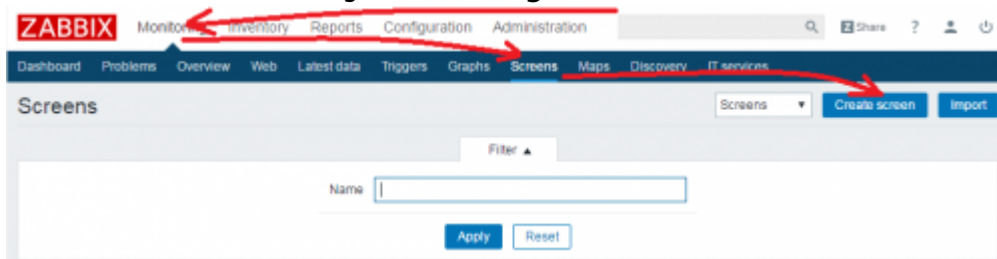
- You should see something like this for your values:

```
sysmon=# select * from mcicurrentvalues ;
      netsensorid      |      mjd      | alarmlevel | value
-----+-----+-----+-----
TTW1Dewar_TempAmbient  | 57861.58190972 |          3 |    500
TTW1Dewar_TempFirstStage | 57861.58190972 |          3 |     27
TTW1Dewar_TempSecondStage | 57861.58190972 |          3 |      9
TTW1Dewar_Pressure     | 57861.58190972 |          3 |   3.446
(4 rows)
```

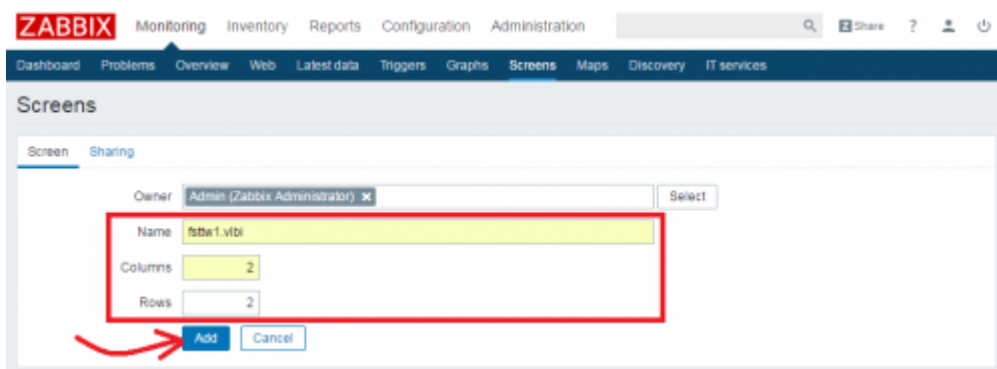
- End with Ctrl-Shift-'D'

5) Create individual screens and maps

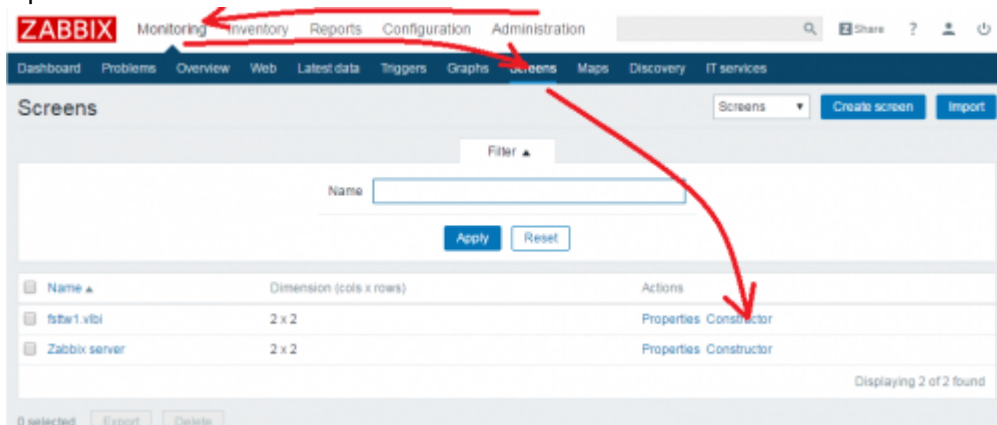
- Create a new screen using “**Monitoring→Screens→Creat screen**”



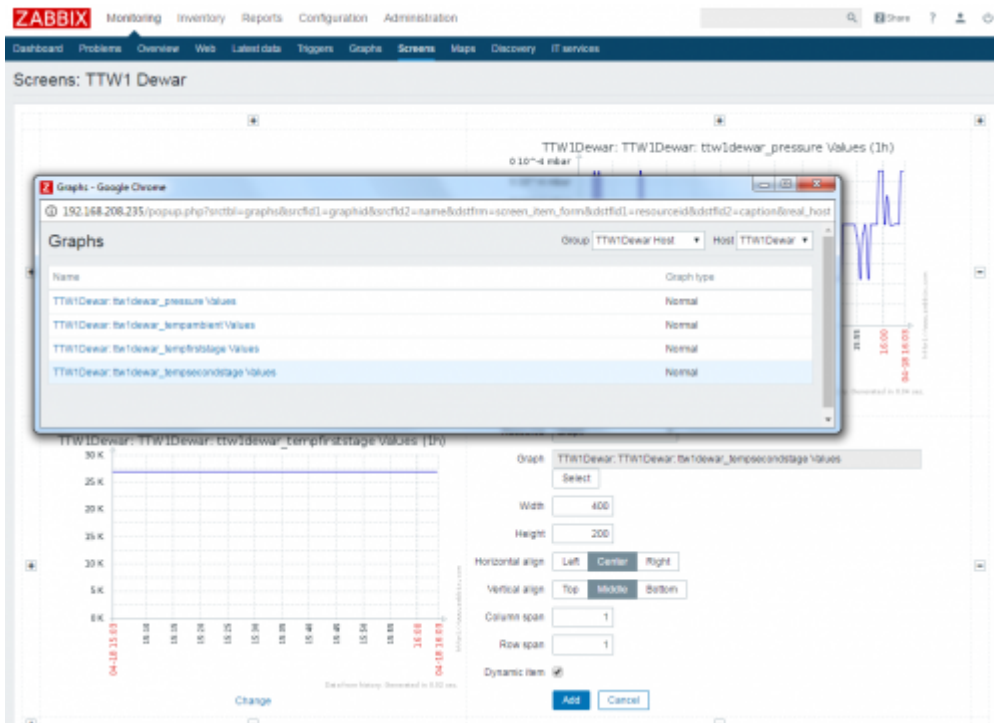
- Define a name for the screen and the dimensions as number of rows and columns and push “Add”.



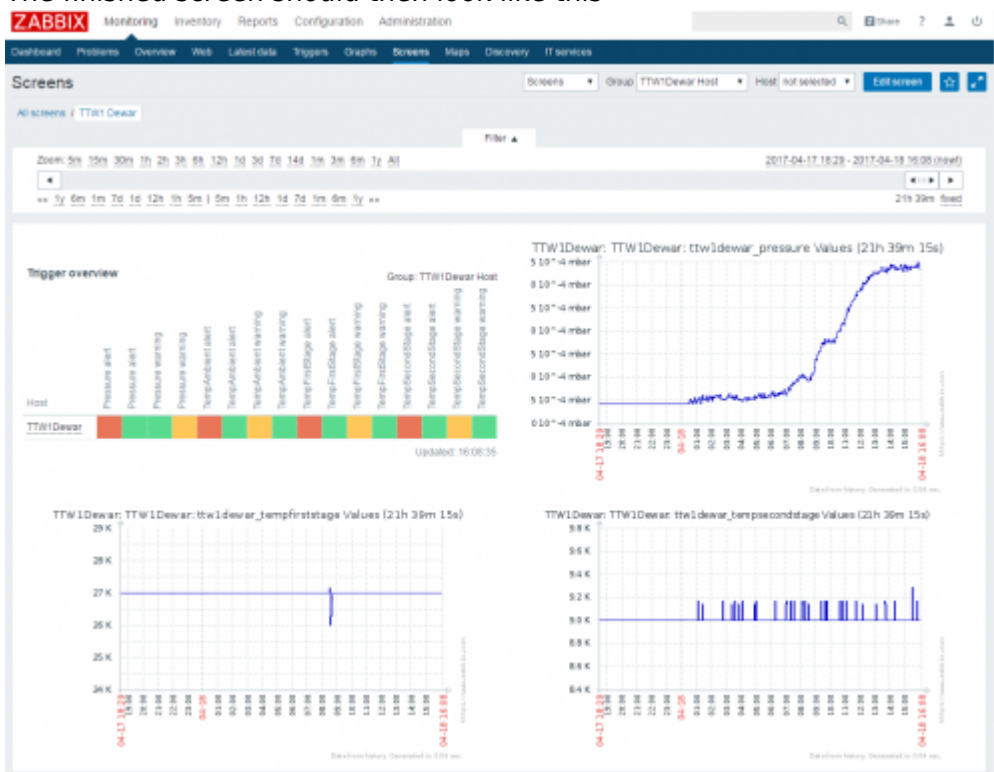
- Open the “Constructor” of the screen



- Push the “Change” link for each of the individual fields on the screen.



- Do this for all individual graphs and elements which should be shown on the screen.
- The finished screen should then look like this



From:
<http://wiki.wtz/> - Geodetic Observatory - Wiki

Permanent link:
http://wiki.wtz/doku.php?id=vlbi:sysmon:002_zabbix_add_monitoring_for_sysmonnode

Last update: 2017/04/26 18:14

Install and configure the monitoring of IVS Seamless Auxiliary Data (operation, diagnostic, and analysis data)

The data from other IVS and other telescope sites are interesting for operation, diagnostic, and analysis. The data are sent by the telescopes. The incoming data are managed by SysMon or not yet existing converters to save them as files and to present them with Zabbix. There is a web archive containing the data history over years in individual formats on the system monitoring PC.

The following description is not completely realized yet but in preparation.

[A general explanation can be found here:](#)

ivstaskforceseamlessauxiliarydata.pdf

1) Used data formats

- All data arrive as files in a special incoming folder.
- The current complete file with all operation, diagnostic, and analysis data follows a simple, self-explaining structure of the following type:

```
• <eQuickStatusInfo>
  Service = IVS
  Stationname = WETTZELL
  StationIVSCode = Wz
  Schedule = k14242wz
  Status = [eRC] Recording<br>
  DateTime = 2014.242.07:42:52
  TimeNext = 07:42:53
  Source = 0016+731
  Scan = k14242_wz_242-0742
  Mark5VSN = BKG-E002
  Mark5Volume = 1470.0
  Mark5Used = 0.8
  RightAscension = 00h19m45.79s
  Declination = 73d27m30.00s
  Azimuth = 337.5532
  Elevation = 43.4183
  CableDelay = 0.006511
  SystemTemperatureIFA = 34
  SystemTemperatureIFB = 98
  SystemTemperatureIFC = 32
  SystemTemperatureIFD = 0
  MeteorologyTemperature = 16.8
  MeteorologyHumidity = 86.3
  MeteorologyPressure = 948.7
</eQuickStatusInfo>
```

- A simplified version containing just analysis data can be combined of the following blocks, where DateTime and StationIVSCode is optional (DateTime will be automatically added by the system monitoring server according to the local time of the server):
- for meteo data

```
• <eQuickStatusInfo>
  Stationname = WETTZELL
  StationIVSCode = Wz
  DateTime = 2014.242.07:42:52
  MeteorologyTemperature = 16.8
  MeteorologyHumidity = 86.3
  MeteorologyPressure = 948.7
</eQuickStatusInfo>
```

- for cable measurements data

```
• <eQuickStatusInfo>
  Stationname = WETTZELL
  StationIVSCode = Wz
  DateTime = 2014.242.07:42:52
  CableDelay = 0.006511
</eQuickStatusInfo>
```

- for system temperatures data

```
• <eQuickStatusInfo>
  Stationname = WETTZELL
  StationIVSCode = Wz
  DateTime = 2014.242.07:42:52
  SystemTemperatureIFA = 34
  SystemTemperatureIFB = 98
  SystemTemperatureIFC = 32
  SystemTemperatureIFD = 0
</eQuickStatusInfo>
```

- or in a combined file

```
• <eQuickStatusInfo>
  Stationname = WETTZELL
  StationIVSCode = Wz
  DateTime = 2014.242.07:42:52
  CableDelay = 0.006511
  SystemTemperatureIFA = 34
  SystemTemperatureIFB = 98
  SystemTemperatureIFC = 32
  SystemTemperatureIFD = 0
  MeteorologyTemperature = 16.8
```

```
MeteorologyHumidity = 86.3  
MeteorologyPressure = 948.7  
</eQuickStatusInfo>
```

- The idea is to send the data regularly and not only during a session, where the focus is on meteorological data.
- **Attention: The following first attempt just implements meteorology processing.**

2) Register your sites

- To register your site, please send an email to neidhardt@fs.wettzell.de

3) Sending data

4) Accessing data

From:

<http://wiki.wtz/> - Geodetic Observatory - Wiki

Permanent link:

http://wiki.wtz/doku.php?id=vlbi:sysmon:003_zabbix_add_monitoring_for_seamless_auxiliary_data



Last update: **2017/04/26 17:29**

6 Appendix: openMoniCA (on Ubuntu 16.04 LTS)

Test-analysis by Edoardo Barbieri (TUM, Jumping JIVE)

Features

Name	Access control	License	Platform	Distributed monitoring	Web App	Trigger/alerts
MoniCA	yes	Open source	Java	Yes	Yes	Yes

Logical grouping	Trending	Latest release version	Latest release date	IPv6	SNMP	Data Storage method
yes	Yes	1	6 November 2016	Yes	Yes	MySQL

Requirements:

- Ubuntu desktop 16.04 LTS (this guide should work on other system as well, though this is the one used as test)
- **ant** and **JDK** installed

if not already installed please follow this links

<https://www.digitalocean.com/community/tutorials/how-to-install-java-on-ubuntu-with-apt-get>

<https://www.linuxhelp.com/how-to-install-apache-ant-on-ubuntu/>

Download:

Please download the source code from the following git repository

<https://github.com/davidbrodrick/open-monica>

From terminal:

```
svn checkout https://github.com/davidbrodrick/open-monica.git open-monica-master
```

Or using the download button. Uncompress the zip file if you used the button.

Preinstallation steps

Go in the main folder of the software

```
cd open-monica-master/trunk/  
#or  
cd open-monica-master/
```

Depending on how you downloaded the source code.

Copy the default configuration files from subdirectory default-files/ to the official configuration folder config/

```
cp default-files/monitor* config/  
cp default-files/log4j.properties config/
```

This will be the directory where we can modify the configuration.
The file log4j.properties should be also copied in the main directory since this might give problems when we run the software:

```
cp default-files/log4j.properties .
```

Building

From the main folder (where the build.xml file is)

```
cd open-monica-master/trunk/
```

Invoke ant

```
ant
```

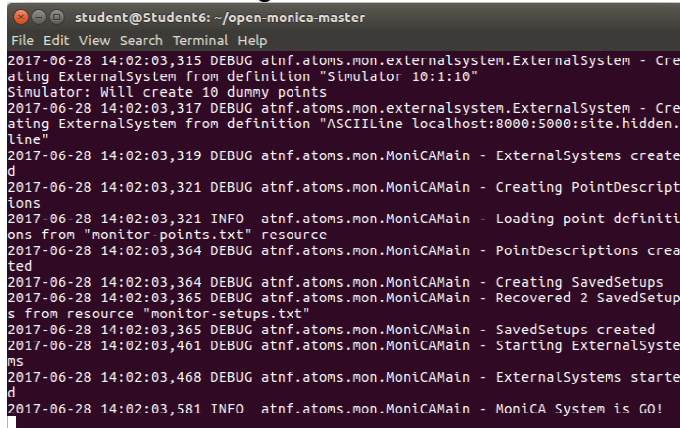
This command has to be invoked any time we make changes in the configuration files to apply those changes.
This will **compile the source and configuration files** and produce an executable .jar file.

Launching

To launch the server go to the main folder where the executable file is present and type:

```
java -jar open-monica.jar
```

This will launch MoniCA server on your machine. You should see on the terminal something like this

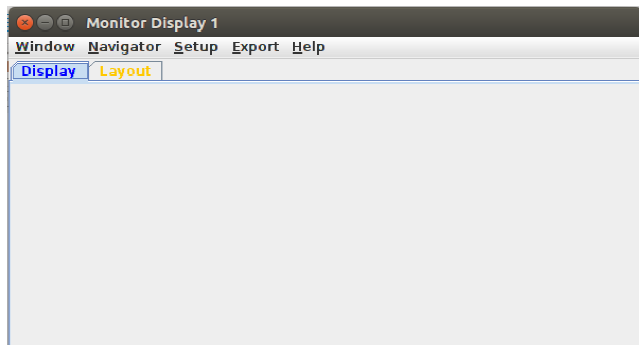


```
student@Student6: ~/open-monica-master
File Edit View Search Terminal Help
2017-06-28 14:02:03,315 DEBUG atnf.atoms.mon.externalsystem.ExternalSystem - Cre
ating ExternalSystem from definition "Simulator 10:1:10"
Simulator: Will create 10 dummy points
2017-06-28 14:02:03,317 DEBUG atnf.atoms.mon.externalsystem.ExternalSystem - Cre
ating ExternalSystem from definition "ASCIIline localhost:8000:5000:site.hidden.
line"
2017-06-28 14:02:03,319 DEBUG atnf.atoms.mon.MonicaMain - ExternalSystems create
d
2017-06-28 14:02:03,321 DEBUG atnf.atoms.mon.MonicaMain - Creating PointDescript
ions
2017-06-28 14:02:03,321 INFO atnf.atoms.mon.MonicaMain - Loading point definiti
ons from "monitor.points.txt" resource
2017-06-28 14:02:03,364 DEBUG atnf.atoms.mon.MonicaMain - PointDescriptions crea
ted
2017-06-28 14:02:03,364 DEBUG atnf.atoms.mon.MonicaMain - Creating SavedSetups
2017-06-28 14:02:03,365 DEBUG atnf.atoms.mon.MonicaMain - Recovered 2 SavedSetup
s from resource "monitor.setups.txt"
2017-06-28 14:02:03,365 DEBUG atnf.atoms.mon.MonicaMain - SavedSetups created
2017-06-28 14:02:03,461 DEBUG atnf.atoms.mon.MonicaMain - Starting ExternalSyste
ms
2017-06-28 14:02:03,468 DEBUG atnf.atoms.mon.MonicaMain - ExternalSystems starte
d
2017-06-28 14:02:03,581 INFO atnf.atoms.mon.MonicaMain - MoniCA System is GO!
```

To run the client type in the same folder

```
java -cp open-monica.jar atnf.atoms.mon.gui.MonFrame
```

Whit this command the client should show up on your screen



Deployment

If you wish to deploy the system on the machine, go again in the main directory where the build.xml file is present and type

```
sudo ant install
#or if you want to deploy to another directory
sudo ant -Dprefix=/my/install/dir/ install
```

The default installation directory if you don't use any prefix is /usr/local
Now you will find scripts to facilitate the starting/stopping of MoniCA under

```
/my/install/dir/bin/open-monica-server.sh start
/my/install/dir/bin/open-monica-client.sh
```

It is possible that the script to start the server does not work properly because of a bug that has still not been fixed. In this case you just need to start it manually with the java command

```
java -jar /my/install/dir/lib/open-monica.jar
```


7 Appendix: Telegraf - InfluxDB - Grafana (TIG) for the NASA Field System

Presentation by David Horsley (GSFC,NASA,NVI) during TOW 2017 at the MIT Haystack Observatory (use with special permission by David Horsley).

Acknowledgement: Special thanks to David Horsley for his permission to include the TIG documentation and for his willingness of future support to integrate TIG.

TIG for VLBI Operations

Dave Horsley david.e.horsley@nasa.gov

Feb 2017

- Introduction
- Server
 - Installation
 - Configuration
- Clients
 - Installation
 - Configuration
- Working directly with InfluxDB
 - Metadata
 - Basic Queries
 - Functions
- Working with Grafana
 - Adding the Database
 - Creating a Dashboard
 - Importing Dashboards
 - Other topics
- Using InfluxDB with other tools
 - Python
- Creating new collectors
 - Shell
 - Go
 - Python
- Advanced Web Setup
 - Reverse Proxy
 - HTTPS
- Advanced Data-flow Models

Introduction

This document is a work-in-progress, please send comments, questions and feedback to Dave.

The Telegraf, InfluxDB and Grafana (TIG) provide a system for collecting, storing, processing, and visualizing time-series data. The three component are loosely coupled together and each swapped for an alternative package. The purpose of this document give an overview of these tools use in VLBI operations and to guide a user through the installation process. The reader is expected to be competent with a Linux OS.

The role of components are as follows:

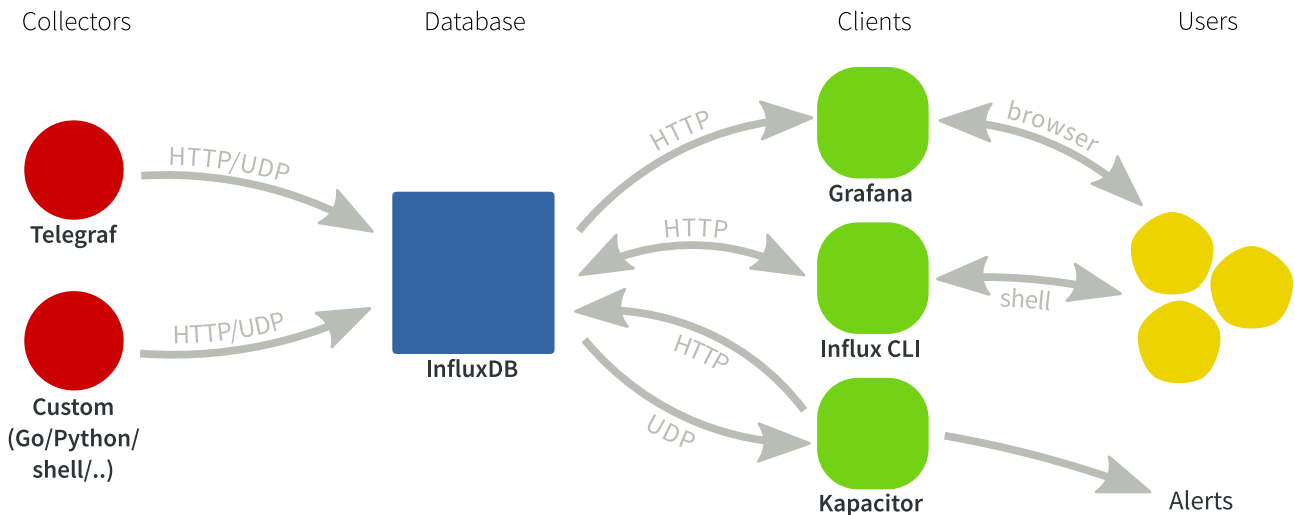


Figure 1: Data flow overview in the TIG suite.

- **Telegraf** collects data from different sources. Telegraf runs on every computer where you want to collect statistics. Telegraf includes plugins for collecting data on things such as:
 - disk usage and load
 - system load
 - network load and performance
 - process statistics
 - system sensors

The VLBI branch, provided in the FS repository, contains plugins for:

- The Field System (log, schedule, some RDBE data)
- Modbus Antennas (Currently Patriot 12m of the AuScope/GGAO generation)
- MET4 meteorological system via metserver
- RDBE multicast

Telegraf also large range of DevOps tools, which VLBI users may be less interested in, for example:

- web servers
 - mail servers
 - database servers
 - message queues
- **InfluxDB** is a time-series database. It offerers high-performance compression and retrieval for this type of data. It also has functions for processing and manipulating the data. It is similar to relational databases you may be familiar with, but is far more efficient at handling

time-series data. While InfluxDB has an SQL-like query language, it is distinct and it is best to consider it as a new system.

Like an SQL type database, InfluxDB method of getting data is a push model. This means the clients, the programs with the data, initiate the connection and write to the database. If you require a fetch model, you must write your own *collector* program. Telegraf fill this role for some purposes.

The load on the system it runs on can be fairly high, depending on the number of points you are monitoring. For this reason, it is worth doing some testing and tuning if you wish to run it on your FS PC. If you can, it is best to run the database server on a separate machine.

- The third component **Grafana** provides the graphical user interface. It allows you to plot historical data, and build (near) real-time dashboards for any metrics that are being written to the database. Grafana should be run on a computer that can access InfluxDB server(s) and the computer(s) you want to monitor from. Grafana runs a web server and you connect to it via your web browser. I have found Google Chrome to give superior performance for Grafana.

Each project is open-source with paid support. [Grafana.net](https://grafana.net) provide premium support for Grafana and [InfluxData](https://influxdata.com) provide the same for Telegraf and InfluxDB. InfluxData also maintain the other open-source packages Chronograf (similar to Grafana), and Kapacitor (used for alerts and data processing). I will not cover these here, only because I have do not have much experience with them, however both look promising. InfluxData also maintain a commercial version of InfluxDB with cluster support and admin tools aimed at larger scales.

These instructions will cover setup and configuration of:

- A **server** in a central location, which we will install **InfluxDB** and **Grafana**. This sever should be accessible from all PCs you want to monitor and all PCs you want to monitor from. It does not need to be at the station or a computer you use for monitoring.
- A collection of **client** computer you want to monitor, on which we will install **Telegraf**.

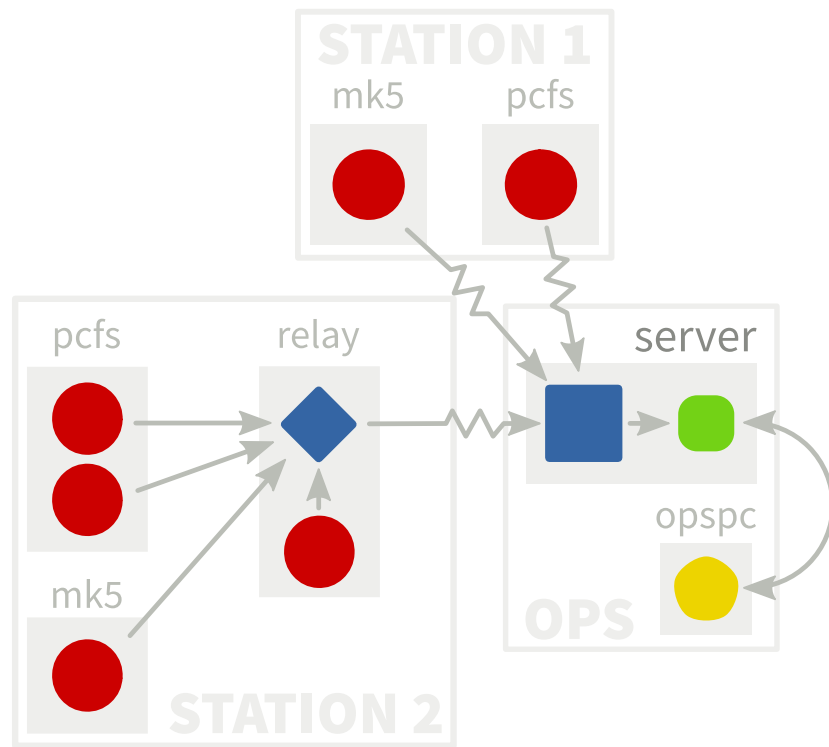


Figure 2: Example Setup. As in the introduction, red circles represent collectors; blue square, the database; green rounded square, the database clients; and yellow pentagons, the user interfaces. Arrows indicate the flow of data.

Figure 2 show schematic of the architecture we will setup.

If you monitor only one station from on site, then you can likely ignore a lot of these detail and let Telegraf write directly to the database like in Station 1 of the figure.

If you *do* have multiple stations, or you monitor from a remote location, you have a few choices of where to keep the database. The setup we will guide you through here is easy to install and manage, as well as less expensive, but maybe be less resilient to poor network conditions.

Telegraf can tolerate short to medium size network interruptions, by holding the latest points in memory until it can write to the database. This is the method used by Station 1 in Figure 2. The number of points Telegraf holds is configurable, limited by RAM/swap, so you can set it high enough to buffer an average outage.

If you write you own collector, you will need to do this yourself. We will give some example code of this later. There is also [InfluxDB-Relay](#), which can proxy collector's writes to the database. This method is used by Station 2 in Figure 2. All clients write to the relay, which presents the same interface as the database, which then forwards them on if it can, and buffers them in memory if it can't. We will not cover setup of the relay here.

If you find this setup is not adequate, you may need to run multiple database servers. See [Advanced Data-flow Models](#) for details.

Server

Installation

*The commands in this section should be run as **root**.*

For this setup, we assume you use a Debian based system for your server; however, all packages can run on different distributions and operating systems. If you are using a different distribution or operating system, follow installation documentation for [InfluxDB](#) and [Grafana](#)

Installation is managed through the systems package manager apt using dedicated repositories. The repositories are signed, so first import InfluxData's and Grafana's key GPG keys:

```
curl -sL https://repos.influxdata.com/influxdb.key | apt-key add -
curl -sL https://packagecloud.io/gpg.key | apt-key add -
```

Now, add the repositories to the package manager by creating the file `/etc/apt/sources.list.d/tig.list` with contents (uncommenting where necessary)

```
#####
## Grafana repo
## Use for all Debian/Ubuntu variants
deb https://packagecloud.io/grafana/stable/debian/ jessie main

#####
## InfluxData repo
## Uncomment the appropriate line

## Wheezy
#deb https://repos.influxdata.com/debian wheezy stable
#
## Jessie
#deb https://repos.influxdata.com/debian jessie stable
#
## For Ubuntu, replace xenial with appropriate codename
## if you dont know this run:
## source /etc/os-release && echo $VERSION
#deb https://repos.influxdata.com/ubuntu xenial stable
```

Now, update the package manager's database

```
apt-get update
```

and install the InfluxDB and Grafana

```
apt-get install influxdb grafana
```

InfluxDB will be configured to automatically start on boot.

To enable Grafana to start on boot:

- For systemd based distributions, ie. Ubuntu ≥ 15.04 or Debian ≥ 8 (jessie), use

```
systemctl daemon-reload
systemctl enable grafana-server
```

And start the server

```
systemctl start grafana-server
```

- For older SysVinit based distributions use

```
update-rc.d grafana-server defaults
```

And start the server

```
service grafana-server start
# or /etc/init.d/grafana-server start
```

InfluxDB and Grafana should now be installed and running on your server.

If you like, you can also install Telegraf on your this. This is useful for monitoring disk usage and load. If you don't need the VLBI fork, you can run `apt-get install telegraf` to get the standard version from the InfluxData repository.

You should now be able to access Grafana by entering `http://<server address>:3000` in a web browser. InfluxDB is also running an HTTP server on `<server address>:8083`, but you will not see anything there with browser.

Configuration

InfluxDB

For a complete overview InfluxDB's configuration see the [official documentation](#)

InfluxDB's configuration is located in `/etc/influxdb/influxdb.conf`. The one thing variable you may need to change is the location of the permanent storage. By default, this is set to `/var/lib/influxdb/data`. If this is not acceptable, it can be changed by setting the `dir` variable of section `[data]`.

By default your InfluxDB server will be accessible at port 8083 on your server. It is not configured with authentication or authorization. If you wish to allow access from the internet, you should add users and authorization.

If you do edit the configuration, be sure to restart the server.

InfluxDB is now ready to start accepting data.

Grafana

For a complete overview Grafana's configuration see the [official documentation](#)

Grafana's server configuration is located in `/etc/grafana/grafana.ini`. To begin with, you should not need to change this.

We will cover initial setup in [Working with Grafana](#).

Clients

Installation

*The commands in this section should be run as **root**.*

On any PC you wish to install the VLBI branch of Telegraf, for example your Field System PC, add the FS repository by creating the file `/etc/apt/sources.list.d/lupus.list` with contents

```
deb http://user:pass@lupus.gsfc.nasa.gov/fs/debian wheezy main
```

where `user` and `pass` are your username and password for the GSFC Field System repository.

Get David Horsley's GPG key:

```
apt-key adv --keyserver keys.gnupg.net --recv-keys 6E2CE741
```

then update the package database and install the package

```
apt-get update
apt-get install telegraf-vlbi
```

Telegraf is setup to run on startup.

Configuration

For full details on configuring Telegraf, see the [official documentation](#)

The Telegraf come with a range of useful plugins enabled by default, but you will need to set a few variables to get it to write to your database. This is done by editing the file `/etc/telegraf/telegraf.conf`.

General Telegraf settings

The first item is **Global tags**. These are tags that are added all measurements collected. It's recommended you at-least add a tag for the station. Do this by finding the line

```
# Global tags can be specified here in key="value" format.
[global_tags]
```

and add a tag, eg

```
station="gs"
```

Next you will find the general **Telegraf agent** configuration, beginning with

```
# Configuration for telegraf agent
[agent]
  ## Default data collection interval for all inputs
  interval = "10s"
```


This sets the default period for all collectors. If you're happy with a 10s default period leave this as is. This can be overridden on an input by input basis.

In the same section, you will also find

```
flush_interval = "10s"
```

this configures the rate at which Telegraf flushes its buffer to the database you may wish to make this shorter if your using the database for near real-time displays, or longer if you are concerned with network load.

Outputs

Now configure the **InfluxDB Output**

```
#Configuration for influxdb server to send metrics to
[[outputs.influxdb]]
...
urls = ["http://localhost:8086"]
```

and change localhost to the address (can be IP or DNS name) of your server setup in the previous section. In the same section you will also find a line specifying database:

```
database = "vlbi"
```

It is OK to leave it as this default. If you are configuring the standard Telegraf installation (non-VLBI) you should change this to match the above.

This completes the necessary configuration set of Telegraf, however you likely want to enable some extra inputs

Inputs

The default configuration file for Telegraf has a set of basic PC health input plugins such as CPU usage, Disk usage, Disk IO, Kernel stats, Memory usage, Processes stats, and swap usage.

To enable more specific plugins, uncomment them in /etc/telegraf/telegraf.conf.

For example, on your Field System PC, you will likely want to enable the **Field System** collector so find the [[inputs.fieldsystem]] section in telegraf.conf and remove the # prefix, ie

```
# Poll the Field System state through shared memory.
[[inputs.fieldsystem]]
  ## Rate to poll shared memory variables
  # precision = "100ms"
  ## Collect RDBE phasecal and tsys
  # rdb = false
```

You do not need to uncomment the settings unless you want to change the indicated default.

If you would like to enable the **metserver** collector, uncomment the `[[inputs.met4]]` section. You can also may also like to add extra tags and set a custom poll interval, e.g.:

```
# Query a MET4 meteorological measurements systems via metserver
[[inputs.met4]]
  ## Address of metserver
  address = "127.0.0.1:50001"
  interval = "1m"
  [inputs.met4.tags]
    location = "tower"
```

If you have a supported **antenna**, you can uncomment the `modbus_antenna` section

```
# Query an antenna controller using modbus over TCP.
[[inputs.modbus_antenna]]
  ## Collect data from a modbus antenna controller
  antenna_type = "patriot12m"
  # ip:port
  address = "192.168.1.22:502"
  #slave_id = 0
  ##Timeout in milliseconds
  #timeout = 10000
```

If you want **sensors** measurements, such as CPU temperature, install the `lm-sensors` package with `apt` and add the line to your Telegraf config:

```
[[inputs.sensors]]
```

If you RDBE's and wish to collect data from them, there is two route to get the information. One is via the Field System, which you have already seen. This get `tsys` and `pcal` data calculated by the Field System. The other is via `mutlicast`, which contains raw data used by the FS. To enable collection of this data, uncomment or add:

```
# RDBE UDP Multicast Listener
[[inputs.rdbe_multicast]]
  ## RDBE devices to listen. Can be an ID or a multicast address and port
  # eg.
  # device_ids = ["a","b","c","d"]
  # device_ids = ["239.0.2.40:20024"]
  device_ids = ["a","b","c","d"]
  ## Save Tsys, Pcal, and Raw measurments
  ## these are saved into the "rdbe_multicast_*" measurment
  save_pcal = false
  save_tsys = false
  save_raw = false
  save_statstr = false
```

```
## Extra tags should be added
## eg.
#[inputs.rdbe.tags]
# antenna = "gs"
# foo = "bar"
```

Note this collects a large amount of data, so you may want to use it sparingly.

Working directly with InfluxDB

We give a basic introduction here, but it is recommended you read [Getting Started](#) and [Data Exploration](#) in official InfluxDB documentation.

You should now have some data flowing into your database, so let's start accessing it.

On the server with InfluxDB installed run the command `influx`. This will start a command line client that connects to the database server over HTTP (at `localhost:8086` by default).

I recommend you first run, in the `influx` client, the command

```
precision rfc3339
```

which displays timestamps in RFC3339 time, rather than unix nanoseconds.

Metadata

Now in the `influx` client, run the command

```
show databases
```

You should see an output similar to

```
name: databases
name
----
_internal
vlbi
```

If there is no `vlbi` database, your Telegraf instances are not writing to the database. Check that Telegraf is running on your clients and that you set the `[[outputs.influxdb]]` section of your `telegraf.conf` file.

The `_internal` database stores statistics on InfluxDB.

If you do see `vlbi` in the list, set that as the database for the session with

```
use vlbi
```

Now try running the command

```
show measurements
```

This will give you a list of keys such as `cpu`, `fs`, `mem`. . . . These are the names of *measurements*, which are analogous to tables in a relational database.

Each measurement has a collection of fields and tags which are like columns in a table. The name of a field or tag is called its “key” and the content is its “value”.

The difference between fields and tags are that tags are indexed. This means queries on a tags are very fast. Tag values must be strings, whereas fields can host strings, booleans, integers, and floats, the latter two being 64-bits.

In InfluxDB terms, a measurement and with a set of specified tags is called a *series*. You can see all the series in the database with

```
show series
```

You should get a big list of measurement names and tag key-values pairs.

To see the fields keys and their associated type for a measurement, say `system`,

```
show field keys from system
```

If you have values for this measurement, the query should return

```
name: system
fieldKey      fieldType
-----
load1         float
load15        float
load5         float
n_cpus        integer
n_users       integer
uptime        integer
uptime_format string
```

Basic Queries

Let’s get some actual data out of the database. Run the command

```
select * from system where time > now() - 10m
```

This retrieves values from all field and tag values with measurement name “system” with timestamps after 10 minutes ago. You should get a result like

time	host	load1	load15	load5	n_cpus	n_users	uptime	uptime_format
----	----	-----	-----	-----	-----	-----	-----	-----
2017-02-13T20:35:30Z	fs1	0.1	0.11	0.13	4	18	258884	2 days, 23:54
2017-02-13T20:35:30Z	fs2	1.6	0.51	0.89	4	4	951751	11 days, 0:22
2017-02-13T20:35:40Z	fs1	0.08	0.11	0.12	4	18	258894	2 days, 23:54
2017-02-13T20:35:40Z	fs2	1.72	0.53	0.95	4	4	951761	11 days, 0:22
2017-02-13T20:35:50Z	fs1	0.07	0.11	0.12	4	18	258904	2 days, 23:55

```
2017-02-13T20:35:50Z fs2 1.97 0.56 1.03 4 4 951771 11 days, 0:22
:
```

Each row specifies a *point*. A point is uniquely identified by its timestamp and series (measurement and tag set). Note a series is not defined by the fields. New fields can be added and a field can be empty.

If you just want to get a single value, specify it in the select clause

```
select load1 from system where time > now() - 10m
```

Points with this value not set are ignored.

Caution: be mindful of how much data your query will return; InfluxDB will happily return multi-gigabyte results if you ask for it. If you did not include the `where time > now() - 10m` qualifier above, you will end up with every values in the measurement.

Note that the tag `host` in this query was treated just like another field. Let's instead make use of this tag by using the `group by` operation:

```
select * from system where time > now() - 10m group by host
```

This will give you a table of output for each value of "host" similar to

```
name: system
tags: host=fs1
time                load1 load15 load5 n_cpus n_users uptime uptime_format
----                -
2017-02-13T20:35:30Z 0.1   0.11  0.13  4      18     258884 2 days, 23:54
2017-02-13T20:35:40Z 0.08  0.11  0.12  4      18     258894 2 days, 23:54
2017-02-13T20:35:50Z 0.07  0.11  0.12  4      18     258904 2 days, 23:55
:
```

```
name: system
tags: host=fs2
time                load1 load15 load5 n_cpus n_users uptime uptime_format
----                -
2017-02-13T20:35:30Z 1.6   0.51  0.89  4      4      951751 11 days, 0:22
2017-02-13T20:35:40Z 1.72  0.53  0.95  4      4      951761 11 days, 0:22
2017-02-13T20:35:50Z 1.97  0.56  1.03  4      4      951771 11 days, 0:22
:
```

This "group by" feature is particularly useful when you want to compare tags against each other. If you only want only values from one host, specify it in the `where` command:

```
select * from system where time > now() - 10m and where host='fs1'
```

To limit the number of results returned by a query it is often useful to use the `limit n` command. For example

```
> select temperature from met limit 1
name: met
time                temperature
----                -
2012-03-21T18:13:00Z 18.7
```

Notice this gives the *first* result in the database because by default results are ordered by ascending time. You can override this by specifying `order by time desc` to get the reverse behaviour. The combination of these two commands is useful for getting the latest point in the database.

For example, to get the latest schedule reported by the Field System use the query

```
select schedule from fs order by time desc limit 1
```

Functions

Let's look at some other useful queries. A powerful feature of InfluxDB is its functions such as `mean`, `median`, `stddev`, `max`, `min`. You can see the full list in the [official documentation](#).

For example, if you have points in your `met` measurement you can get try the query

```
select mean(temperature) from met where time > now() - 5d group by station
```

This returns the mean temperature for each station over the last 5 days. The timestamp is the start of the window. **Important note:** if you did not include the `group by station` portion, this would get the mean of *all* stations over the last 5 days. In functions *tags are automatically merged unless you specify it or use a "group by"*.

You can also apply this function over windows. For example get the mean temperature at station 'gs' over 12 hour windows beginning 5 days ago.

```
> select mean(temperature) from met where time > now() - 5d and station='gs' group by time(12h)
name: met
time                mean
----                -
2017-02-09T12:00:00Z 0.8118450448931444
2017-02-10T00:00:00Z
2017-02-10T12:00:00Z 0.09354291840138866
2017-02-11T00:00:00Z 1.7690925925927894
2017-02-11T12:00:00Z 8.632935185183568
2017-02-12T00:00:00Z 7.424282407405593
2017-02-12T12:00:00Z 7.503481481481151
2017-02-13T00:00:00Z 6.117377314814672
2017-02-13T12:00:00Z 4.5796948438333756
2017-02-14T00:00:00Z -2.760842592592547
2017-02-14T12:00:00Z -1.1050233784917751
```

Again, timestamps are the start of the window.

Note, in my example, there is a blank region, this is because no data was collected in this window. By default, if there are no points in a particular window, the function outputs `null` (the absence of data). This can be overridden by with the `fill` option, for example to use linearly interpolation use `fill(linear)`:

```
> select mean(temperature) from met where time > now() - 5d and station='gs' group by time(12h) fi
name: met
time                mean
----                -
2017-02-09T12:00:00Z 0.8029195834044832
2017-02-10T00:00:00Z 0.4482312509029359
2017-02-10T12:00:00Z 0.09354291840138866
2017-02-11T00:00:00Z 1.7690925925927894
2017-02-11T12:00:00Z 8.632935185183568
2017-02-12T00:00:00Z 7.424282407405593
2017-02-12T12:00:00Z 7.503481481481151
2017-02-13T00:00:00Z 6.117377314814672
2017-02-13T12:00:00Z 4.5796948438333756
2017-02-14T00:00:00Z -2.760842592592547
2017-02-14T12:00:00Z -1.0162018284920666
```

Other arguments are none, null, previous, or any constant. [See the documentation.](#)

Another function is `max`. This is a different kind of function. While `mean` is an “aggregation”, meaning it aggregates the data in a window; `max` is a selector, meaning it selects a value from the window. The in a selector the timestamp is preserved.

For example, to get the maximum temperature in the database and the station that recorded it

```
select max(temperature),station from met
```

We have just covered the basics of InfluxDB and there are more features to learn. These include [more functions](#), [database management](#), [sub-queries](#), [continuous queries](#), [retention policies](#). The documentation is thorough and accessible.

Working with Grafana


Grafana is well documented. We will get you started here, but we recommend reading the [full documentation](#)

To access Grafana, open a browser and direct it to `http://<server>:3000` (unless you changed the default port)

The first time you login to Grafana, the default username and password is “admin” (for both). You will be prompted to change this.

Adding the Database

To begin with, you will need to add your database to Grafana. Do this by


1. Select item  > **Data Sources** from drop-down menu in the top left.
2. Press **Add data source**

3. From the “Type” drop-down menu, select **InfluxDB**
4. Set a **name**, eg “influxdb-vlbi”
5. Check **default**
6. Enter the **address** of your InfluxDB server. This is likely `http://localhost:8086` if Grafana and InfluxDB are hosted on the same machine.
7. Set access to **proxy**. This means the Grafana server will poll the database. This makes using Grafana from the Internet easier.
8. Set Database to **vlbi**

Everything else you can leave as-is. Press **add** to finish.

Creating a Dashboard

A *dashboard* is single page with a collection of *panels*.

- To create a dashboard select the menu item  > **Dashboards** > **New**.
- You will be presented with a new empty page and options for a new panel. Panels are created in rows and you have an empty row. **Create a new Graph panel** by selecting it from the list. This will create an empty panel. It will have no data points because we haven't given it a query.
- **Edit the panel** by
 - Getting the *panel menu* by pressing the panel title,
 - then selecting “Edit”.
- This will bring up the graph options. By default you should be on the “Metrics” tab of the Graph editor with the list of queries for this panel. **Open the query editor** by pressing the text of the query.
- Choose your measurement you want to query by pressing “**select measurement**”. This will give you a drop-down menu of all the measurements in the database. For this example, let's **select the “cpu” measurement**. You can either begin typing the name or select it with the mouse.
- Now choose a field by **pressing value** in `field(value)`. For this example, let's **choose usage_user**. Again, you can select it by with the mouse or begin typing pressing enter to complete.

You should now see a time series plot of the CPU usage.

If multiple hosts are writing to this field this graph will be misleading. Notice Grafana has automatically added a `mean` function to your query along with a `group by time($interval)`.

The `$interval` part is a Grafana variable which scales with the time range you are viewing. This is a very convenient feature, but recall that InfluxDB groups tags together when a function is used. This means what is displayed is the average of *all* hosts, which is probably not particularly useful.

- To plot the multiple hosts separately, add group by host by **pressing “+”** at the end of the `GROUP BY` row of the query editor and selecting **`tag(host)`**. You should now see a graph for each host that is writing to that field.
- The automatic names of the host are fairly ugly. Let’s add aliases to the graphs by **entering `$tag_host usage`** in the `ALIAS BY` field. `$tag_host` is an automatic variable added by Grafana and takes the value of the tag host.
- The unit of `usage_user` is percent of CPU time, so let’s add this to the axis. **Select the “Axes” tab** in the panel editor window. Under “Left Y”, **select `Unit > none > percent (0-100)`**
- Let’s add a better title to the panel by selecting the “General” tab and entering, say, “CPU usage”.
- Return to the dashboard by pressing “Back to dashboard” on the top menu-bar or by pressing Escape.
- Experiment with exploring the data.
 - Try zooming-in to a time range by clicking and dragging in the Graph panel.
 - Select a time range from the top right.
 - Try using the keyboard to navigate. See list of keyboard shortcuts by pressing “?”.

Now let’s try making a near real-time display.

- Open the time editor by pressing the time button in the top right of the page. Enter
 - From: `now-5m`
 - To: `now`
 - Refreshing every: `5s`
 then press “apply”

You may notice if you see your graphs disappear when you zoom into a short time range. This is because our query is returning some empty windows (remember we are using `group by time($interval)`) and we are using `fill(null)`. Grafana’s default behaviour is to break lines on ‘null’. This is handy to see when data stopped but of course, if your data is surrounded by empty windows, you’re not going to see anything!

Fortunately, Grafana has a way to deal with this. In the “metrics” tab of a Graph panel, there is a “Group by time interval”. This allows you to set a limit on the size of the `$interval` variable. So you could put in `>10s` if you’re sampling at 10s intervals. This can also be set to a default for the whole data source.

The other ways of dealing with this are:

1. Changing the DB query to fill with something other than ‘null’. This is done with either `fill(none)` which just doesn’t return empty windows, or by `fill(x)` which fills empty windows with value `x`.
2. Changing the graph panel’s behaviour on nulls. This is found by under the “Display” tab of a Graph panel.

The “Group by time interval” setting is probably the best way to deal with it unless you have some less common need.

Importing Dashboards

You can also import and export dashboards in JSON format.

If you have met data in your database you can try try importing our prepared dashboard. This dashboard uses some more complex features to show the full range of data, which is particularly useful for seeing anomalies.

To import this dashboard:

- Download [met-dashboard.json](#)
- In Grafana, from the Dashboards dropdown menu, select import
- Select “Upload .json File” and find where you save the file
- Select your data source if you need to.

If you would like to import historical data to the “met” measurement you can try using the [Weather Log Importer](#). You will need [Go](#) installed.

Other topics

Other topics that are worth learning about in Grafana but we haven’t covered here are:

- Other panel types and adding new ones
- Users, groups and permissions
- Templating
- Annotations

Using InfluxDB with other tools

This section is a work-in-progress

As well as Grafana, you can also easily access the data in the database via your own tools. There is probably already a client library available for your favorite programming language. Have a look at the [list of client libraries](#).

If you are building real-time plots, you can get the latest points by using the query (for example)

```
select log from fs order by time desc limit 1
```

Python

Using [InfluxDB-Python](#) and with [pandas](#) has proven particularly powerful.

The InfluxDB-Python has helper functions to import your queries as as time-series Dataframes. You can then use all the tools of Pandas such as interpolating two series together and plotting via matplotlib.

For example, this script get Azimuth and Elevation from the antenna measurement and the tsys data from `fs_rdbe_tsys` and plot the average in bins over the az-el plane.

Note *there is currently a bug in the python library which results in queries being truncated to 10000 points*

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import influxdb

client = influxdb.DataFrameClient(host='localhost',
                                  port=8086,
                                  database="vlbi")

TIME_RANGE = "time > now() - 60d"

results = client.query(
    "select Azimuth1, Elevation1 from antenna where %s" % TIME_RANGE,
    chunked=True, # Currently does not work in 1.2
)
azel = results['antenna'].groupby(level=0).first()
# Map Az to [-180, 180]
azel["Azimuthreal"] = np.mod(azel["Azimuth1"]+180, 360)-180

results = client.query(
    "select mean(chan_0010) from fs_rdbe_tsys \
    where rdbe = 'b' and %s group by time(1s) fill(none)" %
    TIME_RANGE,
    chunked=True, # Currently does not work in 1.2
)
tsys10 = results['fs_rdbe_tsys'].groupby(level=0).first()
tsys10[tsys10['mean'] > 1000] = np.nan;
tsys10.plot()
plt.savefig("tsys10.png")
```

```
#Concat and forward fill
s = pd.concat([azel, tsys10], axis=1).ffill()

ax = s.plot.hexbin(x="Azimuth1", y="Elevation1",
                  C="mean",
                  reduce_C_function=np.mean,
                  gridsize=70,
                  cmap=plt.cm.YlOrRd
                  )
ax.set_xlabel("Azimuth")
ax.set_ylabel("Elevation")
plt.savefig("heatmap.png")
```

Creating new collectors

InfluxDB takes in data over HTTP. This makes it easy to write client libraries with any programming language.

There is probably already a client library available for your favorite programming language. Have a look at the [list of client libraries](#).

Shell

A very basic option is to use the `curl` program.

```
#!/bin/sh
##
DB=station
PRECISION=s # or [n,u,ms,s,m,h]; determines the meaning of the timestamp

URL="http://localhost:8086/write?db=$DB&precision=$PRECISION"

DATA='weather,station=ashington temperature=35 pressure=1024.5 humidity=95.1 1484842058'

curl -i -XPOST $URL --data-binary $DATA
```

The contents of `$DATA` are in the InfluxDB Line Protocol. This is a text based format for writing points to InfluxDB and takes the form

```
<measurement>[, <tag_key>=<tag_value>, ...] <field_key>=<field_value>[, ...] [<timestamp>]
```

Each line, separated by the newline character `\n`, represents a single point in InfluxDB. For full details on the InfluxDB line protocol see the [Official Documentaiton](#).

This example writes a point to of measurement type “weather” with tag “station” set to “ashington” fields “temperature”, “pressure” and “humidity” set to floating point values at the time 2017-01-19T16:07:38+00:00 (1484842058 unix time)

In this example, the time stamps are in UNIX time (seconds since 1970-01-01T00:00:00Z, not counting leap seconds). The meaning of the time stamp is determined by the `PRECISION` variable which has been set to “s” for seconds. If, for example `PRECISION` is set to `n` for nanoseconds (the default), the time stamp is interpreted as UNIX nano seconds. In general it is best to use the lowest precision you can, as this improves the performance and compression of the database.

If you do not include the timestamp, the servers time is used with nanosecond precision.

Go

Go has a client library written and supported by the InfluxDB team. See the [InfluxDB Client](#).

For example usage, see the [Weather Log Importer](#)

Telegraf

Alternatively, you can add your own plugins to Telegraf which is itself it written in Go.

Creating input plugins for Telegraf has the advantage that your connection, buffer and configuration are all managed for you. It also makes your setup more easy to manage and, since Telegraf supports multiple output types, so you won't be tightly coupled to InfluxDB.

You will need to have Go installed and setup on some computer, although not necessarily a Field System pc, or even Linux.

If you want to add your own collectors to the VLBI branch of Telegraf, start by getting the main source

```
go get github.com/influxdata/telegraf
```

then add the VLBI repository and checkout the VLBI branch

```
cd $GOPATH/src/github.com/influxdata/telegraf # $GOPATH=~/.go if not set.  
git remote add lupus http://lupus.gsfc.nasa.gov/fs/src/telegraf.git  
git fetch lupus  
git checkout vlbi
```

If you want to build Telegraf with Field System support, you will need to get the Field System Go library:

```
cd $GOPATH/go/src  
git clone http://lupus.gsfc.nasa.gov/fs/src/fs-go.git fs
```

Input plugins are stored in `plugins/inputs`. You will likely find it easiest to copy a preexisting plugin as a base. The `met4` plugin is particularly simple

```
cd ~/go/src/github.com/influxdata/telegraf/plugins/inputs
cp -r met4 myplugin
cd myplugin
mv met.go myplugin.go
```

And edit `myplugin.go`. Add your plugin to the import declaration in `telegraf/plugins/inputs/all/all.go`.

To build Telegraf, run

```
cd /path/to/telegraf
make
```

Which will create a statically linked binary at `$GOPATH/bin/telegraf`. If you are cross-compiling this for a Field System PC, instead run:

```
GOOS=linux GOARCH=386 make
```

You can copy the binary `$GOPATH/bin/telegraf` to the FS pc.

To test your plugin, create a sample configuration file and run it

```
telegraf --input-filter myplugin config > telegraf.conf
telegraf --config telegraf.conf -test
```

To build a release Debian package:

```
./scripts/build.py --package --version="1.1.1-vlbi-0.2.4" --platform=linux --arch=all --release
```

Python

There is a 3rd-party supported python library for dealing with InfluxDB connections at [InfluxDB-Python](#).

To install, use Python's package manager (probably as root):

```
pip install influxdb
```

For a usage demonstration see the [included example](#) or the [official examples](#)

Advanced Web Setup

If you wish to make Grafana accessible via the open Internet, you have some options:

Directly via port 3000. This is the default setup and perfectly fine. You may need your network administrator to open this port a firewall for you.

A slightly nicer way is allow access directly to Grafana via port 80, HTTP's default. To do this, give Grafana permissions to bind to privileged ports with

```
sudo setcap 'cap_net_bind_service=+ep' /usr/sbin/grafana-server
```

then set `http_port = 80` in `/etc/grafana/grafana.ini`.

Again, you may need your network administrator to open this port a firewall for you.

Reverse Proxy

A third option, and the most versatile, is to use another web server as a reverse proxy. This is useful if you already run a web server on your network and want Grafana to appear as a subdirectory on that server. The web server and Grafana do not need to be on the same computer

No matter which web server you use, you will need tell Grafana where it is located. Do this by setting, in `/etc/grafana/grafana.ini`,

```
root_url = https://my.external.website.edu/grafana
```

Apache 2

I haven't tested this, your mileage may vary.

You will need to activate the proxy module for Apache. As root run

```
a2enmod proxy_http
```

Next add the following to you Virtual Host configuration for the external site, likely in `/etc/apache2/sites-enabled/default`

```
ProxyPass /grafana http://internal.grafana.location:3000/
ProxyPassReverse /grafana http://internal.grafana.location:3000/
```

When you're done, reload the configuration

```
service apache2 reload
```

Nginx

For Nginx, find the configuration for your external site, likely `/etc/nginx/sites-available/default`.

In the root level, add Grafana as an upstream server:

```
upstream grafana {
    server internal.grafana.location:3000;
    keepalive 15; # Not necessary may give performance gains
}
```

Next, find the configuration for the site server, starting with

```
server {
    listen 80; # Or 443 for HTTPS
```

...

And add

```
location /grafana/ {
    proxy_pass http://grafana/;
    proxy_redirect      default;

    # Not necessary may give performance gains
    proxy_buffering on;
    proxy_buffers 8 128k;
    proxy_buffer_size 128k;

    proxy_set_header    Host            $host;
    proxy_set_header    X-Real-IP       $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_http_version  1.1;
    proxy_set_header    Connection "";
}
```

Check your configuration is valid

```
nginx -t
```

And reload

```
nginx -s reload
```

HTTPS

If you wish to open Grafana or InfluxDB to the Internet, it is advisable to configure HTTPS. This is not documented here.

Advanced Data-flow Models

If you have multiple station or monitor from a remote location, you have a few choices of where to keep the database. If you do not, you can skip to [Installation](#).

Run a central database (Recommended)

This is easier to setup and manage, as well as less expensive. In this model, all stations and client write to the single central database at the operations center. See the figure

Telegraf will tolerate network interruptions, to some extent, by holding the latest points in memory. The number of points it holds is configurable, so you can set it high enough to buffer an average outage.

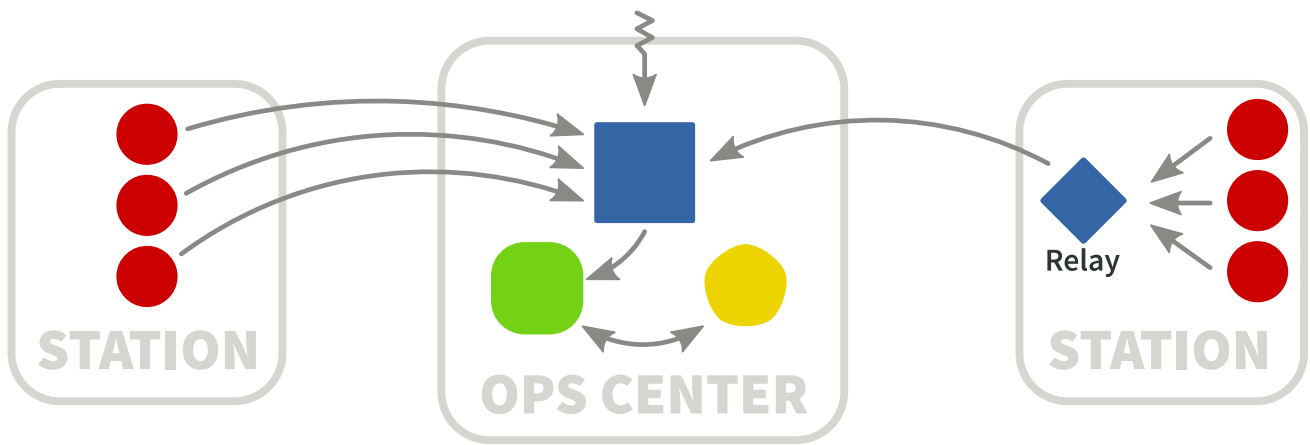


Figure 3: Single Central Database model. As in the introduction, red circles represent collectors; blue squares, the database; green rounded squares, the database clients; and yellow pentagons, the user. Arrows indicate the flow of data.

If you write your own collector, you will need to do this yourself. There is a program called **InfluxDB-Relay**, which can proxy collector's writes to the database. All clients write to the relay instead of the remote server, which then forwards them on if it can, and buffers them in memory if it can't. This may be a good option if you are concerned about some client running out of memory during a network outage.

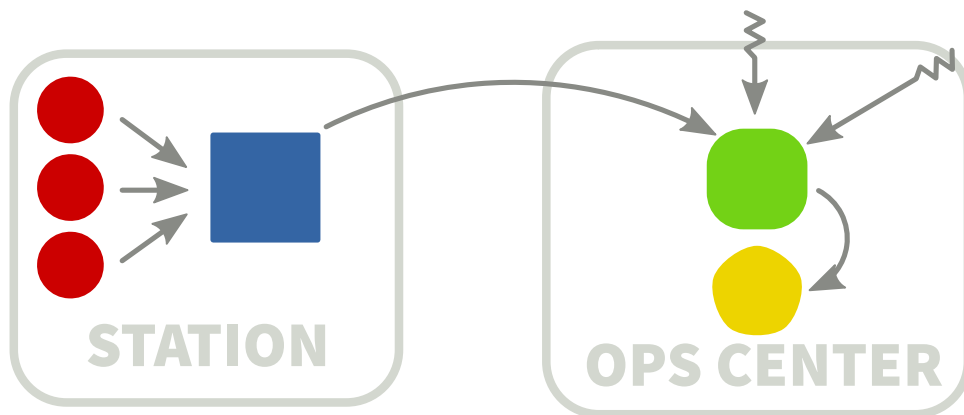


Figure 4: Decentralized model.

Run a database at each station

This has the advantage that if the network connection is lost, clients will continue to write to their local database. It is also advantageous if there are local operators that wish to look use the data.

This has the disadvantage that you will need a system capable of running the database and storing the data at each station. It can also be slow when you are querying the database remotely.

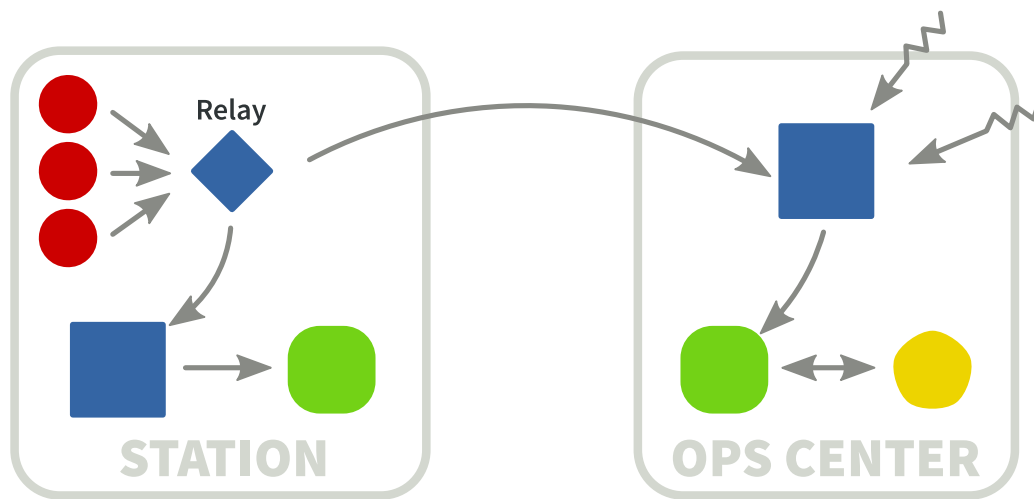


Figure 5: Multiple Database model.

Run databases at stations and control center

The setup would be fairly involved, but you get the best of both options. You can configure “retention” policies at the stations, so only a certain period of records are kept there. [InfluxDB-Relay](#) can be used to write to local and remote databases at the same time moderate small outages. For large outages, a program would need to be run to sync the databases.

