Administrator Handbook

NESTOR® Version

Georg Bahlon
Natalia Hoesl
DI Dr. Andreas Lang-Nevyjel
Bernhard Mallinger

Administrator Handbook: NESTOR® Version

by Georg Bahlon, Natalia Hoesl, DI Dr. Andreas Lang-Nevyjel, and Bernhard Mallinger

Release: 1.5 -This Document was generated 2015-05-27

Abstract

This document is the official documentation for **NESTOR®**. It explains general concepts of the software, gives an overview of it's components and walks you through various installation and administration tasks, the appendix documents and explains the stable part of the API.

Please write to **<nestor@init.at>** to contact our developer and support team.

1. Introduction 1
1.1. Target audience 1
1.2. Prerequisite 1
1.3. Symbol description 1
2. About 3
2.1. What is **NESTOR®** and what is it for? 3
2.2. Much more than just a webfrontend! 3
2.3. System Requirements 4
2.3.1. Common requirements 4
2.3.2. Database requirements 5
3. Installation 7
3.1. Supported operating systems 7
3.1.1. Repository access 7
3.1.2. Two different ways to install the software 7
3.1.3. Automatically installation with *install_icsw.py* 7
3.1.4. Manually installation 8
3.1.5. Repository 8
3.1.6. Repository directories 8
3.1.7. Debian repositories 9
3.1.8. Ubuntu repositories 9
3.1.9. OpenSUSE and SLES repositories 9
3.1.10. CentOS and RHELrepositories 10
3.1.11. Database setup 11
3.1.12. Portnumber for accessing the webfrontend 13
3.2. Installation 14
3.2.1. Install icsw packages 14
3.3. Installation on virtual machine 14
3.3.1. KVM libvirt/qemu 15
3.3.2. VMware 15
3.4. Upgrade software packages 15
3.5. Server control 16
3.5.1. Show server status 16
3.5.2. icsw commands 16
3.6. License administration 18
3.6.1. Login without licenses 18
3.6.2. License package structure 19
3.6.3. Upload license keyfile 19
3.6.4. Show license status 20
3.6.5. License time periods 21
3.6.6. License parameter limitations 21
3.6.7. License violation 21
4. Core concepts 23
4.1. Services and components 23
4.1.1. We do not need to reinvent the wheel 23
4.1.2. Common components 23
4.1.3. Monitoring components 23
4.1.4. HPC components 24
4.1.5. Clusterdatabase 24
4.1.6. Dataflow between webfrontend and database 25
4.2. Run important services 25
5. Webfrontend 27
5.1. First connection 27
5.2. Areas 27
5.2.1. Menu area (1) 28
5.2.2. Sidebar device tree (2) 29
5.2.3. Main area (3) 31
5.2.4. Cluster server information 32
6. Concept of operation 35

6.1. Working with the web front-end 35
6.1.1. Preselection and submenu 35
6.1.2. Preselection and homebutton 36
6.1.3. Two-stage selection 37
6.2. Input field filter/regex, hide/show column, pagination and sorting 37
6.2.1. Input field filter/regex 37
6.2.2. Hide/show columns 38
6.2.3. pagination 39
6.2.4. Sort column 39
6.3. Reversion 40
6.3.1. History overview 40
6.3.2. Revert to former version 41
7. User and group management 43
7.1. Create user or group 43
7.2. Create group form 43
7.3. Create user form 44
7.4. Permission System 45
7.4.1. Permission 45
7.4.2. Permission level 47
8. Package installation 49
8.1. Preparing installation for package install 49
8.1.1. Server settings 49
8.1.2. Client settings 49
8.1.3. Server config file /etc/sysconfig/package-server 50
8.1.4. Client config file /etc/sysconfig/package-client 50
8.2. Install packages 51
8.2.1. Install packages using package manager 51
8.2.2. Install packages using directory upload 53
8.3. Delete packages 54
9. RMS - Resource Management System 55
9.1. Introduction of RMS 55
9.1.1. Environment variables 55
9.1.2. Installation of RMS 56
9.1.3. RMS web front-end 56
9.2. Job management system in SGE 58
9.2.1. SGE commands 59
9.2.2. Job submission via command line 61
9.2.3. Job submission via web front-end 61
10. Virtual Desktop 63
10.1. Prerequisites 63
10.2. Connect to virtual desktop 64
10.3. Change settings 65
11. Cluster setup 67
11.1. Basic requirements 67
11.1.1. Needed system packages 67
11.1.2. Install required system services 67
11.1.3. Configuration of system services 68
11.1.4. Needed **NESTOR®** packages 68
11.1.5. Install required **NESTOR®** packages 69
11.2. Node installation requirements 69
11.2.1. Create the partition 69
11.2.2. Create the kernel 70
11.2.3. Create the image 71
11.3. Device and network configuration 72
11.3.1. Two different networks and peering 72
11.3.2. Server and node configuration 74
11.3.3. Nodeboot 75
12. Extensions 79

12.1. Shared script directories 79
13. Debugging and Error hunting 81
13.1. General information 81
13.2. Show errors 81
13.3. Node information 82
13.4. Logging 82
13.4.1. Directories for log files 82
13.4.2. Automatic log mail delivery system 82
13.4.3. icsw logwatch 83
13.4.4. Service communication ports 83
14. Frequently Asked Questions 85
14.1. FAQ 85
Glossary 95

Chapter 1. Introduction

1.1. Target audience

Before you start to dive deep into the documentation, we think it is fair to let you know if you get the right information out of the document or not. This documentation is intended for system administrators who want to get into monitoring and cluster management. It is also intended for user who only have to operate with the software but don't do any configurations.

The handbook provides also some background information about **LINUX®** commands in general, i.e. in context with package installation.

It does not deal with general monitoring themes or basic principles of monitoring or cluster management. If you wish to learn something about that, please save your time and look for a more suitable document in the world wide web or in your local specialised bookstore.

1.2. Prerequisite

Below list shows you which prerequisite of users and administrators should be fulfiled to get into monitoring or cluster management with software by **init.at ltd.**.

- Experience with **LINUX®** in general
- Experience with the **LINUX®** command line, e.g **bash**, **zsh** or other
- Experience with standard html browser
- Experience with network settings

1.3. Symbol description

For easier document handling on some place you will notice small icons with the following meanings:

Table 1.1. Symboltable

	**Link inside the documentation**
	**Mailto Link**
	**Internet http link**
	**Link to the glossary**
	**Mark some very important statement**
	**Moving mouse to menu with given name**
	**Work on this content is in progress**
	**Left mouse click**
	**Right mouse click**
	**Doublemouse click**

Chapter 2. About

2.1. What is NESTOR® and what is it for?

Purpose of **NESTOR®** is the HPC Cluster Management. It allows administrators to setup and manage a huge number of nodes in clusters or even more than one cluster at once.

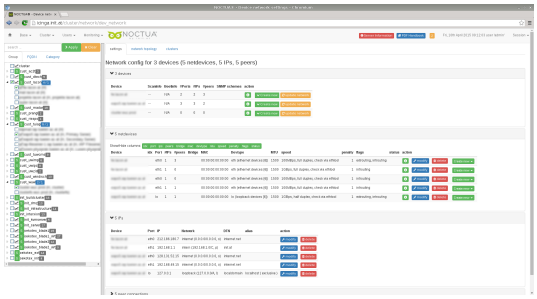
Especially in cooperation with **Monitoring** it offers both, management and monitoring of your cluster, two essential parts every admin of HPC sooner or later have to think about.

2.2. Much more than just a webfrontend!

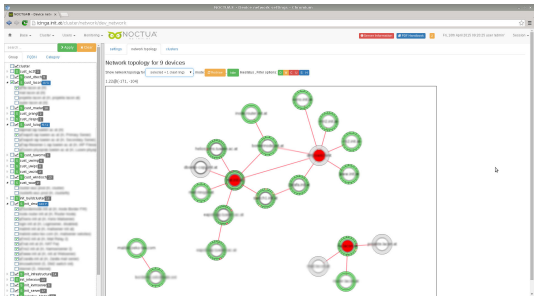
Even if the main task of **NESTOR®** is to ease configuration and administration of icinga, there are some special and unique features icinga and other software solutions do not provide.

Following list contains the exclusive components that makes our software unique and unreachable on software market:

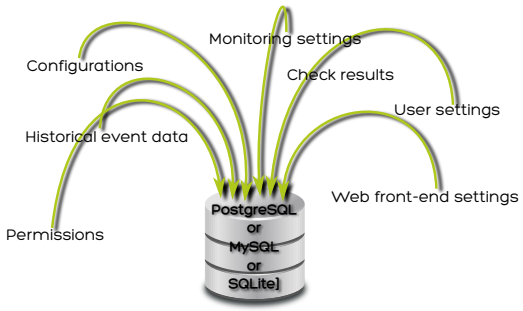
Web front-end - Userfriendly and easy to use web front-end to access to all data, configurations and settings.



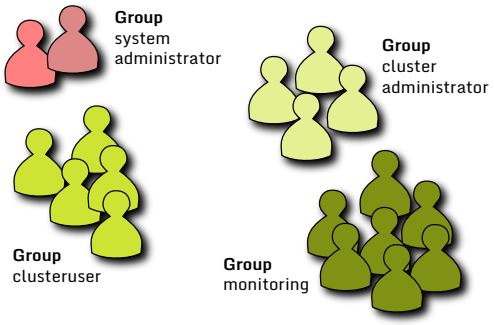
Peering - Possibility to connect monitored devices to peers. Displays whole network topology of connected devices.



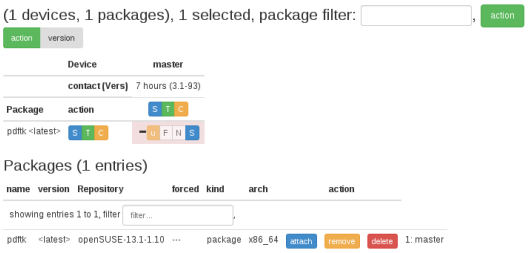
Central database - One central storage for all data like configs, settings, logging data user data and much more.



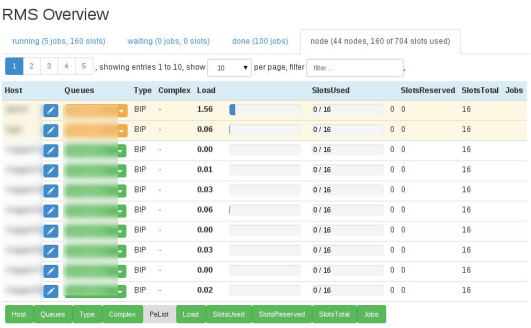
User Management - Administrator tool to manage groups, users and corresponding permissions.



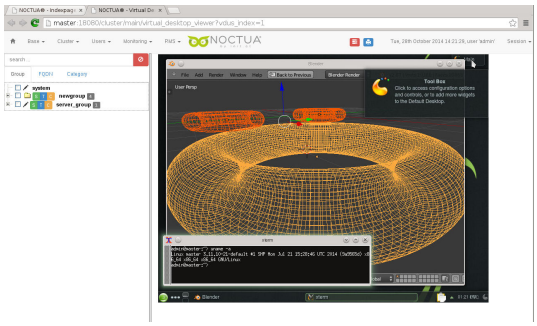
Packageinstall - Option to install system packages over the web front-end. (Operating system package management will take care for the exact procedure of installation.)



RMS - Resource Management System - Job management system for user to easy handle their jobs via the web front-end.



Virtual desktops - Automatic virtual desktop management system. Full access to remote machines displayed inside of your favorite browser.



Netboot - Facility to install and control nodes over the network

2.3. System Requirements

In this section you will find out about what technical requirements **NESTOR®** has.

2.3.1. Common requirements

Like every other software, **NESTOR®** has also certain system requirements. Because **NESTOR®** is open source Software, everybody who has enough programming knowledge could be able to port it to other open source software systems.

The good news: You don't have to port anything if you already use one of the following LINUX distributions:

- Debian
- Ubuntu
- CentOS
- Opensuse
- SLES

For exact versions please take a look into Installation Chapter.

2.3.2. Database requirements

Monitoring configurations are stored in databases for faster access and therefore faster reaction time and further more flexible administration of data.

NESTOR® is using *Django* as its database interface so every database which is compatible with Django can be used. The recommended Database is PostgreSQL (mostly due to license issues)

MySQL database is not supported any more.

Chapter 3. Installation

3.1. Supported operating systems

Software packages are available for following operating systems:

- Debian Squeeze (6.x)
- Debian Wheezy (7.x)
- Ubuntu 12.04
- CentOS 6.5
- openSuSE (12.1, 12.3, 13.1)
- SLES 11 (SP1, SP2, SP3)

It might be very well possible to get the software up and running on different platforms - but this type of operation is not tested and supported at all.

3.1.1. Repository access

There is no public access to our repository directories, therefore first, you have to contact us to get a valid **LOGINNAME** and **PASSWORD**.

> Contact us by mail either **support@init.at** or **nestor@init.at** or just call **+43-1-522 53 77** to get your individual access data.

After receiving you access data you are able to use below mentioned repositories.

3.1.2. Two different ways to install the software

To install the software on your operating system there are two ways to go.

You can either automatically install the software by downloading and running our **install_icsw.py** install script or manually by adding the below listed repositories with your individual access data and use your package manager as usual.

The automatically installation via script is well recommended because it is very comfortable and handles most of installation scenarios.

> ⚠Please only do a manually installation if you really know what you are doing.

3.1.3. Automatically installation with *install_icsw.py*

In order to use the install script, first you have to contact us to get your individual access data.

1. Download the script named `install_icsw.py` from our download portal.
2. As user **root** run the script with your repository access data as follows:

`install_icsw.py` [-u *USERNAME*] [-p *PASSWORD*] [-n *CLUSTERNAME*]

The script does following:

1. Determine which operating system is running
2. Add the necessary repositories with the required access data in your operating systems repository directory
3. Refresh your package cache
4. Install the software
5. Automatically integration of valid license files to the system

3.1.4. Manually installation

After receiving your individual access data do following steps to install the software manually:

1. Add suitable repositories for your operating system
2. Refresh your package cache with your os package manager
3. Install the software (for details see section installation)
4. Integrate your received or downloaded license files

3.1.5. Repository

There are two main repositories you have to deal with to install the software.

Repositories are available for stable (2.5) and for master (devel) versions of above mentioned operating systems. The operating system running on your hardware and the version of the software you want, determine which repository configuration you must use for your package manager.

icsw-devel — Repository for **latest** releases of **i**nit **c**luster **s**oft **w**are.

The current developer Version, containing the newest functions and modules. Very fast update and change cycle due to active development. Sometimes a bug could slipped in but usually it works fine. From time to time it will be merged into stable.

icsw-2.5 — Repository for **stable** releases of **i**nit **c**luster **s**oft **w**are.

The current stable version for productive environment. Most features and functions are included and there are no knowing bugs.

Based on the above mentioned operating system, repository and desired software version, resulting repositories can be added.

3.1.6. Repository directories

There are two different ways to add new repositories for monitoring software by **init.at ltd.** to the operating system. It can be added all at once in one central file or in a repository directory. For each operating system there are special repository directories.

Table 3.1. For debian based systems

Debian wheezy	`/etc/apt/sources.list.d/`
Debian squeeze	`/etc/apt/sources.list.d/`
Ubuntu 12.04	`/etc/apt/sources.list.d/`

Table 3.2. For suse based systems

OpenSUSE	`/etc/zypp/repos.d/`
SLES	`/etc/zypp/repos.d/`

Table 3.3. For red-hat based systems

CentOS	`/etc/yum.repos.d/`

3.1.7. Debian repositories

One file repository

Following you can see some examples of `source.list` content. This are the lines you must add to your `/etc/apt/sources.list`

icsw-devel

The relevant parts for deb based package manager looks like this for **devel** version and **wheezy** :

```
deb     http://LOGINNAME:PASSWORD@www.initat.org/cluster/DEBs/debian_wheezy/
icsw-devel wheezy main
```

icsw-2.5

The relevant parts for deb based package manager looks like this for **stable** version and **wheezy** :

```
deb     http://LOGINNAME:PASSWORD@www.initat.org/cluster/DEBs/debian_wheezy/
icsw-2.5 wheezy main
```

3.1.8. Ubuntu repositories

For ubuntu 12.04 you can also add repositories either in one file, into `/etc/apt/sources.list` or in repository directory `/etc/apt/sources.list.d`

One file repository

icsw-devel

```
deb     http://LOGINNAME:PASSWORD@www.initat.org/cluster/DEBs/ubuntu_12.04/
icsw-devel precise main
```

icsw-2.5

```
deb     http://LOGINNAME:PASSWORD@www.initat.org/cluster/DEBs/ubuntu_12.04/
icsw-2.5 precise main
```

3.1.9. OpenSUSE and SLES repositories

Debian and ubuntu use an other package manager than CentOS, OpenSUSE or SLES. For that reason, on rpm based operating systems `sources.list` does not exist. Rather there are a few files for repository management not only one. All relevant repository files stays in the directory `/etc/zypp/repos.d/` .

SUSE 13.1 and icsw-devel version

```
[cluster_devel_remote]
name=cluster_devel_remote
enabled=1
autorefresh=0
baseurl=http://LOGINNAME:PASSWORD@www.initat.org/cluster/RPMs/suse_13.1/icsw-devel
type=rpm-md
```

SUSE 13.1 and icsw-2.5 version

```
[cluster_devel_remote]
name=cluster_devel_remote
enabled=1
autorefresh=0
baseurl=http://LOGINNAME:PASSWORD@www.initat.org/cluster/RPMs/suse_13.1/icsw-2.5
type=rpm-md
```

Direct links to repositories

Alternative it's possible to download repositories direct from internet instead of editing files manually. There are two URLs you can get repositories from.

- For **deb** repositories look at [http://www.initat.org/cluster/DEBs/]
- For **rpm** repositories look at [http://www.initat.org/cluster/RPMs/]

Don't forget to request for access data to the repository directories otherwise you can not access it.

3.1.10. CentOS and RHELrepositories

Repository directory is `/etc/yum.repos.d/` . Place your desired *.repo files inside this directory, do a **yum check-update** and you are ready to install the software.

CentOS 6.5 icsw-devel

```
[initat_cluster]
autorefresh=1
enabled=1
type=rpm-md
name=initat_cluster
baseurl=http://LOGINNAME:PASSWORD@www.initat.org/cluster/RPMs/rhel_6.2/icsw-devel
```

CentOS 6.5 icsw-2.5

```
[initat_cluster]
autorefresh=1
enabled=1
type=rpm-md
name=initat_cluster
baseurl=http://LOGINNAME:PASSWORD@www.initat.org/cluster/RPMs/rhel_6.2/icsw-2.5
```

3.1.11. Database setup

Before continuing the server installation it is worth to say something about the database because it is one of most important parts of the software. It contains all settings, configurations, users and much more in one single database. This is the reason why it is easy to migrate or backup data and this is why it lowers the monitoring effort in comparison to a simple icinga installation.

POSTGRESQL

After a basic installation of the server, normally only a SQLite database exists. To start with monitoring it suffices completely but it must be mentioned that due to database limitations of SQLite we recommend you to switch to a better database solution e.g. SQL.

So before installing the software, two scenarios are possible.

1. SQL Database already exists
2. SQL Database does not exists

The server can handle both states, in case of an existing database we want the server doing some migrations for us, in the other case we want to do an initial database setup.

> **Note**
>
> Please keep in mind that ***icsw setup*** neither installs a new database nor configure your local database settings.
>
> It links your SQL database with the server environment.

icsw setup runs in an interactive mode and is responsible for a couple of basic setting:

- Creates suitable database schemata
- Creates an administrator account and an initial password for first login into the web front-end
- Creates the database config file `/etc/sysconfig/cluster/db.cf`

The script asks you for

1. Install **postgresql-server**
2. Install **python-modules-psycopg2**
3. Either run following command to migrate an former existing database:

   icsw setup --ignore-existing

   or run following command to create a completely new database:

   icsw setup

After running *icsw setup*, the script awaits some input from the admin.

```
possible choices for DB engine: psql, sqlite
DB engine (psql)                :
DB host (localhost)             :
DB user (cdbuser)               :
```

```
DB name (cdbase)                :
DB passwd (bTMZPUYMiR)          :
DB port (5432)                  :
```

Take the suggested defaults with the "ENTER" key or insert own data. By now you have to have a faultless PostgreSQL installation setup to allow the software to connect to the database.

In case something goes wrong the script displays possible steps have to be done.

Most of conflicts at this time are wrong permissions to the database or generally a wrong database installation and setup. If below steps can not solve the problem please take a look into your database manual or ask the database administrator to find out how to setup database with correct permissions.

Login into your database (commonly done with **su postgres** followed by a simple **psql**)and type following commands to create the right user and a new and empty database:

```
CREATE USER cdbuser LOGIN NOCREATEDB UNENCRYPTED PASSWORD 'my_password_123';
CREATE DATABASE cdbase OWNER cdbuser;
```

This will create a new database user with the name **cdbuser**, the desired password **my_password_123** and a new, empty database called **cdbase**

After successful creation of database user and database, we have to edit `/var/lib/pgsql/data/pg_hba.conf` (OpenSUSE) to setup correct permissions to the database. Comment out other lines so you have only the bolow three remaining.

```
local   cdbase          cdbuser                                 md5
host    cdbase          cdbuser         127.0.0.1/32            md5
host    cdbase          cdbuser         ::1/128                 md5
```

To be on the safe side it is recommended trying to login manually to the database. If you are able to connect manually likely the script is able too.

After everything goes well we get a successful connection message

```
dsn is 'dbname=cdbase user=cdbuser host=localhost password=bTMZPUYMiR
port=5432'
connection successful
```

Once connected successful to the database the script runs the migration for you. Finally you have an installation with a PostgreSQL database.

The database access data for the server is stored in `/etc/sysconfig/cluster/db.cf` created by *icsw setup*, a sample file is provided under `/etc/sysconfig/cluster/db.cf.sample`. If you want to connect via local socket leave DB_HOST empty. Either fill in the user and database information manually or run *icsw setup* for an assisted config file creation.

Every daemon and process from the server is using this file to gain access to the database. The File has to be readable for the following system entities:

- The user of the uwsgi-processes (wwwrun on SUSE systems)
- The system group idg

A typical set of rights would look like

```
-rw-r----- 1 wwwrun idg 156 May 7 2013 /etc/sysconfig/cluster/db.cf
```

Manually database backup

Although the software do periodic backups, it could be necessary to do a database backup by hand. For PostgreSQL there is a special dump command:

pg_dump -Fc -U cdbuser cdbase > DATABASE_BACKUP_NAME

This single line is enough to copy your whole database to a file.

Manually database restore

> **Important**
>
> ⚠ ATTENTION: ALL STORED DATA WILL BE LOST AFTER DROPPING YOUR DATABASE

A few more action is needed to restore database backup. First of all, before we are able to restore our database backup cleanly, we have to drop (delete) all database contents. After database contents are dropped we are able to import data into the existing and empty database.

Delete database contents:

su postgres -c "psql -c \"DROP DATABASE cdbase; \""

Restore database:

pg_restore -c -C -F c DATABASE_BACKUP_NAME | psql -U postgres

3.1.12. Portnumber for accessing the webfrontend

The Webfrontend for your server can be accessed via

`http://SERVERNAME/cluster or by http://IP_ADDRESS:80/cluster/`

The Webfrontend for your server can be accessed via

`http://SERVERNAME/cluster or by http://IP_ADDRESS:80/cluster/`

In case you run `setup_noctua.sh` script manually, the portnumber will be rewritten to **18080**. So you can access the web front-end by this url:

`http://SERVERNAME:18080/cluster or http://IP_ADDRESS:18080/cluster`

Also you can use your server localhost alias for accessing the front-end:

`http://localhost:18080/cluster or http://localhost:18080/cluster`

3.2. Installation

3.2.1. Install icsw packages

After added desired repositories for your operating system it is time to install the software packages itself and configure it. There are three main packages you have to install to get a basic server running:

1. **icsw-server**
2. **icsw-client**
3. **icsw-dependencies**

These packages contains all necessary services, binaries, libraries and dependencies for a clean and proper Installation.

If you also want to get access to the server by web GUI, you additionally need to install the **nginx-init** package and run the nginx-init http server.

For SUSE operating systems a installation command should look like following one:

zypper ref; zypper install icsw-server icsw-client icsw-dependencies

While accessing the repositories you will be prompted for valid username and password. Type in your received access data on the terminal and continue installation.

Postinstall

To guarantee the maximum possible flexibility, we decided to involve the system administrator into the installation procedure. After installation you will get a note on **stdout** how to create a new database configuration.

```
icsw setup command
```

Components of icsw-server

This is the basic package you have to install to run a server

Components of icsw-client

This is the basic package you have to install to run a client

Components of icsw-dependencies

This package provides system wide dependencies the other both can not provide.

3.3. Installation on virtual machine

Alternative to usual installation of binary packages via repositories and the operating system package manager like **zypper** , **apt-get** or **yum**, you can use a virtual machine with an ready to go installation. We distribute two popular VM image file formats running with *libvirt/qemu* and *vmware* . For information how to set up your VM environment, please take a look at the corresponding documentation of your VM vendor.

3.3.1. KVM libvirt/qemu

Following steps have to be done to run a KVM libvirt/qemu virtual machine with preinstalled **NESTOR®**;

1. Download the KVM/libvirt image and move it into the right image directory e.g. `/usr/local/share/images/`.
2. Copy an existing *.xml or create a new one
3. Edit your new *.xml file
4. Define your new virtual machine

> For virtual machine installation it could be necessary to do individual network setup (bridge) of your VM host.

Finally if your machine is setup correct, only you have to do is to start the virtual machine and have fun with monitoring.

3.3.2. VMware

You have to convert our virtual machine image for *kvm/qemu* into an suitable format for Vmware.

3.4. Upgrade software packages

From time to time, new software packages were built and can be downloaded. Especially for the master development branch there are frequent updates which can be applied to get new functions or features or simply fixing some bugs. Update period for master is about every second day.

The stable banch gets less frequent updates than the master version. Because it is the stable branch, most updates for stable affected security issues und bugfixes. Really big updates are done only if the master is stable enough for productive environment. The update period time is about 4-6 month.

The update procedure is very comfortable, it based on the system integrated package manager, for example **zypper** in OpenSUSE or **apt-get** in debian.

Comands for updating/upgrading **all** installed software by package manager are:

zypper ref; zypper dup	Refresh repositories and do whole system upgrade in OpenSUSE
apt-get update; apt-get dist-upgrade	Refresh repositories and do whole system upgrade in debian

Of course, you are also able to only update single packages, for example the package **handbook-init** . The command looks similar to the command used to update all packages:

zypper ref; zypper up handbook-init	Refresh repositories and do single package upgrade, in this case upgrade of package **handbook-init** in OpenSUSE
apt-get update; apt-get upgrade handbook-init	Refresh repositories and do single package upgrade, in this case upgrade of package **handbook-init** in debian

> ⚠It is not recommended to update only one single package except you know exactly what you are doing.

For other distributions please look into your distributors package management description.

3.5. Server control

After successful installation of **NESTOR®**; first of all you have to check if all necessary services are running. Type following command into the terminal:

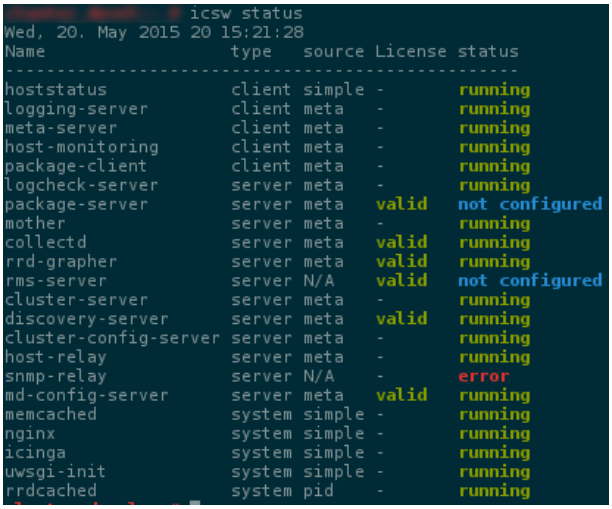
3.5.1. Show server status

icsw service status

Get more information about possible flags with: **icsw --help**

One of the most common flag is **-v**. This shows additionally the version number of each package like shown below.

Figure 3.1. icsw status command



An other common flag is **-a**. With this flag, the script shows additional information:

- Thread info
- pids
- runlevels
- Memory

Take a look into our **command reference** to learn more about *icsw* command.

3.5.2. icsw commands

The main command, which can be used to manage the cluster components, is the **icsw** command. The following table present the frequently used examples of the icsw tool:

Table 3.4. icsw - command overview

icsw-command	Functionality
`icsw service { status,start,stop,restart,debug,state }`	Show status or control icsw services

icsw-command	Functionality
`icsw state { overview,enable,disable }`	Show the state overview or enable/disable services These are the service states the which are managed by the meta-server
`icsw logwatch [ -f ] [ --system-filter ] system-name`	The logwatch command is intended to show logging messages to stdout. The **-f** option is used to append data while the file grows. The **--system-filter** option limits the logging output to a specific service. If used without any arguments it displays logging messages for all running services.
`icsw license { show_cluster_id,register_cluster,lock,unlock,show_locks }`	With the icsw license command administrators are able to lock, unlock or show locked licenses and devices from the license system. You can also show your cluster ID or register a cluster. For more information about the lock or unlock command take a look the section called “ Lock command to fall below the parameter limitation ”
`icsw setup [ service-name ]`	Create database and perform initial setup. There are many options and arguments for this command, please take a look into the  for further information.

Short overview about icsw commands

Table 3.5. icsw - service

icsw-command	Functionality
`icsw service status [ service-name ]`	Displays the status of the server and all of its services With the "service-name" it displays only status of given "service-name"
`icsw service start [ service-name ]`	Starts the service with "service-name". Without service-name option the command initiates the start of all cluster components. Warning!: if the service is disabled in the meta-server, then in some minutes the meta-server will stop the service again.
`icsw service stop [ service-name ]`	This command stops the service with the "service-name" or all services of cluster instance. ⚠Warning!: Be sure, that the service is disabled in meta-server. Otherwise the service will be start by meta-server in few minutes again.

icsw-command	Functionality
`icsw service restart [ service-name ]`	This command restarts the service with the given "service-name" or, if no service-name is used, it restarts all services of cluster instance.
`icsw service debug` { service-name }	Using the debug option is like to start a service in foreground. In contrast with services started as daemon (background), the service probably displays some stdout messages.
`icsw service state overview [ service-name ]`	Provides the state/status info about all services or specific service, if there is the "service-name"
`icsw service state enable` { service-name }	Enables the service in meta-server settings(database). That means, the meta-server is responsible for running of the service. If the service is not active, in few minutes the meta-server will start the service again.
`icsw service state disable` { service-name }	Disables the services in meta-server settings (database). Henceforward the meta-server controls, if the service is not running.

The icsw **service** command

3.6. License administration

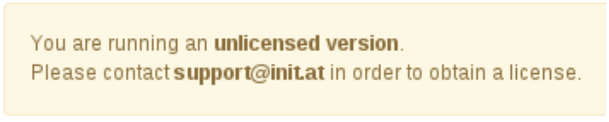
All core features, also called basis features, of the software are licensed under an open source license and for free but some enterprise features of **NESTOR®** are not. There are licenses for this enterprise features and you have to buy licenses in order to get the features working.

This section guides you through the process of getting licenses, understanding the concept of license limitation and manageing licenses.

3.6.1. Login without licenses

If you never have applied licenses to your server then you are running an unlicensed software version. To remind you of this fact you will see notification messages on some pages, for example on the login page and also on the dashboard after succesfull login.

Figure 3.2. Unlicensed notification



Also you can check your license state by navigating to **Session**  **License**.

There you will see three different drop down windows:

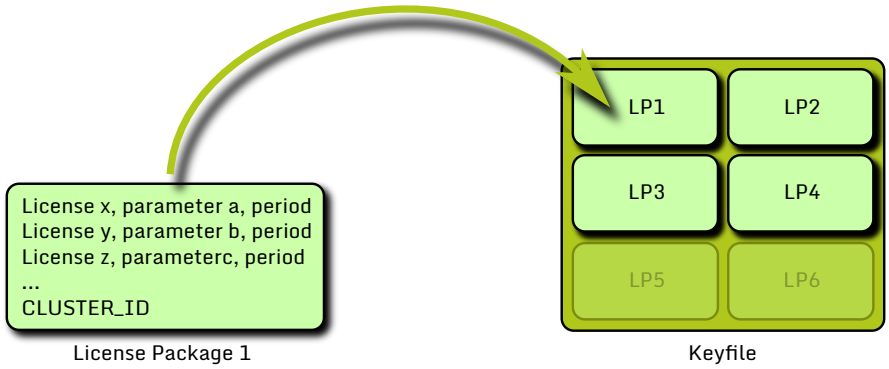
- Your licenses for this cluster
- License packages
- Upload license file

Figure 3.3. License overview

3.6.2. License package structure

Shipped Licenses are keyfiles containing information about licensed features, license period and license parameter. Each license is associated to one specific Cluster ID. A keyfile can contain one or more license packages assigned with one or more cluster IDs.

Figure 3.4. License package



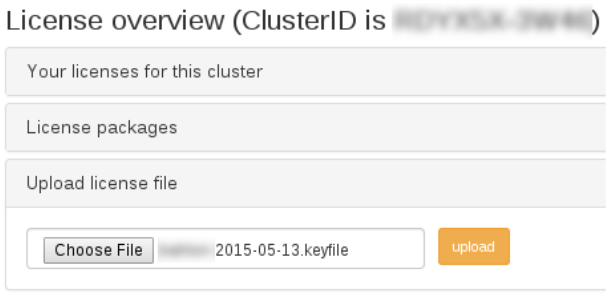
The license package dropdown window shows you an content overview of uploaded keyfiles.

Keyfiles containing more than one single license package will be displayed in separate tabs inside of the **License package** dropdown window.

3.6.3. Upload license keyfile

You can't expand any dropdown menu by left clicking on the arrow beside the dropdown except the **Upload license file** one. Use the **Choose File** button to select your valid license keyfile. After selecting your keyfile its name will be displayed on the right side of the button:

Figure 3.5. License overview



Push the **upload** button to integrate your valid license keyfile into the server and activate aquired enterprise features.

3.6.4. Show license status

After uploading your valid license keyfile to the server, your license overview imediately will be updated and shows your purchased licenses. Generally, if you want to see your license overview navigate to **Session** **License** to display the license status.

Figure 3.6. icsw status command

License overview (ClusterID is [redacted])

Your licenses for this cluster

License	Description	Parameters	Status		Type	License package
MD Config Server	Monitoring Daemon Configuration Writer	**Device**: 100 (used: 0) **Service**: 1000 (used: 0)	Valid	until 2015-06-21	Test	Test package NOCTUA 21.05.2015
Key Performance Indicators	Calculate key figures to measure the performance of your cluster					
Graphing	Comprehensive graphical evaluation using RRDs	**Device**: 100 (used: 3)	Valid	until 2015-06-21	Test	Test package NOCTUA 21.05.2015
Discovery Server	Automatical configuration using SNMP	**Device**: 100 (used: 1)	Valid	until 2015-06-21	Test	Test package NOCTUA 21.05.2015
Monitoring dashboard	Central monitoring unit consisting of livestatus, geolocation and maplocation	**Device**: 100 (used: 9) **Service**: 1000 (used: 81)	Valid	until 2015-06-21	Test	Test package NOCTUA 21.05.2015
Distributed Monitoring	Distribute monitoring load to multiple workers					
Database snapshot	Keep track of the configuration changes	**Device**: 100 (used: 13)	Valid	until 2015-06-21	Test	Test package NOCTUA 21.05.2015
Notifications	Status notifications via mail and text messages	**Device**: 100 (used: 0) **Service**: 1000 (used: 1)	Valid	until 2015-06-21	Test	Test package NOCTUA 21.05.2015
Network Weathermap	Overview of relevant network utilization data					
Rootkit Hunter	Security scan for installations					
Reporting	Generate graphs to view the state of your cluster					
License Optimization Management	Interactive graphic license utilization evaluation					
Virtual Desktop	Manage virtual desktop sessions on your cluster					
HPC Workbench	Convenient interface for job management					
Package Install	Configure repositories and install packages on your nodes					
Resource Management System	Overview over your job system					
Netboot	Manage the boot process of your nodes					

License License name

Description Description of the specific license

Parameter Parameter value is the limitation of licenses in context of

- Device
- Service
- User
- External license

Used licenses and amount will be displayed as info window

Figure 3.7. icsw license service used

Status **Valid in future**, license will be valid from point of time in future

Valid, license is active and valid until displayed date

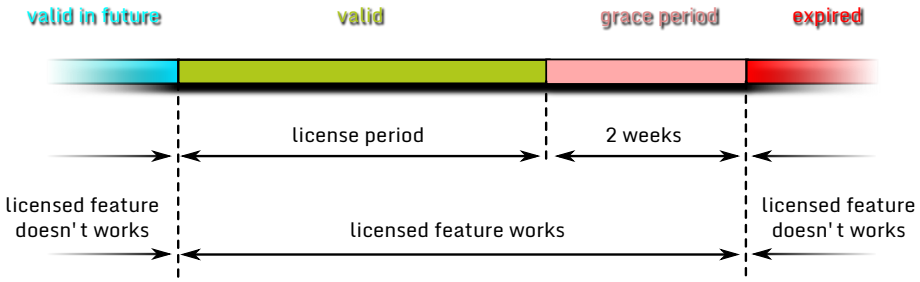
Grace, license is in grace time. It is still active until the grace timeperiod of **2 weeks** is over.

	Expired, license is out of grace time period.
License package	Name of used license package

3.6.5. License time periods

There are four different periods or states of licenses. Dependent of the period or the state a licensed feature is working or not.

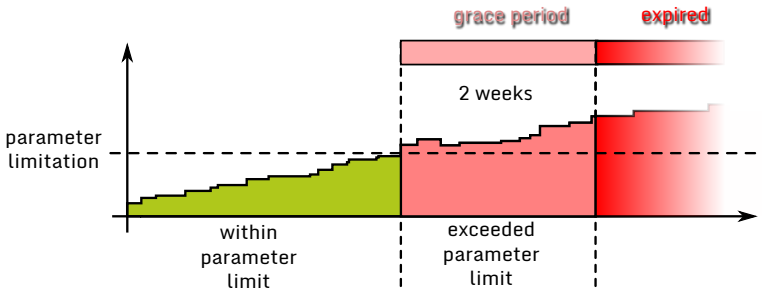
Figure 3.8. License states



3.6.6. License parameter limitations

The second factor which decides if a license is valid or not is the parameter limitation. Dependent of purchased parameter amount and of used parameter the license can be valid, in grace time or expired.

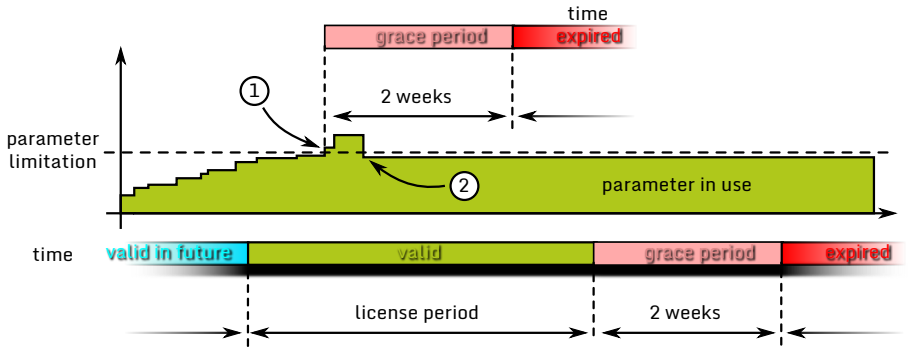
Figure 3.9. License states



3.6.7. License violation

Licenses can be violated by exceeding the license time period or by exceeding one of the license parameters limitation. In case of exceeding the license parameter limitation also a grace time period starts which is totaly independent of the license time period grace time.

Figure 3.10. License states



Lock command to fall below the parameter limitation

For a small violation of parameter limitation there is a **lock** command to get back within parameter limit.

In Figure 3.10, “ License states ” marked with (1) you can see a license violation by exceeding the parameter limitation. Because of a violated license the grace time period starts immediatelly. In your license overview you will notice a warning message for this violated license.

Now, you have two options.

1. Purchase an extended license for that feature to increase the parameter limitation

2. Lock the license parameter to get back below parameter limitation

There is a special command for the second option:

`icsw license lock` { -d *`DEVICE`* } { -l *`LICENSE`* }

After locking licenses [illustrated on (2) of Figure 3.10, “ License states ”], the **used** license amount falls below the parameter limit and is valid again.

If you want to know if any licenses are locked, use following command:

`icsw license show_locks`
It displays all locked licenses and corresponding devices.

Chapter 4. Core concepts

4.1. Services and components

This sections explains the core concepts behind **NESTOR®**. It gives a top level overview of it's components and capabilities.

Following some separate components are listed and to which part of the software it belongs to.

4.1.1. We do not need to reinvent the wheel

As relative complex software **NESTOR®** use some well known frameworks and technology we absolutely have to mention.

Used Frameworks

django	Web application framework written in python.
angularjs	Opensource framework developed by google™.
bootstrap	CSS Framework developed by twitter™

Used software solutions

icinga	Industy standard software for monitoring devices.
rrd-tool	RRDtool is the OpenSource industry standard, high performance data logging and graphing system for time series data. RRDtool can be easily integrated in shell scripts, perl, python, ruby, lua or tcl applications.
nginx	Powerfull small and fast webserver.
SGE	Son of grid engine.

4.1.2. Common components

Parts which both software packages needs to works properly.

Components

meta-server	This daemon is responsible for restarting services that might have crashed or were otherwise killed. This functionality should be taken over by systemd. `/var/lib/meta_server` contains the relevant information about which services should be running.
logging-server	Creates the structure needed for receiving logs via rsyslog or syslog-ng.

4.1.3. Monitoring components

Monitoring in context of hard- and software will be practiced to get information about specified systems.

Components

md-config-server	Responsible for configuration of icinga. Interacts with database.

cluster-server	Responsible for writing config files and general coordination of the cluster. Listens on port **TCP/8004** . cluster-server.py is a general purpose server that handles various tasks like writing /etc/hosts, generating a valid DHCP configuration, configuration of the BIND nameserver, feeding LDAP and / or YP servers, ...
collclient.py	Client part of the host-monitoring collserver.py.
ccollclientzmq	Frontend program to talk to collrelay.

4.1.4. HPC components

NESTOR® consist of many different parts and services. Each of these services perform a specific set of tasks in the cluster. Most of these services are network enabled and listen on a specific port for commands. The following list tries to give an overview about the most important parts.

The general idea of **NESTOR®** is simple. Create an image, a kernel and a set of configuration files to be used by your nodes and distribute them to the nodes. The distribution is done via *PXE*. **NESTOR®** enables you to describe the node specific configurations in Python.

Components

cluster-config-server	Generates the files for the Clusternodes (based on the config stored in the Database) to make the nodes distinguishable.
mother	Creates the tftpboot /ethernet structure, monitors the installation progress of the nodes. Listens on ports **TCP/8000 TCP/8001** . Mother provides access to *IPMI* as well.
package-server	Provides repositories available for installation by the package-client. Listens on port **TCP/8007**.
package-client	Install required software by using the locally available package management commands. **zypper**, **yum** or **apt-get**.
hoststatus	A small program written in C that transmits node status messages to the cluster-server. The hoststatus is written in C to be easily includable in the initial ramdisk.It listens on port TCP/2002. hoststatus is in the package child
logcheck-server	Do log rotation and deletes logs older than a specified time range.
rms-server.py	Provides integration of **NESTOR®** with *SGE*. The commands **sns** and **sjs** rely on it.
discovery-server	Daemon for automatic configuration of devices.

4.1.5. Clusterdatabase

The database where all the configuration data of **NESTOR®** installation is stored is generally referred as the "clusterdatabase".

Throughout this document we might refer to:

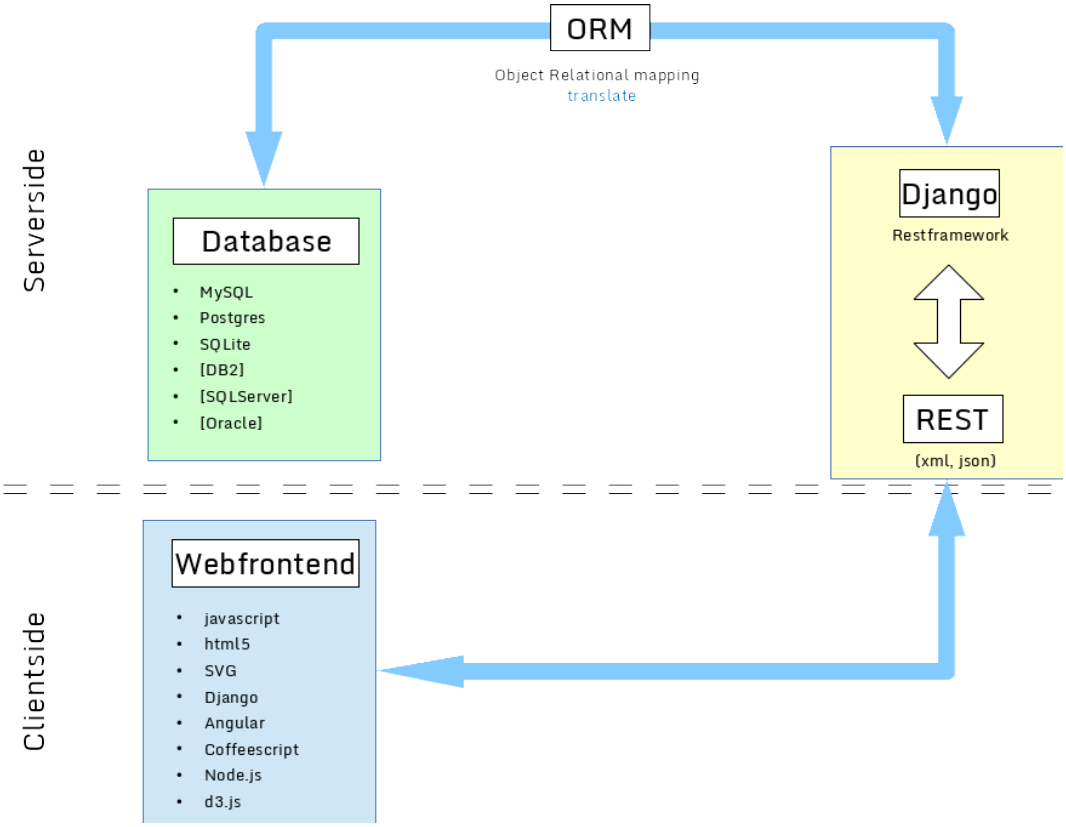
Server side scripts . Are generally services or scripts that are running on the clusterserver. Most of them need database connectivity to function properly.

Node side scripts . On the other hand, node side scripts are daemons or scripts that are generally run on a node of the cluster. Node side scripts don't require access to the cluster Database.

4.1.6. Dataflow between webfrontend and database

The graphic shows just the part between database and webfrontend, only a small but essential part of monitoring.

Figure 4.1. Webfrontend and database dataflow



Dataflow with help of django and django REST framework.

4.2. Run important services

Because **NESTOR®** consists of many different parts working together, it is not obligatory to run every service at once. Services like **package-install** or **discovery-server** are not essential to operate monitoring or cluster management.

For that reason, default installation of **NESTOR®** is rudimentary. Special services and functions **are not** activated by default. Activation of certain services force the user to push some buttons or move some lever.

Two spots where you can activate services are:

- cluster server information
- Device Config

Chapter 5. Webfrontend

5.1. First connection

Most configuration in **NESTOR®** administrators have to do, is accessible over a standard html compatible browser like Mozilla Firefox™ or Google Chrome™. Once **NESTOR®** is installed and all required services are running, all you have to do is to connect to the server via browser.

Type in

```
http://SERVER-IP-ADDRESS:80/cluster/
```

or

```
http://SERVERNAME/cluster/
```

in your browser addressbar to connect to the server. If you connect the first time to the server you will be redirected to the account info page.

Important

⚠You really have to change your password now. If you don't change it, **NESTOR®** takes his own during installation procedure generated password you never seen before and next time you can not log in.

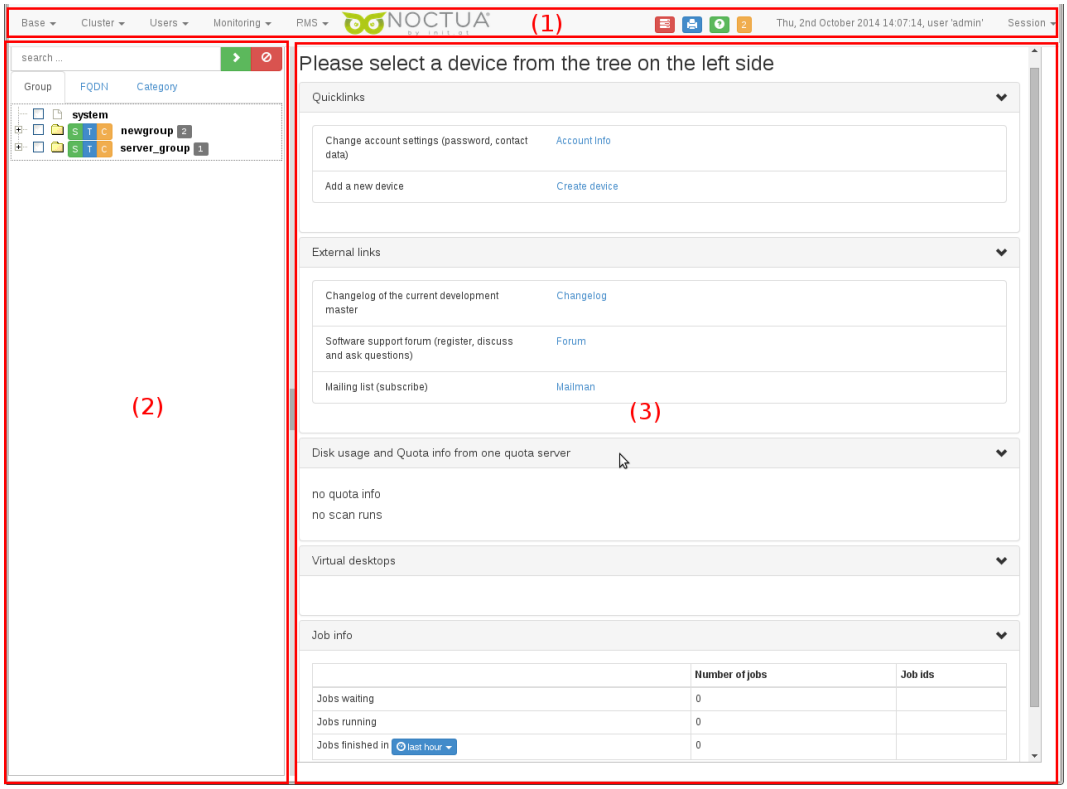
If you run the **setup_noctua.sh** script manually, it generates you a new admin password. In this case you must look for the password in your shell output.

5.2. Areas

NESTOR® webfrontend offers you a very clear view. There are three areas you will work with:

- Menu area (1)
- Sidebar device tree (2)
- Main area (3)

Figure 5.1. Three areas



Areas you'll see after login

5.2.1. Menu area (1)

In the menu area you'll find submenus, buttons, date, time and user section.

Submenus

1. Base
2. Users
3. Monitoring
4. Session

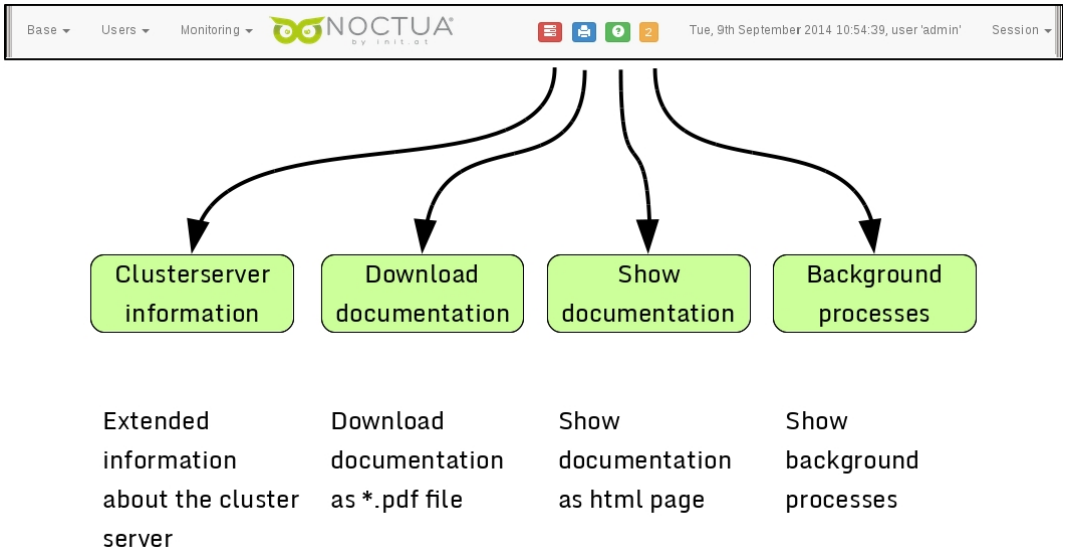
NESTOR® offers some additional menus:

1. RMS - Resource management System
2. Cluster

Buttons

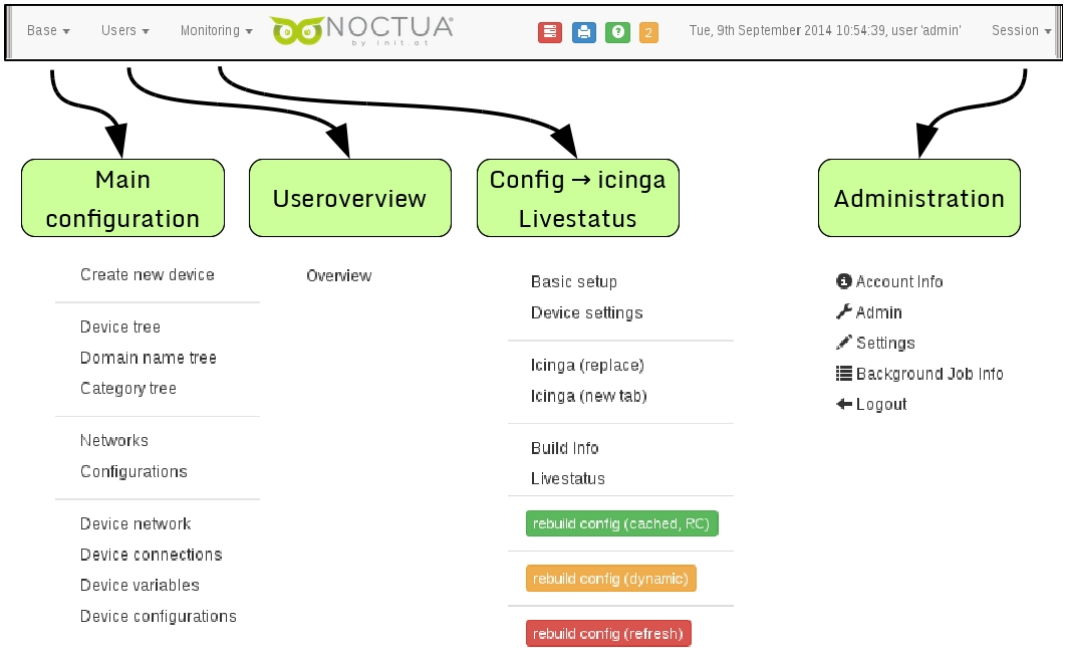
1. cluster server information
2. show cluster handbook as pdf
3. show index
4. number of background jobs

Figure 5.2. Menu buttons



Buttons and submenus for **NESTOR®**

Figure 5.3. Menus



Menus for **NESTOR®**

5.2.2. Sidebar device tree (2)

In the tree area you can find your device group tree and associated devices. Located on top, there is a searchfield and 2 buttons.

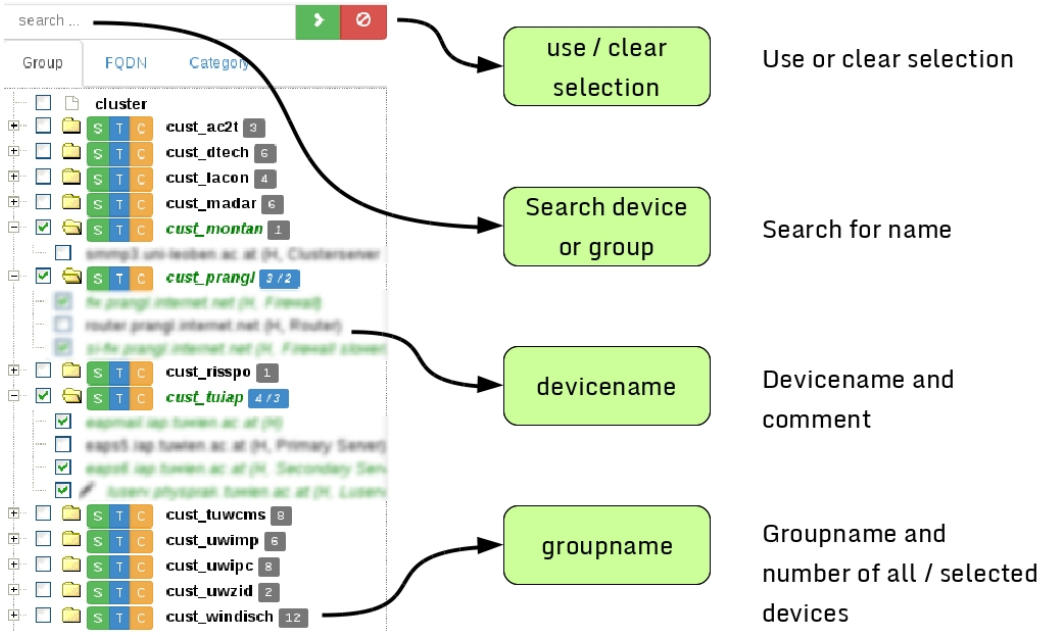
1. Searchfield
2. use selection Button (green with arrow)
3. clear selection Button (red with circle)

4. Group

5. FQDN (Full Qualified Domain Name)

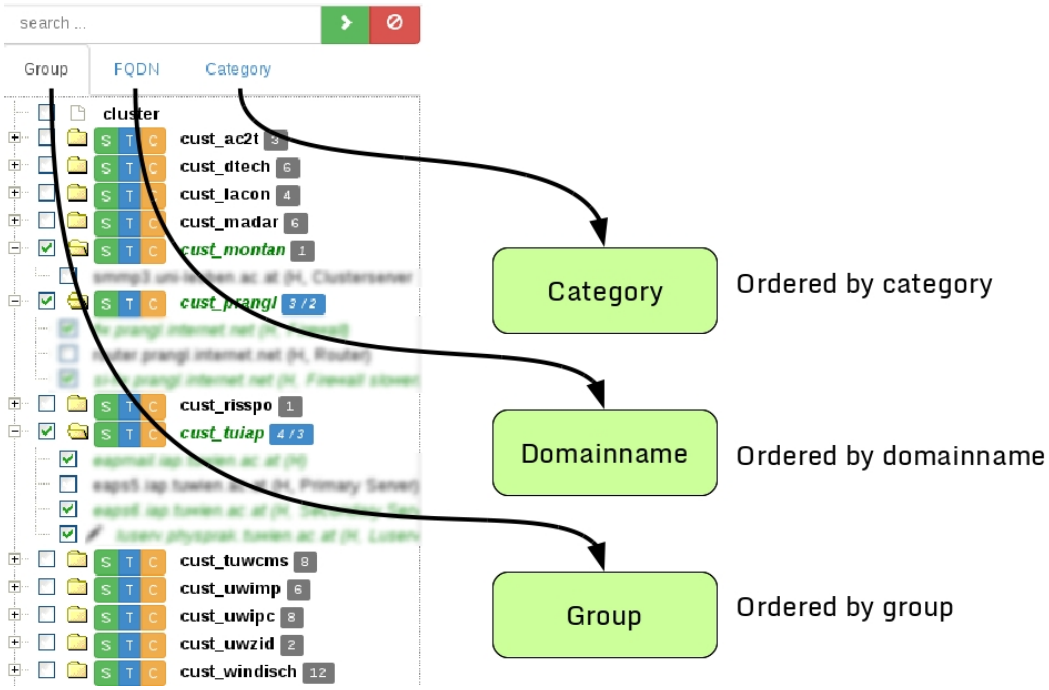
6. Category

Figure 5.4. Devicetree



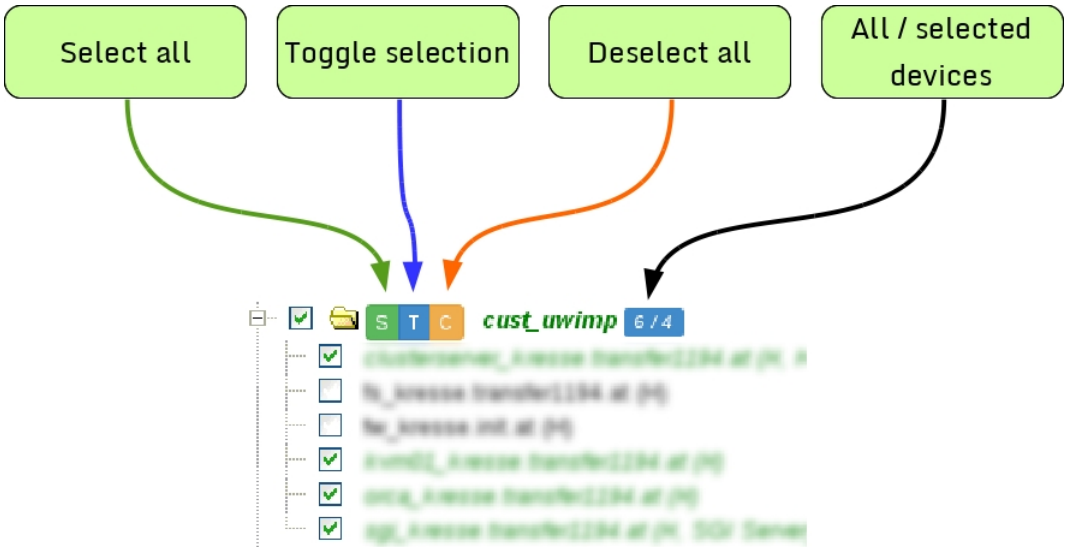
NESTOR®

Figure 5.5. Devicetree



NESTOR®

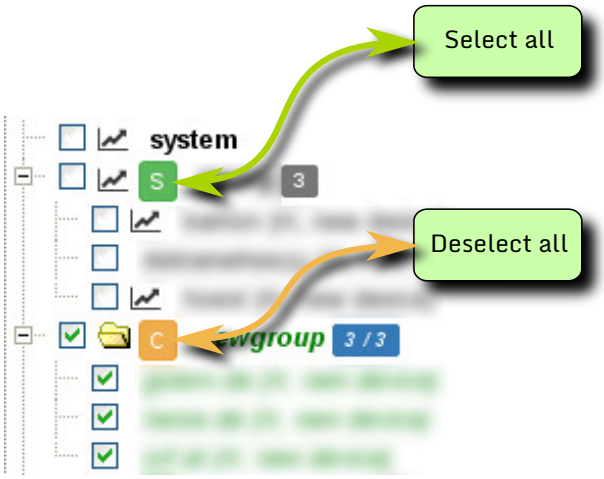
Figure 5.6. STC Selection buttons



Buttons to select, deselect or toggle selection

Alternativ there is an one button selection method.

Figure 5.7. S/C Selection button

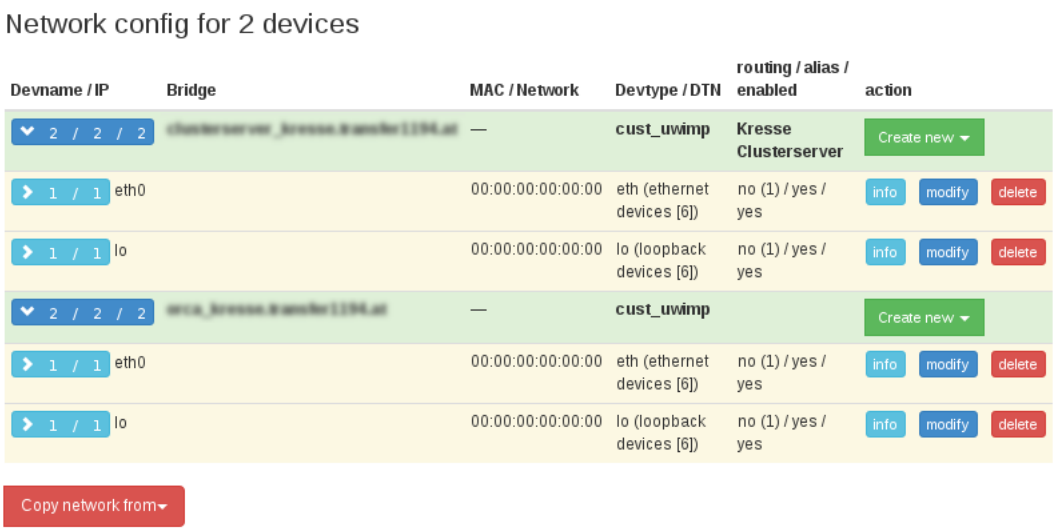


One button selection, either select all or deselect all

5.2.3. Main area (3)

All the configurations and input takes place in the main area. According to the selected or preselected devices and settings, corresponding page appears.

Figure 5.8. Possible main area



One possible view of main area after select some devices in "device network"

5.2.4. Cluster server information

The cluster server information button shows three overview tabs, one tab with information about **definied cluster roles**, one tab with information about server results and one with information about the **server** itself.

Once called, **server information** restarts the **memcached** daemon.

Cluster roles definied

Inside this upper tab, there is a table showing the **Name**, **reachableIP** and the defined **cost** of each of them. This tab is only for displaying information.

Each of the defined roles provides special functionality to the server.

One server result

Also a tab only for displaying general information.

- valid
- name
- Result
- max Memory
- total Memory

One Server checked

This is the only one tab inside of server information, which allows you to control something. You are able to control services as you do by command line.

Following information will be displayed:

Server information

Instance	Name of service
Type	Type of service {node, server,system}
Check	Kind of Check
Installstatus	Status if service is installed or not
Version number	Versionnumber of installed service
Processnumber	Number of processes started
Memory usage	Displays memory usage as number and as statusbar
Action Buttons	Button to apply action to the services

Figure 5.9. Cluster server information

Instance	Type	Check	icinga.init.at, 939.343 MB max / 3.389 GB total Memory				
hoststatus	client	simple	3.0–391	1	2 MB		disable
logging-server	client	meta	3.0–391	3	18 MB		
meta-server	client	meta	3.0–391	5	66 MB		
host-monitoring	client	meta	3.0–391	17	70 MB		disable
package-client	client	meta	3.0–391	6	16 MB		disable
logcheck-server	server	meta	3.0–391	not running			enable
package-server	server	meta	3.0–391	11	153 MB		disable
mother	server	meta	3.0–391	not running			enable
collectd	server	meta	3.0–391	28	499 MB		disable
memcached	system	simple		1	25 MB		
nginx	system	simple		9	18 MB		
icinga	system	simple		2	57 MB		disable
uwsgi-init	system	simple		16	939 MB		
rrdcached	system	pid_file		1	6 MB		
rrd-grapher	server	meta	3.0–391	11	249 MB		disable
rms-server	server	meta	3.0–391	not running			enable
cluster-server	server	meta	3.0–391	11	214 MB		disable
discovery-server	server	meta	3.0–391	23	169 MB		disable
cluster-config-server	server	meta	3.0–391	not running			enable
host-relay	server	meta	3.0–391	9	94 MB		disable
snmp-relay	server	meta	3.0–391	18	215 MB		disable
md-config-server	server	meta	3.0–391	27	653 MB		disable

Backgroundinformation about running or stopped services

Chapter 6. Concept of operation

6.1. Working with the web front-end

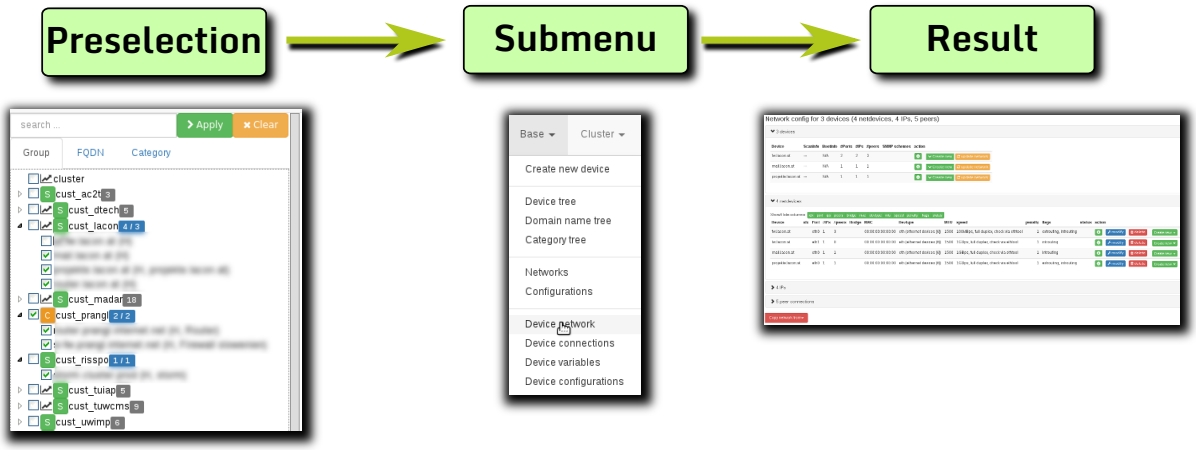
The workflow inside of the web front-end follows a special pattern. This workflow repeats for specific actions and therefore it is worth to mention and learn it. We divide this section into four subsections listed below to show the difference of each one. Of course it is possible to get similar results by different ways but each way have advantages and disadvantages or is more or less efficient.

There are also software regions like **Nodeboot** which are only accessible by one single way.

6.1.1. Preselection and submenu

All preselections are done in the sidebar device tree Section 5.2.2, “ Sidebar device tree (2) ”. Select groups or devices and click in top menu on desired submenu to access .

Figure 6.1. Submenu method



Device network for preselected devices as result

Submenu method is recommended for working with multiple devices.

Following areas can be accessed by the submenu method:

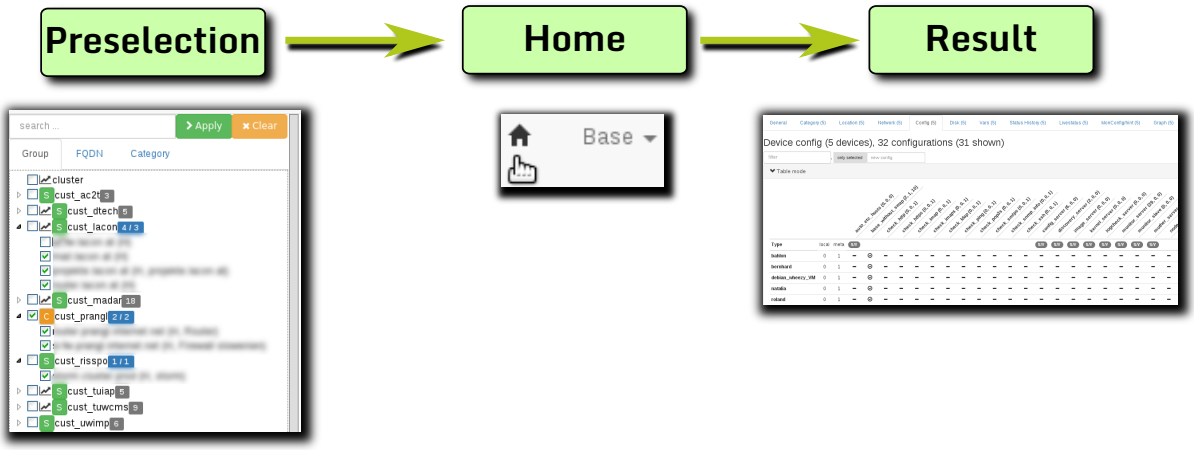
- Device Tree
- Device Variable
- Device Network
- Device Configuration
- Nodeboot
- Package Install
- Device Settings
- Monitoring Overview
- Livestatus

6.1.2. Preselection and homebutton

The method by home button is more general. Again preselection of devices or groups takes place in sidebar device tree but afterwards instead of choosing a submenu, this time we click on the **home** button.

As result we get some useful tabs for each device.

Figure 6.2. Home button method



Device configurations for preselected devices as result

Home button method is recommended for working with multiple devices.

Following areas can be accessed by the home button method:

- General
- Category
- Location
- Network
- Config
- Disk
- Vars
- Status History (Detailed information about this tab, see in the chapter **Status History** at [http://www.initat.org/hpc_monitoring/wp-content/uploads/2015/03/corvus_handbook.pdf])
- Livestatus
- MonConfig/hint
- Graph

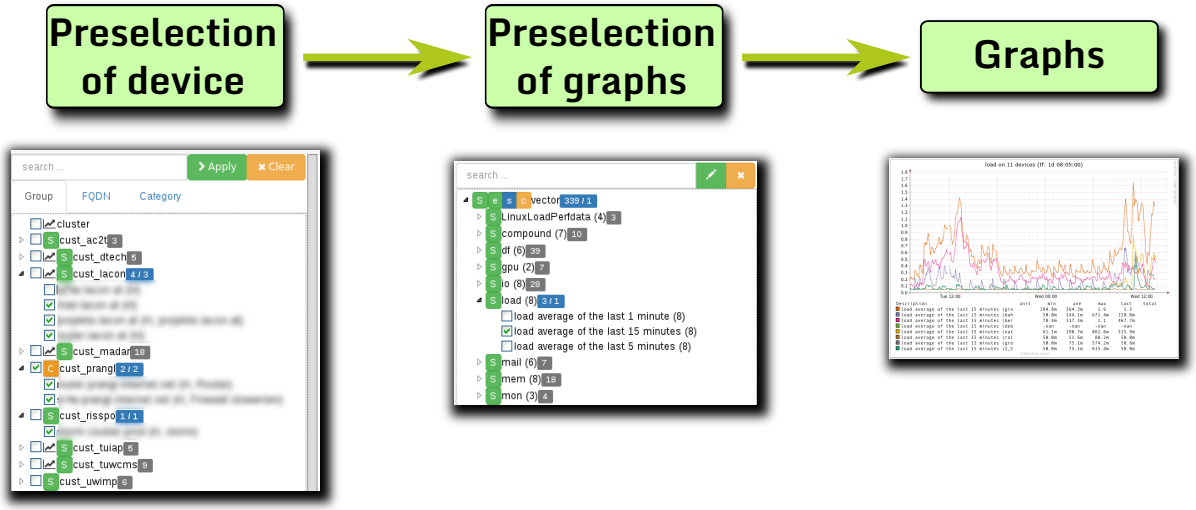
By direct click on the device name in **sidebar device tree** we get similar overview tabs like by home button method but only for one single device.

Direct method is recommended for working with single devices.

6.1.3. Two-stage selection

For graphing there is the requirement first to select devices, then select the wanted graphs and finally draw this graphs for selected devices. The graph preselection will remain unaffected if you change the device preselection. After changing device preselection you must push the **Apply** button. Same is true for drawing graphs.

Figure 6.3. Two-stage selection



Preselected devices and preselected graphs results in graphs

6.2. Input field filter/regex, hide/show column, pagination and sorting

We implemented three little helping functions to ease the handling with large and complex tables inside of the web front-end. This helping functions are located on several places/pages e.g. like **sidebar device tree**, **device tree** or **device configurations** mostly on top of the specific place/page.

6.2.1. Input field filter/regex

Most common auxiliary function is the filter input field on top of device trees or configuration tables. Most simple usage of this filter field is to insert some text string or number to filter for. If you do so for example in **sidebar device tree** then only matching devices will be selected.

Table 6.1. Examples for filter function in sibebar device tree

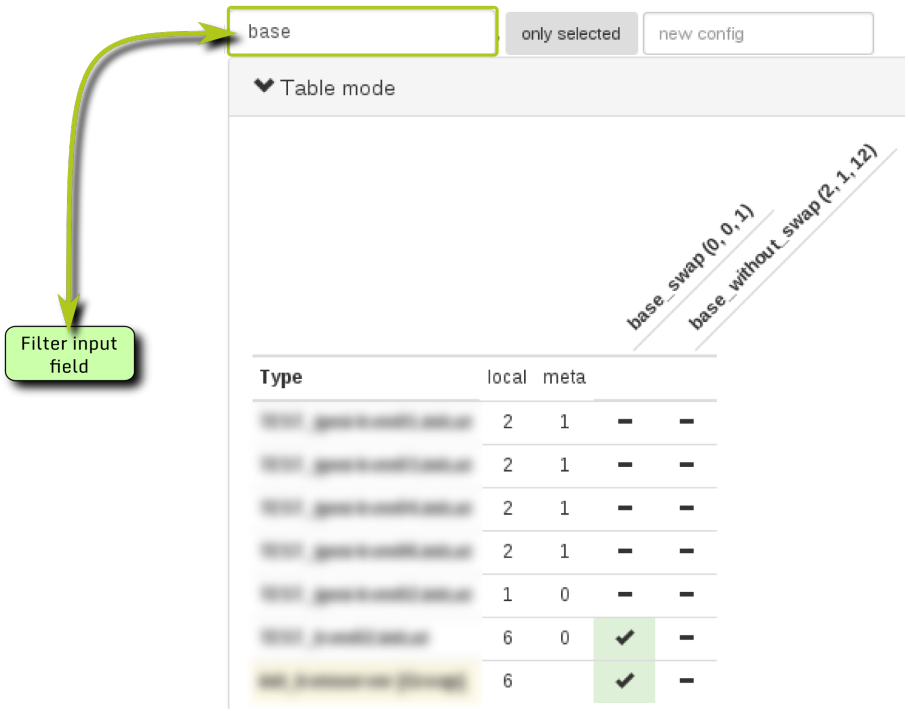
Regular expression match character	**Matching description**	**Example**	**Result**
^	Starting possition of line	`^node`	Select all devices whose device name **begin** with the string **node**.
[0-9]	Range of numbers	`node[0-5]`	Select all devices which called node immediatelly followed by a **number between 0 and 5**

Regular expression match character	Matching description	Example	Result
$	End of line	`[0-9]$`	Select all devices whose name **end** with a **numeric between 0 and 9**.
\d	Digit	`\d$`	Select all devices whose name **end** with a **digit**.

Regular expressions for input fields.

Further information on regular expressions filter can be found in the world wide web by looking for **javascript regex**.

Figure 6.4. Input field filter for device configurations

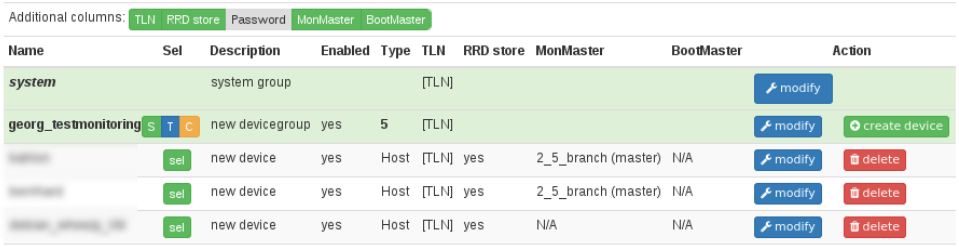


Displays all device configuration entries beginning with **base**, even they are not selected.

6.2.2. Hide/show columns

An other auxiliary function in handling with tables are the **show** and **hide** buttons on top of tables. With this buttons you can easily show or hide specific table columns.

Figure 6.5. Show and hide buttons for table columns



Device tree with hidden **Password** column.

6.2.3. pagination

To ease displaying of longer lists and to avoid to much page scrolling there is also a simple pagination function built into the software. With pagination we are able to limit entry output on pages to a specific number. Only the choosen numer of etries will be displayed, the other entries, if there are some, will be divided on separate pages which can be accessed via the page button.

Figure 6.6. Pages and entries per page

Shows 20 entries per page, divided into three pages

6.2.4. Sort column

Last but not least we'd like to mention the column sort function. It also could be very useful to display only desired data.

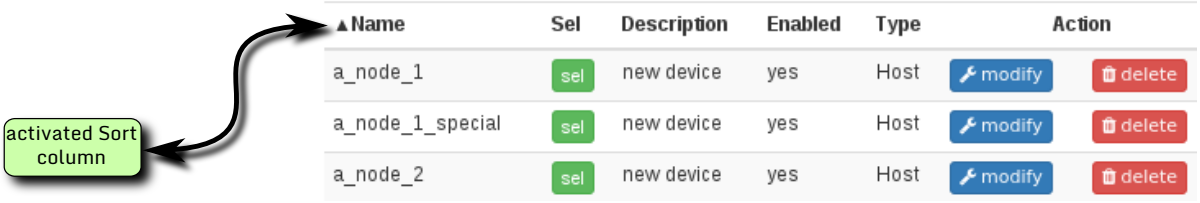
Not all columns provide this sort function, but most of them do. The function will be toggled by clicking onto the column name. If the function is activated there is a small triangle left beside the column name pointing with its tip either in top direction for **ascending** or in bottom direction for **descending** sorting. If no triangle is visible sorting function is deactivated.

Sorting method is:

1. First numerical

2. Second alphabetical

Figure 6.7. Sort column

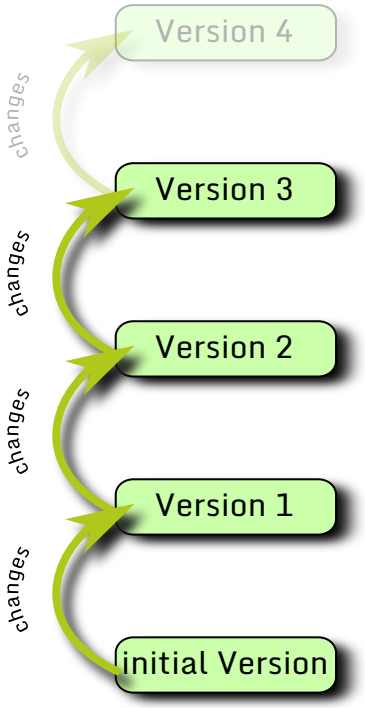


Activated ascending sorting marked with a small black triangle pointing in top direction.

6.3. Reversion

Sometimes it could be very neccessary to undo former applied changes, for example if you have a typo in an script, variable or whereever or if someone else has applied some changes and you want to see the state before and after this changes.

Figure 6.8. Reversion principle



Reversion chain

Newest data and changes will be attached on top.

6.3.1. History overview

Our developer created a reversion function not only to display what changes were done but also to go back in change history to desired state and drop changes which were done afterwards. History reversion can be found in top menu at **Cluster** **History**.

The reversion function is based upon the central database (default is PostgreSQL) so in principle every change written into the database can be reversed. Normally there are lot of different data stored in the database to ensure every component works fine but it makes no sense to provide the reversion feature to all of this collected data. For normal user and administrators it is completely enough to revert changes which were done via the web front-end.

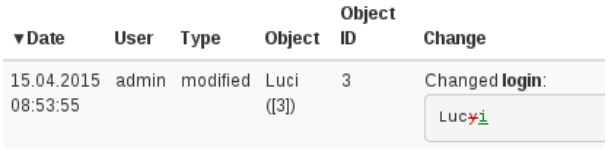
For example three new user were added in **User Management**. Like shown below, system history lists all relevant database entries for each of them.

Figure 6.9. System history

Database excerpt for **User** data

Now lets suppose there was a typo in one of these names e.g **Lucy** was changed to **Luci**. If we take a look into the system history under **User** we get exactly this change displayed in an diff-like style.

Figure 6.10. System history



Displayed diff in system history after changed user name

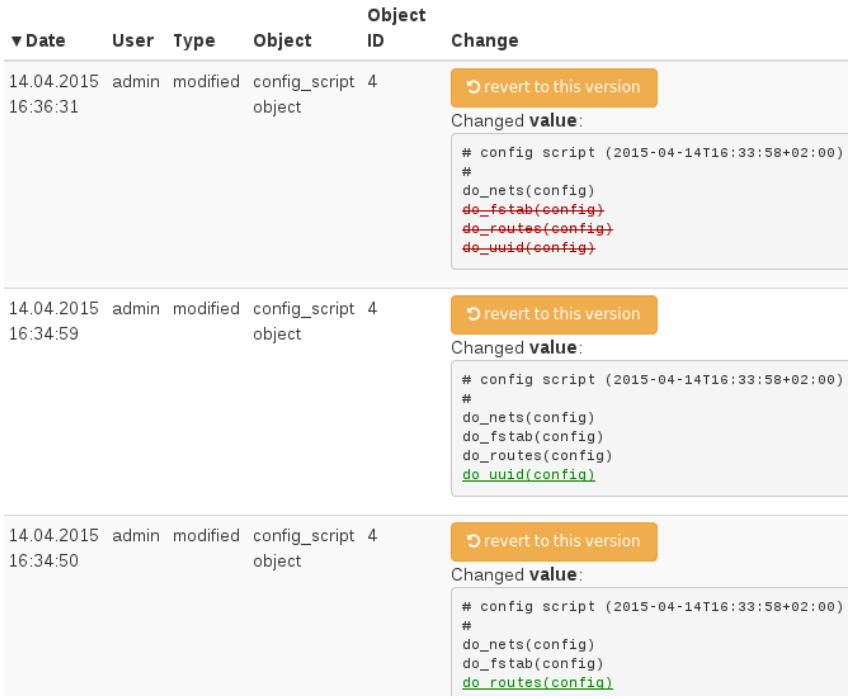
6.3.2. Revert to former version

If you are not only satisfied to display changes but also really want to go back to an earlier version there is the **revert to this version** button.

For example, someone changed directory paths in an script located at **Base** **Configurations** and you would like to display that changes, simply navigate to the **History** tab of **Modification** window to get a list containing all applied changes up to now.

Next step to do is to mark your desired version out of the list. Now, you can either apply reversion by clicking on the **Modify** button or just switch to the script editor to check how the script looks like after reverting.

Figure 6.11. Reversion overview



Typical colored diff output

Table 6.2. Colorcode for reversion

green	inserted character
red	deleted character
black	unchanged character

We can see all changes at a glance in above shown figure.

Chapter 7. User and group management

7.1. Create user or group

After installation of **NESTOR®** the user **admin** and the group **admingrp** already exists. This is the user you have to change password for after first login into your fresh installed system.

User admin has all possible rights and permissions to add, to modify and to delete devices/groups etc. User admin is also able to do reconfiguration of database and of course able to add or delete new user.

If you want to set restrictions for some user or groups, for example for external staff, you have to create this new restricted user/group with following buttons:

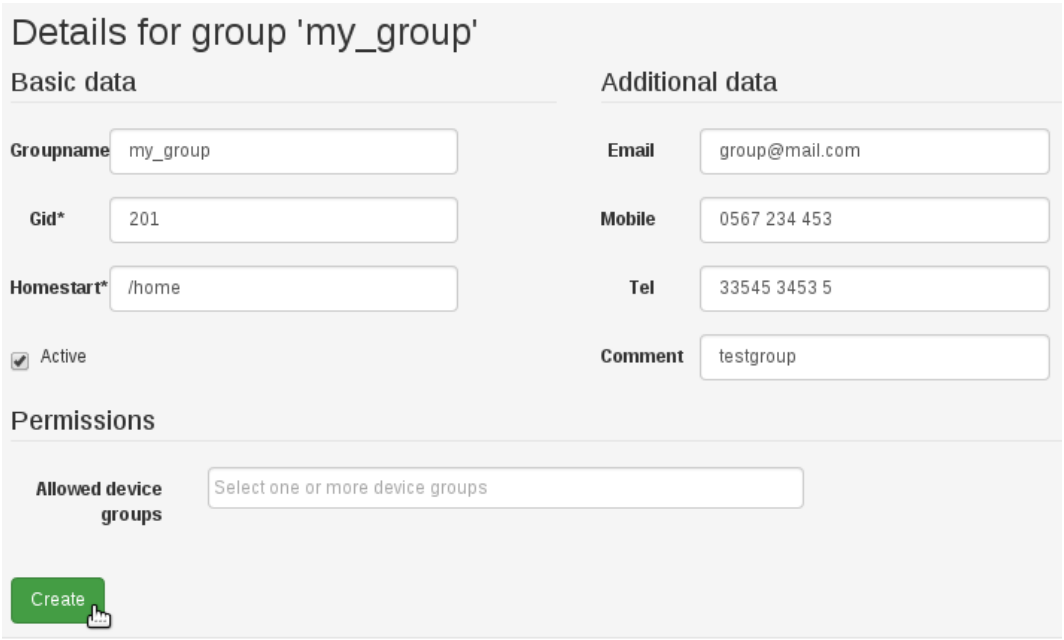
Figure 7.1. Userbuttons

Create user, group or sync users Button

7.2. Create group form

To add a new group in user management, klick the *"create group"* button, fill out the form and confirm your input by klicking the *"Create"* button.

Figure 7.2. Group create form



Basic settings form to create group

The form is self-explanatory, but some input should be mentioned anyway:

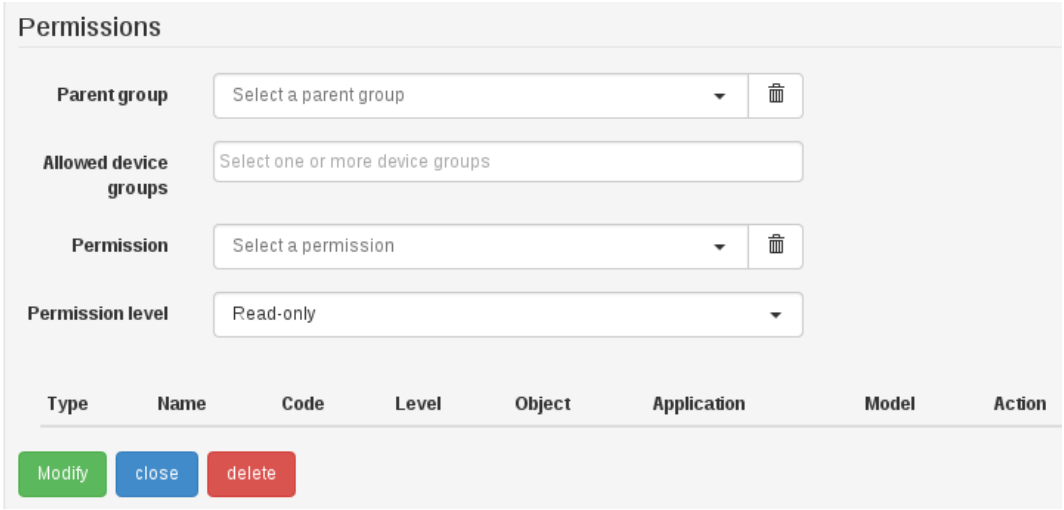
Gid* Internal group ID

Device group permissions Set basic permissions to get access to selected devicegroup

Another extended form can be shown by clicking the new created group in the user/group tree:

A more complex permission system appears.

Figure 7.3. Extended permissions



More complex permission settings

7.3. Create user form

Similar structure and procedure is true for creating new user.

Also here we must mention some contents:

Uid*	Internal user ID
Parent group	Is the superior group
Secondary group	Operating system group
Is superuser	Owns all rights and permissions like the admin own

7.4. Permission System

7.4.1. Permission

The permission system is divided into several parts which covers certain functions. Some permissions depend on other permissions, or in other words, chainpermissions. The more permissions user get the more powerfull they can act. The user "admin" or "superuser" is the most powerfull user. Admin have all possible rights and permissions.

Below is a list with permissions and what their functions are.

background_job

Show background jobs (G)	Shows additional menu button: **Session**  **Background Job Info**

config

modify global configurations (G)	Shows additional menu button: **Base**  **Configurations**

device

Access to device graphs (G/O)	Shows **graphs** tab for selected devices. Depends on possibility to choose devices (**acess all devices**)
change disk setup (G/O)	Shows disk tab for selected devices. Depends on possibility to choose devices (**acess all devices**)
Change basic settings (G/O)	Change basic settings (General) for selected devices. Depends on possibility to choose devices (**acess all devices**)
Change boot settings (G/O)	Shows new top-menu named **Cluster**
Change configuration (G/O)	Show **Config** tab for selected devices. Depends on possibility to choose devices (**acess all devices**)
Change device category (G/O)	Show **Category** tab for selected devices. Depends on possibility to choose devices (**acess all devices**)
Change device connection (G/O)	Shows new top-menu: **Base**  **Device connections**
Change device location (G/O)	Show **Location** tab for selected devices. Depends on possibility to choose devices (**acess all devices**)

Change device monitoring config (G/O)	Shows 3 new tabs for selected devices: • Livestatus • Monconfig • MonHint
Change network (G/O)	Shows new top menu content: **Base**  **device network** Depends on possibility to choose devices (**acess all devices**)
Change variables (G/O)	Show **vars** tab for selected devices and new top menu: **Base**  **Device variables**. Depends on possibility to choose devices (**acess all devices**)
access all devices (G)	The main permission to show devices. Most of above permissions depends on it. Shows existing devices in device tree on the left.

group

Group administrator (G/O)	...

image

Modify images (G)	...

kernel

Modify kernels (G)	...

mon_check_command

Change monitor settings (G)	Shows new top menu content under: **Monitoring**  **Basic Setup / Build Info**

network

modify global network settings (G)	...
show network clustering (G)	...

package

access package install site (G)	Shows new top menu under: **Cluster**  **Package install**. Additional software packages can be choosen and installed by this menu button.

partition_fs

modify partitions (G)	...

user

Administrator (G/O)	Shows new top menu content unter: **Session**  **Admin**
Change RMS settings (G/O)	...
Modify category tree (G)	Shows new top menu content unter: **Base**  **Category tree**
modify device tree (G)	Shows 2 new top menu content unter: **Base**  **Crerate new device / Device tree /**
modify domain name tree (G)	Shows new top menu content unter **Base**  **Domain name tree**
start and stop server processes (G/O)	...

7.4.2. Permission level

The permission level defines what can be done by users. In combination with the permission itself, administrators are more flexible in assigning rights and permissions to user or to groups.

Below are 4 main permission levels which can be assigned.

Read-only	Permits the user to read data. User can't change, create or delete data.
Modify	Permits the user to change existing data. Includes read-only level.
Modify, Create	Permits user to change and create new data. Deletion is not possible.
Modify, Create, Delete	All Permissions are granted.

Chapter 8. Package installation

Installation of packages via the webfrontend is another helpful feature provided by **NESTOR®**. It offers you to install software packages on one or many systems over the webfrontend, without the need to login on each local machine and install packages manually by terminal command.

Your **NESTOR®** operates as central package installation entity, stores its repositories in the database and can also distribute its repositories to connected nodes.

It's a huge ease for user with less experience to do software installation with a few clicks instead of typing long and cryptic terminal commands.

In this section you can learn how to setup this feature, how to configure and how to use it.

8.1. Preparing installation for package install

Two important services for this function are: **package-client, package-server**

First of all you have to install both, **package-client** on the client machine(s) and **package-server** on your server machine

Before you are able to install packages by the webfrontend, you have to configure your machines appropriate. Not only the server-side configuration but also the client-side configuration is essential to make installation and distribution of packages working.

8.1.1. Server settings

1. On top menu, go to **Session**  **Settings**. Enable the Button for **package installation** (package) and re-load the page.

2. Click your server device from device-tree on the left side, go into **Config** tab, and activate the **package_server** config.

3. Start the package-server by navigating to **cluster server information** and open the lower dropdown menu called **One Server checked** with click on the arrow. Push "Action" button for **package-server** and choose start if it is not already running.

So far, your server is ready for package installation. Also the clients/nodes have to be prepared for package installation.

Note

Be aware of the fact that all clients with repositories pointing to other networks or to the internet must be able to establish connection to there.

Easy way to ensure this is the usage of **iptables** on the server-side.

8.1.2. Client settings

1. First step to setup package-installation is to enter your server (package-server) IP-address (or hostname) in `/etc/packageserver` on the client machine.

2. Make sure **package-client** service is installed and running on the nodes/clients. To check the status of package-client use **icsw service status** command. Status of package-client should be "running".

   Alternative use **`rcpackage-client status`** to display service status.

8.1.3. Server config file /etc/sysconfig/package-server

The main configuration file for package-server is `/etc/sysconfig/package-server`. The content should be self-explanatory and looks like this:

Table 8.1. package-server config options

options	default value	description
`PID_NAME=`	`package-client/package-client`	Name of PID files
`KILL_RUNNING=`	`True`	
`USER=`	`idpacks`	Username
`GROUP=`	`idg`	Groupname
`GROUPS=`	`['idg']`	
`LOG_DESTINATION=`	`uds:/var/lib/logging-server/py_log_zmq`	Destination of log files
`LOG_NAME=`	`package-server`	Name of log file
`SERVER_PUB_PORT=`	`8007`	Server port for communication with client
`NODE_PORT=`	`2003`	Client port for communication with server
`DELETE_MISSING_REPOS=`	`False`	Capability to deleting missing repos

8.1.4. Client config file /etc/sysconfig/package-client

The main configuration file for package-client is `/etc/sysconfig/package-client`. It content should be self-explanatory and looks like this:

Table 8.2. package-client config options

options	default value	description
`PID_NAME=`	`package-client/package-client`	Name of PID files
`KILL_RUNNING=`	`True`	
`COM_PORT=`	`2003`	Client port for communication with server
`SERVER_COM_PORT=`	`8007`	Server port for communication with client
`LOG_DESTINATION=`	`uds:/var/lib/logging-server/py_log_zmq`	Destination of log files
`LOG_NAME=`	`package-client`	Name of log file
`NICE_LEVEL=`	`15`	Nice level the log daemon running at
`MODIFY_REPOS=`	`False`	Capability to modify repositories

options	default value	description
PACKAGE_SERVER_FILE=	/etc/packageserver	
PACKAGE_SERVER_ID_FILE=	/etc/packageserver_id	

Important

⚠Set "MODIFY_REPOS=False" to forbid repository modification on client side.

8.2. Install packages

There are two common ways to install additional packages.

- Package installation with operating system package manager
- Package installation with package upload in directory

Usually the first method is recommended for standard installation of available packages. All software and packages your running system provides, can be installed via "Package install". It starts your system package-manager in background (apt-get, yum, zypper) and install selected packages on selected nodes.

8.2.1. Install packages using package manager

1. In top menu, go to **Cluster** **Package install**.
2. In **Package_repositories** tab [Figure 8.1, " Package repositories tab "], push the **reload** button to update you repositories.
3. Go to **Package search** tab [Figure 8.2, " Package search tab "] and search for the packages you want to install on the system.
4. If there are some results, list all matching packages with the **show results** button. In below appeared list choose your desired package version by pushing one of the the right buttons (**take exact/take latest**).
5. Go to **Install** tab [Figure 8.3, " Package install tab "], select devices package should be installed for and push "attach" button.
6. On top, a new button named **action** appears. Push the button, choose "Target state" **install** and submit your settings. The package will be installed automatic on your selected nodes.

Figure 8.1. Package repositories tab

Repositories (11 entries, services), reload | Sync to clients | Clear caches

showing entries 1 to 6, show 10 per page, only enabled hide only published show

Name	Alias	Enabled	AutoRefresh	GPGcheck	publish	URL	Pri	Service	action
cluster-devel	cluster-devel	yes	---	yes	yes	http://www.initat.org/cluster/RPMs/suse_13.1/cluster-devel	99		toggle publish delete
extra	extra	yes	---	yes	yes	http://www.initat.org/cluster/RPMs/suse_13.1/extra	99		toggle publish delete
im_non_oss	im_non_oss	yes	---	yes	yes	http://im.init.at/suse/13.1/non-oss	99		toggle publish delete
im_non_oss_updates	im_non_oss_updates	yes	---	yes	yes	http://im.init.at/suse/13.1-non-oss-updates/	99		toggle publish delete
im_oss_updates	im_oss_updates	yes	---	yes	yes	http://im.init.at/suse/13.1-updates	99		toggle publish delete
monit-devel	monit-devel	yes	---	yes	yes	http://www.initat.org/cluster/RPMs/suse_13.1/monit-devel	99		toggle publish delete

Figure 8.2. Package search tab

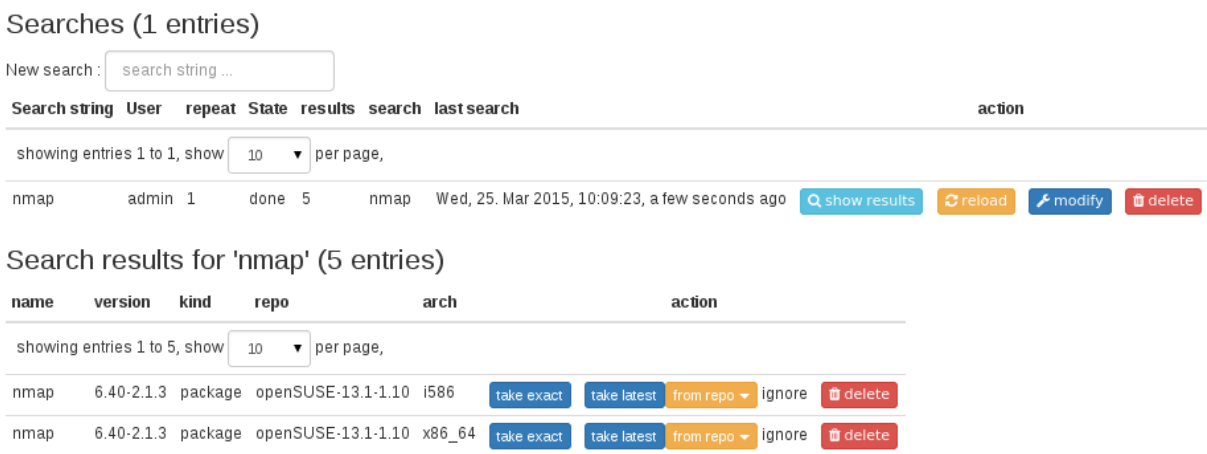


Figure 8.3. Package install tab

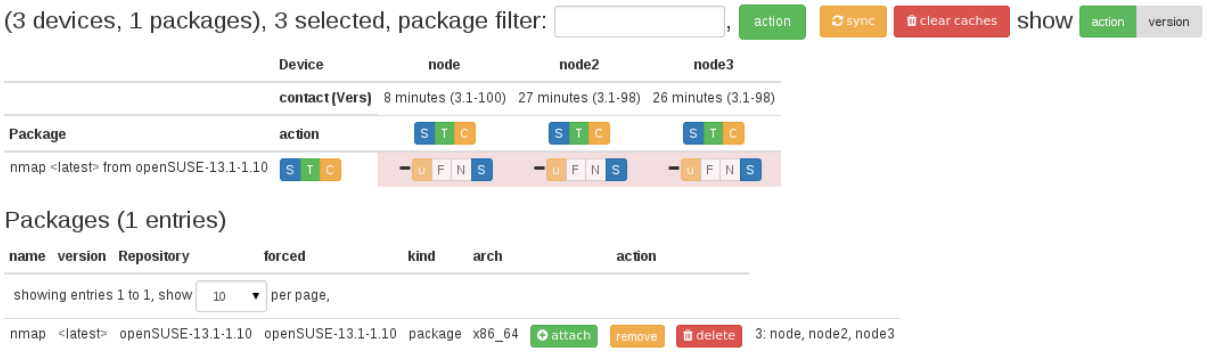
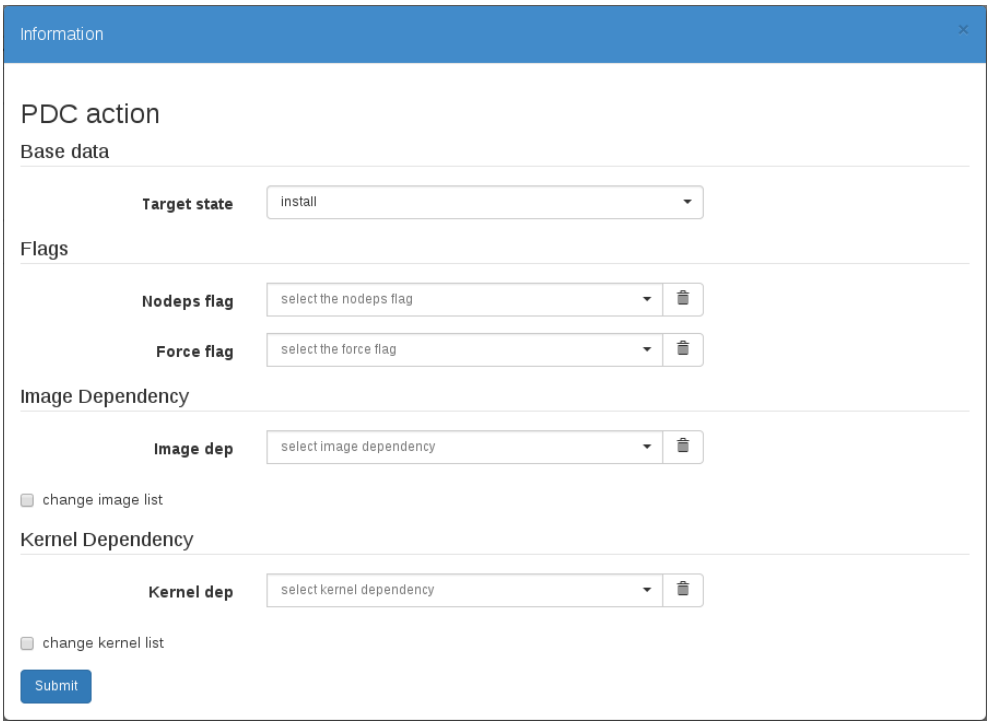


Figure 8.4. Package install action window



8.2.2. Install packages using directory upload

If your system do not provide some packages you really want to install, there is an other way to go. In this special case you can either download fitting binary packages from external sources and place it in the right directory or you can compile and build your own package from sourcecode.

Upload binary packages

1. Upload your package into your upload directory on your server: `/opt/cluster/system/packages/`

   ### Note

   Create the directory in case it does not exist.

2. Execute the update script `update_repo.sh` in `/opt/cluster/system/packages/` to refresh your repositories.

   Script content for **Red Hat** based systems:

   ```
   #!/bin/bash
   cd /opt/cluster/system/packages
   createrepo .
   yum clean all
   yum makecache
   ```

   Script content for **Suse** based systems:

   ```
   #!/bin/bash
   cd /opt/cluster/system/packages
   createrepo .
   zypper ar /opt/cluster/system/packages local_repository
   zypper ref
   ```

3. Maybe you have to "Sync to clients"/"Clear caches" to get the new repositories on all nodes.

4. Now, if you search after uploaded package you should get some results. To install uploaded packages follow the same procedure as install packages from system package manager mentioned in Section 8.2.1, “ Install packages using package manager ”.

Compile, make and upload packages from source

1. Download source files and extract it.

2. Compile your software as usual and install it (.configure ; make; make install).

3. Once your package is installed, use **make_package.py** to create a new *.rpm package.

4. Run the **update_repo.sh** to refresh your repositories.

5. Maybe you have to "Sync to clients"/"Clear caches" to get the new repositories on all nodes.

6. On top, a new button "action" appears. Push the button, choose "Target state" install and submit your settings. The package will be installed automatic on your selected nodes.

8.3. Delete packages

To delete packages do following steps:

1. In top menu navigate to **Cluster**  **Package install**, and choose the **Install** tab.
2. Select packages and nodes to delete it from.
3. Push the **Action** button and choose **erase** from **Target state** dropdown menu. To finish deletion click on the **Submit** button.

Chapter 9. RMS - Resource Management System

An essential aspect in **NESTOR®** is the job management system. Main reason for using clusters is a higher computing power to calculate jobs. The calculation of data will be splitted into pieces and every node or slot can calculate each piece separately, this results in a higher speed of calculation. The organisation of slots, cluster and jobdistribution is done by the *SGE - son of grid engine*. SGE provides special commands and tools to control jobs distributed to the nodes.

The RMS is the coupling between the SGE and our web front-end. With enabled RMS you are able to manage jobs without any using of SGE commands.

9.1. Introduction of RMS

Like mentioned before, the RMS is a powerful addon for managing jobs on clusters. It consists of packages and services working together to provide management functions for transmitted jobs.

Important parts of RMS are:

SGE part

- SGE - Son of Grid Engine
- Commandline tools like:
  - qdel
  - qstat
  - qacct

  Look command-reference or manual page of `sge_intro` to show complete list of commands.

  man sge_intro

init.at part

- rms-server - Server between SGE and Webfrontend
- Webfrontend
- Commandline tools like:
  - sjs
  - sns

Both commands sjs and sns are links to `/opt/cluster/bin/sgestat.py`.

9.1.1. Environment variables

Environment variables for setting up RMS can be found under `/etc/`

- `/etc/sge_cell`

  Name of SGE

- `/etc/sge_server`

  Hostname or IP address of sge server.

- `/etc/sge_root`

  Directory sge installs to.

9.1.2. Installation of RMS

To get RMS working it is not enough only to install the package, you must also edit some config files and build the SGE part manually. Below step by step how to install RMS will help you installing RMS and run the required services.

Even it should be obvious, before you are able to install RMS make sure you already installed **NESTOR®** and its dependencies.

1. RMS package comes with installation of **icsw-server**
2. Set environment variables in `/etc/sge_cell`, `/etc/sge_server` and `/etc/sge_root`

   Setting of environment variables must be done **before** compiling SGE!

3. Download the latest version (Latest version for 2014.09.25 is 8.1.7) of SGE package from https://arc.liv.ac.uk/trac/SGE

   wget http://arc.liv.ac.uk/downloads/SGE/releases/8.1.7/sge-8.1.7.tar.gz

4. Extract `sge-8.1.7.tar.gz` archive to `/src/`, change into extracted directory and run our buildscript placed under `/opt/cluster/sge/build_sge6x.sh`.

   tar xzf sge-8.1.7.tar.gz

   cd /src/source/

   /opt/cluster/sge/build_sge6x.sh

   If your system can not compile and output some error messages, make sure you already installed necessary build-tools and development packages. Dependent of your operating system package names and count could differ.

5. Now directories under `/opt/sge62` exists and service **sge_qmaster** is running.

   Test if sge_qmaster is running:

   ps aux | grep sge_qmaster

6. Set $PATH variables by running script located under `/etc/profile.d/batchsys.sh`

   . /etc/profile.d/batchsys.sh

7. Run followed scripts:

   /opt/cluster/sge/create_sge_links.py and **/opt/cluster/sge/modify_sge_config.sh**

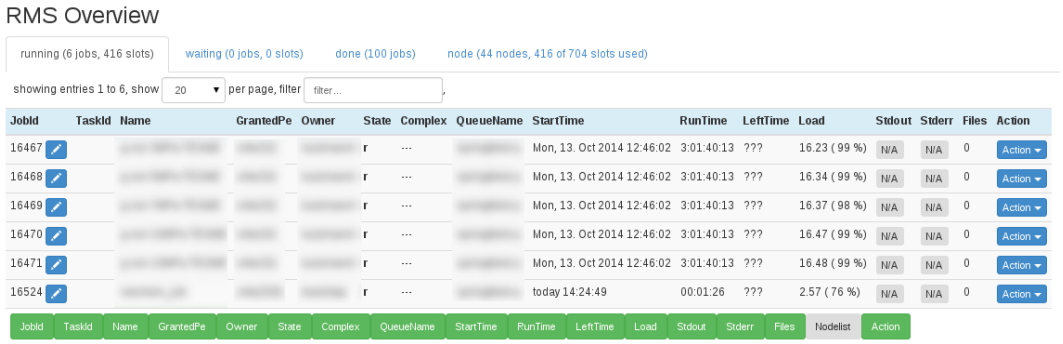
9.1.3. RMS web front-end

RMS overview provides 4 tabs. Not only for displaying information but also to control jobs. There are a couple of green buttons on the bottom of overview page to hide or unhide columns.

Running jobs

The first tab of RMS overview displays current **running** jobs in the grid engine. You can get some background information like jobids, owner, runtime or nodelist of each job. On the right side there is an action button to delete or force delete running jobs.

Figure 9.1. RMS running jobs

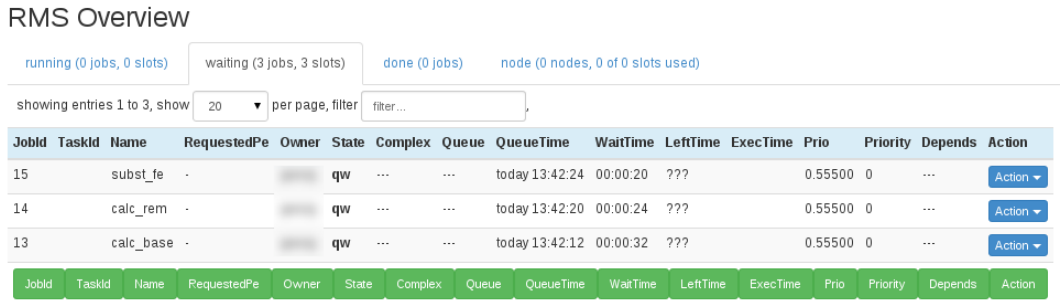


Current running jobs with disabled nodelist column

Waiting jobs

The second tab of RMS overview displays the current **waiting** jobs. This are jobs waiting in the SGE queue for execution. Among other infos, it shows the "WaitTime", "Depends" and the "LeftTime".

Figure 9.2. RMS waiting jobs

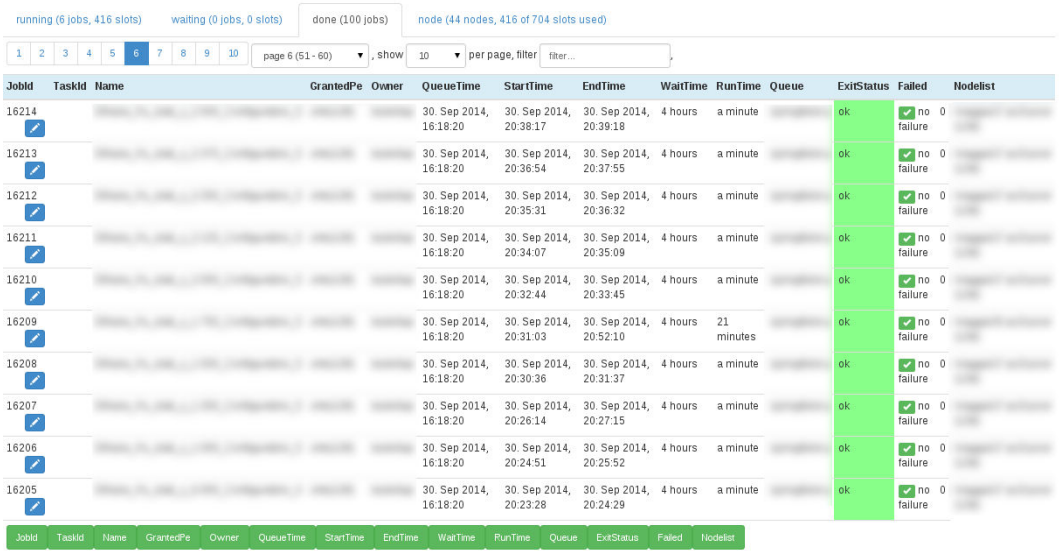


Current waiting jobs

Done jobs

The third tab of RMS overview displays **done** jobs and specific columns like "ExitStatus", "Failed" or "Run-Time".

Figure 9.3. RMS done jobs

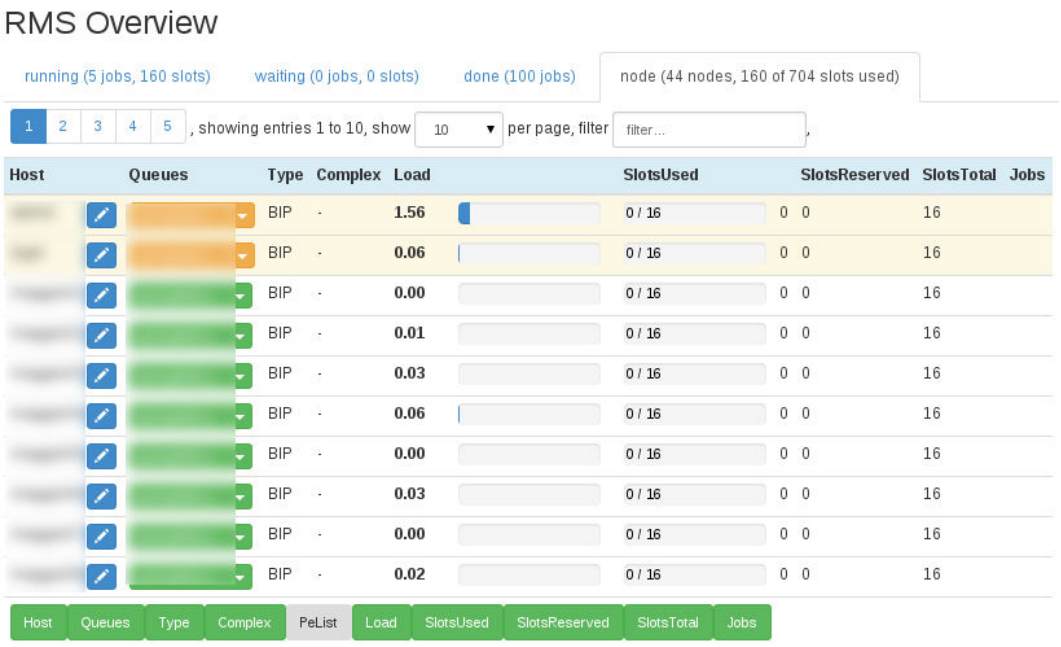


Done jobs

Nodes

The fourth tab of RMS overview displays the **nodes** itself. You can enable or disable queues or, if it exists, display graphs of choosen nodes.

Figure 9.4. RMS nodes



Node overwiew

9.2. Job management system in SGE

For direct usage of the SGE, there are a couple of commands:

9.2.1. SGE commands

Commands the SGE provides are:

qacct

qacct extracts arbitrary accounting information from the cluster logfile.

qalter

qalter changes the characteristics of already submitted jobs.

qconf

Queue Configuration, allows the system administrator to add, delete, and modify the current Grid Engine configuration, including queue management, host management, complex management and user management.

qdel

Provides a means for a user/operator/manager to delete one or more jobs.

qevent

qevent provides a means of watching Grid Engine events and acting on jobs finishing.

qhold

Qhold holds back submitted jobs from execution.

qhost

qhost displays status information about Grid Engine execution hosts.

qlogin

qlogin initiates a telnet or similar login session with automatic selection of a suitable host.

qmake

qmake is a replacement for the standard Unix make facility. It extends make with an ability to distribute independent make steps across a cluster of suitable machines.

qmod

qmod allows the owner(s) of a queue to suspend and enable queues, e.g. all queues associated with his machine (all currently active processes in this queue are also signaled) or to suspend and enable jobs executing in the queues.

qmon

qmon provides a Motif command interface to all Grid Engine functions. The status of all, or a private selection of, the configured queues is displayed on-line by changing colors at corresponding queue icons.

qping

qping can be used to check the status of Grid Engine daemons.

qquota

qquota provides a status listing of all currently used resource quotas (see sge_resource_quota(5)).

qresub

qresub creates new jobs by copying currently running or pending jobs.

qrls

qrls releases holds from jobs previously assigned to them e.g. via qhold(1) (see above).

qrdel

qrdel provides the means to cancel advance reservations.

qrsh

qrsh can be used for various purposes such as providing remote execution of interactive applications via Grid Engine comparable to the standard Unix facility rsh, to allow for the submission of batch jobs which, upon execu- tion, support terminal I/O (standard/error output and standard input) and terminal control, to provide a batch job submission client which remains active until the job has finished or to allow for the Grid Engine-controlled remote execution of the tasks of parallel jobs.

qrstat

qrstat provides a status listing of all advance reservations in the cluster.

qrsub

qrsub is the user interface for submitting an advance reservation to Grid Engine.

qselect

qselect prints a list of queue names corresponding to specified selection criteria. The output of qselect is usually fed into other Grid Engine commands to apply actions on a selected set of queues.

qsh

qsh opens an interactive shell (in an xterm(1)) on a low loaded host. Any kind of interactive job can be run in this shell.

qstat

qstat provides a status listing of all jobs and queues associated with the cluster.

qtcsh

qtcsh is a fully compatible replacement for the widely known and used Unix C-Shell (csh) derivative tcsh. It pro- vides a command-shell with the extension of transparently distributing execution of designated applications to suitable and lightly loaded hosts via Grid Engine.

qsub

qsub is the user interface for submitting a job to Grid Engine.

For more information please take a look into the well written man pages of each *"q"* command.

9.2.2. Job submission via command line

Common way to submit jobs to the cluster is to use grid engines *"q"* commands. Assumed that your cluster configuration is correct, running jobs on cluster is as easy as running jobs on local machines.

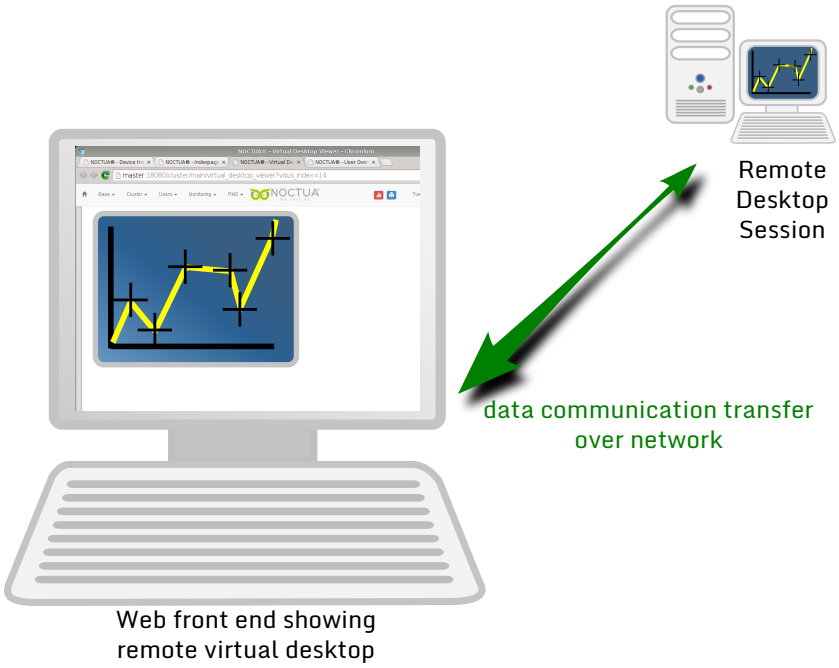
Following steps have to be done to transfer jobs to queue: COMMING SOON

9.2.3. Job submission via web front-end

Chapter 10. Virtual Desktop

Virtual desktop is a technology to transfer display output from remote graphic cards to your local machine graphic card.

Figure 10.1. VNC functionality



Basic illustration to explain vnc technology

User are often forced to work on remote machines because of computation power, license issues or simply geographical distance. In this cases, user usually have to start their remote desktop manually via a command line or similar tools.

With our virtual desktop technology there is no need to manually start anything. The back-end of the server takes care of sessions, ports, passwords etc., makes the relevant settings and saves it in the global database for you. Not only the settings and configurations will be done automatically by the back-end but also in cooperation with the web front-end it provides the display output.

That way you are able to access and work on remote machines via the web front-end on your favorite browser.

10.1. Prerequisites

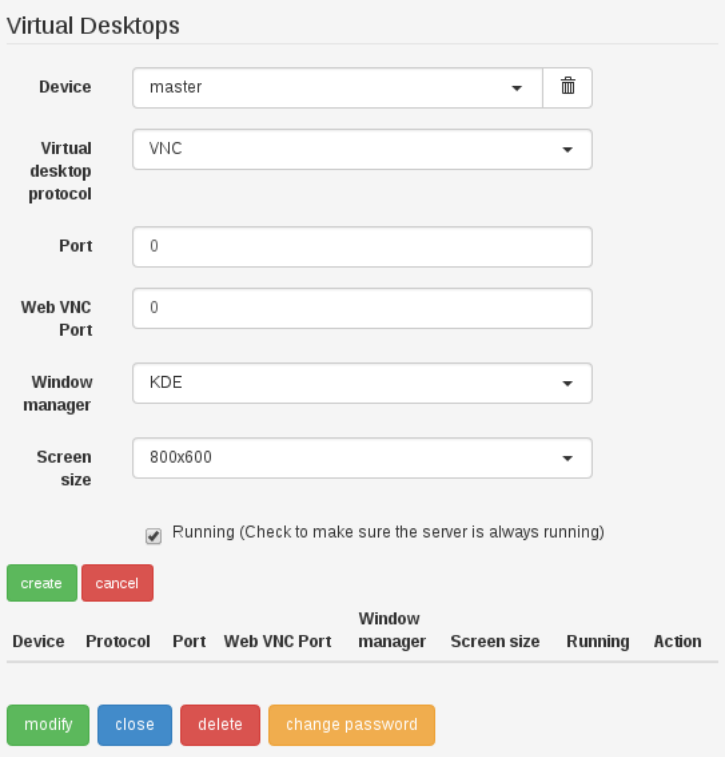
Important

⚠There must be the same **user** with the same **user_id** within the user management system of your server and on the machine which starts the vnc-server session.

If it is not true, virtual desktop session can not be used.

To activate the virtual desktop technology, first of all you have to define a **Virtual Desktop session** in **User Management**. In the main menu on top of the page navigate to **Users** **Overview**, do left mouse click on the admin user.

Figure 10.2. Virtual desktop session



Before using virtual desktops you have to define a session for it.

Virtual desktop settings

Device	Please insert text here...
Virtual desktop protocol	Protocol which will be used for virtual desktop session
Port	Portnumber of connecting client - if set to "0", port will be random
Web VNC Port	Portnumber of vnc server
Window manager	Window manager system for systems with more than one window manager
Screen size	Preset of virtual desktop size. It's the windowsize the virtual desktop will be displayed into.
Running	Checkbox to make sure the server is always running

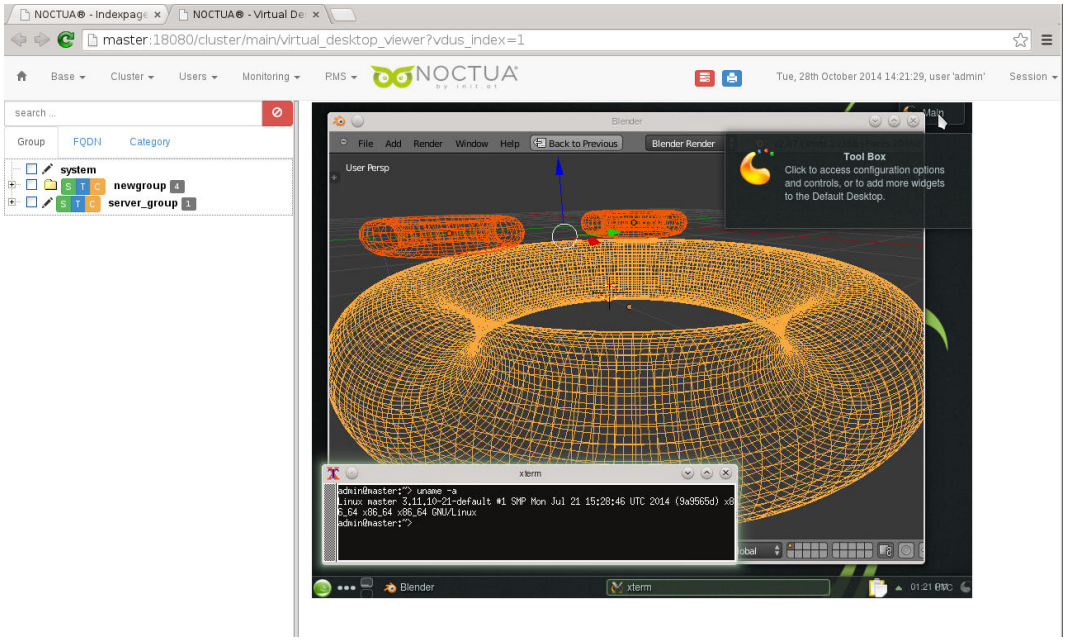
After at least one virtual desktop session is defined, the back-end takes control of the further process. It looks continuously every 5 minutes for a running vnc-server. After discovering a running vnc-server, there will be new entries and buttons in virtual desktop tab.

Now you have the choice to view your remote desktop in the main home page or in a new browser tab.

10.2. Connect to virtual desktop

Connection to remote desktop is as simple as login to your local system, even more simple like this. Just push one of the buttons and enjoy your virtual desktop inline or in new opened tab.

Figure 10.3. Virtual KDE Session



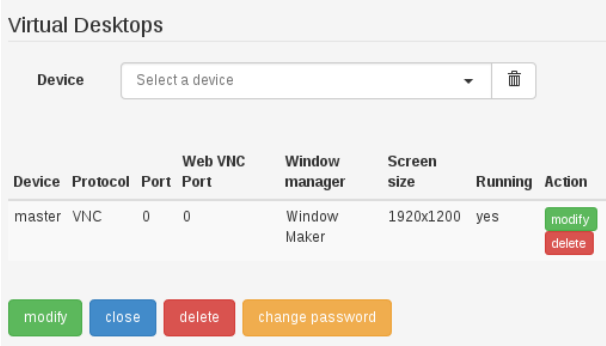
KDE session inside web front end with started 3D software and xterminal

10.3. Change settings

To change your window manager or change the virtual desktop screen size, simply navigate to **Users Overview** and choose the user of virtual desktop session.

Scroll down to section "Virtual Desktops", change setting and push the modify button to change settings.

Figure 10.4. Modify settings for vnc



Important

⚠Keep in mind that running processes and programms will be halt after modifying virtual desktop settings.

Chapter 11. Cluster setup

NESTOR® assists system administrators to set up and manage their cluster with comfortable support and help of the web front-end. Web front-end, GUI? It really do. Imagine you have set up each of the single cluster machine by hand. Who have enough time to do that? First group have the time to do it, but does it flows and goes on or is it a difficult birth?

Typically a cluster consists of many single nodes, each connected over network interfaces building together a cluster. For easier understanding, the following section refers to one server and only three single nodes. Other nodes can be set up exactly the same way, or, if node installation varied, with small adaption to it.

11.1. Basic requirements

Before we are able to run a cluster installation, we have to make sure that the basic system services and components are working regulary. Following necessary requirements are given to set up a cluster.

Needed system services	Services and daemons provided by the operating system
Needed cluster services	Services and daemons provided by **NESTOR®**
At least two different networks	We need two separate networks, one for internal communication with the nodes, and an other for external communication. Find further information in section Separate network

11.1.1. Needed system packages

⚠ All following services and package names are referred to a SUSE distribution. Notations for other distributions may differ.

dhcp-server

Automatically distribution of IP address will be done by *DHCP* requests. Therefore we must install and run a DHPC daemon on our **NESTOR®** server.

tftp

Because of required network access on node and server side, and in order to be able booting single nodes over network, we have to run a *TFTP* daemon.

nfs-kernel-server

Function of the network file system server (*NFS*) is to supply file access over ethernet. While tftp, due to the fact that it's very simple, is responsible for node boot process, responsibility of nfs is the really high performance file transfer over ethernet.

xinetd

The *xinetd* daemon is responsible for starting services after got network requests on defined ports.

11.1.2. Install required system services

To install all above mentioned and required services on SUSE systems, use the following command line: **zypper ref; zypper in dhcp-server tftp nfs-kernel-server xinetd**

Use the common operating system package manager for other distributions.

11.1.3. Configuration of system services

It is not enough only to install the needed services, they also have to be configured properly to make a **NESTOR®** cluster installation possible.

DHCP configuration

To start with, insert one of your **NESTOR®** network devices into the dhcpd configuration file located in `/etc/sysconfig/dhcpd`

```
DHCPD_INTERFACE="eth1"
```

Start or restart the daemon e.g with **rcdhcpd restart**

Later you have do do extended configuration with help of

```
cluster-server.py  -c write_dhcpd_config  -D authoritative:True
```

TFTP xinetd configuration

Set the value for **disable** in `/etc/xinetd.d/tftp` to **no**.

Also set the value for **server_args** in `/etc/xinetd.d/tftp` from **/srv/tftpboot** to **/tftpboot**. Restart the *xinetd* daemon with **rcxinetd restart**

Create sybolic links for tftpboot

We have also to create a symbolic link from / to `/opt/cluster/system/tftpboot` and name it **tftpboot**.

Start all necessary services

Make sure all mentioned services start properly, even after your cluster server reboots. To start services in open SUSE use following command: **`rcnfsserver start`**. To check if a service is already running in open SUSE use following command: **`rcnfsserver status`**.If you can not start a service, for example the *nfsserver* it may be helpful to start **rpcbind** first.

11.1.4. Needed NESTOR® packages

Also the following necessary **NESTOR®** packages have to be installed first in order to get a running cluster installation setup.

mother

One of the most important main parts of **NESTOR®**. Mother is responsible for:

- Scanning it's kernel directories and adds the new kernel into the database.

discovery-server

As the name supposes, in context of **NESTOR®**, this service ease the cluster network setup.

cluster-config-server

cluster-server

package-server

Manages installation of software packages over the web front-end. You can find more details about it in section **package-install**

logcheck-server

11.1.5. Install required NESTOR® packages

Install all needed packages with an one line command like done for system packages. To do it on SUSE systems, use the following command line:

zypper ref; zypper in icsw-server

11.2. Node installation requirements

We need three indigrients to "cook" our node installation soup.

partition	System partition template
kernel	**LINUX ™** kernel and initrd with essential kernel modules
image	Basic image which should be used for the nodes

11.2.1. Create the partition

Create a new partition table under **Cluster**  **Partition overview**. Important aspect here is the Nodeboot check box explained further down.

Figure 11.1. Partition overview table

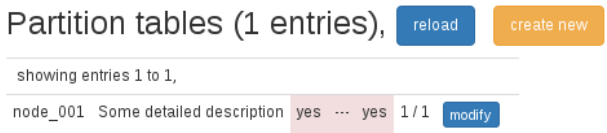
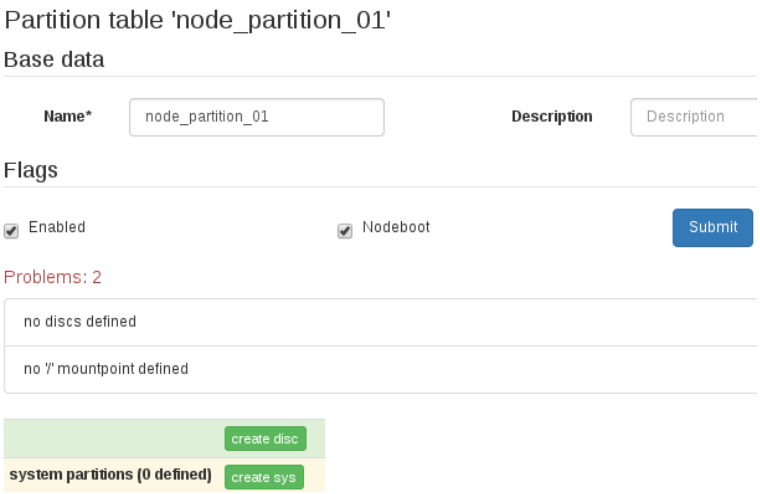


Figure 11.2. Create partition



Columns in partition table overview

Name Partition name

Description More precise description of the partition

Insert a name for the partition, and, if you want, insert also a partition description. Two problems will be immediately displayed on the web front-end. In fact no defined discs and no present mountpoint. To solve this problems do following:

1. **create new disk**

2. **create new partition, set partition numer and set valid mount point**

   Set the size to "0" to use the whole disk.

Please pay attention to the naming of discs which orientates on **UNIX** conventions and therefore used the common devicename standard like: `/dev/sd`

The first part */dev/* represents the device tree. The second part *sd* represents a mass storage device like HDD od SSD. To create a disk simply insert a letter between *a* (first registered device) and *z* (last registered device).

⚠Do not insert any partition number to the device name. This will be done in the create partition dialogue.

After creating a disk, create partition button will be visible.

Make sure to set the *nodeboot* flag of the partition.

11.2.2. Create the kernel

Next, we need a LINUX™ kernel. It will be handled like a template kernel for the nodes. There are two common ways to create a kernel. You can either compile a completely new kernel or use an existing one.

Copy existing kernel with copy_local_kernel.sh script

The **copy_local_kernel.sh** script located in `/opt/cluster/bin` is designed to copy the kernel and it's belonging modules of the cluster server into the right directories.

The script also copies the *system.map*, the kernel *config* files, the kernel *modules*, kernel *firmware* and also generates a dummy *initrd* file.

`copy_local_kernel.sh [ KERNEL_NAME ] [ KERNEL_DIR ]`

In case you got following error message:

```
system target directory /opt/cluster/system/tftpboot/kernels does not exist
```

try the command with the **--init** flag: **copy_local_kernel.sh --init**

Manual copying kernel

Copy the compiled kernel to the newly created folder. Rename it to bzImage. Copy the System.map to the folder. Copy the config file of the kernel to the newly created folder . Rename it to .config. If you are running the same kernel you can do zcat /proc/config.gz > .config Copy /lib/modules of the desired kernel to the newly created folder + "/lib". Copy /lib/firmware to the new folder "/lib" + firmware + "kernel-name". Create a bz2 tar of the ./lib/modules tar -cj lib > modules.tar.bz2 Copy the firmware to lib/firmware/(kernel-version)

Create initrd for kernel

In order to get every single part for a proper booting system together we have to create a suitable **initrd**. ⚠For booting procedure essential modules and driver (e.g. filesystem driver, network driver etc.) have to be included into the *initrd* to make the boot proces work.

Run the script **populate_ramdisk.py** to create the initial ramdisk.

`populate_ramdisk.py [ -m LIST_OF_MODULES ] [ KERNEL_DIRECTORY ]`
Use **`populate_ramdisk.py -L -L`** and **`populate_ramdisk.py -L`** to show all the generated ramdisks and the included modules.

The list of built in modules will most likely be the modules used for your node networkcard or filesystemdriver. Watch for file missing errors and install the missing binaries to your system. killall for example would be found in the package psmisc. Restart mother. The mother scans its kernel directory and adds the new kernel into the database. The populate_ramdisk.py script copies the stages scripts /opt/cluster/lcs and some binaries into an initrd + kernel usable for pxe booting. Now you can associated your first node with the new kernel via operation > nodeboot.

So our command to populate initrd should look like this:

```
populate_ramdisk.py -m virtio_net,virtio_pci --set-master-server -i /tftpboot/kernels/3.11.10-25-default/
```

11.2.3. Create the image

The third component you absolutely need to run a cluster and install nodes over network is an image. It contains all needed software packages for your node installation. We need a basic image which will be served to the local nodes and represents a pattern for all node installations. Also here we have to choose if we want to install all image components manually or if we want to be assisted by a script named **make_image.py**.

Make image directory structure

First of all we need an image directory structure. A good choice to start with is `/opt/cluster/system/`

Within, create following subdirectories:

```
/opt/cluster/system/images
```

```
/opt/cluster/system/images/suse_131/etc/zypp/repos.d
```

Copy local repositories

Copy your local repository files into the just created image repository directory.

cp -a /etc/zypp/repos.d/* suse_131/etc/zypp/repos.d/

Install all needed image packages

Now we have to install the os packages into our created directory. First make a *refresh*(ref) and then install your desired packages.

zypper -R /opt/cluster/system/images/suse_131/ ref

zypper -R /opt/cluster/system/images/suse_131/ in icsw-client icsw-server icsw-dependencies bash aaa_base glibc net-tools sshd openssh psmisc util-linux pam libxml2-tools loadmodules vim

Finally create your new image

After all needed packages are downloaded, start creating the image with help of **make_image.py -v --build-image**

11.3. Device and network configuration

Apart from required services, kernel image and partitions, it is totally necessary to integrate our nodes into our network and peering. But first we have to build up a network configuration consists of two serarate networks. Once handware network settings were done we can switch to the web front-end.

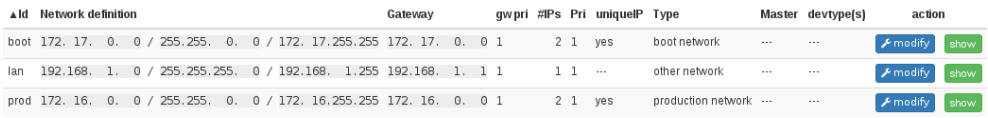
11.3.1. Two different networks and peering

In order to booting nodes over *PXE* we have to define two separate networks. The first network is intended for the boot procedure of the node. The second network will be used for the node installation over *NFS* and later in productive state.

We need this two different networks both as setting in **NESTOR®** and as real hardware network settings.

Below figure shows you a typically **NESTOR®** network setting for one single **cluster server** and two different **networks**. Of course, there are three networks if we also count the local area network but for our purpose we concentrate only on the **class B** networks beginning with **172**.

Figure 11.3. Two different networks



One boot and one prod(uction) network defined

Also important is, that your cluster server really has the same hardware network settings. In our example, we use two network devices(eth1 and eth1:prod) providing our two different networks for booting and productive and additional the local area network (eth0). So we get following **ifconfig -a** output:

```
eth0      Link encap:Ethernet  HWaddr 32:34:03:32:32:DD
          inet addr:192.168.1.239  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::3034:3ff:fe32:32dd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
          RX packets:71510 errors:0 dropped:10 overruns:0 frame:0
          TX packets:10156 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7598935 (7.2 Mb)  TX bytes:5595522 (5.3 Mb)

eth1      Link encap:Ethernet  HWaddr 52:54:00:DE:96:95
          inet addr:172.17.1.1 Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::5054:ff:fede:9695/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:179865 errors:0 dropped:0 overruns:0 frame:0
          TX packets:193583 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17279193 (16.4 Mb)  TX bytes:306664808 (292.4 Mb)

eth1:prod Link encap:Ethernet  HWaddr 52:54:00:DE:96:95
          inet addr:172.16.1.1  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

Adding nodes

The **device network** settings are also very important to boot nodes over network. That are the settings which must be done over web front end. Each node needs following device network settings:

- One network device
- Two IP addresses, each for boot and prod
- Peering connection to the cluster server

For a better understanding there are already two nodes (node, node2) with a complete correct setup. Now we have to setup node3 the same way.

Lets start with adding a new node. In web front end, navigate to **Base** **Device tree**. Create a new device, create optional a device group before, and name it. Essential settings here are:

1. **Bootserver**

   This should be the machine our nodes boots from

2. **Root passwd**

   This will be your node root password, if leaved empty the default password will be **init4u**

Note

Root passwd can be set not only for first installation of nodes but also for every boot procedure.

After created a new node device, navigate to **Base**  **Device network** to create a new netdevice, assign two IP addresses to it and set the peering.

Figure 11.4. Comparison of complete (node, node2) and incomplete (node3) settings

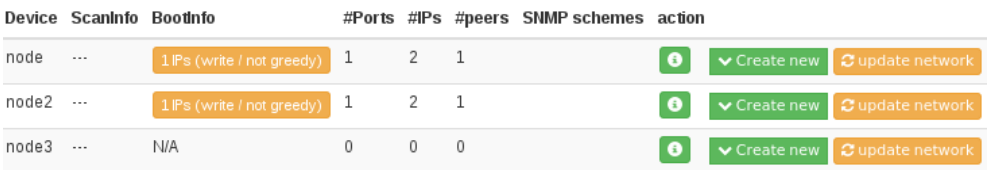


Figure shows node3 without any netdevice, IP address and peer

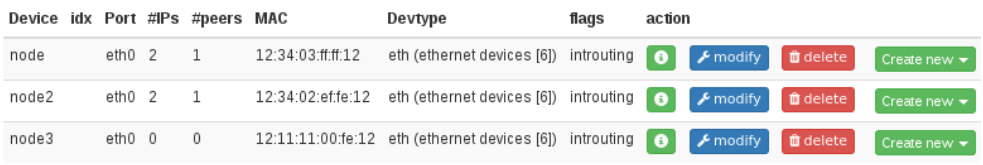
1. Click on the **Create new** button and create a new **netdevice**.

   In hardware tab type in the **MAC address** of your device. This is the only way your node can be identified by your cluster server because it has no IP at booting time.

   Also type in your network driver, for example **virtio_net** if you want to boot a virtual machine with this driver.

   Mark the **inter device routing** checkbox in the basic settings tab. After creation of your new netdevice

   Figure 11.5. New created netdevice for node3



   Node3 still without assigned IP address and peer

2. Create two IP adress for your device, each for boot network and for prod network by click on the **create new** button beside of your network device and choose **IP** from the drop down menu.

   Type in your desired IP adress and choose the right network.

3. Last but not least you have to create a peer or network topology connection to make sure the node is connected with the cluster server.

   Like above, click the **Create new** button, but this time choose *network topology connection* to make a link between node3 and cluster_server.

11.3.2. Server and node configuration

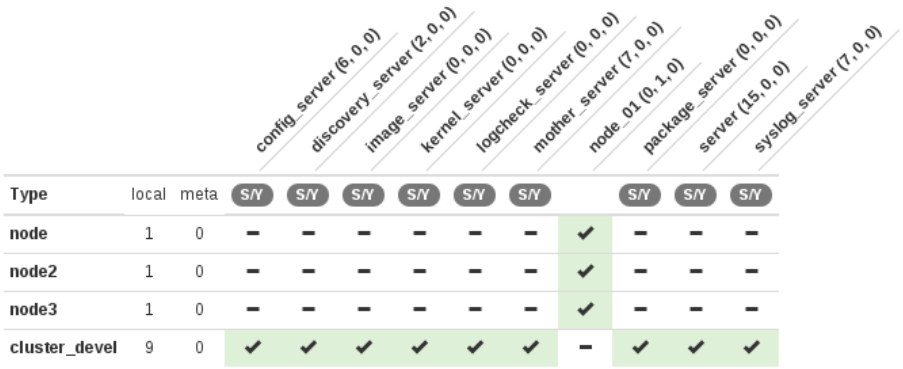
An other important point in context of cluster setup is the right device configuration. Here, we enable the **NESTOR®** cluster software, especially the cluster server device, to act as an cluster server. Analogue is true for the nodes.

You have to enable at least following device configurations for the server:

- server
- mother-server
- kernel-server
- image-server

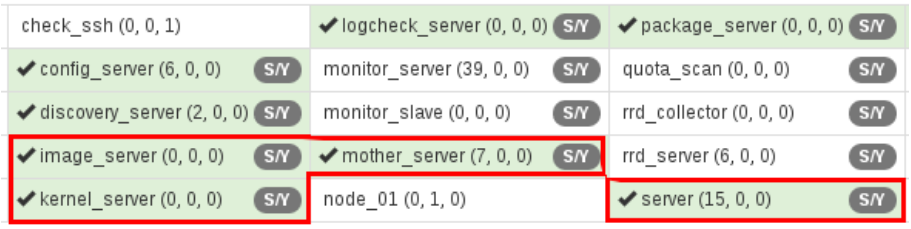
Also you have to enable node configuration for each node

Figure 11.6. Table mode



Configuration for cluster server and nodes

Figure 11.7. List mode



Configuration for cluster server in List mode view

11.3.3. Nodeboot

After all packages are installed, configured and are running, it is time for the very funny part of **NESTOR®**. To control nodes before and after installation there is a tool called **Cluster** **nodeboot**. Nodeboot enables you to control your nodes in every way. You can also display log lines for each node or with the **macbootlog** button for all nodes. It is possible to control one single node or to control all nodes at once. Nodeboot allows you to pick one of the following:

- target state
- kernel
- image
- partition
- bootdevice

Nodeboot provides in addition to above choices a **soft control** button, an **hard control** button, an **action** button and a **log** button.

Figure 11.8. Overview of all existing nodes

Nodeboot with two selected devices, ready for global actions

Use the "STC" global action buttons or the "sel" buttons to select nodes for modifications or for actions.

Possible **soft controls** for nodes are:

- reboot
- halt
- poweroff

Possible **hard controls** for nodes are:

- 
- 
- 

Content of the **modify** button depends on weather a header button is selected or not. For all selected header buttons the **modify** popup window looks like this:

Figure 11.9. Nodeboot modify popup window

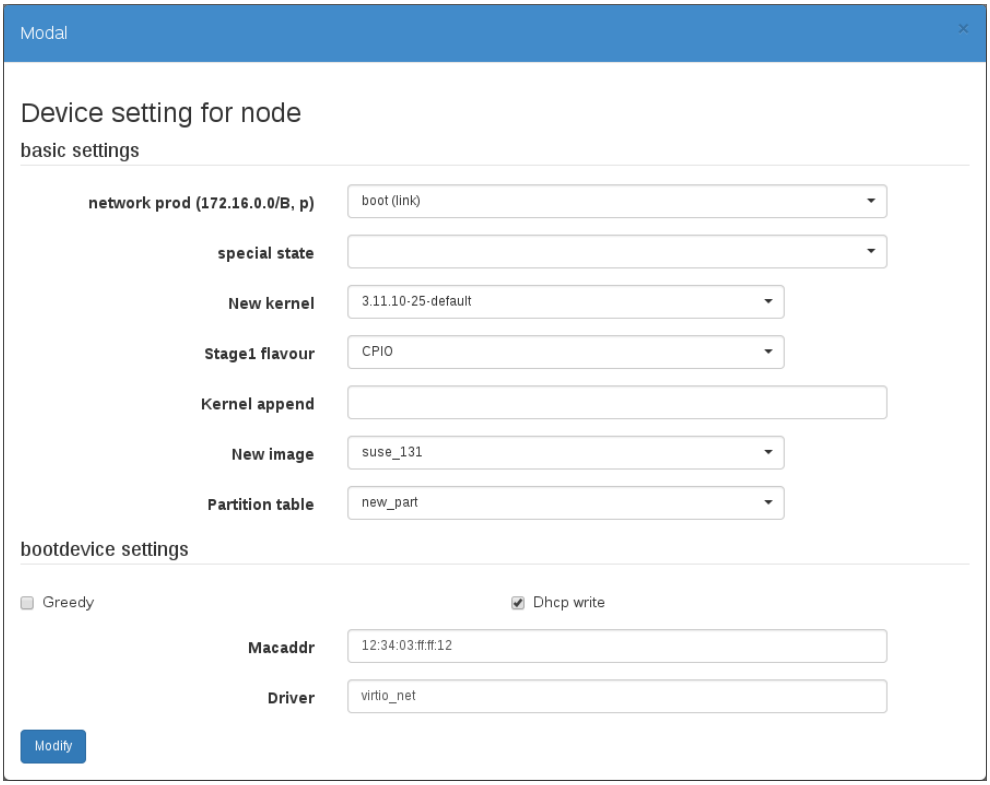


Table 11.1. basic settings

Target state	**boot (link)**	Boot the node
	installation (link,ins)	Install node
	boot_clean (link,ratain)	Boot the node
	installation_clean (link,ins,ratain)	Install node
special state	**mentest(mem)**	Run memory test at boot time
	boot_local(loc)	
	boot_iso (iso)	Boot from CD or DVD ROM?
New kernel	kernel_1, kernel_2, ...	List of availabe kernels
Stage 1 flafour	CPIO	
	CramFS	
	ext2 via loopback	
Kernel append	-	
New image	Image_1, Image_2, ...	List of available images
Partition table	Partition_table_1, Partition_table_2,...	List of available partitions

Table 11.2. bootdevice settings

greedy	-	Checkbox to activate
Dhcp write	-	Checkbox to activate
Macaddr	-	MAC address of the node
Driver	-	Network driver of the node

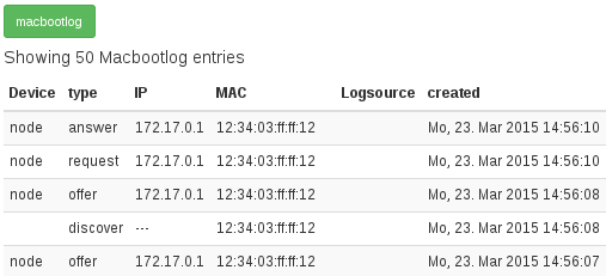
Logging is also possible with nodeboot. To display your nodes logging messages simply click on the **show** button on the right side.

Figure 11.10. Opened logging table

Source	User	Status	Number of log lines: 1368, show 5	when
	---	ok	built config in 0.84 seconds	a few seconds ago
	---	ok	*got partition created after 1 request(s)	a few seconds ago
	---	ok	*got target_state boot (prod_net prod, rsync-state is disabled, no compression) after 1 request(s)	a few seconds ago
	---	ok	mounting config	a few seconds ago
	---	ok	Requesting modules	a few seconds ago
	---	ok	start syslog	a few seconds ago

An other useful logging view is **macbootlog**.

Figure 11.11. Macbootlog



Chapter 12. Extensions

One of the main advantages in contrast with proprietary software is the ability to extend or adapt functionality to user-defined targets. There are some documented APIs which allows you to customise and optimise the work-flow and integration into your companys facility. 

12.1. Shared script directories

There is a place for user scripts under `/opt/cluster/share` 

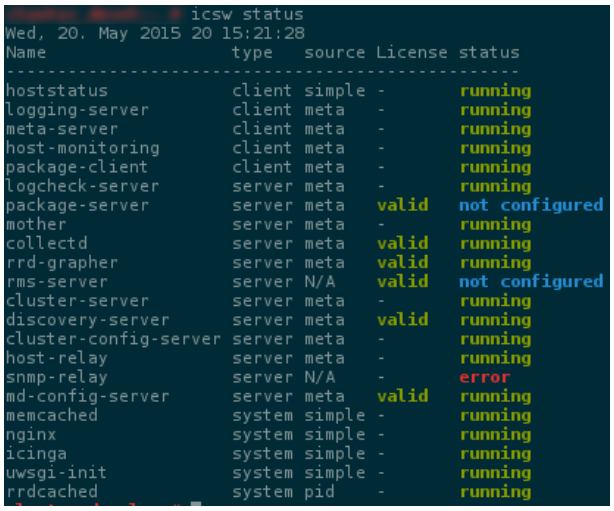
Chapter 13. Debugging and Error hunting

13.1. General information

To obtain information about the general status of your server use **icsw service status**.

Example 13.1. Using icsw status to view services status

Figure 13.1. icsw status command



13.2. Show errors

To show the last errors from the logfile you can use **lse** .

`lse` [-l *`Error number`*]
For more information type **`lse --help`** .

Example 13.2. Using lse to display the last error

```
clusterserver:~ # lse -l 1


Found 40 error records
Error 40 occured yesterday, 17:12:47, pid 11507, uid/gid is (30/8 [wwwrun/www]), sou
  0 (err) :    IOS_type             : error
  1 (err) :    args                 : None
  2 (err) :    created              : 1409152367.94
  3 (err) :    exc_info             : None
  4 (err) :    exc_text             : None
  5 (err) :    filename             : routing.py
  6 (err) :    funcName             : _build_resolv_dict
  7 (err) :    gid                  : 8
  8 (err) :    levelname            : err
  9 (err) :    levelno              : 40
 10 (err) :    lineno               : 179
```

```
11 (err) :    message              : device 'METADEV_server_group' (srv_type grapher) has
```

13.3. Node information

Retrieving node information in an automated fashion is often useful in hunting down errors and bugs. To retrieve information about the nodes use **collclient.py** .

`collclient.py` [--host *Nodename*] [command]

For more information execute **`collclient.py --help`**

Example 13.3. Retrieving information from nodes

```
clusterserver:~ # collclient.py --host node01 df
```

13.4. Logging

The server provides its own logging service. Like usual in *NIX environments there are special directories log-files will be written into. Access to these log files is given by the command **lse**. Of course it is also possible to read the logfiles directly by your favorite editor.

13.4.1. Directories for log files

In case of something goes wrong the logging-server writes its logs under `/var/log/cluster/logging-server/[HOSTNAME]/`.

Denotation of log files and subdirectories is related to the service which writes the log. For example if the meta-server can not start some service it will write its log into the directory

If you want to see background information for package-installation on some nodes the file you have to check is `package-client`. Analogue this is true for server side, this time the filename is `package-server`

Files called `*.bz2` are compressed logging backup files.

13.4.2. Automatic log mail delivery system

Critical errorlogs will also be delivered by mail. So you do not have to check your logs permanent, you will be notified by mail about critical errors.

Setting for recipient of errorlog mails is stored in `/etc/sysconfig/logging-server`.

Another configuration file for mail notification is `/etc/sysconfig/meta-server`.

Replace the given mailaddress in the line containing **TO_ADDR=** with your desired mail address.

By uncommenting and editing the line beginning with **#FROM_ADDR=** you are able to set the *sender "From"* name of recived emails.

```
# from name and addr
FROM_NAME=pythonerror
#FROM_ADDR=localhost.localdomain
# to addr
TO_ADDR=mymail@gmail.com
# mailserver
```

```
MAILSERVER=localhost
```

After editing the logging-server configuration file, the logging-server daemon must be restarted:

icsw service restart logging-server

The new configuration take effect after restart logging-server daemon.

13.4.3. icsw logwatch

A very handy command to read out logfiles is **icsw logwatch**. Logwatch makes it possible to display logs for different services and daemons at once. Even if you don't know in which file the logs are written to you are able to watch it. Thats the reason why logwatch is an allround tool for logging.

As usual for log files they have a typical output format.

Table 13.1. Logwatch columns

Column number	Column name	Example
1	Date and time	Thu Apr 09 17:58:41 2015
2	Device	2_5_branch
3	System (logging daemon)	/collectd-init
4	Node	/---
5	Loglevel	warn
6	Processname,processid	MainThread.19137
7	Logmessage	sending 733 bytes to vector_socket

Without any parameter logwatch.py displays the last 400 lines of logging messages for all services writing log-files. With **icsw logwatch -n 20** you can limit output lines to the last 20.

A very useful parameter for **icsw logwatch** is **--system-filter**. This flag restricts the output to one single daemon (service) e.g

`icsw logwatch` [--system-filter *`rrd`*]
displays only log messages related to an rrd (daemon) service.

With the **-f** flag it is possible to view logs in realtime. Use

`icsw logwatch` [-f] [--machine *`MACHINE`*] [-n *`N`*] [--system-filter *`rrd`*]
to output appended data as the file grows. Try **icsw logwatch --help** to list all possible options.

13.4.4. Service communication ports

In case of malfunction it is very likely that the portnumber will be written into a logfile or apears in the web front-end. To find out which service or process causes the error we have to know which service communicates on which port. Following table shows you a little summary of common services and their communicating ports.

Table 13.2. Portnumber and services

Service	Port
md-config-server	8010
rrd-grapher	8003, 8003
logging-server	8011

Service	Port
meta-server	8012
discovery-server	8006
cluster-server	8004

Chapter 14. Frequently Asked Questions

This is a Collection of repeated Questions.

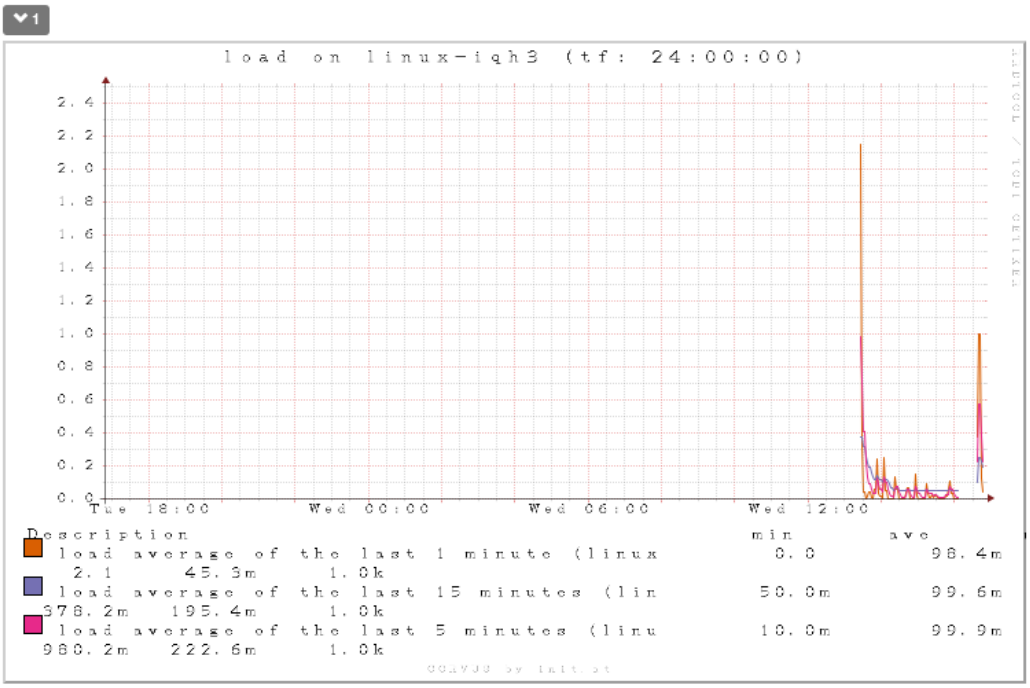
14.1. FAQ

Miscellaneous questions

14.1.1. Bad looking font in RRD Graph

14.1.1. Why are my fonts looks so ugly?

1.



ugly looking fonts due to wrong font setup

If you get something like in the picture above, you have to install **fetchmsttfonts** (OpenSUSE) and ... (debian) package.

14.1.2. Server Error (500)

14.1.2. Why i get Server Error (500)?

1.

This is a server internal error, likely the server can't find some files. Take a look into /var/log/nginx/error.log for detailed error message. Also the **lse** command could be helpful.

14.1.3. Unable to connect to the web front end

14.1.3. Why i can not connect to the web front end?

1.

For some reason the webserver nginx doesn't run. Start it manually, for example with "icsw service nginx start"

14.1.4. An error occurred

14.1.4. Why i get a message "An error occured"?
1.

Please wait a moment till database connection is active and reload the page. If you still get this message after waiting a time you have to start uwsgi-init, for example with "service uwsgi-init start"

With **top** you can display a job list. If there is something like **yuglify** in the top row than wait some time until it disapears. After that and after reloading the page the error message should dissapears.

14.1.5. Configurations seems to be ignored

14.1.5. I changed my configurations but it seems to be ignored.
1.

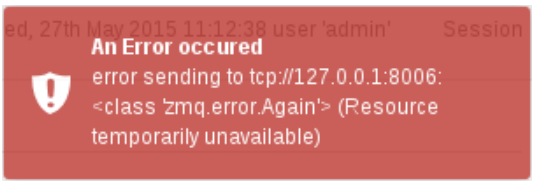
For some changes in your configuration you have to **rebuild config (cached, RC)** first. If your config is stored in cache, you have even to **rebuild config (refresh)**

14.1.6. An Error occured

14.1.6. Why does my discovery not working?
1.

Most likely the discovery-server service is not running. Make sure the discovery-server is installed and running. Run the **icsw service status** command and look for "discovery-server". If it is not running, start it either by commandline **icsw service start discovery-server** or via the webfrontend in top menu under **server information** **One server checked**

Figure 14.1. discovery-server not defined in routing



Errormessage

An other possible reason for that malfunction could be disabled **discovery server** config for your monitoring server. To enable it select your monitoring server device, navigate to the config tab and select the **discovery server** config.

After that you have to wait some time or refresh the memcached by left clicking on the server information button as described in Figure 5.9, “ Cluster server information ”

14.1.7. Slow network topology graph

14.1.7. Why is my network topology graph so slowly?
1.

Sometimes complex network topology slows down display output in firefox. This issue affects firefox up to version 31.0. Reason is likely bad javascript interpretation on firefox side. If you get bad graphic display performance, try to use another browser e.g. chromium or google chrome™.

14.1.8. Lost password

14.1.8. Ilost my password, how can i get a new one?
1.

A short guide how to reset a login password by direct access to the database via **clustershell** follows:

1. Open a terminal (e.g. xterm, Konsole, gnometerminal) on your system and start the clustershell:

   clustershell

   ```
   Python 2.7.8 (default, Jul 29 2014, 08:10:43)
   [GCC 4.8.1 20130909 [gcc-4_8-branch revision 202388]] on linux2
   Type "help", "copyright", "credits" or "license" for more information.
   (InteractiveConsole)
   >>>
   ```

2. Import relevant database content

   from initat.cluster.backbone.models import user

3. Define new variable to work with:

   my_user = user.objects.get(login="admin")

4. Set your new password.

   my_user.password="MY-new-_passW0Rd17"

   > **Important**
   >
   > ⚠Please set a secure password with more than 8 character

5. Control your new set password:

   print my_user.password

6. Save your new created password to the database:

   my_user.save()

7. Exit the clustershell

   exit()

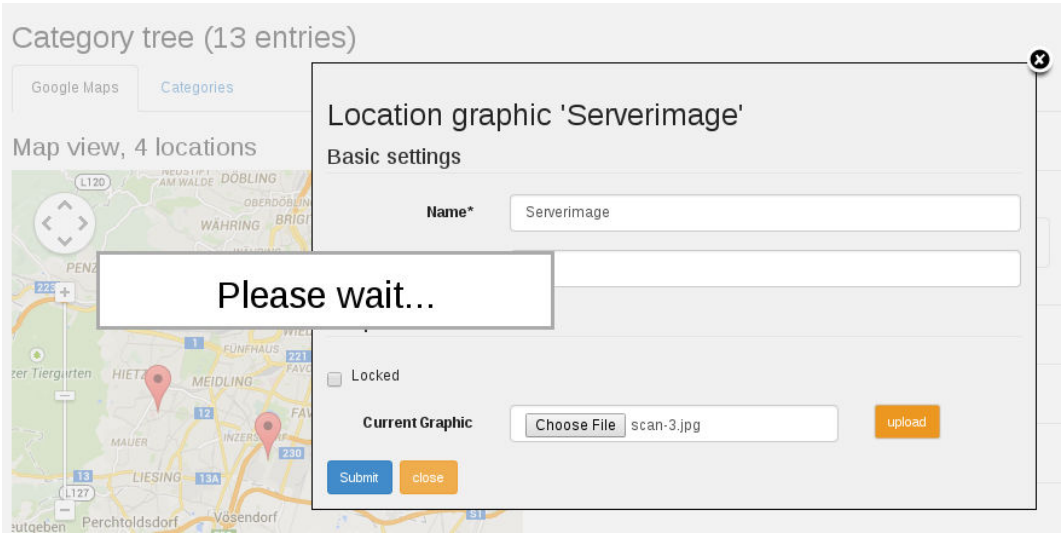
From now you are able to login with your new password.

14.1.9. "Please wait..." after add location gfx

14.1.9. What if "Please wait..." message occures for loger time?
1.

If you must wait long time while pending upload and the infolabel "Please wait..." is shown after upload image with add location gfx button, reload the page to resolv this issue.

Figure 14.2. Please wait ...



"Please wait..." message after image upload

14.1.10. Weird mouse events on virtual desktop

14.1.1 The mouse pointer position is wrong, what can i do to resolv this?
0.1.

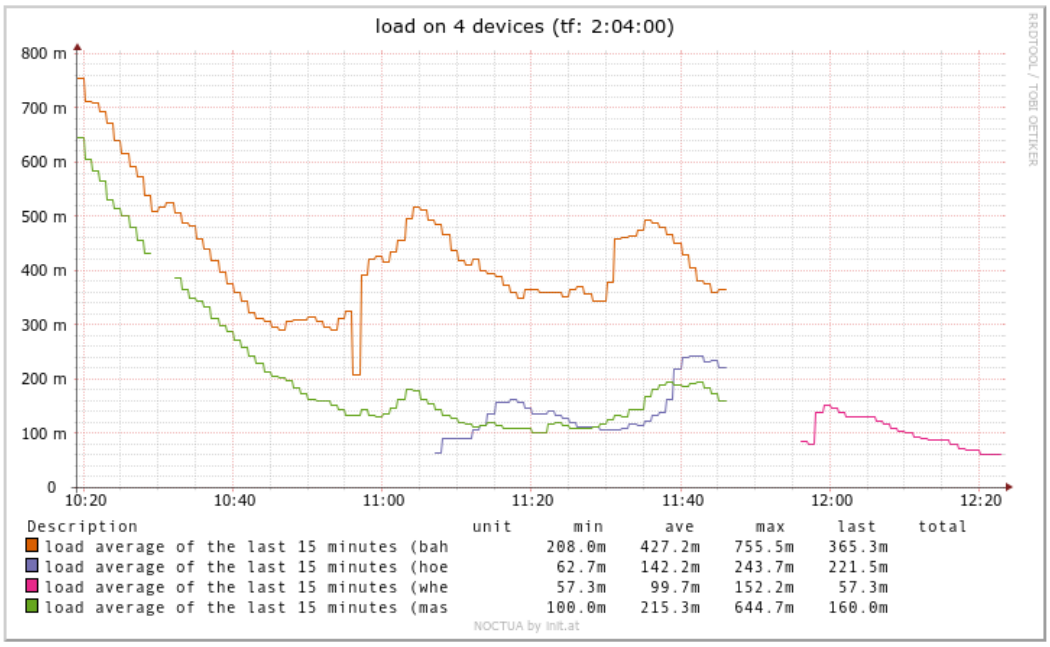
Some vnc-server tends to break correct mouse pointer handling in virtual desktop. To get back correct mouse pointer, log out of your session and back in again.

14.1.11. Asynchron graphs

14.1.1 Why are my rrd graphs ascnchron?
1.1.

If you get wrong graphs, for example 1 hour in past or 1 hour in future like the pink graph line below, make sure to set the correct timezone and times on the affected machines.

Figure 14.3. Wrong graphs due to wrong timezone



Wrong drawn graphs because of wrong timezone

> **Important**
>
> ⚠Generally, make always sure to set the correct timezone and times on every machine. This is essential for a proper running monitoring or cluster management.

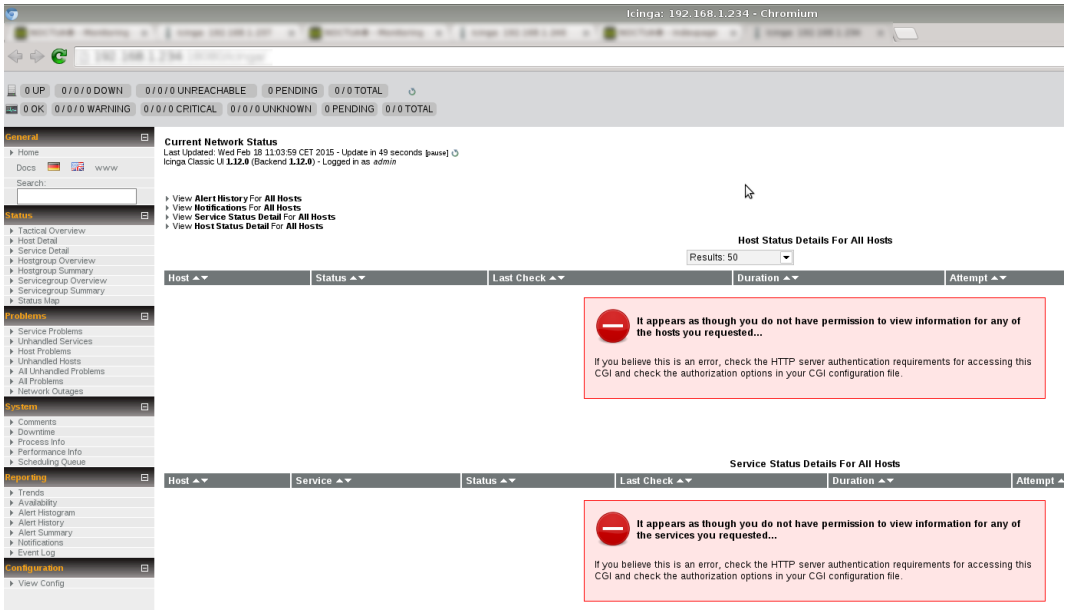
14.1.12. I have no permissions to icinga

14.1.1 2.1. How can i get the right permissions to access the icinga view?

To get the right permissions to icinga, you have to have at least one contact in **Monitoring** **Basic setup** **Contacts** and you must be loged in with this contact.

Rebuild your icinga config to apply your new contact entry. **Monitoring** **rebuild config (refresh)**

Figure 14.4. No permission to icinga



Without at least one contact you are not able to use the icinga view

14.1.13. Can not reach any network devices

14.1.1 Why i can not reach any network device?
3.1.

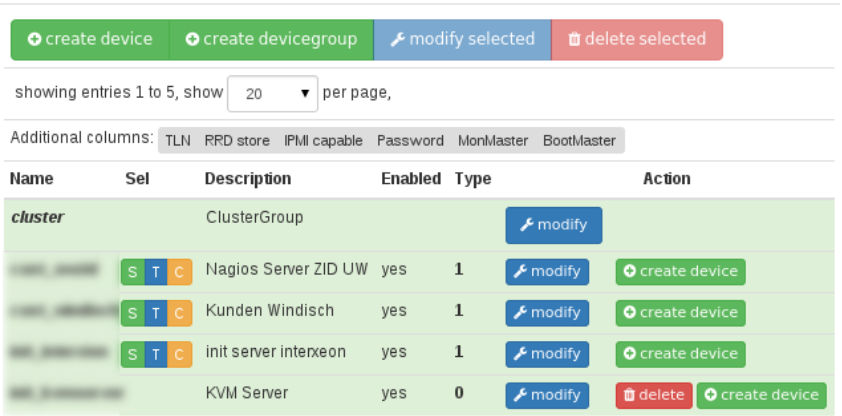
> ⚠Due to the fact that your network will be mapped into the database, you have to make sure your server finds itself in the database.
>
> To make this sure it is essential to name your **server device** equal as your **server hostname**.

14.1.14. Unable to delete group from device tree

14.1.1 How can i delete groups?
4.1.

There is no delete button visible for preselected device groups in device tree.

Figure 14.5. Deleting groups



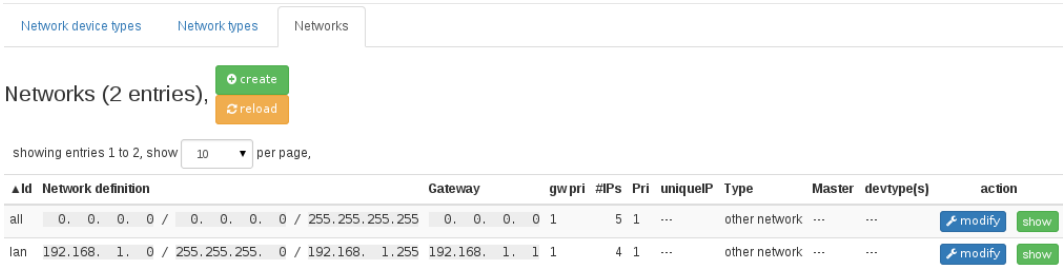
Reason for this behavior is that there are disabled devices in this group. First delete this disabled devices and finally you are also able to delete the group.

14.1.15. No "IP address" dropdown in device network

14.1.1 5.1. Why there is no "IP address" dropdown in device network??

There is no "IP address" dropdown button visible in device network until at least one network is defined in **Base** **Networks** **Networks**

Figure 14.6. Right networks setup



Two defined networks

14.1.16. Could not connect to server: Connection refused

14.1.1 6.1. I get server error on port 5432, how can i resolv the problem?

After running *icsw service status* script, a python error with Port *5432* occours. Generally, if you get error messages with the port number **5432**, the reason is likely that your postgres server (listens by default on port 5432) is down.

```
[...]
django.db.utils.OperationalError: could not connect to server: Connection refuse
Is the server running on host "localhost" (::1) and accepting
TCP/IP connections on port 5432?
```

To check if your postgres database server is running type in one of the following commands, depend on your os:

rcpostgres status

service postgresql status

systemctl status postgresql.service

Make sure to start the postgres server at boot time by enabling your operating system start scripts. Replace *status* with *start* to start the database server.

Also make sure the service starts after rebooting the *system*.

14.1.17. Internal Server Error

14.1.1 7.1. I get an internal server error in web front end, how can i resolv the problem?

After server installation and database setup, it could happen that you get an **Internal Server Error**. Try restarting your **uwsgi-init** service with this command:**rcuwsgi-init restart**.

The restart of uwsgi service results in an running **yuglify** process which generates all static files. After all static files were generated, you should get access to your web front-end.

14.1.18. License warning apears on every page

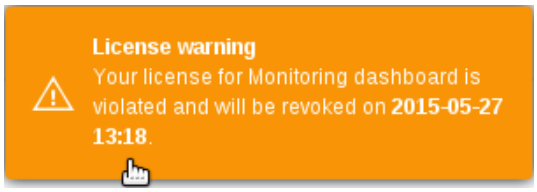
14.1.1 8.1. I get an license warning message on every single page, how can i resolve this?

A license warning message apears on the top right side, each time a new page is loaded.

This warning means that one of your licenses is either out of date and for this reason in grace time, or used devices/services/user for this license exceeds its license limitation. In both cases the license is violated and from that moment the grace period begins to run. Grace period for licenses is **2 weeks** long.

In grace period, functionality of the software is as usual but you will see this license violation warning on each new loaded page

Figure 14.7. License warning



License violation warning

You have to get a new license or expand your existing one to avoid the license violation message. Please contact us by mail `<support@init.at>` or by phone **+43-1-522 53 77** to request for a extended licenses.

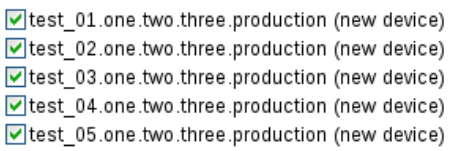
An other method to get again into a valid license limitation is to **lock** licenses for some devices.

14.1.19. Reverse domain tree node order

14.1.1 9.1. How can i reverse the domain tree node order?

Suppose you have a couple devices with following domain tree node order:

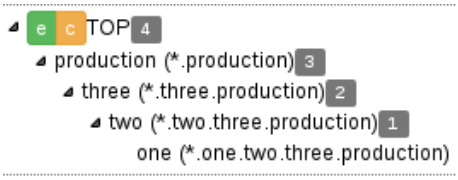
Figure 14.8. Device domain name



Original device domain name

Domain tree node structure for above device domains looks like this:

Figure 14.9. Domain name tree



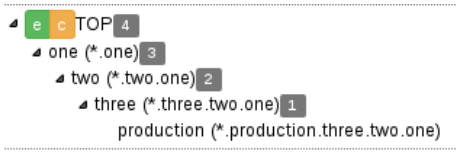
Original domain name tree

Now for example you want to revert the order of the device domains.

First step you have to do is to change the domain name tree itself. Navigate to **Base** **Domain name tree** and modify the entry called **one** and choose as parent the top level node [TLN].

Do the same with the other entries until your tree looks like this:

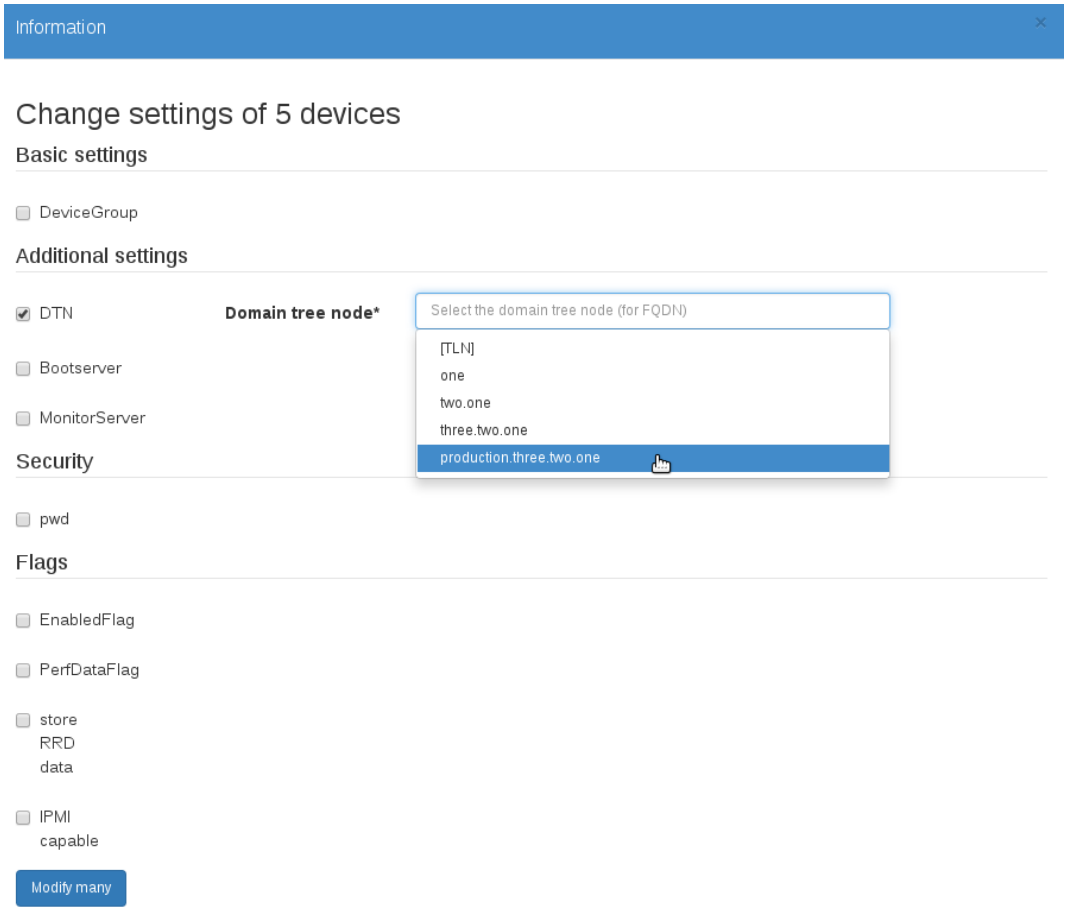
Figure 14.10. Reverted Domain name tree



Reverted domain name tree

All you have to do now is to change your domain tree node for your devices. Select your desired devices in the device tree sidebar on left side and navigate to **Base** **Device tree**. Push the **modify selected** button, select the **DTN** checkbox and choose your reverted domain tree node from the list.

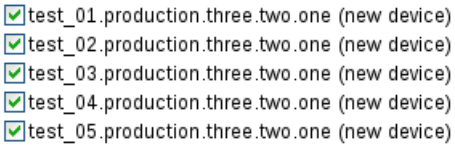
Figure 14.11. Modify devices



Modify devices to use your reverted *DTN*

The result should looks like this:

Figure 14.12. Reverted device domain names



Glossary

DHCP	Dynamic Host Configuration Protocol, The Dynamic Host Configuration Protocol (DHCP) is a standardized network protocol used on Internet Protocol (IP) networks for dynamically distributing network configuration parameters, such as IP addresses for interfaces and services. With DHCP, computers request IP addresses and networking parameters automatically from a DHCP server, reducing the need for a network administrator or a user to configure these settings manually.
Django	Django is a free and open source web application framework, written in Python
DTN	Domain Tree Node, is the tree structure of fully quallified domain names.
icinga	Industy standard software for monitoring devices.
IPMI	Intelligent Platform Managemnt Interface
NFS	Network File System, is a distributed file system protocol allowing a user on a client computer to access files over a network much like local storage is accessed.
nginx	Small and fast http server similar to apache
PXE	Preboot Execution Environment
rrd-tool	RRDtool is the OpenSource industry standard, high performance data logging and graphing system for time series data.
SGE	The "Son of Grid Engine", community project of Sun Grid Engine. [https://arc.liv.ac.uk/trac/SGE]
TFTP	Trivial File Transfer Protocol is a simple, lock-step, file transfer protocol which allows a client to get from or put a file onto a remote host. One of its primary uses is in the early stages of nodes booting from a Local Area Network. TFTP has been used for this application because it is very simple to implement.
xinetd	An open-source super-server daemon which runs on many Unix-like systems and manages Internet-based connectivity. It offers a more secure extension to or version of inetd, the Internet daemon, thus most modern Linux distributions have switched to it.