If you are concerned about breaking your installation, and you have not studied the topics involved, DO NOT DO ANY OF THE FOLLOWING! THIS IS NOT A MODIFICATION THAT HAS BEEN OFFICIALLY SANCTIONED BY THE NAS4Free TEAM!! ANY OF THESE CHANGES, !!!INCLUDING DATA LOSS!!! ARE YOUR OWN CHOSEN RISK.

Actually 2014-07-01 this page is a bit outdated, yes it's possible to install jails by hand, but actually is more secure and easy to use some jail manager like TheBrig or Finch, you can still meet very usefull info on this page, but we recommend to use:

Finch: http://dreamcat4.github.io/finch/jails-how-to/

TheBrig: https://github.com/fsbruva/thebrig/tree/working

you can see several forums post of howto install on XigmaNAS:

TheBrig: http://xigmanas.com/forums/viewtopic.php?f=79&t=3894/

Finch: http://xigmanas.com/forums/viewtopic.php?f=79&t=6260

In spite of “easier to do” that the above makes possible, jails are still completely and utterly unsupported. If you use either of the above, you should not expect any support for them.

TODO

1. Separate Advanced Topics
2. Spell-checking
3. init-script
4. jail_FOO_name=

Jails on XigmaNAS: a simple primer

What is a Jail and how do I control it?

A jail behaves like an isolated computer inside your computer. Talking jails, the outside system, the one that runs XigmaNAS, is called the host, and any system locked in a jail is, well, a jail. The host has total control and can see every jail's files, processes and other resources. From a jail, only resources assigned to this particular jail are visible. Jails may have jails, too. A portion of the host's file system may be made available to a jail.

Files that belong to the jail will show up somewhere in the file system of your host. This includes the OS' files as well as your hand crafted software. Part of the magic of jails is locking processes to this portion of the host's file system. Jails do a bit more than chroot, though.

A jail may run a different version of FreeBSD than the host. Only the host should run the most recent

OS version. A jail with a more recent OS version than the host will probably not run at all. This is a problem with XigmaNAS as host because updating a jail may have to be postponed until the next XigmaNAS release.

Jails are deeply integrated into FreeBSD, whether you use them or not. In fact, many standard commands on FreeBSD support jails in their options. Jails are developing fast. When you read something on the net about what can and what cannot be done with a jail, the information may be simply outdated.

There are two ways of controlling jails on FreeBSD:

1. the builtin commands
2. everything else

If you chose the builtin commands, there is a ton of things you can do, but you will have to stick to the CLI. There are some things external jail managers do that the builtin system cannot do. If you need those, or if you want convenience, go for a jail manager. The most modern and best maintained jail manager (as of this writing) may be iocage. Keep in mind that installation of external software may require a XigmaNAS “full installation”, whereas the recommended XigmaNAS installation type is “embedded”, in which the distribution runs unchanged from an isolated medium like a USB stick. If you value your host's primary designation, you may want to stick to the embedded flavour.

This is where XigmaNAS compatible jail managers like theBrig or bastille come into play. The may be installed as additional software on an embedded install but will survive version upgrades. Also, these managers will permit you to do some basic jail maintenance, such as creating, deleting, starting and stopping jails. To some of us, this is all that will be required (most of the time). One big advantage of bastille is that it comes with a CLI interface that permits some of the fancier stuff, too. Bastille is very nearly self-contained, even if it is not absolutely zero dependency as claimed. TheBrig can do more tricks, but it's not maintained any more.

XigmaNas specialties

When running a jail on XigmaNAS, your host will be primarily busy doing XigmaNAS stuff. A jail is subordinate to the host's business and can use only the resources the host spares. However, there may be good reasons for running jails on XigmaNAS:

- you need additional software running all the time and have no other machine but your XigmaNAS
- you require services besides XigmaNAS' capabilities and separate hardware would be overkill
- you want to isolate some software from the sensitive NAS file system

Personally, I needed to host some lightweight but important business logic and I wanted to move that logic around between machines, like for development or desaster recovery. I also wanted different containers for development and productive systems, possibly with the option for archiving ancient system versions.

My personal advice is to go for a XigmaNAS jail manager to get going with jails. Make sure you pick a manager that will permit FreeBSD upgrades just to the level of your host. If it does not, you will eventually be required to move all your jailed applications into a new jail. This may be a lot of work and sticking to the old jail may be no option if OS support ends.

Bastille jail manager

(to be continued)

HowTo Setup a jail on top of XigmaNAS

System information & Introduction

Hostname	xigmanas.local
IPv4	192.168.0.4
Version	9.0.0.1 Sandstorm (rev. 50)
Built on	Sat Apr 7 15:36:46 CEST 2012
OS Version	FreeBSD 9.0-RELEASE (revision 199506)
Platform	x64-embedded on Intel(R) Pentium(R) 4 CPU 2.80G

~~**Here you can find a list of commands I used to setup the prototype jail on my x64 box: Command Summary UPDATE neccessary**~~

Install/Run XigmaNAS

(if you haven't done so already)

Install: See Installation and Configuration Overview

LiveCD: See SUG Section 2.2-Using XigmaNAS with the CDROM and a removable disk (LiveCD mode).

This method should work with just about any version of XigmaNAS. Although it is probably best to use the Full platform, this method should work with all three.

Setup your system

IP, DNS, Gateway, Harddrives, SSH, etc.. Make sure everything is working well. The majority of these commands are done via CLI with SSH so make sure that is working too.

add the jail_enable = yes in system webgui

Create Folder Tree and Mount Point

Remember that all references to /mnt/data must be changed in your NAS to mnt/yourmountpoint

This will create the initial folder structure and point /mnt/data/jail to /jail for ease of jail creation.

```
# mkdir /jail
```

```
# mkdir /mnt/data/jail
# mkdir /mnt/data/jail/{work,proto,conf}
# mount_nullfs /mnt/data/jail /jail
```

/jail/work	for downloads,temporary files.
/jail/proto	the prototype jail, later more.
/jail/conf	the configuration and runtime files.

You can place your jails wherever you want. In this HowTo we use mount_nullfs to access our jail folder on the storage devices under a new namespace (/jail).

Download/Extract FreeBSD Base System

Download

You have to download the FreeBSD base system to get all neccessary binaries, configfiles and scripts.

```
# cd /jail/work
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/releases/`uname -m`/`uname -
m`/`uname -r | cut -d- -f1-2`/base.txz
# fetch ftp://ftp.freebsd.org/pub/FreeBSD/releases/`uname -m`/`uname -
m`/`uname -r | cut -d- -f1-2`/lib32.txz
```

This looks complicated but it will make sure you download the correct architecture and release. Also, you do not need to download the 32 bit libraries on a 32 bit system.

Extract

...the FreeBSD base system to your /jail/proto directory (256M base + 56 M lib32)

```
# tar xvf /jail/work/base.txz -C /jail/proto/
# tar xvf /jail/work/lib32.txz -C /jail/proto/
```

Install jail binaries

Create the file structure for your jail runtime files.

```
# cd /jail
# mkdir -p conf/root/{etc/rc.d/,usr/bin,usr/sbin}
```

Copy the necessary rc-script and binaries to conf/

```
# cp proto/etc/rc.d/jail conf/root/etc/rc.d/
# cp proto/usr/sbin/{jail,jexec,jls} conf/root/usr/sbin/
```

```
# cp proto/usr/bin/mktemp conf/root/usr/bin/
```

Pit Stop

To make sure we are on track, let's make a Pit Stop and review what we have done so far. Please follow this checklist and short summary to avoid later problems:

- Jail folder tree created and mount point to /jail set.
- Downloaded and extracted FreeBSD base for the correct architecture.
- Create the config folder with the necessary files and sub-folders.

```
/jail/conf/root/
        |-- etc
        |   `-- rc.d
        |       `-- jail
        `-- usr
            |-- bin
            |   `-- mktemp
            `-- sbin
                |-- jail
                |-- jexec
                `-- jls
```

> The output above was created with tree - list contents of directories in a tree-like format

Configuring the prototype jail

Setting a nameserver for proto

Copy /etc/resolve/conf from the XigmaNAS host:

```
# cp /etc/resolv.conf /jail/proto/etc/
```

Setting timezone

Copy the necessary timezone file from: /jail/proto/usr/share/zoneinfo/ To: /jail/proto/etc/localtime

Example: (Obviously choosing your correct timezone)

```
# cp /jail/proto/usr/share/zoneinfo/Europe/Berlin /jail/proto/etc/localtime
```

Create fstab file

If we want to mount some local or remote drives in our jail we have to add them to:

/jail/conf/fstab.<jailname>

In our case we create an empty file for proto:

```
# touch /jail/conf/fstab.proto
```

PD: in case you need to mount a folder inside your jail you need to create the folder on /mnt inside the jail and edit fstab.prot to desired value. Eg:

```
mkdir /jail/proto/mnt/Video
```

```
nano /jail/conf/fstab.proto
```

and this can be his content (remember to edit to yours pool):

```
/mnt/data/Video /jail/proto/mnt/Video nullfs ro 0 0
```

Create rc.conf.local:

```
# touch conf/rc.conf.local
# nano conf/rc.conf.local
```

Add These Lines:

```
### EXAMPLE - MODIFY TO FIT YOUR NEEDS ###
jail_enable="YES" # enable jails YES|NO
jail_list="proto" # name of the jails to start "proto www..."
jail_proto_rootdir="/jail/proto"      # path to our jail
jail_proto_hostname="proto.domain.local"  # hostname
jail_proto_ip="192.168.1.201"     # ip of the jail
jail_proto_interface="em0"        # Network Interface to use, replace on
your NAS interface name
jail_proto_devfs_enable="YES"     # use devfs
jail_proto_mount_enable="YES"     # mount YES|NO
jail_proto_fstab="/jail/conf/fstab.proto" # File with Filesystems to mount
```

Change the settings to fit your needs and save. Especially *jail_proto_interface* and *jail_proto_ip*.

Finish Setup and Start the Jail

To mount /jail and start the jails we need a very simple script.

```
  # nano /jail/conf/jail_start
```

Add these lines:

```
#!/bin/tcsh -x
#mounting to /jail
mkdir /jail
mount_nullfs /mnt/data/jail /jail
```

```
# copy jail binaries to /usr, not needed if N4F is 454 or up
# because Daoyama include needed files, uncoment if you use low .454 version
# cp -r /jail/conf/root/ /
# link config files to /etc
ln -s /jail/conf/rc.conf.local /etc
#start all jails
/etc/rc.d/jail start
```

Save and make the script executable:

```
# chmod 755 /jail/conf/jail_start
```

Open the WebGUI and add a command script.

System|Advanced|Command Scripts:

Command:	/mnt/data/jail/conf/jail_start
Type:	PostInit

Save and apply, now reboot your server.

After a successful reboot you can check your new jail: (via CLI/SSH)

```
# jls
    JID  IP Address      Hostname                      Path
    1    192.168.1.201   proto.domain.local            /jail/proto
```

If jls command does not work type the command below then try again (we're working on this issue).

```
#rehash
```

Then enter it:

```
# jexec 1 csh   # enter jail 1 in our case proto
```

Where jexec **1**, reflects the JID from the output above.

Deleting a Jail

Once you are ready to delete a jail, there are a few steps to ensure success. First is to stop the running jail in question.

```
nas4free# /etc/rc.d/jail stop proto
```

Additionally, the proto.fstab file needs to be removed, and all mention of the jail needs to be removed from the rc.conf.local file.


Don't forget to remove the jail name from the jail_list = “proto” line!

The last step is to forcibly change the flags within the directory:

```
nas4free# chflags -R noschg /mnt/data/jail/proto
nas4free# rm -rf /mnt/data/jail/proto
```

If it is the last jail to be deleted, then the entirety of /mnt/data/jail can also be deleted.


Most users can stop here!! The next set of steps is for advanced users!!!

Advanced Topics

Strip down proto jail

… or how to build a new world

Our current jail (proto) includes many applications from the FreeBSD base system. Many of them we don't need inside a jail.

A few examples are:

- Bootloader
- Firewall - should be handled by the host.
- USB support - ditto
- Sendmail - no need to run it in every jail.
- Wifi Support.
- Rescue System.
- etc.

Just to name only a few.

After entering the jail everything we do is not really NAS4Free related anymore. You can read any FreeBSD documentation about jails and how to setup applications inside a jail, but I'd still like to show you my method for stripping down my jail (proto).

A few words about file systems.

All I describe is independent from the underlying file system. In practice it is fine to work with jails on top of a zfs file system, the snapshot and clone features are very handy. But I assume we are running only a few jails on top of NAS4Free so I don't care about easy, read-only parts in jails. Here I will set-up every jail independently. The main reason is that you can download a pre-built jail and put it on your NAS4Free with less then 5 commands.

* Download * Unzip * Edit a configfile (set hostname, IP ...)

NOTE: I will use - xigmanas# - for commands I'm running on the XigmaNAS host. And - proto# - for commands I'm run inside the proto jail.

Assuming you are connected to your XigmaNAS server via SSH or Terminal and have root permission.

```
xigmanas# jls
JID  IP Address      Hostname                      Path
1    192.168.1.201   proto.domain.local            /jail/proto
xigmanas# jexec 1 csh
```

The Plan

To stripdown and customize our prototype jail we need the FreeBSD source code for our release. In this case for FreeBSD 9.0. We recompile our complete proto jail from the FreeBSD source code. Sounds very complicated but it isn't, also it looks very “urgent” if a friend visits you and sees all this code running down the screen 😃.

NOTE: The default editor for FreeBSD systems are:

- ee - EasyEditor
- vi - No Comment

If you'd like to use nano you have to install it first. Execute the following commands inside your jail to download and install the nano editor.

```
proto# pkg install nano
proto# rehash
```

In my examples I will use 'ee' to show that I opened a editor.

Getting FreeBSD sources

We will use svn to fetch the sourcecode from the FreeBSD server. In order accomplish this, carry out the following commands INSIDE THE JAIL!

```
proto# pkg_add -r subversion
```

The previous instructions for this wiki uses csup, which obtained the most recent STABLE codebase. The command to download this same portion of the source code is:

```
proto# svn co svn://svn.freebsd.org/base/stable/9 /usr/src
```

This tells the svn utility to CheckOut the most recent stable codebase, and storing the source in the jail's /usr/src.

The download will take some time. After it is done, you'll find the source code in /usr/src.

/etc/src.conf

In this file we enter everything we do not need in our jail.

The following command will create a basic src.conf, just copy and paste it to your command line:

```
proto# cat << ! > /etc/src.conf
WITHOUT_ACPI=                   true
WITHOUT_ATM=                    true
WITHOUT_BIND=                   true
WITHOUT_BLUETOOTH=              true
WITHOUT_BOOT=                   true
WITHOUT_CALENDAR=               true
WITHOUT_DICT=                   true
WITHOUT_EXAMPLES=               true
WITHOUT_FORTH=                  true
WITHOUT_GAMES=                  true
WITHOUT_GCOV=                   true
WITHOUT_GPIB=                   true
WITHOUT_GROFF=                  true
WITHOUT_HTML=                   true
WITHOUT_I4B=                    true
WITHOUT_IPFILTER=               true
WITHOUT_IPX=                    true
WITHOUT_KERBEROS=               true
WITHOUT_LPR=                    true
WITHOUT_MAILWRAPPER=            true
WITHOUT_MAN=                    true
WITHOUT_NCP=                    true
WITHOUT_NETCAT=                 true
WITHOUT_NIS=                    true
WITHOUT_NLS=                    true
WITHOUT_NLS_CATALOGS=           true
WITHOUT_NS_CACHING=             true
WITHOUT_OBJC=                   true
WITHOUT_PF=                     true
WITHOUT_RCMDS=                  true
WITHOUT_RCS=                    true
WITHOUT_RESCUE=                 true
WITHOUT_SENDMAIL=               true
WITHOUT_SHAREDOCS=              true
WITHOUT_USB=                    true
WITHOUT_WPA_SUPPLICANT_EAPOL=   true
WITHOUT_ZFS=                    true
!
```

And ...

```
proto# ee /etc/src.conf
```

… to take a look into the file.

You can find a more or less complete list of possible /etc/src.conf entries in WITHOUT_XXX.

buildworld

The process of recompiling the complete system from source is called buildworld. Depending on your computer and the configuration in your src.conf this can take from 20 minutes to 1 hour.

For security some things are disallowed inside a jail. For example to “change the flags” of a file. We have to set a new sysctrl value via the WebGUI on the NAS4Free host to allow this, we need it during our buildworld.

Open the WebGUI and go to tab:

System|Advanced|sysctl.conf

And set following MIB:

```
Name: security.jail.chflags_allowed
Value: 1
Comment: Allow chflags inside jails
```

Following this configuration change, you need to reboot to have it take effect. Note that the next instruction takes place within the proto jail.

Start buildworld:

```
proto# cd /usr/src
proto# make -j4 buildworld
```

The option -j sets the number of parallel processes during a build. I usually use one per core.

After the buildworld is complete we install our new world.

```
proto# make installworld
```

Well, I wish it would be so easy to rebuild the real world.

Now we can use “installworld” to setup other jail's. More in the next chapter.

Creating A New Jail

Now we have our proto jail which is ready to use. It is possible to create a copy of this folder and after adding a new entry to /jail/conf/rc.conf.local, it would be ready to fire up, but we will do it in FreeBSD style, which is to say, we use our previous buildworld to create a new jail.

In this section of the HowTo, we create a new jail called 'www'. It will later contain an Apache webserver + PHP + MySQL database. This is a good base to create some additional services like a server for a Content Management System like Joomla or a backup server with BackupPC. The good thing about jails is you can create as many as you want. Virtual machines like vmware need plenty of CPU power just to run, but our jails have nearly 0 overhead.

It is also possible to put each service in its own jail (apache jail, mysql jail, etc.) but here I like to use jails for easily adding services to XigmaNAS, not to play ISP for thousands of customers. But I'm talking too much.

Lets set-up the jail, and I assume you'll get some ideas what to do with it.

We start on the XigmaNAS host. Create a new folder for the new jail:

```
xigmanas# mkdir /jail/www
```

Mount the www folder to your running proto jail:

```
xigmanas# mount_nullfs /jail/www /jail/proto/mnt/
```

Enter the proto jail:

```
xigmanas# jexec 1 csh
proto#
```

Installworld to the www jail mounted on /mnt:

```
proto# cd /usr/src
proto# make installworld DESTDIR=/mnt
```

It should be clear what DESTDIR= means:

```
proto# make distribution DESTDIR=/mnt
```

“**make installworld**” will only install the necessary binaries for a functioning system, “**make distribution**” will install the default configuration files as well.

```
proto# exit
```

Now we leave our proto jail and prepare the www jail for a first start:

```
xigmanas# umount /jail/www
```

Don't forget to set the nameserver and timezone, see Configuring the prototype jail.

Last but not least, we modify /jail/conf/rc.conf.local and create fstab.www:

```
xigmanas# touch /jail/conf/fstab.www
xigmanas# nano /jail/conf/rc.conf.local
```

And add something like:

```
jail_www_rootdir="/jail/www"
jail_www_hostname="www.domain.local"
jail_www_ip="192.168.1.202"
jail_www_interface="em0"
jail_www_devfs_enable="YES"
jail_www_mount_enable="YES"
jail_www_fstab="/jail/conf/fstab.www"
```

Dont forget to modify:

```
jail_list="proto www"
```

If you want to start the www jail on boot.

Our www jail should now be ready to start.

```
xigmanas# /etc/rc.d/jail start www
Configuring jails:.
Starting jails: www.domain.local.
nas4free# jls
JID  IP Address      Hostname                      Path
   4  192.168.1.202   www.domain.local              /jail/www
   3  192.168.1.201   proto.domain.local            /jail/proto
```

Ok, if the www jail is up and running it is time to add some stuff to make it useful, but first a few words about the FreeBSD ports and packages.

FreeBSD ports and packages

Here I refer to the FreeBSD Handbook, there is no better explanation.

Chapter 4 Installing Applications: Packages and Ports

Installing ports and packages are a part of the basic knowledge so read it carefully.

Getting ports

It's time to hurry up a little. There is lots of documentation around on how to setup the www jail. Here I'll use 'portsnap' to get the FreeBSD portstree. Also currently we do not share the portstree with other jails.

Lets dowload the portstree to start over.

```
xigmanas# jexec 4 csh
www#
```

```
www# portsnap fetch extract
```