SKYLABLE

S^X Manual

CONTENTS

CHAPTER 1

INTRODUCTION

Welcome to Skylable S^{X}, a complete private cloud framework. With Skylable S^{X} you can create flexible, reliable and secure storage solutions, which can be accessed from all popular platforms.

1.1 S^{X} FEATURES

This software has been designed and built with usability in mind. Some of the great features of Skylable S^{X} include:

- **Fast and lightweight protocol**
  Never transfer the same data twice. The S^{X} protocol transfers only the differences between the local copy of a file and the data already stored in the cluster. Additionally, it transfers data to/from all SX nodes in parallel to maximize speed.

- **Replication**
  Choose how many times you want your data to be replicated. You can set different levels of replica for different data and find the perfect balance between reliability and efficiency.

- **Rack Awareness**
  Ensure data gets replicated across different racks or regions by grouping nodes into zones.

- **Deduplication**
  If you upload 10 copies of the same data, that data takes the same space as 1 copy. If you upload 10 files, which only differ in a few bytes, just the differences between the files will take up additional space and the rest will be deduplicated.

- **Encryption**
  Client-side encryption with AES256, HTTPS communication, and all the security best practices to keep your data safe.

- **Revisions and undelete**
  S^{X} can optionally keep multiple revisions of your files and allow you to go back in time to previous versions of your files. You can also restore a file that has been accidentally deleted.

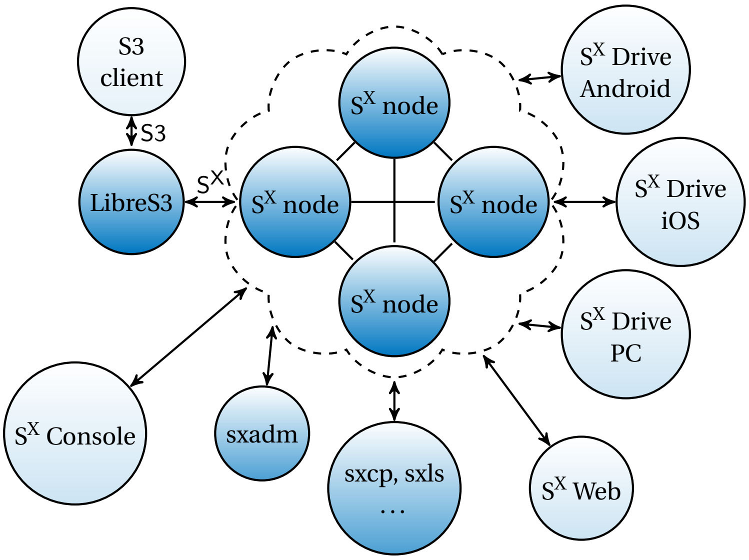


Figure 1.1: Skylable cloud ecosystem

- **S3 Support**
  Need a drop-in replacement for S3? Install S^X together with LibreS3 and change a single setting in your S3 clients and tools: that's all you need to do to switch to Skylable.
- **Mobile and Desktop clients**
  Keep your files synchronized across all your devices with SXDrive: available for Windows/MacOSX/Linux on desktop and iOS/Android on mobile.

1.2 SKYLABLE ECOSYSTEM

This manual covers the open-source Skylable S^X software for UNIX platforms. S^X is used to create data clusters and forms the base for the Skylable platform, which consists of multiple components as shown on figure 1.1 With Skylable S^X one can build a cloud consisting of many nodes, which can be accessed in different ways and from multiple platforms. The client applications include command-line tools, which are part of the Skylable S^X itself, as well as desktop and mobile apps.

The command line tools shipped with Skylable S^X provide a typical UNIX experience and together with external tools and scripts can be used to automate various processes, such as backups.

S^X Drive for Linux, Windows and OS X keeps remote data synchronized with a local directory. The synchronization works both ways, and the latest data is always available on the local machine and in the cloud.

S^X Drive for Android and iOS provides an instant access to the cloud from mobile devices. One can upload or download any documents or photos and keep favourite files automatically updated.

S^X Web provides a web interface to the cloud. Users can access all their data right from web browsers. S^X Web additionally provides an easy way to share files with other people.

S^X Console provides a web-based management console for the cluster. The cluster administrators can perform all regular operations right from a browser. S^X Console makes it easy to manage users, volumes, monitor health and status of the cluster.

Finally, with **LibreS3**, the Skylable cloud becomes available to clients compatible with the S3 protocol. LibreS3 implements a large subset of the S3 API and translates it to the S^X protocol. It makes possible to use existing solutions such as s3cmd or DragonDisk with Skylable S^X.

CHAPTER 2

INSTALLATION / UPGRADE

Skylable S^X is tested on all popular UNIX platforms, including Linux, FreeBSD, and OS X. We try to support as many platforms as possible, if you have troubles installing, compiling or running our software on your platform please let us know.

2.1 MINIMUM REQUIREMENTS

The default setup described in this manual requires 2GB of RAM available for each node. S^X can also be installed on machines with lower resources, such as embedded ARM devices, but that requires advanced configuration not covered by the manual.

2.2 BINARY PACKAGES

The binary packages are available for all popular Linux distributions, and this is the easiest and recommended way to install Skylable S^X. Please visit
`http://www.skylable.com/download/sx`
for the up-to-date list of supported distributions and installation instructions.

2.3 SOURCE CODE

In order to compile S^X from source, you will need the following packages to be installed together with their development versions:

- OpenSSL/NSS
- libcurl $\geq$ 7.34.0 (otherwise the embedded one will be used)
- zlib
- FUSE $\geq$ 2.7.0 (otherwise sxfs will not be compiled)

For example, on Debian run:

```
# apt-get install libssl-dev libcurl4-openssl-dev libz-dev fuse libfuse-dev
```

COMPILATION

The software is based on autoconf, and you can just perform the standard installation steps. The following commands install all the software in `/usr/local`:

```
$ ./configure && make
# make install
```

The rest of the manual assumes that S^X was installed from a binary package, so some paths may be different.

2.4 Upgrade existing cluster

To take advantage of new features and improvements, it's recommended to keep the cluster software up to date. The upgrade procedure has been simplified and automated as much as possible to allow a smooth update of a live cluster.

Upgrading a single node

It is recommended to upgrade one node at a time. In case of a problem, the other nodes will stay unaffected and will be able to serve data to the clients. First install the latest version of Skylable S^X in the same way as the previous deployment. Then run the following command:

```
# sxsetup --upgrade
Updating sxhttpd.conf (vts)
Upgrading local node...
[sx_storage_upgrade]: Performing integrity check on /var/lib/sxserver/storage
[sx_storage_upgrade]: Integrity check completed in 0s
[sx_storage_upgrade]: Upgrading local databases, this may take a while...
[sx_storage_upgrade]: Committing changes
[sx_storage_upgrade]: Schema upgrade completed in 0s
[sx_storage_upgrade]: Successfully upgraded all DBs
[sx_storage_upgrade]: Storage closed in 1s
[upgrade_node]: Storage is up to date
Starting SX.fcgi
Starting sxhttpd
SX node started successfully
Moving remote sxsetup.conf into cluster settings...

More nodes in the cluster require upgrading.
```

The above is the expected output when not all of the nodes have been updated yet. After upgrading the entire cluster the expected output is:

```
# sxsetup --upgrade
Local node is up-to-date.
Versions:
        192.168.1.101: 2.0 (2.0)
        192.168.1.102: 2.0 (2.0)
        192.168.1.103: 2.0 (2.0)
Cluster already fully upgraded

All components of the cluster are up-to-date!
```

IMPORTANT: Due to substantial changes and improvements, all nodes must be upgraded to 2.0 otherwise some operations will only work in read-only mode.

Chapter 3

Cluster deployment

3.1 Requirements

S^X by default operates on the port 443 or 80, which needs to be available on a given IP address[1]. You can build just a single-node S^X cluster, however for data safety reasons it is recommended to create at least two nodes and use replica higher than 1. You can add more nodes to the cluster at any time.

Firewall rules

Some systems, such as Fedora, block most services by default. In order to allow SX over https on Fedora run the following commands:

```
# firewall-cmd --permanent --add-service=https
# firewall-cmd --reload
```

Clock synchronization

Please make sure that clocks are properly synchronized on all nodes by running NTP.

Disk space

S^X doesn't pre-allocate the disk space — you will need to monitor the nodes to make sure they have enough physical space available for S^X operations.

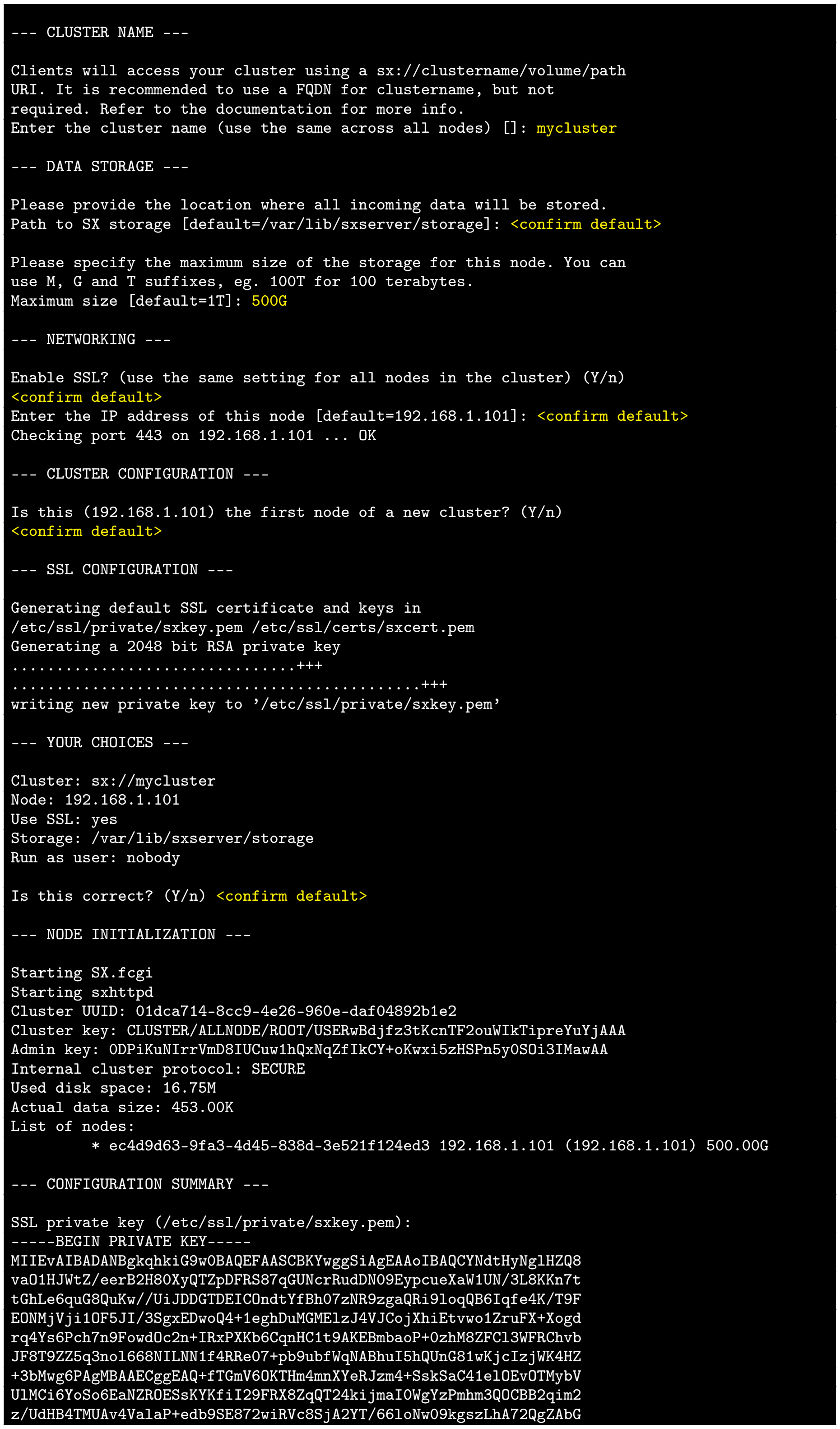
3.2 Creating the first node

Setting up the first node initializes the cluster and makes S^X ready to use. The `sxsetup` tool presented below performs an automated configuration of the S^X server, which includes creating a local data storage, SSL certificate, and default admin account. You will only need to answer a few basic questions!

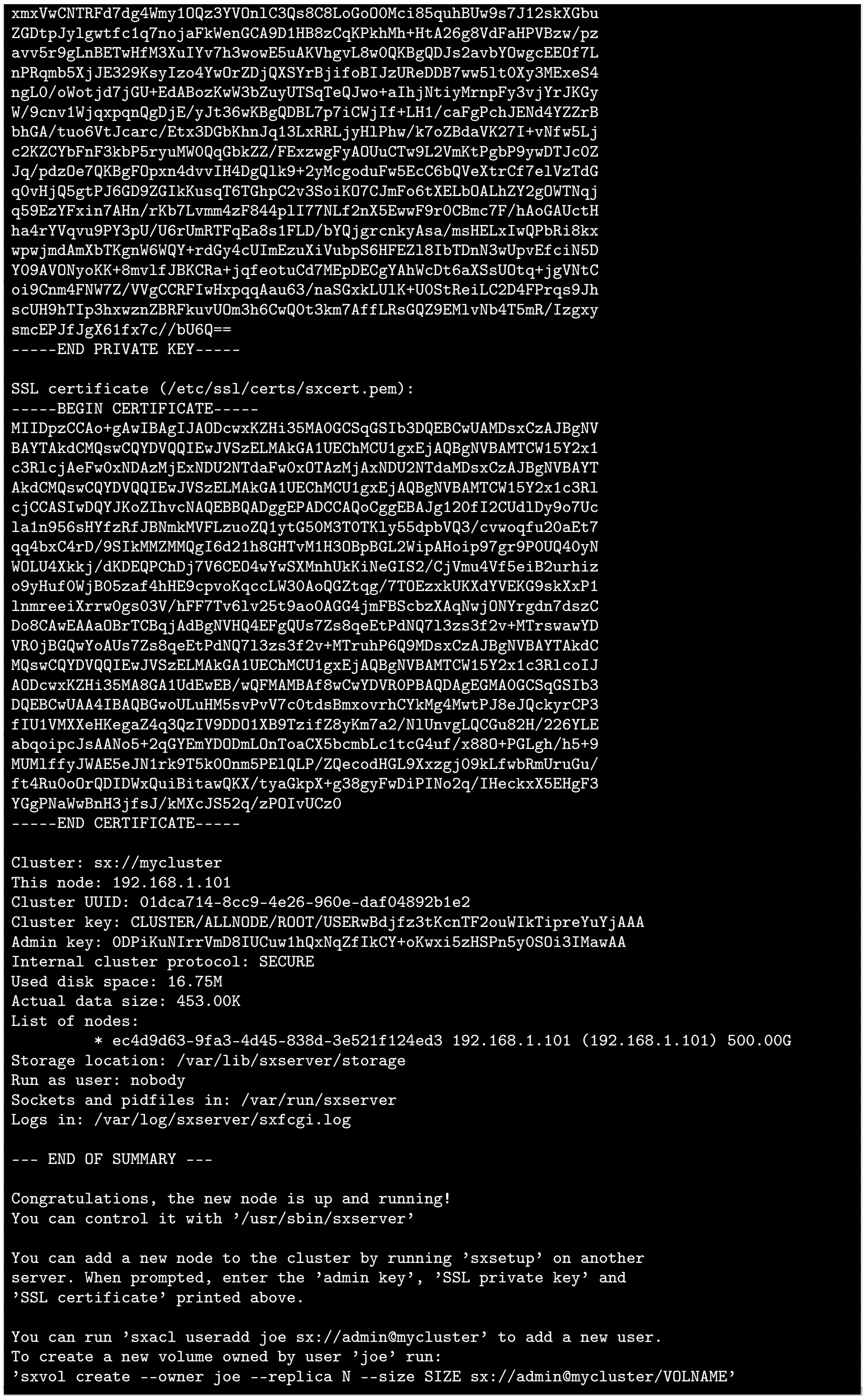
In the example we assume the IP address of the first node is **192.168.1.101**, the name of the cluster is **mycluster**, and S^X was installed from a binary package. In many some cases (eg. the path to S^X storage) we assume the default values, but you may want to customize them.

```
# sxsetup
--- SKYLABLE SX CONFIGURATION SCRIPT ---

The script will help you to create or extend a Skylable SX data
cluster.
```

[1] You can choose a custom port when running `sxsetup` in advanced mode.



```
--- CLUSTER NAME ---

Clients will access your cluster using a sx://clustername/volume/path
URI. It is recommended to use a FQDN for clustername, but not
required. Refer to the documentation for more info.
Enter the cluster name (use the same across all nodes) []: mycluster

--- DATA STORAGE ---

Please provide the location where all incoming data will be stored.
Path to SX storage [default=/var/lib/sxserver/storage]: <confirm default>

Please specify the maximum size of the storage for this node. You can
use M, G and T suffixes, eg. 100T for 100 terabytes.
Maximum size [default=1T]: 500G

--- NETWORKING ---

Enable SSL? (use the same setting for all nodes in the cluster) (Y/n)
<confirm default>
Enter the IP address of this node [default=192.168.1.101]: <confirm default>
Checking port 443 on 192.168.1.101 ... OK

--- CLUSTER CONFIGURATION ---

Is this (192.168.1.101) the first node of a new cluster? (Y/n)
<confirm default>

--- SSL CONFIGURATION ---

Generating default SSL certificate and keys in
/etc/ssl/private/sxkey.pem /etc/ssl/certs/sxcert.pem
Generating a 2048 bit RSA private key
................................+++
..............................................+++
writing new private key to '/etc/ssl/private/sxkey.pem'

--- YOUR CHOICES ---

Cluster: sx://mycluster
Node: 192.168.1.101
Use SSL: yes
Storage: /var/lib/sxserver/storage
Run as user: nobody

Is this correct? (Y/n) <confirm default>

--- NODE INITIALIZATION ---

Starting SX.fcgi
Starting sxhttpd
Cluster UUID: 01dca714-8cc9-4e26-960e-daf04892b1e2
Cluster key: CLUSTER/ALLNODE/ROOT/USERwBdjfz3tKcnTF2ouWIkTipreYuYjAAA
Admin key: 0DPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5y0SOi3IMawAA
Internal cluster protocol: SECURE
Used disk space: 16.75M
Actual data size: 453.00K
List of nodes:
         * ec4d9d63-9fa3-4d45-838d-3e521f124ed3 192.168.1.101 (192.168.1.101) 500.00G

--- CONFIGURATION SUMMARY ---

SSL private key (/etc/ssl/private/sxkey.pem):
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCYNdtHyNglHZQ8
vaO1HJWtZ/eerB2H8OXyQTZpDFRS87qGUNcrRudDN09EypcueXaW1UN/3L8KKn7t
tGhLe6quG8QuKw//UiJDDGTDEICOndtYfBh07zNR9zgaQRi9loqQB6Iqfe4K/T9F
EONMjVji1OF5JI/3SgxEDwoQ4+1eghDuMGMElzJ4VJCojXhiEtvwo1ZruFX+Xogd
rq4Ys6Pch7n9FowdOc2n+IRxPXKb6CqnHC1t9AKEBmbaoP+0zhM8ZFCl3WFRChvb
JF8T9ZZ5q3nol668NILNN1f4RRe07+pb9ubfWqNABhuI5hQUnG81wKjcIzjWK4HZ
+3bMwg6PAgMBAAECggEAQ+fTGmV6OKTHm4mnXYeRJzm4+SskSaC41elOEvOTMybV
UlMCi6YoSo6EaNZROESsKYKfiI29FRX8ZqQT24kijmaI0WgYzPmhm3QOCBB2qim2
z/UdHB4TMUAv4ValaP+edb9SE872wiRVc8SjA2YT/66loNw09kgszLhA72QgZAbG
```



```
xmxVwCNTRFd7dg4Wmy1OQz3YVOnlC3Qs8C8LoGoOOMci85quhBUw9s7J12skXGbu
ZGDtpJylgwtfc1q7nojaFkWenGCA9D1HB8zCqKPkhMh+HtA26g8VdFaHPVBzw/pz
avv5r9gLnBETwHfM3XuIYv7h3wowE5uAKVhgvL8w0QKBgQDJs2avbYOwgcEEOf7L
nPRqmb5XjJE329KsyIzo4YwOrZDjQXSYrBjifoBIJzUReDDB7ww5lt0Xy3MExeS4
ngL0/oWotjd7jGU+EdABozKwW3bZuyUTSqTeQJwo+aIhjNtiyMrnpFy3vjYrJKGy
W/9cnv1WjqxpqnQgDjE/yJt36wKBgQDBL7p7iCWjIf+LH1/caFgPchJENd4YZZrB
bhGA/tuo6VtJcarc/Etx3DGbKhnJq13LxRRLjyHlPhw/k7oZBdaVK27I+vNfw5Lj
c2KZCYbFnF3kbP5ryuMW0QqGbkZZ/FExzwgFyAOUuCTw9L2VmKtPgbP9ywDTJc0Z
Jq/pdzOe7QKBgFOpxn4dvvIH4DgQlk9+2yMcgoduFw5EcC6bQVeXtrCf7elVzTdG
q0vHjQ5gtPJ6GD9ZGIkKusqT6TGhpC2v3SoiKO7CJmFo6tXELbOALhZY2gOWTNqj
q59EzYFxin7AHn/rKb7Lvmm4zF844plI77NLf2nX5EwwF9r0CBmc7F/hAoGAUctH
ha4rYVqvu9PY3pU/U6rUmRTFqEa8s1FLD/bYQjgrcnkyAsa/msHELxIwQPbRi8kx
wpwjmdAmXbTKgnW6WQY+rdGy4cUImEzuXiVubpS6HFEZl8IbTDnN3wUpvEfciN5D
YO9AVONyoKK+8mvlfJBKCRa+jqfeotuCd7MEpDECgYAhWcDt6aXSsUOtq+jgVNtC
oi9Cnm4FNW7Z/VVgCCRFIwHxpqqAau63/naSGxkLUlK+UOStReiLC2D4FPrqs9Jh
scUH9hTIp3hxwznZBRFkuvUOm3h6CwQ0t3km7AffLRsGQZ9EMlvNb4T5mR/Izgxy
smcEPJfJgX61fx7c//bU6Q==
-----END PRIVATE KEY-----

SSL certificate (/etc/ssl/certs/sxcert.pem):
-----BEGIN CERTIFICATE-----
MIIDpzCCAo+gAwIBAgIJAODcwxKZHi35MA0GCSqGSIb3DQEBCwUAMDsxCzAJBgNV
BAYTAkdCMQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1
c3RlcjAeFw0xNDAzMjExNDU2NTdaFw0xOTAzMjAxNDU2NTdaMDsxCzAJBgNVBAYT
AkdCMQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1c3Rl
cjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJg12OfI2CUdlDy9o7Uc
la1n956sHYfzRfJBNmkMVFLzuoZQ1ytG5OM3TOTKly55dpbVQ3/cvwoqfu2OaEt7
qq4bxC4rD/9SIkMMZMMQgI6d21h8GHTvM1H3OBpBGL2WipAHoip97gr9POUQ40yN
WOLU4Xkkj/dKDEQPChDj7V6CEO4wYwSXMnhUkKiNeGIS2/CjVmu4Vf5eiB2urhiz
o9yHuf0WjBO5zaf4hHE9cpvoKqccLW30AoQGZtqg/7TOEzxkUKXdYVEKG9skXxP1
lnmreeiXrrw0gs03V/hFF7Tv6lv25t9ao0AGG4jmFBScbzXAqNwjONYrgdn7dszC
Do8CAwEAAaOBrTCBqjAdBgNVHQ4EFgQUs7Zs8qeEtPdNQ7l3zs3f2v+MTrswawYD
VR0jBGQwYoAUs7Zs8qeEtPdNQ7l3zs3f2v+MTruhP6Q9MDsxCzAJBgNVBAYTAkdC
MQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1c3RlcoIJ
AODcwxKZHi35MA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQDAgEGMA0GCSqGSIb3
DQEBCwUAA4IBAQBGwoULuHM5svPvV7c0tdsBmxovrhCYkMg4MwtPJ8eJQckyrCP3
fIU1VMXXeHKegaZ4q3QzIV9DDO1XB9TzifZ8yKm7a2/NlUnvgLQCGu82H/226YLE
abqoipcJsAANo5+2qGYEmYDODmLOnToaCX5bcmbLc1tcG4uf/x88O+PGLgh/h5+9
MUMlffyJWAE5eJN1rk9T5k0Onm5PElQLP/ZQecodHGL9Xxzgj09kLfwbRmUruGu/
ft4Ru0oOrQDIDWxQuiBitawQKX/tyaGkpX+g38gyFwDiPINo2q/IHeckxX5EHgF3
YGgPNaWwBnH3jfsJ/kMXcJS52q/zPOIvUCz0
-----END CERTIFICATE-----

Cluster: sx://mycluster
This node: 192.168.1.101
Cluster UUID: 01dca714-8cc9-4e26-960e-daf04892b1e2
Cluster key: CLUSTER/ALLNODE/ROOT/USERwBdjfz3tKcnTF2ouWIkTipreYuYjAAA
Admin key: 0DPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5y0SOi3IMawAA
Internal cluster protocol: SECURE
Used disk space: 16.75M
Actual data size: 453.00K
List of nodes:
         * ec4d9d63-9fa3-4d45-838d-3e521f124ed3 192.168.1.101 (192.168.1.101) 500.00G
Storage location: /var/lib/sxserver/storage
Run as user: nobody
Sockets and pidfiles in: /var/run/sxserver
Logs in: /var/log/sxserver/sxfcgi.log

--- END OF SUMMARY ---

Congratulations, the new node is up and running!
You can control it with '/usr/sbin/sxserver'

You can add a new node to the cluster by running 'sxsetup' on another
server. When prompted, enter the 'admin key', 'SSL private key' and
'SSL certificate' printed above.

You can run 'sxacl useradd joe sx://admin@mycluster' to add a new user.
To create a new volume owned by user 'joe' run:
'sxvol create --owner joe --replica N --size SIZE sx://admin@mycluster/VOLNAME'
```

When the script finishes successfully, the node is already functional. Please notice the admin key listed at the end of the summary: it will be needed for both adding more nodes and accessing the cluster. You can always retrieve the admin key with the following command:

```
# sxsetup --info
--- SX INFO ---
SX Version: 2.0
Cluster name: mycluster
Cluster port: 443
Cluster UUID: 01dca714-8cc9-4e26-960e-daf04892b1e2
Cluster key: CLUSTER/ALLNODE/ROOT/USERwBdjfz3tKcnTF2ouWIkTipreYuYjAAA
Admin key: 0DPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5y0S0i3IMawAA
Internal cluster protocol: SECURE
Used disk space: 16.75M
Actual data size: 453.00K
List of nodes:
         * ec4d9d63-9fa3-4d45-838d-3e521f124ed3 192.168.1.101 (192.168.1.101) 500.00G
Storage location: /var/lib/sxserver/storage
SSL private key: /etc/ssl/private/sxkey.pem
SX Logfile: /var/log/sxserver/sxfcgi.log
```

That's it — your SX storage is already up and running! You can now go to the next step and add more nodes or go to the next chapter and learn how to perform basic client operations.

3.3 Adding more nodes to the cluster

Follow these steps to add a new node to the cluster:

- Run `sxsetup --info` on one of the nodes of the cluster
- Collect the following information:
  - Cluster name
  - Admin key
  - One of the IP addresses from the list of nodes
- Install S^X using a binary package or source code
- Run `sxsetup` and provide the collected information. Below we assume the new node is 192.168.1.102 and its size is 250 GB.

```
# sxsetup
--- SKYLABLE SX CONFIGURATION SCRIPT ---

The script will help you to create or extend a Skylable SX data
cluster.

--- CLUSTER NAME ---

Clients will access your cluster using a sx://clustername/volume/path
URI. It is recommended to use a FQDN for clustername, but not
required. Refer to the documentation for more info.
Enter the cluster name (use the same across all nodes) []: mycluster

--- DATA STORAGE ---

Please provide the location where all incoming data will be stored.
Path to SX storage [default=/var/lib/sxserver/storage]: <confirm default>

Please specify the maximum size of the storage for this node. You can
use M, G and T suffixes, eg. 100T for 100 terabytes.
Maximum size [default=1T]: 250G

--- NETWORKING ---
```

```
Enable SSL? (use the same setting for all nodes in the cluster) (Y/n)
<confirm default>
Enter the IP address of this node [default=192.168.1.102]: <confirm default>
Checking port 443 on 192.168.1.102 ... OK

--- CLUSTER CONFIGURATION ---

Is this (192.168.1.102) the first node of a new cluster? (Y/n) n
Please provide the IP address of a working node in 'mycluster'.
IP address: 192.168.1.101

The admin key is required to join the existing cluster.
If you don't have it, run sxsetup --info on 192.168.1.101.
Below you can provide the key itself or path to the file
containing the key.
Admin key or path to key-file:
0DPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5y0SOi3IMawAA

--- SSL CONFIGURATION ---
Automatically obtained SSL private key from 192.168.1.101

Automatically obtained SSL certificate from 192.168.1.101

--- YOUR CHOICES ---

Cluster: sx://mycluster
Node: 192.168.1.102
Use SSL: yes
Storage: /var/lib/sxserver/storage
Run as user: nobody

Is this correct? (Y/n) <confirm default>

--- NODE INITIALIZATION ---

Connecting to 192.168.1.101
Server certificate:
        Subject: C=UK; L=London; O=SX; CN=mycluster
        Issuer: C=UK; L=London; O=SX; CN=mycluster
        SHA1 fingerprint: 627917198424168ad0c144e721567eb4ebc90db1

Do you trust this SSL certificate? [y/N] y
Starting SX.fcgi
Starting sxhttpd
SX node started successfully
Cluster UUID: 01dca714-8cc9-4e26-960e-daf04892b1e2
Cluster key: CLUSTER/ALLNODE/ROOT/USERwBdjfz3tKcnTF2ouWIkTipreYuYjAAA
Admin key: 0DPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5y0SOi3IMawAA
Internal cluster protocol: SECURE
Used disk space: 16.75M
Actual data size: 453.00K
List of nodes:
    - ec4d9d63-9fa3-4d45-838d-3e521f124ed3 192.168.1.101 (192.168.1.101) 500.00G
    * 02e01f5d-80d8-4a01-b1f7-a56eecb8aef5 192.168.1.102 (192.168.1.102) 250.00G

--- CONFIGURATION SUMMARY ---

SSL private key (/etc/ssl/private/sxkey.pem):
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCYNdtHyNglHZQ8
va01HJWtZ/eerB2H80XyQTZpDFRS87qGUNcrRudDN09EypcueXaW1UN/3L8KKn7t
tGhLe6quG8QuKw//UiJDDGTDEICOndtYfBh07zNR9zgaQRi9loqQB6Iqfe4K/T9F
EONMjVji1OF5JI/3SgxEDwoQ4+1eghDuMGMElzJ4VJCojXhiEtvwo1ZruFX+Xogd
rq4Ys6Pch7n9Fowd0c2n+IRxPXKb6CqnHC1t9AKEBmbaoP+0zhM8ZFCl3WFRChvb
JF8T9ZZ5q3nol668NILNN1f4RRe07+pb9ubfWqNABhuI5hQUnG81wKjcIzjWK4HZ
+3bMwg6PAgMBAAECggEAQ+fTGmV6OKTHm4mnXYeRJzm4+SskSaC41el0EvOTMybV
UlMCi6YoSo6EaNZROESsKYKfiI29FRX8ZqQT24kijmaI0WgYzPmhm3QOCBB2qim2
z/UdHB4TMUAv4ValaP+edb9SE872wiRVc8SjA2YT/66loNw09kgszLhA72QgZAbG
xmxVwCNTRFd7dg4Wmy1OQz3YVOnlC3Qs8C8LoGoOOMci85quhBUw9s7J12skXGbu
ZGDtpJylgwtfc1q7nojaFkWenGCA9D1HB8zCqKPkhMh+HtA26g8VdFaHPVBzw/pz
avv5r9gLnBETwHfM3XuIYv7h3wowE5uAKVhgvL8w0QKBgQDJs2avbYOwgcEEOf7L
nPRqmb5XjJE329KsyIzo4YwOrZDjQXSYrBjifoBIJzUReDDB7ww5lt0Xy3MExeS4
ngLO/oWotjd7jGU+EdABozKwW3bZuyUTSqTeQJwo+aIhjNtiyMrnpFy3vjYrJKGy
W/9cnv1WjqxpqnQgDjE/yJt36wKBgQDBL7p7iCWjIf+LH1/caFgPchJENd4YZZrB
```

```
bhGA/tuo6VtJcarc/Etx3DGbKhnJq13LxRRLjyHlPhw/k7oZBdaVK27I+vNfw5Lj
c2KZCYbFnF3kbP5ryuMW0QqGbkZZ/FExzwgFyAOUuCTw9L2VmKtPgbP9ywDTJc0Z
Jq/pdzOe7QKBgFOpxn4dvvIH4DgQlk9+2yMcgoduFw5EcC6bQVeXtrCf7elVzTdG
q0vHjQ5gtPJ6GD9ZGIkKusqT6TGhpC2v3SoiKO7CJmFo6tXELbOALhZY2gOWTNqj
q59EzYFxin7AHn/rKb7Lvmm4zF844plI77NLf2nX5EwwF9r0CBmc7F/hAoGAUctH
ha4rYVqvu9PY3pU/U6rUmRTFqEa8s1FLD/bYQjgrcnkyAsa/msHELxIwQPbRi8kx
wpwjmdAmXbTKgnW6WQY+rdGy4cUImEzuXiVubpS6HFEZl8IbTDnN3wUpvEfciN5D
YO9AVONyoKK+8mvlfJBKCRa+jqfeotuCd7MEpDECgYAhWcDt6aXSsUOtq+jgVNtC
oi9Cnm4FNW7Z/VVgCCRFIwHxpqqAau63/naSGxkLUlK+U0StReiLC2D4FPrqs9Jh
scUH9hTIp3hxwznZBRFkuvUOm3h6CwQ0t3km7AffLRsGQZ9EMlvNb4T5mR/Izgxy
smcEPJfJgX61fx7c//bU6Q==
-----END PRIVATE KEY-----


SSL certificate (/etc/ssl/certs/sxcert.pem):
-----BEGIN CERTIFICATE-----
MIIDpzCCAo+gAwIBAgIJAODcwxKZHi35MA0GCSqGSIb3DQEBCwUAMDsxCzAJBgNV
BAYTAkdCMQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1
c3RlcjAeFw0xNDAzMjExNDU2NTdaFw0xOTAzMjAxNDU2NTdaMDsxCzAJBgNVBAYT
AkdCMQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1c3Rl
cjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJg120fI2CUdlDy9o7Uc
la1n956sHYfzRfJBNmkMVFLzuoZQ1ytG5OM3TOTKly55dpbVQ3/cvwoqfu20aEt7
qq4bxC4rD/9SIkMMZMMQgI6d21h8GHTvM1H3OBpBGL2WipAHoip97gr9P0UQ40yN
WOLU4Xkkj/dKDEQPChDj7V6CEO4wYwSXMnhUkKiNeGIS2/CjVmu4Vf5eiB2urhiz
o9yHuf0WjB05zaf4hHE9cpvoKqccLW30AoQGZtqg/7TOEzxkUKXdYVEKG9skXxP1
lnmreeiXrrw0gs03V/hFF7Tv6lv25t9ao0AGG4jmFBScbzXAqNwjONYrgdn7dszC
Do8CAwEAAaOBrTCBqjAdBgNVHQ4EFgQUs7Zs8qeEtPdNQ7l3zs3f2v+MTrswawYD
VR0jBGQwYoAUs7Zs8qeEtPdNQ7l3zs3f2v+MTruhP6Q9MDsxCzAJBgNVBAYTAkdC
MQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1c3RlcoIJ
AODcwxKZHi35MA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQDAgEGMA0GCSqGSIb3
DQEBCwUAA4IBAQBGwoULuHM5svPvV7c0tdsBmxovrhCYkMg4MwtPJ8eJQckyrCP3
fIU1VMXXeHKegaZ4q3QzIV9DDO1XB9TzifZ8yKm7a2/NlUnvgLQCGu82H/226YLE
abqoipcJsAANo5+2qGYEmYDODmLOnToaCX5bcmbLc1tcG4uf/x88O+PGLgh/h5+9
MUMlffyJWAE5eJN1rk9T5k0Onm5PElQLP/ZQecodHGL9Xxzgj09kLfwbRmUruGu/
ft4Ru0oOrQDIDWxQuiBitawQKX/tyaGkpX+g38gyFwDiPINo2q/IHeckxX5EHgF3
YGgPNaWwBnH3jfsJ/kMXcJS52q/zPOIvUCz0
-----END CERTIFICATE-----


Cluster: sx://mycluster
This node: 192.168.1.102
Port number: 443
Cluster UUID: 01dca714-8cc9-4e26-960e-daf04892b1e2
Cluster key: CLUSTER/ALLNODE/ROOT/USERwBdjfz3tKcnTF2ouWIkTipreYuYjAAA
Admin key: 0DPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5y0SOi3IMawAA
Internal cluster protocol: SECURE
Used disk space: 16.75M
Actual data size: 453.00K
List of nodes:
    - ec4d9d63-9fa3-4d45-838d-3e521f124ed3 192.168.1.101 (192.168.1.101) 500.00G
    * 02e01f5d-80d8-4a01-b1f7-a56eecb8aef5 192.168.1.102 (192.168.1.102) 250.00G
Storage location: /var/lib/sxserver/storage
Run as user: nobody
Sockets and pidfiles in: /var/run/sxserver
Logs in: /var/log/sxserver/sxfcgi.log

--- END OF SUMMARY ---

Congratulations, the new node is up and running!
You can control it with '/usr/sbin/sxserver'

You can add a new node to the cluster by running 'sxsetup' on another
server. When prompted, enter the 'admin key', 'SSL private key' and
'SSL certificate' printed above.

You can run 'sxacl useradd joe sx://admin@mycluster' to add a new user.
To create a new volume owned by user 'joe' run:
'sxvol create --owner joe --replica N --size SIZE sx://admin@mycluster/VOLNAME'
```

The node successfully joined the cluster — at the end of the summary you can see the current list of nodes in the cluster. Repeat the same steps to add more nodes to the cluster.

3.4 AUTOMATIC NODE CONFIGURATION

The process of adding new nodes can be automated with the use of `--config-file` option of `sxsetup`. In the following example we assume the cluster has been configured to use a couple of nodes as described in the previous section, and we will be adding a third node with the IP address of 192.168.1.103 and size of 250GB, which has the S^X software installed the same way as on the other nodes. We will use the `sxsetup.conf` file from the node 192.168.1.102 as a template, which has the following content:

```
# cat /etc/sxserver/sxsetup.conf
############################################################################
#                    !!! DO NOT EDIT THIS FILE !!!                         #
#                                                                          #
#    This file was generated during node creation with sxsetup.            #
#    Some of the variables defined below are used by sxserver and other    #
#    scripts, however the main purpose of this file is to provide         #
#    a template for creating new nodes with sxsetup --config-file.         #
#    Changing parameters such as SX_NODE_SIZE directly in this file        #
#    will have no effect *after* the node was created.                     #
#                                                                          #
############################################################################
SX_CLUSTER_NAME="mycluster"
SX_DATA_DIR="/var/lib/sxserver/storage"
SX_RUN_DIR="/var/run/sxserver"
SX_LIB_DIR="/var/lib/sxserver"
SX_LOG_FILE="/var/log/sxserver/sxfcgi.log"
SX_NODE_SIZE="250G"
SX_NODE_IP="192.168.1.102"
SX_NODE_INTERNAL_IP=""
SX_EXISTING_NODE_IP="192.168.1.1"
SX_SERVER_USER="nobody"
SX_SERVER_GROUP="nogroup"
SX_CHILDREN_NUM="24"
SX_RESERVED_CHILDREN_NUM="8"
SX_PORT="443"
SX_USE_SSL="yes"
SX_SSL_KEY_FILE="/etc/ssl/private/sxkey.pem"
SX_SSL_CERT_FILE="/etc/ssl/certs/sxcert.pem"
SX_SSL_KEY="-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCYNdtHyNglHZQ8
vaO1HJWtZ/eerB2H8OXyQTZpDFRS87qGUNcrRudDN09EypcueXaW1UN/3L8KKn7t
tGhLe6quG8QuKw//UiJDDGTDEICOndtYfBh07zNR9zgaQRi9loqQB6Iqfe4K/T9F
EONMjVji1OF5JI/3SgxEDwoQ4+1eghDuMGMElzJ4VJCojXhiEtvwo1ZruFX+Xogd
rq4Ys6Pch7n9FowdOc2n+IRxPXKb6CqnHC1t9AKEBmbaoP+0zhM8ZFCl3WFRChvb
JF8T9ZZ5q3nol668NILNN1f4RRe07+pb9ubfWqNABhuI5hQUnG81wKjcIzjWK4HZ
+3bMwg6PAgMBAAECggEAQ+fTGmV6OKTHm4mnXYeRJzm4+SskSaC41elOEvOTMybV
UlMCi6YoSo6EaNZROESsKYKfiI29FRX8ZqQT24kijmaI0WgYzPmhm3QOCBB2qim2
z/UdHB4TMUAv4ValaP+edb9SE872wiRVc8SjA2YT/66loNw09kgszLhA72QgZAbG
xmxVwCNTRFd7dg4Wmy1OQz3YVOnlC3Qs8C8LoGoOOMci85quhBUw9s7J12skXGbu
ZGDtpJylgwtfc1q7nojaFkWenGCA9D1HB8zCqKPkhMh+HtA26g8VdFaHPVBzw/pz
avv5r9gLnBETwHfM3XuIYv7h3wowE5uAKVhgvL8w0QKBgQDJs2avbYOwgcEEOf7L
nPRqmb5XjJE329KsyIzo4YwOrZDjQXSYrBjifoBIJzUReDDB7ww5lt0Xy3MExeS4
ngL0/oWotjd7jGU+EdABozKwW3bZuyUTSqTeQJwo+aIhjNtiyMrnpFy3vjYrJKGy
W/9cnv1WjqxpqnQgDjE/yJt36wKBgQDBL7p7iCWjIf+LH1/caFgPchJENd4YZZrB
bhGA/tuo6VtJcarc/Etx3DGbKhnJq13LxRRLjyHlPhw/k7oZBdaVK27I+vNfw5Lj
c2KZCYbFnF3kbP5ryuMW0QqGbkZZ/FExzwgFyAOUuCTw9L2VmKtPgbP9ywDTJcOZ
Jq/pdzOe7QKBgFOpxn4dvvIH4DgQlk9+2yMcgoduFw5EcC6bQVeXtrCf7elVzTdG
q0vHjQ5gtPJ6GD9ZGIkKusqT6TGhpC2v3SoiKO7CJmFo6tXELbOALhZY2gOWTNqj
q59EzYFxin7AHn/rKb7Lvmm4zF844plI77NLf2nX5EwwF9r0CBmc7F/hAoGAUctH
ha4rYVqvu9PY3pU/U6rUmRTFqEa8s1FLD/bYQjgrcnkyAsa/msHELxIwQPbRi8kx
wpwjmdAmXbTKgnW6WQY+rdGy4cUImEzuXiVubpS6HFEZl8IbTDnN3wUpvEfciN5D
YO9AVONyoKK+8mvlfJBKCRa+jqfeotuCd7MEpDECgYAhWcDt6aXSsUOtq+jgVNtC
oi9Cnm4FNW7Z/VVgCCRFIwHxpqqAau63/naSGxkLUlK+UOStReiLC2D4FPrqs9Jh
scUH9hTIp3hxwznZBRFkuvUOm3h6CwQ0t3km7AffLRsGQZ9EMlvNb4T5mR/Izgxy
smcEPJfJgX61fx7c//bU6Q==
-----END PRIVATE KEY-----"
SX_SSL_CERT="-----BEGIN CERTIFICATE-----
MIIDpzCCAo+gAwIBAgIJAODcwxKZHi35MA0GCSqGSIb3DQEBCwUAMDsxCzAJBgNV
BAYTAkdCMQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1
c3RlcjAeFw0xNDAzMjExNDU2NTdaFw0xOTAzMjAxNDU2NTdaMDsxCzAJBgNVBAYT
```

```
AkdCMQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1c3Rl
cjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJg12OfI2CUdlDy9o7Uc
la1n956sHYfzRfJBNmkMVFLzuoZQ1ytG5OM3TOTKly55dpbVQ3/cvwoqfu2OaEt7
qq4bxC4rD/9SIkMMZMMQgI6d21h8GHTvM1H3OBpBGL2WipAHoip97gr9POUQ40yN
WOLU4Xkkj/dKDEQPChDj7V6CEO4wYwSXMnhUkKiNeGIS2/CjVmu4Vf5eiB2urhiz
o9yHuf0WjBO5zaf4hHE9cpvoKqccLW30AoQGZtqg/7TOEzxkUKXdYVEKG9skXxP1
lnmreeiXrrw0gs03V/hFF7Tv6lv25t9ao0AGG4jmFBScbzXAqNwjONYrgdn7dszC
Do8CAwEAAaOBrTCBqjAdBgNVHQ4EFgQUs7Zs8qeEtPdNQ7l3zs3f2v+MTrswawYD
VR0jBGQwYoAUs7Zs8qeEtPdNQ7l3zs3f2v+MTruhP6Q9MDsxCzAJBgNVBAYTAkdC
MQswCQYDVQQIEwJVSzELMAkGA1UEChMCU1gxEjAQBgNVBAMTCW15Y2x1c3RlcoIJ
AODcwxKZHi35MA8GA1UdEwEB/wQFMAMBAf8wCwYDVR0PBAQDAgEGMA0GCSqGSIb3
DQEBCwUAA4IBAQBGwoULuHM5svPvV7c0tdsBmxovrhCYkMg4MwtPJ8eJQckyrCP3
fIU1VMXXeHKegaZ4q3QzIV9DDO1XB9TzifZ8yKm7a2/NlUnvgLQCGu82H/226YLE
abqoipcJsAANo5+2qGYEmYDODmLOnToaCX5bcmbLc1tcG4uf/x88O+PGLgh/h5+9
MUMlffyJWAE5eJN1rk9T5k0Onm5PElQLP/ZQecodHGL9Xxzgj09kLfwbRmUruGu/
ft4Ru0oOrQDIDWxQuiBitawQKX/tyaGkpX+g38gyFwDiPINo2q/IHeckxX5EHgF3
YGgPNaWwBnH3jfsJ/kMXcJS52q/zPOIvUCz0
-----END CERTIFICATE-----"
SX_CFG_VERSION="3"
SX_CLUSTER_UUID="01dca714-8cc9-4e26-960e-daf04892b1e2"
SX_CLUSTER_KEY=CLUSTER/ALLNODE/ROOT/USERwBdjfz3tKcnTF2ouWIkTipreYuYjAAA
SX_ADMIN_KEY=ODPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5yOSOi3IMawAA
```

As instructed in the header, we shouldn't modify the original file. Instead, we will copy the file to `/root/sxsetup.conf` on the new node and update `SX_NODE_IP` to point to 192.168.1.3 with the other settings left untouched. After that we run `sxsetup` on the new node as follows:

```
# sxsetup --config-file /root/sxsetup.conf
Using config file /root/sxsetup.conf
[...]
Cluster: sx://mycluster
This node: 192.168.1.103
Port number: 443
Cluster UUID: 01dca714-8cc9-4e26-960e-daf04892b1e2
Cluster key: CLUSTER/ALLNODE/ROOT/USERwBdjfz3tKcnTF2ouWIkTipreYuYjAAA
Admin key: ODPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5yOSOi3IMawAA
Internal cluster protocol: SECURE
Used disk space: 16.75M
Actual data size: 453.00K
List of nodes:
    - ec4d9d63-9fa3-4d45-838d-3e521f124ed3 192.168.1.101 (192.168.1.101) 500.00G
    - 02e01f5d-80d8-4a01-b1f7-a56eecb8aef5 192.168.1.102 (192.168.1.102) 250.00G
    * 912b6125-9228-4227-93ce-57f6f6e248c0 192.168.1.103 (192.168.1.103) 250.00G
Storage location: /var/lib/sxserver/storage
Run as user: nobody
Sockets and pidfiles in: /var/run/sxserver
Logs in: /var/log/sxserver/sxfcgi.log

--- END OF SUMMARY ---

Congratulations, the new node is up and running!
You can control it with '/usr/sbin/sxserver'
[...]
```

The node has been automatically configured and successfully joined the cluster.

CHAPTER 4

CLUSTER MANAGEMENT

4.1 LOCAL NODE STATUS AND CONFIGURATION

You can check status of a specific node by running `sxserver status` on that node:

```
# sxserver status
--- SX STATUS ---
sx.fcgi is running (PID 14394)
sxhttpd is running (PID 14407)
```

Run `sxsetup --info` to display the node's configuration:

```
# sxsetup --info
--- SX INFO ---
SX Version: 2.0
Cluster name: mycluster
Cluster port: 443
Cluster UUID: 01dca714-8cc9-4e26-960e-daf04892b1e2
Cluster key: CLUSTER/ALLNODE/ROOT/USERwBdjfz3tKcnTF2ouWIkTipreYuYjAAA
Admin key: 0DPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5y0S0i3IMawAA
Internal cluster protocol: SECURE
Used disk space: 16.75M
Actual data size: 453.00K
List of nodes:
         * ec4d9d63-9fa3-4d45-838d-3e521f124ed3 192.168.1.101 (192.168.1.101) 500.00G
Storage location: /var/lib/sxserver/storage
SSL private key: /etc/ssl/private/sxkey.pem
SX Logfile: /var/log/sxserver/sxfcgi.log
```

This gives you the information about local services and disk usage, but also provides the admin key, which is needed for accessing the cluster itself.

4.2 ADMINISTRATOR ACCESS

During cluster deployment a default admin account gets created and initialized. For security reasons, the account uses a randomly generated key instead of a password. You should be able to access the cluster from any node using `sx://admin@mycluster` profile. In order to manage the cluster remotely or from another system account, you need to initialize access to the cluster using `sxinit` [1]. In the example below we use the default admin account created during cluster setup. Since “mycluster” is not a DNS name, we need to point `sxinit` to one of the nodes of the cluster — this will allow it automatically discover the IP addresses of the other nodes. Additionally, we create an alias `@cluster`, which later can be used instead of `sx://admin@mycluster`.

[1] For more information about access profiles please see section 5.1 on page 32.

```
$ sxinit --key -l 192.168.1.101 -A @cluster sx://admin@mycluster
Warning: self-signed certificate:

        Subject: C=GB, ST=UK, O=SX, CN=mycluster
        Issuer: C=GB, ST=UK, O=SX, CN=mycluster
        SHA1 Fingerprint: 84:EF:39:80:1E:28:9C:4A:C8:80:E6:56:57:A4:CD:64:2E:23:99:7A

Do you trust this SSL certificate? [y/N] y
Trusting self-signed certificate
Please enter the user key: ODPiKuNIrrVmD8IUCuw1hQxNqZfIkCY+oKwxi5zHSPn5yOSOi3IMawAA
```

4.3 User management

S^X similarly to UNIX systems supports two types of users: regular and administrators. A new cluster has only a single 'admin' user enabled by default. The administrators can perform all cluster operations and access all data in the cluster (except for encrypted volumes), while the regular users can only work with volumes they have access to. It is recommended to only use the admin account for administrative purposes and perform regular operations as a normal user.

Creating a new user

Use `sxacl useradd` to add new users to the cluster:

```
$ sxacl useradd jeff @cluster
Enter password for user 'jeff'
Enter password:
Re-enter password:
User successfully created!
Name: jeff
Key : FqmlTd9CWZUuPBGMdjE46DaT1/3kx+EYbahlrhcdVpy/9ePfrtWCIgAA
Type: normal

Run 'sxinit sx://jeff@mycluster' to start using the cluster as user 'jeff'.
```

By default a regular user account gets created and the key is generated from the password. The user can later authenticate both using the password or the key. It's also possible to automatically generate a random key by passing the `--generate-key` option.

Listing users

In order to list all users in the cluster run:

```
$ sxacl userlist @cluster
admin (admin)
jeff (normal)
```

Only cluster administrators can list users.

Key and password management

S^X uses special authentication keys, which are either randomly generated or based on passwords. It is possible to obtain the existing key or issue a new one for any user in the cluster. To retrieve the current authentication key for user 'jeff' run:

```
$ sxacl usergetkey jeff @cluster
5tJdVr+RSpA/IPuFeSwUeePtKdbDLWUKqoaoZLkmCcXTw5qzPg5e7AAA
```

A new password/key can be set at any time by running:

```
$ sxacl usernewkey jeff @sxtest
Enter new password for user 'jeff'
Enter password:
```

```
Re-enter password:
Key successfully changed!
Name   : jeff
New key: FqmlTd9CWZUuPBGMdjE46DaT1/3MSHk9TLH27dFf5Zd61lEbWEeAqgAA
Run 'sxinit sx://jeff@sxtest' and provide the new key for user 'jeff'.
```

As long as the user can access the cluster, it can change its own key. The cluster administrator can force a key change for any user, what can also be used to temporarily block access to the cluster for a specified user.

Removing a user

Use `sxacl userdel` to permanently delete a user from the cluster:

```
$ sxacl userdel jeff @cluster
User 'jeff' successfully removed.
```

All volumes owned by the user will be automatically reassigned to the cluster administrator performing the removal.

4.4 Volume management

Volumes are logical partitions of the S^X storage, which are of specific size and accessible by a particular group of users. The volumes can be used in connection with client side filters to perform additional operations, such as compression or encryption. Only cluster administrators can create and remove volumes.

Creating a plain volume

Below we create a basic volume of size 50GB owned by the user 'jeff' and fully replicated on two nodes.

```
$ sxvol create -o jeff -r 2 -s 50G @cluster/vol-jeff
Volume 'vol-jeff' (replica: 2, size: 50G, max-revisions: 1) created.
```

By default, a volume will only keep a single revision of each file (`max-revisions` parameter set to 1). The revisions are previous versions of the file stored when the file gets modified. For example, when a volume gets created with `max-revisions` set to 3, and some file gets modified multiple times, then the latest 3 versions of the file will be preserved. All revisions are accounted for their size. See the information about `sxrev` in section 5.2 on page 33 on how to manage file revisions.

Creating a filtered volume

Filters are client side plugins, which perform operations on files or their contents, before and after they get transferred to/from the S^X cluster. When a filter gets assigned to a volume, all remote clients will be required to have support for that particular filter in order to access the volume. Run the following command to list the available filters:

```
$ sxvol filter --list
Name            Ver     Type        Short description
----            ---     ----        -----------------
undelete        1.2     generic     Backup removed files
zcomp           1.2     compress    Zlib Compression Filter
aes256          2.0     crypt       Encrypt data using AES-256-CBC-HMAC-512
attribs         1.3     generic     File Attributes
```

We will create an encrypted volume for user 'jeff'. To obtain more information about the `aes256` filter run:

```
$ sxvol filter -i aes256
'aes256' filter details:
Short description: Encrypt data using AES-256-CBC-HMAC-512 mode.
Summary: The filter automatically encrypts and decrypts all data using
         OpenSSL's AES-256 in CBC-HMAC-512 mode.
Options:
        setkey (set a permanent key when creating a volume)
        paranoid (don't use key files at all - always ask for a password)
        encrypt_filenames: enable encryption of filenames (may be slow with big number of
     files)
        salt:HEX (force given salt, HEX must be 32 chars long)
UUID: 15b0ac3c-404f-481e-bc98-6598e4577bbd`
Type: crypt
Version: 2.0
```

Now run the following command to create an encrypted volume:

```
$ sxvol create -o jeff -r 2 -s 50G -f aes256 @cluster/vol-jeff-aes
Volume 'vol-jeff-aes' (replica: 2, size: 50G, max-revisions: 1) created.
```

The user will be asked to set a new password while accessing the volume for the first time.

Listing all volumes

To get a list of all volumes in the cluster run `sxls` with the cluster argument as an administrator. When the same command is run by a normal user, it will list all volumes, to which the user has access.

```
$ sxls -lH @cluster
  VOL  rep:2  rev:1  rw  -       0  50.00G  0% sx://admin@mycluster/vol-jeff
  VOL  rep:2  rev:1  rw  aes256  0  50.00G  0% sx://admin@mycluster/vol-jeff-aes
```

The `-l (--long-format)` flag makes `sxls` provide more information about the volumes, and `-H` converts all sizes into a human readable form. The parameters right after the volume marker `VOL` are: number of replicas, maximum number of revisions, access permissions for the user performing the listing (in this case for the administrator), active filter, used space, size of the volume, and the usage percentage.

Managing volume permissions

Cluster administrators and volume owners can grant or revoke access to the volumes to other users. The owner can also grant another user the manager privilege, which allows to manage the volume permissions. To list the current access control list for the volume `vol-jeff` run:

```
$ sxacl volshow @cluster/vol-jeff
admin: read write
jeff: read write manager owner
(all admin users): read write admin
```

To grant full access to user 'bob' run:

```
$ sxacl volperm --grant=read,write bob @cluster/vol-jeff
New volume ACL:
admin: read write
bob: read write
jeff: read write manager owner
(all admin users): read write admin
```

User 'bob' can now upload, download and remove files from the volume but cannot make any changes to the volume settings (this is restricted to admins, managers and owners). To revoke write access from user 'bob' run:

```
$ sxacl volperm --revoke=write bob @cluster/vol-jeff
New volume ACL:
admin: read write
bob: read
jeff: read write manager owner
(all admin users): read write admin
```

Now ‘bob’ can only read files but cannot upload or remove anything.

Changing volume settings

Some of the volume settings such as its size or ownership can be modified at a later time. For example, the cluster administrator may want to extend a volume size or shrink it to forbid users from storing more data — when the new size is lower than the current space usage of the volume the existing contents will remain untouched — but in order to upload more data to the volume, the user will have to make enough space to satisfy the new limit.

To resize the volume ‘vol-jeff’ to 100GB run:

```
$ sxvol modify --size 100G @cluster/vol-jeff
```

4.5 Node management

In section 3.3 on page 11 we described how to add new nodes to a cluster. This section covers other modifications to an existing cluster, such as node repair, resize or delete. In the examples below we will manage a cluster with four nodes, 500GB each, with an administrator profile configured as `@cluster2`.

Remote cluster status

To get information about remote cluster status run the following command:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Current configuration: 536870912000/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     536870912000/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     536870912000/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     536870912000/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.7) - checksum: 18024964248989723179
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 536870912000,
      status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 536870912000,
      status: follower, online: yes
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 536870912000,
      status: leader, online: yes
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 536870912000,
      status: follower, online: yes
```

The first line provides the list of cluster nodes in the following format:

```
SIZE/IP_ADDRESS[/INTERNAL_IP_ADDRESS]/UUID
```

where `SIZE` is in bytes and `UUID` is a unique identifier assigned to a node when joining the cluster.

In order to get information about the individual nodes run (`-H` converts sizes to human readable form):

```
$ sxadm cluster --list-nodes -H @cluster2
Node d3f8ad83-d003-4aaa-bbfb-73359af85991 status:
    Versions:
        SX: 2.0
        HashFS: SX-Storage 2.0
    System:
        Name: Linux
        Architecture: x86_64
        Release: 3.2.0-4-amd64
        Version: #1 SMP Debian 3.2.51-1
        CPU(s): 8
        Endianness: little-endian
        Local time: 2015-05-07 17:03:21 CEST
        UTC time: 2015-05-07 15:03:21 UTC
    Network:
        Public address: 192.168.100.1
        Internal address: 192.168.100.1
    Storage:
        Storage directory: /var/lib/sxserver/storage
        Allocated space: 259.47G
        Used space: 227.30G
    Storage filesystem:
        Block size: 4.00K
        Total size: 1.14T
        Available: 644.28G
        Used: 53.28%
    Memory:
        Total: 31.36G
[...]
```

Rebalance mode

After making any change to the cluster, it will automatically enter into a rebalance mode. The rebalance process makes the data properly distributed among the nodes according to the new cluster scheme. During the rebalance all data operations on volumes can be performed as usual, but no changes to the cluster itself are accepted. When the cluster is rebalancing, it reports its new configuration in the status output under *"Target configuration"*. **IMPORTANT:** All nodes should be online during the rebalance process. It is not possible to cancel a running rebalance nor make any changes to the cluster during the process.

Cluster resize

The first modification we will perform is a global cluster resize. `sxadm cluster --resize` provides an easy way to shrink or grow the entire cluster, with changes applied to all nodes proportionally to their current capacity in the cluster. In our cluster all four nodes have equal sizes, therefore growing the cluster by 400GB, should result in each node being resized by 100GB:

```
$ sxadm cluster --resize +400G @cluster2
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Target configuration: 644245094400/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Current configuration: 536870912000/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     536870912000/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     536870912000/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     536870912000/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Operating mode: read-write
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.8) - checksum: 14098478712246199608
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 644245094400,
      status: follower, online: yes, activity: Relocation complete
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      status: follower, online: yes, activity: Relocation complete
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 644245094400,
      status: leader, online: yes, activity: Relocation complete
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 644245094400,
      status: follower, online: yes, activity: Relocation complete
```

All nodes were properly resized. When the rebalance process finishes, *"Target configuration"* will become *"Current configuration"*.

Node resize

In order to modify a single node, we will use a generic option `cluster --modify`, which takes a new configuration of the cluster. First, we obtain the current configuration:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Current configuration: 644245094400/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Operating mode: read-write
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.9) - checksum: 18024963750773516843
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 644245094400,
      status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      status: follower, online: yes
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 644245094400,
      status: leader, online: yes
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 644245094400,
      status: follower, online: yes
```

In order to change the size of the node 192.168.100.1 to 700GB, we provide a new configuration of the cluster with an updated specification of that node that includes the new size and the other values left untouched:

```
$ sxadm cluster --modify 751619276800/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3 @cluster2
```

It's very important to provide proper node UUIDs, otherwise the cluster won't be able to recognize the node changes. When the rebalance finishes, the new configuration of the cluster is:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Current configuration: 751619276800/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Operating mode: read-write
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.10) - checksum: 18024964785860635179
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 751619276800,
      status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      status: follower, online: yes
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 644245094400,
      status: leader, online: yes
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 644245094400,
      status: follower, online: yes
```

Node removal

To easily remove a single node from the cluster, log into the node and run `sxsetup --deactivate`:

```
# sxsetup --deactivate
This option will relocate all data stored on this node to other nodes in
the cluster and deactivate the node. The procedure can take more time
depending on the data size and the network speed. Please do not interrupt
it and don't turn off the node until the operation is finished.
Do you want to continue? (y/N) y
Waiting for cluster to finish data relocation...
Sending SIGTERM to 32644
Sending SIGTERM to 32654
Waiting for  32644 32654
The node has been successfully deactivated.
```

A generic approach to remove one or mode nodes at the same time, requires removing node specifications from the current cluster configuration. In order to remove the node 192.168.100.4 from the cluster discussed in previous sections, provide a new cluster configuration **without** specification of the node 192.168.100.4:

```
$ sxadm cluster --modify 751619276800/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9 @cluster2
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Target configuration: 751619276800/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
Current configuration: 751619276800/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.11) - checksum: 16329829800547562843
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 751619276800,
      status: follower, online: yes, activity: Relocating data (1510 out of ~50733 blocks
     processed)
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      status: follower, online: yes, activity: Relocating data (2217 out of ~48215 blocks
     processed)
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 644245094400,
      status: leader, online: yes, activity: Relocating data (2053 out of ~49712 blocks
     processed)
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 644245094400,
      status: leaving, online: yes, activity: Relocating data (3696 out of ~38212 blocks
     processed)
```

The rebalance process will move all the data out of the node 192.168.100.4 and deactivate it. When the node disappears from `cluster --info` output, it's no longer part of the cluster and can be disabled physically. Using the above approach it is possible to remove more nodes at the same time, by dropping their specifications from the cluster configuration.

Creating a bare node

A bare node is a node, which is prepared to join a specific cluster, but is not a part of the cluster yet. Bare nodes can be configured in order to replace existing nodes or to join multiple nodes at once to the cluster, rather than doing that one by one. A bare node can be configured in an automatic way, similarly to the process described in section 3.4 on page 14 — the only difference is that the option `--bare` must be additionally passed to `sxsetup`. It can also be configured in interactive mode, similarly to adding a new node as described in section 3.3 on page 11, by running `sxsetup --bare` and answering the questions.

```
# sxsetup --bare
[...]
SX node started successfully
```

```
Bare node created. Use 'sxadm cluster --modify' to join it to the cluster
or perform another operation.
Node specification: 500G/192.168.100.5
```

When the setup is finished, it provides a node specification string, which can be used with cluster modification options. Please notice the bare node has no UUID assigned — it will get it when joining the target cluster.

Performing multiple changes at once

Adding new nodes with `sxsetup` is a serialized process — one node is joined to a cluster — a rebalance is triggered and then another node can be added. With `sxadm cluster --modify` multiple operations can be merged and performed at once, resulting in a single and shorter data rebalance process. In the following example, we will replace a couple of nodes in the cluster, by adding two larger nodes and removing two existing smaller nodes. First, we obtain the current cluster configuration:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Current configuration: 536870912000/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     536870912000/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     536870912000/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     536870912000/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.11) - checksum: 16116260632263325108
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 536870912000,
      status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 536870912000,
      status: follower, online: yes
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 536870912000,
      status: leader, online: yes
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 536870912000,
      status: follower, online: yes
```

It tells us there are four 500GB nodes. Now we create a couple of bare nodes: 192.168.100.5 and 192.168.100.6, both 1TB in size:

```
-- on node 192.168.100.5 --
# sxsetup --bare
[...]
SX node started successfully
Bare node created. Use 'sxadm cluster --modify' to join it to the cluster
or perform another operation.
Node specification: 1T/192.168.100.5
```

```
-- on node 192.168.100.6 --
# sxsetup --bare
[...]
SX node started successfully
Bare node created. Use 'sxadm cluster --modify' to join it to the cluster
or perform another operation.
Node specification: 1T/192.168.100.6
```

With the following command, we will remove nodes 192.168.100.3 and 192.168.100.4 and add a couple of larger nodes 192.168.100.5 and 192.167.100.6. In order to do that, we provide a new cluster configuration, consisting of the current specifications for nodes 192.168.100.1 and 192.168.100.2 as well as the bare nodes:

```
$ sxadm cluster --modify 536870912000/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     536870912000/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2 1T/192.168.100.5
     1T/192.168.100.6 @cluster2
```

After issuing the command, the rebalance process is started, which moves all data from the nodes 192.168.100.3 and 192.168.100.4 and balances the data across the cluster, which now also includes the 1TB nodes:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Target configuration: 536870912000/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     536870912000/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     1099511627776/192.168.100.5/42ea1ec2-4127-491a-9ff9-d9fdfd7c92d0
     1099511627776/192.168.100.6/5f26e559-fca0-44aa-b2d6-eb6e8e1156b1
Current configuration: 536870912000/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     536870912000/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     536870912000/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     536870912000/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.12) - checksum: 16116260632263325108
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 536870912000,
      status: follower, online: yes, activity: Relocating data (1510 out of ~37733 blocks
     processed)
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 536870912000,
      status: follower, online: yes, activity: Relocating data (2217 out of ~38215 blocks
     processed)
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 536870912000,
      status: leaving, online: yes, activity: Relocating data (2053 out of ~39712 blocks
     processed)
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 536870912000,
      status: leaving, online: yes, activity: Relocating data (3696 out of ~38212 blocks
     processed)
  * node 42ea1ec2-4127-491a-9ff9-d9fdfd7c92d0: addr: 192.168.100.5, capacity 1099511627776,
      status: joining, online: yes, activity: Relocating data (2976 out of ~43398 blocks
     processed)
  * node 5f26e559-fca0-44aa-b2d6-eb6e8e1156b1: addr: 192.168.100.6, capacity 1099511627776,
      status: joining, online: yes, activity: Relocating data (3153 out of ~43343 blocks
     processed)
```

When the rebalance finishes, the cluster consists of two 500GB nodes: 192.168.100.1 and 192.168.100.2 and two 1TB nodes: 192.168.100.5 and 192.168.100.6:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Current configuration: 536870912000/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     536870912000/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     1099511627776/192.168.100.5/42ea1ec2-4127-491a-9ff9-d9fdfd7c92d0
     1099511627776/192.168.100.6/5f26e559-fca0-44aa-b2d6-eb6e8e1156b1
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.12) - checksum: 16116260632263325108
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 751619276800,
      status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      status: follower, online: yes
  * node 42ea1ec2-4127-491a-9ff9-d9fdfd7c92d0: addr: 192.168.100.5, capacity 1099511627776,
      status: leader, online: yes
  * node 5f26e559-fca0-44aa-b2d6-eb6e8e1156b1: addr: 192.168.100.6, capacity 1099511627776,
      status: follower, online: yes
```

The nodes 192.168.100.3 and 192.168.100.4 are no longer part of the cluster and can be turned off.

4.6 Zone configuration

By default, replication is performed across individual nodes. S^X provides a mechanism to group the nodes into zones, which can be enabled when the cluster should be rack aware or its data distributed across different regions (geo-replica).

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
```

```
Current configuration: 644245094400/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Operating mode: read-write
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.9) - checksum: 18024963750773516843
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 644245094400,
      status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      status: follower, online: yes
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 644245094400,
      status: leader, online: yes
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 644245094400,
      status: follower, online: yes
```

The test cluster has 4 nodes and allows a maximum of 4 replicas. With the replica 2, all data gets replicated twice on 2 different nodes of the cluster. If the nodes are located in two different racks or datacenters, it might happen the replica is made on two nodes in the same location. One can ensure the data gets distributed across different locations, by grouping the zones into nodes. The zone configuration is passed as the last argument to the already discussed `cluster --modify` option, which takes a new configuration of the cluster. The format of the zone entry is:

```
ZoneName1:UUID1,UUID2,...;ZoneName2:UUID3,UUID4,...
```

In the following example, we will group nodes 192.168.100.1 and 192.168.100.2 into zone "Rack1", and nodes 192.168.100.3 and 192.168.100.4 into "Rack2". To do this, we call `cluster --modify` with the current configuration and append the zone configuration after the node list:

```
$ sxadm cluster --modify 644245094400/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
     "Rack1:d3f8ad83-d003-4aaa-bbfb-73359af85991,abc2ed51-b4a8-46b6-a8ac-0beb58e697d2;
     Rack2:a343b7f9-0bef-4f03-8c6f-526ca12d75a9,b9b05fc7-7a4b-417d-853b-ac56ed32f5d3"
     @cluster2
```

After changing the zone configuration, the cluster needs to relocate the data according to the new distribution rules. With the new cluster configuration, the maximum replica is now 2 (the zone configuration would fail to apply, if the cluster contained volumes with replica higher than 2), and the data will be replicated across two zones. In a zone-enabled cluster, the maximum replica always equals to the number of zones plus the number of nodes, which are not part of any zone. When the data relocation is complete, the cluster will report the zone for each node:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Current configuration: 644245094400/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3 Rack1:d3f8ad83-d003-4
     aaa-bbfb-73359af85991,abc2ed51-b4a8-46b6-a8ac-0beb58e697d2;Rack2:a343b7f9-0bef-4f03-8
     c6f-526ca12d75a9,b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Operating mode: read-write
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.10) - checksum: 18024963750773516843
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 644245094400,
      zone: Rack1, status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      zone: Rack1, status: follower, online: yes
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 644245094400,
      zone: Rack2, status: leader, online: yes
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 644245094400,
      zone: Rack2, status: follower, online: yes
```

With this cluster configuration, volumes with replica 2 will be always fully replicated in both zones Rack1 and Rack2 (the data will be load-balanced across the nodes within each zone).

4.7 Cluster backup and restore

This section describes how to backup and restore the entire cluster.

Backup

The cluster backup can be automated with the `sxdump` tool, available from `https://pypi.python.org/pypi/sxdump/`. After installing `sxdump` (manually or automatically with `pip`), run the following command:

```
# sxdump --backup-dir /var/backups/sx/ sx://admin@mycluster
Generating sx-backup.sh and sx-restore.sh in the current directory
Review sx-backup.sh (uncomment/edit paths as necessary)
Run sx-backup.sh on the old cluster
Review sx-restore.sh (uncomment/edit paths as necessary)
Stop old cluster
Use sxsetup --config-file to setup the new cluster
Run sx-restore.sh on the new cluster
```

It creates two shell scripts in the current directory: `sx-backup.sh` and `sx-restore.sh`. The first script will backup all data from the cluster, while the other contains information on how to recreate the cluster structure, including all volumes, users, ACLs, and settings. Running `sx-backup.sh` will create a copy of all files in the cluster:

```
# ./sx-backup.sh

Backing up volume sx://admin@sxtest/vol1

Downloading /video.mkv (size: 1.22GB)
Transferred 1.22GB in 8s (@154.35MB/s)
[...]
```

When the script finishes, the data from the cluster will be backed up in `/var/backups/sx`. **No data from encrypted volumes will be backed up — those have to be processed manually.**

Restore

In order to restore the cluster, including all volumes, users, and ACLs run `sx-restore.sh` created by `sxdump` against a new cluster. You may need to edit the file in case the cluster name or location of the backup changed.

4.8 Cluster healing

It may happen one or more nodes are permanently lost due to external causes. When that happens, some operations might only be possible in read-only mode, until the broken nodes get replaced or removed from the cluster. S^{X} can automatically detect offline or broken nodes and automatically disable them. It uses the Raft algorithm, which achieves consensus via an elected leader. The leader checks if it has received heartbeats from all nodes in the cluster, and if some nodes didn't respond for a specified amount of time, they can be blacklisted. Due to the design of the Raft algorithm and the required majority, the automatic healing will only work for clusters with 3 or more nodes. For smaller clusters the operation has to be performed manually, as described below.

Configuring auto-healing

The following cluster settings are used to fine-tune the heartbeat and auto-healing process:

- **hb_keepalive**
  This option sets the interval between hearbeats (in seconds). The default value is 20 seconds.

- **hb_warntime**
  The hb_warntime option is used to specify how quickly heartbeat should issue a warning, that a node is unreachable. The default setting is 120 seconds.

- **hb_initdead**
  The option is used to set the time that it takes to declare a node dead when heartbeat is first started. It helps to avoid false reports in case a node or its operating system takes more time to start proper network operations. The default value for this option is 120 seconds.

- **hb_deadtime**
  This option sets the time, after which an unreachable node is considered dead and gets marked as broken. A broken node should be later replaced, as described below. This option is turned off by default.

You can obtain the current value of any setting by running the following command:

```
$ sxadm cluster --get-param=hb_keepalive @cluster2
hb_keepalive=20
```

To automatically mark broken nodes and disable them from regular cluster operations after 10 minutes of downtime, set the following options:

```
$ sxadm cluster --set-param="hb_initdead=600" @cluster2
$ sxadm cluster --set-param="hb_deadtime=600" @cluster2
hb_keepalive=20
```

Now, when the cluster detects a node, which didn't perform a valid heartbeat for more than 10 minutes, it will be automatically set as faulty and the cluster will allow write operations again. In the example below, the node 192.168.100.4 has been set as faulty:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Current configuration: 644245094400/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.9) - checksum: 18024963750773516843
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 644245094400,
      status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      status: follower, online: yes
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 644245094400,
      status: leader, online: yes
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.4, capacity: 644245094400,
      status: ** FAULTY **, online: ** NO **
```

The faulty nodes should be later replaced as described below.

Marking broken nodes manually

If the automatic detection and marking of broken nodes is not active (eg. when there's less than 3 nodes in the cluster and the Raft algorithm cannot be activated), the bad nodes can still be marked manually. Run the following command and provide the full specification of the broken node to mark it faulty:

```
$ sxadm cluster --set-faulty
     644245094400/192.168.100.4/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3 @cluster2
```

Cluster with faulty nodes, which have been marked, will properly operate in read-write mode, however no changes to the cluster structure will be allowed until the faulty nodes get properly replaced as described in the next subsection.

Replacing broken nodes

Skylable S^X provides an option to automatically rebuild a lost node and gather as much data as possible from other nodes. **Please never use this method against properly working nodes:** it assumes the node's data is permanently lost and can only retrieve missing data for volumes with replica higher than 1 — healthy nodes can be replaced using `--modify` option as described in the previous section. In the following example, we assume the node 192.168.100.4 is no longer available (just lost or already marked as faulty) and we will replace it with a new node 192.168.100.5. First we need to prepare a bare node 192.168.100.5 of the exact size as the broken node we are replacing, in this case it's 600GB:

```
-- on node 192.168.100.5 --
# sxsetup --bare
[...]
SX node started successfully
Bare node created. Use 'sxadm cluster --modify' to join it to the cluster
or perform another operation.
Node specification: 600G/192.168.100.5
```

Now we issue the following command, which uses the specification (size and UUID) of the broken node but points to the new IP address:

```
$ sxadm cluster --replace-faulty
    644245094400/192.168.100.5/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3 @cluster2
```

The broken node is immediately replaced with the new one, and the healing process is started:

```
$ sxadm cluster --info @cluster2
Cluster UUID: cc8ab859-619e-4806-ade6-c32ab2db1665
Operating mode: read-write
Current configuration: 644245094400/192.168.100.1/d3f8ad83-d003-4aaa-bbfb-73359af85991
     644245094400/192.168.100.2/abc2ed51-b4a8-46b6-a8ac-0beb58e697d2
     644245094400/192.168.100.3/a343b7f9-0bef-4f03-8c6f-526ca12d75a9
     644245094400/192.168.100.5/b9b05fc7-7a4b-417d-853b-ac56ed32f5d3
Distribution: 872eeecb-ebf9-4368-8150-beb23cd44edf(v.10) - checksum: 18024963750773516843
State of nodes:
  * node d3f8ad83-d003-4aaa-bbfb-73359af85991: addr: 192.168.100.1, capacity: 644245094400,
      status: follower, online: yes
  * node abc2ed51-b4a8-46b6-a8ac-0beb58e697d2: addr: 192.168.100.2, capacity: 644245094400,
      status: follower, online: yes
  * node a343b7f9-0bef-4f03-8c6f-526ca12d75a9: addr: 192.168.100.3, capacity: 644245094400,
      status: leader, online: yes
  * node b9b05fc7-7a4b-417d-853b-ac56ed32f5d3: addr: 192.168.100.5, capacity: 644245094400,
      status: follower, online: yes, activity: Healing blocks
```

During the repair process client operations should be back to normal. The same steps can be used to replace a broken node without changing its IP address, in that case the bare node must be prepared and available with the IP address of the broken one. It is also possible to repair more than one node at a time by passing more node specifications to `--replace-faulty`.

Setting read-only mode

The cluster can be set into read-only mode to perform maintainance or temporarily stop clients from uploading new data by running the following command:

```
$ sxadm cluster --set-mode=ro @cluster2
Successfully switched cluster to read-only mode
```

To switch the cluster back to read-write mode, run the same command with `rw` argument:

```
$ sxadm cluster --set-mode=rw @cluster2
Successfully switched cluster to read-write mode
```

Checking storage integrity

S^X provides a tool that performs a deep check of the storage structure and verifies if the data on disk, stored in a special format called *HashFS*, is not corrupted. The tool will calculate and compare checksums of all data blocks and report any inconsistencies. In order to perform this check, the cluster needs to be set to read-only mode or a particular node needs to be temporarily turned off (with `sxserver stop`). When the cluster is in read-only mode, you can perform the check on all nodes at the same time. Run the following command on the node you want to check (run `sxsetup --info` if you don't remember the location of the storage):

```
# sxadm node --check /var/lib/sxserver/storage/
[sx_hashfs_check]: Integrity check started
HashFS is clean, no errors found
```

Compacting node data

The local storage of S^X consists of special databases and data files, which store the information about files and cluster configuration as well as the actual blocks of data. When a file gets deleted, all blocks belonging to that file, which are not shared by other files are marked as free and can be reused. S^X will only allocate more space on disk, if it cannot reuse existing blocks within data files. It will, however, not return the free blocks to the system automatically. If there's a need to free some disk space, the local storage can be easily compacted:

```
# sxsetup --compact
The node will be stopped now and started again when compacting is finished
Sending SIGTERM to 28310
Sending SIGTERM to 28320
Waiting for  28310 28320
Operation complete (disk space freed: 1.95G)
Starting SX.fcgi
Starting sxhttpd
SX node started successfully
```

In case compacting doesn't free any disk space, force the garbage collector to mark blocks of recently removed files as free and then try again. The garbage collector is run with the following command:

```
$ sxadm cluster --force-gc @cluster2
```

Data recovery

It is possible to recover local data in case a node gets damaged. Please perform the following command and `sxadm` will try to extract as much data as possible from the local storage:

```
# sxadm node --extract=/tmp/RECOVERED /var/lib/sxserver/storage/
Finished data extraction from node /var/lib/sxserver/storage/
```

CHAPTER 5

Client operations

5.1 Access profiles

Using `sxinit` one can configure access for multiple users and clusters. The access profiles have the format of `sx://[username@]cluster_name`. When the username is omitted, `sxinit` will ask for it and `sx://cluster_name` will be the default profile for a given cluster.

Adding profiles

To add an access profile for the user ‘jeff’ and the local cluster created in previous chapters run the following command:

```
$ sxinit -l 192.168.1.101 -A @jeff sx://jeff@mycluster
Warning: self-signed certificate:

        Subject: C=GB, ST=UK, O=SX, CN=mycluster
        Issuer: C=GB, ST=UK, O=SX, CN=mycluster
        SHA1 Fingerprint: 84:EF:39:80:1E:28:9C:4A:C8:80:E6:56:57:A4:CD:64:2E:23:99:7A

Do you trust this SSL certificate? [y/N] y
Trusting self-signed certificate
Please enter the user key: FqmlTd9CWZUuPBGMdjE46DaT1/3kx+EYbahlrhcdVpy/9ePfrtWCIgAA
```

Since “mycluster” is not a DNS name, we had to point `sxinit` to one of the nodes of the cluster. That allowed it to connect and discover all the other nodes. We also created the alias `@jeff`, which will be used for convenience.

Listing access profiles

To list all configured access profiles run:

```
$ sxinit --list
sx://jeff@mycluster                         @jeff
sx://admin@mycluster                        @cluster
```

Deleting profiles

To delete a profile run the following command and provide the full profile name or its alias as follows:

```
$ sxinit --delete @somealias
```

It will delete the alias `@somealias` and the profile associated with it.

5.2 Working with files

S^X provides easy to use file tools, which resemble typical UNIX commands. Since S^X is an object storage and not a filesystem, there are some fundamental differences, though. One of them is lack of “real” directories: each file (object) has assigned a full path that uniquely identifies it and the path is not a part of any tree structure. S^X does simulate a directory structure by matching the subpaths, for example `/path/file1` and `/path/file2` will be presented as contents of the directory `/path/` just like on a typical filesystem. However, the directory `/path/` is only emulated (and is not assigned to any object), and therefore it’s perfectly legit to also have a file with a path `/path`, which doesn’t conflict with the other two files at all!

In the following subsections we present the command line tools and show how to use them to perform common tasks.

sxcp: upload and download files

`sxcp` can copy files and entire directories from and to Skylable S^X clusters. It can also copy data between two different SX clusters. By default, for each file a progress bar is displayed, which shows the transfer speed and the estimated time of arrival. `sxcp` makes use of all the advanced features S^X, such as deduplication and transfer resuming to minimize the bandwidth usage.

Use `sxcp -r` to recursively upload directories to the remote volume:

```
$ sxcp -r /home/jeff/VMs/ @jeff/vol-jeff/VMimages/
Uploading /home/jeff/VMs/FreeBSD 10.0/FreeBSD 10.0.vmdk (size: 4.91GB)
  14% [====>                                            ]  55.11MB/s ETA 44s
```

`sxcp` shows the average speed of the tranfer and how long it will take. The great feature of S^X is the already mentioned transfer resuming, which allows to continue the transfer in case it was interrupted. Below we interrupt the transfer of the large file and repeat the same copy again:

```
$ sxcp -r /home/jeff/VMs/ @jeff/vol-jeff/VMimages/
Uploading /home/jeff/VMs/FreeBSD 10.0/FreeBSD 10.0.vmdk (size: 4.91GB)
  94% [=============================================>   ]  55.11MB/s ETA 5s
^CProcess interrupted
$ sxcp -r /home/jeff/VMs/ @jeff/vol-jeff/VMimages/
  97% [++++++++++++++++++++++++++++++++++++++++++++++=> ]  52.17MB/s ETA 2s
```

The second `sxcp` call automatically finds out, which blocks of the file have been already transferred and only uploads the missing ones. The transfer resuming works in a similar way for file downloads.

`sxcp` can copy files between different volumes, also on different clusters, and comes with other useful features, such as bandwidth limiting. See `man sxcp` for the usage details and other examples.

sxls: list volumes and files

With `sxls` one can discover, which volumes are accessible on the cluster and then list their contents. To get the list of volumes, which user ’jeff’ can access run:

```
$ sxls -lH @jeff
  VOL  rep:2  rev:1  rw  -  12.83G  50.00G jeff 25% sx://jeff@mycluster/vol-jeff
```

With `-l (--long)` and `-H (--human-readable)` options `sxls` displays the list of available volumes, together with additional information such as the replica count, maximum number of revisions per file, permissions, size, usage, and the owner name.

Running `sxls` against the volume without any arguments returns the first level of files, similarly to the command `ls`:

```
$ sxls @jeff/vol-jeff
sx://jeff@mycluster/vol-jeff/VMimages/
```

To list the volume recursively, with more information about files and human readable sizes run:

```
$ sxls -rlH @jeff/vol-jeff
2014-11-17 14:03           31 sx://jeff@mycluster/vol-jeff/VMimages/Debian-MIPS/bridge.sh
2014-11-17 14:03      245.88M sx://jeff@mycluster/vol-jeff/VMimages/Debian-MIPS/
     debian_squeeze_mips_standard.qcow2
2014-11-17 14:03        4.10M sx://jeff@mycluster/vol-jeff/VMimages/Debian-MIPS/initrd.gz
2014-11-17 14:03          139 sx://jeff@mycluster/vol-jeff/VMimages/Debian-MIPS/run
2014-11-17 14:03          677 sx://jeff@mycluster/vol-jeff/VMimages/Debian-MIPS/start.sh
2014-11-17 14:03        6.61M sx://jeff@mycluster/vol-jeff/VMimages/Debian-MIPS/vmlinux
     -2.6.32-5-4kc-malta
2014-11-17 14:03        1.41G sx://jeff@mycluster/vol-jeff/VMimages/Debian-PPC/
     debian_squeeze_powerpc_standard.qcow2
2014-11-17 14:04          349 sx://jeff@mycluster/vol-jeff/VMimages/Debian-PPC/start.sh
2014-11-17 14:02        4.91G sx://jeff@mycluster/vol-jeff/VMimages/FreeBSD 10.0/FreeBSD
     10.0.vmdk
2014-11-17 14:03      693.12M sx://jeff@mycluster/vol-jeff/VMimages/FreeBSD 10.0/FreeBSD
     -10.0-BETA1-amd64-disc1.iso
```

SXMV: MOVE OR RENAME FILES

`sxmv` can move files or group of files into new locations. It can be used to just rename individual files or move entire groups to another cluster. In contrast to the command `mv`, renaming a directory with `sxmv` requires prividing the recursive flag `-r`. That's because of the design of the object storage and lack of real directories as described at the beginning of this chapter. In order to rename a directory, `sxmv` has to rename all the files (objects), which share the same directory path. In the example below we rename the directory 'VMimages' to 'VMs' and list the new volume structure in basic mode:

```
$ sxmv -r @jeff/vol-jeff/VMimages/ @jeff/vol-jeff/VMs/
$ sxls -r @jeff/vol-jeff
sx://jeff@mycluster/vol-jeff/VMs/Debian-MIPS/bridge.sh
sx://jeff@mycluster/vol-jeff/VMs/Debian-MIPS/debian_squeeze_mips_standard.qcow2
sx://jeff@mycluster/vol-jeff/VMs/Debian-MIPS/initrd.gz
sx://jeff@mycluster/vol-jeff/VMs/Debian-MIPS/run
sx://jeff@mycluster/vol-jeff/VMs/Debian-MIPS/start.sh
sx://jeff@mycluster/vol-jeff/VMs/Debian-MIPS/vmlinux-2.6.32-5-4kc-malta
sx://jeff@mycluster/vol-jeff/VMs/Debian-PPC/debian_squeeze_powerpc_standard.qcow2
sx://jeff@mycluster/vol-jeff/VMs/Debian-PPC/start.sh
sx://jeff@mycluster/vol-jeff/VMs/FreeBSD 10.0/FreeBSD 10.0.vmdk
sx://jeff@mycluster/vol-jeff/VMs/FreeBSD 10.0/FreeBSD-10.0-BETA1-amd64-disc1.iso
```

SXRM: REMOVE FILES

The equivalent of the system command `rm` in S^X is `sxrm`. Similarly to other tools, it can handle individual files or entire directories in recursive mode. Below we first check the current space usage for the volume, then remove a directory with some large files (using a wildcard to match it), and check the usage again:

```
$ sxls -l @jeff
  VOL  rep:2  rev:1  rw  -  7.24G   50.00G  14% sx://jeff@mycluster/vol-jeff
$ sxrm -r @jeff/vol-jeff/VMs/FreeBSD*
Deleted 2 file(s)
$ sxls -lH @jeff
  VOL  rep:2  rev:1  rw  -  1.66G   50.00G  14% sx://jeff@mycluster/vol-jeff
```

SXREV: MANAGE FILE REVISIONS

The S^{X} volumes can be configured[1] to keep multiple revisions of files. For example, if a volume was created with an option to keep 3 revisions, every time a specific file gets modified the previous copy will be preserved and the latest 3 versions of the file will be available for download. A revision is only created when the new file is different from the existing one. The tools such as `sxcp` or `sxls` will always operate on the latest revision. In order to access and manage the older revisions, one has to use `sxrev`.

In the examples below we will operate on the volume `vol-jeff-rev`, which was configured to store up to 3 revisions for each file and the example file `document.pdf` was already updated a few times. In order to list all of its revisions, run the following command:

```
$ sxrev list @jeff/vol-jeff-rev/document.pdf
Revisions for file @jeff/vol-jeff-rev/document.pdf (most recent first):
1.      2014-11-18 12:05 size:128026 rev:"2014-11-18 12:05:00.938:
     d2bc1190a0f70f4b4925d702e0d567a7"
2.      2014-11-18 11:54 size:105866 rev:"2014-11-18 11:54:42.362:1
     fc102f66cabd0e8daac8e1279b54c0a"
3.      2014-11-18 10:23 size:93545  rev:"2014-11-18 10:23:22.188:
     d3b1fb1d7e4219ab4a8d1fc7c8edff0c"
```

The first revision on the list is the latest one, which is also visible to other tools. In order to restore an older revision of a file, it needs to be copied into a new destination. By default `sxrev` asks interactively, which revision should be copied as on the example below:

```
$ sxrev copy @jeff/vol-jeff-rev/document.pdf ~/document-prev.pdf
Revisions for file @jeff/vol-jeff-rev/document.pdf (most recent first):
1.      2014-11-18 12:05 size:128026 rev:"2014-11-18 12:05:00.938:
     d2bc1190a0f70f4b4925d702e0d567a7"
2.      2014-11-18 11:54 size:105866 rev:"2014-11-18 11:54:42.362:1
     fc102f66cabd0e8daac8e1279b54c0a"
3.      2014-11-18 10:23 size:93545  rev:"2014-11-18 10:23:22.188:
     d3b1fb1d7e4219ab4a8d1fc7c8edff0c"
Choose revision to copy: 2
Copy operation completed successfully
```

The same operation can be performed in non-interactive mode by providing the revision string as an argument:

```
$ sxrev copy -r "2014-11-18 11:54:42.362:1fc102f66cabd0e8daac8e1279b54c0a" @jeff/vol-jeff-
     rev/document.pdf ~/document-prev.pdf
Copy operation completed successfully
```

The size of all revisions adds up to the volume usage, that's why one may want to remove specific revisions (eg. for large media files). When a file with multiple revisions gets deleted with `sxrm`, all of the revisions get removed automatically as well. With `sxrev delete` only specific revisions can be deleted and it works similarly to `sxrev copy`. In the example below we remove the two oldest revisions, in both interactive and non-interactive modes:

```
$ sxrev delete @jeff/vol-jeff-rev/document.pdf
Revisions for file @jeff/vol-jeff-rev/document.pdf (most recent first):
1.      2014-11-18 12:05 size:128026 rev:"2014-11-18 12:05:00.938:
     d2bc1190a0f70f4b4925d702e0d567a7"
2.      2014-11-18 11:54 size:105866 rev:"2014-11-18 11:54:42.362:1
     fc102f66cabd0e8daac8e1279b54c0a"
3.      2014-11-18 10:23 size:93545  rev:"2014-11-18 10:23:22.188:
     d3b1fb1d7e4219ab4a8d1fc7c8edff0c"
```

[1] See section 4.4 on page 18 on how to create and configure volumes.

```
Choose revision to delete: 2
Delete operation completed successfully
$ sxrev delete -r "2014-11-18 10:23:22.188:d3b1fb1d7e4219ab4a8d1fc7c8edff0c" @jeff/vol-jeff-
     rev/document.pdf
Delete operation completed successfully
$ sxrev list @jeff/vol-jeff-rev/document.pdf
Revisions for file @jeff/vol-jeff-rev/document.pdf (most recent first):
1.      2014-11-18 12:05 size:128026 rev:"2014-11-18 12:05:00.938:
     d2bc1190a0f70f4b4925d702e0d567a7"
```

The volume owner can also change the maximum number of revisions kept for files by issuing the following command:

```
$ sxvol modify --max-revisions=5 @jeff/vol-jeff-rev
Volume revisions limit changed to 5
```

5.3 Mounting remote volumes

For systems that support the FUSE library, S^X provides a tool called `sxfs`, which allows mounting remote volumes as local filesystems. Run the following command to mount the remote volume with default settings:

```
$ mkdir ~/vol-jeff
$ sxfs @jeff/vol-jeff ~/vol-jeff
Using default tempdir: /var/tmp/sxfs-20151203162702-GYDfg4
```

By default `sxfs` will wait for each operation and report the result to the application performing the action (eg. `cp`). When the option `--use-queues` is enabled, write and delete operations will be queued and performed in the background. This improves the interaction with the mounted volume, however errors might not be reported back to the application. See the `sxfs(1)` manpage for information about all available options. Depending on the OS, run `fusermount -u` or `umount` on the mount point to unmount the filesystem.

CHAPTER 6

ADVANCED

This chapter describes some advanced details of the S^X design.

6.1 DATA DISTRIBUTION

Files are divided into blocks of equal size, which depend on the file's size, and distributed among S^X nodes using a consistent hashing algorithm. This ensures that the storage among all nodes in the cluster is properly balanced, and that when nodes are added or removed from the cluster only a minimal amount of data gets moved.

The cluster exposes an HTTP(S) REST API designed around deduplicated storage: equal blocks of data get stored only once. This has the advantage of reduced bandwidth usage, better resume handling, and that the data is immutable (only the metadata is mutable).

In the example shown in figure 6.1 there is a file divided into 10 blocks, where the first and last two blocks are equal — only the first block of them is uploaded. The rest

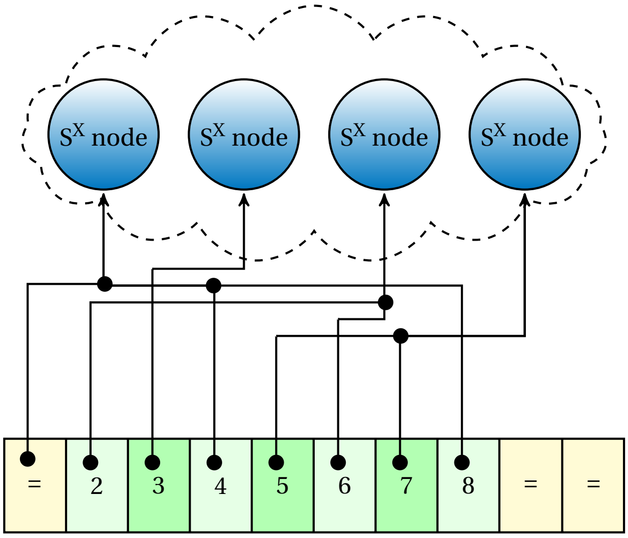


Figure 6.1: Data distribution

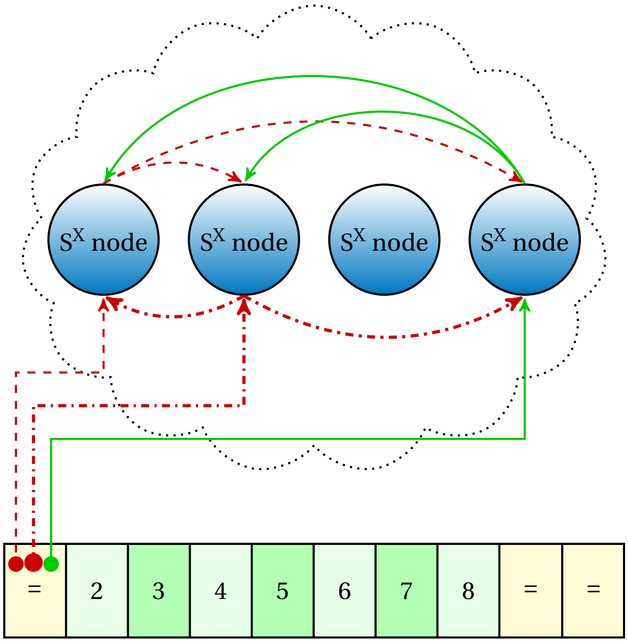


Figure 6.2: Data replication

of the blocks are distributed among all nodes in the cluster. Inevitably some nodes will receive more than one block, in which case the protocol supports an efficient batched upload and download mode.

In this example each block of data was stored only once, what is called *replica 1*. If one of the nodes that stores a block of the file goes down, the file will be unreachable. To make the data more resistant one should use a replica count higher than 1, which means that the data will be duplicated on multiple nodes. A volume with replica 2 can survive a loss of at most 1 node, with replica 3 the loss of at most 2 nodes, etc. The maximum replica count is limited by the number of nodes. High replica counts increase reliability for downloads, at the cost of increased latency on uploads, and lower fault-tolerance on uploads (all replica nodes must be up for uploads to succeed).

The handling of replicas is illustrated in figure 6.2 for a replica 3 volume. The client tries to upload the data to a given node, which is then responsible for replicating the data inside the cluster asynchronously. If the client fails to upload to a specific node it can retry on the next one, and so on. Each time the receiving node will replicate the data inside the cluster.

6.2 GLOBAL OBJECTS

Users, volumes and privileges are stored globally — on each node in the cluster — and changing them requires cooperation of all nodes.

user each user is issued an authentication token. All requests are signed using HMAC and the authentication token.

node	volA_r1		volB_r2		volC_r3		users		
	ACL	files	ACL	files	ACL	files	admin	u1	u2
node1	✓	✓	✓	✗	✓	✗	✓	✓	✓
node2	✓	✗	✓	✓	✓	✓	✓	✓	✓
node3	✓	✗	✓	✓	✓	✓	✓	✓	✓
node4	✓	✗	✓	✗	✓	✓	✓	✓	✓

Table 6.1: Global objects

admin users the privileged users can perform all administrative tasks such as volume and user management

volumes used to group several files, owned by a specific user. Each volume has a replica count and metadata associated with it.

volume ACL the volume owner by default has full access to the volume. The owner (and the cluster admin) can grant and revoke permissions for other users[1].

As shown in Table 6.1 the volume names, privileges and users are stored on all nodes. However the volume's contents are stored only on a subset of nodes:

volA_r1 is a volume with replica 1: its data and filenames are only stored in one place (no copies)

volB_r2 is a volume with replica 2: the data is always stored on (at least) 2 distinct nodes, and its filenames are stored on exactly 2 specific distinct nodes

volC_r3 is a volume with replica 3: the data is always stored on (at least) 3 distinct nodes, and its filenames are stored on exactly 3 specific distinct nodes

and so on...

The cluster *can* have multiple volumes with different replica counts at the same time. A volume can also have an arbitrary metadata attached to.

6.3 Jobs

Certain operations, such as finalizing a file upload (which may require replication of data), can take a long time. To avoid blocking other operations, all taks which involve more than one node create a job and the S^X clients poll for the job's outcome instead of blocking and waiting for it to finish. This allows to speed up recursive uploads for example: each file creates a new job and the client only waits for completion at the end of the recursive upload. The cluster tries its best to retry when transient errors occur internally. In case of a failure, it will abort or undo the operation and report the status to the client. The jobs are also used for conflict resolution: on conflicting operations, for example creating two users with same name, only one job is guaranteed to "win" and all the others will be aborted.

[1] It's impossible to revoke privileges for admin users, they always have full access to all volumes.

CHAPTER 7

Troubleshooting

7.1 Frequently Asked Questions

If you face an issue, please have a look into our FAQ database at http://www.skylable.com/docs/faq. It's constantly updated to provide solutions and answers to frequent questions from our users.

7.2 Mailing list

In case you cannot find a solution to your problem, please subscribe to our mailing list sx-users at http://lists.skylable.com and post your question there.

7.3 Bug reporting

If you believe you've found a bug in S^{X} please enter it into our Bugzilla tracker at https://bugzilla.skylable.com. Please try to provide as much details as possible. If you report a problem with the client, you can generate a report on the system and client configuration by running:

```
$ sxreport-client
Report stored in sxreport-client-1418047578.log
You can attach it to a bugreport at https://bugzilla.skylable.com
```

When reporting a problem with one of the server components, you can use sxreport-server instead:

```
# sxreport-server
Anonymized report stored in sxreport-server-1418047910-anon.log
You can attach it to a bugreport at https://bugzilla.skylable.com
```

It creates a report with all IP addresses, URLs, and usernames anonymized by default.