By Falko Timme
Published: 2007-01-25 14:33

Back Up Linux And Windows Systems With BackupPC

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 01/19/2007

This tutorial shows how you can back up Linux and Windows systems with BackupPC. BackupPC acts as a server and is installed on a Linux system, and from there it can connect to all Linux and Windows systems in your local network to back them up and restore them without interfering with the user's work on that system. On the clients minimal to no configuration is needed. BackupPC supports full and incremental backups, and it comes with a neat web frontend for the administrator and normal user so that backups and recoveries can be managed through a web browser. It should be noted, however, that BackupPC does file-based backups, not bit-wise backups like Ghost4Linux, for example, so it is not made for disk/partition imaging.

BackupPC comes with a clever pooling scheme that minimizes disk storage and disk I/O. Identical files across multiple backups of the same or different PCs are stored only once resulting in substantial savings in disk storage and disk I/O.

I want to say first that this is not the only way of setting up such a system. There are many ways of achieving this goal but this is the way I take. I do not issue any guarantee that this will work for you!

1 Preliminary Note

In this tutorial I will set up BackupPC on a Debian Etch server with the static IP address *192.168.0.100* and the hostname *server1.example.com*. The procedure should be similar or the same on Debian Sarge and Ubuntu.

BackupPC stores the backups in */var/lib/backuppc*, so I've made that an extra partition of about 90 GB on the BackupPC server. It should be noted that this is still very small, especially if you want to back up lots of machines, so bigger is better. Also, you should consider using **LVM** for the */var/lib/backuppc* partition so that you can enlarge it whenever you need it. You can learn more about LVM in this tutorial:
http://www.howtoforge.com/linux_lvm

I will show how to back up and restore a Ubuntu Edgy Eft desktop system with the hostname/NetBIOS name *falko-desktop* and a Windows XP laptop with the NetBIOS name *notebook*. Both get their network settings through DHCP.

BackupPC supports three transfer methods, *smb*, *rsync/rsyncd*, and *tar*. *rsync* is the recommended method for Linux systems, so I will use that for the Ubuntu desktop, and *smb* is the preferred method for Windows systems, so I use it for the Windows laptop.

2 Install BackupPC

On *server1.example.com*, we run this command as root to install BackupPC:

```
apt-get install backuppc rsync libfile-rsyncp-perl par2 smbfs
```

You willl be asked a few questions:

```
Workgroup/Domain Name:
 Modify smb.conf to use WINS settings from DHCP? <-- No
```

Then you will see this message:

```
Web administration default user created

BackupPC can be managed through its web interface:
  http://server1.example.com/backuppc/


For that purpose, a web user named 'backuppc' with 'dAamcQIH' as password has been created. You can change this password by
running 'htpasswd /etc/backuppc/htpasswd backuppc'.
```

Please make a note of the web frontend password (*dAamcQIH* in this case) for the user *backuppc*. If you don't like the password, you can change it by running


```
htpasswd /etc/backuppc/htpasswd backuppc
```

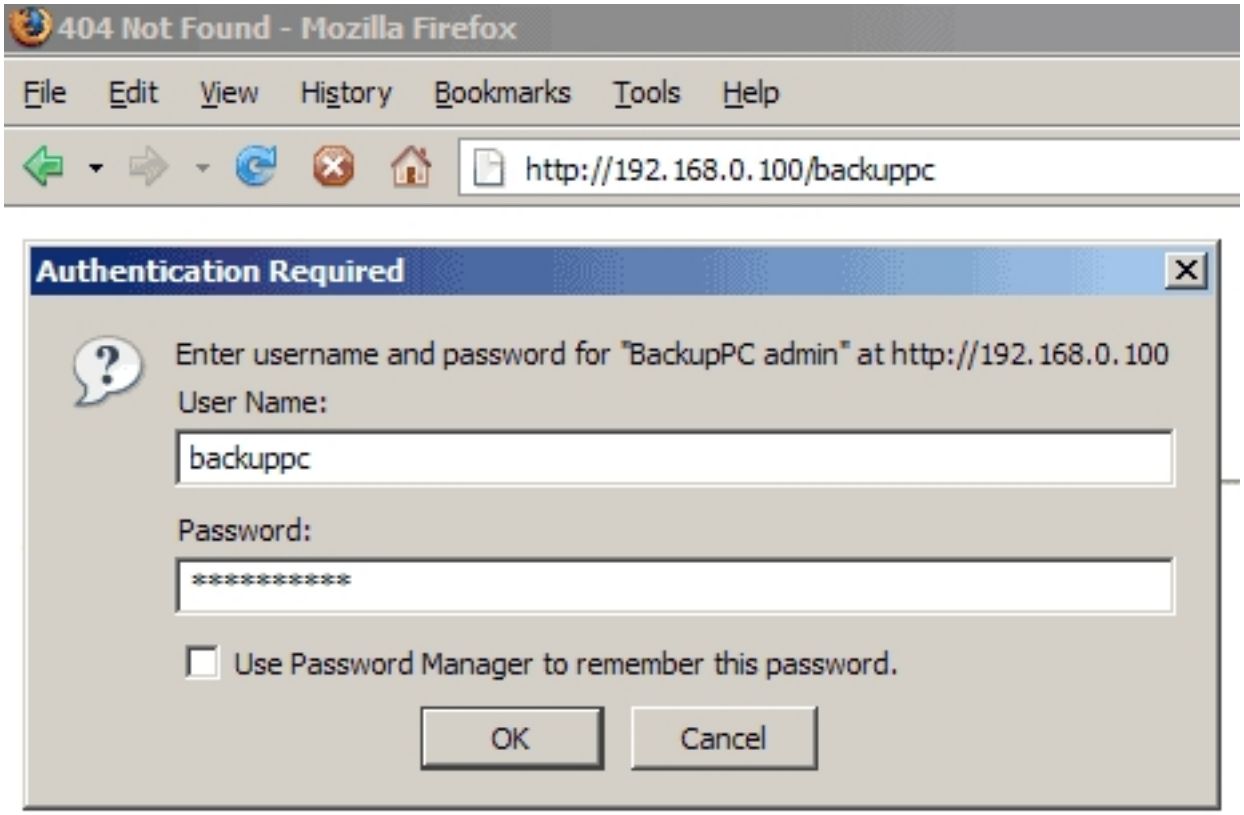
Then open a browser and type in *http://192.168.0.100/backuppc/* (or *http://server1.example.com/backuppc/* if you have a proper DNS record for *server1.example.com* in your network). If you get a login prompt, everything is fine, if not (like on my installation), run this command:

```
dpkg-reconfigure backuppc
```

You will see this question:

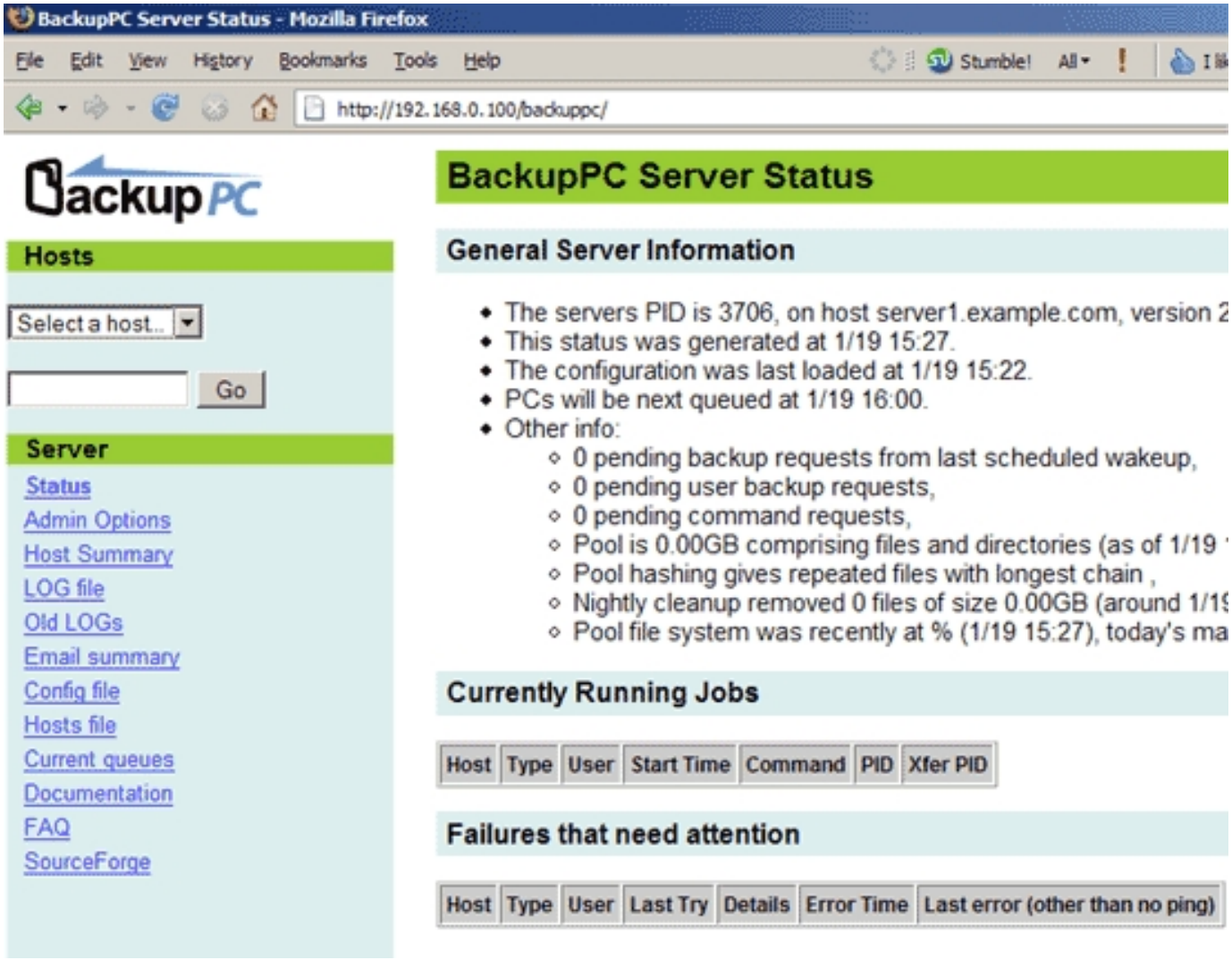
Add aliases for /backuppc/ to your apache config files ?

Afterwards, go to *http://192.168.0.100/backuppc/* again and log in with the username *backuppc* and *backuppc*'s password:

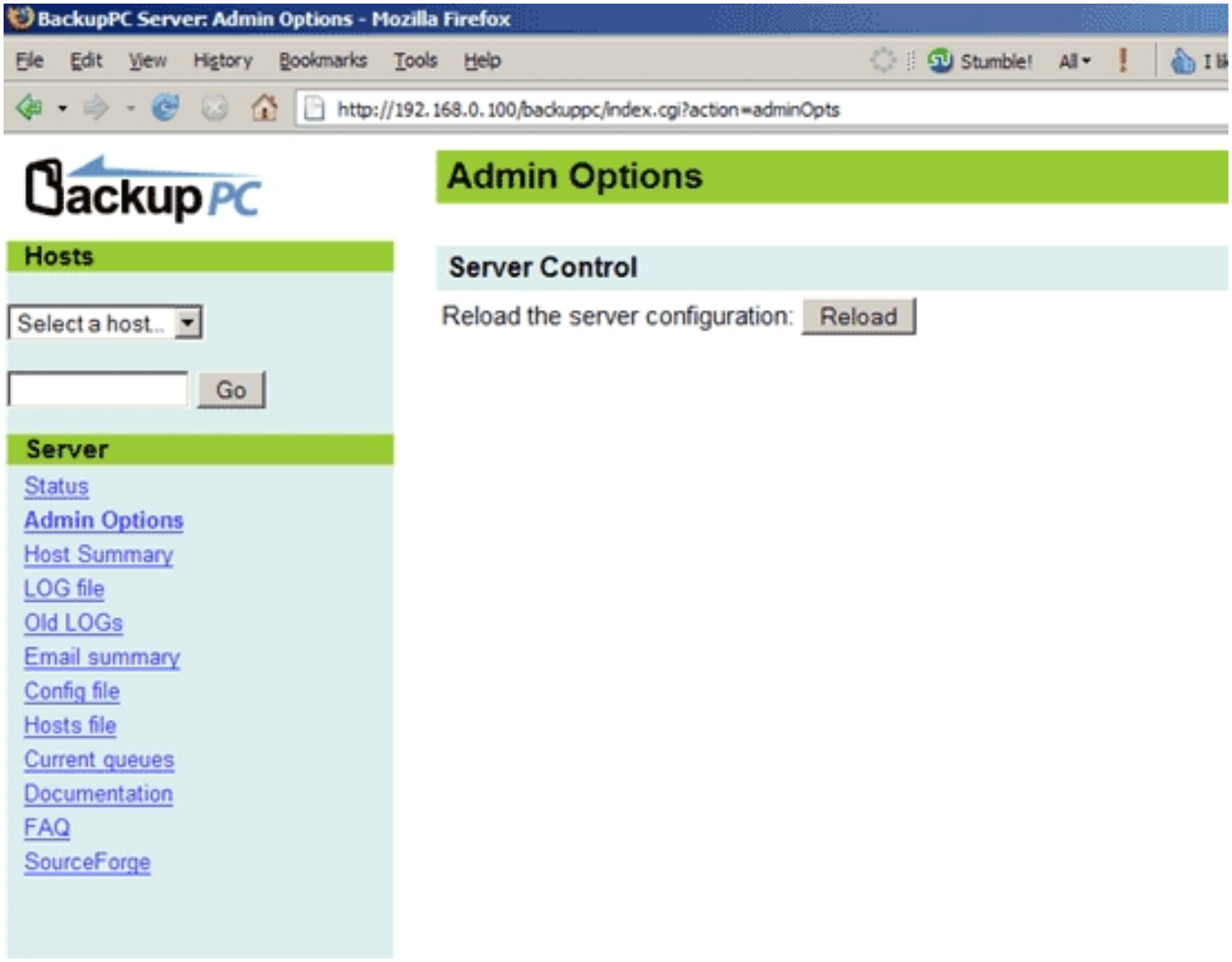


3 The BackupPC Web Interface

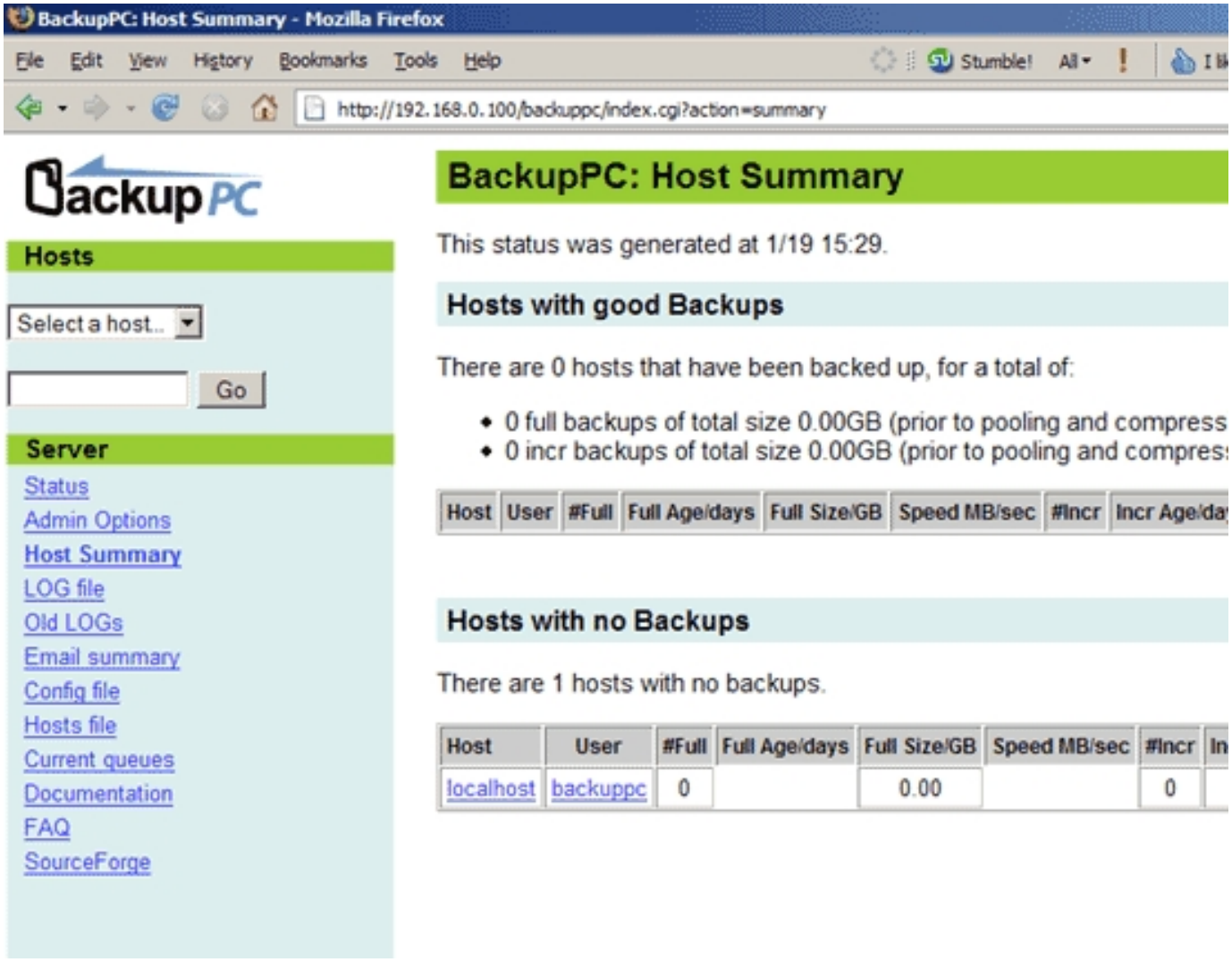
Now that you are logged in to BackupPC's web interface, you should browse the menu to become familiar with it. The entry page is the `Status` page:



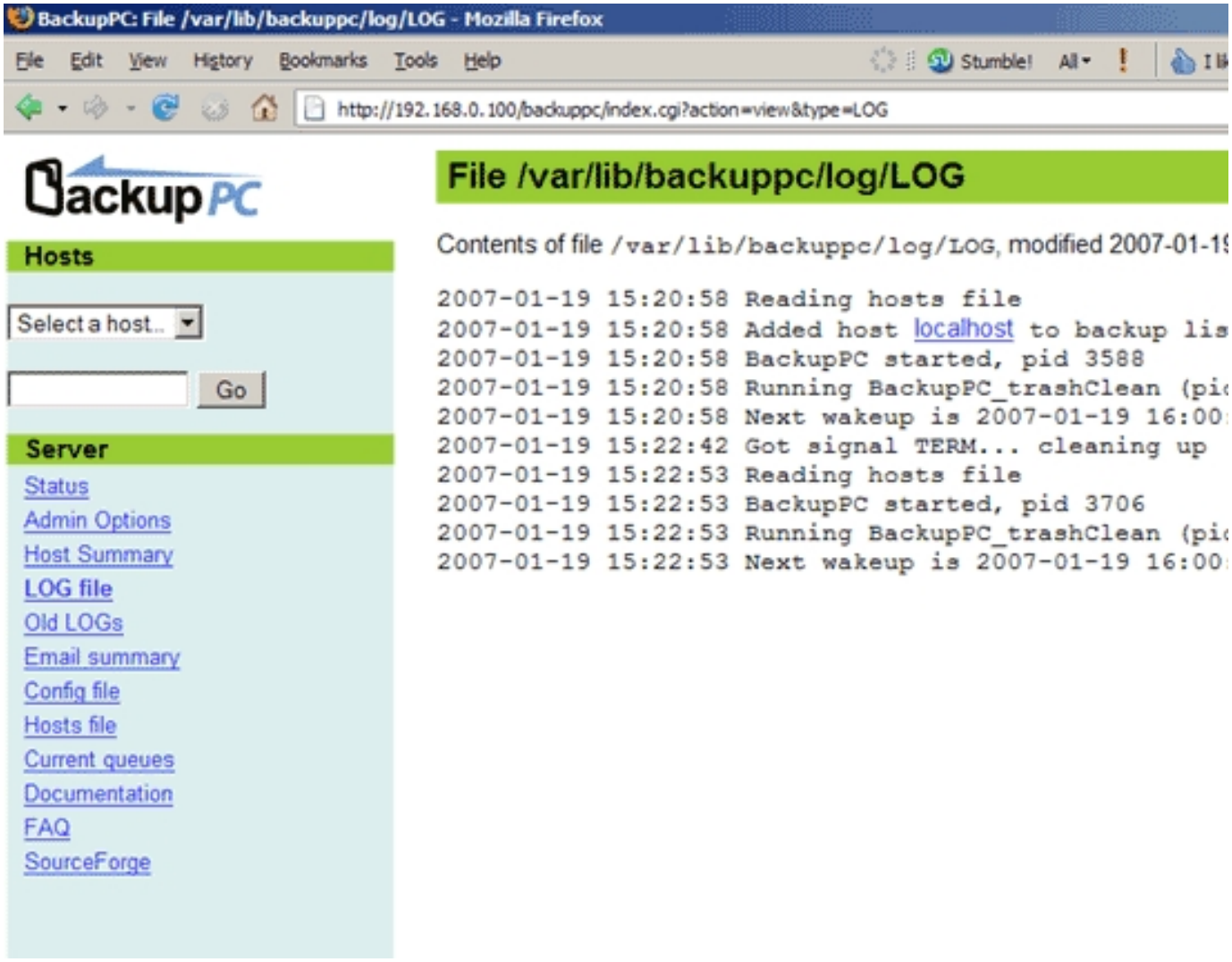
`Admin Options` allows you to reload the server configuration:



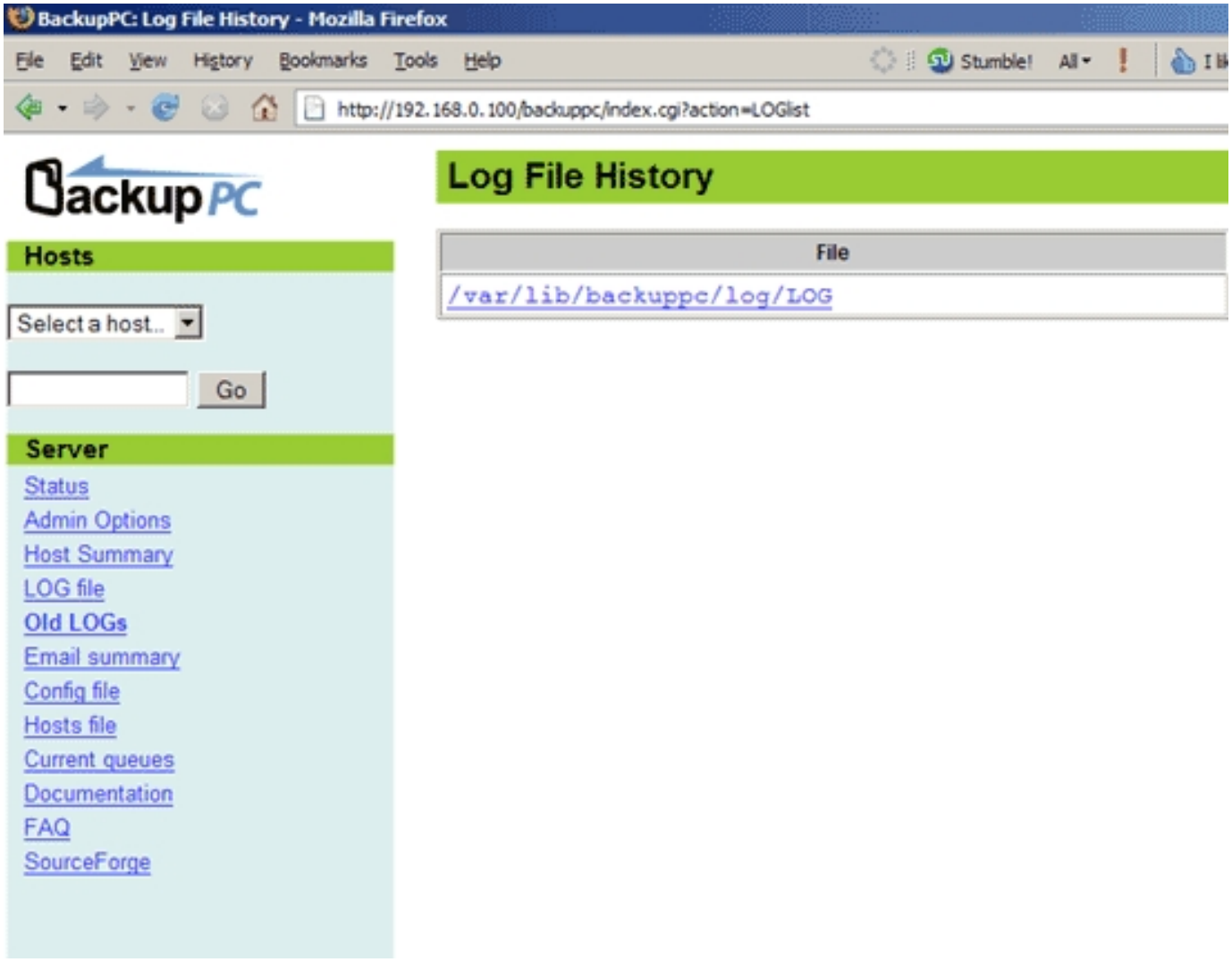
`Host Summary` shows a summary of all configured clients and their backup status. Currently only `localhost` is configured as a client for backups. We will remove `localhost` later on because we don't want to back it up.



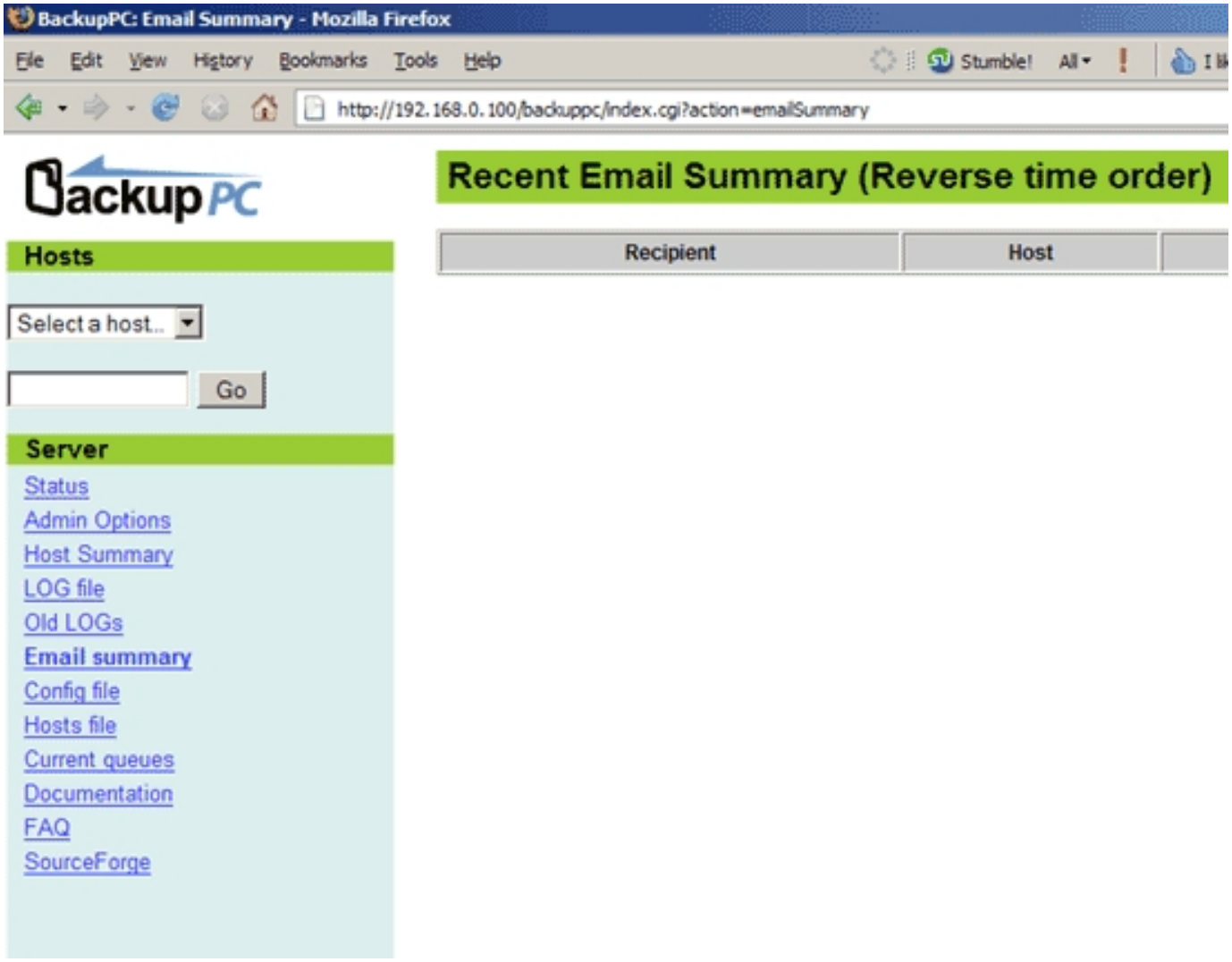
Under *LOG file* you can find the latest log entries:



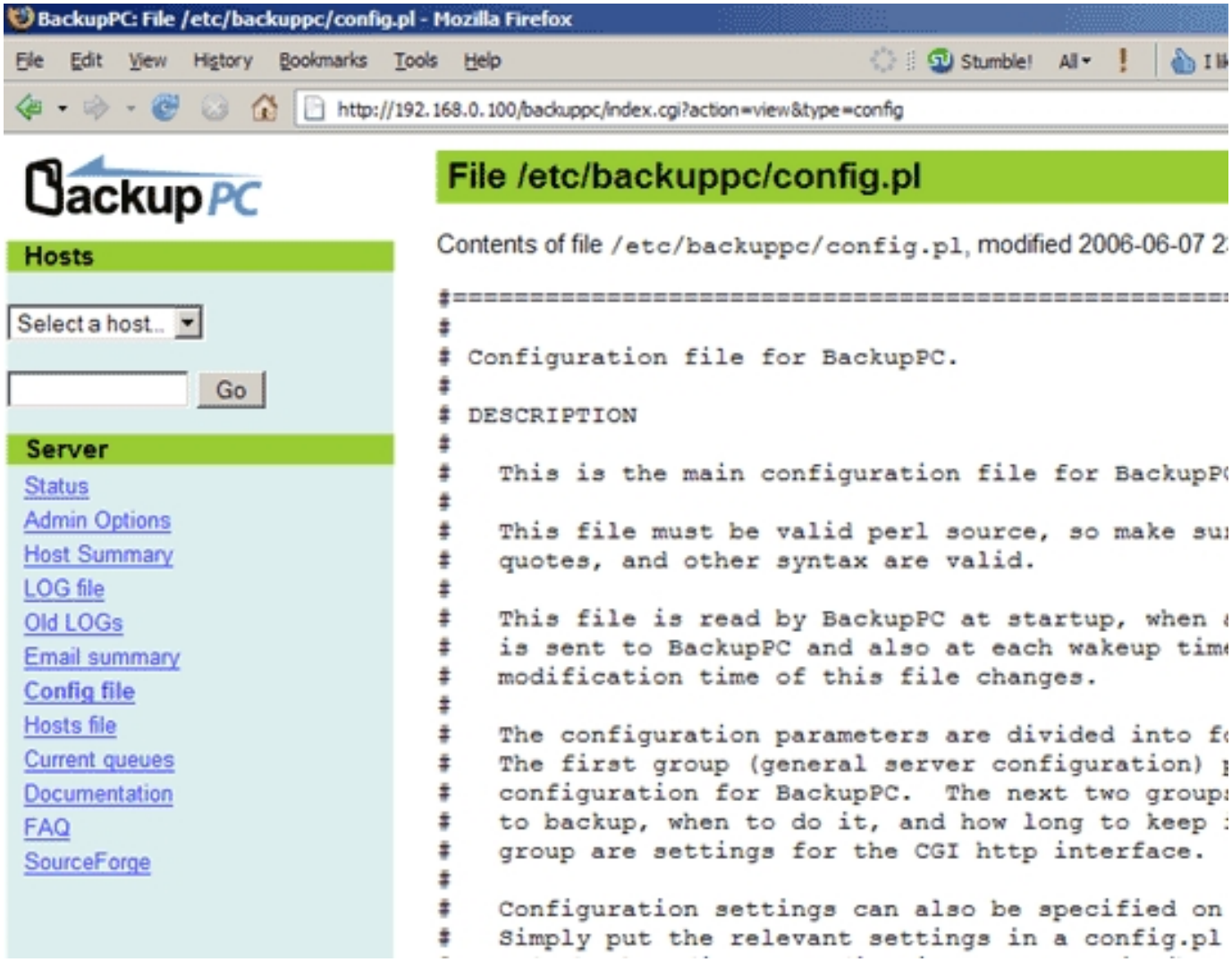
`Old LOGs` shows which log files are available. BackupPC has a built-in log rotation mechanism, so there will be more than one log file over time. The max. amount of log files can be specified in */etc/backuppc/config.pl*.



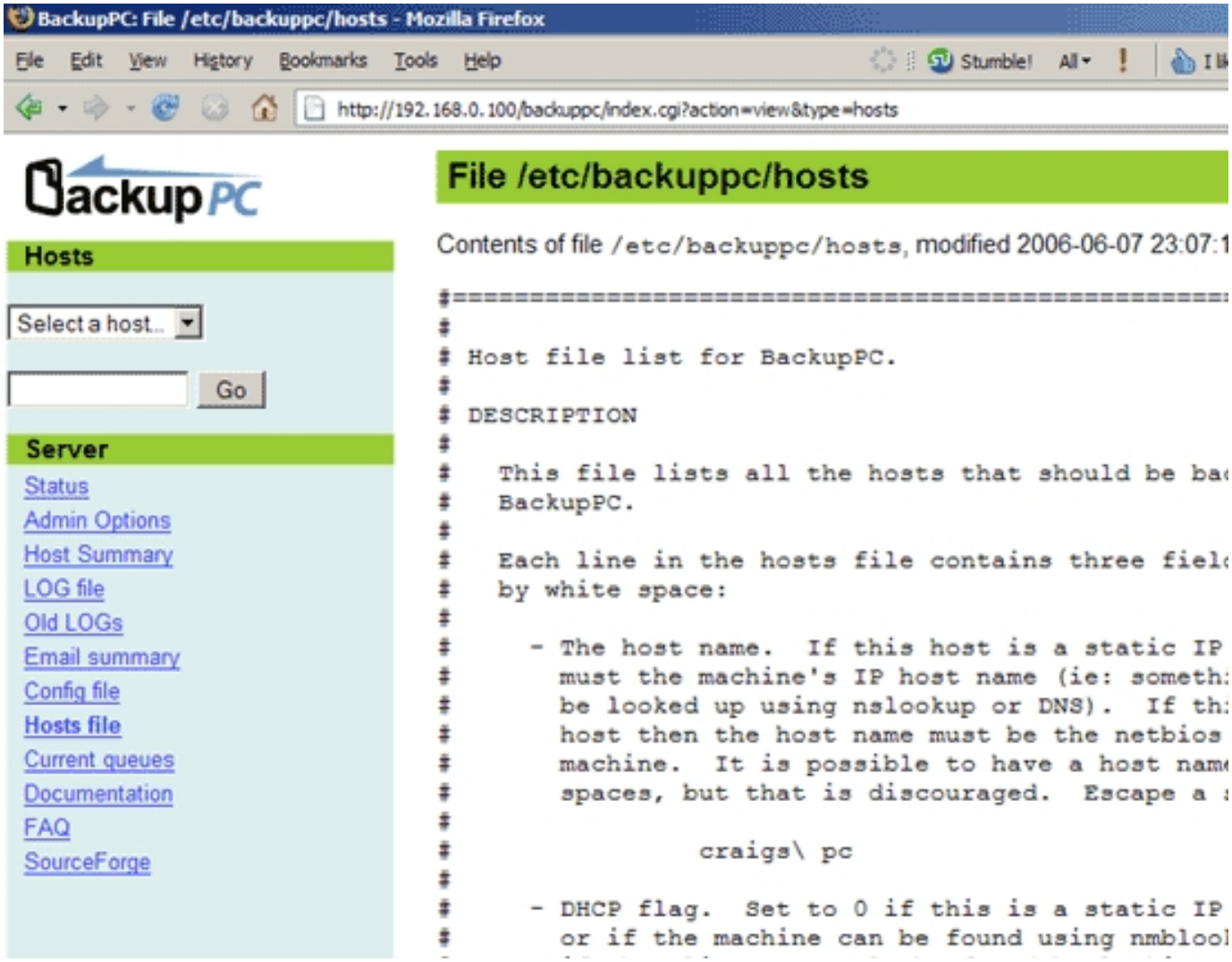
Email summary shows a list of backup reports that have been sent by email. Currently it is empty:



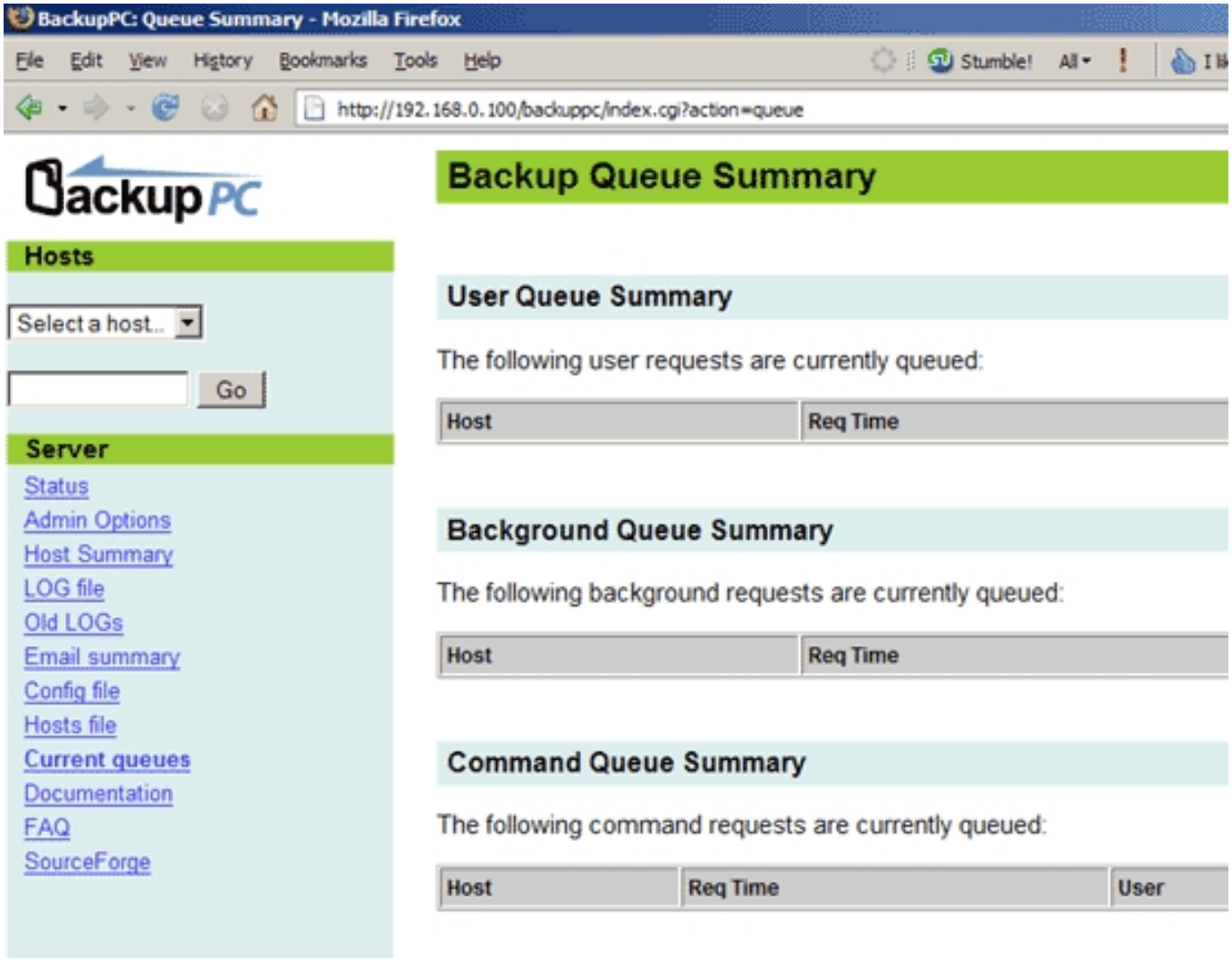
`Config file` shows the contents of BackupPC's main configuration file, `/etc/backuppc/config.pl`:



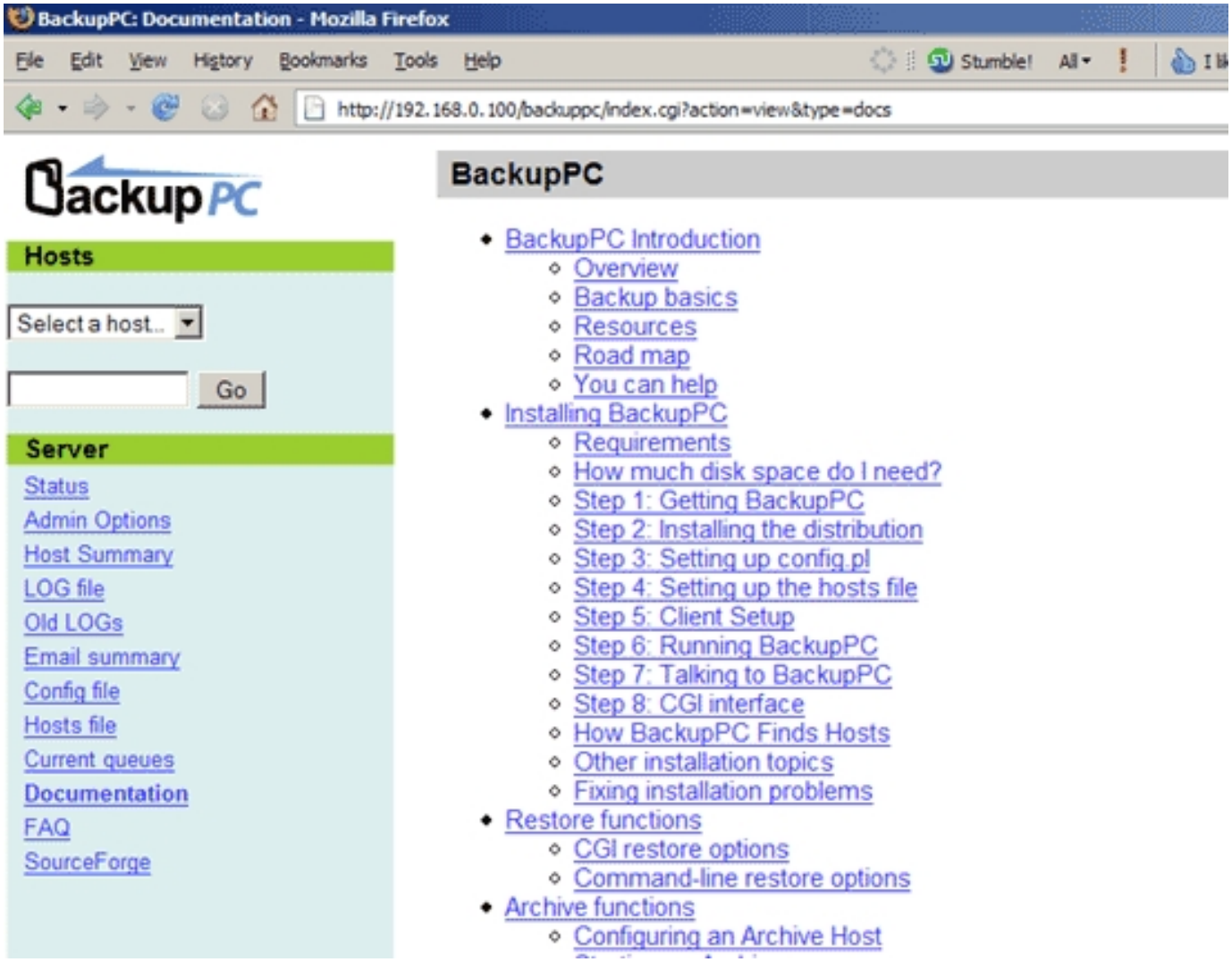
`Hosts file` shows the contents of *`/etc/backuppc/hosts`* (the file where the backup clients are specified):



Current queues shows requests that currently aren't processed:



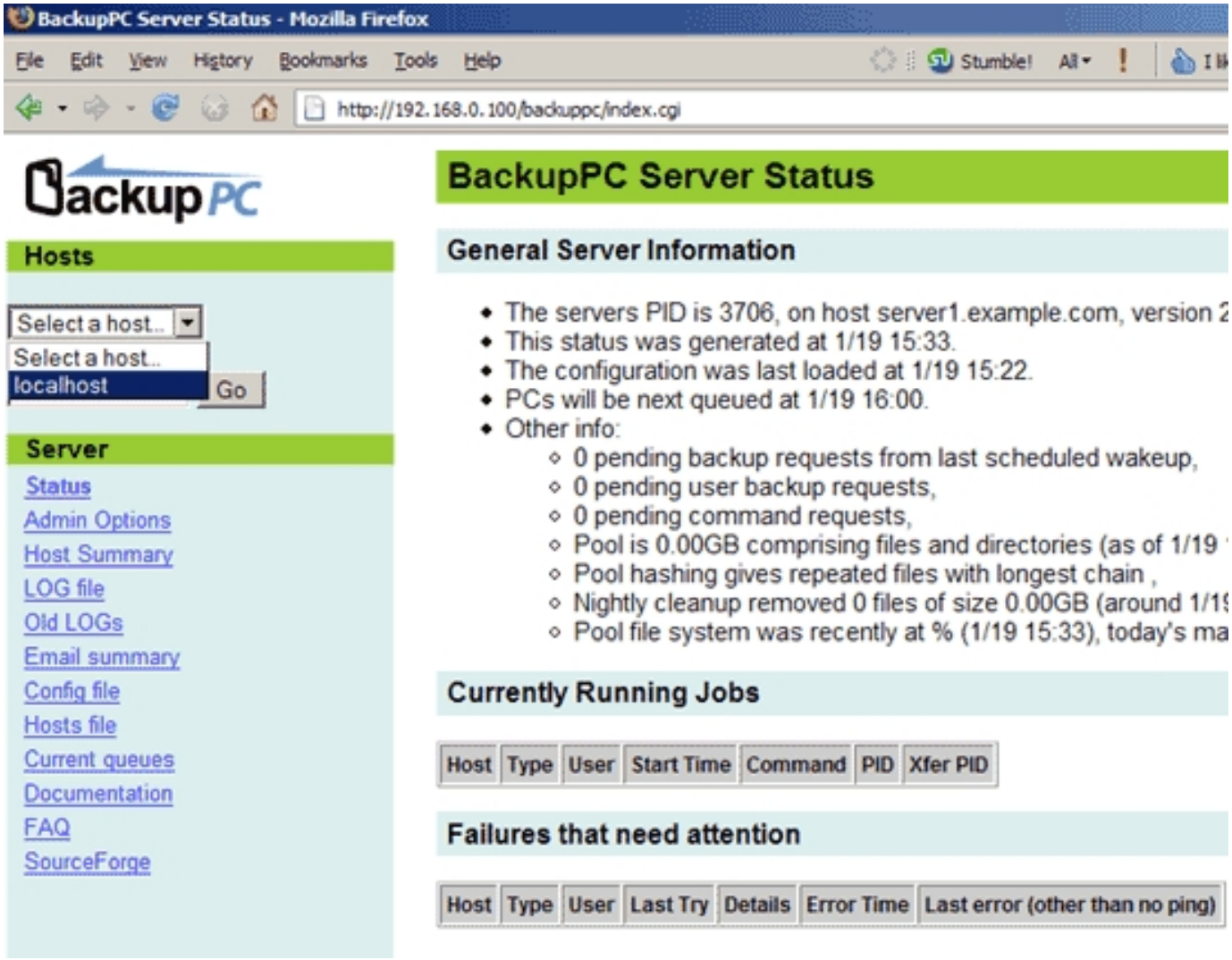
`Documentation` contains the whole BackupPC documentation. You should consider studying it later on:

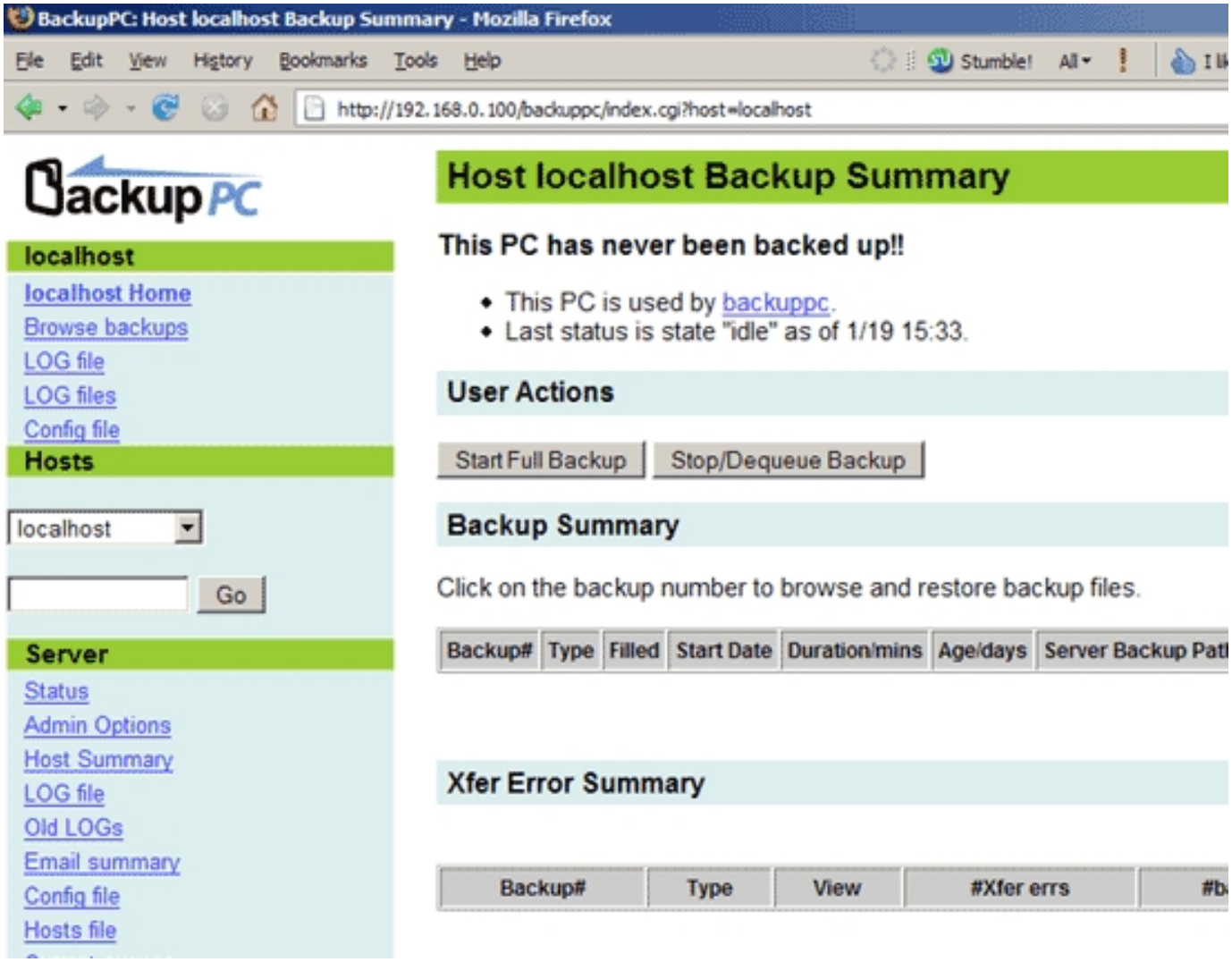


FAQ and *SourceForge* are external links to the BackupPC FAQ and BackupPC's SourceForge project site.

In the drop-down menu under *Hosts* you find all backup clients that are currently configured. *localhost* is in there by default, but as I said before, we will

remove it.





4 The BackupPC Configuration Files

BackupPC's configuration files are in the `/etc/backuppc` directory. The most important ones are `/etc/backuppc/config.pl` and `/etc/backuppc/hosts`.

You should open them now to become familiar with all configuration options. Both files are heavily commented, so by reading them you should understand what they do. For example, in */etc/backuppc/config.pl* you can define how often and when BackupPC wakes up to back up the clients, when/how often it does full or incremental backups, how many log files it should keep, etc.

```
vi /etc/backuppc/config.pl
```

```
vi /etc/backuppc/hosts
```

Don't modify them now - most default values are ok to get started.

5 Add The Ubuntu System falko-desktop As A Backup Client

To add a client to BackupPC, we must add it to */etc/backuppc/hosts*. That file has the following format:

```
host        dhcp    user      moreUsers
hostname1     0     user1     anotheruser,athirduser
hostname1     1     user2     stillanotheruser
```

The first column contains the hostname/NetBIOS name of the client (e.g. *falko-desktop*), the second column whether the client is configured through DHCP, the third column contains a user that is allowed to log into the BackupPC web interface to manage that backup client, and the fourth (optional) column contains additional users that can do the same.

Although *falko-desktop* is configured through DHCP, *dhcp* should only be set to 1 if the following tests fail:

```
nmblookup falko-desktop
```

If the test is successful, it should display *falko-desktop*'s current IP address:

```
querying falko-desktop on 192.168.0.255
192.168.0.213 falko-desktop<00>
```

Now that you know *falko-desktop*'s current IP address, run the same test in the other direction:

```
nmblookup -A 192.168.0.213
```

If successful, the output should look like this:

```
Looking up status of 192.168.0.213
        FALKO-DESKTOP   <00> -         B <ACTIVE>
        FALKO-DESKTOP   <03> -         B <ACTIVE>
        FALKO-DESKTOP   <20> -         B <ACTIVE>
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>
        MSHOME          <00> - <GROUP> B <ACTIVE>
        MSHOME          <1d> -         B <ACTIVE>
        MSHOME          <1e> - <GROUP> B <ACTIVE>

        MAC Address = 00-00-00-00-00-00
```

Both tests were successful, so we can set *dhcp* to 0, and we put *falko-desktop 0 falko* into */etc/backuppc/hosts* and comment out the *localhost* line:

```
vi /etc/backuppc/hosts
```

```
[...]
falko-desktop   0       falko
#localhost   0       backuppc
```

If the tests had not been successful, we would have put *falko-desktop 1 falko* into */etc/backuppc/hosts* instead, and in addition to that we would have had to change the variable *$Conf{DHCPAddressRanges}* in */etc/backuppc/config.pl*.

We have added the user *falko* to the hosts file. That's the system user whom BackupPC will send email reports regarding *falko-desktop*, so we should create that user on the BackupPC server now:

```
useradd -m falko

passwd falko
```

In addition to that we must create a password for *falko* for the BackupPC web interface:

```
htpasswd /etc/backuppc/htpasswd falko
```

(If you like you can now open a second browser and log in as *falko* on *http://192.168.0.100/backuppc/*.)

Now we must edit */etc/backuppc/config.pl*. The backup method for *falko-desktop* will be *rsync*, so we set *$Conf{XferMethod} = 'rsync';*. *falko-desktop* has just one (big) partition (*/*), so we set *$Conf{RsyncShareName} = '/';*. If you have multiple partitions to backup, you'd specify something like *$Conf{RsyncShareName} = ['/', '/var', '/data', '/boot'];* instead.

rsync will be tunneled through SSH (port 22). SSH needs to know the hostname or IP address of the client. Unless you have a DNS server that resolves *falko-desktop* to its IP address, SSH will not be able to resolve the name *falko-desktop*. Therefore we replace *$host* with *$hostIP* in

$Conf{RsyncClientCmd}* and *$Conf{RsyncClientRestoreCmd}.

```
vi /etc/backuppc/config.pl
```

```
[...]
$Conf{XferMethod} = 'rsync';

$Conf{RsyncClientPath} = '/usr/bin/rsync';
$Conf{RsyncClientCmd} = '$sshPath -q -x -l root $hostIP $rsyncPath $argList+';
$Conf{RsyncClientRestoreCmd} = '$sshPath -q -x -l root $hostIP $rsyncPath $argList+';
$Conf{RsyncShareName} = '/';
#$Conf{RsyncShareName} = ['/', '/var', '/data', '/boot'];
[...]
```

(You might wonder now how we can specify options for multiple clients because the options for *notebook* will be different from those for *falko-desktop*. It works like this: the options in */etc/backuppc/config.pl* are **global** values that are valid for all clients **unless** they have their onw configuration file in */etc/backuppc*, named after their hostname. So for *notebook* we will create */etc/backuppc/notebook.pl*, and it will contain only the values that are different from the ones in */etc/backuppc/config.pl*. In fact, we could have created */etc/backuppc/falko-desktop.pl* for *falko-desktop* instead of modifying */etc/backuppc/config.pl*.)

Next restart BackupPC:

```
/etc/init.d/backuppc restart
```

5.1 Configure The SSH Tunnel

The *rsync* backup will be tunneled through SSH. The backup is run as the user *backuppc*, therefore this user must be able to login to *falko-desktop* as root without being prompted for a password. Therefore we must exchange public keys to allow password-less logins for *backuppc*.

First we must log in on *falko-desktop* on the shell and create a root login (if you don't use Ubuntu you most probably have one already):

falko-desktop:

```
sudo passwd root

sudo su
```

Now that you're logged in as root, install OpenSSH and rsync:

```
apt-get install rsync ssh openssh-server
```

Then create a private/public key pair:

```
ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
  Enter file in which to save the key (/root/.ssh/id_rsa):
 Created directory '/root/.ssh'.
  Enter passphrase (empty for no passphrase):
 Enter same passphrase again:
 Your identification has been saved in /root/.ssh/id_rsa.
  Your public key has been saved in /root/.ssh/id_rsa.pub.
```

```
  The key fingerprint is:
0f:95:00:4b:fd:c3:cc:0b:1f:2b:00:c9:29:bf:ca:4e root@falko-desktop
```

If you don't have a DNS record for *server1.example.com*, you should add *server1.example.com* to */etc/hosts* now:

```
vi /etc/hosts
```

```
[...]
192.168.0.100   server1.example.com server1
[...]
```

Next we create a private/public key pair on *server1.example.com*. We must do this as the user *backuppc*!

server1.example.com:

```
su backuppc

ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
  Enter file in which to save the key (/var/lib/backuppc/.ssh/id_rsa):
 Created directory '/var/lib/backuppc/.ssh'.
  Enter passphrase (empty for no passphrase):
 Enter same passphrase again:
 Your identification has been saved in /var/lib/backuppc/.ssh/id_rsa.
  Your public key has been saved in /var/lib/backuppc/.ssh/id_rsa.pub.
  The key fingerprint is:
```

```
74:20:65:73:47:1c:cb:ba:5d:9b:5d:56:cf:91:1a:1a backuppc@server1.example.com
```

Then we copy the public key to *falko-desktop*. Make sure you use *falko-desktop*'s current IP address in the *scp* command:

```
cp ~/.ssh/id_rsa.pub ~/.ssh/BackupPC_id_rsa.pub

scp ~/.ssh/BackupPC_id_rsa.pub root@192.168.0.213:/root/.ssh/
```

```
The authenticity of host '192.168.0.213 (192.168.0.213)' can't be established.
  RSA key fingerprint is 9b:66:3e:ce:b4:8d:63:00:ba:87:14:b2:94:03:cb:a8.
  Are you sure you want to continue connecting (yes/no)?
 Warning: Permanently added '192.168.0.213' (RSA) to the list of known hosts.
  root@192.168.0.213's password:
 BackupPC_id_rsa.pub 100% 410 0.4KB/s 00:00
```

Next we append *backuppc*'s public key to *~/.ssh/authorized_keys2* on *falko-desktop* (we do this as root):

```
cat ~/.ssh/BackupPC_id_rsa.pub >> ~/.ssh/authorized_keys2
```

If you have a proper DNS record for *server1.example.com* or added it to *falko-desktop*'s */etc/hosts* file, you can now open *~/.ssh/authorized_keys2* and add *from="server1.example.com"* at the beginning of the file. Thus only *server1.example.com* can enjoy password-less logins. (If *server1.example.com* cannot be resolved on *falko-desktop*, then don't add *from="server1.example.com"*)

```
vi ~/.ssh/authorized_keys2
```

```
from="server1.example.com" ssh-rsa AAAAB3[...]FMZpdAj8Hs9107tZ97Rq2oO/Zw== backuppc@server1.example.com
```

Then copy *root@falko-desktop*'s public key to *server1.example.com* (make sure you use the correct IP address):

```
scp ~/.ssh/id_rsa.pub root@192.168.0.100:/var/lib/backuppc/.ssh/client_id_rsa.pub
```

```
The authenticity of host '192.168.0.100 (192.168.0.100)' can't be established.
  RSA key fingerprint is 29:40:1c:c0:40:f8:e1:4c:68:47:36:b3:f3:53:b1:38.
  Are you sure you want to continue connecting (yes/no)?
 Warning: Permanently added '192.168.0.100' (RSA) to the list of known hosts.
  root@192.168.0.100's password:
 id_rsa.pub 100% 400 0.4KB/s 00:00
```

Back on *server1.example.com*, we append *root@falko-desktop*'s public key to *~/.ssh/known_hosts*. Make sure you're still logged in as the user *backuppc*!

server1.example.com:

```
cat ~/.ssh/client_id_rsa.pub >> ~/.ssh/known_hosts
```

Then we switch back to the root user and delete */var/lib/backuppc/.ssh/client_id_rsa.pub*:

```
su
```



```
rm -f /var/lib/backuppc/.ssh/client_id_rsa.pub
```

Then become *backuppc* again and change the permissions of the *~/.ssh* directory:

```
su backuppc

chmod -R go-rwx ~/.ssh
```

Do the same on *falko-desktop* (as root):

falko-desktop:

```
chmod -R go-rwx ~/.ssh
```

Then go back to *server1.example.com* and make sure you're still logged in as *backuppc*. Run the following test command to see if *falko-desktop* prompts you for a password. If you did everything right, it shouldn't. (Make sure to use *falko-desktop*'s current IP address!)

server1.example.com:

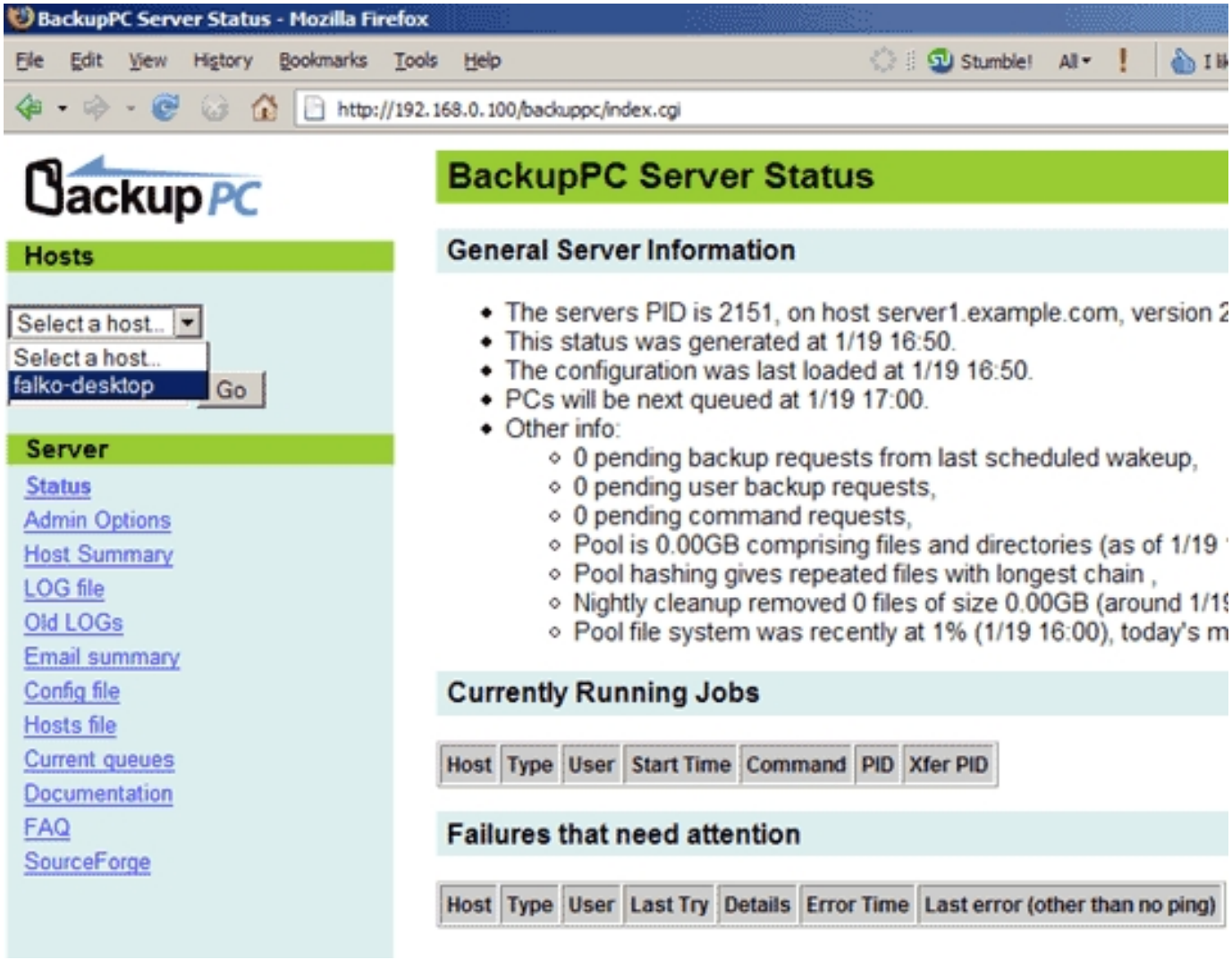
```
ssh -l root 192.168.0.213 whoami
```

The output should simply be

root

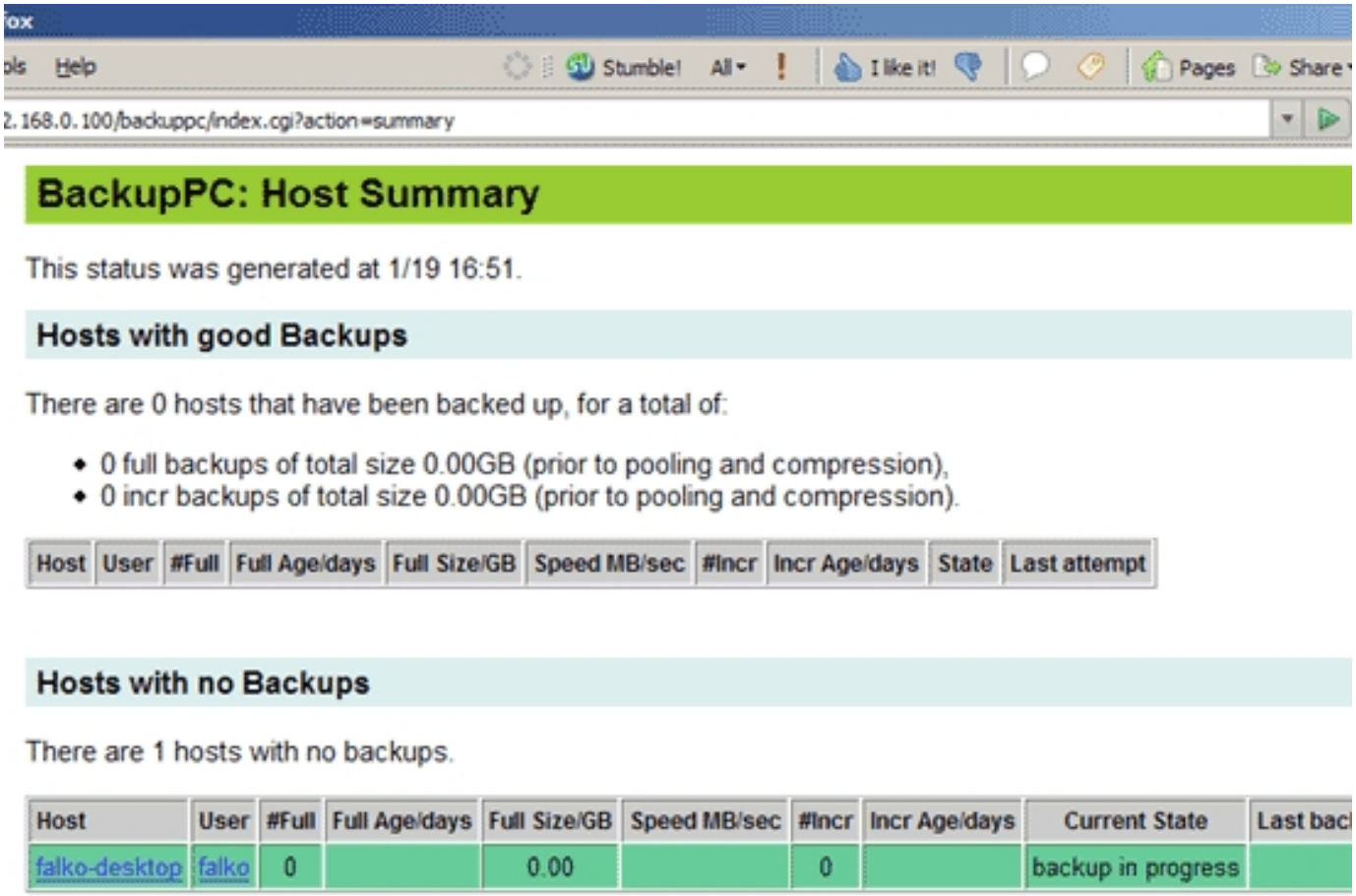
6 Our First Backup

Now that our SSH tunnel is working, we can reload the BackupPC web interface at *http://192.168.0.100/backuppc/*. You should now see *falko-desktop* in the list of hosts, and *localhost* has gone:

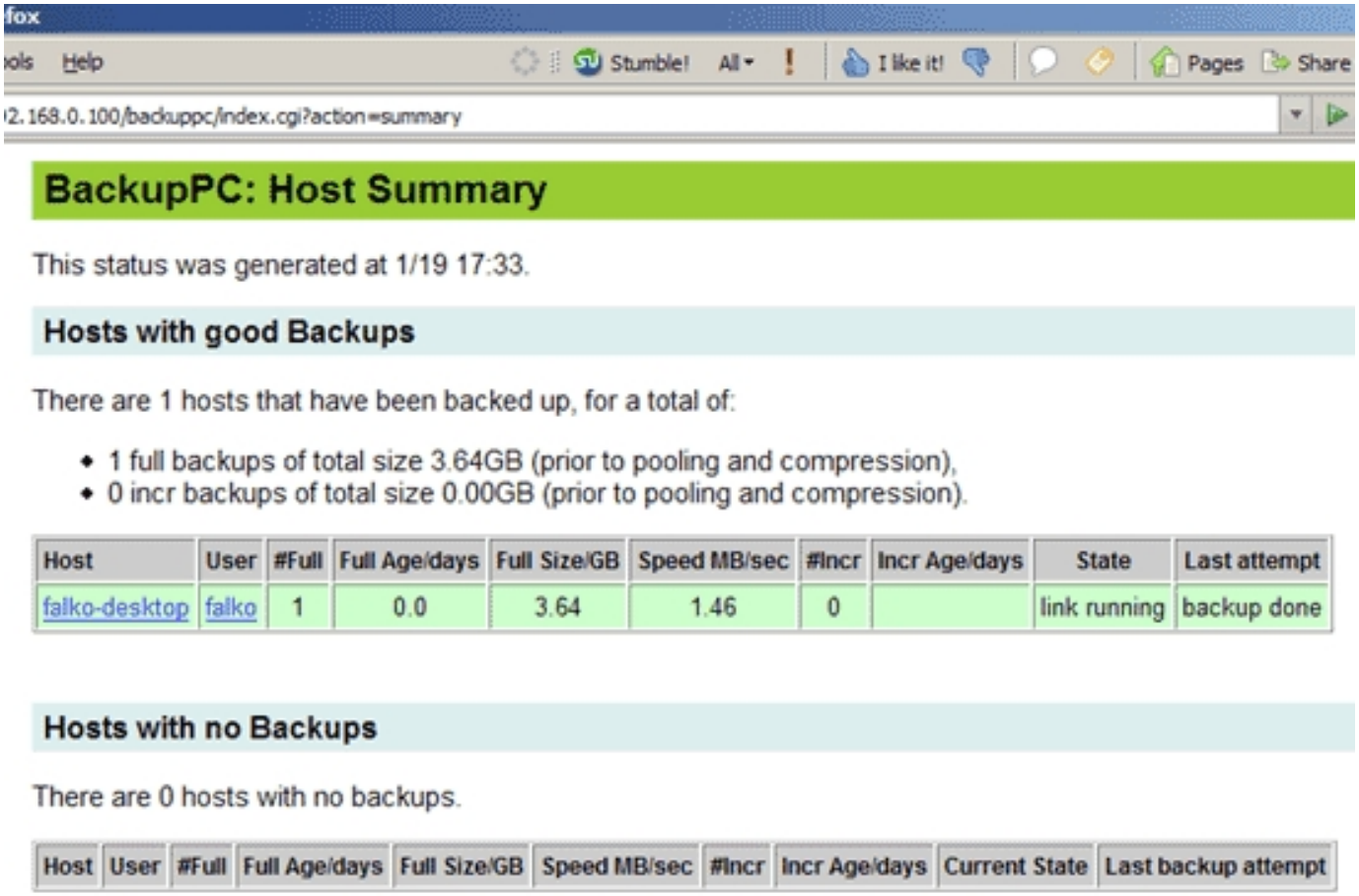


You can wait until BackupPC starts its first backup of `falko-desktop` (in the default configuration it does so at the first full hour), or you start it manually through the web interface.

While the backup is in progress, the *Host Summary* page looks like this:



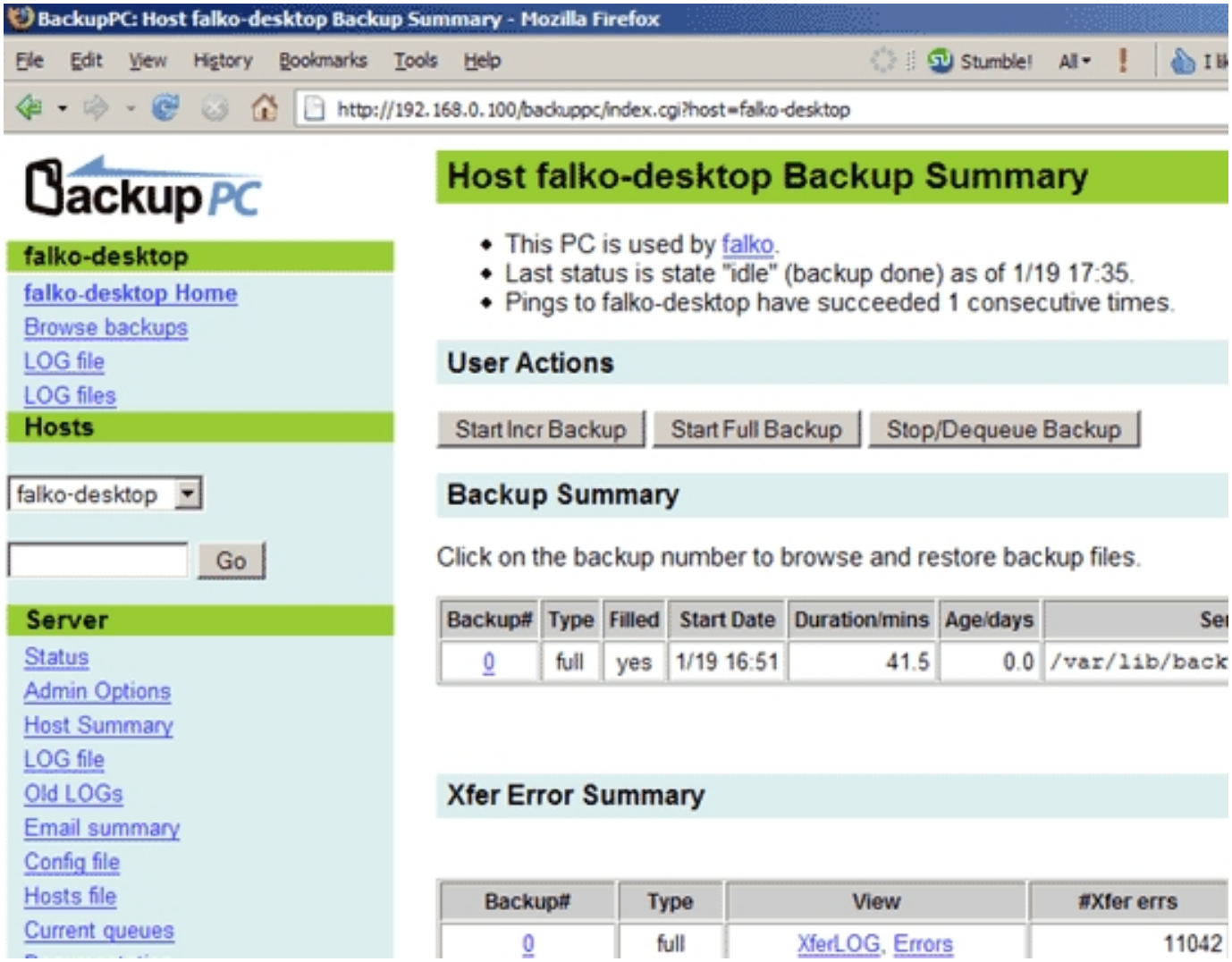
When the backup is finished, it looks like this:



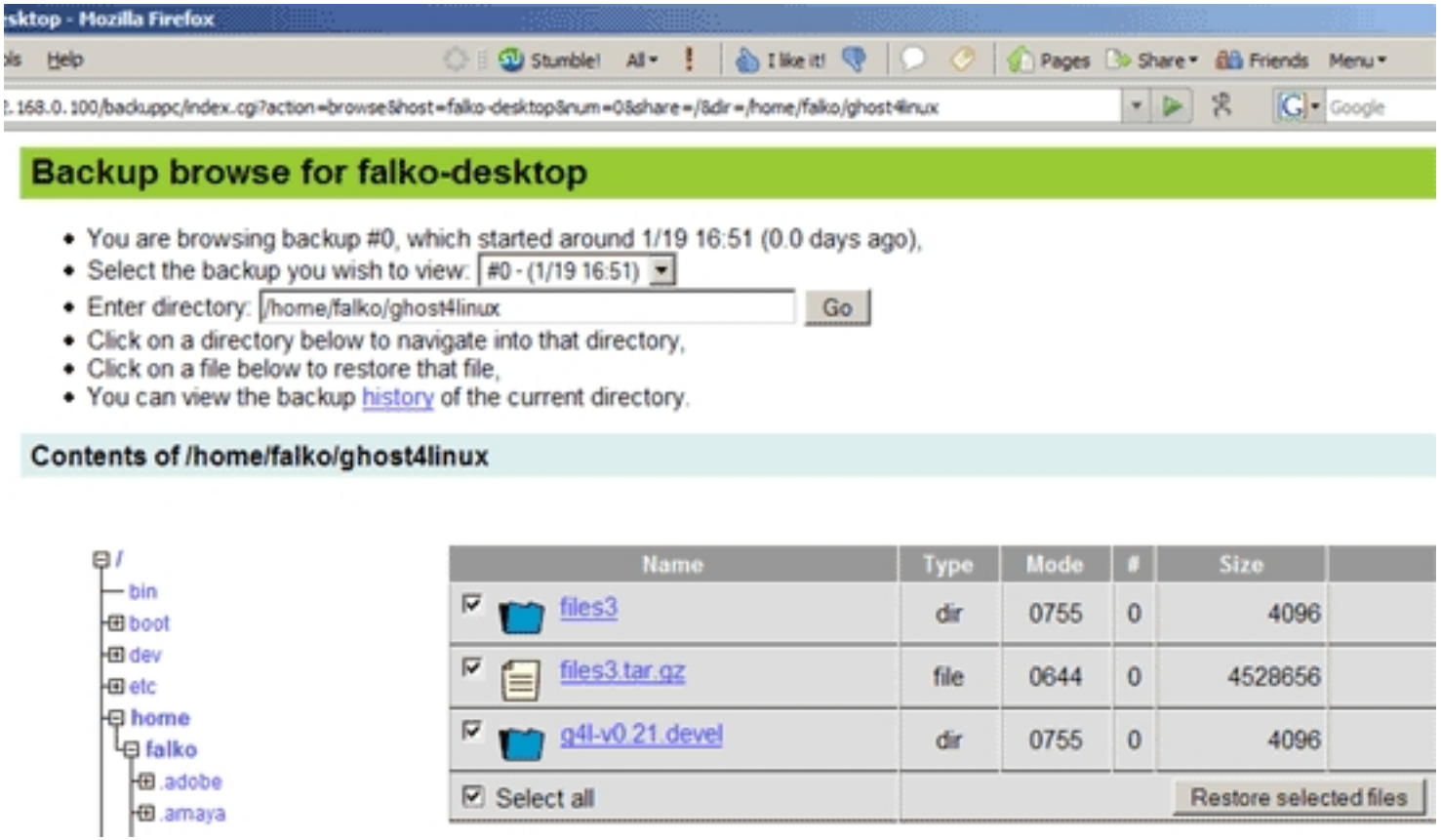
7 Our First Recovery

For testing purposes we can now do our first recovery of files. Go to *falko-desktop Home* and click on the number of the backup under *Backup Summary*

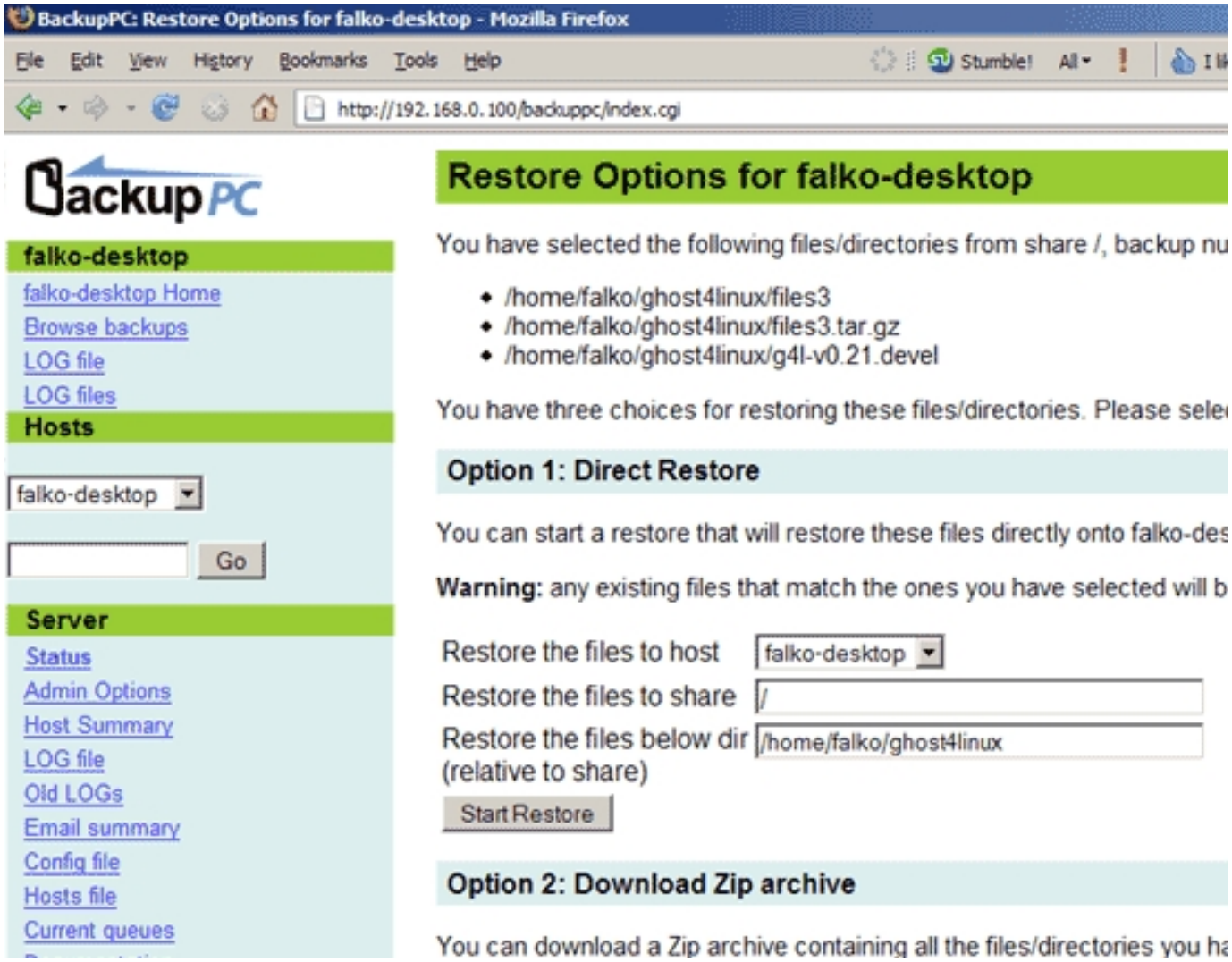
(in this case `0`):



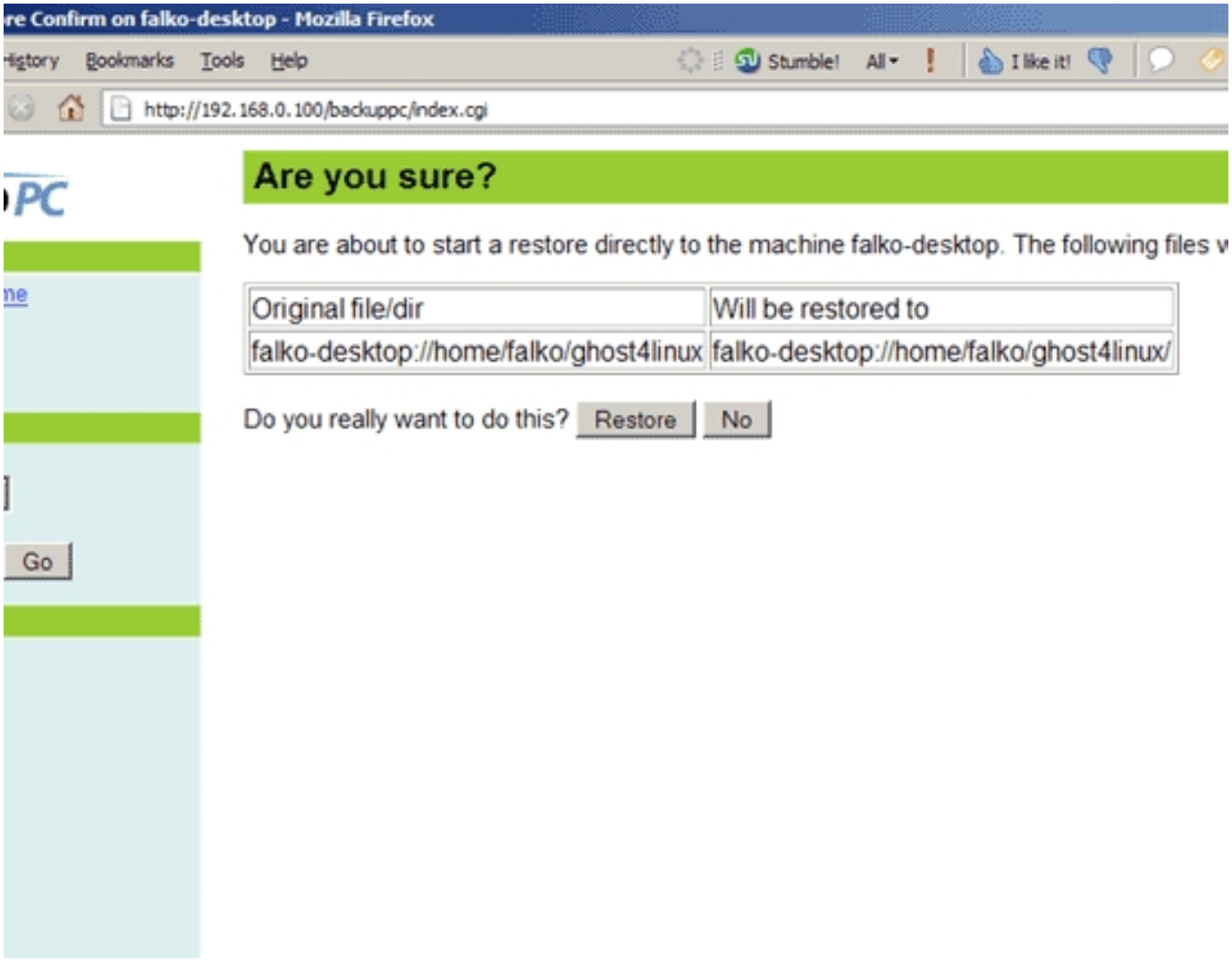
On the next page you will see a tree of files and directories from *`falko-desktop`*. Browse the tree and select the files/directories you want to restore, then click on *`Restore selected files`*:



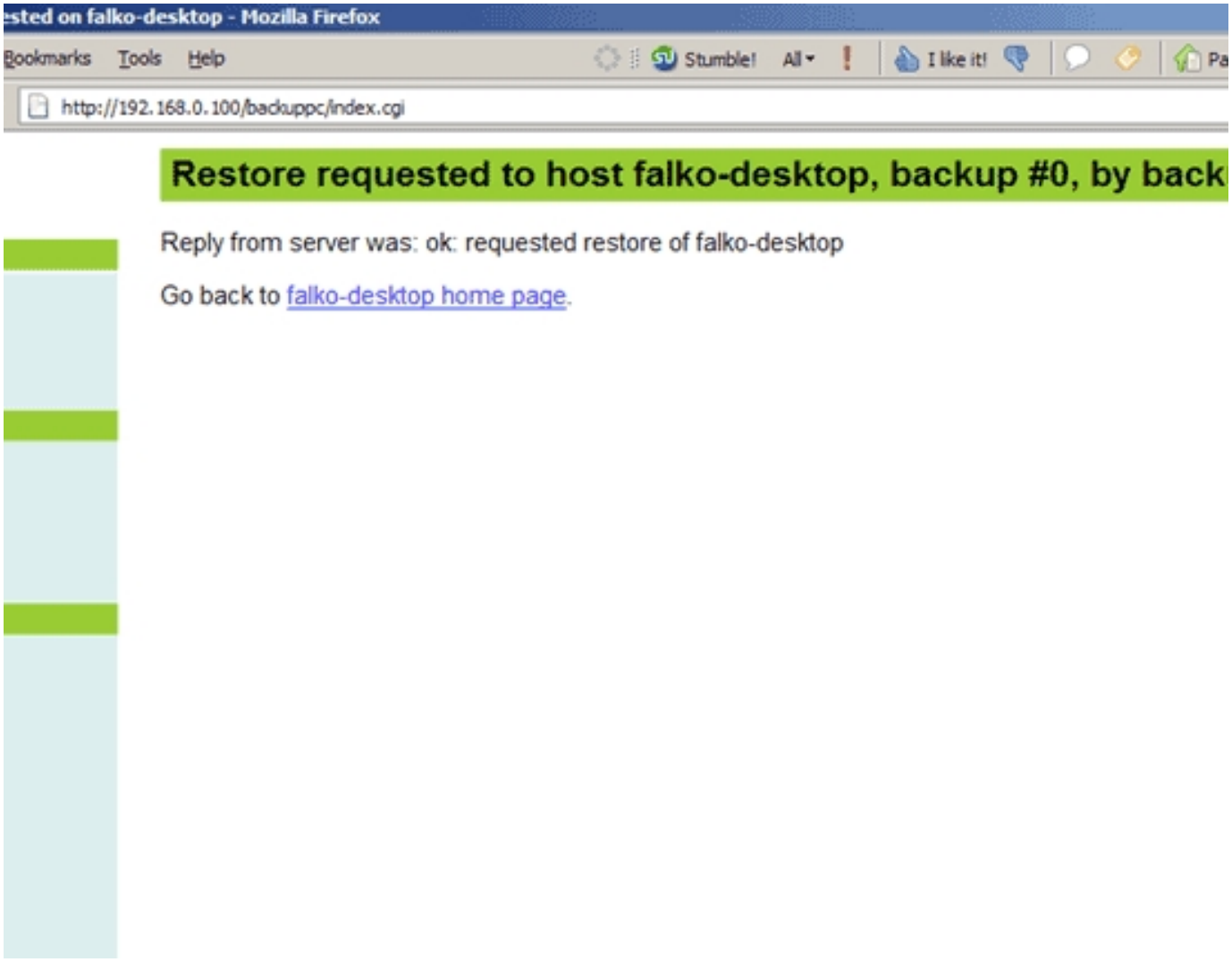
The next page shows you three different methods to restore the files/directories. In this case I select the first one (*`Direct Restore`*) and click on *`Start Restore`*:

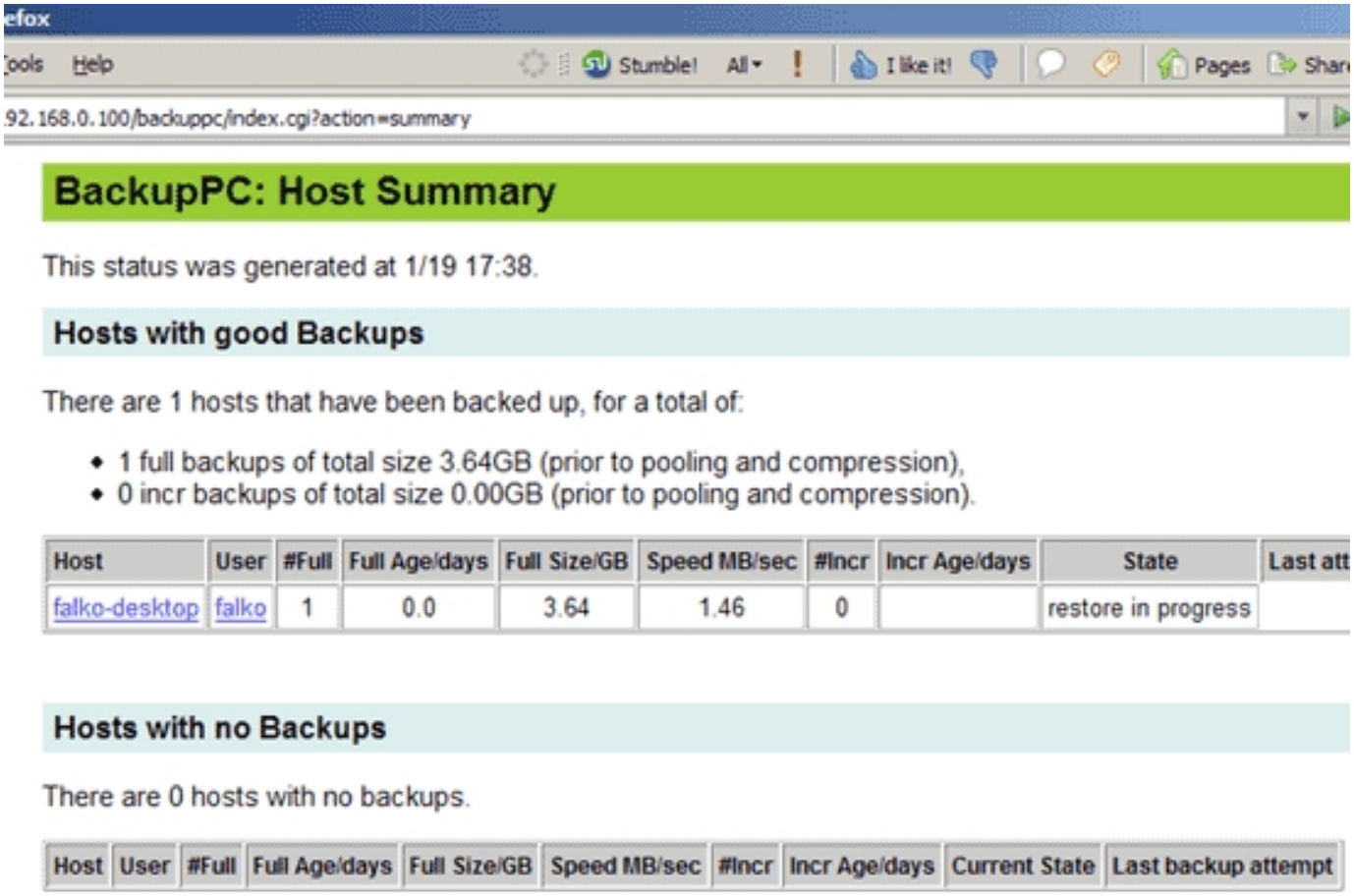


Confirm this by clicking on *Restore*:



The recovery begins:

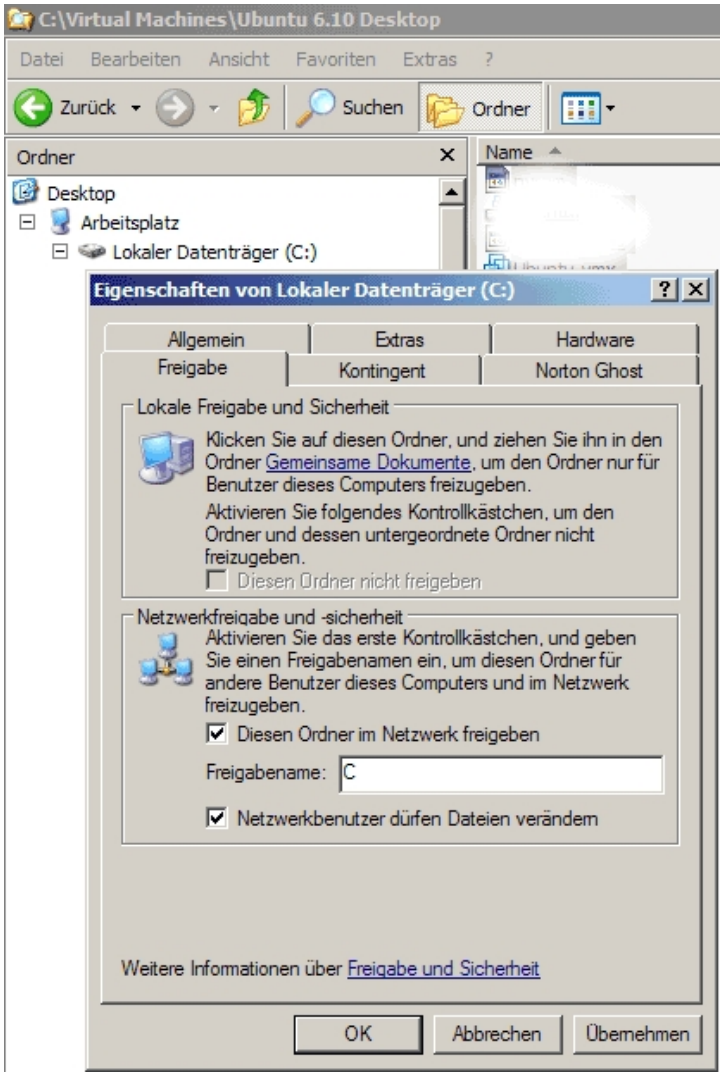




8 Add The Windows System notebook As A Backup Client

To make backups of Windows systems using `smb`, we must share the folder that we want to back up. In this case I want to backup the whole `C:` drive, so I

right-click on it in the Windows Explorer and go to *Sharing*. As share name I specify *C*.



Next, on *server1.example.com*, I add a line for *notebook* to */etc/backuppc/hosts* (I do this as root). In this case I use *falko* again as the user. If you use a different username than before, make sure you create it on the system and for the BackupPC web interface, as shown in chapter 5.

server1.example.com:

```
vi /etc/backuppc/hosts
```

```
[...]
falko-desktop   0       falko
notebook    0       falko
#localhost   0       backuppc
```

Then create the file */etc/backuppc/notebook.pl* and add all options that are different from the ones in */etc/backuppc/config.pl*. In this case we add *$Conf{XferMethod} = 'smb';*. Our share name is *C*, so we put *$Conf{SmbShareName} = 'C';* into it as well as the username and password for the Windows share:

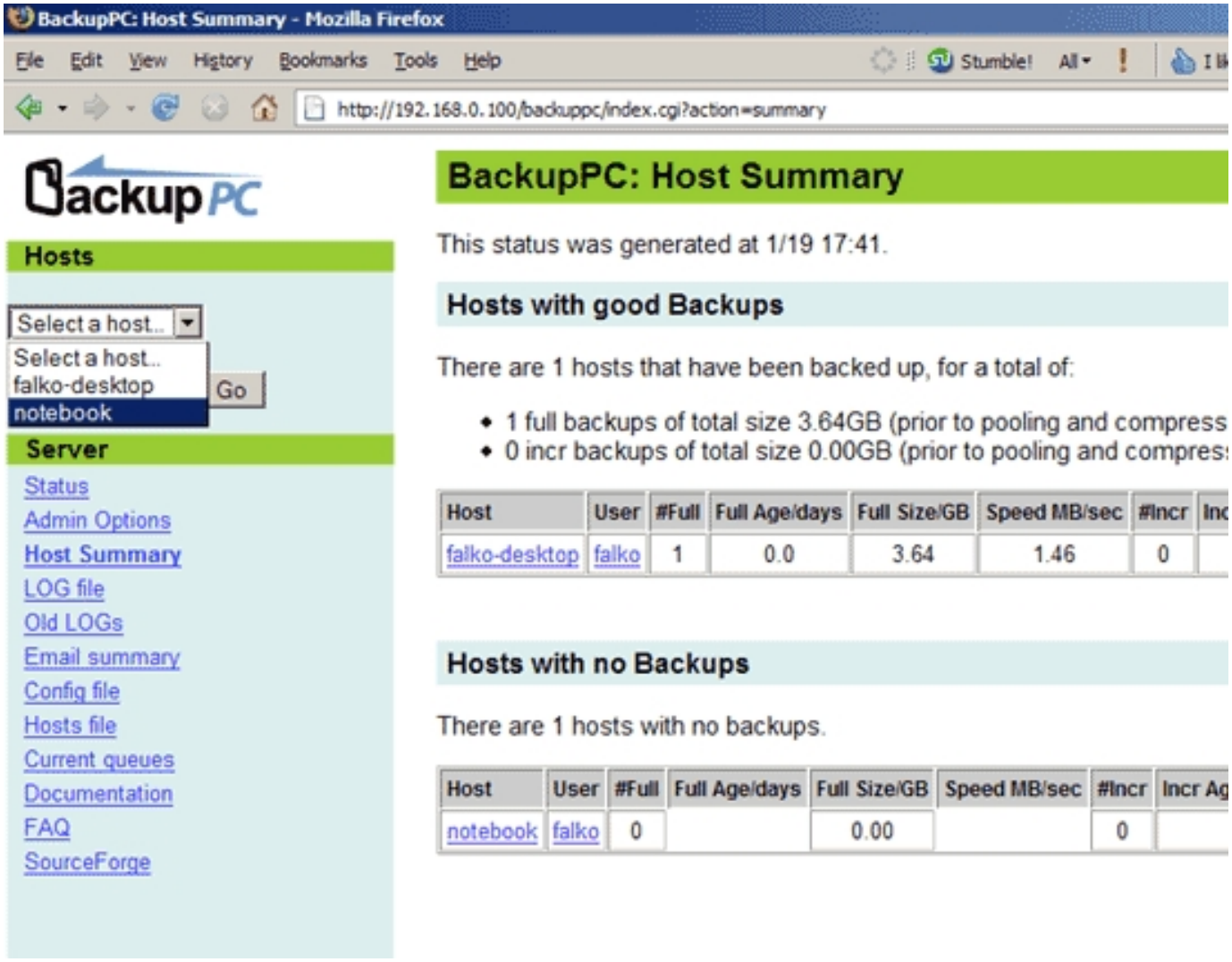
```
vi /etc/backuppc/notebook.pl
```

```
$Conf{SmbShareName} = 'C';
$Conf{SmbShareUserName} = 'username';
$Conf{SmbSharePasswd} = 'password';
$Conf{XferMethod} = 'smb';
```

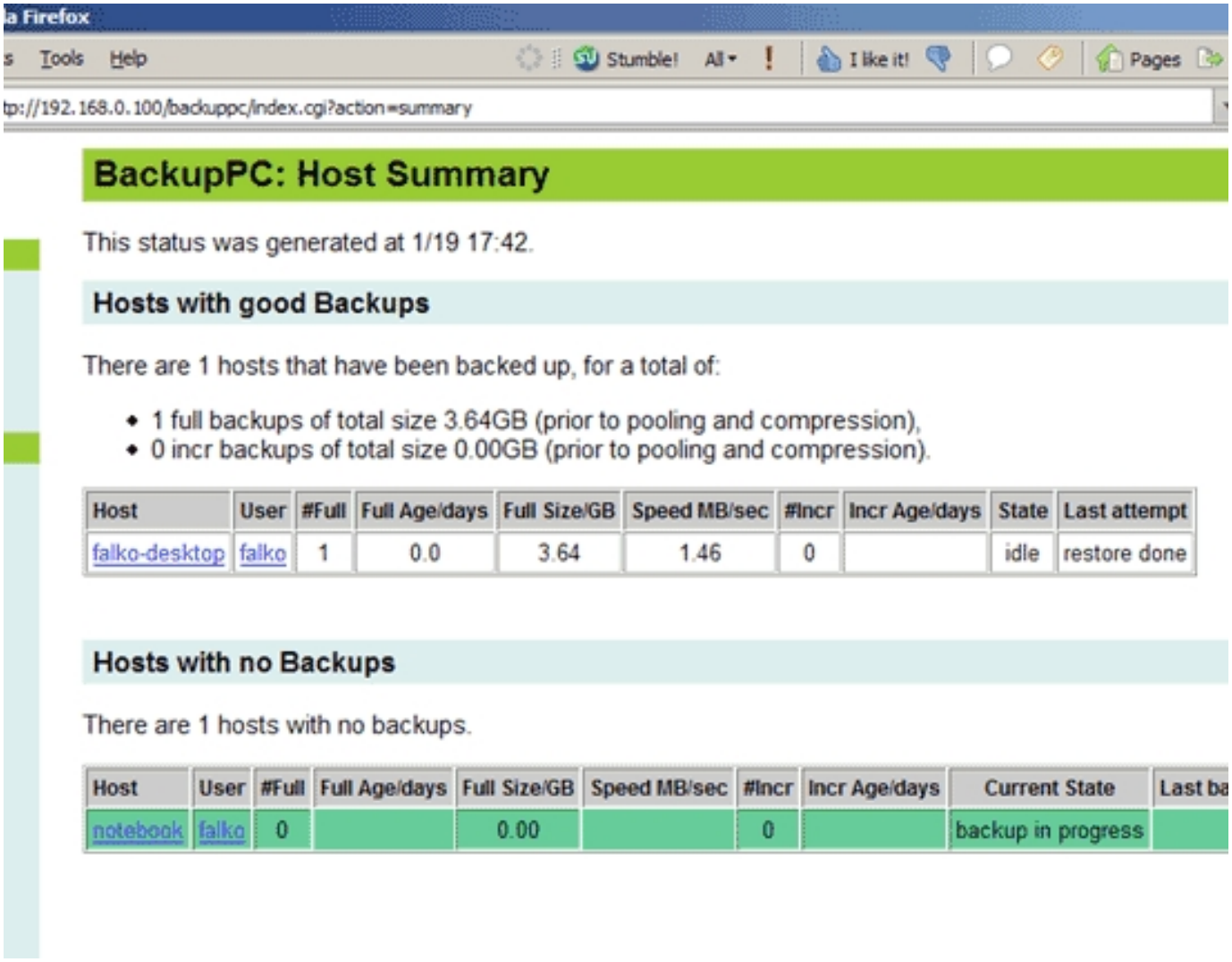
Then restart BackupPC:

```
/etc/init.d/backuppc restart
```


Then reload the BackupPC web interface again. You should now find *notebook* in the list of clients:



You can start the first backup of *notebook* manually or wait until BackupPC starts it:



That's it already for Windows clients.

Happy backupping! ;-)

9 Links

- BackupPC: **http://backuppc.sourceforge.net**
- BackupPC Documentation: **http://backuppc.sourceforge.net/faq/BackupPC.html**