

kathara lab

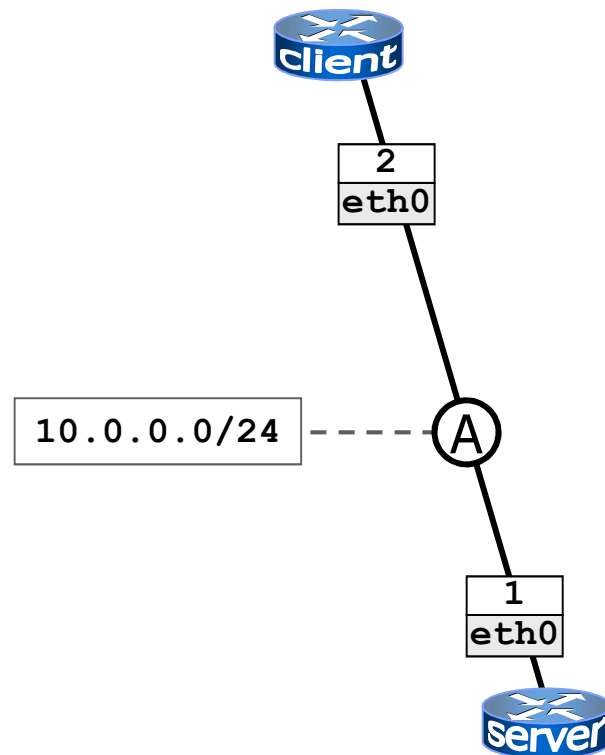
web server and browser

Version	1.1
Author(s)	Giuseppe Di Battista, Maurizio Patrignani, Massimo Rimondini
E-mail	contact@kathara.org
Web	http://www.kathara.org/
Description	A lab showing the operation of a web server accessed by a browser client – kathara simplified version of the corresponding netkit lab vers. 1.2

copyright notice

- All the pages/slides in this presentation, including but not limited to, images, photos, animations, videos, sounds, music, and text (hereby referred to as “material”) are protected by copyright.
- This material, with the exception of some multimedia elements licensed by other organizations, is property of the authors and/or organizations appearing in the first slide.
- This material, or its parts, can be reproduced and used for didactical purposes within universities and schools, provided that this happens for non-profit purposes.
- Information contained in this material cannot be used within network design projects or other products of any kind.
- Any other use is prohibited, unless explicitly authorized by the authors on the basis of an explicit agreement.
- The authors assume no responsibility about this material and provide this material “as is”, with no implicit or explicit warranty about the correctness and completeness of its contents, which may be subject to changes.
- This copyright notice must always be redistributed together with the material, or its portions.

lab topology




lab description

- server
 - runs apache2 (with a default configuration)
- client
 - the user can launch a text-based web browser (**links**) to check the server operation

server

- the user can check that apache2 is up and running by using the following command:



```
server:~# /etc/init.d/apache2 status
Apache is running (pid 485)..
server:~# █
```

- we have put a test html page, located in `/var/www/html/index.html`

```
<html><body><h1>Hello!</h1></body></html>
```

client

- the user is supposed to start the web browser **links** on the client



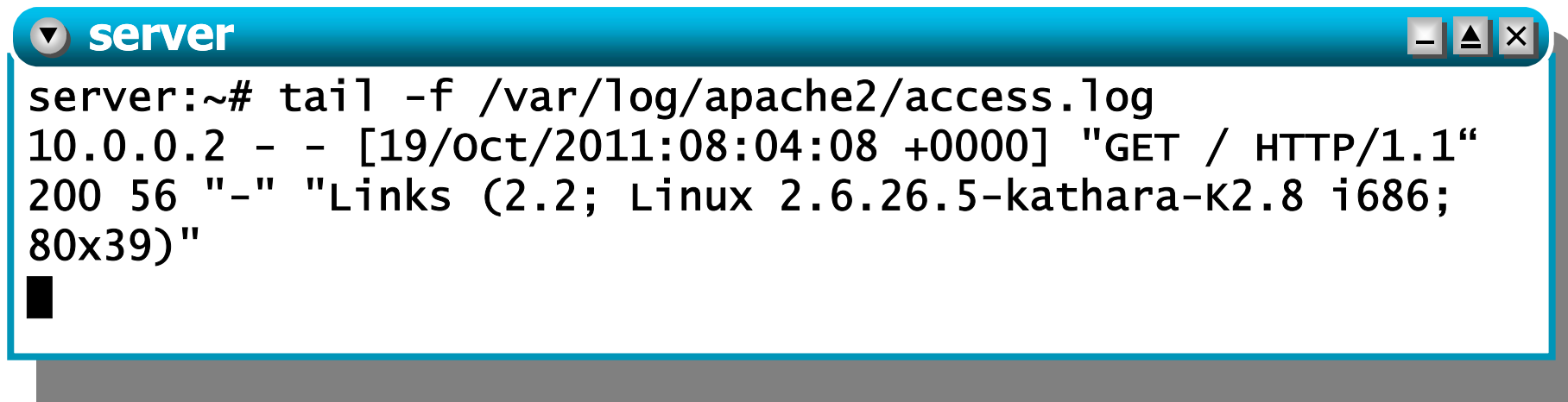
- an empty screen is presented to the user...
- to access the menu bar, press F10
- using the cursor keys, select "Go to URL" and press Enter

client

- enter the following URL:
`http://10.0.0.1/`
- you should get a screen saying “Hello!”

server (again)

- to monitor accesses to the web server you can use the following command (on the server):



```
server:~# tail -f /var/log/apache2/access.log
10.0.0.2 - - [19/Oct/2011:08:04:08 +0000] "GET / HTTP/1.1"
200 56 "-" "Links (2.2; Linux 2.6.26.5-kathara-K2.8 i686;
80x39)"
```


server (again)

- to monitor errors on the web server you can use the following command (on the server):

```
server
server:~# tail -f /var/log/apache2/error.log
[Wed Nov 14 15:57:58 2012] [notice] Apache/2.2.9 (Debian)
configured -- resuming normal operations
[Wed Nov 14 16:14:07 2012] [notice] caught SIGTERM, shutting
down
█
```



tip: very useful when debugging configurations

apache modules

- most of apache's functionalities are built-in
 - retrieve the list using `apache2 -l`
- others can be added by enabling modules
 - to enable a module:

▼ server

```
server:~# a2enmod rewrite
Enabling module rewrite.
Run '/etc/init.d/apache2 restart' to activate new
configuration!
server:~# █
```



apache must be (re)started afterwards

apache modules

- available modules are located in `/etc/apache2/mods-available`
- enabled modules are located in `/etc/apache2/mods-enabled`
- `a2enmod` puts a symbolic link from the relevant file(s) in `/etc/apache2/mods-available` to `/etc/apache2/mods-enabled`
- `a2dismod` removes these symbolic links

some useful apache modules

<code>userdir</code>	enables per-user web sites (this feature does not work with Kathará)
<code>rewrite</code>	implements URL rewriting
<code>proxy</code>	implements a proxy/gateway
<code>cgi/cgid</code>	supports execution of CGI scripts

per-directory configuration

- apache allows configuration changes on a per-directory basis
- creating a special file
`/some/path/.htaccess` with apache configuration statements applies those statements to all files and subdirectories inside `/some/path`
 - `.htaccess` files can be nested in a directory tree
 - nested files override their parents

per-directory configuration

■ sample configuration statements:

■ restrict access from specific hosts

```
Deny from example.org test.com 10.0.0 192.168.0.0/24
```

■ perform URL rewriting

- (transparently) redirect to other sites

■ restrict access to a specific subdirectory

- enable client-side authentication

■ change name of file containing the default page

```
DirectoryIndex pippo.html
```

■ enable/disable directory indexing

```
Options -Indexes
```

exercise: per-directory configuration

- when a resource name is not specified in the URL, apache serves `index.html` from the requested path
- hands-on:
 - edit file `/var/www/html/.htaccess` and add the following directive:
`DirectoryIndex custom_file.html`
 - rename previously created file `/var/www/html/index.html` to `custom_file.html`
 - try accessing `http://10.0.0.1/` from client
 - rename `custom_file.html` back to `index.html` and try accessing the page again