netkit lab
walkthrough

<table>
<thead>
<tr>
<th>Version</th>
<th>1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Massimo Rimondini</td>
</tr>
<tr>
<td>E-mail</td>
<td><a href="mailto:contact@netkit.org">contact@netkit.org</a></td>
</tr>
<tr>
<td>Web</td>
<td><a href="http://www.netkit.org/">http://www.netkit.org/</a></td>
</tr>
<tr>
<td>Description</td>
<td>a step-by-step example showing how to set up a complete netkit lab with a few technologies</td>
</tr>
</tbody>
</table>
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.
walk through what?

- the goal of this lab is to:
  - put together some technologies presented in other labs
  - step-by-step show how to set up a netkit lab from scratch

- prerequisites
  - it is advisable to take a look at the following netkit labs beforehand:
    - two hosts
    - static routing
    - web server
    - dns
the goal
The goal

- Authority for org
- Root name server
- Authority for net

Insert this additional statement in the configuration of `ns-org` and `ns-net`:
```plaintext
options {
    allow-recursion {0/0; ;;};
};
```

Client uses `ns-org` as local name server; has `client.org` as DNS name.

Server hosts `http://server.net/~guest/`.

Static routing is illustrated with networks and routers.

Networks:
- 192.168.0.0/25
- 192.168.0.128/25
- 10.0.0.0/24
1. create an empty directory where to put the lab
2. set up physical topology
3. set up routing
4. set up additional technologies
   - in this lab:
     - web server
     - dns
vademecum

1. create an empty directory where to put the lab
2. set up physical topology
3. set up routing
4. set up additional technologies
   - in this lab:
     - web server
     - dns
1. create an empty directory where to put the lab

```bash
user@localhost:~$ mkdir mylab
user@localhost:~$ cd mylab
user@localhost:~/mylab$
```

note: here we use terminal commands for any operations if you feel more comfortable with a graphical file manager/editor, feel free to use it.
vademecum

1. create an empty directory where to put the lab
2. set up physical topology
3. set up routing
4. set up additional technologies
   - in this lab:
     - web server
     - dns
2. physical topology

- first of all, we tell netkit which virtual machines (=network nodes) the network consists of
2. physical topology

Insert this additional statement in the configuration of `ns-org` and `ns-net`:
```
options {
    allow-recursion {0/0; };
};
```

Server `r1` uses `ns-org` as local name server; has `client.org` as DNS name.

Server `r2` hosts `http://server.net/~guest/`.

Client uses `ns-org` as local name server; has `client.org` as DNS name.

Authority for `org`.

Authority for `net`.

Root name server.

Static routing.
2. physical topology

- In NetKit, each virtual machine corresponds to a directory in the lab.

```bash
user@localhost:~:/mylab$ mkdir client server r1 r2
user@localhost:~:/mylab$ mkdir client server r1 r2
user@localhost:~:/mylab$ mkdir client server r1 r2
user@localhost:~:/mylab$ mkdir client server r1 r2
```

NetKit - [lab: walkthrough]
2. physical topology

- now, we tell netkit about how virtual machines (=network nodes) are interconnected
- this information goes into file `lab.conf`
2. physical topology

Insert this additional statement in the configuration of `ns-org` and `ns-net`:
```bash
options {
  allow-recursion {0/0; };
};
```

R1 uses `ns-org` as local name server; has `client.org` as DNS name.

Server hosts `http://server.net/~guest/`.
2. physical topology

lab.conf

client[0]=A
r1[0]=A
r1[1]=C
ns-org[0]=C
ns-root[0]=C
ns-net[0]=C
r2[0]=B
r2[1]=C
server[0]=B
2. physical topology

- **lab.conf**
  - client[0]=A
  - r1[0]=A
  - r1[1]=C
  - ns-org[0]=C
  - ns-root[0]=C
  - ns-net[0]=C
  - r2[0]=B
  - r2[1]=C
  - server[0]=B

- **r1**’s interface **eth0** is connected to collision domain **A**
- **r1**’s interface **eth1** is connected to collision domain **C**
vademecum

1. create an empty directory where to put the lab
2. set up physical topology
3. set up routing
4. set up additional technologies
   - in this lab:
     - web server
     - dns
3. routing

- The configuration of IP routing consists of two parts:
  - Assignment of IP addresses to network interfaces
    - Achieved using the `ifconfig` command
  - Configuration of static routing
    - Achieved using the `route` command

- All these things are commands that virtual machines must run at startup
  - We put them inside `.startup` files

Tip: these files are very similar to each other, so `copy&paste` is your friend ;-)
3. routing

- **authority for org**
- **root name server**
- **authority for net**

---

**Insert this additional statement in the configuration of ns-org and ns-net:**
```
options {
    allow-recursion {0/0; };
};
```

---

**server**

**uses ns-org as local name server; has client.org as DNS name**

**hosts**
```
http://server.net/~guest/
```

---

**client**

---

**static routing**

---

**root name server**

---

**3. routing**
3. routing

```bash
ifconfig eth0 192.168.0.111 netmask 255.255.255.128 up
go	route add default gw 192.168.0.1 dev eth0
```
3. routing

r1.startup

ifconfig eth0 192.168.0.1 netmask 255.255.255.128 up
ifconfig eth1 10.0.0.1 netmask 255.255.255.0 up
route add -net 192.168.0.128/25 gw 10.0.0.2 dev eth1

each router must learn about non-adjacent networks (only)
3. routing

r2.startup

ifconfig eth0 192.168.0.129 netmask 255.255.255.128 up
ifconfig eth1 10.0.0.2 netmask 255.255.255.0 up
route add -net 192.168.0.0/25 gw 10.0.0.1 dev eth1

each router must learn about non-adjacent networks (only)
3. routing

```
ifconfig eth0 10.0.0.3 netmask 255.255.255.0 up
route add -net 192.168.0.0/25 gw 10.0.0.1 dev eth0
route add -net 192.168.0.128/25 gw 10.0.0.2 dev eth0
```
3. routing

ifconfig eth0 10.0.0.4 netmask 255.255.255.0 up
route add -net 192.168.0.0/25 gw 10.0.0.1 dev eth0
route add -net 192.168.0.128/25 gw 10.0.0.2 dev eth0
3. routing

- ns-net.startup

  ifconfig eth0 10.0.0.5 netmask 255.255.255.0 up

  route add -net 192.168.0.0/25 gw 10.0.0.1 dev eth0
  route add -net 192.168.0.128/25 gw 10.0.0.2 dev eth0
3. routing

server.startup

ifconfig eth0 192.168.0.222 netmask 255.255.255.128 up
route add default gw 192.168.0.129 dev eth0
3. routing

- at this point it is strongly advised to start the lab and check that the routing works

```
user@localhost:~/mylab$ lstart
```

```
client:~# ping 192.168.0.222
```
3. routing

- at this point it is strongly advised to start the lab and check that the routing works
  - if it doesn’t...
    - ...check physical topology (lab.conf)
    - ...check boot-time virtual machine messages (errors printed in blue are relevant)
    - ...check routing tables (route -n)
    - ...

© Computer Networks Research Group Roma Tre  
netkit – [ lab: walkthrough ]  
last update: Nov 2015
vademecum

1. create an empty directory where to put the lab
2. set up physical topology
3. set up routing
4. set up additional technologies
   - in this lab:
     - web server
     - dns
4. web server

- first of all, we need to instruct server to start the web server (apache) at boot time
- since we must set up a user’s web site, we also need to enable apache’s userdir module
- this is achieved by adding lines to server.startup
4. web server

- the module must be enabled before **apache** is started

```sh
ifconfig eth0 192.168.0.222 netmask 255.255.255.128 up
route add default gw 192.168.0.129 dev eth0
a2enmod userdir
/etc/init.d/apache2 start
```
4. web server

- now, we create a simple home page for user guest (the only non-root user that is available by default in netkit)

- according to the default configuration of module userdir, this page must be placed in /home/guest/public_html/index.html

- therefore, we put it in server/home/guest/public_html/index.html
4. web server

hosts
http://server.net/~guest/

server/home/guest/public_html/index.html

<html>
<body>
  Hello!
</body>
</html>
4. web server

- at this point we can start the lab and check that the web server works

```bash
user@localhost:~/mylab$ lstart
```

```bash
client:~# links http://server.net/~guest
```

© Computer Networks Research Group Roma Tre

netkit – [ lab: walkthrough ]
vademecum

1. create an empty directory where to put the lab
2. set up physical topology
3. set up routing
4. set up additional technologies
   ▪ in this lab:
     ▪ dns
     ▪ web server
4. dns

- first of all, we need to instruct some virtual machines to start a name server software (bind) at boot time
- we need to add a line to .startup files
4. dns

ns-org.startup

ifconfig eth0 10.0.0.3 netmask 255.255.255.0 up

route add -net 192.168.0.0/25 gw 10.0.0.1 dev eth0
route add -net 192.168.0.128/25 gw 10.0.0.2 dev eth0
/etc/init.d/bind start
4. dns

ns-root.startup

ifconfig eth0 10.0.0.4 netmask 255.255.255.0 up

route add -net 192.168.0.0/25 gw 10.0.0.1 dev eth0
route add -net 192.168.0.128/25 gw 10.0.0.2 dev eth0

/etc/init.d/bind start

ns-root

ns-org

ns-net

r1

r2

server

client

ns-net

ns-org

ns-root

r1

r2

server

client.startup

lab.conf

ns-net.startup

ns-org.startup

ns-root.startup

r1.startup

r2.startup

server.startup
4. dns

```
ns-net.startup

ifconfig eth0 10.0.0.5 netmask 255.255.255.0 up
route add -net 192.168.0.0/25 gw 10.0.0.1 dev eth0
route add -net 192.168.0.128/25 gw 10.0.0.2 dev eth0
/etc/init.d/bind start
```
4. dns

- now, we need to configure the dns service
- dns configuration consists of some files inside /etc/bind/
  - we create these files inside each virtual machine’s subdirectory
    - ns-org/etc/bind
    - ns-root/etc/bind
    - ns-net/etc/bind
4. dns

- but, hey... dns configuration is rather tricky!

**tip:** copy files from the dns netkit lab and adjust them as needed
4. dns

- download the dns lab from the netkit web site
- copy files from the dns lab to our lab as follows:

<table>
<thead>
<tr>
<th>from (dns lab)</th>
<th>to (our lab)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dnsroot/etc/bind/db.root</td>
<td>ns-root/etc/bind</td>
</tr>
<tr>
<td>dnsroot/etc/bind/named.conf</td>
<td></td>
</tr>
<tr>
<td>dnsorg/etc/bind/db.org</td>
<td>ns-org/etc/bind</td>
</tr>
<tr>
<td>dnsorg/etc/bind/db.root</td>
<td></td>
</tr>
<tr>
<td>dnsorg/etc/bind/named.conf</td>
<td></td>
</tr>
<tr>
<td>dnsnet/etc/bind/db.net</td>
<td>ns-net/etc/bind</td>
</tr>
<tr>
<td>dnsnet/etc/bind/db.root</td>
<td></td>
</tr>
<tr>
<td>dnsnet/etc/bind/named.conf</td>
<td></td>
</tr>
</tbody>
</table>
4. dns

- trim named.conf contents to the essential

```plaintext
zone "." {
  type master;
  file "/etc/bind/db.root";
};
```
4. dns

- trim named.conf contents to the essential and add the requested allow-recursion statement

```
ns-org/etc/bind/named.conf
```

```plaintext
options {
    allow-recursion {0/0; }; 
};
zone "." {
    type hint;
    file "/etc/bind/db.root";
};
zone "org" {
    type master;
    file "/etc/bind/db.org";
};
```

insert this additional statement in the configuration of ns-org and ns-net:
```
options {
    allow-recursion {0/0; }; 
};
```
4. dns

- trim named.conf contents to the essential and add the requested allow-recursion statement

```plaintext	ns-net/etc/bind/named.conf

options {
    allow-recursion {0/0; };
};

zone "." {
    type hint;
    file "/etc/bind/db.root";
};

zone "net" {
    type master;
    file "/etc/bind/db.net";
};
```

insert this additional statement in the configuration of ns-org and ns-net:
```plaintext	ns-org/etc/bind/named.conf

options {
    allow-recursion {0/0; };
};
```

netkit – [ lab: walkthrough ]
4. dns

- configure authoritative information
  - on **ns-root** we just need to update:
    - the address of the root name server
    - the address of the delegated name servers

```text
$TTL 60000 @
IN SOA ROOT-SERVER. root.ROOT-SERVER. (2006031201 ; serial
28800 ; refresh
14400 ; retry
3600000 ; expire
0 ; negative cache ttl
)

@ IN NS ROOT-SERVER. ROOT-SERVER. IN A 10.0.0.4
org. IN NS dnsorg.org. IN A 10.0.0.3
net. IN NS dnsnet.net. IN A 10.0.0.5
```

netkit – [ lab: walkthrough ]
4. dns

- configure authoritative information
  - on `ns-org` we need to:
    - update the address of the authority for `org`
    - add a record for the `client` machine
    - discard all the rest (we have no further delegations in this lab)

```
ns-org/etc/bind/db.org

$TTL 60000
@ IN SOA dnsorg.org. root.dnsorg.org. (2006031201 ; serial 28800 ; refresh 14400 ; retry 3600000 ; expire 0 ; negative cache ttl)

@ IN NS dnsorg.org.

dnsorg IN A 10.0.0.3

client IN A 192.168.0.111

has `client.org` as DNS name
```
4. dns

- configure authoritative information
- on ns-net we need to:
  - update the address of the authority for net
  - add a record for the server machine
  - discard all the rest (we have no further delegations in this lab)

```plaintext
ns-net/etc/bind/db.net

$TTL 60000
@ IN SOA dnsnet.net. root.dnsnet.net. (2006031201 ; serial 28800 ; refresh 14400 ; retry 3600000 ; expire 0 ; negative cache ttl)

@ IN NS dnsnet.net.

server IN A 192.168.0.222
```

© Computer Networks Research Group Roma Tre

netkit – [ lab: walkthrough ]
4. dns

- configure hints
  - on all the (non-root) name servers we need to update the address of the root name server

```plaintext
ns-org/etc/bind/db.root
.
IN NS ROOT-SERVER.
ROOT-SERVER. IN A 10.0.0.4
```

© Computer Networks Research Group Roma Tre

netkit – [ lab: walkthrough ]
4. dns

- configure hints
  - on all the (non-root) name servers we need to update the address of the root name server

```
ns-net/etc/bind/db.root

. IN NS ROOT-SERVER.
ROOT-SERVER. IN A 10.0.0.4
```

netkit – [ lab: walkthrough ]
4. dns

- last, but not least
  - configure a resolver for **client**!
  - in Linux, this goes to /
    `/etc/resolv.conf`, therefore we put it
    in **client/etc/resolv.conf**

```bash
client/etc/resolv.conf

nameserver 10.0.0.3
```
4. dns

- at this point it is a good idea to start the lab and check that the dns works

```
user@localhost:~/mylab$ lstart
```

```
client:~# dig server.net
```

```
4. dns

- at this point it is a good idea to start the lab and check that the dns works
  - if it doesn’t...
    - ...check boot-time virtual machine messages (errors printed in blue are relevant) to see if bind has failed starting
    - ...check /var/log/syslog (that’s where bind logs its errors)
    - ... query for intermediate information (e.g., from client perform an iterative query to get the address of the root name server)
    - ...

netkit – [ lab: walkthrough ]
that’s it!

- after stopping it, the lab can be packed in a tar.gz file for redistribution