



HOW to SET UP a VIRTUAL NETWORK ENVIRONMENT with SOLARIS CONTAINERS

A Guide for Experimenting with Project Crossbow

> **OpenSolaris™** How To Guides



Jeff McMeekin, OpenSolaris Networking Product Manager

Version 1.0 | Last updated: 09/29/09

About This OpenSolaris How To Guide

This OpenSolaris Networking How To Guide provides step-by-step instructions for familiarizing yourself with the Network Virtualization and Network Resource Management features introduced in OpenSolaris 2009.06.

For more information about OpenSolaris networking, see <http://www.opensolaris.com/learn/features/networking/>

Contents

Overview	Page 1
VNIC: Virtual Network Interface	Page 1
Etherstub: Virtual Network Switch	Page 2
Flow: Managed Network Traffic Policy	Page 2
Network Virtualization and Server Virtualization	Page 3
Example 1: VNICs and Flows on a Physical Interface	Page 3
Example 2: Network in a Box	Page 5
For More Information	Page 12
Appendix	Page 13

OpenSolaris Networking How To Guide

Overview

OpenSolaris 2009.06 provides powerful new networking features for network resource management and network virtualization.

- Network virtualization takes server virtualization to the next level - the ability to virtualize entire network topologies of servers, routers, switches, and firewalls all running on a single platform and requiring no additional investment in networking hardware. Network virtualization can be used for a variety of purposes, from prototyping, to developing and testing, to service deployment.
- Network resource management adds the capability to specify for network interfaces
 - Bandwidth limits
 - Traffic priorities
 - CPU assignments for handling network traffic. Resource management properties can be assigned to either physical or virtual network interfaces with no extra performance overhead.

This How To Guide shows how to get started using network virtualization and network resource management in OpenSolaris. Specifically we will look at:

- VNICs: Virtual Network Interface Controllers (VNICs), the fundamental building block of network virtualization. VNICs are created and assigned IP addresses as communication end points.
- Etherstubs: Virtual switches that allow switching traffic within a system.
- Flows: Per link (virtual or physical) management to attain Quality of Service (QoS) goals by setting bandwidth limits and traffic priorities.

VNIC: Virtual Network Interface

VNICs are created on top of physical interfaces or on top of etherstubs, and from the system's point of view are exactly like physical interfaces. VNICs can be plumbed, configured with IP addresses, and given their own TCP options. Tools such as `ifconfig`, `dladm`, and `snoop` all work on VNICs.

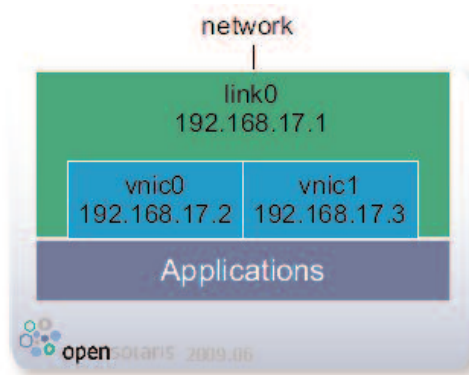


Diagram 1

The diagram shows two VNICs on a single physical interface. Each interface – physical or virtual – has its own IP address. Note that the underlying link does not have to be plumbed or configured.

Etherstub: Virtual Network Switch

Etherstubs are software switches that connect VNICs (shown in the following diagram as the box labeled etherstub0). VNICs that are created on top of the same etherstub can send packets to one another. VNICs on an etherstub cannot send packets directly to the physical network nor to a VNIC on another etherstub, but packets can be routed. So in this example the operating system routes packets between the external link, link0 and the internal switch, ethersrub0.

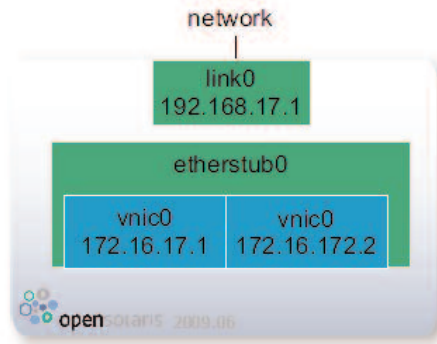


Diagram 2

Note that routing packets between the external link0 and the internal switch etherstub0 is done by the operating system. In a subsequent example we will illustrate how to do this.

Flow: Managed Network Traffic Policy

A flow is created on top of an interface – physical or virtual – and describes network traffic in terms of application (port), protocol, source, and destination, and provides a handle that can be used to implement resource policies for bandwidth and priority.

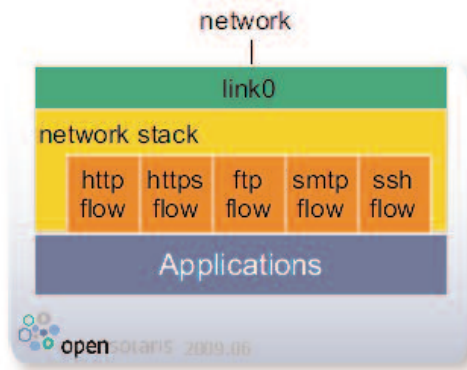


Diagram 3

The diagram shows an interface with several TCP flows: web, secure web, file transfer, email, and secure login. Any traffic that does not match any of the flows passes through the network stack without any policies imposed (other than policies applied to the interface itself). Traffic that does match one of the flows will be limited by the bandwidth and priority assigned to that flow.

Network Virtualization and Server Virtualization

VNICs can be assigned to virtual servers. This document uses Solaris Containers as the underlying virtualization technology, but VNICs also work with the xVM Hypervisor. Containers are a superset of zones as they also include resource management facilities.

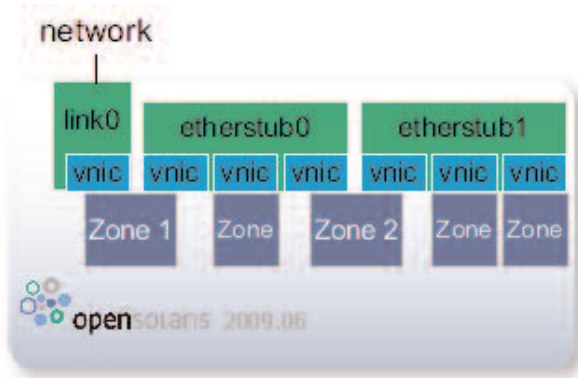


Diagram 4

In the diagram, zones with VNICs on the same etherstub can send packets directly to each other. Zone 1 could route between etherstub0 and the physical network. Zone 2 could route between etherstub0 and etherstub1.

Example 1: VNICs and Flows on a Physical Interface

This example shows how to:

- create a VNIC
- create a flow
- apply bandwidth limits to VNICs and flows
- set priority on a flow

This example sets up a VNIC with its own IP address for a dedicated application, and it sets up several flows on the main interface to manage traffic. The network configuration looks like this:

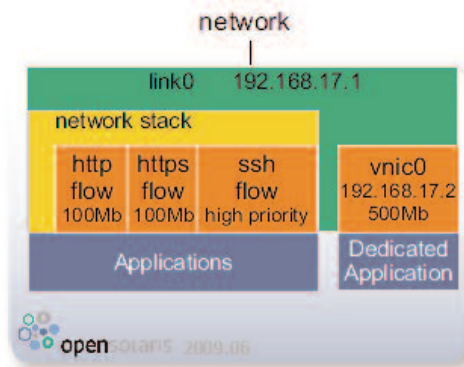


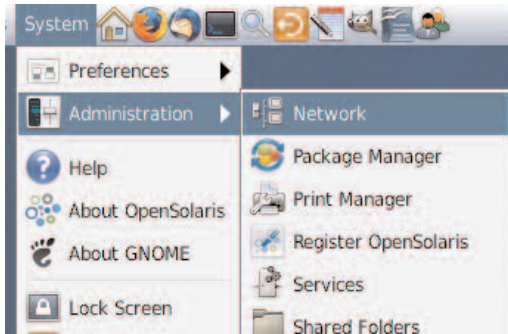
Diagram 5

Two flows and one VNIC have bandwidth limits; one flow has high priority.

Step 1: Disable nwamd

By default, OpenSolaris uses `nwamd`, the Network AutoMagic configuration daemon, to automatically configure the network. For experimenting with network virtualization and network resource management, it is best to disable `nwamd`. This can be accomplished in one of two ways:

i) Select from the menu bar: **System-> Administration-> Network**. This will pop up a window asking if you would like to manually manage your network. Select 'Manual' and then using the network administration tool set up the IP address, mask, routing, etc.



ii) From the console and using the command line tools:

- Run these commands to switch from NWAM to manual configuration:

```
svcadm disable network/physical:nwam
svcadm enable network/physical:default
```

- Configure the network by editing these files:

```
/etc/hostname.<interface>
/etc/defaultrouter
/etc/nsswitch.conf
/etc/resolv.conf
```

- Restart the network service so the configuration takes effect:

```
svcadm restart network/physical:default
```

The reason for running these commands from the console rather than over the network (for example over an ssh connection) is to avoid losing connectivity to the system during reconfiguration.

Step 2: Create the VNIC

```
# dladm show-phys
LINK           MEDIA           STATE    SPEED  DUPLEX    DEVICE
bge0           Ethernet        up       1000   full      bge0
# dladm create-vnic -l bge0 vnic0
# dladm show-vnic vnic0
LINK           OVER            SPEED    MACADDRESS          MACADDRTYPE      VID
vnic0         bge0           1000    2:8:20:5e:69:f2    random            0
# ifconfig vnic0 plumb
# ifconfig vnic0 192.168.17.2/24 up
#
```

Step 3: Create the Flows

```
# flowadm add-flow -l bge0 -a transport=TCP,local_port=80 http-flow
# flowadm add-flow -l bge0 -a transport=TCP,local_port=443 https-flow
# flowadm add-flow -l bge0 -a transport=TCP,local_port=22 ssh-flow
#
```

Step 4: Limit VNIC Traffic; Limit Web (HTTP and HTTPS) Traffic; Make SSH Traffic High Priority

```
# dladm set-linkprop -p maxbw=500M vnic0
# flowadm set-flowprop -p maxbw=100M http-flow
# flowadm set-flowprop -p maxbw=100M https-flow
# flowadm set-flowprop -p priority=high ssh-flow
#
```

Step 5: Review the Configuration

```
# dladm show-linkprop -p maxbw vnic0
LINK          PROPERTY      PERM  VALUE      DEFAULT      POSSIBLE
vnic0         maxbw         rw    500        --           --
# flowadm show-flowprop http-flow
FLOW          PROPERTY      VALUE      DEFAULT      POSSIBLE
http-flow     maxbw         100        --           100M
http-flow     priority      --         --           --
# flowadm show-flowprop https-flow
FLOW          PROPERTY      VALUE      DEFAULT      POSSIBLE
https-flow    maxbw         100        --           100M
https-flow    priority      --         --           --
# flowadm show-flowprop ssh-flow
FLOW          PROPERTY      VALUE      DEFAULT      POSSIBLE
ssh-flow      maxbw         --         --           --
ssh-flow      priority      high       --           high
#
```

Although not shown it is also possible to specify the properties at the time of creation of a VNIC.

Example 2: Network in a Box

This example is a multi-tiered application. There are several web servers running in dedicated Solaris Containers that are exposed to the physical network. There are several database servers running in dedicated Solaris Containers that are not exposed to the physical network. The web servers share a back-end etherstub (virtual switch) called webswitch0 in the following diagram. The database servers also share a backend etherstub, labeled dbswitch0 in the diagram. The network will be configured so that the database containers will have no direct access to the physical network, which securely insulates them from outside communications. The network and zone configuration looks like this:

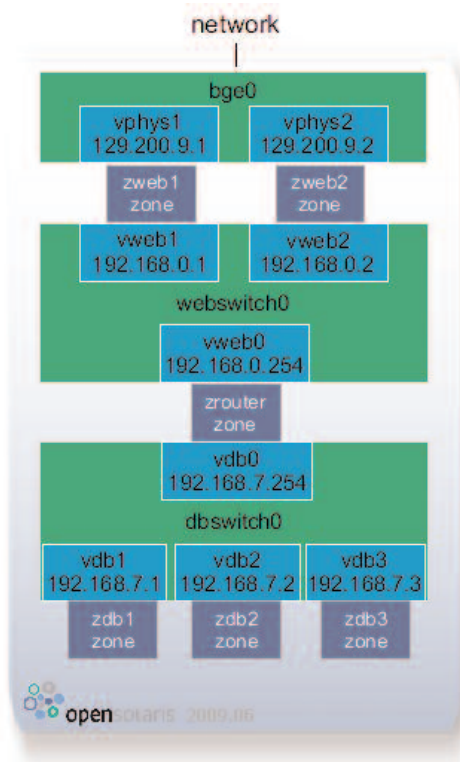


Diagram 6

Step 1: Disable NWAM

See Example 1 above.

Step 2: Create the Etherstubs and VNICs

In this step, create the two virtual switches (etherstubs) and the VNICs used in the six zones, as pictured in Diagram 6. First find out what your physical NIC name is:

```
# dladm show-phys
LINK  MEDIA      STATE  SPEED  DUPLEX  DEVICE
bge0  Ethernet   up     1000   full    bge0
```

For this system, there is only one physical interface, 'bge0'. We will start by creating two virtual NICs on the physical interface, bge0.

```
# dladm create-vnic -l bge0 vphys1
# dladm create-vnic -l bge0 vphys2
```

Then create the virtual switches (etherstubs) and VNICs for the rest of the interfaces shown in Diagram 6:

```
# dladm create-etherstub webswitch0
# dladm create-etherstub dbswitch0
# dladm create-vnic -l webswitch0 vweb0
# dladm create-vnic -l webswitch0 vweb1
# dladm create-vnic -l webswitch0 vweb2
# dladm create-vnic -l dbswitch0 vdb0
# dladm create-vnic -l dbswitch0 vdb1
# dladm create-vnic -l dbswitch0 vdb2
# dladm create-vnic -l dbswitch0 vdb3
```

Check the setup:

```
# dladm show-vnic
```

LINK	OVER	SPEED	MACADDRESS	MACADDRTYPE	VID
vphys1	bge0	100	2:8:20:d1:b6:26	random	0
vphys2	bge0	100	2:8:20:ae:69:f7	random	0
vweb0	webswitch0	0	2:8:20:ae:89:e0	random	0
vweb1	webswitch0	0	2:8:20:77:c6:1	random	0
vweb2	webswitch0	0	2:8:20:51:35:ec	random	0
vdb0	dbswitch0	0	2:8:20:b1:85:b1	random	0
vdb1	dbswitch0	0	2:8:20:d7:6f:df	random	0
vdb2	dbswitch0	0	2:8:20:22:38:2e	random	0
vdb3	dbswitch0	0	2:8:20:f5:b5:ce	random	0

Step 3: Create the ZFS Pool for Storing the Zones

We are going to create a single Solaris Zone and then, to reduce the time to create additional ones, we will clone the first. Cloned zones require sharing a ZFS dataset, so creating a ZFS dataset is the first step.

```
# zfs create -o mountpoint=/zonefs rpool/zonefs
# chmod 700 /zonefs
```

Step 4: Create a Zone for Cloning

OpenSolaris is a constantly evolving operating system out of which the next major release of Solaris will be derived. As such, at any given time some parts of it may be undergoing transitions. A good example of this is Solaris Zones. Because of the new packaging system for OpenSolaris, the implementation of Solaris Zones is still under development. Under Solaris 10, creation of zones takes a few seconds. The current version of zones under OpenSolaris takes a couple of minutes to create a zone because in the current implementation, zone creation requires connection to the package repository.

Since we are creating 6 zones, the more efficient approach is to create one from scratch, and then make 6 clones. Cloning takes only a few seconds.

```
# zonecfg -z zclone
zclone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zclone> create
zonecfg:zclone> set zonepath=/zonefs/zclone
zonecfg:zclone> set ip-type=exclusive
zonecfg:zclone> verify
zonecfg:zclone> commit
zonecfg:zclone> exit
# zoneadm -z zclone install
```

The 'install' step can take several minutes while packages are installed. You may see some errors from sys-unconfig like:

```
rm: /zonefs/zweb1/root/etc/vfstab.sys-u: No such file or directory
grep: can't open /zonefs/zweb1/root/etc/dumpadm.conf
```

which may be ignored. Now boot the clone zone:

```
# zoneadm -z zclone boot ; zlogin -C zclone
```

We put the zlogin on the same line so that the command will be immediately executed and you will see the zclone zone boot including the initial configuration of the SMF services. You want this step to complete so that when you clone this zone you won't have to wait for each clone to configure SMF services.

You know it is safe to halt the clone when you are prompted to select a terminal type. Don't select one. Instead enter ~. to escape to the global zone,

```
Type the number of your choice and press Return: ~.
```

and then halt zclone.

```
# zoneadm -z zclone halt
```

Step 5: Create an Example Zone

In Diagram 6 we see 6 zones. Ordinarily setting up 6 zones would take considerable typing.

The approach we will take is to walk through setting up one zone as an example and then suggest that you use some scripts that are provided in the appendix to automate the process.

We are going to create one of the database zones, zdb1, i.e. one of the zones which would contain a database in our example network. We aren't actually going to provision with a database server, since this example is only about setting up the networking infrastructure.

First we get the configuration information from the clone zone:

```
# zonecfg -z zclone export | zonecfg -z zdb1 -f -
```

Then designate where to set up the zone files, add the VNIC, (vdb1) and create the zdb1 zone:

```
# zonecfg -z zdb1 "set zonepath=/zonefs/zdb1"
# zonecfg -z zdb1 "add net;set physical=vdb1;end"
# zoneadm -z zdb1 clone zclone
```

Note: As when you created zclone, ignore the errors from sys-unconfig.

For ZFS-based zones, you have to 'ready' them to see the zone's file system from the global zone. Try exploring /zonefs/zdb1 before and after executing the following command.

```
# zoneadm -z zdb1 ready
```

Now we need to configure zdb1 and give it a network personality. As you saw above when you created the zclone zone, when you zlogin'ed to zclone, you were asked to specify a terminal type. If you had answered, you would have gone through the entire sysidtool configuration process (see sysidtool(1M) man page for details). In other words you would have

been asked for hostname, IP address, name services, time zone location, all the information for which you are prompted after doing a `sys-unconfig` on a Solaris 10 system.

The good news is that you can set up a configuration file, `sysidcfg`, in advance, and copy it into the zone so that you won't be asked any questions. The less good news is that this is another area where a change in implementation is coming and what you see in OpenSolaris 2009.06 around `sysidtool` will change. For OpenSolaris 2009.06, not all the features of the Solaris 10 version of the `sysidcfg` files are supported. One example is the `root_password` property. OpenSolaris uses a different password encryption mechanism that `sysidtool` does not recognize. Fortunately OpenSolaris can process passwords generated on earlier versions of Solaris 10. So the work around is to generate the `root_password` encryption on a Solaris 10 system, and cut and paste that into the input for the configuration file. In this case the password encrypted string corresponds to the plain text "crossbow".

In this next step we will create the `sysidcg` file for pre-configuring the `zdb1` zone so that when we login into the zone the first time, we won't be prompted to enter any configuration information.

You will want to create the file `/zonefs/zdb1/root/etc/sysidcfg` which contains the following lines:

```
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=rJmv5LUXM10cU
network_interface=PRIMARY {
hostname=zdb1
ip_address=192.168.7.1
netmask=255.255.255.0
protocol_ipv6=no
default_route=none
}
name_service=NONE
```

Boot the zone `zdb1` and check that it is running.

```
# zoneadm -z zdb1 boot
# zoneadm list -cv zdb1
```

ID	NAME	STATUS	PATH	BRAND	IP
0	global	running	/	native	shared
78	zdb1	running	/zonefs/zdb1	ipkg	excl
-	zclone	installed	/zonefs/zclone	ipkg	excl

Step 6: Create the Rest of the Zones

As you can see from the above, generating the configuration takes quite a bit of typing. To simplify creating the zones, there are scripts for both creating and cleaning up zones in the Appendix. You should copy and paste those scripts into the appropriate file names. See the Appendix for more instructions.

Assuming you put the scripts in `/opt/CrossbowHowTo`, then you should see them all with:

```
# cd /opt/CrossbowHowTo/scripts
# ls
cleanupallzones.sh      cleanupzone.sh          createzone-2vnic.sh
cleanupdladm.sh         createzone-1vnic.sh
```

To start, you already created `zdb1` in Step 5 so now create the other two.

Create `zdb2` and `zdb3` zones:

```
# sh -x createzone-1vnic.sh zdb2 vdb2 zdb2 192.168.7.2
# sh -x createzone-1vnic.sh zdb3 vdb3 zdb3 192.168.7.3
```

Run the scripts with the `'-x'` flag to the shell to print out the commands as they are executed. That makes it a bit easier to follow along. You will see some errors as you did in Step 5, from `sys-unconfig`

```
rm: /zonefs/zrouter/root/etc/vfstab.sys-u: No such file or directory
grep: can't open /zonefs/zrouter/root/etc/dumpadm.conf
```

which you can ignore.

Now create the router zone. To keep the example simple we are going to use the default Solaris router for handling routing tasks. If the routing had been more complex or we wanted to use other routing protocols, we could have used the open source `quagga` project, which can be found in the OpenSolaris package repository. See `quagga(8)`.

Note the router zone has two VNICs so we use a different script. You will create the router zone here, and configure the routing in it later.

```
# sh -x createzone-2vnics.sh zrouter vweb0 zrouter 192.168.0.254
vdb0 zrouter-vdb0 192.168.7.254
```

And finally, create the two zones in which the web servers would be run.

Create `zweb1` and `zweb2` zones. You will need to replace the `129.200.9` addresses with addresses on your subnet, since the `vphys1` and `vphys2` addresses are seen by external systems.

```
# sh -x createzone-2vnics.sh zweb1 vphys1 zweb1 129.200.9.1
vweb1 zweb1-vweb 192.168.0.1
# sh -x createzone-2vnics.sh zweb2 vphys2 zweb2 129.200.9.2
vweb2 zweb2-vweb 192.168.0.2
```

We won't actually install a web server. We only illustrate in this example how to set up the networking for this arrangement.

Finally verify the setup:

```
# zoneadm list -cv
ID NAME          STATUS      PATH                      BRAND  IP
0  global         running    /                          native shared
78 zdb1           running    /zonefs/zdb1             ipkg   excl
81 zdb2           running    /zonefs/zdb2             ipkg   excl
82 zdb3           running    /zonefs/zdb3             ipkg   excl
83 zrouter        running    /zonefs/zrouter          ipkg   excl
84 zweb1          running    /zonefs/zweb1            ipkg   excl
85 zweb2          running    /zonefs/zweb2            ipkg   excl
-  zclone         installed  /zonefs/zclone           ipkg   excl
```

Step 7: Set up Routing for the Zones

Now you must log in and specify some default routing information for each zone.

Always log into a zone the first time using the `-C` flag. This will invoke `sysidtool` to prompt you for input. You provided this information prior to booting via the `sysidcfg` file, so you shouldn't be prompted. But if you made any changes to the `sysidcfg` file contents, you may trigger the prompting.

```
# zlogin -C zweb1
[Connected to zone 'zweb1' console]
```

If you don't see the following login prompt, press the return key:

```
zweb1 console login: root
Password: crossbow
Last login: Wed Sep  9 14:52:56 on console
Sun Microsystems Inc.   SunOS 5.11       snv_111b November 2008
root@zweb1:~#
root@zweb1:~# route -p add net 192.168.7.0/24 192.168.0.254
root@zweb1:~# logout
zweb1 console login:~.
```

Again, we use `“~.”` to exit `zlogin` and return to the global zone.

Next `zlogin` in to `zweb2` and specify the same routing:

```
root@zweb2:~# route -p add net 192.168.7.0/24 192.168.0.254
```

Now `zlogin` to each of the database zones and provide the route to the `192.168.0` network:

```
root@zdb1:~# route -p add net 192.168.0.0/24 192.168.7.254
```

```
root@zdb2:~# route -p add net 192.168.0.0/24 192.168.7.254
```

```
root@zdb3:~# route -p add net 192.168.0.0/24 192.168.7.254
```

Finally configure `rrouter`, which routes packets between the webserver zones and the database zones. However before doing this, try `zlogin` to `zweb1` and

```
# ping 192.168.7.254
```

which should work and

```
# ping 192.168.7.1
```

which should not work because you haven't told the `rrouter` zone to forward packets between the `192.168.7` and `192.168.0` networks.

Next set up `ip forwarding` in the `rrouter` zone:

```
# zlogin -C rrouter
.
.
# svcadm enable network/ipv4-forwarding:default
```

You can now try various experiments by using the ping or traceroute commands between various zones:

- Try to reach zweb1 or zweb2 from an external system on the same subnet as those addresses (129.200.1 in this example). You should succeed.
- Try to reach one of the IP addresses on the 192.168.0 or 192.168.7 networks from an external system and you should not be successful because IP forwarding is not on for zweb1 or zweb2.
- Try to reach one of the database zones, zdb1, zdb2, zd3 (192.168.7.1, 2, or 3) from zweb1 or zweb2, and because IP forwarding is turned on for zrouter, you should be successful.
- And try the opposite- from zdb1, zdb2, or zdb 3, try to reach 192.168.0.1 or 2. Again because of the zrouter configuration you should succeed.

Clean Up

Finally, see the appendix for three scripts to clean up the zones and the VNICs and restore the system to it's original state.

```
# sh cleanupallzones.sh
# sh cleanupdladm.sh
# zfs destroy -r rpool/zonefs
```

Conclusion

Hopefully this network-in-a-box How To Guide will give you some ideas for future experimentation. With basic OpenSolaris capabilities you can easily set up fairly complex environments. And using additional facilities not discussed here but available through the OpenSolaris repository, like IP Filter firewall and Quagga routing package you can address very complex networking requirements.

For More Information

Much of this How To Guide concerns the details of creating zones so you may find the Solaris 10 Containers How To Guide useful. See http://www.sun.com/software/solaris/howto_guides.jsp.

There is a Crossbow demonstration graphical software package available that allows you to drag and drop to create configurations as used in this guide, and set properties, like bandwidth, for the VNICs you create. See www.opensolaris.org/os/project/crossbow.

Appendix

You will find four scripts in this appendix. The recommendation is to cut and paste their contents into the suggested file names using `gedit` to create the files on OpenSolaris.

Following is the first script, `createzone-lvnic.sh` for creating a zone with one VNIC.

```
#!/bin/sh
#
# FILENAME:   createzone-lvnic.sh
#
# Usage:
# createzone-lvnic.sh <zonename> <vnic1> <hostname for vnic1> <ip address for vnic1>

if [ $# != 4 ]
then
    echo "Usage: createzone-lvnic.sh <zonename> <vnic1> <hostname for vnic1>
<ip addr for vnic1>"
    exit 1
fi
ZONENAME=$1
VNIC1=$2
VNIC1HOSTNAME=$3
VNIC1IP=$4

echo "Create zone $ZONENAME" with
echo "  Interface $VNIC1, hostname $VNIC1HOSTNAME at IP Address $VNIC1IP"

zonecfg -z zclone export | zonecfg -z $ZONENAME -f -
zonecfg -z $ZONENAME "set zonepath=/zonefs/$ZONENAME"
# add the VNIC
zonecfg -z $ZONENAME "add net;set physical=$VNIC1;end"

# now create the new zone from the clone
zoneadm -z $ZONENAME clone zclone
#
# For ZFS based zones, you have to 'ready' them to see the zone's file
# system from the global zone.
#
zoneadm -z $ZONENAME ready

# Note, you can not generate a root password under OpenSolaris
# and use that encrypted string for the root_password property.
# The work around is to generate a password on Solaris 10 instead,
# and use that encryption (from /etc/shadow) for the root_password
# setting.

cat > /zonefs/$ZONENAME/root/etc/sysidcfg << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=rJmv5LUXM10cU
network_interface=PRIMARY {
hostname=$VNIC1HOSTNAME
ip_address=$VNIC1IP
netmask=255.255.255.0
protocol_ipv6=no
default_route=none
}
name_service=NONE
_EOF_

zoneadm -z $ZONENAME boot

#END FILE createzone-lvnic.sh
```

Following is the second script, createzone-2vnics.sh, for creating a zone with 2 VNICS.

```
#!/bin/sh
#
# FILENAME:   createzone-2vnics.sh
#
# Usage:
# create2zones.sh <zonename> <vnic1> <hostname for vnic1> <ip address for vnic1>
# <vnic2> <hostname for vnic2> <ip address for vnic2>

if [ $# != 7 ]
then
    echo "usage: createzone-2vnics.sh <zonename> <vnic1> <hostname for vnic1>
<ip address for vnic1> <vnic2> <hostname for vnic2> <ip address for vnic2>"
    exit 1
fi
ZONENAME=$1
VNIC1=$2
VNIC1HOSTNAME=$3
VNIC1IP=$4
VNIC2=$5
VNIC2HOSTNAME=$6
VNIC2IP=$7

echo "Create zone $ZONENAME" with
echo "  Interface $VNIC1, hostname $VNIC1HOSTNAME at IP Address $VNIC1IP"
echo "  Interface $VNIC2, hostname $VNIC2HOSTNAME at IP Address $VNIC2IP"

zonecfg -z zclone export | zonecfg -z $ZONENAME -f -
zonecfg -z $ZONENAME "set zonepath=/zonefs/$ZONENAME"
# add the two VNICS
zonecfg -z $ZONENAME "add net;set physical=$VNIC1;end"
zonecfg -z $ZONENAME "add net;set physical=$VNIC2;end"
# now create the new zone from the clone
zoneadm -z $ZONENAME clone zclone

# For ZFS based zones, you have to 'ready' them to see the zone's
# file system from the global zone.

zoneadm -z $ZONENAME ready

# Note, you can not generate a root password under OpenSolaris
# and use that encrypted string for the root_password property.
# The work around is to generate a password on Solaris 10 instead,
# and use that encryption (from /etc/shadow) for the root_password
# setting.
cat > /zonefs/$ZONENAME/root/etc/sysidcfg << _EOF_
terminal=vt100
system_locale=C
timezone=US/Pacific
nfs4_domain=dynamic
security_policy=NONE
root_password=rJmv5LUXM10cU
network_interface=$VNIC2 {
hostname=$VNIC2HOSTNAME
ip_address=$VNIC2IP
netmask=255.255.255.0
protocol_ipv6=no
default_route=none
}
network_interface=$VNIC1 {
primary
hostname=$VNIC1HOSTNAME
ip_address=$VNIC1IP
netmask=255.255.255.0
protocol_ipv6=no
default_route=none
}
name_service=NONE
_EOF_

zoneadm -z $ZONENAME boot

# END FILE   createzone-2vnics.sh
```

Following is the third script, cleanupzone.sh. This can be used to completely remove a zone.

```
#!/bin/sh
#
# FILENAME:  cleanupzone.sh
#
# Usage: cleanupzone.sh <zone name>
#
# This will completely remove a zone from the system
#
if [ $# != 1 ]
then
    echo "Usage: cleanupzone <zone name>"
    exit 1
fi
echo 'zoneadm -z '$1' halt'
zoneadm -z $1 halt
echo 'zoneadm -z '$1' uninstall -F'
zoneadm -z $1 uninstall -F
echo 'zonecfg -z '$1' delete -F'
zonecfg -z $1 delete -F

#
# END FILE cleanupzone.sh
```

This fourth script can be used to remove all the zones.

```
#!/bin/sh
#
# FILENAME:  cleanupallzones.sh
#
# Usage: cleanupallzones.sh
#
# This will completely remove all the crossbow demo zones
#
sh cleanupzone.sh zweb1
sh cleanupzone.sh zweb2
sh cleanupzone.sh zrouter
sh cleanupzone.sh zdb1
sh cleanupzone.sh zdb2
sh cleanupzone.sh zdb3
sh cleanupzone.sh zclone

#
# END FILE cleanupallzones.sh
```

The fifth script is for after removing all the zones from a system. Use this script to remove all the VNICs and etherstubs that you created.

```
#!/bin/sh
#
# FILENAME:  cleanupdladm.sh
#
# Usage: cleanupdladm
#
# This will completely remove the crossbow network-in-a-box
# VNICs and etherstubs
#
dladm delete-vnic vphys1
dladm delete-vnic vphys2
dladm delete-vnic vweb0
dladm delete-vnic vweb1
dladm delete-vnic vweb2
dladm delete-vnic vdb0
dladm delete-vnic vdb1
dladm delete-vnic vdb2
dladm delete-vnic vdb3
dladm delete-etherstub webswitch0
dladm delete-etherstub dbswitch0

#
# END FILE cleanupdladm.sh
```

opensolaris.com

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 1-650-960-1300 or 1-800-555-9SUN Web sun.com



©2009 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Solaris and OpenSolaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.
09/09