NAME

fanatic – fan bridge configuration and test wizard

SYNOPSIS

fanatic [<cmd> [<arg> ...]]
fanatic configure
fanatic deconfigure
fanatic test

DESCRIPTION

fanatic is the configuration and test wizard for Fan networking. It is designed to simplify the process of configuring persistent Fan bridges. It will also configure both LXD and Docker to use those bridges as required.

It also can be used to configure Fan networking without any user interaction (see NON-INTERACTIVE USAGE below).

A Fan network is a mechanism for expanding the range of IP addresses available to a system. It is most useful for containers systems such as Docker and LXC/LXD, but it can be used in other contexts as well.

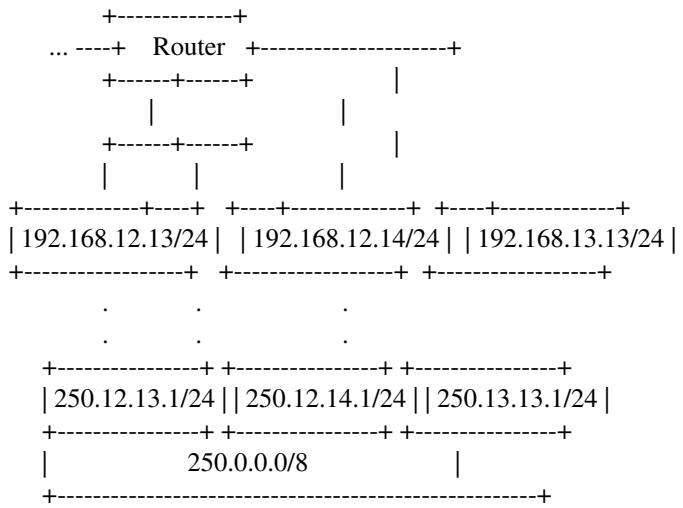
TERMINOLOGY

The Fan mapping is defined by a combination of an **underlay** and an **overlay** address. Each of these is defined in CIDR network address form. The overlay prefix defines the overall network prefix for the overlay slice. The underlay prefix defines the prefix bits to strip to obtain the slice identifier. This is concatenated with the overlay prefix to calculate the **overlay slice** which is the range of the Fan network that is assigned to a specific single address within the underlay network. This way each address within a 192.168.0.0/16 underlay, mapped to a 250.0.0.0/8 overlay would cover the range:

250.x.y.0/(8 + (32-16) = 24)

Thus 8 (32-24) address bits will equate to 254 additional addresses for each address mapped from the underlay network (you cannot use the first and last address within an IPv4 subnet).

The following diagram shows a more advanced example with three hosts which will be joined in a single Fan network. Two of the hosts are in the same subnet (192.168.12.0/24) and the other in a separate subnet (192.168.13.0/24). Since all three will share the same overlay network (250.0.0.0/8) we would use an underlay of 192.168.0.0/16. The choice of the underlay prefix depends on the network layout which is mapped into an overlay network (which overlay prefix is used does not matter). So, if we really only were interested in joining 192.168.12/13.0/24, we could use an underlay of 192.168.12.0/23 which would maximize the number of addresses per slice but the CIDR network addresses of the slices would be less obvious to read.



For the example above, a /16 over /8 layout was chosen because that moves numbers in the dotted decimal

notation in a way that is simpler to read.

The overlay prefix is 8 bits. The underlay prefix takes the lower 16 bits of each host's physical interface and shifts those left until hitting the least significant bit of the overlay (32-16-8=8). The result is, each interface of the host that is part of the Fan overlay gets a /24 (=8+16) slice of the Fan network. The 250.x.y.1 address is assigned to the fan-250 bridge on each host and would act as the **gateway** address of any container that runs within the overlay slice.

COMMAND SYNTAX

Running **fanatic** with no arguments is equivalent to running **fanatic configure**.

fanatic [configure]
Interactively configure the local system for Fan. This action will identify the interface to be used and offer to configure an appropriate Fan over the addresses on that interface. It will further offer to configure LXD and/or Docker to use this interface. Finally, it will offer to test the configuration as applied both locally and with an optional remote host.

fanatic deconfigure
Reverses the interactive configuration above. It will deconfigure both LXD and Docker (if configured) and then deconfigure the Fan. These steps are only performed if the configuration was generated by **fanatic**.

fanatic test
Runs the testing phase of the interactive setup. This action enables retesting of a previously configured system. Only those pieces configured will be tested.

fanatic help
Shows basic help including usage information and pointers to further help on use of **fanatic**.

NON-INTERACTIVE USAGE

It is possible to run any sub-phase of the configure, deconfigure or test phases, which are done by **fanatic** in interactive mode, individually and without user interaction:

fanatic enable-fan -u <underlay> -o <overlay>
Configure a Fan over the specified interface.

fanatic disable-fan -u <underlay> -o <overlay>
Deconfigure a Fan over the specified interface.

fanatic enable-lxd -u <underlay> -o <overlay>
Configure LXD for use on the Fan over the specified interface.

fanatic disable-lxd -u <underlay> -o <overlay>
Deconfigure LXD for use on the Fan over the specified interface.

fanatic enable-docker -u <underlay> -o <overlay>
Configure docker for use on the Fan over the specified interface.

fanatic disable-docker -u <underlay> -o <overlay>
Deconfigure docker for use on the Fan over the specified interface.

test-host -u <underlay> -o <overlay> -r <remote host IP>
Test host-to-host Fan functionality. This requires you to be running the same test script on both hosts in the test.

test-local-lxd -u <underlay> -o <overlay>
Test the local LXD configuration by creating a container and running host-to-host testing against that container.

test-local-docker -u <underlay> -o <overlay>
Test the local Docker configuration by creating a container and running host-to-host testing against that container.

USE WITH LXC/LXD

Once the Fan bridges are configured (and LXD is installed), LXC/LXD can be configured to use a specific Fan bridge for the network device of containers. This can be done by either modifying the default template or by creating a separate template which can be used when creating containers. The latter is what **fanatic enable-lxd** does internally. The same could be done manually by setting link and MTU as shown below:

```
lxc.network.link = <Fan bridge name>
lxc.network.mtu = 1450
```

USE WITH DOCKER

Once the Fan bridges are configured (and docker installed), a docker network must be configured (similar to the LXD template). New containers can then be configured to use the Fan network. This is how the **fanatic enable-docker** call internally works. The new network will be a bridge network type named after the name of the Fan bridge which is to be used. If this were done manually, the docker command would look like:

```
# sudo docker network create --driver bridge \
    -o "com.docker.network.bridge.name=<bridge name>" \
    -o "com.docker.network.driver.mtu=1450" \
    --subnet <overlay> --ip-range <slice subnet> \
    --gateway <slice host> \
    --aux-address "reserved0=<slice host>" \
    <bridge name>
```

This defines the complete Fan overlay network to be accessed through the Fan bridge given and limits the scope of addresses used for containers to a given **slice subnet** (which is the CIDR notation of the Fan overlay slice that is assigned to the given Fan bridge). All addresses, within that range, which are not to be used need to be declared individually. Usually this is the **slice host** which is the address of the host within the overlay slice (and assigned to the Fan bridge). For example, if the overlay network were a 250.0.0.0/8 and the address of the host in the underlay address space were u.u.x.y/16 the Fan bridge would be called fan-250 and have a 250.x.y.1/24 assigned to it which would be declared as the gateway for the 250.x.y.0/24 overlay slice.

LIMITATIONS

Since docker (at the time of writing) does not support to allow an external DHCP service to be used, there is no way to safely mix docker and LXD containers in the same Fan overlay slice. Both types of containers can be mixed though in the same Fan overlay, as long as they use their individual slice.

EXAMPLES

Assuming that a Fan network should be attached to the primary interface of the host without any user interaction, the first step is to figure out which is the default interface:

```
# ip route show | awk '$1 == "default"{print $NF}'
ens3
```

Next we need to find out what IPv4 address this interface is configured to:

```
# ip address show ens3 | awk '$1 == "inet"{print $2}'
192.168.2.120/24
```

If all hosts which participate in the Fan network are located in 192.168.2.0/24, then we can use a /24 underlay address and get 65535 addresses per slice instead of the 254 if we had to include hosts from the 192.168.0.0/16 range.

For all of the following commands of this example we use the address of the interface with a prefix of 20 which assumes we only care for hosts in 192.168.[0-15].0/24 and results in 4095 addresses per slice. Using the interface address tells fanatic to only configure this single interface.

```
# sudo fanatic enable-fan -u 192.168.2.120/20 -o 250.0.0.0/8
configuring fan underlay:192.168.2.120/20 overlay:250.0.0.0/8

# sudo fanatic enable-lxd -u 192.168.2.120/20 -o 250.0.0.0/8
configuring LXD for underlay:192.168.2.120/20 \
```

```
  overlay:250.0.0.0/8 (fan-250)
Profile fan-250 created

# sudo fanatic test-local-lxd -u 192.168.2.120/20 -o 250.0.0.0/8
local lxd test: creating test container (Ubuntu:lts) ...
Creating fanatic-test
Starting fanatic-test
lxd test: Waiting for addresses on eth0 ...
lxd test: Waiting for addresses on eth0 ...
lxd test: Waiting for addresses on eth0 ...
test master: ping test (250.39.132.96) ...
test slave: ping test (250.39.128.1) ...
test master: ping test ... PASS
test master: short data test (250.39.128.1 -> 250.39.132.96) ...
test slave: ping test ... PASS
test slave: short data test (250.39.132.96 -> 250.39.128.1) ...
test slave: short data ... PASS
test master: short data ... PASS
test master: long data test (250.39.128.1 -> 250.39.132.96) ...
test slave: long data test (250.39.132.96 -> 250.39.128.1) ...
test master: long data ... PASS
test slave: long data ... PASS
local lxd test: destroying test container ...
local lxd test: test complete PASS (master=0 slave=0)
```

SEE ALSO

fanctl(8), **/usr/share/doc/ubuntu-fan/README**

AUTHOR(s)

Andy Whitcroft <apw@canonical.com>,
Stefan Bader <stefan.bader@canonical.com>