Using the Katzenpost Docker test network

Table of Contents

Katzenpost provides a ready-to-deploy Docker image for developers who need a non-production test environment for developing and testing client applications and server side plugins. By running this image on a single computer, you avoid the need to build and manage a complex multi-node mix net. The image can also be run using Podman [https://podman.io/]

The test mix network includes the following components:

- Three directory authority (PKI [https://katzenpost.network/docs/specs/pki.html]) nodes
- Six mix [https://katzenpost.network/docs/specs/mixnet/] nodes, including one node serving also as both gateway and service provider
- A ping utility, **run-ping**

Requirements

Before running the Katzenpost docker image, make sure that the following software is installed.

- A Debian GNU Linux [https://debian.org] or Ubuntu [https://ubuntu.com] system
- Git [https://git-scm.com/]
- Go [https://go.dev/]
- GNU Make [https://www.gnu.org/software/make/]
- Prometheus [https://prometheus.io/docs/introduction/overview/]
- Docker [https://www.docker.com], Docker Compose [https://docs.docker.com/compose/], and (optionally) Podman [https://podman.io]

Note

If both Docker and Podman are present on your system, Katzenpost uses Podman. Podman is a drop-in daemonless equivalent to Docker that does not require superuser privileges to run.

On Debian, these software requirements can be installed with the following commands (running as superuser). **Apt** will pull in the needed dependencies.

```
# apt update
```

```
# apt install git golang make docker docker-compose podman
```

Preparing to run the container image

Complete the following procedure to obtain, build, and deploy the Katzenpost test network.

1. Install the Katzenpost code repository, hosted at `https://github.com/katzenpost`. The main Katzenpost repository contains code for the server components as well as the docker image. Clone the repository with the following command (your directory location may vary):

   ```
   ~$ git clone https://github.com/katzenpost/katzenpost.git
   ```

2. Navigate to the new `katzenpost` subdirectory and ensure that the code is up to date.

   ```
   ~$ cd katzenpost
   ~/katzenpost$ git checkout main
   ~/katzenpost$ git pull
   ```

3. (Optional) Create a development branch and check it out.

   ```
   ~/katzenpost$ git checkout -b devel
   ```

4. (Optional) If you are using Podman, complete the following steps:

   1. Point the DOCKER_HOST environment variable at the Podman process.

      ```
      $ export DOCKER_HOST=unix:///var/run/user/$(id -u)/podman/podman.sock
      ```

   2. Set up and start the Podman server (as superuser).

      ```
      $ podman system service -t 0 $DOCKER_HOST &
      $ systemctl --user enable --now podman.socket
      ```

Operating the test mixnet

Navigate to `katzenpost/docker`. The `Makefile` contains target operations to create, manage, and test the self-contained Katzenpost container network. To invoke a target, run a command with the using the following pattern:

```
~/katzenpost/docker$ make target
```

Running **make** with no target specified returns a list of available targets.

Table 1. Table 1: Makefile targets

[none]	Display this list of targets.
start	Run the test network in the background.
stop	Stop the test network.
wait	Wait for the test network to have consensus.
watch	Display live log entries until **Ctrl-C**.
status	Show test network consensus status.
show-latest-vote	Show latest consensus vote.
run-ping	Send a ping over the test network.
clean-bin	Stop all components and delete binaries.

clean-local	Stop all components, delete binaries, and delete data.
clean-local-dryrun	Show what clean-local would delete.
clean	Same as **clean-local**, but also deletes `go_deps` image.

Starting and monitoring the mixnet

The first time that you run **make start**, the Docker image is downloaded, built, installed, and started. This takes several minutes. When the build is complete, the command exits while the network remains running in the background.

```
~/katzenpost/docker$ make start
```

Subsequent runs of **make start** either start or restart the network without building the components from scratch. The exception to this is when you delete any of the Katzenpost binaries (dirauth.alpine, server.alpine, etc.). In that case, **make start** rebuilds just the parts of the network dependent on the deleted binary. For more information about the files created during the Docker build, see the section called “Network topology and components”.

Note

When running **make start** , be aware of the following considerations:

- If you intend to use Docker, you need to run **make** as superuser. If you are using **sudo** to elevate your privileges, you need to edit `katzenpost/docker/Makefile` to prepend **sudo** to each command contained in it.
- If you have Podman installed on your system and you nonetheless want to run Docker, you can override the default behavior by adding the argument **docker=docker** to the command as in the following:

  ```
  ~/katzenpost/docker$ make run docker=docker
  ```

After the **make start** command exits, the mixnet runs in the background, and you can run **make watch** to display a live log of the network activity.

```
~/katzenpost/docker$ make watch
    ...
    <output>
    ...
```

When installation is complete, the mix servers vote and reach a consensus. You can use the **wait** target to wait for the mixnet to get consensus and be ready to use. This can also take several minutes:

```
~/katzenpost/docker$ make wait
    ...
    <output>
    ...
```

You can confirm that installation and configuration are complete by issuing the **status** command from the same or another terminal. When the network is ready for use, **status** begins returning consensus information similar to the following:

```
~/katzenpost/docker$ make status
    ...
    00:15:15.003 NOTI state: Consensus made for epoch 1851128 with 3/3 signatur
    ...
```

Testing the mixnet

At this point, you should have a locally running mix network. You can test whether it is working correctly by using **run-ping**, which launches a packet into the network and watches for a successful reply. Run the following command:

```
~/katzenpost/docker$ make run-ping
```

If the network is functioning properly, the resulting output contains lines similar to the following:

```
19:29:53.541 INFO gateway1_client: sending loop decoy
    !19:29:54.108 INFO gateway1_client: sending loop decoy
    19:29:54.632 INFO gateway1_client: sending loop decoy
    19:29:55.160 INFO gateway1_client: sending loop decoy
    !19:29:56.071 INFO gateway1_client: sending loop decoy
    !19:29:59.173 INFO gateway1_client: sending loop decoy
    !Success rate is 100.000000 percent 10/10)
```

lf **run-ping** fails to receive a reply, it eventually times out with an error message. If this happens, try the command again.

Note

If you attempt use **run-ping** too quickly after starting the mixnet, and consensus has not been reached, the utility may crash with an error message or hang indefinitely. If this happens, issue (if necessary) a **Ctrl-C** key sequence to abort, check the consensus status with the **status** command, and then retry **run-ping**.

Shutting down the mixnet

The mix network continues to run in the terminal where you started it until you issue a **Ctrl-C** key sequence, or until you issue the following command in another terminal:

```
~/katzenpost/docker$ make stop
```

When you stop the network, the binaries and data are left in place. This allows for a quick restart.

Uninstalling and cleaning up

Several command targets can be used to uninstall the Docker image and restore your system to a clean state. The following examples demonstrate the commands and their output.

- **clean-bin**

  To stop the network and delete the compiled binaries, run the following command:

  ```
  ~/katzenpost/docker$ make clean-bin
  ```

  ```
  [ -e voting_mixnet ] && cd voting_mixnet && DOCKER_HOST=unix:///run/user/1
  Stopping voting_mixnet_auth3_1        ... done
  Stopping voting_mixnet_servicenode1_1 ... done
  Stopping voting_mixnet_metrics_1      ... done
  Stopping voting_mixnet_mix3_1         ... done
  Stopping voting_mixnet_auth2_1        ... done
  Stopping voting_mixnet_mix2_1         ... done
  Stopping voting_mixnet_gateway1_1     ... done
  Stopping voting_mixnet_auth1_1        ... done
  Stopping voting_mixnet_mix1_1         ... done
  Removing voting_mixnet_auth3_1        ... done
  Removing voting_mixnet_servicenode1_1 ... done
  ```

```
Removing voting_mixnet_metrics_1      ... done
Removing voting_mixnet_mix3_1         ... done
Removing voting_mixnet_auth2_1        ... done
Removing voting_mixnet_mix2_1         ... done
Removing voting_mixnet_gateway1_1     ... done
Removing voting_mixnet_auth1_1        ... done
Removing voting_mixnet_mix1_1         ... done
removed 'running.stamp'
rm -vf ./voting_mixnet/*.alpine
removed './voting_mixnet/echo_server.alpine'
removed './voting_mixnet/fetch.alpine'
removed './voting_mixnet/memspool.alpine'
removed './voting_mixnet/panda_server.alpine'
removed './voting_mixnet/pigeonhole.alpine'
removed './voting_mixnet/ping.alpine'
removed './voting_mixnet/reunion_katzenpost_server.alpine'
removed './voting_mixnet/server.alpine'
removed './voting_mixnet/voting.alpine'
```

This command leaves in place the cryptographic keys, the state data, and the logs.

- **clean-local-dryrun**

  To diplay a preview of what **clean-local** would remove, without actually deleting anything, run the following command:

  ```
  ~/katzenpost/docker$ make clean-local-dryrun
  ```

- **clean-local**

  To delete both compiled binaries and data, run the following command:

  ```
  ~/katzenpost/docker$ make clean-local
  ```

  ```
  [ -e voting_mixnet ] && cd voting_mixnet && DOCKER_HOST=unix:///run/user/1
  Removing voting_mixnet_mix2_1         ... done
  Removing voting_mixnet_auth1_1        ... done
  Removing voting_mixnet_auth2_1        ... done
  Removing voting_mixnet_gateway1_1     ... done
  Removing voting_mixnet_mix1_1         ... done
  Removing voting_mixnet_auth3_1        ... done
  Removing voting_mixnet_mix3_1         ... done
  Removing voting_mixnet_servicenode1_1 ... done
  Removing voting_mixnet_metrics_1      ... done
  removed 'running.stamp'
  rm -vf ./voting_mixnet/*.alpine
  removed './voting_mixnet/echo_server.alpine'
  removed './voting_mixnet/fetch.alpine'
  removed './voting_mixnet/memspool.alpine'
  removed './voting_mixnet/panda_server.alpine'
  removed './voting_mixnet/pigeonhole.alpine'
  removed './voting_mixnet/reunion_katzenpost_server.alpine'
  removed './voting_mixnet/server.alpine'
  removed './voting_mixnet/voting.alpine'
  git clean -f -x voting_mixnet
  Removing voting_mixnet/
  git status .
  On branch main
  Your branch is up to date with 'origin/main'.
  ```

- **clean**

  To stop the the network and delete the binaries, the data, and the go_deps image, run the following command as superuser:

  ```
  ~/katzenpost/docker$ sudo make clean
  ```

Network topology and components

The Docker image deploys a working mixnet with all components and component groups needed to perform essential mixnet functions:

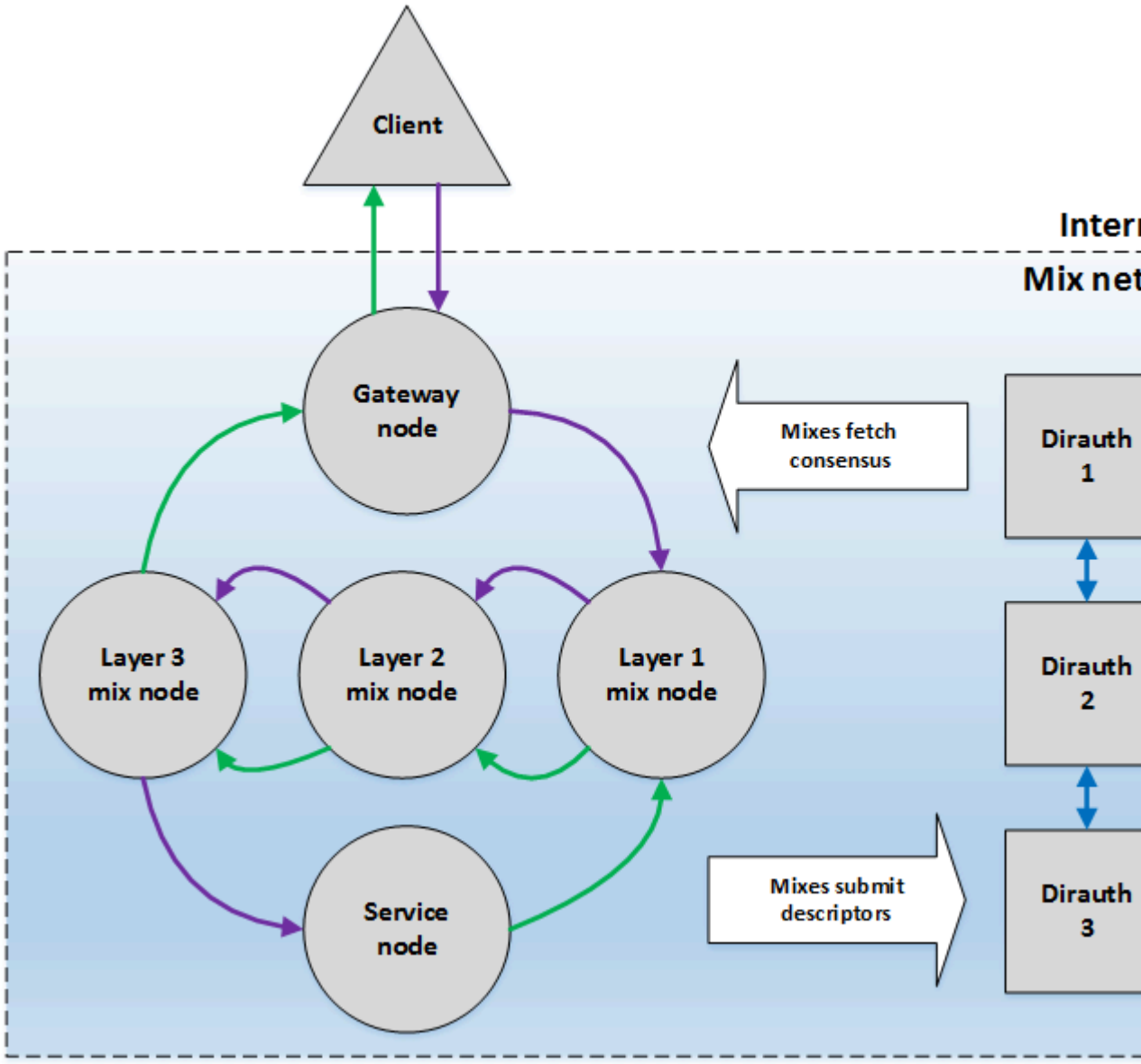
- message mixing (including packet reordering, timing randomization, injection of decoy traffic, obfuscation of senders and receivers, and so on)
- service provisioning
- internal authentication and integrity monitoring
- interfacing with external clients

Warning

While suited for client development and testing, the test mixnet omits performance and security redundancies. *Do not use it in production.*

The following diagram illustrates the components and their network interactions. The gray blocks represent nodes, and the arrows represent information transfer.

Figure 1. Test network topology



On the left, the **Client** transmits a message (shown by purple arrows) through the **Gateway node**, across three **mix node** layers, to the **Service node**. The **Service node** processes the request and responds with a reply (shown by the green arrows) that traverses the **mix node** layers before exiting the mixnet via the **Gateway node** and arriving at the **Client**.

On the right, directory authorities **Dirauth 1**, **Dirauth 2**, and **Dirauth 3** provide PKI services. The directory authorities receive **mix descriptors** from the other nodes, collate these into a **consensus document** containing validated network status and authentication materials , and make that available to the other nodes.

The elements in the topology diagram map to the mixnet's component nodes as shown in the following table. Note that all nodes share the same IP address (127.0.0.1, i.e., localhost), but are accessed through different ports. Each node type links to additional information in Components and configuration of the Katzenpost mixnet [https://katzenpost.network/docs/admin_guide/components.html].

Table 2. Table 2: Test mixnet hosts

Node type	Docker ID	Diagram label	IP address	TCP port
Directory authority [https://katzenpost.network/docs/admin_guide/components.html#intro-dirauth]	auth1	Dirauth1	127.0.0.1 (localhost)	30001
	auth2	Dirauth 2		30002
	auth3	Dirauth 3		30003
Gateway node [https://katzenpost.network/docs/admin_guide/components.html#intro-gateway]	gateway1	Gateway node		30004
Service node [https://katzenpost.network/docs/admin_guide/components.html#intro-service]	servicenode1	Service node		30006
Mix node [https://katzenpost.network/docs/admin_guide/components.html#intro-mix]	mix1	Layer 1 mix node		30008
	mix2	Layer 2 mix node		30010
	mix3	Layer 3 mix node		30012

The Docker file tree

The following tree [https://manpages.debian.org/bookworm/tree/tree.1.en.html] output shows the location, relative to the `katzenpost` repository root, of the files created by the Docker build. During testing and use, you would normally touch only the TOML configuration file associated with each node, as highlighted in the listing. For help in understanding these files and a complete list of configuration options, follow the links in Table 2: Test mixnet hosts.

```
katzenpost/docker/voting_mixnet/
|---auth1
|   |---authority.toml
|   |---identity.private.pem
|   |---identity.public.pem
|   |---katzenpost.log
|   |---link.private.pem
|   |---link.public.pem
|   |---persistence.db
|---auth2
|   |---authority.toml
|   |---identity.private.pem
|   |---identity.public.pem
|   |---katzenpost.log
|   |---link.private.pem
|   |---link.public.pem
|   |---persistence.db
|---auth3
```

```
|   |---authority.toml
|   |---identity.private.pem
|   |---identity.public.pem
|   |---katzenpost.log
|   |---link.private.pem
|   |---link.public.pem
|   |---persistence.db
|---client
|   |---client.toml
|---client2
|   |---client.toml
|---dirauth.alpine
|---docker-compose.yml
|---echo_server.alpine
|---fetch.alpine
|---gateway1
|   |---identity.private.pem
|   |---identity.public.pem
|   |---katzenpost.log
|   |---katzenpost.toml
|   |---link.private.pem
|   |---link.public.pem
|   |---management_sock
|   |---spool.db
|   |---users.db
|---memspool.alpine
|---mix1
|   |---identity.private.pem
|   |---identity.public.pem
|   |---katzenpost.log
|   |---katzenpost.toml
|   |---link.private.pem
|   |---link.public.pem
|---mix2
|   |---identity.private.pem
|   |---identity.public.pem
|   |---katzenpost.log
|   |---katzenpost.toml
|   |---link.private.pem
|   |---link.public.pem
|---mix3
|   |---identity.private.pem
|   |---identity.public.pem
|   |---katzenpost.log
|   |---katzenpost.toml
|   |---link.private.pem
|   |---link.public.pem
|---panda_server.alpine
|---pigeonhole.alpine
|---ping.alpine
|---prometheus.yml
|---proxy_client.alpine
|---proxy_server.alpine
|---running.stamp
|---server.alpine
|---servicenode1
|   |---identity.private.pem
|   |---identity.public.pem
```

```
|   |---katzenpost.log
|   |---katzenpost.toml
|   |---link.private.pem
|   |---link.public.pem
|   |---management_sock
|   |---map.storage
|   |---memspool.13.log
|   |---memspool.storage
|   |---panda.25.log
|   |---panda.storage
|   |---pigeonHole.19.log
|   |---proxy.31.log
|---voting_mixnet
```

Examples of complete TOML configuration files are provided in Appendix: Configuration files from the Docker test mixnet [https://katzenpost.network/docs/admin_guide/docker-config-appendix.html].