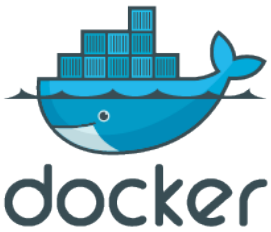
Databag Encrypted Chat

Selfhost your own End-to-End encrypted Webapplication Chat.

Overview

What is Databag?



Databag is a lightweight, decentralized messaging platform designed for efficiency and privacy. It enables direct communication between users and server nodes with a focus on end-to-end encryption and federation.

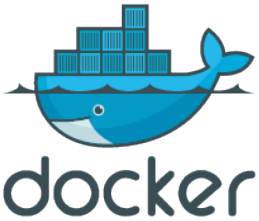
Keyfeatures

- **Decentralized** communication (direct app-to-server node connection)
- **Federated** network (users on different nodes can communicate)
- **Public-Private key based identity** (not tied to blockchain or domains)
- **End-to-End encryption** (even admins can't see sealed topics)
- **Audio and Video Calls** (supports NAT traversal with a relay server)
- **Topic-based threads** (messages are organized by topics)
- **Unlimited participants** in threads
- **Low latency** with websockets for real-time communication
- **Mobile alerts** for messages, contacts, and calls
- **Multi-Factor Authentication** (supports TOTP apps)
- **Runs on minimal hardware** (e.g., Raspberry Pi Zero)

Getting Started

Getting Started

Step-by-Step Setup Guide for Databag with Traefik



This guide will walk you through the setup process of deploying Databag, a decentralized and lightweight messaging system, in combination with Traefik as a reverse proxy using Docker Swarm. The configuration ensures that Databag runs efficiently across your Swarm nodes with persistent data storage.

Step 1: Set Up Data Directory in GlusterFS

To ensure that Databag's data is available across all Docker Swarm nodes, you will need to create a shared directory using GlusterFS. This directory will store the application's data.

Run the following command:

```
mkdir /mnt/glustermount/data/databagchat_data
```

This directory will be used to store Databag’s data, making it accessible across all nodes in the swarm.

Step 2: Create a `docker-compose.yaml` File

Next, you need to create a `docker-compose.yaml` file that defines the Databag service and its configuration. This file will also include Traefik as the reverse proxy to handle incoming HTTP/HTTPS traffic.

1. **Create the file**:
   Navigate to the directory where you want to store your Docker Compose file, and create it with the following command:

   ```
   nano docker-compose.yaml
   ```

2. **Edit the Admin Password**:
   Replace `${DATABAG_PASSWORD}` with a secure password or define it as an environment variable for added security. This password will be used to manage your Databag instance.

Here's the Docker Compose file structure:

```
version: "3.8"


services:
  databag:
    image: balzack/databag:0.1.17
    environment:
      # The ADMIN environment variable sets the admin password for the Databag application.
      # Replace ${DATABAG_PASSWORD} with your actual password or securely manage it via a secrets manager.
      - ADMIN=${DATABAG_PASSWORD}
    volumes:
      # The following volume ensures persistent storage for Databag's data across Swarm nodes.
      - /mnt/glustermount/data/databagchat_data:/var/lib/databag
    networks:
```

```
      - management_net
    deploy:
      # Deploy in "replicated" mode ensures the service runs across the number of replicas specified below.
      # In this case, the replicas are set to 1, meaning only one instance of the Databag service will run.
      mode: replicated
      replicas: 1
      labels:
        - 'traefik.enable=true'
        - 'traefik.http.routers.databag.rule=Host(`databag.domain.tld`)'
        - 'traefik.http.routers.databag.entrypoints=websecure'
        - 'traefik.http.routers.databag.tls.certresolver=leresolver'
        - 'traefik.http.services.databag.loadbalancer.server.port=7000'
        - 'traefik.docker.network=management_net'

networks:
  management_net:
    external: true
```

Optional: Set Additional Environment Variables

Here are some additional environment variables that can be set for further customization:

- **`ADMIN`**: Sets the admin password for the Databag application. Replace `${DATABAG_PASSWORD}` with a strong, secure password.
- **`DEV`**: Set this to `1` if you want to launch the server manually for development purposes.

> IF YOU WANT TO USE AUDIO/VIDEO CALL FUNCTION, GO VISIT **THIS ARTICLE**.

Step 3: Deploy the Stack in Docker Swarm

Once the Docker Compose file is configured, you can deploy the stack to Docker Swarm. This will ensure that Databag and Traefik are running across your nodes.

Deploy the stack using the following command:

```
docker stack deploy -c docker-compose.yaml databag
```

This command will start the Databag service with the Traefik reverse proxy, ensuring that traffic is correctly routed to your Databag instance.

>

> You can now access your Databag Server underhttps://databag.domain.tld/ and Login as Admin to configure more details.

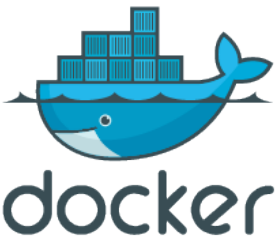
Conclusion

By following this guide, you have successfully set up Databag with Traefik in a Docker Swarm environment. The use of GlusterFS ensures persistent storage across your nodes, while Traefik manages traffic to your Databag service. You can now manage your decentralized chat service with a secure setup, fully capable of scaling across multiple nodes in your Swarm cluster.

Getting Started

Step-by-Step Setup Guide Coturn - TURN Server

GabrielTanner Website



This step-by-step article will guide you through setting up Coturn, a TURN server, using Docker Swarm and Traefik as a reverse proxy. You will configure Coturn to help WebRTC function by handling NAT traversal issues in peer-to-peer connections.

Prerequisites

- Linux server with Docker Swarm and Traefik installed. -> Check this Article
- A domain name for the TURN server (optional but recommended).
- GlusterFS or similar shared storage system (optional for Docker Swarm). -> Check this Article
- Knowledge of setting up Docker, Docker Compose, and basic networking. -> Check this Article

Step 1: Set Up Data Directory in GlusterFS

If you use Docker Swarm and want data shared across all nodes, create a directory in GlusterFS for persistent data.

```
mkdir -p /mnt/glustermount/data/coturn_data
```

Step 2: Create and Customize `turnserver.conf`

Create a configuration file for Coturn (`turnserver.conf`) that defines essential settings like server realm, authentication, ports, and SSL certificates.

```
sudo nano /mnt/glustermount/data/coturn_data/turnserver.conf
```

Also Create a Logfile for Persistant Data in GlusterFS:

```
sudo nano /mnt/glustermount/data/coturn_data/turnserver.log
```

Single Configure Steps

1. Add the following content to define your Coturn server realm and server name. Replace the placeholder values according to your needs.

   ```
   # TURN server name and realm
   realm=<DOMAIN>
   server-name=<SERVER_NAME>
   ```

2. After that, add the `external-ip` key to define your server's IP-Address and the `listening-ip` key to specify which IP-Addresses the Coturn server should listen to (0.0.0.0 tells the server to listen to all IP-Addresses).

   ```
   # IPs the TURN server listens to
   listening-ip=0.0.0.0

   # External IP-Address of the TURN server
   external-ip=IP_ADDRESS
   ```

3. Next you can define the port your server will listen on and the ports for further configuration.

   ```
   # Main listening port
   listening-port=3478

   # Further ports that are open for communication
   min-port=10000
   ```

```
max-port=20000
```

4. Then you can continue by defining the directory for your logs and enable the verbose logging mode.

```
# Use fingerprint in TURN message
fingerprint

# Log file path
log-file=/var/log/turnserver.log

# Enable verbose logging
verbose
```

5. Lastly, you can enable authentication for your TURN server using the `user` and `lt-cred-mech` keys.

```
# Specify the user for the TURN authentication
user=test:test123

# Enable long-term credential mechanism
lt-cred-mech
```

These configuration blocks will result in the following file:

```
# TURN server name and realm
realm=DOMAIN
server-name=turnserver

# Use fingerprint in TURN message
fingerprint

# IPs the TURN server listens to
listening-ip=0.0.0.0

# External IP-Address of the TURN server
external-ip=IP_ADDRESS
```

```
# Main listening port
listening-port=3478

# Further ports that are open for communication
min-port=10000
max-port=20000

# Log file path
log-file=/mnt/glustermount/data/coturn_data/turnserver.log

# Enable verbose logging
verbose

# Specify the user for the TURN authentification
user=test:test123

# Enable long-term credential mechanism
lt-cred-mech

# If running coturn version older than 4.5.2, uncomment these rules and ensure
# that you have listening-ip set to ipv4 addresses only.
# Prevent Loopback bypass https://github.com/coturn/coturn/security/advisories/GHSA-6g6j-r9rf-cm7p
#denied-peer-ip=0.0.0.0-0.255.255.255
#denied-peer-ip=127.0.0.0-127.255.255.255
#denied-peer-ip=::1
```

Once you’re done, save and exit your file.

You can further customize your configuration for your own needs by changing the give keys’ values or by adding new ones. You can reference the original configuration, which provides essential documentation for the most important options.

Step 3: Create PID Folder & Create CoturnUser

1. Create a System User for Coturn: This command creates a system user for Coturn with no login shell for security purposes:

```
sudo useradd -r -s /bin/false coturn
```

2. Set Ownership: Change the ownership of the Coturn data directory to the newly created Coturn user:

```
sudo chown -R coturn:coturn /mnt/glustermount/data/coturn_data
```

3. Set Correct Permissions: Ensure that only the Coturn user has access to this directory by setting the proper permissions:

```
sudo chmod -R 700 /mnt/glustermount/data/coturn_data
```

> These steps will ensure that Coturn has the correct permissions to access and write files in its data directory.

Step 4: Set Up Docker Compose for Coturn

Now, set up a `docker-compose.yaml` file to run Coturn inside Docker Swarm.

```
nano /mnt/glustermount/data/coturn_data/docker-compose.yaml
```

Add the following Docker Compose configuration:

```
coturn:
  image: coturn/coturn:4.5.2
  environment:
    - TURN_REALM=<DOMAIN>
    - TURN_LISTEN_PORT=3478
  volumes:
    - /mnt/glustermount/data/coturn_data/turnserver.conf:/etc/coturn/turnserver.conf:ro
    - /mnt/glustermount/data/coturn_data/pid:/var/run
  networks:
    - management_net
```

```
    deploy:
      mode: replicated
      replicas: 1

networks:
  management_net:
    external: true
```

Step 5: Port Forwarding

For external access, you need to set up **port forwarding** on your router or firewall. Forward the following ports:

- **3478 TCP/UDP**: TURN Port.
- **49152–65535 UDP**: Ensure this range of ports (or at least some) is open for relayed media traffic.

These ports allow external clients to connect to your VPN server and administrators to access the web interface.

> If you dont know how to do that go visit this Website from NordVPN.

Conclusion

Setting up a TURN server with Coturn in Docker Swarm can simplify peer-to-peer communication in WebRTC applications, especially in networks where NAT traversal is required. With Traefik as a reverse proxy, you can easily manage your server through HTTPS and make it accessible from the internet. By following this guide, you've built a robust, scalable, and secure TURN server setup.