Configuring Doctrine for Email Queue Management in Mautic with Cron Jobs

Configuring Doctrine for Email Queue Management in Mautic with Cron Jobs

Overview

This guide provides a detailed process to configure Mautic to use Doctrine for managing email queues, implementing cron jobs to automate the process, and ensuring efficient and stable email delivery.

Step 1: Setting Up Doctrine for Email Queues

Option 1: Configuring via local.php File

1. **Locate the local.php File**
   - The local.php file is typically found in the app/config directory of your Mautic installation.
2. **Edit the Configuration File**
   - Add or modify the following settings in the local.php file:

```
<?phpreturn array(// Other Mautic configuration settings...'me
ssenger' => [    'default_bus' => 'messenger.bus.default',
'transports' => [        'doctrine' => [            'dsn' => '
doctrine://default?table_name=messenger_messages',        ],
  ],],// Additional configuration settings...);
```

3. **Save the File**
   - Save the changes. These settings will be applied during the next

queue processing.

Option 2: Configuring via Mautic's Queue Configuration Screen

2. **Access the Queue Configuration Screen**
    - Log in to your Mautic dashboard.
    - Navigate to *Settings* > *Queue Settings*.
3. **Configure the Queue Settings**
    - **Queue Transport**: Doctrine
    - **Scheme**: doctrine
    - **Host**: default
    - **Table Name**: messenger_messages
4. **Save the Configuration**
    - Click *Save & Close* to apply the changes.

Step 2: Implementing Cron Jobs for Automated Email Sending

Creating the Shell Script

1. **Create the Script**
    - Save the following script as /path/to/mautic/var/lock/consume_mautic.sh:

```
#!/bin/bash# Path to the lock fileLOCKFILE="/path/to/mautic/var/lock/consume_mautic.lock"LOGFILE="/path/to/mautic/var/logs/mautic_consume_detailed.log"# Define the lock file timeout (9 minutes = 540 seconds)LOCKFILE_TIMEOUT=540# Function to log messageslog() {    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" >> "$LOGFILE"}log "Script started."# Check if the lock file existsif [ -e $LOCKFILE ]; then    LOCKFILE_AGE=$(($(date +%s) - $(stat -c %Y $LOCKFILE)))    if [ $LOCKFILE_AGE -ge $LOCKFILE_TIMEOUT ]; then        log "Lock file is older than $LOCKFILE_TIMEOUT seconds. Removing stale lock file."        rm -f $LOCKFILE    else        log "Process is already running. Exiting."    exit 1    fifi# Create the lock filetouch $LOCKFILElog "Lo
```

```
ck file created."# Run the messenger:consume command for email
s with increased verbosity and log the outputlog "Starting to
consume messages..."/usr/bin/php /path/to/mautic/bin/console m
essenger:consume email --limit=60 --time-limit=480 --memory-li
mit=128M -vvv >> "$LOGFILE" 2>&1log "Finished consuming messag
es."# Remove the lock file when donerm -f $LOCKFILElog "Lock f
ile removed."log "Script finished."
```

2. **Make the Script Executable**
    - Grant execute permissions to the script:

```
chmod +x /path/to/mautic/var/lock/consume_mautic.sh
```

Configuring the Cron Job

2. **Open the Crontab**
    - Edit your crontab with the following command:

```
crontab -e
```

3. **Add the Cron Job**
    - Add this line to your crontab:

```
*/10 * * * * /path/to/mautic/var/lock/consume_mautic.sh >> /pa
th/to/mautic/var/logs/mautic_consume.log 2>&1
```

This schedules the script to run every 10 minutes, logging output to mautic_consume.log.

Setting Up Directories and Permissions

2. **Create Necessary Directories**
    - Ensure the required directories exist:

```
mkdir -p /path/to/mautic/var/lockmkdir -p /path/to/mautic/var/
```

```
logs
```

3. **Check Permissions**
    - Verify that the user running the cron job has appropriate permissions for these directories.

Testing the Setup

2. **Run the Script Manually**
    - Test the script by running it manually:

```
/path/to/mautic/var/lock/consume_mautic.sh
```

3. **Check the Logs**
    - Monitor the log file to ensure the script is processing emails correctly:

```
tail -f /path/to/mautic/var/logs/mautic_consume.log
```

Step 3: Monitoring and Troubleshooting

Disk Space and Table Management

- **Warning**: The messenger_messages table can fill up quickly, especially under heavy load. Regularly monitor disk space and consider implementing automated cleanup routines to prevent crashes.
- **Alternative Queues**: If Doctrine struggles with high volumes, consider switching to a more scalable queue system like Redis or RabbitMQ.

Additional Logging

- **Verbose Logs**: The -vvv flag in the script provides detailed logs that can help diagnose issues.

Common Issues

- **Lock File Persistence**: Ensure the script removes the lock file after processing to avoid blocking future executions.
- **Message Backlog**: If the messenger_messages table fills up, consider adjusting cron timings or migrating to a different transport.