Radicale

Github | Official Website

> There is no official docker image. This was the most up to date project.

Step 1: Create Configuration Directory

Create a directory to store your Radicale configuration files:

```
mkdir -p /srv/Files/Radicale/config
```

Step 2: Create Configuration File

Create a file named `config` inside the `config` directory:

```
nano /srv/Files/Radicale/config/config
```

Paste the following configuration into the `config` file:

```
# -*- mode: conf -*-
# vim:ft=cfg

# Config file for Radicale - A simple calendar server
#
# Place it into /etc/radicale/config (global)
# or ~/.config/radicale/config (user)
#
# The current values are the default ones


[server]

# CalDAV server hostnames separated by a comma
# IPv4 syntax: address:port
# IPv6 syntax: [address]:port
```

```
# Hostname syntax (using "getaddrinfo" to resolve to IPv4/IPv6 adress(es)): hostname:port
# For example: 0.0.0.0:9999, [::]:9999, localhost:9999
#hosts = localhost:5232
hosts = 0.0.0.0:5232

# Max parallel connections
#max_connections = 8

# Max size of request body (bytes)
#max_content_length = 100000000

# Socket timeout (seconds)
#timeout = 30

# SSL flag, enable HTTPS protocol
#ssl = False

# SSL certificate path
#certificate = /etc/ssl/radicale.cert.pem

# SSL private key
#key = /etc/ssl/radicale.key.pem

# CA certificate for validating clients. This can be used to secure
# TCP traffic between Radicale and a reverse proxy
#certificate_authority =

# SSL protocol, secure configuration: ALL -SSLv3 -TLSv1 -TLSv1.1
#protocol = (default)

# SSL ciphersuite, secure configuration: DHE:ECDHE:-NULL:-SHA (see also "man openssl-ciphers")
#ciphersuite = (default)

# script name to strip from URI if called by reverse proxy
#script_name = (default taken from HTTP_X_SCRIPT_NAME or SCRIPT_NAME)


[encoding]
```

```
# Encoding for responding requests
#request = utf-8

# Encoding for storing local collections
#stock = utf-8



[auth]

# Authentication method
# Value: none | htpasswd | remote_user | http_x_remote_user | dovecot | ldap | oauth2 | pam |
denyall
#type = denyall
type = htpasswd

# Cache logins for until expiration time
#cache_logins = false

# Expiration time for caching successful logins in seconds
#cache_successful_logins_expiry = 15

## Expiration time of caching failed logins in seconds
#cache_failed_logins_expiry = 90

# URI to the LDAP server
#ldap_uri = ldap://localhost

# The base DN where the user accounts have to be searched
#ldap_base = ##BASE_DN##

# The reader DN of the LDAP server
#ldap_reader_dn = CN=ldapreader,CN=Users,##BASE_DN##

# Password of the reader DN
#ldap_secret = ldapreader-secret

# Path of the file containing password of the reader DN
#ldap_secret_file = /run/secrets/ldap_password
```

```
# the attribute to read the group memberships from in the user's LDAP entry (default: not set)
#ldap_groups_attribute = memberOf

# The filter to find the DN of the user. This filter must contain a python-style placeholder
for the login
#ldap_filter = (&(objectClass=person)(uid={0}))

# the attribute holding the value to be used as username after authentication
#ldap_user_attribute = cn

# Use ssl on the ldap connection
#ldap_use_ssl = False

# The certificate verification mode. NONE, OPTIONAL, default is REQUIRED
#ldap_ssl_verify_mode = REQUIRED

# The path to the CA file in pem format which is used to certificate the server certificate
#ldap_ssl_ca_file =

# Connection type for dovecot authentication (AF_UNIX|AF_INET|AF_INET6)
# Note: credentials are transmitted in cleartext
#dovecot_connection_type = AF_UNIX

# The path to the Dovecot client authentication socket (eg. /run/dovecot/auth-client on
Fedora). Radicale must have read / write access to the socket.
#dovecot_socket = /var/run/dovecot/auth-client

# Host of via network exposed dovecot socket
#dovecot_host = localhost

# Port of via network exposed dovecot socket
#dovecot_port = 12345

# IMAP server hostname
# Syntax: address | address:port | [address]:port | imap.server.tld
#imap_host = localhost

# Secure the IMAP connection
# Value: tls | starttls | none
```

```
#imap_security = tls

# OAuth2 token endpoint URL
#oauth2_token_endpoint = <URL>

# PAM service
#pam_serivce = radicale

# PAM group user should be member of
#pam_group_membership =

# Htpasswd filename
#htpasswd_filename = /etc/radicale/users
htpasswd_filename = /config/users

# Htpasswd encryption method
# Value: plain | bcrypt | md5 | sha256 | sha512 | autodetect
# bcrypt requires the installation of 'bcrypt' module.
#htpasswd_encryption = autodetect
htpasswd_encryption = bcrypt

# Enable caching of htpasswd file based on size and mtime_ns
#htpasswd_cache = False

# Incorrect authentication delay (seconds)
#delay = 1

# Message displayed in the client when a password is needed
#realm = Radicale - Password Required

# Convert username to lowercase, must be true for case-insensitive auth providers
#lc_username = False

# Strip domain name from username
#strip_domain = False


[rights]
```

```
# Rights backend
# Value: authenticated | owner_only | owner_write | from_file
#type = owner_only

# File for rights management from_file
#file = /etc/radicale/rights

# Permit delete of a collection (global)
#permit_delete_collection = True

# Permit overwrite of a collection (global)
#permit_overwrite_collection = True



[storage]

# Storage backend
# Value: multifilesystem | multifilesystem_nolock
#type = multifilesystem

# Folder for storing local collections, created if not present
#filesystem_folder = /var/lib/radicale/collections
filesystem_folder = /data/collections

# Folder for storing cache of local collections, created if not present
# Note: only used in case of use_cache_subfolder_* options are active
# Note: can be used on multi-instance setup to cache files on local node (see below)
#filesystem_cache_folder = (filesystem_folder)

# Use subfolder 'collection-cache' for 'item' cache file structure instead of inside
collection folder
# Note: can be used on multi-instance setup to cache 'item' on local node
#use_cache_subfolder_for_item = False

# Use subfolder 'collection-cache' for 'history' cache file structure instead of inside
collection folder
# Note: use only on single-instance setup, will break consistency with client in multi-
instance setup
#use_cache_subfolder_for_history = False
```

```
# Use subfolder 'collection-cache' for 'sync-token' cache file structure instead of inside
collection folder
# Note: use only on single-instance setup, will break consistency with client in multi-
instance setup
#use_cache_subfolder_for_synctoken = False

# Use last modifiction time (nanoseconds) and size (bytes) for 'item' cache instead of SHA256
(improves speed)
# Note: check used filesystem mtime precision before enabling
# Note: conversion is done on access, bulk conversion can be done offline using storage
verification option: radicale --verify-storage
#use_mtime_and_size_for_item_cache = False

# Use configured umask for folder creation (not applicable for OS Windows)
# Useful value: 0077 | 0027 | 0007 | 0022
#folder_umask = (system default, usual 0022)

# Delete sync token that are older (seconds)
#max_sync_token_age = 2592000

# Skip broken item instead of triggering an exception
#skip_broken_item = True

# Command that is run after changes to storage, default is emtpy
#  Supported placeholders:
#   %(user): logged-in user
#  Command will be executed with base directory defined in filesystem_folder
#  For "git" check DOCUMENTATION.md for bootstrap instructions
# Example: git add -A && (git diff --cached --quiet || git commit -m "Changes by
\"%(user)s\"")
#hook =

# Create predefined user collections
#
# json format:
#
#  {
#    "def-addressbook": {
```

```
#       "D:displayname": "Personal Address Book",
#       "tag": "VADDRESSBOOK"
#    },
#    "def-calendar": {
#       "C:supported-calendar-component-set": "VEVENT,VJOURNAL,VTODO",
#       "D:displayname": "Personal Calendar",
#       "tag": "VCALENDAR"
#    }
#  }
#
#predefined_collections =


[web]

# Web interface backend
# Value: none | internal
#type = internal


[logging]

# Threshold for the logger
# Value: debug | info | warning | error | critical
#level = info

# Don't include passwords in logs
#mask_passwords = True

# Log bad PUT request content
#bad_put_request_content = False

# Log backtrace on level=debug
#backtrace_on_debug = False

# Log request header on level=debug
#request_header_on_debug = False

# Log request content on level=debug
```

```
#request_content_on_debug = False

# Log response content on level=debug
#response_content_on_debug = False

# Log rights rule which doesn't match on level=debug
#rights_rule_doesnt_match_on_debug = False

# Log storage cache actions on level=debug
#storage_cache_actions_on_debug = False

[headers]

# Additional HTTP headers
#Access-Control-Allow-Origin = *


[hook]

# Hook types
# Value: none | rabbitmq
#type = none
#rabbitmq_endpoint =
#rabbitmq_topic =
#rabbitmq_queue_type = classic


[reporting]

# When returning a free-busy report, limit the number of returned
# occurences per event to prevent DOS attacks.
#max_freebusy_occurrence = 10000
```

Step 3: Create Users File

Create a file named `users` inside the `config` directory:

```
nano /srv/Files/Radicale/config/users
```

Each line in the `users` file should contain a username and a bcrypt-hashed password, separated by a colon (`:`). Use a tool like Browserling's BCrypt Generator to generate the hashed passwords. The file should look like this:

```
john:$2a$10$l1Se4qIaRlfOnaC1pGt32uNe/Dr61r4JrZQCNnY.kTx2KgJ70GPSm
sarah:$2a$10$lKEHYHjrZ.QHpWQeB/feWe/0m4ZtckLI.cYkVOITW8/0xoLCp1/Wy
```

Step 4: Create and Run Docker Container

Create a `docker-compose.yml` file with the following content to define your Docker service:

```
services:
  docker-radicale:
    container_name: radicale
    ports:
      - 104.152.49.17:5232:5232
    init: true
    read_only: true
    security_opt:
      - no-new-privileges:true
    cap_drop:
      - ALL
    cap_add:
      - CHOWN
      - SETUID
      - SETGID
      - KILL
    deploy:
      resources:
        limits:
          pids: 50
          memory: 256M
    healthcheck:
      test: curl --fail http://localhost:5232 || exit 1
      interval: 30s
      retries: "3"
    volumes:
      - /srv/Files/Radicale/data:/data
      - /srv/Files/Radicale/config:/config:ro
    image: tomsquest/docker-radicale
```

```
networks: {}
```

Run the following command to start the Docker container:

```
docker-compose up -d
```

This will start the Radicale server with the specified configuration.

Revision #3
Created 21 March 2025 10:15:43 by Admin
Updated 21 March 2025 10:42:24 by Admin