{%- from "redis/map.jinja" import cluster with context -%}

# Example sentinel.conf

# By default Redis does not run as a daemon. Use 'yes' if you need it.

# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.

daemonize yes

# When running daemonized, Redis writes a pid file in /var/run/redis.pid by

# default. You can specify a custom pid file location here.

pidfile /var/run/redis/redis-sentinel.pid

# Specify the log file name. Also the empty string can be used to force

# Redis to log on the standard output. Note that if you use standard

# output for logging but daemonize, logs will be sent to /dev/null

logfile /var/log/redis/redis-sentinel.log

# By default Redis listens for connections from all the network interfaces

# available on the server. It is possible to listen to just one or multiple

# interfaces using the "bind" configuration directive, followed by one or

# more IP addresses.

#

# Examples:

#

# bind 192.168.1.100 10.0.0.1

bind {{ cluster.sentinel.address }}

# port <sentinel-port>

# The port that this sentinel instance will run on

port {{ cluster.sentinel.port }}

# sentinel announce-ip <ip>

# sentinel announce-port <port>

#

# The above two configuration directives are useful in environments where,

# because of NAT, Sentinel is reachable from outside via a non-local address.

#

# When announce-ip is provided, the Sentinel will claim the specified IP address

# in HELLO messages used to gossip its presence, instead of auto-detecting the

# local address as it usually does.

#

# Similarly when announce-port is provided and is valid and non-zero, Sentinel

# will announce the specified TCP port.

#

# The two options don't need to be used together, if only announce-ip is

# provided, the Sentinel will announce the specified IP and the server port

# as specified by the "port" option. If only announce-port is provided, the

# Sentinel will announce the auto-detected local IP and the specified port.

#

# Example:

#

# sentinel announce-ip 1.2.3.4

# dir <working-directory>

# Every long running process should have a well-defined working directory.

# For Redis Sentinel to chdir to /tmp at startup is the simplest thing

# for the process to don't interfere with administrative tasks such as

# unmounting filesystems.

dir /var/lib/redis

# sentinel monitor <master-name> <ip> <redis-port> <quorum>

#

# Tells Sentinel to monitor this master, and to consider it in O\_DOWN

# (Objectively Down) state only if at least <quorum> sentinels agree.

#

# Note that whatever is the ODOWN quorum, a Sentinel will require to

# be elected by the majority of the known Sentinels in order to

# start a failover, so no failover can be performed in minority.

#

# Slaves are auto-discovered, so you don't need to specify slaves in

# any way. Sentinel itself will rewrite this configuration file adding

# the slaves using additional configuration options.

# Also note that the configuration file is rewritten when a

# slave is promoted to master.

#

# Note: master name should not include special characters or spaces.

# The valid charset is A-z 0-9 and the three characters ".-\_".

sentinel monitor master\_1 {{ cluster.master.host }} {{ cluster.master.port }} {{ cluster.quorum }}

# sentinel auth-pass <master-name> <password>

#

# Set the password to use to authenticate with the master and slaves.

# Useful if there is a password set in the Redis instances to monitor.

#

# Note that the master password is also used for slaves, so it is not

# possible to set a different password in masters and slaves instances

# if you want to be able to monitor these instances with Sentinel.

#

# However you can have Redis instances without the authentication enabled

# mixed with Redis instances requiring the authentication (as long as the

# password set is the same for all the instances requiring the password) as

# the AUTH command will have no effect in Redis instances with authentication

# switched off.

#

# Example:

#

# sentinel auth-pass mymaster MySUPER--secret-0123passw0rd

{%- if cluster.enabled and cluster.get("password", None) %}

sentinel auth-pass master\_1 {{ cluster.password }}

{%- endif %}

# sentinel down-after-milliseconds <master-name> <milliseconds>

#

# Number of milliseconds the master (or any attached slave or sentinel) should

# be unreachable (as in, not acceptable reply to PING, continuously, for the

# specified period) in order to consider it in S\_DOWN state (Subjectively

# Down).

#

# Default is 30 seconds.

sentinel down-after-milliseconds master\_1 30000

# sentinel parallel-syncs <master-name> <numslaves>

#

# How many slaves we can reconfigure to point to the new slave simultaneously

# during the failover. Use a low number if you use the slaves to serve query

# to avoid that all the slaves will be unreachable at about the same

# time while performing the synchronization with the master.

sentinel parallel-syncs master\_1 1

# sentinel failover-timeout <master-name> <milliseconds>

#

# Specifies the failover timeout in milliseconds. It is used in many ways:

#

# - The time needed to re-start a failover after a previous failover was

# already tried against the same master by a given Sentinel, is two

# times the failover timeout.

#

# - The time needed for a slave replicating to a wrong master according

# to a Sentinel current configuration, to be forced to replicate

# with the right master, is exactly the failover timeout (counting since

# the moment a Sentinel detected the misconfiguration).

#

# - The time needed to cancel a failover that is already in progress but

# did not produced any configuration change (SLAVEOF NO ONE yet not

# acknowledged by the promoted slave).

#

# - The maximum time a failover in progress waits for all the slaves to be

# reconfigured as slaves of the new master. However even after this time

# the slaves will be reconfigured by the Sentinels anyway, but not with

# the exact parallel-syncs progression as specified.

#

# Default is 3 minutes.

sentinel failover-timeout master\_1 180000

# SCRIPTS EXECUTION

#

# sentinel notification-script and sentinel reconfig-script are used in order

# to configure scripts that are called to notify the system administrator

# or to reconfigure clients after a failover. The scripts are executed

# with the following rules for error handling:

#

# If script exits with "1" the execution is retried later (up to a maximum

# number of times currently set to 10).

#

# If script exits with "2" (or an higher value) the script execution is

# not retried.

#

# If script terminates because it receives a signal the behavior is the same

# as exit code 1.

#

# A script has a maximum running time of 60 seconds. After this limit is

# reached the script is terminated with a SIGKILL and the execution retried.

# NOTIFICATION SCRIPT

#

# sentinel notification-script <master-name> <script-path>

#

# Call the specified notification script for any sentinel event that is

# generated in the WARNING level (for instance -sdown, -odown, and so forth).

# This script should notify the system administrator via email, SMS, or any

# other messaging system, that there is something wrong with the monitored

# Redis systems.

#

# The script is called with just two arguments: the first is the event type

# and the second the event description.

#

# The script must exist and be executable in order for sentinel to start if

# this option is provided.

#

# Example:

#

# sentinel notification-script mymaster /var/redis/notify.sh

# CLIENTS RECONFIGURATION SCRIPT

#

# sentinel client-reconfig-script <master-name> <script-path>

#

# When the master changed because of a failover a script can be called in

# order to perform application-specific tasks to notify the clients that the

# configuration has changed and the master is at a different address.

#

# The following arguments are passed to the script:

#

# <master-name> <role> <state> <from-ip> <from-port> <to-ip> <to-port>

#

# <state> is currently always "failover"

# <role> is either "leader" or "observer"

#

# The arguments from-ip, from-port, to-ip, to-port are used to communicate

# the old address of the master and the new address of the elected slave

# (now a master).

#

# This script should be resistant to multiple invocations.

#

# Example:

#

# sentinel client-reconfig-script mymaster /var/redis/reconfig.sh