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

# Redis configuration file example

# Note on units: when memory size is needed, it is possible to specifiy

# it in the usual form of 1k 5GB 4M and so forth:

#

# 1k => 1000 bytes

# 1kb => 1024 bytes

# 1m => 1000000 bytes

# 1mb => 1024\*1024 bytes

# 1g => 1000000000 bytes

# 1gb => 1024\*1024\*1024 bytes

#

# units are case insensitive so 1GB 1Gb 1gB are all the same.

# 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-server.pid

# Accept connections on the specified port, default is 6379.

# If port 0 is specified Redis will not listen on a TCP socket.

port {{ server.bind.port }}

# If you want you can bind a single interface, if the bind option is not

# specified all the interfaces will listen for incoming connections.

#

bind {{ server.bind.address }}

# Specify the path for the unix socket that will be used to listen for

# incoming connections. There is no default, so Redis will not listen

# on a unix socket when not specified.

#

# unixsocket /var/run/redis/redis.sock

# Close the connection after a client is idle for N seconds (0 to disable)

timeout 300

# Set server verbosity to 'debug'

# it can be one of:

# debug (a lot of information, useful for development/testing)

# verbose (many rarely useful info, but not a mess like the debug level)

# notice (moderately verbose, what you want in production probably)

# warning (only very important / critical messages are logged)

loglevel notice

# Specify the log file name. Also 'stdout' 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-server.log

# To enable logging to the system logger, just set 'syslog-enabled' to yes,

# and optionally update the other syslog parameters to suit your needs.

# syslog-enabled no

# Specify the syslog identity.

# syslog-ident redis

# Specify the syslog facility. Must be USER or between LOCAL0-LOCAL7.

# syslog-facility local0

# Set the number of databases. The default database is DB 0, you can select

# a different one on a per-connection basis using SELECT <dbid> where

# dbid is a number between 0 and 'databases'-1

databases 16

################################ SNAPSHOTTING #################################

#

# Save the DB on disk:

#

# save <seconds> <changes>

#

# Will save the DB if both the given number of seconds and the given

# number of write operations against the DB occurred.

#

# In the example below the behaviour will be to save:

# after 900 sec (15 min) if at least 1 key changed

# after 300 sec (5 min) if at least 10 keys changed

# after 60 sec if at least 10000 keys changed

#

# Note: you can disable saving at all commenting all the "save" lines.

save 900 1

save 300 10

save 60 10000

# Compress string objects using LZF when dump .rdb databases?

# For default that's set to 'yes' as it's almost always a win.

# If you want to save some CPU in the saving child set it to 'no' but

# the dataset will likely be bigger if you have compressible values or keys.

rdbcompression yes

# The filename where to dump the DB

dbfilename dump.rdb

# The working directory.

#

# The DB will be written inside this directory, with the filename specified

# above using the 'dbfilename' configuration directive.

#

# Also the Append Only File will be created inside this directory.

#

# Note that you must specify a directory here, not a file name.

dir /var/lib/redis

################################# REPLICATION #################################

# Master-Slave replication. Use slaveof to make a Redis instance a copy of

# another Redis server. Note that the configuration is local to the slave

# so for example it is possible to configure the slave to save the DB with a

# different interval, or to listen to another port, and so on.

#

# slaveof <masterip> <masterport>

# If the master is password protected (using the "requirepass" configuration

# directive below) it is possible to tell the slave to authenticate before

# starting the replication synchronization process, otherwise the master will

# refuse the slave request.

#

# masterauth <master-password>

# When a slave lost the connection with the master, or when the replication

# is still in progress, the slave can act in two different ways:

#

# 1) if slave-serve-stale-data is set to 'yes' (the default) the slave will

# still reply to client requests, possibly with out of data data, or the

# data set may just be empty if this is the first synchronization.

#

# 2) if slave-serve-stale data is set to 'no' the slave will reply with

# an error "SYNC with master in progress" to all the kind of commands

# but to INFO and SLAVEOF.

#

slave-serve-stale-data yes

################################## SECURITY ###################################

# Require clients to issue AUTH <PASSWORD> before processing any other

# commands. This might be useful in environments in which you do not trust

# others with access to the host running redis-server.

#

# This should stay commented out for backward compatibility and because most

# people do not need auth (e.g. they run their own servers).

#

# Warning: since Redis is pretty fast an outside user can try up to

# 150k passwords per second against a good box. This means that you should

# use a very strong password otherwise it will be very easy to break.

#

# requirepass foobared

# Command renaming.

#

# It is possilbe to change the name of dangerous commands in a shared

# environment. For instance the CONFIG command may be renamed into something

# of hard to guess so that it will be still available for internal-use

# tools but not available for general clients.

#

# Example:

#

# rename-command CONFIG b840fc02d524045429941cc15f59e41cb7be6c52

#

# It is also possilbe to completely kill a command renaming it into

# an empty string:

#

# rename-command CONFIG ""

################################### LIMITS ####################################

# Set the max number of connected clients at the same time. By default there

# is no limit, and it's up to the number of file descriptors the Redis process

# is able to open. The special value '0' means no limits.

# Once the limit is reached Redis will close all the new connections sending

# an error 'max number of clients reached'.

#

# maxclients 128

# Don't use more memory than the specified amount of bytes.

# When the memory limit is reached Redis will try to remove keys with an

# EXPIRE set. It will try to start freeing keys that are going to expire

# in little time and preserve keys with a longer time to live.

# Redis will also try to remove objects from free lists if possible.

#

# If all this fails, Redis will start to reply with errors to commands

# that will use more memory, like SET, LPUSH, and so on, and will continue

# to reply to most read-only commands like GET.

#

# WARNING: maxmemory can be a good idea mainly if you want to use Redis as a

# 'state' server or cache, not as a real DB. When Redis is used as a real

# database the memory usage will grow over the weeks, it will be obvious if

# it is going to use too much memory in the long run, and you'll have the time

# to upgrade. With maxmemory after the limit is reached you'll start to get

# errors for write operations, and this may even lead to DB inconsistency.

#

# maxmemory <bytes>

# MAXMEMORY POLICY: how Redis will select what to remove when maxmemory

# is reached? You can select among five behavior:

#

# volatile-lru -> remove the key with an expire set using an LRU algorithm

# allkeys-lru -> remove any key accordingly to the LRU algorithm

# volatile-random -> remove a random key with an expire set

# allkeys->random -> remove a random key, any key

# volatile-ttl -> remove the key with the nearest expire time (minor TTL)

# noeviction -> don't expire at all, just return an error on write operations

#

# Note: with all the kind of policies, Redis will return an error on write

# operations, when there are not suitable keys for eviction.

#

# At the date of writing this commands are: set setnx setex append

# incr decr rpush lpush rpushx lpushx linsert lset rpoplpush sadd

# sinter sinterstore sunion sunionstore sdiff sdiffstore zadd zincrby

# zunionstore zinterstore hset hsetnx hmset hincrby incrby decrby

# getset mset msetnx exec sort

#

# The default is:

#

# maxmemory-policy volatile-lru

# LRU and minimal TTL algorithms are not precise algorithms but approximated

# algorithms (in order to save memory), so you can select as well the sample

# size to check. For instance for default Redis will check three keys and

# pick the one that was used less recently, you can change the sample size

# using the following configuration directive.

#

# maxmemory-samples 3

############################## APPEND ONLY MODE ###############################

# By default Redis asynchronously dumps the dataset on disk. If you can live

# with the idea that the latest records will be lost if something like a crash

# happens this is the preferred way to run Redis. If instead you care a lot

# about your data and don't want to that a single record can get lost you should

# enable the append only mode: when this mode is enabled Redis will append

# every write operation received in the file appendonly.aof. This file will

# be read on startup in order to rebuild the full dataset in memory.

#

# Note that you can have both the async dumps and the append only file if you

# like (you have to comment the "save" statements above to disable the dumps).

# Still if append only mode is enabled Redis will load the data from the

# log file at startup ignoring the dump.rdb file.

#

# IMPORTANT: Check the BGREWRITEAOF to check how to rewrite the append

# log file in background when it gets too big.

appendonly no

# The name of the append only file (default: "appendonly.aof")

# appendfilename appendonly.aof

# The fsync() call tells the Operating System to actually write data on disk

# instead to wait for more data in the output buffer. Some OS will really flush

# data on disk, some other OS will just try to do it ASAP.

#

# Redis supports three different modes:

#

# no: don't fsync, just let the OS flush the data when it wants. Faster.

# always: fsync after every write to the append only log . Slow, Safest.

# everysec: fsync only if one second passed since the last fsync. Compromise.

#

# The default is "everysec" that's usually the right compromise between

# speed and data safety. It's up to you to understand if you can relax this to

# "no" that will will let the operating system flush the output buffer when

# it wants, for better performances (but if you can live with the idea of

# some data loss consider the default persistence mode that's snapshotting),

# or on the contrary, use "always" that's very slow but a bit safer than

# everysec.

#

# If unsure, use "everysec".

appendfsync {% if pillar.redis.server.appendfsync is defined %}{{ pillar.redis.server.appendfsync|replace("False","no") }}{% else %}everysec{% endif %}

# appendfsync always

# appendfsync no

# When the AOF fsync policy is set to always or everysec, and a background

# saving process (a background save or AOF log background rewriting) is

# performing a lot of I/O against the disk, in some Linux configurations

# Redis may block too long on the fsync() call. Note that there is no fix for

# this currently, as even performing fsync in a different thread will block

# our synchronous write(2) call.

#

# In order to mitigate this problem it's possible to use the following option

# that will prevent fsync() from being called in the main process while a

# BGSAVE or BGREWRITEAOF is in progress.

#

# This means that while another child is saving the durability of Redis is

# the same as "appendfsync none", that in pratical terms means that it is

# possible to lost up to 30 seconds of log in the worst scenario (with the

# default Linux settings).

#

# If you have latency problems turn this to "yes". Otherwise leave it as

# "no" that is the safest pick from the point of view of durability.

no-appendfsync-on-rewrite no

################################ VIRTUAL MEMORY ###############################

# Virtual Memory allows Redis to work with datasets bigger than the actual

# amount of RAM needed to hold the whole dataset in memory.

# In order to do so very used keys are taken in memory while the other keys

# are swapped into a swap file, similarly to what operating systems do

# with memory pages.

#

# To enable VM just set 'vm-enabled' to yes, and set the following three

# VM parameters accordingly to your needs.

# vm-enabled no

# vm-enabled yes

# This is the path of the Redis swap file. As you can guess, swap files

# can't be shared by different Redis instances, so make sure to use a swap

# file for every redis process you are running. Redis will complain if the

# swap file is already in use.

#

# The best kind of storage for the Redis swap file (that's accessed at random)

# is a Solid State Disk (SSD).

#

# \*\*\* WARNING \*\*\* if you are using a shared hosting the default of putting

# the swap file under /tmp is not secure. Create a dir with access granted

# only to Redis user and configure Redis to create the swap file there.

vm-swap-file /var/lib/redis/redis.swap

# vm-max-memory configures the VM to use at max the specified amount of

# RAM. Everything that deos not fit will be swapped on disk \*if\* possible, that

# is, if there is still enough contiguous space in the swap file.

#

# With vm-max-memory 0 the system will swap everything it can. Not a good

# default, just specify the max amount of RAM you can in bytes, but it's

# better to leave some margin. For instance specify an amount of RAM

# that's more or less between 60 and 80% of your free RAM.

vm-max-memory 0

# Redis swap files is split into pages. An object can be saved using multiple

# contiguous pages, but pages can't be shared between different objects.

# So if your page is too big, small objects swapped out on disk will waste

# a lot of space. If you page is too small, there is less space in the swap

# file (assuming you configured the same number of total swap file pages).

#

# If you use a lot of small objects, use a page size of 64 or 32 bytes.

# If you use a lot of big objects, use a bigger page size.

# If unsure, use the default :)

vm-page-size 32

# Number of total memory pages in the swap file.

# Given that the page table (a bitmap of free/used pages) is taken in memory,

# every 8 pages on disk will consume 1 byte of RAM.

#

# The total swap size is vm-page-size \* vm-pages

#

# With the default of 32-bytes memory pages and 134217728 pages Redis will

# use a 4 GB swap file, that will use 16 MB of RAM for the page table.

#

# It's better to use the smallest acceptable value for your application,

# but the default is large in order to work in most conditions.

vm-pages 134217728

# Max number of VM I/O threads running at the same time.

# This threads are used to read/write data from/to swap file, since they

# also encode and decode objects from disk to memory or the reverse, a bigger

# number of threads can help with big objects even if they can't help with

# I/O itself as the physical device may not be able to couple with many

# reads/writes operations at the same time.

#

# The special value of 0 turn off threaded I/O and enables the blocking

# Virtual Memory implementation.

vm-max-threads 4

############################### ADVANCED CONFIG ###############################

# Hashes are encoded in a special way (much more memory efficient) when they

# have at max a given numer of elements, and the biggest element does not

# exceed a given threshold. You can configure this limits with the following

# configuration directives.

hash-max-zipmap-entries 512

hash-max-zipmap-value 64

# Similarly to hashes, small lists are also encoded in a special way in order

# to save a lot of space. The special representation is only used when

# you are under the following limits:

list-max-ziplist-entries 512

list-max-ziplist-value 64

# Sets have a special encoding in just one case: when a set is composed

# of just strings that happens to be integers in radix 10 in the range

# of 64 bit signed integers.

# The following configuration setting sets the limit in the size of the

# set in order to use this special memory saving encoding.

set-max-intset-entries 512

# Active rehashing uses 1 millisecond every 100 milliseconds of CPU time in

# order to help rehashing the main Redis hash table (the one mapping top-level

# keys to values). The hash table implementation redis uses (see dict.c)

# performs a lazy rehashing: the more operation you run into an hash table

# that is rhashing, the more rehashing "steps" are performed, so if the

# server is idle the rehashing is never complete and some more memory is used

# by the hash table.

#

# The default is to use this millisecond 10 times every second in order to

# active rehashing the main dictionaries, freeing memory when possible.

#

# If unsure:

# use "activerehashing no" if you have hard latency requirements and it is

# not a good thing in your environment that Redis can reply form time to time

# to queries with 2 milliseconds delay.

#

# use "activerehashing yes" if you don't have such hard requirements but

# want to free memory asap when possible.

activerehashing yes

################################## INCLUDES ###################################

# Include one or more other config files here. This is useful if you

# have a standard template that goes to all redis server but also need

# to customize a few per-server settings. Include files can include

# other files, so use this wisely.

#

# include /path/to/local.conf

# include /path/to/other.conf