{%- from "gnocchi/map.jinja" import cfg,server,statsd with context %}

{%- set storage = cfg.get('storage', {}) %}

{%- set \_database = cfg.database %}

{%- set connection\_x509\_ssl\_option = '' %}

{%- if \_database.get('x509',{}).get('enabled',False) %}

{%- set connection\_x509\_ssl\_option = '&ssl\_ca=' ~ \_database.x509.ca\_file ~ '&ssl\_cert=' ~ \_database.x509.cert\_file ~ '&ssl\_key=' ~ \_database.x509.key\_file %}

{%- elif \_database.get('ssl',{}).get('enabled',False) %}

{%- set connection\_x509\_ssl\_option = '&ssl\_ca=' ~ \_database.ssl.get('cacert\_file', cfg.cacert\_file) %}

{%- endif %}

[DEFAULT]

#

# From cotyledon

#

# Enables or disables logging values of all registered options when starting a

# service (at DEBUG level). (boolean value)

# Note: This option can be changed without restarting.

#log\_options = true

# Specify a timeout after which a gracefully shutdown server will exit. Zero

# value means endless wait. (integer value)

# Note: This option can be changed without restarting.

#graceful\_shutdown\_timeout = 60

#

# From gnocchi

#

# If set to true, the logging level will be set to DEBUG. (boolean value)

#debug = false

debug = {{ cfg.debug }}

# If set to true, the logging level will be set to INFO. (boolean value)

#verbose = true

# Base directory for log files. If not set, logging will go to stderr. (string

# value)

#log\_dir = <None>

# (Optional) Name of log file to send logging output to. If no default is set,

# logging will go to stderr as defined by use\_stderr. (string value)

#log\_file = <None>

# Coordination driver URL (string value)

#coordination\_url = <None>

{%- if cfg.coordination\_backend is defined %}

{%- if cfg.get('coordination\_backend', {}).redis is defined and cfg.coordination\_backend.get('engine', redis) == 'redis' %}

{%- set conn = cfg.coordination\_backend.redis %}

{%- if conn.sentinel is defined %}

coordination\_url = {{ conn.get('scheme', 'redis') }}://{{ conn.user }}:{{ conn.password }}@{{ conn.sentinel.host }}:{{ conn.sentinel.get('port', '26379') }}?sentinel={{ conn.sentinel.master\_name }}{%

if conn.db is defined -%}&db={{ conn.db }}{% endif -%}{%

if conn.sentinel.fallback is defined %}{% for member in conn.sentinel.fallback -%}&sentinel\_fallback={{ member.host }}:{{ member.get('port', '26379') }}{%- endfor -%}{% endif -%}

{%- else %}

coordination\_url = {{ conn.get('scheme', 'redis') }}://{{ conn.user }}:{{ conn.password }}@{{ conn.host }}:{{ conn.get('port', '6379') }}{% if conn.db is defined -%}?db={{ conn.db }}{% endif -%}

{%- endif %}

{%- elif cfg.coordination\_backend.url is defined %}

coordination\_url = {{ cfg.coordination\_backend.url }}

{%- endif %}

{%- endif %}

# Number of threads to use to parallelize some operations. Default is set to

# the number of CPU available. (integer value)

# Minimum value: 1

# Deprecated group/name - [storage]/aggregation\_workers\_number

#parallel\_operations = <None>

{%- if storage.aggregation\_workers is defined %}

parallel\_operations = {{ storage.aggregation\_workers }}

{%- elif cfg.parallel\_operations is defined %}

parallel\_operations = {{ cfg.parallel\_operations }}

{%- endif %}

# Use syslog for logging. (boolean value)

#use\_syslog = false

{%- if cfg.use\_syslog is defined %}

use\_syslog = {{ cfg.use\_syslog }}

{%- endif %}

# Enable journald for logging. If running in a systemd environment you may wish

# to enable journal support. Doing so will use the journal native protocol

# which includes structured metadata in addition to log messages. (boolean

# value)

#use\_journal = false

{%- if cfg.use\_journal is defined %}

use\_journal = {{ cfg.use\_journal }}

{%- endif %}

# Syslog facility to receive log lines. (string value)

#syslog\_log\_facility = user

{%- if cfg.syslog\_log\_facility is defined %}

syslog\_log\_facility = {{ cfg.syslog\_log\_facility }}

{%- endif %}

[api]

#

# From gnocchi

#

# Path to API Paste configuration. (string value)

#paste\_config = api-paste.ini

# Authentication mode to use. (string value)

# Possible values:

# remoteuser - <No description provided>

# keystone - <No description provided>

# basic - <No description provided>

#auth\_mode = basic

auth\_mode = {{ server.identity.engine }}

# The maximum number of items returned in a single response from a collection

# resource (integer value)

#max\_limit = 1000

{%- if server.get('api', {}).max\_limit is defined %}

max\_limit = {{ server.api.max\_limit }}

{%- endif %}

# Number of seconds before timeout when attempting to do some operations.

# (integer value)

# Minimum value: 0

# Deprecated group/name - [api]/refresh\_timeout

#operation\_timeout = 10

{%- if server.get('api', {}).refresh\_timeout is defined %}

operation\_timeout = {{ server.api.refresh\_timeout }}

{%- elif server.api.operation\_timeout is defined %}

operation\_timeout = {{ server.api.operation\_timeout }}

{%- endif %}

# Host to listen on (host address value)

#host = 0.0.0.0

# Port to listen on (port value)

# Minimum value: 0

# Maximum value: 65535

#port = 8041

# Socket type to use for uWSGI:

# \* http: support HTTP/1.1 and keepalive,

# but not chunked encoding (InfluxDB)

# \* http-socket/socket: support chunked encoding, but require a upstream HTTP

# Server for HTTP/1.1, keepalive and HTTP protocol correctness.

# (string value)

# Possible values:

# http - <No description provided>

# http-socket - <No description provided>

# socket - <No description provided>

#uwsgi\_mode = http

[archive\_policy]

#

# From gnocchi

#

# Default aggregation methods to use in created archive policies (list value)

#default\_aggregation\_methods = mean,min,max,sum,std,count

{%- if server.get('archive\_policy', {}).default\_aggregation\_methods is defined %}

default\_aggregation\_methods = {{ ','.join(server.archive\_policy.default\_aggregation\_methods) }}

{%- endif %}

[cors]

{%- if server.cors is defined %}

{%- set \_data = server.cors %}

{%- include "oslo\_templates/files/rocky/oslo/\_cors.conf" %}

{%- endif %}

[incoming]

{%- set \_storage = storage.get('incoming', {}) %}

{%- if \_storage.driver is defined %}

driver = {{ \_storage.driver }}

{% include "gnocchi/files/" + cfg.version|string + "/storage/\_" + \_storage.driver + ".conf" %}

{%- endif %}

[indexer]

#

# From gnocchi

#

# Indexer driver to use (string value)

#url = <None>

url = {{ cfg.database.engine }}+pymysql://{{ cfg.database.user }}:{{ cfg.database.password }}@{{ cfg.database.host }}/{{ cfg.database.name }}?charset=utf8{{ connection\_x509\_ssl\_option|string }}

{%- if server.identity.engine == 'keystone' %}

[keystone\_authtoken]

{%- set \_data = server.identity %}

{%- if 'cacert\_file' not in \_data.keys() %}{% do \_data.update({'cacert\_file': server.cacert\_file}) %}{% endif %}

{%- set auth\_type = \_data.get('auth\_type', 'password') %}

{%- if server.get('cache',{}).members is defined and 'cache' not in \_data.keys() %}

{% do \_data.update({'cache': server.cache}) %}

{%- endif %}

{%- include "oslo\_templates/files/rocky/keystonemiddleware/\_auth\_token.conf" %}

{%- include "oslo\_templates/files/rocky/keystoneauth/\_type\_" + auth\_type + ".conf" %}

{%- endif %}

[oslo\_middleware]

{%- set \_data = server %}

{%- include "oslo\_templates/files/rocky/oslo/\_middleware.conf" %}

[oslo\_policy]

{%- if server.policy is defined %}

{%- set \_data = server.policy %}

{%- include "oslo\_templates/files/rocky/oslo/\_policy.conf" %}

{%- endif %}

[statsd]

#

# From gnocchi

#

# The listen IP for statsd (host address value)

#host = 0.0.0.0

{%- if statsd.get('bind', {}).address is defined %}

host = {{ statsd.bind.address }}

{%- endif %}

# The port for statsd (port value)

# Minimum value: 0

# Maximum value: 65535

#port = 8125

{%- if statsd.get('bind', {}).port is defined %}

port = {{ statsd.bind.port }}

{%- endif %}

# Resource UUID to use to identify statsd in Gnocchi (unknown value)

#resource\_id = <None>

{%- if statsd.get('enabled', False) %}

resource\_id = {{ statsd.resource\_id }}

{%- endif %}

# Creator value to use to identify statsd in Gnocchi (string value)

#creator = <None>

{%- if statsd.user\_id is defined and statsd.project\_id is defined %}

creator = {{ statsd.user\_id }}:{{ statsd.project\_id }}

{%- endif %}

# Archive policy name to use when creating metrics (string value)

#archive\_policy\_name = <None>

{%- if statsd.archive\_policy\_name is defined %}

archive\_policy\_name = {{ statsd.archive\_policy\_name }}

{%- endif %}

# Delay between flushes (floating point value)

#flush\_delay = 10

{%- if statsd.flush\_delay is defined %}

flush\_delay = {{ statsd.flush\_delay }}

{%- endif %}

[storage]

{%- set \_storage = storage %}

{%- if \_storage.driver is defined %}

driver = {{ \_storage.driver }}

{% include "gnocchi/files/" + cfg.version|string + "/storage/\_" + \_storage.driver + ".conf" %}

{%- endif %}

file\_basepath = /var/lib/gnocchi

{%- if server.cache is defined %}

{%- set \_data = server.cache %}

[cache]

{%- include "oslo\_templates/files/rocky/oslo/\_cache.conf" %}

{%- endif %}

{%- if cfg.configmap is defined %}

{%- set \_data = cfg.configmap %}

{%- include "oslo\_templates/files/configmap/configmap.conf" %}

{%- endif %}