{%- 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]

debug = {{ cfg.debug }}

[api]

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

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

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

{%- endif %}

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

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

{%- endif %}

[archive\_policy]

{%- 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 %}

{%- if server.cors.allowed\_origin is defined %}

allowed\_origin = {{ server.cors.allowed\_origin }}

{%- endif %}

{%- if server.cors.allow\_credentials is defined %}

allow\_credentials = {{ server.cors.allow\_credentials }}

{%- endif %}

{%- if server.cors.expose\_headers is defined %}

expose\_headers = {{ server.cors.expose\_headers }}

{%- endif %}

{%- if server.cors.max\_age is defined %}

max\_age = {{ server.cors.max\_age }}

{%- endif %}

{%- if server.cors.allow\_methods is defined %}

allow\_methods = {{ server.cors.allow\_methods }}

{%- endif %}

{%- if server.cors.allow\_headers is defined %}

allow\_headers = {{ server.cors.allow\_headers }}

{%- endif %}

{%- 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]

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

[keystone\_authtoken]

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

user\_domain\_id = {{ server.identity.get('domain', 'default') }}

project\_domain\_id = {{ server.identity.get('domain', 'default') }}

project\_name = {{ server.identity.tenant }}

username = {{ server.identity.user }}

password = {{ server.identity.password }}

auth\_uri = {{ server.identity.get('protocol', 'http') }}://{{ server.identity.host }}:{{ server.identity.private\_port }}

auth\_url = {{ server.identity.protocol }}://{{ server.identity.host }}:{{ server.identity.port }}

{%- if server.identity.protocol == 'https' %}

cafile={{ server.identity.get('cacert\_file', server.cacert\_file) }}

{%- endif %}

{%- if server.identity.auth\_type is defined %}

auth\_type = {{ server.identity.auth\_type }}

{%- endif %}

{%- endif %}

{%- if server.get('cache', {}).members is defined %}

memcached\_servers={%- for member in server.cache.members %}{{ member.host }}:{{ member.port }}{% if not loop.last %},{% endif %}{%- endfor %}

{%- endif %}

{%- if server.get('cache', {}).token\_cache\_time is defined %}

token\_cache\_time = {{ server.cache.token\_cache\_time }}

{%- endif %}

{%- if server.cache.get('security', {}).get('enabled', False) %}

memcache\_security\_strategy = {{ server.cache.security.get('strategy', 'ENCRYPT') }}

{%- if server.cache.security.secret\_key is not defined or not server.cache.security.secret\_key %}

{%- do salt.test.exception('gnocchi.server.cache.security.secret\_key is not defined: Please add secret\_key') %}

{%- else %}

memcache\_secret\_key = {{ server.cache.security.secret\_key }}

{%- endif %}

{%- endif %}

{%- if server.get('cache', {}).memcache\_pool\_dead\_retry is defined %}

memcache\_pool\_dead\_retry = {{ server.cache.memcache\_pool\_dead\_retry }}

{%- endif %}

{%- if server.get('cache', {}).memcache\_pool\_maxsize is defined %}

memcache\_pool\_maxsize = {{ server.cache.memcache\_pool\_maxsize }}

{%- endif %}

{%- if server.get('cache', {}).memcache\_pool\_socket\_timeout is defined %}

memcache\_pool\_socket\_timeout = {{ server.cache.memcache\_pool\_socket\_timeout }}

{%- endif %}

{%- if server.get('cache', {}).memcache\_pool\_unused\_timeout is defined %}

memcache\_pool\_unused\_timeout = {{ server.cache.memcache\_pool\_unused\_timeout }}

{%- endif %}

{%- if server.get('cache', {}).memcache\_pool\_conn\_get\_timeout is defined %}

memcache\_pool\_conn\_get\_timeout = {{ server.cache.memcache\_pool\_conn\_get\_timeout }}

{%- endif %}

[metricd]

{%- if server.get('metricd', {}).workers is defined %}

workers = {{ server.metricd.workers }}

{%- endif %}

{%- if server.get('metricd', {}).metric\_processing\_delay is defined %}

metric\_processing\_delay = {{ server.metricd.metric\_processing\_delay }}

{%- endif %}

{%- if server.get('metricd', {}).metric\_reporting\_delay is defined %}

metric\_reporting\_delay = {{ server.metricd.metric\_reporting\_delay }}

{%- endif %}

{%- if server.get('metricd', {}).metric\_cleanup\_delay is defined %}

metric\_cleanup\_delay = {{ server.metricd.metric\_cleanup\_delay }}

{%- endif %}

{%- if server.get('metricd', {}).worker\_sync\_rate is defined %}

worker\_sync\_rate = {{ server.metricd.worker\_sync\_rate }}

{%- endif %}

{%- if server.get('metricd', {}).processing\_replicas is defined %}

processing\_replicas = {{ server.metricd.processing\_replicas }}

{%- endif %}

[oslo\_middleware]

{%- if server.enable\_proxy\_headers\_parsing is defined %}

enable\_proxy\_headers\_parsing = {{ server.enable\_proxy\_headers\_parsing }}

{%- endif %}

[statsd]

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

host = {{ statsd.bind.address }}

{%- endif %}

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

port = {{ statsd.bind.port }}

{%- endif %}

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

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

{%- endif %}

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

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

{%- endif %}

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

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

{%- endif %}

{%- 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 %}

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

aggregation\_workers\_number = {{ storage.aggregation\_workers }}

{%- endif %}

{%- 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 %}

{%- if cfg.configmap is defined %}

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

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

{%- endif %}