{%- set log\_handlers = [] -%}

{%- set app\_name = \_data.app\_name -%}

{%- for log\_handler\_name, log\_handler\_attrs in \_data.log\_handlers.items() %}

{%- if log\_handler\_attrs.get('enabled', False) %}

{%- do log\_handlers.append(log\_handler\_name) -%}

{%- endif %}

{%- endfor %}

[loggers]

keys = root, {{ app\_name }}

[handlers]

keys = {{ log\_handlers | join(", ") }}

[formatters]

keys = context, default{% if \_data.log\_handlers.get('fluentd', {}).get('enabled', False) %}, fluentd{% endif %}

[logger\_root]

level = {{ \_data.get('loggers', {}).get('root', {}).get('level', 'INFO') }}

handlers = {{ log\_handlers | join(", ") }}

[logger\_{{ app\_name }}]

level = {{ \_data.get('loggers', {}).get(app\_name, {}).get('level', 'INFO') }}

handlers = {{ log\_handlers | join(", ") }}

qualname = {{ app\_name }}

propagate = 0

[logger\_amqp]

level = {{ \_data.get('loggers', {}).get('amqp', {}).get('level', 'WARNING') }}

handlers = {{ log\_handlers | join(", ") }}

qualname = amqp

[logger\_amqplib]

level = {{ \_data.get('loggers', {}).get('amqplib', {}).get('level', 'WARNING') }}

handlers = {{ log\_handlers | join(", ") }}

qualname = amqplib

[logger\_sqlalchemy]

level = {{ \_data.get('loggers', {}).get('sqlalchemy', {}).get('level', 'WARNING') }}

handlers = {{ log\_handlers | join(", ") }}

qualname = sqlalchemy

# "level = INFO" logs SQL queries.

# "level = DEBUG" logs SQL queries and results.

# "level = WARNING" logs neither. (Recommended for production systems.)

[logger\_boto]

level = {{ \_data.get('loggers', {}).get('boto', {}).get('level', 'WARNING') }}

handlers = {{ log\_handlers | join(", ") }}

qualname = boto

# NOTE(mikal): suds is used by the vmware driver, removing this will

# cause many extraneous log lines for their tempest runs. Refer to

# https://review.openstack.org/#/c/219225/ for details.

[logger\_suds]

level = {{ \_data.get('loggers', {}).get('suds', {}).get('level', 'INFO') }}

handlers = {{ log\_handlers | join(", ") }}

qualname = suds

[logger\_eventletwsgi]

level = {{ \_data.get('loggers', {}).get('eventletwsgi', {}).get('level', 'WARNING') }}

handlers = {{ log\_handlers | join(", ") }}

qualname = eventlet.wsgi.server

{% if \_data.log\_handlers.get('fluentd').get('enabled', False) -%}

[handler\_fluentd]

class = fluent.handler.FluentHandler

args = ("openstack.{{ service\_name | replace("-", ".", 1) }}", 'localhost', 24224)

formatter = fluentd

{%- endif %}

{% if \_data.log\_handlers.watchedfile.enabled -%}

[handler\_watchedfile]

class = handlers.WatchedFileHandler

args = ("/var/log/{{ app\_name }}/{{ service\_name }}.log",)

formatter = context

{%- endif %}

{% if \_data.log\_handlers.get('ossyslog', {}).get('enabled', False) -%}

{%- set ossyslog\_args = \_data.log\_handlers.ossyslog.get('args', {}) -%}

[handler\_ossyslog]

class = oslo\_log.handlers.OSSysLogHandler

# the OSSysLogHandler uses 'syslog' lib, where the LOG\_\* facilities are already \*8

# but in the context where the args are evaluated we have access only to Python's

# handlers.SysLogHandler.LOG\_\* constants that \_ARE\_NOT\_ multiplied by 8.

# To not have a completely magic single int in the rendered template,

# we multiply it here.

args = ( 8 \* handlers.SysLogHandler.{{ ossyslog\_args.get('facility', 'LOG\_USER') }}, )

formatter = context

{%- endif %}

[formatter\_context]

class = oslo\_log.formatters.ContextFormatter

[formatter\_default]

format = %(message)s

{% if \_data.log\_handlers.get('fluentd').get('enabled', False) -%}

[formatter\_fluentd]

class = oslo\_log.formatters.FluentFormatter

{%- endif %}