Why is my gunicorn process ignoring the log-level setting with Django?

Asked 9 years, 9 months ago Modified 9 years, 9 months ago Viewed 27k times

8

I have Nginx, Gunicorn, and Django all running on the same Ubuntu EC2 instance. I have a fairly conventional setup and would like to log all gunicorn errors to a particular file.

My configuration for Gunicorn is:

```
#!/bin/bash

NAME="server"
GUNICORNDIR=/ebs/env/bin
DJANGODIR=/ebs/server/
SOCKFILE=/tmp/gunicorn.sock
LOGFILE=/var/log/gunicorn/gunicorn.error
USER=ubuntu
GROUP=ubuntu
NUM_WORKERS=5
TIMEOUT=60
DJANGO_SETTINGS_MODULE=settings
DJANGO_WSGI_MODULE=wsgi

echo "Starting $NAME"

RUNDIR=$(dirname $SOCKFILE)
test -d $RUNDIR || mkdir -p $RUNDIR

exec $GUNICORNDIR/gunicorn ${DJANGO_WSGI_MODULE}:application \
  --name $NAME \
  --workers $NUM_WORKERS \
  --timeout=$TIMEOUT \
  --user=$USER --group=$GROUP \
  --log-level=error --log-file=$LOGFILE \
  --bind=unix:$SOCKFILE
```

However with this configuration I am getting all logs from DEBUG and above written to the file. My log-level parameter appears to be getting ignored.

What I am looking for is to only have these types of log messages written:

```
2014-01-02 13:54:53 [3327] [CRITICAL] WORKER TIMEOUT (pid:3338)
```

```
2014-01-02 13:54:53 [3327] [CRITICAL] WORKER TIMEOUT (pid:3338)
```

I thought that the Django logging config specified in my settings.py might be interfering so I added a handler and a logger to try and target gunicorn but that did not work.

```
'handlers': {
'gunicorn': {
    'level': 'ERROR',
    'class': 'logging.handlers.RotatingFileHandler',
    'formatter': 'verbose',
    'filename': '/ebs/log/gunicorn.error',
    'maxBytes': 1024 * 1024 * 100,
},
}
'loggers': {
'gunicorn.errors': {
'level': 'ERROR',
'handlers': ['gunicorn'],
'propagate': False,
},
```

Here are the versions that I am running

Django 1.5.4 Nginx nginx/1.1.19 Gunicorn 18.0

Any thoughts on what is wrong here?

** Update **

Here is what my django logging config looks like:

```
LOGGING = {
'version': 1,
'disable_existing_loggers': True,
'root': {
    'level': 'WARNING',
    'handlers': ['sentry'],
},
'formatters': {
    'verbose': {
        'format': '%(levelname)s %(asctime)s %(module)s %(process)d %(thread)d %
(message)s'
    },
```

```
    'simple': {
        'format': '%(levelname)s %(asctime)s -- %(message)s'
    }
},
'handlers': {
    'sentry': {
        'level': 'ERROR',
        'class': 'raven.contrib.django.raven_compat.handlers.SentryHandler',
    },
    'sentry_file': {
        'level': 'ERROR',
        'class': 'logging.handlers.RotatingFileHandler',
        'formatter': 'verbose',
        'filename': '/ebs/log/sentry_log.txt',
        'maxBytes': 1024 * 1024 * 100,  # 100 mb
    },
    'celery': {
        'level': 'DEBUG',
        'class': 'logging.handlers.RotatingFileHandler',
        'filename': '/ebs/log/celery/celery.log',
        'formatter': 'verbose',
        'maxBytes': 1024 * 1024 * 100,
    },
    'apps': {
        'level': 'DEBUG',
        'class': 'logging.handlers.RotatingFileHandler',
        'formatter': 'verbose',
        'filename': '/ebs/log/apps.log',
        'maxBytes': 1024 * 1024 * 100,
    },
    'apps.dss': {
        'level': 'DEBUG',
        'class': 'logging.handlers.RotatingFileHandler',
        'formatter': 'verbose',
        'filename': '/ebs/log/dss_apps.log',
        'maxBytes': 1024 * 1024 * 100,
    },

},
'loggers': {
    'django.db.backends': {
        'level': 'DEBUG',
        'handlers': ['sentry'],
        'propagate': False,
    },
    'sentry': {
        'level': 'DEBUG',
        'handlers': ['sentry'],
        'propagate': False,
    },
```

```
    'sentry.errors': {
        'level': 'ERROR',
        'handlers': ['sentry_file', 'sentry'],
        'propagate': False,
    },
    'celery': {
        'level': 'INFO',
        'handlers': ['sentry', 'celery'],
        'propagate': False
    },
    'apps': {
        'level': 'DEBUG',
        'handlers': ['apps', 'sentry'],
        'propagate': False
    },
    'apps.dss' : {
        'level': 'DEBUG',
        'handlers': ['apps.dss', 'sentry'],
        'propagate': False,
    },
},
}
```

python django logging nginx gunicorn

Share Improve this question Follow

edited Jan 6, 2014 at 15:10

asked Jan 2, 2014 at 14:12

Dana Ford

205 ● 1 ● 4 ● 7

Do you get gunicorn DEBUG messages in the log, or those from Django? – sk1p Jan 2, 2014 at 16:43

The DEBUG messages in the file are coming from Django. – Dana Ford Jan 2, 2014 at 20:35

Please post the full django logging configuration, there might be your problem – sk1p Jan 2, 2014 at 21:34

1 Answer

Sorted by: Highest score (default)

The `--log-level` setting of gunicorn only affects gunicorns own error logging facility. But the standard error and standard output of your application will also end up in the gunicorn log. I think you might have a `StreamHandler` somewhere in your Django logging config. `StreamHandler` logs to `stderr`

8 by default, so it ends up in your gunicorn log. Remove the `StreamHandler` or increase the level to fix your problem.

Share Improve this answer Follow

answered Jan 3, 2014 at 1:12

sk1p

6,655 ● 30 ● 35

Thanks sk1p, the StreamHandler was definitely a step in the right direction. I had a "console" handler that used the StreamHandler class and that was being written to the gunicorn logs. Removing that stopped DEBUG logs being written to gunicorn. However I still appear to be getting some django logs in my gunicorn log. I'm not getting all but definitely seeing IntegrityErrors in there. I have updated my question with my current logging config. – Dana Ford Jan 6, 2014 at 15:08

Are those `IntegrityError`s caught by sentry? Are they formatted like your usual log messages? If not, they might end up another way on your stderr... – sk1p Jan 6, 2014 at 17:44

Interesting point. After looking closer all the django logs that are in gunicorn are uncaught exceptions (i.e. i'm not explicitly handling them in the django code). So for example IntegrityErrors, MultipleObjectsReturned, DoesNotExist, etc. Sentry catches all exceptions so perhaps they are using StreamHandler behind the scenes. I'll dig deep, thanks for the guidance. – Dana Ford Jan 6, 2014 at 19:18

1 Indeed, raven adds a `StreamHandler` to some loggers which are not handled by sentry by default... – sk1p Jan 6, 2014 at 19:49

1 @sk1p I am also facing the same problem. Read about logger-class also but I didn't get clear idea how to solve the problem. For the information, I am using werkzeug wsgi with supervisor. here is my supervisor config command `gunicorn --log-level=debug  --logger-class=simple -b :3000 -k gevent -w 5 server:app` – Black_Rider Nov 26, 2014 at 17:17