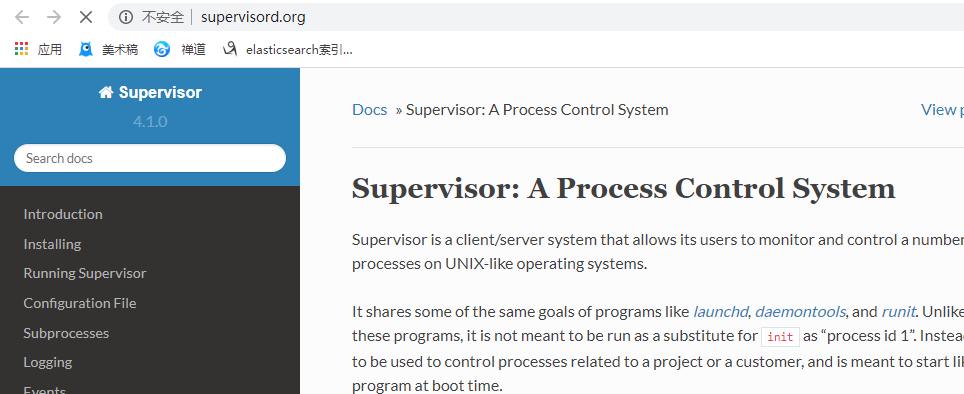
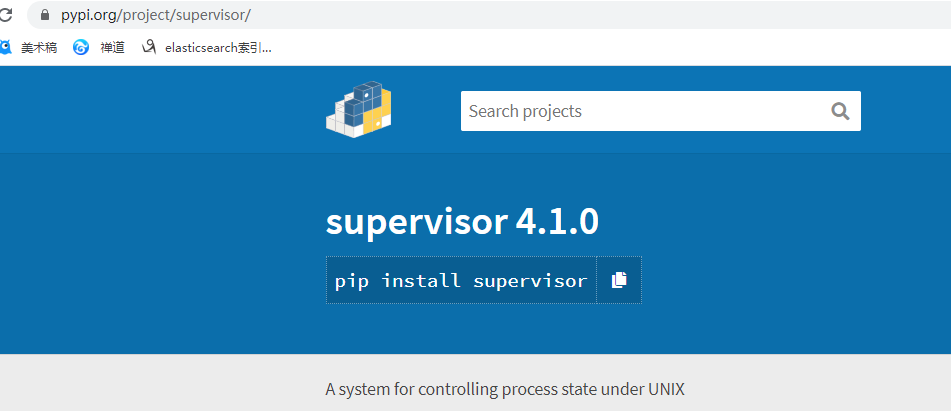
# Supervisor进程管理企业实战

作者：李立坤 [www.likun666.com](http://www.likun666.com)

## Supervisor概述

Supervisor是用Python开发的一套通用的进程管理程序，能将一个普通的命令行进程变为后台daemon，并监控进程状态，异常退出时能自动重启。它是通过fork/exec的方式把这些被管理的进程当作supervisor的子进程来启动，这样只要在supervisor的配置文件中，把要管理的进程的可执行文件的路径写进去即可。也实现当子进程挂掉的时候，父进程可以准确获取子进程挂掉的信息的，可以选择是否自己启动和报警。supervisor还提供了一个功能，可以为supervisord或者每个子进程，设置一个非root的user，这个user就可以管理它对应的进程。Supervisor官网地址为：<http://supervisord.org/>。





## Supervisor工作特性

**supervisor管理这些进程是把他们当做自己的子进程来管理，还有supervisor不能管理以daemon形式运行的进程，supervisor只能管理前台运行的进程。**

## **Supervisor组件剖析**

Supervisor由4大组件组成，分别为supervisord，supervisorctl，Web Server，XML-RPC 接口。

### Supervisord

主进程,负责管理进程的server，它会根据配置文件创建指定数量的应用程序的子进程，管理子进程的整个生命周期，对crash的进程重启，对进程变化发送事件通知等。同时内置web server和XML-RPC Interface，轻松实现进程管理。。该服务的配置文件在/etc/supervisor/supervisord.conf。

### **Supervisorctl**

客户端的命令行工具，提供一个类似shell的操作接口，通过它你可以连接到不同的supervisord进程上来管理它们各自的子程序，命令通过UNIX socket或者TCP来和服务通讯。用户通过命令行发送消息给supervisord，可以查看进程状态，加载配置文件，启停进程，查看进程标准输出和错误输出，远程操作等。服务端也可以要求客户端提供身份验证之后才能进行操作。

### **Web Server**

Superviosr提供了web server功能，可通过web控制进程(需要设置[inethttpserver]配置项)。

### **XML-RPC Interface**

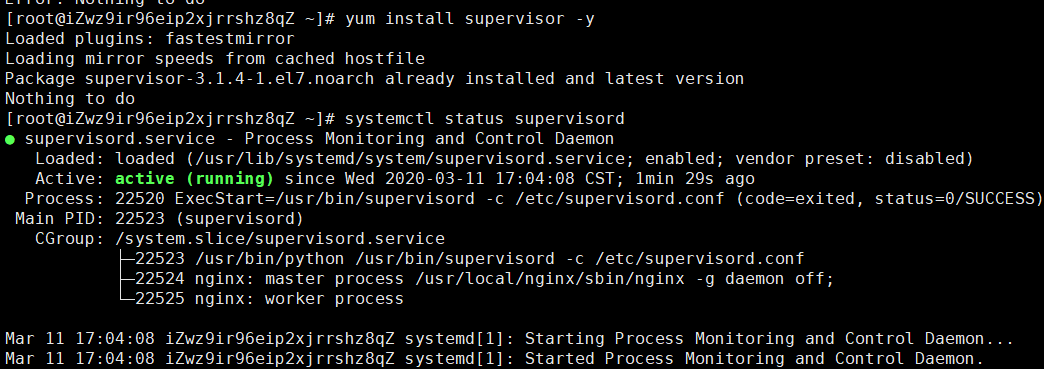
XML-RPC接口， 就像HTTP提供WEB UI一样，用来控制supervisor和由它运行的程序。

## Supervisor企业实战

### **Supervisor安装**

安装supervisor比较简单，具体操作如下：

|  |
| --- |
| yum install epel\* -y  yum install supervisor -y  systemctl start supervisord  systemctl enable supervisord |



### **Supervisor常用命令**

|  |
| --- |
| supervisorctl status //查看所有进程的状态  supervisorctl stop es //停止es  supervisorctl start es //启动es  supervisorctl restart //重启es  supervisorctl update //配置文件修改后使用该命令加载新的配置  supervisorctl reload //重新启动配置中的所有程序 |

### **Supervisor核心配置文件详解**

以下为supervisord的默认配置文件，可直接使用，具体如下：

|  |
| --- |
| ; Sample supervisor config file.  [unix\_http\_server]  file=/var/run/supervisor/supervisor.sock ; (the path to the socket file)  ;chmod=0700 ; sockef file mode (default 0700)  ;chown=nobody:nogroup ; socket file uid:gid owner  ;username=user ; (default is no username (open server))  ;password=123 ; (default is no password (open server))  ;[inet\_http\_server] ; inet (TCP) server disabled by default  ;port=127.0.0.1:9001 ; (ip\_address:port specifier, \*:port for all iface)  ;username=user ; (default is no username (open server))  ;password=123 ; (default is no password (open server))  [supervisord]  logfile=/var/log/supervisor/supervisord.log ; (main log file;default $CWD/supervisord.log)  logfile\_maxbytes=50MB ; (max main logfile bytes b4 rotation;default 50MB)  logfile\_backups=10 ; (num of main logfile rotation backups;default 10)  loglevel=info ; (log level;default info; others: debug,warn,trace)  pidfile=/var/run/supervisord.pid ; (supervisord pidfile;default supervisord.pid)  nodaemon=false ; (start in foreground if true;default false)  minfds=1024 ; (min. avail startup file descriptors;default 1024)  minprocs=200 ; (min. avail process descriptors;default 200)  ;umask=022 ; (process file creation umask;default 022)  ;user=chrism ; (default is current user, required if root)  ;identifier=supervisor ; (supervisord identifier, default is 'supervisor')  ;directory=/tmp ; (default is not to cd during start)  ;nocleanup=true ; (don't clean up tempfiles at start;default false)  ;childlogdir=/tmp ; ('AUTO' child log dir, default $TEMP)  ;environment=KEY=value ; (key value pairs to add to environment)  ;strip\_ansi=false ; (strip ansi escape codes in logs; def. false)  ; the below section must remain in the config file for RPC  ; (supervisorctl/web interface) to work, additional interfaces may be  ; added by defining them in separate rpcinterface: sections  [rpcinterface:supervisor]  supervisor.rpcinterface\_factory = supervisor.rpcinterface:make\_main\_rpcinterface  [supervisorctl]  serverurl=unix:///var/run/supervisor/supervisor.sock ; use a unix:// URL for a unix socket  ;serverurl=http://127.0.0.1:9001 ; use an http:// url to specify an inet socket  ;username=chris ; should be same as http\_username if set  ;password=123 ; should be same as http\_password if set  ;prompt=mysupervisor ; cmd line prompt (default "supervisor")  ;history\_file=~/.sc\_history ; use readline history if available  ; The below sample program section shows all possible program subsection values,  ; create one or more 'real' program: sections to be able to control them under  ; supervisor.  ;[program:theprogramname]  ;command=/bin/cat ; the program (relative uses PATH, can take args)  ;process\_name=%(program\_name)s ; process\_name expr (default %(program\_name)s)  ;numprocs=1 ; number of processes copies to start (def 1)  ;directory=/tmp ; directory to cwd to before exec (def no cwd)  ;umask=022 ; umask for process (default None)  ;priority=999 ; the relative start priority (default 999)  ;autostart=true ; start at supervisord start (default: true)  ;autorestart=true ; retstart at unexpected quit (default: true)  ;startsecs=10 ; number of secs prog must stay running (def. 1)  ;startretries=3 ; max # of serial start failures (default 3)  ;exitcodes=0,2 ; 'expected' exit codes for process (default 0,2)  ;stopsignal=QUIT ; signal used to kill process (default TERM)  ;stopwaitsecs=10 ; max num secs to wait b4 SIGKILL (default 10)  ;user=chrism ; setuid to this UNIX account to run the program  ;redirect\_stderr=true ; redirect proc stderr to stdout (default false)  ;stdout\_logfile=/a/path ; stdout log path, NONE for none; default AUTO  ;stdout\_logfile\_maxbytes=1MB ; max # logfile bytes b4 rotation (default 50MB)  ;stdout\_logfile\_backups=10 ; # of stdout logfile backups (default 10)  ;stdout\_capture\_maxbytes=1MB ; number of bytes in 'capturemode' (default 0)  ;stdout\_events\_enabled=false ; emit events on stdout writes (default false)  ;stderr\_logfile=/a/path ; stderr log path, NONE for none; default AUTO  ;stderr\_logfile\_maxbytes=1MB ; max # logfile bytes b4 rotation (default 50MB)  ;stderr\_logfile\_backups=10 ; # of stderr logfile backups (default 10)  ;stderr\_capture\_maxbytes=1MB ; number of bytes in 'capturemode' (default 0)  ;stderr\_events\_enabled=false ; emit events on stderr writes (default false)  ;environment=A=1,B=2 ; process environment additions (def no adds)  ;serverurl=AUTO ; override serverurl computation (childutils)  ; The below sample eventlistener section shows all possible  ; eventlistener subsection values, create one or more 'real'  ; eventlistener: sections to be able to handle event notifications  ; sent by supervisor.  ;[eventlistener:theeventlistenername]  ;command=/bin/eventlistener ; the program (relative uses PATH, can take args)  ;process\_name=%(program\_name)s ; process\_name expr (default %(program\_name)s)  ;numprocs=1 ; number of processes copies to start (def 1)  ;events=EVENT ; event notif. types to subscribe to (req'd)  ;buffer\_size=10 ; event buffer queue size (default 10)  ;directory=/tmp ; directory to cwd to before exec (def no cwd)  ;umask=022 ; umask for process (default None)  ;priority=-1 ; the relative start priority (default -1)  ;autostart=true ; start at supervisord start (default: true)  ;autorestart=unexpected ; restart at unexpected quit (default: unexpected)  ;startsecs=10 ; number of secs prog must stay running (def. 1)  ;startretries=3 ; max # of serial start failures (default 3)  ;exitcodes=0,2 ; 'expected' exit codes for process (default 0,2)  ;stopsignal=QUIT ; signal used to kill process (default TERM)  ;stopwaitsecs=10 ; max num secs to wait b4 SIGKILL (default 10)  ;user=chrism ; setuid to this UNIX account to run the program  ;redirect\_stderr=true ; redirect proc stderr to stdout (default false)  ;stdout\_logfile=/a/path ; stdout log path, NONE for none; default AUTO  ;stdout\_logfile\_maxbytes=1MB ; max # logfile bytes b4 rotation (default 50MB)  ;stdout\_logfile\_backups=10 ; # of stdout logfile backups (default 10)  ;stdout\_events\_enabled=false ; emit events on stdout writes (default false)  ;stderr\_logfile=/a/path ; stderr log path, NONE for none; default AUTO  ;stderr\_logfile\_maxbytes=1MB ; max # logfile bytes b4 rotation (default 50MB)  ;stderr\_logfile\_backups ; # of stderr logfile backups (default 10)  ;stderr\_events\_enabled=false ; emit events on stderr writes (default false)  ;environment=A=1,B=2 ; process environment additions  ;serverurl=AUTO ; override serverurl computation (childutils)  ; The below sample group section shows all possible group values,  ; create one or more 'real' group: sections to create "heterogeneous"  ; process groups.  ;[group:thegroupname]  ;programs=progname1,progname2 ; each refers to 'x' in [program:x] definitions  ;priority=999 ; the relative start priority (default 999)  ; The [include] section can just contain the "files" setting. This  ; setting can list multiple files (separated by whitespace or  ; newlines). It can also contain wildcards. The filenames are  ; interpreted as relative to this file. Included files \*cannot\*  ; include files themselves.  [include]  files = supervisord.d/\*.conf |

Supervisor配置文件详解，具体如下：

|  |
| --- |
| [unix\_http\_server]  file=/tmp/supervisor.sock ;UNIX socket 文件，supervisorctl 会使用;chmod=0700 ;socket文件的mode，默认是0700;chown=nobody:nogroup ;socket文件的owner，格式：uid:gid  ;[inet\_http\_server] ;HTTP服务器，提供web管理界面;port=127.0.0.1:9001 ;Web管理后台运行的IP和端口，如果开放到公网，需要注意安全性;username=user ;登录管理后台的用户名;password=123 ;登录管理后台的密码  [supervisord]  logfile=/tmp/supervisord.log ;日志文件，默认是 $CWD/supervisord.log  logfile\_maxbytes=50MB ;日志文件大小，超出会rotate，默认 50MB，如果设成0，表示不限制大小  logfile\_backups=10 ;日志文件保留备份数量默认10，设为0表示不备份  loglevel=info ;日志级别，默认info，其它: debug,warn,trace  pidfile=/tmp/supervisord.pid ;pid 文件  nodaemon=false ;是否在前台启动，默认是false，即以 daemon 的方式启动  minfds=1024 ;可以打开的文件描述符的最小值，默认 1024  minprocs=200 ;可以打开的进程数的最小值，默认 200  [supervisorctl]  serverurl=unix:///tmp/supervisor.sock ;通过UNIX socket连接supervisord，路径与unix\_http\_server部分的file一致;serverurl=http://127.0.0.1:9001 ; 通过HTTP的方式连接supervisord  ; [program:xx]是被管理的进程配置参数，xx是进程的名称[program:xx]  command=/opt/apache-tomcat-8.0.35/bin/catalina.sh run ; 程序启动命令  autostart=true ; 在supervisord启动的时候也自动启动  startsecs=10 ; 启动10秒后没有异常退出，就表示进程正常启动了，默认为1秒  autorestart=true ; 程序退出后自动重启,可选值：[unexpected,true,false]，默认为unexpected，表示进程意外杀死后才重启  startretries=3 ; 启动失败自动重试次数，默认是3  user=tomcat ; 用哪个用户启动进程，默认是root  priority=999 ; 进程启动优先级，默认999，值小的优先启动  redirect\_stderr=true ; 把stderr重定向到stdout，默认false  stdout\_logfile\_maxbytes=20MB ; stdout 日志文件大小，默认50MB  stdout\_logfile\_backups = 20 ; stdout 日志文件备份数，默认是10; stdout 日志文件，需要注意当指定目录不存在时无法正常启动，所以需要手动创建目录（supervisord 会自动创建日志文件）  stdout\_logfile=/opt/apache-tomcat-8.0.35/logs/catalina.out  stopasgroup=false ;默认为false,进程被杀死时，是否向这个进程组发送stop信号，包括子进程  killasgroup=false ;默认为false，向进程组发送kill信号，包括子进程  ;包含其它配置文件[include]  files = relative/directory/\*.ini ;可以指定一个或多个以.ini结束的配置文件 |

## **Supervisor管理Nginx程序实战**

在上面我们学习了supervisor的核心配置文件，我们会发现其实管理某个程序只需要编写某个程序的配置文件即可。比如说管理nginx程序，只需要编写nginx的supervisor的配置，在supervisor核心配置文件中有一个配置files = supervisord.d/\*.conf代表了将导入里面的子配置文件，这个类似nginx的虚拟主机配置文件。子配置文件的后缀可以在supervisor主配置文件中定义，默认后缀是.ini结尾，我这里修改为了.conf结尾。Supervisor管理nginx的配置文件如下：

|  |
| --- |
| [program:nginx]  directory=/usr/local/nginx/sbin  command=/usr/local/nginx/sbin/nginx -g 'daemon off;'  autostart=true  autorestart=true  stderr\_logfile=/tmp/nginx\_stderr.log  stdout\_logfile=/tmp/nginx\_stdout.log  user = root  stopsignal=INT  startsecs=10  startretries=5  stopasgroup=true |

Supervisor管理nginx的配置文件详解：

|  |
| --- |
| [program: nginx]  ;管理的子进程。后面的是名字，最好写的具有代表性，避免日后“认错”  command=/usr/local/nginx/sbin/nginx -g 'daemon off;'  ;我们的要启动进程的命令路径，可以带参数。  directory=/usr/local/nginx/sbin ;  ;进程运行前，会先切换到这个目录  autorestart=true；  ;设置子进程挂掉后自动重启的情况，有三个选项，false,unexpected和true。false表示无论什么情况下，都不会被重新启动；unexpected表示只有当进程的退出码不在下面的exitcodes里面定义的退出码的时候，才会被自动重启。当为true的时候，只要子进程挂掉，将会被无条件的重启  autostart=true ;  ;如果是true的话，子进程将在supervisord启动后被自动启动，默认就是true  stderr\_logfile=/tmp/nginx\_error.log ;  ;日志，没什么好说的  stdout\_logfile=/tmp/nginx\_stdout.log ;  ;日志  environment=ASPNETCORE\_ENVIRONMENT=Production ;  ;这个是子进程的环境变量,默认为空  user=nginx ;  ;可以用来管理该program的用户  stopsignal=INT  ;进程停止信号，可以为TERM, HUP, INT, QUIT, KILL, USR1等,默认为TERM  startsecs=10 ;  ;子进程启动多少秒之后,此时状态如果是running,我们认为启动成功了,默认值1  startretries=5 ; 当进程启动失败后，最大尝试的次数。当超过5次后，进程的状态变为FAIL  stopasgroup=true  ;这个东西主要用于，supervisord管理的子进程，这个子进程本身还有子进程。那么我们如果仅仅干掉supervisord的子进程的话，子进程的子进程有可能会变成孤儿进程。所以可以设置这个选项，把整个该子进程的整个进程组干掉。默认false |

直接重启supervisor即可，操作如下：

|  |
| --- |
| systemctl restart supervisord  systemctl status supervisord  supervisorctl status |

