1. 框架&流程
   1. 系统架构
   2. 业务流程
   3. 代码组织
      1. bmeMonitor.py: web入口
      2. com: 基础库，包括cmdLib.py命令行执行、confLib.py配置文件解析、dynsqlLib.py动态sql、ftpLib.py下载、httpclientLib.py请求、jsonLib.py库、logLib.py日志处理、mailLib.py邮件发送、msgSend.py短信发送、mysqlLib.py业务sql、netLib.py网络基础库、scheduleLib.py任务调度、sftpLib.py下载等
      3. conf: 配置文件，后面再解释用途
      4. dispatch: 分布式任务队列客户端、服务端、以及工作端实现
      5. log: 系统日志
      6. package: 外部模块
      7. scripts: 监控脚本存放目录
      8. start\_bmeMonitor.sh/ stop\_bmeMonitor.sh: web启停脚本
      9. start\_taskRunner.sh/ stop\_taskRunner.sh: taskrunner启停脚本
      10. static: web相关js、imag、css存放路径
      11. taskRunner.py: 监控任务启动器
      12. template: web相关html存放路径, 其中demo目录存放主战文件, plugins存放监控插件文件
   4. 流程详解
      1. taskRunner.py启动流程: taskRunner.py启动读取conf/period\_task.conf任务项配置文件启动Scheduler，Scheduler根据配置项不断提交新任务，taskRunner.py日志在log/taskrunner.log中查看，taskRunner.py提交任务,提交的任务插入到数据表tbl\_monitortask中，返回新增的任务id，并初始化任务状态status为0，表示任务处于初始化状态。log/taskrunner.log中也会打印相关信息。
      2. taskRunner.py调用分布式任务client端，即dispatch/gearclient/gearClient.py接口，gearClient.py负责提交任务到任务调度server，如果提交成功，修改tbl\_monitortask数据表任务status状态为1，log/taskrunner.log中也会打印相关信息。如果提交异常，status修改为2。任务结束。
      3. dispatch/gearwork/gearWork.py中的work事先向dispatch/gearserver/java-gearman-service-0.6.6.jar注册信息，work支持多进程、支持分布式部署。
      4. client端提交任务之后，server端任务队列收到任务，根据之前的注册信息调用相应的远端work进程启动监控任务，work进程根据server端传递的参数调用相应的脚本，同时修改数据库status状态为3执行中。监控任务执行完毕，修改status状态为4,执行成功。Work进程会捕获scripts脚本异常，如果有异常抛出，修改status状态为5执行异常。
      5. 具体的scripts脚本需要继承Monitor.py基类，并且重写其monitor\_alarm\_proc方法，获取监控数据上报到监控数据表&告警数据到告警数据表。
      6. 至此，后端监控任务整个流程结束。
2. 新增/修改监控项操作步骤
   1. tbl\_monitorobj新增监控对象记录，tbl\_monitorobj对应监控树前三级目录
   2. tbl\_monitormetrics新增监控指标，tbl\_monitormetrics对应监控树第四级目录，即最后一级目录
   3. 修改conf/monitor.conf，新增监控项web配置，用于web前端监控树展示配置
   4. stop\_bmeMonitor.sh start\_bmeMonitor.sh重启bmeMonitor.py，展示树刷新，如果没有刷新，检查conf/monitor.conf语法
   5. 新增监控脚本，继承Monitor.py，并且重写其monitor\_alarm\_proc方法，获取监控数据，上报监控告警。
   6. 修改conf/period\_task.conf，新增监控项任务配置，stop\_taskRunner.sh start\_taskRunner.sh重启taskRunner.py。任务生效。
   7. 前端web插件配置在conf/monitor.conf中，开发示例详见template/plugins/1.html等。