# 每日价格更新

命名：dailyPriceUpdate

概要：每天凌晨0点，将所有当天生效的价格更新到产品表中

# 订单拆分

命名：orderDivision

概要：

流程：订单保存后（afterUpdate Hook），如果订单已确定分拣中心，则退出流程，否则，首先建立一个订单数组，之后遍历所有的订单明细，首先将第一条订单明细的分拣中心，设置为订单的分拣中心，并将原始订单加入到订单数组中；然后，后续如果有和订单分拣中心不同的订单明细，则将该订单明细从原始订单中移除，并到订单数组查询是否有分拣中心相符的订单，如果有，则将该订单明细加入到这个订单；如果没有，则新建一个订单，基本信息都从原始订单拷贝，但订单所属分拣中心设置为这个不同的中心，并将订单明细存到这个新建订单中。最后将订单数组中所有的订单都刷新总价，最后都保存一遍。

注意事项：

1. 之所以要在afterUpdate中操作，而不是beforeSave，是因为订单明细是放在relation字段中的，在save之前无法查询。
2. 云函数的开始需要判定是否已确定分拣中心，目的是防止最后的订单保存动作又再次触发该云函数，进入死循环。
3. 结合第2条，如果一个订单的分拣中心已经有值，那么前端需控制，在修改订单时，只能选择该分拣中心的产品。

# 订单状态变更

命名：setOrderStatu

参数：

1. orderOids：需变更的订单ObjectId数组
2. statu：目标状态

参数示例：

{

"orderOids":["560220b960b2af3955a5578f","56021e4160b29460f7f2a804"],

statu:2

}

返回值：

response只有success状态，没有error状态

1. success：数组，其中存放了所有更新成功的订单对象
2. failed：数组，其中存放了所有操作失败的error对象。

每日订货数量更新/每日实际拣货数量更新

流程：订单细节更新后（afterUpdate Hook），如果last Count和orderDetailProductCount的值不同（包括lastCount not exists，这时视lastCount等于0），则计算出两者差值（orderDetailProductCount-lastCount），将差值用原子运算更新到每日订货数量上，更新完毕，将前者的值设为等于后者；如果

根据订单的线路设置订单的配送员