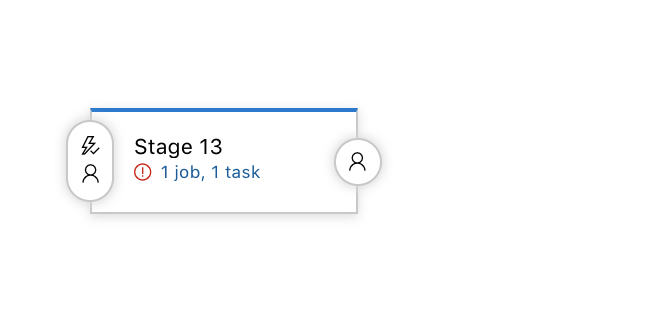
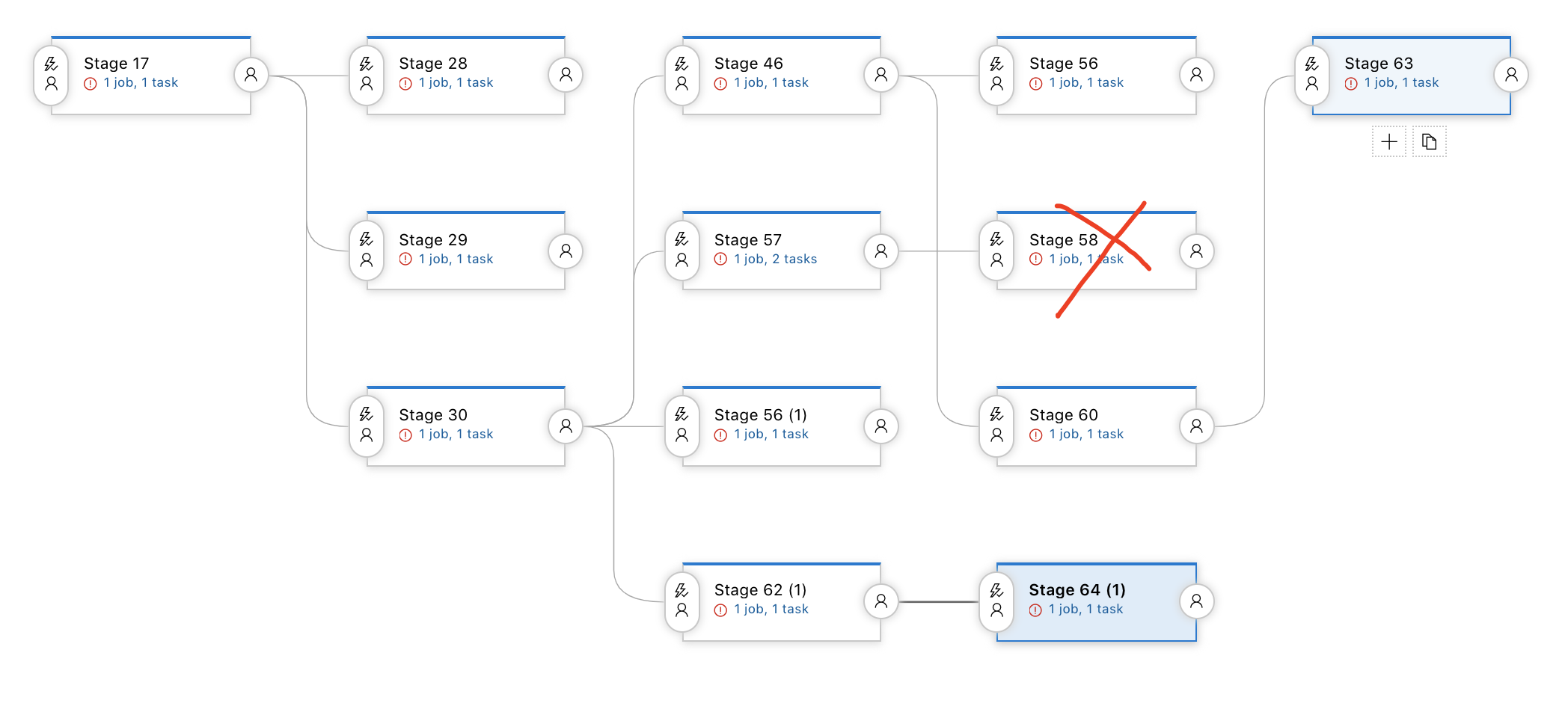
# 节点删除操作

## 如果删除的是唯一节点



直接删除即可

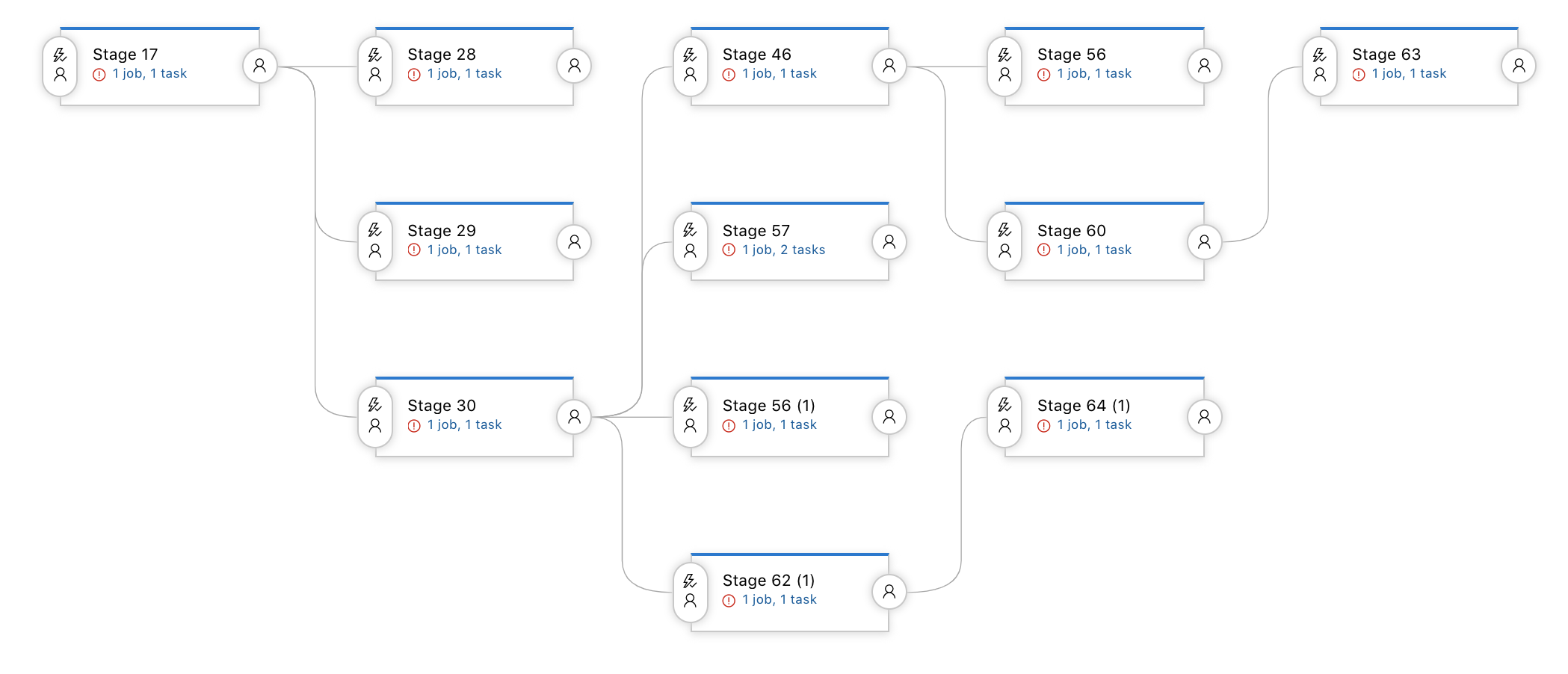
## 如果删除的是叶子节点：



* 只需处理该列的节点：

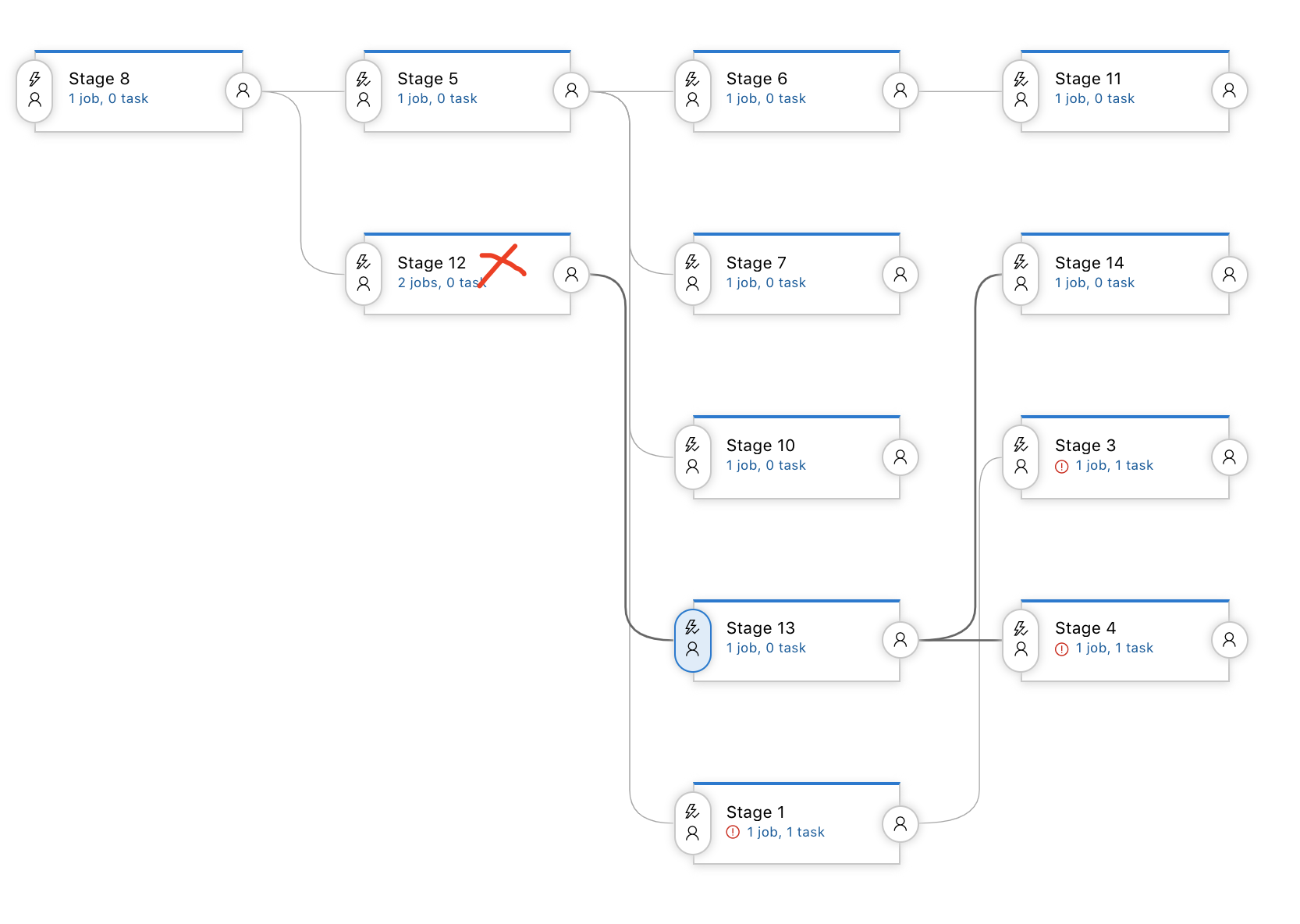
该节点的坐标为x = 3, y = 1=>（1, 3）,遍历该列，更新纵坐标y大于1的节点坐标即可

期望结果是



## 如果删除的节点有子节点

例一：子节点上没有可以产生多条连线的前置节点，或者说节点的入口只有一条



操作：删除Stage12

Azure更新如下



我们期望更新如下：

尽量最小更新节点的位置以及连线

* Stage13替补Stage12的位置（1，1），其子节点Stage14直接替补Stage13的原位置（3，2），如果还有子节点，Stage4在第三列的空行上更新，位置为（5，2）
* Stage13有2个子节点且2个子节点均为叶子节点，因此，叶子节点所在的列x = 3，遍历该列的节点纵坐标，记录空行，更新最后一列的伪代码
* 如果子节点的前置节点还包含其他并行节点，参考

/\*\*

\* 遍历某一列

\* *@param* {\*} col

\* *@param* {\*} row

\*/

searchColNodes(*col*) {

*const* res = []

for (*let* i = 0; i < this.data.length; i++) {

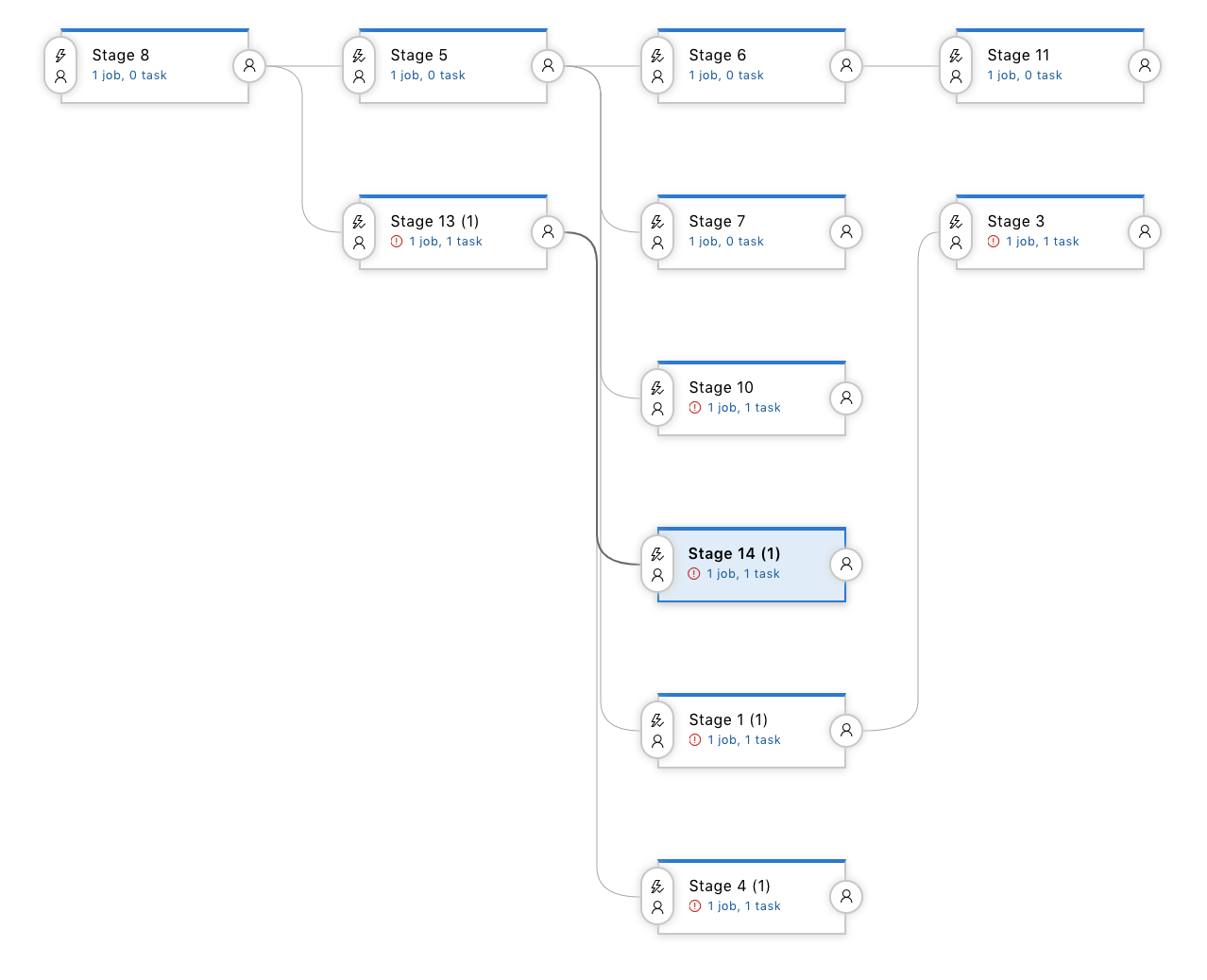
res.push(this.data[i][*col*])

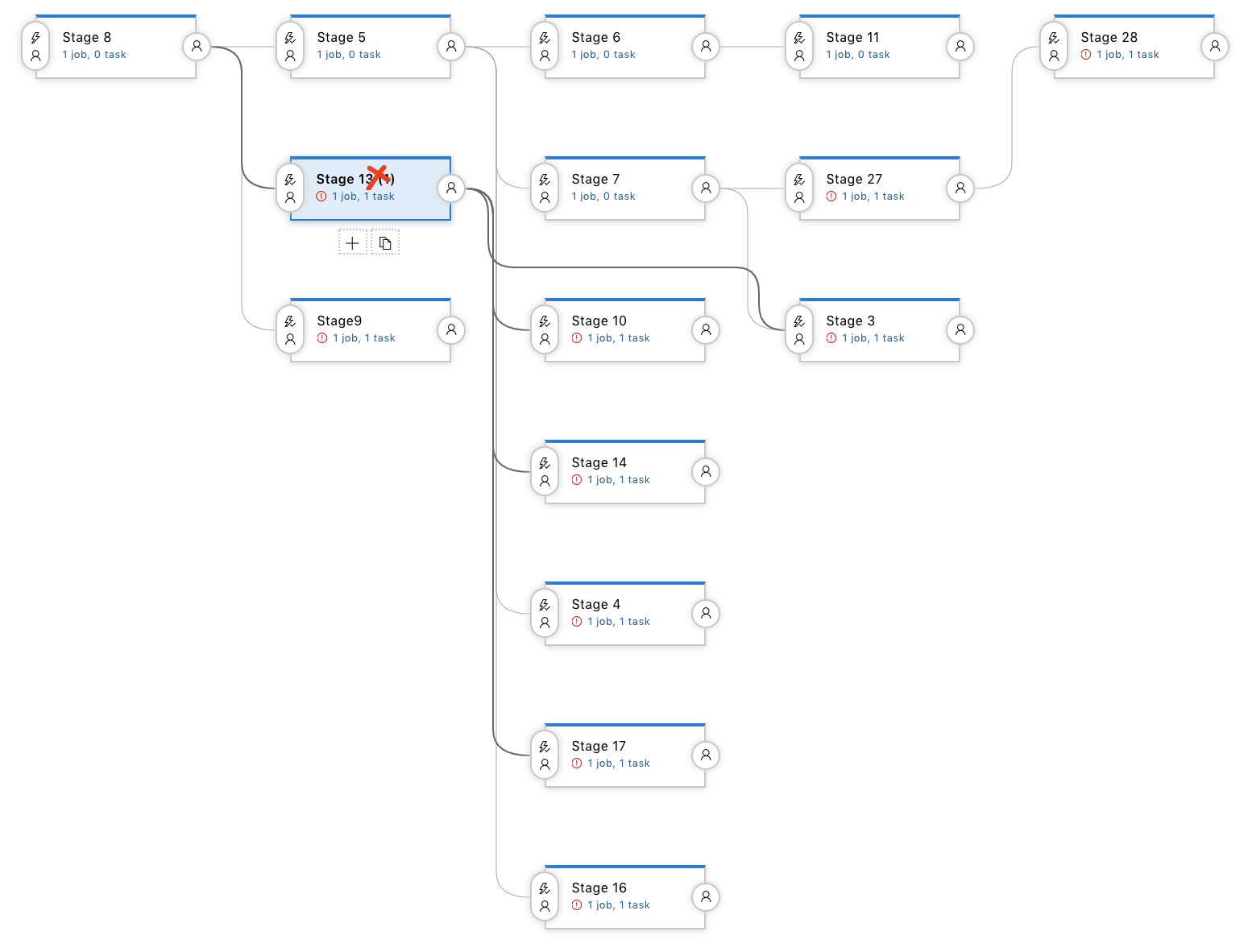
}

return res

}

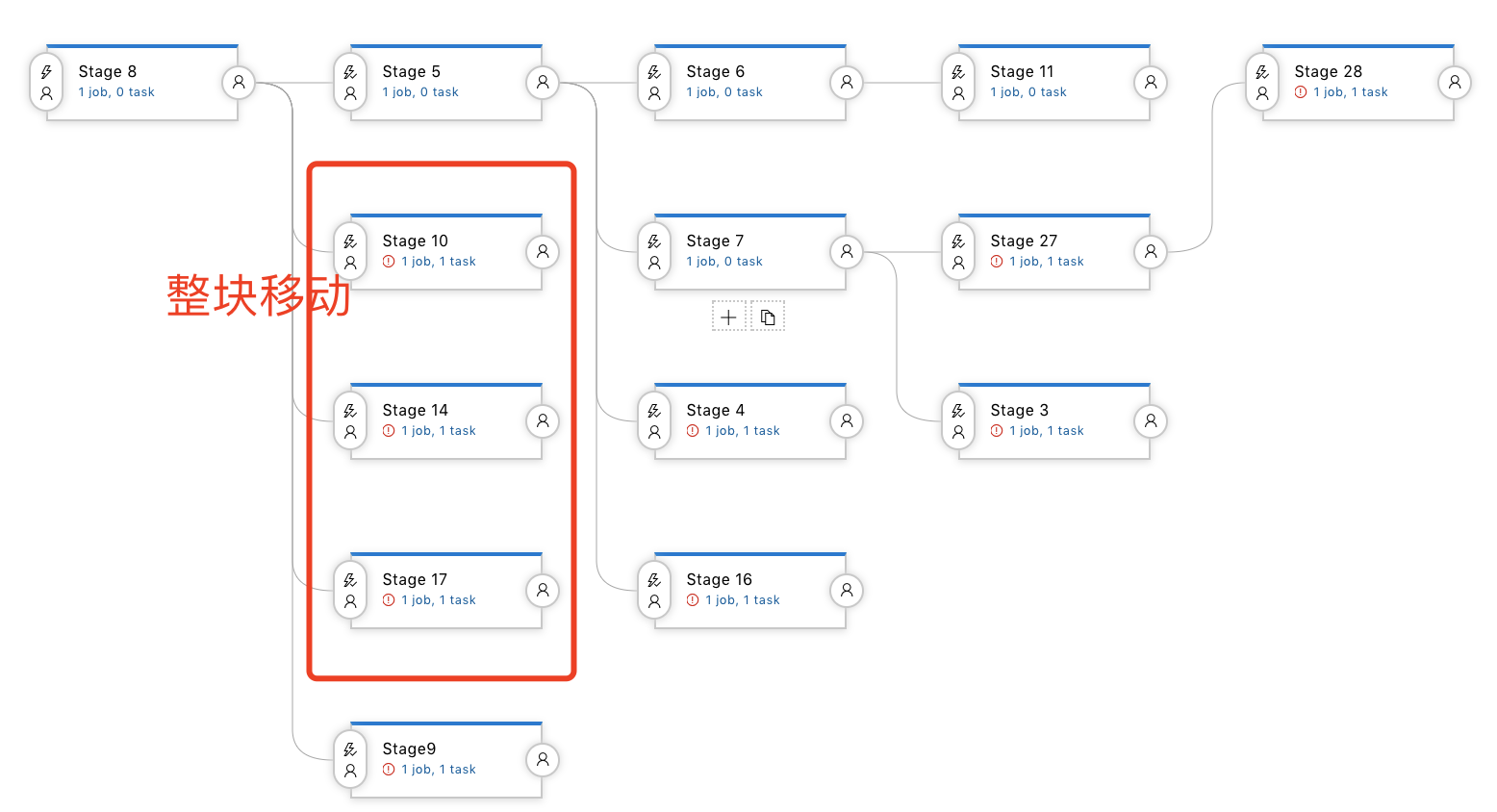
* fn(*col*) {
* *const* emptyRows = []
* for (*let* i = 0; i < this.data.length; i++) {
* if (this.data[i][*col*] && emptyRows.length) {
* *const* { x, y, id } = this.data[i][*col*]
* *const* newRow = emptyRows.shift()
* this.data[newRow][*col*] = {
* x,
* y: newRow,
* id
* }
* } else {
* emptyRows.push(i)
* }
* }
* }

  
例二：子节点上有多个入口



Stage12的直接子节点为stage13

Stage13的子节点Stage3的前置为Stage7和Stage13，如图情况不需要移动该节点，azure表现如下：



该情况下期望得到的结果是：

不移动其他节点的位置

