1. location 正确写法

location = / {

# 精确匹配 / ，主机名后面不能带任何字符串

[ configuration A ]

}

location / {

# 因为所有的地址都以 / 开头，所以这条规则将匹配到所有请求

# 但是正则和最长字符串会优先匹配

[ configuration B ]

}

location /documents/ {

# 匹配任何以 /documents/ 开头的地址，匹配符合以后，还要继续往下搜索

# 只有后面的正则表达式没有匹配到时，这一条才会采用这一条

[ configuration C ]

}

location ~ /documents/Abc {

# 匹配任何以 /documents/Abc 开头的地址，匹配符合以后，还要继续往下搜索

# 只有后面的正则表达式没有匹配到时，这一条才会采用这一条

[ configuration CC ]

}

location ^~ /images/ {

# 匹配任何以 /images/ 开头的地址，匹配符合以后，停止往下搜索正则，采用这一条。

[ configuration D ]

}

location ~\* \.(gif|jpg|jpeg)$ {

# 匹配所有以 gif,jpg或jpeg 结尾的请求

# 然而，所有请求 /images/ 下的图片会被 config D 处理，因为 ^~ 到达不了这一条正则

[ configuration E ]

}

location /images/ {

# 字符匹配到 /images/，继续往下，会发现 ^~ 存在

[ configuration F ]

}

location /images/abc {

# 最长字符匹配到 /images/abc，继续往下，会发现 ^~ 存在

# F与G的放置顺序是没有关系的

[ configuration G ]

}

location ~ /images/abc/ {

# 只有去掉 config D 才有效：先最长匹配 config G 开头的地址，继续往下搜索，匹配到这一条正则，采用

[ configuration H ]

}

location ~\* /js/.\*/\.js

顺序 no优先级：  
(location =) > (location 完整路径) > (location ^~ 路径) > (location ~,~\* 正则顺序) > (location 部分起始路径) > (/)

上面的匹配结果  
按照上面的location写法，以下的匹配示例成立：

* / -> config A：精确完全匹配，即使/index.html也匹配不了
* /downloads/download.html -> config B：匹配B以后，往下没有任何匹配，采用B
* /images/1.gif -> configuration D：匹配到F，往下匹配到D，停止往下

/images/abc/def -> config D  
最长匹配到G，往下匹配D，停止往下  
你可以看到 任何以/images/开头的都会匹配到D并停止，FG写在这里是没有任何意义的，H是永远轮不到的，这里只是为了说明匹配顺序

/documents/document.html -> config C  
匹配到C，往下没有任何匹配，采用C

/documents/1.jpg -> configuration E  
匹配到C，往下正则匹配到E

/documents/Abc.jpg -> config CC  
最长匹配到C，往下正则顺序匹配到CC，不会往下到E