# 镜像启动命令：

docker run -itd -p 8083:8083 {image-name} ./server

image-name最新为image-server-gdal3-etcd3-aws:1.1.2

镜像内部默认8083端口，-v 需要添加数据目录映射

可选默认参数：

--port 8083

--thread 12

--statistic\_size 1024

--gdal\_cache\_size “1000”

--etcd\_v2\_host “127.0.0.1”

--etcd\_v2\_port “2379”

--file\_cache\_dir “/mnt/data/cache”硬盘缓存目录

--etcd\_v3\_address “http://127.0.0.1:4001” 注：如果有多个ip，用逗号隔开

--s3\_cache\_dir “pie-engine-test/cache” 使用s3缓存目录，不设置表示不使用

--s3\_pyramid\_dir “pie-engine-test/make-tile-test” 设置默认s3 pyramid 目录, 使用aws s3自定义金字塔瓦片缓存

如果使用集群部署方式，即开启了多个影像服务，并且设定了"style"，则需要开启etcd。因为渲染"style"需要保存在etcd中。如果只开启一个服务，但是设定了"style"，也是不需要开启etcd的。注：v3只能在linux下用

## 不使用etcd举例：

docker run -itd -p 8083:8083 –v /mnt:/mnt {image-name} ./server

## 使用etcd举例：

启动etcd：

docker run --rm -d --network app-tier --name etcd-server --publish 2379:2379 --publish 2380:2380 --env ALLOW\_NONE\_AUTHENTICATION=yes --env ETCD\_ADVERTISE\_CLIENT\_URLS=http://etcd-server:2379 bitnami/etcd:latest

docker run --rm -d --network app-tier --name etcd-server --publish 2379:2379 --publish 2380:2380 --env ETCD\_MAX\_REQUEST\_BYTES=33554432 --env ALLOW\_NONE\_AUTHENTICATION=yes --env ETCD\_ADVERTISE\_CLIENT\_URLS=http://etcd-server:2379 bitnami/etcd:latest

启动服务：

docker run -d -p8083:8083 --rm --network app-tier -v/mnt:/mnt {image-name} ./server --use\_etcd\_v3 --etcd\_v3\_address <http://etcd-server:2379>

## 使用硬盘（共享存储）缓存

启动服务时，添加参数--file\_cache\_dir 指定缓存的存放目录。服务返回请求瓦片的同时会将该瓦片写入刚才指定的目录，下次请求时，会直接到目录读取并返回请求数据。

例如：docker run -itd -p 8083:8083 {image-name} ./server --file\_cache\_dir "/mnt/data/cache"

## 支持aws s3存储：

数据前缀为 /vsis3或者s3://

### 不使用缓存

docker run -d --rm -p30803:8083 -v/data:/data -e AWS\_REGION=cn-northwest-1 -e AWS\_SECRET\_ACCESS\_KEY=<key> -e AWS\_ACCESS\_KEY\_ID=<key> -e AWS\_S3\_ENDPOINT=s3.cn-northwest-1.amazonaws.com.cn {image-name} ./server

### 使用缓存

可以指定一个bucket用来写入缓存，把请求的瓦片返回的同时，写入s3，下次请求的时候，会直接到s3去找该瓦片

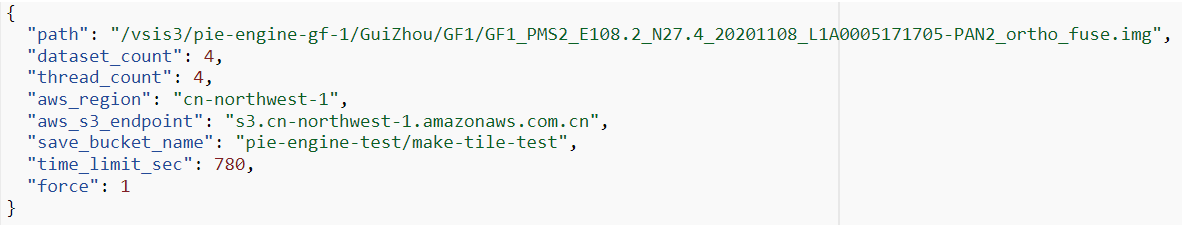
启动：docker run -it --rm -p30014:8083 -v/mnt/data:/mnt/data -e AWS\_REGION=cn-northwest-1 -e AWS\_SECRET\_ACCESS\_KEY=uGXq6F4CXnVsRXTU/bLiBFJLjgpD+MPFrTM+z13e -e AWS\_ACCESS\_KEY\_ID=AKIAT2NCQYSI3X7D52BZ -e AWS\_S3\_ENDPOINT=s3.cn-northwest-1.amazonaws.com.cn {image-name} ./server --s3\_cache\_dir "pie-engine-test/NN/image-server-cache1"

前端默认使用该缓存。若前端想强制不使用该缓存，则需要加参数cache=0或者cache=false，如果不加，则默认使用缓存：http://161.189.202.63:30014/v1?request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}&cache=0&info={"info":[{"path":"/vsis3/pie-engine-test/NN/world.tif"},{"path":"/vsis3/pie-engine-test/NN/mosaic.tif"}]}

瓦片会写入到s3 的pie-engine-test/NN/image-server-cache1下面。瓦片是可以复用的，使用md5加密命名，根据"info"内容和索引号生成的md5命名瓦片

## 使用aws lambda创建s3自定义金字塔瓦片缓存

该算法已经封装为aws lambda上传到了aws。如果使用了aws-tile工具创建了缓存，例如



则参数值 "pie-engine-test/make-tile-test" 需要在服务启动时，作为--s3\_pyramid\_dir参数传入。

。1.3节的缓存是动态生成瓦片的同时写入s3，提升下次访问的速度，并不能提升首次瓦片生成的响应速度，与本节金字塔瓦片不同。aws-tile是预先生成原始数据的切片，可以提升首次及后续访问响应速度。

也可以在url的info参数中显式指定，好处是可以分别指定http://localhost:8083/v1?request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}&info={"info":[{"path":"d:/linux\_share/DEM-Gloable32.tif","s3cachekey":"pie-engine-test/tile-cache"},{"path": "d:/linux\_share/t/1.tiff"}]}

# WMTS前端调用两种方式：

## 前端调用方式一：每次请求时指定请求的数据路径和style(可以省略)，有GET和POST两种方法。支持集群部署，不需要etcd。"info"字段为json数组，支持单张或多张影像。GET和POST内容是一致的，均为json格式。

### GET

http://localhost:8083/v1?request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}&info={info":[{"path":"d:/linux\_share/DEM-Gloable32.tif","style":{"stretch":{"kind":"histogramEqualize","percent":0.0}}},{"path":"d:/linux\_share/t/1.tiff","style":{"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}}],"format":"png"}

format支持"webp","jpg","png"三种，默认是"webp"。style和format可以省略：

http://localhost:8083/v1?request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}&info={"info":[{"path":"d:/linux\_share/DEM-Gloable32.tif"},{"path": "d:/linux\_share/t/1.tiff"}]}

设置影像外部无效值，去黑边

http://10.1.33.45:8083/v1?cache=false&request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}&info={"info":[{"path":"/mnt/hgfs/Z/ai\_data/source\_data/20210314/123/北京市.img","style":{"stretch":{"externalNodataValue":0}}}]}

举例: mapbox前端代码：



### POST

http://localhost:8083/v1?request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}

postbody体内容：{"info":[{"path":"d:/1.tif","style":{"kind":"trueColor", "bandMap" : [1, 2, 3] , "bandCount" : 3, "stretch" : {"kind":"percentMinimumMaximum", "percent" : 3.0}}},{"path":"d:/2.tif", "style" : {"kind":"trueColor", "bandMap" : [1, 2, 3] , "bandCount" : 3, "stretch" : {"kind":"percentMinimumMaximum", "percent" : 3.0}}}]}

## 前端调用方式二：通过接口注册。使用这种方式并且部署多个服务时，需要指定etcd启动；只启动一个服务，不用启动etcd。Windows下etcd需要使用—enable-v2方式启动（etcd默认是以v3方式启动），目前影像服务linux版支持etcd v3，windows版不支持etcd v3。group可以任意指定，根据业务需要来定。调用UpdateDataStyle，产生一个uid，前端把这个uid写到url里，就不用每次都指定数据和style。

### UpdateDataStyle

<http://localhost:8083/v1?request=UpdateDataStyle>

方法：post

body:

{"info":[{"path":"d:/linux\_share/world.tif","style":{"stretch":{"kind":"standardDeviation","scale":0.5}}},{"path":"d:/linux\_share/t/1.tiff","style":{"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}}]}

返回值：B09650397BE37501724001783F22E8C0

url:

[http://localhost:8083/v1?request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}](http://localhost:8083/v1?request=GetTile&TILECOL=%7bx%7d&TILEROW=%7by%7d&TILEMATRIX=%7bz%7d)&key=B09650397BE37501724001783F22E8C0

## 前端调用方式三：AddImages，GetImages，ClearImages ，这三个接口与图层id group名字相关

### AddImages

<http://localhost:8083/v1?request=AddImages>

方法：post

body:

{"group":"一月", "images":["1.tif", "2.tif", "3.tif", "d:/xxx.tif"]}

### GetImages

<http://localhost:8083/v1?request=GetImages>

方法：post

body:

{"group":"一月"}

### SetImages

<http://localhost:8083/v1?request=SetImages>

方法：post

body:

{"group":"一月", "images":["1.tif", "2.tif", "3.tif", "d:/xxx.tif"]}

### GetGroups

<http://localhost:8083/v1?request=GetGroups>

方法：get

### AddGroups

<http://localhost:8083/v1?request=AddGroups>

方法：post

body:

["group1","group2","group3"]

### RemoveGroups

方法：post

body：

["group1","group2","group3"]

### GetGroupEnvelope

<http://localhost:8083/v1?request=GetGroupEnvelope>

方法：post

body:

["groupname1","groupname2"]

### RemoveImages

<http://localhost:8083/v1?request=RemoveImages>

方法：post

body:

{"group":"groupname1", "images":["d:/linux\_share/mosaic.tif","d:/xxx.tif"]}

### ClearImages

<http://localhost:8083/v1?request=ClearImages>

方法：post

body:

{"group":"一月"}

### UpdateStyle

<http://localhost:8083/v1?request=UpdateStyle>&style={styleId}

方法：post

body:

单个：{"style":{"bandCount":3,"bandMap":[3,2,1]}}

或者数组形式：

[{"style":{"bandCount":3,"bandMap":[1,2,3]}}, {"style":{"bandCount":3,"bandMap":[3,2,1]}}]

url参数styleId为自定义唯一值。如果之前未设置styleId，则创建。如果之前添加过styleId，则修改。

说明：单个style，则图层所有数据都使用该style；数组style，对应图层的相应的数据，一一对应

### GetStyle

<http://localhost:8083/v1?request=GetStyle>&style={styleId}

方法：get

返回值：

与UpdateStyle的body内容相同

### GetGroupCacheState

http://localhost:8083/v1?request=GetGroupCacheState

方法：post

Body：["one","two","3"]

返回值：[true,true,false]

### SetGroupCacheState

设置图层缓存是否开启。需要启动时指定缓存路径--file\_cache\_dir或者—s3\_cache\_dir

<http://localhost:8083/v1?request=SetGroupCacheState>

方法：post

body：[{"group":"one","cache":true},{"group":"two","cache":true},{"group":"3","cache":false}]

返回值：ok

### ClearGroupCache

清空图层缓存

<http://localhost:8083/v1?request=ClearGroupCache>

方法：post

Body：["one","two","3"]

返回值：ok

### mapbox前端地址

[http://localhost:8083/v1?layer=groupone&request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}](http://localhost:8083/v1?layer=groupone&request=GetTile&TILECOL=%7bx%7d&TILEROW=%7by%7d&TILEMATRIX=%7bz%7d)

[http://localhost:8083/v1?layer=groupone&style=mystyle&request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}](http://localhost:8083/v1?layer=groupone&style=mystyle&request=GetTile&TILECOL=%7bx%7d&TILEROW=%7by%7d&TILEMATRIX=%7bz%7d) &format=jpg

或者使用info显式指定style：

http://localhost:8083/v1?layer=groupone&request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}&info={"info":[{"style":{"bandCount":3,"bandMap":[1,1,1]}, "s3cachekey":"key"},{"style":{"bandCount":3,"bandMap":[1,2,3]}}]}

# WMS调用接口：

将“request=GetTile&TILECOL={x}&TILEROW={y}&TILEMATRIX={z}”改为“service=wms&request=GetMap&BBOX=0.0,0.0,800.0,600.0&width=500&height=1000”。其中BBOX顺序为left,bottom,right,top。坐标默认为webmecator。其他参数与wmts类似

# 影像信息查询接口：

[http://localhost:8083/v1?request=](http://localhost:8083/v1?request=UpdateDataStyle)GetLayInfo

方法：post

body：

["c:/test/world.tif", "c:/test/abc.tif"]

返回值：

[{"left":0.0,"right":500.0,"top":1000.0,"bottom":0.0,"epsg":4326},{"left":0.0,"right":500.0, "top": 1000.0, "bottom": 0.0, "epsg":-1}]

[http://localhost:8083/v1?request=](http://localhost:8083/v1?request=UpdateDataStyle)GetEnvelope

方法：post

body：

["c:/test/world.tif", "c:/test/abc.tif"]

返回值：格式与上面相同，但是单位是经纬度

# 波段信息查询：

<http://localhost:8083/v1?request=GetImageInfo>

方法：post

body:

["c:/test/world.tif", "c:/test/abc.tif"]

返回值：

[{"pyramid":false,"palette":true,"band\_count":3,"envelope":{"left":-180.000000,"right":180.000000,"top":90.000000,"bottom":-90.000000,"epsg":4326},"bands":[{"name":"B1","min":0.000000,"max":255.000000},{"name":"B2","min":2.000000,"max":255.000000},{"name":"B3","min":0.000000,"max":255.000000}]},{"pyramid":false,"palette":true,"band\_count":4,"envelope":{"left":108.871878,"right":109.338674,"top":34.616937,"bottom":34.229933,"epsg":4326},"bands":[{"name":"B1","min":138.000000,"max":750.000000},{"name":"B2","min":105.000000,"max":937.000000},{"name":"B3","min":66.000000,"max":882.000000},{"name":"B4","min":49.000000,"max":767.000000}]}]

pyramid 影像是否有金字塔文件。palette 影像是否含有颜色对照表。如果palette为true，会有lut字段。含义请参照6.3节lut字段。

# 生成拇指图

<http://localhost:8083/v1?request=GetThumbnail>

方法:post

Body:

{"info":[{"path":"d:/linux\_share/world.tif"}],"width":500,"height":500}

width和height可以省略，默认500。支持多张影像镶嵌显示，wgs84投影

返回值：拇指图图片

# 状态查询接口

http://localhost:8083/v1?request=Inspect

方法：get

返回值：

[{"id":"","tile\_count":440,"max\_qps":6,"average\_time":80.906818,"last\_time":1631695102531}]

id为docker容器id，tile\_count为瓦片数，max\_qps为最大并发量，average\_time 平均瓦片处理时间，last\_time最后一次响应请求的时刻

# 样式说明：

get和post的json格式是一样的，如下例子1,2,3

例子1：

{"info":[{"path":"d:/1.tif","style":{"kind":"trueColor","bandMap":[1, 2, 3] ,"bandCount":3, "stretch":{"kind":"percentMinimumMaximum","percent":3.0}}},{"path":"d:/2.tif","style":{"kind":"trueColor","bandMap":[1,2,3],"bandCount":3,"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}}]}

例子2：

{"info":[{"path":"d:/1.tif","style":{"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}},{"path":"d:/2.tif","style":{"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}}]}

例子3：

{"info":[{"path":"d:/linux\_share/DEM-Gloable32.tif","style":{"stretch":{"kind":"standardDeviation","scale":0.5}}},{"path":"d:/linux\_share/t/1.tiff","style":{"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}}]}

style字段可以省略，则使用默认显示方式，规则为：数据中包含色彩对照表的，使用palette模式；单波段数据，使用dem模式；其他使用trueColor模式。单波段影像使用dem模式可以减少IO，提升效率。数据是否含有颜色对照表，可以通过GetImageInfo的字段palette获取

info的stretch支持下面四种拉伸方式，stretch里kind缺省值为percentMinimumMaximum，percent缺省值为0.0

kind:取值类型 trueColor, palette, dem，缺省值为trueColor

## trueColor，真彩色模式

{"style":{"kind":"trueColor","bandMap":[1,2,3],"bandCount":3,"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}}

{"style":{"kind":"trueColor","bandMap":[1,2,3],"bandCount":3,"stretch":{"kind":"minimumMaximum","minimum":[0.0, 0.0, 0.0],"maximum":[255.0,255.0,255.0]}}}

{"style":{"kind":"trueColor","bandMap":[1,2,3],"bandCount":3,"stretch":{"kind":"histogramEqualize","percent":0.0}}}

{"style":{"kind":"trueColor","bandMap":[1,2,3],"bandCount":3,"stretch":{"kind":"standardDeviation","scale":2.05}}}

可以省略一些字段，例如：

{"style":{"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}}

## palette，颜色对照表模式

{"style":{"kind":"palette"}}。没有其他参数

## dem，颜色带模式

{"style":{"kind":"dem","lut":[[0,255,0,0],[128,0,0,255],[255,0,255,0]],"bandMap":[1],"stretch":{"kind":"percentMinimumMaximum","percent":3.0}}}

stretch参数用法与trueColor相同

lut指定颜色带，[0,255,0,0],对应含义为[索引位置(0到255),颜色R分量，颜色G分量，颜色B分量]。[0,255,0,0]也可以写为[0,255,0,0,255],最后一个255指定颜色alpha(透明度)分量，不指定alpha则默认为255。

最多可以指定256个分组。上例中指定了三个分组，其他颜色通过线性插值来计算。lut的缺省值为从0到255的颜色分量值为0到255，即为从黑到白的颜色条带。

可以写为{"style":{"kind":"dem"}}，其他位置使用缺省值。

下面可以指定高程（数值）-10000对应颜色分组0，即[0,0,0,255]，高程（数值）8000对应颜色分组255，即[255,255,0,0]，其他高程（数值）的颜色由线性插值计算得到

{"info":[{"path":"d:/linux\_share/DEM-Gloable32.tif","style":{"kind":"dem","lut":[[0,0,0,255],[128,0,255,0],[255,255,0,0]],"bandMap":[1],"stretch":{"kind":"minimumMaximum","minimum":[-10000],"maximum":[8000]}}}]}