

TiTiler: Create your own Dynamic Tile server?



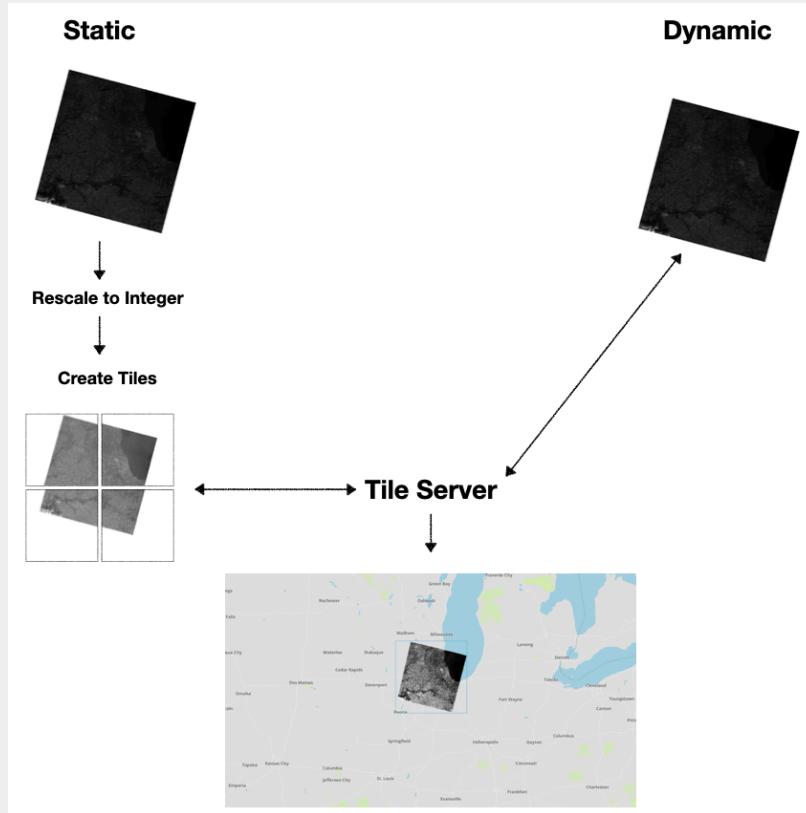
Vincent Sarago

 [vincentsarago](#)

 [@_VincentS_ / @cogeotiff](#)



Dynamic Tiling?



TiTiler



developmentseed/titiler



A modern dynamic tile server built on top of FastAPI and Rasterio/GDAL.

[CI](#) [passing](#) [codecov](#) [96%](#) [pypi package](#) [v0.5.1](#) [license](#) [MIT](#) [launch](#) [binder](#) [docker hub](#) [latest](#)

Documentation: devseed.com/titiler/

Source Code: [developmentseed/titiler](https://github.com/developmentseed/titiler)

It's a set of Python
modules to create
dynamic tiles services.

```
python -m pip install \  
    titiler.core titiler.mosaic titiler.application
```

Features

- Built on top of FastAPI
- [Cloud Optimized GeoTIFF](#) support
- [SpatioTemporal Asset Catalog](#) support
- Multiple projections support (see [TileMatrixSets](#)) via [morecantile](#).
- JPEG / JP2 / PNG / WEBP / GTIFF / NumpyTile output format support
- OGC WMTS support
- Automatic OpenAPI documentation (FastAPI builtin)
- Virtual mosaic support (via [MosaicJSON](#))
- Example of AWS Lambda / ECS deployment (via CDK)



Ready to use

```
uvicorn titiler.application.main:app --reload
```

titiler 0.5.1 OAS3

/openapi.json

A lightweight Cloud Optimized GeoTIFF tile server

Cloud Optimized GeoTIFF

SpatioTemporal Asset Catalog

MosaicJSON

TileMatrixSets

Health Check

Schemas



developmentSEED

Ready to use

The screenshot shows a web-based API documentation interface for the titiler service. At the top, there's a logo for titiler 0.6.1 OAS and an OpenAPI JSON link. Below that, a brief description states: "A lightweight Cloud Optimized GeoTIFF tile server". The main content area is titled "Cloud Optimized GeoTIFF" and lists various API endpoints with their methods and URLs:

- GET /cog/bounds Bounds
- GET /cog/info Info
- GET /cog/info.geojson Info Geojson
- GET /cog/statistics Statistics
- POST /cog/statistics Geojson Statistics
- GET /cog/tiles/{TileMatrixSetId}/{z}/{x}/{y}{#scale}x.{format} Tile
- GET /cog/{TileMatrixSetId}/tilejson.json Tilejson
- GET /cog/tilejson.json Tilejson
- GET /cog/{TileMatrixSetId}/WMTSCapabilities.xml Wmts
- GET /cog/WMTSCapabilities.xml Wmts
- GET /cog/point/{lon},{lat} Point
- GET /cog/preview.{format} Preview
- GET /cog/preview Preview
- GET /cog/crop/{minx},{miny},{maxx},{maxy}/{width}x{height}.{format} Part
- GET /cog/crop/{minx},{miny},{maxx},{maxy}.{format} Part
- POST /cog/crop/{width}x{height}.{format} Geojson Crop
- POST /cog/crop.{format} Geojson Crop
- POST /cog/crop Geojson Crop
- GET /cog/validate Cog Validate
- GET /cog/viewer Cog Demo

```
$ curl http://127.0.0.1:8000/cog/statistics?url=https://ard.maxar.com/samples/v2/colorado-wildfire/13/031113332203/2020-09-21/10300100ABAAB400-visual.tif

{
  "1": {
    "min": 0,
    "max": 255,
    "mean": 66.62077331542969,
    "count": 1048576,
    "sum": 69856944,
    "std": 31.176787274043534,
    "median": 59,
    "majority": 57,
    "minority": 253,
    "unique": 256,
    "histogram": [
      [
        13371,
        282335,
        533309,
        116823,
        41002,
        25664,
        17414,
        12269,
        6103,
        286
      ],
      [
        0,
        25.5,
        51,
        76.5,
        102,
        127.5,
        153,
        178.5,
        204,
        229.5,
        255
      ]
    ],
    "valid_percent": 100,
    "masked_pixels": 0,
    "valid_pixels": 1048576,
    "percentile_2": 28,
    "percentile_98": 174
  },
  "2": {...},
  "3": {...}
}
```



Customize

```
from titiler.core.factory import TilerFactory
from titiler.core.errors import DEFAULT_STATUS_CODES, add_exception_handlers

from fastapi import FastAPI

app = FastAPI(
    title="My super app", description="It's something great",
)

cog = TilerFactory(router_prefix="cog", add_preview=False, add_part=False)
app.include_router(cog.router, prefix="cog", tags=["Cloud Optimized GeoTIFF"])
add_exception_handlers(app, DEFAULT_STATUS_CODES)
```

My super app 0.1.0 OAS3

[openapi.json](#)

It's something great

Cloud Optimized GeoTIFF

<small>GET</small>	/cog/bounds	Bounds	<small>▼</small>
<small>GET</small>	/cog/info	Info	<small>▼</small>
<small>GET</small>	/cog/info.geojson	Info Geojson	<small>▼</small>
<small>GET</small>	/cog/statistics	Statistics	<small>▼</small>
<small>POST</small>	/cog/statistics	Geojson Statistics	<small>▼</small>
<small>GET</small>	/cog/tiles/{TileMatrixSetId}/{z}/{x}/{y}@{scale}x.{format}	Tile	<small>▼</small>



developmentSEED

Tiler Factories

Tiler factories are helper functions that let users create a FastAPI router with a minimal set of endpoints.

```
from fastapi import FastAPI
from titiler.core.factory import TilerFactory
app = FastAPI(description="A lightweight Cloud Optimized GeoTIFF tile server")
cog = TilerFactory()
app.include_router(cog.router, tags=["Cloud Optimized GeoTIFF"])
```

Simple COG

STAC

```
from fastapi import FastAPI
from rio_tiler.io import STACReader # rio_tiler.io.STACReader is a MultiBaseReader
from titiler.core.factory import MultiBaseTilerFactory
app = FastAPI(description="A lightweight STAC tile server")
cog = MultiBaseTilerFactory(reader=STACReader)
app.include_router(cog.router, tags=["STAC"])
```

Multibands
scene

```
from fastapi import FastAPI, Query
from rio_tiler_pds.landsat.aws import LandsatC2Reader # MultiBandReader

from titiler.core.factory import MultiBandTilerFactory

def SceneIDParams(sceneid: str = Query(..., description="Landsat Scene ID")) -> str:
    """Use `sceneid` in query instead of url."""
    return sceneid

app = FastAPI(description="A lightweight Landsat Collection 2 tile server")
cog = MultiBandTilerFactory(reader=LandsatC2Reader, path_dependency=SceneIDParams)
app.include_router(cog.router, tags=["Landsat"])
```

Endpoint dependencies

We use FastAPI dependency injection to define endpoints inputs.

```
import os
import re

from fastapi import FastAPI, HTTPException, Query

from titiler.core.dependencies import DefaultDependency
from titiler.mosaic.factory import MosaicTilerFactory

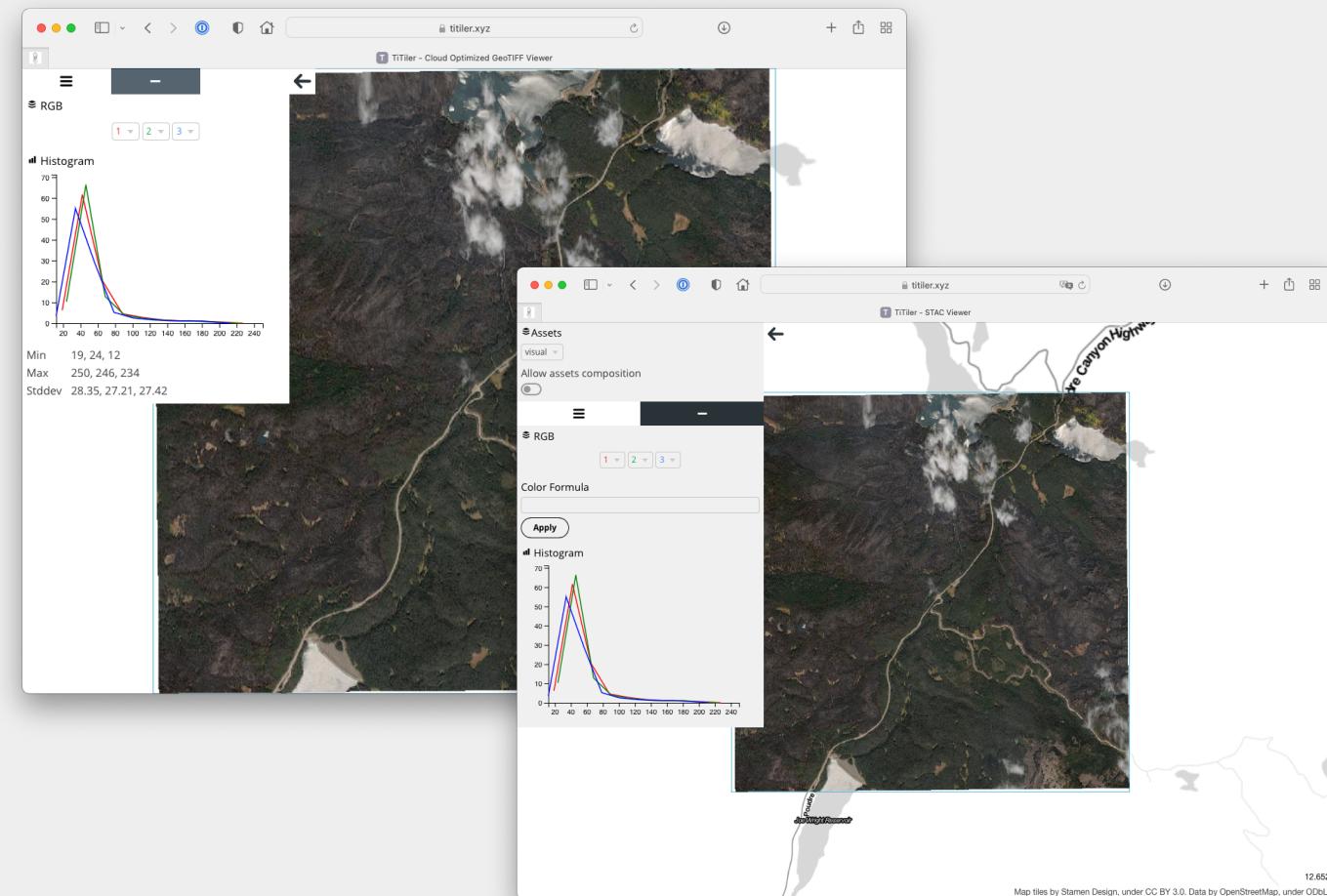
MOSAIC_BACKEND = os.getenv("TITILER_MOSAIC_BACKEND")
MOSAIC_HOST = os.getenv("TITILER_MOSAIC_HOST")

def MosaicPathParams(
    mosaic: str = Query(..., description="mosaic name")
) -> str:
    """Create dataset path from args"""
    # mosaic name should be in form of `{user}.{layername}`
    if not re.match(self.mosaic, r"^[a-zA-Z0-9-_]{1,32}\.{a-zA-Z0-9-_}{1,32}$"):
        raise HTTPException(
            status_code=400,
            detail=f"Invalid mosaic name {self.input}.",
        )

    return f"{MOSAIC_BACKEND}{MOSAIC_HOST}/{self.input}.json.gz"

app = FastAPI()
mosaic = MosaicTilerFactory(path_dependency=MosaicPathParams)
app.include_router(mosaic.router)
```

Viewers



developmentSEED

TiTiler based

TiTiler-PgSTAC

TiTiler ❤ PgSTAC

Connect PgSTAC and TiTiler.

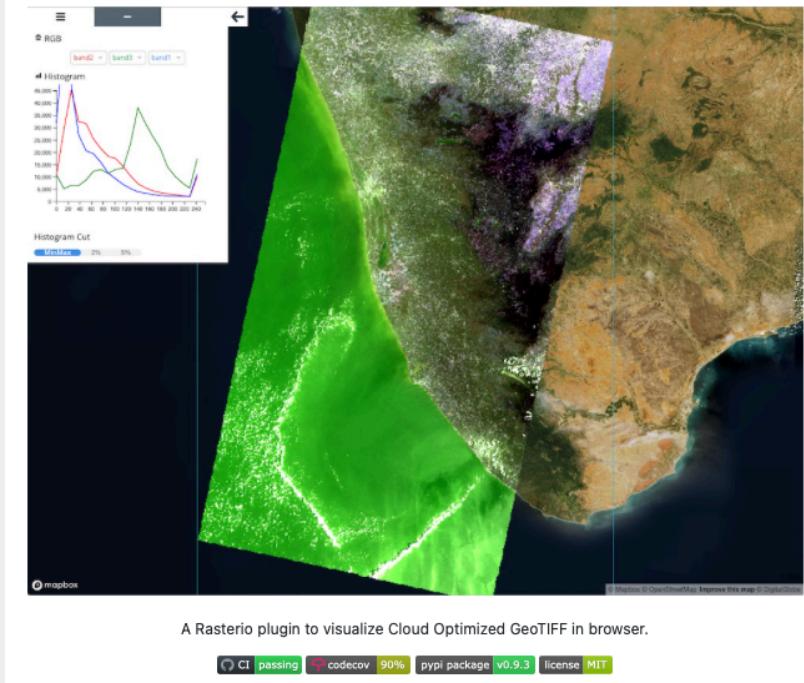


<https://github.com/stac-utils/titiler-pgstac>



developmentSEED

rio-viz



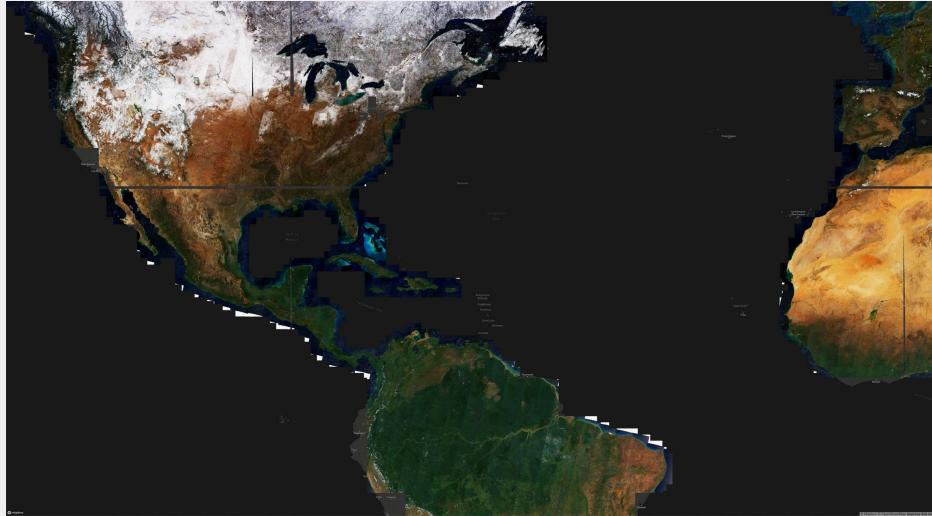
<https://github.com/developmentseed/rio-viz>



developmentSEED

titiler-digitaltwin

TiTiler based



<https://github.com/developmentseed/titiler-digitaltwin>



<https://github.com/developmentseed/titiler>

<https://developmentseed.org/titiler/>

<https://github.com/developmentseed/titiler/discussions>

<https://titiler.xyz>



API documentations: </docs>

TiTiler Online documentations: <https://developmentseed.org/titiler/>

Links

Thanks

Vincent Sarago - @_VincentS_ / @cogeotiff

