TiTiler:
Create your own Dynamic Tile server?

development SEED

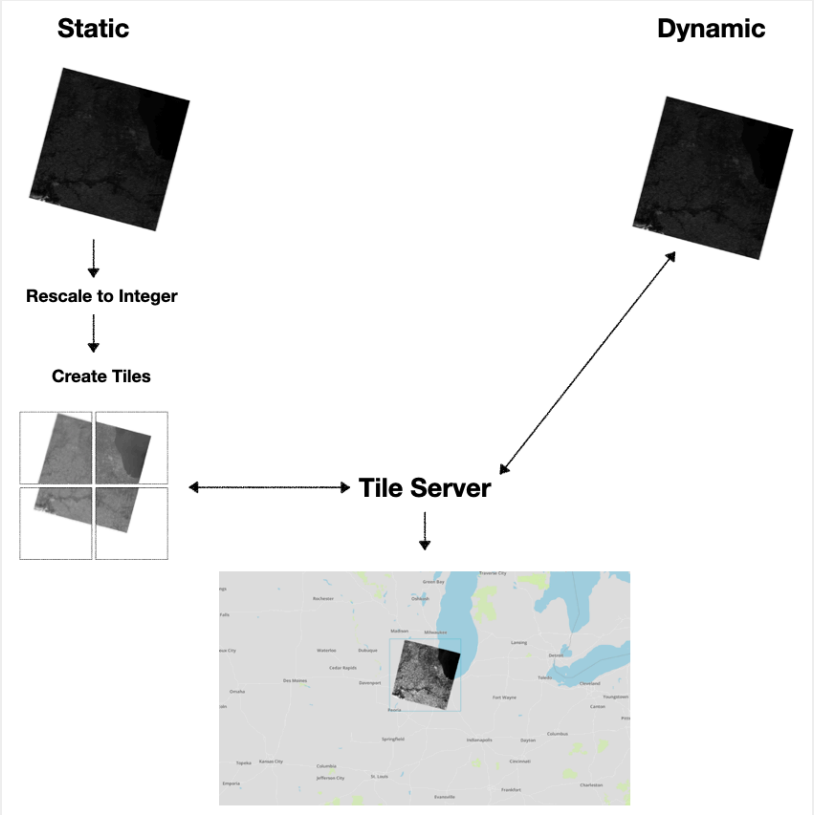
Cloud Native Geospatial Outreach Day - April 19, 2022

Vincent Sarago

vincentsarago

@_VincentS_ / @cogeotiff

Dynamic Tiling?



TiTiler

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
```

Ready to use


```
$ 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

```python
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)
```

Tiler Factories

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

Simple COG

```python
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"])
```

STAC

```python
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

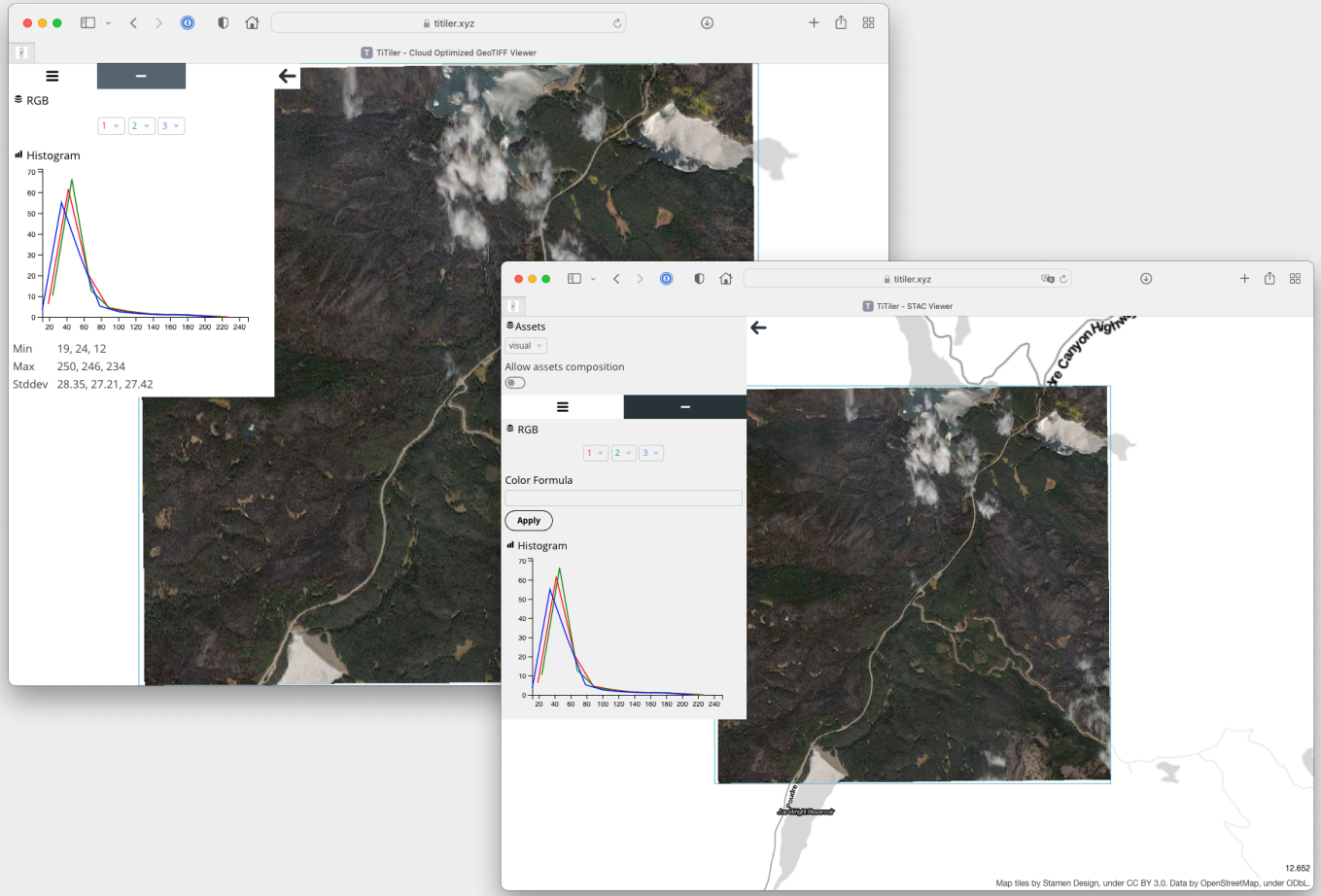
```python
from fastapi import FastAPI, Query
from rio_tiler_pds.landsat.aws import LandsatC2Reader # MultiBandReader

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.

```python
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

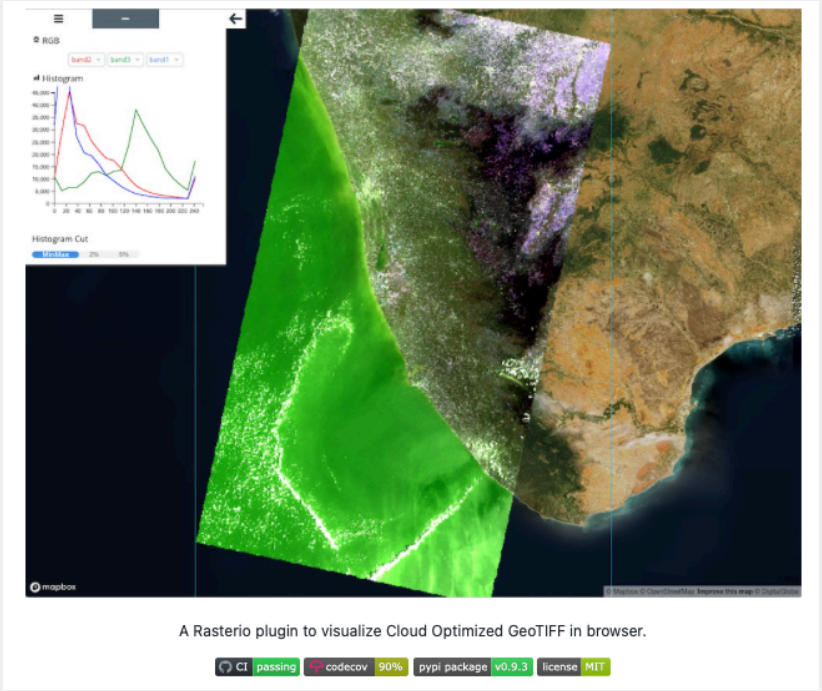
TiTiler based

TiTiler-PgSTAC

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

TiTiler based

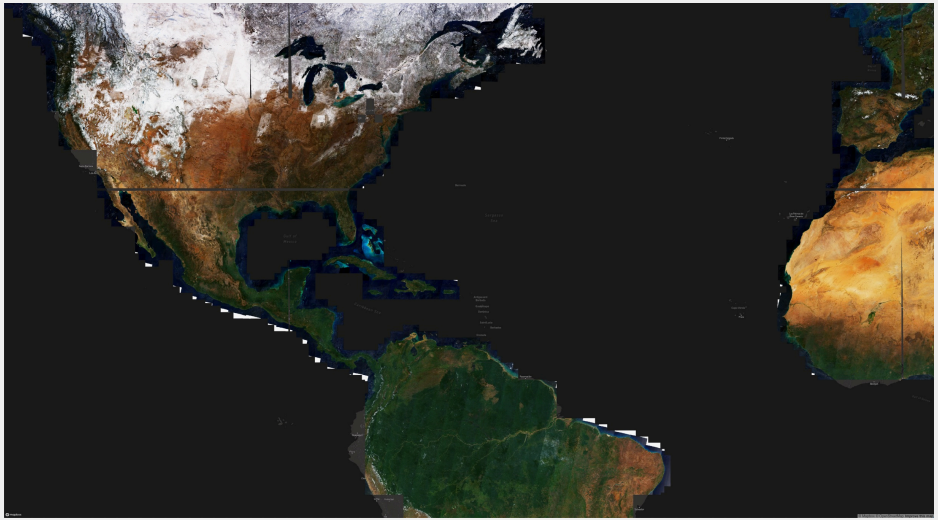
rio-viz



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

TiTiler based

titiler-digitaltwin



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

Links

https://github.com/developmentseed/titiler
https://developmentseed.org/titiler/
https://github.com/developmentseed/titiler/discussions

https://titiler.xyz

Thanks

Vincent Sarago - @_VincentS_ / @cogeotiff

development SEED