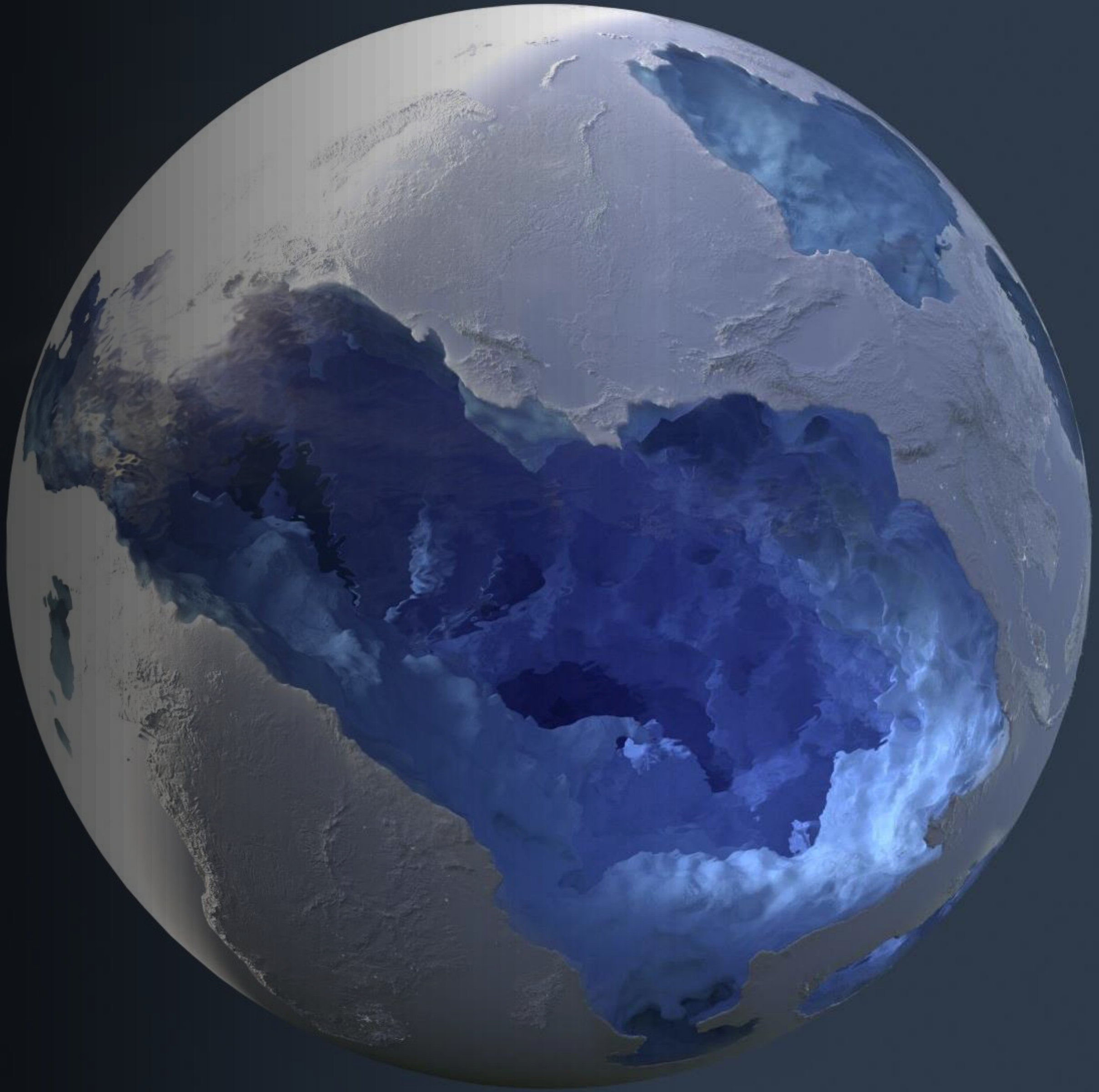
Using COMTiles to reduce the hosting costs of large map tilesets in the cloud

FOSS4G 2022

Markus Tremmel

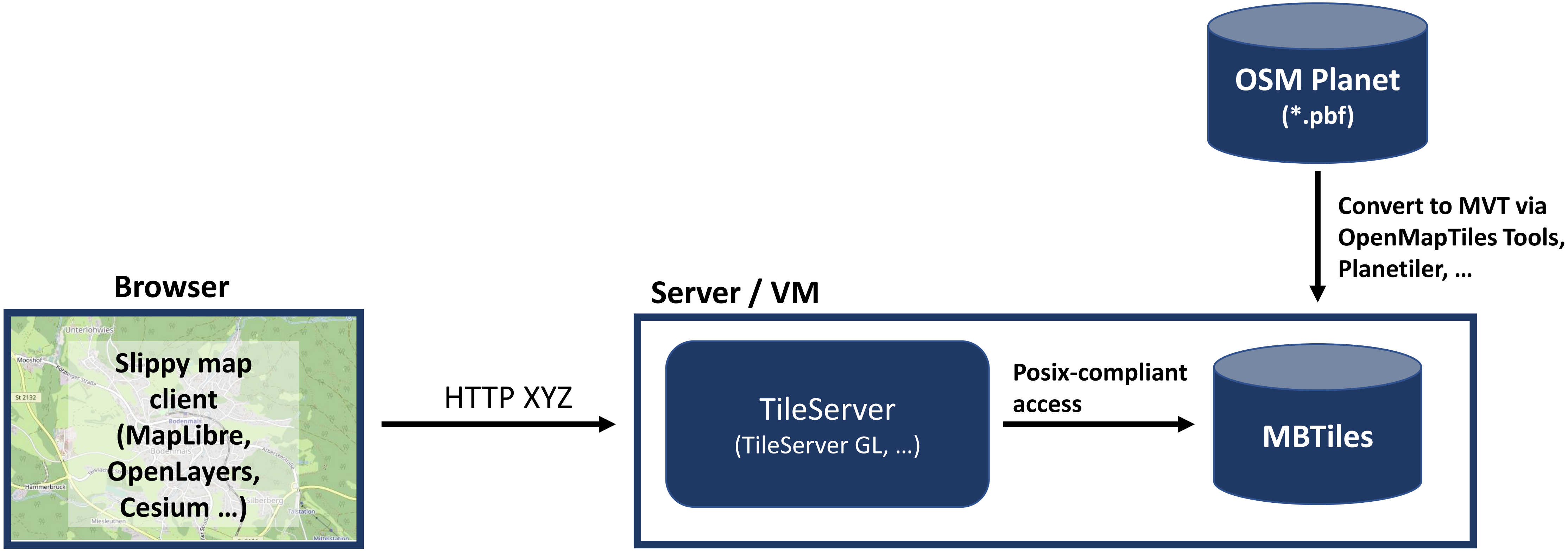


About me

- Geospatial Software Architect at Rohde and Schwarz in Germany
- Lecturer at the Deggendorf Institute of Technology (DIT) in applied computer science with focus on spatial systems
- Main (research) interests: Cloud Native Geospatial, 3D Maps (3DTiles), Map Renderer

DEGGENDORF
INSTITUTE OF
TECHNOLOGY

Conventional deployment of map tilesets



OSM Planet
(*.pbf)

Convert to MVT via
OpenMapTiles Tools,
Planetiler, …

Browser

Slippy map client (MapLibre, OpenLayers, Cesium …)

HTTP XYZ

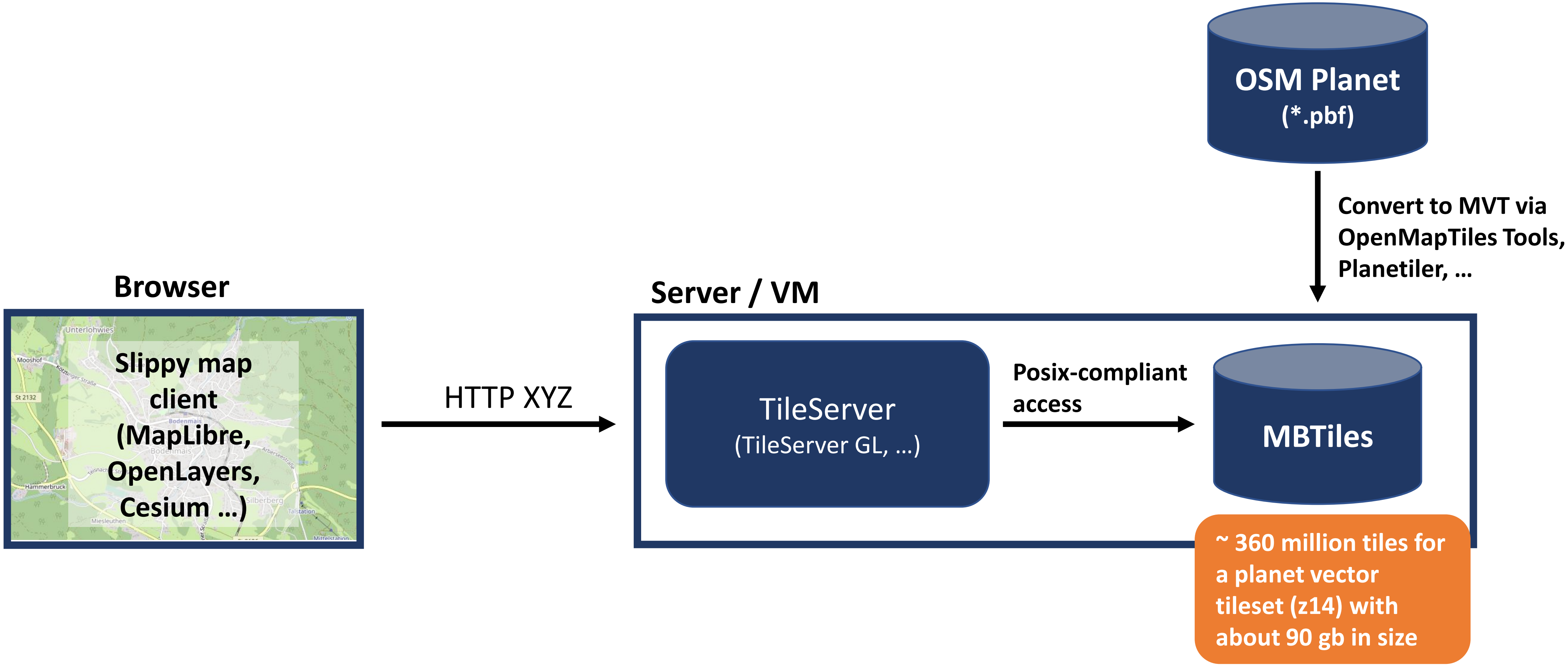
Server / VM

TileServer
(TileServer GL, …)

Posix-compliant access

MBTiles

Conventional deployment of map tilesets



OSM Planet
(*.pbf)

Convert to MVT via OpenMapTiles Tools, Planetiler, …

Browser

Slippy map client (MapLibre, OpenLayers, Cesium …)

HTTP XYZ

Server / VM

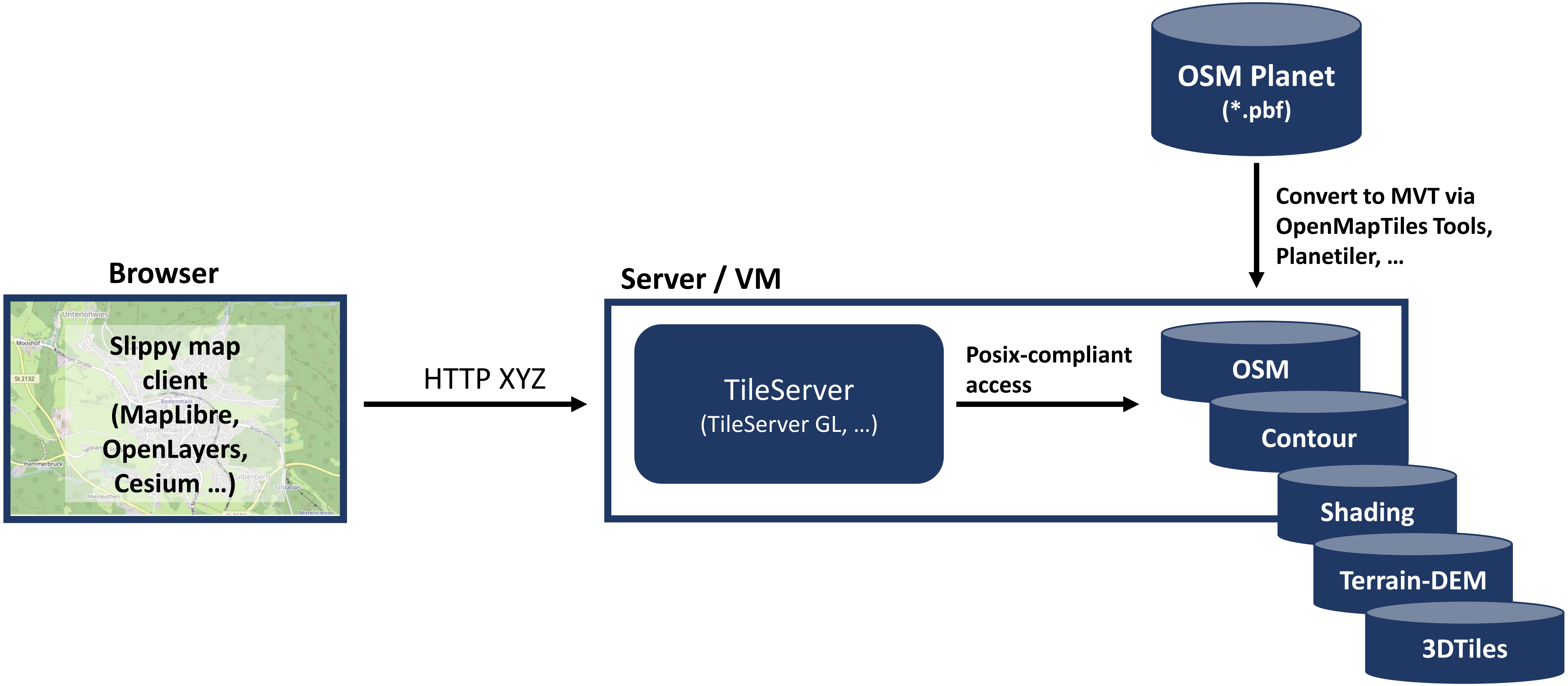
TileServer
(TileServer GL, …)

Posix-compliant access

MBTiles

~ 360 million tiles for a planet vector tileset (z14) with about 90 gb in size

Conventional deployment of map tilesets



OSM Planet
(*.pbf)

Convert to MVT via
OpenMapTiles Tools,
Planetiler, …

Browser

Slippy map
client
(MapLibre,
OpenLayers,
Cesium …)

HTTP XYZ

Server / VM

TileServer
(TileServer GL, …)

Posix-compliant
access

OSM

Contour

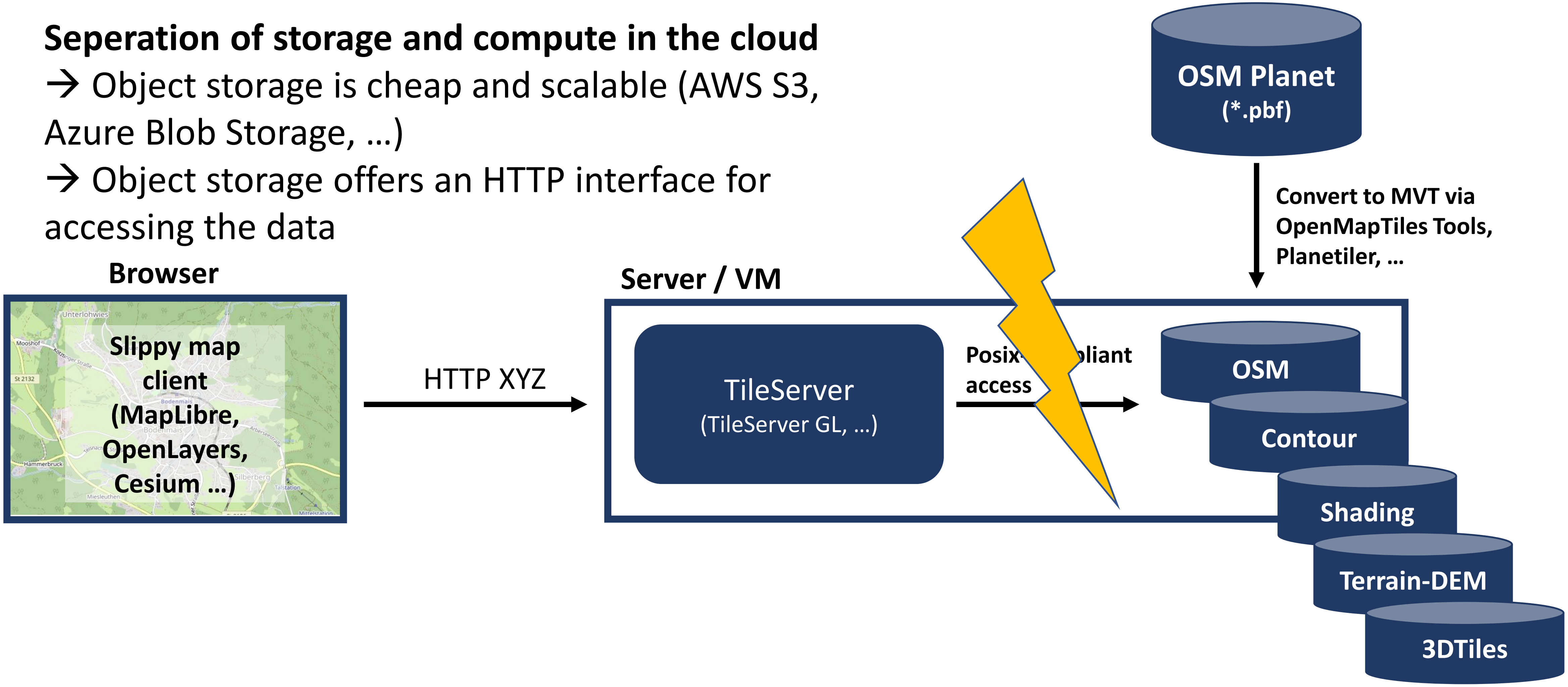
Shading

Terrain-DEM

3DTiles

Conventional deployment of map tilesets

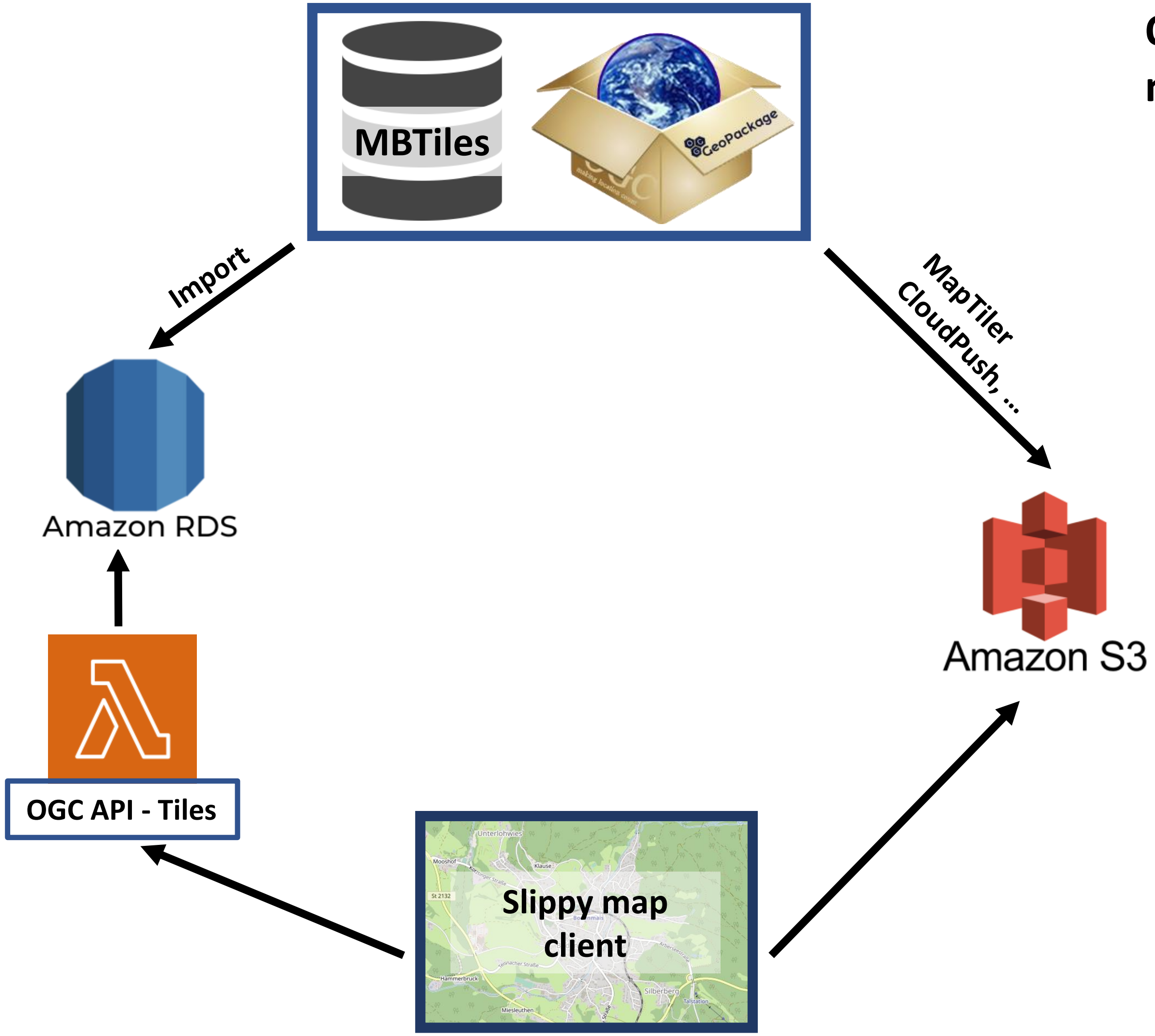
Seperation of storage and compute in the cloud
→ Object storage is cheap and scalable (AWS S3, Azure Blob Storage, …)
→ Object storage offers an HTTP interface for accessing the data



Options for deploying map tilesets in the cloud

Option 1: backend with database and tileserver

Option 2: directly hosting the map tiles on an object storage



MBTiles

GeoPackage

Import

MapTiler CloudPush, …

Amazon RDS

OGC API - Tiles

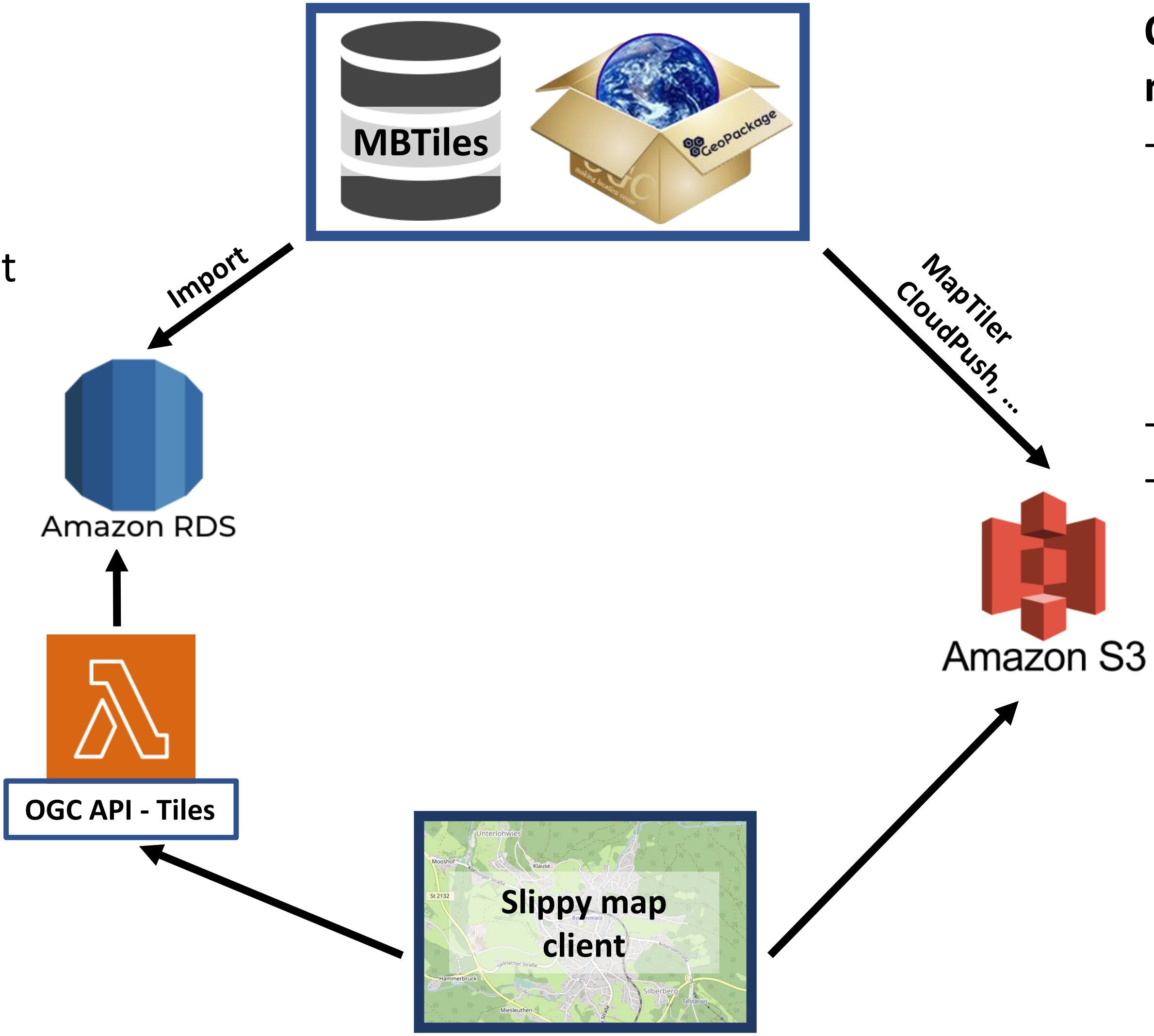
Amazon S3

Slippy map client

Options for deploying map tilesets in the cloud

Option 1: backend with database and tileserver

- significantly more expensive compared to hosting only on an object storage
- …



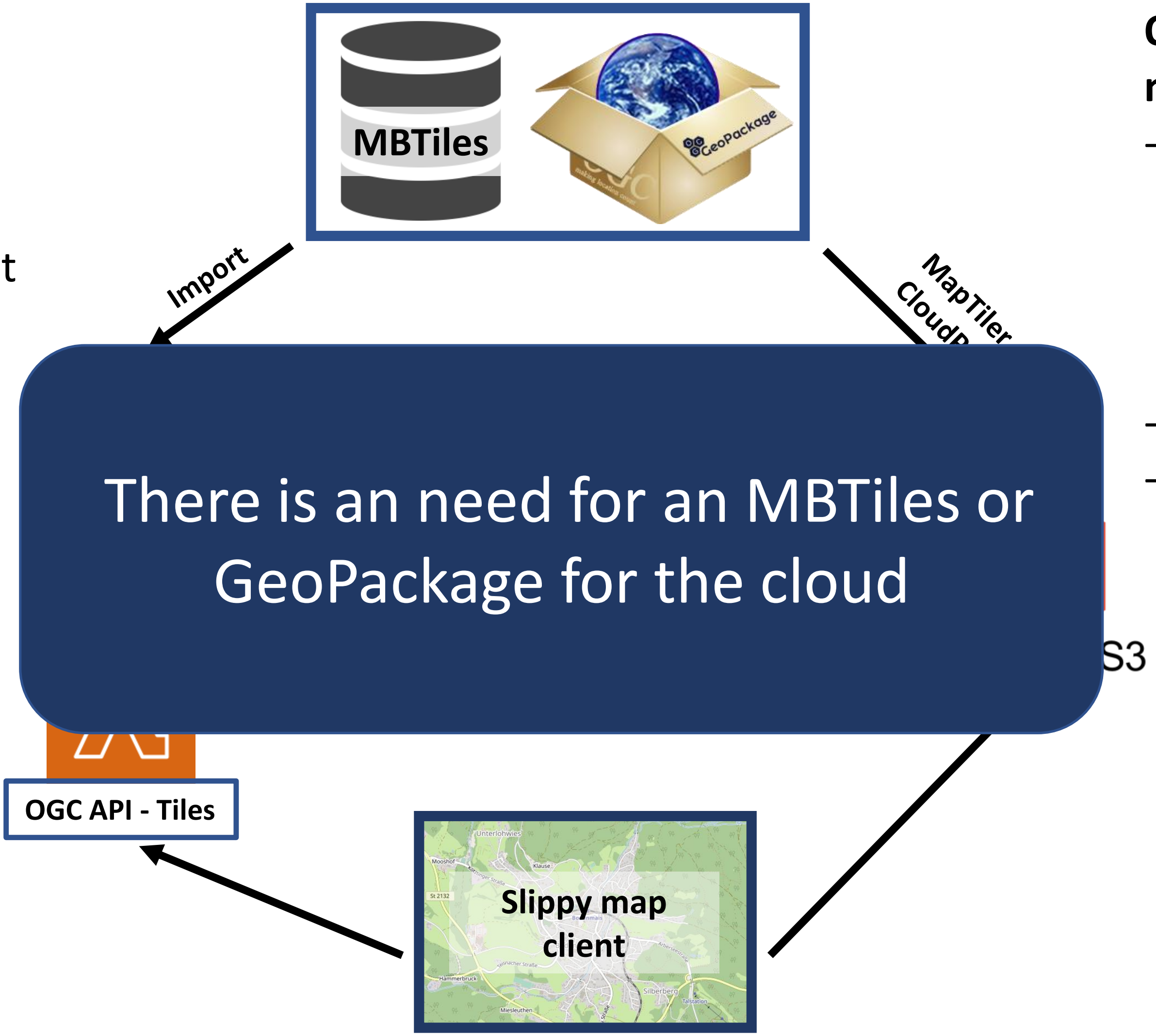
Option 2: directly hosting the map tiles on an object storage

- Loading up hundreds of millions of tiles to an cloud storage for a planet scale vector tiles dataset is expensive
- Many repeated (raster) tiles
- ….

Options for deploying map tilesets in the cloud

Option 1: backend with database and tileserver

- significantly more expensive compared to hosting only on an object storage
- …



There is an need for an MBTiles or GeoPackage for the cloud

Option 2: directly hosting the map tiles on an object storage

- Loading up hundreds of millions of tiles to an cloud storage for a planet scale vector tiles dataset is expensive
- Many repeated (raster) tiles
- ….

Categories of Cloud Optimized Formats

Raster (Satellite)
Cloud Optimized GeoTiff (COG)

Point Clouds
Cloud Optimized Point Cloud

Vector
FlatGeobuf
GeoParquet

N-dimensional Arrays
Zarr
TileDB

Tile Archives
PMTiles
COMTiles

Categories of Cloud Optimized Formats

Raster (Satellite)
Cloud Optimized GeoTiff (COG)

Point Clouds
Cloud Optimized Point Cloud

Vector
FlatGeobuf
GeoParquet

Focus on Analysis
-> (heavy) analytical workflows/processing

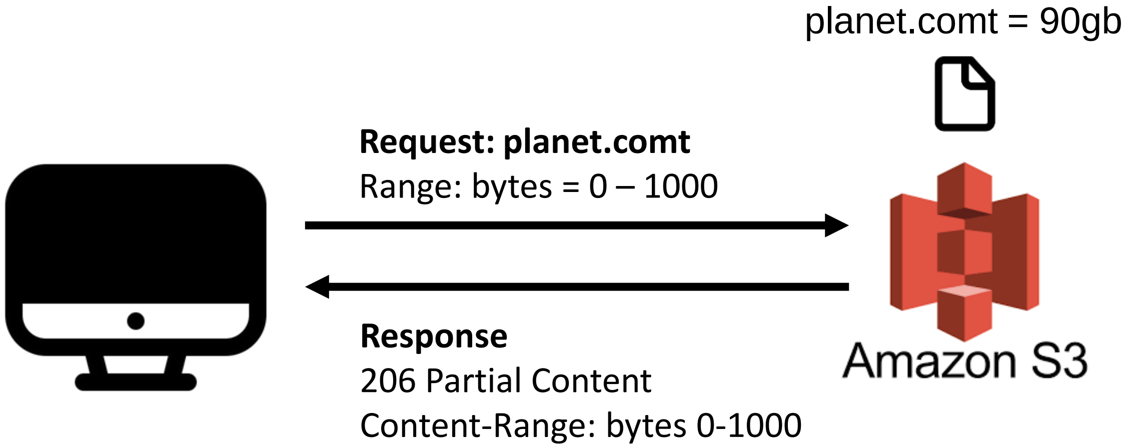
N-dimensional Arrays
Zarr
TileDB

Tile Archives
PMTiles
COMTiles

Focus on Visualization
-> “Slippy map user experience“

Cloud native principles

- Objects storage like AWS S3
- Read portions (chunks) of a file via HTTP range requests
- (streamable) spatial index
- Metadata for describing the tileset



planet.comt = 90gb

Request: planet.comt
Range: bytes = 0 – 1000

Response
206 Partial Content
Content-Range: bytes 0-1000

COMTiles

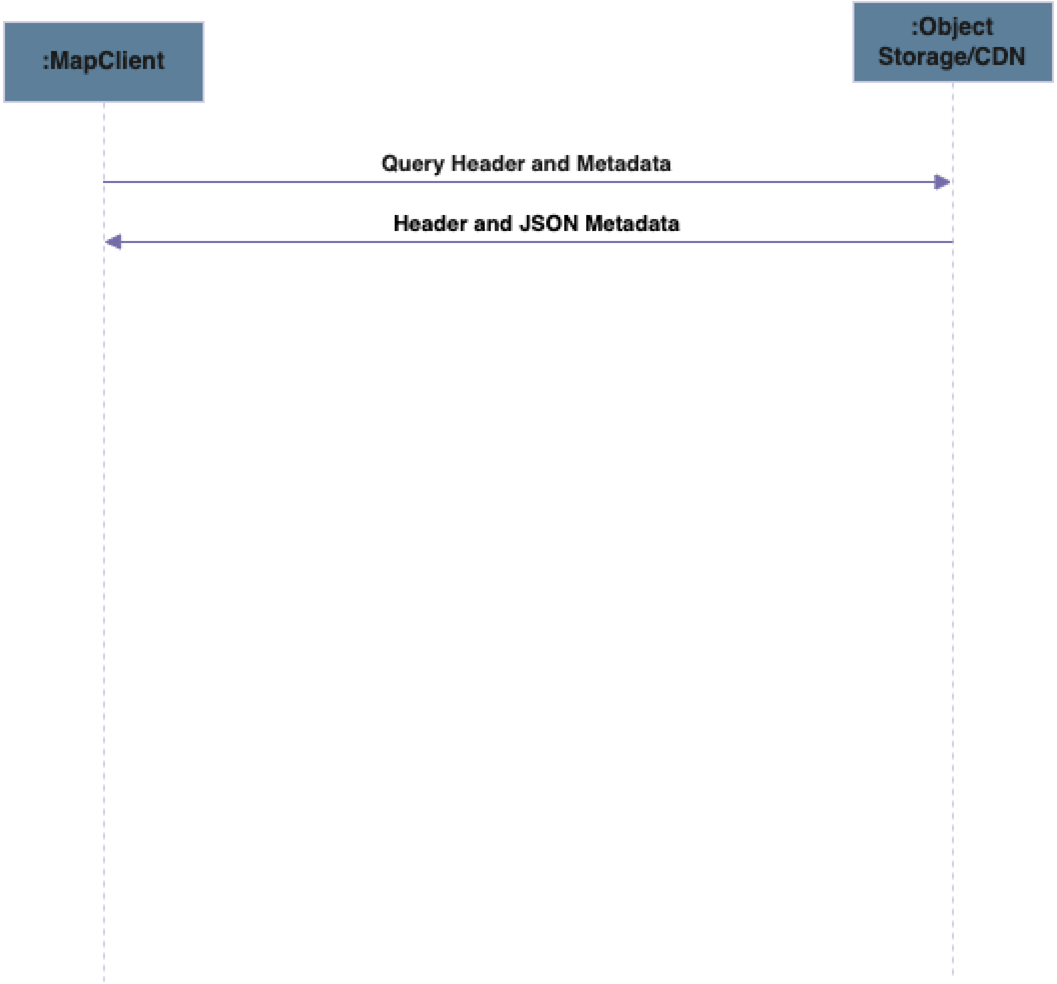
Based on the ideas of Cloud Optimized GeoTIFF and extended for the usage of raster and in particular vector map tilesets

COMTiles are a streamable and read optimized file archive for hosting map tiles at global scale on a cloud object storage

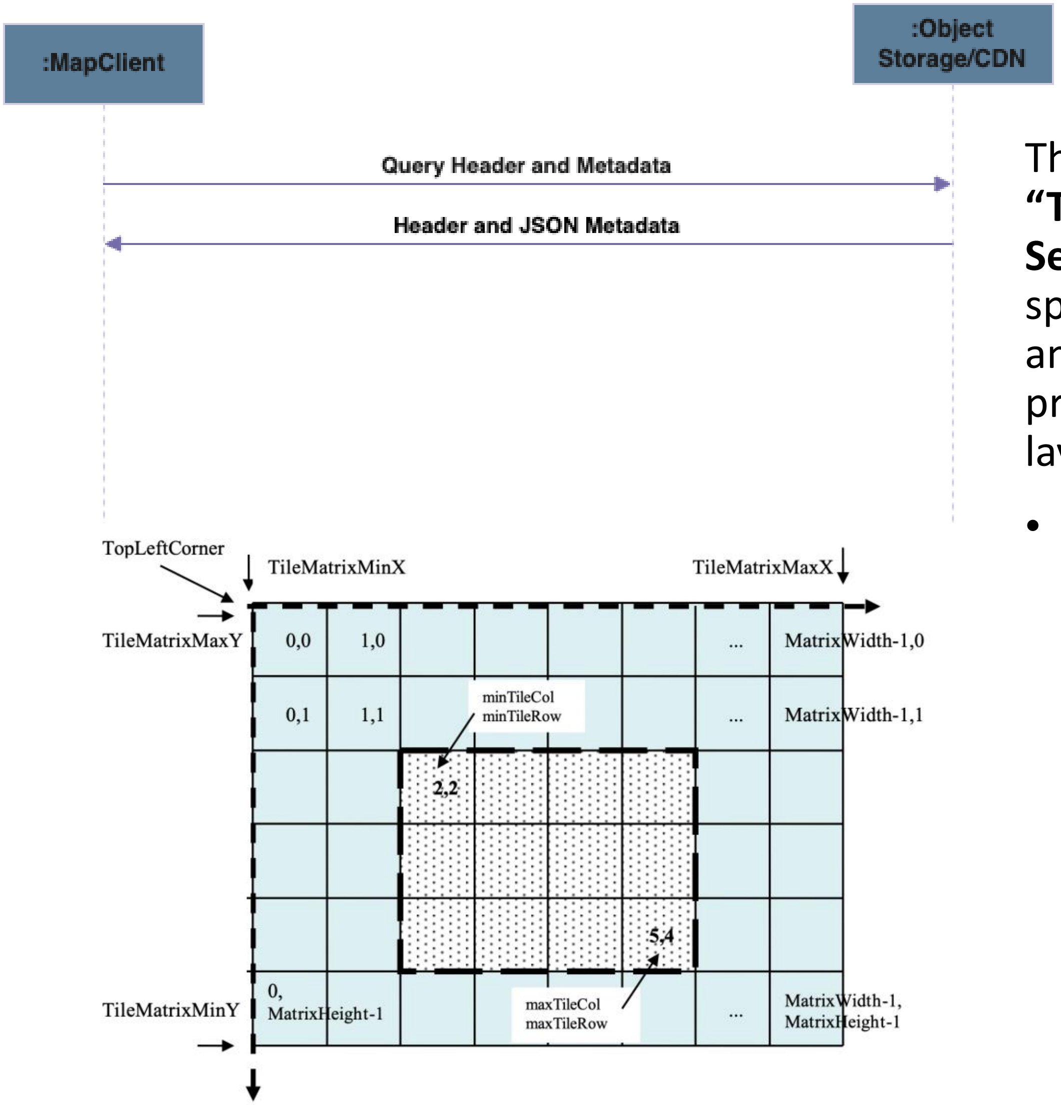
The map tiles can be accessed directly from a browser via HTTP GET range requests

The main focus of COMTiles is to significantly reduce costs and simplify the hosting of large raster and vector tilesets at global scale in the cloud because no backend (database, server) is needed

COMTiles sequence



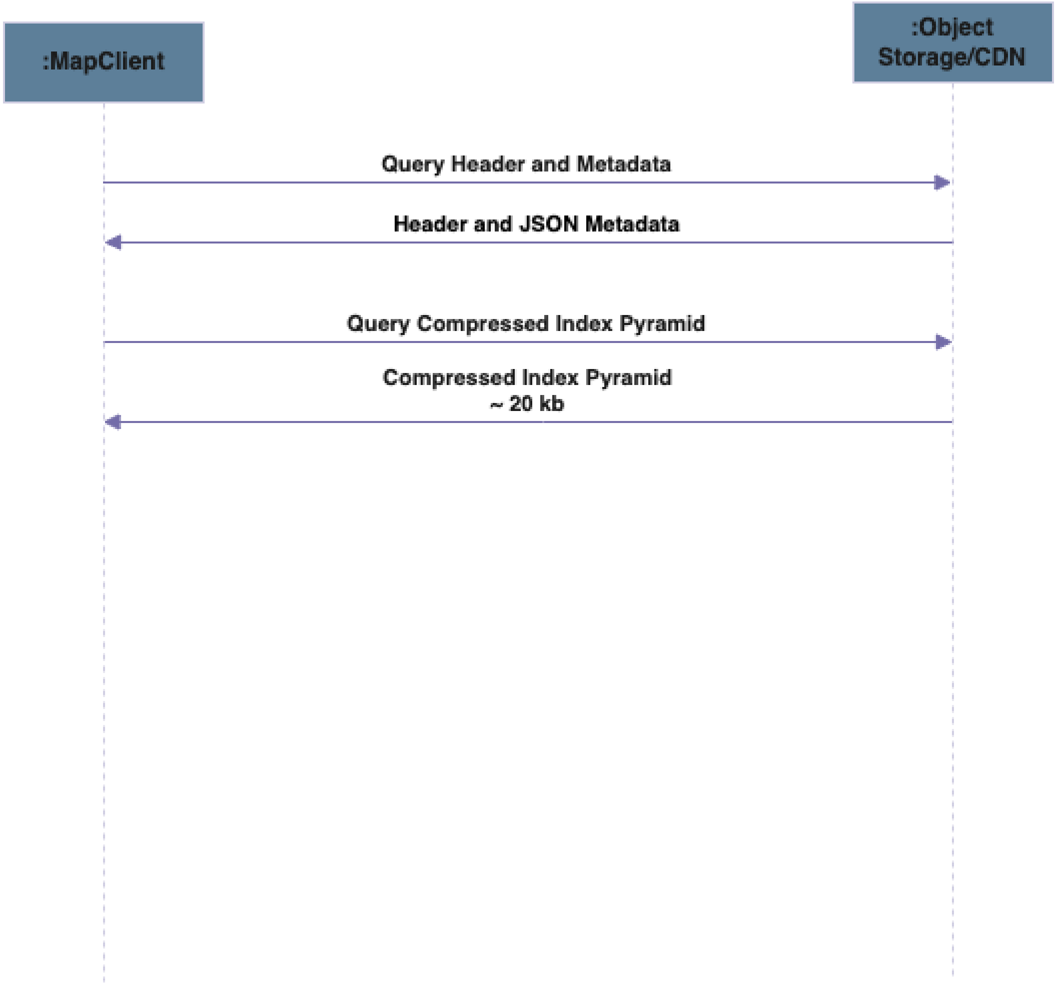
COMTiles sequence



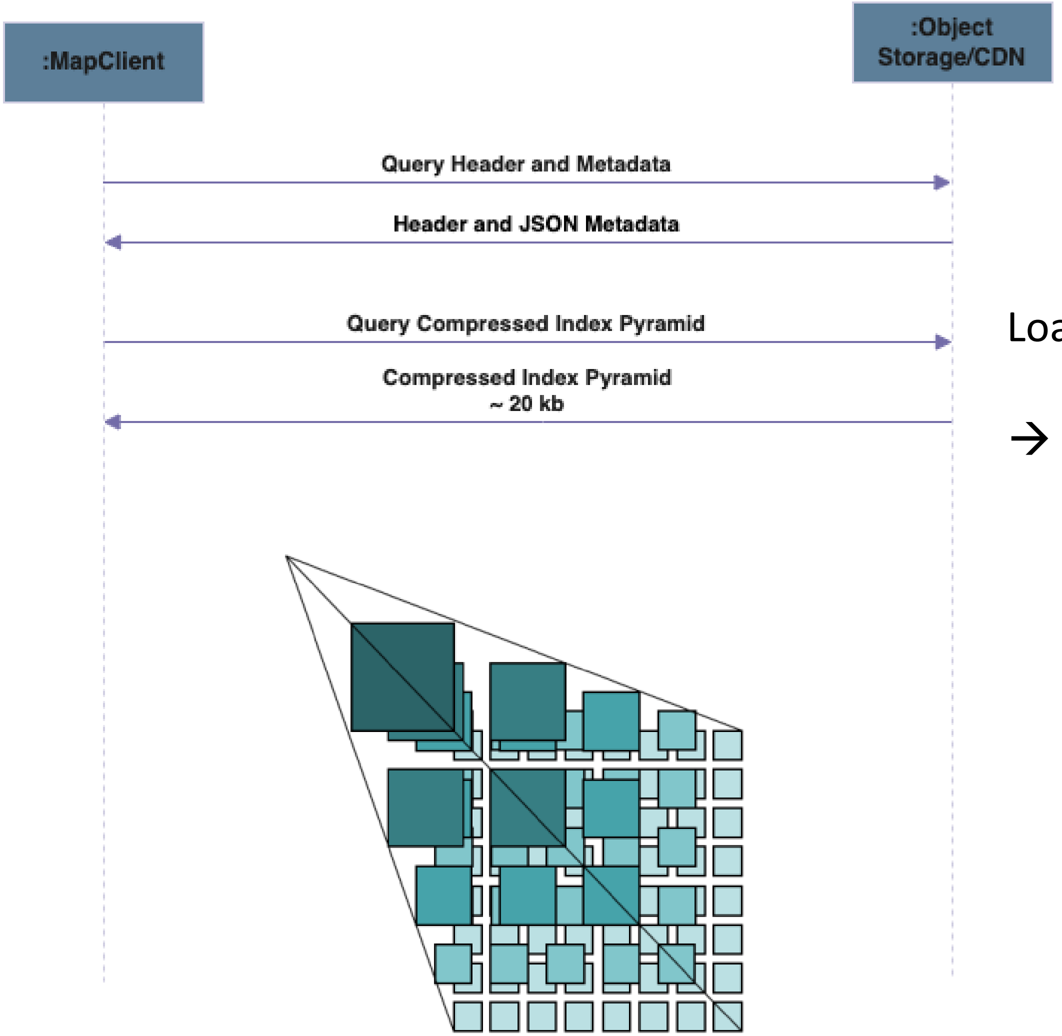
The metadata are based on the **“Two Dimensional Tile Matrix Set and Tile Set Metadata“** specification (currently draft) and extend it with additional properties about the index layout

- Concepts
  - **TileSet metadata** -> attribution, crs, layers, dataType, …
  - **TileMatrixSet** -> Common TileMatrixSetDefinitions like WebMercatorQuad
  - **TileMatrixSetLimits** -> minTileRow, maxTileRow, minTileCol, maxTileCol, …

COMTiles sequence



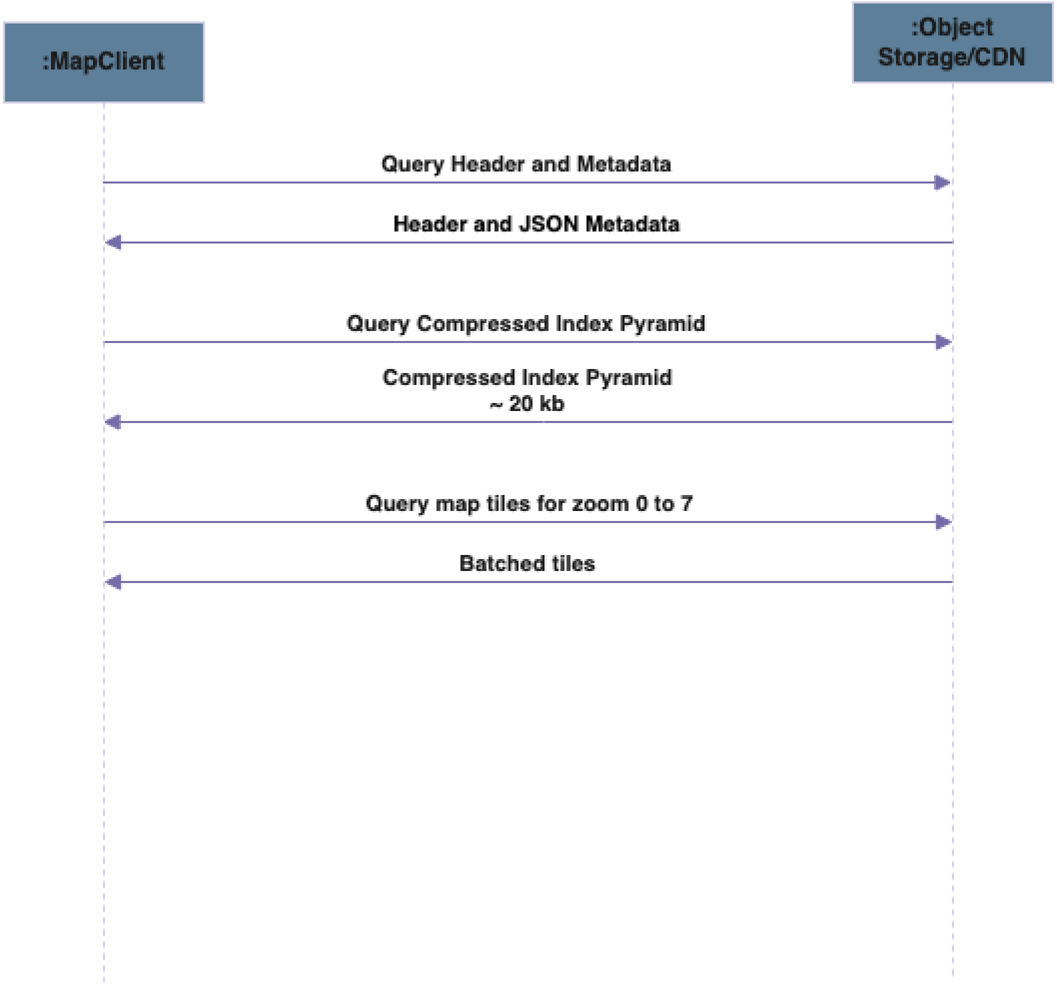
COMTiles sequence



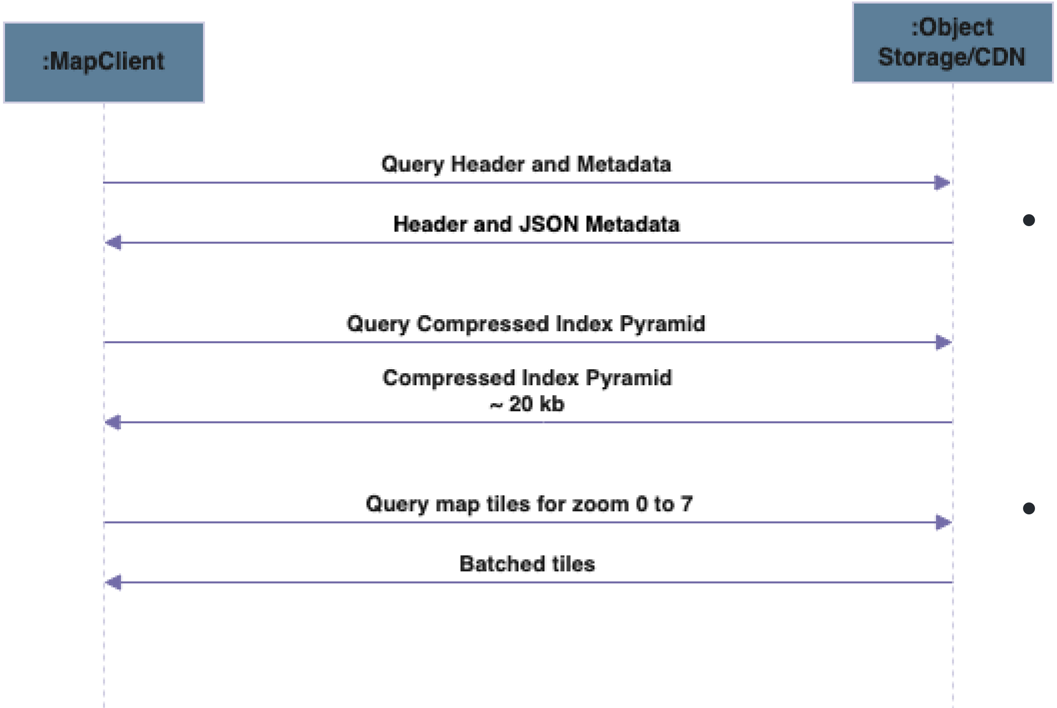
Load overview index records for exploring the map

→ ~ 20k index records for zoom 0 to 7 for a planet tileset

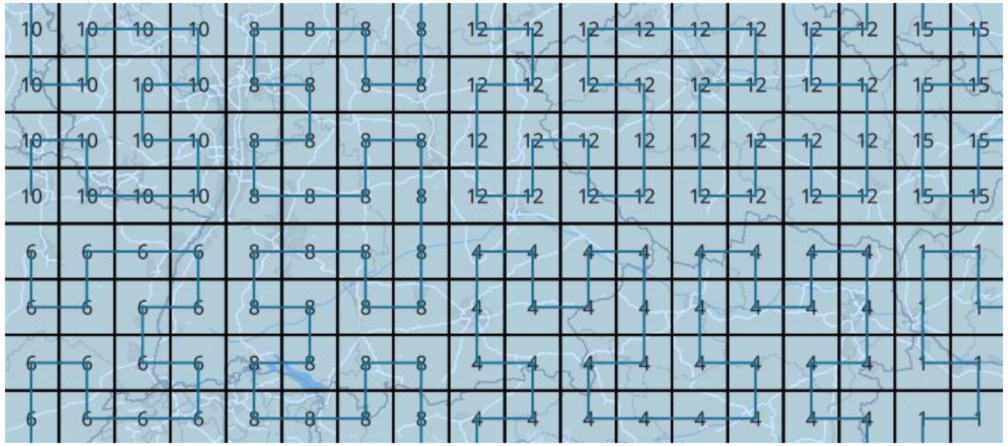
COMTiles sequence



COMTiles sequence

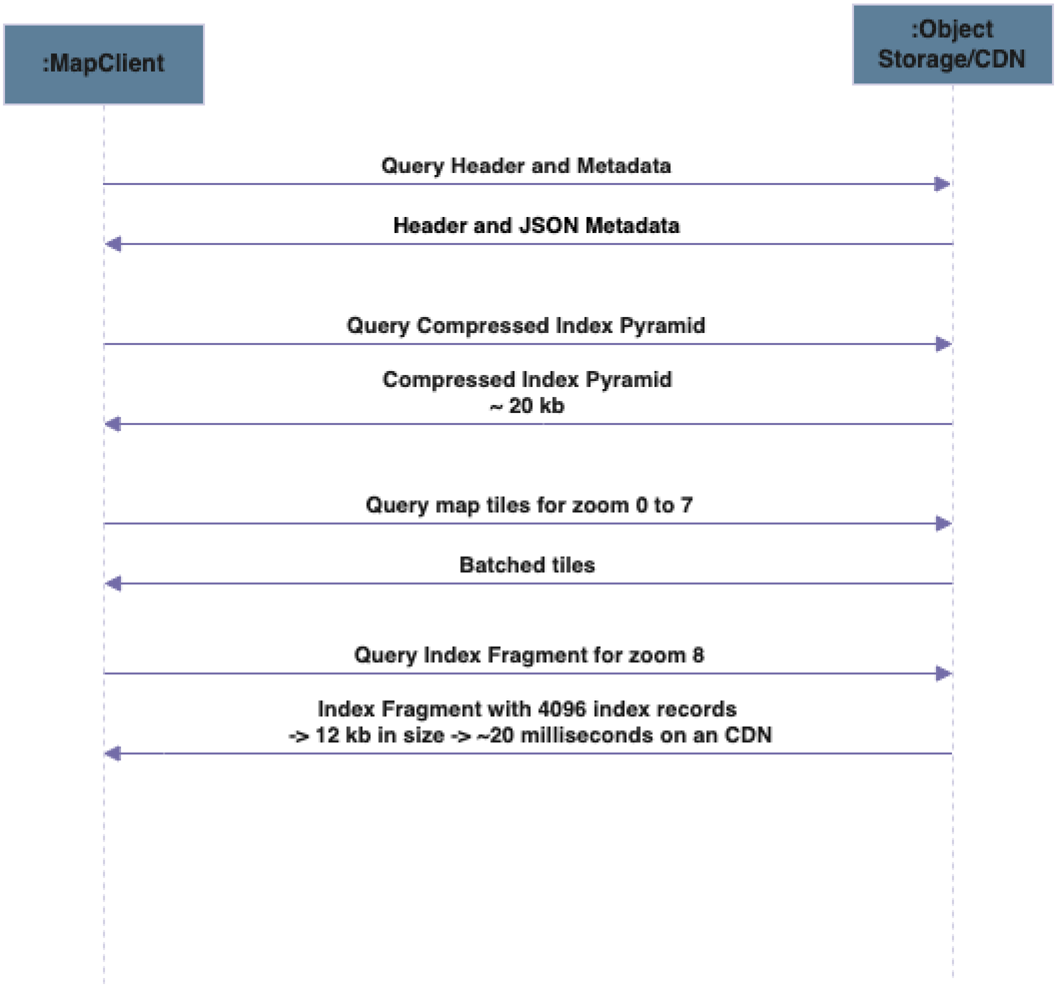


:MapClient
:Object Storage/CDN
Query Header and Metadata
Header and JSON Metadata
Query Compressed Index Pyramid
Compressed Index Pyramid ~ 20 kb
Query map tiles for zoom 0 to 7
Batched tiles



- The individual tile requests can be batched to improve performance (for HTTP/1.1 requests) and in particular to reduce the transfer costs
- This can reduce the number of tile requests by up to 90%

COMTiles sequence



:MapClient

:Object Storage/CDN

Query Header and Metadata

Header and JSON Metadata

Query Compressed Index Pyramid

Compressed Index Pyramid
~ 20 kb

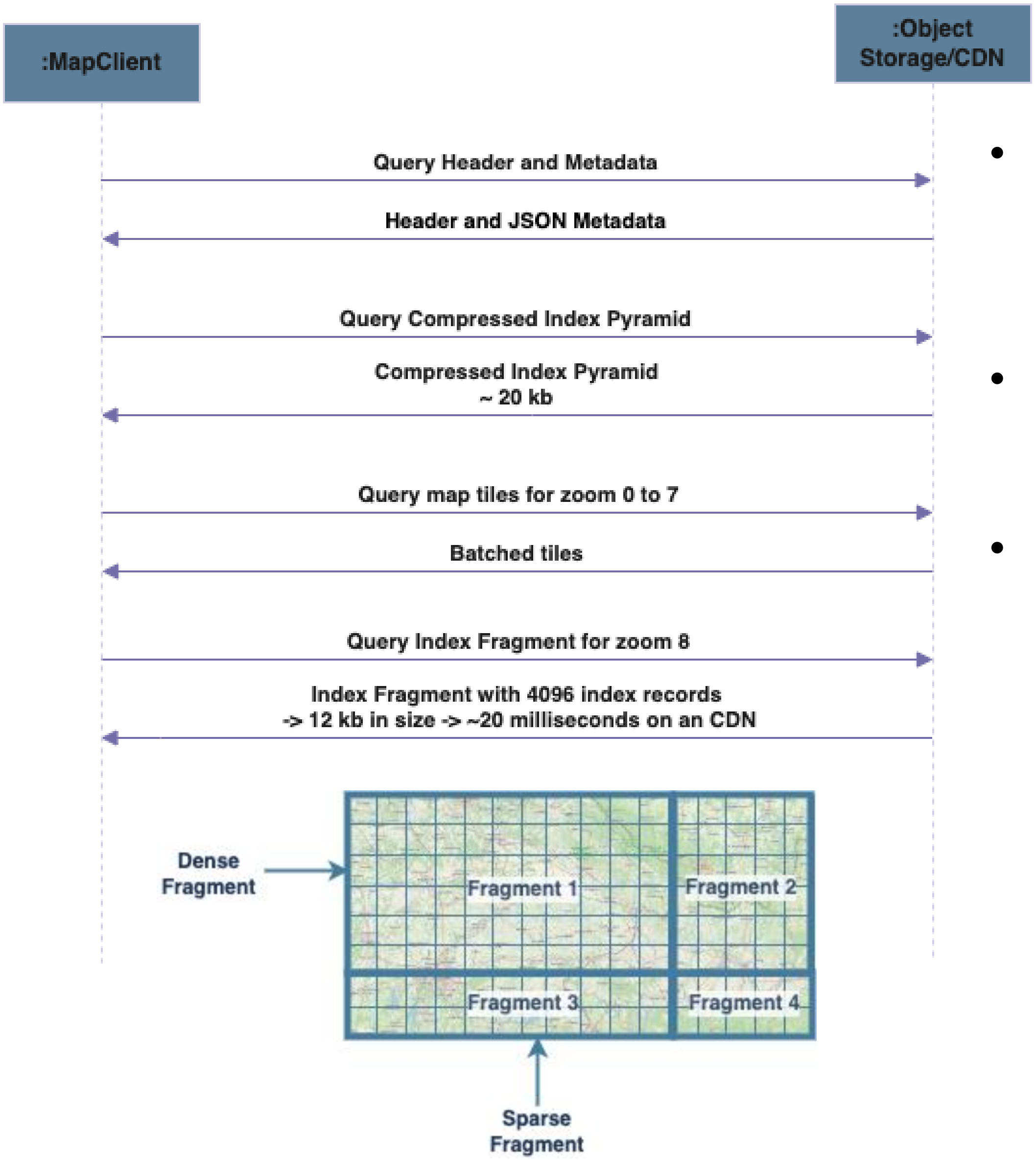
Query map tiles for zoom 0 to 7

Batched tiles

Query Index Fragment for zoom 8

Index Fragment with 4096 index records
-> 12 kb in size -> ~20 milliseconds on an CDN

COMTiles sequence



:MapClient

:Object Storage/CDN

Query Header and Metadata

Header and JSON Metadata

Query Compressed Index Pyramid

Compressed Index Pyramid
~ 20 kb

Query map tiles for zoom 0 to 7

Batched tiles

Query Index Fragment for zoom 8

Index Fragment with 4096 index records
-> 12 kb in size -> ~20 milliseconds on an CDN

Dense Fragment

Fragment 1

Fragment 2

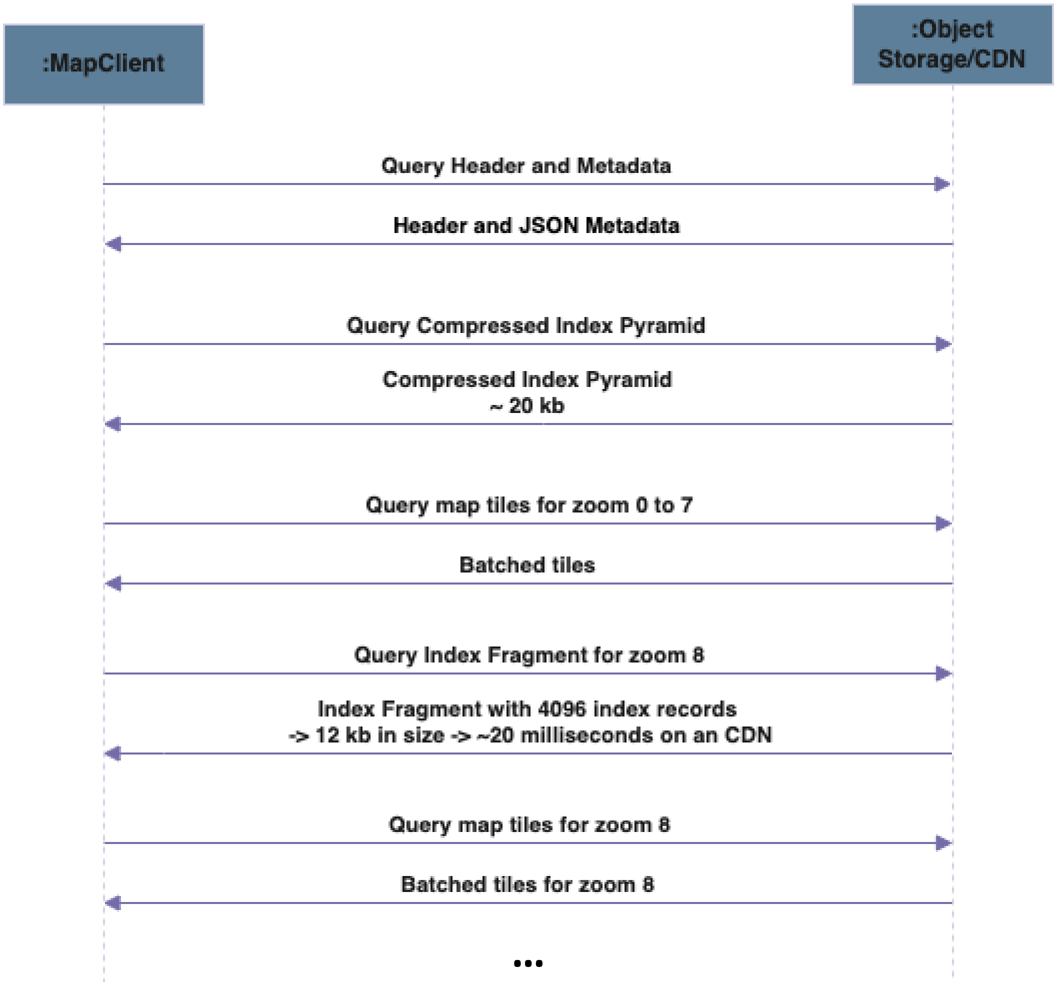
Fragment 3

Fragment 4

Sparse Fragment

- For higher zoom levels portions of the index are lazy loaded via **index fragments**
- The index fragments are ordered on a **space-filling curve** (Hilbert or Z-Order)
- Most of the time only one additional pre-fetch per zoom level is needed before accessing the actual map tiles for the current viewport of the map -> **this is bearly or not at all noticeable for the user regarding the user experience**

COMTiles sequence



:MapClient

:Object Storage/CDN

Query Header and Metadata

Header and JSON Metadata

Query Compressed Index Pyramid

Compressed Index Pyramid
~ 20 kb

Query map tiles for zoom 0 to 7

Batched tiles

Query Index Fragment for zoom 8

Index Fragment with 4096 index records
-> 12 kb in size -> ~20 milliseconds on an CDN

Query map tiles for zoom 8

Batched tiles for zoom 8

...

Challanges

- Additional requests for fetching portions of the index → a “servless“ Tile Server (e.g. hosted on AWS Lambda or CloudFlare Workers) can be used to eliminate the additional index request and to further minimize the latency
- Storing large tilesets on an object storage is cheap, but egress costs can get very expensive → with “requester pays“ the users have to pay for the egress costs
- Mulipart ranges are not supported by the cloud provdiers on object storage only on a CDN
- Most cloud provider doesn‘t support HTTP/2 on object storage only on a CDN
- Compression via Content-Encoding header is not supported in http range reqeusts → Compression Streams API also not (yet) supported on all browsers
- Automatic (dynamic) compression is supported for regular HTTP requests by the CDNs but not for range requests
- Readoptimized format → updates are expensive

Next Steps

- Cloud optimized 3D Tiles
  - The 3DTiles Next implicit tiling extension has now support for fixed spatial data structures (quadtree for 2D or octree for 3D) which allow random access
  - Combining the COMTiles approach with 3D Tiles Next for building a cloud optimized 3D geospatial format
- Cloud optimized vector tiles
  - Column-oriented approach like Parquet for the properties of the features to achive better compression
  - Faster decoding (e.g. zero copy) inspired by GLTF (3DTIles)
  - Make a specific layer of a tile queryable via additonal secondary index (sidecar) → can reduce the number of features up to 50% depending on the style

Thank You!

Email: markus.tremmel23@gmail.com

GitHub: @mactrem