This example demonstrates the AggressiveSplittingPlugin for splitting the bundle into multiple smaller chunks to improve caching. This works best with an HTTP2 web server, otherwise, there is an overhead for the increased number of requests.

AggressiveSplittingPlugin splits every chunk until it reaches the specified `maxSize`. In this example, it tries to create chunks with <50kB raw code, which typically minimizes to ~10kB. It groups modules by folder structure, because modules in the same folder are likely to have similar repetitive text, making them gzip efficiently together. They are also likely to change together.

AggressiveSplittingPlugin records its splitting in the webpack records. When it is next run, it tries to use the last recorded splitting. Since changes to application code between one run and the next are usually in only a few modules (or just one), re-using the old splittings (and chunks, which are probably still in the client's cache), is highly advantageous.

Only chunks that are bigger than the specified `minSize` are stored into the records. This ensures that these chunks fill up as your application grows, instead of creating many records of small chunks for every change.

If a module changes, its chunks are declared to be invalid and are put back into the module pool. New chunks are created from all modules in the pool.

There is a tradeoff here:

The caching improves with smaller `maxSize`, as chunks change less often and can be reused more often after an update.

The compression improves with bigger `maxSize`, as gzip works better for bigger files. It's more likely to find duplicate strings, etc.

The backward compatibility (non-HTTP2 client) improves with bigger `maxSize`, as the number of requests decreases.

```
_{{webpack.config.js}}_
```

Info

Unoptimized

```
_{{stdout}}_
```

Production mode

```
_{{production:stdout}}_
```

Records

```
_{{dist/records.json}}_
```