

From warp to axum

Lighthouse transition journey

Léa Narzis

Fellow, Ethereum Protocol Fellowship



Introduction

Introduction



warp

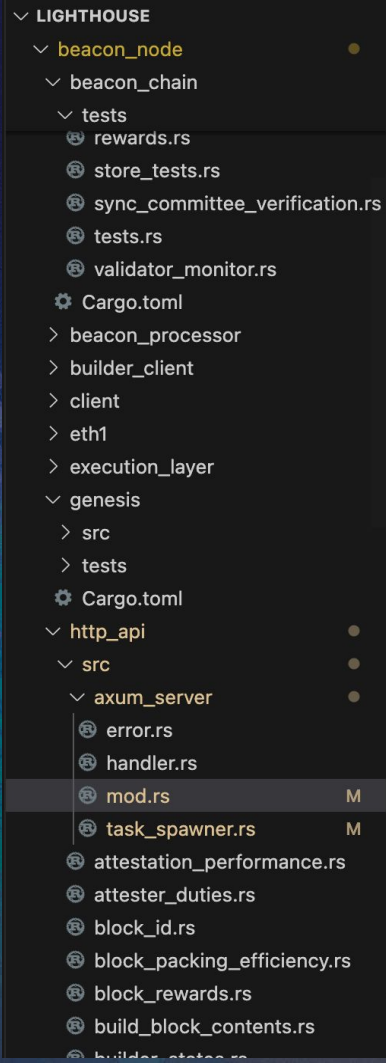
VS



axum

The reasons for the transition

- Ease of use :
“warp's type system forces to handle types in a difficult manner, sometimes leading to allocate components on the heap”
- Moving to modular approach : make the api's endpoint definitions more modular by defining trait functions, and have a BeaconApi trait
- Unify stack



Scopes limitation

- Limited to beacon node API endpoints

Refactor challenges

Developer challenges : moving fast ?

- Only removing the warp crate and replaced it by the axum ?

Testing performances : conditions

- Warmed up with initial requests
- Sent 100 requests per type of endpoint
- Comparison against warp for total execution time and average response time per request

Testing performances' results

Performance Comparison (100 requests per implementation):

Framework & Configuration	Total Time	Avg Time/Request	vs. Warp
Warp	68.417μs	684ns	baseline
Axum (no task spawner)	103.292μs	1.032μs	1.51x slower
Axum (with task spawner)	823.75μs	8.237μs	12.04x slower

Next steps

Transition timeline

Optimize task spawner

2 day – 1 week

Refactor axum task spawner to be initiated with the server initialization

Check tests

Prepare global plan to transition

1 week

Open issue tracking and split into smallest issues with lighthouse's team

Deliver the tasks :D

1 month

Deliver task by task following the issues list

... to the final goal

List endpoints in ethereum-apis




1 week

Add all beacon node
endpoints in BeaconApi
Trait

Personal insights

Personal insights

- Learned to break down huge refactors into manageable pieces
- Familiarity with Beacon Node API: got used to the beacon node API part of the Lighthouse repository
- Better developer instincts: smarter reflexes for approaching complex tasks as a developer



**Thanks for
the guidance and help,
Let's continue to transition.**