

Vinbero: The Modular Server

v0.3.1





목차

1. 간략한 소개
2. 만들게 된 계기
3. 아키텍처
4. 테스트 & 배포 & 개발환경
5. 프로젝트 홍보
6. 변경사항
7. 앞으로의 계획

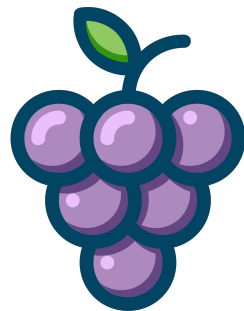
간략한 소개





vinbero

- 모듈형 서버
- 코어와 모듈들로 분리
- 코어는 모듈들을 로딩하고 초기화, 정리를 함
- 모듈은 여러개의 하위 모듈들을 가질 수 있음
- 모듈은 하위 모듈의 함수들을 호출하여 통신함
- 로딩되는 모듈에 따라 완전히 다른 동작 가능

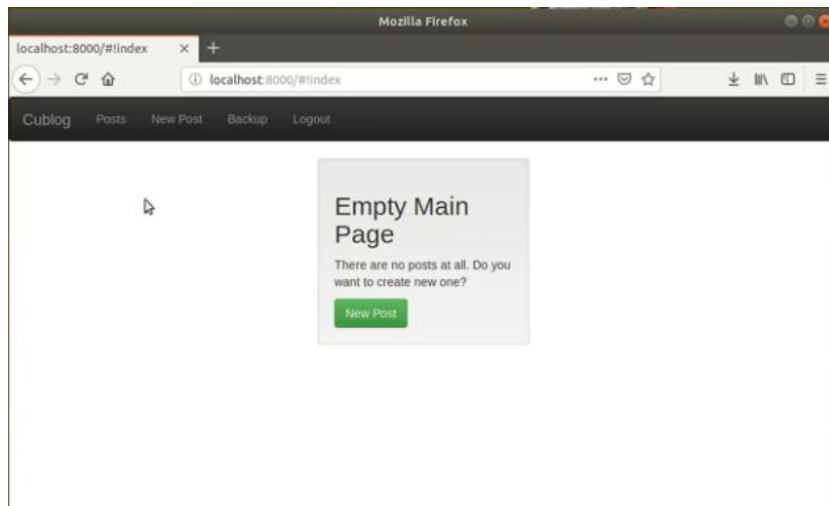


vinbero는
에스페란토어로
포도를 의미합니다



간략한 소개

vinbero 블로그 예제



<https://www.youtube.com/watch?v=43bqzvlO3mk>

블로그 부분은 lua,
javascript로 작성

만들게 된 계기





만들게 된 계기

서버를 만들 때 마다 반복되는 패턴...

- server socket 생성
- bind(), listen(), accept()
- 멀티 쓰레드 vs 멀티 프로세스
- IO multiplexing
- 응용 계층 프로토콜 구현



어떻게 하면 이러한
반복을 줄일 수
있을까?



만들게 된 계기

서버를 기능별로
잘게 쪼개서 모듈로 만들자!



만들게 된 계기

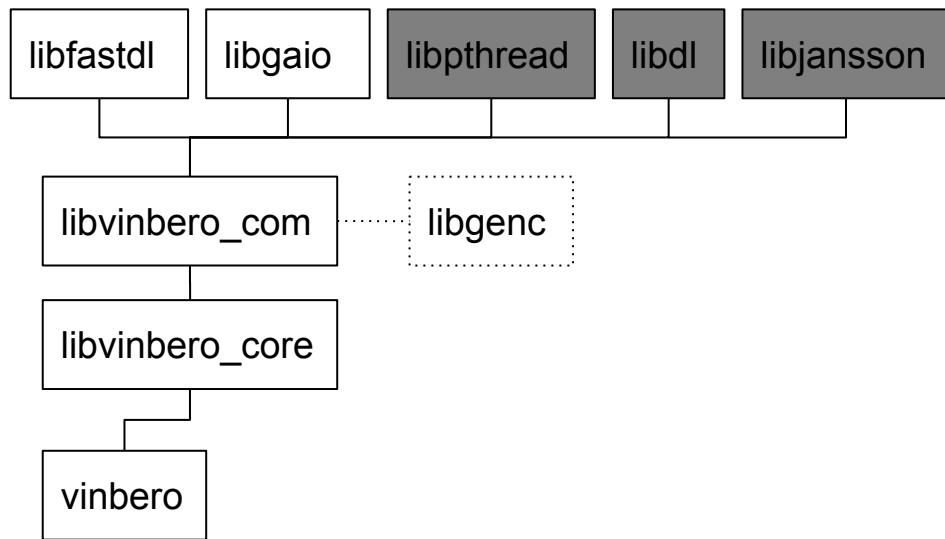
서버의 코어에서는
모듈 관리만 하자!

아키텍처





vinbero 실행파일



header only library



3rd party library



라이브러리 설명

→ **libgenc**

c 자료구조 라이브러리

→ **libgaio**

io 추상화 라이브러리

→ **libfastdl**

해쉬 테이블을 이용해 libdl 성능 개선

→ **libvinbero_com**

모듈 공통 루틴 라이브러리

→ **libvinbero_core**

vinbero 코어

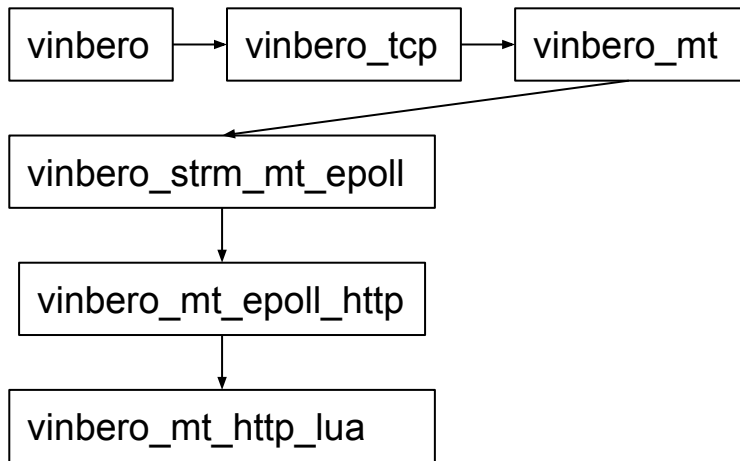


모듈 목록

- `vinbero_tcp`
- `vinbero_mt`
- `vinbero_strm_mt_epoll`
- `vinbero_mt_epoll_tls`
- `vinbero_mt_epoll_http`
- `vinbero_iplogger`
- `vinbero_mt_http_lua`



모듈 로딩 모습



```
{
  "core": {
    "config": {"vinbero.setUid": 1001},
    "next": ["vinbero_tcp"]
  },
  "vinbero_tcp": {
    "paths": ["/usr/local/lib/vinbero/vinbero_tcp.so", "/usr/lib/vinbero/vinbero_tcp.so"],
    "config": {"vinbero_tcp.port": 80, "vinbero_tcp.reuseAddress": true},
    "next": ["vinbero_mt"]
  },
  "vinbero_mt": {
    "paths": ["/usr/local/lib/vinbero/vinbero_mt.so", "/usr/lib/vinbero/vinbero_mt.so"],
    "config": {"vinbero_mt.workerCount": 4},
    "next": ["vinbero_strm_mt_epoll"]
  },
  "vinbero_strm_mt_epoll": {
    "paths": ["/usr/local/lib/vinbero/vinbero_strm_mt_epoll.so",
"/usr/lib/vinbero/vinbero_strm_mt_epoll.so"],
    "config": {"vinbero_strm_mt_epoll.clientTimeoutSeconds": 3},
    "next": ["vinbero_mt_epoll_http"]
  },
  "vinbero_mt_epoll_http": {
    "paths": ["/usr/local/lib/vinbero/vinbero_mt_epoll_http.so",
"/usr/lib/vinbero/vinbero_mt_epoll_http.so"],
    "config": {},
    "next": ["vinbero_mt_http_lua"]
  },
  "vinbero_mt_http_lua": {
    "paths": ["/usr/local/lib/vinbero/vinbero_mt_http_lua.so",
"/usr/lib/vinbero/vinbero_mt_http_lua.so"],
    "config": {
      "vinbero_mt_http_lua.scriptFile": "/srv/app.lua",
      "vinbero_mt_http_lua.scriptArg": {}
    },
    "next": []
  }
}
```



그런데
모듈간의 호환성은
어떻게 검사할까?



영감: Go 언어의 interface

```
package main

import "fmt"
import "math"

type geometry interface {
    area() float64
    perim() float64
}

type rect struct {
    width, height float64
}

type circle struct {
    radius float64
}

func (r rect) area() float64 {
    return r.width * r.height
}

func (r rect) perim() float64 {
    return 2*r.width + 2*r.height
}

func (c circle) area() float64 {
    return math.Pi * c.radius * c.radius
}

func (c circle) perim() float64 {
    return 2 * math.Pi * c.radius
}

func measure(g geometry) {
    fmt.Println(g)
    fmt.Println(g.area())
    fmt.Println(g.perim())
}

func main() {
    r := rect{width: 3, height: 4}
    c := circle{radius: 5}

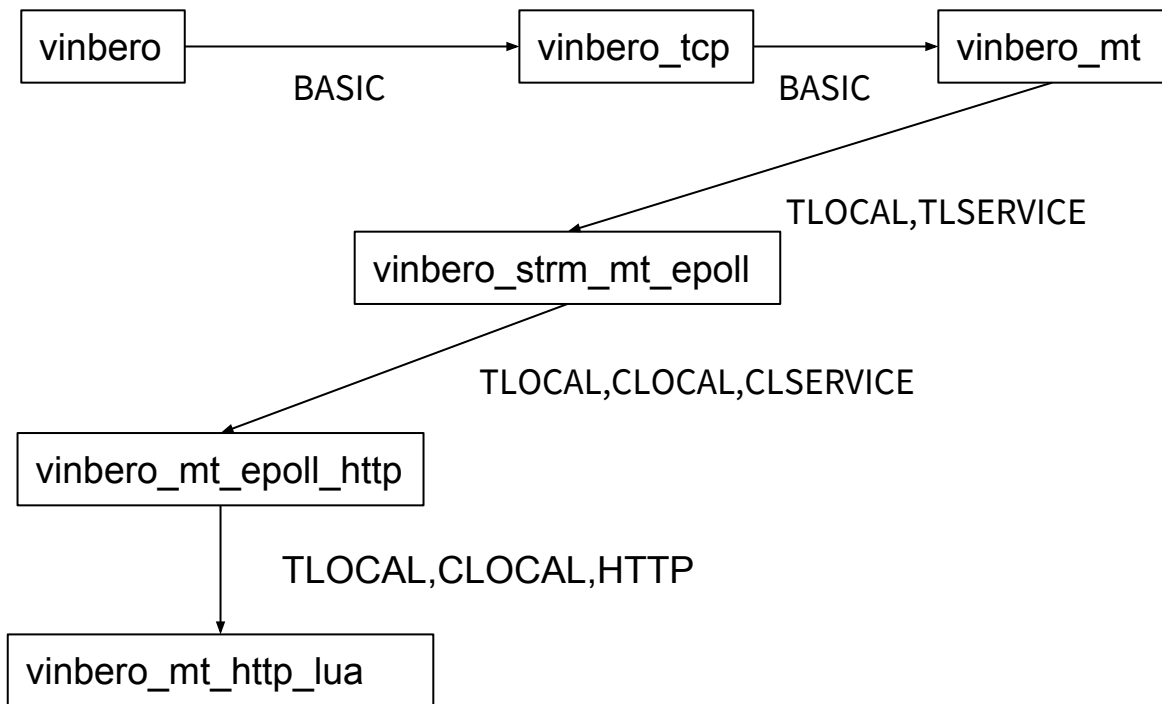
    measure(r)
    measure(c)
}
```




함수들의 집합인 interface를 정의하자!



interface와 함께 보는 모듈 로딩



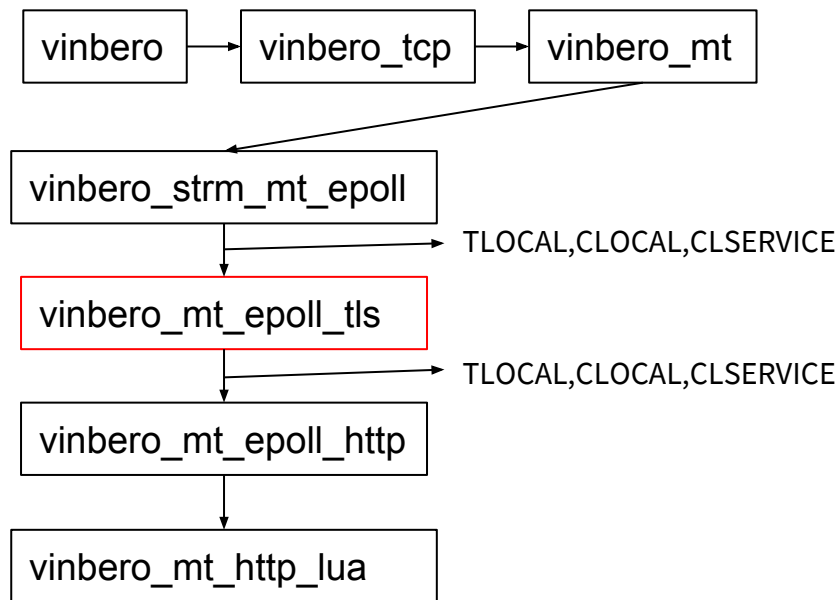
interface 그룹이
일치하는 모듈끼리
연결 가능



모듈 로딩 응용



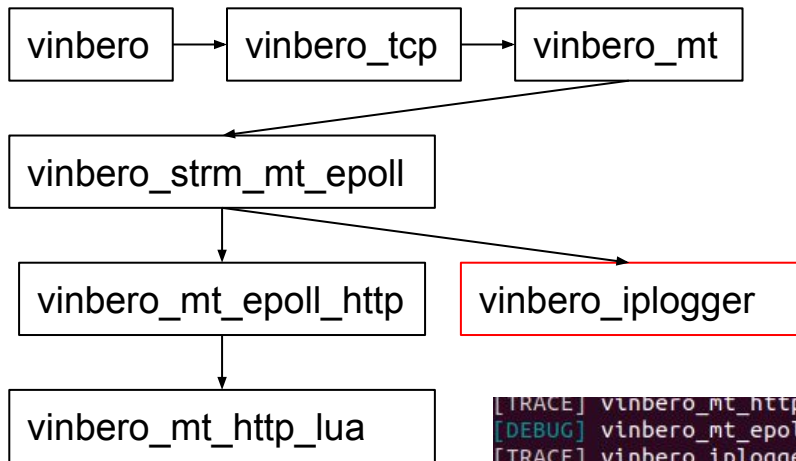
모듈 로딩 모습 (https 서버)



interface 그룹이
일치하므로 모듈
끼워넣기 가능



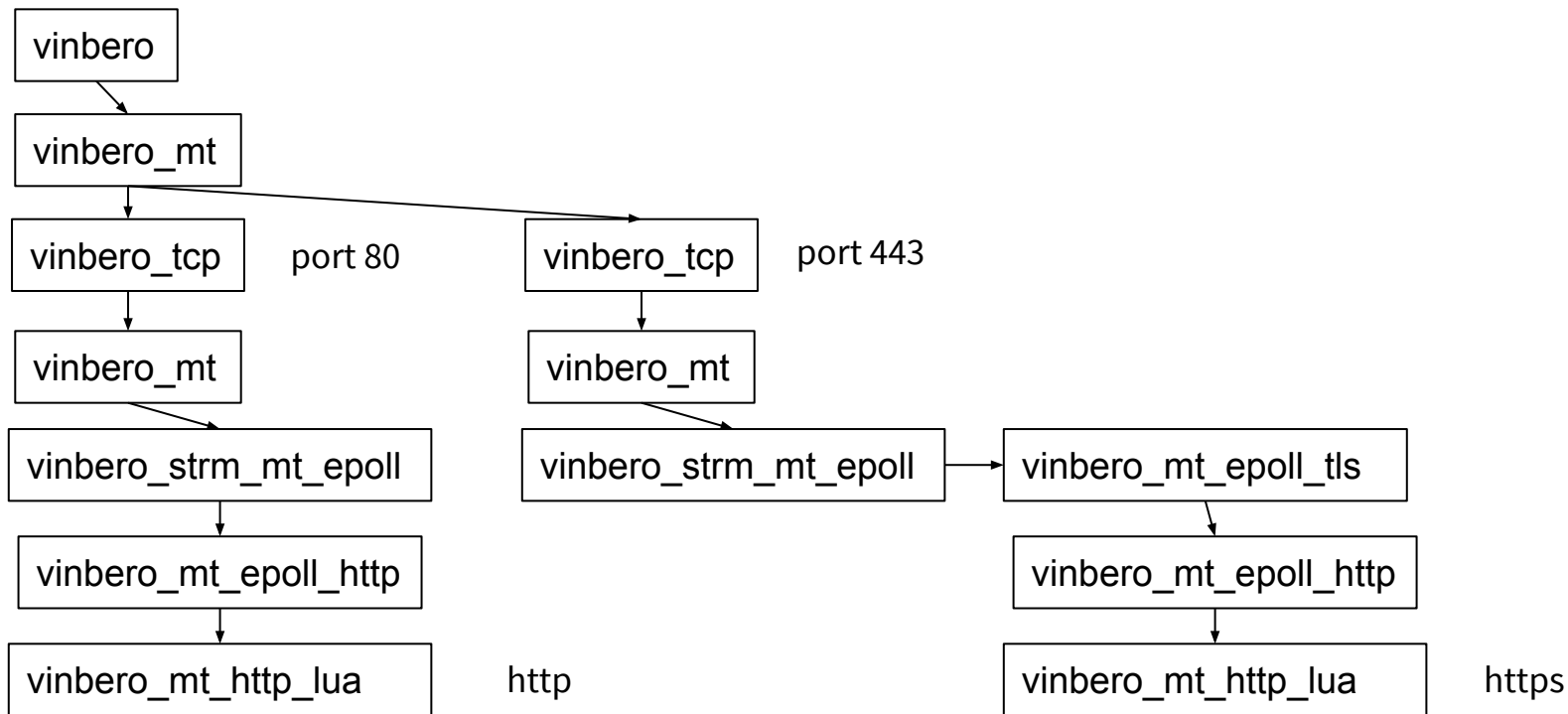
한개 이상의 하위 모듈 로딩도 가능



```
[TRACE] vinbero_mt_http_lua.c: 870: in vinbero_trace HTTP_onRequestFinish()
[DEBUG] vinbero_mt_epoll_http.c: 491: HTTP connection Keep-Alive
[TRACE] vinbero_iplogger.c: 84: in vinbero_iface_CLSERVICE_call()
[INFO ] vinbero_iplogger.c: 92: SOCKET 12 IP: 127.0.0.1
[TRACE] vinbero_strm_mt_epoll.c: 168: in vinbero_strm_mt_epoll_destroyChildClModules()
[TRACE] vinbero_mt_epoll_http.c: 531: in vinbero_iface_CLOCAL_destroy()
```



다수의 서버 소켓 (구현 예정)





테스트 & 배포 & 개발환경





테스트 & 배포 & 개발환경

Travis-ci를 이용한 테스트 자동화

The screenshot displays the Travis CI web interface for the repository `vinbero / vinbero`. The top navigation bar includes links for `About Us`, `Blog`, `Status`, `Documentation`, and `Help`, along with a `Sign in with GitHub` button. A green banner at the top right states: "Help make Open Source a better place and start building better software today!".

The main content area shows the repository name `vinbero / vinbero` with a `Build passing` status. Below this, there are tabs for `Current`, `Branches`, `Build History`, and `Pull Requests`. The `Current` tab is active, displaying a build summary for the `master` branch with the commit `b6fca4c`. The build status is `#249 passed`, and it ran for 45 seconds, completed about 3 hours ago. The build was triggered by `Byeonggon Lee` and used the `gcc C` compiler.

Below the build summary, there are links for `Job log` and `View config`. The `Job log` section is expanded, showing a terminal output of the build process. The log includes the following steps and commands:

- 1 Worker information
- 0 Build system information
- 156 \$ git clone --depth=50 --branch=master https://github.com/vinbero/vinbero.git vinbero/vinbero
- 157 \$ git submodule update --init --recursive
- 173 \$ export TRAVIS_COMPILER=gcc
- 174 \$ export CC=gcc
- 175 \$ export CC_FOR_BUILD=gcc
- 176 \$ gcc --version
- 177 gcc (Ubuntu 5.4.0-6ubuntu1-16.04.11) 5.4.0 20160609
- 178 Copyright (c) 2015 Free Software Foundation, Inc.
- 179 This is free software; see the source for copying conditions. There is NO
- 180 warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

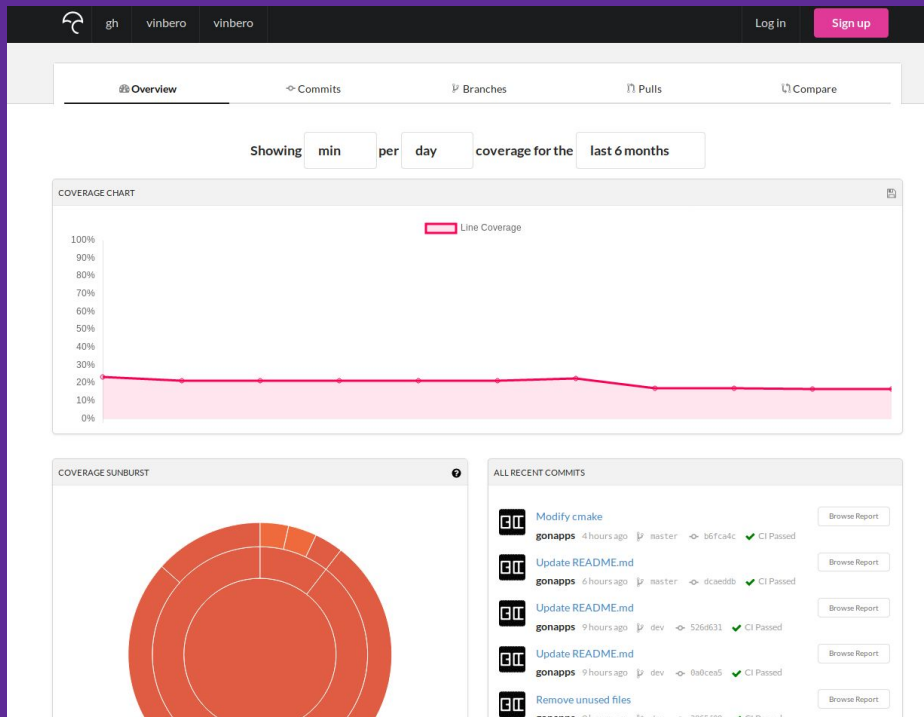
The log output is displayed in a dark-themed terminal window with a `Raw log` button in the top right corner.



테스트 & 배포 & 개발환경

codecov.io를

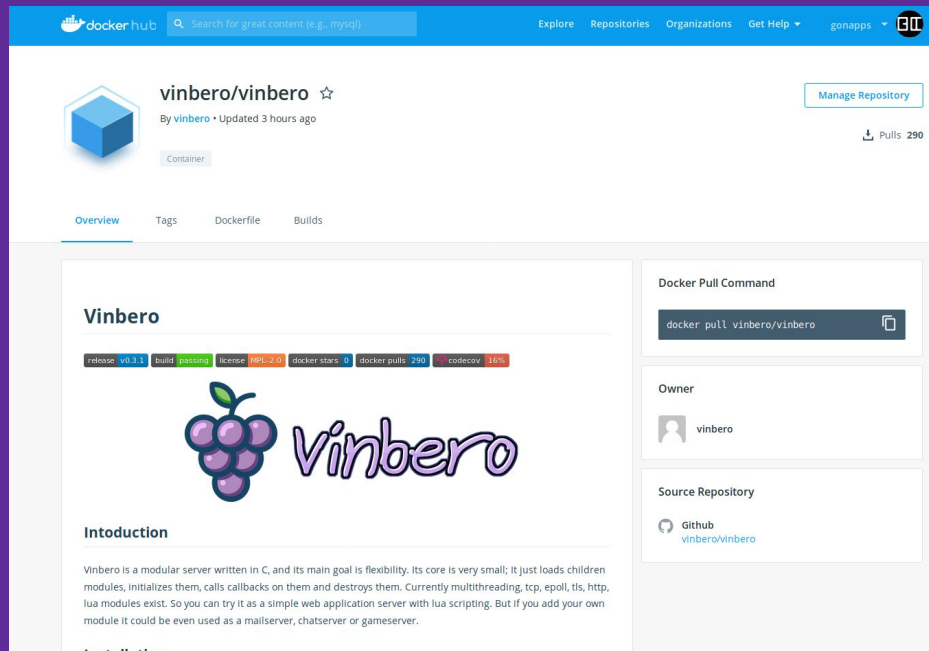
이용한 코드
커버리지





테스트 & 배포 & 개발환경

Docker를 이용한 배포 자동화



The screenshot shows the Docker Hub page for the `vinbero/vinbero` repository. The page header includes the Docker Hub logo, a search bar, and navigation links: Explore, Repositories, Organizations, Get Help, and gonapps. The repository name `vinbero/vinbero` is displayed with a star icon and the text "By vinbero • Updated 3 hours ago". A "Manage Repository" button is in the top right. Below the repository name, there's a "Container" label and a "Pulls 290" indicator. The main content area has tabs for Overview, Tags, Dockerfile, and Builds. The Overview tab is active, showing the repository's metadata: release v0.3.1, build pass, license MIT, 94% 2.0, docker stars 0, docker pulls 290, and codecov 10%. The Vinbero logo, featuring a bunch of purple grapes and the word "Vinbero", is prominently displayed. Below the logo, the "Introduction" section describes Vinbero as a modular server written in C, designed for flexibility and ease of use. The right sidebar contains the "Docker Pull Command" section with the command `docker pull vinbero/vinbero`, the "Owner" section showing the user "vinbero", and the "Source Repository" section linking to the GitHub repository `vinbero/vinbero`.



개발도 Docker 안에서 합니다

vinbero/vinbero_mt_http_lua:dev

```
File Edit View Search Terminal Help
root@e65d6a4deda7: /usr/src
# ls
libfastdl      libvinbero_con  vinbero_mt      vinbero_mt_http_lua
libgato        vinbero         vinbero_mt_epoll_http  vinbero_strm_mt_epoll
libgenc        vinbero-ifaces  vinbero_mt_epoll_tls   vinbero_tcp
root@e65d6a4deda7: /usr/src
```

[9:32:34]

[9:32:35]

[0] 0:zsh* 1:ash- "e65d6a4deda7" 09:32 19-Feb-19

변경사항





→ jansson 의존성 감소

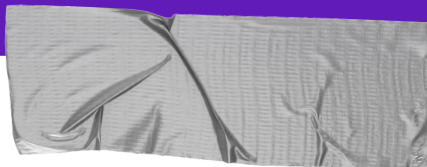
전에는 config.json 파일을 읽으면
libjansson의 json_t* 타입으로 저장을 했지만
추후 json 이외의 설정파일을 지원하기 위해
vinbero_com_Object 타입 사용

→ AUTHORS 추가

빌드 할 때마다 master branch의 커밋 횟수와
이름이 쓰여진 AUTHORS 파일 자동 생성

→ Request body 버그 수정

http request body의 데이터를 잘못 읽는
버그 수정



→ Blog 예제 도커화

vinbero_mt_http_lua에서 실행 가능한

lua로 작성 된 블로그 예제를 도커
컨테이너로 배포

→ codecov.io 적용

코드 커버리지를 나타내주는 **codecov.io**를
적용

→ config 파일에 path 대신 paths 사용

/usr/local/lib/vinbero, /usr/lib/vinbero 등 여러
위치에서 모듈을 로딩할 수 있게끔

config 파일에 path 항목 대신 paths 항목을
사용



- ➔ **.travis.yml 수정**
ubuntu trusty 대신 ubuntu xenial 사용
- ➔ **MPLv2 대신 MPL-2.0으로 표기**
SPDX 표기에 맞게 MPL-2.0으로 라이선스 표기 변경
- ➔ **함수, 매크로 이름 변경**
이름을 간략하게 변경
- ➔ **libgenc 해쉬 테이블에 SipHash 적용**
해쉬 테이블에 SipHash를 사용해 충돌을 줄임
- ➔ **libfastdl에 libgenc의 Hash table 적용**
uthash 라이브러리 의존성 제거



모듈에 메타데이터 추가

`elf` 파일에 섹션을 추가하려 했으나

읽기가 번거로워

간단하게 리터럴을 반환하는 함수로 정의



모듈 호환 검사 강화

코어가 모듈의 메타데이터를 읽어서

모듈간의 호환여부 검사



`cmake` 수정

반복되는 `cmake` 코드를 별도의 `cmake`

파일로 만들어서 `submodule`로 분리

프로젝트 홍보

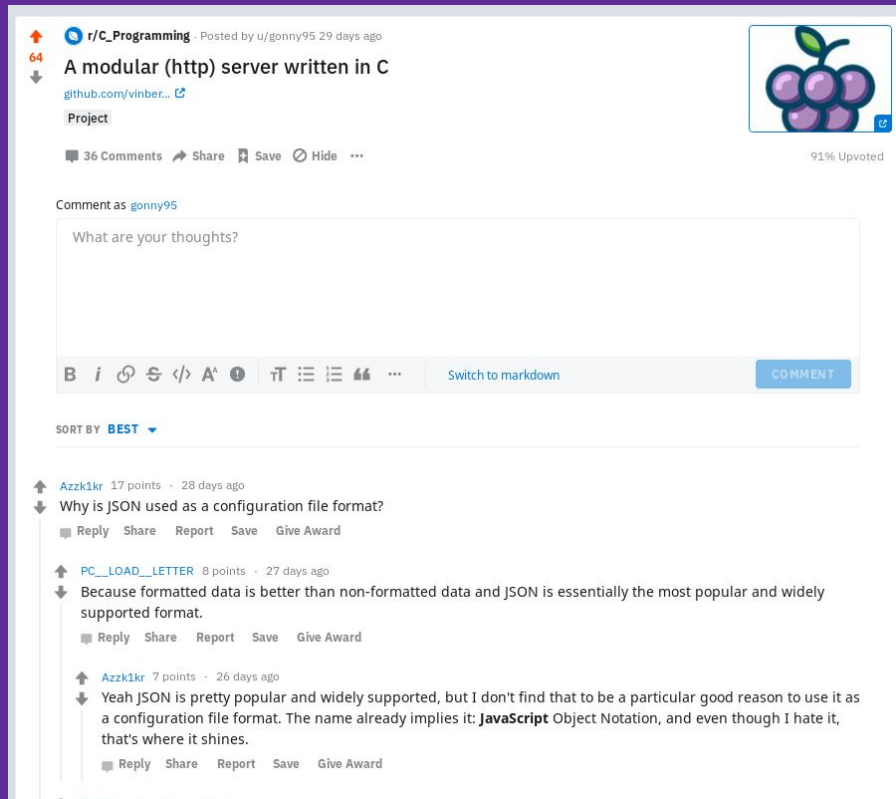




컨트리뷰터를
모으기
위해서는
홍보가
필요하다



reddit 등의 해외 사이트에 홍보

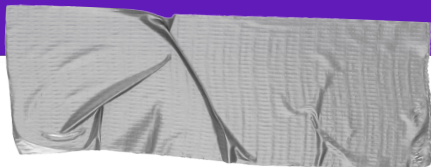




컨트리뷰터가
모일 때 까지
지속적으로
홍보할
예정이다

앞으로의 계획





- 모듈 메타데이터 완벽 지원
- `pathInfo` 처리 기능을
lua에서 `http` 모듈로 옮기기
- lua 모듈에서 `client`
`socket`에서 `ip` 등의
데이터를 읽어올 수 있게
만든다
- 인터페이스 하나씩을 모두
레포로 만든다
- 인터페이스의 버전 관리

감사합니다

