About Beast
Beast is the HTTP and WebSockets C++ library used by Ripple on production servers. For more information on Beast, visit:
https://github.com/vinniefalco/Beast

About Ripple
Ripple develops and operates rippled, a decentralized peer to peer digital currency and blockchain ledger system. For more information on rippled and Ripple, please visit:
https://github.com/ripple/rippled
http://ripple.com

NuDB
Looking for more libraries? Check out NuDB, a fast key/value insert-only database for SSD drives in C++11. It is header-only, open source, with documentation, tests, benchmarks, and 96% code coverage. For more information please visit:
https://github.com/vinniefalco/NuDB

About the Author
Vinnie Falco is a programmer at Ripple with 34 years of experience and author of the Gnutella-compatible file sharing software BearShare. Visit his other open source repositories at
https://github.com/vinniefalco

ripple

Introducing Beast: HTTP and WebSockets C++ Library

* HTTP/WebSockets using Boost.Asio
* Header-only, C++11/14, open source
* Symmetric: Build clients and servers
* Synchronous and Asynchronous APIs
* With docs, tests, and benchmarks
* Running now on production servers!

Home:
https://github.com/vinniefalco/Beast

By Vinnie Falco

ripple

About Beast
Beast is the HTTP and WebSockets C++ library used by Ripple on production servers. For more information on Beast, visit:
https://github.com/vinniefalco/Beast

About Ripple
Ripple develops and operates rippled, a decentralized peer to peer digital currency and blockchain ledger system. For more information on rippled and Ripple, please visit:
https://github.com/ripple/rippled
http://ripple.com

NuDB
Looking for more libraries? Check out NuDB, a fast key/value insert-only database for SSD drives in C++11. It is header-only, open source, with documentation, tests, benchmarks, and 96% code coverage. For more information please visit:
https://github.com/vinniefalco/NuDB

About the Author
Vinnie Falco is a programmer at Ripple with 34 years of experience and author of the Gnutella-compatible file sharing software BearShare. Visit his other open source repositories at
https://github.com/vinniefalco

Copyright
NuDB, Beast are Copyright 2015-2016 by Vinnie Falco
Distributed under the Boost Software License, Version 1.0.
See license at http://www.boost.org/LICENSE_1_0.txt.

ripple

Introducing Beast: HTTP and WebSockets C++ Library

* HTTP/WebSockets using Boost.Asio
* Header-only, C++11/14, open source
* Symmetric: Build clients and servers
* Synchronous and Asynchronous APIs
* With docs, tests, and benchmarks
* Running now on production servers!

Home:
https://github.com/vinniefalco/Beast

By Vinnie Falco

ripple

```
// WebSocket Example

#include <beast/core/to_string.hpp>
#include <beast/websocket.hpp>
#include <boost/asio.hpp>
#include <iostream>
#include <string>

int main()
{
    // Connect to remote host
    //
    auto const host = "echo.websocket.org";
    boost::asio::io_service ios;
    boost::asio::ip::tcp::resolver r{ios};
    boost::asio::ip::tcp::socket sock{ios};
    boost::asio::connect(sock, r.resolve(
        boost::asio::ip::tcp::resolver::query{
            host, "80"}));

    // Handshake and send message
    //
    beast::websocket::stream<
        boost::asio::ip::tcp::socket&> ws{sock};
    ws.handshake(host, "/");
    ws.write(
        boost::asio::buffer("Hello, world!"));

    // Receive message, print, and close
    //
    beast::streambuf sb;
    beast::websocket::opcode op;
    ws.read(op, sb);

    ws.close(
        beast::websocket::close_code::normal);
    std::cout << to_string(sb.data()) << "\n";
}
```

```
// HTTP Example

#include <beast/http.hpp>
#include <boost/asio.hpp>
#include <iostream>
#include <string>

int main()
{
    // Connect to remote host
    //
    auto const host = "boost.org";
    boost::asio::io_service ios;
    boost::asio::ip::tcp::resolver r{ios};
    boost::asio::ip::tcp::socket sock{ios};
    boost::asio::connect(sock,
        r.resolve(boost::asio::ip::tcp::
            resolver::query{host, "http"}));

    // Create and Send HTTP request
    //
    beast::http::request_v1<
        beast::http::empty_body> req;
    req.method = "GET";
    req.url = "/";
    req.version = 11;
    req.headers.insert("Host", host + ":" +
        std::to_string(
            sock.remote_endpoint().port()));
    req.headers.insert("User-Agent", "Beast");
    beast::http::prepare(req);
    beast::http::write(sock, req);

    // Receive, print HTTP response
    //
    beast::streambuf sb;
    beast::http::response_v1<
        beast::http::streambuf_body> resp;
    beast::http::read(sock, sb, resp);
    std::cout << resp;
}
```


```
// WebSocket Example

#include <beast/core/to_string.hpp>
#include <beast/websocket.hpp>
#include <boost/asio.hpp>
#include <iostream>
#include <string>

int main()
{
    // Connect to remote host
    //
    auto const host = "echo.websocket.org";
    boost::asio::io_service ios;
    boost::asio::ip::tcp::resolver r{ios};
    boost::asio::ip::tcp::socket sock{ios};
    boost::asio::connect(sock, r.resolve(
        boost::asio::ip::tcp::resolver::query{
            host, "80"}));

    // Handshake and send message
    //
    beast::websocket::stream<
        boost::asio::ip::tcp::socket&> ws{sock};
    ws.handshake(host, "/");
    ws.write(
        boost::asio::buffer("Hello, world!"));

    // Receive message, print, and close
    //
    beast::streambuf sb;
    beast::websocket::opcode op;
    ws.read(op, sb);

    ws.close(
        beast::websocket::close_code::normal);
    std::cout << to_string(sb.data()) << "\n";
}
```

```
// HTTP Example

#include <beast/http.hpp>
#include <boost/asio.hpp>
#include <iostream>
#include <string>

int main()
{
    // Connect to remote host
    //
    auto const host = "boost.org";
    boost::asio::io_service ios;
    boost::asio::ip::tcp::resolver r{ios};
    boost::asio::ip::tcp::socket sock{ios};
    boost::asio::connect(sock,
        r.resolve(boost::asio::ip::tcp::
            resolver::query{host, "http"}));

    // Create and Send HTTP request
    //
    beast::http::request_v1<
        beast::http::empty_body> req;
    req.method = "GET";
    req.url = "/";
    req.version = 11;
    req.headers.insert("Host", host + ":" +
        std::to_string(
            sock.remote_endpoint().port()));
    req.headers.insert("User-Agent", "Beast");
    beast::http::prepare(req);
    beast::http::write(sock, req);

    // Receive, print HTTP response
    //
    beast::streambuf sb;
    beast::http::response_v1<
        beast::http::streambuf_body> resp;
    beast::http::read(sock, sb, resp);
    std::cout << resp;
}
```