# Introduction:

## Purpose:

This project comes from an idea that is to build a MQTT Broker Webserver for any MQTT applications. It can be an open source and easily installed in any machines or clouds. In addition, the Webserver must be stable, reliable and realtime due to there are thousands of connections and actions need to be done handled.

That means the MQTT Broker need to process these tasks simulanous and also have good errors handle.

## Idea:

The idea leads to selection of AKKA framework and Scala programming language. The advantages of these couple base on the structure Actor model of AKKA and

The knowlegde in this

## Model:

# MQTT Model & MQTT Broker:

(TBD)

# AKKA Toolkit:

## Overview:

AKKA is an open-source set of libraries and runtime for building highly concurrent, distributed, and fault tolerant applications on the JVM. Akka is written in Scala, with language bindings provided for both Scala and Java.

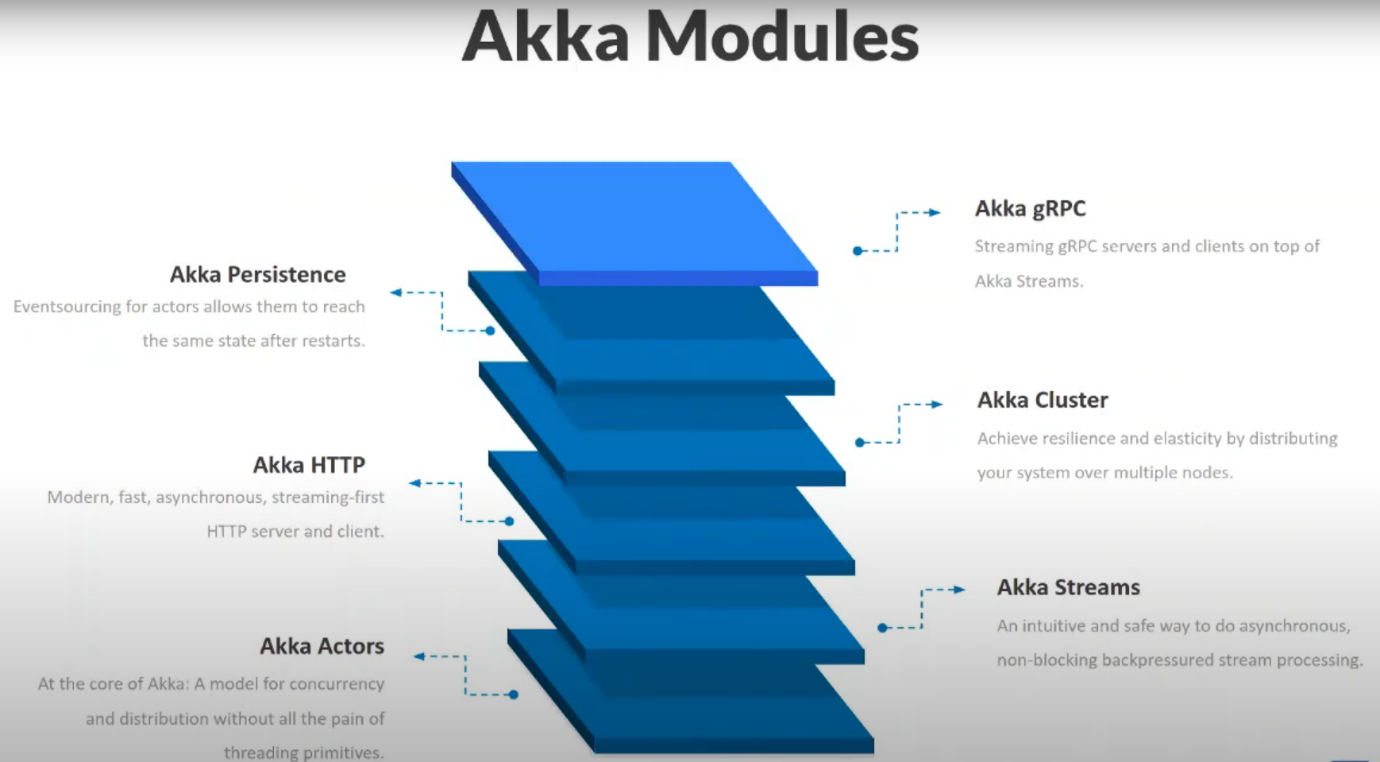
Akka’s approach to handling concurrency is based on the Actor Model.

## Specifications:

## Advantages:

# AKKA Libraries:

## Introduction:



## AKKA Actors:

A model for concurrency and distribution without all the pain of threading primitives.

## AKKA Streams:

An intuitive and safe way to do asynchronous, non-blocking backpressured stream processing.

## AKKA HTTP:

Modern, fast, asynchronous, streaming-first HTTP server and client.

The AKKA HTTP modules implement a full server and client side HTTP stack in top of AKKA actor and akka stream.

Core server API:

focus on the essential functionality of HTTP/1.1 server.

* + Connection management.
  + Parsing and rendering of msg and header.
  + Timeout management (requests and connections)
  + Response ordering.

High-level Server-Side API/Routing DSL: none core features of typical HTTP server

* + Route
  + Directive
  + Rejections
  + Exception Handling
* Testing

It's not a frame work but rather a more general toolkit for providing and consuming HTTP-based services.

You get to pick the APIs level of abstraction that is most suitable for your app.

Fully asynchronous and non-blocking.