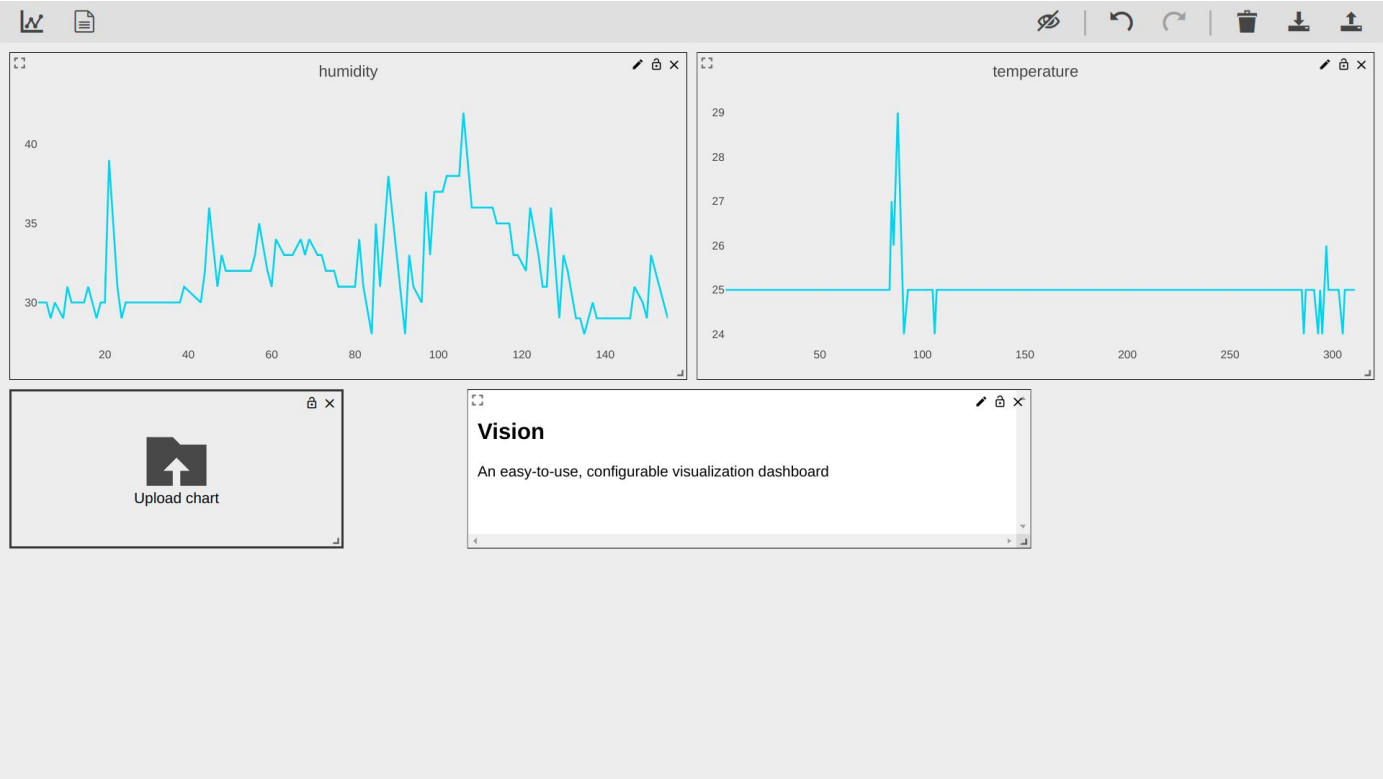
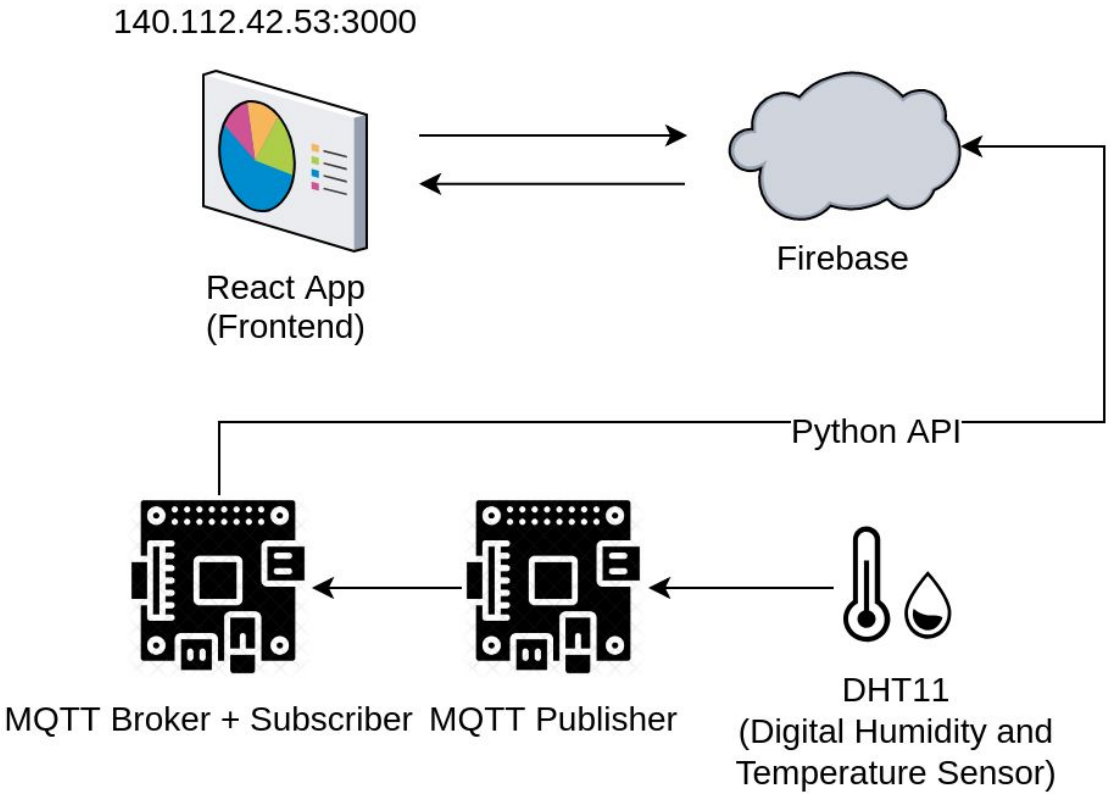
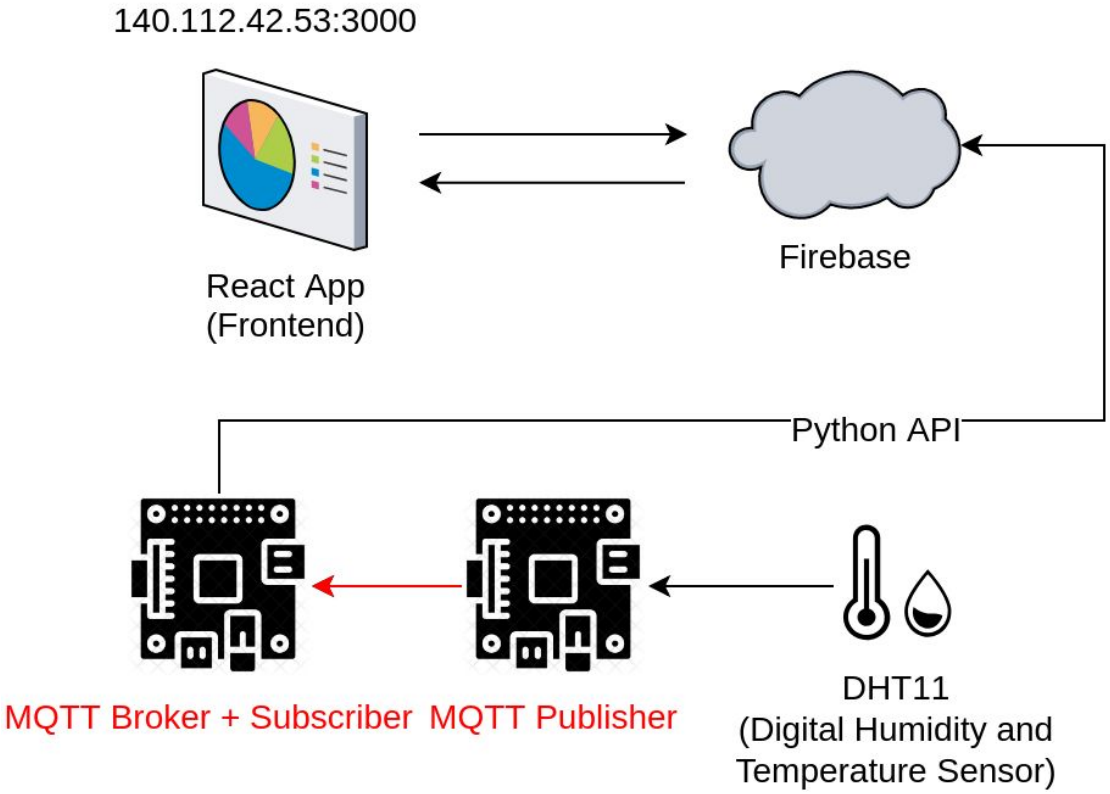
Vision



An easy-to-use, configurable visualization dashboard

Data Visualization with Firebase



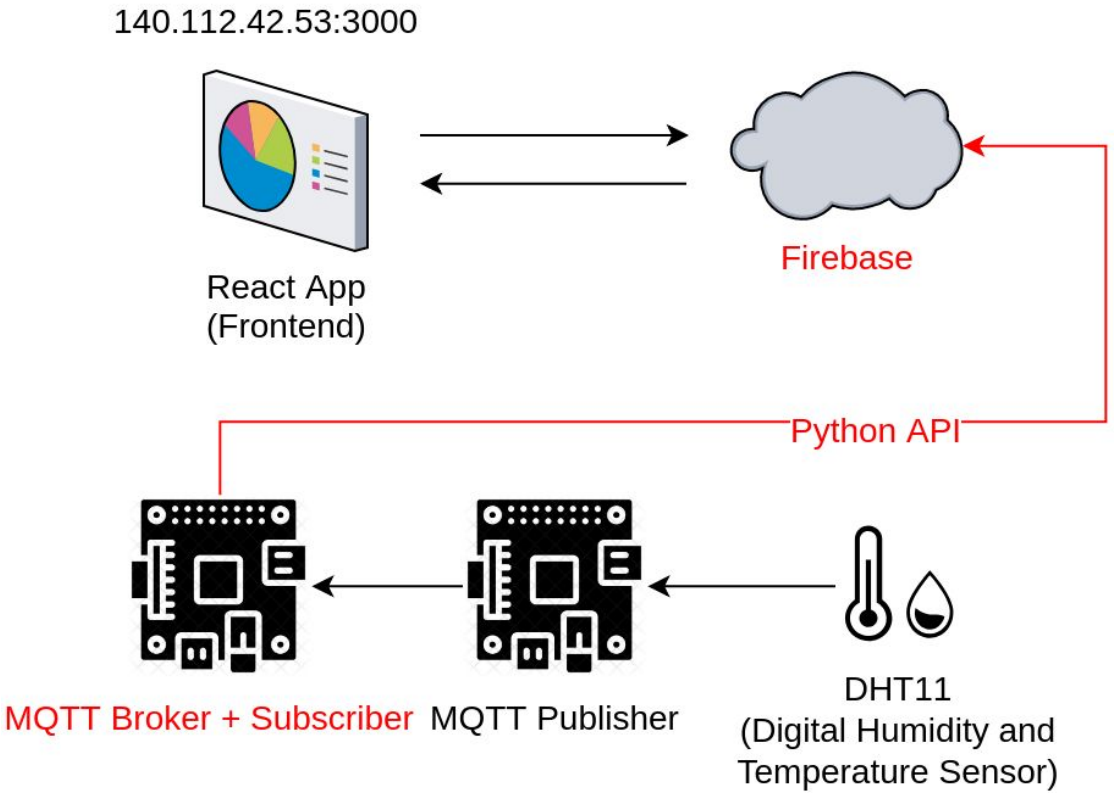

140.112.42.53:3000
React App
(Frontend)
Firebase
Python API
MQTT Broker + Subscriber
MQTT Publisher
DHT11
(Digital Humidity and
Temperature Sensor)

Publisher

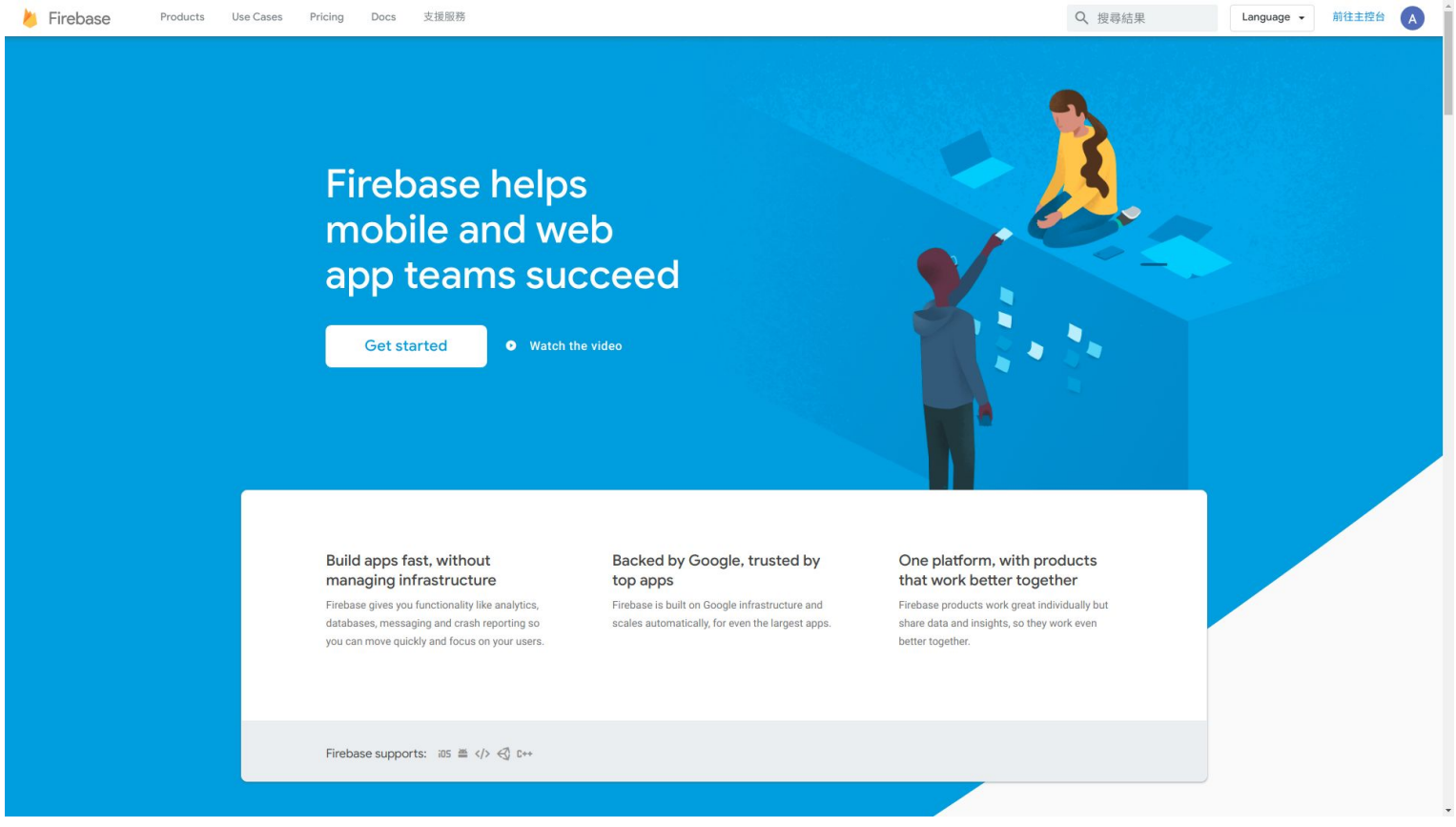
```
import paho.mqtt.client as mqtt
client = mqtt.Client()
client.connect('192.168.1.38') # broker ip
client.publish('MyHome/Bedroom/AirConditioning/Temperature', str(temperature))
client.publish('MyHome/Bedroom/AirConditioning/Humidity', str(humidity))
```

Subscriber

```
def on_connect(client, userdata, flags, rc):
    print("Connected")
    client.subscribe("MyHome/Bedroom/AirConditioning/#")

def on_message(client, userdata, message):
    #do something here

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("localhost")
```


140.112.42.53:3000
React App
(Frontend)
Firebase
Python API
MQTT Broker + Subscriber
MQTT Publisher
DHT11
(Digital Humidity and
Temperature Sensor)

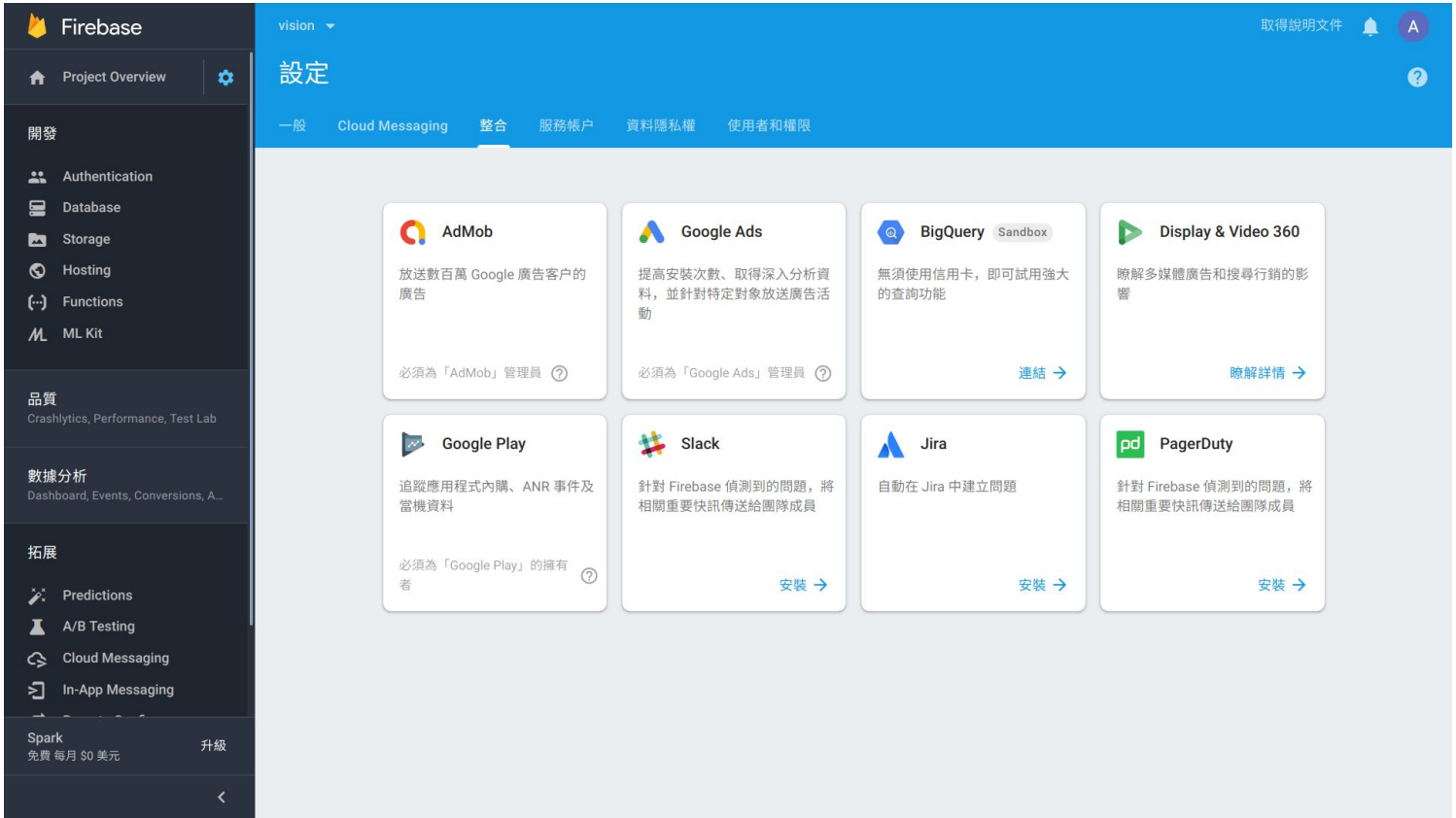
Firebase


Firebase
Products
Use Cases
Pricing
Docs
支援服務
搜尋結果
Language
前往主控台
A
Firebase helps mobile and web app teams succeed
Get started
Watch the video
Build apps fast, without managing infrastructure
Firebase gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users.
Backed by Google, trusted by top apps
Firebase is built on Google infrastructure and scales automatically, for even the largest apps.
One platform, with products that work better together
Firebase products work great individually but share data and insights, so they work even better together.
Firebase supports:

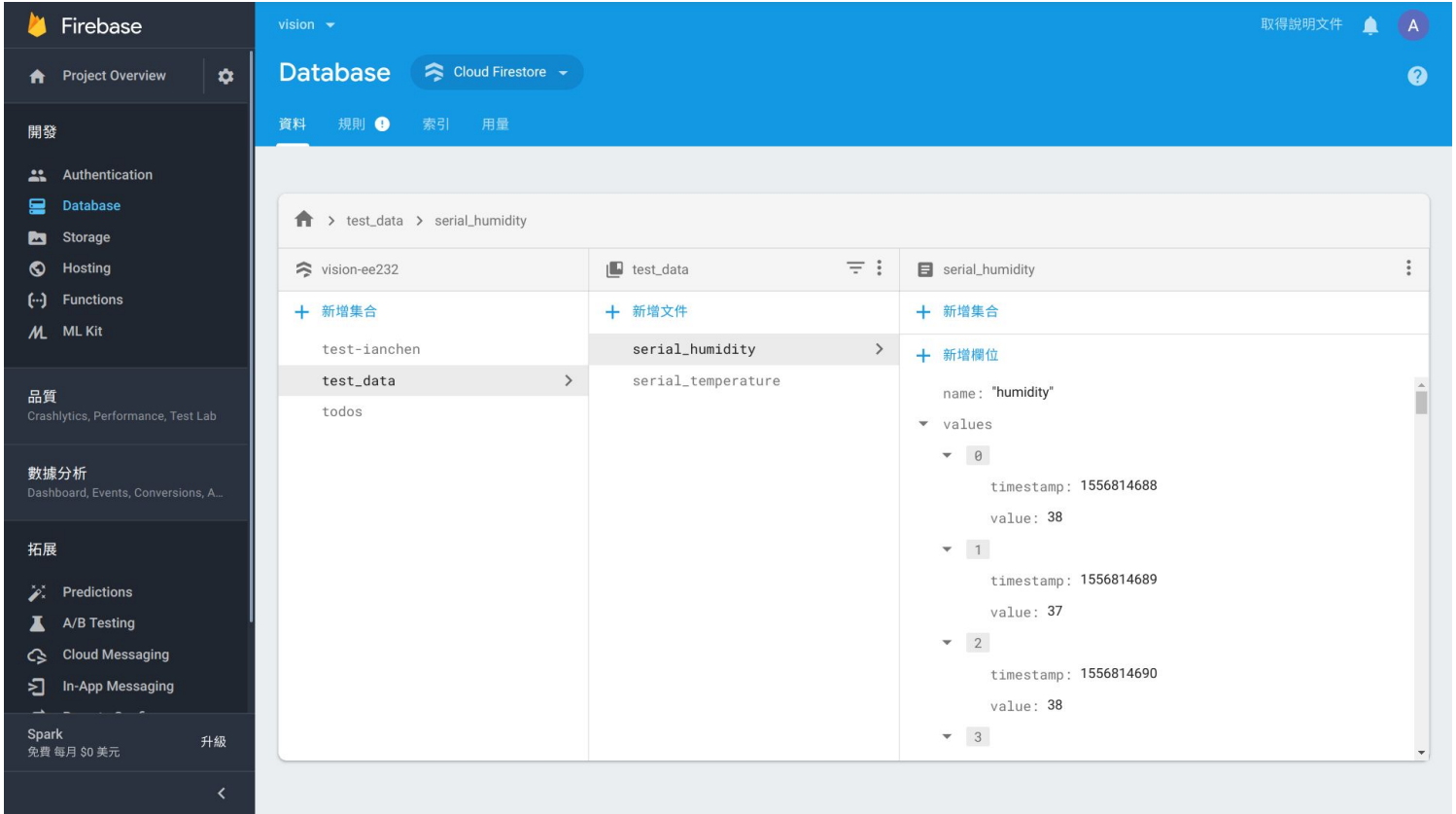
Firebase


Firebase
vision
取得說明文件
A
Project Overview
開發
Authentication
Database
Storage
Hosting
Functions
ML Kit
品質
Crashlytics, Performance, Test Lab
數據分析
Dashboard, Events, Conversions, A...
拓展
Predictions
A/B Testing
Cloud Messaging
In-App Messaging
Spark
免費 每月 $0 美元
升級
vision
Spark 方案
將 Firebase 新增至應用程式即可開始使用
有了我們的 Core SDK，您就能使用大部分的 Firebase 功能，並可透過數據分析查看 iOS 和 Android 應用程式的相關分析資料
iOS
首先請新增應用程式
在數毫秒內即可儲存及同步處理應用程式資料

Firebase


Firebase
Project Overview
開發
Authentication
Database
Storage
Hosting
Functions
ML Kit
品質
Crashlytics, Performance, Test Lab
數據分析
Dashboard, Events, Conversions, A...
拓展
Predictions
A/B Testing
Cloud Messaging
In-App Messaging
Spark
免費 每月 $0 美元
升級
vision
取得說明文件
設定
一般
Cloud Messaging
整合
服務帳戶
資料隱私權
使用者和權限
AdMob
放送數百萬 Google 廣告客戶的廣告
必須為「AdMob」管理員
Google Ads
提高安裝次數、取得深入分析資料，並針對特定對象放送廣告活動
必須為「Google Ads」管理員
BigQuery
Sandbox
無須使用信用卡，即可試用強大的查詢功能
連結
Display & Video 360
瞭解多媒體廣告和搜尋行銷的影響
瞭解詳情
Google Play
追蹤應用程式內購、ANR 事件及當機資料
必須為「Google Play」的擁有者
Slack
針對 Firebase 偵測到的問題，將相關重要快訊傳送給團隊成員
安裝
Jira
自動在 Jira 中建立問題
安裝
PagerDuty
針對 Firebase 偵測到的問題，將相關重要快訊傳送給團隊成員
安裝

Firebase - Database

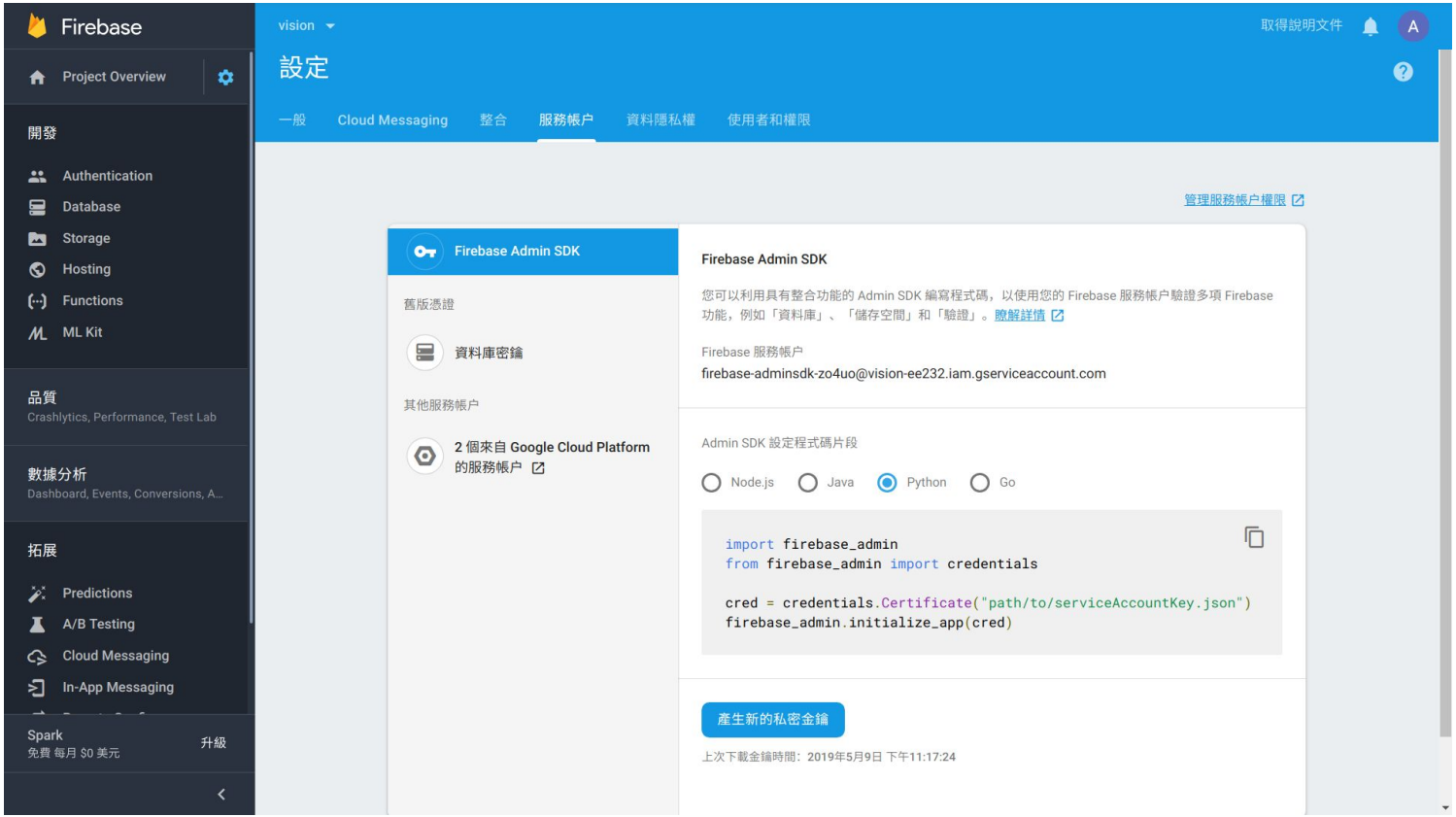


Firebase Admin SDK

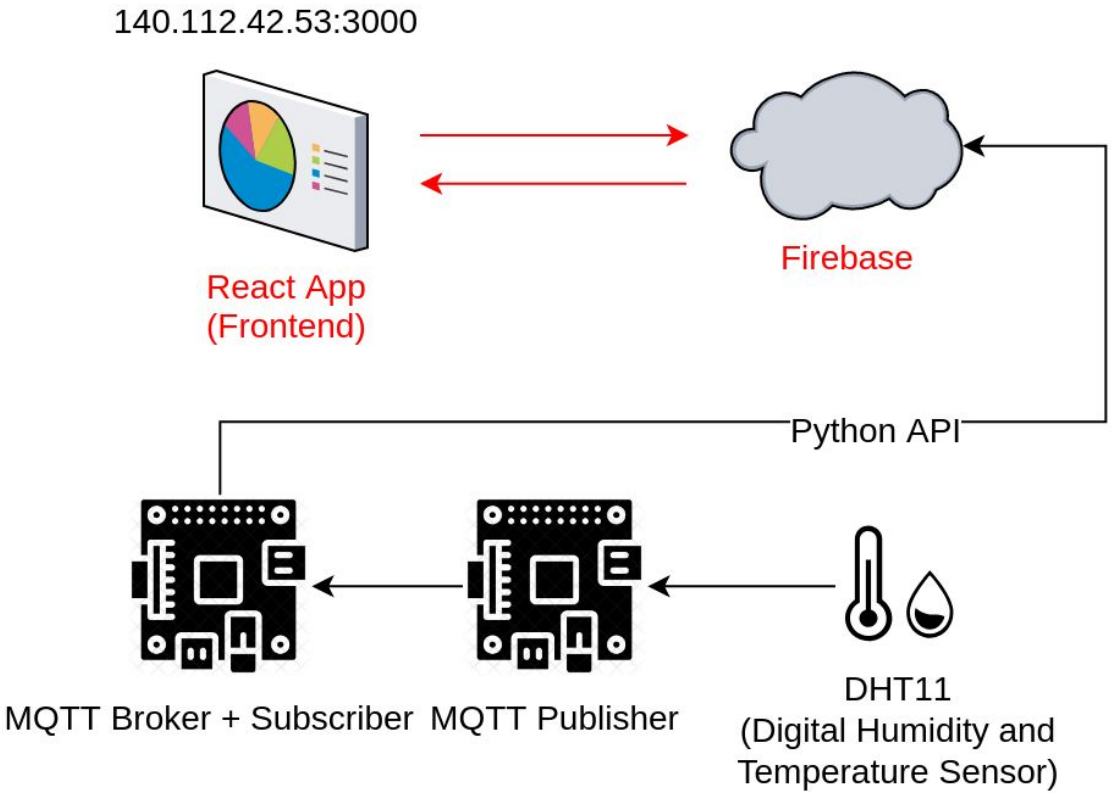
Node.js Java Python Go C#

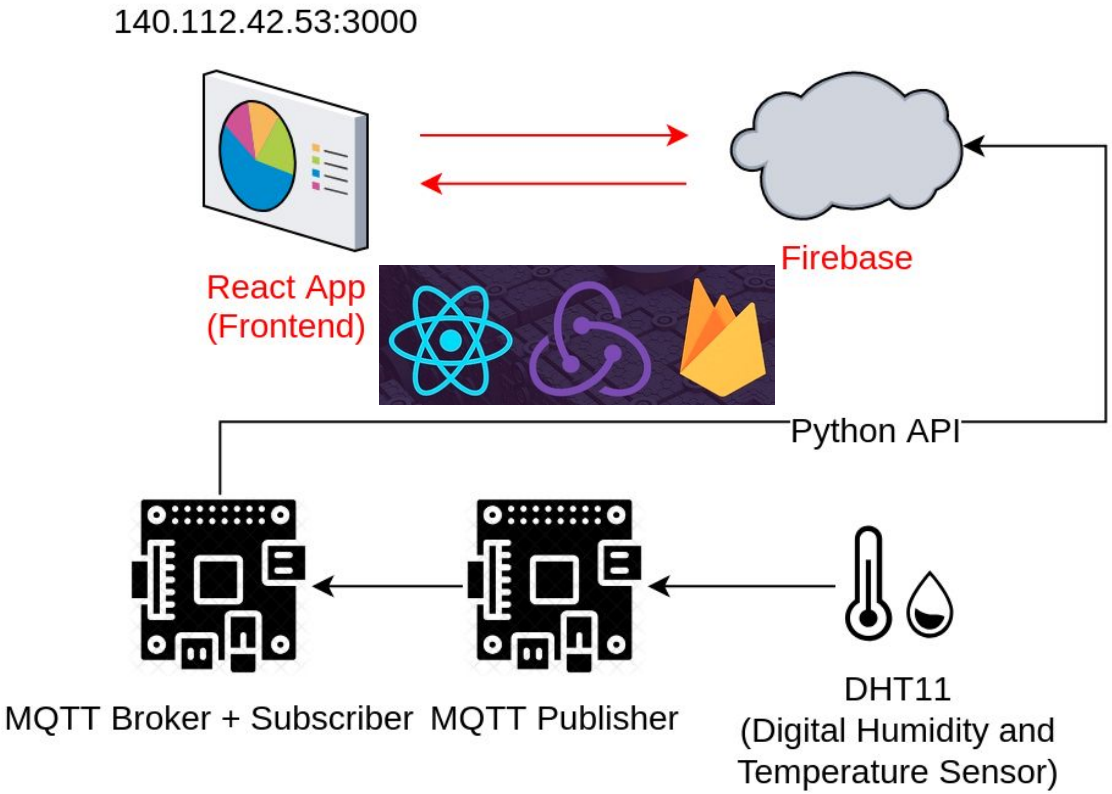
```python
# Import the Firebase service
from firebase_admin import auth

# Initialize the default app
default_app = firebase_admin.initialize_app(cred)
print(default_app.name)  # "[DEFAULT]"

# Retrieve services via the auth package...
# auth.create_custom_token(...)
```

Firebase Admin SDK



```python
cred = credentials.Certificate("./firebase_python_api/vision-admin.json")
firebase_admin.initialize_app(cred)

db = firestore.client()
doc_ref_temp = db.collection('test_data').document('serial_temperature')
doc_ref_humidity = db.collection('test_data').document('serial_humidity')

def on_message(client, userdata, message):

    data_type = message.topic.split("/")[-1]
    data = {'timestamp': int(time.time()), 'value': float(message.payload)}
    if data_type == "Humidity":
        doc_ref_humidity.update({'values': ArrayUnion([data])})

    elif data_type == "Temperature":
        doc_ref_temp.update({'values': ArrayUnion([data])})
```

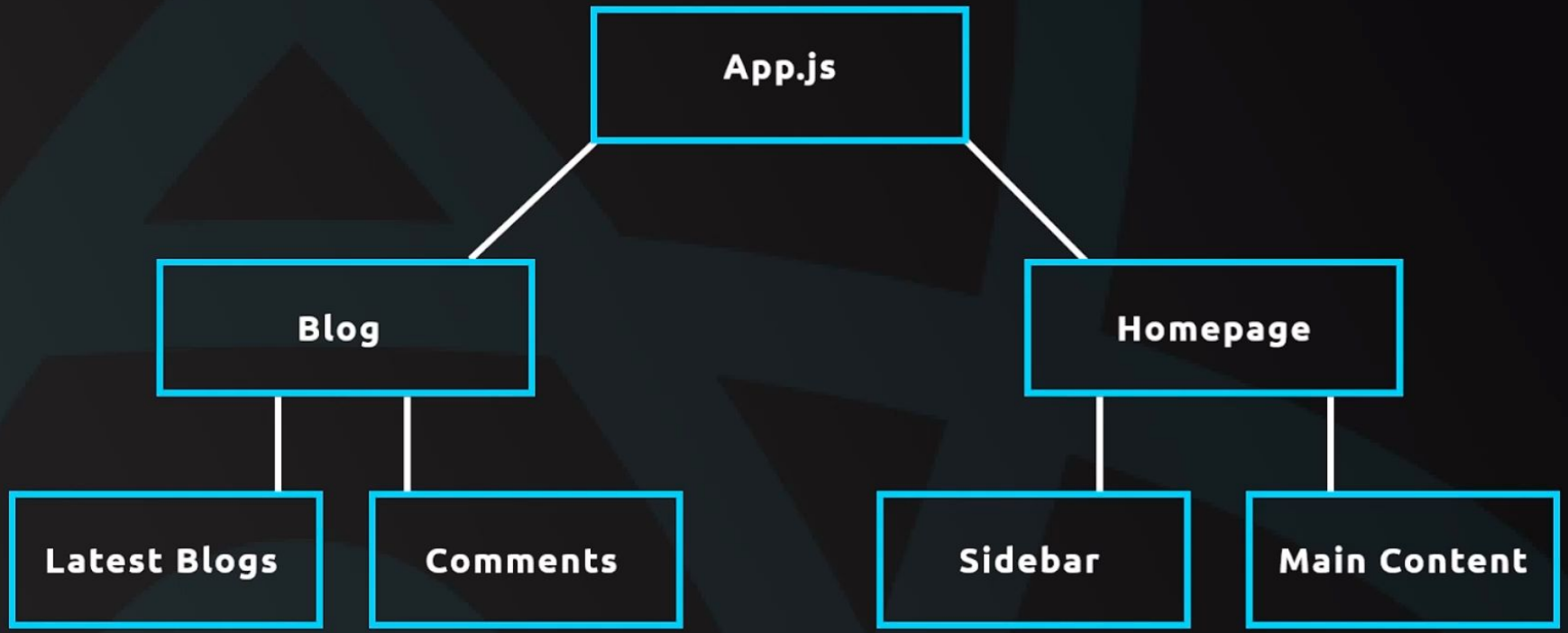

140.112.42.53:3000
React App
(Frontend)
Firebase
Python API
MQTT Broker + Subscriber
MQTT Publisher
DHT11
(Digital Humidity and
Temperature Sensor)


140.112.42.53:3000
Firebase
React App
(Frontend)
Python API
MQTT Broker + Subscriber
MQTT Publisher
DHT11
(Digital Humidity and
Temperature Sensor)

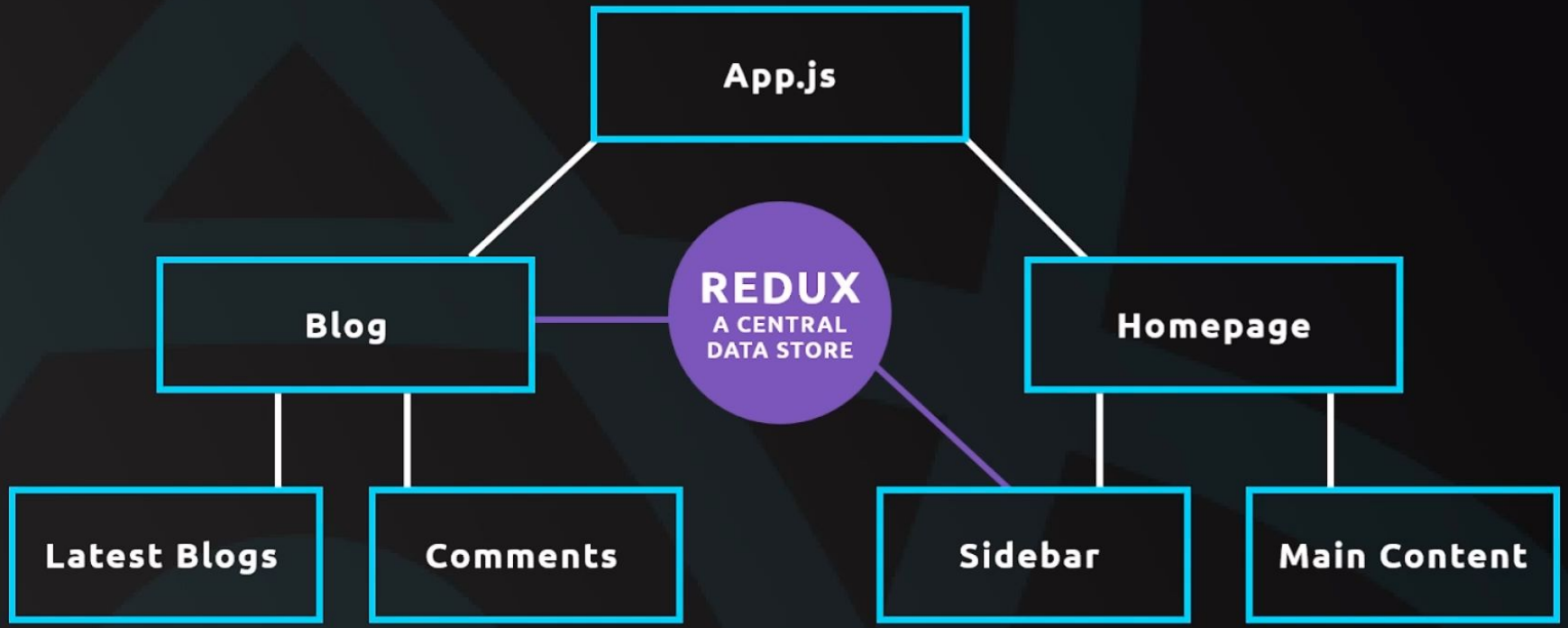
What is Redux?

A predictable state container for JavaScript apps.

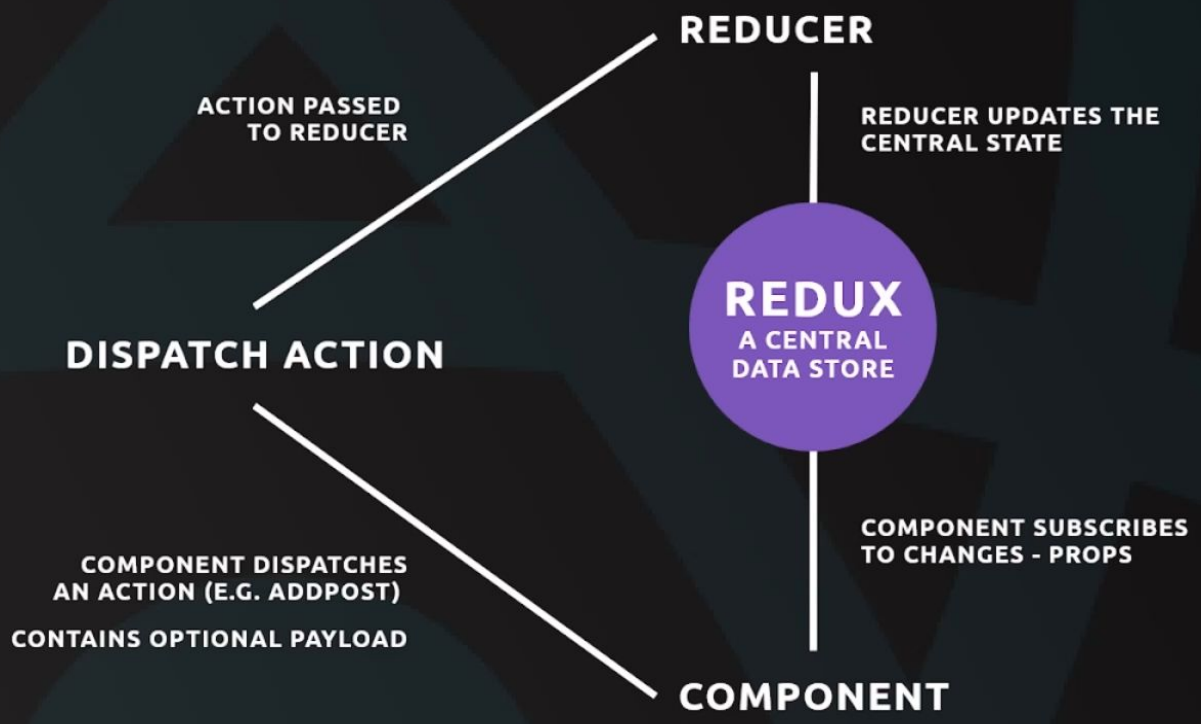
Redux


App.js
Blog
Homepage
Latest Blogs
Comments
Sidebar
Main Content

Redux


App.js
Blog
REDUX
A CENTRAL
DATA STORE
Homepage
Latest Blogs
Comments
Sidebar
Main Content

Redux


REDUCER
ACTION PASSED TO REDUCER
REDUCER UPDATES THE CENTRAL STATE
REDUX
A CENTRAL DATA STORE
DISPATCH ACTION
COMPONENT SUBSCRIBES TO CHANGES - PROPS
COMPONENT DISPATCHES AN ACTION (E.G. ADDPOST)
CONTAINS OPTIONAL PAYLOAD
COMPONENT

```
1  const { createStore } = Redux;
2
3  const initState ={
4    todos: [],
5    notes: []
6  }
7
8  function myreducer(state = initState, action){
9    console.log("action:", action, "state:", state);
10 }
11 const store = createStore(myreducer);
12
13 store.subscribe(() => {
14   console.log("action occured");
15 })
16
17 const todoAction = { type: "ADD_TODO", todo: "buy milk"};
18 store.dispatch(todoAction)
```

Predictable

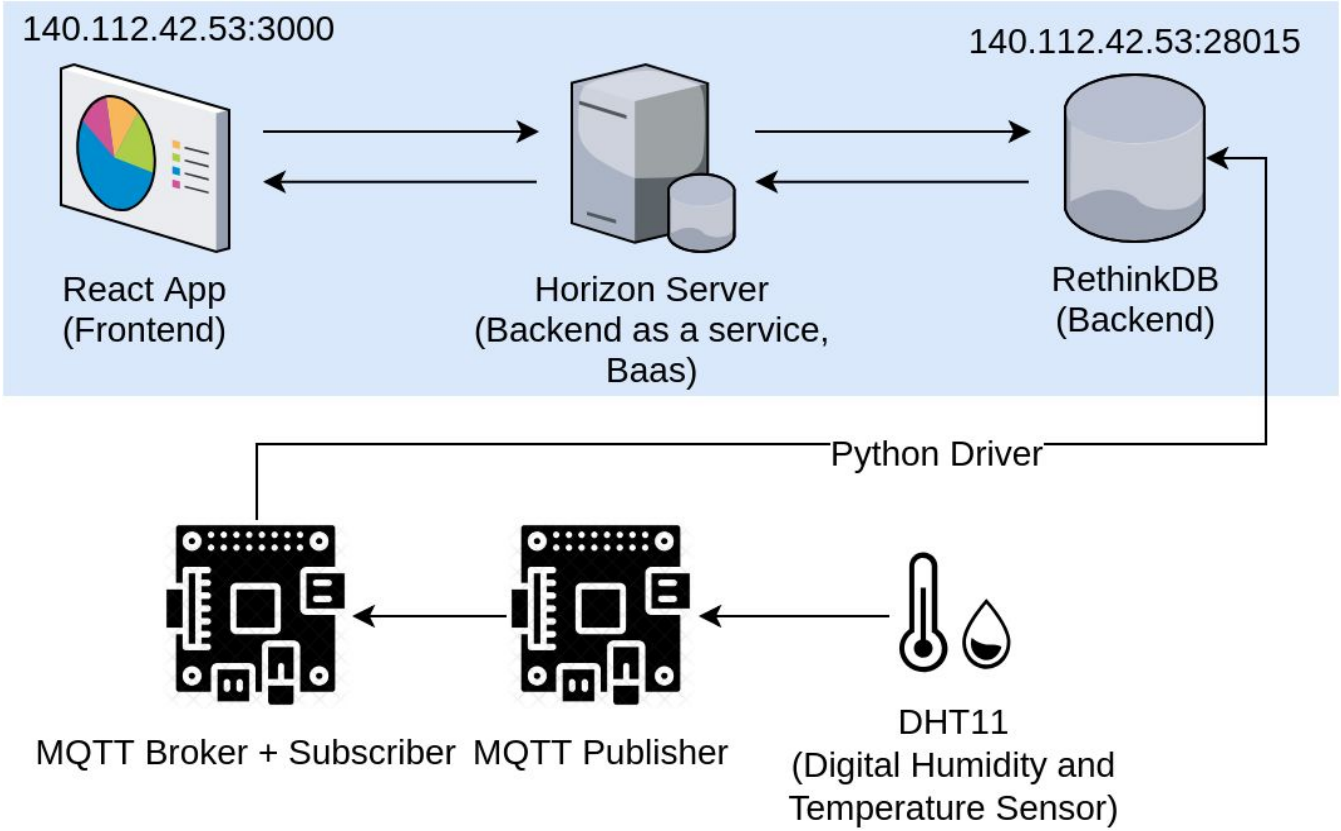
Redux attempts to make state mutations predictable by imposing certain restrictions on how and when updates can happen. These restrictions are reflected in the three principles of Redux:

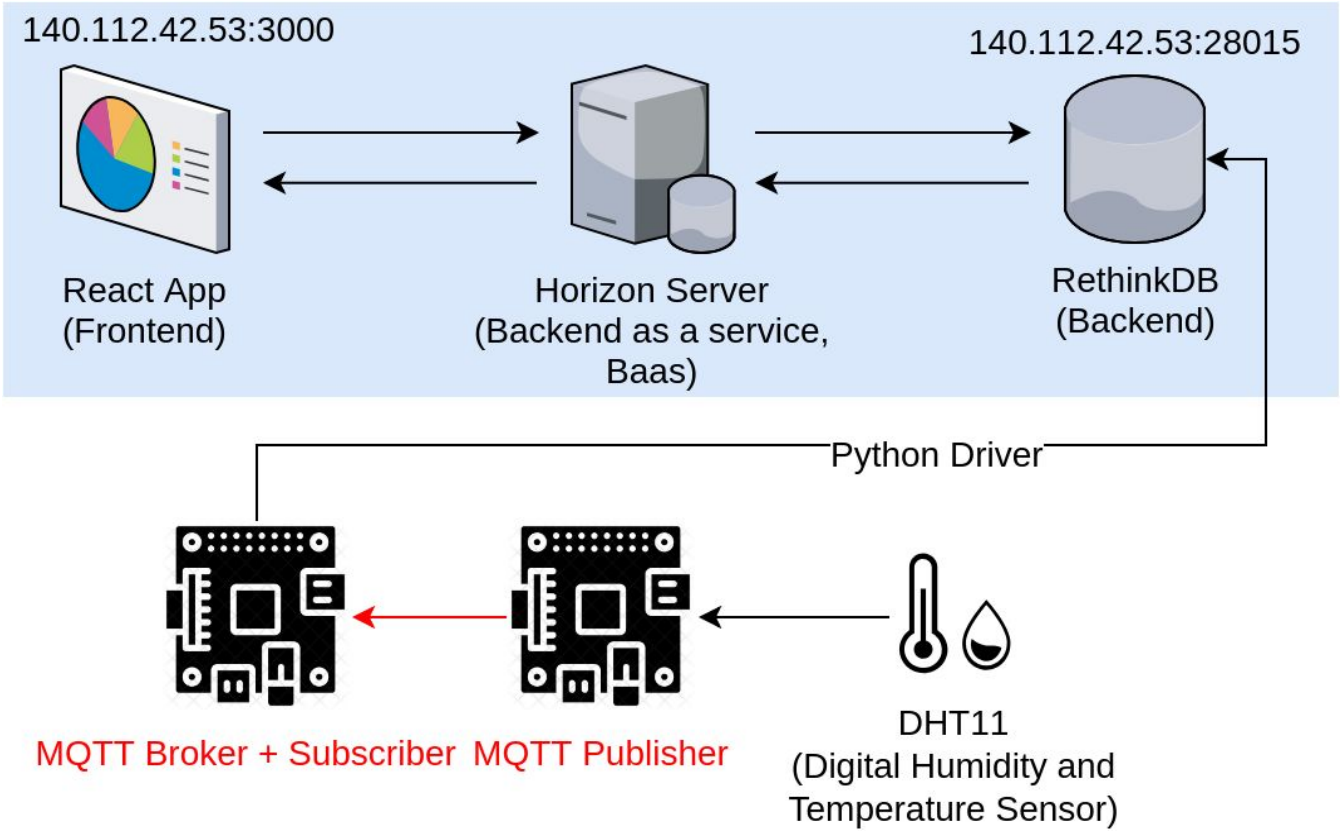
- **Single source of truth**: The state of your whole application is stored as a tree of plain objects and arrays within a single **store**. (How much you put in the store is up to you - not all data needs to live there.)
- **State is read-only**: State updates are caused by *dispatching* an **action**, which is a plain object describing what happened. The rest of the app is not allowed to modify the state tree directly.
- **Changes are made with pure functions**: All state updates are performed by pure functions called **reducers**, which are `(state, action) => newState`

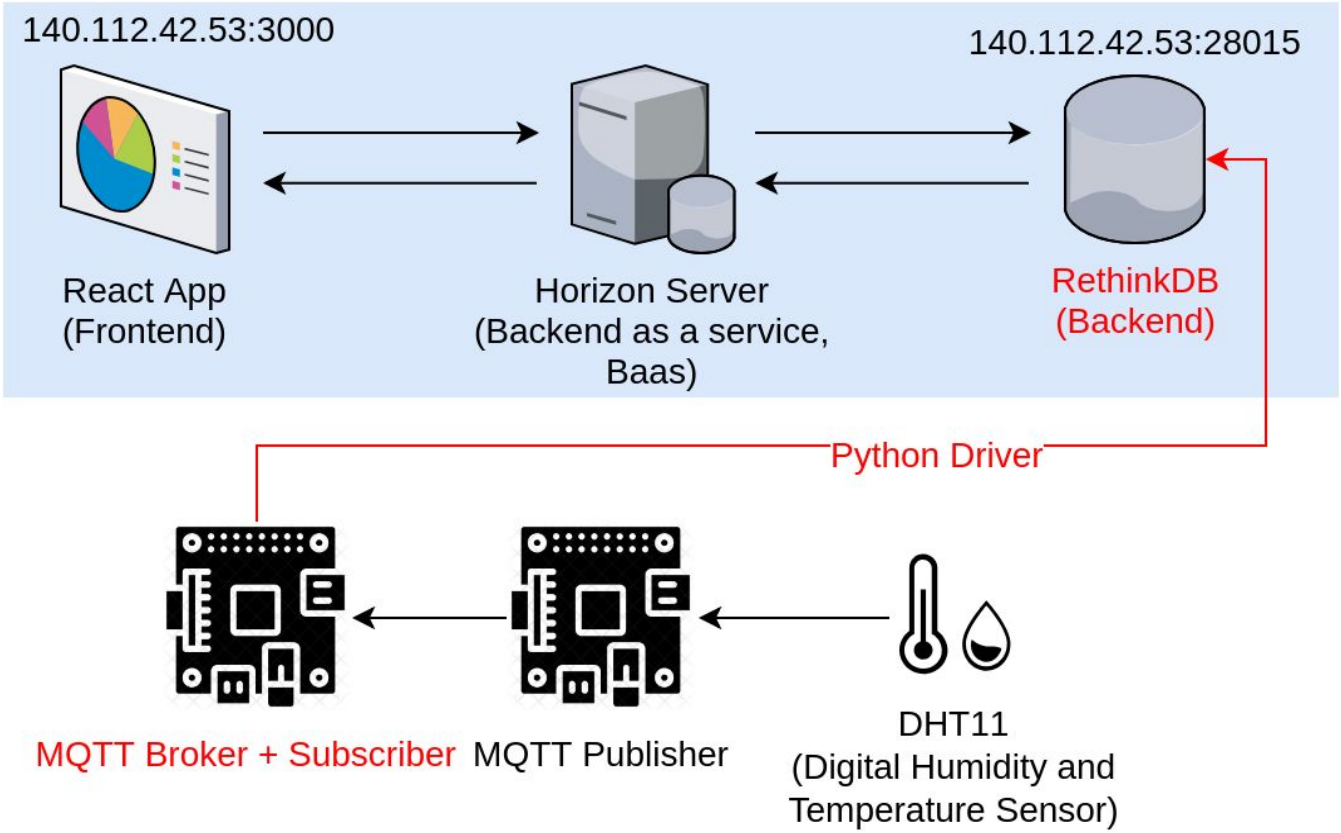
```
1  import React, {Component} from 'react'
2  import {connect} from 'react-redux'
3
4  class Home extends Component {
5    render(){
6      // do somethings here
7
8      handleClick = () =>{
9        this.props.addTodo(this.props.content);
10     }
11
12     const {todos} = this.props
13   }
14 }
15 const mapStateToProps = (sate) =>{
16   return todos: state.todos
17 }
18
19 const mapDispatchToProps = (dispatch) =>{
20   return addTodo: (cotent) => {dispatch({type: 'ADD_TODO', todos:content)}
21 }
22
23 export default connect(mapStateToProps, mapDispatchToProps)(Home)
```

```
import { createStore, combineReducers, compose } from 'redux'
import { reduxFirestore, firestoreReducer } from 'redux-firestore'
import firebase from 'firebase/app'
import 'firebase/auth'
import 'firebase/database'
import 'firebase/firestore'

const firebaseConfig = {} // from Firebase Console
const rfConfig = {} // optional redux-firestore Config Options

// Initialize firebase instance
firebase.initializeApp(firebaseConfig)
// Initialize Cloud Firestore through Firebase
firebase.firestore();

// Add reduxFirestore store enhancer to store creator
const createStoreWithFirebase = compose(
  reduxFirestore(firebase, rfConfig), // firebase instance as first argument, rfConfig as optional second
)(createStore)

// Add Firebase to reducers
const rootReducer = combineReducers({
  firestore: firestoreReducer
})

// Create store with reducers and initial state
const initialState = {}
const store = createStoreWithFirebase(rootReducer, initialState)
```

Data Visualization with RethinkDB




140.112.42.53:3000
140.112.42.53:28015
React App
(Frontend)
Horizon Server
(Backend as a service,
Baas)
RethinkDB
(Backend)
Python Driver
MQTT Broker + Subscriber
MQTT Publisher
DHT11
(Digital Humidity and
Temperature Sensor)


140.112.42.53:3000
140.112.42.53:28015
React App
(Frontend)
Horizon Server
(Backend as a service,
Baas)
RethinkDB
(Backend)
Python Driver
MQTT Broker + Subscriber
MQTT Publisher
DHT11
(Digital Humidity and
Temperature Sensor)

RethinkDB

What is RethinkDB?

- **Open-source** database for building realtime web applications
- **NoSQL** database that stores schemaless JSON documents
- **Distributed** database that is easy to scale
- **High availability** database with automatic failover and robust fault tolerance

RethinkDB client drivers

RethinkDB

faq docs api community blog

Getting started
- Thirty-second quickstart
- Install the server
- Install client drivers
- Start a RethinkDB server

The ReQL query language

The RethinkDB data model

ReQL in practice

Administration

Deployment

Troubleshooting

Integration

RethinkDB architecture

Installing RethinkDB client drivers

Official drivers

JavaScript | Ruby | Python

Java

Current community-supported drivers

These drivers have been updated to use the JSON driver protocol and at least RethinkDB 2.0 ReQL.

C# bchavez	C# mfenniak	C++
Clojure	Common Lisp	Dart
Delphi	Elixir	Erlang
Go	Haskell	JS neumino
Lua	Nim	Perl
PHP	R	Rust
Swift		

RethinkDB Dashboard Tables Servers Data Explorer Logs Search

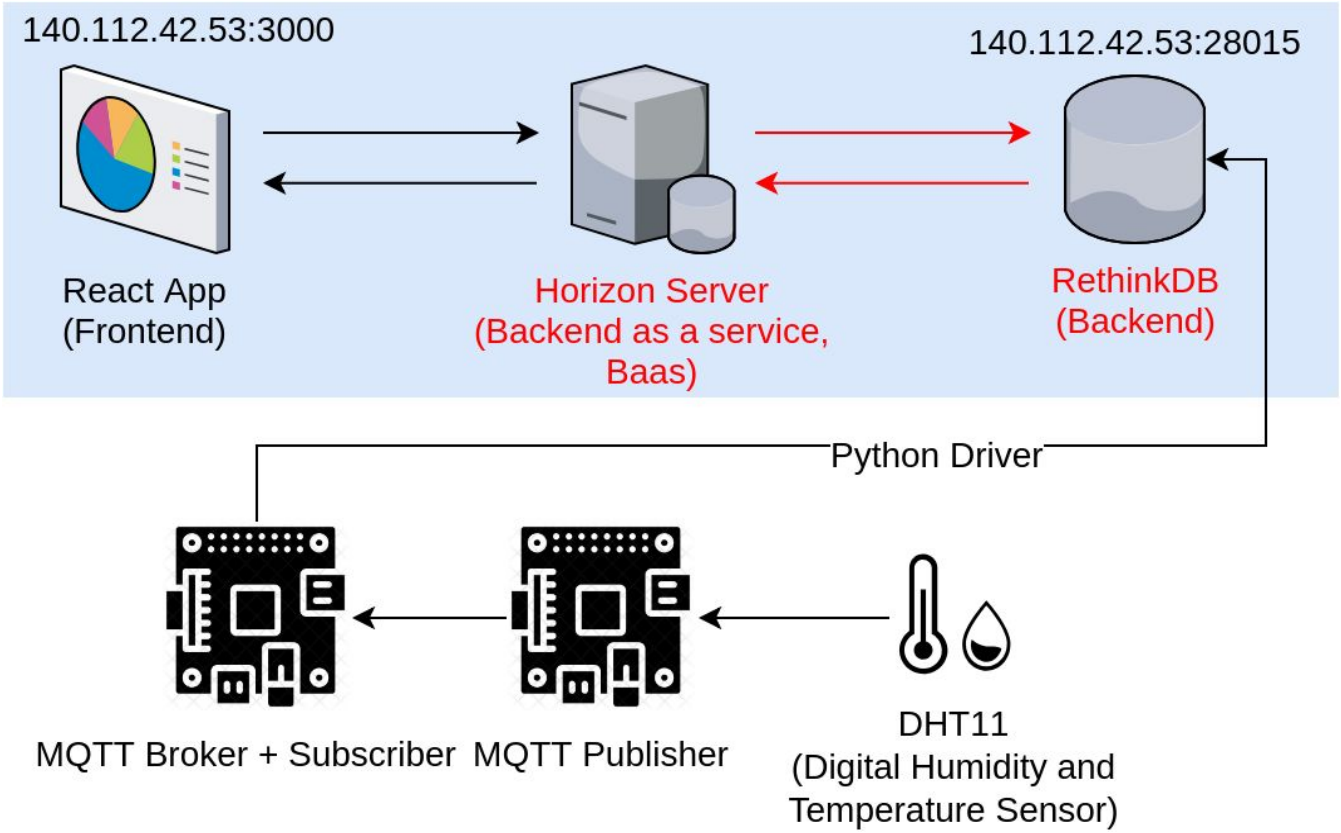
Connected to xeon_e5_1620_q... | Issues No issues | Servers 1 connected | Tables 8/8 ready

Tables in the cluster

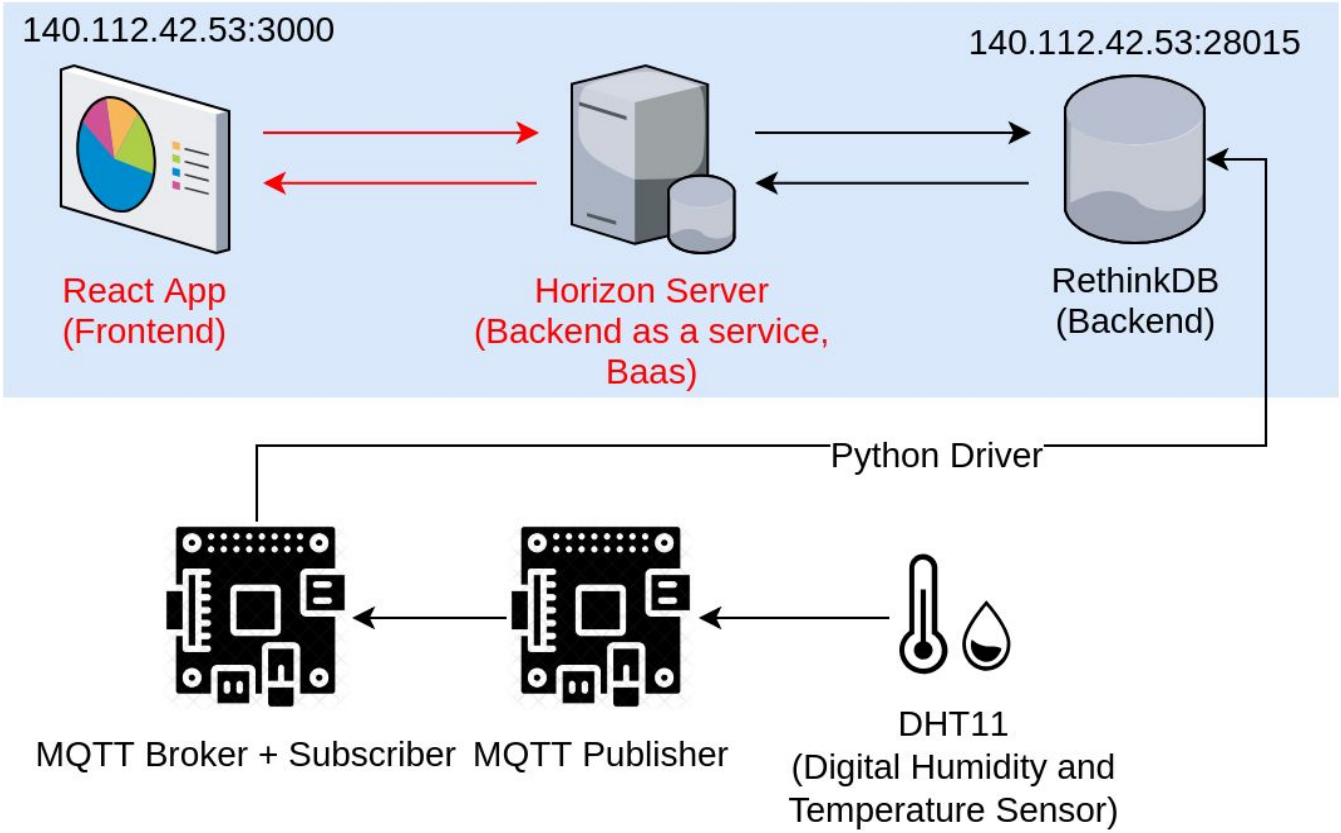
+ Add Database | – Delete selected tables

DATABASE			
example_app		+ Add Table	Delete Database
firetony	1 shard, 1 replica	Ready 1/1	
hz_collections	1 shard, 1 replica	Ready 1/1	
hz_groups	1 shard, 1 replica	Ready 1/1	
hz_users_auth	1 shard, 1 replica	Ready 1/1	
messages	1 shard, 1 replica	Ready 1/1	
tv_shows	1 shard, 1 replica	Ready 1/1	
users	1 shard, 1 replica	Ready 1/1	
pulse		+ Add Table	Delete Database
pulse	1 shard, 1 replica	Ready 1/1	
test		+ Add Table	Delete Database

```python
rdb = r.RethinkDB()
rdb.connect('140.112.42.53', 28015).repl()

def on_message(client, userdata, message):
    data_type = message.topic.split("/")[-1]
    data = {'timestamp': int(time.time()), 'value': float(message.payload)}
    if data_type == "Humidity":
        rdb.db('example_app').table('firetony').get(1002).update(
            {"values": rdb.row["values"].append(data)}
        ).run()
    elif data_type == "Temperature":
        rdb.db('example_app').table('firetony').get(1003).update(
            {"values": rdb.row["values"].append(data)}
        ).run()
```


140.112.42.53:3000
140.112.42.53:28015
React App
(Frontend)
Horizon Server
(Backend as a service,
Baas)
RethinkDB
(Backend)
Python Driver
MQTT Broker + Subscriber
MQTT Publisher
DHT11
(Digital Humidity and
Temperature Sensor)

- A **backend server** built with Node.js and RethinkDB that supports data persistence, realtime streams, input validation, user authentication, and permissions
- A **JavaScript client library** that developers can use on the frontend to store JSON documents in the database, perform queries, and subscribe to live updates
- A **command-line tool** that can generate project templates, start up a local Horizon development server, and help you deploy your Horizon application to the cloud


140.112.42.53:3000
140.112.42.53:28015
React App
(Frontend)
Horizon Server
(Backend as a service,
Baas)
RethinkDB
(Backend)
Python Driver
MQTT Broker + Subscriber
MQTT Publisher
DHT11
(Digital Humidity and
Temperature Sensor)

Connect to horizon server

```
1  // Initialize Horizon instance
2  const horizon = Horizon({host: host + ":" + port});
3  // Initialize firetony table
4  const serial_data = horizon(table);
5
6  getRethinkdbData(serial_data, store);
7  serial_data.watch().subscribe((docs) => { getRethinkdbData(serial_data, store)})
```

Reference

https://firebase.google.com/docs/admin/setup

https://github.com/Destinia/Vision

https://blog.isquaredsoftware.com/presentations/workshops/redux-fundamentals/

https://redux.js.org/basics/basic-tutorial

https://redux.js.org/advanced/advanced-tutorial

https://www.youtube.com/watch?v=OxIDLw0M-m0

https://www.youtube.com/watch?v=Oi4v5uxTY5o

https://www.rethinkdb.com/docs/install-drivers/

https://github.com/rethinkdb/horizon-docs/blob/master/getting-started.md

https://github.com/prescottprue/redux-firestore