前端配置管理系统从0到1

陈韩杰 @CntChen

Why ?

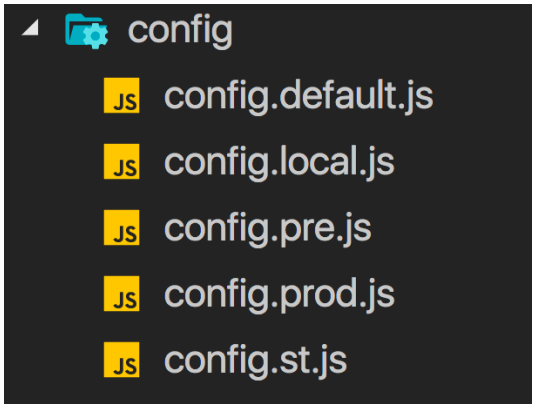
配置无处不在

```
<!-- 使用 zookeeper 广播注册中心暴露服务地址 -->
<dubbo:registry address="zookeeper://127.0.0.1:2181"/>
```

Dubbo RPC ZK 配置

```
➜  / ping -h
ping: option requires an argument -- h
usage: ping [-AaDdfnoQqRrv] [-c count] [-G sweepmaxsize]
            [-g sweepminsize] [-h sweepincrsize] [-i wait]
            [-l preload] [-M mask | time] [-m ttl] [-p pattern]
            [-S src_addr] [-s packetsize] [-t timeout][-W waittime]
            [-z tos] host
```

ping 命令参数



不同环境的配置文件

配置是什么

配置是独立于应用的只读变量

配置伴随应用的整个生命周期

配置有多种加载方式

为什么需要配置

为什么需要配置

配置提升应用的适应性

降低工作难度，提升工作效率

配置需要管理

配置需要分离

配置随业务线性增长

错误的配置会导致程序故障

解决方案

配置分离

网络分发

统一管理

} 配置管理系统

配置管理系统

名称	语言 / 框架	介绍	特点
ACM(阿里云)	云服务	在微服务、DevOps、大数据等场景下极大地减轻配置管理的工作量	云服务, 实时推送, 版本管理
Apollo(携程)	Spring Boot 和 Spring Cloud	分布式配置中心, 适用于微服务配置场景	配置实时生效, 客户端支持 Java、.Net
Disconf(百度)	SpringMvc	Distributed Configuration Management Platform(分布式配置管理平台)	配置实时生效
Spring Cloud Config	Spring	Spring Cloud Config provides server and client-side support for externalized configuration in a distributed system	Git 来存储配置信息
Etcd(CoreOS)	go	Distributed reliable key-value store for the most critical data of a distributed system	gRPC, Secure

前端业务现状

没有配置 / 配置在代码中

后台业务接口返回

CC 平台的*渠道综合管理系统*

前端业务面临问题

没有配置分离

效率低下

职责不明

新增配置难

散落

配置管理系统

名称	语言 / 框架	介绍	特点
ACM(阿里云)	云服务	在微服务、DevOps、大数据等场景下极大地减轻配置管理的工作量	云服务, 实时推送, 版本管理
Apollo(携程)	Spring Boot 和 Spring Cloud	分布式配置中心, 适用于微服务配置场景	配置实时生效, 客户端支持 Java、.Net
Disconf(百度)	SpringMvc	Distributed Configuration Management Platform(分布式配置管理平台)	配置实时生效
Spring Cloud Config	Spring	Spring Cloud Config provides server and client-side support for externalized configuration in a distributed system	Git 来存储配置信息
Etcd(CoreOS)	go	Distributed reliable key-value store for the most critical data of a distributed system	gRPC, Secure

没有适合前端的系统

已有开源系统主要支持分布式后台服务

技术栈与后台紧密结合

配置项数据不灵活

无法与前端系统直接交互

没有适合前端的系统

没有适合前端业务场景的, 开箱即用的,开源的系统

我们需要，业界没有


不要给我说什么
react/angular/Typescript /vu
es6/es7/babel/await /async
webpack/Browserify
老夫写代码就用
jQuery！
复制！
粘贴！
拿起键
就是
干！

我们需要，业界没有

M

BSS / moonlight-editor
前端业务配置管理系统 moonlight 的 Web 管理网站

M

BSS / moonlight-node-server
前端业务配置管理系统 moonlight 的 Node 服务器

M

BSS / moonlight-mongodb
前端业务配置管理系统 moonlight 的 mongoDB 数据库

M

BSS / moonlight-plan
前端业务配置管理系统 moonlight 的开发文档和开发计划

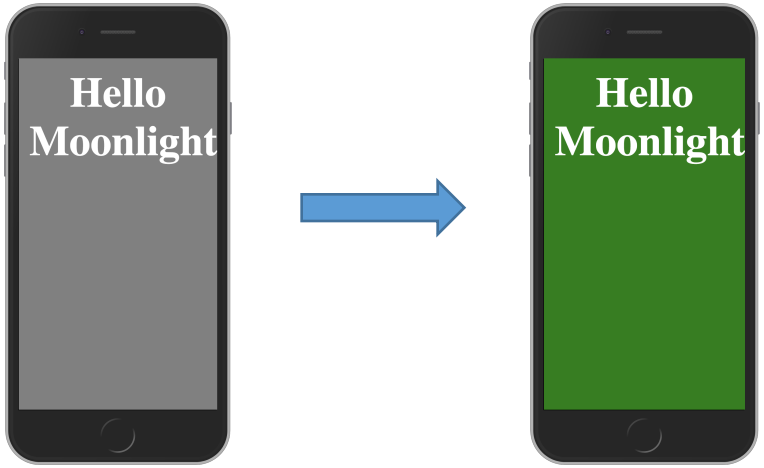
Hello Moonlight

举个例子

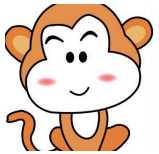
Moonlight Basic Demo
http://10.75.112.153:4430/dev/moonlight/moonlight-dem...

场景一：这个需求很简单

这个需求很简单，页面换个背景色，**今晚上线!**



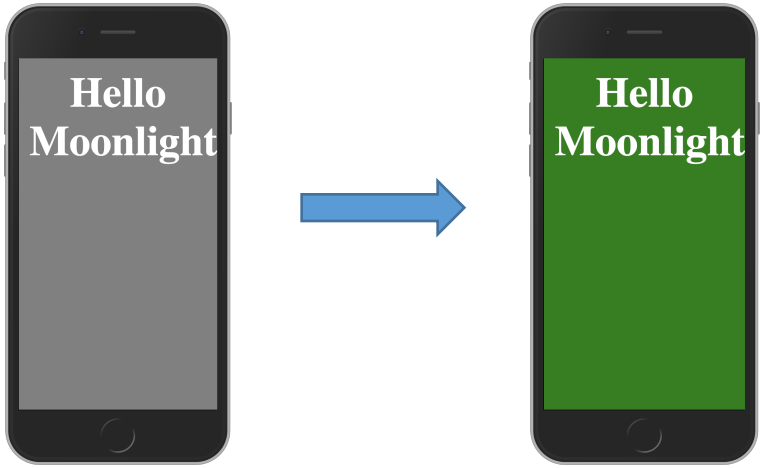
需求沟通 => 开发 => 测试 => 持续集成 => 走OA
=> 运维预发布 => 发布 => 生产验证，**最快明天!**



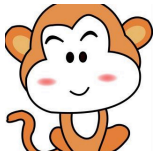
今晚加班!

场景一：这个需求很简单

这个需求很简单，页面换个背景色，今晚上线！



前端配置管理系统已经支持，两分钟后上线.

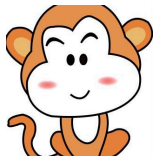


今晚开黑！

场景二：今晚三点告诉你点事

今晚三点合作方服务下线，页面的话术需要调整.

发布版本无法准确把控时间，无法支持!

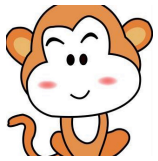


+1

场景二：今晚三点告诉你点事

今晚三点合作方服务下线，页面的话术需要调整.

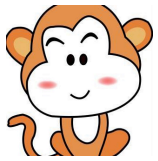
没问题！前端配置管理系统支持，需求将定时上线.



场景三：五分钟内给我处理好

页面出问题了，五分内给我处理好. @dog @monkey @panda

不是我改的，我不知道.



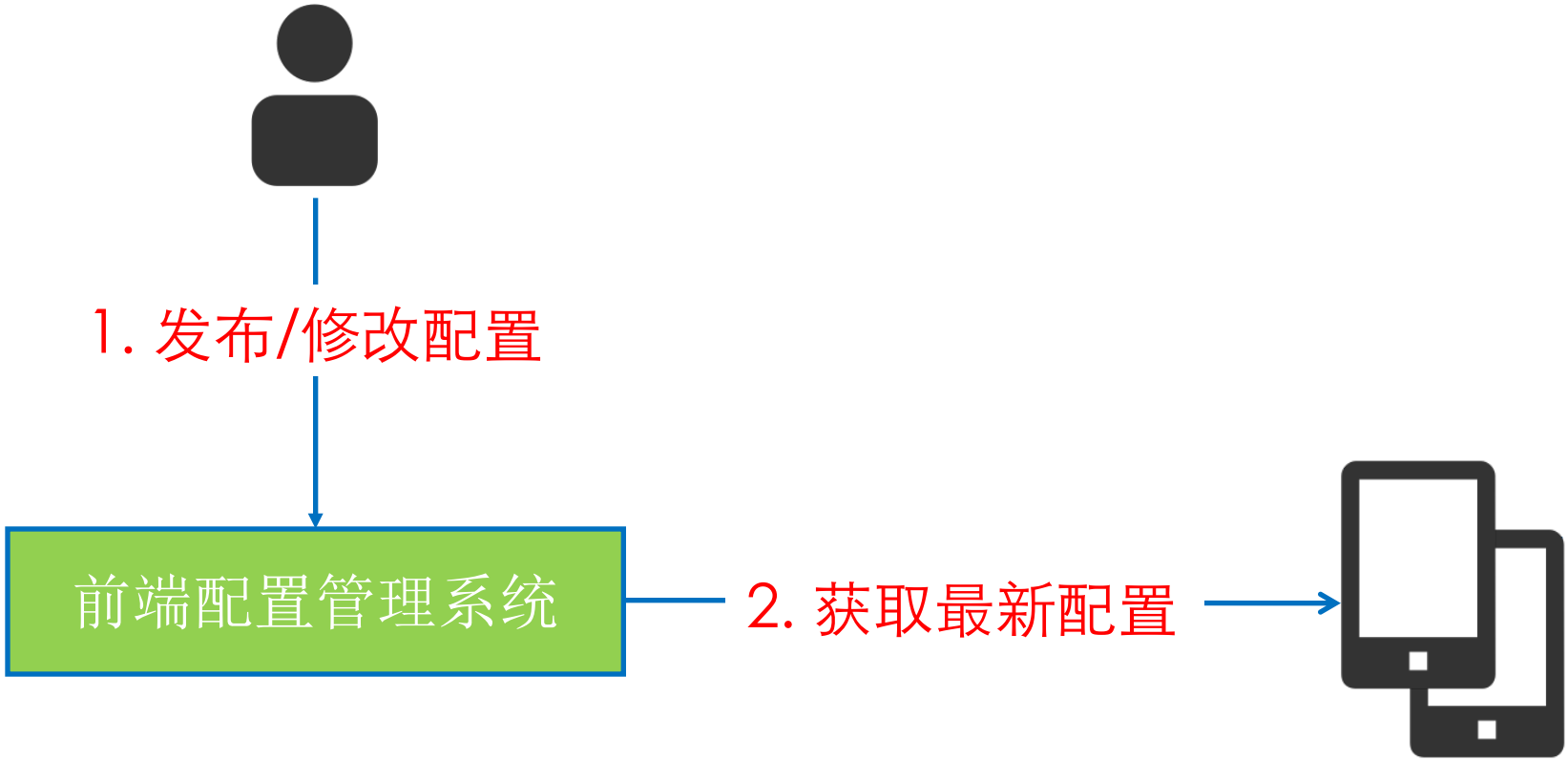
不是我支持上线的，我不知道.

场景三: 五分钟内给我处理好

页面出问题了, 五分内给我处理好. @dog @monkey @panda

已经回退配置, 优先保障业务可用.

我们做了啥?



What is Moonlight

前端配置管理系统

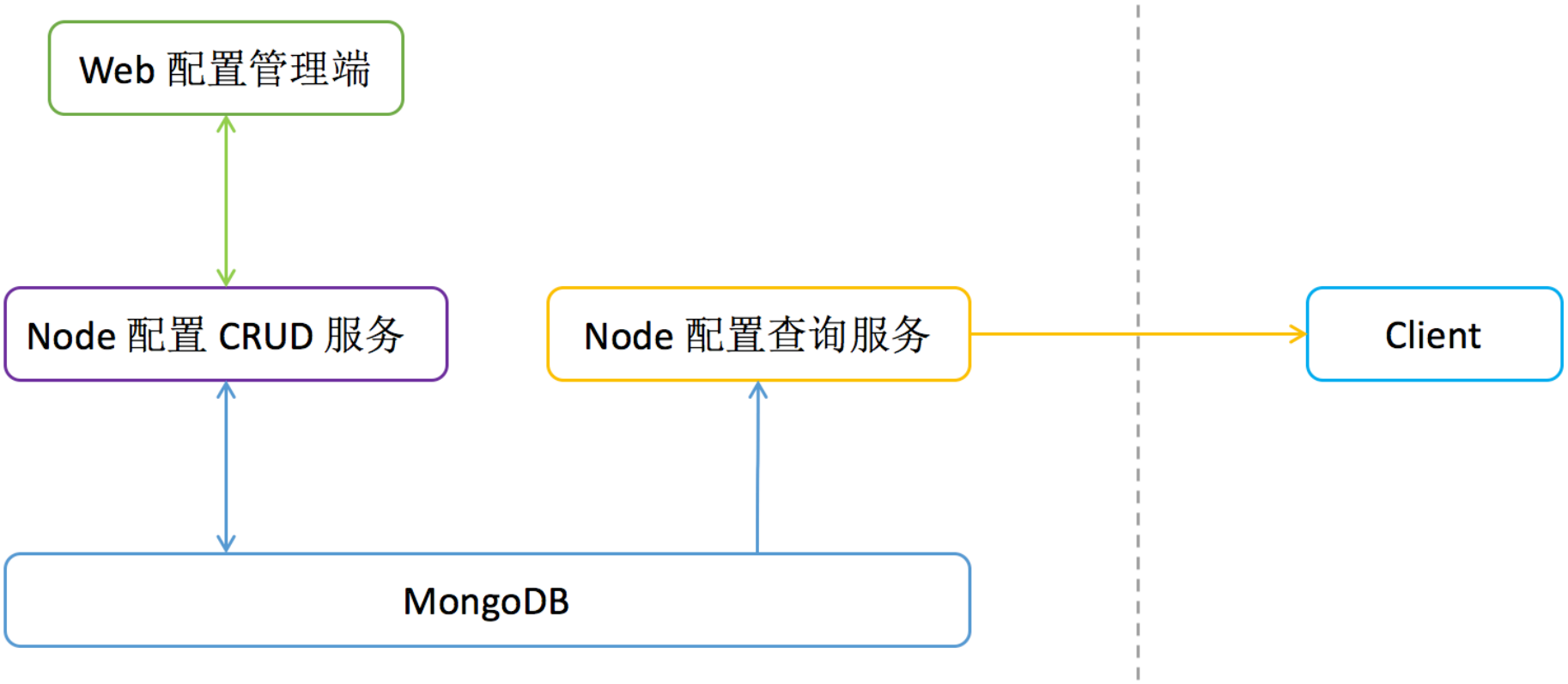
统一管理前端散落的配置项

提供可视化的配置项增删改查操作

提供统一的配置项查询接口

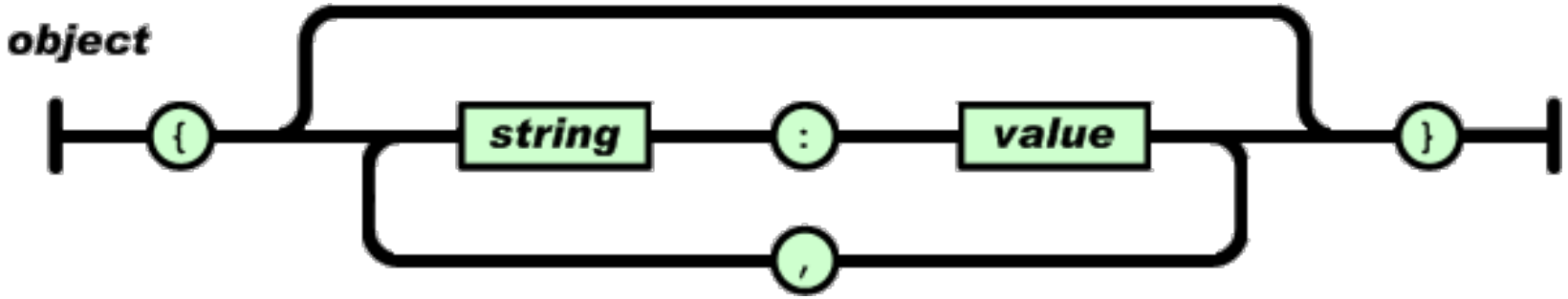
前端友好的技术栈

系统架构

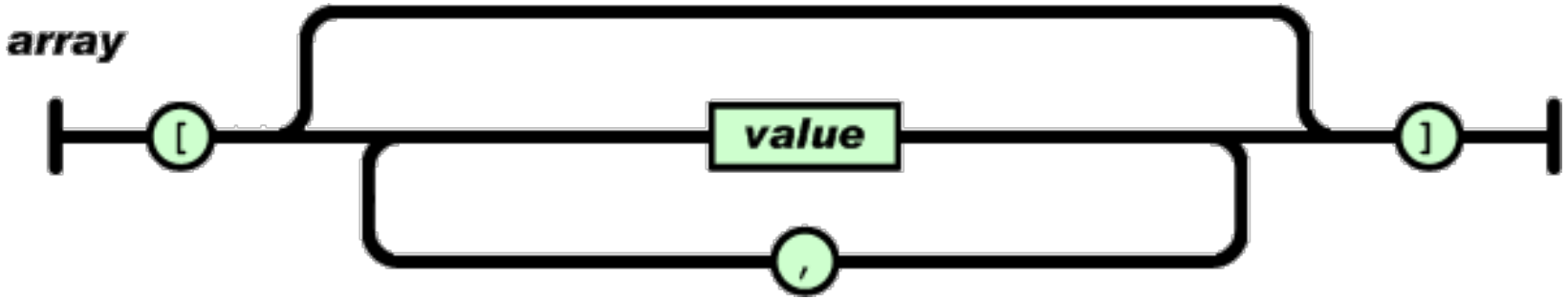


配置项是JSON

JavaScript Object Notation

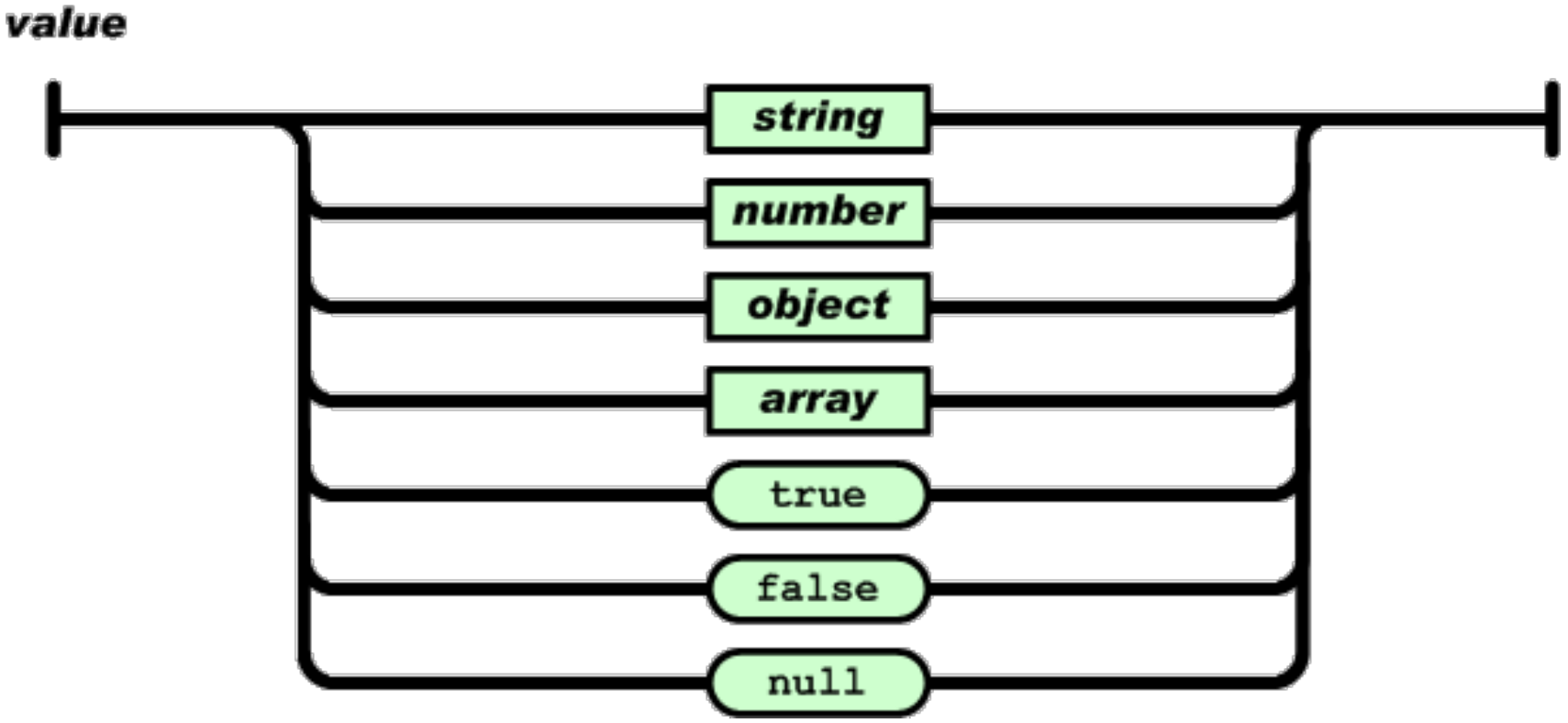


或



配置项是JSON

JavaScript Object Notation



配置项是JSON

```
{
  backgroundColor: "gray",
  title: "Hello Moonlight"
}
```

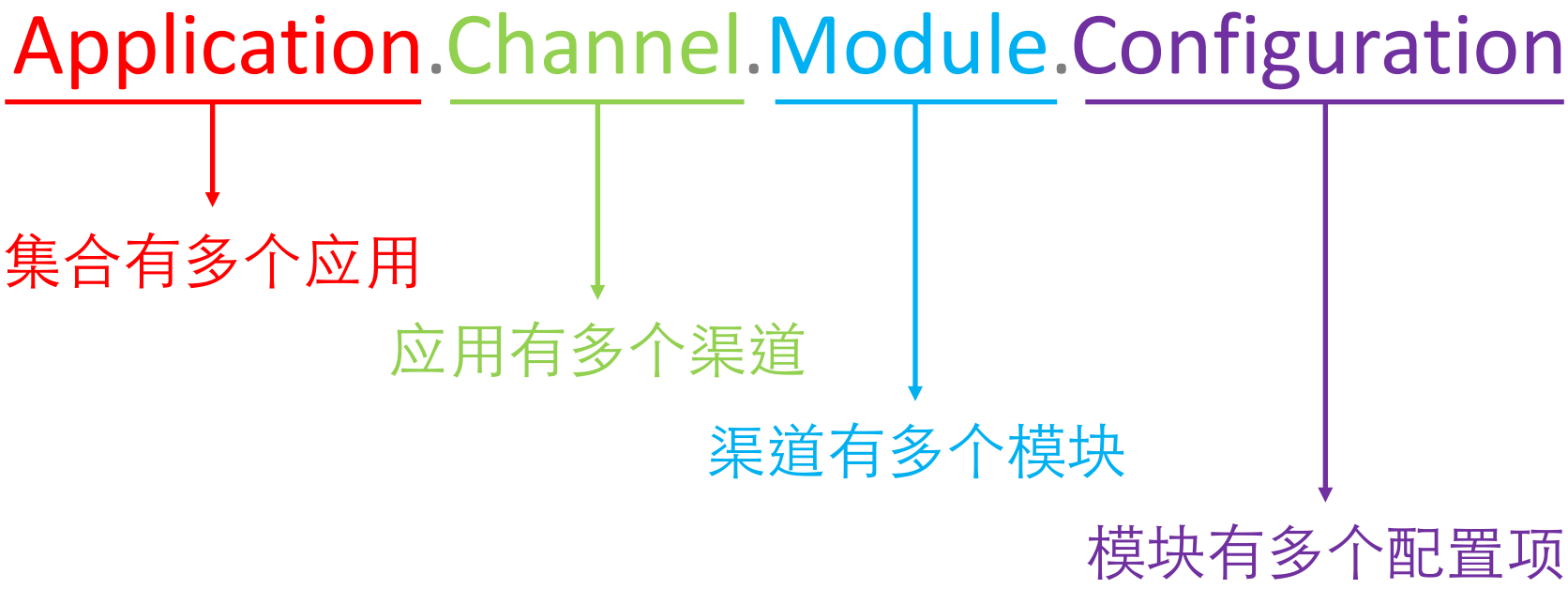

```
[
  {
    link: "http://www.cmbchina.com/",
    title: "招商银行",
    logo: "http://www.cmbchina.com/images/main_logo.gif"
  },
  {
    link: "http://www.chinaunicom.com/",
    title: "中国联通",
    logo: "http://www.chinaunicom.com/images/icon.jpg"
  }
]
```

命名空间

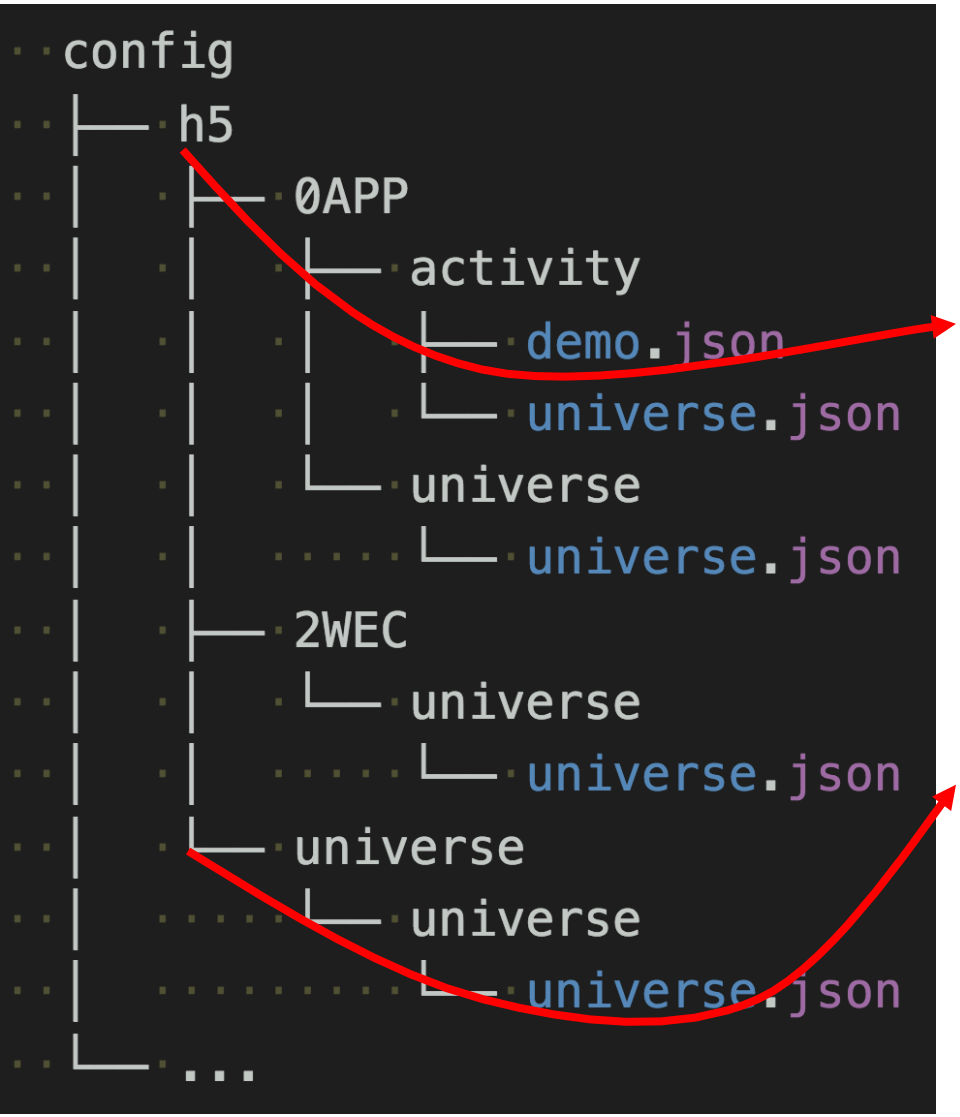
Application.Channel.Module.Configuration

命名空间是层级关系的映射

命名空间


Application.Channel.Module.Configuration
集合有多个应用
应用有多个渠道
渠道有多个模块
模块有多个配置项

命名空间



h5.0APP.activity.demo

前端页面 => APP渠道 => 活动模块 => Demo页面

h5.universe.universe.universe

前端页面 => 全渠道 => 通用模块 => 通用配置

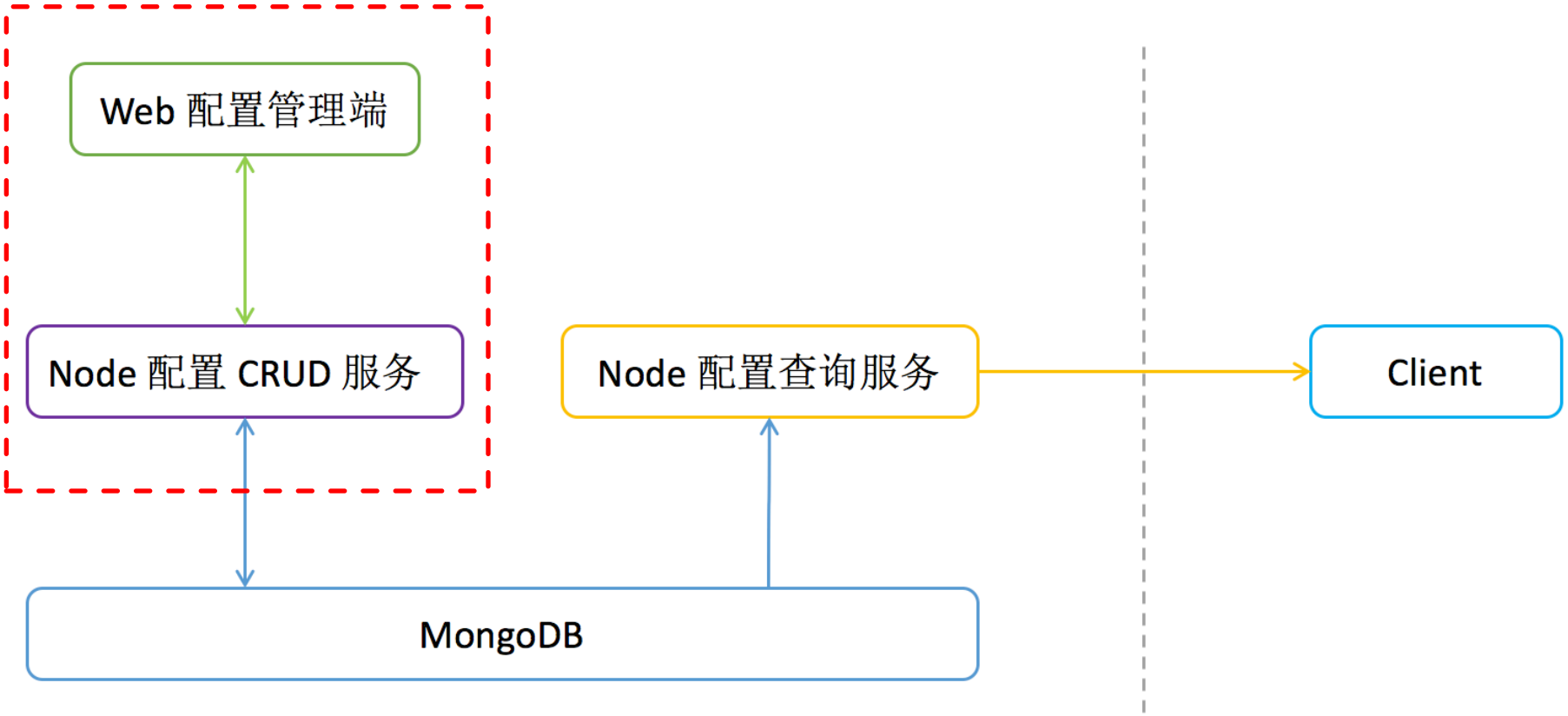
命名空间

贴合业务场景

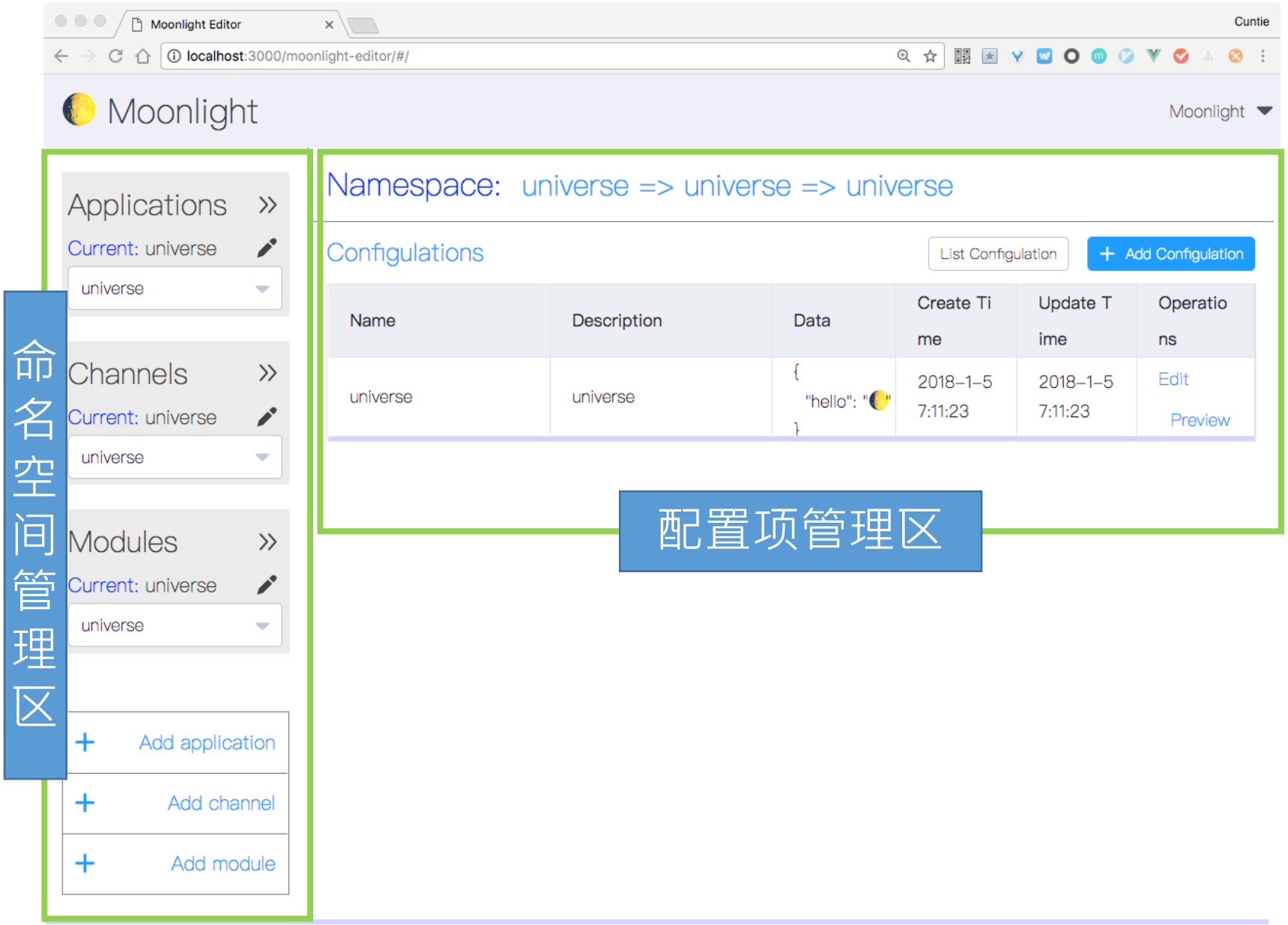
有利于配置项的归类和管理

减少配置项的命名冲突

Web 配置管理端



Web 配置管理端


Moonlight Editor
Cuntie
localhost:3000/moonlight-editor/#/
Moonlight
Moonlight
Applications
Current: universe
universe
Channels
Current: universe
universe
Modules
Current: universe
universe
Add application
Add channel
Add module
Namespace: universe => universe => universe
Configulations
List Configulation
Add Configulation
Name
Description
Data
Create Time
Update Time
Operations
universe
universe
{ "hello": "
}
2018-1-5 7:11:23
2018-1-5 7:11:23
Edit
Preview
命名空间管理区
配置项管理区

命名空间管理区

可视化的命名空间增删改查

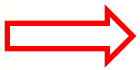
配置项管理区

可视化的配置项增删改查

JSON Schema

Vocabulary(词法) for annotate and validate JSON.

```
{
  title: "Configulation for Moonligh",
  type: "object"
  properties: {
    hello: {
      default: "🌓",
      description: "moonlight",
      type: "string"
    }
  },
}
```



```
{
  hello: "🌓"
}
```

JSON Schema

```
{
  title: "Configulation for Moonligh",
  type: "object"
  properties: {
    hello: {
      default: "🌒",
      description: "moonlight",
      type: "string"
    }
  },
}
```

```
{
  hello: 318
}
```

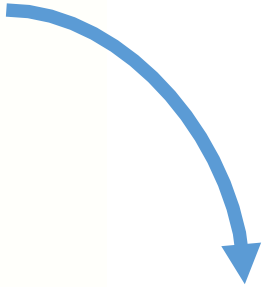
Eclipse: root.hello ==> Value must be of type string

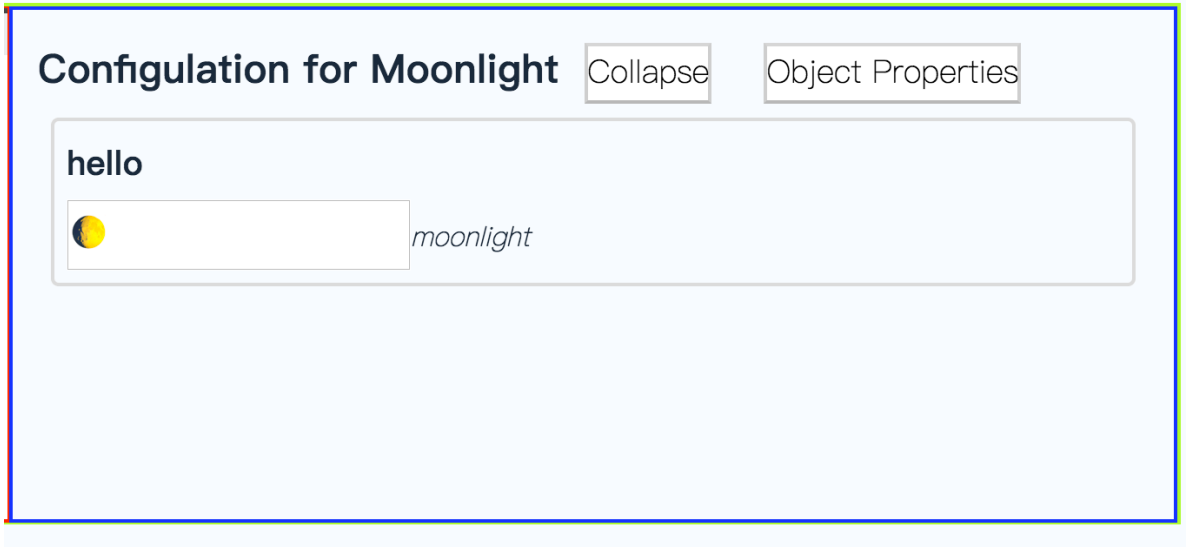
```
{
  hello: "🌒",
  world: "🌒"
}
```

Eclipse: root ==> No additional properties allowed, but property world is set

JSON Schema

```
{
  title: "Configulation for Moonligh",
  type: "object"
  properties: {
    hello: {
      default: "🌓",
      description: "moonlight",
      type: "string"
    }
  },
}
```





JSON Schema

开发人员编辑 Schema，加深对接口数据结构的理解

非开发人员编辑配置数据，使用 Schema 做数据校验

Schema 是一种语法，可以生成可编辑的表单

定时发布

指定未来时间对配置项做变更，秒级别的变更延迟

定时发布

定时发布列表

定时时间	数据	是否上线	提交日志	操作
2018-2-23 13:12:10	{ "hello": "🌗" }	true		删除
2018-2-24 11:13:7	{ "hello": "🌗" }	true		删除

Data

```
1 {
2   hello: "🌗"
3 }
```

Configulation for Moonlight Collapse Object Properties

hello

🌗 *moonlight*

确认 重置 代码和表单 只看代码 只看表单

Publish

Publish Status online

增加定时发布计划

版本化

记录配置项的变更历史,快速回退到指定版本

版本化

命名空间 universe/universe/unive

配置项名称 universe

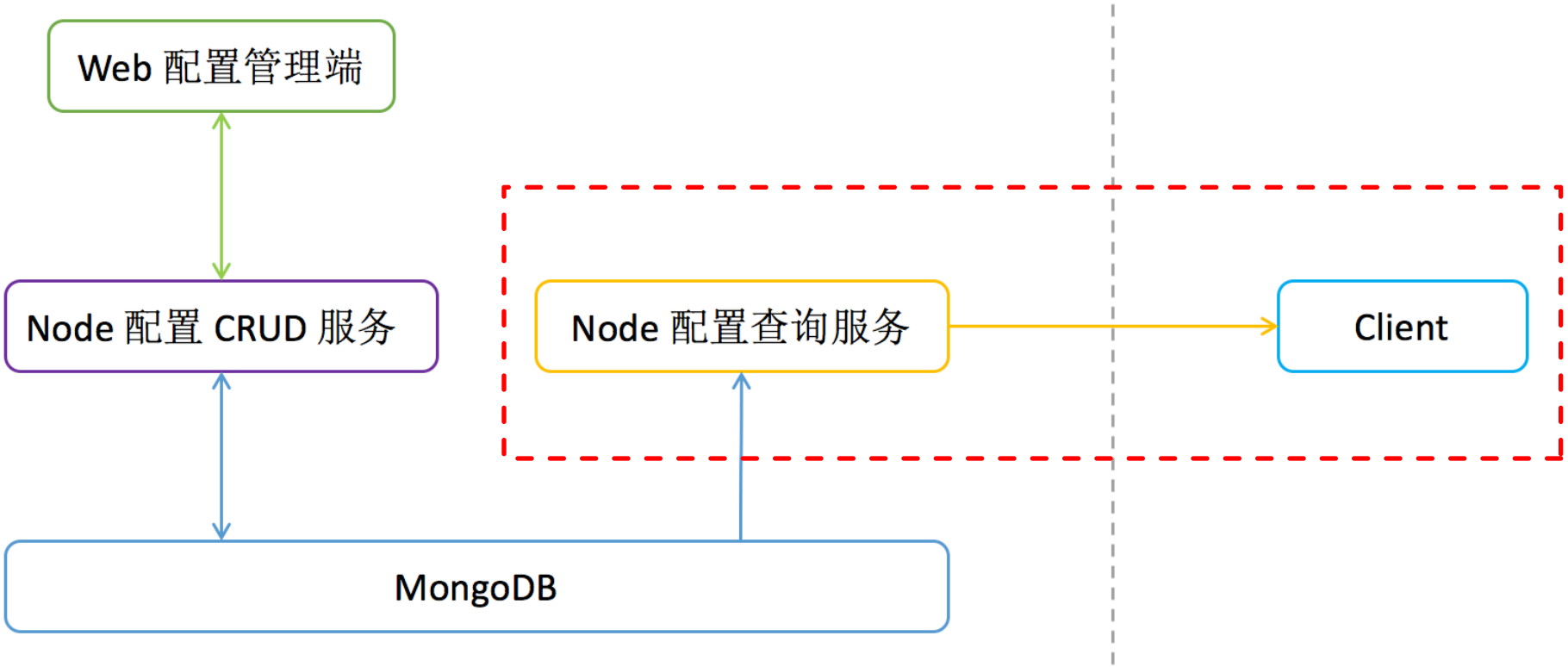
当前版本 4

回退到的版本 3

版本	数据	提交日志	提交时间	操作
0	{ "hello": "🌓" }	init	2018-1-5 7:11:23	对比 选中
1	{ "hello": "🌓" }	update	2018-1-10 19:2:19	对比 选中
2	{ "hello": "🌓" }	revert	2018-1-10 19:3:43	对比 选中
3	{ "hello": "🌓" }	schedule	2018-1-26 7:13:24	对比 选中
4	{ "hello": "🌓" }	revert	2018-2-6 11:57:25	对比 选中

回退

客户端接入



只要一个JSON

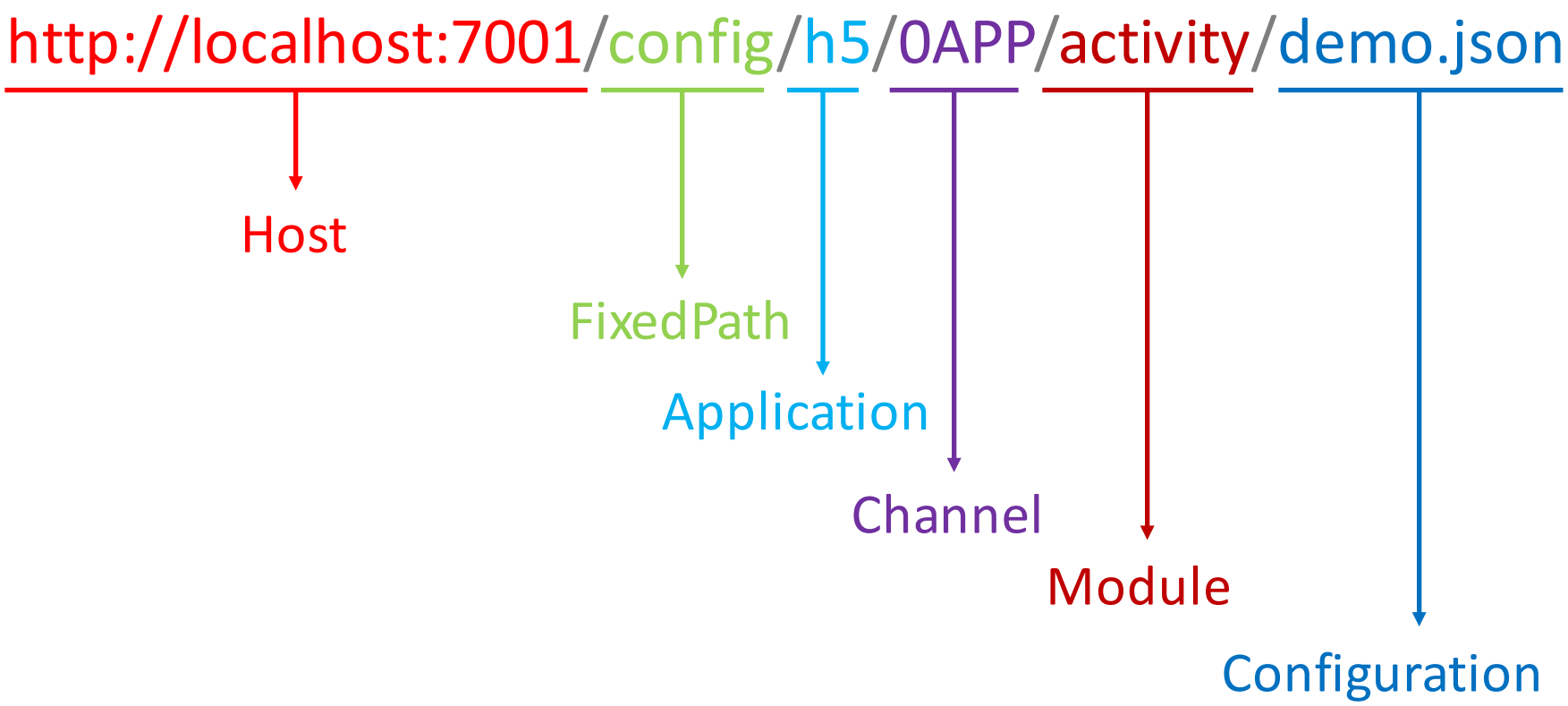
```
{
  backgroundColor: "gray",
  title: "Hello Moonlight"
}
```

```
{
  data: {
    title: "Hello Moonlight",
    backgroundColor: "gray"
  },
  ret: "0",
  errCode: "",
  errMsg: ""
}
```

只有一个接口

http://localhost:7001/config/h5/0APP/activity/demo.json

只有一个接口


http://localhost:7001/config/h5/0APP/activity/demo.json
Host
FixedPath
Application
Channel
Module
Configuration

系统特性

统一管理不同应用，不同模块的配置

配置项数据字段灵活，配置变更稳健

配置支持版本化管理和定时更新

简单易用

适用场景举例

活动页面链接替换

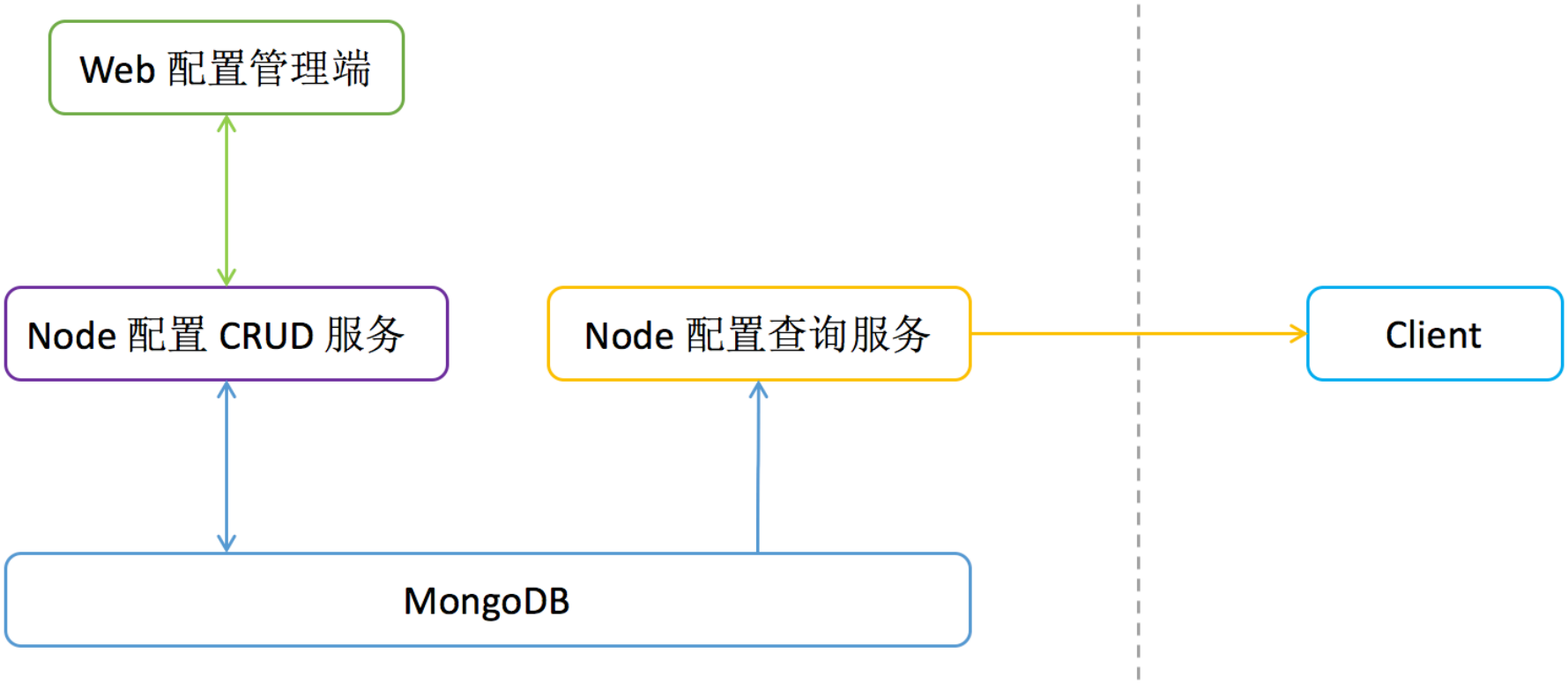
帮助中心文本

功能特性开关

所有由前端写死，又有独立修改需求的数据

How

系统架构



源码划分

moonlight-mongodb

moonlight-node-server

moonlight-editor

技术栈

MongoDB
&
Docker

Node.js
&
Egg.js

Vue.js
&
Element.js

Moonlight-mongodb

docker 运行 mongoDB, 将数据库数据保存到 host 目录.

Moonlight-mongodb

一行命令初始化数据库

```
$ docker-compose -f docker-compose.dev.yml up --build
```

一行命令重置数据库

```
$ rm -rf ./db && git checkout ./db
```

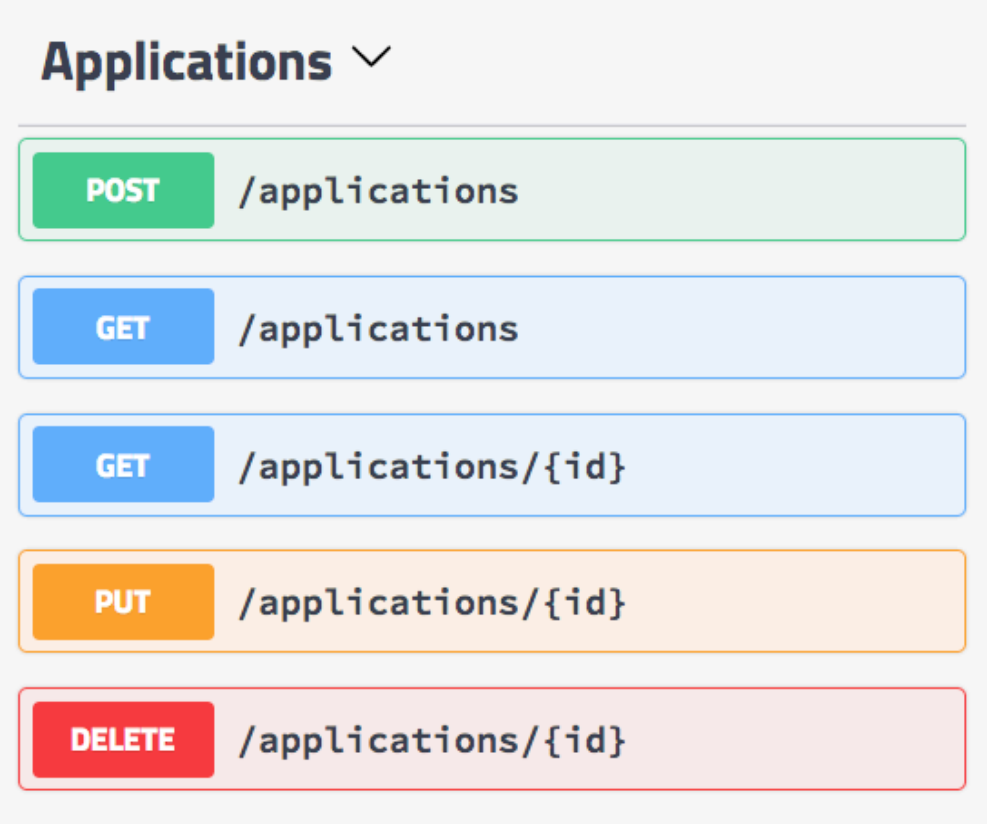
Moonlight-node-server

连接 MongoDB 数据库, 提供操作数据库的 WEB 接口.

Mongoose

```
  const applicationSchema = new Schema({
    name: { type: String, unique: true, required: true },
    description: String,
    createTime: { type: Date, default: Date.now },
    updateTime: { type: Date, default: Date.now },
  });

  return mongoose.model('Application', applicationSchema);
```

RESTful


Applications
POST
/applications
GET
/applications
GET
/applications/{id}
PUT
/applications/{id}
DELETE
/applications/{id}

接口文档

“后台和前端同一个人做的项目，基本没有文档.”

接口文档

```
/**
 * @swagger
 * /applications/{id}:
 *   get:
 *     tags:
 *       - Applications
 *     description: return an application match the query conditions
 *     parameters:
 *       - name: id
 *         in: path
 *         description: query conditions -- name
 *         type: string
 *         required: true
 *     responses:
 *       200:
 *         description: an application
 *         schema:
 *           $ref: '#/definitions/Application'
 */
```

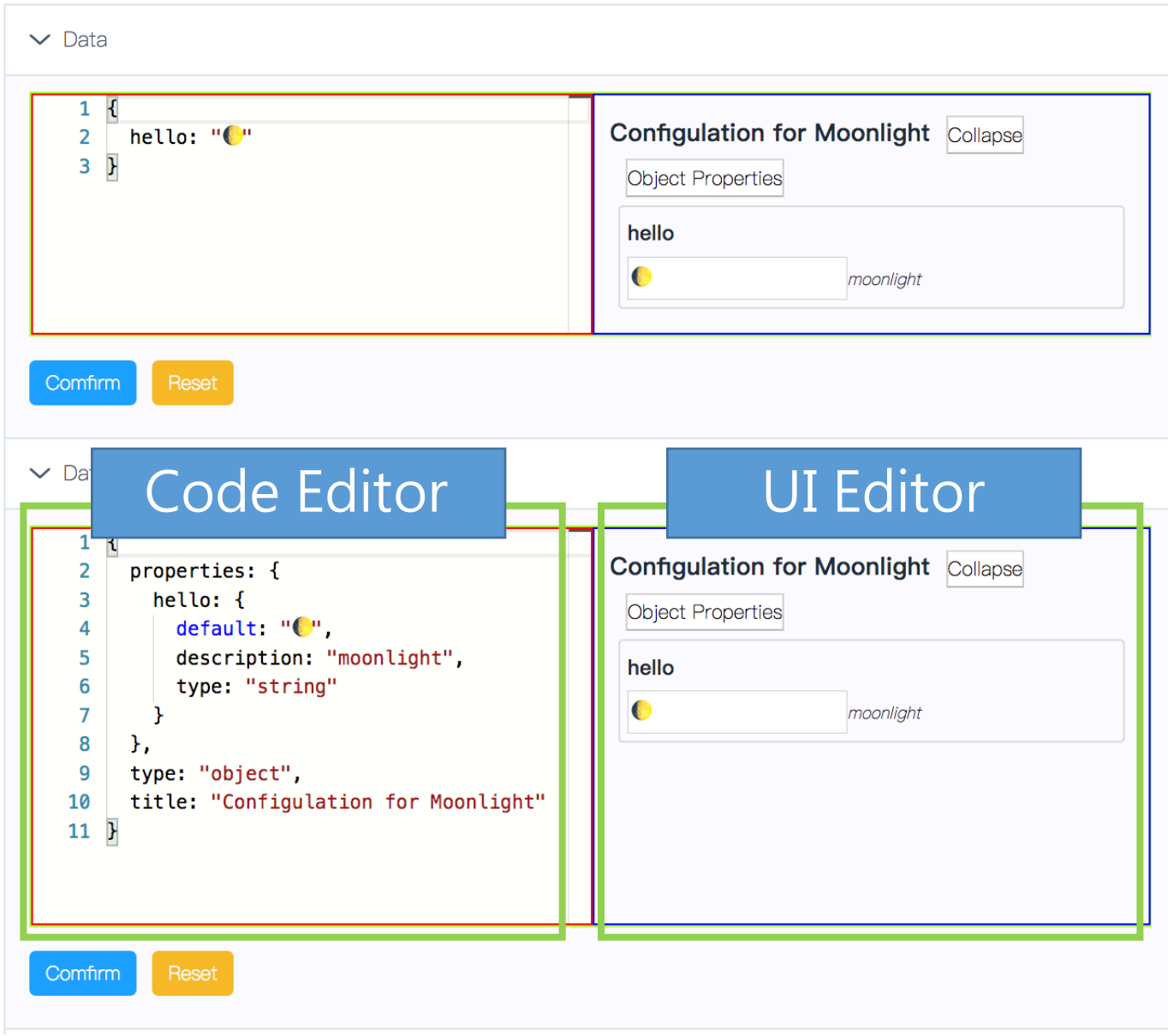
单元测试

```
DBTest for Configulation
  ✓ createConfigulation
  ✓ queryConfigulation
  ✓ queryManyConfigulations
  ✓ updateConfigulation
  ✓ deleteConfigulation
  ✓ deleteManyConfigulations
  ✓ queryConfigulationById
  ✓ updateConfigulationById
  ✓ deleteConfigulationById

DBTest for Module
  ✓ createModule
  ✓ queryModule
  ✓ queryManyModules
  ✓ updateModule
  ✓ deleteModule
  ✓ deleteManyModules


51 passing (979ms)
```

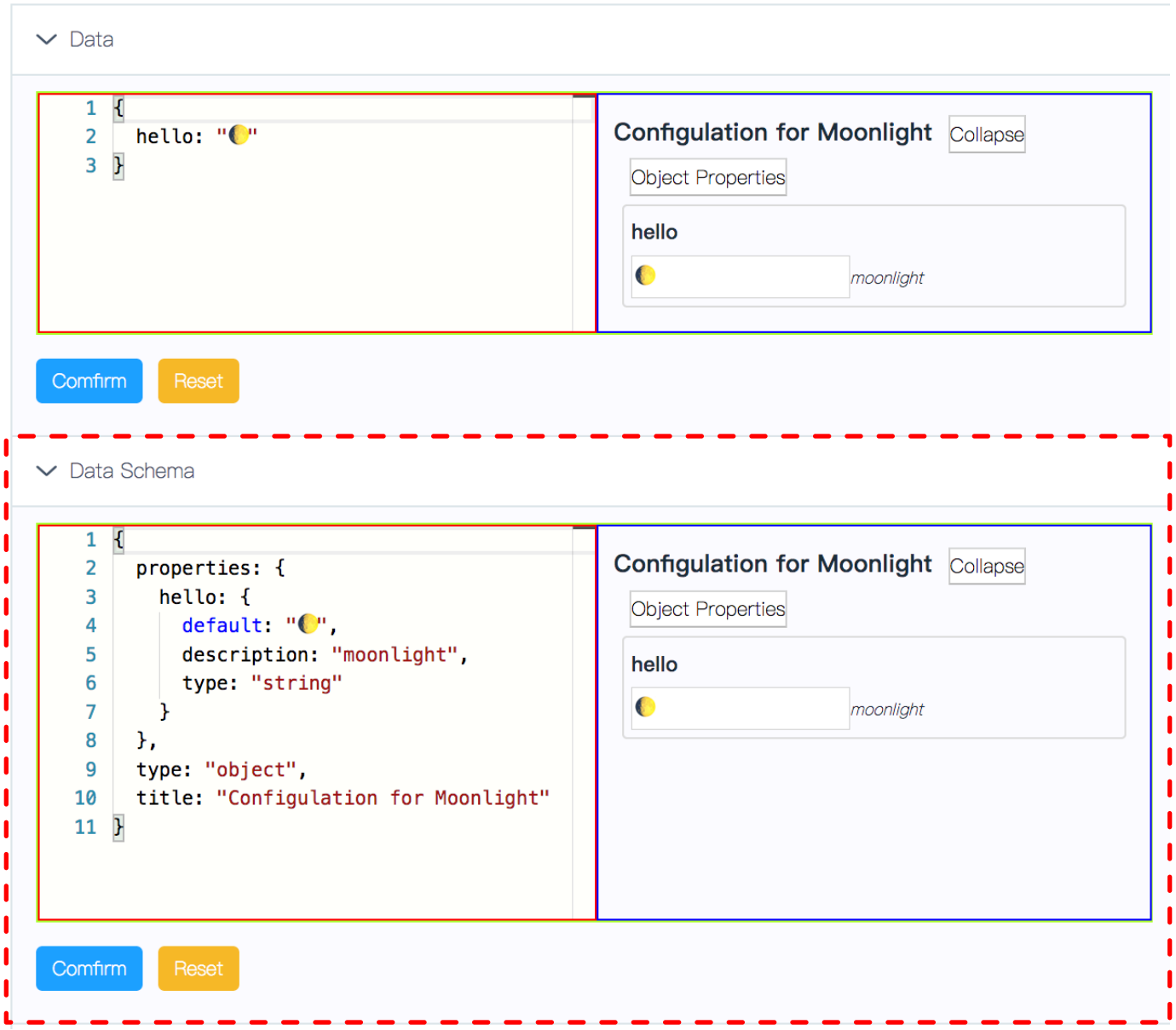
Moonlight-editor

可视化的配置项 CRUD 操作.

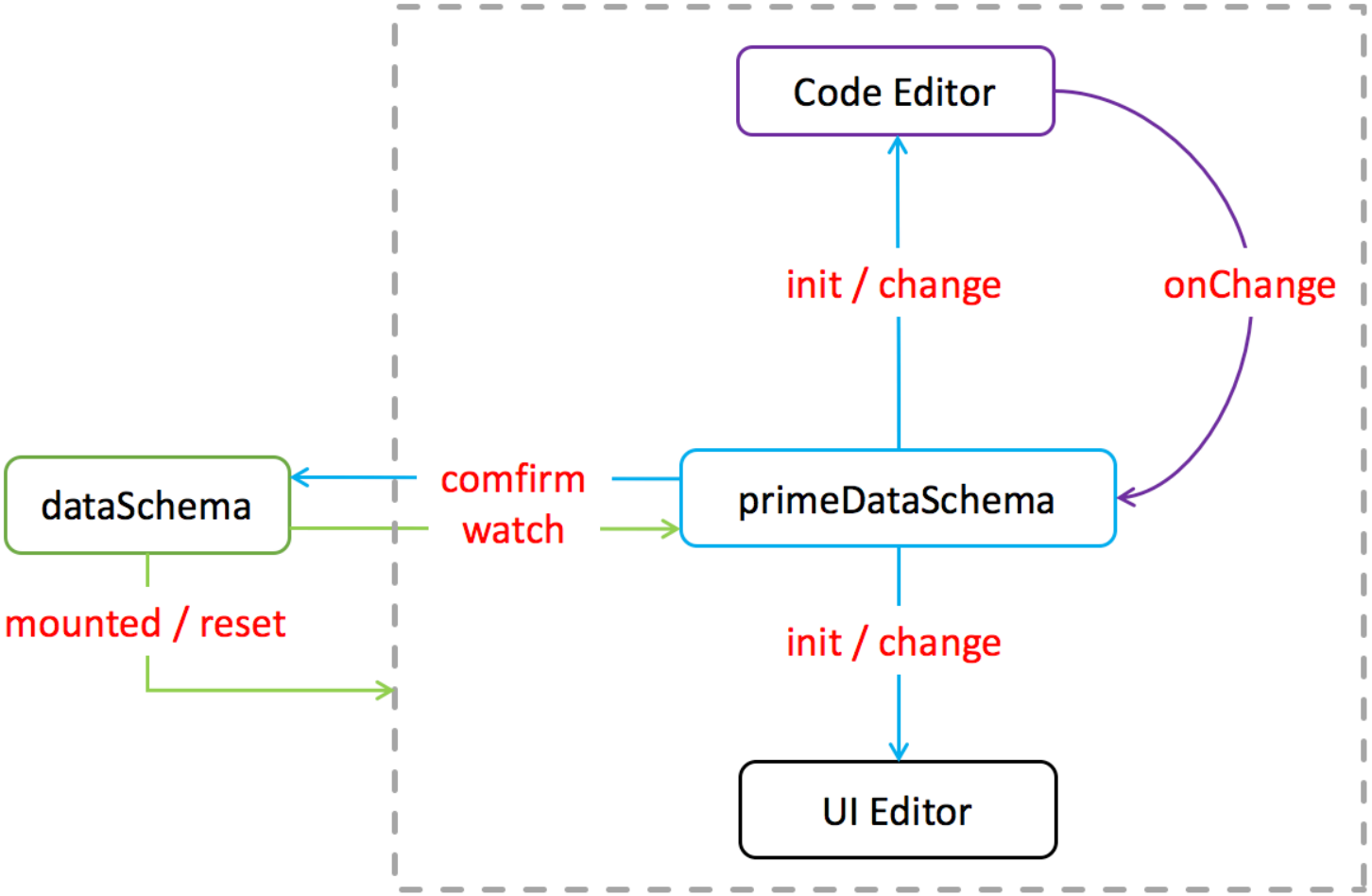
Moonlight-editor



Moonlight-editor

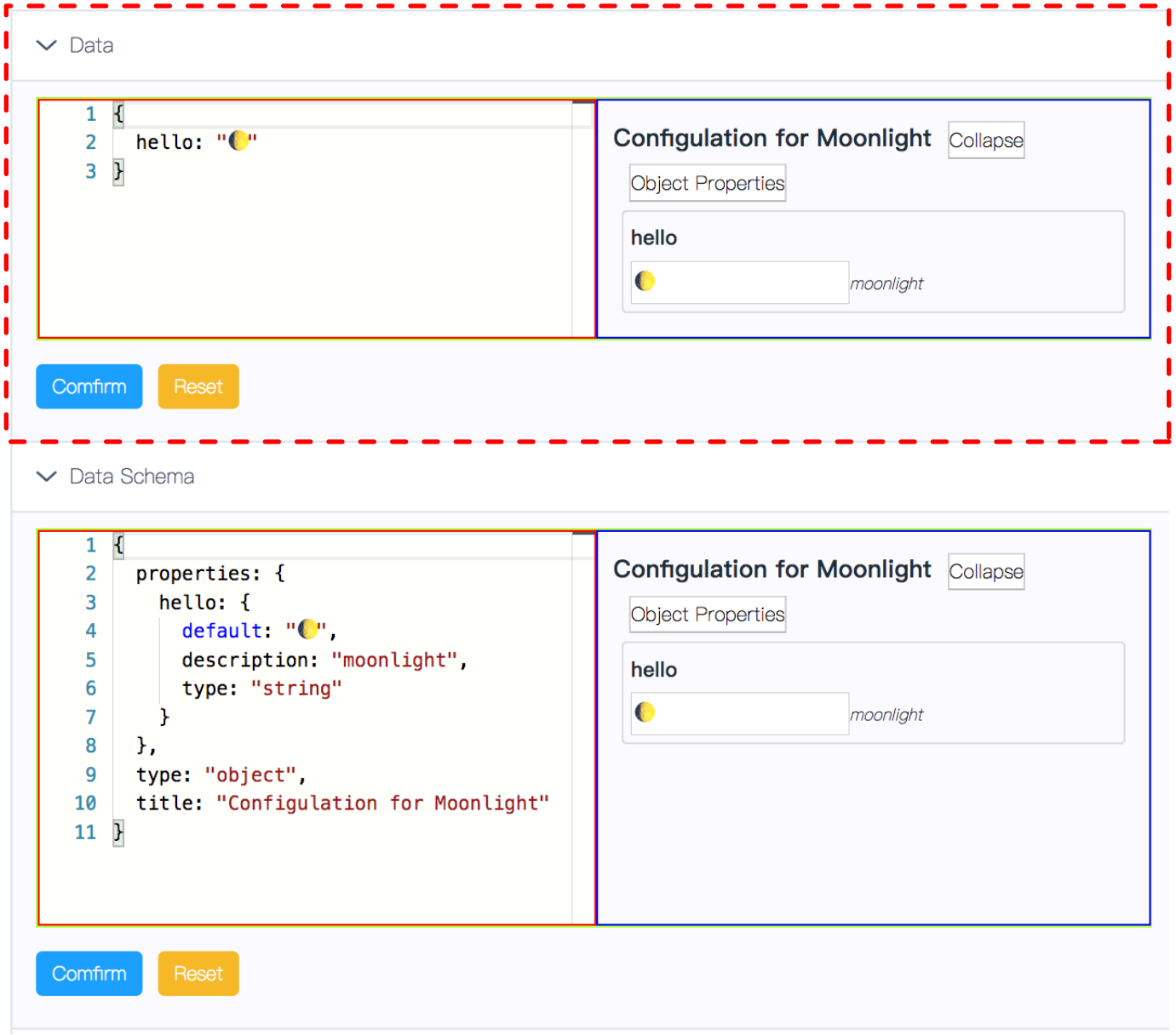


Moonlight-editor

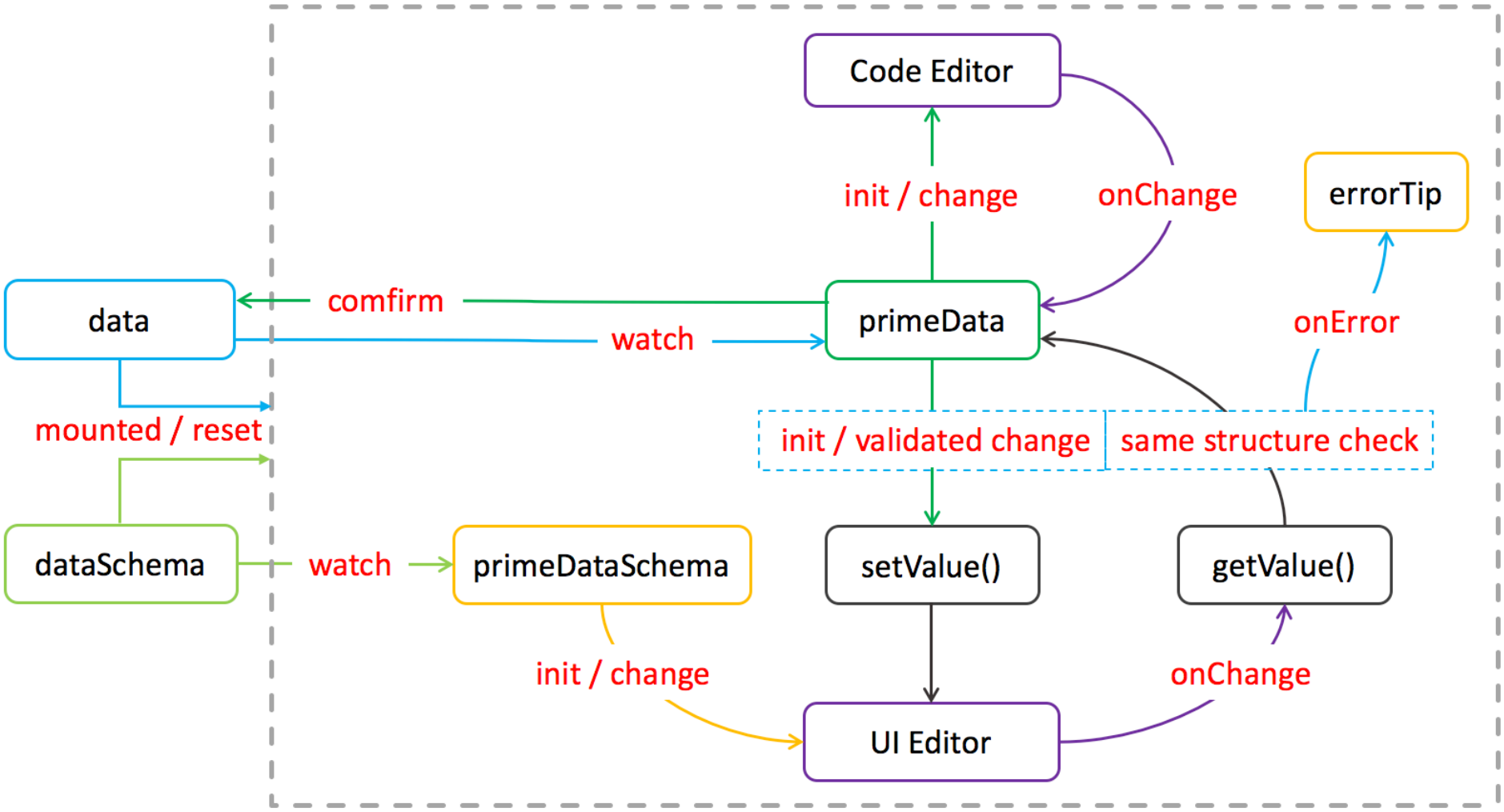


Schema Editor 组件

Moonlight-editor



Moonlight-editor



JSON Editor 组件

上线上线

可用性保障

可用性保障

配置是前端业务初始化和运行的依赖

新上线的系统

MongoDB + Node 新服务架构

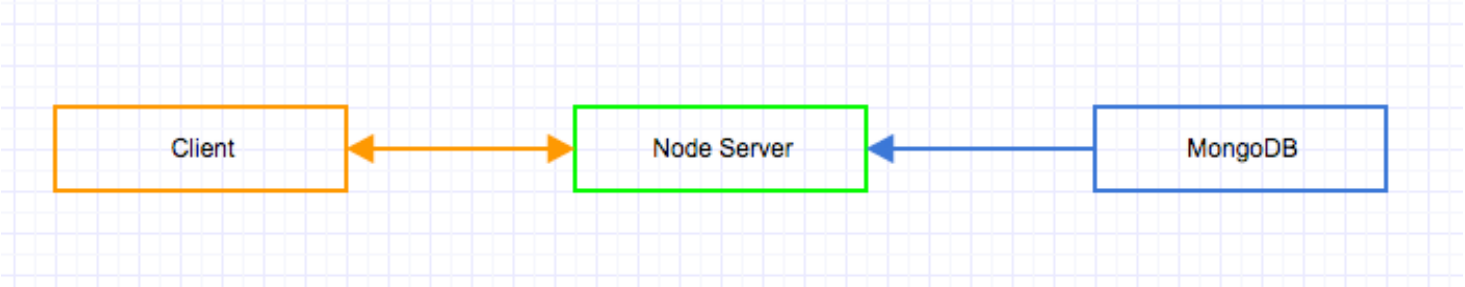
不可用场景

服务器性能占满

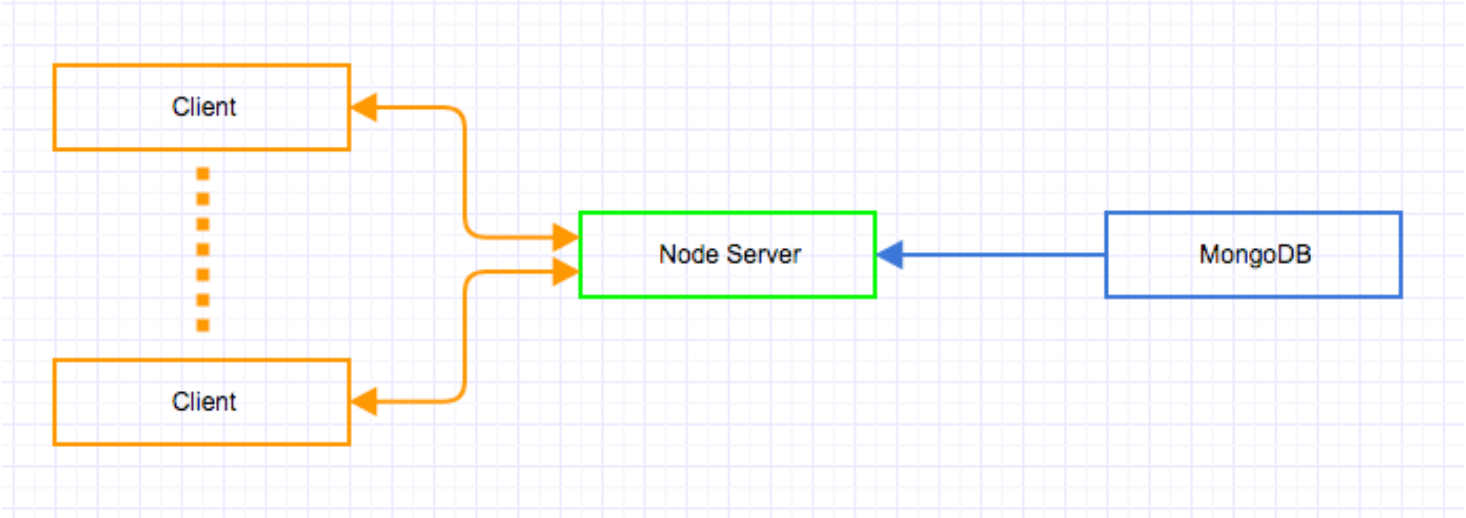
服务器物理失效/宕机

服务器代码逻辑缺陷

可用性保障



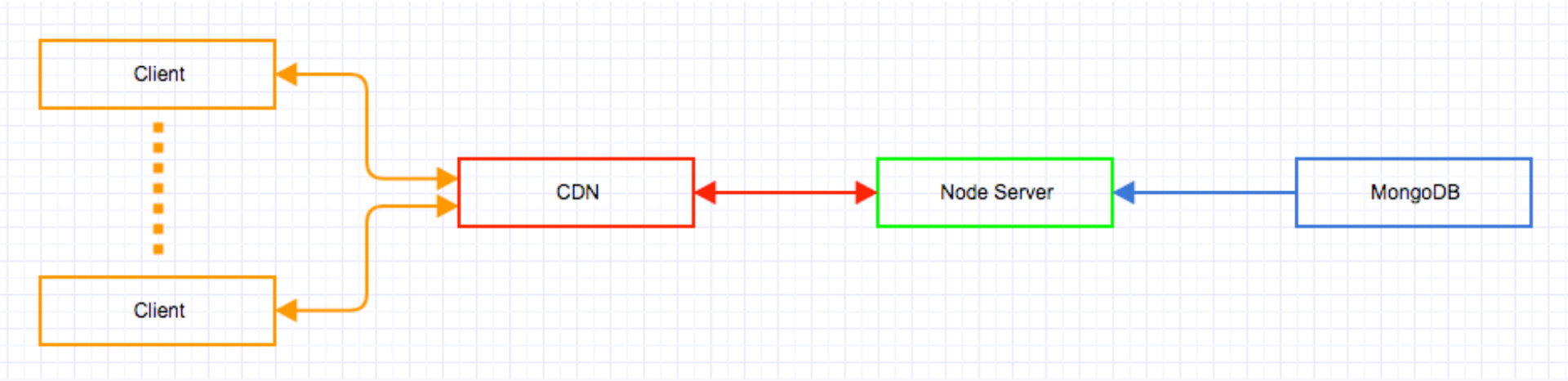
客户端直接连接



用户汹涌澎湃而来

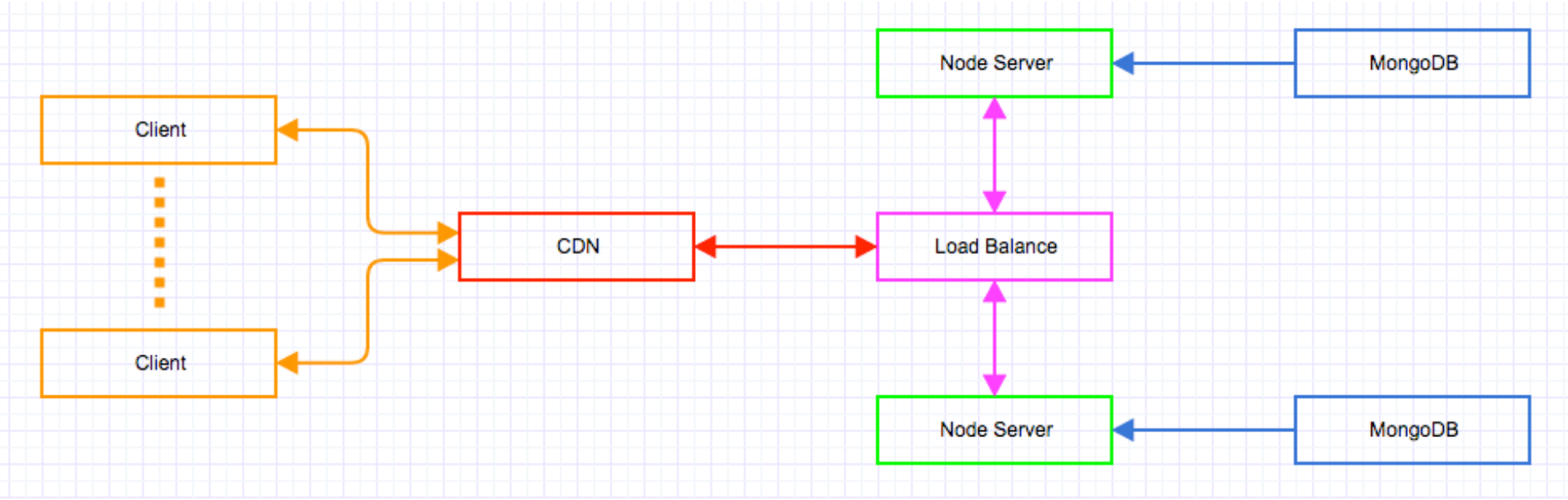
可用性保障一: CDN 缓存

http://localhost:7001/config/h5/0APP/activity/demo.json



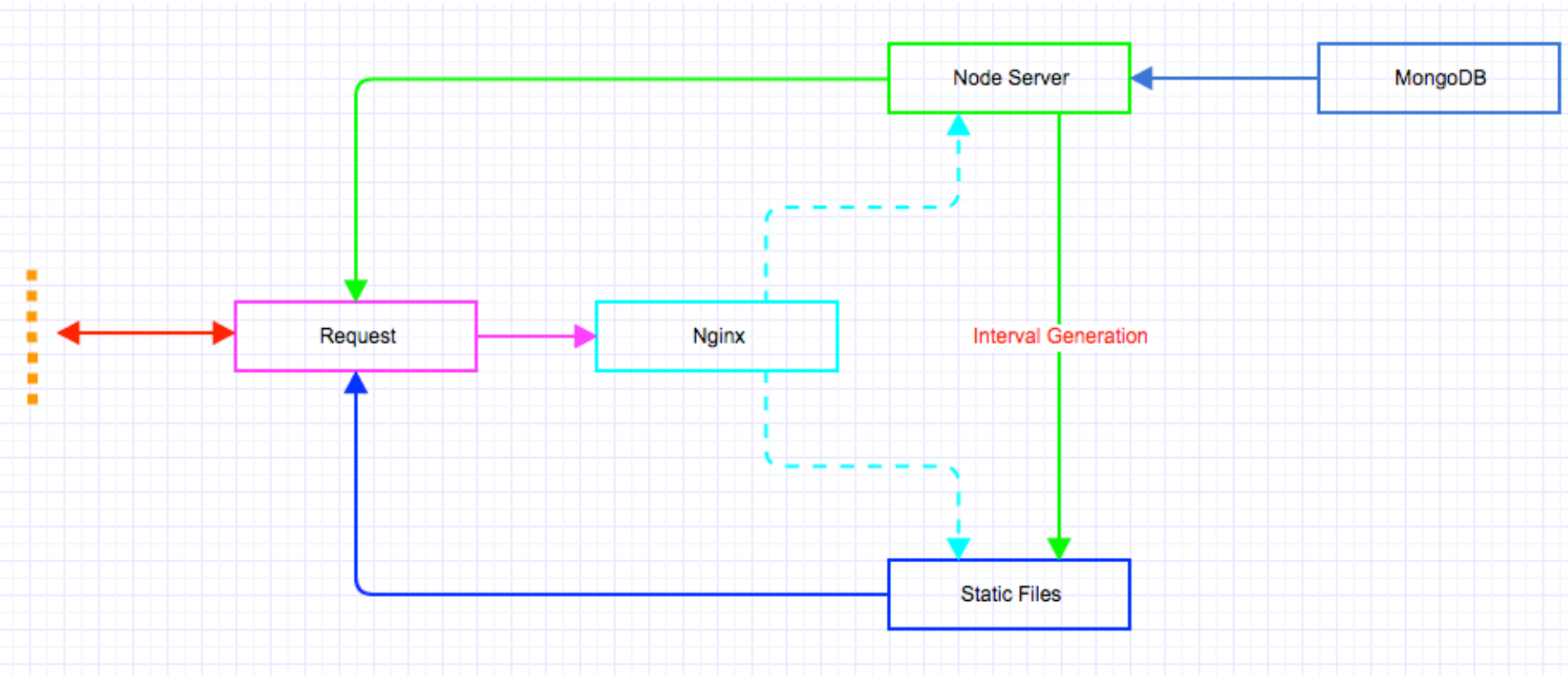
CDN缓存减少服务器负担

可用性保障二: 多服务器负载均衡



多服务器负载均衡

可用性保障三: Nginx Failback

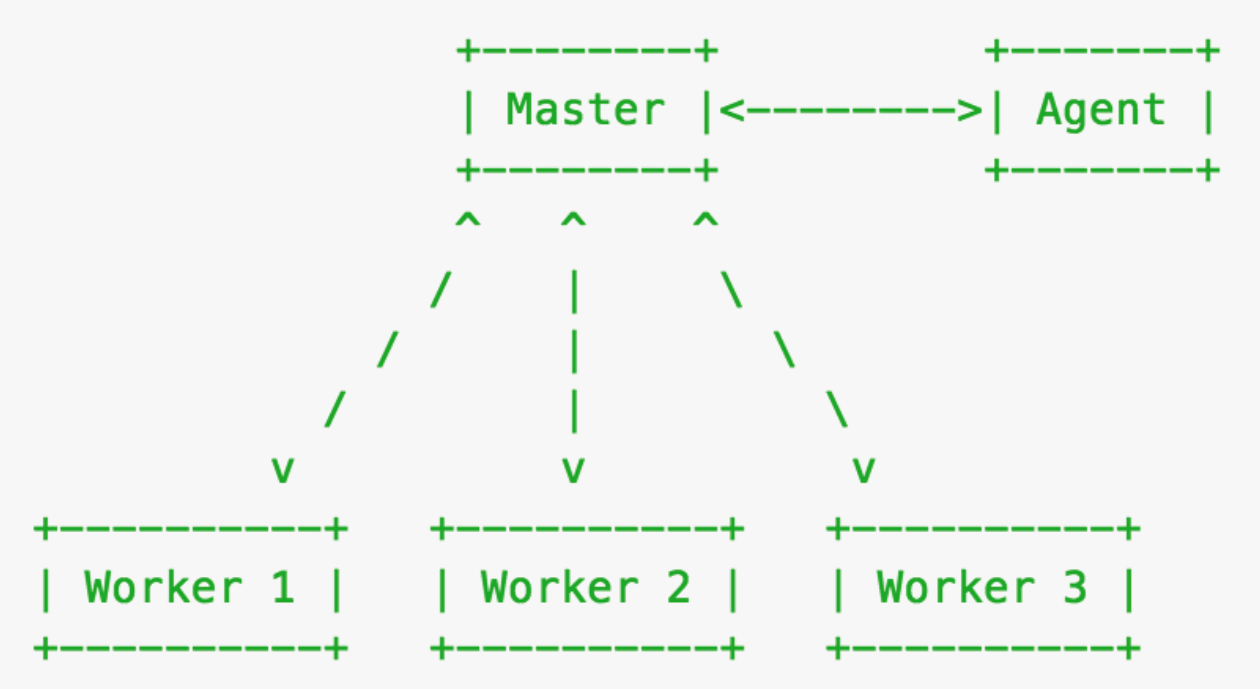


Nginx 静态资源回退保障

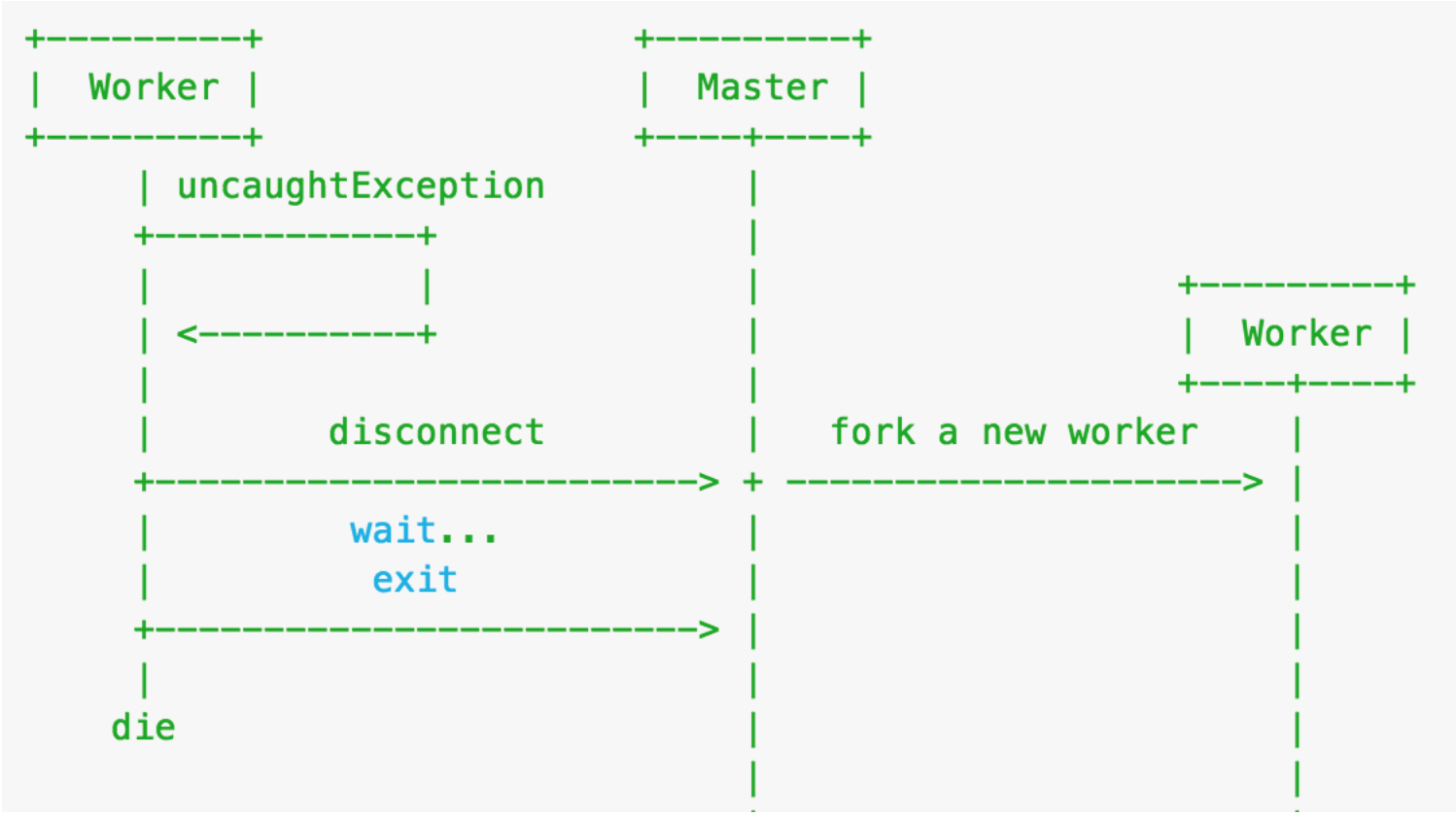
可用性保障三：Nginx Failback

```
server {
  listen 80 default_server;

  location /moonlight-node-server/ {
    proxy_pass http://127.0.0.1:7001/;

    proxy_intercept_errors on;
    error_page  500 502 503 504 402 403 401 = @moonlight-node-server-fallback;
  }

  location @moonlight-node-server-fallback {
    rewrite ^(.*)moonlight-node-server/(.*)$ $1$2 break;
    root /Users/cntchen/dev/moonlight/moonlight-node-server/dist/;
    try_files $uri $uri/ =404;
  }
}
```

可用性保障四: Egg.js 多进程模型



可用性保障四: Egg.js 多进程模型



可用性保障五: 客户端默认配置

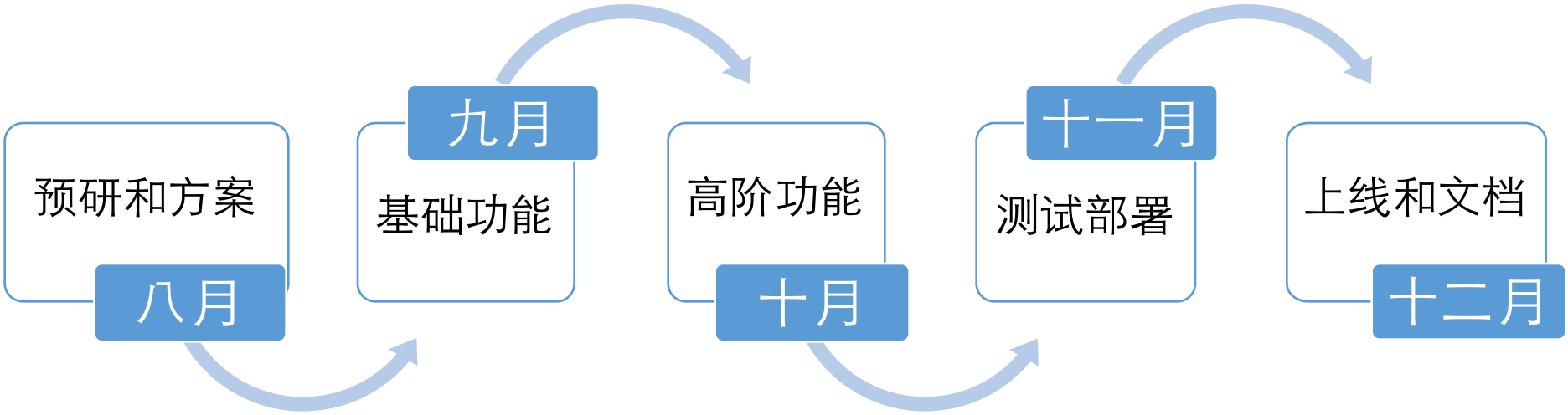
```
const config = await getConfig({
  url: `http://${window.location.hostname}`
    + ':7001/config/h5/0APP/activity/demo.json',
  defaultConfig: {
    title: 'Default Title',
    backgroundColor: 'lightblue',
  },
  timeout: 5000,
});
```

客户端默任配置和错误上报

可用性保障

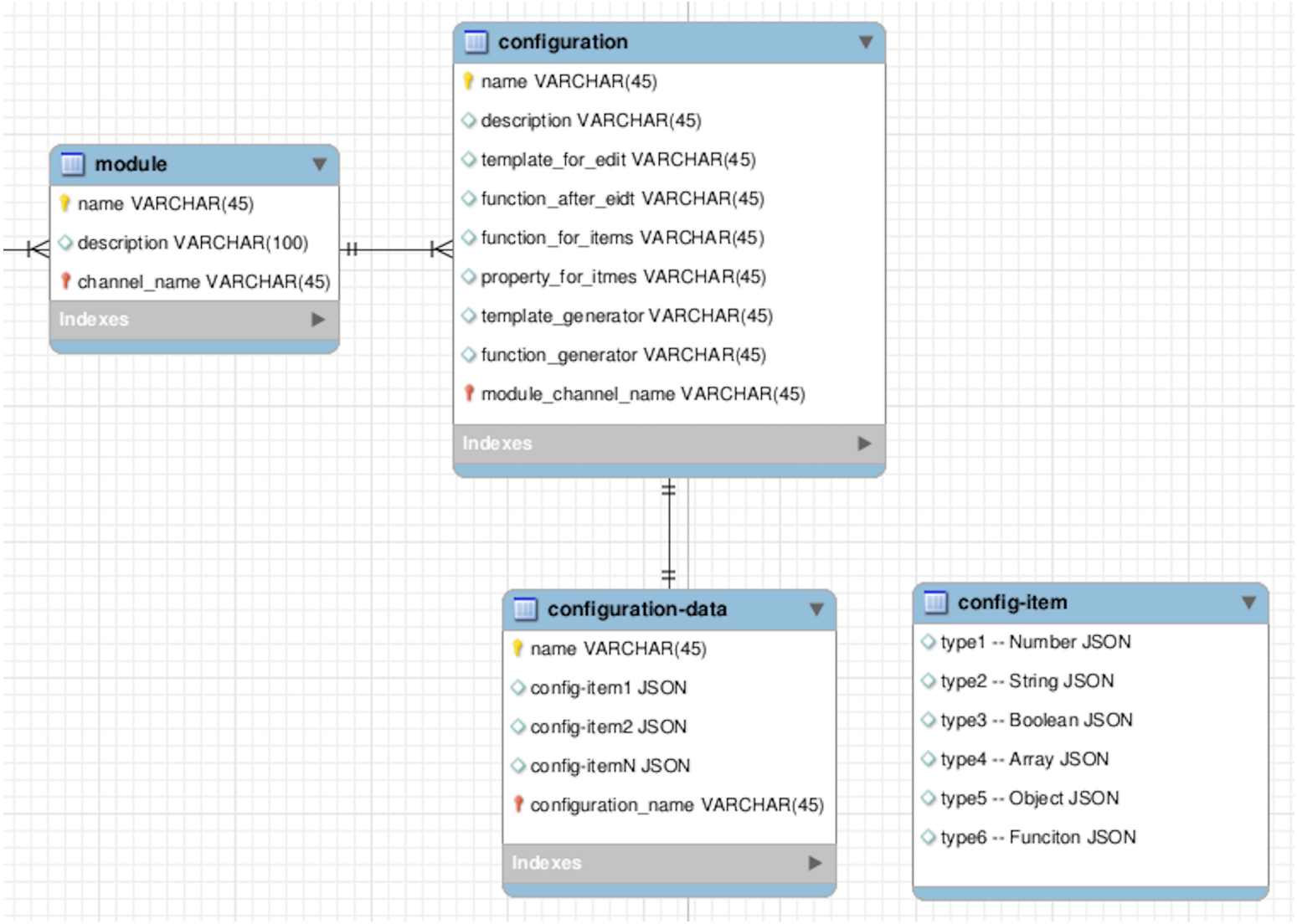
开发总结

开发历程

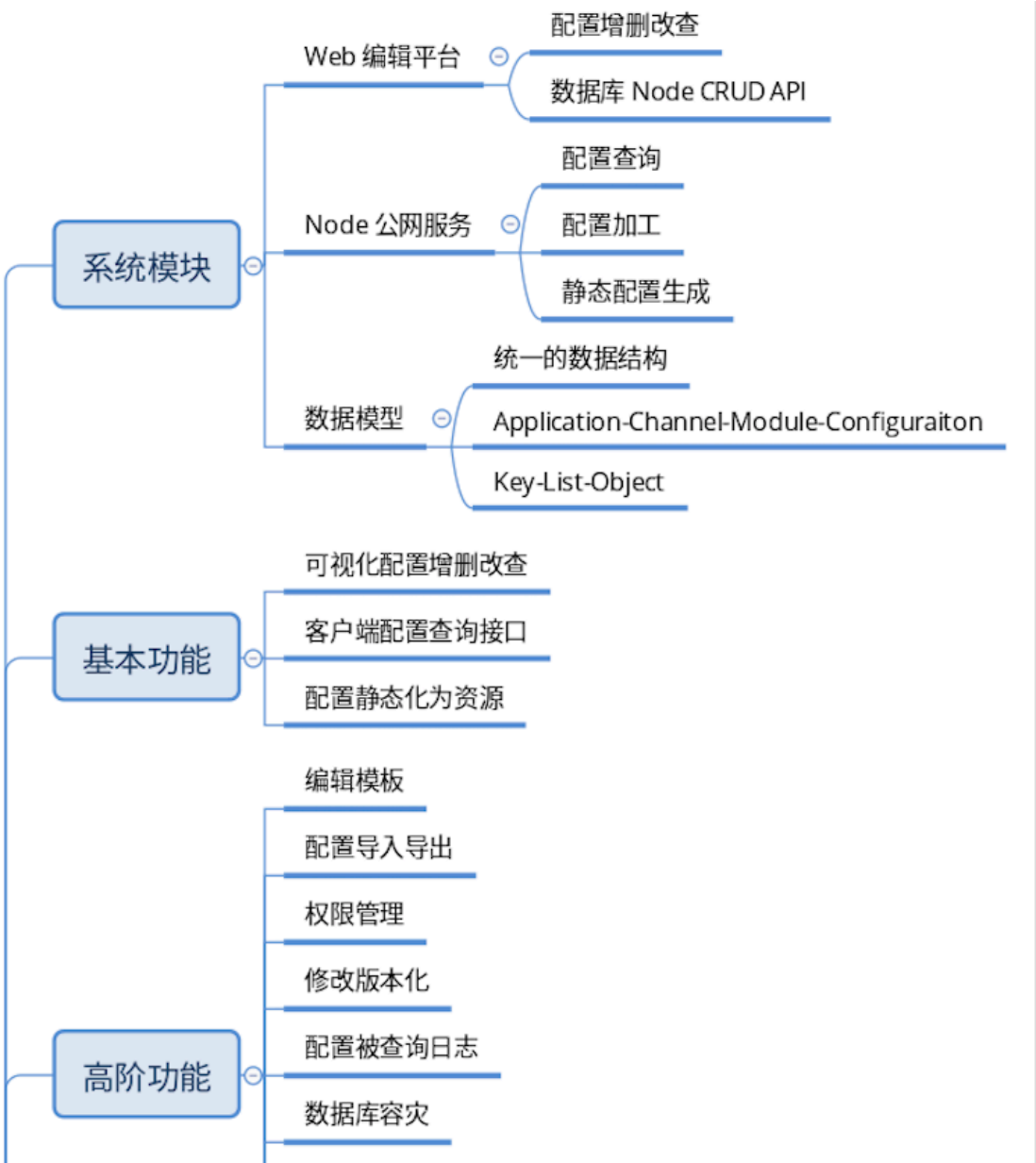


计划先行

计划先行


configuration
name VARCHAR(45)
description VARCHAR(45)
template_for_edit VARCHAR(45)
function_after_eidt VARCHAR(45)
function_for_items VARCHAR(45)
property_for_itmes VARCHAR(45)
template_generator VARCHAR(45)
function_generator VARCHAR(45)
module_channel_name VARCHAR(45)
Indexes
module
name VARCHAR(45)
description VARCHAR(100)
channel_name VARCHAR(45)
Indexes
configuration-data
name VARCHAR(45)
config-item1 JSON
config-item2 JSON
config-itemN JSON
configuration_name VARCHAR(45)
Indexes
config-item
type1 -- Number JSON
type2 -- String JSON
type3 -- Boolean JSON
type4 -- Array JSON
type5 -- Object JSON
type6 -- Funciton JSON

计划先行


系统模块
Web 编辑平台
配置增删改查
数据库 Node CRUD API
Node 公网服务
配置查询
配置加工
静态配置生成
数据模型
统一的数据结构
Application-Channel-Module-Configuraiton
Key-List-Object
基本功能
可视化配置增删改查
客户端配置查询接口
配置静态化为资源
高阶功能
编辑模板
配置导入导出
权限管理
修改版本化
配置被查询日志
数据库容灾

计划先行


MoonLight
Help
CntChen
Application
当前: 招联金融
招联金融
金融云平台
招联商城
All
Channel
当前: H5
Android
iOS
H5
All
Module
当前: Activity
Activity
Help Desk
Hack Tec
All
Add Application
Add Channel
Add Module
招联金融 => H5页面 => 活动中心
.....
提交修改
回滚
提交历史
模块管理
配置项列表
搜索配置项
.....
Add Configulartion
发布状态
生效时间
Key
Value
描述
最后修改人
最后修改时间
被调用情况
Operation
[已发布] 已生效 100小时 Activity1 {isOpen: true} 活动配置 Cntchen 1970-01-01 今天100次
[已下线] 已生效 100天 Activity2 {who: "CntChen" } 活动配置 Cntchen 1970-01-01 10天内没调用
[已发布] 将于 10.10 生效 Activity3 {url: "www.github.com"} 活动配置 Cntchen 1970-01-01 App 渠道
[已发布] 10小时后生效 Activity4 Column 2 活动配置 Cntchen 1970-01-01 无调用
[已下线] 已生效 100天 Activity4 Column 2 活动配置 Cntchen 1970-01-01 无调用
[已下线] 已生效 100天 Activity4 Column 2 活动配置 Cntchen 1970-01-01 无调用

快速迭代

M **BSS / moonlight-editor**
前端业务配置管理系统 moonlight 的 Web 管理网站

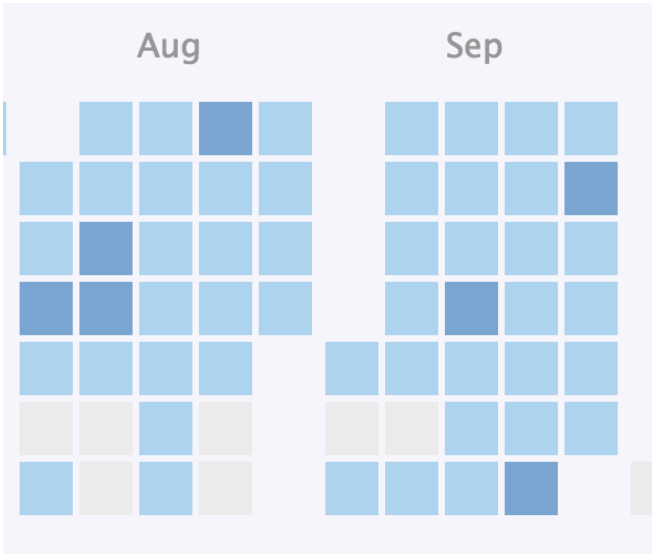
M **BSS / moonlight-node-server**
前端业务配置管理系统 moonlight 的 Node 服务器

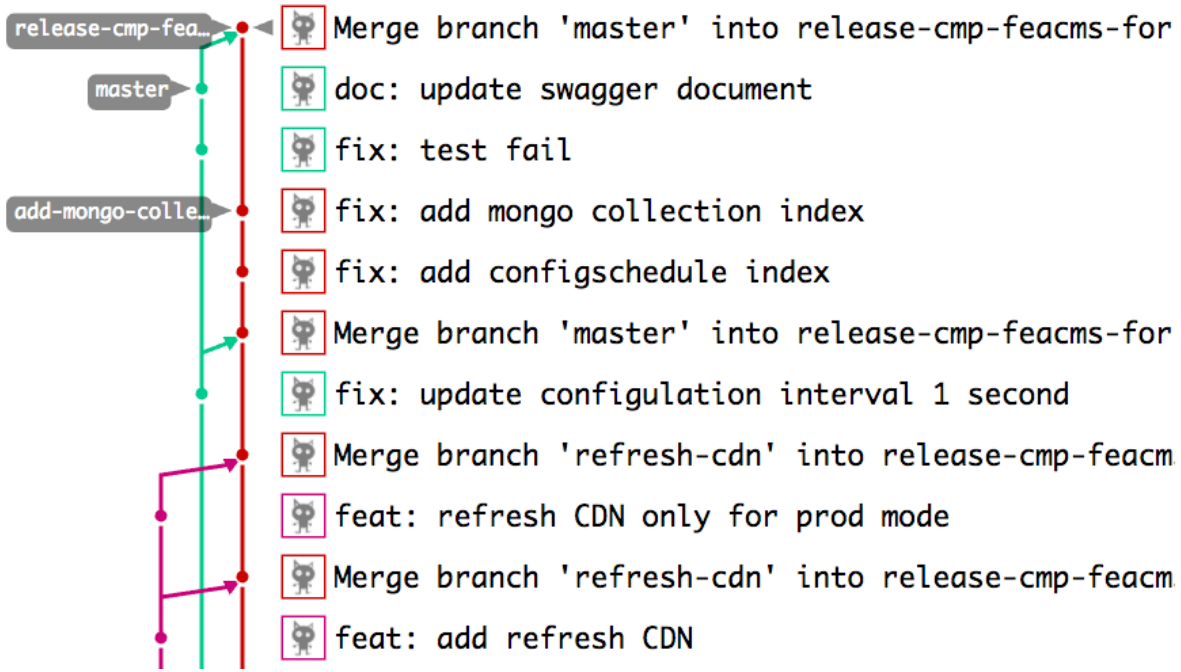
M **BSS / moonlight-mongodb**
前端业务配置管理系统 moonlight 的 mongoDB 数据库

M **BSS / moonlight-plan**
前端业务配置管理系统 moonlight 的开发文档和开发计划

Project	Commits	Branchs
moonlight-plan	30	1
moonlight-node-server	144	6
moonlight-editor	130	3
moonlight-mongodb	19	1

快速迭代


Aug
Sep


release-cmp-fea…
master
add-mongo-colle…
Merge branch 'master' into release-cmp-feacms-for
doc: update swagger document
fix: test fail
fix: add mongo collection index
fix: add configschedule index
Merge branch 'master' into release-cmp-feacms-for
fix: update configulation interval 1 second
Merge branch 'refresh-cdn' into release-cmp-feacm
feat: refresh CDN only for prod mode
Merge branch 'refresh-cdn' into release-cmp-feacm
feat: add refresh CDN

多写文档

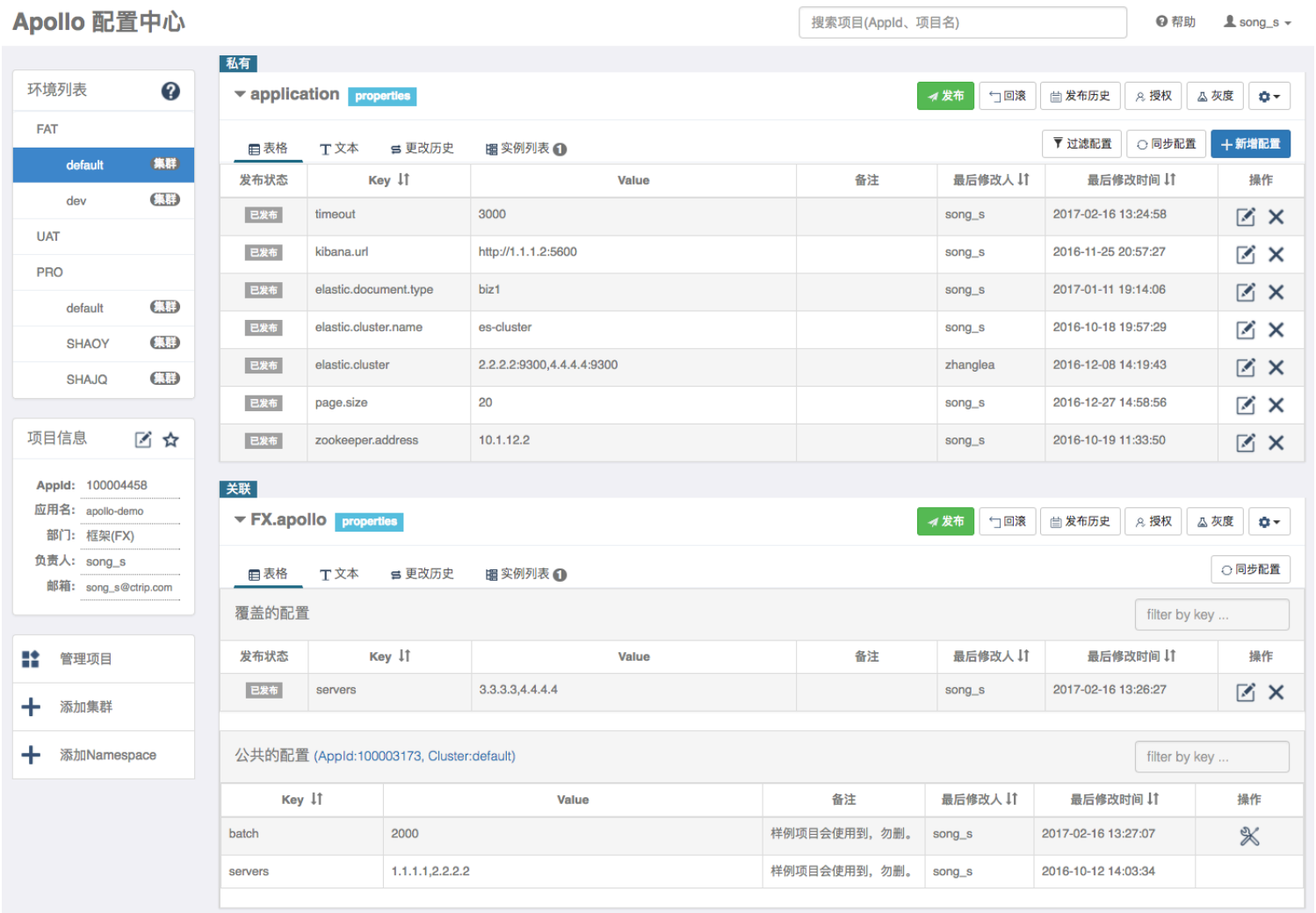
每个项目都有 README

关键技术的选型和引入方式都有文档

部署和运维有可重放的操作文档

Moonlight ?

Moonlight from Apollo



https://github.com/ctripcorp/apollo 3k+ stars

Moonlight from Apollo

Apollo

build passing

```
We choose to go to the moon in this decade and do the other things,
not because they are easy, but because they are hard.
-- John F. Kennedy, 1962
```

Welcome to the Apollo GitHub.

Apollo is an open autonomous driving platform. It is a high performance flexible archite
driving capabilities. For business contact, please visit http://apollo.auto

https://github.com/ApolloAuto/apollo 7k+ stars

参考文献

- 动态调整的基础 —— 配置中心(阿里天猫)
> http://www.infoq.com/cn/articles/the-foundation-of-dynamic-adjustment-configuration-center

- 一篇好TM长的关于配置中心的文章(阿里中间件)
> http://jm.taobao.org/2016/09/28/an-article-about-config-center/

- Apollo配置中心设计(携程)
> https://github.com/ctripcorp/apollo/wiki/Apollo配置中心设计

- 前端配置管理系统技术方案
> http://git.mucfc.com/BSS/moonlight-plan

- 前端配置管理系统操作文档
> http://git.mucfc.com/BSS/moonlight-editor/wikis/home

依赖的开源项目

- Vue.js
- Element
- Json-editor
- Monaco-editor
- JSON Schema
- Egg.js
- Mongoose
- Swagger-jsdoc

Q & A