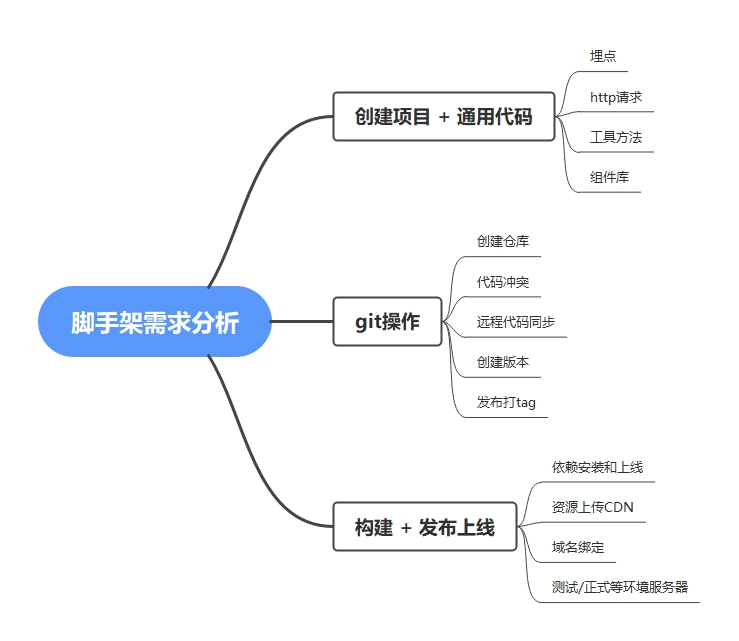
# 痛点分析

* 团队规模足够大的项目规范把控。
* 创建项目/组件时，存在大量代码拷贝，容易导致错漏，导致效率问题。
* 项目模板不统一，构建工具、lint、开发工具配置、ui库、git钩子等等不一致，导致项目管理混乱，项目之间切换对于易踩坑的点容易重复陷入，不同的构建工具不同的库导致学习成本增加，而花时间学习这些并非提升。
* 协同开发时，由于git操作不规范，导致分支混乱，操作耗时。
* 发布上线耗时，容易出现各种错误：制定标准的上线流程规范并集成到脚手架。
* 线上项目报错无法定位

# 需求分析

* 通用的研发脚手架。
* 通用项目/组件创建能力。
* 模板支持定制，定制后能够快速生效。
* 模板支持快速接入，极低的接入成本。
* 集成通用的工具集，http、lint、commit钩子、错误收集、性能分析等。
* 通用组件发布能力。
* 发布过程自动完成标准的git操作。
* 发布成工后自动删除分支并打上tag。
* 发布后自动完成云构建、OSS上传、CDN上传、域名绑定。
* 发布过程支持测试、正式等环境。

## 需求分析图示

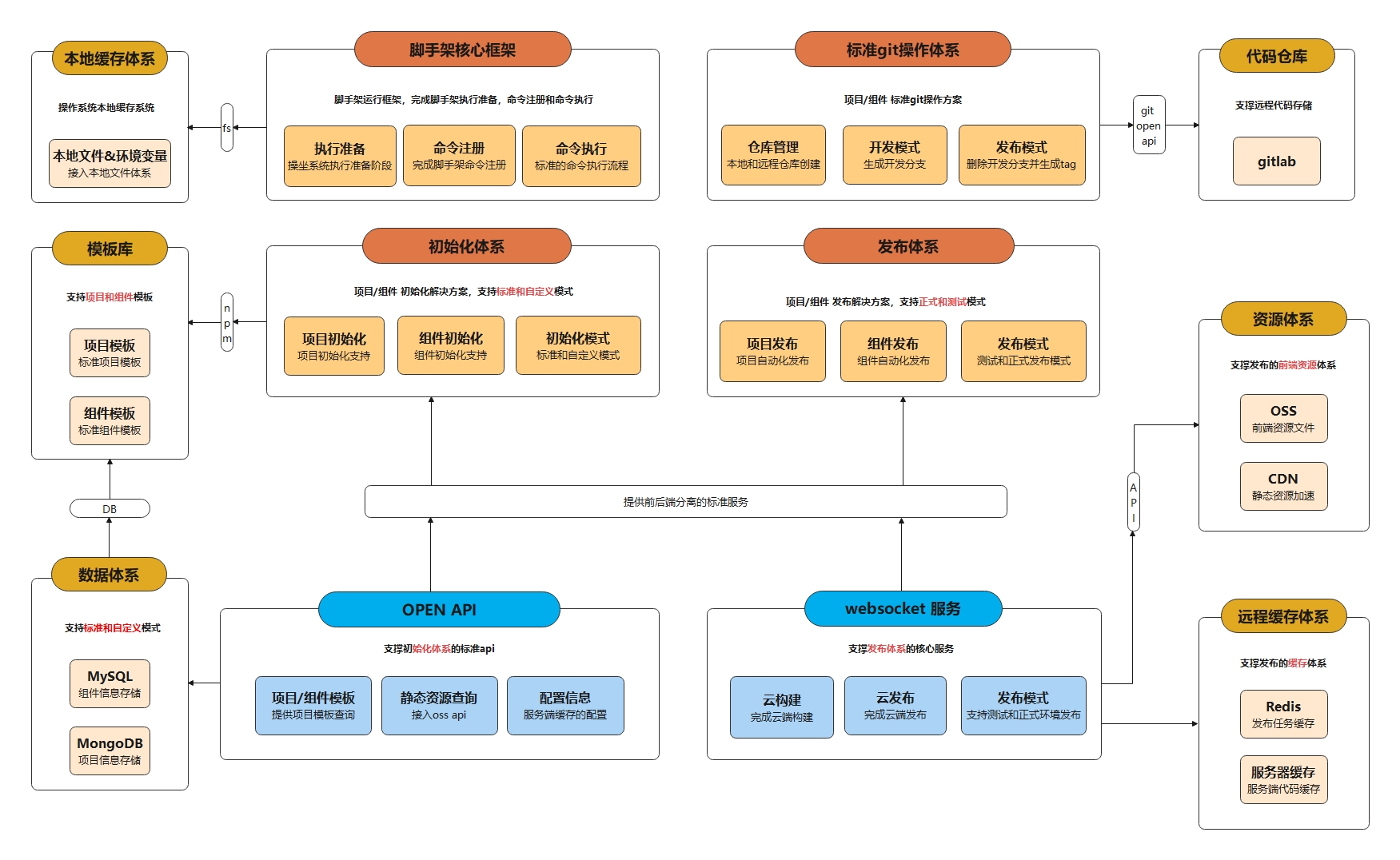


# 目标成果输出

* 实现可兼容windows、macOS 的脚手架工具，通过cli 初始化具有规范和工具链统一的项目。
* 该脚手架低耦合，功能可细粒度拆分，在一定成度上升级功能无需升级本地系统安装的脚手架，仅需远程升级功能包。
* 生成的项目模板列表可通过访问数据库动态获取。
* 该脚手架项目需要有严格的包管理工具。
* 输出模块化的库可以任意插拔随意组合。

# 脚手架设计架构图

## 图示



## Q & A

### 脚手架核心框架

Q：脚手架核心框架的本地缓存体系

A：脚手架初始化过程中，会用到某些配置信息在本地的缓存。有两种缓存方式，本地文件&环境变量

Q: 脚手架需要check root启动

A：通过root启动生成的文件其它账户是无法操作的，需要进行root降级

### 初始化体系

Q: 初始化体系中的模板库

A：模板库存储在npm中，因为npm天然适合做版本管理的特点，适合做模板库的存储。

Q: 初始化体系 为什么需要后台提供的标准api

A: 初始化体系中创建项目，希望实现通用能力，需要接入外部项目，这是一个动态过程，比如init 可选择的类型可动态配置。

Q：初始化过程支持 项目/组件 标准和自定义模式？

A： 标准模式按照创建项目过程中按照标准流程一步一步向下，自定义模式根据自己选择，比如可以选择初始化某些字段，具有更高的自由度。

Q：open api中静态资源查询能力？

A： 静态资源指的是组件当中会做一些查询，可能会查询到组件最终的一个构建结果，这个结果放在静态资源库。

Q：open api中的配置信息？

A：可以实现某一个细节点做成动态的，这个信息应该存储在配置文件，通过api调取配置文件

### 发布体系

Q: 发布体系为什么需要websocket服务

A: 发布体系依赖于websocket服务。存在云构建，云发布。虽然可以调用api让后台执行任务，但是前台无法知道。websocket可以做实时通知。

Q：发布体系中依赖的资源体系和远程缓存体系

资源体系：发布体系中依赖的资源体系，最终发布是会发布到资源体系。静态资源文件上传到oss，oss是不能加速的,所以oss同步到cdn实现加速

远程缓存体系： 做云构建过程中重要内容的缓存，比如公司内部脚手架被同时调用构建能力，是并行操作，会创建多条任务 。redis是缓存服务器，可以缓存多条任务。云构建过程中，需要把代码下载到本地进行一系列操作，云构建过程中代码是没有地方存放的，服务端缓存存放代码构建产物

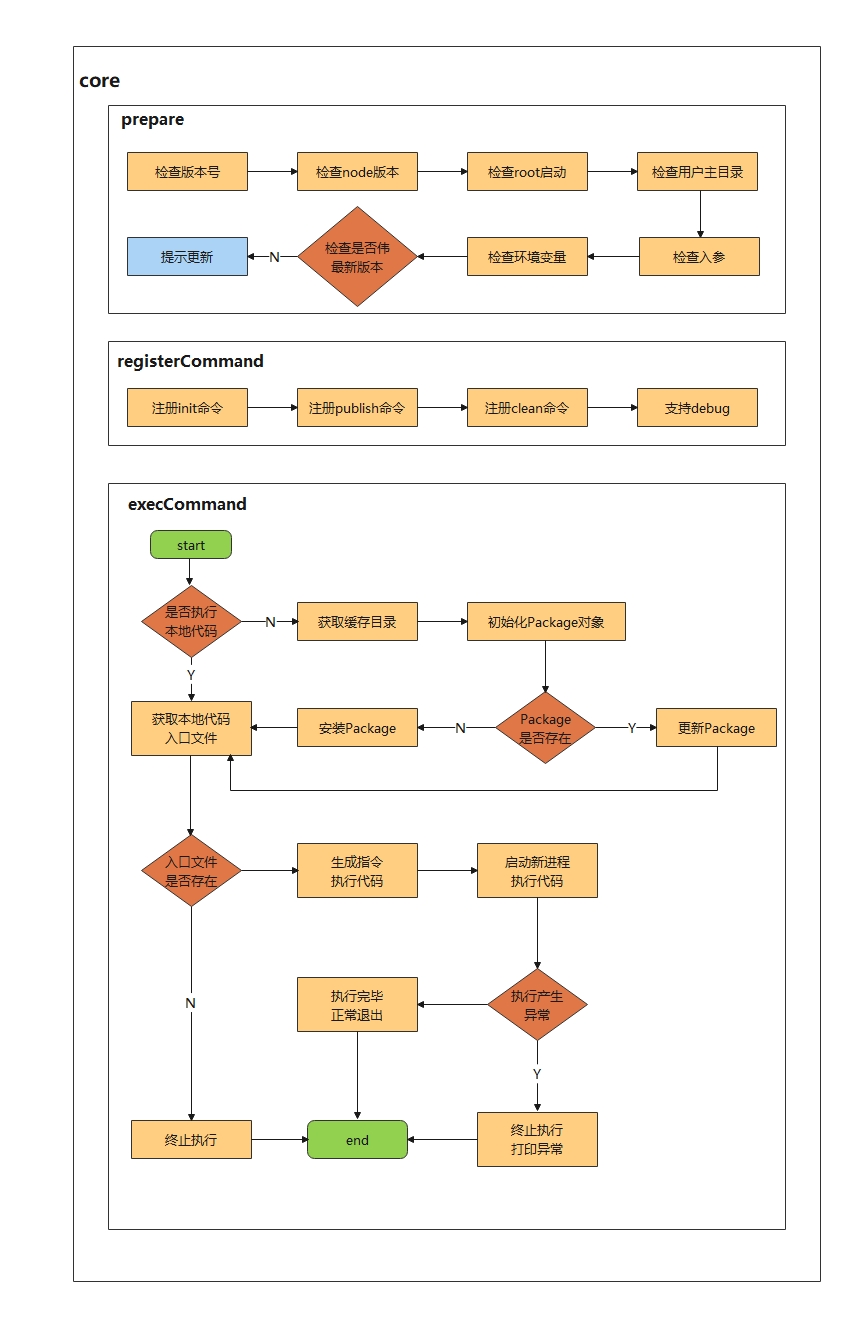
### 标准git 操作体系

# 脚手架拆包策略

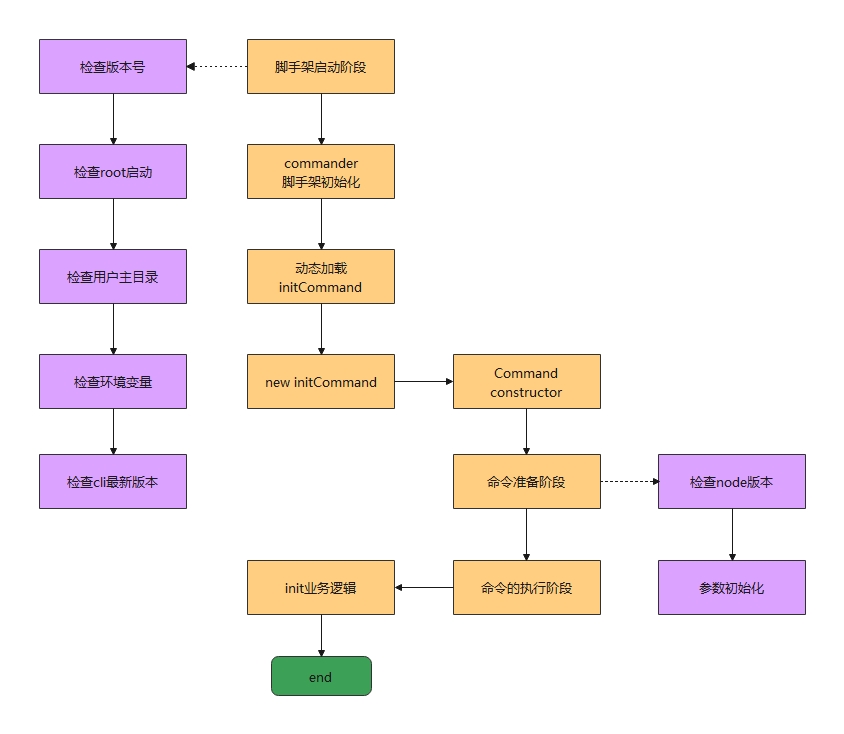
* 核心流程：core
* 命令：commands
* 初始化
* 发布
* 清除缓存
* 模型层：models
* Project 项目
* Component组件
* Npm 模块
* Git 仓库
* 支撑模块：utils
* Git 操作
* 云构建
* 工具方法
* API 请求
* Git API

# 模块方案

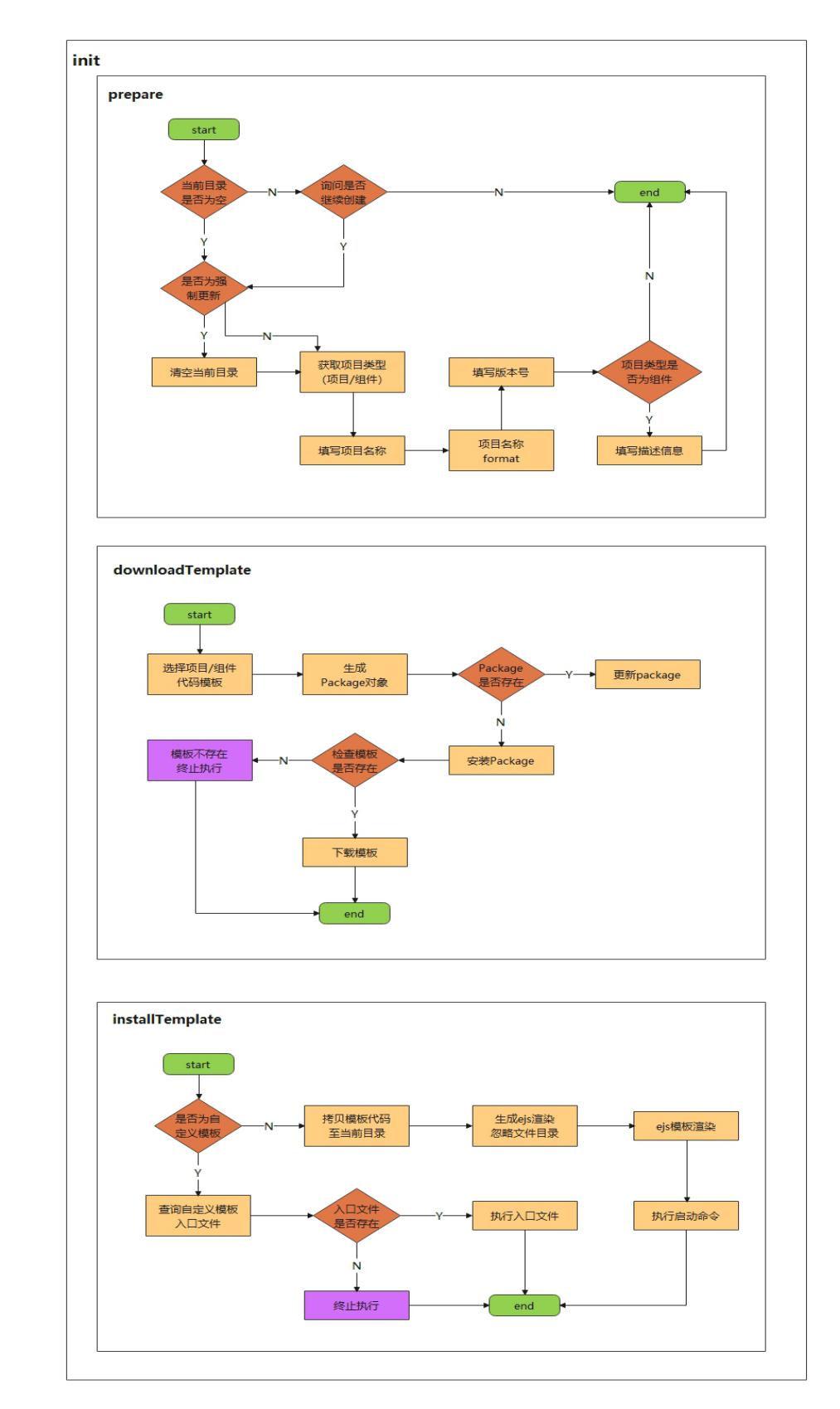
## 脚手架核心core模块方案



## 通用commander模块方案



## 脚手架创建项目功能方案



## 项目/组件模板设计

react-standard-template

   template

     ...

     package.json

   package.json

* react-standard-template 为npm的包，子 package.json 管理该npm包。
* template 包含实际的模板文件/文件夹，文件可做配置ejs模板渲染，子 package.json 部分属性做动态化模板渲染。

## 数据库设计

获取模板列表:

   {

        "name":"React 标准项目模板",

        "npmName":"@<% orgnizationName %>/react-standard-template",

        "version":"1.0.0",

        "type":"normal",

        "startCommand":"npm start",

        "tag":[

            "project"

        ],

        "buildPath":"build"

    },

    {

        "name":"Vue3 标准项目模板",

        "npmName":"@<% orgnizationName %>/vue3-standard-template",

        "version":"1.0.0",

        "type":"normal",

        "startCommand":"npm run serve",

        "ignore":[

            "\*\*/public/\*\*"

        ],

        "tag":[

            "project"

        ],

        "buildPath":"dist"

    },

    {

        "name":"Vue3 标准组件模板",

        "npmName":"@<% orgnizationName %>/vue3-component",

        "version":"1.0.0",

        "type":"normal",

        "tag":[

            "component"

        ],

        "startCommand":"npm run dev",

        "buildPath":"dist",

        "examplePath":"examples"

    },

# 技术储备

Node.js 、fs、lerna.js

Lerna.js 作为多包管理工具，用来管理开发模块化库时多个包之间的版本依赖。

## 相关npm包

* import-local
* commander
* npm-log
* fs-extra
* semver：做版本比对
* colors
* user-home：拿到用户主目录
* dote-env： 拿到环境变量
* root-check：做root账户的检查和降级
* Inquirer 用户输入交互界面

## 其它

支撑初始化体系的标准api应用 Express/koa2、mysql/mongodb