开源社
kaiyuanshe

开源组织的基础设施工程化指南

快速构建开源组织的工程化基础设施

李宇飞 2022.10.30

目录
CONTENTS

开源社 kaiyuanshe

以云原生配置语言 CUE 为线索，介绍如何构建开源工程化基础设施。

概览

开源社 kaiyuanshe

基础设施即代码

Terraform

如何管理开源组织的
工程化基础设施

ChatOps

Prow

如何构建维护者和贡
献者之间的桥梁

持续交付工作流

Dagger

如何更高效地实施
CI & CD 流水线

开源社
kaiyuanshe

配置管理与 Terraform

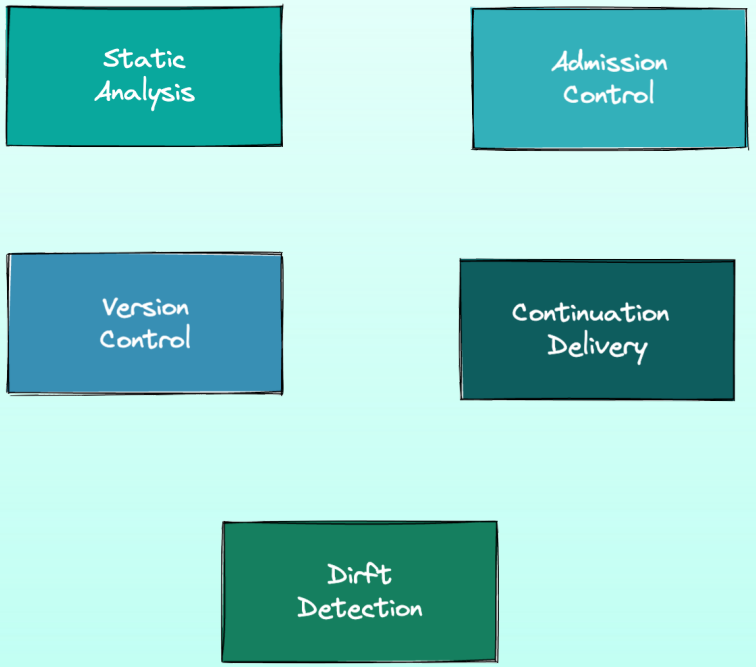
步步常由逆境行，极知造物欲其成。

—— 陆游

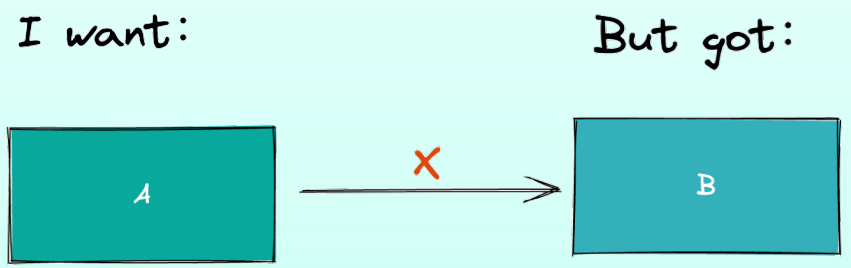
开源组织配置管理的挑战

开源社 kaiyuanshe

治理复杂度高



配置漂移



配置管理的演进

开源社 kaiyuanshe

相关研究

- 可参考 Mark Burgess 在配置管理领域的论文《Configurable immunity for evolving human-computer systems》中：关于收敛算子和不动点的理论的研究。
- 配置管理工具的目标是，将「**系统状态**」逐渐收敛至特定的不动点。

常见配置管理工具

CFEngine

CHEF

ANSIBLE

puppet

HashiCorp Terraform

Pulumi

配置管理的演进

开源社 kaiyuanshe

命令式 API

```
POST /orgs/{org}/repos
{
  org: 'ORG',
  name: 'Hello-World',
}
PATCH /orgs/{org}/repos
{
  org: 'ORG',
  name: 'Hello-World',
  description: "A demo for coscon22!"
}
```

声明式 API

```
{
  org: 'ORG',
  name: 'Hello-World',
+  description: "A demo for coscon22!"
}
```

I will do → I want to do

配置管理的演进

开源社 kaiyuanshe

Multi-Cloud

Imperative API

Declarative API

HashiCorp Terraform

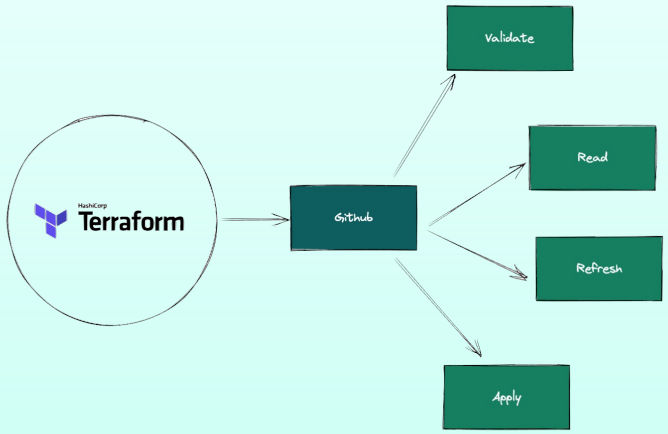
什么是 Terraform

开源社 kaiyuanshe

资源定义

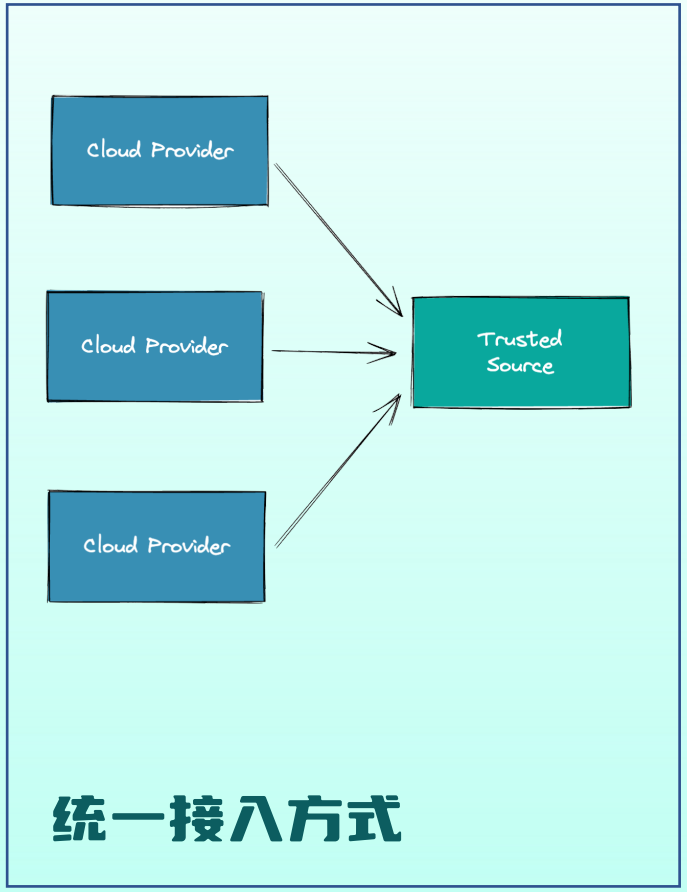
```
resource "github_team" "some_team" {
  name        = "SomeTeam"
  description = "Some cool team"
}

resource "github_team_membership" "some_team_membership" {
  team_id  = github_team.some_team.id
  username = "SomeUser"
  role     = "member"
}
```

资源编排

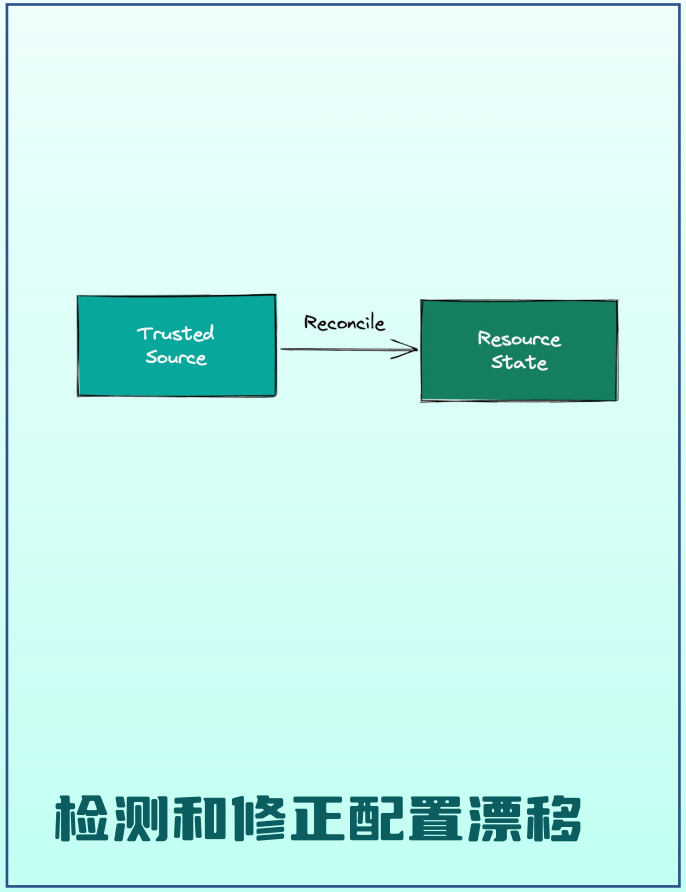


- **核心进程**：解析资源定义，编排资源
- **Provider**：执行实际的资源操作

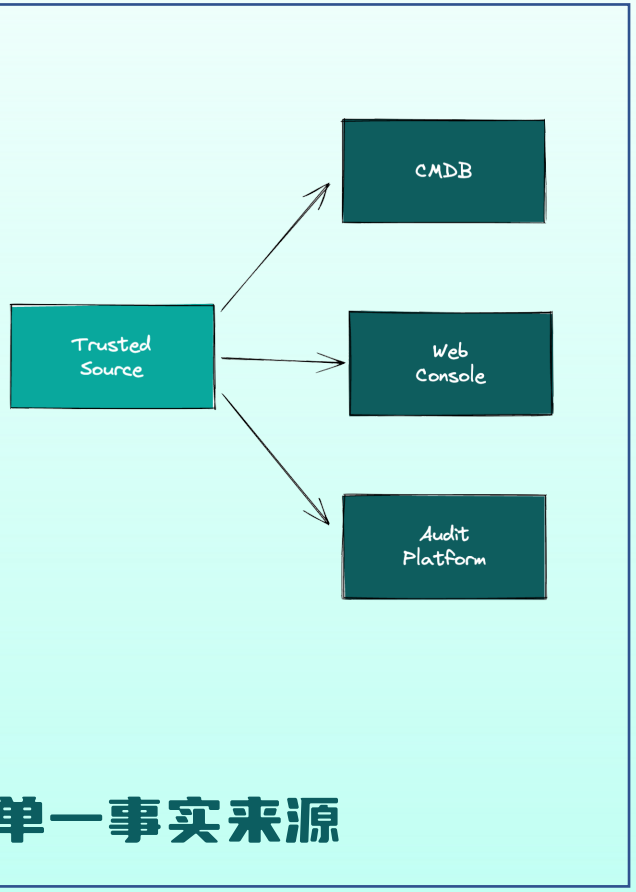
Terraform 的优势



统一接入方式



检测和修正配置漂移



单一事实来源

CUE lang + Terraform

开源社 kaiyuanshe

{JSON}

YAML

...

生态融合

```
package coscon22

import (
  "yufeiminds.com/coscon22/repos"
)
```

```
#ReuseCode: {
  arguments: {
    text: string
  }

  output: "Hello, \(arguments.text)"
}
```

代码复用

```
cue.mod
├── dagger.mod
├── dagger.sum
├── module.cue
├── pkg
|   ├── dagger.io
|   |   ├── cue.mod
|   |   └── dagger
|   └── universe.dagger.io
|       ├── terraform
|       ├── cue.mod
└── usr
    └── yufeiminds.com
        └── coscon22
```

抽象隔离

小结

开源社 kaiyuanshe

- 配置管理演进的两个阶段：命令式 API、声明式 API
- 基础设施即代码的优势：
  - 统一接入方式
  - 避免配置漂移
  - 构建单一事实来源
- 配置语言带来的提升：
  - 抽象隔离
  - 代码复用
  - 融合其它的生态
- 演示：如何用 CUE 和 Terraform 管理 GitHub 组织

开源社
kaiyuanshe

Kubernetes Prow

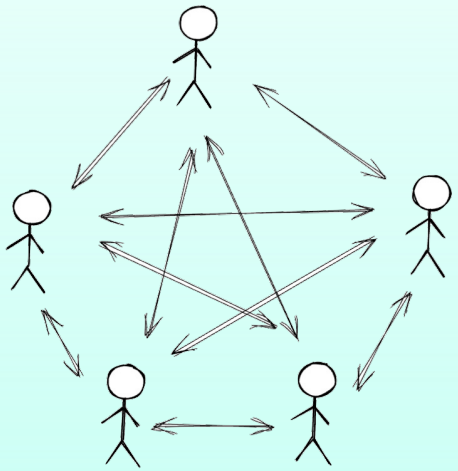
步步常由逆境行，极知造物欲其成。

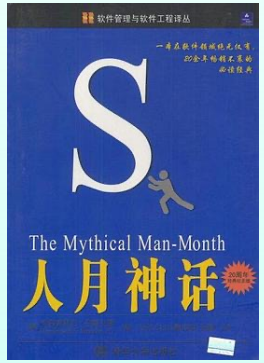
—— 陆游

开源组织协作面临的挑战

开源社 kaiyuanshe

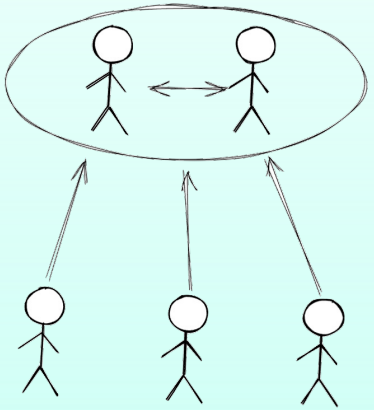
Brooks 定律（传统组织）

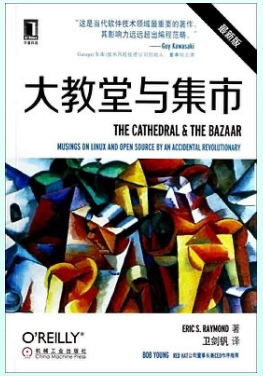




沟通是一张完全图

贡献者阶梯（开源组织）





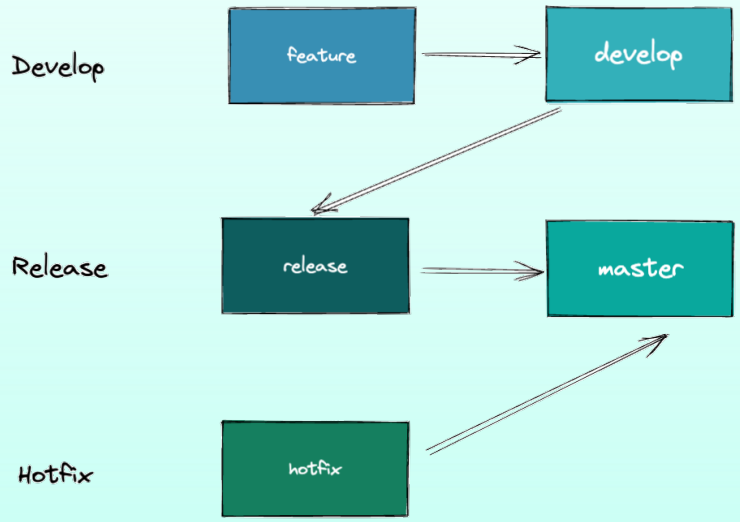
贡献单向流动到核心团队
仅在核心团队有 Brooks 开销

如何**优化贡献流向维护者的路径**，使其更加**自动化**，是开源工程化所面临的挑战。

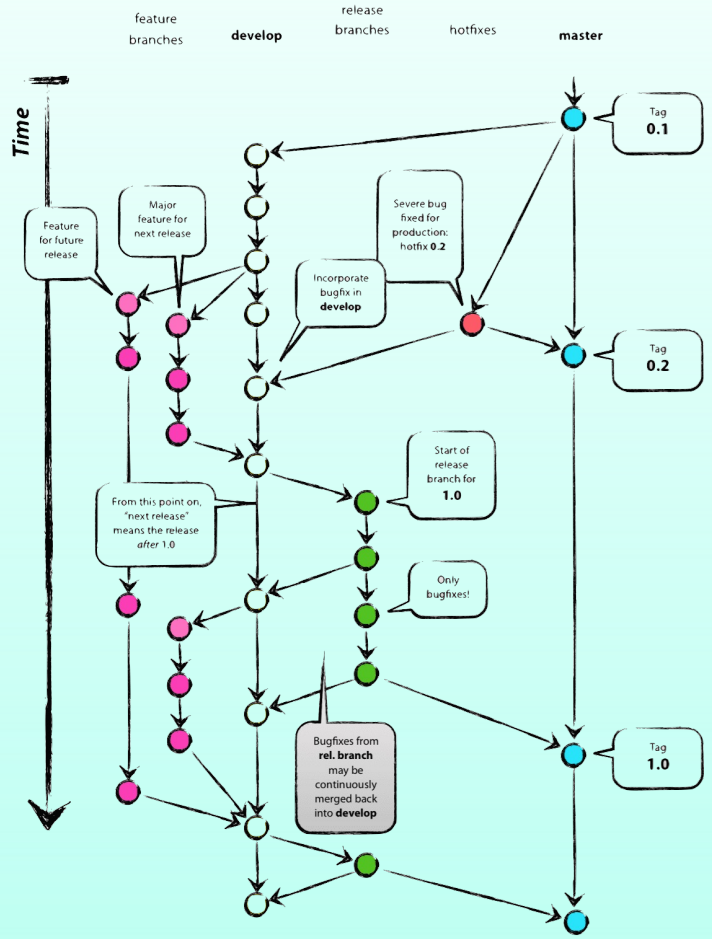
开源协作模型：Git Flow

开源社 kaiyuanshe

代码流动



- 长期分支：master、develop
- 支持分支：feature、release、hotfix

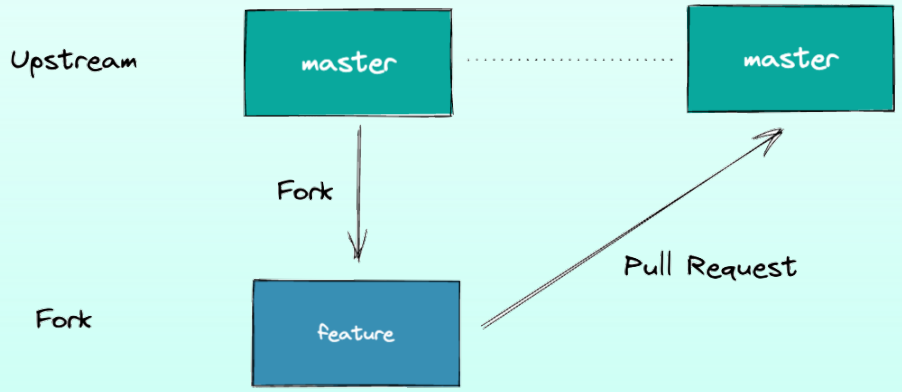


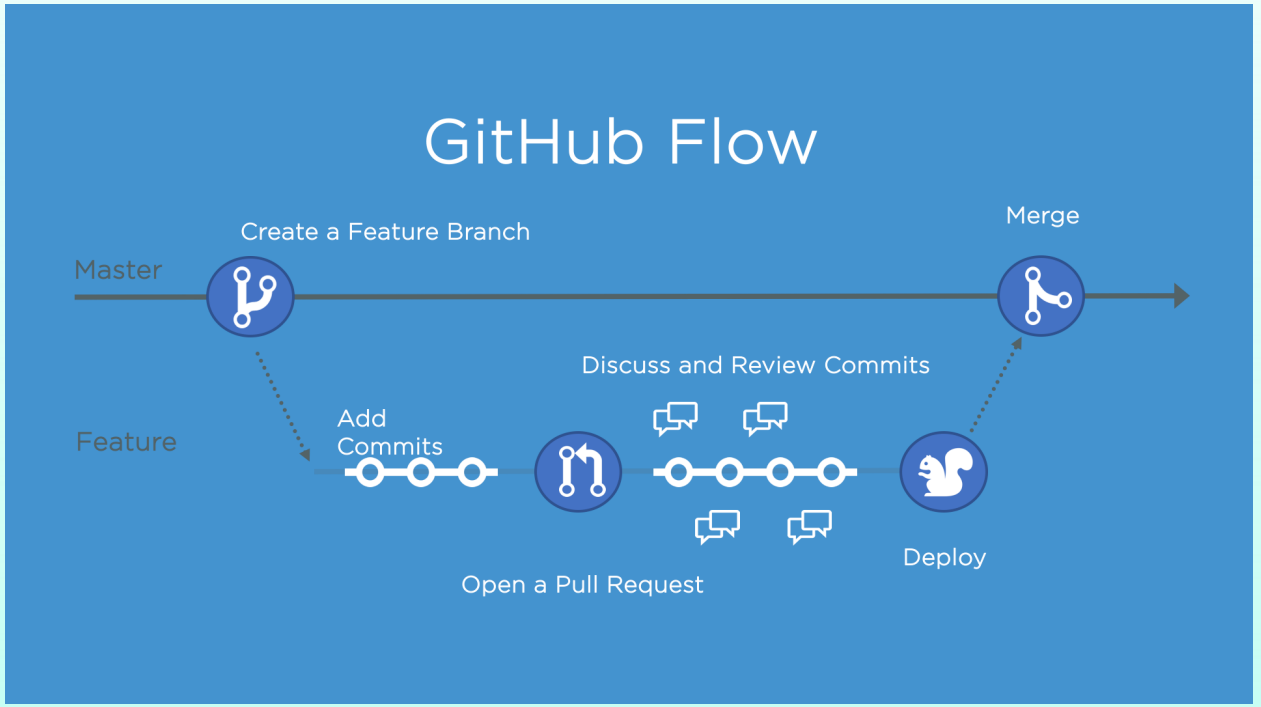
图：Vincent Driessen
《A successful Git branching model》

开源协作模型：GitHub Flow

开源社 kaiyuanshe

代码流动





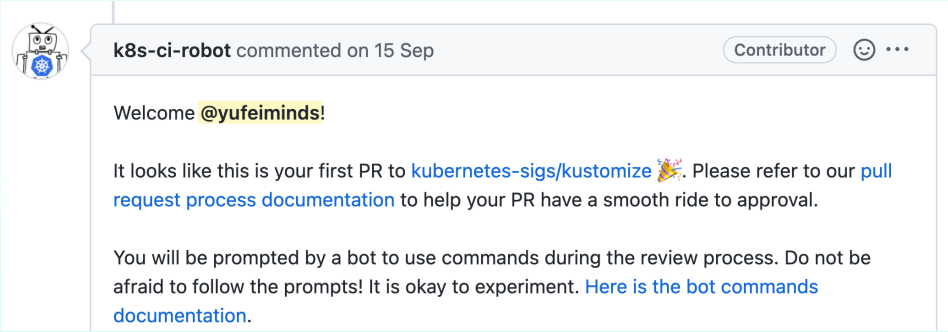
- 简化的分支管理
- 聚焦在 Pull Request

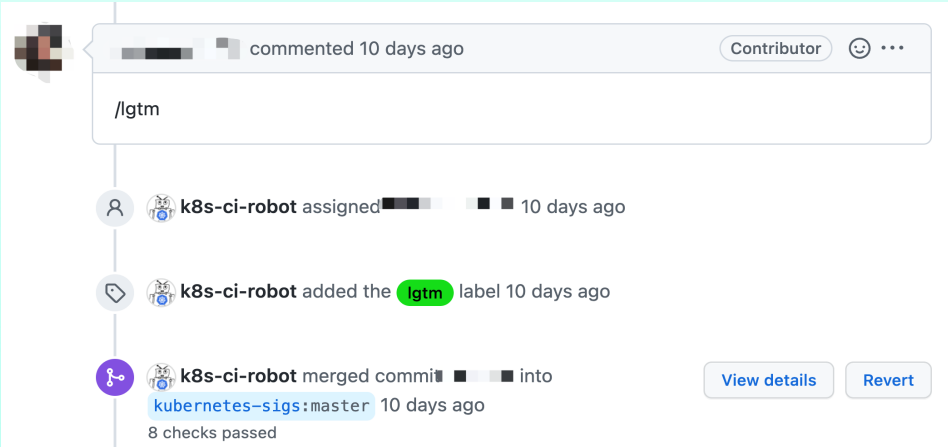
图：Vincent Driessen
《A successful Git branching model》

GH Flow & Chat-Ops

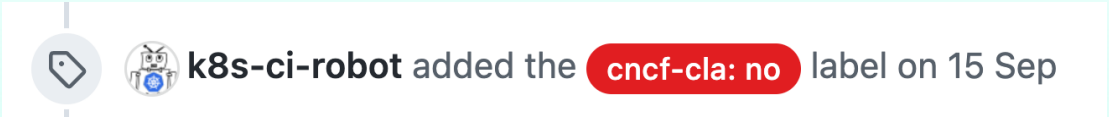
开源社 kaiyuanshe

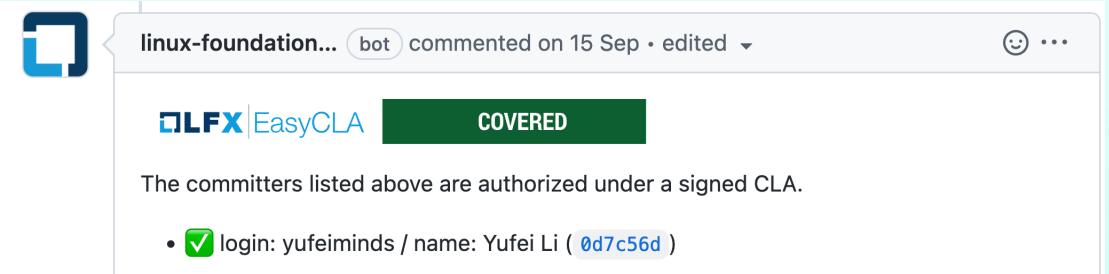
流程引导

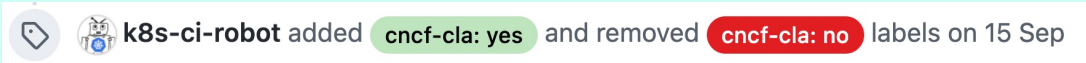




状态追踪



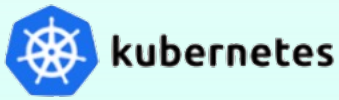


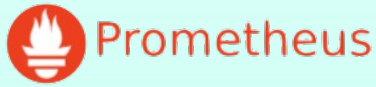


- 以 PR 为中心
- 机器人流程自动化

哪些项目在使用 Prow？

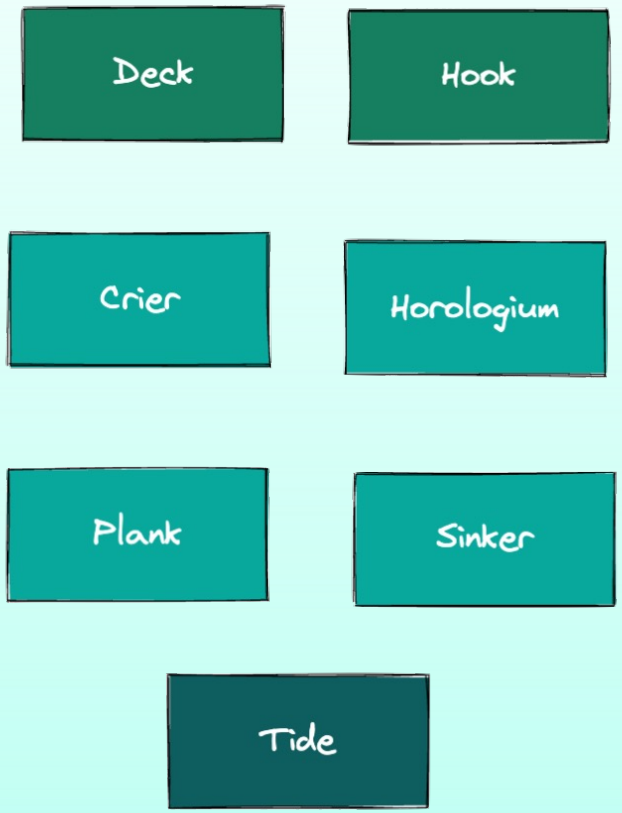
开源社 kaiyuanshe





Prow 的系统架构

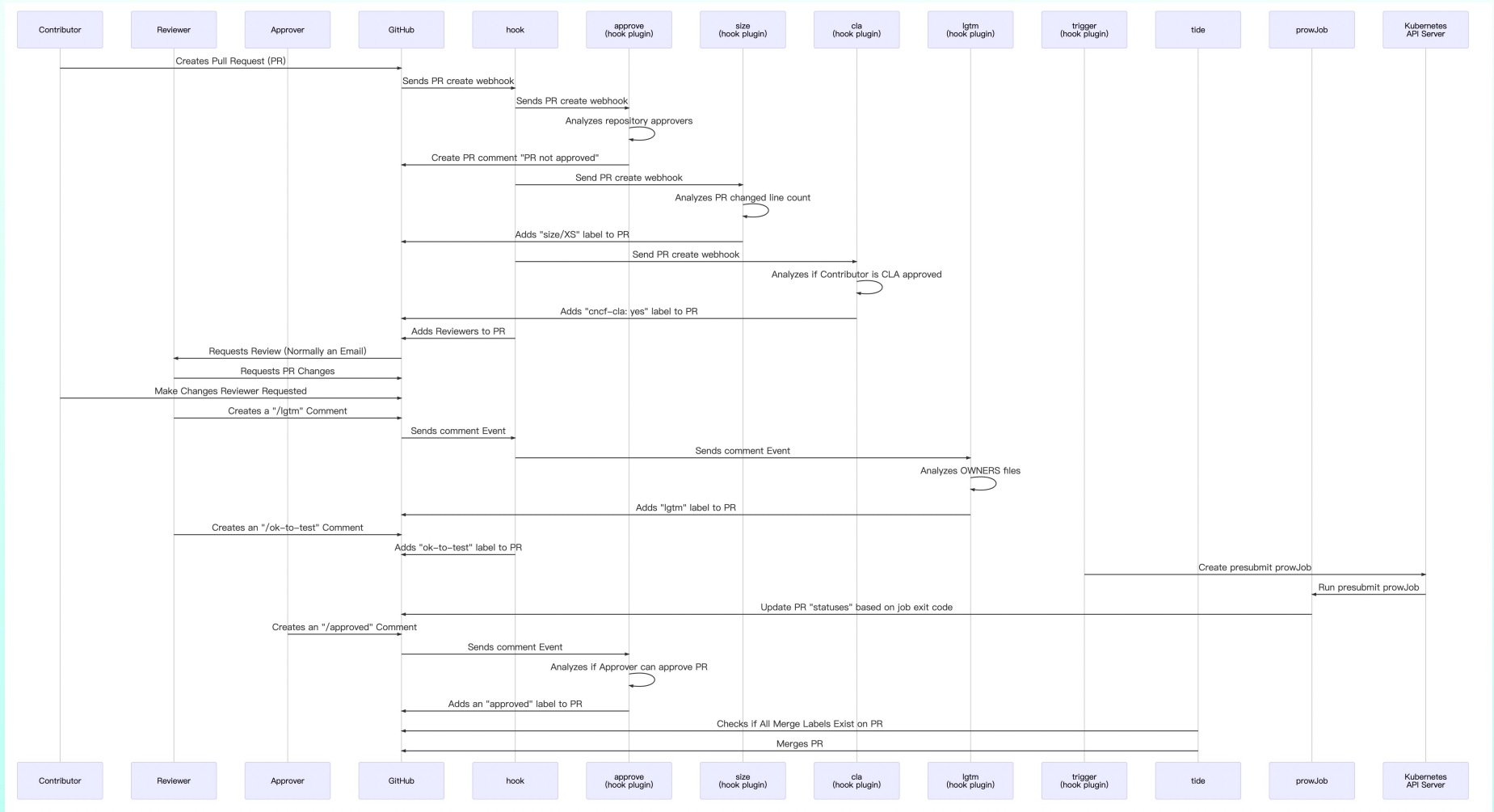
组件清单	介绍
Deck	可视化面板
Crier	同步任务状态
Hook	接收 GitHub Webhook
Horologium	触发定时任务
Plank	管理任务生命周期
Sinker	清理遗留任务和 Pod
Tide	自动合并 PR



图：Vincent Driessen
《A successful Git branching model》

Prow 工作原理解析

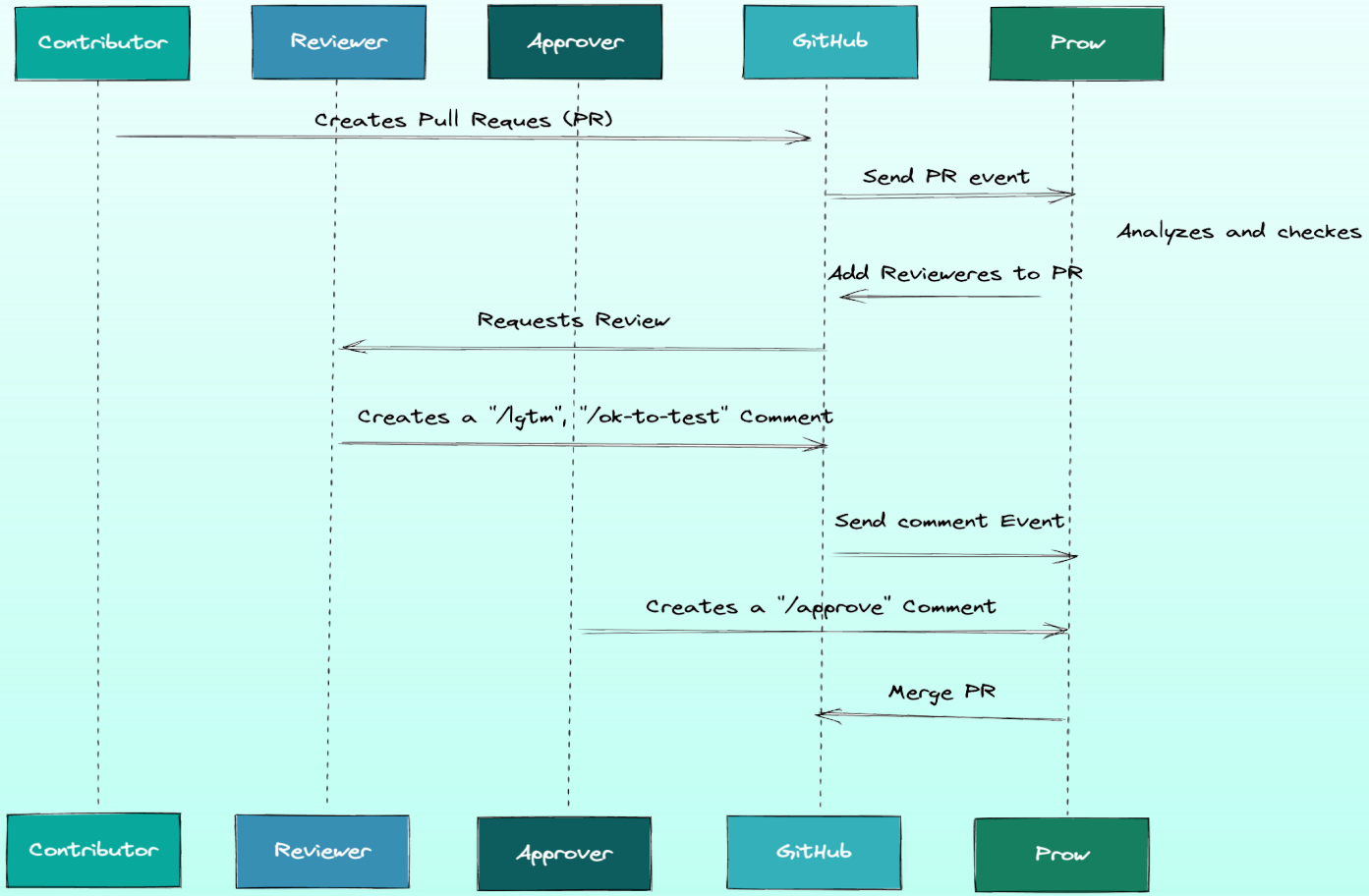
开源社 kaiyuanshe



太复杂?
没关系~
请看下一页!

Prow 工作原理解析

开源社 kaiyuanshe



小结

开源社 kaiyuanshe

- 介绍了开源协作的三种工作流：
  - **Git Flow**：常见于组织内单仓库的协作
  - **GitHub Flow**：常见于开源社区
  - **Chat-Ops：机器人自动化工作流，常见于现代化的开源社区协作**
- 介绍了 **Kubernetes Prow**：
  - Kubernetes 官方使用的 CI/CD 系统
  - 用于实现基于 PR 的 Chat-Ops 开源协作工作流
- Prow 给开源贡献带来的提升包括：
  - **流程优化**：给了贡献者一个清晰的指引，让开源贡献者对整个 PR 流程的关键节点有明确的预期
  - **职责分离**：明确了不同角色职责的边界，在贡献的各个阶段自动化地引入不同的角色，减轻了维护者的工作负担

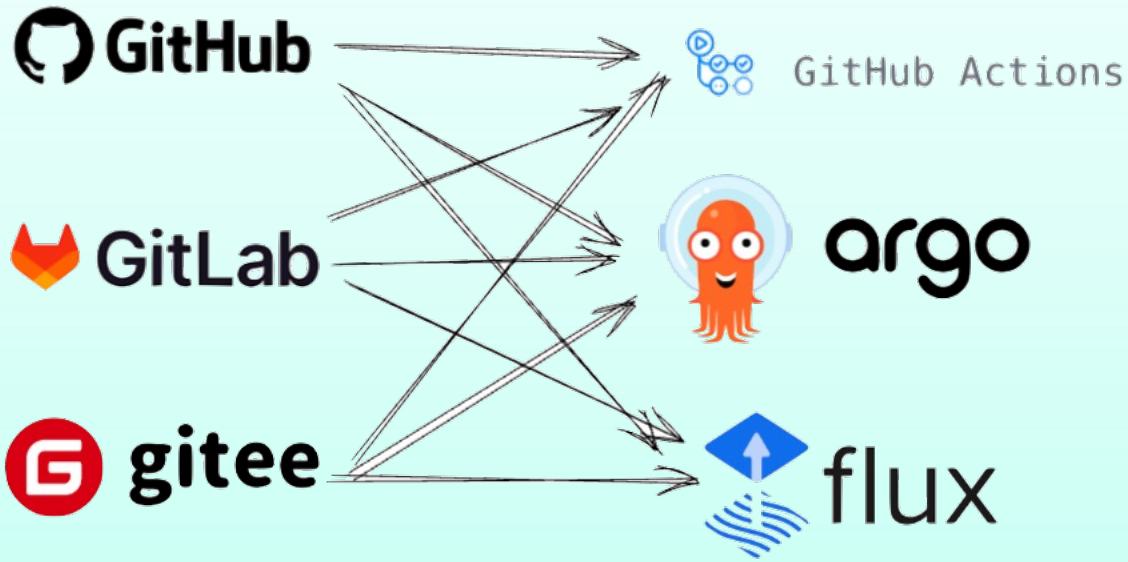
开源社
kaiyuanshe

Dagger

步步常由逆境行，极知造物欲其成。

—— 陆游

CI & CD 面临的挑战



如何尽可能**规约管道中的环节**，减少可能的**路径损耗**，是开源工程化所面临的挑战。

王二的一天

开源社 kaiyuanshe

Git push!

2000 years later ...

又写错了!

CI 不能本地执行

Python?

CI 上拉个 Python 镜像?

镜像在哪?

不想写 Shell

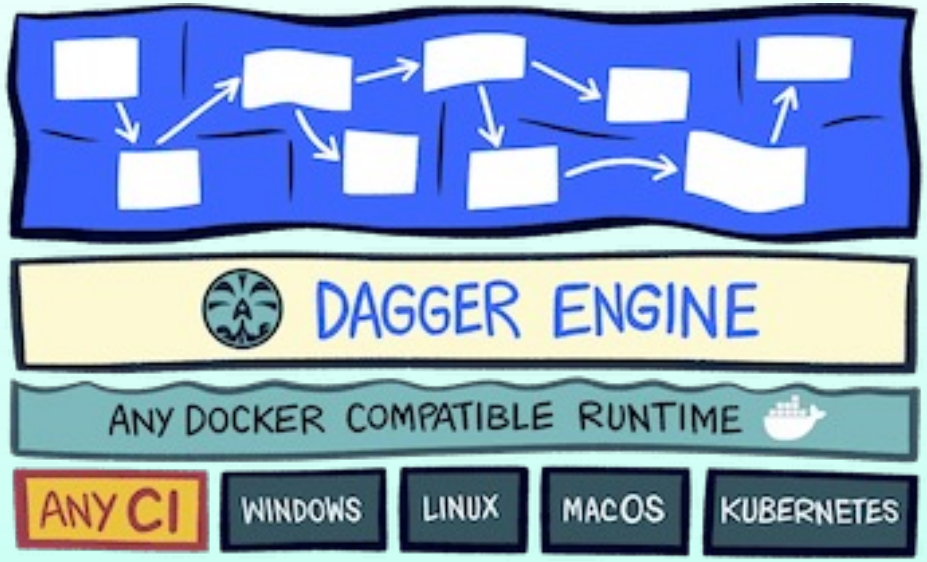
重写!

全部重写!

我想换 CI/CD 平台

Dagger 工作原理解析

开源社 kaiyuanshe



All in container!

Dagger 工作原理解析

开源社 kaiyuanshe

Plan & Action & Client

```
dagger do build
```

- Plan 是 Dagger 的入口
- Action 是具体被执行的单元
- Client 是与外部系统交互的方式
- Action 之间也可以互相引用
- 引用关系构成的图被 Dagger 解释执行

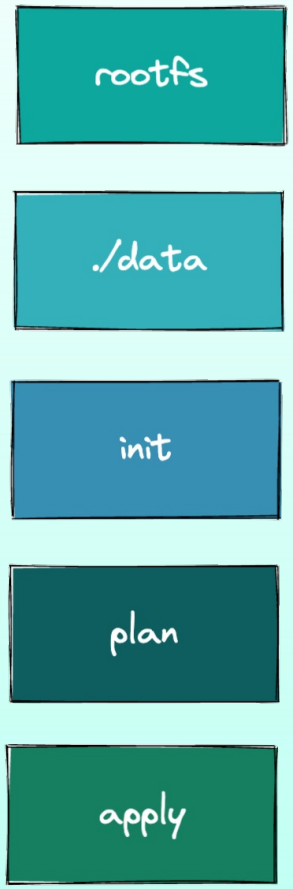
```
dagger.#Plan & {
  client: {
    filesystem: {
      // ...
    }
    env: {
      // ...
    }
  }
  actions: {
    deps: docker.#Build & {
      // ...
    }
    test: bash.#Run & {
      // ...
    }
    build: {
      run: bash.#Run & {
        // ...
      }
      contents: core.#Subdir & {
        // ...
      }
    }
    deploy: netlify.#Deploy & {
      // ...
    }
  }
}
```

Dagger 工作原理解析

开源社 kaiyuanshe

Filesystem

- 类似于 Docker 的 Layer FS
- Dagger 首先加载 rootfs
- 根据依赖顺序层层叠加
- Layer 有缓存，因此重复执行很快




```
dagger.#Plan & {
 actions: test: {
  tfSource: core.#Source & {
   path: "./data"
  }

  applyWorkflow: {
   init: terraform.#Init & {
    source: tfSource.output
   }

   plan: terraform.#Plan & {
    source: init.output
   }

   apply: terraform.#Apply & {
    source: plan.output
   }
  }
 }
}
```

Dagger 工作原理解析

开源社 kaiyuanshe

标准库 / 生态集成

- CUE 上游维护了一系列标准库
- Dagger Team维护了与系统交互的基础库
- Dagger 社区维护了与生态系统集成的库
- 开发者可以根据上述实现，封装自己的库实现

```
import (
  // cue standard library
 "encoding/json"

  // maintain by dagger.io
 "dagger.io/dagger"
 "dagger.io/dagger/core"

  // contribute by community
 "universe.dagger.io/alpha/terraform"

  // developer by ourself
 "yufeiminds.com/coscon22"
)
```

小结

开源社 kaiyuanshe

- Dagger 解决了什么问题：
  - **通过 Docker 底层技术解除 Git 供应商锁定**
  - **通过 CUE 混合配置来源**
  - **提升 CI/CD 的可复用性和可编程性**
- Dagger 的基本介绍：
  - Plan
  - Action
  - Client
  - File System
- 演示：使用 CUE + Terraform + Dagger 实现开源组织的成员和仓库管理

开源社 kaiyuanshe

Talk is cheap, show me the code!

Demo 演示

使用上述工具实现开源组织的成员和仓库管理

https://github.com/yufeiminds/coscon22

总结

开源社 kaiyuanshe

Terraform

- **挑战：优化基础设施的维护成本，减少「配置漂移」**
- 介绍：IaC 的优势：统一接入方式、避免配置漂移、构建单一事实来源
- 介绍：CUE 在 IaC 中的作用：抽象隔离、代码复用、融合其它的生态

Kubernetes Prow

- **挑战：优化贡献流向维护者的路径，使其更加流畅和自动化**
- 介绍：Git Flow、GitHub Flow、Chat-Ops
- 介绍：Prow 的系统架构和工作流

Dagger

- **挑战：尽可能规约管道中的环节，减少可能的路径损耗**
- 介绍：Dagger 的基本知识
- **演示：**使用 CUE + Terraform + Dagger 实现开源组织的成员和仓库管理

开源社
kaiyuanshe

一直以来我有一个理想，
希望中国的基础软件厂商，
逐渐重视工具和工程化基础设施在开源治理中的作用，
让先进的生产力成为技术进步的二阶推进器。

开源社
kaiyuanshe

THANK YOU

QUESTIONS?

微信公众号：开源社KAIYUANSHE
视频号：开源社KAIYUANSHE
新浪微博：开源社
B站：2020开源社

简书：开源社
头条：开源社
Facebook：KaiyuansheChina
Twitter：KAIYUANSHE

扫码关注开源社公众号