KusionStack:“后云原生时代” 应用规模化运维解决方案

李大元

Agenda

About me

- 李大元（花名：达远）
- Kusion 项目负责人
- 蚂蚁集团 PaaS 核心团队，IaC 基础平台负责人

- Github: https://github.com/KusionStack/kusion
- Website: https://kusionstack.io

KusionStack 微信群

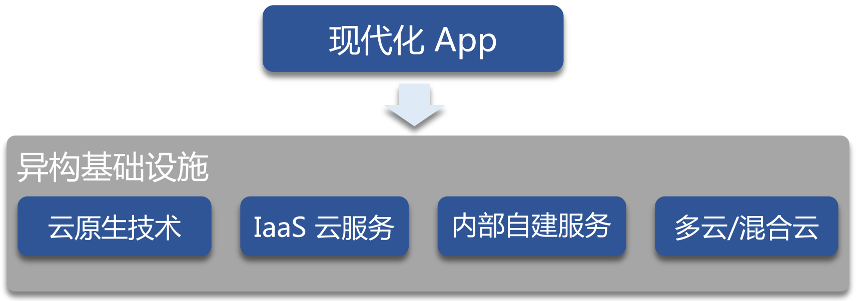
KusionStack 官方用户群
78 人

扫一扫群二维码，立刻加入该群。

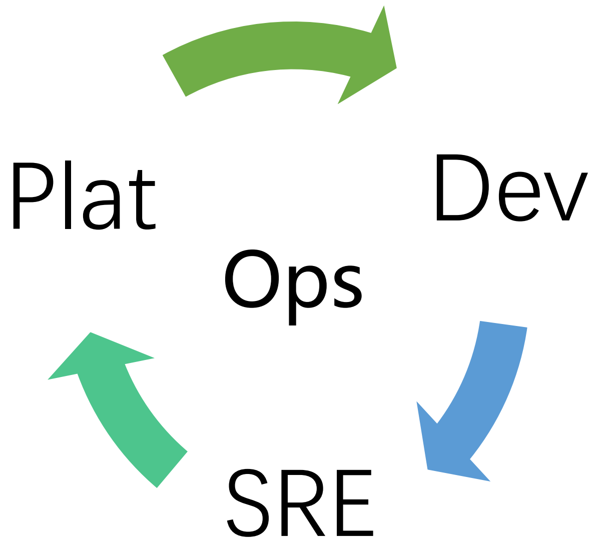
KusionStack 小助手

Origin

- 云原生不再是新技术而是“标配”，我们已经进入到“后云原生时代”
  - 距离 K8s 第一个 commit 已经过去 8 年多了
  - 现代化应用：云原生技术 + IaaS 云服务 + 内部自建服务 + 多云/混合云
  - 只针对 K8s 的运维工具已经不能满足我们的诉求

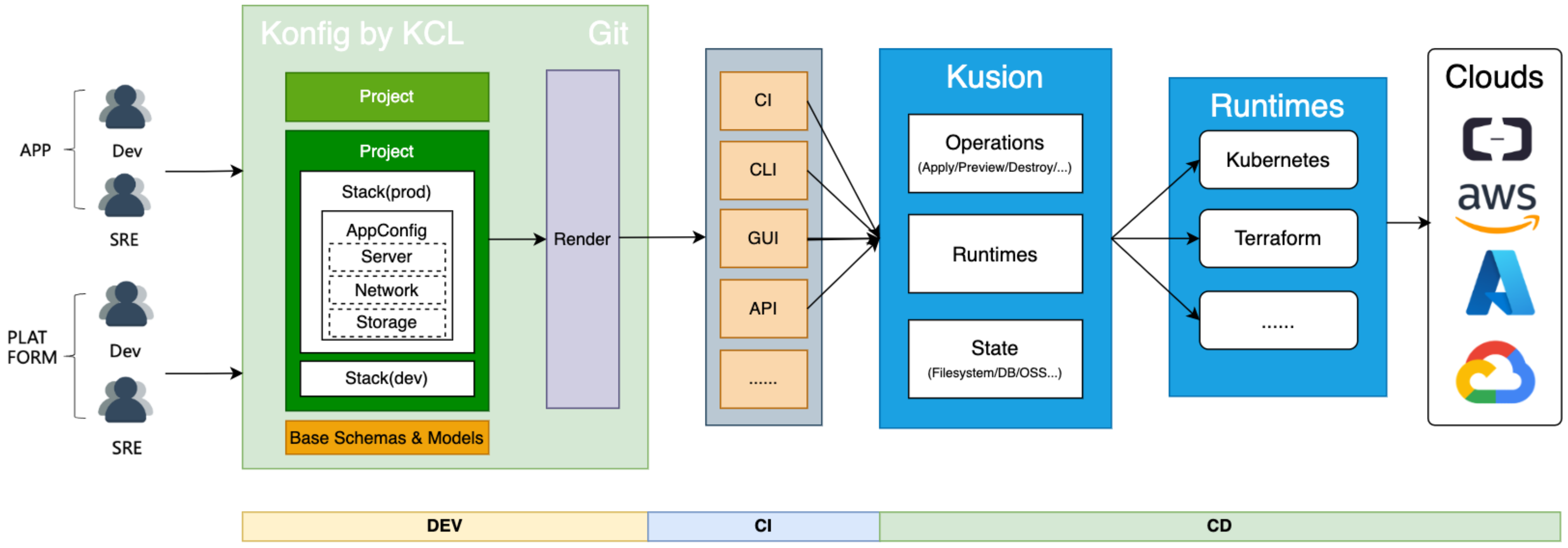


- 异构基础设施规模化运维，需要更高效的团队协同机制
  - 在规模化运维中，多个 Platform 团队之间需要协同工作
  - 在规模化运维中， App Dev 与 Platform Dev 之间需要协同工作
  - “工单模式”运维平台不够开放，“硬编码”对接基础设施，效率低、迭代慢



What is KusionStack

Codify, **Collaborate** and **Automate** modern App operations across Kubernetes and clouds



Highlights

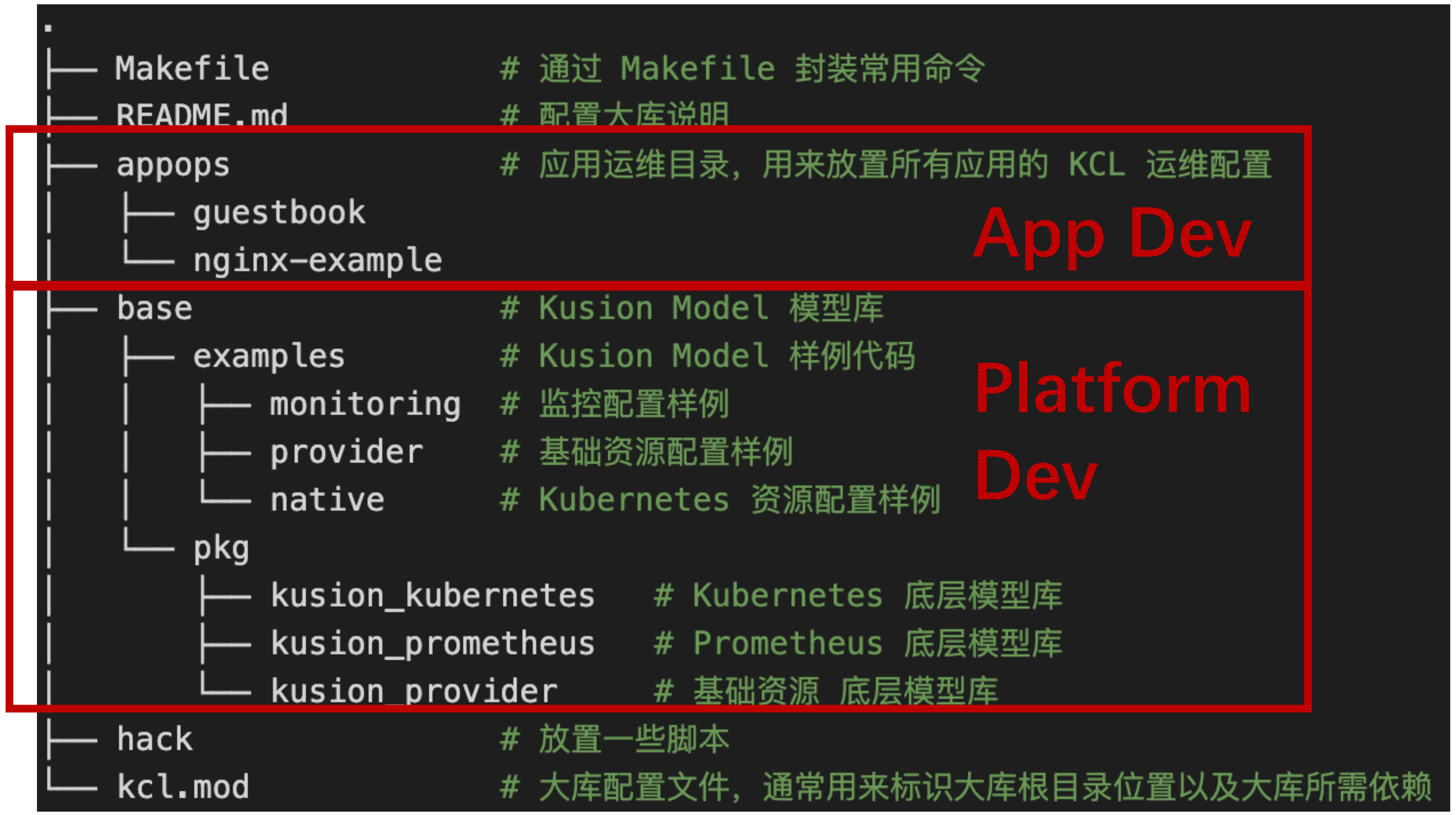
- 以应用为中心
  - 应用全方位配置管理，包括计算、网络、存储等所有与应用有关配置
  - 应用全生命周期管理，从第一行配置代码到生产可用

- 统一运维 “后云原生时代” 应用的异构基础设施
  - K8s 友好的工作流，为 K8s 资源提供可观测性、健康检查等高阶能力，释放云原生技术红利
  - 复用 Terraform 生态，统一的工作流运维 K8s、Terraform 多运行时资源

- 规模化协同平台
  - 通过代码抽象、组合等方式，屏蔽基础设施复杂性，应用可以简单、灵活配置所需基础设施
  - App Dev 和 Platform Dev 关注点分离，底层能力迭代无需平台介入，直接供 App 使用
  - 纯客户端方案，风险 “左移” ，尽早发现问题

Architecture

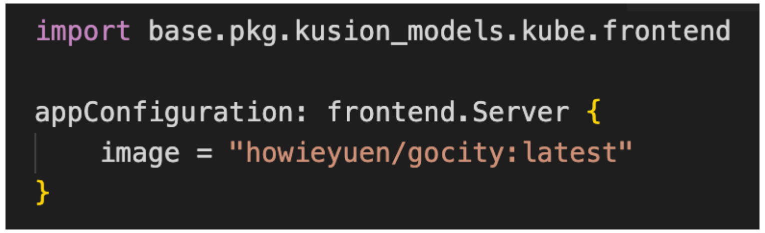
Konfig：统一配置大库



```
.
├── Makefile                # 通过 Makefile 封装常用命令
├── README.md               # 配置大库说明
├── appops                  # 应用运维目录，用来放置所有应用的 KCL 运维配置
│   ├── guestbook
│   └── nginx-example
├── base                    # Kusion Model 模型库
│   ├── examples            # Kusion Model 样例代码
│   │   ├── monitoring      # 监控配置样例
│   │   ├── provider        # 基础资源配置样例
│   │   └── native          # Kubernetes 资源配置样例
│   └── pkg
│       ├── kusion_kubernetes   # Kubernetes 底层模型库
│       ├── kusion_prometheus   # Prometheus 底层模型库
│       └── kusion_provider     # 基础资源 底层模型库
├── hack                    # 放置一些脚本
└── kcl.mod                 # 大库配置文件，通常用来标识大库根目录位置以及大库所需依赖
```

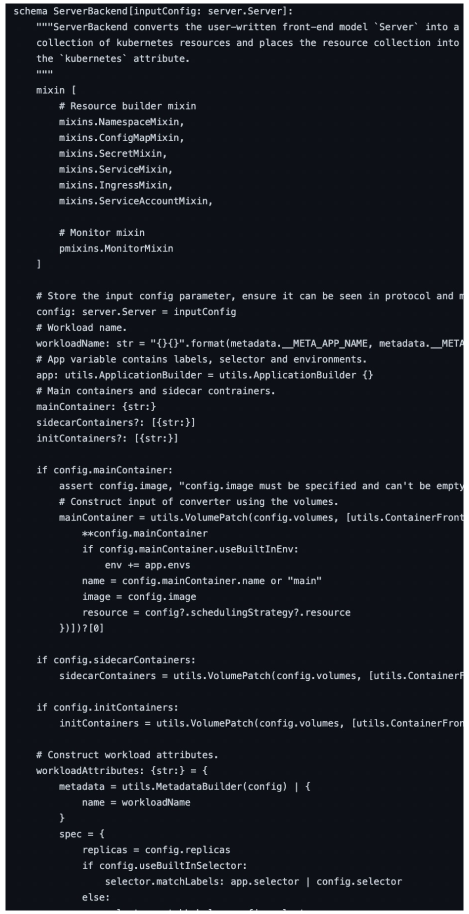
Architecture

KCL：配置策略语言



```
import base.pkg.kusion_models.kube.frontend

appConfiguration: frontend.Server {
    image = "howieyuen/gocity:latest"
}
```

前端模型（App Dev）



```
schema ServerBackend[inputConfig: server.Server]:
    """ServerBackend converts the user-written front-end model `Server` into a
    collection of kubernetes resources and places the resource collection into
    the `kubernetes` attribute.
    """
    mixin [
        # Resource builder mixin
        mixins.NamespaceMixin,
        mixins.ConfigMapMixin,
        mixins.SecretMixin,
        mixins.ServiceMixin,
        mixins.IngressMixin,
        mixins.ServiceAccountMixin,

        # Monitor mixin
        pmixins.MonitorMixin
    ]

    # Store the input config parameter, ensure it can be seen in protocol and m
    config: server.Server = inputConfig
    # Workload name.
    workloadName: str = "{}{}".format(metadata.__META_APP_NAME, metadata.__META
    # App variable contains labels, selector and environments.
    app: utils.ApplicationBuilder = utils.ApplicationBuilder {}
    # Main containers and sidecar contrainers.
    mainContainer: {str:}
    sidecarContainers?: [{str:}]
    initContainers?: [{str:}]

    if config.mainContainer:
        assert config.image, "config.image must be specified and can't be empty
        # Construct input of converter using the volumes.
        mainContainer = utils.VolumePatch(config.volumes, [utils.ContainerFront
            **config.mainContainer
            if config.mainContainer.useBuiltInEnv:
                env += app.envs
            name = config.mainContainer.name or "main"
            image = config.image
            resource = config?.schedulingStrategy?.resource
        })])?[0]

    if config.sidecarContainers:
        sidecarContainers = utils.VolumePatch(config.volumes, [utils.ContainerF

    if config.initContainers:
        initContainers = utils.VolumePatch(config.volumes, [utils.ContainerFron

    # Construct workload attributes.
    workloadAttributes: {str:} = {
        metadata = utils.MetadataBuilder(config) | {
            name = workloadName
        }
        spec = {
            replicas = config.replicas
            if config.useBuiltInSelector:
                selector.matchLabels: app.selector | config.selector
            else:
```

后端模型（Platfor Dev）

Deployment

Service

ConfigMap

Database

Monitor

… …

K8s、TF 资源

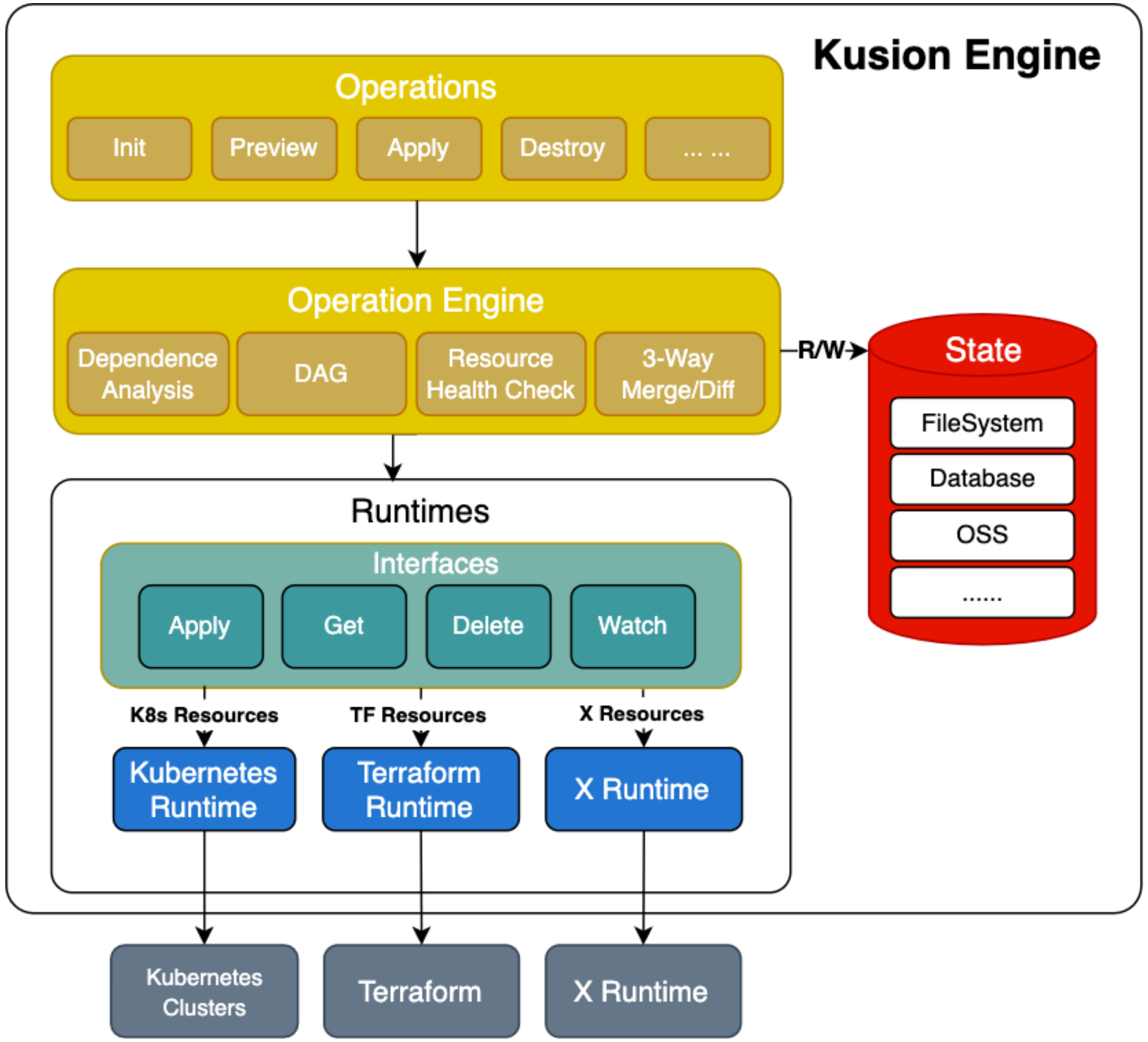
Architecture

Engine：运维操作核心引擎

Operations：为所有 Kusion 运维命令提供资源解析、编排、健康检查等核心能力

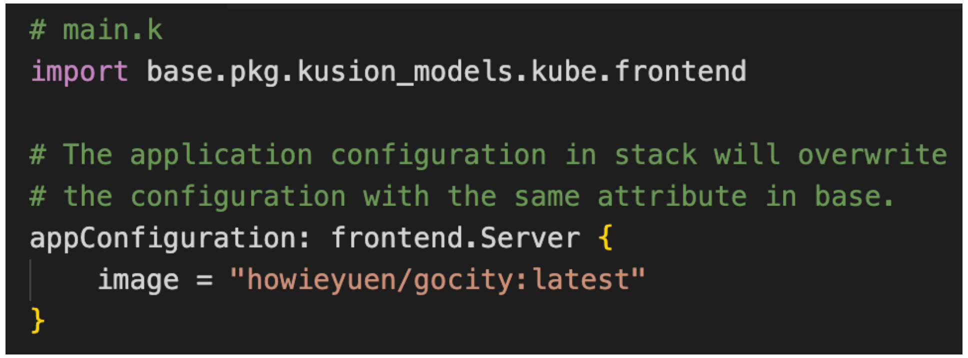
Runtimes：Kusion 管理的基础设施运行时，通过统一的接口与异构基础设施交互

State：集群中真实资源在 Kusion 中的映射

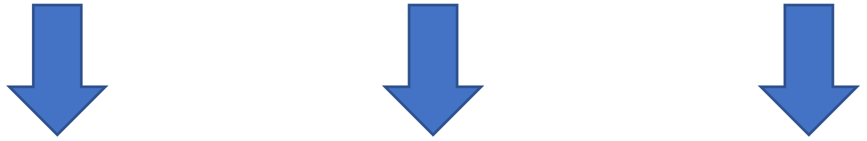


Demo

kusion apply --watch



```
# main.k
import base.pkg.kusion_models.kube.frontend

# The application configuration in stack will overwrite
# the configuration with the same attribute in base.
appConfiguration: frontend.Server {
    image = "howieyuen/gocity:latest"
}
```



Namespace Deployment Service

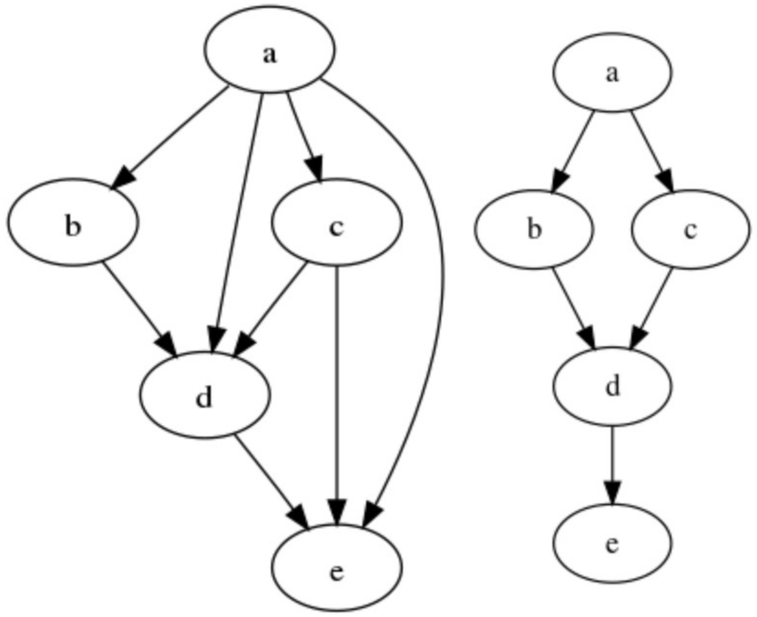
```
➜  dev git:(main) ✗ kusion apply --watch
```

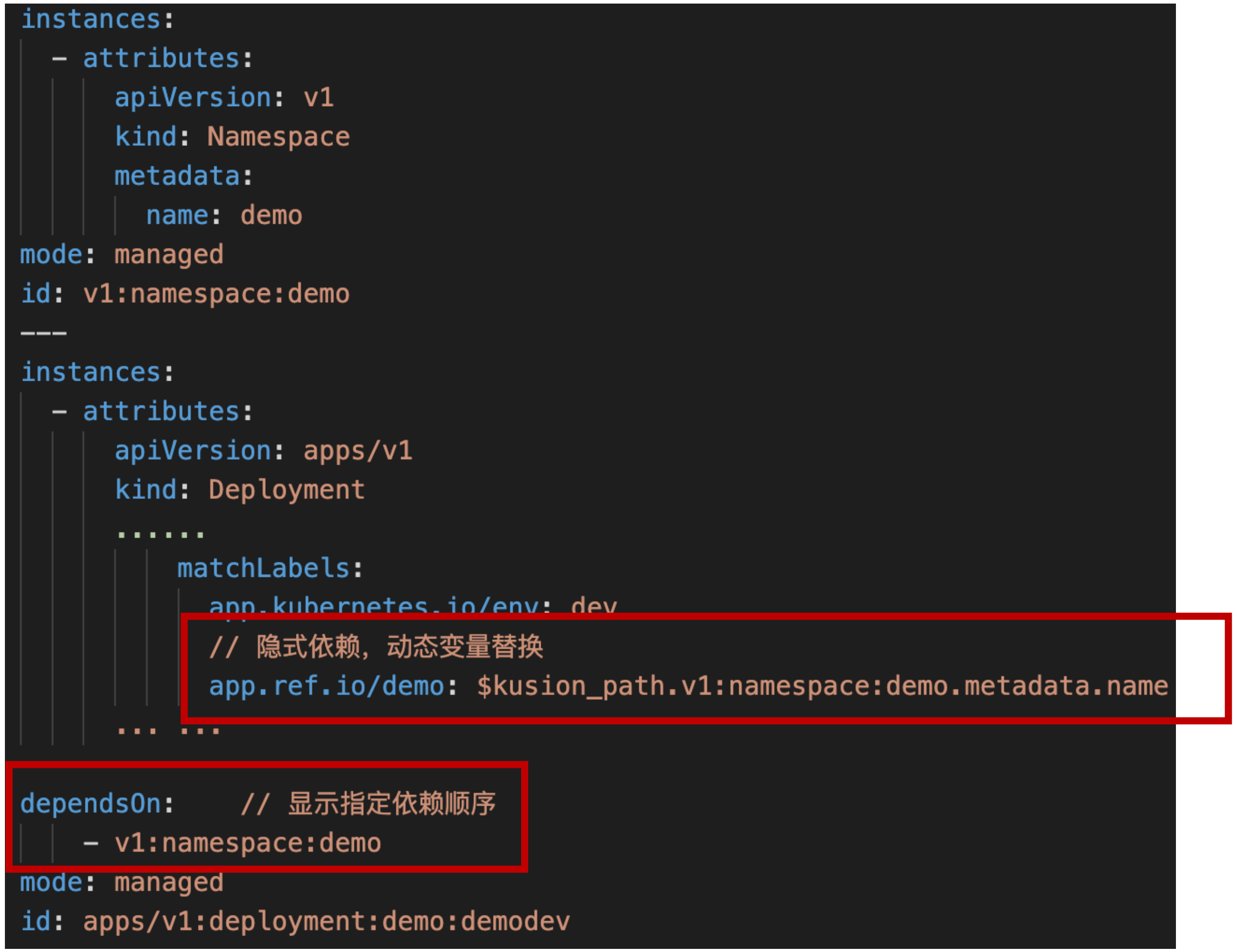
https://kusionstack.io/docs/user_docs/getting-started/usecase

Key technology

资源依赖分析与执行

- 显示依赖
- 隐式依赖
- DAG (Directed Acyclic Graph)
- [Transitive reduction]



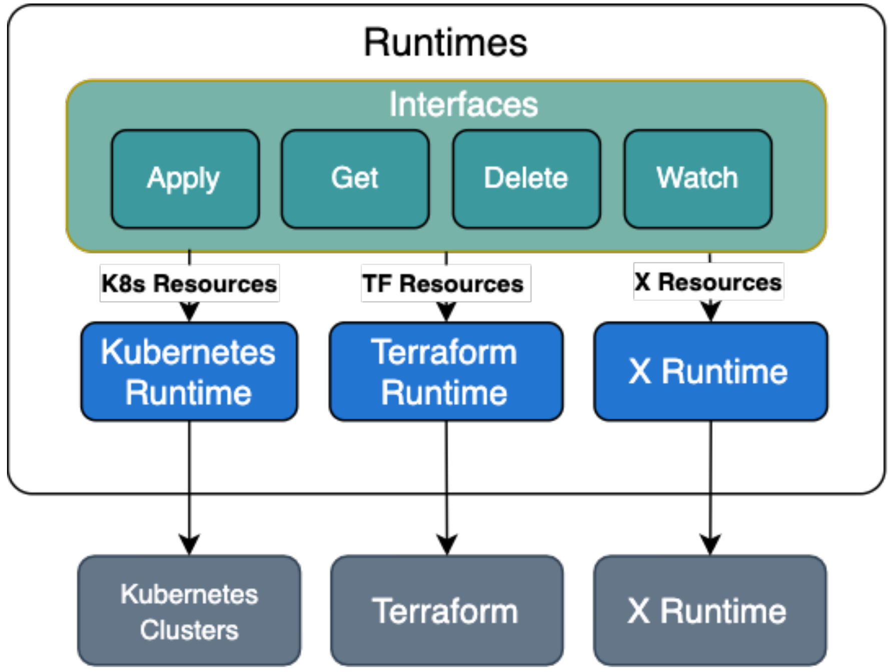


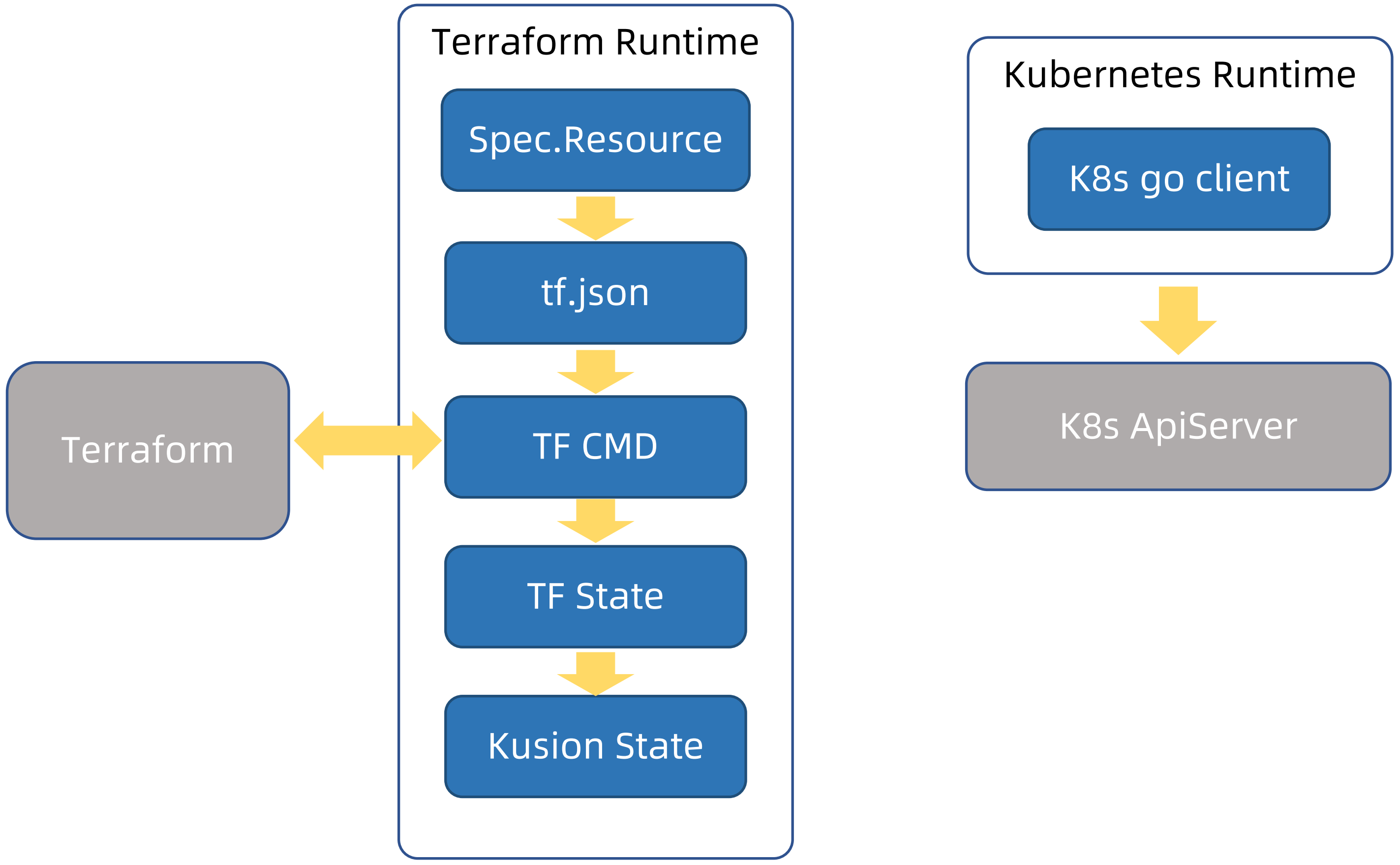
```yaml
instances:
  - attributes:
      apiVersion: v1
      kind: Namespace
      metadata:
        name: demo
mode: managed
id: v1:namespace:demo
---
instances:
  - attributes:
      apiVersion: apps/v1
      kind: Deployment
      ......
        matchLabels:
          app.kubernetes.io/env: dev
          // 隐式依赖，动态变量替换
          app.ref.io/demo: $kusion_path.v1:namespace:demo.metadata.name
      ... ...

dependsOn:    // 显示指定依赖顺序
    - v1:namespace:demo
mode: managed
id: apps/v1:deployment:demo:demodev
```

Key technology

异构 Runtime 管理

- 统一管理接口
- 混合资源编排
- K8s 与 Terraform 适配

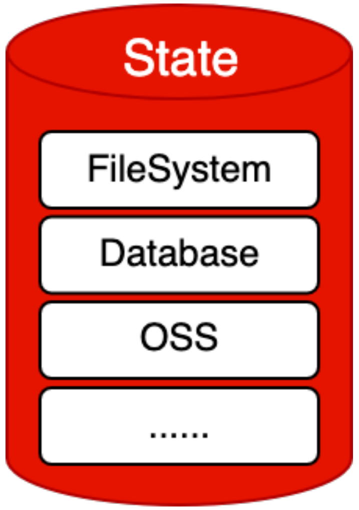




Key technology

State 管理

- 多元存储介质适配
- State 存在的必要性
- 3-way live diff 算法



Without State

Step	Operation	Konfig	Cluster
1	Create	A, B	A', B'
2	Delete B	A	?

With State

Step	Operation	Konfig	Cluster	State
1	Create	A, B	A', B'	A', B'
2	Delete B	A	A'	A'

3-way live diff

prior(State)	plan	live	action
1	1	1	update
1	1	0	update
1	0	1	delete
1	0	0	delete, shouldn't happen
0	1	1	update
0	1	0	create
0	0	1	never reach
0	0	0	never reach

Practice in AntGroup

1K/day	10K+/day	1 : 9	60K+
Pipelines	KCL Compilations	Plat : App Dev	Commits
~400	1500+	~600K	3M+
Contributors	Projects	KCL Codes	YAML

Tech Roadmap

KCL

- More Friendly for Dev
- Wider Ecological Integration
- Powerful Lang & Compiler Capabilities
- Advanced Technology Exploration

v0.4.3 — 2022.9

- Lang Simplification Stage 1
- KCL APIs by Rust
- Completely KCL Tools Support: lint, test, ...
- MThe Compiler Natively WASM execution

v0.5 — 2022.12

- Compiler Decorator Extension
- Policy & Flow Capability Enhancement
- Model Registry & Package Management
- More LSP Based IDEs
- Common Domain Language Programming Framework : Compiler-Base Stage 1

v0.6 — 2023.3

- Lang Simplification Stage 2
- Reverse type inference
- Incremental compilation
- Multi Runtime/Backend

v0.7 — 2023.6

- CFG-Based KCL IR
- Garbage collector
- JIT Compiler
- Compiler-Base Stage 2

Kusion & Konfig

v0.7 — 2022.9

- **Kusion (Resource):** Hybrid resource operation like Terraform and Kubernetes in an unified way
- **Kusion (Resource):** Kubernetes native resource health check
- **Security :** Kusion E2E test framework

v0.8 — 2022.12

- **Konfig (Model):** Support Aliyun ACK, ASM, Prometheus
- **Konfig (Toolbox):** Structure validation
- **Kusion (Resource):** Customimze resource health check
- **Security :** KCL Secret Management
- **IDE:** Kusion Operations Integration

v0.9 — 2023.3

- **Konfig (Model):** Support AWS EKS, App Mesh, AMP
- **Konfig (Toolbox):** Dependency analysis
- **Kusion (Operation):** Advanced workflow
- **Security:** Third-party KMS integration

v0.10 — 2023.6

- **Konfig (Model):** Support Aliyun ECS, SLB, RDS
- **Konfig (Toolbox):** Pipeline Notification
- **Kusion (Operation):** Progressive rollout
- **Kusion (Operation):** Login identity
- **Kusion (Operation):** Pre/Post Hook
- **Kusion (Operation):** Operation REST

Welcome to join us

Fork me on GitHub

- Web Site
  - https://kusionstack.io/

- Source Code
  - https://github.com/KusionStack/kusion
  - https://github.com/KusionStack/KCLVM
  - https://github.com/KusionStack/konfig
  - https://github.com/KusionStack/community

- Contact
  - https://github.com/KusionStack/community#contact

- Twitter
  - @KusionStack

蚂蚁集团 PaaS 核心团队

Thank you

KusionStack Team