KCL: A Declarative Language for Large-scale Configuration and Policy Management

KCL 配置策略语言

徐鹏飞（Peefy）

Ant Group

Mar. 2023

Agenda

01 Background

02 Design

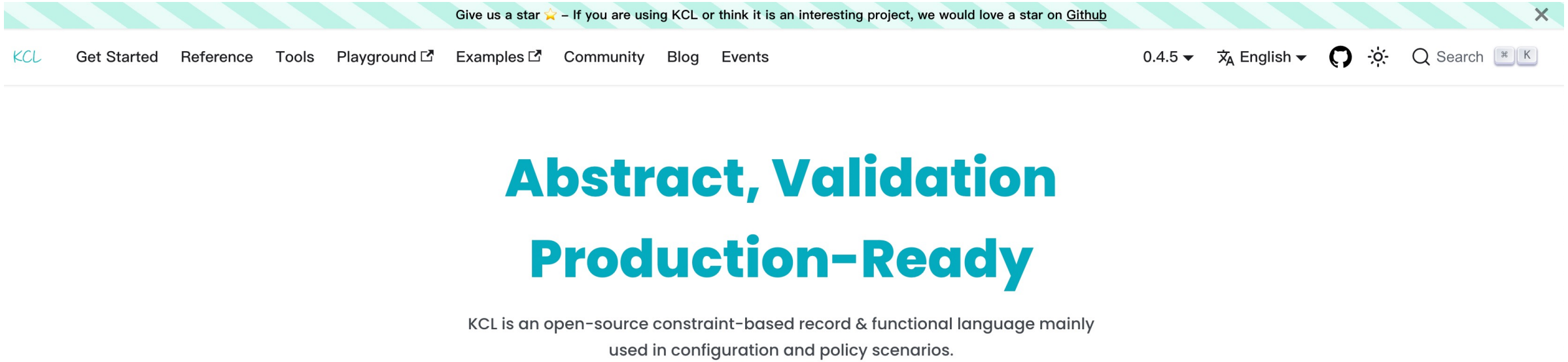
03 Scenarios

04 Evaluation

Background

01

About Me



- **徐鹏飞（Peefy）**
- **KCL Maintainer**

- https://github.com/Peefy
- **Repo**: https://github.com/KusionStack/KCLVM
- **Website**: https://kcl-lang.io/

Why KCL

KUSION STACK

- **Hide infrastructure and platform details to reduce the burden of developers.**
  - Abstraction
  - Solve issues on YAML/Template bloat
  - Language enhancement: logic, type, function and package.

- **Large-scale configuration management without side effects cross teams.**
  - Stability
  - Scalability
  - Automation
  - High performance
  - Package distribute and sharing

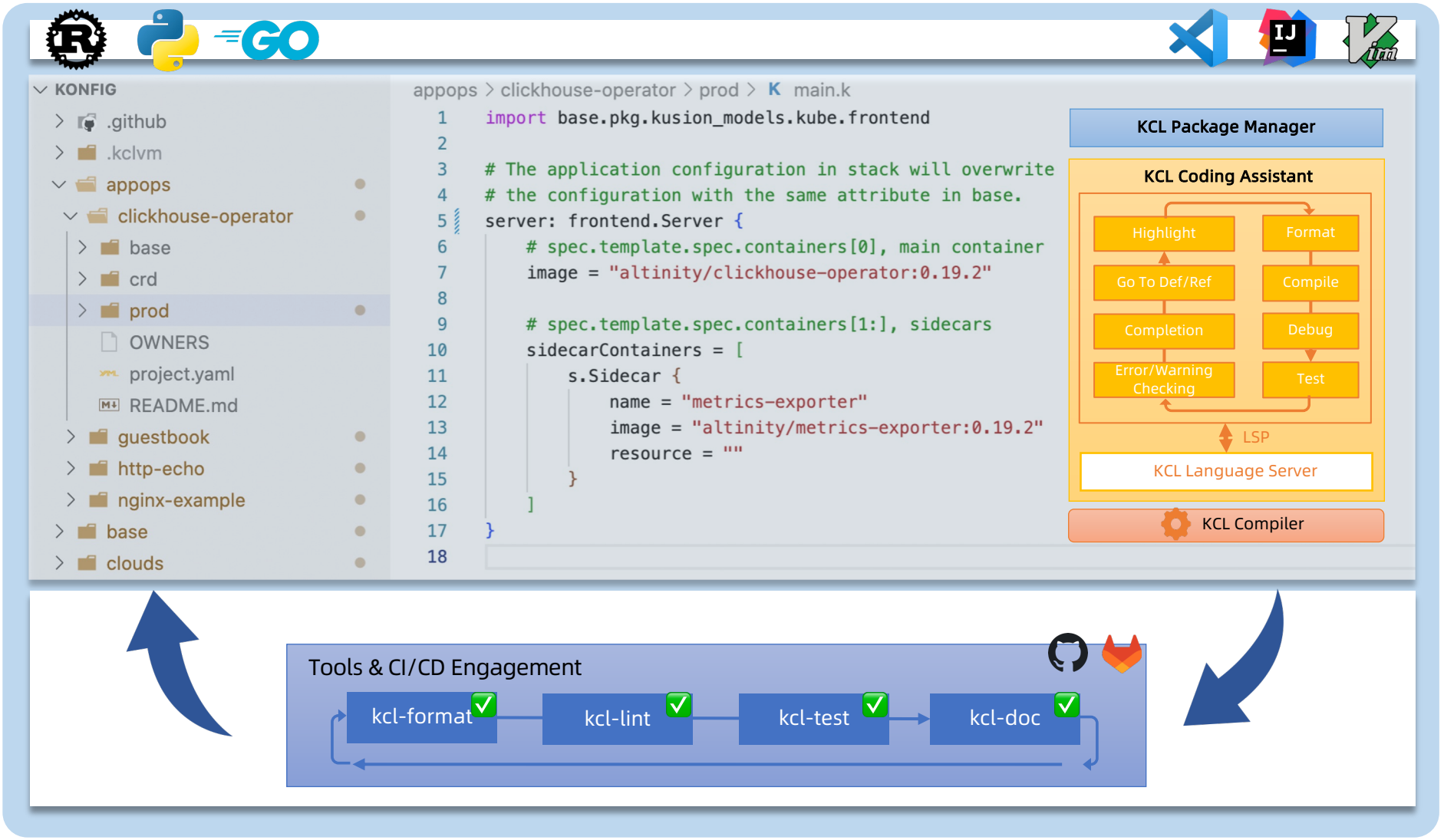
- **Enhancement for configuration tools e.g., Helm, Kustomize.**
  - Validating
  - Editing

Design

02

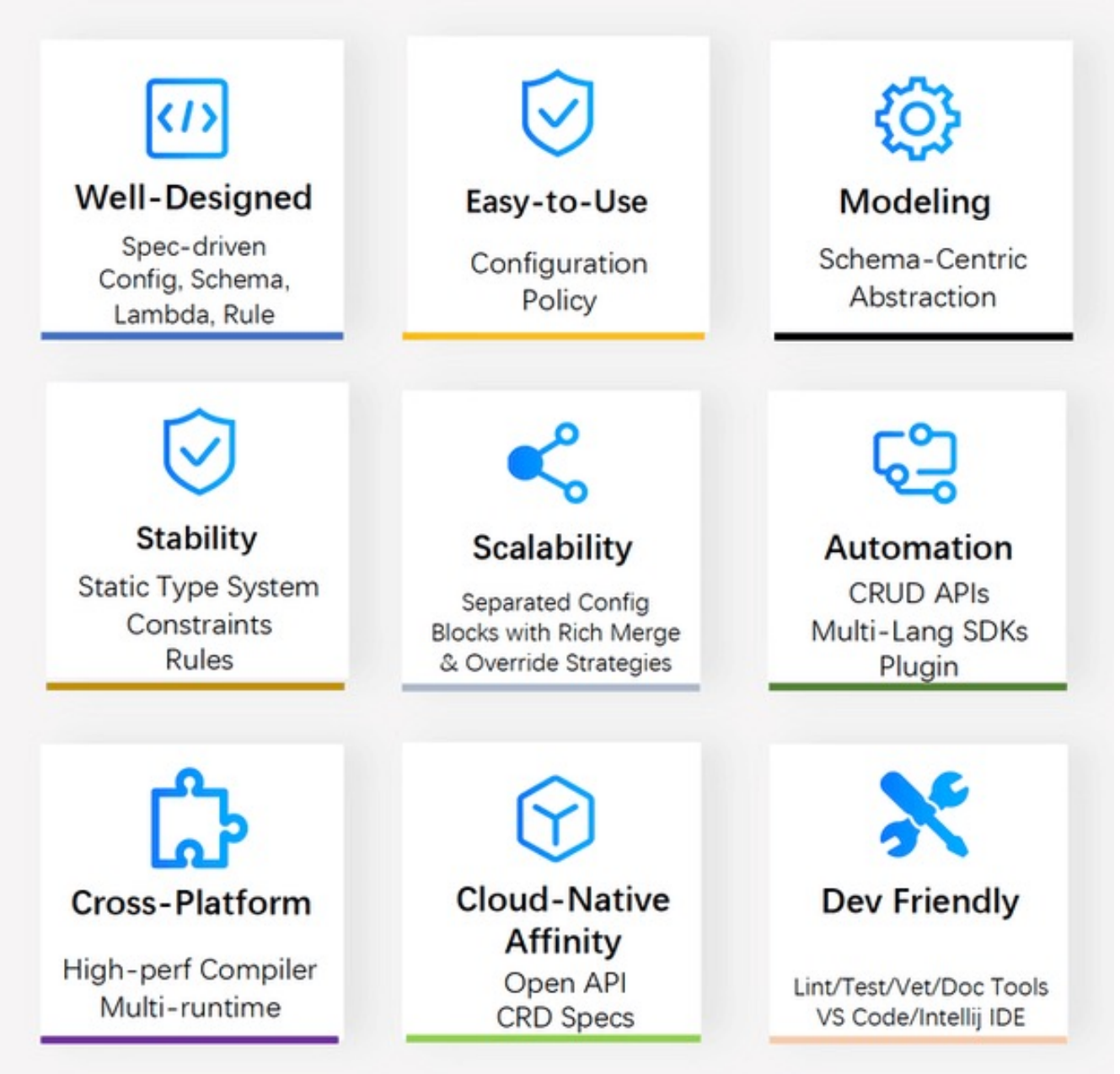
Overview

Language + Tools + IDEs + SDKs + Plugins



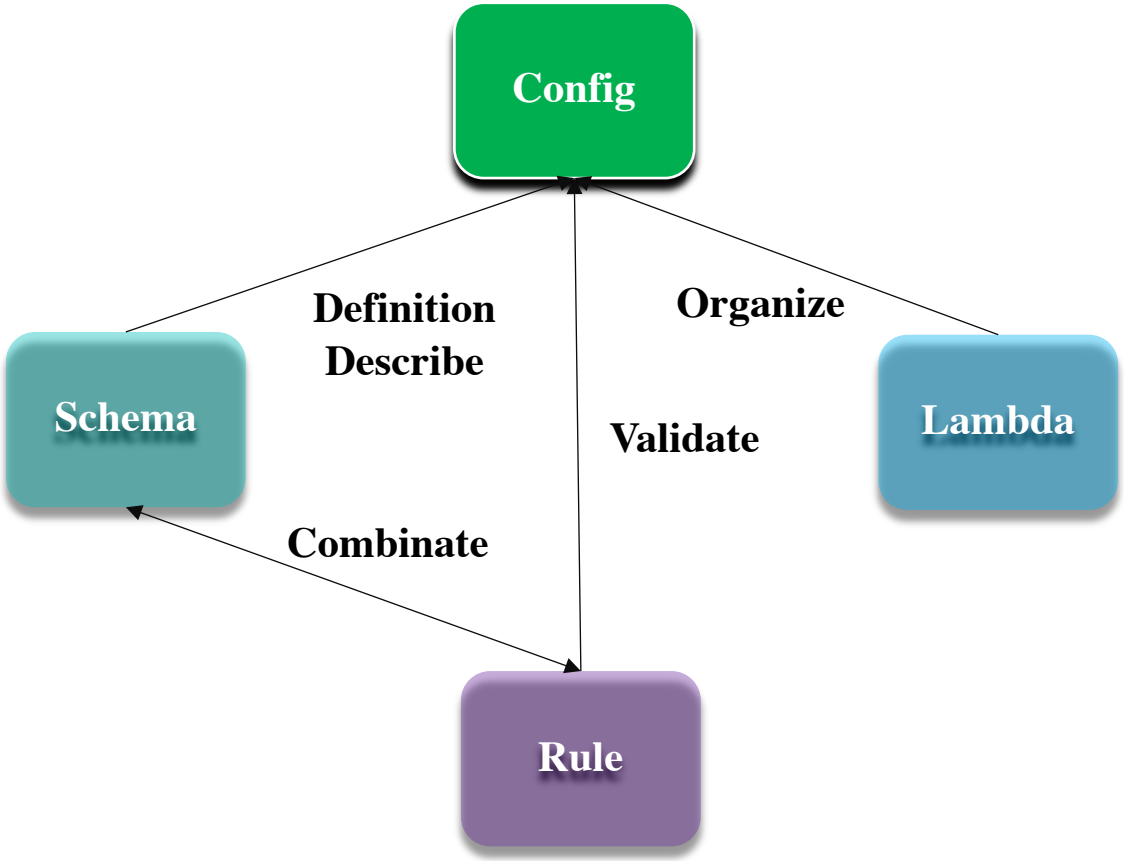
https://github.com/KusionStack/KCLVM

Features


Well-Designed
Spec-driven
Config, Schema,
Lambda, Rule
Easy-to-Use
Configuration
Policy
Modeling
Schema-Centric
Abstraction
Stability
Static Type System
Constraints
Rules
Scalability
Separated Config
Blocks with Rich Merge
& Override Strategies
Automation
CRUD APIs
Multi-Lang SDKs
Plugin
Cross-Platform
High-perf Compiler
Multi-runtime
Cloud-Native
Affinity
Open API
CRD Specs
Dev Friendly
Lint/Test/Vet/Doc Tools
VS Code/Intellij IDE

Concepts

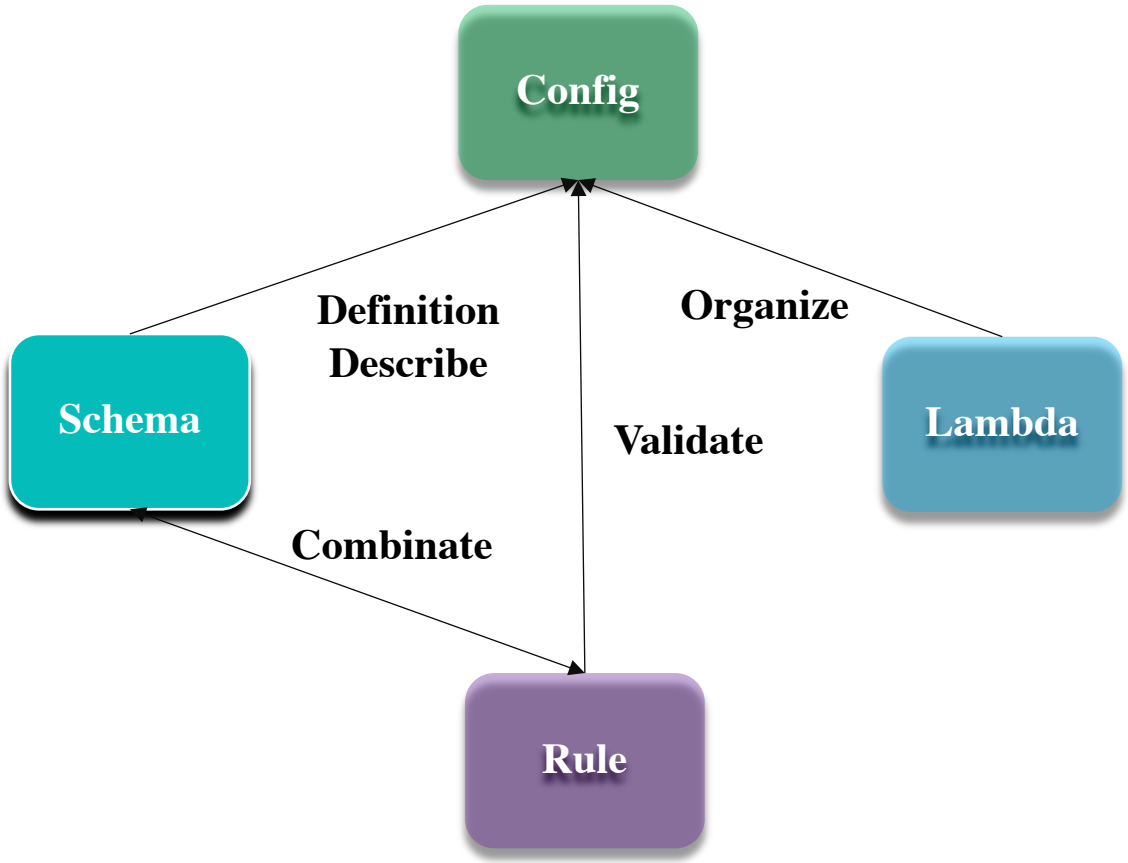
KCL = Config + Schema + Rule + Lambda




```
import kubernetes.core.v1
# Create a kubernetes deployment resource.
deployment = v1.Deployment {
    metadata.name = "nginx"
    metadata.labels.app = metadata.name
    spec = {
        replicas = 3
        selector.matchLabels.app = metadata.name
        template = {
            metadata.labels.app = metadata.name
            spec.containers = [{
                name = metadata.name
                image = "nginx:1.14.2"
                ports = [{ containerPort = 80 }]
            }]
        }
    }
}
```

Concepts

KCL = Config + Schema + Rule + Lambda



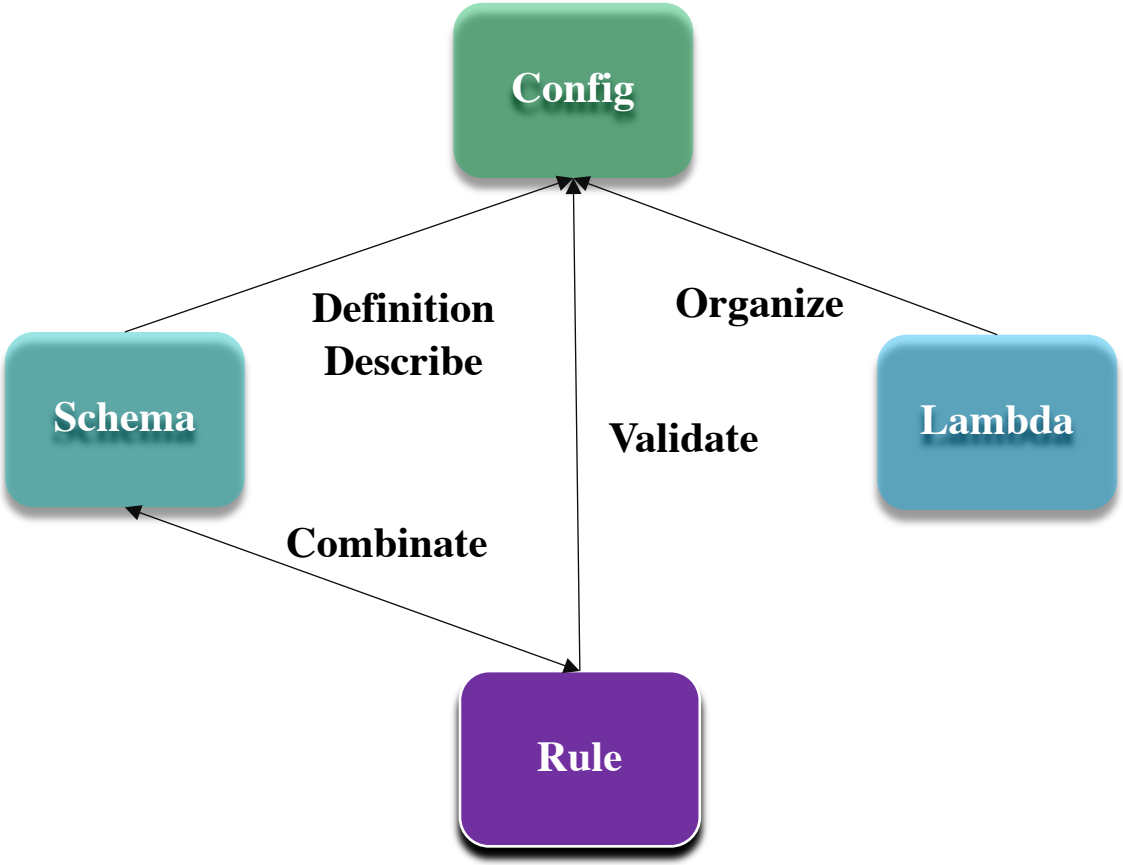

```
import units

type UnitType = units.NumberMultiplier
# Define a schema named Resource with
# three attributes and constraits.
schema Resource:
    cpu: int | UnitType = 1
    memory: UnitType = 1024Mi
    disk: UnitType = 10Gi

    check:
        0 < cpu <= 64
        0 < memory <= 64Gi
        0 < disk <= 1Ti
```

Concepts

KCL = Config + Schema + Rule + Lambda



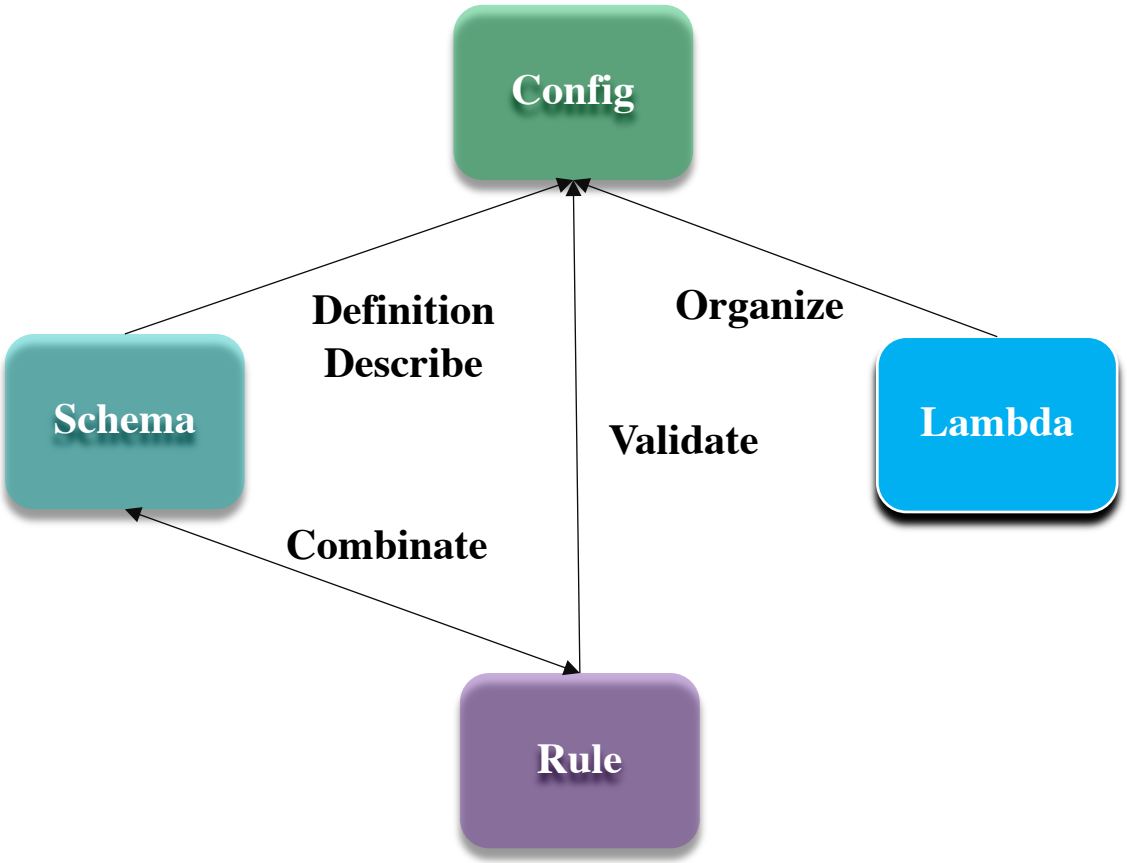

```
data: Data = option("data")
input: Input = option("input")
# Define a RBAC rule
rule Allow:
    any grant in UserIsGranted() {
        input.action == grant.action and input.type == grant.type
    }
    any user in data.user_roles[input.user] { user == "admin" }

rule UserIsGranted:
    [
        grant
        for role in data.user_roles[input.user]
        for grant in data.role_grants[role]
    ]

allow = Allow() or False
```

Concepts

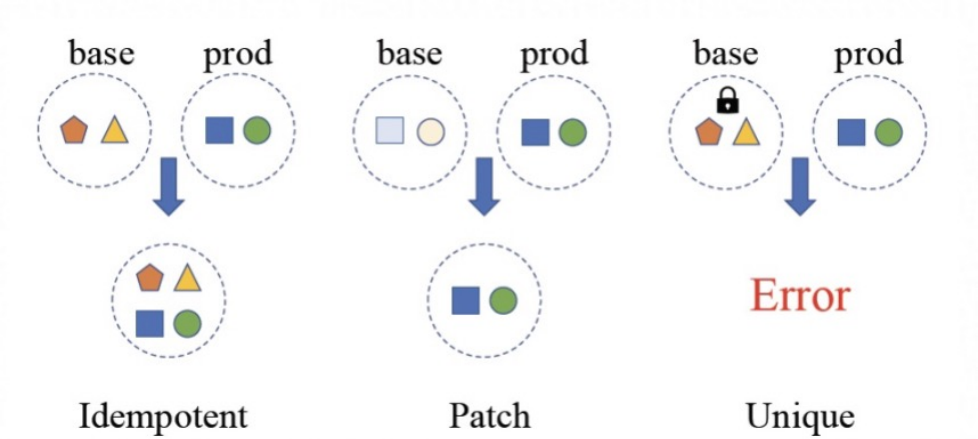
KCL = Config + Schema + Rule + Lambda




```
# Transform input resrource and change annotations.
transformer = lambda res {
    res | {
        metadata.annotations: {
            "managed-by" = "kcl"
        }
    } if res.kind == "Deployment" else res
}

output = [transformer(res) for res in option("input")]
```

Configuration Merge



- base.k

```
# Application Configuration
appConfiguration: frontend.Server {
    # Main Container Configuration
    mainContainer.ports = [
        {containerPort = 80}
    ]
    image = "nginx:1.7.8"
}
```

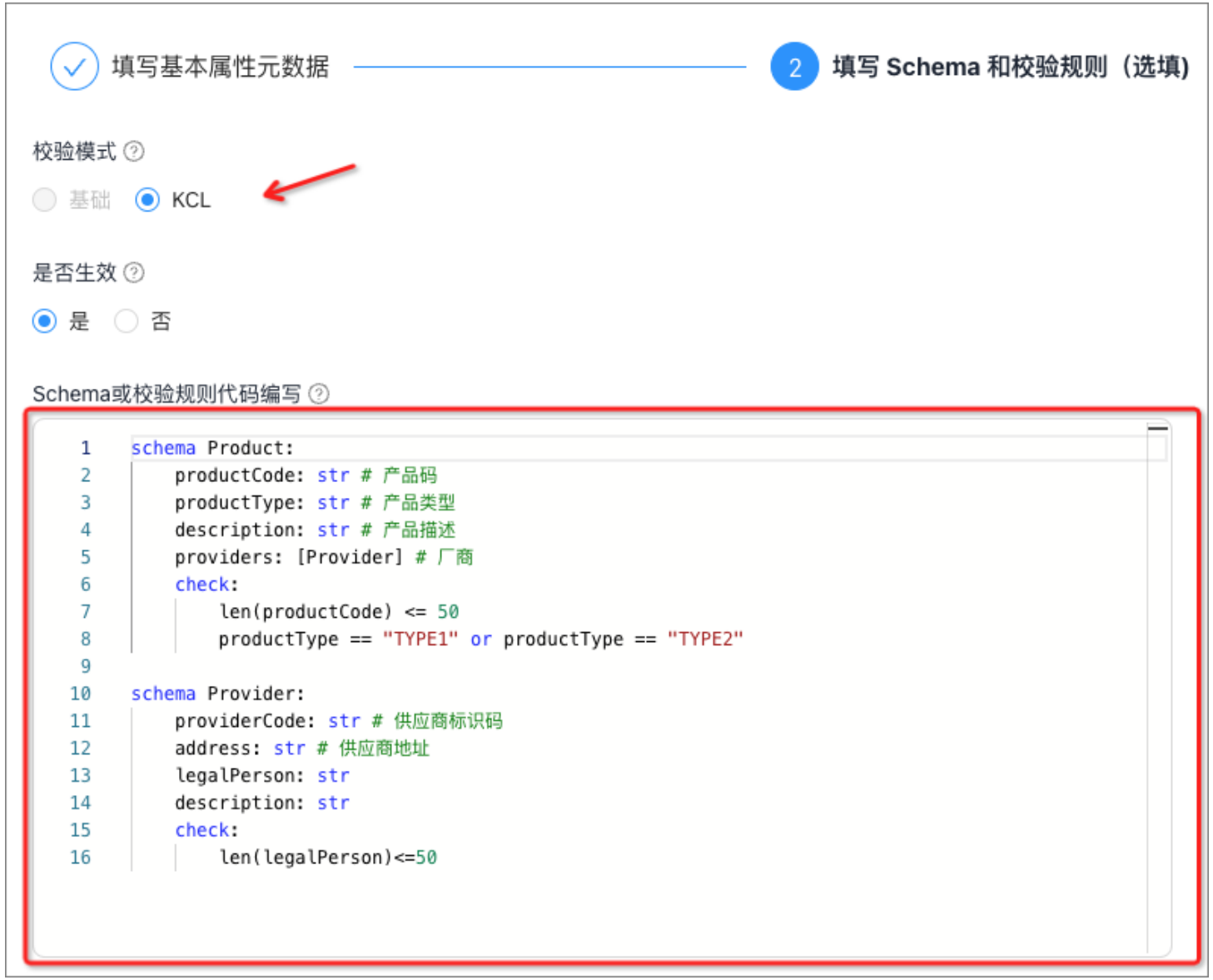
- prod.k

```
appConfiguration: frontend.Server {
    schedulingStrategy.resource = res.Resource {
        cpu = 100m
        memory = 100Mi
        disk = 1Gi
    }
}
```

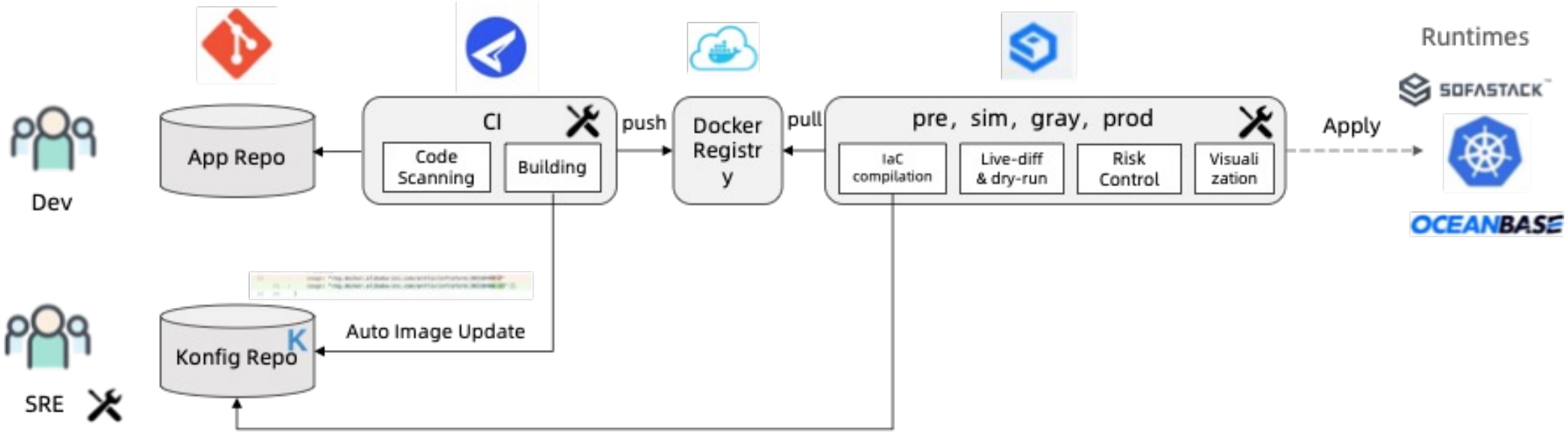
Equivalent code

```
# Application Configuration
appConfiguration: frontend.Server {
    # Main Container Configuration
    mainContainer.ports = [
        {containerPort = 80}
    ]
    image = "nginx:1.7.8"
    schedulingStrategy.resource = res.Resource {
        cpu = 100m
        memory = 100Mi
        disk = 1Gi
    }
}
```

Validation


填写基本属性元数据
2
填写 Schema 和校验规则（选填）
校验模式
基础
KCL
是否生效
是
否
Schema或校验规则代码编写
schema Product:
productCode: str # 产品码
productType: str # 产品类型
description: str # 产品描述
providers: [Provider] # 厂商
check:
len(productCode) <= 50
productType == "TYPE1" or productType == "TYPE2"
schema Provider:
providerCode: str # 供应商标识码
address: str # 供应商地址
legalPerson: str
description: str
check:
len(legalPerson)<=50

Automation



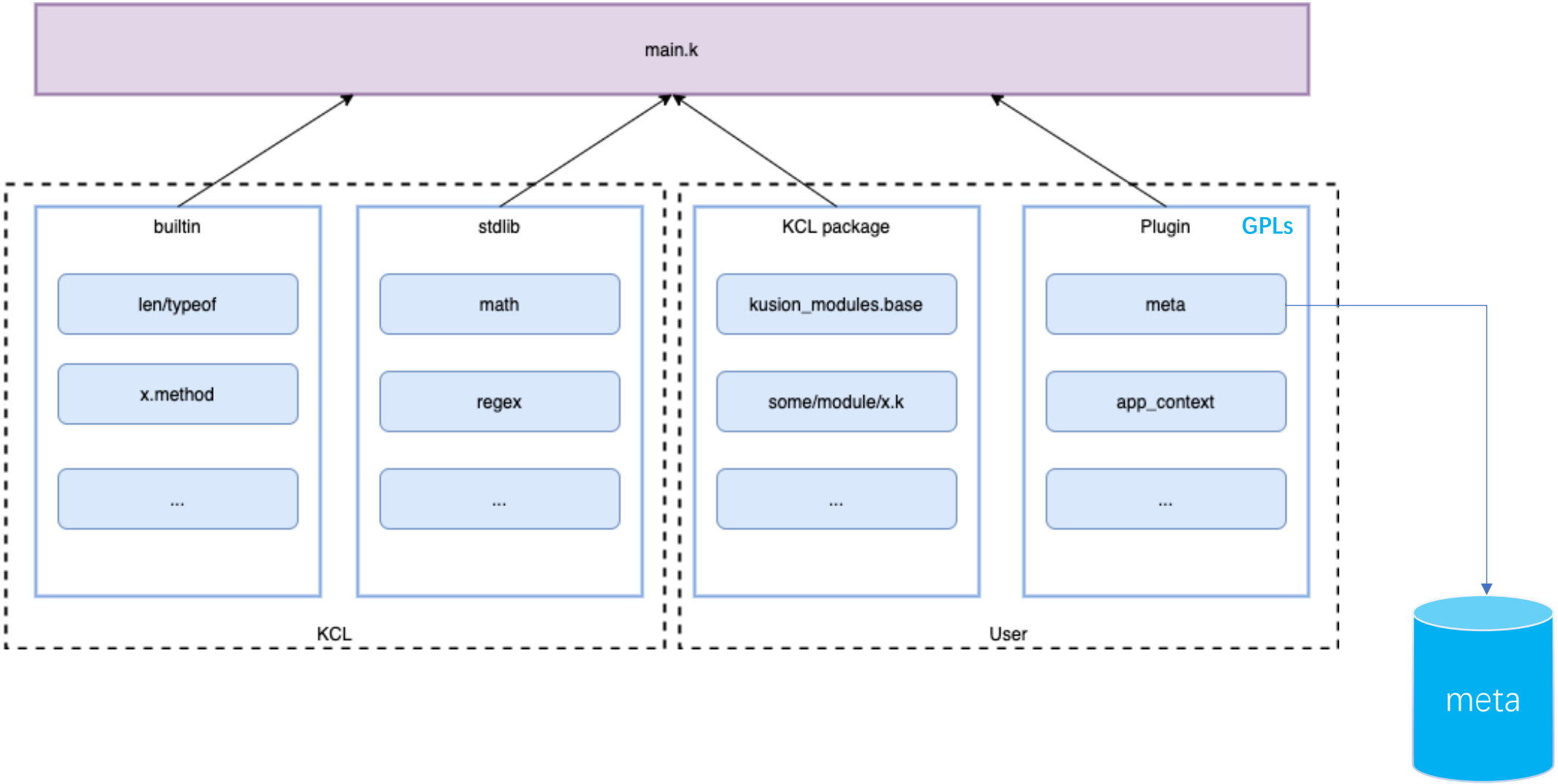

```
import base.pkg.kusion_models.kube.frontend
import base.pkg.kusion_models.kube.frontend.service
import base.pkg.kusion_models.kube.frontend.container
import base.pkg.kusion_models.kube.templates.resource as res_tpl

# Application Configuration
appConfiguration: frontend.Server {
    # Main Container Configuration
    mainContainer = container.Main {
        ports = [
            {containerPort = 80}
        ]
    }
    image = "nginx:1.7.8"
}
```

```
import base.pkg.kusion_models.kube.frontend
import base.pkg.kusion_models.kube.frontend.service
import base.pkg.kusion_models.kube.frontend.container
import base.pkg.kusion_models.kube.templates.resource as res_tpl

# Application Configuration
appConfiguration: frontend.Server {
    # Main Container Configuration
    mainContainer = container.Main {
        ports = [
            {containerPort = 80}
        ]
    }
    image = "nginx:1.7.9"
}
```

Modules

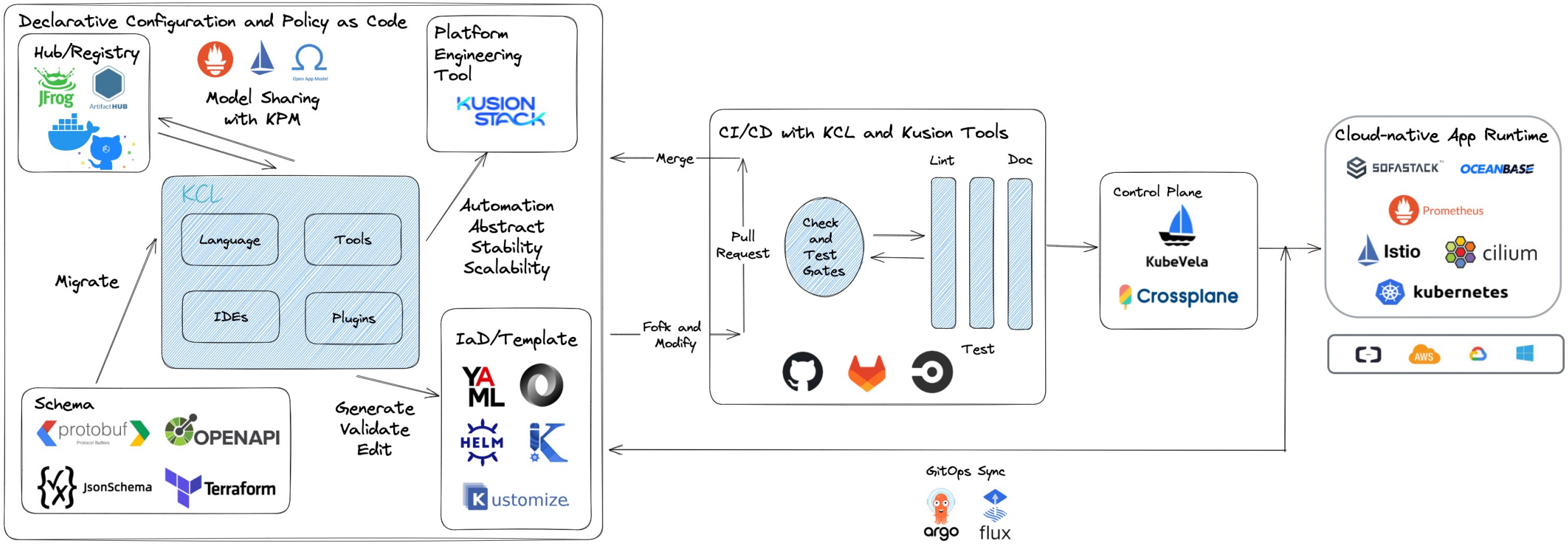


100+ built-in functions, modules with additional KPM package manager tools and plugins

Scenarios

03

Integrations



- **Enhancement for configuration tools:** Abstract, Generate, Validate, Edit
- **Easy schema migration, integration, distribution and usage**
- **Glue of IaC/IaD** through declarative configuration and policy language
- **Automation and GitOps Friendly**

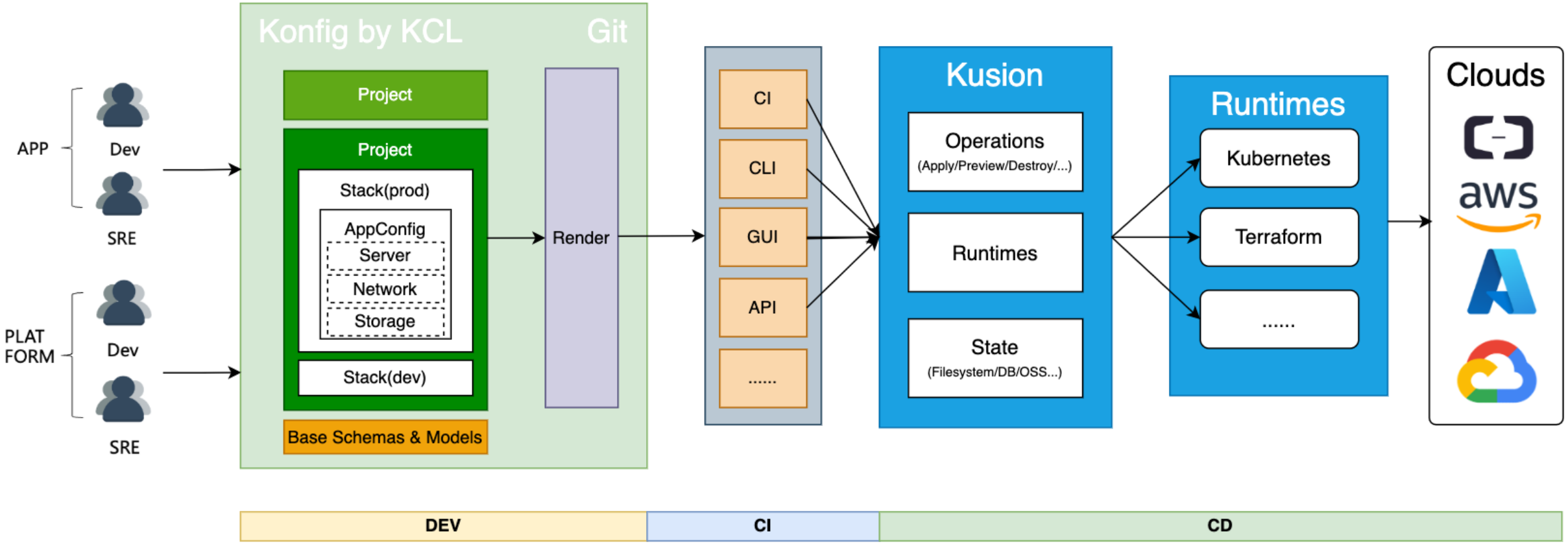
Working with KusionStack

Self-service application deployment tools and workflow for Kubernetes and Clouds

Manage App **from the first code to production-ready** with KCL, Kusion and Konfig

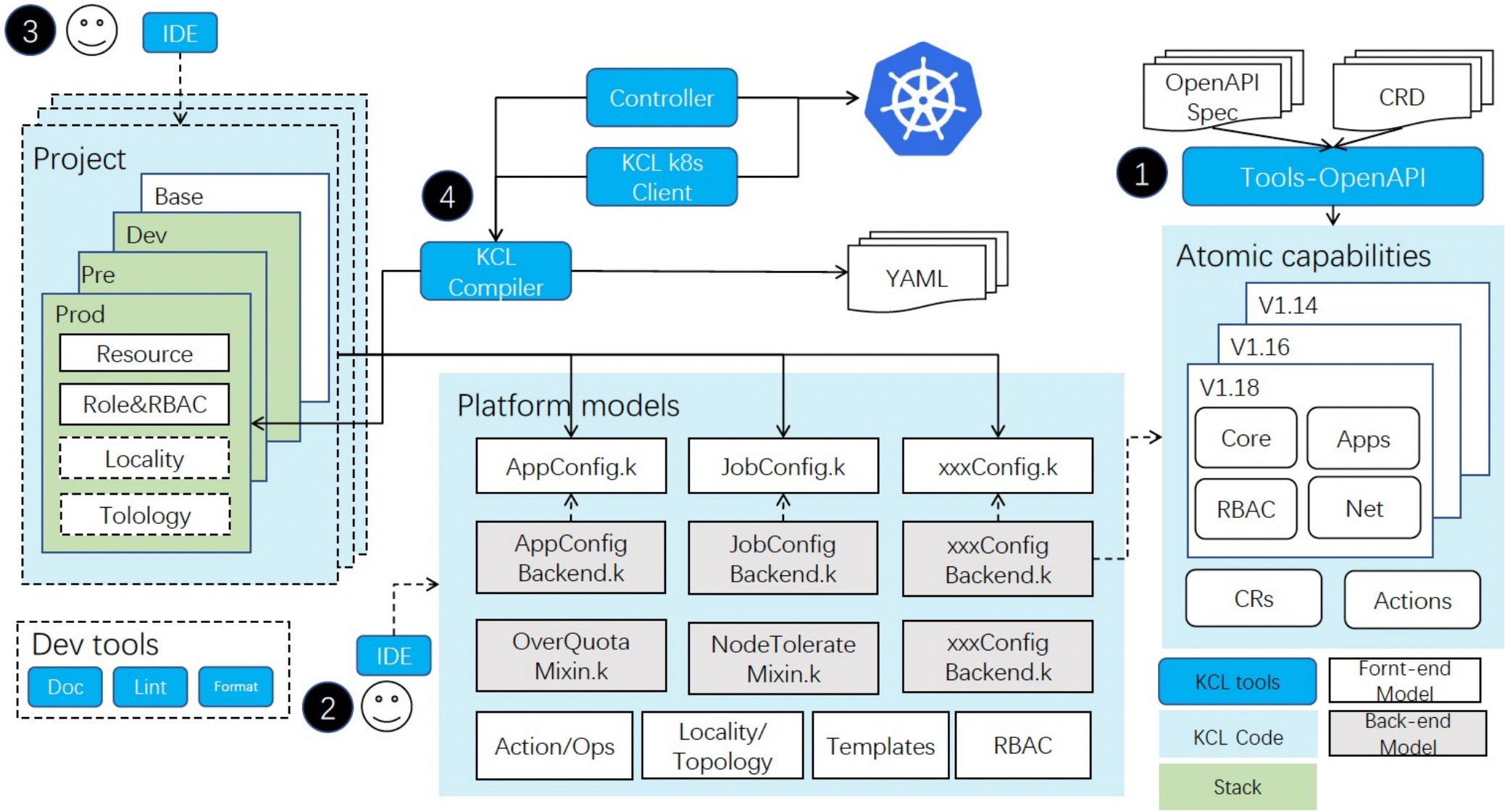
Orchestrate hybrid runtime resources such as Kubernetes, clouds and customized infrastructures in a unified way

Write once, render dynamically, deliver to any cloud

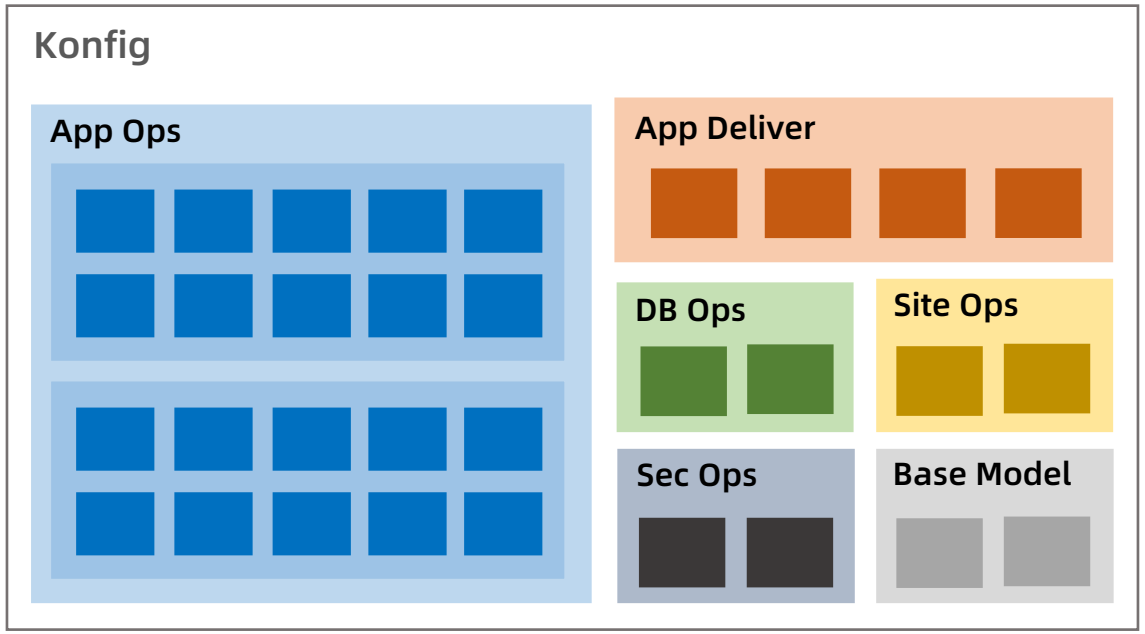


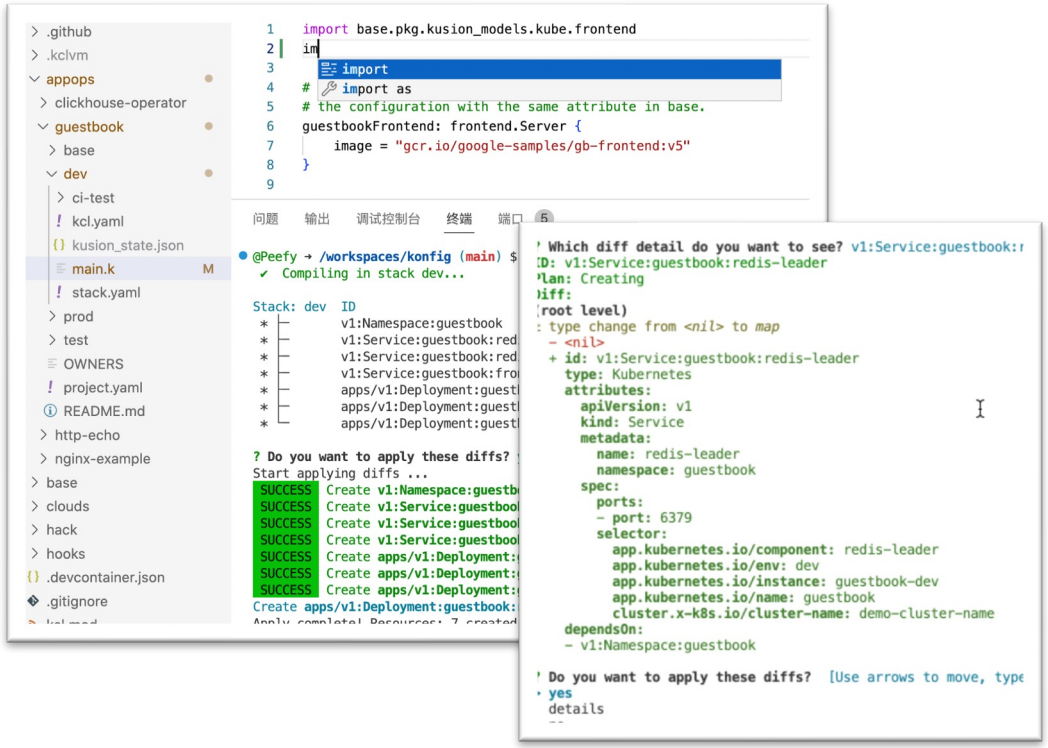
https://github.com/KusionStack/kusion

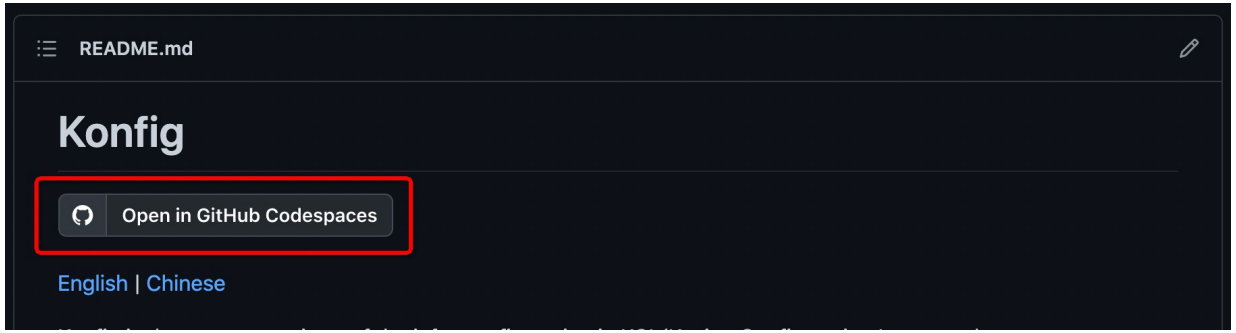
Workflow


3
IDE
Project
Base
Dev
Pre
Prod
Resource
Role&RBAC
Locality
Tolology
Dev tools
Doc
Lint
Format
2
IDE
4
Controller
KCL k8s Client
KCL Compiler
YAML
Platform models
AppConfig.k
JobConfig.k
xxxConfig.k
AppConfig Backend.k
JobConfig Backend.k
xxxConfig Backend.k
OverQuota Mixin.k
NodeTolerate Mixin.k
xxxConfig Backend.k
Action/Ops
Locality/ Topology
Templates
RBAC
OpenAPI Spec
CRD
1
Tools-OpenAPI
Atomic capabilities
V1.14
V1.16
V1.18
Core
Apps
RBAC
Net
CRs
Actions
KCL tools
Fornt-end Model
KCL Code
Back-end Model
Stack

Workspace







https://github.com/KusionStack/konfig

Examples


资源管理器
KONFIG [CODESPACES]
.github
.kclvm
appops
clickhouse-operator
base
crd
prod
ci-test
kcl.yaml
kusion_state.json
main.k
stack.yaml
OWNERS
project.yaml
README.md
guestbook
http-echo
nginx-example
wordpress
base
clouds
hack
hooks
.devcontainer.json
.gitignore
kcl.mod
LICENSE
Makefile
README-zh.md
README.md
大纲
时间线
main.k
appops > clickhouse-operator > prod > main.k
import base.pkg.kusion_models.kube.frontend
# The applicati
# the configura
appConfiguratio
# spec.temp
image = "al
# spec.temp
sidecarCont
s.Sidec
nam
ima
res
转到定义 ⌘F12
转到引用 ⇧F12
快速查看
Find All References ⇧⌥F12
更改所有匹配项 ⌘F2
格式化文档 ⇧⌥F
重构... ^⇧R
共享
剪切
复制
复制为
粘贴
添加到监视
运行到光标处
Kusion: Compile
Kusion: Preview Live Diff and Apply
命令面板... ⇧⌘P
问题
输出
调试控制台
bash
Welcome to Codespac
ined by the prebuild configuration for your repository. Learn more: https://aka.ms/ghcs-prebuild.
To explore VS Code
d Palette (Cmd/Ctrl + Shift + P)
Edit away, then run
nning in the browser.
@Peefy → /workspaces/k
阿里语雀

Evaluation

04

Related Works

Pros.

- Easy to write and read
- Rich multi-language API
- Various Path Tools

Cons.

- Redundant information
- Insufficient functionality e.g. abstraction, constraint, …

Tech.

- JSON
- YAML

Product

- Kustomize
- …

Pros.

- Simple config logic support
- Dynamic argument input

Cons.

- Increase of argument makes it difficult to maintain
- Insufficient functionality e.g. abstraction, constraint, …

Tech.

- Velocity
- Go Template

Product

- Helm
- …

Pros.

- Required programming features
- Code modularity
- Templates & Data abstraction

Cons.

- Insufficient type constraints
- Insufficient restraint ability
- Runtime error

Tech.

- GCL
- HCL
- JSONNET
- …

Product

- Terraform
- …

Pros.

- Rich config constraint syntax
- Unified type & value constraint
- Configuration conflict checking

Cons.

- Difficult to configuration override for multi-environment scenarios
- Runtime checks and limited performance

Tech.

- CUE
- …

Product

- KubeVela
- …

Pros.

- Model-centric & constraint-centric
- Scalability on separated block writing with rich merge strategies
- Static type system & analysis
- High Performance

Cons.

- Expansion of different models requires investment in R&D

Tech.

- **KCL**
- …

Product

- KusionStack
- …

Adopters and Partners

Key Results at Ant Group

15	32	1500+	100+	500+
BG	BU	Projects	Clusters	Committers
1K/day	30K+	100K+	~1M	10K+/day
Pipelines	PRs	Commits	KCL Codes	KCL Compilations

Roadmap

- **v0.4.6 Release**
- Ease of Use and Error Improvement
- Helm KCL Schema Plugin Design
- KPT KCL SDK Deign
- KCL KPM Pre-release (Git Repo, Lock, Dependency management)
- ConfVet Design
- Go SDK: Better Installation, Go Struct/KCL Schema Conv API，List Instance API, etc.
- KCL New IDE Plugin Pre-release
- KCL Playground Code Share

2023.4

- **v0.5.0 Release**
- Improvement of Grammar and Standard Libraries
- KCL Abstract/Constraint Specification Standards and Best Practices Manual
- Helm KCL Schema Plugin Release
- KPT KCL SDK Release
- KCL KPM Release
- KCL New Test Tool Design
- ConfVet Release
- More SDK Support e.g., Java, Node.js
- KCL New IDE Plugin Stable-release (Syntax error recovery, semantic highlighting, semantic completion, etc.)
- More KCL Playground Code Samples

2023.6

- **v0.6.0 Release**
- KCL New UI Design
- Kustomize KRM KCL Plugin Design
- KCL KPM OCI Registry
- KCL New Test Tool Release
- CompilerBase Core Components
- More KCL IDE Support e.g., Vim

2023.9

- **v0.7.0 Release**
- KCL New UI Release
- Kustomize KRM KCL Plugin Release
- KCL KPM S3/OSS
- CompilerBase Grammar and Sematic

2023.12

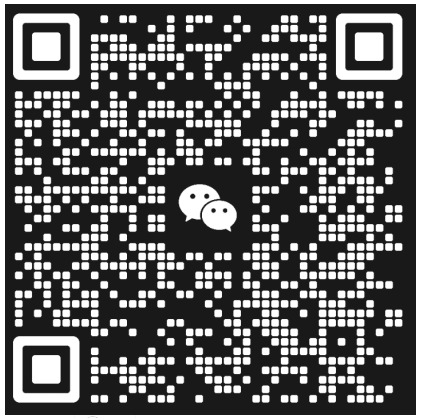
Language
Tool
SDK
Scene & Integration
Technical Subproject
IDE Plugin

https://github.com/KusionStack/KCLVM/issues/29

Resources

- **Website**
  - https://kcl-lang.io/
  - https://kusionstack.io/
- **Repo**
  - https://github.com/KusionStack/KCLVM
  - https://github.com/KusionStack/kusion
  - https://github.com/KusionStack/konfig
- **Community**
  - https://github.com/KusionStack/community#contact
  - https://github.com/KusionStack/community
- **Twitter**
  - @KusionStack

欢迎大家用钉钉扫码
或钉钉搜索：42753001
加入 KusionStack 官方交流群



欢迎大家用微信扫码
加入 KusionStack 官方微信群

KUSION STACK

THANKS