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

KCL 配置策略语言

徐鹏飞 Peefy

AntGroup

Dec. 10, 2022

Agenda

01 Background

02 Design

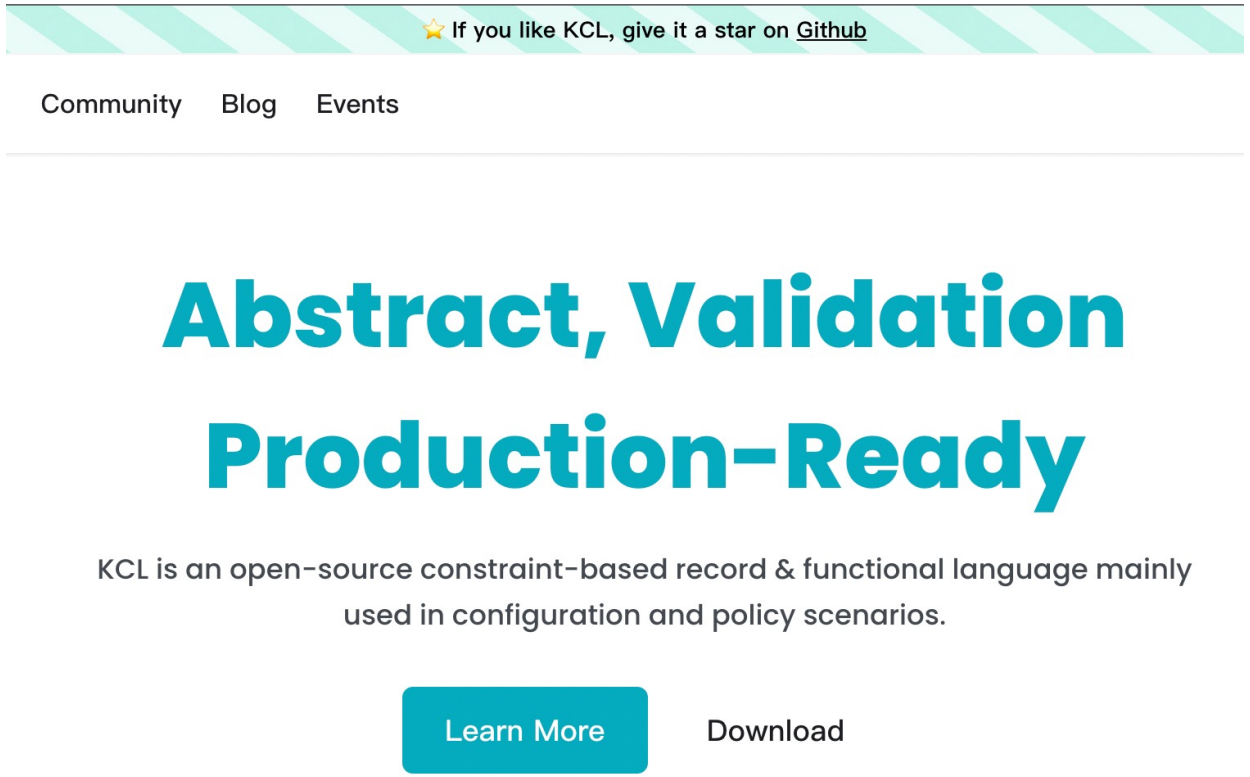
03 Use Cases

04 Evaluation

Background

01

About Me

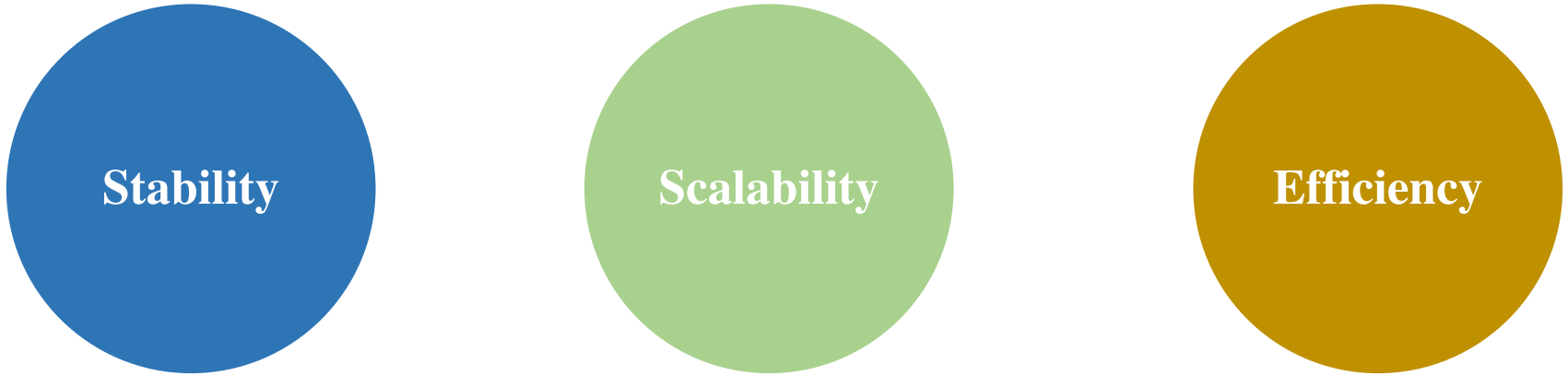


- **徐鹏飞（零执）**
- 蚂蚁集团 Ant Group
- KCL Maintainer

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

Background

Configurations and Policies are important and crucial



Time	Event
2021.07	The Bilibili website in China went down because SLB Lua configuration code fell into an infinite loop with calculation errors
2021.10	KT Company in South Korea suffers major network interruption nationwide due to wrong routing configuration

Why Design KCL

Growing importance of modeling, constraint and scalability

Structured KV

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
- …

Templated KV

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
- …

Programmable KV

Pros.

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

Cons.

- Insufficient type constraints
- Insufficient restraint ability
- Runtime error

Tech.

- GCL
- HCL
- JSONNET
- …

Product

- Terraform
- …

Typed KV

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
- …

Modeled KV

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

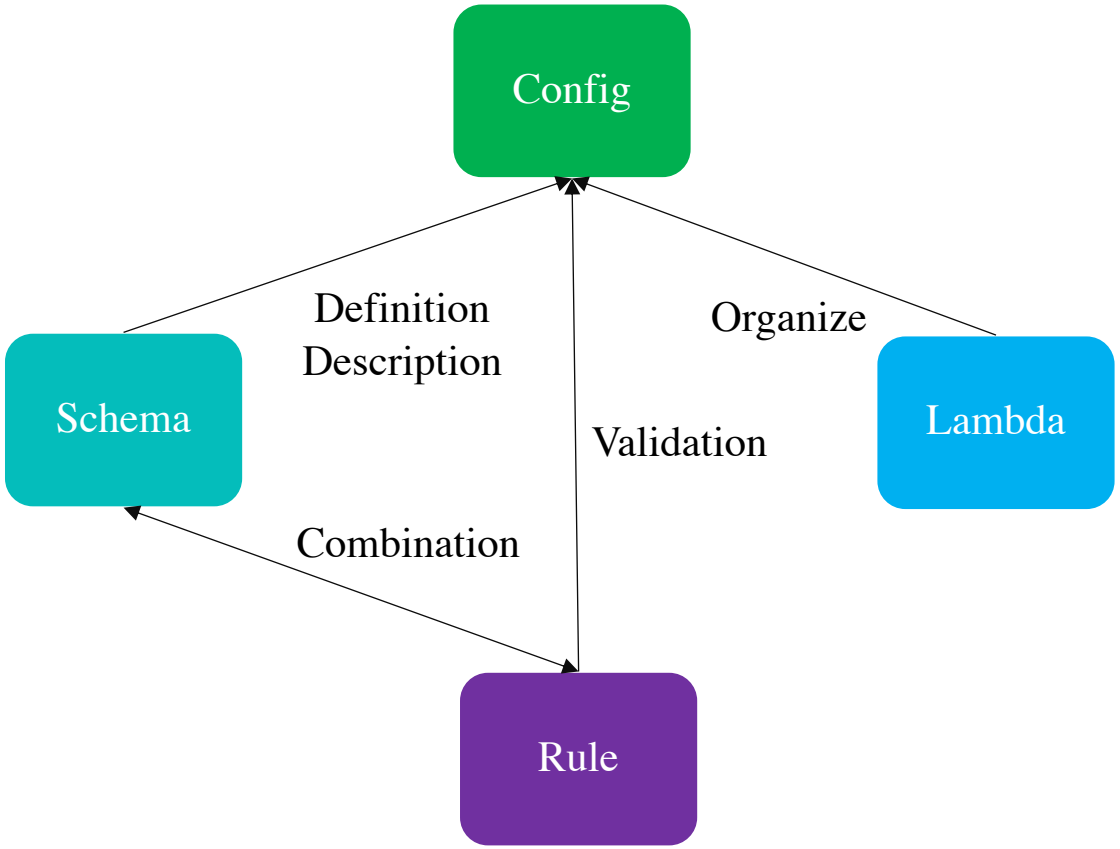
- KusionStack
- …

Design

02

Overview

KCL is an open source constraint-based record & functional language mainly used in configuration and policy scenarios




```
import units

type UnitType = units.NumberMultiplier
sc
```

```
import kubernetes.core.v1

deployment = v1.Deployment {
    metadata.name = "nginx-deployment"
    metadata.labels.app = "nginx"
    spec = {
        replicas = 3
        selector.matchLabels.app = "nginx"
    }
}
```

```
mixin OverQuotaMixin:
    overQuotaToleration = Toleration {
        key = "sigma.ali/is-over-quota"
        operator = "Equal"
        value = "true"
        effect = "NoSchedule"
    } if data.overQuota else Undefined

    overQuotaMatchExpression = NodeSelectorRequirement {
        key = "sigma.ali/is-over-quota"
        operator = "In"
        values = ["true"]
    } if data.overQuota else Undefined
```

https://github.com/KusionStack/KCLVM

Model

KUSION STACK

```
schema App:
    domainType: "Standard" | "Customized" | "Global"
    containerPort: int
    volumes: [Volume]
    services: [Service]

    check:
        1 <= containerPort <= 65535, "containerPort must be between 1 and 65535"
        all service in services {
            service.clusterIP == "None" if service.type == "ClusterIP"
        }
        all volume in volumes {
            volume.mountPath not in ["/", "/boot", "/home", "dev", "/etc", "root"]
        }
```

Abstraction & Validation

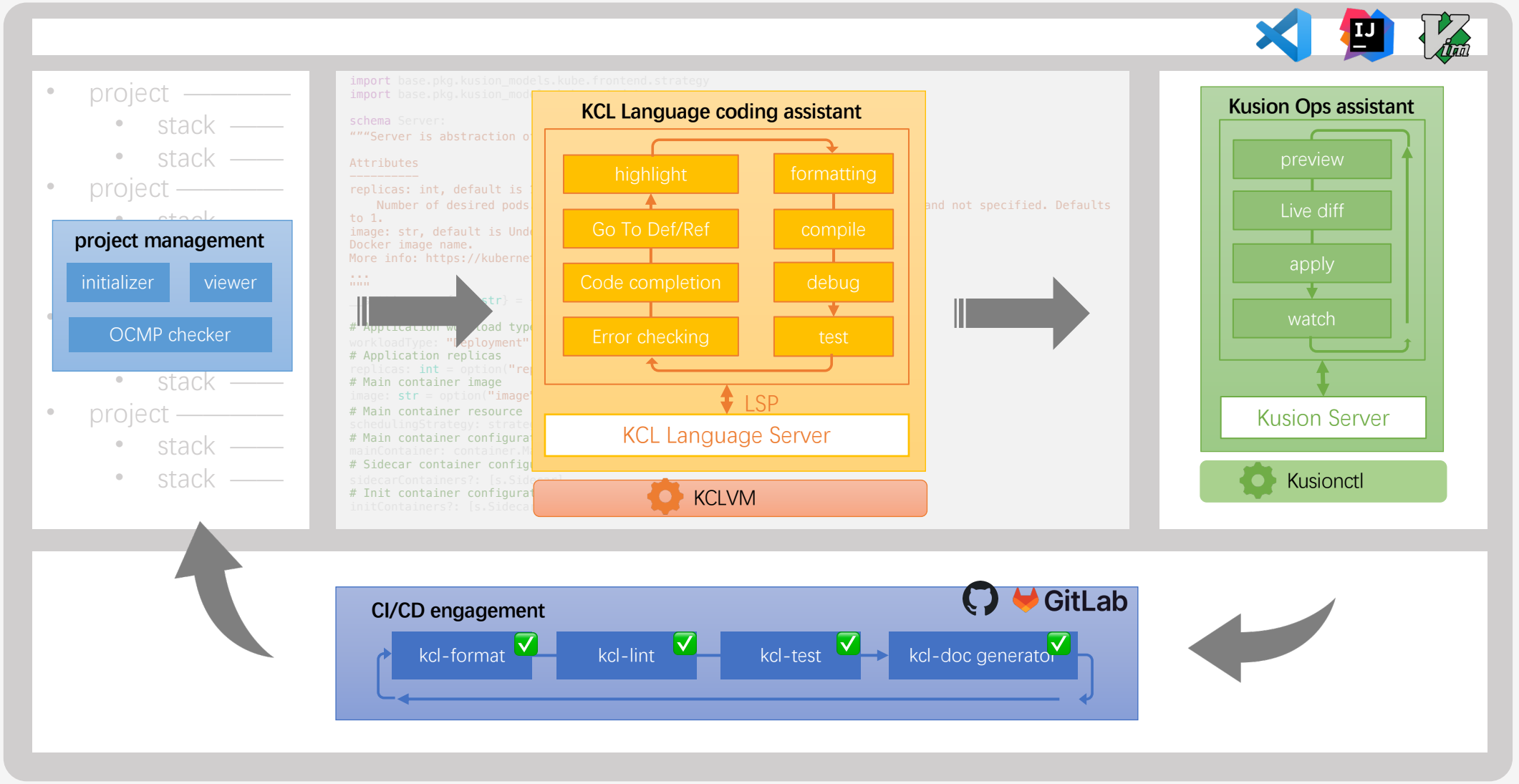
Config

KUSION STACK

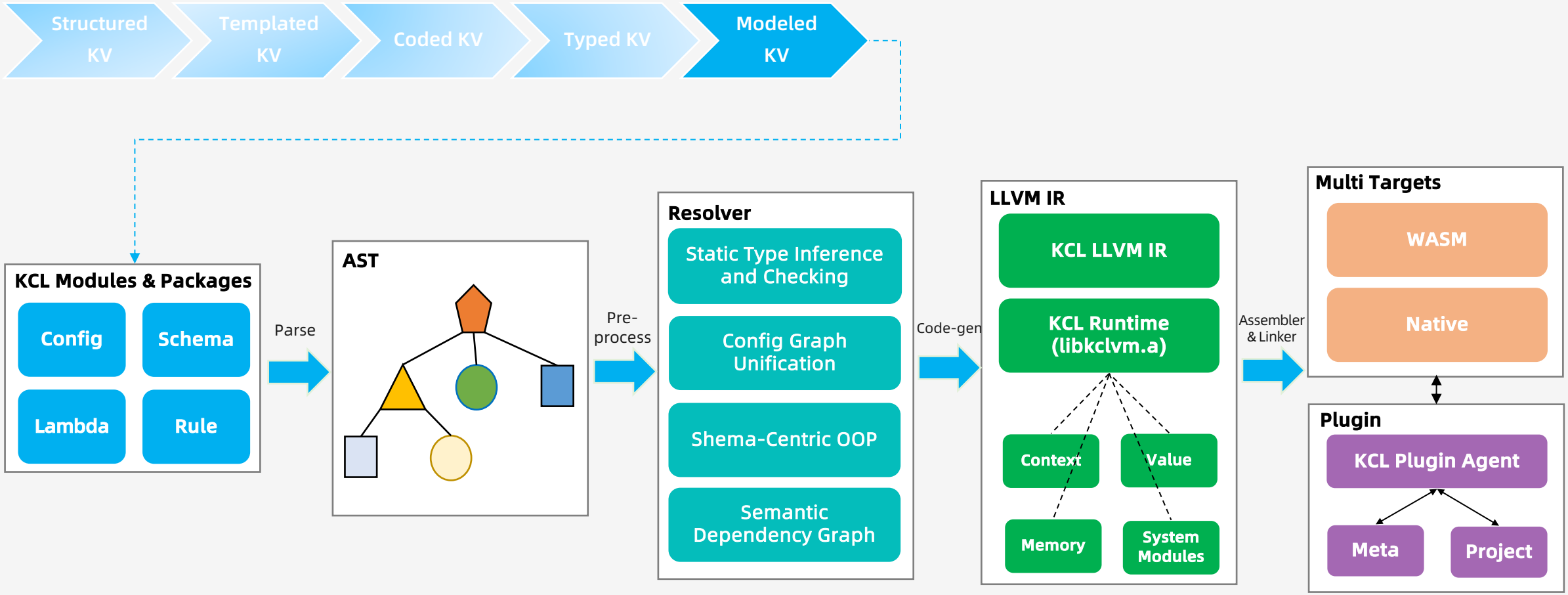
```
import base.pkg.kusion_models.kube.frontend

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

Simple and Core Patter $k\ op\ (T)\ v,$

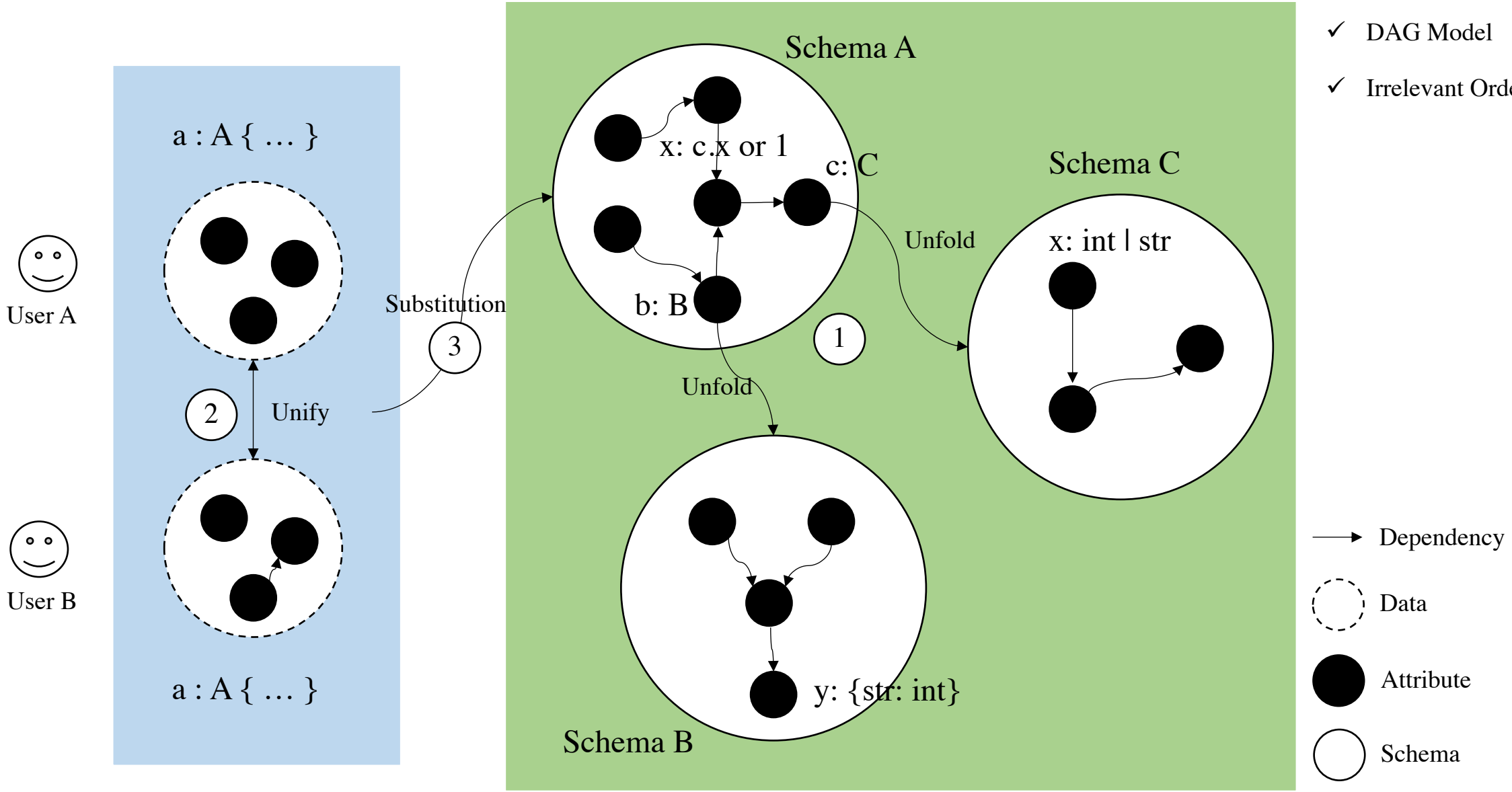
KCL Tools & IDE Workspace



Tech

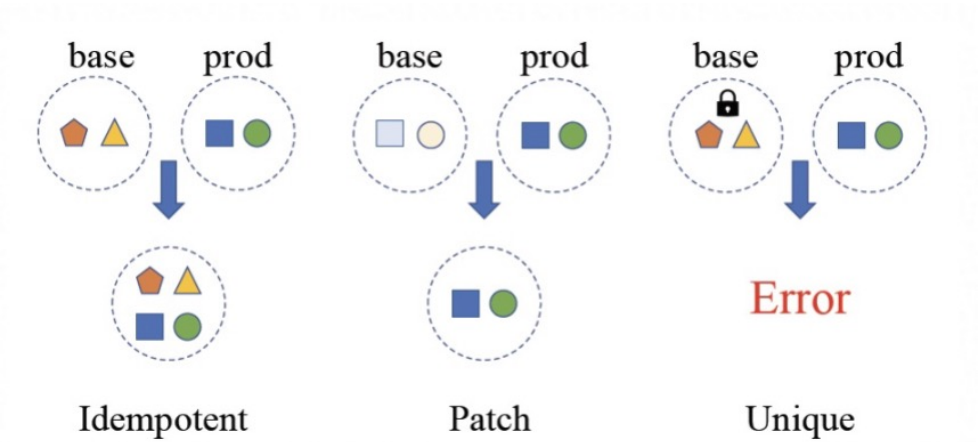

Structured KV
Templated KV
Coded KV
Typed KV
Modeled KV
KCL Modules & Packages
Config
Schema
Lambda
Rule
Parse
AST
Pre-process
Resolver
Static Type Inference and Checking
Config Graph Unification
Shema-Centric OOP
Semantic Dependency Graph
Code-gen
LLVM IR
KCL LLVM IR
KCL Runtime (libkclvm.a)
Context
Value
Memory
System Modules
Assembler & Linker
Multi Targets
WASM
Native
Plugin
KCL Plugin Agent
Meta
Project
Spec Driven
KCL Spec
KCL Multi-Language Spec
KCL OpenAPI Spec

Compilation



https://kcl-lang.github.io/docs/events/2022/kcl_paper

Configuration Merge

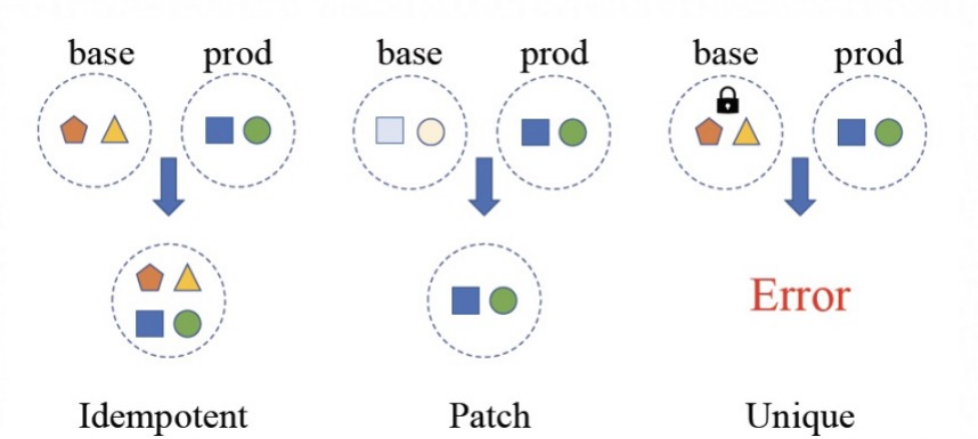


Pattern $$\sigma = \sum_{i=1}^{N} \{k_i, v_i, o_i\},$$

Merge $$\sigma_u \cup \sigma_v = \sum_{j=1}^{N_u+N_v} \{k_j, v_j, o_j\}.$$

Result $$\sum_{k=1}^{M} \{k_s, v_k, o_k\} = \{k_s, \{v_1, o_1\} \oplus \{v_2, o_2\} \oplus \ldots \oplus \{v_M, o_M\}\},$$

Configuration Merge



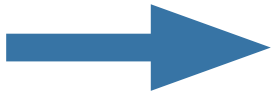
- base.k

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

- prod.k

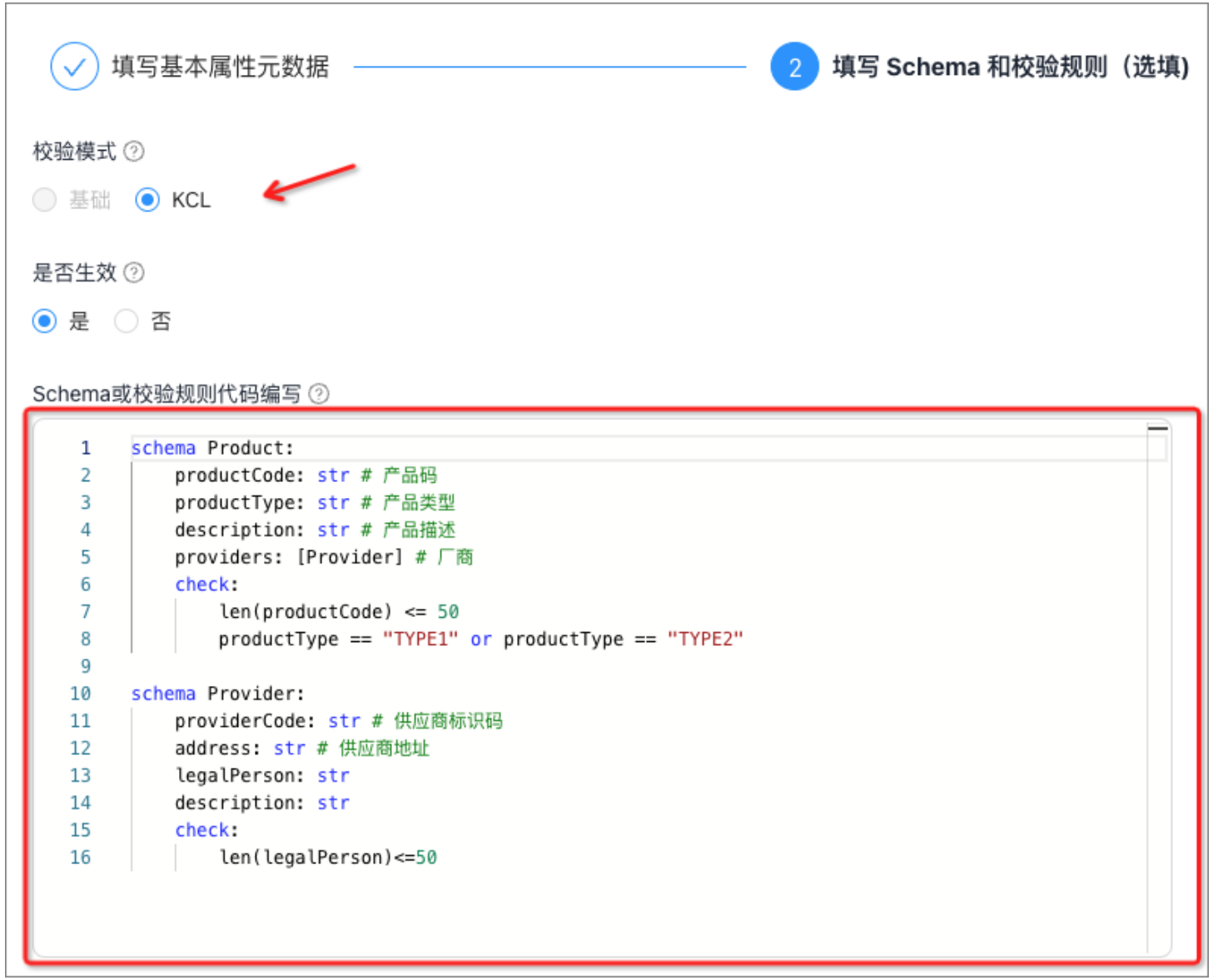
```
appConfiguration: frontend.Server {
    schedulingStrategy.resource = res.Resource {
        cpu = 100m
        memory = 100Mi
        disk = 1Gi
    }
}
```

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

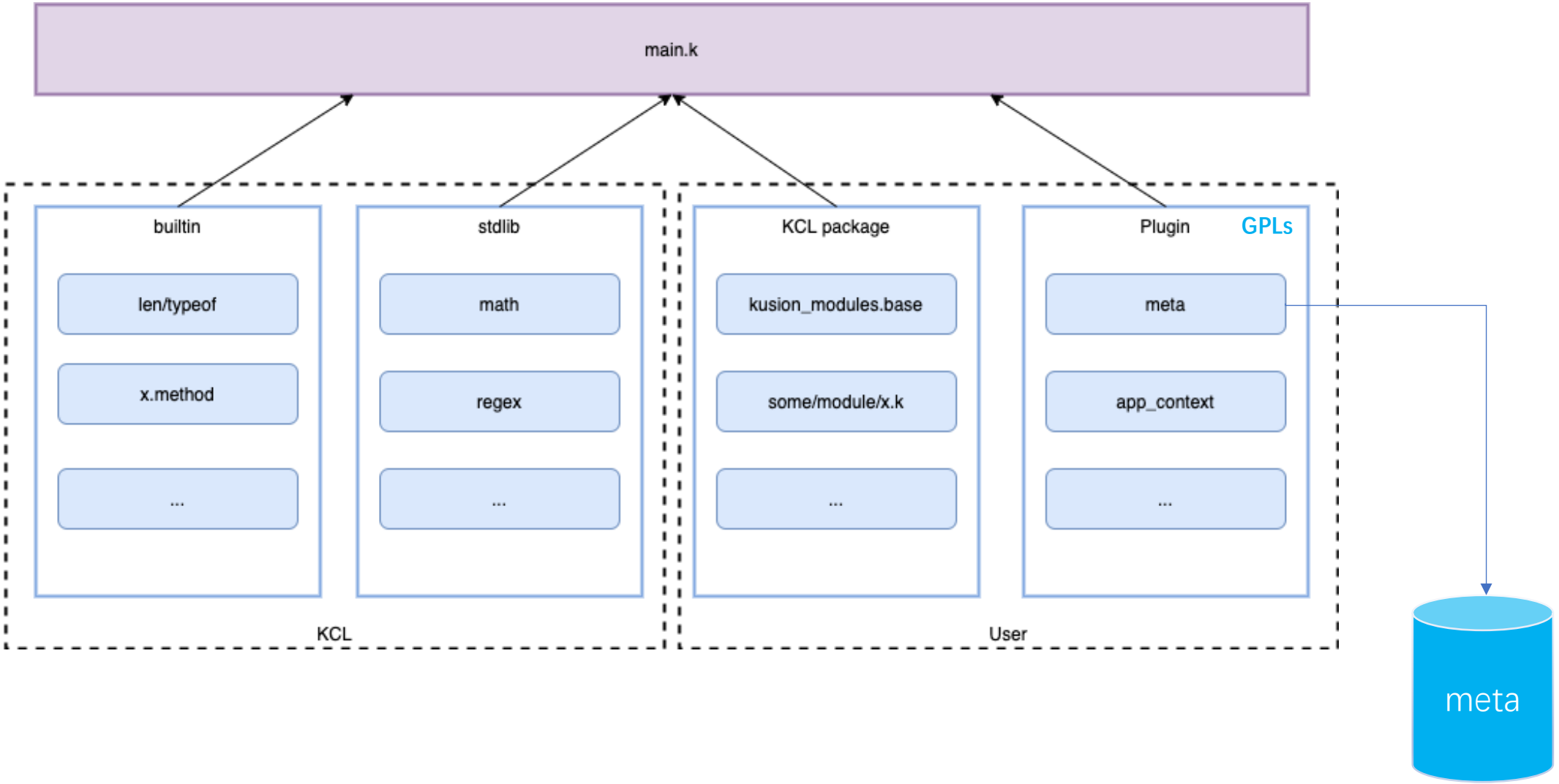
KUSION STACK

kcl -O appConfiguration.image='"nginx:1.7.9"'

```
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"
}
```

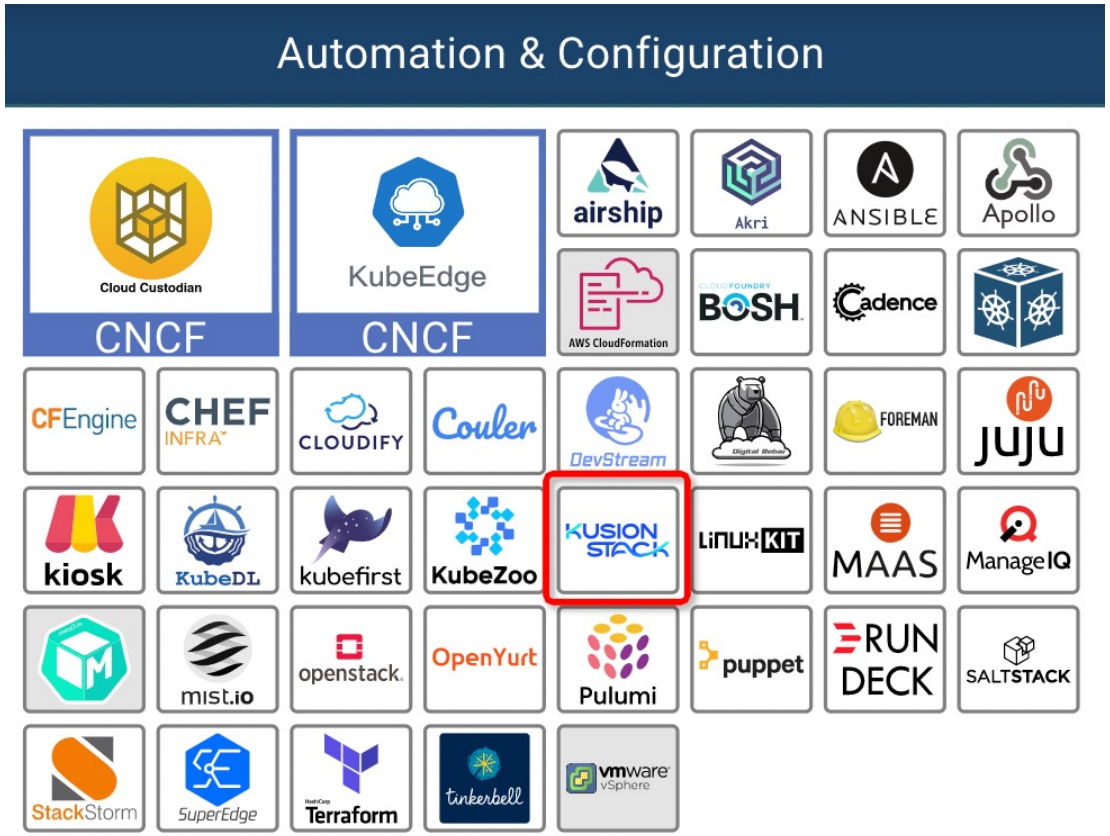
Modules


main.k
builtin
len/typeof
x.method
...
stdlib
math
regex
...
KCL
KCL package
kusion_modules.base
some/module/x.k
...
Plugin
GPLs
meta
app_context
...
User
meta

Use cases

03

Scenarios



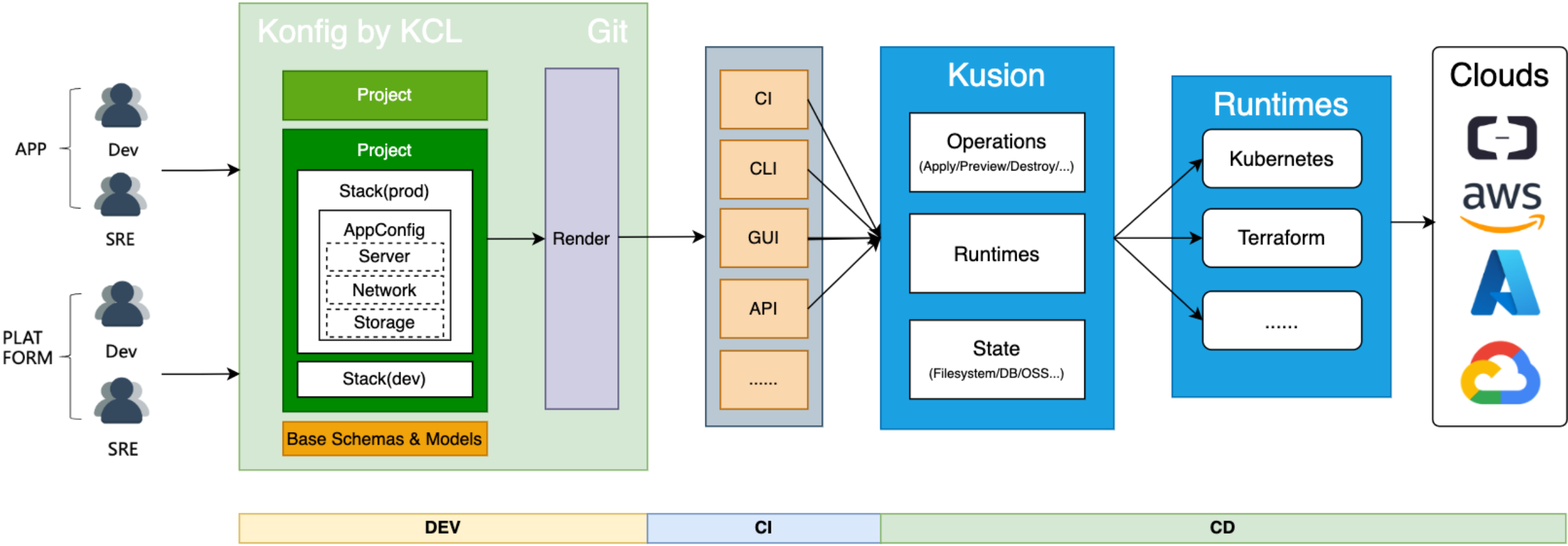
Configuration & Automation

- Infra
- UI
- API
- Tools
- CI/CD Components
- Database
- App
- Kubernetes
- Data

Security & Compliance

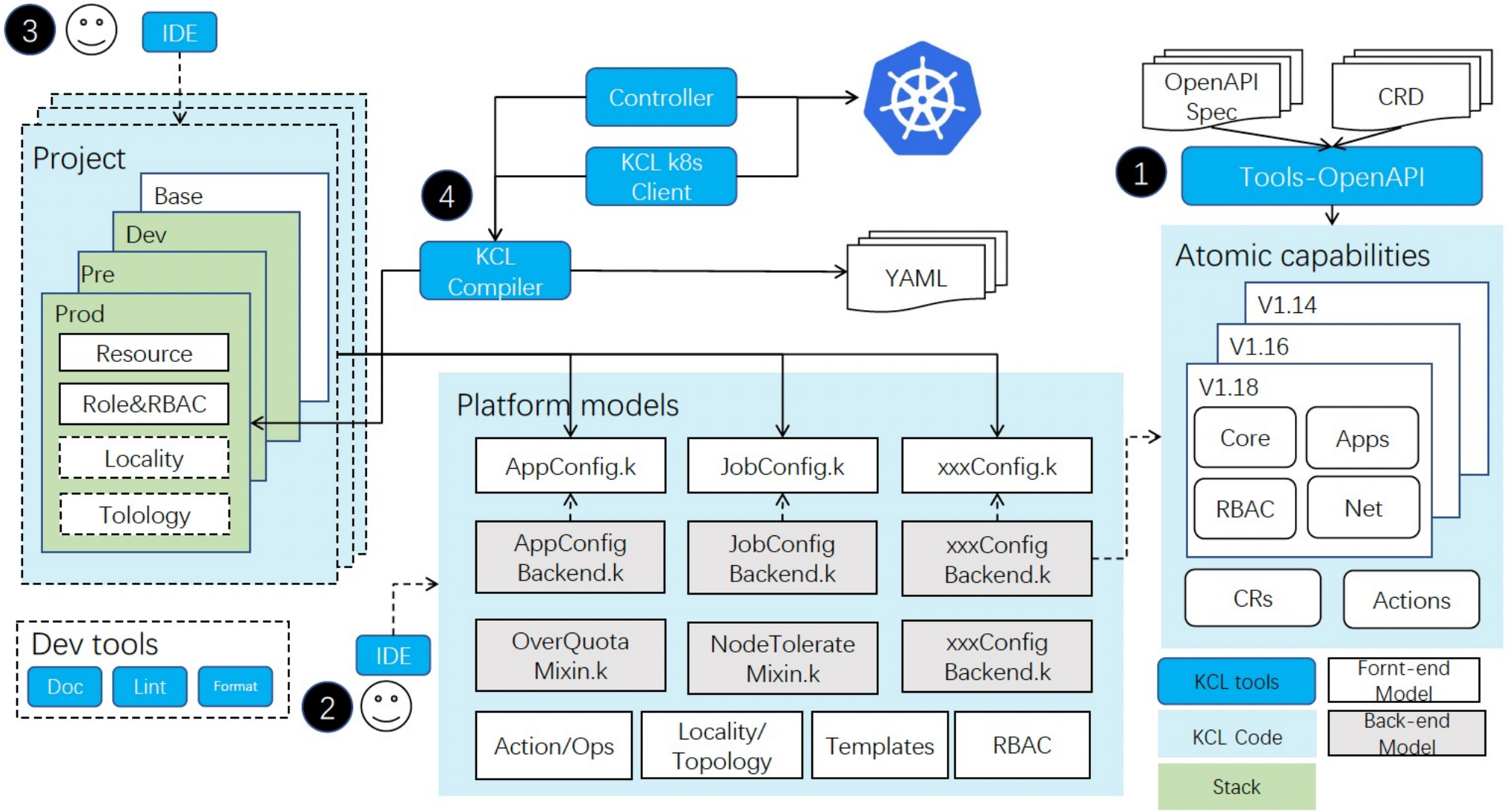
- Kubernetes
- Data
- PL
- CI/CD Components
- Database

Working with KusionStack

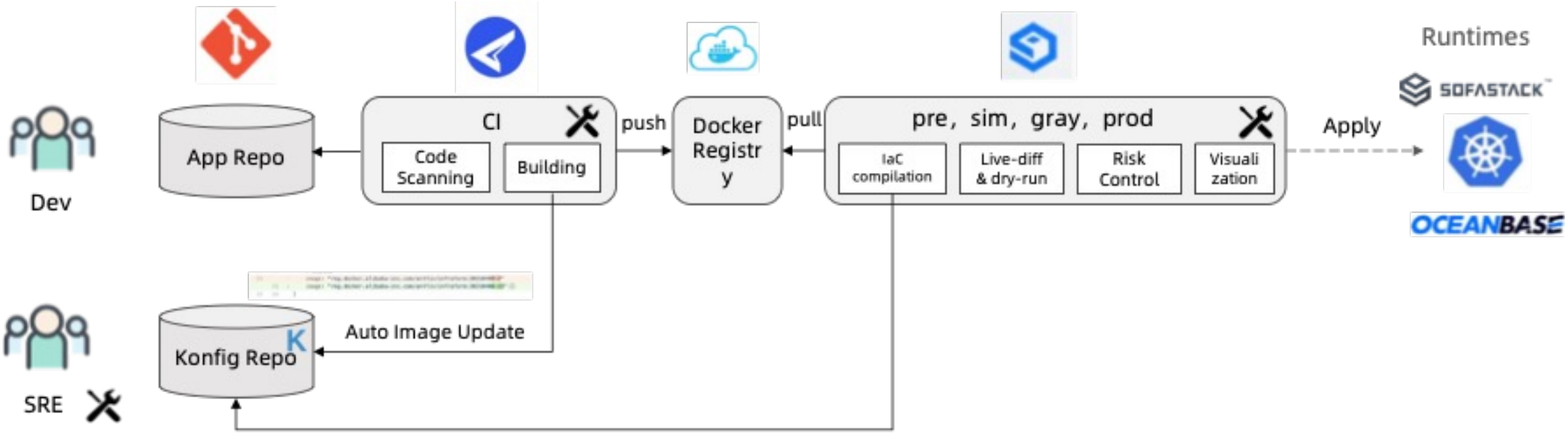


https://github.com/KusionStack/kusion

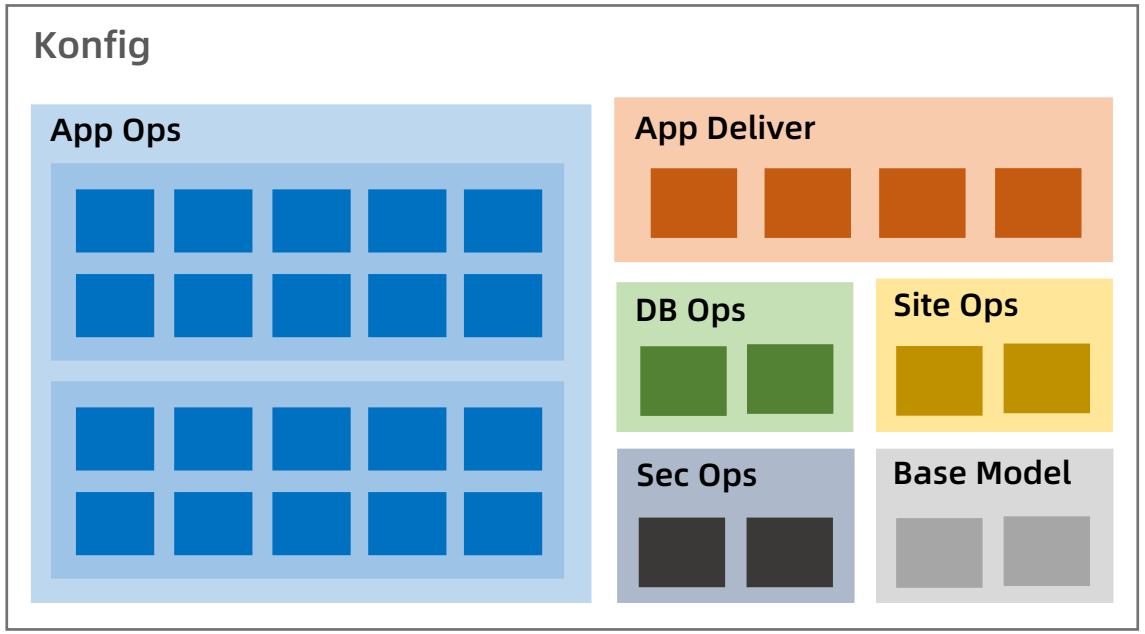
Workflow


3
IDE
Project
Base
Dev
Pre
Prod
Resource
Role&RBAC
Locality
Tolology
Controller
KCL k8s Client
4
KCL Compiler
YAML
OpenAPI Spec
CRD
1
Tools-OpenAPI
Atomic capabilities
V1.14
V1.16
V1.18
Core
Apps
RBAC
Net
CRs
Actions
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
Dev tools
Doc
Lint
Format
IDE
2
KCL tools
Fornt-end Model
KCL Code
Back-end Model
Stack

Automation Integration



Workspace



.github	fix lint check action after merging dependency parsing	2 months ago
appops	feat: new sample with crd, clickhouse-operator	3 months ago
base	cleancode: base/pkg/kusion_prometheus/monitoring/v1alpha1 (#61)	22 days ago
hack	fix lint check script	2 months ago
hooks	fresh start	7 months ago
.devcontainer.json	add .devcontainer.json (#64)	5 days ago
.gitignore	fresh start	7 months ago
LICENSE	chore: refactor README.md and add LICENSE	7 months ago
Makefile	fresh start	7 months ago
README-zh.md	add an "Open in GitHub Codespaces" badge (#65)	5 days ago
README.md	add an "Open in GitHub Codespaces" badge (#65)	5 days ago
kcl.mod	fresh start	7 months ago

README.md

Konfig

Open in GitHub Codespaces

English | Chinese

https://github.com/KusionStack/konfig

Examples

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

appConfiguration: frontend.Server {
    image = "nginx"
}
```

```
schema Server:
    """Server is the abstraction of Deployment and Statefu
    image: str, default is Undefined, required.
```

Config
Start your cloud-native journey with scalable config

Schema

Lambda

Rule

Examples

Evaluation

04

Related Works

Table 2: Features of configuration languages

	KCL	GCL	CUE	Jsonnet	HCL	Dhall
Variables	✓	✓	✓	✓	✓	✓
Reference	✓	✓	✓	✓	✓	✓
Data types	✓	✓	✓	✓	✓	✓
Schema	✓	✓	✓	✗	✗	✓
Inherited	schema	tuple	✗	obejct	data	data
Arithmetic&Logic	✓	✓	✓	✓	✓	✓
Loop	list/dict/schema comprehension	list comprehension	list comprehension	list/object comprehension	for splat (comprehension)	list generate function
Conditional	✓	✓	✓	✓	✓	✓
Built-in function	✓	✓	✓	✓	✓	✓
Function definition	✓	✗	✗	✓	✗	✓
Import	✓	✓	✓	✓	✓	✓
Type check	✓	✓	✓	✗	✗	✓
Testable	✓	✗	✗	✓	✗	✓
Mixin	✓	✗	✗	✗	✗	✗
Data integration	✓	✗	✗	JSON	✗	✗
Dynamic configuration	✓	✗	✗	✓	✗	✗
Policy	✓	✗	✗	✗	✗	✗
Merge	idempotent merge/ patch/ unique configuration	patch	idempotent merge	patch	✗	idempotent merge/ patch

Performance

- Jsonnet (test.jsonnet)

```
local a(x, y) = std.max(x, y);
{
    temp: [a(1, 2) for i in std.range(0, 10000)],
}
```

- Terraform HCL (test.tf. Since the terraform `range` function only supports up to 1024 iterators, the `range(10000)` is divided into 10 sub ranges)

```
output "r1" {
  value = {for s in range(0, 1000) : format("a%d", s) => max(1, 2)}
}
```

- CUE (test.cue)

```
import "list"

temp: {
        for i, _ in list.Range(0, 10000, 1) {
                "a\(i)": list.Max([1, 2])
        }
}
```

- KCL (test.k)

```
a = lambda x: int, y: int -> int {
    max([x, y])
}
temp = {"a${i}": a(1, 2) for i in range(10000)}
```

KCL v0.4.3 Running time (including compilation+runtime)	CUE v0.4.3 Running time (including compilation+runtime)	Jsonnet v0.18.0 Running time (including compilation+runtime)	HCL in Terraform v1.3.0 Running time (including compilation+runtime
440 ms (kclvm_cli run test.k)	6290 ms (cue export test.cue)	1890 ms (jsonnet test.jsonnet)	1774 ms (terraform plan –parallelism=1)

Paper

KUSION STACK

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

XiaoDong Duo[1], Pengfei Xu[1], Zheng Zhang[1], Shushan Chai[1], Rui Xia[1], and Zhe Zong[1]

Ant Group
lingzhi.xpf@antgroup.com

Abstract. In recent years, the diversification, complexity, immediacy, and scale of delivery and operational requirements have increased exponentially. For example, delivering and managing complex service mesh and various cloud-native technologies, supporting a variety of operations on infrastructures, such as database, load balancer, dynamic configuration, etc, configuring monitoring for all types of applications, and arranging a range of variety of services and applications to regions across data centers or public clouds. These are just a glimpse of the leopard, and it is actually difficult to list them all. By summarizing and abstracting, we propose the KCL declarative language, development mechanism, and consistent workflow. Through the language model and constraint capabilities, we can improve the large-scale efficiency and liberate multi-team collaborative productivity of operational development and operation systematically while ensuring stability for large-scale configuration and policy management. KCL helps users of various roles to complete operational development and operation tasks in a simple, scalable, stable, efficient, divided-and-conquered manner. To date, the KCL has been used in more than 800 projects, and the average configuration writing and distributing time is shortened from more than 25 days to 2 days.

Keywords: Language · Delivery · Operation · Stability · Collaborative.

https://kcl-lang.github.io/docs/events/2022/kcl_paper

Who is Using KCL

Key Results at Ant Group

10 K/day	~800 K	~23 K
KCL Compilations	KCL Code	PRs
~1000	~500	~80 K
Projects	Contributors	Commits

Roadmap

More Friendly for Dev

Wider Ecological Integration

Advanced Technology Exploration

Powerful Lang & Compiler Capabilities

v0.4.3

- Lang Simplification (Spec, Syntax)
- KCL APIs by Rust
- Completely KCL Tools Support: lint, test, ...
- MThe Compiler Natively WASM execution

2022.9

v0.4.4

- Compiler Decorator Extension
- Flow & Policy Capability Enhancement
- Model Registry & Package Management
- More LSP Based IDEs
- Common Domain Language Programming Framework : Compiler-Base v0.1 (Common Components)

2022.12

v0.5

- Lang Simplification (Syntax, Semantics)
- Reverse type inference
- Incremental compilation
- Multi Runtime/Backend
- Configure Ecosystem Integration

2023.3

v0.6

- CFG-Based KCL IR
- Garbage collector
- JIT Compiler
- Compiler-Base v0.2 (Compiler Components, Interface)

2023.6

Resources

- Web Site
  - https://kcl-lang.github.io/
  - https://kusionstack.io/
- Source Code
  - https://github.com/KusionStack/KCLVM
  - https://github.com/KusionStack/kusion
  - https://github.com/KusionStack/konfig
- Contact
  - https://github.com/KusionStack/community#contact
  - https://github.com/KusionStack/community
- Twitter
  - @KusionStack

欢迎大家用微信扫码
加入 KusionStack 官方微信群

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

KUSION STACK

THANKS