KusionStack Origin, present and future

KusionStack Team

Agenda

01 Origin

02 Overview

03 Solution

04 Tech

05 Scene

06 Practice

07 Future

Origin

Cloud-native technologies

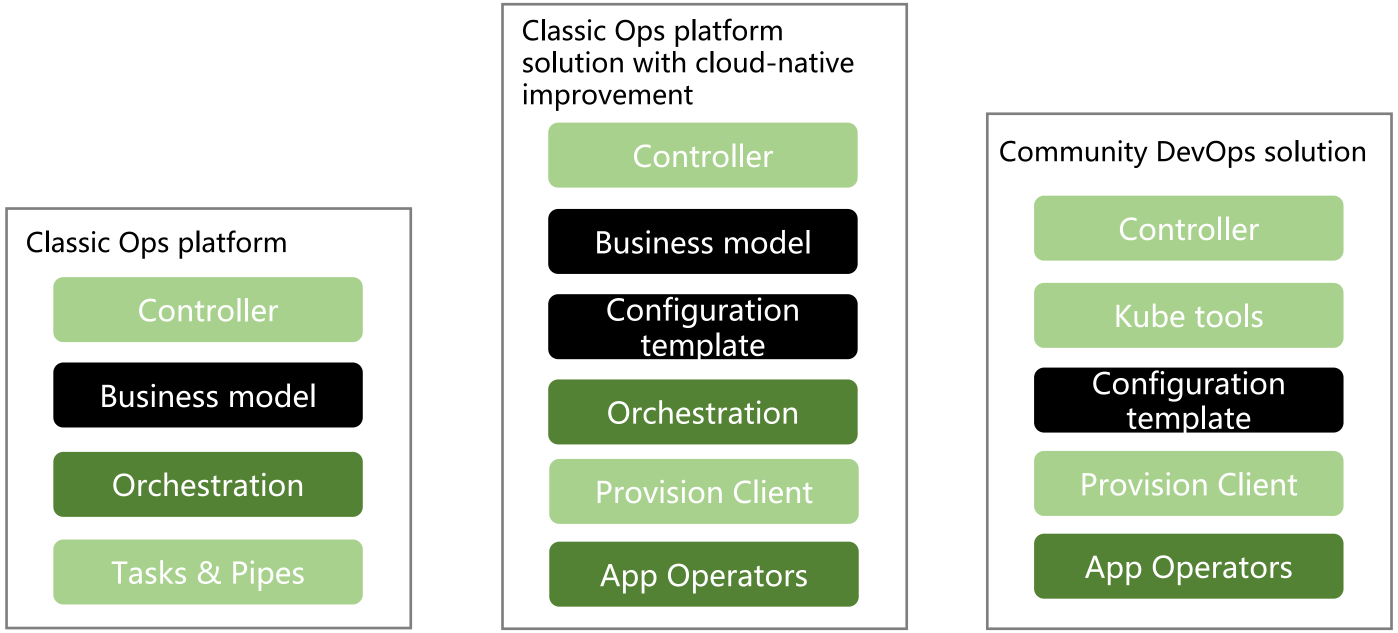
Are eating the world

- Becoming first-party approach for enterprises and cloud vendors, forming a global community-driven ecosystem
- Modern applications relies on hybrid technologies of cloud-native and non-cloud-native, PaaS and IaaS on multi-clouds
- Collaborative DevOps among application, SRE and platform developers is the key to Ops efficiency at scale across multi-phases and multi-envs
- Continuous improvement in **abstraction**, **management** and **user-experience** above the kube ecosystem is undergoing

Diversity, scale and change

Create ongoing challenges

- Classic Ops platform: insufficient openness, flexibility and scalability
- Community DevOps tool: don't meet 'enterprise-grade' needs

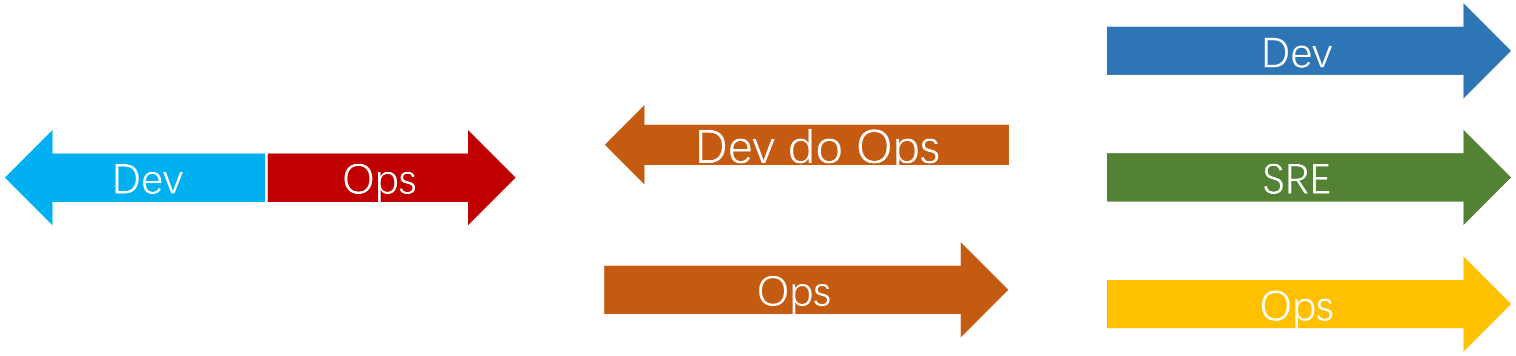


What we tried and not that successful

The darker the part, the more complex, the faster the change

Ops practice and ecology

Are stilling evolving



Dev Ops

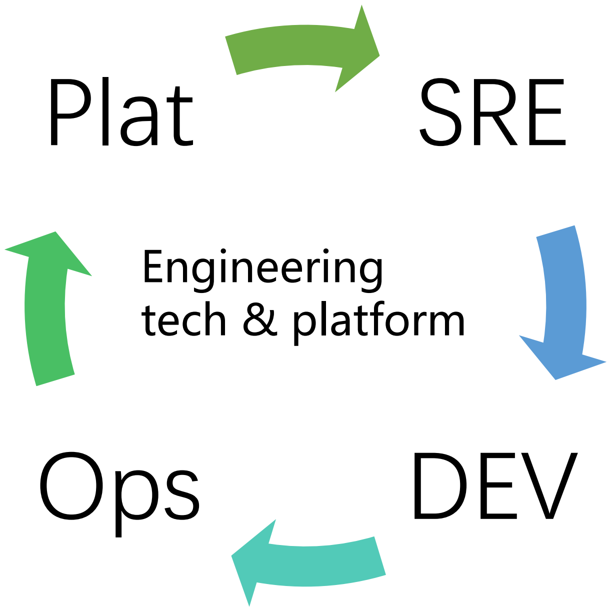
Dev do Ops

Ops

Dev

SRE

Ops



Plat

SRE

Engineering tech & platform

Ops

DEV

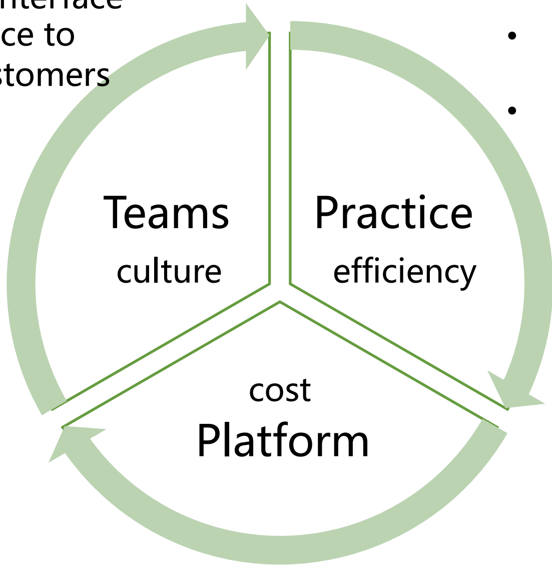
OVERVIEW

Collaborative, Creative

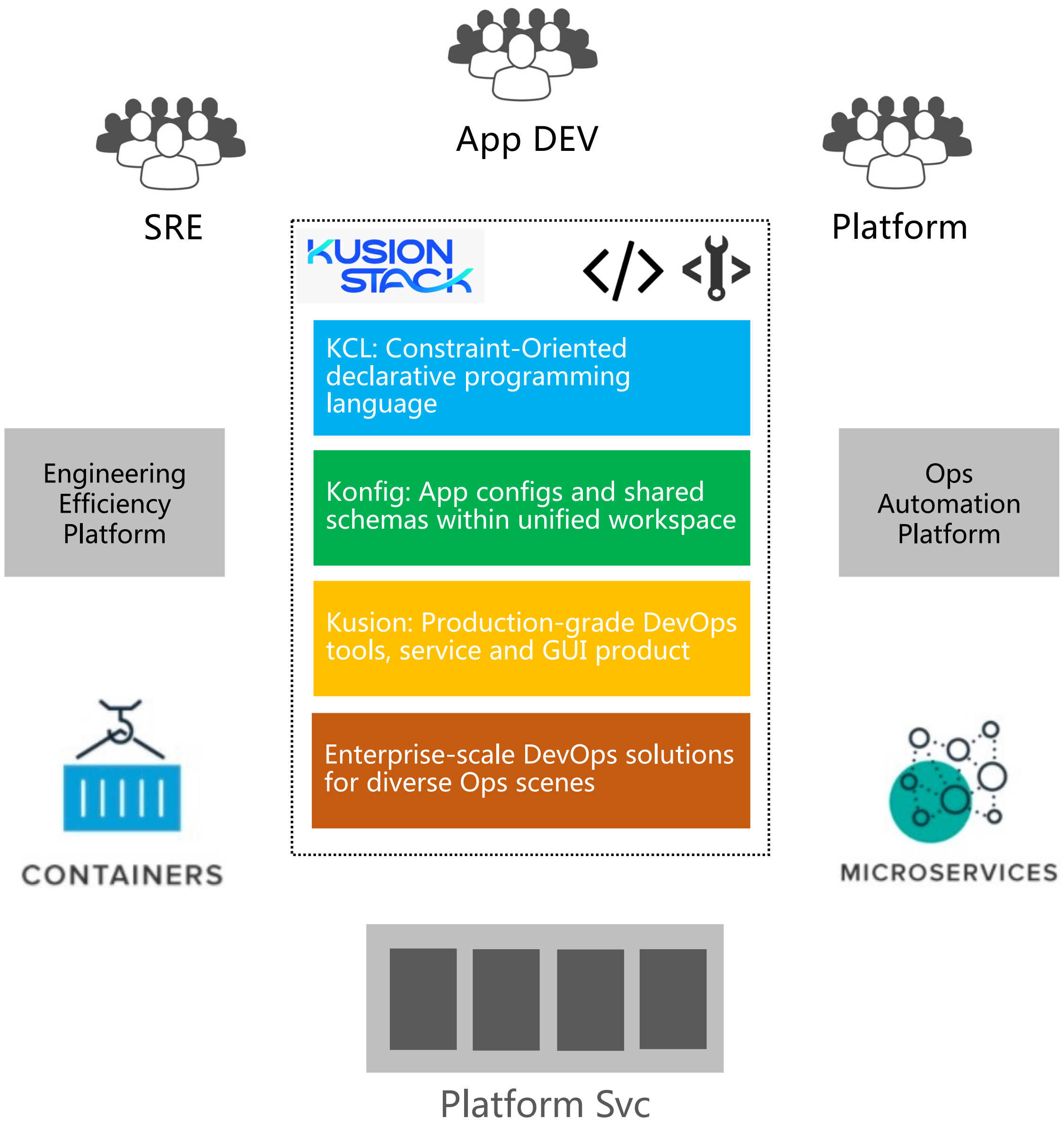
Make scaled DevOps possible

- Collaborate and share across teams
- One-stop working space and interface
- Better service to internal customers

- Coding everything
- Efficient Ops business development
- Manage change based on commit
- Left-shifted inspection and analysis
- Weakening the process with practice



- Highly open CI/CD/CDRA platform
- Unified and single-source　‘fact’ management mechanism
- Extensible to all ops scenarios
- Continue to face new challenges at enterprise scale



KCL: Constraint-Oriented declarative programming language

Konfig: App configs and shared schemas within unified workspace

Kusion: Production-grade DevOps tools, service and GUI product

Enterprise-scale DevOps solutions for diverse Ops scenes

A stack to Delivery App, by Platform as Code

Make scaled delivery agile

Enterprise Declarative DevOps

App Centric Shipping Anywhere

Unified Stack Modern Arch & Workflow

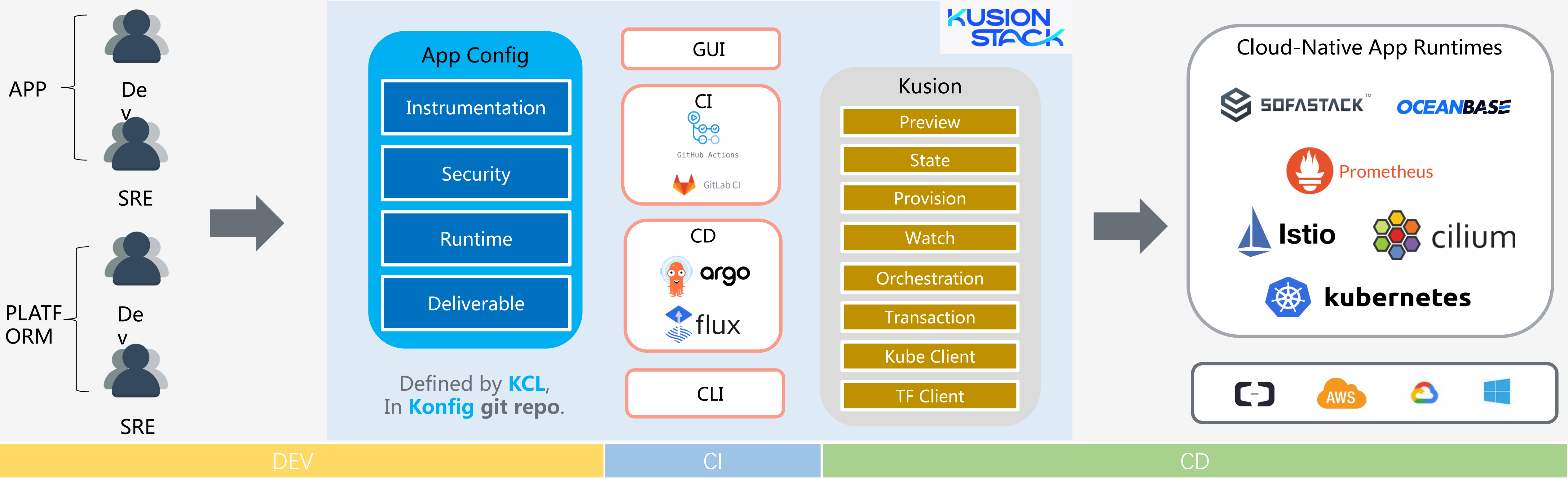
Multi-{Teams, Roles}
Programmable DevOps

App Ops Lifetime Config
Multi-{Tenant, Env} Solution

Collaboration
Diversified Automation

Declarative Ops
Dev-friendly Experience

Multi-Runtimes
Multi-Clouds



APP

De
v

SRE

PLATF
ORM

De
v

SRE

KUSION STACK

App Config

- Instrumentation
- Security
- Runtime
- Deliverable

Defined by **KCL**,
In **Konfig git repo**.

GUI

CI
GitHub Actions
GitLab CI

CD
argo
flux

CLI

Kusion

- Preview
- State
- Provision
- Watch
- Orchestration
- Transaction
- Kube Client
- TF Client

Cloud-Native App Runtimes

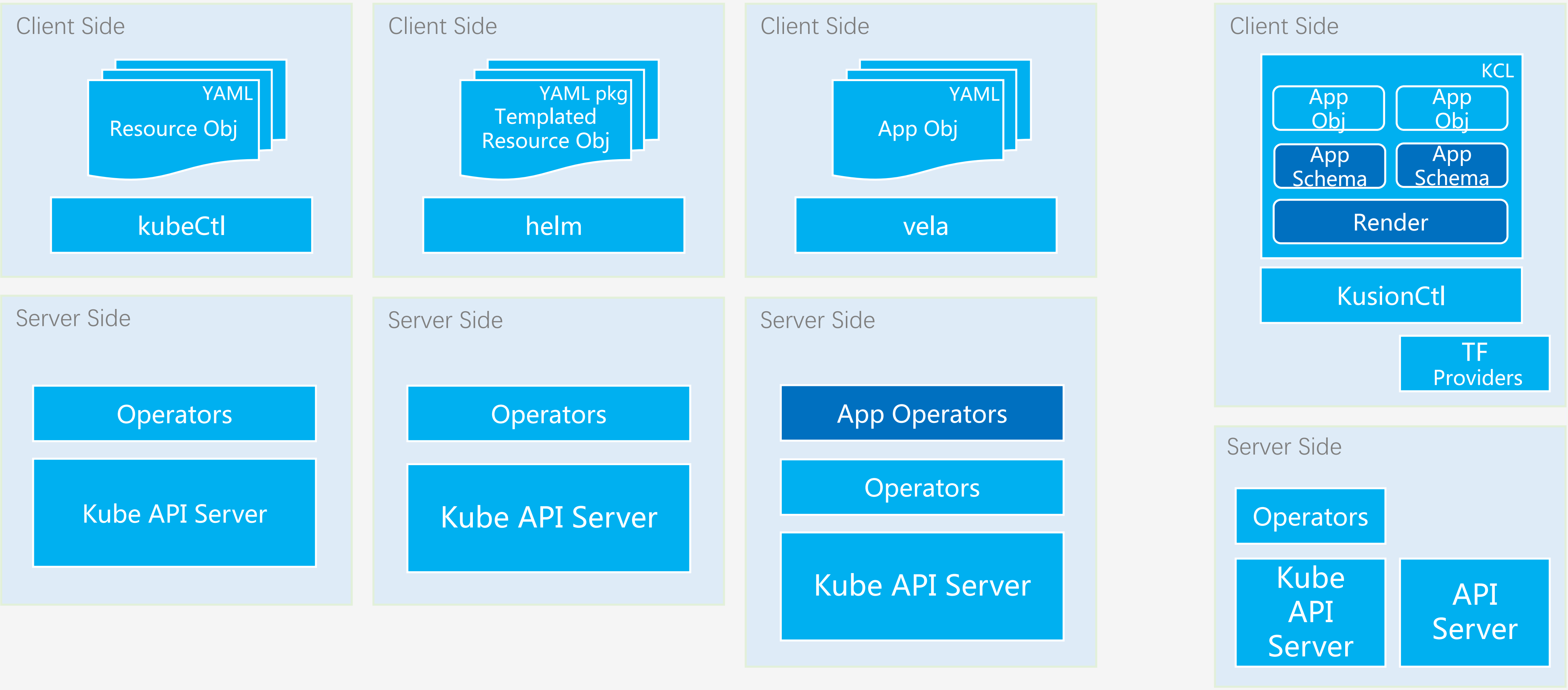
SOFASTACK™ OCEANBASE
Prometheus
Istio cilium
kubernetes

AWS

DEV | CI | CD

Solution

Kube Ops with X



Client Side
YAML
Resource Obj
kubeCtl

Server Side
Operators
Kube API Server

Client Side
YAML pkg
Templated Resource Obj
helm

Server Side
Operators
Kube API Server

Client Side
YAML
App Obj
vela

Server Side
App Operators
Operators
Kube API Server

Client Side
KCL
App Obj
App Obj
App Schema
App Schema
Render
KusionCtl
TF Providers

Server Side
Operators
Kube API Server
API Server

- App Centric
  - Modeling
  - Abstraction
  - Customization
  - Combination
  - Policy
- Pure Client Solution
  - Coded by KCL
  - Lightweight
  - Flexible
  - Scalable
  - Portable
  - Left-shifted stability
- Hybrid-platform
  - On Kube & TF
  - On Multi-Clouds
  - Provision
  - Orchestration
  - Visualization
  - …
- E2E Support
  - Dev
  - CI
  - CD

➢ **Typical tool：Kustomize**

➢ **Typical tool：Helm**

➢ **Typical tool：KubeVela**

➢ **KusionStack**

KusionStack - KCL

KCL - An Open Source Constraint-Based Record & Functional Language

Well-Designed

Spec-driven
Config, Schema,
Lambda, Rule

Easy to Use

In Configuration
Policy cases

Modeling

Schema-Centric
Abstraction

Stability

Static Type System
Constraints
Rules

Scalability

Separated Config Blocks
Rich Merge & Override
Strategies

Automation

CRUD APIs
Multi-Lang SDKs
Plug-ins

Cross-Platform

High-Performance
Multi-Runtime

Cloud-Native Affinity

Open API/CRD
Specs/YAML Spec

Dev Friendly

Lint/Test/Vet/Doc
Tools
VS Code/Intellij IDE

KusionStack - Konfig & Kusion

An abstraction and management layer to deliver modern app

Monorepo

Organize all app confs in one repo with scalable project & stack structure

Multiple Hierarchies

Natively support multi-tenant and multi-environment configuration

Extendable Models

Extendable and reusable modeling by schema, mixin and other KCL mechanisms

Vendor Agnostic

Write once, deliver any runtime, any cloud through a consistent workflow

Lifecycle Management

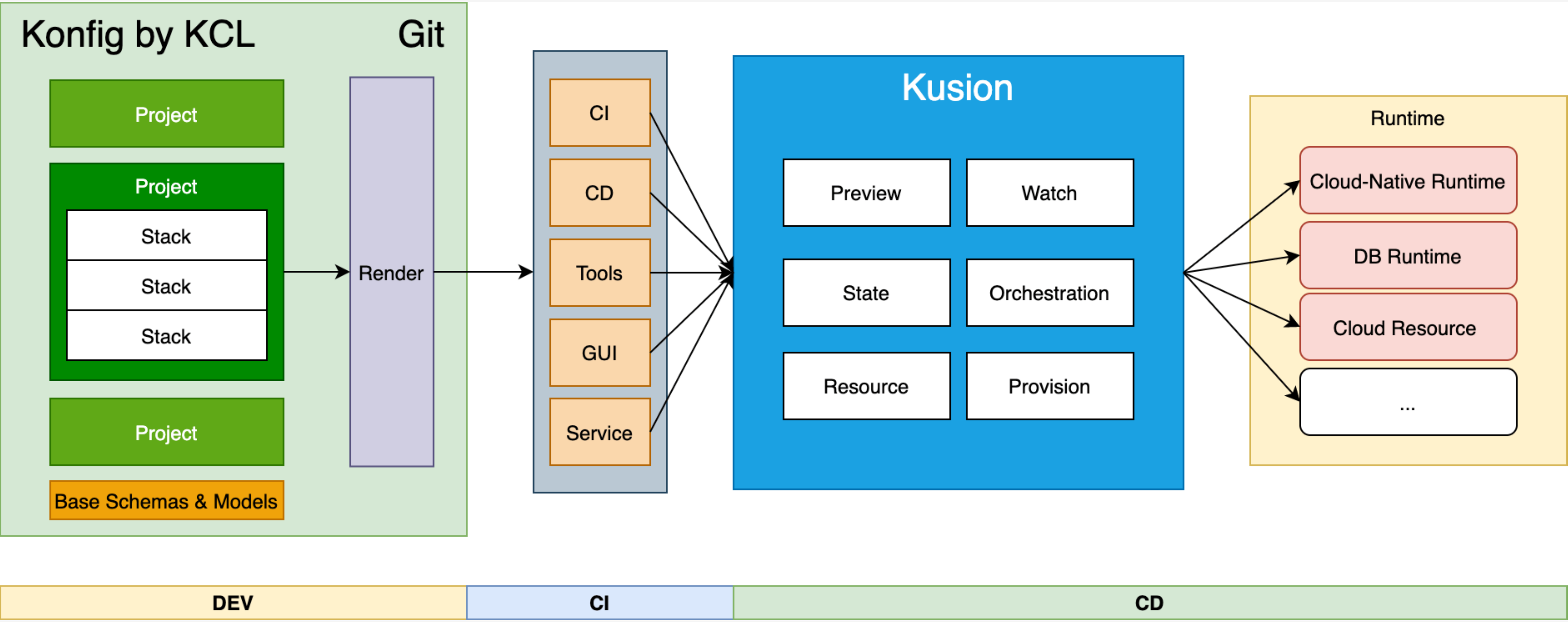
Manage app from the first code to production-ready across multi-phases

Hybrid Resource

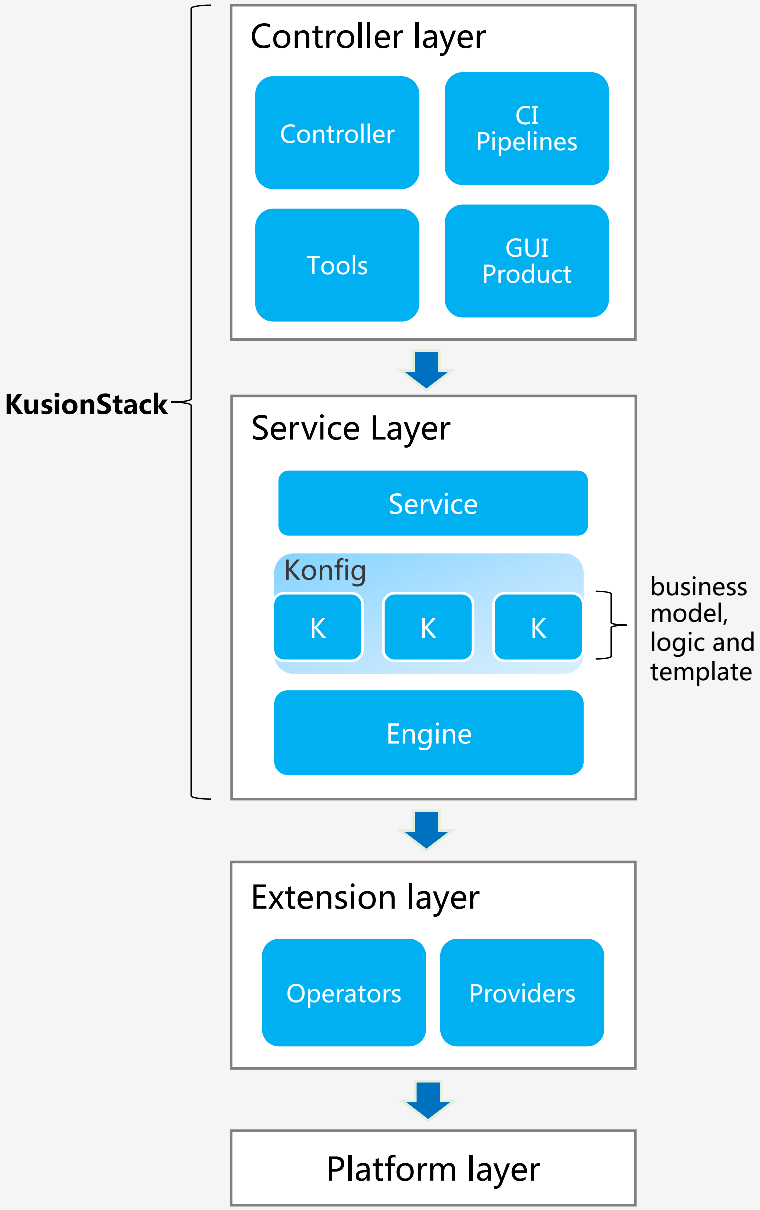
Orchestrate resources on various runtime in a managed manner

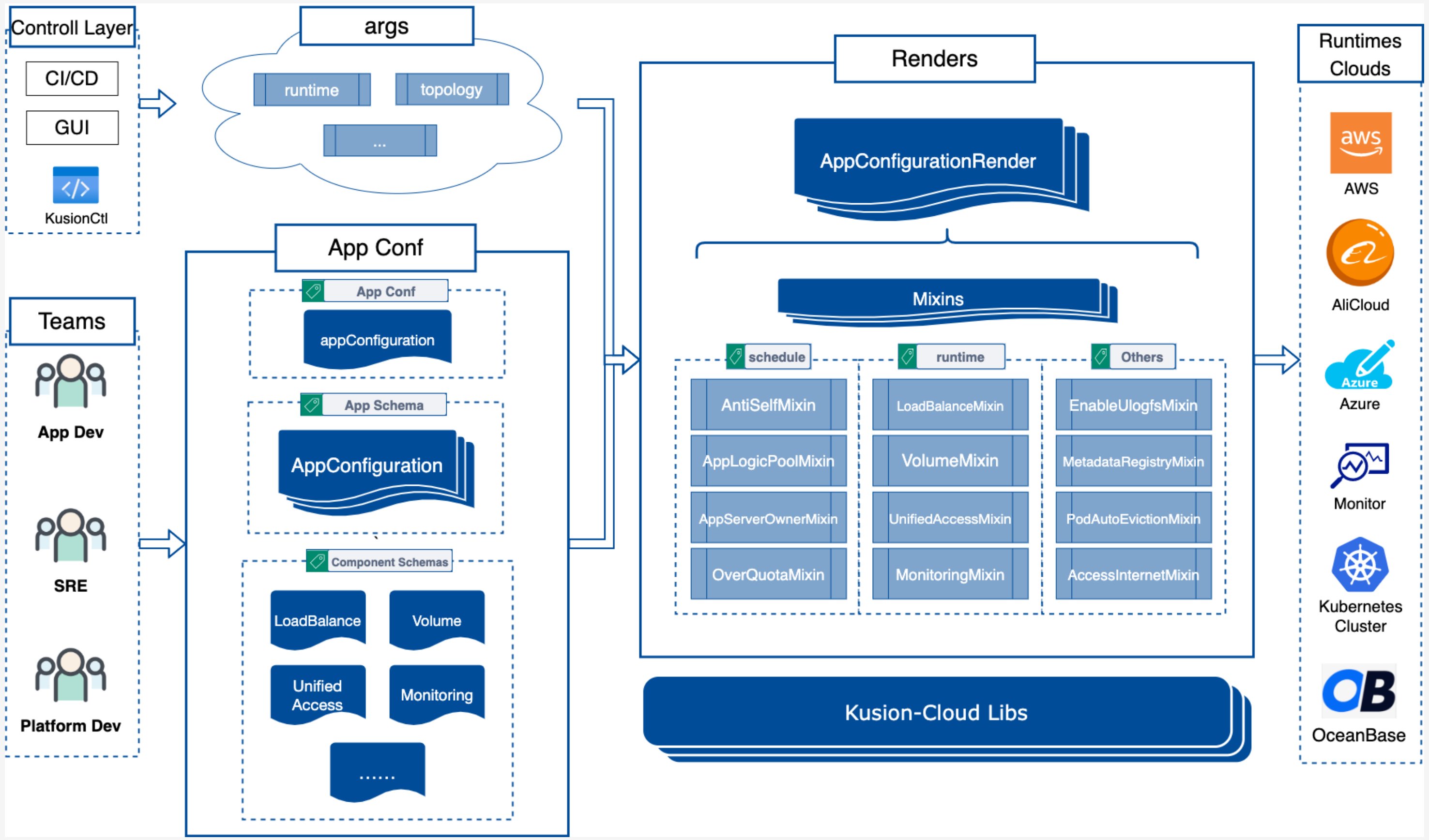
Enterprise Solutions

Scaled and flexible development and automation

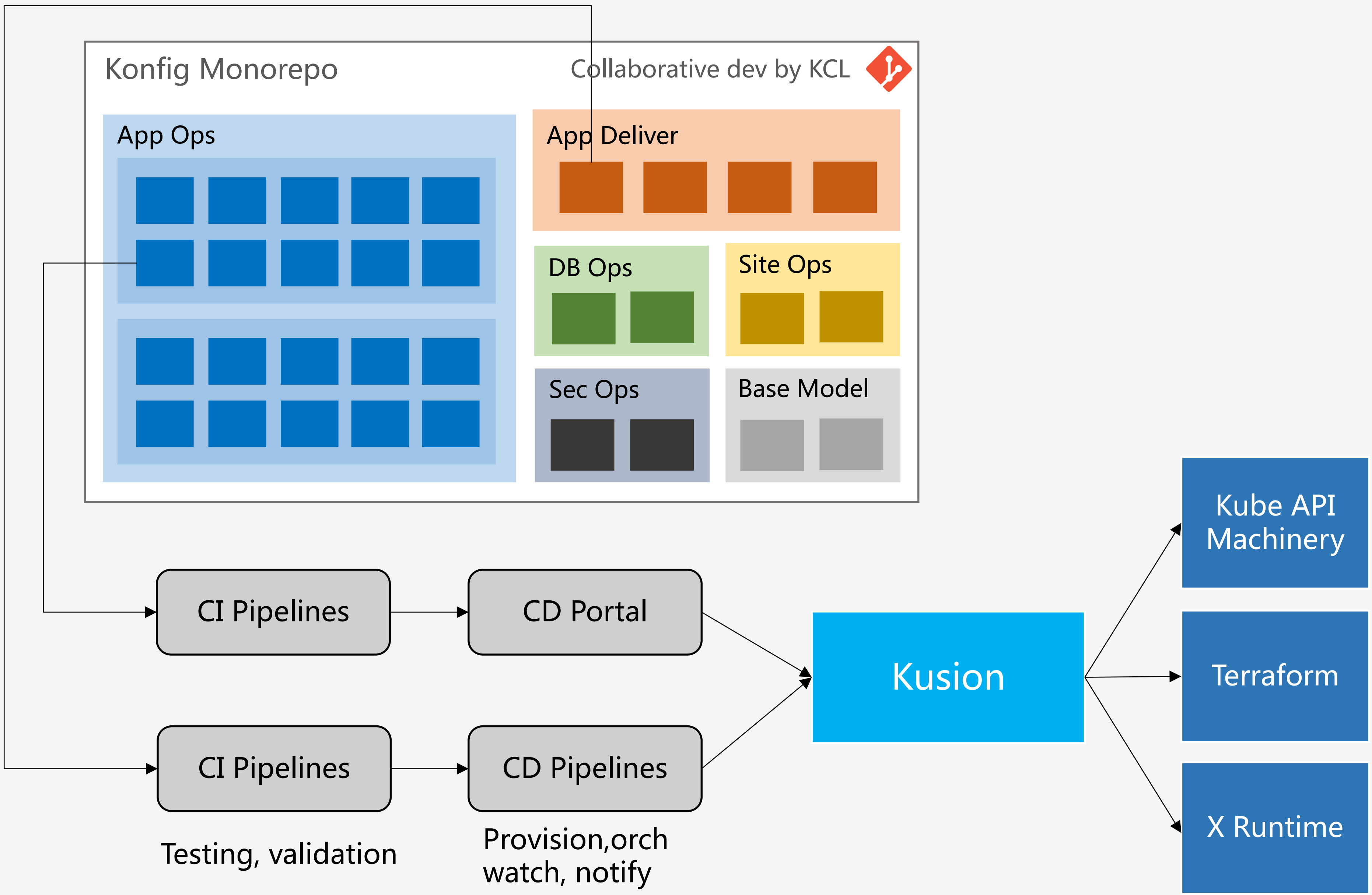


Layers & Collaboration





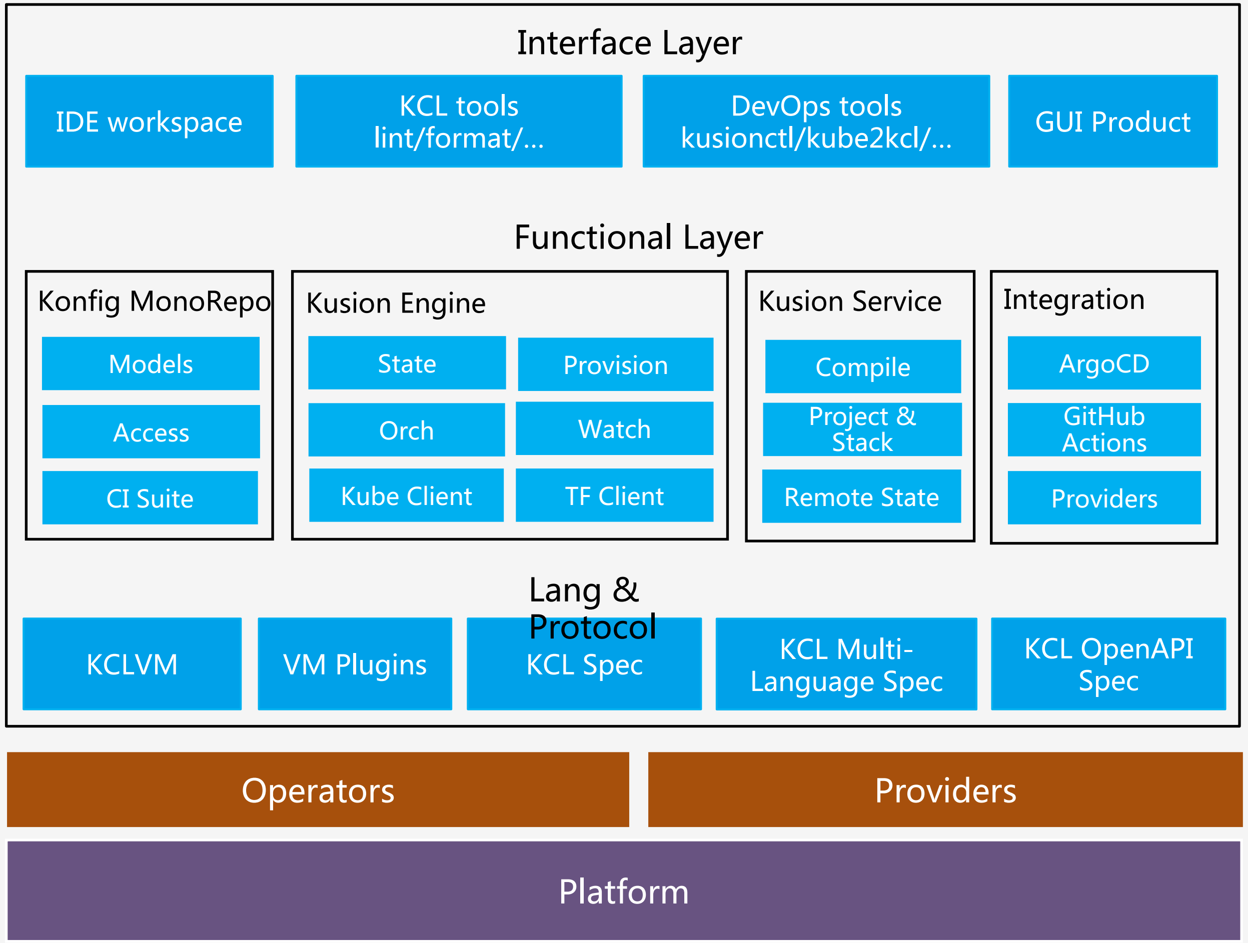
Automation Workflow



Tech

KusionStack Arch

Lang, tools, interface and workspace



Interface Layer

IDE workspace

KCL tools
lint/format/...

DevOps tools
kusionctl/kube2kcl/...

GUI Product

Functional Layer

Konfig MonoRepo

Models

Access

CI Suite

Kusion Engine

State

Provision

Orch

Watch

Kube Client

TF Client

Kusion Service

Compile

Project & Stack

Remote State

Integration

ArgoCD

GitHub Actions

Providers

Lang & Protocol

KCLVM

VM Plugins

KCL Spec

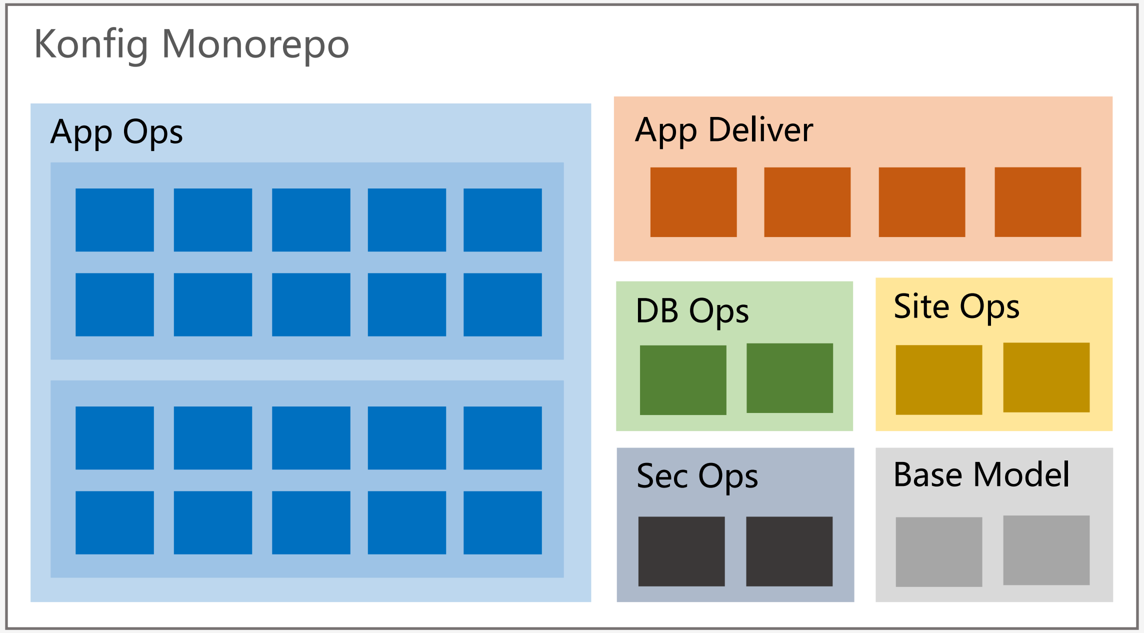
KCL Multi-Language Spec

KCL OpenAPI Spec

Operators

Providers

Platform



Konfig Monorepo

App Ops

App Deliver

DB Ops

Site Ops

Sec Ops

Base Model

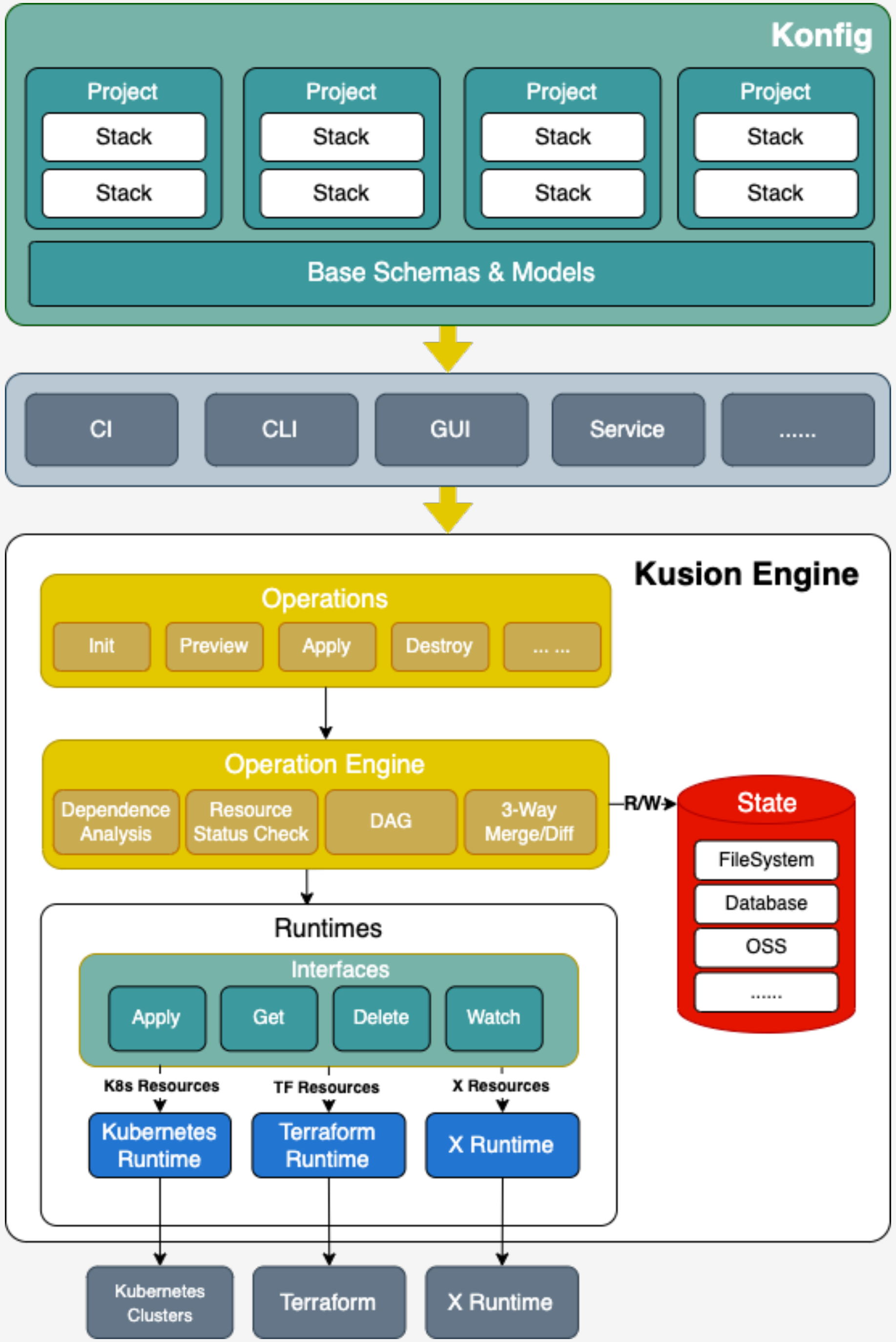
Konfig & Kusion

Managed resource across multiple runtimes

Operation Engine: provide core features to support all Kusion operations

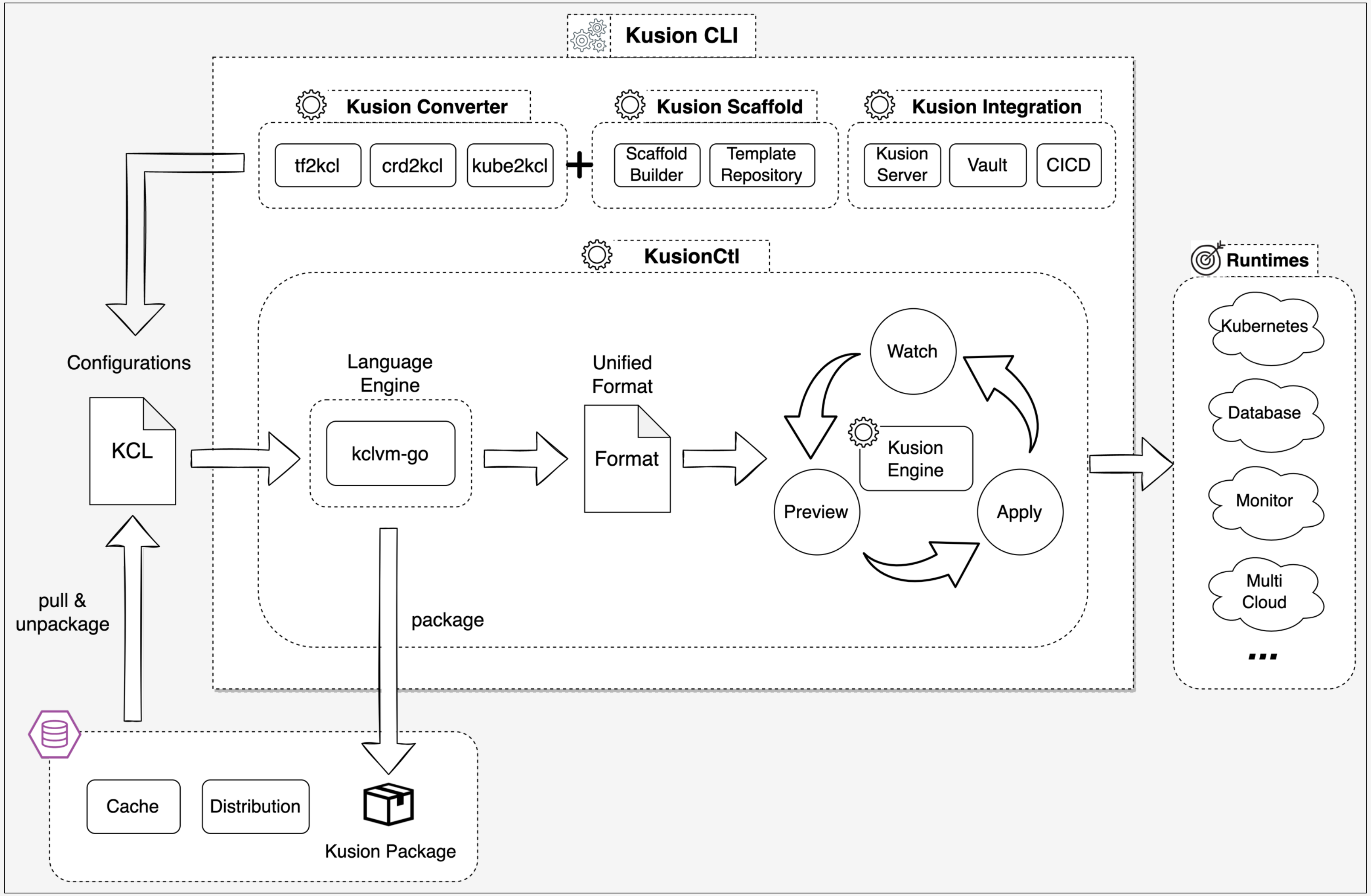
Runtimes: represent actual infrastructure runtimes managed by Kusion.

State: a mapping between resources in Konfig and the actual infra resource



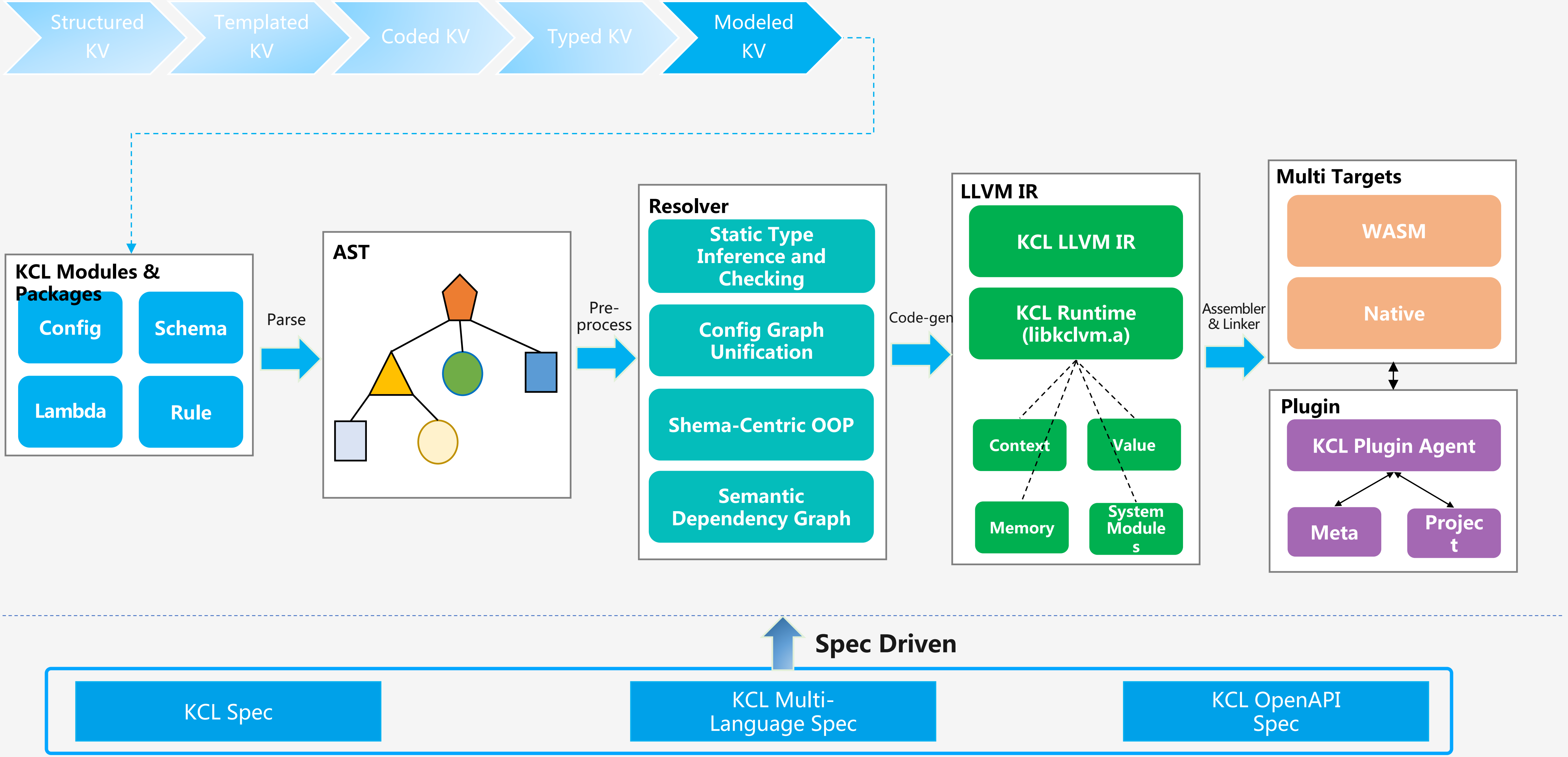
Kusion Tools

Delivery workflow easier



KCL

Config, Schema, Lambda, Rule



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 Module s

Assembler & Linker

Multi Targets

WASM

Native

Plugin

KCL Plugin Agent

Meta

Projec t

Spec Driven

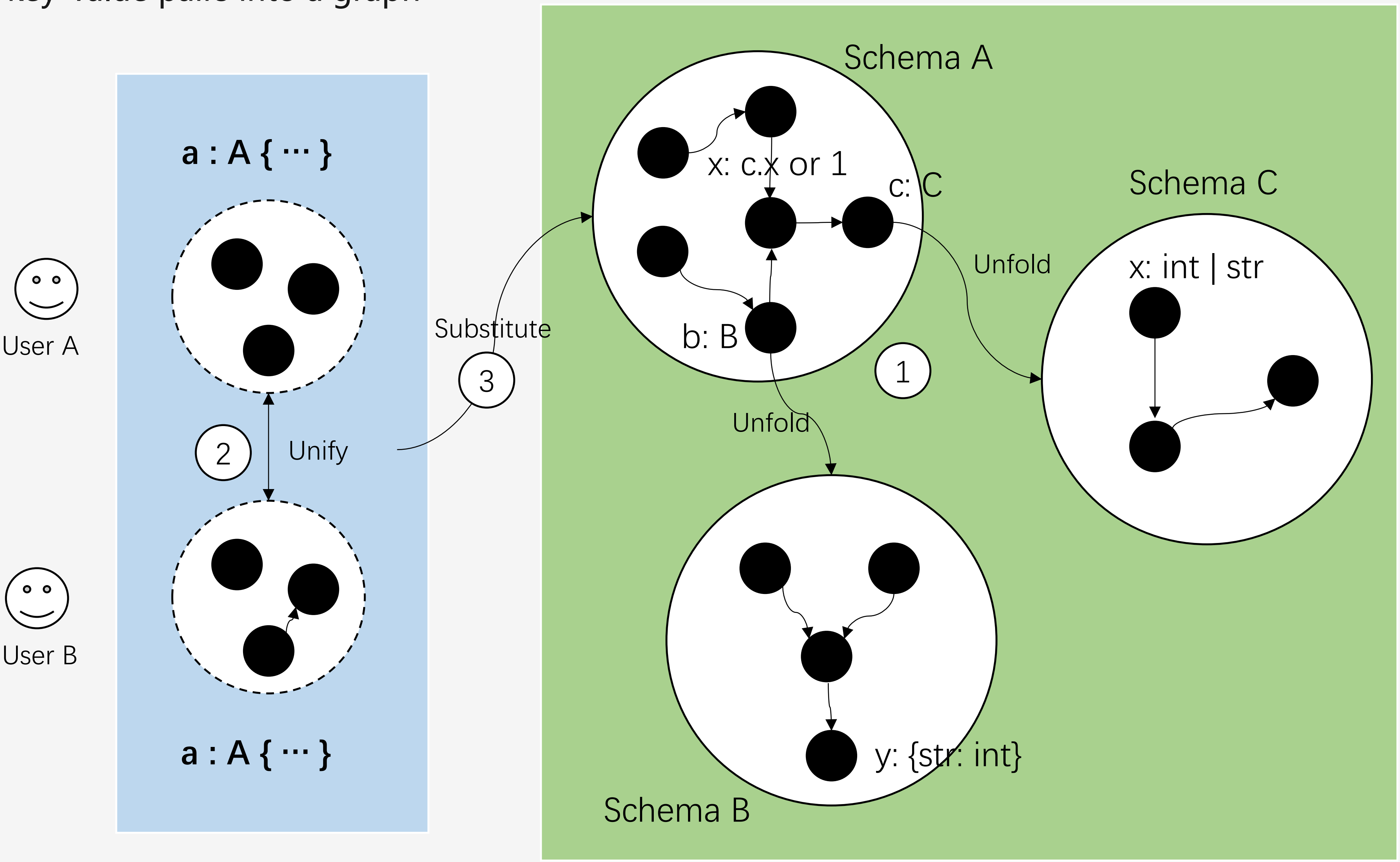
KCL Spec

KCL Multi-Language Spec

KCL OpenAPI Spec

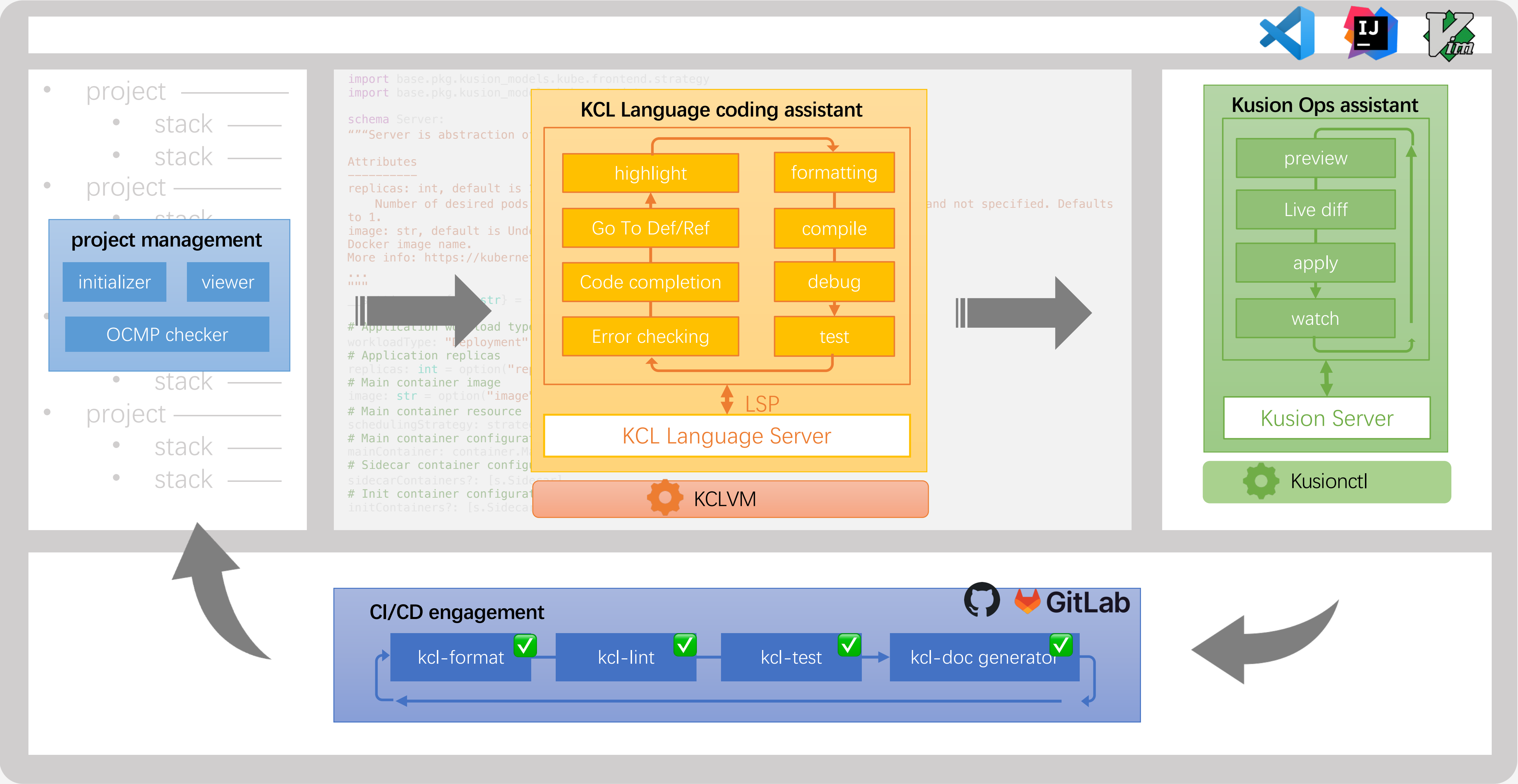
KCL Internal Graph Model

Weave key-value pairs into a graph



KCL Tools & IDE Workspace

Make Ops collaborative coding and work happy



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

schema Server:
"""Server is abstraction of

Attributes
----------
replicas: int, default is 1
    Number of desired pods                                    and not specified. Defaults
to 1.
image: str, default is Unde
Docker image name.
More info: https://kubernet
...
"""

# Application workload type
workloadType: "Deployment"
# Application replicas
replicas: int = option("rep
# Main container image
image: str = option("image"
# Main container resource
schedulingStrategy: strateg
# Main container configurat
mainContainer: container.Ma
# Sidecar container config
sidecarContainers?: [s.Sidecar]
# Init container configurat
initContainers?: [s.Sidecar
```

Scene

Users of Kusionized DevOps

App Dev

Roles

- End user

Goals

- Deliver and ops my app easier
- On any desired env and cloud

Favors

- Implicit and app-oriented working interface and process above infrastructure details
- Minimal investment in learning and practice in infrastructure and operation details

Pain points

- Too many fragmented technologies, processes and user interfaces in deliver and ops
- Too many infrastructure-oriented details to learn
- Growing cloud platforms to use

SRE

Roles

- Enabler
- End user

Goals

- Keep infra and ops stable, measurable and manageable

Favors

- Participate directly in the work of platform design and construction to make the infrastructure more reliable and easy-to-use for app developers
- Deliver and manage apps that require high stability through easy-to-use tech and tools

Pain points

- Unable to directly participate in the construction of the platform
- Platform capabilities related to stability cannot be used by app developers faster

Platform Dev

Roles

- Provider
- Enabler
- End user

Goals

- Growing the capabilities and technologies of the platform

Favors

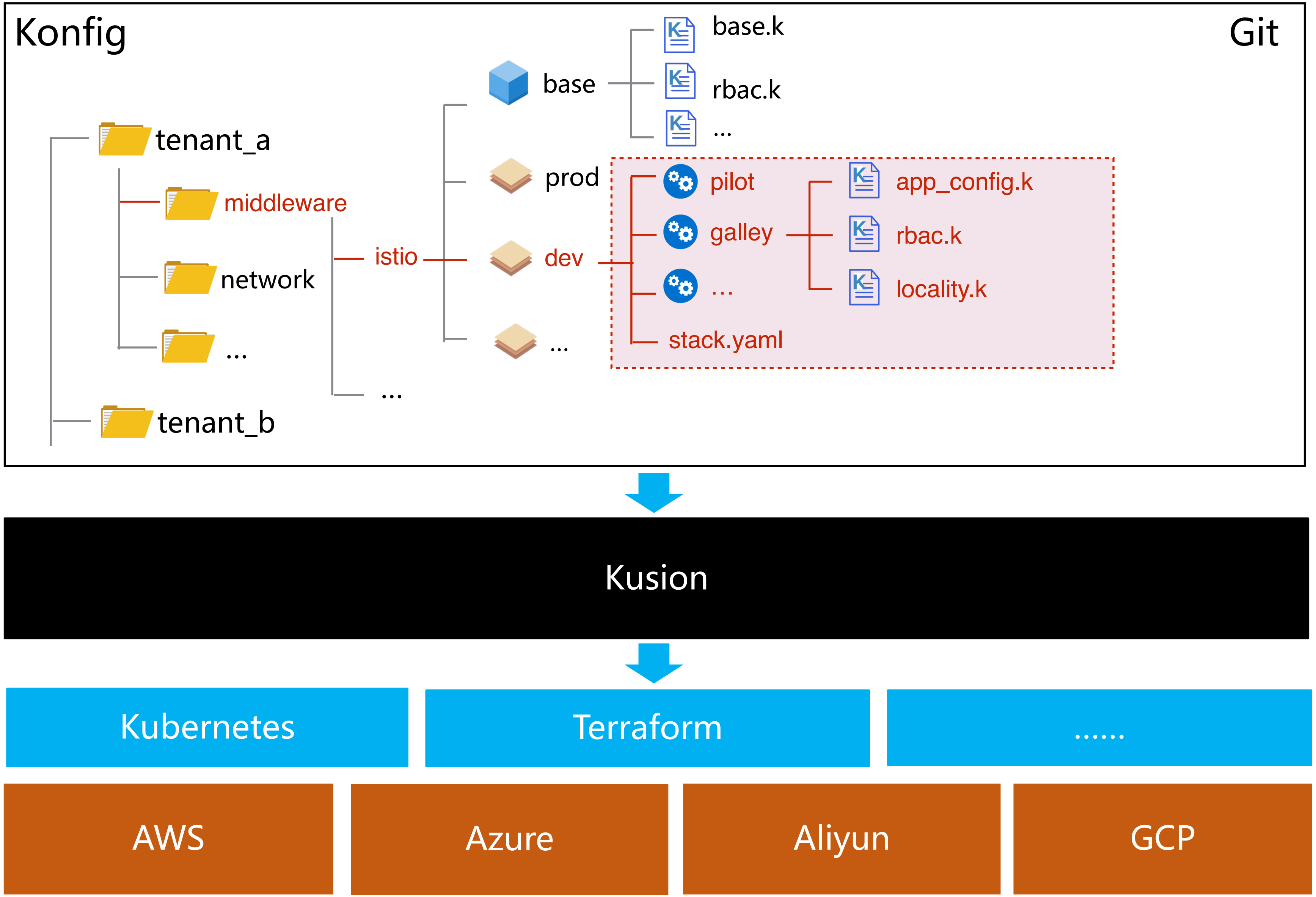
- Application developers can use platform capabilities independently
- Deliver platform apps using open-source tech and tools in an explicit way

Pain points

- Unable to invest more time in technology and R&D rather than user-side supporting
- Unable to make app developers to access platform capabilities in a uniform, stable and low-cost way

Multi-tenant, Multi- scenario, Multi-cloud

Centrally defined, globally delivery



Practice in AntGroup

Scaled

Ongoing app & infra delivery and ops

1500+

Projects

~100

Clusters

Practice

Efficiently enable business success

1K/day

Pipelines

10 K/day

KCL Compilations

1: 9

Plat: App Dev

6 scenarios

One-Stop

2 hours – 2 day

Feature Dev Period

200 - 300/day

Commits

Dev & SRE

Multi-Role DevOps

Hybrid cloud

Delivery

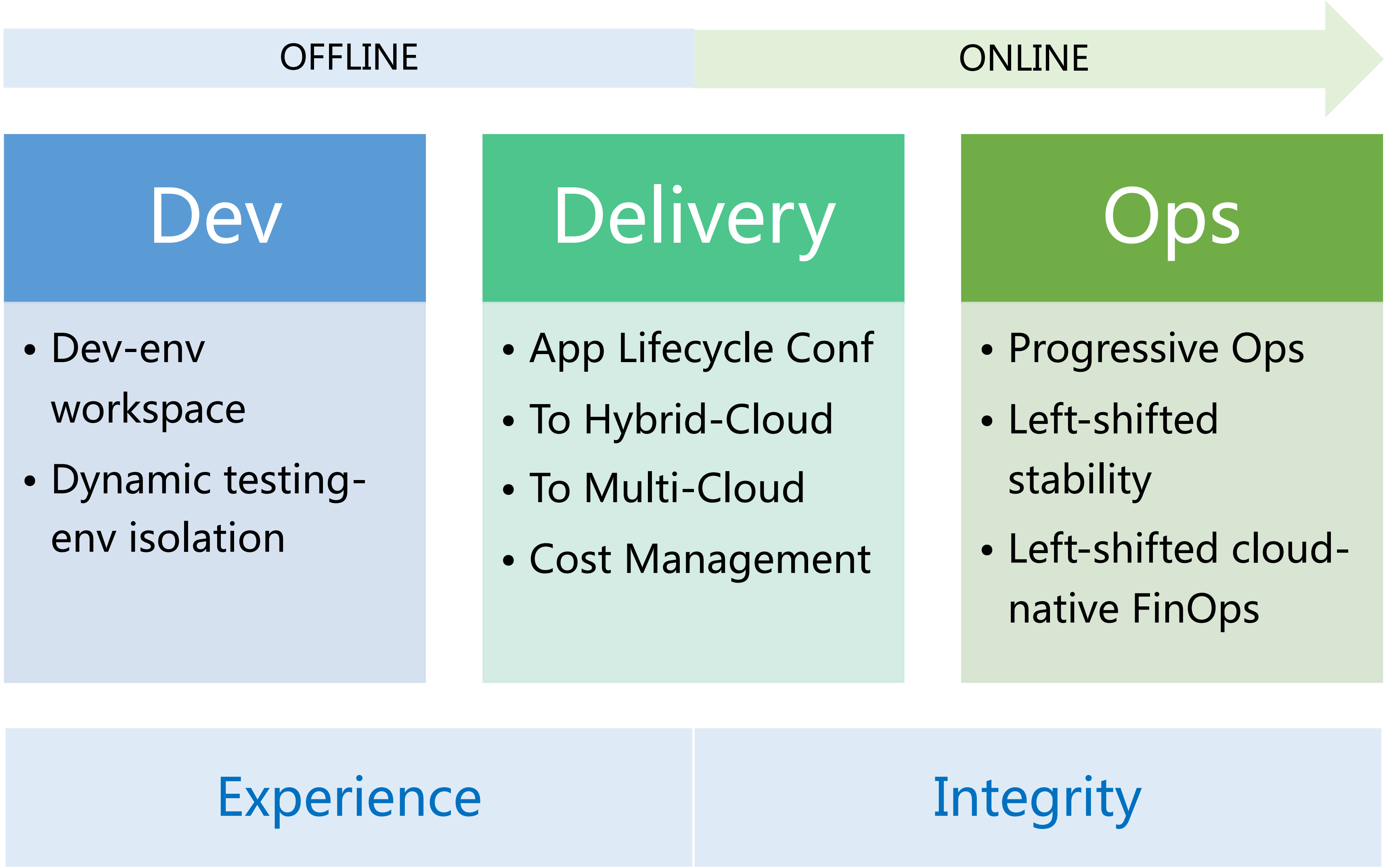
Culture

Precipitate engineer culture, share domain knowledge

~400	60000+	13000+	~600000
Contributors	Commits	PRs	KCL Code

Future

Scene



OFFLINE

ONLINE

Dev

- Dev-env workspace
- Dynamic testing-env isolation

Delivery

- App Lifecycle Conf
- To Hybrid-Cloud
- To Multi-Cloud
- Cost Management

Ops

- Progressive Ops
- Left-shifted stability
- Left-shifted cloud-native FinOps

Experience

Integrity

Tech Roadmap

KCL

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

v0.4.3

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

v0.5

- 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

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

v0.7

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

2022.9

2022.12

2023.3

2023.6

v0.7

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

v0.8

- **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

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

v0.10

- **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

Kusion & Konfig

Resources

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

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

- Twitter
  - @KusionStack

Thank you

KusionStack Team