



KUBERNETES

# 基于云原生供应链的配置策略管理新范式

## Kubernetes Resource Model (KRM) KCL Specification

演讲人：徐鹏飞

COMMUNITY DAYS  
HANGZHOU 2023



# 目录

01  
背景

02  
概念

03  
体验

04  
实践

CONTENTS

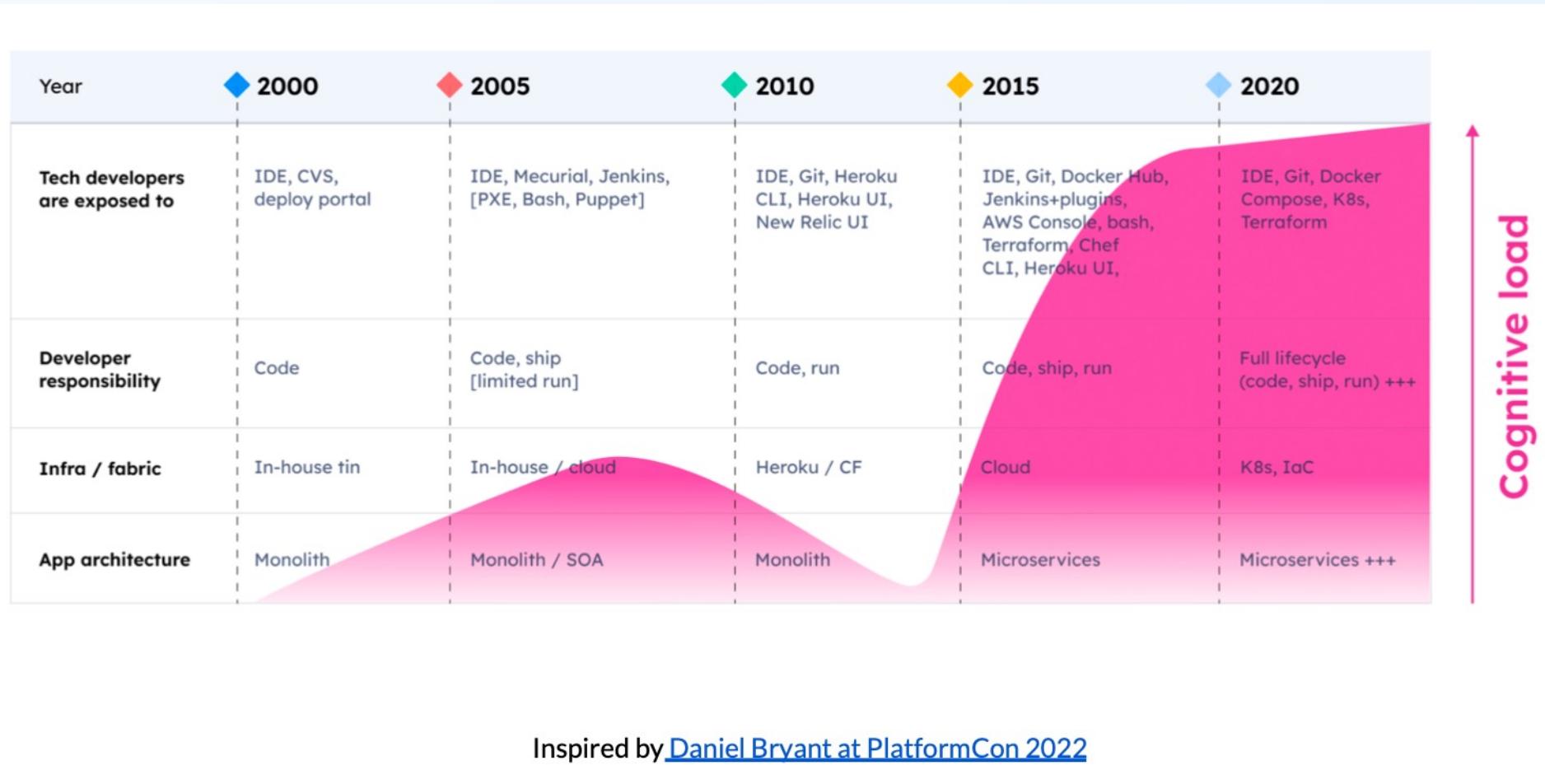
# 01. 背景

为什么我们需要一个新范式





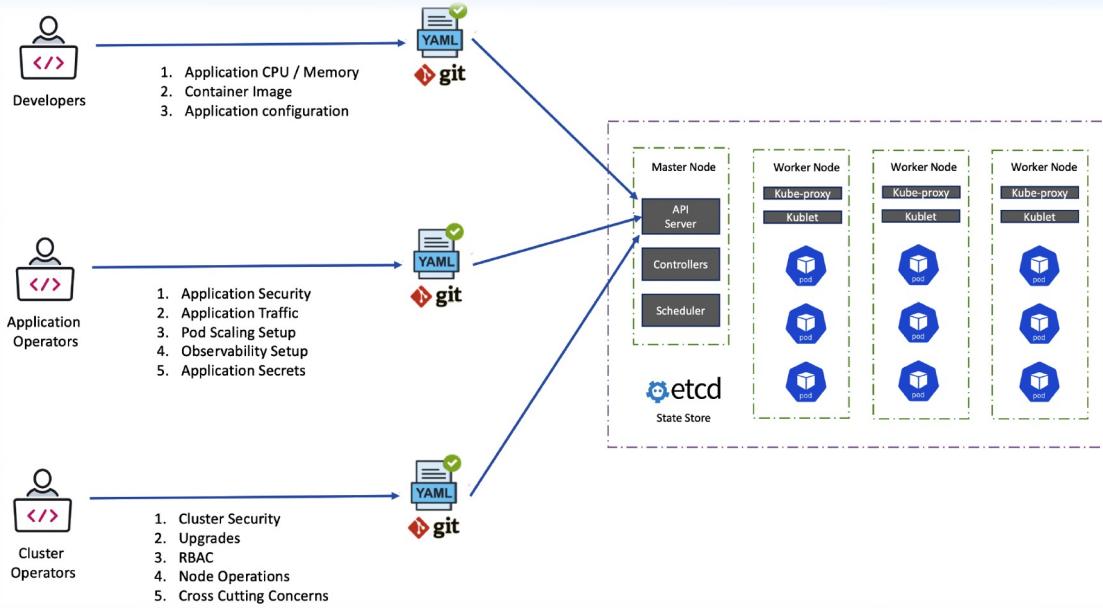
# 背景



随着云原生技术的发展，Kubernetes 底座和 Terraform 等 IaC 工具已成为越来越流行的管理和部署基于（云）API 的应用程序工具



# 问题



## 认知负担

- 应用开发人员需要面对复杂的基础设施和平台概念
- 不像云基础设施配置有 Terraform 等 IaC 工具，Kubernetes 是平台的平台，特别是在客户端缺乏轻量的配置组合和校验工具

## 静态配置

- YAML 膨胀，维度爆炸
- 跨团队配置协作负担和配置漂移

## 效率、可靠性低

- 缺乏标准的测试验证手段，大多是胶水代码或者脚本的拼盘
- 缺乏高效配置协同的工具，大多通过人肉拉群解决

减轻基础设施对开发人员的**认知负担**，提高配置管理**效率**

Kubernetes 中的声明式应用管理: [https://docs.google.com/document/d/1cLPGweVEYrVqQvBLJg6sxV-TrE5Rm2MNOBA\\_cxZP2WU/edit#](https://docs.google.com/document/d/1cLPGweVEYrVqQvBLJg6sxV-TrE5Rm2MNOBA_cxZP2WU/edit#)

CNCF 平台工程白皮书: <https://tag-app-delivery.cncf.io/whitepapers/platforms/>

Google SRE 工作手册: <https://sre.google/workbook/configuration-specifics/>

# 02. 概念

KCL、KRM KCL 规范、语言核心元素



面向云原生领域的专用配置策略语言 (2022.6 开源, 2023.9 成为 CNCF 基金会托管的 Sandbox 项目)

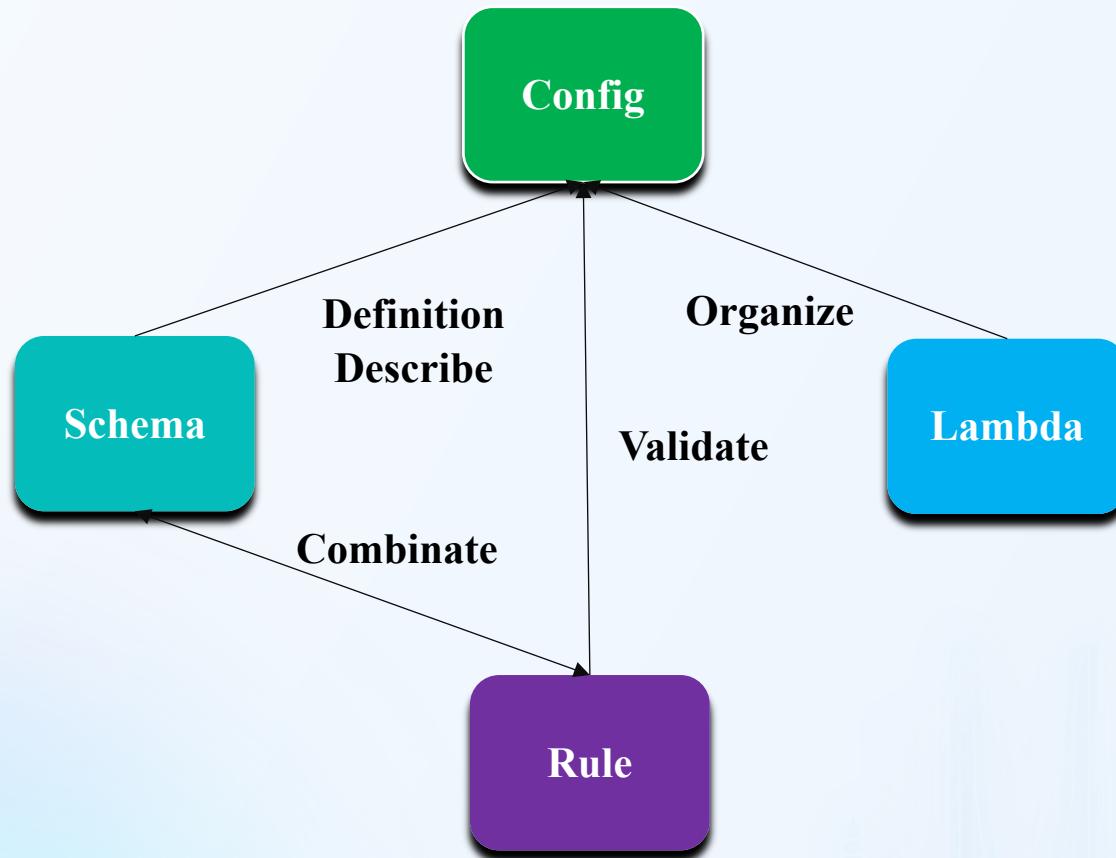


- ✓ **领域特定:** 以收敛的语言和工具集合解决领域问题近乎无限的变化和复杂性
- ✓ **以应用为中心:** 开发者可以理解的声明式应用配置模型
- ✓ **关注点分离:** 应用/平台 Dev/SRE
- ✓ **动态配置管理:** 多租户多环境、可编程可扩展
- ✓ **风险左移:** 实时的配置错误提示
- ✓ **可复用扩展:** OCI 等标准软件供应链集成和包管理工具支持, 官方 Registry 提供 60+ 模型包
- ✓ **引擎解耦:** 建立在一个完全开放的云原生世界当中, 几乎不与任何编排/引擎工具或者 Kubernetes 控制器绑定, 可同时为客户端和运行时场景提供 API 抽象、组合和校验的能力



## Config + Schema + Rule + Lambda

Pattern:  $k = (T)v$



```
import k8s.core.v1
# Create a Kubernetes Deployment resource.
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"
                ports = [{ containerPort = 80 }]
            ]
        }
    }
}
```

可复用可扩展、抽象和组合能力、稳定性、高性能



- Mutation

```
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
metadata:
  name: set-annotations
  metadata:
    annotations:
      krm.kcl.dev/version: 0.0.1
      krm.kcl.dev/type: mutation
      documentation: >-
        Add or change annotations
spec:
  params:
    toAdd: addValue
  source: oci://ghcr.io/kcl-lang/set-annotation
```

- Validation

```
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
metadata:
  name: https-only
  metadata:
    annotations:
      krm.kcl.dev/version: 0.0.1
      krm.kcl.dev/type: validation
      documentation: >-
        Requires Ingress resources to be HTTPS only. Ingress resources must
        include the `kubernetes.io/ingress.allow-http` annotation, set to `false`.
        By default a valid TLS {} configuration is required, this can be made
        optional by setting the `tlsOptional` parameter to `true`.
        More info: https://kubernetes.io/docs/concepts/services-networking/ingress/#tls
spec:
  source: oci://ghcr.io/kcl-lang/https-only
```

- Abstraction

```
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
metadata:
  name: web-service
  metadata:
    annotations:
      krm.kcl.dev/version: 0.0.1
      krm.kcl.dev/type: abstraction
      documentation: >-
        Web service application abstraction
spec:
  params:
    name: app
    containers:
      nginx:
        image: nginx
        ports:
          containerPort: 80
    labels:
      name: app
  source: oci://ghcr.io/kcl-lang/web-service
```

input KRM items

functionConfig

KCL Function

output KRM items

results

- 遵循统一的 KRM Function 规范
- 多种代码源支持: OCI, Git, Https, Filesystem...
- 可编程可扩展: 使用 KCL 语言简单编写模型

# 03. 体验

语言、工具、云原生集成



# Workspace



## Language + Tools + IDEs + IDEs + SDKs + Plugins

The screenshot shows a code editor interface with several panels:

- Top Bar:** Icons for R, Python, and Go.
- Left Sidebar:** A file tree for a project named "KONFIG". It includes subfolders like ".github", ".kclvm", "appops", "clickhouse-operator", "prod", "guestbook", "http-echo", "nginx-example", "base", and "clouds".
- Code Editor:** A snippet of K8s configuration in main.k. The code defines a "server" object with an "image" of "altinity/clickhouse-operator:0.19.2" and a "sidecarContainers" array containing a "Sidecar" object with an "image" of "altinity/metrics-exporter:0.19.2".
- KCL Package Manager:** A panel showing the "KCL Coding Assistant" with buttons for Highlight, Format, Go To Def/Ref, Compile, Completion, Debug, Error/Warning Checking, and Test. It also shows the "KCL Language Server" and "KCL Compiler".
- Bottom Panel:** A "Tools & CI/CD Engagement" section showing a workflow: kcl-format → kcl-lint → kcl-test → kcl-doc. Each step has a green checkmark icon.



# IDE Extension



- VS Code

main.k — König

```
main.k 3, M X

appops > clickhouse-operator > prod > main.k
1 import base.pkg.kusion_models.kube.frontend
2 import base. expected one of ["identifier", "literal", "(", "[", "{"]
3 examples
4 # The applic pkg
5 # the configuration with the same attribute
6 appConfiguration: frontend.Server {
7     # spec.template.spec.containers[0], mai
8     image = "altinity/clickhouse-operator:0
9
10    # spec.template.spec.containers[1:], si
11    sidecarContainers = [
12        s.Sidecar {
13            name = "metrics-exporter"
14            image = "altinity/metrics-exporter"
15            resource = ""
16        }
17    ]
18 }
19 }
```

- Idea

hello-kcl – hello.k

```
import .templates.resources

schema Server:
    """
    Server schema describes the de...
    """
    name: str
    # todo: image must be set dyna...
    image: str
    replica: int = 1 # default to ...
    resources: {str:str}

myApp = Server{
    name: "myApp",
    image: "demo/myApp",
    resources: resources.large
}
```

- NeoVim

nginx.k • x

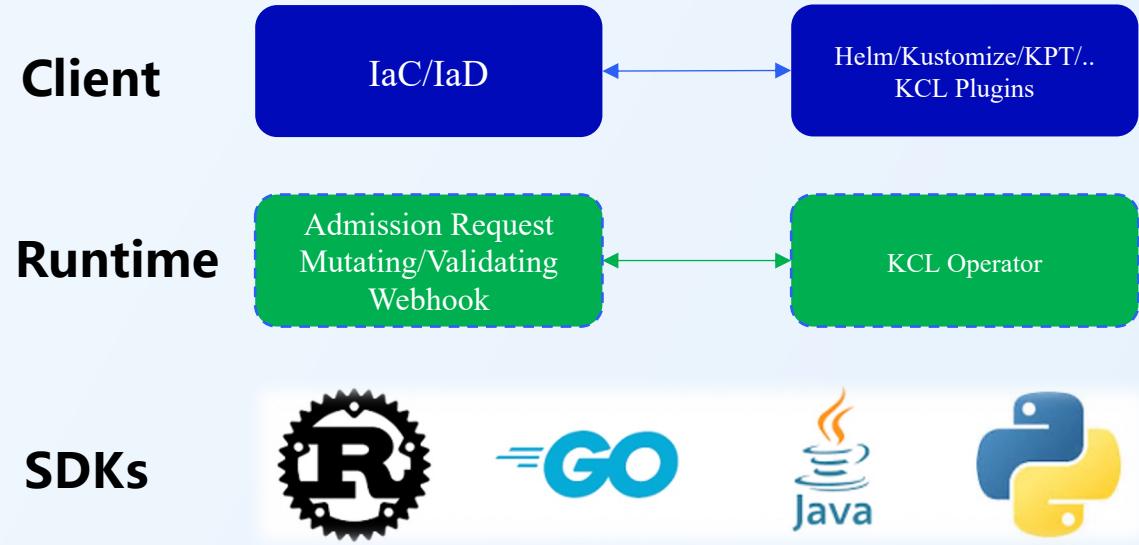
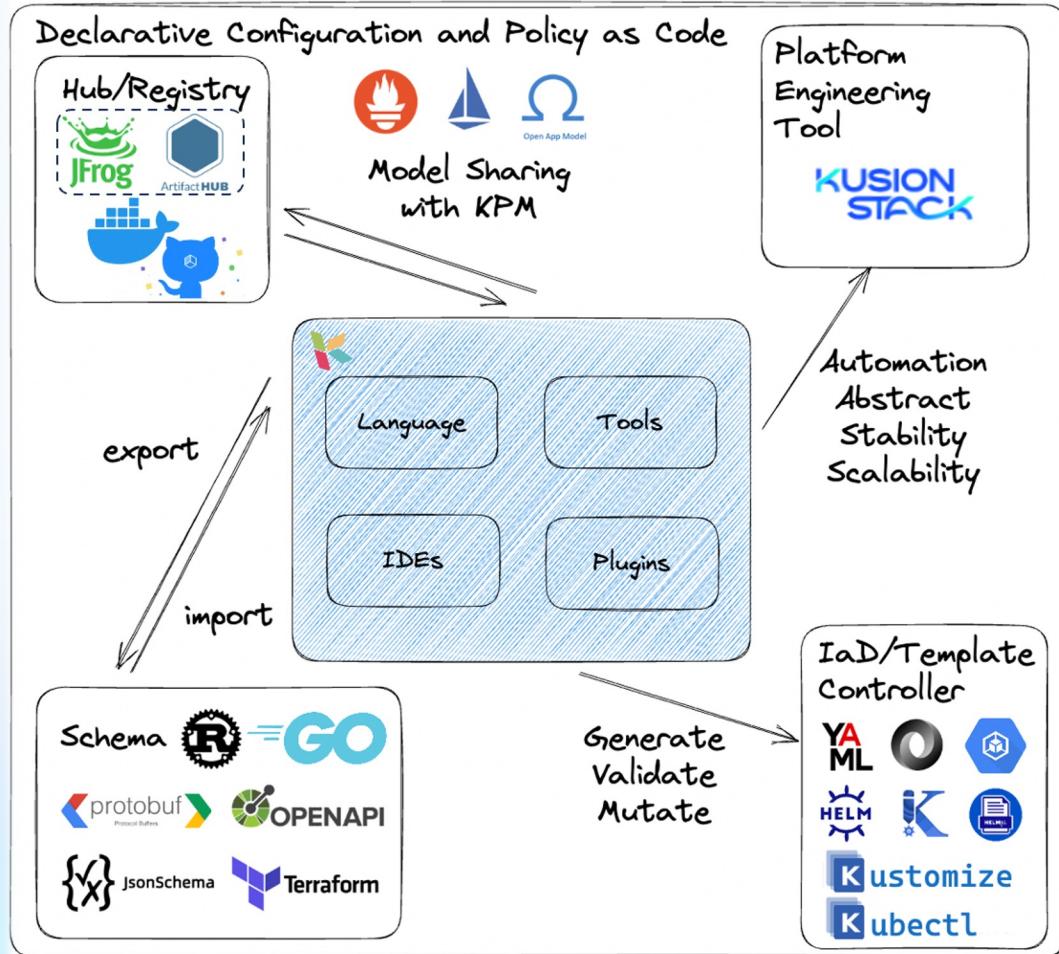
```
29 import_json ■ Module 'json' imported but unused
30 schema Nginx:
31     """Schema for Nginx configuration files"""
32     http: Http ■ name 'Http' is not defined
33
34     # schema Http:
35     #     server: Server
36
37 schema Server:
38     listen: int | str # The attribute `listen` can be int type or a string type.
39     location?: Location # Optional, but must be non-empty when specified
40
41 schema Location:
42     root: str
43     index: str
44
45 schema Person:
46     name: str
47     age: int
48
49 # schema Foo:
50
51 x = Person{
52     name: "foo"
53     age:1 ■ expected one of ["identifier", "literal", "(", "[", "{"] got newline
54 }
55
56 nginx = Nginx {
57     http.server = [
58         listen = 80
59         location = [
60             root = "/var/www/html"
61             index = "index.html"
62         ]
63     ]
64 }
```

configuration/nginx.k 3

- name 'Http' is not defined (CompileError) [5, 11]
- expected one of ['identifier', 'literal', '(', '[', '{'] got newline (InvalidSyntax) [30, 9]
- Module 'json' imported but unused (UnusedImportWarning) [1, 1]

main kcl 17 2 2 ▲1

# Integration



- **多语言 SDK:** Rust, Go, Python, Java SDK
- **包管理支持:** KPM 工具和多种 Registry /Hub 支持
- **数据和 Schema 集成:** KCL Import/Export 工具
- **运行时集成:** 使用 KCL Operator 而不是重复开发 Kubernetes Admission Webhook
- **KRM 支持:** 统一的规范和插件支持 e.g., kubectl-kcl plugin, helm-kcl plugin, helmfile-kcl plugin, kustomize-kcl plugin, kpt-kcl-plugin ...
- **平台工程工具支持:** 作为 DCM 语言配合不同引擎/编排器进行应用交付



# Artifact Hub KCL Integration (Staging)



Artifact HUB DOCS STATS

Artifact Hub is a web-based application that enables finding, installing, and publishing packages and configurations for CNCF projects. For example, this could include Helm charts and plugins, Falco configurations, Open Policy Agent (OPA) and Gatekeeper policies, OLM operators, Tinkerbell actions, kubectl plugins, Tekton tasks and pipelines, KEDA scalers, CoreDNS plugins, Keptn integrations, container images, Kubewarden policies, Kyverno policies, Knative client, Backstage plugins, Argo templates, KubeArmor policies and KCL packages.

Argo templates

Backstage plugins

Container images

CoreDNS plugins

Falco rules

Helm charts and plugins

KCL packages

KEDA scalers

Keptn integrations

Knative client plugins

Kubectl plugins

KubeArmor policies

Kubewarden policies

Kyverno policies

OLM operators

OPA and Gatekeeper policies

Tekton tasks and pipelines

Tinkerbell actions

*ArtifactHub KCL Package 预览版: <https://staging.artifacthub.io/>*



# Registry & Modules



The screenshot shows the left sidebar of the Artifact Hub website with a dark theme. The sidebar includes links for Welcome, TOPICS (Repositories, Argo templates, Backstage plugins, Containers images, CoreDNS plugins, Falco rules, Gatekeeper policies, Helm charts, Helm plugins, KCL packages), KEDA scalers, Keptn integrations, and k8s. The k8s section is expanded, showing items like 'Published on Jul 18 by The KCL Programming Language' for 'k8s', 'set-annotation', and 'web-service'. The main content area is titled 'KCL packages repositories' and explains how KCL packages repositories are expected to be hosted in GitHub, GitLab or Bitbucket repos. It provides URL formats for GitHub, GitLab, and Bitbucket, notes that the master branch is used by default, and provides a code snippet for repository structure.

Documentation / Topics / Repositories / KCL packages

## KCL packages repositories

KCL packages repositories are expected to be hosted in GitHub, GitLab or Bitbucket repos. When adding to Artifact Hub, the url used **must** follow the following format:

- `https://github.com/user/repo[/path/to/packages]`
- `https://gitlab.com/user/repo[/path/to/packages]`
- `https://bitbucket.org/user/repo[/path/to/packages]`

By default the `master` branch is used, but it's possible to specify a different one from the UI.

Please NOTE that the repository URL used when adding the repository to Artifact Hub **must NOT** contain platform specific parts, like `tree/branch`, just the path to your packages like it would show in the filesystem.

The `path/to/packages` provided can contain metadata for one or more packages. Each package version separate folder, and it's up to you to decide if you want to publish one or multiple versions of your package.

The structure of a repository with multiple packages and versions could look something like this:

```
$ tree path/to/packages
path/to/packages
└── artifacthub-repo
    └── vml
```

- **开箱即用:** 一行命令添加依赖 e.g., `kpm add k8s`, 现阶段官网模型 **60+** (欢迎共建)
- **多种场景支持:** 配置编辑、校验、模型抽象, Kubernetes 生态模型, Terraform 生态模型, 应用配置, ...
- **多种 Registry /Hub 支持:** Docker Hub, ghcr.io, Harbor, ...
- **Artifact Hub KCL Integration (WIP):** 支持浏览器查询、身份认证等

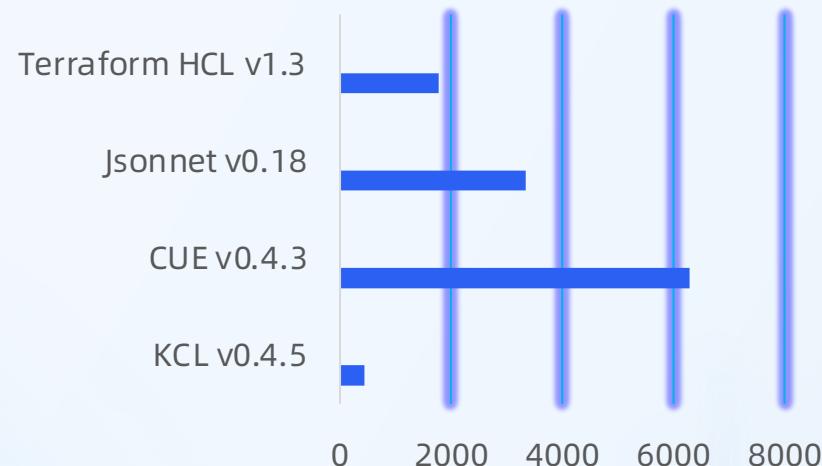


# Performance



## Loop and Function

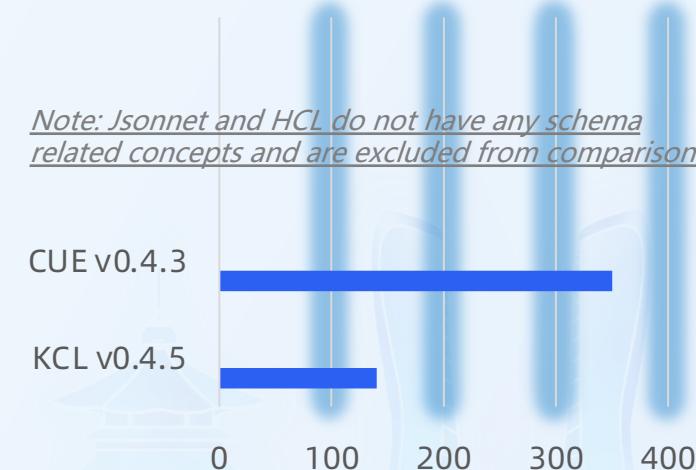
```
a = lambda x: int, y: int -> int {  
    max([x, y])  
}  
temp = {"a${i}": a(1, 2) for i in range(10000)}
```



*Test environment: single core macOS 10.15.7 CPU: i7-8850H 2.6GHz 32GB 2400Mhz DDR4 No NUMA, e2e run time (ms)*

## Kubernetes Configuration

```
import kubernetes.api.apps.v1  
  
deployment = v1.Deployment {}
```



*Note: Jsonnet and HCL do not have any schema related concepts and are excluded from comparison*

# 04. 实践

Mutation、Validation、Abstraction





# Kubernetes Mutation



## Before

```
9 db/           1 import os
10 kube-manifests/ 2 from os import listdir
11   k8s-sidekiq.yml 3
12   k8s-cms-web.yml 4 import ruamel.yaml
13   test-k8s-cms-web.yml.b 5
14 lib/          6 yaml = ruamel.yaml.YAML()
15 log/          7 yaml.preserve_quotes = True
16 node_modules/ 8 yaml.explicit_start = True
17 ops/scripts/ 9
18   __init__.py 10 def main():
19     db_migrate.sh* 11   home_dir = os.path.join(os.path.dirname(__file__), "../../kube-manifests")
20     gen_data.sh* 12   print(home_dir)
21     gen_pod_env.py 13   for manifest in listdir(home_dir):
22     gen_sed_cmd.py 14     if not manifest.endswith("yaml"):
23     merge_key.py 15       continue
24     secrets_manager_utils. 16     p = os.path.join(home_dir, manifest)
25   public/ 17     with open(p, "r") as f:
26     script/ 18       content = yaml.load_all(f)
27     spec/ 19     res = []
28   storage/ 20     for item in content:
29     tmp/ 21       if item["kind"].lower() == "deployment":
30     ALLOWED_PASSWORDS 22         cs = item["spec"]["template"]["spec"]["containers"]
31     app.json 23         for c in cs:
32     babel.config.js 24           c["env"].append({
33     bitbucket-pipelines.yml 25             "name": "test_name",
34     buildspec.yml 26             "value": "test_value"
35     CHANGELOG.md 27           })
36     cms-pre-production-23070 28         res.append(item)
37     cms-pre-production-23070 29     print(res)
38     cms-pre-production-23070 30     with open(p, "w") as f:
39     cms-pre-production-23070 31       yaml.dump_all(res, f)
40     cms-pre-production-23070 32
41     cms-pre-production-23070 33 if __name__ == '__main__':
42     cms-pre-production-23070 34     main()
43     cms-pre-production-23070
44     cms-pre-production-23070
45     cms-prod-230705-buildspel
46     cms-prod-230705-buildspel
47     cms-prod-230705-dockerfil
gyuchen/fuse/fuseigned-cms NORMAL ops/scripts/secrets_manager_utils.py
ops/scripts/secrets_manager_utils.py 34L 1089R 3M 3X
```

## After

```
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
spec:
  params:
    env:
      - name: test_name
        value: test value
  source: oci://ghcr.io/kcl-lang/append-env
```

OCI Registry 上存放的 KCL 代码包

```
items = [item | {
  if item.kind == "Deployment":
    spec.template.spec.containers: [
      env += option("params").env
    } for container in item.spec.template.spec.containers]
} for item in option("items") or []]
```

- 避免脚本、胶水代码拼盘
- 可复用，可测试

<https://github.com/kcl-lang/krm-kcl>



# Kubernetes Validation

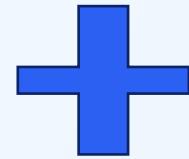


Disallow Service Load Balancer Module

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
  type: LoadBalancer # 错误地设置了 LoadBalancer
```

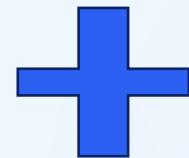
Allow Https Only Module

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  # Required the annotation kubernetes.io/ingress.allow-http: "false"
  name: tls-example-ingress
spec:
  tls:
    - hosts:
        - https-example.foo.com
        secretName: testsecret-tls
  rules:
    - host: https-example.foo.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: service1
                port:
                  number: 80
```



```
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
metadata:
  name: disallow-svc-lb
spec:
  source: oci://ghcr.io/kcl-lang/disallow-svc-lb
```

将 K8s Manifests 和 KCL 策略同时使用，检查 K8s Manifests 配置是否合规



```
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
metadata:
  name: https-only
spec:
  source: oci://ghcr.io/kcl-lang/https-only
```

- 统一界面：KCL 可以同时编写 Schema 和约束条件，无需配合 OpenAPI Schema/JSON Schema 使用

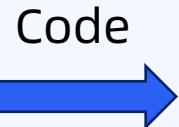
<https://github.com/kcl-lang/krm-kcl>



# Kubernetes Abstraction



UI/CLI/API



Code

## Standalone KCL Form

```
import .app

app.App {
    name = "app"
    containers.nginx = {
        image = "nginx"
        ports = [{containerPort = 80}]
    }
    service.ports = [{ port = 80 }]
}
```

## KRM KCL Form

```
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
metadata:
  name: web-service
  annotations:
    krm.kcl.dev/version: 0.0.1
    krm.kcl.dev/type: abstraction
    documentation: >-
      Web service application abstraction
spec:
  params:
    name: app
  containers:
    nginx:
      image: nginx
      ports:
        containerPort: 80
    labels:
      name: app
  source: oci://ghcr.io/kcl-lang/web-service
```

## Generate



## Kubernetes Manifests

```
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
  namespace: nginx-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app.kubernetes.io/name: nginx-example
  template:
    metadata:
      labels:
        app.kubernetes.io/env: dev
        app.kubernetes.io/instance: nginx-example-dev
        app.kubernetes.io/component: nginx-example-dev
        app.kubernetes.io/instance: nginx-example-dev
    spec:
      containers:
        - image: nginx:latest
          name: main
          ports:
            - containerPort: 80
              protocol: TCP
          resources:
            limits:
              cpu: 100m
              memory: 100Mi
            ephemeral-storage: 1Gi
          requests:
            cpu: 100m
            memory: 100Mi
            ephemeral-storage: 1Gi
```

- Kubernetes API 自由组合/抽象，关注点分离

<https://github.com/kcl-lang/krm-kcl>



- Step 1. 在集群当中安装 KCL Operator
- Step 2. Apply KCL and K8s manifests

```
kubectl apply -f- << EOF
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
metadata:
  name: set-annotation
spec:
  params:
    annotations:
      managed-by: kcl-operator
      source: oci://ghcr.io/kcl-lang/set-annotation
EOF
```

```
kubectl apply -f- << EOF
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  annotations:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
EOF
```

- Step 3. 获得资源 Mutation/Validation 结果

```
kubectl get po nginx -o yaml | grep kcl-operator
managed-by: kcl-operator
```

- 少数几行 KCL 代码即可完成对应配置编辑/校验功能 (客户端和运行时代码可以复用)
- 无需开发额外的 Kubernetes Webhook 编辑和验证配置
- 多种代码源支持 OCI, Git, Https, Filesystem, ...

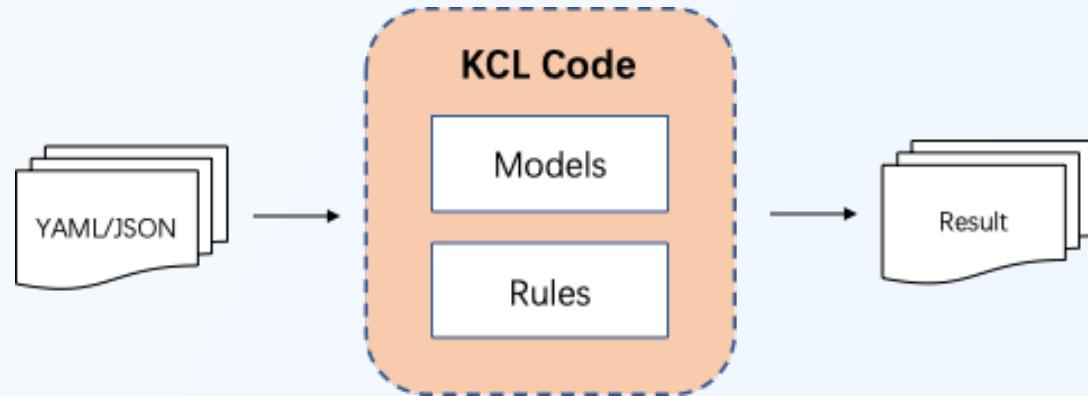


# Terraform

## Configuration Validation & Generation



### Validation Process

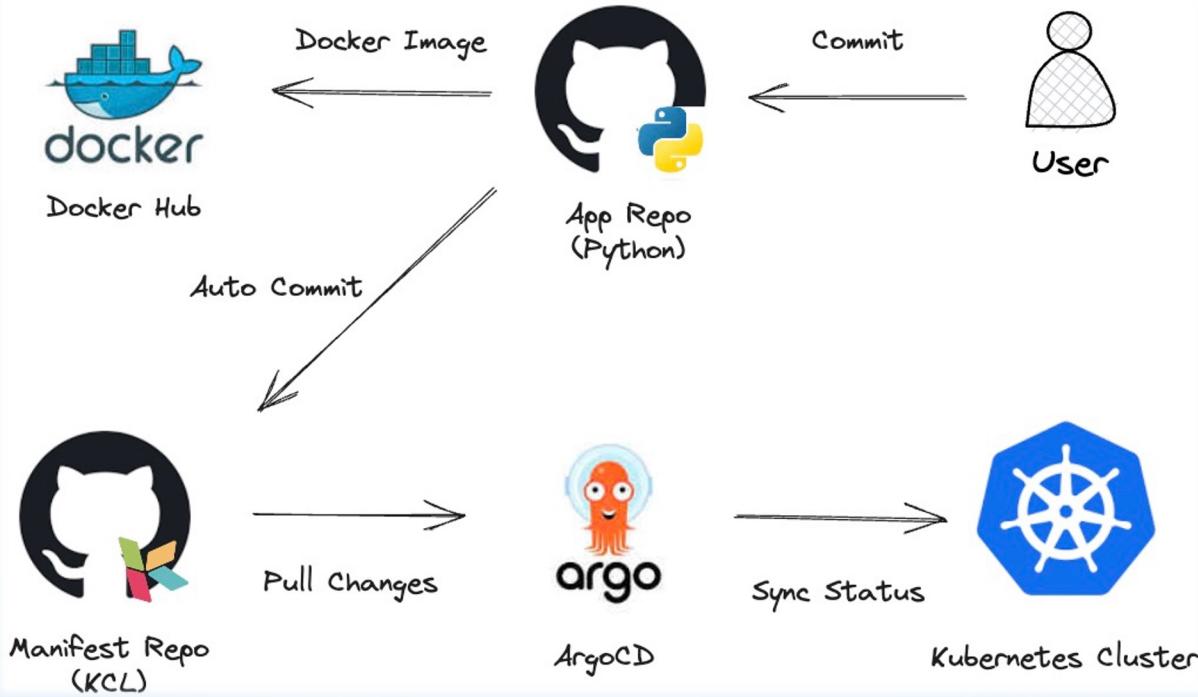


- ✓ **多种数据支持:** JSON/YAML/...
- ✓ **结构定义:** Schema 结构化定义及自定义错误支持
- ✓ **生态集成:** OpenAPI/Terraform Provider Schema 转换 KCL Schema 支持
- ✓ **开箱即用:** 丰富的配置策略模型库和代码示例 (欢迎共建)



kcl-vet tfplan.json main.k

[https://kcl-lang.io/docs/user\\_docs/guides/working-with-terraform/validation](https://kcl-lang.io/docs/user_docs/guides/working-with-terraform/validation)



## Commit

```
kcl code set image to kclang/flask_demo:6428cff4309afc8c1c40ad180bb9...  
...cf82546be3e
```

main

github-actions[bot] committed 3 minutes ago

Showing 1 changed file with 1 addition and 1 deletion.

```
diff --git a/main.k8s b/main.k8s
--- a/main.k8s
+++ b/main.k8s
@@ -3,7 +3,7 @@ config = app.App {
  name = "flask_demo"
  containers: [
    flask_demo = {
-      image = "kclang/flask_demo:f1f2cbc0c4555d141e9f642fb12edaf34d0b723"
+      image = "kclang/flask_demo:6428cff4309afc8c1c40ad180bb9cf82546be3e"
    }
  ]
}
```

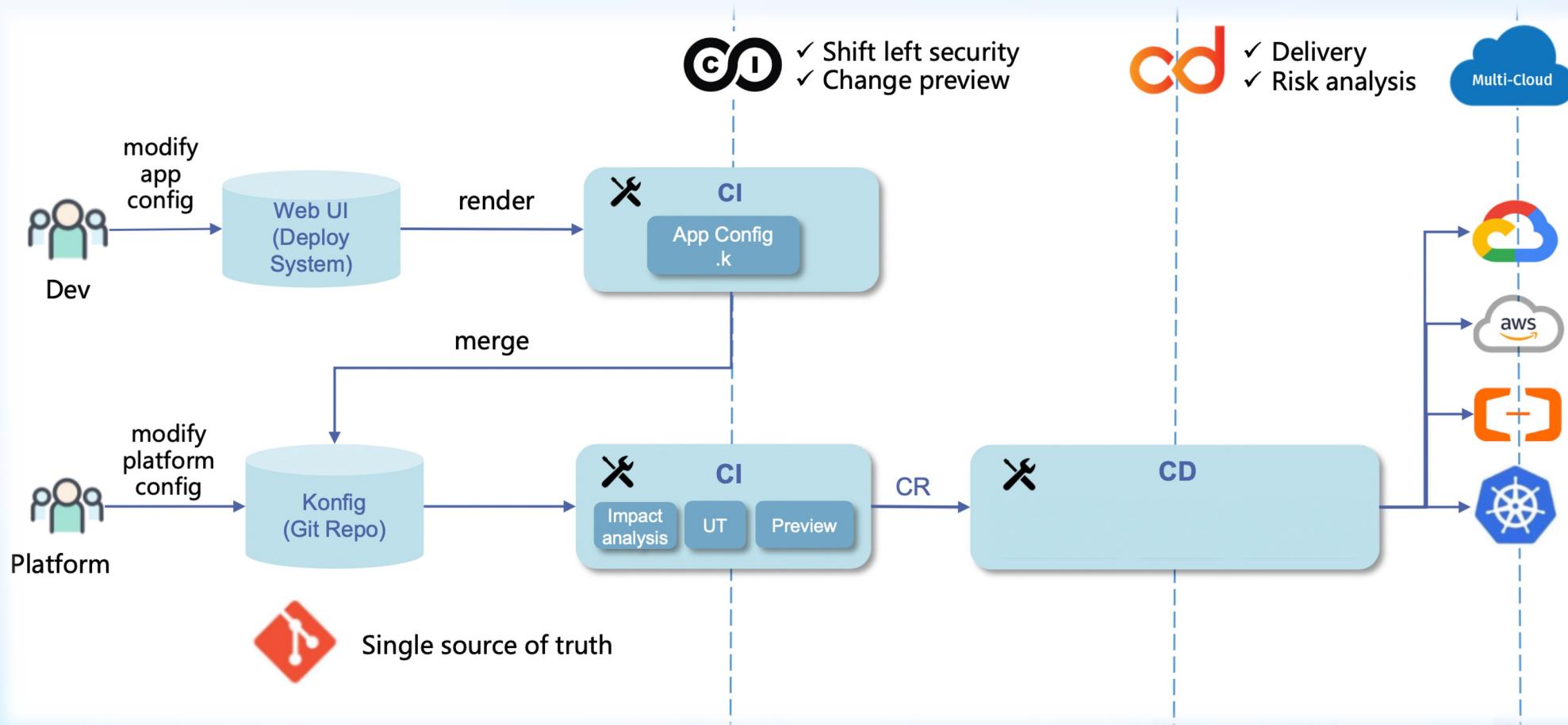
配置驱动的工作流：同时支持 *Standalone KCL* 和 *KRM KCL* 配置格式  
多种 CI/CD 和 GitOps 工具支持 e.g., ArgoCD

[https://kcl-lang.io/docs/user\\_docs/guides/gitops/gitops-quick-start](https://kcl-lang.io/docs/user_docs/guides/gitops/gitops-quick-start)



# App Delivery

For Kubernetes and Cloud Resources



多种支持应用交付引擎支持: 如 *KusionStack, KubeVela*

<https://kusionstack.io>  
<https://kubevela.net>



# App Delivery

Demo



```
→ bin git:(watch-ui) ./kusion apply -w /Users/yuanhao/opensource/KusionStack/konfig/base/examples/server/app_service/prod --watch --yes
✓ Compiling in stack prod...
```



# Mutation, Validation, Abstraction

## Production-Ready

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

- 从云原生配置策略及软件供应链层面解决**认知负担高、静态配置、效率和稳定性等问题**
- 通过定义合适的 API 抽象，配置编辑策略隐藏基础设施和平台细节，减轻开发人员的负担。
- 通过更现代的声明式配置策略语言和工具，KRM KCL 规范, OCI Registry 和 Artifact Hub 等，帮助不同团队/角色之间更轻松地共享、传播和交付配置和策略。 (**欢迎共建配置和策略模型** 



## More Resources



- **官方网站**

- <https://kcl-lang.io/>
- <https://kusionstack.io/>

- **GitHub**

- <https://github.com/kcl-lang>
- <https://github.com/kusionStack>

- **Twitter**

- [@kcl language](https://twitter.com/kcl_language)

- **Slack**

- *CNCF KCL Slack Channel: <https://cloud-native.slack.com/archives/C05TC96NWN8>*

**钉钉(DingTalk ID 42753001)**



**微信公众号**





# THANKS

感谢您的观看

COMMUNITY DAYS  
HANGZHOU 2023

KUBERNETES





# Appendix

## Community Projects



### 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
- Helm
- Helmfile
- ...

### Programmable KV

#### Pros.

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

#### Cons.

- Insufficient type constraints
- Insufficient restraint ability
- Runtime error

#### Tech.

- GCL
- HCL
- Bicep
- Starlark
- Jsonnet
- CEL
- OPA/Rego
- ...

#### Product

- Terraform
- Tanka
- Radius
- ytt
- kpt

### 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
- Nickel
- ...

#### 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
- KRM-KCL Tools and Operators