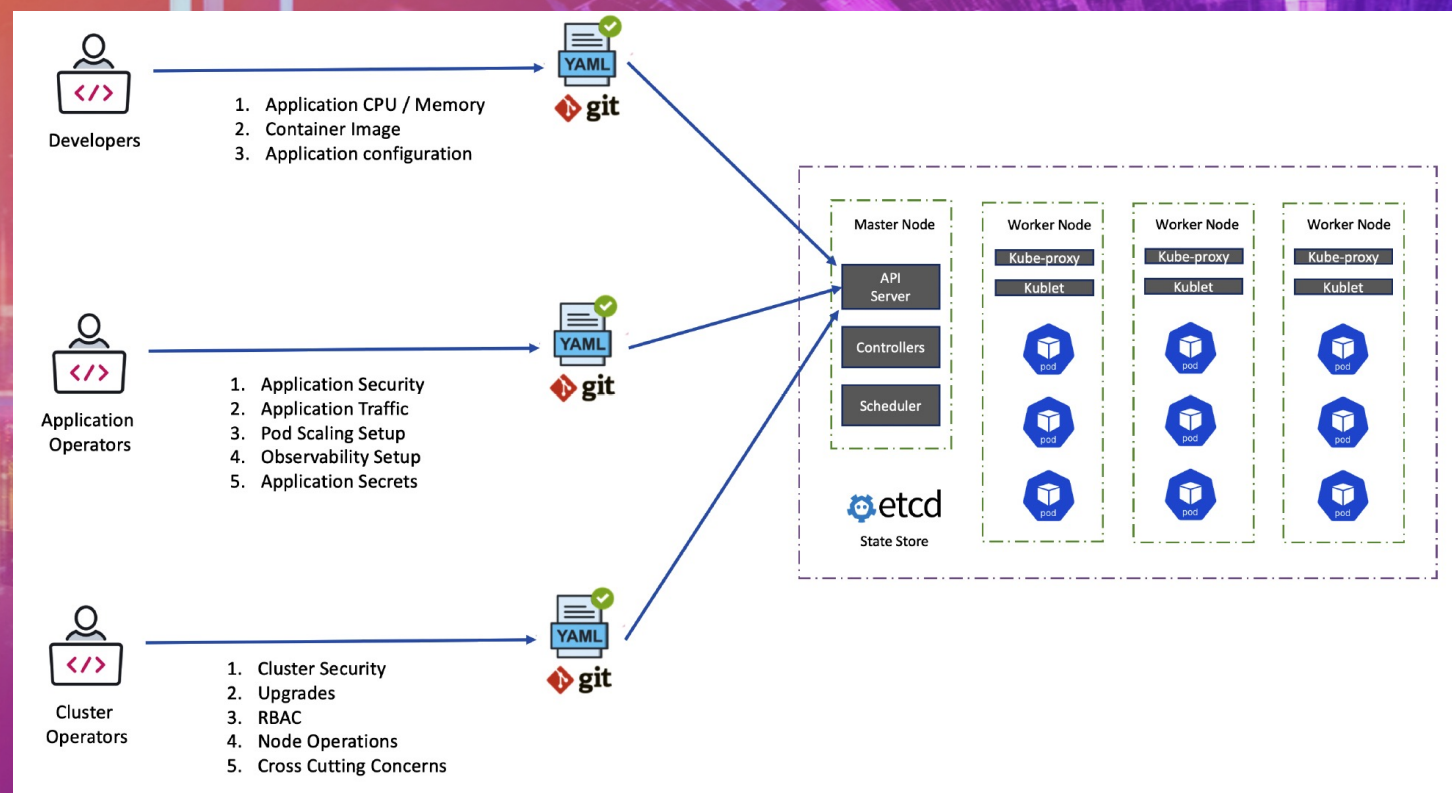


# KCL 配置策略语言

演讲者：徐鹏飞



# 背景



## 认知负担

- K8s 等基础设施概念复杂 (500+ 模型, 2000+ 字段)
- 复杂应用 YAML 配置繁琐 (10000+ 行)

## 静态配置缺陷

- 碎片化配置维度爆炸, 难以收敛 (多环境、多租户)
- 三方应用配置难以定制和修改

## 效率/可靠性低

- 无统一的工程技术栈, 配置变更效率, 稳定性和扩展性较差, 复用程度低 (<10%)
- 缺乏标准的测试、校验和约束手段, 通常辅以难以维护和复杂的脚本或胶水代码

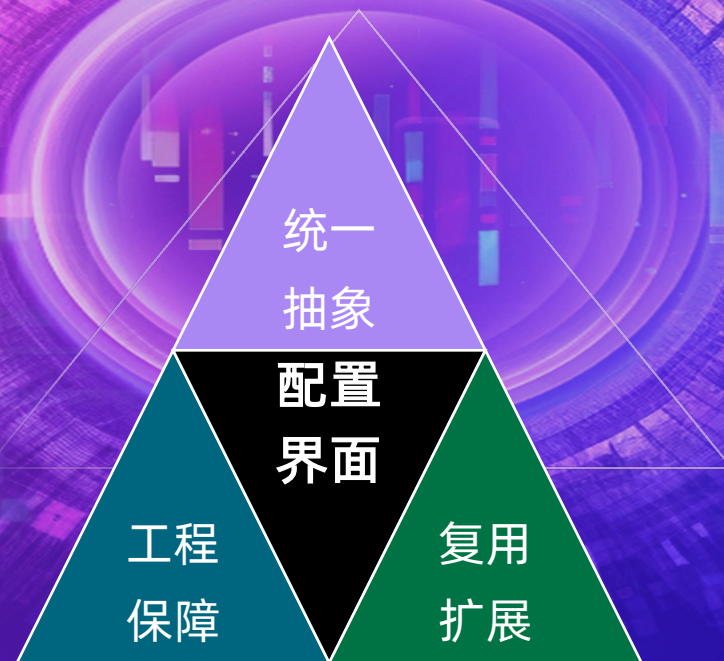
诉求: 降低基础设施对开发者负担, 提高配置管理效率



# 技术路径

问题抽象：规模化配置管理问题

问题拆解：配置管理问题的核心是配置语言问题，上层的自动化系统和业务流程围绕语言展开



技术路径：通过重塑配置界面和动态配置管理屏蔽基础设施和平台细节降低认知负担，提高配置管理效率



# 概览

Language + Tools + IDEs + SDKs + Plugins

The image displays a development environment for KCL (Kubernetes Configuration Language). The top bar features logos for Python, Go, and other languages, along with IDE icons for VS Code, IntelliJ, and Vim.

**Left Panel (KONFIG):** A file explorer showing a project structure under 'appops > clickhouse-operator > prod'. Files include OWNERS, project.yaml, and README.md.

**Center Panel (Code Editor):** Shows a KCL file named 'main.k' with the following content:

```
1 import base.pkg.kusion_models.kube.frontend
2
3 # The application configuration in stack will overwrite
4 # the configuration with the same attribute in base.
5 server: frontend.Server {
6   # spec.template.spec.containers[0], main container
7   image = "altinity/clickhouse-operator:0.19.2"
8
9   # spec.template.spec.containers[1:], sidecars
10  sidecarContainers = [
11    s.Sidecar {
12      name = "metrics-exporter"
13      image = "altinity/metrics-exporter:0.19.2"
14      resource = ""
15    }
16  ]
17 }
18
```

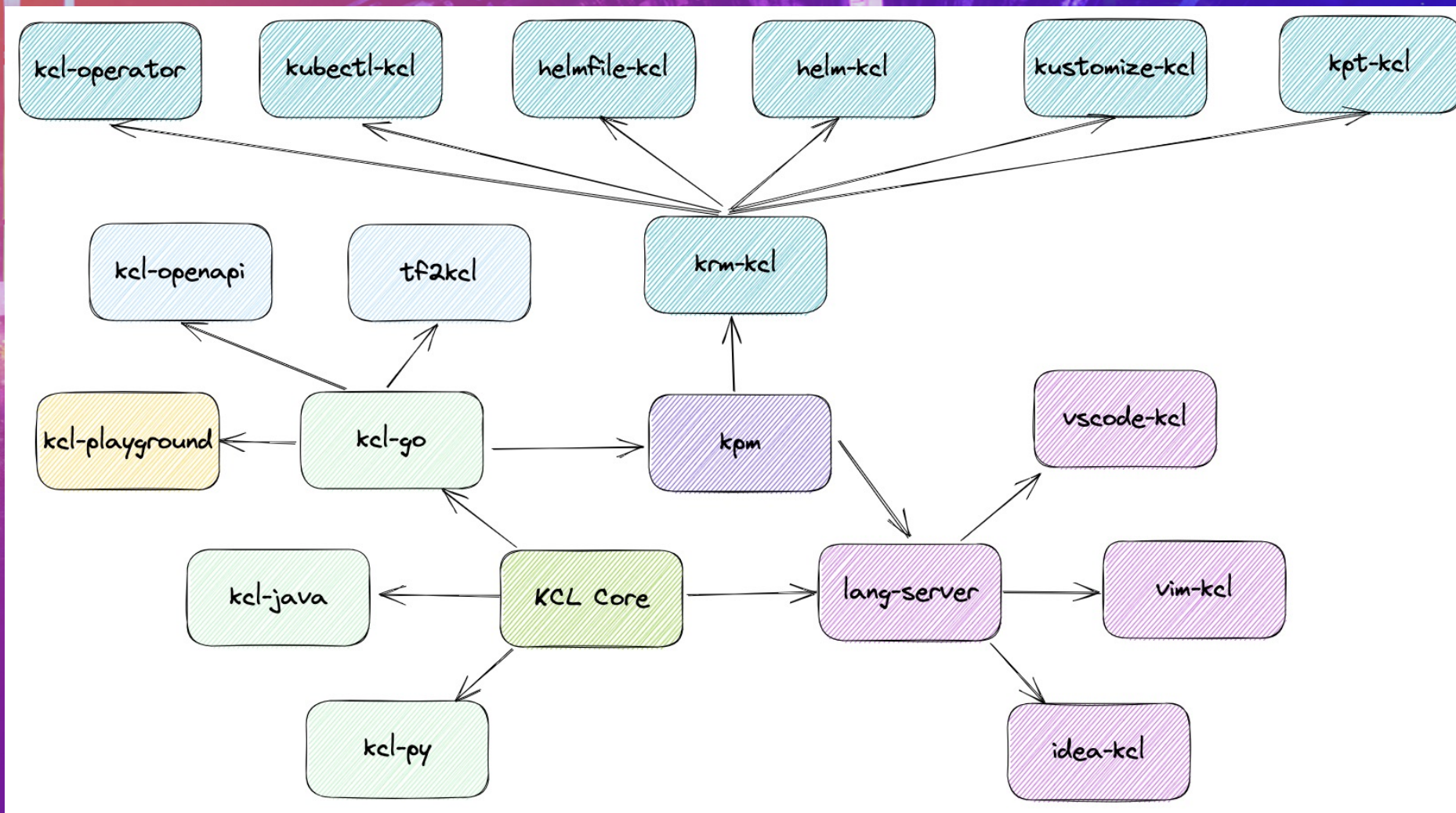
**Right Panel (KCL Package Manager):** A diagram showing the toolchain components:

- KCL Coding Assistant:** Contains functions like Highlight, Format, Go To Def/Ref, Compile, Completion, Debug, Error/Warning Checking, and Test.
- KCL Language Server:** Connected to the Coding Assistant via LSP (Language Server Protocol).
- KCL Compiler:** The underlying compiler.

**Bottom Panel (Tools & CI/CD Engagement):** A workflow diagram showing a sequence of tools: kcl-format, kcl-lint, kcl-test, and kcl-doc, each with a green checkmark indicating successful execution. Arrows indicate the flow from left to right, with a return arrow from the end back to the start.



# 组成

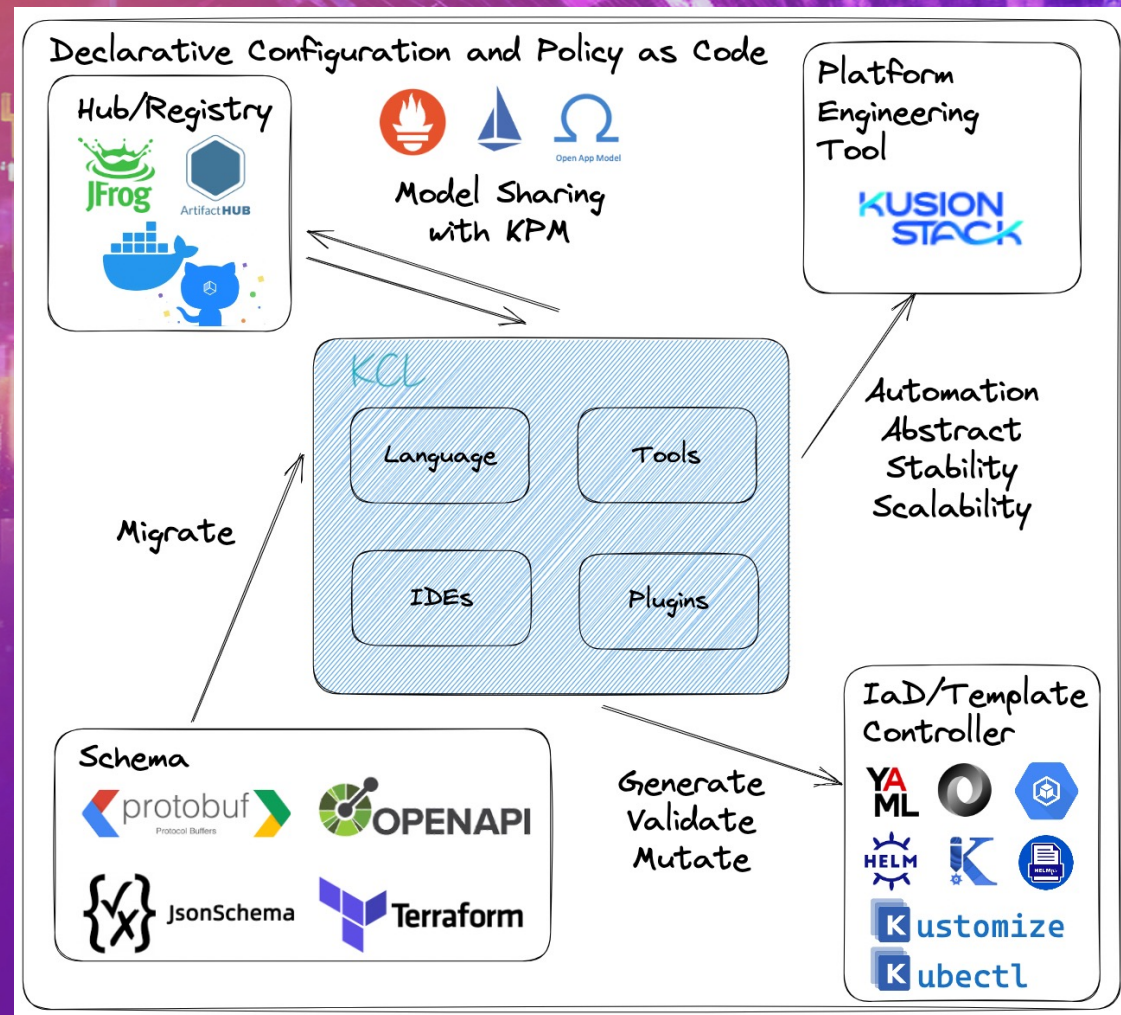


## How to (For App/Platform dev & SRE)

- Lang
- SDKs
- Tools
- IDE
- Playground
- UI
- Cloud-native Tool Integrations



# 集成



Client

IaC/IaD

Helm/Kustomize/KPT/..  
KCL Plugins

Server

Admission Request

Mutating/Validating  
Webhook

Binding



Enhancement/Glue

- **KRM Support:** Unified KRM KCL Spec
- **Migration:** Data, Schema
- **Sharing:** Modules, Functions
- **GitOps:** ArgoCD Integration
- **Lang Binding**



# KRM KCL 规范

## • 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://kusionstack/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://kusionstack/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://kusionstack/web-service
```

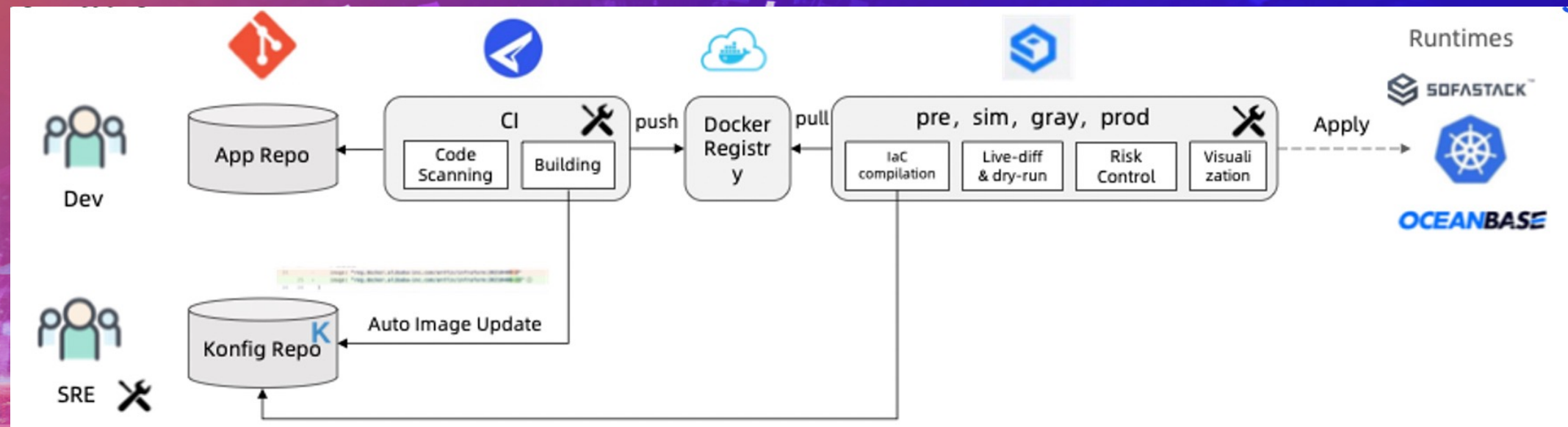
## Guides for Developing KCL

Here's what you can do in the KCL script:

- Read resources from `option("resource_list")`. The `option("resource_list")` complies with the [KRM Functions Specification](#). You can read the input resources from `option("resource_list")["items"]` and the `functionConfig` from `option("resource_list")["functionConfig"]`.
- Return a KPM list for output resources.
- Return an error using `assert {condition}, {error_message}`.
- Read the environment variables. e.g. `option("PATH")` (Not yet implemented).
- Read the OpenAPI schema. e.g. `option("open_api")["definitions"]["io.k8s.api.apps.v1.Deployment"]` (Not yet implemented).



# 自动化



```
1 import base.pkg.kusion_models.kube.frontend
2 import base.pkg.kusion_models.kube.frontend.service
3 import base.pkg.kusion_models.kube.frontend.container
4 import base.pkg.kusion_models.kube.templates.resource as res_tpl
5
6 # Application Configuration
7 appConfiguration: frontend.Server {
8   # Main Container Configuration
9   mainContainer = container.Main {
10     ports = [
11       {containerPort = 80}
12     ]
13   }
14-  image = "nginx:1.7.8"
15 }
16
```

```
1 import base.pkg.kusion_models.kube.frontend
2 import base.pkg.kusion_models.kube.frontend.service
3 import base.pkg.kusion_models.kube.frontend.container
4 import base.pkg.kusion_models.kube.templates.resource as res_tpl
5
6 # Application Configuration
7 appConfiguration: frontend.Server {
8   # Main Container Configuration
9   mainContainer = container.Main {
10     ports = [
11       {containerPort = 80}
12     ]
13   }
14+  image = "nginx:1.7.9"
15 }
16
```



# 示例

The image shows a screenshot of the Visual Studio Code editor interface. On the left, a file explorer shows a project structure under 'KONFIG [CODESPACES]' with folders like '.github', '.kclvm', and 'appops'. The 'prod' folder is expanded, showing files like 'main.k' which is selected. The main editor area displays the content of 'main.k', which is a Kusion configuration file. A context menu is open over the code, listing various actions such as '转到定义' (Go to Definition), 'Find All References', '更改所有匹配项' (Change All Matches), and '命令面板...' (Command Palette...). The terminal at the bottom shows a 'bash' prompt and some output text.

```
main.k
1 import base.pkg.kusion_models.kube.frontend
2
3 # The application
4 # the configuration
5 appConfiguration
6 # spec.template
7 image = "alibabacloud/k8s-frontend"
8
9 # spec.template
10 sidecarContainers
11   s.SidecarContainer
12     name
13     image
14     resources
15   }
16 }
17 }
18
```

Context Menu Options:

- 转到定义 (Go to Definition) - ⌘F12
- 转到引用 (Go to References) - ⌘F12
- 快速查看 (Quick Peek) - >
- Find All References - ⌘⇧F12
- 更改所有匹配项 (Change All Matches) - ⌘F2
- 格式化文档 (Format Document) - ⌘⇧F
- 重构... (Refactor...) - ⌘⇧R
- 共享 (Share) - >
- 剪切 (Cut)
- 复制 (Copy)
- 复制为 (Copy As) - >
- 粘贴 (Paste)
- 添加到监视 (Add to Watch)
- 运行到光标处 (Run to Cursor)
- Kusion: Compile
- Kusion: Preview Live Diff and Apply
- 命令面板... (Command Palette...) - ⌘⇧P



# 联系我们

- **Website**
  - <https://kcl-lang.io/>
  - <https://kusionstack.io/>
- **Repo**
  - <https://github.com/kcl-lang/kcl>
  - <https://github.com/KusionStack/kusion>
  - <https://github.com/KusionStack/konfig>
- **Community**
  - <https://github.com/KusionStack/community#contact>
  - <https://github.com/KusionStack/community>
- **Twitter**
  - [@KusionStack](https://twitter.com/KusionStack)



扫一扫群二维码，立刻加入该群。

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