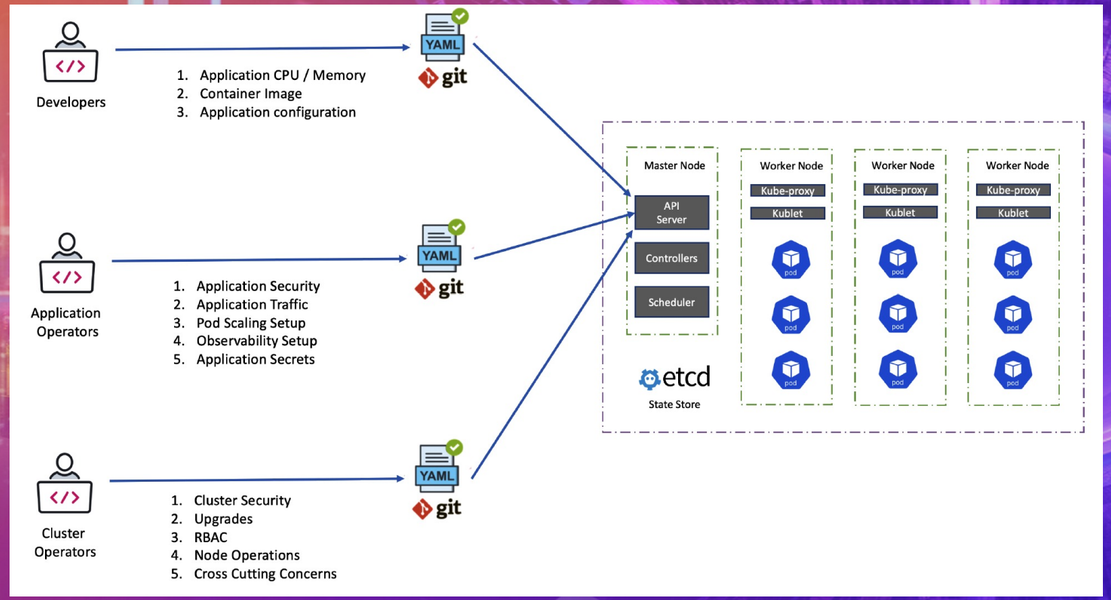
# KCL 配置策略语言

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

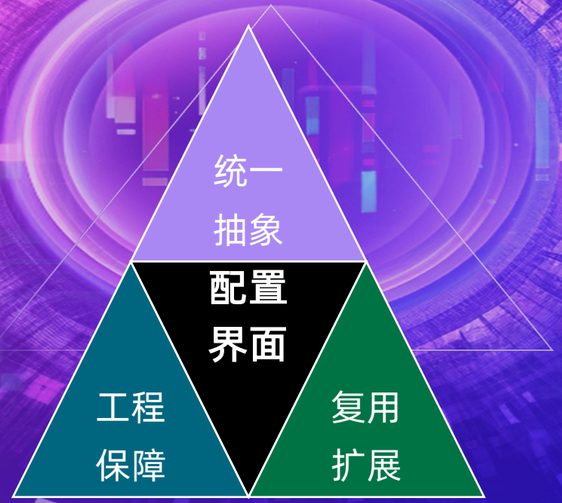
**诉求：**降低基础设施对开发者负担，提高配置管理效率

*Declarative Application Management in Kubernetes: https://docs.google.com/document/d/1cLPGweVEYrVqOvBLJg6sxV-TrE5Rm2MNOBA_cxZP2WU/edit#*
*Google SRE Workbook: https://sre.google/workbook/configuration-specifics/*

# 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

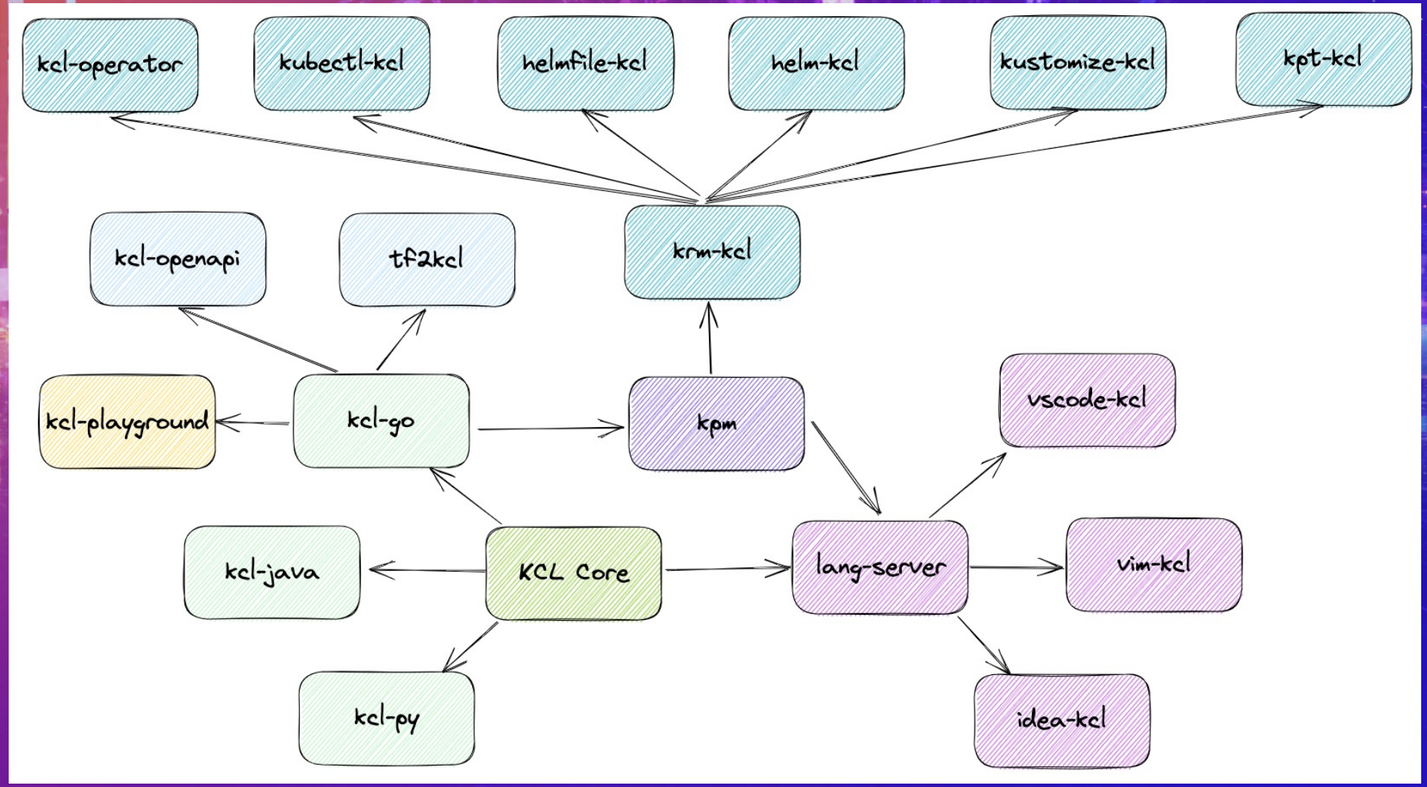



**技术路径：**通过重塑配置界面和动态配置管理屏蔽基础设施和平台细节
降低认知负担，提高配置管理效率

# 概览

**Language + Tools + IDEs + SDKs + Plugins**

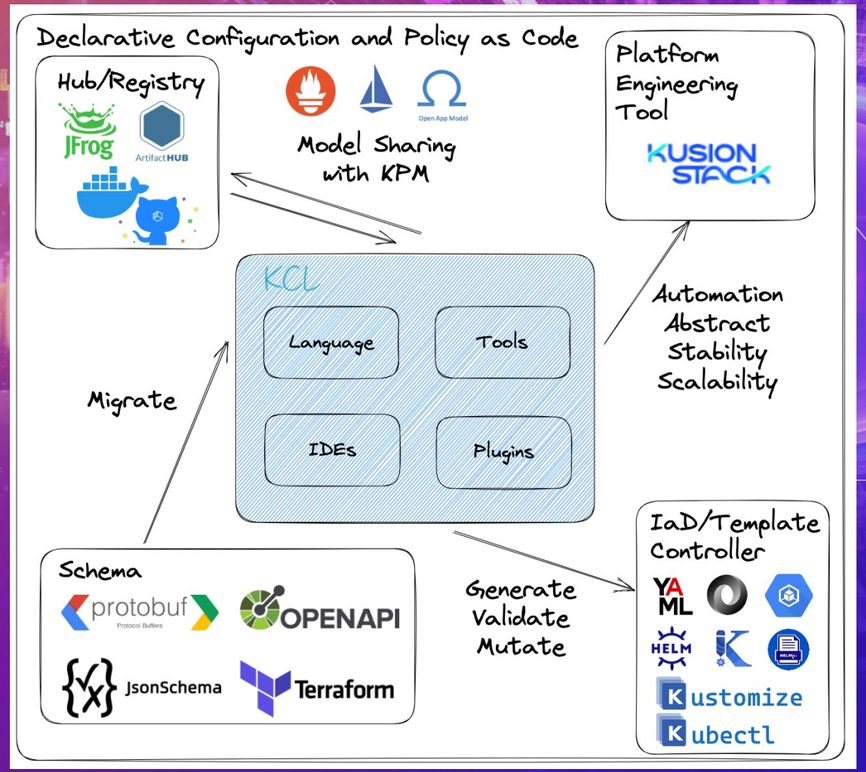
# 组成




**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** Rust, GO, Java, Python

**Enhancement/Glue**

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

# KRM KCL 规范

- **Mutation**

```yaml
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**

```yaml
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**

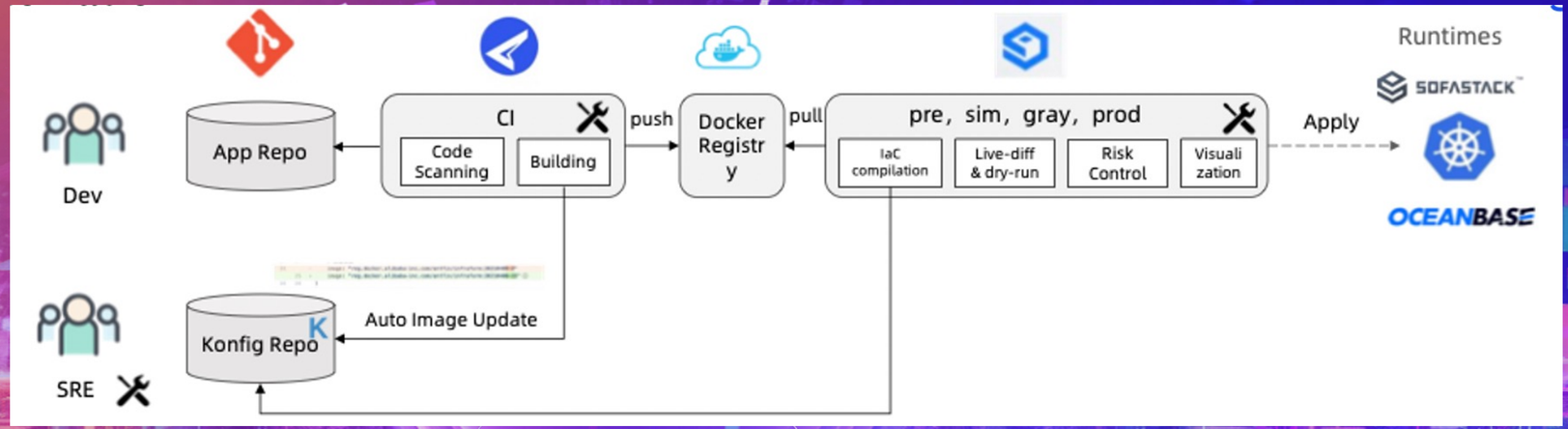
```yaml
apiVersion: krm.kcl.dev/v1alpha1
kind: KCLRun
metadata:
  name: web-service
  metadata:
    annotations:
      krm.kcl.dev/version: 0.0.1
      krm.kcl.dev/type: abstraction
      documentation: >-
        Web service application abstraction
spec:
  params:
    name: app
    containers:
      ngnix:
        image: ngnix
        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



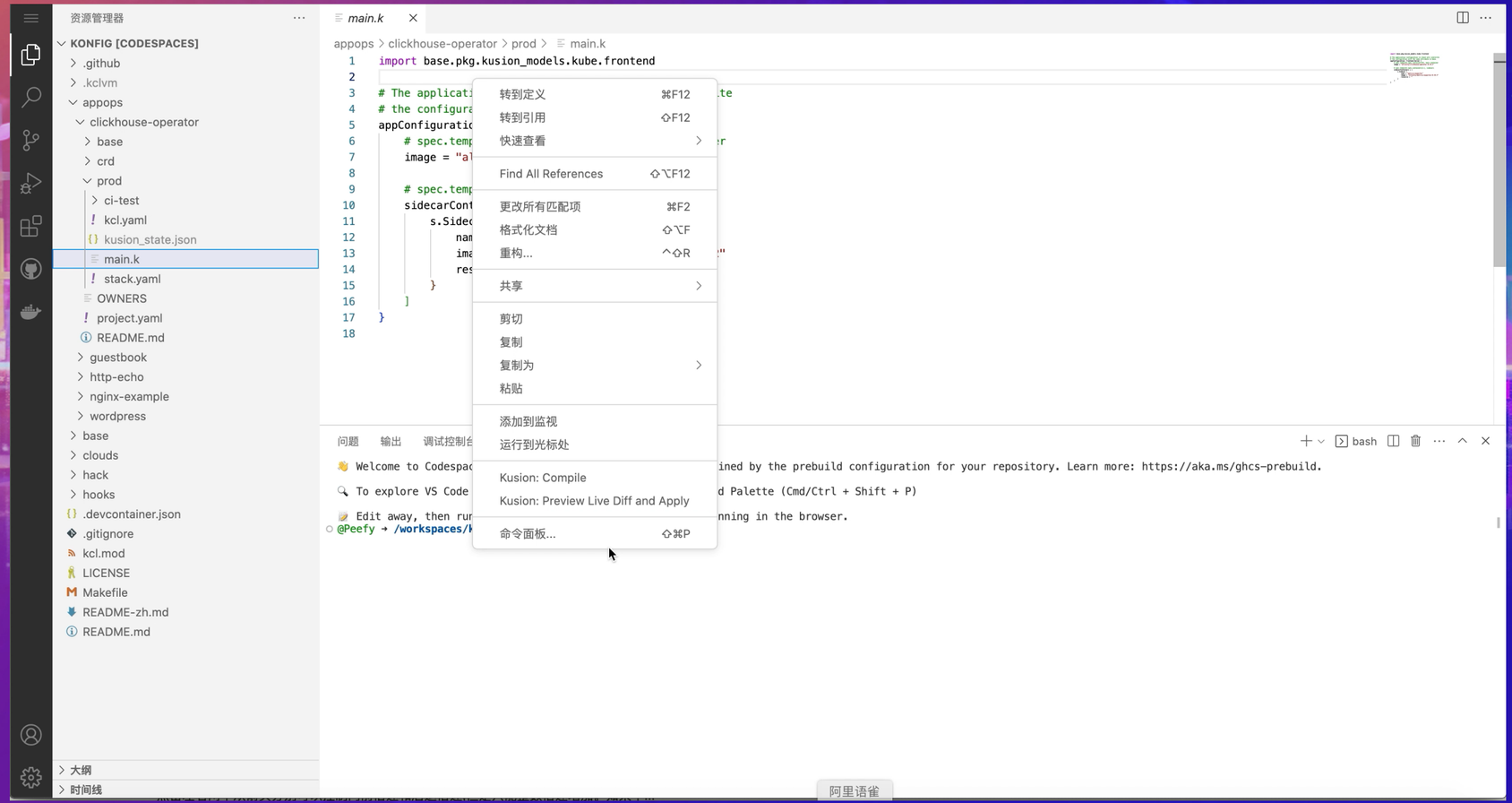

```
import base.pkg.kusion_models.kube.frontend
import base.pkg.kusion_models.kube.frontend.service
import base.pkg.kusion_models.kube.frontend.container
import base.pkg.kusion_models.kube.templates.resource as res_tpl

# 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"
}
```

# 示例


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

# 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


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