

ArchGuard 架构治理



ArchGuard

开源架构治理平台

ArchGuard 是一个主要针对于分布式场景下的架构治理平台。它可以在开发过程中,帮助架构师、开发人员分析系统间的远程服务依赖情况、数据库依赖、API 依赖等;根据架构评估模型,对于整体架构进行评估并提出改进建议,达到架构治理的目标。



微信公众号

1.

架构可视化

(基于 C4 模型可视化
系统现状)

2.

架构分析

(代码、服务、数据库、
代码变更等)

3.

架构治理

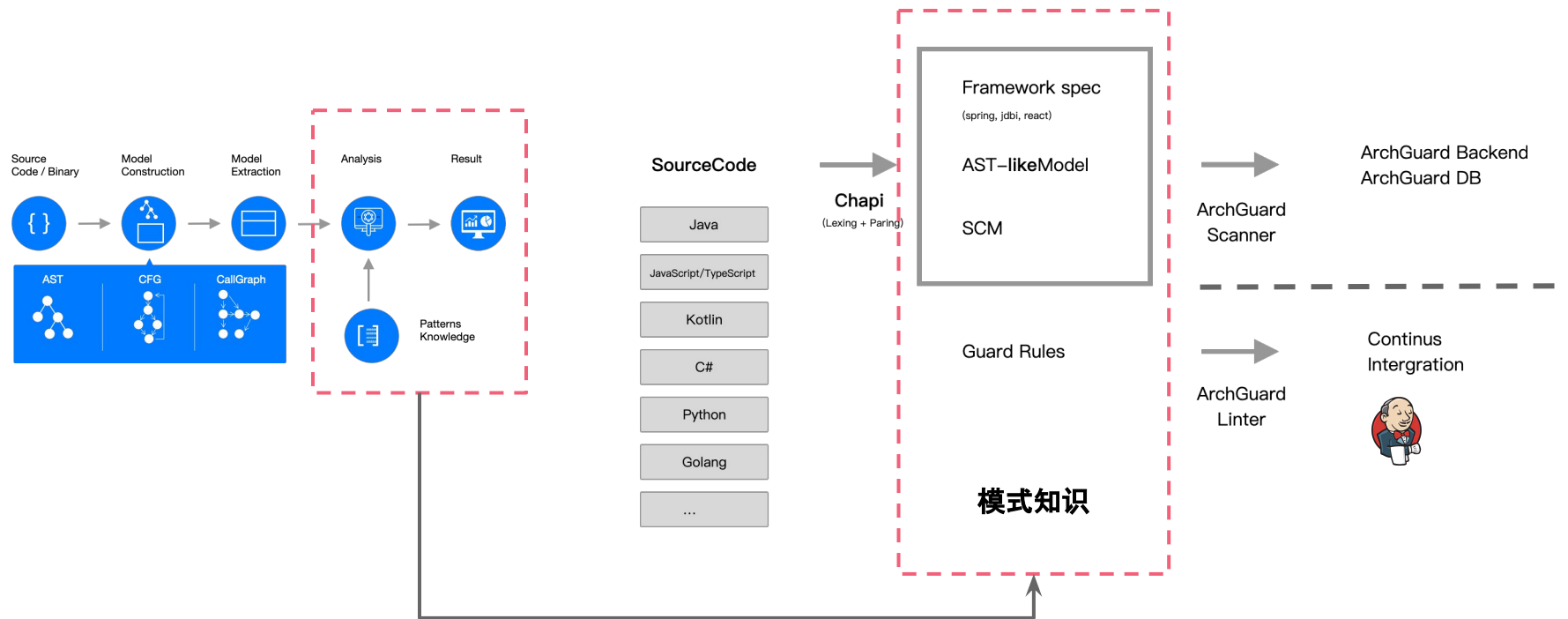
(基于规则与表达式持
续守护系统架构)

AGENDA

- 可扩展的 Scanner
 - 提供HTTP接口
 - Analyser插件化
- 后续计划
 - 架构模型 + 架构治理模型
 - 架构工作台

可扩展的 SCANNER

ArchGuard 数据流



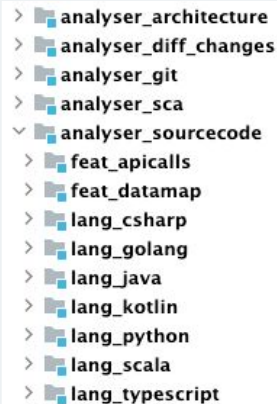
非必要不重构

- 旧的SourceCode Scanner包含生成语法树、扫描API/DB等多个功能, 打包出来有40M+
- 所有Scanner由ArchGuard后端直接调用, 增加、修改Scanner也需要修改后端代码
- Scanner通过直连后端数据库回写数据, 既不安全也不灵活
- 响应社区:
 - 集成到自身的架构管控平台
 - HTTP接口、RPC调用、结合链路追踪...

Scanner是什么

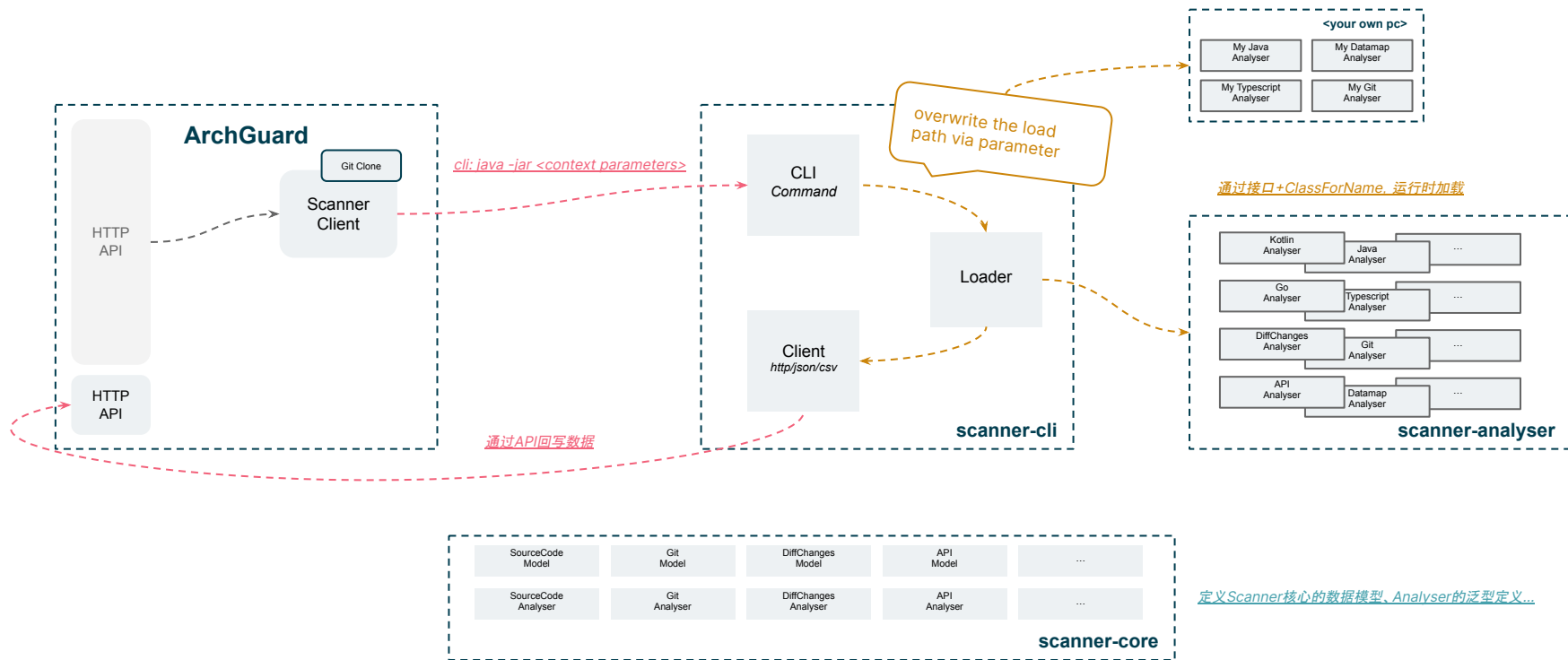
源代码分析？元数据抽取？数据处理？图计算？规则判断？

- **Scanner**: 作为ArchGuard的一个模块存在, 和Backend、Frontend同级
 - **Analyser**: 针对某一种输出进行分析, 输出标准的模型数据
 - SourceCode: 输入源代码 → 输出DataStructure数据(对象元数据)
 - API: 输入DataStructure → 输出API模型
 - Datamap: 输入DataStructure → 输出数据库地图
 - Git: 输入git仓库 → 输出git概述信息
 - Diff Changes: 输入git仓库 → 输出文件修改记录
 - SCA: 输入源代码 → 输出依赖关系
 - **Lint**: 针对某一种标准的模型数据, 输出提示信息或打断CI



```
> analyser_architecture
> analyser_diff_changes
> analyser_git
> analyser_sca
▼ analyser_sourcecode
  > feat_apicalls
  > feat_datamap
  > lang_csharp
  > lang_golang
  > lang_java
  > lang_kotlin
  > lang_python
  > lang_scala
  > lang_typescript
```

整体架构



提供HTTP接口

HTTP API

Client
http/json/csv

```
@RestController
@RequestMapping("/api/scanner/{systemId}/reporting")
class ScannerReportingController(...) {

    @PostMapping("/class-items")
    fun saveClassItems(@PathVariable systemId: String, @RequestParam language: String, @RequestParam path: String, @RequestBody input: List<CodeDataStruct>, ) {...}

    @PostMapping("/container-services")
    fun saveContainerServices(@PathVariable systemId: String, @RequestParam language: String, @RequestParam path: String, @RequestBody input: List<ContainerService>) {...}

    @PostMapping("/datamap-relations")
    fun saveRelations(@PathVariable systemId: String, @RequestParam language: String, @RequestParam path: String, @RequestBody input: List<CodeDatabaseRelation>) {...}

    @PostMapping("/git-logs")
    fun saveGitLogs(@PathVariable systemId: Long, @RequestBody input: GitLogs) {...}

    @PostMapping("/diff-changes")
    fun saveDiffs(@PathVariable systemId: Long, @RequestParam since: String, @RequestParam until: String, @RequestBody input: List<ChangedCall>) {...}

    @PostMapping("/sca-dependencies")
    fun saveDependencies(@PathVariable systemId: Long, @RequestBody input: List<CompositionDependency>) {...}
}
```

```
class ArchGuardHttpClient(...) : ArchGuardClient {

    override fun saveDataStructure(codes: List<CodeDataStruct>) {...}

    override fun saveApi(apis: List<ContainerService>) {...}

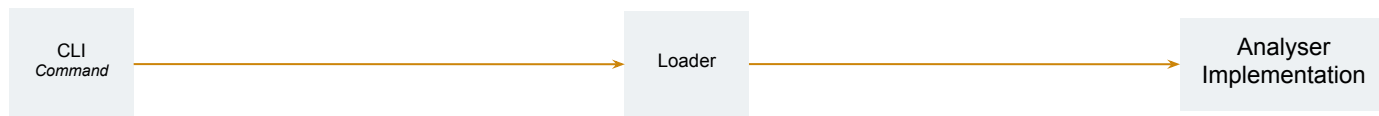
    override fun saveRelation(records: List<CodeDatabaseRelation>) {...}

    override fun saveGitLogs(gitLogs: GitLogs) {...}

    override fun saveDiffs(calls: List<ChangedCall>) {...}

    override fun saveDependencies(dependencies: List<CompositionDependency>) {...}
}
```

Analyser插件化



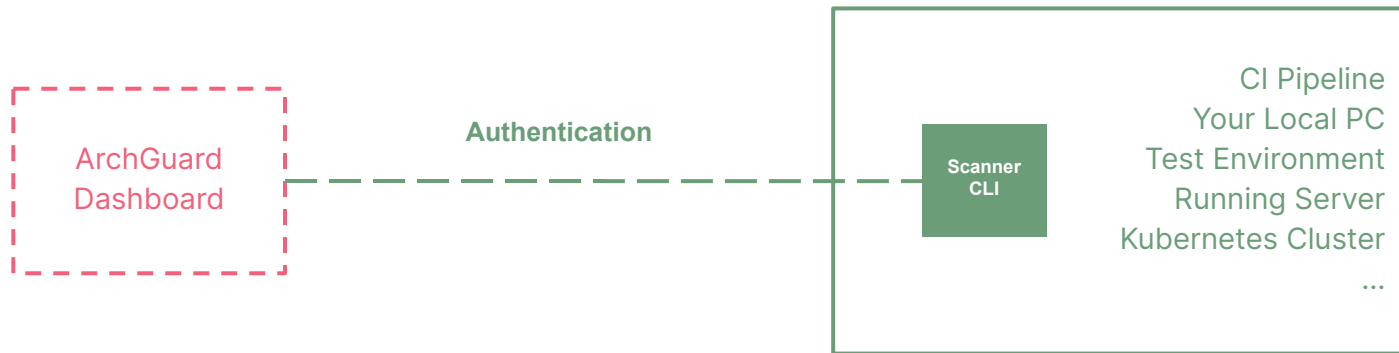
```
@Serializable
data class AnalyserSpec(
    val identifier: String,
    val host: String,
    val version: String,
    val jar: String,
    val className: String,
)
```

```
private val officialSpec = AnalyserSpec(
    identifier = "kotlin",
    host = "https://github.com/archguard/scanner/releases/download/v2.0.0",
    version = "2.0.0",
    jar = "lang_kotlin-2.0.0-all.jar",
    className = "KotlinAnalyser",
)

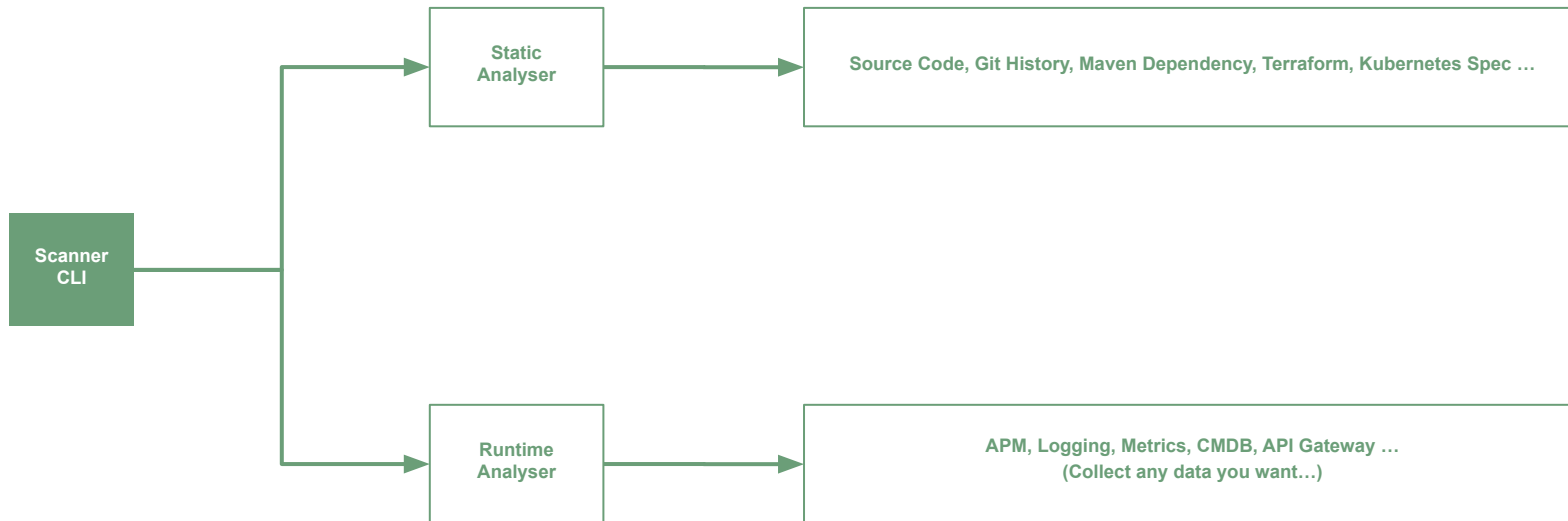
private val localSpec = AnalyserSpec(
    identifier = "kotlin",
    host = "/var/lib/scanner",
    version = "0.0.1",
    jar = "anddd7-lang_kotlin-0.0.1-all.jar",
    className = "MyKotlinAnalyser",
)

private val remoteSpec = AnalyserSpec(
    identifier = "kotlin",
    host = "https://github.com/anddd7/scanner/releases/download/v1.0.0",
    version = "1.0.0",
    jar = "lang_kotlin-1.0.0-all.jar",
    className = "MyV1KotlinAnalyser",
)
```

可提供的扩展能力



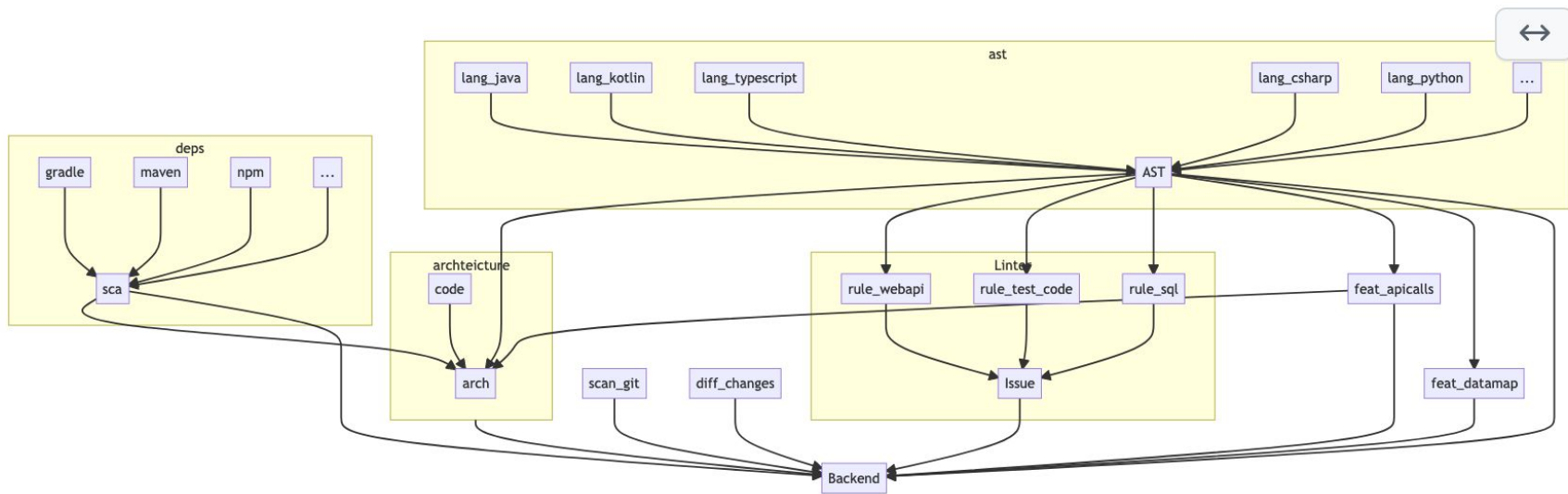
可提供的扩展能力



Next

Next: 汇聚数据, 处理数据

让数据聚合起来 => 架构模型、架构治理模型

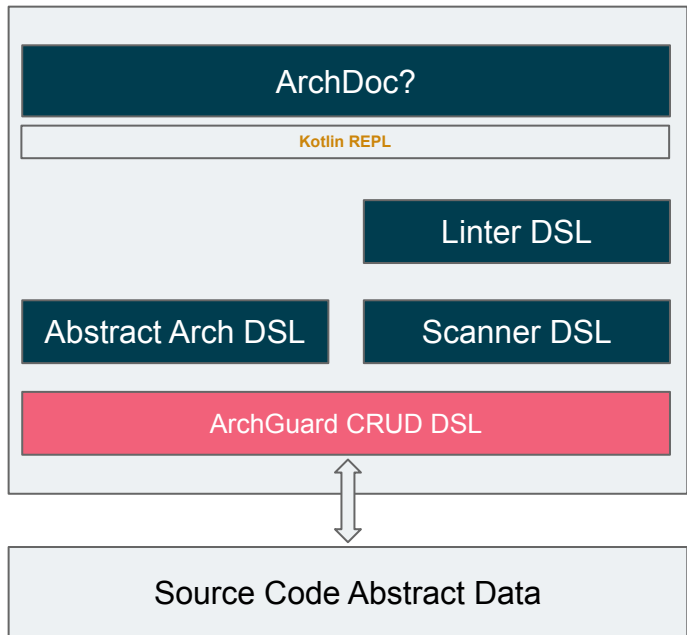


<https://github.com/archguard/archguard-wiki/blob/main/architecture/ArchitectureView.md>

615

下下一步:架构工作台(PoC)

Everything as DSL + 架构即代码的 ArchGuard 实现



ArchGuard

Welcome to join us~