CiVers
CITATION OF VERSIONED WEB PAGES BY PID

DEUTSCHES ARCHÄOLOGISCHES INSTITUT

@civers@social.bund.de

CiVers

Ein Projekt zur Entwicklung einer nachnutzbaren Informationsinfrastruktur-Software zur on-demand-Zitation von Fachinformationen als versionierte Webressourcen

CiVers-Team
Dr. Marcel Riedel, Manager für Content- und Data-Publishing
Fabian Riebschläger, Leiter Forschungsdatenmanagement
Giulia Russo, Projekt-Management
Ammar Saeed, Software-Entwicklung

Propylaeum FACHINFORMATIONSDIENST ALTERTUMSWISSENSCHAFTEN

GBV | VZG

DataCite FIND, ACCESS, AND REUSE DATA

DFG Deutsche Forschungsgemeinschaft

CiVers System Overview

Two-Part Architecture: Archive Generation + Web Interface

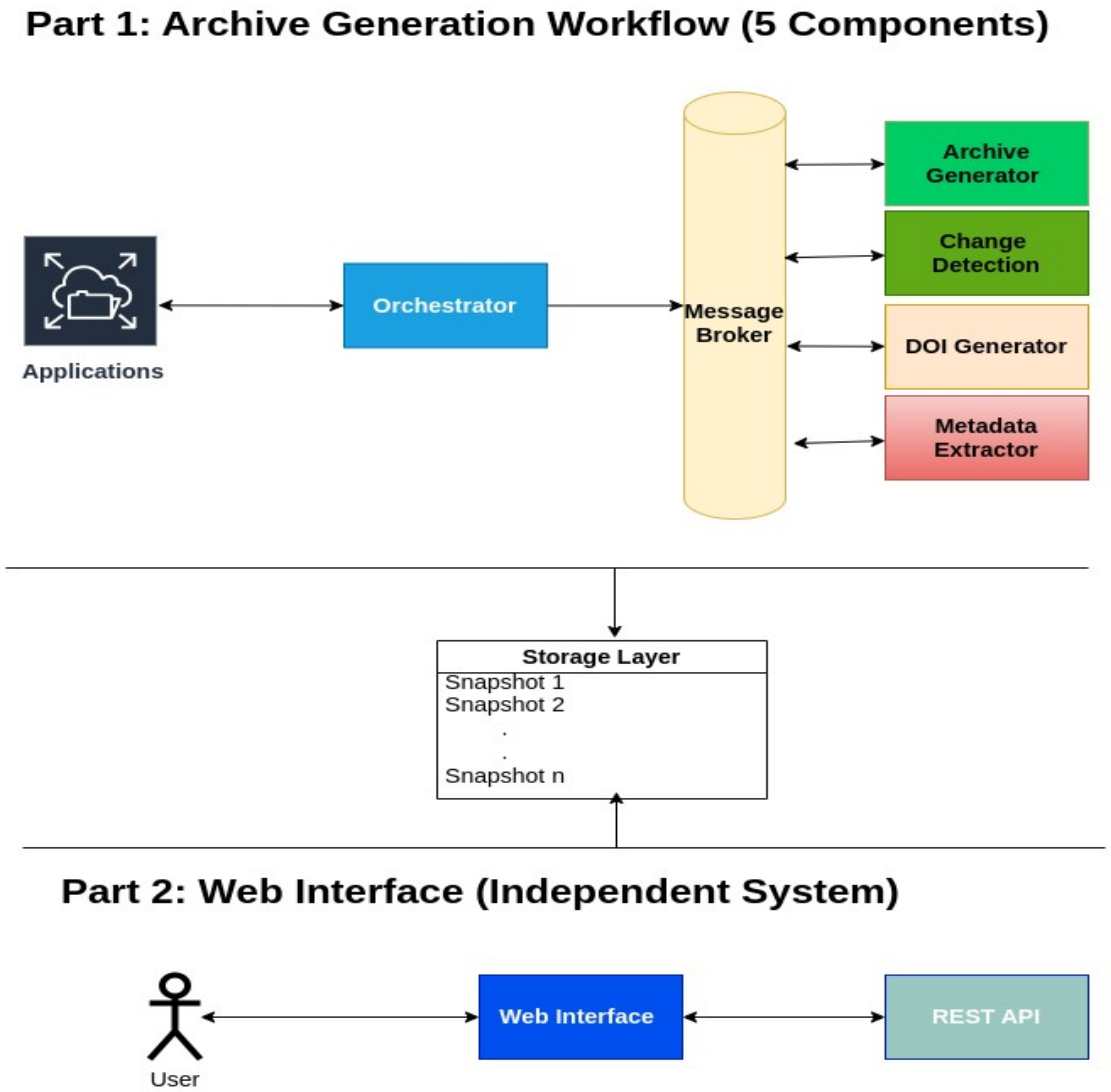
Part 1: Archive Generation Workflow (5 Components)

- **Orchestrator**: Central coordinator managing the entire workflow
- **Archive Generator:** Creates multiple format preservations (WARC,Singlefile, HTML, screenshots)
- **Metadata Extractor:** Extracts and converts to academic standards (DataCite)
- **Change Detection:** Monitors websites for content changes
- **DOI Generator:** Assigns persistent identifiers

Part 2: Web Interface (Independent System)

- **Standalone web application**: for accessing archived content
- **DOI Resolution**: Where DOI URLs resolve to archived content
- **REST API**: for querying the archives
- **REPLAY**: replay warc file
- **Download:** the artifacts like (warc, Singlefile, HTML, screenshots)

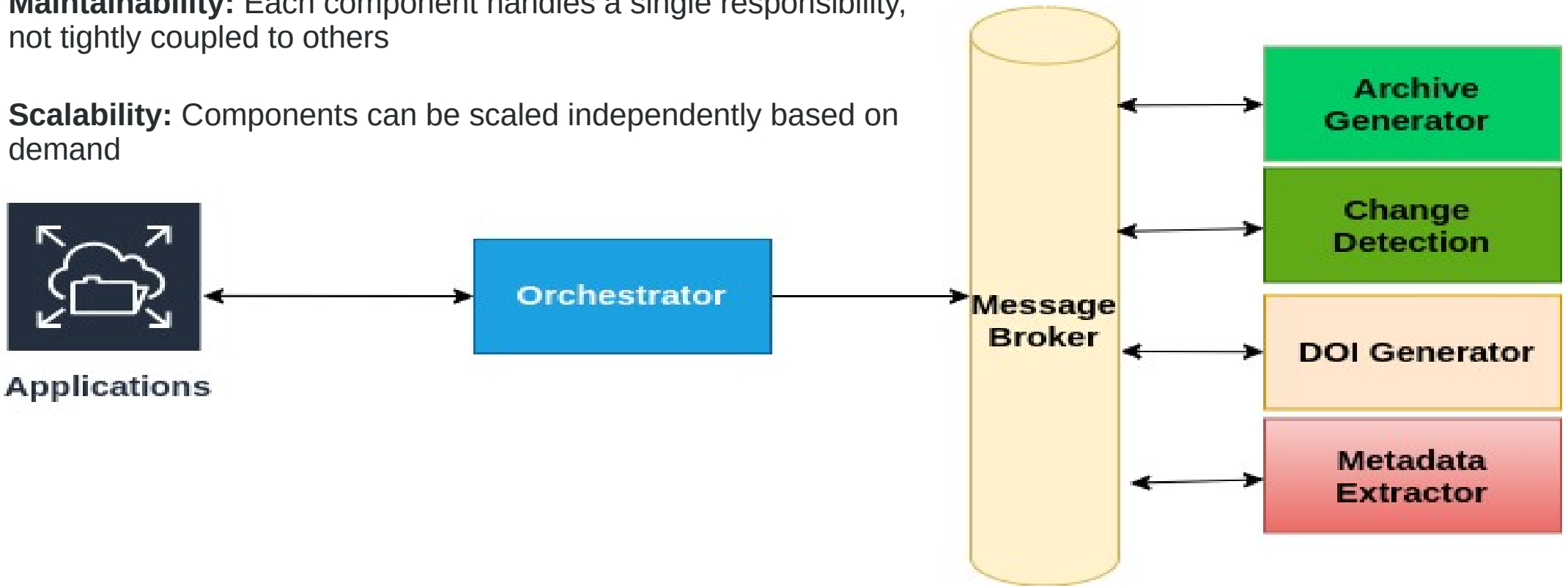
Key Design Principle:
Independence: Web Interface can operate without Archive Generation
Contingency: If institution stops hosting archive generation, web interface continues serving existing archives



Architecture Benefits: Maintainability and Scalability

Maintainability: Each component handles a single responsibility, not tightly coupled to others

Scalability: Components can be scaled independently based on demand

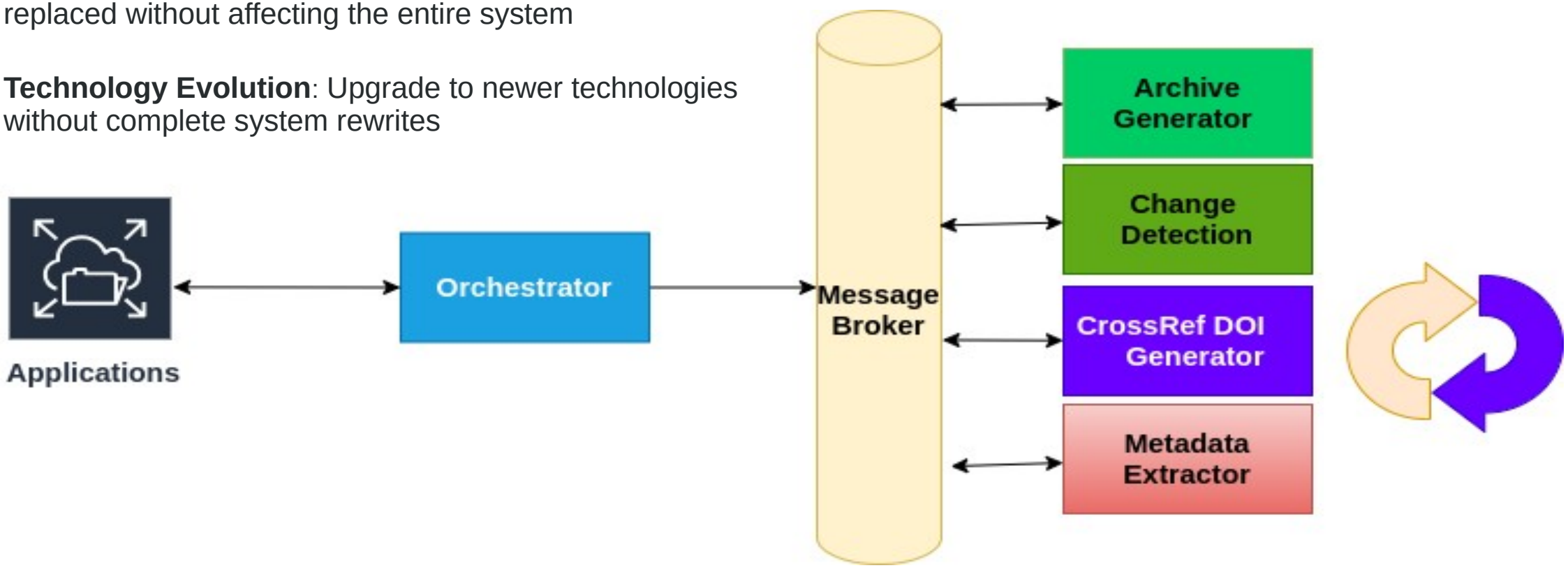


Example: Need more archiving capacity? Scale just the Archive Generator without touching other components

Architecture Benefits: Flexibility and Technology Evolution

Flexibility: Components can be changed, removed, or replaced without affecting the entire system

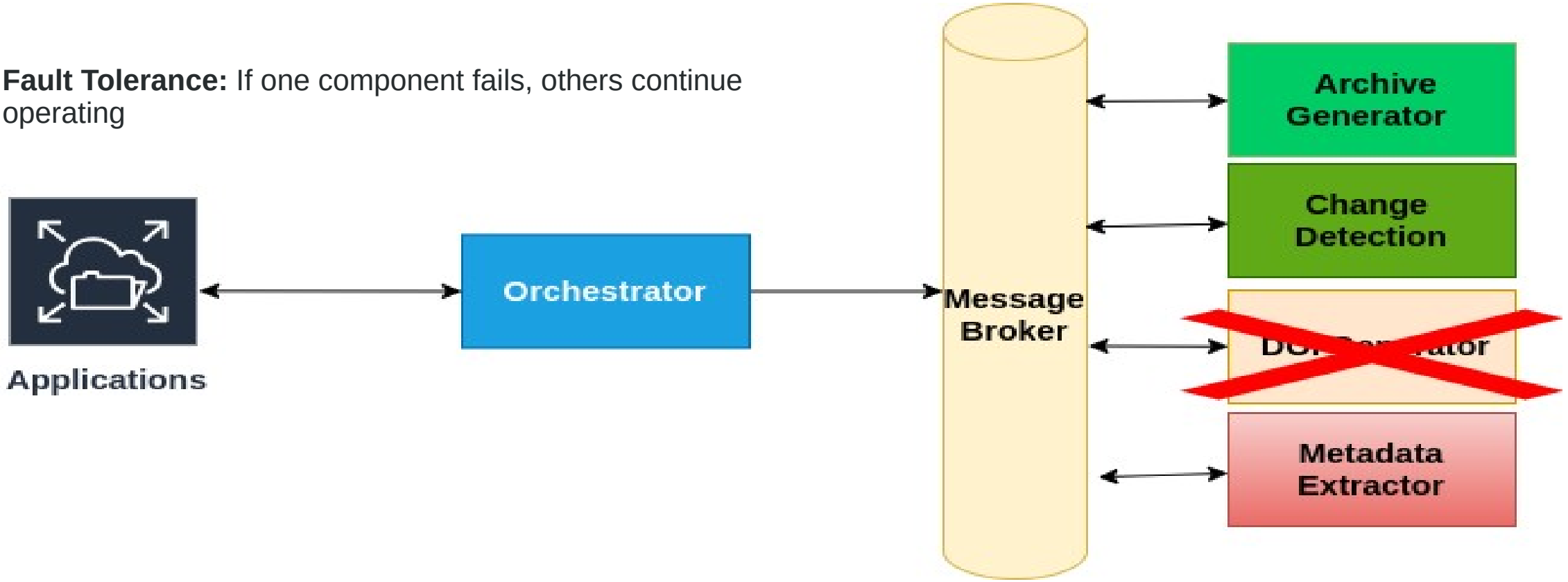
Technology Evolution: Upgrade to newer technologies without complete system rewrites



Example: Switching from our DOI Generator to CrossRef DOI Generator seamlessly

Architecture Benefits: Fault Tolerance

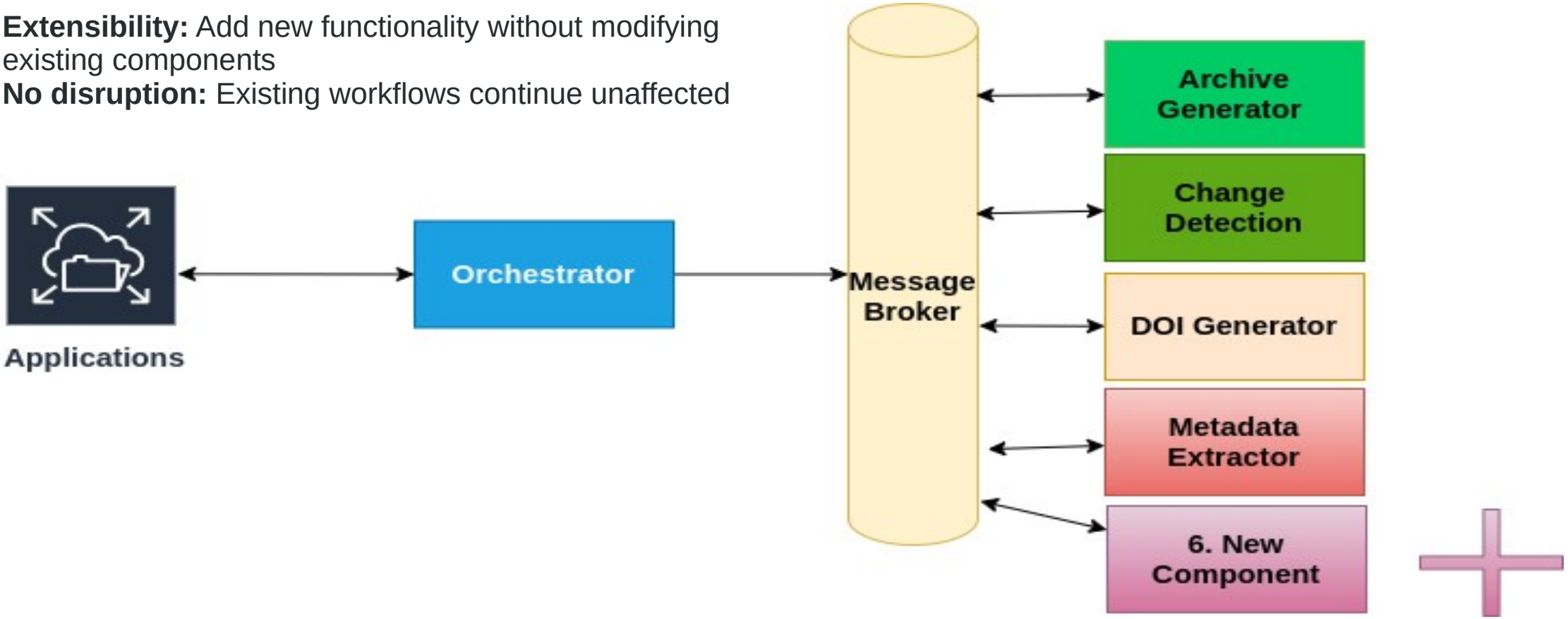
Fault Tolerance: If one component fails, others continue operating



Example: DOI Generator down? Archives, metadata, and change detection keep running

Architecture Benefits: Extensibility

Extensibility: Add new functionality without modifying existing components
No disruption: Existing workflows continue unaffected



Examples: Adding Quality checking, enhanced metadata enrichment, external repository integration

Current Implementation Status

Currently Operational (Ready for Use)

Archive Generator: Multi-format web content preservation

- WARC files (industry standard)
- SingleFile (self-contained HTML)
- HTML snapshots
- Screenshots (PNG format)

Metadata Extractor: Academic metadata processing

- JSON-LD extraction from web pages
- DataCite 4.6 format conversion
- Academic citation generation

Web Interface: Archive access and browsing:

- REST API for archive queries
- WARC file replay functionality
- Download access to all archive formats

Under Development (Coming Soon 2026)

Orchestrator: Workflow coordination and component management

- Purpose: Orchestrate the entire archive workflow

Change Detection: Automated website monitoring

- Purpose: Detect content changes and trigger re-archiving

DOI Generator: Official persistent identifier assignment

- Purpose: Generate official DOIs through DataCite API

Technology Stack

Core Languages

- **Python 3.12+**

Packages Used:

- **Harvard Scoop** – an external open-source project for web page capture.

Architecture & Infrastructure

- **Apache Kafka** – Event-driven messaging, scalable, reliable, industry standard
- **Docker** – Consistent deployment, easy maintenance, cloud/on-prem ready

Web Technologies

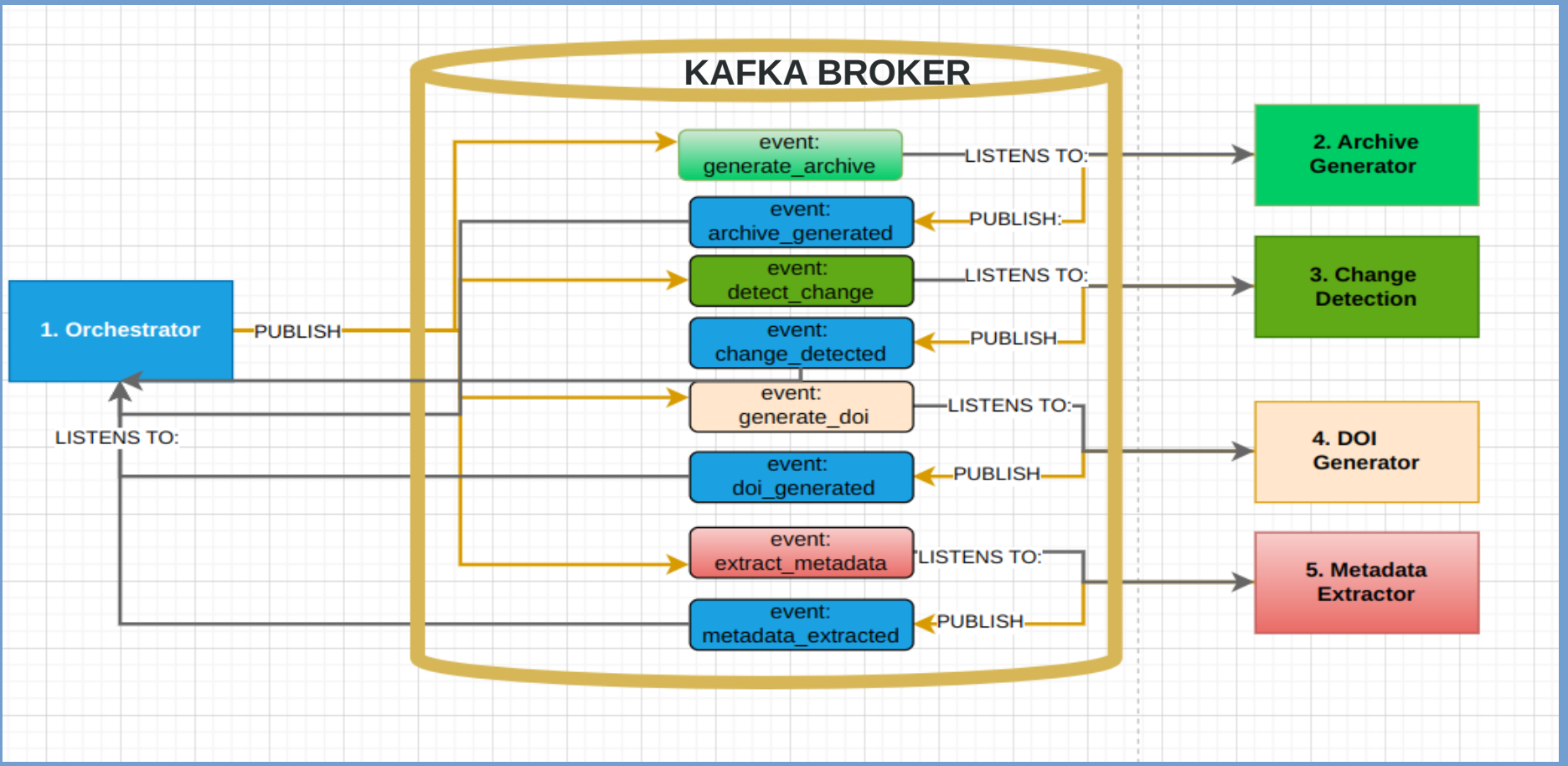
- **FastAPI** – Modern, high-performance Python web framework with auto API docs & validation
- **AlpineJs**– a lightweight JavaScript framework

Event-Driven Communication 1/2

Message Flow Pattern:

1. **Event Listening**: Each component subscribes to specific event topics in the message broker
2. **Action Triggering:** When a relevant event arrives, the component automatically starts its designated task
3. **Task Completion:** Upon finishing, the component publishes a completion event with results
4. **Workflow Coordination:** The Orchestrator manages the overall workflow by triggering initial events and monitoring completion events

Event-Driven Communication 2/2



LIVE DEMO!