Kickstart your **AsyncAPI** journey with **Springwolf**

Timon Back
Maintainer @ Springwolf

Situations in the enterprise

What is the problem?

No documentation

- No docs
- Informal documentation
- “API does not work”

Implementation drift

- Refactor breaks API
- Testing schema by hand
- “API does not work”

Broken environment

- Accidental poison pill
- Non-monitored DLQs

No discoverability

- No central hub nor single point of contact
- Missing API overview & governance

Situations in the enterprise

Why is it a problem?

No documentation

- High communication
- Frustration
- Details are easily missed

Implementation drift

- Higher rate of defects
- Less confidence
- “I do not touch that code”

Broken environment

- Has testing/user impact
- Longer cycle time
- Requires fixing (invisible work)

No discoverability

- High communication
- Who is impacted by API change?
- No API AI readiness

Executive Summary

Springwolf automates event-driven documentation and lets teams focus on building the best APIs

Instrument application

- Add Springwolf
- Optional: Add springwolf-ui

Verify API changes

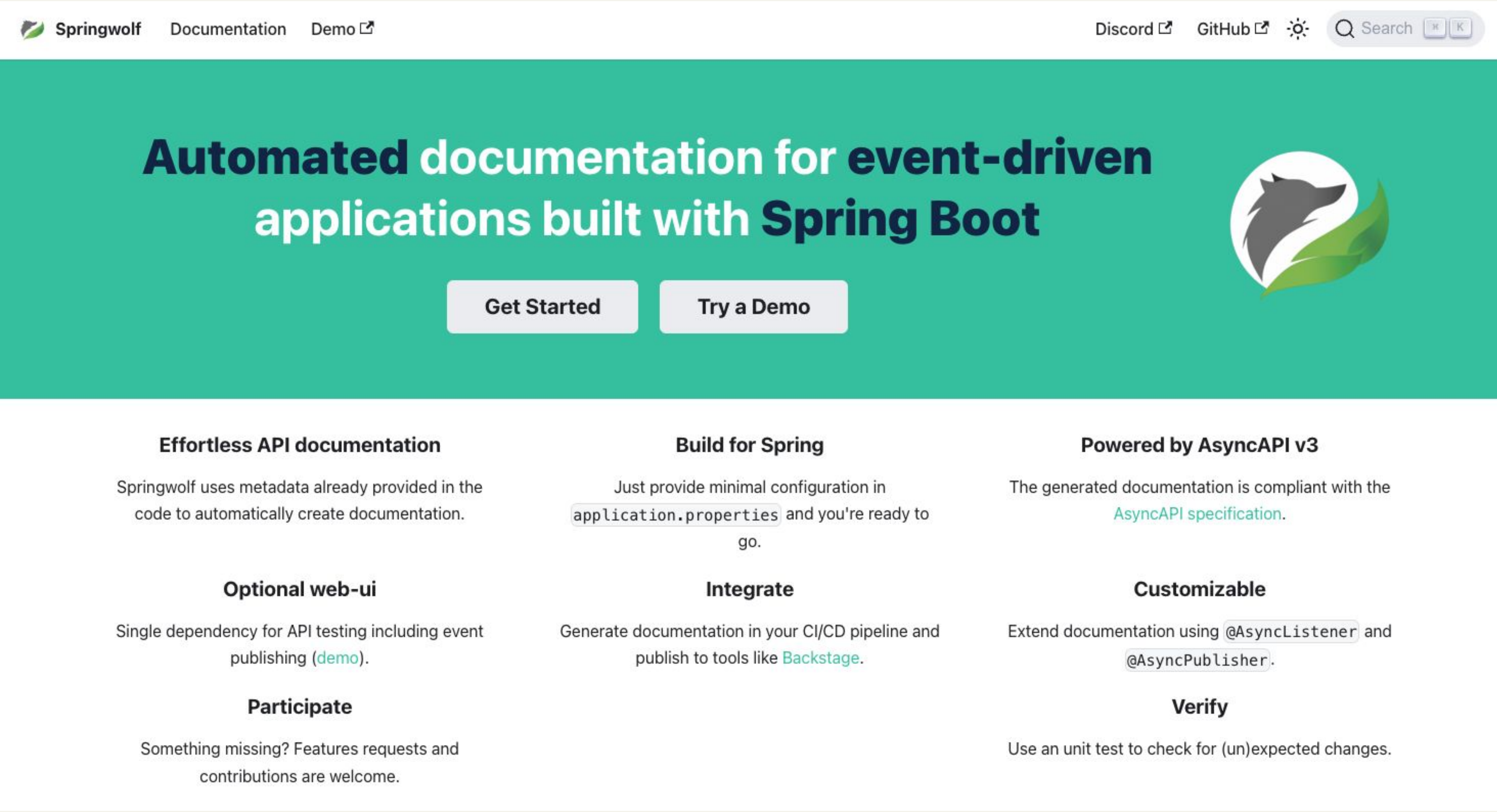
- Persist asyncapi.yaml to repo
- Verify using test

Benefits: team-level

Use API Hub

- Integrate in API Hubs
  (i.e. Backstage, Eventcatalog)
- Discover APIs of other teams

Benefits: team & org-level



Automated documentation for event-driven applications built with Spring Boot

Effortless API documentation

Springwolf uses metadata already provided in the code to automatically create documentation.

Build for Spring

Just provide minimal configuration in `application.properties` and you're ready to go.

Powered by AsyncAPI v3

The generated documentation is compliant with the AsyncAPI specification.

Optional web-ui

Single dependency for API testing including event publishing (demo).

Integrate

Generate documentation in your CI/CD pipeline and publish to tools like Backstage.

Customizable

Extend documentation using `@AsyncListener` and `@AsyncPublisher`.

Participate

Something missing? Features requests and contributions are welcome.

Verify

Use an unit test to check for (un)expected changes.

Spec or Code First?

It probably depends on your workflows / culture

Spec First

1. Committee plans API contract upfront
2. AsyncAPI as artifact
3. Use code-generator to generate schema classes & methods/interfaces

Issues

- Requires detailed, upfront planning
- Inflexible for API changes
- Sometimes, better API contract is discovered during implementation
- Code generator may be limited in features

Committee + Code Generator

Code First

1. Developers implement API based on requirements
2. AsyncAPI as output artifact

Issues

- Review of API best practices lays in the team
- Implementation tends to be sequential (one first, afterwards other side)
- More flexibility and multiple styles for the same thing

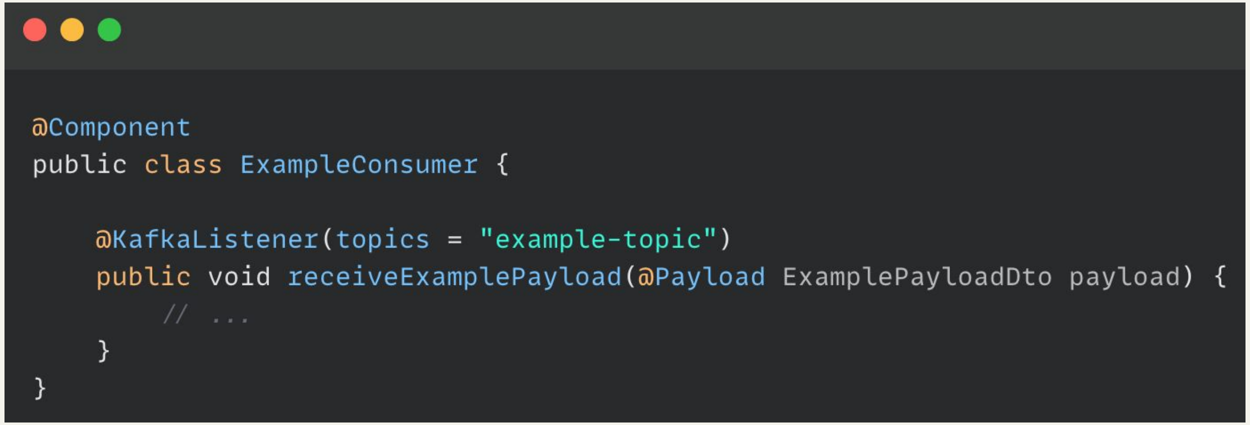
Team ownership + **Springwolf**

Automatic AsyncAPI artifact generation by analyzing source code

Basics: Spring Boot

JVM (Java, Kotlin, ...) is widely popular for developing applications
Within the JVM ecosystem, Spring Boot is the most used web framework

Developers configure applications using *annotations*, for example Kafka:

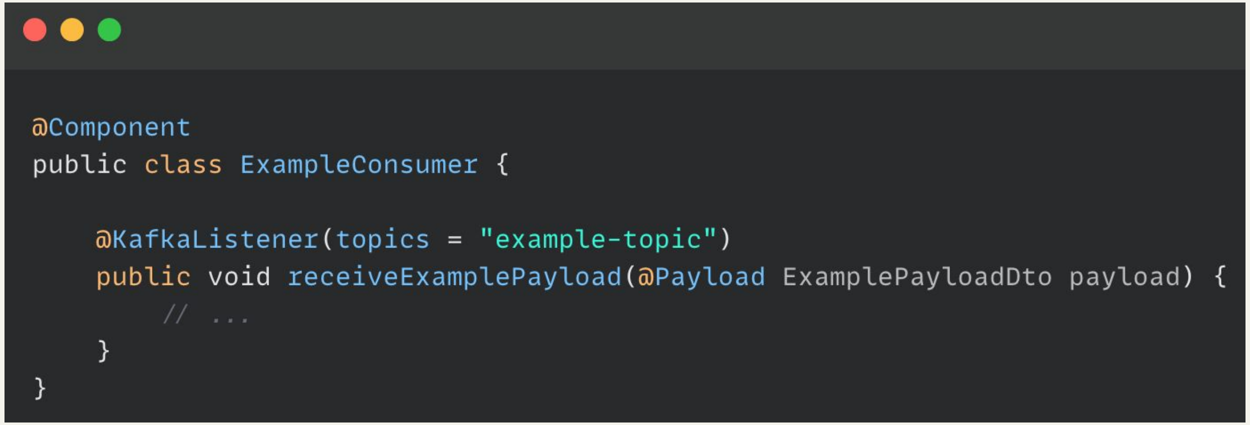


```
@Component
public class ExampleConsumer {

    @KafkaListener(topics = "example-topic")
    public void receiveExamplePayload(@Payload ExamplePayloadDto payload) {
        // ...
    }
}
```

It’s hidden in the code

Springwolf analyses annotations and methods

Springwolf builds upon these annotations to create the documentation automatically



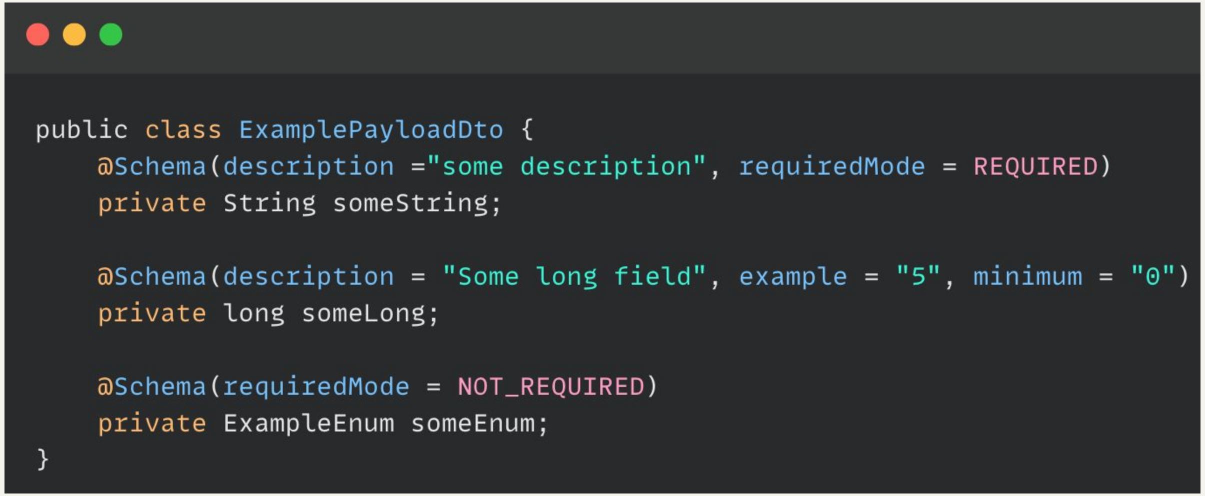
```java
@Component
public class ExampleConsumer {

    @KafkaListener(topics = "example-topic")
    public void receiveExamplePayload(@Payload ExamplePayloadDto payload) {
        // ...
    }
}
```

It’s hidden in the code

Springwolf analyses annotations and methods and classes

OpenAPI Swagger @Schema annotation is re-used (although not required)

Documentation should have high locality with the actual code



```java
public class ExamplePayloadDto {
    @Schema(description ="some description", requiredMode = REQUIRED)
    private String someString;

    @Schema(description = "Some long field", example = "5", minimum = "0")
    private long someLong;

    @Schema(requiredMode = NOT_REQUIRED)
    private ExampleEnum someEnum;
}
```

Demo time

https://github.com/timonback/springwolf-demo

Follow step by step using the commits in the repo

1. Spring Boot Initializr Setup
2. Add Kafka setup
3. Add Springwolf
4. Add Springwolf-ui, incl. message publishing (optional)
5. Verify using test
6. Document Kafka producer

Using Springwolf is easy

https://www.springwolf.dev/docs/quickstart

1. Add dependency

```
implementation 'io.github.springwolf:springwolf-kafka:1.17.0'
runtimeOnly    'io.github.springwolf:springwolf-ui:1.17.0' // optional
```

2. Add mandatory configuration to application.properties

```
springwolf.docket.base-package=io.github.springwolf.example.consumers

springwolf.docket.info.title=${spring.application.name}
springwolf.docket.info.version=1.0.0

springwolf.docket.servers.kafka-server.protocol=kafka
springwolf.docket.servers.kafka-server.host=${kafka.bootstrap.servers}
```

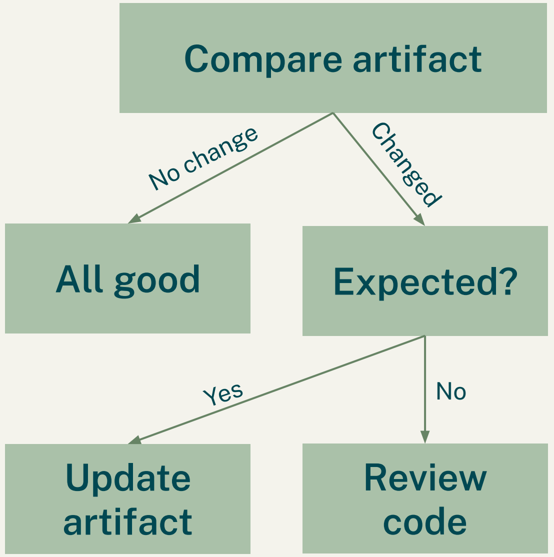
3. Open http://localhost:8080/springwolf/docs.yaml

Continuously verify API contracts and protect against unexpected changes

AsyncAPI as build artifact

Generated AsyncAPI is derived from the application

Want to detect changes?
→ Compare against earlier artifacts



Advice:

1. **Commit artifact into repository once**
2. **Automatically verify using test**

Contract tests

Verification across teams

Artifact is identical to code, therefore fulfils own publishing contract

Subscribers can be verified using tools in the ecosystem (i.e. microcks)

Technically, comparison of the AsyncAPI spec is sufficient - no runtime or example payloads necessary

Seamlessly integrate into software catalogs

API Hub Integration

Advantages

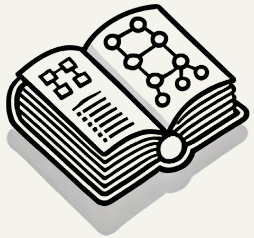
- Central place
- Discoverability through search
- **Many integrations including AsyncAPI using the asyncapi.yaml artifact**
- Documentation can live in team repos and is fetched
- Service Graphs
- Visualizes dependencies
- Supports governance

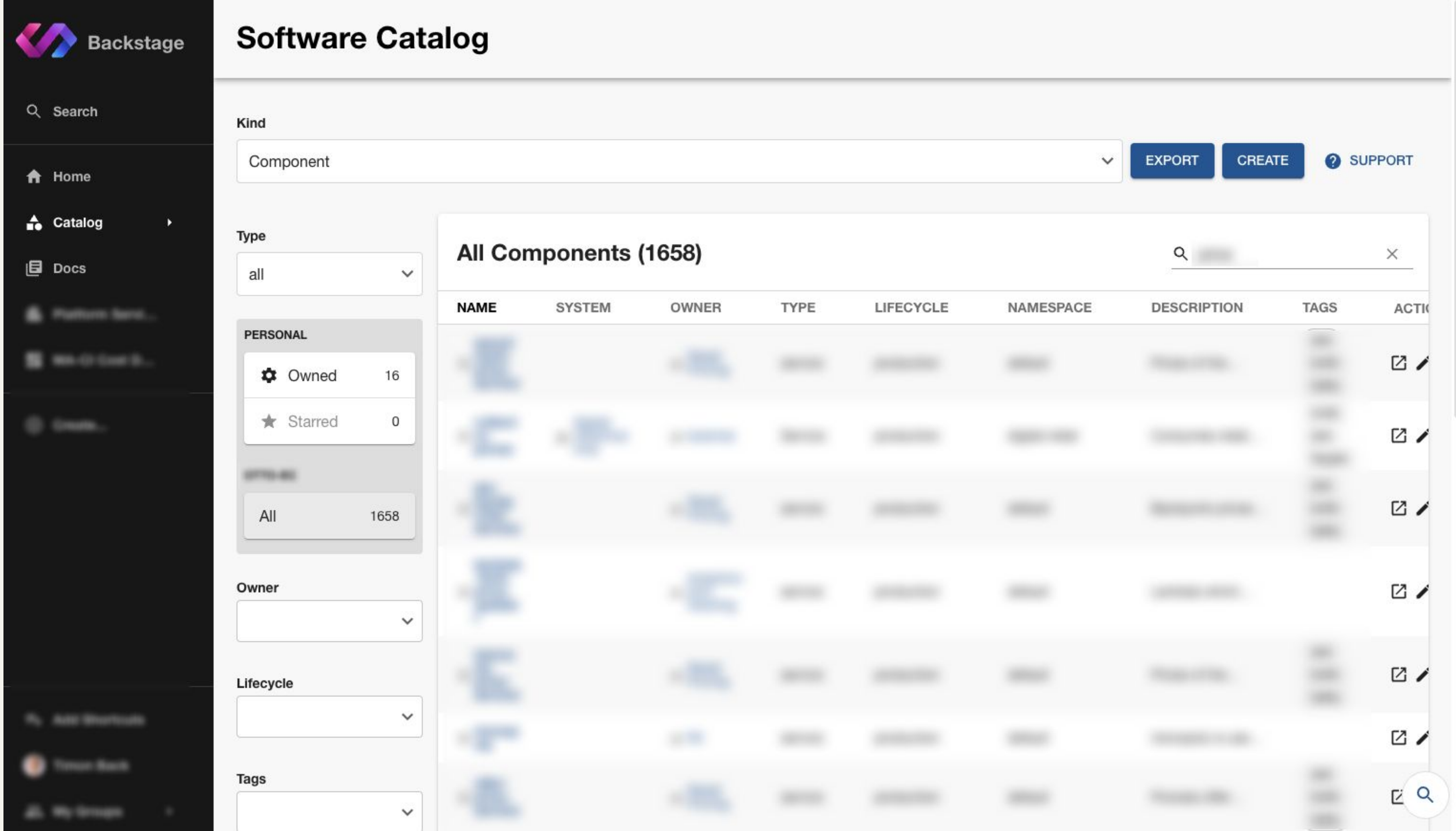
Backstage (example)

Backstage is an open source framework and restores order to your microservices and infrastructure — without compromising autonomy.

EventCatalog (example)

A single source of truth for your architecture, schemas, and ownership — so your team can ship faster and safer.





Backstage

Search

Home

Catalog

Docs

Software Catalog

Kind

Component

EXPORT

CREATE

SUPPORT

Type

all

PERSONAL

Owned	16
Starred	0
All	1658

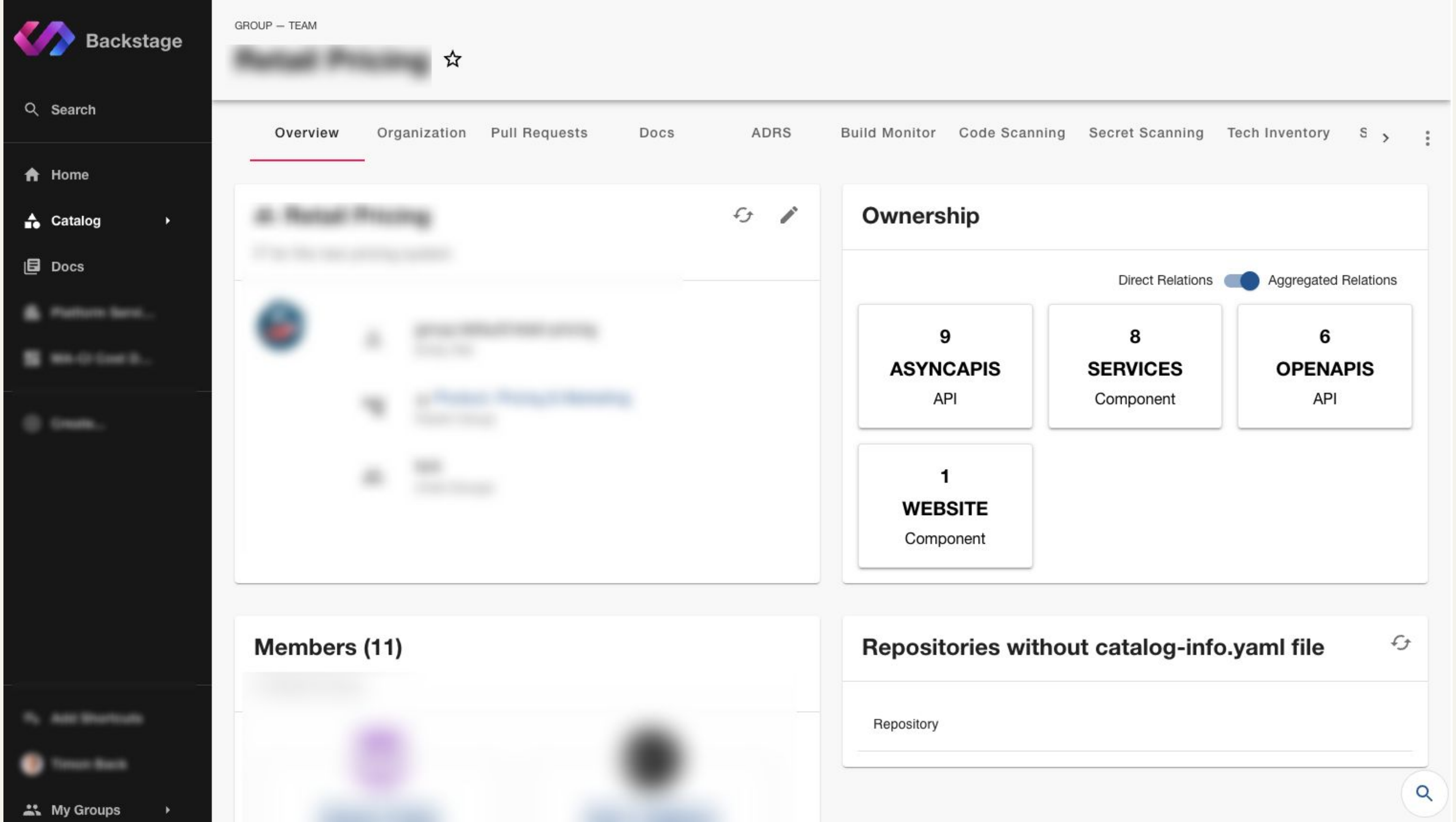
Owner

Lifecycle

Tags

All Components (1658)

NAME	SYSTEM	OWNER	TYPE	LIFECYCLE	NAMESPACE	DESCRIPTION	TAGS	ACTI
[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	



Ownership

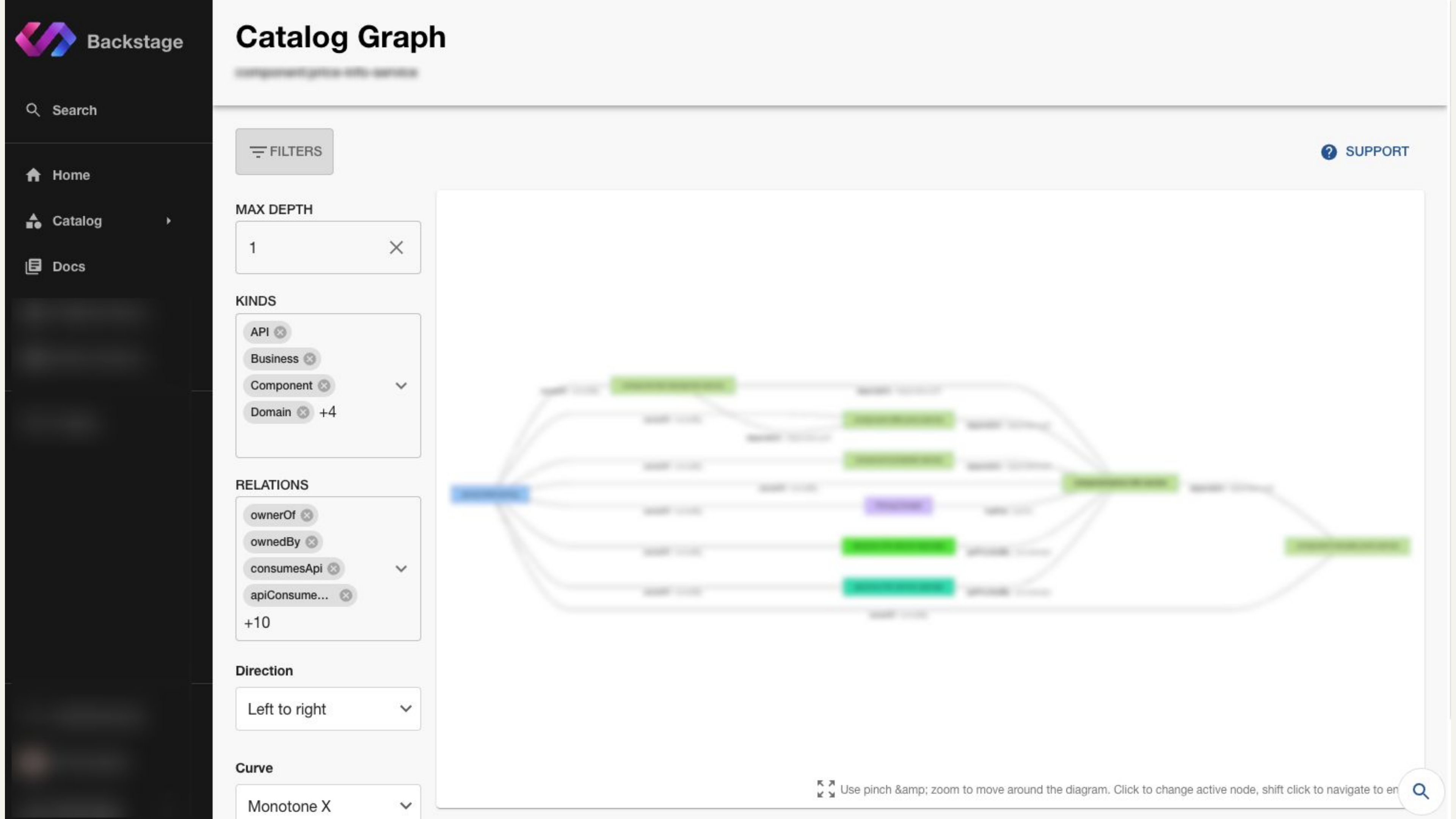
Direct Relations / Aggregated Relations

9 ASYNCAPIS API	8 SERVICES Component	6 OPENAPIS API
1 WEBSITE Component		

Members (11)

Repositories without catalog-info.yaml file

Repository



Catalog Graph

MAX DEPTH: 1

KINDS: API, Business, Component, Domain +4

RELATIONS: ownerOf, ownedBy, consumesApi, apiConsume… +10

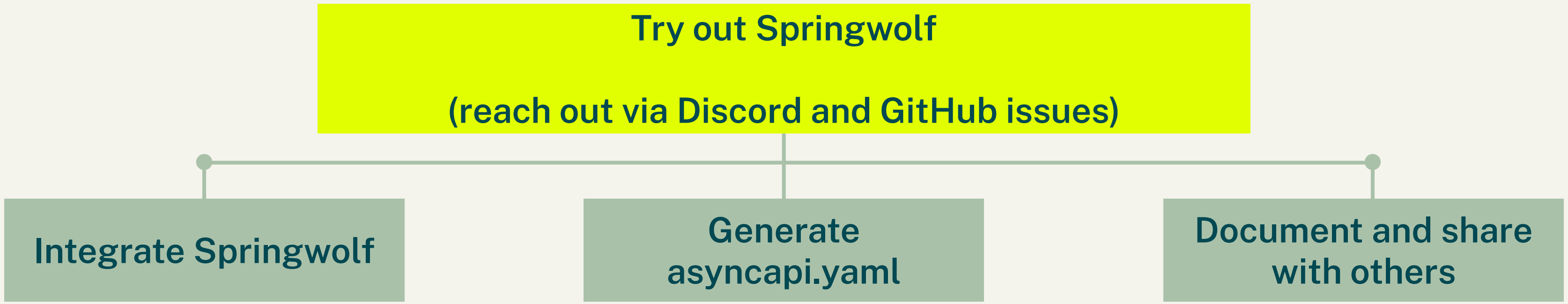
Direction: Left to right

Curve: Monotone X

Use pinch &amp; zoom to move around the diagram. Click to change active node, shift click to navigate to en

Conclusion

Kickstart your AsyncAPI journey


Try out Springwolf
(reach out via Discord and GitHub issues)
Integrate Springwolf
Generate asyncapi.yaml
Document and share with others

Supports RabbitMQ, Kafka, AWS SNS/SQS, JMS, WebSocket out-of-the-box
More via custom annotations

Go to https://www.springwolf.dev/docs/quickstart

Questions?

Demo at:
https://demo.springwolf.dev

Start here:
https://www.springwolf.dev