

# Cloud Service Strategies

**Prof. Dr. Dirk Riehle**

**Friedrich-Alexander University Erlangen-Nürnberg**

**COSS C04**

Licensed under CC BY 4.0 International

# Agenda

1. Feature differentiation
2. The open core model
3. Licensing strategies
4. Cloud computing challenges
5. Use of control mechanisms
6. Labor economics
7. Single-vendor life-cycle

# 1. Feature Differentiation

## Structure product and services so that you

1. Maximize conversion to paying customer
2. While benefiting from user community
3. And keeping the competition at bay

# Which Feature [1] Goes Where? (Recap)

- **Community edition**

- Core product
  - Core software
    - **Provided under an open source license**
  - Some complementary artifacts
  - Self-help services

- **Commercial edition**

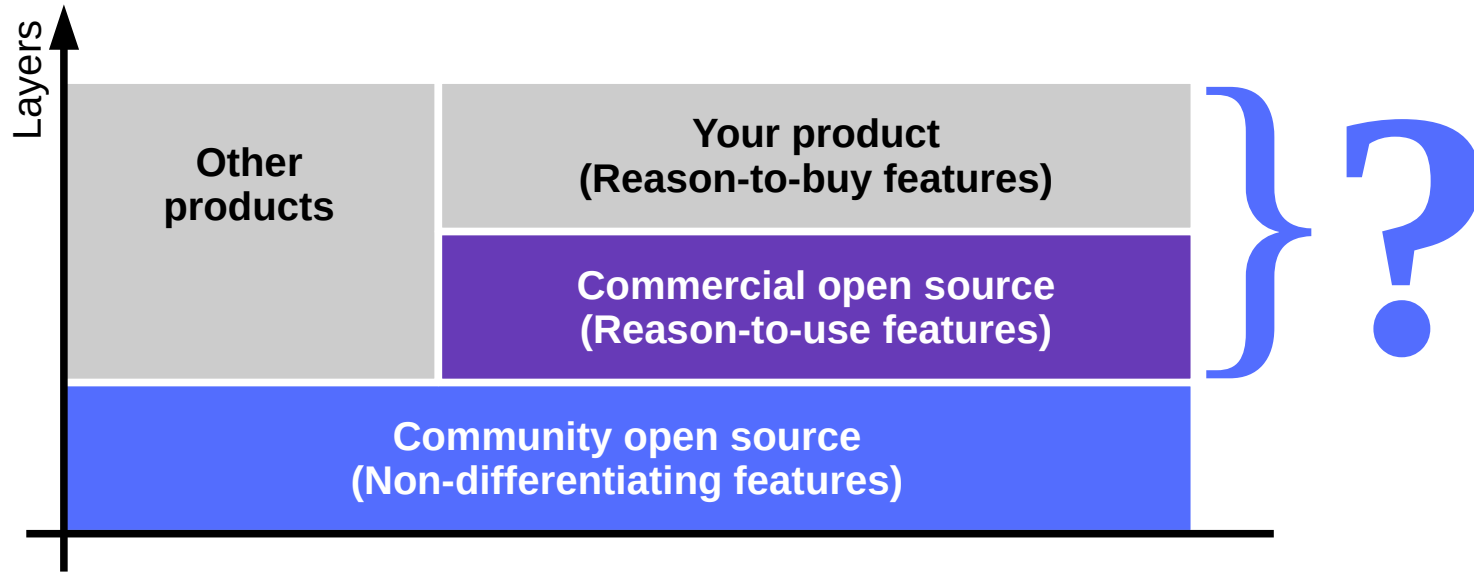
- **Core product**
  - **Core software**
    - **Provided under a commercial license**
  - **Additional functionality**
  - **Complementary artifacts**
  - **Self-help services**
- **Basic product =**
  - **Core product +**
  - **Fitness for use**
  - **Certification**
  - **Support services**
- **Whole product =**
  - **Basic product +**
  - **Training**
  - **Consulting**
  - **Operations**

[1] A (software) feature is a distinguishing characteristic of a software item (for example, performance, portability, or functionality) [IEEE 829]

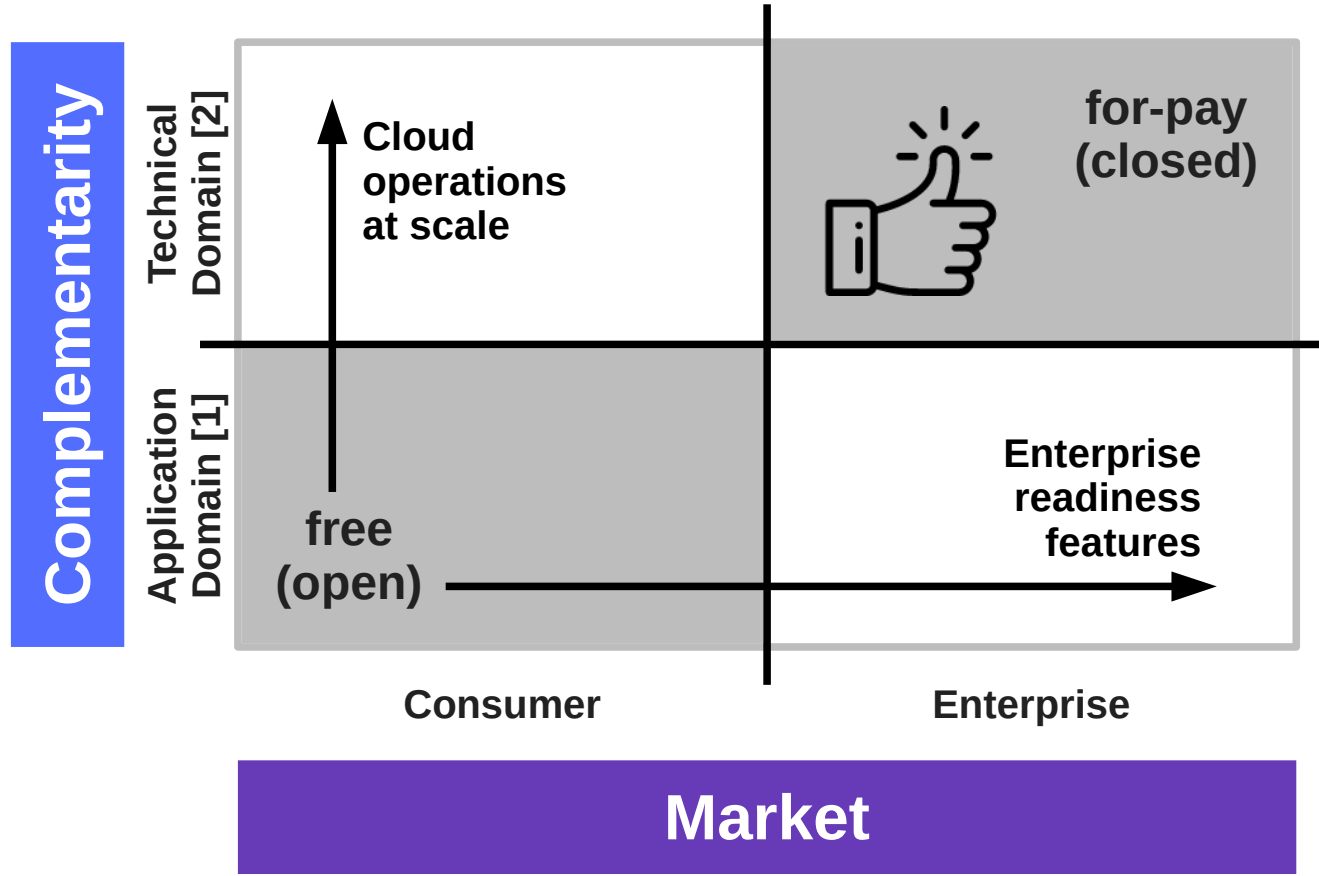
# A Commercial Open Source Feature Classification

- Non-differentiating
  - The feature is competitively not differentiating and readily available elsewhere
- Reason-to-use (value creation)
  - Users come to your software, because the feature is not ubiquitous
- Reason-to-buy (value appropriation)
  - Users upgrade to paying customers to receive this feature

# Open / Closed Software Feature Differentiation



# How to Think About Feature Differentiation

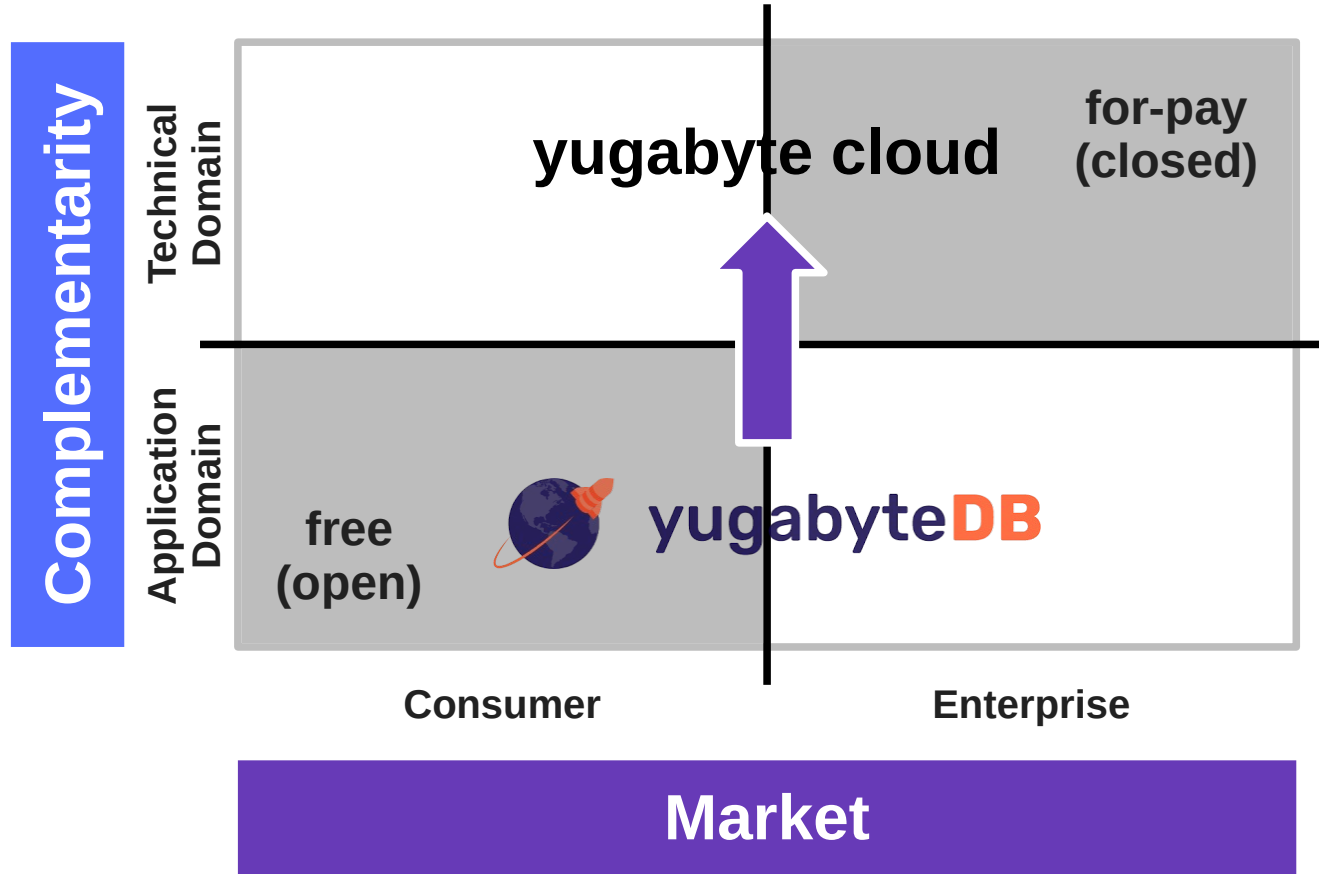


[1] Application domain = **business purpose** of software = functional requirements

[2] Technical domain = support infrastructure = non-functional requirements like costs of operations



# Yugabyte 2020 (Example Feature Differentiation)



## 2. The Open Core Model

# Intellectual Property Modularity

- Intellectual property (IP) modularity
  - The practice of splitting IP into modules of different licenses

Software  
component 1  
(license 1)

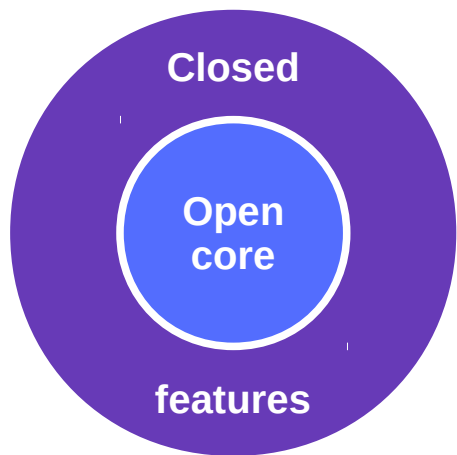
Software  
component 2  
(license 2)

# Open / Closed IP Modularity Examples

	open license	closed license	
<b>Software</b>	BI Report Editor	BI Report Generator	<ul style="list-style-type: none"><li>• Copyright</li><li>• Patents</li></ul>
<b>Hardware</b>	Tensorflow (Software)	Tensorflow Hardware	<ul style="list-style-type: none"><li>• Patents</li><li>• Trade secrets</li></ul>
<b>Service</b>	Database Software	Cloud Ops Software	<ul style="list-style-type: none"><li>• Copyright</li><li>• Patents</li></ul>

# Intellectual Property Modularity

- Open core model
  - A particular form of IP modularity where there is
    - An “open” software core available under an open source license and
    - Software extensions of the core are available only under a commercial license



# What do Open Core Users Worry About?

- That the vendor
  - Withholds critical features
  - Stops updating old features
  - Delays delivery of new features
  - Lacks backwards compatibility

# The Commercial Open Source Pledge

- Design choices for a pledge to create trust
  - Always open source or not
  - Permissive or copyleft
  - Allow competition or not
  - Copyright transfer or not
  - Support community or not
  - Influence roadmap or not

[1] See <https://dirkriehle.com/2019/06/11/the-commercial-open-source-pledge/>

# The Best Complement

- The best complement
  - Has high value to users
  - Can obviously not be free
- Hardware and cloud are today's best complements



### 3. Licensing Strategies

## Structure product and services so that you

1. Maximize conversion to paying customer
2. While benefiting from user community
3. And keeping the competition at bay

# Commercial Forks of Commercial Open Source Software

Compiere®



Nagios®



# IP Rights Management (Recap)

- Intellectual property rights imperative (of single-vendor open source)
  - “Always act in such a way that you, and only you, possess the right to provide the open source project under a license of your choice.” [1]
- Use contributor agreement to maintain ownership
  - Almost all single-vendor open source firms require copyright transfer for any contributions to maintain full IP ownership [2]

[1] Riehle, D. (2009). [The Intellectual Property Rights Imperative](#).

[2] All you really need is a relicensing right though

# Dual / Multi-Licensing

- Dual licensing / multi-licensing
  - The practice of licensing a piece of software under two or more licenses
- Commercial open source licensing
  - At least one open source and one commercial license
- Just which open source license(s)?
  - Basic idea: Use copyleft to keep competitors away

# Product Types

- Application
  - A piece of software that **can be used as is** for a business purpose
  - Example applications
    - A financial accounting system
    - A compiler
  - Applications dominated the second wave of commercial open source
- Component
  - A piece of software that **needs to be integrated** with other components
  - Example components
    - Functional library
    - Database system
  - Components dominate the current third wave of commercial open source

# Resulting Cloud Licensing Structures (Until About 2018)

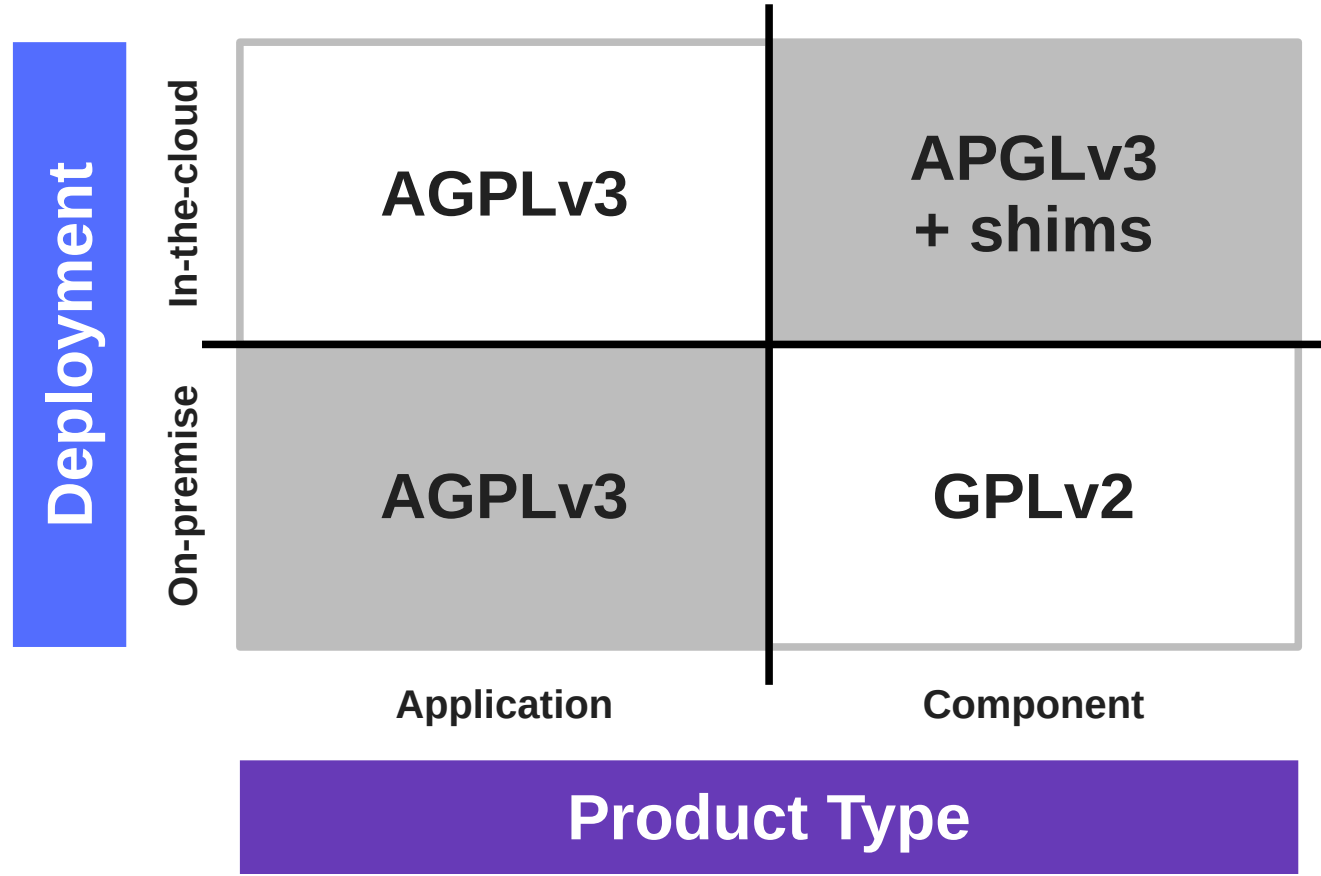
## Application



## Component



# Choice of Open Source License in Commercial Open Source





## 4. Cloud Computing Challenges

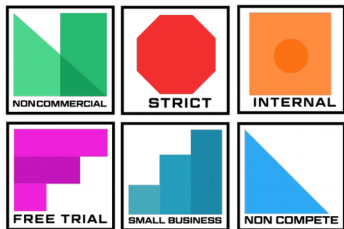
# The Move Into the Cloud

- A tectonic shift
  - (Almost) everything is moving into the cloud
- Open source
  - Becomes an on-ramp to the cloud
- Conversion is
  - From self-hosted to vendor-hosted
- The hyperscalers
  - Are the new competition



# The New “Source Available” License Category

- A new license category
  - “Like open source, but if you want to compete with us, you are not allowed to”
- Examples of new licenses
  - Server Side Public License (SSPL), introduced in 2018 by MongoDB
  - Business Source License (BSL), introduced in 2016 by MariaDB
  - Redis Source Available License (RSAL), introduced in 2019 by Redis Labs
- The Polyform Project [1]



## What is PolyForm?

PolyForm is a project to draft and make freely available plain-language source code licenses with limited rights.




[1] See <https://polyformproject.org/>

# The 2018 MongoDB License Change

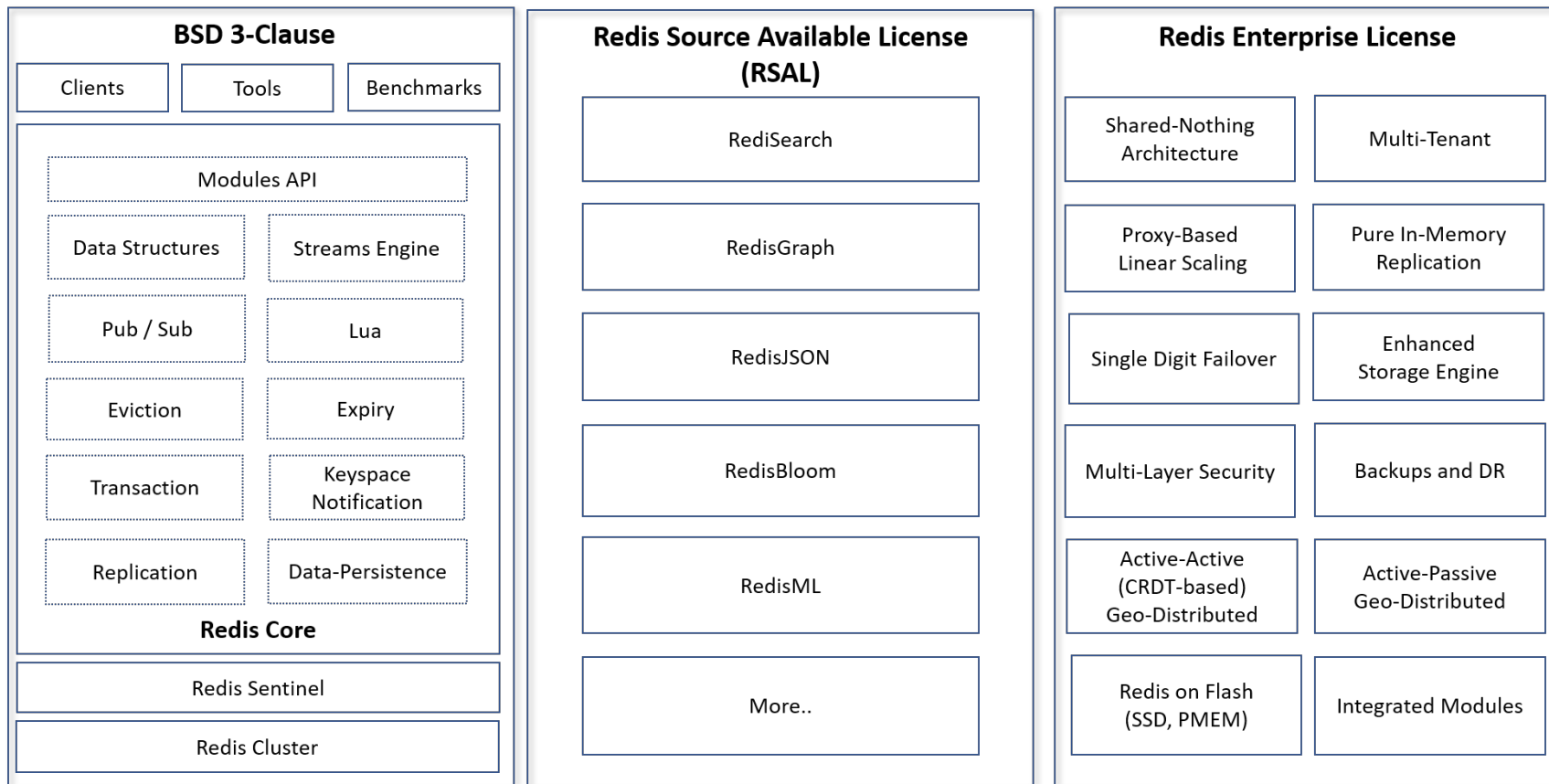


Component	From-License	To-License
Community server	AGPLv3 (and commercial)	SSPL (and commercial)
Connectors and drivers	Apache 2.0 (and commercial)	Apache 2.0 (and commercial)
Cloud management	Commercial (only)	Commercial (only)

# Other (Similar) Licensing Changes

Who?	What?	When?	From License	To License
	MaxScale (Proxy Server)	2016	GPLv2	BSL
	Extensions	2019	Apache 2.0	CCL
	Extensions	2019	AGPLv3	RSAL

# Redis After Licensing Change (AGPLv3 to RSAL) [1]



[1] <https://redislabs.com/blog/redis-labs-modules-license-changes/>

# Triple-Licensing Components to Keep Competitors at Bay

## Dual-Licensing



## Triple-Licensing



## 5. Use of Control Mechanisms

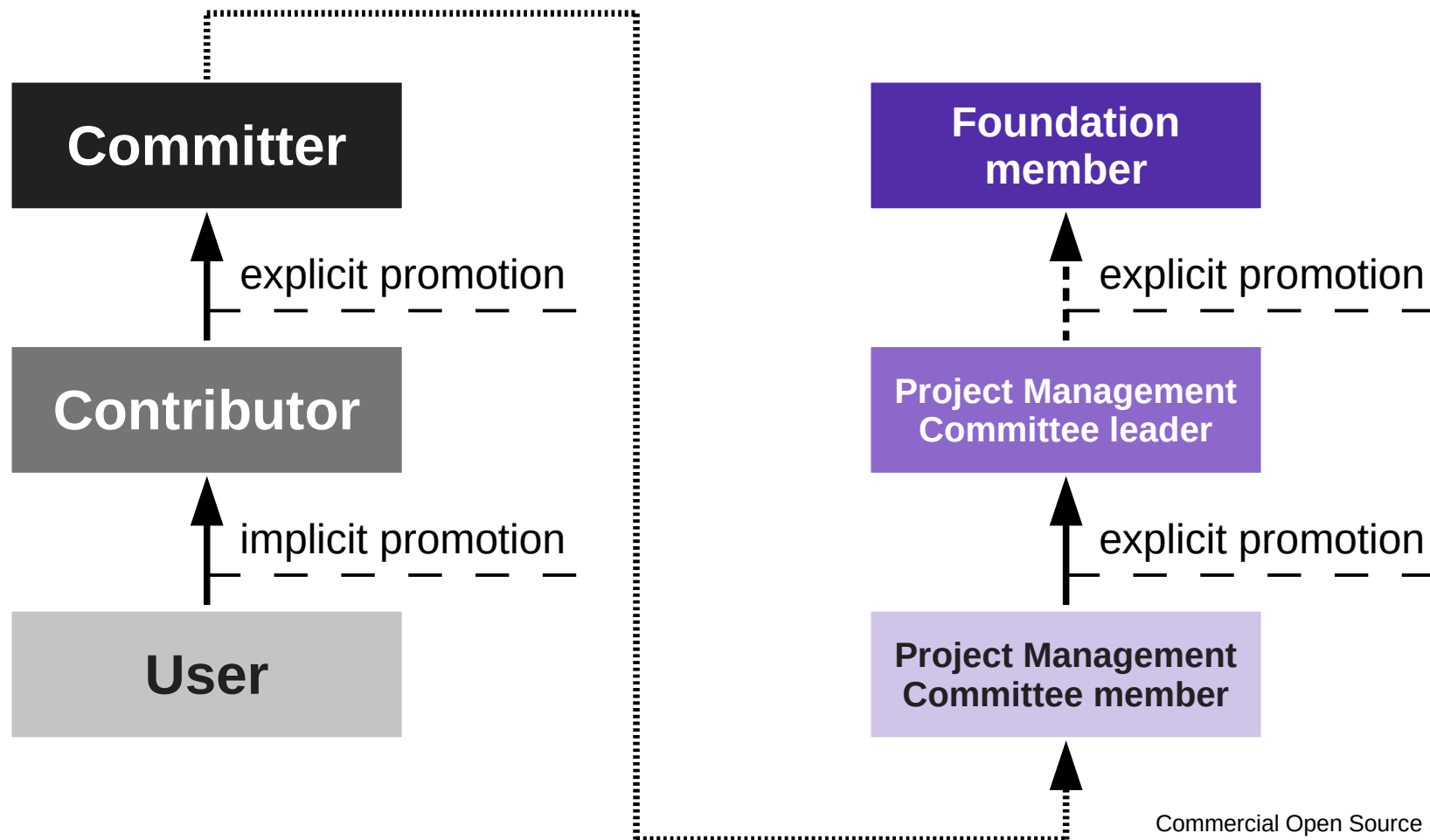


# Community vs. Commercial Open Source

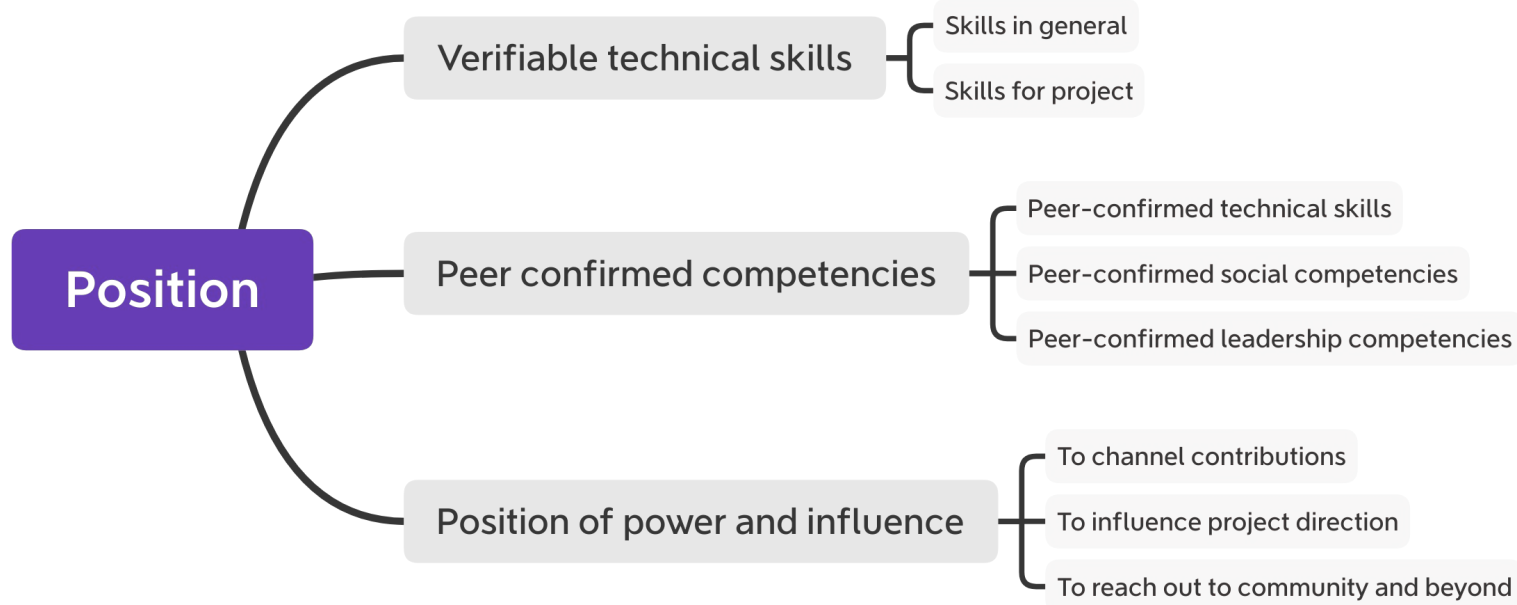
	Community Open Source		Commercial Open Source
	Traditional	Foundation	
Copyright control	Distributed	Foundation	Company
Patent ownership	Individual	Agreement	Company
Trademark control	Individual, if any	Foundation	Company
Domain ownership	Individual	Foundation	Company
Social leadership	Distributed	Distributed	Company
Committer rights	Earned	Earned	Assigned

## 6. Labor Economics

# Position in Project = Status (Recap)



# Value of Status in Open Source Project to Employers

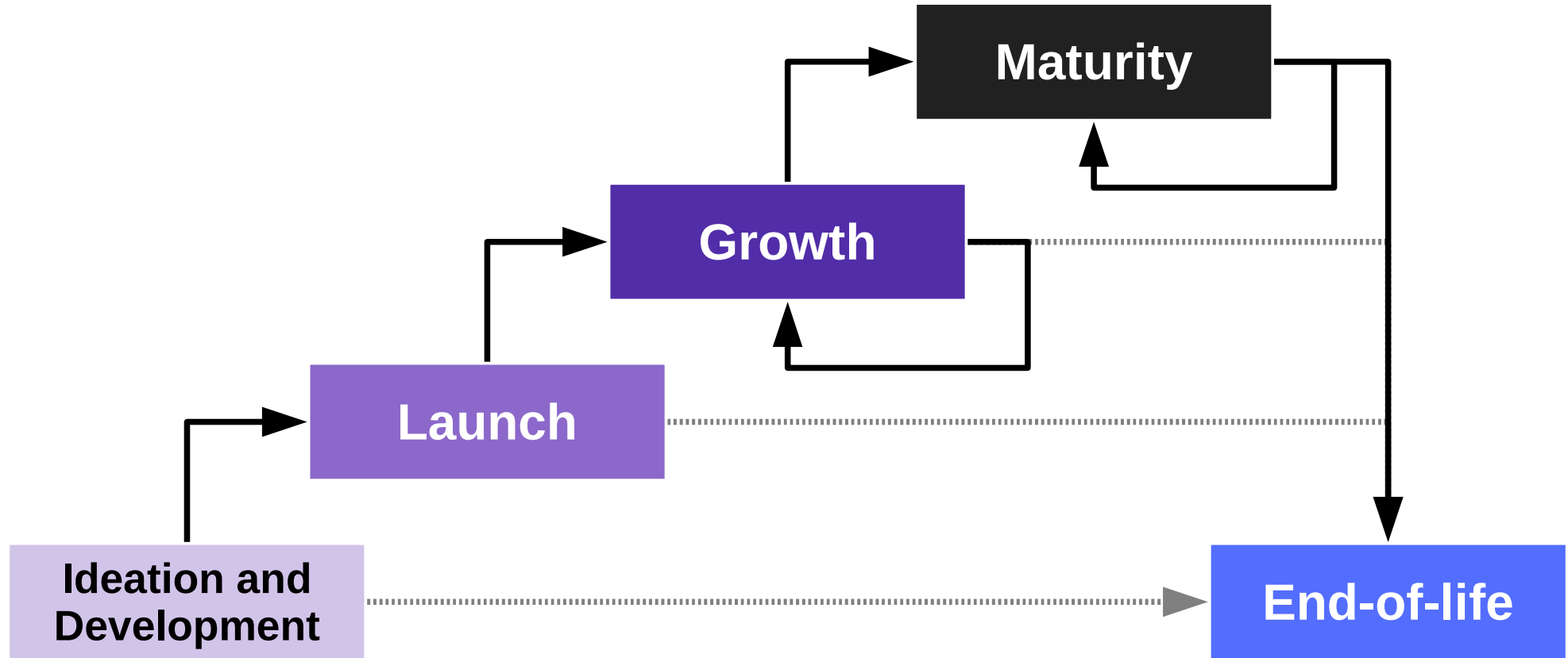


# Resulting Value to Developer

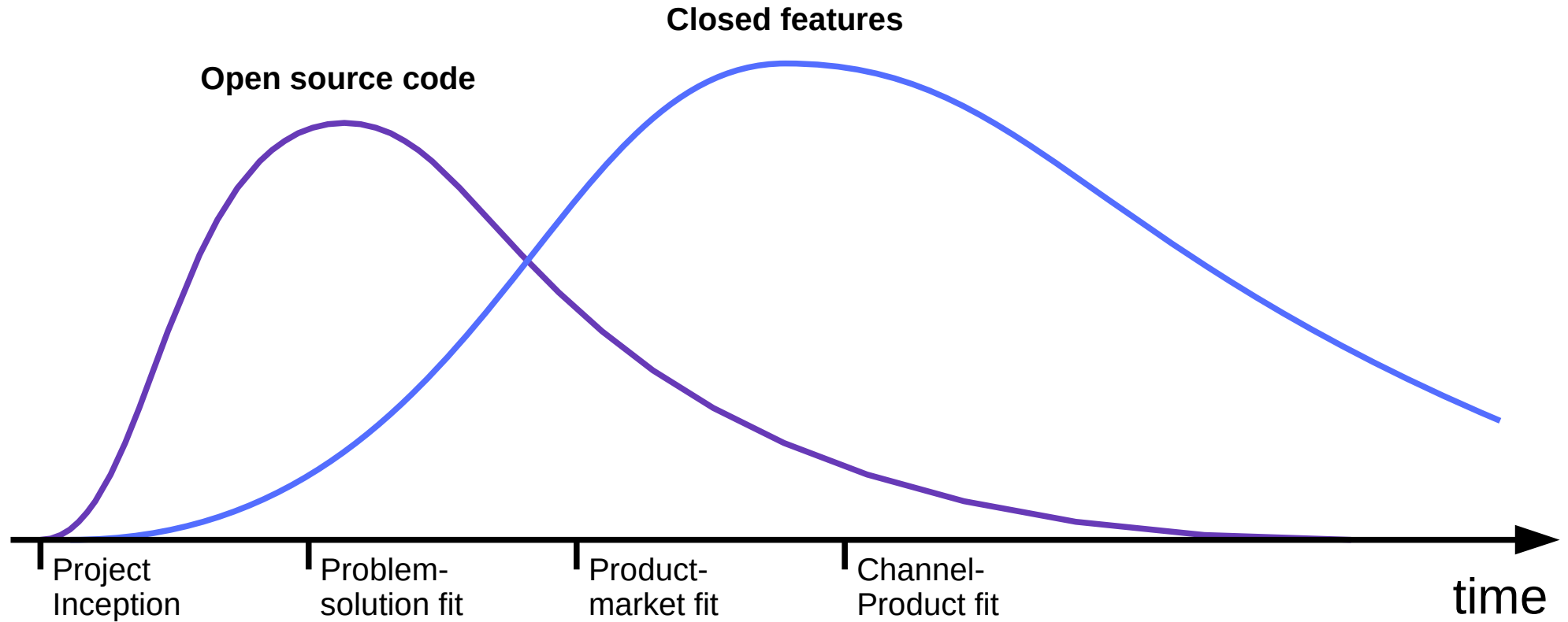
- Better negotiation position
- Higher job security
- Richer job experience

## 7. The Single-Vendor Life-cycle

# Product Life-cycle (Recap)



# Where New Features (Innovation) Goes





# Life-cycle of Single-Vendor Firms

- Early years
  - Full fair open source play
- Growth years
  - Full fair open source play
- Maturity
  - Increased closing of product

# Summary

1. Feature differentiation
2. The open core model
3. Licensing strategies
4. Cloud computing challenges
5. Use of control mechanisms
6. Labor economics
7. Single-vendor life-cycle

# Thank you! Questions?

[dirk.riehle@fau.de](mailto:dirk.riehle@fau.de) – <http://osr.cs.fau.de>

[dirk@riehle.org](mailto:dirk@riehle.org) – <http://dirkriehle.com> – [@dirkriehle](#)

# Credits and License

- Original version
  - © 2020 Dirk Riehle, some rights reserved
  - Licensed under [Creative Commons Attribution 4.0 International License](#)
- Contributions
  - None yet