FaaS

Antonio Brogi

Department of Computer Science
University of Pisa

AWS Lambda

- You can **run code** without provisioning or managing servers, all **with zero administration**
- Just upload your code and **Lambda takes care of running and scaling** your code with high availability
- You can set up your code to **automatically trigger from other AWS services** or call it directly from any web or mobile app
- You **pay only for the compute time** you consume

AWS Lambda

https://www.youtube.com/watch?v=eOBq__h4OJ4

AWS Lambda

Easy to use

- Upload your function / design it in AWS IDE / select from list of pre-built samples
- Select the event source to monitor (e.g. S3 bucket)

Then Lambda

- triggers your function automatically when an event occurs
- handles all capacities, scaling, patching and admin of the infrastructure to run your code – and publishes real time metrics and logs
- with a low cost service (low fee per request)

https://www.youtube.com/watch?v=PEatXsXIkLc

Pricing

Monthly compute = $0.0000166667 per GB-s (free tier = 400,000 GB-s)
Monthly request = $0.20 per 1M requests (free tier = 1M requests per month)

Your application:

- 3M requests per month
- average function execution duration is 120 ms
- function configured with 1536 MB of memory

Monthly compute charges

((3,000,000 x 0.12 x 1536 / 1024) – 400,000) x 0.0000166667) = $ 2.33

Monthly request charges

(3,000,000 – 1,000,000) x 0,2 / 1,000,000 = $ 0.40

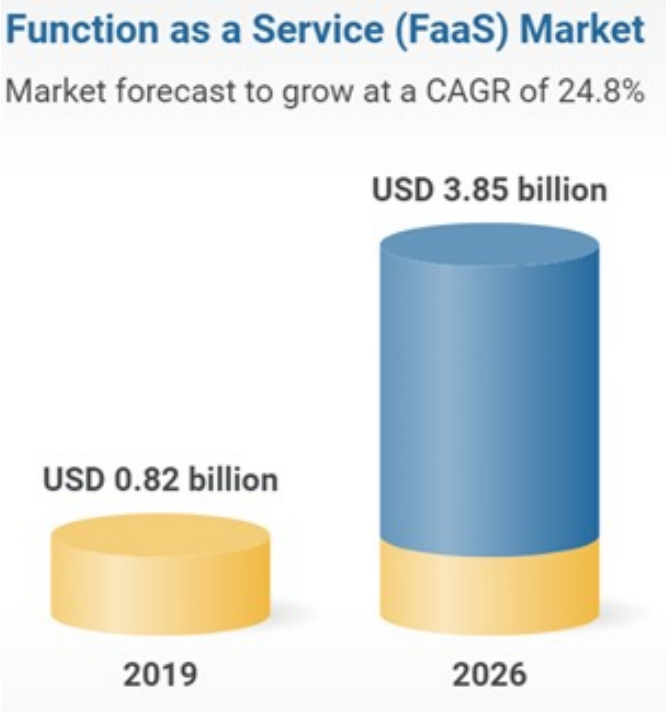
Total monthly charges = $ 2.73

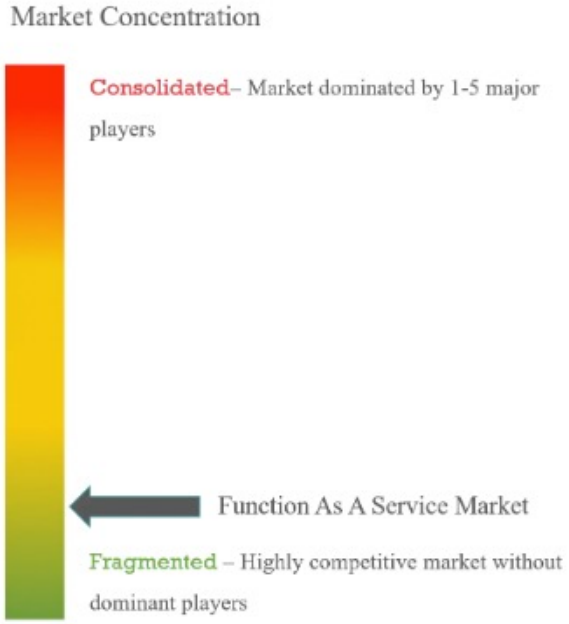
[Why the name «lambda»]

$$\lambda x \,.\, f(x)$$

https://www.youtube.com/watch?v=DSrg7hG-jV4 [0:00 – 2:04]

FaaS market





researchandmarkets.com

mordorintelligence.com

How to choose which FaaS platform to use?

Comparison of 10 FaaS platforms

Commercial

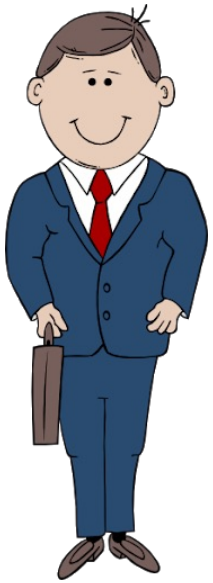
- AWS Lambda
- Google Cloud Functions
- MS Azure Functions

Open source

- Apache Openwhisk
- Fission
- Fn
- Knative
- Kubeless
- Nuclio
- OpenFaaS

V. Yussupov, J. Soldani, U. Breitenbuecher, A. Brogi, F. Leymann. FaaSten your decisions: A classification framework and technology review of Function-as-a-Service platforms. Journal of Systems and Software. 2021.

Two views



Business view

- Licensing
- Installation
- Source code
- Interface
- Community
- Documentation

Technical view

- Development
- Version management
- Event sources
- Function orchestration
- Testing and debugging
- Observability
- Application delivery
- Code reuse
- Access management

Business view: Licensing

- ➔ All open source platforms use a permissive license, with Apache 2.0 being the most used.
- ➔ All commercial FaaS platforms use proprietary licenses, with MS Azure Functions also releasing some of its components as open source projects.

	License	Type
Apache Openwhisk	Apache 2.0	Permissive
AWS Lambda	AWS service terms	Proprietary
Fission	Apache 2.0	Permissive
Fn	Apache 2.0	Permissive
Google Cloud Functions	Google Cloud platform terms	Proprietary
Knative	Apache 2.0	Permissive
Kubeless	Apache 2.0	Permissive
MS Azure Functions	SLA for Azure Functions	Proprietary
Nuclio	Apache 2.0	Permissive
OpenFaaS	MIT	Permissive

Business view: Installation

- ➔ Among commercial platforms, only Azure Functions has some parts installable on-premises.
- ➔ Installable platforms support multiple target hosts, and Kubernetes is the most supported.

	Type	Target hosts
Apache Openwhisk	Installable	Docker, Kubernetes, Linux, MacOS, Mesos, Windows
AWS Lambda	as-a-service	n/a
Fission	Installable	Kubernetes
Fn	Installable	Docker, Linux, MacOS, Unix, Windows
Google Cloud Functions	as-a-service	n/a
Knative	Installable	Kubernetes
Kubeless	Installable	Kubernetes, Linux, MacOS, Windows
MS Azure Functions	as-a-service, installable	Linux, Kubernetes, MacOS, Windows
Nuclio	as-a-service, installable	Kubernetes
OpenFaaS	Installable	Docker[a], faasd, Kubernetes, OpenShift

Business view: Source code

- All open source platforms are hosted on GitHub, and most of them are implemented in Go.
- Azure Functions is the only commercial platform that is partially open sourced.

	Availability	Open source repository	Programming language
Apache Openwhisk	Open source	GitHub	Scala
AWS Lambda	Closed source	n/a	n/a
Fission	Open source	GitHub	Go
Fn	Open source	GitHub	Go
Google Cloud Functions	Closed source	n/a	n/a
Knative	Open source	GitHub	Go
Kubeless	Open source	GitHub	Go
MS Azure Functions	Open source[a]	GitHub	C#
Nuclio	Open source	GitHub	Go
OpenFaaS	Open source	GitHub	Go

Business view: Interface

- ➔ All platforms provide CLIs, whereas APIs and GUIs are not always provided.
- ➔ Open source platforms vary considerably in the way they can be administered.

		Apache Openwhisk	AWS Lambda	Fission	Fn	Google Cloud Functions	Knative	Kubeless	MS Azure Functions	Nuclio	OpenFaaS
Type	cli	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	api	✓	✓	✓	✓	✓	✓	×	✓	×	✓
	gui	×	✓	×	✓	✓	×	×	✓	✓	✓
App. Man.	create	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	retrieve	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	update	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	delete	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Plat. Adm.	deployment	✓	×	✓	✓	×	✓	✓	×	✓	✓
	configuration	✓	×	✓	✓	×	✓	✓	×	✓	×
	enactment	✓	×	✓	✓	×	✓	✓	×	✓	✓
	termination	×[a]	×	×[a]	✓	×	×[a]	×[a]	×	×[a]	×[a]
	undeployment	×[a]	×	×[a]	×[a]	×	×[a]	×[a]	×	×[a]	×[a]

[a]Termination/undeployment can be achieved by stopping/uninstalling the platform instance with host commands.

Business view: Community

- ➔ OpenFaaS, Apache Openwhisk, and Knative have the highest ratings on GitHub in terms of stars, contributors, and commits, respectively.
- ➔ Stackoverflow questions show a drastic difference in interest between commercial and open source platforms.

		Apache Openwhisk	AWS Lambda	Fission	Fn	Google Cloud Functions	Knative	Kubeless	MS Azure Functions	Nuclio	OpenFaaS
GitHub	Stars	4.8K	n/a	5.2K	4.6K	n/a	3K[a]	5.8K	n/a	3.3K	17.9K
	Forks	932	n/a	487	349	n/a	637[a]	591	n/a	332	1.5K
	Issues	274	n/a	215	125	n/a	223[a]	164	n/a	51	62
	Commits	2.8K	n/a	1.2K	3.4K	n/a	4.7K[a]	1K	n/a	1.4K	1.9K
	Contributors	180	n/a	104	86	n/a	185[a]	89	n/a	55	147
SO	Questions	198	16.8K	7	25	10.1K	71	8	7.2K	3	29

[a]Values for the function hosting component of Knative, i.e., Knative Serving.

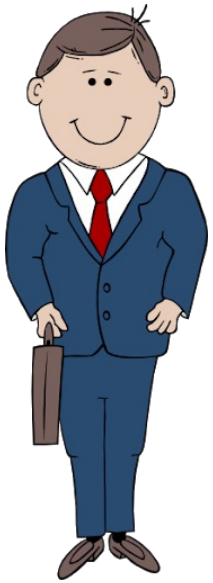
Business view: Documentation

- ➔ All platforms provide application deployment and platform usage documentation.
- ➔ Platform development and architecture are not always documented by open source platforms.

		Apache Openwhisk	AWS Lambda	Fission	Fn	Google Cloud Functions	Knative	Kubeless	MS Azure Functions	Nuclio	OpenFaaS
App.	Development	✓	✓	✓	✓	✓	×	✓	✓	×	×
	Deployment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Platform	Usage	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Development	✓	×	✓	×[a]	×	✓	✓	×	×[a]	✓
	Deployment	✓	×	✓	✓	×	✓	✓	✓	✓	✓
	Architecture	✓	×	✓	✓	×	×	✓	×	✓	✓

[a]Only providing guidelines/code of conduct for contributing to the project.

Two views



Business view

- Licensing
- Installation
- Source code
- Interface
- Community
- Documentation

Technical view

- Development
- Version management
- Event sources
- Function orchestration
- Testing and debugging
- Observability
- Application delivery
- Code reuse
- Access management

Technical view: Development

- ➔ Java, Node.js, and Python are the most supported runtimes, with Docker also being quite popular to help customizing runtimes.
- ➔ IDEs and text editor plugins are mainly offered by commercial platforms.

Classification of considered FaaS platforms, based on the *Development* category in the *technical* view of our classification framework. D and E are used to denote that quotas are set for *deployment package size* and *execution time*, respectively. The abbreviation "n/s" stays for "not specified", meaning that no related information is in the documentation.

	IDEs and Text Editors	Client Libraries	Quotas
Apache Openwhisk	Visual Studio Code[a], Xcode[a]	Go, Node.js	E[b]
AWS Lambda	AWS Cloud9, Eclipse, Toolkit for JetBrains, Visual Studio, Visual Studio Code	Go, Java, MS .NET, Node.js, Python, Ruby, C++	D, E
Fission	n/s	n/s	n/s
Fn	n/s	Go	n/s
Google Cloud Functions	n/s	Dart, Go, Java, MS .NET, Node.js, Python, Ruby	D, E
Knative	n/s	n/s	n/s
Kubeless	Visual Studio Code	n/s	n/s
MS Azure Functions	Visual Studio, Visual Studio Code	Java, MS .NET, Python	D, E[b]
Nuclio	Jupyter Notebooks	Go, Java, MS .NET, Python	n/s
OpenFaaS	n/s	n/s	n/s

[a]Deprecated/No longer maintained.

[b]Bounded by default, but can be configured to unset quota.

Technical view: Versioning

➔ Most open source platforms employ implicit versioning, without providing dedicated mechanisms opposing to commercial platforms.

***D**edicated mechanisms vs. **I**mplicit mechanisms (e.g., encoding version identifiers as part of function or application names or namespaces)*

Classification of considered FaaS platforms, based on the *Version Management* category in the *technical* view of our classification framework. D and I are used to denote the possible values *dedicated mechanisms* and *implicit versioning*, respectively. The abbreviation "n/s" stays for "not specified", meaning that a platform does not explicitly mention the versioning of serverless applications.

	Application versions	Function versions
Apache Openwhisk	I	I
AWS Lambda	D	D
Fission	I	I
Fn	I	D
Google Cloud Functions	I	I
Knative	n/s	D
Kubeless	n/s	I
MS Azure Functions	I	I
Nuclio	n/s	D
OpenFaaS	n/s	I

Technical view: Event sources

- ➔ All platforms support synchronous, HTTP-based function invocation, whereas asynchronous calls are supported by few platforms.
- ➔ More than a half of open source platforms do not support data store event sources.
- ➔ Schedulers and stream processing platforms are supported by most platforms. Messaging is also widely supported.
- ➔ More than a half of reviewed platforms document integration of custom event sources.

Technical view: Function orchestration

➔ More than a half of reviewed FaaS platforms support function orchestration using either custom DSLs or orchestrating functions.

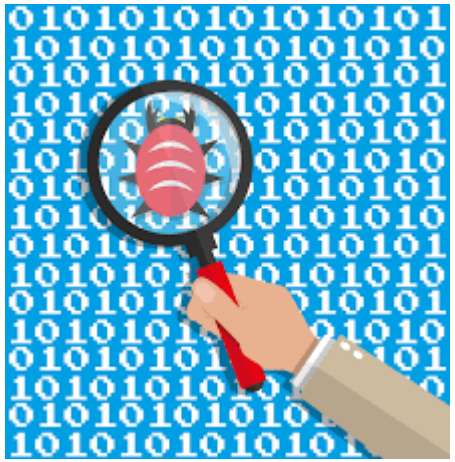
Classification of considered FaaS platforms, based on the *Function Orchestration* category in the *technical* view of our classification framework. C denotes *custom DSL*, and O denotes *orchestrating function*, with the list of supported programming languages for developing orchestrating functions given in square braces. The abbreviations "n/s" and "n/a" stay for "not specified" and "not applicable", respectively.

	Orchestrator	Workflow definition	Control flow described	Quotas
Apache Openwhisk	Openwhisk composer	O [JavaScript, Python]	✓	Execution time
AWS Lambda	AWS step functions	C	✓	Execution time, I/O size
Fission	Fission workflows	C	✓	n/s
Fn	Fn Flow	O [Java]	✓	n/s
Google Cloud Functions	n/s	n/a	n/a	n/a
Knative	Knative eventing	C	✓[a]	n/s
Kubeless	n/s	n/a	n/a	n/a
MS Azure Functions	Azure durable functions	O [C#, JavaScript]	✓	n/s
Nuclio	n/s	n/a	n/a	n/a
OpenFaaS	n/s	n/a	n/a	n/a

[a]Only sequence and parallel execution are supported.

Technical view: Testing & debugging

- The majority of reviewed platforms support functional testing and debugging of functions.
- Commercial platforms offer more sophisticated options, whereas open source platforms mainly rely on test calls and log-based debugging.



Technical view: Observability

➔ Commercial platforms offer dedicated tooling for the monitoring and logging of functions, whereas open source platforms rely on the integration of third-party tooling.

	Monitoring	Logging	Tooling Integr.
Apache Openwhisk	Kamon, Prometheus, Datadog	Logback (slf4j)	n/s
AWS Lambda	AWS CloudWatch	AWS CloudTrail, CloudWatch	n/s
Fission	Istio + Prometheus	Fission Logger + InfluxDB, Istio + Jaeger	Using a service mesh
Fn	Prometheus, Zipkin, Jaeger	Docker container logs	Push-based
Google Cloud Functions	Google Cloud Operations	Google Cloud Operations	n/s
Knative	Prometheus + Grafana, Zipkin, Jaeger	ElasticSearch + Kibana, Google Cloud Operations	Push-based
Kubeless	Prometheus + Grafana	n/s	n/s
MS Azure Functions	Azure Application Insights	Azure Application Insights	n/s
Nuclio	Prometheus, Azure Application Insights	stdout, Azure Application Insights	Push-based, pull-based
OpenFaaS	OpenFaaS Gateway + Prometheus	Kubernetes cluster API, Swarm cluster API, Loki	Pull-based

Technical view: Application delivery

- ➔ Most reviewed platforms follow a declarative approach to automate application deployment.
- ➔ Commercial platforms natively support CI/CD throughout dedicated tooling, whereas open source platforms (apart from OpenFaaS) do not document integration with CI/CD pipelines.

Classification of considered FaaS platforms, based on the *Application Delivery* category in the *technical* view of our classification framework. P and T denote *Platform-native tooling* and *third party tooling*, respectively. The abbreviation "n/s" stays for "not specified", meaning that no related information is documented.

	Deployment automation	CI/CD
Apache Openwhisk	wskdeploy (P)	n/s
AWS Lambda	AWS Cloud Formation (P), AWS SAM (P)	AWS CodePipeline(P)
Fission	Fission CLI (P)	n/s
Fn	Fn CLI (P)	n/s
Google Cloud Functions	Google Cloud Deployment Manager	Cloud Build (P)
Knative Kubeless	Kubernetes (P)[a] Kubernetes (P)[a], Serverless Framework (T)	n/s n/s
MS Azure Functions	Azure Resource Manager (P)	Azure Pipelines (P), Azure App Service (P), Jenkins (T)
Nuclio	nuctl (P)[a]	n/s
OpenFaaS	faas-cli (P)	OpenFaaS Cloud (P), faas-cli (P), Circle CI (T), CodeFresh (T), Drone CI (T), GitLab CI/CD (T), Jenkins (T), Travis (T)

[a]Using Kubernetes specification with Custom Resource Definitions.

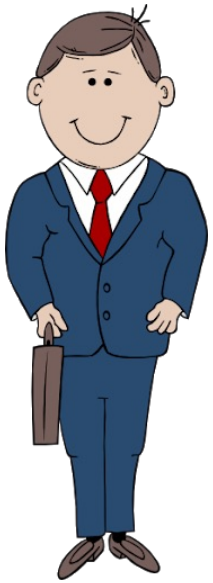
Technical view: Code reuse

➔ Only AWS Lambda and MS Azure Functions are accompanied by a function marketplace.

Technical view: Access management

- ➔ Commercial platforms natively support authentication and resource access control.
- ➔ Open source platforms typically rely on features offered by the hosting environment to enforce authentication and resource access control.

Two views



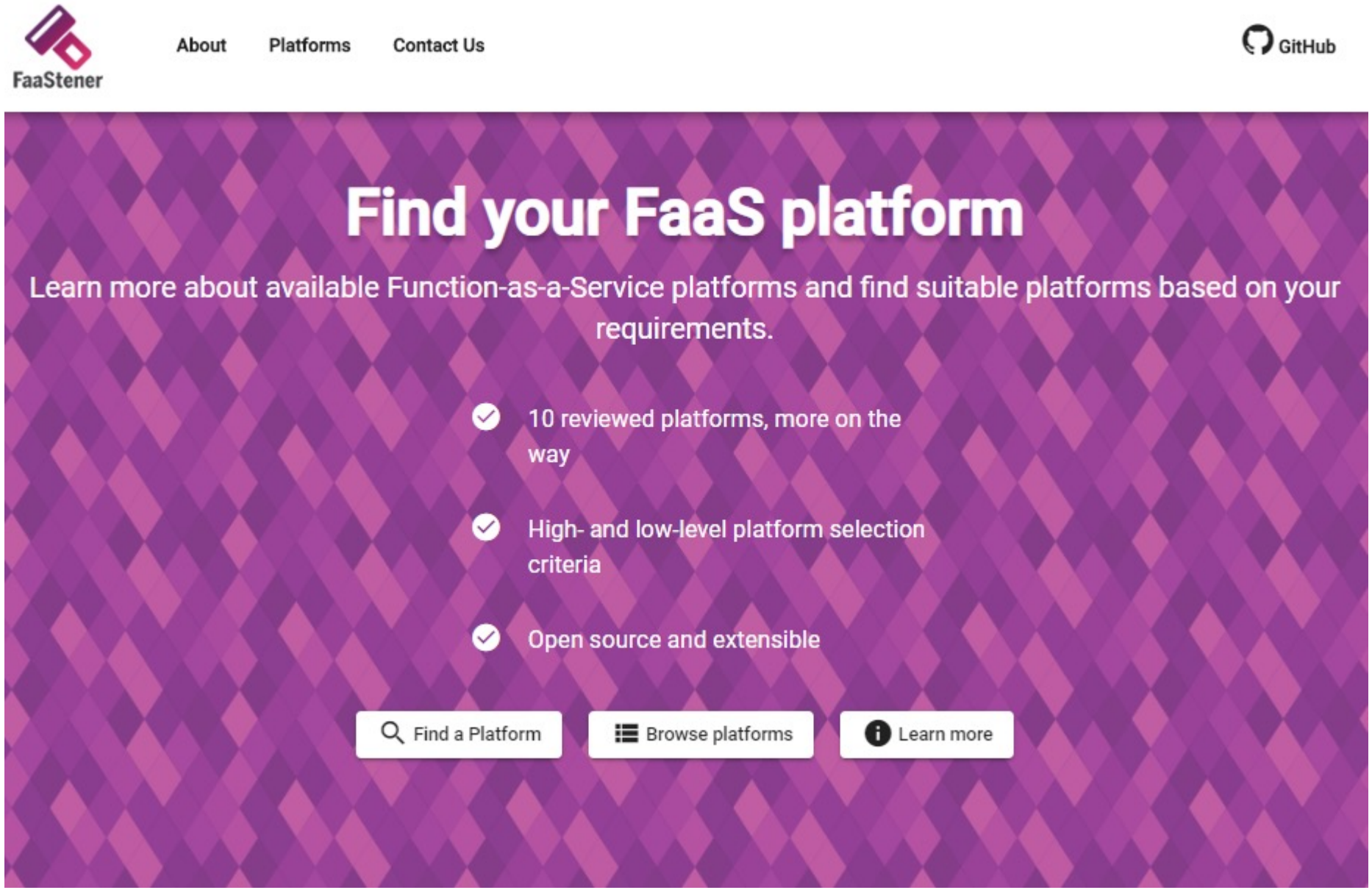
Business view

- Licensing
- Installation
- Source code
- Interface
- Community
- Documentation

Technical view

- Development
- Version management
- Event sources
- Function orchestration
- Testing and debugging
- Observability
- Application delivery
- Code reuse
- Access management

FaaStener prototype



https://faastener.github.io/home

Definition - Vendor lock-in makes a customer dependent on a vendor for products or services, unable to use another vendor without substantial switching costs.

Lock-in risks with FaaS?

"The way that AWS Lambda or OpenWhisk calls into your serverless code can vary. Coupling your code too tightly to a serverless vendor's APIs can make it hard to move the code to another platform."

Dean Hallman - CTO of WireSoft

"It is possible to architect applications that are serverless yet portable"

Jason Umiker, AWS

"The challenge with lock-in is derived from the tight coupling between the serverless functions and the cloud provider services"

Yaron Haviv - CTO of Iguazio

'Lambda and serverless is one of the worst forms of proprietary lock-in we've ever seen in the history of humanity'

Alex Polvi , CoreOS CEO