

Container Patterns

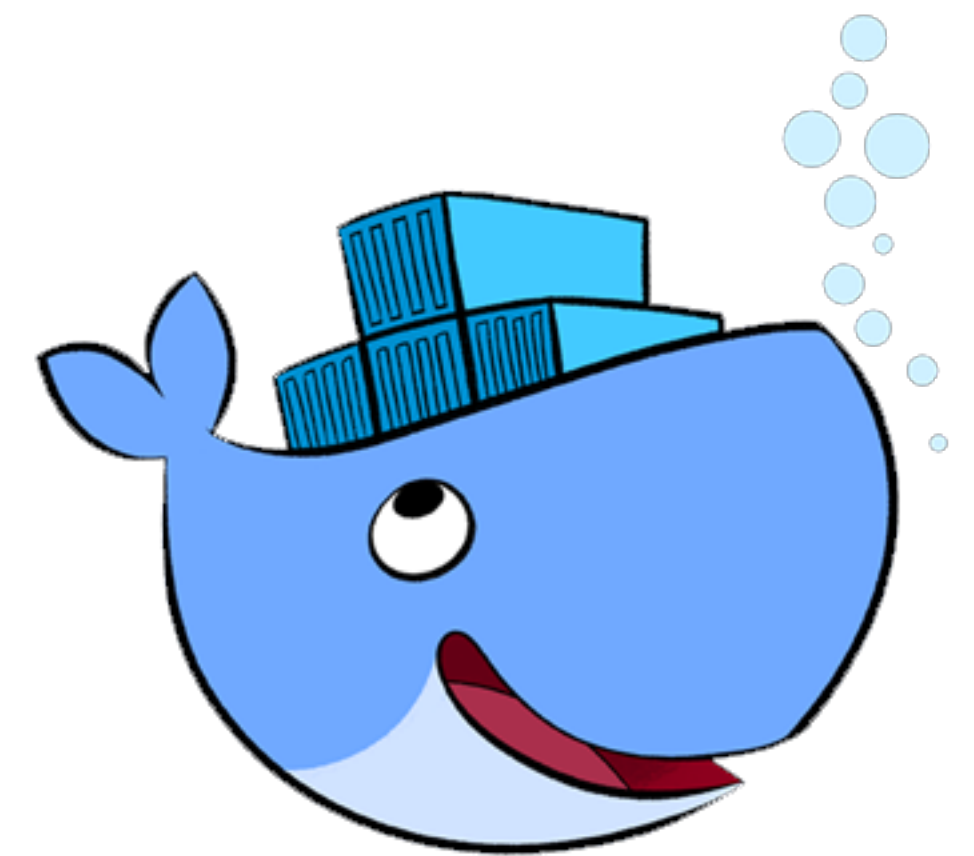
Matthias Lübken

QCon
LONDON



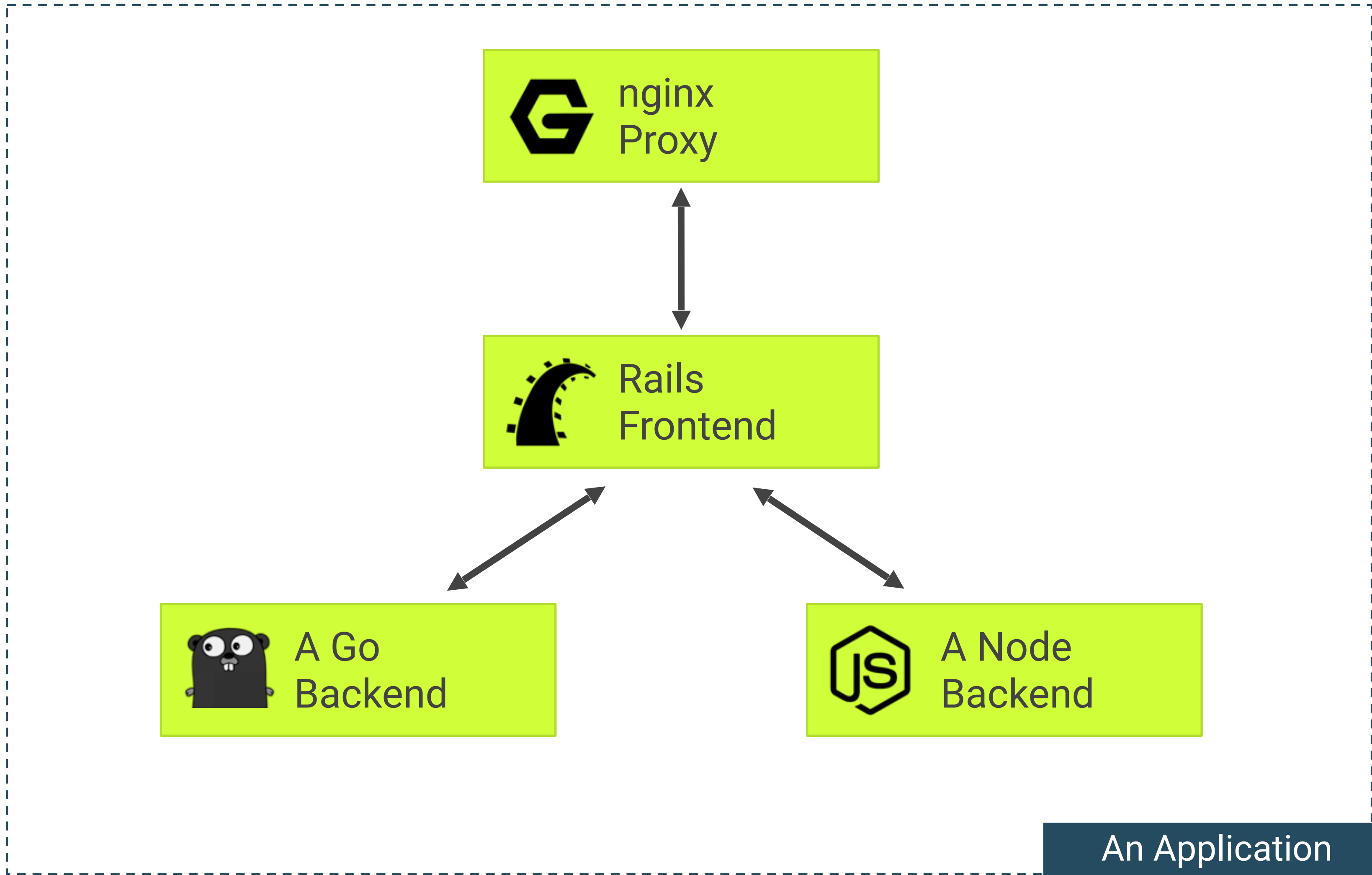
[@luebken](#)

“ Easily create lightweight,
portable, self-sufficient containers
from any **application.**”



“App Container (appc) is a well-specified and community developed specification for **application** containers.”







Independently
releasable

Separate
processing types

Different
loads

Reuse

Different
teams

POCs

Crash
isolation

Use different
languages / versions / libraries



How does a good building block look like?

How do we assemble them?

Related Work

- 12factor.net apps
- Cloud-native application architectures:
Matt Stine Free Ebook
- Microservices
- Continuous Delivery

Container Patterns

- For designing “cloud” applications.
- Container **runtime agnostic**.
- Are there **general applicable** patterns?
- How would we **describe** them?
- What are concrete **examples** and best-practices?

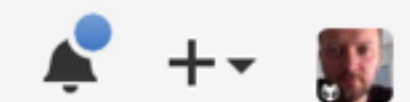


**I WANT YOU
FOR AN ADVENTURE**



This repository Search

[Pull requests](#) [Issues](#) [Gist](#)



[luebken](#) / **container-patterns**

[Watch](#) 0 [Star](#) 0 [Fork](#) 0

[Code](#)

[Issues](#) 0

[Pull requests](#) 1

[Wiki](#)

[Pulse](#)

[Graphs](#)

[Settings](#)

No description or website provided. — [Edit](#)

[1 commit](#)

[2 branches](#)

[0 releases](#)

[1 contributor](#)

Branch: [master](#)

[New pull request](#)

[New file](#)

[Upload files](#)

[Find file](#)

[SSH](#)

[git@github.com:luebken/co](#)



[Download ZIP](#)

[luebken](#) Initial commit

Latest commit a87df51 8 days ago

[LICENSE](#)

Initial commit

8 days ago

[README.md](#)

Initial commit

8 days ago

[README.md](#)

container-patterns

github.com/luebken/container-patterns





How does a good building block look like?

How do we assemble them?

“Module-Container”

A “Module-Container” is a well behaving building block in the architecture of an application.

A Module Container is

1. Linux process >_

2. API 🗑️

3. Descriptive 🏷️

4. Disposable 🗑️

5. Immutable 🔒

6. Self-contained 📦

7. Small ↗️


```
last pid: 40514; load averages: 1.04, 1.02, 0.97 up 92+00:21:50 08:00:59
23 processes: 2 running, 21 sleeping
CPU: 70.0% user, 0.0% nice, 30.0% system, 0.0% interrupt, 0.0% idle
Mem: 291M Active, 434M Inact, 210M Wired, 43M Cache, 111M Buf, 960K Free
Swap: 2012M Total, 8828K Used, 2003M Free
```

PID	USERNAME	THR	PRI	NICE	SIZE	RES	STATE	TIME	WCPU	COMMAND
39181	root	1	119	0	20204K	6240K	RUN	199:07	100.00%	perl5.16.2
20489	root	60	44	0	1543M	279M	ucond	10:01	0.00%	java
683	nagios	1	44	0	12004K	152K	select	5:28	0.00%	nrpe2
697	root	1	44	0	13096K	1080K	select	1:48	0.00%	sendmail
579	nsLCD	6	44	0	17160K	2816K	select	1:02	0.00%	nsLCD
704	root	1	76	0	1008K	152K	sleep	0:17	0.00%	cron
594	root	1	44	0	1008K	64K	select	0:14	0.00%	syslogd
694	root	1	44	0	27236K	392K	select	0:13	0.00%	sshd
700	smmsp	1	44	0	13096K	796K	pause	0:02	0.00%	sendmail
457	root	1	44	0	5248K	308K	select	0:00	0.00%	devd
32579	root	1	44	0	23848K	1812K	wait	0:00	0.00%	login
40514	root	1	44	0	10432K	1920K	RUN	0:00	0.00%	top
40505	oper	1	44	0	11308K	2476K	pause	0:00	0.00%	tcsh
40509	root	1	45	0	11308K	2552K	pause	0:00	0.00%	tcsh
40508	root	1	44	0	25992K	2316K	select	0:00	0.00%	sudo
20478	root	1	76	0	8344K	1336K	wait	0:00	0.00%	sh
743	root	1	76	0	6916K	16K	ttyin	0:00	0.00%	getty
747	root	1	76	0	6916K	16K	ttyin	0:00	0.00%	getty

1. Linux Process

1. Linux Process

- React to signals
- Return exit codes
- Use standard streams
- Handle arguments

Examples: [module-container.md#1-linux-process](#)



2. API



2. API

- ENV variables
- Available ports
- Volume mounts
- Lifecycle hooks

Examples: [module-container.md#2-api](#)



朝北开路

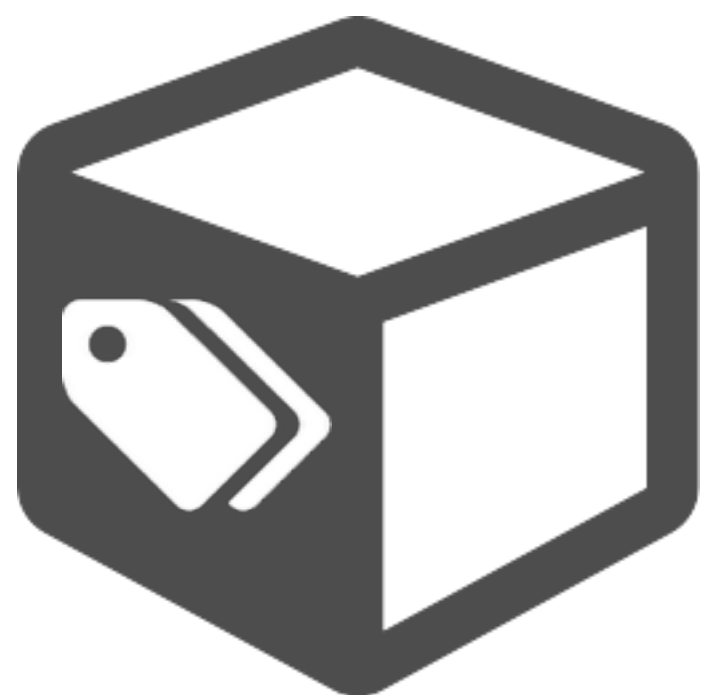
3. Descriptive

1 -

3. Descriptive

- Use standard labels (e.g. proposal generic labels)
 - url, summary, vcs-url ...
- Use custom labels:
 - api.ENV
 - api.EXPOSE
 - api.LINKS

Examples: module-container.md#3-descriptive



Dockerfile

```
1 FROM node:0.10-slim
2 MAINTAINER matthias.luebken@gmail.com
3 WORKDIR /app
4
5 # install dependencies
6 ADD package.json /app/
7 RUN npm install
8
9 # install app
10 ADD server.js /app/
11
12 # Describe container dependencies
13 LABEL api.LINKS.redis="" \
14     api.LINKS.redis.image="redis:latest" \
15     api.LINKS.redis.port="6379" \
16     api.LINKS.redis.description="For caching requests to OWM API." \
17     api.LINKS.redis.mandatory="true"
18
19 # Set and describe available ENVs
20 ENV OPENWEATHERMAP_APIKEY=182564eaf55f709a58a13c40086fb5bb
21 LABEL api.ENV.OPENWEATHERMAP_APIKEY="" \
22     api.ENV.OPENWEATHERMAP_APIKEY.description="Access key for OpenWeatherMap. See
23     api.ENV.OPENWEATHERMAP_APIKEY.mandatory="false"
24
25 # Expose and describe available ports
26 EXPOSE 1337
27 LABEL api.EXPOSE.1337="" \
28     api.EXPOSE.1337.protocol="http" \
29     api.EXPOSE.1337.description="The main endpoint of this service."
30
31 ENTRYPOINT ["node", "server.js"]
```



```
-----  
Image: luebken/currentweather-nodejs:latest  
-----
```

```
Author: matthias.luebken@gmail.com
```

```
Size: 158 MB
```

```
Created: 2016-03-07 17:25  
-----
```

```
Container API:
```

```
* Required Links:
```

```
- Image : redis:latest
```

```
> Port : 6379
```

```
> Description : For caching requests to OWM API.
```

```
> Mandatory : true
```

```
* Required ENVs:
```

```
- OPENWEATHERMAP_APIKEY
```

```
> default value : 182564eaf55f709a58a13c40086fb5bb
```

```
> description : Access key for OpenWeatherMap. See http://
```

```
> mandatory : false
```

```
* Available ports:
```

```
- 1337/tcp {}
```

```
* Volumes:
```

POC github.com/luebken/container-api

A photograph of a row of blue recycling bins on a sidewalk. The bins are lined up against a brick wall. Above the bins, there are several air conditioning units mounted on the wall. To the left, there are several bicycles parked. A white line is visible on the asphalt road in the foreground. The text "4. Disposable" is overlaid in white on a dark blue semi-transparent background across the middle of the image.

4. Disposable

4. Disposable

- Don't rely on a particular instance
- Be aware of shots at your cattle
- Be robust against sudden death

Examples: [module-container.md#4-disposable](#)





5. Immutable

5. Immutable

- Don't change your container after build
- Strive for a dev/prod parity

Examples: [module-container.md#5-immutable](#)



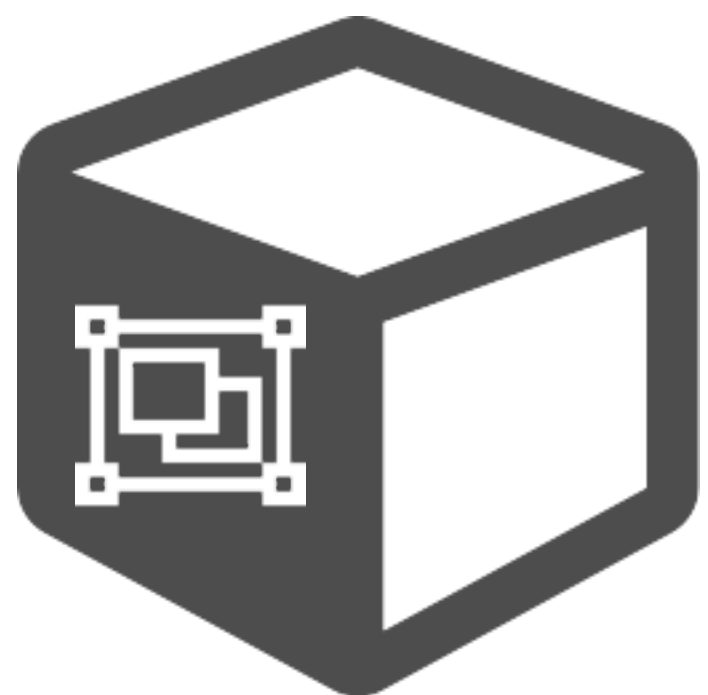


6. Self-contained

6. Self-contained

- Add dependencies on build time
- Sensible defaults

Examples: [module-container.md#6-self-contained](#)





7. Small

7. Small

- Don't use large base images
- Use the minimal footprint e.g. Alpine

Examples: [module-container.md#6-small](#)



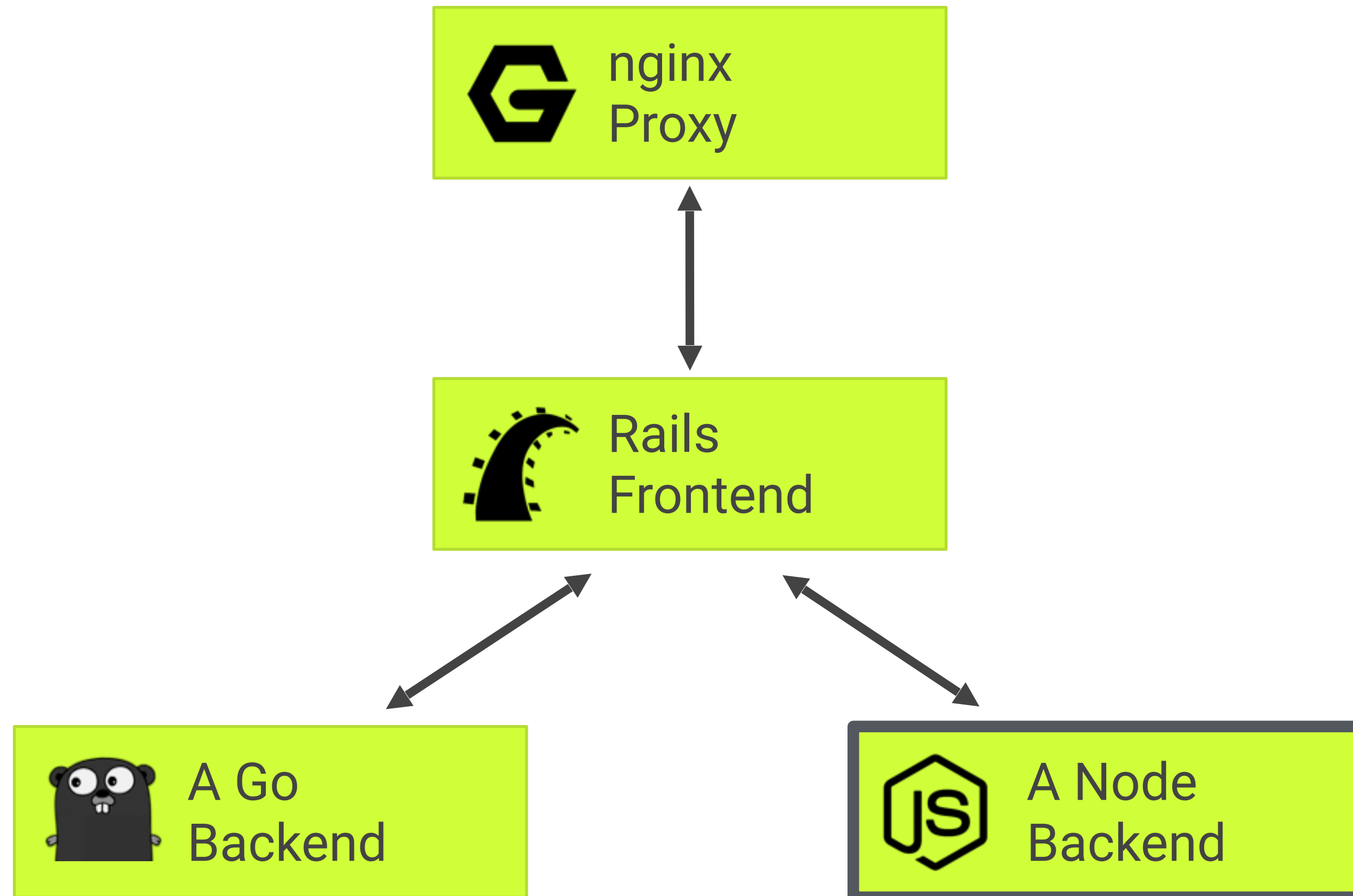
Recap: A Module Container is

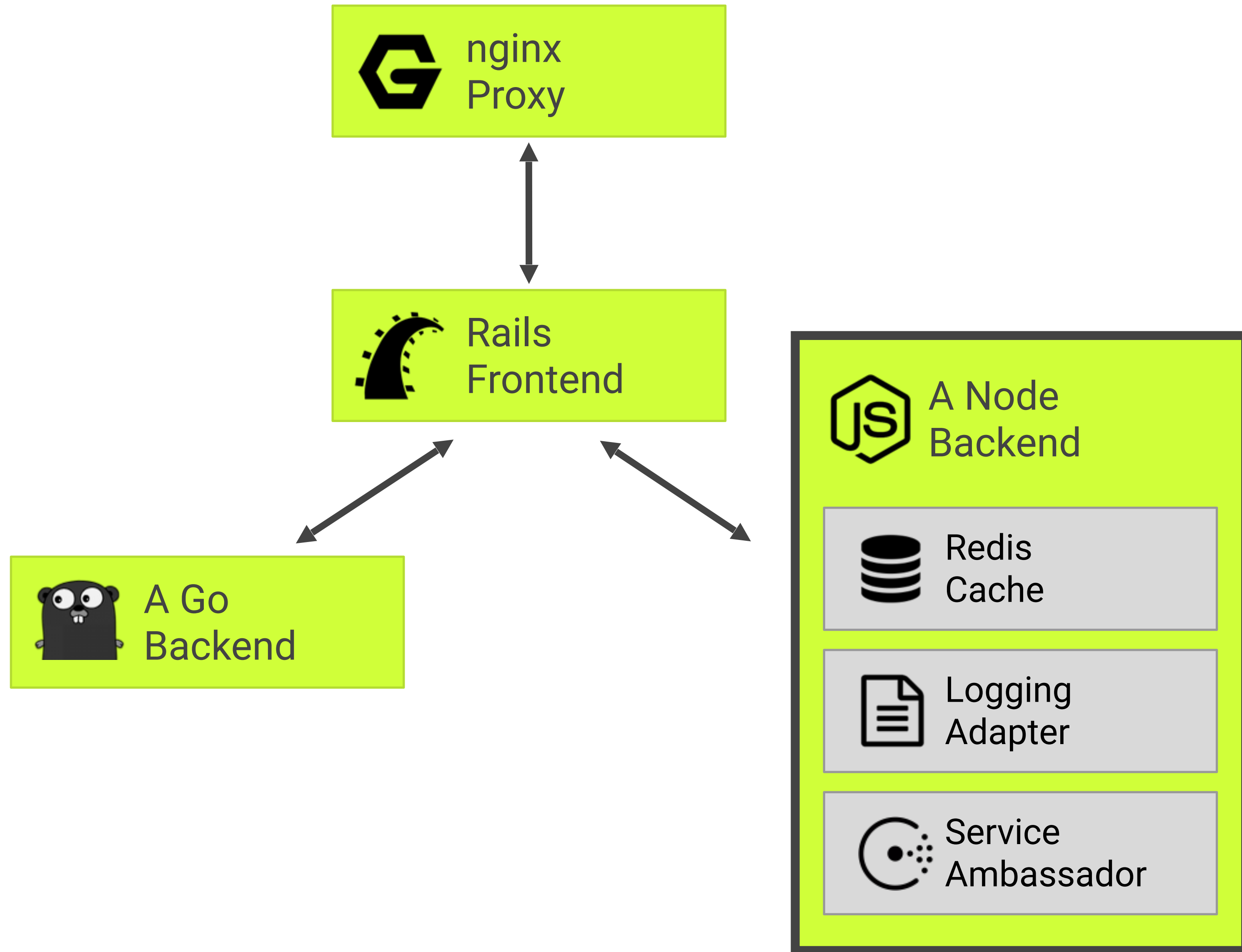
1. Linux process >_
2. API 🗡️
3. Descriptive 🏷️
4. Disposable 🗑️
5. Immutable 🔒
6. Self-contained 📦
7. Small ↗️

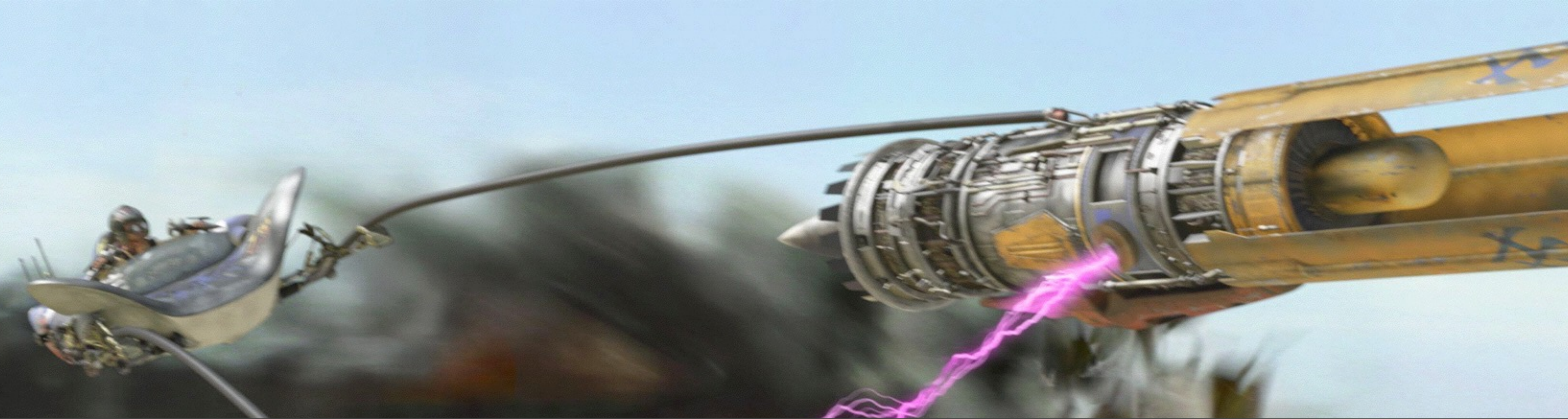


How does a good building block look like?

How do we assemble them?







A group of closely related containers.
Deployed as a single unit
and share namespaces.



A Node
Backend



Redis
Cache



Logging
Adapter

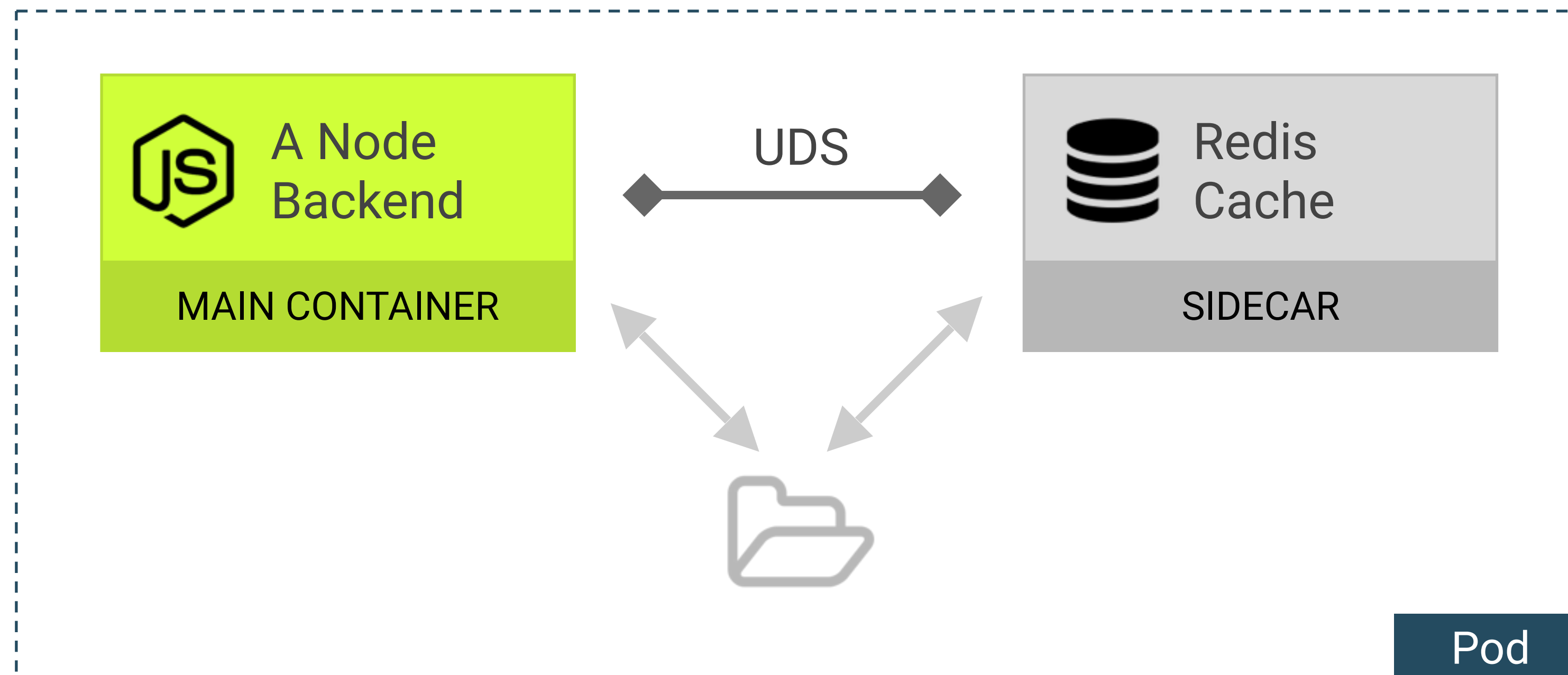


Service
Ambassador

Pattern: Sidecar / Sidekick

Enhance & extend the main container.

K8S: transparently. Netflix: platform features.





A Node
Backend



Redis
Cache



Logging
Adapter

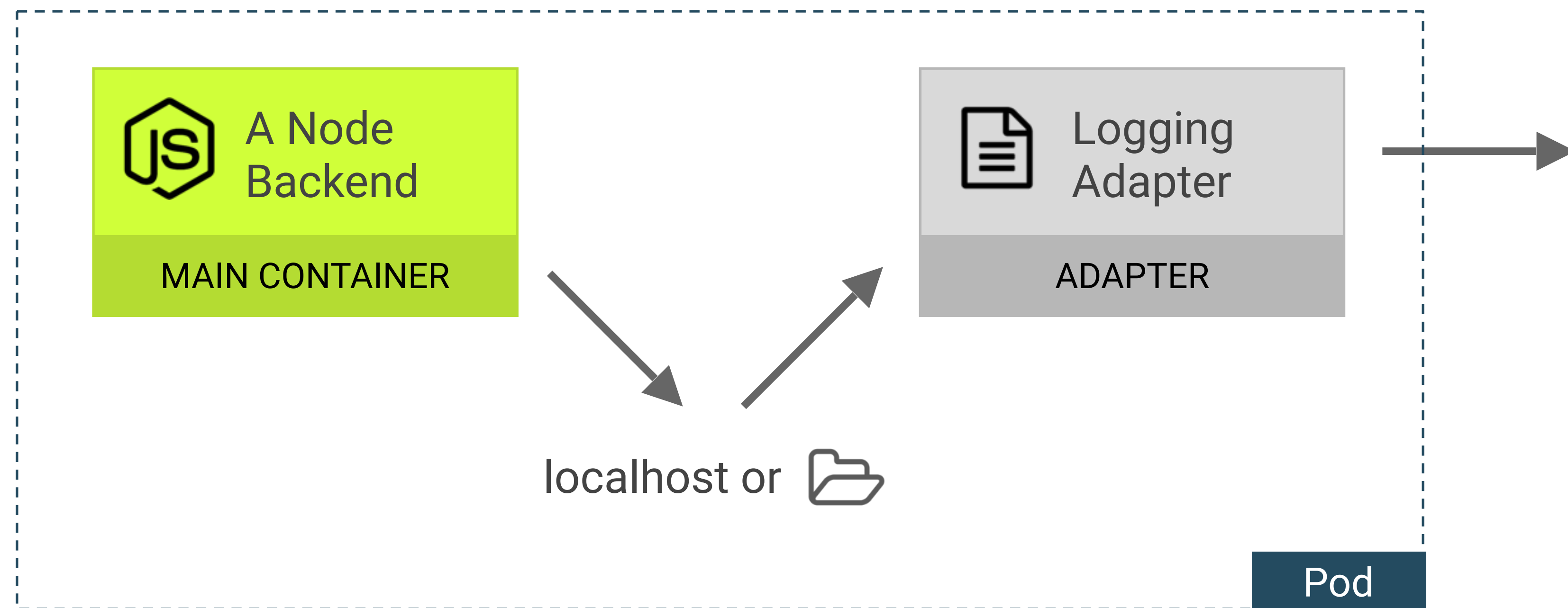


Service
Ambassador

Pattern: Adapter

Standardise and normalize output.

E.g. logging and metrics.





A Node
Backend



Redis
Cache



Logging
Adapter

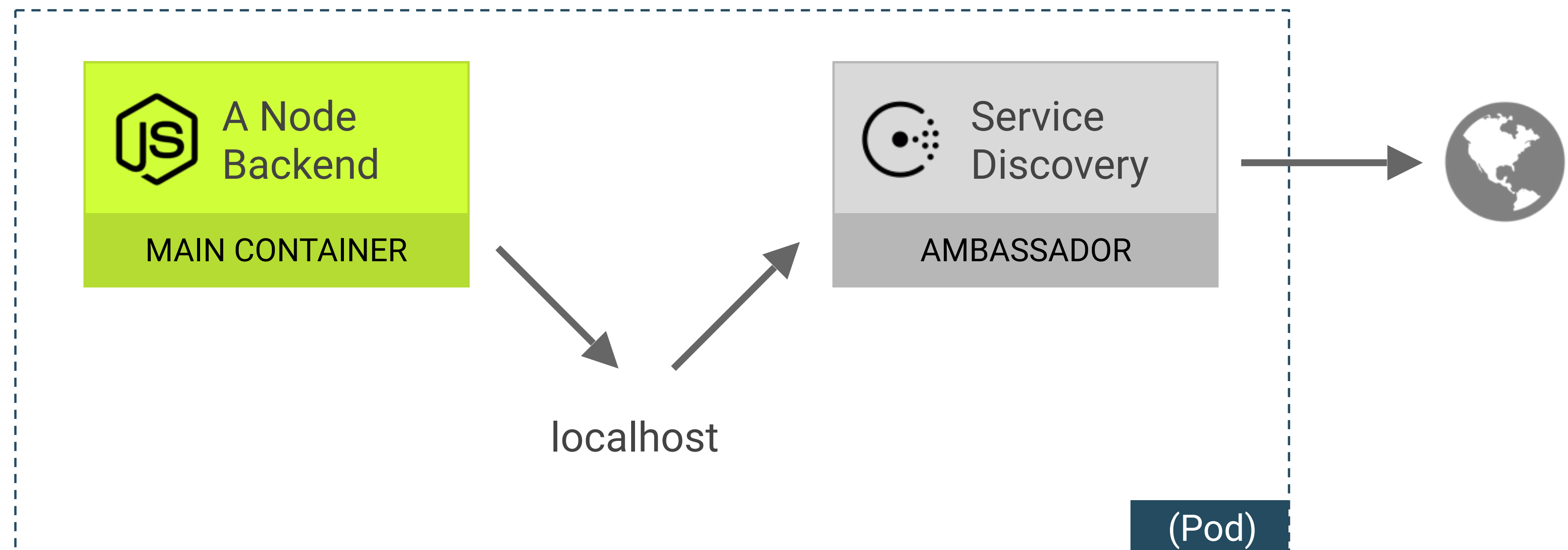


Service
Ambassador

Pattern: Ambassador

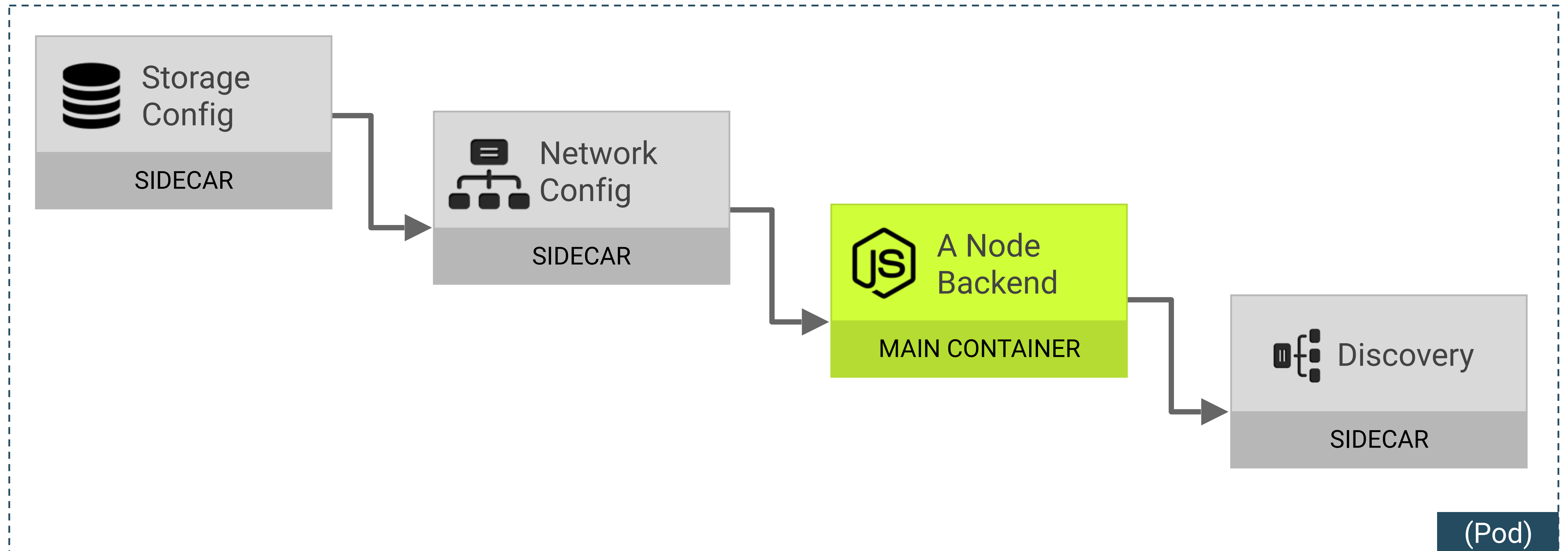
Proxy a local connection to the world:

Service Discovery, Client Side LB, Circuit Breaker



Pattern: Container chains

Defined order of starting and stopping sidecar containers.



Summary: Container Patterns QCON 08.03.2016

How does a good building block look like?

And how would you assemble them?

Module container

1. Linux process
2. API
3. Descriptive
4. Disposable
5. Immutable
6. Self-contained
7. Small

Composite

- Sidecar
- Adapter
- Ambassador
- Chains



Dankeschön.

matthias@luebken.com

[@luebken](https://www.instagram.com/luebken)

Credits

- <https://www.flickr.com/photos/skynoir/8241460998> (Cover image)
- <https://www.flickr.com/photos/amlz/8664728590> (Lego)
- <https://www.flickr.com/photos/guidedbycthulhu/6810361241> (Socket)
- <https://www.flickr.com/photos/seektan/2074853585/> (Label)
- <https://www.flickr.com/photos/gullevek/2122873934> (Trash)
- <https://www.flickr.com/photos/grantmac/4852826923> (Lock)
- <https://www.flickr.com/photos/mhirano/13236048424> (Hand)