Transparent Data Driven API SAP-HANA db exporters

Dario Maiocchi
dmaiocchi@suse.com

SHAP-TEAM

Index:

Theory:

- Short excursion to Amazon AWS API Rationale

- An API for prometheus exporters for hanadb

Demos:

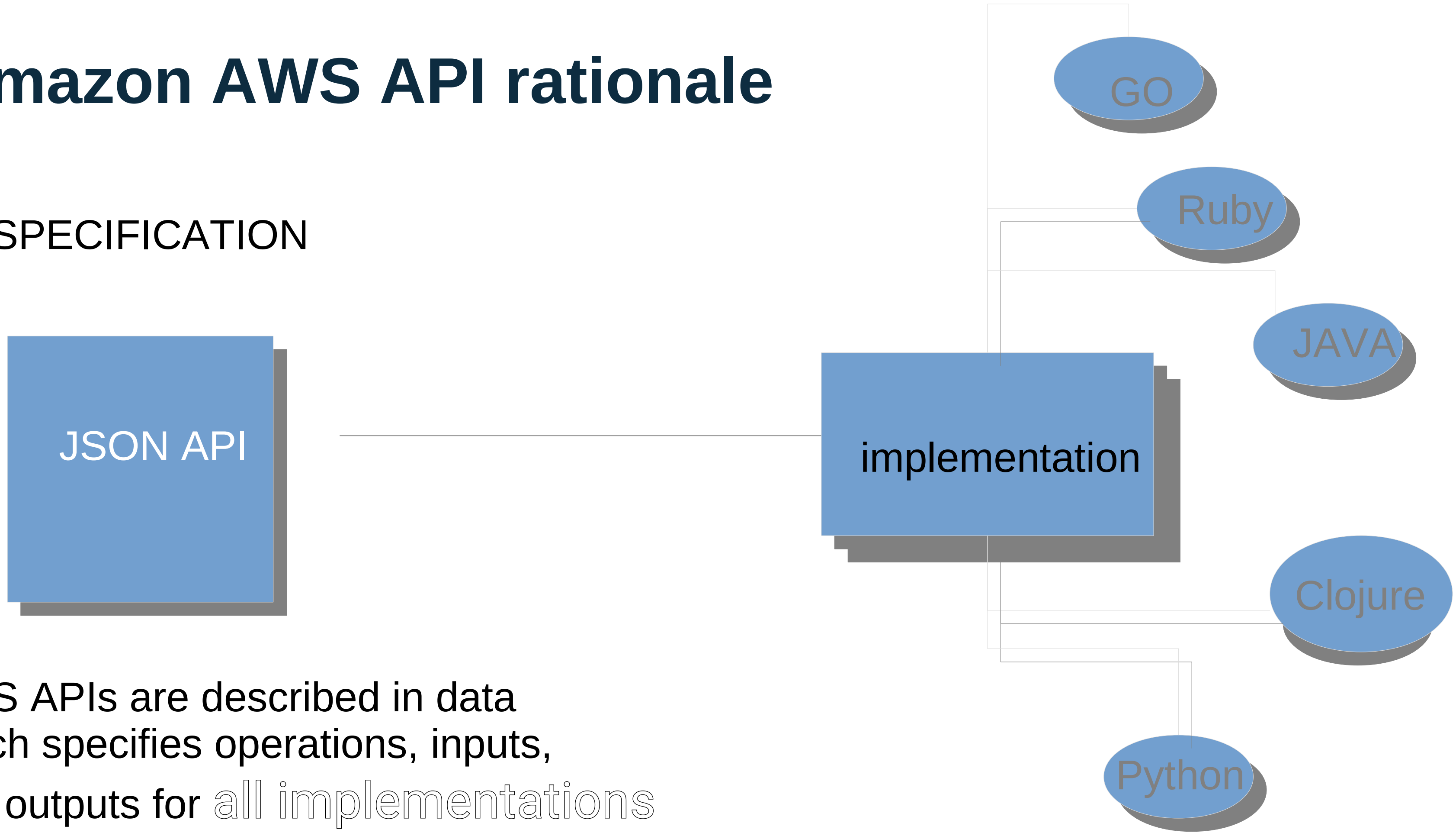
- Applications

WHAT IS AN API?

You will understand this image later on

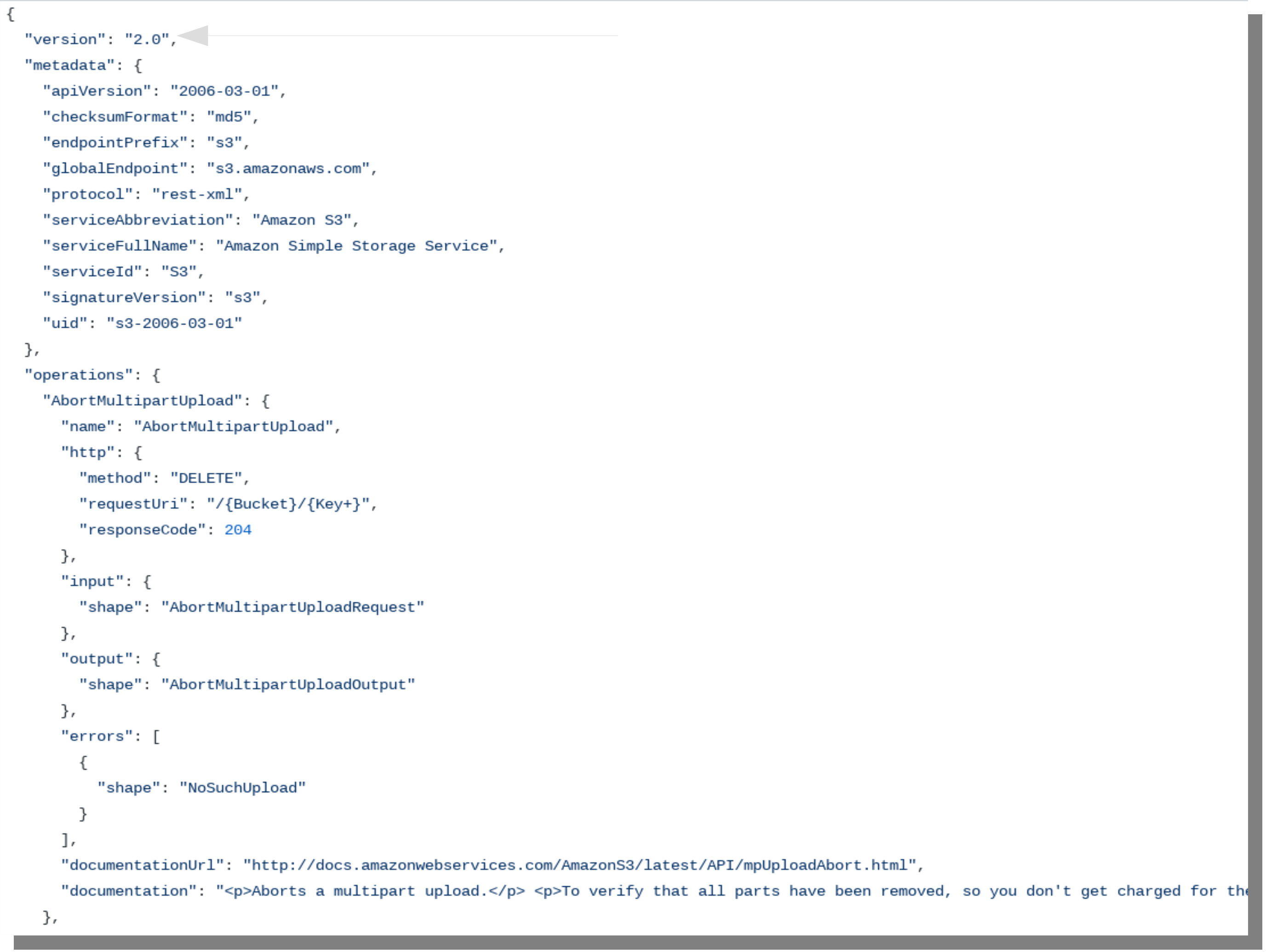
Amazon AWS API rationale

SPECIFICATION



AWS APIs are described in data
which specifies operations, inputs,
and outputs for all implementations

Specification with versioning



```
{
  "version": "2.0",
  "metadata": {
    "apiVersion": "2006-03-01",
    "checksumFormat": "md5",
    "endpointPrefix": "s3",
    "globalEndpoint": "s3.amazonaws.com",
    "protocol": "rest-xml",
    "serviceAbbreviation": "Amazon S3",
    "serviceFullName": "Amazon Simple Storage Service",
    "serviceId": "S3",
    "signatureVersion": "s3",
    "uid": "s3-2006-03-01"
  },
  "operations": {
    "AbortMultipartUpload": {
      "name": "AbortMultipartUpload",
      "http": {
        "method": "DELETE",
        "requestUri": "/{Bucket}/{Key+}",
        "responseCode": 204
      },
      "input": {
        "shape": "AbortMultipartUploadRequest"
      },
      "output": {
        "shape": "AbortMultipartUploadOutput"
      },
      "errors": [
        {
          "shape": "NoSuchUpload"
        }
      ],
      "documentationUrl": "http://docs.amazonwebservices.com/AmazonS3/latest/API/mpUploadAbort.html",
      "documentation": "<p>Aborts a multipart upload.</p> <p>To verify that all parts have been removed, so you don't get charged for the
    },
```

References:

* Interesting GitHub Projects:

https://github.com/weavejester/integrant

* Video talks:

- Cognitech Clojure aws-api https://www.youtube.com/watch?v=ppDtDP0Rntw

- Data Driven https://www.youtube.com/watch?v=zznwKCifC1A

* AWS JSON: API

https://github.com/aws/aws-sdk-js/blob/master/apis/s3-2006-03-01.normal.json

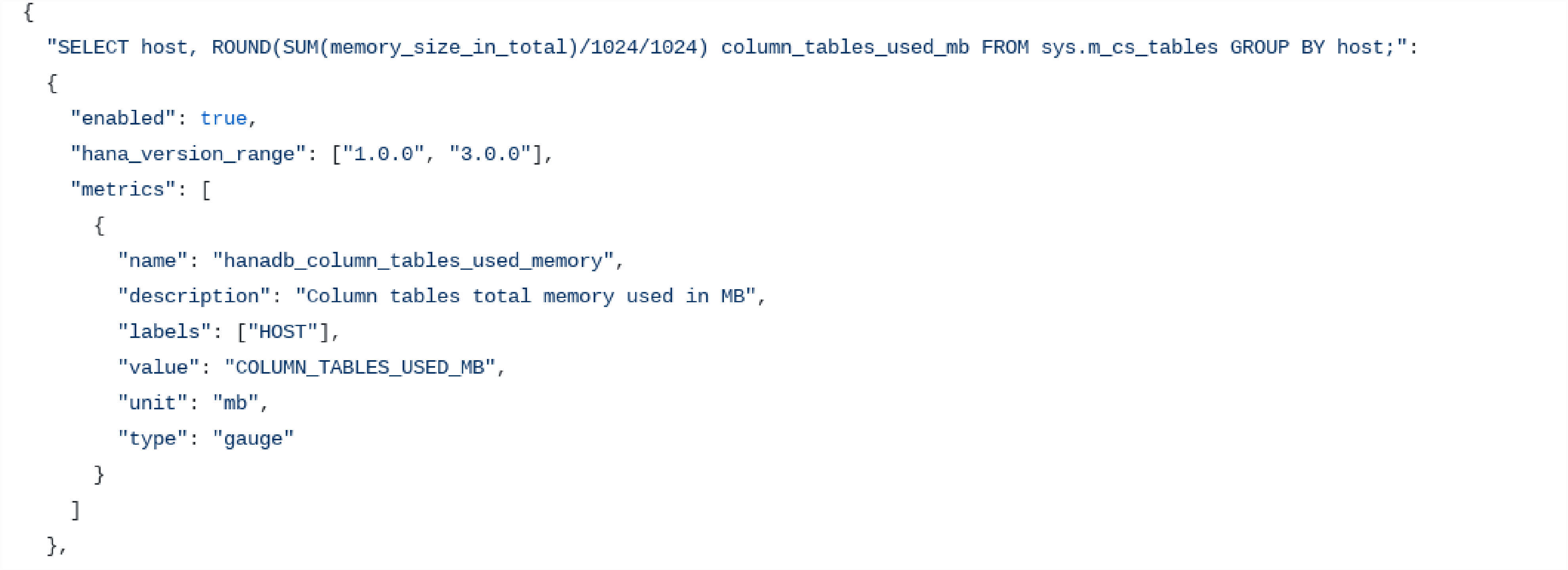
An API for prometheus exporters for hanadb

2 Differents entry-points:

- metrics.json (most important about metrics)

- config.json (configuration of exporter and database)

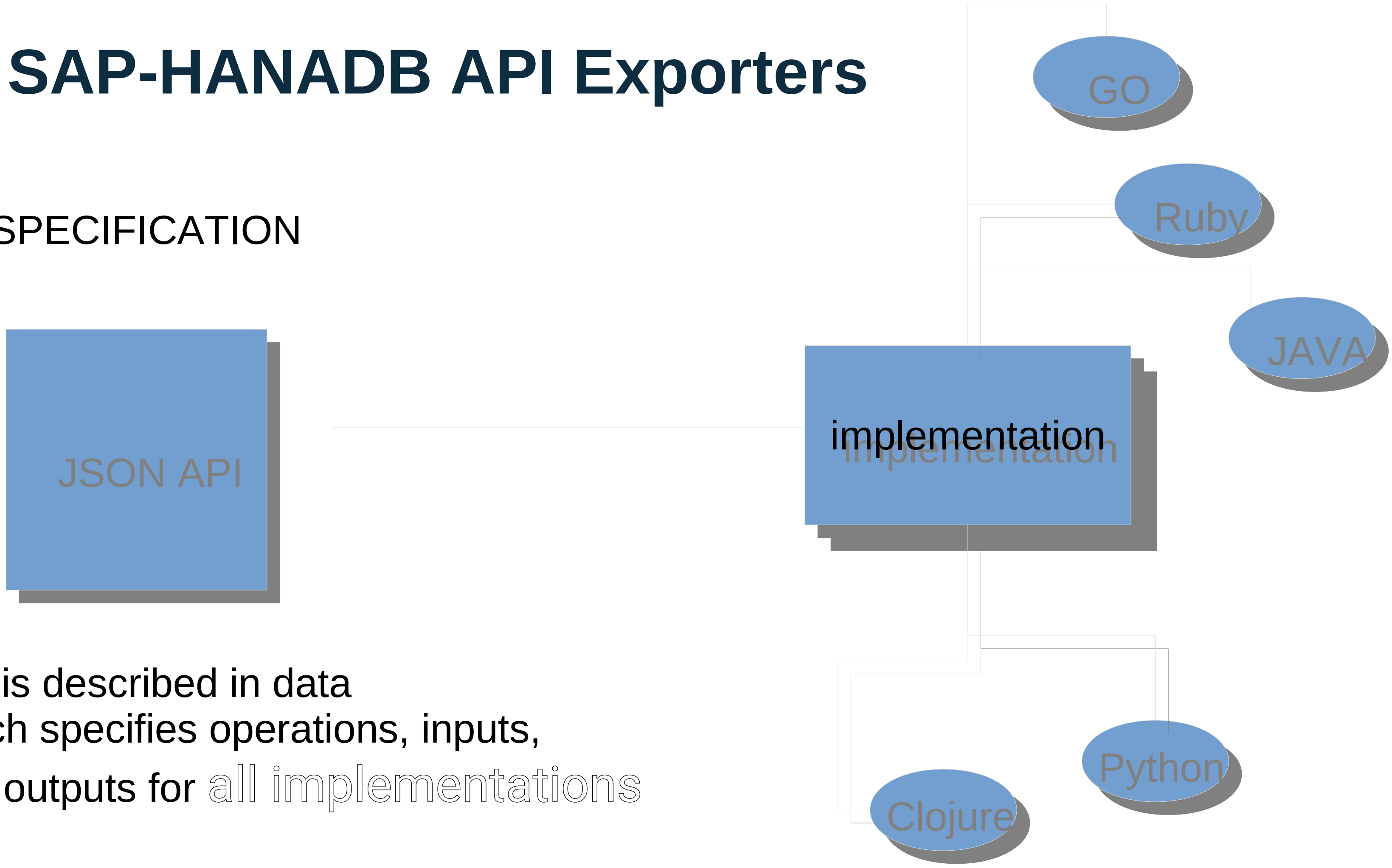
An API for prometheus exporters for hanadb



```
{
  "SELECT host, ROUND(SUM(memory_size_in_total)/1024/1024) column_tables_used_mb FROM sys.m_cs_tables GROUP BY host;":
  {
    "enabled": true,
    "hana_version_range": ["1.0.0", "3.0.0"],
    "metrics": [
      {
        "name": "hanadb_column_tables_used_memory",
        "description": "Column tables total memory used in MB",
        "labels": ["HOST"],
        "value": "COLUMN_TABLES_USED_MB",
        "unit": "mb",
        "type": "gauge"
      }
    ]
  },
```

SAP-HANADB API Exporters

SPECIFICATION



API is described in data
which specifies operations, inputs,
and outputs for all implementations

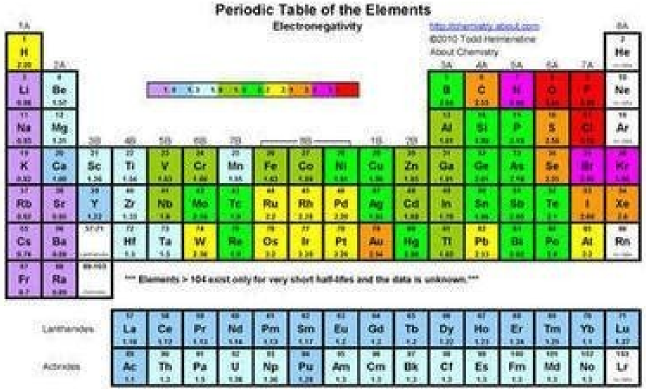
RATIONALE:

Having different exporter implementations, allow us to examining the current API , and examine an exporter from a different point of view. (from API and implementation pov)

GOOD NEWS:

Most of the current API can be implemented from other sap-hanadb exporters.

An Application:



Periodic Table of the Elements

Electronegativity

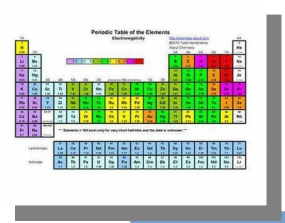
http://chemistry.about.com
©2010 Todd Helmenstine
About Chemistry

*** Elements > 104 exist only for very short half-lifes and the data is unknown.***

1A
2A
3B
4B
5B
6B
7B
8B
1B
2B
3A
4A
5A
6A
7A
8A
H
He
Li
Be
B
C
N
O
F
Ne
Na
Mg
Al
Si
P
S
Cl
Ar
K
Ca
Sc
Ti
V
Cr
Mn
Fe
Co
Ni
Cu
Zn
Ga
Ge
As
Se
Br
Kr
Rb
Sr
Y
Zr
Nb
Mo
Tc
Ru
Rh
Pd
Ag
Cd
In
Sn
Sb
Te
I
Xe
Cs
Ba
Hf
Ta
W
Re
Os
Ir
Pt
Au
Hg
Tl
Pb
Bi
Po
At
Rn
Fr
Ra
Lanthanides
La
Ce
Pr
Nd
Pm
Sm
Eu
Gd
Tb
Dy
Ho
Er
Tm
Yb
Lu
Actinides
Ac
Th
Pa
U
Np
Pu
Am
Cm
Bk
Cf
Es
Fm
Md
No
Lr

SAP-HANADB CLOJURE architecture

SPECIFICATION


Periodic Table of the Elements
Electronegativity

libraries/dependencies

Metrics.json

Config.json

Official Hana driver.jar

Clojure codebase

Prometheus clojure lib

metrics

A LISP REPL DEMO?!

Clojure (https://clojure.org/about/rationale) :

Ecosystem in HANA-SAP

+ Dependencies mgmt is advanced and professional

+ Due to JVM and jars there is no issue with multi-arch

+ Immutable data + first-class functions

+ concurrency by design (immutable data, no locking/mutex)

+ interactive language (REPL) = unique interactive programming exp

+ declarative code is more readable. Less code then imperative.

+ %50, less bug in immutable, data-driven functional languages.

References:

https://clojure.org/about/rationale

REPL vs rest of world:
https://docs.cider.mx/cider/usage/interactive_programming.html

Recommended videos:

GOTO 2018 • Functional Programming in 40 Minutes • Russ Olsen
https://www.youtube.com/watch?v=0if71HOyVjY

WHAT IS A REPL vs programming SHELLS:
https://www.youtube.com/watch?v=Qx0-pViyIDU