Software evolution in microservice artefacts

Josef Spillner <josef.spillner@zhaw.ch>
Service Prototyping Lab (blog.zhaw.ch/splab)

Sep 03, 2019 | INFORTE Summer School, Tampere

I.
Artefacts Primer

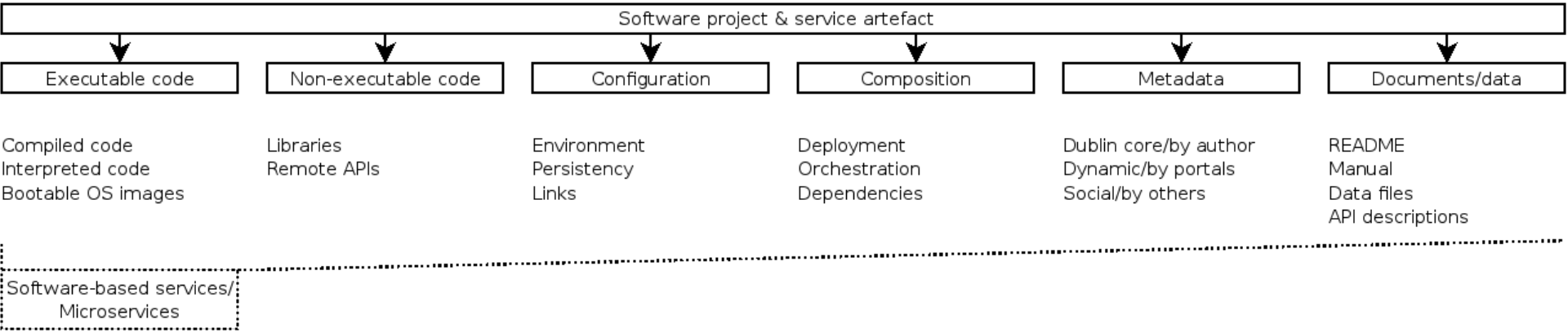
Software-as-a-Service

(Micro-)Service

- uniform description and invocation
  - description languages
  - messaging protocols & exchange patterns
- discoverable
  - registries
- encapsulation of implementation / separation of interface
  - software implementation is common (often polyglot)
    → SaaS, also: PaaS, FaaS, ... (→ subset of XaaS)
- composable/non-discriminating
- networked
- access models (free, subscription, contract, ...)

Software artefacts

- composite software - mixed-technology artefacts
- multiple file types often combined in artefact technologies
- shared via repositories/hubs/marketplaces



(often: multi-category artefacts, e.g. deployment for SaaS = composition + configuration + metadata)

Software code & non-code artefacts

Software applications/services := bundle of artefacts

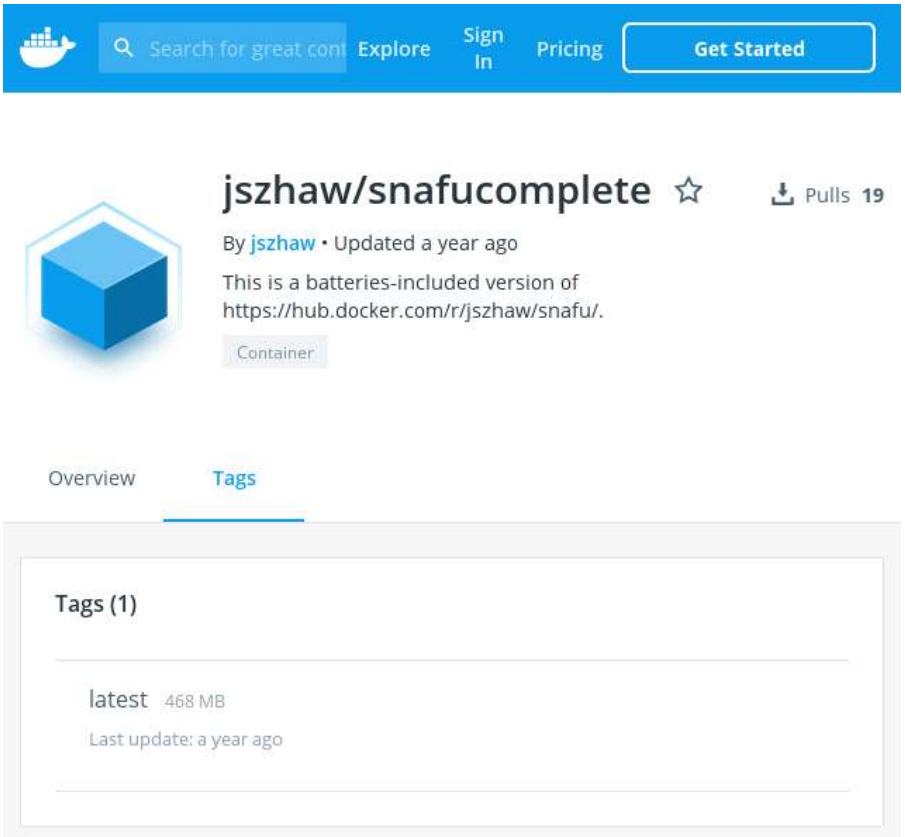
- typical formats: WAR, SAR, BPR, ZIP, ..., Helm charts

Artefact categories

- executable and non-executable code
  - *.class, *.jar, *.py(c), ...
- service descriptions
  - WSDL, WADL, WSML, GUIDD, USDL, RAML, Swagger/OpenAPI
- SLA templates
  - WSLA, WS-Agreement
- deployment descriptors
  - infrastructure: HOT, Cloudformation etc.
  - BPEL: PDD
  - servlets: web.xml
  - Kubernetes: YAML/JSON files, Helm templates
  - Lambda functions: SAM templates
  - ...

Metadata

Example: Docker Hub




```
{
  "name": "image_name",
  "data": {
    "1560176146": {
      "pull_count": 1721041870,
      "star_count": 6989,
      "is_migrated": false,
      "is_automated": false,
      "number_of_tags": 278,
      "full_size": 35224772,
      "os": "linux",
      "architecture": "amd64",
      "latest": true
    }
  }
}
```


→ access: API (results limit), web crawling (hits frequency limit)

Metadata

Typical content:

- “Dublin Core metadata“ for authored works
  - title, creator, subject, description, publisher, contributor, ..., rights
- developers/maintainers with contact information
- version specification
- dependencies
- social: ratings, comments
- auto-generated: downloads

Example: Helm‘s Chart.yml

```
apiVersion: v1
name: ExampleChart
version: 1.0
kubeVersion: 1.7a
description: This is just an example chart.
keywords:
  - example
  - nonsense
home: http://inforte.jyu.fi/events/SW_evolution
sources:
  - https://github.com/serviceprototypinglab
maintainers:
  - name: Josef Spillner
    email: josef.spillner@zhaw.ch
    url: https://blog.zhaw.ch/splab/
engine: gotpl
icon: icons/myicon.png
appVersion: 2.0alpha2
deprecated: true
tillerVersion: The version >2.0.0
```

~~dependencies:~~

inferred from present subdirectories charts/*

Code

Programming language
- Java, Python, Go, JavaScript, Solidity, ...

Language version and (runtime) flavour
- Java 6 vs. Java 9, CPython vs. PyPy

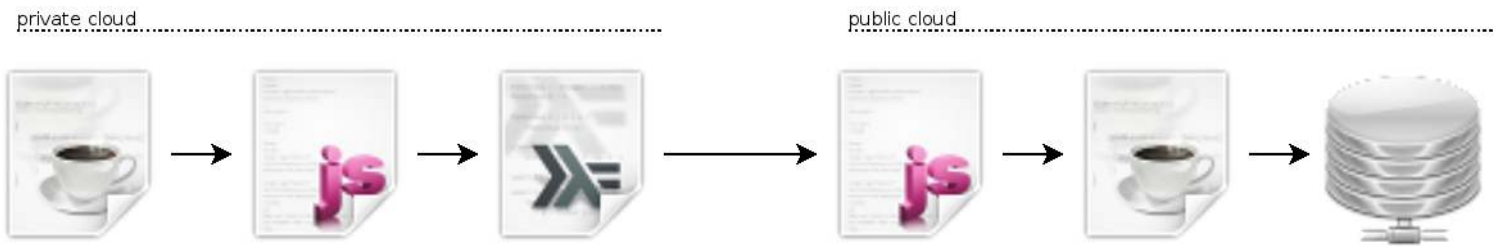
Representation
- source code, byte code, compiled code
- flat file, archive file, locator
- include paths

Context
- completeness, up-to-dateness, genuineness
- degree of deviation
- automated build instructions
- relationships to other code units

```
// Global (instance-wide) scope; this
// computation runs at instance cold-start
const instanceVar = heavyComputation();

/**
 * HTTP Cloud Function declaring a variable.
 *
 * @param {Object} req
   Cloud Function request context.
 * @param {Object} res
   Cloud Function response context.
 */
exports.scopeDemo = (req, res) => {
  // Per-function scope; this
  // computation runs every time
  // this function is called
  const functionVar = lightComputation();

  res.send(`Per instance: ${instanceVar},
    per function: ${functionVar}`);
};
```



Declarative deployment descriptors

Deployment: code (scripts) vs. declarative descriptor

Typical content:

- name within the target system
- runtime limits
- access control
- logging setup
- network setup: port numbers, throttling, ...

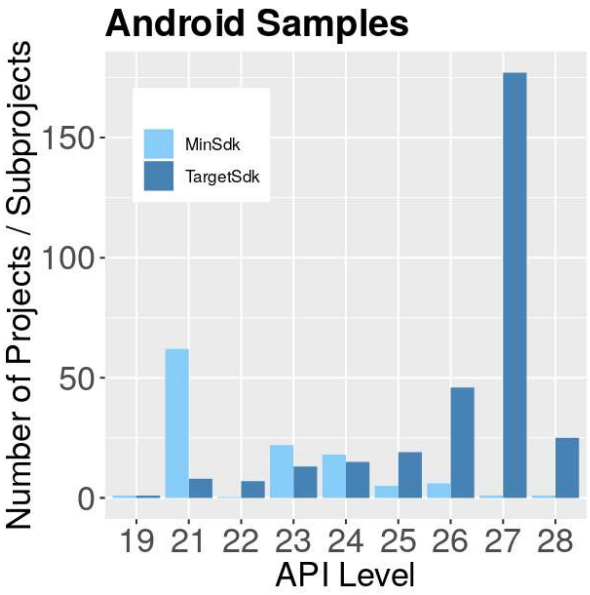
Example: Helm‘s deployment template for Joomla
(not fully declarative;
some metadata present)

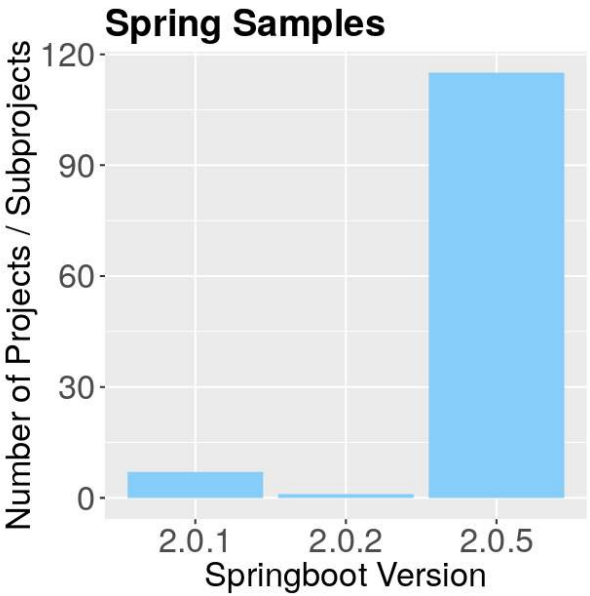
```
{{- if or .Values.mariadb.enabled .... -}}
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: {{ template "joomla.fullname" . }}
  labels:
    app: {{ template "joomla.fullname" . }}
    chart: {{ template "joomla.chart" . }}
    release: {{ .Release.Name | quote }}
    heritage: {{ .Release.Service | quote }}
spec:
  selector:
    matchLabels:
      app: {{ template "joomla.fullname" . }}
      release: "{{ .Release.Name }}"
  replicas: 1
  template:
    metadata:
      labels:
        app: {{ template "joomla.fullname" . }}
    spec:
      {{- if .Values.image.pullSecrets }}
      imagePullSecrets:
      {{- range .Values.image.pullSecrets }}
        - name: {{ . }}
      {{- end}}
      {{- end }}
      containers:                                        .........
```

Software artefacts evolution

Implicit hypothesis: Correlation with software evolution

Lifecycle
- branches & releases
- numbering schemes
- change patterns over time and/or over versions



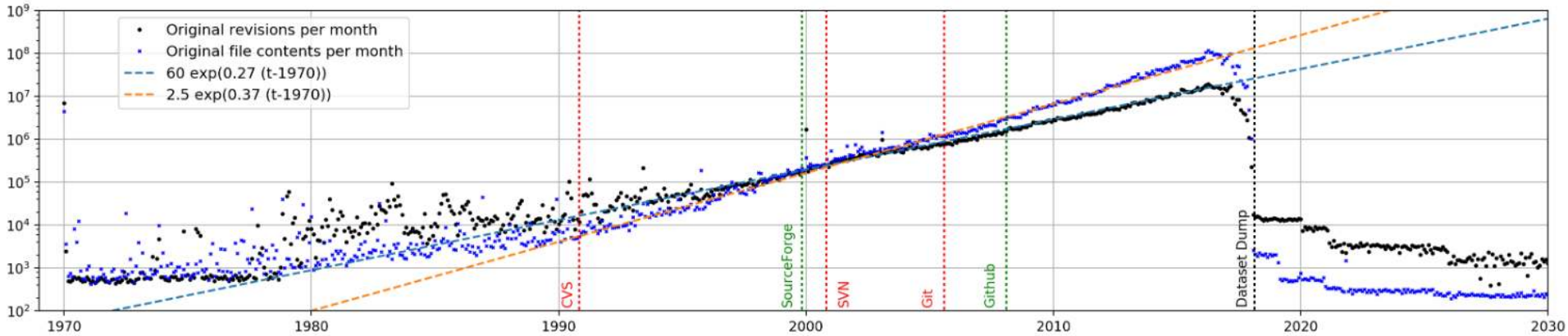


G. Menezes, B. Cafeo, A. Hora: Framework Code Samples: How Are They Maintained and Used by Developers? arXiv:1907.05564

Code vs. artefact considerations

Software Heritage study - 4 billion code file artefacts, 1 billion commits

- exponential growth rate
- missing link to artefacts, but strong link to repository technologies
  - higher-level features & release/duplication efforts
  - data point: since SourceForge inception,
    - original revisions double every ~30mo
    - original code files double every ~22mo
  - data point: 70% of code on GitHub → file-level exact clones

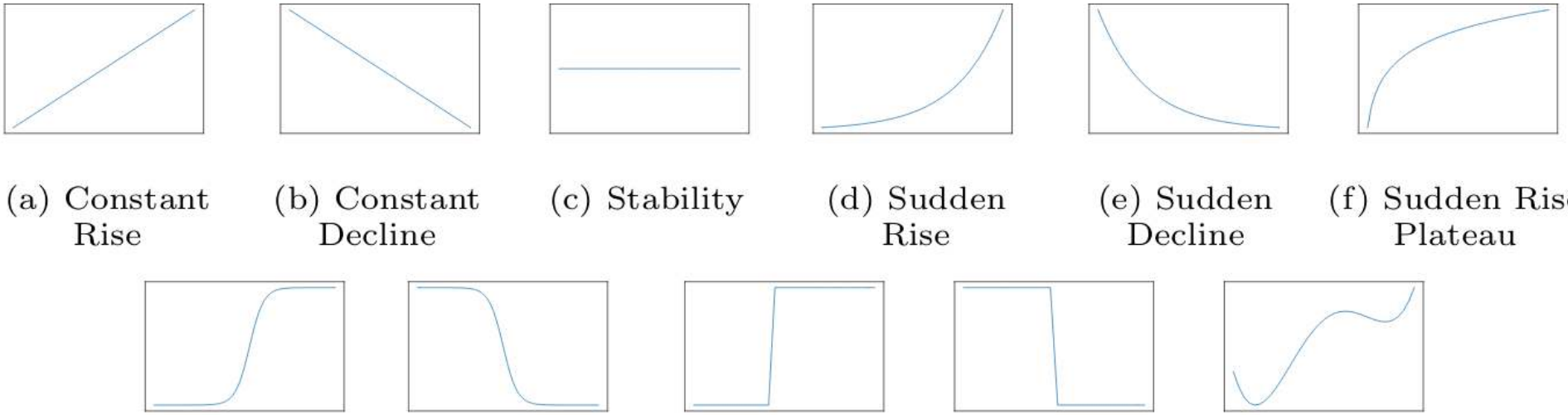


G. Rousseau, R. Di Cosmo, S. Zacchiroli: Growth and Duplication of Public Source Code over Time: Provenance Tracking at Scale. arXiv:1906.08076

Software evolution patterns

(per metric)

(unsuitable names)



(a) Constant Rise (b) Constant Decline (c) Stability (d) Sudden Rise (e) Sudden Decline (f) Sudden Rise Plateau

(g) Plateau Gradual Rise (h) Plateau Gradual Decline (i) Plateau Sudden Rise (j) Plateau Sudden Decline (k) Instability

B.G. Mateus, M. Martinez: On the adoption, usage and evolution of Kotlin Features on Android development. arXiv:1907.09003

II.
Practical Examples

Excursus: Docker Compose files

Compose: Tool for defining and running multi-container applications.

- services, networks, volumes

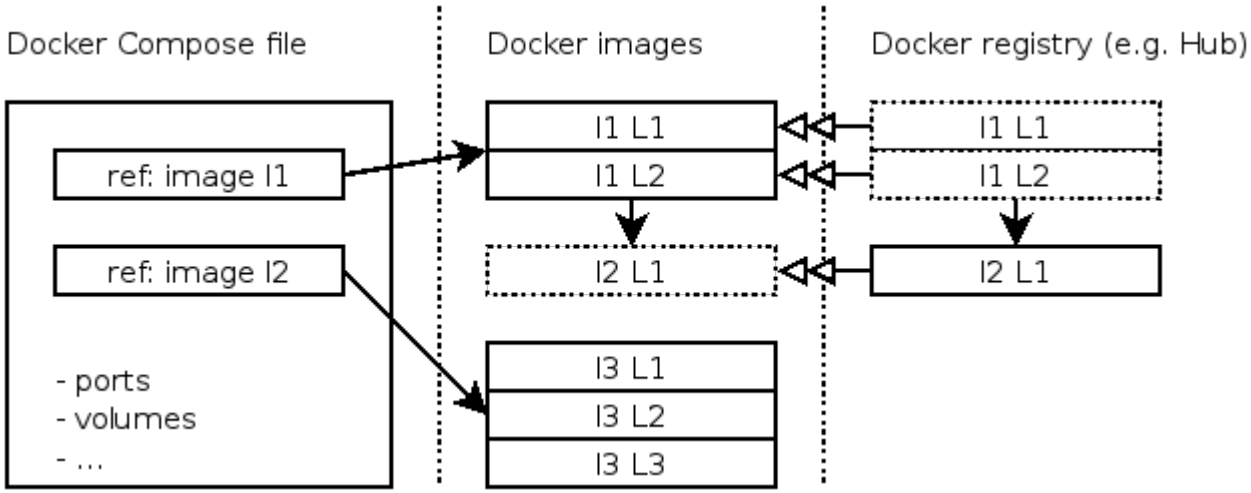
Version 1 - Nov 2015
Version 2 - Feb 2016
Version 3 - Jan 2017

Quality checks:
containers (external tool): *hadolint*, compositions: *none*

Repositories:
only generic code hosting, no dedicated hub site

Docker Compose in detail

Single compose file



Docker Compose artefact quality

Duplication

- common in copy&paste-style development
- duplication vs. unique assignments
  - service name
  - container name
  - image
  - port

Invalid specifications

- common when changing environments, e.g. dev-prod or dev1-dev2
- path references and remote locators
  - volume directory
  - image/version in non-standard registry

```yaml
version: '3'
services:
  nginx:
    container_name: some-nginx
    image: nginx:1.13
    restart: always
    ports:
    - 3306:80
    - 443:443
    expose:
     - '80'
    volumes:
    - ./nginx/conf.d:/etc/nginx/conf.d
    depends_on:
    - app

  nginx:
    container_name: some-mysql
    image: mysql/mysql-server:5.7
    environment:
     MYSQL_DATABASE: test
     MYSQL_ROOT_PASSWORD: hellokoding
     MYSQL_ROOT_HOST: '%'
    ports:
    - "3306:3306"
    restart: always
```

Excursus: Helm apps & charts

Helm: Package manager for the cloud-native landscape

Created in 2015 as a Deis (now MS) installer
Out of K8s incubator in 2017

80 stable charts, 53 maints

KubeApps UI/marketplace (KubeApps Hub) in 2017
Helm summit in February 2018 with 179 participants (out of 250)
CNCF top-level (incubator) project since May 2018

177 stable charts, 142 maints
50k montly downloads
>30k charts on Github/<u>similar?</u>

Quality checks (v2.8): *helm lint*

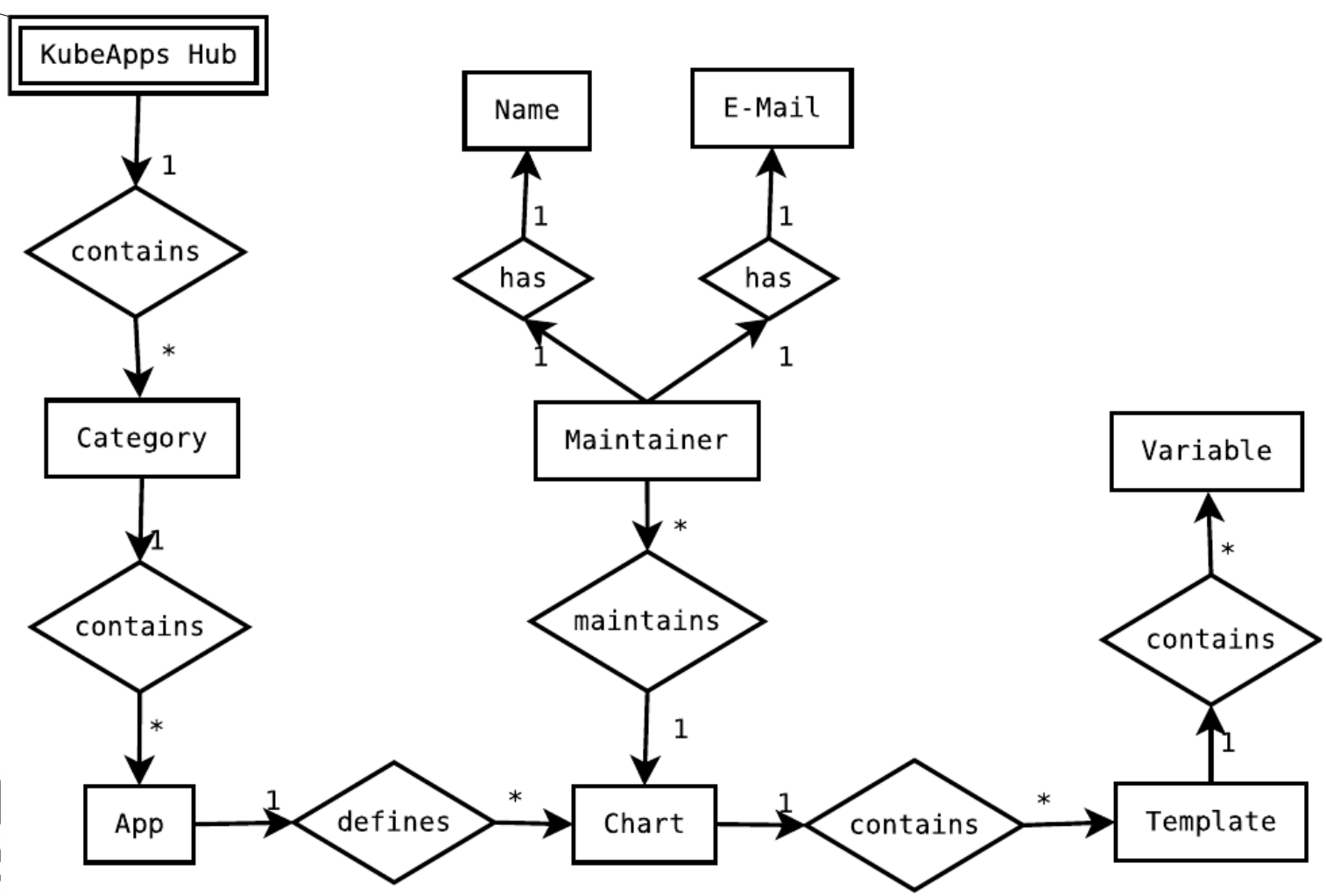
»This command takes a path to a chart and runs a series of tests to verify that the chart is well-formed.«

<u>sufficient?</u>

Now: Helm Hub & Vand.io, >700 total charts (all branches), ca. 300 stable - e.g.: helm install stable/drupal to get CMS

Helm in detail: entities & relationships

similar: Helm Hub, Vand.io, ...

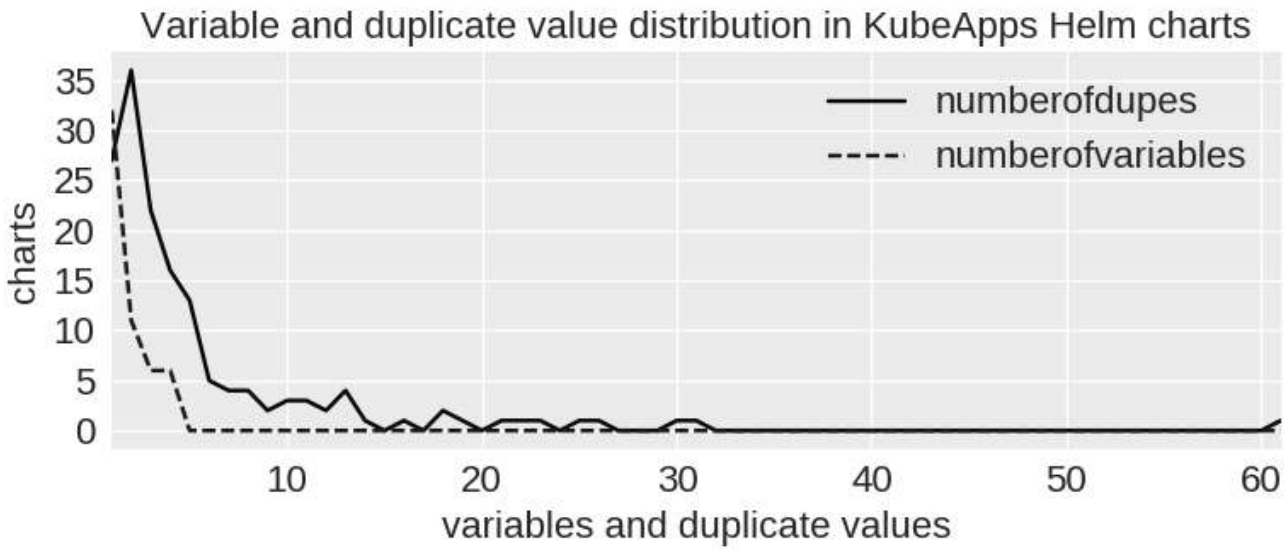


Helm artefact quality

Equivalence to code smells? Anti-patterns? Non-adherence to best practices?

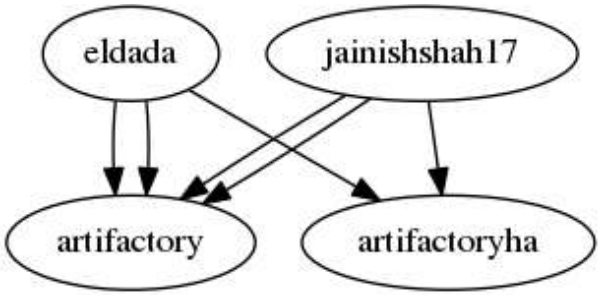
Concrete issues with Helm charts:

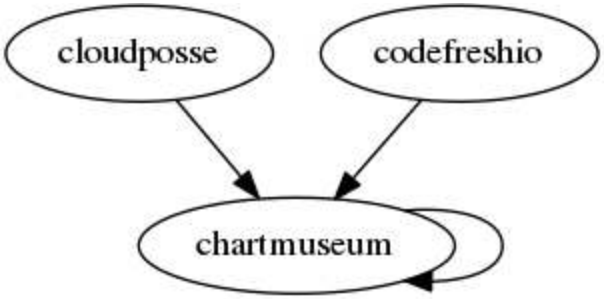
- multiple versions in one repository category (e.g., “stable“)
- equivalence of chart name and maintainer name
- duplicate values not expressed as variables
- (dependencies: transitive issues, error propagation & aggravation)

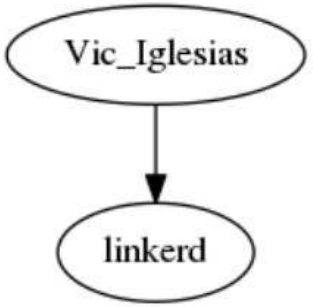


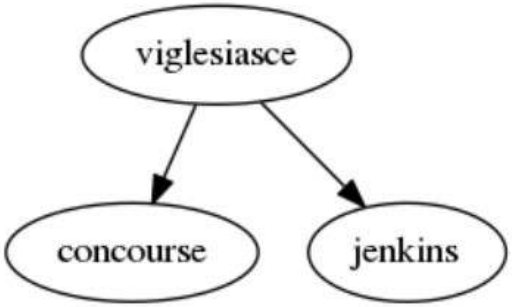
Helm artefact quality

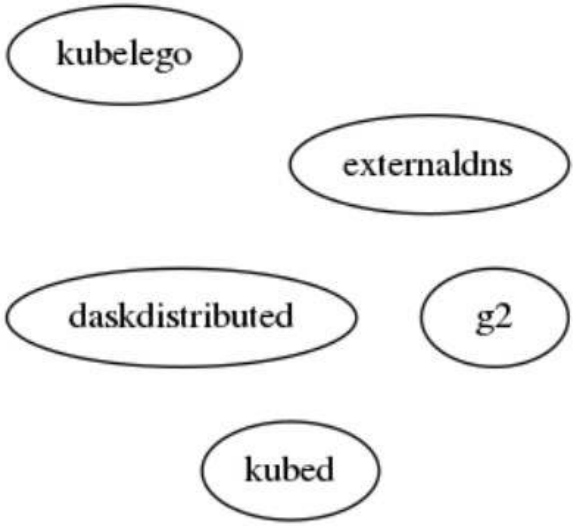
Invalid maintainer→chart relationship graph (DAG) identification











Helm artefact quality

Effect of mandatory (albeit low-barrier) quality gates

Chart-related KubeApps Hub metrics

Chart metrics	Affected	Percentage	Base metric
Charts	177	100.00	–
Irregularity: no maintainer	10	5.65	Charts
Irregularity: name collision	1	0.56	Charts
Irregularity: multiple versions	5	2.82	Charts

Chart-related GitHub metrics

Chart metrics	Affected	Percentage	Base metric
Charts	115	100.00	–
Irregularity: no maintainer	95	82.61	Charts
Irregularity: name collision	2	1.74	Charts
Irregularity: multiple versions	0	0.00	Charts

*without fork analysis

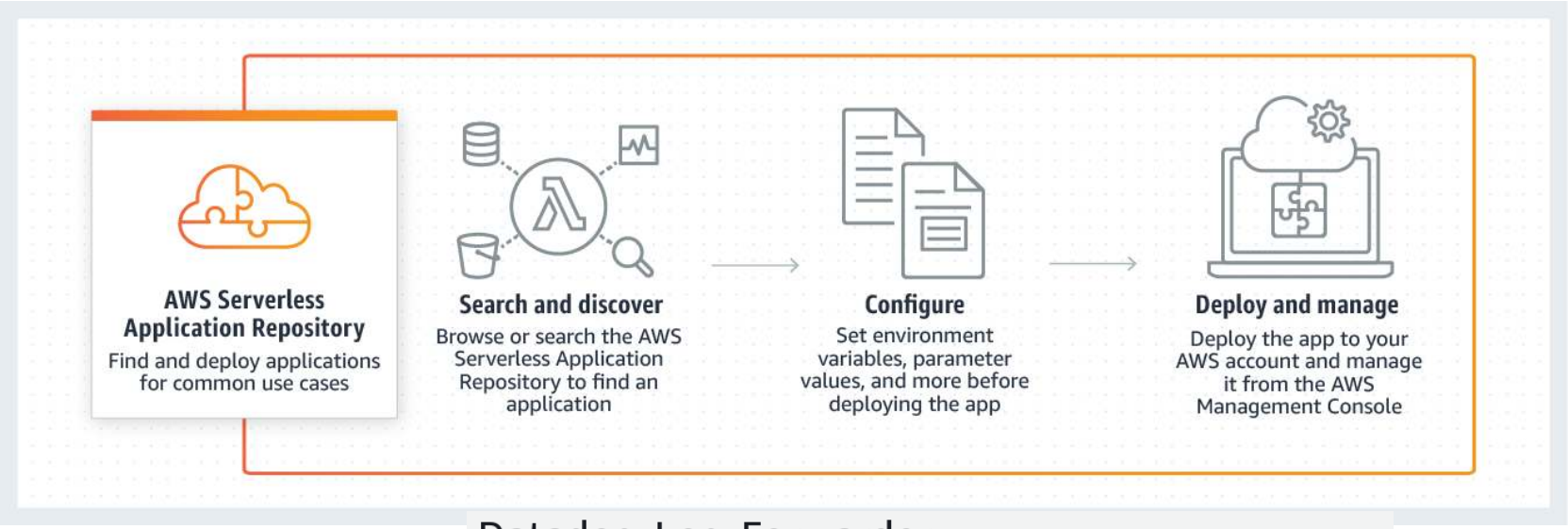
Excursus: AWS Lambda & SAM

Lambda: Cloud functions running as short-lived containers

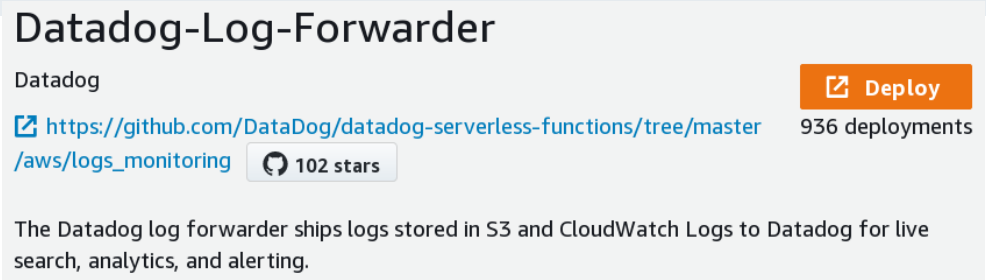

```
def lambda_handler(event, context):
        '''event: dict
        context: meta information obj
        returns: dict, string, int, ...'''
        # ...
        return "result"
```

SAM: Serverless Application Model

- vendor-specific cloud function bundle technology

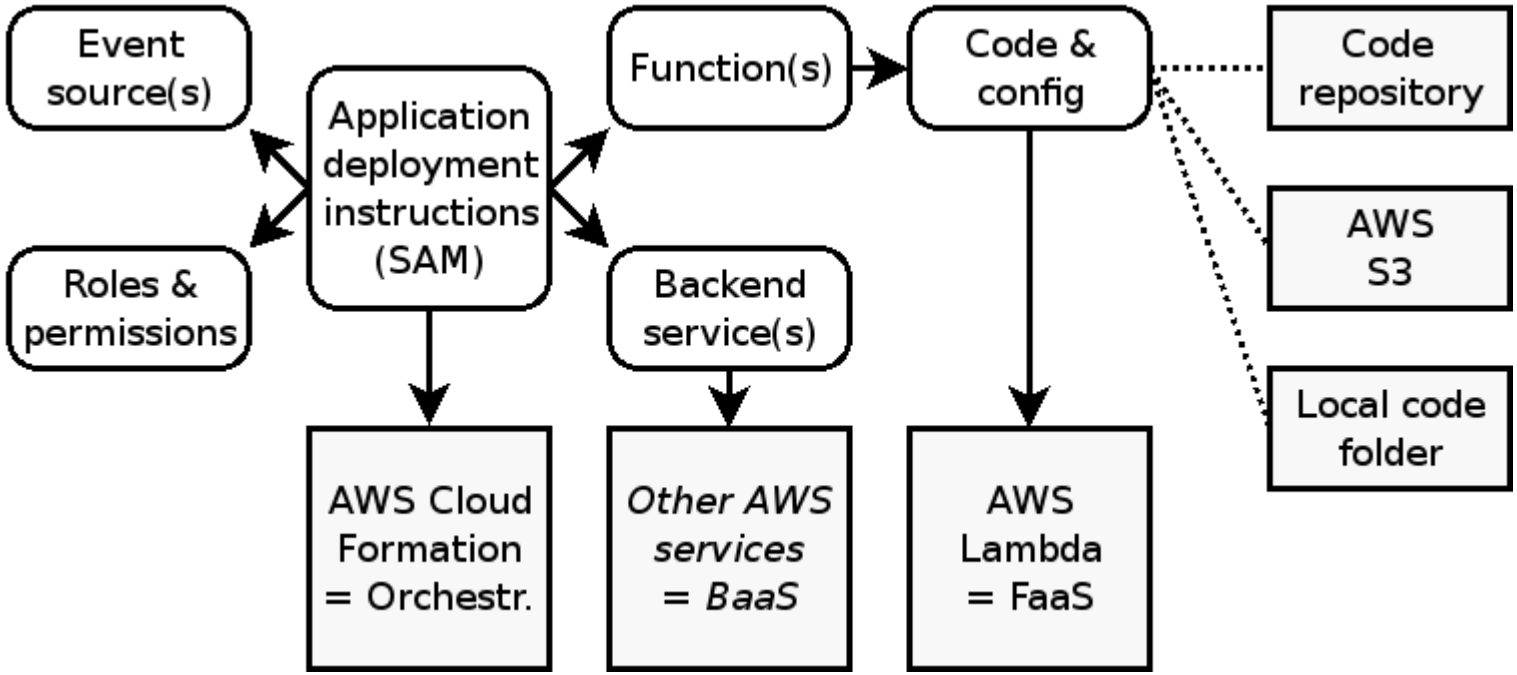


- public repository available



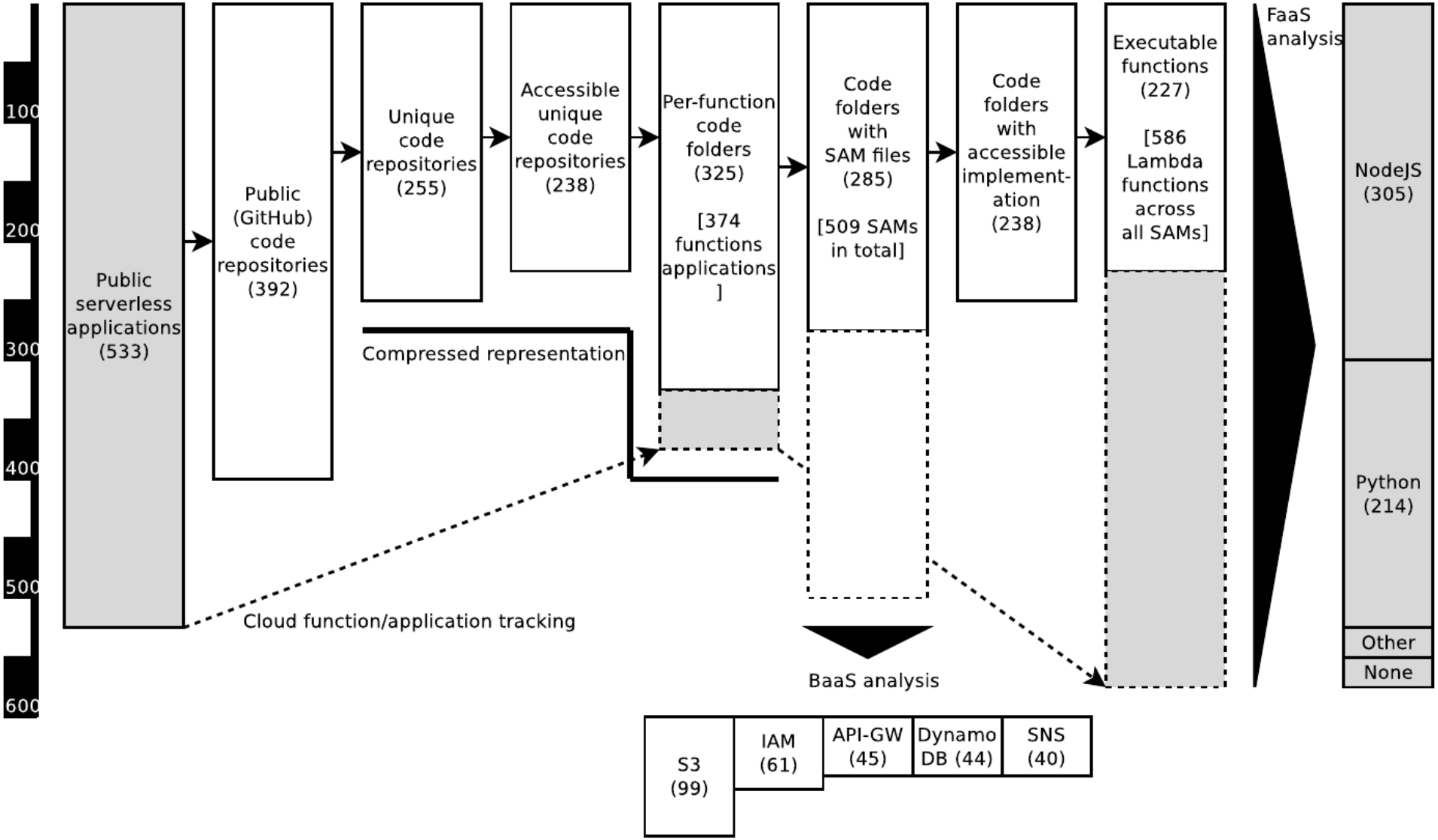
Lambda/SAM in detail: topology

AWS-defined artefact topology



Lambda/SAM artefact quality

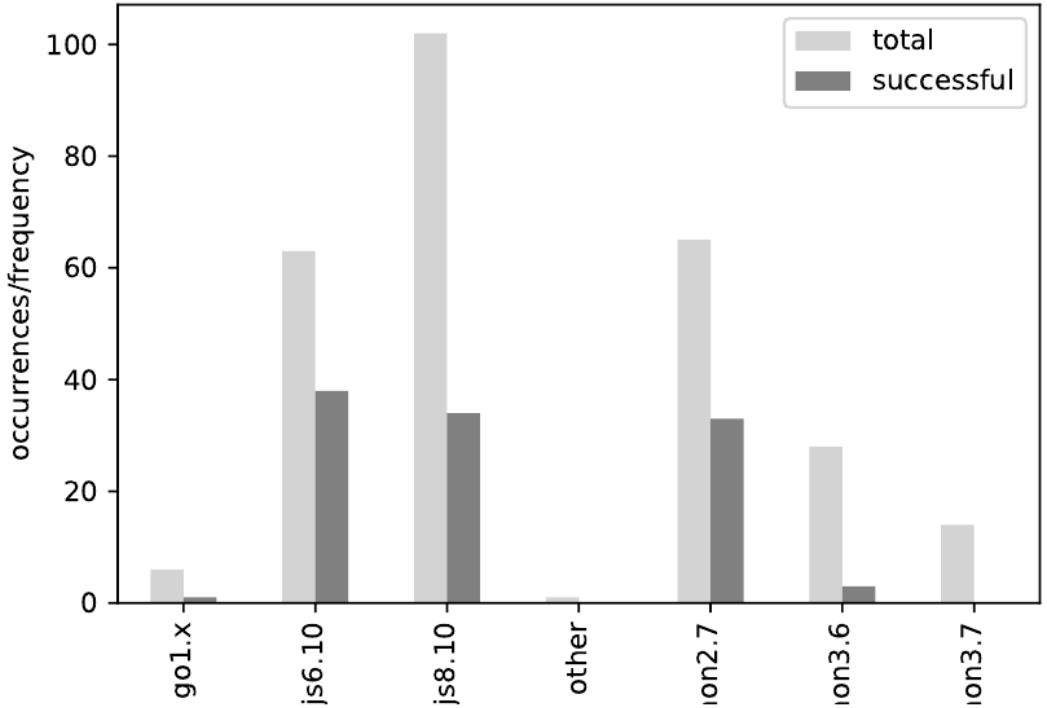
Drill-down approach: Issues at SAR level



Lambda/SAM artefact quality

Generic execution measurements

- caveat: incomplete test event generation + semantics

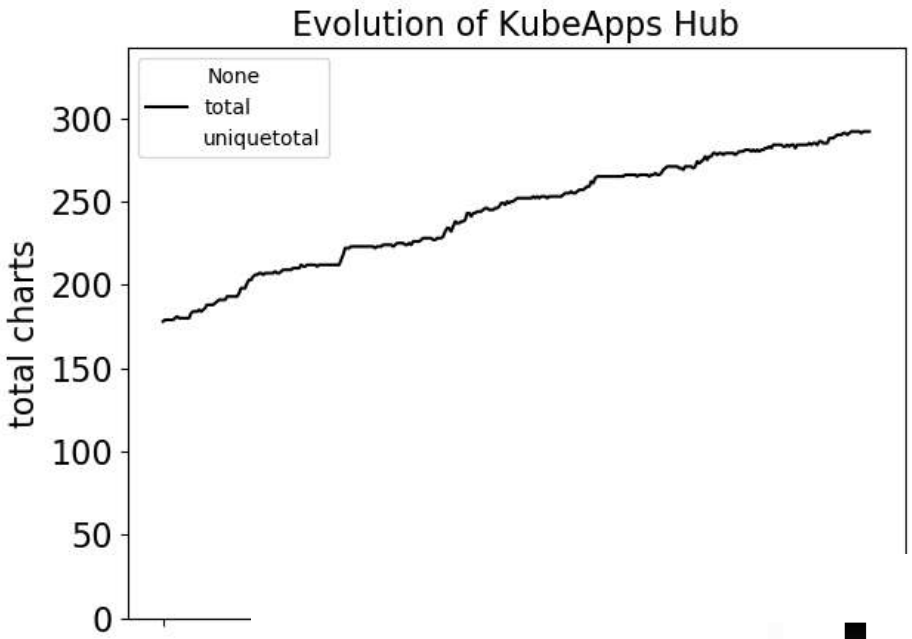


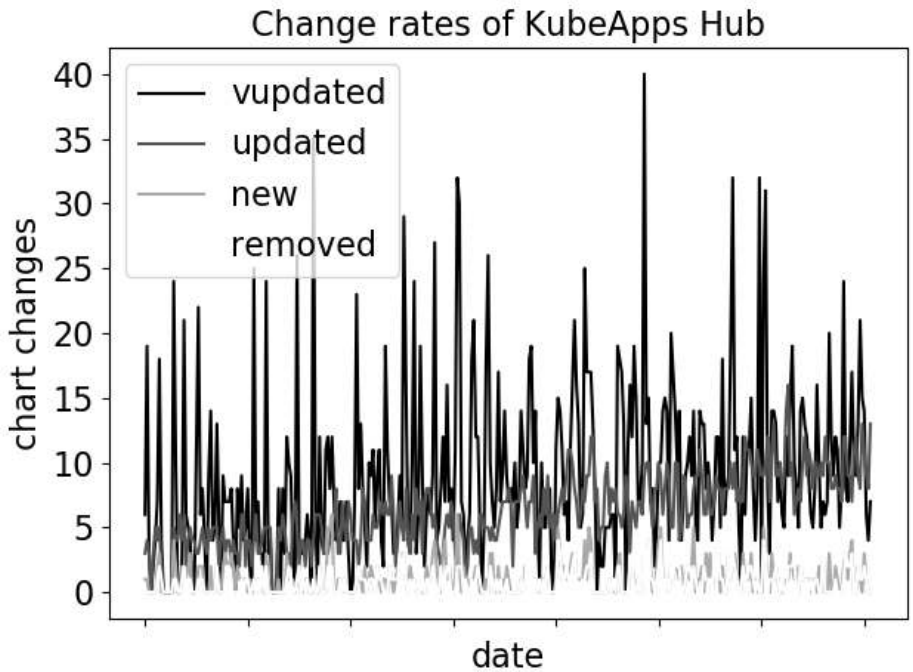
category	execution time (ms)	billed time (ms)	memory size (MB)	memory used (MB)
successful	78.36	174.23	131.30	22.86
failed	131.31	213.81	295.32	27.18
overall	106.76	195.46	301.92	26.80

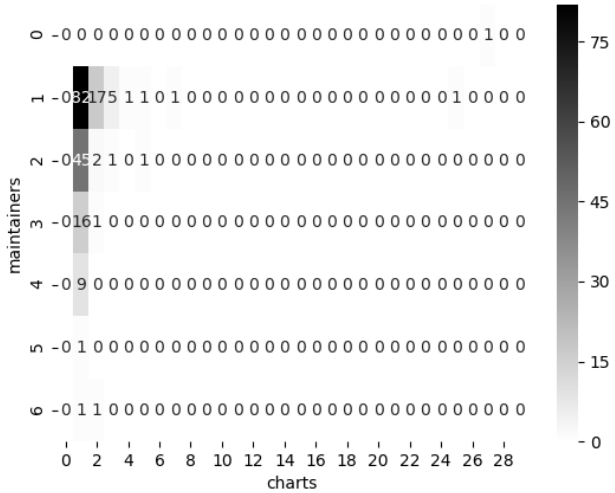
III.
Evolution Observation

Helm evolution

Helm charts (May‘18-May‘19):

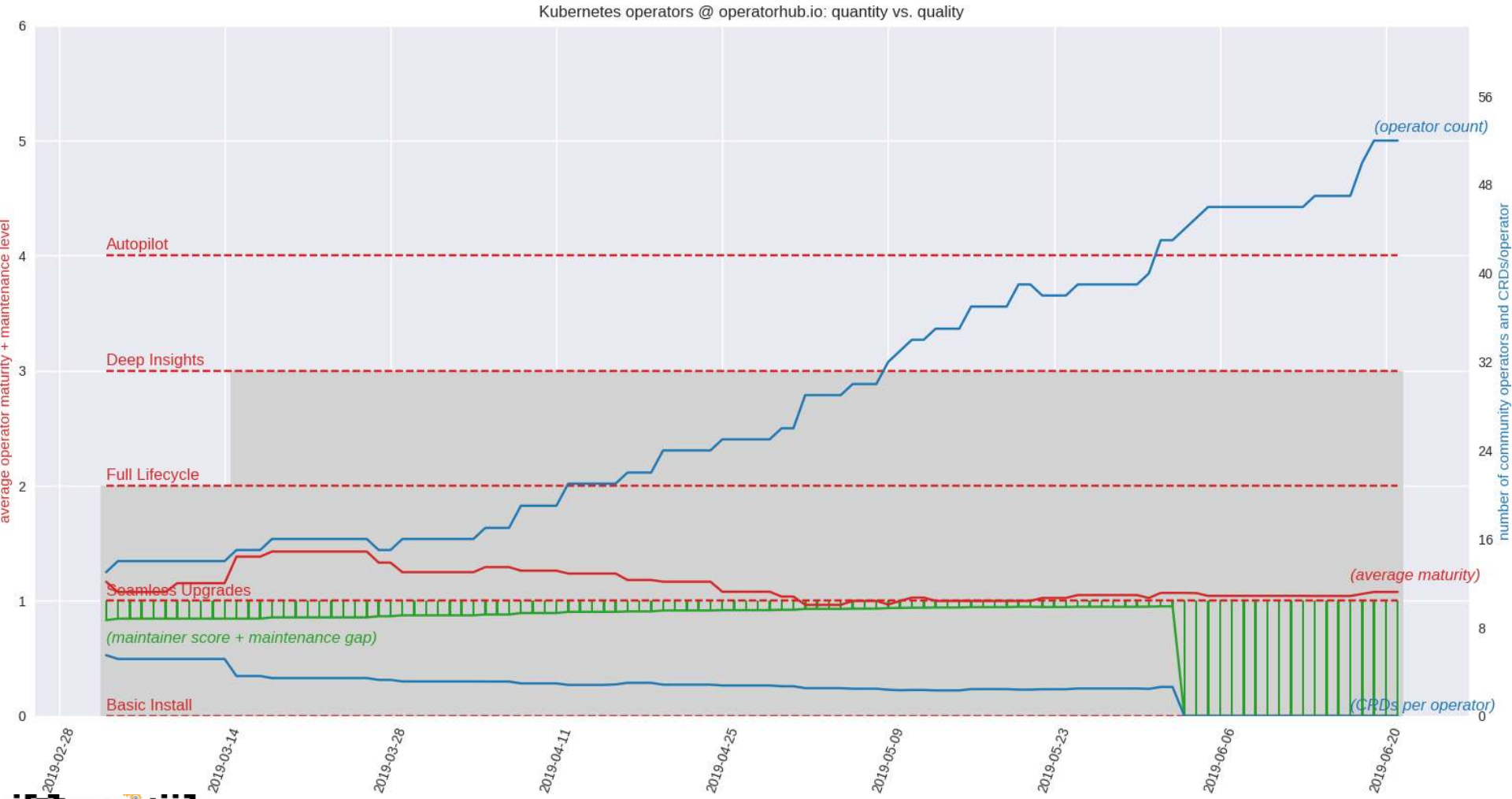






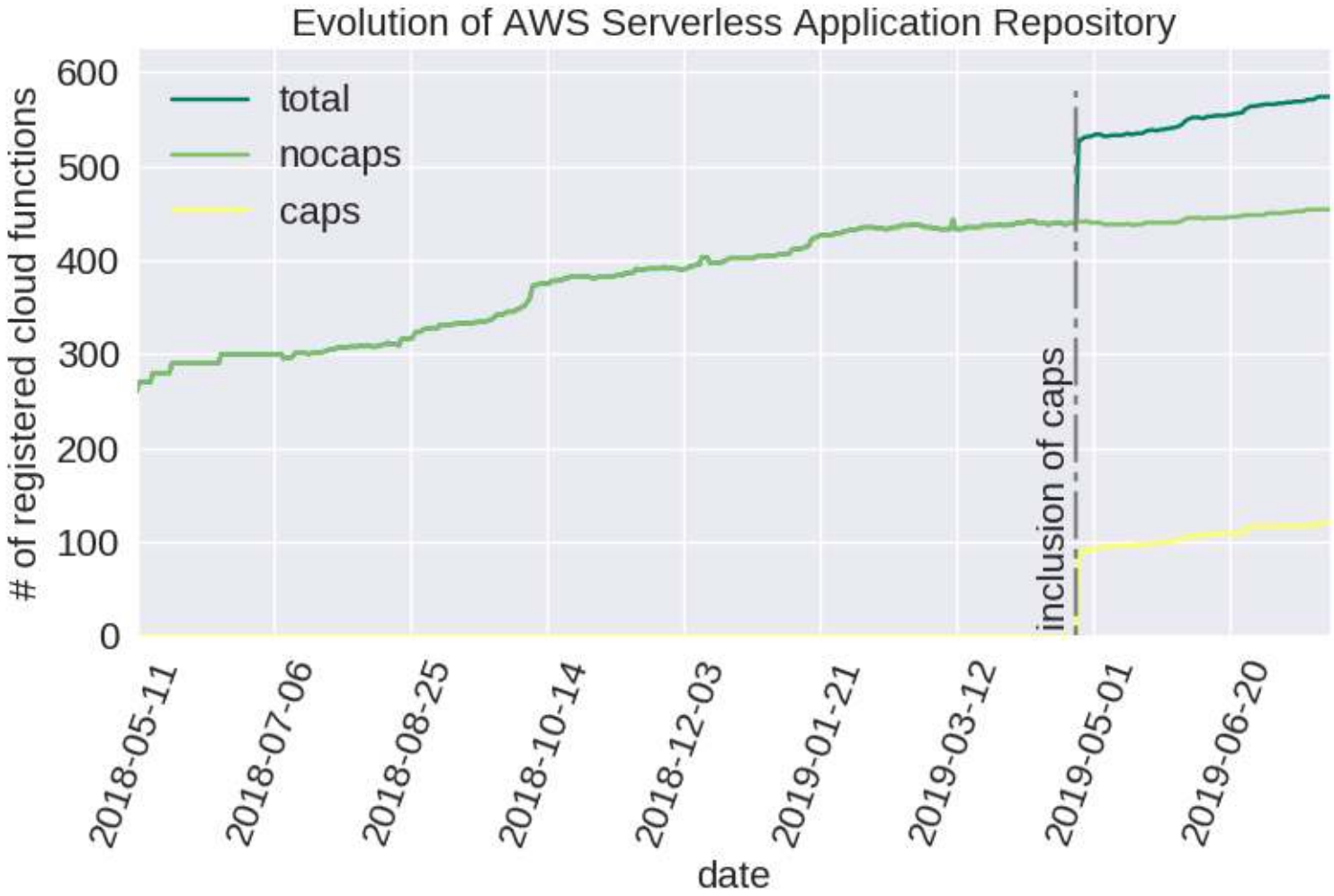
Kubernetes operators evolution

Operators (Feb‘19-Jun‘19):



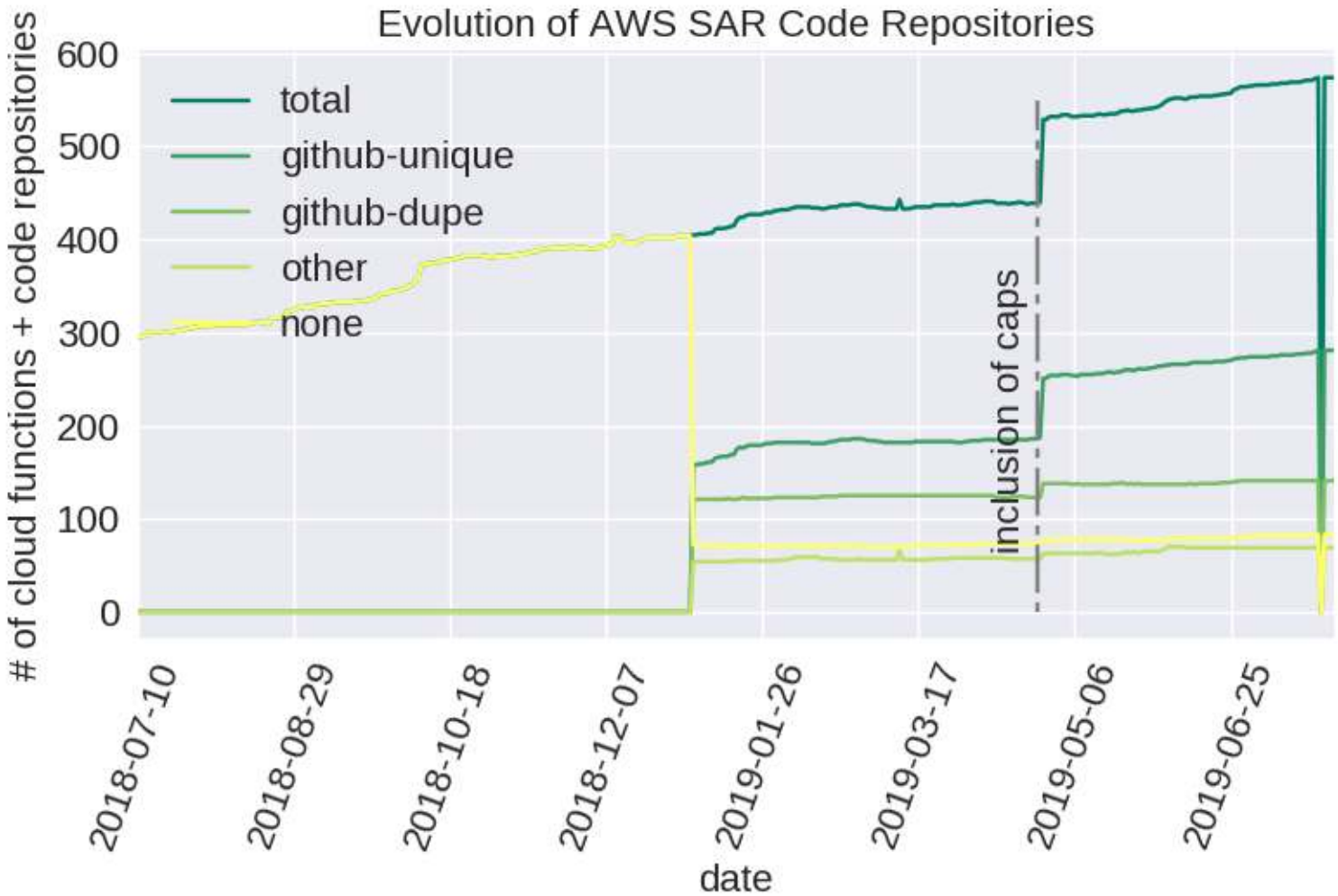
Lambda/SAM evolution

SAMs at AWS SAR (May‘19-Jul‘19)



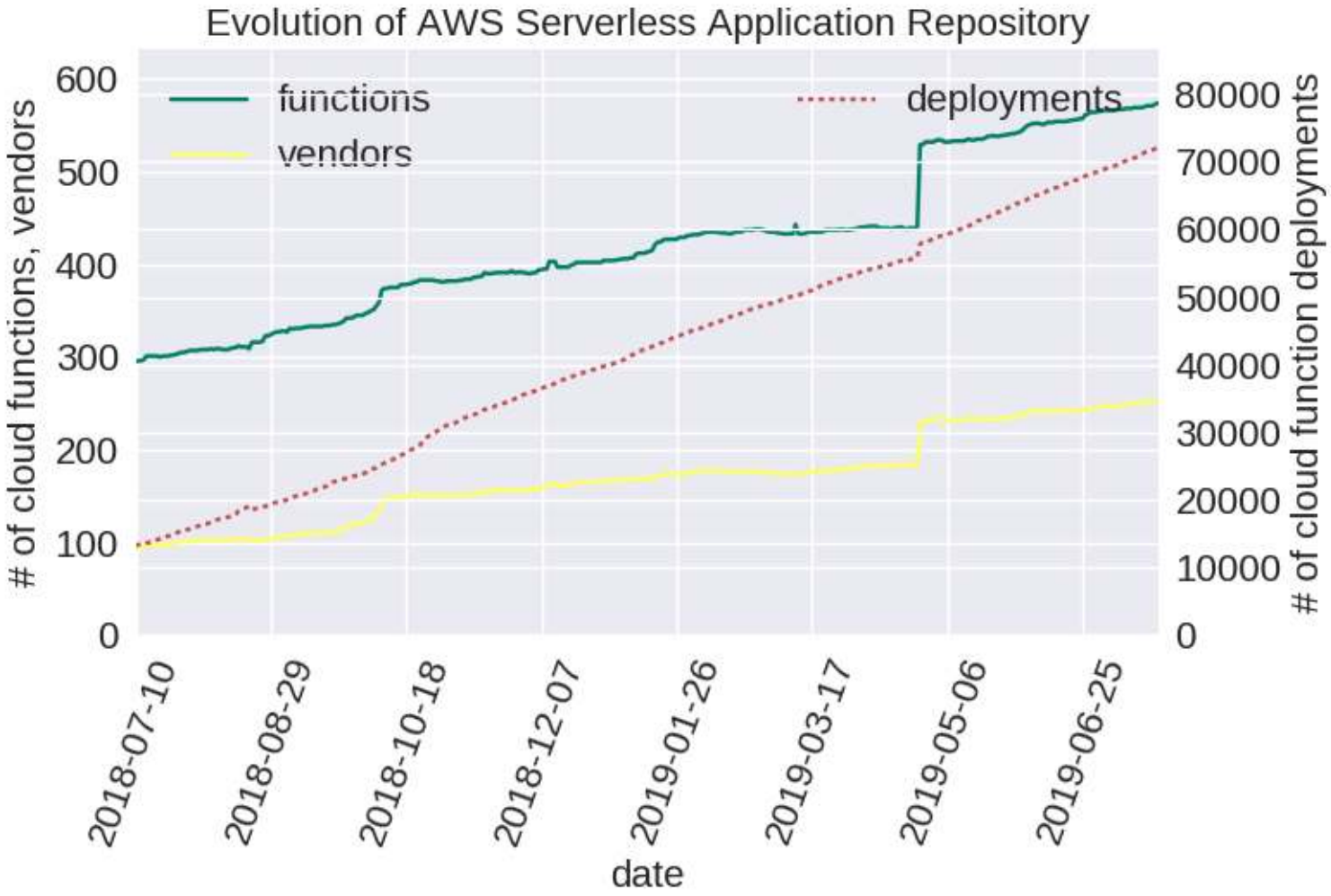
Lambda/SAM evolution

SAM-referenced code repositories at AWS SAR (Jul‘19-Jul‘19)



Lambda/SAM evolution

Vendors and deployments at AWS SAR (Jul‘19-Jul‘19)

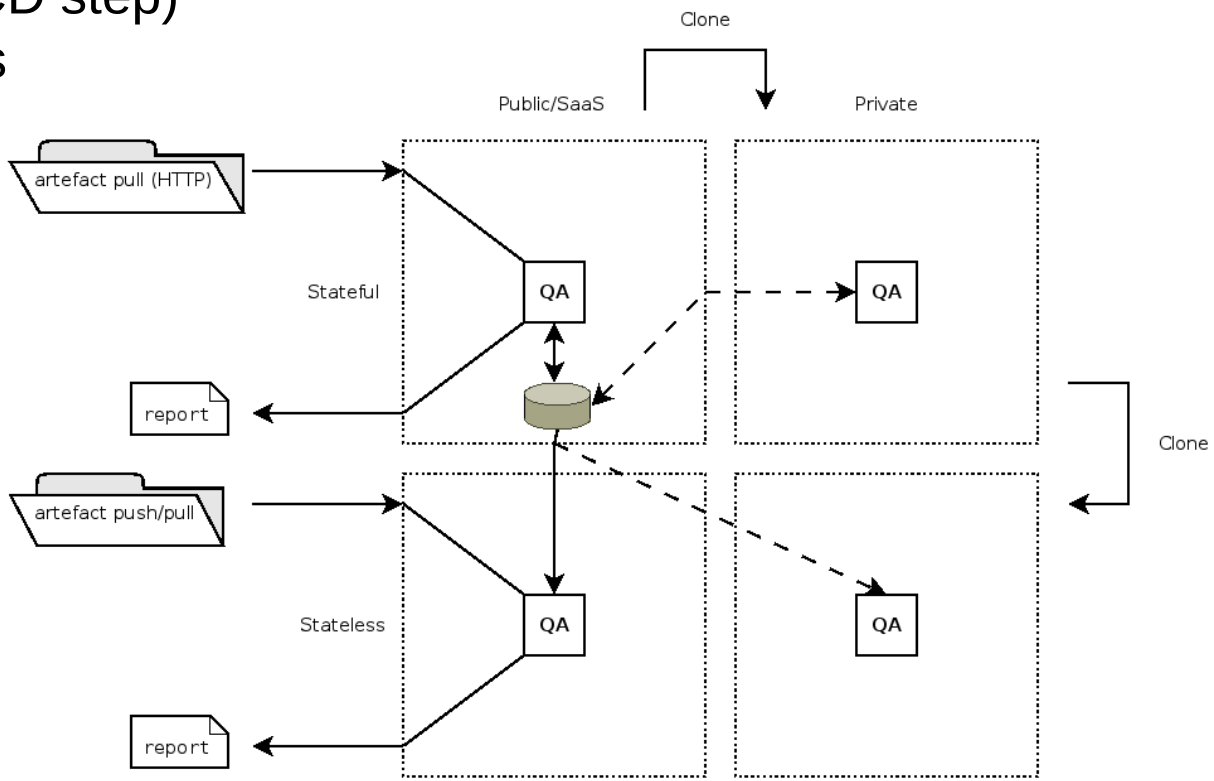


Remarks on evolution tracking

Developer interest:
- on-demand
- real-time QA (e.g. in CI/CD step)
- for limited set of artefacts

Researcher interest
- regular batch generation of results
- not time-critical
- holistic view including history

Hybrid assessment
- important for regressions →



➤➤ Rewind

Key points

Summary

- Software development mindset: code/artefact/(micro)service levels
- Quality assessment perspective
- Temporal perspective - software (artefact) evolution + quality evolution

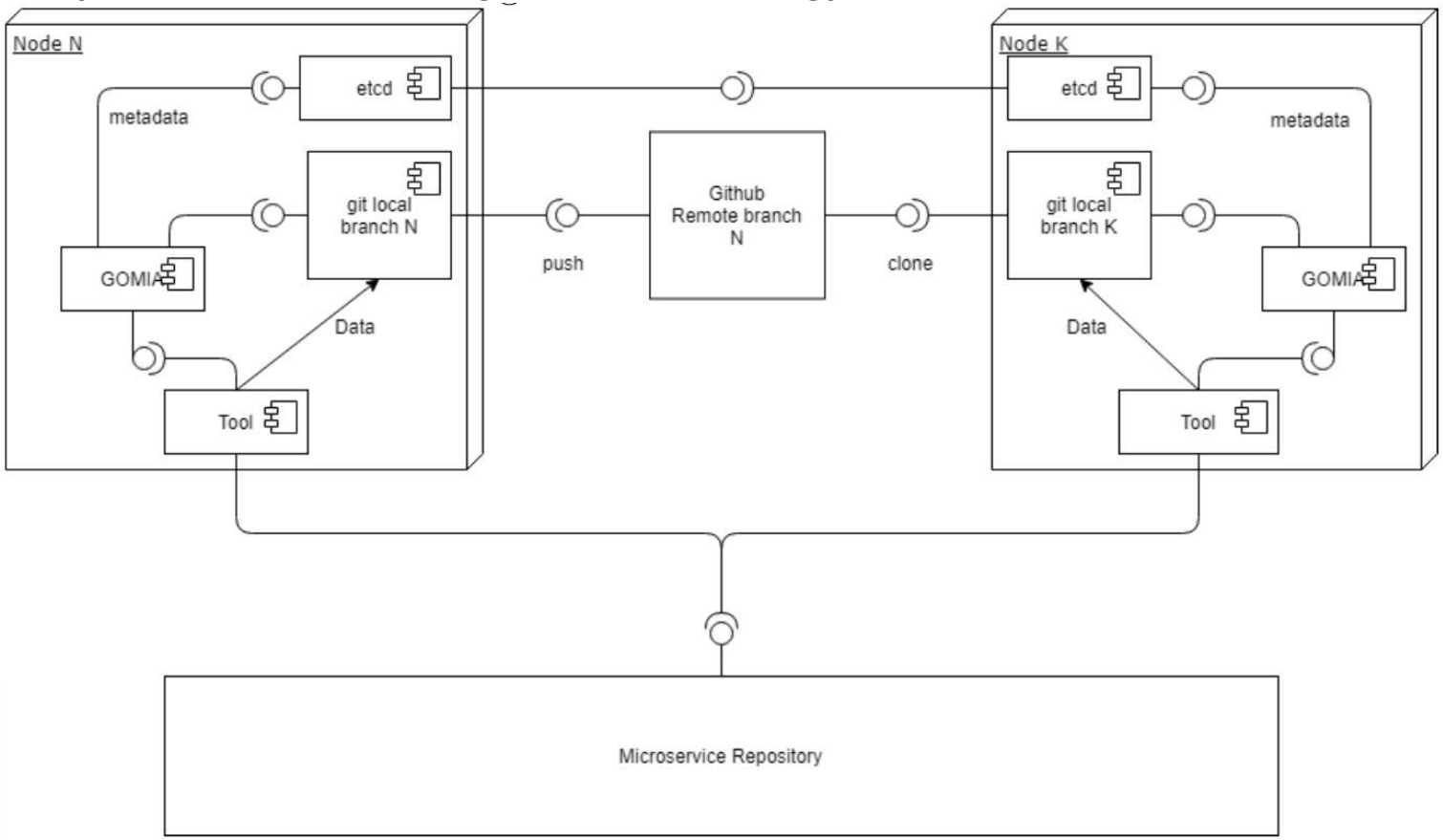
Dual purpose effort

- practical tools for developers
- insightful research infrastructure for scientists

Research on artefact quality+evolution

MAO = Microservice Artefact Observatory

Artefact type-specific checkers (tools) + decentralised architecture + smartness (resilience, self-regression testing)



https://mao-mao-research.github.io/

Research results

HelmQA checker

HelmQA by Service Prototyping Lab, Zurich University of Applied Sciences
Follow the links to per-chart quality reports.

aerospike-0.1.7.tgz
anchore-engine-0.1.7.tgz
artifactory-5.4.6-3.tgz
artifactory-7.1.5.tgz
artifactory-ha-0.1.9.tgz
bitcoind-0.1.3.tgz
buildkite-0.2.3.tgz
burrow-0.4.3.tgz
centrifugo-2.0.1.tgz
cert-manager-0.2.9.tg

HelmQA advice on chart traefik-1.1.0-a.tgz / data point 2018-05-31

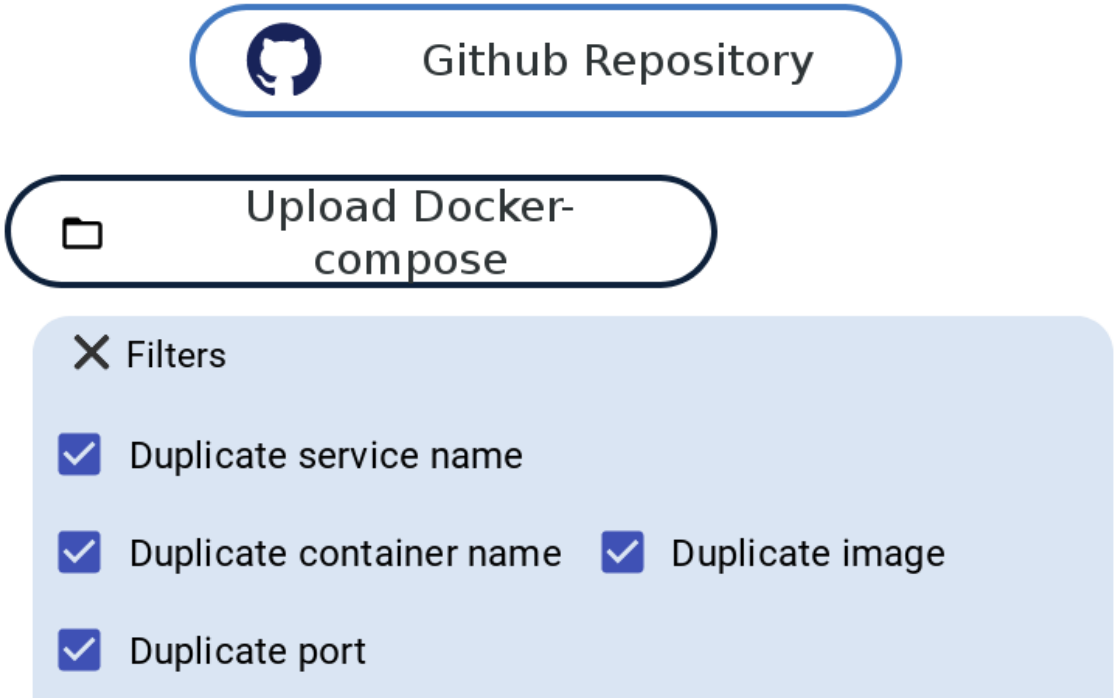
Issue: multiple:traefik-1.9.0.tgz,traefik-1.1.2-h.tgz,traefik-1.2.1-b.tgz,traefik-1.1.1-a.tgz,
Explanation:
Two or more versions of the same chart appear in the same repository.
Rationale:
Developers may not understand which version to use.
Recommended action:
Unless there are compelling technical reasons, the advice is to stick to a single version per cat

Issue: duplicatevalues
Explanation:
The templates contain values with at least two duplicates which are candidates for template v
Rationale:
Configuration duplication should be avoided. While false positives may occur, in some cases th
Recommended action:
Review the list (and if available, the suggested diff) to decide which duplicates should be remo
The following occurrences of duplicate values have been found. Each entry consists of the valu
[['80', 6], ['10', 4], ['1', 4], ['TEMPLATE-dashboard', 3]]
diff sketch available

Research results

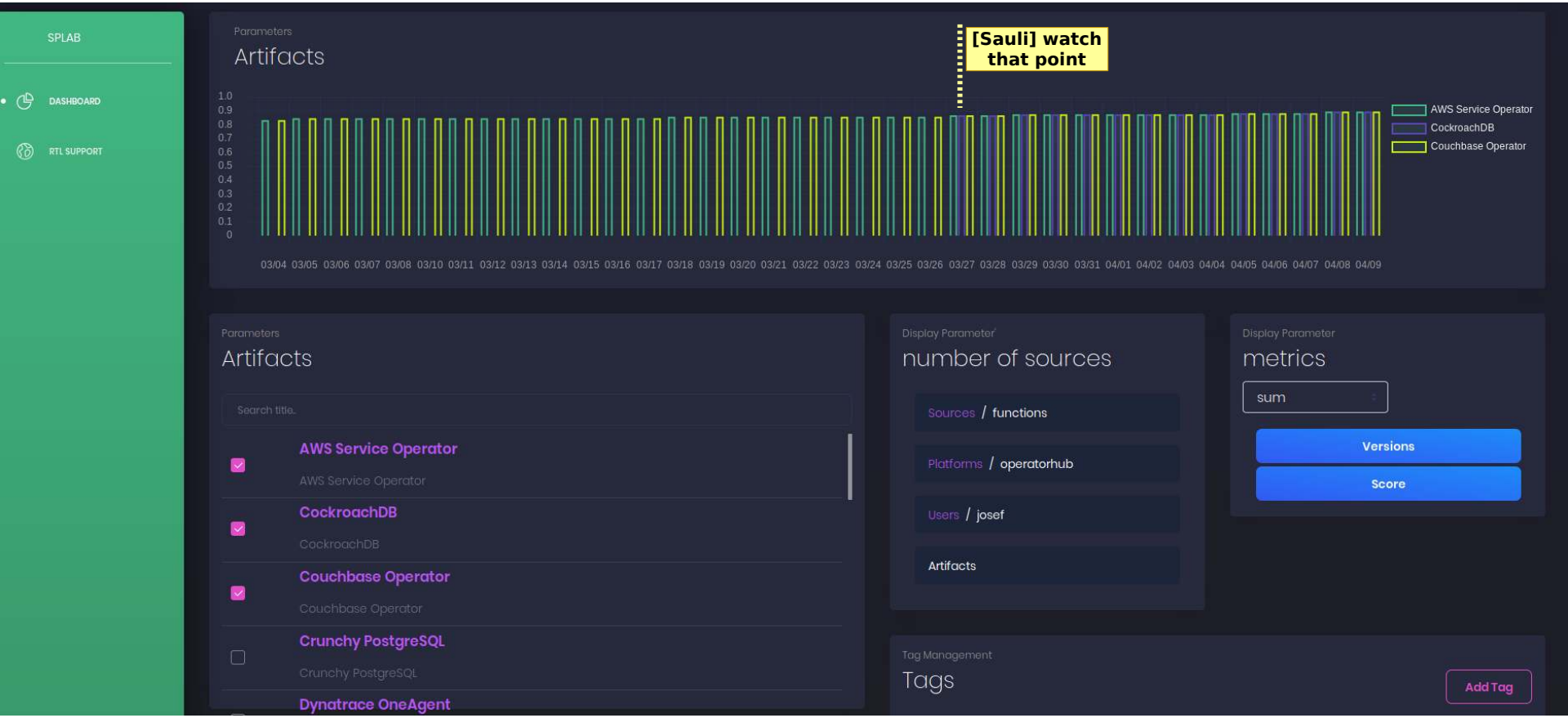
Docker Compose checker

Docker Compose Validator!



Research results

Observatory dashboard across artefact types



- specialised roles (developer, observatory operator, statistician)
- flagging & debating research data “points of interest“ with peers/citizens