What Quality Aspects Influence the Adoption of Docker Images?

Giovanni Rosa, Simone Scalabrino, Gabriele Bavota and Rocco Oliveto

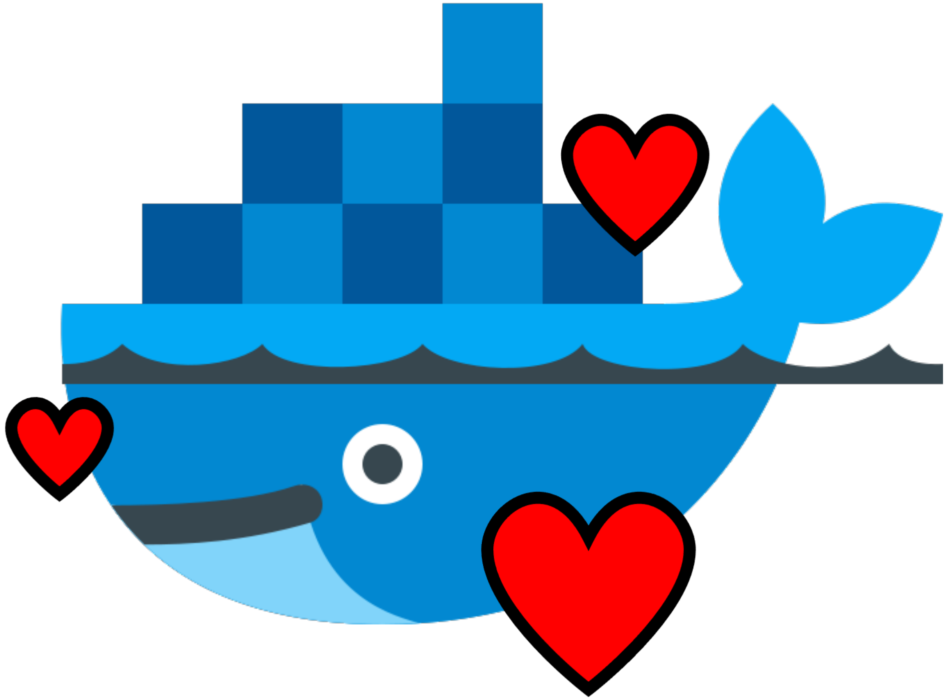
@giovannipink

University of Molise, Italy

ASE 2023

38th International Conference on Automated Software Engineering

11-15 September 2023, ECCL (Luxembourg)



#1 most-desired

and

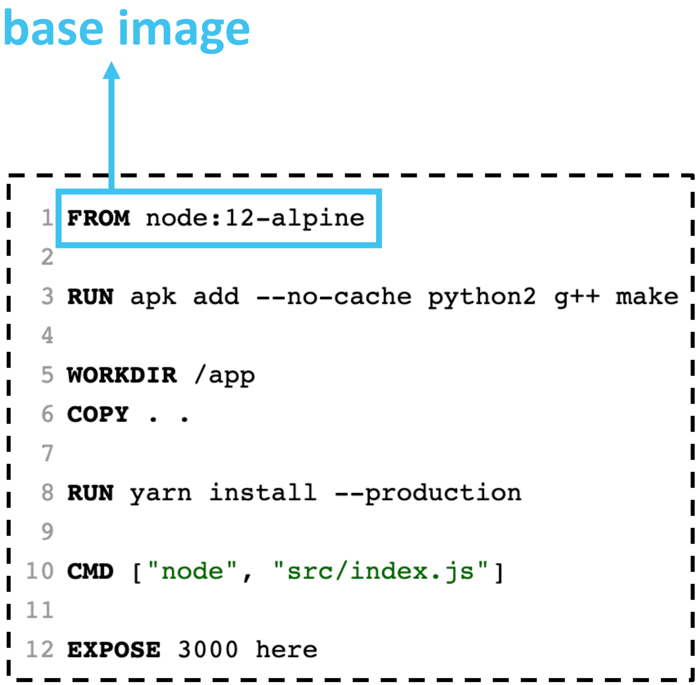
#1 most-used

dev tool

2023 Developer Survey

Why Docker?

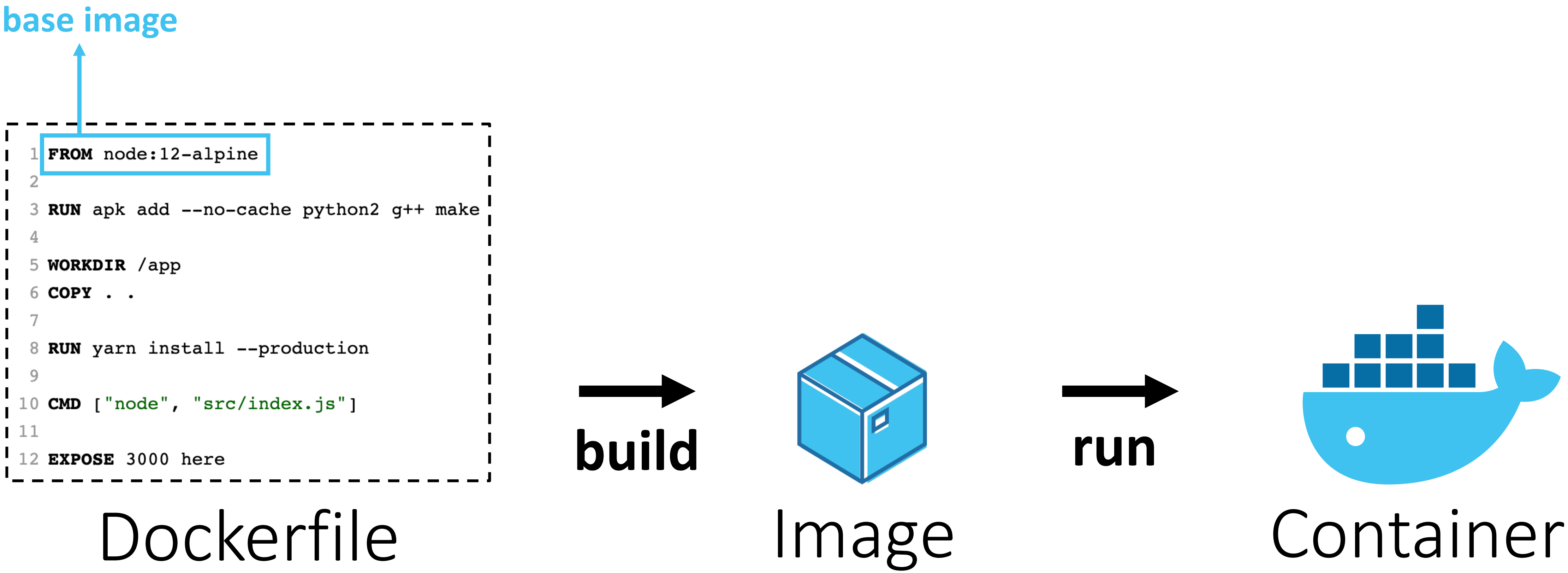
base image



```
FROM node:12-alpine

RUN apk add --no-cache python2 g++ make

WORKDIR /app
COPY . .

RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000 here
```

Dockerfile

Docker in a nutshell



base image

```
FROM node:12-alpine

RUN apk add --no-cache python2 g++ make

WORKDIR /app
COPY . .

RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000 here
```

Dockerfile

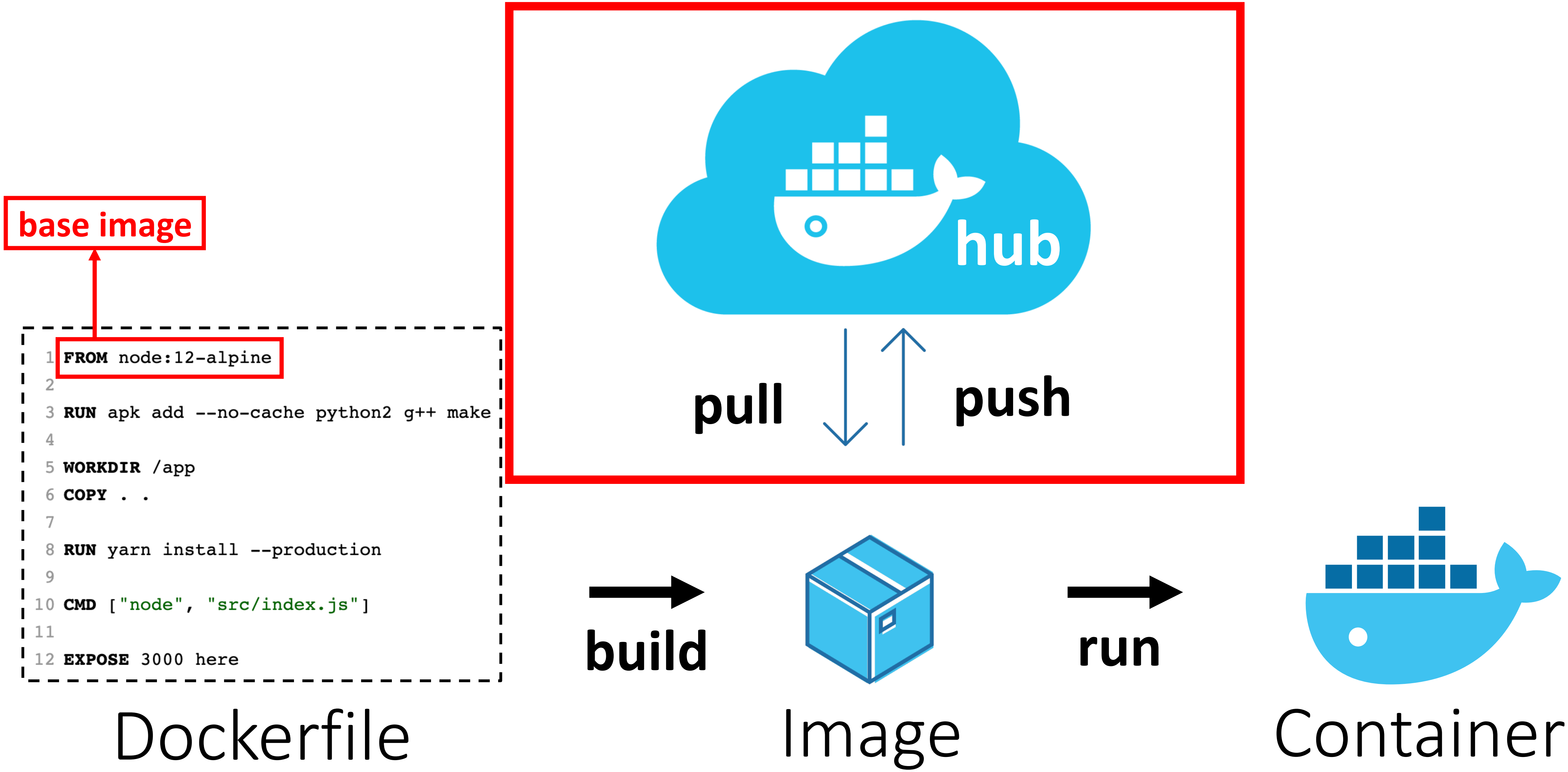
build

Image

run

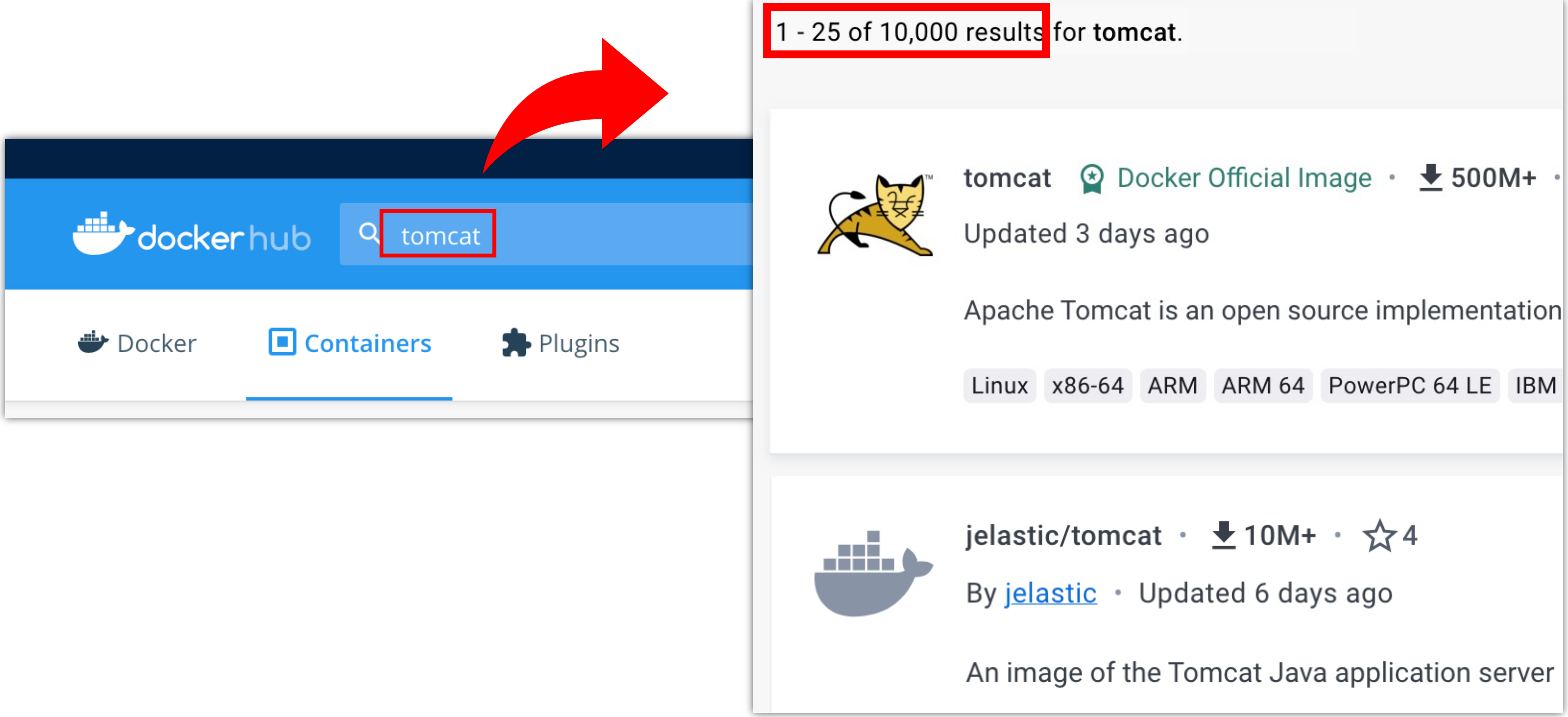
Container

Docker in a nutshell



```dockerfile
FROM node:12-alpine

RUN apk add --no-cache python2 g++ make

WORKDIR /app
COPY . .

RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000 here
```

Docker in a nutshell



1 - 25 of 10,000 results for **tomcat**.

tomcat · Docker Official Image · 500M+ ·

Updated 3 days ago

Apache Tomcat is an open source implementation

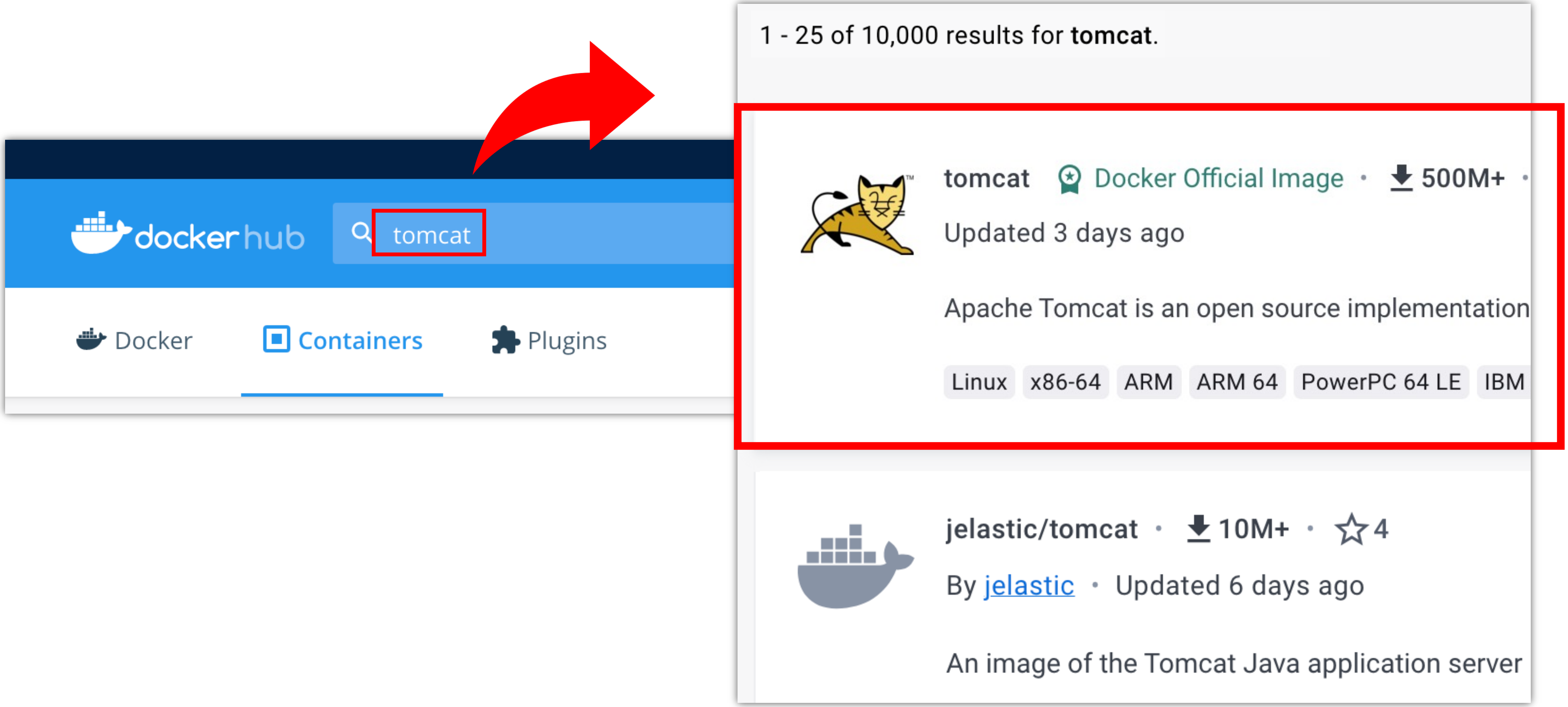
Linux x86-64 ARM ARM 64 PowerPC 64 LE IBM

jelastic/tomcat · 10M+ · ☆ 4

By jelastic · Updated 6 days ago

An image of the Tomcat Java application server

Which Docker image to choose?



1 - 25 of 10,000 results for **tomcat**.

tomcat Docker Official Image • 500M+ •
Updated 3 days ago

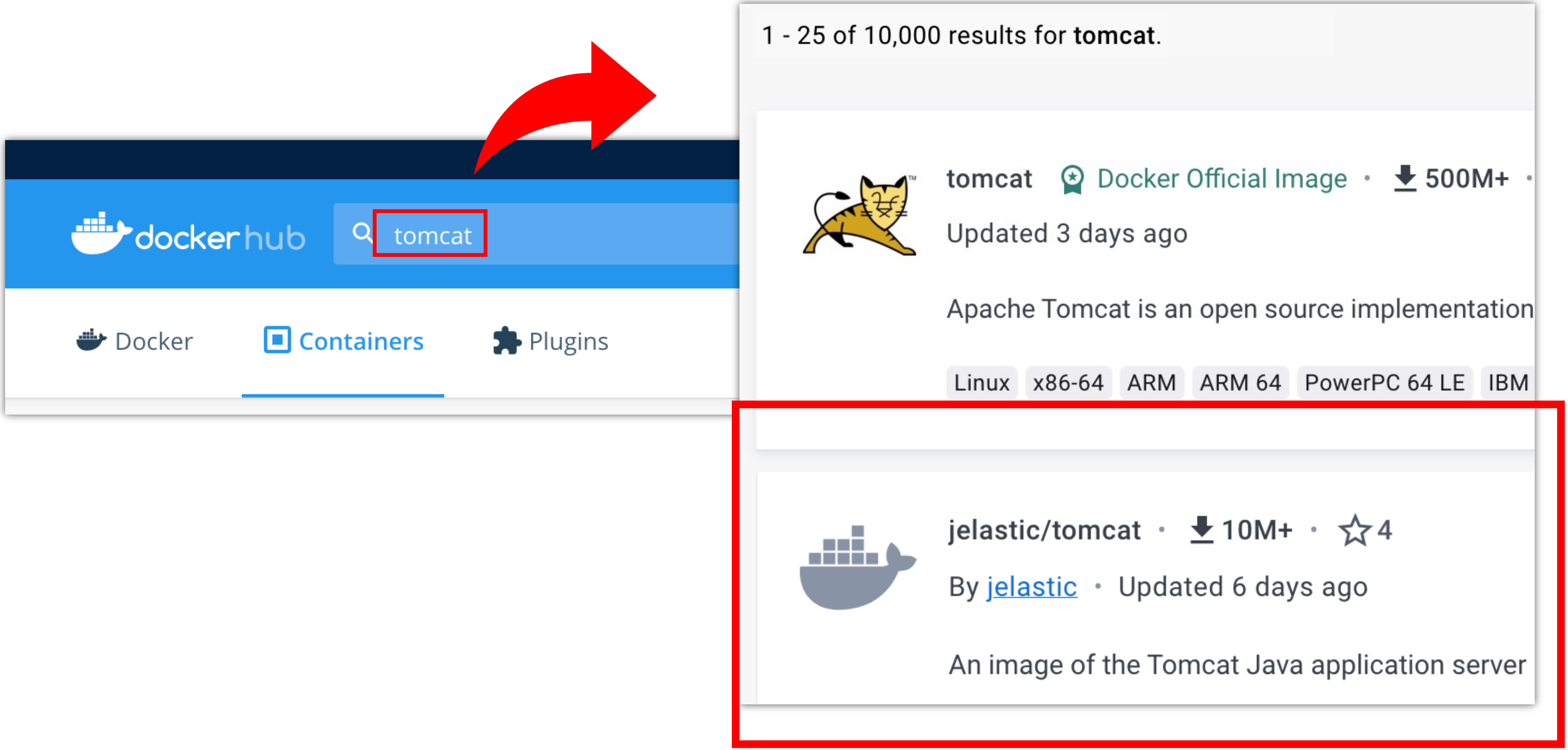
Apache Tomcat is an open source implementation

Linux x86-64 ARM ARM 64 PowerPC 64 LE IBM

jelastic/tomcat • 10M+ • ☆4
By jelastic • Updated 6 days ago

An image of the Tomcat Java application server

Which Docker image to choose?



1 - 25 of 10,000 results for **tomcat**.

tomcat Docker Official Image • 500M+ •

Updated 3 days ago

Apache Tomcat is an open source implementation

Linux x86-64 ARM ARM 64 PowerPC 64 LE IBM

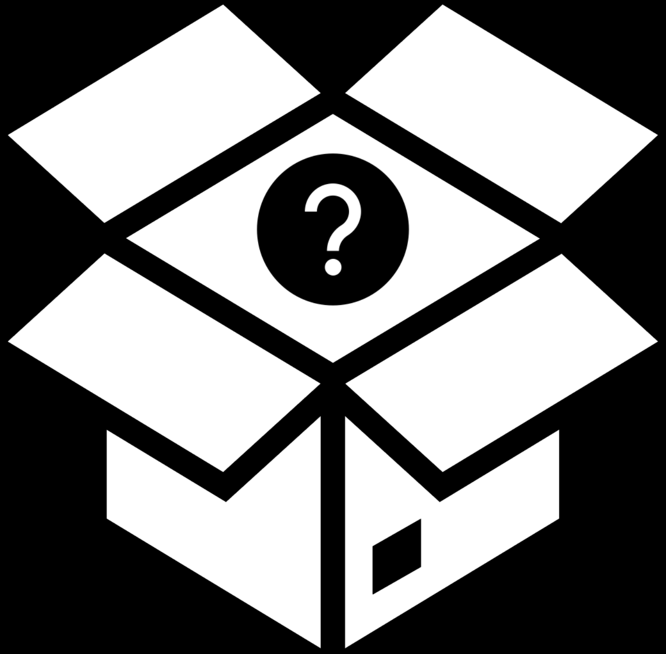
jelastic/tomcat • 10M+ • ☆ 4

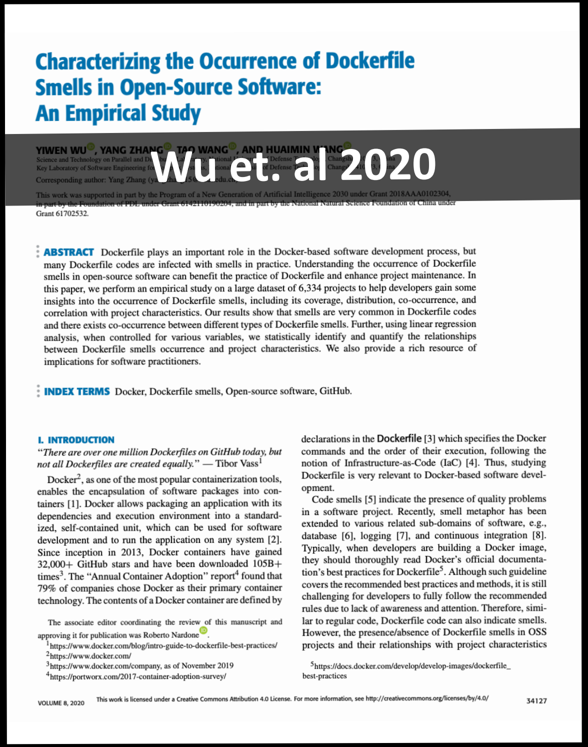
By jelastic • Updated 6 days ago

An image of the Tomcat Java application server

Which Docker image to choose?

How to describe a «good» Docker image?





Characterizing the Occurrence of Dockerfile Smells in Open-Source Software: An Empirical Study

YIWEN WU, YANG ZHANG, TAO WANG, AND HUAIMIN WANG
Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha 410073, China
Key Laboratory of Software Engineering for Complex Systems, National University of Defense Technology, Changsha 410073, China
Corresponding author: Yang Zhang (yangzhang15@nudt.edu.cn)

This work was supported in part by the Program of a New Generation of Artificial Intelligence 2030 under Grant 2018AAA0102304, in part by the Foundation of PDL under Grant 6142110190204, and in part by the National Natural Science Foundation of China under Grant 61702532.

Wu et. al 2020

ABSTRACT Dockerfile plays an important role in the Docker-based software development process, but many Dockerfile codes are infected with smells in practice. Understanding the occurrence of Dockerfile smells in open-source software can benefit the practice of Dockerfile and enhance project maintenance. In this paper, we perform an empirical study on a large dataset of 6,334 projects to help developers gain some insights into the occurrence of Dockerfile smells, including its coverage, distribution, co-occurrence, and correlation with project characteristics. Our results show that smells are very common in Dockerfile codes and there exists co-occurrence between different types of Dockerfile smells. Further, using linear regression analysis, when controlled for various variables, we statistically identify and quantify the relationships between Dockerfile smells occurrence and project characteristics. We also provide a rich resource of implications for software practitioners.

INDEX TERMS Docker, Dockerfile smells, Open-source software, GitHub.

I. INTRODUCTION

"*There are over one million Dockerfiles on GitHub today, but not all Dockerfiles are created equally.*" — Tibor Vass[1]

Docker[2], as one of the most popular containerization tools, enables the encapsulation of software packages into containers [1]. Docker allows packaging an application with its dependencies and execution environment into a standardized, self-contained unit, which can be used for software development and to run the application on any system [2]. Since inception in 2013, Docker containers have gained 32,000+ GitHub stars and have been downloaded 105B+ times[3]. The "Annual Container Adoption" report[4] found that 79% of companies chose Docker as their primary container technology. The contents of a Docker container are defined by declarations in the Dockerfile [3] which specifies the Docker commands and the order of their execution, following the notion of Infrastructure-as-Code (IaC) [4]. Thus, studying Dockerfile is very relevant to Docker-based software development.

Code smells [5] indicate the presence of quality problems in a software project. Recently, smell metaphor has been extended to various related sub-domains of software, e.g., database [6], logging [7], and continuous integration [8]. Typically, when developers are building a Docker image, they should thoroughly read Docker's official documentation's best practices for Dockerfile[5]. Although such guideline covers the recommended best practices and methods, it is still challenging for developers to fully follow the recommended rules due to lack of awareness and attention. Therefore, similar to regular code, Dockerfile code can also indicate smells. However, the presence/absence of Dockerfile smells in OSS projects and their relationships with project characteristics

The associate editor coordinating the review of this manuscript and approving it for publication was Roberto Nardone.

[1] https://www.docker.com/blog/intro-guide-to-dockerfile-best-practices/
[2] https://www.docker.com/
[3] https://www.docker.com/company, as of November 2019
[4] https://portworx.com/2017-container-adoption-survey/
[5] https://docs.docker.com/develop/develop-images/dockerfile_best-practices

"smells are very common in Dockerfile codes"

Characterizing the Occurrence of Dockerfile Smells in Open-Source Software: An Empirical Study

YIWEN WU, YANG ZHANG, TAO WANG, AND HUAIMIN WANG

Science and Technology on Parallel and Di
Key Laboratory of Software Engineering fo

Corresponding author: Yang Zhang (ya

This work was supported in part by the Program of a New Generation of Artificial Intelligence 2030 under Grant 2018AAA0102304, in part by the Foundation of PDL under Grant 6142110190204, and in part by the National Natural Science Foundation of China under Grant 61702532.

ABSTRACT Dockerfile plays an important role in the Docker-based software development process, but many Dockerfile codes are infected with smells in practice. Understanding the occurrence of Dockerfile smells in open-source software can benefit the practice of Dockerfile and enhance project maintenance. In this paper, we perform an empirical study on a large dataset of 6,334 projects to help developers gain some insights into the occurrence of Dockerfile smells, including its coverage, distribution, co-occurrence, and correlation with project characteristics. Our results show th
very common in Dockerfile codes
and there exists co-occurrence between different types of D
ing linear regression
analysis, when controlled for various variables, we sta
tionships
between Dockerfile smells occurrence and project cha
implications for software practitioners.

INDEX TERMS Docker, Dockerfile smells, Open-sou

I. INTRODUCTION

"*There are over one million Dockerfiles on GitHub t*
not all Dockerfiles are created equally." — Tibor V

Docker[2], as one of the most popular containeriza
enables the encapsulation of software packages
tainers [1]. Docker allows packaging an applicati
dependencies and execution environment into
ized, self-contained unit, which can be used
development and to run the application on an
Since inception in 2013, Docker containers
32,000+ GitHub stars and have been downl
times[3]. The "Annual Container Adoption" re
79% of companies chose Docker as their pr
technology. The contents of a Docker contai

The associate editor coordinating the review o
approving it for publication was Roberto Nardone

[1] https://www.docker.com/blog/intro-guide-to-do

[2] https://www.docker.com/

[3] https://www.docker.com/company, as of Nove

[4] https://portworx.com/2017-container-adoptio

VOLUME 8, 2020 This work is licensed under a Creat

Wu et. al 2020

A Study of Security Vulnerabilities on Docker Hub

Rui Shu, Xiaohui Gu and William Enck

ABSTRACT

Keywords

1. INTRODUCTION

Shu et. al 2017

"smells are very common in Dockerfile codes"

"images contain more than 180 vulnerabilities on average"

Characterizing the Occurrence of Dockerfile Smells in Open-Source Software: An Empirical Study

YIWEN WU, YANG ZHANG, TAO WANG, AND HUAIMIN WANG

Wu et. al 2020

This work was supported in part by the Program of a New Generation of Artificial Intelligence 2030 under Grant 2018AAA0102304, in part by the Foundation of PDL under Grant 6142110190204, and in part by the National Natural Science Foundation of China under Grant 61702532.

ABSTRACT Dockerfile plays an important role in the Docker-based software development process, but many Dockerfile codes are infected with smells in practice. Understanding the occurrence of Dockerfile smells in open-source software can benefit the practice of Dockerfile and enhance project maintenance. In this paper, we perform an empirical study on a large dataset of 6,334 projects to help developers gain some insights into the occurrence of Dockerfile smells, including its coverage, distribution, co-occurrence, and correlation with project characteristics. Our results show th … very common in Dockerfile codes and there exists co-occurrence between different types of D … sing linear regression analysis, when controlled for various variables, we sta … tionships between Dockerfile smells occurrence and project cha … implications for software practitioners.

INDEX TERMS Docker, Dockerfile smells, Open-sou

I. INTRODUCTION

"*There are over one million Dockerfiles on GitHub to … not all Dockerfiles are created equally.*" — Tibor V

Docker[2], as one of the most popular containeriza … enables the encapsulation of software packages … tainers [1]. Docker allows packaging an applicati … dependencies and execution environment into … ized, self-contained unit, which can be used … development and to run the application on an … Since inception in 2013, Docker containers … 32,000+ GitHub stars and have been downl … times[3]. The "Annual Container Adoption" re … 79% of companies chose Docker as their pr … technology. The contents of a Docker contai

The associate editor coordinating the review o … approving it for publication was Roberto Nardone

[1] https://www.docker.com/blog/intro-guide-to-do

[2] https://www.docker.com/

[3] https://www.docker.com/company, as of Nove

[4] https://portworx.com/2017-container-adoptio

VOLUME 8, 2020 This work is licensed under a Creat

A Study of Security Vulnerabilities on Docker Hub

Shu et. al 2017

Empirical Software Engineering (2020) 25:4250–4281
https://doi.org/10.1007/s10664-020-09873-0

Check for updates

Too many images on DockerHub! How different are images for the same system?

Md Hasan Ibrahim[1] · Mohammed Sayagh[1] · Ahmed E. Hassan[1]

Ibrahim et. al 2020

Published online: 28 August 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract
Containerization is a technique used to encapsulate a software system and its dependencies into one isolated package, which is called a container. The goal of these containers is to deploy or replicate a software system on various platforms and environments without facing any compatibility or dependency issues. Developers can instantiate these containers from images using Docker; one of the most popular containerization platforms. Furthermore, many of these images are publicly available on DockerHub, on which developers can share their images with the community who in turn can leverage such publicly available image. However, DockerHub contains thousands of images for each software system, which makes the selection of an image a nontrivial task. In this paper, we investigate the differences among DockerHub images for five software systems and 936 images with the goal of helping Docker tooling creators and DockerHub better guide users select a suitable image. We observe that users tend to download the official images (images that are provided by Docker itself) when there exist a large number of image choices for each single software system on the community images (images that are provided by the community developers), which are in many cases more resource efficient (have less duplicate resources) and have less security vulnerabilities. In fact, we observe that 27% (median), 35% (median), 6% (median), and 9% (median) of the DockerHub Debian, Centos, Ubuntu, and Alpine based images are identical to another image across all the studied software systems. Furthermore, 26% (median), 49% (median), and 8% (median) of the Alpine, Debian, and Ubuntu based community images are more resource efficient than their respective official images across all the five studied software systems. 7% (median) of the community Debian based images have less security vulnerabilities than their respective official images across the four studied software systems, for which an official Debian based image exists. Unfortunately, the … studied images do not guide users when selecting an image (the … highlight the particularities of the image), … to distinguish

"smells are very common
in Dockerfile codes"

"images contain more than
180 vulnerabilities on average"

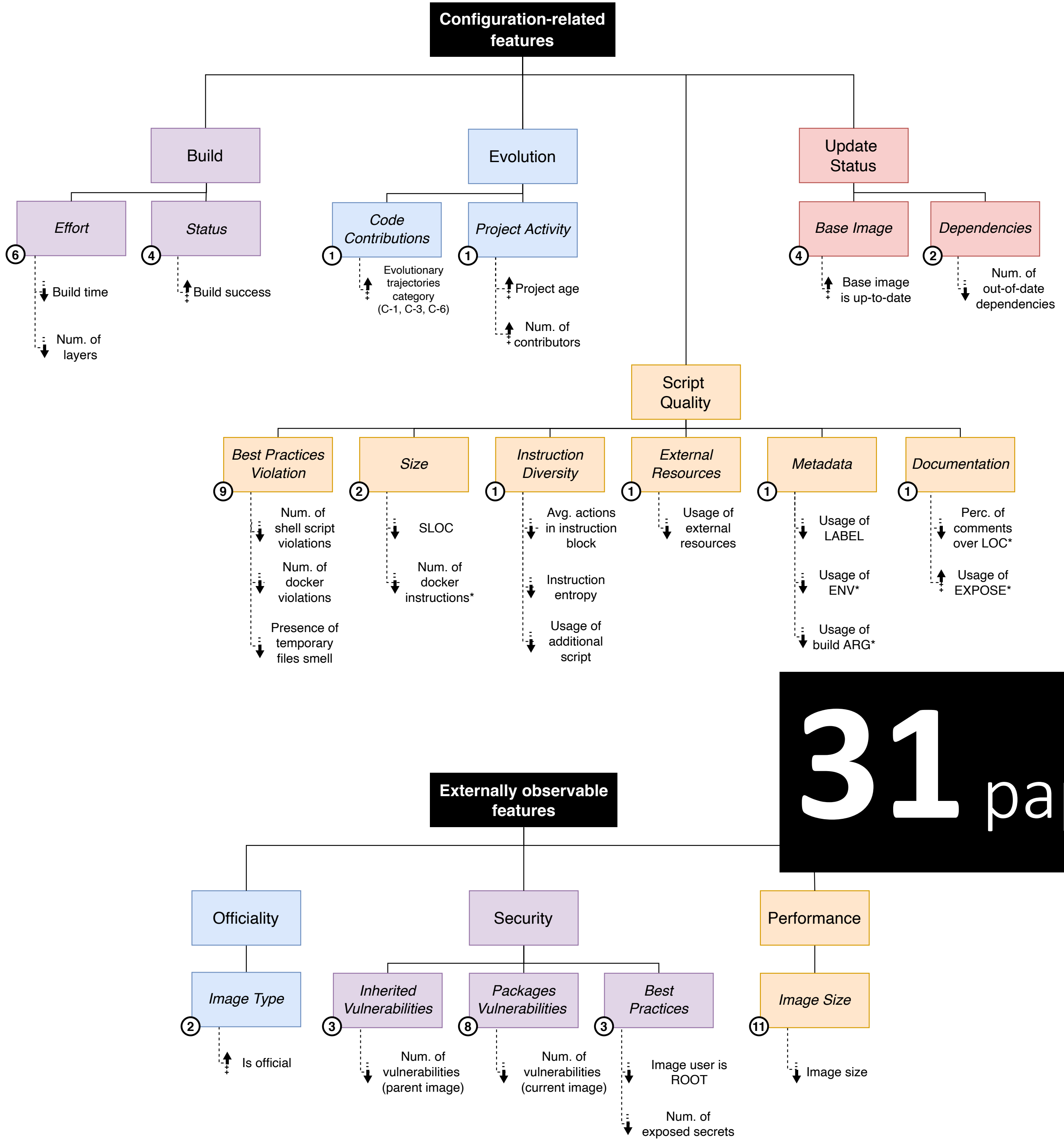
"community images are more
resource-efficient and have
fewer vulnerabilities"

What **quality features** characterize the adoption of a Docker image (and its Dockerfile)

Step 1:

Learning from the Literature

Literature review



Configuration-related features

- Build
  - *Effort* 6
    - ↓ Build time
    - ↓ Num. of layers
  - *Status* 4
    - ↑ Build success
- Evolution
  - *Code Contributions* 1
    - ↑ Evolutionary trajectories category (C-1, C-3, C-6)
  - *Project Activity* 1
    - ↑ Project age
    - ↑ Num. of contributors
- Update Status
  - *Base Image* 4
    - ↑ Base image is up-to-date
  - *Dependencies* 2
    - ↓ Num. of out-of-date dependencies
- Script Quality
  - *Best Practices Violation* 9
    - ↓ Num. of shell script violations
    - ↓ Num. of docker violations
    - ↓ Presence of temporary files smell
  - *Size* 2
    - ↓ SLOC
    - ↓ Num. of docker instructions*
  - *Instruction Diversity* 1
    - ↓ Avg. actions in instruction block
    - ↓ Instruction entropy
    - ↓ Usage of additional script
  - *External Resources* 1
    - ↓ Usage of external resources
  - *Metadata* 1
    - ↓ Usage of LABEL
    - ↓ Usage of ENV*
    - ↓ Usage of build ARG*
  - *Documentation* 1
    - ↓ Perc. of comments over LOC*
    - ↑ Usage of EXPOSE*

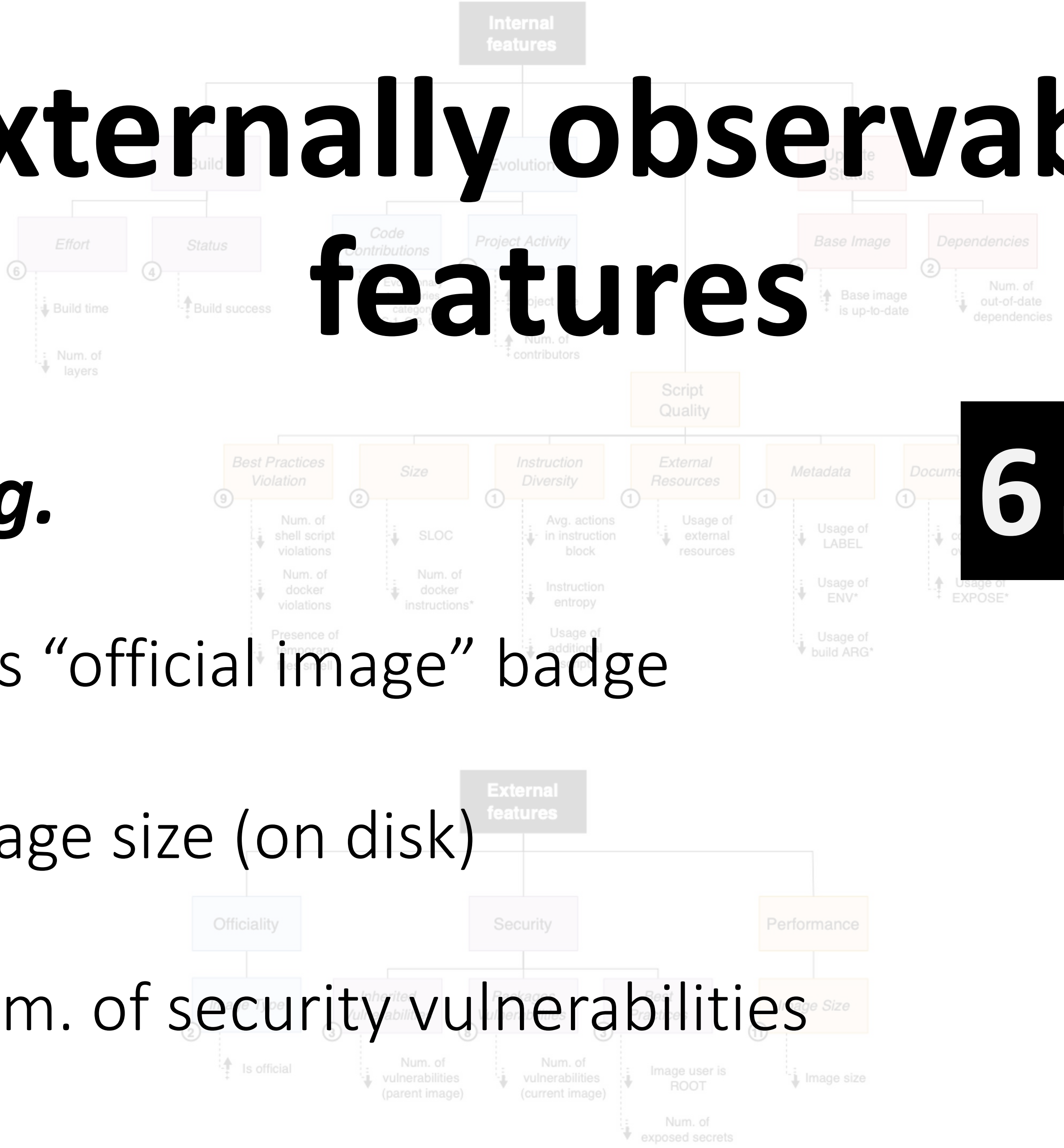
Externally observable features

- Officiality
  - *Image Type* 2
    - ↑ Is official
- Security
  - *Inherited Vulnerabilities* 3
    - ↓ Num. of vulnerabilities (parent image)
  - *Packages Vulnerabilities* 8
    - ↓ Num. of vulnerabilities (current image)
  - *Best Practices* 3
    - ↓ Image user is ROOT
    - ↓ Num. of exposed secrets
- Performance
  - *Image Size* 11
    - ↓ Image size

31 papers

Taxonomy of quality metrics

Externally observable features



6 metrics

e.g.

Has “official image” badge

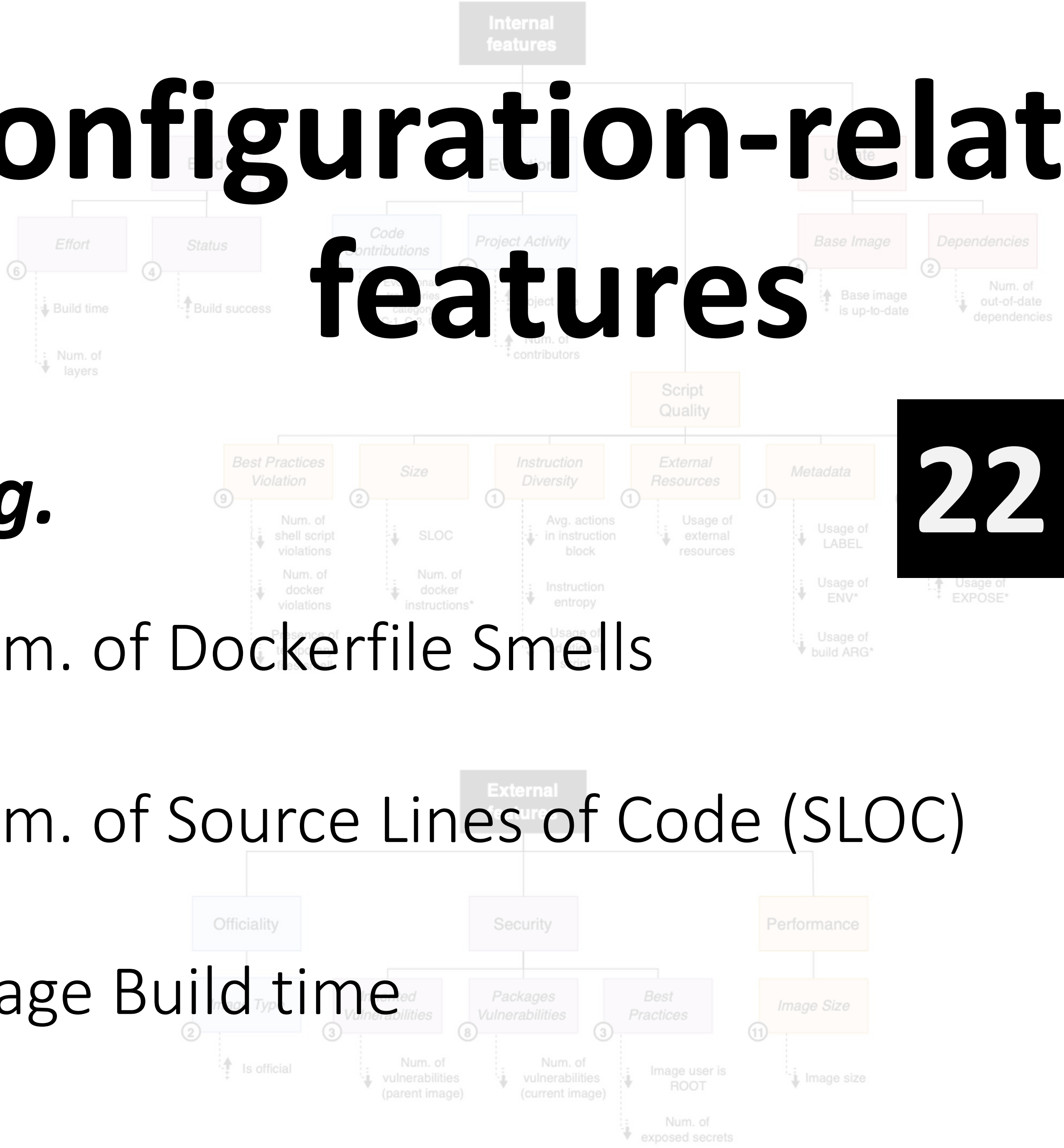
Image size (on disk)

Num. of security vulnerabilities

….

Taxonomy of quality metrics

Configuration-related features



22 metrics

e.g.

Num. of Dockerfile Smells

Num. of Source Lines of Code (SLOC)

Image Build time

…

Step 2:

Catching the Developers’ Preferences

RQ1

Can the externally observable features explain the developers’ preference for a Docker image?

RQ1: Context

~50k

open-source repos

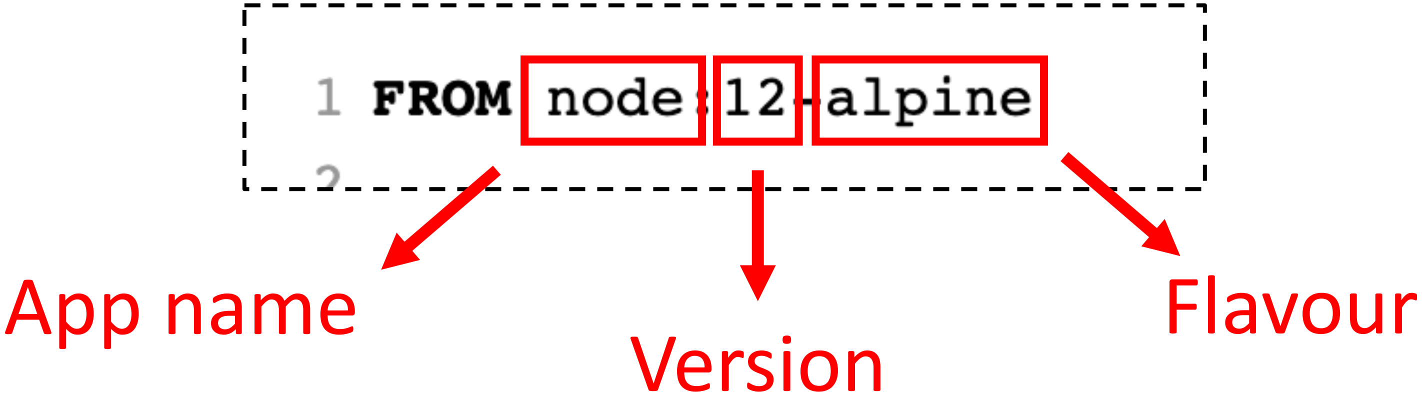
→

~2.4k Docker images

RQ1: Context

~50k open-source repos

~2.4k Docker images



```
FROM node:12-alpine
```

10 most-used apps

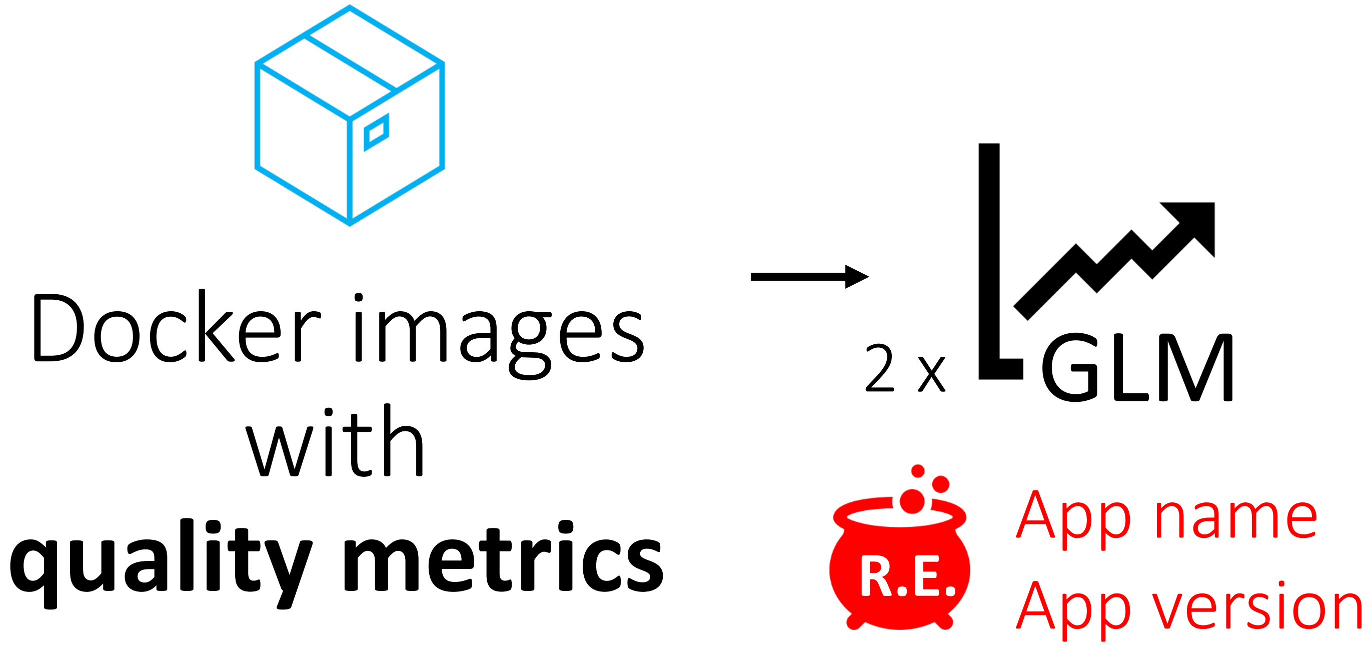
RQ1: Experiment

Docker images
with
quality metrics

→

GLM

RQ1: Experiment



Docker images with **quality metrics**

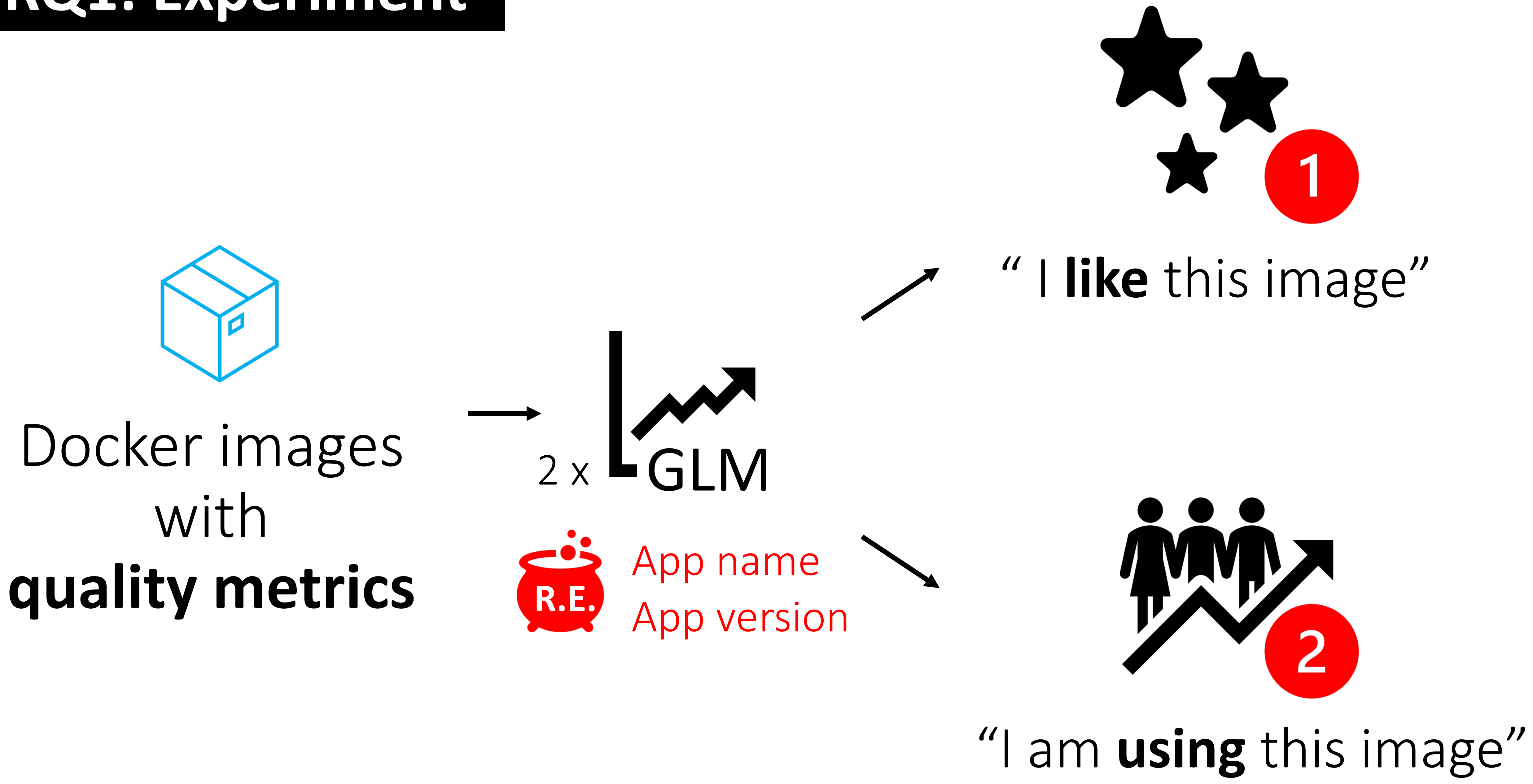
2 x GLM

R.E.

App name

App version

RQ1: Experiment



Docker images with **quality metrics**

2 x GLM

App name

App version

“ I **like** this image”

“I am **using** this image”

RQ1: Results

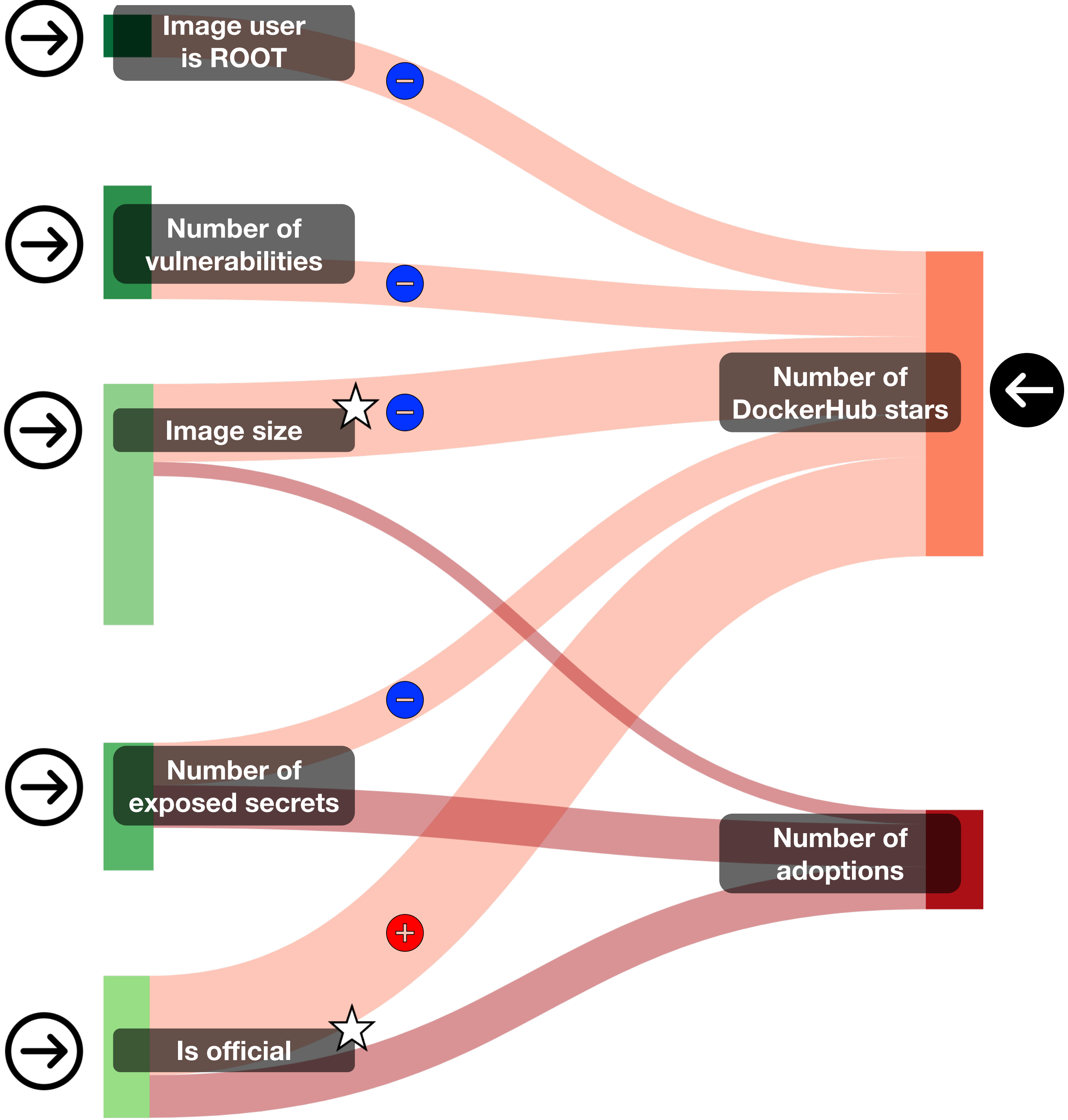


Image user is ROOT
Number of vulnerabilities
Image size
Number of exposed secrets
Is official
Number of DockerHub stars
Number of adoptions
−
−
−
−
+
☆
☆

RQ1: Results

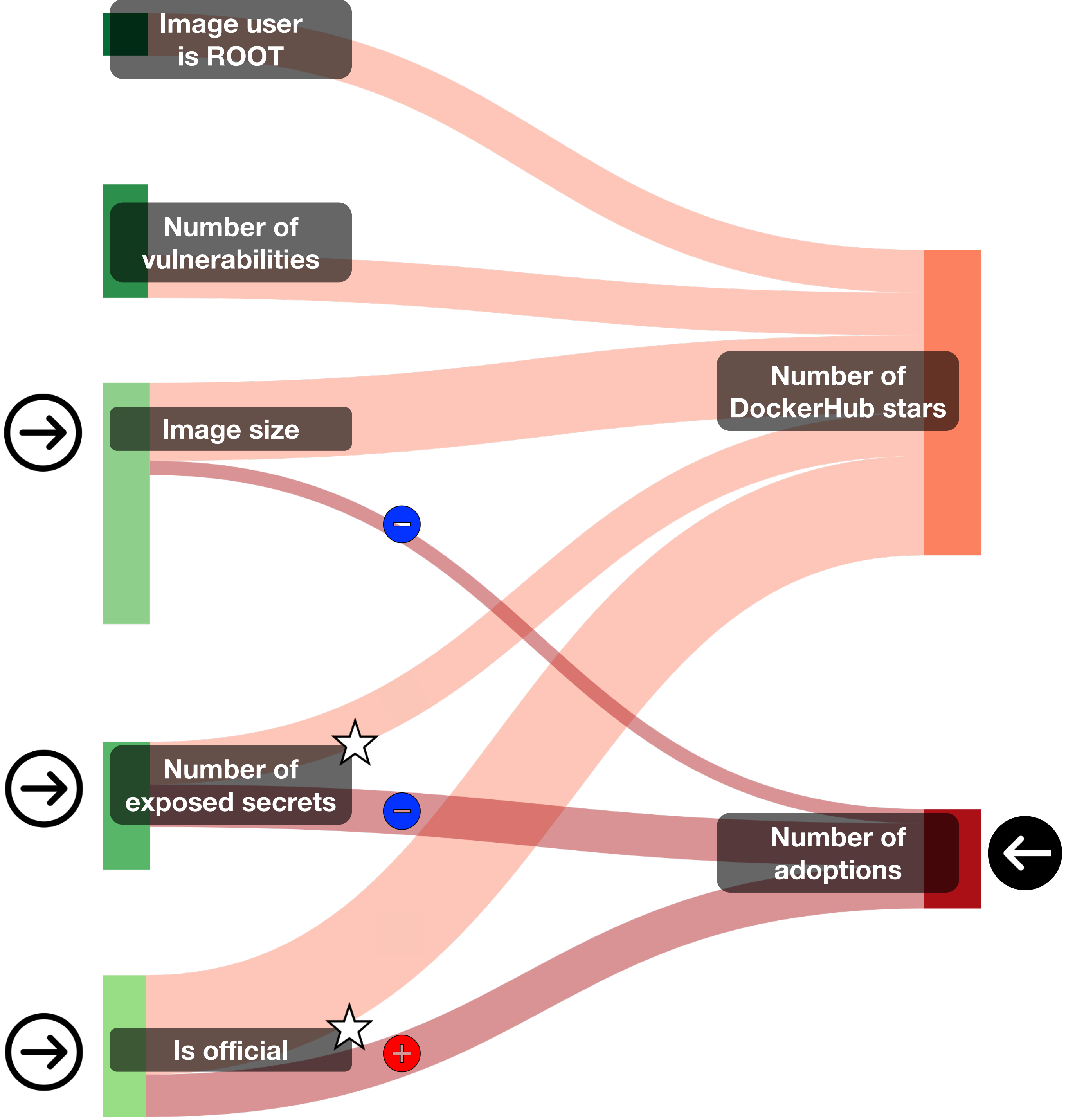


Image user is ROOT
Number of vulnerabilities
Image size
Number of exposed secrets
Is official
Number of DockerHub stars
Number of adoptions
−
−
+

RQ2

Are configuration-related features correlated with externally observable features ?

RQ2: Context

2.4k

Docker images

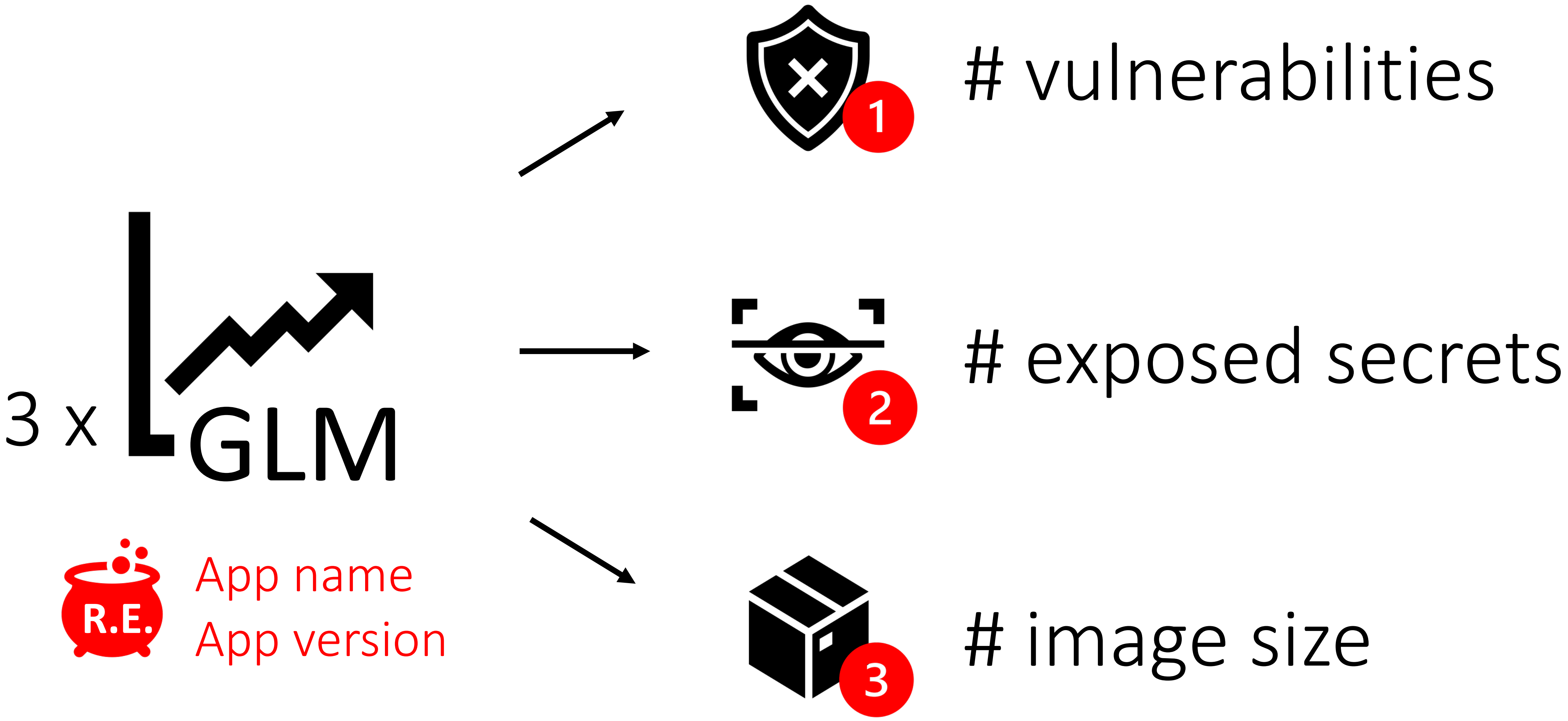
10

most-used apps

→

~300 with source Dockerfiles

RQ2: Experiment



3 x GLM

R.E.

App name

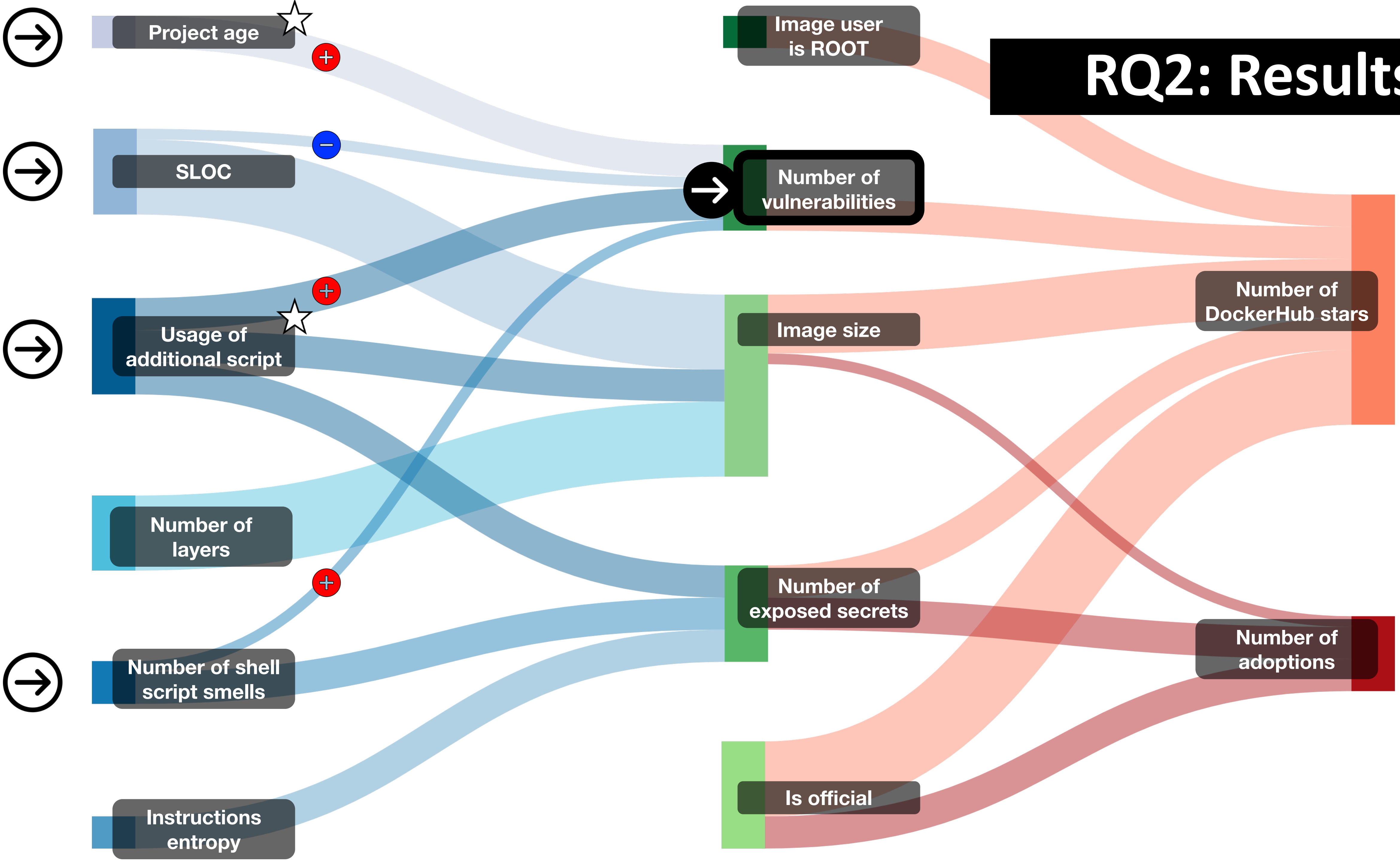
App version

1 # vulnerabilities

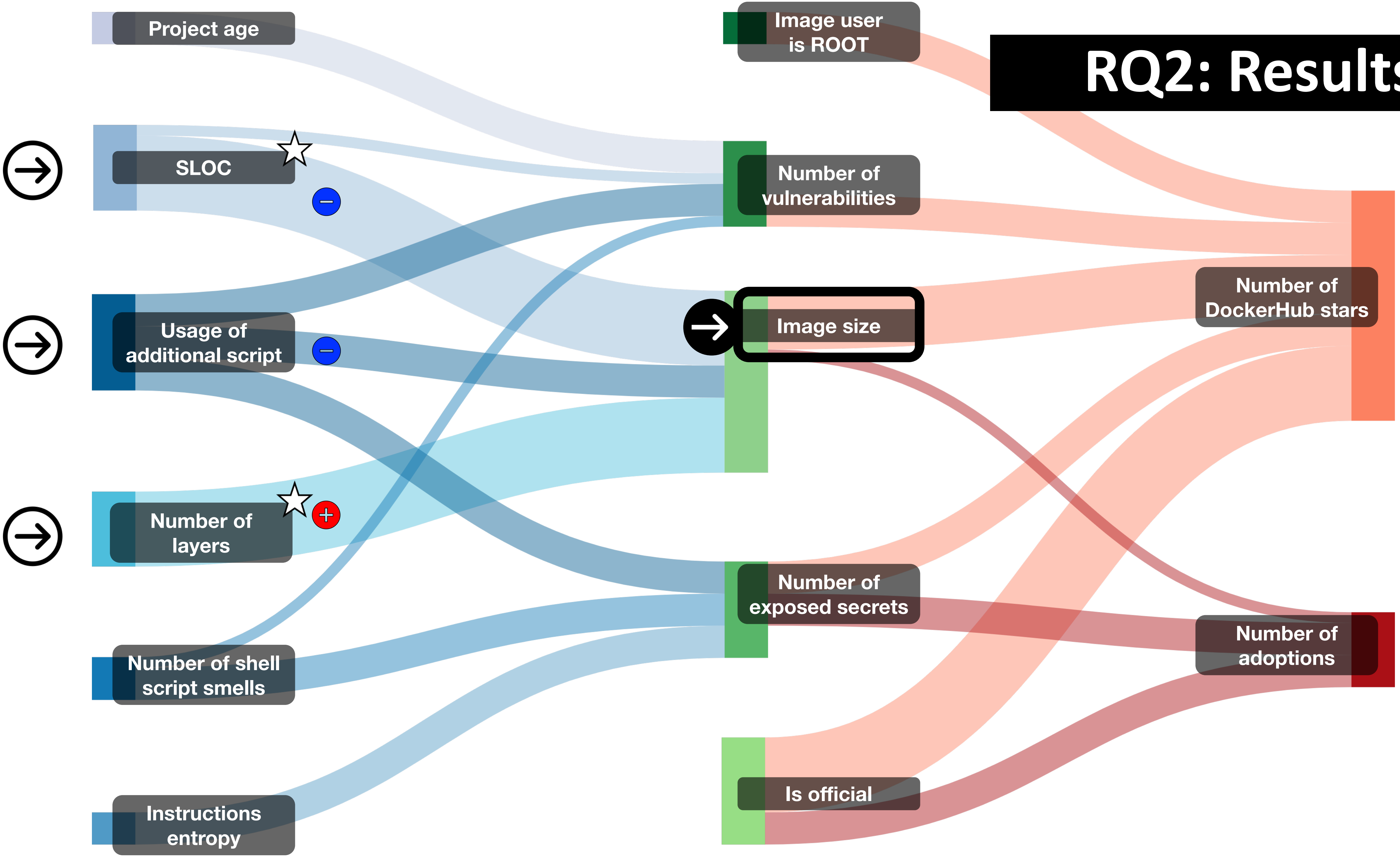
2 # exposed secrets

3 # image size

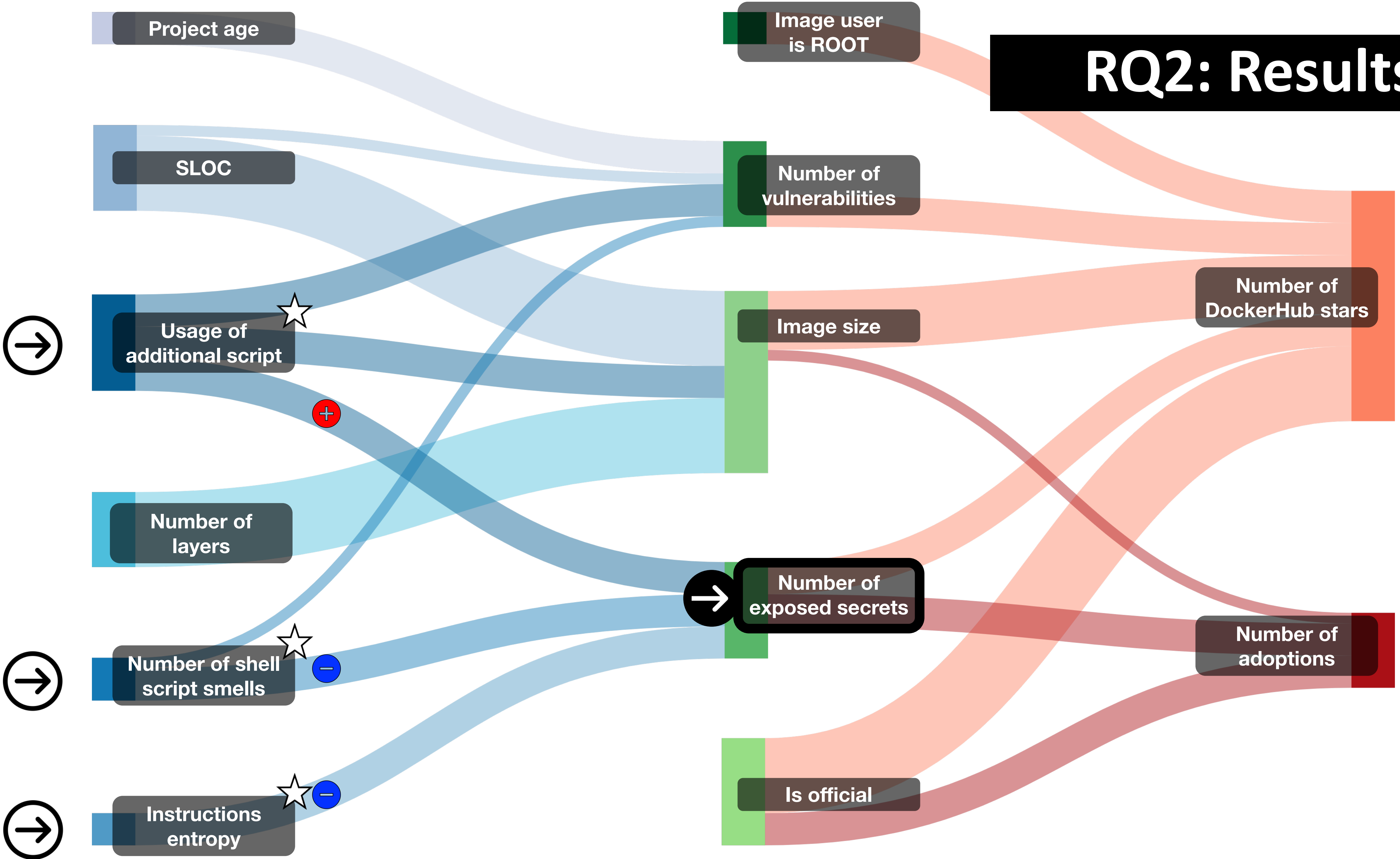
RQ2: Results


Project age
SLOC
Usage of additional script
Number of layers
Number of shell script smells
Instructions entropy
+
−
+
+
Image user is ROOT
Number of vulnerabilities
Image size
Number of exposed secrets
Is official
Number of DockerHub stars
Number of adoptions

RQ2: Results

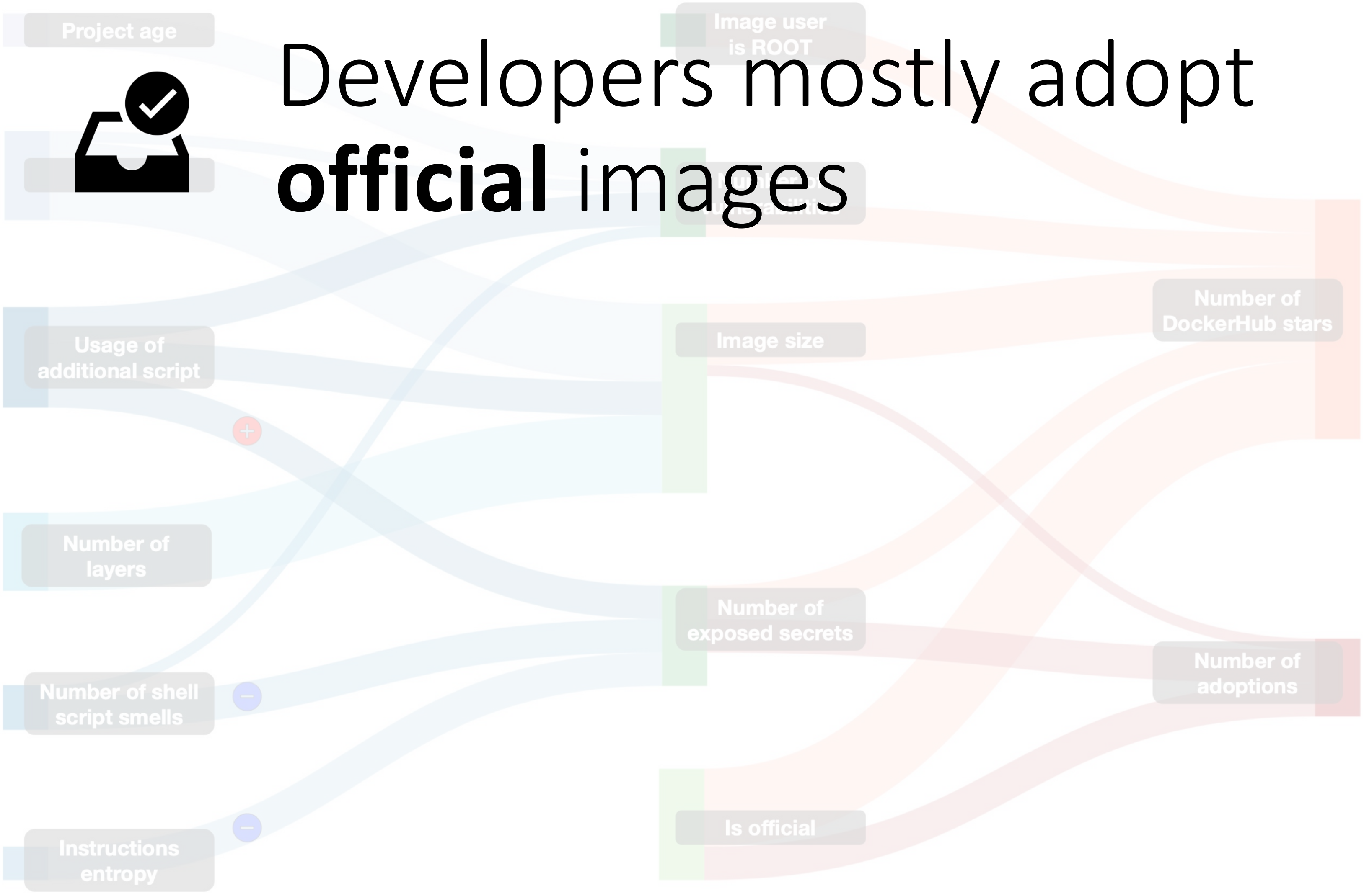

Project age
SLOC
Usage of additional script
Number of layers
Number of shell script smells
Instructions entropy
Image user is ROOT
Number of vulnerabilities
Image size
Number of exposed secrets
Is official
Number of DockerHub stars
Number of adoptions
−
−
+

RQ2: Results

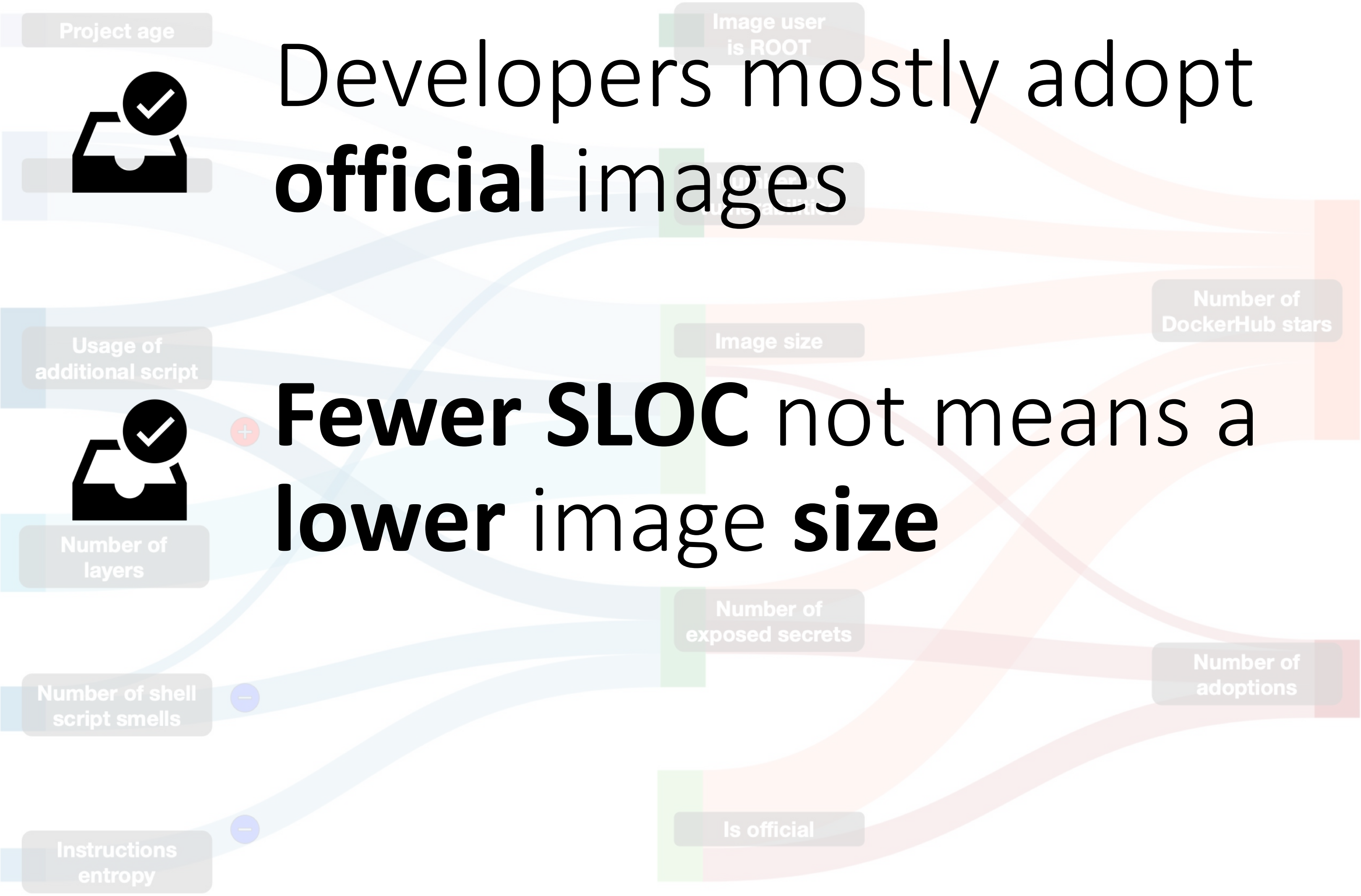

Project age
SLOC
Usage of additional script
Number of layers
Number of shell script smells
Instructions entropy
Image user is ROOT
Number of vulnerabilities
Image size
Number of exposed secrets
Is official
Number of DockerHub stars
Number of adoptions
+
−
−

Takeaways

Developers mostly adopt **official** images



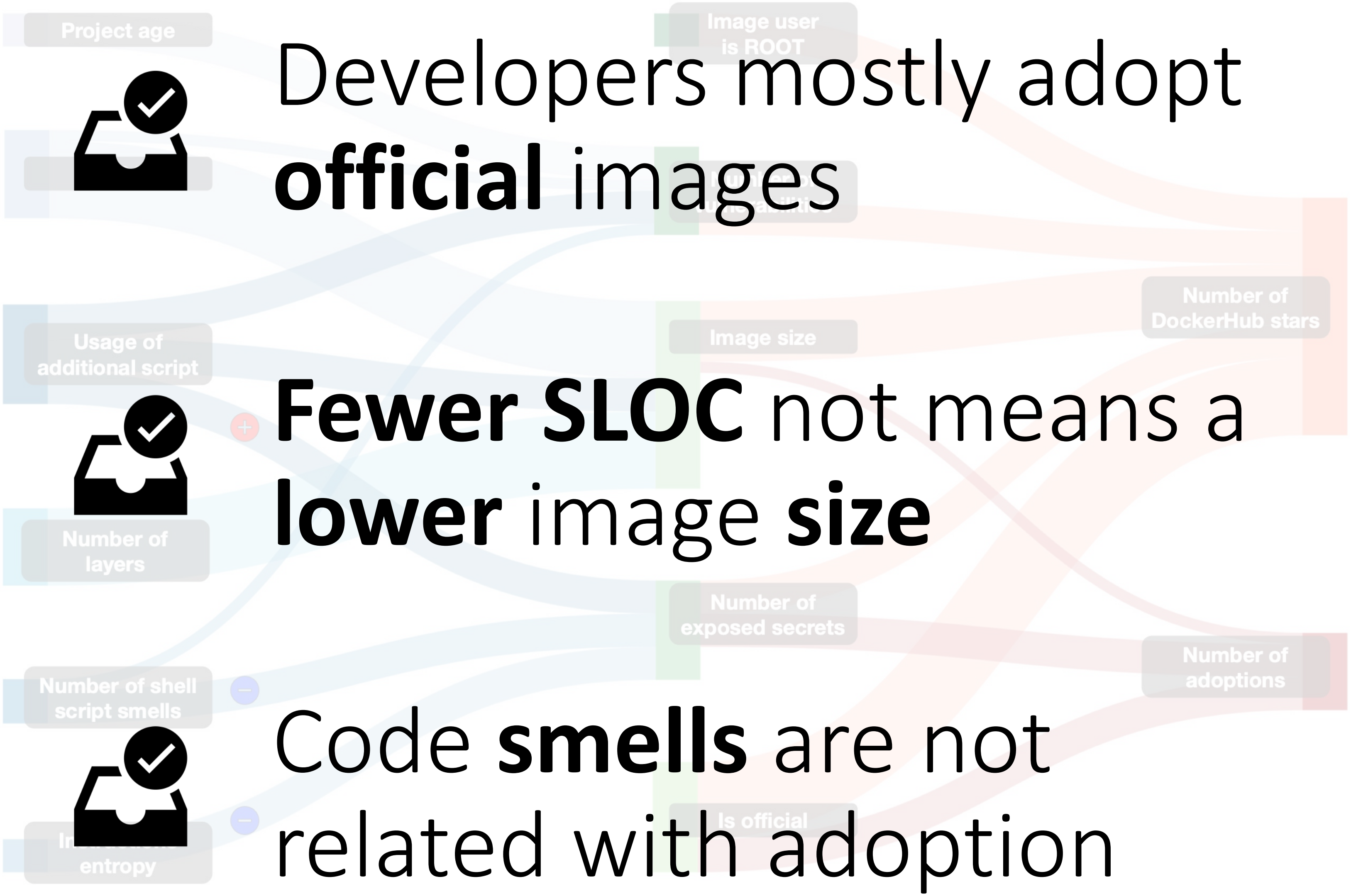
Takeaways



✓ Developers mostly adopt **official** images

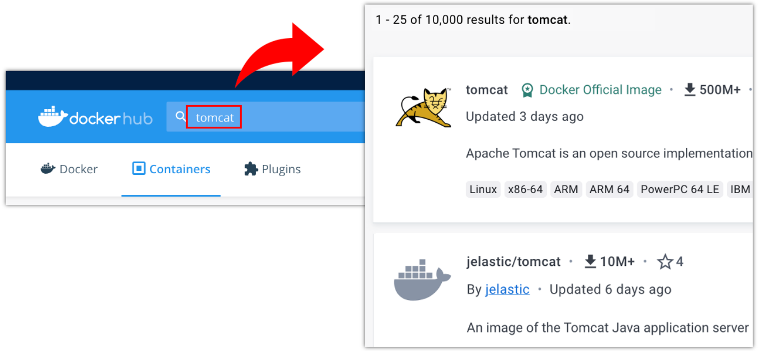
✓ **Fewer SLOC** not means a **lower** image **size**

Takeaways


Project age
Image user is ROOT
Usage of additional script
Image size
Number of DockerHub stars
+
Number of layers
Number of exposed secrets
Number of shell script smells
−
Number of adoptions
−
Is official
entropy

- Developers mostly adopt **official** images
- **Fewer SLOC** not means a **lower** image **size**
- Code **smells** are not related with adoption

Summary



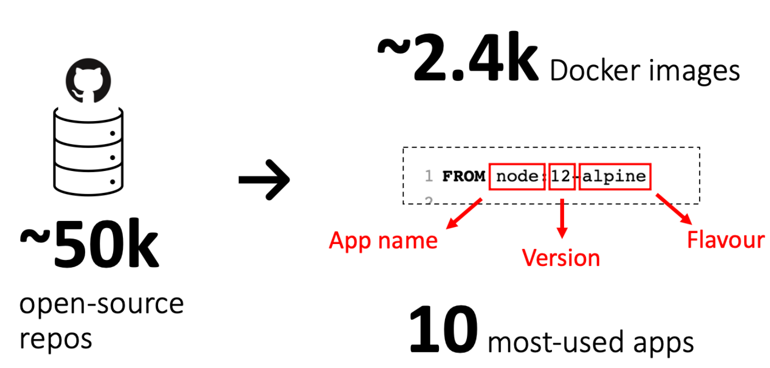
1 - 25 of 10,000 results for **tomcat**.

tomcat · Docker Official Image · 500M+ ·
Updated 3 days ago
Apache Tomcat is an open source implementation
Linux x86-64 ARM ARM 64 PowerPC 64 LE IBM

jelastic/tomcat · 10M+ · ☆4
By jelastic · Updated 6 days ago
An image of the Tomcat Java application server

Which Docker image to choose?

RQ1: Context

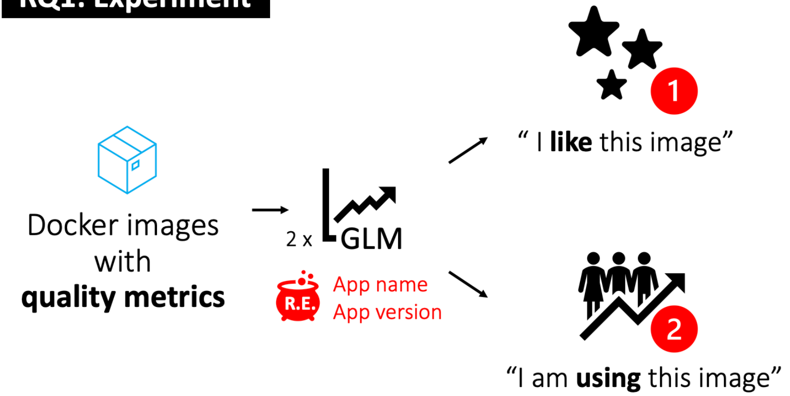


~50k open-source repos → ~2.4k Docker images

```
1 FROM node:12-alpine
```

10 most-used apps

RQ1: Experiment

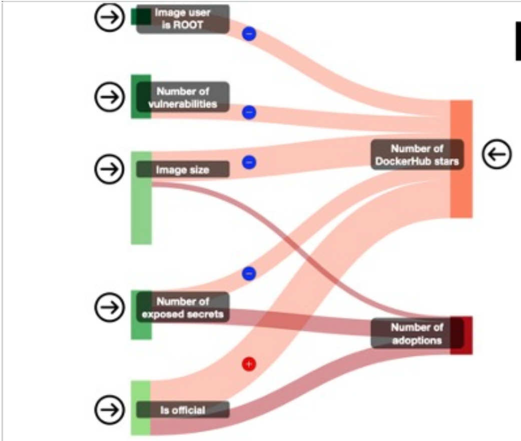


Docker images with **quality metrics** → 2 x GLM

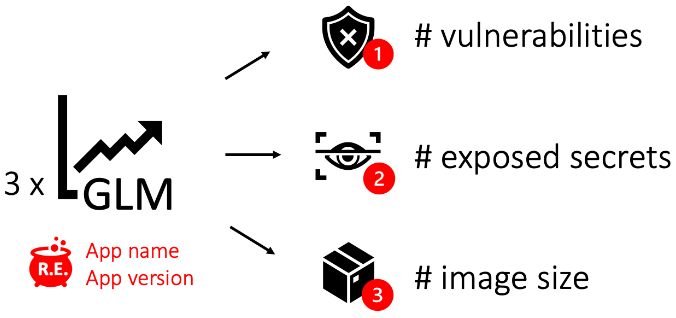
“ I **like** this image”

“I am **using** this image”

RQ1: Results



RQ2: Experiment



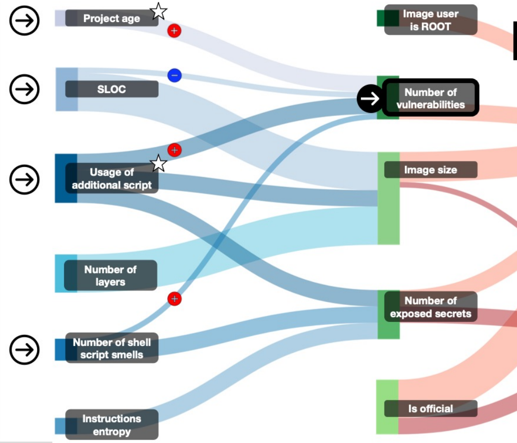
3 x GLM

vulnerabilities

exposed secrets

image size

RQ2: Results



Giovanni Rosa

giovanni.rosa@unimol.it

Get the paper here! →