Tackle Containerization Advisor (TCA) for Legacy Applications

Accelerate your journey to Kubernetes with the Konveyor Community

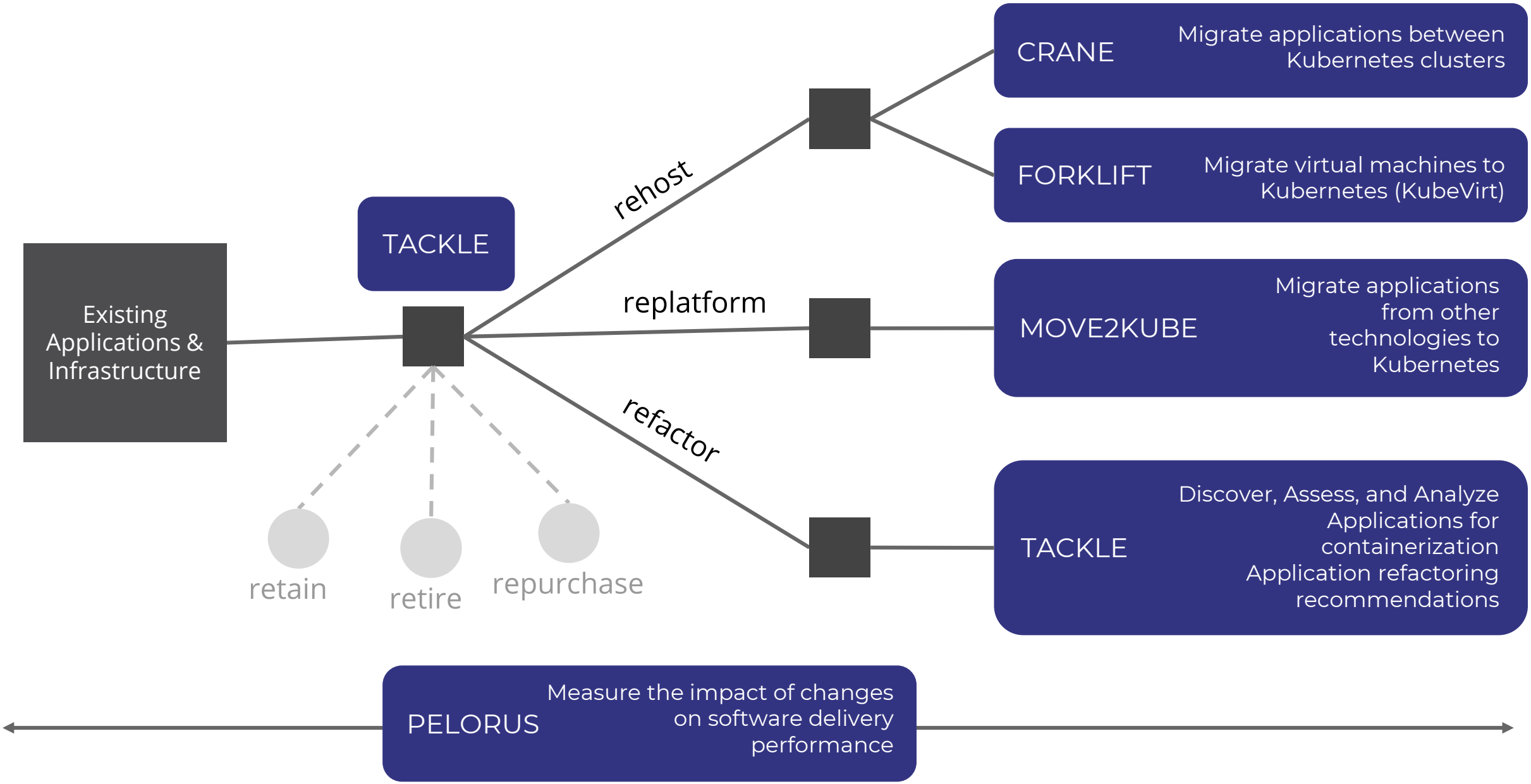
A community of people passionate about helping others modernize and migrate their applications to the hybrid cloud by **building tools and best practices on how to break down monoliths, adopt containers, and embrace Kubernetes.**

www.konveyor.io

Watch The Presentation

https://youtu.be/S8ISWz87rIk

Konveyor Community Projects



Tackle Container Advisor (TCA)

Anup K. Kalia, Jin Xiao, Mihir Choudhury, Lambert P. Wassi, Divya Sankar, AI Ishida, Maja Vukovic
IBM Research Hybrid Cloud

Salai Siva Madhavan, Jayanta K. Mal, Divakar Mysore
IBM GBS

Pavan Kapanipathy, Dinesh Garg, Saswati Dana, Alexander Gray, Salim Roukos
IBM Research AI

August, 2021

Purpose of TCA

Why TCA?

Clients could have a large application portfolio with 1000s or even 10,000s of applications

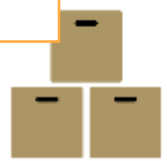
Examples:

- Application 1: cobol java javascript: : , , unix/mainframe, unix/mainframe: unknown , db2 oracle 10g oracle 8i vsam other - db2 v9, , ibm-db2-v9, oracle-oracledb-10g, oracle-oracle db-8i
- Application 2: mq client 7.5 rightfax client 10 sql server 2008 connect direct 4.5, 4.6 & connect direct file agent tws zcentric 8.4, , windows, windows: no, sqlserver
- Application 3: openedge: : tomcat 3/4/5/6/7, , hp unix / rhel / win desktop, hp unix / rhel / win desktop: yes, openedge

Manual verification takes week or even months to go through each application and determine container candidates

TCA provides an automated step that could map applications to container candidates in different catalogs (e.g., DockerHub, Openshift)

This would significantly reduce efforts to determine container candidates and improve in recommendation accuracy



What TCA really does?

Given an application such as the following, TCA recommends container candidates based on a specific catalog. For example, the following

Application 1: cobol java javascript: : , , unix/mainframe, unix/mainframe: unknown , db2 oracle 10g oracle 8i vsam other - db2 v9, , ibm-db2-v9, oracle-oracledb-10g, oracle-oracle db-8i, sql

TCA maps them with candidate container images such as

DockerHub:

1. DB2: https://hub.docker.com/r/ibmcom/db2,
2. Oracle Database: https://hub.docker.com/_/oracle-database-enterprise-edition

Openshift:

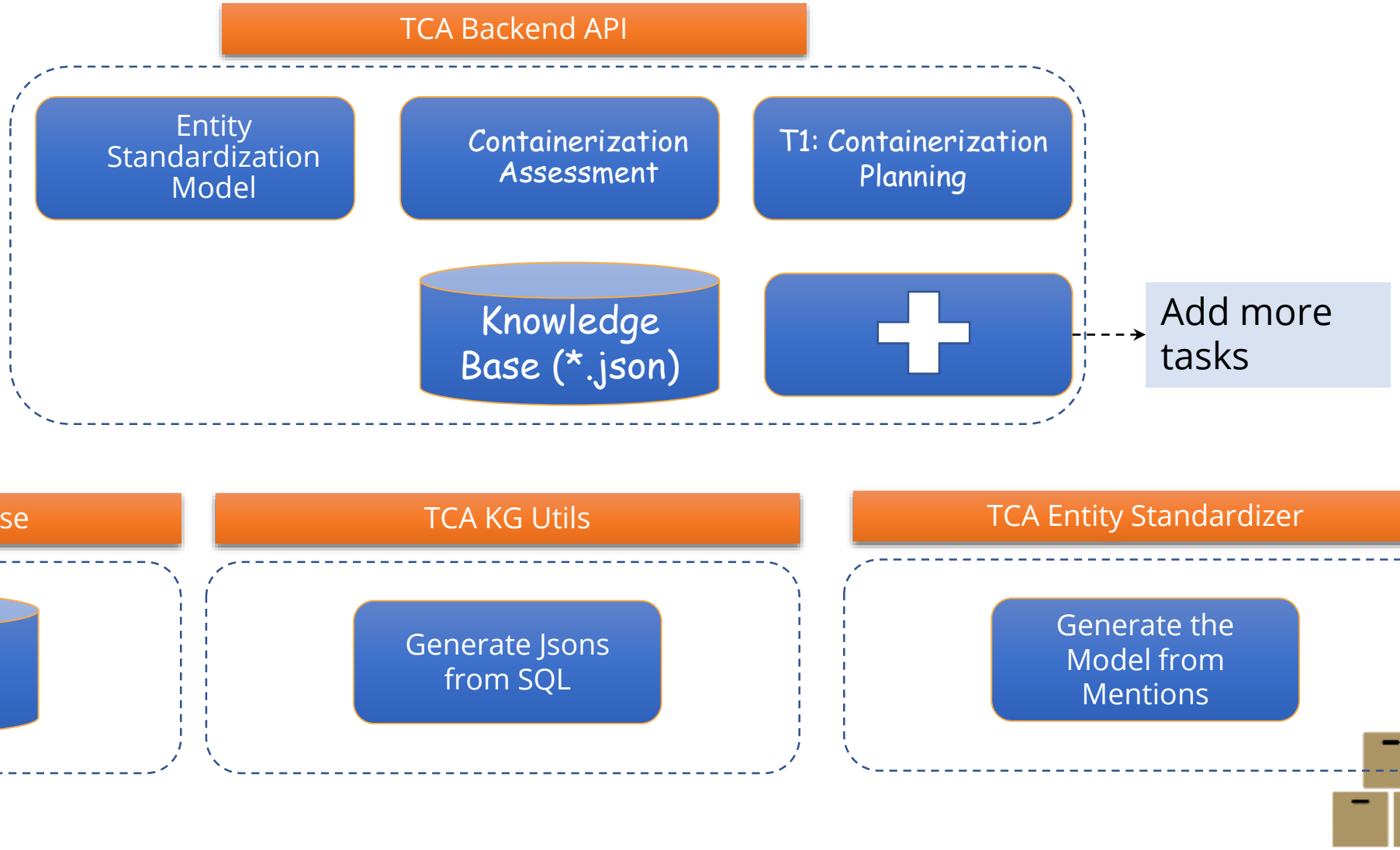
1. DB2: https://catalog.redhat.com/software/applications/detail/823403
2. Oracle Database: https://catalog.redhat.com/software/applications/detail/837443

TCA determines certain other aspects such as.

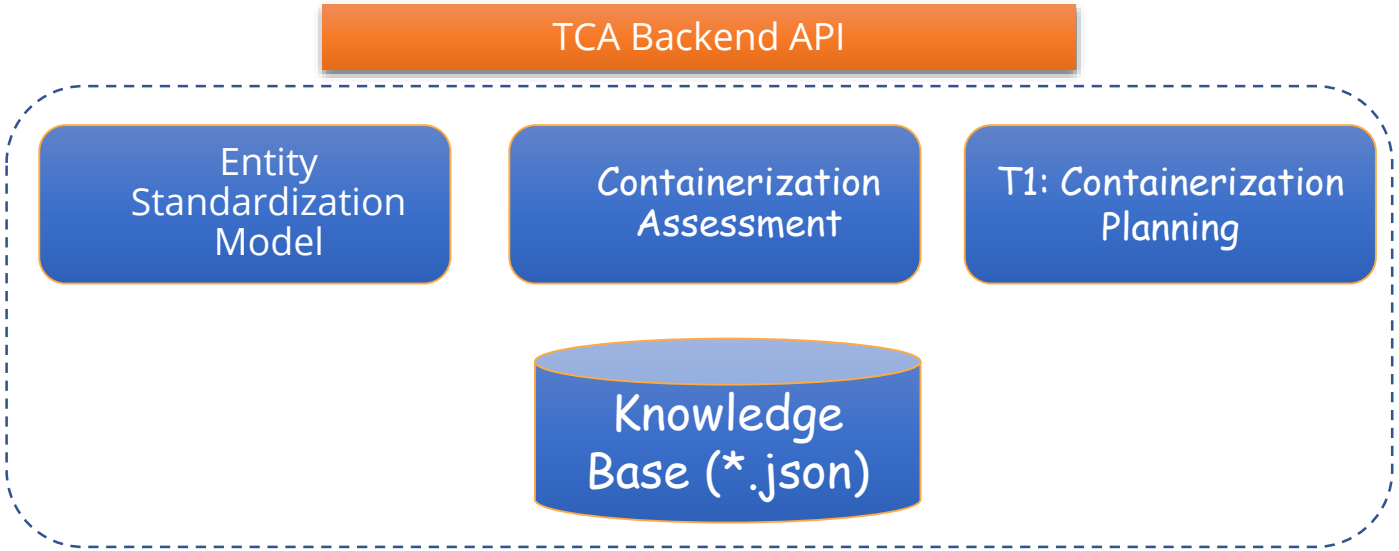
1. **VSAM is Unknown,**
2. **SQL is a General entity, and**
3. **Unix/Mainframe cannot not be mapped to any candidate container,**

Thus, TCA provides an overall picture of what can or cannot be mapped to containers. Thus, TCA is useful for clients generate a broader picture for modernization advisory

Components in TCA



Running TCA Backend API



Two ways to run the backend API

1. git clone the tackle-container-advisor repository
2. cd tackle-container-advisor/ and run the shell script “run.sh”

OR

OR you can cd into the following repository and run the docker command
cd aca_backend_api/
“docker-compose -f 'docker-compose-api.yml' --env-file ./config.ini up -d –build”

Containerization Assessment API Output

POST /containerization-assessment Invoke do_plan method in assessment class to initiate assessment process

Assessment Input

Edit Value | Model


```
[
  {
    "application_name": "App1",
    "application_description": "string",
    "component_name": "string",
    "operating_system": "string",
    "programming_languages": "c#/winforms",
    "middleware": "iis 7.5, iis 8.5",
    "database": "sqlserver",
    "integration_services_and_additional_softwares": "string",
    "technology_summary": "string"
  }
]
```


Assessment Output

Response body


```
{
  "status": 201,
  "message": "Assessment completed successfully!",
  "assessment": [
    {
      "Name": "App1",
      "Desc": "string",
      "Cmpt": "string",
      "OS": "{}",
      "Lang": "{'c#': {'C#': ''}}",
      "App Server": "{'iis 7.5': {'IIS': '7.5'}, 'iis 8.5': {'IIS': '8.5'}}",
      "Dependent Apps": "{'sqlserver': {'MS SQL Server': ''}}",
      "Runtime": "{}",
      "Libs": "{}",
      "Reason": "",
      "KG Version": "1.0.0"
    }
  ]
}
```


server response codes and their descriptions

Code	Description
200	Success
201	Assessment Completed successfully!
400	Input data format doesn't match the format expected by TCA
401	Unauthorized, missing or invalid access token
500	Internal Server Error, missing or wrong config of RBAC access token validation url

The different Reason codes and their description

Code	Description
101	Medium or low confidence for the inferred data
102	General technologies detected
103	Unknown technologies detected

Containerization Recommender API Output

POST /containerization-planning Invoke do_plan method in planning class to initiate planning process

Container Recommendation Input

Edit Value | Model


```
[
  {
      "Name": "App1",
      "Desc": "string",
      "Cmpt": "string",
      "OS": "{}",
      "Lang": "{'c#': {'C#': ''}}",
      "App Server": "{'iis 7.5': {'IIS': '7.5'}, 'iis 8.5': {'IIS': '8.5'}}",
      "Dependent Apps": "{'sqlserver': {'MS SQL Server': ''}}",
      "Runtime": "{}",
      "Libs": "{}",
      "Reason": "",
      "KG Version": "1.0.0"
    }
]
```


Container Recommendation Output

Response body

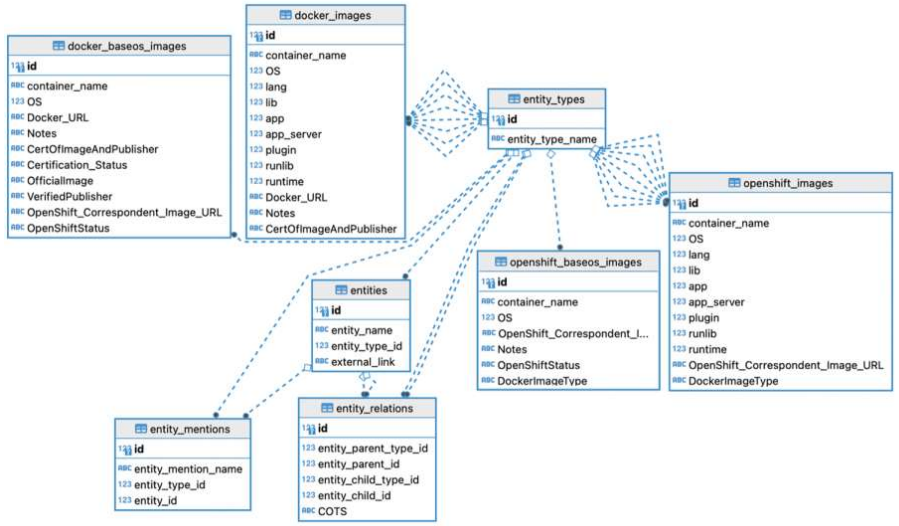

```
{
  "status": 201,
  "message": "Container recommendation generated!",
  "containerization": [
    {
      "Name": "App1",
      "Desc": "string",
      "Cmpt": "string",
      "Valid": true,
      "Ref Dockers": "1. {'Microsoft SQL Server(Verified Publisher)': 'https://hub.docker.com/_/microsoft-mssql-server'}\n2. Windows IIS(Verified Publisher)|https://hub.docker.com/_/microsoft-windows-servercore-iis",
      "Confidence": 0.93,
      "Reason": "Additional Installations in container image 2:C#",
      "Recommend": "Containerize"
    }
  ]
}
```


The different server response codes and their description

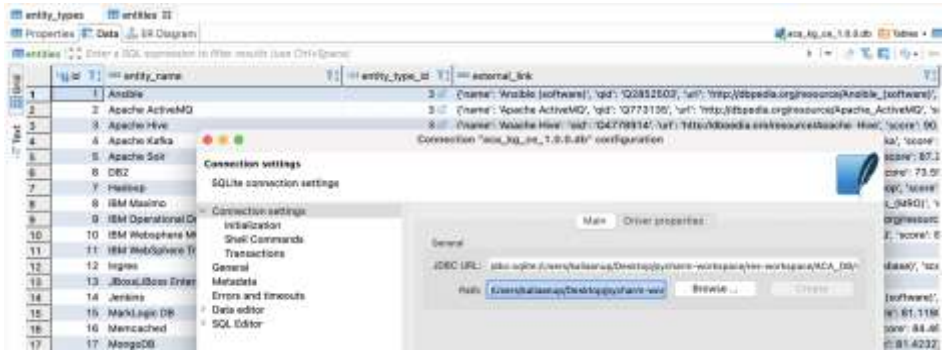
Code	Description	Recommendation
200	Success	Containerize
201	Container recommendation generated!	Partially Containerize
400	Input data format doesn't match the format expected by TCA	[] – No recommendation
401	Unauthorized, missing or invalid access token	
500	Internal Server Error, missing or wrong config of RBAC access token validation url	

Augmenting Knowledge Base

Entity Relationship Diagram of TCA's KG



Upload TCA's KG into Dbeaver as a .db file



Run the SQL queries for adding new entities into TCA KG

- INSERT INTO entity_types(entity_type_name) VALUES(?)
- INSERT INTO entities(entity_name, entity_type_id, external_link) VALUES(?,?,?)
- INSERT INTO entity_mentions(entity_mention_name, entity_type_id, entity_id) VALUES(?,?,?)
- INSERT INTO entity_relations(entity_parent_type_id, entity_parent_id, entity_child_type_id, entity_child_id, COTS) VALUES(?,?,?,?,?)
- INSERT INTO docker_baseos_images(container_name, OS, Docker_URL, Notes, CertOfImageAndPublisher, Certification_Status, OfficialImage, VerifiedPublisher, OpenShift_Correspondent_Image_URL, OpenShiftStatus) VALUES(?,?,?,?,?,?,?,?,?,?)
- INSERT INTO docker_images(container_name, OS, lang, lib, app, app_server, plugin, runlib, runtime, Docker_URL, Notes, CertOfImageAndPublisher) VALUES(?,?,?,?,?,?,?,?,?,?,?,?)

Incorporate updated KG

- Generate a new .SQL file
- Update the existing SQL file with the new file
- Go to the main repository tackle-container-advisor
- Run the shell script "setup.sh"

Add a New KG

To run TCA with a new Knowledge Base, please perform the following steps

1) Replace the existing .sql file with the new <new_db>.sql file in the aca_db folder

1) Change the config.ini file in the aca_entity_standardizer folder as follows
   - db_path = aca_db/<new_db>.db

2) Change the config.ini in the TCA_kg_utils folder as follows
   - db_path = aca_db/<new_db>.db

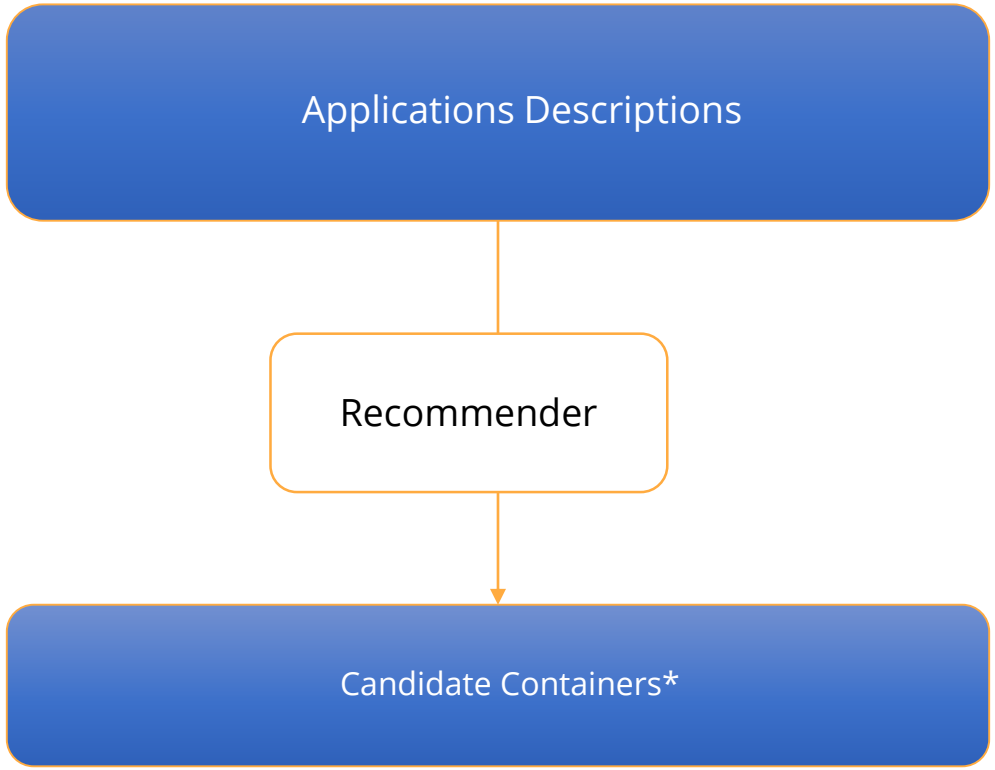
3) Modify the setup.sh script to reflect the sql and db file accordingly
   - aca_sql_file="<new_db>.sql"
   - aca_db_file="<new_db>.db"

4) Run the TCA's environment setup by running the following script
   - sh setup.sh

1) Modify the clean.sh script to reflect the sql and db file accordingly
   - aca_db_file=” <new_db>.db"

Technical Deep Dive into TCA

Challenges



- Inputs will be noisy and hence they need to be standardized
  - They could have abbreviations, numbers, punctuations and symbols (e.g., rhel, ODM, WAS, AS/400)
  - They could have multiple variants (e.g., oracle 10g, oracle-oracle-10g)
  - They could have redundant information (e.g., oracle 10g, oracle-oracle-10g)
  - They could have unknowns that may not have been captured by the model (e.g., considering vsam was unseen)
  - They lack contexts such as surrounding words or entities

Building a recommendation module needs enormous data for almost every downstream task

*Candidate container recommendation is one kind of downstream task, there could be more such as 6R disposition, version recommendation, etc.

Knowledge Base

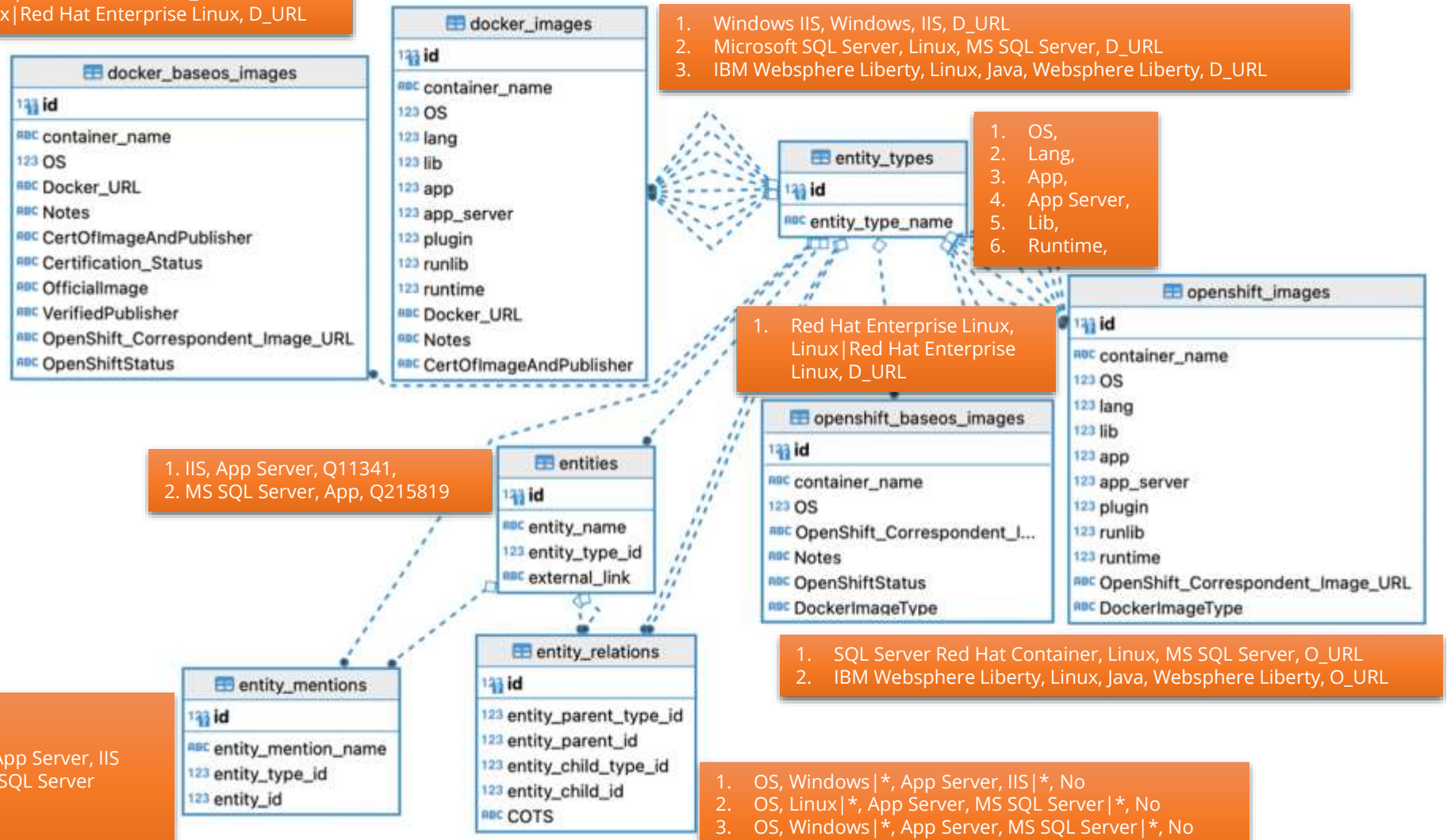
1. Windows Server Core, Windows|Windows Server, D_URL
2. Red Hat Enterprise Linux, Linux|Red Hat Enterprise Linux, D_URL

1. Windows IIS, Windows, IIS, D_URL
2. Microsoft SQL Server, Linux, MS SQL Server, D_URL
3. IBM Websphere Liberty, Linux, Java, Websphere Liberty, D_URL

1. OS,
2. Lang,
3. App,
4. App Server,
5. Lib,
6. Runtime,

1. Red Hat Enterprise Linux, Linux|Red Hat Enterprise Linux, D_URL

1. IIS, App Server, Q11341,
2. MS SQL Server, App, Q215819



1. SQL Server Red Hat Container, Linux, MS SQL Server, O_URL
2. IBM Websphere Liberty, Linux, Java, Websphere Liberty, O_URL

1. IIS, App Server, IIS
2. IIS webserver, App Server, IIS
3. Internet Information Services, App Server, IIS
4. Microsoft SQL Server, App, MS SQL Server
5. MSSQL, App, MS SQL Server
6. MS SQL, App, MS SQL Server
7. SQL Server, App, MS SQL Server

1. OS, Windows|*, App Server, IIS|*, No
2. OS, Linux|*, App Server, MS SQL Server|*, No
3. OS, Windows|*, App Server, MS SQL Server|*, No
4. OS, Unix|*, App Server, MS SQL Server|*, No

Knowledge Base Details

Entity Types: 12, Entities: 610

Types	Entities
App	268
Lang	88
Technology	54
OS	51
Lib	43
App Server	34
Plugin	27
Runlib	15
Runtime	13
HW	9
VM	7
Storage	1

Entities: 447 out of 610 are mapped to Wikidata

Entity Mentions: 4883

Entity Relations: 1267

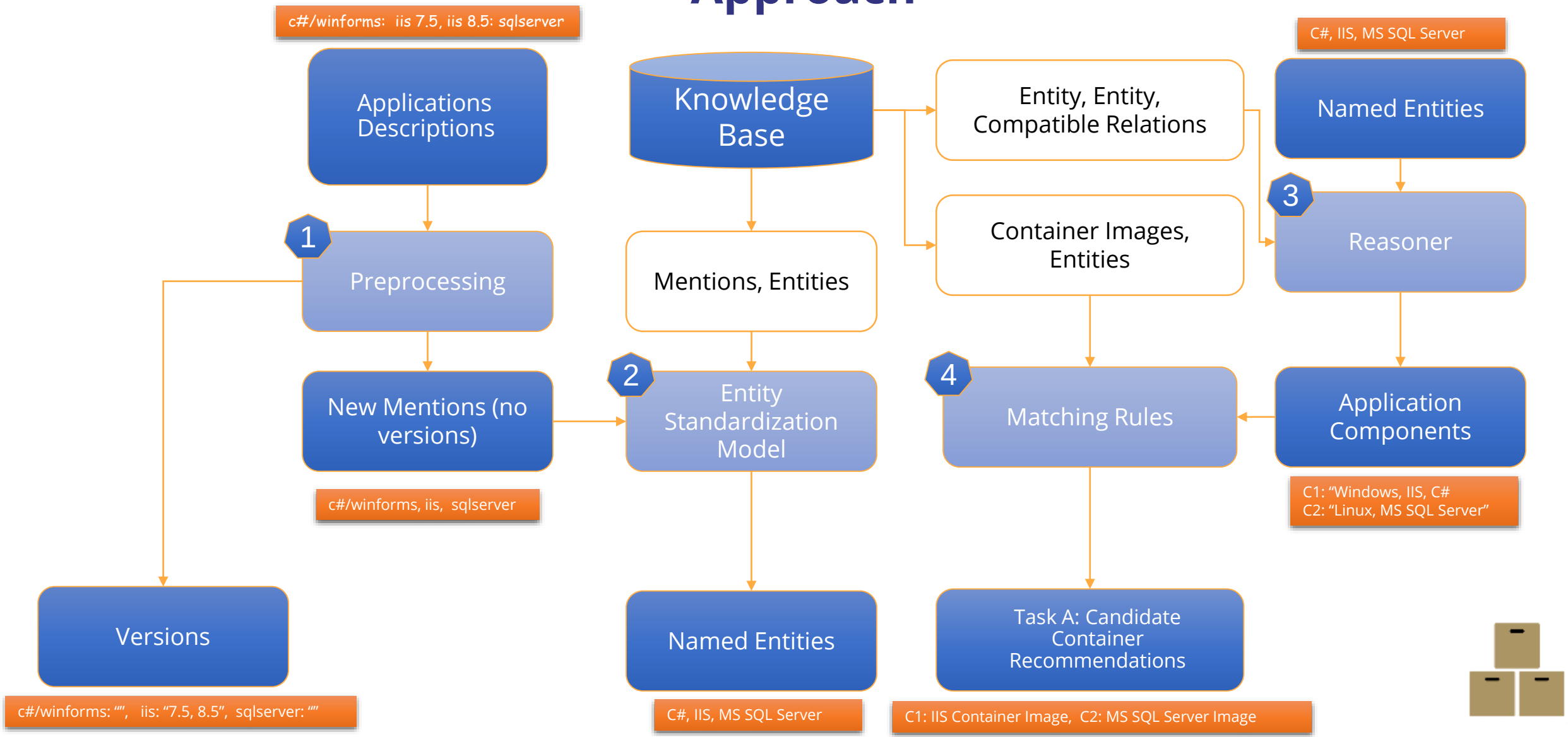
Docker Images: 164

Openshift Images: 54

Docker Base OS Images: 14

Openshift Base OS Images: 1

Approach

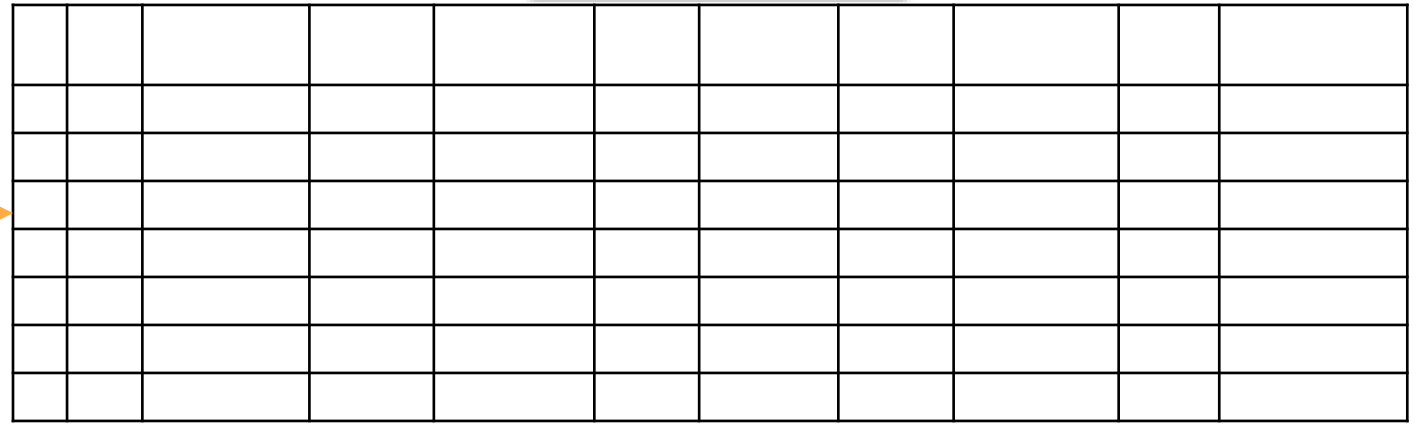

c#/winforms: iis 7.5, iis 8.5: sqlserver
Applications Descriptions
1
Preprocessing
New Mentions (no versions)
c#/winforms, iis, sqlserver
Versions
c#/winforms: "", iis: "7.5, 8.5", sqlserver: ""
Knowledge Base
Mentions, Entities
2
Entity Standardization Model
Named Entities
C#, IIS, MS SQL Server
Entity, Entity, Compatible Relations
Container Images, Entities
4
Matching Rules
Task A: Candidate Container Recommendations
C1: IIS Container Image, C2: MS SQL Server Image
C#, IIS, MS SQL Server
Named Entities
3
Reasoner
Application Components
C1: "Windows, IIS, C#
C2: "Linux, MS SQL Server"

Entity Standardization Model

D1 = ['IIS', 'IIS']
D2 = ['IIS', 'Microsoft IIS Webserver']
D3 = ['IIS', 'Microsoft Internet Information Services']
D4 = ['MS SQL Server', 'Microsoft SQL Server']
D5 = ['MS SQL Server', 'MSSQL']
D6 = ['MS SQL Server', 'MS SQL']
D7 = ['MS SQL Server', 'SQL Server']

fit_transform

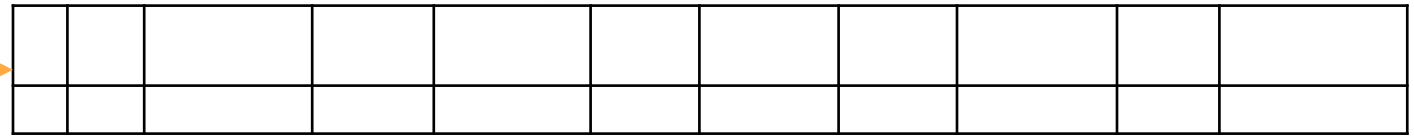
tfidf vectors



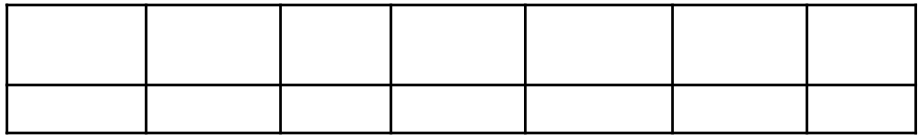
query = ['MS SQL']

transform

query vector



cosine similarity (tfidf, query)



D6 = ['MS SQL Server', 'MS SQL']

'MS SQL Server', 'App'

• Anup K. Kalia, Raghav Batta, Jin Xiao, Mihir Choudhury and Maja Vukovic. *ACA: Application Containerization Advisory Framework for Modernizing Legacy Applications*. IEEE International Conference on Cloud Computing (Cloud) [Work-in-progress], sept, pages 1--3, 2021.

Reasoner

input: c#/winforms: iis 7.5, iis 8.5: sqlserver

Compatibility Relationship

1. 'Windows': ['IIS', 'MS SQL Server', C#],
2. 'Linux': ['MS SQL Server'],

Component Detection

Component 1: 'Windows, IIS, C#',
Component 2: 'Linux, MS SQL Server'

input: cobol java javascript, db2 oracle 10g oracle 8i vsam other - db2 v9, , ibm-db2-v9, oracle-oracledb-10g, oracle-oracle db-8i,

Compatibility Relationship

'Windows': ['DB2', 'Oracle Database', 'COBOL', 'Java', 'JavaScript'],
'Linux': ['DB2', 'Oracle Database', 'COBOL', 'Java', 'JavaScript'],

Component Detection

'Component 1': 'Linux, DB2, COBOL, Java, JavaScript',
'Component 2': 'Linux, Oracle Database, COBOL, Java, JavaScript',

Candidate Container Recommender

Scores Table:

$$\text{Confidence} = \frac{\text{Score Obtained}}{\text{Score Expected}} = \frac{60}{70} = 0.86$$

Demo

Join the Konveyor Community

www.konveyor.io