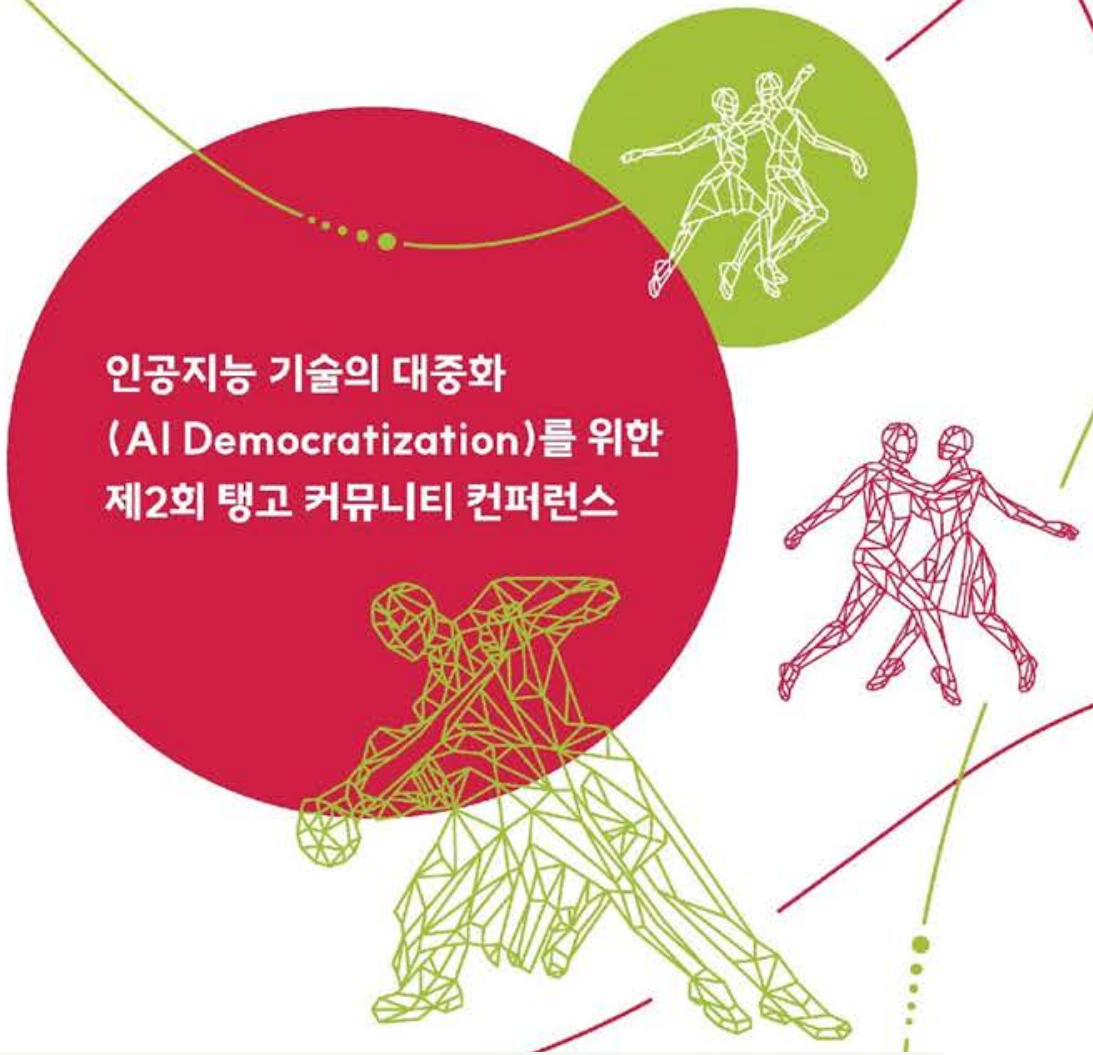
TANGO

TANGO GitHub 개발자 가이드



성명 김홍숙 소속 ETRI

주관 ETRI 한국전자통신연구원 (TANGO) 주최 과학기술정보통신부 IITP 정보통신기획평가원

후원 lablup weda tesla System (사)한국인공지능협회 SNUH 서울대학교병원 고려대학교 KOREA UNIVERSITY 홍익대학교 HONGIK UNIVERSITY 중앙대학교 Georgia Tech

목 차

TANGO based on MSA

- MSA (Micro Service Architecture) using Docker Container
  - structures an application as a collection of services

- A microservice [1]
  - is responsible for a single capability.
  - is individually deployable.
  - consists of one or more processes.
  - owns its own data store.
  - replaceable.

- A small team can maintain a few handfuls of microservices.

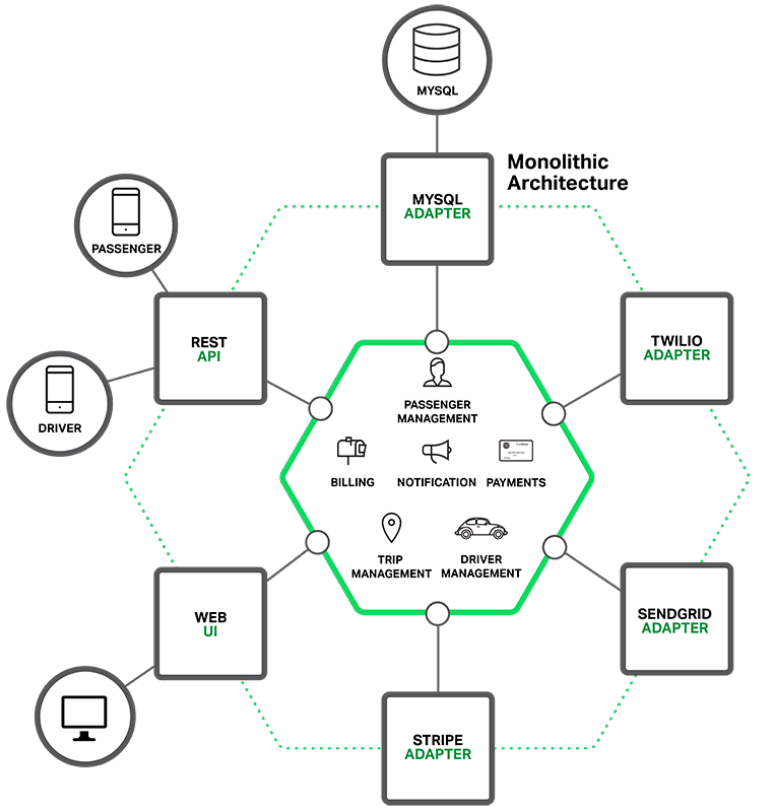
[1] Manning Microservice in .Net, 2021

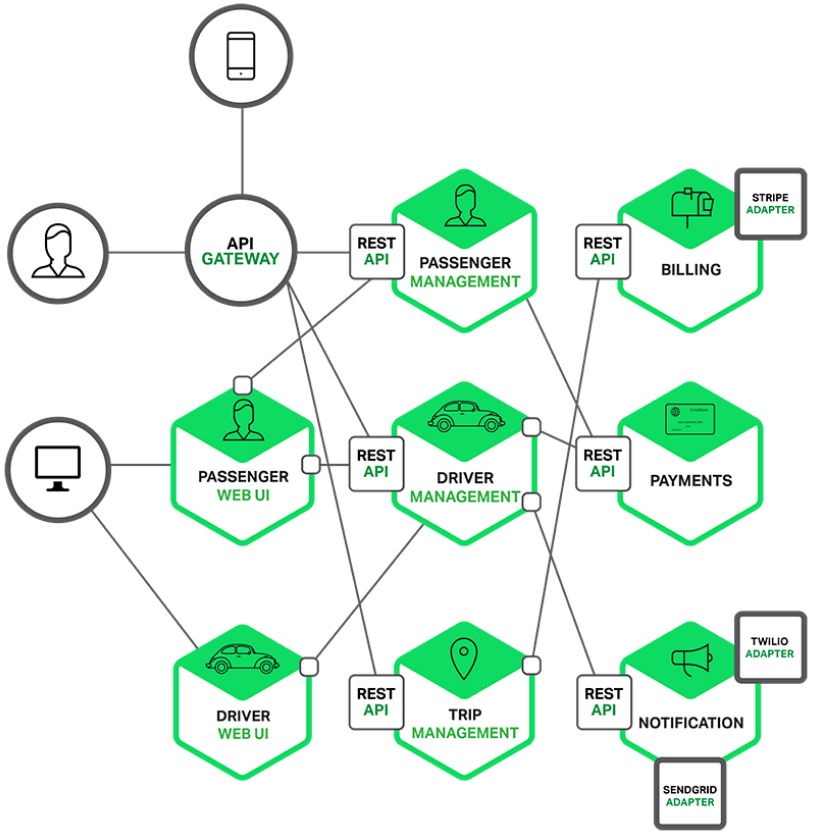
TANGO based on MSA

- Microservices are independently deployable modules[2].
- MSA Benefits[3]:
  - Highly maintainable and testable
  - Loosely coupled
  - Independently deployable
  - Organized around business capabilities
  - Owned by a small team

[2] Manning - Microservices - A Practical Guide on Principles, Concepts, and Recipes 2019
[3] https://microservices.io/

TANGO based on MSA





source: https://www.nginx.com/blog/introduction-to-microservices/

TANGO Wiki

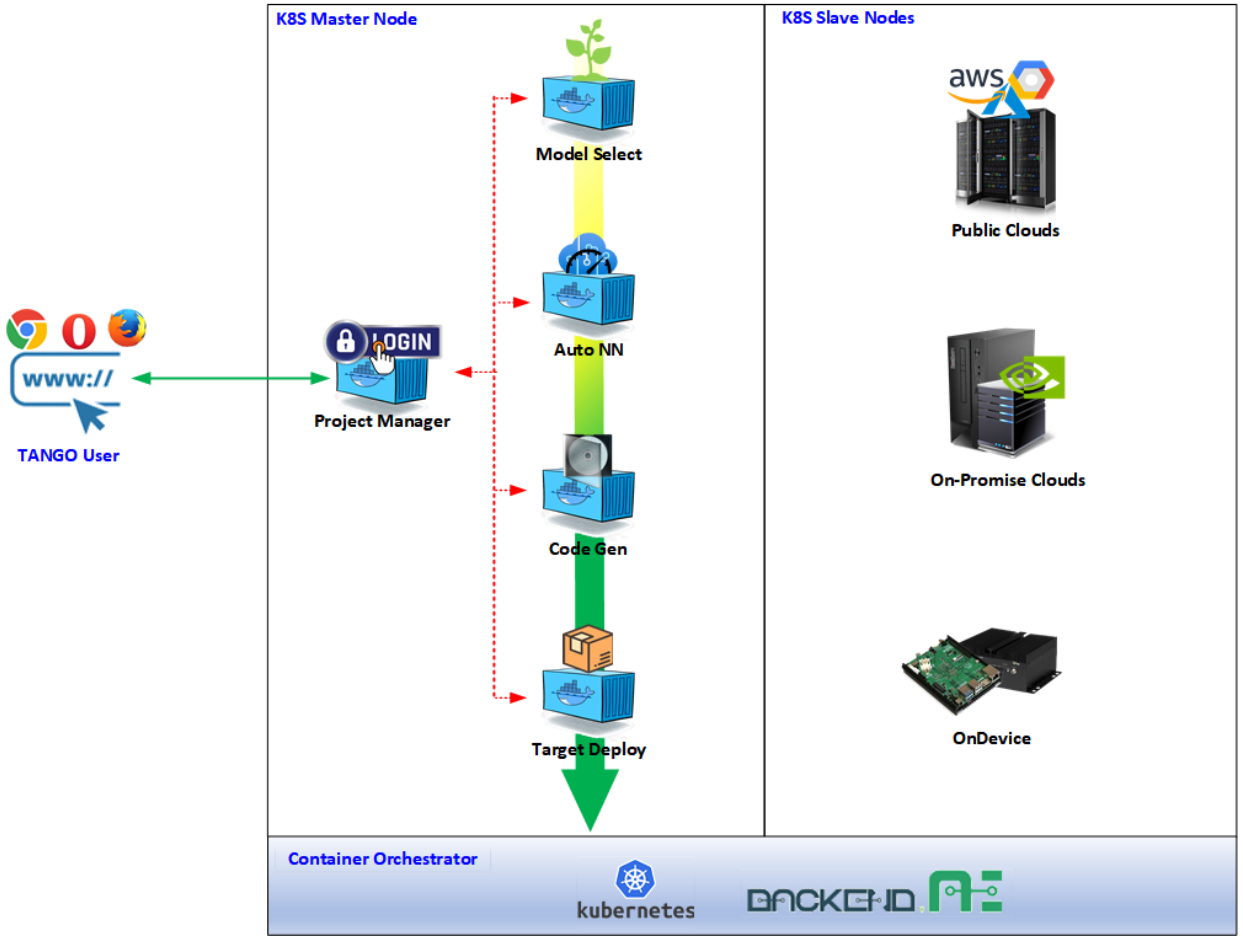
- https://github.com/ML-TANGO/TANGO/wiki
- All Documents maintained as Wiki
  - Guides
    - TANGO Architecture
    - Exchange Data among Containers
    - Rest API
    - Container Port Map
  - HowTo
    - Common HowTos
  - References
    - Git, GitHub, Docker, Docker-compose

1. TANGO Architecture

- Containers in TANGO
  - A main Container: Project Manager
  - Member Containers
- From TANGO Project Creation To Deployment onto Target Devices
- Overall Control Flow in TANGO Project

1. TANGO Architecture

- Containers in TANGO
  - Main Container
  - Member Containers



1. TANGO Architecture

- From TANGO Project Creation To Deployment onto Target Devices
- Overall Control Flow in TANGO Project
  - Project Configuration
    - Member Container Readiness Check
    - Dataset and Target Selection
    - Workflow Definition
  - Project Workflow
    - using REST API: `start()`, `stop()`, `status_request()`, `status_report()`

2. Exchange Data among Containers

Exchange Data among Containers

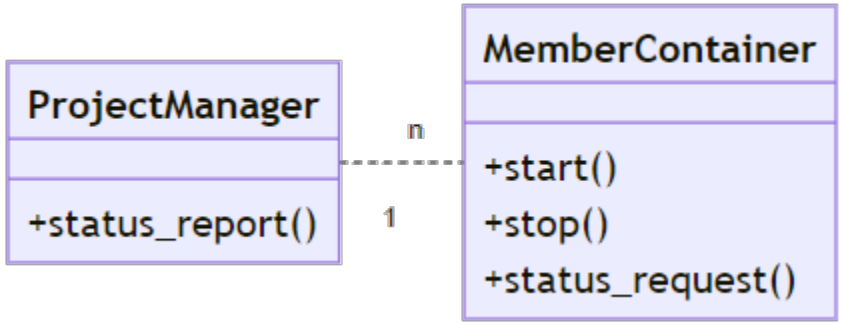
- using Docker host volume
- via volumes: in docker-compose.yaml

```
$ tree ./TANGO/
./TANGO/
└── shared
    ├── common
    │   └── user_id
    │       └── project_id
    │           ├── project_info.yaml     // generated by project manager
    │           ├── data_set.yaml         // generated by labelling
    │           ├── basemodel.yaml        // generated by Base Model Select
    │           ├── model_x.json          // generated by Visualizer
    │           ├── neural_net_info.yaml  // generated by AutoNN
    │           ├── best.onnx             // generated by AutoNN
    │           ├── best.pt               // generated by AutoNN
    │           ├── model.py              // generated by AutoNN
    │           ├── deployment.yaml       // generated by Code_Gen
    │           ├── nn_model              // folder genearted by Code_gen
    │           └── nn_model.zip          // zip of nn_model folder for OnDevice developers
    │
    └── datasets
        └── coco
            ├── train        // folder for train images
            ├── train.txt
            ├── test         // folder for test images
            ├── test.txt
            ├── val          // foler for val images
            └── val.txt
```

3. REST API

REST API

- Exposed API from Main Container
  - `status_report()`
- Exposed API form Member Containers
  - `start()`
  - `stop()`
  - `status_request()`



Note on API provider and consumer

- Don't confuse between API provider and API consumer.

3. REST API

REST API

Core APIs in TANGO

- In HTTP GET Message format

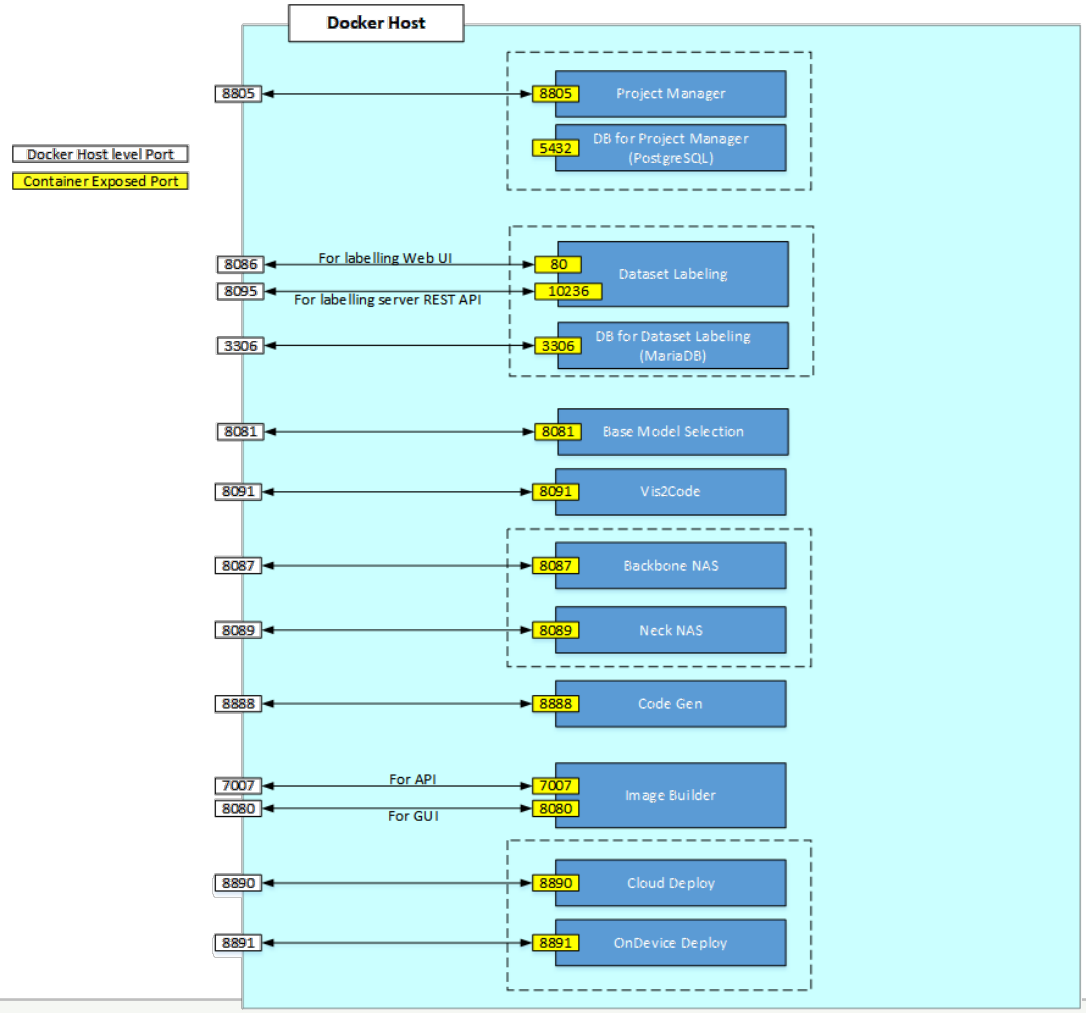
Type	Expose REST APIs
Main	http://<DOCKER_HOST_IP>:<PROJECT_MANAGER_PORT>/**status_report**? container_id=<container_id>&user_id=<user_id>&project_id=<project_id>&status=<status>
Member	http://<DOCKER_HOST_IP>:<MEMBER_CONTAINER_PORT>/**start**? user_id=<user_id>&project_id=<project_id>
	http://<DOCKER_HOST_IP>:<MEMBER_CONTAINER_PORT>/**stop**? user_id=<user_id>&project_id=<project_id>
	http://<DOCKER_HOST_IP>:<MEMBER_CONTAINER_PORT>/**status_request**? user_id=<user_id>&project_id=<project_id>

- All TANGO member container should implement following APIs at least;
  - `start()`,
  - `stop()`, and
  - `status_request()`

4. Container Port Map

Container Port Map

- for identifying the containers
- all containers publish its port
  - via `ports:` in `docker-compose.yaml`



1. How service defined and TANGO app runs?

Dockerfle, Docker Compose, and docker-compose.yaml

- A **`Dockerfile`** is a simple text file that contains the commands a user could call to assemble an image
- **Docker Compose** is a tool for defining and running multi-container Docker applications.
- **Docker Compose**
  - define the services that make up your app in **`docker-compose.yml`** so they can be run together in an isolated environment.
  - gets an app running in one command by just running '*`docker-compose up`*'.
- **Docker compose** uses the `Dockerfile`
  - if you add the *`build`* command to your project's **`docker-compose.yml`** .

IV. How to Build TANGO

```
# Change working directory to TANGO top directory
$ cd ~/work/TANGO

# Build TANGO Docker images and run containers
$ docker-compose up -d --build
or
$ docker-compose up -d

# Cleanup TANGO images, containers and volumes
$ docker-compose down --rmi all --volumes
or
$ docker-compose down --rmi all
```

TANGO

감사합니다.

주관 ETRI 한국전자통신연구원 (TANGO) 주최 과학기술정보통신부 IITP 정보통신기획평가원

후원 UOS lablup weda tesla System (사)한국인공지능협회 Korea Artificial Intelligence Association SNUH 서울대학교병원 고려대학교 KOREA UNIVERSITY 홍익대학교 HONGIK UNIVERSITY 중앙대학교 Georgia Tech