DEPLOYMENT GUIDELINES
AI SERVICE

Document ID	: deployment-guideline-for-ai-service
Version	: 0.1.0
Number of pages	: 15

Revision History

Rev. No,	**Date (YYYY-MM-DD)**	**Add/Delete/Update**	**Section No. changed**	**Changes**	**Author**	**Review by**	**Approved by**
0.1.0	2024-06-04	Add	All	Init document	TienLN		
0.1.1	2024-06-06	Update	Deployment	Infa req	TienLN	HuyTran	
0.1.2	2024-06-17	Update/ Delete		Removed AudioGen information	John M		

Table of contents

1. Architecture overview

1.1 Overview

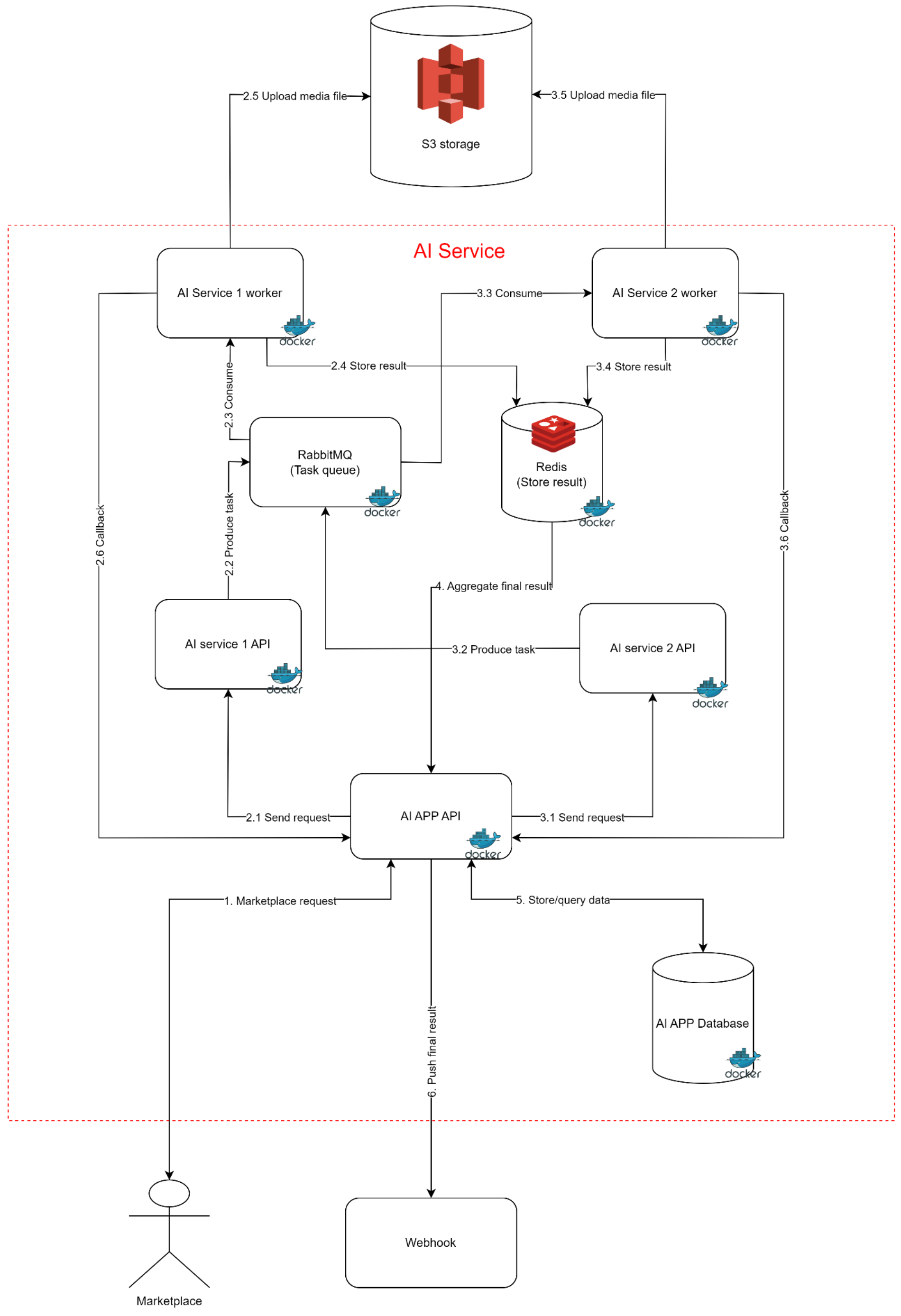

Figure 1. AI app architecture overview

Table 1. Describe component in AI service architecture

No.	Category	Component	Description	AI scope	Note
1	Client	Client	AI marketplace	N	
2	Client callback	Client callback	Get results after a task has been completed	N	
3	AI App API	AI App API	Receive request from client (marketplace) and process. Manage stages to solve a specific task.	Y	
4	Storage	S3 storage	Store media data (Including request data from the client and AI result data). Services send media file metadata to each other	N	
5		AI APP Database	Store user request information for statistics	Y	
6		Redis	Store temporary results of tasks and requests: Will be deleted after a period of time	Y	
7	Queue	RabbitMQ	Store pending tasks	Y	
8	AI service 1	AI service 1 API	Receive requests to process tasks and push tasks into the queue	Y	
9		AI service worker	Pull the task in the queue and process it	Y	
10	AI service 2	AI service 2 API	Receive requests to process tasks and push tasks into the queue	Optional	Whether or not depends on the AI problem
11		AI service worker	Pull the task in the queue and process it	Optional	

With a basic AI problem: AI service will require a minimum of 6 components (docker container). The number of components will increase when the AI problem is complex and needs to be divided into multiple stages for processing

Examples for simple AI problems: Person detection, face detection, face embedding, face searching… For each problem like this, 6 components need to be developed and deployed

For more complex problems such as face recognition, it needs to be processed through many steps: face detection, then face embedding and finally searching. There are up to 3 stages in this problem, so there will be up to 3 AI services. Therefore, the number of components that need to be developed and deployed is 10 (add 2 components for each added service)

2. Package

2.1 Docker image

No.	Component	Docker image name	Type
1	RabbitMQ	rabbitmq:3.8.14-management-alpine	Opensource
2	Redis	redis:6-alpine	
3	AI APP database	postgres:12-alpine	
4	AI APP API	`your_registry_url/your_repo:version`	Self-build
5	AI model API		
6	AI worker		
7	AI mode API		
8	AI model worker		

2.2 Build docker image

Step 1: Clone the project

```
$ git clone https://github.com/any-ai-model
# The Git repository URL may change when delivering the source code to the customer. Carefully check the provided URL.
```

Step 2: Update submodule

```
$ cd any-ai-model
$ submodule update --init --recursive
```

If the source code provided is a compressed file, skip the above two steps

Step 3: Build image
Edit script: ./scripts/build_dockerfile.sh
Update the ECR, and version in ./version.json

```
#!/bin/bash

ECR="your_registry_url/your_repo"
TAG=$(jq -r '.api_version' version.json)
SERVICE="your-ai-app"
IMAGE_NAME="${ECR}:${SERVICE}_${TAG}"

docker build -t $IMAGE_NAME -f dockerfile .
echo  image $IMAGE_NAME is built
```

```
$ cd your-ai-app
```

Build image

```
$ chmod +x ./scripts/build_dockerfile.sh
$ ./scripts/build_dockerfile.sh
```

If no errors occur, the terminal will show

```
$ your_registry_url/your_repo:your-ai-app_0.1.4 is built
```

Step 4: Push image to registry

Step 4.1: Login to registry

Install awscli

```
$ pip install awscli
# or pip3 install awscli
# If you're using a virtual environment created with venv or conda, make
sure it's activated before proceeding.
```

Configure AWS information

```
$ aws configure
# AWS Access Key ID [None]: Enter your access key ID
# AWS Secret Access Key [None]: Enter your secret  access key
# Default region name [None]: Enter region
# Default output format [None]: Can skip it
```

For the next time, there is no need to do the above 2 steps in step 4.1 section
Get the password

```
$ aws ecr get-login-password
# The terminal will generate a string (password)
```

Login repository: Make sure repository has been created

```
$ docker login -u AWS -p <<generated_string_from_above_command>>
<<repository_URL_in_ECR>>
```

Step 4.2: Push docker image
Push docker image

```
$ docker push docker_image_name:tag
```

3. Deployment

3.1 Infrastructure requirements

No.	Category	Item	Required
1	Hardware	CPU	
2		CPU arch	x64
3		RAM	16G
4		NVIDIA GPU	12GB
5		Storage (root)	32GB
6	Software	OS	Ubuntu 20.04 or later
7		Docker	26.x.x or later
8		Docker-compose	1.29.x or later

3.2 Deployment procedure

Step 1: Pull docker image

Step 1.1 Login registry

Do the same as step 4.1 in the package section

Step 1.2 Pull image

```
$ docker pull docker_image_name:tag
# docker_image_name:tag => It is an pushed image in the package section
```

Step 2: Deploy service

Make a deployment directory

```
$ cd ../../../deployment_space_dirctory_path
$ mkdir your-ai-app
$ cd your-ai-app
$ mkdir v0.1.4
$ cd v0.1.4
# v0.1.4: Change to match the version to be deployed
```

Create a docker-compose.yml file

```
$ vim docker-compose.yml
```

Create .env file

```
$ vim .env
```

Run service

```
# make sure you are in the same directory with docker-compose.yml file
$ docker-compose up
```

Check the terminal to determine if any errors occurred. If not, Ctrl + C to stop running service

Run service in detached mode

```
$ docker-compose up -d
```