中国联通 China unicom | UniAI 联通元景

元景DS MaaS软件包安装教学

联通数据智能有限公司

2025年2月

目录

CONTENTS

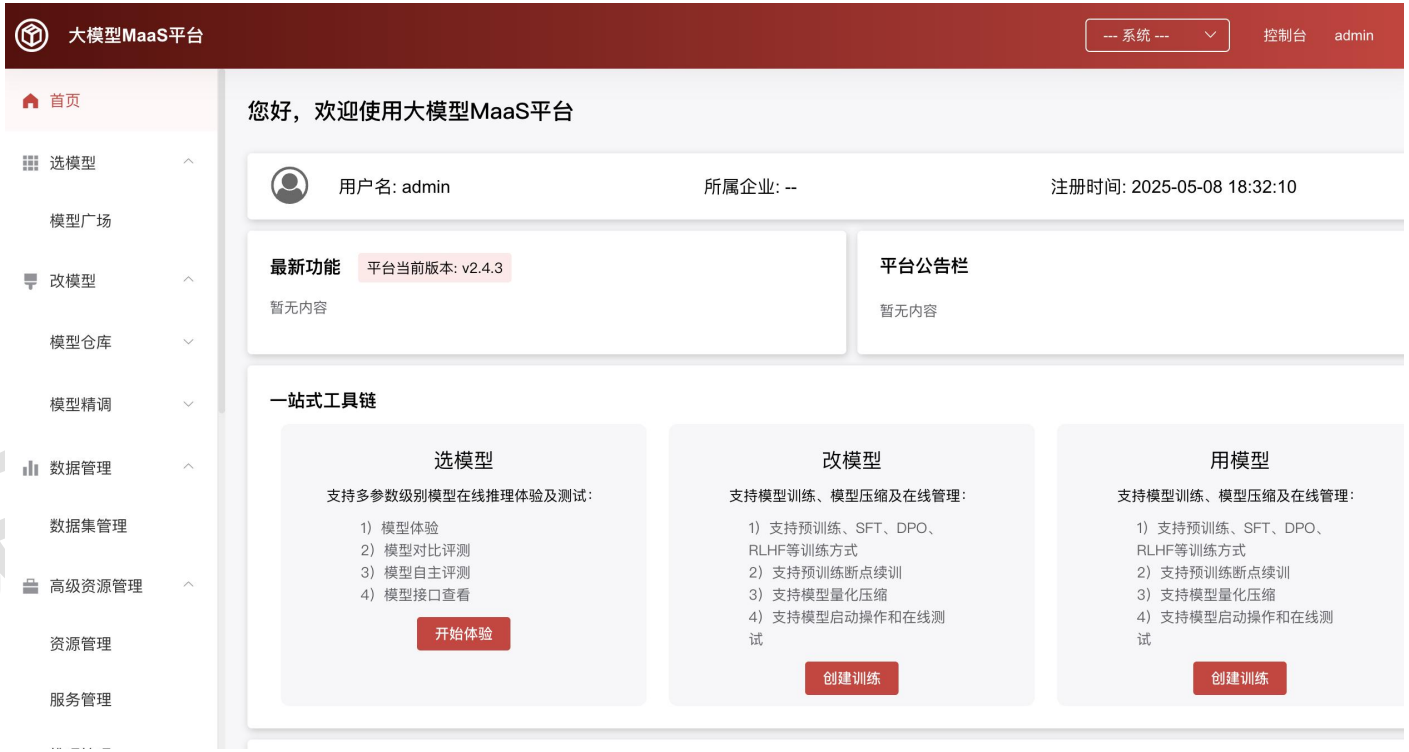
课程安排

5月12日	9:40 - 11: 30	Deepseek 满血版部署 MaaS 软件包部署 配置模型接入
	14:00 - 15:00	Deepseek 蒸馏版部署 MaaS平台功能
	15:10 - 17:10	学员练习：使用堡垒机 + 部署MaaS平台
	19:00 - 20:30	学员练习：启动模型推理 + 验证MaaS平台功能
5月13日	8:30 - 10:00	考试： MaaS 软件包安装（基础） 演示MaaS平台功能（附加）

课程安排

考试内容：

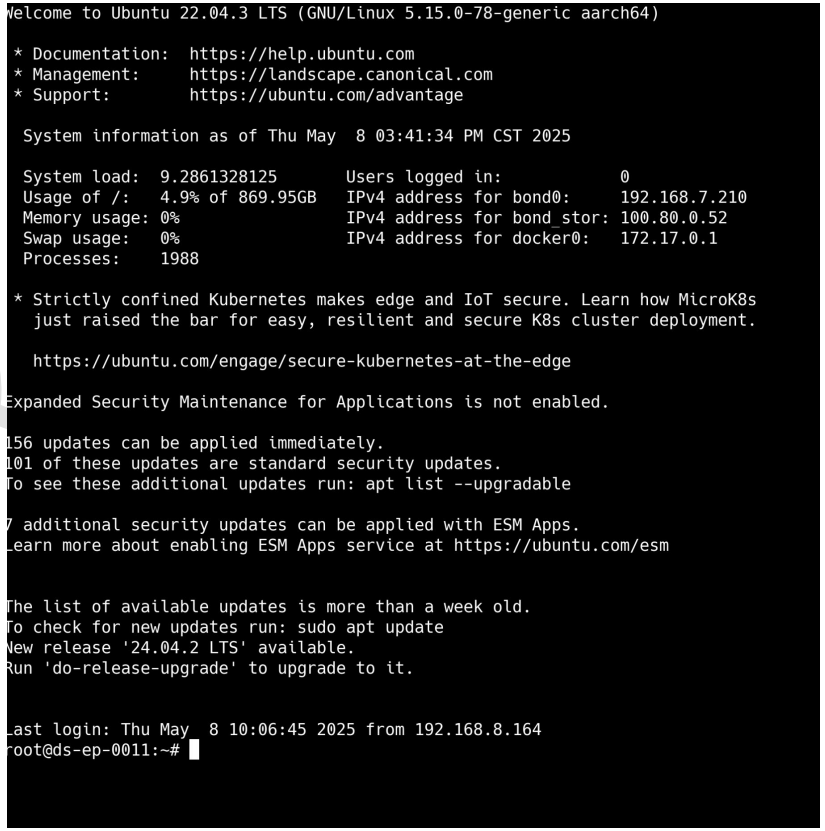
1. 部署maas平台（80%）
2. 演示附加功能（20%）
   - 安全护栏
   - 知识库上传 + 私域rag问答
   - 插件管理
   - 创建原生应用
   - 模型训练
   - 模型接入



Deepseek-R1部署

1. 准备条件：两台机器（e.g. 两台8卡 910B机器）

2. 注意事项：
   - 所有操作均需在两台机器上执行
   - 选择其中一台作为**主节点**，另一台作为**从节点**
   - 两台机器之间可以互相ping通
   - 一旦选定，在配置过程中必须保持一致
   - 提前准备好模型文件及配置文件



Deepseek-R1部署

部署过程：

1. 加载镜像：

docker load -i mindie_2.0.T9.B020-800I-A2-py3.11-openeuler24.03-lts-aarch64.tar

2. 获取npu ip：

for i in {0..7};do hccn_tool -i $i -ip -g; done

记录每张卡的ip地址

3. 配置 ranktable：

进入DeepSeek-R1-w8a8/deepseek_r1_w8a8目录下

vim ranktable.json

正确配置两台机器的 ip 和 npu ip，主节点在前，从节点在后（只需修改 ip 相关信息，所有 id 编号无需修改，两台机器上的 ranktable.json 应完全一致）

4. 修改 daemon_envs.sh：

在 MIES_CONTAINER_IP 字段填入各自的 ip（两台机器上的daemon_envs.sh 除 IP 以外的字段必须保持一致）

Deepseek-R1部署

5. 修改 config.json：

将 ipAddress 和 managementIpAddress 修改为主节点 ip（两台机器上的 config.json 应完全一致）

6. 检查挂载路径

docker_run.sh 中的挂载路径与主机上存放文件的路径是否一致，映射到容器内的路径不要修改，检查镜像版本是否匹配

7. 启动服务：

在两台机器上同时运行 bash docker_run.sh（如分别运行，先执行主节点，后执行从节点，时间差控制在 5 秒内）

8. 查看模型加载情况：

docker logs -f --tail 20 deepseek_r1_w8a8

待出现 daemon start success 字样，则表示启动成功，模型加载需 15 分钟左右

9. 模型接口测试：

curl -X POST http://192.168.0.21:2025/v1/chat/completions -H "Content-Type: application/json" -d '{"model":"deepseek-r1","messages":[{"role":"user","content":"9.8 和 9.11 哪个大？"}],"stream":true}'

MaaS软件包安装

1. 查看系统架构/GPU型号：

arch 指令可查看系统架构，若输出为 **aarch64**，则为 **arm64** 系统；若输出为 **x86_64**, 则为 **amd64** 系统。

查看gpu型号的指令有**npu-smi info**（华为）、**nvidia-smi** （英伟达）

查看系统架构指令	结果	架构
arch	aarch64	arm64
arch	x86_64	amd64

查看GPU型号	适配硬件
npu-smi info	华为 GPU
nvidia-smi	英伟达 GPU

2. 查看磁盘空间：

通过 df -h 找出可用空间最大的数据盘，建议为1T以上；

若数据盘空间不足，可使用 lsblk 查看有无可挂载硬盘进行挂载。

```
(base) root@baremetal-910b-0001:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           151G  8.2M  151G   1% /run
/dev/sda5       870G  359G  477G  43% /
tmpfs           755G  418M  754G   1% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/sda2       452M  126M  292M  31% /boot
/dev/sda1       1.1G  6.4M  1.1G   1% /boot/efi
/dev/nvme0n1    2.9T  2.2T  566G  80% /data
/dev/nvme1n1    2.9T  634G  2.2T  23% /data1
```

```
(base) root@baremetal-910b-0001:~# lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop1     7:1    0  59.5M  1 loop /snap/core20/2503
loop2     7:2    0  77.4M  1 loop /snap/lxd/29353
loop3     7:3    0  79.5M  1 loop /snap/lxd/31335
loop4     7:4    0  59.6M  1 loop /snap/core20/2573
loop5     7:5    0  38.7M  1 loop /snap/snapd/23772
loop6     7:6    0  44.3M  1 loop /snap/snapd/24509
sda       8:0    0 893.8G  0 disk
├─sda1    8:1    0     1G  0 part /boot/efi
├─sda2    8:2    0   500M  0 part /boot
├─sda3    8:3    0     4G  0 part
├─sda4    8:4    0     4G  0 part [SWAP]
├─sda5    8:5    0 884.1G  0 part /
└─sda6    8:6    0    65M  0 part
nvme1n1 259:0    0   2.9T  0 disk /data1
nvme0n1 259:1    0   2.9T  0 disk /data
```

MaaS软件包安装

3. 建立文件夹以及软链接：

- 找到最大的数据盘后，在数据盘下建立/cuai文件夹（假设数据盘挂载在/data目录），并创建软链接。

  mkdir -p /data/cuai

  ln -s /data/cuai /cuai

- 根据版本号在/cuai下创建文件夹（本次部署以2.4.3版本为例）

  mkdir /cuai/maas-2.4.3

4. 建立docker文件软连接（可选）：

若根目录下至少有**800G**空间，可以不建立**docker**软链接，视情况而定

```
sudo systemctl stop docker.service # 停止docker服务
sudo docker info # 查看docker根目录
sudo mv /var/lib/docker /var/lib/docker.bak # 备份docker文件
ln -s /home/docker /var/lib # 创建docker文件软连接
sudo systemctl start docker.service # 重启docker服务
```

MaaS软件包安装

5. 将部署包放至/cuai/maas-2.4.3目录下：

本次培训所用部署包内容如下：

.env.secret：密钥文件

license：存放license的文件夹，里面包括 平台license.license 和 算法license

linux.arm64.aio_std.huawei: 微服务镜像及文件包

plarform-images-250429.tar.gz：platform中间件镜像包

platform.v2.4.3.tar.gz: platform文件包

```
.env.secret
license/
linux.arm64.aio_std.huawei/
platform-images-250429.tar.gz
platform.v2.4.3.tar.gz
```

MaaS软件包安装

6. 解压platform及platform-image压缩包：

进入maas-2.4.3目录下

tar -zxvf platform.v2.4.3.tar.gz

tar -zxvf platform-images-250429.tar.gz -C platform

7. 将 .env.secret 文件放在 platform目录下

cp /cuai/maas-2.4.3/.env.secret /cuai/maas-2.4.3/platform/

8. 修改.env文件：

拷贝.env.bak并改名为.env

cp .env.bak .env

修改.env文件

vim .env

```
CUAI_VERSION=v2.4.3

# amd64 / arm64
CUAI_ARCH=arm64
# aio_std / aio_rag
CUAI_EDITION=aio_std
# 1:nvidia / 2:huawei / 3:aicp / 4:mx
CUAI_PLATFORM=2
# nvidia / huawei / mx
CUAI_PLATFORM_GPU=huawei

# MaaS对外IP
CUAI_EXTERNAL_IP=192.168.0.46

# 前端调用服务端通用前缀（大部分情况为空字符串，
CUAI_REQUEST_COMMON_PREFIX=''
```

MaaS软件包安装

9. 获取平台 license 以及算法 license:

本次培训无需执行：

```
# 进入platform目录下
make plt-sign
# 进入auth/arm64/
cd auth/arm64
# 将其中的apply.apply交给相关人员，获取平台license和算法license
```

注意：本次培训中已经将生成好的license文件放在maas-2.4.3目录中，执行以下操作即可：

```
# 平台license放在该目录下（platform/auth/arm64）
cp /cuai/maas-2.4.3/license/license.license /cuai/maas-2.4.3/platform/auth/arm64/
# 将算法license放在 /conf 目录下
mkdir /conf
cp /cuai/maas-2.4.3/license/license /conf/
```

MaaS软件包安装

10. 创建网络：

make -f Makefile.deploy create-docker-net

创建后可使用docker network ls查看

11. 安装显卡组件（只有华为的显卡才需要）

make -f Makefile.deploy uninstall-npu-exporter

make -f Makefile.deploy install-npu-exporter

12. 一键部署：

make -f Makefile.deploy --no-print-directory deploy-all-in-one

该命令时间较长，请耐心等待，并查看有无报错

进阶内容

deploy命令详解：

deploy-all-in-one:

make -f Makefile.deploy prepare-deploy

make -f Makefile.deploy unzip-all-file

make -f Makefile.deploy unzip-all-image

make -f Makefile.deploy load-all-image

make -f Makefile.deploy run-all-in-one

make update-frontend

在一键部署命令中，综合了设置环境变量，解压各服务压缩包，解压各服务镜像文件，以及加载镜像，最后按顺序启动各个微服务。

进阶内容


```
prepare-deploy:
        echo "vm.max_map_count = 262144" >> /etc/sysctl.conf & sysctl -p
        cp -a bin/${CUAI_ARCH}/docker-compose /usr/bin
        chmod +x /usr/bin/docker-compose
```

主要用于配置系统环境和安装 Docker Compose。

```
echo "vm.max_map_count = 262144" >> /etc/sysctl.conf & sysctl -p
```

- 增加系统允许的最大内存映射区域数量
- 许多服务需要大量的内存映射区域，默认值通常较低，此设置可避免内存映射失败的问题

```
cp -a bin/${CUAI_ARCH}/docker-compose /usr/bin
```

- 将特定架构的 Docker Compose 二进制文件复制到系统可执行路径

```
chmod +x /usr/bin/docker-compose
```

- 为复制到/usr/bin目录下的 Docker Compose 文件添加可执行权限，确保系统可以将该文件作为命令执行

进阶内容

unzip-all-file:

make -f Makefile.deploy unzip-frontend

make -f Makefile.deploy unzip-all-model

make -f Makefile.deploy unzip-agent-file

make -f Makefile.deploy unzip-train-file

make -f Makefile.deploy unzip-data-model-file

该指令分别解压了前端文件，模型文件，agent、train、data-model微服务文件。

```
unzip-frontend:
    @echo "unzip frontend from: frontend"
    @find frontend -type f -name "*.tar.gz" | while read gzfile; do \
        tar -zxvf $$gzfile -C frontend; \
    done
```

进阶内容


```
unzip-all-model:
    mkdir -p $(CUAI_PROJECT_DIR)/aimodel/static/bigmodel/models
    @echo "unzip models from: ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/models"
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/models -type f -name "*.tar" | while read tarfile; do
        tar -xvf $$tarfile -C $(CUAI_PROJECT_DIR)/aimodel/static/bigmodel/models; \
    done
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/models -type f -name "*.tar.gz" | while read gzfile;
        tar -zxvf $$gzfile -C $(CUAI_PROJECT_DIR)/aimodel/static/bigmodel/models; \
    done
    chmod -R 750 $(CUAI_PROJECT_DIR)/aimodel/static/bigmodel/models

unzip-agent-file:
    mkdir -p $(CUAI_PROJECT_DIR)/extend
    @echo "unzip agent files from: ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/agents/files"
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/agents/files -type f -name "*.tar.gz" | while read gz
        tar -zxvf $$gzfile -C $(CUAI_PROJECT_DIR)/extend; \
    done
```

进阶内容

unzip-all-image:

```
make -f Makefile.deploy unzip-platform-image
make -f Makefile.deploy unzip-agent-image
make -f Makefile.deploy unzip-train-image
make -f Makefile.deploy unzip-infer-image
make -f Makefile.deploy unzip-compress-image
make -f Makefile.deploy unzip-data-model-image
make -f Makefile.deploy unzip-monitor-image
```

该指令分别解压了各服务的镜像文件。

进阶内容


```
unzip-platform-image:
    @echo "unzip platform images from: image/${CUAI_ARCH}"
    @find image/${CUAI_ARCH} -type f -name "*.gz" | while read gzfile; do \
        tar -zxvf $$gzfile -C image/${CUAI_ARCH}; \
    done

unzip-agent-image:
    @echo "unzip agent images from: ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/agents"
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/agents -maxdepth 1 -type f -name "*.gz" | while read
        tar -zxvf $$gzfile -C ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/agents; \
    done

unzip-train-image:
    @echo "unzip train images from: ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/trains"
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/trains -maxdepth 1 -type f -name "*.gz" | while read
        tar -zxvf $$gzfile -C ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/trains; \
    done

unzip-infer-image:
    @echo "unzip infer images from: ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/infers"
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/infers -maxdepth 1 -type f -name "*.gz" | while read
        tar -zxvf $$gzfile -C ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/infers; \
    done
```

进阶内容

load-all-image:

make -f Makefile.deploy load-platform-image

make -f Makefile.deploy load-agent-image

make -f Makefile.deploy load-train-image

make -f Makefile.deploy load-infer-image

make -f Makefile.deploy load-compress-image

make -f Makefile.deploy load-data-model-image

make -f Makefile.deploy load-monitor-image

该指令加载了各服务的镜像文件（解压后）

进阶内容


```
load-platform-image:
    @echo "load platform images from: image/${CUAI_ARCH}"
    @find image/${CUAI_ARCH} -type f -name "*.tar" | while read tarfile; do \
        docker load -i $$tarfile; \
    done

load-agent-image:
    @echo "load agent images from: ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/agents"
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/agents -maxdepth 1 -type f -name "*.tar" | while read
        docker load -i $$tarfile; \
    done

load-train-image:
    @echo "load train images from: ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/trains"
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/trains -maxdepth 1 -type f -name "*.tar" | while read
        docker load -i $$tarfile; \
    done

load-infer-image:
    @echo "load infer images from: ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/infers"
    @find ../linux.${CUAI_ARCH}.${CUAI_EDITION}.${CUAI_PLATFORM_GPU}/infers -maxdepth 1 -type f -name "*.tar" | while read
```

进阶内容


```
run-all-in-one:
    # --- middleware ---
ifeq ($(CUAI_DB_NAME), tidb)
    echo "Starting TiDB components..."
    make run-tidb & sleep 20
    make init-tidb-password & sleep 10
    make init-tidb-data & sleep 10
    make run-sensitive-tidb & sleep 20
    make init-sensitive-tidb-password & sleep 10
    make init-sensitive-tidb-data & sleep 10
else ifeq ($(CUAI_DB_NAME), mysql)
    echo "Starting MySQL components..."
    make run-mysql & sleep 20
    make run-mysql-worker & sleep 20
    make run-sensitive-mysql & sleep 20
endif
    make run-redis & sleep 20
    make run-minio & sleep 20
    make run-etcd & sleep 20
    make run-kafka & sleep 20
    make run-kong-pg & sleep 60
    make run-kong-pg-worker & sleep 30
    make run-kong & sleep 20
    make run-konga-pg & sleep 60
    make run-konga-pg-worker & sleep 30
    make run-konga & sleep 20
    make run-es-setup & sleep 20
    make run-es & sleep 20
    make run-kibana & sleep 20
    make run-daemon & sleep 20
    make run-prometheus-node-exporter & sleep 5
    make run-prometheus & sleep 5
```

进阶内容


```
    # --- microservice ---
ifeq (${CUAI_ARCH}, arm64)
    make run-model-calculate-similarity-service & sleep 10
    make run-model-referee-evaluation-service & sleep 10
    make run-model-safe-evaluation-service & sleep 10
    make run-model-safety-2classification-service & sleep 10
    make run-model-safety-enhance-service & sleep 10
endif
    make run-model-data-clean-service & sleep 10
    make run-model-sensitive-word-service & sleep 10
    make run-plugin & sleep 10
    make run-auth-service & sleep 20
    make run-iam & sleep 10
    make run-infer & sleep 10
    make run-app-record & sleep 10
    make run-operate & sleep 10
```

平台功能演示

中国联通 China unicom | UniAI 联通元景

深耕场景致胜
自主淬炼固本