

一、概述

OPENI是一基于启智开源项目的集群管理工具和资源调度平台。平台提供了一系列接口，能够支持主流的深度学习框架，如CNTK, TensorFlow等。

OPENI分管理用户和普通用户，该手册主要供普通用户使用。用户首先创建任务，提交任务至平台执行。用户可通过任务列表查看已有任务，并通过ssh工具或tracking page查看和管理运行中的任务。

OPENI会帮助用户管理硬件资源，调度资源创建容器，运行用户任务。

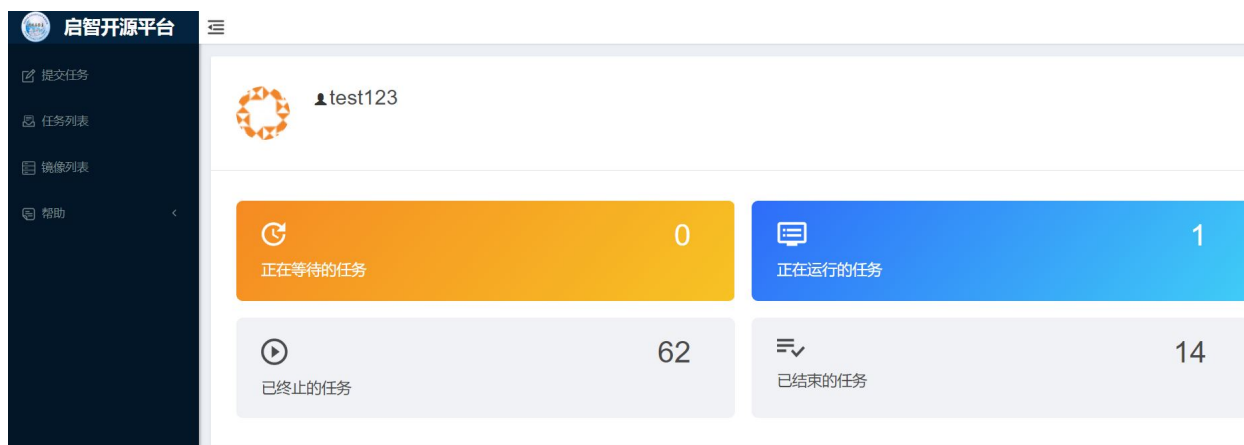
二、使用说明

前提：假定用户已成功部署OPENI平台（部署文档请参考.....），OPENI对外访问地址为<http://192.168.113.221:9286>，用户可使用浏览器访问该地址，推荐浏览器为chrome。用户可请求管理员注册账号（包括用户名及密码），如用户名=test123，密码=123456。

1. 浏览器中打开OPENI地址，进入login界面，输入用户名密码后点击登录：



2. 进入登录页后可看到用户当前任务面板，包括当前用户不同状态下任务的总数。左侧边栏包括提交任务、任务列表、镜像列表和帮助。



3. 提交任务，参数除了minSucceededTaskCount、minFailedTaskCount和retryCount，其余为必填项。具体用法参见第三节任务类型。

The 'Create Task' form includes the following fields and sections:

- 项目名称 (Project Name):** Name for the job, need to be unique. Value must match the pattern `^[A-Za-z0-9-_-]+$.`
- 镜像 (Image):** A dropdown menu to select a container image.
- retryCount:** Job retry count, job will automatically restart when it failed.
- Task1 Section:**
 - Name:** name
 - task number:** Value must be greater than 0.
 - minSucceededTaskCount:** null or no less than 1
 - minFailedTaskCount:** null or no less than 1
 - cpu number:** Value must be greater than 0.
 - gpu number:** no less than 0
 - memory:** Value must be at least 100. MB
 - Shared memory:** no more than memory size. MB
 - command:** Executable command for tasks in the task role, can not be empty. Value must match the pattern `^$+.`
- Buttons:** 导入 (Import), 导出 (Export), 提交 (Submit)

4. 参数填写完毕后，点击提交后，即完成第一个任务的提交。

后续步骤为以提交任务的查看和管理：

5. 可点击任务列表，查看当前用户的所有任务

The 'Task List' page shows a table of tasks for user test123. The table has columns: Job, User, Virtual Cluster, Start Time, Duration, Retries, Status, Stop, and Resubmit.

Job	User	Virtual Cluster	Start Time	Duration	Retries	Status	Stop	Resubmit
test-q1ha3s	test123	default	2018/12/04 上午10:47:22	7m 0s	0	Stopped	Stop	Resubmit
test-km0ko	test123	default	2018/12/04 上午10:44:42	2m 20s	0	Stopped	Stop	Resubmit
test-b60w4l	test123	default	2018/12/04 上午10:39:33	4m 28s	0	Stopped	Stop	Resubmit
test-pc7wbj	test123	default	2018/12/04 上午10:32:55	4m 45s	0	Stopped	Stop	Resubmit
test-bllfcj	test123	default	2018/12/04 上午10:32:08	32s	0	Stopped	Stop	Resubmit
test-j95io	test123	default	2018/12/04 上午10:30:29	30s	0	Stopped	Stop	Resubmit
mxnet-distributed-001-jnhg8o	test123	default	2018/12/04 上午10:27:57	30s	0	Failed	Stop	Resubmit
mxnet-distributed-001-je9h47	test123	default	2018/12/04 上午10:23:19	48s	0	Failed	Stop	Resubmit
mxnet-distributed-001-n9xzh9	test123	default	2018/12/04 上午09:28:43	43s	0	Failed	Stop	Resubmit
mxnet-distributed-001-qsebkx	test123	default	2018/12/03 下午20:15:06	44s	0	Failed	Stop	Resubmit
mxnet-distributed-001-ngxs4k	test123	default	2018/12/03 下午20:06:03	47s	0	Failed	Stop	Resubmit
mxnet-v6hesx	test123	default	2018/12/03 下午19:47:06	11m 4s	0	Succeeded	Stop	Resubmit
mxnet-distributed-001-15y9h4	test123	default	2018/12/03 下午19:38:52	37s	0	Failed	Stop	Resubmit
mxnet-7e6fwd	test123	default	2018/12/03 下午19:33:16	33s	0	Succeeded	Stop	Resubmit

6. 可点击镜像列表，查看平台提供的各种镜像，用于创建用户的容器



三、任务类型：

1、提交单一任务

1) 登录系统后，用户需要先提交任务，提交任务有两种方式：a、通过导入json文件提交，b、在页面填写

- 复制以下内容保存为 `job.json` 文件

```
{
  "jobName": "JOBNAME",
  "image": "192.168.113.221:5000/user-images/deepo:v2.0",
  "taskRoles": [
    {
      "name": "demo",
      "taskNumber": 1,
      "cpuNumber": 4,
      "memoryMB": 8196,
      "gpuNumber": 1,
      "command": "sleep TIME (该值需要修改)"
    }
  ]
}
```

```
}
```

其中，`JOBNAME` 用户自定义，提交时会自动增加后缀；`TIME` 自定义，一旦提交，该容器申请的资源为用户独占。

参数述说：

jobName	任务的名称，全局唯一
image	任务引用镜像
taskRole	当前任务中的角色，list列表，可添加多个
taskRole.minFailedTaskCount	最小失败任务数量，当taskRole中任务失败的个数大于等于该值时，判定该job为失败
taskRole.minSucceededTaskCount	最小成功任务数量，当taskRole中任务成功的个数大于等于该值时，判定该job为成功
taskRole.name	taskRole的名称
taskRole.taskNumber	当前任务角色中任务的个数（可理解为启动相同任务的容器个数）
taskRole.cpuNumber	当前任务的CPU个数（单个容器CPU个数）
taskRole.memoryMB	当前任务的内存（单个容器内存）
taskRole.gpuNumber	当前任务的GPU个数（单个容器GPU个数）
taskRole.command	当前任务需要运行的命令

- 选择左侧边栏 `提交任务`，点击 `choose file`（英文系统）或 `选择文件`（中文系统）选择保存的job.json文件后点击右下方 `提交` 即可，点击 `提交` 前，用户也可在界面修改提交的任务内容
- 若用户需要自定义job.json文件，可以参考下文 `准备提交脚本`，当前实现，`command` 中命令执行完成后，容器会自动销毁

四、任务查看

SSH to container_1543732416493_0004_01_000002

Step 1: Open a Bash shell terminal.

Step 2: Copy and save private key as a file named application_1543732416493_0004:

```
-----BEGIN RSA PRIVATE KEY-----
MIIIEowIBAAKCAQEAsPB35R11GwL/41eZ1VxEOnRVZ/sF+f4jiXPTWeVRHcmM962
pXo7vOdJF3Ms7mJEmIyJc/6feSHYxmGPof07IkSz+tJyrBXMqKi98GoZntbOmaLm
6ZsbQBwqjCeToxh0Zu3lT/XKlaaDNZj7y4mzMzGZ4VtMLOx0bYzrTwRGZuuKXDMs
9RP10fLRQoZzNA0xDhp3WoC4Wf+D6GT31wTiPfZL2qVOnzYLYC0cDUghSe6aq5mf
nz45ZBDcT8PbB4N87bGFQ50Wuq4r/nwu0pNkN6IDj5IAG/2h0Gn19ThfsMyri7d
okLWTjEN52TZ01FD0GBZAS+TRhnde/GlQ9A+8wIDAQABoIBAG7Wl3SdMEc8aMij
j1St2Ht5diqNf1/rR54DORl2wnVB4edqIqWj3RnnE80hs8qcOfYJSu0jWghKI2c
jZnYkKfVu5xqCbUsbmCuJNS10djDHGX30oipRbeyLSDghcYLwpbbfOLKyB1YUibX
7xq/5kYyUv9s7lrWvRs6tXzippa9PStqB22GHmPR2Ao+txal0s0IjQlN6TNsN96
3fE6sTuMLIoBZF08sC3Z59AhtdzN8AtytRaktEBXMZRXiltznTA6UMijeM4hgk73
HZm9dRuLoRe1ps+eEhoKVhsQV67TRTn7QxJbVtgEAJK+R1ttEzK03CtYVYZtoW5O
mJfIbPECgYEA2nGwZB65KSjZn51xZGRV+Rn2Kcl5aJOnWIpoKYd51002riHf1N+
PEvuKCBvArdwPJQZoxntm/E/88joGntD1lqfhdqwxWa2kEAmuYGvI9VfV5UP/am
DwKgr1Dk/cuCE/uHka8v4CFW09+GvAAD0Xa4U6k/Fr+g8erXSXEhH0sCgYEAz1wL
BuT7juvvifnYvqKVh0zAHYyIlgYixNsgz4kypJJYHJ4n1WA1k17NFQy8JdJBoAom
itoGHB+D0KpFYG9m0Jr+L7Tsus8/Gs13Ik961eysZVpSlzrtiA8jXlX4byOJh7oA
pLthiOVnM/I2JBm60FbQ4P5W4xzABbmZPQjGDfKcGyB4b5jTaW5m3lBrfKpXEEGd
XMSq8NPkYQB6H33XhP0DZNBVrFk847a6Qxn+oTlSCpMjsnoXwtmNmQiqu13KC56N
K1S6lH+39rcyD5fnzVx1Y4eCikhkIvQVXl1tCNCsobiM+JPLuS+hW5cpab+Z1ioQ8
j8i6xplzTx3JgLBdWszp1QKBgQCIAtyswgeLIJC3V01pZaV4N/mweZZWV729rCX
0GsqaLXapnAKOIbbwRf+UF0IWywN9g+HXICgl1owjbNYWT0EaYgZsq0YfoHbPw6
wh1VMm+F72/bCGkDZMKTPxq8c3fjUgrYAouSbL574TTICyRxQXCzE8EUKSh+2NV9
pkKaqQKBGHEsxMDWnvZukJ0ZuVpS8cS9tIj/5//NAZ4IHOalNV3PQY358dccq5/M
PJC449cdjXC12YKvope7eAhR+rtbcDAKISmNHfEAbAELYjkMyuXWT8F7KnYwz7QC
ygAmFArijJfkWcPsZqOf5IGDb0tnJ4vz2g36Hwky1PgiCe30DaD
-----END RSA PRIVATE KEY-----
```

Step 3: Set correct permission for the key file:

```
chmod 600 application_1543732416493_0004
```

Step 4: Connect to the container:

```
ssh -i application_1543732416493_0004 -p 10014 root@192.168.113.226
```

Close

- 使用ssh工具进行登录 linux/mac下按照sshinfo的步骤操作：

第一步：复制上图step2中的密钥并保存在本地文件命名 application_1530350083555_0723

第二步：修改文件权限：`chmod 600 application_1530350083555_0723`

第三步：连接容器：`ssh -i application_1530350083555_0723 -p 10014 root@192.168.113.226`

windows下：

第一步：打开cmd，创建密钥文件 `application_1530350083555_0723`（同linux方法）

第二步：使用修改密钥文件权限：

`cacls application_1530350083555_0723 /G user:F` 其中，`application_1530350083555_0723` 为文件名称，`/G` 指赋予用户指定权限，`user`为当前操作系统用户名称，`F`为全部权限。

第三步：登录 `ssh -i application_1530350083555_0723 -p 10014 root@192.168.113.226`

- 使用ssh工具进行数据和源代码传输

linux下使用SCP命令传输数据：

`scp -i your_ssh_key -P your_ssh_port your_file ssh_username@ssh_ip:/userhome/` windows下使用winscp软件进行传输，[参考方法](#)

注：每个用户都有唯一路径/userhome，用户可将个人资料、代码、数据放置在该路径下，云主机其他路径下内容会在重启时删除且无法恢复。

II、提交多任务

1) 页面提交，

The screenshot shows the 'Create Task' (创建任务) page. The 'Project Name' (项目名称) is 'tensorflow-distributed-012-jd9cq5'. The 'Image' (镜像) is '192.168.113.221:5000/user-images/deepocv2.0'. The 'retryCount' is 0. The 'Name' is 'worker', 'task number' is 3, 'minSucceededTaskCount' is 2, and 'minFailedTaskCount' is 1. The 'cpu number' is 2, 'gpu number' is 2, 'memory' is 16384 MB, and 'Shared memory' is 'no more than memory size'. The 'command' field contains a complex shell script for running TensorFlow distributed training. The 'worker' role is selected in the task roles list.

其中增加task后，即可在创建任务时生成多个容器，完成多容器任务

2) json文件提交

```
{
  "jobName": "tensorflow-distributed",
  "image": "192.168.113.221:5000/user-images/deepo:v2.0",

  "taskRoles": [
    {
      "name": "ps",
      "taskNumber": 2,
      "cpuNumber": 2,
      "memoryMB": 8192,
      "gpuNumber": 0,
      "minSucceededTaskCount": 2,
```



```
    "minFailedTaskCount": 1,
    "command": "python /gdata/tensorflow-distributed/code/mnist_replica.py --
num_gpus=0 --batch_size=32 --data_dir=/gdata/tensorflow-distributed/data --
train_dir=/userhome/tensorflow-distributed/output --
ps_hosts=$PAI_TASK_ROLE_ps_HOST_LIST --worker_hosts=$PAI_TASK_ROLE_worker_HOST_LIST
--job_name=ps --task_index=$PAI_CURRENT_TASK_ROLE_CURRENT_TASK_INDEX"
  },
  {
    "name": "worker",
    "taskNumber": 2,
    "cpuNumber": 2,
    "memoryMB": 16384,
    "gpuNumber": 2,
    "minSucceededTaskCount": 2
    "minFailedTaskCount": 1,
    "command": "python /gdata/tensorflow-distributed/code/mnist_replica.py --
num_gpus=2 --batch_size=32 --data_dir=/gdata/tensorflow-distributed/data --
train_dir=/userhome/tensorflow-distributed/output --
ps_hosts=$PAI_TASK_ROLE_ps_HOST_LIST --worker_hosts=$PAI_TASK_ROLE_worker_HOST_LIST
--job_name=worker --task_index=$PAI_CURRENT_TASK_ROLE_CURRENT_TASK_INDEX"
  }
],
"retryCount": 0
}
```

具体可参见[github](#)

三、自定义镜像

平台提供的云主机不能满足个人需求时，可以根据个人需求创建自定义镜像：

用户可以参考[dockerhub](#)教程，打包镜像到dockerhub，提交任务时指明来自于dockerhub的path即可，比如：若使用deepo的镜像，可以填写 `ufoym/deepo`。也可以将dockerfile上传到平台共有路径，并通知管理员build镜像并加入镜像列表即可。

四、更多经典案例

更多案例可以参考[github](#)，其中包括入门级案例和进阶分布式案例等，也可参考项目。