

基于阿里云云效Flow的python Web服务框架与部署实验

实验目的

- 理解并掌握 Python 虚拟环境的应用价值与管理操作。
- 熟练掌握 Docker 命令与 Dockerfile 构建逻辑，能够基于 Dockerfile 创建容器镜像。
- 熟练掌握阿里云云服务器 ECS、阿里云云效代码管理 Codeup 与流水线 Flow，以及阿里云容器镜像服务 ACR 的相关配置与管理操作。
- 理解并掌握 Python 项目的多种部署方式，初步体验流水线的应用模式。

实验要求

本实验提供一个简易的 Python Web 服务范例。该服务基于 Python Sanic 框架构建实现了设备基本信息的可视化展示。要求学生下载代码后跟随实验步骤完成下实验内容：

- 首先，分别以「虚拟环境(venv)」形式与容器形式在阿里云云服务器 ECS 上直接部署范例服务。
- 随后，基于阿里云云效流水线 Flow，分别以「虚拟环境(venv)」形式与容器形式部署范例服务，初步体验流水线在 DevOps 持续集成/持续部署(CI/CD)中的重要性作用。

通过以上实验内容，综合应用阿里云云服务器 ECS、云效代码管理 Codeup、云效流水线 Flow，初步搭建体验 DevOps CI/CD 环境。

实验环境

- 本地实验环境: Git, python3, Python Venv, Sanic
- 阿里云云服务ECS: docker, python3, Python Venv, Sanic
- 阿里云云效: 代码管理Codeup, 流水线Flow
- 阿里云容器镜像服务(ACR)
- Python Web服务范例[exp_pyms_demo.tar.gz](#)

实验步骤

初始范例代码

- 服务范例[exp_pyms_demo](#) 文件组织结构

```
.  
├── devices.csv      # 存放设备信息的 csv 文件  
├── requirements.txt # 运行所需安装的 Python 库  
└── server.py        # Python Sanic 服务执行文件，服务使用8000端口启动  
└── static           # 静态文件夹，包含 html, js 和 css 文件  
    ├── index.html    # 网页页面，URL 为 <ip>:8000/  
    ├── index.js      # JavaScript 文件  
    └── style.css     # 样式表文件
```

1 directory, 6 files

构建实验环境

准备本地开发环境

在本地环境安装Python第三方库

在本地工作目录 `exp_pyms_demo` (先创建本地工作文件夹`exp_pyms_demo`, 然后`cd`进入) 创建 Python 虚拟环境 `pyms_venv` , 使用`source` 命令激活虚拟环境。

Python 虚拟环境可以实现为不同的项目设置独立的依赖库，以实现在统一系统中不同项目所需软件包的隔离。例如开发环境中的两个项目都依赖同一个软件包，但依赖于不同的版本，或者一些依赖是相互冲突的，那么使用 Python 虚拟环境就能为不同的项目创建独立的虚拟环境。

```
python3 -m venv pyms_venv # 创建虚拟环境  
source pyms_venv/bin/activate # 激活环境
```

配置 `requirements.txt` 文件，提供要安装的 Python 库名，本实验仅需 Python Web 服务框架 `Sanic`。

根据 `requirements.txt` 文件安装第三方库。

输入以下指令

```
touch requirements.txt
```

打开`requirements.txt` 写入一下内容:

```
sanic==20.12.4
```

然后执行以下指令:

```
pip install -r requirements.txt
```

```
beryl@beryl-virtual-machine:~/exp_pyms_demo$ python3 -m venv pymv_venv
beryl@beryl-virtual-machine:~/exp_pyms_demo$ source pymv_venv/bin/activate
(pymv_venv) beryl@beryl-virtual-machine:~/exp_pyms_demo$ touch requirements.txt
(pymv_venv) beryl@beryl-virtual-machine:~/exp_pyms_demo$ pip install -r requirements.txt
Collecting sanic==20.12.4
  Downloading sanic-20.12.4-py3-none-any.whl (80 kB)
[██████████] 80.2/80.2 kB 570.0 kB/s eta 0:00:00
Collecting http tools>=0.0.10
  Using cached http tools-0.6.4-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (442 kB)
Collecting websockets<10.0->=0.1
```

准备云服务器ECS

(已实现)

在ECS安装docker引擎和python3

(已实现)

基于阿里云容器镜像服务ACR配置镜像仓库

(已实现)

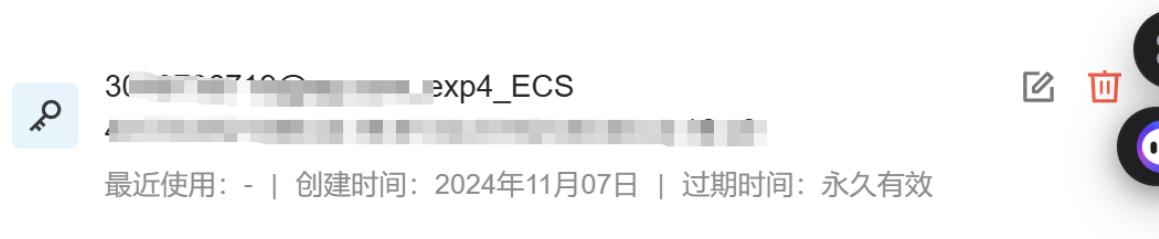
基于阿里云云效代码管理Codeup创建代码库

- 在Codeup新建远程仓库
 - 访问 [云效代码管理 Codeup](#) 产品首页，登陆后若首次使用，则需创建或者加入一个企业。
 - 新建代码库，命名为 `exp_pyms_demo`，新建时请注意勾选「创建.gitignore」，并选择Python模版。
 - 新建后，Codeup 默认创建 Master 分支并提供推荐 .gitignore 文件。
- 配置SSH访问Codeup远程仓库
 - 根据 [如何配置SSH密钥及自定义SSH认证密钥的路径-云效-阿里云帮助中心](#)，分别在本地环境与 ECS 服务器上创建 SSH 公钥私钥，并在 Codeup 对应页面添加公钥，保证本地环境与 ECS 服务器均可基于 SSH 访问 Codeup 远程仓库。本地环境在之前的实验已经配置了SSH，现在只需要在 ECS 上配置SSH即可。

```
root@iz2vcfk3gmwgxxap6tntf5Z:~# ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -C "3049736719@qq.com"
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:sppJ48cu4mVX6FLCgI2FjT6Wmexcjh+Ha8KpIdPiSAw 3049736719@qq.com
The key's randomart image is:
+--[ED25519 256]--+
| +.
| o=.
| = *
| 0 + .
| E = + + S
| o= + = +
| *o+ X.+
|=B.O.Bo
|+o+=+.
+---[SHA256]---+
root@iz2vcfk3gmwgxxap6tntf5Z:~# ls ~/.ssh/
authorized_keys  id_ed25519  id_ed25519.pub  known_hosts  known_hosts.old
root@iz2vcfk3gmwgxxap6tntf5Z:~# cat ~/.ssh/id_ed25519.pub
ssh-ed25519 [REDACTED] [REDACTED] [REDACTED] [REDACTED] kbc 3049736719@qq.com
m
```

已有公钥 · 3

[如何同时使用多个SSH Key?](#)



基于阿里云云服务器ECS直接部署范例服务

以下操作均在阿里云ECS服务器上进行

获取范例服务包并解压

如下图所示，利用ssh命令将`exp_pyms_demo.tar.gz`文件从github远程仓库下载到ECS服务器上。

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~# ssh -T git@github.com
Hi xinyuyaoer! You've successfully authenticated, but GitHub does not provide shell access.
root@iZ2vcfk3gmwgxxap6tntf5Z:~# git clone git@github.com:xinyuyaoer/exp4_file.git
Cloning into 'exp4_file'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
root@iZ2vcfk3gmwgxxap6tntf5Z:~# ls
beryl_exp_pyms_data exp4_file snap
root@iZ2vcfk3gmwgxxap6tntf5Z:~# cd exp4_file
root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file# ls
exp_pyms_demo.tar.gz
root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file# tar -xzvf exp_pyms_demo.tar.gz
exp_pyws_demo/
exp_pyws_demo/static/
exp_pyws_demo/static/index.js
exp_pyws_demo/static/style.css
exp_pyws_demo/static/index.html
exp_pyws_demo/devices.csv
exp_pyws_demo/server.py
exp_pyws_demo/requirements.txt
root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file# cd exp_pyws_demo
root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file/exp_pyws_demo# ls -F
devices.csv requirements.txt server.py static/
```

在虚拟环境部署范例服务

创建Python虚拟环境`venv`并安装`requirements.txt`指定的相关依赖:

```
cat requirements.txt # 查看所需依赖
python3 -m venv exp_venv # 创建虚拟环境
source exp_venv/bin/activate # 激活虚拟环境
pip3 install -r requirements.txt # 安装所需依赖
```

```
No VM guests are running. Outdated hypervisor (qemu) binaries on this host.
root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file/exp_pyws_demo# python3 -m venv exp_venv
root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file/exp_pyws_demo# ls
devices.csv exp_venv requirements.txt server.py static
root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file/exp_pyws_demo# source exp_venv/bin/activate
(exp_venv) root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file/exp_pyws_demo# pip3 install -r requirements.txt
Looking in indexes: http://mirrors.cloud.aliyuncs.com/pypi/simple/
DEPRECATION: The HTML index page being used (http://mirrors.cloud.aliyuncs.com/pypi/simple/sanic/) is not a proper HTML 5 document. This is in violation of PEP 503 which requires these pages to be well-formed HTML 5 documents. Please reach out to the owners of this index page, and ask them to update this index page to a valid HTML 5 document. pip 22.2 will enforce this behavior change. Discussion can be found at https://github.com/pypa/pip/issues/10825
Collecting sanic
  Downloading http://mirrors.cloud.aliyuncs.com/pypi/packages/76/88/4c61ced275fa8978775e4380c423250161470ef5b418e94685128f161102/sanic-24.6.0-py3-none-any.whl (244 kB)
    244.9/244.9 KB 15.8 MB/s eta 0:00:00
DEPRECATION: The HTML index page being used (http://mirrors.cloud.aliyuncs.com/pypi/simple/sanic/)
```

启动Python Web服务:

```
python3 server.py
```

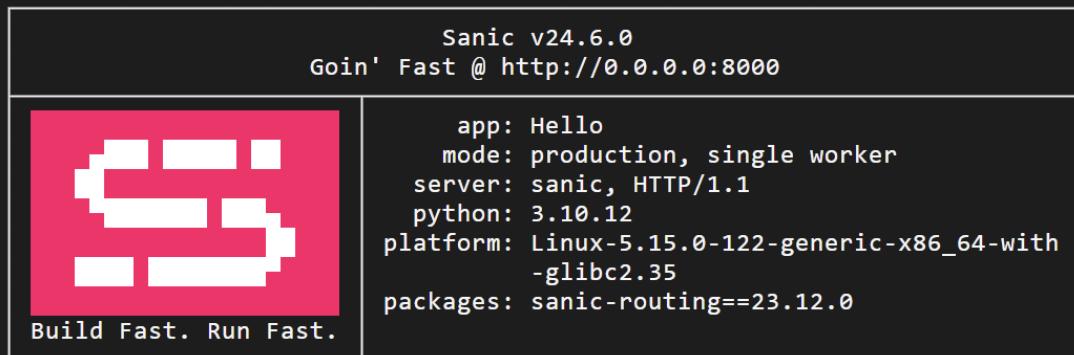
遇到`UnboundLocalError: local variable 'worker_state' referenced before assignment` 问题，这有可能是8000端口被占用了

解决方案 你可以通过以下命令查找占用端口 8000 的进程，并终止它：

```
sudo lsof -i :8000
sudo docker ps
sudo docker stop <容器ID>
```

然后重复前面的命令，成功安装sanic。

```
(exp_venv) root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file/exp_pyws_demo# python3 server.py
Main 2024-11-07 17:29:15 +0800 INFO:
```



```
Main 2024-11-07 17:29:15 +0800 WARNING: Sanic is running in PRODUCTION mode. Consider using
--debug' or '--dev' while actively developing your application.
Srv 0 2024-11-07 17:29:17 +0800 INFO: Starting worker [121781]
```

选择某台能访问 ECS IP 的终端设备，使用浏览器访问`<IP>:8000/`，查看服务页面输出。

```
# 查看公网ip的指令
curl ifconfig.me
```

Welcome to Devices Page

Devices Information

Device ID	Name	Type	Hardware Model	Hardware SN	Software Version	Software Last Update	Network Type	Network MAC	Network IPv4	Status
7094306256256503808	device1	camera	Raspberry Pi 4	00001	1.1	2023-07-10 12:05:23	WiFi	00:00:00:00:00:02	10.0.0.2	alive
7094306256256503809	device2	server	Raspberry Pi 4	00005	5.4	2023-07-13 21:05:23	5G	00:00:00:00:00:08	101.2.15.8	offline
7094306256256503810	device3	client	PC	acd5d3	11	2023-07-15 21:05:23	eth	00:00:00:00:00:12	192.168.1.123	alive

查看无误后，键入`ctrl+c`停止服务。随后退出虚拟环境`venv`，删除虚拟目录

```
deactivate # 退出虚拟环境
rm -r exp_venv # 删除
```

以容器形式部署范例服务

在 `exp_pyws_demo` 目录下创建 `Dockerfile` 文件，将 Python Web 服务范例代码打包进容器镜像。

```
# 创建Dockerfile文件  
touch Dockerfile  
  
# 编写Dockerfile文件  
nano Dockerfile
```

在 `Dockerfile` 文件中，编写如下内容：

```
# 基于 Python 最新基础镜像  
FROM python:3.10-slim-buster  
LABEL maintainer=<hyx_beryl@qq.com>  
  
# 工作目录  
WORKDIR /app  
  
# 安装python库  
COPY requirements.txt requirements.txt  
RUN apt-get update && apt-get install -y \  
    && pip3 install -r requirements.txt -i  
    https://pypi.tuna.tsinghua.edu.cn/simple  
  
# 清理 apt 软件包缓存  
RUN rm -rf /var/lib/apt/lists/*  
  
# 复制当前文件夹中全部文件到镜像中  
COPY ..  
  
# 暴露服务端口 8000  
EXPOSE 8000  
  
# 容器启动时执行命令  
CMD ["python3", "server.py"]
```

使用 `docker build` 命令构建镜像，完成后通过 `docker images` 可查询。

```
docker build -t exp_pyws_demo .  
docker images
```

```
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp4_file/exp_pyws_demo# nano Dockerfile
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp4_file/exp_pyws_demo# docker build -t exp_pyws_demo .
[+] Building 489.6s (11/11) FINISHED                                            docker:default
=> [internal] load build definition from Dockerfile                         0.0s
=> => transferring dockerfile: 572B                                         0.0s
=> [internal] load metadata for docker.io/library/python:3.10-slim-buster 5.4s
=> [internal] load .dockerignore                                         0.0s
=> => transferring context: 2B                                           0.0s
=> [1/6] FROM docker.io/library/python:3.10-slim-buster@sha256:37aa274c2d00 0.0s
=> [internal] load build context                                         0.0s
=> => transferring context: 801B                                         0.0s
=> CACHED [2/6] WORKDIR /app                                         0.0s
=> CACHED [3/6] COPY requirements.txt requirements.txt                  0.0s
=> [4/6] RUN apt-get update && apt-get install -y && pip3 install -r    482.4s
=> [5/6] RUN rm -rf /var/lib/apt/lists/*                                0.4s
=> [6/6] COPY . .                                                 0.1s
=> exporting to image                                                 1.1s
=> => exporting layers                                              1.1s
=> => writing image sha256:4c9a02d520b7ee3637a9e211e38ece24417a70a7197d14df 0.0s
=> => naming to docker.io/library/exp_pyws_demo                      0.0s
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp4_file/exp_pyws_demo#
```

```
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp4_file/exp_pyws_demo# docker images
REPOSITORY
  TAG      IMAGE ID      CREATED        SIZE
exp_pyws_demo
  latest   4c9a02d520b7  9 minutes ago  179MB
crpi-y10v9g9mjw5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/exp_pyms_api_demo
  1.0      3f3e002f19de  2 days ago   98.5MB
crpi-y10v9g9mjw5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu
  latest   e6ee475e1f11  7 days ago   221MB
hello-world
  latest   d2c94e258dc8  18 months ago  13.3kB
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp4_file/exp_pyws_demo#
```

基于镜像exp_pyws_demo 启动容器:

```
docker run -d --restart=always --name beryl_exp_pyws_demo -p 8000:8000
exp_pyws_demo
```

```
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp4_file/exp_pyws_demo# docker run -d --restart=always --name beryl_exp_pyws_demo -p 8000:8000 exp_pyws_demo
d81bdd9a65c9801ce22160d37e624b671b26d1bf12629da246b5a33469d7a8f1
```

选择一台终端设备，使用浏览器访问访问 ECS 公网IP URL <http://<IP>:8088/> 查看服务页面输出。此外也可在 ECS 中使用curl 命令访问 ECS 本地服务 URL，观察输出以确认服务部署是否生效。

```
curl http://127.0.0.1:8000
```

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~/exp4_file/exp_pyws_demo# curl http://127.0.0.1:80
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome to Devices Page</title>
    <link rel="stylesheet" type="text/css" href="./style.css">
</head>
<body>

    <div id="loading"><p id="loading-text">Loading...</p></div>
    <div class="page">
        <h1>Welcome to Devices Page</h1>
        <table id="devicesInformation">
            <caption><p>Devices Information</p></caption>
            <tr>
                <th id="deviceID">Device ID</th>
                <th id="name">Name</th>
                <th id="type">Type</th>
                <th id="hardwareModel">Hardware Model</th>
                <th id="hardwareSn">Hardware SN</th>
                <th id="softwareVersion">Software Version</th>
                <th id="softwareLastUpdate">Software Last Update</th>
                <th id="networkType">Network Type</th>
                <th id="networkMAC">Network MAC</th>
                <th id="networkIpv4">Network IPv4</th>
                <th id="status">Status</th>
            </tr>
            </table>
    </div>
```

Welcome to Devices Page

Devices Information

Device ID	Name	Type	Hardware Model	Hardware SN	Software Version	Software Last Update	Network Type	Network MAC	Network IPv4	Status
7094306256256503808	device1	camera	Raspberry Pi 4	00001	1.1	2023-07-10 12:05:23	WiFi	00:00:00:00:02	10.0.0.2	alive
7094306256256503809	device2	server	Raspberry Pi 4	00005	5.4	2023-07-13 21:05:23	5G	00:00:00:00:08	101.2.15.8	offline
7094306256256503810	device3	client	PC	acd5d3	11	2023-07-15 21:05:23	eth	00:00:00:00:12	192.168.1.123	alive

测试确认部署无误，关闭容器，删除容器与镜像。

```
docker stop exp_pyws_demo
docker rm exp_pyws_demo
docker rmi exp_pyws_demo
```

基于阿里云云效Flow在虚拟环境部署范例服务

配置Git仓库, 创建分支venv

先创建一个工作文件夹exp_flow,然后进行git拉取自己放在Codeup远程仓库上的项目, 将范例服务包下载到本地环境(继续使用之前已经搭建好的虚拟机)后, 解压进入目录exp_pymys_demo , 将该目录作为Git工作目录。

此时查看目录，内容如下：

```
git clone git@codeup.aaaaaaaaaaaaaaaaaaaaa:f371e/exp_pyms_demo.git
ls # 查看
cd exp_pyms_demo
tar -xzvf exp_pyms_demo.tar.gz
cd exp_pyws_demo
ls -a -p
```

```
beryl@beryl-virtual-machine:~/exp_flow$ git clone git@codeup.aliyun.com:67037224
49e9309ce56b9b0f/exp_pyms_demo.git
正克隆到 'exp_pyms_demo'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
接收对象中: 100% (6/6), 4.15 KiB | 1.38 MiB/s, 完成.
beryl@beryl-virtual-machine:~/exp_flow$ ls
exp_pyms_demo
beryl@beryl-virtual-machine:~/exp_flow$ cd exp_pyms_demo
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo$ tar -xzvf exp_pyms_demo.ta
r.gz
exp_pyws_demo/
exp_pyws_demo/static/
exp_pyws_demo/static/index.js
exp_pyws_demo/static/style.css
exp_pyws_demo/static/index.html
exp_pyws_demo/devices.csv
exp_pyws_demo/server.py
exp_pyws_demo/requirements.txt
[REDACTED]
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo$ cd exp_pyws_demo
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ ls -a -p
./ ... devices.csv requirements.txt server.py static/
```

做到这一步我们已经和远程仓库 `exp_pyms_demo` 建立关联了

查看当前所在分支与工作目录文件状态。每次开工之前，首先从远程仓库 `git pull`

```
git status
git pull origin master
ls -a -F
```

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。
```

未跟踪的文件：

(使用 "git add <文件>..." 以包含要提交的内容)

```
./
```

提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git pull origin master
来自 codeup.aliyun.com:6703722449e9309ce56b9b0f/exp_pyms_demo
* branch            master      -> FETCH_HEAD
已经是最新的。
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ ls -a -F
./  ../  devices.csv  requirements.txt  server.py  static/
```

通过 `git status` 观察工作目录文件状态，随后全部纳入跟踪且暂存，再提交。

```
git status # 检查当前仓库状态

git add . # 将所有文件加入暂存区（追踪文件）
git commit -m"init" # 提交改动，添加提交信息 "init"

git push -u origin master # 推送当前分支到远程仓库的 master 分支，并与远程仓库关联

git checkout -b venv # 创建并切换到一个名为 venv 的新分支
git status
git push --set-upstream origin venv # 将 venv 分支推送到远程仓库，并与远程的 venv 分支建立追踪关系
```

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git status
位于分支 master
您的分支与上游分支 'origin/master' 一致。
```

未跟踪的文件:

(使用 "git add <文件>..." 以包含要提交的内容)

./

提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git add .
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git commit --m "init"
[master 0a52c3f] init
 6 files changed, 224 insertions(+)
 create mode 100644 exp_pyws_demo/devices.csv
 create mode 100644 exp_pyws_demo/requirements.txt
 create mode 100644 exp_pyws_demo/server.py
 create mode 100755 exp_pyws_demo/static/index.html
 create mode 100755 exp_pyws_demo/static/index.js
 create mode 100755 exp_pyws_demo/static/style.css
```

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git push -u
origin master
```

枚举对象中: 11, 完成.

对象计数中: 100% (11/11), 完成.

使用 2 个线程进行压缩

压缩对象中: 100% (9/9), 完成.

写入对象中: 100% (10/10), 3.02 KiB | 618.00 KiB/s, 完成.

总共 10 (差异 0) , 复用 0 (差异 0) , 包复用 0

To codeup.aliyun.com:6703722449e9309ce56b9b0f/exp_pyms_demo.git

794b0ca..0a52c3f master -> master

分支 'master' 设置为跟踪来自 'origin' 的远程分支 'master'。

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git checkout
 -b venv
```

切换到一个新分支 'venv'

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git status
```

位于分支 venv

无文件要提交，干净的工作区

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git push --set-upstream origin venv
```

总共 0 (差异 0) , 复用 0 (差异 0) , 包复用 0

To codeup.aliyun.com:6703722449e9309ce56b9b0f/exp_pyms_demo.git

* [new branch] venv -> venv

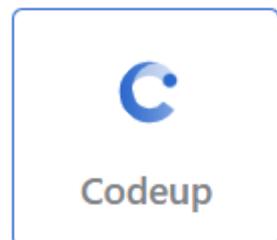
分支 'venv' 设置为跟踪来自 'origin' 的远程分支 'venv'。

配置云效Flow以支持ECS虚拟环境部署

根据模版创建流水线

在云效流水线 Flow 系统新建流水线，选择云效预置模版Python·测试、构建、部署到阿里云ECS/自有主机」。将「测试」、「构建」、「部署」三个阶段命名分别改为「代码测试」、「制品构建」以及「服务部署」。

选择代码源



Codeup

歆毓xinyu的云效Code... + 添加服务连接

+ 新建代码库

代码仓库 exp_pyms_demo

默认分支 ? venv

过滤规则 ?

同时克隆子模块 ? (disabled)

自定义克隆深度 ? (disabled)

开启代码源触发 ? (enabled)

! Flow将在您的代码源中自动添加Webhook

触发事件 代码提交 TAG创建 合并请求完成后 X

□ 合并请求 新建/更新

Webhook <http://flow-openapi.aliyun.com/scm/v>

开启分支或标签过滤 ! Deprecated

开启代码路径过滤 (特定路径下代码发生更新触发)

开启分支模式 ?

工作目录 ?

exp_pyms_demo

配置阶段[代码测试]

本阶段模版设置有「Python 单元测试」与「Python 代码扫描」两个任务。因本实验暂不考虑单元测试，故删除「**Python 单元测试**」任务。而 Python 代码扫描 任务保持模版默认配置即可。

配置阶段[制品构建]

本阶段模版设置有一个任务「Python 构建上传到仓库」，包含「Python 构建」与「构建物上传」两个步骤。本实验无需对 Python 脚本进行特殊构建，故只需使用默认设置将代码制品打包上传至阿里云提供的「云效公有存储空间」即可。

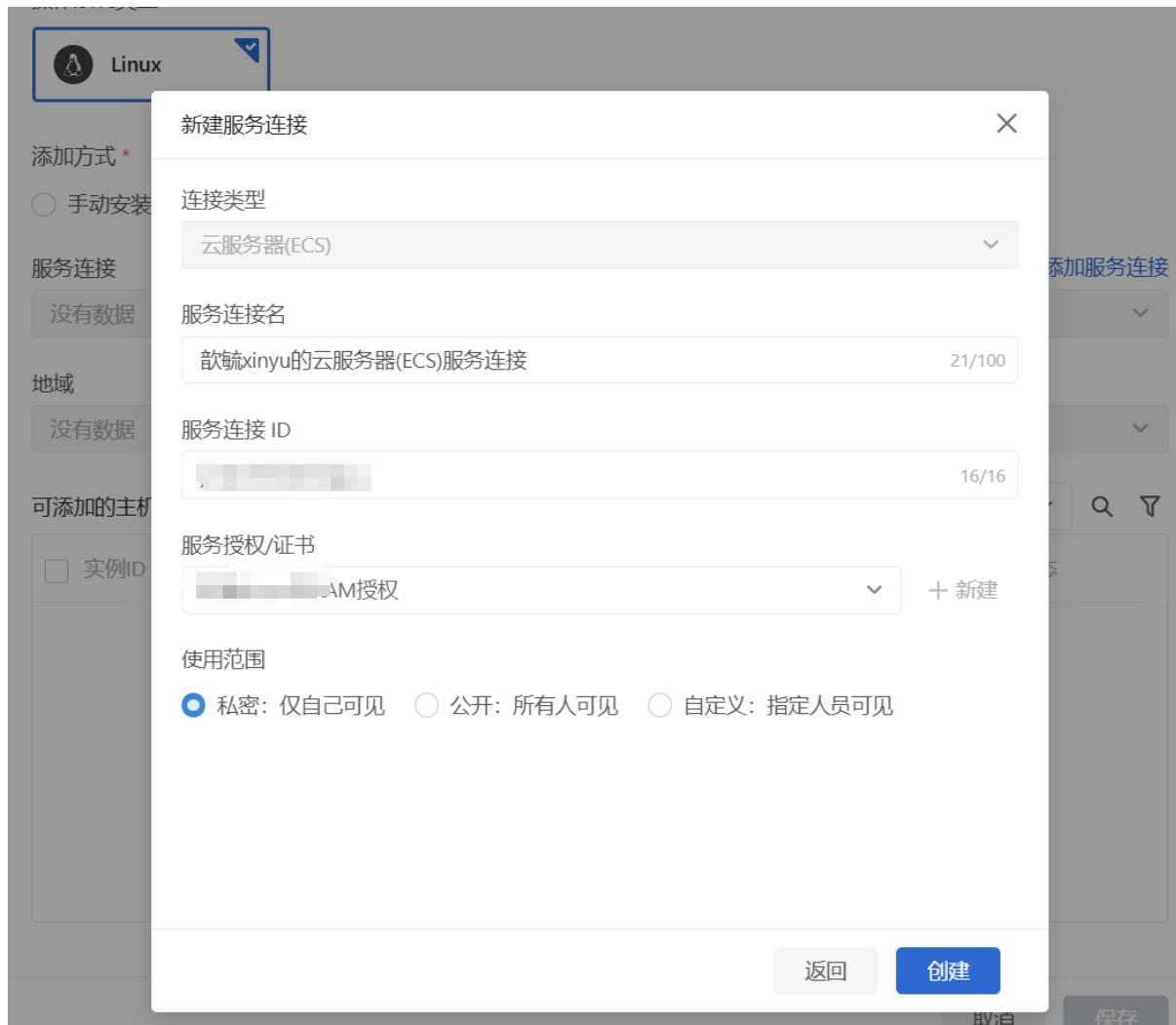
配置阶段[服务部署]

本阶段模版设置有一个任务「主机部署」，其中

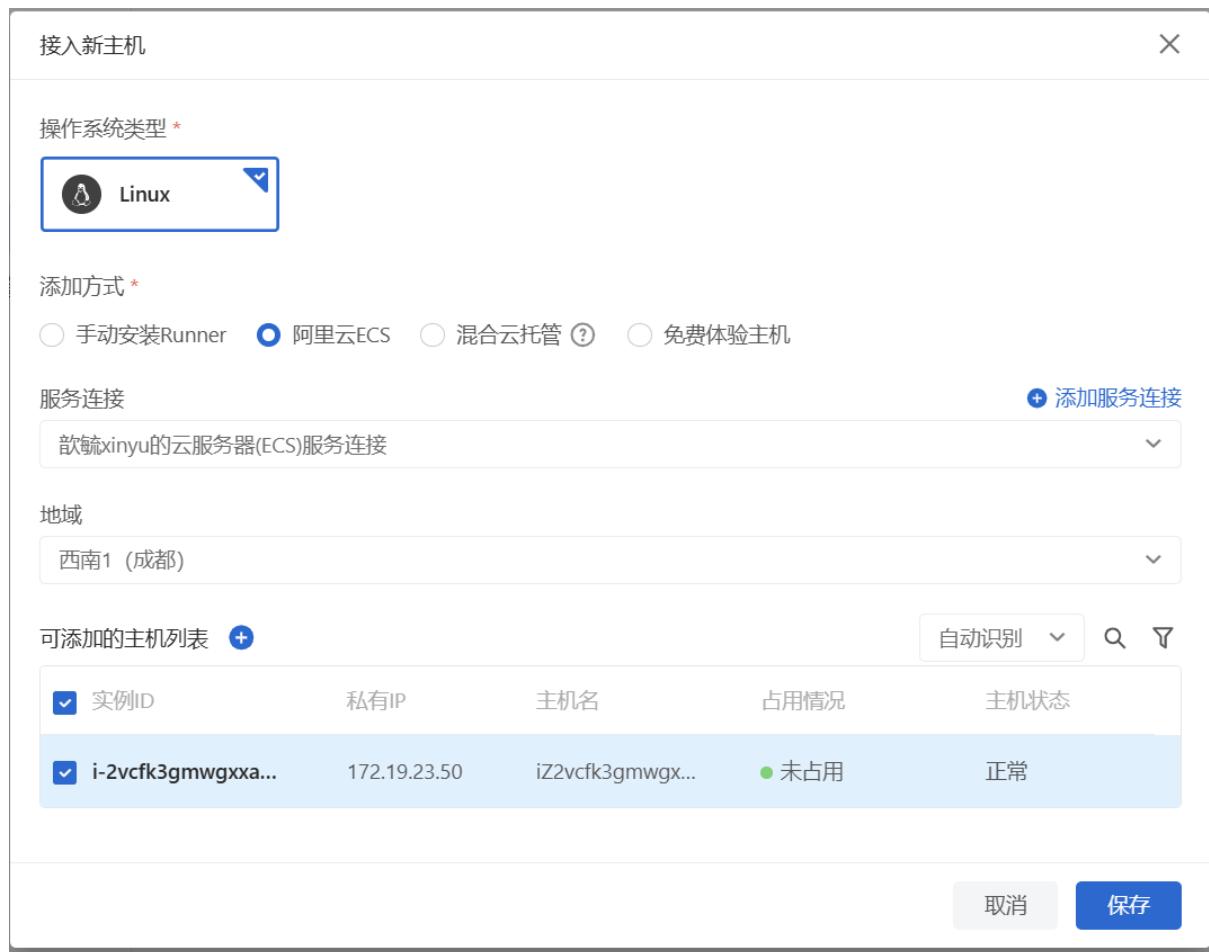
- 「制品」选择上一步「构建物上传」步骤设定的制品名称 `Artifacts_${PIPELINE_ID}`
- 「主机组」
 - 首次使用时，需要点击「新建主机组」，主机名设置 `bery1_exp_flow` 「添加新主机」弹出框中选择「阿里云ECS」。



- 添加服务器连接 添加方式选择「阿里云ECS」，点击「添加服务连接」后如图操作。



- 选择地域，并选择实例进行创建



- 配置部署

- 「下载路径」保持默认即 `/home/admin/app/package.tgz`
- 「执行用户」保持默认即 `root`。
- 「部署脚本」定义如下：

```
# 清理上一次部署时的运行环境
pkill python3
rm -rf ~/exp_pyws_demo

# 解压代码制品到指定运行目录
mkdir ~/exp_pyws_demo
tar -xvzf /home/admin/app/package.tgz -C ~/exp_pyws_demo
# 特别注意，此处路径必须与「部署配置->下载路径」一致
# 特别注意2，-C 后面的路径一定是你在ECS上对应的exp_pyws_demo路径
cd ~/exp_pyws_demo/exp_pyws_demo
# 特别注意3，路径一定是你在ECS上对应的exp_pyws_demo路径

# 建立 Python 虚拟环境并安装依赖
python3 -m venv exp_venv
source exp_venv/bin/activate
pip3 install -r requirements.txt

# 运行 Web 服务
nohup python3 server.py > /dev/null 2>&1 &
```

```
# nohup: 忽略挂起信号, 终端退出不会影响运行。  
# > /dev/null: 将 server.py 的日志输出重定向到 /dev/null, 不显示在终端。  
# 2>&1: 将标准错误输出也重定向到标准输出。  
# &: 让程序在后台运行, 不会因为终端退出而中断。
```

运行流水线

点击「保存并运行」按钮，在弹出的对话框中点击「运行」按钮，流水线即被触发逐一执行阶段任务直至最终完成服务部署。

The screenshot shows the Aliyun DevOps Pipeline interface. At the top, there's a navigation bar with tabs: '最近运行' (Recent Runs) which is selected, '运行历史' (Run History), and '统计报表' (Statistics Report). To the right are buttons for '编辑' (Edit), '运行' (Run), and more options. Below the header, it says '#1 ✓ 运行成功'. It also displays trigger information: 触发信息 - 欲知xinyu · 页面手动触发, 开始时间 2024-11-08 09:48:10, 持续时间 44秒. On the right, there are summary numbers: 3 代码变更, 1 运行产物, 17 环境变量.

The main area shows four stages of the pipeline:

- 流水线源 · 1**: Shows a project named "...xp_pyms_demo" with files: venv, 0a52c3fd, init.
- 代码测试**: Shows a Python code scan task named "Python 代码扫描" completed in 12 seconds. It has 3 errors, 2 warnings, 1 critical error, and 0 general errors.
- 制品构建**: Shows a Python build task named "Python 构建上传到仓库" completed in 12 seconds. It includes an artifact named Artifacts_3690572.
- 服务部署**: Shows a host deployment task named "主机部署" completed in 20 seconds. It includes a deployment detail link.

此时阿里云 ECS 已经基于容器启动了 Python web 服务，通过 curl 命令可测试服务接口是否可访问。

```
curl http://127.0.0.1:8000
```

```
root@iZ2vcfk3gmwgxxap6tntf5Z:~# curl http://127.0.0.1:8000
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome to Devices Page</title>
    <link rel="stylesheet" type="text/css" href="./style.css">
</head>
<body>

    <div id="loading"><p id="loading-text">Loading...</p></div>
    <div class="page">
        <h1>Welcome to Devices Page</h1>
        <table id="devicesInformation">
            <caption><p>Devices Information</p></caption>
            <tr>
                <th id="deviceID">Device ID</th>
                <th id="name">Name</th>
                <th id="type">Type</th>
                <th id="hardwareModel">Hardware Model</th>
                <th id="hardwareSn">Hardware SN</th>
                <th id="softwareVersion">Software Version</th>
                <th id="softwareLastUpdate">Software Last Update</th>
                <th id="networkType">Network Type</th>
                <th id="networkMAC">Network MAC</th>
                <th id="networkIpv4">Network IPv4</th>
                <th id="status">Status</th>
            </tr>
            </table>
    </div>
</body>

<script type="text/javascript" src="index.js"></script>
</html> ?
```

同样也可以选择某台能访问 ECS IP 的终端设备，使用浏览器访问<IP>:8000/，可查看服务页面输出。

修改设别文件, 提交代码并触发流水线

在本地环境(Ubuntu)修改devices.csv, 添加一条设备信息, 保存并退出

```
nano devices.csv
```

```
# 添加以下内容
7097988787878778889,device4,client,PC,00003,10,2023-09-01
00:00:00,WiFi,00:00:00:00:16,156.123.3.14,alive
```

直接提交并推送远程仓库并触发流水线执行。

```
git commit -a -m "docs: Add a device info in devices.csv"
git push origin venv
```

```

beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ nano devices.csv
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ cat devices.csv
device_id, name, type, hardware_model, hardware_sn, software_version, software_last_update, network_type, network_mac, network_ipv4, status
7094306256256503808, device1, camera, Raspberry Pi 4, 00001, 1.1, 2023-07-10 12:05:23, WiFi, 00:00:00:00:02, 10.0.0.2, alive
7094306256256503809, device2, server, Raspberry Pi 4, 00005, 5.4, 2023-07-13 21:05:23, 5G, 00:00:00:00:08, 101.2.15.8, offline
7094306256256503810, device3, client, PC, acd5d3, 11, 2023-07-15 21:05:23, eth, 00:00:00:00:12, 192.168.1.123, alive
7097988787878778889, device4, client, PC, 00003, 10, 2023-09-01 00:00:00, WiFi, 00:00:00:00:16, 156.123.3.14, alive
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git commit -a -m "docs: Add a device info in devices.csv"
[venv e2e3427] docs: Add a device info in devices.csv
 1 file changed, 1 insertion(+)
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo/exp_pyws_demo$ git push origin venv
枚举对象中: 7, 完成.
对象计数中: 100% (7/7), 完成.
使用 2 个线程进行压缩
压缩对象中: 100% (4/4), 完成.
写入对象中: 100% (4/4), 533 字节 | 533.00 KiB/s, 完成.
总共 4 (差异 2) , 复用 0 (差异 0) , 包复用 0
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp_pyws_demo/exp_pyws_demo# cat devices.csv
device_id, name, type, hardware_model, hardware_sn, software_version, software_last_update, network_type, network_mac, network_ipv4, status
7094306256256503808, device1, camera, Raspberry Pi 4, 00001, 1.1, 2023-07-10 12:05:23, WiFi, 00:00:00:00:02, 10.0.0.2, alive
7094306256256503809, device2, server, Raspberry Pi 4, 00005, 5.4, 2023-07-13 21:05:23, 5G, 00:00:00:00:08, 101.2.15.8, offline
7094306256256503810, device3, client, PC, acd5d3, 11, 2023-07-15 21:05:23, eth, 00:00:00:00:12, 192.168.1.123, alive
7097988787878778889, device4, client, PC, 00003, 10, 2023-09-01 00:00:00, WiFi, 00:00:00:00:16, 156.123.3.14, alive
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp_pyws_demo/exp_pyws_demo# 
```

流水线自动执行完毕后，使用浏览器重新访问该服务，可发现设备信息表多出一行，更新成功。

Welcome to Devices Page

Devices Information

Device ID	Name	Type	Hardware Model	Hardware SN	Software Version	Software Last Update	Network Type	Network MAC	Network IPv4	Status
7094306256256503808	device1	camera	Raspberry Pi 4	00001	1.1	2023-07-10 12:05:23	WiFi	00:00:00:00:02	10.0.0.2	alive
7094306256256503809	device2	server	Raspberry Pi 4	00005	5.4	2023-07-13 21:05:23	5G	00:00:00:00:08	101.2.15.8	offline
7094306256256503810	device3	client	PC	acd5d3	11	2023-07-15 21:05:23	eth	00:00:00:00:12	192.168.1.123	alive
7097988787878778889	device4	client	PC	00003	10	2023-09-01 00:00:00	WiFi	00:00:00:00:16	156.123.3.14	alive

完成测试后关闭范例服务

```

ps aux | grep server.py # 查看正在运行的 server.py 进程，找到root目录下进程号 (PID)
sudo kill 进程号 (PID) 终止该进程： 使用 kill 命令终止进程号 (你刚刚找到的)的
server.py 进程
ps aux | grep server.py # 再次检查 server.py 是否仍在运行:

```

```
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp_pyws_demo/exp_pyws_demo# ps aux | grep server.py
root      138814  0.4  2.0  45800 34268 ?        S1   18:55  0:01 python3 server.py
root      140388  0.0  0.1   6616  2320 pts/0    S+   19:02  0:00 grep --color=auto server.py
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp_pyws_demo/exp_pyws_demo# sudo kill 138814
root@iZ2vcfk3gmwgxxap6tnf5Z:~/exp_pyws_demo/exp_pyws_demo# ps aux | grep server.py
root      140649  0.0  0.1   6616  2400 pts/0    S+   19:03  0:00 grep --color=auto server.py
```

基于阿里云云效 Flow 以容器形式部署范例服务

配置Git仓库, 创建分支docker

在本地环境从当前 `venv` 分支中创建检出 `docker` 分支。

```
git checkout -b docker
git branch
```

准备 `Dockerfile` 文件，用于在流水线中制作承载范例服务的 Docker 镜像。

```
nano Dockerfile
```

在 `Dockerfile` 文件中，编写如下内容：

```
# 基于 Python 最新基础镜像，从 https://docker.1panel.live 拉取
FROM docker.1panel.live/library/python:3.10-slim-buster
LABEL maintainer="<ljh2629827042@qq.com>"

# 工作目录
WORKDIR /app

# 安装python库
COPY requirements.txt requirements.txt
RUN apt-get update && apt-get install -y \
    && pip3 install -r requirements.txt -i
https://pypi.tuna.tsinghua.edu.cn/simple

# 清理 apt 软件包缓存
RUN rm -rf /var/lib/apt/lists/*

# 复制当前文件夹中全部文件到镜像中
COPY . .

# 暴露服务端口 8080
EXPOSE 8080

# 容器启动时执行命令
CMD ["python3", "server.py"]
```

注意暴露的服务端口要和 `server.py` 中一致。然后执行下面的指令：

```
git add .
git commit -m "docs: Add Dockerfile"
git push --set-upstream origin docker # 推送 docker 分支到远程仓库并设置追踪关系
```

```
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo$ git add .
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo$ git commit -m "docs: Add Dockerfile"
[docker 6d438b0] docs: Add Dockerfile
 9 files changed, 272 insertions(+), 1 deletion(-)
 create mode 100644 Dockerfile
 create mode 100644 devices.csv
 create mode 100644 exp_pyws_demo/Dockerfile
 create mode 100644 requirements.txt
 create mode 100644 server.py
 create mode 100755 static/index.html
 create mode 100755 static/index.js
 create mode 100755 static/style.css
beryl@beryl-virtual-machine:~/exp_flow/exp_pyms_demo$ git push --set-upstream origin docker
枚举对象中: 8, 完成.
对象计数中: 100% (8/8), 完成.
使用 2 个线程进行压缩
压缩对象中: 100% (5/5), 完成.
写入对象中: 100% (5/5), 1.17 KiB | 399.00 KiB/s, 完成.
总共 5 (差异 1) , 复用 0 (差异 0) , 包复用 0
to codeup.aliyun.com:6703722449e9309ce56b9b0f/exp_pyms_demo.git
 * [new branch]      docker -> docker
分支 'docker' 设置为跟踪来自 'origin' 的远程分支 'docker'。
```

配置云效Flow以支持ECS容器镜像部署

基于前面配置的云效 Flow 流水线进行修改，以支持 ECS 容器镜像模式部署范例服务。即，在云效控制台页面选中之前配置的流水线，点击「编辑」按钮，进入流水线配置页面。其中「代码测试」阶段保持不变，其他三个阶段均需要修改。

配置阶段「流水线源」

将「默认分支」修改为「docker」分支。

The screenshot shows a list of branches for a repository named 'exp_pyms_demo'. The '默认分支' (Default Branch) is set to 'docker', which is highlighted with a red border. There is also a dropdown arrow icon next to the branch names.

配置阶段「镜像构建」

将原「制品构建」阶段更名为「镜像构建」，首先新建「并行任务」用于构建镜像随后再删除原有任务「Python 构建上传到仓库」(否则若该阶段没有任务存在，自身也会被自动删除)

新建「并行任务」时，在弹出页面中选择「镜像构建」，再选择「Python 镜像构建」

随后配置任务「Python 镜像构建」，在步骤「镜像构建并推送至阿里云镜像仓库个人版」中：

- 点击「添加服务链接」，根据页面提示创建新的服务链接，「使用范围」设为私密。
- 选择「地域」，如自己的ACR在哪就在哪，我的在西南1(成都)
- 点击「仓库」旁的 + 按钮，根据页面提示「新建」仓库，命名空间建议设定为「exp_<name>」(因该命名空间为全局命名空间，故而必须唯一)，仓库名称则统一定义为「exp_pyms_demo」。
- 其他配置保持默认即可，其中标签:默认为 \${DATETIME}; Dockerfile路径:默认为 Dockerfile即从仓库目录中直接查找

最后删除「Python 构建上传到仓库」

The screenshot shows the configuration interface for a build task. On the left, there's a flowchart-like diagram with nodes: '镜像构建' (Image Build), 'Python 镜像构建' (Python Image Build) which is highlighted in blue, and '并行任务' (Parallel Task). The main configuration area on the right includes:

- 任务名称:** Python 镜像构建
- 构建集群:** 云效北京构建集群
- 构建环境:** 指定容器环境 (build-steps/alinux3)
- 下载流水线源:** 下载全部流水线源
- 任务步骤:**
 - 安装 Python
 - 执行命令
 - 镜像构建并推送至ACR (个人版)
- 步骤名称:** 镜像构建并推送至ACR (个人版)
- 选择服务连接:** [选择项]
- 地域:** 西南1 (成都)
- 仓库:** 1/exp_pyms_demo
- 使用 VPC 地址推送镜像:** (未选中)
- 标签:** \${DATETIME}

配置阶段「服务部署」

首先新建「并行任务」用于容器部署，随后再删除原有任务「主机部署」(否则若该阶段没有任务存在，自身也会被自动删除)

新建「并行任务」时，在弹出页面中选择「部署」，再选择「Docker部署」。

在具体配置「Docker部署」任务之前，先点击页面顶部的「变量和缓存」，新建字，必须采符变量 **ACR_LOGIN_PWD**，默认值为登陆阿里云容器镜像服务 ACR 时所需密码，用「私密模式」保存。该变量将用于后续任务配置中的「部署脚本」。

变量名称	默认值	描述	私密模式
ACR_LOGIN_PWD	*****	*****	<input checked="" type="checkbox"/>

完成「变量和缓存」的操作后，回到「流程配置」页面，在「服务部署」阶段配置「Docker部署」任务：

- 「主机组」选择刚刚所建主机组
- 「部署配置」
 - 「执行用户」保持默认即 root。
 - 「部署脚本」定义如下，使用时请更换阿里云账号 username 以及地域region。(事实上，建议将 username、region(或者整个 URL)、容器名称都设置为Shell 变量，以增加脚本的通用性，请自行尝试。)

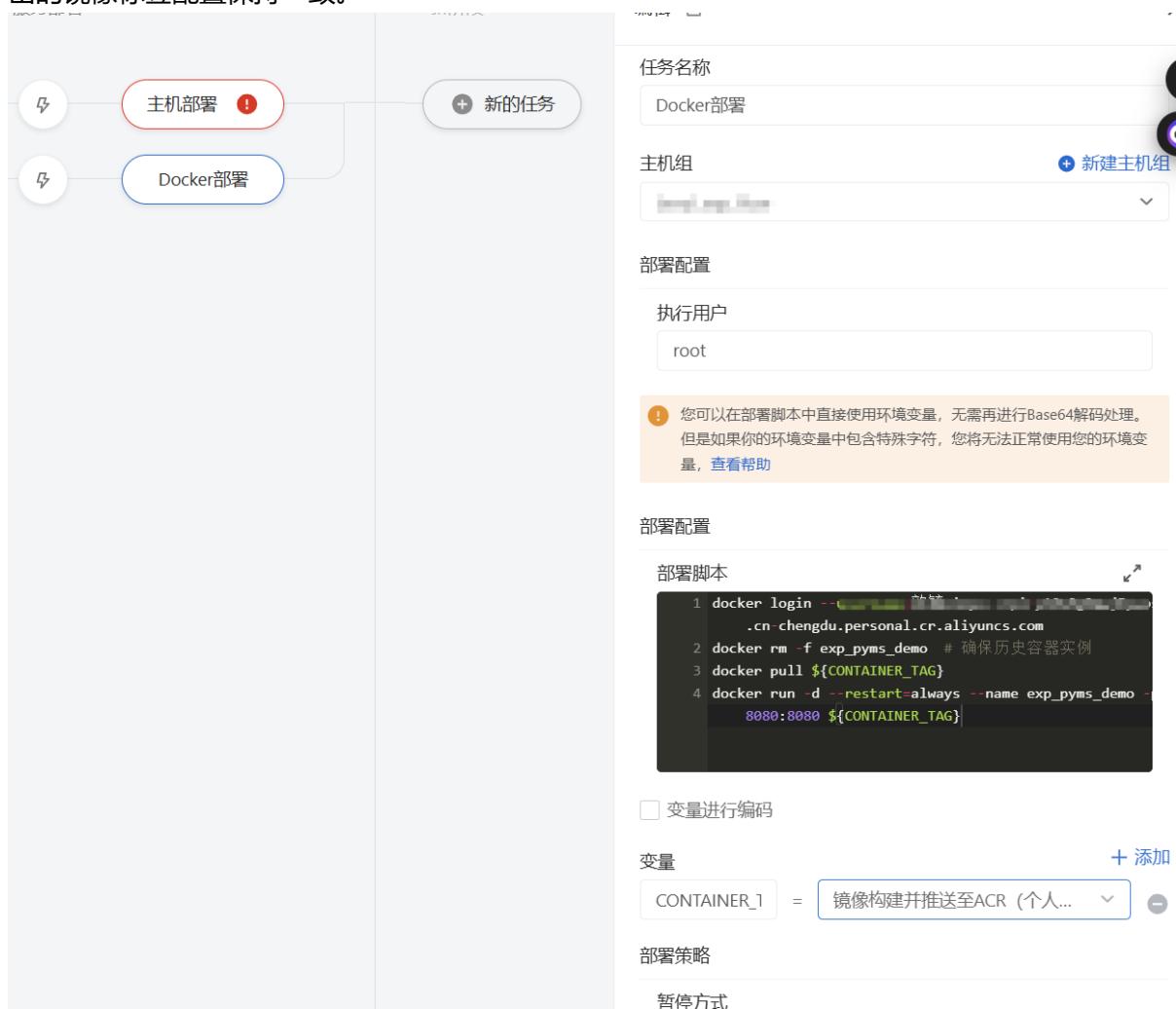
```
docker login --username=<你的名字> --password=${ACR_LOGIN_PWD}
registry.cxxxxxx.aliyuncs.com (可以在ACR中直接复制)

docker rm -f exp_pyms_demo # 确保历史容器实例

docker pull ${CONTAINER_TAG}
docker run -d --restart=always --name exp_pyms_demo -p 8080:8080 ${CONTAINER_TAG}
```

- 「变量」
 - 点击「添加」新增变量，选择「上游任务 制品/镜像 下载地址」，新增变量「CONTAINER_TAG」，变量值选择 标签 \${DATATIME}，与上一阶段任务「Python 镜像构建」产

出的镜像标签配置保持一致。



运行流水线

点击「保存并运行」按钮，在弹出的对话框中点击「运行」按钮，流水线即被触发逐一执行阶段任务直至最终完成服务部署。



运行完成后，在阿里云 ECS 上执行 docker images 与 docker ps 命令查看到生成的镜像和容器。此时使用浏览器重新访问该服务，可见如前面的页面输出。

```
root@iZ2vcfk3gmwgxxap6tnhf5Z:~# docker images
REPOSITORY          IMAGE ID      CREATED        SIZE
crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/exp_pyms_demo   2024-1
1-08-21-28-08      19dd5fc04290   4 minutes ago  395MB
crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/exp_pyms_demo   2024-1
1-08-21-04-42      d6b7fa2296bc   22 minutes ago  395MB
crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/npu           latest
                                         e6ee475e1f11   8 days ago    221MB
hello-world         d2c94e258dcb   18 months ago   13.3kB  🛡

root@iZ2vcfk3gmwgxxap6tnhf5Z:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
NAMES
56427cbef2c4      crpi-y10v9g9mwj5yaosh.cn-chengdu.personal.cr.aliyuncs.com/myexp_1/exp_pym
s_demo:2024-11-08-21-28-08   "python3 server.py"   3 minutes ago   Up 3 minutes   8080/t
cp, 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp   exp_pyms_demo  🛡
root@iZ2vcfk3gmwgxxap6tnhf5Z:~#
```

修改代码文件, 提交代码并触发流水线

观察刚刚运行完流水线之后，在本地环境根据「代码测试」阶段「Python 代码扫描」是存在问题的，根据上次执行所提示的信息修订 server.py 存在的格式问题后提交、推送。接下来修改server.py文件, 提交并推送远程仓库并触发流水线执行。执行完发现结果有变化，实验成功。

触发信息 欢乐tinyu 在 [exp_pyms_demo](#) 提交代码至 docker 触发 | 开始时间 2024-11-08 21:40:23 | 持续时间 2分23秒 ②

代码变更 运行产物 环境

阶段	状态	持续时间	详细信息
代码测试	通过	11秒	4 总数, 2 阻塞, 2 严重, 0 一般
镜像构建	通过	1分38秒	日志
服务部署	通过	31秒	部署详情