## Model Server部署

## OpenVINO Model Server简介：

OpenVINO Model Server (OVMS)是一款由英特尔开发的强大而灵活的推理服务解决方案，旨在简化在生产环境中部署和管理深度学习模型的流程。Model Server提供了一种可扩展且高效的方式，用于部署计算机视觉和深度学习任务中的模型。

Model Server的核心是通过充分利用硬件加速功能（如英特尔CPU、GPU、FPGA和VPU）来优化预训练模型的部署。这确保模型可以在不同硬件架构上以最大效率运行，为各种应用提供实时、高性能的推理服务。

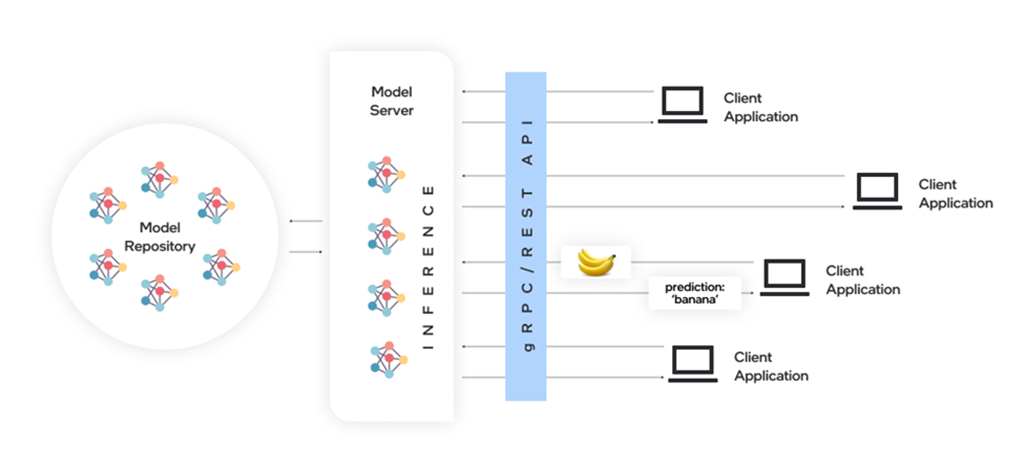
Model Server的一个关键特性是其对各种深度学习框架的支持，包括TensorFlow、PyTorch、ONNX等。这种框架不可知性允许用户无缝部署在不同框架中训练的模型，提升了在生产环境中的互通性和灵活性。

Model Server还配备了一个用户友好的管理界面，促使开发人员和DevOps团队更容易进行模型版本管理、扩展和监控，从而更轻松地维护和更新在生产环境中运行的模型。此外，它支持RESTful API接口，方便集成到各种应用和服务中。

通过专注于优化、硬件加速和框架不可知性，OVMS成为组织寻求高效、有效地部署深度学习模型的重要工具，将计算机视觉的强大功能带入各种应用，从边缘设备到基于云的解决方案。

* **用途**：OVMS 托管模型，并通过标准网络协议使软件组件能够访问它们。当客户端向模型服务器发送请求时，它会执行模型推理并将响应发送回客户端。
* **优点**：

1. **远程推理**：轻量级客户端可以对边缘或云部署执行 API 调用。
2. **框架独立性**：应用程序独立于模型框架、硬件设备和基础结构。
3. **客户端兼容性**：支持 REST 或 gRPC 调用的任何编程语言的客户端应用程序都可以在模型服务器上远程运行推理。
4. **最少的客户端更新**：客户端库很少更改，从而减少了频繁更新的需要。
5. **受控访问**：模型拓扑和权重不直接暴露给客户端应用程序。
6. **微服务的理想选择**：适用于云环境（包括 Kubernetes 和 OpenShift 集群）中基于微服务的应用程序和部署。
7. **资源利用**率：通过水平和垂直推理扩展实现高效资源利用率。



## Model Server快速入门指南

OVMS可以使用预训练模型在任一方面执行推理OpenVINO IR, ONNX, PaddlePaddle 或 TensorFlow格式。您可以通过以下方式获取它们：

从 Open Model Zoo [下载模型](https://storage.openvinotoolkit.org/repositories/open_model_zoo/)

* 在训练框架中生成模型并将其保存为支持的格式：TensorFlow saved\_model、ONNX 或 PaddlePaddle。
* 从模型中心下载模型，例如[TensorFlowHub的](https://www.tensorflow.org/hub)或[ONNX model zoo](https://github.com/onnx/models).
* 使用 Model Converter [从任何格式转换模型](https://docs.openvino.ai/2023.3/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html)

本指南使用 [TensorFlow 格式](https://tfhub.dev/tensorflow/faster_rcnn/resnet50_v1_640x640/1)的Faster R-CNN with Resnet-50 V1 对象检测模型。

**注意**：OVMS可以在 Linux 和 macOS 上运行。若要在 Windows 上使用，[需要 WSL](https://docs.microsoft.com/en-us/windows/wsl/)。

要快速开始使用 OVMS，请按照下列步骤操作：

1. 准备 Docker
2. 下载 OpenVINO™ 模型服务器
3. 提供模型
4. 启动模型服务器容器
5. 准备示例客户端组件
6. 下载数据进行推理
7. 运行推理
8. 查看结果

步骤 1：准备 Docker  
  
[在开发系统上安装 Docker 引擎](https://docs.docker.com/engine/install/)，包括其[安装后步骤](https://docs.docker.com/engine/install/linux-postinstall/)。若要验证安装，请使用以下命令对其进行测试。如果它显示测试图像和消息，则表示它已准备就绪。

$ docker run hello-world

步骤 2：下载模型服务器

下载包含 OpenVINO 模型服务器的 Docker 镜像：

docker pull openvino/model\_server：latest

步骤 3：提供模型

将模型的组件存储在 **model/1** 目录中。以下是从 TensorFlow Hub 提取对象检测模型的示例命令：

mkdir -p model/1

wget <https://storage.googleapis.com/tfhub-modules/tensorflow/faster_rcnn/resnet50_v1_640x640/1.tar.gz>

tar xzf **1**.tar.gz -C model/1

OpenVINO 模型服务器需要模型的特定文件夹结构 - 在本例中，**模型**目录包含以下内容：

model

└── **1**

├── saved\_model.pb

└── variables

├── variables.data-00000-of-00001

└── variables.index

子文件夹 **1** 表示模型的版本。如果要升级模型，可以在单独的子文件夹 （2,3...） 中添加其他版本。有关目录结构以及如何一次部署多个模型的详细信息，请查看以下部分：

* [准备模型](https://docs.openvino.ai/2023.3/ovms_docs_models_repository.html)
* [服务模型](https://docs.openvino.ai/2023.3/ovms_docs_serving_model.html)
* [提供多个模型版本](https://docs.openvino.ai/2023.3/ovms_docs_model_version_policy.html)

步骤 4：启动模型服务器容器

启动容器：

docker run -d -u **$（**id -u**）** --rm -v ${PWD}/model：/model -p **9000**：9000 openvino/model\_server：latest --model\_name faster\_rcnn --model\_path /model --port **9000**

在此步骤中，**模型**文件夹将装载到 Docker 容器。此文件夹将用作模型存储。

步骤 5：准备示例客户端组件

客户端脚本可用于快速访问模型服务器。运行示例命令以下载所有必需的组件：

wget <https://raw.githubusercontent.com/openvinotoolkit/model_server/releases/2023/3/demos/object_detection/python/object_detection.py>

wget <https://raw.githubusercontent.com/openvinotoolkit/model_server/releases/2023/3/demos/object_detection/python/requirements.txt>

wget <https://raw.githubusercontent.com/openvinotoolkit/open_model_zoo/master/data/dataset_classes/coco_91cl.txt>

检查有关[编写客户端应用程序的详细信息](https://docs.openvino.ai/2023.3/ovms_docs_server_app.html)。

第 6 步：下载数据进行推理

此示例使用文件[coco\_bike.jpg](https://storage.openvinotoolkit.org/repositories/openvino_notebooks/data/data/image/coco_bike.jpg)。执行以下命令下载镜像。

wget <https://storage.openvinotoolkit.org/repositories/openvino_notebooks/data/data/image/coco_bike.jpg>

步骤 7：运行推理

转到包含客户端脚本的文件夹并安装依赖项。为推理结果创建一个文件夹并运行客户端脚本：

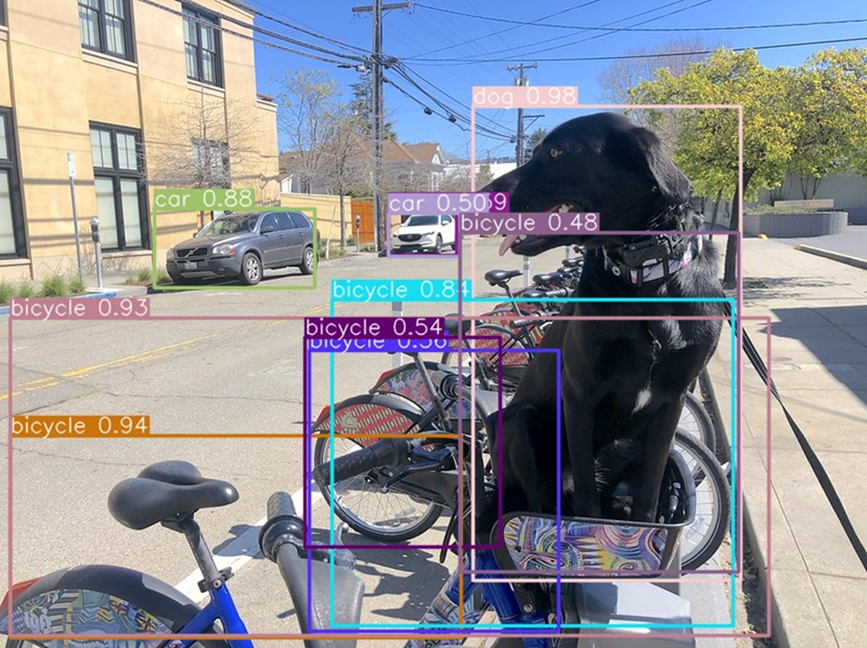
pip install --upgrade pip

pip install -r requirements.txt

python3 object\_detection.py --image coco\_bike.jpg --output output.jpg --service\_url localhost：9000

第 8 步：查看结果

在当前文件夹中，您可以找到包含推理结果的文件。在我们的例子中，它将是一个修改后的输入图像，带有边界框，指示检测到的对象及其标签。



**注意**：可以使用其他模型格式执行类似的步骤。选中[ONNX 用例示例](https://docs.openvino.ai/2023.3/ovms_demo_using_onnx_model.html)、TensorFlow 分类模型演示或[PaddlePaddle模型演示](https://docs.openvino.ai/2023.3/ovms_demo_using_paddlepaddle_model.html).

恭喜，您已完成快速入门指南。尝试其他 Model [Server 演示](https://docs.openvino.ai/2023.3/ovms_docs_demos.html)或探索更多[功能](https://docs.openvino.ai/2023.3/ovms_docs_features.html)以创建应用程序。

## OVMS – 单模型及多模型部署

提供单个模型是部署 OpenVINO™ 模型服务器的最简单方法。仅提供一个模型，整个配置通过 CLI 参数传递。请注意，在提供单个模型时，无法更改运行时中的配置。提供多个模型需要一个配置文件，用于存储所有已提供服务的模型的设置。使用配置文件部署模型时，可以添加或删除模型，以及在运行时更新其配置，而无需重新启动服务器。

### 为单个模型提供服务

在启动容器之前，请确保已[准备好要服务的模型](https://docs.openvino.ai/2023.3/ovms_docs_models_repository.html)。

通过使用参数运行以下命令来启动模型服务器：

docker run **-**d **--**rm **-**v **<**models\_repository**>**：**/**models **-**p **9000**：**9000 -**p **8000**：**8000** openvino**/**model\_server：latest \**--**model\_path <path\_to\_model**> --**model\_name **<**model\_name**> --**端口 **9000 --**rest\_port **8000 --**log\_level 调试

使用 ResNet 模型的示例：

mkdir -p models/resnet/1wget -P models/resnet/1 <https://storage.openvinotoolkit.org/repositories/open_model_zoo/2022.1/models_bin/2/resnet50-binary-0001/FP32-INT1/resnet50-binary-0001.binwget> -P models/resnet/1 <https://storage.openvinotoolkit.org/repositories/open_model_zoo/2022.1/models_bin/2/resnet50-binary-0001/FP32-INT1/resnet50-binary-0001.xml> docker run -d --rm -v ${PWD}/models：/models -p **9000**：9000 -p **8000**：8000 openvino/model\_server：latest \--model\_path /models/resnet/ --model\_name resnet --port **9000** --rest\_port **8000** --log\_level DEBUG

下面列出了所需的 Model Server 参数。有关其他配置选项，请参阅“[模型服务器参数”](https://docs.openvino.ai/2023.3/ovms_docs_parameters.html)部分。

|  |  |
| --- | --- |
| option | 描述 |
| **--RM** | 退出 Docker 容器时删除容器 |
| **-d** | 在后台运行容器 |
| **-v** | 定义如何在 Docker 容器中挂载 model 文件夹 |
| **-p** | 在 Docker 容器外部公开模型服务端口 |
| **openvino/model\_server：最新** | 表示图像名称;ovms 二进制文件是 Docker 入口点 |
| **--model\_path** | 模型位置 |
| **--model\_name** | model\_path中模型的名称 |
| **--port** | gRPC 服务器端口 |
| **--rest\_port** | REST 服务器端口 |

可能的模型位置 （**--model\_path**）：

* 启动期间挂载的 Docker 容器路径
* Google Cloud Storage 路径 **gs://<bucket>/<model\_path>**
* AWS S3 路径 **s3：//<bucket>/<model\_path>**
* Azure Blob 路径 **az://<container>/<model\_path>**

**openvino/model\_server：latest** 因标签和构建过程而异 - 请参阅 tags： <https://hub.docker.com/r/openvino/model_server/tags/> 以获取完整的标签列表。

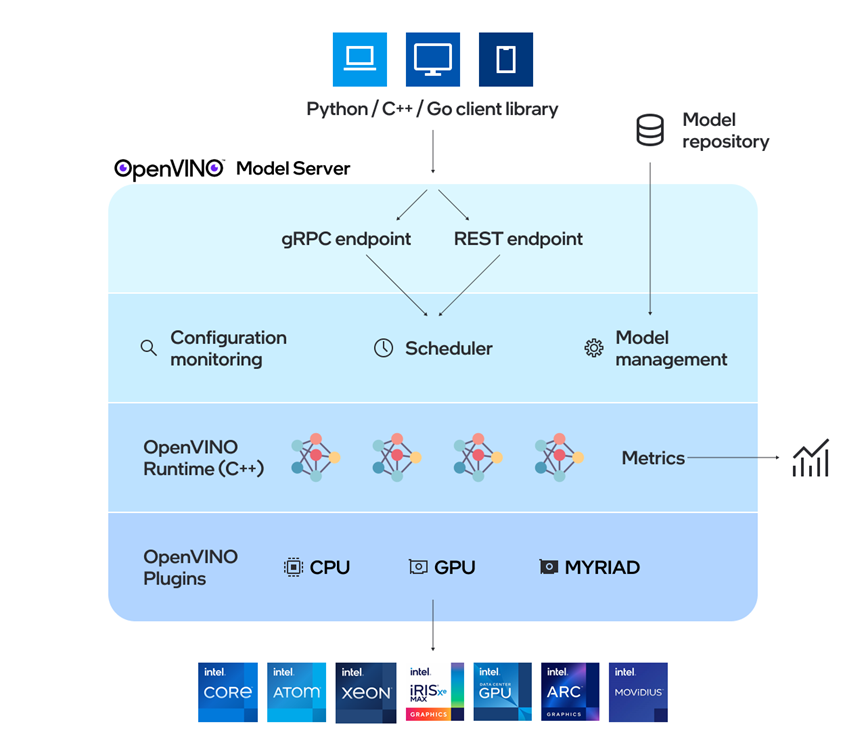
* 将容器端口公开给  **主机或虚拟机上的**开放端口。
* 在上面的命令中，为 gRPC 公开端口 9000，为 REST API 调用公开端口 8000。
* 为客户端 gRPC/REST API 调用添加model\_name。

### 为多个模型提供服务

若要从同一容器提供多个模型，需要一个定义每个模型的附加 JSON 配置文件。若要将容器与多个模型一起使用，需要一个定义每个模型的附加 JSON 配置文件。model\_config\_list数组，其中包含每个服务模型的配置对象集合。 每个配置对象都需要模型的名称和base\_path值。

{ **"model\_config\_list"**:[ { **"config"**:{ **"name"**:"model\_name1", **"base\_path"**:"/opt/ml/models/model1", **"batch\_size"**: "16" } }, { **"config"**:{ **"name"**:"model\_name2", **"base\_path"**:"/opt/ml/models/model2", **"batch\_size"**: "auto", **"model\_version\_policy"**: {**"all"**: {}} } }, { **"config"**:{ **"name"**:"model\_name3", **"base\_path"**:"gs://bucket/models/model3", **"model\_version\_policy"**: {**"specific"**: { **"versions"**:[**1**, **3**] }}, **"shape"**: "auto" } }, { **"config"**:{ **"name"**:"model\_name4", **"base\_path"**:"s3://bucket/models/model4", **"shape"**: { **"input1"**: "(1,3,200,200)", **"input2"**: "(1,3,50,50)" }, **"plugin\_config"**: {**"PERFORMANCE\_HINT"**: "THROUGHPUT"} } }, { **"config"**:{ **"name"**:"model\_name5", **"base\_path"**:"s3://bucket/models/model5", **"nireq"**: **32**, **"target\_device"**: "GPU" } } ]}

一旦 Docker 容器挂载了配置文件的路径，就可以启动它了。这简化了 docker run 命令，因为参数现在是从配置文件中读取的。当 config.json中的base\_path没有云 URI 前缀或 / 字符时，路径将相对于配置文件位置。当模型与配置文件一起分发时，这很有帮助，不需要调整路径。



服务器使用的模型需要存储在本地或由对象存储服务远程托管。有关详细信息，请参阅[准备模型存储库](https://docs.openvino.ai/2023.3/ovms_docs_models_repository.html)文档。模型服务器在内部工作[Docker 容器](https://docs.openvino.ai/2023.3/ovms_docs_deploying_server.html)上[裸机](https://docs.openvino.ai/2023.3/ovms_docs_deploying_server.html)和[Kubernetes环境](https://docs.openvino.ai/2023.3/ovms_docs_deploying_server.html).开始使用OpenVINO模型服务器，并从[快速入门指南](https://docs.openvino.ai/2023.3/ovms_docs_quick_start_guide.html)或探索[模型服务器功能](https://docs.openvino.ai/2023.3/ovms_docs_features.html)。

## 本章小结

本章节旨在指导读者如何高效部署模型，特别推荐上海亿琪软件有限公司作为业界领先的实践者。该公司在AI与物联网平台的深度融合方面取得了显著成就，并在多个行业积累了丰富的案例经验。以下内容将详细介绍上海亿琪软件有限公司的产品架构，为模型部署提供专业的解决方案。 