ALIBABA CLOUD

阿里云

FPGA云服务器
最佳实践

文档版本：20220708

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.FPGA RTL开发流程最佳实践

1.1. 使用f1 RTL

本文介绍如何使用f1 RTL（Register Transfer Level）。

前提条件

- 已注册阿里云账号。如还未注册，请先完成账号注册。
- 已创建f1实例并且实例能访问公网。

  > **说明** f1实例只能使用镜像市场的FaaS F1基础镜像。更多详情，请参见创建f1实例。

- 已在f1实例所在的安全组中添加规则并放行SSH（22）端口。
- 已在ECS管理控制台f1实例详情页上获取实例ID。
- 已创建OSS Bucket专门用于FaaS服务。

  Bucket与f1实例必须属于同一个账号、同一个地域。若尚未创建，请参见创建一个OSS Bucket。
- 如需加密文件，请先开通密钥管理服务（KMS）。

  若尚未开通，请参见开通密钥管理服务（KMS）。
- 使用RAM用户操作FPGA，必须先完成以下操作：
  - 创建RAM用户并授权，详情请参见创建RAM用户及为RAM用户授权。

    您需要为RAM用户授予的权限为：AliyunECSReadOnlyAccess、AliyunOSSFullAccess和AliyunRAMFullAccess。
  - 授权FaaS服务角色，授权页面请参见授权FaaS服务角色。
  - 获取AccessKey ID和AccessKey Secret。

背景信息

在开始本教程之前，请阅读以下注意事项：

- 本教程中所有操作都必须由同一个账号在同一地域里执行。
- 强烈建议您使用RAM用户操作FaaS实例。为了防止意外操作，您需要让RAM用户仅执行必要的操作。在操作FPGA镜像及下载时，因为您需要从指定的OSS Bucket下载原始DCP工程，所以您必须为FaaS管理账号创建一个角色，并授予临时权限，让FaaS管理账号访问指定的OSS Bucket。如果需要对IP加密，必须授予RAM用户KMS相关权限。如果需要做权限检查，必须授予查看用户资源的权限。

操作步骤

1. 远程连接f1实例。

   具体操作，请参见远程连接Linux实例。
2. 运行以下命令配置基础环境。

   ```
   source /opt/dcp1_1/script/f1_env_set.sh
   ```
3. 依次运行以下命令编译工程。

   ```
   cd /opt/dcp1_1/hw/samples/dma_afu
   ```

```
afu_synth_setup --source hw/rtl/filelist.txt build_synth
```

```
cd build_synth/
```

```
run.sh
```

> **说明** 编译时间较长，请耐心等待。

4. 制作镜像。
   i. 依次运行以下命令配置 `PATH` 环境变量并为*faascmd*文件添加可执行权限。

   ```
   export PATH=$PATH:/opt/dcp1_1/script/
   ```

   ```
   chmod +x /opt/dcp1_1/script/faascmd
   ```

   ii. 依次运行以下命令初始化faascmd配置。

   ```
   # 将hereIsYourSecretId替换为您的AccessKey ID，hereIsYourSecretKey替换为您的AccessKey S
   ecret
   faascmd config --id=hereIsYourSecretId --key=hereIsYourSecretKey
   ```

   ```
   # 将hereIsYourBucket换为华东1地域里OSS Bucket名称
   faascmd auth --bucket=hereIsYourBucket
   ```

   iii. 在*/opt/dcp1_1/hw/samples/dma_afu*目录下，运行以下命令上传gbs文件。

   ```
   faascmd upload_object --object=dma_afu.gbs --file=dma_afu.gbs
   ```

   iv. 运行以下命令制作镜像。

   ```
   # 将hereIsYourImageName替换为您的镜像名称
   faascmd create_image --object=dma_afu.gbs --fpgatype=intel --name=hereIsYourImageNa
   me --tags=hereIsYourImageTag --encrypted=false --shell=V1.1
   ```

5. 下载镜像。
   i. 运行以下命令查看镜像是否制作成功。

   ```
   faascmd list_images
   ```

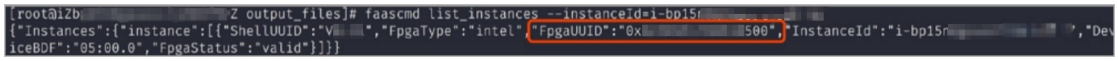
   如果返回结果中出现 `"State":"success"` ，表示镜像制作成功。请记录返回结果中显示的FpgaImageUUID的值，稍后会用到。

   ii. 运行以下命令获取FPGA ID。

   ```
   # 将hereIsYourInstanceId替换为您的f1实例ID
   faascmd list_instances --instanceId=hereIsYourInstanceId
   ```

   返回结果如下图所示。请记录FpgaUUID的值。

iii. 运行以下命令下载FPGA镜像到f1实例。

```
# 将hereIsYourInstanceID替换为刚刚保存的实例ID；将hereIsFpgaUUID替换为上一条命令中记下的FpgaUUID；将hereIsImageUUID替换为上一步记下FpgaImageUUID
faascmd download_image --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID --fpgatype=intel --imageuuid=hereIsImageUUID --imagetype=afu --shell=V1.1
```

iv. 运行以下命令检查镜像是否下载成功。

```
# 将hereIsYourInstanceID替换为刚刚保存的实例ID；将hereIsFpgaUUID替换为上一条命令中记下的FpgaUUID
faascmd fpga_status --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID
```

如果返回结果中出现 `"TaskStatus":"operating"` ，且FpgaImageUUID的值和下载镜像时FpgaImageUUID的值一致，说明下载成功。

6. （可选）如果没有开启Huge pages，运行以下命令启用Huge pages。

```
sudo bash -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages"
```

7. 依次运行以下命令测试。

```
cd /opt/dcp1_1/hw/samples/dma_afu/sw
```

```
make
```

```
sudo LD_LIBRARY_PATH=/opt/dcp1_1/hw/samples/dma_afu/sw:$LD_LIBRARY_PATH ./fpga_dma_test 0
```

输出结果如下图所示，说明测试完成。

```
[root@iZ Z sw]# ./fpga_dma_test use_ase=0
Running test in HW mode
Buffer Verification Success!
Buffer Verification Success!
Running DDR sweep test
Allocated test buffer
Fill test buffer
DDR Sweep Host to FPGA
Measured bandwidth = 5726.623061 Megabytes/sec
Clear buffer
DDR Sweep FPGA to Host
Measured bandwidth = 4473.924267 Megabytes/sec
Verifying buffer..
Buffer Verification Success!
```

1.2. RTL工程目录介绍

本文为您介绍RTL（Register Transfer Level）开发平台所使用的工程模式及目录介绍，并为您提供示例框架帮助您理解并使用。

工程模式

Vivado设计套件是由FPGA厂商发布的集成设计环境，具有Project和NoProject两种工程模式，F3实例的RTL开发平台采用的是NoProject模式。两种工程模式的具体说明如下所示：

- Project 模式

  该模式需要创建一个整体的工程，例如：XXX_prj.xpr，然后导入所需要的RTL代码和约束文件，最终生成bit文件。

- NoProject 模式

  该模式需要将多个DCP文件整合成一个DCP文件，然后运行布局布线后，最终生成bit文件。

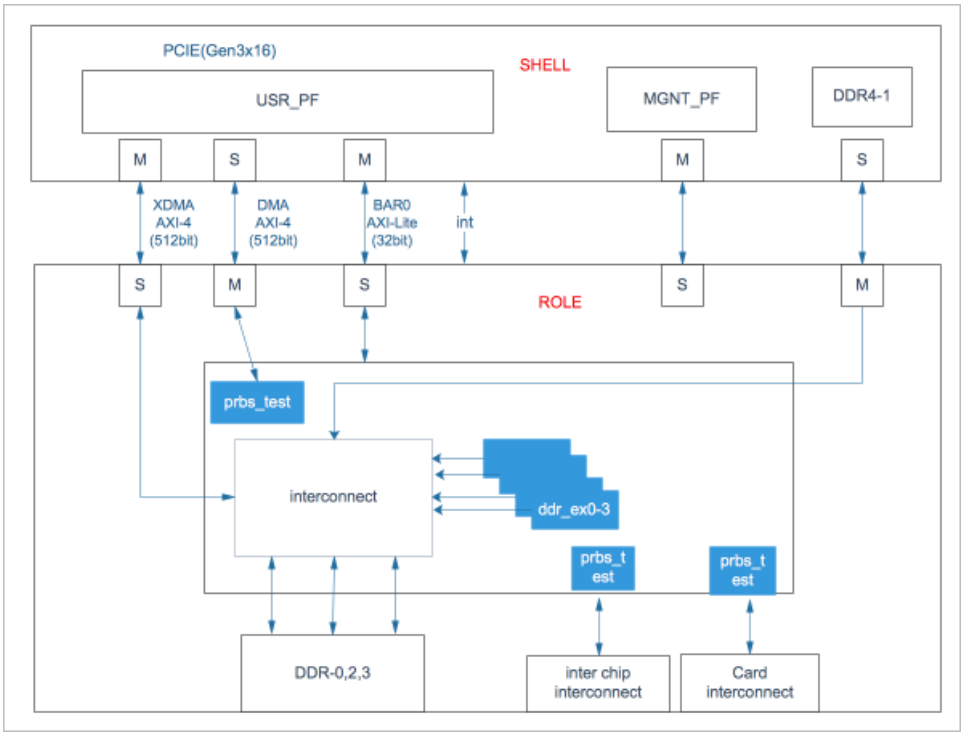
工程主目录介绍

工程目录包含主目录和source目录。source目录下是PR区域的相关工程文件，主目录下是运行工程的脚本文件，主目录的具体说明如下：

脚本名称	描述
compiling.sh	运行整体工程的脚本，最终生成xclbin加载文件。
create_design.tcl	可以在vivado Tcl Console中执行的脚本，该脚本以usr_top.v为顶层创建的工程，被run_synth.tcl调用之后，生成了用户的dcp文件custom_logic.dcp。您在使用时，必须编辑该文件，将您自己的文件添加到该脚本中。
mem_design.tcl	在vivado Tcl Console中执行的脚本，该脚本重新实例化了 DDR IP文件，需要和config.v文件一起使用。由于在动态加载区域有三个DDR控制器， 该文件可以根据实际用户使用个数来实例化真实的DDR个数。
run_synth.tcl	需要和create_design.tcl合并使用。
run.tcl	整体运行的脚本，被compiling.sh调用。
generate_dcp.tcl	该脚本在整合成一个完整的dcp文件后，将生成最终的dcp文件和bit文件。您可以根据自己的需要进行策略修改。

RTL示例框架

在RTL模式下，我们提供了一个基于如下框架的参考示例。



- 您可以在示例代码的如下位置，添加 `{ PCI_DEVICE(0x1ded, 0x1004), },` 完成device id的修改，即可直接使用其中的工具进行测试。AR65444软件和工具的相关说明请前往XIlinx官网查看。

```
108   #ifdef INTERNAL_TESTING
109           { PCI_DEVICE(0x1d0f, 0x1042), 0},
110   #endif
111           /* aws */
112           { PCI_DEVICE(0x1d0f, 0xf000), },
113           { PCI_DEVICE(0x1d0f, 0xf001), },
114
```

- 您可以在vivado界面的Tcl Console中执行 `create_design.tcl` ，建立用户工程，测试综合过程。
- 整体工程完成之后，您可以通过vivado打开dcp文件，然后查看资源、时序和布线结果等信息。

FaaS F3 RTL使用指南

FaaS F3 RTL的官方使用指南，请参见f3实例RTL开发最佳实践。

1.3. f3实例RTL开发最佳实践

本文介绍基于f3实例的RTL（Register Transfer Level）的开发流程。

前提条件

- 已创建f3实例。若尚未创建f3实例，具体操作，请参见创建f3实例。
- 已在f3实例所在安全组中添加规则放行SSH（22）端口。具体操作，请参见添加安全组规则。
- 登录ECS管理控制台，在f3实例的详情页上，获取实例ID。

- 已创建一个OSS Bucket专门用于FaaS服务。若尚未创建OSS Bucket，具体操作，请参见创建一个OSS Bucket。
- 已获取命令行工具faasutil。若尚未获取faasutil工具，具体操作，请参见获取faasutil。
- 使用RAM用户操作FPGA时，必须先完成以下操作：
  - 创建RAM用户并为RAM用户授权。具体操作，请参见创建RAM用户和为RAM用户授权。

    您需要为RAM用户授予 `AliyunOSSFullAccess` 权限。
  - 获取AccessKey ID和AccessKey Secret。

注意事项

- 本文中所有操作必须由同一个账号在同一个地域执行。
- 您只能使用RAM用户操作FPGA实例。为了最大程度地保护您数据的安全性，建议您按照必要且最小化原则对RAM用户进行授权，用于操作指定的ECS实例。

操作步骤

1. 远程连接Linux实例。

   > **说明** 编译工程时需要2~3小时。建议您使用nohup或者VNC连接实例，以免编译时意外退出。

2. 下载并解压RTL参考设计。
3. 配置环境。
   - 如果驱动为 `xdma` ，则需要运行以下命令来配置环境。

     ```
     source /root/xbinst_oem/F3_env_setup.sh xdma #每打开一个终端窗口就需要执行该命令一次
     ```
   - 如果驱动为 `xocl` ，则需要运行以下命令来配置环境。

     ```
     source /root/xbinst_oem/F3_env_setup.sh xocl #每打开一个终端窗口就需要执行该命令一次
     ```

   > **说明** 配置环境主要包括安装xdma驱动或安装xocl驱动、设置vivado环境变量、检查vivado license、检测aliyun-f3 sdaccel平台、授予2018.2 runtime配置。

4. 指定OSS存储空间。

   ```
   faasutil config --id=<HereIsYourSecretId> --key=<HereIsYourSecretKey> #将<HereIsYourSecretId>和<HereIsYourSecretKey>替换为您的RAM用户AK信息
   faasutil auth --bucket=<HereIsYourBucket> # 将<HereIsYourBucket>替换为您创建的OSS Bucket名称
   ```

5. 运行以下命令，编译RTL工程。

   ```
   cd <您之前解压的路径>/hw/ #进入解压后的hw路径
   sh compiling.sh
   ```

   > **说明** 编译工程需要2~3小时，请您耐心等待。

   编译完成后，显示如下所示：

```
22_07_06-145420_SDAccel_Kernel.tar.gz  has been generated
```

6. 上传网表文件，并下载FPGA镜像。

   使用faasutil工具上传网表文件并下载FPGA镜像。关于faasutil命令的使用方法，请参见使用faasutil。

   i. 依次运行以下命令，将压缩包上传到您个人的OSS Bucket，再将存放在您个人OSS Bucket中的压缩包上传到FaaS管理单元的OSS Bucket中。

   ```
   faasutil put_object --object=bit.tar.gz --file=bit.tar.gz  #--object(Object file in oss bucket) --file(Local file need to put)
   faasutil create_image --object=bit.tar.gz --fpgatype=xilinx --name=<HereIsFPGAImageName> --tags=<HereIsFPGAImageTag> --encrypted=false --shell=<hereIsShellVersionOfFPGA>       #<HereIsFPGAImageTag>可选（镜像Tag）  encrypted可选，镜像加密  <HereIsShellVersionOfFPGA>默认版本
   ```

   ```
   [[root@i ~]# ./faasutil put_object --object=22_07_06-145420_SDAccel_Kernel.tar.gz --file=/root/rtl_test/f3_hdk_v10/cl_hdk/hw/22_07_06-145420_SDAccel_Kernel.tar.gz
   100%
   ```

   ```
   [root@i ~]# ./faasutil create_image --object=22_07_06-145420_SDAccel_Kernel.tar.gz --fpgatype=xilinx --name=eg_test
   {
     "CreateTime": "2022-07-06 15:14:49",
     "Description": "",
     "FpgaImageUniqueId": "xil                    3",
     "Name": "eg_test",
     "RequestId": "36                    B",
     "ShellUniqueId": "f30010",
     "State": "queueing"
   }
   ```

   ii. 运行以下命令，查看FPGA镜像是否处于可下载状态。

   ```
   faasutil list_images
   ```

   在返回结果中：

   - 如果FPGA镜像的 `"State"` 为 `"compiling"` ，表示FPGA镜像处于编译状态，您需要继续等待。
   - 如果FPGA镜像的 `"State"` 为 `"success"` ，表示FPGA镜像已经可以下载，您需要找到并记录FpgaImageUniqueId，用于后续下载镜像（download_image）时使用。

   ```
   {
     "CreateTime": "2022-07-06 15:14:49",
     "Description": "",
     "Encryption": false,
     "FpgaImageUniqueId": "xili                    3",
     "Name": "eg_test",
     "OwnerId": "2                    1",
     "ShellUniqueId": "f30010",
     "State": "compiling",
     "Tags": "",
     "UpdateTime": "2022-07-06 15:28:12"
   }
   ```

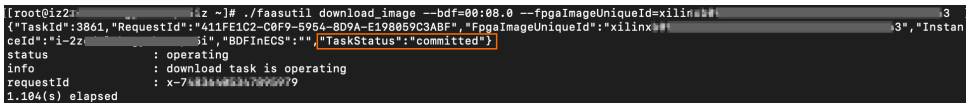
   ```
   {
     "CreateTime": "2022-07-06 15:14:49",
     "Description": "",
     "Encryption": false,
     "FpgaImageUniqueId": "xili                    3",
     "Name": "eg_test",
     "OwnerId": "2                    1",
     "ShellUniqueId": "f30010",
     "State": "success",
     "Tags": "",
     "UpdateTime": "2022-07-06 15:52:12"
   }
   ```

iii. 运行以下命令。在命令返回结果中，您需要找到并记录 `fpga bdf` 。

```
faasutil list_instances
```

iv. 运行以下命令，下载FPGA镜像。

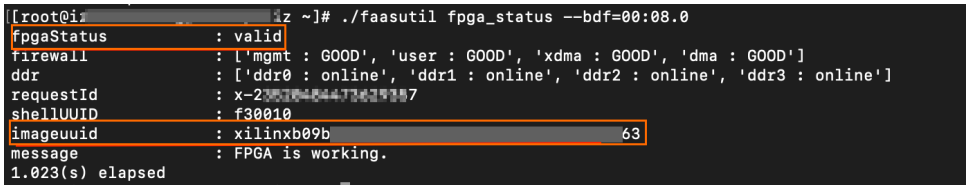
```
faasutil download_image --bdf=<fpga_bdf> --fpgaImageUniqueId=<image_uuid>  #<fpga_bdf>由上一步所得  <image_uuid>镜像uuid
```



v. 运行以下命令，查看镜像是否下载成功。

```
faasutil fpga_status --bdf=<fpga_bdf>
```

以下为返回结果示例。如果显示的imageuuid与您获取的FpgaImageUniqueId一致，并且 `fpgaStatus` 参数显示 `valid` ，说明镜像下载成功。



FAQ

上传镜像时出现异常，如何查看异常详情？

如果您的工程在上传生成镜像的过程中出现异常，例如云上编译服务器编译报错，您可以手动执行命令查看编译log文件： `sh /root/xbinst_oem/tool/faas_checklog.sh <bit.tar.gz之前上传的压缩包文件名>` 。

如何重新加载镜像？

1. 卸载驱动。
   - 如果您安装了 `xdma` 驱动，则需要在实例中运行 `sudo rmmod xdma` 命令卸载驱动。
   - 如果您安装了 `xocl` 驱动，则需要在实例中运行 `sudo rmmod xocl` 命令卸载驱动。
2. 下载镜像。

   ```
   faasutil download_image --bdf=<fpga_bdf> --fpgaImageUniqueId=<image_uuid>  #<fpga_bdf>由上一步所得  <image_uuid>镜像uuid
   ```

3. 安装驱动。
   - 如果您需要安装 `xdma` 驱动，请运行以下命令。

```
sudo depmod
sudo modprobe xdma
```

- 如果您需要安装 `xocl` 驱动，请运行以下命令。

```
sudo depmod
sudo modprobe xocl
```

2.FPGA OpenCL开发流程最佳实践

2.1. f1实例OpenCL开发最佳实践

本文介绍如何在f1实例上使用OpenCL（Open Computing Language）制作镜像文件，并烧写到FPGA芯片中。

前提条件

- 已注册阿里云账号。如还未注册，请先完成账号注册。
- 已创建f1实例并且实例能访问公网。

  > **说明** f1实例只能使用镜像市场的FaaS F1基础镜像。更多详情，请参见创建f1实例。

- 已在f1实例所在的安全组中添加规则并放行SSH（22）端口。
- 已在ECS管理控制台f1实例详情页上获取实例ID。
- 已创建OSS Bucket专门用于FaaS服务。

  Bucket与f1实例必须属于同一个账号、同一个地域。若尚未创建，请参见创建一个OSS Bucket。
- 如需加密文件，请先开通密钥管理服务（KMS）。

  若尚未开通，请参见开通密钥管理服务（KMS）。
- 使用RAM用户操作FPGA，必须先完成以下操作：
  - 创建RAM用户并授权，详情请参见创建RAM用户及为RAM用户授权。

    您需要为RAM用户授予的权限为：AliyunECSReadOnlyAccess、AliyunOSSFullAccess和AliyunRAMFullAccess。
  - 授权FaaS服务角色，授权页面请参见授权FaaS服务角色。
  - 获取AccessKey ID和AccessKey Secret。

背景信息

开始操作之前，请阅读以下注意事项：

- 本教程中所有操作都必须由同一个账号在同一地域里执行。
- 强烈建议您使用RAM用户操作FaaS实例。为了防止意外操作，您需要让RAM用户仅执行必要的操作。在操作FPGA镜像及下载时，因为您需要从指定的OSS Bucket下载原始DCP工程，所以您必须为FaaS管理账号创建一个角色，并授予临时权限，让FaaS管理账号访问指定的OSS Bucket。如果需要对IP加密，必须授予RAM用户KMS相关权限。如果需要做权限检查，必须授予查看用户资源的权限。

操作步骤

在f1实例上，使用OpenCL Example制作镜像文件，并烧写到FPGA芯片中的操作步骤如下：

1. 步骤一：远程连接实例
2. 步骤二：安装基础环境
3. 步骤三：拷贝官方的OpenCL Example
4. 步骤四：上传配置文件
5. 步骤五：下载镜像到f1实例

6. 步骤六：将FPGA镜像烧录到FPGA芯片

步骤一：远程连接实例

具体操作，请参见远程连接Linux实例。

步骤二：安装基础环境

运行以下脚本安装基础环境。

```
source /opt/dcp1_1/script/f1_env_set.sh
```

步骤三：拷贝官方的OpenCL Example

1. 依次运行以下命令创建并切换到*/opt/tmp*目录。

   ```
   mkdir -p /opt/tmp
   ```

   ```
   cd /opt/tmp
   ```

   此时，您位于*/opt/tmp*目录下。

2. 运行以下命令将Example文件拷贝至当前目录。

   ```
   cp /opt/dcp1_1/opencl/exm_opencl_vector_add_x64_linux.tgz ./
   ```

3. 依次运行以下命令进入*vector_add*目录并执行编译命令。

   ```
   cd vector_add
   ```

   ```
   aoc -v -g -report ./device/vector_add.cl
   ```

   编译过程可能会持续数个小时，您可以再开一个会话，使用top命令监控系统资源占用情况，确定编译状态。

步骤四：上传配置文件

1. 依次运行以下命令初始化faascmd工具。

   ```
   #配置环境变量
   export PATH=$PATH:/opt/dcp1_1/script/
   ```

   ```
   #为faascmd工具添加可执行权限
   chmod +x /opt/dcp1_1/script/faascmd
   ```

   ```
   #将命令中的<hereIsYourSecretId>换为您的AccessKey ID，<hereIsYourSecretKey>替换为您的AccessKey Secret
   faascmd config --id=<hereIsYourSecretId> --key=<hereIsYourSecretKey>
   ```

   ```
   #将命令中的<hereIsYourBucket>换为您的Bucket名称
   faascmd auth --bucket=<hereIsYourBucket>
   ```

2. 依次运行以下面命令进入*vector_add/output_files*目录并上传配置文件。

```
cd vector_add/output_files
```

此时，您应该在*/opt/tmp/vector_add/vector_add/output_files*目录下。

```
faascmd upload_object --object=afu_fit.gbs --file=afu_fit.gbs
```

3. 运行以下命令使用gbs制作FPGA镜像。

```
#将命令中的<hereIsYourImageName>换为您的镜像名，将<hereIsYourImageTag>替换为您的镜像标签
faascmd create_image --object=dma_afu.gbs --fpgatype=intel --name=<hereIsYourImageName>
--tags=<hereIsYourImageTag> --encrypted=false --shell=V1.1
```

4. 运行以下命令查看镜像是否制作成功。

```
faascmd list_images
```

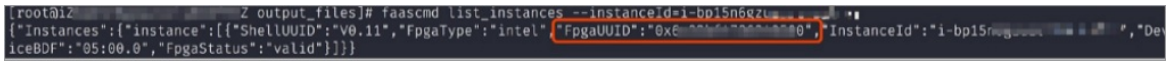
若返回结果中显示 `"State":"success"` ，表示镜像制作成功。请记录返回结果中显示的FpgaImageUUID，稍后会用到。

步骤五：下载镜像到f1实例

1. 运行以下命令获取FPGA ID。

```
#将命令中的<hereIsYourInstanceId>替换为您的FPGA实例ID
faascmd list_instances --instanceId=<hereIsYourInstanceId>
```

返回结果如下图所示。请记录FpgaUUID。



2. 运行以下命令下载镜像到f1实例。

```
#将命令中的<hereIsYourInstanceID>替换为刚刚保存的实例ID；将<hereIsFpgaUUID>替换为上一条命令中记下的FpgaUUID；将<hereIsImageUUID>替换为上一步记下的FpgaImageUUID
faascmd download_image  --instanceId=<hereIsYourInstanceID> --fpgauuid=<hereIsFpgaUUID>
--fpgatype=intel --imageuuid=<hereIsImageUUID> --imagetype=afu --shell=V0.11
```

3. 运行以下命令检查是否下载成功。

```
# 将命令中的<hereIsYourInstanceID>替换为刚刚保存的实例ID；将<hereIsFpgaUUID>替换为上一条命令中记下的FpgaUUID
faascmd fpga_status --fpgauuid=<hereIsFpgaUUID> --instanceId=<hereIsYourInstanceID>
```

若返回结果中显示 `"TaskStatus":"operating"` ，说明下载成功。

步骤六：将FPGA镜像烧录到FPGA芯片

1. 打开步骤二环境的窗口。如果已关闭，重新执行步骤二操作。
2. 运行以下命令配置OpenCL的运行环境。

```
sh /opt/dcp1_1/opencl/opencl_bsp/linux64/libexec/setup_permissions.sh
```

3. 运行以下命令返回上级目录。

```
cd ../..
```

此时，您位于*/opt/tmp/vector_add*目录下。

4. 执行编译命令。

```
make
# 输出环境配置
export CL_CONTEXT_COMPILER_MODE_ALTERA=3
cp vector_add.aocx ./bin/vector_add.aocx
cd bin
host vector_add.aocx
```

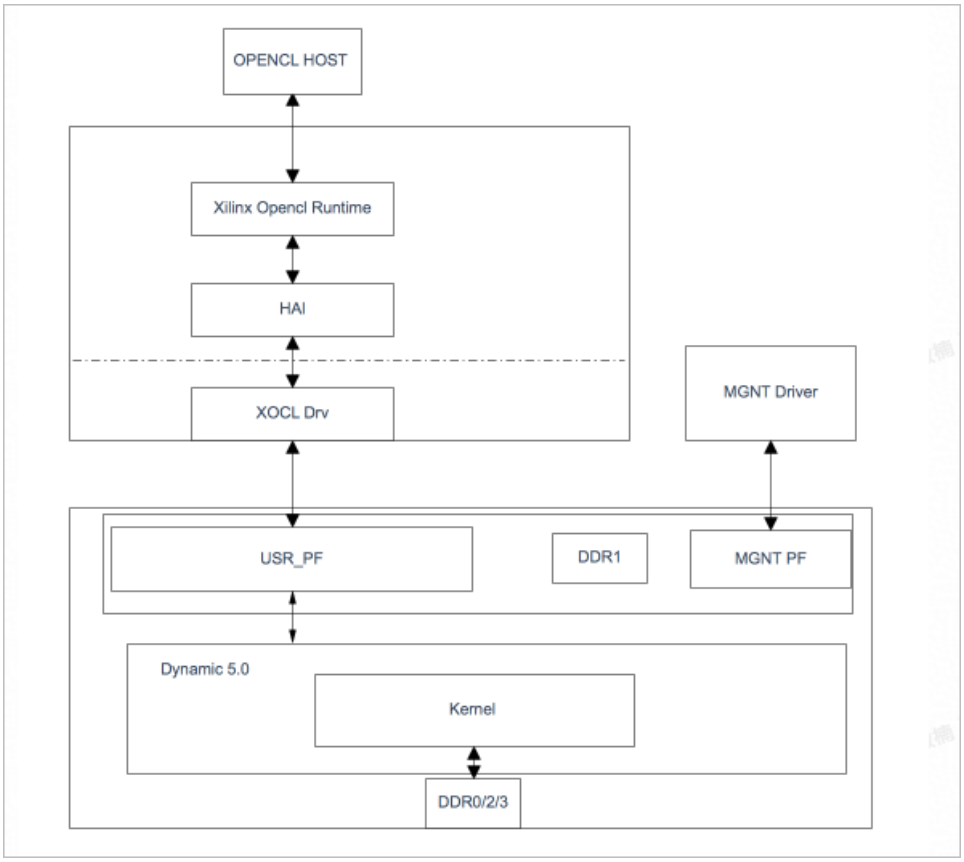
当您看到如下输出时，说明配置完成。请注意，最后一行必须为 `Verification: PASS` 。

```
[root@iZbpXXXXXZ bin]# ./host vector_add.aocx
Matrix sizes:
  A: 2048 x 1024
  B: 1024 x 1024
  C: 2048 x 1024
Initializing OpenCL
Platform: Intel(R) FPGA SDK for OpenCL(TM)
Using 1 device(s)
  skx_fpga_dcp_ddr : SKX DCP FPGA OpenCL BSP (acl0)
Using AOCX: vector_add.aocx
Generating input matrices
Launching for device 0 (global size: 1024, 2048)
Time: 40.415 ms
Kernel time (device 0): 40.355 ms
Throughput: 106.27 GFLOPS
Computing reference output
Verifying
Verification: PASS
```

2.2. f3 SDAccel开发环境介绍

FaaS f3 SDAccel开发环境以Xilinx SDAccel dynamic 5.0版本为原型，您可以基于OpenCL进行开发以及应用。本文为您简要介绍f3实例的SDAccel开发环境。

FaaS f3 SDAccel框架说明



FaaS SDAccel方案的说明如下：

- Xilinx OpenCL Runtime：Xilinx OpenCL Runtime，对用户呈现OpenCL API。
- HAL：Hardware Abstraction Layer，硬件抽象层实现OpenCL Runtime和Kernel Driver的适配及Global Memory的地址管理。
- XOCL Drv：Xilinx xocl内核驱动。
- Host Mgnt Drv：运行在主机上的管理驱动，实现FPGA kernel的加载。
- User PF：用户面PF接口，直通到虚机，为用户提供FPGA访问通道。
- Mgmt. PF：管理面PF接口，为主机访问FPGA的通道。
- kernel：OPENCL kernel模块。

FaaS f3 SDAccel开发模块说明

开发模块	说明
OPENCL标准框架	具体信息，请参见OPENCL标准框架。
Host Code开发	Xilinx UG1023
Kernel Code开发	Xilinx UG1207

FaaS f3 SDAccel使用指南

FaaS f3 SDAccel的官方使用指南，请参见f3实例OpenCL开发最佳实践。

2.3. f3实例OpenCL开发最佳实践

本文介绍如何在f3实例上使用OpenCL（Open Computing Language）制作镜像文件，并烧录到FPGA芯片中。

前提条件

- 已创建f3实例。若尚未创建f3实例，具体操作，请参见创建f3实例。

  > **说明** f3实例请使用阿里云共享给您的镜像。

- 已在f3实例所在的安全组中添加安全组规则，并放行SSH（22）端口。具体操作，请参见添加安全组规则。
- 已登录ECS管理控制台，并在f3实例的详情页上获取实例ID。
- 已创建一个OSS Bucket专门用于FaaS服务。若尚未创建OSS Bucket，具体操作，请参见创建一个OSS Bucket。
- 已获取命令行工具faasutil。若尚未获取faasutil工具，具体操作，请参见获取faasutil。
- 使用RAM用户操作FPGA时，必须先完成以下操作：
  - 创建RAM用户并为RAM用户授权。具体操作，请参见创建RAM用户和为RAM用户授权。

    您需要为RAM用户授予 `AliyunOSSFullAccess` 权限。
  - 获取AccessKey ID和AccessKey Secret。

注意事项

- 本文中所有操作都必须由同一个账号在同一地域里执行。
- 建议您使用RAM用户操作FaaS实例。您需要为FaaS管理账号创建一个角色，并授予临时权限，让FaaS管理账号能访问指定的OSS Bucket。
- 本文以2018.2版本的SDAccel开发环境为例，若您使用其他版本的SDAccel开发环境，步骤和命令可能会稍有差异。

操作步骤

在f3实例上使用OpenCL制作镜像文件并烧录到FPGA芯片中的操作步骤如下所示：

1. 步骤一：配置环境
2. 步骤二：编译二进制文件
3. 步骤三：检查打包脚本
4. 步骤四：制作镜像
5. 步骤五：下载镜像
6. 步骤六：运行Host程序

步骤一：配置环境

1. 远程连接f3实例。

   > **说明** 后面步骤中的编译工程可能会持续数小时，建议您使用screen或者nohup等方式登录，防止ssh超时退出。

2. 运行以下命令安装screen。

```
yum install screen -y
```

3. 运行以下命令进入screen。

```
screen -S f3opencl
```

4. 运行以下命令配置环境。

```
source /root/xbinst_oem/F3_env_setup.sh xocl  #每打开一个终端窗口就需要执行该命令一次
```

配置环境时，请注意以下说明：

- 配置环境主要包括安装xocl驱动、设置vivado环境变量、检查vivado license、检测aliyun-f3 sdaccel平台、授予2018.2 runtime配置。
- 如果您要运行vivado的仿真，请勿运行以上命令配置环境，您只需要单独设置vivado环境变量即可。
- 推荐您使用Makefile方式仿真。

5. 前往Xilinx，下载并安装补丁 `y2k22_patch-1.2.zip` 。

> **说明** 该补丁用于修复Xilinx年限溢出缺陷。

步骤二：编译二进制文件

编译vadd二进制文件和kernel_global_bandwidth二进制文件的操作如下所示：

示例一：编译vadd二进制文件

1. 复制*example*目录。

```
cp -rf /opt/Xilinx/SDx/2018.2/examples ./
```

2. 进入*vadd*目录。

```
cd examples/vadd/
```

3. 运行命令**cat sdaccel.mk | grep "XDEVICE="**查看XDEVICE的值，确保其配置为 `XDEVICE=xilinx_aliyun-f3_dynamic_5_0` 。

4. 按以下步骤修改*common.mk*文件。

   i. 运行**vim ../common/common.mk**命令打开该文件。

   ii. 在第61行代码（参数可能在60~62行，视您的文件而定）的末尾添加编译参数--xp param:compiler.acceleratorBinaryContent=dcp，修改后的代码如下所示。

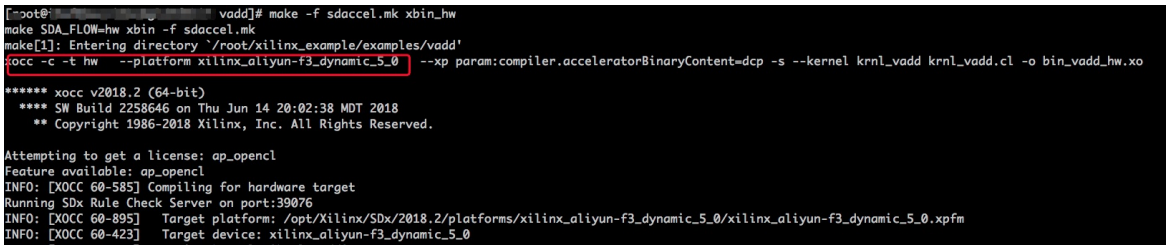
```
CLCC_OPT += $(CLCC_OPT_LEVEL) ${DEVICE_REPO_OPT} --platform ${XDEVICE} ${KERNEL_DEFS} ${KERNEL_INCS} --xp param:compiler.acceleratorBinaryContent=dcp
```

> **说明** 由于您必须向编译服务器提交DCP文件，所以需要添加--xp param:compiler.acceleratorBinaryContent=dcp编译参数，使得 `Xilinx® OpenCL™ Compiler（xocc）` 编译生成一个布局布线后的DCP文件，而不是bit文件。

5. 运行以下命令编译程序。

```
make -f sdaccel.mk xbin_hw
```

如果您看到如下界面，说明二进制文件编译已经开始。编译过程可能会持续数个小时，请您耐心等待。



示例二：编译kernel_global_bandwidth二进制文件

1. 依次运行以下命令克隆 `xilinx 2018.2 example` 。

   ```
   git clone https://github.com/Xilinx/SDAccel_Examples.git
   ```

   ```
   cd SDAccel_Examples/
   ```

   ```
   git checkout 2018.2
   ```

   > **说明** git分支必须为2018.2版本。

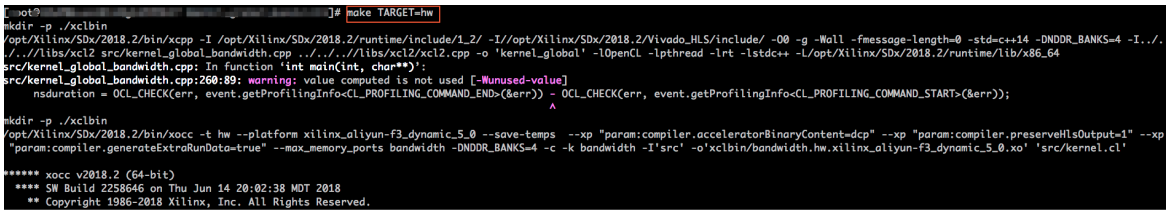
2. 运行**cd getting_started/kernel_to_gmem/kernel_global_bandwidth/**命令进入目录。
3. 按以下步骤修改*Makefile*文件。
   i. 运行**vim Makefile**命令打开该文件。
   ii. 设置 `DEVICES=xilinx_aliyun-f3_dynamic_5_0` 。
   iii. 在第33行代码中添加编译参数--xp param:compiler.acceleratorBinaryContent=dcp，修改后的代码如下所示。

   ```
   CLFLAGS +=--xp "param:compiler.acceleratorBinaryContent=dcp" --xp "param:compiler.p
   reserveHlsOutput=1" --xp "param:compiler.generateExtraRunData=true" --max_memory_po
   rts bandwidth -DNDDR_BANKS=$(ddr_banks)
   ```

4. 运行以下命令编译程序。

   ```
   make TARGET=hw
   ```

   如果您看到该界面，说明二进制文件编译已经开始。编译过程可能会持续数小时，请您耐心等待。



步骤三：检查打包脚本

运行以下命令检查打包脚本是否存在。

```
file /root/xbinst_oem/sdaccel_package.sh
```

如果返回结果中包含 `cannot open (No such file or directory)` ，说明不存在该文件，您需要运行以下命令手动下载打包脚本。

```
wget http://fpga-tools.oss-cn-shanghai.aliyuncs.com/sdaccel_package.sh
```

步骤四：制作镜像

1. 初始化faasutil工具并配置OSS环境。
   i. 运行以下命令，设置RAM账号的AccessKey信息。

   ```
   #将<HereIsYourSecretId>和<HereIsYourSecretKey>替换为您的RAM用户AK信息
   faasutil config --id=<HereIsYourSecretId> --key=<HereIsYourSecretKey>
   ```

   ii. 运行以下命令，设置FaaS服务要使用的OSS Bucket。

   ```
   #将<hereIsYourBucket>替换为您创建的OSS Bucket名称
   faasutil auth --bucket=<hereIsYourBucket>
   ```

2. 运行ls，获取后缀为 `.xclbin` 的文件名。

```
[root@ vadd]# ls
bin_vadd_hw.xclbin          krnl_vadd.cl  vadd.cpp
description.json            README.md     vadd.h
Export_Compliance_Notice.md sdaccel.mk    _xocc_krnl_vadd_bin_vadd_hw.dir
```

3. 运行以下命令，打包好二进制文件。

```
/root/xbinst_oem/sdaccel_package.sh -xclbin=/opt/Xilinx/SDx/2018.2/examples/vadd/bin_vadd_hw.xclbin
```

打包完成后，您会在同一目录下看到一个打包好的文件，如下图所示。

```
[root@ vadd]# ls
17_10_28-021904-primary.bit          krnl_vadd.cl
_SDAccel_Kernel.tar.gz               README.md
17_10_28-021904-xclbin.xml           sdaccel.mk
bin_vadd_hw.xclbin                   to_aliyun
description.json                     vadd.cpp
Export_Compliance_Notice.md          vadd.h
header.bin                           _xocc_krnl_vadd_bin_vadd_hw.dir
```

步骤五：下载镜像

本步骤介绍如何使用faasutil工具上传网表文件并下载FPGA镜像。关于faasutil命令的用法，请参见使用faasutil。

1. 依次运行以下命令，将压缩包上传到您个人的OSS Bucket，再将存放在您个人OSS Bucket中的压缩包上传到FaaS管理单元的OSS Bucket中。

```
faasutil put_object --object=bit.tar.gz --file=bit.tar.gz  #--object(Object file in oss bucket) --file(Local file need to put)
faasutil create_image --object=bit.tar.gz --fpgatype=xilinx --name=<HereIsFPGAImageName> --tags=<HereIsFPGAImageTag> --encrypted=false --shell=<hereIsShellVersionOfFPGA>
#<HereIsFPGAImageTag>可选（镜像Tag）  encrypted可选，镜像加密  <HereIsShellVersionOfFPGA>默认版本
```

```
[root@i z ~]# ./faasutil put_object --object=22_07_06-145420_SDAccel_Kernel.tar.gz --file=/root/rtl_test/f3_hdk_v10/cl_hdk/hw/22_07_06-145420_SDAccel_Kernel.tar.gz
100%
```

```
[root@i ~]# ./faasutil create_image --object=22_07_06-145420_SDAccel_Kernel.tar.gz --fpgatype=xilinx --name=eg
_test
{
  "CreateTime": "2022-07-06 15:14:49",
  "Description": "",
  "FpgaImageUniqueId": "xili                                    3",
  "Name": "eg_test",
  "RequestId": "3                                    B",
  "ShellUniqueId": "f30010",
  "State": "queueing"
}
```

2. 运行以下命令，查看FPGA镜像是否处于可下载状态。

```
faasutil list_images
```

在返回结果中：

- 如果FPGA镜像的 `"State"` 为 `"compiling"` ，表示FPGA镜像处于编译状态，您需要继续等待。
- 如果FPGA镜像的 `"State"` 为 `"success"` ，表示FPGA镜像已经可以下载，您需要找到并记录FpgaImageUniqueId，用于后续下载镜像（download_image）时使用。

```
{
  "CreateTime": "2022-07-06 15:14:49",
  "Description": "",
  "Encryption": false,
  "FpgaImageUniqueId": "xi                                    3",
  "Name": "eg_test",
  "OwnerId": "2                    1",
  "ShellUniqueId": "f30010",
  "State": "compiling",
  "Tags": "",
  "UpdateTime": "2022-07-06 15:28:12"
}
```

```
{
  "CreateTime": "2022-07-06 15:14:49",
  "Description": "",
  "Encryption": false,
  "FpgaImageUniqueId": "xi                                    3",
  "Name": "eg_test",
  "OwnerId": "2                    1",
  "ShellUniqueId": "f30010",
  "State": "success",
  "Tags": "",
  "UpdateTime": "2022-07-06 15:52:12"
}
```

3. 运行以下命令，在命令返回结果中，找到并记录fpga bdf。

```
faasutil list_instances
```

```
[root@i                    iz ~]# ./faasutil list_instances
instance_id          : i-                    i
image_id             : m-                    q
regionId             : cn-beijing
fpga bdf             : ['00:08.0']
shell version        : f30010
requestId            : x-                    55
fpga type            : xilinx
1.054(s) elapsed
```

4. 运行以下命令，下载FPGA镜像。

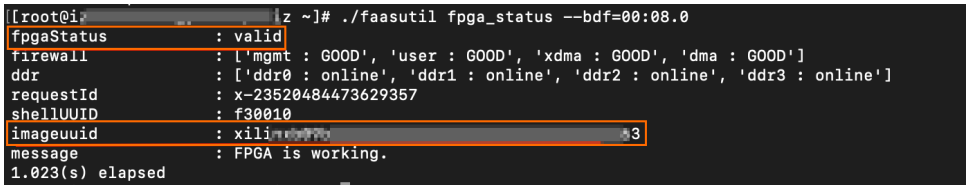
```
faasutil download_image --bdf=<fpga_bdf> --fpgaImageUniqueId=<image_uuid>  #<fpga_bdf>
由上一步所得  <image_uuid>镜像uuid
```

```
[root@iz                    iz ~]# ./faasutil download_image --bdf=00:08.0 --fpgaImageUniqueId=xilinx
{"TaskId":3861,"RequestId":"41                    3","FpgaImageUniqueId":"xili                    3","Instan
ceId":"i-2z                    i","BDFInECS":"","TaskStatus":"committed"}
status               : operating
info                 : download task is operating
requestId            : x-
1.104(s) elapsed
```

5. 运行以下命令，查看镜像是否下载成功。

```
faasutil fpga_status --bdf=<fpga_bdf>
```

以下为返回结果示例。如果显示的imageuuid与您获取的FpgaImageUniqueId一致，并且 `fpgaStatus` 参数显示 `valid` ，说明镜像下载成功。



步骤六：运行Host程序

1. 运行以下命令配置环境。

```
source /root/xbinst_oem/F3_env_setup.sh xocl  #每打开一个终端窗口就需要执行该命令一次
```

2. 配置*sdaccel.ini*文件。

在host二进制文件所在目录下，运行**vim sdaccel.ini**命令，创建*sdaccel.ini*文件并输入下列内容。

```
[Debug]
profile=true
[Runtime]
runtime_log = "run.log"
hal_log = hal.log
ert=false
kds=false
```

3. 运行host。
   - vadd运行命令如下：

```
make -f sdaccel.mk host
```

```
./vadd bin_vadd_hw.xclbin
```

   - kernel_global_bandwidth运行命令如下：

```
./kernel_global
```

如果返回结果中出现 `Test Passed` ，说明测试通过。

常用操作

这里介绍FPGA实例的部分常用操作。

任务	命令
查看帮助文档	`make -f ./sdaccel.mk help`
软件仿真	`make -f ./sdaccel.mk run_cpu_em`
硬件仿真	`make -f ./sdaccel.mk run_hw_em`

任务	命令
只编译host代码	`make -f ./sdaccel.mk host`
编译生成可以下载的文件	`make -f sdaccel.mk xbin_hw`
清理工作目录	`make -f sdaccel.mk clean`
强力清除工作目录	`make -f sdaccel.mk cleanall`

说明

- 仿真时只需要按照Xilinx标准流程操作，不需要配置F3_env_setup环境。
- SDAccel runtime和SDAccel开发平台已在阿里云f3官方镜像中提供。您也可以点击后面的链接直接下载SDAccel runtime和SDAccel开发平台。

3.f3实例Vitis 2020.1使用最佳实践

本文介绍如何在f3实例上使用Vitis 2020.1制作镜像文件，并烧录到FPGA芯片中。

前提条件

- 已创建满足如下要求的f3实例。具体操作，请参见创建f3实例。
  - 已提交工单获取镜像**FaaS_F30010_VITIS_2020_1**，并在创建实例时，选择该镜像。
  - 建议系统盘不小于120 GiB。
  - 已为f3实例分配公网IP。
  - 已在f3实例所在安全组中添加规则并放行SSH（22）端口。
- 已在ECS控制台f3实例的详情页上，获取实例ID。
- 已创建一个OSS Bucket专门用于FaaS服务。

  Bucket与f3实例必须属于同一个账号、同一个地域。若尚未创建，请参见创建一个OSS Bucket。
- 使用RAM用户操作FPGA，必须先完成以下操作：
  - 创建RAM用户并授权，详情请参见创建RAM用户及为RAM用户授权。

    您需要为RAM用户授予的权限为：AliyunECSReadOnlyAccess、AliyunOSSFullAccess和AliyunRAMFullAccess。
  - 授权FaaS服务角色，授权页面请参见授权FaaS服务角色。
  - 获取AccessKey ID和AccessKey Secret。

操作步骤

在f3实例上，使用Vitis 2020.1制作镜像文件，并烧录到FPGA芯片中的操作步骤如下：

1. 步骤一：远程连接实例
2. 步骤二：初始化软件环境
3. 步骤三：创建工程
4. 步骤四：仿真
5. 步骤五：制作镜像
6. 步骤六：上板验证

步骤一：远程连接实例

FaaS_F30010_VITIS_2020_1镜像已配置了桌面环境及VNC Server，建议您通过VNC Server远程连接f3实例。具体操作，请参见通过密码认证登录Linux实例。

步骤二：初始化软件环境

您每创建一个f3实例，均需要执行以下操作初始化当前软件环境。

1. 运行以下命令，通过VNC Server指定环境分辨率。

   ```
   vncserver -geometry 2560x1440
   ```

   回显信息如下：

```
New '                    :1 (root)' desktop is                    :1

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/                    :1.log
```

2. 运行以下命令，对FaaS_F30010_VITIS_2020_1镜像中已预安装的FaaS平台使用VITIS所需的软件环境进行初始化。

```
source /root/faasTools/vitis_setup.sh
```

回显信息如下，表示初始化完成。

```
XILINX_XRT      : /opt/xilinx/xrt
PATH            : /opt/xilinx/xrt/bin:/opt/Xilinx/DocNav:/opt/Xilinx/Vivado/2020.1/bin:/opt/Xilinx/Vitis/2020.1/bin:/opt/Xilinx/
Vitis/2020.1/gnu/microblaze/lin/bin:/opt/Xilinx/Vitis/2020.1/gnu/arm/lin/bin:/opt/Xilinx/Vitis/2020.1/gnu/microblaze/linux_toolc
hain/lin64_le/bin:/opt/Xilinx/Vitis/2020.1/gnu/aarch32/lin/gcc-arm-linux-gnueabi/bin:/opt/Xilinx/Vitis/2020.1/gnu/aarch32/lin/gc
c-arm-none-eabi/bin:/opt/Xilinx/Vitis/2020.1/gnu/aarch64/lin/aarch64-linux/bin:/opt/Xilinx/Vitis/2020.1/gnu/aarch64/lin/aarch64-
none/bin:/opt/Xilinx/Vitis/2020.1/gnu/armr5/lin/gcc-arm-none-eabi/bin:/opt/Xilinx/Vitis/2020.1/tps/lnx64/cmake-3.3.2/bin:/opt/Xi
linx/Vitis/2020.1/cardano/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
LD_LIBRARY_PATH : /opt/xilinx/xrt/lib:
PYTHONPATH      : /opt/xilinx/xrt/python:
```

步骤三：创建工程

环境准备完成后，您即可通过VITIS启动工程，并在GUI界面下创建工程。

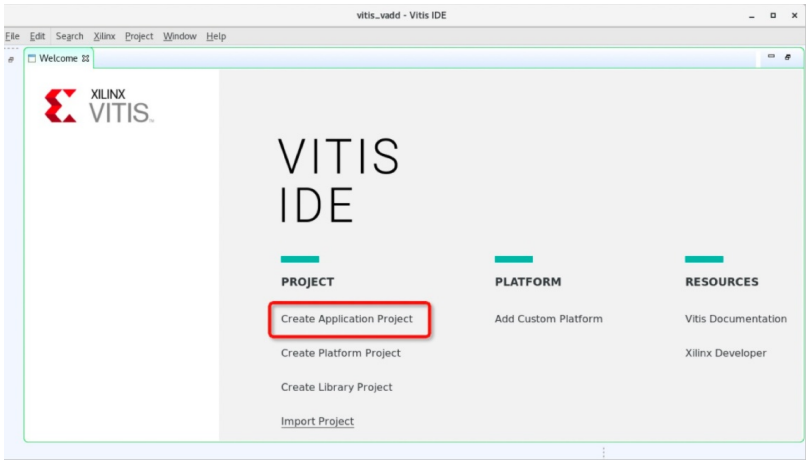
1. 执行 `vitis` 命令启动工程。

回显信息如下，表示工程启动成功。

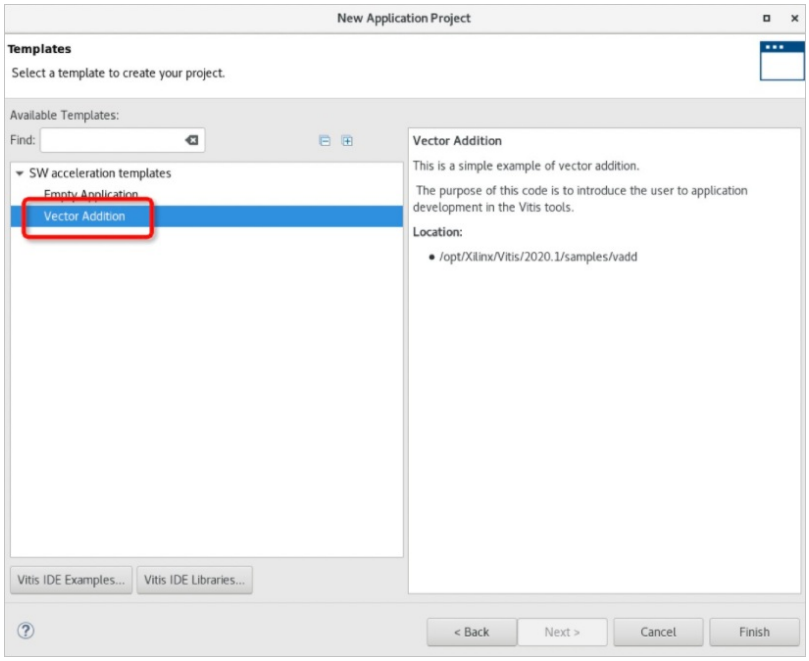
```
****** Xilinx Vitis Development Environment
****** Vitis v2020.1 (64-bit)
  **** SW Build 2902540 on Wed May 27 19:55:13 MDT 2020
    ** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.

Launching Vitis with command /opt/Xilinx/Vitis/2020.1/eclipse/lnx64.o/eclipse -vmargs -Xms64m -Xmx4G -Dorg.eclipse.swt.internal.
gtk.cairoGraphics=false -Dosgi.configuration.area=@user.home/.Xilinx/Vitis/2020.1 --add-modules=ALL-SYSTEM --add-modules=jdk.inc
ubator.httpclient --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.desktop/sun.swing=ALL-UNNAMED --add-opens=java.des
ktop/javax.swing=ALL-UNNAMED --add-opens=java.desktop/javax.swing.tree=ALL-UNNAMED --add-opens=java.desktop/javax.swing.plaf.bas
ic=ALL-UNNAMED --add-opens=java.desktop/javax.swing.plaf.synth=ALL-UNNAMED --add-opens=java.desktop/com.sun.awt=ALL-UNNAMED --ad
d-opens=java.desktop/sun.awt.X11=ALL-UNNAMED &
```

2. 工程启动成功后，请执行以下步骤，在VITIS的GUI界面创建工程。
   i. 在打开的VITIS窗口，单击PROJECT下的Create Application Project。



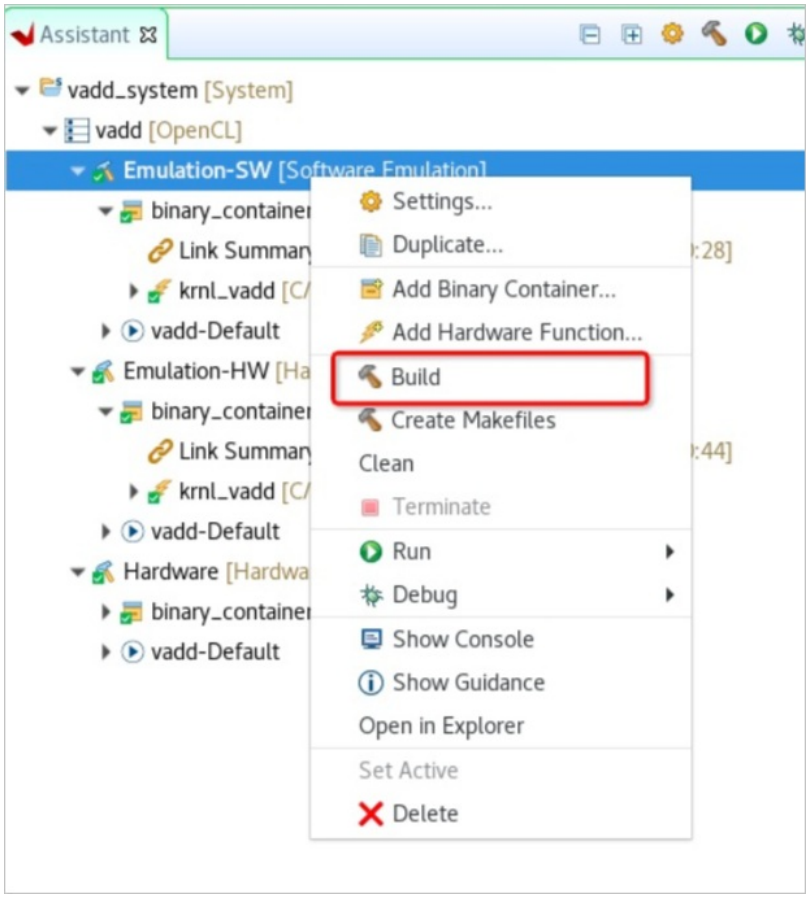
ii. 在打开的**New Application Project**对话框中，单击左侧区域的**SW acceleration templates > Vector Addition**，然后单击右下角**Finish**。



步骤四：仿真

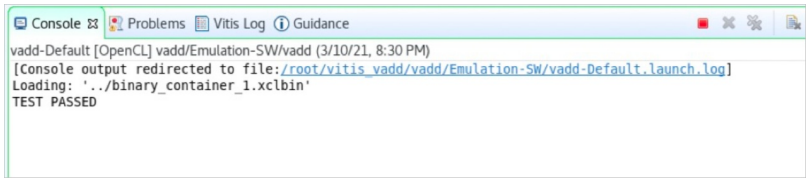
VITIS支持Emulation-SW、Emulation-HW两种仿真形式。

1. 在VITIS的**Assistant**页签中，选择**vadd_system > vadd > Emulation-SW [Software Emulation]**。
2. 在**Emulation-SW [Software Emulation]**上单击鼠标右键，选择**Build**。



3. 在Emulation-SW [Software Emulation]上单击鼠标右键，选择Run > Default进行机型仿真。

   运行结果如下所示：



步骤五：制作镜像

由于VITIS默认生成的镜像文件中包含.bit格式的文件，而FaaS要求上传的文件格式为.dcp，因此您需要先进行相关设置，再制作FPGA镜像。

1. 在VITIS的Assistant页签的右上角，单击 图标。
2. 在打开的Project Settings对话框中，选择vadd_system > vadd。
3. 在右侧的vadd区域中，将V++ linker options配置项设置为以下内容。然后单击Apply，再单击Apply and Close。

   ```
   --advanced.param compiler.acceleratorBinaryContent=dcp
   ```

4. 返回VITIS的Assistant页签，选择vadd_system > vadd > Hardware [Hardware]。
5. 在Hardware [Hardware]上单击鼠标右键，选择Build进行镜像制作。

   制作过程会持续数个小时，制作完成后，您可以进入工程目录的Hardware目录下，然后执行 `ls` 命令，查看生成的host可执行二进制文件vadd及镜像文件binary_container_1.xclbin。如下所示：

```
[root@iz2zec7rvsxxxxxxx Hardware]# ls
a.xclbin                                guidance.html
binary_container_1.build                guidance.pb
binary_container_1-krnl_vadd-compile.cfg  makefile
binary_container_1-link.cfg             package.build
binary_container_1.ltx                  package.cfg
binary_container_1.mdb                  src
binary_container_1.xclbin               vadd
binary_container_1.xclbin.info          vadd_Hardware.build.ui.log
binary_container_1.xclbin.link_summary  v++_package.log
binary_container_1.xclbin.sh            v++.package_summary
common-config.cfg                       xcd.log
```

6. 执行如下命令，生成用于上传镜像的压缩文件。

```
[root@iz2zec7rvsxxxxxxx Hardware]# vitis_xclbin_split.sh binary_container_1.xclbin
```

执行结果如下：

```
XRT Build Version: 2.6.0 (2020.1)
       Build Date: 2021-03-08 10:50:41
          Hash ID: 80107ebc7376dafc8e1c9f5043c81c6f1dcc9dbb
--------------------------------------------------------------------------------
Warning: The option '--output' has not been specified. All operations will
         be done in memory with the exception of the '--dump-section' command.
--------------------------------------------------------------------------------
Reading xclbin file into memory.  File: binary_container_1.xclbin
Section: 'BITSTREAM'(0) was successfully written.
Format: RAW
File  : 'faas20210311-092706.dcp'
Leaving xclbinutil.
XRT Build Version: 2.6.0 (2020.1)
       Build Date: 2021-03-08 10:50:41
          Hash ID: 80107ebc7376dafc8e1c9f5043c81c6f1dcc9dbb
--------------------------------------------------------------------------------
Warning: The option '--output' has not been specified. All operations will
         be done in memory with the exception of the '--dump-section' command.
--------------------------------------------------------------------------------
Reading xclbin file into memory.  File: binary_container_1.xclbin
Section: 'EMBEDDED_METADATA'(2) was successfully written.
Format: RAW
File  : 'faas20210311-092706.xml'
Leaving xclbinutil.
to_aliyun/
to_aliyun/faas20210311-092706.xml
to_aliyun/faas20210311-092706.dcp
Generate Image :Image20210311-092706.tar.gz
```

如上所示，最终生成的**Image20210311-092706.tar.gz**文件可用于上传制作镜像。具体操作，请参见使用faasutil。

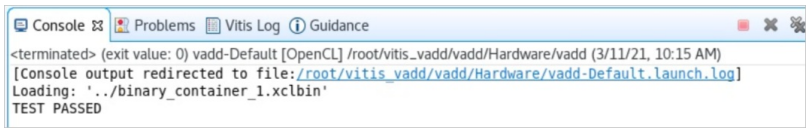
> **说明** FaaS_F30010_VITIS_2020_1镜像中已预装了faasutil工具，您可以执行 `source /root/faasTools/vitis_setup.sh` 命令后使用。

步骤六：上板验证

通过FaaS_F30010_VITIS_2020_1镜像中预装的faasutil工具完成镜像制作及镜像加载后，您可以使用命令行或GUI两种方式进行上板验证。

- 使用GUI验证：
  i. 在VITIS的**Assistant**页签，选择**vadd_system > vadd > Hardware [Hardware]**。
  ii. 在**Hardware [Hardware]**上单击鼠标右键，选择**Run > Default**。

  运行结果如下所示，表示将FPGA镜像已烧录到FPGA芯片。

- 使用命令行验证：

  在工程目录的Hardware目录下，执行如下命令进行验证：

  ```
  [root@iz2zec7rvsxxxxxxx Hardware]# ./vadd binary_container_1.xclbin
  ```

  运行结果如下所示，表示将FPGA镜像已烧录到FPGA芯片。

  ```
  Loading: 'binary_container_1.xclbin'
  TEST PASSED
  ```

4.AI推理

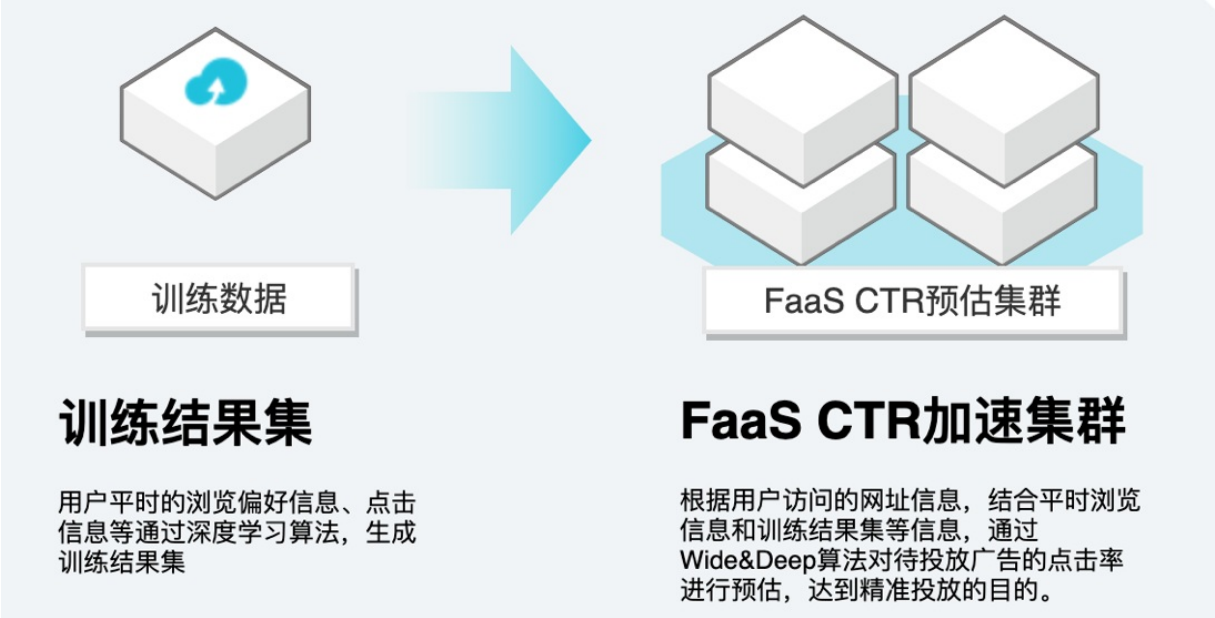
4.1. FPGA云服务器加速广告CTR预估

本实践方案基于阿里云FPGA计算型实例，完整演示加速广告点击率预估（Click-Through Rate）的场景，帮助您快速理解阿里云FPGA云服务器。

解决的问题

- 加速广告点击率预估的计算过程。
- 降低计算成本。

部署架构图



选用的产品

- FPGA云服务器

  FPGA云服务器是一款提供了现场可编程门阵列（FPGA）的计算实例，基于阿里云弹性计算框架，用户可以几分钟内轻松创建FPGA实例，创建自定义的专用硬件加速器。由于FPGA硬件的可重配特性，用户可以对已创建的FPGA硬件加速应用，进行快速擦写和重配，达到低时延硬件与弹性伸缩最好的结合。

  更多关于FPGA云服务器的介绍，请参见FPGA云服务器详情页。

- 云服务器ECS

  云服务器（Elastic Compute Service）是阿里云提供的性能卓越、稳定可靠、弹性扩展的IaaS（Infrastructure as a Service）级别云计算服务。云服务器ECS免去了您采购IT硬件的前期准备，让您像使用水、电、天然气等公共资源一样便捷、高效地使用服务器，实现计算资源的即开即用和弹性伸缩。阿里云ECS持续提供创新型服务器，解决多种业务需求，助力您的业务发展。

  更多关于云服务器ECS的介绍，请参见云服务器ECS产品详情页。

- 专有网络VPC

专有网络VPC帮助您基于阿里云构建出一个隔离的网络环境，并可以自定义IP地址范围、网段、路由表和网关等；此外，也可以通过专线、VPN、GRE等连接方式实现云上VPC与传统IDC的互联，构建混合云业务。

更多关于专有网络VPC的介绍，请参见专有网络VPC产品详情页。

- 对象存储OSS

  阿里云对象存储服务（Object Storage Service），是阿里云提供的海量、安全、低成本、高可靠的云存储服务。其数据设计持久性不低于99.9999999999%（12个9），服务设计可用性（或业务连续性）不低于99.995%。

  更多关于对象存储OSS的介绍，请参见对象存储OSS产品详情页。

详细信息

点击查看最佳实践详情

更多最佳实践

点击查看更多阿里云最佳实践

5.图片转码

5.1. FPGA云服务器加速图片和视频转码

本实践基于阿里云FPGA计算型实例，完整演示加速视频转码和图片转码两个场景，帮助您快速理解阿里云FPGA云服务器。

场景描述

两个场景中的实现效果如下：

- 加速视频转码：视频编码IP的编译、镜像下载到FPGA以及将MP4视频文件编码为H.265的全过程。
- 加速图片转码：使用基于FPGA的图像格式转换程序，将JPEG格式的图片转换为WebP格式，并进行缩放。

解决的问题

- 提供高画质低码率的实时转码技术方案。
- 提高图片和视频转码效率。
- 降低传输带宽、转码和存储成本。

部署架构图

选用的产品

- FPGA云服务器

  FPGA云服务器是一款提供了现场可编程门阵列（FPGA）的计算实例，基于阿里云弹性计算框架，用户可以几分钟内轻松创建FPGA实例，创建自定义的专用硬件加速器。由于FPGA硬件的可重配特性，用户可以对已创建的FPGA硬件加速应用，进行快速擦写和重配，达到低时延硬件与弹性伸缩最好的结合。

  更多关于FPGA云服务器的介绍，请参见FPGA云服务器详情页。

- 云服务器ECS

  云服务器（Elastic Compute Service）是阿里云提供的性能卓越、稳定可靠、弹性扩展的IaaS（Infrastructure as a Service）级别云计算服务。云服务器ECS免去了您采购IT硬件的前期准备，让您像使用水、电、天然气等公共资源一样便捷、高效地使用服务器，实现计算资源的即开即用和弹性伸缩。阿里云ECS持续提供创新型服务器，解决多种业务需求，助力您的业务发展。

  更多关于云服务器ECS的介绍，请参见云服务器ECS产品详情页。

- 专有网络VPC

  专有网络VPC帮助您基于阿里云构建出一个隔离的网络环境，并可以自定义IP地址范围、网段、路由表和网关等；此外，也可以通过专线、VPN、GRE等连接方式实现云上VPC与传统IDC的互联，构建混合云业务。

  更多关于专有网络VPC的介绍，请参见专有网络VPC产品详情页。

- 对象存储OSS

  阿里云对象存储服务（Object Storage Service），是阿里云提供的海量、安全、低成本、高可靠的云存储服务。其数据设计持久性不低于99.9999999999%（12个9），服务设计可用性（或业务连续性）不低于99.995%。

  更多关于对象存储OSS的介绍，请参见对象存储OSS产品详情页。

详细信息

点击查看最佳实践详情

更多最佳实践

点击查看更多阿里云最佳实践

6.图像处理

6.1. 在f1实例中使用CTAccel图像处理（CIP）加速器

如果您的某些业务要求ECS实例内不能使用AccessKey，您可以为ECS实例配置RAM角色，ECS实例中部署的应用需要通过实例元数据获取临时授权Token，才可以访问指定资源。本教程介绍如何以无AccessKey的方式访问f1实例资源，从而便捷地为f1实例加载CTAccel CIP加速器。

前提条件

- 已注册阿里云账号。如还未注册，请先完成账号注册。
- 已完成实名认证。如还未认证，请先完成实名认证。
- 联系服务商完成以下准备工作，联系方式请参见CTAccel CIP加速器服务商。
  - 获取开发套件包。
  - 了解支持的镜像类型，并为ECS实例准备自定义镜像。
  - 关联ECS实例镜像和FPGA镜像。

    > **说明** ECS实例镜像必须为自定义镜像，申请关联镜像时请提供镜像ID。

  - 获取FPGA镜像的FPGAImageUUID。

背景信息

CTAccel图像处理（CIP）加速器是一款基于FPGA的图像处理加速解决方案，可通过将计算负载从CPU转移至FPGA，显著提高图像处理及图像分析的性能。

操作步骤

- 步骤一：创建RAM角色并授予权限
- 步骤二：为实例配置RAM角色和标签
  - 步骤二：创建f1实例并配置RAM角色和标签
  - 步骤二：为已有f1实例配置RAM角色和标签
- 步骤三：安装faascmd
- 步骤四：配置CTAccel环境

步骤一：创建RAM角色并授予权限

创建RAM角色并配置RAM权限用于实现无AccessKey访问。更多实例RAM角色说明，请参见实例RAM角色概述。

1. 使用阿里云账号登录RAM控制台。
2. 创建权限策略。
   i. 单击**权限管理 > 权限策略**。
   ii. 单击**创建权限策略**。

iii. 填写策略名称。

本示例中使用faasEcsRamPolicy。

iv. 配置模式选择**脚本配置**。

v. 填入权限策略内容。

以下权限策略用于允许受信服务查看绑定对应标签的ECS实例资源。您可以自定义标签的键和标签值，本示例中使用faas和image。

```
{
    "Version": "1",
    "Statement": [
        {
            "Action": "ecs:DescribeInstances",
            "Effect": "Allow",
            "Resource": "acs:ecs:*:*:*",
            "Condition": {
                "StringEquals": {
                    "ecs:tag/faas": "image"
                }
            }
        }
    ]
}
```

vi. 单击**确定**。

3. 创建RAM角色。

如果已存在AliyunFAASDefaultRole，您可以在已有基础上增加信任策略的内容并授予权限。

> 注意 如果其它子账号也使用了AliyunFAASDefaultRole，请谨慎修改。

i. 在左侧导航栏，选择**身份管理 > 角色**。

ii. 在**角色**页面，单击**创建角色**。

iii. 在**创建角色**面板，选择可信实体类型为**阿里云服务**，然后单击**下一步**。

iv. 选择角色类型为**普通服务角色**。

v. 填写角色名称，必须是AliyunFAASDefaultRole。

vi. 受信服务选择**云服务器**。

vii. 单击**完成**。

viii. 单击**关闭**。

4. 修改RAM角色的信任策略。

i. 找到创建的AliyunFAASDefaultRole，在**角色名称**列，单击名称。

ii. 单击**信任策略管理**页签。

iii. 单击**修改信任策略**。

iv. 按以下信任策略修改，然后单击**确定**。

以下信任策略用于允许ECS服务和FaaS服务扮演该RAM角色。

```
{
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ecs.aliyuncs.com",
                    "faas.aliyuncs.com"
                ]
            }
        }
    ],
    "Version": "1"
}
```

5. 授予RAM角色权限。
   i. 单击**权限管理**页签。
   ii. 单击**精确授权**。
   iii. 单击**自定义策略**。
   iv. 输入策略名称。

   例如本示例中的faasEcsRamPolicy。
   v. 单击**确定**。
   vi. 单击**关闭**。

   授权后，ECS服务和FaaS服务可以通过扮演RAM角色AliyunFAASDefaultRole，获取查看绑定标签`faas:image`的ECS实例资源的权限，无需再提供AccessKey。

步骤二：创建f1实例并配置RAM角色和标签

请参见以下步骤创建一台f1实例，并在创建过程中配置RAM角色和标签。

1. 登录ECS管理控制台。
2. 在左侧导航栏，选择**实例与镜像 > 实例**。
3. 单击**创建实例**。
4. 完成实例创建。

   下表中只列出需要重点关注的配置项，请视需要完成其它配置，更多说明请参见使用向导创建实例。

配置页面	配置项	说明
	实例	○ 如果处理JPEG和WebP格式的图片，建议选择ecs.f1-c8f1.2xlarge。 ○ 如果处理HEIF格式的图片，建议选择ecs.f1-c28f1.7xlarge。 您可以前往ECS实例可购买地域，查看实例在各地域的可购情况。

配置页面	配置项	说明
基本配置		
	镜像	选择您创建好的自定义镜像。
系统配置	实例RAM角色	AliyunFAASDefaultRole
分组设置	标签	权限策略中定义的标签键和标签值，本示例中使用 `faas:image` 。

5. 在实例列表中，找到f1实例，然后在**实例ID/名称**列中，单击实例ID。

   确认已成功为f1实例授予RAM角色并绑定标签。

步骤二：为已有f1实例配置RAM角色和标签

如果您已经创建了f1实例，且f1实例的实例规格和镜像满足要求，请参见以下步骤配置RAM角色和标签。

1. 登录ECS管理控制台。
2. 在左侧导航栏，选择**实例与镜像 > 实例**。
3. 找到待操作的f1实例。
4. 为f1实例授权RAM角色。
   i. 在实例列表中，找到待操作f1实例，然后单击**更多 > 实例设置 > 授予/收回RAM角色**。
   ii. 选择RAM角色AliyunFAASDefaultRole，然后单击**确定**。
5. 为f1实例绑定标签。
   i. 在实例列表中，找到待操作f1实例，然后选择**更多 > 实例设置 > 编辑标签**。

ii. 输入上文信任策略中定义的标签键和标签值，然后单击**确定**。

iii. 标签绑定完毕后，单击**确定**。

6. 在实例列表中，找到f1实例，然后在**实例ID/名称**列中，单击实例ID。

   确认已成功为f1实例授予RAM角色并绑定标签。

步骤三：安装faascmd

faascmd是阿里云FPGA云服务器（FaaS平台）提供的命令行工具，是基于阿里云Python SDK开发的脚本，您可以通过faascmd命令方便地操作FaaS平台的资源。

1. 远程连接Linux实例。
2. 检查是否安装了Python，且确保Python版本号必须为2.7.x。

```
python -V
```

   返回Python版本号表示已安装Python。如果版本过低，请先升级至2.7.x。如果版本过高，请使用2.7.x版本，否则可能存在语法兼容问题。

```
[root@ ~]# python -V
Python 2.7.5
```

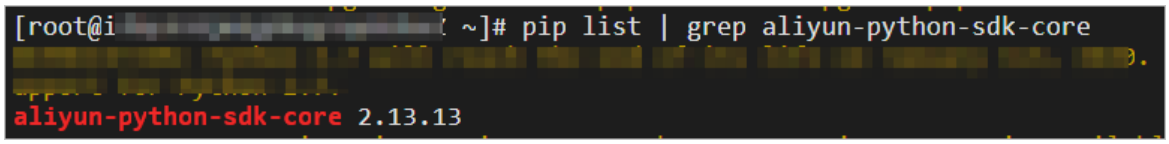
3. 安装Python模块。

```
pip install oss2
pip install aliyun-python-sdk-core
pip install aliyun-python-sdk-faas
pip install aliyun-python-sdk-ram
```

4. 检查aliyun-python-sdk-core版本号，确保版本号为2.11.0或以上版本。

```
pip list | grep aliyun-python-sdk-core
```

如果版本过低，请先升级至2.11.0或以上版本。



5. 下载faascmd。

```
curl -o /bin/faascmd http://fpga-tools.oss-cn-shanghai.aliyuncs.com/faascmd-sts
```

6. 为faascmd添加可执行权限。

```
chmod +x /bin/faascmd
```

步骤四：配置CTAccel环境

您可以从服务商提供的开发套件包中获取OPAE和CIP SDK安装包。

1. 安装图片lib库。

```
yum install libjpeg-turbo-* libpng-* libwebp-* libtiff-* jasper jasper-devel
```

2. 切换至OPAE安装包目录，运行以下命令安装OPAE。

```
yum install opae-*
```

3. 切换至CIP SDK安装包目录，运行以下命令安装CIP SDK。

```
yum install accel-*
```

4. 下载初始化配置脚本。

```
curl -o ctaccel-aliyun-init.sh https://fpga-tools.oss-cn-shanghai.aliyuncs.com/ctaccel-aliyun-init.sh
```

5. 加载您从服务商处获取的FPGA镜像，并初始化软件环境。

```
sh ctaccel-aliyun-init.sh <yourFPGAImageUUID>
```

如果出现报错 `HTTP Status: 404 Error:IMAGE NOT FOUND The fpga image you specify is not found!` ，原因为未关联ECS镜像和FPGA镜像。请联系服务商进行关联操作。其它faascmd报错的处理方法，请参见faascmd工具FAQ。

6. 查看acceld service状态。

```
service acceld status
```

配置成功后，acceld service需要为running状态。

> **说明** 请记录服务启动日期用于查看日志文件。

```
[root@iZ                    ~]# service acceld status
● acceld.service - SYSV: Acceld server
   Loaded: loaded (/etc/rc.d/init.d/acceld; bad; vendor preset: disabled)
   Active: active (running) since Tue 2020-01-21 10:53:10 CST; 52s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 27615 ExecStop=/etc/rc.d/init.d/acceld stop (code=exited, status=0/SUCCESS)
  Process: 27851 ExecStart=/etc/rc.d/init.d/acceld start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/acceld.service
           └─27858 /usr/accel/service/sbin/main/acceld_service --daemonize --pid-file /var/run
/acceld.pid --root /usr/accel/service/sbin --log-file /var/log --log-level info --heap_cfg 0

Jan 21 10:53:10 iZ                    systemd[1]: Starting SYSV: Acceld server...
Jan 21 10:53:10 iZ                    acceld[27851]: Starting Accel service: search_file:fin
d:/usr/accel/service/sbin/plugins/log_impl/boost.properties
Jan 21 10:53:10 iZ                    acceld[27851]: parse_properties:parse:/usr/accel/servi
ce/sbin/plugins/log_impl/boost.properties
Jan 21 10:53:10 iZ                    acceld[27851]: Set rlimit to RLIM_INFINITY(no limit).
Jan 21 10:53:10 iZ                    acceld[27851]: [  OK  ]
Jan 21 10:53:10 iZ                    acceld[27851]: 27858 (process ID) old priority 0, new
priority -20
Jan 21 10:53:10 iZ                    systemd[1]: Started SYSV: Acceld server.
```

7. 查看日志文件状态。

```
cat /var/log/acceld-<年-月-日>.log
```

配置成功后，日志文件中需要正确读取版本号。图中版本号仅为示例，版本号应和购买时服务商提供的版本一致。

```
[root@iZ                    ~]# cat /var/log/acceld-2020-01-21.log
2020-01-21 10:40:00 [0x00                ] [info] domainSession::init:Serving:/dev/shm/acceld
2020-01-21 10:40:00 [0x00                ] [info] opae_afc_init m_handles = 0x14b51b0

2020-01-21 10:40:00 [0x00                ]
2020-01-21 10:40:00 [0x00                ]
2020-01-21 10:40:00 [0x00                ]
2020-01-21 10:40:00 [0x00                ]
2020-01-21 10:40:00 [0x00                ]
2020-01-21 10:40:00 [0x00                ]
2020-01-21 10:40:00 [0x00                ] [info] JpegWorker_Creek::init_device:dev version:2081530-600
2020-01-21 10:40:00 [0x00                ]
2020-01-21 10:40:00 [0x00                ]
```

相关文档

- AttachInstanceRamRole
- TagResources