ALIBABA CLOUD

Alibaba Cloud

容器服务Kubernetes版
基因计算服务AGS用户指南

文档版本：20201109

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.AGS概览

阿里云基因计算AGS（Alibaba Cloud Genomics Service）是由阿里云推出极速、低成本、高精度的基因组测序二级分析的云服务，主要提供以容器平台为基础的生物信息工作流服务和无需搭建集群，开箱即用的加速API服务。本文介绍什么是AGS及其应用优势。

什么是AGS

AGS主要应用于基因组测序二级分析，通过AGS加速API只需要15分钟即可完成一个30X WGS的基因比对、排序、去重、变异检测全流程，相比经典流程可加速120倍，比目前全球最快的FPGA/GPU方案仍能提速2~4倍。

通过分析个体基因序列的突变机制，可为遗传病检测、肿瘤筛查等提供有力支撑，未来将在临床医学和基因诊断方面发挥巨大作用。人类全基因组有约30亿个碱基对，一个30X的WGS测序数据量大约在100GB。AGS在计算速度、精准度、成本、易用性、与上游测序仪的整合度上具有极大优势，同时适用于DNA的SNP/INDEL以及CNV结构变异检测，以及DNA/RNA病毒检测等场景。

AGS的优势

- 极速、精准。

  经过实际测试，整套方案在15分钟内完成了8组30X WGS样本二级分析处理。在保证精度的前提下，实现15分钟对7200亿碱基拼装、排序、去重、变异检测，完成基因检测全流程120倍加速。且通过NA12878测试数据集与金标准VCF比较，二级分析的精度高于或等于BWA-0.7.17/GATK 4.1.3的数据产出，SNP精度到达99.80%。

数据集：30X NA12878			
SNP	**RECALL**	**PRECISION**	**F1**
GATK 4.1版	99.86%	99.79%	99.82%
AGS版	99.86%	99.80%	99.83%
INDEL	**RECALL**	**PRECISION**	**F1**
GATK 4.1版	99.28%	99.70%	99.49%
AGS版	99.27%	99.68%	99.47%

- 成本大幅优化。

  阿里云ACK/AGS提供云上PaaS加速能力，以混合云方式协助华大基因完成自主测序仪大批量下机数据二级分析。同时实现二级分析计算行业内低成本，缩短交付周期95%。

- 适用场景广。

  在保证分析通量的同时满足灵活性需求，可根据不同平台和数据定制分析流程。为各大测序服务商、研究机构等提供更简单更高效的存储、自动化分析、数据传输、项目协作以及生物信息工具开发等方面的解决方案。

  AGS能够提供Kubernetes-native工作流机制，帮助用户在Kubernetes集群上运行支持DAG的工作流。在处理基因计算，数据计算等场景具有良好的通用性。

- 简单易用。

  AGS凭借云端的自动伸缩特性，实现大规模弹性调度计算。在使用上，该方案用户无需关心基因数据处理过程中的计算资源、处理逻辑、数据缓存等细节，只需将下机数据（FASTQ文件）上传至OSS，以及授权Bucket给AGS服务，即可高效、快速完成整个数据分析流程，并将结果数据上传到用户期望的存储空间。

相关文档

除了上述所提及特点，AGS产品还成功解决了工作流程组装管理，海量数据存储、迁移与传输、安全合规等行业痛点问题。详情请参见以下文档。

- 通过AGS处理全基因组测序WGS
- 通过AGS排查病毒序列
- 通过AGS分析肿瘤样本
- 工作流在Kubernetes集群中的实践
- AGS命令行帮助

2.通过AGS进行群体多样本联合变异检测

对于一个家族或者相似的群体来说，变异往往具有一定的相似性。通过对一个家族或者群体进行联合变异的检测，可以有效提高变异检测的准确率。本文介绍如何通过命令行调用AGS服务API进行群体联合变异检测。

前提条件

已提交阿里云基因服务公测申请，加入公测白名单。

> **说明** 如果您使用的是子账号，公测申请时请提供子账号的UID。请登录账号管理控制台查询UID。

背景信息

通过AGS全基因组的变异检测（WGS）生成具备更多未突变点位覆盖信息的GVCF，详情请参见GVCF。通过AGS的merge工具生成多样本的GVCF，详情请参见CombineGVCFs，为后续的变异矫正提供数据支撑。

AGS

准备工作

1. 下载和安装AGS，请参见AGS命令行帮助。
2. 配置AGS。

   ```
   ags config init
   ```

3. 确认Bucket的Owner。

   确保指定Bucket的Owner是您当前的账号，否则建议您新建一个Bucket，请参见创建存储空间。

   > **说明** 如果您使用的是子账号，需要为子账号授予AliyunOSSFullAccess权限，请参见为RAM用户授权。同时建议您重新创建OSS Bucket，以确保该账号是所使用Bucket的Owner。

   ```
   ossutil stat oss://<your new bucket name>
   ```

4. 授权AGS服务读写权限和GetBucketInfo权限。

   ```
   Usage:
   ossutil cp -r <local dir of fastq> <path of  oss bucket >
   e. g.
   ossutil cp -r ./MGISEQ oss://my-test-shenzhen/MGISEQ
   ```

5. 通过ossutil上传FASTQ或者GVCF数据到OSS Bucket。

   关于ossutil的下载和安装，请参见下载和安装ossutil。

   > **说明** 目前只支持人类基因数据WGS、WES等，暂不支持甲基化数据，以及动植物数据的比对。

```
Usage:
ossutil cp -r <local dir of fastq> <path of  oss bucket >
e. g.
ossutil cp -r ./MGISEQ oss://my-test-shenzhen/MGISEQ
```

启动群体联合变异检测

1. 生成GVCF文件。

   关于参考基因组--reference，推荐和默认使用 `hg19` 基因组（hs37d5版本）。并且各个样本 `--reference-group` 中的SampleName（SM）须为不同，如以下所示，*MGISEQ_NA12878_hs37d5_6.gvcf*采用的SM为 *12878*，*MGISEQ_NA12878_hs37d5_5.gvcf*采用的SM为 *12878_1*。

   > **说明** 如果您已经有生成的GVCF文件，可以跳过该步骤，直接进行群体变异合并检测。

```
Usage:
ags remote run wgs \
--region cn-shenzhen # region of oss, e.g. cn-shenzhen, cn-beijing and etc\
--fastq1 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz # filename of fastq pair 2, fastq-path\filename \
--fastq2 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz  # filename of fastq pair 1\
--bucket my-test-shenzhen # Bucket name\
--output-bam bam/MGISEQ_NA12878_hs37d5.bam, # Output BAM to bucket,  By default empty, non output of BAM \
--output-gvcf gvcf/MGISEQ_NA12878_hs37d5_5.gvcf # Output filename \
--service "s" #SLA: [n:normal|s:silver|g:gold|p:platinum]\
--reference [hg19|hg38|<reference path on OSS>] # hg19: it is hs37d5 version, GRCh37/hg19 include decoy contig, no support for UCSC hg19. hg38: GRCh38/hg38 include decoy
--reference-group "\"@RG\\tID:TEST\\tSM:12878\\tPL:MGISEQ2000\"" # allow to specify reference groups for PL/SM/ID and etc

e.g.

ags remote run wgs \
--region cn-shenzhen \
--fastq1 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz  \
--fastq2 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz  \
--bucket my-test-shenzhen \
--output-gvcf gvcf/MGISEQ_NA12878_hs37d5_5.gvcf \
--output-bam bam/MGISEQ_NA12878_hs37d5_5.bam \
--service "s" \
--reference hg19 \
--reference-group "\"@RG\\tID:TEST\\tSM:12878_1\\tPL:MGISEQ2000\""

ags remote run wgs \
--region cn-shenzhen \
--fastq1 MGISAMPLE001 \
--fastq2 MGISAMPLE001  \
--bucket my-test-shenzhen \
--output-gvcf gvcf/MGISEQ_NA12878_hs37d5_6.gvcf \
--output-bam bam/MGISEQ_NA12878_hs37d5_6.bam \
--service "s" \
--reference hg19 \
--reference-group "\"@RG\\tID:TEST\\tSM:12878\\tPL:MGISEQ2000\""
```

2. 启动群体变异合并检测。

   通过merge调用生产多样本GVCF的合并GVCF *cohort.gvcf.gz*，相当于*GATK CombineGVCFs*结果。而联合基因型分析结果*output.vcf.gz*，相当于*GATK GenotypeGVCFs*的结果。

```
Usage:
ags remote merge \
--region cn-shenzhen # region of oss, e.g. cn-shenzhen, cn-beijing and etc\
--bucket my-test-shenzhen # Bucket name\
--gvcf gvcf/MGISEQ_NA12878_hs37d5_5.gvcf # filename of gvcf 1\
--gvcf gvcf/MGISEQ_NA12878_hs37d5_6.gvcf # filename of gvcf 2\
--gvcf gvcf/MGISEQ_NA12878_hs37d5_7.gvcf # filename of gvcf 3\
--output-vcf merge/output.vcf.gz # Output filename of vcf, it is simiar to result of GATK GenotypeGVCFs
\
--output-gvcf merge/cohort.gvcf.gz # Output filename of gvcf, it is similar to result of GATK CombineGV
CFs\
--service "s" #SLA: [s:silver|g:gold|p:platinum]\

e.g.
ags remote merge \
--region cn-shenzhen \
--bucket my-test-shenzhen \
--gvcf gvcf/MGISEQ_NA12878_hs37d5_5.gvcf \
--gvcf gvcf/MGISEQ_NA12878_hs37d5_6.gvcf \
--output-vcf merge/output.vcf.gz \
--output-gvcf merge/output.gvcf.gz \
--service "s"
INFO[0001] {"JobName":"merge-hck8x"}
INFO[0001] Job submit succeed
```

3. 列出远程任务。

```
Usage:
ags remote list
e.g.
ags remtoe list
```

```
+---------------+------------------------------+------------+
|  JOB NAME   |       CREATE TIME        | JOB STATUS |
+---------------+------------------------------+------------+
| merge-hck8x  | 2020-08-31 20:57:06 +0000 UTC | Running   |
| merge-5c4lt  | 2020-08-31 18:01:38 +0000 UTC | Succeeded  |
+---------------+------------------------------+------------+
```

2.1. 通过AGS处理全基因组测序WGS

通过AGS可以快速处理全基因组测序WGS（Whole Genome Sequencing）的全流程任务，包括基因比对、排序、去重和变异检测。本文介绍如何通过AGS命令行管理WGS工作流。

前提条件

已提交阿里云基因服务公测申请，加入公测白名单。

> **说明** 如果您使用的是子账号，公测申请时请提供子账号的UID。请登录账号管理控制台查询UID。

准备工作

完成权限的配置和数据的准备。

1. 配置AGS。

   关于AGS的下载和安装，请参见AGS命令行帮助。

   ```
   ags config init
   ```

2. 准备Bucket，并授权AGS服务读写权限和GetBucketInfo权限。

   > **说明**
   > - 请确保指定Bucket的Owner是您当前的账号，否则建议您新建一个Bucket后再进行GetBucketInfo的授权。
   > - 如果您使用的是子账号，需要为子账号授予AliyunOSSFullAccess权限，请参见为RAM用户授权。
   > - 如果您使用的是子账号，建议您重新创建OSS Bucket，以确保该账号是所使用Bucket的Owner。执行以下命令确认Bucket的Owner。
   >
   >   ```
   >   ossutil stat oss://<your new bucket name>
   >   ```

   ```
   Usage:
   ags config oss <your bucket name>

   e.g.
   ags config oss my-test-shenzhen
   ```

3. 通过ossutil上传FASTQ数据到OSS Bucket。

   关于ossutil的下载和安装，请参见下载和安装。

   > **说明** 目前只支持人类基因数据 WGS、WES等，暂不支持甲基化数据，以及动植物数据的比对。

```
Usage:

ossutil cp -r <local dir of fastq> <path of  oss bucket >

e. g.
ossutil cp -r ./MGISEQ oss://my-test-shenzhen/MGISEQ
```

启动WGS流程

关于参考基因组--reference，推荐和默认使用 `hg19` 基因组（hs37d5版本）。

> **说明** `hg19` 基因组（hs37d5版本）主要有以下特点：
>
> - 不包含ALT contigs
> - Hard mask了chrY上的PARs区域
> - 包含decoy contig
>
> 虽然AGS是ALT-Aware，可以识别并处理ALT contigs，而 `UCSC hg19` 基因组包含了ALT contigs，但是由于 `UCSC hg19` 基因组不具备后面两个特点，仍会造成变异检测的质量下降。详情参见此处。

```
Usage:

ags remote run wgs \
--region cn-shenzhen # region of oss, e.g. cn-shenzhen, cn-beijing and etc\
--fastq1 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz # filename of fastq pair 2, fast
q-path\filename \
--fastq2 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz  # filename of fastq pair 1\
--bucket my-test-shenzhen # Bucket name\
--output-bam bam/MGISEQ_NA12878_hs37d5.bam, # Output BAM to bucket,  By default empty, non output
of BAM \
--output-vcf vcf/MGISEQ_NA12878_hs37d5_5.vcf # Output filename \
--service "g" #SLA: [n:normal|s:silver|g:gold|p:platinum]\
--reference [hg19|hg38|<reference path on OSS>] # hg19: it is hs37d5 version, GRCh37/hg19 include decoy co
ntig, no support for UCSC hg19. hg38: GRCh38/hg38 include decoy
--reference-group "\"@RG\\tID:TEST\\tSM:12878\\tPL:MGISEQ2000\"" # allow to specify reference groups for
PL/SM/ID and etc
e.g.
ags remote run wgs \
--region cn-shenzhen \
--fastq1 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz \
--fastq2 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz \
--bucket my-test-shenzhen \
```

```
--output-vcf vcf/MGISEQ_NA12878_hs37d5_5.vcf \
--output-bam bam/MGISEQ_NA12878_hs37d5_5.bam \
--service "s" \
--reference hg19

### 批量多Lane多样本的下机样本的处理
MGISAMPLE001是一组WGS多Lane测序样本，可以通过指定样本目录--fastq1 MGISAMPLE001，--fastq2 MGISAMP
LE001 实现对多Lane测序结果的合并和计算。
oss://my-test-shenzhen/MGISAMPLE001/L1/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz
oss://my-test-shenzhen/MGISAMPLE001/L2/MGISEQ2000_PCR-free_NA12878_1_V100003043_L02_1.fq.gz
oss://my-test-shenzhen/MGISAMPLE001/L1/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz
oss://my-test-shenzhen/MGISAMPLE001/L2/MGISEQ2000_PCR-free_NA12878_1_V100003043_L02_2.fq.gz

ags remote run wgs \
--region cn-shenzhen \
--fastq1 MGISAMPLE001 \
--fastq2 MGISAMPLE001  \
--bucket my-test-shenzhen \
--output-vcf vcf/MGISEQ_NA12878_hs37d5_6.vcf \
--output-bam bam/MGISEQ_NA12878_hs37d5_6.bam \
--service "g" \
--reference hg19

ags remote run wgs \
--region cn-shenzhen \
--fastq1 MGISAMPLE002 \
--fastq2 MGISAMPLE002  \
--bucket my-test-shenzhen \
--output-vcf vcf/MGISEQ_NA12878_hs37d5_7.vcf \
--output-bam bam/MGISEQ_NA12878_hs37d5_7.bam \
--service "g" \
--reference hg19
```

单击调用AGS WGS，将为您演示如何通过命令调用AGS WGS。

启动Mapping流程

通过--fastq1和--fastq2指定*fastq*，通过--output指定*bam*的输出路径。

```
Usage:

ags remote run mapping \
--region cn-shenzhen # region of oss, e.g. cn-shenzhen, cn-beijing and etc\
--fastq1 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz # filename of fastq pair 2, fast
q-path\filename \
--fastq2 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz  # filename of fastq pair 1\
--bucket my-test-shenzhen # Bucket name\
--output-bam bam/MGISEQ_NA12878_hs37d5.bam # Output filename of BAM \
--service "g" #SLA: [n:normal|s:silver|g:gold|p:platinum]\
--markdup [true|false|default true] #Mark Duplicated, by default true
--reference [hg19|hg38|<reference path on OSS>]

e.g.

ags remote run mapping \
--region cn-shenzhen \
--fastq1 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz  \
--fastq2 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz  \
--bucket my-test-shenzhen \
--output-bam bam/MGISEQ_NA12878_hs37d5.bam # Output filename of BAM \
--service "g" \
--markdup "true" \
--reference hg19
```

单击调用AGS Mapping，将为您演示如何通过命令调用AGS Mapping。

列出远程流程

```
Usage:
ags remote list

e.g.
ags remtoe list
+---------------+------------------------------+
|  JOB NAME   |       CREATE TIME       |
+---------------+------------------------------+
| wgs-gpu-ckw96 | 2020-01-07 19:08:32 +0000 UTC |
| wgs-gpu-djzws | 2020-01-07 18:31:22 +0000 UTC |
| wgs-gpu-pd659 | 2020-01-03 20:34:09 +0000 UTC |
+---------------+------------------------------+
```

获取流程的详细信息

```
Usage:
ags remote get <workflow id> --show
--show show detail of input parameters of workflow

e.g.
ags remote get wgs-gpu-sjtlw
+---------------+-----------------+-----------+----------------------------+----------+----------------------------+
|  JOB NAME   | JOB NAMESPACE  | STATUS  |      CREATE TIME       | DURATION |      FINISH TIME       |
+---------------+-----------------+-----------+----------------------------+----------+----------------------------+
| wgs-gpu-sjtlw | XXXXXXXXXXXXXXXX | Succeeded | 2020-01-07 21:38:05 +0800 CST | 12m25s   | 2020-01-07 21:50
:30 +0800 CST |
+---------------+-----------------+-----------+----------------------------+----------+----------------------------+

ags remote get wgs-gpu-97xfn --show

+---------------+-----------------+-----------+----------------------------+----------+----------------------------+
|  JOB NAME   | JOB NAMESPACE  | STATUS  |      CREATE TIME       | DURATION |      FINISH TIME       |
+---------------+-----------------+-----------+----------------------------+----------+----------------------------+
| wgs-gpu-sjtlw | XXXXXXXXXXXXXXXX | Succeeded | 2020-01-07 21:38:05 +0800 CST | 12m25s   | 2020-01-07 21:50
:30 +0800 CST |
+---------------+-----------------+-----------+----------------------------+----------+----------------------------+


+----------------------+-------------------------------+
|    JOB DETAIL     |                   |
+----------------------+-------------------------------+
| wgs_reference_file   | hg19                |
| wgs_service       | g                   |
| wgs_oss_region     | cn-shenzhen             |
| wgs_fastq_first_name | MGISAMPLE001            |
| wgs_fastq_second_name | MGISAMPLE001            |
| wgs_bucket_name     | my-test-shenzhen          |
| wgs_vcf_file_name   | vcf/MGISEQ_NA12878_hs37d5_6.vcf |
| wgs_bam_file_name   | bam/MGISEQ_NA12878_hs37d5_6.bam |
+----------------------+-------------------------------+
```

取消运行中的工作流

```
Usage:

ags remote cancel  <workflow id>

e.g.

ags remote cancel wgs-gpu-zls6r
INFO[0000] Successed to cancel wgs-gpu-zls6r
```

删除结束的工作流

您可以删除成功和失败的工作流，但不能删除运行中的工作流。

```
Usage:

ags remote remove <workflow id>

e.g.

ags remote remove wgs-gpu-zls6r
INFO[0000] Successed to remove wgs-gpu-zls6r
```

2.2. 通过AGS排查病毒序列

在病原体鉴定过程中，通过对比待测样本与已知病原体基因组数据库，可以高效准确的对待测样本进行鉴定分析。AGS服务提供了针对宏基因组测序数据的快速比对能力。本文介绍如何通过AGS排查待测病原体。

前提条件

已提交阿里云基因服务公测申请，加入公测白名单。

> **说明** 如果您使用的是子账号，公测申请时请提供子账号的UID。请登录账号管理控制台查询UID。

背景信息

新冠肺炎发病后数周内都可以在上呼吸道或下呼吸道中检测到新型冠状病毒SARS-CoV-2 RNA（以下简称“新冠病毒”）。目前，已经有众多新冠病毒RT-PCR试剂盒可供选择，虽然PCR操作成本低、实效性高。但由于受病毒浓度和试剂盒质量等因素的影响，容易出现假阴性，需要多次检测确认，而且一次检测只能对一种特定病毒进行排查。

泛病原体检测的基因组下一代测序（Metagenomics Next-generation Sequencing, mNGS）技术直接从临床样本中随机抽取一定比例的核酸片段（包括大量人源核酸和少量微生物核酸）进行测序、数据库比对和生物信息分析，进而对病原微生物进行无偏性鉴定。该技术为新冠病毒SARS-CoV-2 RNA的早期发现和准确测序做出很大贡献。

相较于RT-PCR试剂盒检测，虽然mNGS方式检测分析周期相对较长，但准确率高，检测全面，可以一次排查多种病毒，同时也可监测病毒在传播过程中可能发生的变异，增强对病毒的防控。临床上，针对荧光定量PCR无法确诊的疑似患者进行二次检测可进一步提升检测结果的准确性，有效防止病毒变异产生的漏检。基于核酸序列比对的分析方式，一旦病原体的基因组已知，通过更新数据库，就可以实现高效准确检测病原体。

阿里云基因计算服务AGS提供了针对mNGS宏基因组测序数据的快速比对能力，对一组肺泡采样测序的宏基因组数据3.2Gbase（22M reads），60秒内可以完成和已知的病原体基因组包括新型冠状病毒SARS-CoV-2，39种BetaCov RNA，以及9334种已知病毒的参考序列的比对，并且支持自定义的病毒库的上传和比对。对于疾控中心，医院，实验室只需要一个阿里云的对象存储Bucket，以及命令行ags就可以完成整个的比对过程，并拿到高质量的匹配reads的数据和初步质量报告，为多种病原体检测，进一步的新冠病毒的蛋白质研究和变异研究提供了快捷准确的数据支撑。

欢迎基因测序厂商，疾控中心，医院，学校，制药企业申请使用。

准备工作

1. 配置AGS。AGS的下载和安装，请参见AGS命令行帮助。命令示例：

   ```
   ags config init
   ```

2. 下载安装ossutil命令行，详情请参见下载和安装。
3. 注册阿里云账号，开通对象存储OSS服务并创建存储空间（Bucket）用于存放mNGS测序数据（例如：oss://my-test-shenzhen）。详情请参见开始使用阿里云OSS。

   > **说明** 如果您使用的是子账号，建议您重新创建OSS Bucket，以确保该账号是所使用Bucket的Owner。执行以下命令确认Bucket的Owner。
   >
   > ```
   > ossutil stat oss://<your new bucket name>
   > ```

4. 为服务AGS服务授予Bucket的GetBucketInfo权限。

   > **说明**
   >
   > - 请确保指定Bucket的Owner是您当前的账号，否则建议您新建一个Bucket后再进行GetBucketInfo的授权。
   > - 如果您使用的是子账号，需要为子账号授予AliyunOSSFullAccess权限，请参见为RAM用户授权。

   命令示例：

   ```
   ags config oss <bucket_name>
   ```

rna-mapping使用示例

单击检测AGS rna-mapping病毒，将为您演示如何通过命令检测AGS rna-mapping病毒。

与新冠病毒的比对

1. 运行ossutil命令，将mNGS测序数据上传至存储空间（Bucket）。命令示例：

```
ossutil cp  ICU6G_S2_L001_R1_001.fastq.gz oss://my-test-shenzhen/cov2-samples/
ossutil cp  ICU6G_S2_L001_R2_001.fastq.gz oss://my-test-shenzhen/cov2-samples/
```

2. 运行比对任务，对比mNGS数据和已知RNA序列数据。

   本文以比对ICU6G_S2_L001测序样本和新冠病毒的相似度为例，运行任务如下。

   命令格式：

```
Usage:
ags remote run rna-mapping \ # <rna-mapping>: RNA 序列的比对任务
--region <region_id> \ #
<cn-shenzhen|cn-beijing|...>: 地域ID，目前支持深圳和北京。
--bucket <bucket_name> \ # <bucket_name> 对象存储bucket的名称
--fastq1 <path_fq1> \ # 双端测序数据fq1相对路径
--fastq2 <path_fq2>\ # 双端测序数据fq2的相对路径
--output-bam <path_of_output_bam> \ #产出比对结果bam的输出路径，报告也在同样位置，以.txt结尾
--reference [sars-cov-2 | betacov-ncbi-39 | viral-9334 | <path of RNA library reference in specified bucket
>] # 参考序列预置了新型冠状病毒sars-cov-2和目前已经知道的39种betacov的冠状病毒，可以指定自定义的病毒
序列库
```

   命令示例：

```
ags remote run rna-mapping \
--region cn-shenzhen \
--fastq1 cov2-samples/ICU6G_S2_L001_R1_001.fastq.gz \
--fastq2 cov2-samples/ICU6G_S2_L001_R2_001.fastq.gz  \
--bucket my-test-shenzhen \
--output-bam bam/ICU6G_S2.bam  \
--reference sars-cov-2



INFO[0002] {"JobName":"rna-mapping-gpu-2ms6w"}
INFO[0002] Job submit succeed
```

3. 检查比对任务和比对结果。

   在这个比对任务示例中，10M reads和新冠病毒序列MN908947.3比对产生了3629个高质量重合的reads，并在新冠病毒特征区间排列分（AS）超过120分的序列（reads）条数有404个。说明可以精确的从此样本的测序数据中检测出SARS-CoV-2 RNA的序列。

   比对结果示例：

```
ags remote get rna-mapping-gpu-2ms6w --show
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+
|    JOB NAME     | JOB NAMESPACE  | STATUS  |       CREATE TIME       | DURATION |       FINISH TIME
| TOTAL READS | TOTAL BASES |
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+
| rna-mapping-gpu-2ms6w | XXXXXXXXXXXX | Succeeded | 2020-03-04 16:40:30 +0800 CST | 43s     | 2020-03-
04 16:41:13 +0800 CST |   10369818 |  1456539874 |
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+



+--------------------------------+-------------------------------------------+
|       JOB DETAIL       |                           |
+--------------------------------+-------------------------------------------+
| rna_matached_reads         |                      480 |
| rna_is_sars_cov2          | True                      |
| rna_mapping_oss_region       | cn-shenzhen                 |
| rna_mapping_fastq_second_name  | cov2-samples/ICU6G_S2_L001_R2_001.fastq.gz |
| rna_mapping_no_unmapped      |                           |
| rna_mapping_service        | s                         |
| rna_matached_reads_alignment   |                      404 |
| rna_high_quality_mapped      |                     3629 |
| rna_mapping_fastq_first_name   | cov2-samples/ICU6G_S2_L001_R1_001.fastq.gz |
| rna_mapping_mark_dup        |                           |
| rna_mapping_reference_file_name | sars-cov-2                |
| rna_cov_detail_file         | bam/ICU6G_S2.bam.cov.txt          |
| rna_mapping_bam_file_name     | bam/ICU6G_S2.bam              |
| rna_mapping_bucket_name      | my-test-shenzhen              |
+--------------------------------+-------------------------------------------+
```

4. 执行ossutil命令，下载对比数据和简单报告。

   命令示例：

```
ossutil ls oss://my-test-shenzhen/bam/ICU6G_S2.bam
LastModifiedTime            Size(B)  StorageClass   ETAG                  ObjectName
2020-03-04 16:41:11 +0800 CST    356320    Standard   9596D012A30438A0073A2A0B38F5D578    oss://m
y-test-shenzhen/bam/ICU6G_S2.bam
2020-03-04 16:41:11 +0800 CST     2889    Standard   63175E7180D110BA9D3BAB34F4313C59    oss://my
-test-shenzhen/bam/ICU6G_S2.bam.cov.txt
2020-03-04 16:41:11 +0800 CST     396    Standard   940D51FF7ECFF60B5E5A41D1F635180D    oss://my-
test-shenzhen/bam/ICU6G_S2.bam.summary.json

ossutil cp oss://my-test-shenzhen/bam/HKU2_160660.summary.json .
ossutil cp -r oss://my-test-shenzhen/bam/ICU6G_S2.bam.cov.txt .
ossutil cp oss://my-test-shenzhen/bam/HKU2_160660.bam .
```

报告示例：

```
cat bam/ICU6G_S2.bam.cov.txt

Summary:
High Quality Mapped Reads is: 3629
Matched reads in orf1ab range is: 480
Matched reads in orf1ab range with alignment score (AS) is greater than 120: 404
/data/cov2-samples_ICU6G_S2_L001_R1_001.fastq.gz-output/ICU6G_S2.bam is similar to SARS-CoV-2 w
ith very high mappQ and AS reads: True
     21571    21581    21591    21601    21611    21621    21631
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNN
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCC
CTGC
ATGTT GTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCC
TGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGT                  CAATTACCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGA      CCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCC
CTGC
ATGTTTGTTTTTCTTGTTTTATTGCCA  agtctctagtcagtgtgttaatcttacaaccagaactcaattaccccctgc
atgtttgtttttcttgttttattgcca      AGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCC
CTGC
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattaccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattaccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattaccccctgc
```

```
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCC
CTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCC
CTGC
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCC
CTGC
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCC
CTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCC
CTGC
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
 TGTTTGTTTTTCTTGTTTT   CACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
 tgtttgtttttcttgtttt
  TTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCT
GC
   gtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
```

5. 进一步分析比对数据。您可以通过samtools stats， plot-bamstat 等工具将比对数据进一步分析coverage, depth等的相似度，还可以进一步实现蛋白质组成分析，以及变异分析。

6. 重复上述步骤可实现不同样本之间的比对。

与已知的39个betaCov病毒的比对

1. 运行ossutil命令，将mNGS测序数据上传至存储空间（Bucket）。命令示例：

```
ossutil cp  ICU6G_S2_L001_R1_001.fastq.gz oss://my-test-shenzhen/cov2-samples/
ossutil cp  ICU6G_S2_L001_R2_001.fastq.gz oss://my-test-shenzhen/cov2-samples/
```

2. 运行对比任务。

   命令示例：

```
ags remote run rna-mapping \
--region cn-shenzhen \
--fastq1 cov2-samples/ICU6G_S2_L001_R1_001.fastq.gz \
--fastq2 cov2-samples/ICU6G_S2_L001_R2_001.fastq.gz  \
--bucket my-test-shenzhen \
--output-bam bam/ICU6G_S2_virus.bam  \
--reference betacov-ncbi-39

INFO[0011] {"JobName":"rna-mapping-gpu-6mpcc"}
INFO[0011] Job submit succeed
```

3. 检查比对任务和比对结果。

   比对结果示例：

```
ags remote get rna-mapping-gpu-6mpcc --show
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+
|    JOB NAME     | JOB NAMESPACE  | STATUS  |      CREATE TIME       | DURATION |       FINISH TIME
| TOTAL READS | TOTAL BASES |
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+
| rna-mapping-gpu-6mpcc | XXXXXXXXX | Succeeded | 2020-03-04 17:36:21 +0800 CST | 40s     | 2020-03-04 1
7:37:01 +0800 CST |   10369818 |  1456539874 |
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+
# 2014 mapped reads detected, but no mapped reads found in range
+---------------------------------+-------------------------------------------+
|       JOB DETAIL        |                           |
+---------------------------------+-------------------------------------------+
| rna_mapping_reference_file_name | betacov-ncbi-39                 |
| rna_matached_reads_alignment   |                          0 |
| rna_mapping_bam_file_name     | bam/ICU6G_S2_virus.bam              |
| rna_mapping_fastq_first_name   | cov2-samples/ICU6G_S2_L001_R1_001.fastq.gz |
| rna_mapping_oss_region       | cn-shenzhen                    |
| rna_cov_detail_file         | bam/ICU6G_S2_virus.bam.cov.txt         |
| rna_mapping_no_unmapped      |                           |
| rna_matached_reads          |                        0 |
| rna_mapping_mark_dup        |                           |
| rna_mapping_service         | s                         |
| rna_high_quality_mapped      |                         2014 |
| rna_mapping_bucket_name      | my-test-shenzhen                |
| rna_mapping_fastq_second_name  | cov2-samples/ICU6G_S2_L001_R2_001.fastq.gz |
| rna_is_sars_cov2           | False                         |
+---------------------------------+-------------------------------------------+
```

与自定义病毒库的比对

1. 从NCBI GeneBank下载reference序列合并成为一个多contig的参考序列。

   示例：

   搜索核酸包含“betacov”的所有参考系列， 并下载参考系列。

2. 把下载的序列文件 `sequence.fa` 改名为 `betacov-ncbi-test.fa` 。
3. 上传reference到存储空间。命令示例：

```
ossutil cp betacov-ncbi-test.fa oss://my-test-shenzhen/ref/
```

4. 提交比对任务，指定reference的路径。命令示例：

```
ags remote run rna-mapping \
--region cn-shenzhen \
--fastq1 cov2-samples/ICU6G_S2_L001_R1_001.fastq.gz \
--fastq2 cov2-samples/ICU6G_S2_L001_R2_001.fastq.gz \
--bucket my-test-shenzhen \
--output-bam bam/ICU6G_S2_virus.bam \
--reference ref/betacov-ncbi-test.fa

INFO[0002] {"JobName":"rna-mapping-gpu-69mwb"}
INFO[0002] Job submit succeed
```

5. 查看比对报告和获取匹配的比对数据。

   对比结果示例：

```
ags remote get rna-mapping-gpu-69mwb --show
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+
|    JOB NAME     | JOB NAMESPACE  | STATUS  |       CREATE TIME       | DURATION |       FINISH TIME
| TOTAL READS | TOTAL BASES |
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+
| rna-mapping-gpu-69mwb | 1365606736606053 | Succeeded | 2020-03-04 17:47:00 +0800 CST | 40s     | 202
0-03-04 17:47:40 +0800 CST |   10369818 |  1456539874 |
+----------------------+-----------------+-----------+-----------------------------+----------+-----------------------------+-----
--------+-------------+



+---------------------------------+-------------------------------------------+
|       JOB DETAIL        |                           |
+---------------------------------+-------------------------------------------+
| rna_mapping_fastq_first_name   | cov2-samples/ICU6G_S2_L001_R1_001.fastq.gz |
| rna_mapping_fastq_second_name  | cov2-samples/ICU6G_S2_L001_R2_001.fastq.gz |
| rna_mapping_mark_dup        |                           |
| rna_mapping_oss_region       | cn-shenzhen                   |
| rna_cov_detail_file         | bam/ICU6G_S2_virus.bam.cov.txt        |
| rna_is_sars_cov2          | False                         |
| rna_mapping_bam_file_name     | bam/ICU6G_S2_virus.bam             |
| rna_mapping_service         | s                           |
| rna_matached_reads_alignment   |                          0 |
| rna_high_quality_mapped      |                       2014 |
| rna_mapping_bucket_name      | my-test-shenzhen                |
| rna_mapping_no_unmapped      |                           |
| rna_mapping_reference_file_name | ref/betacov-ncbi-test.fa            |
| rna_matached_reads          |                          0 |
+---------------------------------+-------------------------------------------+
+---------------------------------+-------------------------------------------+
```

6. 下载比对数据做进一步分析。命令示例：

```
ossutil ls oss://my-test-shenzhen/bam/ICU6G_S2_virus.bam
LastModifiedTime           Size(B)  StorageClass  ETAG                    ObjectName
2020-03-04 17:47:38 +0800 CST    753458    Standard  DF7B1A6CA5AF5DE6BF4FFDBB6DEF71C3    oss://
my-test-shenzhen/bam/ICU6G_S2_virus.bam
2020-03-04 17:47:38 +0800 CST      1474    Standard  9D7968A779A0DE7C1993CC2A8D0E5A56    oss://my
-test-shenzhen/bam/ICU6G_S2_virus.bam.cov.txt
2020-03-04 17:47:38 +0800 CST       397    Standard  81170E30BAAFEB947A2238E015171A51    oss://my-t
est-shenzhen/bam/ICU6G_S2_virus.bam.summary.json
Object Number is: 3

ossutil cp oss://my-test-shenzhen/bam/ICU6G_S2_virus.bam.summary.json .

cat bam/ICU6G_S2_virus.bam.summary.json
{
  "total_reads":10369818,
  "total_bases":1456539874,
  "pass_vendor_filter_reads":10369818,
  "mapped_reads":6736,
  "pair_reads":6680,
  "properly_paired_reads":6520,
  "mapq_40_to_inf_reads":2030,
  "mapq_30_to_40_reads":0,
  "mapq_20_to_30_reads":1,
  "mapq_10_to_20_reads":3,
  "mapq_0_to_10_reads":23,
  "mapq_0_reads":10367761,
  "GC":"46.499%",
  "total_alignment":2057,
  "supplementary_alignment":0
}%

ossutil cp oss://my-test-shenzhen/bam/ICU6G_S2_virus.bam .
samtools view bam/ICU6G_S2_virus.bam
```

2.3. 通过AGS分析肿瘤样本

通过AGS调用 `mutect2` 任务来检测体细胞短突变，短突变包括单核苷酸（SNV）以及插入和缺失（InDel）的突变。本文介绍如何通过AGS分析肿瘤样本。

背景信息

AGS `mutect2` 支持两种模式的典型场景：

- 肿瘤加正常样本模式：肿瘤样本在分析过程跳过正常人的胚系变异。
- 肿瘤模式：对单个肿瘤样本的比对数据进行分析。

`mutect2` 的体系变异检测是保持了和GATK4.1.3一致的变异检测方式，但提供了30~80倍的加速。针对90Gbase的比对数据，10分钟内可以完成变异检测。

mutect2使用示例

单击检测AGS Mutect2肿瘤，将为您演示如何通过命令检测AGS Mutect2肿瘤。

在肿瘤加正常样本模式下分析样本

以给定匹配的正常样本作为基准，`mutect2` 仅检测体细胞变异。`mutect2` 会根据提供的证据（例如在匹配的正常人中），实现跳过在胚系中明显存在的变异的逻辑，以避免在胚系事件上花费计算资源。

- 用法

  执行命令：

  ```
  ags remote run mutect2 \
  ```

  返回结果：

  ```
  --region cn-shenzhen # region of oss, e.g. cn-shenzhen, cn-beijing and etc\
  --bucket my-test-shenzhen # Bucket name\
  --input-bam-tumor bam/HKU2_160660.bam #Tumor sample bam file\
  --input-bam-normal  bam/MGISEQ_NA12878_RG_HG38.bam  # Optional normal sample bam \
  --bed bed/performance.blocks.exp.bed # Optional target bed \
  --output-vcf vcf/HKU2_160660.vcf  # Output filename\
  --service "s" #SLA: [n:normal|s:silver|g:gold|p:platinum]\
  --reference [hg19|hg38|<reference path on OSS>] # hg19: it is hs37d5 version, GRCh37/hg19 include decoy contig, no support for UCSC hg19. hg38: GRCh38/hg38 include decoy
  ```

- 结果示例：

```
--region cn-shenzhen \
--bucket my-test-shenzhen \
--input-bam-tumor bam/HKU2_160660.bam \
--input-bam-normal  bam/MGISEQ_NA12878_RG.bam \
--output-vcf vcf/HKU2_160660.vcf \
--service "s"  \
--reference hg19
INFO[0001] {"JobName":"mutect2-gpu-vp7d9"}
INFO[0001] Job submit succeed

ags remote get mutect2-gpu-vp7d9 --show
+------------------+------------------+---------+-----------------------------+---------------+-------------+-------------+
|   JOB NAME   | JOB NAMESPACE  | STATUS |      CREATE TIME      |  DURATION   | TOTAL READS | TOTAL BASES |
+------------------+------------------+---------+-----------------------------+---------------+-------------+-------------+
| mutect2-gpu-vp7d9 | XXXXXXXXX | Running | 2020-04-10 16:02:39 +0800 CST | 36.311883677s |       0 |       0 |
+------------------+------------------+---------+-----------------------------+---------------+-------------+-------------+



+-------------------------+--------------------------+
|     JOB DETAIL     |                |
+-------------------------+--------------------------+
| mutect2_reference_group |                |
| mutect2_oss_region     | cn-shenzhen          |
| mutect2_bucket_name    | my-test-shenzhen      |
| mutect2_output_vcf_name | vcf/HKU2_160660.vcf    |
| mutect2_reference_file  | hg19              |
| mutect2_input_bam_tumor | bam/HKU2_160660.bam    |
| mutect2_input_bam_normal | bam/MGISEQ_NA12878_RG.bam |
| mutect2_input_bed     |                |
| mutect2_service       | s              |
+-------------------------+--------------------------+
```

在单独肿瘤样本模式下分析样本

此模式对单一类型的样本（例如肿瘤或正常样本）进行分析。

- 用法

  执行命令：

```
ags remote run mutect2 \
```

返回结果：

```
--region cn-shenzhen # region of oss, e.g. cn-shenzhen, cn-beijing and etc\
--bucket my-test-shenzhen # Bucket name\
--input-bam-tumor bam/HKU2_160660.bam #Tumor／Normal sample bam file\
--output-vcf vcf/HKU2_160660.vcf  # Output filename\
--service "s" #SLA: [n:normal|s:silver|g:gold|p:platinum]\
--reference [hg19|hg38|<reference path on OSS>] # hg19: it is hs37d5 version, GRCh37/hg19 include decoy contig, no support for UCSC hg19. hg38: GRCh38/hg38 include decoy
```

- 结果示例：

```
--region cn-shenzhen \
--bucket my-test-shenzhen \
--input-bam-tumor bam/HKU2_160660.bam \
--output-vcf vcf/HKU2_160660.all.vcf \
--service "s" \
--reference hg19
INFO[0001] {"JobName":"mutect2-gpu-6tc8s"}
INFO[0001] Job submit succeed

ags remote get mutect2-gpu-6tc8s --show
+------------------+-----------------+-----------+----------------------------+----------+----------------------------+-----------
-+-------------+
|    JOB NAME    | JOB NAMESPACE  | STATUS  |       CREATE TIME       | DURATION |       FINISH TIME       | T
OTAL READS | TOTAL BASES |
+------------------+-----------------+-----------+----------------------------+----------+----------------------------+-----------
-+-------------+
| mutect2-gpu-6tc8s | XXXXXXXXXX | Succeeded | 2020-04-10 15:51:59 +0800 CST | 4m12s    | 2020-04-10 15:56:
11 +0800 CST |        0 |        0 |
+------------------+-----------------+-----------+----------------------------+----------+----------------------------+-----------
-+-------------+



+-------------------------+-------------------------+
|      JOB DETAIL      |                         |
+-------------------------+-------------------------+
| mutect2_oss_region     | cn-shenzhen          |
| mutect2_input_bam_tumor  | bam/HKU2_160660.bam    |
| mutect2_input_bam_normal |                      |
| mutect2_input_bed      |                      |
| mutect2_output_vcf_name  | vcf/HKU2_160660.all.vcf |
| mutect2_bucket_name     | my-test-shenzhen      |
| mutect2_reference_file  | hg19                 |
| mutect2_reference_group |                      |
| mutect2_service       | s                    |
+-------------------------+-------------------------+
```

3.AGS无服务器化API参考

3.1. 通过AGS查询所有工作流

调用DescribeWorkflows查询已创建的所有工作流。

请求信息

请求行RequestLine

```
GET /gs/workflows HTTP/1.1
```

特有请求头RequestHead

无，请参见公共请求和返回结果。

请求体RequestBody

无

返回信息

返回行ResponseLine

```
HTTP/1.1 200 OK
```

特有返回头ResponseHead

无，请参见公共请求和返回结果。

返回体ResponseBody

```
{
 "jobs": [
  {
   "create_time": "2020-01-15T14:13:16Z",
   "cluster_id": "cb1a7214cfc0b41d9bb086affc2d8f51c",
   "job_name": "mapping-gpu-mhhgh"
  },
  {
   "create_time": "2020-01-15T13:19:26Z",
   "cluster_id": "cb1a7214cfc0b41d9bb086affc2d8f51c",
   "job_name": "mapping-gpu-98wt4"
  },
  {
   "create_time": "2020-01-15T13:18:52Z",
   "cluster_id": "cb1a7214cfc0b41d9bb086affc2d8f51c",
   "job_name": "wgs-gpu-qb4dk"
  }
 ]
}
```


返回体解释

名称	类型	描述
cluster_id	String	集群ID。
job_name	String	工作流名称。
create_time	String	工作流创建时间。

示例

请求示例（Python）

```
#!/usr/bin/env python
#coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
import os
client = AcsClient(os.environ['accessKeyID'], os.environ['accessKeySecret'], 'cn-beijing')

request = CommonRequest()
request.set_accept_format('json')
request.set_method('GET')
request.set_protocol_type('https') # https | http
request.set_domain('cs.cn-beijing.aliyuncs.com')
request.set_version('2015-12-15')

request.add_query_param('RegionId', "cn-beijing")
request.add_header('Content-Type', 'application/json')
request.set_uri_pattern('/gs/workflows')

response = client.do_action_with_exception(request)

print(response)
```

3.2. 通过AGS查询单个工作流

调用DescribeWorkflow查询单个工作流的详细信息。

请求信息

请求行RequestLine

```
GET /gs/workflow/{workflowName} HTTP/1.1
```

特有请求头RequestHead

无，请参见公共请求和返回结果。

请求体RequestBody

无

返回信息

返回行ResponseLine

```
HTTP/1.1 200 OK
```

特有返回头ResponseHead

无，请参见公共请求和返回结果。

返回体ResponseBody


```
{
 "create_time": "2020-01-15 16:30:25 +0800 CST",
 "duration": "1h15m33.529968361s",
 "finish_time": "0001-01-01 00:00:00 +0000 UTC",
 "input_data_size": "0",
 "job_name": "wgs-gpu-97xfn",
 "job_namespace": "1171330362041663",
 "output_data_size": "0",
 "status": "Running",
 "total_bases": "0",
 "total_reads": "0",
 "user_input_data": "{\"wgs_oss_region\":\"cn-shenzhen\",\"wgs_fastq_first_name\":\"fastq/huada/MGISEQ-200019SZ0002402\",\"wgs_fastq_second_name\":\"fastq/huada/MGISEQ-200019SZ0002402\",\"wgs_bucket_name\":\"gene-shenzhen\",\"wgs_vcf_file_name\":\"output/vcf/huada.vcf\",\"wgs_bam_file_name\":\"output/bam/huada.bam\",\"wgs_reference_file\":\"hg19\",\"wgs_service\":\"g\"}"
}
```


返回体解释

名称	类型	描述
create_time	String	工作流创建时间。
duration	String	工作流经过时长。
finish_time	String	任务结束时间。
input_data_size	String	输入数据大小。
job_name	String	工作流名称。
job_namespace	String	工作流所在命名空间。
output_data_size	String	输出数据大小。
status	String	工作流当前状态。
total_bases	String	碱基对个数。

名称	类型	描述
total_reads	String	Reads个数。
user_input_data	String	用户输入参数。

示例

请求示例（Python）

```
#!/usr/bin/env python
#coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
import os
client = AcsClient(os.environ['accessKeyID'], os.environ['accessKeySecret'], 'cn-beijing')

request = CommonRequest()
request.set_accept_format('json')
request.set_method('GET')
request.set_protocol_type('https') # https | http
request.set_domain('cs.cn-beijing.aliyuncs.com')
request.set_version('2015-12-15')

request.add_query_param('RegionId', "cn-beijing")
request.add_header('Content-Type', 'application/json')
request.set_uri_pattern('/gs/workflow/wgs-gpu-97xfn')

response = client.do_action_with_exception(request)

print(response)
```

3.3. 通过AGS创建基因工作流

调用StartWorkflow创建一个新的基因工作流。

请求信息

请求行RequestLine

```
POST /gs/workflow  HTTP/1.1
```

特有请求头RequestHead

无，请参见公共请求和返回结果。

请求体RequestBody

```
这里以mapping为例
{
    "workflow_type": "mapping",
    "service": "s" (#SLA: [n: normal|s: silver|g: gold|p: platinum])
    "mapping_oss_region": "cn-shenzhen",
    "mapping_fastq_first_filename": "MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz",
    "mapping_fastq_second_filename": "MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz",
    "mapping_bucket_name": "gene-shenzhen",
    "mapping_fastq_path": "fastq/MGISEQ2000",
    "mapping_reference_path": "reference/hg19", [Optional]
    "mapping_is_mark_dup": "true",
    "mapping_bam_out_path": "output/bamDirName",
    "mapping_bam_out_filename": "abc.bam"
}


这里以WGS为例
{
    "workflow_type": "wgs",
    "service": "s" (#SLA: [n: normal|s: silver|g: gold|p: platinum]),
    "wgs_oss_region": "cn-shenzhen",
    "wgs_fastq_first_filename": "MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz",
    "wgs_fastq_second_filename": "MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz",
    "wgs_bucket_name": "gene-shenzhen",
    "wgs_fastq_path": "fastq/MGISEQ2000",
    "wgs_reference_path": "reference/hg19",
    "wgs_vcf_out_path": "output/vcf",
    "wgs_vcf_out_filename": "abc.vcf"
}
```

请求体解释

名称	类型	描述
workflow_type	String	工作流类型，可选值：wgs或mapping。
service	String	SLA类型，可选值：n、s、g、p。

名称	类型	描述
mapping_oss_region	String	mapping oss数据的存放region。
mapping_fastq_first_filename	String	mapping的第一个fastq文件名。
mapping_fastq_second_filename	String	mapping的第二个fastq文件名。
mapping_bucket_name	String	存放mapping的bucket名称。
mapping_fastq_path	String	mapping的fastq文件路径。
mapping_reference_path	String	mapping的reference文件位置。
mapping_is_mark_dup	String	是否进行dup。
mapping_bam_out_path	String	bam文件输出路径。
mapping_bam_out_filename	String	bam文件输出名称。
wgs_oss_region	String	wgs oss数据的存放region。
wgs_fastq_first_filename	String	wgs的第一个fastq文件名。
wgs_fastq_second_filename	String	wgs的第二个fastq文件名。
wgs_bucket_name	String	存放wgs的bucket名称。
wgs_fastq_path	String	wgs的fastq文件路径。
wgs_reference_path	String	wgs的reference文件路径。
wgs_vcf_out_path	String	wgs的vcf输出路径。
wgs_vcf_out_filename	String	wgs的vcf输出文件名称。

返回信息

返回行ResponseLine

```
HTTP/1.1 200 OK
```

特有返回头ResponseHead

无，请参见公共请求和返回结果。

返回体ResponseBody

```
这里以mapping为例
{
    JobName: mapping-gpu-66xv7
  }

这里以wgs为例
  {
    JobName: wgs-gpu-tvltf
  }
```

返回体解释

名称	类型	描述
JobName	String	工作流名称

示例

请求示例（Python）

```
#!/usr/bin/env python
# coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
import os

os.environ.setdefault('DEBUG', 'sdk')

client = AcsClient(os.environ['accessKeyID'], os.environ['accessKeySecret'], 'cn-beijing')

request = CommonRequest()
request.set_accept_format('json')
request.set_method('POST')
request.set_protocol_type('https') # https | http
request.set_domain('cs.cn-beijing.aliyuncs.com')
request.set_version('2015-12-15')

request.add_query_param('RegionId', "cn-shenzhen")
request.add_header('Content-Type', 'application/json')
request.set_uri_pattern('/gs/workflow')
body = '''{"cli_version":"v1.0.1-882299b","wgs_bucket_name":"my-test-shenzhen","wgs_fastq_first_name"
:"MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz","wgs_fastq_second_name":"MGISE
Q/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz","wgs_oss_region":"cn-shenzhen","wgs_ref
erence_file":"hs37d5","wgs_service":"g","wgs_vcf_File_name":"vcf/MGISEQ_NA12878_hs37d5_13.vcf","wo
rkflow_type":"WGS"}'''
request.set_content(body.encode('utf-8'))
response = client.do_action_with_exception(request)
```

3.4. 通过AGS取消运行中的工作流

调用CancelWorkflow取消正在运行中的工作流。

请求信息

请求行RequestLine

```
PUT /gs/workflow/{workflowName}     HTTP/1.1
```

特有请求头RequestHead

无，请参见公共请求和返回结果。

请求体RequestBody

```
{
  "action": "cancel"
}
```

请求体解释

名称	类型	描述
action	String	执行的操作，目前只支持cancel。

返回信息

返回行ResponseLine

```
HTTP/1.1 200 OK
```

特有返回头ResponseHead

无，请参见公共请求和返回结果。

返回体ResponseBody

无

示例

请求示例（Python）

```
#!/usr/bin/env python
#coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
import os
client = AcsClient(os.environ['accessKeyID'], os.environ['accessKeySecret'], 'cn-beijing')

request = CommonRequest()
request.set_accept_format('json')
request.set_method('PUT')
request.set_protocol_type('https') # https | http
request.set_domain('cs.cn-beijing.aliyuncs.com')
request.set_version('2015-12-15')

request.add_query_param('RegionId', "cn-beijing")
request.add_header('Content-Type', 'application/json')
request.set_uri_pattern('/gs/workflow/wgs-gpu-97xfn')
body = '''{"action": "cancel"}'''
request.set_content(body.encode('utf-8'))

response = client.do_action_with_exception(request)

print(response)
```

3.5. 通过AGS删除工作流

调用RemoveWorkflow删除某个指定工作流。

请求信息

请求行Request Line

```
DELETE /gs/workflow/{workflowName}  HTTP/1.1
```

特有请求头Request Head

无，请参见公共请求和返回结果。

请求体Request Body

无

返回信息

返回行ResponseLine

```
HTTP/1.1 200 OK
```

特有返回头ResponseHead

无，请参见公共请求和返回结果。

返回体ResponseBody

无

示例

请求示例（Python）

```
#!/usr/bin/env python
#coding=utf-8

from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
import os
client = AcsClient(os.environ['accessKeyID'], os.environ['accessKeySecret'], 'cn-beijing')

request = CommonRequest()
request.set_accept_format('json')
request.set_method('DELETE')
request.set_protocol_type('https') # https | http
request.set_domain('cs.cn-beijing.aliyuncs.com')
request.set_version('2015-12-15')

request.add_query_param('RegionId', "cn-beijing")
request.add_header('Content-Type', 'application/json')
request.set_uri_pattern('/gs/workflow/wgs-gpu-97xfn')


response = client.do_action_with_exception(request)

print(response)
```

4.WDL工作流

4.1. 创建WDL工作流

WDL（Workflow Description Language）是由Broad Institute开发的一种流程开发语言，简单易用，能够有效的提高生物信息工作流的构建效率。本文介绍如何通过AGS在ACK集群上编写并执行WDL工作流。

前提条件

- 已创建ACK集群，详细介绍请参见创建Kubernetes托管版集群。
- 已拥有一个或多个存储服务（NAS存储卷或者符合NFS协议的文件存储、OSS存储），用于存储输入数据和输出结果，详细介绍请参见使用NAS动态存储卷。

在ACK上运行WDL的优势

- 兼容社区CronwellServer，完整兼容WDL的流程定义，对遗留流程无需修改，便可以通过AGS在ACK上运行WDL流程。WDL的详细介绍请参见WDL。
- 对Task资源申请优化，通过Pod Guarantee QoS方式，避免资源过度争取造成节点负载过高和效率下降。
- 与阿里云存储的无缝整合，目前支持直接访问OSS和NAS，并支持多数据源的挂载。

与AGS工作流的区别

- CronwellServer在以下方面仍然落后于云原生的AGS工作流。
  - 资源控制粒度（CPU、Mem min、Mem max）方面。
  - 调度优化，自动重试，资源上限的动态调整。
  - 监控、日志等方面。
- 在集群资源使用水位上低于AGS，在批量样本投递成功率上也低于AGS工作流。对于大批量重复性的工作流仍然建议改造成原生的AGS工作流来提升效率，详细介绍请参见创建工作流。
- 对于高CPU消耗的Mapping、HC、Mutecv2等流程可以使用AGS API来降低处理成本节省和加速，详细介绍请参见通过AGS处理全基因组测序WGS。

步骤一：部署应用

构建WDL工作流所需的组件被打包成为一个Helm Chart，作为一个应用放在应用市场中，避免了复杂的环境配置，方便进行一键部署。

1. 登录容器服务管理控制台。
2. 在控制台左侧导航栏中，单击**市场 > 应用目录**。
3. 在**应用目录**页面右侧搜索框中搜索ack-ags-wdl，找到ack-ags-wdl，然后单击ack-ags-wdl。
4. 在**应用目录 - ack-ags-wdl**页面单击**参数**页签。
5. 配置应用参数。

```
# PVCs to be mounted
naspvcs:
 - naspvc1
 - naspvc2
osspvcs:
 - osspvc1
```

```
 - osspvc2

# config in transfer-pvc.yaml、storageclass.yaml
naspvc1:
 # modify it to actual url of NAS/NFS server .
 server : "XXXXXX-fbi71.cn-beijing.nas.aliyuncs.com"
 # absolute path of your data on NAS/NFS, pls modify it to your actual path of data root
 path: "/tarTest" #eg "/tarTest"
 # strorage driver. flexVolume or csi,default is csi, if your kubernetes use flexVolume,chang it to flexVolu
me
 driver: "csi" # by default is csi, you could change to flexVolume for early version of K8s (<= 1.14.x)
 nasbasepath: "/ags-wdl-nas"
 # mountoptions. if you use local NFS, modify it to your mount options.
 # config in storageclass.yaml 、transfer-pvc.yaml
 mountVers: "3" #mount version-svc:
 mountOptions: "nolock,tcp,noresvport"  # mount options, for local NFS server, you could remove nor
esvport option

naspvc2:
 # modify it to actual url of NAS/NFS server .
 server : "XXXXXXX-fbi71.cn-beijing.nas.aliyuncs.com"
 # absolute path of your data on NAS/NFS, pls modify it to your actual path of data root
 path: "/tarTest/bwatest" #eg "/tarTest"
 # strorage driver. flexVolume or csi,default is csi, if your kubernetes use flexVolume,chang it to flexVolu
me
 driver: "csi" # by default is csi, you could change to flexVolume for early version of K8s (<= 1.14.x)
 nasbasepath: "/ags-wdl-nas2"
 # mountoptions. if you use local NFS, modify it to your mount options.
 # config in storageclass.yaml 、transfer-pvc.yaml
 mountVers: "3" #mount version-svc:
 mountOptions: "nolock,tcp,noresvport"  # mount options, for local NFS server, you could remove nor
esvport option

osspvc1:
 # modify it to actual bucket name
 bucket: "oss-test-tsk"
 # modify it to actual bucket url
 url: "oss-cn-beijing.aliyuncs.com"
 # mount options. It is not changed by default.
 options: "-o max_stat_cache_size=0 -o allow_other"
```

```
 akid: "XXXXXXX"
 aksecret: "XXXXXXX"
 # absolute path of your data on OSS
 path: "/"
 ossbasepath: "/ags-wdl-oss"

osspvc2:
 bucket: "oss-test-tsk"
 url: "oss-cn-beijing.aliyuncs.com"
 options: "-o max_stat_cache_size=0 -o allow_other"
 akid: "XXXXXXX"
 aksecret: "XXXXXXX"
 path: "/input"
 ossbasepath: "/ags-wdl-oss-input"

# Relative root path of input data and output data in your wdl.json, your path should add basepath bef
ore the relative path of the mount.
# e.g.
# {
# "wf.bwa_mem_tool.reference": "/ags-wdl/reference/subset_assembly.fa.gz",#reference filename。
# "wf.bwa_mem_tool.reads_fq1": "/ags-wdl/fastq_sample/SRR1976948_1.fastq.gz",#fastq1 filename
# "wf.bwa_mem_tool.reads_fq2": "/ags-wdl/fastq_sample/SRR1976948_2.fastq.gz",#fastq2 filename
# "wf.bwa_mem_tool.outputdir": "/ags-wdl/bwatest/output", #output path,the result will output in xx
xxxxxx.cn-beijing.nas.aliyuncs.com:/mydata_root/bwatest/output,you should make sure the path exist
s.
# "wf.bwa_mem_tool.fastqFolder": "fastqfolder" #任务执行的工作目录
# }

# workdir, you can choose nasbasepath or ossbasepath as your workdir
workdir : "/ags-wdl-oss"
# provisiondir,you can choose nasbasepath or ossbasepath as your provisiondir. task will create a volu
me dynamic to input/output date.
provisiondir: "/ags-wdl-oss-input"
#Scheduling the task to the specified node. e.g. nodeselector: "node-type=wdl"
nodeselector: "node-type=wdl"
#config in cromwellserver-svc.yaml
cromwellserversvc:
 nodeport : "32567" # cromwellserver nodeport, for internal access, you could use LB instead for extern
al access to cromwell server.

#config in config.yaml
```

```
config:
 # project namespace. default is wdl，Generally, no change is required
 namespace: "wdl" # namespace
 #tesk backand domain name. default is tesk-api，Generally, no change is required
 teskserver: "tesk-api"
 #cromwellserver domain name,default is cromwellserver，Generally, no change is required
 cromwellserver: "cromwellserver"
 #cromwellserver port,default is 8000，Generally, no change is required
 cromwellport: "8000"
```

- 配置应用存储卷。

  应用存储卷包括NAS和OSS。

参数	说明	是否必须配置	配置方法
naspvcs	需要挂载的NAS存储卷	是	当需要挂载的NAS存储卷PVC为 naspvc1 、naspvc2 时，配置如下。 naspvcs: - naspvc1 - naspvc2
osspvcs	需要挂载的OSS存储卷	是	当需要挂载的OSS存储卷PVC为 osspvc1 、osspvc2 时，配置如下。 osspvcs: - osspvc1 - osspvc2

- 配置NAS存储。

  每一个NAS存储卷对应执行任务容器中的一个路径 nasbasepath 。配置 naspvcs 需要为每一个NAS存储卷进行详细的参数配置。

  > **说明** 其他参数配置保持默认即可。

参数	说明	是否必须配置	配置方法
server	NAS IP	是	输入和输出数据。需要修改为您自己NAS的url，和path。

参数	说明	是否必须配置	配置方法
path	挂载的子目录	是	
driver	Flexvolume or CSI	是	集群的存储驱动，默认为CSI。当集群使用Flexvolume时，需将该参数修改为Flexvolume。
nasbasepath	NAS目录在应用中的映射目录	是	NAS目录在应用中的映射目录，在后续输入数据和输出数据时，需要将绝对目录中的 `server:path` 替换为 `basepath` 。例如， `server` 为192.168.0.1， `path` 为*/wdl*， `basepath` 为*/ags-wdl*，输入数据test.fq.gz在NAS中的存储目录为*192.168.0.1:/wdl/test/test.fq.gz* 。当需要填写输入路径时，应填写为*/ags-wdl/test/test.fq.gz*，此时应用能够访问到输入数据。本文后面均以 `server` 为192.168.0.1， `path` 为*/wdl*， `basepath` 为*/ags-wdl*作为示例。
mountOptions	挂载选项	是	挂载参数。当使用您自己的NFS时间时，可以修改该参数进行适配。
mountVers	挂载版本	是	

- 配置OSS存储。

  每一个OSS存储卷对应执行任务容器中的一个路径 `ossbasepath` 。配置 `osspvcs` 需要为每一个NAS存储卷进行详细的参数配置。

参数	说明	是否必须配置	配置方法
bucket	oss bucket	是	需要修改为您自己OSS的bucket name和url。
url	oss url	是	

参数	说明	是否必须配置	配置方法
options	挂载参数	是	默认为 `-o max_stat_cache_size=0 -o allow_other` ，不用修改。
akid	akid	是	您的ak信息，将会以Secret的方式部署在集群中。
aksecret	aksecret	是	
path	挂载的子目录	是	需要挂载的bucket的子目录。
ossbasepath	OSS目录在应用中的映射目录	是	OSS目录在应用中的映射目录，在后续输入数据和输出数据时，需要将绝对目录中的 `bucket:path` 替换为 `ossbasepath` 。例如， `bucket` 为shenzhen-test， `path` 为*/wdl*， `ossbasepath` 为*/ags-wdl*，输入数据test.fq.gz在oss中的存储目录为*shenzhen-test:/wdl/test/test.fq.gz*。当需要填写输入路径时，应填写为*/ags-wdl/test/test.fq.gz*，此时应用能够正确的访问到输入数据。

- 配置其他参数。

参数	说明	是否必须配置	配置方法
workdir	工作目录	是	需要在配置的 `nasbasepath` 、 `ossbasepath` 中选择一个作为工作目录，任务运行过程中的中间文件（script、error等）均会生成在该目录下。

参数	说明	是否必须配置	配置方法
provisiondir	动态创建pv目录	是	需要在配置的 nasbasepath 、 ossbasepath 中选择一个作为动态创建PV的目录，任务运行过程中的输入输出数据均会生成在该目录下创建的PV中。
nodeselector	任务调度支持label	否	如果需要将任务调度到特定的label机器时，可以填写该字段。配置 node-type=wdl 会调度所有的任务到 node-type=wdl 的节点上。
nodeport	cromwellserver的服务暴露端口	是	cromwellserver的服务暴露端口。当您需要用自己的Widdler客户端提交任务时，需要用到该参数，默认为32567。
namespace	项目命名空间	是	应用部署的命名空间，默认为wdl。
teskserver	tesk服务域名	是	tesk的服务域名，默认无需更改。
cromwellserver	cromwellserver	是	cromwellserver的服务域名，默认无需更改。
cromwellport	cromwellserver端口	是	cromwellserver的服务端口，默认无需更改。

6. 部署应用。

   在**应用目录 - ack-ags-wdl**页面右侧选择集群和命名空间，设置发布名称，单击**创建**。

   > **说明**

   执行以下命令，返回的结果显示cromwellcli、cromwellserver、tesk-api三个组件正常运行，说明部署成功。

   ```
   kubectl get pods -n wdl
   ```

   系统输出类似如下结果。

```
NAME                READY  STATUS    RESTARTS  AGE
cromwellcli-85cb66b98c-bv4kt     1/1    Running   0      5d5h
cromwellserver-858cc5cc8-np2mc   1/1    Running   0      5d5h
tesk-api-5d8676d597-wtmhc        1/1    Running   0      5d5h
```

步骤二：提交任务

在集群外部可以通过AGS和命令行提交任务，推荐使用AGS提交任务。

通过AGS提交任务

新版本AGS-CLI提供了向集群提交WDL任务的功能，您只需下载AGS，配置crowellserver地址即可。下载AGS请参见AGS命令行帮助。

1. 创建*bwa.wdl*文件和*bwa.json*文件。
   - b*wa.wdl*文件示例

```
task bwa_mem_tool {
 Int threads
 Int min_seed_length
 Int min_std_max_min
 String reference
 String reads_fq1
 String reads_fq2
 String outputdir
 String fastqFolder
 command {
     mkdir -p /bwa/${fastqFolder}
     cd /bwa/${fastqFolder}
     rm -rf SRR1976948*
     wget https://ags-public.oss-cn-beijing.aliyuncs.com/alignment/subset_assembly.fa.gz
     wget https://ags-public.oss-cn-beijing.aliyuncs.com/alignment/SRR1976948_1.fastq.gz
     wget  https://ags-public.oss-cn-beijing.aliyuncs.com/alignment/SRR1976948_2.fastq.gz
     gzip -c ${reference} > subset_assembly.fa
     gunzip -c ${reads_fq1} | head -800000 > SRR1976948.1
     gunzip -c ${reads_fq2} | head -800000 > SRR1976948.2
     bwa index subset_assembly.fa
     bwa aln subset_assembly.fa SRR1976948.1 > ${outputdir}/SRR1976948.1.untrimmed.sai
     bwa aln subset_assembly.fa SRR1976948.2 > ${outputdir}/SRR1976948.2.untrimmed.sai
 }
 output {
  File sam1 = "${outputdir}/SRR1976948.1.untrimmed.sai"
  File sam2 = "${outputdir}/SRR1976948.2.untrimmed.sai"
 }
 runtime {
  docker: "registry.cn-hangzhou.aliyuncs.com/plugins/wes-tools:v3"
  memory: "2GB"
  cpu: 1
 }
}
workflow wf {
 call bwa_mem_tool
}
```

- *bwa.json*文件示例

```
{
"wf.bwa_mem_tool.reference": "subset_assembly.fa.gz",#reference filename。
"wf.bwa_mem_tool.reads_fq1": "SRR1976948_1.fastq.gz",#fastq1 filename
"wf.bwa_mem_tool.reads_fq2": "SRR1976948_2.fastq.gz",#fastq2 filename
"wf.bwa_mem_tool.outputdir": "/ags-wdl/bwatest/output", #output path,the result will output in 1
92.168.0.1:/wdl/bwatest/output,you should make sure the path exists.
"wf.bwa_mem_tool.fastqFolder": "fastqfolder" #任务执行的工作目录
}
```

2. 将访问集群中的32567端口的流量转发到service cromwellserver上。

```
kubectl port-forward svc/cromwellserver 32567:32567
```

3. 配置cromwellserver地址。

   输入以下命令。

```
ags config init
```

   系统输出类似如下结果。

```
Please input your AccessKeyID
xxxxx
Please input your AccessKeySecret
xxxxx
Please input your cromwellserver url
192.168.0.1:32567
```

4. 在本地提交WDL任务。

   输入以下命令。

```
ags wdl run resource/bwa.wdl resource/bwa.json --watch #watch参数可以让该命令进行一个同步等待，直到该任务成功或失败。
```

   系统输出类似如下结果。

```
INFO[0000] bd747360-f82c-4cd2-94e0-b549d775f1c7 Submitted
```

5. （可选）您还可以在本地进行查询、删除WDL任务。
   - 查询WDL任务。

- 通过explain命令查询WDL任务。

  输入以下命令。

  ```
  ags wdl explain bd747360-f82c-4cd2-94e0-b549d775f1c7
  ```

  系统输出如下结果。

  ```
  INFO[0000] bd747360-f82c-4cd2-94e0-b549d775f1c7 Running
  ```

- 通过query命令查询WDL任务。

  输入以下命令。

  ```
  ags wdl query bd747360-f82c-4cd2-94e0-b549d775f1c7
  ```

  系统输出如下结果。

  ```
  INFO[0000] bd747360-f82c-4cd2-94e0-b549d775f1c7 {"calls":{"end":"0001-01-01T00:00:00.000Z","executionStatus":null,"inputs":null,"start":"0001-01-01T00:00:00.000Z"},"end":"0001-01-01T00:00:00.000Z","id":"b3aa1563-6278-4b2e-b525-a2ccddcbb785","inputs":{"wf_WGS.Reads":"/ags-wdl-nas/c.tar.gz"},"outputs":{},"start":"2020-10-10T09:34:56.022Z","status":"Running","submission":"2020-10-10T09:34:49.989Z"}
  ```

- 删除WDL任务。

  输入以下命令。

  ```
  ags wdl abort bd747360-f82c-4cd2-94e0-b549d775f1c7
  ```

  系统输出如下结果。

  ```
  INFO[0000] bd747360-f82c-4cd2-94e0-b549d775f1c7 Aborting
  ```

通过命令行提交任务

您需要编写WDL文件和输入的JSON文件，通过提供的镜像提交任务，其中cromwellserver的地址为集群IP:nodeport。

1. 创建*wa.wdl*文件和*bwa.json*文件。
   - *bwa.wdl*文件示例

```
task bwa_mem_tool {
 Int threads
 Int min_seed_length
 Int min_std_max_min
 String reference
 String reads_fq1
 String reads_fq2
 String outputdir
 String fastqFolder
 command {
     mkdir -p /bwa/${fastqFolder}
     cd /bwa/${fastqFolder}
     rm -rf SRR1976948*
     wget https://ags-public.oss-cn-beijing.aliyuncs.com/alignment/subset_assembly.fa.gz
     wget https://ags-public.oss-cn-beijing.aliyuncs.com/alignment/SRR1976948_1.fastq.gz
     wget  https://ags-public.oss-cn-beijing.aliyuncs.com/alignment/SRR1976948_2.fastq.gz
     gzip -c ${reference} > subset_assembly.fa
     gunzip -c ${reads_fq1} | head -800000 > SRR1976948.1
     gunzip -c ${reads_fq2} | head -800000 > SRR1976948.2
     bwa index subset_assembly.fa
     bwa aln subset_assembly.fa SRR1976948.1 > ${outputdir}/SRR1976948.1.untrimmed.sai
     bwa aln subset_assembly.fa SRR1976948.2 > ${outputdir}/SRR1976948.2.untrimmed.sai
 }
 output {
  File sam1 = "${outputdir}/SRR1976948.1.untrimmed.sai"
  File sam2 = "${outputdir}/SRR1976948.2.untrimmed.sai"
 }
 runtime {
  docker: "registry.cn-hangzhou.aliyuncs.com/plugins/wes-tools:v3"
  memory: "2GB"
  cpu: 1
 }
}
workflow wf {
 call bwa_mem_tool
}
```

- *bwa.json*文件示例

```
{
"wf.bwa_mem_tool.reference": "subset_assembly.fa.gz",#reference filename。
"wf.bwa_mem_tool.reads_fq1": "SRR1976948_1.fastq.gz",#fastq1 filename
"wf.bwa_mem_tool.reads_fq2": "SRR1976948_2.fastq.gz",#fastq2 filename
"wf.bwa_mem_tool.outputdir": "/ags-wdl/bwatest/output", #output path,the result will output in 1
92.168.0.1:/wdl/bwatest/output,you should make sure the path exists.
"wf.bwa_mem_tool.fastqFolder": "fastqfolder" #任务执行的工作目录
}
```

2. 提交WDL任务。

   输入以下命令。

   ```
   docker run -e CROMWELL_SERVER=192.16*.*.** -e CROMWELL_PORT=30384 registry.cn-beijing.aliyuncs.
   com/tes-wes/cromwellcli:v1 run resources/bwa.wdl resources/bwa.json
   ```

   系统输出类似如下结果。

   ```
   ----------Cromwell Links----------
   http://192.16*.*.**:30384/api/workflows/v1/5d7ffc57-6883-4658-adab-3f508826322a/metadata
   http://192.16*.*.**:30384/api/workflows/v1/5d7ffc57-6883-4658-adab-3f508826322a/timing
   {
     "status": "Submitted",
     "id": "5d7ffc57-6883-4658-adab-3f508826322a"
   }
   ```

4.2. WDL帮助示例

本文介绍WDL工作流的帮助示例。

查询任务列表

输入以下命令，查询WDL工作流的任务列表。

```
ags wdl list
```

预期输出：

```
+----------+--------------------------+----------------------------------+------------+
| JOB NAME |     CREATE TIME      |          JOB ID          | JOB STATUS |
+----------+--------------------------+----------------------------------+------------+
| wf     | 2020-10-21T06:34:10.859Z | 87546541-f41d-4745-9d1e-4b9b0f27b033 | Running   |
| wf     | 2020-10-21T05:52:28.360Z | cc195aee-41c7-44b0-89e5-4e94ba47f56b | Succeeded |
| wf     | 2020-10-21T05:52:08.340Z | 7209675e-4277-4d68-8848-725abca6b143 | Succeeded |
| wf     | 2020-10-21T03:50:01.010Z | 05ed1979-39d4-41db-8535-446919088a2b | Succeeded |
| wf     | 2020-10-21T03:45:40.783Z | b2394c34-9086-441f-af49-02735b5710ed | Succeeded |
+----------+--------------------------+----------------------------------+------------+
```

获取任务详细信息

输入以下命令，获取任务详细信息。

> **说明** 您可以在任务详细信息中获取JOB ID、错误日志保存路径等信息。

```
ags wdl query cc195aee-41c7-44b0-89e5-4e94ba47f56b
```

预期输出：

```
{
  "workflowName": "wf",
  "workflowProcessingEvents": [
    {
      "description": "Finished",
      "timestamp": "2020-10-21T05:53:05.109Z",
      "cromwellId": "cromid-89e8e7d",
      "cromwellVersion": "54-36be57c-SNAP"
    },
    {
      "timestamp": "2020-10-21T05:52:28.359Z",
      "cromwellVersion": "54-36be57c-SNAP",
      "cromwellId": "cromid-89e8e7d",
      "description": "PickedUp"
    }
  ],
  "metadataSource": "Unarchived",
  "actualWorkflowLanguageVersion": "draft-2",
  "submittedFiles": {
    "workflow": "task bwa_mem_tool {\n  String outputdir\n  String fastqFolder\n  String cpunum\n  Stri
ng bamfilename\n  command {\n    cd ${fastqFolder}\n    cp gene.bam.bai gene.bam.bai.test\n  }\n  runti
me {\n    docker: \"registry.cn-beijing.aliyuncs.com/shuangkun/genetools:v1.1\"\n    memory: \"2GB\"\n    cpu:
```

me {\n docker: \"registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1\"\n memory: \"2GB\"\n cpu: 1\n }\n }\ntask agstask {\n String config\n command {\n mkdir /root/.ags\n cp ${config} /root/.ags/\n ags remote list > /ags-wdl-nas/remote.txt\n }\n runtime {\n docker: \"registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1\"\n memory: \"2GB\"\n cpu: 1\n }\n }\nworkflow wf {\n call agstask\n call bwa_mem_tool\n} \n",
"root": "",
"options": "{\n\n}",
"inputs": "{\"wf.agstask.config\":\"/ags-wdl-nas/bwatest/ags/config\",\"wf.bwa_mem_tool.bamfilename\":\"gene.bam\",\"wf.bwa_mem_tool.cpunum\":\"6\",\"wf.bwa_mem_tool.fastqFolder\":\"/ags-wdl-nas/sample/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz\",\"wf.bwa_mem_tool.outputdir\":\"/ags-wdl-nas/bwatest/output\"}",
"workflowUrl": "",
"labels": "{}"
},
"calls": {
"wf.bwa_mem_tool": [
{
"executionStatus": "Done",
"stdout": "/ags-wdl-nas/wf/cc195aee-41c7-44b0-89e5-4e94ba47f56b/call-bwa_mem_tool/execution/stdout",
"backendStatus": "Complete",
"commandLine": "cd /ags-wdl-nas/sample/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz\ncp gene.bam.bai gene.bam.bai.test",
"shardIndex": -1,
"outputs": {},
"runtimeAttributes": {
"preemptible": "false",
"failOnStderr": "false",
"continueOnReturnCode": "0",
"docker": "registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1",
"maxRetries": "0",
"cpu": "1",
"memory": "2 GB"
},
"callCaching": {
"allowResultReuse": false,
"effectiveCallCachingMode": "CallCachingOff"
},
"inputs": {
"fastqFolder": "/ags-wdl-nas/sample/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz",
"outputdir": "/ags-wdl-nas/bwatest/output",

```
        "cpunum": "6",
        "bamfilename": "gene.bam"
      },
      "returnCode": 0,
      "jobId": "task-6dbeff7e",
      "backend": "TESK",
      "end": "2020-10-21T05:53:04.317Z",
      "dockerImageUsed": "registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1",
      "stderr": "/ags-wdl-nas/wf/cc195aee-41c7-44b0-89e5-4e94ba47f56b/call-bwa_mem_tool/execution/stderr",
      "callRoot": "/ags-wdl-nas/wf/cc195aee-41c7-44b0-89e5-4e94ba47f56b/call-bwa_mem_tool",
      "attempt": 1,
      "executionEvents": [
        {
          "startTime": "2020-10-21T05:53:03.351Z",
          "description": "UpdatingJobStore",
          "endTime": "2020-10-21T05:53:04.317Z"
        },
        {
          "startTime": "2020-10-21T05:52:29.408Z",
          "endTime": "2020-10-21T05:52:29.408Z",
          "description": "Pending"
        },
        {
          "startTime": "2020-10-21T05:52:29.970Z",
          "endTime": "2020-10-21T05:53:03.351Z",
          "description": "RunningJob"
        },
        {
          "endTime": "2020-10-21T05:52:29.957Z",
          "startTime": "2020-10-21T05:52:29.408Z",
          "description": "RequestingExecutionToken"
        },
        {
          "description": "WaitingForValueStore",
          "startTime": "2020-10-21T05:52:29.957Z",
          "endTime": "2020-10-21T05:52:29.957Z"
        },
        {
          "startTime": "2020-10-21T05:52:29.957Z",
          "endTime": "2020-10-21T05:52:29.970Z",
```

```
            "description": "PreparingJob"
          }
        ],
        "start": "2020-10-21T05:52:29.408Z"
      }
    ],
    "wf.agstask": [
      {
        "executionStatus": "Done",
        "stdout": "/ags-wdl-nas/wf/cc195aee-41c7-44b0-89e5-4e94ba47f56b/call-agstask/execution/stdout",
        "backendStatus": "Complete",
        "commandLine": "mkdir /root/.ags\ncp /ags-wdl-nas/bwatest/ags/config /root/.ags/\nags remote list
> /ags-wdl-nas/remote.txt",
        "shardIndex": -1,
        "outputs": {},
        "runtimeAttributes": {
          "preemptible": "false",
          "failOnStderr": "false",
          "continueOnReturnCode": "0",
          "docker": "registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1",
          "maxRetries": "0",
          "cpu": "1",
          "memory": "2 GB"
        },
        "callCaching": {
          "allowResultReuse": false,
          "effectiveCallCachingMode": "CallCachingOff"
        },
        "inputs": {
          "config": "/ags-wdl-nas/bwatest/ags/config"
        },
        "returnCode": 0,
        "jobId": "task-1aa59269",
        "backend": "TESK",
        "end": "2020-10-21T05:53:03.297Z",
        "dockerImageUsed": "registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1",
        "stderr": "/ags-wdl-nas/wf/cc195aee-41c7-44b0-89e5-4e94ba47f56b/call-agstask/execution/stderr",
        "callRoot": "/ags-wdl-nas/wf/cc195aee-41c7-44b0-89e5-4e94ba47f56b/call-agstask",
        "attempt": 1,
        "executionEvents": [
```

```
        {
          "startTime": "2020-10-21T05:52:29.965Z",
          "description": "RunningJob",
          "endTime": "2020-10-21T05:53:02.773Z"
        },
        {
          "description": "PreparingJob",
          "startTime": "2020-10-21T05:52:29.957Z",
          "endTime": "2020-10-21T05:52:29.965Z"
        },
        {
          "description": "WaitingForValueStore",
          "startTime": "2020-10-21T05:52:29.957Z",
          "endTime": "2020-10-21T05:52:29.957Z"
        },
        {
          "startTime": "2020-10-21T05:52:29.408Z",
          "endTime": "2020-10-21T05:52:29.957Z",
          "description": "RequestingExecutionToken"
        },
        {
          "startTime": "2020-10-21T05:52:29.408Z",
          "description": "Pending",
          "endTime": "2020-10-21T05:52:29.408Z"
        },
        {
          "endTime": "2020-10-21T05:53:03.297Z",
          "startTime": "2020-10-21T05:53:02.773Z",
          "description": "UpdatingJobStore"
        }
      ],
      "start": "2020-10-21T05:52:29.408Z"
    }
  ]
},
"outputs": {},
"workflowRoot": "/ags-wdl-nas/wf/cc195aee-41c7-44b0-89e5-4e94ba47f56b",
"actualWorkflowLanguage": "WDL",
"id": "cc195aee-41c7-44b0-89e5-4e94ba47f56b",
"inputs": {
  "wf.agstask.config": "/ags-wdl-nas/bwatest/ags/config",
```

```
    "wf.bwa_mem_tool.fastqFolder": "/ags-wdl-nas/sample/MGISEQ2000_PCR-free_NA12878_1_V100003043
_L01_1.fq.gz",
    "wf.bwa_mem_tool.outputdir": "/ags-wdl-nas/bwatest/output",
    "wf.bwa_mem_tool.cpunum": "6",
    "wf.bwa_mem_tool.bamfilename": "gene.bam"
  },
  "labels": {
    "cromwell-workflow-id": "cromwell-cc195aee-41c7-44b0-89e5-4e94ba47f56b"
  },
  "submission": "2020-10-21T05:52:13.812Z",
  "status": "Succeeded",
  "end": "2020-10-21T05:53:05.109Z",
  "start": "2020-10-21T05:52:28.360Z"
}
```


挂载PV

1. 登录容器服务管理控制台。
2. 在控制台左侧导航栏中，单击**集群**。
3. 在**集群列表**页面中，单击目标集群名称或者目标集群右侧**操作**列下的**详情**。
4. 在集群管理页左侧导航栏中单击**发布**。
5. 在**发布**页面单击**Helm**页签，单击ack-ags-wdl**操作**列的**更新**。
6. 在**更新发布**对话框的文本框中挂载PV，以下以naspvc3为例。
   i. 添加PV名称。在**更新发布**对话框的文本框中找到 `naspvcs` 字段，在 `naspvcs` 字段下增加所要添加的PV名称。

   ```
   naspvcs:
   - naspvc1
   - naspvc2
   - naspvc3
   ```

   ii. 添加PV配置内容。在**更新发布**对话框的文本框添加PV配置内容，然后单击**更新**。

   ```
   naspvc3:
    driver: csi
    mountOptions: nolock,tcp,noresvport
    mountVers: "3"
    nasbasepath: /ags-wdl-nas3
    path: /tarTest/sample
    server: xxxxxxxx.cn-beijing.nas.aliyuncs.com
   ```

   完成挂载PV后，后续创建的任务可以读取或写入新挂载的PV。

4.3. 通过AGS服务调用加速工作流

通过本地的WDL工作流和远程的AGS工作流组成的混合工作流，可以将耗时长、计算量大、标准流程的任务放在AGS服务端中执行。将耗时短、所需资源有限、需要自行定制流程的任务放在本地的WDL工作流中执行，数据通过OSS进行中间流转，从而有效的提高任务执行效率、节省资源成本。本文将通过RemoteApi实现Mapping过程为例，演示如何编写并运行一个AGS混合工作流。

前提条件

- 已提交阿里云基因服务公测申请，加入公测白名单。

  > **说明** 如果您使用的是子账号，公测申请时请提供子账号的UID。请登录账号管理控制台查询UID。

- 已在应用目录中安装ack-ags-wdl，详细介绍请参见创建WDL工作流的步骤一：部署应用。

操作步骤

1. 配置AGS。
   i. 下载和安装AGS，详细介绍请参见AGS命令行帮助。

   ```
   ags config init
   ```

   配置AGS完成后，会自动生成AGS配置文件*config*。
   ii. 将AGS配置文件*config*保存在*/ags-wdl-nas/bwatest/ags/config*路径下。
2. 创建*agsHybrid.wd*文件。

```
task agstask {
  String method
  String region
  String file1
  String file2
  String bucket
  String outputbam
  String ref
  String service
  String config
  command {
    mkdir /root/.ags
    cp ${config} /root/.ags/
    ags remote run ${method} --region ${region} --fastq1 ${file1} --fastq2 ${file2} --bucket ${bucket} --out
put-bam ${outputbam} --reference ${ref} --service ${service} --watch
  }
  runtime {
  docker: "registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1"
  memory: "20GB"
```

```
  cpu: 6
  }
}
task downloaddata {
  String osspath
  String naspath
  command {
    cp ${osspath}/gene.bam ${naspath}
  }
  runtime {
  docker: "ubuntu"
  memory: "2GB"
  cpu: 1
  }
}

task bwa_mem_tool {
  String outputdir
  String fastqFolder
  String cpunum
  String bamfilename
  command {
    cd ${fastqFolder}
    samtools index -@ ${cpunum} ${bamfilename}
  }
  runtime {
  docker: "registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1"
  memory: "20GB"
  cpu: 6
  }
}

task uploaddata {
  String osspath
  String naspath
  command {
    cp ${naspath}/gene.bam.bai ${osspath}
  }
  runtime {
  docker: "ubuntu"
  memory: "2GB"
```

```
  memory: 10G
  cpu: 1
  }
 }
 workflow wf {
   call agstask
   call downloaddata
   call bwa_mem_tool
   call uploaddata
 }
```

3. 创建*agsHybrid.json*文件。

   以下文件中的*/ags-wdl-oss/*、*/ags-wdl-nas/*需要设置为部署ack-ags-wdl的Bucket和NAS的路径。

```
{
  "wf.agstask.config": "/ags-wdl-nas/bwatest/ags/config",
  "wf.agstask.method": "mapping",
  "wf.agstask.region": "shenzhen",
  "wf.agstask.file1": "sample/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz",
  "wf.agstask.file2": "sample/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz",
  "wf.agstask.bucket": "my-test-shenzhen",
  "wf.agstask.outputbam": "output/bam/gene.bam",
  "wf.agstask.ref": "hg19",
  "wf.agstask.service": "s",
  "wf.downloaddata.osspath": "/ags-wdl-oss/output/bam",
  "wf.downloaddata.naspath": "/ags-wdl-nas/sample",
  "wf.bwa_mem_tool.cpunum": "6",#cpu num
  "wf.bwa_mem_tool.bamfilename": "gene.bam",
  "wf.bwa_mem_tool.outputdir": "/ags-wdl-nas/bwatest/output", #output path,the result will output in 192.168.0.1:/wdl/bwatest/output,you should make sure the path exists.
  "wf.bwa_mem_tool.fastqFolder": "/ags-wdl-nas/sample/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz", #任务执行的工作目录
  "wf.uploaddata.naspath": "/ags-wdl-nas/sample",
  "wf.uploaddata.osspath": "/ags-wdl-oss/output/bam",
  }
```

4. 创建混合工作流。

```
ags wdl run agsHybrid.wdl agsHybrid.json
```

   创建混合工作流成功后，会自动执行混合工作流。

   找到ags-wdl-oss对应Bucket，进入Bucket下的*output/bam*，可以看到*gene.bam.ba*文件。说明混合工作流创建并执行成功。