ALIBABA CLOUD

Alibaba Cloud

Container Service for Kubernetes

User Guide for Genomics Service

Document Version: 20220513

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.AGS overview

Alibaba Cloud Genomics Service (AGS) is an ultra-fast, cost-effective, and high-precision genome sequencing and secondary data analysis service that is developed by Alibaba Cloud. AGS provides containerized workflows for genomics computing based on Container Service for Kubernetes (ACK). AGS provides an out-of-the-box acceleration API. You can call this API without the need to create a cluster. This topic describes AGS and its benefits.

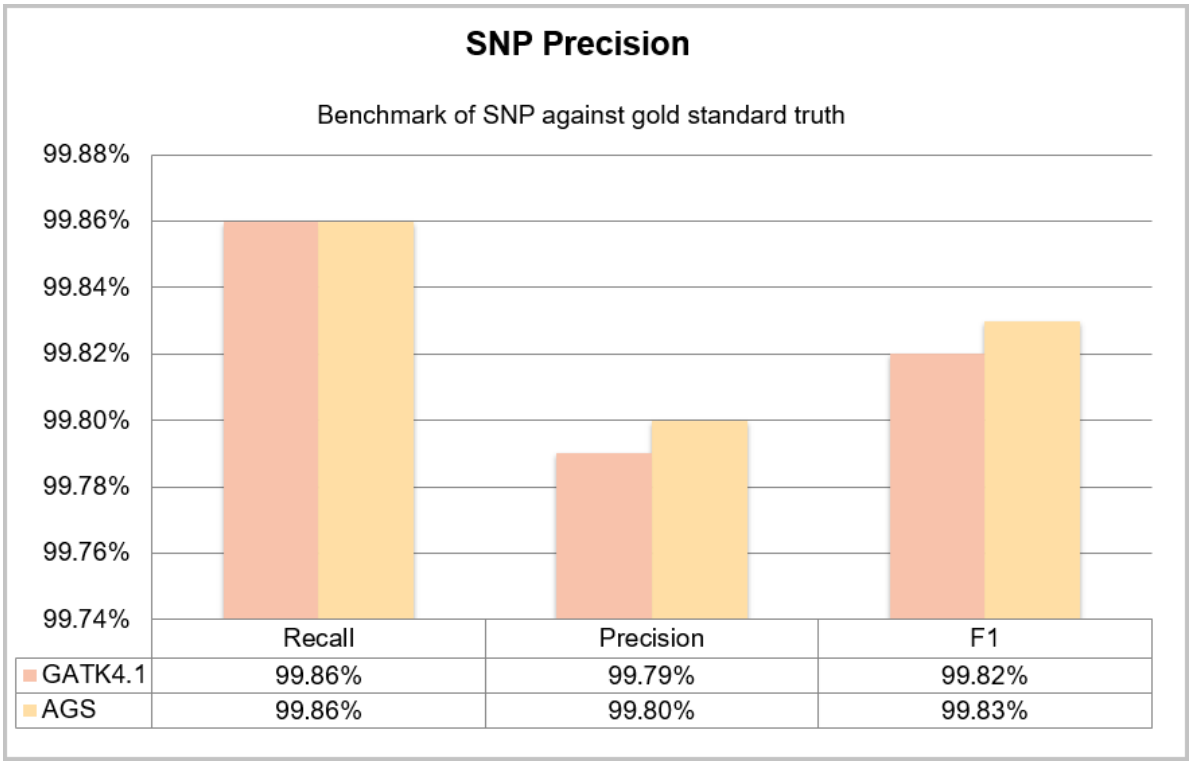
What is AGS

AGS is commonly used in genome sequencing and secondary data analysis. It requires only 15 minutes to complete a high-precision 30x whole genome sequencing (WGS) process, which includes gene comparison, sequencing, deduplication, and mutation detection. Compared with the traditional solution, AGS offers a 120 times speed improvement and is 2 to 4 times faster than the globally used FPGA/GPU solution.

AGS provides strong support for genetic disease detection and cancer screening by analyzing and identifying the mutation mechanism of personal genome sequences. In the future, it will play a bigger role in clinical medicine and genetic diagnosis. The whole human genome consists of about three billion base pairs. One 30x WGS sample is about 100 GB in size. AGS offers great benefits in terms of computation speed, precision, cost, usability, and compatibility with upstream sequencers. It is also suitable in scenarios such as SNP/INDEL and copy number variation (CNV) detection in DNAs, and DNA and RNA virus detection.

Benefits

- Fast speed and high precision. In actual tests, the solution completes the sequencing and secondary analysis of eight 30x WGS samples within 15 minutes. The solution can be used to assemble, sequencing, deduplicate, and detect mutation on 720 billion base pairs within 15 minutes. The solution achieves a 120 times speed improvement without compromising the precision. If you compare the NA12878 sample and gold-standard Variant Call Format (VCF) files, the SNP precision reaches 99.80% and the precision of secondary analysis is higher than or equivalent to the output of BWA-0.7.17/GATK 4.1.3.



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

- Cost-effectiveness. ACK and AGS provide PaaS-based acceleration capabilities and assist BGI Group to perform secondary data analysis on large-scale FASTQ data from genome sequencers based on a hybrid cloud architecture. This solution reduces the costs of secondary data analysis and shortens the delivery cycle by 95%.
- Wide application scope. AGS provides the following features to enable a wide application scope:
  - AGS meets flexible needs and supports custom analysis processes for different platforms and data types without compromising the throughput. AGS provides major sequencing service providers and research institutions with solutions for more simplified and efficient storage, automated analysis, data transmission, project collaboration, and bioinformatics tool development.
  - AGS supports Kubernetes-native workflows and allows you to execute DAG-structured workflows in Kubernetes clusters. AGS is also applicable in scenarios such as genomics computing and data processing.
- Easy of use. AGS supports elastic scheduling in large-scale computing scenarios based on the auto scaling feature. In the practice of this solution, you do not need to plan computing resources,

processing logic, and data caching. You only need to upload data in FASTQ files to Object Storage Service (OSS) and authorize AGS to access the OSS buckets. The data analysis process will be efficiently completed and the result will be uploaded to the specified storage.

References

In addition, AGS also fixes the issues in workflow management, massive data storage, migration and transmission, and security compliance requirements. For more information, see the following topics:

- Use AGS to process WGS tasks
- Use AGS to perform virus sequencing
- Use AGS to analyze tumor samples
- Use workflows in Kubernetes clusters
- Introduction to AGS CLI

2.AGS acceleration API

2.1. Use AGS to process WGS tasks

Alibaba Cloud Genomics Compute Service (AGS) enables rapid processing of whole genome sequencing (WGS) tasks, including gene comparison, sequencing, deduplication, and variant detection. This topic describes how to manage WGS workflows by using AGS CLI.

Prerequisites

You have applied for the public preview of AGS and are added to the whitelist.

> **Note** If your account is a Resource Access Management (RAM) user, you must provide the UID of the RAM user when you apply for the public preview. You can log on to the Account Management console to obtain the UID of the RAM user.

Preparations

Grant permissions and prepare data.

1. Set up AGS.

   For more information about how to download and install AGS CLI, see Introduction to AGS CLI.

   ```
   ags config init
   ```

2. Prepare an Object Storage Bucket (OSS) bucket and grant AGS the read and write permissions on the bucket and the permissions to call GetBucketInfo.

   > **Note**
   > - Make sure that the OSS bucket that you want to use is owned by the current account. Otherwise, we recommend that you create a new OSS bucket within the account and grant the permissions to call GetBucketInfo.
   > - If your account is a RAM user, you must attach the AliyunOSSFullAccess policy to the RAM user. For more information, see Grant permissions to a RAM user.
   > - If your account is a RAM user, we recommend that you create a new OSS bucket within the account to ensure that the OSS bucket is owned by the RAM user. You can run the following command to check the owner of the OSS bucket:
   >
   > ```
   > ossutil stat oss://<your new bucket name>
   > ```

   ```
   Usage:
   ags config oss <your bucket name>
   e.g.
   ags config oss my-test-shenzhen
   ```

3. Upload FASTQ data to the OSS bucket by using ossutil.

   For more information about how to download and install ossutil, see 下载和安装.

> **Note** AGS supports only WGS and whole exome sequencing (WES) of human genome data. The comparisons of methylated genome data, plant genome data, and animal genome data are not supported.

- Upload data in simple mode.

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

- Upload a large amount of data.

  Refer to the following example to organize samples in pairs and store them in different directories.

```
./samples
├── sample1
│   ├── fastq_L1.tgz
│   └── fastq_L2.tgz
└── sample2
      ├── fastq_L1.tgz
      └── fastq_L2.tgz
```

  Run the following command to upload data in batches. `ossutil` commands support resumable upload.

```
Usage:
ossutil --recursive cp -r -u --parallel < number of concurrent tasks>  <local dir of
fastq or nfs path> <path of  oss bucket >
e. g.
ossutil --recursive cp -r -u  --parallel 16  ./samples  oss://mybucket/samples
```

Start a WGS workflow

We recommend that you use version hs37d5 of human reference genome `hg19`. This is also the default genome.

> **Note** Version hs37d5 of human reference genome `hg19` has the following characteristics:
>
> - Excludes alternative (ALT) contigs.
> - Hard masks pseudoautosomal regions (PARs) on the Y chromosome (chrY).
> - Includes decoy contigs.
>
> AGS is ALT-aware, which enables AGS to identify and process ALT contigs. Genome `UCSC hg19` includes ALT contigs but `does not` have the other two characteristics, which decreases the accuracy of variant detection. For more information, click Which human reference genome to use?.

```
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 pa
ir 2, fastq-path\filename \
--fastq2 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz  # filename of fastq p
air 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 i
nclude decoy contig, no support for UCSC hg19. hg38: GRCh38/hg38 include decoy
--reference-group "\"@RG\\tID:TEST\\tSM:12878\\tPL:MGISEQ2000\"" # allow to specify referen
ce 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
### Process FASTQ files that include multiple lanes and samples in batches
MGISAMPLE001 is a set of WGS sequencing samples of multiple lanes. You can combine and comp
ute the sequencing results of multiple lanes by specifying the sample directory --fastq1 MG
ISAMPLE001 or --fastq2 MGISAMPLE001.
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
```

Click Use AGS to run a WGS workflow to watch a demonstration of how to use AGS CLI to run a WGS workflow.

Start a Mapping workflow

Use **--fastq1** and **--fastq2** to specify *FASTQ files* and use **--output** to specify the output path of *bam*.

```
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 pa
ir 2, fastq-path\filename \
--fastq2 MGISEQ/MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz  # filename of fastq p
air 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
```

Click Use AGS to run a Mapping workflow. Use AGS to run a Mapping workflow to watch a demonstration of how to use AGS CLI to run a Mapping workflow.

List remote workflows

```
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 |
+---------------+-------------------------------+
```

Obtain workflow details

```
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  |
+----------------------+----------------------------------+
```

Cancel a running workflow

```
Usage:
ags remote cancel  <workflow id>
e.g.
ags remote cancel wgs-gpu-zls6r
INFO[0000] Successed to cancel wgs-gpu-zls6r
```

Remove a finished workflow

You can remove successful and failed workflows. However, you cannot remove running workflows.

```
Usage:
ags remote remove <workflow id>
e.g.
ags remote remove wgs-gpu-zls6r
INFO[0000] Successed to remove wgs-gpu-zls6r
```

2.2. Use AGS to analyze tumor samples

Alibaba Cloud Genomics Service (AGS) allows you to call `mutect2` tasks to detect somatic mutations, including single-nucleotide variants (SNVs) and insertions and deletions (Indels). This topic describes how to use AGS to analyze tumor samples.

Context

AGS `mutect2` supports the following scenarios:

- Tumor and normal samples: Skip germline mutations in normal individuals during the analytics process.
- Tumor samples: Perform correlation analytics on individual tumor samples.

`mutect2` adopts the same mutation detection methods as GATK4.1.3 but provides a speed that is 30 to 80 times faster. AGS can complete mutation detection on 90 billion base pairs within 10 minutes.

mutect2 example

Click Use AGS mutect2 to analyze tumor samples to watch a video tutorial of how to use AGS mutect2 to analyze tumor samples.

Analyze tumor and normal samples

Based on normal samples with a given match, `mutect2` detects only somatic mutations. `mutect2` will skip the logic of mutations that are clearly normal in the germline based on the provided evidence, such as matched normal individuals. This avoids wasting computing resources on germline events.

- Usage

  Run the following command:

```
ags remote run mutect2 \
--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
```

  The following output is returned:

- Sample output:

```
ags remote run mutect2 \
--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          |   DURA
TION    | 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                         |
+-------------------------+---------------------------+
```

Analyze individual tumor samples

mutect2 performs analytics on individual samples, such as tumor samples or normal samples.

- Usage

  Run the following command:

```
ags remote run mutect2 \
--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
```

  The following output is returned:

- Sample output:

```
ags remote run mutect2 \
--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          | DURA
TION |          FINISH TIME          | TOTAL 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                       |
+-------------------------+-------------------------+
```

2.3. Use AGS to perform virus sequencing

In the process of pathogen identification, it is an accurate and effective strategy to compare the samples under test with the genome databases of known pathogens. Alibaba Cloud Genomics Service (AGS) supports quick comparison of metagenome sequencing data. This topic describes how to use AGS to perform virus sequencing.

Prerequisites

You have applied for the public preview of AGS and your application is approved.

> **Note** If your account is a Resource Access Management (RAM) user, you must also provide the UID of the RAM user for the application. You can log on to the Account Management console to obtain the UID of the RAM user.

Context

Coronavirus SARS-CoV-2 RNA (hereinafter referred to as COVID-19) can be detected in the upper or lower respiratory tract with a few weeks after the onset of the disease. At present, many types of RT-PCR kits are available to diagnose COVID-19. RT-PCR tests are cost-effective and can provide feedback within a short period of time. However, due to factors such as virus concentration and kit quality, RT-PCR tests are prone to false negatives. Multiple tests are required for confirmation, and only one specific virus can be checked at a time.

Metagenomic next-generation sequencing (mNGS) for pan-pathogen detection directly extracts a certain proportion of nucleic acid fragments (including a large amount of human nucleic acids and a small amount of microbial nucleic acids) from clinical samples for sequencing, data comparison with database engines, and bioinformation analysis. This enables unbiased identification of pathogenic microorganisms. This technology has made great contributions to the early detection and accurate sequencing of COVID-19.

Compared with RT-PCR tests, mNGS testing requires longer detection periods but provides higher accuracy. It can check multiple virus types at a time and monitor the mutations that may occur during the transmission of viruses. This allows you to enhance the prevention and control of viruses. Clinically, it is advisable to check suspected patients again, especially those who cannot be diagnosed by fluorescent quantitative PCR. This further improves the accuracy of test results and effectively prevents false negatives caused by virus mutations. After the genome of the pathogen is identified and updated in databases, the pathogen can be accurately detected by comparing nucleic acid sequences.

AGS supports quick comparison of mNGS data. It can complete the comparison of 3.2 billion base pairs (22 million reads) of an alveolar sample and the reference sequence of known pathogen genomes (including COVID-19, 39 BetaCov RNAs, and 9,334 known viruses) within 60 seconds. It also allows you to upload custom virus libraries to compare with samples under test. You only need to create an Object Storage Service (OSS) bucket and run AGS commands to complete the entire comparison process. AGS provides reports about high-quality alignments of reads and supports detection of multiple pathogen types. This provides solid data support for researches on the protein and mutations of COVID-19.

Genome sequencing service providers, disease control centers, hospitals, universities and research institutes, and pharmaceutical companies are all welcome to apply for trial.

Preparations

1. Set up AGS. For more information about how to download and install AGS CLI, see Introduction to AGS CLI.

   Sample command:

   ```
   ags config init
   ```

2. For more information about how to download and install ossutil, see 下载和安装.
3. Create an Alibaba Cloud account, activate OSS, and create an OSS bucket to store mNGS data. For example, the URL of the OSS bucket may be oss://my-test-shenzhen. For more information, see Get started with OSS.

   > **Note** If your account is a RAM user, we recommend that you create a new OSS bucket under the account to ensure that the OSS bucket is owned by the RAM user. You can run the following command to check the owner of the OSS bucket:
   >
   > ```
   > ossutil stat oss://<your new bucket name>
   > ```

4. Grant AGS the permissions to call GetBucketInfo. This allows AGS to obtain information about your

OSS bucket.

> **Note**
>
> - Make sure that the OSS bucket that you want to use is owned by the current account. Otherwise, we recommend that you create a new OSS bucket under the account and grant the permissions to call GetBucketInfo.
> - If your account is a RAM user, you must attach the AliyunOSSFullAccess policy to the RAM user. For more information, see Grant permissions to a RAM user.

Sample command:

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

Examples of rna-mapping

Click Use AGS to test rna-mapping viruses to watch a video tutorial of how to use AGS CLI to test rna-mapping viruses.

Compare with COVID-19

1. Run ossutil commands to upload mNGS data to your OSS bucket.

   Sample command:

   ```
   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. Run a task to compare mNGS data with the known RNA sequence data.

   This topic compares sequencing sample ICU6G_S2_L001 with COVID-19.

   Sample command:

   ```
   Usage:
   ags remote run rna-mapping \ # <rna-mapping>: RNA sequence comparison task.
   --region <region_id> \ #
   <cn-shenzhen|cn-beijing|... >: region ID. Only the China (Shenzhen) and China (Beijing)
   regions are supported.
   --bucket <bucket_name> \ # <bucket_name> The name of the OSS bucket.
   --fastq1 <path_fq1> \ # Path of sequence data fq1.
   --fastq2 <path_fq2>\ # Path of sequence data fq2.
   --output-bam <path_of_output_bam> \ # Output path of comparison results in BAM format.
   The report is in the same path in TXT format.
   --reference [sars-cov-2 | betacov-ncbi-39 | viral-9334 | <path of RNA library reference
   in specified bucket >] # The reference sequence presents COVID-19 and 39 known BetaCov
   RNAs. You can also specify a custom virus sequence library.
   ```

   Sample command:

```
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. Check the comparison results.

   In the preceding example, the comparison of 10 million reads of mNGS data and COVID-19 sequence MN908947.3 produced 3,629 high-quality alignments of reads, and the number of reads whose alignment score (AS) exceeds 120 in the characteristic interval of COVID-19 was 404. This indicates that COVID-19 RNA sequences can be accurately detected in the sequence data.

   Sample results:

```
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 | 43
s      | 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. Run the following ossutil command to download the results and report.

   Sample command:

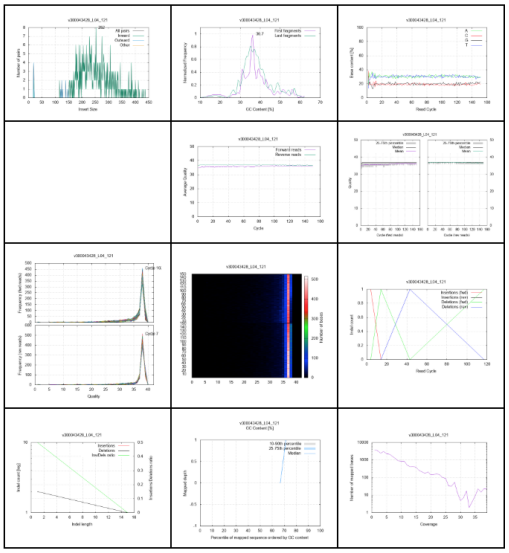
```
ossutil ls oss://my-test-shenzhen/bam/ICU6G_S2.bam
LastModifiedTime                   Size(B)  StorageClass   ETAG
ObjectName
2020-03-04 16:41:11 +0800 CST       356320      Standard   9596D012A30438A0073A2A0B38F5
D578      oss://my-test-shenzhen/bam/ICU6G_S2.bam
2020-03-04 16:41:11 +0800 CST         2889      Standard   63175E7180D110BA9D3BAB34F431
3C59      oss://my-test-shenzhen/bam/ICU6G_S2.bam.cov.txt
2020-03-04 16:41:11 +0800 CST          396      Standard   940D51FF7ECFF60B5E5A41D1F635
180D      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 .
```

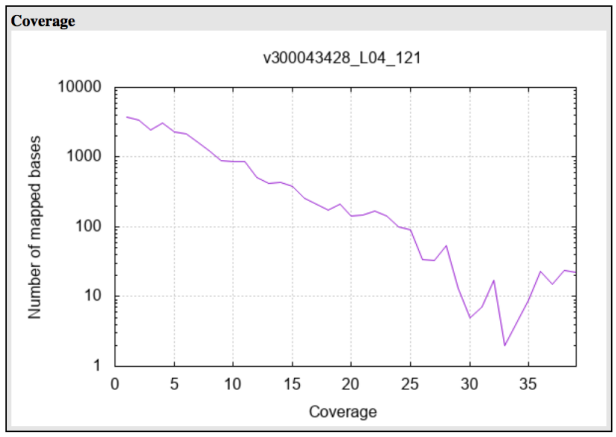
Sample report:

```
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 with very high mappQ and AS reads: True
        21571     21581     21591     21601     21611     21621     21631
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
ATGTT GTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGT                           CAATTACCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGA         CCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCA  agtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgcca         AGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
ATGTTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
atgtttgtttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
 TGTTTGTTTTTCTTGTTTT     CACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
 tgtttgtttttcttgtttt
   TTTGTTTTTCTTGTTTTATTGCCACTAGTCTCTAGTCAGTGTGTTAATCTTACAACCAGAACTCAATTACCCCCTGC
      gttttcttgttttattgccactagtctctagtcagtgtgttaatcttacaaccagaactcaattacccccctgc
```

5. Further analyze the data.

   You can use tools such as samtools stats and plot-bamstats to further analyze the data and explore their similarities in coverage and depth. You can also perform protein composition analysis and mutation analysis.



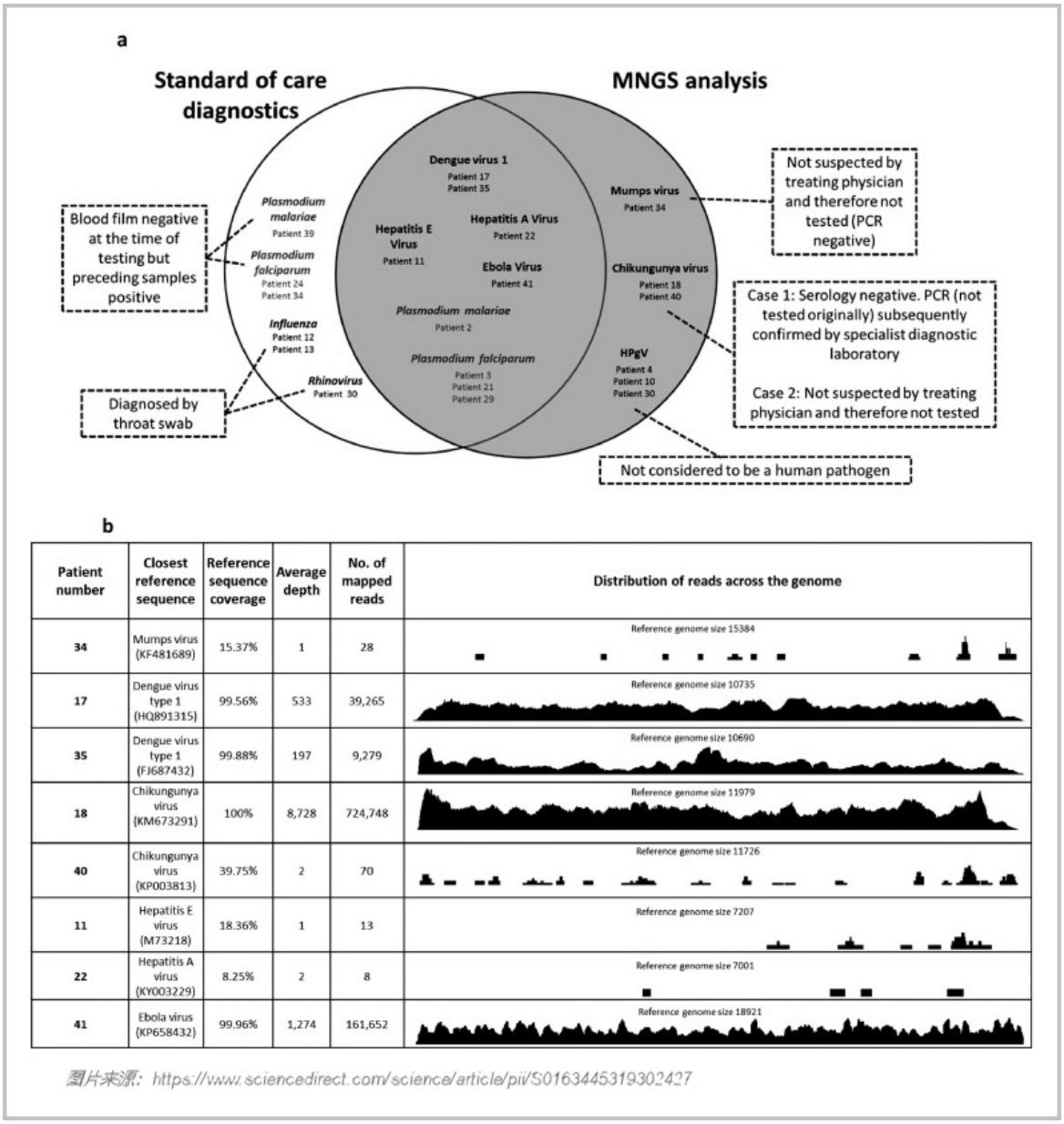


Reads

total:	1,575	
filtered:	0	(0.0%)
non-primary:	0	
duplicated:	9	(0.6%)
mapped:	1,575	(100.0%)
zero MQ:	0	(0.0%)
avg read length:	150	

Bases

total:	236,250	
mapped:	168,030	(71.1%)
error rate:	0.26%	



b

Patient number	Closest reference sequence	Reference sequence coverage	Average depth	No. of mapped reads	Distribution of reads across the genome
34	Mumps virus (KF481689)	15.37%	1	28	Reference genome size 15384
17	Dengue virus type 1 (HQ891315)	99.56%	533	39,265	Reference genome size 10735
35	Dengue virus type 1 (FJ687432)	99.88%	197	9,279	Reference genome size 10690
18	Chikungunya virus (KM673291)	100%	8,728	724,748	Reference genome size 11979
40	Chikungunya virus (KP003813)	39.75%	2	70	Reference genome size 11726
11	Hepatitis E virus (M73218)	18.36%	1	13	Reference genome size 7207
22	Hepatitis A virus (KY003229)	8.25%	2	8	Reference genome size 7001
41	Ebola virus (KP658432)	99.96%	1,274	161,652	Reference genome size 18921

图片来源: https://www.sciencedirect.com/science/article/pii/S0163445319302427

6. Repeat the preceding steps to compare COVID-19 with different samples.

Compare with 39 known BetaCov RNAs

1. Run ossutil commands to upload mNGS data to your OSS bucket.

   Sample command:

```
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. Run a comparison task as shown in the following command.

   Sample command:

```
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. Check the comparison results.

   Sample results:

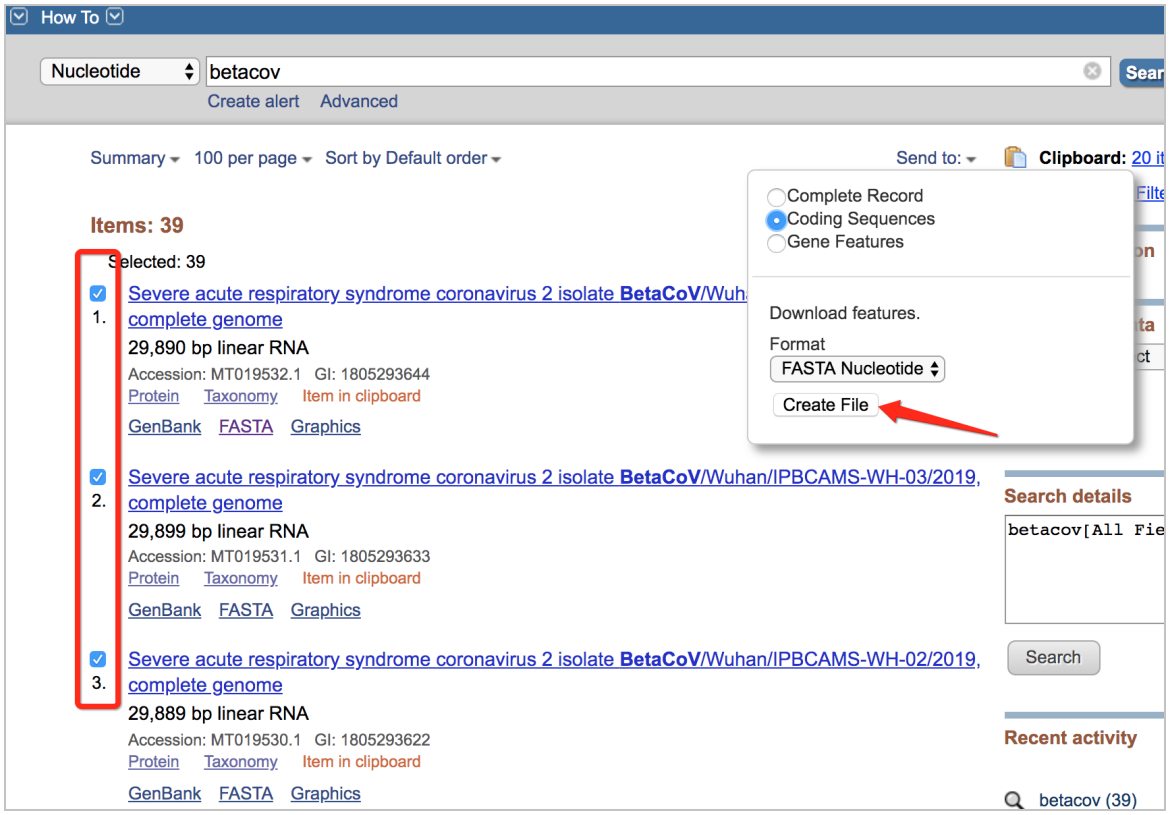
```
ags remote get rna-mapping-gpu-6mpcc --show
+----------------------+-----------------+-----------+------------------------------
+----------+---------------------------------+-------------+-------------+
|       JOB NAME       |  JOB NAMESPACE  |  STATUS   |          CREATE TIME
| DURATION |          FINISH TIME           | TOTAL READS | TOTAL BASES |
+----------------------+-----------------+-----------+------------------------------
+----------+---------------------------------+-------------+-------------+
| rna-mapping-gpu-6mpcc | XXXXXXXXX | Succeeded | 2020-03-04 17:36:21 +0800 CST | 40s
| 2020-03-04 17:37:01 +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                                         |
+---------------------------------+-----------------------------------------------+
```

Compare with a custom virus library

1. Download reference sequences from NCBI GeneBank and merge them into multiple contigs.

   Example:

Search reference sequences by keyword betacov and download the matches.



2. Rename the downloaded sequence file `sequence.fa` to `betacov-ncbi-test.fa`.
3. Upload the reference sequence file to your OSS bucket.

   Sample command:

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

4. Run a comparison task and specify the path of the reference sequence, as shown in the following command.

   Sample command:

   ```
   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. View the report and download the results.

   Sample results:

```
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      | 2020-03-04 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. Download the result data for further analysis.

   Sample command:

```
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   DF7B1A6CA5AF5DE6BF4FFDBB6DEF
71C3      oss://my-test-shenzhen/bam/ICU6G_S2_virus.bam
2020-03-04 17:47:38 +0800 CST         1474      Standard   9D7968A779A0DE7C1993CC2A8D0E
5A56      oss://my-test-shenzhen/bam/ICU6G_S2_virus.bam.cov.txt
2020-03-04 17:47:38 +0800 CST          397      Standard   81170E30BAAFEB947A2238E01517
1A51      oss://my-test-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
```

2.4. AGS API reference

2.4.1. DescribeWorkflows

Queries all workflows.

Debugging

OpenAPI Explorer automatically calculates the signature value. For your convenience, we recommend that you call this operation in OpenAPI Explorer. OpenAPI Explorer dynamically generates the sample code of the operation for different SDKs.

Request syntax

```
GET /gs/workflows HTTP/1.1
Content-Type:application/json
Common request parameters
```

Request parameters

None

Response syntax

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "jobs" : [ {
    "cluster_id" : "String",
    "job_name" : "String",
    "create_time" : "String"
  } ]
}
```

Response parameters

Response body parameters

Parameter	Type	Example	Description
jobs	Array of job		The list of the jobs.
cluster_id	String	cb1a7214cfc0b41d9bb086affc2d8f51c	The ID of the ACK cluster.
job_name	String	wgs-gpu-qb4dk	The name of the workflow.
create_time	String	2020-01-15T13:18:52Z	The time when the workflow was created.

Examples

```
GET /gs/workflows HTTP/1.1
Host:cs.aliyuncs.com
Content-Type:application/json
Common request parameters
```

Sample success responses

`XML` format

```
HTTP/1.1 200 OK
Content-Type:application/xml
<jobs>
    <cluster_id>cb1a7214cfc0b41d9bb086affc2d8f51c</cluster_id>
    <job_name>wgs-gpu-qb4dk</job_name>
    <create_time>2020-01-15T13:18:52Z</create_time>
</jobs>
```

`JSON` format

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "jobs" : [ {
    "cluster_id" : "cb1a7214cfc0b41d9bb086affc2d8f51c",
    "job_name" : "wgs-gpu-qb4dk",
    "create_time" : "2020-01-15T13:18:52Z"
  } ]
}
```

Error codes

For a list of error codes, visit the API Error Center.

2.4.2. DescribeWorkflow

Queries information about a specified workflow.

Debugging

OpenAPI Explorer automatically calculates the signature value. For your convenience, we recommend that you call this operation in OpenAPI Explorer. OpenAPI Explorer dynamically generates the sample code of the operation for different SDKs.

Request syntax

```
GET /gs/workflow/workflowName HTTP/1.1
Content-Type:application/json
```

Request parameters

Request path parameters

Parameter	Type	Required	Example	Description
workflowName	String	Yes	mapping-gpu-mhhgh	The name of the workflow that you want to query.

Response syntax

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "create_time" : "String",
  "duration" : "String",
  "finish_time" : "String",
  "input_data_size" : "String",
  "job_name" : "String",
  "job_namespace" : "String",
  "output_data_size" : "String",
  "status" : "String",
  "total_bases" : "String",
  "total_reads" : "String",
  "user_input_data" : "String"
}
```

Response parameters

Response body parameters

Parameter	Type	Example	Description
create_time	String	2020-01-15 16:30:25 +0800 CST	The time when the workflow was created.
duration	String	1h15m33.529968361s	The duration of the workflow.
finish_time	String	0001-01-01 00:00:00 +0000 UTC	The time when the workflow ended.
input_data_size	String	0	The size of the input data.
job_name	String	wgs-gpu-97xfn	The name of the workflow.
job_namespace	String	1171330362041663	The namespace to which the workflow belongs.
output_data_size	String	0	The size of the output data.
status	String	Running	The current state of the workflow.
total_bases	String	0	The number of base pairs.
total_reads	String	0	The number of reads.

Parameter	Type	Example	Description
user_input_data	String	{\"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\"}	The user input parameters.

Examples

```
GET /gs/workflow/mapping-gpu-mhhgh HTTP/1.1
Host:cs.aliyuncs.com
Content-Type:application/json
```

Sample success responses

`XML` format

```
HTTP/1.1 200 OK
Content-Type:application/xml
<create_time>2020-01-15 16:30:25 +0800 CST</create_time>
<duration>1h15m33.529968361s</duration>
<finish_time>0001-01-01 00:00:00 +0000 UTC</finish_time>
<input_data_size>0</input_data_size>
<job_name>wgs-gpu-97xfn</job_name>
<job_namespace>1171330362041663</job_namespace>
<output_data_size>0</output_data_size>
<status>Running</status>
<total_bases>0</total_bases>
<total_reads>0</total_reads>
<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\"}</user_input_data>
```

`JSON` format

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "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/hua
da/MGISEQ-200019SZ0002402\\\",\\\"wgs_bucket_name\\\":\\\"gene-shenzhen\\\",\\\"wgs_vcf_fil
e_name\\\":\\\"output/vcf/huada.vcf\\\",\\\"wgs_bam_file_name\\\":\\\"output/bam/huada.bam\
\\",\\\"wgs_reference_file\\\":\\\"hg19\\\",\\\"wgs_service\\\":\\\"g\\\"}"
}
```

Error codes

For a list of error codes, visit the API Error Center.

2.4.3. StartWorkflow

Creates a workflow.

Debugging

OpenAPI Explorer automatically calculates the signature value. For your convenience, we recommend that you call this operation in OpenAPI Explorer. OpenAPI Explorer dynamically generates the sample code of the operation for different SDKs.

Request syntax

```
POST /gs/workflow HTTP/1.1
Content-Type:application/json
{
  "workflow_type" : "String",
  "service" : "String",
  "mapping_oss_region" : "String",
  "mapping_fastq_first_filename" : "String",
  "mapping_fastq_second_filename" : "String",
  "mapping_bucket_name" : "String",
  "mapping_fastq_path" : "String",
  "mapping_reference_path" : "String",
  "mapping_is_mark_dup" : "String",
  "mapping_bam_out_path" : "String",
  "mapping_bam_out_filename" : "String",
  "wgs_oss_region" : "String",
  "wgs_fastq_first_filename" : "String",
  "wgs_fastq_second_filename" : "String",
  "wgs_bucket_name" : "String",
  "wgs_fastq_path" : "String",
  "wgs_reference_path" : "String",
  "wgs_vcf_out_path" : "String",
  "wgs_vcf_out_filename" : "String"
}
```

Request parameters

Request body parameters

Parameter	Type	Required	Example	Description
workflow_type	String	Yes	mapping	The type of the workflow. Valid values: wgs and mapping.
service	String	No	s	The type of service-level agreement (SLA). Valid values: • s: the silver level (S-level). It requires 1 extra minute to process every 1.5 billion base pairs beyond the limit of 90 billion base pairs. • g: the gold level (G-level). It requires 1 extra minute to process every 2 billion base pairs beyond the limit of 90 billion base pairs. • p: the platinum level (P-level). It requires 1 extra minute to process every 3 billion base pairs beyond the limit of 90 billion base pairs.

Parameter	Type	Required	Example	Description
mapping_oss_region	String	No	cn-hangzhou	The region where the Object Storage Service (OSS) bucket that stores the data of the mapping workflow is deployed.
mapping_fastq_first_filename	String	No	MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz	The name of the first FASTQ file of the mapping workflow.
mapping_fastq_second_filename	String	No	MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz	The name of the second FASTQ file of the mapping workflow.
mapping_bucket_name	String	No	gene-shenzhen	The name of the OSS bucket that stores the data of the mapping workflow.
mapping_fastq_path	String	No	fastq/MGISEQ2000	The path of the FASTQ files of the mapping workflow.
mapping_reference_path	String	No	reference/hg19	The path of the reference files of the mapping workflow.
mapping_is_mark_dup	String	No	true	Specifies whether to mark duplicate values.
mapping_bam_out_path	String	No	output/bamDirName	The output path of the Binary Alignment Map (BAM) file.
mapping_bam_out_filename	String	No	abc.bam	The name of the output BAM file.
wgs_oss_region	String	No	cn-shenzhen	The region where the OSS bucket that stores the data of the whole genome sequencing (WGS) workflow is deployed.

Parameter	Type	Required	Example	Description
wgs_fastq_first_filename	String	No	MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz	The name of the first FASTQ file of the WGS workflow.
wgs_fastq_second_filename	String	No	MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz	The name of the second FASTQ file of the WGS workflow.
wgs_bucket_name	String	No	gene-shenzhen	The name of the OSS bucket that stores the data of the WGS workflow.
wgs_fastq_path	String	No	fastq/MGISEQ2000	The path of the FASTQ files of the WGS workflow.
wgs_reference_path	String	No	reference/hg19	The path of the reference files of the WGS workflow.
wgs_vcf_out_path	String	No	output/vcf	The output path of the Variant Call Format (VCF) file.
wgs_vcf_out_filename	String	No	abc.vcf	The name of the output VCF file.

Response syntax

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "JobName" : "String"
}
```

Response parameters

Response body parameters

Parameter	Type	Example	Description
JobName	String	mapping-gpu-66xv7	The name of the workflow that is created.

Examples

```
POST /gs/workflow HTTP/1.1
Host:cs.aliyuncs.com
Content-Type:application/json
{
  "workflow_type" : "mapping",
  "service" : "s",
  "mapping_oss_region" : "cn-hangzhou",
  "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",
  "mapping_is_mark_dup" : "true",
  "mapping_bam_out_path" : "output/bamDirName",
  "mapping_bam_out_filename" : "abc.bam",
  "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"
}
```

Description of the sample request

````
Create a WGS workflow:
```
POST /gs/workflow HTTP/1.1
Content-Type:application/json
{
  "workflow_type" : "wgs",
  "service" : "s",
  "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"
}
```
````

Sample success responses

`XML` format

```
HTTP/1.1 200 OK
Content-Type:application/xml
<JobName>mapping-gpu-66xv7</JobName>
```

`JSON` format

```
HTTP/1.1 200 OK
Content-Type:application/json
{
  "JobName" : "mapping-gpu-66xv7"
}
```

Error codes

For a list of error codes, visit the API Error Center.

2.4.4. CancelWorkflow

Cancels a running workflow.

Debugging

OpenAPI Explorer automatically calculates the signature value. For your convenience, we recommend that you call this operation in OpenAPI Explorer. OpenAPI Explorer dynamically generates the sample code of the operation for different SDKs.

Request syntax

```
PUT /gs/workflow/workflowName HTTP/1.1
Content-Type:application/json
{
  "action" : "String"
}
```

Request parameters

Request path parameters

Parameter	Type	Required	Example	Description
workflowName	String	Yes	mapping-gpu-mhhgh	The name of the workflow that you want to cancel.

Request body parameters

Parameter	Type	Required	Example	Description
action	String	Yes	cancel	The operation that you want to perform. Valid value: cancel.

Response syntax

```
HTTP/1.1 200 OK
```

Response parameters

None

Examples

```
PUT /gs/workflow/mapping-gpu-mhhgh HTTP/1.1
Host:cs.aliyuncs.com
Content-Type:application/json
{
  "action" : "cancel"
}
```

Sample success responses

`JSON` format

```
HTTP/1.1 200 OK
```

Error codes

For a list of error codes, visit the API Error Center.

2.4.5. RemoveWorkflow

Deletes a workflow.

Debugging

OpenAPI Explorer automatically calculates the signature value. For your convenience, we recommend that you call this operation in OpenAPI Explorer. OpenAPI Explorer dynamically generates the sample code of the operation for different SDKs.

Request syntax

```
DELETE /gs/workflow/workflowName HTTP/1.1
Content-Type:application/json
```

Request parameters

Request path parameters

Parameter	Type	Required	Example	Description
workflowName	String	Yes	mapping-gpu-98wt4	The name of the workflow that you want to delete.

Response syntax

```
HTTP/1.1 200 OK
```

Response parameters

None

Examples

```
DELETE /gs/workflow/mapping-gpu-98wt4 HTTP/1.1
Host:cs.aliyuncs.com
Content-Type:application/json
```

Sample success responses

`JSON` format

```
HTTP/1.1 200 OK
```

Error codes

For a list of error codes, visit the API Error Center.

3.AGS workflow

3.1. Create a workflow

Workflows developed by Alibaba Cloud are based on Argo Workflows and are used to implement container orchestration in Kubernetes. Each step in a workflow is a container. This topic describes how to create a workflow in the Container Service for Kubernetes (ACK) console or by using a CLI.

Prerequisites

- An ACK cluster is created. For more information, see Create an ACK managed cluster.
- You are connected to the cluster by using kubectl. For more information, see Connect to ACK clusters by using kubectl.
- The workflow feature is enabled for your account.

> **Note** If the workflow feature is not enabled for your account, .

Context

Workflows developed by Alibaba Cloud are based on Argo Workflows and are used to implement container orchestration in Kubernetes. Each step in a workflow is a container.

Workflows are implemented by using Kubernetes CustomResourceDefinitions(CRDs). Therefore, you can use kubectl to manage workflows and integrate them with other Kubernetes services, such as volumes, Secrets, and role-based access control (RBAC). At the backend, the workflow controller provides a complete set of workflow features, such as parameter substitution, storage, loops, and recursion.

You can create workflows in the ACK console or by using a CLI.

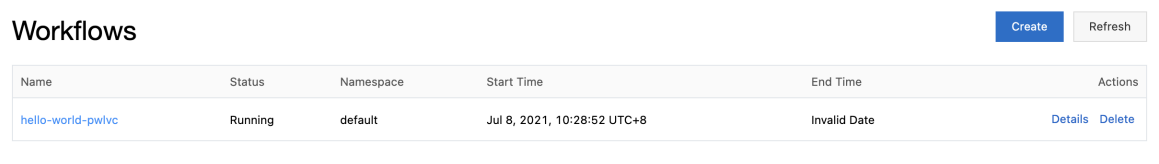
Create a workflow in the ACK console

1.
2.
3.
4. In the left-side navigation pane of the details page, choose **Applications > Workflows**.
5. On the **Workflows** page, click **Create** in the upper-right corner of the page.
6. In the **Create from Template** dialog box, configure the workflow and click **Create**.
   - **Cluster**: Select the cluster where you want to deploy the workflow. The workflow will be deployed in the selected cluster.
   - **Namespace**: Select the namespace to which the workflow belongs. By default, the `default` namespace is selected.
   - **Sample Template**: ACK provides YAML templates of various resource types to simplify the deployment of workflows. You can also create a custom template based on YAML syntax to customize resource definitions.

   The following code provides an example on how to create a Hello World workflow:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow                  # The type of Kubernetes object.
metadata:
  generateName: hello-world-    # The name of the workflow.
spec:
  entrypoint: whalesay          # Invoke the template of a whalesay image.
  templates:
  - name: whalesay              # The name of the template.
    container:
      image: docker/whalesay
      command: [cowsay]
      args: ["hello world"]
      resources:                # The resource limits.
        limits:
          memory: 32Mi
          cpu: 100m
```

7. After the workflow is created, you can view the created workflow on the **Workflows** page.



You can click **Details** in the Actions column to view information about the workflow and pods.

Create a parameters workflow by using a CLI

1. Create an *arguments-parameters.yaml* file based on the following code:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-parameters-
spec:
  # Specify "hello world" as the value of the parameter to invoke the template of a wha
lesay image.
  entrypoint: whalesay
  arguments:
    parameters:
    - name: message
      value: hello world
  templates:
  - name: whalesay
    inputs:
      parameters:
      - name: message       # The description of the parameter.
    container:
      # Use the input parameter args to run cowsay.
      image: docker/whalesay
      command: [cowsay]
      args: ["{{inputs.parameters.message}}"]
```

2. Run the following command to deploy the parameters workflow:

```
ags submit arguments-parameters.yaml -p message="goodbye world"
```

You can also use workflow templates provided in Sample workflow templates to create other types of workflow.

Ags CLI is a command-line tool developed by Alibaba Cloud. It is compatible with open source Argo. Ags CLI provides an easy way to submit, check, modify, and delete workflows. For more information, see Introduction to AGS CLI.

3.2. Sample workflow templates

This topic provides multiple sample workflow templates that can be used to create workflows.

Steps

This type of workflow template can be used to create multi-step workflows, define more than one template in a workflow specification, and create nested workflows. We recommend that you read the comments to ensure the readability of the code.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: steps-
spec:
  entrypoint: hello-hello-hello
  # This spec contains two templates: hello-hello-hello and whalesay
  templates:
  - name: hello-hello-hello
    # Instead of just running a container
    # This template has a sequence of steps
    steps:
    - - name: hello1            # hello1 is run before the following steps
        template: whalesay
        arguments:
          parameters:
          - name: message
            value: "hello1"
    - - name: hello2a           # double dash => run after previous step
        template: whalesay
        arguments:
          parameters:
          - name: message
            value: "hello2a"
      - name: hello2b           # single dash => run in parallel with previous step
        template: whalesay
        arguments:
          parameters:
          - name: message
            value: "hello2b"
  # This is the same template as from the previous example
  - name: whalesay
    inputs:
      parameters:
      - name: message
    container:
      image: docker/whalesay
      command: [cowsay]
      args: ["{{inputs.parameters.message}}"]
```

The preceding workflow prints a hello-hello-hello template that contains three distinct steps. The first step named *hello1* runs in sequence. The next two steps named *hello2a* and *hello2b* run in parallel with each other. You can run commands in the AGS command-line interface (CLI) to display the running records of this workflow specification. The following tree-structure diagram shows that the steps *hello2a* and *hello2b* are performed in parallel with each other.

The following output is returned:

```
STEP                                  PODNAME
 ✔ arguments-parameters-rbm92
 ├---✔ hello1                  steps-rbm92-2023062412
 └-·-✔ hello2a                 steps-rbm92-685171357
   └-✔ hello2b                 steps-rbm92-634838500
```

DAG

This type of workflow template can also be used to specify the sequence of steps in a workflow. You can define the workflow as a directed acyclic graph (DAG) by specifying the dependencies of each task. This method allows you to simplify complex workflows and enable a maximum number of tasks to run in parallel with each other.

The following workflow template shows that Task A has no dependencies and runs first. After Task A is completed, Tasks B and C run in parallel with each other. Then, after Tasks B and C are completed, Task D runs.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: dag-diamond-
spec:
  entrypoint: diamond
  templates:
  - name: echo
    inputs:
      parameters:
      - name: message
    container:
      image: alpine:3.7
      command: [echo, "{{inputs.parameters.message}}"]
  - name: diamond
    dag:
      tasks:
      - name: A
        template: echo
        arguments:
          parameters: [{name: message, value: A}]
      - name: B
        dependencies: [A]
        template: echo
        arguments:
          parameters: [{name: message, value: B}]
      - name: C
        dependencies: [A]
        template: echo
        arguments:
          parameters: [{name: message, value: C}]
      - name: D
        dependencies: [B, C]
        template: echo
        arguments:
          parameters: [{name: message, value: D}]
```

A dependency graph may have multiple roots. The template that is called by a DAG or Steps template can be a DAG or Steps template. This allows you to separate a complex workflow into multiple parts that are easy to manage.

Secrets

This type of workflow template supports the same Secret syntax and mechanisms as the Kubernetes pod specification. You can access a Secret that serves as an environment variable or volume by using this template.

```
# To run this example, first create the secret by running:
# kubectl create secret generic my-secret --from-literal=mypassword=S00perS3cretPa55word
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: secret-example-
spec:
  entrypoint: whalesay
  # To access secrets as files, add a volume entry in spec.volumes[] and
  # then in the container template spec, add a mount using volumeMounts.
  volumes:
  - name: my-secret-vol
    secret:
      secretName: my-secret     # name of an existing k8s secret
  templates:
  - name: whalesay
    container:
      image: alpine:3.7
      command: [sh, -c]
      args: ['
        echo "secret from env: $MYSECRETPASSWORD";
        echo "secret from file: `cat /secret/mountpath/mypassword`"
      ']
      # To access secrets as environment variables, use the k8s valueFrom and
      # secretKeyRef constructs.
      env:
      - name: MYSECRETPASSWORD  # name of env var
        valueFrom:
          secretKeyRef:
            name: my-secret     # name of an existing k8s secret
            key: mypassword     # 'key' subcomponent of the secret
      volumeMounts:
      - name: my-secret-vol     # mount file containing secret at /secret/mountpath
        mountPath: "/secret/mountpath"
```

Scripts & Results

In most cases, a template is required to run a script specified in the workflow specification. The following example shows how to use a template to run the specified script:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: scripts-bash-
spec:
  entrypoint: bash-script-example
  templates:
  - name: bash-script-example
    steps:
    - - name: generate
        template: gen-random-int-bash
    - - name: print
        template: print-message
        arguments:
          parameters:
          - name: message
            value: "{{steps.generate.outputs.result}}"  # The result of the here-script
  - name: gen-random-int-bash
    script:
      image: debian:9.4
      command: [bash]
      source: |                                           # Contents of the here-script
        cat /dev/urandom | od -N2 -An -i | awk -v f=1 -v r=100 '{printf "%i\n", f + r * $1
/ 65536}'
  - name: gen-random-int-python
    script:
      image: python:alpine3.6
      command: [python]
      source: |
        import random
        i = random.randint(1, 100)
        print(i)
  - name: gen-random-int-javascript
    script:
      image: node:9.1-alpine
      command: [node]
      source: |
        var rand = Math.floor(Math.random() * 100);
        console.log(rand);
  - name: print-message
    inputs:
      parameters:
      - name: message
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo result was: {{inputs.parameters.message}}"]
```

The script keyword allows you to specify the script body by using the source tag. This allows you to create a temporary file that contains the script body. Then, you can specify the name of the temporary file as the final parameter in the command. The command must be the interpreter that runs the script body.

The script feature can also be used to assign the standard output of the script to a special output parameter named result. This allows you to use the result of running the script in the rest of the workflow specification. In the preceding example, the result is echoed by the print-message template.

Output Parameters

This type of workflow template allows you to use the result of a step as a parameter, rather than as an artifact. In addition to the results of scripts, you can also use the results from a type of step. These results apply to condition tests, loops, and arguments. Output parameters are used in a similar way as script results. However, the values of output parameters are set to the content of a generated file rather than the content of stdout.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: output-parameter-
spec:
  entrypoint: output-parameter
  templates:
  - name: output-parameter
    steps:
    - - name: generate-parameter
        template: whalesay
    - - name: consume-parameter
        template: print-message
        arguments:
          parameters:
          # Pass the hello-param output from the generate-parameter step as the message inp
ut to print-message
          - name: message
            value: "{{steps.generate-parameter.outputs.parameters.hello-param}}"
  - name: whalesay
    container:
      image: docker/whalesay:latest
      command: [sh, -c]
      args: ["echo -n hello world > /tmp/hello_world.txt"]  # generate the content of hello
_world.txt
    outputs:
      parameters:
      - name: hello-param   # name of output parameter
        valueFrom:
          path: /tmp/hello_world.txt  # set the value of hello-param to the contents of thi
s hello-world.txt
  - name: print-message
    inputs:
      parameters:
      - name: message
    container:
      image: docker/whalesay:latest
      command: [cowsay]
      args: ["{{inputs.parameters.message}}"]
```

However, DAG template uses a task prefix to specify another task. For example, `{{tasks.generate-parameter.outputs.parameters.hello-param}}` can be specified.

Loops

This type of workflow template allows you to configure a loop workflow.

- The following template is used to iterate over a set of inputs:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: loops-
spec:
  entrypoint: loop-example
  templates:
  - name: loop-example
    steps:
    - - name: print-message
        template: whalesay
        arguments:
          parameters:
          - name: message
            value: "{{item}}"
        withItems:              # invoke whalesay once for each item in parallel
        - hello world           # item 1
        - goodbye world         # item 2
  - name: whalesay
    inputs:
      parameters:
      - name: message
    container:
      image: docker/whalesay:latest
      command: [cowsay]
      args: ["{{inputs.parameters.message}}"]
```

- The following template is used to iterate over sets of items:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: loops-maps-
spec:
  entrypoint: loop-map-example
  templates:
  - name: loop-map-example
    steps:
    - - name: test-linux
        template: cat-os-release
        arguments:
          parameters:
          - name: image
            value: "{{item.image}}"
          - name: tag
            value: "{{item.tag}}"
        withItems:
        - { image: 'debian', tag: '9.1' }       #item set 1
        - { image: 'debian', tag: '8.9' }       #item set 2
        - { image: 'alpine', tag: '3.6' }       #item set 3
        - { image: 'ubuntu', tag: '17.10' }     #item set 4
  - name: cat-os-release
    inputs:
      parameters:
      - name: image
      - name: tag
    container:
      image: "{{inputs.parameters.image}}:{{inputs.parameters.tag}}"
      command: [cat]
      args: [/etc/os-release]
```

- The following template is used to deliver lists of items as parameters:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: loops-param-arg-
spec:
  entrypoint: loop-param-arg-example
  arguments:
    parameters:
    - name: os-list                                    # a list of items
      value: |
        [
          { "image": "debian", "tag": "9.1" },
          { "image": "debian", "tag": "8.9" },
          { "image": "alpine", "tag": "3.6" },
          { "image": "ubuntu", "tag": "17.10" }
        ]
  templates:
  - name: loop-param-arg-example
    inputs:
      parameters:
      - name: os-list
    steps:
    - - name: test-linux
        template: cat-os-release
        arguments:
          parameters:
          - name: image
            value: "{{item.image}}"
          - name: tag
            value: "{{item.tag}}"
        withParam: "{{inputs.parameters.os-list}}"      # parameter specifies the list to
iterate over
  # This template is the same as in the previous example
  - name: cat-os-release
    inputs:
      parameters:
      - name: image
      - name: tag
    container:
      image: "{{inputs.parameters.image}}:{{inputs.parameters.tag}}"
      command: [cat]
      args: [/etc/os-release]
```

- The following template is used to dynamically generate the list of items to be iterated over:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: loops-param-result-
spec:
  entrypoint: loop-param-result-example
  templates:
  - name: loop-param-result-example
    steps:
    - - name: generate
        template: gen-number-list
    # Iterate over the list of numbers generated by the generate step above
    - - name: sleep
        template: sleep-n-sec
        arguments:
          parameters:
          - name: seconds
            value: "{{item}}"
        withParam: "{{steps.generate.outputs.result}}"
  # Generate a list of numbers in JSON format
  - name: gen-number-list
    script:
      image: python:alpine3.6
      command: [python]
      source: |
        import json
        import sys
        json.dump([i for i in range(20, 31)], sys.stdout)
  - name: sleep-n-sec
    inputs:
      parameters:
      - name: seconds
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo sleeping for {{inputs.parameters.seconds}} seconds; sleep {{inputs.par
ameters.seconds}}; echo done"]
```

Conditionals

This type of workflow template supports conditional execution. The following example shows a sample template named coinflip.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: coinflip-
spec:
  entrypoint: coinflip
  templates:
  - name: coinflip
    steps:
    # flip a coin
    - - name: flip-coin
        template: flip-coin
    # evaluate the result in parallel
    - - name: heads
        template: heads                 # call heads template if "heads"
        when: "{{steps.flip-coin.outputs.result}} == heads"
      - name: tails
        template: tails                 # call tails template if "tails"
        when: "{{steps.flip-coin.outputs.result}} == tails"
  # Return heads or tails based on a random number
  - name: flip-coin
    script:
      image: python:alpine3.6
      command: [python]
      source: |
        import random
        result = "heads" if random.randint(0,1) == 0 else "tails"
        print(result)
  - name: heads
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was heads\""]
  - name: tails
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was tails\""]
```

Recursion

Workflow templates can recursively invoke each other. The following template is a variation of the preceding coinflip template. In this variation template, the coin is flipped until it lands on heads.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: coinflip-recursive-
spec:
  entrypoint: coinflip
  templates:
  - name: coinflip
    steps:
    # flip a coin
    - - name: flip-coin
        template: flip-coin
    # evaluate the result in parallel
    - - name: heads
        template: heads                 # call heads template if "heads"
        when: "{{steps.flip-coin.outputs.result}} == heads"
      - name: tails                     # keep flipping coins if "tails"
        template: coinflip
        when: "{{steps.flip-coin.outputs.result}} == tails"
  - name: flip-coin
    script:
      image: python:alpine3.6
      command: [python]
      source: |
        import random
        result = "heads" if random.randint(0,1) == 0 else "tails"
        print(result)
  - name: heads
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was heads\""]
```

The following output shows the result of running templates several times to flip the coin:

```
ags get coinflip-recursive-tzcb5
STEP                          PODNAME                                MESSAGE
 ✔ coinflip-recursive-vhph5
 ├---✔ flip-coin              coinflip-recursive-vhph5-2123890397
 └-·-✔ heads                  coinflip-recursive-vhph5-128690560
   └-○ tails
STEP                           PODNAME                                 MESSAGE
 ✔ coinflip-recursive-tzcb5
 ├---✔ flip-coin               coinflip-recursive-tzcb5-322836820
 └-·-○ heads
   └-✔ tails
     ├---✔ flip-coin           coinflip-recursive-tzcb5-1863890320
     └-·-○ heads
       └-✔ tails
         ├---✔ flip-coin       coinflip-recursive-tzcb5-1768147140
         └-·-○ heads
           └-✔ tails
             ├---✔ flip-coin   coinflip-recursive-tzcb5-4080411136
             └-·-✔ heads       coinflip-recursive-tzcb5-4080323273
               └-○ tails
```

In the first round of running the template to flip the coin, the coin lands on heads and the coin is no longer flipped. In the second round of flipping the coin, the coin lands on tails three times before it lands on heads. Then, the coin is no longer flipped.

Exit handlers

This type of workflow template is run at the end of a workflow, regardless of whether the workflow succeeds or fails.

Exit handlers apply to the following operations:

- Clean up data after a workflow is run.
- Send notifications of a workflow status by emails or Slack messages.
- Post the successful or failed state to a webhook result. For example, a GitHub build result can be posted.
- Resubmit a workflow or submit a new workflow.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: exit-handlers-
spec:
  entrypoint: intentional-fail
  onExit: exit-handler                  # invoke exit-hander template at end of the workflo
w
  templates:
  # primary workflow template
  - name: intentional-fail
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo intentional failure; exit 1"]
  # Exit handler templates
  # After the completion of the entrypoint template, the status of the
  # workflow is made available in the global variable {{workflow.status}}.
  # {{workflow.status}} will be one of: Succeeded, Failed, Error
  - name: exit-handler
    steps:
    - - name: notify
        template: send-email
      - name: celebrate
        template: celebrate
        when: "{{workflow.status}} == Succeeded"
      - name: cry
        template: cry
        when: "{{workflow.status}} ! = Succeeded"
  - name: send-email
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo send e-mail: {{workflow.name}} {{workflow.status}}"]
  - name: celebrate
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo hooray!"]
  - name: cry
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo boohoo!"]
```

Timeouts

This type of workflow template can be used to limit the timeout of a workflow. In such template, you can set the variable *activeDeadlineSeconds* to the required timeout value.

```
# To enforce a timeout for a container template, specify a value for activeDeadlineSeconds.
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: timeouts-
spec:
  entrypoint: sleep
  templates:
  - name: sleep
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo sleeping for 1m; sleep 60; echo done"]
    activeDeadlineSeconds: 10           # terminate container template after 10 seconds
```

Volumes

This type of workflow template is used to manage volumes. In the following example, a volume is dynamically created, and then used in a two-step workflow:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: volumes-pvc-
spec:
  entrypoint: volumes-pvc-example
  volumeClaimTemplates:                 # define volume, same syntax as k8s Pod spec
  - metadata:
      name: workdir                     # name of volume claim
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi                  # Gi => 1024 * 1024 * 1024
  templates:
  - name: volumes-pvc-example
    steps:
    - - name: generate
        template: whalesay
    - - name: print
        template: print-message
  - name: whalesay
    container:
      image: docker/whalesay:latest
      command: [sh, -c]
      args: ["echo generating message in volume; cowsay hello world | tee /mnt/vol/hello_wo
rld.txt"]
      # Mount workdir volume at /mnt/vol before invoking docker/whalesay
      volumeMounts:                     # same syntax as k8s Pod spec
      - name: workdir
        mountPath: /mnt/vol
  - name: print-message
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo getting message from volume; find /mnt/vol; cat /mnt/vol/hello_world.txt
"]
      # Mount workdir volume at /mnt/vol before invoking docker/whalesay
      volumeMounts:                     # same syntax as k8s Pod spec
      - name: workdir
        mountPath: /mnt/vol
```

You can use volumes to move large amounts of data from one step in a workflow to another. Specific volumes can be accessed from multiple steps at the same time based on your business requirements.

If you want to access an existing volume, instead of dynamically creating or destroying a volume, you can use or modify the volume in the following example:

```
# Define Kubernetes PVC
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my-existing-volume
spec:
  accessModes: [ "ReadWriteOnce" ]
  resources:
    requests:
      storage: 1Gi
---
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: volumes-existing-
spec:
  entrypoint: volumes-existing-example
  volumes:
  # Pass my-existing-volume as an argument to the volumes-existing-example template
  # Same syntax as k8s Pod spec
  - name: workdir
    persistentVolumeClaim:
      claimName: my-existing-volume
  templates:
  - name: volumes-existing-example
    steps:
    - - name: generate
        template: whalesay
    - - name: print
        template: print-message
  - name: whalesay
    container:
      image: docker/whalesay:latest
      command: [sh, -c]
      args: ["echo generating message in volume; cowsay hello world | tee /mnt/vol/hello_wo
rld.txt"]
      volumeMounts:
      - name: workdir
        mountPath: /mnt/vol
  - name: print-message
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo getting message from volume; find /mnt/vol; cat /mnt/vol/hello_world.txt
"]
      volumeMounts:
      - name: workdir
        mountPath: /mnt/vol
```

Daemon Containers

Workflows can start containers (also known as daemon containers) that run in the backend when the workflows continue to run. The daemons are automatically destroyed when the workflow exits the template scope in which the daemon is invoked. Daemon containers can be used to start the services to be tested, or in a fixture test or other tests. Daemon containers can also be used to run large simulations to set a database as a daemon. You can use the daemon to collect and organize the results. Compared with sidecars, daemons can run over multiple steps or even the entire workflow.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: daemon-step-
spec:
  entrypoint: daemon-example
  templates:
  - name: daemon-example
    steps:
    - - name: influx
        template: influxdb              # start an influxdb as a daemon (see the influxdb t
emplate spec below)
    - - name: init-database             # initialize influxdb
        template: influxdb-client
        arguments:
          parameters:
          - name: cmd
            value: curl -XPOST 'http://{{steps.influx.ip}}:8086/query' --data-urlencode "q=
CREATE DATABASE mydb"
    - - name: producer-1                # add entries to influxdb
        template: influxdb-client
        arguments:
          parameters:
          - name: cmd
            value: for i in $(seq 1 20); do curl -XPOST 'http://{{steps.influx.ip}}:8086/wr
ite?db=mydb' -d "cpu,host=server01,region=uswest load=$i" ; sleep .5 ; done
      - name: producer-2                # add entries to influxdb
        template: influxdb-client
        arguments:
          parameters:
          - name: cmd
            value: for i in $(seq 1 20); do curl -XPOST 'http://{{steps.influx.ip}}:8086/wr
ite?db=mydb' -d "cpu,host=server02,region=uswest load=$((RANDOM % 100))" ; sleep .5 ; done
      - name: producer-3                # add entries to influxdb
        template: influxdb-client
        arguments:
          parameters:
          - name: cmd
            value: curl -XPOST 'http://{{steps.influx.ip}}:8086/write?db=mydb' -d 'cpu,host
=server03,region=useast load=15.4'
    - - name: consumer                  # consume intries from influxdb
        template: influxdb-client
        arguments:
          parameters:
          - name: cmd
            value: curl --silent -G http://{{steps.influx.ip}}:8086/query?pretty=true --dat
a-urlencode "db=mydb" --data-urlencode "q=SELECT * FROM cpu"
```

```
  - name: influxdb
    daemon: true                        # start influxdb as a daemon
    container:
      image: influxdb:1.2
      restartPolicy: Always             # restart container if it fails
      readinessProbe:                   # wait for readinessProbe to succeed
        httpGet:
          path: /ping
          port: 8086
  - name: influxdb-client
    inputs:
      parameters:
      - name: cmd
    container:
      image: appropriate/curl:latest
      command: ["/bin/sh", "-c"]
      args: ["{{inputs.parameters.cmd}}"]
      resources:
        requests:
          memory: 32Mi
          cpu: 100m
```

A DAG template uses a task prefix to specify another task. For example, `{{tasks.influx.ip}}` can be specified.

Sidecars

A sidecar is a container that runs in the same pod where the main container is executed. Sidecars allow you to create a pod that contains multiple containers.

In the following workflow template of the sidecar type, a sidecar container is created to run NGINX as a simple web server. The sidecar container may be ready at any time in a workflow. Therefore, the main container checks the health status of the sidecar container in cycles. When the main container detects that the sidecar container is in the ready state, the main container starts to process requests for the application. We recommend that you use this mechanism for multi-container systems. Before Container Service for Kubernetes (ACK) can run your application, all required services must be ready.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: sidecar-nginx-
spec:
  entrypoint: sidecar-nginx-example
  templates:
  - name: sidecar-nginx-example
    container:
      image: appropriate/curl
      command: [sh, -c]
      # Try to read from nginx web server until it comes up
      args: ["until `curl -G 'http://127.0.0.1/' >& /tmp/out`; do echo sleep && sleep 1; do
ne && cat /tmp/out"]
    # Create a simple nginx web server
    sidecars:
    - name: nginx
      image: nginx:1.13
```

Kubernetes Resources

This type of workflow template allows you to manage ACK resources. You can use this type of workflow template to create, delete, or update ACK resources.

```
# in a workflow. The resource template type accepts any k8s manifest
# (including CRDs) and can perform any kubectl action against it (e.g. create,
# apply, delete, patch).
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: k8s-jobs-
spec:
  entrypoint: pi-tmpl
  templates:
  - name: pi-tmpl
    resource:                   # indicates that this is a resource template
      action: create            # can be any kubectl action (e.g. create, delete, apply, pa
tch)
      # The successCondition and failureCondition are optional expressions.
      # If failureCondition is true, the step is considered failed.
      # If successCondition is true, the step is considered successful.
      # They use kubernetes label selection syntax and can be applied against any field
      # of the resource (not just labels). Multiple AND conditions can be represented by co
mma
      # delimited expressions.
      # For more details: https://kubernetes.io/docs/concepts/overview/working-with-objects
/labels/
      successCondition: status.succeeded > 0
      failureCondition: status.failed > 3
      manifest: |               #put your kubernetes spec here
        apiVersion: batch/v1
        kind: Job
        metadata:
          generateName: pi-job-
        spec:
          template:
            metadata:
              name: pi
            spec:
              containers:
              - name: pi
                image: perl
                command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
              restartPolicy: Never
          backoffLimit: 4
```

Resources that are created based on this type of workflow template are independent of the workflow. If you want to delete a resource in a workflow when you delete the workflow, you can use the Kubernetes garbage collection feature and the workflow resources as an owner reference.

Note

When you patch the Kubernetes resources, the resources gain the mergeStrategy attribute. The attribute can be set to *strategy*, *merge*, or *json*. By default, strategy is used. The strategy value cannot be used to patch custom resources. You must use one of the other two mergeStrategy values. The following example shows the Custom Resource Definition that defines the CronTab:

```
apiVersion: "stable.example.com/v1"
kind: CronTab
spec:
  cronSpec: "* * * * */5"
  image: my-awesome-cron-image
```

You can modify the preceding CronTab by using the following workflow:

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: k8s-patch-
spec:
  entrypoint: cront-tmpl
  templates:
  - name: cront-tmpl
    resource:
      action: patch
      mergeStrategy: merge                # Must be one of [strategic merge json]
      manifest: |
        apiVersion: "stable.example.com/v1"
        kind: CronTab
        spec:
          cronSpec: "* * * * */10"
          image: my-awesome-cron-image
```

References

- For more information about workflow resources, see Argo workflow templates by example.
- For more information about the sample templates, see Sample templates.

3.3. Enable the web-based workflow UI

This topic describes how to enable and access the web-based workflow UI by creating an Ingress. You can view the status of all workflows and the container logs for each step of a workflow by using the web-based workflow UI.

Prerequisites

- A Container Service for Kubernetes (ACK) cluster is created. For more information, see Create an ACK managed cluster.
- A master node is connected. For more information, see Connect to ACK clusters by using kubectl.

Procedure

1. Run the htpasswd command to generate an *auth* file. You can store the username and password in the file.

   Run the following command:

   ```
   htpasswd -c auth workflow
   ```

   The following output is returned:

   ```
   New password: <workflow>
   New password:
   Re-type new password:
   Adding password for user workflow
   ```

2. Run the following command to create a Secret and store the encrypted file in the ACK cluster:

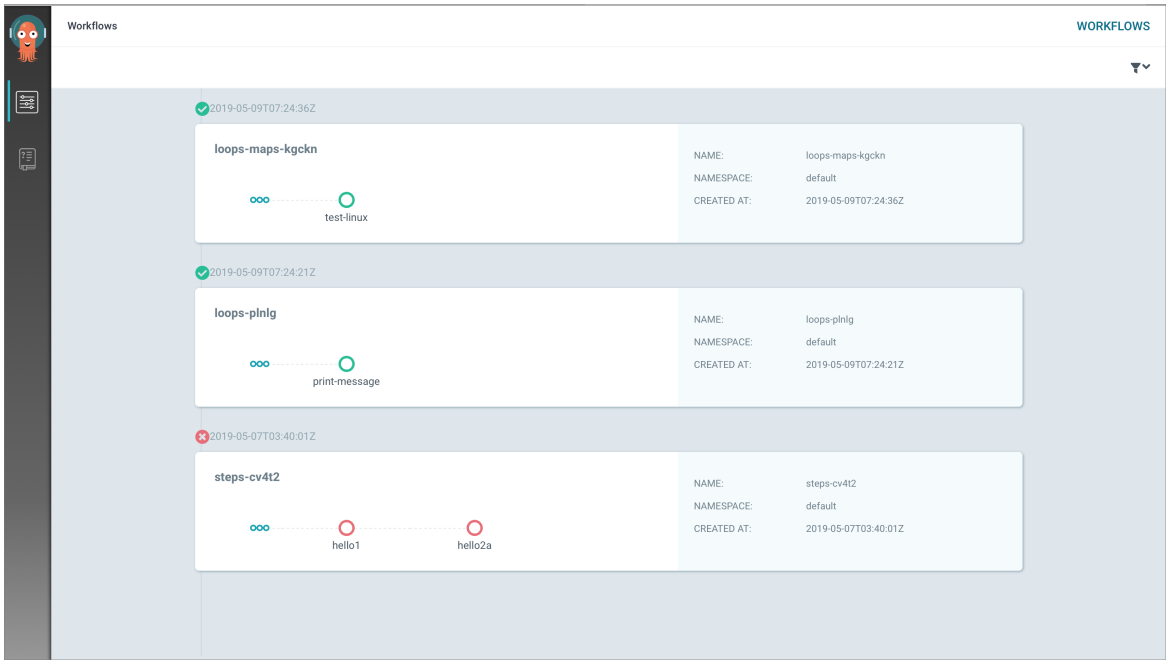
   ```
   kubectl create secret generic workflow-basic-auth --from-file=auth -n argo
   ```

3. Create an *ingress yaml* file, copy the following content to the file, and then run the `kubectl apply -f ingress.yaml` command to create the Ingress named workflow-ingress.

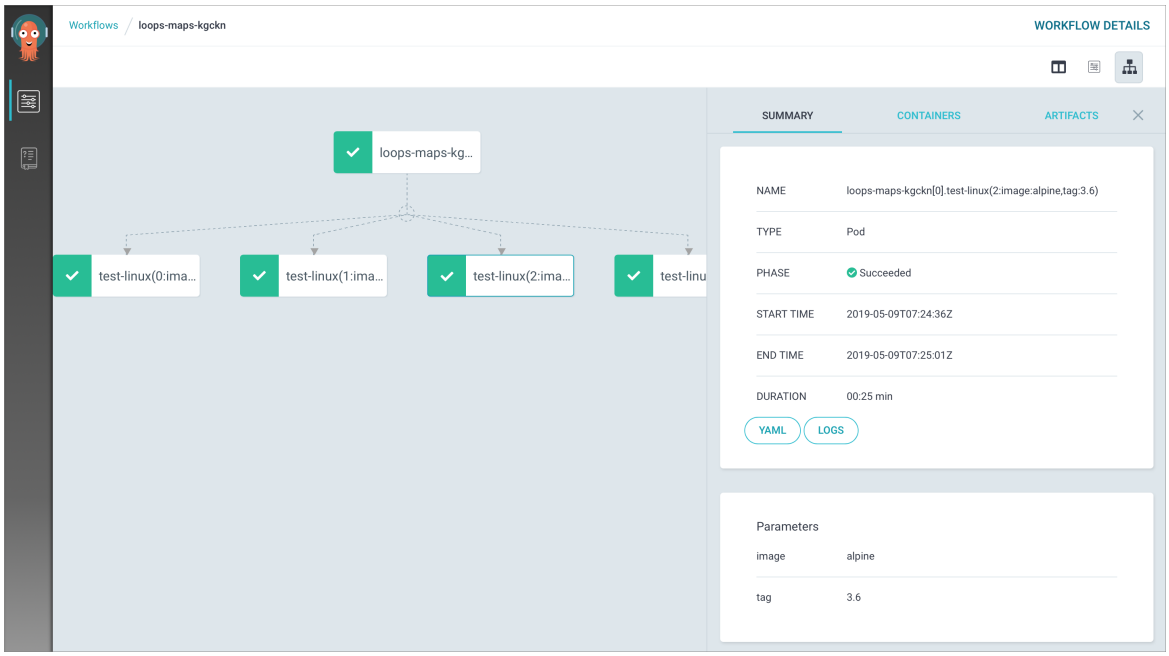
   ```
   apiVersion: networking.k8s.io/v1
   kind: Ingress
   metadata:
     name: workflow-ingress
     namespace: argo
     annotations:
       # type of authentication
       nginx.ingress.kubernetes.io/auth-type: basic
       # name of the secret that contains the user/password definitions
       nginx.ingress.kubernetes.io/auth-secret: workflow-basic-auth
       # message to display with an appropriate context why the authentication is required
       nginx.ingress.kubernetes.io/auth-realm: 'Authentication Required - workflow'
   spec:
     rules:
     - host: workflow.<yourTestHost>
       http:
         paths:
         - path: /
           backend:
             service:
               name: argo-ui
               port:
                 number: 80
           pathType: ImplementationSpecific
   ```

   Note In the preceding code block, the value of `host` must be replaced with the value of Testing Domain in the **Cluster Information** section of the ACK cluster. For example, the value can be set to workflow.cfb131.cn-zhangjiakou.alicontainer.com.

4. Open your browser, enter *workflow.<yourTestHost>* into the address bar, and then provide the required password to view the web-based workflow UI.



You can view the status of the workflow based on your business requirements.



3.4. Introduction to AGS CLI

Alibaba Cloud Genomics Compute Service (AGS) CLI is a generic command-line interface (CLI) for genomic compute services of Alibaba Cloud. AGS CLI integrates the features of Argo and provides support for commands of other Alibaba Cloud service add-ons.

Download and install AGS CLI

By default, AGS CLI uses Alibaba Cloud AccessKey pairs to access other Alibaba Cloud services. AGS CLI uses Log Service to collect pod logs. To use Log Service in a cluster of Container Service for Kubernetes (ACK), you must enable Log Service when you create the ACK cluster.

Run the following command to download AGS CLI and make the downloaded file executable.

```
wget http://ags-hub.oss-cn-hangzhou.aliyuncs.com/ags-linux && chmod +x ags-linux && mv ags-linux  /usr/local/bin/ags
```

> **Note**
>
> You can run the `ags config init` to enter the required information in AGS CLI. By default, after AGS CLI is initialized, the configuration files are stored in the *~/.ags/config* directory. You can run the `ags config show` command to display the configuration. In the configuration, AccessKeySecret is encrypted.
>
> If you want to use Log Service to collect logs, you must configure ags config. For security purposes, we recommend that you create a separate AccessKey pair for AGS CLI and grant the permissions to access Log Service.

If you use a managed Kubernetes cluster, use kubectl to connect to the cluster and run the following command on Cloud Shell to use AGS CLI. For more information, see Connect to ACK clusters by using kubectl.

```
wget http://ags-hub.oss-cn-hangzhou.aliyuncs.com/ags-linux && chmod +x ags-linux && mv ags-linux  /usr/local/bin/ags
```

Features of AGS CLI

- Fully compatible with Argo commands.
- Allows you to query pod logs in Log Service after pods are deleted.
- Allows you to use kubectl commands to manage clusters.
- Supports the install and uninstall commands. You can quickly install and uninstall tools that are required by AGS CLI.
- Allows you to run the get workflow command to query the resource usage of a workflow.
- Supports special characters such as underscores (_) in YAML templates.
- Allows you to use the securityContext feature for security settings.
- Synchronizes the pending and fails states between pods and workflows.
- Allows you to enable automatic retry by using YAML files.
- Allows you to retry a workflow from a failed step.
- Supports the Elastic Container Instance (ECI) serverless Kubernetes architecture.

AGS CLI overview

```
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags
ags is the command line interface to Alibaba Cloud Genomics Compute Service
Usage:
 ags [flags]
 ags [command]
Available Commands:
 completion  output shell completion code for the specified shell (bash or zsh)
 config      setup ags client necessary info
 delete      delete a workflow and its associated pods
 get         display details about a workflow
 help        Help about any command
 install     install ags
 kubectl     kubectl command
 lint        validate a file or directory of workflow manifests
 list        list workflows
 logs        view logs of a workflow
 resubmit    resubmit a workflow
 resume      resume a workflow
 retry       retry a workflow
 submit      submit a workflow
 suspend     suspend a workflow
 terminate   terminate a workflow
 uninstall   uninstall ags
 version     Print version information
 wait        waits for a workflow to complete
 watch       watch a workflow until it completes
Flags:
     --as string                      Username to impersonate for the operation
     --as-group stringArray           Group to impersonate for the operation, this flag can
be repeated to specify multiple groups.
     --certificate-authority string   Path to a cert file for the certificate authority
     --client-certificate string      Path to a client certificate file for TLS
     --client-key string              Path to a client key file for TLS
     --cluster string                 The name of the kubeconfig cluster to use
     --context string                 The name of the kubeconfig context to use
 -h, --help                           help for ags
     --insecure-skip-tls-verify       If true, the server's certificate will not be checked
for validity. This will make your HTTPS connections insecure
     --kubeconfig string              Path to a kube config. Only required if out-of-cluste
r
 -n, --namespace string               If present, the namespace scope for this CLI request
     --password string                Password for basic authentication to the API server
     --request-timeout string         The length of time to wait before giving up on a sing
le server request. Non-zero values should contain a corresponding time unit (e.g. 1s, 2m, 3
h). A value of zero means don't timeout requests. (default "0")
     --server string                  The address and port of the Kubernetes API server
     --token string                   Bearer token for authentication to the API server
     --user string                    The name of the kubeconfig user to use
     --username string                Username for basic authentication to the API server
Use "ags [command] --help" for more information about a command.
```

Install/Uninstall

You can run the **ags install** command to install tools that are required by AGS CLI. You do not need to manually install these tools.

You can run the **ags uninstall** command to uninstall all tools that are installed in AGS CLI.

3.5. AGS examples

This topic provides examples about how to use Alibaba Cloud Genomics Service (AGS).

Prerequisites

- A Container Service for Kubernetes (ACK) cluster is created. For more information, see Create an ACK managed cluster.
- You are connected to the cluster. For more information, see Connect to ACK clusters by using kubectl.

Log

1. Run the `ags config sls` command to install and configure Log Service in AGS.

   Native Argo can retrieve the log data of a pod only from the node where the pod is deployed. If the pod or the node is deleted, the log data is also deleted. This brings challenges to troubleshooting. After log data is persisted to Log Service, AGS can retrieve the log data from Log Service even if the node is deleted.

2. Run the `ags logs` command to query the log data of a workflow.

   You can run the `ags logs POD/WORKFLOW` command to query the log data of a pod or a workflow.

```
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags logs
view logs of a workflow
Usage:
 ags logs POD/WORKFLOW [flags]
Flags:
 -c, --container string    Print the logs of this container (default "main")
 -f, --follow              Specify if the logs should be streamed.
 -h, --help                help for logs
 -l, --recent-line int     how many lines to show in one call (default 100)
     --since string        Only return logs newer than a relative duration like 5s, 2m,
or 3h. Defaults to all logs. Only one of since-time / since may be used.
     --since-time string   Only return logs after a specific date (RFC3339). Defaults t
o all logs. Only one of since-time / since may be used.
     --tail int            Lines of recent log file to display. Defaults to -1 with no
selector, showing all log lines otherwise 10, if a selector is provided. (default -1)
     --timestamps          Include timestamps on each line in the log output
 -w, --workflow            Specify that whole workflow logs should be printed
```

> **Note**
>
> - If the node where the pod is deployed exists, AGS retrieves the log data of the pod from the node. All flags are compatible with the native Argo command.
> - If the pod is deleted, AGS retrieves the log data from Log Service. By default, the latest 100 log entries are returned. You can use **-l flag** to specify the number of log entries to return.

List

You can set the --limit parameter to specify the number of workflow entries that you want to query.

```
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags remote list --limit 8
+----------------------+-------------------------------+------------+
|       JOB NAME       |          CREATE TIME          | JOB STATUS |
+----------------------+-------------------------------+------------+
| merge-6qk46          | 2020-09-02 16:52:34 +0000 UTC | Pending    |
| rna-mapping-gpu-ck4cl | 2020-09-02 14:47:57 +0000 UTC | Succeeded  |
| wgs-gpu-n5f5s        | 2020-09-02 13:14:14 +0000 UTC | Running    |
| merge-5zjhv          | 2020-09-02 12:03:11 +0000 UTC | Succeeded  |
| merge-jjcw4          | 2020-09-02 10:44:51 +0000 UTC | Succeeded  |
| wgs-gpu-nvxr2        | 2020-09-01 22:18:44 +0000 UTC | Succeeded  |
| merge-4vg42          | 2020-09-01 20:52:13 +0000 UTC | Succeeded  |
| rna-mapping-gpu-2ss6n | 2020-09-01 20:34:45 +0000 UTC | Succeeded  |
```

Run kubectl commands

You can run the following command to query the state of a pod. AGS also allows you to run other kubectl commands.

```
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags get test-v2
Name:                test-v2
Namespace:           default
ServiceAccount:      default
Status:              Running
Created:             Thu Nov 22 11:06:52 +0800 (2 minutes ago)
Started:             Thu Nov 22 11:06:52 +0800 (2 minutes ago)
Duration:            2 minutes 46 seconds
STEP           PODNAME             DURATION  MESSAGE
● test-v2
└---● bcl2fq  test-v2-2716811808  2m
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags kubectl describe pod test-v2-2716811808
Name:               test-v2-2716811808
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               cn-shenzhen.i-wz9gwobtqrbjgfnqxl1k/192.168.0.94
Start Time:         Thu, 22 Nov 2018 11:06:52 +0800
Labels:             workflows.argoproj.io/completed=false
                    workflows.argoproj.io/workflow=test-v2
Annotations:        workflows.argoproj.io/node-name=test-v2[0].bcl2fq
                    workflows.argoproj.io/template={"name":"bcl2fq","inputs":{},"outputs":{}
,"metadata":{},"container":{"name":"main","image":"registry.cn-hangzhou.aliyuncs.com/dahu/c
url-jp:1.2","command":["sh","-c"],"ar...
Status:             Running
IP:                 172.16. *. ***
Controlled By:      Workflow/test-v2
```

After you run the ags kubectl command, the state of describe pod is returned. AGS supports all native kubectl commands.

Run ossutil commands

After AGS is initialized, you can run the following commands to upload and query files:

```
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags oss cp test.fq.gz oss://my-test-shenzhen/fasq/
Succeed: Total num: 1, size: 690. OK num: 1(upload 1 files).
average speed 3000(byte/s)
0.210685(s) elapsed
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags oss ls oss://my-test-shenzhen/fasq/
LastModifiedTime                   Size(B)  StorageClass   ETAG
ObjectName
2020-09-02 17:20:34 +0800 CST          690      Standard   9FDB86F70C6211B2EAF95A9B06B14F7E
oss://my-test-shenzhen/fasq/test.fq.gz
Object Number is: 1
0.117591(s) elapsed
```

You can run the ags oss command to upload and download files. AGS supports all native ossutil commands.

View the resource usage of a workflow

1. Create the *arguments-workflow-resource.yaml* file and copy the following content into the file. Run the `ags submit arguments-workflow-resource.yaml` command to specify resource requests.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  name: test-resource
spec:
  arguments: {}
  entrypoint: test-resource-
  templates:
  - inputs: {}
    metadata: {}
    name: test-resource-
    outputs: {}
    parallelism: 1
    steps:
    - - arguments: {}
        name: bcl2fq
        template: bcl2fq
  - container:
      args:
      - id > /tmp/yyy;echo `date` > /tmp/aaa;ps -e -o comm,euid,fuid,ruid,suid,egid,fgi
d,gid,rgid,sgid,supgid
        > /tmp/ppp;ls -l /tmp/aaa;sleep 100;pwd
      command:
      - sh
      - -c
      image: registry.cn-hangzhou.aliyuncs.com/dahu/curl-jp:1.2
      name: main
      resources:                #don't use too much resources
        requests:
          memory: 320Mi
          cpu: 1000m
    inputs: {}
    metadata: {}
    name: bcl2fq
    outputs: {}
```

2. Run the `ags get test456 --show` command to query the resource usage of a workflow.

   The CPU usage (cores/hours) of test456 and the pod is returned in this example.

```
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags get test456 --show
Name:                test456
Namespace:           default
ServiceAccount:      default
Status:              Succeeded
Created:             Thu Nov 22 14:41:49 +0800 (2 minutes ago)
Started:             Thu Nov 22 14:41:49 +0800 (2 minutes ago)
Finished:            Thu Nov 22 14:43:30 +0800 (27 seconds ago)
Duration:            1 minute 41 seconds
Total CPU:           0.02806    (core*hour)
Total Memory:        0.00877    (GB*hour)
STEP           PODNAME             DURATION  MESSAGE  CPU(core*hour)  MEMORY(GB*hour)
✔ test456                                              0               0
└---✔ bcl2fq  test456-4221301428  1m                   0.02806         0.00877
```

Configure securityContext

1. Create the *arguments-security-context.yaml* file and copy the following content to the file. Run the `ags submit arguments-security-context.yaml` command to use Pod Security Policy (PSP) for permission control.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  name: test
spec:
  arguments: {}
  entrypoint: test-security-
  templates:
  - inputs: {}
    metadata: {}
    name: test-security-
    outputs: {}
    parallelism: 1
    steps:
    - - arguments: {}
        name: bcl2fq
        template: bcl2fq
  - container:
      args:
      - id > /tmp/yyy;echo `date` > /tmp/aaa;ps -e -o comm,euid,fuid,ruid,suid,egid,fgi
d,gid,rgid,sgid,supgid
        > /tmp/ppp;ls -l /tmp/aaa;sleep 100;pwd
      command:
      - sh
      - -c
      image: registry.cn-hangzhou.aliyuncs.com/dahu/curl-jp:1.2
      name: main
      resources:                #don't use too much resources
        requests:
          memory: 320Mi
          cpu: 1000m
    inputs: {}
    metadata: {}
    name: bcl2fq
    outputs: {}
    securityContext:
      runAsUser: 800
```

Configure automatic retry with YAML

Unexpected errors may occur for some bash commands. You can execute the command again to fix this issue. AGS provides an automatic retry mechanism based on YAML. When the system fails to run a command in a pod, AGS automatically executes the command again. You can set the maximum number of retry attempts.

1. Create the *arguments-auto-retry.yaml* file and copy the following content to the file. Run the `ags submit arguments-auto-retry.yaml` command to configure the automatic retry mechanism for a workflow.

```
# This example demonstrates the use of retries for a single container.
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: retry-container-
spec:
  entrypoint: retry-container
  templates:
  - name: retry-container
    retryStrategy:
      limit: 10
    container:
      image: python:alpine3.6
      command: ["python", -c]
      # fail with a 66% probability
      args: ["import random; import sys; exit_code = random.choice([0, 1, 1]); sys.exit
(exit_code)"]
```

Retry a workflow from a failed step

A workflow consists of a number of steps. A step may fail when you run a workflow. AGS allows you to retry the workflow from a failed step.

1. Run the `ags get test456 --show` command to find the step when the test456 workflow fails.

```
[root@iZwz92q9h36kv8posr0i6uZ ~]# ags get test456 --show
Name:                test456
Namespace:           default
ServiceAccount:      default
Status:              Succeeded
Created:             Thu Nov 22 14:41:49 +0800 (2 minutes ago)
Started:             Thu Nov 22 14:41:49 +0800 (2 minutes ago)
Finished:            Thu Nov 22 14:43:30 +0800 (27 seconds ago)
Duration:            1 minute 41 seconds
Total CPU:           0.0572   (core*hour)
Total Memory:        0.01754    (GB*hour)
STEP           PODNAME             DURATION  MESSAGE  CPU(core*hour)  MEMORY(GB*hour)
 ✔ test456                                                0               0
 └---✔ bcl2fq  test456-4221301428  1m                     0.02806         0.00877
 └---X bcl2fq  test456-4221301238  1m                    0.02806         0.00877
```

2. Run the `ags retry test456` command to retry the test456 workflow from the failed step.

Run a workflow by using ECIs

For more information about Elastic Container Instance (ECI), see Elastic Container Instance.

Install AGS before you use ECI to run a workflow. For more information, see Download and install AGS CLI.

1. Run the `kubectl get cm -n argo` command to query the name of the YAML file that corresponds to the workflow.

```
[root@iZwz9f4ofes6kbo01ipuf1Z ~]# kubectl get cm -n argo
NAME                           DATA      AGE
workflow-controller-configmap  1         4d
```

2. Run the `kubectl get cm -n argo workflow-controller-configmap -o yaml` command to open the YAML file *workflow-controller-configmap.yaml*, and copy the following content to the file. The content overwrites the existing information in the YAML file.

```
apiVersion: v1
data:
  config: |
    containerRuntimeExecutor: k8sapi
kind: ConfigMap
```

3. Run the `kubectl delete pod <podName>` command to restart the Argo controller.

> **Note** *podName* is the name of the pod on which the workflow is running.

4. Create the *arguments-workflow-eci.yaml* file and copy the following content to the file. Run the `ags submit arguments-workflow-eci.yaml` command to add the nodeSelector and tolerations labels to the container that runs in the ECI.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-
spec:
  entrypoint: whalesay
  templates:
  - name: whalesay
    container:
      image: docker/whalesay
      command: [env]
      #args: ["hello world"]
      resources:
        limits:
          memory: 32Mi
          cpu: 100m
    nodeSelector:               # add nodeSelector
      type: virtual-kubelet
    tolerations:                # add tolerations
    - key: virtual-kubelet.io/provider
      operator: Exists
    - key: alibabacloud.com
      effect: NoSchedule
```

Query the actual and peak resource usage of a workflow

The AGS workflow controller automatically queries the actual resource usage of pods per minute through metrics-server. It also calculates the total and peak resource usage of each pod.

Run the `ags get steps-jr6tw --metrics` command to query the actual and peak resource usage of a workflow.

```
➜  ags get steps-jr6tw --metrics
Name:                steps-jr6tw
Namespace:           default
ServiceAccount:      default
Status:              Succeeded
Created:             Tue Apr 16 16:52:36 +0800 (21 hours ago)
Started:             Tue Apr 16 16:52:36 +0800 (21 hours ago)
Finished:            Tue Apr 16 19:39:18 +0800 (18 hours ago)
Duration:            2 hours 46 minutes
Total CPU:           0.00275    (core*hour)
Total Memory:        0.04528    (GB*hour)
STEP            PODNAME                 DURATION  MESSAGE  CPU(core*hour)  MEMORY(GB*hour)
MaxCpu(core)  MaxMemory(GB)
 ✔ steps-jr6tw                                              0               0
0             0
 └---✔ hello1   steps-jr6tw-2987978173  2h                 0.00275         0.04528
0.000005      0.00028
```

Set workflow priorities

You can set the priority of a workflow to high, medium, or low to prioritize urgent tasks. A workflow with a higher priority can preempt the resources of a workflow with a lower priority.

- You can use the following method to set a high priority for a pod:

  Create the *arguments-high-priority-taskA.yaml* file and copy the following content to the file. Run the `ags submit arguments-high-priority-taskA.yaml` command to set a high priority for Task A.

  ```
  apiVersion: scheduling.k8s.io/v1beta1
  kind: PriorityClass
  metadata:
    name: high-priority
  value: 1000000
  globalDefault: false
  description: "This priority class should be used for XYZ service pods only."
  ```

- You can use the following method to set a medium priority for a pod:

  Create the *arguments-high-priority-taskB.yaml* file and copy the following content to the file. Run the `ags submit arguments-high-priority-taskB.yaml` command to set a medium priority for Task B.

  ```
  apiVersion: scheduling.k8s.io/v1beta1
  kind: PriorityClass
  metadata:
    name: medium-priority
  value: 100
  globalDefault: false
  description: "This priority class should be used for XYZ service pods only."
  ```

- You can use the following method to set a high priority for a workflow:

  Create the *arguments-high-priority-Workflow.yaml* file and copy the following content to the file. Run the `ags submit arguments-high-priority-Workflow.yaml` command to set a high priority for all pods of the workflow.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow                  # new type of k8s spec
metadata:
  generateName: high-proty-   # name of the workflow spec
spec:
  entrypoint: whalesay          # invoke the whalesay template
  podPriorityClassName: high-priority # workflow level priority
  templates:
  - name: whalesay              # name of the template
    container:
      image: ubuntu
      command: ["/bin/bash", "-c", "sleep 1000"]
      resources:
        requests:
          cpu: 3
```

The following example describes how to set priorities for pods in a workflow. Assume that the workflow has two pods and you set a high priority for one pod and a medium priority for the other pod. Then, the pod with the high priority can preempt the resources of the pod with the medium priority.

1. Create the *arguments-high-priority-steps.yaml* file and copy the following content to the file. Run the `ags submit arguments-high-priority-steps.yaml` command to set different priorities for pods.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: steps-
spec:
  entrypoint: hello-hello-hello
  templates:
  - name: hello-hello-hello
    steps:
    - - name: low
        template: low
    - - name: low-2
        template: low
      - name: high
        template: high
  - name: low
    container:
      image: ubuntu
      command: ["/bin/bash", "-c", "sleep 30"]
      resources:
        requests:
          cpu: 3
  - name: high
    priorityClassName: high-priority # step level priority
    container:
      image: ubuntu
      command: ["/bin/bash", "-c", "sleep 30"]
      resources:
        requests:
          cpu: 3
```

2. The following result indicates that the pod with a high priority preempts the resources of the other pod and the pod with a medium priority is deleted.

```
Name:                steps-sxvrv
Namespace:           default
ServiceAccount:      default
Status:              Failed
Message:             child 'steps-sxvrv-1724235106' failed
Created:             Wed Apr 17 15:06:16 +0800 (1 minute ago)
Started:             Wed Apr 17 15:06:16 +0800 (1 minute ago)
Finished:            Wed Apr 17 15:07:34 +0800 (now)
Duration:            1 minute 18 seconds
STEP            PODNAME                 DURATION  MESSAGE
 ✖ steps-sxvrv                                    child 'steps-sxvrv-1724235106' failed
 ├---✔ low      steps-sxvrv-3117418100  33s
 └-·-✔ high     steps-sxvrv-603461277   45s
   └-⚠ low-2    steps-sxvrv-1724235106  45s       pod deleted
```

Note A pod with a higher priority can preempt the resources of a pod with a lower priority. This stops the tasks that are running on the pod with a lower priority. Proceed with caution.

Workflow Filter

When a workflow contains a large number of pods, you can use the workflow filter in the ags get workflow command to query pods that are in a specified state.

1. Run the `ags get <Pod name> --status Running` command to query pods that are in the Running state.

```
[root@iZ8vb19036cvgu1tpxkzl1Z workflow-prioty]# ags get pod-limits-n262v --status Runni
ng
Name:                pod-limits-n262v
Namespace:           default
ServiceAccount:      default
Status:              Running
Created:             Wed Apr 17 15:59:08 +0800 (1 minute ago)
Started:             Wed Apr 17 15:59:08 +0800 (1 minute ago)
Duration:            1 minute 17 seconds
Parameters:
  limit:             300
STEP                   PODNAME                      DURATION  MESSAGE
 ● pod-limits-n262v
   ├-● run-pod(13:13)  pod-limits-n262v-3643890604  1m
   ├-● run-pod(14:14)  pod-limits-n262v-4115394302  1m
   ├-● run-pod(16:16)  pod-limits-n262v-3924248206  1m
   ├-● run-pod(17:17)  pod-limits-n262v-3426515460  1m
   ├-● run-pod(18:18)  pod-limits-n262v-824163662   1m
   ├-● run-pod(20:20)  pod-limits-n262v-4224161940  1m
   ├-● run-pod(22:22)  pod-limits-n262v-1343920348  1m
   ├-● run-pod(2:2)    pod-limits-n262v-3426502220  1m
   ├-● run-pod(32:32)  pod-limits-n262v-2723363986  1m
   ├-● run-pod(34:34)  pod-limits-n262v-2453142434  1m
   ├-● run-pod(37:37)  pod-limits-n262v-3225742176  1m
   ├-● run-pod(3:3)    pod-limits-n262v-2455811176  1m
   ├-● run-pod(40:40)  pod-limits-n262v-2302085188  1m
   ├-● run-pod(6:6)    pod-limits-n262v-1370561340  1m
```

2. Run the `ags get <Pod name> --sum-info` command to query the current states of pods.

```
[root@iZ8vb19036cvgu1tpxkzl1Z workflow-prioty]# ags get pod-limits-n262v --sum-info --s
tatus Error
Name:                pod-limits-n262v
Namespace:           default
ServiceAccount:      default
Status:              Running
Created:             Wed Apr 17 15:59:08 +0800 (2 minutes ago)
Started:             Wed Apr 17 15:59:08 +0800 (2 minutes ago)
Duration:            2 minutes 6 seconds
Pending:             198
Running:             47
Succeeded:           55
Parameters:
  limit:             300
STEP                 PODNAME  DURATION  MESSAGE
 ● pod-limits-n262v
```

Use Autoscaler in the agility edition

Before you use Autoscaler in the agility version, make sure that the following resources are available:

- A virtual private cloud (VPC).
- A VSwitch.
- A security group.
- The internal endpoint of APIServer of the agility edition.
- The specification for node scaling.
- An Elastic Compute Service (ECS) instance that can access the Internet.

Perform the following operations in the AGS command-line interface:

```
$ags config autoscaler Enter the required information based on the tip: Please input VSwitc
hes separated with commas (,).
vsw-hp3cq3fnv47bpz7x58wfe
Please input security group id
sg-hp30vp05x6tlx13my0qu
Please input the instanceTypes with comma separated
ecs.c5.xlarge
Please input the new ecs ssh password
xxxxxxxx
Please input k8s cluster APIServer address like(192.168.1.100)
172.24.61.156
Please input the autoscaling mode (current: release. Type enter to skip.)
Please input the min size of group (current: 0. Type enter to skip.)
Please input the max size of group (current: 1000. Type enter to skip.)
Create scaling group successfully.
Create scaling group config successfully.
Enable scaling group successfully.
Succeed
```

After you complete the preceding operations, log on to the Auto Scaling console to check the scaling group that you have created.

Configure and use a ConfigMap

By default, hostNetwork is used in this example.

1. Run the `kubectl get cm -n argo` command to query the name of the YAML file that corresponds to a workflow.

   ```
   [root@iZ8vb19036cvgu1tpxkzl1Z ~]# kubectl get cm -n argo
   ```

   ```
   NAME                           DATA   AGE
   workflow-controller-configmap  1      6d23h
   ```

2. Run the `kubectl edit cm workflow-controller-configmap -n argo` command to open the *workflow-controller-configmap.yaml* file and copy the following content to the file:

```
data:
  config: |
    extraConfig:
      enableHostNetwork: true
      defaultDnsPolicy: Default
```

The following shows the details about the updated *workflow-controller-configmap.yaml* file:

```
apiVersion: v1
data:
  config: |
    extraConfig:
      enableHostNetwork: true
      defaultDnsPolicy: Default
kind: ConfigMap
metadata:
  name: workflow-controller-configmap
  namespace: argo
```

3. After you complete the configuration, newly deployed workflows use *hostNetwork* by default and the value of dnsPolicy is set to *Default*.
4. (Optional)If a PSP is configured, add the following content to the YAML file of the PSP:

```
hostNetwork: true
```

> **Note** If the hostNetwork parameter can be found in the YAML file, you must change the value of the parameter to *true*.

A YAML template contains the following content:

3.6. Use AGS to speed up a workflow

You can compile a local workflow and an Alibaba Cloud Genomics Service (AGS) workflow to form a hybrid workflow. We recommend that you run time-consuming tasks that have a standard workflow and require a large number of computing resources in the AGS workflow, and run other tasks in the local workflow. Workflow data transfers through Object Storage Service (OSS). This improves operation efficiency and saves resources. This topic describes how to use AGS to create a mapping that allows you to compile and run a hybrid workflow.

Prerequisites

The permissions to use AGS are granted. AGS is in public preview. Before you use this feature, make sure that you have the permissions. To apply for permissions, visit AGS application.

> **Note** If you log on as a RAM user, enter the ID of the RAM user when you apply for the use of AGS. You can obtain the ID of the RAM user in the Account Management console.

Procedure

1. Configure AGS.

   Download and install AGS. For more information, see Introduction to AGS CLI.

```
ags config init
```

After you configure AGS, the AGS configuration file *config* is generated.

2. Create a ConfigMap.

```
kubectl create configmap config --from-file=~/.ags/config -n argo
```

3. Add the following content to the *hybridStorage.yaml* file.

You can configure the following settings based on your needs: the endpoint and path of the NAS volume, and the AccessKey ID and AccessKey secret of the OSS bucket.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gene-shenzhen-cache-nas
  namespace: argo
  labels:
    alicloud-pvname: gene-shenzhen-cache-nas
spec:
  capacity:
    storage: 20Gi
  storageClassName: alicloud-nas
  accessModes:
    - ReadWriteMany
  csi:
    driver: nasplugin.csi.alibabacloud.com
    volumeHandle: gene-shenzhen-cache-nas-pvc
    volumeAttributes:
      server: "xxxxxxxx-fbi71.cn-beijing.nas.aliyuncs.com"
      path: "/tarTest"
  mountOptions:
  - nolock,tcp,noresvport
  - vers=3
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gene-shenzhen-cache-nas-pvc
  namespace: argo
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 20Gi
  storageClassName: alicloud-nas
  selector:
    matchLabels:
      alicloud-pvname: gene-shenzhen-cache-nas
---
apiVersion: v1
kind: Secret
metadata:
  name: oss-secret
```

```
  namespace: argo
stringData:
  akId: xxxxxxxxxx
  akSecret: xxxxxxxxxxx
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gene-shenzhen-cache-oss-pvc
  namespace: argo
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 200Gi
  selector:
    matchLabels:
      alicloud-pvname: gene-shenzhen-cache-oss
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: gene-shenzhen-cache-oss
  namespace: argo
  labels:
    alicloud-pvname: gene-shenzhen-cache-oss
spec:
  capacity:
    storage: 200Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  csi:
    driver: ossplugin.csi.alibabacloud.com
    volumeHandle: gene-shenzhen-cache-oss //Set the value to the name of the persistent
volume (PV).
    nodePublishSecretRef:
      name: oss-secret
      namespace: argo
    volumeAttributes:
      bucket: "oss-test-tsk"
      url: "oss-cn-beijing.aliyuncs.com"
      otherOpts: "-o max_stat_cache_size=0 -o allow_other"
```

4. Create PVs and persistent volume claims (PVCs).

```
kubectl create -f hybridStorage.yaml
```

5. Add the following content to the *hybrid.yaml* file.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow                  #new type of k8s spec
metadata:
  generateName: mpileup-    #name of workflow spec
```

```
  generateName: mpileup      #name of workflow spec
  namespace: argo
spec:
  entrypoint: mpileup         #invoke the whalesay template
  arguments:
    parameters:
    # fastq define
    - name: bucket
      value: "my-test-shenzhen"
    - name: fastqDir
      value: "sample"
    - name: fastq1
      value: "MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_1.fq.gz"
    - name: fastq2
      value: "MGISEQ2000_PCR-free_NA12878_1_V100003043_L01_2.fq.gz"
    - name: reference
      value: "hg19"
    # gene define
    - name: bamDir
      value: "outbam/bam"
    - name: vcfDir
      value: "output/vcf"
    - name: bamFileName
      value: "gene.bam"
    - name: cpuNumber
      value: "32"
    - name: vcfFileName
      value: "gene.vcf"
    - name: service
      value: "s"
  volumes:
  - name: workdir
    persistentVolumeClaim:
      claimName: gene-shenzhen-cache-nas-pvc
  - name: outputdir
    persistentVolumeClaim:
      claimName: gene-shenzhen-cache-oss-pvc
  - name: agsconfig
    configMap:
      name: config
  templates:
  #It is executed remotely and accelerated
  - name: agsmapping
    container:
      image: registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1
      imagePullPolicy: Always
      command: [sh,-c]
      args:
      - ags remote run mapping --region cn-shenzhen --fastq1 sample/{{workflow.paramete
rs.fastq1}} --fastq2 sample/{{workflow.parameters.fastq2}} --bucket {{workflow.paramete
rs.bucket}} --output-bam {{workflow.parameters.bamDir}}/{{workflow.parameters.bamFileNa
me}} --reference {{workflow.parameters.reference}} --service s --watch;
      volumeMounts:
      - name: agsconfig
        mountPath: /root/.ags/config
```

```
      subPath: config
#Download BAM file from OSS to NAS
- name: mpileupprepare
  container:
    image: registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1
    imagePullPolicy: Always
    command: [sh,-c]
    args:
    - cp /output/outbam/bam/gene.bam /data/sample/.
    resources:
      requests:
        memory: 8Gi
        cpu: 4
    volumeMounts:
    - name: workdir
      mountPath: /data
    - name: outputdir
      mountPath: /output
#It is executed locally
- name: samtoolsindex
  retryStrategy:
    limit: 3
  container:
    image: registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1
    command: [sh,-c]
    args:
    - samtools index -@ {{workflow.parameters.cpuNumber}} /data/sample/gene.bam
    volumeMounts:
    - name: workdir
      mountPath: /data
    resources:
      requests:
        memory: 20Gi
        cpu: 4
##Upload index file from NAS to OSS
- name: upload
  retryStrategy:
    limit: 3
  container:
    image: registry.cn-beijing.aliyuncs.com/shuangkun/genetool:v1.1
    command: [sh,-c]
    args:
    - cp /data/sample/gene.bam.bai /output/outbam/bam/.
    volumeMounts:
    - name: workdir
      mountPath: /data
    - name: outputdir
      mountPath: /output
    resources:
      requests:
        memory: 20Gi
        cpu: 4
- name: mpileup
  dag:
```

```
      tasks:
      - name: agsmappingtask
        template: agsmapping
      - name: download-data
        dependencies: [agsmappingtask]
        template: mpileupprepare
      - name: index
        dependencies: [download-data]
        template: samtoolsindex
      - name: upload-data
        dependencies: [index]
        template: upload
```

6. Create a hybrid workflow.

```
kubectl create -f hybrid.yaml
```

After you create a hybrid workflow, the workflow automatically runs.

Find the bucket that corresponds to gene-shenzhen-cache-oss-pvc, and go to the */outbam/bam/* path of the bucket. If *gene.bam.bai* exists, the hybrid workflow is created and running properly.

4.WDL workflow

4.1. Create a WDL workflow

Workflow Description Language (WDL) is a language developed by Broad Institute. WDL specifies data processing workflows with a human-readable and writable syntax. You can use WDL to create bioinformatics workflows in an efficient manner. This topic describes how to use Alibaba Cloud Genomics Service (AGS) to create and run a WDL workflow in a Container Service for Kubernetes (ACK) cluster.

Prerequisites

- An ACK cluster is created. For more information, see Create an ACK managed cluster.
- One or more storage services, such as Apsara File Storage NAS (NAS), Object Storage Service (OSS), or file systems that support the Network File System (NFS) protocol, are deployed. These storage services are used to store input and output data. For more information, see Mount a dynamically provisioned NAS volume.

Benefits of running WDL workflows in ACK clusters

- WDL is compatible with open source Cromwell, which supports WDL scripts. You can directly run WDL workflows in ACK clusters without the need to modify the legacy workflows. For more information about WDL, see WDL.
- WDL allows you to assign the Guaranteed quality of service (QoS) class to pods to optimize resource allocation for tasks. This prevents load increases and performance degradation caused by resource contention on nodes.
- WDL is seamlessly integrated with Alibaba Cloud storage services. WDL can directly access OSS and NAS, and allows you to ingest data from multiple data sources.

Differences between WDL workflows and AGS workflows

- Compared with workflows run by a Cromwell server, cloud-native AGS workflows have the following benefits:
  - Resource control based on parameters such as CPU, Mem min, and Mem max.
  - Scheduling optimization, automatic retry, and dynamic adjustments of resource quotas.
  - Monitoring and logging.
- WDL has a lower resource usage level than AGS workflows. Therefore, the success rate of delivering multiple samples at the same time is also lower than AGS workflows. To create a large number of recurring workflows, we recommend that you use AGS workflows. For more information, see Create a workflow.
- To reduce the cost and accelerate CPU-intensive tasks such as mapping, HaplotypeCaller (HC), and Mutect2, we recommend that you call the AGS API. For more information, see Use AGS to process WGS tasks.

Step 1: Deploy an application

Select all components that are required to create a WDL workflow and package them into a Helm chart. You can upload the chart to the Marketplace module in the ACK console. This way, you can install these components in an application that is deployed by using the chart from Marketplace with a few clicks.

1.
2.
3. On the **Marketplace** page, click the **App Catalog** tab. Then, find and click **ack-arms-pilot**.
4. On the **ack-ags-wdl** page, click **Deploy**.
5. In the **Deploy** wizard, select a cluster and namespace, and then click **Next**.
6. On the **Parameters** wizard page, configure the parameters and click **OK**.

```
# PVCs to be mounted
naspvcs:
  - naspvc1
  - naspvc2
osspvcs:
  - osspvc1
  - osspvc2
# Complete the configurations in the transfer-pvc.yaml and storageclass.yaml files.
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 ro
ot
  path: "/tarTest" #eg "/tarTest"
  # strorage driver. flexVolume or csi,default is csi, if your kubernetes use flexVolum
e,chang it to flexVolume
  driver: "csi" # by default is csi, you could change to flexVolume for early version o
f K8s (<= 1.14.x)
  nasbasepath: "/ags-wdl-nas"
  # mountoptions. if you use local NFS, modify it to your mount options.
  # Complete the configurations in the storageclass.yaml and transfer-pvc.yaml files.
  mountVers: "3" #mount version-svc:
  mountOptions: "nolock,tcp,noresvport"  # mount options, for local NFS server, you cou
ld remove noresvport 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 ro
ot
  path: "/tarTest/bwatest" #eg "/tarTest"
  # strorage driver. flexVolume or csi,default is csi, if your kubernetes use flexVolum
e,chang it to flexVolume
  driver: "csi" # by default is csi, you could change to flexVolume for early version o
f K8s (<= 1.14.x)
  nasbasepath: "/ags-wdl-nas2"
  # mountoptions. if you use local NFS, modify it to your mount options.
  # Complete the configurations in the storageclass.yaml and transfer-pvc.yaml files.
  mountVers: "3" #mount version-svc:
  mountOptions: "nolock,tcp,noresvport"  # mount options, for local NFS server, you cou
ld remove noresvport 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 a
dd basepath before the relative path of the mount.
# e.g.
# {
# "wf.bwa_mem_tool.reference": "/ags-wdl/reference/subset_assembly.fa.gz",# The names o
f the reference files.
# "wf.bwa_mem_tool.reads_fq1": "/ags-wdl/fastq_sample/SRR1976948_1.fastq.gz",#fastq1 fi
lename
# "wf.bwa_mem_tool.reads_fq2": "/ags-wdl/fastq_sample/SRR1976948_2.fastq.gz",#fastq2 fi
lename
# "wf.bwa_mem_tool.outputdir": "/ags-wdl/bwatest/output", #output path,the result will
output in xxxxxxxx.cn-beijing.nas.aliyuncs.com:/mydata_root/bwatest/output,you should m
ake sure the path exists.
# "wf.bwa_mem_tool.fastqFolder": "fastqfolder" # The working directory of the task.
# }
# 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 wil
l create a volume 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 i
nstead for external access to cromwell server.
#config in config.yaml
config:
  # The project namespace. Default value: wdl. You do not need to modify the value in m
ost scenarios.
  namespace: "wdl" # namespace
  # The TESK backand domain name. Default value: tesk-api. You do not need to modify th
e value in most scenarios.
  teskserver: "tesk-api"
  # The domain name of the Cromwell server. Default value: cromwellserver. You do not n
eed to modify the value in most scenarios.
  cromwellserver: "cromwellserver"
  # The port of the Cromwell server. Default value: 8000. You do not need to modify the
value in most scenarios.
  cromwellport: "8000"
```

```
cromwellport: "8000"
```

- Mount volumes to store application data.

  You can use NAS and OSS volumes to store application data.

Parameter	Description	Required	Configuration method

naspvcs	The NAS file system that you want to mount to the ACK cluster as a volume.	Yes	If you want to mount two persistent volume claims (PVCs) named `naspvc1` and `naspvc2`, add the following configuration:
			`naspvcs:`
			`- naspvc1`
			`- naspvc2`

osspvcs	The OSS volumes that you want to mount to your cluster.	Yes	If you want to mount two PVCs named `osspvc1` and `osspvc2`, add the following configuration:
			`osspvcs:`
			`- osspvc1`
			`- osspvc2`

- Configure the NAS volume.

  Each NAS volume is mounted to a container path that is specified by the `nasbasepath` parameter. To set the `naspvcs` parameter, you must configure each NAS volume that you want to mount.

  > **Note** Use the default settings for other parameters.

Parameter	Description	Required	Configuration method
server	NAS IP	Yes	The mount target of the NAS file system that is used to store input and output data. You must change the values of the parameters to the URL and path of your NAS file system.
path	The subdirectory of the NAS file system that you want to mount.	Yes	

Parameter	Description	Required	Configuration method
driver	Flexvolume or CSI	Yes	The volume plug-in that is used by your cluster. The default plug-in is Container Storage Interface (CSI). If your cluster uses FlexVolume, set the value to Flexvolume.
nasbasepath	The application directory to which you want to mount the NAS subdirectory.	Yes	The command or script directory of the application to which you want to mount the NAS subdirectory. To write data to and read data from the NAS file system, you must replace `server:path` in the absolute path with `nasbasepath`. For example, you replace *xxx-xxx.cn-beijing.nas.aliyuncs.com:/wdl* with */ags-wdl*. In this case, the NAS path of the input file `test.fq.gz` is *xxx-xxx.cn-beijing.nas.aliyuncs.com:/wdl/test/test.fq.gz*. The application can use the */ags-wdl/test/test.fq.gz* path to access the input data. In the following examples, `server` is set to `xxx-xxx.cn-beijing.nas.aliyuncs.com`, the absolute path is set to */wdl*, and *nasbasepath* is mapped to */ags-wdl*.

Parameter	Description	Required	Configuration method
mountOptions	Mount options.	Yes	Set the parameters that are required to mount the NAS file system. If you use a self-managed NAS file system, you must modify this parameter.
mountVers	The version of the NFS protocol.	Yes	

- Configure the OSS volume.

  Each OSS volume is mounted to a container path that is specified by the `nasbasepath` parameter. To set the `osspvcs` parameter, you must configure each OSS volume that you want to mount.

Parameter	Description	Required	Configuration method
bucket	oss bucket	Yes	Change the values of the parameters to the name and URL of your OSS bucket.
url	oss url	Yes	
options	Mount parameters.	Yes	Default value: `-o max_stat_cache_size=0 -o allow_other`. You do not need to modify the value.
akid	AccessKey ID	Yes	The AccessKey information of your account that is stored in a Secret in the cluster.
aksecret	AccessKey Secret	Yes	
path	The subdirectory of the OSS bucket that you want to mount.	Yes	Specify the subdirectory of the OSS bucket that you want to mount.

Parameter	Description	Required	Configuration method
ossbasepath	The application directory to which you want to mount the OSS subdirectory.	Yes	The application directory to which you want to mount the OSS subdirectory. To write data to and read data from the OSS bucket, you must replace `bucket:path` in the absolute path with `ossbasepath`. For example, `bucket` is set to shenzhen-test, `path` is set to */wdl*, and `ossbasepath` is set to */ags-wdl*. The OSS path of the input file test.fq.gz is *shenzhen-test:/wdl/test/test.fq.gz*. When you enter the path of the input file, enter */ags-wdl/test/test.fq.gz*. Then, the application can access the input data.

- Set other parameters.

Parameter	Description	Required	Configuration method
workdir	WorkingDir	Yes	Specify `nasbasepath` or `ossbasepath` as the directory. All files including scripts and error log files that are generated by tasks are stored in the specified directory.
provisiondir	The directory to which dynamically provisioned persistent volumes (PVs) are mounted.	Yes	Specify `nasbasepath` or `ossbasepath` as the directory. All input and output data generated by tasks is stored in the PVs that are mounted to the specified directory.

Parameter	Description	Required	Configuration method
nodeselector	The labels that are used to schedule tasks.	No	You can set this parameter to schedule tasks to nodes with specified labels. If you set `node-type=wdl`, all tasks are scheduled to nodes with the label `node-type=wdl`.
nodeport	The port that you want to open on the Cromwell server.	Yes	Enter the port that you want to open on the Cromwell server. If you want to use the Widdler command-line tool to submit tasks, you must set this parameter. Default value: 32567.
namespace	The namespace that is used by the project.	Yes	Enter the namespace where the application is deployed. Default value: wdl.
teskserver	The domain name of the `tesk` service.	Yes	Enter the domain name of the `tesk` service. You can use the default value.
cromwellserver	CromwellServer	Yes	Enter the domain name of the Cromwell server. You can use the default value.
cromwellport	The port that you want to open on the Cromwell server.	Yes	Enter the port that you want to open on the Cromwell server. You can use the default value.

Run the following command. If the output shows that the **cromwellcli**, **cromwellserver**, and **tesk-api** components run as expected, the application is deployed.

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

Expected output:

```
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
```

Step 2: Submit tasks

You can use AGS or a CLI to submit tasks to a cluster from your on-premises machine. We recommend that you use AGS to submit tasks.

Use AGS to submit tasks

The latest version of the AGS CLI allows you to submit WDL tasks. To use AGS to submit tasks, download the AGS CLI and specify the address of the Cromwell server. For more information about how to download the AGS CLI, see Introduction to AGS CLI.

1. Create a *bwa.wdl* file and a *bwa.json* file.
   - Sample *bwa.wdl* file:

```
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.fa
stq.gz
        wget  https://ags-public.oss-cn-beijing.aliyuncs.com/alignment/SRR1976948_2.f
astq.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
}
```

- Sample *bwa.json* file:

```
{
"wf.bwa_mem_tool.reference": "subset_assembly.fa.gz",# The name of the reference file
.
"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 xxx-xxx.cn-beijing.nas.aliyuncs.com:/wdl/bwatest/output,you should make sur
e the path exists.
"wf.bwa_mem_tool.fastqFolder": "fastqfolder" # The working directory of the task.
}
```

2. Redirect requests that are destined for port 32567 to the Cromwell server.

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

3. Run the following command to configure the endpoint of the Cromwell server.

```
ags config init
```

Expected output:

```
Please input your AccessKeyID
xxxxx
Please input your AccessKeySecret
xxxxx
Please input your cromwellserver url
xxx-xxx.cn-beijing.nas.aliyuncs.com:32567
```

4. Run the following command to submit a WDL task:

```
ags wdl run resource/bwa.wdl resource/bwa.json --watch # The watch parameter can be use
d to specify that the command is synchronous. The next task does not start until the cu
rrent one succeeds or fails.
```

Expected output:

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

5. (Optional)You can query or delete WDL tasks from your on-premises machine.
   - Query WDL tasks.
     - Run the following explain command to query a WDL task:

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

Expected output:

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

- Run the following query command to query a WDL task:

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

  Expected output:

  ```
  INFO[0000] bd747360-f82c-4cd2-94e0-b549d775f1c7 {"calls":{"end":"0001-01-01T00:00:0
  0.000Z","executionStatus":null,"inputs":null,"start":"0001-01-01T00:00:00.000Z"},"e
  nd":"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"}
  ```

- Run the command to delete a WDL task:

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

  Expected output:

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

Use the CLI to submit tasks

To use the CLI to submit tasks, you must create a WDL file and a JSON file. Then, use an image provided by AGS to submit tasks. Specify the endpoint of the Cromwell server in the following format: cluster IP:node port.

1. Create a *bwa.wdl* file and a *bwa.json* file.
   - Sample *bwa.wdl* file:

```
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.fa
stq.gz
        wget  https://ags-public.oss-cn-beijing.aliyuncs.com/alignment/SRR1976948_2.f
astq.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
}
```

- Sample *bwa.json* file:

```
{
"wf.bwa_mem_tool.reference": "subset_assembly.fa.gz",# The name of the reference file
.
"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 xxx-xxx.cn-beijing.nas.aliyuncs.com:/wdl/bwatest/output,you should make sur
e the path exists.
"wf.bwa_mem_tool.fastqFolder": "fastqfolder" # The working directory of the task.
}
```

2. Run the following command to submit a WDL task:

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

Expected output:

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

4.2. WDL examples

This topic provides examples about how to use Workflow Description Language (WDL).

Query job list

Run the following command to query a list of jobs for the WDL workflow:

```
ags wdl list
```

Expected output:

```
+----------+--------------------------+--------------------------------------+------------+
| 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  |
+----------+--------------------------+--------------------------------------+------------+
```

Query details about a job

Run the following command to query the detailed information about a job:

> **Note** The information includes the ID of the job and the path where error log is stored.

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

Expected output:


```
{
    "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    String bamfilename\n    command {\n        cd ${fastqFolder}\n        cp
gene.bam.bai gene.bam.bai.test\n    }\n    runtime {\n    docker: \"registry.cn-beijing.ali
yuncs.com/shuangkun/genetool:v1.1\"\n    memory: \"2GB\"\n    cpu: 1\n    }\n  }\ntask agst
ask {\n    String config\n    command {\n        mkdir /root/.ags\n        cp ${config} /ro
ot/.ags/\n        ags remote list > /ags-wdl-nas/remote.txt\n    }\n    runtime {\n    dock
er: \"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.fastqFol
der\":\"/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_me
m_tool/execution/stdout",
                "backendStatus": "Complete",
                "commandLine": "cd /ags-wdl-nas/sample/MGISEQ2000_PCR-free_NA12878_1_V10000
3043_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_V1000
03043_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_me
m_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"
```

```
                        "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-agstas
k/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-agstas
k/execution/stderr",
                "callRoot": "/ags-wdl-nas/wf/cc195aee-41c7-44b0-89e5-4e94ba47f56b/call-agst
ask",
                "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_V
100003043_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"
}
```

Mount a PV

1.
2.
3.
4. In the left-side navigation pane of the Container Service for Kubernetes (ACK) console, click **Releases**.
5. On the **Releases** page, click the **Helm** tab, find the release name ack-ags-wdl, and then click **Update** in the **Actions** column.
6. In the **Update Release** dialog box, mount a persistent volume (PV) to the cluster. The naspvc3 PV is used as an example.
   i. Specify the name of the PV.

   In the **Update Release** dialog box, find the `naspvcs` field in the code editor and add the following content to the `naspvcs` field:

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

   ii. Add configuration to the PV.

   In the **Update Release** dialog box, add the following configuration to the code editor and click **Update**.

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

   After the PV is mounted, new jobs of the workflow can read data from or write data to the PV.

4.3. Use AGS to speed up a workflow

You can compile a local workflow written in Workflow Description Language (WDL) and an Alibaba Cloud Genomics Service (AGS) workflow to form a hybrid workflow. We recommend that you run time-consuming tasks that have a standard workflow and require a large number of computing resources in the AGS workflow, and run other tasks in the local WDL workflow. Workflow data transfers through Object Storage Service (OSS). This improves operation efficiency and saves resources. This topic describes how to use AGS to create a mapping that allows you to compile and run a hybrid workflow.

Prerequisites

- The permissions to use AGS are granted. AGS is in public preview. Before you use this feature, make sure that you have the permissions. To apply for permissions, visit AGS application.

> **Note** If you log on as a RAM user, enter the ID of the RAM user when you apply for the use of AGS. You can obtain the ID of the RAM user in the Account Management console.

- On the App Catalog page of the ACK console, ack-ags-wdl is installed.

Procedure

1. Configure AGS.
   i. Download and install AGS. For more information, see Introduction to AGS CLI.

   ```
   ags config init
   ```

   After you configure AGS, the AGS configuration file *config* is generated.
   ii. Save *config* in the */ags-wdl-nas/bwatest/ags/config* path.
2. Add the following content to the *agsHybrid.wd* file.

```
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} --output-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"
    cpu: 1
    }
}
workflow wf {
    call agstask
    call downloaddata
    call bwa_mem_tool
    call uploaddata
}
```

3. Add the following content to the *agsHybrid.json* file.

   You must mount ack-ags-wdl to the OSS bucket and the NAS volume. In the following code block, replace */ags-wdl-oss/* and */ags-wdl-nas/* with the paths where you mount ack-ags-wdl.

```
{
     "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 resul
t 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", #The path where the tasks are running.
     "wf.uploaddata.naspath": "/ags-wdl-nas/sample",
     "wf.uploaddata.osspath": "/ags-wdl-oss/output/bam",
     }
```

4. Create a hybrid workflow.

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

After you create a hybrid workflow, the workflow automatically runs.

Find the bucket that corresponds to ags-wdl-oss, and go to the *output/bam* path of the bucket. If *gene.bam.bai* exists, the hybrid workflow is created and running properly.