Batch Compute

Getting Started

Product Documentation

Contents

Getting Started

Preparation

Last updated：2024-01-13 11:19:29

Step1： Sign up for a Tencent Cloud account

If you don't have a Tencent Cloud account yet, please log in to Tencent Cloud official website and click **Sign up** in the top-right corner to register a Tencent Cloud account. For more information, please see How to Sign up for Tencent Cloud.

Step2： Get SecretId and SecretKey

After BatchCompute is activated, if you want to use TencentCloud API, SDK, and CLI to manipulate its APIs, you will need the `SecretId` and `SecretKey`. Please create or view them in the API Key Console.

Step3： Understand basic concepts

To better use the product, you should understand some concepts used in the product. You can also get product information and documentation on the product introduction page.

Step4： Understand COS

As the standard output and remote storage mapping of BatchCompute are related to COS, you should have a basic understanding of COS.

Using CLI - Submit a Job

Last updated：2024-01-13 11:19:28

Scenario

This document describes how to submit a simple job by using TCCLI. The following example shows how to add up the numbers in the Fibonacci sequence. The Python code is specified by the `Command` field of `Application`. The returned result is saved in the configured stdout output location.

Prerequisites

You can get prepared by referring to the steps in Preparation.

Steps

Installing and Configuring TCCLI

1. Install TCCLI by referring to Preparation.
2. Run the following command to verify whether TCCLI is successfully installed:

```
tccli batch help
```

The returned result is as follows, indicating that TCCLI is successfully installed:

```
NAME
     batch
DESCRIPTION
     batch-2017-03-12
USEAGE
     tccli batch <action> [--param...]
OPTIONS
     help
     show the tccli batch help info
     --version
     specify a batch api version
```

```
AVAILABLE ACTION
     DescribeComputeEnv
     Used to query details of the computing environment
     CreateTaskTemplate
     Used to create a task template
```

3. Configure TCCLI by referring to Preparation.

Creating a COS Bucket to Store Results

In this example, the returned result is directly output to the standard output of the system. Batch can collect `stdout` and `stderr` from the standard output and upload them to the specified COS bucket upon task completion. You need to create a bucket and a subfolder for storage in advance.

Create the COS bucket and subfolder based on the instructions in Prepare the COS Directory.

Job Configuration

You can acquire and modify the official example to create a Batch computing environment under a personal account.

Learn each configuration item in the computing environment by referring to the following information.

```
tccli batch SubmitJob --version 2017-03-12 --Job '{
    "JobName": "TestJob",        // Job name
    "JobDescription": "for test ",     // Job description
    "Priority": "1",             // Job priority
    "Tasks": [                   // Task list (this example only contains one task)
        {
            "TaskName": "Task1",    // Task 1 name
            "Application": {         // Task execution command
                "DeliveryForm": "LOCAL",     // Runs the local command.
                "Command": "python -c \\"fib=lambda n:1 if n<=2 else fib(n-1)+fib(n
            },
            "ComputeEnv": {          // Computing environment configuration
                "EnvType": "MANAGED",    // Computing environment types: MANAGED and
                "EnvData": {         // Specific configuration (The current type is
                    "InstanceType": "S1.SMALL1",     // CVM instance type
                    "ImageId": "img-m4q71qnf",       // CVM image ID (o be replaced)
                }
            },
            "RedirectInfo": {        // Configuration of standard output redirection
                "StdoutRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqclou
                "StderrRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqclou
            }
        }
    ]
}'
--Placement'{
    "Zone": "ap-guangzhou-2"    // Availability zone (to be replaced)
}'
```

`SubmitJob` Command

Below is an example of the `SubmitJob` command in Batch:

```
tccli batch SubmitJob --version 2017-03-12  --Job '{"JobName": "TestJob",  "JobDesc
```

The `SubmitJob` command includes the following 3 parameters:

Parameter	Description
version	Version, which is fixed to 2017-03-12.
Job	Job configuration (in JSON format). For more information, see the example.
Placement	Availability zone for job execution.

Replace the fields marked with **to be replaced** in the example with your actual information before running the command. Those parameters include the custom image ID, VPC-related information, COS bucket address, SecretId, and SecretKey. For more information, see Modifying Configuration.
Copy the complete content by clicking the **Copy** button next to the example. Replace the **to be replaced** content with your actual information before running the command.
For more information about job configuration, see Configuring a Job.

Modifying Configuration

Setting ImageId

```
"ImageId": "To be replaced"
```

Use a configured image based on the Cloud-init service. The following images are available for direct use:

CentOS 6.5 image, with an ID of **img-m4q71qnf**

Windows Server 2012 official image, with an ID of **img-er9shcln**.

Configuring StdoutRedirectPath and StderrRedirectPath

```
"StdoutRedirectPath": "To be replaced", "StderrRedirectPath": "To be replaced"
```

Enter the endpoint obtained in the **Creating a COS Bucket to Store Results** step above to StdoutRedirectPath and StderrRedirectPath.

(Optional) Modifying the Availability Zone

```
--Placement '{"Zone": "ap-guangzhou-2"}'
```

This example applies for resources in Guangzhou Zone 2. You can select an availability zone based on the default region configured in TCCLI and apply for resources.
For more information about regions and availability zones, see Regions and Availability Zones.

Viewing the Result

The returned result is as follows, indicating successful execution:

```
{
    "RequestId": "db84b7f8-ff8e-4c11-81c1-9a3b02652a46",
    "JobId": "job-cr8qyyh9"
}
```

Run the following command to view the submitted job information through DescribeJob:

```
$ tccli batch DescribeJob  --version 2017-03-12 --JobId job-cr8qyyh9
{
  "EndTime": "2019-10-08T07:28:07Z",
  "JobState": "SUCCEED",
  "TaskInstanceMetrics": {
     ...
  },
  "Zone": "ap-guangzhou-2",
  "TaskMetrics": {
      ...
  },
  "JobName": "TestJob",
  "Priority": 1,
  "RequestId": "0e5c5ea5-ef25-4f90-b355-cfaa8057d473",
  "TaskSet": [
      {
          ...
      }
  ],
  "StateReason": null,
  "JobId": "job-cr8qyyh9",
  "DependenceSet": [],
  "CreateTime": "2019-10-08T07:27:17Z"
}
```

Run the following command to view the job list of the current region through DescribeJobs:

```
$ tccli batch DescribeJobs  --version 2017-03-12
```

More Features

The preceding example shows a single-task job and includes only the fundamental features, but does not include the mapping remote storage capability. For more information about Batch, see the following topics or the API documentation:

Simplified operations: Batch provides robust capabilities and diverse configuration items, with support for simple call through scripts. For more information, see Preparations and Quick Start.
Running Remote Package: Batch provides a set of **custom images and remote packages with command line support** to fully meet your technical requirements. For more information, see Running Remote Package.
Remote storage mapping: Batch optimizes storage access by simplifying access to the remote storage service in the form of operations on the local file system. For more information, see Mapping Remote Storage.

Using CLI - Compute Environment

Last updated：2024-01-13 11:19:28

Operation Scenarios

This document describes how to create a compute environment on TCCLI, submit a job to it, and then terminate it.

Prerequisites

You can get prepared as instructed in Preparations.

Directions

Installing and configuring TCCLI

1. Install TCCLI as instructed in Installing TCCLI.
2. Run the following command to verify whether TCCLI is successfully installed:

```
tccli batch help
```

If the following is returned, the installation is successful.

```
NAME
     batch
DESCRIPTION
     batch-2017-03-12
USEAGE
     tccli batch <action> [--param...]
OPTIONS
     help
     show the tccli batch help info
     --version
     specify a batch api version
```

```
AVAILABLE ACTION
     DescribeComputeEnv
     This API is used to query compute environment details
     CreateTaskTemplate
     This API is used to create a task template
```

3. Configure TCCLI as instructed in Configuring TCCLI.

Creating COS bucket for result storage

In this example, the returned result is directly output to the standard output of the system. BatchCompute can collect `stdout` and `stderr` from the standard output and upload them to the specified COS bucket upon task completion. You need to create a bucket and a subfolder for storage in advance.
Create the COS bucket and subfolder as instructed in Preparing COS Directory.

Creating compute environment

You can get and modify the official sample to create a BatchCompute compute environment under your personal account. The configuration items in the compute environment are as described below:
You can also refer to APIs related to compute environment, such as CreateComputeEnv.

```
tccli batch CreateComputeEnv --version 2017-03-12 --ComputeEnv '{
    "EnvName": "test compute env",           // Compute environment name
    "EnvDescription": "test compute env",    // Compute environment description
    "EnvType": "MANAGED",                    // Compute environment type, which is `
    "EnvData": {                             // Specific configuration (please see t
        "InternetAccessible": {
            "PublicIpAssigned": "TRUE",
            "InternetMaxBandwidthOut": 50
        },
        "LoginSettings": {
            "Password": "*****"              // Login password (to be replaced)
```

```
          },
          "InstanceType": "S1.SMALL1",        // CVM instance type
          "ImageId": "img-xxxxyyyy"           // CVM image ID (to be replaced)
      },
      "DesiredComputeNodeCount": 2            // Number of desired compute nodes
  }'
  --Placement'{
      "Zone": "ap-guangzhou-2"                // AZ (to be replaced)
  }'
```

Sample request

```
tccli batch CreateComputeEnv --version 2017-03-12  --ComputeEnv '{"EnvName": "test
```

Sample return

In the following returned value, `EnvId` indicates the unique ID of a BatchCompute compute environment.

```
{
    "EnvId": "env-jlatqfkn",
    "RequestId": "297ed003-7373-4950-9721-242d3d40b3ca"
}
```

Viewing compute environment list

Sample request

Run the following command to view the list of compute environments:

```
tccli batch DescribeComputeEnvs --version 2017-03-12
```

Sample return (some information omitted)

```
{
    "TotalCount": 1,
    "ComputeEnvSet": [
        {
            "EnvId": "env-jlatqfkn",
            "ComputeNodeMetrics": {
                ...
            },
            "EnvType": "MANAGED",
            "DesiredComputeNodeCount": 2,
            "EnvName": "test compute env",
            "Placement": {
                ...
            },
            "CreateTime": "2019-10-08T08:55:12Z"
        }
    ],
    "RequestId": "7a1f9338-0118-46bf-b59f-60ace9f154f5"
}
```

Viewing specified compute environment

Sample request

Run the following command to view a specified compute environment:

```
tccli batch DescribeComputeEnv --version 2017-03-12 --EnvId env-jlatqfkn
```

Sample return (some information omitted)

```
{
    "EnvId": "env-jlatqfkn",
    "ComputeNodeMetrics": {
        ...
    },
    "EnvType": "MANAGED",
    "DesiredComputeNodeCount": 2,
    "ComputeNodeSet": [
        ...
    ],
    "RequestId": "407de39c-1c3d-489e-9a35-5257ae561e87",
    "Placement": {
        ...
    },
    "EnvName": "test compute env",
    "CreateTime": "2019-10-08T08:55:12Z"
}
```

Submitting job to specified compute environment

Sample request

Replace related information in the command as needed and run the following command to submit a job to a specified compute environment.

```
tccli batch SubmitJob --version 2017-03-12  --Job '{"JobName": "test job", "JobDesc
```

Sample return

```
{
    "RequestId": "d6903404-5765-474b-b516-39137456fa5a",
    "JobId": "job-qjq3mqp7"
}
```

Terminating compute environment

Sample request

Run the following command to terminate a compute environment:

```
tccli batch DeleteComputeEnv --version 2017-03-12 --EnvId env-jlatqfkn
```

Sample return

```
{
    "RequestId": "029becda-2a4e-4989-aa77-6fbb5a873555"
}
```

Configuring a Job

Last updated：2024-01-13 11:19:28

1. Introduction

The job configuration of BatchCompute is provided in JSON format as described below. The job listed below consists of 2 tasks.

```
{
    "JobName": "TestJob",       // Job name
    "JobDescription": "for test ",     // Job description
    "Priority": "1",            // Job priority
    "Tasks": [                  // Task list (this example has two tasks)
        {
            // Task 1 (the simplest task configuration without any non-mandatory op
            "TaskName": "Task1",    // Task 1 name
            "Application": {         // Task execution command
                "DeliveryForm": "LOCAL",     // Delivery method of application
                "Command": "echo hello"      // Command content (output hello)

            },
            "ComputeEnv": {          // Compute environment configuration
                "EnvType": "MANAGED",    // Compute environment types: managed and u
                "EnvData": {         // Specific configuration (current type is mana
                    "InstanceType": "S1.SMALL1",     // CVM instance type
                    "ImageId": "img-m4q71qnf",       // CVM image ID
                }
            },
            "RedirectInfo": {        // Configuration of standard output redirection
                "StdoutRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqclou
                "StderrRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqclou
            },
            "Authentications": [         // Authentication information (optional, us
                {
                    "Scene": "COS",      // Scenario (currently in COS)
                    "SecretId": "***",   // SecretId (to be replaced)
                    "SecretKey": "***"   // SecretKey (to be replaced)
                }
            ]
        },
        {
            // Task 2
            "TaskName": "Task2",    // Task 2 name
            "TaskInstanceNum": 1,    // Number of Task 2 concurrent instances
            "Application": {         // Task execution command
                "DeliveryForm": "LOCAL",     // Run the local command
                "Command": "python -c \\"fib=lambda n:1 if n<=2 else fib(n-1)+fib(n
            },
            "ComputeEnv": {          // Compute environment configuration
                "EnvType": "MANAGED",    // Compute environment types: managed and u
                "EnvData": {         // Specific configuration (current type is mana
                    "InstanceType": "S1.SMALL1",     // CVM instance type
                    "ImageId": "img-m4q71qnf",       // CVM image ID (replaceable)
                    "VirtualPrivateCloud": {         // CVM network configuration (o
                        "VpcId": "vpc-cg18la4l",              // VpcId (to be replac
                        "SubnetId": "subnet-8axej2jc"            // SubnetId (to be
                    },
                    "SystemDisk": {                     // CVM system disk configuratio
                        "DiskType": "CLOUD_BASIC",
                        "DiskSize": 50
                    },
                    "DataDisks": [                      // CVM data disk configuration
                        {
                            "DiskType": "CLOUD_BASIC",
                            "DiskSize": 50
                        }
                    ]
                }
            },
            "RedirectInfo": {        // Configuration of standard output redirection
                "StdoutRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqclou
                "StderrRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqclou
            },
            "MaxRetryCount": 1,          // Maximum number of retry attempts
            "Authentications": [         // Authentication information (optional, us
                {
                    "Scene": "COS",      // Scenario (currently in COS)
                    "SecretId": "***",   // SecretId (to be replaced)
                    "SecretKey": "***"   // SecretKey (to be replaced)
                }
            ]
        }
    ],
    "Dependences": [
        {
            "StartTask": "Task1",
            "EndTask": "Task2"
        }
    ]
}
```

2. Details

I. Job

A job is a unit submitted by Batch. It contains its own information and the information about one or more tasks as well as the dependencies between tasks.

Name	Type	Required	Description

JobName	String	No	Job name
JobDescription	String	No	Job description
Priority	Integer	Yes	Job priority. Task (Task) and task instance (TaskInstance) inherit the job priority
Task.N	array of Task objects	Yes	Task details
Dependences.N	array of Dependence objects	No	Dependency information

II. Tasks

A job can contain multiple tasks. A task describes the environment (model, system, image) that the actual computing process depends on, the code package to be executed, command line, storage, network and other related information in batch data computing.

Parameter	Type	Required	Description	Value
TaskName	String	Yes	A unique task name in a job	Task1
TaskInstanceNum	Integer	Yes	Number of instances launched	1
Application	Application object	Yes	Application information	-
ComputeEnv	ComputeEnv object	Yes	Operating environment information	-
RedirectInfo	RedirectInfo object	Yes	Path for Redirection	-
InputMappings	array of InputMapping object	No	Input mapping	-
OutputMappings	array of OutputMapping object	No	Output mapping	-
Authentications	array of Authentication object	No	Information for authorization	-
MaxRetryCount	Integer	No	The maximum number of retry attempts after a task failed	3
Timeout	Integer	No	Timeout after a task is launched (in sec)	3,600

Application

Parameter	Type	Required	Description	Example
Command	String	Yes	Task execution command	
DeliveryForm	String	Yes	Delivery method of application	**LOCAL** (local), **PACKAGE** (remote code package)
PackagePath	String	No	The path to remote code package. It should be in .tgz format.	``http://batchdemo-1251783334.cosgz.myqcloud.com/codepkg/cod

ComputeEnv

Parameter	Type	Required	Description	Example
EnvType	String	Yes	Compute environment management types	**MANAGED**, **UNMANAGED**
EnvData	EnvData object	Yes	Compute environment's parameters	-

EnvData

Parameter	Type	Required	Description	Example
InstanceType	String	Yes	CVM instance type, required for managed environments	S1.SMALL1
ImageId	String	Yes	CVM image ID, required for managed environments	img-m4q71qnf
others	others	no	Refer to the parameters provided in the CVM API document Instance Creation	SystemDisk, DataDisks, and VirtualPrivateCloud are supported

RedirectInfo

Parameter	Type	Required	Description	Example

StdoutRedirectPath	String	No	Path for standard output redirection	cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/logs/
StderrRedirectPath	String	No	Path for standard error redirection	cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/logs/

InputMapping

Parameter	Type	Required	Description	Example
SourcePath	String	Yes	Source path	cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/input/
DestinationPath	String	Yes	Destination path	/data/input/

OutputMapping

Parameter	Type	Required	Description	Example
SourcePath	String	Yes	Source path	/data/output/
DestinationPath	String	Yes	Destination path	cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/output/

Authentication

Leave it blank if you’re the owner of the COS resource specified in storage mapping and log redirection. Otherwise please enter the secret to access the COS resource.

Name	Type	Required	Description
Scene	String	Yes	Authentication scenario, such as `COS`
SecretId	String	Yes	SecretId
SecretKey	String	Yes	SecretKey

III. Task Dependence

It describes the execution order of tasks. If a job contains two tasks: StartTask for Task 1 and EndTask for Task 2, Task 2 is launched only after the execution of Task 1 is completed. When Task 2 is completed, the entire job is finished.

Parameter	Type	Required	Description	Example
StartTask	String	Yes	Name of start task in a dependency	Task1
EndTask	String	Yes	Name of end task in a dependency	Task2

Using Console

Last updated：2024-01-13 11:19:28

Getting Started

This document describes how to submit a job with the BatchCompute Console. The steps are detailed below.

Preparations

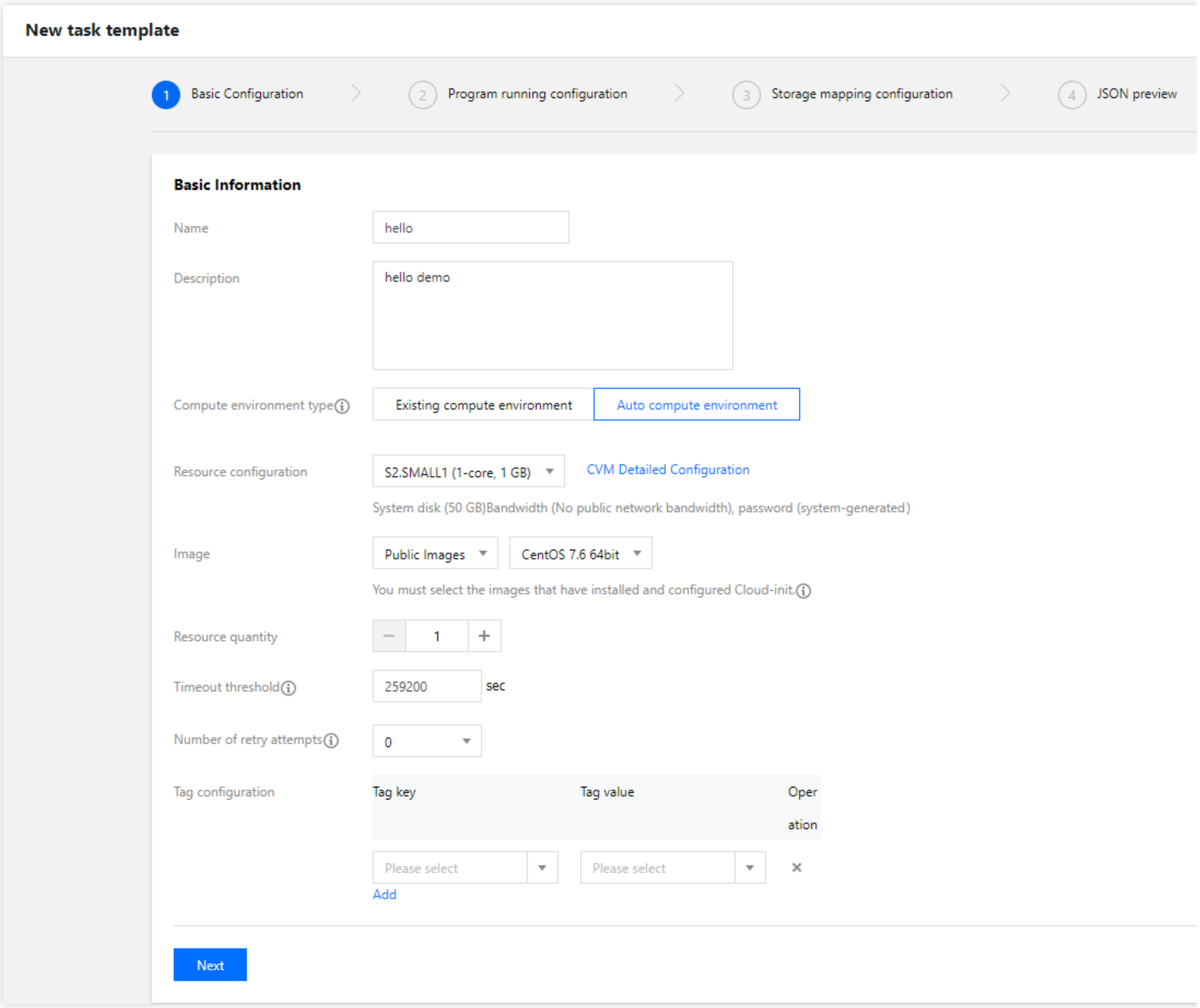
Prepare a COS bucket. If you have not created a bucket, create one as instructed in Creating Bucket.

Logging in to the Console

Log in to the BatchCompute console. If you have not activated the BatchCompute service, activate it as prompted on the homepage.

Creating a Task Template

1. Click **Task Template** in the left sidebar, and select a target region, such as **Guangzhou**, at the top of the **Task Template** page.
2. Click **Create**. On the **New task template** page, create a template, as shown below:


New task template

1 Basic Configuration
2 Program running configuration
3 Storage mapping configuration
4 JSON preview

Basic Information

Name: hello

Description: hello demo

Compute environment type: Existing compute environment | [✓ Auto compute environment]

Resource configuration: S2.SMALL1 (1-core, 1 GB) CVM Detailed Configuration

System disk (50 GB)Bandwidth (No public network bandwidth), password (system-generated)

Image: Public Images CentOS 7.6 64bit

You must select the images that have installed and configured Cloud-init.

Resource quantity: 1

Timeout threshold: 259200 sec

Number of retry attempts: 0

Tag configuration:

Tag key	Tag value	Operation
Please select	Please select	×

The main parameters are described as follows:

Name: Enter a custom name, such as **hello**.

Description: Enter a custom description, such as **hello demo**.

Compute environment type:

Existing compute environment: Select an existing compute environment.

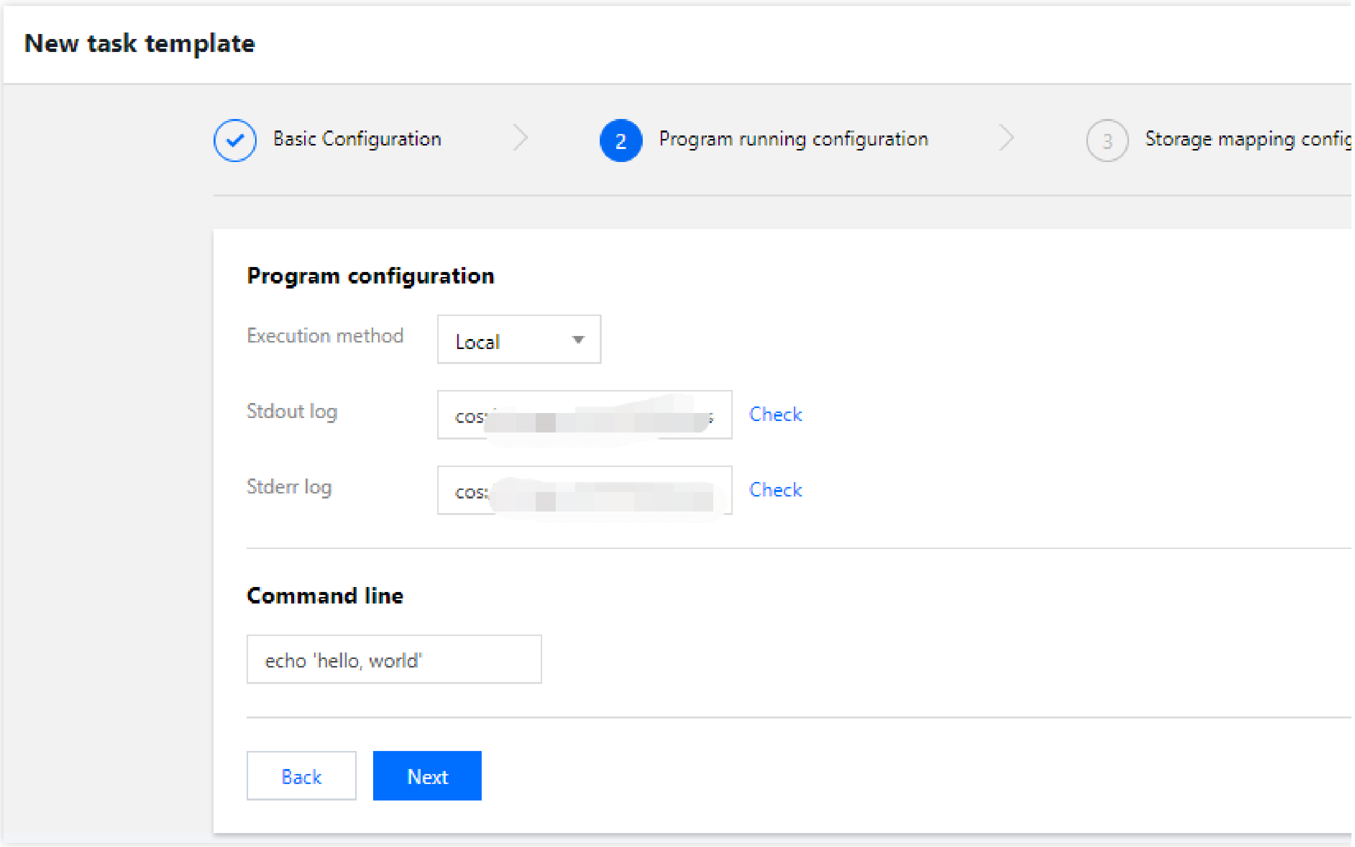
Auto compute environment: You do not need to create a fixed compute environment in advance. After the job is submitted, a CVM instance is automatically created to run the job. After the job is completed, the CVM instance is automatically terminated.

Image: Select an image as needed.

Resource quantity: Example: 1.

Timeout threshold and **Number of retry attempts**: Keep the default values.

3. Click **Next**. Configure application information, as shown below:


New task template

Basic Configuration > 2 Program running configuration > 3 Storage mapping config

Program configuration

Execution method: Local

Stdout log: cos [redacted] Check

Stderr log: cos [redacted] Check

Command line

echo 'hello, world'

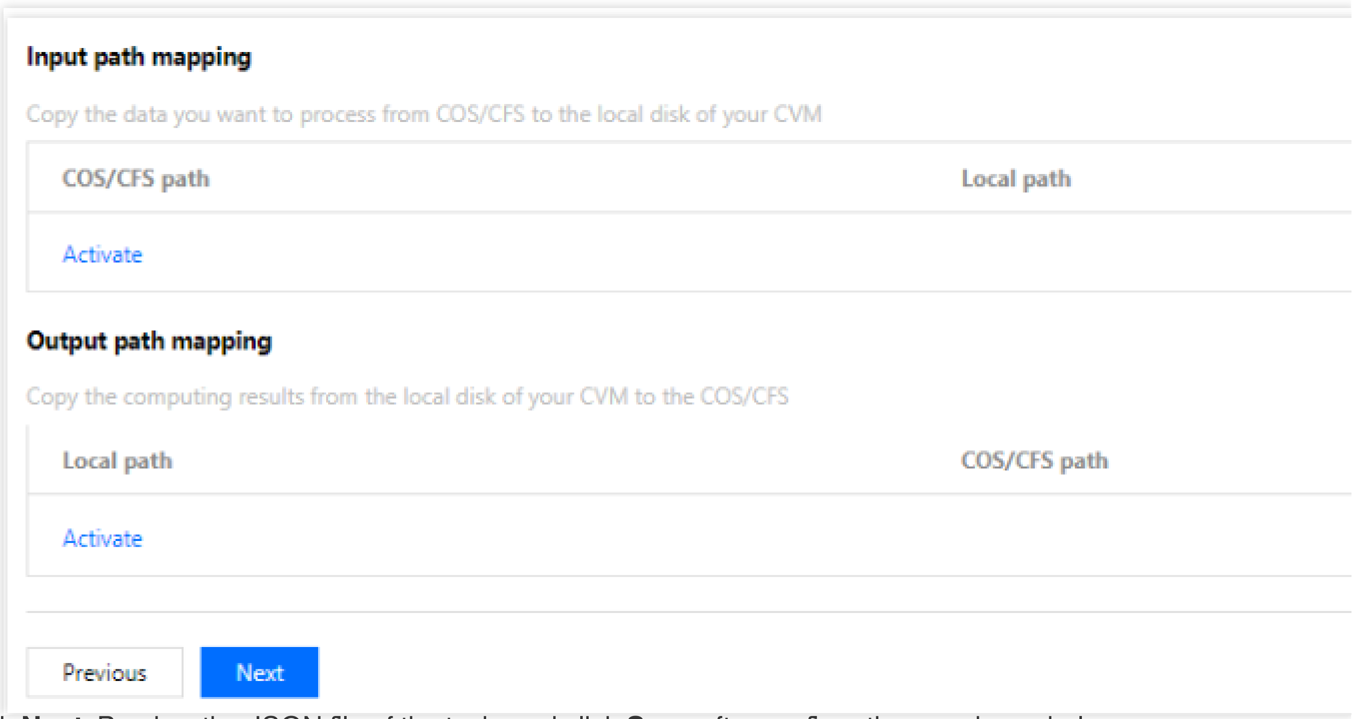
Execution method: Select **Local**.

Stdout log: For more information about the format, see Entering COS & CFS Paths.

Stderr log: The same as above.

Command line: Enter **echo 'hello, world'**.

4. Click **Next**. Configure the storage mapping, as shown below:



Input path mapping

Copy the data you want to process from COS/CFS to the local disk of your CVM

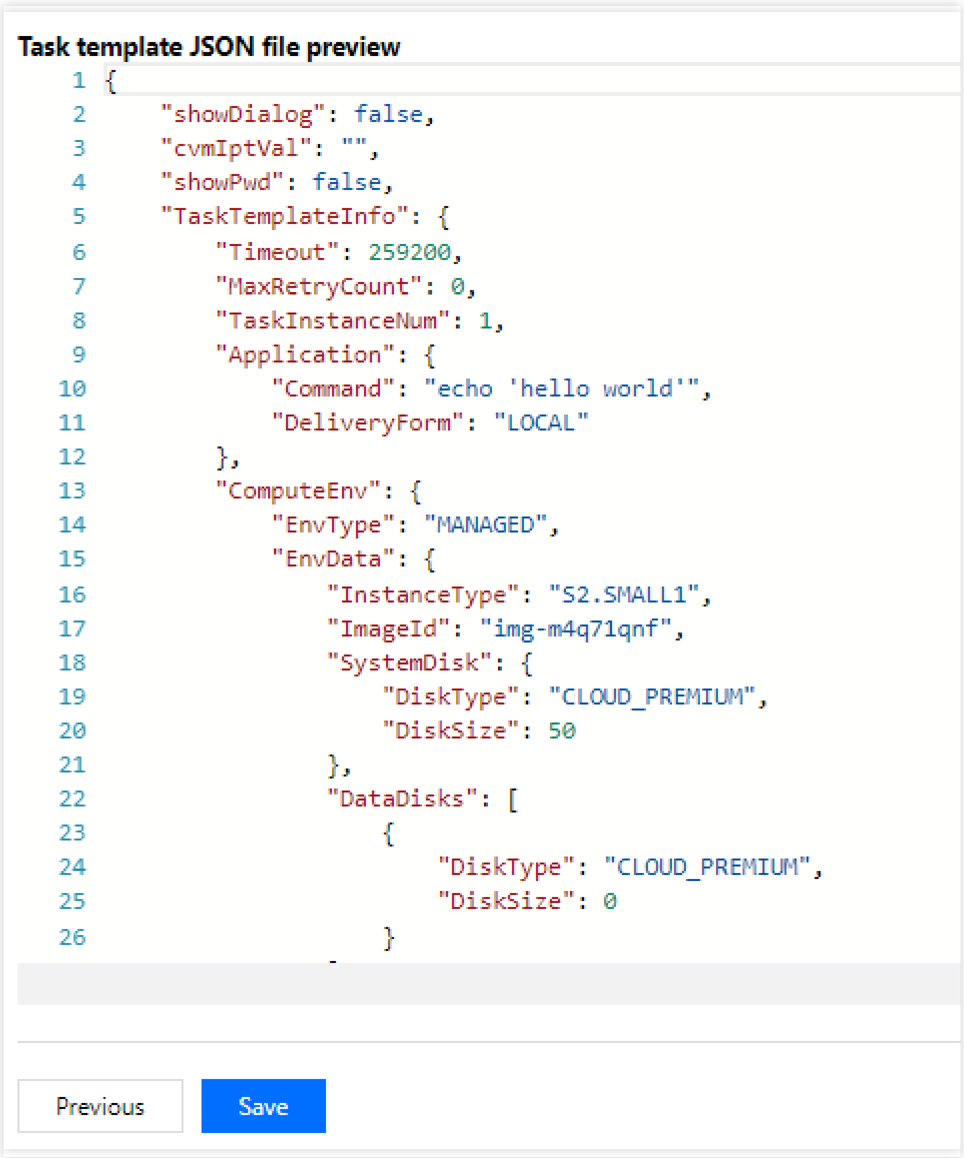
COS/CFS path	Local path
Activate	

Output path mapping

Copy the computing results from the local disk of your CVM to the COS/CFS

Local path	COS/CFS path
Activate	

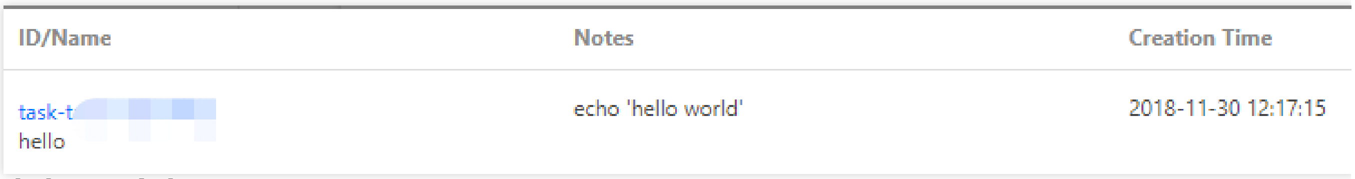
5. Click **Next**. Preview the JSON file of the task, and click **Save** after confirmation, as shown below:



Task template JSON file preview

```
{
    "showDialog": false,
    "cvmIptVal": "",
    "showPwd": false,
    "TaskTemplateInfo": {
        "Timeout": 259200,
        "MaxRetryCount": 0,
        "TaskInstanceNum": 1,
        "Application": {
            "Command": "echo 'hello world'",
            "DeliveryForm": "LOCAL"
        },
        "ComputeEnv": {
            "EnvType": "MANAGED",
            "EnvData": {
                "InstanceType": "S2.SMALL1",
                "ImageId": "img-m4q71qnf",
                "SystemDisk": {
                    "DiskType": "CLOUD_PREMIUM",
                    "DiskSize": 50
                },
                "DataDisks": [
                    {
                        "DiskType": "CLOUD_PREMIUM",
                        "DiskSize": 0
                    }
```

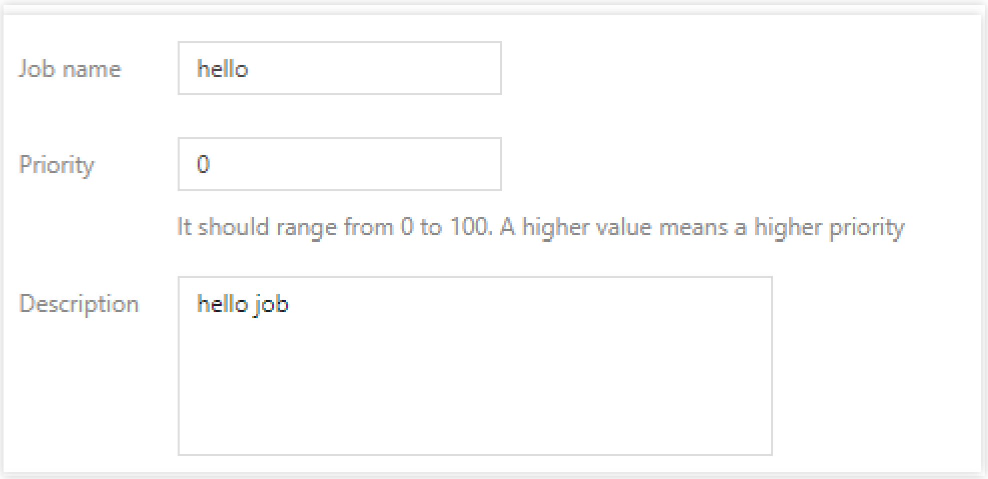
6. View the created template on the **Task Template** page, as shown below:



ID/Name	Notes	Creation Time
task-t[redacted] hello	echo 'hello world'	2018-11-30 12:17:15

Submitting a Job

1. Click **Jobs** in the left sidebar, and select a target region, such as **Guangzhou**, at the top of the **Jobs** page.
2. Click **Create**. On the **New job** page, create a job, as shown below:



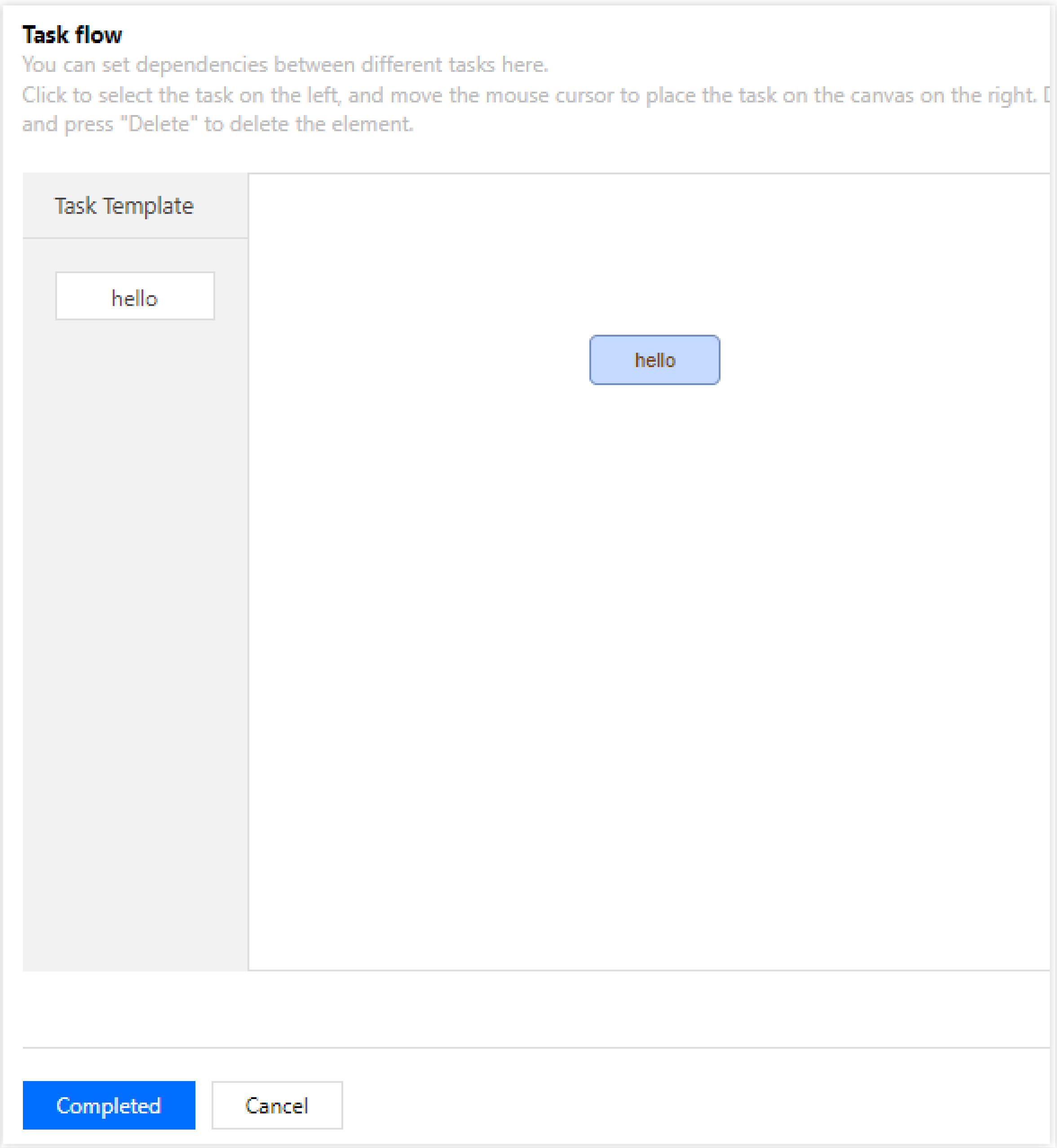
Job name	hello
Priority	0
	It should range from 0 to 100. A higher value means a higher priority
Description	hello job

Job name: Enter a custom name, such as **hello**.

Priority: Keep the default value.

Description: Enter a custom description, such as **hello demo**.

3. On the left side of the **Task flow** page, find the **hello** task and drag it to the canvas on the right, as shown below:

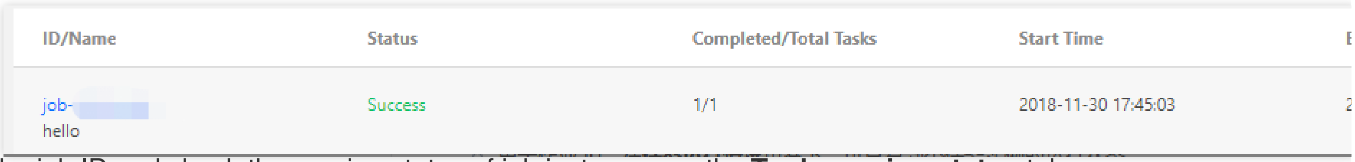


Task flow

You can set dependencies between different tasks here.

Click to select the task on the left, and move the mouse cursor to place the task on the canvas on the right.
and press "Delete" to delete the element.

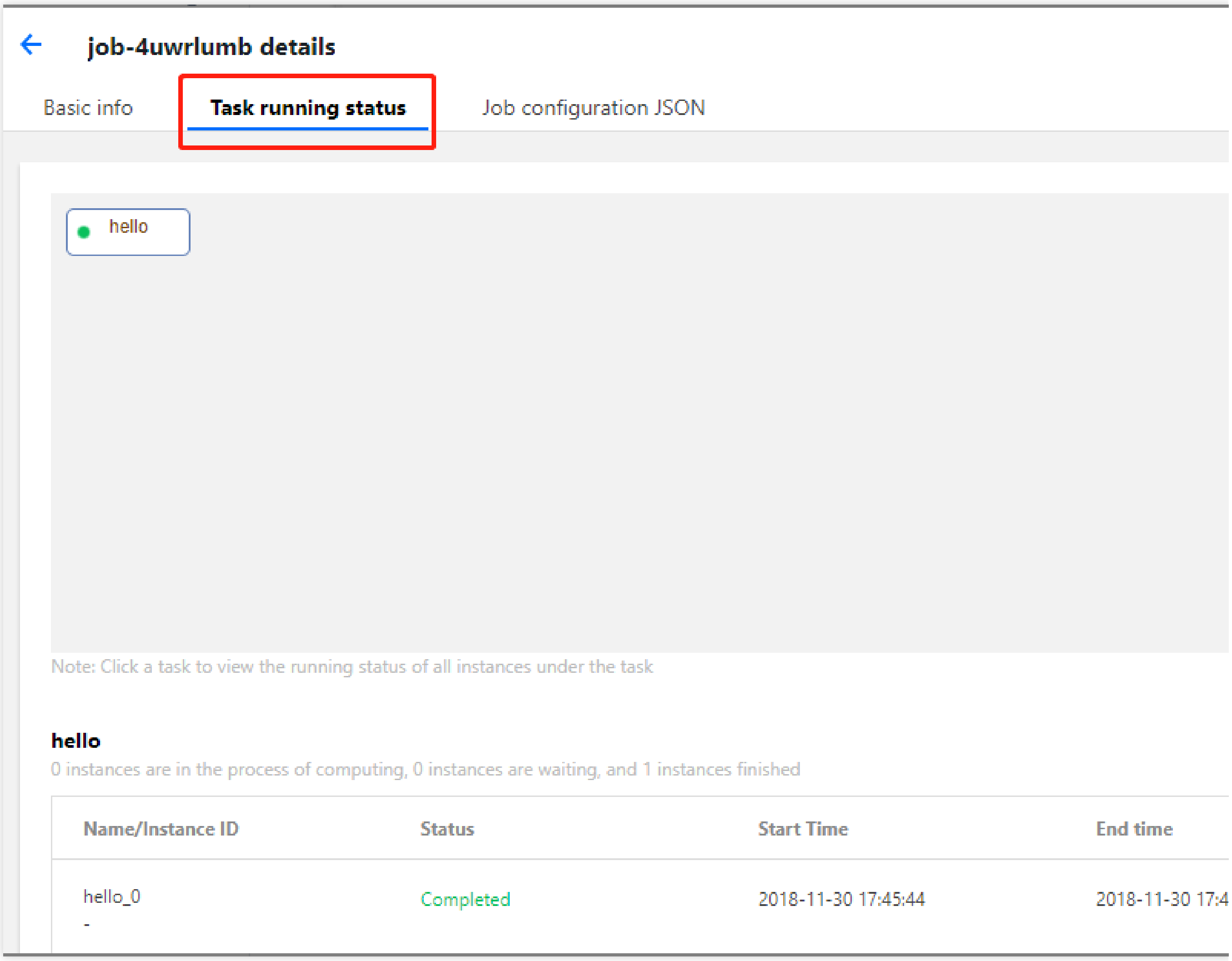
4. Confirm the configurations in the task details area on the right side of the **Task flow** page and click **Completed**.
5. Check the running status of the created job on the **Jobs** page, as shown below:



ID/Name	Status	Completed/Total Tasks	Start Time
job-[redacted] hello	Success	1/1	2018-11-30 17:45:03

Click the job ID and check the running status of job instances on the **Task running status** tab.

Click **View Log** to check the standard outputs and errors of job instances, as shown below:



job-4uwrlumb details

Note: Click a task to view the running status of all instances under the task

hello

0 instances are in the process of computing, 0 instances are waiting, and 1 instances finished

Name/Instance ID	Status	Start Time	End time
hello_0 -	Completed	2018-11-30 17:45:44	2018-11-30 17:4

Subsequent Operations

This document only provides a simple example of single-task job, without using the remote storage mapping, and only showing the most basic capabilities to users. You can continue to test the higher-level capabilities of BatchCompute according to the console user guide:

Various CVM configurations: BatchCompute provides a variety of CVM configuration options. You can customize your own CVM configuration based on your business scenario.

Executable remote code package: Technically, BatchCompute can fully satisfy your business needs by combining the custom image, remote code package, and command line features.

Remote storage mapping: BatchCompute optimizes storage access and simplifies access to remote storage services into operations in the local file system.