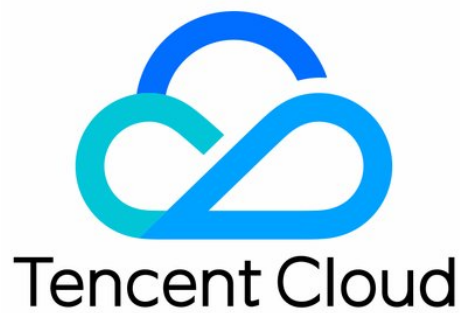


Batch Compute

Getting Started

Product Documentation



Copyright Notice

©2013-2019 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

 Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Getting Started

- Preparation

- Using CLI - Submit a Job

- Using CLI - Compute Environment

- Configuring a Job

- Using Console

Getting Started

Preparation

Last updated : 2020-05-13 17:20:20

1. 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](#).

2. 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](#).

3. 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](#).

4. 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 : 2019-10-18 14:35:22

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
USAGE
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-2); print(fib(20))¥" " // Command content (Fibonacci summation)
      },
      "ComputeEnv": { // Computing environment configuration
        "EnvType": "MANAGED", // Computing environment types: MANAGED and UNMANAGED
        "EnvData": { // Specific configuration (The current type is MANAGED. Refer to the CVM instance creation description)
          "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.myqcloud.com/logs/", // Standard output (to be replaced)
        "StderrRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/logs/" // Standard error (to be replaced)
      }
    }
  ],
  "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", "JobDescription": "for test",
"Priority": "1", "Tasks": [{"TaskName": "Task1", "TaskInstanceNum": 1, "Application": {"DeliveryForm":
"LOCAL", "Command": "python -c ¥"fib=lambda n:1 if n<=2 else fib(n-1)+fib(n-2); print(fib(20))¥" }, "Co
mputeEnv": {"EnvType": "MANAGED", "EnvData": {"InstanceType": "S1.SMALL1", "ImageId": "To be replaced" }
}, "RedirectInfo": {"StdoutRedirectPath": "To be replaced", "StderrRedirectPath": "To be replaced"}, "Ma
xRetryCount": 1 } ] }' --Placement '{"Zone": "ap-guangzhou-2}"'
```

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 : 2020-05-13 17:20:21

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
USAGE
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 `managed` here
  "EnvData": { // Specific configuration (please see the CVM instance creation description)
    "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 compute env", "EnvDescription": "test compute env", "EnvType": "MANAGED", "EnvData": {"InstanceType": "S1.SMALL2", "ImageId": "to be replaced", "LoginSettings": {"Password": "to be replaced"}, "InternetAccessible": {"PublicIpAssigned": "TRUE", "InternetMaxBandwidthOut": 50}, "SystemDisk": {"DiskType": "CLOUD_BASIC", "DiskSize": 50 }}, "DesiredComputeNodeCount": 2 }' --Placement '{"Zone": "ap-guangzhou-2}"
```

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", "JobDescription": "xxx", "Priority": "1", "Tasks": [{"TaskName": "hello2", "TaskInstanceNum": 1, "Application": {"DeliveryForm": "LOCAL", "Command": "python -c ¥"fib=lambda n:1 if n<=2 else fib(n-1)+fib(n-2); print(fib(20))¥" }, "EnvId": "to be replaced", "RedirectInfo": {"StdoutRedirectPath": "to be replaced", "StderrRedirectPath": "to be replaced"} } ]}' --Placement '{"Zone": "ap-guangzhou-2}"'
```

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 : 2021-03-25 16:48:05

1. Brief Description

The job configuration in BatchCompute is provided in JSON format. A brief description of the configuration is given below. The following job contains 2 tasks.

```
{
  "JobName": "TestJob", // Job name
  "JobDescription": "for test ", // Job description
  "Priority": "1", // Job priority
  "Tasks": [ // Task list (two tasks in this example)
    {
      // Task 1 (the most simplified task configuration with all non-essential options removed)
      "TaskName": "Task1", // Task 1 name
      "Application": { // Task execution command
        "DeliveryForm": "LOCAL", // Application delivery method
        "Command": "echo hello" // Command content (to output hello)
      },
      "ComputeEnv": { // Compute environment configuration
        "EnvType": "MANAGED", // Compute environment types: MANAGED or UNMANAGED
        "EnvData": { // Specific configuration (the current type is MANAGED. For more information, please see CV
          M instance creation description)
        "InstanceType": "S1.SMALL1", // CVM instance type
        "ImageId": "img-m4q71qnf", // CVM image ID
        }
      },
      "RedirectInfo": { // Configuration of standard output redirection
        "StdoutRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/logs/", // Standard output (to
          be replaced)
        "StderrRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/logs/" // Standard error (to be
          replaced)
      },
      "Authentications": [ // Authentication information (optional, for use in case of COS of others)
        {
          "Scene": "COS", // Scenarios (COS currently)
          "SecretId": "***", // SecretId (to be replaced)
          "SecretKey": "***" // SecretKey (to be replaced)
        }
      ]
    },
    {
      // Task 2
      "TaskName": "Task2", // Task 2 name
      "TaskInstanceNum": 1, // Number of concurrent instances in task 2
    }
  ]
}
```

```
"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-2); print(fib(20))¥" " // Command content (Fibonacci summation)
},
"ComputeEnv": { // Compute environment configuration
"EnvType": "MANAGED", // Compute environment types: MANAGED or UNMANAGED
"EnvData": { // Specific configuration (the current type is MANAGED. For more information, please see CVM instance creation description)
"InstanceType": "S1.SMALL1", // CVM instance type
"ImageId": "img-m4q71qnf", // CVM image ID (to be replaced)
"VirtualPrivateCloud": { // CVM network configuration (optional)
"VpcId": "vpc-cg18la4l", // VpcId (to be replaced)
"SubnetId": "subnet-8axej2jc" // SubnetId (to be replaced)
},
"SystemDisk": { // CVM system disk configuration
"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.myqcloud.com/logs/", // Standard output (to be replaced)
"StderrRedirectPath": "cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/logs/" // Standard error (to be replaced)
},
"MaxRetryCount": 1, // Maximum number of retries
"Authentications": [ // Authentication information (optional, for use in case of COS of others)
{
"Scene": "COS", // Scenarios (COS currently)
"SecretId": "***", // SecretId (to be replaced)
"SecretKey": "***" // SecretKey (to be replaced)
}
]
},
"Dependences": [
{
"StartTask": "Task1",
"EndTask": "Task2"
}
]
}
```

2. Detailed Description

I. Job

A job is a unit of submission in BatchCompute. In addition to its own information, it also contains information about one or more tasks and dependencies among the 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 priority of the job
Tasks.N	array of Task objects	Yes	Task information
Dependencies.N	array of Dependence objects	No	Dependency

II. Task

A job can contain multiple tasks. A task mainly describes the information about the environment (model, system, image) on which the actual computation process depends in BatchCompute, code package and command line executed, storage and network.

Name	Type	Required	Description	Example
TaskName	String	Yes	Task name, which should be unique within a job	Task1
TaskInstanceNum	Integer	Yes	Number of running task instances	1
Application	Application object	Yes	Application information	-
ComputeEnv	ComputeEnv object	Yes	Running environment information	-
RedirectInfo	RedirectInfo object	Yes	Redirection path	-
InputMappings	array of InputMapping object	No	Input mapping	-
OutputMappings	array of OutputMapping object	No	Output mapping	-

Authentications	array of Authentication object	No	Authentication information	-
MaxRetryCount	Integer	No	Maximum number of retries after a task fails	3
Timeout	Integer	No	Timeout period after a task starts in seconds	3600

Application

Name	Type	Required	Description	Example
Command	String	Yes	Task execution command	
DeliveryForm	String	Yes	Application delivery method	LOCAL: local method; PACKAGE: remote code package method
PackagePath	String	No	Remote code package path, which must be in .tgz format	<code>http://batchdemo-1251783334.cosgz.myqcloud.com/codepkg/codepkg.tgz</code> (only for PACKAGE)

ComputeEnv

Name	Type	Required	Description	Example
EnvType	String	Yes	Compute environment management type, including MANAGED and UNMANAGED	LOCAL: local method; PACKAGE: remote code package method
EnvData	EnvData object	Yes	Compute environment's specific parameters	-

EnvData

Name	Type	Required	Description	Example
InstanceType	String	Yes	CVM instance type, which is required for the MANAGED type	S1.SMALL1
ImageId	String	Yes	CVM image ID, which is	img-m4q71qnf

			required for the MANAGED type	
others	others	No	Please see the parameters provided in CVM API document Creating Instances	SystemDisk , DataDisks , VirtualPrivateCloud and other parameters are supported

RedirectInfo

Name	Type	Required	Description	Example
StdoutRedirectPath	String	No	Standard output redirection path	cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/logs/
StderrRedirectPath	String	No	Standard error redirection path	cos://dondonbatchv5-1251783334.cosgz.myqcloud.com/logs/

InputMapping

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

OutputMapping

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

Authentication

If the COS path (storage mapping, log redirection) is a COS address you own, you don't need to enter this parameter. If you need to access COS of others, you need to enter the corresponding access key.

Name	Type	Required	Description
Scene	String	Yes	Authentication scenario such as COS

SecretId	String	Yes	SecretId
SecretKey	String	Yes	SecretKey

III. Dependency

It describes the relationship between tasks. For example, if a job contains two tasks (Task1 as `StartTask` and Task2 as `EndTask`), Task2 will be started after Task1 is executed, and after Task2 is executed, it can be deemed that the job is completed.

Name	Type	Required	Description	Example
StartTask	String	Yes	Dependency start task name	Task1
EndTask	String	Yes	Dependency end task name	Task2

Using Console

Last updated : 2020-06-23 15:52:58

Quick Start

This document describes how to submit a job in the BatchCompute Console. The steps are as follows:

Preparations

Prepare a [COS](#) bucket. If you haven't created a bucket yet, create one as instructed in [Creating Bucket](#).

Logging in to console

If you haven't activated the BatchCompute service, activate it as prompted on the [BatchCompute Console](#) page.

Creating task template

1. Select **Task Template** on the left sidebar and select the target region at the top of the page, such as "Guangzhou".

2. Click **Create** to enter the "Create Task Template" page and create a template as shown below:

Basic info

Name

Description

Resource configuration [CVM Detailed Configuration](#)
System disk (50 GB)Bandwidth (No public network bandwidth), password (system-generated)

Resource quantity units

Timeout ⓘ s

Number of retries ⓘ

Image

Main parameters include:

- **Name:** custom name, such as `hello` .
- **Description:** custom description, such as `hello demo` .
- **Compute Environment Type:**
 - **Existing Compute Environment:** you can select an existing compute environment.
 - **Automatic Compute Environment:** you don't need to pre-create a fixed compute environment. After a job is submitted, a CVM instance will be created automatically to run the task and terminated automatically after task completion.
- **Image:** select as needed.
- **Number of Resources:** such as 1.
- **Timeout Period and Number of Retries:** keep the default values.

3. Click **Next** to configure the program information as shown below:

Program configuration

Execution method

Stdout log [Check](#)

Stderr log [Check](#)

Command line

- **Execution Method:** select "Local".
- **Stdout Log:** for more information on the format, please see [How to Enter COS and CFS Paths](#).
- **Stderr Log:** same as Stdout log.
- **Command Line:** `echo 'hello, world' .`

4. Click **Next** to set 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
<input type="text"/>	<input type="text"/>
Activate	

Output path mapping

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

Local path	COS/CFS path
<input type="text"/>	<input type="text"/>
Activate	

5. Click **Next**, preview the task's JSON file, and click **Save** after confirming that everything is correct.

Task template JSON file preview

```

1 {
2   "showDialog": false,
3   "cvmIptVal": "",
4   "showPwd": false,
5   "TaskTemplateInfo": {
6     "Timeout": 259200,
7     "MaxRetryCount": 0,
8     "TaskInstanceNum": 1,
9     "Application": {
10      "Command": "echo 'hello world'",
11      "DeliveryForm": "LOCAL"
12    },
13    "ComputeEnv": {
14      "EnvType": "MANAGED",
15      "EnvData": {
16        "InstanceType": "S2.SMALL1",
17        "ImageId": "img-m4q71qnf",
18        "SystemDisk": {
19          "DiskType": "CLOUD_PREMIUM",
20          "DiskSize": 50
21        },
22        "DataDisks": [
23          {
24            "DiskType": "CLOUD_PREMIUM",
25            "DiskSize": 0
26          }
27        ]
28      }
29    }
30  }
31 }

```

Previous Save

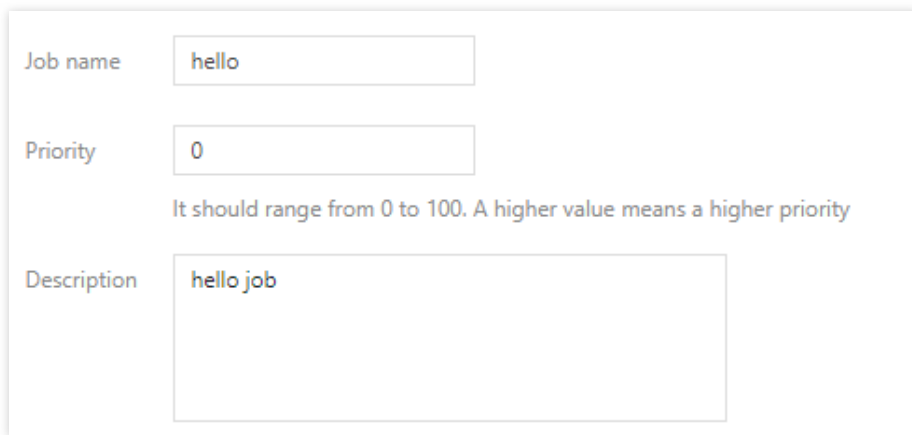
6. After creating a template successfully, you can view it on the "Task Template" list page as shown below:

ID/Name	Notes	Creation Time	Operation
task-tmpl-ckgudkbg hello	echo 'hello world'	2018-11-30 12:17:15	Delete

Submitting job

1. Select **Job** on the left sidebar and select the target region at the top of the page, such as "Guangzhou".

2. Click **Create** to enter the "Create Job" page and configure job information as shown below:



The screenshot shows a form with three input fields:

- Job name:** A text input field containing the value "hello".
- Priority:** A text input field containing the value "0". Below this field is a note: "It should range from 0 to 100. A higher value means a higher priority".
- Description:** A larger text input field containing the value "hello job".

- **Job Name:** custom name, such as `hello` .
 - **Priority:** keep the default value.
 - **Description:** custom description, such as `hello demo` .
3. Select the "hello" task on the left on the "Task Flow" page and drag the task 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. Click and press "Delete" to delete the element.

Task Template

hello

hello

Completed Cancel

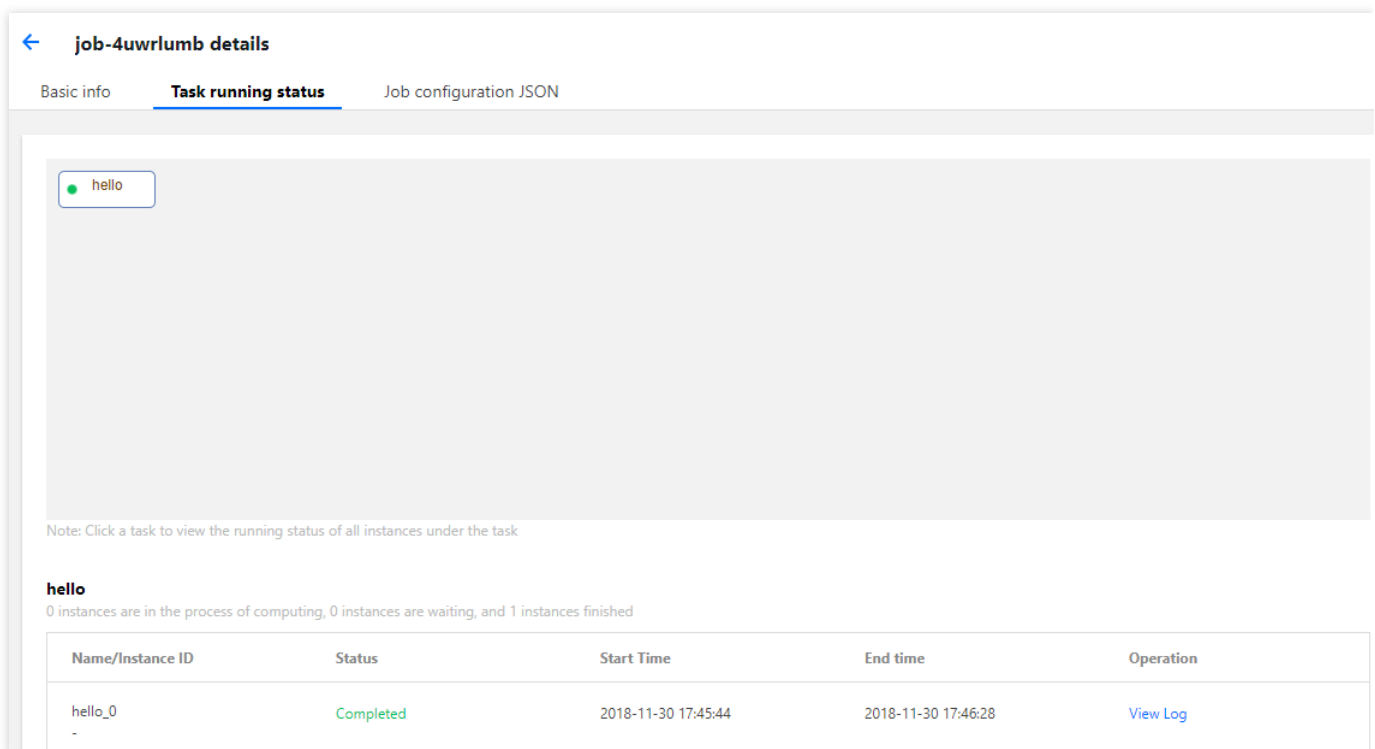
4. In "Task Details" on the right on the "Task Flow" page, confirm that the configuration is correct, and click **Complete**.

5. You can view the running status of a job on the "Job" list page as shown below:

ID/Name	Status	Completed/Total Tasks	Start Time	End time	Operation
job-4uwr1umb hello	Success	1/1	2018-11-30 17:45:03	2018-11-30 17:46:28	Delete

- Click a job ID to view the running status of each task instance in the **Task Running Status** tab.

- Click **Query Log** to view the standard output and standard error of a task instance as shown below:



job-4uwrlumb details

Basic info **Task running status** Job configuration JSON

hello

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	Operation
hello_0	Completed	2018-11-30 17:45:44	2018-11-30 17:46:28	View Log

What's Next?

In this simplest example, the job only contains a single task. It does not use remote storage mapping, but only shows the most basic capabilities. You can continue to test the advanced capabilities of BatchCompute as instructed in 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.
- **Remote code package execution:** technically, BatchCompute can fully satisfy your business needs by combining **custom image, remote code package, and command line**.
- **Remote storage mapping:** BatchCompute optimizes storage access and simplifies access to remote storage services into operations in the local file system.