

Alibaba Cloud Resource Orchestration Service

User Guide

Issue: 20200604









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 due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 contact Alibaba Cloud directly if you discover any errors in this document.

Document conventions

Style	Description	Example
	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.
	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.
	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.
	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 <code>cd /d C:/window</code> command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
{ } or {a b}	This format is used for a required value, where only one item can be selected.	switch {active stand}

Contents

Legal disclaimer.....	I
Document conventions.....	I
1 Overview.....	1
2 Resource Type Index.....	2
3 View resource types.....	9
4 Use Aliyun Serverless VSCode Extension to create a template..	11
5 Stack groups.....	13
5.1 Overview.....	13
5.2 Grant permissions on stack group operations.....	17
5.3 Create a stack group.....	20
5.4 Add a stack instance.....	22
5.5 Update a stack group.....	24
5.6 Override stack group parameter values.....	25
5.7 Delete a stack group.....	26
5.8 Status codes for stack groups and stack instances.....	27
6 Template syntax.....	29
6.1 Template structure.....	29
6.2 Parameters.....	31
6.3 Resources.....	36
6.4 Outputs.....	40
6.5 Functions.....	41
6.6 Mappings.....	69
6.7 Conditions.....	70
6.8 Pseudo parameters.....	73
6.9 Metadata.....	75
7 Custom resources.....	78
7.1 Overview.....	80
7.2 Resource references.....	82
7.2.1 Custom resource references.....	83
7.2.2 Custom resource request objects.....	83
7.2.3 Custom resource response objects.....	87
7.3 Request types.....	88
7.3.1 Custom resource request types.....	89
7.3.2 Create.....	89
7.3.3 Update.....	92
7.3.4 Delete.....	96
8 Change set management.....	100
8.1 Data structure.....	100

8.2 Create a stack.....	107
8.3 Update a stack.....	108
8.3.1 Overview.....	108
8.3.2 Create a change set.....	110
8.3.3 View a change set.....	115
8.3.4 Execute a change set.....	118
8.3.5 Delete a change set.....	119
9 Drift detection.....	121
9.1 Overview.....	121
9.2 Detect drift on a stack.....	126
9.3 Detect drift on a resource.....	131
9.4 Resource types that support drift detection.....	133
9.5 Detect drift on a stack group.....	133
9.6 Correct drift on a stack.....	137

1 Overview

Resource Orchestration Service (ROS) allows you to use templates to create, configure, and manage a group of resources.

To define a template that requires information about resources, you can view Alibaba Cloud resources and resource types supported by ROS on the **Resource Types** page in the ROS console. For more information, see the following topic:

- [View resource types](#)
- [Resource Type Index](#)

For details about the template syntax, see the following topics:

- [Template structure](#)
- [Parameters](#)
- [Resources](#)
- [Outputs](#)
- [Functions](#)
- [Mappings](#)
- [Conditions](#)

Use a template to create stacks

If you have not created a stack template, select one from the sample templates in the ROS console to create a stack based on your needs. For more information, see [Create resource stacks using a template](#).

If you have already created a stack template, select your template on the **My Templates** page in the ROS console. ROS will create all the resources and their dependencies that are defined in the template.

2 Resource Type Index

This topic provides a resource type index for you to query.

API Gateway	<ul style="list-style-type: none">• #unique_14: Create an API.• #unique_15: Create an application.• #unique_16: Authorize the access permission of the application to the API.• #unique_17: Create a custom domain name for the API Group.• #unique_18: Publish an API or switch the API version.• #unique_19: Create an API Group.• #unique_20: Create a backend signature key.• #unique_21: Bind API and backend signature key• #unique_22: Create variables for the test, pre-release, and production environments of the API Group.• #unique_23: Create a user-defined traffic control policy.• #unique_24: Bind user-defined traffic control to the API• #unique_25: Configure VPC authorization so that VPC APIs can provide external services.	
ActionTrail	<ul style="list-style-type: none">• #unique_26: Stores audit data to a specified OSS bucket.• #unique_27: Enable or disable logging for a trail.	
BSS	#unique_28 : Wait until the order ends	
CDN	<ul style="list-style-type: none">• #unique_29: Add a CDN domain name• #unique_30: Batch domain name configuration	

CEN	<ul style="list-style-type: none">• #unique_31: Create a CEN instance• #unique_32: Attach a network to a CEN instance• #unique_33: Create a bandwidth package.• #unique_34: Bind the bandwidth package to the specified Cen instance• #unique_35: Set the cross-region interconnection bandwidth in the bandwidth package.• #unique_36: Publish the route entry of the VPC or VBR attached to the CEN instance to the CEN instance.
CLOUDFW	<ul style="list-style-type: none">• #unique_37: Adds an IP address book, ECS tag address book, port address book, and domain address book.• #unique_38: Add an access control policy
CMS	<ul style="list-style-type: none">• #unique_39: Create an application Group.• #unique_40: Create or modify an Event Alarm rule.• #unique_41: Add or modify the sending target of the rule.
DNS	<ul style="list-style-type: none">• #unique_42: Add a domain name.• #unique_43: Add a Domain Group.• #unique_44: Add a DNS record.

ECS	<ul style="list-style-type: none">• #unique_45: Automatic snapshot policy• #unique_46: The bandwidth package used to create the NAT gateway.• #unique_47: Create a cloud Assistant Command• #unique_48: Copy a custom image to another region• #unique_49: Create a custom image• #unique_50: Create a DDH.• #unique_51: Create a deployment set.• #unique_52: Create a data disk.• #unique_53: Associate the data disk with the corresponding ECS• #unique_54: Configure the forwarding table in the NAT gateway.• #unique_55: Create a single ECS instance.• #unique_56: Create a single ECS instance by cloning an ECS instance.• #unique_57: Create one or more ECS instances.• #unique_58: Clone one or more ECS instances based on an ECS instance.• #unique_59: Trigger a cloud assistant command for the ECS instance• #unique_60: Creates an ECS instance launch template.• #unique_61: Attach an Eni to a VPC• #unique_62: Authorizes an Eni.• #unique_63: Create a subscription ECS instance• #unique_64: Clone a group of subscription ECS instances.• #unique_65: Configure the SNAT table in the NAT gateway.• #unique_66: Configure the traffic ingress rule of the security group.• #unique_67: Creates a disk snapshot.• #unique_68: Create an SSH key pair• #unique_69: Bind an SSH key pair to an ECS instance• #unique_70: Create a VPC• #unique_71: Create a vSwitch in the VPC.• #unique_72: Assign an IPv6 address to the Eni• #unique_73: Assign secondary private IP addresses to the Eni.
-----	---

EMR	#unique_74 : Create an E-MapReduce cluster.
ESS	<ul style="list-style-type: none"> • #unique_75: Create a metric alarm task. • #unique_76: Create lifecycle hooks for scaling groups. • #unique_77: Create a scaling configuration. • #unique_78: Create a scaling Group • #unique_79: Enables a scaling group. • #unique_80: Create a scaling rule • #unique_81: Creates a scheduled task.
FC	<ul style="list-style-type: none"> • #unique_82: Create a custom domain name • #unique_83: Create a function. • #unique_84: Create a function Compute Service. • #unique_85: Trigger the function.
FOAS	<ul style="list-style-type: none"> • #unique_86: Create a cluster for an exclusive mode order • #unique_87: Create a project in the cluster
KMS	<ul style="list-style-type: none"> • #unique_88: Create a CMK. • #unique_89: Create an alias for the CMK
MarketPlace	#unique_90 : Buy Marketplace resources.
MNS	<ul style="list-style-type: none"> • #unique_91: Describes the subscription relationship. • #unique_92: Create a topic.
MongoDB	<ul style="list-style-type: none"> • #unique_93: Create an apsaradb for MongoDB instance.
NAS	<ul style="list-style-type: none"> • #unique_94: Create a permission group. • #unique_95: Create a permission rule. • #unique_96: Create a file system. • #unique_97: Create a mount point.
OOS	<ul style="list-style-type: none"> • #unique_98: Starts an execution. • #unique_99: Create a template.
OSS	#unique_100 : Create an OSS bucket.
OTS	<ul style="list-style-type: none"> • #unique_101: Create a table store instance. • #unique_102: Bind a table store instance to a VPC • #unique_103: Create a table.

PVTZ	<ul style="list-style-type: none"> • #unique_104: Create a private zone • #unique_105: Add a record for the private zone. • #unique_106: Associate or disassociate the private Zone from the VPC list.
RAM	<ul style="list-style-type: none"> • #unique_107: Obtain the AccessKey of the specified user • #unique_108: Attach an authorization policy to a specified role • #unique_109: Create a RAM Group. • #unique_110: Create a RAM policy • #unique_111: Create a RAM role • #unique_112: Add a subaccount to a RAM Group
RDS	<ul style="list-style-type: none"> • #unique_113: Create an account for database management • #unique_114: Authorizes an account to access the database. • #unique_115: Create a database instance. • #unique_116: Modifies a database parameter list. • #unique_117: Modify the instance access whitelist • #unique_118: Create a subscription database instance. • #unique_119: Create a read-only instance for an instance.
Redis	<ul style="list-style-type: none"> • #unique_120: Create an apsaradb for Redis instance. • #unique_121: Create a subscription apsaradb for Redis instance. • #unique_122: Set the IP address whitelist of the apsaradb for Redis instance.
ROS	<ul style="list-style-type: none"> • #unique_123: Create a custom resource. • #unique_124: Create an instance for processing UserData messages. • #unique_125: Create an instance for sending and receiving messages during UserData execution. • #unique_126: Create a nested resource stack.

SAG	<ul style="list-style-type: none"> • #unique_127: Bind an ACL to a Smart Access Gateway instance • #unique_128: Add an access control rule • #unique_129: Create a CCN instance • #unique_130: Create an access control. • #unique_131: Attach the Smart Access Gateway instance to a CCN instance
SLB	<ul style="list-style-type: none"> • #unique_132: Create an access control list • #unique_133: Add backend servers like server load balancer • #unique_134: Add backend servers to a VServer Group • #unique_135: Upload certificate • #unique_136: Create a domain name extension. • #unique_137: Create a server load balancer listener • #unique_138: Create a server load balancer instance • #unique_139: Clone an SLB instance • #unique_140: Create an active/standby server group • #unique_141: Add forwarding rules for HTTP or HTTPS listeners • #unique_142: Create a virtual server group and add backend servers to the server load balancer instance.
SLS	<ul style="list-style-type: none"> • #unique_143: Apply the log configuration of log service to the Machine Group. • #unique_144: Create an index for a specified Logstore • #unique_145: Create a logstore under a log project • #unique_146: Create a log service Machine Group • #unique_147: Create a log project
UIS	<ul style="list-style-type: none"> • #unique_148: Create a UIS instance. • #unique_149: Create a tunnel connection • #unique_150: Add an access point instance for a created instance

VPC	<ul style="list-style-type: none">• #unique_151: Bind an EIP• #unique_152: Bind two router interfaces to be interconnected• #unique_153: Initiate a router interface connection• #unique_154: Create a router interface.• #unique_155: Create a custom route table• #unique_156: Bind the custom route table to the vSwitch in the same VPC.• #unique_157: Add an SNAT entry to the SNAT table.
POLARDB	<ul style="list-style-type: none">• #unique_158: Create a POLARDB cluster• #unique_159: Create a database under the POLARDB cluster• #unique_160: Add POLARDB cluster nodes• #unique_161: Modify the IP address list that is allowed to access the database cluster.• #unique_162: Create an account for the POLARDB database.• #unique_163: Authorize a regular account

3 View resource types

You can view supported resource types and their details in the Resource Orchestration Service (ROS) console. You can create stack templates based on the properties of resource types and specify specific requirements for resources.

Procedure

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Resource Types**.

On the **Resource Types** page, you can view all supported resource types.

3. Click **Type** or **View Details** in the **Actions** column corresponding to a resource type to view **Properties**, **Response Parameters**, and **Sample Template**.

- **Properties**

The **Properties** tab shows configuration options that you can set for a resource when creating a stack template. ECS instance types, images, and security groups are examples of properties. The following table describes the metadata of each property.

Property	Description
key	The name of the property.
type	The data type of the property.
required	Specifies whether the property is required.
immutable	Specifies whether the property is immutable.
update_allowed	Specifies whether the property can be updated. If the property of a created resource instance can be updated, you can change the property value in the template.
description	The detailed description of the property.
constraints	The value range of this property.

Property	Description
schema	The sub-properties of the property.

- **Response Parameters**

The **Response Parameters** tab shows property values that you can obtain after resources are created in the ROS console. ECS instance ID, private IP address, and public IP address are examples of response parameters.

- **Sample Template**

The **Sample Template** tab shows the sample template for creating resources in the ROS console.

4 Use Aliyun Serverless VSCode Extension to create a template

This topic describes how to use Aliyun Serverless VSCode Extension to create a template.

The Visual Studio Code (VS Code) extension provides syntax prompts for ROS template files. It also allows you to develop and debug Function Compute functions on your local PC. For more information, see [serverless-vscode](#). You must install [Visual Studio Code](#) to use the VS Code extension.

Syntax prompts:

- Automatic completion: Resource property configurations in a template file are completed automatically based on prompts provided based on the indentation level.
- Configuration validation: All resource configurations in a template file are validated based on the specifications in [Serverless Application Model](#).
- Hovering prompts: Context-sensitive help is provided for all resource configurations in a template file. When you move the pointer over a resource key, property information such as property name, type, and document URL is displayed.

Install the extension

1. Open VS Code and use the shortcut key Ctrl+Shift+X to go to the extension marketplace.
2. Search for **Aliyun Serverless** and install the extension.
3. Restart VS Code.

If the **ALIYUN:FUNCTION COMPUTE** icon is displayed in the left-side activity bar, the extension is installed.

Bind an Alibaba Cloud account

1. Start **ALIYUN:FUNCTION COMPUTE**.
2. Click **Bind New Account** under the **REMOTE RESOURCES** option. In the text box that appears, enter the account ID, AccessKey ID, AccessKey secret, and local name of the account in sequence.
3. After the account is bound, the name and region are displayed in the lower-left corner, which indicates that the account is activated.

Create a template

1. Open a folder in VS Code, and the extension is still in the inactive state. Click the **ALIYUN: FUNCTION COMPUTE** icon to activate the extension.
2. Create a `template.yml` file.
3. Enter `ros` in the first line, and press the Enter key twice. The system prompts you for the required parameters. Select **Resources**, or enter the initial letters and then a list of options are provided.
4. If you enter keywords of a resource type under **Resources**, all related resources appear. For example, if you enter `ECS`, all resources related to `ECS` appear. After you select a resource type, a schema is generated. When you move the pointer over **Properties**, all properties and data types of the resource type appear. You can search for the names of properties by fuzzy match. When required properties are missing, alerts are triggered for **Properties**, and prompts are displayed in the console.

5 Stack groups

5.1 Overview

Stack groups extend the functionality of stacks by enabling you to manage stacks across multiple accounts and regions with a single operation. Using an administrator account, you can define and manage an ROS template, and use the template as the basis for provisioning stacks into selected target accounts across specified regions. Stack groups simplify workflows and minimize maintenance costs.

Stack groups

A stack group lets you use a single ROS template to create stacks in multiple Alibaba Cloud accounts across regions. All resources included in each stack are defined by the ROS template of the stack group. When creating the stack group, you must specify the template to use, as well as any parameters and capabilities that the template requires.

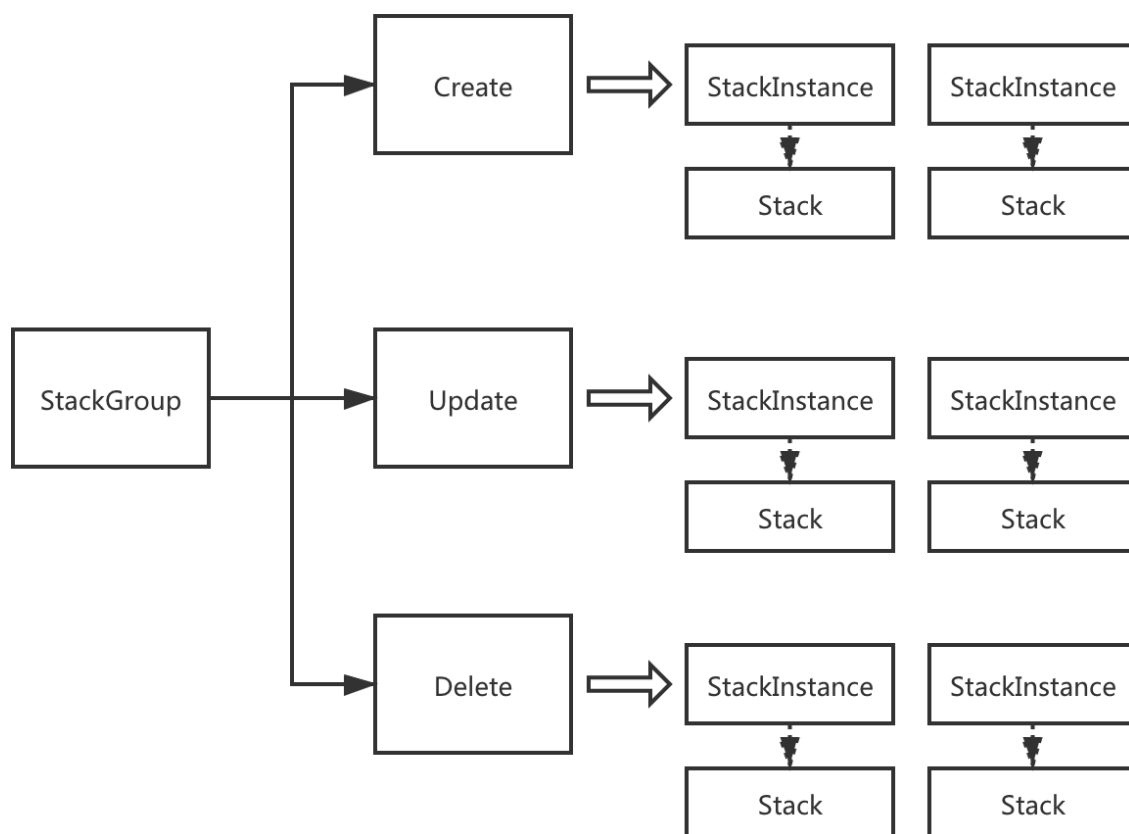
After defining a stack group, you can create, update, or delete stacks in the target accounts and regions you specify. When you create, update, or delete stacks, you can also specify operation preferences. For example, you can specify the number of accounts in which operations are performed on stacks concurrently, the order of regions in which you want the operations to be performed, and the failure tolerance beyond which stack operations stop.

A stack group is a regional resource. If you create a stack group in a region such as China (Hangzhou), you cannot see it or change it in other regions.

Stack instances

A stack instance is a reference to a stack in a target account within a region. A stack instance corresponds to a stack. However, a stack instance can exist without a stack. For example, if the stack could not be created for some reason, the stack instance shows the reason for stack creation failure. A stack instance can only belong to a single stack group.

The following figure shows the logical relationships among stack groups, stack operations, and stacks. When you update a stack group, all associated stack instances are updated throughout all accounts and regions.



Accounts

The accounts required for using a stack group are as follows:

- Administrator account

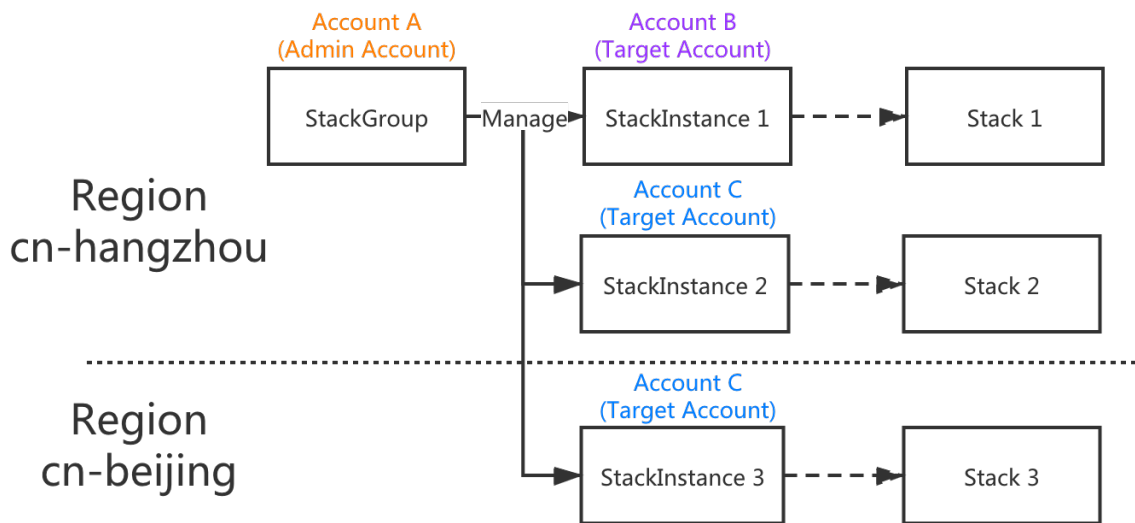
An administrator account is the Alibaba Cloud account in which you create stack groups. After logging on to the Resource Orchestration Service (ROS) console with the administrator account, you can create a stack group and its stack instances. The stack corresponding to each stack instance is created in a target account within a region.

- Target account

A target account is the account into which you create, update, or delete one or more stacks in your stack group. Before you can use a stack group to create stacks in a target account, you must set up a trust relationship between the administrator and target accounts. An administrator account can also be a target account.

Relationships among stack groups, stack instances, stacks, regions, and accounts

The following figure shows the relationships among stack groups, stack instances, stacks, regions, and accounts.



- Stack groups are region-specific. For example, stack groups created in the China (Hangzhou) region cannot be viewed in the China (Beijing) region.
- Stack groups can be used to provision resources across multiple accounts and regions. Assume that a stack group is created by using an administrator account (Account A). You can create multiple stack instances in the stack group. When creating stack instances, you can specify the target accounts and regions. In the stack group created by using Account A, you can create Stack Instance 1 for Account B and Stack Instance 2 for Account C within the China (Hangzhou) region, and create Stack Instance 3 for Account C within the China (Beijing) region.
- A stack instance is a reference to a stack.
 - A stack is created when you create a stack instance.
 - When deleting a stack instance, you can choose to delete or retain its corresponding stack.
 - Deleting a stack will not affect the stack instance.

Stack group deployment options

When creating or updating a stack group, adding a stack instance, or deleting stacks from a stack group, you can configure the following stack group parameters:

- MaxConcurrentCount/MaxConcurrentPercentage

You can specify the maximum number or percentage of target accounts in which an operation is performed at one time. For example, if you are deploying stacks to five target accounts within two regions, setting MaxConcurrentCount to 3 or MaxConcurr

entPercentage to 60 will deploy stacks to three accounts within the first region, then the other two accounts within the first region, before moving on to the next region and performing the same operation.

If the specified MaxConcurrentPercentage does not represent a whole number of your specified accounts, ROS rounds down. For example, if you are deploying stacks to five target accounts, and you set MaxConcurrentPercentage to 50, ROS deploys two stacks concurrently.

- FailureToleranceCount/FailureTolerancePercentage

You can specify the maximum number or percentage of stack operation failures that can occur per region, beyond which ROS stops an operation automatically. For example, if you are creating stacks in five target accounts within two regions, setting FailureToleranceCount to 2 or FailureTolerancePercentage to 40 means that a maximum of two stack creation failures can occur in a region for the operation to continue. If stack creation fails in a third target account within the same region, ROS stops the operation. If the number of target accounts in which stack creation fails does not exceed 2 in both regions, ROS considers the operation successful.

If the specified FailureTolerancePercentage does not represent a whole number of your specified accounts, ROS rounds down. For example, if you are deploying stacks to five target accounts, and you set FailureTolerancePercentage to 50, ROS allows a maximum of two target accounts in which stack creation fails.

Configuring the preceding parameters helps to control the time and number of failures allowed to successfully perform stack group operations, and prevent you from losing stack resources.

Related operations

The following table lists stack group operations.

Operation	Description
Grant permissions on stack group operations	Before using a stack group, you must create the necessary RAM roles and grant required permissions to ROS.

Operation	Description
Create a stack group	When creating a stack group, you must specify an ROS template that you want to use to create stacks, the target accounts in which you want to create stacks, and the regions in which you want to deploy stacks to your target accounts. A stack group ensures consistent deployment of identical stack resources with identical settings to all specified target accounts within the regions you choose.
Add a stack instance	In a stack group, you can add stack instances by specifying target accounts and regions.
Update a stack group	You can update a stack group to modify its template, parameter settings, administrator role, target accounts, and regions.
Override stack group parameter values	To modify the parameter values for stacks in a stack group, you can override the values specified in the stack group.
Delete a stack group	When you no longer need a stack group and its stacks, you can delete the stacks and all of their associated resources from the specified target accounts within the specified regions.

5.2 Grant permissions on stack group operations

ROS deploys stacks corresponding to stack instances in a stack group by assuming roles. Before you use a stack group, create the necessary RAM roles and grant required permissions to ROS.

Context

Before you use a stack group, set permissions for the following accounts:

- An administrator account

An administrator account is the Alibaba Cloud account in which you create stack groups.

- A target account

A target account is the account into which you create stacks in your stack group.

To set account permissions, perform the following steps:

- Set permissions in the console
 - [Set administrator account permissions in the console](#)
 - [Set target account permissions in the console](#)
- [Use ROS templates to set permissions](#)

Set administrator account permissions in the console

Create a role named **AliyunROSStackGroupAdministrationRole** for the administrator account, and grant permissions specified in the **AssumeRole-AliyunROSStackGroupExecutionRole** policy.

1. Create a role named **AliyunROSStackGroupAdministrationRole** and grant permissions to ROS.
 - a) Log on to the [RAM console](#) with an Alibaba Cloud account.
 - b) In the left-side navigation pane, click **RAM Roles**.
 - c) On the RAM Roles page, click **Create RAM Role**.
 - d) Select **Alibaba Cloud Service** as the trusted entity type, and click **Next**.
 - e) Set **RAM Role Name** to **AliyunROSStackGroupAdministrationRole** and **Select Trusted Service** to **Resource Orchestration Service**.
 - f) Click **OK**.
2. Create a permission policy named **AssumeRole-AliyunROSStackGroupExecutionRole**.
 - a) In the left-side navigation pane, choose **Permissions > Policies**.
 - b) On the Policies page, click **Create Policy**.
 - c) Set **Policy Name** to **AssumeRole-AliyunROSStackGroupExecutionRole** and **Configuration Mode** to **Script**.

In the **Policy Document** section, enter the following policy content to allow the **AliyunROSStackGroupAdministrationRole** role to assume the **AliyunROSStackGroupExecutionRole** role:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "acs:ram::*:role/AliyunROSStackGroupExecutionRole"
    }
  ],
  "Version": "1"
}
```

- d) Click **OK**.

3. Grant the permissions in the **AssumeRole-AliyunROSStackGroupExecutionRole** policy statement to **AliyunROSStackGroupAdministrationRole**.

- a) In the left-side navigation pane, click **RAM Roles**.
- b) In the **RAM Role Name** column, click **AliyunROSStackGroupAdministrationRole**.
- c) On the basic information page of **AliyunROSStackGroupAdministrationRole**, click **Add Permissions**.
- d) In the **Add Permissions** pane, set **Principal** to **AliyunROSStackGroupAdministrationRole** and **Custom Policy** to **AssumeRole-AliyunROSStackGroupExecutionRole**.
- e) Click **OK**.

Set target account permissions in the console

Create a role named **AliyunROSStackGroupExecutionRole** for the target account, and grant permissions specified in the **AdministratorAccess** policy. This set of permissions is required when you use a stack group to create stacks in a target account.

1. Create a role named **AliyunROSStackGroupExecutionRole**.

- a) Log on to the [RAM console](#).
- b) In the left-side navigation pane, click **RAM Roles**.
- c) On the RAM Roles page, click **Create RAM Role**.
- d) Select **Alibaba Cloud Account** as the trusted entity type, and then click **Next**.
- e) Set **RAM Role Name** to **AliyunROSStackGroupExecutionRole** and **Select Trusted Alibaba Cloud Account** to **Other Alibaba Cloud Account**, and enter the administrator account ID.
- f) Click **OK**.

2. Grant permissions specified in the **AdministratorAccess** policy to **AliyunROSStackGroupExecutionRole**.

- a) In the left-side navigation pane, click **RAM Roles**.
- b) In the **RAM Role Name** column, click **AliyunROSStackGroupExecutionRole**.
- c) On the basic information page of **AliyunROSStackGroupExecutionRole**, click **Add Permissions**.
- d) In the **Add Permissions** pane, set **Principal** to **AliyunROSStackGroupExecutionRole** and **System Policy** to **AdministratorAccess**.
- e) Click **OK**.

Use ROS templates to set permissions

You can use ROS templates to create execution roles for the administrator and target accounts, and grant operation permissions on stack groups and stacks.

1. Log on to the [ROS console](#).
2. Use the [AliyunROSStackGroupAdministrationRole](#) template to create an administrator role and permissions for the administrator account.
3. Use the [AliyunROSStackGroupExecutionRole](#) template to create an execution role and permissions for the target account.

5.3 Create a stack group

A stack group lets you use a single ROS template to create stacks in multiple Alibaba Cloud accounts across regions. All resources included in each stack are defined by the ROS template of the stack group.

Prerequisites

Ensure that you have set permissions for the administrator and target accounts. For more information, see [Grant permissions on stack group operations](#).

Context

When creating a stack group, you must specify an ROS template that you want to use to create stacks, the target accounts in which you want to create stacks, and the regions in which you want to deploy stacks to your target accounts. A stack group ensures consistent deployment of identical stack resources with identical settings to all specified target accounts within the regions you choose.

Procedure

1. Prepare a template.

Before creating a stack group, you must prepare an ROS template and define the resources to be created. The following example provides a template that creates a VPC, VSwitch, security group, and ECS instance:

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "InstanceType": {
      "Type": "String",
      "Default": "ecs.g6.large"
    }
  },
  "Resources": {
```

```

    "Vpc": {
      "Type": "ALIYUN::ECS::VPC",
      "Properties": {
        "CidrBlock": "192.168.0.0/16"
      }
    },
    "VSwitch": {
      "Type": "ALIYUN::ECS::VSwitch",
      "Properties": {
        "CidrBlock": "192.168.0.0/24",
        "VpcId": {
          "Ref": "Vpc"
        }
      },
      "ZoneId": {
        "Fn::Select": [
          "0",
          {
            "Fn::GetAZs": {
              "Ref": "ALIYUN::Region"
            }
          }
        ]
      }
    },
    "SecurityGroup": {
      "Type": "ALIYUN::ECS::SecurityGroup",
      "Properties": {
        "VpcId": {
          "Ref": "Vpc"
        }
      }
    },
    "InstanceGroup": {
      "Type": "ALIYUN::ECS::InstanceGroup",
      "Properties": {
        "MaxAmount": 2,
        "InstanceType": {
          "Ref": "InstanceType"
        },
        "ImageId": "centos_7",
        "VpcId": {
          "Ref": "Vpc"
        },
        "VSwitchId": {
          "Ref": "VSwitch"
        },
        "SecurityGroupId": {
          "Ref": "SecurityGroup"
        }
      }
    },
    "Outputs": {
      "InstanceIds": {
        "Value": {
          "Fn::GetAtt": ["InstanceGroup", "InstanceIds"]
        }
      }
    }
  }
}

```

2. Log on to the [Resource Orchestration Service \(ROS\) console](#).

3. Select the region where you want to create the stack group from the drop-down list in the upper-left corner.
4. In the left-side navigation pane, click **Stack Groups**.
5. On the **Stack Groups** page, click **Create Stack Group**.
6. In the **Select Template** step of the **Create Stack Group** wizard, enter the ROS template content and click **Next**.

You can select an existing template or use a sample template.

A template is a JSON or YAML file encoded in UTF-8. For more information about templates, see [Template structure](#).

7. In the **Configure Template Parameters** step of the **Create Stack Group** wizard, set **Stack Group Name** and **Parameters** as prompted, and click **Next**.
8. In the **Configure Stack Group** step of the **Create Stack Group** wizard, set **Admin Role** and **Execution Role** as prompted, and click **Next**.

If you have set role permissions, click **Next**.

9. In the **Set Deployment Options** step of the **Create Stack Group** wizard, configure the accounts, regions, maximum number of concurrent accounts, and fault tolerance for the stack group as prompted, and click **Next**.

For information about how to set **Maximum Number of Concurrent Accounts** and **Fault Tolerance**, see [Stack group deployment options](#).

10. In the **Confirm** step of the **Create Stack Group** wizard, click **Create Stack Group**.

You can view information about the created stack group on the **Stack Groups** page.

5.4 Add a stack instance

To add a stack instance to a stack group, you must specify the target account and region. A stack instance can only belong to a single stack group.


Prerequisites

Ensure that you have created a stack group. For more information, see [Create a stack group](#).

Context

To add a stack to a stack group, you must first create a stack instance in the stack group, and then create the stack in the specified target account and region. Each stack instance corresponds to a stack.

Procedure

1. Log on to the [Resource Orchestration Service \(ROS\) console](#).
2. Select the region where the target stack group resides from the drop-down list in the upper-left corner.
3. In the left-side navigation pane, click **Stack Groups**.
4. On the **Stack Groups** page, click  in the Actions column corresponding to the stack group, and choose **Add Stack to Stack Group** from the shortcut menu.
5. In the **Set Deployment Options** step of the **Add Stack to Stack Group** wizard, configure the accounts, regions, maximum number of concurrent accounts, and fault tolerance for the stack group as prompted, and click **Next**.

For information about how to set **Maximum Number of Concurrent Accounts** and **Fault Tolerance**, see [Stack group deployment options](#).

6. In the **Specify Override Settings** step of the **Add Stack to Stack Group** wizard, select whether to override the parameter values as prompted, and click **Next**.

When adding a stack to a stack group, you can re-define the parameter values in the template. After the stack is created, go to the **Stacks** page. Click the stack ID in the **Stack Name** column, and then click the **Parameters** tab to view the re-defined parameter values.

7. In the **Confirm** step of the **Add Stack to Stack Group** wizard, configure the override values of the stack instance as prompted, and click **Add**.
8. View the stack instance.
 - a) On the **Stack Groups** page, click the stack group name.
 - b) On the stack group management page, click the **Instances** tab.

You can view the account, region, ID, and status of the stack instance.

After the stack instance is added, **Expired** is displayed in the **Instance Status** column because the stack is being created. After the stack is created, **Latest** is displayed in the **Instance Status** column. On the **Stacks** page, you can view the stack creation status.

5.5 Update a stack group

You can update a stack group to modify its template, parameter settings, administrator role, target roles, fault tolerance, and number of concurrent accounts.

Prerequisites

Ensure that you have created a stack group. For more information, see [Create a stack group](#).

Context

You can update the template and parameter values of a stack group. If you update a stack group template, all associated stacks will be updated.

Procedure

1. Log on to the [Resource Orchestration Service \(ROS\) console](#).
2. Select the region where the target stack group resides from the drop-down list in the upper-left corner.
3. In the left-side navigation pane, click **Stack Groups**.
4. On the **Stack Groups** page, click **Update** in the Actions column corresponding to the stack group.
5. In the **Select Template** step of the **Modify Stack Group** wizard, select and configure a template as prompted, and click **Next**.

If you select **Replace Current Template**, the stack group template and all stacks in the stack group will be updated. If you only need to update some stacks, specify the target accounts and regions in the **Set Deployment Options** step.
6. In the **Configure Template Parameters** step of the **Modify Stack Group** wizard, set **Stack Group Description** and **Parameters** as prompted, and click **Next**.
7. In the **Configure Stack Group** step of the **Modify Stack Group** wizard, set **Admin Role** and **Execution Role** as prompted, and click **Next**.
8. In the **Set Deployment Options** step of the **Modify Stack Group** wizard, configure the accounts, regions, maximum number of concurrent accounts, and fault tolerance for the stack group as prompted, and click **Next**.

For information about how to set **Maximum Number of Concurrent Accounts** and **Fault Tolerance**, see [Stack group deployment options](#).
9. In the **Confirm** step of the **Modify Stack Group** wizard, click **Confirm**.

10. Click **Confirm.**

To confirm the update configuration, click **Next**. After you confirm that the configuration is correct, click **Confirm** to update the stack group.


5.6 Override stack group parameter values

To modify the parameter values for stacks in a stack group, you can override the values specified in the stack group.

Prerequisites

Ensure that you have created a stack group. For more information, see [Create a stack group](#).

Procedure

1. Log on to the [ROS console](#).
2. In the upper-left corner, select the region where the target stack group resides from the drop-down list.
3. In the left-side navigation pane, click **Stack Groups**.
4. On the **Stack Groups** page, click  in the Actions column corresponding to the stack group, and choose **Override Stack Group Parameters** from the shortcut menu.
5. In the **Set Deployment Options** step of the **Override Stack Group Parameter Value** wizard, configure the accounts, regions, maximum number of concurrent accounts, and fault tolerance for the stack group, and click **Next**.

For information about how to set **Maximum Number of Concurrent Accounts** and **Fault Tolerance**, see [Stack group deployment options](#).
6. In the **Specify Override Settings** step of the **Override Stack Group Parameter Value** wizard, configure the override values of the parameters, and click **Next**.
7. In the **Confirm** step of the **Override Stack Group Parameter Value** wizard, click **Override**.

On the **Stacks** page, you can click a stack ID in the **Stack Name** column, and then click the **Parameters** tab to view the override values of the stack group.

5.7 Delete a stack group

When you no longer need a stack group and its stacks, you can delete the stack instances and all of their associated resources from the specified target accounts within the specified regions.

Prerequisites

Ensure that you have created a stack group. For more information, see [Create a stack group](#).


Context

- A stack group can be deleted only when the stack group does not contain any stack instances.
- Deleting a stack will not affect the stack instance.
- When deleting a stack instance, you can select whether to delete the stack.

Procedure

1. Log on to the [Resource Orchestration Service \(ROS\) console](#).
2. Select the region where the target stack group resides from the drop-down list in the upper-left corner.
3. In the left-side navigation pane, click **Stack Groups**.

4. Delete the stack instances in the stack group.

- a) On the **Stack Groups** page, click  in the Actions column corresponding to the stack group, and choose **Delete Stack from Stack Group** from the shortcut menu.
- b) In the **Set Deployment Options** step of the **Delete Stack from Stack Group** wizard, configure the accounts, regions, maximum number of concurrent accounts, and fault tolerance for the stack group, and click **Next**.
- c) In the **Confirm** step of the **Delete Stack from Stack Group** wizard, confirm the accounts, regions, maximum number of concurrent accounts, and fault tolerance for the stack group, and click **Delete**.

For information about how to set **Maximum Number of Concurrent Accounts** and **Fault Tolerance**, see [Stack group deployment options](#).

- d) In the confirmation dialog box that appears, select the method to delete a stack, and click **OK**.
 - If you set **Method to Delete the Stack** to **Release Resources**, the stack will be deleted. In this case, all resources created in the stack will also be released. Use caution when performing this operation.
 - If you set **Method to Delete the Stack** to **Retain Resources**, the stack will not be deleted.

5. Delete the stack group.

On the **Stack Groups** page, click **Delete** in the Actions column corresponding to the stack group.

5.8 Status codes for stack groups and stack instances

ROS generates status codes for operations on stack groups and stack instances.

Stack group operations

Stack group operation status	Description
RUNNING	The operation is in progress.
SUCCEEDED	The operation has been completed and has not exceeded the operation's fault tolerance capability.
FAILED	The number of stacks on which the operation failed exceeded the user-defined failure tolerance.

Stack group operation status	Description
STOPPING	The operation is being stopped.
STOPPED	The operation has been stopped.

Stack instance operations

Stack instance operation status	Description
CURRENT	The stack is up to date with the stack group. The template and parameters of the stack are consistent with those of the stack group.
OUTDATED	<p>The stack is not up to date with the stack group. The template and parameters of the stack are inconsistent with those of the stack group. Examples:</p> <ul style="list-style-type: none">• The stack group and some stack instances have been updated. In this case, the status of stack instances that have not been updated is OUTDATED.• ROS failed to assume the required role to create or update a stack instance and the stack creation or update was not performed. In this case, the status of the stack instance is OUTDATED.• ROS assumed the required role to create or update a stack instance, but the stack creation or update failed. In this case, the status of the stack instance is OUTDATED.

6 Template syntax

6.1 Template structure

A template is a JSON file encoded using UTF-8. It is used to create stacks and serves as the blueprint for the underlying infrastructure and architecture. Templates define the configurations and dependencies of Alibaba Cloud resources.

ROS template structure

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",

  "Description" : "The description of the template, which is used to provide information
such as application scenarios and architecture of the template.",
  "metadata" : {
    // The template metadata that provides information such as layout for visualizations.
  },
  "Parameters" : {
    // The parameters you can specify when creating a stack.
  },

  "Mappings": {
    // The mapping tables. Mapping tables are nested tables.
  },

  "Conditions": {
    // The conditions defined using internal condition functions. These conditions
determine when to create associated resources.
  },

  "Resources" : {
    // The detailed information of resources, including configurations and dependencies.
  },

  "Outputs" : {
    // The outputs that are used to provide useful information such as resource properties
. You can use the ROS console or API to obtain this information.
  }
}
```

ROSTemplateFormatVersion

The template versions supported by ROS. Current version: 2015-09-01.

(Optional) Description

The description of the template, which is used to provide information such as the application scenarios and architecture of the template. A detailed description can help users better understand the content of the template.

(Optional) Metadata

The metadata of the template, in JSON format.

(Optional) Parameters

The parameters you can specify when creating a stack. An ECS instance type is often defined as a parameter. Parameters have default values. Parameters can improve the flexibility and reusability of the template. When you create a stack, select a suitable specification based on your needs.

For more information, see [Parameters](#).

(Optional) Mappings

Mappings are defined as nested mapping tables. You can use `Fn::FindInMap` to select values through corresponding keys. You can also use parameter values as keys. For example, you can search the region-image mapping table for desired images by region.

For more information, see [Mappings](#).

(Optional) Condition

The conditions defined using `Fn::And`, `Fn::Or`, `Fn::Not`, and `Fn::Equals`. Separate multiple conditions with commas (,). The system evaluates all the conditions in the template before creating or updating a stack. All resources associated with true conditions are created, and all resources associated with false conditions are ignored.

For more information, see [Conditions](#).

(Optional) Resources

The detailed information of resources in the stack that is created based on the template. This information includes resource dependencies and configurations.

For more information, see [Resources](#).

(Optional) Outputs

The outputs that are used to provide useful information such as resource properties. You can use the ROS console or API to obtain this information.

For more information, see [Outputs](#).

6.2 Parameters

The Parameters section improves the flexibility and reusability of a template. When creating a stack, you can specify parameter values as needed.

For example, a web application requires a stack that contains one Server Load Balancer (SLB) instance, two Elastic Compute Service (ECS) instances, and one ApsaraDB for RDS instance. If the web application has a heavy workload, you can select an ECS instance with a higher specification when creating the stack. Otherwise, you can select an ECS instance with a lower specification. The following example shows how to define the ECS instance type parameter:

```
"Parameters" : {
  "InstanceType" : {
    "Type" : "String",
    "AllowedValues":["ecs.t1.small","ecs.s1.medium", "ecs.m1.medium", "ecs.c1.large"],
    "Default": "ecs.t1.small",
    "Label": "The ECS instance type",
    "Description" : "The type of the ECS instance you want to create. Default value: ecs.t1.small. Valid values: ecs.t1.small, ecs.s1.medium, ecs.m1.medium, and ecs.c1.large."
  }
}
```

The value of the InstanceType parameter can be specified when you create stacks based on templates. If this parameter is not specified, the default value `ecs.t1.small` is used.

The following example shows how to reference the InstanceType parameter when you define a resource:

```
"Webserver" : {
  "Type" : "ALIYUN::ECS::Instance",
  "InstanceType": {
    "Ref": "InstanceType"
  }
}
```

Syntax

Each parameter consists of a name and properties. The parameter name can only contain letters and digits and must be unique in the template. You can use the Label field to define user-friendly parameter names.

Parameter properties

Property	Required	Description
Type	Yes	<p>The data type of the parameter.</p> <p>String A string value. Example:</p> <p>Number An integer or floating-p</p> <p>CommaDelimitedList A group of strings or numbers indexed using Fn::Select</p> <p>Json A JSON string. Example:</p> <p>Boolean A Boolean value. Example:</p>
Default	No	If you do not specify a value when creating a stack, ROS checks whether a default value is defined in the template. If a default value is found, it is used. Otherwise, an error is returned.
AllowedValues	No	The list of valid parameter values.
AllowedPattern	No	The regular expression that is used to check whether the value of a parameter is a string. If the input is not a string, an error is returned.
MaxLength	No	The integer that specifies the maximum length of a parameter value that is of the string type.
MinLength	No	The integer that specifies the minimum length of a parameter value that is of the string type.

Property	Required	Description
MaxValue	No	The integer that specifies the maximum length of a parameter value that is of the number type.
MinValue	No	The integer that specifies the minimum length of a parameter value that is of the number type.
NoEcho	No	Specifies whether to return parameter values when the GetStack operation is called. If this parameter is set to true, only asterisks (**) are provided.
Description	No	The string that describes the parameter.
ConstraintDescription	No	The description of the parameter constraints.
Label	No	The alias name of the parameter, encoded in UTF-8. When web forms are generated using templates, labels can be mapped to parameter names.

Property	Required	Description
AssociationProperty	No	Specifies that the validity of a parameter value is automatically verified and valid values for the parameter are provided. This parameter must be specified in the "ROS resource type:property" format (spaces are not allowed). For example, "ALIYUN::ECS::Instance:ImageId" indicates that the parameter can be referenced by the ImageId property of the ALIYUN::ECS::Instance resource type. The ROS console verifies whether the specified image ID is available and lists other values in a drop-down list. Only ImageId verification is supported.
Confirm	No	Specifies whether to enter the parameter value for a second time when the NoEcho parameter is set to true. Default value: false . Note that this parameter can be set to true only when you use it with string type parameters and the NoEcho parameter is set to true.

Examples

In the following example, two parameters are defined.

- `username`: a string type parameter with a default value of anonymous. Valid values: anonymous, user-one, and user-two. The username must contain 6 to 12 characters in length.



Notice:

The default value must also meet the length and valid values requirements.

- **password**: a string type parameter with no default value. If you set the NoEcho parameter to true, the GetStack operation does not return any parameter values. The password must contain 1 to 41 characters in length and can contain letters and digits.

```
"Parameters" : {  
  "username" : {  
    "Label": "Username",  
    "Description": "Enter the username",  
    "Default": "anonymous",  
    "Type": "String",  
    "MinLength": "6",  
    "MaxLength": "12",  
    "AllowedValues": ["anonymous", "user-one", "user-two"]  
  },  
  "password" : {  
    "Label": "Password",  
    "NoEcho": "True",  
    "Description": "Enter the password",  
    "Type": "String",  
    "MinLength": "1",  
    "MaxLength": "41",  
    "AllowedPattern": "[a-zA-Z0-9]*"  
  }  
}
```

Pseudo parameters

Pseudo parameters are internal parameters provided by the ROS engine. They can be referenced like user-defined parameters and their values are determined while ROS is running. Supported pseudo parameters are as follows:

- **ALIYUN::StackName**: the name of the stack.
- **ALIYUN::StackId**: the ID of the stack.
- **ALIYUN::Region**: the region where the stack resides.
- **ALIYUN::AccountId**: the user ID of the stack.
- **ALIYUN::NoValue**: specifies that a specific resource property is deleted when the resource is created or updated.

6.3 Resources

This topic describes the properties of each resource and dependencies of resources in a stack. A resource can be referenced by other resources and Outputs.

Syntax

Each resource consists of an ID and a description. A resource description is enclosed in braces {}. Multiple resources are separated with commas (,). The following sample code shows the Resources syntax:

```
"Resources" : {
  "Resource1 ID" : {
    "Type" : "The resource type",
    "Condition": "The condition that specifies whether to create the resource",
    "Properties" : {
      The description of the resource properties
    }
  },
  "Resource2 ID" : {
    "Type" : "The resource type",
    "Condition": "The condition that specifies whether to create the resource",
    "Properties" : {
      The description of the resource properties
    }
  }
}
```

Parameter description:

- A resource ID is unique in a template. You can use the resource ID to reference the resource in other parts of the template.
- The Type parameter specifies the type of resource that is being declared. For example, ALIYUN::ECS::Instance indicates that the resource is an Elastic Cloud Service (ECS) instance. For more information about the resource types in Resource Orchestration Service (ROS), see [Resource Type Index](#).
- The Properties section provides additional options that you can specify for a resource. For example, you must specify an image ID for each Alibaba Cloud ECS instance.

Examples

```
"Resources" : {
  "ECSInstance" : {
    "Type" : "ALIYUN::ECS::Instance",
    "Properties" : {
      "ImageId" : "m-25l0rcfjo"
    }
  }
}
```

```
}
```

If a resource does not need properties to be declared, omit the **Properties** section of that resource.

Property values can be text strings, string lists, Boolean values, referenced parameters, or return values of functions. Text strings are enclosed with double quotation marks ("). String lists are enclosed with brackets []. Return values of intrinsic functions and referenced parameters are enclosed with braces {}. The preceding rules also apply when property values are the combinations of text strings, string lists, referenced parameters, and return values of functions.

The following sample code shows how to declare different types of properties:

```
"Properties" : {  
  "String" : "string",  
  "LiteralList" : [ "value1", "value2" ],  
  "Boolean" : "true"  
  "ReferenceForOneValue" : { "Ref" : "ResourceID" },  
  "FunctionResultWithFunctionParams" : {  
    "Fn::Join" : [ "%", [ "Key=", { "Ref" : "SomeParameter" } ] ] }  
}
```

DeletionPolicy

The **DeletionPolicy** parameter specifies whether to retain a resource when its stack is deleted. The following sample code shows how to use the **DeletionPolicy** parameter to retain an ECS instance when the stack is deleted.

```
"Resources" : {  
  "ECSInstance" : {  
    "Type" : "ALIYUN::ECS::Instance",  
    "Properties" : {  
      "ImageId" : "m-25l0rcfjo"  
    },  
    "DeletionPolicy" : "Retain"  
  }  
}
```

In this example, the ECS instance is retained when the stack created based on the template is deleted.

DependsOn

The **DependsOn** parameter allows you to create a specific resource after creating its dependent resource. If you specify the **DependsOn** parameter for a resource, the resource is created only after its dependent resource specified by the **DependsOn** parameter is created.

**Notice:**

The **Condition** parameter of the resource that is specified by the **DependsOn** parameter can be set to False. The value of False does not affect the resource creation.

The following sample code shows how to configure dependent ECS instances:

- Configure a single dependent resource:

```
"DependsOn": "ResourceName"
```

- Configure multiple dependent resources:

```
"DependsOn": [
    "ResourceName1",
    "ResourceName2"
]
```

In the following example, WebServer is created only after DatabaseServer is created:

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Resources" : {
    "WebServer": {
      "Type": "ALIYUN::ECS::Instance",
      "DependsOn": "DatabaseServer"
    },
    "DatabaseServer": {
      "Type": "ALIYUN::ECS::Instance",
      "Properties": {
        "ImageId": "m-25l0r****",
        "InstanceType": "ecs.t1.small"
      }
    }
  }
}
```

Condition

The **Condition** parameter specifies whether to create the resource. The resource can be created only when the **Condition** parameter is set to True.

In the following example, WebServer is created only if the condition determined by the **MaxAmount** parameter is true:

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Parameters": {
    "MaxAmount": {
      "Type": "Number",
      "Default": 1
    }
  },
  "Conditions": {
    "CreateWebServer": {"Fn::Not": {"Fn::Equals": [0, {"Ref": "MaxAmount"}]}}
  }
}
```



```

"Resources" : {
  "WebServer": {
    "Type": "ALIYUN::ECS::InstanceGroup",
    "Condition": "CreateWebServer",
    "Properties": {
      "ImageId": "m-25l0r****",
      "InstanceType": "ecs.t1.small",
      "MaxAmount": {"Ref": "MaxAmount"}
    }
  },
  "DatabseServer": {
    "Type": "ALIYUN::ECS::Instance",
    "Properties": {
      "ImageId": "m-25l0r****",
      "InstanceType": "ecs.t1.small"
    }
  }
}
}

```

Resource declaration example

The following sample code shows how to declare a resource:

```

"Resources" : {
  "WebServer": {
    "Type": "ALIYUN::ECS::Instance",
    "Properties": {
      "ImageId": "m-25l0r****",
      "InstanceType": "ecs.t1.small",
      "SecurityGroupId": "sg-25zwc****",
      "ZoneId": "cn-beijing-b",
      "Tags": [{
        "Key": "Department1",
        "Value": "HumanResource"
      }, {
        "Key": "Department2",
        "Value": "Finance"
      }]
    }
  },
  "ScalingConfiguration": {
    "Type": "ALIYUN::ESS::ScalingConfiguration",
    "Properties": {
      "ImageId": "ubuntu1404_64_20G_aliaegis_2015****.vhd",
      "InstanceType": "ecs.t1.small",
      "InstanceId": "i-25xhh****",
      "InternetChargeType": "PayByTraffic",
      "InternetMaxBandwidthIn": 1,
      "InternetMaxBandwidthOut": 20,
      "SystemDisk_Category": "cloud",
      "ScalingGroupId": "bwhtvpcBckYac9fe3vd0****",
      "SecurityGroupId": "sg-25zwc****",
      "DiskMappings": [
        {
          "Size": 10
        },
        {
          "Category": "cloud",
          "Size": 10
        }
      ]
    }
  }
}

```

```

    }
  }
}

```

6.4 Outputs

The Outputs section is used to define the values returned when the GetStack operation is called. For example, if you define an ECS instance ID as an output item, the ECS instance ID is returned when the GetStack operation is called.

Syntax

Each output item consists of an ID and a description. All output descriptions are enclosed in braces ({}). Multiple output items are separated by commas (,). Each output item can have multiple values in an array format. The following example shows the Outputs syntax:

```

"Outputs" : {
  "Output1 ID" : {
    "Description": "The description of the output item",
    "Condition": "The condition that specifies whether to provide resource properties",
    "Value": "The output value expression"
  },
  "Output2 ID" : {
    "Description": "The description of the output item",
    "Condition": "The condition that specifies whether to provide resource properties",
    "Value" : [
      "Output value expression 1",
      "Output value expression 2",
      ...
    ]
  }
}

```

- Output ID: the ID of the output item. Duplicate IDs are not allowed within a template.
- Description: Optional. The description of the output item.
- Value: Required. The value returned when the GetStack operation is called.
- Condition: Optional. The condition that specifies whether to create a resource and provide its information. The resource is created and its information is provided only when the specified condition is true.

In the following example, WebServer is created only if the condition determined by the MaxAmount parameter is true:

```

{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Parameters": {
    "MaxAmount": {
      "Type": "Number",
      "Default": 1
    }
  }
}

```

```

    }
  },
  "Conditions": {
    "CreateWebServer": {"Fn::Not": {"Fn::Equals": [0, {"Ref": "MaxAmount"}]}}
  }
  "Resources": {
    "WebServer": {
      "Type": "ALIYUN::ECS::InstanceGroup",
      "Condition": "CreateWebServer",
      "Properties": {
        "ImageId": "m-25l0rcfjo",
        "InstanceType": "ecs.t1.small",
        "MaxAmount": {"Ref": "MaxAmount"}
      }
    }
  }
  "Outputs": {
    "WebServerIP": {
      "Condition": "CreateWebServer",
      "Value": {
        "Fn::GetAtt": ["WebServer", "PublicIps"]
      }
    }
  }
}

```

Examples

In the following example, there are two output items. The first output item contains the value of the InstanceId parameter of WebServer, and the second output item contains the values of the PublicIp and PrivateIp parameters of WebServer.

```

"Outputs": {
  "InstanceId": {
    "Value": {"Fn::GetAtt": ["WebServer", "InstanceId"]}
  },
  "PublicIp & PrivateIp": {
    "Value": [
      {"Fn::GetAtt": ["WebServer", "PublicIp"]},
      {"Fn::GetAtt": ["WebServer", "PrivateIp"]}
    ]
  }
}

```

6.5 Functions

Resource Orchestration Service (ROS) provides several built-in functions to help you manage stacks. You can use built-in functions to define Resources and Outputs.

You can use the following built-in functions in templates: Fn::Str, Fn::Base64Encode, Fn::Base64Decode, Fn::FindInMap, Fn::GetAtt, Fn::Join, Fn::Select, Ref, Fn::GetAZs, Fn::Replace, Fn::Split, Fn::Equals, Fn::And, Fn::Or, Fn::Not, Fn::If, Fn::ListMerge, Fn::

GetJsonValue, Fn::MergeMapToList, Fn::Avg, Fn::SelectMapList, Fn::Add, Fn::Calculate, Fn::GetAtt, and Fn::Sub.

Fn::Str

The Fn::Str function is used to return the string representation of a value.

Declaration

```
"Fn::Str" : toString
```

Parameters

`toString`: the value of the number or integer type to be converted to a string.

Return value

The string representation of the value.

Examples

```
{"Fn::Str": 123456}
```

"123456" is returned in this example.

Fn::Base64Encode

The Fn::Base64Encode function is used to return the Base64 representation of the input string.

Declaration

```
"Fn::Base64Encode": "stringToEncode"
```

Parameters

`stringToEncode`: the string to be encoded in Base64.

Return value

The Base64 representation of the input string.

Examples

```
{"Fn::Base64Encode": "string to encode"}
```

c3RyaW5nIHRvIGVuY29kZQ== is returned in this example.

Fn::Base64Decode

The `Fn::Base64Decode` function is used to return a string decoded from a Base64-encoded string.

Declaration

```
{"Fn::Base64Decode": "stringToEncode"}
```

Parameters

`stringToDecode`: the string decoded from the Base64-encoded string.

Return value

The string decoded from the Base64-encoded string.

Examples

```
{"Fn::Base64Decode": "c3RyaW5nIHRvIGVuY29kZQ=="}
```

`string to encode` is returned in this example.

Fn::FindInMap

The `Fn::FindInMap` function is used to return the values based on keys in a two-level mapping that is declared in the Mappings section.

Declaration

```
"Fn::FindInMap": ["MapName", "TopLevelKey", "SecondLevelKey"]
```

Parameters

- `MapName`: the ID of a mapping declared in the Mappings section that contains keys and values.
- `TopLevelKey`: the top-level key name. The value is a list of key-value pairs.
- `SecondLevelKey`: the second-level key name. The value is a string or a number.

Return value

The value that is assigned to the `SecondLevelKey` parameter.

Examples

The `ImageId` property must be specified when you create a `WebServer` instance. The Mappings section describes the `ImageId` mappings by region. The Parameters section describes the regions that must be specified by template users. `Fn::FindInMap` finds the

corresponding ImageId mapping in RegionMap based on the region specified by a user, and then finds the corresponding ImageId in the mapping.

- MapName is set to the map of interest, which is "RegionMap" in this example.
- TopLevelKey is set to the region where the stack is created, which is { "Ref" : "regionParam" } in this example.
- SecondLevelKey is set to the required architecture, which is "32" in this example.

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "regionParam": {
      "Description": "The region where the ECS instance is created",
      "Type": "String",
      "AllowedValues": [
        "hangzhou",
        "beijing"
      ]
    }
  },
  "Mappings": {
    "RegionMap": {
      "hangzhou": {
        "32": "m-25l0rcfjo",
        "64": "m-25l0rcfj1"
      },
      "beijing": {
        "32": "m-25l0rcfj2",
        "64": "m-25l0rcfj3"
      }
    }
  },
  "Resources": {
    "WebServer": {
      "Type": "ALIYUN::ECS::Instance",
      "Properties": {
        "ImageId": {
          "Fn::FindInMap": [
            "RegionMap",
            { "Ref": "regionParam" },
            "32"
          ]
        },
        "InstanceType": "ecs.t1.small",
        "SecurityGroupId": "sg-25zwc****",
        "ZoneId": "cn-beijing-b",
        "Tags": [
          {
            "Key": "key1",
            "Value": "value1"
          },
          {
            "Key": "key2",
            "Value": "value2"
          }
        ]
      }
    }
  }
}
```

```
}
```

Supported functions

- `Fn::FindInMap`
- `Ref`

Fn::GetAtt

The `Fn::GetAtt` function is used to return the value of a property from a resource in a template.

Declaration

```
"Fn::GetAtt": ["resourceID", "attributeName"]
```

Parameters

- `resourceID`: the ID of the resource.
- `attributeName`: the name of the resource property.

Return value

The value of the resource property.

Examples

The `ImageId` property of `MyEcsInstance` is returned in this example.

```
{"Fn::GetAtt" : ["MyEcsInstance", "ImageID"]}
```

Fn::Join

The `Fn::Join` function is used to append a set of values into a single value that is separated by a specified delimiter.

Declaration

```
{"Fn::Join": ["delimiter", ["string1", "string2", ... ]]}
```

Parameters

- `delimiter`: the value used to divide the string. The delimiter value can be left blank so that all the values are directly combined.
- `["string1", "string2", ...]`: the list of values that are combined into a string.

Return value

The combined string.

Examples

```
{"Fn::Join": [ ";", ["a", "b", "c"] ]}
```

"a;b;c" is returned in this example.

Supported functions

- Fn::Base64Encode
- Fn::GetAtt
- Fn::Join
- Fn::Select
- Ref

Fn::Select

The Fn::Select function is used to return a single data element from a list of data elements by an index.

Declaration

- The following example assumes that the list of data elements is an array:

```
"Fn::Select": ["index", ["value1", "value2", ... ]]
```

- The following example assumes that the list of data elements is a mapping table:

```
"Fn::Select": ["index", {"key1": "value1", ... }]
```

Parameters

index: the index of the object data element. If the list of data elements is an array, the index must be an integer ranging from 0 to N-1, where N indicates the number of elements in the array. If the list of data elements is a mapping table, the index must be a key in the mapping table.

If the corresponding value of the index cannot be found, the system returns an empty string .

Return value

The object data element

Examples

- The following example assumes that the list of data elements is an array:

```
{"Fn::Select": ["1", ["apples", "grapes", "oranges", "mangoes"]]}
```

"grapes" is returned in this example.

- The following example assumes that the list of data elements is a mapping table:

```
{"Fn::Select": ["key1", {"key1": "grapes", "key2": "mangoes"}]}
```

"grapes" is returned in this example.

- The following example assumes that the list of data elements is a comma-delimited list:

```
"Parameters": {
  "userParam": {
    "Type": "CommaDelimitedList",
    "Default": "10.0.XX.XX, 10.0.XX.XX, 10.0.XX.XX"
  }
}

"Resources": {
  "resourceID": {
    "Properties": {
      "CidrBlock": {"Fn::Select": ["0", {"Ref": "userParam"}]}
    }
  }
}
```

Supported functions

For the Fn::Select index value, you can use the Ref function.

For the Fn::Select list of data elements, you can use the following functions:

- Fn::Base64Encode
- Fn::FindInMap
- Fn::GetAtt
- Fn::Join
- Fn::Select
- Ref

Ref

The Ref function is used to return the value of a specified parameter or resource.

If the specified parameter is a resource ID, the value of the resource is returned. Otherwise, the system will return the value of the specified parameter.

Declaration

```
"Ref": "logicalName"
```

Parameters

logicalName: the logical name of the resource or parameter that you want to reference.

Return value

The value of the resource or parameter.

Examples

The following example uses the Ref function to specify regionParam as the region parameter for RegionMap of WebServer:

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "regionParam": {
      "Description": "The region where the ECS instance is created",
      "Type": "String",
      "AllowedValues": [
        "hangzhou",
        "beijing"
      ]
    }
  },
  "Mappings": {
    "RegionMap": {
      "hangzhou": {
        "32": "m-25l0rcfjo",
        "64": "m-25l0rcfj1"
      },
      "beijing": {
        "32": "m-25l0rcfj2",
        "64": "m-25l0rcfj3"
      }
    }
  },
  "Resources": {
    "WebServer": {
      "Type": "ALIYUN::ECS::Instance",
      "Properties": {
        "ImageId": {
          "Fn::FindInMap": [
            "RegionMap",
            {"Ref": "regionParam"},
            "32"
          ]
        },
        "InstanceType": "ecs.t1.small",
        "SecurityGroupId": "sg-25zwc****",
        "ZoneId": "cn-beijing-b",
        "Tags": [
          {
            "Key": "tiantt",
            "Value": "ros"
          }
        ]
      }
    }
  }
}
```

```

    },
    {
      "Key": "tiantt1",
      "Value": "ros1"
    }
  ]
}
}
}
}
}
}

```

Supported functions

When you use Ref function you cannot use other functions in it at the same time. You must specify a string value for the resource logical ID.

Fn::GetAZs

The Fn::GetAZs function is used to return a list of zones for a specified region.



Note:

Use the function only for ECS and VPC resources.

Declaration

```
"Fn::GetAZs": "region"
```

Parameters

region: the ID of the region.

Return value

The list of zones for the specified region.

Examples

The following example demonstrates how to create an ECS instance in the first zone of a specified region:

```

{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Resources" : {
    "WebServer": {
      "Type": "ALIYUN::ECS::Instance",
      "Properties": {
        "ImageId": "centos7u2_64_40G_cloudinit_2016072****",
        "InstanceType": "ecs.n1.tiny",
        "SecurityGroupId": "sg-2zedcm7ep5quses0****",
        "Password": "Ros1****",
        "AllocatePublicIP": true,
        "InternetChargeType": "PayByTraffic",
        "InternetMaxBandwidthIn": 100,
        "InternetMaxBandwidthOut": 100,
        "SystemDiskCategory": "cloud_efficiency",

```

```
    "IoOptimized": "optimized",
    "ZoneId": {"Fn::Select": ["0", {"Fn::GetAZs": {"Ref": "ALIYUN::Region"}}]}
  },
  "Outputs": {
    "InstanceId": {
      "Value": {"Fn::GetAtt": ["WebServer", "InstanceId"]}
    },
    "PublicIp": {
      "Value": {"Fn::GetAtt": ["WebServer", "PublicIp"]}
    }
  }
}
```

Supported functions

- Fn::Base64Encode
- Fn::FindInMap
- Fn::GetAtt
- Fn::Join
- Fn::Select
- Ref

Fn::Replace

The Fn::Replace function is used to replace a specified substring contained in a string with a new substring.

Declaration

```
{"Fn::Replace": [{"object_key": "object_value"}, "object_string"]}
```

Parameters

- `object_key`: the substring to be replaced.
- `object_value`: the new substring to replace the previous substring.
- `object_string`: the string whose `object_key` is replaced.

Return value

The string after replacement.

Examples

The following example demonstrates how to replace "print" with "echo" in the specified script:

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",
```

```

"Resources": {
  "WebServer": {
    "Type": "ALIYUN::ECS::Instance",
    "Properties": {
      "ImageId": "centos_7_2_64_40G_base_20170222****",
      "InstanceType": "ecs.n1.medium",
      "SecurityGroupId": "sg-94q49****",
      "Password": "MytestPassword****",
      "IoOptimized": "optimized",
      "VSwitchId": "vsw-94vdv****",
      "VpcId": "vpc-949uz****",
      "SystemDiskCategory": "cloud_ssd",
      "UserData": {"Fn::Replace": [{"print": "echo"},
        {"Fn::Join": ["", [
          "#! /bin/sh\n",
          "mkdir ~/test_ros\n",
          "print hello > ~/1.txt\n"
        ]]}]}
    }
  },
  "Outputs": {
    "InstanceId": {
      "Value": {"Fn::GetAtt": ["WebServer", "InstanceId"]}
    },
    "PublicIp": {
      "Value": {"Fn::GetAtt": ["WebServer", "PublicIp"]}
    }
  }
}

```

Supported functions

- Fn::Base64Encode
- Fn::GetAtt
- Fn::Join
- Fn::Select
- Ref

Fn::Split

The Fn::Split function is used to split a string into a list of values separated by a specified delimiter and return the list.

Declaration

```
"Fn::Split": ["delim", "original_string"]
```

Parameters

- `delim`: the specified delimiter, which can be `,`, `;`, `\n`, `\t`.
- `original_string`: the string to be split.

Return value

A list of string values.

Examples

- The following example assumes that the list of data elements is an array:

```
{"Fn::Split": [";", "foo; bar; achoo"]}
```

["foo", " bar", "achoo "] is returned in this example.

- The following example uses Fn::Split to split InstanceIds:

```
{
  "Parameters": {
    "InstanceIds": {
      "Type": "String",
      "Default": "instane1_id,instance2_id,instance2_id"
    }
  },
  "Resources": {
    "resourceID": {
      "Type": "ALIYUN::SLB::BackendServerAttachment",
      "Properties": {
        "BackendServerList": {
          "Fn::Split": [
            ",",
            {
              "Ref": "InstanceIds"
            }
          ]
        }
      }
    }
  }
}
```

Supported functions

- Fn::Base64Encode
- Fn::FindInMap
- Fn::GetAtt
- Fn::Join
- Fn::Select
- Fn::Replace
- Fn::GetAZs
- Fn::If
- Ref

Fn::Equals

The Fn::Equals function is used to compare whether two values are equal. If the two values are equal, true is returned. If the two values are not equal, false is returned.

Declaration

```
{"Fn::Equals": ["value_1", "value_2"]}
```

Parameters

value: the values to be compared.

Return value

true or false.

Examples

The following example uses Fn::Equals to define a condition in the Conditions section:

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "EnvType": {
      "Default": "pre",
      "Type": "String"
    }
  },
  "Conditions": {
    "TestEqualsCond": {
      "Fn::Equals": [
        "prod",
        {"Ref": "EnvType"}
      ]
    }
  }
}
```

Supported functions

- Fn::Or
- Fn::Not
- Fn::Equals
- Fn::FindInMap
- Fn::And
- Ref

Fn::And

The Fn::And function is used to represent the AND operator, and must contain at least two conditions. If all the specified conditions are evaluated as true, true is returned. If any condition is evaluated as false, false is returned.

Declaration

```
{"Fn::And": ["condition", {...}]}
```

Parameters

condition: the condition to be evaluated.

Return value

true or false.

Examples

The following example uses Fn::And to define a condition in the Conditions section:

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Parameters":{
    "EnvType":{
      "Default":"pre",
      "Type":"String"
    }
  },
  "Conditions": {
    "TestEqualsCond": {"Fn::Equals": ["prod", {"Ref": "EnvType"}]},
    "TestAndCond": {"Fn::And": ["TestEqualsCond", {"Fn::Equals": ["pre", {"Ref": "EnvType"}]}]}
  }
}
```

Supported functions

- Fn::Or
- Fn::Not
- Fn::Equals
- Fn::FindInMap
- Fn::And
- Ref

Fn::Or

The Fn::Or function is used to represent the OR operator, and must contain at least two conditions. If any specified condition is evaluated as true, true is returned. If all the conditions are evaluated as false, false is returned.

Declaration

```
{"Fn::Or": ["condition", {...]}
```

Parameters

condition: the condition to be evaluated.

Return value

true or false.

Examples

The following example uses Fn::Or to define a condition in the Conditions section:

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Parameters":{
    "EnvType":{
      "Default":"pre",
      "Type":"String"
    }
  },
  "Conditions": {
    "TestEqualsCond": {"Fn::Equals": ["prod", {"Ref": "EnvType"}]},
    "TestOrCond": {"Fn::And": ["TestEqualsCond", {"Fn::Equals": ["pre", {"Ref": "EnvType"}]}]}
  }
}
```

Supported functions

- Fn::Or
- Fn::Not
- Fn::Equals
- Fn::FindInMap
- Fn::And
- Ref

Fn::Not

The Fn::Not function is used to represent the NOT operator. If a condition is evaluated as false, true is returned. If a condition is evaluated as true, false is returned.

Declaration

```
{"Fn::Not": "condition"}
```

Parameters

condition: the condition to be evaluated.

Return value

true or false.

Examples

The following example uses Fn::Not to define a condition in the Conditions section:

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Parameters":{
    "EnvType":{
      "Default":"pre",
      "Type":"String"
    }
  },
  "Conditions": {
    "TestNotCond": {"Fn::Not": {"Fn::Equals": ["pre", {"Ref": "EnvType"}]}}
  }
}
```

Supported functions

- Fn::Or
- Fn::Not
- Fn::Equals
- Fn::FindInMap
- Fn::And
- Ref

Fn::If

This function returns one of two possible values. If a specified condition is evaluated as true, one value is returned. If the specified condition is evaluated as false, the other value is returned. The property values of Resources and Outputs in templates support the Fn::If function. You can use the ALIYUN::NoValue pseudo parameter as the return value to delete the corresponding property.

Declaration

```
{"Fn::If": ["condition_name", "value_if_true", "value_if_false"]}
```

Parameters

- `condition_name`: the name of the condition in the Conditions section. A condition is referenced by using the condition name.
- `value_if_true`: If the specified condition is evaluated as true, this value is returned.
- `value_if_false`: If the specified condition is evaluated as false, this value is returned.

Examples

The following example demonstrates how to determine whether to create a data disk based on input parameters:

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "EnvType": {
      "Default": "pre",
      "Type": "String"
    }
  },
  "Conditions": {
    "CreateDisk": {
      "Fn::Equals": [
        "prod",
        {
          "Ref": "EnvType"
        }
      ]
    }
  },
  "Resources": {
    "WebServer": {
      "Type": "ALIYUN::ECS::Instance",
      "Properties": {
        "DiskMappings": {
          "Fn::If": [
            "CreateDisk",
            [
              {
                "Category": "cloud_efficiency",
                "DiskName": "FirstDataDiskName",
                "Size": 40
              },
              {
                "Category": "cloud_ssd",
                "DiskName": "SecondDataDiskName",
                "Size": 40
              }
            ],
            {
              "Ref": "ALIYUN::NoValue"
            }
          ]
        }
      }
    }
  }
}
```

```

    },
    "VpcId": "vpc-2zew9pxh2yirtzqxd****",
    "SystemDiskCategory": "cloud_efficiency",
    "SecurityGroupId": "sg-2zece6wcqriejf1v****",
    "SystemDiskSize": 40,
    "ImageId": "centos_6_8_64_40G_base_20170222****",
    "IoOptimized": "optimized",
    "VSwitchId": "vsw-2zed9txvy7h2srqo6****",
    "InstanceType": "ecs.n1.medium"
  }
}
},
"Outputs": {
  "InstanceId": {
    "Value": {
      "Fn::GetAtt": [
        "WebServer",
        "InstanceId"
      ]
    }
  },
  "ZoneId": {
    "Value": {
      "Fn::GetAtt": [
        "WebServer",
        "ZoneId"
      ]
    }
  }
}
}
}
}
}

```

Supported functions

- Fn::Or
- Fn::Not
- Fn::Equals
- Fn::FindInMap
- Fn::And
- Ref

Fn::ListMerge

The Fn::ListMerge function is used to merge multiple lists into one list.

Declaration

```

{"Fn::ListMerge": [{"list_1_item_1", "list_1_imte_2", ...}, {"list_2_item_1", "list_2_imte_2", ...}]}

```

Parameters

- ["list_1_item_1", "list_1_imte_2", ...]: the first list to merge.
- ["list_2_item_1", "list_2_imte_2", ...]: the second list to merge into the first list.

Examples

The following example demonstrates how to attach two ECS instance groups to an SLB instance:

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Resources" : {
    "LoadBalancer": {
      "Type": "ALIYUN::SLB::LoadBalancer",
      "Properties": {
        "LoadBalancerName": "ros",
        "AddressType": "internet",
        "InternetChargeType": "paybybandwidth",
      }
    },
    "BackendServer1": {
      "Type": "ALIYUN::ECS::InstanceGroup",
      "Properties": {
        "ImageId": "m-2ze9uqi7wo61hwep****",
        "InstanceType": "ecs.t1.small",
        "SecurityGroupId": "sg-2ze8yxgempcdsq3i****",
        "MaxAmount": 1,
        "MinAmount": 1
      }
    },
    "BackendServer2": {
      "Type": "ALIYUN::ECS::InstanceGroup",
      "Properties": {
        "ImageId": "m-2ze9uqi7wo61hwep****",
        "InstanceType": "ecs.t1.small",
        "SecurityGroupId": "sg-2ze8yxgempcdsq3i****",
        "MaxAmount": 1,
        "MinAmount": 1
      }
    },
    "Attachment": {
      "Type": "ALIYUN::SLB::BackendServerAttachment",
      "Properties": {
        "LoadBalancerId": {"Ref": "LoadBalancer"},
        "BackendServerList": { "Fn::ListMerge": [
          {"Fn::GetAtt": ["BackendServer1", "InstanceIds"]},
          {"Fn::GetAtt": ["BackendServer2", "InstanceIds"]}
        ] }
      }
    }
  }
}
```

Supported functions

- Fn::Base64Encode
- Fn::GetAtt
- Fn::Join
- Fn::Select
- Ref

- Fn::Join
- Fn::If

Fn::GetJsonValue

The Fn::GetJsonValue function is used to resolve a JSON string and obtain its key value from the first layer.

Declaration

```
{"Fn::GetJsonValue": ["key", "json_string"]}
```

Parameters

- `key`: the key value.
- `json_string`: the specified JSON string to be resolved.

Examples

In the following example, the WebServer instance executes UserData and returns a JSON string, and the WebServer2 instance then obtains the corresponding key value from the string.

```
{
  "ROSTemplateFormatVersion" : "2015-09-01",
  "Resources" : {
    "WebServer": {
      "Type": "ALIYUN::ECS::Instance",
      "Properties": {
        "ImageId" : "m-2ze45uwova5fedlu****",
        "InstanceType": "ecs.n1.medium",
        "SecurityGroupId": "sg-2ze7pxymaix640qr****",
        "Password": "Wenqiao****",
        "IoOptimized": "optimized",
        "VSwitchId": "vsw-2zei67xd9nhcqxzec****",
        "VpcId": "vpc-2zevx9ios1rszqv0a****",
        "SystemDiskCategory": "cloud_ssd",
        "UserData": {"Fn::Join": ["", [
          "#! /bin/sh\n",
          "mkdir ~/test_ros\n",
          "print hello > ~/1.txt\n",
          "Fn::GetAtt": ["WaitConHandle", "CurlCli"],
          "\n",
          "Fn::GetAtt": ["WaitConHandle", "CurlCli"],
          "-d '{\"id\": \"1\", \"data\": [\"1111\", \"2222\"]}'\n"
        ]}],
        "PrivateIpAddress": "192.168.XX.XX",
        "HostName": "userdata-1"
      }
    },
    "WaitConHandle": {
      "Type": "ALIYUN::ROS::WaitConditionHandle"
    },
    "WaitCondition": {
      "Type": "ALIYUN::ROS::WaitCondition",
```

```

    "Properties": {
      "Handle": {"Ref": "WaitConHandle"},
      "Timeout": 900
    }
  },
  "WebServer2": {
    "Type": "ALIYUN::ECS::Instance",
    "Properties": {
      "ImageId": "m-2ze45uwova5fedlu****",
      "InstanceType": "ecs.n1.medium",
      "SecurityGroupId": "sg-2ze7pxymaix640qr****",
      "Password": "Wenqiao****",
      "IoOptimized": "optimized",
      "VSwitchId": "vsw-2zei67xd9nhcqxyzec****",
      "VpcId": "vpc-2zevx9ios1rszqv0a****",
      "SystemDiskCategory": "cloud_ssd",
      "UserData": {
        {"Fn::Join": ["", [
          "#! /bin/sh\n",
          "mkdir ~/test_ros\n",
          "echo hello > ~/1.txt\n",
          "server_1_token=",
          {"Fn::GetJsonValue": ["1", {"Fn::GetAtt": ["WaitCondition", "Data"]}]}],
        "\n"
      ]}],
      "PrivateIpAddress": "192.168.XX.XX",
      "HostName": "userdata-2"
    }
  },
  "Outputs": {
    "InstanceId": {
      "Value": {"Fn::GetAtt": ["WebServer", "InstanceId"]}
    },
    "PublicIp": {
      "Value": {"Fn::GetAtt": ["WebServer", "PublicIp"]}
    }
  }
}

```

Supported functions

- Fn::Base64Encode
- Fn::GetAtt
- Fn::Join
- Fn::Select
- Ref
- Fn::Join
- Fn::If

Fn::MergeMapToList

The Fn::MergeMapToList function is used to merge multiple mappings into a list of mapping elements.

Declaration

```
{"Fn::MergeMapToList": [{"key_1": ["key_1_item_1", "key_1_item_2", ...]}, {"key_2": ["key_2_item_1", "key_2_item_2", ...]}, ... ]}
```

Parameters

- {"key_1": ["key_1_item_1", "key_1_item_2", ...]}: the first mapping to merge. The "key_1" value must be a list. "key_1" is the key for each mapping in the list of merged mappings. The "key_1" value is "key_1_item_1" for the first merged mapping and "key_1_item_2" for the second merged mapping. All values follow the same format. The length of the final list of merged mappings is the length of the longest list "key_x" from all mappings being merged. If a "key_y" list is shorter, the last element of the list is repeated until the list is the longest.
- {"key_2": ["key_2_item_1", "key_2_item_2", ...]}: the second mapping to merge into the first mapping. The "key_2" value must be a list. "key_2" is the key for each mapping in the merged list. The "key_2" value is "key_2_item_1" for the first merged mapping and "key_2_item_2" for the second merged mapping.

Examples

- The following example demonstrates how to merge three mappings. The length of the list based on the key values for each mapping is the same.

```
{
  "Fn::MergeMapToList": [
    {"key_1": ["key_1_item_1", "key_1_item_2"]},
    {"key_2": ["key_2_item_1", "key_2_item_2"]},
    {"key_3": ["key_3_item_1", "key_3_item_2"]}
  ]
}
```

The merged result is as follows:

```
[
  {
    "key_1": "key_1_item_1",
    "key_2": "key_2_item_1",
    "key_3": "key_3_item_1"
  },
  {
    "key_1": "key_1_item_2",
    "key_2": "key_2_item_2",
    "key_3": "key_3_item_2"
  }
]
```



```
]
```

- The length of the list based on the key values for each mapping varies in the following example:

```
{
  "Fn::MergeMapToList": [
    {"key_1": ["kye_1_item_1", "kye_1_item_2"]},
    {"key_2": ["kye_2_item_1", "kye_2_item_2", "key_2_item_3"]},
    {"key_3": ["kye_3_item_1", "kye_3_item_2"]}
  ]
}
```

The merged result is as follows:

```
[
  {
    "key_1": "kye_1_item_1",
    "key_2": "kye_2_item_1",
    "key_3": "kye_3_item_1"
  },
  {
    "key_1": "kye_1_item_2",
    "key_2": "kye_2_item_2",
    "key_3": "kye_3_item_2"
  },
  {
    "key_1": "kye_1_item_2",
    "key_2": "kye_2_item_3",
    "key_3": "kye_3_item_2"
  }
]
```

- In the following template example, all instances created in WebServer are added to the VServer group of an SLB instance:

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Resources": {
    "WebServer": {
      "Type": "ALIYUN::ECS::InstanceGroupClone",
      "Properties": {
        "SourceInstanceId": "i-xxxxx",
        "Password": "Hello****",
        "MinAmount": 1,
        "MaxAmount": 1
      }
    },
    "CreateVServerGroup": {
      "Type": "ALIYUN::SLB::VServerGroup",
      "Properties": {
        "LoadBalancerId": "lb-****",
        "VServerGroupName": "VServerGroup-****",
        "BackendServers": {
          "Fn::MergeMapToList": [
            {"Port": [6666, 9090, 8080]},
            {"ServerId": {"Fn::GetAtt": ["WebServer", "InstanceIds"]}},
            {"Weight": [20, 100]}
          ]
        }
      }
    }
  }
}
```

```
}  
  }  
} }
```

Supported functions

- Fn::Base64Encode
- Fn::GetAtt
- Fn::Join
- Fn::Select
- Ref
- Fn::Join
- Fn::If
- Fn::ListMerge
- Fn::GetJsonValue

Fn::Avg

The Fn::Avg function is used to return the average value of a set of numbers.

Declaration

```
{"Fn::Avg": [ndigits, [number1, number2, ... ]]}
```

Parameters

- **ndigits**: the number of decimal places to report. This parameter value must be an integer.
- **[number1, number2, ...]**: the set of numbers for which the average value will be calculated. Each element in the group must be either a number or a string that can be converted into a number.

Return value

The average value of the set of numbers.

Examples

```
{ "Fn::Avg": [ 1, [1, 2, 6.0] ] }  
{ "Fn::Avg": [ 1, ['1', '2', '6.0'] ] }
```

3.0 is returned in this example.

Supported functions

- Fn::GetAtt
- Ref

Fn::SelectMapList

The Fn::SelectMapList function is used to return a list of map elements.

Declaration

```
{"Fn::SelectMapList": ["key2", [{"key1": "value1-1", "key3": "value1-3"}, {"key1": "value2-1", "key2": "value2-2"}, {"key1": "value3-1", "key2": "value3-2"}, ... ]]}
```

Parameters

- `key2`: the key to be queried in the map.
- `[{"key1": "value1-1", "key3": "value1-3"}, ...]`: the list of maps.

Return value

A list of key values for all maps in the map list.

Examples

```
{
  "Fn::SelectMapList": [
    "key2",
    [
      {"key1": "value1-1", "key3": "value1-3"},
      {"key1": "value2-1", "key2": "value2-2"},
      {"key1": "value3-1", "key2": "value3-2"}
    ]
  ]
}
```

`["value2-2","value3-2"]` is returned in this example.

Fn::Add

The Fn::Add function is used to sum the values of parameters.

Declaration

```
{"Fn::Add": [{"Product": "ROS"}, {"Fn": "Add"}]}
```

Parameters

- The parameters must be arranged as a list.
- The parameters in the list can be of the Number, List, or Dictionary type. All the parameters must be of the same type. The list must contain at least two parameters.

Return value

If the parameter values are numbers, sum the parameter values. If the parameter values are lists, concatenate the values. If the parameter values are dictionaries, merge the values, and overwrite the former parameter value with the latter one if the two parameters have the same key.

Examples

```
{
  "Fn::Add": [
    {"Product": "ROS"},
    {"Fn": "Add"}
  ]
}
```

`{"Fn": "Add", "Product": "ROS"}` is returned in this example.

Fn::Calculate

The Fn::Calculate function is used to calculate an expression in string format.

Declaration

```
{"Fn::Calculate" : [expression, ndigits, [<number1>, <number2>, ... ]]}
```

Parameters

- `expression`: the expression in string format.
- `ndigits`: the number of decimal places to report. This parameter value must be 0 or an positive integer. This parameter takes effect only if the expression contains floating-point numbers.
- `[<number0>, <number1>, <number2>, ...]`: an optional parameter. You can define `{n}` in the expression, where `n` indicates the index of a specific number. You can replace `{n}` with the number while calculating the expression.

Return value

The calculation result of the expression.

Examples

```
{"Fn::Calculate": ["(2+3)/2*3-1", 1]}
{"Fn::Calculate": ["(2.0+3)/2*3-1", 1]}
{"Fn::Calculate": ["({1}+3)/2*3-1", 1, [3, 5, 6]]}
```

The calculation result is as follows:

```
6
6.5
```

Fn::Sub

The Fn::Sub function substitutes variables in an input string by using values that you specify.

Declaration

```
{ "Fn::Sub": [ String, { Var1Name: Var1Value, Var2Name: Var2Value, ... } ] }
```

Parameters

- String

A string with variables that ROS substitutes with specified values during runtime. Write variables in the `${VarName}` format. Variables can be template parameter names, pseudo parameter names, resource logical IDs, resource attributes, or variables in key-value mappings. If you specify only template parameter names, pseudo parameter names, resource logical IDs, and resource attributes, you do not need to specify variables in key-value mappings.

If you specify template parameter names, pseudo parameter names, or resource logical IDs, such as `${MyParameter}`, ROS returns the same values as if you used the `Ref` built-in function. If you specify resource attributes such as `${MyInstance.InstanceId}`, ROS returns the same values as if you used the `Fn::GetAtt` built-in function.

To use the combination of `${}` as normal characters without being escaped, add an exclamation point (!) after the open brace, such as `${! Literal}`. ROS resolves this text as `${ Literal}`.

- VarName

The name of a variable that you included in the String parameter.

- VarValue

The value that ROS substitutes for the associated variable name during runtime.

If key-value mappings are not required, you can use the following declaration:

```
{ "Fn::Sub": String }
```

Return value

ROS returns the substituting string and substitute the values of all variables.

Examples

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "VpcName": {
      "Type": "String",
      "Default": "vpc"
    }
  },
  "Resources": {
    "Vpc": {
      "Type": "ALIYUN::ECS::VPC",
      "Properties": {
        "VpcName": {
          "Ref": "VpcName"
        },
        "CidrBlock": "10.0.XX.XX"
      }
    }
  },
  "Outputs": {
    "Pseudo": {
      "Value": {
        "Fn::Sub": [
          "Var1: ${Var1}, Var2: ${Var2}, StackName: ${ALIYUN::StackName}, Region: ${ALIYUN::Region}",
          {
            "Var1": "Var1Value",
            "Var2": "Var2Value"
          }
        ]
      }
    },
    "VpcId": {
      "Value": {
        "Fn::Sub": "Return value of the resource: ${Vpc.VpcId}. Resource ID: ${Vpc}."
      }
    }
  }
}
```

In this example, the following values are returned:

```
Var1: Var1Value, Var2: Var2Value, StackName: SubTest, Region: cn-hangzhou
```

```
Return value of the resource: vpc-bp11eu7avmtvr37hl****. Resource ID: vpc-bp11eu7avmtvr37hl****.
```

6.6 Mappings

The Mappings section is a key-value mapping table. When mappings are used in Resources or Outputs definitions, use `Fn::FindInMap` to find their values by using corresponding keys.

Syntax

A mapping consists of one or more key-value pairs where both the keys and values can be strings or numbers. Multiple mappings are separated by commas (,). Mapping names must be unique. Mappings must be pure data and cannot parse functions.

Examples

The following example shows a correct mapping definition:

```
"Mappings": {
  "ValidMapping": {
    "TestKey1": {"TestValu1": "value1"},
    "TestKey2": {"TestValu2": "value2"},
    1234567890: {"TestValu3": "value3"},
    "TestKey4": {"TestValu4": 1234}
  }
}
```

The following example shows an incorrect mapping definition:

```
"Mappings": {
  "InvalidMapping1": {
    "ValueList": ["foo", "bar"],
    "ValueString": "baz"
  },
  "InvalidMapping2": ["foo", {"bar": "baz"}],
  "InvalidMapping3": "foobar"
}
```

The following example shows how to use `Fn::FindInMap` to find the return value:

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "regionParam": {
      "Description": "The region where the ECS instance is created",
      "Type": "String",
      "AllowedValues": [
        "hangzhou",
        "beijing"
      ]
    }
  },
  "Mappings": {
```

```

"RegionMap": {
  "hangzhou": {
    "32": "m-25l0rcfjo",
    "64": "m-25l0rcfj1"
  },
  "beijing": {
    "32": "m-25l0rcfj2",
    "64": "m-25l0rcfj3"
  }
},
"Resources": {
  "WebServer": {
    "Type": "ALIYUN::ECS::Instance",
    "Properties": {
      "ImageId": {
        "Fn::FindInMap": [
          "RegionMap",
          {
            "Ref": "regionParam"
          }
        ],
        "32"
      ]
    },
    "InstanceType": "ecs.t1.small",
    "SecurityGroupId": "sg-25zwc3se0",
    "ZoneId": "cn-beijing-b",
    "Tags": [
      {
        "Key": "Department1",
        "Value": "HumanResource"
      },
      {
        "Key": "Department2",
        "Value": "Finance"
      }
    ]
  }
}
}
}

```

6.7 Conditions

There are four conditional operators: `Fn::And`, `Fn::Or`, `Fn::Not`, and `Fn::Equals`. These operators, along with the parameters that you specify when creating or updating a stack, are used to evaluate each condition. You can reference other conditions, parameters, and mappings in your condition. Conditions are used in resource and output definitions to establish dependencies. Use `Fn::If` or `Condition` in resource and output definitions to implement conditions.

Syntax

Each condition consists of a condition name and a condition body. The condition name is a string. The condition body starts with `Fn::And`, `Fn::Or`, `Fn::Not`, or `Fn::Equals`. You can

reference other conditions in your condition and separate multiple conditions with commas (,). Each condition name must be unique.

The following functions can be used, but not as the outermost functions:

"Fn::Select", "Fn::Join", "Fn::Split", "Fn::Replace", "Fn::Base64Encode", "Fn::Base64Decode", "Fn::MemberListToMap", "Fn::If", "Fn::ListMerge", "Fn::GetJsonValue", "Fn::MergeMapToList", "Fn::SelectMapList", "Fn::Add", "Fn::Avg", "Fn::Str", "Fn::Calculate", "Ref"(parameter references only), and "Fn::FindInMap".

Examples

The following example shows how to define conditions:

```
"Conditions": {
  "DevEnv": {"Fn::Equals": ["Dev", {"Ref": "EnvType"}]},
  "UTEnv": {"Fn::Equals": ["UT", {"Ref": "EnvType"}]},
  "PREEnv": {"Fn::Not": {"Fn::Or": ["DevEnv", "UTEnv"]}},
  "ProdEnv": {"Fn::And": [{"Fn::Equals": ["Prod", {"Ref": "EnvType"}]}, "PREEnv"]}
}
```

The following example shows how to use conditions in a resource definition. In this example, a condition is used to determine whether to create a data disk and an OSS bucket for an ECS instance based on the value of the EnvType parameter.

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "EnvType": {
      "Default": "pre",
      "Type": "String"
    }
  },
  "Conditions": {
    "CreateProdRes": {
      "Fn::Equals": [
        "prod",
        {
          "Ref": "EnvType"
        }
      ]
    }
  },
  "Resources": {
    "WebServer": {
      "Type": "ALIYUN::ECS::Instance",
      "Properties": {
        "DiskMappings": {
          "Fn::If": [
            "CreateProdRes",
            [
              {
                "Category": "cloud_efficiency",
                "DiskName": "FirstDataDiskName",
                "Size": 40
              }
            ]
          ]
        }
      }
    }
  }
}
```

```

        },
        {
            "Category": "cloud_ssd",
            "DiskName": "SecondDataDiskName",
            "Size": 40
        }
    ],
    {
        "Ref": "ALIYUN::NoValue"
    }
]
},
{
    "VpcId": "vpc-2zew9pxh2yirtzqxdboi1",
    "SystemDiskCategory": "cloud_efficiency",
    "SecurityGroupId": "sg-2zece6wcqriejf1v****",
    "SystemDiskSize": 40,
    "ImageId": "centos_6_8_64_40G_base_20170222.vhd",
    "IoOptimized": "optimized",
    "VSwitchId": "vsw-2zed9txvy7h2srqo6jmgq",
    "InstanceType": "ecs.n1.medium"
}
},
{
    "OssBucket": {
        "Type": "ALIYUN::OSS::Bucket",
        "Condition": "CreateProdRes",
        "Properties": {
            "AccessControl": "private",
            "BucketName": "myprodbucket"
        }
    }
},
{
    "Outputs": {
        "InstanceId": {
            "Value": {
                "Fn::GetAtt": [
                    "WebServer",
                    "InstanceId"
                ]
            }
        }
    }
},
{
    "OssDomain": {
        "Condition": "CreateProdRes",
        "Value": {
            "Fn::GetAtt": [
                "OssBucket",
                "DomainName"
            ]
        }
    }
}
}
}
}

```

6.8 Pseudo parameters

Pseudo parameters are fixed parameters provided by the orchestration engine of Resource Orchestration Service (ROS). They can be referenced like user-defined parameters and their values are determined while ROS is running.

ROS provides the following pseudo parameters:

- `ALIYUN::StackName`: obtains the name of the stack.
- `ALIYUN::StackId`: obtains the ID of the stack.
- `ALIYUN::Region`: obtains the region where the stack resides.
- `ALIYUN::AccountId`: obtains the account ID of the stack.
- `ALIYUN::TenantId`: obtains the Alibaba Cloud account ID of the stack.
- `ALIYUN::NoValue`: If you use `ALIYUN::NoValue` for an optional property when creating or updating a stack, the property will be deleted. If you use `ALIYUN::NoValue` for a required property, the default value of the property depends on its data type. For example, if the property is of the String type, its value will be an empty string. If the property is of the Integer type, its value will be 0. If the property is of the Array type, its value will be an empty array.

Examples

JSON format

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Parameters": {
    "EnvType": {
      "Default": "pre",
      "Type": "String"
    }
  },
  "Conditions": {
    "CreateDisk": {
      "Fn::Equals": [
        "prod",
        {
          "Ref": "EnvType"
        }
      ]
    }
  }
},
  "Resources": {
    "WebServer": {
      "Type": "ALIYUN::ECS::Instance",
      "Properties": {
        "DiskMappings": {
```

```

      "Fn::If": [
        "CreateDisk",
        [
          {
            "Category": "cloud_efficiency",
            "DiskName": "FirstDataDiskName",
            "Size": 40
          },
          {
            "Category": "cloud_ssd",
            "DiskName": "SecondDataDiskName",
            "Size": 40
          }
        ],
        {
          "Ref": "ALIYUN::NoValue"
        }
      ]
    },
    "VpcId": "vpc-m5eebunc50zfbmts7****",
    "SystemDiskCategory": "cloud_efficiency",
    "SecurityGroupId": "sg-m5eagh7rzys2z8sa****",
    "SystemDiskSize": 40,
    "ImageId": "centos_7",
    "IoOptimized": "optimized",
    "VSwitchId": "vsw-m5eem62p9729y6gps****",
    "InstanceType": "ecs.c5.large"
  }
},
"Outputs": {
  "StackName": {
    "Value": {
      "Ref": "ALIYUN::StackName"
    }
  },
  "StackId": {
    "Value": {
      "Ref": "ALIYUN::StackId"
    }
  },
  "Region": {
    "Value": {
      "Ref": "ALIYUN::Region"
    }
  },
  "UserID": {
    "Value": {
      "Ref": "ALIYUN::AccountId"
    }
  }
}
}
}

```

YAML format

```

ROSTemplateFormatVersion: '2015-09-01'
Parameters:
  EnvType:
    Default: pre
    Type: String
Conditions:
  CreateDisk:

```

```

Fn::Equals:
- prod
- Ref: EnvType
Resources:
  WebServer:
    Type: ALIYUN::ECS::Instance
    Properties:
      DiskMappings:
        Fn::If:
          - Createdisk
          - Category: cloud_efficiency
            DiskName: FirstDataDiskName
            Size: 40
          - Category: cloud_ssd
            DiskName: SecondDataDiskName
            Size: 40
          - Ref: ALIYUN::NoValue
        VpcId: vpc-m5eebunc50zfbmts7****
        SystemDiskCategory: cloud_efficiency
        SecurityGroupId: sg-m5eagh7rzys2z8sa****
        SystemDiskSize: 40
        ImageId: centos_7
        IoOptimized: optimized
        VSwitchId: vsw-m5eem62p9729y6gps****
        InstanceType: ecs.c5.large
    Outputs:
      StackName:
        Value:
          Ref: ALIYUN::StackName
      StackId:
        Value:
          Ref: ALIYUN::StackId
      Region:
        Value:
          Ref: ALIYUN::Region
      UserID:
        Value:
          Ref: ALIYUN::AccountId

```

6.9 Metadata

Metadata is used to group the parameters defined in the Parameters section and define tags for each group.

When you create a stack in the [ROS console](#), the parameters defined in Parameters are displayed in the **Parameters** section of the **Configure Template Parameters** step.

Syntax

```

"Metadata": {
  "ALIYUN::ROS::Interface": {
    "ParameterGroups": [// ParameterGroups is required.
      <Group1>,
      <Group2>,
      ...
    ],
    "TemplateTags": [<Optional. A custom string.>]
  }
}

```

```
}
```

Group Syntax

```
{
  "Parameters": [// Parameters is required.
    <Parameter1>,
    <Parameter2>,
    ...
  ],
  "Label": {// Label is required.
    "default": <custom string>
  }
}
```

Examples

```
{
  "ROSTemplateFormatVersion": "2015-09-01",
  "Description": "Metadata demo",
  "Metadata": {
    "ALIYUN::ROS::Interface": {
      "ParameterGroups": [
        {
          "Parameters": [
            "VpcName",
            "VpcCidrBlock"
          ],
          "Label": {
            "default": "VPC"
          }
        },
        {
          "Parameters": [
            "VswName",
            "VswCidrBlock"
          ],
          "Label": {
            "default": "VSwitch"
          }
        }
      ],
      "TemplateTags": [
        "VPC-VSwitch"
      ]
    },
    "Parameters": {
      "VpcName": {
        "Type": "String",
        "Label": "Name",
        "Description": "The name must be 2 to 128 characters in length and can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter.",
        "Default": "MyVPC"
      },
      "VswName": {
        "Type": "String",
        "Label": "Name",
        "Description": "The name must be 2 to 128 characters in length and can contain letters, digits, underscores (_), and hyphens (-). It must start with a letter.",
        "Default": "MyVSwitch"
      }
    }
  }
}
```

```

    "VpcCidrBlock": {
      "Type": "String",
      "AllowedValues": [
        "10.0.0.0/8",
        "172.16.0.0/12",
        "192.168.0.0/16"
      ],
      "Description": "The CIDR block of the VPC. To customize the CIDR block, change the available values of this parameter in the template. The CIDR block must be contained within the range of available values.",
      "Label": "IPv4 CIDR block",
      "Default": "192.168.0.0/16"
    },
    "VswCidrBlock": {
      "Type": "String",
      "Description": "The configured CIDR block must be a subset of the CIDR block assigned to the VPC where the VSwitch resides and cannot be in use by other VSwitches.",
      "Label": "IPv4 CIDR block",
      "Default": "192.168.1.0/24"
    }
  },
  "Resources": {
    "VSwitch": {
      "Type": "ALIYUN::ECS::VSwitch",
      "Properties": {
        "VpcId": {
          "Ref": "VPC"
        },
        "ZoneId": {
          "Fn::Select": [
            "1",
            {
              "Fn::GetAZs": {
                "Ref": "ALIYUN::Region"
              }
            }
          ]
        }
      }
    },
    "CidrBlock": {
      "Ref": "VswCidrBlock"
    },
    "VSwitchName": {
      "Ref": "VswName"
    }
  },
  "VPC": {
    "Type": "ALIYUN::ECS::VPC",
    "Properties": {
      "CidrBlock": {
        "Ref": "VpcCidrBlock"
      },
      "VpcName": {
        "Ref": "VpcName"
      }
    }
  }
}

```

7 Custom resources

Custom resources allow you to write custom configuration logic in templates. Resource Orchestration Service (ROS) runs this logic anytime you create, update (if you have modified the custom resources), or delete stacks. For example, you can use custom resources to include resource types that are not supported in ROS. This way, you can manage all your related resources in a single stack.

You can use the `ALIYUN::ROS::CustomResource` or `Custom::MyCustomResourceTypeName` resource type to define custom resources in your templates. Custom resources require a property to specify the target to which ROS sends requests. The property is a service token that can be a Message Service (MNS) topic or queue, a Function Compute function, or an HTTP or HTTPS URL.

Custom resources must send responses to presigned response URLs. If they cannot send responses to ROS, ROS does not receive a response and the stack operation fails. `ResponseURL` is used in the Internet and `InnerResponseURL` is used in the internal network of Alibaba Cloud.

How custom resources work

Custom resource operations involve three parties.

- **template developer**
 - The template developer creates a template that includes a custom resource type. The template developer specifies the service token and all the input data in the template.
- **custom resource provider**
 - The custom resource provider owns the custom resource and determines how to handle and respond to requests from ROS. The custom resource provider must provide a service token for the template developer.
- **ROS**
 - During a stack operation, ROS sends a request to a service token specified in the template and waits for a response before proceeding.

The template developer and custom resource provider can be the same person or entity, and the process is the same. The following steps describe the general process:

1. The template developer defines a custom resource in a template. The template contains a service token and all input data parameters. The custom resource determines whether the input data parameters are required. The service token is always required.

The service token specifies the location to which ROS sends requests, such as to an MNS topic ARN or to a Function Compute function ARN. For more information, see [#unique_123](#). The service token and the structure of the input data are defined by the custom resource provider.

2. When a template is used to create, update, or delete a custom resource, ROS sends a request to the specified service token. There are no limits on the region of the service token.

A ROS request contains information such as the request type and a presigned URL to which the custom resource sends the request. For more information, see [Custom resource request objects](#).

The following sample data shows the content that a ROS request contains:

```
{
  "RequestType": "Create",
  "RequestId": "unique id for this create request",
  "ResponseURL": "pre-signed-url-for-create-response",
  "InnerResponseURL": "pre-signed-inner-url-for-create-response",
  "ResourceType": "Custom::MyCustomResourceType",
  "LogicalResourceId": "name of resource in template",
  "StackId": "stack id",
  "StackName": "stack name",
  "ResourceOwnerId": "resource owner id",
  "CallerId": "caller id",
  "RegionId": "region id",
  "ResourceProperties": {
    "key1": "string",
    "key2": [ "list" ],
    "key3": { "key4": "map" }
  }
}
```

3. The custom resource provider processes the ROS request and returns a SUCCESS or FAILED response to the presigned URL. The custom resource provider provides the response URL that contains JSON-formatted response data.

The custom resource provider can also include name-value pairs that the template developer can access in the response. For example, the response can contain output

data if the request succeeds or an error message if the request fails. For more information, see [Custom resource response objects](#).

The custom resource provider is responsible for listening and responding to the request. For example, the custom resource provider must listen and respond to MNS topic notifications that are sent to a specific topic ARN. ROS waits and listens for a response in the presigned URL location.

The following sample data shows the content that a custom resource can include in a response:

```
{
  "Status": "SUCCESS",
  "RequestId": "unique id for this create request (copied from request)",
  "LogicalResourceId": "name of resource in template (copied from request)",
  "StackId": "stack id (copied from request)",
  "PhysicalResourceId": "required vendor-defined physical id that is unique for that vendor",
  "Data": {
    "keyThatCanBeUsedInGetAtt1": "data for key 1",
    "keyThatCanBeUsedInGetAtt2": "data for key 2"
  }
}
```

4. After receiving a SUCCESS response, ROS proceeds with the stack operation. If a FAILED response or no response is returned, it indicates that the operation fails. All the output data from the custom resource is stored in the presigned URL location. The template developer can retrieve the data by using the Fn::GetAtt function.

7.1 Overview

You can use custom resources to customize logic configuration in templates. ROS runs the logic configuration each time you create, update, or delete stacks. If you want to include resources that are not available as ROS resource types, you can use custom resources to include those resources. This way, you can manage all your related resources in a single stack.

Custom resources

You can use ALIYUN::ROS::CustomResource or Custom::MyCustomResourceTypeName to define custom resources in templates. Custom resources require one property: the service token that specifies where the ROS requests are sent. The service token can be a Function Compute function, an MNS topic, an MNS queue, or an HTTP or HTTPS URL.

Custom resources must send responses to a presigned URL. If they cannot send responses to ROS, ROS will not receive a response and the stack operation fails. ResponseURL is used

to retrieve responses over the Internet. `IntranetResponseURL` is used to retrieve responses over the internal network.

How custom resources work

Any action taken for a custom resource involves three parties.

- **Template developer:** creates a template that includes a custom resource type. The template developer specifies the service token and all input parameters in the template.
- **Custom resource provider:** owns the custom resource and determines how to handle and respond to requests from ROS. The custom resource provider must provide a service token that the template developer uses.
- **ROS:** sends a request to a specified service token in the template during a stack operation, and then waits for a response before proceeding with the stack operation.

The template developer and custom resource provider can be the same person or entity, but the process is same. The following steps describe the general process:

1. The template developer defines a custom resource in a template that includes a service token and any input parameters. Depending on the custom resource, the input parameters may be required while the service token is always required.

The service token specifies where ROS sends requests to, such as to an MNS topic ARN or to a Function Compute function ARN. For more information, see [#unique_123](#). The service token and the structure of the input data is defined by the custom resource provider.

2. When you use templates to create, update, or delete custom resources, ROS sends requests to specified service tokens. The service token can be used in any regions.

In the request, ROS includes information such as the request type and a presigned URL, to which the custom resource sends responses. For more information, see [Custom resource request objects](#).

The following sample data shows what ROS includes in a request.

```
{
  "RequestType": "Create",
  "RequestId": "unique id for this create request",
  "ResponseURL": "pre-signed-url-for-create-response",
  "IntranetResponseURL": "pre-signed-intranet-url-for-create-response",
  "ResourceType": "Custom::MyCustomResourceType",
  "LogicalResourceId": "name of resource in template",
  "StackId": "stack id",
  "StackName": "stack name",
  "ResourceOwnerId": "resource owner id",
  "CallerId": "caller id",
```

```
"RegionId": "region id",
"ResourceProperties" : {
  "key1" : "string",
  "key2" : [ "list" ],
  "key3" : { "key4" : "map" }
}
```

3. The custom resource provider processes the ROS request and returns a response of SUCCESS or FAILED to the presigned URL. The custom resource provider provides the response in a JSON-formatted file and uploads it to the presigned URL.

In the response, the custom resource provider can also include name-value pairs that the template developer can access. For example, the response can include output data if the request succeeds or an error message if the request fails. For more information about responses, see [Custom resource response objects](#).

The custom resource provider is responsible for listening and responding to the request. For example, for Alibaba Cloud MNS notifications, the custom resource provider must listen and respond to notifications that are sent to a specific topic ARN. ROS waits and listens for a response in the presigned URL location.

The following sample data shows what a custom resource can include in a response:

```
{
  "Status": "SUCCESS",
  "RequestId": "unique id for this create request (copied from request)",
  "LogicalResourceId": "name of resource in template (copied from request)",
  "StackId": "stack id (copied from request)",
  "PhysicalResourceId": "required vendor-defined physical id that is unique for that vendor",
  "Data": {
    "keyThatCanBeUsedInGetAtt1": "data for key 1",
    "keyThatCanBeUsedInGetAtt2": "data for key 2"
  }
}
```

4. After obtaining a SUCCESS response, ROS proceeds with the stack operation. If a FAILED response or no response is returned, the operation fails. Any output data from the custom resource is stored in the presigned URL location. The template developer can use the Fn::GetAtt function to retrieve that data.

7.2 Resource references

This section provides details about the following fields:

- The JSON request and response fields that are used in messages sent to and from ROS when a custom resource is provided.
- The required fields for requests to, and responses to, the custom resource provider in response to stack creation, update, and deletion.

Topics

- [Custom resource request objects](#)
- [Custom resource response objects](#)
- [Custom resource request types](#)

7.2.1 Custom resource references

This section provides details about the JSON request and response fields that are used in messages sent to and from ROS when a custom resource is provided. The section also provides details about the required fields for requests to, and responses to, the custom resource provider in response to stack creation, update, and deletion.

- [Custom resource request objects](#)
- [Custom resource response objects](#)
- [Custom resource request types](#)

7.2.2 Custom resource request objects

This topic describes the types and fields of requests from template developers.

Template developer request properties

The template developer uses [#unique_123](#) to specify a custom resource in a template.

ALIYUN::ROS::CustomResource contains three properties: ServiceToken, Parameters, and Timeout.

- ServiceToken
 - The service token that is obtained from the custom resource provider to access the service. The service token can be a Function Compute function, an MNS topic, an MNS queue, or an HTTP or HTTPS URL. The service token can be used in any regions.
 - Required: yes.
 - Type: string.
- Parameters
 - When you create, update, or delete stacks, all fields in the resource properties are sent to the custom resource provider in the ResourceProperties field of the request

- . The custom resource provider defines both the names and valid content of these fields.
- Required: no.
- Type: map.
- Timeout
 - The timeout period for ROS to wait for the callback notification from the custom resource provider. Valid values: 1 to 43200. Unit: seconds.
 - Required: no.
 - Type: number.
 - Default value: 60.

Request fields for the custom resource provider

These fields are sent in JSON requests from ROS to the custom resource provider.

- RequestType
 - The request type is set by the ROS stack operation (CreateStack, UpdateStack, or DeleteStack) that was initiated by the template developer for the stack that contains the custom resource.
 - Valid values: Create, Update, and Delete. For more information, see [Custom resource request types](#).
 - Required: yes.
 - Type: string.
- ResponseURL
 - The presigned Internet URL. The URL receives responses from the custom resource provider to ROS.
 - Required: yes.
 - Type: string.
- IntranetResponseURL
 - The presigned internal URL that can be used in ECS. The URL receives responses from the custom resource provider to ROS.
 - Required: yes.
 - Type: string.

- **StackId**
 - The ID that identifies the stack that contains custom resource.
 - Required: yes.
 - Type: string.
- **StackName**
 - The name of the stack that contains the custom resource.
 - Required: yes.
 - Type: string.
- **ResourceOwnerId**
 - The ID of the Alibaba Cloud account to which the stack with the custom resource belongs.
 - Required: yes.
 - Type: string.
- **CallerId**
 - The ID of the Alibaba Cloud account or RAM user that performs the stack operation.
 - Required: yes.
 - Type: string.
- **RegionId**
 - The region ID of the stack that contains the custom resource.
 - Required: yes.
 - Type: string.
- **RequestId**
 - The unique ID of the request.
 - You can use RequestId in combination with StackId to form a value that uniquely identifies a request on a particular custom resource.
 - Required: yes.
 - Type: string.

- **ResourceType**
 - The resource type that the template developer selects for the custom resource in the template. Custom resource type names can be up to 68 characters in length and can contain letters, digits, underscores (_), at signs (@), and hyphens (-).
 - Required: yes.
 - Type: string.
- **LogicalResourceId**
 - The name (logical ID) selected by the template developer for the custom resource in the ROS template. This field facilitates communication between the custom resource provider and the template developer.
 - Required: yes.
 - Type: string.
- **PhysicalResourceId**
 - The required physical ID defined by the custom resource provider. The ID is unique to the provider.
 - Required: This field is always sent with Update and Delete requests, and never sent with Create requests.
 - Type: string.
- **ResourceProperties**
 - This field contains the content of the Properties object sent by the template developer . The content is defined by the custom resource provider.
 - Required: yes.
 - Type: JSON object.
- **OldResourceProperties**
 - Used only for Update requests. This field contains the resource properties that were declared previous to the update request.
 - Required: This field is always sent with Update requests, and never sent with Create and Delete requests.
 - Type: JSON object.

7.2.3 Custom resource response objects

This topic describes response fields of custom resources.

Response header for a custom resource provider

The response header must contain the following fields:

- Content-type: Set the value to "application/json".
- Date: the request time in GMT format. Example: "Tue, 26 Nov 2019 08:46:44 GMT".

Response fields for a custom resource provider

A custom resource provider includes the following properties when it sends the JSON file to the presigned URL (ResponseURL or InnerResponseURL).

The total size of the response body cannot exceed 4,096 bytes.

- Status
 - The status value sent by the custom resource provider in response to a request generated by ROS.
 - The value must be either SUCCESS or FAILED.
 - Required: yes.
 - Type: string.
- Reason
 - The reason for a failure response.
 - Valid only when the Status field is set to FAILED.
 - Required only when the Status field is set to FAILED. Otherwise, it is optional.
 - Type: string.
- PhysicalResourceId
 - An identifier unique to the custom resource provider. The value can be up to 255 bytes in size. The value cannot be an empty string and must be identical for all responses for the same resource.
 - Valid only when the Status field is set to SUCCESS. This response value must be passed when you create a stack that contains a custom resource. This response value must be copied from the request when you update or delete a stack that contains a custom resource.
 - Required only when the Status field is set to SUCCESS. Otherwise, it is optional.
 - Type: string.

- **StackId**
 - The ID that identifies the stack that contains the custom resource. This response value must be copied from the request.
 - Required: yes.
 - Type: string.
- **RequestId**
 - The unique ID of the request. This response value must be copied from the request.
 - Required: yes.
 - Type: string.
- **LogicalResourceId**
 - The name (logical ID) selected by the template developer for the custom resource in the ROS template. This response value must be copied from the request.
 - Required: yes.
 - Type: string.
- **Data**
 - Optional. The custom resource provider-defined name-value pairs to send with the response. You can use `Fn::GetAtt` to access the values provided here by name in the template.
 - Required: no.
 - Type: JSON object.

7.3 Request types

When a template developer creates, updates, or deletes a stack that contains a custom resource, the request type is specified in the `RequestType` field of the provider request object sent by ROS.

Each request type has a particular set of fields that are sent with the request, including a response URL (`ResponseURL` or `InnerResponseURL`) provided by the custom resource provider. The provider must respond to the URL with either a `SUCCESS` or `FAILED` result within the timeout period (1 to 43,200 seconds). After the timeout period, the request times out. Each result also has a particular set of fields required by ROS.

This topic describes information and examples about the request and response fields for each request type.

Topics

- [Create](#)
- [Update](#)
- [Delete](#)

7.3.1 Custom resource request types

When a template developer creates, updates, or deletes a stack that contains a custom resource, the request type is specified in the `RequestType` field of the provider request object sent by ROS.

Each request type has a particular set of fields that are sent with the request, including a response URL (`ResponseURL` or `InnerResponseURL`) provided by the custom resource provider. The provider must respond to the URL with either a `SUCCESS` or `FAILED` result within the timeout period (1 to 43,200 seconds). After the timeout period, the request times out. Each result also has a particular set of fields required by ROS.

This section provides information and examples about the request and response fields for each request type.

Topics

- [Create](#)
- [Update](#)
- [Delete](#)

7.3.2 Create

When the template developer creates a stack that contains a custom resource, Resource Orchestration Service (ROS) sends a request where the `RequestType` field is set to `Create` to the custom resource provider.

Request

A create request contains the following fields:

- `RequestType`

This field is set to `Create`.

- `ResponseURL`

The presigned public URL. This URL receives responses from the custom resource provider to ROS.

- **IntranetResponseURL**

The presigned internal URL. You can use this URL in Elastic Compute Service (ECS). This URL receives responses from the custom resource provider to ROS.

- **StackId**

The ID of the stack that contains the custom resource.

- **StackName**

The name of the stack that contains the custom resource.

- **ResourceOwnerId**

The ID of the Alibaba Cloud account to which the stack with the custom resource belongs

.

- **CallerId**

The ID of the Alibaba Cloud account or RAM user that is used to perform this operation.

- **RegionId**

The region ID of the stack that contains the custom resource.

- **RequestId**

The unique ID of the request.

- **ResourceType**

The resource type selected by the template developer for the custom resource in the template. The name of the custom resource type can be up to 68 characters in length and can contain letters, digits, underscores (_), at signs (@), and hyphens (-).

- **LogicalResourceId**

The name (logical ID) selected by the template developer for the custom resource in the template.

- **ResourceProperties**

This field contains the parameters in the Properties section of the request sent by the template developer. The content is defined by the custom resource provider.

Example

```
{
  "RequestType" : "Create",
  "RequestId" : "unique id for this create request",
  "ResponseURL" : "pre-signed-url-for-create-response",
  "IntranetResponseURL" : "pre-signed-intranet-url-for-create-response",
  "ResourceType" : "Custom::MyCustomResourceType",
  "LogicalResourceId" : "name of resource in template",
```

```
"StackId" : "stack id",
"StackName" : "stack name",
"ResourceOwnerId" : "resource owner id",
"CallerId" : "caller id",
"RegionId" : "region id",
"ResourceProperties" : {
  "key1" : "string",
  "key2" : [ "list" ],
  "key3" : { "key4" : "map" }
}
```

Responses

Success

When the create request is sent, the custom resource provider must send a response that contains the following fields to ROS:

- Status

This field must be set to SUCCESS.

- RequestId

The unique ID of the request. The response value must be copied from the request.

- LogicalResourceId

The name (logical ID) selected by the template developer for the custom resource in the template. The response value must be copied from the request.

- StackId

The ID of the stack that contains the custom resource. The response value must be copied from the request.

- PhysicalResourceId

This value must be unique to the custom resource provider. The value can be up to 255 bytes in length. The value cannot be an empty string and must be identical for the same resource in all responses.

- Data

Optional. The name-value pairs to be sent in the response. These pairs are defined by the custom resource provider. You can use the Fn::GetAtt function to access the values provided here by name in the template.

Example

```
{
  "Status" : "SUCCESS",
  "RequestId" : "unique id for this create request (copied from request)",
```

```
"LogicalResourceId" : "name of resource in template (copied from request)",
"StackId" : "stack id (copied from request)",
"PhysicalResourceId" : "required vendor-defined physical id that is unique for that
vendor",
"Data" : {
  "keyThatCanBeUsedInGetAtt1" : "data for key 1",
  "keyThatCanBeUsedInGetAtt2" : "data for key 2"
}
}
```

Failed

When the create request fails, the custom resource provider must send a response that contains the following fields to ROS:

- Status

This field must be set to FAILED.

- Reason

The reason for the response failure.

- RequestId

The unique ID of the request. The response value must be copied from the request.

- LogicalResourceId

The name (logical ID) selected by the template developer for the custom resource in the template. The response value must be copied from the request.

- StackId

The ID of the stack that contains the custom resource. The response value must be copied from the request.

Example

```
{
  "Status" : "FAILED",
  "Reason" : "Required failure reason string",
  "RequestId" : "unique id for this create request (copied from request)",
  "LogicalResourceId" : "name of resource in template (copied from request)",
  "StackId" : "stack id (copied from request)"
}
```

7.3.3 Update

When changes are made to custom resource properties in a template, Resource

Orchestration Service (ROS) sends a request where the RequestType field is set to Update

to the custom resource provider. Custom resource code does not have to detect changes because it knows that its properties have changed after Update is called.

Request

A update request contains the following fields:

- RequestType

This field is set to Update.

- ResponseURL

The presigned public URL. This URL receives responses from the custom resource provider to ROS.

- IntranetResponseURL

The presigned internal URL. You can use this URL in Elastic Compute Service (ECS). This URL receives responses from the custom resource provider to ROS.

- StackId

The ID of the stack that contains the custom resource.

- StackName

The name of the stack that contains the custom resource.

- ResourceOwnerId

The ID of the Alibaba Cloud account to which the stack with the custom resource belongs .

- CallerId

The ID of the Alibaba Cloud account or RAM user that is used to perform this operation.

- RegionId

The region ID of the stack that contains the custom resource.

- RequestId

The unique ID of the request.

- ResourceType

The resource type selected by the template developer for the custom resource in the template. The name of the custom resource type can be up to 68 characters in length and can contain letters, digits, underscores (_), at signs (@), and hyphens (-).

- LogicalResourceId

The name (logical ID) selected by the template developer for the custom resource in the template.

- PhysicalResourceId

The required physical ID defined by the custom resource provider. This ID is unique to the provider.

- ResourceProperties

This field contains the parameters in the Properties section of the request sent by the template developer. The content is defined by the custom resource provider.

- OldResourceProperties

This field contains the original parameters in the Properties section previously defined by the template developer.

Example

```
{
  "RequestType": "Update",
  "RequestId": "unique id for this update request",
  "ResponseURL": "pre-signed-url-for-update-response",
  "IntranetResponseURL": "pre-signed-intranet-url-for-create-response",
  "ResourceType": "Custom::MyCustomResourceType",
  "LogicalResourceId": "name of resource in template",
  "PhysicalResourceId": "custom resource provider-defined physical id",
  "StackId": "stack id",
  "StackName": "stack name",
  "ResourceOwnerId": "resource owner id",
  "CallerId": "caller id",
  "RegionId": "region id",
  "ResourceProperties": {
    "key1": "new-string",
    "key2": [ "new-list" ],
    "key3": { "key4": "new-map" }
  },
  "OldResourceProperties": {
    "key1": "string",
    "key2": [ "list" ],
    "key3": { "key4": "map" }
  }
}
```

Responses

Success

When the update request is sent, the custom resource provider must send a response that contains the following fields to ROS:

- Status

This field must be set to SUCCESS.

- RequestId

The unique ID of the request. The response value must be copied from the request.

- LogicalResourceId

The name (logical ID) selected by the template developer for the custom resource in the template. The response value must be copied from the request.

- StackId

The ID of the stack that contains the custom resource. The response value must be copied from the request.

- PhysicalResourceId

The value of this field cannot be changed. The response value must be copied from the request.

- Data

Optional. The name-value pairs to be sent in the response. These pairs are defined by the custom resource provider. You can use the Fn::GetAtt function to access the values provided here by name in the template.

Example

```
{
  "Status": "SUCCESS",
  "RequestId": "unique id for this update request (copied from request)",
  "LogicalResourceId": "name of resource in template (copied from request)",
  "StackId": "stack id (copied from request)",
  "PhysicalResourceId": "custom resource provider-defined physical id",
  "Data": {
    "keyThatCanBeUsedInGetAtt1": "data for key 1",
    "keyThatCanBeUsedInGetAtt2": "data for key 2"
  }
}
```

Failed

When the update request fails, the custom resource provider must send a response that contains the following fields to ROS:

- Status

This field must be set to FAILED.

- Reason

The reason for the response failure.

- RequestId

The unique ID of the request. The response value must be copied from the request.

- LogicalResourceId

The name (logical ID) selected by the template developer for the custom resource in the template. The response value must be copied from the request.

- StackId

The ID of the stack that contains the custom resource. The response value must be copied from the request.

Example

```
{
  "Status": "FAILED",
  "Reason": "Required failure reason string",
  "RequestId": "unique id for this update request (copied from request)",
  "LogicalResourceId": "name of resource in template (copied from request)",
  "StackId": "stack id (copied from request)"
}
```

7.3.4 Delete

When the template developer deletes a stack that contains a custom resource, Resource Orchestration Service (ROS) sends a request where the `RequestType` field is set to `Delete` to the custom resource provider. To delete the stack, the custom resource provider must successfully respond to the delete request.

Request

A delete request contains the following fields:

- RequestType

This field is set to `Delete`.

- ResponseURL

The presigned public URL. This URL receives responses from the custom resource provider to ROS.

- IntranetResponseURL

The presigned internal URL. You can use this URL in Elastic Compute Service (ECS). This URL receives responses from the custom resource provider to ROS.

- StackId

The ID of the stack that contains the custom resource.

- StackName

The name of the stack that contains the custom resource.

- ResourceOwnerId

The ID of the Alibaba Cloud account to which the stack with the custom resource belongs
.

- CallerId

The ID of the Alibaba Cloud account or RAM user that is used to perform this operation.

- RegionId

The region ID of the stack that contains the custom resource.

- RequestId

The unique ID of the request.

- ResourceType

The resource type selected by the template developer for the custom resource in the template. The name of the custom resource type can be up to 68 characters in length and can contain letters, digits, underscores (_), at signs (@), and hyphens (-).

- LogicalResourceId

The name (logical ID) selected by the template developer for the custom resource in the template.

- PhysicalResourceId

The required physical ID defined by the custom resource provider. This ID is unique to the provider.

- ResourceProperties

This field contains the parameters in the Properties section of the request sent by the template developer. The content is defined by the custom resource provider.

Example

```
{
  "RequestType" : "Delete",
  "RequestId" : "unique id for this delete request",
  "ResponseURL" : "pre-signed-url-for-delete-response",
  "IntranetResponseURL" : "pre-signed-intranet-url-for-create-response",
  "ResourceType" : "Custom::MyCustomResourceType",
  "LogicalResourceId" : "name of resource in template",
```

```
"PhysicalResourceId" : "custom resource provider-defined physical id",
"StackId" : "stack id",
"StackName" : "stack name",
"ResourceOwnerId" : "resource owner id",
"CallerId" : "caller id",
"RegionId" : "region id",
"ResourceProperties" : {
  "key1" : "string",
  "key2" : [ "list" ],
  "key3" : { "key4" : "map" }
}
```

Responses

Success

When the delete request is sent, the custom resource provider must send a response that contains the following fields to ROS:

- Status

This field must be set to SUCCESS.

- RequestId

The unique ID of the request. The response value must be copied from the request.

- LogicalResourceId

The name (logical ID) selected by the template developer for the custom resource in the template. The response value must be copied from the request.

- StackId

The ID of the stack that contains the custom resource. The response value must be copied from the request.

- PhysicalResourceId

The value of this field cannot be changed. The response value must be copied from the request.

Example

```
{
  "Status" : "SUCCESS",
  "RequestId" : "unique id for this delete request (copied from request)",
  "LogicalResourceId" : "name of resource in template (copied from request)",
  "StackId" : "stack id (copied from request)",
  "PhysicalResourceId" : "custom resource provider-defined physical id"
}
```

Failed

When the delete request fails, the custom resource provider must send a response that contains the following fields to ROS:

- Status

This field must be set to FAILED.

- Reason

The reason for the response failure.

- RequestId

The unique ID of the request. The response value must be copied from the request.

- LogicalResourceId

The name (logical ID) selected by the template developer for the custom resource in the template. The response value must be copied from the request.

- StackId

The ID of the stack that contains the custom resource. The response value must be copied from the request.

Example

```
{
  "Status" : "FAILED",
  "Reason" : "Required failure reason string",
  "RequestId" : "unique id for this delete request (copied from request)",
  "LogicalResourceId" : "name of resource in template (copied from request)",
  "StackId" : "stack id (copied from request)"
}
```

8 Change set management

8.1 Data structure

This topic describes the data structure of a change set.

Change

Parameter	Type	Description	
ResourceChange	Structure	The resource that ROS will change and operation that ROS will perform.	
Type	String	The type of the object to which the changes are made. Only <code>Resource</code> is supported. Set the value to <code>Resource</code> .	

ResourceChange

Parameter	Type	Description	
Action	String	The operation that you want to perform. Valid values: <ul style="list-style-type: none">• <code>Add</code>: creates resources.• <code>Modify</code>: modifies resources.• <code>Remove</code>: releases resources.	

Parameter	Type	Description
Details	Array	The detailed changes that are made to resources. This parameter is displayed only when the Action parameter is set to Modify .
LogicalResourceId	String	The logical ID of the resource as defined in the template.
PhysicalResourceId	String	The physical ID of the resource. When the Action parameter is set to Add , no physical ID is available because the resource has not been created.

Parameter	Type	Description	
Replacement	String	<p>Specifies whether to replace resources by creating new resources and deleting the original ones when the Action parameter is set to Modify. The value of this parameter depends on the RequiresRecreation value in the ResourceTargetDefinition structure. Examples:</p> <ul style="list-style-type: none"> • If the RequiresRecreation parameter is set to Always and the Evaluation parameter is set to Static, the value of the Replacement parameter is True. • If the RequiresRecreation parameter is set to Always and the Evaluation parameter is set to Dynamic, the value of the Replacement parameter is Conditionally. <p>If there are multiple changes on a single resource and each change has a different value for the RequiresRecreation parameter, the value of the Replacement</p>	

Parameter	Type	Description	
ResourceType	String	The type of the ROS resource.	
Scope	String array	The parameter that triggers updates when the Action parameter is set to Modify. Valid values: <ul style="list-style-type: none">• Properties• Metadata• DeletionPolicy	

ResourceChangeDetail

Parameter	Type	Description	
ChangeSource	String	<p>The reason for triggering updates. Valid values:</p> <ul style="list-style-type: none"> ResourceReference: The referenced physical ID of another resource may be changed. ParameterReference: The referenced parameter is changed. ResourceAttribute: The referenced output attribute of another resource may be changed. DirectModification: The template is modified. Automatic: If the nested stack created by using the ALIYUN::ROS::Stack resource has not been modified, ROS sets the ChangeSource parameter to Automatic. This is because the template of the nested stack may have already been changed. ROS does not update to the nested stack template before you update the parent stack. 	
		<ul style="list-style-type: none"> System: Specific conditions or internal 	

Parameter	Type	Description	
CausingEntity	String	<p>The objects associated with the ChangeSource parameter. The mappings are as follows:</p> <ul style="list-style-type: none"> • When the ChangeSource parameter is set to ResourceReference, the CausingEntity value indicates the resource name. • When the ChangeSource parameter is set to ParameterReference, the CausingEntity value indicates the parameter name. • When the ChangeSource parameter is set to ResourceAttribute, the CausingEntity value indicates the attribute name, in the "resource name.attribute name" format. • When the ChangeSource parameter is set to DirectModification, the CausingEntity parameter is empty by default. • When the ChangeSource parameter is set to Automatic, the CausingEntity 	

Parameter	Type	Description
Evaluation	String	<p>Specifies whether ROS can determine the target values and whether original values are changed to target values before a change set is executed. Valid values:</p> <ul style="list-style-type: none">• Static When this parameter is set to Static, ROS can determine the target values and original values are changed to target values.• Dynamic When this parameter is set to Dynamic, ROS cannot determine the target values. When ROS updates the stack, the target values depend on the results of internal functions such as Ref or Fn::GetAtt.
Target	Structure	The detailed information about parameters that trigger updates.

ResourceTargetDefinition

Parameter	Type	Description
Attribute	String	Specifies the parameter that triggers updates. Valid values: <ul style="list-style-type: none">• Properties• Metadata• DeletionPolicy
Name	String	When the Attribute parameter is set to Properties, this parameter takes effect and the value indicates the specific attribute name. Otherwise, the value is set to null.
RequiresRecreation	String	Specifies whether attribute changes lead to resource recreation when the Attribute parameter is set to Properties. Valid values: <ul style="list-style-type: none">• Never• Conditionally• Always

8.2 Create a stack

ROS allows you to create stacks by using change sets. You can create change sets by calling an API operation or using Alibaba Cloud CLI. You can check and modify the stacks before executing change sets. The new stacks will only become valid after the change sets have been executed.

Create a stack by calling an API operation

You can call the [#unique_192](#) operation to create a change set for a new stack and call the [#unique_193](#) operation to preview the stack configuration.

If a stack is created when you create a change set, the stack will be in the `REVIEW_IN_PROGRESS` state. You cannot create new change sets for a stack in this state. If the new stack does not meet the configuration requirements, you can delete the stack and recreate a change set to create another stack.

Create a stack through Alibaba Cloud CLI

You can use the **aliyun ros CreateChangeSet** command to create a stack.

When you create a stack by creating a change set, you must set the `ChangeSetType` parameter to `CREATE` and specify the stack name, template, stack parameters, and change set name. In this example, a stack named `test-create-change-set` and a change set named `test-create-change-set` are created.

```
aliyun ros CreateChangeSet --ChangeSetType CREATE --TemplateURL oss://ros-templates/test-change-set.json? RegionId=cn-hangzhou --StackName test-create-change-set --ChangeSetName test-create-change-set --Parameters.1.ParameterKey Count --Parameters.1.ParameterValue 1
```

8.3 Update a stack

8.3.1 Overview

You can use the change set feature to update running stacks. You can preview how proposed changes to a stack may affect your running resources. ROS makes changes to stacks only after you execute change sets. You can use the ROS console, Alibaba Cloud CLI, or ROS API to create and manage change sets.

Limits

The limits on change sets are as follows:

- A stack can contain up to 20 change sets.
- Change sets only reflect stack changes. They do not indicate whether stacks have been updated.
- Change sets do not check whether the upper limit of your account has been reached, whether resources that cannot be updated will be updated, or whether your account has sufficient permissions to modify resources. Each of these limitations can cause stack updates to fail. If stack updates fail, ROS attempts to roll back your resources to their original status.

Stack update process

The process of using change sets to update a stack is as follows:

1. Create a change set by submitting changes for the stack that you want to update. You can submit a modified template or modified input parameter values. ROS compares your stack with the submitted changes and generates a change set.
2. View the stack settings and resources that will be changed in the change set. For example, you can view the resources that ROS will add, modify, or delete.
3. Optional. If you need to make additional changes before you execute a change set, create additional change sets. Creating multiple change sets can help you understand and evaluate how different changes will affect your resources. You can also delete change sets that are no longer needed to avoid executing inapplicable change sets accidentally.
4. Execute the change set that you need to apply to your stack.

**Notice:**

After you execute a change set, other change sets associated with the stack will become invalid and be deleted.

5. The stack is being updated. After the update is complete, you can view the update result.

Features

The following table describes the features of change sets.

Feature	Description
Create a change set	If you need to create a change set for a running stack, follow the instructions in this topic to modify the template or template parameters.
View a change set	After you create a change set, you can view the change records and JSON-formatted change details.
Execute a change set	After you create a change set for a stack, you must execute the change set to update the stack.
Delete a change set	If a change set is no longer needed, you can delete it to avoid unexpected or accidental stack changes.

8.3.2 Create a change set

If you need to create a change set for a running stack, follow the instructions in this topic to modify the template or template parameters. ROS generates a change set by comparing the stack with template changes that you submitted.

Prerequisites

Make sure that you have created a stack. For more information, see [#unique_200](#).

Create a change set in the ROS console

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Stacks**.
3. Click **Create Change Set** in the Actions column corresponding to a stack name.

You can also click a stack ID in the **Stack Name** column. On the stack management page, click the **Change Sets** tab. Click **Create Change Set**.

4. In the **Select Template** step of the **Create Change Set** wizard, select a template and click **Next**.

Homepage / Stacks / Create Change Set

← Create Change Set

1 Select Template 2 Configure Template Parameters 3 Configure Change Set 4 Confirm

Specify Template
A template is a JSON or YAML file that describes the resources and properties of a stack.

Select an Existing Template Use a Sample Template

Template Import Method
☐ Use URL ☒ Enter Template Content ☐ My Templates ☐ Upload Template

Template Content
JSON YAML

```
1 {
2   "ROSTemplateFormatVersion": "2015-09-01",
3   "Resources": {
4     "oss-triggertrigger-fnfoss-t": {
5       "Type": "ALIYUN::FC::Trigger",
6       "Properties": {
7         "TriggerConfig": {
8           "filter": {
9             "Key": {
10              "Prefix": "source/",
11              "Suffix": ""
12            }
13          }
14        }
15      }
16    }
17  }
```

Next Cancel

5. In the **Configure Template Parameters** step of the **Create Change Set** wizard, configure **Change Set Name** and the parameters and click **Next**.

The screenshot shows the 'Create Change Set' wizard interface. At the top, the breadcrumb 'Homepage / Stacks / Create Change Set' is visible. The main heading is 'Create Change Set' with a back arrow. Below the heading is a progress bar with four steps: 1. Select Template (completed), 2. Configure Template Parameters (current step, highlighted in blue), 3. Configure Change Set, and 4. Confirm. The main content area is divided into two sections. The first section, 'Change Set Name', has a red asterisk indicating a required field. It contains a text input field with a blurred placeholder and a note: 'The name can be up to 255 characters in length and can contain letters, digits, hyphens (-), and underscores (_). It must start with a letter.' The second section, 'Parameters', contains a table with one row: 'FunExecuteVersion' with a value of '3' in the input field. At the bottom, there are five buttons: 'Previous', 'Create Change Set' (highlighted in blue), 'Next', 'Preview', and 'Cancel'.

Homepage / Stacks / Create Change Set

← Create Change Set

1 Select Template 2 **Configure Template Parameters** 3 Configure Change Set 4 Conf

* Change Set Name

The name can be up to 255 characters in length and can contain letters, digits, hyphens (-), and underscores (_). It must start with a letter.

Parameters

FunExecuteVersion	3
-------------------	---

Previous **Create Change Set** Next Preview Cancel

6. In the **Configure Change Set** step of the **Create Change Set** wizard, configure **Stack Policy**, **Rollback on Failure**, and **Timeout Period**, and then click **Next**.

Homepage / Stacks / Create Change Set

← Create Change Set

✓ Select Template

✓ Configure Template Parameters

3 Configure Change Set

4 Conf

Stack Policy (Optional)

☒ No Stack Policy ☐ Input Stack Policy

Rollback on Failure

☐ Enabled

Timeout Period(Rollback is performed when the stack is not created or updated within this period.)

The value is denoted in minutes, and must be a positive integer ranging from 10 to 1440.

Previous

Create Change Set

Next

Cancel

7. In the **Confirm step of the **Create Change Set** wizard, click **Create Change Set**.**

Homepage / Stacks / Create Change Set

← Create Change Set

✓ Select Template

✓ Configure Template Parameters

✓ Configure Change Set

4 Confirm

Configure Template

Stack Name
detectMaskTest_ChangeSet_2020-04-26

Resource Stack Description
-

Parameters

Key	Value
FunExecuteVersion	3

Configured Stack Options

Stack Policy (Optional)	-
Temporary Stack Policy (Optional)	-
Rollback on Failure	Disabled
Timeout Period	1,440

Previous

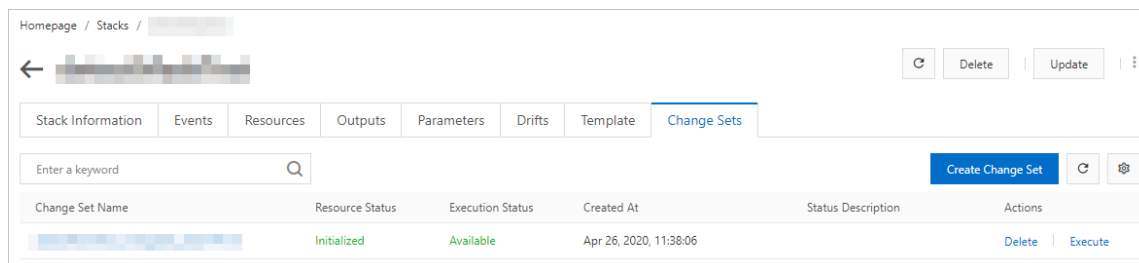
Create Change Set

Cancel

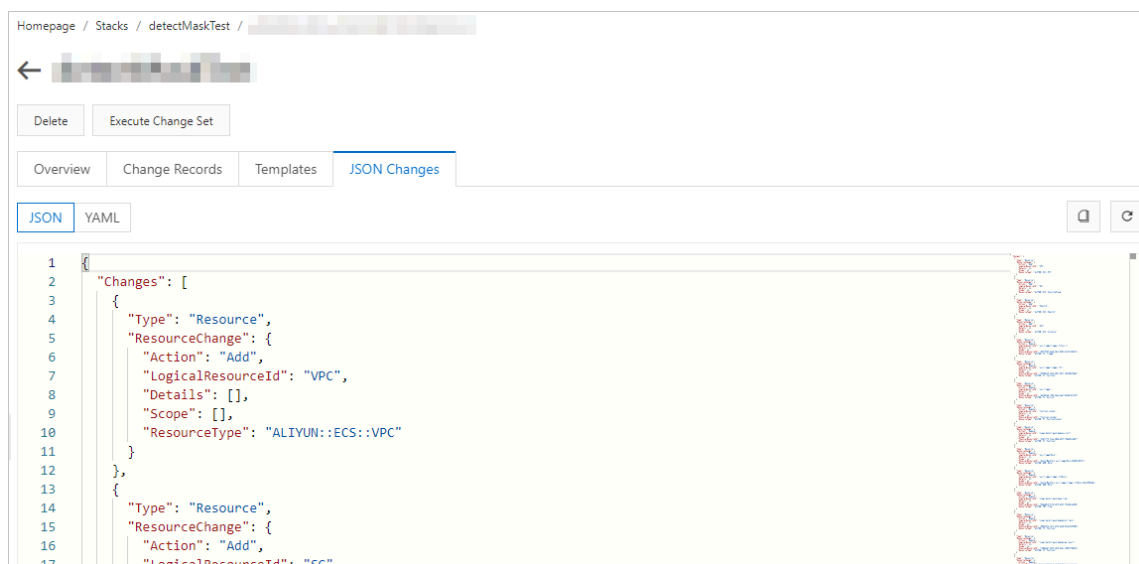
8. View the change set of the stack.

- On the **Stacks** page, click the stack ID in the Stack Name column.
- On the stack management page, click **Change Sets**.

You can view the name, resource status, and execution status of the change set.



You can view the details of the change set.



Create a change set through Alibaba Cloud CLI

You can use the **aliyun ros CreateChangeSet** command to create a change set.

You can specify new parameter values or modify parameters by using command options and submit the template changes. In the following example, a change set named test-

change-set is created for a stack by using the current template (oss://ros-templates/test-change-set.json? RegionId=cn-hangzhou):

```
aliyun ros CreateChangeSet --TemplateURL oss://ros-templates/test-change-set.json
? RegionId=cn-hangzhou --StackId <stack_id> -- ChangeSetName test-change-set --
Parameters.1.ParameterKey Count --Parameters.1.ParameterValue 1
```

8.3.3 View a change set

After you create a change set, you can view the change records and JSON-formatted change details.

Prerequisites

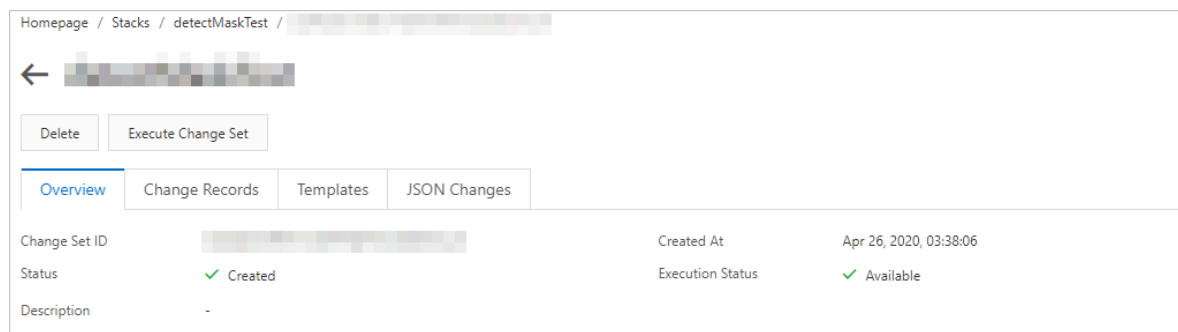
Make sure that you have created a change set. For more information, see [Create a change set](#).

View a change set in the ROS console

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Stacks**.
3. Click a stack ID in the **Stack Name** column. On the stack management page, click **Change Sets** to view the change sets of the stack.
4. Click the name of the target change set.

On the change set details page, view the basic information, change records, templates, and JSON-formatted change details of the change set.

The **Change Records** and **JSON Changes** tabs display the changes in the template. If you need to make additional changes to the template, you can create additional change sets.



View a change set through Alibaba Cloud CLI

1. Use the **aliyun ros ListChangeSets** command to view the ID of the stack to which the change set belongs.

View the stack ID of the change set as follows:

```
aliyun ros ListChangeSets --StackId <stack_id> --RegionId <region_id>
```

The returned information of the stack to which the change set belongs is as follows:

```
{
  "TotalCount": 1,
  "PageSize": 10,
  "RequestId": "A94A31B7-EC3A-4528-90D8-FA31FA4D13BB",
  "PageNumber": 1,
  "ChangeSets": [
    {
      "Status": "CREATE_COMPLETE",
      "ChangeSetId": "<change_set_id>",
      "ExecutionStatus": "AVAILABLE",
      "CreateTime": "2020-03-03T06:36:20",
      "ChangeSetType": "UPDATE",
      "RegionId": "cn-hangzhou",
      "ChangeSetName": "test-change-set",
      "StackName": "test-change-set",
      "StackId": "<stack_id>"
    }
  ]
}
```

2. Use the **aliyun ros GetChangeSet** command to view the change set ID.

View the change set ID as follows:

```
aliyun ros ListChangeSets --StackId <stack_id> --RegionId <region_id>
```

```
aliyun ros GetChangeSet --ChangeSetId <change_set_id> --RegionId <region_id>
```

The returned change set information is as follows:

```
{
  "ExecutionStatus": "AVAILABLE",
  "Parameters": [
    {
      "ParameterValue": "<account_id>",
      "ParameterKey": "ALIYUN::AccountId"
    },
    {
      "ParameterValue": "None",
      "ParameterKey": "ALIYUN::NoValue"
    },
    {
      "ParameterValue": "cn-hangzhou",
      "ParameterKey": "ALIYUN::Region"
    }
  ]
}
```

```

        "ParameterValue": "<stack_id>",
        "ParameterKey": "ALIYUN::StackId"
    },
    {
        "ParameterValue": "test-change-set",
        "ParameterKey": "ALIYUN::StackName"
    },
    {
        "ParameterValue": "<tenant_id>",
        "ParameterKey": "ALIYUN::TenantId"
    },
    {
        "ParameterValue": "1",
        "ParameterKey": "Count"
    }
],
"TimeoutInMinutes": 60,
"Changes": [
    {
        "Type": "Resource",
        "ResourceChange": {
            "LogicalResourceId": "WaitConditionHandle",
            "Replacement": "False",
            "PhysicalResourceId": "WaitConditionHandle",
            "ResourceType": "ALIYUN::ROS::WaitConditionHandle",
            "Action": "Modify",
            "Details": [
                {
                    "Evaluation": "Static",
                    "Target": {
                        "Name": "Count",
                        "RequiresRecreation": "Never",
                        "Attribute": "Properties"
                    },
                    "CausingEntity": "Count",
                    "ChangeSource": "ParameterReference"
                },
                {
                    "Evaluation": "Dynamic",
                    "Target": {
                        "Name": "Count",
                        "RequiresRecreation": "Never",
                        "Attribute": "Properties"
                    },
                    "ChangeSource": "DirectModification"
                }
            ],
            "Scope": [
                "Properties"
            ]
        }
    }
],
"ChangeSetId": "<change_set_id>",
"StackId": "<stack_id>",
"DisableRollback": false,
"ChangeSetName": "test-change-set",
"ChangeSetType": "UPDATE",
"StackName": "test-change-set",
"Status": "CREATE_COMPLETE",
"CreateTime": "2020-03-03T06:36:20",
"RegionId": "cn-hangzhou",
"RequestId": "DB9B48C8-C22D-4009-A3B0-85FDF3D26D2D"

```

```
}
```

The Changes property lists changes made to the resources. For more information, see [Data structure](#).

8.3.4 Execute a change set

After you create a change set for a stack, you must execute the change set to update the stack.

Prerequisites

Make sure that you have created a change set. For more information, see [Create a change set](#).

Context



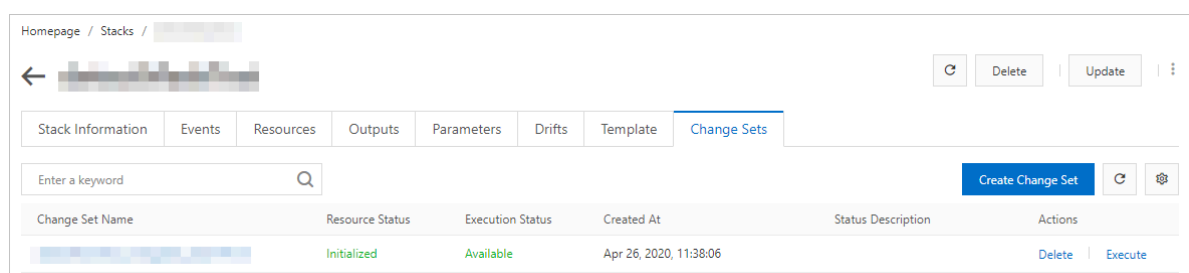
Notice:

After you execute a change set for a stack, other change sets associated with the stack will become invalid and be deleted. If the stack fails to be updated, you must create a new change set.

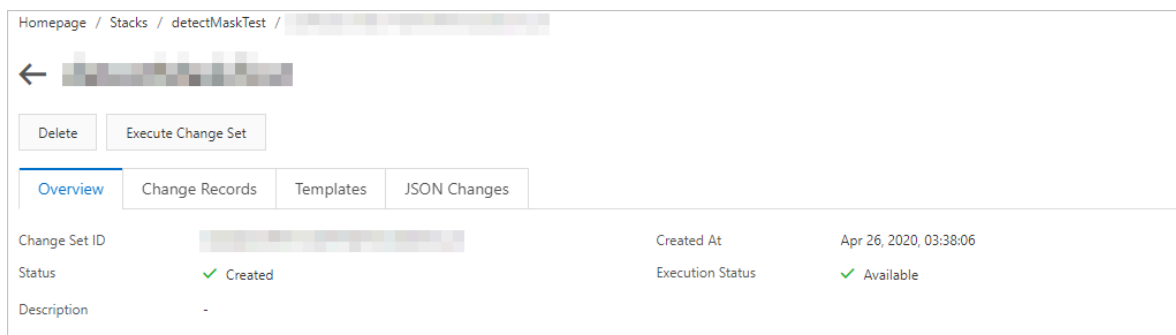
If you execute a change set on a stack that has a stack policy, the policy will be executed when the stack is updated. You cannot specify a temporary stack policy that overwrites an existing policy when you execute a change set. To update protected resources, you must update the stack policy or directly update the stack.

Execute a change set in the ROS console

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Stacks**.
3. Click a stack ID in the **Stack Name** column. On the stack management page, click **Change Sets**.
4. On the **Change Sets** tab, find the change set that you want to execute and click **Execute** in the **Actions** column. ROS immediately begins updating the stack.



5. Optional. On the Change Sets tab, click the name of the change set that you want to execute. On the change set management page, click **Execute Change Set**. ROS immediately begins updating the stack.



Execute a change set through Alibaba Cloud CLI

You can use the **aliyun ros ExecuteChangeSet** command to execute a change set.

Specify the ID of the change set that you want to execute as follows:

```
aliyun ros ExecuteChangeSet --ChangeSetId <change_set_id> --RegionId <region_id>
```

After you run this command, ROS starts updating the stack. If you need to view the stack update progress, use the **aliyun ros GetStack** command.

8.3.5 Delete a change set

If a change set is no longer needed, you can delete it to avoid unexpected or accidental stack changes. ROS will retain all change sets that have not been deleted until the stack is updated.

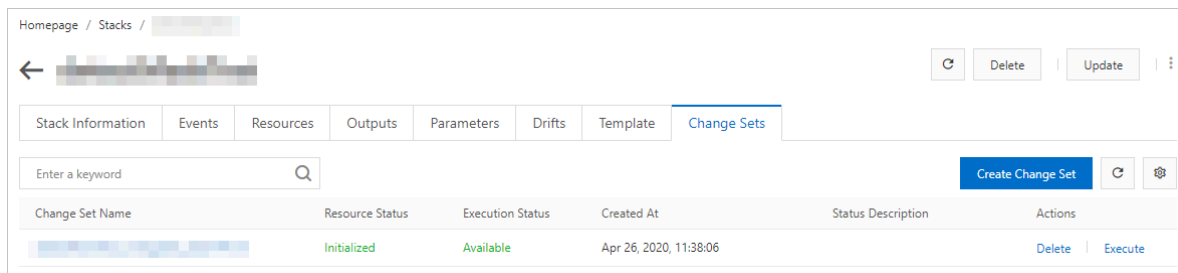
Prerequisites

Make sure that you have created a change set. For more information, see [Create a change set](#).

Delete a change set in the ROS console

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Stacks**.
3. Click a stack ID in the **Stack Name** column. On the stack management page, click **Change Sets**.

- On the **Change Sets** tab, find the change set that you want to delete and click **Delete** in the **Actions** column.



- In the message that appears, click **OK**.

Delete a change set through Alibaba Cloud CLI

You can use the **aliyun ros DeleteChangeSet** command to delete a change set.

Specify the ID of the change set that you want to delete as follows:

```
aliyun ros DeleteChangeSet --ChangeSetId <change_set_id> --RegionId <region_id>
```

9 Drift detection

9.1 Overview

Resource Orchestration Service (ROS) provides the drift detection feature to identify the configuration changes in your stacks that are beyond the control of ROS. You can then take corrective measures to re-synchronize resources with their template definitions. For example, you can update the affected resources or use the template correction function to correct the template. Drift correction helps you ensure that changes in resources can be identified and the relevant actions can be taken.

You can make direct changes to the resources in a stack without updating the template in ROS. For example, you can use the ECS console to update an ECS instance that is created as part of an ROS resource stack. Changes made directly to the resources in a stack can cause issues when you want to update or delete the stack. You can use the drift detection feature to identify the configuration changes of resources in a stack that are made beyond the management scope of ROS.

The drift detection feature enables you to detect whether the actual configuration of a stack differs or has drifted from its expected configuration. You can use ROS to detect drift on an entire stack or on individual resources within the stack. For example, you can detect whether a property or resource of the stack is deleted.

- A resource is considered to have drifted if one of its actual property values differs from the expected property value.
- A stack is considered to have drifted if one or more of its resources have drifted.

To determine whether a resource has drifted, ROS compares the expected resource property values defined in the template with the actual values of the resource properties.

**Note:**

A resource is considered to have drifted if one or more of its properties values have been deleted or modified. ROS generates details on each resource in the stack that has drifted.

Precautions

ROS only detects drift on resources that support drift detection. Resources that do not support drift detection are assigned the NOT_CHECKED state. For more information, see [Resource types that support drift detection](#).

You can detect drift on stacks in one of the following states:

- CREATE_COMPLETE
- UPDATE_COMPLETE
- ROLLBACK_COMPLETE
- ROLLBACK_FAILED
- CHECK_COMPLETE

When detecting drift on a stack, ROS does not detect drift on any nested stacks that belong to this stack. To detect drift on nested stacks, you must initiate a drift detection operation on the nested stacks.

ROS determines drift of property values either through templates or specifying template parameters. Drift detection does not detect the default values of resource properties. If you want to track the drift of a resource property, you must explicitly set the property value even if the value is the same as the default value.

To detect drift on stacks, you must have the following permissions:

- Read permission on resources that support drift detection in the stack. For example, if the stack contains an ALIYUN::VPC::EIP resource, you must have the vpc:DescribeEipAddresses permission to detect drift on the stack.
- To detect drift on stacks, you must have the ros:DetectStackDrift permission.
- To detect drift on resources, you must have the ros:DetectStackResourceDrift permission.

In some cases, ROS may not be able to return accurate drift detection results. We recommend that you familiarize yourself with these cases to avoid incorrectly interpreting drift detection results.

- In some cases, objects contained in property arrays are reported as drift. In fact, these are default values provided to the properties from the underlying service responsible for the resource.

- You can specify certain resource properties in your template. If ROS cannot compare these properties with the actual properties in the stack resources, these properties are not included in drift detection results. The types of such properties are as follows:
 - Properties that ROS cannot map back to their actual resource properties in the template.
 - Property values that the service responsible for the resource does not return.
 - Property values that are designed to never be returned by the service responsible for the resource. These property values may contain confidential information such as passwords or other sensitive data that must not be exposed.
 - Resource properties that are not supported by ROS.

You can query whether the resource properties support drift detection. For more information, see [#unique_204](#). For example, you can query the return values of the ALIYUN::ESS::ScalingRule resource. In the return values, the last `SupportDriftDetection` field indicates whether the resource supports drift detection. If the value of this field is `true`, the resource supports drift detection and each property contains a `SupportDriftDetection` field that indicates whether the property supports drift detection.

```
{
  ...
  "ResourceType": "ALIYUN::ESS::ScalingRule",
  "Properties": {
    "ScalingRuleName": {
      ...
      "SupportDriftDetection": true
    },
    ...
  },
  "SupportDriftDetection": true
}
```

Drift detection status codes

The following section describes the status types of drift detection:

- Drift detection operation status: describes the current status of the drift detection operation.


- Drift status of stack groups, stack instances, or stacks.
 - Stack group drift status: describes the drift status of a stack group based on the drift status of the stack instances that belong to the group.
 - Stack instance drift status: describes the drift status of a stack instance based on the drift status of the stack associated with the instance.
 - Stack drift status: describes the drift status of a stack based on the drift status of its resources.
- Resource drift status: describes the drift status of an individual resource.

The following table describes the status codes assigned by ROS to stack drift detection operations.

Drift detection operation status	Description
DETECTION_COMPLETE	The stack drift detection operation has been successfully completed for all resources in the stack that support drift detection.
DETECTION_FAILED	The stack drift detection operation has failed for at least one resource in the stack.
DETECTION_IN_PROGRESS	The stack drift detection operation is in progress.

The following table describes the drift status codes assigned by ROS to stacks.

Drift status	Description
DRIFTED	<ul style="list-style-type: none">• For a stack: The stack differs or has drifted from its expected configuration. A stack is considered to have drifted if one or more of its resources have drifted.• For a stack instance: The stack instance is considered to have drifted if the stack associated with it has drifted.• For a stack group: The stack group is considered to have drifted if one or more of its stack instances have drifted.
NOT_CHECKED	ROS has not detected whether the stack, stack instance, or stack group differs from its expected configuration.

Drift status	Description
IN_SYNC	<p>The current configuration of each resource that supports drift detection matches its expected configuration.</p> <div> Note: Stacks, stack instances, or stack groups are also assigned the IN_SYNC state when they have no resources that support drift detection.</div>

The following table describes the drift status codes assigned by ROS to stack resources.

Resource drift status	Description
DELETED	The resource differs from its expected configuration because the resource has been deleted.
MODIFIED	The resource differs from its expected configuration.
NOT_CHECKED	ROS has not detected whether the resource differs from its expected configuration.
IN_SYNC	The current configuration of the resource matches its expected configuration.

The following table describes the different types of status codes assigned by ROS to resource properties that differ from their expected configuration.

Property difference type	Description
ADD	A value has been added to a resource property that is of the array or list data type.
REMOVE	The property has been removed from the current resource configuration.
NOT_EQUAL	The current property value differs from the expected value as defined in the template.

References

The following table describes the topics related to drift detection.

Topic	Description
Detect drift on a stack	You can perform drift detection on a stack to determine whether the stack has drifted from its expected configuration. The operation returns the details about the drift status of the resources in the stack that support drift detection.
Detect drift on a resource	You can perform drift detection on individual resources in a stack to determine whether the resources have drifted from the expected configuration.
Resource types that support drift detection	ROS detects drift only on resources that support drift detection.
Detect drift on a stack group	You can detect whether any stack instances in a stack group differ from their expected configurations or have drifted.
Correct drift on a stack	Stack drift correction helps ensure the consistency between resource configurations and their template definitions. Changes can be synchronized between stack configurations and template definitions.

9.2 Detect drift on a stack

You can perform drift detection on a stack to determine whether the stack has drifted from its expected configuration. The operation returns the details about the drift status of the resources in the stack that support drift detection.

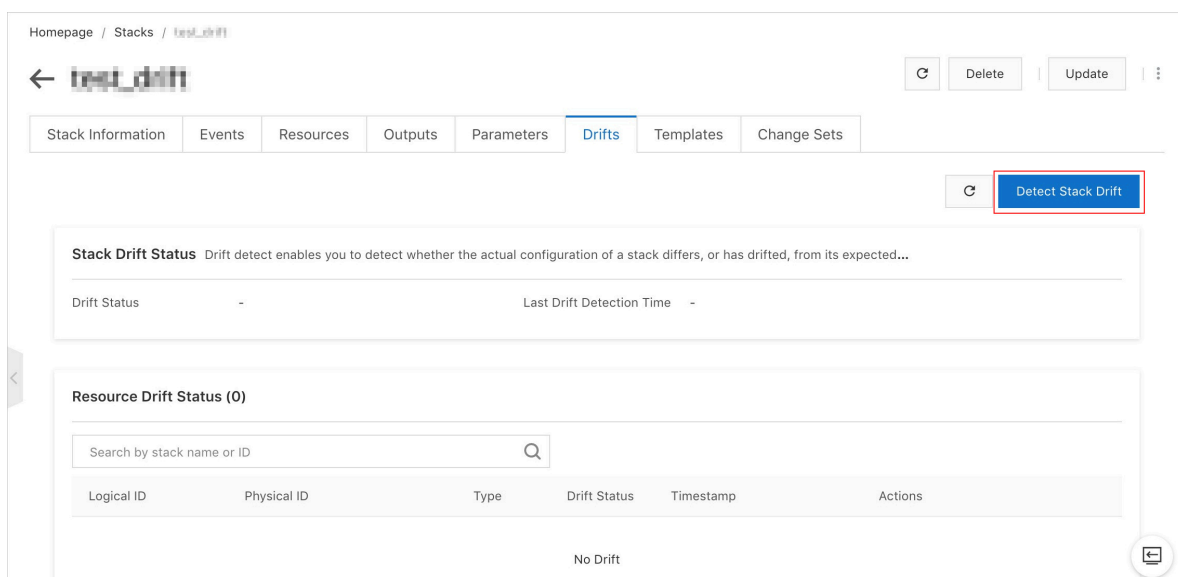
Prerequisites

Make sure that you have created a stack. For more information, see [#unique_200](#).

Detect drift in the ROS console

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Stacks**.
3. On the **Stacks** page, click the ID under the stack name.
4. Click the **Drifts** tab.

5. On the **Drifts** tab, click **Detect Stack Drift**.



Note:

- ROS displays a prompt, indicating that a drift detection operation is initiated for the selected stack.
- You can perform only one drift detection operation on a single stack at a time.
- The drift detection operation may take several minutes. The actual time taken for a drift detection operation to complete depends on the number of resources in the stack. ROS continues the drift detection operation even after you close the information window.

6. View drift detection results.

- In the **Basic Information** section of the **Stack Information** tab, view the drift status and last drift detection time of the stack.

Homepage / Stacks / test_drift

← test_drift

Refresh Delete Update

Stack Information Events Resources Outputs Parameters Drifts Templates Change Sets

Basic Information

Stack Name	test_drift	Region	cn-beijing
Stack ID	test_drift-2020-03-31-19:46:39	Created At	Mar 31, 2020, 19:46:39
Timeout (Minutes)	60	Rollback on Failure	Yes
Status	Created	Status Description	Stack CREATE completed successfully
Drift Status	Drifted Correct	Last Drift Detect Time	Mar 31, 2020, 19:49:56

Tags: Edit

- In the **Stack Drift Status** section of the **Drifts** tab, view the drift status and last drift detection time of the stack.

Homepage / Stacks / test_drift

← test_drift

Refresh Delete Update

Stack Information Events Resources Outputs Parameters Drifts Templates Change Sets

Refresh Detect Stack Drift

Stack Drift Status Drift detect enables you to detect whether the actual configuration of a stack differs, or has drifted, from its expected...

Drift Status	Drifted Correct	Last Drift Detection Time	Mar 31, 2020, 19:49:56
--------------	-----------------	---------------------------	------------------------

Resource Drift Status (1)

Search by stack name or ID

Logical ID	Physical ID	Type	Drift Status	Timestamp	Actions
ScalingGroup	test_drift-2020-03-31-19:46:39	ALIYUN::ESS	Drifted	Mar 31, 2020, 19:49:56	View Drift Details Detect Resource Drift

- In the **Resource Drift Status** section of the **Drifts** tab, click **View Drift Details** in the Actions column corresponding to the resource. The physical ID, drift status, type, and last drift detection time of the resource are displayed.

Homepage / Stacks / Resource stack details / **ScalingGroup**


← **ScalingGroup** Drift Details Detect Resource Drift

Resource Drift Overview

Physical ID	aliyun-ros-2015-01-06-123456789012	Drift Status	Drifted
Type	ALIYUN::ESS::ScalingGroup	Last Drift Detection Time	Mar 31, 2020, 19:49:56

Difference (1)

<input type="checkbox"/>	Property	Change	Expected Value	Current Value
<input type="checkbox"/>	/ScalingGroupName	Drifted	test1	test2

- 
Note:
 A stack is considered to have drifted if one or more of its resources have drifted.

Detect drift through Alibaba Cloud CLI

You can use the following aliyun ros commands through Alibaba Cloud Command Line Interface (Alibaba Cloud CLI) to detect drift on a stack.

Subcommand	Description
DetectStackDrift	Initiates a drift detection operation on a stack.
GetStackDriftDetectionStatus	Monitors the status of the drift detection operation on a stack.
ListStackResourceDrifts	Queries detailed information about resource drift in a stack.

- You can use the `DetectStackDrift` subcommand to detect drift on a stack. The stack ID must be specified. Resource names can also be specified to filter resources on which you want to detect drift.

```
$ aliyun ros DetectStackDrift --StackId 4334b961-3bfd-419e-9a00-23a95e*****
{
  "DriftDetectionId": "13b48934-6818-4765-8ae1-744241*****",
  "RequestId": "B288A0BE-D927-4888-B0F7-B35EF84B6E6F"
}
```

- The stack drift detection operation may take a long period of time. You can use the `GetStackDriftDetectionStatus` subcommand to monitor the status of the drift detection

operation. This subcommand obtains the stack drift detection ID returned by the `DetectStackDrift` subcommand.

In the following example, the drift detection ID of the stack returned by the `DetectStackDrift` subcommand in the preceding example is passed as a parameter to the `GetStackDriftDetectionStatus` subcommand. This parameter returns operation details, indicating that the drift detection operation is completed.

```
$ aliyun ros GetStackDriftDetectionStatus --StackDriftDetectionId 13b48934-6818-4765-8ae1-744241*****
{
  "RequestId": "52398D3A-E868-4F95-8B5E-6A2DFB778B16",
  "DriftDetectionTime": "2020-03-17T07:21:17",
  "DriftDetectionStatusReason": "Detect stack drift successfully",
  "DriftedStackResourceCount": 2,
  "DriftDetectionStatus": "DETECTION_COMPLETE",
  "StackDriftStatus": "DRIFTED",
  "DriftDetectionId": "13b48934-6818-4765-8ae1-744241*****",
  "StackId": "4334b961-3bfd-419e-9a00-23a95e*****"
}
```

- After the stack drift detection operation is completed, you can use the `ListStackResourceDrifts` subcommand to view the results, including actual and expected property values for resources that have drifted.

```
$ aliyun ros ListStackResourceDrifts --StackId 4334b961-3bfd-419e-9a00-23a95e*****
{
  "ResourceDrifts": [
    {
      "ResourceDriftStatus": "MODIFIED",
      "LogicalResourceId": "Vpc1",
      "PropertyDifferences": [
        {
          "ActualValue": "test11",
          "PropertyPath": "/Description",
          "ExpectedValue": "test1",
          "DifferenceType": "NOT_EQUAL"
        }
      ],
      "PhysicalResourceId": "vpc-m5euqfvmzygb7xq*****",
      "ExpectedProperties": "{\"CidrBlock\": \"192.168.0.0/16\", \"Description\": \"test1\", \"VpcName\": \"test1\"}",
      "DriftDetectionTime": "2020-03-17T07:21:17",
      "ResourceType": "ALIYUN::ECS::VPC",
      "ActualProperties": "{\"CidrBlock\": \"192.168.0.0/16\", \"Description\": \"test11\", \"VpcName\": \"test1\"}",
      "StackId": "4334b961-3bfd-419e-9a00-23a95e*****"
    },
    {
      "ResourceDriftStatus": "DELETED",
      "LogicalResourceId": "Vpc2",
      "PhysicalResourceId": "vpc-m5exf3skxrxtvk*****",
      "DriftDetectionTime": "2020-03-17T07:21:17",
      "ResourceType": "ALIYUN::ECS::VPC",
      "StackId": "4334b961-3bfd-419e-9a00-23a95e*****"
    }
  ],
}
```

```
"RequestId": "8E1DE57B-6124-482B-8283-EF5562653308"
}
```

9.3 Detect drift on a resource

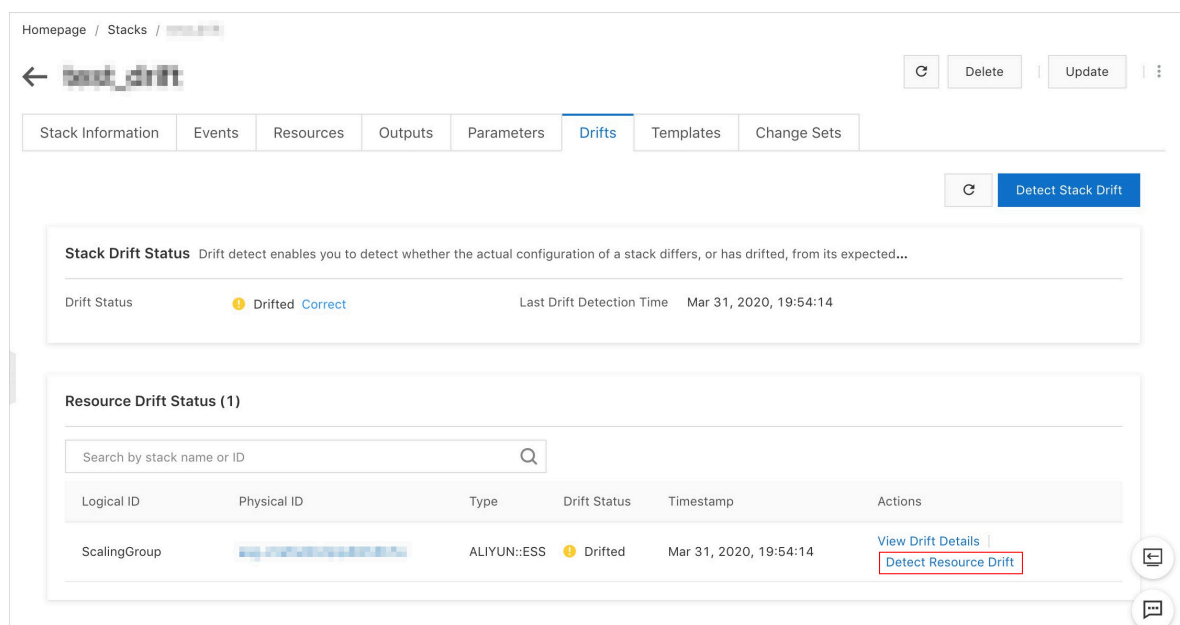
You can perform drift detection on individual resources in a stack to determine whether the resources have drifted from the expected configuration.

Prerequisites

Make sure that you have performed a drift detection operation on the entire stack. For more information, see [Detect drift on a stack](#).

Detect drift in the ROS console

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Stacks**.
3. On the **Stacks** page, click the ID under the stack name.
4. Click the **Drifts** tab.
5. In the **Resource Drift Status** section, click **Detect Resource Drift** in the Actions column corresponding to the resource on which you want to detect drift.



6. In the **Resource Drift Status** section, click **View Drift Details** in the Actions column corresponding to the resource. The physical ID, drift status, type, and last drift detection time of the resource are displayed.

Homepage / Stacks / Resource stack details / ScalingGroup Drift Details

← ScalingGroup Drift Details Detect Resource Drift

Resource Drift Overview

Physical ID	ALIYUN::ESS::ScalingGroup	Drift Status	Drifted
Type	ALIYUN::ESS::ScalingGroup	Last Drift Detection Time	Mar 31, 2020, 19:54:14

Difference (1)

Property	Change	Expected Value	Current Value
/ScalingGroupName	Drifted	test1	test3

Detect drift through Alibaba Cloud CLI

You can use the `aliyun ros DetectStackResourceDrift` command through Alibaba Cloud Command Line Interface (Alibaba Cloud CLI) to detect drift on an individual resource and view the logical resource ID and the stack to which the resource belongs.

```
$ aliyun ros DetectStackResourceDrift --StackId 4334b961-3bfd-419e-9a00-23a95e*****
--LogicalResourceId Vpc1
{
  "ResourceDriftStatus": "MODIFIED",
  "LogicalResourceId": "Vpc1",
  "PropertyDifferences": [
    {
      "ActualValue": "test11",
      "PropertyPath": "/Description",
      "ExpectedValue": "test1",
      "DifferenceType": "NOT_EQUAL"
    }
  ],
  "RequestId": "A488767B-7440-4A74-81FD-BCF91A2EE1BB",
  "PhysicalResourceId": "vpc-m5euqfvmzygb7xq*****",
  "ExpectedProperties": "{\"CidrBlock\": \"192.168.0.0/16\", \"Description\": \"test1\", \"VpcName\": \"test1\"}",
  "DriftDetectionTime": "2020-03-17T08:35:34",
  "ResourceType": "ALIYUN::ECS::VPC",
  "ActualProperties": "{\"CidrBlock\": \"192.168.0.0/16\", \"Description\": \"test11\", \"VpcName\": \"test1\"}",
  "StackId": "4334b961-3bfd-419e-9a00-23a95e*****"
}
```

```
}
```

9.4 Resource types that support drift detection

This topic lists the resources types that support drift detection.

**Note:**

More and more resource types will support drift detection in the future.

Service	Resource type
ECS	ALIYUN::ECS::VPC ALIYUN::ECS::VSwitch ALIYUN::ECS::NatGateway ALIYUN::ECS::SecurityGroup
ESS	ALIYUN::ESS::ScalingRule ALIYUN::ESS::ScalingConfiguration ALIYUN::ESS::ScalingGroup ALIYUN::ESS::ScalingGroupEnable
RAM	ALIYUN::RAM::Role
SLB	ALIYUN::SLB::Listener ALIYUN::SLB::LoadBalancer ALIYUN::SLB::VServerGroup ALIYUN::SLB::BackendServerAttachment
VPC	ALIYUN::VPC::EIP ALIYUN::VPC::EIPAssociation

9.5 Detect drift on a stack group

You can detect whether any stack instances in a stack group differ from their expected configurations or have drifted.

Prerequisites

Make sure that you have created a stack group. For more information, see [Create a stack group](#).

Context

When ROS performs drift detection on stack groups, stacks associated with stack instances in stack groups are also detected for drift. ROS compares the current configuration with the expected configuration of each resource in the stack based on the template and specified parameters. If the current configuration differs from the expected configuration, the resource is considered to have drifted.

- A stack and its associated stack instance are considered to have drifted if one or more of its resources have drifted.
- A stack group is considered to have drifted if one or more of its stack instances have drifted.

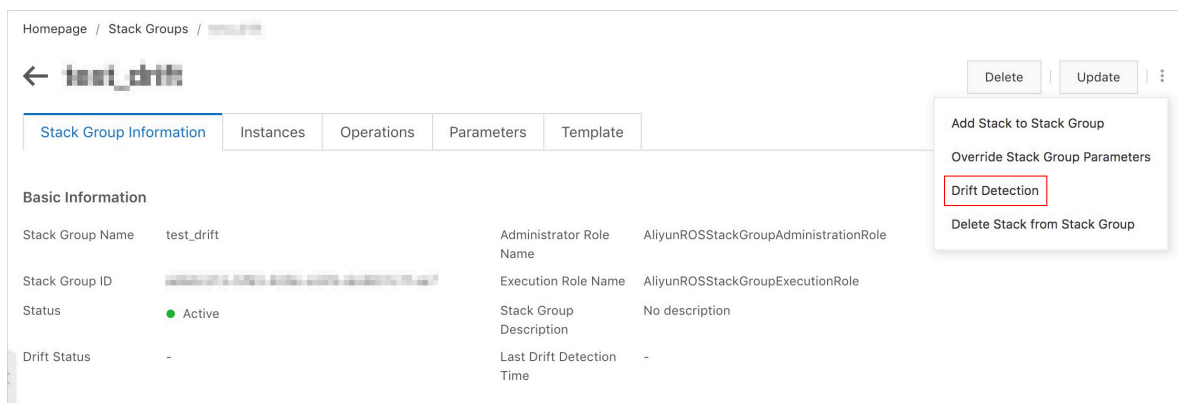
Drift refers to changes in stack configurations not performed by using ROS. Changes made by using ROS to individual stacks directly are not considered as drift. These changes do not include changes made to stack groups. Assume that you have a stack that is associated with a stack instance in a stack group. If you use ROS to update the stack based on a different template, the stack is not considered to have drifted even if this stack has a different template from other stacks in the same stack group. This is because the stack still matches its expected template and parameter configurations in ROS.

During the drift detection on a stack group, each stack in the stack group is detected for drift. Property values that overwrite the original ones specified in the template are used by ROS for drift detection on the stack. If you detect drift directly on stacks associated with stack instances, you cannot view the drift results on the **Stack Groups** page.

Detect drift in the ROS console

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Stack Groups**.
3. On the **Stack Groups** page, click the ID under the stack group name.

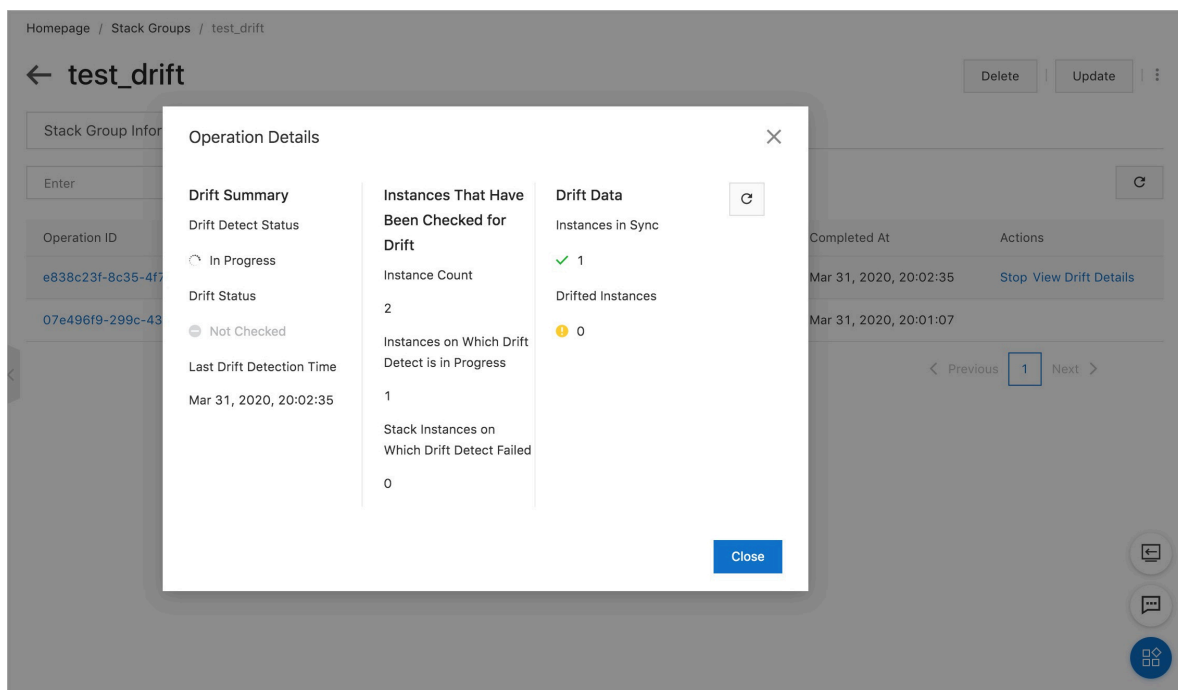
4. On the **Stack Group Information** tab, choose  > **Drift Detection**.



Note:

ROS displays a prompt, indicating that a drift detection operation is initiated for the selected stack group.

5. In the **Drift Detection** dialog box that appears, configure **Maximum Number of Concurrent Accounts** and **Fault Tolerance** and click **OK**.
6. Optional. Click the **Operations** tab, find the drift detection operation, click **View Drift Details**, and view the progress of the drift detection operation.



Note:

- You can perform only one drift detection operation on a single stack group at a time. ROS continues the drift detection operation even after you close the information window.
- The drift detection operation may take several minutes. The actual time taken for a drift detection operation to complete depends on the number of stack instances and resources in the stack group.

7. Click the **Instances** tab to view the drift detection results.

Homepage / Stack Groups / **test_drift**

← **test_drift** Delete Update ⋮

Stack Group Information **Instances** Operations Parameters Template

Enter

Stack Instance Account	Stack Instance Region	Stack ID	Instance Status	Status Reason	Drift Status	Last Drift Detection Time
1754580903499898	cn-beijing	1754580903499898-test-drift-stack	● Current	No description	✓ In Sync	Mar 31, 2020, 20:02:30
1754580903499898	cn-hangzhou	1754580903499898-test-drift-stack	● Current	No description	✓ In Sync	Mar 31, 2020, 20:02:35

< Previous **1** Next >



Note:

You can view IDs of the stacks associated with stack instances in the **Stack ID** column. You can view the drift status of stacks in the **Drift Status** column. A stack is considered to have drifted if one or more of its resources have drifted.

To view the drift detection results of a stack associated with a specified stack instance, you can record the Alibaba Cloud account of the stack instance and the stack name and region. Then you can log on to the Alibaba Cloud account to which the stack instance belongs to view the drift detection results. For more information, see [Detect drift on a stack](#).

Detect drift through Alibaba Cloud CLI

You can use the following `aliyun ros` commands through Alibaba Cloud Command Line Interface (Alibaba Cloud CLI) to detect drift on a stack group.

Subcommand	Description
DetectStackGroupDrift	Initiates a drift detection operation on a stack group.

Subcommand	Description
GetStackGroupOperation	Monitors the status of the drift detection operation on a stack group.
StopStackGroupOperation	Stops a drift detection operation on a stack group.

After the drift detection operation is completed, you can use the following subcommands to query the required drift information:

- You can use the `GetStackGroup` subcommand to query the detailed information about the stack group, including details about the last completed drift detection operation on the stack group. The information about the drift detection operations in progress is not included.
- You can use the `ListStackInstances` subcommand to query the list of stack instances in a stack group, including the drift status and last drift detection time of each instance.
- You can use the `GetStackInstance` subcommand to query the detailed information about a specified stack instance, including its drift status and last drift detection time.

9.6 Correct drift on a stack

Stack drift correction helps ensure the consistency between resource configurations and their template definitions. Changes can be synchronized between stack configurations and template definitions.

Context

You can use one of the following methods to correct drift on a stack:

- You can update stacks to synchronize resource configurations with their template definitions. For more information, see [#unique_209](#).
- You can correct templates to synchronize template definitions with the current resource configurations. This topic describes how to correct templates.

Correct templates in the ROS console

1. Log on to the [ROS console](#).
2. In the left-side navigation pane, click **Stacks**.
3. On the **Stacks** page, click the ID under the stack name.
4. Click the **Drifts** tab.
5. On the **Drifts** tab, click **Detect Stack Drift**.

6. In the Stack Drift Status section, click **Correct.**

The screenshot shows the 'Stack Drift Status' section of the console. At the top, there's a breadcrumb 'Homepage / Stacks / test_stack' and a back arrow. Below the breadcrumb are tabs: 'Stack Information', 'Events', 'Resources', 'Outputs', 'Parameters', 'Drifts' (selected), 'Templates', and 'Change Sets'. On the right, there are buttons for 'Refresh', 'Delete', 'Update', and a menu icon. Below the tabs is a 'Detect Stack Drift' button. The main content area has a 'Stack Drift Status' section with a description: 'Drift detect enables you to detect whether the actual configuration of a stack differs, or has drifted, from its expected...'. Below this, it shows 'Drift Status' as 'Drifted' with a yellow warning icon and a red box around the 'Correct' button. The 'Last Drift Detection Time' is 'Mar 31, 2020, 19:54:14'. Below this is a 'Resource Drift Status (1)' section with a search bar and a table. The table has columns: 'Logical ID', 'Physical ID', 'Type', 'Drift Status', 'Timestamp', and 'Actions'. The table contains one row for 'ScalingGroup' with physical ID 'asg-2zefiu9n2qqs9xih4b7a', type 'ALIYUN::ESS', drift status 'Drifted' with a yellow warning icon, and timestamp 'Mar 31, 2020, 19:54:14'. The 'Actions' column has links for 'View Drift Details' and 'Detect Resource Drift'. There are also chat and help icons on the right.

Homepage / Stacks / test_stack

← test_stack

Stack Information Events Resources Outputs Parameters **Drifts** Templates Change Sets

Refresh Delete Update

Refresh Detect Stack Drift

Stack Drift Status Drift detect enables you to detect whether the actual configuration of a stack differs, or has drifted, from its expected...

Drift Status Drifted **Correct** Last Drift Detection Time Mar 31, 2020, 19:54:14

Resource Drift Status (1)

Search by stack name or ID

Logical ID	Physical ID	Type	Drift Status	Timestamp	Actions
ScalingGroup	asg-2zefiu9n2qqs9xih4b7a	ALIYUN::ESS	Drifted	Mar 31, 2020, 19:54:14	View Drift Details Detect Resource Drift

**Note:**

The **Correct** button is displayed only for stacks that have drifted.

7. In the Drift Correction dialog box, select the resources to be corrected.

The screenshot shows the 'Drift Correction' dialog box. It has a title bar with a close button. Inside, there's a 'Resources' section with a 'Preview' button. Below this is a table with columns: 'Logical ID', 'Physical ID', 'Drift Status', and 'Timestamp'. The table contains one row for 'ScalingGroup' with physical ID 'asg-2zefiu9n2qqs9xih4b7a', drift status 'Drifted' with a yellow warning icon, and timestamp 'Mar 31, 2020, 19:54:14'. The 'Logical ID' and 'Physical ID' columns have checkboxes, and the 'ScalingGroup' row is selected. At the bottom right, there are 'OK' and 'Cancel' buttons.

Drift Correction

Resources Preview

Logical ID	Physical ID	Drift Status	Timestamp	
<input checked="" type="checkbox"/>	ScalingGroup	asg-2zefiu9n2qqs9xih4b7a	Drifted	Mar 31, 2020, 19:54:14

OK Cancel

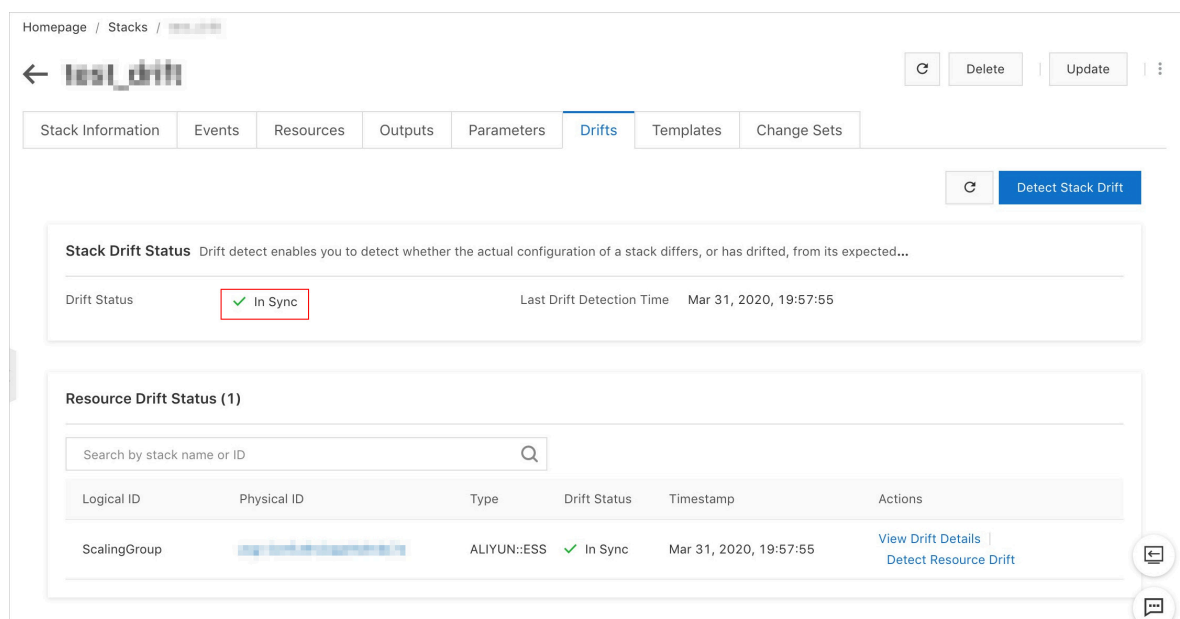
8. In the **Drift Correction** dialog box, click **Preview** to compare the template content before and after correction.

The template content before correction is displayed on the left side. The template content after correction is displayed on the right side.



9. Click **OK**.

After the correction is completed, perform a drift detection operation again and view the detection results.



Correct templates through Alibaba Cloud CLI

You can use the `aliyun ros UpdateStackTemplateByResources` command through Alibaba Cloud Command Line Interface (Alibaba Cloud CLI) to correct templates. The parameters

are the same as those of the **UpdateStackTemplateByResources** operation. For more information, see [#unique_210](#).

```
$ aliyun ros UpdateStackTemplateByResources --StackId 4334b961-3bfd-419e-9a00-23a95e*****
{
  "RequestId": "B288A0BE-D927-4888-B0F7-B35EF84B6E6F",
  "NewTemplateBody": "{\"ROSTemplateFormatVersion\": \"2015-09-01\", \"Resources\": {\"Vpc\": {\"Type\": \"ALIYUN::ECS::VPC\", \"Properties\": {\"VpcName\": \"test\", \"CidrBlock\": \"192.168.0.0/16\", \"Description\": \"test2\"}}}, \"Outputs\": {\"VpcId\": {\"Value\": {\"Fn::GetAtt\": [\"Vpc\", \"VpcId\"]}}}}",
  "OldTemplateBody": "{\"ROSTemplateFormatVersion\": \"2015-09-01\", \"Resources\": {\"Vpc\": {\"Type\": \"ALIYUN::ECS::VPC\", \"Properties\": {\"VpcName\": \"test\", \"CidrBlock\": \"192.168.0.0/16\", \"Description\": \"test1\"}}}, \"Outputs\": {\"VpcId\": {\"Value\": {\"Fn::GetAtt\": [\"Vpc\", \"VpcId\"]}}}}"
```