CloudFormation Tutorial

Version: ZStack 3.10.0

Issue: V3.10.0

ZStack

Copyright Statement

Contents

Copyright Statement I
1 Introduction 1
2 Preparations 2
3 Typical Practice Workflow 3
4 Resource Stack 4
5 Stack Template 11
6 Sample Template 18
7 Designer 21
8 Typical Scenario-based Practice 23
8.1 Sample Template-based Practice 23
8.2 Designer-based Practice 33
9 Appendices 41
9.1 Stack Template Syntax 41
9.1.1 Parameters 42
9.1.2 Resources 45
9.1.3 Outputs 49
9.1.4 Functions 51
9.1.5 Mappings 61
9.2 Resource Index 63
9.2.1 Resource 63
9.2.2 Action 64
Glossary 66

1 Introduction

ZStack CloudFormation is a service that helps you simplify the cloud computing resource management and automate the deployment and O&M. With a resource stack template, you can define what cloud resources you need, the dependency between the resources, and the resource configuration. With the CloudFormation engine, CloudFormation can provide automatic batch deployment and resource configuration, as well as easy lifecycle management of cloud resources. You can also use API and SDK to integrate the automatic O&M capabilities.

The advantages of CloudFormation are as follows:

1. You only need to create a stack template or modify an existing one to define what cloud resources you need, the dependency between the resources, and the resource configuration. With the CloudFormation engine, CloudFormation will automatically complete the creation and configuration of all resources.
2. The Cloud provides sample templates and a designer to create stack templates quickly.
3. You can dynamically update a stack template based on your business needs, and then you can update the related resource stack to flexibly meet the needs of business development.
4. If you no longer need a resource stack, you can simply one-click delete it, which also deletes all of the resources in the stack.
5. You can reuse an existing stack template to quickly duplicate all stack resources without repeated configuration.
6. You can flexibly combine cloud services based on different scenarios to meet the needs of automatic maintenance.

2 Preparations

This Tutorial will elaborate on how to use CloudFormation.

Before you start using CloudFormation, install the latest version of ZStack and complete the initialization wizard.

For more information, see the installation and deployment section in the *User Guide*.

3 Typical Practice Workflow

By using CloudFormation, you can create a stack template that describes all the resources you want, and CloudFormation can quickly create and configure those resources. You can also easily manage the collection of resources as a single unit.

The typical workflow of CloudFormation is as follows:

1. Prepare a stack template.
   - First, you need to prepare a stack template. CloudFormation will create and configure a related resource stack based on the template prepared.
   - You can check whether the sample template provided by the Cloud meets the business needs in advance. If it meets the needs, you can use the sample template directly to create a related resource stack.
   - If the sample template does not meet the business needs, you can create a new template or modify an existing one. For more information about how to create a custom template, see the *Stack Template* topic. Furthermore, you can use the designer to fast create a custom template. For more information about the designer, see the *Designer* topic.
2. Create a resource stack.
   - If the sample template meets the business needs, you can use the sample template directly to create a resource stack. For more information about how to create a resource stack by using a sample template, see the *Sample Template* topic.
   - You can also create a resource stack by using a custom template. For more information about how to create a resource stack by using a custom template, see the *Stack Template* topic. Furthermore, you can use the designer to fast create a resource stack. For more information about the designer, see the *Designer* topic.
3. Manage the resource stack.
   - CloudFormation provides lifecycle management of the resource stack.
   - You can check the information of all resources in the resource stack.
   - When you delete a resource stack, you delete the resource stack and all of its resources.
   - For more information about the resource stack management, see the *Resource Stack* topic.

4 Resource Stack

CloudFormation allows you to quickly create and configure a group of resources by using a stack template. This group of resources is defined as a resource stack. You can manage resources in a stack by creating, updating, or deleting the stack.

Supported operations on resource stack are as follows:

- Create a resource stack
- Get the details of a resource stack
- Delete a resource stack

Create a Resource Stack

In the navigation pane of the ZStack Private Cloud UI, choose **Platform O&M** > **CloudFormation** > **Resource Stack**. On the **Resource Stack** page, click **Create Resource Stack**. Then, the **Create Resource Stack** page is displayed.

To create a resource stack, follow the steps below:

1. Configure the following parameters:

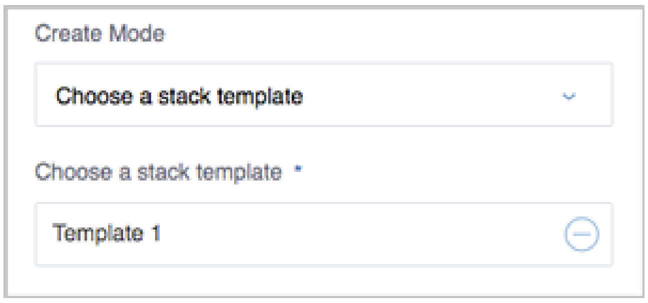
- **Zone**: The current zone is automatically displayed.
- **Name**: Enter a name for the resource stack.
- **Description**: Optional. Enter a description for the resource stack.
- **Timeout**: Specify the timeout during the resource stack creation. If the timeout period expires before the resource stack creation completes, CloudFormation marks the resource stack as failed. The timeout is 60 minutes by default.
- **Rollback on failure**: Specify whether to roll back the resource stack if the creation fails. The checkbox is selected by default.
- **Create Mode**: Select the resource stack creation mode.

  The following three methods for creating a resource stack are available:

  - **Choose a stack template**: Select a custom template or a sample template for resource stack creation.

    As shown in *Figure 4-1: Choose a stack template*.

Figure 4-1: Choose a stack template



Create Mode

Choose a stack template

Choose a stack template *

Template 1

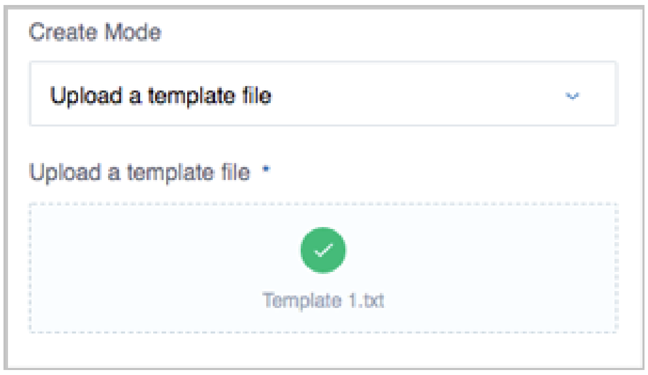
Note:

For more information about how to create a custom template, see the *Stack Template* topic.

- **Upload a template file**: Upload a UTF8-encoded template file for resource stack creation.

  As shown in *Figure 4-2: Upload a template file*.

Figure 4-2: Upload a template file



Create Mode

Upload a template file

Upload a template file *

Template 1.txt

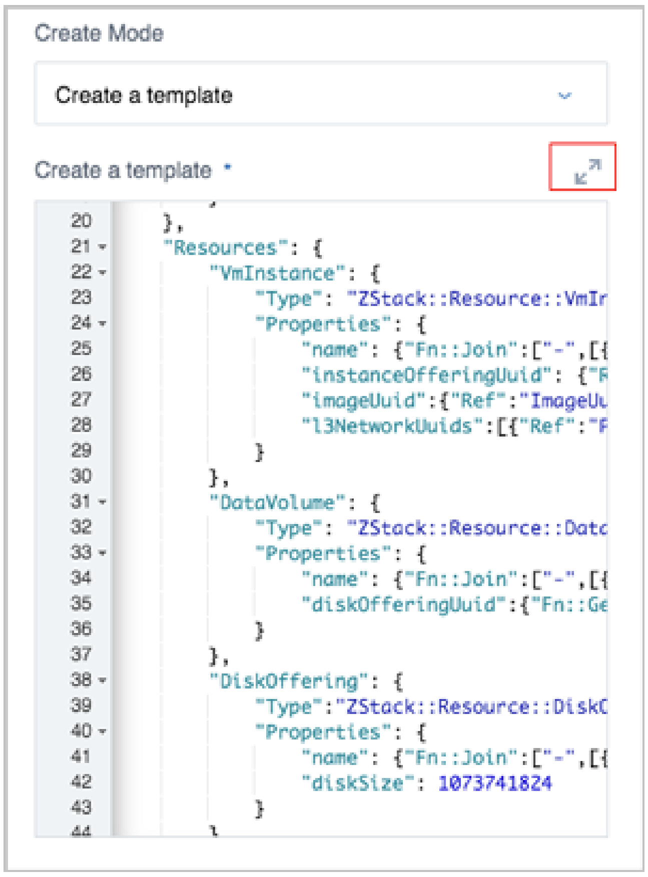
Note:

For more information about the template syntax, see the *Stack Template Syntax* topic.

- **Create a template**: Create a template in a text editor for resource stack creation.

  As shown in *Figure 4-3: Create a template*.

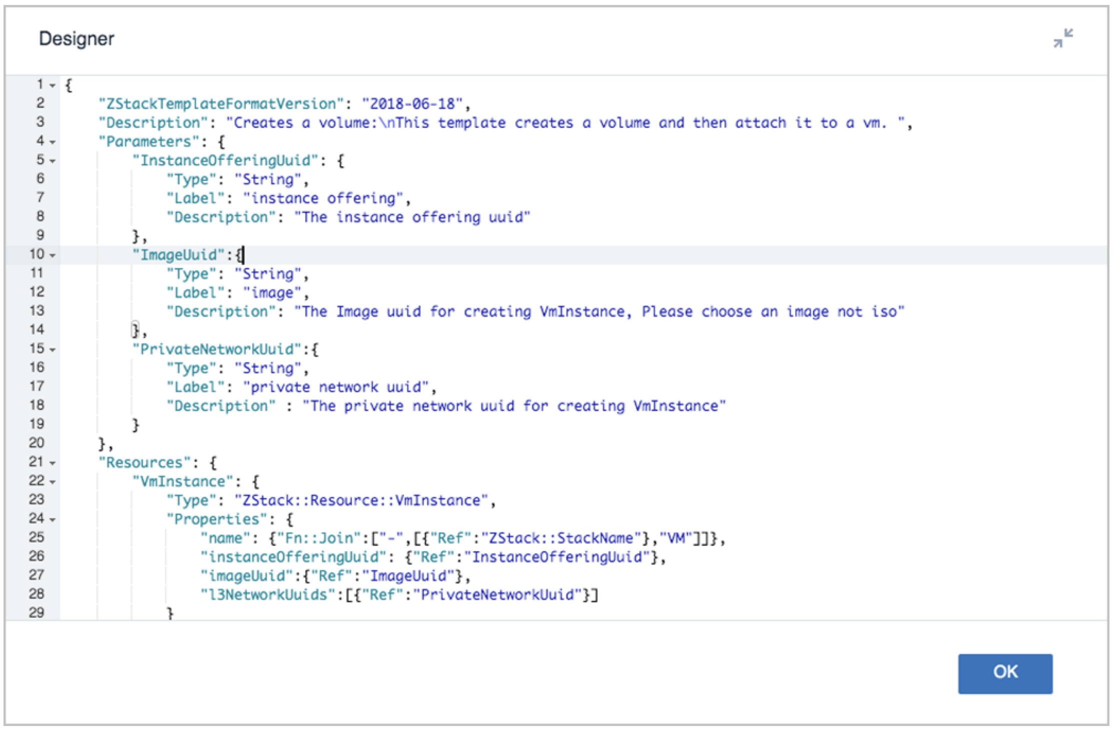
Figure 4-3: Create a template



```
    },
    "Resources": {
        "VmInstance": {
            "Type": "ZStack::Resource::VmIr
            "Properties": {
                "name": {"Fn::Join":["-",[{
                "instanceOfferingUuid": {"R
                "imageUuid":{"Ref":"ImageUu
                "l3NetworkUuids":[{"Ref":"F
            }
        },
        "DataVolume": {
            "Type": "ZStack::Resource::Datc
            "Properties": {
                "name": {"Fn::Join":["-",[{
                "diskOfferingUuid":{"Fn::Ge
            }
        },
        "DiskOffering": {
            "Type":"ZStack::Resource::DiskC
            "Properties": {
                "name": {"Fn::Join":["-",[{
                "diskSize": 1073741824
            }
```

You can zoom in the text editor, as shown in *Figure 4-4: Zoom in the text editor*.

Figure 4-4: Zoom in the text editor



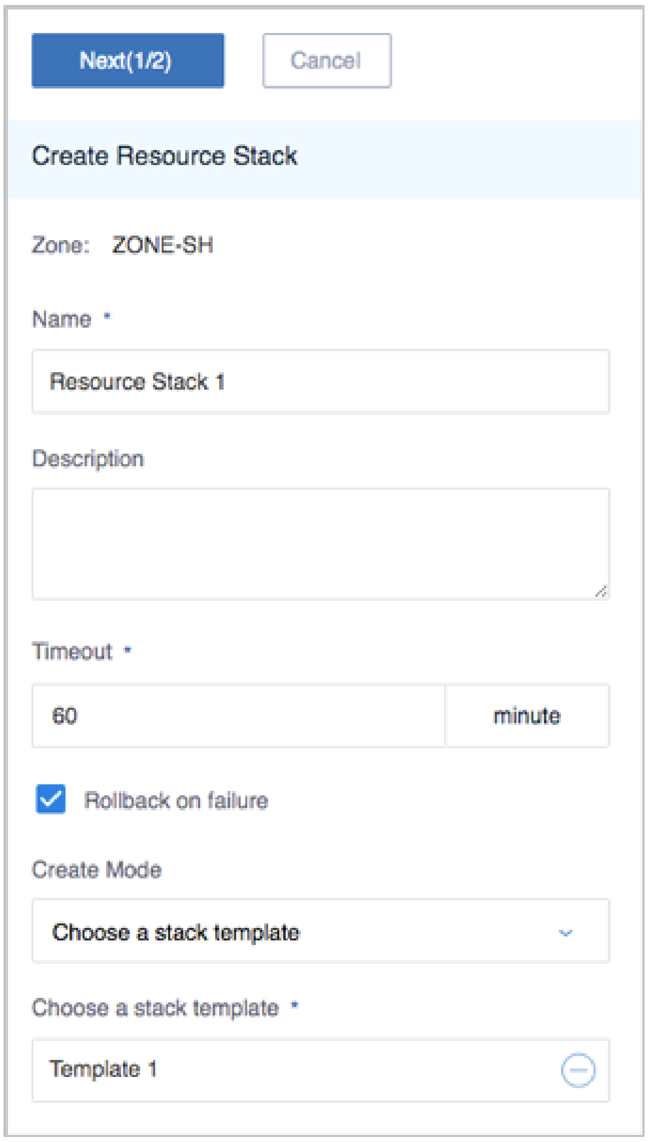
```
{
    "ZStackTemplateFormatVersion": "2018-06-18",
    "Description": "Creates a volume:\nThis template creates a volume and then attach it to a vm. ",
    "Parameters": {
        "InstanceOfferingUuid": {
            "Type": "String",
            "Label": "instance offering",
            "Description": "The instance offering uuid"
        },
        "ImageUuid":{
            "Type": "String",
            "Label": "image",
            "Description": "The Image uuid for creating VmInstance, Please choose an image not iso"
        },
        "PrivateNetworkUuid":{
            "Type": "String",
            "Label": "private network uuid",
            "Description" : "The private network uuid for creating VmInstance"
        }
    },
    "Resources": {
        "VmInstance": {
            "Type": "ZStack::Resource::VmInstance",
            "Properties": {
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName"},"VM"]]},
                "instanceOfferingUuid": {"Ref":"InstanceOfferingUuid"},
                "imageUuid":{"Ref":"ImageUuid"},
                "l3NetworkUuids":[{"Ref":"PrivateNetworkUuid"}]
            }
```

Note:

For more information about the template syntax, see the *Stack Template Syntax* topic.

As shown in *Figure 4-5: Create resource stack 1*. Click **Next**.

Figure 4-5: Create resource stack 1



Create Resource Stack

Zone: ZONE-SH

Name *

Resource Stack 1

Description

Timeout *

60 minute

✓ Rollback on failure

Create Mode

Choose a stack template

Choose a stack template *

Template 1

2. Specify each parameter according to the needed stack resource. Different parameters are specified according to different types of resource stacks.

Note:

- Mechanism:

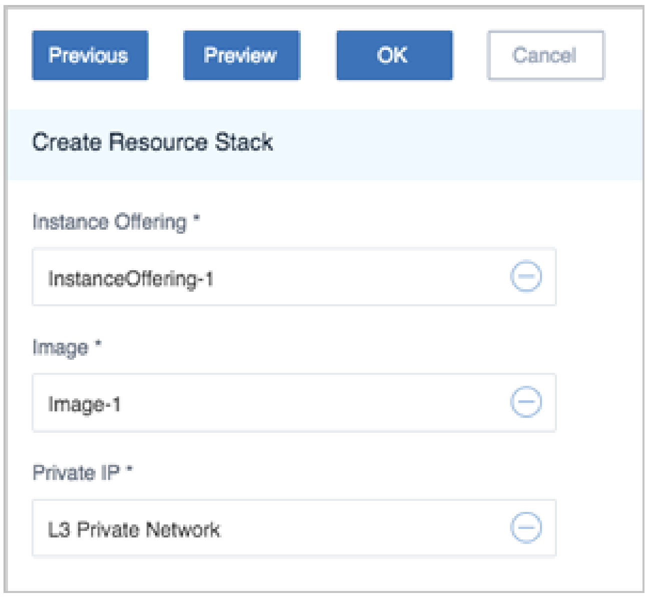
  For the stack template you submitted, if CloudFormation decides what resource the `Resource` filed in `Parameters` is specified as, the UI will provide a drop-down menu for you to select a corresponding resource. Otherwise, the UI will provide an input box for you to enter a field value (a string or a number).

The following is an example of creating a resource stack by using a selected custom template above. CloudFormation will automatically create a VM instance and attach a volume to it. Configure the following parameters:

- **Instance Offering**: Select the instance offering for VM instance creation.
- **Image**: Select the image for VM instance creation.
- **Private IP**: Select the private network for VM instance creation.

As shown in *Figure 4-6: Create resource stack 2*. Click **OK**. Then, the resource stack creation starts.

Figure 4-6: Create resource stack 2



Create Resource Stack

Instance Offering *

InstanceOffering-1

Image *

Image-1

Private IP *

L3 Private Network

Note:

- Before the resource stack creation starts, you can click **Preview** to check the resource list to be created.
- It will take some time to create a resource stack. Please wait for the completion.

Get the Details of a Resource Stack

On the **Resource Stack** page, select a resource stack and expand its details page. You can get the current details of the resource stack, including basic attributes, resource stack content, resource, event, and audit.

- Basic attributes: Displays the current status, name, description, and UUID of the resource stack . The name and description can be modified.

- Resource stack content: Includes the details of the template and the parameters configuration.
    - Template: Displays the details of the template used by the resource stack.
    - Parameters: Displays the details of the parameters specified for resource stack creation.
- Resource: Displays the details of all resources in the resource stack.
- Event: Displays each event in the resource stack lifecycle.
- Audit: Checks related operations about the resource stack.

Delete a Resource Stack

You can delete a resource stack if you no longer need it.

Note:

- When you delete a resource stack, all resources in the stack will be deleted by default.
- In the stack template used by a resource stack, if the `DeletionPolicy` field is set to `Retain`, all resources in the stack will be retained after you delete the resource stack. For more information about `DeletionPolicy`, see the *Resources* topic.

5 Stack Template

You can quickly create a resource stack based on a stack template.

CloudFormation provides two types of stack templates: sample template and custom template. This topic mainly introduces the custom template. For information about the sample template, see the *Sample Template* topic.

Supported operations on stack template are as follows:

- Create a stack template
- Get the details of a stack template
- Enable a stack template
- Disable a stack template
- Generate a resource stack
- Modify a stack template
- Delete a stack template

Create a Stack Template

In the navigation pane of the ZStack Private Cloud UI, choose **Platform O&M** > **CloudFormation** > **Stack Template**. On the **Stack Template** page, click **Create Stack Template**. Then, the **Create Stack Template** page is displayed. Configure the following parameters:

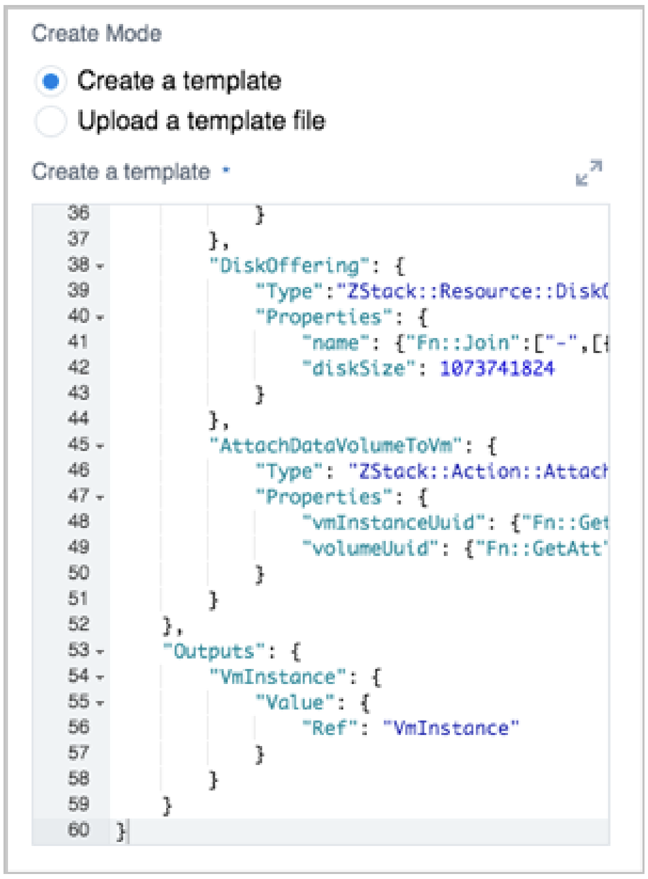
- **Name**: Enter a name for the stack template.
- **Description**: Optional. Enter a description for the stack template.
- **Create Mode**: Select the stack template creation mode.

  The following two methods for creating a stack template are available:

  - **Create a template**: Create a template in a text editor.

    As shown in *Figure 5-1: Create a template*.

Figure 5-1: Create a template



Create Mode

[✓ Create a template]

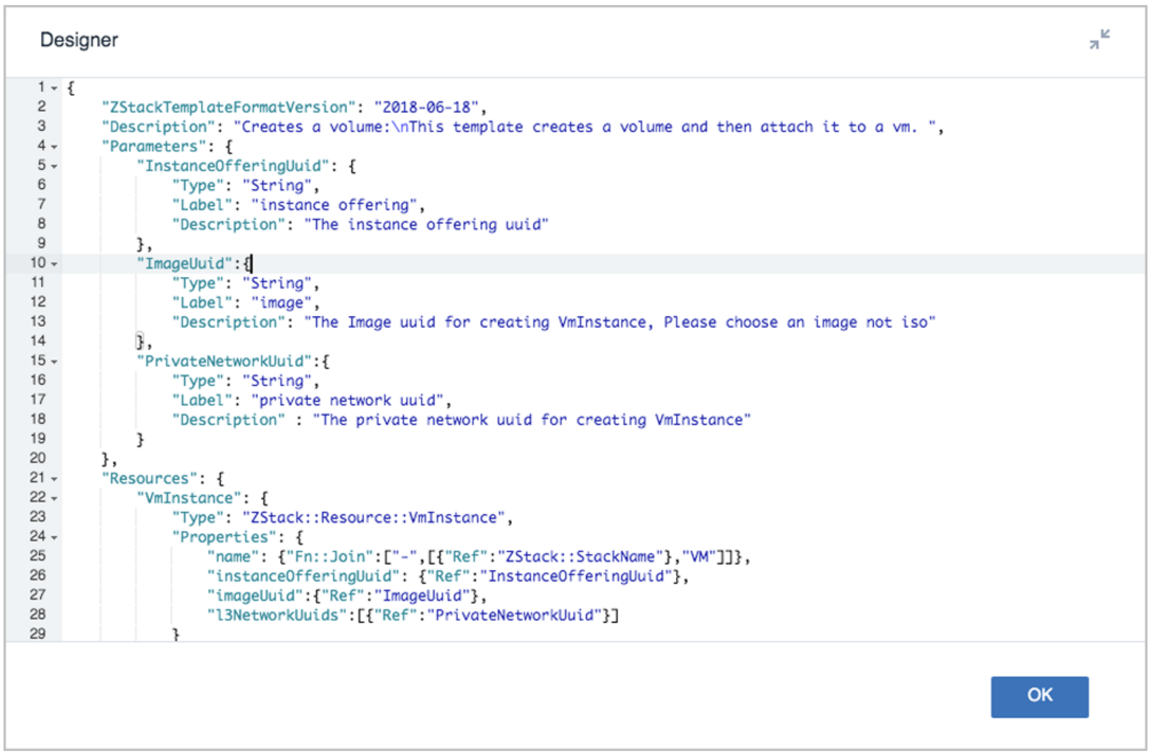
Upload a template file

Create a template *

```
            }
        },
        "DiskOffering": {
            "Type":"ZStack::Resource::Disk(
            "Properties": {
                "name": {"Fn::Join":["-",[{
                "diskSize": 1073741824
            }
        },
        "AttachDataVolumeToVm": {
            "Type": "ZStack::Action::Attach
            "Properties": {
                "vmInstanceUuid": {"Fn::Get
                "volumeUuid": {"Fn::GetAtt'
            }
        }
    },
    "Outputs": {
        "VmInstance": {
            "Value": {
                "Ref": "VmInstance"
            }
        }
    }
}
```

You can zoom in the text editor, as shown in *Figure 5-2: Zoom in the text editor*.

Figure 5-2: Zoom in the text editor



```
{
    "ZStackTemplateFormatVersion": "2018-06-18",
    "Description": "Creates a volume:\nThis template creates a volume and then attach it to a vm. ",
    "Parameters": {
        "InstanceOfferingUuid": {
            "Type": "String",
            "Label": "instance offering",
            "Description": "The instance offering uuid"
        },
        "ImageUuid":{
            "Type": "String",
            "Label": "image",
            "Description": "The Image uuid for creating VmInstance, Please choose an image not iso"
        },
        "PrivateNetworkUuid":{
            "Type": "String",
            "Label": "private network uuid",
            "Description" : "The private network uuid for creating VmInstance"
        }
    },
    "Resources": {
        "VmInstance": {
            "Type": "ZStack::Resource::VmInstance",
            "Properties": {
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName"},"VM"]]},
                "instanceOfferingUuid": {"Ref":"InstanceOfferingUuid"},
                "imageUuid":{"Ref":"ImageUuid"},
                "l3NetworkUuids":[{"Ref":"PrivateNetworkUuid"}]
            }
```

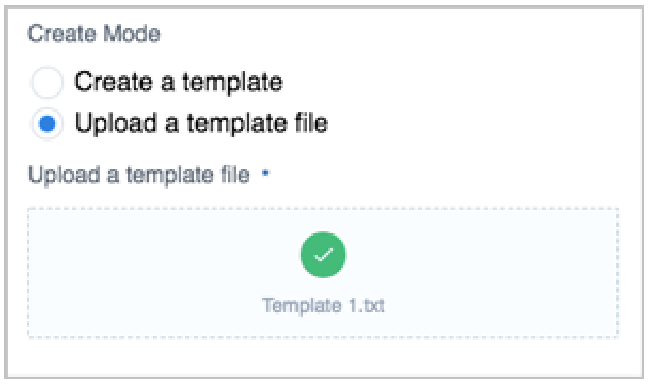
Note:

For more information about the template syntax, see the *Stack Template Syntax* topic.

- **Upload a template file**: Upload a UTF8-encoded template file.

  As shown in *Figure 5-3: Upload a template file*.

Figure 5-3: Upload a template file



Create Mode

Create a template [✓ Upload a template file]

Upload a template file *

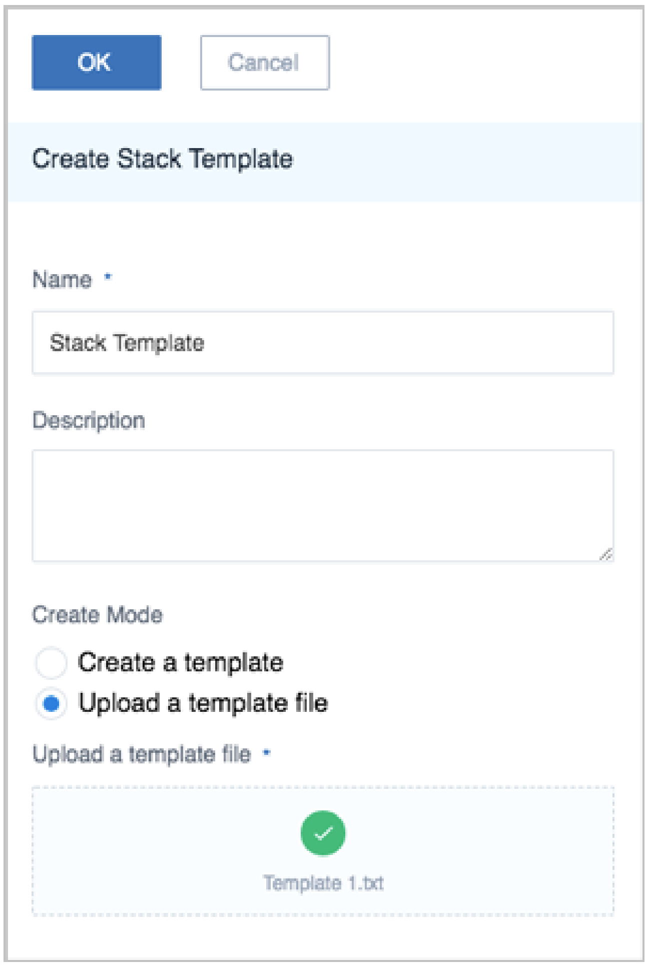
Template 1.txt

Note:

For more information about the template syntax, see the *Stack Template Syntax* topic.

As shown in *Figure 5-4: Create a stack template*. Click **OK**.

Figure 5-4: Create a stack template



Create Stack Template

Name *

Stack Template

Description

Create Mode

Create a template
● Upload a template file

Upload a template file *

Template 1.txt

Get the Details of a Stack Template

On the **Stack Template** page, select a stack template and expand its details page. You can get the current details of the stack template, including basic attributes, stack template content, and audit.

- Basic attributes: Displays the current status, name, description, UUID, and md5sum of the stack template. The name and description can be modified.
- Stack template content: Displays the details of the template.

Note:

For more information about the template syntax, see the *Stack Template Syntax* topic.

- Audit: Checks related operations about the template.

Enable/Disable a Stack Template

- Enable a stack template: If a stack template is disabled, you can enable it as needed.
- Disable a stack template: You can disable a stack template as needed.

Note:

You are not allowed to create a resource stack by using a disabled stack template.

Generate a Resource Stack

On the **Stack Template** page, select a stack template and click **Actions** > **Generate Resource Stack**. Then, the **Generate Resource Stack** page is displayed.

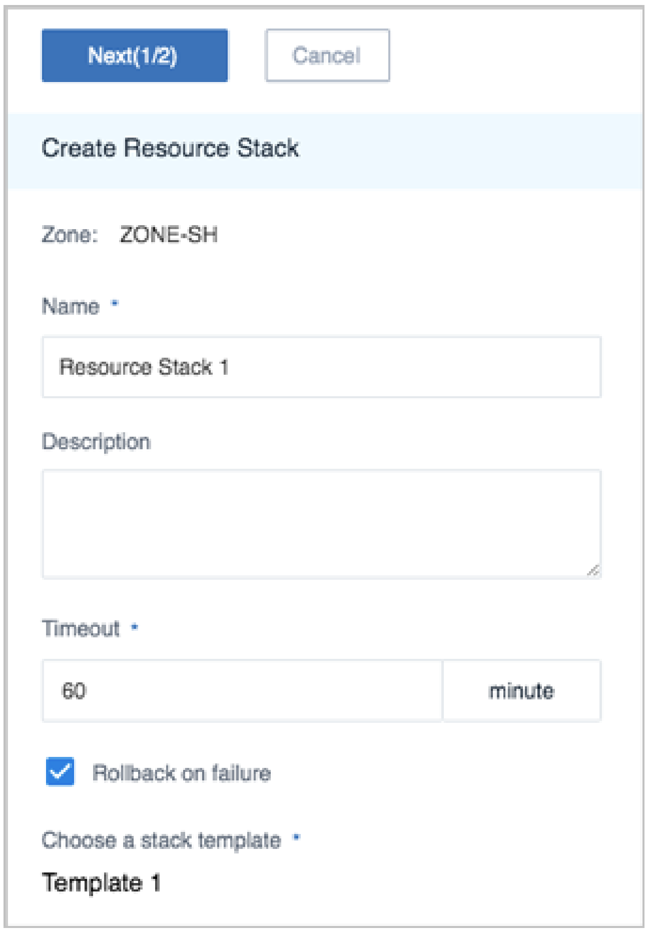
To create a resource stack by using a custom template, follow the steps below:

1. Configure the following parameters:

- **Zone**: The current zone is automatically displayed.
- **Name**: Enter a name for the resource stack.
- **Description**: Optional. Enter a description for the resource stack.
- **Timeout**: Specify the timeout during the resource stack creation. If the timeout period expires before the resource stack creation completes, CloudFormation marks the resource stack as failed. The timeout is 60 minutes by default.
- **Rollback on failure**: Specify whether to roll back the resource stack if the creation fails. The checkbox is selected by default.
- **Choose a stack template**: The selected template is automatically displayed.

As shown in *Figure 5-5: Create resource stack 1*. Click **Next**.

Figure 5-5: Create resource stack 1



Create Resource Stack

Zone: ZONE-SH

Name *

Resource Stack 1

Description

Timeout *

60 minute

✓ Rollback on failure

Choose a stack template *

Template 1

2. Specify each parameter according to the needed stack resource. Different parameters are specified according to different types of resource stacks.

Note:

- Mechanism:

  For the stack template you submitted, if CloudFormation decides what resource the `Resource` filed in `Parameters` is specified as, the UI will provide a drop-down menu for you to select a corresponding resource. Otherwise, the UI will provide an input box for you to enter a field value (a string or a number).

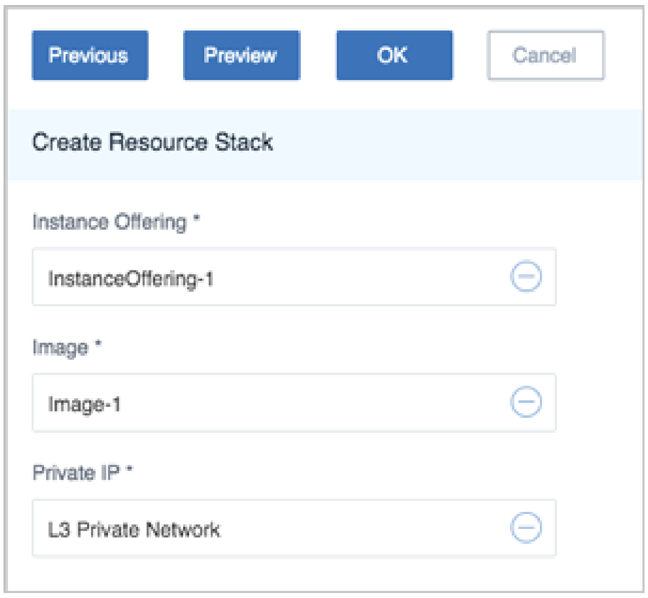
The following is an example of creating a resource stack by using a selected custom template above. CloudFormation will automatically create a VM instance and attach a volume to it. Configure the following parameters:

- **Instance Offering**: Select the instance offering for VM instance creation.

- **Image**: Select the image for VM instance creation.
- **Private IP**: Select the private network for VM instance creation.

As shown in *Figure 5-6: Create resource stack 2*. Click **OK**. Then, the resource stack creation starts.

Figure 5-6: Create resource stack 2



Create Resource Stack

Instance Offering *
InstanceOffering-1

Image *
Image-1

Private IP *
L3 Private Network

Note:

- Before the resource stack creation starts, you can click **Preview** to check the resource list to be created.
- It will take some time to create a resource stack. Please wait for the completion.

Modify a Stack Template

You can modify a stack template in a text editor.

Delete a Stack Template

You can delete a stack template if you no longer need it.

Constraints

- The maximum size of a template is 4 MB.
- If you use the CLI or API to create a stack, you can upload a template with a maximum size of 64 KB.

6 Sample Template

The Cloud provides some commonly used sample templates for you to quickly create resource stacks in CloudFormation.

Supported operations on sample template are as follows:

- Get the details of a sample template
- Generate a resource stack by using a sample template

Get the Details of a Sample Template

In the navigation pane of the ZStack Private Cloud UI, choose **Platform O&M** > **CloudFormation** > **Sample Template**. On the **Sample Template** page, select a sample template and expand its details page. You can get the details of the sample template, including basic attributes, stack template content, and audit.

- Basic attributes: Displays the current status, name, description, UUID, and md5sum of the sample template.

  Note:
  The sample template remains enabled and cannot be modified.

- Stack template content: Displays the details of the template.

  Note:
  For more information about the template syntax, see the *Stack Template Syntax* topic.

- Audit: Checks related operations about the template.

Generate a Resource Stack by Using a Sample Template

On the **Sample Template** page, select a sample template and click **Actions** > **Generate Resource Stack**. Then, the **Generate Resource Stack** page is displayed.

To create a resource stack by using a sample template, follow the steps below:

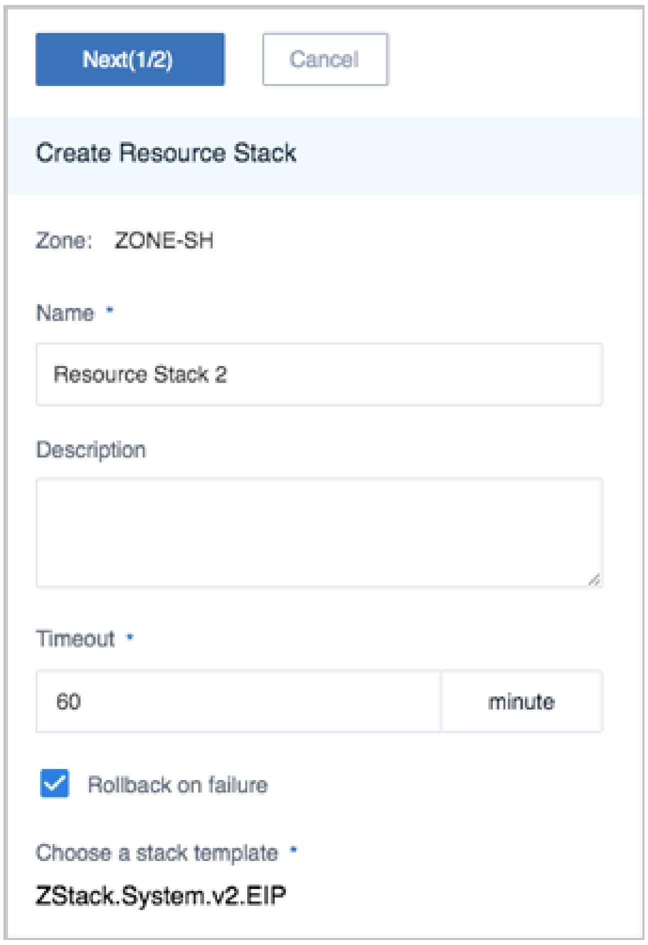
1. Configure the following parameters:

   - **Zone**: The current zone is automatically displayed.
   - **Name**: Enter a name for the resource stack.
   - **Description**: Optional. Enter a description for the resource stack.

- **Timeout**: Specify the timeout during the resource stack creation. If the timeout period expires before the resource stack creation completes, CloudFormation marks the resource stack as failed. The timeout is 60 minutes by default.
- **Rollback on failure**: Specify whether to roll back the resource stack if the creation fails. The checkbox is selected by default.
- **Choose a stack template**: The selected template is automatically displayed.

As shown in *Figure 6-1: Create resource stack 1*. Click **Next**.

Figure 6-1: Create resource stack 1



Create Resource Stack

Zone: ZONE-SH

Name *

Resource Stack 2

Description

Timeout *

60 minute

✓ Rollback on failure

Choose a stack template *

ZStack.System.v2.EIP

2. Specify each parameter according to the needed stack resource. Different parameters are specified according to different types of resource stacks.

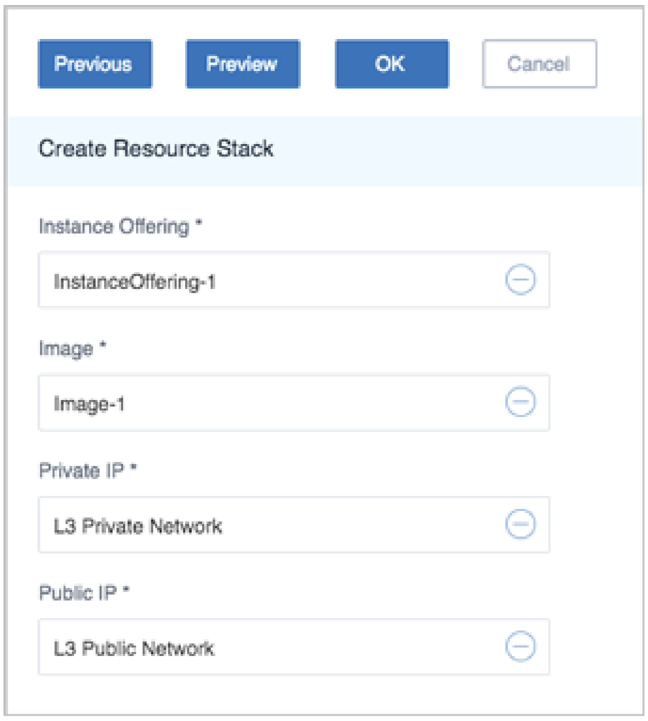
   The following is an example of creating a resource stack by using the selected sample template (**ZStack.System.v1.EIP**) above. CloudFormation will automatically create an EIP and attach it to a VM instance. Configure the following parameters:

   - **Instance Offering**: Select the instance offering for VM instance creation.

- **Image**: Select the image for VM instance creation.
- **Private IP**: Select the private network for VM instance creation.
- **Public IP**: Select the public network for VIP provision. The EIP service can be provided by a VIP.

As shown in *Figure 6-2: Create resource stack 2*. Click **OK**. Then, the resource stack creation starts.

Figure 6-2: Create resource stack 2



Create Resource Stack

Instance Offering *
InstanceOffering-1

Image *
Image-1

Private IP *
L3 Private Network

Public IP *
L3 Public Network

Note:

- Before the resource stack creation starts, you can click **Preview** to check the resource list to be created.
- It will take some time to create a resource stack. Please wait for the completion.

7 Designer

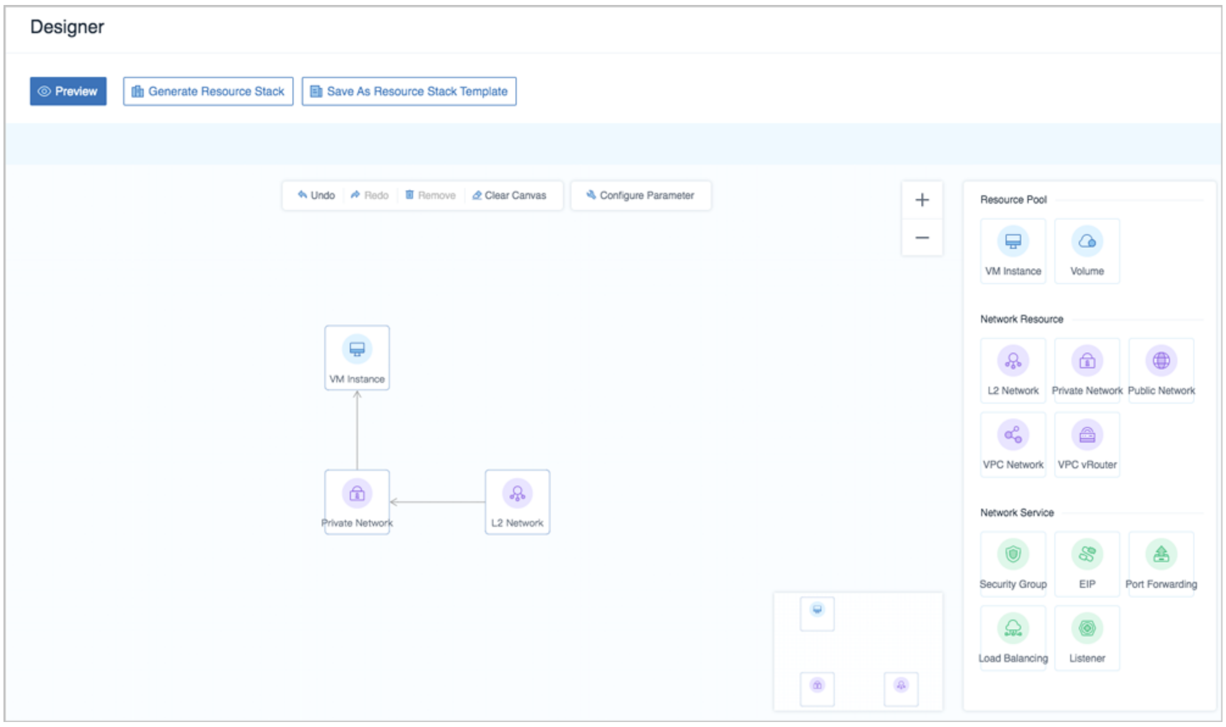
With CloudFormation Designer, you can view the graphic representations of resources in a stack template, and author and edit the stack template in a more visual and simple way.

Designer Interface Overview

In the navigation pane of the ZStack Private Cloud UI, choose **Platform O&M** > **CloudFormation** > **Designer**. Then, the **Designer** interface is displayed.

As shown in *Figure 7-1: Designer*.

Figure 7-1: Designer



The Designer panes and its main components are as follows:

- Toolbar at the top

  The toolbar provides quick access to commands for common actions, such as previewing a template, generating a resource stack, and saving the diagram as a stack template.

- Resource pool pane on the right

  The resource pool pane lists all of the template resources that you can add to your template in Designer. You can add resources by dragging them from the resource pool pane to the canvas. The supported resources are listed as follows:

  - Resource pool: VM instance, Volume
  - Network resource: L2 Network, Private Network, Public Network, VPC Network, and VPC vRouter
  - Network service: Security Group, EIP, Port Forwarding, Load Balancing, and Listener
- Canvas pane in the middle

  The canvas pane displays the template resources as a diagram.

  - You can add resources, create relationships between resources, and arrange their layout.
  - You can undo or redo changes, remove resources, and clear the canvas. The changes that you make in the canvas automatically modify the template.
  - You can specify the details of the template such as resource properties or template parameters with the **Configure Parameter** button.
  - You can drag the thumbnail at bottom right to adjust the canvas pane to fit your template's diagram.
  - You can zoom in or zoom out the canvas by clicking the **+** or **-** button.

8 Typical Scenario-based Practice

8.1 Sample Template-based Practice

Context

This topic introduces how to use the sample template **ZStack.System.v1.VPC** to quickly deploy the VPC network.

Procedure

1. Prepare a stack template.

   In the navigation pane of the ZStack Private Cloud UI, choose **Platform O&M** > **CloudFormation** > **Sample Template**. On the **Sample Template** page, select the sample template **ZStack.System.v1.VPC** and expand its details page. You can get the details of the sample template as below.

```
{
    "ZStackTemplateFormatVersion": "2018-06-18",
    "Description": "Creates VPC network. This template creates a VPC
 network. Make sure that the public network and management network
are working as expected. Note that the VXLAN VTEP CIDR is required
.",
    "Parameters": {
        "VrouterImageUrl": {
            "Type": "String",
            "Label":"vRouter image",
            "Description":"vRouter image URL",
            "DefaultValue": "http://cdn.zstack.io/product_downloads/
vrouter/2.3/zstack-vrouter-2.3.2.qcow2"
        },
        "VmImageUrl": {
            "Type": "String",
            "Label": "VM image url",
            "Description":"VM image url",
            "DefaultValue": "http://cdn.zstack.io/zstack_repo/latest
/zstack-image-1.4.qcow2"
        },
        "BackupStorage":{
            "Type": "CommaDelimitedList",
            "Label": "BackupStorage UUID",
            "Description":"BackStorage UUID"
        },
        "ManagementNetworkUuid":{
            "Type": "String",
            "Label": "Management network",
            "Description":"You can use public network as management
network"
        },
        "PublicNetworkUuid":{
            "Type": "String",
            "Label": "Public network",
            "Description":"Public network UUID"
        },
        "ZoneUuid":{
            "Type": "String",
            "Label": "Zone",
            "Description":"Zone UUID"
        },
        "ClusterUuid":{
            "Type": "String",
            "Label": "Cluster",
            "Description":"Cluster UUID"
        },
        "Cidr":{
            "Type": "String",
            "Description":"VTEP CIDR. Use the correct CIDR",
            "DefaultValue":"{10.0.0.0/8}"
        },
        "Vni":{
            "Type": "Number",
            "DefaultValue":222
        },
        "StartVni":{
            "Type": "Number",
            "DefaultValue":100
        },
        "EndVni":{
            "Type": "Number",
            "DefaultValue":300
        },
        "StartIp":{
            "Type": "String",
            "DefaultValue":"192.168.20.2"
        },
        "EndIp":{
            "Type": "String",
            "DefaultValue":"192.168.20.200"
        },
        "Netmask":{
            "Type": "String",
            "DefaultValue":"255.255.255.0"
        },
        "Gateway":{
            "Type": "String",
            "DefaultValue":"192.168.20.1"
        }
    },
    "Resources": {
        "VrouterImage": {
            "Type": "ZStack::Resource::Image",
            "Properties": {
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, {"Ref":"ZStack::StackUuid"},{"Ref":"ZStack::AccountUuid"},{"Ref
":"ZStack::AccountName"},"Vrouter-Image"]]},
                "url": {"Ref":"VrouterImageUrl"},
                "system": true,
                "format": "qcow2",
                "backupStorageUuids":{"Ref":"BackupStorage"}
            }
        },
        "VMImage": {
            "Type": "ZStack::Resource::Image",
            "Properties": {
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "VmImage"]]},
                "url": {"Ref":"VmImageUrl"},
                "format": "qcow2",
                "backupStorageUuids":{"Ref":"BackupStorage"}
            }
        },
        "VirtualRouterOffering":{
            "Type":"ZStack::Resource::VirtualRouterOffering",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "Vrouter-Offering"]]},
                "zoneUuid":{"Ref":"ZoneUuid"},
                "managementNetworkUuid":{"Ref":"ManagementNetworkUui
d"},
                "publicNetworkUuid":{"Ref":"PublicNetworkUuid"},
                "imageUuid":{"Fn::GetAtt":["VrouterImage", "uuid"]},
                "cpuNum":2,
                "memorySize":2147483648
            }
        },
        "VpcVRouter":{
            "Type":"ZStack::Resource::VpcVRouter",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "VPC-Router"]]},
                "virtualRouterOfferingUuid":{"Fn::GetAtt":["
VirtualRouterOffering","uuid"]}
            }
        },
        "L2VxlanNetworkPool":{
            "Type":"ZStack::Resource::L2VxlanNetworkPool",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "L2VxlanNetworkPool"]]},
                "zoneUuid":{"Ref":"ZoneUuid"}
            }
        },
        "VniRange":{
            "Type":"ZStack::Resource::VniRange",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "VniRange"]]},
                "startVni":{"Ref":"StartVni"},
                "endVni":{"Ref":"EndVni"},
                "l2NetworkUuid":{"Fn::GetAtt":["L2VxlanNetworkPool
","uuid"]}
            }
        },
        "L2VxlanNetwork":{
            "Type":"ZStack::Resource::L2VxlanNetwork",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "L2VxlanNetwork"]]},
                "poolUuid":{"Fn::GetAtt":["L2VxlanNetworkPool","uuid
"]},
                "zoneUuid":{"Ref":"ZoneUuid"},
                "vni":{"Ref":"Vni"}
            }
        },
        "VpcL3Network":{
            "Type":"ZStack::Resource::L3Network",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "VPC-Network"]]},
                "l2NetworkUuid":{"Fn::GetAtt":["L2VxlanNetwork","
uuid"]},
                "category":"Private",
                "type":"L3VpcNetwork",
                "systemTags":["networkservices::VRouter"]
            }
        },
        "InstanceOffering":{
            "Type":"ZStack::Resource::InstanceOffering",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "1cpu","4G"]]},
                "cpuNum": 1,
                "memorySize" : 4294967296
            }
        },

        "AttachL3ToVm":{
            "Type":"ZStack::Action::AttachL3NetworkToVm",
            "Properties":{
                "vmInstanceUuid": {"Fn::GetAtt":["VpcVRouter","uuid
"]},
                "l3NetworkUuid":{"Fn::GetAtt":["VpcL3Network","uuid
"]}
            },
            "DependsOn":[{"Ref":"AddIpRange"}]
        },
        "AddIpRange" :{
            "Type":"ZStack::Action::AddIpRange",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "iprange"]]},
                "l3NetworkUuid":{"Fn::GetAtt":["VpcL3Network","uuid
"]},
                "startIp":{"Ref":"StartIp"},
                "endIp":{"Ref":"EndIp"},
                "netmask":{"Ref":"Netmask"},
                "gateway":{"Ref":"Gateway"}
            }
        },
        "AttachL2NetworkToCluster":{
            "Type":"ZStack::Action::AttachL2NetworkToCluster",
            "Properties":{
                "l2NetworkUuid":{"Fn::GetAtt":["L2VxlanNetworkPool
","uuid"]},
                "clusterUuid":{"Ref":"ClusterUuid"},
                "systemTags":[{"Fn::Join":["::",["l2NetworkUuid
",{"Fn::GetAtt":["L2VxlanNetwork","uuid"]},"clusterUuid",{"Ref":"
ClusterUuid"},"cidr",{"Ref":"Cidr"}]]}]
            }
        },
        "TestVm":{
            "Type":"ZStack::Resource::VmInstance",
            "Properties":{
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"}, "TestVm"]]},
                "instanceOfferingUuid": {"Fn::GetAtt":["InstanceOf
fering","uuid"]},
                "l3NetworkUuids": [{"Fn::GetAtt":["VpcL3Network","
uuid"]}],
                "imageUuid": {"Fn::GetAtt":["VMImage", "uuid"]}
            },
            "DependsOn":[{"Ref":"AttachL3ToVm"}]
        }
    },
    "Outputs": {
        "vpc": {
            "Value": {
                "Ref": "VpcL3Network"
            }
        }
    }
}
```

The template above includes the following five top-level fields:

- `"ZStackTemplateFormatVersion": "2018-06-18"`

  It declares the version of the template.

- `"Description": "Creates VPC network. This template creates a VPC network. Make sure that the public network and management network are working as expected. Note that the VXLAN VTEP CIDR is required."`

  It declares the description of the template.

- `"Parameters": { }`

  It declares a list of parameters in the template.

  In this example, it declares the following parameters:

  - `VrouterImageUrl`
  - `VmImageUrl`
  - `BackupStorage`
  - `ManagementNetworkUuid`
  - `PublicNetworkUuid`
  - `ZoneUuid`
  - `ClusterUuid`
  - `Cidr`
  - `Vni`
  - `StartVni`
  - `EndVni`
  - `StartIp`
  - `EndIp`

        - `Netmask`
        - `Gateway`
    - `"Resources": { }`

      It declares the resources to be created by the template.

      In this example, it declares the following resources to be created:

        - Add a vRouter image.
        - Add a VM image.
        - Create a vRouter offering.
        - Create a VPC vRouter.
        - Create a VXLAN pool.
        - Create a L2 VXLAN network.
        - Create a VPC network.
        - Create an instance offering.
        - Attach the VPC network to the VM instance.
        - Specify the IP range of the VPC network.
        - Attach the L2 VXLAN network to the cluster.
        - Create a VM instance.

      The properties of the resources declared in `"Resources": { }` can reference the parameters declared in `"Parameters": { }`.
    - `"Outputs": { }`

      After the declared resources complete their creations, it provides useful information such as resource properties.

    For more information about the template syntax, see the *Stack Template Syntax* topic.

2. Create a resource stack by using a sample template.

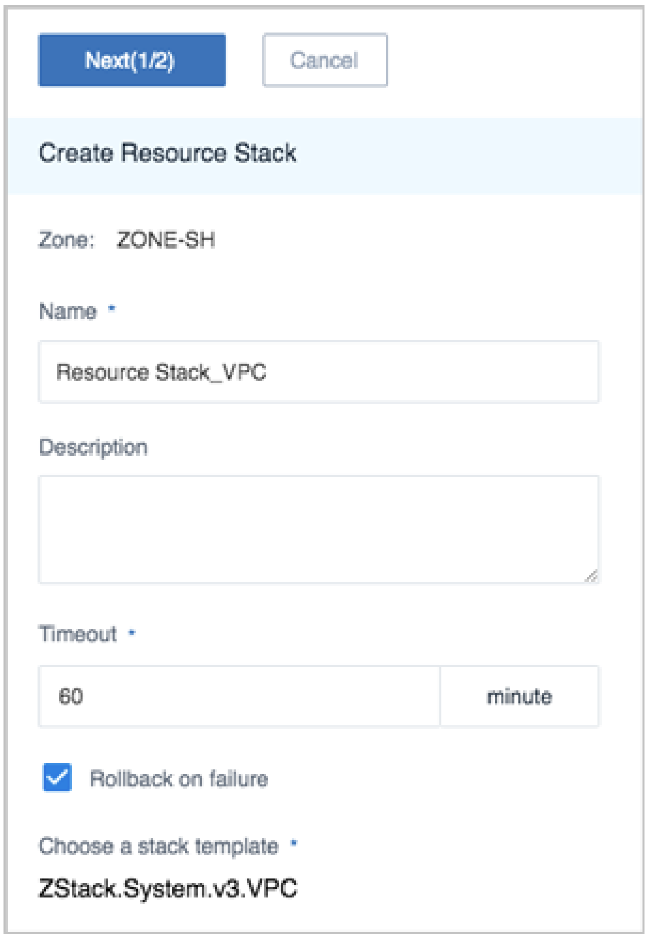
   On the **Sample Template** page, select the sample template **ZStack.System.v1.VPC** and click **Actions** > **Generate Resource Stack**. Then, the **Generate Resource Stack** page is displayed.

   1. Configure the following parameters:
      - **Zone**: The current zone is automatically displayed.
      - **Name**: Enter a name for the resource stack.

- **Description**: Optional. Enter a description for the resource stack.
- **Timeout**: Specify the timeout during the resource stack creation. If the timeout period expires before the resource stack creation completes, CloudFormation marks the resource stack as failed. The timeout is 60 minutes by default.
- **Rollback on failure**: Specify whether to roll back the resource stack if the creation fails. The checkbox is selected by default.
- **Choose a stack template**: The selected template is automatically displayed.

As shown in *Figure 8-1: Create resource stack 1*. Click **Next**.

Figure 8-1: Create resource stack 1



Create Resource Stack

Zone: ZONE-SH

Name *

Resource Stack_VPC

Description

Timeout *

60 minute

✓ Rollback on failure

Choose a stack template *

ZStack.System.v3.VPC

2. Specify each parameter according to the needed stack resource. Different parameters are specified according to different types of resource stacks.

   - **vRouter Image URL**: Add a vRouter image for VPC vRouter creation
   - **VM Image URL**: Add an image for VM instance creation
   - **Backup Storage**: Select a backup storage.

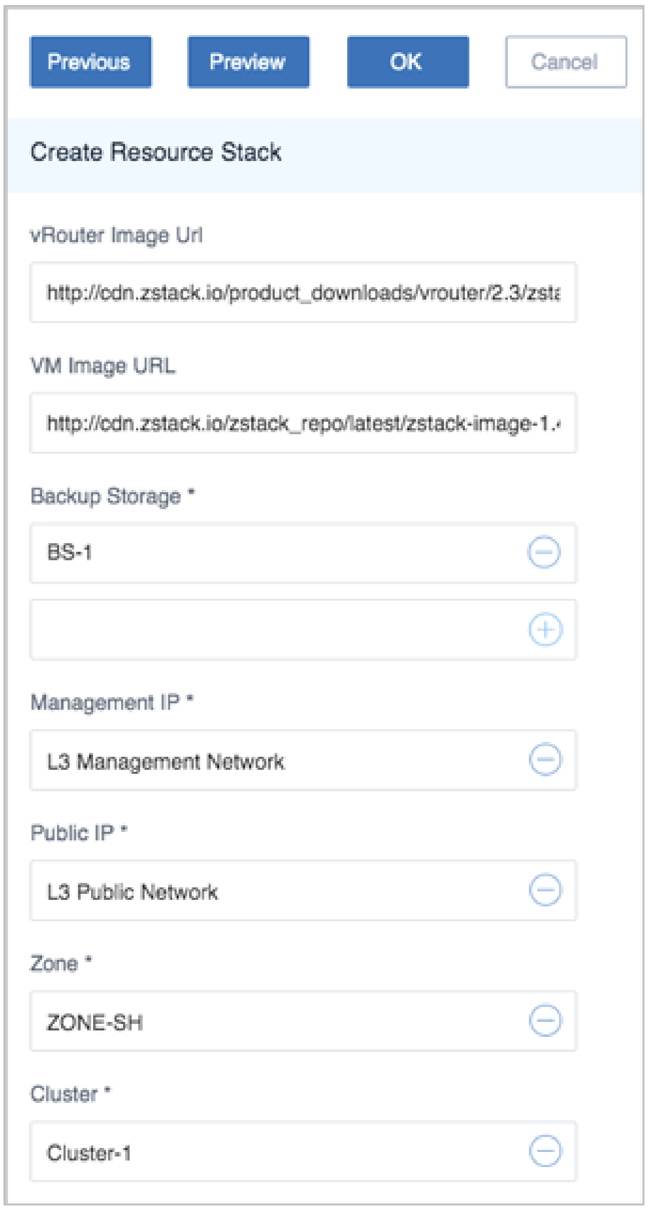
- **Management IP**: Select an existing management network.

> **Note:**
> We recommend that you deploy a separate management network that is isolated from public network for better security and stability.

- **Public IP**: Select an existing public network.
- **Zone**: The current zone is automatically displayed.
- **Cluster**: Optional. You can select the cluster loaded by the VXLAN pool.
- **VTEP CIDR**: Set the CIDR corresponding to VTEP.
- **Vni**: Optional. You can select a specified Vni from the VXLAN pool. If this field is blank, the system will automatically allocate a Vni.
- **Start Vni**: Set the start Vni in the VXLAN pool.
- **End Vni**: Set the end Vni in the VXLAN pool.
- **Start IP**: Set the start IP address of the VPC network.
- **End IP**: Set the end IP address of the VPC network.
- **Netmask**: Set the netmask of the VPC network.
- **Gateway**: Set the gateway of the VPC network.

As shown in *Figure 8-2: Create resource stack 2*. Click **OK**. Then, the resource stack creation starts.

Figure 8-2: Create resource stack 2



Create Resource Stack

vRouter Image Url: http://cdn.zstack.io/product_downloads/vrouter/2.3/zsta

VM Image URL: http://cdn.zstack.io/zstack_repo/latest/zstack-image-1.

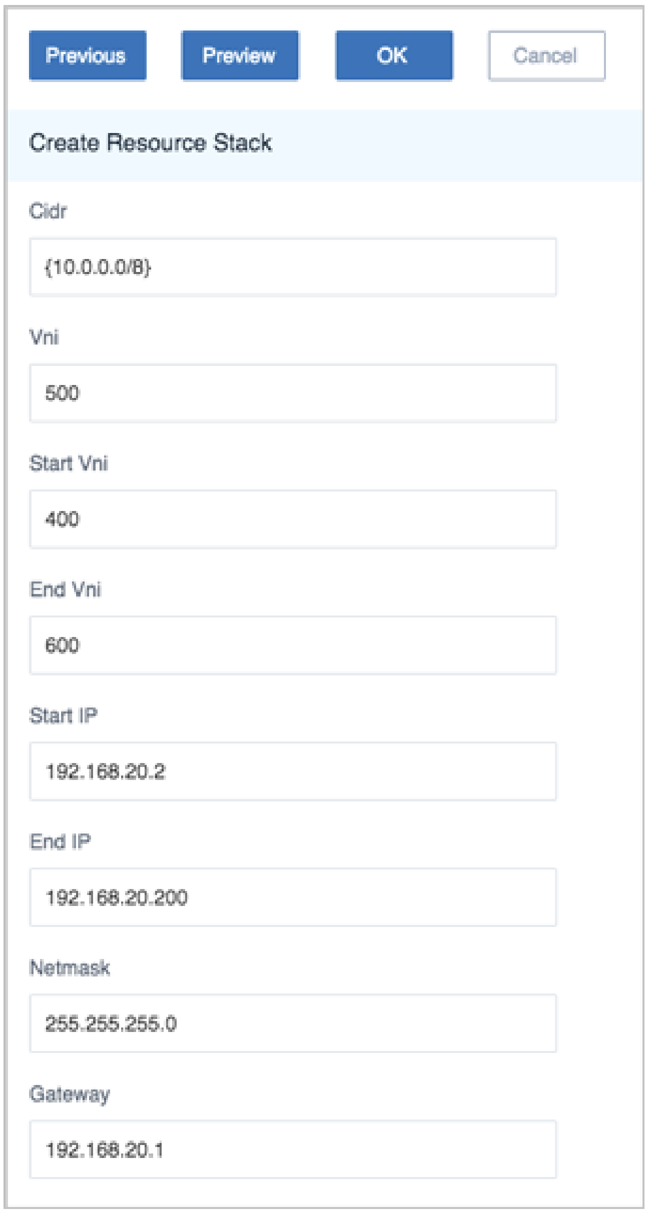
Backup Storage *: BS-1

Management IP *: L3 Management Network

Public IP *: L3 Public Network

Zone *: ZONE-SH

Cluster *: Cluster-1



Create Resource Stack

Cidr	{10.0.0.0/8}
Vni	500
Start Vni	400
End Vni	600
Start IP	192.168.20.2
End IP	192.168.20.200
Netmask	255.255.255.0
Gateway	192.168.20.1

Note:

- Before the resource stack creation starts, you can click **Preview** to check the resource list to be created.
- It will take some time to create a resource stack. Please wait for the completion.

3. Manage the resource stack.

After a resource stack is successfully created, you can click the stack name on the **Resource Stack** page to view the stack status and details.

- Basic attributes: Displays the current status, name, description, and UUID of the resource stack. The name and description can be modified.

- Resource stack content: Includes the details of the template and the parameters configuration.
  - Template: Displays the details of the template used by the resource stack.
  - Parameters: Displays the details of the parameters specified for resource stack creation.
- Resource: Displays the details of all resources in the resource stack.
- Event: Displays each event in the resource stack lifecycle.
- Audit: Checks related operations about the resource stack.

You can delete the resource stack if you no longer need it.

8.2 Designer-based Practice

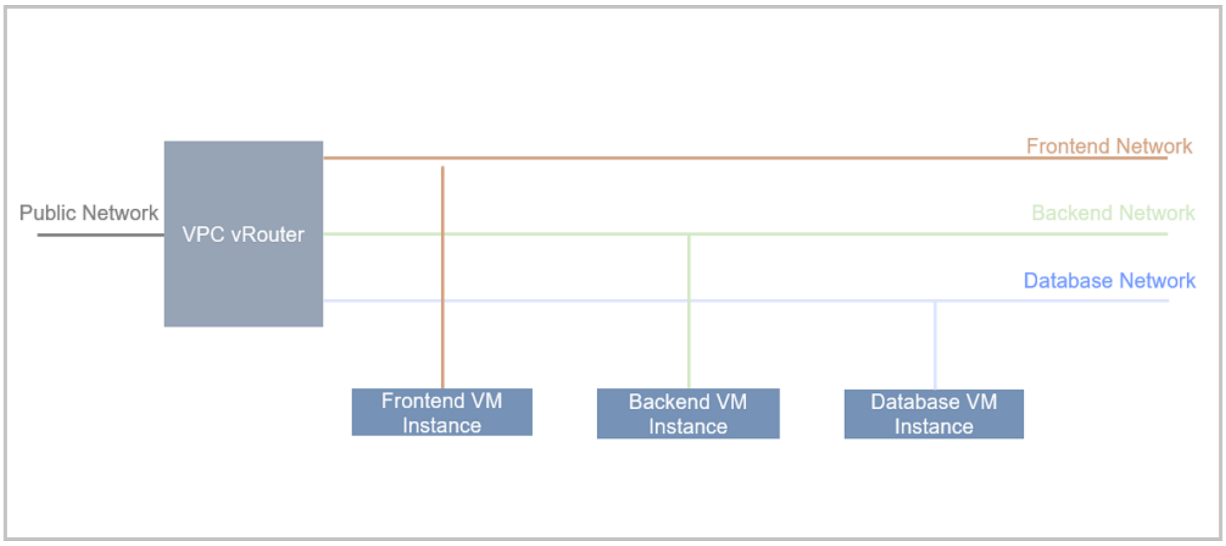
Prerequisites

You can use the designer of CloudFormation to create multiple networks and deploy different services in the networks.

This topic describes how to use the designer to create three virtual private cloud (VPC) networks and separately deploy frontend services, backend services, and database services in the networks . This separation ensures network security for your services.

The *Figure 8-3: Multi-layer Service Deployment Diagram* figure shows the multi-layer deployment of services.

Figure 8-3: Multi-layer Service Deployment Diagram



Context

Deployment process:

1. Drag resources from the Resource Pool pane and drop the resources on the canvas.
2. Edit the properties of the resources.
3. Drag connections between resources to establish relationships.
4. Generate a resource stack.
5. Check whether the three VM instances are connected.

The following tables show the information of the VPC vRouter and VM instances that are used in this topic. You can configure the devices based on your business requirements.

1. VPC vRouter

Table 8-1: VPC vRouter Configuration

Name	L2 Network	VPC Network	IP Range
VPC vRouter	L2Network-1	Frontend Network	192.168.0.0/24
		Backend Network	192.168.100.0/24
		Database Network	192.168.200.0/24

2. VM instance

Table 8-2: VM Instance Configuration

Name	Network	IP Address
Frontend VM Instance	Frontend Network	192.168.0.100
Backend VM Instance	Backend Network	192.168.100.100
Database VM Instance	Database Network	192.168.200.100

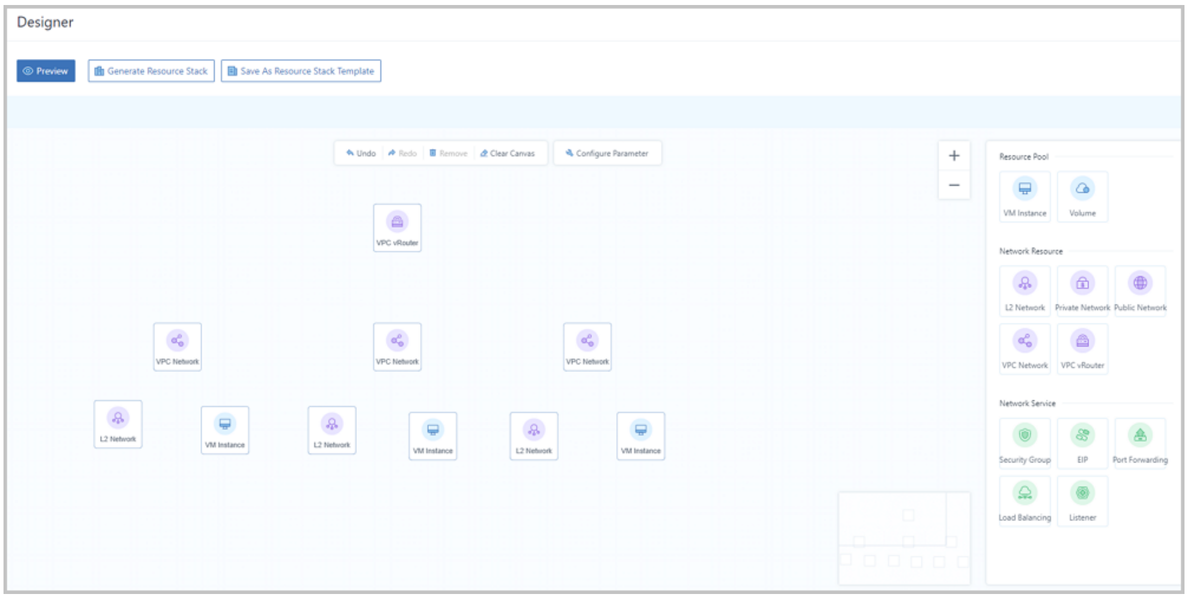
Procedure

1. Drag resources from the Resource Pool pane and drop the resources on the canvas.

   In the navigation pane of the ZStackPrivate Cloud UI, choose **Platform O&M** > **CloudFormation** > **Designer**. On the **Designer** page, drag an L2 network, three VPC networks, a VPC vRouter, and three VM instances from the Resource Pool pane and drop the resources on the canvas.

   The *Figure 8-4: Resources on the Canvas* figure shows the resources on the canvas.

Figure 8-4: Resources on the Canvas

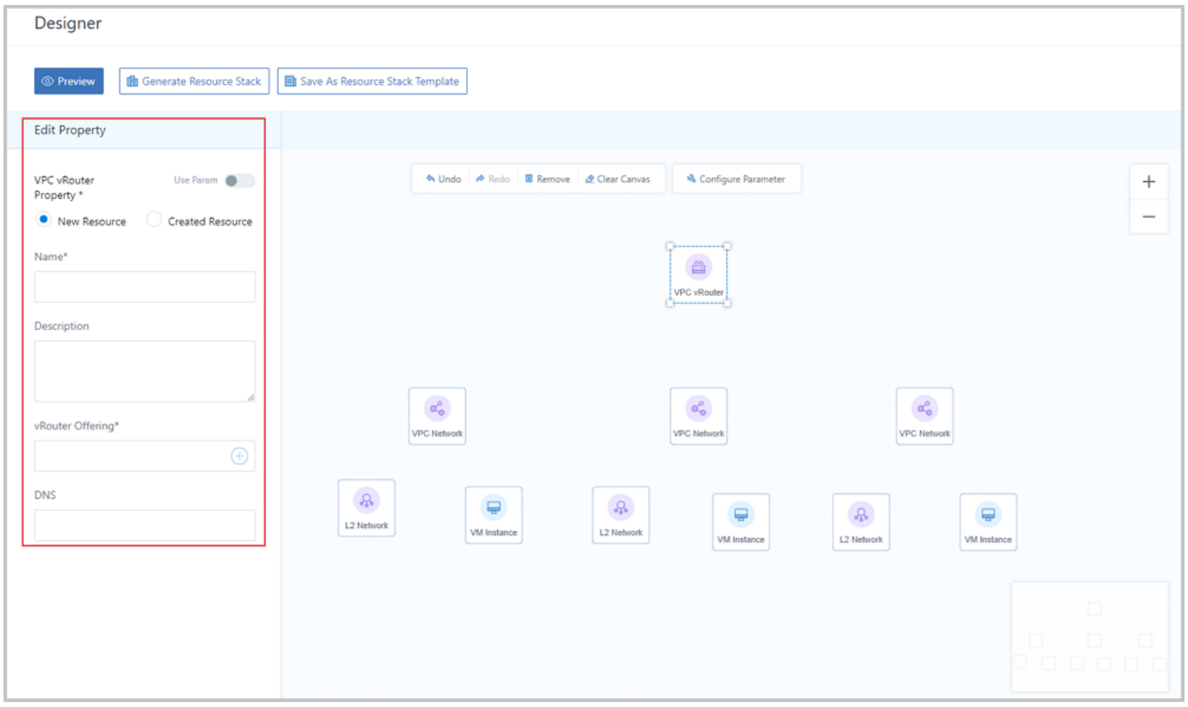


2. Edit the properties of the resources.

   Click the icon of a resource. In the **Edit Property** pane, edit the properties of the resource based on the information listed in the preceding *tables*.

   The *Figure 8-5: Edit Property* figure shows the properties of a VPC vRouter.

Figure 8-5: Edit Property



3. Drag connections between resources to establish relationships.

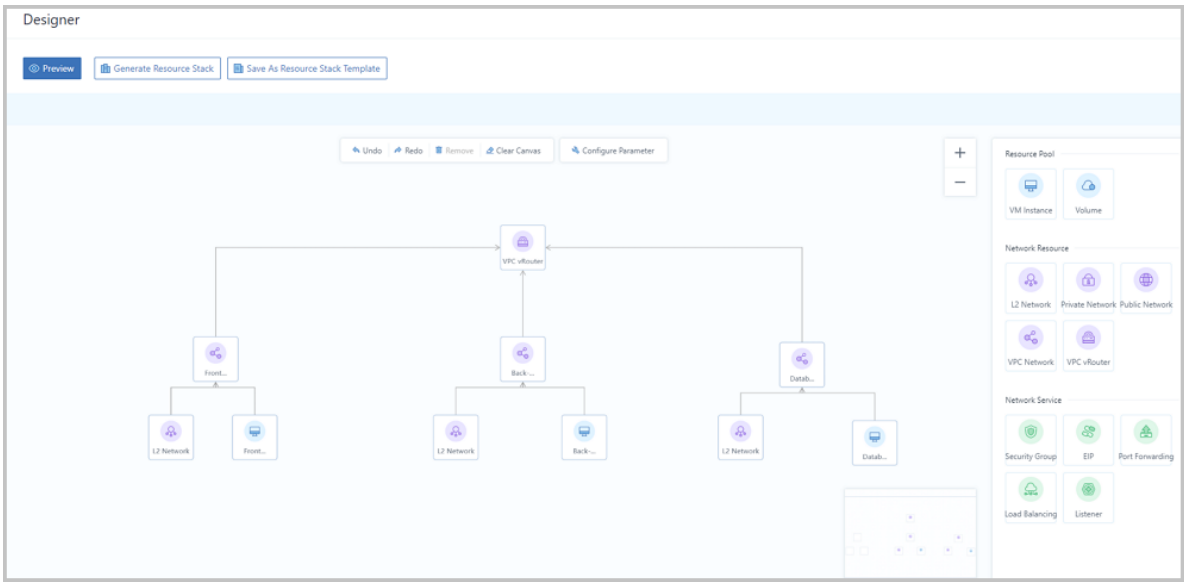
   Click the icon of a resource. Four dots appear in the border lines of the resource icon. Select a dot and drag a line from the dot to establish a relationship with another resource.

   Note:

   After you drag a line from a dot to establish a relationship with another resource, four dots appear in the border lines of the resource.

   The *Figure 8-6: Drag Connections Between Resources* figure shows how resources are connected.

   Figure 8-6: Drag Connections Between Resources



4. Generate a resource stack.

   Click the **Generate Resource Stack** button. On the **Create Resource Stack** page, configure the following parameters:

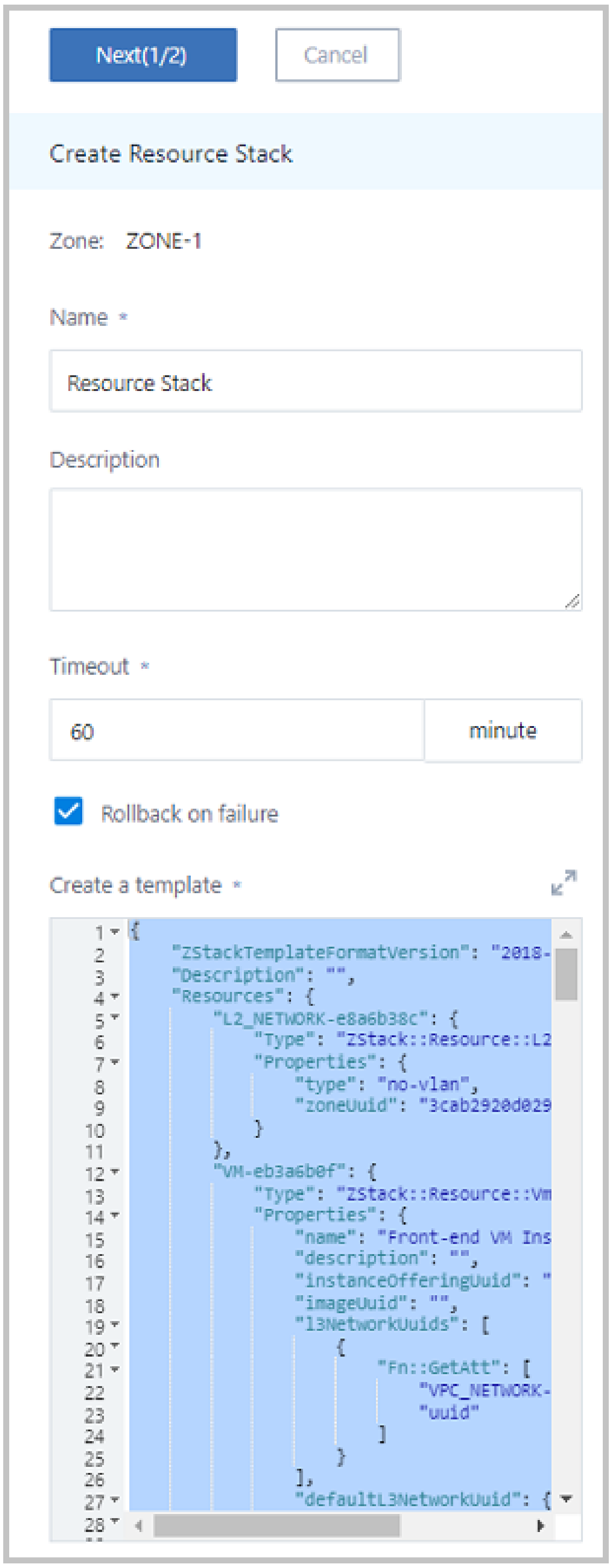
   - **Zone**: The zone to which the resource stack belongs. The value is defaulted to the name of current zone. You do not need to set this parameter.
   - **Name**: The name of the resource stack.
   - **Description**: Optional. The description of the resource stack.
   - **Timeout**: The length of time before a resource stack creation times out. Default value: 60. Unit: minutes.

- **Rollback on failure**: Specifies whether to delete created resources in case of a creation failure of a resource stack. This checkbox is selected by default.
- **Create a template**: The script that the designer generates based on your resource configuration.

Click **Next(1/2)** > **OK**.

The *Figure 8-7: Create Resource Stack* figure shows the parameters of a resource stack.

Figure 8-7: Create Resource Stack



Create Resource Stack

Zone: ZONE-1

Name *

Resource Stack

Description

Timeout *

60 minute

☑ Rollback on failure

Create a template *

```
{
    "ZStackTemplateFormatVersion": "2018-
    "Description": "",
    "Resources": {
        "L2_NETWORK-e8a6b38c": {
            "Type": "ZStack::Resource::L2
            "Properties": {
                "type": "no-vlan",
                "zoneUuid": "3cab2920d029
            }
        },
        "VM-eb3a6b0f": {
            "Type": "ZStack::Resource::Vm
            "Properties": {
                "name": "Front-end VM Ins
                "description": "",
                "instanceOfferingUuid": "
                "imageUuid": "",
                "l3NetworkUuids": [
                    {
                        "Fn::GetAtt": [
                            "VPC_NETWORK-
                            "uuid"
                        ]
                    }
                ],
                "defaultL3NetworkUuid": {
```

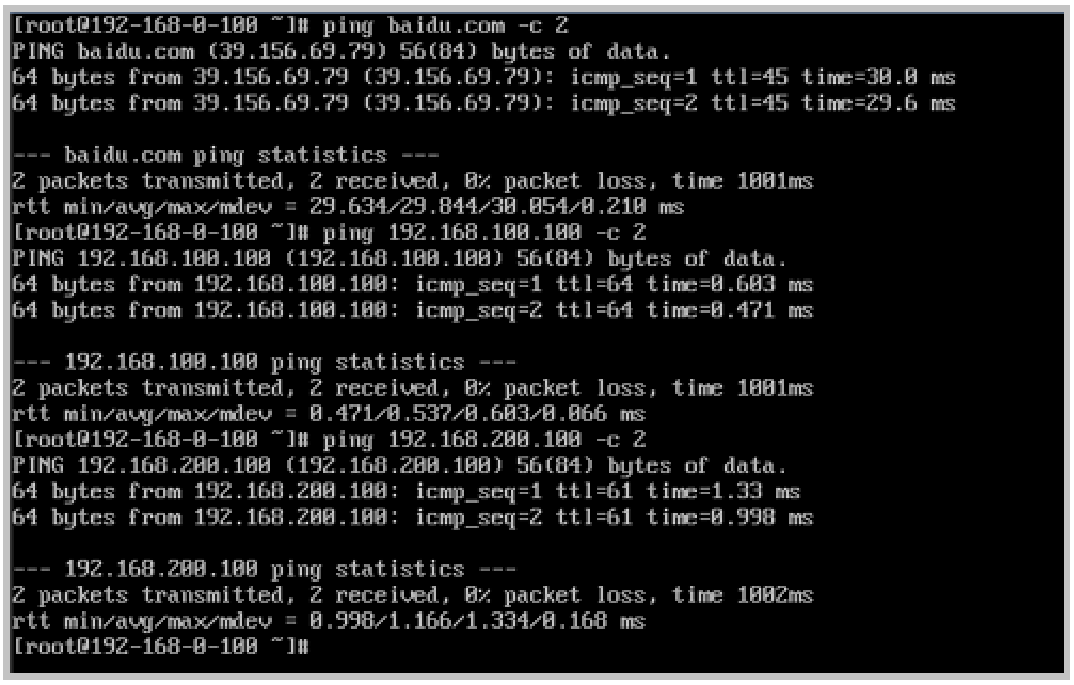
5. Check whether the three VM instances are connected.

Use the **`ping`** command to check whether the three VM instances are connected.

Expected results: The three VM instances can connect to each other and can access the Internet.

Actual results: The three VM instances are connected, as shown in the *Figure 8-8: Check Connection of VM Instances* figures.

Figure 8-8: Check Connection of VM Instances

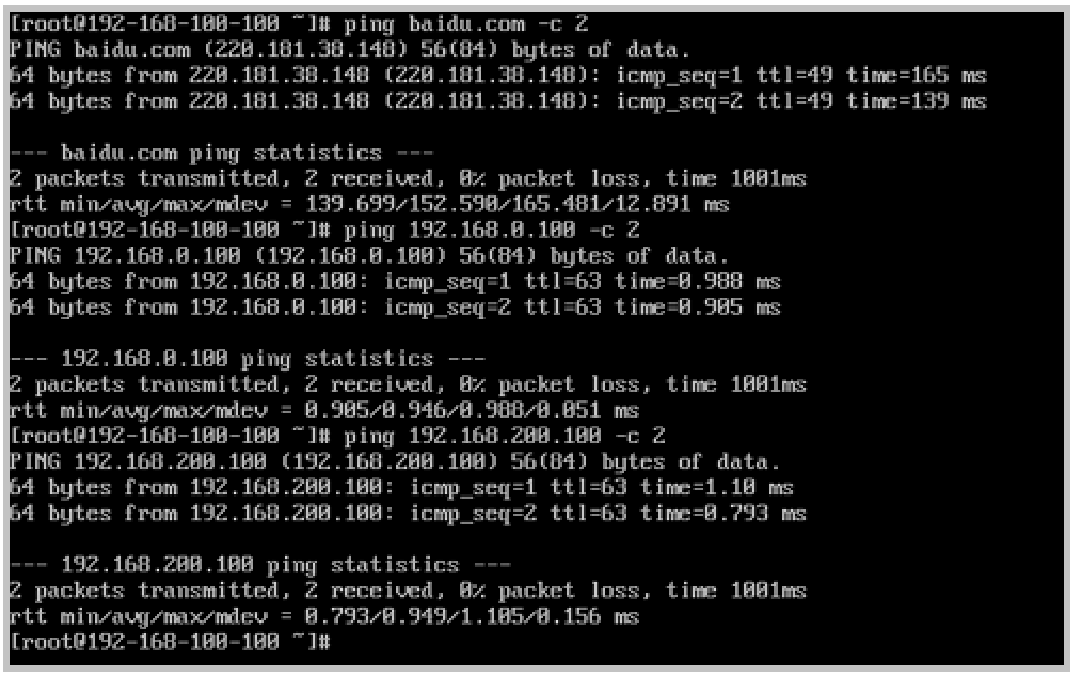


```
[root@192-168-0-100 ~]# ping baidu.com -c 2
PING baidu.com (39.156.69.79) 56(84) bytes of data.
64 bytes from 39.156.69.79 (39.156.69.79): icmp_seq=1 ttl=45 time=30.0 ms
64 bytes from 39.156.69.79 (39.156.69.79): icmp_seq=2 ttl=45 time=29.6 ms

--- baidu.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 29.634/29.844/30.054/0.210 ms
[root@192-168-0-100 ~]# ping 192.168.100.100 -c 2
PING 192.168.100.100 (192.168.100.100) 56(84) bytes of data.
64 bytes from 192.168.100.100: icmp_seq=1 ttl=64 time=0.603 ms
64 bytes from 192.168.100.100: icmp_seq=2 ttl=64 time=0.471 ms

--- 192.168.100.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.471/0.537/0.603/0.066 ms
[root@192-168-0-100 ~]# ping 192.168.200.100 -c 2
PING 192.168.200.100 (192.168.200.100) 56(84) bytes of data.
64 bytes from 192.168.200.100: icmp_seq=1 ttl=61 time=1.33 ms
64 bytes from 192.168.200.100: icmp_seq=2 ttl=61 time=0.998 ms

--- 192.168.200.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.998/1.166/1.334/0.168 ms
[root@192-168-0-100 ~]#
```

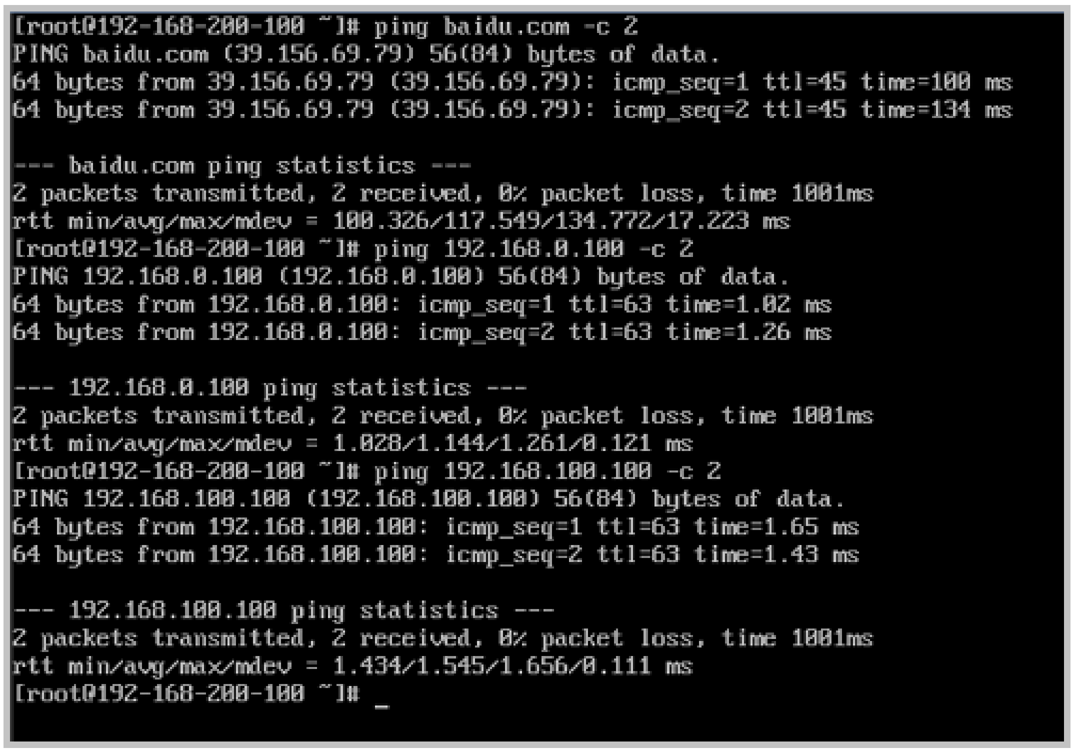


```
[root@192-168-100-100 ~]# ping baidu.com -c 2
PING baidu.com (220.181.38.148) 56(84) bytes of data.
64 bytes from 220.181.38.148 (220.181.38.148): icmp_seq=1 ttl=49 time=165 ms
64 bytes from 220.181.38.148 (220.181.38.148): icmp_seq=2 ttl=49 time=139 ms

--- baidu.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 139.699/152.590/165.481/12.891 ms
[root@192-168-100-100 ~]# ping 192.168.0.100 -c 2
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=63 time=0.988 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=63 time=0.905 ms

--- 192.168.0.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.905/0.946/0.988/0.051 ms
[root@192-168-100-100 ~]# ping 192.168.200.100 -c 2
PING 192.168.200.100 (192.168.200.100) 56(84) bytes of data.
64 bytes from 192.168.200.100: icmp_seq=1 ttl=63 time=1.10 ms
64 bytes from 192.168.200.100: icmp_seq=2 ttl=63 time=0.793 ms

--- 192.168.200.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.793/0.949/1.105/0.156 ms
[root@192-168-100-100 ~]#
```



```
[root@192-168-200-100 ~]# ping baidu.com -c 2
PING baidu.com (39.156.69.79) 56(84) bytes of data.
64 bytes from 39.156.69.79 (39.156.69.79): icmp_seq=1 ttl=45 time=100 ms
64 bytes from 39.156.69.79 (39.156.69.79): icmp_seq=2 ttl=45 time=134 ms

--- baidu.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 100.326/117.549/134.772/17.223 ms
[root@192-168-200-100 ~]# ping 192.168.0.100 -c 2
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=63 time=1.02 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=63 time=1.26 ms

--- 192.168.0.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.028/1.144/1.261/0.121 ms
[root@192-168-200-100 ~]# ping 192.168.100.100 -c 2
PING 192.168.100.100 (192.168.100.100) 56(84) bytes of data.
64 bytes from 192.168.100.100: icmp_seq=1 ttl=63 time=1.65 ms
64 bytes from 192.168.100.100: icmp_seq=2 ttl=63 time=1.43 ms

--- 192.168.100.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.434/1.545/1.656/0.111 ms
[root@192-168-200-100 ~]# _
```

What's next

In this topic, you learned how to use the designer of CloudFormation to deploy different services in different VPC networks.

9 Appendices

9.1 Stack Template Syntax

Stack template is a UTF8-encoded file.

You can quickly create a resource stack based on a stack template. With a stack template, you can define what cloud resources you need, the dependency between the resources, and the resource configuration. CloudFormation analyzes the stack template and automatically creates and configures all resources.

Stack Template Structure

The stack template structure is as follows.

```
{
    "ZStackTemplateFormatVersion" : "YYYY-MM-DD",
    "Description" : "The description of the stack template, which is
 used to provide information such as application scenarios and the
structure of the stack template.",
    "Parameters" : {
      // The parameters you can specify when creating a resource stack
.
    },
    "Mappings" : {
      // The mapping tables. Mapping tables are nested tables.
    },
    "Resources" : {
      // The detailed information of resources, including configurat
ions and dependencies.
    },
    "Outputs" : {
      // The outputs that are used to provide useful information such
as resource properties. You can use API to obtain this information.
  }
```

- **`ZStackTemplateFormatVersion` (Required)**

  The version of the stack template.

  - Format: `YYYY-MM-DD`
- **`Description` (Optional)**

  The description of the stack template, which is used to provide information such as application scenarios and the structure of the stack template.

  - A detailed description can help users better understand the content of the stack template.
- **`Parameters` (Optional)**

  The parameters you can specify when creating a resource stack.

  - For example, an instance offering is often defined as a parameter.
  - Parameters have default values.
  - Parameters can improve the flexibility and reusability of the stack template.
  - For more information about **Parameters**, see the *Parameters* topic.
- **`Mappings` (Optional)**

  The mapping tables. Mapping tables are nested tables.

  - You can use `Fn::FindInMap` to select values through corresponding keys.
  - You can use parameter values as keys.
  - For example, you can search the region-image mapping table for desired images by region.
  - For more information about **Mappings**, see the *Mappings* topic.
- **`Resources` (Optional)**

  The detailed information of resources, including configurations and dependencies.

  - For more information about **Resources**, see the *Resources* topic.
- **`Outputs` (Optional)**

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

  - For more information about **Outputs**, see the *Outputs* topic.

9.1.1 Parameters

The parameters you can specify when creating a resource stack.

- When you create a stack template, you can use parameters to improve the flexibility and reusability of the stack template.
- When you create a resource stack, you can specify parameter values as needed.

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.

The properties of `Parameters` are as follows.

Property	Description	Required	Example
Type	The parameter type. Options: • `String` • `Number` (An integer or floating-point number) • `CommaDelimitedList` (List<String> in Java equivalently) • `Boolean`	Yes	`"Type": "String"`
Label	The alias of the parameter. When forms are previewed or generated using templates, labels can be mapped to parameter names.	No	`"Label": "The password of the VM instance"`
Description	The string that describes the parameter.	No	`"Description": "The login password of the VM instance"`
NoEcho	Specifies whether to mask a parameter value as asterisks (*****). If you set this property to true, CloudFormation returns the parameter value masked as asterisks (*****).	No	`"NoEcho": true` **Note:** This property cannot be configured currently.
DefaultValue	The default value of the parameter.	No	`"DefaultValue": "password"`

CloudFormation also provides some pseudo parameters.

- Pseudo parameters are parameters that are predefined by CloudFormation. They can be referenced directly. You do not need to declare them in `Parameters`. (You are actually not allowed to declare them.)

- The values of the pseudo parameters are determined when CloudFormation is running.

Supported pseudo parameters are as follows.

Pseudo Parameter Name	Description
ZStack::StackName	The name of the current resource stack
ZStack::StackUuid	The UUID of the current resource stack
ZStack::AccountUuid	The AccountUuid of the current resource stack
ZStack::AccountName	The AccountName of the current resource stack

Example

The following example shows the `Parameters` syntax.

```
"Parameters" : {
  "username" : {
    "Label": "Login name",
    "Description" : "Login name",
    "DefaultValue": "root",
    "Type" : "String"
  },
  "password" : {
    "Label": "Password",
    "NoEcho" : "true",
    "Description" : "Login password",
    "Type" : "String",
  }
}
```

In this example, two parameters are declared in `Parameters`.

- `username`
  - A string type parameter with a default value of **root**.
  - The username must contain **2** to **12** characters.

  Note:

  The default value of `username` must also meet the length and valid values requirements.
- `password`
  - A string type parameter with no default value.
  - If you set `NoEcho` to true, CloudFormation returns the parameter value masked as asterisks (*****).

Note:

The `NoEcho` property cannot be configured currently.

- The password must contain **6** to **41** characters.
- The password can contain uppercase/lowercase letters and digits.

9.1.2 Resources

The detailed information of resources, including configurations and dependencies.

- `Resources` can reference `Parameters`, `Mappings`, and `Functions`.
- `Resources` can be referenced by other `Resources` and `Outputs`.

Syntax

Each resource consists of a logical UUID and a description.

- All resource descriptions are enclosed in braces `{ }`.
- Multiple resources are separated with commas `,`.

The key fields of `Resources` are as follows.

Key Field	Description	Required	Example
Type	The type of the resource that is being declared. Options: • `Resource` • `Action`	Yes	• `"Type": " ZStack:: Resource:: VmInstance"` • `"Type": " ZStack::Action ::AddIpRange"` • For more information, see *`Type`*
Properties	The resource properties that specify parameters for resource creation.	Yes	For more information, see *`Properties`*
DependsOn	Specifies that the creation of a specific resource follows another.	No	• `"DependsOn ": [{"Ref": " WebServer1"}]`

Key Field	Description	Required	Example
			• For more information, see *DependsOn*
DeletionPolicy	• Specifies whether to retain a resource after its stack is deleted. Options: — `Retain` — `Delete` • If this field is set to `Retain`, a specific resource and its dependent resource will be retained. (The system automatically retains the dependent resource.) • The default value is `Delete`.	No	• `"DeletionPolicy": "Retain"` • For more information, see *DeletionPolicy*
Description	The string that describes the resource .	No	• `"Description" : "attach ip range to l3 network"`

Example

The following example shows the `Parameters` syntax.

```
"Resources" : {
    "UUID-1" : {
        "Description" : "The resource description",
        "Type" : "The resource type",
        "Properties" : {
            The resource properties
        }
    },
    "UUID-2" : {
        "Description" : "The resource description"
        "Type" : "The resource type",
        "Properties" : {
            The resource properties
        },
        "DependsOn":"The dependent resource. Take UUID-1 for example.
Note that this dependent resource should be contained in the context
.",
        "DeletionPolicy":"The deletion policy"
    }
}
```

In this example, two resources are declared in `Resources`. The description of key fields are as follows:

- `Resource UUID`
  - `UUID-1` and `UUID-2` are the resource UUIDs. Both of them are variables.
  - You can use the resource UUID to reference the resource in other parts of the template.
  - The resource UUID is unique in a template.
- `Type`
  - The type of the resource that is being declared, including the `Resource` type and `Action` type.
  - For example, `"Type": "ZStack::Resource::VmInstance"` indicates that the resource is a VM instance. `"Type": "ZStack::Action::AddIpRange"` indicates the IP range to be added.
  - For more information about the resource types supported by CloudFormation, see the *Resource Index* topic.
- `Properties`
  - The resource properties that specify parameters for resource creation.
  - The following example shows the `Properties` syntax.

    ```
    "Resources" : {
        "InstanceOffering" : {
            "Type" : "ZStack::InstanceOffering",
            "Properties" : {
                "cpuNum" : "1",
                "cpuSpeed" : "1",
                "memorySize" : "1073741824",
                "name" : "instance-offering",
                "type" : "UserVm",
                "sortKey": 0,
                "allocatorStrategy": "LeastVmPreferredHostAllocatorS
    trategy"
            }
        }
    }
    ```
  - The rules of defining resource property values are as follows:

- Property values can be text strings, string lists, boolean values, references, 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 references are enclosed with braces `{}`.
- The preceding rules also apply when property values are the combinations of text strings , string lists, references, and return values of functions.
- The following example 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
" } ] ] }
}
```

— If the resource does not require you to declare a property, this part can be skipped.

- `DependsOn`

— With the `DependsOn` attribute, you can specify that the creation of a specific resource follows another.

— When you add a `DependsOn` attribute to a resource, the resource is created only after the creation of the resource specified in the `DependsOn` attribute.

— The following example shows the `DependsOn` syntax.

```
{
    "ZStackTemplateFormatVersion" : "2018-06-18",
    "Resources" : {
        "WebServer": {
            "Type": "ZStack::Resource::VmInstance",
            "DependsOn": "DatabseServer"
        },
        "DatabseServer": {
            "Type": "ZStack::Resource::VmInstance",
            "Properties": {
                "name": {"Fn::Join":["-",[{"Ref":"ZStack::
StackName"},"VM"]]},
                "instanceOfferingUuid": {"Ref":"InstanceOf
feringUuid"},
                "imageUuid":{"Ref":"ImageUuid"},
                "l3NetworkUuids":[{"Ref":"PrivateNetworkUuid"}],
                "dataDiskOfferingUuids":[{"Ref":"DiskOfferingUuid
"}],
          "hostUuid":{"Ref":"HostUuid"}
            }
        }
    }
}
```

This example indicates that WebServer is created only after DatabaseServer is created.

- `DeletionPolicy`

— `DeletionPolicy` specifies whether to retain a resource after its stack is deleted.

— `DeletionPolicy` has two options: `Retain` and `Delete`.

  - `Delete`: The default value, which indicates that after a resource stack is deleted, all resources in the stack will be deleted.
  - `Retain`: If the `DeletionPolicy` attribute is set to `Retain`, the specific resource will be retained after its stack is deleted. Furthermore, the dependent resource of the resource will also be retained. (The system automatically retains the dependent resource.)

  The following example shows that the VM instance is retained after the template-based resource stack is deleted.

```
"Resources" : {
    "VMInstance" : {
        "Type" : "ZStack::Resource::VmInstance",
        "Properties": {
            "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName
"},"VM"]]},
            "instanceOfferingUuid": {"Ref":"InstanceOf
feringUuid"},
            "imageUuid":{"Ref":"ImageUuid"},
            "l3NetworkUuids":[{"Ref":"PrivateNetworkUuid"}],
            "dataDiskOfferingUuids":[{"Ref":"DiskOfferingUuid
"}],
      "hostUuid":{"Ref":"HostUuid"}
    },
    "DeletionPolicy" : "Retain"
  }
}
```

9.1.3 Outputs

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

Syntax

Each output item consists of a UUID and a description.

- All output descriptions are enclosed in braces `{}`.
- Multiple output items are separated with commas `,`.

The key fields of `Outputs` are as follows.

Key Field	Description	Required	Example
Description	The string that describes the output.	No	• `"Description" : "print l3 network"` • For more information, see *`Description`*
Value	The content of the output.	Yes	• `"Value" : {"Ref": " WebServer1"}` • For more information, see *`Value`*

Example

The following example shows the `Outputs` syntax.

```
"Outputs" : {
    "UUID-1" : {
        "Description" : "The output description",
        "Value" : "The output content"
    },
    "UUID-2" : {
        "Description" : "The output description",
        "Value" : "The output content"
    }
}
```

In this example, two outputs are declared in `Outputs`. The description of key fields are as follows:

- `Output UUID`

  — The output UUID is unique in a template.

- `Description`

  — The description of the output item. The description must be a string.

- `Value`

  — The content of the output.

  — The following example shows the `Value` syntax.

```
{
 "ZStackTemplateFormatVersion": "2018-06-18",
 "Description": "This template creates a VM instance with a
volume attached (based on the local storage). Prerequisites: The
```

```
 instance offering, image, disk offering, private network, and
available host are created.",
 "Parameters": {
     "InstanceOfferingUuid": {
         "Type": "String",
               "Label": "Instance Offering",
   "Description": "The instance offering uuid"
     },
     "ImageUuid": {
   "Type": "String",
               "Label": "Image",
   "Description": "The Image uuid for creating VmInstance. Please
choose an image not iso"
     },
     "PrivateNetworkUuid": {
   "Type": "String",
               "Label": "Private Network",
   "Description" : "The private network uuid for creating
VmInstance"
           },
     "DiskOfferingUuid": {
   "Type": "String",
               "Label": "Disk Offering",
   "Description": "Volume size offering uuid"
     },
     "HostUuid": {
   "Type": "String",
               "Label": "Host",
   "Description": "Host uuid, that vm will start on"
     }
 },
 "Resources": {
     "VmInstance": {
   "Type": "ZStack::Resource::VmInstance",
   "Properties": {
       "name": {"Fn::Join":["-",[{"Ref":"ZStack::StackName"},"VM
"]]},
       "instanceOfferingUuid": {"Ref":"InstanceOfferingUuid"},
       "imageUuid":{"Ref":"ImageUuid"},
       "l3NetworkUuids":[{"Ref":"PrivateNetworkUuid"}],
       "dataDiskOfferingUuids":[{"Ref":"DiskOfferingUuid"}],
       "hostUuid":{"Ref":"HostUuid"}
   }
     }
 },
 "Outputs": {
     "VmInstance": {
         "Value": {
       "Ref": "VmInstance"
   }
     }
 }
}
```

In this example, there is one output item, which contains the property value of `VmInstance`.

9.1.4 Functions

CloudFormation provides several intrinsic functions for your resource stack management. You can use intrinsic functions to define `Resources`, `Outputs`, and `Mappings`.

Supported intrinsic functions are as follows:

- `Fn::Base64`
- `Fn::FindInMap`
- `Fn::GetAtt`
- `Fn::Join`
- `Fn::Split`
- `Fn::Select`
- `Ref`
- `Fn::If`
- `Fn::Equal`
- `Fn::And`
- `Fn::Not`
- `Fn::Or`

Fn::Base64

It returns the Base64 representation of an input string.

- **Declaration**

  ```
  "Fn::Base64" : stringToEncode
  ```

- **Parameter**

  ▬ `stringToEncode`: The string value you want to convert to Base64.

- **Example**

  ```
  "Fn::Base64" : "password"
  ```

- **Return Value**

  The Base64 representation of the input string.

  `"cGFzc3dvcmQ="` is returned in this example.

Fn::FindInMap

It returns the values corresponding to keys in a two-level mapping that is declared in `Mappings`.

- **Declaration**

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

- **Parameters**

- `MapName`: The ID of a mapping declared in `Mappings` that contains the 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 number.

- **Example**

```
"Fn::FindInMap" : ["RegionMap", "cn-shanghai", "32"]
```

- **Return Value**

  The value that is assigned to `SecondLevelKey`.

  - `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 `"cn-shanghai"` in this example.
  - `SecondLevelKey` is set to the required architecture, which is `"32"` in this example.

- **Supported Functions**

  The following functions can be nested in `Fn::FindInMap`:

  - `Fn::FindInMap`
  - `Ref`

Fn::GetAtt

It returns the value of a property from a resource in a template.

- **Declaration**

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

- **Parameters**

  - `resourceUuid`: The UUID of the resource.
  - `attributeName`: The name of the resource property.

- **Example**

```
"Fn::GetAtt" : ["MyVMInstance", "ImageUuid"]
```

- **Return Value**

  The property value.

  In this example, the returned resource UUID is the `"ImageUuid"` property of `"MyVMInstance"`.

Fn::Join

It appends a set of values into a single value separated with a specified delimiter.

- **Declaration**

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

- **Parameters**

  - `delimiter`: The value used to divide a 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.

- **Example**

```
"Fn::Join" : ["-", ["a", "b", "c"]]
```

- **Return Value**

  The combined string.

  `"a-b-c"` is returned in this example.

- **Supported Functions**

  The following functions can be nested in `Fn::Join`:

  - `Fn::Base64`
  - `Fn::GetAtt`
  - `Fn::Join`
  - `Fn::Select`
  - `Ref`

Fn::Split

It splits a string into a list of values separated with a specified delimiter.

- **Declaration**

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

- **Parameters**

  - `delimiter`: The value used to divide a string. This value can be a comma `,`, a semicolon `;`, a `\n`, or a `\t`.
  - `original_string`: The string to be split.

- **Example**

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

- **Return Value**

  A list of string values.

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

- **Supported Functions**

  The following functions can be nested in `Fn::Split`:

  - `Fn::Base64`
  - `Fn::FindInMap`
  - `Fn::GetAtt`
  - `Fn::Join`
  - `Fn::Select`
  - `Ref`

Fn::Select

It returns a single object from a list of objects by using an index.

- **Declaration**

  - If the list of objects is an array, the syntax is as follows.

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

  - If the list of objects is a mapping table, the syntax is as follows.

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

- **Parameters**

  - `index`: The index of the object you want to retrieve.

    - If the list of objects 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 objects 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.

- **Examples**

- Example 1: The list of objects is an array.

```
"Fn::Select" : ["2", ["foo", " bar", "achoo"]]
```

- Example 2: The list of objects is a mapping table.

```
"Fn::Select" : ["shape", {"shape": "circle", "height": "80"}]
```

- Example 3: The list of objects is a `CommaDelimitedList`.

```
"Parameters" : {
 "userParam": {
           "Type": "CommaDelimitedList",
           "Default": "10.0.100.0/24, 10.0.101.0/24, 10.0.102.0/24
"
       }
},
"Resources": {
 "resourceUuid": {
           "Properties": {
               "CidrBlock": {"Fn::Select" : [0, {"Ref":"userParam
"}]}
           }
       }
}
```

Note:

- If there is no nested function in `Fn::Select`, the first parameter is a string. For more information, see Example 1 (`"2"`) and Example 2 (`"shape"`).
- If there is a nested function in `Fn::Select`, the first parameter is a number. For more information, see Example 3 (`0`).

- **Return Value**

  The selected object.

  - Example 1: `"achoo"` is returned.
  - Example 2: `"circle"` is returned.
  - Example 3: `"10.0.100.0/24"` is returned.

- **Supported Functions**

  - For the `Fn::Select` index value, you can use `Ref` as a nested function in `Fn::Select`.
  - For the `Fn::Select` list of objects, you can use the following as nested functions in `Fn::Select`:
    - `Fn::Base64`
    - `Fn::FindInMap`

- `Fn::GetAtt`
- `Fn::Join`
- `Fn::Select`
- `Ref`

Ref

It returns the value of a specified parameter or resource.

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

- **Declaration**

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

- **Parameter**

  — `logicalName`: The logical name of the resource or parameter to reference.

- **Example**

  `diskOfferingParam` is specified as follows.

```
"diskOfferingParam": {
 "allocatorStrategy": "DefaultPrimaryStorageAllocationStrategy",
       "diskSize": "21474836480",
       "type": "DefaultDiskOfferingType",
       "sorkKey": "0"
}
```

```
"Ref":"diskOfferingParam"
```

- **Return Value**

  The value of the resource or parameter.

  In this example, the value of `diskOfferingParam` is returned.

```
{
 "allocatorStrategy": "DefaultPrimaryStorageAllocationStrategy",
       "diskSize": "21474836480",
       "type": "DefaultDiskOfferingType",
       "sorkKey": "0"
}
```

- **Supported Functions**

  You are not allowed to use any nested functions in `Ref`. `Ref` should be specified as a string of `resourceUuid`.

Fn::If

If a specified condition is evaluated as `true`, a value is returned. If the specified condition is evaluated as `false`, a different value is returned.

- The `Resources` and `Outputs` property values in templates support the `Fn::If` function.
- **Declaration**

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

- **Parameters**
  - `condition_name`: The name of the condition in `Conditions`. 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.
- **Example**

```
"Fn::If": ["condition16", "vm-true", "vm-false"]
```

- **Return Value**

  The value based on an evaluated result of a specified condition.

  In this example, if `condition16` is evaluated as `true`, `vm-true` is returned. If `condition16` is evaluated as `false`, `vm-false` is returned.
- **Supported Functions**

  The following functions can be nested in `Fn::If`:

  - `Fn::FindInMap`
  - `Ref`
  - `Fn::Equal`
  - `Fn::And`
  - `Fn::Not`
  - `Fn::Or`

Fn::Equal

It compares 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::Equal": [value_1, value_2]
```

- **Parameter**

  — `value`: The values to be compared. These values can be of any type.

- **Example**

```
"Fn::Equal": [true, false]
```

- **Return Value**

  `true` or `false`.

  In this example, `false` is returned.

- **Supported Functions**

  The following functions can be nested in `Fn::Equal`:

  - `Fn::FindInMap`
  - `Ref`
  - `Fn::Equal`
  - `Fn::And`
  - `Fn::Not`
  - `Fn::Or`

Fn::And

It represents 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, ...]
```

- **Parameter**

  — `condition`: The condition to be evaluated.

- **Example**

```
"Fn::And": [true, false]
```

- **Return Value**

  `true` or `false`.

In this example, `false` is returned.

- **Supported Functions**

  The following functions can be nested in `Fn::And`:

  - `Fn::FindInMap`
  - `Ref`
  - `Fn::Equal`
  - `Fn::And`
  - `Fn::Not`
  - `Fn::Or`

Fn::Not

It represents 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
  ```

- **Parameter**

  — `condition`: The condition to be evaluated.

- **Example**

  ```
  "Fn::Not": true
  ```

- **Return Value**

  `true` or `false`.

  In this example, `false` is returned.

- **Supported Functions**

  The following functions can be nested in `Fn::Not`:

  - `Fn::FindInMap`
  - `Ref`
  - `Fn::Equal`
  - `Fn::And`
  - `Fn::Not`
  - `Fn::Or`

Fn::Or

It represents 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, ...]
```

- **Parameter**

  — `condition`: The condition to be evaluated.

- **Example**

```
"Fn::Or": [true, false]
```

- **Return Value**

  `true` or `false`.

  In this example, `true` is returned.

- **Supported Functions**

  The following functions can be nested in `Fn::Or`:

  - `Fn::FindInMap`
  - `Ref`
  - `Fn::Equal`
  - `Fn::And`
  - `Fn::Not`
  - `Fn::Or`

9.1.5 Mappings

The mapping tables. Mapping tables are nested tables.

- 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 `Key-Value` pairs.

- Both the keys and values can be strings or numbers.
- Multiple mappings are separated with commas `,`.
- Mapping names must be unique.

Example

The following example shows the `Mappings` syntax.

```
"Mappings" : {
 "Mapping01" : {
            "Key01" : {
                "Name" : "Value01"
            },
            "Key02" : {
                "Name" : "Value02"
            },
            "Key03" : {
                "Name" : "Value03"
            }
        }
}
```

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

```
{
 "ZStackTemplateFormatVersion": "2018-06-18",
        "Parameters": {
            "regionParam": {
                "Description": "Select the region for VM instance
creation.",
                "Type": "String",
                "AllowedValues": ["cn-hangzhou", "cn-shanghai"]
            }
        },
        "Mappings" : {
            "ImageInRegions" : {
                "cn-hangzhou" : { "32" : "imageUuid-1", "64" : "
imageUuid-2" },
                "cn-shanghai" : { "32" : "imageUuid-3", "64" : "
imageUuid-4" }
            }
        },
        "Resources": {
            "WebServer": {
                "Type": "ZStack::Resource::VmInstance",
                "Properties": {
                    "name" : "test-vm",
                    "imageUuid" : {"Fn::FindInMap": ["ImageInRegions",
 {"Ref":"regionParam"}, "64"]},
                    "instanceOfferingUuid": {"Ref":"instanceOf
feringUuid"},
                    "l3NetworkUuids": [{"Ref":"l3NetworkUuid"}]
                },
                "DeletionPolicy": "Retain"
           }
        }
}
```

9.2 Resource Index

When you create a stack template, you can use `Type` and `Properties` to declare detailed requirements of the resources you need.

Two resource `types` supported by CloudFormation are as follows:

- `Resource`
- `Action`

9.2.1 Resource

Table 9-1: Resource Index of the `Resource` Type

Resource	References
ZStack::Resource::VmInstance	*CreateVmInstance*
ZStack::Resource::DataVolume	*CreateDataVolume*
ZStack::Resource::Image	*AddImage*
ZStack::Resource::RootVolumeTemplate	*CreateRootVolumeTemplateFromRootVolume*
ZStack::Resource::DataVolumeTemplate	*CreateDataVolumeTemplateFromVolume*
ZStack::Resource::AffinityGroup	*CreateAffinityGroup*
ZStack::Resource::InstanceOffering	*CreateInstanceOffering*
ZStack::Resource::DiskOffering	*CreateDiskOffering*
ZStack::Resource::L2VxlanNetworkPool	*CreateL2VxlanNetworkPool*
ZStack::Resource::L2NoVlanNetwork	*CreateL2NoVlanNetwork*
ZStack::Resource::L2VlanNetwork	*CreateL2VlanNetwork*
ZStack::Resource::L2VxlanNetwork	*CreateL2VxlanNetwork*
ZStack::Resource::L3Network	*CreateL3Network*
ZStack::Resource::VRouterRouteTable	*CreateVRouterRouteTable*
ZStack::Resource::VpcVRouter	*CreateVpcVRouter*
ZStack::Resource::SecurityGroup	*CreateSecurityGroup*
ZStack::Resource::SecurityGroupRule	*AddSecurityGroupRule*
ZStack::Resource::Vip	*CreateVip*
ZStack::Resource::Eip	*CreateEip*

Resource	References
ZStack::Resource::PortForwardingRule	*CreatePortForwardingRule*
ZStack::Resource::LoadBalancer	*CreateLoadBalancer*
ZStack::Resource::LoadBalancerListener	*CreateLoadBalancerListener*
ZStack::Resource::IPsecConnection	*CreateIPsecConnection*
ZStack::Resource::VirtualRouterOffering	*CreateVirtualRouterOffering*
ZStack::Resource::VniRange	*CreateVniRange*
ZStack::Resource::UserTag	*CreateUserTag*
ZStack::Resource::DataVolumeFromVolume Template	*CreateDataVolumeFromVolumeTemplate*
ZStack::Resource::Tag	*CreateTag*

9.2.2 Action

Table 9-2: Resource Index of the `Action` Type

Action	References
ZStack::Action::AddIpRange	*AddIpRange*
ZStack::Action::AddDnsToL3Network	*AddDnsToL3Network*
ZStack::Action::AddVmToAffinityGroup	*AddVmToAffinityGroup*
ZStack::Action::AddVRouterRouteEntry	*AddVRouterRouteEntry*
ZStack::Action::AddCertificateToLoad BalancerListener	*AddCertificateToLoadBalancerListener*
ZStack::Action::AddIpRangeByNetworkCidr	*AddIpRangeByNetworkCidr*
ZStack::Action::AddVmNicToLoadBalancer	*AddVmNicToLoadBalancer*
ZStack::Action::AddVmNicToSecurityGroup	*AddVmNicToSecurityGroup*
ZStack::Action::AddRemoteCidrsToIPse cConnection	*AddRemoteCidrsToIPsecConnection*
ZStack::Action::AttachEip	*AttachEip*
ZStack::Action::AttachDataVolumeToVm	*AttachDataVolumeToVm*
ZStack::Action::AttachPortForwardingRule	*AttachPortForwardingRule*
ZStack::Action::AttachIsoToVmInstance	*AttachIsoToVmInstance*
ZStack::Action::AttachPciDeviceToVm	*AttachPciDeviceToVm*

Action	References
ZStack::Action::AttachUsbDeviceToVm	*AttachUsbDeviceToVm*
ZStack::Action::AttachL2NetworkToCluster	*AttachL2NetworkToCluster*
ZStack::Action::AttachL3NetworkToVm	*AttachL3NetworkToVm*
ZStack::Action::AttachNetworkServiceToL3Network	*AttachNetworkServiceToL3Network*
ZStack::Action::AttachSecurityGroupToL3Network	*AttachSecurityGroupToL3Network*
ZStack::Action::AttachL3NetworksToIPsecConnection	*AttachL3NetworksToIPsecConnection*
ZStack::Action::AttachVRouterRouteTableToVRouter	*AttachVRouterRouteTableToVRouter*
ZStack::Action::AddCertificateToLoadBalancerListener	*AddCertificateToLoadBalancerListener*
ZStack::Action::AddHostRouteToL3Network	*AddHostRouteToL3Network*
ZStack::Action::SetL3NetworkRouterInterfaceIp	*SetL3NetworkRouterInterfaceIp*
ZStack::Action::AddDnsToVpcRouter	*AddDnsToVpcRouter*
ZStack::Action::AttachTagToResources	*AttachTagToResources*
ZStack::Action::UpdateTag	*UpdateTag*

Glossary

Zone

A zone is a logical group of resources such as clusters, L2 networks, and primary storages. Zone is the largest resource scope defined in ZStack.

Cluster

A cluster is a logical group of analogy hosts (compute nodes). Hosts in the same cluster must be installed with the same operating system, have the same network configuration, and be able to access the same primary storage. In a real data center, a cluster usually maps to a rack.

Management Node

A management node is a host with operating system installed to provide UI management and Cloud deployment.

Compute Node

A compute node is a physical server (also known as a host) that provides VM instances with compute, network, and storage resources.

Primary Storage

A primary storage is a storage server used to store disk files in VM instances. Local storage, NFS, Ceph, Shared Mount Point, and Shared Block are supported.

Backup Storage

A backup storage is a storage server used to store image template files. ImageStore, SFTP (Community Edition), and Ceph are supported. We recommend that you deploy backup storage separately.

ImageStore

ImageStore is a type of backup storage. You can use ImageStore to create images for VM instances that are in the running state and manage image version updates and release. ImageStore allows you quickly upload, download, export images, and create image snapshots as needed.

VM Instance

A VM instance is a virtual machine instance running on a host. A VM instance has its own IP address to access public network and run application services.

Image

An image is an image template used by a VM instance or volume. Image templates include system volume images and data volume images.

Volume

A volume can either be a data volume or a root volume. A volume provides storage to a VM instance. A shared volume can be attached to one or more VM instances.

Instance Offering

An instance offering is a specification of the VM instance CPU and memory, and defines the host allocator strategy, disk bandwidth, and network bandwidth.

Disk Offering

A disk offering is a specification of a volume, which defines the size of a volume and how the volume will be created.

L2 Network

An L2 network is a layer 2 broadcast domain used for layer 2 isolation. Generally, L2 networks are identified by names of devices on the physical network.

L3 Network

An L3 network is a collection of network configurations for VM instances, including the IP range, gateway, and DNS.

Public Network

A public network is generally allocated with a public IP address by Network Information Center (NIC) and can be connected to IP addresses on the Internet.

Private Network

A private network is the internal network that can be connected and accessed by VM instances.

L2NoVlanNetwork

L2NoVlanNetwork is a network type for creating an L2 network. If L2NoVlanNetwork is selected, VLAN settings are not used for host connection.

L2VlanNetwork

L2VlanNetwork is a network type for creating an L2 network. If L2VlanNetwork is selected, VLAN settings are used for host connection and need to be configured on the corresponding switches in advance.

VXLAN Pool

A VXLAN pool is an underlay network in VXLAN. You can create multiple VXLAN overlay networks (VXLAN) in a VXLAN pool. The overlay networks can operate on the same underlay network device.

VXLAN

A VXLAN network is a L2 network encapsulated by using the VXLAN protocol. A VXLAN network belongs to a VXLAN pool. Different VXLAN networks are isolated from each other on the L2 network.

vRouter

A vRouter is a custom Linux VM instance that provides various network services.

Security Group

A security group provides L3 network firewall control over the VM instances. It can be used to set different security rules to filter IP addresses, network packet types, and the traffic flow of network packets.

EIP

An elastic IP address (EIP) is a method to access a private network through a public network.

Snapshot

A snapshot is a point-in-time capture of data status in a disk. A snapshot can be either an automatic snapshot or a manual snapshot.