资源编排

用户指南

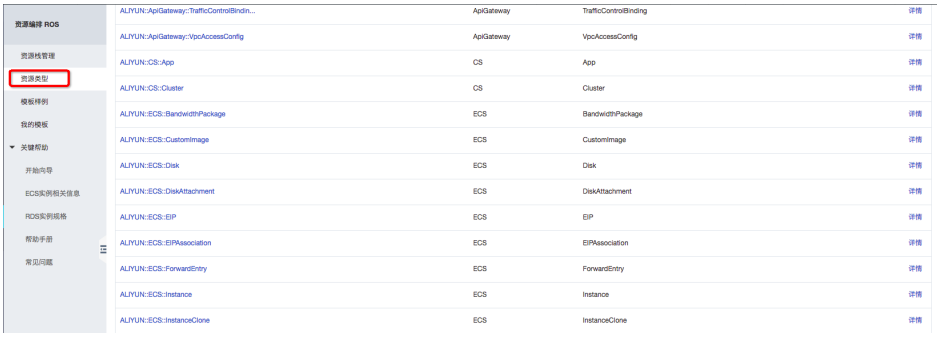
用户指南

查看资源类型详情

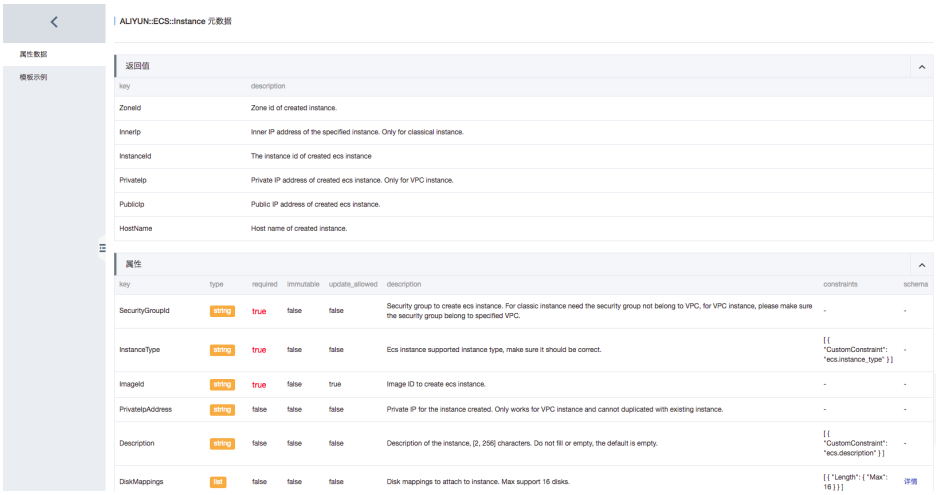
您可以通过控制台查看编排服务支持的资源类型详情。

1.进入 **ROS 控制台**；

2.选择 **资源类型**；



3.可以看到目前编排服务支持的所有资源类型。单击 **类型名称** 或者右侧的 **详情** 按钮，可以进行查看。例如单击 **ALIYUN::ECS::Instance**；



一个资源由**属性**和**返回值**两部分组成。属性在模板中申明所需资源的详细要求，例如 ECS 的规格，镜像，安全组等等；返回值则是资源编排创建资源后，用户可以通过资源栈获取的实例属值，例如 ECS 实例的 ID，私网 IP，公网 IP 等等。

4.每个属性的元信息包含下面几项：

key	属性名称

type	属性可取值类型
required	是否为必填属性
immutable	是否为不可变属性
update_allowed	是否支持更新，当属性支持更新的时候，用户可以通过修改模板中属性的值，然后通过资源栈的更新操作，变更已创建资源实例的属性。
description	该属性的详细说明
constraints	该属性可取值的范围
schema	该属性的子属性

1. 单击**模板示例**可以查看如何在模板中申明该资源

ALIYUN::CS::App	创建基于 Docker 集群的应用
ALIYUN::CS::Cluster	创建 Docker 集群
ALIYUN::ECS::Disk	创建数据盘
ALIYUN::ECS::DiskAttachment	把数据盘关联到相应的ECS
ALIYUN::ECS::EIP	购买弹性 IP
ALIYUN::ECS::EIPAssociation	把弹性 IP 关联到 相应的 ECS
ALIYUN::ECS::Snapshot	创建磁盘快照
ALIYUN::ECS::CustomImage	创建自定义镜像
ALIYUN::ECS::Instance	创建单个 ECS 实例
ALIYUN::ECS::InstanceClone	基于某个 ECS 克隆单个新的 ECS 实例
ALIYUN::ECS::InstanceGroup	创建多个 ECS 实例
ALIYUN::ECS::InstanceGroupClone	基于某个 ECS 实例克隆多个新的 ECS 实例

ALIYUN::ECS::SecurityGroup	创建安全组
ALIYUN::ECS::SecurityGroupEgress	配置安全组流量出规则
ALIYUN::ECS::SecurityGroupIngress	配置安全组流量入规则
ALIYUN::ECS::JoinSecurityGroup	添加 ECS 到安全组
ALIYUN::ECS::SecurityGroupClone	克隆安全组
ALIYUN::ECS::SSHKeyPair	创建 SSH 密钥对
ALIYUN::ECS::SSHKeyPairAttachment	绑定 SSH 密钥和 ECS 实例
ALIYUN::ECS::NatGateway	创建专有网络 NAT 网关
ALIYUN::ECS::BandwidthPackage	创建 NAT 网关所使用的带宽包
ALIYUN::ECS::ForwardEntry	配置 NAT 网关中的转发表
ALIYUN::ECS::SNatEntry	配置 NAT 网关中的 SNAT 表
ALIYUN::ECS::Route	配置专有网络路由表
ALIYUN::ECS::VPC	创建专有网络
ALIYUN::ECS::VSwitch	创建专有网络中的虚拟交换机
ALIYUN::ESS::ScalingConfiguration	创建伸缩配置
ALIYUN::ESS::ScalingGroup	创建伸缩组
ALIYUN::ESS::ScalingGroupEnable	开通伸缩组
ALIYUN::MEMCACHE::Instance	创建 memcache 实例
ALIYUN::MONGODB::Instance	创建 MongoDB 实例
ALIYUN::MarketPlace::Order	购买云市场资源
ALIYUN::MarketPlace::Image	购买云市场镜像
ALIYUN::MarketPlace::ImageSubscription	订阅云市场镜像
ALIYUN::OSS::Bucket	创建 OSS bucket 实例
ALIYUN::RAM::AccessKey	获取指定用户的 Access Key
ALIYUN::RAM::Group	创建 RAM 组
ALIYUN::RAM::ManagedPolicy	创建 RAM 策略
ALIYUN::RAM::Role	创建 RAM 角色
ALIYUN::RAM::User	创建 RAM 子账号
ALIYUN::RAM::UserToGroupAddition	添加子账号到 RAM 组
ALIYUN::RDS::DBInstance	创建 RDS 实例
ALIYUN::RDS::DBInstanceParameterGroup	修改 RDS 实例参数
ALIYUN::RDS::DBInstanceSecurityIps	配置 RDS 实例的白名单
ALIYUN::REDIS::Instance	创建 Redis 实例

ALIYUN::ROS::WaitCondition	处理 UserData 返回消息
ALIYUN::ROS::WaitConditionHandle	给 UserData 提供返回消息的命令并接收消息
ALIYUN::SLB::BackendServerAttachment	添加 ECS 实例到 SLB 实例
ALIYUN::SLB::Listener	配置 SLB 的监听参数
ALIYUN::SLB::LoadBalancer	创建 SLB 实例
ALIYUN::SLB::LoadBalancerClone	基于某个 SLB 实例克隆单个新的 SLB 实例
ALIYUN::SLB::VServerGroup	创建虚拟服务器组并添加该虚拟服务器组到 SLB 实例
ALIYUN::SLB::BackendToVServerGroupAddition	添加后端服务器到已存在的虚拟服务器组
ALIYUN::SLS::ApplyConfigToMachineGroup	给机器组指定 SLS 日志配置
ALIYUN::SLS::MachineGroup	创建所有需要通过 Logtail 客户端收集日志的 ECS 云机器组
ALIYUN::VPC::RouterInterface	创建路由器接口
ALIYUN::VPC::PeeringRouterInterfaceBinding	设置路由接口的对端信息
ALIYUN::VPC::PeeringRouterInterfaceConnection	从指定路由器接口发起链接
ALIYUN::ApiGateway::Group	创建 API 分组
ALIYUN::ApiGateway::StageConfig::SignatureBinding	创建 API 分组中测试、预发、线上环境变量
ALIYUN::ApiGateway::Api	创建 API
ALIYUN::ApiGateway::App	创建应用
ALIYUN::ApiGateway::Authorization	给 API 授权应用的访问权限
ALIYUN::ApiGateway::CustomDomain	给 API 分组创建绑定自定义域名
ALIYUN::ApiGateway::Deployment	发布 API 或快速切换 API 版
ALIYUN::ApiGateway::Signature	创建后端签名密钥
ALIYUN::ApiGateway::SignatureBinding	绑定 API 与后端签名密钥
ALIYUN::ApiGateway::TrafficControl	创建用户自定义的流控策略
ALIYUN::ApiGateway::TrafficControlBinding	给 API 绑定用户自定义流控
ALIYUN::ApiGateway::VpcAccessConfig	配置 VPC 授权以便专有网络的 API 能对外提供服务

资源栈管理

资源编排（ROS）通过模板创建一组阿里云资源。ROS 把这组资源定义为一个栈（stack）。ROS 通过 stack 管理，维护这组阿里云的资源。本文档介绍如何在资源编排控制台新建资源栈。

1. 登录 ROS 控制台。
2. 在 **资源栈管理** 页面，单击 **新建资源栈**。

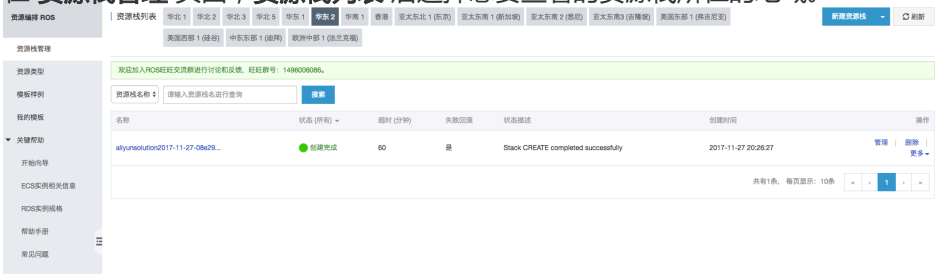
3. 选择地域和输入模板。可以选择模板源为直接输入或引用 URL。模板为 JSON 格式的文本文件，使用 UTF8 编码。如果您还没有模板，有关模板编辑，请参见 **模板结构说明**。您也可以选择 **使用可视化编辑器编辑模板**。如何使用可视化编辑器编辑模板，请参见 **可视化编辑器**。为方便您使用，简化使用步骤，资源编排控制台已经提供了一些常用的模板样例。您可以根据资模板样例快速创建资源站。请参见 **通过模板样例创建资源栈**。

4. 根据您的资源信息输入各个参数，然后单击 **创建**。**注意：** 不同类型的资源栈需要输入的参数不同。以下图片仅为参考示例。

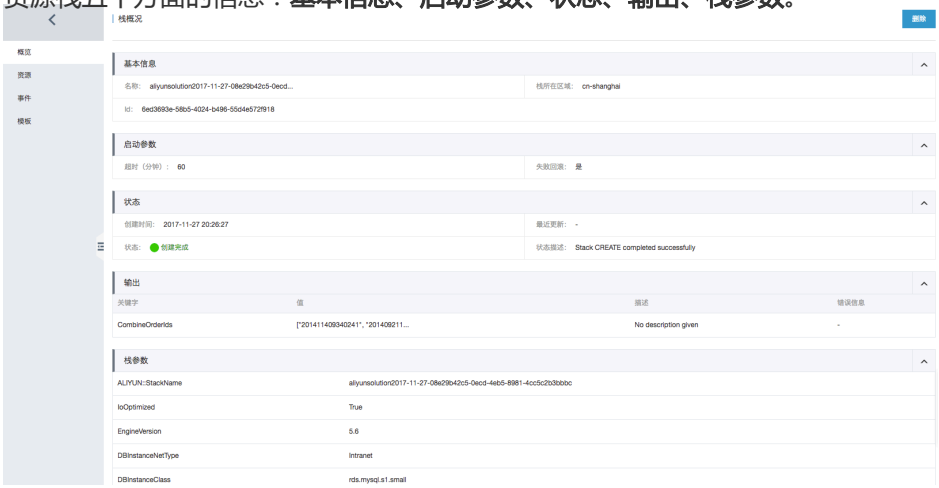
创建资源栈需要一定时长，请等待创建完成。

您可在 **资源栈管理** 页面，查看当前创建的资源栈状态和信息。

1. 登录 ROS 控制台。
2. 在 **资源栈管理** 页面，**资源栈列表** 后选择您要查看的资源栈所在的地域。



3. 单击资源栈名称或者右侧操作栏中的 **管理** 按钮，进入资源栈概览页。在资源栈概览页面，可查看到资源栈五个方面的信息：**基本信息、启动参数、状态、输出、栈参数。**



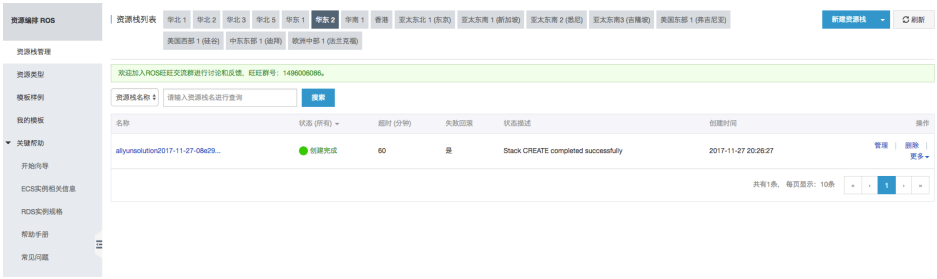
- **基本信息**：包括栈名称、栈 ID、和所在区域。
- **启动参数**：包括创建栈的超时时间，失败是否回滚。
- **状态**：显示堆栈当前的状态。
- **输出**：显示资源栈创建中，用户申明的需要输出的实例信息。
- **栈参数**：显示创建资源栈时，指定的参数，包括 ROS 提供的以 **ALIYUN::** 开始的内部参数。

4. 单击左侧导航栏中的 **资源、事件、模板**，查看资源栈的各方面的详情。
   - **资源**：展示资源栈中所包括的每一个资源的信息。单击 **详情**，可查看资源的详细信息。
   - **事件**： 显示资源栈生命周期中发生的每一个事件。在栈上的任何操作，都会被记录在这里，包括创建，健康检查，删除，更新，回滚等等。
   - **模板**：显示当前资源栈所对应的模板信息。

如果您不再使用某个资源栈，可将该资源栈删除。

资源栈删除方式有两种： **保留资源** 和 **释放资源**。请根据您的实际情况选择删除方式。

1. 登录 ROS 控制台。
2. 在 **资源栈管理** 页面，**资源栈列表** 后选择您要查看的资源栈所在的地域。



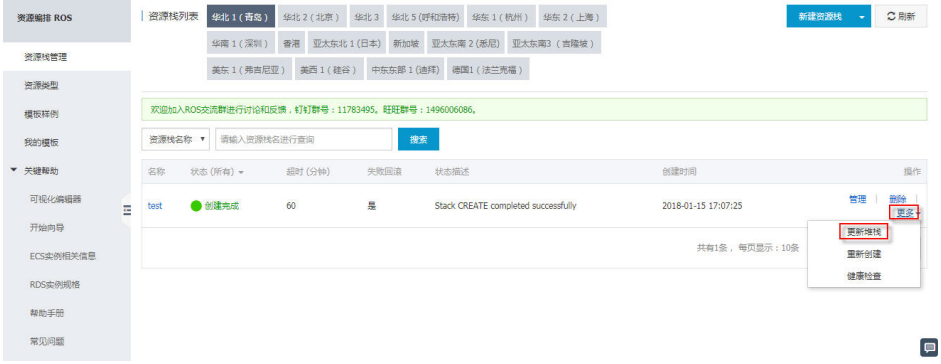
3. 找到要删除的资源栈，单击右侧操作栏中的 **删除** 按钮。
4. 在弹出对话框中选择删除方式：**保留资源** 和 **释放资源**，然后单击 **确定**。选择 **保留资源**，当前资源栈创建的资源将会被保留。选择 **释放资源**，当前资源栈创建的资源将会随资源栈的删除而释放。

如果您所拥有的阿里云资源有所变化，已有的资源栈已不能满足您的业务需求，您可以更新或重建当前的资源栈。

如果您需要修改当前资源栈的地域或模板信息，请 **更新资源栈**。如果您不需要修改当前资源栈模板，而只需要修改资源栈名、创建超时时长、和您的资源实例参数信息，请 **重新创建资源栈**。

更新资源栈

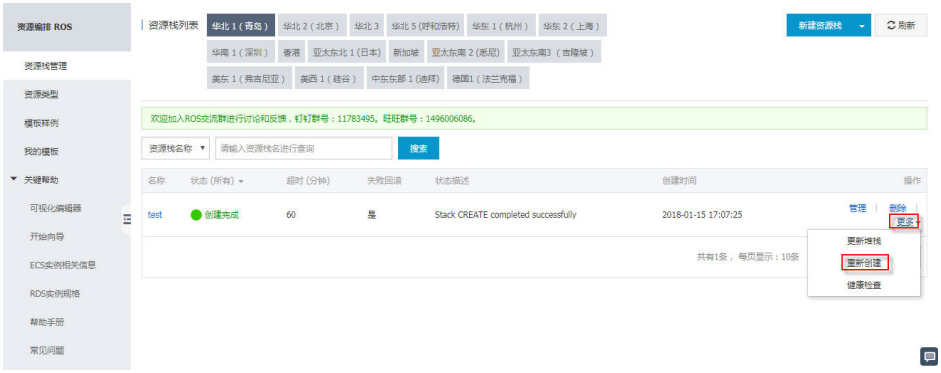
1. 登录 资源编排控制台。
2. 在 **资源栈管理** 页面，**资源栈列表** 后选择您要更新的资源栈所在的地域。
3. 找到您要更新的资源栈。在相应资源栈的操作栏中，单击 **更多**。然后再下拉框中，单击 **更新堆栈**。



4. 修改地域和（或）模板数据信息，然后单击 **下一步**。
5. 修改需要更新的资源实例信息，然后单击 **更新**。

重新创建资源栈

1. 登录 资源编排控制台。
2. 在 **资源栈管理** 页面，**资源栈列表** 后选择您要更新的资源栈所在的地域。
3. 找到您要更新的资源栈。在相应资源栈的操作栏中，单击 **更多**。然后再下拉框中，单击 **重新创建**。



4. 修改需要更新的信息，然后单击 **更新**。

模板语法

模板是一个 JSON 格式的文本文件，使用 UTF8 编码。模板用于创建资源栈，是描述基础设施和架构的蓝图，ROS 的开发者在模板中定义阿里云资源的生产和配置细节，并说明资源间的依赖关系。

ROS 模板结构如下：

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

"Description" : "模板描述信息，可用于说明模板的适用场景、架构说明等。",
"Metadata" : {
// 关于模板的元数据信息，比如存放用于可视化的布局信息。
},
"Parameters" : {
// 定义创建资源栈时模板用户可以定制化的参数。
},

"Mappings" : {
// 定义映射信息表，映射信息是一种多层的 Map 结构。
},

"Conditions": {
// 使用内部条件函数定义条件。这些条件确定何时创建关联的资源。
},

"Resources" : {
// 所需资源的详细定义，包括资源间的依赖关系、配置细节等。
},

"Outputs" : {
// 用于输出一些资源属性等有用信息，可以通过 API 或控制台获取输出的内容。
}
}
```

ROSTemplateFormatVersion（必需）

ROS 支持的模板版本号，当前版本号：2015-09-01。

Description（可选）

模板的描述信息，可用于说明模板的适用场景、架构说明等。通常情况下，对模板进行比较的描述有利于模板的用户理解模板的内容。

Metadata（可选）

模板编写者可以使用 Metadata 来存放与模板相关的元数据信息，内容可以是自由的 JSON 格式。

Parameters（可选）

定义创建资源栈时模板用户可以定制化的参数。比如很多情况下，模板的开发者会把 ECS 的规格设计成一个参数，使用模板创建资源栈时，可以根据实际的评估结果来选择合适的规格，参数支持默认值。使用参数可以增强模板的灵活性，提高复用性。

Mappings（可选）

Mappings 定义了一个多层的映射表，可以通过 Fn::FindInMap 函数来选择 key 对应的值。可用于根据不同的输入参数值作为 Key 来查找映射表。比如，可以根据 Region 不同，自动查找 Region- 镜像映射表，来找到适用的镜像。

Conditions（可选）

Conditions 使用 Fn::And，Fn::Or，Fn::Not，Fn::Equals 定义条件，多个条件之间用“，”隔开。在创建或更新堆栈时，先计算模板中的所有条件，然后再创建资源。会创建与 true 条件关联的所有资源，忽略与 false 条件关联的所有资源。

Resources（可选）

用于详细定义使用该模板创建的资源栈所包含的资源，包括资源间的依赖关系、配置细节等。

Outputs（可选）

用于输出一些资源属性等有用信息，可以通过 API 或控制台获取输出的内容。

参数可用于在资源栈创建时覆盖模板中的某些值，用来提高模板的灵活性和可复用性。

例如，现在有一个模板可以用来创建包含 1 个 负载均衡实例，2 个 ECS 实例，1 个 RDS 实例的 Web 应用。如果该 Web 应用负载较高，可以在创建时选择使用高配的 ECS 实例，否则可以在创建时选择使用低配的 ECS

实例。在这种情况下，可以定义如下的参数：

```
"Parameters" : {
"InstanceType" : {
"Type" : "String",
"AllowedValues":["ecs.t1.small","ecs.s1.medium", "ecs.m1.medium", "ecs.c1.large"],
"Default": "ecs.t1.small",
"Label": "ECS 规格类型",
"Description" : "请选择创建 ECS 示例的配置，默认为 ecs.t1.small，可选 ecs.t1.small, ecs.s1.medium,
ecs.m1.medium，ecs.c1.large。"
}
}
```

上面定义的 InstanceType 参数，允许在用户使用模板创建资源栈时重新赋值。如果用户不设置参数值则使用默认的 ecs.t1.small。在资源定义时，可以引用此参数：

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

语法

每个参数由参数名称和参数属性组成。

参数名称必须为字母数字，并且在同一个模板中不能与其它参数名称重复。可以用 Label 字段来定义友好的参数名，一般在把模板动态生成为 Web 表单时很有用。

参数属性列表：

属性	必需	描述
Type	是	**参数的数据类型。** ***String*** 字符串。如：”ecs.s1.medium”。 ***Number*** 整数或浮点数。如：3.14。 ***CommaDelimitedList*** 一组用逗号分隔的字符串或数字，可通过 Fn::Select 函数索引值。如：”80, foo, bar”。 ***Json*** 一个 Json 格式的字符串。如：{ “foo”：“bar” }。 ***Boolean*** 一个布尔值。如：true 或者 false。

Default	否	在创建资源栈时，如果用户没有传入指定值，编排服务会检查模板中是否有定义默认值，如果有定义默认值，则使用默认值，否则报错。
AllowedValues	否	包含参数允许值的列表。
AllowedPattern	否	一个正则表达式，用于检查用户输入的字符串类型的参数是否匹配，如果用户输入的不是字符串类型，则报错。
MaxLength	否	一个整数值，确定要允许 String 类型使用的字符的最大数目。
MinLength	否	一个整数值，确定要允许 String 类型使用的字符的最小数目。
MaxValue	否	一个数字值，确定要允许 Number 类型使用的最大数字值。
MinValue	否	一个数字值，确定要允许 Number 类型使用的最小数字值。
NoEcho	否	当调用查询堆栈时是否输出参数值。如果将值设置为 true，则只输出星号 (**)。
Description	否	用于描述参数的字符串。
ConstraintDescription	否	用于在违反该参数约束条件时说明该约束条件的字符串。
Label	否	参数别名，支持 UTF-8 字符，通过模板生成 Web 表单时可映射为 label。
AssociationProperty	否	用于自动验证该参数值的合法性并且给该参数提供可选值。 格式：” ROS资源类型:属性” ，其中不允许有空格。 例如：指定 ” ALIYUN::ECS::Instance:ImageId” ，则表示该参数将被资源类型为ALIYUN::ECS::Instance的属性ImageId引用。那么ROS的控制台，将会验证你给改参数指定的镜像Id是否可用，并以下拉框的方式列出其他可选值。 当前只支持针对镜像 Id 参数的验证。

示例

以下示例 Parameters 部分声明有两个参数。username 参数属于 String 类型，默认值为 anonymous。可指定的最小长度为 6，可指定的最大长度为 12，并且允许值为 anonymous、user-one、user-two。注意 username 的默认值也必须符合长度限制和允许值限制。password 参数属于 String 类型，无默认值。将 NoEcho 属性设置为 true 可阻止查询堆栈接口返回参数值。可指定的最小长度为 1，可指定的最大长度为

41。该模式允许小写和大写字母字符和数字。

```
"Parameters" : {
"username" : {
"Label": "用户名",
"Description" : "请输入用户名",
"Default": "anonymous",
"Type" : "String",
"MinLength" : "6",
"MaxLength" : "12",
"AllowedValues": ["anonymous", "user-one", "user-two"]
},
"password" : {
"Label": "密码",
"NoEcho" : "True",
"Description" : "请输入用户密码",
"Type" : "String",
"MinLength" : "1",
"MaxLength" : "41",
"AllowedPattern" : "[a-zA-Z0-9]*"
}
}
```

伪参数

伪参数是由 ROS 编排引擎提供的固定参数，可以和用户定义参数一样被引用，其值在编排运行时确定。目前支持的伪参数如下：

- ALIYUN::StackName - 当前资源栈的名称。
- ALIYUN::StackId - 当前资源栈的 ID。
- ALIYUN::Region - 当前资源栈所在的区域。
- ALIYUN::AccountId - 当前资源栈的 User ID。
- ALIYUN::NoValue - 创建或更新资源时删除资源的某属性。

描述堆栈中每一个资源的属性和依赖关系。一个资源可以被其他资源和 Outputs 所引用。

语法

资源部分由资源 ID 和资源描述组成。所有资源描述都被括在括号里。如果您声明多个资源，则可用逗号将它们分隔开。以下代码段描述了 Resources 的语法结构：

```
"Resources" : {
"资源1 ID" : {
"Type" : "资源类型",
"Condition": "是否创建次资源的条件",
"Properties" : {
资源属性描述
}
```

```
},
"资源 2 ID" : {
"Type" : "资源类型",
"Condition": "是否创建次资源的条件"，
"Properties" : {
资源属性描述
}
}
}
```

资源 ID

资源 ID 在模板中具有唯一性。可使用资源 ID 在模板的其他部分中引用资源。

资源类型

资源类型标识您正在声明的资源的类型。例如，ALIYUN::ECS::Instance 声明阿里云 ECS 实例。有关所有资源的列表，请参阅 资源列表。

资源属性

资源属性是可以为资源指定的附加选项。例如，对于每个阿里云 ECS 实例，必须为该实例指定一个 Image ID。如以下代码段所示：

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

如果资源不需要声明任何属性，那么您可以忽略该资源的属性部分。

属性值可以是文本字符串、字符串列表、布尔值、参数引用或者函数返回的值。如果属性值为文件字符串，该值会被双引号括起来。如果值为任一类型的列表结果，则它会被中括号 ([ ]) 括起来。如果值为内部函数或引用的结果，则它会被大括号 ({ }) 括起来。当您将文字、列表、参考和函数合并起来获取值时，上述规则适用。以下示例说明如何声明不同的属性值类型：

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

DeletionPolicy

利用 DeletionPolicy 属性，用户可以在某个资源的堆栈被删除时保留该资源。如以下代码段所示：

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

在上例中，如果模板对应的堆栈被删除，则会保留 ECSInstance 资源。

DependsOn

使用 DependsOn 属性可以指定特定资源紧跟着另一个资源创建。在您为资源添加 DependsOn 属性时，该资源仅在创建 DependsOn 属性中指定的资源之后创建。

如以下代码段所示，WebServer 将在 DatabaseServer 创建成功后才开始创建：

```
{
"ROSTemplateFormatVersion" : "2015-09-01",
"Resources" : {
"WebServer": {
"Type": "ALIYUN::ECS::Instance",
"DependsOn": "DatabseServer"
},
"DatabseServer": {
"Type": "ALIYUN::ECS::Instance",
"Properties": {
"ImageId" : "m-25l0rcfjo",
"InstanceType": "ecs.t1.small"
}
}
}
}
```

Condition

使用 Condition 属性可以指定是否需要正真创建此资源。当只有 Condition 所指定的条件值为 true 时才创建此资源。

如以下代码段所示，根据 MaxAmount 的值判断否创建 WebServer：

```
{
"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"}
}
},
"DatabseServer": {
"Type": "ALIYUN::ECS::Instance",
"Properties": {
"ImageId" : "m-25l0rcfjo",
"InstanceType": "ecs.t1.small"
}
}
}
}
```

示例

以下示例显示的是典型的资源声明。

```
"Resources" : {
"WebServer": {
"Type": "ALIYUN::ECS::Instance",
"Properties": {
"ImageId" : "m-25l0rcfjo",
"InstanceType": "ecs.t1.small",
"SecurityGroupId": "sg-25zwc3se0",
"ZoneId": "cn-beijing-b",
"Tags": [{
"Key": "Department1",
"Value": "HumanResource"
},{
"Key": "Department2",
"Value": "Finance"
}
]
}
},
"ScalingConfiguration": {
"Type": "ALIYUN::ESS::ScalingConfiguration",
"Properties": {
"ImageId": "ubuntu1404_64_20G_aliaegis_20150325.vhd",
```

```
"InstanceType": "ecs.t1.small",
"InstanceId": "i-25xhhcqbu",
"InternetChargeType": "PayByTraffic",
"InternetMaxBandwidthIn": 1,
"InternetMaxBandwidthOut": 20,
"SystemDisk_Category": "cloud",
"ScalingGroupId": "bwhtvpcBcKYac9fe3vd0kv7E",
"SecurityGroupId": "sg-25zwc3se0",
"DiskMappings": [
{
"Size": 10
},
{
"Category": "cloud",
"Size": 10
}
]
}
}
}
```

在输出部分定义在调用查询堆栈接口时返回的值。例如，用户可以定义 ECS 实例 ID 的输出，然后调用查询堆栈的接口查看该实例 ID。

语法

输出部分由输出 ID 和输出描述组成。所有输出描述都被括在括号里。如果您声明多个输出，则可用逗号将它们分隔开。以下代码段描述了输出部分的语法结构：

```
"Outputs" : {
"输出 1  ID" : {
"Description" : "输出的描述",
"Condition": "是否输出此资源属性的条件",
"Value" : "输出值得表达式"
},
"输出 2 ID" : {
"Description" : "输出的描述",
"Condition": "是否输出此资源属性的条件",
"Value" : "输出值得表达式"
}
}
```

输出 ID

此输出的标识符，在模板中具有唯一性。

Description（可选）

用于描述输出值的 String 类型。

Value（必需）

在调用查询堆栈接口时返回的属性值。

Condition（可选）

使用 Condition 属性可以指定是否需要正真创建此资源，输出资源的信息。当只有 Condition 所指定的条件值为 true 时才创建此资源或输出资源信息。

如以下代码段所示，根据 MaxAmount 的值判断是否创建 WebServer：

```
{
"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"]
}
}
}
}
```

示例

在以下示例中，输出部分有 2 个输出，第一个输出资源 ID 为 WebServer 的 InstanceId 属性，第二个输出资源 ID 为 WebServer 的 PublicIp 属性。

```
"Outputs": {
"InstanceId": {
"Value" : {"Fn::GetAtt": ["WebServer","InstanceId"]}
```

```
},
"PublicIp": {
"Value" : {"Fn::GetAtt": ["WebServer","PublicIp"]}
}
}
```

内部函数

编排服务提供多个内置函数帮助您管理您的堆栈。可以在资源和输出中使用内部函数。

Fn::Base64

内部函数 Fn::Base64 返回输入字符串的 Base64 编码结果。

声明

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

参数

stringToEncode

```
想转换成 Base64 的字符串。
```

返回值

用 Base64 表示方法的原始字符串。

示例

```
"Fn::Base64" : "string to encode"
```

Fn::FindInMap

内部函数 Fn::FindInMap 返回与 Mappings 部分声明的双层映射中的键对应的值。

声明

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

参数

MapName

Mappings 部分中所声明映射的 ID，包含键和值。

TopLevelKey

第一级键，其值是一个键/值对列表。

SecondLevelKey

第二级键，其值是一个字符串或者数字。

返回值

分配给 SecondLevelKey 的值。

示例

下面的示例显示了如何使用 Fn::FindInMap：

```
{
"ROSTemplateFormatVersion": "2015-09-01",
"Parameters": {
"regionParam": {
"Description": "选择创建 ECS 的区域",
"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": "key1",
"Value": "value1"
},{
```

```
"Key": "key2",
"Value": "value2"
}
]
}
}
}
}
```

在上面的示例中，在创建名为 WebServer 的资源时，需要指定 ImageId 属性。在 Mappings 部分描述了根据区域区分的 ImageId 映射。在 Parameters 部分描述了需要用户指定的区域。Fn::FindInMap 会根据用户指定的区域在 RegionMap 中查找对应的 ImageId 映射，然后在映射中找到 32 对应的 ImageId。

MapName 可按照个人意愿设置，在本例中为 “RegionMap” 。

TopLevelKey 设置为创建堆栈的地区，通过使用 { “Ref” : “regionParam” } 由用户确定。

SecondLevelKey 设置为所需的架构，在本例中为 “32” 。

支持的函数

您可以在 Fn::FindInMap 函数中使用以下函数：

Fn::FindInMap

Ref

Fn::GetAtt

内部函数 Fn::GetAtt 返回模板中的资源的属性值。

声明

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

参数

resourceID

```
目标资源的 ID。
```

attributeName

```
```

目标资源的属性名称。

返回值

属性值。

示例

此示例返回 Resource ID 为 MyEcsInstance 的 ImageId 属性。

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

Fn::Join

内部函数 Fn::Join 将一组值连接起来，用特定分隔符隔开。

声明

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

参数

delimiter

分隔符。分隔符可以为空，这样就将所有的值直接连接起来。

[“string1”， “string2”，…]

被连接起来的值列表。

返回值

被连接起来的字符串。

示例

“Fn::Join” : [“,” , [“a”， “b”， “c”]]

返回：” a,b,c” 。

支持的函数

Fn::Base64

Fn::GetAtt

Fn::Join

Fn::Select

Ref

Fn::Select

内部函数 Fn::Select 通过索引返回数据元列表中的单个数据元。

声明

数据元列表可以是一个数组：

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

数据元列表也可以是一个映射表：

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

参数

index

```
待检索数据元的索引。如果数据元列表是一个数组，则索引是 0 到 N-1 之间的某个值，其中 N 代表阵列中元素的数量。如果数据元列表是一个映射表，则索引是映射表中的某个键。
如果根据找不到索引对应的值，则返回空字符串。
```

返回值

选定的数据元。

示例

如果数据元列表是一个数组：

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

此示例返回：” grapes” 。

如果数据元列表是一个映射表：

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

此示例返回：” grapes” 。

如果数据元列表是一个 CommaDelimitedList：

```
"Parameters" : {
"userParam": {
"Type": "CommaDelimitedList",
"Default": "10.0.100.0/24, 10.0.101.0/24, 10.0.102.0/24"
}
}

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

支持的函数

对于 Fn::Select 索引值，您可以使用 Ref 函数。

对于对象的 Fn::Select 列表，您可以使用以下函数：

Fn::Base64

Fn::FindInMap

Fn::GetAtt

Fn::Join

Fn::Select

Ref

Ref

内部函数 Ref 返回指定参数或资源的值。

如果指定参数是 Resource ID，则返回资源的值。

否则认为指定参数是参数，将尝试返回参数的值。

声明

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

参数

logicalName

```
您想引用的资源或参数之逻辑名称。
```

返回值

资源的值或者参数的值。

示例

使用 Ref 函数指定 regionParam 作为 WebServer 的 RegionMap 的区域参数：

```
{
"ROSTemplateFormatVersion": "2015-09-01",
"Parameters": {
"regionParam": {
"Description": "选择创建 ECS 的区域",
"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": "tiantt",
"Value": "ros"
},{
"Key": "tiantt1",
"Value": "ros1"
```

```
}
]
}
}
}
}
```

支持的函数

不能在 Ref 函数中使用任何函数。必须指定作为资源逻辑 ID 的字符串。

Fn::GetAZs

内部函数 Fn::GetAZs 返回指定 Region 的可用区列表。

声明

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

参数

region

```
region ID。
```

返回值

指定 Region 下的可用区列表。

示例

此示例在指定的 Region 的第一个可用区内创建一个 ECS 实例。

```
{
"ROSTemplateFormatVersion" : "2015-09-01",
"Resources" : {
"WebServer": {
"Type": "ALIYUN::ECS::Instance",
"Properties": {
"ImageId" : "centos7u2_64_40G_cloudinit_20160728.raw",
"InstanceType": "ecs.n1.tiny",
"SecurityGroupId": "sg-2zedcm7ep5quses05fs4",
"Password": "Ros12345",
"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"]}
}
}
}
```

支持的函数

Fn::Base64

Fn::FindInMap

Fn::GetAtt

Fn::Join

Fn::Select

Ref

Fn::Replace

内部函数 Fn::Replace 将字符串中的指定子字符串用新字符串替换。

声明

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

参数

{“object_key”：“object_value”}

```
object_key 将要被替换的字符串。
object_value 将要替换成的最终字符串。
```

object_string

将把 object_string 字符串中的所有 object_key 子字符串替换成 object_value 字符串。

返回值

被替换后的字符串。

示例

UserData 所指定的脚本中的 print 将被替换成 echo。

```
{
"ROSTemplateFormatVersion" : "2015-09-01",
"Resources" : {
"WebServer": {
"Type": "ALIYUN::ECS::Instance",
"Properties": {
"ImageId" : "centos_7_2_64_40G_base_20170222.vhd",
"InstanceType": "ecs.n1.medium",
"SecurityGroupId": "sg-94q49gota",
"Password": "MytestPassword1234",
"IoOptimized": "optimized",
"VSwitchId": "vsw-94vdvonyi",
"VpcId": "vpc-949uzr8c9",
"SystemDiskCategory": "cloud_ssd",
"UserData": {"Fn::Replace": [{"print": "echo"},
{"Fn::Join": ["\\n", [
"#!/bin/sh",
"mkdir ~/test_ros",
"print hello > ~/1.txt"
]]}]}
}
}
},
"Outputs": {
"InstanceId": {
"Value" : {"Fn::GetAtt": ["WebServer","InstanceId"]}
},
"PublicIp": {
"Value" : {"Fn::GetAtt": ["WebServer","PublicIp"]}
}
}
}
```

支持的函数

Fn::Base64

Fn::GetAtt

Fn::Join

Fn::Select

Ref

Fn::Split

内部函数 Fn::Split 通过指定分隔符对字符串进行切片，返回所有切片组成的列表。

声明

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

参数

delim

```
分隔符, 例如: ',', ';', '\n', '\t' 等等。
```

original_string

```
将要被切片的字符串
```

返回值

返回切片后所有字符串组成的列表。

示例

如果数据元列表是一个数组：

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

此示例返回：[“foo”， “ bar”， “achoo “]。

下面模板中使用 Fn::Split 对 InstanceIds 进行切片 ：

```
"Parameters" : {
"InstanceIds": {
"Type": "String",
```

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

支持的函数

Fn::Base64

Fn::FindInMap

Fn::GetAtt

Fn::Join

Fn::Select

Fn::Replace

Fn::GetAZs

Fn::If

Ref

Fn::Equals

比较两个值是否相等。如果两个值相等，则返回 true；如果不等，则返回 false。

声明

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

参数

value

要比较的任意类型的值。

返回值

true 或 false。

示例

在 Conditions 中使用 Fn::Equals 定义条件。

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

支持的函数

Fn::Or

Fn::Not

Fn::Equals

Fn::FindInMap

Fn::And

Ref

Fn::And

如果所有指定条件计算为 true，则返回 true；如果任意条件计算为 false，则返回 false。这代表 AND 运算符。最少可以包含两个条件。

声明

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

参数

condition

```
计算为 true 或 false 的条件。
```

返回值

true 或 false。

示例

在 Conditions 中使用 Fn::And 定义一个条件。

```
{
"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"}]}]}
}
}
```

支持的函数

Fn::Or

Fn::Not

Fn::Equals

Fn::FindInMap

Fn::And

Ref

Fn::Or

如果任意一个指定条件计算为 true，则返回 true；如果所有条件都计算为 false，则返回 false。这代表 OR 运算符。最少可以包含两个条件。

声明

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

参数

condition

```
计算为 true 或 false 的条件。
```

返回值

true 或 false。

示例

在 Conditions 中使用 Fn::Or 定义一个条件。

```
{
"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"}]}]}
}
}
```

支持的函数

Fn::Or

Fn::Not

Fn::Equals

Fn::FindInMap

Fn::And

Ref

Fn::Not

对计算为 false 的条件返回 true；对计算为 true 的条件返回 false；这代表 NOT 运算符。

声明

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

参数

condition

```
计算为 true 或 false 的条件。
```

返回值

true 或 false。

示例

在 Conditions 中使用 Fn::Not 定义一个条件。

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

支持的函数

Fn::Or

Fn::Not

Fn::Equals

Fn::FindInMap

Fn::And

Ref

Fn::If

如果指定的条件计算为 true，则返回一个值；如果指定的条件计算为 false，则返回另一个值。在模板 Resources 和 Outputs 属性值中支持 Fn::If 内部函数。您可以使用 ALIYUN::NoValue 伪参数作为返回值来删除相应的属性。

声明

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

参数

condition_name

Conditions 中条件对应的条件名称。通过条件名称引用条件。

value_if_true

当指定的条件计算为 true 时，返回此值。

value_if_false

当指定的条件计算为 false 时，返回此值。

示例

根据输入的参数，确定是否创建数据盘。

```
{
```

```
"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-2zew9pxh2yirtzqxdboi1",
"SystemDiskCategory":"cloud_efficiency",
"SecurityGroupId":"sg-2zece6wcqriejf1v91sr",
"SystemDiskSize":40,
"ImageId":"centos_6_8_64_40G_base_20170222.vhd",
"IoOptimized":"optimized",
"VSwitchId":"vsw-2zed9txvy7h2srqo6jmgq",
"InstanceType":"ecs.n1.medium"
}
}
},
"Outputs":{
"InstanceId":{
"Value":{
"Fn::GetAtt":[
```

```
"WebServer",
"InstanceId"
]
}
},
"ZoneId":{
"Value":{
"Fn::GetAtt":[
"WebServer",
"ZoneId"
]
}
}
}
}
```

支持的函数

Fn::Or

Fn::Not

Fn::Equals

Fn::FindInMap

Fn::And

Ref

Fn::ListMerge

合并多个列表为一个列表。

声明

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

参数

[“list_1_item_1”, “list_1_imte_2”, ...]

```
将要合并的第一个列表
```

[“list_2_item_1”, “list_2_imte_2”, ...]

将要和第一个列表合并的列表

示例

把两个 ECS 组挂载到同一个负载均衡实例。

```
{
"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-2ze9uqi7wo61hwep5q52",
"InstanceType": "ecs.t1.small",
"SecurityGroupId": "sg-2ze8yxgempcdsq3iucsi",
"MaxAmount": 1,
"MinAmount": 1
}
},
"BackendServer2": {
"Type": "ALIYUN::ECS::InstanceGroup",
"Properties": {
"ImageId" : "m-2ze9uqi7wo61hwep5q52",
"InstanceType": "ecs.t1.small",
"SecurityGroupId": "sg-2ze8yxgempcdsq3iucsi",
"MaxAmount": 1,
"MinAmount": 1
}
},
"Attachment": {
"Type": "ALIYUN::SLB::BackendServerAttachment",
"Properties": {
"LoadBalancerId": {"Ref": "LoadBalancer"},
"BackendServerList": { "Fn::ListMerge": [
{"Fn::GetAtt": ["BackendServer1", "InstanceIds"]},
{"Fn::GetAtt": ["BackendServer2", "InstanceIds"]}
]
}
}
}
}
}
```

支持的函数

Fn::Base64

Fn::GetAtt

Fn::Join

Fn::Select

Ref

Fn::Join

Fn::If

Fn::GetJsonValue

解析 JSON 字符串，获取指定的 Key 在第一层所对应的值。

声明

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

参数

key

```
键值
```

json_string

```
指定的需要解析的 JSON 字符串
```

示例

WebServer2 从 WebServer 实例执行完 UserData 返回的 JSON 字符串中，获取对应的值。

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

```
"Resources" : {
"WebServer": {
"Type": "ALIYUN::ECS::Instance",
"Properties": {
"ImageId" : "m-2ze45uwova5fedlufpz7",
"InstanceType": "ecs.n1.medium",
"SecurityGroupId": "sg-2ze7pxymaix640qrg3vu",
"Password": "Wenqiao1234",
"IoOptimized": "optimized",
"VSwitchId": "vsw-2zei67xd9nhcqxzec7qt7",
"VpcId": "vpc-2zevx9ios1rszqv0azijb",
"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": "192.168.2.110",
"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-2ze45uwova5fedlufpz7",
"InstanceType": "ecs.n1.medium",
"SecurityGroupId": "sg-2ze7pxymaix640qrg3vu",
"Password": "Wenqiao1234",
"IoOptimized": "optimized",
"VSwitchId": "vsw-2zei67xd9nhcqxzec7qt7",
"VpcId": "vpc-2zevx9ios1rszqv0azijb",
"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": "192.168.2.111",
```

```
"HostName": "userdata-2"
}
},
},
"Outputs": {
"InstanceId": {
"Value" : {"Fn::GetAtt": ["WebServer","InstanceId"]}
},
"PublicIp": {
"Value" : {"Fn::GetAtt": ["WebServer","PublicIp"]}
}
}
}
```

支持的函数

Fn::Base64

Fn::GetAtt

Fn::Join

Fn::Select

Ref

Fn::Join

Fn::If

Fn::MergeMapToList

将多个 Mapping 合并成一个以 Mapping 为元素的列表。

声明

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

参数

{ “key_1” : [“key_1_item_1”， “key_1_item_2”，...]}

将要合并的第一个 Mapping，"key_1" 所对应的值必须是一个列表。"key_1" 将是合并后的列表元素中每个 Mapping 的一个键，它对应的值在第一个 Mapping 中将是 "key_1_item_1"，在第二个 Mapping 中是 "key_1_item_2"， 第三第四 ... 以此类推。最终合并的列表长度是，所有将要合并的参数 Mapping 中，"key_x" 所对应的列表中最长的长度。如果有的 "key_y" 所对应的列表长度比较短，那么就会重复此列表的最后一个元素，使列表长度都达到最长。

{“key_2”:[“key_2_item_1”，“key_2_item_2”，...]}

将要合并的第二个 Mapping，"key_2" 所对应的值必须是一个列表。"key_2" 将是合并后的列表元素中每个 Mapping 的一个键，它对应的值在第一个 Mapping 中将是 "key_2_item_1"，在第二个 Mapping 中是 "key_2_item_2"， 第三第四 ... 以此类推。

示例

合并三个 Mapping， 每个 Mapping 中键值对应的列表长度一致：

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

最终的合并结果是：

```
[
{
"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"
}
]
```

合并三个 Mapping，每个 Mapping 中键值对应的列表长度不一致：

```
{
"Fn::MeregMapToList": [
{"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"]}
]
}
```

最终的合并结果是：

```
[
{
"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"
}
]
```

下面的模板把 WebServer 中创建的所有实例，都加入到一个负载均衡的虚拟服务器组中。

```
{
"ROSTemplateFormatVersion": "2015-09-01",
"Resources": {
"WebServer": {
"Type": "ALIYUN::ECS::InstanceGroupClone",
"Properties": {
"SourceInstanceId": "i-xxxxx",
"Password": "Hello1234",
"MinAmount": 1,
"MaxAmount": 1
}
},
"CreateVServerGroup": {
"Type": "ALIYUN::SLB::VServerGroup",
"Properties": {
"LoadBalancerId": "lb-yyyy",
"VServerGroupName": "VServerGroup-test",
"BackendServers": {
"Fn::MergeMapToList": [
{"Port": [6666, 9090, 8080]},
{"ServerId": {"Fn::GetAtt": ["WebServer", "InstanceIds"]}},
{"Weight": [20, 100]}
]
}
}
}
}
}
```

支持的函数

Fn::Base64

Fn::GetAtt

Fn::Join

Fn::Select

Ref

Fn::Join

Fn::If

Fn::ListMerge

Fn::GetJsonValue

映像（Mappings）

映像是一个 Key-Value 映射表。在模板的 Resources 和 Outputs 部分可以使用 Fn::FindInMap 内部函数，通过给出 Key 获取映射表的 Value。

语法

映像部分由 Key-Value 对组成。其中 Key 和 Value 可以为字符串类型或者数字类型。如果用户声明多个映射，则可用逗号将它们分隔开，每个映射的名称不能重复。

示例

以下示例正确的映像。

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

以下示例错误的映像。

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

以下示例 Fn::FindInMap。

```
{
"ROSTemplateFormatVersion": "2015-09-01",
"Parameters": {
"regionParam": {
"Description": "选择创建ECS的区域",
"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"
}
]
}
}
}
}
```

条件（Conditions）

每一个条件项由 Fn::And、Fn::Or、Fn::Not、Fn::Equals 定义。根据您在创建或更新堆栈时指定的输入参数值进行计算。在每个条件中都可以引用其他条件、参数值或映射。在模板的 Resources 和 Outputs 部分将条件与资源和资源属性关联起来。条件和 Resource 的关联通过两种方式：Fn::If 函数或者 Resource 资源的 Condition 字段。

语法

每个条件由条件名和条件本身对组成。其中条件名是字符串类型。条件是由 Fn::And、Fn::Or、Fn::Not、Fn::Equals 定义，在本条件中还可以引用其他条件。多个条件用逗号隔开，每个条件名不能重复。

示例

以下示例 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"]}
}
```

以下示例 Conditions 和 Resources 如何关联。本例中会根据用户的 EnvType 参数值决定是否给 ECS instance 创建数据盘，创建 OSS bucket。

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

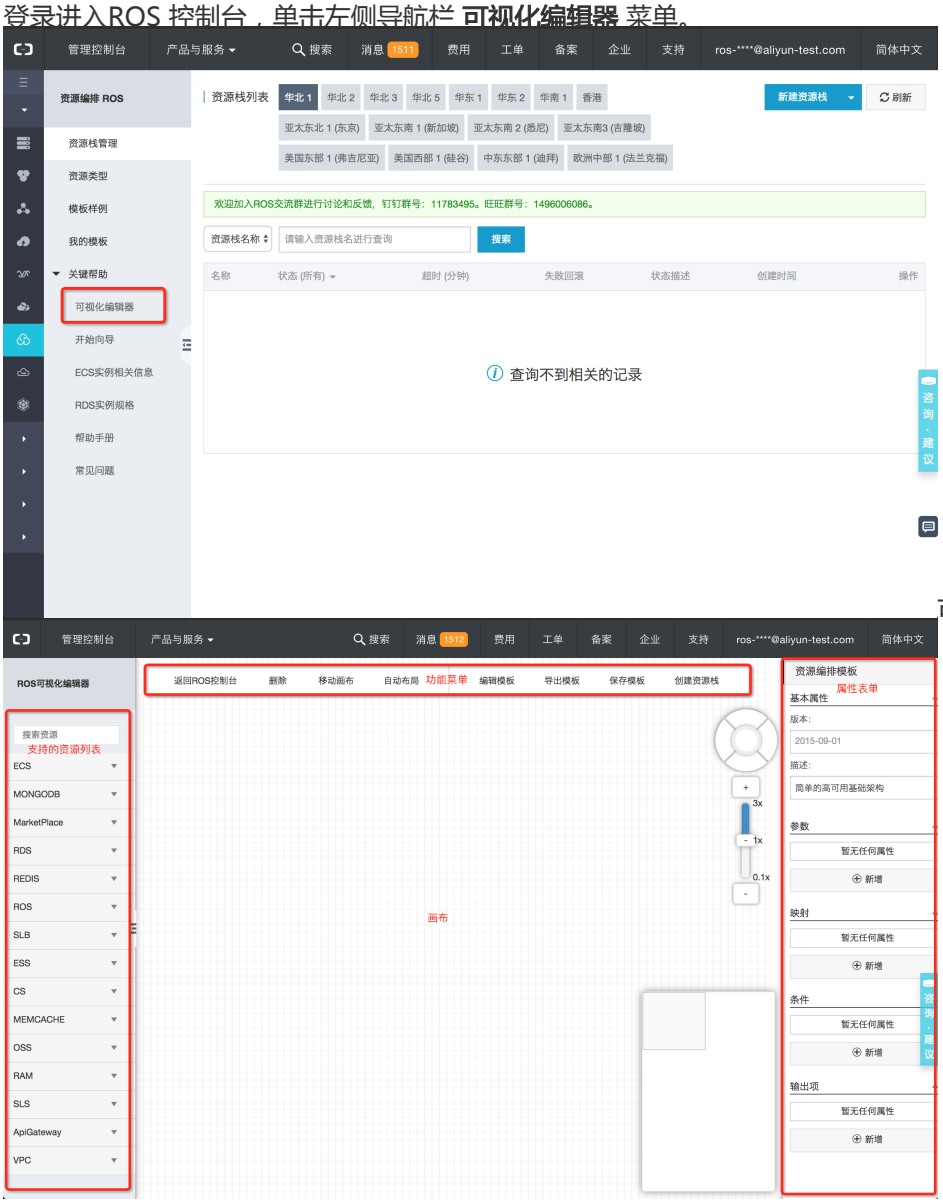
```
"DiskMappings":{
"Fn::If":[
"CreateProdRes",
[
{
"Category":"cloud_efficiency",
"DiskName":"FirstDataDiskName",
"Size":40
},
{
"Category":"cloud_ssd",
"DiskName":"SecondDataDiskName",
"Size":40
}
],
{
"Ref":"ALIYUN::NoValue"
}
]
},
"VpcId":"vpc-2zew9pxh2yirtzqxdboi1",
"SystemDiskCategory":"cloud_efficiency",
"SecurityGroupId":"sg-2zece6wcqriejf1v91sr",
"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"
]
}
```

```
}
}
}
```

ROS 可视化编辑器（ROS Visual Editor）是资源编排服务控制台提供的一个可以降低开发者手工编辑 ROS 模板难度，提高开发者使用效率的可视化工具。可视化编辑器使用可视化的方式，通过拖拽连线的方式建立资源之间的关系，将云资源的编排过程直观、简洁地呈现出来。可以通过便捷的表单来定制云资源，摆脱了以往编写模板时复杂的语法规则、易错的文本格式，以提高用户体验。同时，可视化编辑器可以将已有的模板和资源栈以可视化的方式展示，理清资源之间的依赖关系，看到整个 IT 资源的全貌。

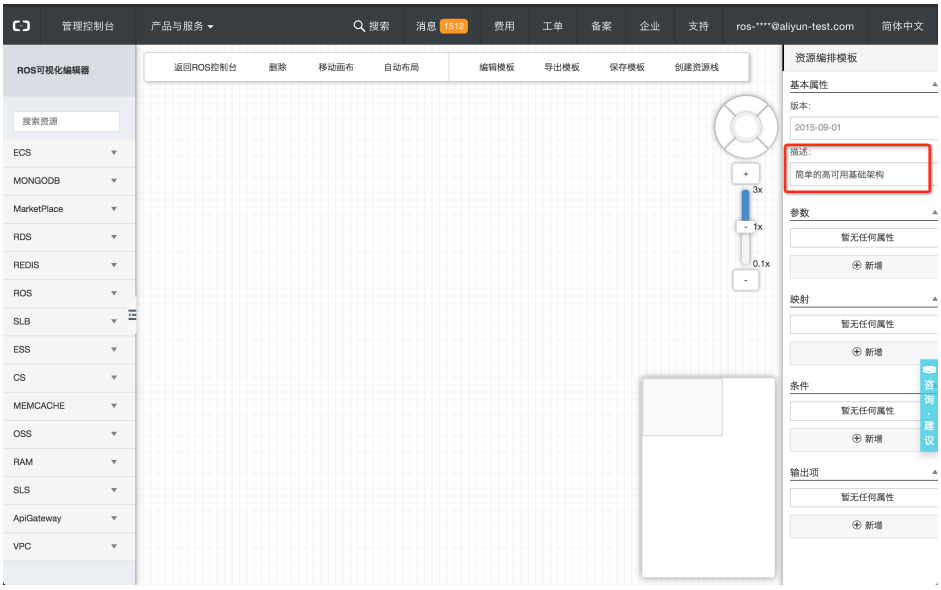
本文介绍如何使用可视化编辑器来创建一个高可用场景模板：负载均衡实例下挂载 1 个 VPC，1 个 VSwitch，1 个安全组，1 个 ECS 组（可以包含多个 ECS 实例）；负载均衡监听 443 和 80 端口。

1. 登录进入ROS 控制台，单击左侧导航栏 **可视化编辑器** 菜单。

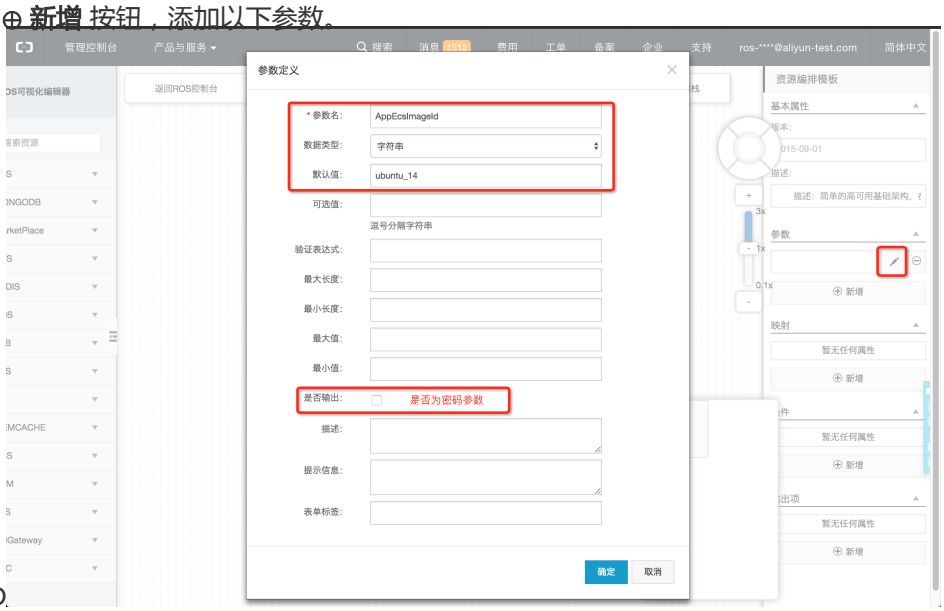


可视化编辑器页面

2. 填写模板描述。输入模板描述，说明模板的功能或用途。

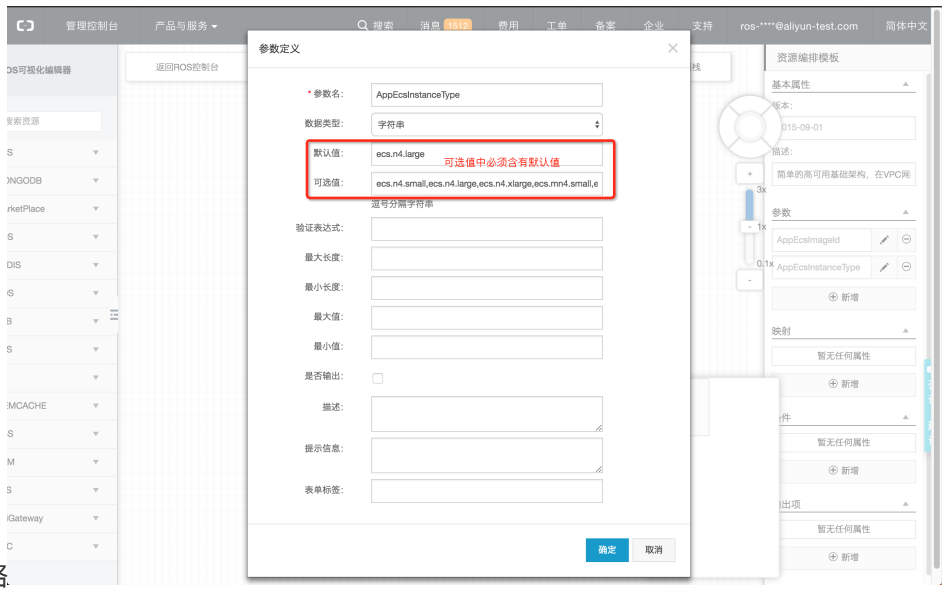


3. 添加参数。单击 ⊕ **新增** 按钮，添加以下参数。



- 镜像 ID

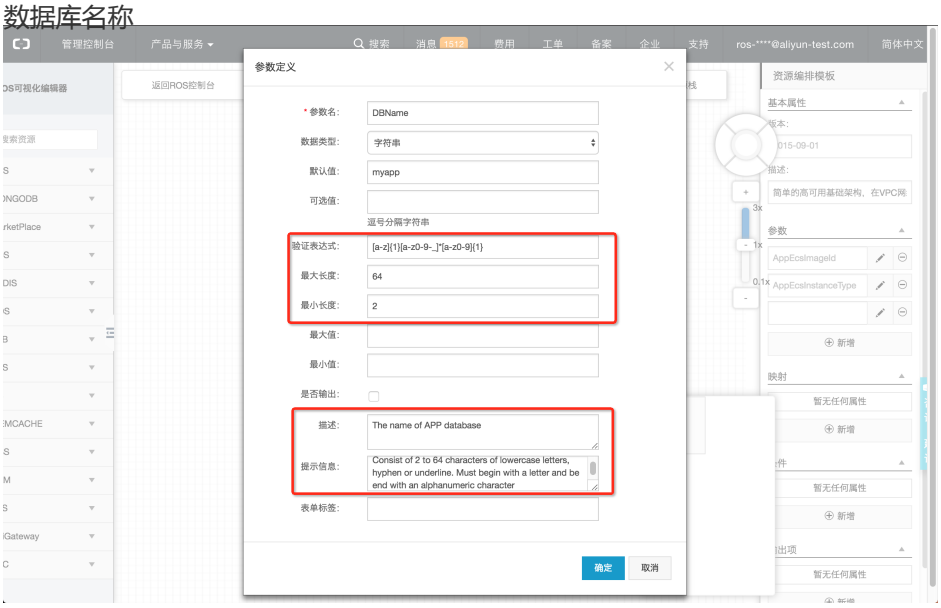
  - 参数名：AppEcsImageId
  - 参数类型：String
  - 参数默认值：ubuntu_14



- 实例规格

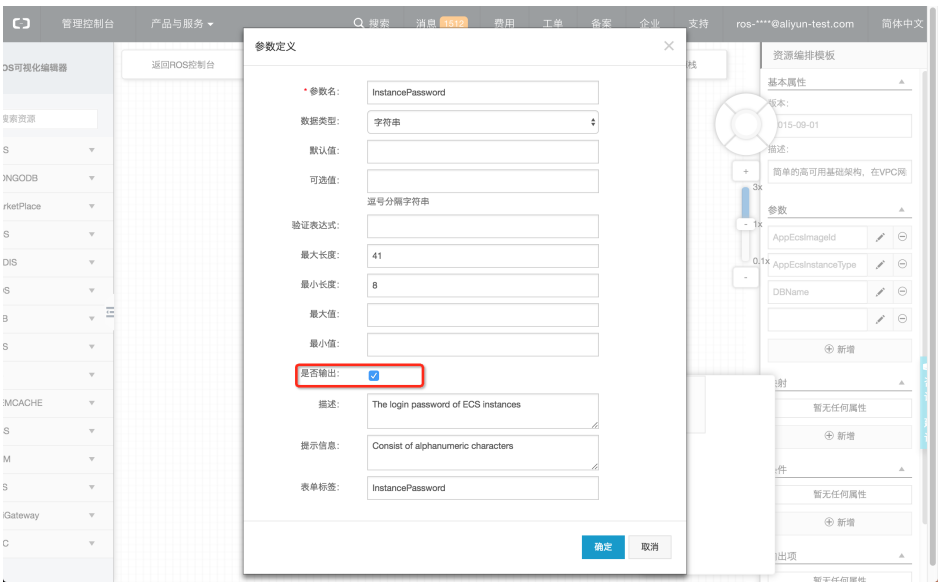
  - 参数名：AppEcsInstanceType
  - 指定默认值和可选值。

- 数据库名称



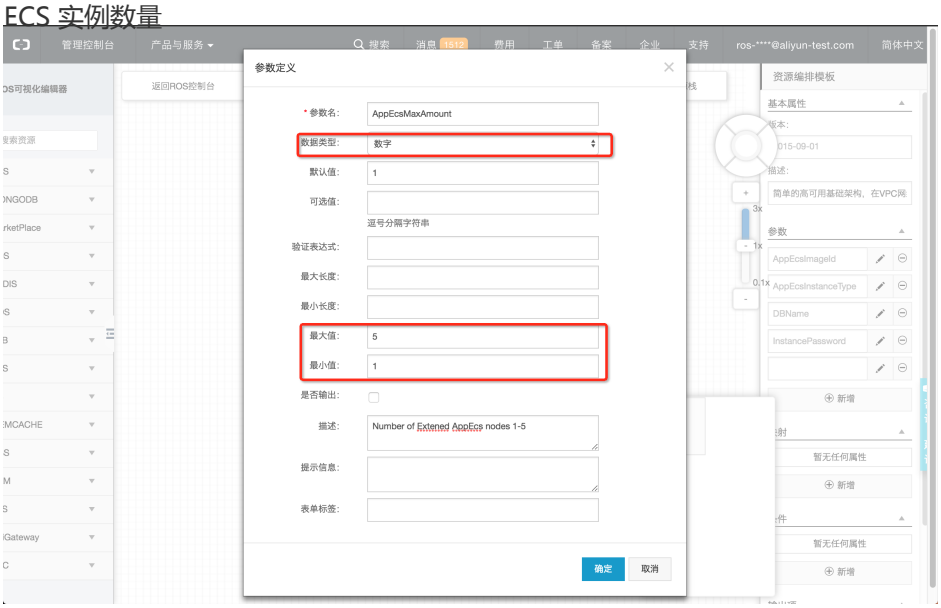
  - 验证表达式：[a-z]{1}[a-z0-9-_]*[a-z0-9]{1}（由小写字母数组下划线组成，必须以字母开头结尾，以字母数字开头结尾。）
  - 最大长度：64
  - 最小长度：2
  - 描述：The name of APP database
  - 提示信息：Consist of 2 to 64 characters of lowercase letters, hyphen or underline. Must begin with a letter and be end with an alphanumeric character

- ECS 实例密码



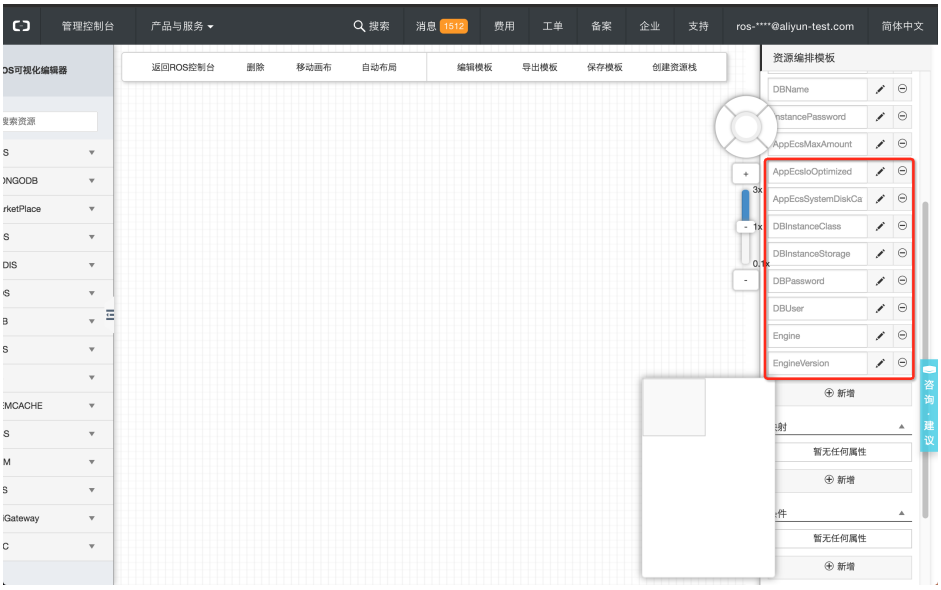
- 是否输出：Enable（NoEcho=true）
- 表单标签：InstancePassword

- ECS 实例数量



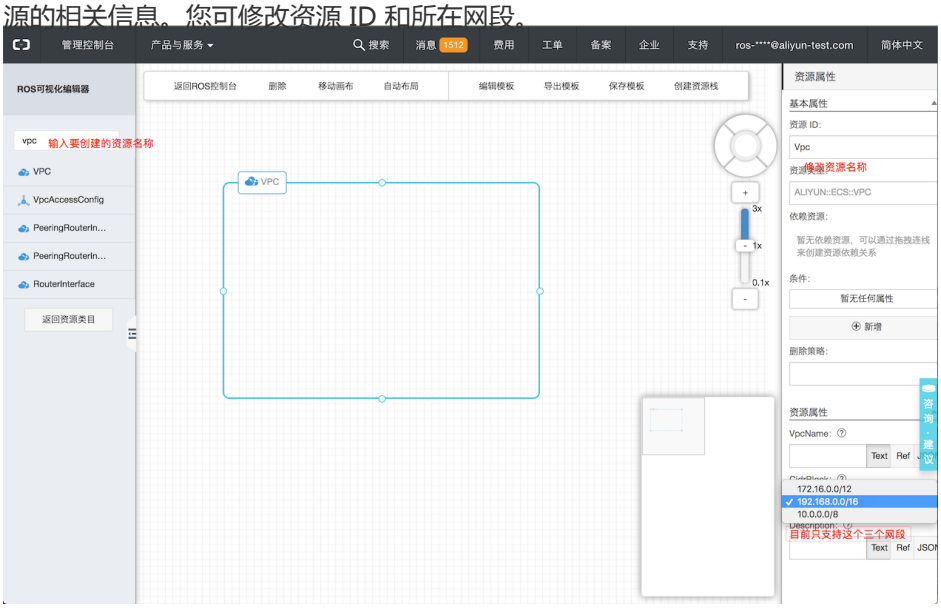
- 数据类型：Number
- 最大值：5
- 最小值：1

- 数据库信息
  - RDS 实例类型
  - RDS 容量
  - 数据库密码
  - RDS 用户名

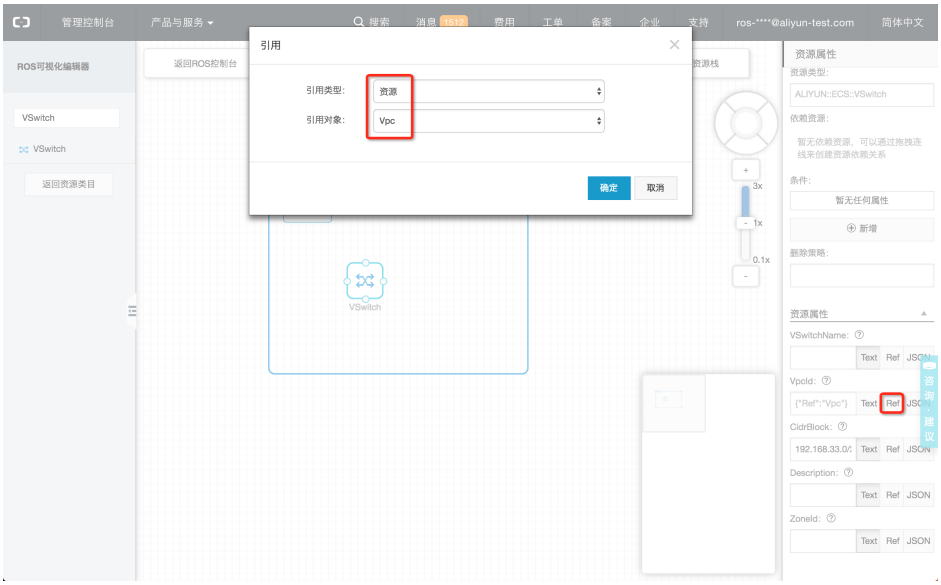


4. 创建资源。从左侧的资源列表搜索栏中，搜索资源名称，然后拖拽相应资源到画布里，再创建资源。

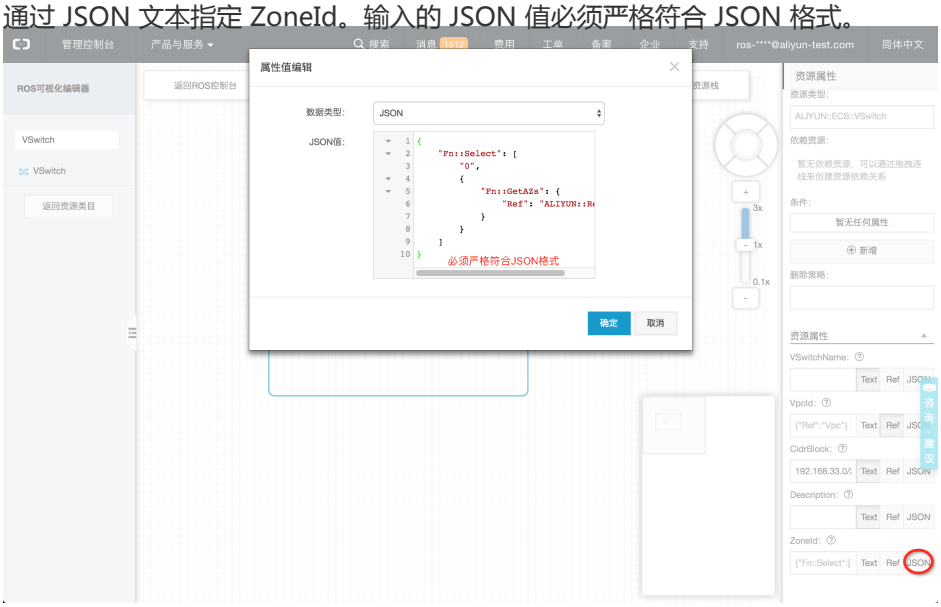
- 创建 VPC。将 VPC 拖到画布里。在画布里，单击 VPC，右侧 **资源属性** 下会显示 VPC 资源的相关信息。您可修改资源 ID 和所在网段。



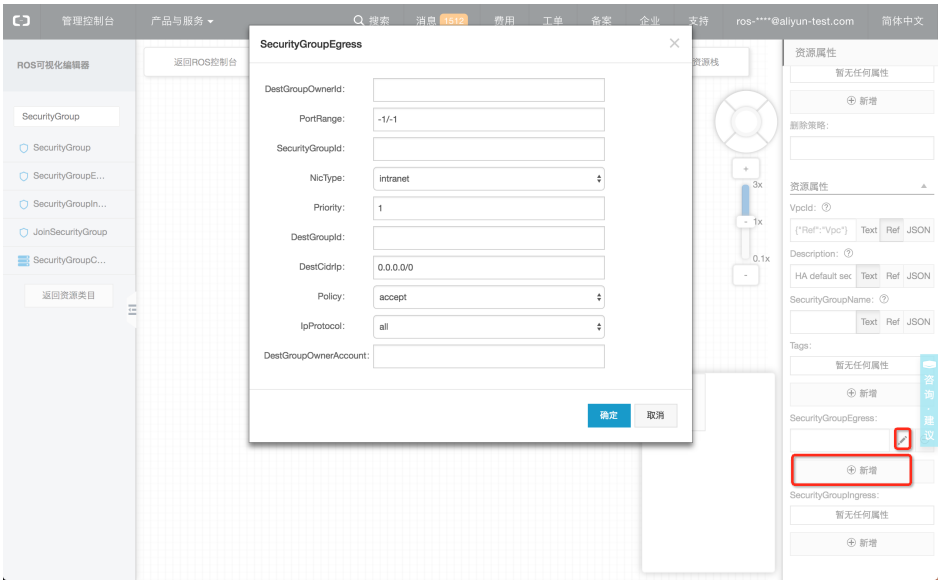
- 创建 VSwitch。由于 VSwitch 依赖于 VPC 存在，所以从左侧的资源列表里搜索到 VSwitch 后，必须将其直接拖拽到 VPC 里面。在画布里，单击 VSwitch 图标，右侧显示其资源属性信息。
    - 通过 Ref 函数指定 VpcId。



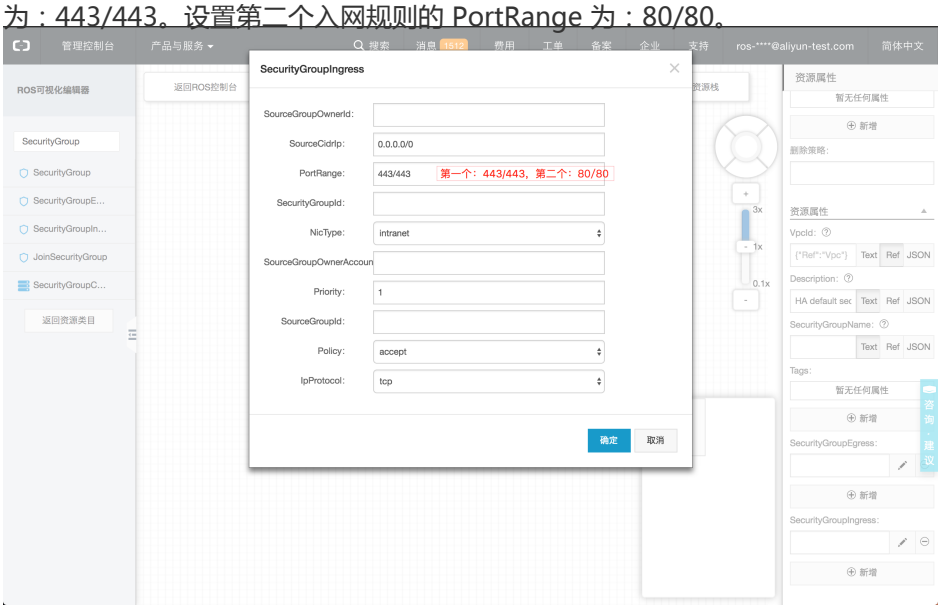
- 通过 JSON 文本指定 ZoneId。输入的 JSON 值必须严格符合 JSON 格式。



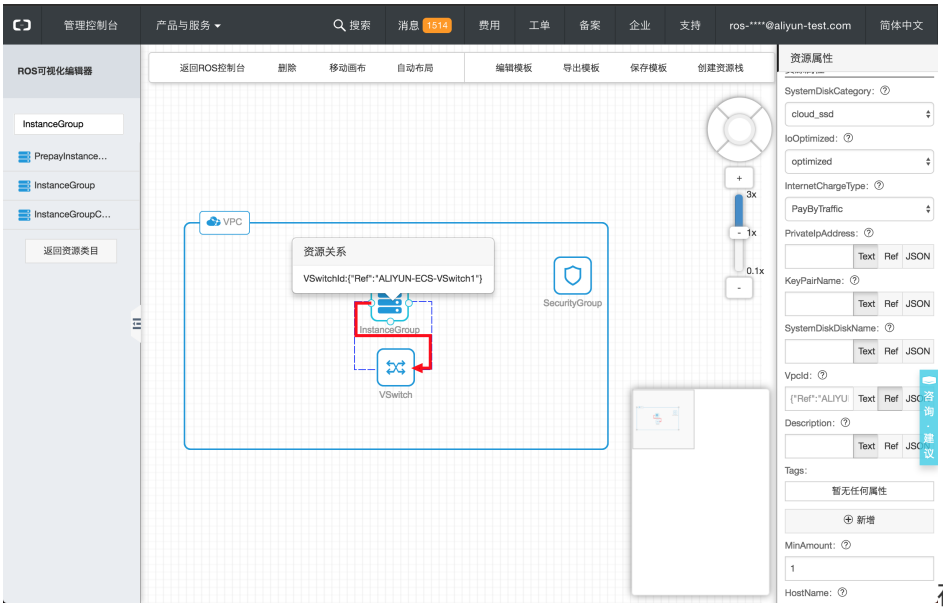
- 创建 SecurityGroup。由于 SecurityGroup 依赖于 VPC 存在，所以从左侧的资源列表里搜索到 SecurityGroup 后，必须将其直接拖拽到 VPC 里面。在画布里，单击 SecurityGroup 图标，右侧显示其资源属性信息。
    - 创建一个出网规则 SecurityGroupEgress。



- 创建两个入网规则 SecurityGroupIngress。设置第一个入网规则的 PortRange 为：443/443。设置第二个入网规则的 PortRange 为：80/80。



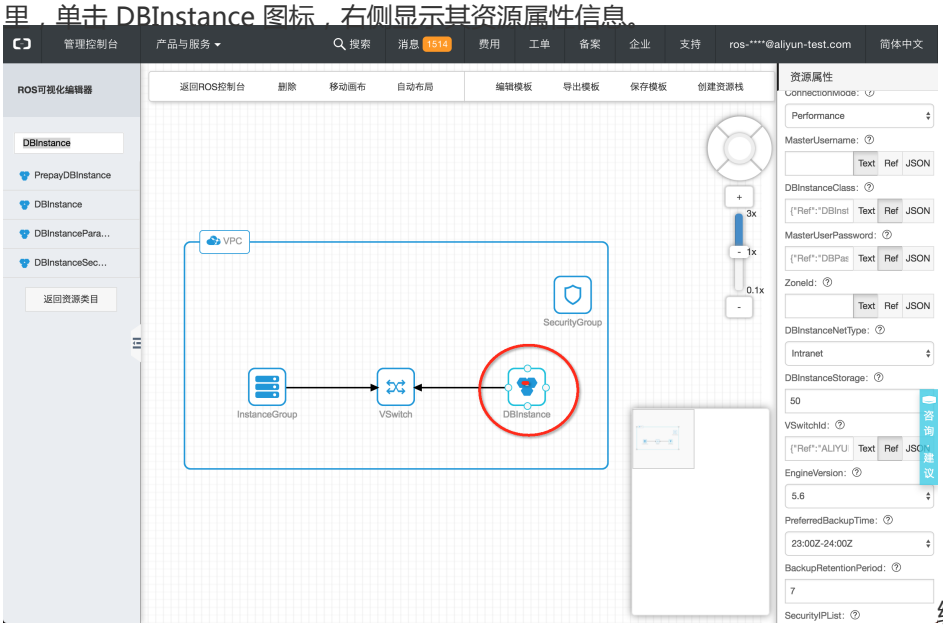
- 创建 InstanceGroup。从左侧的资源列表里搜索 InstanceGroup 后，拖拽到 VPC 里面。在画布里，单击 InstanceGroup 图标，右侧显示其资源属性信息。



在页面右侧编辑资源属性：

- SystemDiskCategory：cloud_ss
- IoOptimized：optimized
- InternetChargeType：PayByTraffic
- ImageId：通过 Ref 按钮引用之前设置的参数 AppEcsImageId。
- Password：通过 Ref 按钮引用之前设置的参数 InstancePassword。
- InstanceType：通过 Ref 按钮引用之前设置的参数 AppEcsInstanceType。
- NetworkType：VPC

- 创建 DBInstance。从左侧的资源列表里搜索 DBInstance 后，拖拽到 VPC 里面。在画布里，单击 DBInstance 图标，右侧显示其资源属性信息。

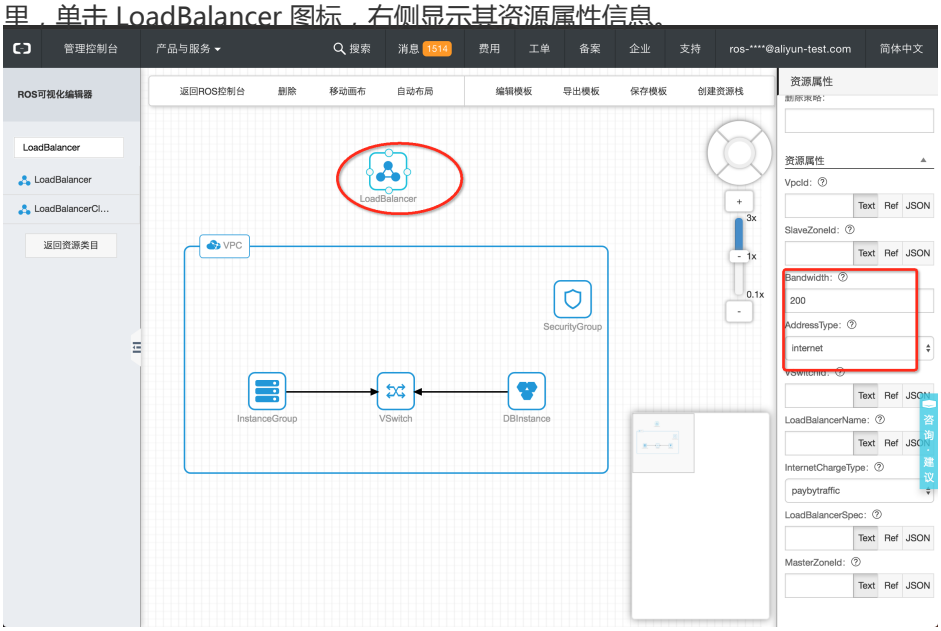


编辑资源属性：

- Engine：MySQL
- MultiZA：false
- DBMappings：新增编辑 DBName=myapp，CharacterSetName=utf8。

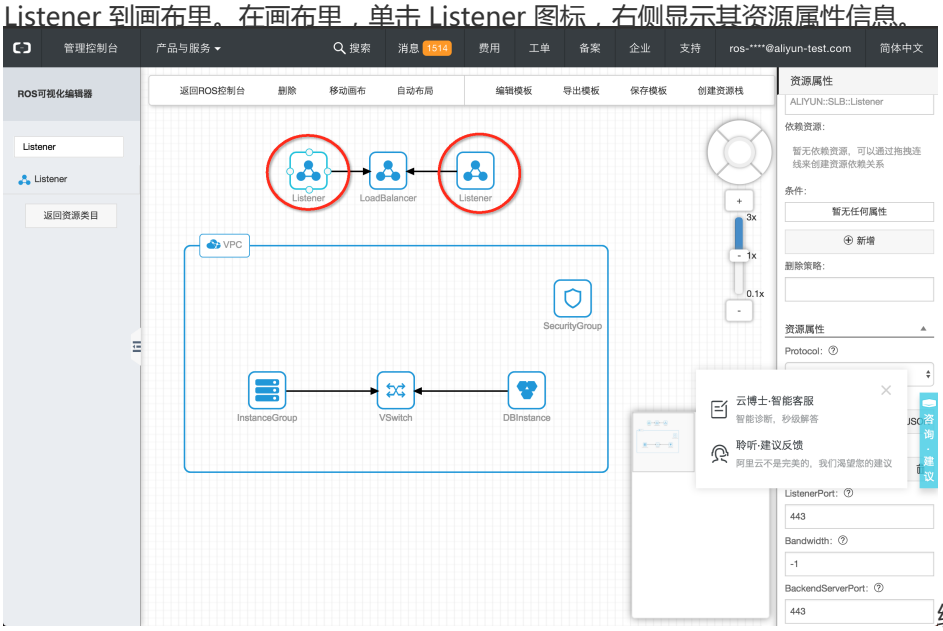
- DBInstanceClass：通过 Ref 按钮引用之前设置的参数 DBInstanceClass。
- MasterUserPassword：通过 Ref 按钮引用之前设置的参数 DBPassword。
- DBInstanceNetType：intranet
- DBInstanceStorage：50
- EngineVersion：5.6
- PreferredBackupTime：23:00Z-24:00Z。
- PreferredBackupPeriod：Monday, Wednesday

- 创建 LoadBalancer。从左侧的资源列表里搜索 LoadBalancer 后，拖拽到画布里。在画布里，单击 LoadBalancer 图标，右侧显示其资源属性信息。



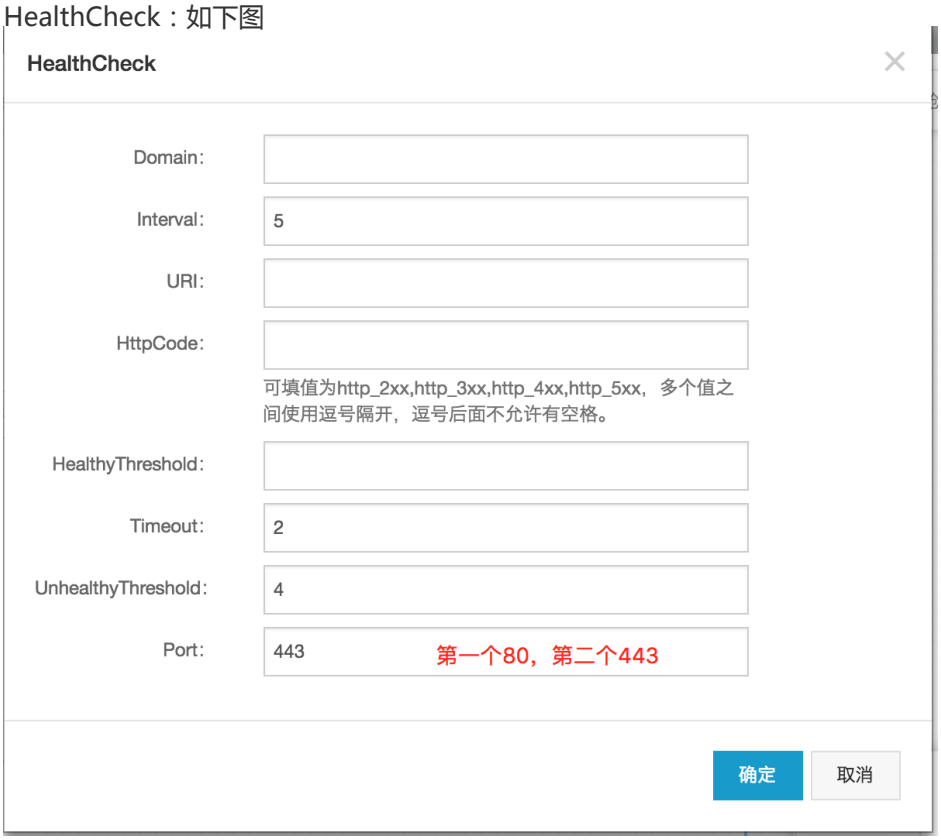
- Bandwidth：200
- AddressType：internet

- 添加两个负载均衡监听 Listener。从左侧的资源列表里搜索 Listener 后，拖拽两个 Listener 到画布里。在画布里，单击 Listener 图标，右侧显示其资源属性信息。

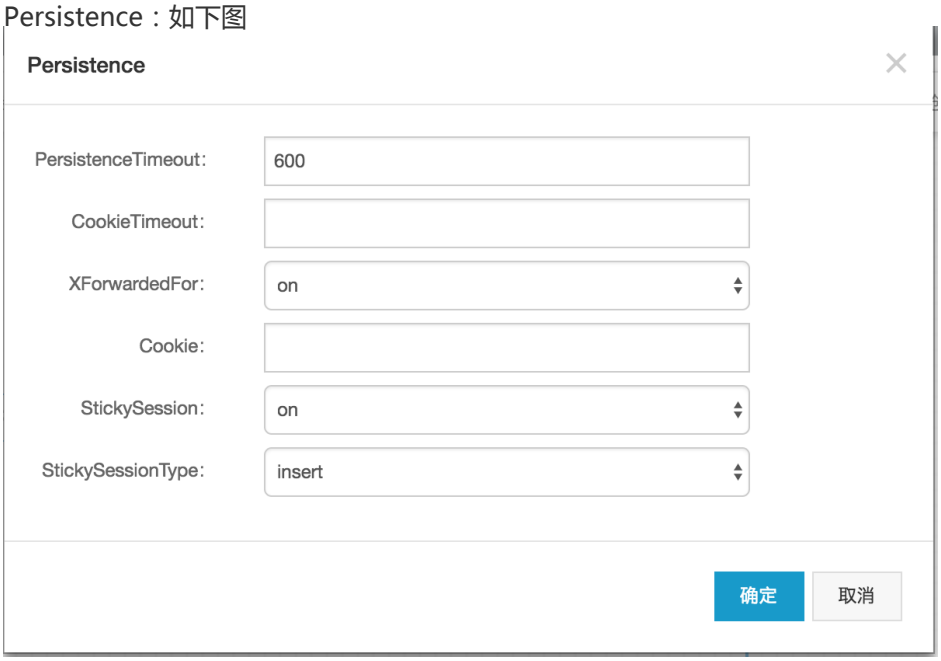


编辑资源属性：

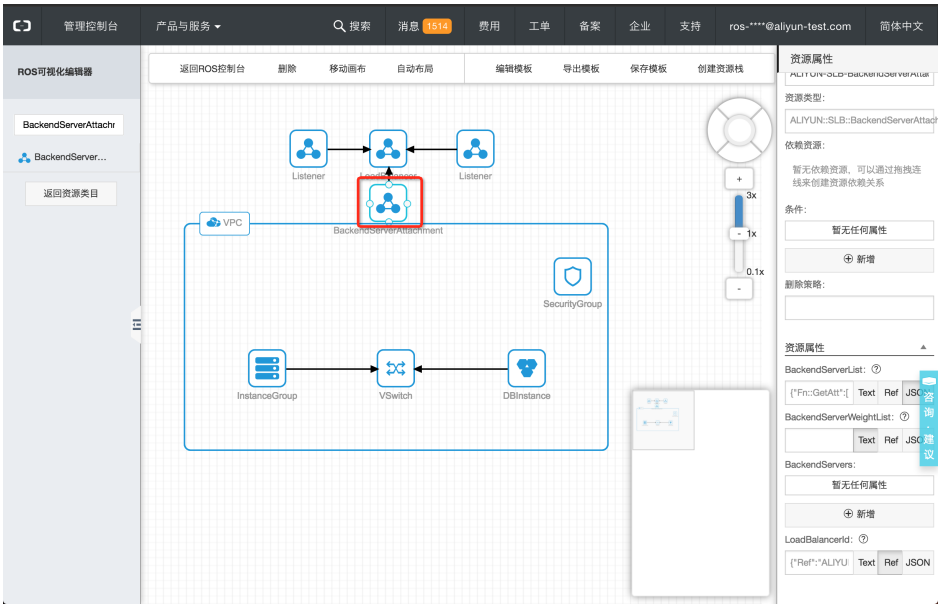
- Protocol：tcp
- BackendServerPort：80 或 443（第一个 80，第二个 443）
- ListenerPort：80 或 443（第一个 80，第二个 443）
- Bandwidth：-1
- HealthCheck：如下图



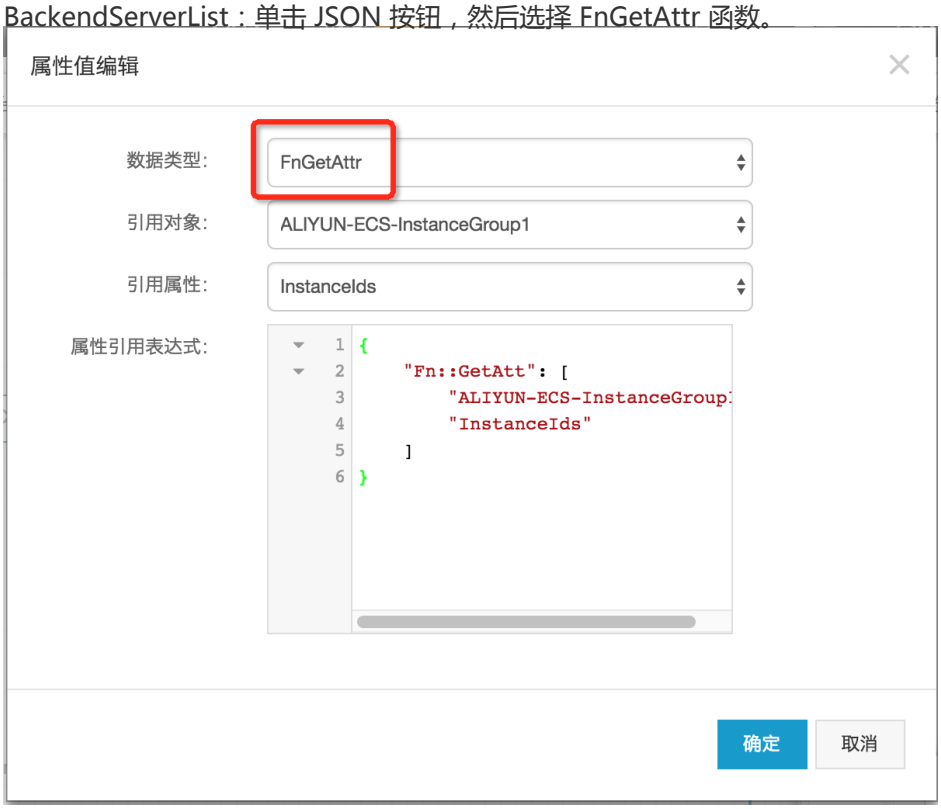
- Persistence：如下图



- 创建 BackendServerAttachment。从左侧的资源列表里搜索 BackendServerAttachment，拖拽到画布里。在画布里，单击 BackendServerAttachment 图标，右侧显示其资源属性信息。



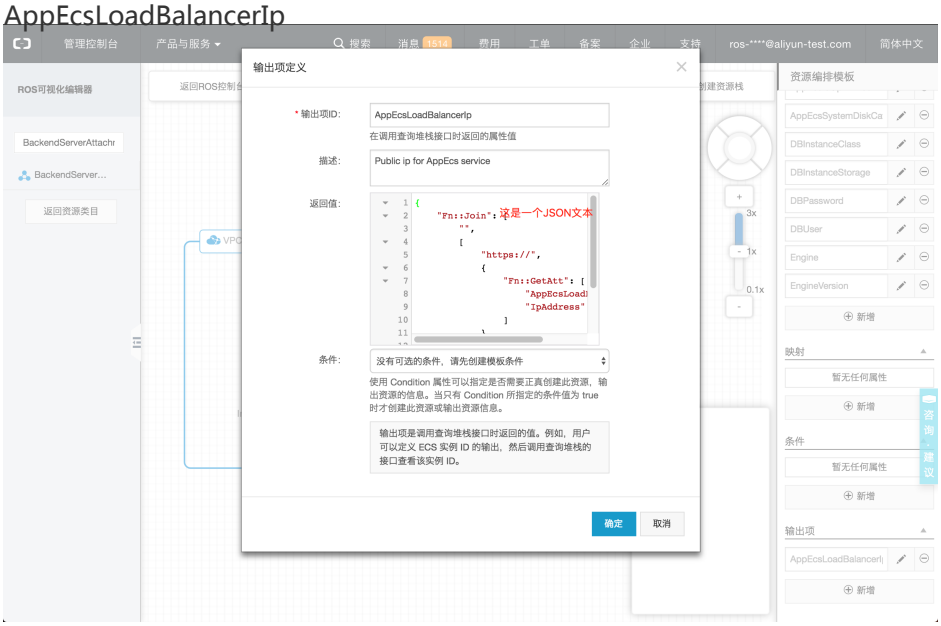
- BackendServerList：单击 JSON 按钮，然后选择 FnGetAttr 函数。



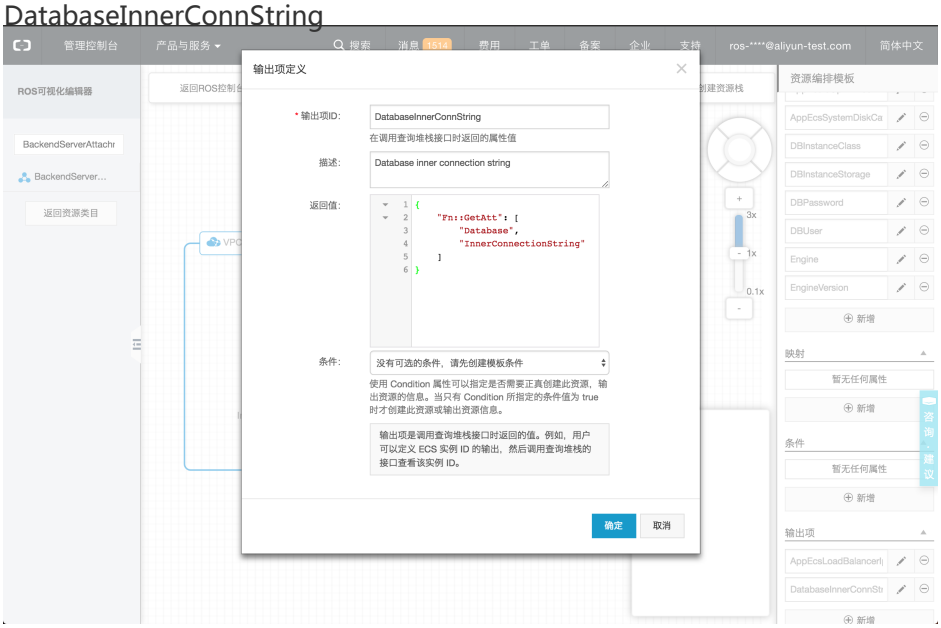
- LoadBalancerId：单击 Ref 按钮，引用资源 ALIYUN-SLB-LoadBalancer1。

5. 添加输出项。单击画布空白处，右侧显示 **资源编排模板**。单击 **输出项** 下的**新增** 按钮，然后单击编辑图标，逐条添加以下输出项。

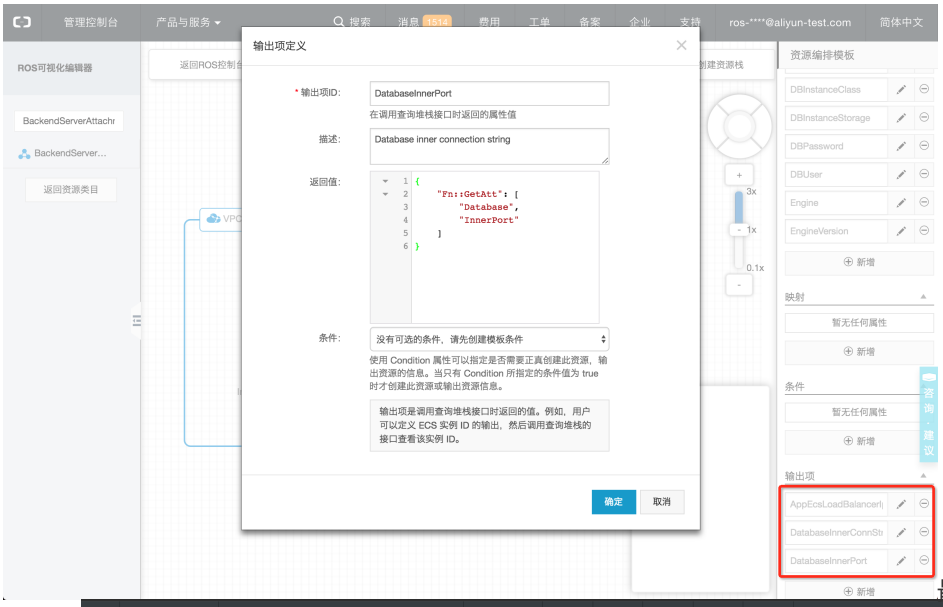
- AppEcsLoadBalancerIp



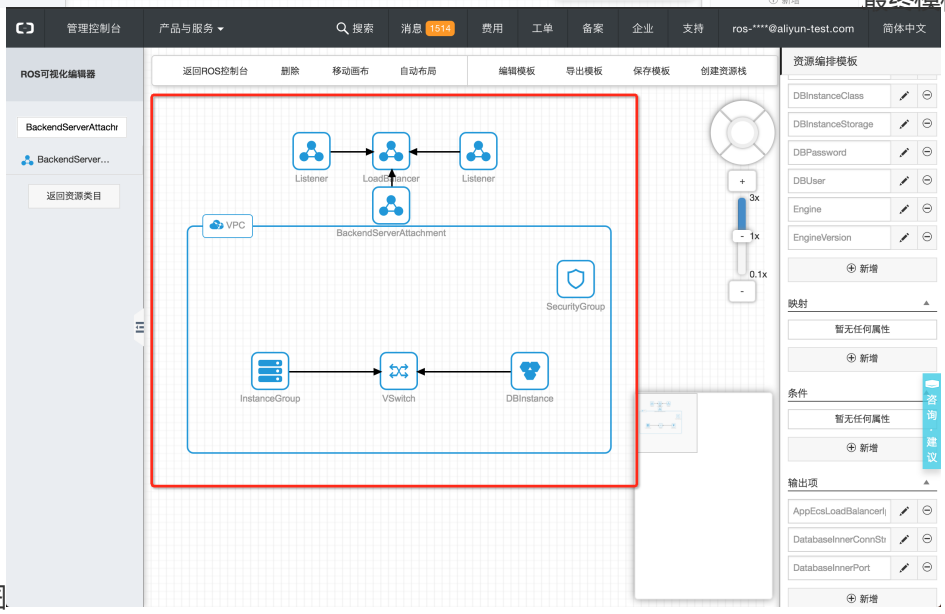
- DatabaseInnerConnString



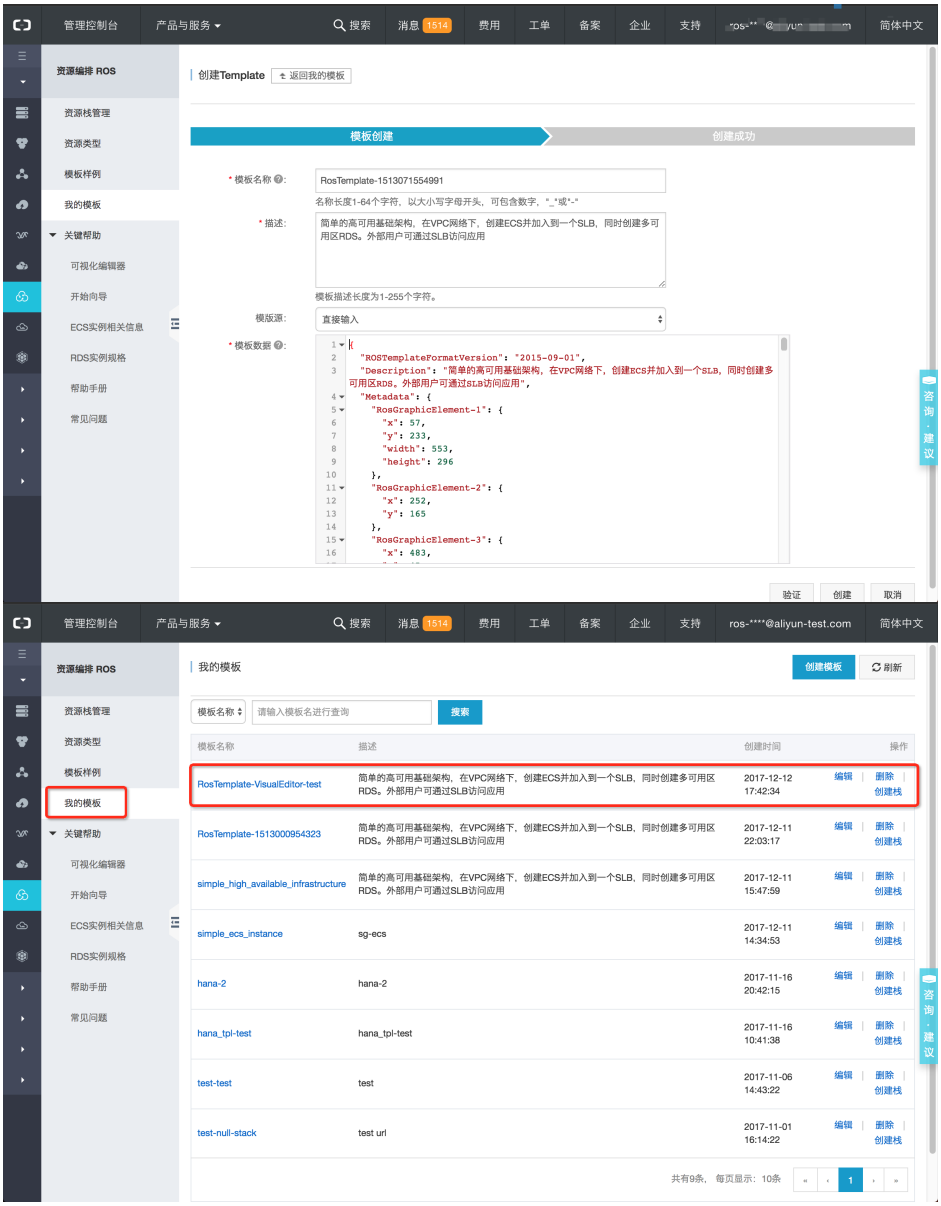
- DatabaseInnerPort



最终模板架构图



6. 保存模板。单击上方功能菜单中的 **保存模板**按钮，页面跳转至 **我的模板** 的 **模板创建** 页面。编辑模板名称，单击 **创建**。


管理控制台
产品与服务
搜索
消息 1514
费用
工单
备案
企业
支持
简体中文
资源编排 ROS
资源栈管理
资源类型
模板样例
我的模板
关键帮助
可视化编辑器
开始向导
ECS实例相关信息
RDS实例规格
帮助手册
常见问题
创建Template
返回我的模板
模板创建
创建成功
模板名称:
RosTemplate-1513071554991
名称长度1-64个字符，以大小写字母开头，可包含数字，"_"或"-"
描述:
简单的高可用基础架构，在VPC网络下，创建ECS并加入到一个SLB，同时创建多可用区RDS。外部用户可通过SLB访问应用
模板描述长度为1-255个字符。
模版源:
直接输入
模板数据:
{
"ROSTemplateFormatVersion": "2015-09-01",
"Description": "简单的高可用基础架构，在VPC网络下，创建ECS并加入到一个SLB，同时创建多可用区RDS。外部用户可通过SLB访问应用",
"Metadata": {
"RosGraphicElement-1": {
"x": 57,
"y": 233,
"width": 553,
"height": 296
},
"RosGraphicElement-2": {
"x": 252,
"y": 165
},
"RosGraphicElement-3": {
"x": 483,
验证
创建
取消
咨询·建议
管理控制台
产品与服务
搜索
消息 1514
费用
工单
备案
企业
支持
ros-****@aliyun-test.com
简体中文
资源编排 ROS
资源栈管理
资源类型
模板样例
我的模板
关键帮助
可视化编辑器
开始向导
ECS实例相关信息
RDS实例规格
帮助手册
常见问题
我的模板
创建模板
刷新
模板名称
请输入模板名进行查询
搜索
模板名称
描述
创建时间
操作
RosTemplate-VisualEditor-test
简单的高可用基础架构，在VPC网络下，创建ECS并加入到一个SLB，同时创建多可用区RDS。外部用户可通过SLB访问应用
2017-12-12 17:42:34
编辑
删除
创建栈
RosTemplate-1513000954323
简单的高可用基础架构，在VPC网络下，创建ECS并加入到一个SLB，同时创建多可用区RDS。外部用户可通过SLB访问应用
2017-12-11 22:03:17
编辑
删除
创建栈
simple_high_available_infrastructure
简单的高可用基础架构，在VPC网络下，创建ECS并加入到一个SLB，同时创建多可用区RDS。外部用户可通过SLB访问应用
2017-12-11 15:47:59
编辑
删除
创建栈
simple_ecs_instance
sg-ecs
2017-12-11 14:34:53
编辑
删除
创建栈
hana-2
hana-2
2017-11-16 20:42:15
编辑
删除
创建栈
hana_tpl-test
hana_tpl-test
2017-11-16 10:41:38
编辑
删除
创建栈
test-test
test
2017-11-06 14:43:22
编辑
删除
创建栈
test-null-stack
test url
2017-11-01 16:14:22
编辑
删除
创建栈
共有9条，每页显示：10条
1
咨询·建议